THE UNIVERSITY OF CHICAGO

ON SEMI-SUPERVISED KERNEL METHODS

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY
VIKAS SINDHWANI

CHICAGO, ILLINOIS
DECEMBER 2007

To my Parents.

ABSTRACT

Semi-supervised learning is an emerging computational paradigm for learning from limited supervision by utilizing large amounts of inexpensive, unsupervised observations. Not only does this paradigm carry appeal as a model for natural learning, but it also has an increasing practical need in most if not all applications of machine learning – those where abundant amounts of data can be cheaply and automatically collected but manual labeling for the purposes of training learning algorithms is often slow, expensive, and error-prone.

In this thesis, we develop families of algorithms for semi-supervised inference. These algorithms are based on intuitions about the natural structure and geometry of probability distributions that underlie typical datasets for learning. The classical framework of Regularization in Reproducing Kernel Hilbert Spaces (which is the basis of state-of-the-art supervised algorithms such as SVMs) is extended in several ways to utilize unlabeled data.

These extensions are embodied in the following contributions:

(1) *Manifold Regularization* is based on the assumption that high-dimensional data truly resides on low-dimensional manifolds. Ambient globally-defined kernels are combined with the intrinsic Laplacian regularizer to develop new kernels which immediately turn standard supervised kernel methods into semi-supervised learners. An outstanding problem of out-of-sample extension in graph transductive methods is resolved in this framework.

(2) *Low-density Methods* bias learning so that data clusters are protected from being cut by decision boundaries at the expense of turning regularization objectives into non-convex functionals. We analyze the nature of this non-convexity and propose deterministic annealing techniques to overcome local minima.

(3) The *Co-regularization* framework is applicable in settings where data appears in multiple redundant representations. Learners in each representation are biased to maximize consensus in their predictions through multi-view regularizers.

(4) We develop l_1 regularization and greedy matching pursuit algorithms for *sparse non-linear manifold regularization.*

(5) We develop specialized *linear algorithms* for very large sparse data matrices, and apply it for probing utility of unlabeled documents for text classification.

(6) Empirical results on a variety of semi-supervised learning tasks suggest that these algorithms obtain state-of-the-art performance.

ACKNOWLEDGMENTS

I am grateful to Partha Niyogi for inspiring me with his vision and deep passion this subject. His constant encouragement, patience and thoughtful advice, spread over the last six years across numerous hours on the blackboard, campus walks and squash courts, have been of immense help to me in maturing as a researcher. I hope to continue to collaborate with Partha.

I thank David McAllester for his support, helpful conversations and the opportunity to intern at the Toyota Technological Institute in the summer of 2004 where parts of this thesis were written. I learnt a lot in David's classes and talks. I admire David for his amazing technical sharpness, and scientific and philosophical views.

Jorge Nocedal graciously admitted me to his optimization classes at Northwestern University. I thank him for going very much out of his way to participate in my thesis committee in the middle of numerous other commitments and to also evaluate its contributions from a solid optimization perspective.

Sathiya Keerthi taught me innumerable nuts and bolts of machine learning and optimization. He graciously offered me consecutive internships at Yahoo! Research in the summer of 2005 and 2006. I continue to enjoy a close friendship and a highly productive collaboration with him.

I first met Olivier Chapelle at Max Planck Institute but developed a friendship with him more recently. Olivier is a role model for machine learning researchers of my generation in subscribing to very high standards of honesty and taste in research.

I thank Misha Belkin, Chu Wei, Subrata Rakshit and Deniz Erdogmus for enjoyable collaborations over the years. I hope we keep working together.

Dinoj Surendran, Irina Matveeva, Varsha Dani, G. Murali Krishnan, Xiaofei He and Hariharan Narayanan have always offered a willing ear for my ideas and questions,

and have welcomed the spirit of learning together.

Finally, I could not have come this far without the love and support of my wife, Deannna Barenboim, my parents, Renu and Subhash Sindhwani, my sister, Pooja Kakar and my brother, Sumeet Chawla.

Thank you all very much.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

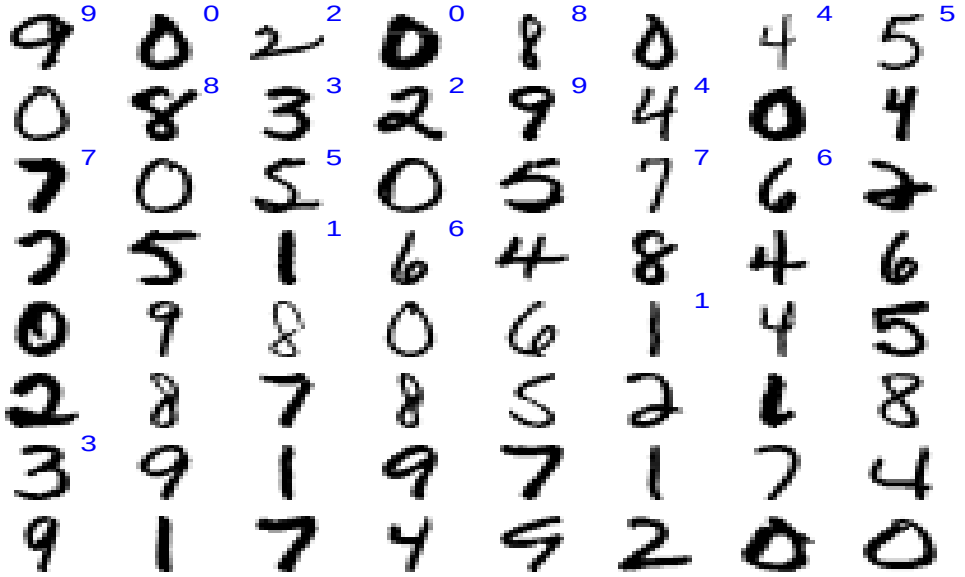
1.1 In "Layman's Terms"

Artificial Intelligence, as a scientific and engineering discipline, has a very challenging goal – to replicate on a machine the unique ability of humans to communicate via elaborate speech and language systems, to recognize and manipulate objects with remarkable dexterity, and to express much celebrated competence in arts, mathematics and science. The central working postulate of this thesis is that the process of Learning is the cornerstone of Intelligence. "Intelligent behaviour" is not programmed or memorized; it arises out of interaction with an environment that can display complex variability, where an internal representation of the world is adaptively constructed through the process of learning.

Numerous learning paradigms have been proposed to deal with the richness and complexity of tasks that an intelligence-capable agent might be required to learn. The contents of this thesis reside within the paradigm of *Learning from Examples*. For illustration, we consider the task of identifying handwritten digits, learnt by millions of humans in their early childhood. A stream of visual stimuli of digits constitutes the learning experience of a child. This stream is sometimes accompanied by labels provided by a supervisor, associating a stimulus with a specific category or desired response. Figure 1.1 shows a visual digit stream where some digits (*labeled examples*) have been labeled by a supervisor.

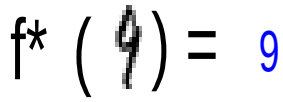
The process of learning from examples, is that of utilizing such a sequence of *training examples* (which we may call $\mathcal{D}$) to find a good map (say $f^\star$) from this space of images (called $\mathcal{X}$) to the space of labels (called $\mathcal{Y} = \{0, 1, \ldots 9\}$), that frequently

Figure 1.1: Examples for Learning: A supervisor provides some labels.



provides the correct response to a new, unseen input, such as in the instance shown in Figure 1.2.

Figure 1.2: Generalization from training examples to new unseen inputs.



Learning Algorithms may be designed to construct such maps by selecting a suitable function $f^\star$ from a *hypothesis space* $\mathcal{H}$ of functions, based on some performance criterion measured on the training examples. Typically, a loss function, which we denote by V, is used to evaluate the predictions given by a function with respect to the labels provided by the supervisor. The hope is that sufficient training results in successful generalization to novel examples and convergence to a function that is optimal in some sense. The human brain presumably constructs such an optimal map when trained in early childhood.

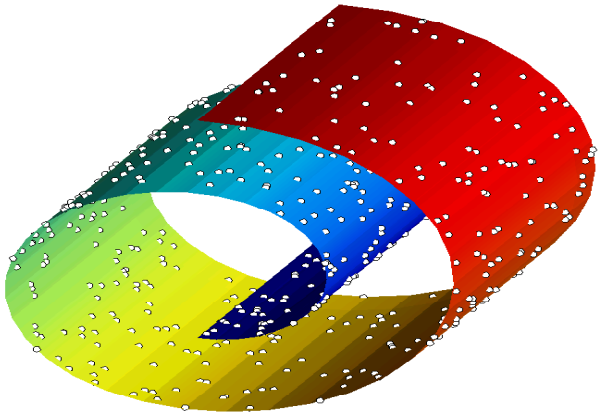
A motivating observation for this thesis is that the human learning apparatus seems to somehow remarkably learn highly complex tasks without much supervised assistance. In the digit recognition example above, each visual stimulus forms a retinal image which is a collection of signals from millions of photoreceptor cells in the human eye. Examples for learning may be considered as points in a very high dimensional abstract image space where dimensions corresponds to the strengths of signals on these cells.

The process of learning to recognize digits may be conceptualized as that of partitioning this image space into regions, and identifying each region with a meaningful perceptual category. Learning mechanisms in the brain presumably construct such perceptual regions. Labeled training examples may be conceptualized as points sampled from an ideal partition, providing partial information to a learner on what the optimal perceptual boundaries should look like. In a very high dimensional space, large number of labeled examples are required for reliable identification of these optimal boundaries. This phenomenon is also called the *curse of dimensionality*. A natural question then is the following: How is the curse of dimensionality overcome by the human brain ?

A key insight into this problem comes by considering the aspect of *perceptual invariance*. Even though stimuli is high dimensional, the variability in the images may be characterized by a low dimensional parameter space. For example, consider the numerous instances of the digit 0 shown in Figure 1.1. These instances differ from one another in the degree of slants, rotations, translations, or line thickness. The set of digit 0 generated by varying these parameters forms a continuous, possibly non-linear *manifold* ("surface") in the stimulus space. This structure has much lower dimensionality as compared to the stimulus space. Thus, examples for learning may become available in a high-dimensional *ambient* space, but may truly all lie on a low

dimensional structure. This situation is depicted in Figure 1.3 where points sampled from an underlying a 2-dimensional manifold are shown in a 3-dimensional ambient space.

Figure 1.3: Examples for learning may be points on a low-dimensional manifold embedded in a high-dimensional space.

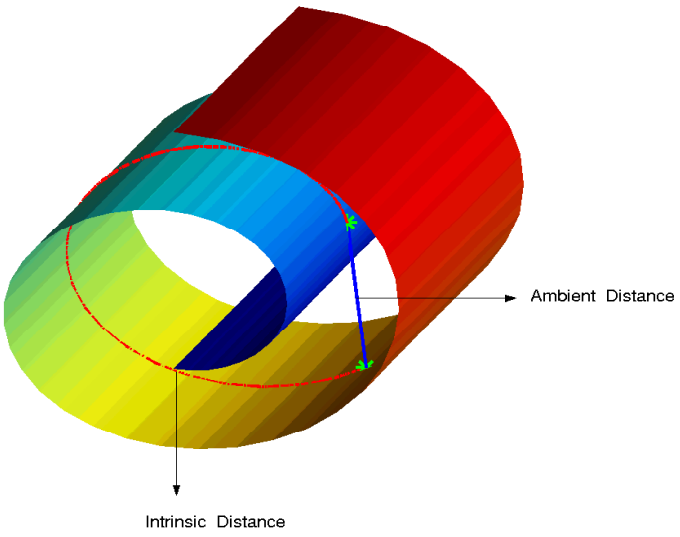


As another example, consider speech production: the articulatory organs can be modeled as a collection of tubes so that the space of speech sounds is parameterized by lengths and widths of the tubes. However, the canonical representation of the raw speech signal is very high-dimensional. In almost any imaginable source of meaningful high-dimensional stimulus space, the space of configurations that are actually realized truly occupies only a tiny portion of the total volume of possibilities available. Perhaps it is possible to escape the curse of dimensionality because the intrinsic dimensionality, or true volume of configurations, of stimuli happens to fortunately be small in natural learning problems. Conversely, one is tempted to conjecture that a task can only be naturally learnt if it has the above property.

Such considerations suggest a few things about the nature of the process of learning. Firstly, it might be meaningful to construct categorical regions taking the intrinsic low-dimensional perceptual structure into account. Figure 1.4 illustrates the

difference between *intrinsic* and the *ambient* distances in the stimulus space – the notions of similarity given by the two can be radically different. The "correct" notion of similarity is important for constructing the right perceptual boundaries.

Figure 1.4: Intrinsic and ambient distances may be very different



Note, also, that within such a framework, examples *not labeled* by a teacher (see Figure 1.1) also have a very significant role to play for learning – even if we dont know what categories they belong to, their collective presence can tell us about what possible stimuli configurations we can encounter. With sufficient unlabeled examples(*observation*), it might be possible to learn from very few labeled examples (*instruction*). Clearly, the human perceptual apparatus is inundated with a constant stream of acoustic utterances and visual imprints with no identifiable categorical markers (plain observation). Only a small amount of instructive feedback seems to be necessary for a child to learn the acoustic-to-phonetic mapping in any language and to learn to seamlessly process visual information. One can argue that the dominant mode of learning is semi-supervised.

In addition to its appeal in perception and cognition, semi-supervised learning has a major engineering significance. In many domains, large amounts of cheaply generated unlabeled examples can often be augmented to a small amount of expensively labeled examples. For example, a search engine may crawl the web and collect billions of documents. No amount of human resource is sufficient to process this ever-growing collection of documents to identify their (categorical) content. In such cases, semi-supervised learning technologies can be advantageously deployed.

The motivating question for this thesis, then, is as follows : *How can the prior knowledge of geometric structures encoded in unlabeled examples be incorporated in algorithmic procedures for learning from examples ? In particular, what is an appropriate mathematical and algorithmic framework for learning from both labeled and unlabeled examples ?*

This thesis develops families of algorithms for semi-supervised learning.

1.2 Learning from Examples

In this section, we introduce the classical paradigm of learning from examples, and describe some key components of the mathematical and algorithmic framework which this thesis will build on. We also setup the notation to be used throughout this thesis. Additionally, an overview is provided on the central problems and techniques in machine learning particularly relevant to this thesis.

1.2.1 Supervised Learning

This section discusses the standard mathematical framework for supervised learning. We assume that stimuli or inputs shown to a learner may be represented by elements of a set $\mathcal{X}$, and categorical assignments or desired responses are elements of a set

$\mathcal{Y}$. An unknown probability distribution $\mathcal{P}$ on the product space $\mathcal{X} \times \mathcal{Y}$ models the variability in the input-output space. A sampling process from this distribution generates a set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{l}$ of *training examples* for the learner. This set contains pairs $(\boldsymbol{x}, y) \in \mathcal{X} \times \mathcal{Y}$ drawn i.i.d from the distribution $\mathcal{P}$.

Within this setup, one may consider regression (where $\mathcal{Y}$ is continuous) or classification (where $\mathcal{Y}$ is a finite discrete set). Much of the subsequent discussion will focus on the canonical machine learning task of binary classification where the set $\mathcal{Y} = \{-1, +1\}$ and examples $\boldsymbol{x} \in \mathcal{R}^d$ may correspondingly be identified as belonging to positive or negative class. The goal of supervised learning is to use the training data $\mathcal{D}$ to construct a function $f : \mathcal{X} \mapsto \{-1, +1\}$. This function is constructed by a *learning algorithm* $\mathcal{A}$ that selects a suitable function from a *hypothesis space* of functions $\mathcal{H}$, based on some goodness criterion that typically involves a *loss function* V measured on the training data $\mathcal{D}$.

Loss Functions : A loss function $V(f(\boldsymbol{x}), y)$ measures the cost of predicting $f(\boldsymbol{x})$ when the true prediction is y. Table 1.1 lists and Figure 1.5 shows different loss functions utilized for regression and classification.

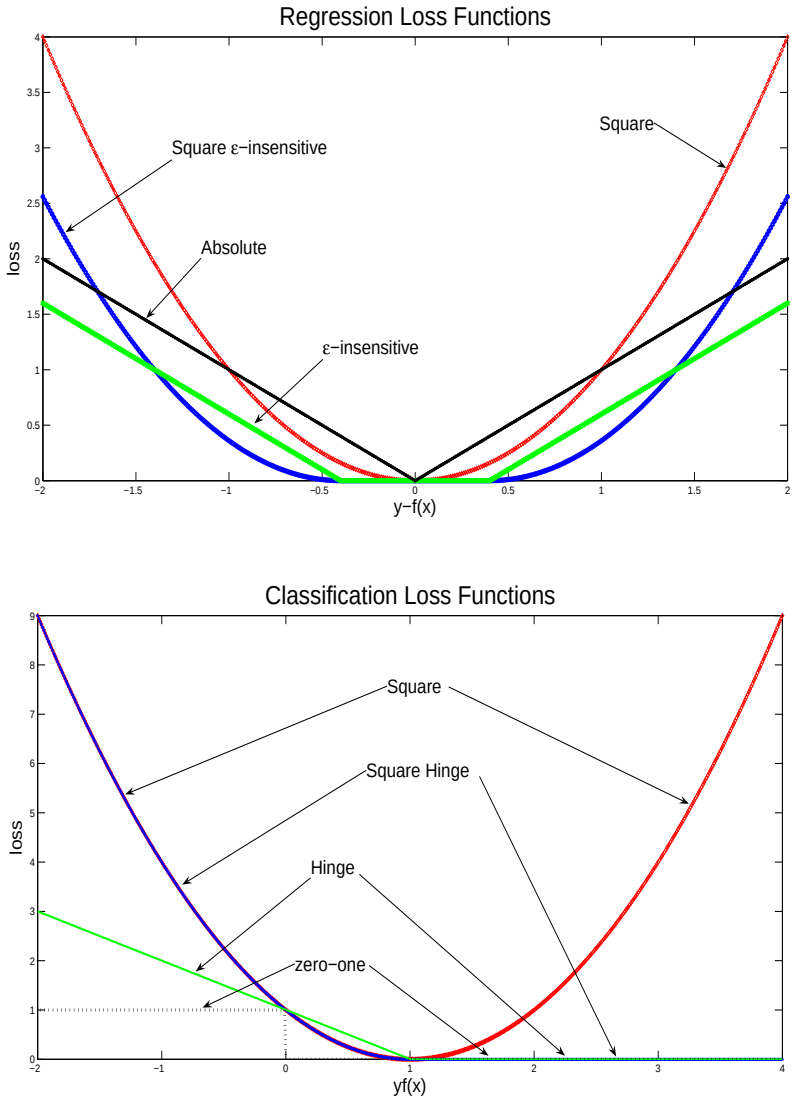
Table 1.1: Loss functions shown in Figure 1.5

Regression	$V(f(\boldsymbol{x}), y)$	Classification	$V(f(\boldsymbol{x}), y)$
Absolute	$\lvert y - f(\boldsymbol{x})\rvert$	zero-one	$\theta(yf(\boldsymbol{x}))$
Square	$(y - f(\boldsymbol{x}))^2$	Square	$(1 - yf(\boldsymbol{x}))^2$
ϵ-insensitive	$\max[\lvert y - f(\boldsymbol{x})\rvert - \epsilon, 0]$	Hinge	$\max[1 - yf(\boldsymbol{x}), 0]$
sqr ϵ-insensitive	$\max[\lvert y - f(\boldsymbol{x})\rvert - \epsilon, 0]^2$	sqr Hinge	$\max[1 - yf(\boldsymbol{x}), 0]^2$

For binary classification, the most natural loss function is the zero-one step function $V(f(\boldsymbol{x}), y) = \theta(yf(\boldsymbol{x}))$ which incurs a loss of 1 when the predicted class and true class do not match ($yf(\boldsymbol{x}) < 0$) and 0 otherwise. In practice, classification algorithms actually use convex loss functions that upper bound the zero-one loss. Convexity

leads to tractable training objective functions. For classification with the hinge loss, large positive values of $yf(\boldsymbol{x})$ are interpreted as confident classifications, and linear penalty is incurred when this value falls below 1. Within various choices for convex loss functions, some such as squared hinge loss happen to also be differentiable.

Figure 1.5: Typical Loss Functions for Learning



Risk Minimization : Given a hypothesis space $\mathcal{H}$, a loss function V, and training data $\mathcal{D}$, the goal of learning may be described as finding the function $g^\star \in \mathcal{H}$ that incurs minimum expected loss $\mathcal{R}[f]$ over the entire space $\mathcal{X} \times \mathcal{Y}$:

$$g^* = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \mathcal{R}[f] = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \int_{\mathcal{X} \times \mathcal{Y}} V(f(\boldsymbol{x}), y) d\mathcal{P}(\boldsymbol{x}, y)$$

However, this minimization cannot be performed since the probability distribution $\mathcal{P}$ is unknown. A natural learning algorithm might replace the above objective function with the average empirical loss $\mathcal{R}_{emp}[f]$ over the training data $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{l}$, and find the function $f^* \in \mathcal{H}$ that minimizes this estimate :

$$f^* = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \mathcal{R}_{emp}[f] = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \frac{1}{l} \sum_{i=1}^{l} V(f(\boldsymbol{x}_i), y_i)$$

This induction principle, called the *Empirical Risk Minimization* (ERM) principle, has been well studied in learning theory. Statistical learning-theoretic questions have been asked about two desirable requirements of the learnt function f^* : (1) In order for f^* to be predictive, the average empirical loss $R_{emp}[f^*]$ and true expected loss $\mathcal{R}[f^*]$ must be close so that empirical loss can be trusted as a true reflection of the performance over the entire space. This is the requirement of *generalization*. (2) The expected loss of f^* must be close to the expected loss of g^*, the function in the hypothesis space that has the best performance. This is the requirement of *consistency*.

Another consideration is the ill-posedness of ERM. An *ill-posed problem* fails to satisfy one of the following criteria for well-posedness : (1) Existence of a solution, (2) Uniqueness of the solution (3) Stability of the solution (continuous dependence on the data). It is easy to exhibit an undesirable solution that minimizes empirical loss :

A rote learner that reproduces labels on training data and predicts 0 elsewhere. For unique, stable and useful solutions, it turns out that we need to impose constraints on the hypothesis space.

The critical insight arising from such considerations, paraphrased from [97], is the following : *For successful generalization from a small set of training examples, one has to look for the optimal relationship between the amount of training data, the quality of the approximation of the data by the function chosen from the hypothesis space, and a measure of capacity of the hypothesis space* $\mathcal{H}$.

This insight is closely related to the Occam's Razor principle ("the simplest explanation is the best"), Minimum Description Length, and Bayesian paradigms that relate capacity of hypothesis spaces to prior beliefs, and the principle of Structural Risk minimization that imposes a hierarchical structure on a rich hypothesis class based on notions of capacity.

In this thesis, we are concerned with the implementation of the above key insight, by the following scheme, known as the method of *Tiknonov Regularization* [94]:

$$f^* = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \mathcal{R}_{emp}[f] + \gamma \Omega[f] \tag{1.1}$$

Given a certain amount of training data, the quality of the approximation of the data by a function f is measured by the empirical loss $\mathcal{R}_{emp}[f]$ (with respect to some loss function V), and the capacity of the hypothesis space is measured indirectly via a penalty function Ω that measures the complexity of functions in the hypothesis space. The *regularization parameter* γ is a real number that is used to trade off data-fit and function complexity. $\mathcal{H}$ is a space of real valued functions that are thresholded for binary classification. Note that the above method is equivalent (via Lagrange

multipliers) to *Ivanov Regularization*:

$$\underset{f \in \mathcal{H}}{\operatorname{argmin}} \mathcal{R}_{emp}[f] \quad \text{subject to:} \quad \Omega[f] < \tilde{\gamma}$$

where we minimize empirical loss over a subset of $\mathcal{H}$ with limited complexity.

Successful implementation of regularization depends critically on the hypothesis space $\mathcal{H}$ and the penalty function Ω.

Let us construct a list of desirable properties of a hypothesis space for learning by regularization:

1. **Rich Function Space :** $\mathcal{H} : \mathcal{X} \mapsto \mathcal{R}$ must be a real-valued function space over X. It is natural to endow $\mathcal{H}$ with a Hilbert space structure. A general-purpose learner needs to be capable of learning tasks where the underlying true dependency may be highly complex. We require $\mathcal{H}$ to contain functions that can well-approximate complex target functions (e.g., hypothesis space dense in an L_2 space of functions). This requirement is not in conflict with the assertion that hypothesis spaces need to be constrained for successful generalization, once we formulate a notion of complexity on this space and use it in the regularization tradeoff.

2. **Defined at all Points :** Evaluation of empirical losses involve evaluating functions f at arbitrary examples $\boldsymbol{x} \in X$ in the training set (since $\mathcal{P}$ is truly unknown, the support of the data is not known apriori). Thus, $f(\boldsymbol{x})$ needs to exist for all $\boldsymbol{x} \in \mathcal{X}$. This innocuous property is not shared by many familiar functions (e.g $f(t) = 1/t$ is not defined at $t = 0$) and function spaces. For example, the Hilbert space of square-integrable functions L_2 actually contains equivalence classes of functions differing on sets of measure zero.

3. **"Truthful" Convergence :** Suppose f^* is the function learnt with data $\mathcal{D}$, and g^* is the best function that can be learnt in the hypothesis space. A basic requirement is that if f^* converges to g^* (it is natural to define convergence in terms of the norm defined by inner product in the Hilbert space) as the amount of data increases, convergence must be pointwise, i.e., for any $\boldsymbol{x} \in \mathcal{X}$, the value $f^*(\boldsymbol{x})$ converges to the value $g^*(\boldsymbol{x})$. This requirement allows $f^*(\boldsymbol{x})$ to be used to predict the value of the true dependency at all points $\boldsymbol{x}$.

   This requirement can be rephrased as follows : If two functions $f, g \in \mathcal{H}$ are close in the sense of distance derived from the inner product defined in the Hilbert space $\mathcal{H}$, then for all $\boldsymbol{x} \in \mathcal{X}$, $f(\boldsymbol{x})$ is close to $g(\boldsymbol{x})$.

   A more technical rephrasal of the above is in terms of requiring the evaluation function at $\boldsymbol{x}$ defined by $\mathcal{E}_{\boldsymbol{x}}[f] = f(\boldsymbol{x})$, that maps functions to their values at $\boldsymbol{x}$, to be a continuous map.

4. **Complexity control :** Given a suitable function space $\mathcal{H}$, we need to define a penalty function $\Omega[f]$ that measures the complexity of $f \in \mathcal{H}$. Given the Hilbert space structure, it is natural to define this penalty function via the norm of the function, i.e, $\Omega[f] = \|f\|_{\mathcal{H}}$.

5. **Tractable Optimization :** Lastly, we require that once we have collected training data, constructed our function space, and chosen suitable loss functions, the optimization problem of learning by regularization, can indeed be tractably solved for a large class of learning problems.

It is remarkable that there exist function spaces called *Reproducing Kernel Hilbert Spaces* that satisfy these requirements, as implications of a simple definition. Consequently, they have found widespread use in statistics and machine learning. A

number of popular algorithms such as Support vector machines, Regularized least squares, Ridge regression, splines, Radial basis functions etc arise out of the powerful framework of regularization in RKHS.

Our focus in this thesis will be on two popular approaches : Support vector machines and Regularized least squares. These are discussed in Section 1.2.4.

1.2.2 Reproducing Kernel Hilbert Spaces

This section is a tutorial overview of Reproducing Kernel Hilbert Spaces (RKHS)[12]. We recall some basic properties of RKHS and develop some intuition as to why they are useful.

RKHS as Hilbert Spaces with a nice property: An RKHS $\mathcal{H}$ is a Hilbert space of functions, i.e., a vector space of functions $f : \mathcal{X} \mapsto \mathcal{R}$ endowed with an inner product $\langle ., . \rangle_{\mathcal{H}}$ such that in the associated norm $\|.\|_{\mathcal{H}}$, $\mathcal{H}$ forms a complete metric space.

Additionally, an RKHS has a nice property : *If two functions $f, g \in \mathcal{H}$ are close (i.e $\|f - g\|_{\mathcal{H}}$ is small), then their values $f(\boldsymbol{x}), g(\boldsymbol{x})$ are close at all points $\boldsymbol{x} \in \mathcal{X}$.*

A formal way of stating this property is in terms of continuity of evaluation functionals. Consider a point $\boldsymbol{x} \in X$. The evaluation functional $\mathcal{E}_{\boldsymbol{x}} : \mathcal{H} \mapsto \mathcal{R}$ is defined by the map $\mathcal{E}_{\boldsymbol{x}}[f] = f(\boldsymbol{x})$, i.e., the functional maps functions to their values at point $\boldsymbol{x}$. The above property follows from the statement that evaluational functionals at all points in $\mathcal{X}$, are continuous on $\mathcal{H}$.

The Reproducing Property: The "reproducing" property arises out of an important property of the inner product expressed by the *Riesz Representation Theorem*. To state this theorem, we first recall the meaning of a linear functional. A linear functional $L : E \mapsto \mathcal{R}$ on a euclidean space E associates a real number $L[e]$ with every

vector $e \in E$ in such a way that $L[\alpha_1 e_1 + \alpha_2 e_2] = \alpha_1 L[e_1] + \alpha_2 L[e_2]$ for all vectors $e_1, e_2 \in E$ and real numbers α_1, α_2. It is easy to see that for a fixed vector $u \in E$, the functional $L[e] = \langle u, e \rangle$ defined via the inner product is a linear functional on E. The Reisz Representation Theorem simply states the converse: *Every continuous linear functional L may be represented as an inner product, i.e., there exists a vector $u \in E$ such that $L[e] = \langle u, e \rangle \ \forall e \in E$.* The element u is called the representer of L.

Since the evaluation functional $\mathcal{E}_{\boldsymbol{x}}$ is a linear continuous functional on $\mathcal{H}$, the Reisz representation theorem guarantees the existence of its representer $k_{\boldsymbol{x}} \in \mathcal{H}$ such that the following property holds $\forall f \in \mathcal{H}$:

$$\mathcal{E}_{\boldsymbol{x}}[f] = \langle k_{\boldsymbol{x}}, f \rangle_{\mathcal{H}} = f(\boldsymbol{x}) \tag{1.2}$$

Since taking the inner product of f with $k_{\boldsymbol{x}}$ produces the value of f at $\boldsymbol{x}$, this property is often called the *reproducing* property. the reproducing property also shows that RKHS functions are pointwise defined.

Kernels: We have seen that the definition of RKHS implies existence of real valued functions $k_{\boldsymbol{x}} \in \mathcal{H} : \mathcal{X} \mapsto \mathcal{R}$ that are representers of the evaluation functional at $\boldsymbol{x}$. With any RKHS, one may associate a function $K : \mathcal{X} \times \mathcal{X} \mapsto \mathcal{R}$ defined as the following map $K(\boldsymbol{x}, \boldsymbol{z}) = \langle k_{\boldsymbol{x}}, k_{\boldsymbol{z}} \rangle_{\mathcal{H}}$. This function is called the *Kernel* of the RKHS. Using the reproducing property, one may see that $K(\boldsymbol{x}, \boldsymbol{z}) = k_{\boldsymbol{x}}(\boldsymbol{z}) = k_{\boldsymbol{z}}(\boldsymbol{x}) = K(\boldsymbol{z}, \boldsymbol{x})$ is a symmetric function. In subsequent discussion, we may use $K_{\boldsymbol{x}}(.)$ or $K(\boldsymbol{x}, .)$ or $k_{\boldsymbol{x}}(.)$ to denote the representer of evaluation at $\boldsymbol{x}$.

At this point, one might ask the following question : If for a Hilbert space $\mathcal{H}$, there is a function $K : \mathcal{X} \times \mathcal{X} \mapsto \mathcal{R}$ such that $K(\boldsymbol{x}, .) \in \mathcal{H}$ and $\forall f \in \mathcal{H} \ \ \langle f, K_{\boldsymbol{x}} \rangle = f(\boldsymbol{x})$, then is it an RKHS, i.e., are all evaluation functionals $\mathcal{E}_{\boldsymbol{x}}, \boldsymbol{x} \in X$ continuous? The answer is yes, and is easily argued from the observation that $\forall \boldsymbol{x} \in \mathcal{X}, \forall f \in$

$\mathcal{H}$ $|\mathcal{E}_{\boldsymbol{x}}[f]| = |f(\boldsymbol{x})| = |\langle f, K_{\boldsymbol{x}}\rangle| \leq \|f\|_{\mathcal{H}}\|K_{\boldsymbol{x}}\|_{\mathcal{H}} = \|f\|_{\mathcal{H}}\sqrt{K(\boldsymbol{x},\boldsymbol{x})}$ where the last fast is the Cauchy-Schwarz inequality.

Additionally, one can confirm from the same argument that convergence in the RKHS norm implies pointwise convergence: $\forall \boldsymbol{x} \in \mathcal{X}, \exists M_{\boldsymbol{x}} \in \mathcal{R}^+$ such that $|f(\boldsymbol{x}) - g(\boldsymbol{x})| \leq M_{\boldsymbol{x}}\|f-g\|_{\mathcal{H}}$. Here, $M_{\boldsymbol{x}} = \sqrt{K(\boldsymbol{x},\boldsymbol{x})}$. Thus, if f converges to g in the RKHS norm, then for all $\boldsymbol{x} \in \mathcal{X}$, $f(\boldsymbol{x})$ converges to $g(\boldsymbol{x})$.

We see that the definition of RKHS directly allows us to satisfy the desirable requirements of pointwise-definition and "truthful" convergence.

We now develop a characterization of RKHS in terms of positive definite functions.

Reproducing Kernels are positive definite functions: A function $K : \mathcal{X} \times \mathcal{X} \mapsto \mathcal{R}$ is positive definite if for any collection of points, i.e $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n \in \mathcal{X}$, the $n \times n$ gram matrix G defined as $G_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a positive definite matrix, i.e., $\forall \boldsymbol{\alpha} \in \mathcal{R}^n$ $\boldsymbol{\alpha}^T G \boldsymbol{\alpha} \geq 0$.

A reproducing kernel is a positive definite function. This can be easily seen through the reproducing property. Given any $\boldsymbol{\alpha} \in \mathcal{R}^n$, $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n \in \mathcal{X}$, consider the function $f(.) = \sum_{i=1}^n \boldsymbol{\alpha}_i K_{\boldsymbol{x}_i}(.)$. Then $f \in \mathcal{H}$ and $\|f\|_{\mathcal{H}} = \langle f, f\rangle_{\mathcal{H}} = \langle \sum_{i=1}^n \boldsymbol{\alpha}_i K_{\boldsymbol{x}_i}, \sum_{i=1}^n \boldsymbol{\alpha}_i K_{\boldsymbol{x}_i}\rangle_{\mathcal{H}} = \boldsymbol{\alpha}^T G \boldsymbol{\alpha} \geq 0$.

Positive definite functions are Reproducing Kernels: The theorem below completes a remarkable characterization of RKHS in terms of positive definite functions:

Theorem 1.2.1 (Moore-Aronszajn)**.** *If $K : \mathcal{X} \times \mathcal{X} \mapsto \mathcal{R}$ is a positive definite function, then there exists a unique RKHS whose reproducing kernel is K. This RKHS is simply the completion of the linear span of the kernel functions $K(\boldsymbol{x}, .)$ endowed with the*

following inner product :

$$\langle \sum_{i=1}^{n} \boldsymbol{\alpha}_i K_(\boldsymbol{x}_i, .), \sum_{j=1}^{m} \beta_j K_(\boldsymbol{z}_j, .) \rangle = \sum_{i=1}^{n} \sum_{j=1}^{m} \boldsymbol{\alpha}_i \beta_j K(\boldsymbol{x}_i, \boldsymbol{z}_j)$$

In subsequent discussion, we use the notation of $\mathcal{H}_K$ for RKHS characterized by the kernel function K and denote the RKHS norm $\|f\|_{\mathcal{H}}$ as $\|f\|_K$.

The following positive definite functions have been extensively used as kernels in Regularization algorithms in RKHS : (here, $x, y \in \mathcal{R}^d$)

$$\begin{aligned} K(\boldsymbol{x}, \boldsymbol{z}) &= \boldsymbol{x}^T y \quad \text{Linear Kernels} \\ K(\boldsymbol{x}, \boldsymbol{z}) &= \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{z}\|^2) \quad \text{Gaussian Kernels} \\ K(\boldsymbol{x}, \boldsymbol{z}) &= (\boldsymbol{x}^T \boldsymbol{z} + 1)^d \quad \text{Polynomial Kernels} \end{aligned}$$

Kernels as inner products in feature spaces: Positive definite kernels are fundamentally related to inner-product spaces. The kernel is considered as an inner product in a high-dimensional feature space $\mathcal{F}$ to which $\mathcal{X}$ is non-linearly mapped via a "feature map".

This characterization can be made precise by the following proposition.

Proposition *A function $K : \mathcal{X} \times \mathcal{X}$ is a positive definite function if and only if there exists a mapping $\phi : \mathcal{X} \mapsto \mathcal{F}$ such that*

$$\forall \boldsymbol{x}, \boldsymbol{z} \in \mathcal{X} : K(\boldsymbol{x}, \boldsymbol{z}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) \rangle_{\mathcal{F}}$$

If $K : \mathcal{X} \times \mathcal{X}$ is a reproducing kernel, one can use the Moore-Aronszajn theorem to produce the feature map $\phi : \boldsymbol{x} \mapsto K(\boldsymbol{x}, .)$ and identify the feature space $\mathcal{F}$ with the RKHS $\mathcal{H}$. This map may be interpreted as turning each point $\boldsymbol{x}$ into a function

$K(\boldsymbol{x}, .)$ which encodes similarity of $\boldsymbol{x}$ with all other points in the space. Conversely, given a feature space mapping defining the kernel function as above, one can establish positive definiteness based on non-negativity of the norm in the feature space : Given a gram matrix G obtained by evaluating the kernel on points $\{\boldsymbol{x}_i\}_{i=1}^n$, it is easy to show that $\forall \boldsymbol{\alpha} \in R^n \;\; \boldsymbol{\alpha}^T G \boldsymbol{\alpha} = \| \sum_{i=1}^n \boldsymbol{\alpha}_i \phi(\boldsymbol{x}_i) \|_{\mathcal{F}} \geq 0$.

Note that the positive definiteness of the kernel function is necessary and sufficient for the existence of such a feature map. The Mercer's theorem, discussed below, is often used to exhibit a feature map, even though the conditions of the theorem are sufficient but not necessary for a feature map to exist.

Integral Operators and Mercer Space Viewpoint: Here we discuss another view of RKHS.

Suppose $\mathcal{X}$ is a finite space i.e $\mathcal{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$. Then the kernel function $K : \mathcal{X} \times \mathcal{X} \mapsto \mathcal{R}$ has a finite $n \times n$ gram matrix defined as $G_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ which is positive-definite symmetric matrix. Consider the eigenvalue decomposition of the gram matrix : $G = UDU^T$. Then, $D = diag(\lambda_1, \lambda_2, ..., \lambda_n)$ is a diagonal matrix of eigenvalues of G given by $\lambda_1 \geq \lambda_2, ..., \geq \lambda_n \geq 0$ and $U = [u_1 ... u_n]$ is an $n \times n$ orthonormal matrix. Denote $u_i(j) = U_{ji}$ be the j^{th} component of the i^{th} eigenvector. Consider a feature map $\phi : \mathcal{X} \mapsto \mathcal{R}^n$ given by $\psi(\boldsymbol{x}_i) = (\sqrt{\lambda_1} u_1(i), ..., \sqrt{\lambda_n} u_n(i))$. Then, the dot product in this feature space can easily be shown to be $\langle \psi(\boldsymbol{x}_i), \psi(\boldsymbol{x}_j) \rangle = K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{k=1}^n \lambda_k u_k(i) u_k(j)$. Thus, the feature map takes $\boldsymbol{x}_i$ to coordinates in a feature space given by the i^{th} element of the eigenvectors scaled by the square root of corresponding eigenvalues.

Analogously, let $\mathcal{X}$ be a compact subset of $\mathcal{R}^d$, and let ν be a measure on X supported on all of X, and let K further be continuous. Here, instead of considering eigensystems of a gram matrix, we consider the integral operator $L_K : L_2(\mathcal{X}, \nu) \mapsto$

$L_2(\mathcal{X}, \nu)$ defined as :

$$L_K[f](t) = \int_X K(\boldsymbol{s}, \boldsymbol{t}) f(\boldsymbol{s}) d\nu(\boldsymbol{s})$$

and study its eigensystem $L_K[f] = \lambda f$ (the compactness of $\mathcal{X}$, and symmetry and positive definiteness of the kernel, result in this integral operator to be compact, self-adjoint, positive, and therefore, possessing a discrete spectrum). If $\{\phi_j\}_{j=1}^{\infty}$ are eigenfunctions of the integral operator with eigenvalue $\{\lambda_j\}_{j=1}^{\infty}$, Mercer's Theorem [77] says that the kernel function can be expanded as a uniformly convergent series $K(x, y) = \sum_{i=1}^{\infty} \lambda_i \phi_i(\boldsymbol{x}) \phi_i(\mathbf{y})$. One may define a Mercer feature map $\psi : X \mapsto l_2$ given by

$$\psi(\boldsymbol{x}) = (\sqrt{\lambda_1}\phi_1(\boldsymbol{x}), \sqrt{\lambda_2}\phi_2(\boldsymbol{x}), ...)$$

and interpret the kernel function as providing the inner product in this feature space.

Consider functions in the span of the eigenfunctions of the kernel,

$$\{f(\boldsymbol{x}) = \sum_i c_i \phi_i(\boldsymbol{x})\} = w^T \psi(x)\}$$

where $w_i = \frac{c_i}{\sqrt{\lambda_i}}$. This space can be shown to be the RKHS associated with the kernel, when restricted to $\|f\|_K < \infty$, where the norm may be defined via the the following inner product : $\langle \sum_i c_i \phi_i(\boldsymbol{x}), \sum_i d_i \phi_i(\boldsymbol{x}) \rangle_K = \sum_i \frac{c_i d_i}{\lambda_i}$ i.e $\| \sum_i c_i \phi_i(\boldsymbol{x}) \|_K^2 = \sum_i \frac{c_i^2}{\lambda_i} = w^T w$. One can check that the reproducing property holds, and that the RKHS defined via the kernel eigenfunctions is the same as the one defined in the Moore-Aronszajn theorem and is independent of the measure ν [38].

When the kernel is strictly positive definite, and all the infinite eigenvalues $\{\lambda_j\}_{j=1}^{\infty}$ are strictly positive, the RKHS is dense in $L_2(X, \nu)$. The gaussian kernel, for instance, defines a dense RKHS and provides a rich hypothesis space for learning.

RKHS norm as Smoothness Penalty: We now suggest intuitive connections between the RKHS norm $\|f\|_K^2$ and the complexity of f in terms of its smoothness. In particular, a large class of kernels can be associated with regularization operators P of so that the RKHS norm $\|f\|_K^2 = \|Pf\|^2$, where $\|Pf\|^2$ is a direct L_2 penalty on large derivatives of f[1]. Here, we consider some examples:

Linear Kernels : Suppose we choose the linear kernel on $\mathcal{X} \subset R^d : K(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{x}^T \boldsymbol{z}$. Consider a function in the RKHS defined by this kernel :

$$f(x) = \sum_{i=1}^{m} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i) = \sum_{i=1}^{m} \alpha_i \boldsymbol{x}_i^T \boldsymbol{x} = (\sum_{i=1}^{m} \alpha_i \boldsymbol{x}_i^T)\boldsymbol{x} = \mathbf{w}^T \boldsymbol{x}$$

where $\mathbf{w} = \sum_{i=1}^{m} \alpha_i x_i$ is the weight vector of the linear function. Thus functions in the RKHS are linear functions $f(x) = \mathbf{w}^T \boldsymbol{x}$. The norm of this function is given by :

$$\|f\|_K^2 = \sum_{i=1}^{m} \alpha_i \alpha_j \boldsymbol{x}_i^T \boldsymbol{x}_j = (\sum_{i=1}^{m} \alpha_i \boldsymbol{x}_i)^T (\sum_{i=1}^{m} \alpha_i \boldsymbol{x}_i) = \mathbf{w}^T \mathbf{w}$$

Thus, the measure of complexity of linear functions is the magnitude of the weight vector, which intuitively is a penalty on smoothness because for $f(x) = \mathbf{w}^T \boldsymbol{x}$, $\mathbf{w}^T \mathbf{w} = (\nabla f)^T (\nabla f)$ where ∇f is the gradient of f.

For classification, there is another interpretation of such a penalty in terms of margin of separation between a classification hyperplanes and point clouds. Consider a collection of points belonging to two classes $D = \{\boldsymbol{x}_i, y_i\}_{i=1}^{l}, \boldsymbol{x}_i \in \mathcal{R}^d, y_i \in \{-1, +1\}$. The distance of a point from the hyperplane $\{\boldsymbol{x} \in \mathcal{R}^d : \mathbf{w}^T \boldsymbol{x} = 0\}$ is given by $\frac{|\mathbf{w}^T \boldsymbol{x}_i|}{\|\mathbf{w}\|}$. Let $d(\mathbf{w}; D)$ be the distance of the hyperplane $\mathbf{w}$ from the nearest point in the set D, i.e, $d(\mathbf{w}; D) = min_{\boldsymbol{x}_i} \frac{|\mathbf{w}^T \boldsymbol{x}_i|}{\|\mathbf{w}\|}$. Note that the set of weight vectors with the

1. This connection is crystallized by identifying Kernels as Greens functions of regularization operators, see [77].

same direction but different magnitudes $\{\alpha \mathbf{w} | \alpha \in \mathcal{R}\, \mathbf{w} \in \mathcal{R}^d, \mathbf{w}^T w = 1\}$ forms an equivalence class whose elements define the same hyperplane. Given a weight vector, one can choose a canonical weight vector from its equivalence class such that $min_{\boldsymbol{x}_i} |\mathbf{w}^T \boldsymbol{x}_i| = 1$. Then, $d(\mathbf{w}; D) = \frac{1}{\|\mathbf{w}\|}$. The support vector machine algorithm finds the weight vector that maximizes $d(\mathbf{w}; D)$ or equivalently minimizes $\mathbf{w}^T \mathbf{w}$ subject to the condition that $\mathbf{w}$ separate the two classes. This *optimal separating hyperplane* separates the two classes with maximum margin.

