Large Scale Semi-supervised Linear SVMs

Vikas Sindhwani and Sathiya Keerthi

University of Chicago

YAHOO! RESEARCH

SIGIR 2006

Semi-supervised Learning (SSL)

Motivation

- Categorize x-billion documents into commercial/non-commercial.
- Traditional machine learning algorithms require labels.
- Labels are expensive/impossible to get.
- But tons of unlabeled data !

Setting

Linear SVMs (S^3VM) for large-scale problems – large number of examples and features – highly sparse – few labels and lots of unlabeled data.

Semi-supervised Learning (SSL)

Motivation

- Categorize x-billion documents into commercial/non-commercial.
- Traditional machine learning algorithms require labels.
- Labels are expensive/impossible to get.
- But tons of unlabeled data !

Setting

Linear SVMs (S^3VM) for large-scale problems – large number of examples and features – highly sparse – few labels and lots of unlabeled data.

Outline

Outline

Outline

Outline

1. Fast (fully supervised) Linear SVMs
2. The cluster assumption for SSL
3. Semi-supervised SVMs
   - An objective function to implement cluster assumption
   - A Scalable Label-switching Algorithm
   - The Problem of Non-convexity
   - A Deterministic Annealing (DA) approach
4. Empirical Studies
5. Extensions

Outline

1. Fast (fully supervised) Linear SVMs
2. The cluster assumption for SSL
3. Semi-supervised SVMs
   - An objective function to implement cluster assumption
   - A Scalable Label-switching Algorithm
   - The Problem of Non-convexity
   - A Deterministic Annealing (DA) approach
4. Empirical Studies
5. Extensions

Outline

1. Fast (fully supervised) Linear SVMs
2. The cluster assumption for SSL
3. Semi-supervised SVMs
   - An objective function to implement cluster assumption
   - A Scalable Label-switching Algorithm
   - The Problem of Non-convexity
   - A Deterministic Annealing (DA) approach
4. Empirical Studies
5. Extensions

Outline

1. Fast (fully supervised) Linear SVMs
2. The cluster assumption for SSL
3. Semi-supervised SVMs
   - An objective function to implement cluster assumption
   - A Scalable Label-switching Algorithm
   - The Problem of Non-convexity
   - A Deterministic Annealing (DA) approach
4. Empirical Studies
5. Extensions

Outline

Outline

Outline

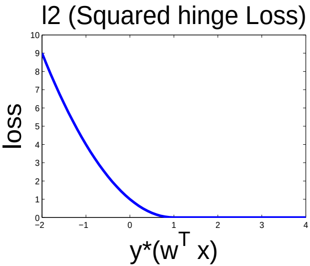
Fast Linear (l_2)-SVMs [Keerthi & Decoste,2005]

Given $\{x_i \in \mathbb{R}^d, y_i = \pm 1\}_{i=1}^{l}$, data matrix X ($l \times d$) is sparse.

Optimization

$$\min_{w \in \mathbb{R}^d} \frac{\lambda}{2}\|w\|^2 + \sum_{i=1}^{l} l_2\left(y_i, w^T x_i\right)$$

- continuously differentiable, whereas standard l_1 loss is not differentiable.
- Primal, unconstrained, direct w optimization, whereas LIBSVM/SVM-light are dual methods.
- Only $X \times \mathrm{vec}$ operations, whereas dual methods deal with dense gram matrix.



Fast Linear (l_2)-SVMs [Keerthi & Decoste,2005]

Algorithm

$$\min_{w \in \mathbb{R}^d} J(w) = \frac{\lambda}{2}\|w\|^2 + \sum_{i \in \mathcal{A}(w)} c_i \left(1 - y_i(w^T x_i)\right)^2$$

where $\mathcal{A}(w) = \{i : y_i(w^T x_i) < 1\}$

- Initialize w_0
- Iterate: k=0,1,2....
  - Regularized Least Squares:
    $\bar{w} = \min_w \frac{\lambda}{2}\|w\|^2 + \sum_{i \in \mathcal{A}(w_k)} c_i \left(1 - y_i(w^T x_i)\right)^2$
  - Set search direction $d = \bar{w} - w_k$
  - Line Search: Solve $\delta^\star = \min_\delta J(w_k + \delta d)$
  - Set new iterate: $w_{k+1} = w_k + \delta^\star(\bar{w} - w_k)$

Fast Linear (l_2)-SVMs [Keerthi & Decoste,2005]

Specialized Conjugate gradient (CGLS) to solve RLS

To get $\bar{w}$, Minimize:

$$\frac{1}{2}w^T[X^TCX + \lambda I]w - [X^TCY]w$$

where X: data matrix (rows are examples), C: diagonal cost matrix, Y label vector – only over $\mathcal{A}(w_k)$

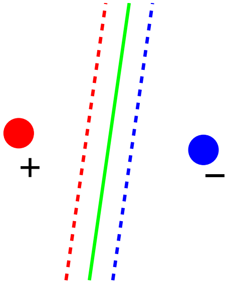
- $|\mathcal{A}(w_k)|$ *may be much smaller than* l.
- Use w_k as the initial seed. Seeding very effective.
- Only operations involving X are matrix-vector products of the form Xp and X^Tz – can be done fast.

