SVM

May 2007

DOE-PI

Dianne P. O'Leary

Speeding the Training of Support Vector Machines and Solution of Quadratic Programs

Dianne P. O'Leary
Computer Science Dept. and
Institute for Advanced Computer Studies
University of Maryland

Jin Hyuk Jung
Computer Science Dept.

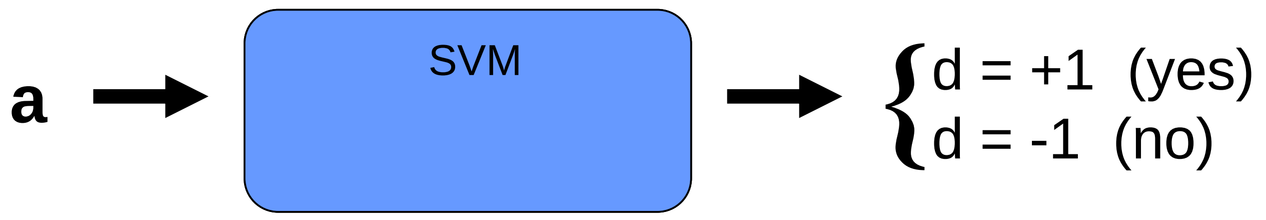
André Tits
Dept. of Electrical and Computer Engineering and
Institute for Systems Research

Work supported by the U.S. Department of Energy.

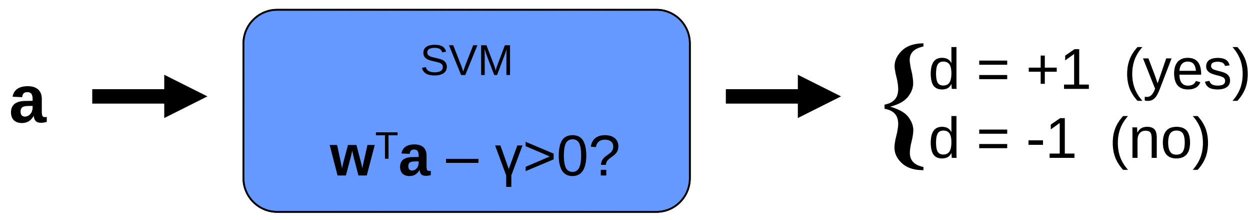
The Plan

- Introduction to SVMs
- Our algorithm
- Convergence results
- Examples

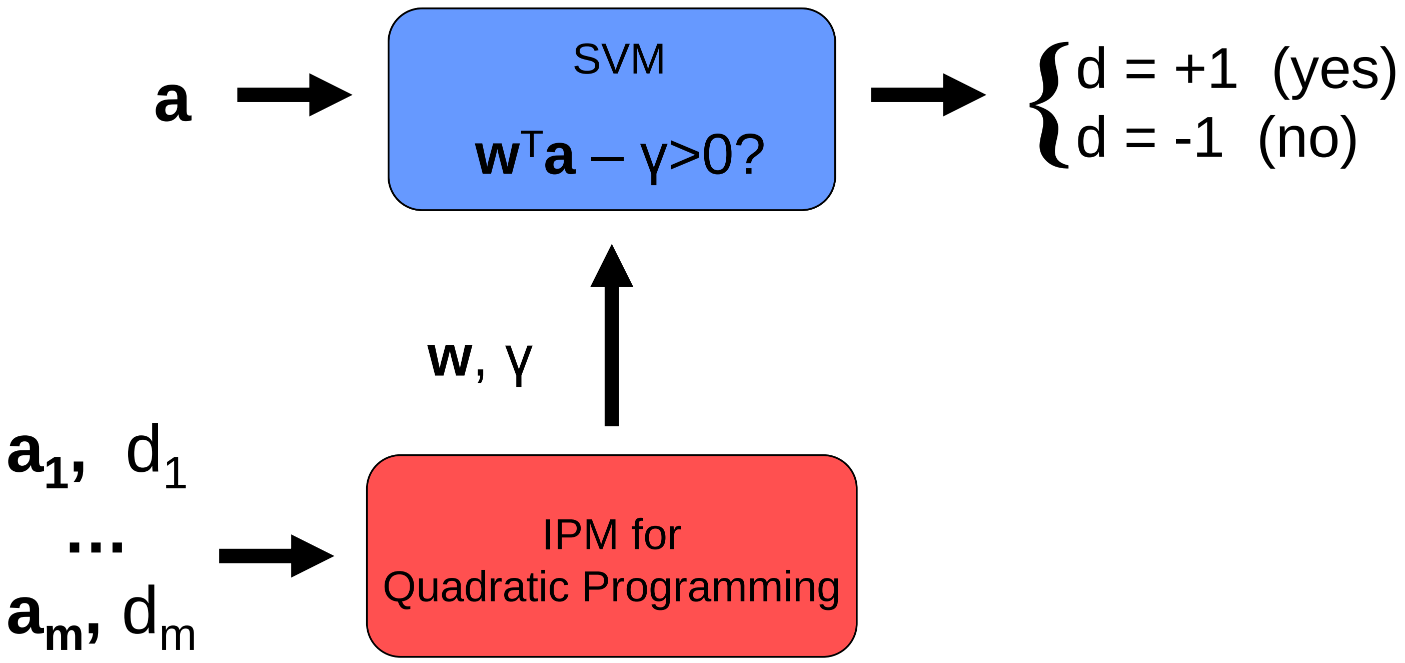
What is an SVM?



What's inside the SVM?



How is an SVM “trained”?



$\mathbf{a}$
SVM
$\mathbf{w}^T\mathbf{a} - \gamma > 0?$
$\begin{cases} d = +1 \text{ (yes)} \\ d = -1 \text{ (no)} \end{cases}$
$\mathbf{w}, \gamma$
$\mathbf{a_1}, d_1$
…
$\mathbf{a_m}, d_m$
IPM for Quadratic Programming

The problem: training an SVM

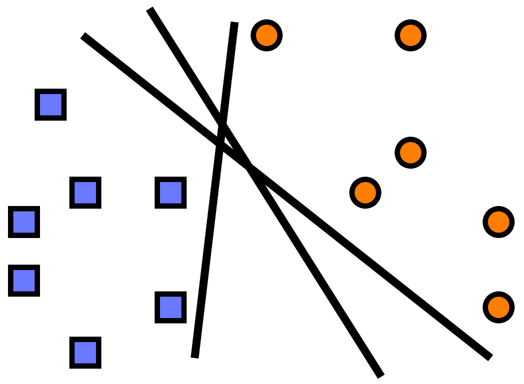
Given: A set of sample data points $\mathbf{a}_i$, in sample space $\mathcal{S}$, with labels $d_i = \pm 1$, $i = 1, \ldots, m$.