Gaussian Kernels : The RKHS associated with the Gaussian kernel $K(\boldsymbol{x}, \boldsymbol{z}) = \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{z}\|^2)$ can be shown to correspond to two equivalent regularization operators P, such that $\|f\|_K^2 = \|Pf\|^2 = \int dx \sum_m \frac{\sigma^{2m}}{m! 2^m} (O^m f(x))^2$ where $O^{2m} = \triangle^m, O^{2m+1} = \nabla \triangle^m$, with $\triangle$ as the laplacian operator and ∇ as the gradient operator. Also, an equivalent representation of P in fourier space is :

$$\|f\|_K^2 = \|Pf\|^2 = \int_{\mathcal{R}^d} |\tilde{f}(\omega)|^2 \exp(\frac{-\sigma^2 \|\omega\|^2}{2}) d\omega$$

where $\tilde{f}(\omega)$ is the fourier transform of $f(x)$. Thus the RKHS norm corresponds to a particular measure of variation in f in terms of magnitudes of its derivatives or high frequency components.

1.2.3 Representer Theorem

Once a kernel function K, or equivalently the associated RKHS $\mathcal{H}_K$ is chosen, the regularization problem may be written as :

$$f^* = \underset{f \in \mathcal{H}_K}{\operatorname{argmin}} R_{emp}[f] + \gamma \|f\|_K^2 \tag{1.3}$$

The remarkable representer theorem shows that the minimizer of this problem has a particular form involving a finite number of variables. Thus, what began as possibly infinite dimensional optimization problem over a function space, is reduced to a finite dimensional problem providing passage to a suite of tractable algorithms.

Theorem 1.2.2. Representer Theorem : *Let $R_{emp}[f]$ be an empirical loss measure over training data $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{l}$ of a function f. Then each minimizer $f^\star \in \mathcal{H}_K$ of the regularized risk*

$$R_{emp}[D] + \gamma \|f\|_{H_K}^2 \tag{1.4}$$

admits a representation of the form $f(\boldsymbol{x}) = \sum_{i=1}^{l} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i)$

Note that R_{emp} might be the empirical average loss of some loss function V. This theorem, however, holds more generally and allows for arbitrary (even non-additive) loss functions, so long as it involves only point evaluations of a function on the training data.

Proof. Decompose any $f \in \mathcal{H}_K$ into a part contained in the span of kernel functions $k(\boldsymbol{x}_1, .), \ldots, k(\boldsymbol{x}_l, .)$, and an orthogonal component:

$$f(\boldsymbol{x}) = f_{||}(\boldsymbol{x}) + f_{\perp}(\boldsymbol{x}) = \sum_{i=1}^{l} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i) + f_{\perp}(\boldsymbol{x}) \tag{1.5}$$

where $f_{\perp} \in \mathcal{H}_K$ and $< f_{\perp}, k(x_i, .) >_{\mathcal{H}_K} = 0$. When evaluated on data, the reproducing property leads to the observation that $f(\boldsymbol{x}_i) = f_{||}(\boldsymbol{x}_i) \;\; \forall 1 \leq i \leq l$:

$$\begin{aligned} f(\boldsymbol{x}_i) &= f_{||}(\boldsymbol{x}_i) + f_{\perp}(\boldsymbol{x}_i) \\ &= < \sum_{i=1}^{l} \boldsymbol{\alpha}_i K(\boldsymbol{x}_j, .), K(\boldsymbol{x}_i, .) > + < f_{\perp}, K(\boldsymbol{x}_i, .) >_{\mathcal{H}_K} \\ &= \sum_{j=1}^{l} \boldsymbol{\alpha}_i K(\boldsymbol{x}_j, \boldsymbol{x}_i) \end{aligned}$$

Since $f_{\perp}$ only increases the norm, and the minimizer must have $f_{\perp} = 0$ i.e the minimizer admit a representation $f(\boldsymbol{x}) = \sum_{i=1}^{l+u} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i)$. □

Taking this form of the minimizer, plugging into the optimization problem, one can convert the original problem into a finite-dimensional problem of finding the coefficients α_i. We demonstrate this for two algorithms in the next section.

1.2.4 Algorithms : RLS and SVM

We now use the representer theorem to demonstrate solutions of optimization problems of the following form :

$$f^* = \underset{f \in \mathcal{H}_K}{\operatorname{argmin}} \frac{1}{l} \sum_{i=1}^{l} V(f(\boldsymbol{x}_i), y_i) + \gamma \|f\|_{\mathcal{H}}^2 \tag{1.6}$$

We focus on two classification algorithms where (1) V is the squared loss leading to Regularized Least Squares algorithm, and (2) V is the hinge loss leading to the Support Vector Machine. These loss functions are shown in Figure 1.5.

Regularized Least Squares

The Regularized Least Squares algorithm is a fully supervised method where we solve:

$$\min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} (y_i - f(\boldsymbol{x}_i))^2 + \gamma \|f\|_K^2 \tag{1.7}$$

The classical Representer Theorem can be used to show that the solution is of the following form:

$$f^\star(\boldsymbol{x}) = \sum_{i=1}^{l} \alpha_i^\star K(\boldsymbol{x}, \boldsymbol{x}_i) \tag{1.8}$$

Substituting this form in the problem above, we arrive at following convex differentiable objective function of the l-dimensional variable $\boldsymbol{\alpha} = [\alpha_1 \dots \alpha_l]^T$:

$$\boldsymbol{\alpha}^* = \operatorname{argmin} \frac{1}{l} (Y - K\boldsymbol{\alpha})^T (Y - K\boldsymbol{\alpha}) + \gamma \boldsymbol{\alpha}^T K \boldsymbol{\alpha} \tag{1.9}$$

where K is the $l \times l$ gram matrix $K_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and Y is the label vector $Y = [y_1 \dots y_l]^T$.

The derivative of the objective function vanishes at the minimizer :

$$\frac{1}{l} (Y - K\boldsymbol{\alpha}^*)^T (-K) + \gamma K \boldsymbol{\alpha}^* = 0$$

which leads to the following solution.

$$\boldsymbol{\alpha}^* = (K + \gamma l I)^{-1} Y \tag{1.10}$$

Support Vector Machines

Here we outline the SVM approach to binary classification problems. For SVMs, the following problem is solved :

$$\min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} (1 - y_i f(\boldsymbol{x}_i))_+ + \gamma \|f\|_K^2$$

where the hinge loss is defined as: $(1 - yf(\boldsymbol{x}))_+ = \max(0, 1 - yf(\boldsymbol{x}))$ and the labels $y_i \in \{-1, +1\}$.

Again, the solution is given by:

$$f^\star(\boldsymbol{x}) = \sum_{i=1}^{l} \alpha_i^\star K(\boldsymbol{x}, \boldsymbol{x}_i) \tag{1.11}$$

Following SVM expositions, the above problem can be equivalently written as:

$$\begin{aligned} \min_{f \in \mathcal{H}_K, \xi_i \in \mathcal{R}} & \frac{1}{l} \sum_{i=1}^{l} \xi_i + \gamma \|f\|_K^2 \\ \text{subject to : } & y_i f(\boldsymbol{x}_i) \geq 1 - \xi_i \quad i = 1, \ldots, l \\ & \xi_i \geq 0 \quad i = 1, \ldots, l \end{aligned} \tag{1.12}$$

Using the Lagrange multipliers technique, and benefiting from strong duality, the above problem has a simpler quadratic dual program in the Lagrange multipliers $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_l]^T \in \mathcal{R}^l$:

$$\begin{aligned} \boldsymbol{\beta}^\star &= \max_{\boldsymbol{\beta}\in\mathcal{R}^l} \sum_{i=1}^{l} \beta_i - \frac{1}{2}\boldsymbol{\beta}^T Q \boldsymbol{\beta} \\ \text{subject to :} \quad & \sum_{i=1}^{l} y_i \beta_i = 0 \\ & 0 \le \beta_i \le \frac{1}{l} \;\; i = 1, \ldots, l \end{aligned} \tag{1.13}$$

where the equality constraint arises due to an unregularized bias term that is often added to the sum in Eqn (1.11), and the following notation is used :

$$\begin{aligned} Y &= diag(y_1, y_2, ..., y_l) \\ Q &= Y\left(\frac{K}{2\gamma}\right)Y \\ \boldsymbol{\alpha}^\star &= \frac{Y\boldsymbol{\beta}^\star}{2\gamma} \end{aligned} \tag{1.14}$$

Here again, K is the gram matrix over labeled points. SVM practitioners may be familiar with a slightly different parameterization involving the C parameter : $C = \frac{1}{2\gamma l}$ is the weight on the hinge loss term (instead of using a weight γ on the norm term in the optimization problem). The C parameter appears as the upper bound (instead of $\frac{1}{l}$) on the values of β in the quadratic program. For additional details on the derivation and alternative formulations of SVMs, see [77].

In the context of this thesis, we make the following note regarding SVMs. Though the traditional presentation of SVMs follows a dual approach, this is not strictly necessary. For instance, if the squared hinge loss is used as the loss function, instead of passing to the dual, one can directly approach the problem in the primal and solve the optimization using primal techniques such as Newton's method or its variants.

Unlike the hinge loss, the squared hinge loss is differentiable and more amenable to unconstrained primal approaches. For recent work on primal SVM formulations, see [23, 62].

1.2.5 Unsupervised Learning

Unsupervised learning algorithms, broadly speaking, attempt to extract some useful structure from a set of unlabeled training examples $D = \{\boldsymbol{x}_i\}_{i=1}^{u}$. An important problem in this category is that of *clustering* where a set of objects need to be partitioned into clusters, such that objects within a cluster are similar, and objects in different clusters are dissimilar (based on some notion of similarity). In this thesis, we will be concerned with a family of clustering algorithms based on spectral cuts on graph representations of data.

Another significant unsupervised learning problem is that of data representation and dimensionality reduction. Such problems arise in many current applications that generate very high dimensional data. In most such applications, more compact low-dimensional representations of the data may be found without losing much information. A very general setup of this problem is when data resides on a low-dimensional manifold embedded in a high-dimensional ambient space. Such a structure may exist in processes generating high dimensional data, but possesses few degrees of freedom. Below we discuss spectral algorithms that construct low-dimensional representation of data generated from an underlying manifold.

Spectral Clustering

Consider a set $U = \{\boldsymbol{x}_i\}_{i=1}^{u}$ of u points that need to be partitioned into two clusters. Also given is a $u \times u$ similarity matrix W so that W_{ij} is a non-negative measure

of similarity between the objects $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. One may consider this information as a weighted, undirected, graph whose vertices are data points and whose edges are weighted by similarity ($W_{ij} = 0$ if no there is no edge between $\boldsymbol{x}_i, \boldsymbol{x}_j$). An optimal partitioning puts similar objects in the same cluster and dissimilar objects in different clusters. Consider an indicator vector $q \in \{-1, +1\}^u$ that defines a partitioning of the vertices into two sets $U(q,-) = \{\boldsymbol{x}_i \in U | q(i) = -1\}$ and $U(q,+) = \{\boldsymbol{x}_i \in U | q(i) = 1\}$. Then with every such partitioning, one can associate the following cost that measures the total similarity of pairs of points belonging to different clusters :

$$J(q) = \sum_{\substack{\boldsymbol{x}_i \in U(q,-) \\ \boldsymbol{x}_j \in U(q,+)}} W_{ij} = \frac{1}{4} \sum_{ij} W_{ij}[q_i - q_j]^2$$

Let $D = diag(d_1, ..., d_u)$ be a diagonal matrix where $d_i = \sum_{j=1}^{u} W_{ij}$ be the degree of vertex x_i. Then the above quantity may be written as :

$$\begin{aligned} J(q) &= \frac{1}{4} \sum_{ij} W_{ij}[q_i - q_j]^2 \\ &= \frac{1}{2} q^T (D - W) q \\ &= \frac{1}{2} q^T L q \end{aligned} \tag{1.15}$$

where L is the *Graph Laplacian* is defined as the matrix $L = D - W$. Note that the constant indicator vector of all-ones or all-minus-ones gives a trivial partition. The *Graph Min-Cut* problem is to find *a non-trivial partition* indicator vector q such that $J(q)$ is minimized. Min-cut is easily solved in polynomial time but there is no control on the sizes of the two clusters obtained.

To get a balanced cut, so that the two clusters $U(q,-)$ and $U(q,+)$ are of the same size, one can add the constraint (assume the number of objects u is even) $\sum_i q_i = 0$

or $q^T \perp \mathbf{1}$ where $\mathbf{1}$ is a u-dimensional vector of ones.

However, this constraint makes the problem NP-complete.

A relaxation of the indicator vectors from $q \in \{-1, +1\}^u$ to $q \in \mathcal{R}^u$ leads to an elegant solution. We need to further impose the condition that $q^T q = 1$ to avoid reducing cost by simply downscaling values of q. Thus the problem becomes :

$$q^* = \underset{q \in \mathcal{R}^u:\ q^T q = 1,\ q \perp \mathbf{1}}{\operatorname{argmin}} q^T L q$$

We note some simple properties of the Graph Laplacian that provide the solution to this problem.

1. L is a symmetric positive-semidefinite matrix. Symmetry is clear by the definition of L; positive definiteness can be seen from the observation the for any $\boldsymbol{x} \in \mathcal{R}^u$, $\boldsymbol{x}^T L \boldsymbol{x} = \sum_{ij} W_{ij} [\boldsymbol{x}_i - \boldsymbol{x}_j]^2 \geq 0$.

2. $q_1 = \mathbf{1}$ is the first eigenvector of L with eigenvalue 0. This can be seen by observing that $L\mathbf{1} = (D - W)\mathbf{1} = D - D = 0$.

3. The second eigenvector q_2 (the so called *Fiedler vector*) is the minimizer of 1.2.5. This can be seen by applying the Courant-Fischer theorem that gives a variational characterization of eigenvalues of a symmetric matrix (say A): The k^{th} eigenvector of A is given by

$$q_k = \underset{\substack{q \perp q_1, q_2, .. q_{k-1} \\ q^T q = 1}}{\operatorname{argmin}} q^T A q$$

and the k^{th} eigenvalue λ_k is the value of the quadratic form at the minimum. In our case, the orthogonality to $\mathbf{1}$ constraint easily fits this theorem since $\mathbf{1}$ is the first eigenvector of L.

Algorithm : The spectral clustering algorithm is simply the following : Given a set of u unlabeled points and a $u \times u$ similarity matrix W, construct the graph laplacian $L = D - W$ where D is a diagonal matrix $D_{ii} = \sum_i W_{ij}$. Compute its second eigenvector $q_2 \in \mathcal{R}^u$. The clustering is given by the sign of q_2 (after thresholding so that cluster sizes are balanced; the objective function is invariant under additions of a constant to q).

Extensions of this basic algorithm have been proposed that involve other objective functions that differ in the way they balance cluster sizes or intra and inter-cluster similarities. These include the Ratio Cut method [52], Normalized Cuts [80] and MinMaxCut [41]. For a tutorial overview, see [101].

Note that the solution of spectral clustering is a bi-partitioning of the set $\{x_i\}_{i=1}^u$. The partitioning indicator function is only defined on this set. Consequently, spectral clustering cannot deal with a new test example that needs to be assigned to its cluster. This is the *problem of out-of-sample extension* in spectral clustering.

Manifold Learning

Among many goals of manifold learning algorithms, we focus on the problem of recovering a low dimensional representation of high dimensional data sampled from an underlying low dimensional manifold $\mathcal{M}$. Algorithmic approaches to this problem attempt to construct a map from this high dimensional data into a single global coordinate system of lower dimensionality that optimally preserves geometric relationships between points on the manifold. Ideally, the lower dimensions correspond to, or allow the identification of the intrinsic degrees of freedom in the the data generating process.

Figure 1.6 shows what this might mean : Points on a 2-dimensional manifold

embedded in 3-dimensions (left panel) may be mapped by a manifold learner to a 2-dimensional space (right panel). Here, the points are laid out as they would be if the manifold was "unfolded". The intrinsic distance relationships are preserved by the map.

In such a situation, classical linear techniques like principal component analysis (PCA) and multi-dimensional scaling may map ambiently close, but intrinsically faraway points, to nearby points in their embeddings. This creates distortions in the local and global geometry of the data. PCA involves projecting data onto directions of maximum variance using top eigenvectors of the data covariance matrix. MDS projects data to a low-dimensional space where pairwise ambient distances are best preserved. These methods work best when the underlying manifold is a linear subspace of the high-dimensional ambient space.

Figure 1.6: Manifold Learning

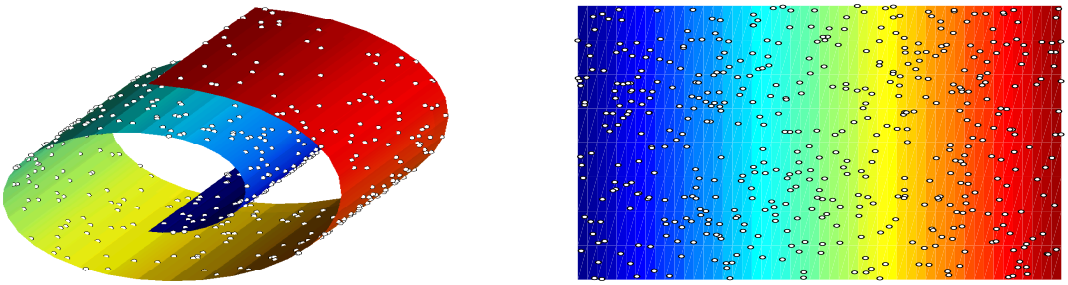
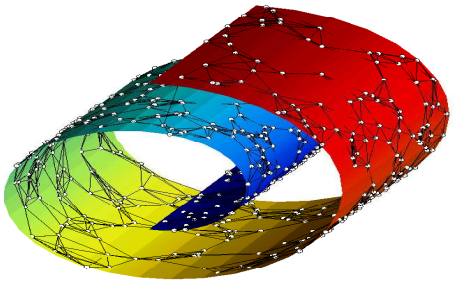


Figure 1.7: Graph Approximation to a Manifold



Non-linear manifold learning techniques have recently been proposed that overcome such shortcomings. We discuss Isomap [56], Locally Linear Embedding [76] and Laplacian Eigenmaps [4] briefly. When given a data set $\{\boldsymbol{x}_i\}_{i=1}^{u}$, these methods begin by constructing a nearest neighbor graph $G(V, E)$. This graph serves as an approximation to the true manifold since locally the ambient distance may be a good estimate of the intrinsic distance. The vertices V of this graph are data points; an edge $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in E$ if the data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are among the nearest neighbors on one another; and the edges may have weights specified by a matrix W_{ij} ($= 0$ if no edge between $\boldsymbol{x}_i, \boldsymbol{x}_j$). Figure 1.7 shows a graph approximation to a manifold.

Isomap : This method applies classical multidimensional scaling on approximations of intrinsic geodesic distances between pairs of points. These estimates are made by computing the shortest path distances in the graph G (whose edges weights are the ambient distances), using e.g Dijkstra's algorithm. MDS then constructs a map, preserving pairwise shortest path distances, to a low-dimensional coordinate system that represents the global structure of the data. However, the requirement of computation of all pairwise distances makes this algorithm somewhat computationally expensive.

Locally Linear Embedding (LLE) : The objective of this algorithm is to recover the global non-linear structure from locally linear approximations to the manifold. Pairwise distance computation for all points is not required. The intuition of the algorithm is that a point and its nearest neighbors span a locally linear space on the manifold, so that each point can be linearly reconstructed in the span of its neighbors. The algorithm first finds the linear coefficients that minimize a reconstruction cost function, and then uses these coefficients to reconstruct low-dimensional coordinates. This involves solving a constrained least squares problem followed by a sparse eigenvalue problem.

Laplacian Eigenmaps : This algorithm looks for optimal embeddings in the sense of best preserving locality. Given the u points in $\mathcal{R}^n$, consider embedding the graph into an m-dimensional space where $m < n$. Let the new coordinates of the i^{th} point be f_i. Then, locality may be best preserved by minimizing $\sum_{i,j=1}^{u} \|f_i - f_j\|^2 W_{ij} = trace(FLF^T)$ over the $m \times u$ matrix F whose columns are $f_1, .., f_u$. Here, L is the graph laplacian. The matrix of eigenvectors corresponding to the lowest eigenvalues of the generalized eigenvalue problem $Lf = \lambda Df$ provide the optimal Y. This method is very closely related to spectral clustering.

For the case of manifold learning, the graph laplacian L, makes an interesting correspondence with the continuous Laplace-Beltrami operator $\mathcal{L}$ on a manifold $\mathcal{M}$ which defines a measure of smoothness $S_{\mathcal{M}}$ with respect to the manifold:

$$S_{\mathcal{M}} = \int_{\mathcal{M}} \|\nabla_{\mathcal{M}} f\|^2 = \int_{\mathcal{M}} f\mathcal{L}[f]$$

where $\nabla_M f$ is the gradient of a function $f : \mathcal{M} \subset \mathcal{R}^n \mapsto \mathcal{R}$ on the manifold, defined at any point $x \in \mathcal{M}$ as the vector in the tangent space T_x at x, such that the directional derivative $df(v)$ along any direction $v \in T_x$ in the tangent space is given by $df(v) = \langle \nabla_{\mathcal{M}} f, v \rangle_{\mathcal{M}}$. The above functional has a discrete approximation to a smoothness functional S_G on the neighborhood graph defined by the Graph Laplacian: $S_G = \sum_{i=1}^{u} (f_i - f_j)^2 W_{ij} = \mathbf{f}^T L\mathbf{f}$ where $\mathbf{f} = (f_1, ..., f_u) \in \mathcal{R}^u$ is a function defined on the graph vertices, f_i specifying the value of the function on vertex $\boldsymbol{x}_i$.

As in Spectral clustering, here too we have the *problem of out-of-sample extension*: how should a novel unseen point be mapped ?

1.3 Semi-supervised Learning in Pictures

We start by providing some intuitions for semi-supervised learning. These intuitions are demonstrated in pictures (Figures 1.8, 1.9 and 1.10).

Consider first the two labeled points (marked "+" and "-") in the left panel of Figure 1.8. Our intuition may suggest that a simple linear separator such as the one shown in Figure 1.8, is an optimal choice for a classifier. Indeed, considerable effort in learning theory has been invested into deriving optimality properties for such a classification boundary.

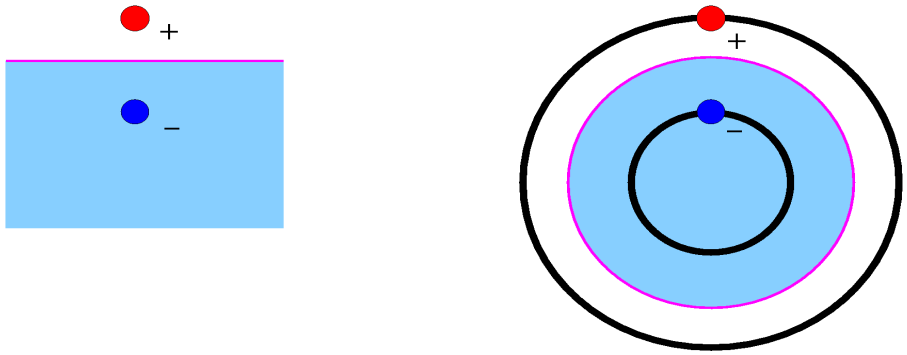


Figure 1.8: Circle

Figure 1.9: Curve

The right panel however shows that the two labeled points are in fact located on two concentric circles of unlabeled data. Looking at the right panel, it becomes clear that the circular boundary is more natural given unlabeled data.

Figure 1.10: Blobs

Consider now the left panel in Figure 1.9. In the absence of unlabeled data the black dot (marked "?") is likely to be classified as blue (marked "-"). The unlabeled data, however, makes classifying it as red (marked "+") seem much more reasonable.

A third example is shown in Figure 1.10. In the left panel, the unlabeled point may be classified as blue (-) to agree with its nearest neighbor. However, unlabeled data shown as grey clusters in the right panel changes our belief.

These examples show how the geometry of unlabeled data may radically change our intuition about classifier boundaries.

Recall now the standard setting of learning from examples. Given a pattern space $\mathcal{X}$, there is a probability distribution $\mathcal{P}$ on $\mathcal{X} \times \mathcal{R}$ according to which examples are generated for function learning. Labeled examples are $(\boldsymbol{x}, y)$ pairs drawn according to $\mathcal{P}$. Unlabeled examples are simply $\boldsymbol{x} \in \mathcal{X}$ sampled according to the marginal distribution $\mathcal{P}_{\mathcal{X}}$ of $\mathcal{P}$.

As we have seen, the knowledge of the marginal $\mathcal{P}_X$ can be exploited for better function learning (e.g., in classification or regression tasks). On the other hand, if there is no identifiable relation between $\mathcal{P}_X$ and the conditional $\mathcal{P}(y|\boldsymbol{x})$, the knowledge of $\mathcal{P}_X$ is unlikely to be of use.

Two possible connections between $\mathcal{P}_X$ and $\mathcal{P}(y|\boldsymbol{x})$ can be stated as the following important assumptions:

1. **Manifold Assumption:** Suppose that the marginal probability distribution underlying the data is supported on a low-dimensional manifold. Then the family of conditional distributions $P(y|\boldsymbol{x})$ is smooth, as a function of $\boldsymbol{x}$, with respect to the underlying structure of the manifold.

2. **Cluster assumption:** The probability distribution $\mathcal{P}$ is such that points in the same "cluster" are likely to have the same label.

We see that the data shown in Figures 1.8 and 1.9 satisfy the manifold assumption.

The picture in Figure 1.10 is meant to show Gaussian clusters. The concentric circles in Figure 1.8 can also be thought as "clusters", although such clusters are highly non-Gaussian and have an interesting geometric structure. One may conjecture that many clusters in real-world datasets have such non-Gaussian structures.

In this thesis, we seek to translate the above intuitions into algoritmic frameworks by implementing manifold and cluster assumptions within Kernel-based Regularization Methods.

1.4 Contributions of this Thesis

In Chapter 2, we outline a graph-based framework called Manifold Regularization for incorporating unlabeled data in a wide variety of Kernel methods, motivated primarily by the manifold assumption. This framework naturally resolves the problem of of out-of-sample extension in Graph transduction, Spectral Clustering and graph-based manifold learning methods. We provide an interpretation in terms of globally defined

data-dependent kernels. This chapter is drawn from the following published papers [7, 86, 81] coauthored with Partha Niyogi and Misha Belkin.

In Chapter 3, we revisit an early framework proposed by V. Vapnik [96] that leads to algorithms such as the Transductive SVM (TSVM) where one optimizes also over unknown labels explicitly to enforce the cluster assumption. TSVMs lead to a non-convex optimization problem and show variable empirical performance on real world datasets. Our contribution is two-fold: we implement a global optimization framework based on a Branch and Bound algorithm to benchmark current implementations and investigate the severity of their susceptibility to local minima. It turns out that the global solution can return extremely good performance where practical implementations fail completely. However, our Branch-and-bound method is only applicable to small datasets. We develop a deterministic annealing heuristic that shows some resistance to suboptimal local minima. This chapter is drawn from the following published papers [84, 27] coauthored with Sathiya Keerthi and Olivier Chapelle.

In Chapter 4, we propose simple regularization algorithms that arise from the following idea first introduced in [18]: If multiple redundant representations of the data is available, then classifiers developed on each should agree in their predictions on the unlabeled examples. A consensus maximization term is added to a joint regularization functional. This framework leads to algorithms that are applicable when different sources of information (features) need to be combined.

In Chapter 5, we study the issue of sparsifying the Manifold Regularization solution with the goal of making training and testing more effecient. Instead of optimizing the objective over an entire RKHS, we restrict our attention to a smalled subspace. This subspace can be constructed in various ways: random subset selection, greedy incremental methods (matching pursuit), l_1 regularization etc. We empirically show that a straightforward application of sparse methods does not lead to satisfactory

performance. Tightening the Manifold regularizer through a sigmoid function or introducing extra degrees of freedom is necessary to recover effective sparse solutions.

In Chapter 6, we address an important special case of the algorithms developed in this thesis. On many applications, linear classifiers are strongly preferred due to their performance and simplicity. We develop semi-supervised linear classifiers that can handle millions of examples and features by exploiting the sparsity of the data-matrix. This chapter is drawn from [85] co-authored with Sathiya Keerthi.

CHAPTER 2

MANIFOLD REGULARIZATION: A GEOMETRIC FRAMEWORK

In this chapter, we present an algorithmic framework for semi-supervised inference based on geometric properties of probability distributions. Our approach brings together Laplacian-based spectral techniques, regularization with kernel methods, and algorithms for manifold learning. This framework provides a natural semi-supervised extension for kernel methods and resolves the problem of out-of-sample inference in graph-based transduction. We discuss an interpretation in terms of a family of globally defined data-dependent kernels and also address unsupervised learning (clustering and data representation) within the same framework. Our algorithms effectively exploit both manifold and cluster assumptions to demonstrate state-of-the-art performance on various classification tasks.

2.1 Introduction

In this chapter, we introduce a framework for data-dependent regularization that exploits the geometry of the probability distribution. While this framework allows us to approach the full range of learning problems from unsupervised to supervised, we focus on the problem of semi-supervised learning.

The problem of learning from labeled and unlabeled data (*semi-supervised* and *transductive* learning) has attracted considerable attention in recent years. Some recently proposed methods include Transductive SVM [97, 57], Cotraining [18], and a variety of graph based methods [17, 29, 91, 65, 89, 108, 110, 112, 63, 59, 2]. We also note the regularization based techniques of [37] and [20]. The latter reference

is closest in spirit to the intuitions in this chapter. We postpone the discussion of related algorithms and various connections until Section 2.6.

As discussed in the previous chapter, the idea of regularization has a rich mathematical history going back to [94], where it is used for solving ill-posed inverse problems. Regularization is a key idea in the theory of splines (e.g., [103]) and is widely used in machine learning (e.g., [47]). Recall that many machine learning algorithms, including Support Vector Machines, can be interpreted as instances of regularization.

Our framework exploits the geometry of the probability distribution that generates the data and incorporates it as an additional regularization term. Hence, there are two regularization terms — one controlling the complexity of the classifier in the *ambient space* and the other controlling the complexity as measured by the *geometry* of the distribution. We consider in some detail the special case where this probability distribution is supported on a submanifold of the ambient space.

The points below highlight several aspects of the current chapter:

1. Our general framework brings together three distinct concepts that have received some independent recent attention in machine learning:

   i. The first of these is the technology of *spectral graph theory* (e.g., see [31]) that has been applied to a wide range of clustering and classification tasks over the last two decades. Such methods typically reduce to certain eigenvalue problems.

   ii. The second is the geometric point of view embodied in a class of algorithms that can be termed as *manifold learning*[1]. These methods attempt to use the geometry of the probability distribution by assuming that its support has the geometric structure of a Riemannian manifold.

   iii. The third important conceptual framework is the set of ideas surround-

1. see `http://www.cse.msu.edu/~lawhiu/manifold/` for a list of references

ing regularization in Reproducing Kernel Hilbert Spaces (RKHS). This leads to the class of *kernel based algorithms* for classification and regression (e.g., see [78, 103, 47]).

We show how these ideas can be brought together in a coherent and natural way to incorporate geometric structure in a kernel based regularization framework. As far as we know, these ideas have not been unified in a similar fashion before.

2. This general framework allows us to develop algorithms spanning the range from unsupervised to fully supervised learning.

   In this chapter, we primarily focus on the semi-supervised setting and present the following algorithms: the Laplacian Regularized Least Squares (hereafter LapRLS) and the Laplacian Support Vector Machines (hereafter LapSVM). These are natural extensions of RLS and SVM respectively. In addition, several recently proposed transductive methods (e.g., [111, 5]) are also seen to be special cases of this general approach.

   In the absence of labeled examples our framework results in new algorithms for unsupervised learning, which can be used both for data representation and clustering. These algorithms are related to Spectral Clustering and Laplacian Eigenmaps [8].

3. We elaborate on the RKHS foundations of our algorithms and show how geometric knowledge of the probability distribution may be incorporated in such a setting through an additional regularization penalty. In particular, a new Representer theorem provides a functional form of the solution when the distribution is known; its empirical version involves an expansion over labeled and unlabeled points when the distribution is unknown. These Representer theorems provide

the basis for our algorithms.

4. Our framework with an ambiently defined RKHS and the associated Representer theorems result in a natural out-of-sample extension from the data set (labeled and unlabeled) to novel examples. This is in contrast to the variety of purely graph based approaches that have been considered in the last few years. Such graph based approaches work in a transductive setting and do not naturally extend to the semi-supervised case where novel test examples need to be classified (predicted). Also see [9, 21] for some recent related work on out-of-sample extensions. We also note that a method similar to our regularized spectral clustering algorithm has been independently proposed in the context of graph inference in [99].

5. We re-interpret our framework in terms of a family of data-dependent norms on Reproducing Kernel Hilbert Spaces (RKHS). These norms allow us to warp the structure of a base RKHS to reflect the underlying geometry of the data. We derive explicit formulas for the corresponding new kernels. Standard supervised learning with this new kernel effectively perform semi-supervised learning. The solution is expressed in terms of the classical standard Representer theorem involving the modified kernel functions centered only the labeled points. This interpretation expands the algorithmic scope of our framework: all kernel methods (such as Support Vector Regression, one-class SVMs etc) can utilize this construction. See [83] for the construction of semi-supervised Gaussian processes in this manner.

2.2 Incorporating Geometry in Regularization

Recall the standard framework of learning from examples. There is a probability distribution $\mathcal{P}$ on $\mathcal{X} \times \Re$ according to which examples are generated for function learning. Labeled examples are $(\boldsymbol{x}, y)$ pairs generated according to P. Unlabeled examples are simply $x \in \mathcal{X}$ drawn according to the marginal distribution $\mathcal{P}_{\mathcal{X}}$ of $\mathcal{P}$.

One might hope that knowledge of the marginal $\mathcal{P}_{\mathcal{X}}$ can be exploited for better function learning (e.g. in classification or regression tasks). Of course, if there is no identifiable relation between $\mathcal{P}_{\mathcal{X}}$ and the conditional $\mathcal{P}(y|\boldsymbol{x})$, the knowledge of $\mathcal{P}_{\mathcal{X}}$ is unlikely to be of much use.

Therefore, we will make a specific assumption about the connection between the marginal and the conditional distributions. We will assume that if two points $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ are *close* in the *intrinsic* geometry of $\mathcal{P}_{\mathcal{X}}$, then the conditional distributions $\mathcal{P}(y|\boldsymbol{x}_1)$ and $\mathcal{P}(y|\boldsymbol{x}_2)$ are similar. In other words, the conditional probability distribution $\mathcal{P}(y|\boldsymbol{x})$ varies smoothly along the geodesics in the intrinsic geometry of $\mathcal{P}_{\mathcal{X}}$.

We utilize these geometric intuitions to extend an established framework for function learning. A number of popular algorithms such as SVM, Ridge regression, splines, Radial Basis Functions may be broadly interpreted as regularization algorithms with different empirical cost functions and complexity measures in an appropriately chosen Reproducing Kernel Hilbert Space (RKHS).

For a Mercer kernel $K : \mathcal{X} \times \mathcal{X} \to \mathcal{R}$, there is an associated RKHS $\mathcal{H}_K$ of functions $\mathcal{X} \to \Re$ with the corresponding norm $\| \ \|_K$. Given a set of labeled examples $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, l$ the standard framework estimates an unknown function by minimizing

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(\boldsymbol{x}_i, y_i, f) + \gamma \|f\|_K^2 \tag{2.1}$$

where V is some loss function, such as squared loss $(y_i - f(\boldsymbol{x}_i))^2$ for RLS or the hinge loss function $\max[0, 1 - y_i f(\boldsymbol{x}_i)]$ for SVM. Penalizing the RKHS norm imposes smoothness conditions on possible solutions. The classical Representer Theorem states that the solution to this minimization problem exists in $\mathcal{H}_K$ and can be written as

$$f^*(\boldsymbol{x}) = \sum_{i=1}^{l} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) \tag{2.2}$$

Therefore, the problem is reduced to optimizing over the finite dimensional space of coefficients α_i, which is the algorithmic basis for SVM, Regularized Least Squares and other regression and classification schemes.

We first consider the case when the marginal distribution is already known.

2.2.1 Marginal $\mathcal{P}_\mathcal{X}$ is known

Our goal is to extend this framework by incorporating additional information about the geometric structure of the marginal $\mathcal{P}_\mathcal{X}$. We would like to ensure that the solution is smooth with respect to both the ambient space and the marginal distribution $\mathcal{P}_\mathcal{X}$. To achieve that, we introduce an additional regularizer:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(\boldsymbol{x}_i, y_i, f) + \gamma_A \|f\|_K^2 + \gamma_I \|f\|_I^2 \tag{2.3}$$

where $\|f\|_I^2$ is an appropriate penalty term that should reflect the intrinsic structure of $\mathcal{P}_\mathcal{X}$. Intuitively, $\|f\|_I^2$ is a smoothness penalty corresponding to the probability distribution. For example, if the probability distribution is supported on a low-dimensional manifold, $\|f\|_I^2$ may penalize f along that manifold. γ_A controls the complexity of the function in the *ambient* space while γ_I controls the complexity of the function in the *intrinsic* geometry of $\mathcal{P}_\mathcal{X}$. It turns out that one can derive an

explicit functional form for the solution f^* as shown in the following theorem.

Theorem 2.2.1. *Assume that the intrinsic regularization term is given by:*

$$\|f\|_I^2 = \int_{\mathcal{X}} fDf d\mathcal{P}_{\mathcal{X}}$$

where D is a bounded operator from the RKHS associated to K to $L^2(\mathcal{P}_{\mathcal{X}})$. Then the solution f^* to the optimization problem in Eqn. 2.3 above exists and admits the following representation:

$$f^*(\boldsymbol{x}) = \sum_{i=1}^{l} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) + \int_{\mathcal{X}} \alpha(\boldsymbol{z}) K(\boldsymbol{x}, \boldsymbol{z})\, d\mathcal{P}_{\mathcal{X}}(\boldsymbol{z}) \tag{2.4}$$

where $\mathcal{M} = \text{supp}\{\mathcal{P}_{\mathcal{X}}\}$ is the support of the marginal $\mathcal{P}_{\mathcal{X}}$.

For a proof we refer the reader to [7].

The Representer Theorem above allows us to express the solution f^* directly in terms of the labeled data, the (ambient) kernel K, and the marginal $\mathcal{P}_{\mathcal{X}}$. If $\mathcal{P}_{\mathcal{X}}$ is unknown, we see that the solution may be expressed in terms of an empirical estimate of $\mathcal{P}_{\mathcal{X}}$. Depending on the nature of this estimate, different approximations to the solution may be developed. In the next section, we consider a particular approximation scheme that leads to a simple algorithmic framework for learning from labeled and unlabeled data.

2.2.2 Marginal $\mathcal{P}_{\mathcal{X}}$ Unknown

In most applications the marginal $\mathcal{P}_{\mathcal{X}}$ is not known. Therefore we must attempt to get empirical estimates of $\mathcal{P}_{\mathcal{X}}$ and $\| \ \|_I$. Note that in order to get such empirical estimates it is sufficient to have *unlabeled* examples.

A case of particular recent interest (e.g., see [76, 92, 8, 43, 34] for a discussion on dimensionality reduction) is when the support of $\mathcal{P}_X$ is a compact submanifold $\mathcal{M} \subset \mathcal{R}^n$. In that case, one natural choice for $\|f\|_I$ is $\int_{x \in \mathcal{M}} \|\nabla_{\mathcal{M}} f\|^2 \, d\mathcal{P}_X(x)$, where $\nabla_{\mathcal{M}}$ is the *gradient* (see, e.g., [42] for an introduction to differential geometry) of f along the manifold $\mathcal{M}$ and the integral is taken over the marginal distribution.

The optimization problem becomes

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(\boldsymbol{x}_i, y_i, f) + \gamma_A \|f\|_K^2 + \gamma_I \int_{\boldsymbol{x} \in \mathcal{M}} \|\nabla_{\mathcal{M}} f\|^2 \, d\mathcal{P}_X(\boldsymbol{x})$$

The term $\int_{\boldsymbol{x} \in \mathcal{M}} \|\nabla_{\mathcal{M}} f\|^2 \, d\mathcal{P}_{\mathcal{X}}(\boldsymbol{x})$ may be approximated on the basis of labeled and unlabeled data using the graph Laplacian associated to the data. While an extended discussion of these issues goes beyond the scope of this work, it can be shown that under certain conditions choosing exponential weights for the adjacency graph leads to convergence of the graph Laplacian to the Laplace-Beltrami operator $\Delta_{\mathcal{M}}$ (or its weighted version) on the manifold. See the Remarks below and [1, 67, 6, 33, 54] for details.

Thus, given a set of l labeled examples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{l}$ and a set of u unlabeled examples $\{\boldsymbol{x}_j\}_{j=l+1}^{j=l+u}$, we consider the following optimization problem :

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(\boldsymbol{x}_i, y_i, f) + \gamma_A \|f\|_K^2 + \frac{\gamma_I}{(u+l)^2} \sum_{i,j=1}^{l+u} (f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j))^2 W_{ij}$$

$$= \operatorname*{argmin}_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(\boldsymbol{x}_i, y_i, f) + \gamma_A \|f\|_K^2 + \frac{\gamma_I}{(u+l)^2} \mathbf{f}^T L \mathbf{f} \tag{2.5}$$

where W_{ij} are edge weights in the data adjacency graph, $\mathbf{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_{l+u})]^T$, and L is the graph Laplacian given by $L = D - W$. Here, the diagonal matrix D

is given by $D_{ii} = \sum_{j=1}^{l+u} W_{ij}$. The normalizing coefficient $\frac{1}{(u+l)^2}$ is the natural scale factor for the empirical estimate of the Laplace operator. We note than on a sparse adjacency graph it may be replaced by $\sum_{i,j=1}^{l+u} W_{ij}$.

The following version of the Representer Theorem shows that the minimizer has an expansion in terms of both labeled and unlabeled examples and is a key to our algorithms.

Theorem 2.2.2. *The minimizer of optimization problem 2.5 admits an expansion*

$$f^*(\boldsymbol{x}) = \sum_{i=1}^{l+u} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) \tag{2.6}$$

in terms of the labeled and unlabeled examples.

The proof is a variation of the standard orthogonality argument and is presented in Section 2.2.3.

Remark 1: Several natural choices of $\| \ \|_I$ exist. Some examples are:

1. Iterated Laplacians $(\Delta_{\mathcal{M}})^k$. Differential operators $(\Delta_{\mathcal{M}})^k$ and their linear combinations provide a natural family of smoothness penalties.

   Recall that the Laplace-Beltrami operator $\Delta_{\mathcal{M}}$ can be defined as the divergence of the gradient vector field $\Delta_{\mathcal{M}} f = \operatorname{div}(\nabla_{\mathcal{M}} f)$ and is characterized by the equality

   $$\int_{\boldsymbol{x} \in \mathcal{M}} f(x) \Delta_{\mathcal{M}} f(\boldsymbol{x}) d\mu = \int_{\boldsymbol{x} \in \mathcal{M}} \|\nabla_{\mathcal{M}} f(\boldsymbol{x})\|^2 \, d\mu$$

   where μ is the standard measure (uniform distribution) on the Riemannian manifold. If μ is taken to be non-uniform, then the corresponding notion is the weighted Laplace-Beltrami operator (e.g., [51]).

2. Heat semigroup $e^{-t\Delta_{\mathcal{M}}}$ is a family of smoothing operators corresponding to the process of diffusion (Brownian motion) on the manifold. One can take $\|f\|_I^2 = \int_{\mathcal{M}} f\, e^{t\Delta_{\mathcal{M}}}(f) d\mathcal{P}_{\mathcal{X}}$. We note that for small values of t the corresponding Green's function (the heat kernel of $\mathcal{M}$), which is close to a Gaussian in the geodesic coordinates, can also be approximated by a sharp Gaussian in the ambient space.

3. Squared norm of the Hessian (cf. [43]). While the Hessian $\mathbf{H}(f)$ (the matrix of second derivatives of f) generally depends on the coordinate system, it can be shown that the Frobenius norm (the sum of squared eigenvalues) of $\mathbf{H}$ is the same in any geodesic coordinate system and hence is invariantly defined for a Riemannian manifold $\mathcal{M}$. Using the Frobenius norm of $\mathbf{H}$ as a regularizer presents an intriguing generalization of thin-plate splines. We also note that $\Delta_{\mathcal{M}}(f) = \mathrm{tr}(\mathbf{H}(f))$.

Remark 2: Why not just use the intrinsic regularizer? Using ambient and intrinsic regularizers jointly is important for the following reasons:

1. We do not usually have access to $\mathcal{M}$ or the true underlying marginal distribution, just to data points sampled from it. Therefore regularization with respect only to the sampled manifold is ill-posed. By including an ambient term, the problem becomes well-posed.

2. There may be situations when regularization with respect to the ambient space yields a better solution, e.g., when the manifold assumption does not hold (or holds to a lesser degree). Being able to trade off these two regularizers may be important in practice.

Remark 3: While we use the graph Laplacian for simplicity, the *normalized Lapla-*

cian

$$\tilde{L} = D^{-1/2} L D^{-1/2}$$

can be used interchangeably in all our formulas. Using $\tilde{L}$ instead of L provides certain theoretical guarantees (see [102]) and seems to perform as well or better in many practical tasks. In fact, we use $\tilde{L}$ in all our empirical studies in Section 2.7. The relation of $\tilde{L}$ to the weighted Laplace-Beltrami operator was discussed in [67].

Remark 4: Note that a global kernel K restricted to $\mathcal{M}$ (denoted by $K_{\mathcal{M}}$) is also a kernel defined on $\mathcal{M}$ with an associated RKHS $\mathcal{H}_{\mathcal{M}}$ of functions $\mathcal{M} \to \mathcal{R}$. While this might suggest

$$\|f\|_I = \|f_{\mathcal{M}}\|_{K_{\mathcal{M}}}$$

($f_{\mathcal{M}}$ is f restricted to $\mathcal{M}$) as a reasonable choice for $\|f\|_I$, it turns out, that for the minimizer f^* of the corresponding optimization problem we get $\|f^*\|_I = \|f^*\|_K$, yielding the same solution as standard regularization, although with a different parameter γ. This observation follows from the restriction properties of RKHS. Therefore it is impossible to have an out-of-sample extension without two *different* measures of smoothness. On the other hand, a different ambient kernel restricted to $\mathcal{M}$ can potentially serve as the intrinsic regularization term. For example, a sharp Gaussian kernel can be used as an approximation to the heat kernel on $\mathcal{M}$. Thus one (sharper) kernel may be used in conjunction with unlabeled data to estimate the heat kernel on $\mathcal{M}$ and a wider kernel for inference.