Typical Behaviour: Reuters CCAT

Finite convergence guaranteed.
804414 examples, 47256 features: #CGLS iterations (10,15,8,2 ; 28,19) –> 7 iterations, Total 80 seconds [3GHz,2GB]

Outline

Cluster Assumption

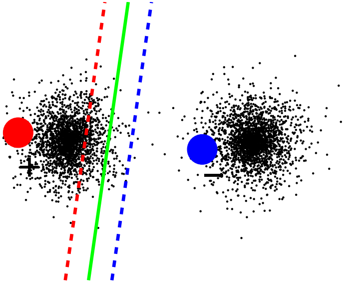


Assumptions

Two points in a cluster have same labels.

Design Principle: Drive the classification hyperplane away from the data – while respecting labels. Decisions should not change within a cluster.

Cluster Assumption

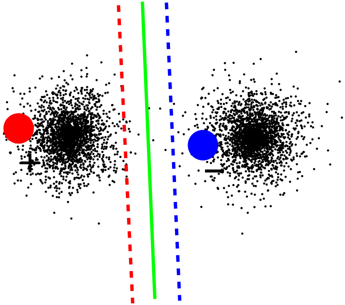


Assumptions

Two points in a cluster have same labels.

Design Principle: Drive the classification hyperplane away from the data – while respecting labels. Decisions should not change within a cluster.

Cluster Assumption

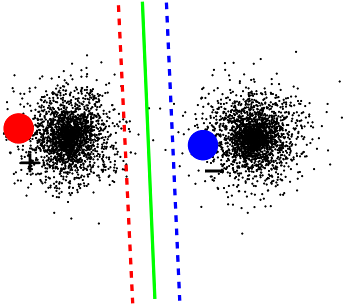


Assumptions

Two points in a cluster have same labels.

Design Principle: Drive the classification hyperplane away from the data – while respecting labels. Decisions should not change within a cluster.

Cluster Assumption



Assumptions

Two points in a cluster have same labels.
Design Principle: Drive the classification hyperplane away from the data – while respecting labels. Decisions should not change within a cluster.

Outline

Vapnik's idea

Given l labeled examples $\{x_i, y_i\}_{i=1}^{l}$, u unlabeled examples x_j'.

Train an SVM while optimizing unknown labels

Solve for weights w and unknown labels $\mathbf{y}' \in \{-1,+1\}^u$,

$$\min_{w,\mathbf{y}'} \overbrace{\frac{\lambda}{2}\|w\|^2 + \underbrace{\frac{1}{l}\sum_{i=1}^{l} l_2\left(y_i, w^T\mathbf{x}_i\right)}_{\text{labeled loss}}}^{\text{standard SVM}} + \underbrace{\frac{\lambda'}{u}\sum_{j=1}^{u} l_2\left(y_j', w^T\mathbf{x}_j'\right)}_{\text{unlabeled loss}}$$

$$\text{subject to:} \quad \frac{1}{u}\sum_{j=1}^{u} \max(0, y_j') = r \qquad \text{(positive class ratio)}$$

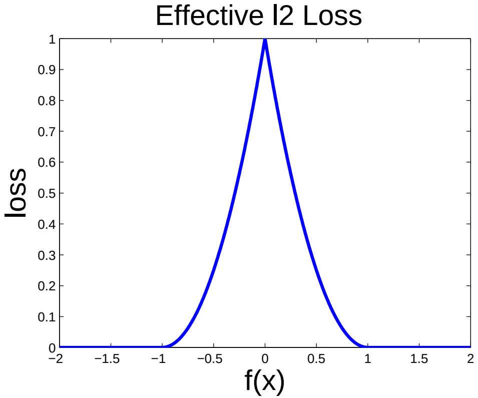
Equivalent Problem

Optimization Problem

$$\min_{w,\mathbf{y}'} J(w,\mathbf{y}') = \frac{\lambda}{2}\|w\|^2 + \frac{1}{l}\sum_{i=1}^{l} l_2\left(y_i, o_i\right) + \frac{\lambda'}{u}\sum_{j=1}^{u} l_2\left(y_j', o_j'\right)$$

$$\min_{w} J(w) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{l}\sum_{i=1}^{l} l_2\left(y_i, o_i\right) + \frac{\lambda'}{u}\sum_{j=1}^{u} \underbrace{\min\left[l_2\left(+, o_j'\right), l_2\left(-, o_j'\right)\right]}_{\text{effective loss } \; l_2'(o_j')}$$

Effective Loss Function Over Unlabeled Examples



- Non-convex
- Penalty if decision surface gets too close to unlabeled examples.

Fast TSVMs

SVM-Light Implementation

- Train an SVM on labeled data.
- Initialize $\mathbf{y}'$ by labeling unlabeled data (fraction r positive).
- Iterate:
  - Optimize w keeping $\mathbf{y}'$ fixed
    Train SVM using SVM-Light with $\mathbf{y}'$ as labels of unlabeled data.
  - Optimize $\mathbf{y}'$ keeping w fixed
    Switch a pair of labels so that objective function *strictly* decreases.

Fast TSVMs

Our Implementation

- Train an SVM on labeled data.
- Initialize $\mathbf{y}'$ by labeling unlabeled data (fraction r positive).
- Iterate:
  - Optimize w keeping $\mathbf{y}'$ fixed
    Train SVM using Fast l_2-SVM with $\mathbf{y}'$ as labels of unlabeled data. Seed previous w.
  - Optimize $\mathbf{y}'$ keeping w fixed
    Switch a pair of labels so that objective function *strictly* decreases.

