Support Vector Machines: Algorithms and Applications

Tong Wen

LBNL

in collaboration with

Alan Edelman

MIT

Jan 17, 2003

Overview

- Support Vector Machines (SVMs) solve classification problems by learning from examples.

- Contents:

  1. Introduction to Support Vector Machines.
  2. Fast SVM training algorithms.
  3. Financial applications of SVMs?

- Notations:

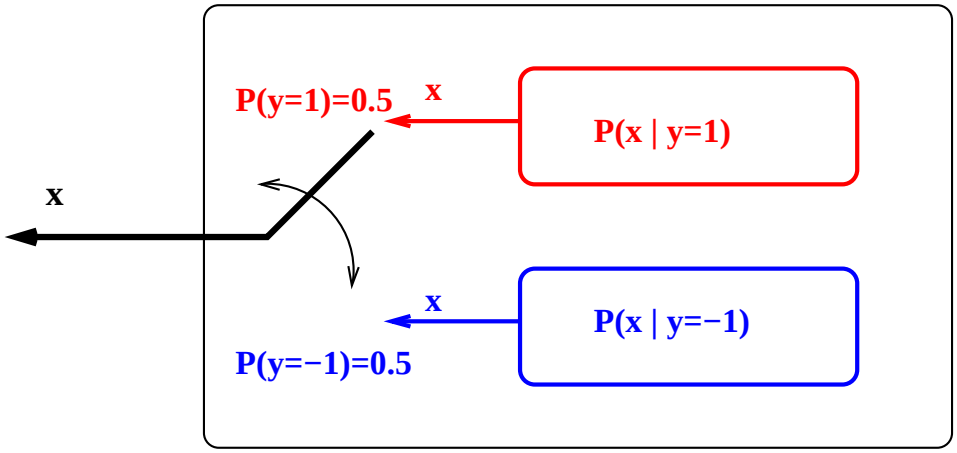
$\boldsymbol{x}_i$	the ith training point
y_i	the label of the ith training point
$\boldsymbol{\alpha}$	the vector of Lagrange multipliers
α_i	the ith Lagrange multiplier
$\boldsymbol{w}$	the vector giving the normal direction of a hyperplane
b	the bias term
n	the dimension of a training point
m	the number of training points
H	the Hessian matrix
X	the matrix whose columns are the training points
P	the matrix that spans the current search space
Q	the orthogonal complement of P
S_1	the set of SVs that are correctly separated
S_2	the set of SVs that are separated with errors
$h(\cdot)$	a decision rule (function)

Introduction to Support Vector Machines

- An example.
- The primal SVM quadratic programming problem.
- The dual SVM quadratic programming problem.
- Properties of Support Vectors(SVs).

An Example

- The mechanism that generates each point x and its color y:



- The main theme of classification is to determine a decision rule:

$$h : \mathbb{X} \subseteq \mathbb{R}^n \rightarrow \mathbb{Y} \subseteq \mathbb{R}.$$

  For this example, $\mathbb{Y} = \{1, -1\}$.

- What is the optimal solution to our detection problem?