Find: A hyperplane $\{\mathbf{x} : \langle \mathbf{w}, \mathbf{x} \rangle - \gamma = 0\}$, such that

$$\operatorname{sign}(\langle \mathbf{w}, \mathbf{a}_i \rangle - \gamma) = d_i,$$

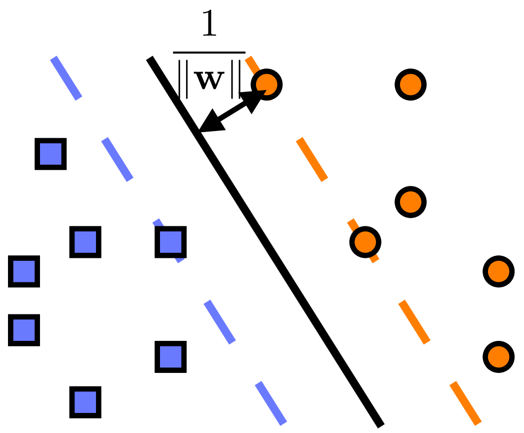
or, ideally,

$$d_i(\langle \mathbf{w}, \mathbf{a}_i \rangle - \gamma) \geq 1.$$



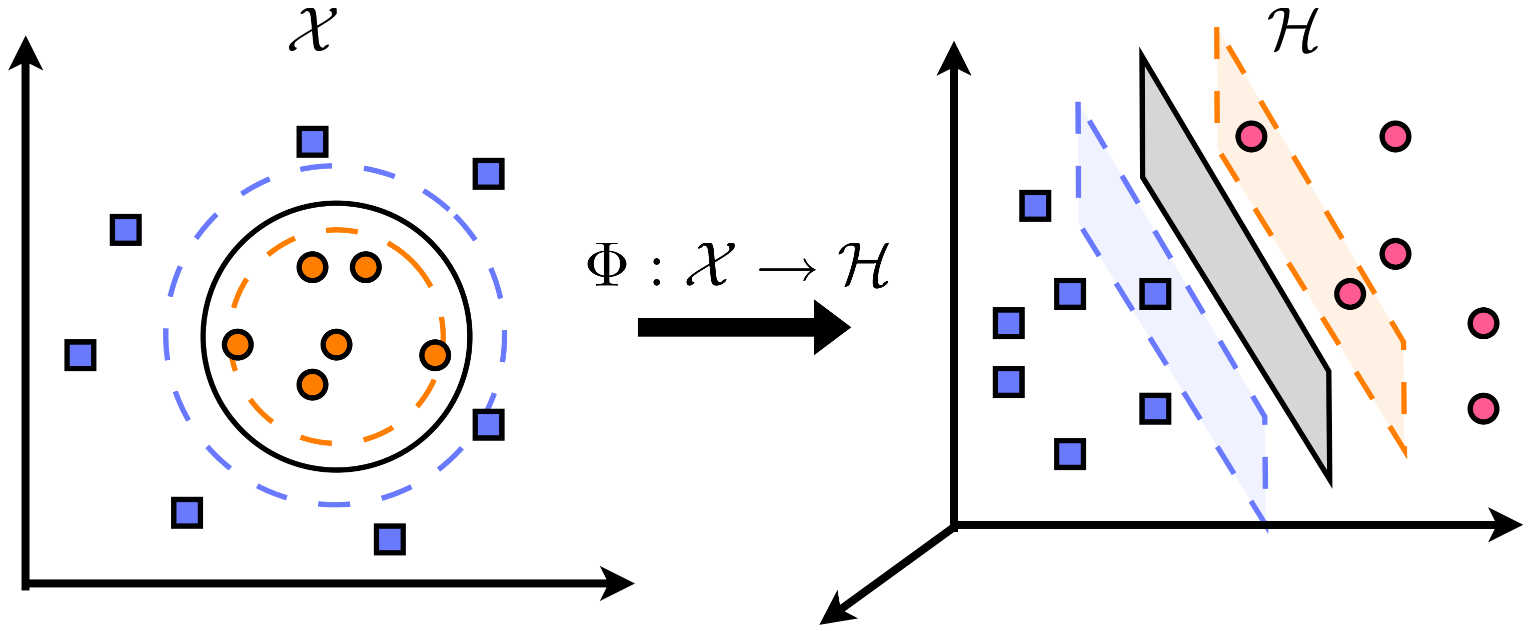
Which hyperplane is best?

We want to maximize the separation margin $1/\|\mathbf{w}\|$.



Generalization 1

We might map a more general separator to a hyperplane through some transformation Φ:



For simplicity, we will assume that this mapping has already been done.

Generalization 2

If there is no separating hyperplane, we might want to balance maximizing the separation margin with a penalty for misclassifying data by putting it on the wrong side of the hyperplane. This is the soft-margin SVM.

- We introduce slack variables $\mathbf{y} \geq \mathbf{0}$ and relax the constraints $d_i(\langle \mathbf{w}, \mathbf{a}_i \rangle - \gamma) \geq 1$ to

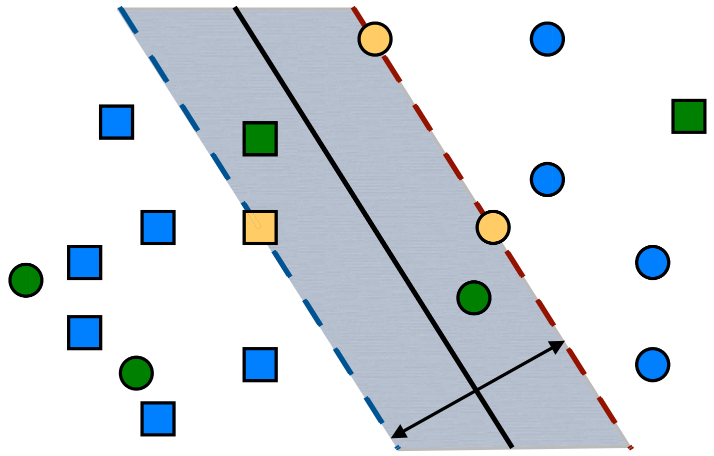
$$d_i(\langle \mathbf{w}, \mathbf{a}_i \rangle - \gamma) \geq 1 - y_i.$$

- Instead of minimizing $\|\mathbf{w}\|$, we solve

$$\min_{\mathbf{w},\gamma,\mathbf{y}} \frac{1}{2}\|\mathbf{w}\|_2^2 + \tau \mathbf{e}^T \mathbf{y}$$

for some $\tau > 0$, subject to the relaxed constraints.