Fast TSVMs

Our Implementation

- Train an SVM on labeled data.
- Initialize $\mathbf{y}'$ by labeling unlabeled data (fraction r positive).
- Iterate:
  - Optimize w keeping $\mathbf{y}'$ fixed
    Train SVM using Fast l_2-SVM with $\mathbf{y}'$ as labels of unlabeled data. Seed previous w.
  - Optimize $\mathbf{y}'$ keeping w fixed
    Switch multiple pairs of labels so that objective function *strictly* decreases.

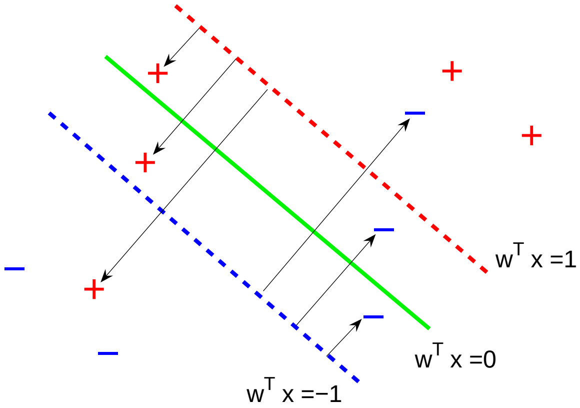
Fast TSVMs

Our Implementation

- Train an SVM on labeled data.
- Initialize $\mathbf{y}'$ by labeling unlabeled data (fraction r positive).
- Iterate:
  - Optimize w keeping $\mathbf{y}'$ fixed
    Train SVM using Fast l_2-SVM with $\mathbf{y}'$ as labels of unlabeled data. Seed previous w.
  - Optimize $\mathbf{y}'$ keeping w fixed
    Switch multiple pairs of labels so that objective function *strictly* decreases.

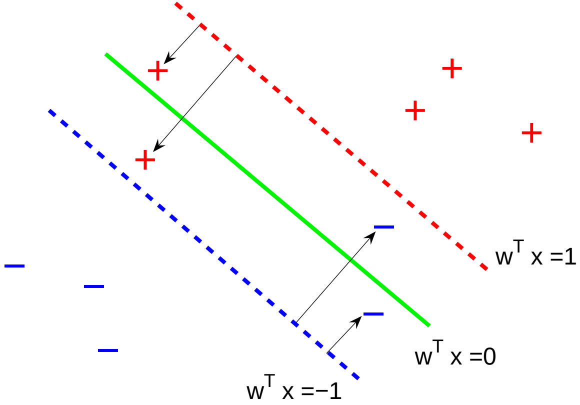
Question: Termination guaranteed – but how many switches and how efficient will it be to retrain so many times ?

Label Switching



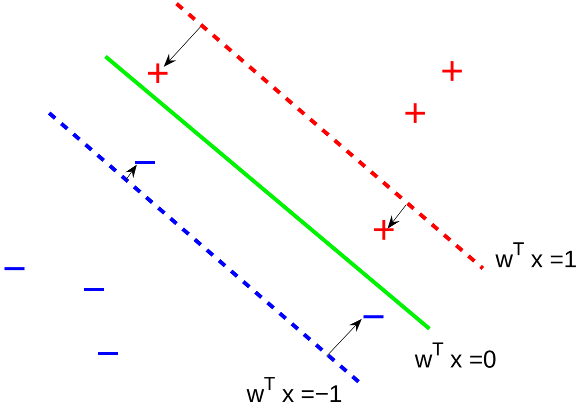
• Sort (currently) $+$ examples by margin error. • Sort (currently) $-$ examples by margin error. • Switch S pairs or until sum of margin errors falls below 2.

Label Switching



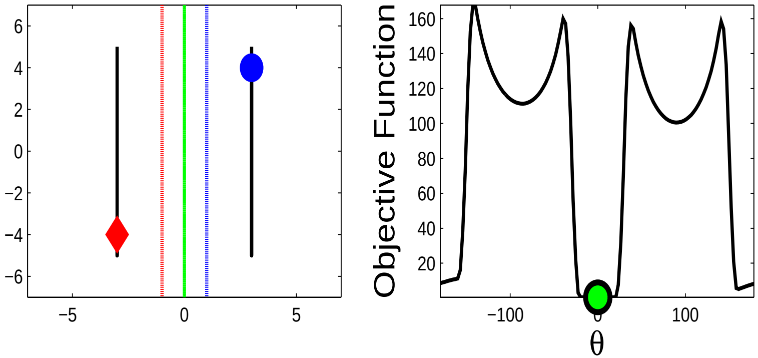
- Sort (currently) $+$ examples by margin error.
- Sort (currently) $-$ examples by margin error.
- Switch S pairs or until sum of margin errors falls below 2.