2.2.3 The Representer Theorem for the Empirical Case

In the case when $\mathcal{M}$ is unknown and sampled via labeled and unlabeled examples, the Laplace-Beltrami operator on $\mathcal{M}$ may be approximated by the Laplacian of the data

adjacency graph (see [1, 20] for some discussion). A regularizer based on the graph Laplacian leads to the optimization problem posed in Eqn. 2.5. We now provide a proof of Theorem 2.2.2 which states that the solution to this problem admits a representation in terms of an expansion over labeled and unlabeled points. The proof is based on a simple orthogonality argument (e.g., [78]).

Theorem 2.2.3. *Any function $f \in \mathcal{H}_K$ can be uniquely decomposed into a component $f_{||}$ in the linear subspace spanned by the kernel functions $\{K(\boldsymbol{x}_i, \cdot)\}_{i=1}^{l+u}$, and a component $f_\perp$ orthogonal to it. Thus,*

$$f = f_{||} + f_\perp = \sum_{i=1}^{l+u} \alpha_i K(\boldsymbol{x}_i, \cdot) + f_\perp \tag{2.7}$$

By the reproducing property, as the following arguments show, the evaluation of f on any data point $\boldsymbol{x}_j$, $1 \leq j \leq l+u$ is independent of the orthogonal component $f_\perp$:

$$f(\boldsymbol{x}_j) = \langle f, K(\boldsymbol{x}_j, \cdot)\rangle = \langle \sum_{i=1}^{l+u} \alpha_i K(\boldsymbol{x}_i, \cdot), K(\boldsymbol{x}_j, \cdot)\rangle + \langle f_\perp, K(\boldsymbol{x}_j, \cdot)\rangle \tag{2.8}$$

Since the second term vanishes, and $\langle K(\boldsymbol{x}_i, \cdot), K(\boldsymbol{x}_j, \cdot)\rangle = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, it follows that $f(\boldsymbol{x}_j) = \sum_{i=1}^{l+u} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Thus, the empirical terms involving the loss function and the intrinsic norm in the optimization problem in Eqn. 2.5 depend only on the value of the coefficients $\{\alpha_i\}_{i=1}^{l+u}$ and the gram matrix of the kernel function.

Indeed, since the orthogonal component only increases the norm of f in $\mathcal{H}_K$:

$$\|f\|_K^2 = \|\sum_{i=1}^{l+u} \alpha_i K(\boldsymbol{x}_i, \cdot)\|_K^2 + \|f_\perp\|_K^2 \geq \|\sum_{i=1}^{l+u} \alpha_i K(\boldsymbol{x}_i, \cdot)\|_K^2$$

It follows that the minimizer of problem 2.5 must have $f_\perp = 0$, and therefore admits

a representation $f^*(\cdot) = \sum_{i=1}^{l+u} \alpha_i K(\boldsymbol{x}_i, \cdot)$.

The simple form of the minimizer, given by this theorem, allows us to translate our extrinsic and intrinsic regularization framework into optimization problems over the finite dimensional space of coefficients $\{\alpha_i\}_{i=1}^{l+u}$, and invoke the machinery of kernel based algorithms. In the next section, we derive these algorithms, and explore their connections to other related work.

2.3 Algorithms

We now discuss standard regularization algorithms (RLS and SVM) and present their extensions (LapRLS and LapSVM respectively). These are obtained by solving the optimization problems posed in Eqn. (2.5) for different choices of cost function V and regularization parameters γ_A, γ_I. To fix notation, we assume we have l labeled examples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{l}$ and u unlabeled examples $\{\boldsymbol{x}_j\}_{j=l+1}^{j=l+u}$. We use K interchangeably to denote the kernel function or the Gram matrix.

2.3.1 Laplacian Regularized Least Squares (LapRLS)

The Laplacian Regularized Least Squares algorithm solves the optimization problem in Eqn. (2.5) with the squared loss function:

$$\min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} (y_i - f(\boldsymbol{x}_i))^2 + \gamma_A \|f\|_K^2 + \frac{\gamma_I}{(u+l)^2} \mathbf{f}^T L \mathbf{f}$$

As before, the Representer Theorem can be used to show that the solution is an expansion of kernel functions over both the labeled and the unlabeled data :

$$f^\star(\boldsymbol{x}) = \sum_{i=1}^{l+u} \alpha_i^\star K(\boldsymbol{x}, \boldsymbol{x}_i) \tag{2.9}$$

Substituting this form in the equation above, as before, we arrive at a convex differentiable objective function of the $l+u$-dimensional variable $\boldsymbol{\alpha} = [\alpha_1 \dots \alpha_{l+u}]^T$:

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha} \in \mathcal{R}^{l+u}}{\operatorname{argmin}} \frac{1}{l}(Y - JK\boldsymbol{\alpha})^T(Y - JK\boldsymbol{\alpha}) + \gamma_A \boldsymbol{\alpha}^T K \boldsymbol{\alpha} + \frac{\gamma_I}{(u+l)^2}\boldsymbol{\alpha}^T KLK\boldsymbol{\alpha} \quad (2.10)$$

where K is the $(l+u) \times (l+u)$ Gram matrix over labeled and unlabeled points; Y is an $(l+u)$ dimensional label vector given by : $Y = [y_1, \dots, y_l, 0, \dots, 0]$ and J is an $(l+u) \times (l+u)$ diagonal matrix given by $J = \text{diag}(1, \dots, 1, 0, \dots, 0)$ with the first l diagonal entries as 1 and the rest 0.

The derivative of the objective function vanishes at the minimizer :

$$\frac{1}{l}(Y - JK\boldsymbol{\alpha})^T(-JK) + (\gamma_A K + \frac{\gamma_I l}{(u+l)^2} KLK)\boldsymbol{\alpha} = 0 \quad (2.11)$$

which leads to the following solution.

$$\boldsymbol{\alpha}^* = (JK + \gamma_A lI + \frac{\gamma_I l}{(u+l)^2} LK)^{-1} Y \quad (2.12)$$

Note that when $\gamma_I = 0$, Eqn. (2.12) gives zero coefficients over unlabeled data, and the coefficients over the labeled data are exactly those for standard RLS.

2.3.2 Laplacian Support Vector Machines

By including the intrinsic smoothness penalty term, we can extend SVMs by solving the following problem:

$$\min_{f \in \mathcal{H}_K} \frac{1}{l}\sum_{i=1}^{l}(1 - y_i f(\boldsymbol{x}_i))_+ + \gamma_A \|f\|_K^2 + \frac{\gamma_I}{(u+l)^2}\mathbf{f}^T L\mathbf{f} \quad (2.13)$$

By the representer theorem,as before, the solution to the problem above is given by:

$$f^\star(x) = \sum_{i=1}^{l+u} \alpha_i^\star K(x, x_i) \tag{2.14}$$

Often in SVM formulations, an unregularized bias term b is added to the above form. Again, the primal problem can be easily seen to be the following:

$$\begin{aligned} \min_{\boldsymbol{\alpha}\in\mathcal{R}^{l+u},\xi\in\mathcal{R}^l} & \frac{1}{l}\sum_{i=1}^{l}\xi_i + \gamma_A\boldsymbol{\alpha}^T K\boldsymbol{\alpha} + \frac{\gamma_I}{(u+l)^2}\boldsymbol{\alpha}^T KLK\boldsymbol{\alpha} \\ \text{subject to : } & y_i(\sum_{j=1}^{l+u}\alpha_j K(x_i,x_j)+b) \geq 1-\xi_i, \quad i=1,\ldots,l \\ & \xi_i \geq 0 \quad i=1,\ldots,l \end{aligned} \tag{2.15}$$

Introducing the Lagrangian, with β_i, ζ_i as Lagrange multipliers:

$$\begin{aligned} L(\boldsymbol{\alpha},\boldsymbol{\xi},b,\boldsymbol{\beta},\boldsymbol{\zeta}) &= \frac{1}{l}\sum_{i=1}^{l}\xi_i + \frac{1}{2}\boldsymbol{\alpha}^T(2\gamma_A K + 2\frac{\gamma_A}{(l+u)^2}KLK)\boldsymbol{\alpha} \\ &\quad - \sum_{i=1}^{l}\beta_i(y_i(\sum_{j=1}^{l+u}\alpha_j K(x_i,x_j)+b)-1+\xi_i) - \sum_{i=1}^{l}\zeta_i\xi_i \end{aligned} \tag{2.16}$$

Passing to the dual requires the following steps:

$$\begin{aligned}
\frac{\partial L}{\partial b} = 0 &\implies \sum_{i=1}^{l} \beta_i y_i = 0 \\
\frac{\partial L}{\partial \xi_i} = 0 &\implies \frac{1}{l} - \beta_i - \zeta_i = 0 \\
&\implies 0 \leq \beta_i \leq \frac{1}{l} \quad (\xi_i, \zeta_i \text{ are non-negative})
\end{aligned} \tag{2.17}$$

Using above identities, we formulate a reduced Lagrangian:

$$\begin{aligned}
L^R(\boldsymbol{\alpha}, \boldsymbol{\beta}) &= \frac{1}{2}\boldsymbol{\alpha}^T(2\gamma_A K + 2\frac{\gamma_I}{(u+l)^2}KLK)\boldsymbol{\alpha} - \sum_{i=1}^{l}\beta_i(y_i\sum_{j=1}^{l+u}\alpha_j K(x_i, x_j) - 1) \\
&= \frac{1}{2}\boldsymbol{\alpha}^T(2\gamma_A K + 2\frac{\gamma_I}{(u+l)^2}KLK)\boldsymbol{\alpha} - \boldsymbol{\alpha}^T K J^T Y \boldsymbol{\beta} + \sum_{i=1}^{l}\beta_i
\end{aligned} \tag{2.18}$$

where $J = [I \;\; 0]$ is an $l \times (l+u)$ matrix with I as the $l \times l$ identity matrix (assuming the first l points are labeled) and $Y = \text{diag}(y_1, y_2, ..., y_l)$.

Taking derivative of the reduced Lagrangian with respect to α:

$$\frac{\partial L^R}{\partial \boldsymbol{\alpha}} = (2\gamma_A K + 2\frac{\gamma_I}{(u+l)^2}KLK)\boldsymbol{\alpha} - KJ^T Y\boldsymbol{\beta}$$

This implies:

$$\boldsymbol{\alpha} = (2\gamma_A I + 2\frac{\gamma_I}{(u+l)^2}LK)^{-1}J^T Y\boldsymbol{\beta}^\star \tag{2.19}$$

Note that the relationship between $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ is no longer as simple as the SVM algorithm. In particular, the $(l+u)$ expansion coefficients are obtained by solving a linear system involving the l dual variables that will appear in the SVM dual problem.

Substituting back in the reduced Lagrangian we get:

$$\boldsymbol{\beta}^* = \max_{\boldsymbol{\beta}\in\mathcal{R}^l} \sum_{i=1}^{l} \beta_i - \frac{1}{2}\boldsymbol{\beta}^T Q \boldsymbol{\beta} \tag{2.20}$$

$$\text{subject to :} \quad \sum_{i=1}^{l} \beta_i y_i = 0$$

$$0 \leq \beta_i \leq \frac{1}{l} \; i = 1, \ldots, l \tag{2.21}$$

where

$$Q = YJK(2\gamma_A I + 2\frac{\gamma_I}{(l+u)^2} LK)^{-1} J^T Y$$

Laplacian SVMs can be implemented by using a standard SVM solver with the quadratic form induced by the above matrix, and using the solution to obtain the expansion coefficients by solving the linear system in Eqn. (2.19).

Note that when $\gamma_I = 0$, the SVM QP and Eqns. (2.20,2.19), give zero expansion coefficients over the unlabeled data. The expansion coefficients over the labeled data and the Q matrix are as in standard SVM, in this case.

The Manifold Regularization algorithms are summarized in the Table 2.1.

Efficiency Issues:

It is worth noting that our algorithms compute the inverse of a dense Gram matrix which leads to $O((l+u)^3)$ complexity. This may be impractical for large datasets. In the case of linear kernels, instead of using Eqn. 2.6, we can directly write $f^\star(\boldsymbol{x}) = \mathbf{w}^T\boldsymbol{x}$ and solve for the weight vector $\mathbf{w}$ using a primal optimization method. This is much more efficient when the data is low-dimensional. For highly sparse datasets, e.g. in text categorization problems, effective conjugate gradient schemes can be utilized in a large scale implementation. For the non-linear case, one may obtain approximate

Table 2.1: Manifold Regularization Algorithms

Input:	l labeled examples $\{(x_i, y_i)\}_{i=1}^{l}$, u unlabeled examples $\{x_j\}_{j=l+1}^{l+u}$
Output:	Estimated function $f : \mathcal{R}^n \rightarrow \mathcal{R}$
Step 1	▶ Construct data adjacency graph with $(l+u)$ nodes using, e.g, k nearest neighbors or a graph kernel. Choose edge weights W_{ij}, e.g. binary weights or heat kernel weights $W_{ij} = e^{-\|x_i - x_j\|^2/4t}$.
Step 2	▶ Choose a kernel function $K(x, y)$. Compute the Gram matrix $K_{ij} = K(x_i, x_j)$.
Step 3	▶ Compute graph Laplacian matrix : $L = D - W$ where D is a diagonal matrix given by $D_{ii} = \sum_{j=1}^{l+u} W_{ij}$. Construct a graph Regularier e.g., $M = L^p$.
Step 4	▶ Choose γ_A and γ_I.
Step 5	▶ Compute $\boldsymbol{\alpha}^*$ using Eqn. (2.12) for squared loss (Laplacian RLS) or using Eqns. (2.20, 2.19) together with the SVM QP solver for soft margin loss (Laplacian SVM). Equivalently, use the data-dependent modified kernel in Eqn. 2.22 in a standard supervised RLS or SVM (section 1.2.4) to compute $\boldsymbol{\alpha}^\star$.
Step 6	▶ Output function $f^*(\boldsymbol{x}) = \sum_{i=1}^{l+u} \alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x})$. If Eqn. 2.22 is used, output the standard solution $f^*(\boldsymbol{x}) = \sum_{i=1}^{l} \alpha_i^* \tilde{K}(\boldsymbol{x}_i, \boldsymbol{x})$

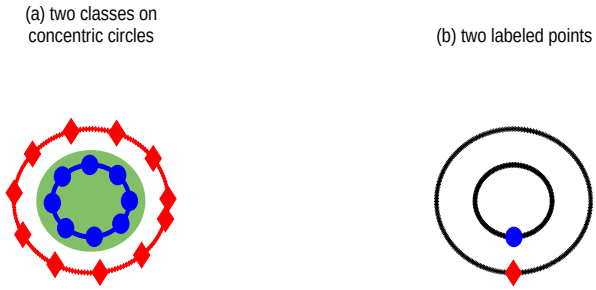
solutions (e.g. using greedy, matching pursuit techniques) where the optimization problem is solved over the span of a small set of basis functions instead of using the full representation in Eqn. 2.6. We investigate these directions in Chapter 5 and Chapter 6. In section 2.7, we evaluate the empirical performance of our algorithms with exact computations as outlined in Table 2.1 with non-linear kernels.

2.4 Data-dependent Kernels

Consider again the picture shown in Fig. 2.1(a). Shown in that figure are two classes of data points in the plane ($\mathcal{R}^2$) such that all data points lie on one of two concentric circles. This represents a two class pattern classification problem where each class is

identified with one of the circles. The decision boundary separating the two classes is non-linear. Let us suppose we use the Gaussian (RBF) kernel $K(\boldsymbol{x}, \boldsymbol{z}) = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{z}\|^2}{2\sigma^2}}$ which defines an RKHS space of functions $\mathcal{H}$ over the two-dimensional plane.

Figure 2.1: A binary classification problem : Classes (diamonds and circles) lie on two concentric circles.



Suppose we are given a small number, l, of labeled example pairs $(\boldsymbol{x}_i, y_i)$ where each $\boldsymbol{x}_i \in \mathcal{R}^2$ and $y_i \in \{-1, +1\}$. Then, in order to learn a good classifier from the labeled examples, one may solve the following regularization problem:

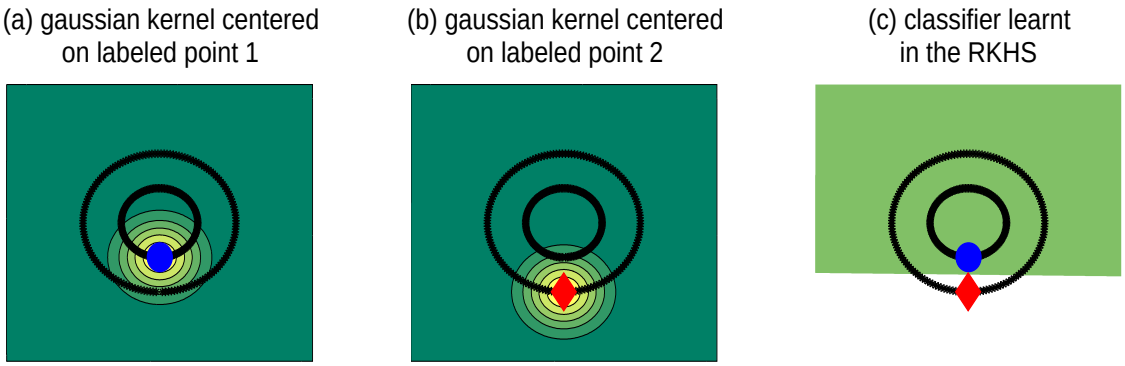
$$f = \arg\min_{h \in \mathcal{H}} \frac{1}{l} \sum_{i=1}^{l} V(h, \boldsymbol{x}_i, y_i) + \gamma \|h\|_{\mathcal{H}}^2$$

where $\|h\|_{\mathcal{H}}$ is the norm of the function h in the RKHS and V is a loss function. By the familiar representer theorem, recall that the solution can be expressed as:

$$f(\boldsymbol{x}) = \sum_{i=1}^{l} \alpha_i k(\boldsymbol{x}, x_i)$$

For illustrative purposes, we consider in Fig. 2.1(b) the case where $l = 2$, i.e., two labeled examples (one positive and one negative) are provided to the learner. Then the learned function would be a linear combination of two Gaussians, each centered on one of the two data points. The contours (level sets) of the Gaussian centered

Figure 2.2: Contours of the Gaussian Kernel and the decision surface



at each datapoint are shown in the Fig. 2(a),(b). Because the Gaussian kernel is isotropic, it has a spherical symmetry. As a result, the decision surface is linear, as shown in Fig. 2(c).

It is clear in our setting that the Gaussian with its spherical symmetry is an unfortunate choice for the kernel as it does not conform to the particular geometry of the underlying classes, and is unable to provide a satisfactory decision surface. The question we set for ourselves in this section is the following:

Can we define a kernel $\tilde{k}$ that is adapted to the geometry of the data distribution?

Such a kernel $\tilde{k}$ must have the property that (i) it is a valid Mercer kernel $\tilde{K} : \mathcal{X} \times \mathcal{X} \rightarrow \Re$ and therefore defines a new RKHS $\tilde{\mathcal{H}}$. (ii) it implements our intuitions about the geometry of the data. Our hope is to obtain an optimization problem over this new RKHS $\tilde{\mathcal{H}}$, given by:

$$g = \arg\min_{h \in \tilde{\mathcal{H}}} \frac{1}{2} \sum_{i=1}^{2} V(h, \boldsymbol{x}_i, y_i) + \|h\|^2_{\tilde{\mathcal{H}}}$$

whose solution $g(x) = \sum_{i=1}^{2} \alpha_i \tilde{k}(\boldsymbol{x}, \boldsymbol{x}_i)$ should be appropriate for our setting.

Notice that g is still a linear combination of two (modified) kernel functions, centered at the two data points in question. Yet, this solution must produce an intuitive decision surface that separates the two circles such as in Fig. 2.1(a). The form of such a Mercer kernel is not a-priori obvious for our picture.

In this section, we will show how to deform the original space to obtain a new RKHS $\tilde{\mathcal{H}}$ to satisfy our objectives, effectively reducing Manifold Regularization to standard supervised learning but using a novel modified kernel. As before, the geometry of the underlying marginal distribution may be estimated from unlabeled data and incorporated into the deformation procedure. The resulting new kernel $\tilde{k}$ can be computed explicitly in terms of unlabeled data. Working with only labeled data in this new RKHS, we can use the full power of *supervised* kernel methods for *semi-supervised* inference.

We highlight the following aspects of this construction:

1. As far as we know, we obtain the first truly data-dependent non-parametric kernel for semi-supervised learning. Prior work on data dependent kernels may be roughly classified into two categories: (a) choosing parameters for some parametric family of kernels, and (b) defining a data dependent kernel on the data points alone (transductive setting).

2. We discuss the basic theoretical properties of this kernel and establish that it is a valid Mercer kernel and therefore defines an RKHS.

3. These developments allow a family of algorithms to be developed based on various choices of the original RKHS, deformation penalties, loss functions and optimization strategies. In particular, standard SVM/RLS with the modified kernel are equivalent to Laplacian SVM/RLS with the base kernel. One can use

this construction as a black box: simply deform a kernel and plug it into standard methods such as Support Vector Regression, One-Class SVMs, Gaussian Processes and so on.

We now continue the discussion above and describe a general scheme for appropriately warping an RKHS.

2.5 Warping an RKHS using Point Cloud Norms

Before proceeding we discuss again the basic properties of RKHS relevant to this section. Let $\mathcal{X}$ be a compact domain in a Euclidean space or a manifold. A complete Hilbert space $\mathcal{H}$ of functions $\mathcal{X} \to \mathcal{R}$, with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ is a Reproducing Kernel Hilbert Space if point evaluation functionals are bounded, i.e., for any $\boldsymbol{x} \in \mathcal{X}, f \in \mathcal{H}$, there is a C, s.t.

$$|f(\boldsymbol{x})| \ \leq C \|f\|_{\mathcal{H}}$$

A symmetric positive semidefinite kernel $K(\boldsymbol{x}, \boldsymbol{z})$ can then be constructed using the Riesz representation theorem for the point evaluation functional:

$$f(\boldsymbol{x}) = \langle f, K(\boldsymbol{x}, \cdot) \rangle_{\mathcal{H}} \qquad K(\boldsymbol{x}, \boldsymbol{z}) = \langle K(\boldsymbol{x}, \cdot), k(z, \cdot) \rangle_{\mathcal{H}}$$

We will now show how a very general procedure to deform the norm $\| \ \|_{\mathcal{H}}$ gives a new RKHS $\tilde{\mathcal{H}}$ whose kernel we will denote by $\tilde{k}(x, z)$.

Let $\mathcal{V}$ be a linear space with a positive semi-definite inner product (quadratic form) and let $S : \mathcal{H} \to \mathcal{V}$ be a bounded linear operator. We define $\tilde{\mathcal{H}}$ to be the space of functions from $\mathcal{H}$ with the modified inner product

$$\langle f, g \rangle_{\tilde{\mathcal{H}}} = \langle f, g \rangle_{\mathcal{H}} + \langle Sf, Sg \rangle_{\mathcal{V}}$$

Proposition 2.5.1. *$\tilde{\mathcal{H}}$ is a Reproducing Kernel Hilbert Space.*

Proof. It is clear that $\tilde{\mathcal{H}}$ is complete, since a Cauchy sequence in the modified norm is also Cauchy in the original norm and therefore converges to an element of $\mathcal{H}$. For the same reason it is clear that point evaluations are bounded as $|f(\boldsymbol{x})| \leq C\|f\|_{\mathcal{H}}$ implies that $|f(\boldsymbol{x})| \leq C\|f\|_{\tilde{\mathcal{H}}}$.

We will be interested in the case when S and $\mathcal{V}$ depend on the data. We notice that while Proposition 2.5.1 is very general, and holds for any choice of S and $\mathcal{V}$, it is not usually easy to connect the kernels k and $\tilde{k}$.

However, as we will show below, for a class of what may be termed "point-cloud norms" this connection can be expressed explicitly.

Given the data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, let $S : \mathcal{H} \to \mathcal{R}^n$ be the evaluation map $S(f) = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n))$. Denote $\mathbf{f} = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n))$. The (semi-)norm on $\mathcal{R}^n$ will be given by a symmetric positive semi-definite matrix M:

$$\|Sf\|_{\mathcal{V}}^2 = \mathbf{f}^t M \mathbf{f}$$

We will derive the exact form for $\tilde{k}(\boldsymbol{x}, \boldsymbol{z})$. Note that $\tilde{\mathcal{H}}$ can be orthogonally decomposed as

$$\tilde{\mathcal{H}} = span\left\{\tilde{k}(\boldsymbol{x}_1, \cdot), \ldots, \tilde{k}(\boldsymbol{x}_n, \cdot)\right\} \oplus \tilde{\mathcal{H}}^{\perp}$$

where $\tilde{\mathcal{H}}^{\perp}$ consists of functions vanishing at all data points. It is clear that for any $f \in \tilde{\mathcal{H}}^{\perp}$, $Sf = 0$ and therefore $\langle f, g\rangle_{\tilde{\mathcal{H}}} = \langle f, g\rangle_{\mathcal{H}}$ for any function g in the space.

We therefore see that for any such $f \in \tilde{\mathcal{H}}^{\perp}$, we have

$$\begin{aligned} f(\boldsymbol{x}) &= \langle f, \tilde{k}(\boldsymbol{x}, \cdot)\rangle_{\tilde{\mathcal{H}}} \quad \text{(reproducing property in } \tilde{\mathcal{H}}) \\ &= \langle f, K(\boldsymbol{x}, \cdot)\rangle_{\mathcal{H}} \quad \text{(reproducing property in } \mathcal{H}) \\ &= \langle f, K(\boldsymbol{x}, \cdot)\rangle_{\tilde{\mathcal{H}}} \quad \text{since } f \in \tilde{\mathcal{H}}^{\perp} \end{aligned}$$

Thus, for any $f \in \tilde{\mathcal{H}}^{\perp}$, we have $\langle f, K(\boldsymbol{x},\cdot) - \tilde{k}(\boldsymbol{x},\cdot)\rangle_{\tilde{\mathcal{H}}} = 0$ or $K(\boldsymbol{x},\cdot) - \tilde{k}(\boldsymbol{x},\cdot) \in (\tilde{\mathcal{H}}^{\perp})^{\perp}$. In other words,

$$K(\boldsymbol{x},\cdot) - \tilde{k}(\boldsymbol{x},\cdot) \in span\left\{(\tilde{k}(\boldsymbol{x}_1,\cdot),\ldots,\tilde{k}(\boldsymbol{x}_n,\cdot)\right\}$$

On the other hand, for any $\boldsymbol{x}_i \in X$ and $f \in \tilde{\mathcal{H}}^{\perp}$ from the definition of the inner product on $\tilde{\mathcal{H}}$ we see $\langle K(\boldsymbol{x}_i,\cdot), f\rangle_{\tilde{\mathcal{H}}} = 0$. Thus, $K(\boldsymbol{x}_i,.) \in (\tilde{\mathcal{H}}^{\perp})^{\perp}$. Therefore, we see that

$$span\{K(\boldsymbol{x}_i,\cdot)\}_{i=1}^{n} \subseteq span\{(\tilde{k}(x_i,\cdot)\}_{i=1}^{n}$$

Also decomposing, $\mathcal{H} = span\left\{K(\boldsymbol{x}_i,\cdot)\right\}_{i=1}^{n} \oplus \mathcal{H}^{\perp}$, it is easy to check that $\tilde{k}(\boldsymbol{x}_i,.) \in (\mathcal{H}^{\perp})^{\perp}$ so that:

$$span\{\tilde{k}(\boldsymbol{x}_i,\cdot)\}_{i=1}^{n} \subseteq span\{K(\boldsymbol{x}_i,\cdot)\}_{i=1}^{n}$$

Thus, the two spans are same and we conclude that

$$\tilde{k}(\boldsymbol{x},\cdot) = K(\boldsymbol{x},\cdot) + \sum_j \beta_j(\boldsymbol{x}) K(\boldsymbol{x}_j,\cdot)$$

where the coefficients β_j depend on $\boldsymbol{x}$.

To find $\beta_j(\boldsymbol{x})$, we look at a system of linear equations generated by evaluating $K(\boldsymbol{x}_i,.)$ at $\boldsymbol{x}$:

$$\begin{aligned} k_{\boldsymbol{x}_i}(\boldsymbol{x}) &= \langle K(\boldsymbol{x}_i,.), \tilde{k}(\boldsymbol{x},\cdot)\rangle_{\tilde{\mathcal{H}}} \\ &= \langle K(\boldsymbol{x}_i,.), K(\boldsymbol{x},\cdot) + \sum_j \beta_j(\boldsymbol{x}) K(\boldsymbol{x}_j,\cdot)\rangle_{\tilde{\mathcal{H}}} \\ &= \langle K(\boldsymbol{x}_i,.), K(\boldsymbol{x},\cdot) + \sum_j \beta_j(\boldsymbol{x}) K(\boldsymbol{x}_j,\cdot)\rangle_{\mathcal{H}} + \mathbf{k_{x_i}}^t M \mathbf{g} \end{aligned}$$

where $\mathbf{k_{x_i}} = (K(\boldsymbol{x}_i,\boldsymbol{x}_1)\ldots K(\boldsymbol{x}_i,\boldsymbol{x}_n))^t$ and $\mathbf{g}$ is the vector given by the components

$\mathbf{g}_k = K(\boldsymbol{x}, \boldsymbol{x}_k) + \sum_j \beta_j(\boldsymbol{x}) K(\boldsymbol{x}_j, \boldsymbol{x}_k)$. This formula provides the following system of linear equations for the coefficients $\beta(\boldsymbol{x}) = (\beta_1(\boldsymbol{x}) \dots \beta_n(\boldsymbol{x}))^T$:

$$(I + MK)\beta(\boldsymbol{x}) = -M\mathbf{k}_{\boldsymbol{x}}$$

where K is the matrix $K_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\mathbf{k}_{\boldsymbol{x}}$, as before, denotes the vector $(K(\boldsymbol{x}_1, \boldsymbol{x}) \dots K(\boldsymbol{x}_n, \boldsymbol{x}))^t$.

Finally, we obtain the following explicit form for $\tilde{k}$:

Proposition 2.5.2. *Reproducing kernel of $\tilde{\mathcal{H}}$:*

$$\tilde{K}(\boldsymbol{x}, \boldsymbol{z}) = K(\boldsymbol{x}, \boldsymbol{z}) - \mathbf{k}_{\boldsymbol{x}}^t (I + MK)^{-1} M \mathbf{k}_{\boldsymbol{z}} \tag{2.22}$$

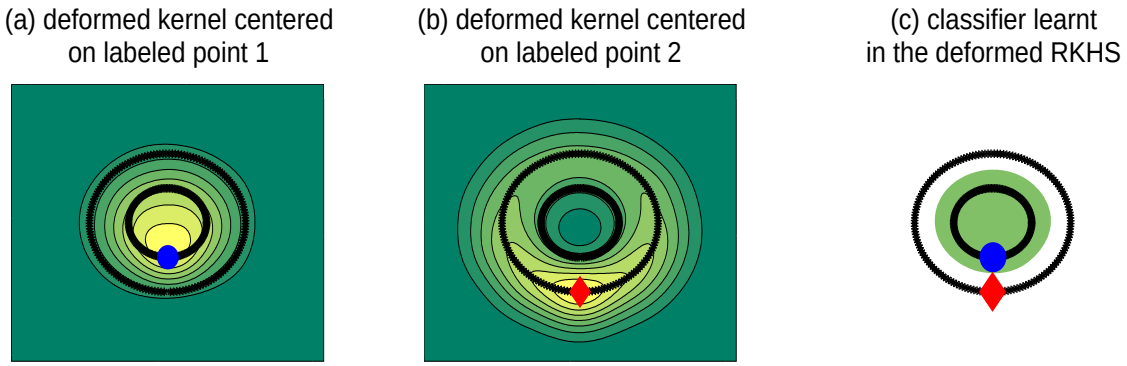
One can observe that the matrix $(I + MK)^{-1}M$ is symmetric. When M is invertible, it equals $(M^{-1} + K)^{-1}$ which is clearly symmetric. When M is singular, one adds a small ridge term to M and then uses a continuity argument.

We see that modifying the RKHS with a point-cloud norm deforms the kernel along a finite-dimensional subspace given by the data.

2.5.1 Choosing the Point Cloud Norm

The key issue is the choice of M, so that the deformation of the kernel induced by the data-dependent norm, is motivated with respect to our intuitions about the data. Such intuitions may be inspired by forms of prior knowledge (e.g, transformation invariances), or, in the case of semi-supervised learning, by the form of the marginal distribution as described by unlabeled data. As developed earlier in this chapter, we utilize a graph regularizer based on the graph Laplacian associated to the point cloud.

Figure 2.3: Contours of the data-dependent modified Kernel



2.5.2 Back to Concentric Circles

The result of modifying the kernel by using the graph Laplacian for the particular case of two circles[2] is shown in Fig. 2.3.

In Fig. 2.3(a) and Fig. 2.3(b) we see level lines for modified kernels centered on two points on smaller and larger circles respectively. We see that as expected the kernel becomes extended along the circle. This distortion of the kernel reflects two intuitions about the natural data: what may be termed "the manifold assumption", i.e. the notion that our regression/classification function is smooth with respect to the underlying probability distribution and the related "cluster assumption" (see e.g [30]), which suggests that classes form distinct "clusters" separated by low density areas. The kernel, such as shown in Fig. 2.3, heavily penalizes changes along the circle, while imposing little penalty on changes in the orthorgonal direction.

Finally, Fig. 2.3(c) shows the class boundary obtained using this new kernel.

2. Each consisting of 150 evenly spaced unlabeled points, with one labeled example

2.6 Related Work and Connections to Other Algorithms

In this section we survey various approaches to semi-supervised and transductive learning and highlight connections of Manifold Regularization to other algorithms.

Transductive SVM (TSVM) [98, 58]: TSVMs are based on the following optimization principle :

$$f^* = \underset{\substack{f\in\mathcal{H}_K \\ y_{l+1},\cdots y_{l+u}}}{\operatorname{argmin}} \; C\sum_{i=1}^{l}(1-y_i f(\boldsymbol{x}_i))_+ + C^*\sum_{i=l+1}^{l+u}(1-y_i f(\boldsymbol{x}_i))_+ + \|f\|_K^2 \qquad (2.23)$$

which proposes a joint optimization of the SVM objective function over binary-valued labels on the unlabeled data and functions in the RKHS. Here, C, C^* are parameters that control the relative hinge-loss over labeled and unlabeled sets. The joint optimization is implemented in [58] by first using an inductive SVM to label the unlabeled data and then iteratively solving SVM quadratic programs, at each step switching labels to improve the objective function. However this procedure is susceptible to local minima and requires an unknown, possibly large number of label switches before converging. Note that even though TSVM were inspired by transductive inference, they do provide an out-of-sample extension. We revisit TSVM style approaches in detail in Chapter 3.

Semi-Supervised SVMs (S^3VM) [11, 49]: S^3VM incorporate unlabeled data by including the minimum hinge-loss for the two choices of labels for each unlabeled example, with the same objective as TSVMs. This is formulated as a mixed-integer program for linear SVMs in [11] and is found to be intractable for large amounts of unlabeled data. [49] reformulate this approach as a concave minimization problem which is solved by a successive linear approximation algorithm. The presentation of these algorithms is restricted to the linear case.

Measure-Based Regularization [20]: The conceptual framework of this work is closest to our approach. The authors consider a gradient based regularizer that penalizes variations of the function more in high density regions and less in low density regions leading to the following optimization principle:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{F}} \sum_{i=1}^{l} V(f(\boldsymbol{x}_i), y_i) + \gamma \int_X \langle \nabla f(\boldsymbol{x}), \nabla f(\boldsymbol{x}) \rangle p(\boldsymbol{x}) d\boldsymbol{x} \tag{2.24}$$

where p is the density of the marginal distribution $\mathcal{P}_{\mathcal{X}}$. The authors observe that it is not straightforward to find a kernel for arbitrary densities p, whose associated RKHS norm is

$$\int \langle \nabla f(\boldsymbol{x}), \nabla f(\boldsymbol{x}) \rangle p(\boldsymbol{x}) dx$$

Thus, in the absence of a representer theorem, the authors propose to perform minimization of the regularized loss on a fixed set of basis functions chosen apriori, i.e, $\mathcal{F} = \{\sum_{i=1}^{q} \alpha_i \phi_i\}$. For the hinge loss, this paper derives an SVM quadratic program in the coefficients $\{\alpha_i\}_{i=1}^{q}$ whose Q matrix is calculated by computing q^2 integrals over gradients of the basis functions. However the algorithm does not demonstrate performance improvements in real world experiments. It is also worth noting that while [20] use the gradient $\nabla f(\boldsymbol{x})$ in the ambient space, we use the gradient over a submanifold $\nabla_{\mathcal{M}} f$ for penalizing the function. In a situation where the data truly lies on or near a submanifold $\mathcal{M}$, the difference between these two penalizers can be significant since smoothness in the normal direction to the data manifold is irrelevant to classification or regression.

Graph Based Approaches See, e.g., [17, 29, 91, 109, 111, 112, 63, 60, 5]: A variety of graph based methods have been proposed for transductive inference. However, these methods do not provide an out-of-sample extension. In [111], nearest

neighbor labeling for test examples is proposed once unlabeled examples have been labeled by transductive learning. In [29], test points are approximately represented as a linear combination of training and unlabeled points in the feature space induced by the kernel. For Graph Regularization and Label Propagation see [90, 3, 111]. [90] discusses the construction of a canonical family of graph regularizers based on the graph Laplacian. [112] presents a non-parametric construction of graph regularizers.

Manifold regularization provides natural out-of-sample extensions to several graph based approaches. These connections are summarized in Table 2.2.

We also note the recent work [40] on out-of-sample extensions for semi-supervised learning where an induction formula is derived by assuming that the addition of a test point to the graph does not change the transductive solution over the unlabeled data.

Cotraining [19]: The Co-training algorithm was developed to integrate abundance of unlabeled data with availability of multiple sources of information in domains like web-page classification. Weak learners are trained on labeled examples and their predictions on subsets of unlabeled examples are used to mutually expand the training set. Note that this setting may not be applicable in several cases of practical interest where one does not have access to multiple information sources.

Bayesian Techniques See e.g., [71, 79, 37]. An early application of semi-supervised learning to Text classification appeared in [71] where a combination of EM algorithm and Naive-Bayes classification is proposed to incorporate unlabeled data. [79] provides a detailed overview of Bayesian frameworks for semi-supervised learning. The recent work in [37] formulates a new information-theoretic principle to develop a regularizer for conditional log-likelihood.

Table 2.2: Connections of Manifold Regularization to other algorithms

Parameters	Corresponding algorithms (square loss or hinge loss)
$\gamma_A \geq 0 \;\; \gamma_I \geq 0$	Manifold Regularization
$\gamma_A \geq 0 \;\; \gamma_I = 0$	Standard Regularization (RLS or SVM)
$\gamma_A \to 0 \;\; \gamma_I > 0$	Out-of-sample extension for Graph Regularization (RLS or SVM)
$\gamma_A \to 0 \;\; \gamma_I \to 0$ $\gamma_I \gg \gamma_A$	Out-of-sample extension for Label Propagation (RLS or SVM)
$\gamma_A \to 0 \;\; \gamma_I = 0$	Hard margin SVM or Interpolated RLS

2.7 Experiments

We performed experiments on synthetic and real world datasets. For detailed experimental benchmark studies, we refer the reader to [25, 82, 87].

2.7.1 Visual Illustration of Ambient-Intrinsic Tradeoff

The two moons dataset is shown in Figure 2.4. The dataset contains 200 examples with only 1 labeled example for each class. Also shown are the decision surfaces of Laplacian SVM for increasing values of the intrinsic regularization parameter γ_I. When $\gamma_I = 0$, Laplacian SVM disregards unlabeled data and returns the SVM decision boundary which is fixed by the location of the two labeled points. As γ_I is increased, the intrinsic regularizer incorporates unlabeled data and causes the decision surface to appropriately adjust according to the geometry of the two classes.

In Figure 2.5, the best decision surfaces across a wide range of parameter settings are also shown for SVM, Transductive SVM and Laplacian SVM. The figure demonstrates how TSVM fails to find the optimal solution, probably since it gets stuck in a local minimum (see Chapter 3). The Laplacian SVM decision boundary seems to be intuitively most satisfying.

Figure 2.4: Laplacian SVM with RBF Kernels for various values of γ_I. Labeled points are shown in color, other points are unlabeled.

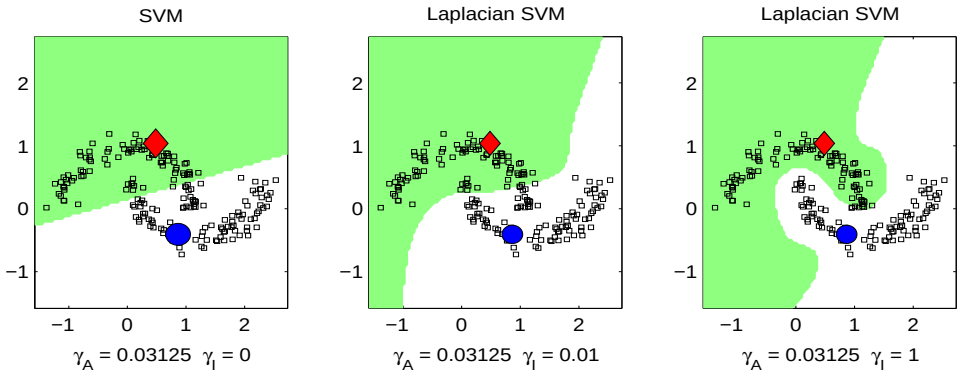
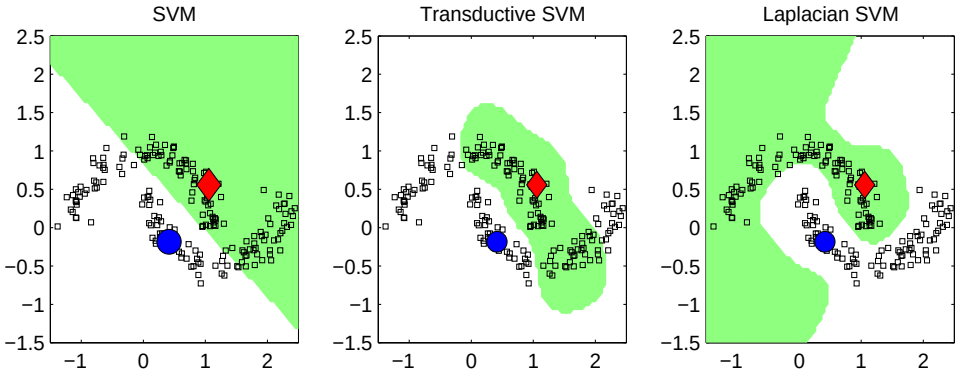


Figure 2.5: Two Moons Dataset: Best decision surfaces using RBF kernels for SVM, TSVM and Laplacian SVM. Labeled points are shown in color, other points are unlabeled.



2.7.2 Spoken Letter Recognition

This experiment was performed on the Isolet database of letters of the English alphabet spoken in isolation (available from the UCI machine learning repository). The data set contains utterances of 150 subjects who spoke the name of each letter of the English alphabet twice. The speakers are grouped into 5 sets of 30 speakers each, referred to as isolet1 through isolet5. For the purposes of this experiment, we chose to train on the first 30 speakers (isolet1) forming a training set of 1560 examples, and

test on isolet5 containing 1559 examples (1 utterance is missing in the database due to poor recording). We considered the task of classifying the first 13 letters of the English alphabet from the last 13. We considered 30 binary classification problems corresponding to 30 splits of the training data where all 52 utterances of one speaker were labeled and all the rest were left unlabeled. The test set is composed of entirely new speakers, forming the separate group isolet5.

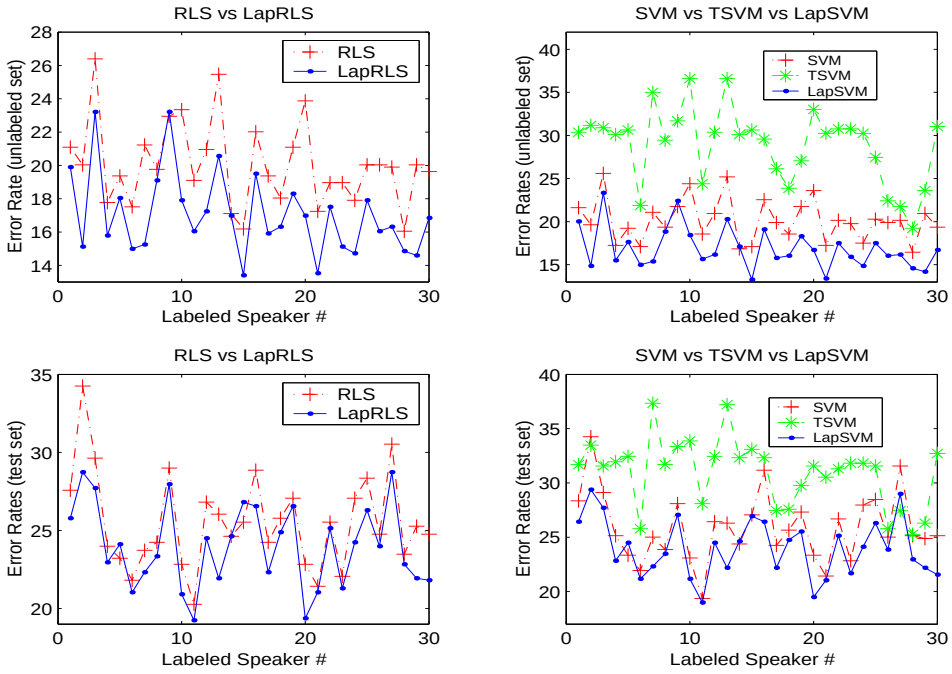


Figure 2.6: Isolet Experiment - Error Rates at precision-recall break-even points of 30 binary classification problems

We chose to train with RBF kernels of width $\sigma = 10$ (this was the best value among several settings with respect to 5-fold cross-validation error rates for the fully supervised problem using standard SVM). For SVM and RLS we set $\gamma l = 0.05$ ($C = 10$) (this was the best value among several settings with respect to mean error rates over the 30 splits). For Laplacian RLS and Laplacian SVM we set $\gamma_A l = \frac{\gamma_I l}{(u+l)^2} = 0.005$.