Jargon



Classifier $\langle \mathbf{w}, \mathbf{x} \rangle - \gamma = 0$: black line
Boundary hyperplanes: dashed lines
$2 \times$ separation margin: length of arrow
Support vectors: On-Boundary (yellow) and Off-Boundary (green)
Non-SV: blue

Key point: The classifier is the same, regardless of the presence or absence of the blue points.

Summary

- The process of determining $\mathbf{w}$ and γ is called training the machine.
- After training, given a new data point $\mathbf{x}$, we simply calculate $\operatorname{sign}(\langle \mathbf{w}, \mathbf{x} \rangle - \gamma)$ to classify it as in either the positive or negative group.
- This process is thought of as a machine – called the support vector machine (SVM).
- We will see that training the machine involves solving a convex quadratic programming problem whose number of variables is the dimension n of the sample space and whose number of constraints is the number m of sample points – typically very large.

Primal and dual

Primal problem:

$$\begin{aligned} \min_{\mathbf{w},\gamma,\mathbf{y}} \ & \tfrac{1}{2}\|\mathbf{w}\|_2^2 + \tau\mathbf{e}^T\mathbf{y} \\ s.t. \quad & \mathbf{D}(\mathbf{A}\mathbf{w} - \mathbf{e}\gamma) + \mathbf{y} \geq \mathbf{e}, \\ & \mathbf{y} \geq \mathbf{0}, \end{aligned}$$

Dual problem:

$$\begin{aligned} \max_{\mathbf{v}} \ & -\tfrac{1}{2}\mathbf{v}^T\mathbf{H}\mathbf{v} + \mathbf{e}^T\mathbf{v} \\ s.t. \quad & \mathbf{e}^T\mathbf{D}\mathbf{v} = 0, \\ & \mathbf{0} \leq \mathbf{v} \leq \tau\mathbf{e}, \end{aligned}$$

where $\mathbf{H} = \mathbf{D}\mathbf{A}\mathbf{A}^T\mathbf{D} \in \mathbb{R}^{m\times m}$ is a symmetric and positive semidefinite matrix with

$$h_{ij} = d_i d_j \langle \mathbf{a}_i, \mathbf{a}_j \rangle.$$

Support vectors

Support vectors (SVs) are the patterns that contribute to defining the classifier.

They are associated with nonzero v_i.

	v_i	s_i	y_i
Support vector	$(0, \tau]$	0	$[0, \infty)$
On-Boundary SV	$(0, \tau)$	0	0
Off-Boundary SV	τ	0	$(0, \infty)$
Nonsupport vector	0	$(0, \infty)$	0

- v_i: dual variable (Lagrange multiplier for relaxed constraints).
- s_i: slack variable for primal inequality constraints.
- y_i: slack variable in relaxed constraints.

Solving the SVM problem

Apply standard optimization machinery:

- Write down the optimality conditions for the primal/dual formulation using the Lagrange multipliers. This is a system of nonlinear equations.

- Apply a (Mehotra-style predictor-corrector) interior point method (IPM) to solve the nonlinear equations by tracing out a path from a given starting point to the solution.

The optimality conditions

$$
\begin{aligned}
\mathbf{w} - \mathbf{A}^T\mathbf{D}\mathbf{v} &= \mathbf{0}, \\
\mathbf{d}^T\mathbf{v} &= 0 \\
\tau\mathbf{e} - \mathbf{v} - \mathbf{u} &= \mathbf{0}, \\
\mathbf{D}\mathbf{A}\mathbf{w} - \gamma\mathbf{d} + \mathbf{y} - \mathbf{e} - \mathbf{s} &= \mathbf{0}, \\
\mathbf{S}\mathbf{v} &= \sigma\mu\mathbf{e}, \\
\mathbf{Y}\mathbf{u} &= \sigma\mu\mathbf{e}, \\
\mathbf{s}, \mathbf{u}, \mathbf{v}, \mathbf{y} &\geq \mathbf{0}.
\end{aligned}
$$

for $\sigma = 0$.

Interior point method (IPM): Let $0 < \sigma < 1$ be a centering parameter and let

$$\mu = \frac{\mathbf{s}^T\mathbf{v} + \mathbf{y}^T\mathbf{u}}{2m}.$$

Follow the path traced as $\mu \to 0$, $\sigma = 1$.

A step of the IPM

At each step of the IPM, the next point on the path is computed using a variant of Newton's method applied to the system of equations.

This gives a linear system for the search direction:

$$\begin{bmatrix} \mathbf{I} & -\mathbf{A}^T\mathbf{D} & & & & \\ & \mathbf{d}^T & & & & \\ & -\mathbf{I} & -\mathbf{I} & & & \\ \mathbf{DA} & & & -\mathbf{I} & \mathbf{I} & -\mathbf{d} \\ & \mathbf{S} & & \mathbf{V} & & \\ & & \mathbf{Y} & & \mathbf{U} & \end{bmatrix} \begin{bmatrix} \Delta \mathrm{w} \\ \Delta \mathrm{v} \\ \Delta \mathrm{u} \\ \Delta \mathrm{s} \\ \Delta \mathrm{y} \\ \Delta \gamma \end{bmatrix} = \begin{bmatrix} -(\mathbf{w} - \mathbf{A}^T\mathbf{Dv}) \\ -\mathbf{d}^T\mathbf{v} \\ -(\tau\mathbf{e} - \mathbf{v} - \mathbf{u}) \\ -(\mathbf{DAw} - \gamma\mathbf{d} + \mathbf{y} - \mathbf{e} - \mathbf{s}) \\ -\mathbf{Sv} \\ -\mathbf{Yu} \end{bmatrix}$$

By block elimination, we can reduce this system to

$$\mathbf{M}\ \Delta \mathrm{w} = \text{some vector}$$

(sometimes called the normal equations).

The matrix for the normal equations

$$\mathbf{M}\ \triangle \mathbf{w} = \text{some vector}$$

$$\mathbf{M} = \mathbf{I} + \mathbf{A}^T \mathbf{D} \boldsymbol{\Omega}^{-1} \mathbf{D} \mathbf{A} - \frac{\bar{\mathbf{d}} \bar{\mathbf{d}}^T}{\mathbf{d}^T \boldsymbol{\Omega}^{-1} \mathbf{d}}.$$

Here, $\mathbf{D}$ and $\boldsymbol{\Omega}$ are diagonal, $\bar{\mathbf{d}} = \mathbf{A}^T \mathbf{D} \boldsymbol{\Omega}^{-1} \mathbf{d}$, and

$$\omega_i^{-1} = \frac{v_i u_i}{s_i v_i + y_i u_i}.$$

Given $\triangle \mathbf{w}$, we can easily find the other components of the direction.