Label Switching

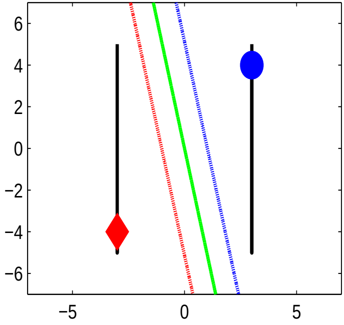


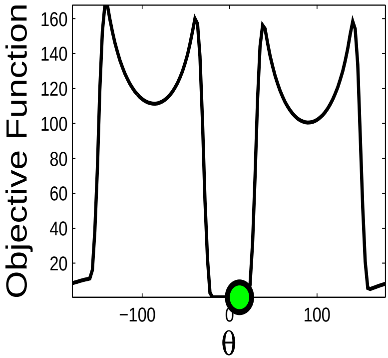
• Sort (currently) $+$ examples by margin error. • Sort (currently) $-$ examples by margin error. • Switch S pairs or until sum of margin errors falls below 2.

Non-convexity can hurt empirical performance

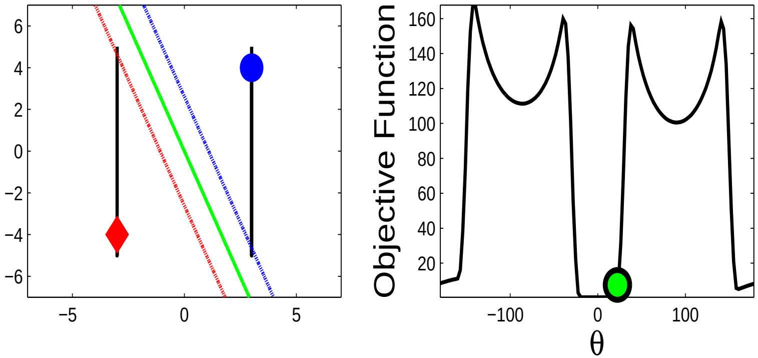


Non-convexity can hurt empirical performance

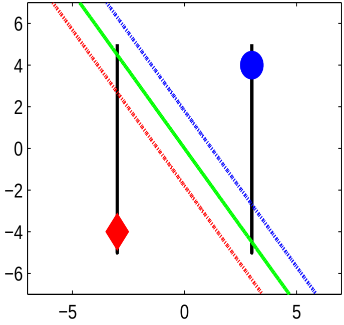


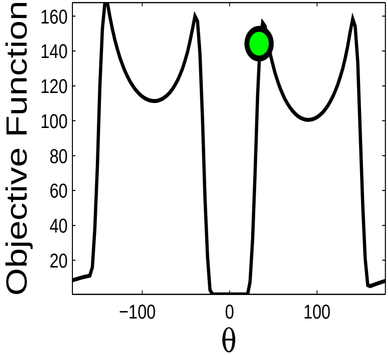


Non-convexity can hurt empirical performance

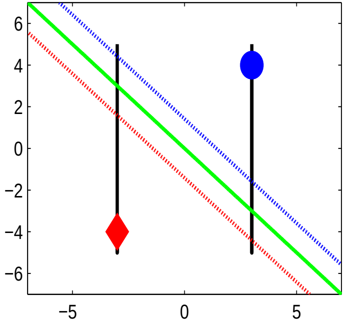


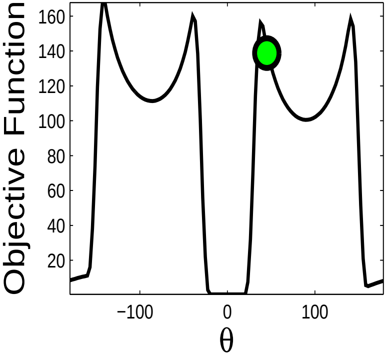
Non-convexity can hurt empirical performance



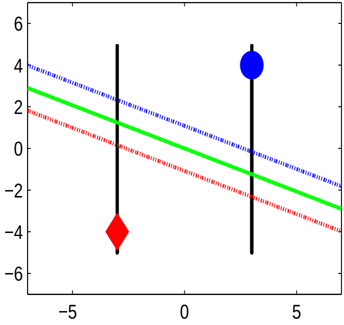


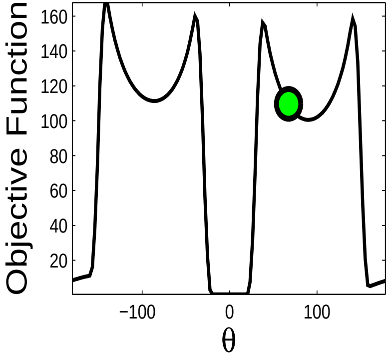
Non-convexity can hurt empirical performance