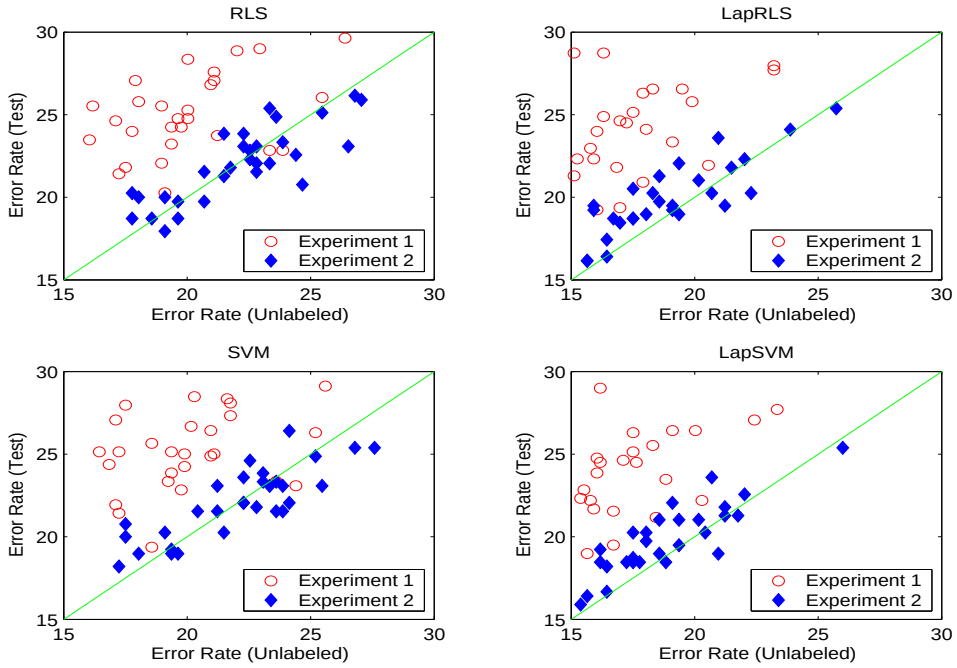


Figure 2.7: Isolet Experiment - Error Rates at precision-recall break-even points on Test set Versus Unlabeled Set. In Experiment 1, the training data comes from Isolet 1 and the test data comes from Isolet5; in Experiment 2, both training and test sets come from Isolet1.

In Figures 2.6, we compare these algorithms. The following comments can be made: (a) LapSVM and LapRLS make significant performance improvements over inductive methods and TSVM, for predictions on unlabeled speakers that come from the same group as the labeled speaker, over all choices of the labeled speaker. (b) On Isolet5 which comprises of a separate group of speakers, performance improvements are smaller but consistent over the choice of the labeled speaker. This can be expected since there appears to be a systematic bias that affects all algorithms, in favor of same-group speakers. To test this hypothesis, we performed another experiment in which the training and test utterances are both drawn from Isolet1. Here, the second utterance of each letter for each of the 30 speakers in Isolet1 was taken away to form the test set containing 780 examples. The training set consisted of the first

utterances for each letter. As before, we considered 30 binary classification problems arising when all utterances of one speaker are labeled and other training speakers are left unlabeled. The scatter plots in Figure 2.7 confirm our hypothesis, and show high correlation between in-sample and out-of-sample performance of our algorithms in this experiment. It is encouraging to note performance improvements with unlabeled data in Experiment 1 where the test data comes from a slightly different distribution. This robustness is often desirable in real-world applications.

Table 2.3: Isolet: one-versus-rest multiclass error rates

Method	SVM	TSVM	LapSVM	RLS	LapRLS
Error (unlabeled)	28.6	46.6	24.5	28.3	24.1
Error (test)	36.9	43.3	33.7	36.3	33.3

In Table 2.3 we report mean error rates over the 30 splits from one-vs-rest 26-class experiments on this dataset. The parameters were held fixed as in the 2-class setting. The failure of TSVM in producing reasonable results on this dataset has also been observed in [60]. With LapSVM and LapRLS we obtain around 3 to 4% improvement over their supervised counterparts.

2.7.3 Text Categorization

We performed Text Categorization experiments on the WebKB dataset which consists of 1051 web pages collected from Computer Science department web-sites of various universities. The task is to classify these web pages into two categories: *course* or *non-course*. We considered learning classifiers using only textual content of the web pages, ignoring link information. A bag-of-word vector space representation for documents is built using the the top 3000 words (skipping HTML headers) having highest mutual

information with the class variable, followed by TFIDF mapping[3]. Feature vectors are normalized to unit length. 9 documents were found to contain none of these words and were removed from the dataset.

For the first experiment, we ran LapRLS and LapSVM in a transductive setting, with 12 randomly labeled examples (3 course and 9 non-course) and the rest unlabeled. In Table 2.4, we report the precision and error rates at the precision-recall break-even point averaged over 100 realizations of the data, and include results reported in [60] for Spectral Graph Transduction, and the Cotraining algorithm [19] for comparison. We used 15 nearest neighbor graphs, weighted by cosine distances and used iterated Laplacians of degree 3. For inductive methods, $\gamma_A l$ was set to 0.01 for RLS and 1.00 for SVM. For LapRLS and LapSVM, γ_A was set as in inductive methods, with $\frac{\gamma_I l}{(l+u)^2} = 100\gamma_A l$. These parameters were chosen based on a simple grid search for best performance over the first 5 realizations of the data. Linear Kernels and cosine distances were used since these have found wide-spread applications in text classification problems, e.g., in [44].

Since the exact datasets on which these algorithms were run, somewhat differ in preprocessing, preparation and experimental protocol, these results are only meant to suggest that Manifold Regularization algorithms perform similar to state-of-the-art methods for transductive inference in text classification problems. The following comments can be made: (a) Transductive categorization with LapSVM and LapRLS leads to significant improvements over inductive categorization with SVM and RLS. (b) [60] reports 91.4% precision-recall breakeven point, and 4.6% error rate for TSVM. Results for TSVM reported in the table were obtained when we ran the TSVM imple-

3. TFIDF stands for Term Frequency Inverse Document Frequency. It is a common document preprocessing procedure, which combines the number of occurrences of a given term with the number of documents containing it.

Table 2.4: Precision and Error Rates at the Precision-Recall Break-even Points of supervised and transductive algorithms.

Method	PRBEP	Error
k-NN [60]	73.2	13.3
SGT [60]	86.2	6.2
Naive-Bayes [19]	—	12.9
Cotraining [19]	—	6.20
SVM	76.39 (5.6)	10.41 (2.5)
TSVM[4]	88.15 (1.0)	5.22 (0.5)
LapSVM	87.73 (2.3)	5.41 (1.0)
RLS	73.49 (6.2)	11.68 (2.7)
LapRLS	86.37 (3.1)	5.99 (1.4)

mentation using SVM-Light software on this particular dataset. The average training time for TSVM was found to be more than 10 times slower than for LapSVM. (c) The Co-training results were obtained on unseen test datasets utilizing additional hyperlink information, which was excluded in our experiments. This additional information is known to improve performance, as demonstrated in [60] and [19].

In the next experiment, we randomly split the WebKB data into a test set of 263 examples and a training set of 779 examples. We noted the performance of inductive and semi-supervised classifiers on unlabeled and test sets as a function of the number of labeled examples in the training set. The performance measure is the precision-recall break-even point (PRBEP), averaged over 100 random data splits. Results are presented in the top panel of Figure 2.8. The benefit of unlabeled data can be seen by comparing the performance curves of inductive and semi-supervised classifiers.

We also performed experiments with different sizes of the training set, keeping a randomly chosen test set of 263 examples. The bottom panel in Figure 2.8 presents the quality of transduction and semi-supervised learning with Laplacian SVM (Laplacian RLS performed similarly) as a function of the number of labeled examples for different

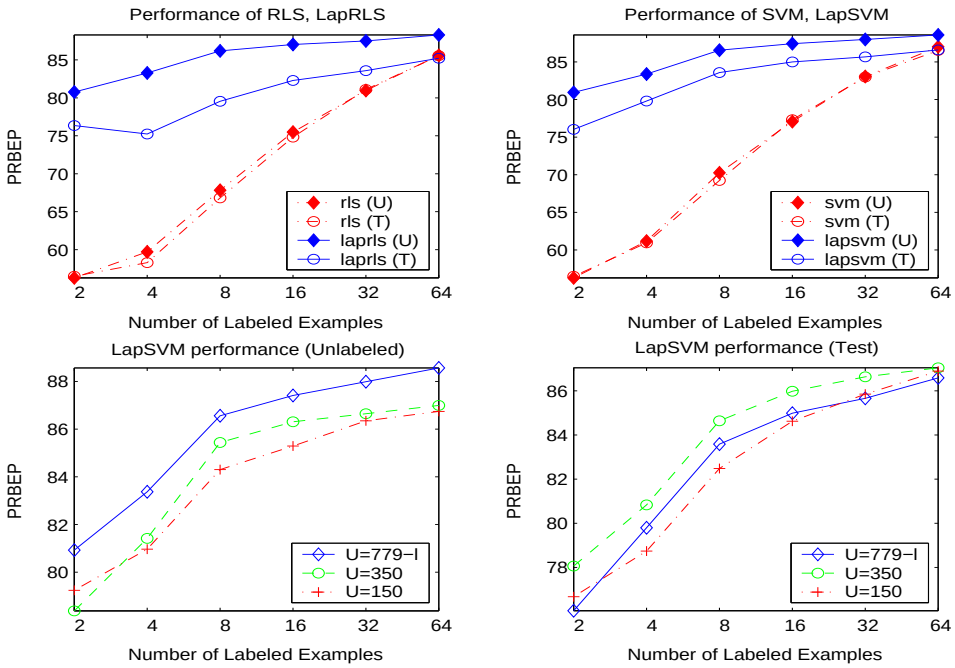


Figure 2.8: WebKb Text Classification Experiment : The top panel presents performance in terms of precision-recall break-even points (PRBEP) of RLS,SVM,Laplacian RLS and Laplacian SVM as a function of number of labeled examples, on Test (marked as T) set and Unlabeled set (marked as U and of size 779-number of labeled examples). The bottom panel presents performance curves of Laplacian SVM for different number of unlabeled points.

amounts of unlabeled data. We find that transduction improves with increasing unlabeled data. We expect this to be true for test set performance as well, but do not observe this consistently possibly since we use a fixed set of parameters that become suboptimal as unlabeled data is increased. The optimal choice of the regularization parameters depends on the amount of labeled and unlabeled data, and should be adjusted by the model selection protocol accordingly.

2.7.4 Results on Benchmark Collections

[30] provide a collection of datasets which have frequently been used for benchmark

studies in Semi-supervised learning. Here, comparisons are made based on results reported in [30] and [59] with transductive graph methods (abbreviated Graph-Trans) such as Graph Regularization [2] and Spectral Graph Transduction (SGT) [59]; the implementation of Transductive SVMs [97] in [57] (TSVM) and in [30] (∇TSVM)); and with other methods proposed in [30] : training an SVM on a graph-distance derived kernel (Graph-density) and Low Density Separation ([30]).

Data Sets

Experiments were performed on five well-known datasets described in Table 2.5.

Table 2.5: Datasets used in the experiments : c is the number classes, d is the data dimensionality, l is the number of labeled examples, n is the total number of examples in the dataset from which labeled, unlabeled and test examples, when required, are drawn.

Dataset	SVM	Laplacian SVM	RLS	Laplacian RLS
g50c	2	50	50	550
Coil20	20	1024	40	1440
Uspst	10	256	50	2007
mac-windows	2	7511	50	1946
Webkb (page)	2	3000	12	1051
Webkb (link)	2	1840	12	1051
Webkb (page+link)	2	4840	12	1051

g50c is an artificial dataset generated from two unit-covariance normal distributions with equal probabilities. The class means are adjusted so that the true bayes error is 5%, and 550 examples are drawn. Coil20 and Uspst datasets pose multiclass image classification problems. Coil20 consists of 32×32 gray scale images of 20 objects viewed from varying angles and Uspst is taken from the USPS (test) dataset for handwritten digit recognition. The text data consists of binary classification problems: mac-win is taken from the 20-newsgroups dataset and the task is to categorize

newsgroup documents into two topics: *mac* or *windows*; and the WebKB dataset which we used in the previous section. For each WebKB document, there are two representations: the textual content of the webpage (which we will call *page* representation) and the anchortext on links on other webpages pointing to the webpage (*link* representation). Following [71], we generated bag-of-words feature vectors for both representations as follows: Documents were tokenized using the Rainbow Text toolkit [69]; HTML-tagged content was skipped and no stoplist or stemming was used; numbers were included in the tokenization. For the page representation, 3000 features were selected according to information gain. For the link representation, 1840 features were generated with no feature selection. The columns of the document-word matrix were scaled based on inverse document frequency weights (IDF) for each word and the resulting TFIDF feature vectors were length normalized. We also considered a joint (*page+link*) representation by concatenating the features.

In the discussion ahead, by a *training set* we will mean the union of the *labeled set* and the *unlabeled set* of examples available to transductive and semi-supervised learners. *Test sets* comprise of examples never seen before.

Transductive Setting

In the transductive setting, the training set comprises of n examples, l of which are labeled (n, l are specified in Table 2.5). In Table 2.6 and Table 2.7, we lay out a performance comparison of several algorithms in predicting the labels of the $n - l$ unlabeled examples. The experimental protocol is based on [59] for the WebKB dataset and [30] for other datasets.

Protocol: For datasets other than WebKB, performance is evaluated by error rates averaged over 10 random choices of the labeled set. Each random set samples each

Table 2.6: Transductive Setting: Error Rates on unlabeled examples. Results on which Laplacian SVMs (LapSVM) and Laplacian RLS (LapRLS) outperform all other methods are shown in bold. Results for Graph-Trans, TSVM,∇TSVM,Graph-density, and LDS are taken from [30]

Dataset → Algorithm ↓	g50c	Coil20	Uspst	mac-win
SVM (n)	4.0 (2.9)	0.0 (0.0)	2.8 (0.8)	2.4 (1.3)
RLS (n)	4.0 (2.7)	0.0 (0.0)	2.5 (1.3)	2.8 (1.7)
SVM (l)	9.7 (1.7)	24.6 (1.7)	23.6 (3.3)	18.9 (5.7)
RLS (l)	8.5 (1.5)	26.0 (1.5)	23.6 (3.5)	18.8 (5.7)
Graph-Trans	17.3	6.2	21.3	11.7
TSVM	6.9	26.3	26.5	7.4
Graph-density	8.3	6.4	16.9	10.5
∇TSVM	5.8	17.6	17.6	5.7
LDS	5.6	4.9	15.8	5.1
LapSVM	**5.4** (0.6)	**4.0** (2.3)	**12.7** (2.3)	10.4 (1.1)
LapRLS	**5.2** (0.7)	**4.3** (1.3)	**12.7** (2.4)	10.0 (1.3)

class at least once (twice for coil20). Results for Graph-Reg, TSVM,∇TSVM,Graph-density, and LDS are taken from [30] where models were selected by optimizing error rates on the unlabeled set giving these methods an unfair advantage. For, LDS, a cross-validation protocol was used in [30]. For LapRLS, LapSVM we preferred to fix $\gamma_A = 10^{-6}, \gamma_I = 0.01$ to reduce the complexity of model selection. Gaussian base kernels and euclidean nearest neighbor graphs with gaussian weights were used. The three parameters : number of nearest neighbors (nn), the degree (p) of the graph Laplacian, and the width (σ) of the Gaussian are chosen based on 5-fold cross-validation performance in a small grid of parameter values. Together, these parameters specify the deformed kernel that incorporates the unlabeled data.

For WebKB (Table 2.7), we evaluated performance by precision-recall breakeven points. Linear Kernels and cosine nearest neighbor graphs with gaussian weights were used. In this case, we fixed $nn = 200$ (as in [59]), $p = 5$ (unoptimized), and σ as the

Table 2.7: Transductive Setting: 100-PRBEP for WebKb on unlabeled examples. Results on which Laplacian SVMs (LapSVM) and Laplacian RLS (LapRLS) outperform all other methods are shown in bold. LapSVM$_{joint}$, LapRLS$_{joint}$ use the sum of graph laplacians in each WebKB representation.

Dataset → Algorithm ↓	WebKB (link)	WebKB (page)	WebKB (page+link)
SVM (n)	5.1 (2.8)	5.3 (4.0)	0.7 (1.4)
RLS (n)	5.6 (2.8)	6.4 (3.8)	2.2 (3.0)
SVM (l)	28.1 (16.1)	24.3 (15.0)	18.2 (15.5)
RLS (l)	30.3 (16.5)	30.2 (15.3)	23.9 (16.1)
Graph-Trans	22.0	10.7	6.6
TSVM	14.5	8.6	7.8
LapSVM	17.2 (9.0)	10.9 (1.2)	**6.4** (0.9)
LapRLS	19.2 (10.0)	11.2 (1.1)	7.5 (1.4)
LapSVM$_{joint}$	**5.7** (1.5)	**6.6** (1.3)	**5.1** (0.9)
LapRLS$_{joint}$	**6.7** (6.2)	**8.9** (3.9)	**5.9** (2.9)

mean edge length in the graph. Since the labeled set is very small for this dataset, we performed model selection (including γ_A, γ_I for LapSVM, LapRLS) for all algorithms by optimizing performance on the unlabeled set.

Discussion: Using the proposed data-dependent semi-supervised kernel (2.22), Laplacian SVM and RLS return the best performance in four of the five datasets. In g50c, performance is close to the bayes optimal. We obtain significant performance gains on Coil20 and Uspst where there are strong indications of a manifold structure. On WebKB, the methods outperform other methods in the *page+link* representation. We also tried the following novel possibility: the point cloud norm was constructed from the mean graph Laplacian over the three representations and used for deforming RKHS in each representation. With this multi-view regularizer, the method significantly outperforms all other methods for all representations.

Finally, note that one can recover the original base kernel by setting $\gamma_I = 0$.

With a good model selection, the proposed methods should never perform worse than inductive methods.

Semi-supervised Setting

In the semi-supervised setting, the training set comprises of $l+u$ examples (l labeled as before and u unlabeled) and the test set comprises of $n-l-u$ examples. Experiments were performed to observe the performance of LapSVM and LapRLS on the test and unlabeled sets to see how well these methods extend to novel out-of-sample examples.

Protocol: We performed a variation of 4-fold cross-validation. The data was divided into four equal chunks: three chunks were combined to form the training set and the remaining formed the test set. Each chunk therefore appeared in the training data thrice and as a test set once. Tables 2.8 and 2.9 report mean performance of LapSVM and LapRLS in predicting the labels of each chunk as a subset of the unlabeled set and as a test set. γ_A, γ_I are optimized for best mean performance; and the other parameters are set as before. For WebKB, is is natural for the four chunks to correspond to the four universities: training on three universities and testing on the fourth. The detailed performance for each university is reported in Table 2.8 for LapSVM (performance is similar for LapRLS).

Discussion: For g50c, mac-win, and WebKB the performance on unlabeled and test subsets is almost indistinguishable. The out-of-sample extension is high quality also for Uspst. For Coil20, we observe an over-deformation phenomenon : the in-sample performance is significantly better than out-of-sample performance. A smoother base kernel and appropriate degree of deformation can remove this difference for coil20.

Table 2.8: Semi-supervised Setting: (WebKB) 100-PRBEP on unlabeled and test examples

View →	link	page	page+link
University ↓	unlab test	unlab test	unlab test
Cornell	26.1 27.3	14.4 14.3	8.0 8.0
Texas	18.8 17.3	19.0 17.8	4.7 5.1
Washington	12.8 13.8	8.7 8.4	4.8 4.5
Wisconsin	18.6 19.3	14.5 15.7	7.1 7.0

Parameters of Deformation

The parameters γ_A, γ_I specify a trade-off between ambient regularization and deformation. In Fig 2.9 we show the performance difference over test sets and unlabeled subsets as a function on the γ_A, γ_I plane. Also shown is the location of the optimal γ_A, γ_I. For a wide range of parameter settings, the performance difference is less than 1% for g50c and mac-win, and less than 2% for coil20 and uspst. In uspst and coil20 we see an expected behaviour : When γ_I is much larger than γ_A, the point cloud norm dominates the regularization and the in-sample performance is found to be much better than the out-of-sample performance. When γ_A is increased, the difference decreases. In general, the optimal performance strikes a good balance between the ambient norm and the degree of deformation.

2.8 Unsupervised and Fully Supervised Cases

While the previous discussion concentrated on the semi-supervised case, our framework covers both unsupervised and fully supervised cases as well.

Table 2.9: Semi-supervised Setting: Error rates on unlabeled and test examples.

Dataset →	g50c	Coil20	Uspst	mac-win
Algorithm ↓	unlab test	unlab test	unlab test	unlab test
SVM	9.7 9.7	21.7 22.6	21.6 22.1	20.9 20.9
RLS	9.1 9.6	21.8 22.6	22.5 23.0	20.9 20.4
LapSVM	4.9 5.0	8.7 14.6	14.9 17.7	9.9 9.7
LapRLS	4.9 4.9	9.40 12.9	14.3 17.0	9.4 9.3

2.8.1 Unsupervised Learning: Clustering and Data Representation

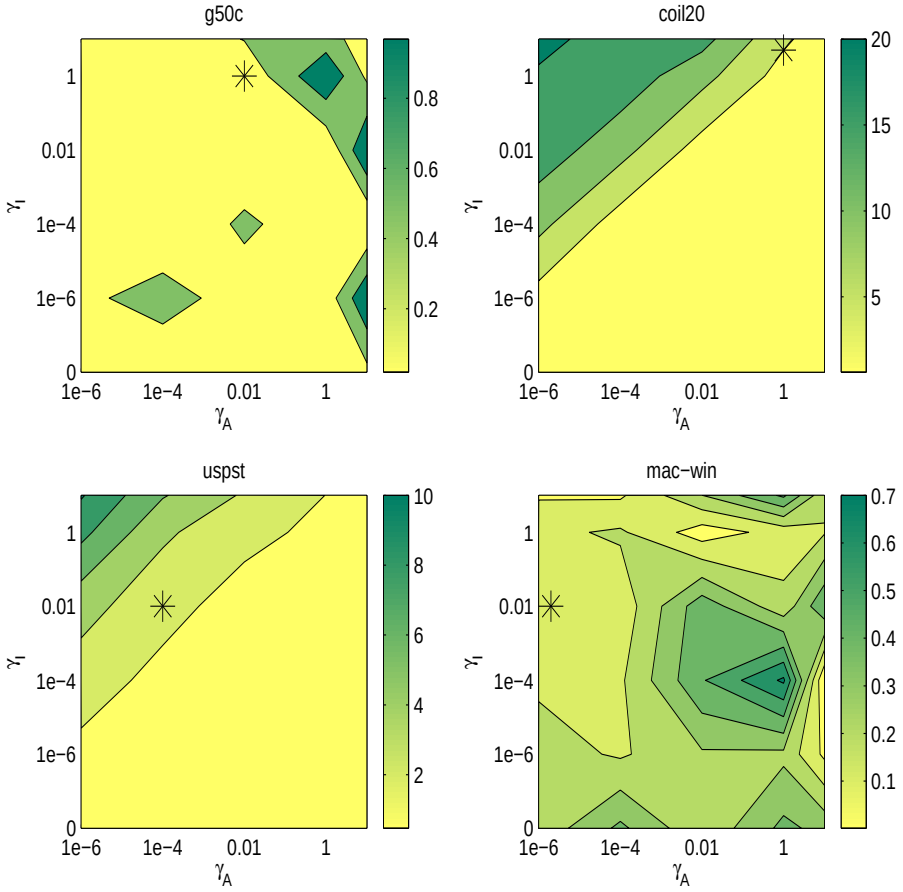
In the unsupervised case one is given a collection of unlabeled data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_u$. Our basic algorithmic framework embodied in the optimization problem in Eqn. 2.3 has three terms: (i) fit to labeled data, (ii) extrinsic regularization and (iii) intrinsic regularization. Since no labeled data is available, the first term does not arise anymore. Therefore we are left with the following optimization problem:

$$\min_{f \in \mathcal{H}_K} \gamma_A \|f\|_K^2 + \gamma_I \|f\|_I^2 \tag{2.25}$$

Of course, only the ratio $\gamma = \frac{\gamma_A}{\gamma_I}$ matters. As before $\|f\|_I^2$ can be approximated using the unlabeled data. Choosing $\|f\|_I^2 = \int_{\mathcal{M}} \langle \nabla_{\mathcal{M}} f, \nabla_{\mathcal{M}} f \rangle$ and approximating it by the empirical Laplacian, we are left with the following optimization problem:

$$f^* = \operatorname*{argmin}_{\substack{\sum_i f(x_i)=0;\ \sum_i f(x_i)^2=1 \\ f \in \mathcal{H}_K}} \gamma \|f\|_K^2 + \sum_{i \sim j} (f(x_i) - f(x_j))^2 \tag{2.26}$$

Figure 2.9: Difference in Error Rates (in percentage on the vertical colorbar) over test sets and unlabeled subsets in the $\gamma_I - \gamma_A$ plane. The optimal mean performance is obtained at the point marked by a black star.



Note that to avoid degenerate solutions we need to impose some additional conditions (cf. [8]). It turns out that a version of Representer theorem still holds showing that the solution to Eqn. 2.26 admits a representation of the form

$$f^* = \sum_{i=1}^{u} \alpha_i K(\boldsymbol{x}_i, \cdot)$$

By substituting back in Eqn. 2.26, we come up with the following optimization problem:

$$\alpha = \underset{\substack{\mathbf{1}^T K\alpha=0 \\ \alpha^T K^2\alpha=1}}{\operatorname{argmin}} \ \gamma\|f\|_K^2 + \sum_{i\sim j}(f(x_i) - f(x_j))^2 \tag{2.27}$$

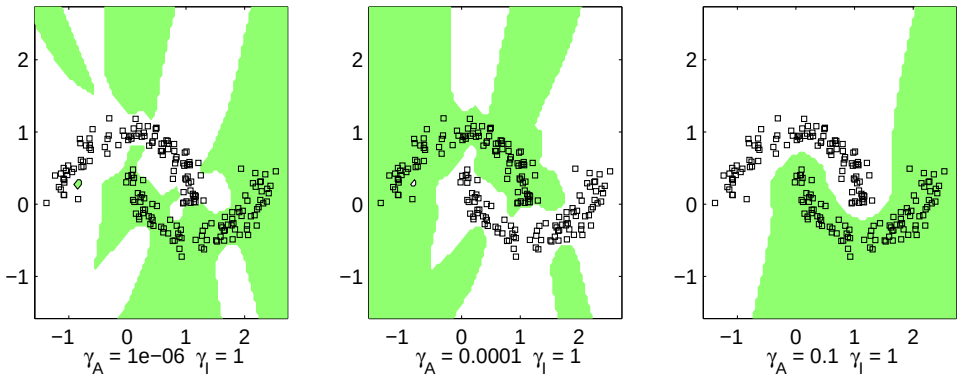
where $\mathbf{1}$ is the vector of all ones and $\alpha = (\alpha_1, \ldots, \alpha_u)$ and K is the corresponding Gram matrix.

Letting P be the projection onto the subspace of $\mathcal{R}^u$ orthogonal to $K\mathbf{1}$, one obtains the solution for the constrained quadratic problem, which is given by the generalized eigenvalue problem

$$P(\gamma K + \ KLK)P\mathbf{v} = \lambda PK^2P\mathbf{v} \tag{2.28}$$

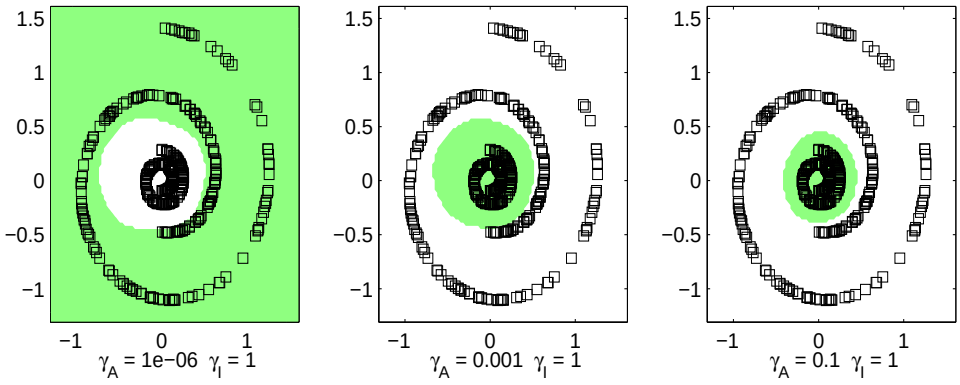
The final solution is given by $\alpha = P\mathbf{v}$, where $\mathbf{v}$ is the eigenvector corresponding to the smallest eigenvalue.

Figure 2.10: Two Moons Dataset: Regularized Clustering



Remark 1: The framework for clustering sketched above provides a regularized version of spectral clustering, where γ controls the smoothness of the resulting function in the ambient space. We also obtain a natural out-of-sample extension for clustering points not in the original data set. Figures 2.10, 2.11 show results of this method on two two-dimensional clustering problems. Unlike recent work [9, 21] on out-of-sample

Figure 2.11: Two Spirals Dataset: Regularized Clustering



extensions, our method is based on a Representer theorem for RKHS.

Remark 2: By taking multiple eigenvectors of the system in Eqn. 2.28 we obtain a natural regularized out-of-sample extension of Laplacian Eigenmaps. This leads to new method for dimensionality reduction and data representation. Further study of this approach is a direction of future research. We note that a similar algorithm has been independently proposed in [99] in the context of supervised graph inference. A relevant discussion is also presented in [53] on the interpretation of several geometric dimensionality reduction techniques as kernel methods.

2.8.2 Fully Supervised Learning

The fully supervised case represents the other end of the spectrum of learning. Since standard supervised algorithms (SVM and RLS) are special cases of manifold regularization, our framework is also able to deal with a labeled dataset containing no unlabeled examples. Additionally, manifold regularization can augment supervised learning with intrinsic regularization, possibly in a class-dependent manner, which

suggests the following algorithm :

$$f^* = \operatorname*{argmin}_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma_A \|f\|_K^2 + \frac{\gamma_I^+}{(u+l)^2} \mathbf{f}_+^T L_+ \mathbf{f}_+ + \frac{\gamma_I^-}{(u+l)^2} \mathbf{f}_-^T L_- \mathbf{f}_- \tag{2.29}$$

Here we introduce two intrinsic regularization parameters γ_I^+, γ_I^- and regularize separately for the two classes : $\mathbf{f}_+$, $\mathbf{f}_-$ are the vectors of evaluations of the function f, and L_+, L_- are the graph Laplacians, on positive and negative examples respectively. The solution to the above problem for RLS and SVM can be obtained by replacing $\gamma_I L$ by the block-diagonal matrix $\begin{pmatrix} \gamma_I^+ L_+ & 0 \\ 0 & \gamma_I^- L_- \end{pmatrix}$ in the manifold regularization formulas given in Section 4.

2.9 Extensions of Manifold Regularization

By setting $M = \frac{\gamma_I}{\gamma_A} L^p$, the modified kernel Eqn. 2.22 allows us to re-interpret manifold regularization within the standard framework of kernel methods. γ_A and γ_I are regularization parameters whose ratio controls the extent of the deformation. In particular, Laplacian SVM (LapSVM) and Laplacian RLS (LapRLS) become standard RLS and SVM using these kernels. Additionally, based on different choices of M, our approach provides a general algorithmic framework for incorporating useful domain structures (e.g invariances) in kernel methods. By utilizing Eqn. 2.22, manifold regularization may be implemented with kernel methods for a variety of semi-supervised regression (e.g., Support Vector Regression), one-class SVM, and Bayesian inference (Gaussian Processes, see [83]) tasks among other possible extensions.

CHAPTER 3

LOW-DENSITY CLASSIFICATION: NON-CONVEX METHODS

An intuitive approach to utilizing unlabeled data in kernel-based classification algorithms is to simply treat the unknown labels as additional optimization variables. For margin-based loss functions, one can view this approach as attempting to learn low-density separators. However, this is a hard optimization problem to solve in typical semi-supervised settings where unlabeled data is abundant. To examine the full potential of this class of methods modulo local minima problems, we apply branch and bound techniques for obtaining exact, globally optimal solutions. Empirical evidence suggests that the globally optimal solution can return excellent generalization performance in situations where other implementations fail completely due to suboptimal local minima. We present a novel deterministic annealing framework to alleviate local minima, where an easier optimization problem is parametrically deformed to the original hard problem and minimizers are smoothly tracked. This technique involves a sequence of convex optimization problems that are exactly and efficiently solved. We present empirical results on several synthetic and real world datasets that demonstrate the effectiveness of our approach.

3.1 The Semi-supervised SVM Formulation

A major line of research on extending SVMs to handle partially labeled datasets is based on the following idea: solve the standard SVM problem while treating the unknown labels as additional optimization variables. By maximizing the margin in the presence of unlabeled data, one learns a decision boundary that traverses through

low data-density regions while respecting labels in the input space. In other words, this approach implements the *cluster assumption* for semi-supervised learning – that points in a data cluster have similar labels. This idea was first introduced in [96] under the name *Transductive SVM*, but since it learns an inductive rule defined over the entire input space, we refer to this approach as *Semi-supervised SVM* (S^3VM).

We return to the problem of binary classification. The training set consists of l labeled examples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{l}$, $y_i = \pm 1$, and u unlabeled examples $\{\boldsymbol{x}_i\}_{i=l+1}^{n}$, with $n = l + u$. In an extended reglarization framework, the following minimization problem is solved in S^3VMs over *both* the RKHS $f \in \mathcal{H}_K$ and the label vector $\mathbf{y}_u := [y_{l+1} \dots y_n]^\top$,

$$\min_{f \in \mathcal{H}_K,\ \mathbf{y}_u} \mathcal{I}(f, \mathbf{y}_u) = \frac{\gamma}{2}\|f\|_K^2 + \frac{1}{2}\sum_{i=1}^{l} V(y_i, o_i) + \frac{\gamma'}{2}\sum_{i=l+1}^{n} V(y_i, o_i) \tag{3.1}$$

where $o_i = f(\boldsymbol{x}_i)$ and V is a loss function. For SVMs, V is commonly taken to be the Hinge loss or the squared Hinge loss.

The first two terms in the objective function $\mathcal{I}$ in (3.1) constitute a standard regularization functional. The third term incorporates unlabeled data. The loss over unlabeled examples is weighted by an additional hyperparameter, γ', which reflects confidence in the cluster assumption. In general, γ and γ' need to be set at different values for optimal generalization performance.

The minimization problem (3.1) is solved under the following class balancing constraint,

$$\frac{1}{u}\sum_{i=l+1}^{n} \max(y_i, 0) = r \quad \text{or equivalently} \quad \frac{1}{u}\sum_{i=l+1}^{n} y_i = 2r - 1. \tag{3.2}$$

This constraint helps in avoiding unbalanced solutions by enforcing that a certain

user-specified fraction, r, of the unlabeled data should be assigned to the positive class. It was introduced with the first S^{3}VM implementation [57]. Since the true class ratio is unknown for the unlabeled data, r is estimated from the class ratio on the labeled set, or from prior knowledge about the classification problem.

There are two broad strategies for minimizing $\mathcal{I}$:

1. **Combinatorial Optimization:** For a given fixed $\mathbf{y}_u$, the optimization over f is a standard minimization of a regularization objective, i.e., usual SVM training if the Hinge loss is being used[1]. Let us define,

$$\mathcal{J}(\mathbf{y}_u) = \min_{f,b} \quad \mathcal{I}(f, \mathbf{y}_u) \tag{3.3}$$

   The goal now is to minimize $\mathcal{J}$ over a set of binary variables. This combinatorial view of the optimization problem is adopted in [57, 13, 105, 84, 10, 27].

2. **Continuous Optimization:**

   We define the effective loss function V' over an unlabeled example $\boldsymbol{x}$ as $V'(f(\boldsymbol{x})) = \min\left[V(f(\boldsymbol{x})), V(-f(\boldsymbol{x}))\right]$ corresponding to making an optimal choice for the unknown label of $\boldsymbol{x}$. Thus, one can formulate an equivalent continuous optimization problem over $\mathcal{H}_K$ alone, with V and V' as the loss functions over labeled and unlabeled examples respectively. The continuous objective function is,

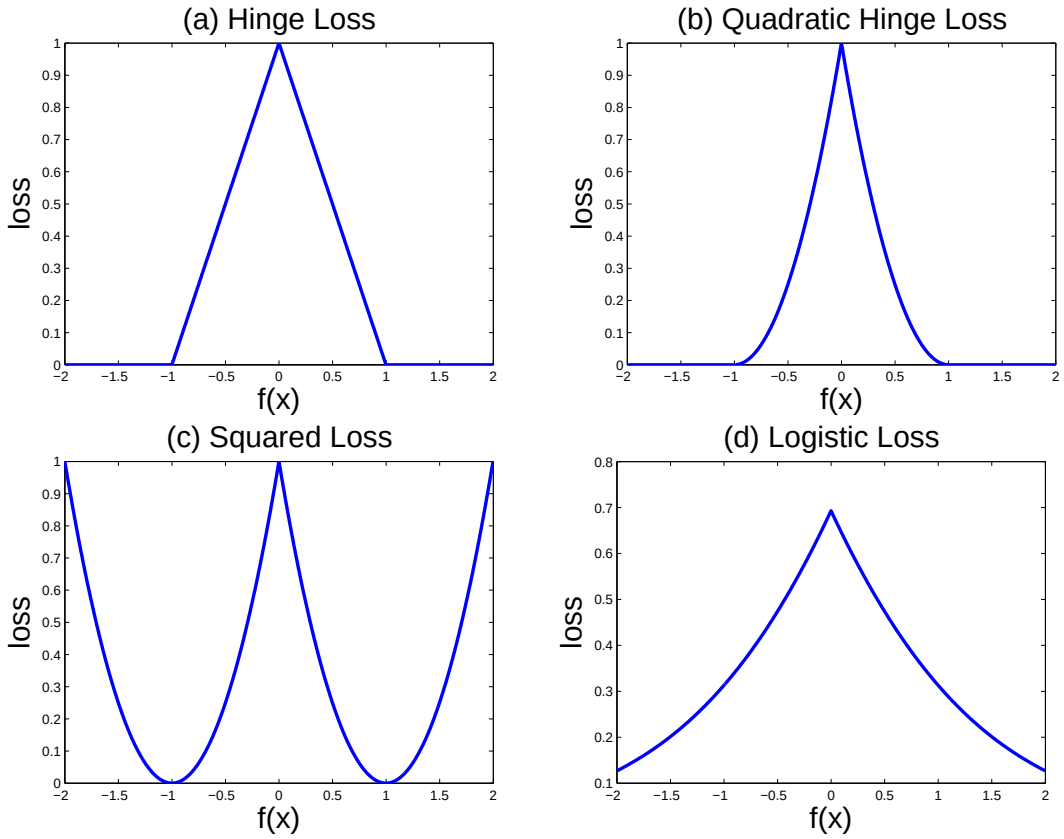
$$\mathcal{J}(f) = \frac{\gamma}{2}\|f\|_K^2 + \frac{1}{2}\sum_{i=1}^{l} V(y_i, f(\boldsymbol{x}_i)) + \frac{\gamma'}{2}\sum_{i=l+1}^{n} V'(f(\boldsymbol{x}_i)) \tag{3.4}$$

   Figure 3.1 shows the shape of V' for common choices of V. Since small outputs are penalized, decision boundaries that pass through a low-density region in the

1. The SVM training is slightly modified to take into account γ' weighted examples

input space are preferred. Thus, this may be seen as a general approach for semi-supervised learning based on the *cluster assumption*: the assumption that the true decision boundary does not cut data clusters.

Figure 3.1: Effective Loss function V'



The combinatorial nature of this problem due to discrete label variables, or equivalently, the non-convexity of the effective loss function V' make Eqn. 3.1 a difficult problem. Currently available approaches [96, 10] for global optimization of Eqn 3.1 for S^3VMs are unrealistically slow for typical semi-supervised problems where unlabeled data is plentiful. On the other hand, many recent techniques [57, 30, 36, 50] are susceptible to local minima on difficult real world classification problems.

The main contributions of this chapter are summarized as follows:

1. We analyze the nature of the difficulty of solving S^3VMs motivated by the following questions: How well do current S^3VM implementations approximate the exact, globally optimal solution of the non-convex problem associated with S^3VMs ? Can one expect significant improvements in generalization performance by better approaching the global solution ? We believe that these questions are of fundamental importance for S^3VM research and are largely unresolved. This is partly due to the lack of simple implementations that practitioners can use to benchmark new algorithms against the global solution, even on small-sized problems.

   We outline a class of Branch and Bound algorithms that are guaranteed to provide the globally optimal solution for S^3VMs. Branch and bound techniques have previously been noted in the context of S^3VM in [104], but no details were presented there. We implement and evaluate a branch and bound strategy that can serve as an upper baseline for S^3VM algorithms. This strategy is not practical for typical semi-supervised settings where large amounts of unlabeled data is available. But we believe it opens up new avenues of research that can potentially lead to more efficient variants.

   Empirical results on some semi-supervised tasks presented in section 3.2.3 show that the exact solution found by branch and bound has excellent generalization performance, while other S^3VM implementations perform poorly. These results also show that S^3VM can compete and even outperform graph-based techniques (e.g., [110, 86]) on problems where the latter class of methods have typically excelled.

2. We present a deterministic annealing framework for global optimization of ob-

jective functions of the form in Eqn 3.1 that scales to larger datasets while providing some resistance to suboptimal local minima. Our approach generates a sequence of optimization problems approaching the given problem with gradually increasing complexity. These objective functions are locally minimized; the solution for one problem is seeded to the next as the initial guess. This strategy falls in a class of homotopy optimization methods, e.g., see [72, 45], and can be also interpreted in terms of maximum entropy principles and deterministic variants of stochastic search techniques [75, 55]. A related class of techniques is the Graduated Non-Convexity method of [16]. Some recent work on semi-supervised learning with similar motivation appears in [24].

3. We derive and evaluate an alternating convex optimization procedure within this framework. This method can utilize off-the-shelf optimization techniques for regularization algorithms. For example, it yields a large scale semi-supervised L_2-SVM for sparse, linear settings [85] when implemented in conjunction with specialized primal methods [62] (see Chapter 6).

4. We present an experimental study demonstrating the importance of the idea of annealing on semi-supervised tasks. On some difficult classification problems, our methods show significant improvements over competing algorithms. Whereas recent efforts for solving Eqn. 3.1 have largely focussed on margin loss functions, our experimental study shows that the classical squared loss can also be very effective for semi-supervised learning within this framework.

This chapter is arranged as follows: In section 3.2 we outline our Branch and Bound approach benchmarking it in sub-section 3.2.3. In section 3.4.1 we outline homotopy methods and deterministic annealing (DA) for global optimization. DA

is presented in section 3.5 and its empirical performance is described in Section 3.6. Section 3.7 concludes this chapter.

3.2 Globally Optimal Branch & Bound Solution

3.2.1 Branch and bound basics

Suppose we want to minimize a function f over a space $\mathcal{X}$, where $\mathcal{X}$ is discrete. A branch and bound algorithm has two main ingredients:

Branching : the region $\mathcal{X}$ is recursively split into smaller subregions. This yields a *tree* structure where each node corresponds to a subregion.

Bounding : consider two (disjoint) subregions (i.e. nodes) A and $B \subset \mathcal{X}$. Suppose that an upper bound (say a) on the best value of f over A is known and a lower bound (say b) on the best value of f over B is known and that $a < b$. Then, we know there is an element in the subset A that is better than all elements of B. So, when searching for the global minimizer we can safely discard the elements of B from the search: the subtree corresponding to B is *pruned*.

3.2.2 Branch and bound for S^3 VM

Recall the combinatorial formulation of S^3VMs in Eqn 3.3. Branch-and-Bound effectively performs an exhaustive search over $\mathbf{y}_u$, pruning large parts of the solution space based on the following simple observation: Suppose that a lower bound on $\min_{\mathbf{y}_u \in A} \mathcal{J}(\mathbf{y}_u)$, for some subset A of candidate solutions, is greater than $\mathcal{J}(\tilde{\mathbf{y}}_u)$ for some $\tilde{\mathbf{y}}_u$, then A can be safely discarded from exhaustive search. The algorithm organizes subsets of solutions into a binary tree (Figure 3.2[2]) where nodes are associated

2. Figure generated by Olivier Chapelle.

with a fixed partial labeling of the unlabeled data set and the two children correspond to the labeling of some new unlabeled point. Thus, the root corresponds to the initial set of labeled examples and the leaves correspond to a complete labeling of the data. Any node is then associated with the subset of candidate solutions that appear at the leaves of the subtree rooted at that node (all possible ways of completing the labeling, given the partial labeling at that node). This subset can potentially be pruned from the search by the Branch-and-Bound procedure if a lower bound over corresponding objective values turns out to be worse than an available solution.

The effectiveness of pruning depends on the following design issues: (1) the lower *bound* at a node and (2) the sequence of unlabeled examples to *branch* on. For the lower bound, we use the objective value of a standard SVM trained on the associated (extended) labeled set[3]. As one goes down the tree, this objective value increases as additional loss terms are added, and eventually equals $\mathcal{J}$ at the leaves. Note that once a node violates the balance constraint, it can be immediately pruned by resetting its lower bound to ∞. We use a labeling-confidence criterion to choose an unlabeled example and a label on which to branch. The tree is explored on the fly by depth-first search. This confidence-based tree exploration is also intuitively linked to Label Propagation methods [110] for graph-transduction. On many small datasets datasets, the branch and bound procedure is able to return the globally optimal solution in reasonable amount of time.