Two variants of the IPM algorithm

- In affine scaling algorithms, $\sigma = 0$. Newton's method then aims to solve the optimality conditions for the optimization problem.

  Disadvantage: We might not be in the domain of fast convergence for Newton.

- In predictor-corrector algorithms, the affine scaling step is used to compute a corrector step with $\sigma > 0$. This step draws us back toward the path.

  Advantage: Superlinear convergence can be proved.

Examining $\mathbf{M} = \mathbf{I} + \mathbf{A}^T\mathbf{D}\mathbf{\Omega}^{-1}\mathbf{D}\mathbf{A} - \frac{\bar{\mathbf{d}}\bar{\mathbf{d}}^T}{\mathbf{d}^T\mathbf{\Omega}^{-1}\mathbf{d}}$.

Most expensive calculation: Forming **M**.

Our approach is to modify Newton's method by using an approximation to the middle and last terms. We'll discuss the middle one; the third is similar.

The middle term is

$$\mathbf{A}^T\mathbf{D}\mathbf{\Omega}^{-1}\mathbf{D}\mathbf{A} = \sum_{i=1}^{m} \frac{1}{\omega_i}\mathbf{a}_i\mathbf{a}_i^T,$$

$$\omega_i^{-1} = \frac{v_i u_i}{s_i v_i + y_i u_i}.$$

Note that ω_i^{-1} is well-behaved except for on-boundary support vectors, since in that case $s_i \to 0$ and $y_i \to 0$.

Our idea:

We only include terms corresponding to constraints we hypothesize are active at the optimal solution.

We could

- ignore the value of d_i.
- balance the number of positive and negative patterns included.

The number of terms is constrained to be

- at least q_L, which is the minimum of n and the number of ω^{-1} values greater than $\theta\sqrt{\mu}$, for some parameter θ.
- at most q_U, which is a fixed fraction of m.

Some related work

- Use of approximations to $\mathbf{M}$ in LP-IPMs dates back to Karmarkar (1984), and adaptive inclusion of terms was studied, for example, by Wang and O'Leary (2000).
- Osuna, Freund, and Girosi (1997) proposed solving a sequence of CQPs, building up patterns as new candidates for support vectors are identified.
- Joachims (1998) and Platt(1999) used variants related to Osuna et al.
- Ferris and Munson (2002) focused on efficient solution of normal equations.
- Gertz and Griffin (2005) used preconditioned cg, with a preconditioner based on neglecting terms in $\mathbf{M}$.

Convergence analysis for Predictor-Corrector Algorithm

All of our convergence results are derived from a general convergence analysis for adaptive constraint reduction for convex quadratic programming (since our problem is a special case).

Assumptions:

- $\mathcal{N}(\mathbf{A}_Q) \cap \mathcal{N}(H) = \{\mathbf{0}\}$ for all index sets Q with $|Q| \geq$ number of active constraints at the solution. (Automatic for SVM training.)
- We start at a point that satisfies the inequality constraints in the primal.
- The solution set is nonempty and bounded.
- The gradients of the active constraints at any primal feasible point are linearly independent.

Theorem: The algorithm converges to a solution point.

Additional Assumptions:

- The solution set contains just one point.
- Strict complementarity holds at the solution, meaning that $\mathbf{s}^* + \mathbf{v}^* > \mathbf{0}$ and $\mathbf{y}^* + \mathbf{u}^* > \mathbf{0}$.

Theorem: The convergence rate is q-quadratic.

Test problems

Provided by Josh Griffin (SANDIA)

Problem	n	Patterns $(+,-)$	SV $(+,-)$	In-bound SVs $(+,-)$
mushroom	276	(4208,3916)	(1146,1139)	(31,21)
isolet	617	(300,7497)	(74,112)	(74,112)
waveform	861	(1692,3308)	(633,638)	(110,118)
letter-recog	153	(789,19211)	(266,277)	(10,30)