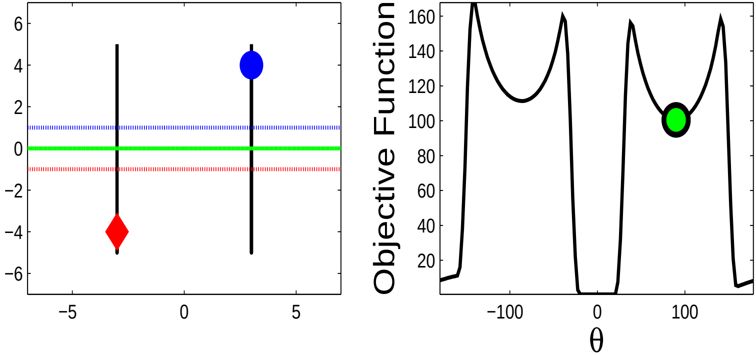


Non-convexity can hurt empirical performance

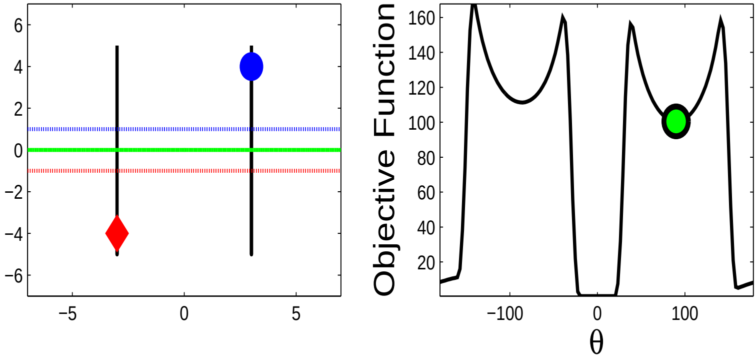




Non-convexity can hurt empirical performance



Non-convexity can hurt empirical performance



Error rates on COIL6: SVM 21.9, TSVM 21.2, ∇ TSVM 21.6

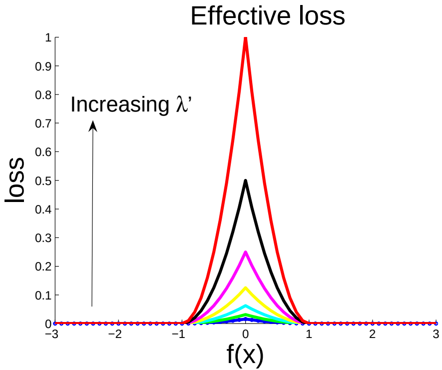
Handling Local Minima

- Start with an easy (unimodal) objective function and gradually increase non-convexity.
- Work with a family of objective functions; parameterically track minimizers.
- $J_{\mathcal{X}'}$ is insensitive to *outside* unlabeled data.

Handling Local Minima

- Start with an easy (unimodal) objective function and gradually increase non-convexity.
- Work with a family of objective functions; parameterically track minimizers.
- $J_{\lambda'}$ is insensitive to *outside* unlabeled data.

$$J_{\lambda'}(w, \mathbf{y}') = \frac{\lambda}{2}\|w\|^2 + \frac{1}{l}\sum_{i=1}^{l} l_2(y_i, o_i) + \frac{\lambda'}{u}\sum_{j=1}^{u} l_2\left(y_j', o_j'\right)$$

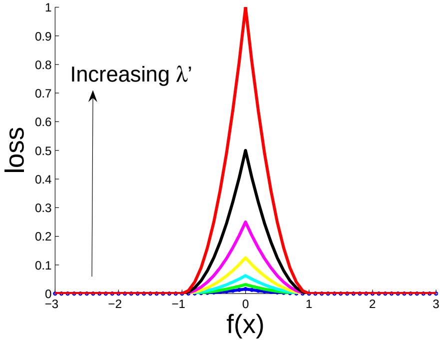


Handling Local Minima

$$J_{\lambda'}(w, \mathbf{y}') = \frac{\lambda}{2}\|w\|^2 + \frac{1}{l}\sum_{i=1}^{l} l_2(y_i, o_i) + \frac{\lambda'}{u}\sum_{j=1}^{u} l_2\left(y_j', o_j'\right)$$

- Start with an easy (unimodal) objective function and gradually increase non-convexity.
- Work with a family of objective functions; parameterically track minimizers.
- $J_{\lambda'}$ is insensitive to *outside* unlabeled data.

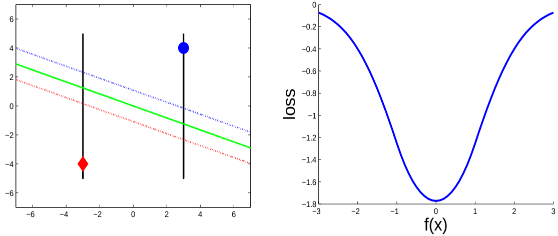
Effective loss



Deterministic Annealing: Intuition

Question

For the decision boundary to locally evolve in a desirable manner, what should the loss function look like ?



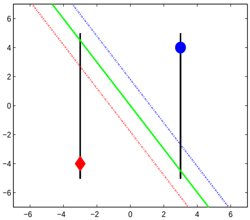
Key Idea

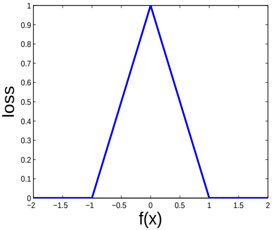
Deform the loss function (objective) as the optimization proceeds; use *outside* unlabeled data.

Deterministic Annealing: Intuition

Question

For the decision boundary to locally evolve in a desirable manner, what should the loss function look like ?





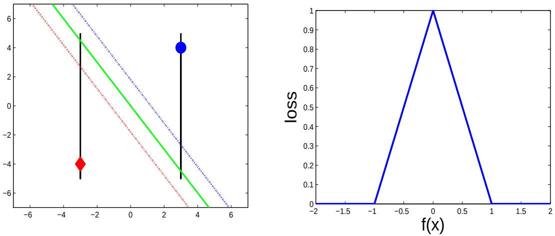
Key Idea

Deform the loss function (objective) as the optimization proceeds; use *outside* unlabeled data.

Deterministic Annealing: Intuition

Question

For the decision boundary to locally evolve in a desirable manner, what should the loss function look like ?