We point the reader to [27] for pseudocode of our algorithm and more details.

3. Note that in this SVM training, the loss terms associated with (originally) labeled and (currently labeled) unlabeled examples are weighted 1 and γ' respectively.

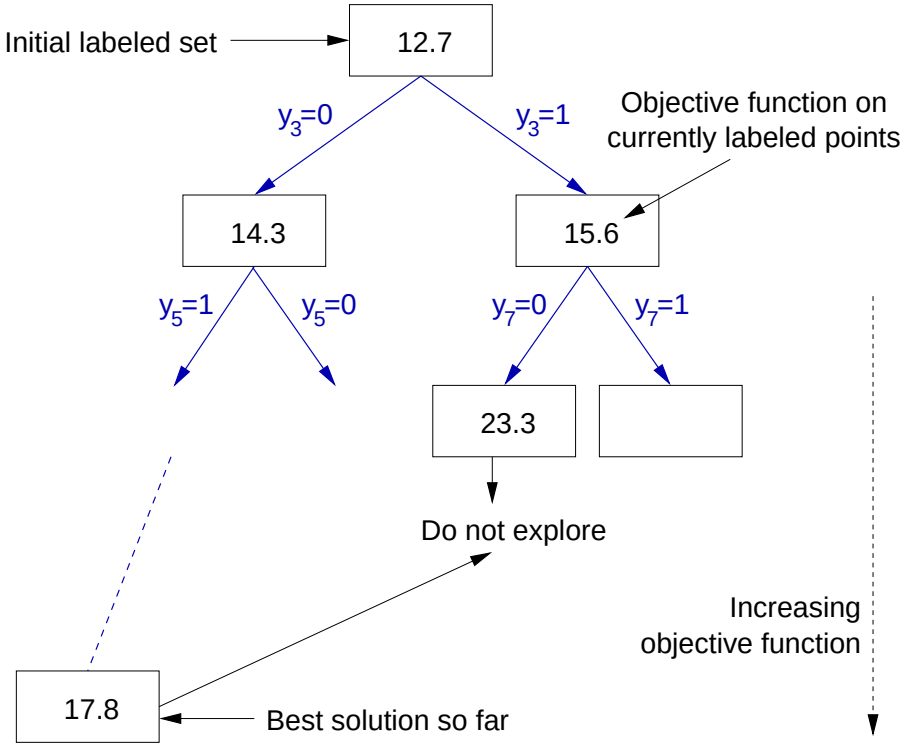


Figure 3.2: Branch-and-Bound Tree

3.2.3 Generalization Performance at Global vs Local Minima

We consider here two datasets where other S^3VM implementations are unable to achieve satisfying test error rates. This naturally raises the following questions: Is this weak performance due to the unsuitability of the S^3VM objective function for these problems or do these methods get stuck at highly sub-optimal local minima ?

Two moons

The "two moons" dataset is now a standard benchmark for semi-supervised learning algorithms. Most graph-based methods easily solve this problem as was demonstrated in the previous chapter, but so far, all S^3VM algorithms find it difficult to construct the right boundary (an exception is a deterministic annealing approach[84] using an

L_1 loss that we describe later in this chapter). We drew 100 random realizations of this dataset, fixed the bandwidth of an RBF kernel to $\sigma = 0.5$ and set $\gamma = 0.1, \gamma' = 1$.

We compared ∇S^{3}VM[30], cS^{3}VM[24], CCCP [36], SVM^{light} [57] and DA [84]. For the first 3 methods, there is no direct way to enforce the constraint (3.2). However, these methods have a constraint that the mean output on the unlabeled point should be equal to some constant. This constant is normally fixed to the mean of the labels, but for the sake of consistency we did a dichotomy search on this constant in order to have (3.2) satisfied.

Results are presented in Table 3.1. Note that the test errors for other S^{3}VM implementations are likely to be improved by hyperparameter tuning, but they will still stay very high. For comparison, we have also included the results of a state-of-the-art graph based method, LapSVM [86] whose hyperparameters were optimized for the test error and the threshold adjusted to satisfy the constraint (3.2).

Table 3.1: Results on the two moons dataset (averaged over 100 random realizations)

	Test error (%)	Objective function
∇S^{3}VM	59.3	13.64
cS^{3}VM	45.7	13.25
CCCP	64	39.55
SVM^{light}	66.2	20.94
DA	34.1	46.85
Branch-Bound	0	7.81
LapSVM	3.7	N/A

Three Cars

Extensive benchmark results reported in [26, benchmark chapter] show that on problems where classes are expected to reside on low-dimensional non-linear manifolds,

e.g., handwritten digits, graph-based algorithms significantly outperform S^3VM implementations.

We consider here such a dataset by selecting three confusible classes from the COIL20 dataset [30] (see figure 3.3). There are 72 images of cars per class, corresponding to rotations of 5 degrees (and thus yielding a one dimensional manifold). We randomly selected 2 images per class to be in the labeled set and the rest being unlabeled. Results are reported in table 3.2. The hyperparameters were chosen to be $\sigma = 3000$ and $\gamma = 0.01, \gamma' = 1$.

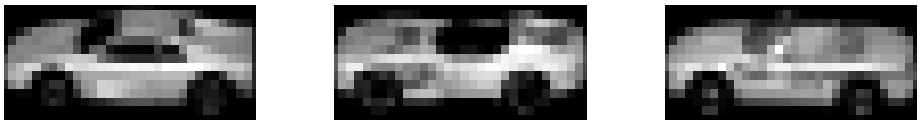


Figure 3.3: The Three Cars dataset, subsampled to 32×32

Table 3.2: Results on the Three Cars dataset (averaged over 10 random realizations)

	Test error (%)	Objective function
∇S^3VM	60.6	267.4
cS^3VM	60.6	235
CCCP	47.5	588.3
SVM^{light}	55.3	341.6
DA	48.2	611
Branch-Bound	0	110.7
LapSVM	7.5	N/A

From Tables 3.1 and 3.2, it appears clearly that (1) the S^3VM objective function leads to excellent test errors; (2) other S^3VM implementations fail completely in finding a good minimum of the objective function and (3) the global S^3VM solution can actually outperform graph-based alternatives even if other S^3VM implementations are not found to be competitive.

3.3 Extensions of Branch and Bound Algorithm

This basic implementation can perhaps be made more efficient by choosing better bounding and branching schemes. Also, by fixing the upper bound as the currently best objective value, we restricted our implementation to follow depth-first search. It is conceivable that breadth-first search is equally or more effective in conjunction with alternative upper bounding schemes. Pruning can be done more aggressively to speed-up termination at the expense of obtaining a solution that is suboptimal within some tolerance (i.e prune B if $a < b - \epsilon$). Finally, we note that a large family of well-tested branch and bound procedures from zero-one quadratic programming literature can be immediately applied to the S^3VM problem for the special case of squared loss. An interesting open question is whether one can provide a guarantee for polynomial time convergence under some assumptions on the data and the kernel.

Concerning the running time of our current implementation, we have observed that it is most efficient whenever the global minimum is significantly smaller than most local minima: in that case, the tree can be pruned efficiently. This happens when the clusters are well separated and γ, γ', σ are in an appropriate range.

3.4 A Deterministic Annealing Formulation

The Branch and Bound implementation does not scale to large datasets, but should instead be considered as a proof of concept: the S^3VM objective function is very well suited for semi-supervised learning and more effort is justified on trying to efficiently find good local minima. With this motivation, in this section we develop a deterministic framework for solving S^3VMs.

3.4.1 Homotopy Methods

The intuition for the deterministic annealing framework for global optimization is simply stated in the following. Consider an unconstrained optimization problem: find $\mathbf{u}^\star = \text{argmin}_{\mathbf{u}\in\mathcal{R}^n}\, \mathcal{C}(\mathbf{u})$ where the objective function $\mathcal{C}(\mathbf{u})$ may have many possible local minima. Instead of directly dealing with such a problem, we first construct a related "easier" objective function $c(\mathbf{u})$. The minimizers of this function are either known, or easy to compute, for example due to convexity. We then gradually deform the easy problem to the original problem by specifying a smooth map, i.e a homotopy $\mathcal{J}(\mathbf{u}, T)$ parameterized by a real-valued variable T, so that $\mathcal{J}(\mathbf{u}, t_1) = c(\mathbf{u})$ and $\mathcal{J}(\mathbf{u}, t_2) = \mathcal{C}(\mathbf{u})$. Typically, one chooses a convex homotopy such as $\mathcal{J}(\mathbf{u}, T) = (1 - T)\ \mathcal{C}(\mathbf{u}) + T\ c(\mathbf{u})$ where $0 \leq T \leq 1$, or a global homotopy such as $\mathcal{J}(\mathbf{u}, T) = \mathcal{C}(\mathbf{u}) + Tc(\mathbf{u})$ where $T \in [0, \infty)$. In practice, T may be varied over an interval starting from t_1 and ending at t_2, in fixed additive steps or by a fixed multiplicative factor. We track local minimizers along the deformation path, at each point starting from the previously computed solution.

Clearly, whether or not this method succeeds in finding a global minimum of $\mathcal{C}(\mathbf{u})$ depends strongly on the choice of the map $\mathcal{J}(\mathbf{u}, T)$ and the manner in which T is varied. For general choices for these, one cannot guarantee a global minimum since there need not be a path in the variable T connecting the sequence of local minimizers to the global minimum, and even if there is one, there is no apriori guarantee that the minimum reached at $T = t_2$ is globally optimal. Moreover, in general, the optimal deformation protocol need not be monotonically increasing or decreasing in T. Inspite of these limitations, in typical applications, it is often more natural to construct $c(\mathbf{u})$ than to find good starting points. A good choice of the homotopy function and deformation protocol and can drastically reduce local minima problems in the starting

and middle stages of the optimization process allowing the focus to be on the globally relevant features of the original objective function.

3.4.2 Deterministic Annealing

Deterministic annealing may be viewed as a homotopy method for dealing with combinatorial optimization problems. This approach involves two steps. In the first step, discrete variables are treated as random variables over which a space of probability distributions is defined. In the second step, the original problem is replaced by a continuous optimization problem of finding a distribution in this space that minimizes the expected value of the objective function. The latter optimization is performed by a homotopy method using negative of the entropy of a distribution as the "easy", convex function. Specifically, one solves:

$$\mathbf{p}^\star = \underset{\mathbf{p}\in\mathcal{P}}{\operatorname{argmin}}\, E_{\mathbf{p}}\mathcal{C}(\mathbf{u}) - TS(\mathbf{p}) \tag{3.5}$$

where $\mathbf{u} \in \{0,1\}^n$ are the discrete variables for the objective function $\mathcal{C}(\mathbf{u})$, $\mathcal{P}$ is a family of probability distributions over $\mathbf{u}$, $E_{\mathbf{p}}$ denotes expectation with respect to a distribution $\mathbf{p}$ and $S(\mathbf{p})$ denotes the entropy of $\mathbf{p}$. Note that if $T = 0$ and $\mathcal{P}$ contains all possible point-mass distributions on $\{0,1\}^n$, then the global minimizer $\mathbf{p}^\star$ puts all its mass on the global minimizer of $\mathcal{C}(\mathbf{u})$. Factorial distributions where the associated random variables are taken to be independent are one such class of distributions. With such a choice for $\mathcal{P}$, the first step of "relaxation" to continuous variables does not lose any optimality. The task of finding a minimizer close to the global minimizer in $\mathcal{P}$ is then left to the homotopy method in the second step.

The choice of entropy in the homotopy is well-motivated in various other ways. If T is non-zero and $\mathcal{P}$ is unrestricted, the minimizer in Eqn. 3.7 is given by the Gibbs

distribution $p^{\star}_{gibbs}(\mathbf{u}) = \frac{\exp(-\mathcal{C}(\mathbf{u})/T)}{\sum_{\{0,1\}^n}\exp(-\mathcal{C}(\mathbf{u})/T)}$. As $T \mapsto 0$, the Gibbs distribution begins to concentrate its mass on the global minimizer of $\mathcal{C}(\mathbf{u})$. Therefore, a stochastic optimization strategy, simulated annealing [64], samples candidate solutions from a Markov process whose stationary distribution is the Gibbs distribution, while gradually lowering T. In deterministic annealing, one attempts to find a distribution in $\mathcal{P}$ that is closest to the Gibbs distribution in the sense of KL-divergence, resulting in an optimization problem that is equivalent to Eqn. 3.7 [14]. Finally, one can also interpret this approach in terms of maximum entropy inference [75].

3.5 Deterministic Annealing for S^{3}VMs

We now apply deterministic annealing for solving Eqn. 3.1 which involves a mix of discrete and continuous variables. The discussion above motivates a continuous objective function,

$$\mathcal{J}_T(f, \mathbf{p}) = E_{\mathbf{p}}\mathcal{I}(f, \mathbf{y}_u) - TS(\mathbf{p}) \tag{3.6}$$

defined by taking the expectation of $\mathcal{I}(f, \mathbf{y}_u)$ (Eqn. 3.1) with respect to a distribution $\mathbf{p}$ on $\mathbf{y}_u$ and including entropy of $\mathbf{p}$ as a homotopy term. Thus, we have:

$$\begin{aligned}\mathcal{J}_T(f, \mathbf{p}) = \frac{\lambda}{2}\|f\|_K^2 + \frac{1}{l}\sum_{i=1}^{l} V\left(y_i f(x_i)\right) \\ +\frac{\lambda'}{u}\sum_{j=1}^{u}\left[p_j V\left(f(\boldsymbol{x}'_j)\right) + (1-p_j)V\left(-f(\boldsymbol{x}'_j)\right)\right] \\ +\frac{T}{u}\sum_{j=1}^{u}\left[p_j \ \log\ p_j + (1-p_j)\ \log\ (1-p_j)\right]\end{aligned} \tag{3.7}$$

where $\mathbf{p} = (p_1 \ldots p_u)$ and p_i may be interpreted as the probability that $y'_i = 1$.

This objective function for a fixed T is minimized under the following class bal-

ancing constraint, in place of the balance constraint in Eqn. 3.2:

$$\frac{1}{u}\sum_{j=1}^{u} p_j = r \tag{3.8}$$

As in the usual formulations, r is treated as a user-provided parameter. It may also be estimated from the labeled examples.

The solution to the optimization problem above

$$(f_T^\star, \mathbf{p}_T^\star) = \underset{f\in\mathcal{H}_K, \mathbf{p}\in[0,1]^u}{\operatorname{argmin}} \mathcal{J}_T(f, \mathbf{p})$$

is tracked as the parameter T is lowered to 0. The final solution is given as $f^\star = \lim_{T\to 0} f_T^\star$. In practice, we monitor the value of the objective function in the optimization path and return the solution corresponding to the minimum value achieved.

3.5.1 Optimization

For any fixed value of T, the problem in Eqns. 3.7, 3.8 is optimized by alternating the minimization over $f \in \mathcal{H}_K$ and $\mathbf{p} \in [0,1]^u$ respectively. Fixing $\mathbf{p}$, the minimization over f can be done by standard techniques for solving weighted regularization problems. Fixing f, the minimization over $\mathbf{p}$ can also be done easily as described below. While the original problem is non-convex, keeping one block of variables fixed yields a convex optimization problem over the other block of variables. Both these convex problems can be solved exactly and efficiently. An additional advantage of such block optimization is that it allows efficient algorithms for training kernel classifiers to be used directly within the deterministic annealing procedure. We note that a similar alternating optimization scheme was proposed in [50] in the context of semi-supervised logistic regression. We now provide some details.

Optimizing f for fixed $\mathbf{p}$

By the Representer theorem, the minimizer over $f \in \mathcal{H}_K$ of the objective function in Eqn. 3.7 for fixed $\mathbf{p}$ is given as:

$$f(\boldsymbol{x}) = \sum_{i=1}^{l} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i) + \sum_{j=1}^{u} \alpha_{l+j} K(\boldsymbol{x}, \boldsymbol{x}'_j) \tag{3.9}$$

The coefficients $\boldsymbol{\alpha} = (\alpha_1 \dots \alpha_{l+u})$ can be computed by solving a finite dimensional optimization problem that arises by substituting this expression in Eqn. 3.7 where the norm $\|f\|_K^2 = \boldsymbol{\alpha}^T K \boldsymbol{\alpha}$. The resulting objective function in $\boldsymbol{\alpha}$ is denoted as $\mathcal{J}_T(\boldsymbol{\alpha}, \mathbf{p})$. Below we explicitly write down the solutions for two common choices of loss functions. One can also solve for $\boldsymbol{\alpha}$ using other optimization techniques and for other choices of loss functions.

Regularized Least Squares (RLS)

For Regularized Least squares (RLS), $V(t) = (1-t)^2/2$. Setting $\nabla_{\boldsymbol{\alpha}} \mathcal{J}_T(\boldsymbol{\alpha}, \mathbf{p}) = 0$ and solving for $\boldsymbol{\alpha}$ we obtain:

$$\boldsymbol{\alpha} = (G + \lambda C)^{-1} Y \tag{3.10}$$

where $Y = [y_1 \dots y_l, (2p_1 - 1) \dots (2p_u - 1)]^T$, G is the gram matrix over the $l+u$ points, C is a diagonal matrix whose first l diagonal entries are l and remaining u diagonal entries are u/λ'. In Eqn. 3.10, note that the matrix $(G + \lambda C)^{-1}$ is independent of $\mathbf{p}$ and therefore needs to be computed only once. Subsequent updates in the iteration only involve matrix vector products. Thus, if the inverse $(G + \lambda C)^{-1}$ can be formed, the updates for $\boldsymbol{\alpha}$ in the deterministic annealing iterations are very cheap.

SVM with Quadratic Hinge Loss

The quadratic hinge loss function is given by $V(t) = \max(0, 1-t)^2/2$. We apply

the primal finite newton methods from [62, 23] to solve Eqn. 3.7 with this loss. A sequence of candidate solutions $\{\boldsymbol{\alpha}^{(k)}\}$ is generated as follows. For any $\boldsymbol{\alpha}^{(k)}$ in the sequence, denote the output, as given by Eqn. 3.9, on any example $\boldsymbol{x}$ as $f^{(k)}(\boldsymbol{x})$ and define the following index sets: $i_0 = \{i : y_i f^{(k)}(\boldsymbol{x}_i) < 1\}$, $i_1 = \{j : f^{(k)}(\boldsymbol{x}'_j) \leq -1\}$, $i_2 = \{j : f^{(k)}(\boldsymbol{x}'_j) \geq 1\}$, and $i_3 = \{j : |f^{(k)}(\boldsymbol{x}'_j)| < 1\}$.

Consider the following objective function in the variable $\boldsymbol{\alpha}$:

$$\mathcal{J}^{(k)}(\boldsymbol{\alpha}) = \frac{\lambda}{2}\boldsymbol{\alpha}^T K \boldsymbol{\alpha} + \frac{1}{l}\sum_{i_0} V_s\left(y_i f(x_i)\right)$$
$$+\frac{\lambda'}{u}\left[\sum_{i_1} p_j V_s\left(f(\boldsymbol{x}'_j)\right) + \sum_{i_2}(1-p_j)V_s\left(-f(\boldsymbol{x}'_j)\right)\right.$$
$$\left. + \sum_{i_3} p_j V_s\left(f(\boldsymbol{x}'_j)\right) + (1-p_j)V_s\left(-f(\boldsymbol{x}'_j)\right)\right]$$

where $V_s(t) = (1-t)^2/2$. This objective function is a local quadratic approximation of the objective function Eqn. 3.7 and simply involves squared loss terms. Denote $\bar{\boldsymbol{\alpha}} = \text{argmin}_{\boldsymbol{\alpha}}\, \mathcal{J}^{(k)}(\boldsymbol{\alpha})$. This can be computed by solving a linear system that arises by setting $\nabla_{\boldsymbol{\alpha}}\mathcal{J}^{(k)}(\boldsymbol{\alpha}) = 0$. Finally, obtain $\boldsymbol{\alpha}^{(k+1)} = \boldsymbol{\alpha}^{(k)} + \delta^\star(\bar{\boldsymbol{\alpha}} - \boldsymbol{\alpha}^{(k)})$ where the step length $\delta^\star$ is obtained by performing an exact line search by solving the one-dimensional problem $\delta^\star = \text{argmin}_{\delta>0}\, \mathcal{J}_T(\boldsymbol{\alpha} + \delta(\bar{\boldsymbol{\alpha}^{(k)}} - \boldsymbol{\alpha}^{(k)}), \mathbf{p})$. This can be done using efficient recursive updates as outlined in [62]. From the arguments in [62], it can be shown that the sequence $\{\boldsymbol{\alpha}^k\}$ starting from any initial point converges in a finite number of steps to the minimizer (in $\boldsymbol{\alpha}$) of $J_T(\boldsymbol{\alpha}, \mathbf{p})$ for a given fixed $\mathbf{p}$. By starting the optimization from the solution of the previous DA iteration ("seeding"), the convergence can be very fast.

Large Scale Implementations

In the case of linear kernels, instead of using Eqn. 3.9 we can directly write $f(\boldsymbol{x}) =$

$\mathbf{w}^T\mathbf{x}$ where updates for the weight vector $\mathbf{w}$ are obtained by the finite Newton procedure outlined above. For large scale problems such as text classification where $(l+u)$ as well as the dimension of $\mathbf{x}$ are possibly large and the data matrix consisting of the $\mathbf{x}_i$ has only a small fraction of nonzero elements, effective conjugate gradient schemes can be used to implement the updates for $\mathbf{w}$. The result is an impressively fast algorithm for such problems. See Chapter 6 for full details.

Optimizing $\mathbf{p}$ for fixed f

For the latter problem of optimizing $\mathbf{p}$ for a fixed f, we construct the Lagrangian: $\mathcal{L} = \mathcal{J}_T(f, \mathbf{p}) - \nu(\frac{1}{u}\sum_{j=1}^{u} p_j - r)$. Solving $\partial\mathcal{L}/\partial p_j = 0$, we get:

$$p_j = \frac{1}{1 + e^{\frac{g_j - \nu}{T}}} \tag{3.11}$$

where $g_j = \lambda'[V(f(\boldsymbol{x}'_j)) - V(-f(\boldsymbol{x}'_j))]$. Substituting this expression in the balance constraint in Eqn. 3.8, we get a one-dimensional non-linear equation in ν: $\frac{1}{u}\sum_{j=1}^{u} \frac{1}{1+e^{\frac{g_i-\nu}{T}}} = r$. The root is computed exactly by using a hybrid combination of Newton-Raphson iterations and the bisection method together with a carefully set initial value.

For a fixed T, the alternate minimization of $f \in \mathcal{H}_K$ and $\mathbf{p}$ proceeds until some stopping criterion is satisfied. A natural criterion is the KL-divergence between values of $\mathbf{p}$ in consecutive iterations. The parameter T is decreased in an outer loop until the total entropy falls below a threshold. Table 3.3 outlines the steps for the algorithm with default parameters. In the rest of this paper, we will abbreviate our method as DA (*loss*) where *loss* is l_1 for hinge loss, l_2 for quadratic hinge loss and *sqr* for squared loss.

Table 3.3: Semi-supervised Learning with Deterministic Annealing.

Inputs:	$\{\boldsymbol{x}_i, y_i\}_{i=1}^{l}, \{\boldsymbol{x}'_j\}_{j=1}^{u}, \lambda, \lambda', r$
Initialize:	Set $\mathbf{p} = (r, \ldots, r) \in \mathcal{R}^u \quad \mathbf{q} = \mathbf{p}$
	Set $T = 10 \quad R = 1.5 \quad \epsilon = 10^{-6}$
loop1	while $S(\mathbf{p}) > \epsilon$ (S denotes entropy)
loop2	while $KL(\mathbf{p}, \mathbf{q}) > \epsilon$ (KL denotes KL-divergence)
	Update $\boldsymbol{\alpha}$ by solving Eqn. 3.7 for fixed $\mathbf{p}$
	Set $\mathbf{q} = \mathbf{p}$
	Set $\mathbf{p}$ according to Eqn. 3.11
	end loop1
	$T = T/R$
	end loop2
	Return $f(\boldsymbol{x}) = \sum_{i=1}^{l} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i) + \sum_{j=1}^{u} \alpha_{l+j} K(\boldsymbol{x}, \boldsymbol{x}'_j)$

3.5.2 Annealing Behaviour of Loss functions

We can develop a good intuition for the working of our method by ignoring the balancing constraint in Eqn. 3.8 and putting together the loss terms in Eqn. 3.7 for a single unlabeled example $\boldsymbol{x}'_j$:

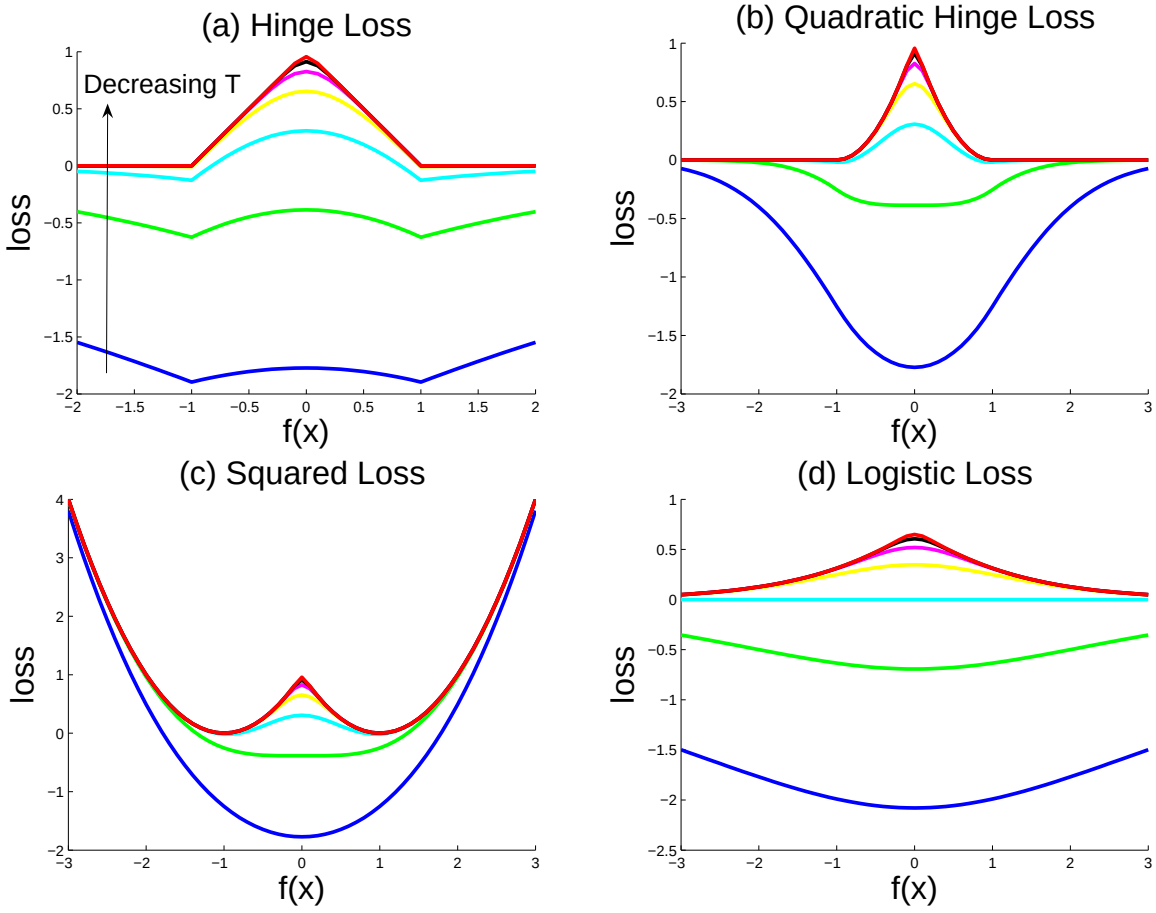
$$\begin{aligned}\Phi_T(f(\boldsymbol{x}'_j), p_j) = p_j V(f(\boldsymbol{x}'_j)) + (1 - p_j) V(-f(\boldsymbol{x}'_j)) \\ + T[p_j \log p_j + (1 - p_j) \log(1 - p_j)]\end{aligned}$$

Keeping f fixed, the optimal value of p_j, say $p_j^\star$, is given by Eqn. 3.11 (with $\nu = 0$). The effective loss function then becomes $V'_T(f(\boldsymbol{x}'_j)) = \Phi_T(f(\boldsymbol{x}'_j), p_j^\star)$.

In Figure 3.4, we plot V'_T as a function of $f(\boldsymbol{x}'_j)$ for different settings of T. The sub-plots show the behavior of V'_T for common choices of V.

As T is decreased from high to low values, we see interestingly different behavior for different loss functions with respect to their shape in the "inner" interval $[-1, 1]$

Figure 3.4: Annealing behavior of loss functions parameterized by T.



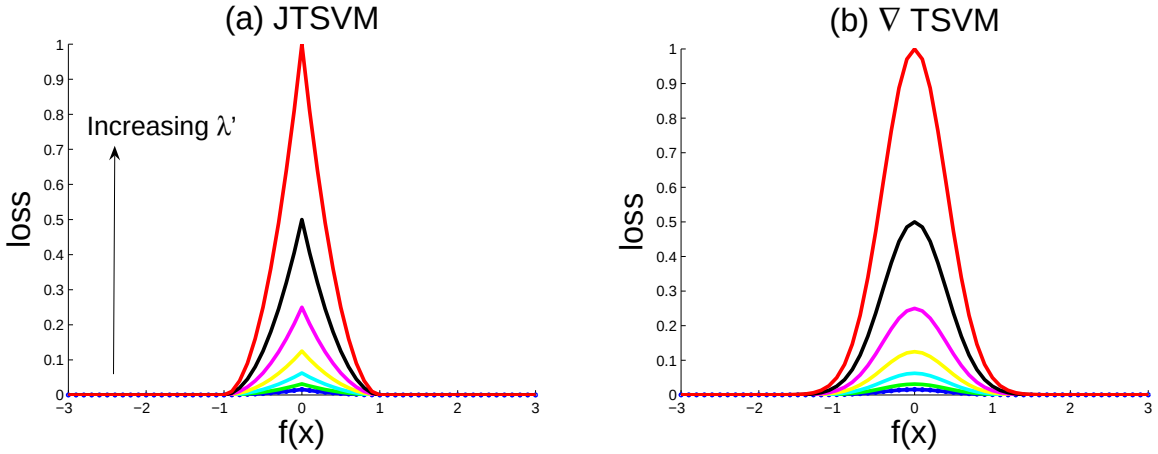
(within the margin) and "outer" interval (outside the margin).

We see that at high values of T, the hinge loss has a sharp upward slope in the outer interval and is almost constant in the inner interval. The other loss functions are unimodal with a minimum at the decision boundary. As T is decreased, the effective loss V'_T gradually deforms into the effective loss V' in the original objective function in Eqns. 3.1,3.4 (also see Figure 3.1).

The Transductive SVM implementations of [57, 30] also solve a sequence of optimization problems with gradually increasing values of λ'. We refer to these implementations as JTSVM and ∇TSVM respectively. The respective effective loss functions are shown in Figure 3.5. We see that in all stages of the optimization, unlabeled ex-

amples in the outer interval do not influence the decision boundary. Other approaches for Transductive SVM, e.g., [50, 36] do not discuss such an annealing component.

Figure 3.5: Loss functions for JTSVM and ∇TSVM parameterized by λ'.



To examine the effectiveness of different annealing strategies and loss functions, we performed experiments on two toy datasets, 2MOONS and 2CIRCLES, with highly non-linear cluster structures. These datasets are shown in Figure 3.6 (a particular labeling is shown). For 10 random choices of 2 labeled examples, we recorded the number of times JTSVM, ∇TSVM and DA produced a decision boundary perfectly classifying the unlabeled data. For JTSVM and DA we report results for Hinge loss (l_1) and quadratic hinge loss (l_2).

The experiment was conducted with RBF kernels. In Table 3.4, we report the best performance for each method over a grid of parameters. We see that DA outperforms ∇TSVM which performs better than JTSVM. In our experiments, DA with Hinge loss succeeded in every trial for both 2CIRCLES and 2MOONS. On the other hand JTSVM failed everytime on 2MOONS and succeeded once on 2CIRCLES.

Figure 3.6:

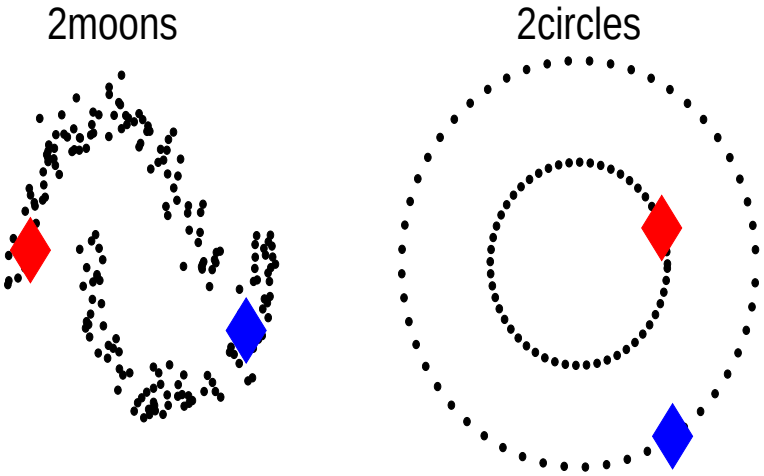


Table 3.4: Number of successes out of 10 trials.

Dataset → Algorithm ↓	2MOONS	2CIRCLES
JTSVM (l_2)	0	1
JTSVM (l_1)	0	1
∇TSVM	3	2
DA (l_2)	6	3
DA (l_1)	**10**	**10**

3.6 Empirical Results

We present an experimental study on a collection of 5 datasets listed in Table 3.5. USPS2 is a subset of the USPS dataset with images of digits 2 and 5. COIL6 is a 6-class dataset derived from a collection of images of objects viewed from different angles. PC-MAC is a subset of the 20-newsgroup text dataset posing the task of categorizing newsgroup documents into two topics: *mac* or *windows*. Finally, ESET2 is a subset of the ISOLET dataset consisting of acoustic features of isolated spoken utterances of 9 confusible letters $\{B, C, D, E, G, P, T, V, Z\}$. We considered the binary classification task of separating the first 4 letters from the rest.

Table 3.5: Datasets with d features, l labeled examples, u unlabeled examples, v validation examples, t test examples.

DATASET	d	l	u	v	t
USPS2	241	93	1000	32	375
COIL6	241	90	1008	30	372
PC-MAC	7511	37	1410	13	486
ESET2	617	33	1305	12	1350

Experimental Protocol

Datasets were split into subsets of labeled, unlabeled, validation and test examples. The sizes of these subsets are recorded in Table 3.5. Results presented in Tables 3.6, 3.7 are averaged over 10 random splits. For each method compared, we set $\lambda' = 1$ and recorded results on each split over the parameter grid defined by

$$\lambda = 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}$$

and widths for RBF kernels in the range

$$\sigma = 1/8, 1/4, 1/2, 1, 2, 4, 8$$

These σ values are relative to a default value based on pairwise distances between examples in the dataset.

To focus our study on quality of optimization and degree of sensitivity to local minima, we chose to construct stratified splits so that each algorithm compared was provided an accurate estimate of class ratios. Parameters were chosen with respect to performance on the validation set for each split. Since model selection in semi-supervised settings with very few labels can often be unreliable and is largely considered to be an open issue, in Tables 3.6, 3.7 we also record the minimum of the mean error rate achieved over the parameter grid. This experimental setup neu-

Table 3.6: Comparison between SVM, JTSVM,∇TSVM and DA (all with quadratic hinge loss (l_2)). For each method, the top row shows mean error rates with model selection; the bottom row shows best mean error rates. u/t denotes error rates on unlabeled and test sets. Also recorded in performance of DA with squared loss (*sqr*).

	USPS2	COIL6	PC-MAC	ESET2
	u/t	u/t	u/t	u/t
SVM	8.0/8.2 7.5/7.8	22.9/23.5 21.5/21.9	21.1/20.0 18.9/17.9	20.9/21.8 19.4/19.7
JTSVM	8.8/8.0 7.6/7.2	21.4/22.8 19.9/21.2	14.1/11.9 10.4/7.0	**10.4/10.5** 9.2/8.9
∇TSVM	7.5/7.7 6.9/7.1	25.0/24.9 21.4/21.6	**7.6/6.9** 5.4/**4.5**	12.2/12.7 8.7/9.1
DA (l_2)	**6.5/6.6** **6.4/6.3**	**15.6/16.4** **13.6/15.0**	11.8/9.4 **5.3**/4.8	10.8/10.7 **8.1/8.5**
DA (*sqr*)	7.3/7.1 5.7/6.3	16.5/16.7 13.8/15.2	11.8/9.4 5.4/4.7	11.6/11.3 9.0/9.4

tralizes any undue advantage a method might receive due to different sensitivities to parameters, class imbalance issues and shortcomings of the model selection protocol.

Comparing DA, JTSVM and ∇TSVM

Table 3.6 presents a comparison between DA, JTSVM and ∇TSVM. The baseline results for SVM using only labeled examples are also provided. Being a gradient descent technique, ∇TSVM requires loss functions to be differentiable; the implementation in [30] uses the l_2 loss function over labeled examples and an exponential loss function over unlabeled examples. The results in Table 3.6 for DA and JTSVM were also obtained using the l_2 loss. Thus, these methods attempt to minimize very similar objective functions over the same range of parameters.

We see that DA is the best performing method on USPS2VS5 and COIL6. The performance improvement with DA is particularly striking on the COIL6 dataset where the TSVM and ∇TSVM performance falls below the SVM baseline. Since this dataset

consists of images of 100 objects randomly grouped into 6 classes, each class is expected to be composed of several clusters. Gaps in the data space for points within the same class probably result in many local minima. The same observations do not hold to the same extent for the ESET2 dataset where the two classes are composed of acoustic sub-clusters. Here, all methods seem to perform similarly though DA returns the best mean performance over the parameter grid. On the PC-MAC text dataset, DA and ∇TSVM out-perform JTSVM. The difference between DA and ∇TSVM is found to be minor in terms of best performance achieved on this dataset. We also observe that these methods also yield good out-of-sample extension on the test set.

Table 3.7: Importance of Annealing: DA versus fixed T (no annealing) optimization. For each method, the top row shows mean error rates with model selection; the bottom row shows best mean error rates. u/t denotes error rates on unlabeled and test sets.

	USPS2 u/t	COIL6 u/t	PC-MAC u/t	ESET2 u/t
DA-	**6.5**/**6.6** **6.4**/**6.3**	**15.6**/**16.4** **13.6**/**15.0**	11.8/9.4 **5.3**/4.8	10.8/10.7 8.1/8.5
T=0.1	8.5/8.4 6.6/6.8	23.9/23.3 20.0/21.0	12.6/9.9 5.7/**4.7**	**8.1**/**8.2** **7.8**/**8.0**
T=0.01	8.9/8.2 7.6/7.0	22.2/23.3 20.1/21.3	17.1/12.7 7.1/5.7	12.9/12.0 8.1/8.5
T=0.001	9.0/8.1 7.9/7.2	23.6/24.4 20.3/21.5	18.6/13.0 9.1/7.3	13.5/12.5 8.8/8.8

Importance of Annealing

In Table 3.7, we show the results obtained with DA for three fixed values of T with no annealing. We see that in most cases, fixed T optimization performs worse than optimization with annealing (gradually decreasing T from high to low values). On COIL, the performance drop is very significant implying that annealing may be critical for hard problems. Cases where fixed T optimization out-performed optimization

with annealing are shown in bold. However, even in these cases, annealing actually achieved a lower value of the objective function but this did not correspond to lower error rates.

Performance with Squared Loss

In Table 3.6 we see that results obtained with the squared loss are also highly competitive with other methods on real world semi-supervised tasks. This is not surprising given the success of the regularized least squares algorithm for classification problems.

3.7 Discussion

Our results establish DA as a highly competitive practical S^3VM algorithm. We believe that as annealing proceeds and the objective function deforms, the interplay between the geometric structure of the data and the inner and outer intervals of the effective loss function is a key issue for global optimization in semi-supervised kernel machines based on Eqn. 3.1. In the early stages of the optimization, an ideal effective loss function should make the decision boundary sufficiently sensitive to unlabeled examples that are "far-away" so that it begins to align with the global geometry of data clusters. When only local adjustments need to be done, the remaining optimization can succeed under weaker deformations. An interesting direction for future work is the design of general homotopies using functions other than the entropy in Eqn. 3.7. Another intriguing possibility is to combine DA with the semi-supervised kernel of Eqn. 2.22 thus implementing manifold and cluster assumptions within the same algorithm.

CHAPTER 4

CO-REGULARIZATION: A MULTI-VIEW APPROACH

The Co-Training algorithm uses unlabeled examples in multiple views to bootstrap classifiers in each view, typically in a greedy manner, and operating under assumptions of view-independence and compatibility. In this chapter, we propose a Co-Regularization framework where classifiers are learnt in each view through forms of multi-view regularization. We propose algorithms within this framework that are based on optimizing measures of agreement and smoothness over labeled and unlabeled examples. These algorithms naturally extend standard regularization methods like Support Vector Machines (SVM) and Regularized Least squares (RLS) for multi-view semi-supervised learning, and inherit their benefits and applicability to high-dimensional classification problems. An empirical investigation is presented that confirms the promise of this approach.

4.1 Introduction

A striking aspect of natural learning is the ability to integrate and process multi-modal sensory information with very little supervisory feedback. The scarcity of labeled examples, abundance of unlabeled data and presence of multiple representations are aspects of several applications of machine learning as well. An example is hypertext classification: Modern search engines can index more than a billion web-pages in a single web-crawl, but only a few can be hand-labeled and assembled into web directories. Each web-page has disparate descriptions: textual content, inbound and outbound hyperlinks, site and directory names, etc. Although traditional machine learning has focussed on two extremes of an information spectrum (supervised and unsupervised learning), a number of recent efforts have considered the middle-ground

of semi-supervised learning, with or without a multi-view component [2, 7, 86, 57, 59, 18, 22, 30, 108].

The Co-Training framework proposed in [18] has been among the first efforts that provided a widely successful algorithm with theoretical justifications. The framework employs two assumptions that allow unlabeled examples in multiple-views to be utilized effectively: (a) the assumption that the target functions in each view agree on labels of most examples (*compatibility* assumption) and (b) the assumption that the views are independent given the class label (*independence* assumption). The first assumption allows the complexity of the learning problem to be reduced by the constraint of searching over compatible functions; and the second assumption allows high performance to be achieved since it becomes unlikely for compatible classifiers trained on independent views to agree on an incorrect label. The co-training idea has become synonymous with a greedy agreement-maximization algorithm that is initialized by supervised classifiers in each view and then iteratively re-trained on boosted labeled sets, based on high-confidence predictions on the unlabeled examples. The original implementation in [18] runs this algorithm on naive-bayes classifiers defined in each view. For more on agreement maximization principles, see [39, 35, 106].

In this chapter, we present a *Co-Regularization* framework for multi-view semi-supervised learning. Our approach is based on implementing forms of multi-view regularization using unlabeled examples. We suggest a family of algorithms within this framework: The Co-Regularized Least Squares (Co-RLS) algorithm performs a joint regularization that attempts to minimize disagreement in a least squared sense; the Co-Regularized Laplacian SVM and Least Squares (Co-LapSVM, Co-LapRLS) algorithms utilize multi-view graph regularizers to enforce complementary and robust notions of smoothness in each view. Manifold Regularization techniques [7, 81, 86] presented in Chapter 2 are employed for Co-LapSVM and Co-LapRLS. Learning is

performed by effectively exploiting useful structures collectively revealed with multiple representations.

We highlight features of the proposed algorithms:

1. These algorithms arise from natural extensions of the classical framework of regularization in Reproducing Kernel Hilbert Spaces. The unlabeled data is incorporated via additional regularizers that are motivated from recognized principles of semi-supervised learning.

2. The algorithms are non-greedy, involve convex cost functions and can be easily implemented.

3. We derive data-dependent kernels that provide a multi-view RKHS where the usual norm corresponds to consensus maximization.

4. The influence of unlabeled data and multiple views can be controlled explicitly. In particular, single view semi-supervised learning and standard supervised algorithms are special cases of this framework.

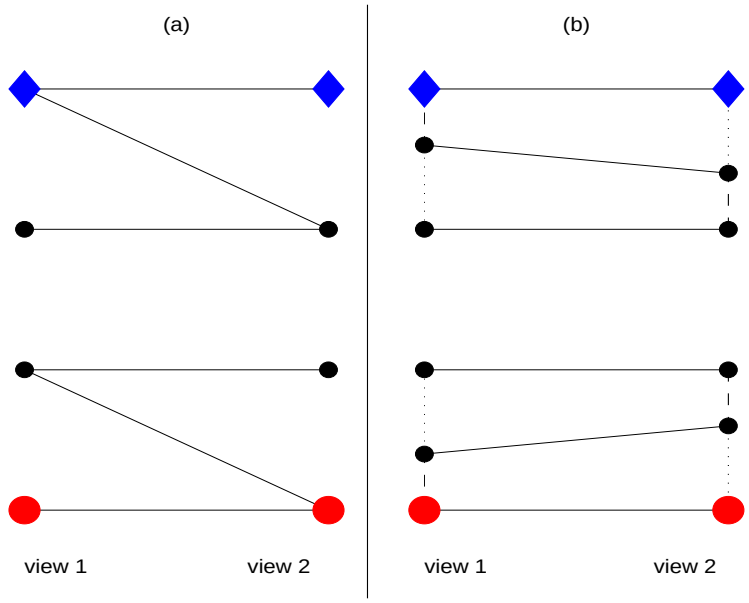
5. Experimental results demonstrate that the proposed methods out-perform standard co-training on synthetic and hypertext classification datasets.

We next setup the problem of semi-supervised learning in multiple views. In subsequent sections, we discuss the Co-Regularization framework, derive underlying data-dependent kernels, and evaluate the empirical performance of our approach.