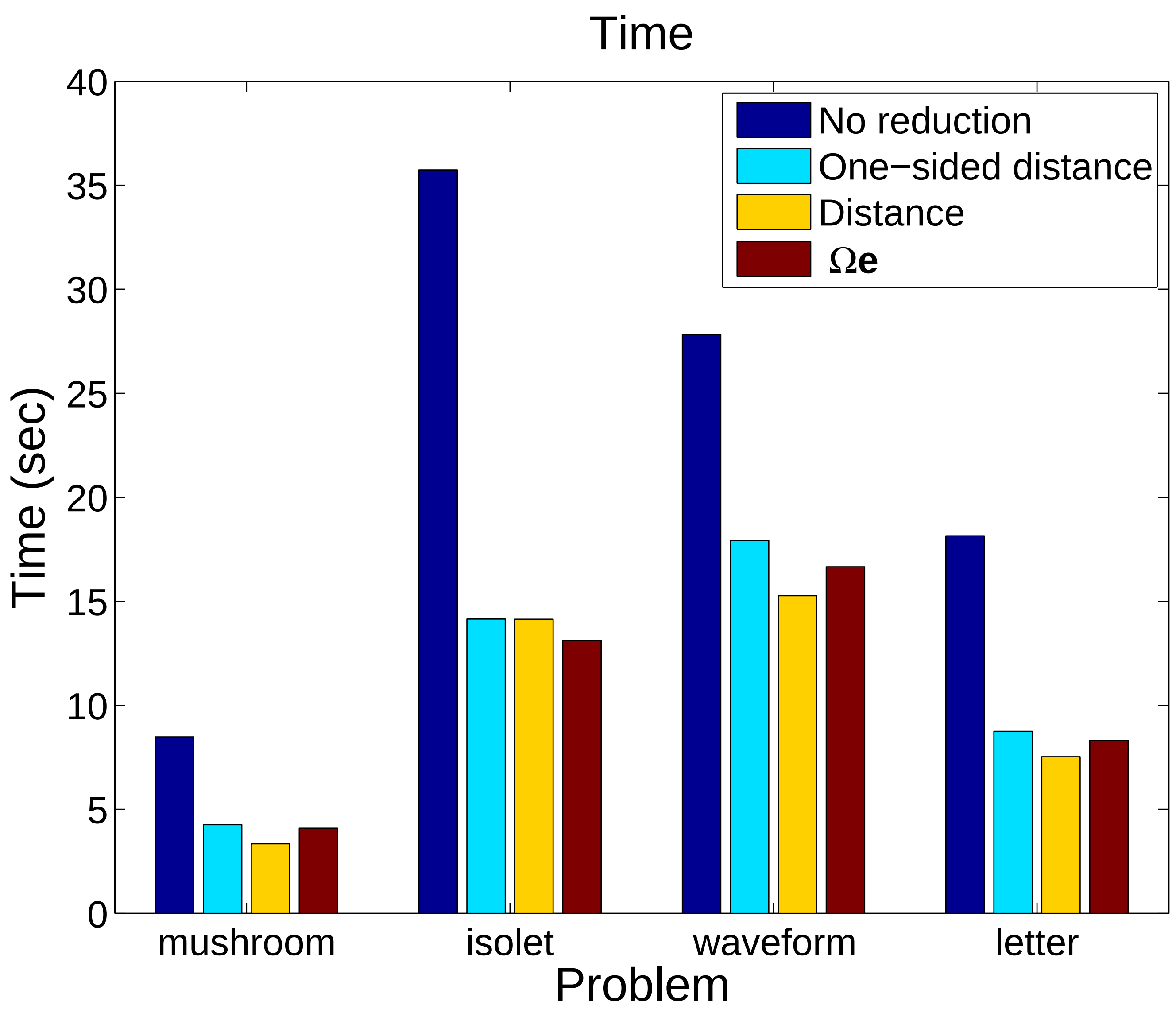
Algorithm variants

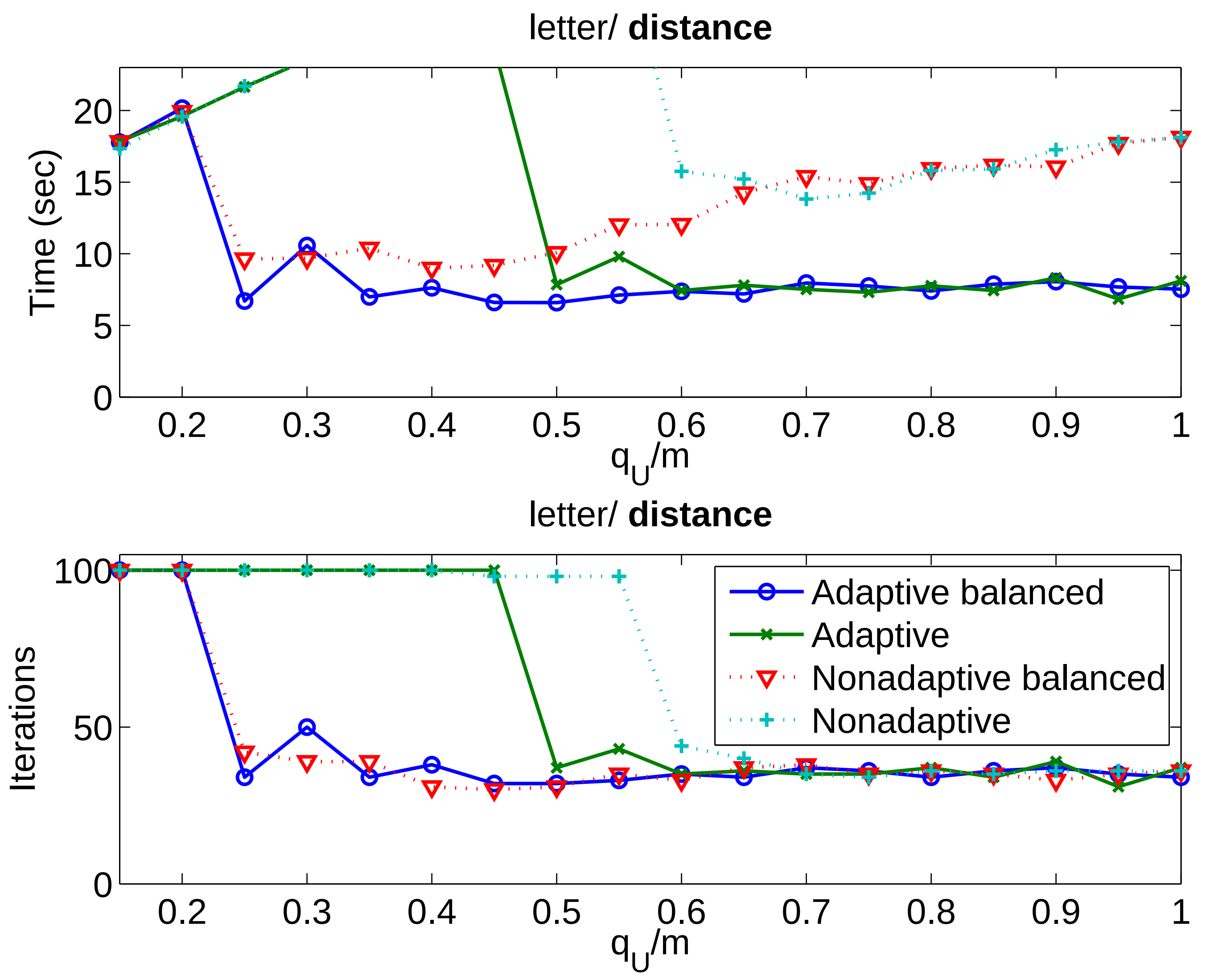
We used the reduced predictor-corrector algorithm, with constraints chosen in one of the following ways:

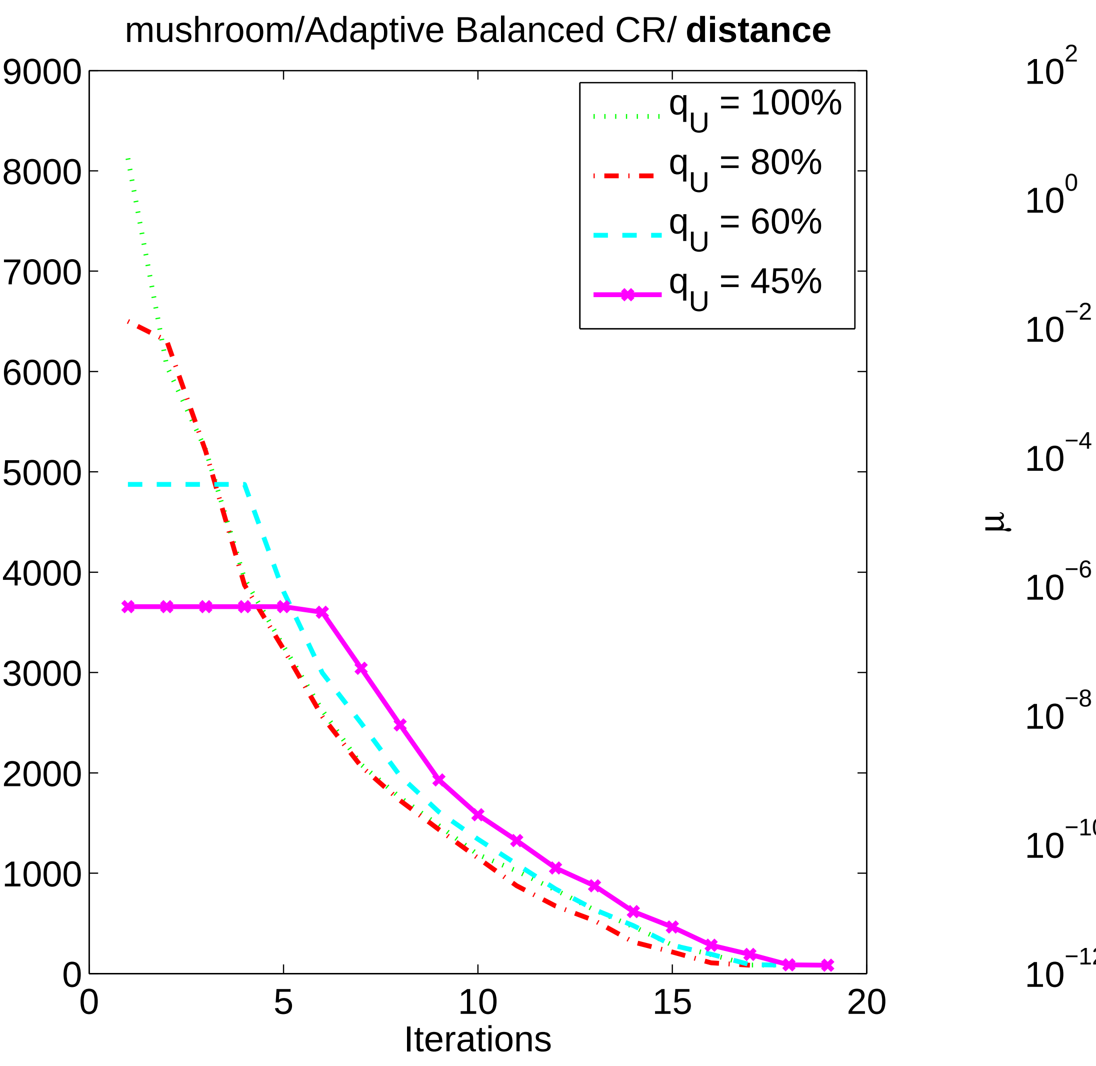
- One-sided distance: Use all points on the wrong side of the boundary planes and all points close to the boundary planes.
- Distance: Use all points close to the boundary planes.
- $\mathbf{\Omega}$: Use all points with large values of ω_i^{-1}.

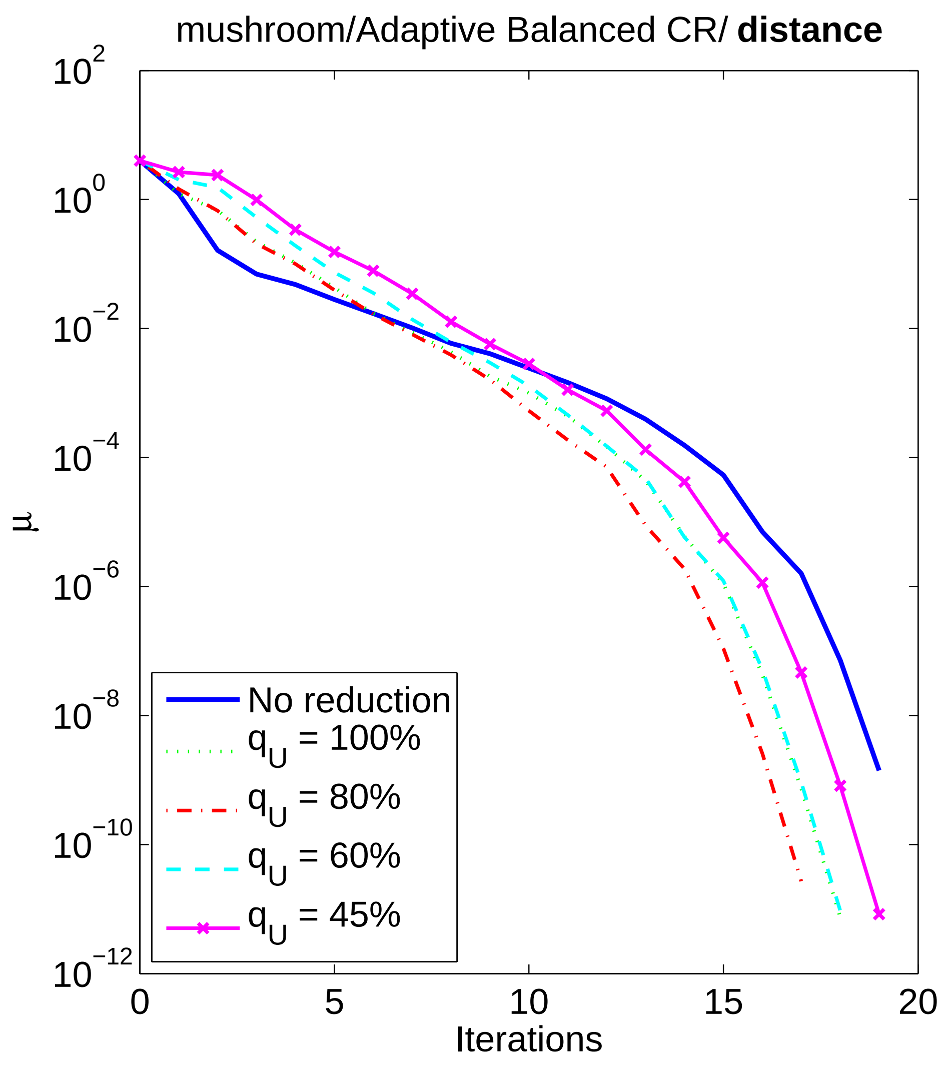
We used a balanced selection, choosing approximately equal numbers of positive and negative examples.

We compared with no reduction, using all of the terms in forming **M**.