Key Idea

Deform the loss function (objective) as the optimization proceeds; use *outside* unlabeled data.

Deterministic Annealing for Semi-supervised SVMs

Another Equivalent Continuous Optimization Problem

"Relax" $\mathbf{y}'$ to $\mathbf{p} = (p_1 \ldots p_u)$ where p_j is *like* the prob that $y'_j = 1$.

$$J(w, \mathbf{p}) = E_{\mathbf{p}} J(w, \mathbf{y}') = \frac{\lambda}{2}\|w\|^2 + \frac{1}{l}\sum_{i=1}^{l} l_2\left(y_i, o_i\right)$$
$$+\frac{\lambda'}{u}\sum_{j=1}^{u}\left[p_j l_2\left(+, o'_j\right) + (1-p_j) l_2\left(-, o'_j\right)\right]$$

Family of Objective Functions: Avg Cost - T Entropy

$$J_T(w, \mathbf{p}) = E_{\mathbf{p}} J(w, \mathbf{y}') - \underbrace{T\ H(\mathbf{p})}_{-\frac{T}{u}\sum_{j=1}^{u}\left[p_j \ \log\ p_j + (1-p_j)\ \log\ (1-p_j)\right]}$$

Deterministic Annealing: Some Quick Comments

Smoothing Interpretation

At high T, spurious & shallow local min are smoothed away.

Deterministic Variant of Simulated Annealing (SA)

SA is a stochastic search technique based on setting up a Markov process whose steady state distribution minimizes $E_{\mathbf{p}}J - TH(\mathbf{p})$. Probabilistic guarantee for global optimum as $T \rightarrow 0$ *very slowly*.

Proven Heuristic

No guarantees, but has a strong record of empirical success.

Deterministic Annealing: Some Quick Comments

Smoothing Interpretation

At high T, spurious & shallow local min are smoothed away.

Deterministic Variant of Simulated Annealing (SA)

SA is a stochastic search technique based on setting up a Markov process whose steady state distribution minimizes $E_{\mathbf{p}}J - TH(\mathbf{p})$. Probabilistic guarantee for global optimum as $T \longrightarrow 0$ *very slowly*.

Proven Heuristic

No guarantees, but has a strong record of empirical success.

Deterministic Annealing: Some Quick Comments

Smoothing Interpretation

At high T, spurious & shallow local min are smoothed away.

Deterministic Variant of Simulated Annealing (SA)

SA is a stochastic search technique based on setting up a Markov process whose steady state distribution minimizes $E_{\mathbf{p}}J - TH(\mathbf{p})$. Probabilistic guarantee for global optimum as $T \longrightarrow 0$ *very slowly*.

Proven Heuristic

No guarantees, but has a strong record of empirical success.

Deterministic Annealing for Semi-supervised SVMs

Full Optimization problem at T

$$\min_{w,\mathbf{p}} J_T(w,\mathbf{p}) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{l}\sum_{i=1}^{l} l_2\left(y_i, o_i\right) + \frac{\lambda'}{u}\sum_{j=1}^{u}\left[p_j l_2\left(+, o_j'\right) + (1-p_j) l_2\left(-, o_j'\right)\right] + \frac{T}{u}\sum_{j=1}^{u}\left[p_j \ \log\ p_j + (1-p_j)\ \log\ p_j\right] \quad \text{s.t} \ \ (1/u)\sum_{j=1}^{u} p_j = r$$

Details

- **Deformation:** T controls non-convexity of $J_T(w,\mathbf{p})$. At $T=0$, reduces to the original non-convex objective function $J(w,\mathbf{p})$.
- **Optimization at** T $(w_T^\star, \mathbf{p}_T^\star) = \text{argmin}_{w,\mathbf{p}} J_T(w,\mathbf{p})$
- **Annealing:** Return: $w^\star = \lim_{T\to 0} w_T^\star$
- **Balance constraint:** $\frac{1}{u}\sum_{j=1}^{u} p_j = r$

Alternating Convex Optimization

At any T, optimize w keeping **p** fixed

- Use Fast l_2-SVMs – two copies of unlabeled data but due to linearity, can reformulate CGLS to work on one.

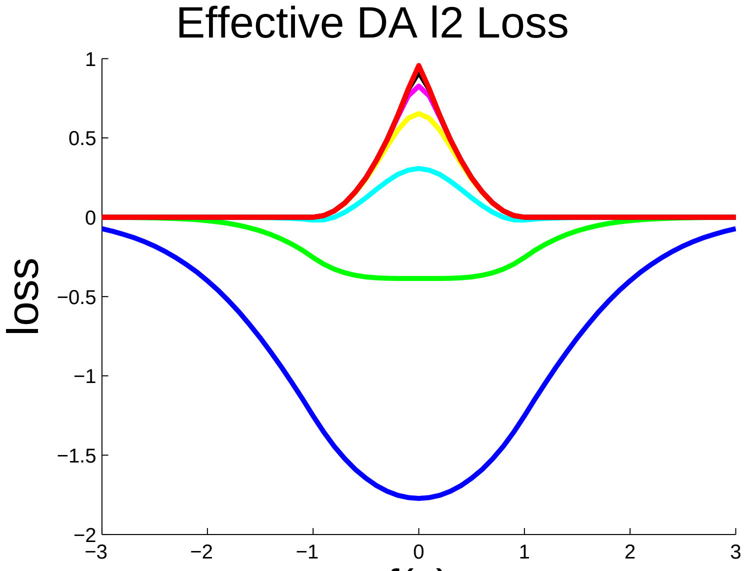
At any T, optimize **p** keeping w fixed