4.2 Multi-View Learning

In the multi-view semi-supervised learning setting, we have labeled examples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{l}$ and unlabeled examples $\{\boldsymbol{x}_i\}_{l+1}^{l+u}$ where each example $x = (\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)})$ is seen in two

Figure 4.1: Bipartite Graph Representation of multi-view learning. The small black circles are unlabeled examples.



views with $\boldsymbol{x}^{(1)} \in \mathcal{X}^{(1)}$ and $\boldsymbol{x}^{(2)} \in \mathcal{X}^{(2)}$. The setup and the algorithms we discuss can also be generalized to more than two views. For the rest of this discussion, we consider binary classification problems where $y_i \in \{-1, 1\}$. The goal is to learn the function pair $f = (f^{(1)}, f^{(2)})$, where $f^{(1)} : \mathcal{X}^{(1)} \mapsto \Re$ and $f^{(2)} : \mathcal{X}^{(2)} \mapsto \Re$ are classifiers in the two views. In this paper, we will focus on how the availability of unlabeled examples and multiple views may be profitably leveraged for learning high-performance classifiers $f^{(1)}, f^{(2)}$ in each view.

How can unlabeled data and its multiple views help? In Figure 4.1(a), we reproduce the bipartite graph representation of the co-training setting, to initiate a discussion. The figure shows the two views of labeled and unlabeled examples, arranged as a bipartite graph. The left and right nodes in the graph are examples as seen in view 1 and view 2 respectively, with edges connecting the two views of an example. The unlabeled examples are shown as small black circles and the other

examples are labeled. The class of compatible pairs of functions identically label two nodes in the same connected component of this graph. This may be interpreted as a requirement of smoothness over the graph for the pair $(f^{(1)}, f^{(2)})$. Thus, unlabeled examples provide empirical estimates of regularizers or measures of smoothness to enforce the right complexity for the pair $(f^{(1)}, f^{(2)})$.

In many applications, it is unrealistic for two examples to share a view exactly. A more realistic situation is depicted in Figure 4.1(b) where three types of edges are shown: the (solid) edges connecting views of each example as in Figure 4.1(a); the (dashed) edges connecting similar examples in each view; and the (dotted) edges connecting examples in each view based on similarity in the other view. The similarity structure in one view induces a complementary notion of similarity in the other views with respect to which regularizers can be constructed using unlabeled data.

In the next section, we describe algorithms that arise from constructions of such regularizers.

4.3 Co-Regularization

The classical regularization framework [74, 77, 97] for supervised learning solves the following minimization problem :

$$f^* = \underset{f \in \mathcal{H}_K}{\operatorname{argmin}} \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + \gamma \|f\|_K^2 \tag{4.1}$$

where $\mathcal{H}_K$ is an Reproducing Kernel Hilbert space (RKHS) of functions with kernel function K; $\{(x_i, y_i)\}_{i=1}^{l}$, is the labeled training set; and V is some loss function, such as squared loss for Regularized Least Squares (RLS) or the hinge loss function for Support Vector Machines (SVM). By the Representer theorem, the minimizer is

a linear combination of kernel functions centered on the data:

$$f(x) = \sum_{i=1}^{l} \alpha_i K(x, x_i)$$

This real-valued function is thresholded and used for binary classification.

In the Co-regularization framework, we attempt to learn the pair $f = (f^{(1)}, f^{(2)})$ in a cross-product of two RKHS defined over the two views, i.e., $f^{(1)} \in \mathcal{H}_{K^{(1)}}$ and $f^{(2)} \in \mathcal{H}_{K^{(2)}}$. The key issue is imposing an appropriate notion of complexity on this pair so that a regularized solution effectively utilizes unlabeled data in the two views. We now describe some ideas.

Co-Regularized Least Squares

A natural idea is to attempt to learn the pair $f = (f^{(1)}, f^{(2)})$ so that each function correctly classifies the labeled examples, and the outputs of the pair agree over unlabeled examples. This suggests the following objective function:

$$\begin{aligned}(f^{(1)*}, f^{(2)*}) = \underset{\substack{f^{(1)} \in \mathcal{H}_{K^{(1)}} \\ f^{(2)} \in \mathcal{H}_{K^{(2)}}}}{\operatorname{argmin}} & \sum_{i=1}^{l} \left[y_i - f^{(1)}(x_i^{(1)}) \right]^2 + \\ & \mu \sum_{i=1}^{l} \left[y_i - f^{(2)}(x_i^{(2)}) \right]^2 + \gamma_1 \|f^{(1)}\|^2_{\mathcal{H}_{K^{(1)}}} + \\ \gamma_2 \|f^{(2)}\|^2_{\mathcal{H}_{K^{(2)}}} + & \frac{\gamma_C}{(l+u)} \sum_{i=1}^{l+u} \left[f^{(1)}(x_i^{(1)}) - f^{(2)}(x_i^{(2)}) \right]^2\end{aligned}$$

Here, μ is a real-valued parameter to balance data fitting in the two views, γ_1, γ_2 are regularization parameters for the RKHS norms in the two views, and γ_C is the coupling parameter that regularizes the pair towards compatibility using unlabeled

data. It is easy to see that a representer theorem holds that expresses the minimizing pair $\left(f^{(1)*}(x^{(1)}), f^{(2)*}(x^{(2)}\right)$ in the following form:

$$\left(\sum_{i=1}^{l+u} \alpha_i K^{(1)}(x^{(1)}, x_i^{(1)}) \quad , \quad \sum_{i=1}^{l+u} \beta_i K^{(2)}(x^{(2)}, x_i^{(2)})\right)$$

The $(l+u)$ dimensional expansion coefficient vectors α, β may be computed by solving the following coupled linear system:

$$\begin{aligned}\left[\frac{1}{l} J K_1 + \gamma_1 I + \frac{\gamma_C}{l+u} K_1\right] \alpha - \frac{\gamma_C}{l+u} K_2 \beta &= \frac{1}{l} Y \\ \left[\frac{\mu}{l} J K_2 + \gamma_2 I + \frac{\gamma_C}{l+u} K_2\right] \beta - \frac{\gamma_C}{l+u} K_1 \alpha &= \frac{\mu}{l} Y\end{aligned}$$

where Y is a label vector given by $Y_i = y_i$ for $1 \leq i \leq l$ and $Y_i = 0$ for $l+1 \leq i \leq l+u$; J is a diagonal matrix given by $J_{ii} = |Y_i|$, and K_1, K_2 are gram matrices of the kernel functions $K^{(1)}, K^{(2)}$ over labeled and unlabeled examples.

When $\gamma_C = 0$, the system ignores unlabeled data and yields an uncoupled pair of solutions corresponding to supervised RLS. We also note a relationship over coefficients corresponding to unlabeled examples: $\gamma_1 \alpha_i = -\gamma_2 \beta_i$ for $l+1 \leq i \leq l+u$. The algorithm appears to work well in practice when orthogonality to the constant function is enforced over the data to avoid all unlabeled examples from being identically classified.

Working with the hinge loss, one can also extend SVMs in a similar manner.

Co-Laplacian RLS and Co-Laplacian SVM

The intuitions from the discussion concerning Figure 4.1(b) is to learn the pair $f = (f^{(1)}, f^{(2)})$ so that each function correctly classifies the labeled examples and

is smooth with respect to similarity structures in both views. These structures may be encoded as graphs on which regularization operators may be defined and then combined to form a multi-view regularizer. The function pair is indirectly coupled through this regularizer.

We assume that for each view (indexed by $s = 1, 2$), we can construct a similarity graph whose adjacency matrix is $W^{(s)}$, where $W^{(s)}_{ij}$ measures similarity between $x^{(s)}_i$ and $x^{(s)}_j$. The Laplacian matrix of this graph is defined as $L^{(s)} = D^{(s)} - W^{(s)}$ where $D^{(s)}$ is the diagonal degree matrix $D^{(s)}_{ii} = \sum_j W^{(s)}_{ij}$. The graph Laplacian is a positive semi-definite operator on functions defined over vertices of the graph. It provides the following smoothness functional on the graph:

$$\mathbf{g}^T L^{(s)} \mathbf{g} = \sum_{ij} (g_i - g_j)^2 W^{(s)}_{ij}$$

where $\mathbf{g}$ is a vector identifying a function on the graph whose value is g_i on node i. Other regularization operators can also be defined using the graph Laplacian [65, 89, 2].

One way to construct a multi-view regularizer is to simply take a convex combination $L = (1 - \alpha)L^{(1)} + \alpha L^{(2)}$ where $\alpha \geq 0$ is a non-negative parameter which controls the influence of the two views. To learn the pair $f = (f^{(1)*}, f^{(2)*})$, we solve the following optimization problems for $s = 1, 2$ using squared loss or hinge loss:

$$\begin{aligned} f^{(s)*} = \underset{f^{(s)} \in \mathcal{H}_{K^{(s)}}}{\operatorname{argmin}} & \frac{1}{l} \sum_{i=1}^{l} V(x^{(s)}_i, y_i, f^{(s)}) + \\ & \gamma^{(s)}_A \|f^{(s)}\|^2_{K^{(s)}} \quad + \quad \gamma^{(s)}_I \mathbf{f}^{(s)T} L \mathbf{f}^{(s)} \end{aligned}$$

where $\mathbf{f}^{(s)}$ denotes the vector $\left(f^{(s)}(x^{(s)}_1), \ldots, f^{(s)}(x^{(s)}_{l+u})\right)^T$; and the regularization

parameters $\gamma_A^{(s)}, \gamma_I^{(s)}$ control the influence of unlabeled examples relative to the RKHS norm.

The solutions to these optimization problems produce Laplacian SVM (for hinge loss) or Laplacian RLS (for squared loss) classifiers trained with the multi-view graph regularizer (see Chapter 2). The resulting algorithms are termed Co-Laplacian SVM and Co-Laplacian RLS respectively.

The solutions are obtained by training a standard SVM or RLS using the following modified kernel function given in Eqn. 2.22:

$$\tilde{K}^{(s)}(x^{(s)}, z^{(s)}) = K^{(s)}(x^{(s)}, z^{(s)}) \quad - \\ \mathbf{k}^{\mathbf{T}}_{\mathbf{x^{(s)}}}(I + MG^{(s)})^{-1} M \mathbf{k}_{\mathbf{z^{(s)}}}$$

where $G^{(s)}$ is the gram matrix of the kernel function $K^{(s)}$; $\mathbf{k}_{\mathbf{x^{(s)}}}$ denotes the vector $\left(K^{(s)}(x_1^{(s)}, x^{(s)}), \ldots, K^{(s)}(x_n^{(s)}, x^{(s)})\right)^T$ and $M = \frac{\gamma_I^{(s)}}{\gamma_A^{(s)}} L$. See Chapter 2 for a derivation.

When $\alpha = 0$ for view 1 or $\alpha = 1$ for view 2, the multi-view aspect is ignored and the pair consists of Laplacian SVM or Laplacian RLS in each view. When $\gamma_I = 0$, the unlabeled data is ignored and the pair consists of standard SVM or RLS classifiers.

The idea of combining graph regularizers and its connection to co-training has been briefly discussed in [59] in the context of applying spectral graph transduction (SGT) in multi-view settings. However, unlike co-training, SGT does not produce classifiers defined everywhere in $X^{(1)}, X^{(2)}$ so that predictions cannot be made on novel test points. By optimizing in reproducing kernel Hilbert spaces defined everywhere, Co-Laplacian SVM and RLS can also extend beyond the unlabeled examples.

4.4 Kernels for Co-Regularization

Let $\{\boldsymbol{x}_i = (x_i^1, x_i^2)\}_{i=1}^n$ with $\boldsymbol{x} \in \mathcal{X} = \mathcal{X}^1 \times \mathcal{X}^2$ be examples appearing in two "view" spaces. Let $K^1 : \mathcal{X}^1 \times \mathcal{X}^1 \mapsto \mathcal{R}$ and $K^2 : \mathcal{X}^2 \times \mathcal{X}^2 \mapsto \mathcal{R}$ be reproducing kernels of spaces $\mathcal{H}^1 : \mathcal{X}^1 \mapsto \mathcal{R}$ and $\mathcal{H}^2 : \mathcal{X}^2 \mapsto \mathcal{R}$ whose norms we will denote as $\|\cdot\|_{\mathcal{H}^1}$ and $\|\cdot\|_{\mathcal{H}^2}$. We will consider the extension of these function spaces to $\mathcal{X}$ (overloaded with the same notation) so that $\mathcal{H}^i : \mathcal{X} \mapsto \mathcal{R}$ defined by $K^i(\boldsymbol{x}, \boldsymbol{z}) = K^i(x^i, z^i)$ and $f^i(\boldsymbol{x}) = f^i(x^i) \in \mathcal{H}^i$ where the superscript i will always run over $1, 2$ and the two overloaded notations might be used interchangeably.

Let us construct the space

$$\mathcal{H} = \mathcal{H}^1 \oplus \mathcal{H}^2 = \{f | f(\boldsymbol{x}) = f^1(\boldsymbol{x}) + f^2(\boldsymbol{x}), f^1 \in \mathcal{H}^1, f^2 \in \mathcal{H}_2\}$$

and impose on it a data-dependent co-regularization norm:

$$\|f\|_{\mathcal{H}}^2 = \min_{\substack{f=f^1+f^2 \\ f^1 \in \mathcal{H}^1, f^2 \in \mathcal{H}^2}} \left(\|f^1\|_{\mathcal{H}^1}^2 + \nu \|f^2\|_{\mathcal{H}^2}^2 + \mu \sum_{i=1}^n \left[f^1(\boldsymbol{x}) - f^2(\boldsymbol{x}) \right]^2 \right) \tag{4.2}$$

where μ, ν are real-valued constants.

Proposition: *The Reproducing Kernel $\mathcal{K}$ for $\mathcal{H}$ is given by:*

$$\begin{aligned} \mathcal{K}(\boldsymbol{x}, \boldsymbol{z}) = & \left[K^1(\boldsymbol{x}, \boldsymbol{z}) + \frac{1}{\nu} K^2(\boldsymbol{x}, \boldsymbol{z}) \right] - \\ & \mu \left(\mathbf{k}_{\boldsymbol{x}}^1 - \frac{1}{\nu} \mathbf{k}_{\boldsymbol{x}}^2 \right)^T \left[I + \mu \left(G^1 + \frac{1}{\nu} G^2 \right) \right]^{-1} \left(\mathbf{k}_{\boldsymbol{z}}^1 - \frac{1}{\nu} \mathbf{k}_{\boldsymbol{z}}^2 \right) \end{aligned} \tag{4.3}$$

where for $i = 1, 2$, $\mathbf{k}_{\boldsymbol{x}}^i = \left[K^i(\boldsymbol{x}, \boldsymbol{x}_1) \ldots K^i(\boldsymbol{x}, \boldsymbol{x}_n) \right]$ and G^i is the associated gram matrix of K^i over the data.

Proof: (Generalizing Theorem 5 in [12]) Let $\mathcal{F} = \mathcal{H}^1 \otimes \mathcal{H}^2 = \{(f^1, f^2) : f^1 \in$

$\mathcal{H}^1, f^2 \in \mathcal{H}^2\}$ with an inner product defined on it as follows:

$$\langle (f^1, f^2), (g^1, g^2) \rangle_{\mathcal{F}} = \langle f^1, g^1 \rangle_{\mathcal{H}^1} + \nu \langle f^2, g^2 \rangle_{\mathcal{H}^2} + \mu (\Delta[f^1, f^2])^T (\Delta[g^1, g^2]) \quad (4.4)$$

where $\Delta[f^1, f^2] = \left[\left(f^1(\boldsymbol{x}_1) - f^2(\boldsymbol{x}_1) \right) \ldots \left(f^1(\boldsymbol{x}_n) - f^2(\boldsymbol{x}_n) \right) \right]^T$.

Define the map,

$$u : \mathcal{F} \mapsto \mathcal{H} \qquad u(f_1, f_2) = f_1 + f_2$$

Let $N = u^{-1}(\{0\})$. The map u is linear and onto. Its kernel N is a subspace of $\mathcal{F}$. Let $((f_n, -f_n))$ be a sequence of elements of N converging to (f^1, f^2). Then (f_n) converges to f^1 in $\mathcal{H}^1$ and $(-f_n)$ converges to f^2 in $\mathcal{H}^2$. Thus, $f_1 = -f_2$ and N is a closed subspace of $\mathcal{F}$.

Let $N^{\perp}$ be the orthogonal complement of N in $\mathcal{F}$ and let v be the restriction of u to $N^{\perp}$. The map v is one-to-one and hence one can define an inner product on $\mathcal{H}$ by setting,

$$\langle f, g \rangle_{\mathcal{H}} = \langle v^{-1}(f), v^{-1}(g) \rangle_{\mathcal{F}} \quad (4.5)$$

With this inner product $\mathcal{H}$ is a Hilbert space of functions. We will show that $\mathcal{K}$ is the Reproducing Kernel of $\mathcal{H}$.

$\mathcal{K}_{\boldsymbol{z}} = \mathcal{K}(\boldsymbol{z}, \cdot) \in \mathcal{H}$ since

$$\mathcal{K}_{\boldsymbol{z}} = h^1 + h^2$$

where

$$h^1 = K^1_{\boldsymbol{z}} - \mu \sum_i \beta_{\boldsymbol{z}i} K^1_{\boldsymbol{x}_i} \in \mathcal{H}^1 \tag{4.6}$$

$$h^2 = \frac{1}{\nu} K^2_{\boldsymbol{z}} + \frac{\mu}{\nu} \sum_i \beta_{\boldsymbol{z}i} K^2_{\boldsymbol{x}_i} \in \mathcal{H}^2 \tag{4.7}$$

$$\text{where } \beta_{\boldsymbol{z}} = \left[I + \mu \left(G^1 + \frac{1}{\nu} G^2 \right) \right]^{-1} \left(\mathbf{k}^1_{\boldsymbol{z}} - \frac{1}{\nu} \mathbf{k}^2_{\boldsymbol{z}} \right) \tag{4.8}$$

It remains to check the reproducing property of $\mathcal{K}$.

Let $f \in \mathcal{H}$, $(f', f'') = v^{-1}(f)$ and $(\mathcal{K}'_{\boldsymbol{z}}, \mathcal{K}''_{\boldsymbol{z}}) = v^{-1}(\mathcal{K}_{\boldsymbol{z}})$. So we have the following (the second by above),

$$\mathcal{K}_{\boldsymbol{z}} = \mathcal{K}'_{\boldsymbol{z}} + \mathcal{K}''_{\boldsymbol{z}} \tag{4.9}$$

$$\mathcal{K}_{\boldsymbol{z}} = h^1 + h^2 \tag{4.10}$$

Subtracting the two equations, this implies that,

$$\left(\mathcal{K}'_{\boldsymbol{z}} - h^1 \right) + \left(\mathcal{K}''_{\boldsymbol{z}} - h^2 \right) = 0 \tag{4.11}$$

In other words,

$$\left(\mathcal{K}'_{\boldsymbol{z}} - h^1, \mathcal{K}''_{\boldsymbol{z}} - h^2 \right) \in N$$

. Since $(f', f'') \in \mathcal{N}^{\perp}$, their inner product in $\mathcal{F}$ is 0,

$$\langle (f', f''), \left(\mathcal{K}'_{\boldsymbol{z}} - h^1, \mathcal{K}''_{\boldsymbol{z}} - h^2 \right) \rangle_{\mathcal{F}} = 0 \tag{4.12}$$

To check the Reproducing property, observe that

$$\begin{aligned}
\langle f, \mathcal{K}_{\boldsymbol{z}} \rangle_{\mathcal{H}} &= \langle v^{-1}(f), v^{-1}(\mathcal{K}_{\boldsymbol{z}}) \rangle_{\mathcal{F}} \\
&= \langle (f', f''), (\mathcal{K}'_{\boldsymbol{z}}, \mathcal{K}''_{(}\boldsymbol{z})) \rangle_{\mathcal{F}} \\
&= \langle (f', f''), (h^1, h^2) \rangle_{\mathcal{F}} \\
&= \langle f', h^1 \rangle_{\mathcal{H}^1} + \nu \langle f'', h^2 \rangle_{\mathcal{H}^2} + \mu (\Delta[f', f''])^T (\Delta[h^1, h^2]) \\
&= \left(f'(\boldsymbol{z}) - \mu \sum_i \beta_{\boldsymbol{z}i} f'(\boldsymbol{x}_i) \right) + \nu \left(\frac{1}{\nu} f''(\boldsymbol{z}) + \frac{\mu}{\nu} \sum_i \beta_{\boldsymbol{z}i} f''(\boldsymbol{x}_i) \right) + \\
&\quad \mu \sum_i \left(f'(\boldsymbol{x}_i) - f''(\boldsymbol{x}_i) \right) \left(h^1(\boldsymbol{x}_i) - h^2(\boldsymbol{x}_i) \right) \\
&= f'(\boldsymbol{z}) + f''(\boldsymbol{z}) \\
&= f(\boldsymbol{z}) \qquad (4.13)
\end{aligned}$$

where the second last step follows because,

$$h^1(\boldsymbol{x}^i) - h^2(\boldsymbol{x}^i) = \beta_{\boldsymbol{z}i}$$

Let $(f^1, f^2) \in \mathcal{F}, f = f^1 + f^2, (g^1, g^2) = (f^1, f^2) - v'(f)$. By the definition of the norm in $\mathcal{F}$,

$$\|(f^1, f^2)\|^2_{\mathcal{F}} = \|f^1\|^2_{\mathcal{H}^1} + \nu \|f^2\|^2_{\mathcal{H}^2} + \mu \sum_i \left[f^1(\boldsymbol{x}_i) - f^2(\boldsymbol{x}_i) \right]^2$$

But $(g^1, g^2) \in N$ and $v^{-1}(f) \in N^{\perp}$. So,

$$\begin{aligned}
\|(f^1, f^2)\|^2_{\mathcal{F}} = \|v^{-1}(f)\|^2_{\mathcal{F}} + \|(g^1, g^2)\|^2_{\mathcal{F}} = \|v^{-1}(f)\|^2_{\mathcal{F}} + \|g^1\|^2_{\mathcal{H}^1} + \\
\nu \|g^2\|^2_{\mathcal{H}^2} + \mu \sum_i \left[g^1(\boldsymbol{x}_i) - g^2(\boldsymbol{x}_i) \right]^2 \qquad (4.14)
\end{aligned}$$

So for the decomposition $f = f^1 + f^2$, we always have,

$$\|f\|^2_{\mathcal{H}} = \|v^{-1}(f)\|^2_{\mathcal{F}} \leq \|f^1\|^2_{\mathcal{H}^1} + \nu\|f^2\|^2_{\mathcal{H}^2} + \mu \sum_i \left[f^1(\boldsymbol{x}_i) - f^2(\boldsymbol{x}_i)\right]^2 \quad (4.15)$$

where the equality holds if and only if $(f^1, f^2) = v^{-1}(f)$. □

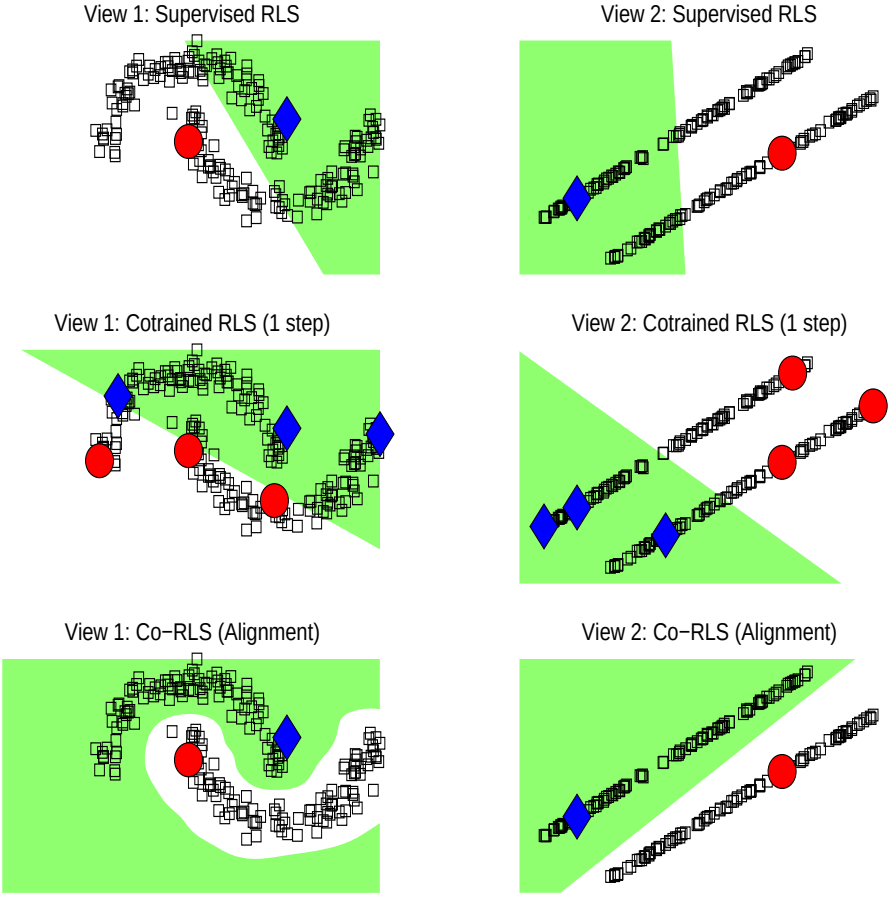
4.5 Experiments

Two-Moons-Two-Lines Toy Example

Figure 4.2 and Figure 4.3 demonstrate Co-Regularization ideas on a toy dataset in which objects in two classes appear as two moons in one view and two oriented lines in another. Class conditional view independence is enforced by randomly associating points on one moon with points on one line. One example is labeled from each class and shown as the large colored diamond and circle; the other examples are unlabeled. We chose a Gaussian kernel for the two moons view and a linear kernel for the two lines view.

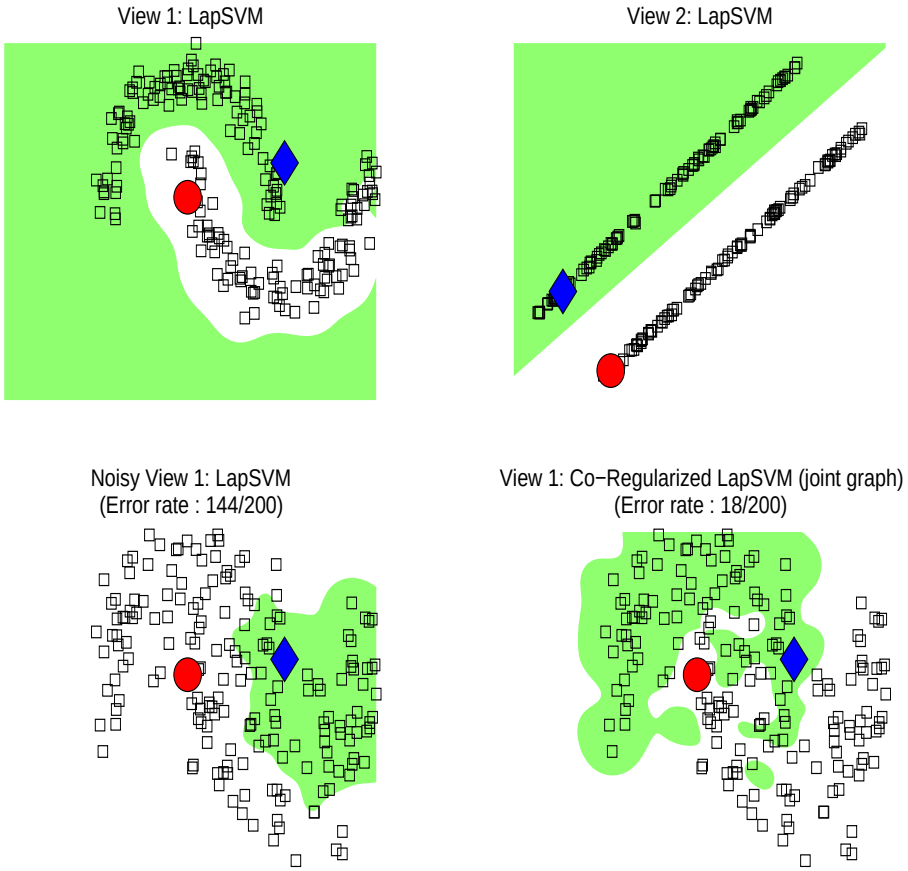
In the top panel of Figure 4.2, we see that a supervised Regularized least squares classifier is unable to produce reasonable classifiers with only 2 labeled examples. In the middle panel, we add two more labeled examples based on the most confident predictions (which are actually incorrect) of the supervised classifiers on the unlabeled data. The middle panel shows the classifiers obtained after 1 iteration of standard co-training with the boosted set of 4 labeled examples. Since greedy co-training does not revise conjectured labels, subsequent training fails to yield good classifiers in either view. By contrast, Co-Regularized Least squares classifiers, shown in panel 3, effectively use the unlabeled data in two views.

Figure 4.2: Two-Moons-Two-Lines : RLS, Co-trained RLS and Co-RLS



In the top panel of Figure 4.3, we show single-view semi-supervised learning with Laplacian SVMs in the two views. We then add noise to the two-moons view so that the two clusters are merged. This is shown in the bottom left panel. In this case, the unlabeled data fails to provide any structure for Laplacian SVM to exploit. However, when the joint graph laplacian is used, the rich structure in the two-lines view can be used to recover good decision boundaries in the two moons view. The bottom right panel shows the boundaries constructed by Co-Laplacian SVM.

Figure 4.3: Two-Moons-Two-Lines : Laplacian SVM and Co-Laplacian SVM



Hypertext Categorization

We considered the WebKB hypertext categorization task studied in [18, 59, 70]. There are 1051 web documents belonging to two classes: *course* or *non-course* from four universities. Only 12 examples are labeled. The two views are the textual content of a webpage (which we will call *page* representation) and the anchortext on links on other webpages pointing to the webpage (*link* representation).

The data was preprocessed into 3000 features for the page-view and 1840 features for the link view using the Rainbow software [69]. We used linear kernels for both views. We also considered a page+link representation with concatenated features.

Table 4.1: Mean precision-recall breakeven points over unlabeled documents for a hypertext classification task.

View → Classifier ↓	link	page	page+ link
RLS (full)	94.4	94.0	97.8
SVM (full)	93.7	93.5	99.0
RLS (12)	72.0	71.6	78.3
SVM (12)	74.4	77.8	84.4
SGT	78.0	89.3	93.4
TSVM	85.5	91.4	92.2
LapRLS	80.8	89.0	93.1
LapSVM	81.9	89.5	93.6
Co-trained RLS	74.8	80.2	-
Co-RLS	80.8	90.1	-
Co-LapRLS1	93.1	90.8	90.4
Co-LapRLS2	94.4	92.0	93.6
Co-trained SVM	88.3	88.7	-
Co-LapSVM1	93.2	93.2	90.8
Co-LapSVM2	94.3	93.3	94.2

The performance of several methods as measured by mean precision-recall breakeven point (PRBEP) is tabulated in Table 4.1. These methods are (a) RLS, SVM on fully labeled data sets and with 12 randomly chosen labeled examples; (b) single-view semi-supervised methods: SGT [59], TSVMs [57], Laplacian SVM, Laplacian RLS [7, 86]; (c) multi-view semi-supervised methods: Co-RLS, Co-trained RLS, Co-trained SVM, Co-LapRLS and Co-LapSVM. In Table 4.1, Co-LapRLS1, Co-LapSVM1 use $\alpha = 0.5$ to combine graph Laplacians in page and link views; and Co-LapRLS2, Co-LapSVM2 use the mean graph Laplacian over page, link and page+link views, to bias classifiers in each view. The performance of supervised classifiers with full labels (RLS (full) and SVM (full)) is the mean PRBEP for 10-fold cross-validation. For all other methods, we average over random choices of 12 labeled examples (making sure that

each class is sampled at least once) and measure the mean PRBEP evaluated over the remaining 1039 examples. We avoided the model selection issue due to the small size of the labeled set and chose best parameters over a small range of values.

The results in table 4.1 suggest that Co-LapSVM and Co-LapRLS are able to effectively use unlabeled examples in the two views. The link and page classifiers using 12 labeled examples, 1039 unlabeled examples and multi-view regularizers match the performance of supervised classifiers with access to all the labels. We also see that Co-RLS outperforms Co-trained RLS. In Table 4.2, we report the performance of Co-Laplacian SVM (using the mean graph Laplacian over the page, link and page+link views) in classifying unlabeled and test web-documents of four universities. The high correlation between performance on unlabeled and unseen test examples suggests that the method provides good extension outside the training set.

Table 4.2: Mean precision-recall breakeven points over test documents and over unlabeled documents (test , unlabeled)

University → View ↓	page+link	page	link
Cornell	91.6 , 90.9	88.9 , 88.8	88.2 , 88.7
Texas	94.8 , 95.5	91.6 , 92.4	90.9 , 93.5
Washington	94.7 , 94.9	94.0 , 93.9	93.7 , 92.4
Wisconsin	92.0 , 91.4	87.6 , 86.6	86.1 , 84.5

4.6 Conclusion

We have proposed extensions of regularization algorithms in a setting where unlabeled examples are easily available in multiple views. The algorithms provide natural extensions for SVM and RLS in such settings. We plan to further investigate the properties of these algorithms and benchmark them on real world tasks.

CHAPTER 5

SPARSE NON-LINEAR MANIFOLD REGULARIZATION

Two of the most challenging themes in high-dimensional data analysis [66] are: (a) Learning with few basis functions (sparse learning), and (b) Learning with few labeled but many unlabeled examples (semi-supervised learning). It is natural to attempt to extend sparse methods to utilize unlabeled data with the goal of handling large-scale applications where labeling is expensive. Greedy matching pursuit and l_1 methods can be immediately combined with Manifold Regularization for sparse, semi-supervised learning. However, through a series of empirical observations, we find that intrinsic graph regularization on restricted function spaces may not be sufficiently effective for semi-supervised learning. We propose modifications for overcoming this problem, leading to significant performance improvements over the basic techniques.

In a typical semi-supervised classification setting, we are given a few labeled examples together with a large collection of unlabeled data from which to estimate an unknown decision surface. Fortunately, on many natural problems, unlabeled examples carry information about the classification boundary that semi-supervised learning algorithms can attempt to exploit with the aim of improving generalization performance. Not only are there labeling constraints in many applications, but also computational or interpretability requirements which make it particularly desirable to learn functions that are compact, in some sense. The twin goal of sparse, semi-supervised learning is to utilize large amounts of unlabeled data, and to do so in a highly compressible manner.

The Manifold Regularization framework for semi-supervised learning (see Chapter 2) is motivated by the assumption that even if data points appear in a high-dimensional ambient space, they truly reside on an underlying low-dimensional man-

ifold on which the decision function changes smoothly. Given l labeled examples $\{\boldsymbol{x}_i, y_i\}_{i=1}^{l}$ and u unlabeled examples $\{\boldsymbol{x}_j\}_{j=l+1}^{n=l+u}$, the following extended regularization problem is solved,

$$\underset{f\in\mathcal{H}_k}{\operatorname{argmin}}\,\mathcal{R}(f) = \frac{\gamma_A}{2}\|f\|_k^2 + \frac{\gamma_I}{2}\boldsymbol{f}^\top M\boldsymbol{f} + \sum_{i=1}^{l} V(y_i, f(\boldsymbol{x}_i)) \tag{5.1}$$

Here, $\mathcal{H}_k$ is an RKHS corresponding to a kernel function k defined over the ambient space, and V is a loss function over labeled examples. The RKHS norm $\|f\|_k$ provides a data-independent measure of smoothness. Together with the loss terms, this norm constitutes a standard objective function for supervised learning where γ_A is the ambient regularization parameter. The additional regularization term, $\frac{\gamma_I}{2}\boldsymbol{f}^\top M\boldsymbol{f}$, measures smoothness with respect to the intrinsic data geometry which is estimated using unlabeled examples. Here, $\boldsymbol{f}^\top = [f(\boldsymbol{x}_1)\ldots f(\boldsymbol{x}_n)]$, γ_I is the intrinsic regularization parameter and M is a graph regularization matrix which is typically derived from the Laplacian of a data-similarity graph over both labeled and unlabeled examples. Solving (1) over ambiently-defined functions resolves the problem of out-of-sample prediction on new test points that arises in graph transduction where $\frac{\gamma_I}{2}\boldsymbol{f}^\top M\boldsymbol{f} + \sum_{i=1}^{l} V(y_i, f_i)$ is minimized over $\boldsymbol{f} \in \mathcal{R}^n$ (all functions defined only over the finite collection of labeled and unlabeled examples). Another key feature of Manifold regularization is that depending on the degree to which the manifold assumption is satisfied, γ_A and γ_I may be traded off to smoothly span solutions from supervised learning (ignore unlabeled data) to graph transduction (trust intrinsic regularizer fully) with out-of-sample extension.

The solution to the minimization problem above, by the Representer theorem, is of the form, $f(\boldsymbol{x}) = \sum_{i=1}^{n} \beta_i K(\boldsymbol{x}, \boldsymbol{x}_i)$, and therefore only the finite set of β_i remain to be estimated. However, in typical semi-supervised situations where n is large, it

becomes expensive to evaluate f. In this chapter, we consider the problem of finding sparse solutions for Manifold Regularization. We begin in Section 5.1 by briefly outlining techniques based on Kernel Matching pursuit [100] and l_1 regularization methods (Lasso and LARS [46, 93]). In Section 5.2, we present comprehensive empirical evidence that minimizing (5.1) over restricted function spaces defined by a sparse set of basis functions does not lead to satisfactory performance. In section 5.3 we suggest modifications that lead to significant improvements over basic techniques. Section 5.4 concludes this chapter. A recent but different approach for sparsified manifold regularization is presented in [95].

5.1 Sparse Manifold Regularization

We restrict our attention to squared loss, $V(y, f(\boldsymbol{x})) = \frac{1}{2}(y - f(\boldsymbol{x}))^2$, since the details are particularly simple in this case. The problem (5.1) then becomes the objective function of the Laplacian Regularized Least Squares (LAPRLS) algorithm. Our observations and methods also hold for Laplacian SVMs which typically perform very similar to LAPRLS (see analysis of benchmarks in [26]).

5.1.1 Greedy Kernel Matching Pursuit

Given that Eqn. 5.1 is a quadratic minimization problem for the squared loss, the kernel matching pursuit ideas of [100] can be easily extended to Eqn. 5.1 to get sparse solutions. Instead of minimizing Eqn. 5.1 over the RKHS $\mathcal{H}_K$, we can restrict the solution to a linear subspace corresponding to a set of d points (it is usual to select these points as a subset of the training points). Define $\mathcal{H}_J = \{\sum_{j=1}^{d} \beta_j k(\boldsymbol{x}_j, .) : j \in$

$J\}$ and minimize $\mathcal{R}$ over $\mathcal{H}_J$, which can be rewritten as

$$\underset{\beta_J}{\operatorname{argmin}} \mathcal{R}_J(\beta_J) = \frac{\gamma_A}{2}\beta_J^T K_{JJ}\beta_J + \frac{\gamma_I}{2}\beta_J^T K_{JN} M K_{NJ}\beta_J + \frac{1}{2}\|Y - K_{LJ}\boldsymbol{\beta}_J\|^2 \quad (5.2)$$

Here, Y is the label vector, N denotes indices of labeled and unlabeled examples, L denotes the indices of labeled examples, and the notation K_{JL} refers to the gram matrix between points in J and L. The basis set J is incrementally built by a greedy approach. There are two approaches for selecting the new basis function to insert. The *pre-backfitting* approach computes $(\beta_J^\star, \beta_j^\star) = \operatorname{argmin}_{(\beta_J,\beta_j)} \mathcal{R}_{J\cup\{j\}}$ and gives the score $\mathcal{R}_{J\cup\{j\}}(\beta_J^\star; \beta_j^\star)$ to a candidate basis function indexed by j; the candidate with the minimum score is picked. This method is expensive because of the need to do a full update for each candidate. The *post-backfitting* method computes $\beta_j^\star = \operatorname{argmin}_{\beta_j} \mathcal{R}_{J\cup\{j\}}(\beta_J, \beta_j)$ and sets the score to $\mathcal{R}_{J\cup\{j\}}(\beta_J; \beta_j^\star)$ keeping β_J fixed at its previous value, i.e., at the minimizer of $\mathcal{R}_J$; this method is cheap and effective and so we employ it in our implementation. By maintaining the Cholesky decomposition of the Hessian of $\mathcal{R}_J$ and updating it as new basis functions are added, solutions can be efficiently computed. For the sake of efficiency, the candidate basis index j is restricted to a set of κ indices, say $\kappa = 50$. Since M is typically the Laplacian (or a power series in it) of a highly sparse graph, matrix vector products with M can be done in $O(n)$ time. Because of this, the $O(nd^2)$ complexity of kernel matching pursuit for selecting d basis functions in the supervised case can be retained also for sparse manifold regularization, even though its quadratic objective function is more complex.

5.1.2 l_1 Regularization

Another effective way of obtaining sparse solutions is via l_1 regularization (the Lasso model [93]) by considering the minimization of

$$\mathcal{R}(\beta) + \gamma_S \|\beta\|_1 \tag{5.3}$$

where β is the coefficient vector corresponding to all basis functions, $\mathcal{R}(\beta)$ is the corresponding objective function Eqn. 5.1, and γ_S is the l_1 regularization parameter. For a given γ_S, we have $\beta_j \neq 0$ for a subset J of indices. Optimality for (5.3) implies

$$g_j + \gamma_S \text{ sign}(\beta_j) = 0 \quad \forall j \in J, \qquad |g_j| < \gamma_S \quad \forall j \notin J \tag{5.4}$$

where g is the gradient of $\mathcal{R}(\beta)$.

As γ_S is decreased from a large value towards zero, β moves from 0 to the solution of (5.1). As shown in [73] the solution path with respect to γ_S is piecewise linear. From (5.4) it is clear that the break points correspond to γ_S values where either $|g_j|$ becomes equal to γ_S for some $j \notin J$ or β_j changes sign for a $j \in J$. Because of occurrences of the second type the size of J can decrease at some break points, but overall, the size of J increases as γ_S is decreased towards zero. Roughly speaking the method has a complexity of $O(n^2 d)$ for choosing d basis functions. The LARS procedure [46] is obtained by following the same sequence of steps, but omitting deletions of indices from J. Often LARS produces a solution path quite close to the Lasso path.

5.2 An Empirical Study

We took 6 datasets (see Table 5.1) frequently used in semi-supervised learning literature and split them into training (labeled and unlabeled) and validation subsets in the ratio 3 : 1. We randomly labeled 5% of the training data (except for 2MOONS where we took one positive and one negative example). All results below are averaged over 10 random choices of labels. Unless specified otherwise, we tune hyper-parameters with respect to the validation set[1] and look at error rates in predicting labels of unlabeled examples; our observations also hold for out-of-sample prediction. 2MOONS and G50C are artificial datasets [30]. Examples in G50C are generated from two standard normal multi-variate Gaussians, the labels correspond to the Gaussians, and the means are located in 50-dimensional space such that the Bayes error is 5%. COIL20BIN is a binary version of the COIL20 dataset [30], which contains images of 20 objects taken at various poses. We took the first 10 objects as positive class and remaining as negative class. USPS25 is taken from the benchmark study in [26] where the task is to separate images of digits 2 and 5 from the remaining digits in the USPS test set. COIL20BIN and USPS25 are thus expected to have multiple sub-clusters with highly non-linear structures in both classes, which makes it challenging to learn with few basis functions as well as few labels. MNIST3VS8 is a subset of the MNIST dataset where we did PCA denoising as a preprocessing step; the task here is to distinguish between digits 3 and 8. Finally, PCMAC is a text dataset that has also previously been used for semi-supervised experiments in [30].

Remark: We use LAPRLS to denote the full solution of (5.1); SPLAPRLS to denote the sparse solution (5.2) where the basis J is specified by context; GREEDY, LASSO,

1. We took a sizeable validation set to avoid the model selection issue which is not the main topic of this paper.

Table 5.1: Datasets with *dim* features; l labeled, u unlabeled, and v validation examples. In experiments below, we focus on basis sets of size $d_{\max}$.

DATASET	*dim*	l	$n = l + u$	v	$d_{\max}$	DOMAIN
2MOONS	2	2	450	150	15	SYNTHETIC
G50C	50	21	412	138	42	SYNTHETIC
COIL20BIN	1024	54	1080	360	108	IMAGE
USPS25	241	57	1125	375	113	IMAGE
MNIST3VS8	100	75	1500	500	150	IMAGE
PCMAC	7511	73	1459	487	146	TEXT

LARS, RANDOM denote the sparse Laplacian RLS where the basis set is obtained by kernel matching pursuit (using post-backfitting which is cheaper), Lasso, Lars or random selection respectively. RLS(l) denotes the RLS solution with only l labels ignoring unlabeled data, RLS(n, d) denotes the RLS solution in the fully supervised situation where all n examples are labeled, but d basis functions are used (the basis set is chosen by standard kernel matching pursuit), RLS(n) denotes the fully supervised case with all the basis functions.

5.2.1 Behaviour of Sparse Greedy Method

We begin by observing the behaviour of GREEDY. First, we optimized LAPRLS over a grid of graph hyper-parameters (number of nearest neighbors, degree of the Laplacian), the width σ of the Gaussian Kernel $K(\boldsymbol{x}, \boldsymbol{z}) = \exp -\frac{\|\boldsymbol{x}-\boldsymbol{z}\|^2}{2\sigma^2}$, and the regularization parameters γ_A, γ_I. Then, keeping these hyperparameters fixed, we ran GREEDY with $d_{\max} = n$, noting the error rate as basis functions are added one by one, and GREEDY gradually approaches the full LAPRLS solution. In Figure 5.1, four learning curves are shown for each dataset: (i) A horizontal "baseline" which corresponds to RLS(l) with optimized hyperparameters. For semi-supervised learning to be useful, this baseline has to be outperformed. (ii) GREEDY, which uses $u = n - l$ unlabeled examples together with l labeled examples, (iii) RANDOM where the ba-

sis functions are inserted randomly (without replacement), and (iv) RLS(n, d) whose performance may be viewed as a lower-bound on the error rates one can achieve with semi-supervised learning. Since the training set is fully labeled, we plot validation performance in this case. The best we can hope is that semi-supervised learning curves would approach this performance even with few basis functions.