Time
No reduction
One−sided distance
Distance
Ωe
Time (sec)
40
35
30
25
20
15
10
5
0
mushroom
isolet
waveform
letter
Problem


letter/ **distance**
Time (sec)
0
5
10
15
20
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
q_U/m
letter/ **distance**
Iterations
0
50
100
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
q_U/m
Adaptive balanced
Adaptive
Nonadaptive balanced
Nonadaptive


mushroom/Adaptive Balanced CR/ **distance**
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
5
10
15
20
Iterations
q_U = 100%
q_U = 80%
q_U = 60%
q_U = 45%


mushroom/Adaptive Balanced CR/ **distance**
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
μ
0
5
10
15
20
Iterations
No reduction
q_U = 100%
q_U = 80%
q_U = 60%
q_U = 45%

Comparison with other software

Problem	Type	LibSVM	SVMLight	Matlab	Ours
mushroom	Polynomial	5.8	52.2	1280.7	
mushroom	Mapping(Linear)	30.7	60.2	710.1	4.2
isolet	Linear	6.5	30.8	323.9	20.1
waveform	Polynomial	2.9	23.5	8404.1	
waveform	Mapping(Linear)	33.0	85.8	1361.8	16.2
letter	Polynomial	2.8	55.8	2831.2	
letter	Mapping(Linear)	11.6	45.9	287.4	13.5

- LibSVM, by Chih-Chung Chang and Chih-Jen Lin, uses a variant of SMO (by Platt), implemented in C
- SVMLight, by Joachims, implemented in C
- Matlab's program is a variant of SMO.
- Our program is implemented in Matlab, so a speed-up may be possible if converted to C.

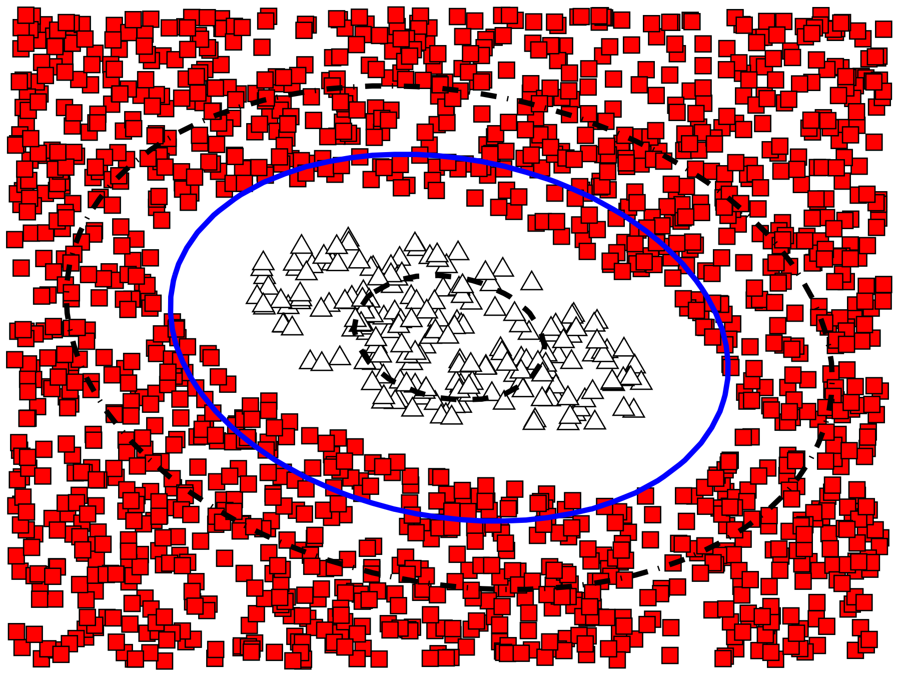
How our algorithm works

To visualize the iteration, we constructed a toy problem with

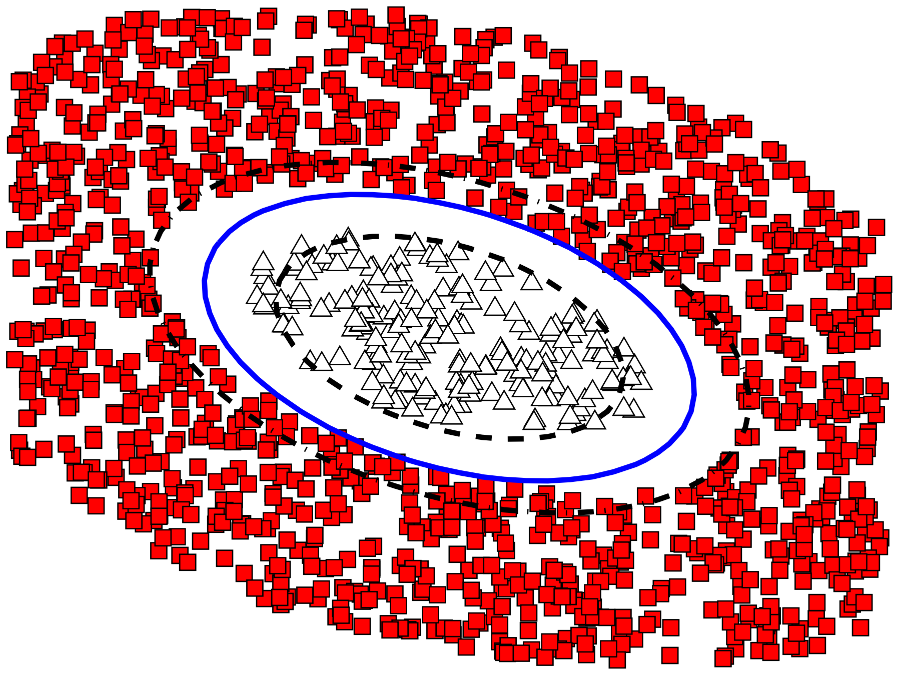
- $n = 2$,
- a mapping Φ corresponding to an ellipsoidal separator.

We now show snapshots of the patterns that contribute to $\mathbf{M}$ as the IPM iteration proceeds.