- $p_j^\star = \frac{1}{1+e^{\frac{g_j-\nu}{T}}} \quad g_j = \lambda' \left[l_2(+, o_j') - l_2(-, o_j') \right]$
- Obtain ν by solving $\frac{1}{u}\sum_{j=1}^{u} \frac{1}{1+e^{\frac{g_j-\nu}{T}}} = r$

Stopping Conditions

- At any T, alternate until $KL(\mathbf{p}_{new}|\mathbf{p}_{old}) < \epsilon$. Obtain $p_T^\star$.
- Reduce T, Seed old $p_T^\star$, until $H(\mathbf{p}_T^\star) < \epsilon$.

DA Effective Loss wrt T



Outline

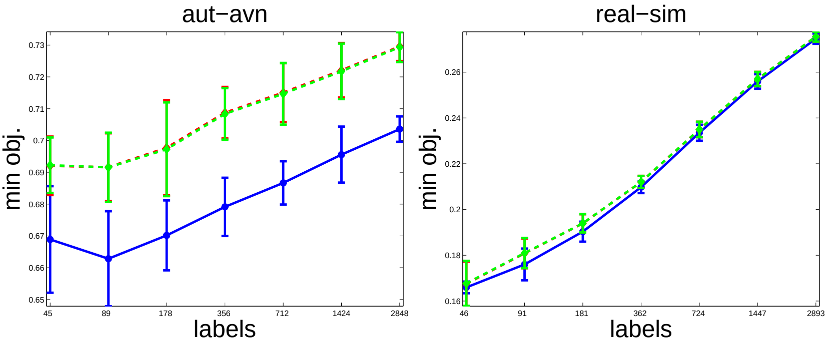
Experiments

Datasets

Dataset	#features	sparsity	#train	#test	r	Source
aut-avn	20707	51.32	35588	35587	0.65	Usenet
real-sim	20958	51.32	36155	36154	0.31	Usenet
ccat	47236	75.93	17332	5787	0.46	Reuters
gcat	47236	75.93	17332	5787	0.30	Reuters
33-36	59072	26.56	41346	41346	0.49	Yahoo!
pcmac	7511	54.58	1460	486	0.51	20NG
rcv1	47236	76.73	804414	-	0.18	Reuters

Quality of Optimization

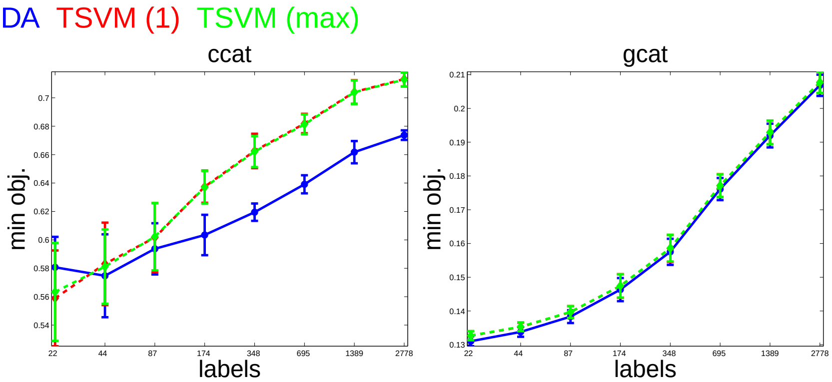
DA TSVM (1) TSVM (max)



Observations

- DA often gives significantly better minimizers [aut-avn,ccat,pcmac].
- Multiple switching TSVM no worse than single switching !

Quality of Optimization

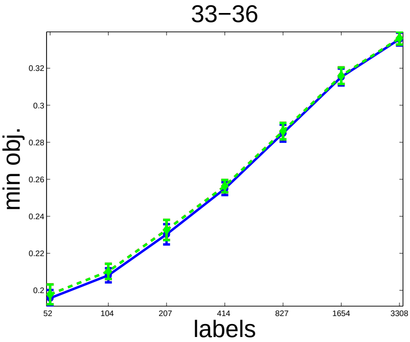


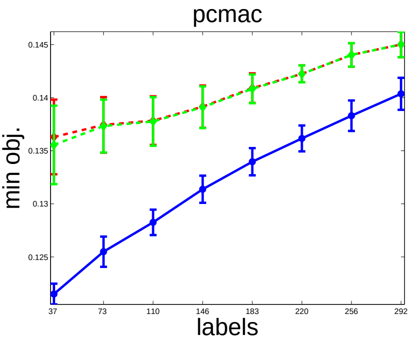
Observations

- DA often gives significantly better minimizers [aut-avn,ccat,pcmac].
- Multiple switching TSVM no worse than single switching !