The following observations can be made: (1) Typically, GREEDY performs better than RANDOM. (2) On most datasets, however, the error rates of GREEDY decrease slowly, beating the RLS(l) baseline only after a fairly large number of basis have been added. (3) We focus on a particular point on the x-axis of these learning curves where the compression is 10%, i.e., $d_{max} = 0.1n$. These d_{max} values are also tabulated in Table 5.1 and marked by the vertical line in Figure 5.1. The parameters were held fixed (at values that optimize LAPRLS) so that the limiting behaviour could be studied. However, when $d_{\max} = 0.1n$, the capacity of the hypothesis space is highly restricted and these hyper-parameters become suboptimal: e.g., on USPS25 , a very narrow kernel width optimizes the full solution but is not suitable when $d_{\max} = 0.1n$. In Table 5.2, we report the GREEDY performance with re-tuned hyperparameters. Typically, in the sparse setting a relatively wider kernel width is prefered. Note that while there are significant improvements with a better choice of hyper-parameters (compare FIXED and TUNED columns of Table 5.2), on many datasets GREEDY still does not significantly outperform the RLS(l) performance where unlabeled data is simply ignored (compare TUNED and RLS(l) columns). The performance is often also "far" from that attained by the full LAPRLS solution (compare LAPRLS and TUNED columns). (4) Finally, note that the behaviour of sparse greedy method in the supervised case (last two columns of Table 5.2) is very different from the semi-supervised case (compare TUNED and LAPRLS columns): In most situations, one can indeed get close to the full solution with greedy basis selection in the supervised case,

but this is not true for the semi-supervised variant of the algorithm.

Table 5.2: Improvements with hyper-parameter tuning.

Dataset	LAPRLS	FIXED	TUNED	RLS(l)	RLS($n, d_{\max}$)	RLS(n)
2MOONS	0.44	16.45	8.42	20.04	0	0
G50C	11.20	26.79	18.11	18.90	4.37	4.35
COIL20BIN	6.63	19.09	9.97	13.04	2.22	0.28
USPS2VS5	7.26	12.49	11.36	11.29	4.27	4.0
MNIST3VS8	3.25	7.06	5.53	5.85	1.40	1.20
PCMAC	9.60	14.65	10.40	14.78	3.08	2.26

5.2.2 Benchmarking the quality of Basis selected

In Table 5.3, we report the error rates given by a number of alternative basis selection schemes which, among the ones already introduced, include (1) UNIFORM: Intuitively, a good choice of basis functions seems to be one that covers the data manifold "uniformly". The graph transduction solution maps nodes in the data-similarity graph to the real line in a locality and label preserving manner. In this method, we sort the outputs of the graph transduction solution and pick the basis functions corresponding to indices at intervals $\lceil n/d \rceil$. (2) GREEDY (RIDGE): In preparation for the multiscale method described next, in this method we replace the ambient RKHS regularizer with $\frac{\gamma_A}{2}\|\beta\|_2^2$ (γ_A was appropriately scaled to maintain the same ratio between the three terms). Except for this difference, this method is identical to GREEDY. (3) GREEDY (MS): In this method, we expand the "dictionary" to include basis functions of multiple kernel widths $2^i\sigma_0, i = -3, \ldots 3$ where σ_0 is the kernel width used for all the other methods. Then GREEDY(RIDGE) is applied in a multiscale setting. The ridge regularizer has to be used instead of the RKHS norm since the gram matrix associated with basis functions at multiple scales is no longer guaranteed to be positive definite. (4) SUP: This method is included as a gold standard for basis selection. We

take the basis set given by the fully supervised sparse greedy method RLS($n, d_{\max}$). From the last two columns of Table 5.2, we know this basis set returns performance close to RLS using all the labels and all the basis functions. Thus, there exists a set of coefficients that go with this basis set that define a very good decision boundary. We then obtained the SPLAPRLS solution on these basis functions. Results are tabulated in the final column of Table 5.3. Of course, this method is not feasible for semi-supervised learning since the labels of unlabeled examples are truly unknown.

Note that all methods use the same hyper-parameters. Thus, the difference in performance is only due to difference in basis set selected.

The following observations can be made: (1) We see that none of the 7 sparse, semi-supervised basis selection methods are able to do substantially better than the tuned GREEDY results reported in Table 5.2. For comparison, the first column reports again the RLS(l) baseline. (2) The final column suggests that there exists better basis functions on which SPLAPRLS performance improves substantially, but semi-supervised schemes are unable to find them. (3) However, on some datasets (USPS25, 2MOONS) even with a high quality basis set, SPLAPRLS is unable to give satisfactory results. As we will see in the following sections, there is scope for significant further improvements on all datasets.

Note that care should be taken in comparing other methods against GREEDY in Table 5.3 since the hyperparameter values were chosen to optimize the GREEDY. In particular, we observed that LASSO performance could be significantly improved on some datasets with hyperparameter tuning, but the main conclusion of this section remains the same: in their current form, the sparse manifold regularizaton techniques we have explored fail to find very good basis sets; and sometimes, even when given a good basis set SPLAPRLS performance may not be completely satisfactory.

Table 5.3: Comparison of Basis Selection Schemes

Dataset → Algorithm ↓	2MOONS	G50C	COIL20BIN	USPS2VS5	MNIST3VS8	PCMAC
RLS(l)	20.04	18.90	13.04	11.29	5.85	14.78
RANDOM	17.50	15.17	12.99	14.31	5.27	12.25
UNIFORM	10.71	16.50	12.43	14.58	5.66	11.67
GREEDY	8.42	18.11	9.97	11.36	5.53	10.40
GREEDY(r)	8.97	15.75	10.37	11.37	5.31	10.21
GREEDY(ms)	18.44	43.30	12.82	15.07	7.54	39.72
LASSO	17.03	17.60	12.75	11.87	5.51	39.05
LARS	17.57	18.06	13.01	12.67	5.80	22.27
SUP	2.21	14.37	8.85	11.10	3.77	7.86

5.3 Intrinsic Regularization On Restricted Function Spaces

Consider two 2-dimensional datasets shown in Figure 5.3.

In the T dataset, the unlabeled data is distributed over two perpendicular lines of unit length corresponding to the two classes respectively so that a horizontal classification boundary is desirable[2]. With respect to this particular geometry, consider the intrinsic norm of a linear function on $\mathcal{R}^2$, $f_w(x,y) = \|w\|\cos(\theta)x + \|w\|\sin(\theta)y$, $(x,y) \in \mathcal{R}^2$, where $\|w\|$ is the ambient norm of the function and θ is the orientation of the associated weight vector with respect to the x-axis. The intrinsic regularizer is given by $\int_{\mathcal{M}} \|\nabla_{\mathcal{M}} f_w\|^2 = (\|w\|cos(\theta))^2 + (\|w\|sin(\theta))^2 = \|w\|^2$ where $\mathcal{M}$ is the data manifold (the two perpendicular lines) and $\nabla_{\mathcal{M}} f$ is the gradient of the function projected along the manifold. Since the ambient and intrinsic norms turn out to be the same, manifold regularization with a linear kernel simply ignores unlabeled data, returning the supervised solution with regularization parameter $(\gamma_A + \gamma_I)$! On the other hand, on the = dataset, the intrinsic norm is $\|w\|^2 cos^2(\theta)$ which, for a fixed ambient norm, is minimized by the horizontal separator as desired. These simple

2. If the data were fully labeled, linear RLS would learn a horizontal separator.

examples illustrate that when working with function spaces of restricted capacity, the intrinsic regularization penalty is appropriate only for a restricted set of geometries, which real-world data may or may not satisfy (see [32] for discussion on related issues). Note that standard graph regularization and manifold regularization with universal kernels do not show similar behaviour as their underlying hypothesis space includes all functions defined over the set of labeled and unlabeled examples. Under the conjecture that such effects also happen in the sparse, semi-supervised situation, we now explore two simple methods to overcome the problem.

5.3.1 Sigmoid Method

In this method, we tighten the intrinsic regularizer by passing outputs through a sigmoid. For a fixed basis set given by the index set J, instead of (5.2) we solve,

$$\underset{\beta_J}{\operatorname{argmin}}\, \tilde{\mathcal{R}}_J(\beta_J) = \frac{\gamma_A}{2}\beta_J^T K_{JJ}\beta_J + \frac{\gamma_I}{2}\boldsymbol{s}^T M \boldsymbol{s} + \frac{1}{2}\|Y - K_{LJ}\boldsymbol{\beta}_J\|^2 \tag{5.5}$$

where $s_i = (1+e^{-\rho o_i})^{-1}$ are values obtained by tightening the outputs $o_i = (K_{NJ}\boldsymbol{\beta}_J)_i$ under a sigmoid function given by a parameter ρ. The gradient of $\tilde{\mathcal{R}}$ can be easily computed and passed to, say, a non-linear conjugate gradient solver. Note that unlike $\mathcal{R}_J$ in (5.2), $\tilde{\mathcal{R}}_J$ is not convex, and we are now open to local minima problems. Non-convexity can be a serious problem for semi-supervised learning as evidenced by the failure of Transductive SVM methods on problems where a strong non-linear manifold structure exists [28]. However, to illustrate the effect of using the new intrinsic regularization term, we report in Table 5.4 the performance of SPLAPRLS, and its sigmoid version for a fixed basis given by RLS$(n, d_{\max})$.

We initialized the sigmoid method from the solution given by RLS$(n, d_{\max})$ (to avoid local minima issues) and ran non-linear CG with a stringent stopping criterion

Table 5.4: Improvements with Sigmoid Method

Dataset	splaprls	Sigmoid
2moons	2.21	0.47
g50c	14.37	11.30
coil20bin	8.85	4.09
usps2vs5	11.10	6.11
mnist3vs8	3.77	1.92
pcmac	7.86	6.42

(to ensure that the initial point is not returned pre-maturely). Results in Table 5.4 should be interpreted with care to be the error rate at approximately the local minima in the non-convex function (5.5) nearest to the fully supervised solution. As can be seen, the results improve significantly. This method is worth exploring in conjunction the greedy basis selection approach. However, we postpone this direction for future work.

5.3.2 Slack Method

Instead of passing the outputs o_i through a sigmoid, in this method we allow "slacks" $o_i + \xi_i$ and setup the problem in the following way,

$$\underset{\boldsymbol{\beta}_J, \boldsymbol{\xi}}{\operatorname{argmin}} \frac{\gamma_A}{2} \boldsymbol{\beta}_J^\top K_{JJ} \boldsymbol{\beta}_J + \frac{\gamma_I}{2} (K_{NJ} \boldsymbol{\beta}_J + \boldsymbol{\xi})^\top M (K_{NJ} \boldsymbol{\beta}_J + \boldsymbol{\xi}) + \frac{1}{2} \|Y - K_{LJ} \boldsymbol{\beta}_J - \boldsymbol{\xi}_L\|^2 + \frac{\nu}{2} \|\boldsymbol{\xi}\|^2 \tag{5.6}$$

Reparameterizing, $\boldsymbol{f} = K_{NJ} \boldsymbol{\beta}_J + \boldsymbol{\xi}$, we can rewrite the problem above as,

$$\underset{\boldsymbol{\beta}_J, \boldsymbol{f}}{\operatorname{argmin}} \bar{\mathcal{R}}_J(\boldsymbol{\beta}_J, \boldsymbol{f}) = \frac{\gamma_A}{2}\boldsymbol{\beta}_J^\top K_{JJ}\boldsymbol{\beta}_J + \frac{\gamma_I}{2}\boldsymbol{f}^\top M\boldsymbol{f} + \frac{1}{2}(\boldsymbol{f} - \bar{Y})^\top I_L(\boldsymbol{f} - \bar{Y}) + \frac{\nu}{2}\|\boldsymbol{f} - K_{NJ}\boldsymbol{\beta}_J\|^2 \tag{5.7}$$

where $\bar{Y}$ has Y for labeled entries and 0 otherwise, and I_L is a diagonal matrix with 1 for labeled entries and 0 otherwise.

The SLACK method may be interpreted as a form of *co-regularized least squares* (see Chapter 4 and [88]) for multiview semi-supervised learning where we maximize consensus between graph transduction and functions in the span of a set of basis functions in a least squares sense (last term above). Here, ν is interepreted as a coupling parameter and the two views may be interpreted as an ambient and an intrinsic view. We propose a simple alternating optimization of the problem above.

Optimizing $\boldsymbol{f}$ keeping $\boldsymbol{\beta}$ fixed: We solve the sparse linear system,

$$(\gamma_I M + I_L + \nu I)\boldsymbol{f} = (\bar{Y} + \nu\mathbf{o}) \tag{5.8}$$

where $\mathbf{o} = K_{NJ}\boldsymbol{\beta}$. Note that this step performs regression on graphs similar to graph transduction, with $\bar{Y} + \nu\mathbf{o}$ as the target vector. Since matrix-vector products with M are cheap due to the sparsity of the underlying graph, and with the use of seeding (starting conjugate gradient iterations from the previous solution), this step is not expensive.

Optimizing $\boldsymbol{\beta}$ keeping $\boldsymbol{f}$ fixed: This reduces to a small linear system,

$$[\gamma_A K_{JJ} + \nu K_{JN}K_{NJ}]\,\boldsymbol{\beta}_J = \nu K_{JN}\boldsymbol{f} \tag{5.9}$$

corresponding to regression on the current $\boldsymbol{f}$ to learn $\boldsymbol{\beta}_J$.

For a given basis set indexed by J, we keep alternating until the relative improvement in the objective function falls below a small threshold.

The GREEDY method can be combined with the SLACK method to find a basis set incrementally in the following way: To score a candidate basis function indexed by j having already selected J, we first find $\beta_j^\star = \text{argmin}_{\beta_j} \bar{R}_{J\cup\{j\}}$ keeping $\boldsymbol{\beta}_J$ and $\boldsymbol{f}$ fixed. A closed form expression can be derived. Next, we compute the revised outputs $\bar{\mathbf{o}} = \mathbf{o} + \beta_j^\star K_{Nj}$ and do a few conjugate gradient iterations[3] for the linear system $(\gamma_I M + I_L + \nu I)\bar{\boldsymbol{f}} = (\bar{Y} + \nu\bar{\mathbf{o}})$ starting from the current $\boldsymbol{f}$. The objective value $\bar{\mathcal{R}}(\beta_J; \beta_j^\star, \bar{\boldsymbol{f}})$ is used as the score for candidate j.

The first column in Table 5.5 reports performance of GREEDY. For the same fixed basis found by GREEDY, SLACK makes significant improvements. Finally, the combination of SLACK and GREEDY is able to recover the performance of full LAPRLS solution on most datasets.

Table 5.5: Performance of GREEDY and improvements with SLACK and its combination with GREEDY

DATASET	GREEDY	SLACK	GREEDY+SLACK	LAPRLS
2MOONS	8.42	4.96	0.45	0.44
G50C	18.11	9.97	9.28	11.20
COIL20BIN	9.97	9.08	8.50	6.63
USPS2VS5	11.36	7.42	6.81	7.26
MNIST3VS8	5.53	3.50	3.20	3.25
PCMAC	10.40	9.24	9.08	9.60

A fully decoupled method

The slack method may be viewed as coupling graph transduction with regression using a small set of basis functions. In a fully decoupled method, graph transduction

3. 50 in our implementation. We also re-tuned γ_A, γ_I, ν on the validation set.

is first used to complete the labeling of the data and then a standard supervised problem is solved. Note that if the real-valued solution of graph transduction is used as the label vector, then this fully decoupled method is a special case of SLACK in the limit $\nu \to 0, \gamma_A \to 0$ such that $\frac{\gamma_A}{\nu}$ is a constant hyperparameter. As before, this method can be combined with GREEDY to find a good basis set. Such a decoupled approach gets the following error rates on our datasets, 2MOONS: 0.45, G50C: 8.44, COIL20BIN: 8.01, USPS25: 9.04, MNIST3VS8: 3.02, PCMAC: 8.09. In problems where the manifold assumption is satisfied to a strong degree and the tradeoff between ambient and intrinsic regularization is not strictly necessary, we recommend this simple approach for sparse, semi-supervised learning, avoiding altogether the complication of using the intrinsic regularizer on restricted function spaces. On other problems coupling might be necessary e.g., USPS25. Another example is demonstrated in [107] where links between documents are used to derive a graph regularizer for web page categorization, but good labels are difficult to induce based on the link graph alone.

5.4 Conclusion

One may be inclined to treating sparse methods as black boxes with which to sparsify semi-supervised solutions. However, when a function space is severely restricted, one needs to be careful whether the manifold/cluster assumptions for semi-supervised learning can still be effectively implemented. We believe that the main contribution of this paper is to provide insight into the nature of sparse manifold regularization. We have suggested modifications to the basic sparse techniques that eventually lead to performance close to the full manifold regularization solution by overcoming problems related to the intrinsic regularizer. As future work, we plan to explore these modifications in conjunction with l_1 regularization and multi-scale methods.

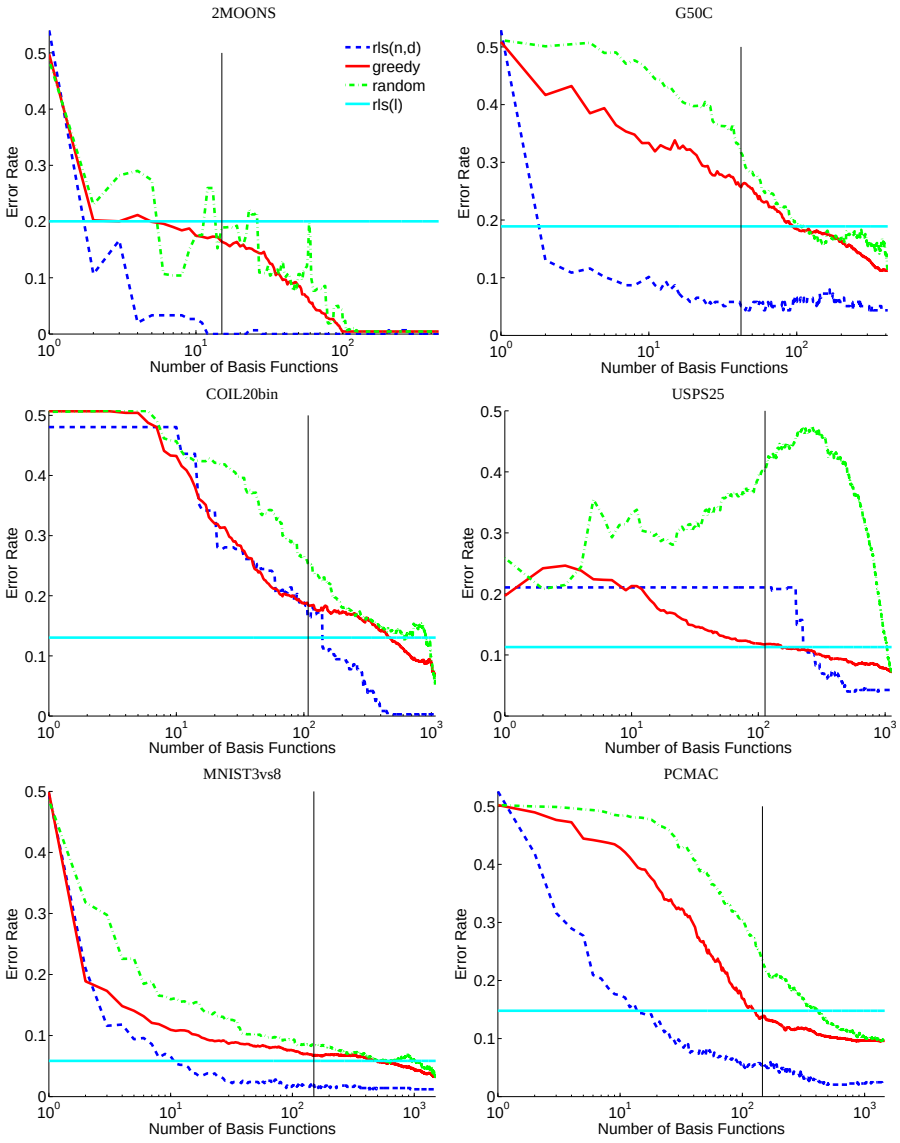


Figure 5.1: How sparse methods approach the full solution.

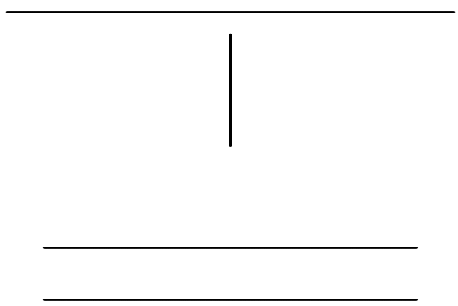


Figure 5.2: T (left) and = (right) datasets

CHAPTER 6

LARGE-SCALE SEMI-SUPERVISED LINEAR METHODS

In this chapter, we present a family of semi-supervised linear support vector classifiers that are designed to handle partially-labeled sparse datasets with possibly very large number of examples and features. At their core, our algorithms employ recently developed Modified Finite Newton primal techniques. We provide (1) a fast, multi-switch implementation of linear Transductive SVM (TSVM) that is significantly more efficient and scalable than currently used dual techniques, (2) a large-scale variant of the Deterministic Annealing (DA) algorithm for optimizing semi-supervised SVMs. We conduct an empirical study on several classification tasks which confirms the value of these methods in large scale semi-supervised settings. Our algorithms are implemented in SVM_{lin}*, a public domain software package.*

6.1 Introduction

Consider the following situation: In a single web-crawl, search engines like Yahoo! and Google index billions of documents. Only a very small fraction of these documents can possibly be hand-labeled by human editorial teams and assembled into topic directories. In information retrieval relevance feedback, a user labels a small number of documents returned by an initial query as being relevant or not. The remaining documents form a massive collection of unlabeled data. Despite its natural and pervasive need, solutions to the problem of utilizing unlabeled data with labeled examples have only recently emerged in machine learning literature. Whereas the abundance of unlabeled data is frequently acknowledged as a motivation in most papers, the true potential of semi-supervised learning in large scale settings is yet to be systematically explored. This appears to be partly due to the lack of scalable tools

to handle large volumes of data.

In this chapter, we propose extensions of linear Support Vector Machines (SVMs) for semi-supervised classification. Linear techniques are often the method of choice in many applications due to their simplicity and interpretability. When data appears in a rich high-dimensional representation, linear functions often provide a sufficiently complex hypothesis space for learning high-quality classifiers. This has been established, for example, for document classification with linear SVMs in numerous studies.

We highlight the main contributions of this chapter.

1. We outline an implementation for a variant of TSVM [57] designed for linear semi-supervised classification on large, sparse datasets. As compared to currently used dual techniques (e.g., as implemented in $\text{SVM}^{\text{light}}$), our method effectively exploits data sparsity and linearity of the problem to provide superior scalability. Additionally, we propose a multiple switching heuristic that further improves TSVM training by an order of magnitude. These speed enhancements turn TSVM into a feasible tool for large scale applications.

2. We propose a large-scale linear variant of the Deterministic Annealing (DA) technique (Chapter 3).

3. We conduct an experimental study on many high-dimensional document classification tasks involving hundreds of thousands of examples. This study clearly shows the utility of these tools for very large scale problems.

4. We outline an implementation of Linear Laplacian RLS and SVM using similar techniques and building on the observations made in Chapter 5. This method can be applied in settings where a similarity graph is available encoding pairwise relations between examples (e.g., hyperlinked documents).

The modified finite Newton algorithm of [62] for fast training of linear SVMs, a key subroutine for our algorithms, is outlined in section 6.2. Its semi-supervised extensions, (TSVM and DA) of this algorithm are presented in sections 6.3 and 6.4. Experimental results with this implementation are reported in section 6.5. Section 6.6 contains some concluding comments.

All the algorithms described in this chapter are implemented in a public domain software, SVM_{lin} (see section 6.5) which can be used for fast training of linear SVMs for supervised and semi-supervised classification problems.

6.2 Modified Finite Newton Linear l_2-SVM

The modified finite Newton l_2-SVM method [62] (abbreviated l_2-SVM-MFN) is a recently developed training algorithm for linear SVMs that is ideally suited to sparse datasets with large number of examples and possibly large number of features.

Given a binary classification problem with l labeled examples $\{\boldsymbol{x}_i, y_i\}_{i=1}^{l}$ where the input patterns $\boldsymbol{x}_i \in \mathbb{R}^d$ (e.g documents) and the labels $y_i \in \{+1, -1\}$, l_2-SVM-MFN provides an efficient primal solution to the following SVM optimization problem:

$$\mathbf{w}^\star = \underset{\mathbf{w}\in\mathbb{R}^d}{\text{argmin}} \frac{1}{2}\sum_{i=1}^{l} l_2(y_i \mathbf{w}^T \boldsymbol{x}_i) + \frac{\gamma}{2}\|\mathbf{w}\|^2 \tag{6.1}$$

where l_2 is the l_2-SVM loss given by $l_2(z) = \max(0, 1-z)^2$, γ is a real-valued regularization parameter[1] and the final classifier is given by $\text{sign}(\mathbf{w}^{\star T}\boldsymbol{x})$.

This objective function differs from the standard SVM problem in some respects. First, instead of using the hinge loss as the data fitting term, the square of the hinge loss (or the so-called quadratic soft margin loss function) is used. This makes the

1. $\gamma = 1/C$ where C is the standard SVM parameter.

objective function continuously differentiable, allowing easier applicability of gradient techniques. Secondly, the bias term ("b") is also regularized. In the problem formulation of Eqn. 6.1, it is implicitly assumed that an additional component in the weight vector and a constant feature in the example vectors have been added to indirectly incorporate the bias. This formulation combines the simplicity of a least squares aspect with algorithmic advantages associated with SVMs.

We consider a version of l_2-SVM-MFN where a weighted quadratic soft margin loss function is used.

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{2} \sum_{i \in \jmath(\mathbf{w})} c_i l_2(y_i \mathbf{w}^T x_i) + \frac{\gamma}{2} \|\mathbf{w}\|^2 \tag{6.2}$$

Here we have rewritten Eqn. 6.1 in terms of the support vector set $\jmath(\mathbf{w}) = \{i : y_i \ (\mathbf{w}^T x_i) < 1\}$. Additionally, the loss associated with the i^{th} example has a cost c_i. $f(\mathbf{w})$ refers to the objective function being minimized, evaluated at a candidate solution $\mathbf{w}$. Note that if the index set $\jmath(\mathbf{w})$ were independent of $\mathbf{w}$ and ran over all data points, this would simply be the objective function for weighted linear regularized least squares (RLS).

Following [62], we observe that f is a strictly convex, piecewise quadratic, continuously differentiable function having a unique minimizer. The gradient of f at $\mathbf{w}$ is given by:

$$\nabla \ f(\mathbf{w}) = \gamma \ \mathbf{w} + X_{\jmath(\mathbf{w})}^T C_{\jmath(\mathbf{w})} \left[X_{\jmath(\mathbf{w})} \mathbf{w} - Y_{\jmath(\mathbf{w})} \right]$$

where $X_{\jmath(\mathbf{w})}$ is a matrix whose rows are the feature vectors of training points corresponding to the index set $\jmath(\mathbf{w})$, $Y_{\jmath(\mathbf{w})}$ is a column vector containing labels for these points, and $C_{\jmath(\mathbf{w})}$ is a diagonal matrix that contains the costs c_i for these points along its diagonal.

l_2-SVM-MFN is a primal algorithm that uses the Newton's Method for unconstrained minimization of a convex function. The classical Newton's method is based on a second order approximation of the objective function, and involves updates of the following kind:

$$\mathbf{w}^{k+1} = \mathbf{w}^k + \delta^k \; n^k \tag{6.3}$$

where the step size $\delta^k \in \mathbb{R}$, and the Newton direction $\mathbf{n}^k \in \mathbb{R}^d$ is given by: $\mathbf{n}^k = -[\nabla^2 \; f(\mathbf{w}^k)]^{-1} \nabla \; f(\mathbf{w}^k)$. Here, $\nabla \; f(\mathbf{w}^k)$ is the gradient vector and $\nabla^2 \; f(\mathbf{w}^k)$ is the Hessian matrix of f at $\mathbf{w}^k$. However, the Hessian does not exist everywhere, since f is not twice differentiable at those weight vectors $\mathbf{w}$ where $\mathbf{w}^T \boldsymbol{x}_i = y_i$ for some index i.[2] Thus a generalized definition of the Hessian matrix is used. The modified finite Newton procedure proceeds as follows. The step $\bar{\mathbf{w}}^k = \mathbf{w}^k + \mathbf{n}^k$ in the Newton direction can be seen to be given by solving the following linear system associated with a weighted linear regularized least squares problem over the data subset defined by the indices $\jmath(\mathbf{w}^k)$:

$$\left[\gamma I + X^T_{\jmath(\mathbf{w}^k)} C_{\jmath(\mathbf{w}^k)} X_{\jmath(\mathbf{w}^k)}\right] \bar{\mathbf{w}}^k = X^T_{\jmath(\mathbf{w}^k)} C_{\jmath(\mathbf{w}^k)} Y_{\jmath(\mathbf{w}^k)} \tag{6.4}$$

where I is the identity matrix. Once $\bar{\mathbf{w}}^k$ is obtained, $\mathbf{w}^{k+1}$ is obtained from Eqn. 6.3 by setting $\mathbf{w}^{k+1} = \mathbf{w}^k + \delta^k(\bar{\mathbf{w}}^k - \mathbf{w}^k)$ after performing an exact line search for δ^k, i.e., by exactly solving a one-dimensional minimization problem:

$$\delta^k = \underset{\delta \geq 0}{\operatorname{argmin}} \, \phi(\delta) = f\left(\mathbf{w}^k + \delta(\bar{\mathbf{w}}^k - \mathbf{w}^k)\right) \tag{6.5}$$

2. In the neighborhood of such a $\mathbf{w}$, the index i leaves or enters $\jmath(\mathbf{w})$. However, at $\mathbf{w}$, $y_i \mathbf{w}^T \boldsymbol{x}_i = 1$. So f is continuously differentiable inspite of these index jumps.

The modified finite Newton procedure has the property of finite convergence to the optimal solution. The key features that bring scalability and numerical robustness to l_2-SVM-MFN are: (a) Solving the regularized least squares system of Eqn. 6.4 by a numerically well-behaved Conjugate Gradient scheme referred to as CGLS [48], which is designed for large, sparse data matrices X. The benefit of the least squares aspect of the loss function comes in here to provide access to a powerful set of tools in numerical computation. (b) Due to the one-sided nature of margin loss functions, these systems are required to be solved over only restricted index sets $\jmath(\mathbf{w})$ which can be much smaller than the whole dataset. This also allows additional heuristics to be developed such as terminating CGLS early when working with a crude starting guess like 0, and allowing the following line search step to yield a point where the index set $\jmath(\mathbf{w})$ is small. Subsequent optimization steps then work on smaller subsets of the data. Below, we briefly discuss the CGLS and Line search procedures. We refer the reader to [62] for full details.

6.2.1 CGLS

CGLS [48] is a special conjugate-gradient solver that is designed to solve, in a numerically robust way, large, sparse, weighted regularized least squares problems such as the one in Eqn. 6.4. Starting with a guess solution, several specialized conjugate-gradient iterations are applied to get $\bar{\mathbf{w}}^k$ that solves Eqn. 6.4. The major expense in each iteration consists of two operations of the form $X_{j(\mathbf{w}^k)}p$ and $X^T_{j(\mathbf{w}^k)}q$. If there are n_0 non-zero elements in the data matrix, these involve $O(n_0)$ cost. It is worth noting that, as a subroutine of l_2-SVM-MFN, CGLS is typically called on a small subset, $X_{j(\mathbf{w}^k)}$ of the full data set. To compute the *exact solution* of Eqn. 6.4, r iterations are needed, where r is the rank of $X_{j(\mathbf{w}^k)}$. But, in practice, such an exact

solution is unnecessary. CGLS uses an effective stopping criterion based on gradient norm for early termination (involving a tolerance parameter ϵ). The total cost of CGLS is $O(t_{cgls}n_0)$ where t_{cgls} is the number of iterations, which depends on ϵ and the condition number of $X_{j(\mathbf{w}^k)}$, and is typically found to be very small relative to the dimensions of $X_{j(\mathbf{w}^k)}$ (number of examples and features). Apart from the storage of $X_{j(\mathbf{w}^k)}$, the memory requirements of CGLS are also minimal: only five vectors need to be maintained, including the outputs over the currently active set of data points.

Finally, an important feature of CGLS is worth emphasizing. Suppose the solution $\mathbf{w}$ of a regularized least squares problem is available, i.e the linear system in Eqn. 6.4 has been solved using CGLS. If there is a need to solve a perturbed linear system, it is greatly advantageous in many settings to start the CG iterations for the new system with $\mathbf{w}$ as the initial guess. This is called *seeding*. If the starting residual is small, CGLS can converge much faster than with a guess of 0 vector. The utility of this feature depends on the nature and degree of perturbation. In l_2-SVM-MFN, the candidate solution $\mathbf{w}^k$ obtained after line search in iteration k is seeded for the CGLS computation of $\bar{\mathbf{w}}^k$. Also, in tuning γ over a range of values, it is valuable to seed the solution for a particular γ onto the next value. For the semi-supervised SVM implementations with l_2-SVM-MFN, we will seed solutions across linear systems with slightly perturbed label vectors, data matrices and costs.

6.2.2 Line Search

Given the vectors $\mathbf{w}^k$,$\bar{\mathbf{w}}^k$ in some iteration of l_2-SVM-MFN, the line search step requires us to solve Eqn. 6.5. The one-dimensional function $\phi(\delta)$ is the restriction of the objective function f on the ray from $\mathbf{w}^k$ onto $\bar{\mathbf{w}}^k$. Hence, like f, $\phi(\delta)$ is also a continuously differentiable, strictly convex, piecewise quadratic function with a

unique minimizer. ϕ' is a continuous piecewise linear function whose root, δ^k, can be easily found by sorting the break points where its slope changes and then performing a sequential search on that sorted list. The cost of this operation is negligible compared to the cost of the CGLS iterations.

6.2.3 Complexity

l_2-SVM-MFN alternates between calls to CGLS and line searches until the support vector set $\jmath(\mathbf{w}^k)$ stabilizes upto a tolerance parameter τ, i.e., if $\forall i \in \jmath(\mathbf{w}^k),\ y_i \bar{\mathbf{w}}^{kT} x_i < 1+\tau$ and $\forall i \notin \jmath(\mathbf{w}^k),\ y_i \bar{\mathbf{w}}^{kT} x_i \geq 1-\tau$. Its computational complexity is $O(t_{mfn} \bar{t}_{cgls} n_0)$ where t_{mfn} is the number of outer iterations of CGLS calls and line search, and $\bar{t}_{cgls}$ is the average number of CGLS iterations. The number of CGLS iterations to reach a relative error of ϵ can be bounded in terms of ϵ and the condition number of the left-hand-side matrix in Eqn 6.4 [15]. In practice, t_{mfn} and $\bar{t}_{cgls}$ depend on the data set and the tolerances desired in the stopping criterion, but are typically very small. As an example of typical behavior: on a Reuters [68] text classification problem (top level category CCAT versus rest) involving 804414 examples and 47236 features, $t_{mfn} = 7$ with a maximum of $t_{cgls} = 28$ CGLS iterations; on this dataset l_2-SVM-MFN converges in about 100 seconds on an Intel 3GHz, 2GB RAM machine[3]. The practical scaling is linear in the number of non-zero entries in the data matrix [62].

6.2.4 Other Loss functions

All the discussion in this chapter can be applied to other loss functions such as Huber's Loss and rounded Hinge loss using the modifications outlined in [62].

We also note a recently proposed linear time training algorithm for hinge loss [61].

3. For this experiment, γ is chosen as in [61]; $\epsilon = \tau = 10^{-6}$.

While detailed comparative studies are yet to be conducted, preliminary experiments have shown that l_2-SVM-MFN and the methods of [61] are competitive with each other (at their default tolerance parameters).

We now assume that in addition to l labeled examples, we have u unlabeled examples $\{\boldsymbol{x}'_j\}_{j=1}^{u}$. Our goal is to extend l_2-SVM-MFN to utilize unlabeled data, typically when $l \ll u$.

6.3 Fast Multi-switch Transductive SVMs

Recall from Chapter 3, Eqn. 3.1, that the following optimization problem is setup for standard TSVM[4]:

$$\min_{\mathbf{w},\{y'_j\}_{j=1}^{u}} \frac{\gamma}{2}\|\mathbf{w}\|^2 + \frac{1}{2l}\sum_{i=1}^{l} V(y_i\ \mathbf{w}^T\boldsymbol{x}_i) + \frac{\gamma'}{2u}\sum_{j=1}^{u} V(y'_j\ \mathbf{w}^T\boldsymbol{x}'_j)$$
$$\text{subject to: } \frac{1}{u}\sum_{j=1}^{u} \max[0, \text{sign}(\mathbf{w}^T\boldsymbol{x}'_j)] = r$$

where for the loss function $V(z)$, the hinge loss $l_1(z) = max(0, 1-z)$ is normally used. We use a slightly different notation than Chapter 3 to match the SVM_{lin} implementation. The labels on the unlabeled data, $y'_1 \ldots y'_u$, are $\{+1, -1\}$-valued variables in the optimization problem. In other words, TSVM seeks a hyperplane w and a labeling of the unlabeled examples, so that the SVM objective function is minimized, subject to the constraint that a fraction r of the unlabeled data be classified positive. Recall that SVM margin maximization in the presence of unlabeled examples can be interpreted as an implementation of the cluster assumption. In the optimization problem above, γ' is a user-provided parameter that provides control over the influ-

4. The bias term is typically excluded from the regularizer, but this factor is not expected to make any significant difference.

ence of unlabeled data. For example, if the data has distinct clusters with a large margin, but the cluster assumption does not hold, then γ' can be set to 0 and the standard SVM is retrieved. If there is enough labeled data, γ, γ' can be tuned by cross-validation. An initial estimate of r can be made from the fraction of labeled examples that belong to the positive class and subsequent fine tuning can be done based on validation performance.

This optimization is implemented in [57] by first using an inductive SVM to label the unlabeled data and then iteratively switching labels and retraining SVMs to improve the objective function. The TSVM algorithm wraps around an SVM training procedure. The original (and widely popular) implementation of TSVM uses the $\text{SVM}^{\text{light}}$ software. There, the training of SVMs in the inner loops of TSVM uses dual decomposition techniques. As shown by experiments in [62], in sparse, linear settings one can obtain significant speed improvements with l_2-SVM-MFN over $\text{SVM}^{\text{light}}$. Thus, by implementing TSVM with l_2-SVM-MFN, we expect similar improvements for semi-supervised learning on large, sparse datasets. The l_2-SVM-MFN retraining steps in the inner loop of TSVM are typically executed extremely fast by using seeding techniques. Additionally, we also propose a version of TSVM where more than one pair of labels may be switched in each iteration. These speed-enhancement details are discussed in the following subsections.

6.3.1 Implementing TSVM Using l_2-SVM-MFN

The TSVM algorithm with l_2-SVM-MFN closely follows the presentation in [57]. A classifier is obtained by first running l_2-SVM-MFN on just the labeled examples. Temporary labels are assigned to the unlabeled data by thresholding the soft outputs of this classifier so that the fraction of the total number of unlabeled examples that are

temporarily labeled positive equals the parameter r. Then starting from a small value of γ', the unlabeled data is gradually brought in by increasing γ' by a certain factor in the outer loop. This gradual increase of the influence of the unlabeled data is a way to protect TSVM from being immediately trapped in a local minimum (see discussion on Annealing in Chapter 3). An inner loop identifies pairs of unlabeled examples with positive and negative temporary labels such that switching these labels would decrease the objective function. l_2-SVM-MFN is then retrained with the switched labels, starting the CGLS/line-search iterations with the current classifier.

6.3.2 Multiple Switching

The TSVM algorithm presented in [57] involves switching a single pair of labels at a time. We propose a variant where upto S pairs are switched such that the objective function improves. Here, S is a user controlled parameter. Setting $S = 1$ recovers the original TSVM algorithm, whereas setting $S = u/2$ switches as many pairs as possible in the inner loop of TSVM. The implementation is conveniently done as follows:

1. Identify unlabeled examples with active indices and currently positive labels. Sort corresponding outputs in ascending order. Let the sorted list be L^+.

2. Identify unlabeled examples with active indices and currently negative labels. Sort corresponding outputs in descending order. Let the sorted list be L^-.

3. Pick pairs of elements, one from each list, from the top of these lists until either a pair is found such that the output from L^+ is greater than the output from L^-, or if S pairs have been picked.

4. Switch the current labels of these pairs.

Using arguments similar to Theorem 2 in [57], we can show the following.

Proposition 6.3.1. *Transductive L_2-SVM-MFN with multiple-pair switching converges in finite number of steps.*

Proof: The outer loop annealing loop clearly terminates in finite number of steps. Each call to l_2-SVM-MFN terminates in finite number of iterations due to Theorem 1 in [62]. We only need to show that label switching loop also has finite termination. Let $J(\mathbf{w}, Y')$ be the value of the TSVM objective function for some candidate weight vector $\mathbf{w}$ and candidate label vector $Y' = [y'_1 \dots y'_u]$ over the unlabeled data. Let $\mathbf{w}(Y'), Y'$ be the operating variables at the end of an iteration of loop 2 where $w(Y') = \text{argmin}_{\mathbf{w}\in\mathbb{R}^d} J(\mathbf{w}, Y')$. After switching labels, let the new operating label vector be Y''. It is easy to see that:

$$J(\mathbf{w}(Y'), Y') > J(w(Y'), Y'') \geq J(\mathbf{w}(Y''), Y'')$$

The second inequality follows since $w(Y'')$ minimizes $J(\mathbf{w}, Y'')$ over all $\mathbf{w}$. To see the first inequality observe that for any pair of data points (say with indices i, j) whose labels are switched, the following conditions are satisfied: $y'_i = 1, y'_j = -1, \mathbf{w}(Y')^T\boldsymbol{x}'_i < 1, -\mathbf{w}(Y')^T\boldsymbol{x}'_j < 1, \mathbf{w}(Y')^T\boldsymbol{x}'_i < \mathbf{w}(Y')^T\boldsymbol{x}'_j$. The terms contributed by this pair to the objective function decrease after switching labels since the switching conditions imply the following:

$$\begin{aligned}\max[0, 1 - \mathbf{w}^T\boldsymbol{x}'_i]^2 + \max[0, 1 + \mathbf{w}^T\boldsymbol{x}'_j]^2 = (1 - \mathbf{w}^T\boldsymbol{x}'_i)^2 + (1 + \mathbf{w}^T\boldsymbol{x}'_j)^2 \\ > (1 + \mathbf{w}^T\boldsymbol{x}'_i)^2 + (1 - \mathbf{w}^T\boldsymbol{x}'_j)^2 \geq \max[0, 1 + \mathbf{w}^T\boldsymbol{x}'_i]^2 + \max[0, 1 - \mathbf{w}^T\boldsymbol{x}'_j]^2\end{aligned}$$

Thus, swapping the labels of multiple pairs that satisfy the switching conditions

reduces the objective function.

Since at the end of consecutive iterations $J(\mathbf{w}(Y'), Y') > J(\mathbf{w}(Y''), Y'')$, Loop 2 must terminate in finite number of steps because there are only a finite number of possible label vectors. □

We are unaware of any prior work that suggests and evaluates this simple multiple-pair switching heuristic. Our experimental results in section 6.5 establish that this heuristic is remarkably effective in speeding up TSVM training while maintaining generalization performance.

6.3.3 Seeding

The effectiveness of l_2-SVM-MFN on large sparse datasets combined with the efficiency gained from seeding $\mathbf{w}$ in the re-training steps (after switching labels or after increasing γ') make this algorithm quite attractive. Consider an iteration of the inner label switching loop of TSVM where a new pair of labels has been switched, and the solution $\mathbf{w}$ from the last retraining of l_2-SVM-MFN is available for seeding. According to Theorem 1 in [62], when the last l_2-SVM-MFN converged, its solution $\mathbf{w}$ is given by the linear system[5]:

$$\left[\gamma I + X_{I(\mathbf{w})}^T C_{I(\mathbf{w})} X_{I(\mathbf{w})}\right] \mathbf{w} = X_{I(\mathbf{w})}^T C_{I(\mathbf{w})} Y$$

where Y is the current label vector. When labels Y_i, Y_j are switched, the label vector is updated as:

$$Y = Y + 2e_{ij}$$

5. The subsequent line search does not change this $\mathbf{w}$; therefore, the optimality conditions are checked immediately after the CGLS step

where e_{ij} is a vector whose elements zero everywhere except in the i^{th} and the j^{th} position which are +1 and -1 or -1 and +1 respectively. Note also that if $i, j \in \jmath(\mathbf{w})$ the re-training of L_2-SVM-MFN with $\mathbf{w}$ as the starting guess immediately encounters a call CGLS to solve the following perturbed system:

$$\left[\gamma I + X_{\jmath(w)}^T C_{\jmath(w)} X_{\jmath(w)}\right] \tilde{\mathbf{w}} = X_{\jmath(w)}^T C_{\jmath(\mathbf{w})} \left[Y + 2e_{ij}\right]$$

The starting residual vector r^0 is given by:

$$\begin{aligned} r^0 &= X_{\jmath(\mathbf{w})}^T C_{\jmath(\mathbf{w})} \left[Y + 2e_{ij}\right] - \left[\gamma I + X_{\jmath(\mathbf{w})}^T C_{\jmath(\mathbf{w})} X_{\jmath(\mathbf{w})}\right] \mathbf{w} \\ &= r(\mathbf{w}) + 2X_{\jmath(\mathbf{w})}^T C_{\jmath(\mathbf{w})} e_{ij} \\ &\leq \epsilon + 2\gamma' \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \end{aligned}$$

where $r(\mathbf{w})$ in the second step is the final residual of $\mathbf{w}$ which fell below ϵ at the convergence of the last re-training. In applications like Text categorization, TFIDF feature vectors are often length normalized and have positive entries. Therefore, $\|\boldsymbol{x}_i - \boldsymbol{x}_j\| \leq \sqrt{2}$. This gives the following bound on the starting residual:

$$r^0 \leq \epsilon + 2\sqrt{2}\gamma'$$

which is much smaller than a bound of $n\sqrt{n}\gamma'$ with a zero starting vector.