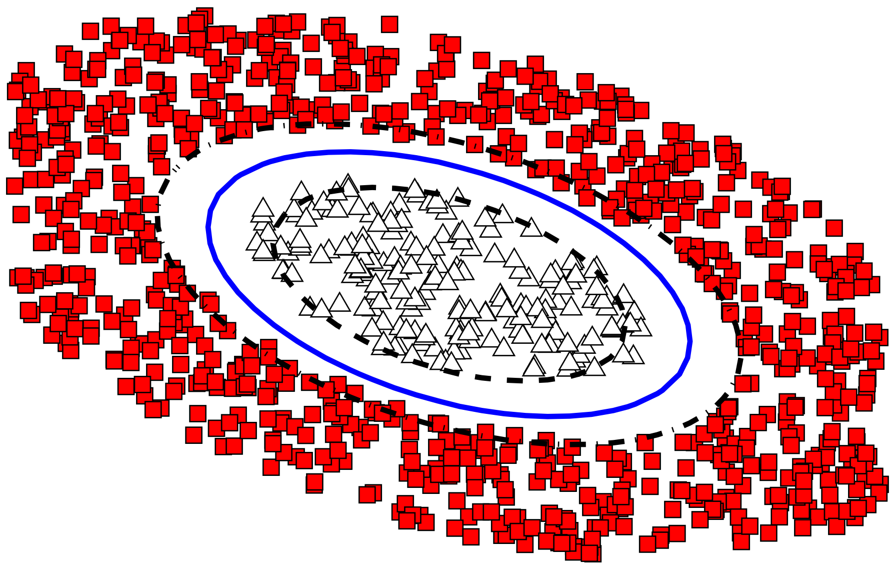
Iteration: 2, # of obs: 1727



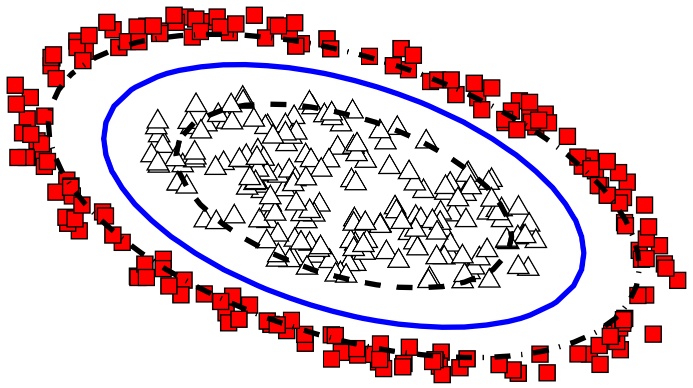
Iteration: 5, # of obs: 1440



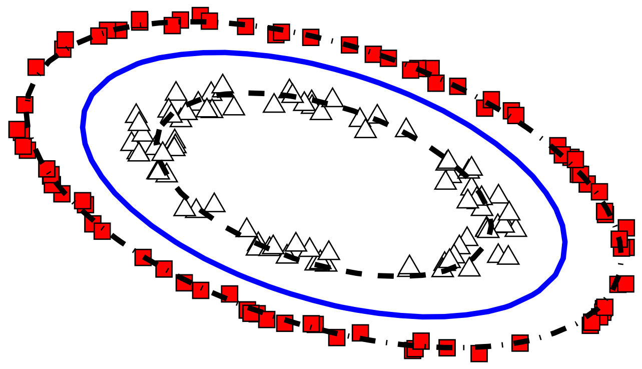
Iteration: 8, # of obs: 1026



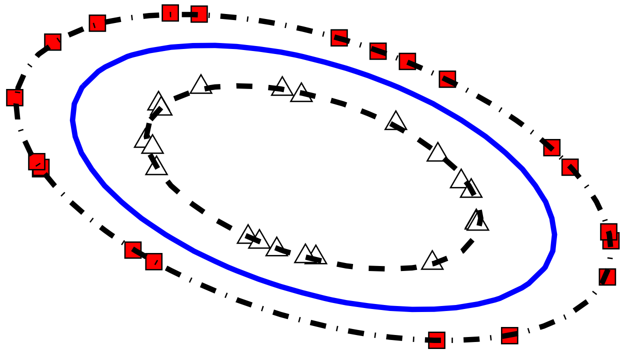
Iteration: 11, # of obs: 376



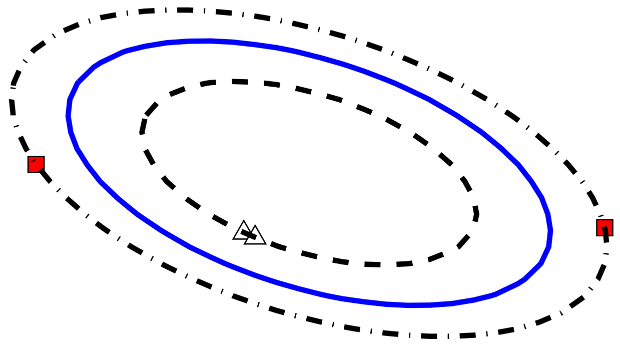
Iteration: 14, # of obs: 170



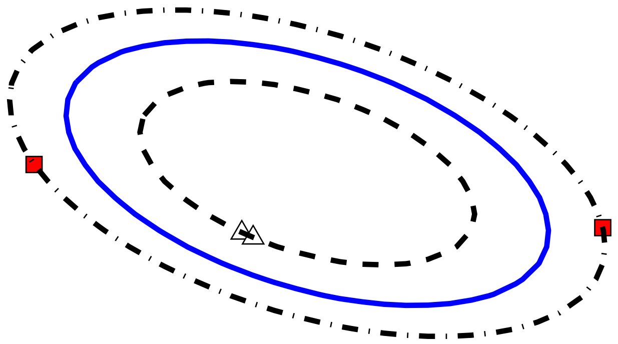
Iteration: 17, # of obs: 42



Iteration: 20, # of obs: 4



Iteration: 23, # of obs: 4



An extension

Recall that the matrix in the dual problem is $\mathbf{H} = \mathbf{D}\mathbf{A}\mathbf{A}^T\mathbf{D}$, and $\mathbf{A}$ is $m \times n$ with $m >> n$.

The elements of $\mathbf{K} \equiv \mathbf{A}\ \mathbf{A}^{\ T}$ are

$$k_{ij} = \mathbf{a}_i^T \mathbf{a}_j,$$

and our adaptive reduction is efficient because we approximate the term $\mathbf{A}^T \mathbf{\Omega}^{-1} \mathbf{A}$ in the formation of $\mathbf{M}$, which is only $n \times n$.

Often we want a kernel function more general than the inner product, so that

$$k_{ij} = k(\mathbf{a}_i, \mathbf{a}_j).$$

This would make $\mathbf{M}$ $m \times m$.

Efficiency is retained if we can approximate $\mathbf{K} \approx \mathbf{L}\mathbf{L}^T$ where $\mathbf{L}$ has rank much less than m. This is often the case, and pivoted Cholesky algorithms have been used in the literature to compute $\mathbf{L}$.

Conclusions

- We have succeeded in significantly improving the training of SVMs that have large numbers of training points.
- Similar techniques apply to general CQP problems with a large number of constraints.
- Savings is primarily in later iterations. Future work will focus on using clustering of patterns (e.g., Boley and Cao (2004)) to reduce work in early iterations.