Quality of Optimization

DA TSVM (1) TSVM (max)

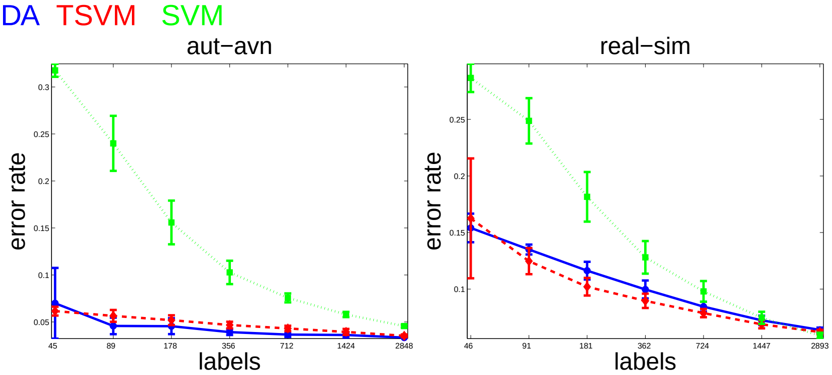




Observations

- DA often gives significantly better minimizers [aut-avn,ccat,pcmac].
- Multiple switching TSVM no worse than single switching !

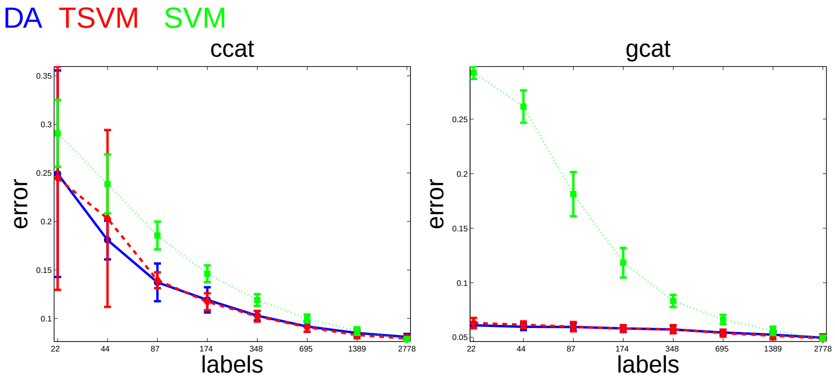
Generalization Performance



Observations

- Unlabeled data always very useful !
- TSVM's worse minimizers also generalize fairly well.
- Max-switching performs as well as single switching.

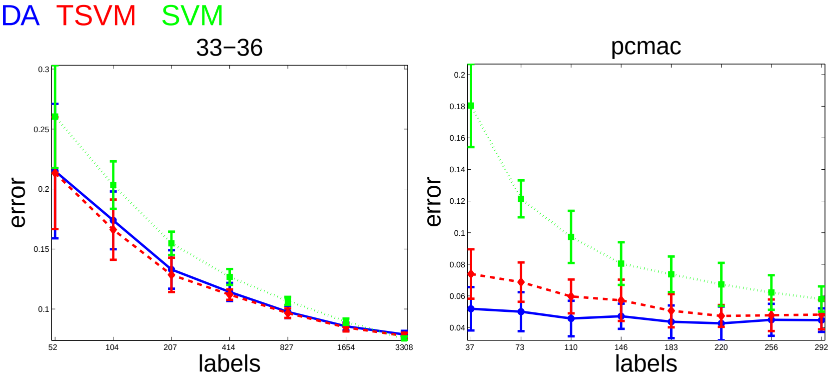
Generalization Performance



Observations

- Unlabeled data always very useful !
- TSVM's worse minimizers also generalize fairly well.
- Max-switching performs as well as single switching.

Generalization Performance



Observations

- Unlabeled data always very useful !
- TSVM's worse minimizers also generalize fairly well.
- Max-switching performs as well as single switching.

Speed comparison with SVM-Light

CPU Time [On a split with fewest labels]

Dataset	$\text{SVM}^{\text{light}}$	TSVM(1)	TSVM(max)	DA
aut-avn	$>$ 1 day	1.6hrs	7min	24min
real-sim	> 1 day	1.7hrs	6min	19min
ccat	4hrs	40min	6.5min	20min
gcat	> 1 day	20min	6min	3min
33-36	14hrs	2hrs	5min	7min
pcmac	167sec	4sec	2sec	12sec

- Massive speedups over SVM-Light
- TSVM(1) < DA < TSVM (max)
- Implemented in Matlab, C much faster. Easily parallelizable.

Larger-Scale Experiment

Reuters C15: 804414 examples, 47256 features (r=0.18)

PRBEP	l=100	l=1000
SVM	74.59	84.79
TSVM	77.73	86.60
DA	78.11	85.79
Obj.Value		
TSVM	0.094673	0.127172
DA	0.08073	0.12194
CPU Time		
TSVM (switches)	1hr 22min (27670)	40min (8506)
DA	2hr 8min	1hr 6min

Summary of Experimental Results

- Unlabeled data is very useful.
- DA better optimizer than TSVM.
- Both compete well in terms of generalization.
- Local minima issues less severe on text than on other domains.
- Massive speedups over SVM-Light. Max-switching TSVM is fastest, DA comparable.

Outline

Extensions

- Handing uncertain class ratios.
- Better annealing sequence for DA.
- Fast l_2-SVMs can be used to implement other SSL assumptions: Manifold (Laplacian SVM) and Co-training (Co-regularization).
- Software implementation available:
  SVM$_{\mathrm{lin}}$: Fast Linear SVM Solvers for Supervised and Semi-supervised Learning,
  http://www.cs.uchicago.edu/∼vikass/svmlin.html