For a fixed γ', the complexity is $O(n_{switches} \bar{t}_{mfn} \bar{t}_{cgls} n_0)$, where $n_{switches}$ is the number of label switches. Typically, $n_{switches}$ is expected to strongly depend on the data set and also on the number of labeled examples. Since it is difficult to apriori estimate the number of switches, this is an issue that is best understood from empirical observations.

6.4 Linear Deterministic Annealing S^{3}VMs

In Chapter 3, we developed a Deterministic Annealing framework that provides better resistance to suboptimal local minima. The linear version of Eqn. 3.7 is as follows[6],

$$\begin{aligned}\mathbf{w}_T^\star = \underset{\mathbf{w},\{p_j\}_{j=1}^u}{\operatorname{argmin}} \quad & \frac{\gamma}{2}\|\mathbf{w}\|^2 + \frac{1}{2l}\sum_{i=1}^{l} l_2(y_i \mathbf{w}^T x_i) \\ & + \frac{\gamma'}{2u}\sum_{j=1}^{u}\Big(p_j l_2(\mathbf{w}^T x'_j) + (1-p_j) l_2(-\mathbf{w}^T x'_j)\Big) \\ & + \frac{T}{2u}\sum_{j=1}^{u}[p_j \ \log \ p_j + (1-p_j) \ \log \ (1-p_j)] \end{aligned} \tag{6.6}$$

Recall that the "temperature" parameter T parameterizes a family of objective functions. The objective function for a fixed T is minimized under the following class balancing constraint:

$$\frac{1}{u}\sum_{j=1}^{u} p_j = r \tag{6.7}$$

where r is the fraction of the number of unlabeled examples belonging to the positive class. As in TSVM, r is treated as a user-provided parameter. It may also be estimated from the labeled examples.

The solution to the optimization problem above is tracked as the temperature parameter T is lowered to 0.

We monitor the value of the objective function in the optimization path and return the solution corresponding to the minimum value achieved.

The optimization is done in stages, starting with high values of T and then grad-

6. We use a slightly different notation to match the SVM_{lin} implementation.

ually decreasing T towards 0. For each T, the objective function in Eqn 6.6 (subject to the constraint in Eqn 6.7) is optimized by alternating the minimization over $\mathbf{w}$ and $\mathbf{p} = [p_1 \dots p_u]$ respectively specializing the algorithm in Section 3.5 to the linear case. Fixing $\mathbf{p}$, the optimization over $\mathbf{w}$ is done by l_2-SVM-MFN with seeding. Fixing $\mathbf{w}$, the optimization over $\mathbf{p}$ can also be done easily as described below. Both these problems involve convex optimization and can be done exactly and efficiently. We now provide some details.

6.4.1 Optimizing $\mathbf{w}$

We describe the steps to efficiently implement the l_2-SVM-MFN loop for optimizing $\mathbf{w}$ keeping $\mathbf{p}$ fixed. The call to l_2-SVM-MFN is made on the data $\hat{X} = \begin{bmatrix} X^T & X'^T & X'^T \end{bmatrix}^T$ whose first l rows are formed by the labeled examples, and the next $2u$ rows are formed by the unlabeled examples appearing as two repeated blocks. The associated label vector and cost matrix are given by

$$\hat{Y} = [y_1, y_2 ... y_l, \overbrace{1, 1, ...1}^{u}, \overbrace{-1, -1 ... -1}^{u}]$$

$$C = diag \left[\overbrace{\frac{1}{l} ... \frac{1}{l}}^{l}, \overbrace{\frac{\gamma' \, p_1}{u} ... \frac{\gamma' \, p_u}{u}}^{u} \overbrace{\frac{\gamma'(1-p_1)}{u} ... \frac{\gamma'(1-p_u)}{u}}^{u} \right] \tag{6.8}$$

Even though each unlabeled data contributes two terms to the objective function, effectively only one term contributes to the complexity. This is because matrix-vector products, which form the dominant expense in l_2-SVM-MFN, are performed only on unique rows of a matrix. The output may be duplicated for duplicate rows. Infact, we can re-write the CGLS calls in l_2-SVM-MFN so that the unlabeled examples appear only once in the data matrix. Consider the CGLS call at some iteration where the

active index set is $\jmath = \jmath(\mathbf{w})$ for some current candidate weight vector $\mathbf{w}$:

$$\left[\gamma I + \hat{X}_\jmath^T C_\jmath \hat{X}_\jmath\right] \ \bar{w} = \hat{X}^T C_\jmath \hat{Y}_\jmath \tag{6.9}$$

Note that if $|\mathbf{w}^T \boldsymbol{x}'_j| \geq 1$, the unlabeled example $\boldsymbol{x}'_j$ appears as one row in the data matrix $\hat{X}_\jmath$ with label given by $-\text{sign}(\mathbf{w}^T \boldsymbol{x}'_j)$. If $|\mathbf{w}^T \boldsymbol{x}'_j| < 1$, the unlabeled example $\boldsymbol{x}'_j$ appears as two identical rows $\hat{X}_\jmath$ with both labels. Let $\jmath_l \in 1 \ldots l$ be the indices of the labeled examples in the active set, $\jmath'_1 \in 1 \ldots u$ be the indices of unlabeled examples with $|\mathbf{w}^T \boldsymbol{x}'_j| \geq 1$ and $\jmath'_2 \in 1 \ldots u$ be the indices of unlabeled examples with $|\mathbf{w}^T \boldsymbol{x}'_j| < 1$. Note that the index of every unlabeled example appears in one of these sets i.e, $\jmath'_1 \cup \jmath'_2 = 1 \ldots u$. Eqn. 6.4 may be re-written as:

$$\left[\gamma I + \frac{1}{l}\sum_{i \in \jmath_l} \boldsymbol{x}_i^T \boldsymbol{x}_i + \frac{\gamma'}{u}\sum_{j \in \jmath'_1} c_j \boldsymbol{x}'^T_j \boldsymbol{x}_j + \frac{\gamma'}{u}\sum_{j \in \jmath'_2} \boldsymbol{x}'^T_j \boldsymbol{x}_j\right] \ \bar{\mathbf{w}} =$$
$$\frac{1}{l}\sum_{i \in \jmath_l} y_i \boldsymbol{x}_i - \frac{\gamma'}{u}\sum_{j \in \jmath'_1} c_j \text{sign}(\mathbf{w}^T \boldsymbol{x}_j)\boldsymbol{x}_j + \frac{\gamma'}{u}\sum_{j \in \jmath'_2} (2p_j - 1)\boldsymbol{x}_j$$

where $c_j = p_j$ if $\text{sign}(\mathbf{w}^T \boldsymbol{x}'_j) = -1$ and $c_j = 1 - p_j$ if $\text{sign}(\mathbf{w}^T \boldsymbol{x}'_j) = 1$. Re-writing in matrix notation, we obtain an equivalent linear system that can be solved by CGLS:

$$\left[\gamma I + \tilde{X}^T \tilde{C} \tilde{X}\right] \ \bar{\mathbf{w}} = \tilde{X}^T \tilde{C} \tilde{Y} \tag{6.10}$$

where $\tilde{X} = [X_{\jmath l}^T \ \ X']$, $\tilde{C}$ is a diagonal matrix and $\tilde{Y}$ is the vector of effectively active

labels. These are defined by:

$$\begin{aligned}
\tilde{C}_{jj} = \frac{1}{l}\ ,\ \tilde{Y}_j = y_i \quad & j \in 1 \ldots |\jmath_l| \\
\tilde{C}_{(j+|\jmath_l|)(j+|\jmath_l|)} = \frac{\gamma' p_j}{u}\ ,\ \tilde{Y}_{j+|\jmath_l|} = 1 \ \text{ if } j \in 1 \ldots u\ , j \in \jmath'_1,\ \text{sign}(\mathbf{w}^T \boldsymbol{x}'_j) = -1 \\
\tilde{C}_{(j+|\jmath_l|)(j+|\jmath_l|)} = \frac{\gamma'(1-p_j)}{u}\ ,\ \tilde{Y}_{j+|\jmath_l|} = -1 \ \text{ if } \ j \in 1 \ldots u \ \ j \in \jmath'_1, \quad \text{sign}(\mathbf{w}^T \boldsymbol{x}'_j) = 1 \\
\tilde{C}_{(j+|\jmath_l|)(j+|\jmath_l|)} = \frac{\gamma'}{u}\ ,\ \tilde{Y}_{j+|\jmath_l|} = (2p_j - 1) \ \text{ if } \ j \in 1 \ldots u \ \ j \in \jmath'_2
\end{aligned} \tag{6.11}$$

Thus, CGLS needs to only operate on data matrices with one instance of each unlabeled example using a suitably modified cost matrix and label vector.

After the CGLS step, one needs to check the optimality conditions. The optimality conditions can be re-written as:

$$\begin{aligned}
\forall\ i \in \jmath_l \quad & y_i \bar{o}_i \leq 1 + \epsilon \\
\forall\ i \in \jmath_l^c \quad & y_i \bar{o}_i \geq 1 - \epsilon \\
\forall\ j \in \jmath'_1 \quad & |\bar{o}'_j| \geq 1 - \epsilon \\
\forall\ j \in \jmath'_2 \quad & |\bar{o}'_j| \leq 1 + \epsilon
\end{aligned}$$

For the subsequent line search step, we reassemble appropriate output and label vectors to call the line search routine outlined in section 6.2.2.

6.4.2 Optimizing $\mathbf{p}$

For the latter problem of optimizing $\mathbf{p}$ for a fixed $\mathbf{w}$, we construct the Lagrangian (see Section 3.5.1) and obtain:

$$p_j = \frac{1}{1 + e^{\frac{g_j - 2\nu}{T}}} \tag{6.12}$$

where $g_j = \gamma'[l_2(w^T x'_j) - l_2(-w^T x'_j)]$. Substituting this expression in the balance constraint in Eqn. 6.7, we get a one-dimensional non-linear equation in 2ν:

$$\frac{1}{u}\sum_{j=1}^{u} \frac{1}{1 + e^{\frac{g_i - 2\nu}{T}}} = r$$

The root is computed by using a hybrid combination of Newton-Raphson iterations and the bisection method together with a carefully set initial value.

6.4.3 Stopping Criteria

For a fixed T, the alternate minimization of $\mathbf{w}$ and $\mathbf{p}$ proceeds until some stopping criterion is satisfied. A natural criterion is the mean Kullback-Liebler divergence (relative entropy) between current values of p_i and the values, say q_i, at the end of last iteration. Thus the stopping criterion for fixed T is:

$$KL(\mathbf{p}, \mathbf{q}) = \sum_{j=1}^{u} p_j \ \log \ \frac{p_j}{q_j} + (1 - p_j) \ \log \ \frac{1 - p_j}{1 - q_j} < u\epsilon$$

A good value for ϵ is 10^{-6}. As T approaches 0, the variables p_j approach 0 or 1. The temperature may be decreased in the outer loop until the total entropy falls below a threshold, which we take to be $\epsilon = 10^{-6}$ as above, i.e.,

$$H(\mathbf{p}) = -\sum_{j=l}^{u} \big(p_j \ \log \ p_j + (1 - p_j) \ \log \ (1 - p_j)\big) < u\epsilon$$

The TSVM objective function,

$$\frac{\gamma}{2}\|\mathbf{w}\|^2 + \frac{1}{2l}\sum_{i=1}^{l} l_2(y_i\ (\mathbf{w}^T\boldsymbol{x}_i) + \frac{\gamma'}{2u}\sum_{j=1}^{u}\max\left[0, 1-|\mathbf{w}^T\boldsymbol{x}'_j|\right]^2$$

is monitored as the optimization proceeds. The weight vector corresponding to the minimum transductive cost in the optimization path is returned as the solution.

6.5 Empirical Study

Semi-supervised learning experiments were conducted to test these algorithms on four medium-scale datasets (aut-avn, real-sim, ccat and gcat) and three large scale (full-ccat,full-gcat,kdd99) datasets. These are listed in Table 6.1. All experiments were performed on Intel Xeon CPU 3GHz, 2GB RAM machines.

Software

For software implementation used for benchmarking in this section, we point the reader to the SVM_{lin} package available at

`http://www.cs.uchicago.edu/~vikass/svmlin.html`

Datasets

The aut-avn and real-sim binary classification datasets come from a collection of UseNet articles[7] from four discussion groups, for simulated auto racing, simulated aviation, real autos, and real aviation. The ccat and gcat datasets pose the problem of separating corporate and government related articles respectively; these are the top-level categories in the RCV1 training data set [68]. full-ccat and full-gcat are

7. Available at: http://www.cs.umass.edu/∼mccallum/data/sraa.tar.gz

the corresponding datasets containing all the 804414 training and test documents in the RCV1 corpus. These data sets create an interesting situation where semi-supervised learning is required to learn different low density separators respecting different classification tasks in the same input space. The kdd99 dataset is from the KDD 1999 competition task to build a network intrusion detector, a predictive model capable of distinguishing between "bad" connections, called intrusions or attacks, and "good" normal connections. This is a relatively low-dimensional dataset containing about 5 million examples.

Table 6.1: Two-class datasets. d : data dimensionality, $\bar{n}_0$: average sparsity, $l+u$: number of labeled and unlabeled examples, t : number of test examples, r : positive class ratio.

Dataset	d	$\bar{n}_0$	$l+u$	t	r
aut-avn	20707	51.32	35588	35587	0.65
real-sim	20958	51.32	36155	36154	0.31
ccat	47236	75.93	17332	5787	0.46
gcat	47236	75.93	17332	5787	0.30
full-ccat	47236	76.7	804414	-	0.47
full-gcat	47236	76.7	804414	-	0.30
kdd99	128	17.3	4898431	-	0.80

For the medium-scale datasets, the results below are averaged over 10 random stratified splits of training (labeled and unlabeled) and test sets. The detailed performance of SVM, DA and TSVM (single and maximum switching) is studied as a function of the amount of labeled data in the training set. For the large scale datasets full-ccat, full-gcat and kdd99, we are mainly interested in computation times; a transductive setting is used to study performance in predicting the labels of unlabeled data on single splits. On full-ccat and full-gcat , we train SVM, DA and TSVM with $l = 100, 1000$ labels; for kdd99 we experiment with $l = 1000$ labels.

Since the two classes are fairly well represented in these datasets, we report error

rates, but expect our conclusions to also hold for other performance measures such as F-measure. We use a default values of $\gamma = 0.001$, and $\gamma' = 1$ for all datasets except[8] for aut-avn and ccat where $\gamma' = 10$. In the outer loops of DA and TSVM, we reduced T and increased γ' by a factor of 1.5 starting from $T = 10$ and $\gamma' = 10^{-5}$.

Minimization of Objective Function

We first examine the effectiveness of DA, TSVM with single switching (S=1) and TSVM with maximum switching (S=u/2) in optimizing the objective function. These three procedures are labeled DA,TSVM(S=1) and TSVM(S=max) in Figure 6.1, where we report the minimum value of the objective function with respect to varying number of labels on aut-avn, real-sim, ccat and gcat.

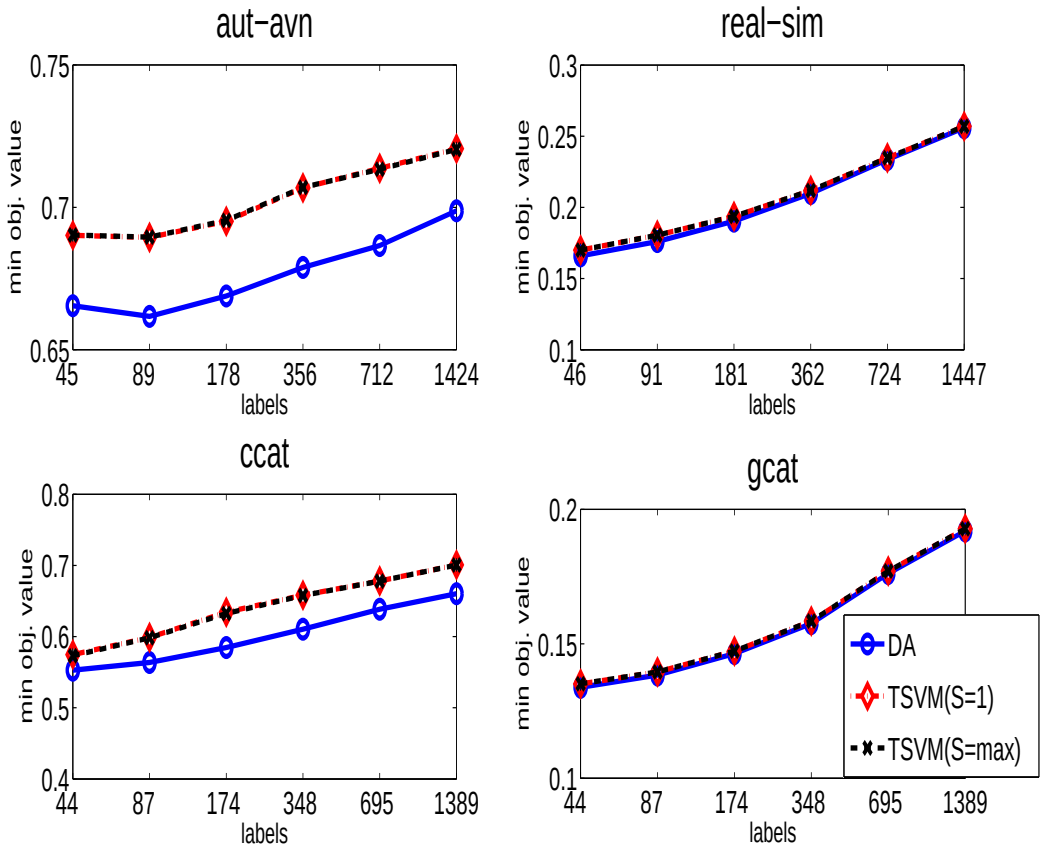
The following observations can be made.

1. Strikingly, *multiple switching leads to no loss of optimization as compared to single switching.* Indeed, the minimum objective value plots attained by single and multiple switching are virtually indistinguishable in Figure 6.1.

2. As compared to TSVM(S=1 or S=max), DA performs significantly better optimization on aut-avn and ccat; and slightly, but consistently better optimization on real-sim and gcat. These observations continue to hold for full-ccat and full-gcat as reported in Table 6.2 where we only performed TSVM experiments with S=max. Table 6.3 reports that DA gives a better minimum on the kdd99 also.

8. This produced better results for both TSVM and DA.

Figure 6.1: DA versus TSVM(S=1) versus TSVM(S=max): Minimum value of objective function achieved.



Generalization Performance

In Figure 6.2 we plot the mean error rate on the (unseen) test set with respect to varying number of labels on aut-avn, real-sim, ccat and gcat. In Figure 6.3, we superimpose these curves over the performance curves of a standard SVM which ignores unlabeled data. Tables 6.4, 6.5 report the corresponding results for full-ccat, full-gcat and kdd99.

The following observations can be made.

1. Comparing the performance of SVM against the semi-supervised algorithms in

Table 6.2: Comparison of minimum value of objective functions attained by TSVM (S=max) and DA on full-ccat and full-gcat.

	full-ccat		full-gcat	
l, u	TSVM	DA	TSVM	DA
100, 402107	0.1947	0.1940	0.1491	0.1491
100, 804314	0.1945	0.1940	0.1500	0.1499
1000, 402107	0.2590	0.2588	0.1902	0.1901
1000, 804314	0.2588	0.2586	0.1907	0.1906

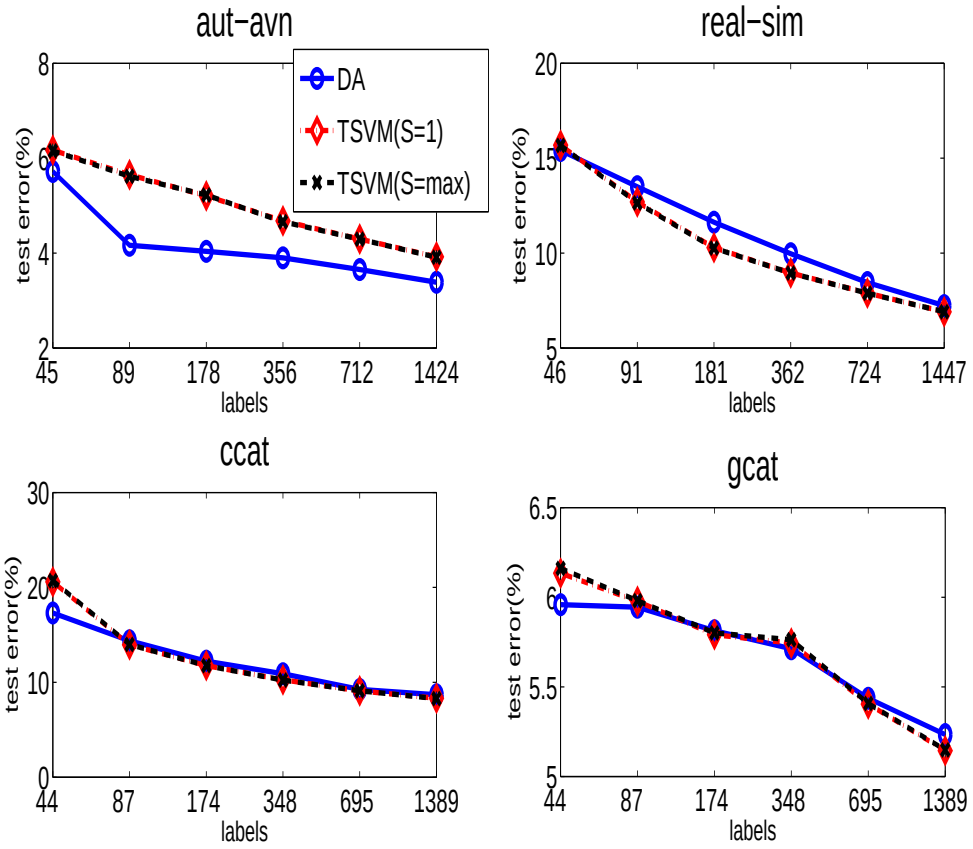
Table 6.3: Comparison of minimum value of objective functions attained by TSVM (S=max) and DA on kdd99

l, u	TSVM	DA
1000, 4897431	0.0066	0.0063

Figure 6.3, the benefit of unlabeled data for boosting generalization performance is evident on all datasets. This is true even for moderate number of labels, though it is particularly striking towards the lower end. On full-ccat and full-gcat too one can see significant gains with unlabeled data. On kdd99, SVM performance with 1000 labels is already very good.

2. In Figure 6.2, we see that on aut-avn, DA outperforms TSVM significantly. On real-sim, TSVM and DA perform very similar optimization of the transduction objective function (Figure 6.1), but appear to return very different solutions. The TSVM solution returns lower error rates, as compared to DA, on this dataset. On ccat, DA performed a much better optimization (Figure 6.1) but this does not translate into major error rate improvements. DA and TSVM are very closely matched on gcat. From Table 6.4 we see that TSVM and DA are competitive. On kdd99 (Table 6.5), DA gives the best results.

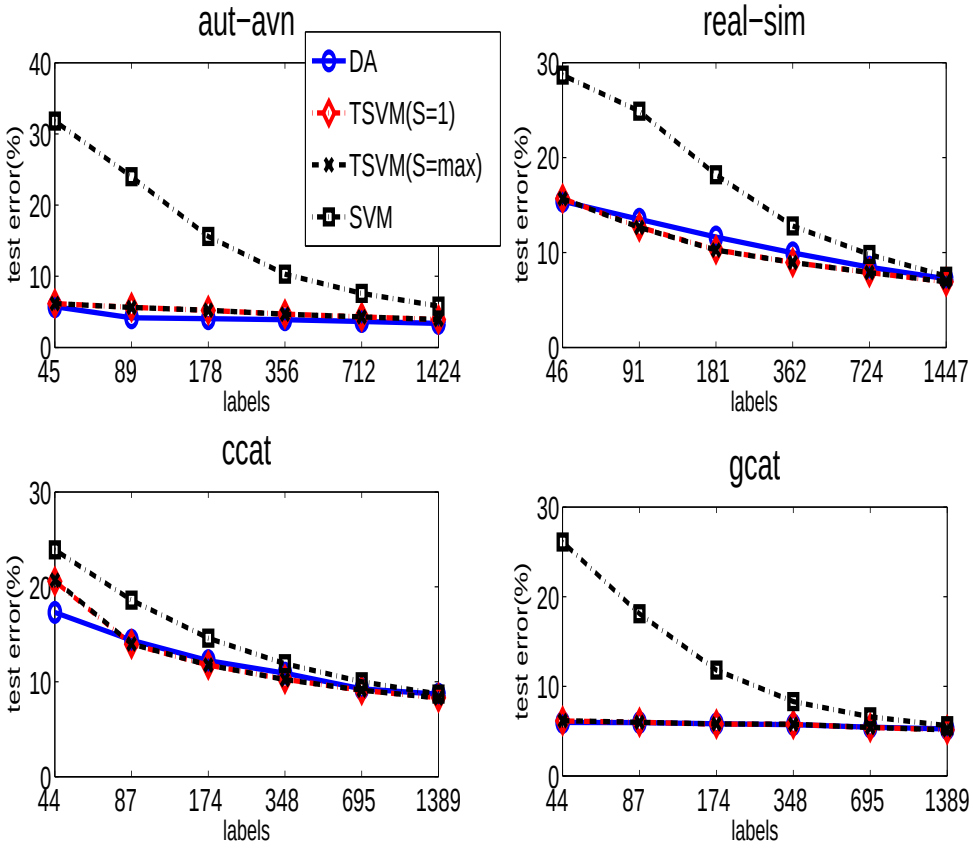
Figure 6.2: Error rates on Test set: DA versus TSVM (S=1) versus TSVM (S=max)



3. On all datasets we found that *maximum switching returned nearly identical performance as single switching*. Since it saves significant computation time, as we report in the following section, our study establishes multiple switching (in particular, maximum switching) as a valuable heuristic for training TSVMs.

4. These observations are also true for in-sample transductive performance for the medium scale datasets (detailed results not shown). Both TSVM and DA provide high quality extension to unseen test data.

Figure 6.3: Benefit of Unlabeled data



Computational Timings

In Figure 6.4 and Tables 6.6, 6.7 we report the computation time for our algorithms. The following observations can be made.

1. From Figure 6.4 we see that the single switch TSVM can be six to seven times slower than the maximum switching variant, particularly when labels are few. DA is significantly faster than single switch TSVM when labels are relatively few, but slower than TSVM with maximum switching.

2. In Table 6.6, we see that doubling the amount of data roughly doubles the

Table 6.4: TSVM (S=max) versus DA versus SVM: Error rates over unlabeled examples in full-ccat and full-gcat.

	full-ccat			full-gcat		
l, u	TSVM	DA	SVM	TSVM	DA	SVM
100, 402107	14.81	14.88	25.60	6.02	6.11	11.16
100, 804314	15.11	13.55	25.60	5.75	5.91	11.16
1000, 402107	11.45	11.52	12.31	5.67	5.74	7.18
1000, 804314	11.30	11.36	12.31	5.52	5.58	7.18

Table 6.5: DA versus TSVM (S = max) versus SVM: Error rates over unlabeled examples in kdd99.

l,u	TSVM	DA	SVM
1000, 4897431	0.48	0.22	0.29

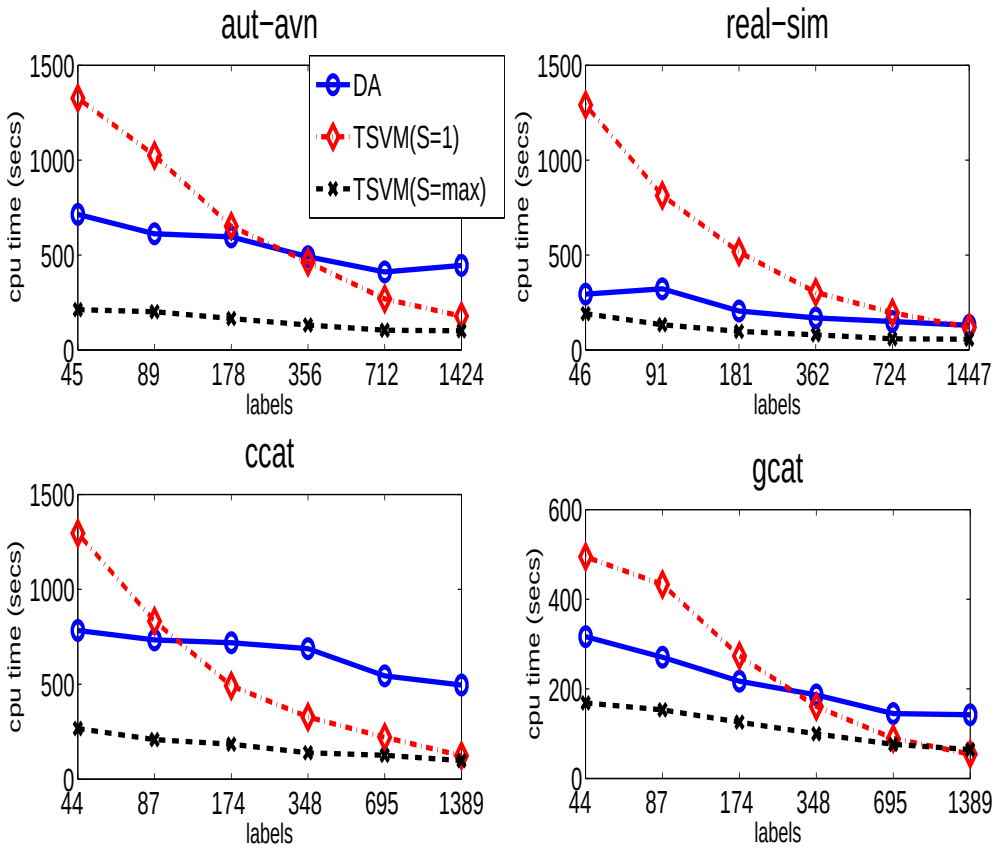
training time, empirically confirming the linear time complexity of our methods. The training time is also strongly dependent on the number of labels. On kdd99 (Table 6.7), the maximum-switch TSVM took around 15 minutes to process the 5 million examples whereas DA took 2 hours and 20 minutes.

3. On medium scale datasets, we also compared against SVM$^{\text{light}}$ which took on the order of several hours to days to train TSVM. We expect the multi-switch TSVM to also be highly effective when implemented in conjunction with the methods of [61].

Importance of Annealing

To confirm the necessity of an annealing component (tracking the minimizer while lowering T) in DA, we also compared it with an alternating $\mathbf{w}$,$\mathbf{p}$ optimization procedure where the temperature parameter is held fixed at $T = 0.1$ and $T = 0.001$. This study showed that annealing is important; it tends to provide higher quality solutions

Figure 6.4: Computation time with respect to number of labels for DA and Transductive l_2-SVM-MFN with single and multiple switches.



as compared to fixed temperature optimization. It is important to note that the gradual increase of γ' to the user-set value in TSVM is also a mechanism to avoid local optima. The non-convex part of the TSVM objective function is gradually increased to a desired value. In this sense, γ' simultaneously plays the role of an annealing parameter and also as a tradeoff parameter to enforce the cluster assumption. This dual role has the advantage that a suitable γ' can be chosen by monitoring performance on a validation set as the algorithm proceeds. In DA, however, we directly apply a framework for global optimization, and decouple annealing from the implementation

Table 6.6: Computation times (mins) for TSVM (S=max) and DA on full-ccat and full-gcat (804414 examples, 47236 features)

	full-ccat		full-gcat	
l, u	TSVM	DA	TSVM	DA
100, 402107	140	120	53	72
100, 804314	223	207	96	127
1000, 402107	32	57	20	42
1000, 804314	70	100	38	78

Table 6.7: Computation time (mins) for TSVM(S=max) and DA on kdd99 (4898431 examples, 127 features)

l, u	TSVM	DA
1000, 4897431	15	143

of the cluster assumption. As our experiments show, this can lead to significantly better solutions on many problems. On time-critical applications, one may tradeoff quality of optimization against time, by varying the annealing rate.

6.6 Conclusion

In this chapter we have proposed a family of primal SVM algorithms for large scale semi-supervised learning based on the finite Newton technique. Our methods significantly enhance the training speed of TSVM over existing methods such as SVM$^{\text{light}}$ and also include a new effective technique based on deterministic annealing. The new TSVM method with multiple switching is the fastest of all the algorithms considered, and also returns good generalization performance. The DA method is relatively slower but sometimes gives the best accuracy. Unlike the non-linear case (Chapter 3) applied to manifold like datasets, on text categorization domains local minima issues appear to be less severe. These algorithms can be very valuable in applied

scenarios where sparse classification problems arise frequently, labeled data is scarce and plenty of unlabeled data is easily available. Even in situations where a good number of labeled examples are available, utilizing unlabeled data to obtain a semi-supervised solution using these algorithms can be worthwhile. For one thing, the semi-supervised solutions never lag behind purely supervised solutions in terms of performance. The presence of a mix of labeled and unlabeled data can provide added benefits such as reducing performance variability and stabilizing the linear classifier weights. A large-scale implementation of Linear Laplacian RLS and SVMs can also be easily developed along the lines of Chapter 5 (i.e., by considering the linear case corresponding to Eqn. 5.6) building on the same core Newton primal optimization techniques covered in this chapter. These directions will be explored in future work.

REFERENCES

[1] M. Belkin. *Problems of Learning on Manifolds*. PhD thesis, The University of Chicago, 2003.

[2] M. Belkin, I. Matveeva, and P. Niyogi. Regularization and regression on large graphs. In *COLT*, 2004.

[3] M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. *COLT*, 2004.

[4] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in Neural Information Processing Systems*, 14:585–591, 2002.

[5] M. Belkin and P. Niyogi. Using manifold structure for partially labeled classification. *Advances in Neural Information Processing Systems*, 15:929–936, 2003.

[6] M. Belkin and P. Niyogi. Towards a theoretical foundation for Laplacian-based manifold methods. *Proc. of COLT*, 2005.

[7] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[8] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15(6):1373–1396, 2003.

[9] Y. Bengio, J.F. Paiement, P. Vincent, and O. Delalleau. Out-of-Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. *Advances in Neural Information Processing Systems*, 16, 2004.

[10] K. Bennett and A. Demiriz. Semi-supervised support vector machines. In *Advances in Neural Information processing systems 12*, 1998.

[11] K. Bennett and A. Demiriz. Semi-supervised support vector machines. *Advances in Neural Information Processing Systems*, 11:368–374, 1999.

[12] A. Berlinet and C. Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer, 2003.

[13] T. De Bie and N. Cristianini. Semi-supervised learning using semi-definite programming. In O. Chapelle, B. Schoëlkopf, and A. Zien, editors, *Semi-supervised learning*. MIT Press, 2006.

[14] G. Bilbro, W. E Snyder, and R. Mann. Mean-field approximation minimizes relative entropy. *Journal of Optical Society of America A*, 8, 1991.

[15] A. Bjork. *Numerical Methods for Least Squares Problems*. SIAM, 1996.

[16] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, 1987.

[17] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. *Proc. 18th International Conf. on Machine Learning*, pages 19–26, 2001.

[18] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, 1998.

[19] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.

[20] O. Bousquet, O. Chapelle, and M. Hein. Measure based regularization. *Advances in Neural Information Processing Systems*, 16, 2004.

[21] M. Brand. Continuous nonlinear dimensionality reduction by kernel eigenmaps. *Int. Joint Conf. Artif. Intel*, 2003.

[22] U. Brefeld and T. Scheffer. Co-em support vector learning. In *ICML*, 2004.

[23] O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19:1155–1178, 2007.

[24] O. Chapelle, M. Chi, and A. Zien. A continuation method for semi-supervised svms. In *International Conference on Machine Learning*, 2006.

[25] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-supervised Learning*. MIT Press, 2006.

[26] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, 2006.

[27] O. Chapelle, V. Sindhwani, and S. Keerthi. Branch and bound for semi-supervised support vector machines. In *Advances in Neural Information Processing Systems*, 2006.

[28] O. Chapelle, V. Sindhwani, and S. Keerthi. Branch and bound for semi-supervised support vector machines. In *Advances in Neural Information Processing Systems*, volume 20, 2007.

[29] O. Chapelle, J. Weston, and B. Scholkopf. Cluster kernels for semi-supervised learning. *Advances in Neural Information Processing Systems*, 15:585–592, 2003.

[30] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

[31] F.R.K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[32] I. Cohen and F. Cozman. Risks of semi-supervised learning. In O. Chapelle, A. Zien, and B. Schölkopf, editors, *Semi-Supervised Learning*, Cambridge, MA, USA, 2006. MIT Press.

[33] R.R. Coifman and S. Lafon. Diffusion maps. *Preprint, Jan*, 2005.

[34] RR Coifman, S. Lafon, AB Lee, M. Maggioni, B. Nadler, F. Warner, and SW Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the National Academy of Sciences*, 102(21):7426–7431, 2005.

[35] M. Collins and Y. Singer. Unsupervised models for named entity classification. In *EMNLP/VLC-99*, 1999.

[36] R. Collobert, F. Sinz, J. Weston, and L. Bottou. Large scale transductive SVMs. *Journal of Machine Learning Research*, 2006.

[37] A. Corduneanu and T. Jaakkola. On information regularization. *Proceedings of the Ninth Annual Conference on Uncertainty in Artificial Intelligence*, 2003.

[38] F. Cucker and S. Smale. On the mathematical foundations of learning. *AMERICAN MATHEMATICAL SOCIETY*, 39(1):1–49, 2002.

[39] S. Dasgupta, M. Littman, and D. McAllester. Pac generalization bounds for co-training. In *NIPS*, 2001.

[40] O. Delalleau, Y. Bengio, and N. Le Roux. Efficient non-parametric function induction in semi-supervised learning. *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT 2005)*, 2005.

[41] C. Ding, X. He, H. Zha, M. Gu, and H. Simon. Spectral min-max cut for graph partitioning and data clustering. *Proceedings of the First IEEE International Conference on Data Mining*, pages 107–114, 2001.

[42] M.P. Do Carmo. *Riemannian Geometry*. Birkhauser, 1992.

[43] D.L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 100(10):5591–5596, 2003.

[44] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. *Proceedings of the Seventh International Conference on Information and Knowledge Management*, 11:16, 1998.

[45] D.M. Dunlavy and D.P. O'Leary. Homotopy Optimization methods for Global Optimization. *Technical Report CS-TR-4773, Univ. of Maryland.*, 2005.

[46] B. Efron, T. Hastie, I.M. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

[47] T. Evgeniou, M. Pontil, and T. Poggio. Regularization Networks and Support Vector Machines. *Advances in Computational Mathematics*, 13(1):1–50, 2000.

[48] A. Frommer and P. Maas. Fast cg-based methods for tikhonov-phillips regularization. *SIAM Journal of Scientific Computing*, 20(5):1831–1850, 1999.

[49] G. Fung and O.L. Mangasarian. Semi-supervised support vector machines for unlabeled data classification. *Optimization Methods and Software*, 15(1):99–05, 2001.

[50] T. Gartner, Q. V. Le, S. Burton, A. J. Smola, and Vishwanathan S.V.N. Large scale multiclass transduction. In *NIPS*, 2005.

[51] A. Grigor'yan. Heat kernels on weighted manifolds and applications. *Cont. Math, 398, 93-191*, 2006.

[52] L. Hagen and AB Kahng. New spectral methods for ratio cut partitioning and clustering. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 11(9):1074–1085, 1992.

[53] J. Ham, D.D. Lee, and L.K. Saul. Semisupervised alignment of manifolds. *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence, Z. Ghahramani and R. Cowell, Eds*, 10:120–127, 2005.

[54] M. Hein, J.Y. Audibert, and U. von Luxburg. From graphs to manifolds-weak and strong pointwise consistency of graph Laplacians. *Proceedings of the 18th Conference on Learning Theory (COLT)*, pages 470–485, 2005.

[55] T. Hofmann and J. M. Buhmann. Pairwise Data Clustering by Deterministic Annealing. *IEEE TPAMI*, 1:1–14, 1997.

[56] V. de Silva J. B. Tenenbaum and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290, 2000.

[57] T. Joachims. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning*, 1999.

[58] T. Joachims. Transductive inference for text classification using support vector machines. *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209, 1999.

[59] T. Joachims. Transductive learning via spectral graph partitioning. In *International Conference on Machine Learning*, 2003.

[60] T. Joachims. Transductive learning via spectral graph partitioning. *Proceedings of the International Conference on Machine Learning*, pages 290–297, 2003.

[61] T. Joachims. Training linear svms in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2006.

[62] S. S. Keerthi and D. M. DeCoste. A modified finite Newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*, 6:341–361, 2005.

[63] C. Kemp, T.L. Griffiths, S. Stromsten, and J.B. Tenenbaum. Semi-supervised learning with trees. *Advances in Neural Information Processing Systems*, 16, 2004.

[64] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. In *Science*, pages 671–680, 1983.

[65] I. R. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *International Conference on Machine Learning*, 2003.

[66] J. Lafferty and L. Wasserman. Challenges in statistical machine learning. *Statistica Sinica*, 16(2):307–323, 2006.

[67] S. Lafon. *Diffusion Maps and Geometric Harmonics*. PhD thesis, Yale University, 2004.

[68] D.D. Lewis, Y. Yang, T. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 2004.

[69] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. In *http://www.cs.cmu.edu/∼mccallum/bow*, 1996.

[70] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *CIKM*, 2000.

[71] K. Nigam, A.K. Mccallum, S. Thrun, and T. Mitchell. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39(2):103–134, 2000.

[72] J. Nocedal and S. Wright, editors. *Numerical Optimzation*. Springer, 2000.

[73] M.R. Osborne, B. Presnell, and B.A. Turlach. On the lasso and its dual. *Journal of Computational and Graphical Statistics*, 9:319–337, 2000.

[74] T. Poggio and F. Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. In *Science*, pages 978–982, 1990.

[75] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *IEEE TPAMI*, 80:2210–2239, 1998.

[76] S.T. Roweis and L.K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290, 2000.

[77] B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[78] B. Scholkopf and A.J. Smola. *Learning with Kernels*. MIT Press Cambridge, Mass, 2002.

[79] M. Seeger. Learning with labeled and unlabeled data. *Inst. for Adaptive and Neural Computation, technical report, Univ. of Edinburgh*, 2001.

[80] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE. Trans. on Pattern Analysis and Machine Intelligence*, 22:888–905, 2000.

[81] V. Sindhwani. Kernel machines for semi-supervised learning. Master's thesis, The University of Chicago, 2004.

[82] V. Sindhwani, M. Belkin, and P. Niyogi. The geometric basis of semi-supervised learning. *Book Chapter in Semi-supervised Learning, Eds. Chapelle, O. and Schölkopf, B. and Zien, A.*, 2006.

[83] V. Sindhwani, W. Chu, and S.S. Keerthi. Semi-supervised Gaussian Process Classifiers . *IJCAI07, International Joint Conference on Artificial Intelligence*, 2007.

[84] V. Sindhwani, S. Keerthi, and O. Chapelle. Deterministic annealing for semi-supervised kernel machines. In *ICML*, 2006.

[85] V. Sindhwani and S. S. Keerthi. Large scale semi-supervised linear SVMs. In *SIGIR*, 2006.

[86] V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: From transductive to semi-supervised learning. In *ICML*, 2005.

[87] V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. *ICML05, 22nd International Conference on Machine Learning*, 2005.

[88] V. Sindhwani, P. Niyogi, and M. Belkin. A co-regularization approach to semi-supervised learning in multiple views. In *Workshop on Learning in Multiple Views, ICML*, 2005.

[89] A. Smola and I. R. Kondor. Kernels and regularization on graphs. In *COLT*, 2003.

[90] A. Smola and R. Kondor. Kernels and regularization on graphs. *Conference on Learning Theory, COLT/KW*, 2003.

[91] M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. *Advances in Neural Information Processing Systems*, 14:945–952, 2002.

[92] J.B. Tenenbaum, V. Silva, and J.C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319, 2000.

[93] R. Tibshirani. Regression selection and shrinkage via the lasso. Technical report, Stanford Univ., 1994.

[94] A.N. Tikhonov. Regularization of incorrectly posed problems. *Sov. Math. Dokl*, 4:1624–1627, 1963.

[95] I. Tsang and J. Kwok. Large scale sparsified manifold regularization. In *NIPS*, 2007.

[96] V. Vapnik and A. Sterin. On structural risk minimization or overall risk in a problem of pattern recognition. *Automation and Remote Control*, 10(3):1495–1503, 1977.

[97] V. N. Vapnik. *Statistical Learning Theory.* John Wiley & Sons, Inc., New York, 1998.

[98] V.N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[99] J.P. Vert and Y. Yamanishi. Supervised graph inference. *Advances in Neural Information Processing Systems*, 17:1433–1440, 2005.

[100] P. Vincent and Y. Bengio. Kernel matching pursuit. *Machine Learning*, 48:165–187, 2002.

[101] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 2007.

[102] U. von Luxburg, M. Belkin, and O. Bousquet. Consistency of spectral clustering. *Max Planck Institute for Biological Cybernetics Technical Report TR*, 134, 2004.

[103] G. Wahba. *Spline models for observational data*. Society for Industrial and Applied Mathematics Philadelphia, Pa, 1990.

[104] W. Wapnik and A. Tscherwonenkis. *Theorie der Zeichenerkennung*. Akademie Verlag, Berlin, 1979.

[105] L. Xu, J. Neufeld, B. Larson, and D. Schuurmans. Maximum margin clustering. In *Advances in Neural Information Processing Systems*, 2004.

[106] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*, 1995.

[107] T. Zhang, A. Popescul, and B. Dom. Linear prediction models with graph regularization for web-page categorization. In *KDD*, 2006.

[108] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In S. Thrun, L. Saul, and B. Schölkopf, editors,

Advances in Neural Information Processing Systems, volume 16, pages 321–328, Cambridge, MA, USA, 2004. MIT Press.

[109] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Scholkopf. Learning with local and global consistency. *Advances in Neural Information Processing Systems*, 16:321–328, 2004.

[110] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report 02-107, CMU-CALD, 2002.

[111] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. *Proceedings of the 20th International Conference on Machine Learning*, 2003.

[112] X. Zhu, J. Kandola, Z. Ghahramani, and J. Lafferty. Nonparametric Transforms of Graph Kernels for Semi-Supervised Learning. *Advances in neural information processing systems*, 17, 2005.