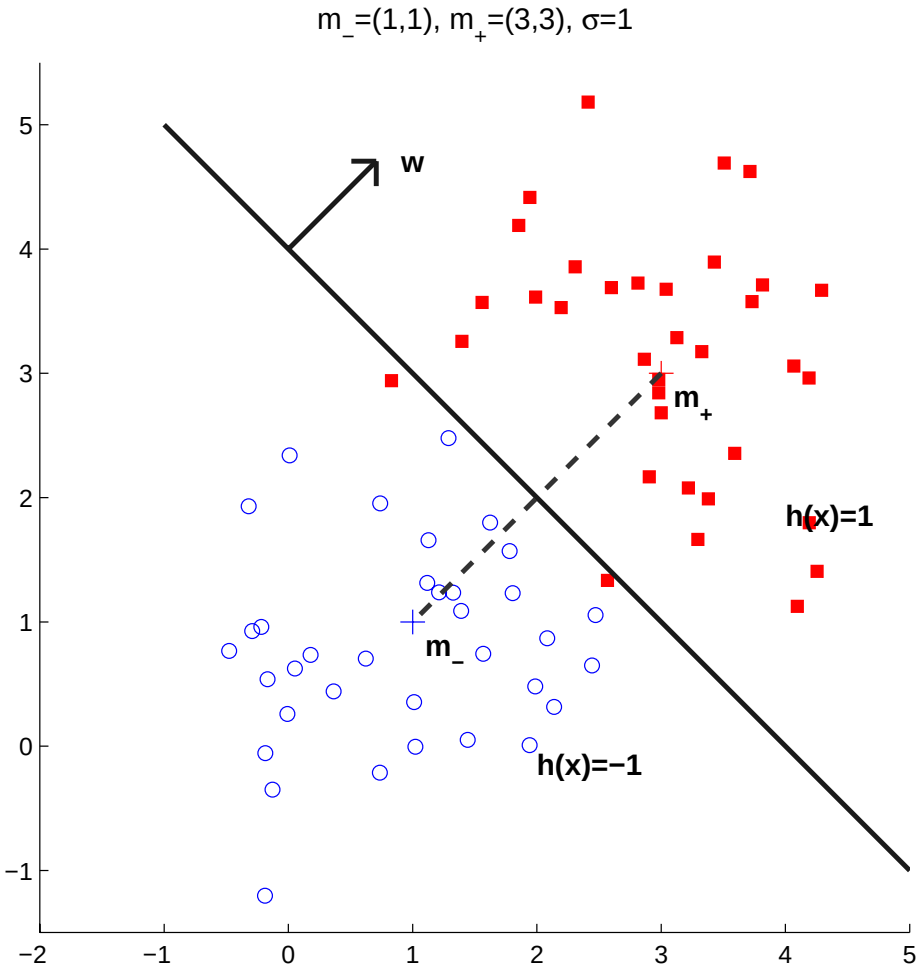
An Example

- By Bayes' Rule, the maximum likelihood decision rule is

$$h_{\mathrm{ML}}(\boldsymbol{x}) = \underset{\hat{y}\in\{\pm 1\}}{\operatorname{argmax}}\, p(\boldsymbol{x} \mid y = \hat{y}),$$

  which minimize the risk

$$R[h] = E[(\frac{1}{2}|y - h(\boldsymbol{x})|)].$$



Support Vector Machines

- The decision rule $h_{\mathsf{ML}}(\cdot)$ depends on $P(\boldsymbol{x}, y)$.

- In general, statistical information such as $P(\boldsymbol{x}, y)$ is not available. A distribution-free approach is needed.

- The SVM decision rule is determined based on the training examples:

$$S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}.$$

- The training error:

$$R_S[h] = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} |y_i - h(\boldsymbol{x_i})|.$$

- Overfitting vs. the training error.

Support Vector Machines

- In the previous example,

$$h_{\mathsf{ML}}(\boldsymbol{x}) = \mathsf{sgn}(\boldsymbol{w}^T\boldsymbol{x} + b).$$

- Every decision rule geometrically corresponds to a separating boundary.

- If $f_{\mathsf{SB}}(\boldsymbol{x}) = 0$ defines the separating boundary, then the corresponding decision rule has a general form

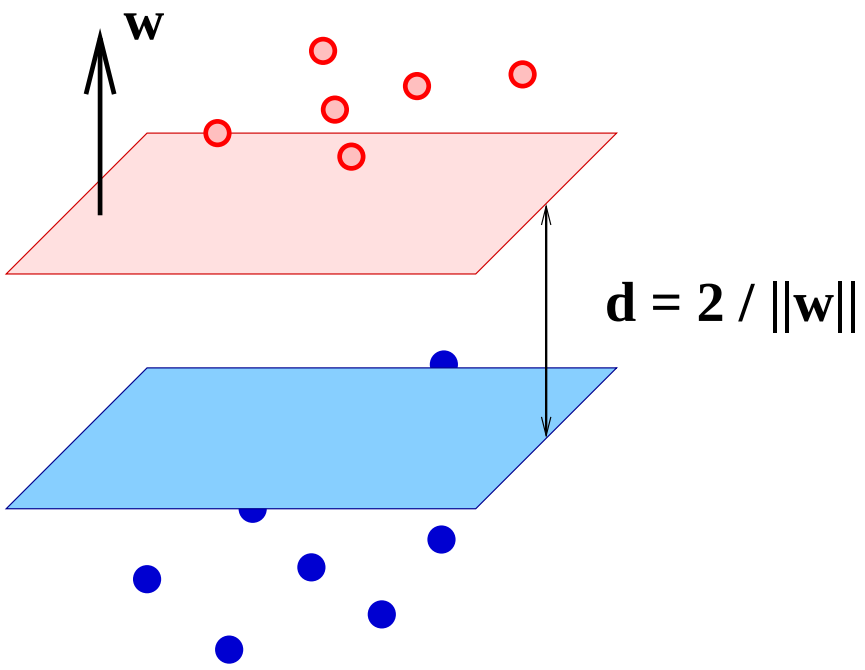
$$h(\boldsymbol{x}) = \mathsf{sgn}(f_{\mathsf{SB}}(\boldsymbol{x})).$$

- The SVM decision rule:

$$\begin{aligned} h_{\mathsf{SVM}}(\boldsymbol{x}) &= \mathsf{sgn}(\boldsymbol{w}^T\boldsymbol{x} + b), \\ h_{\mathsf{SVM}}(\boldsymbol{x}) &= \mathsf{sgn}(\boldsymbol{w}^T\phi(\boldsymbol{x}) + b) \\ &= \mathsf{sgn}(\sum_{i=1}^{m} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b). \end{aligned}$$

Support Vector Machines

- SVMs use two parallel hyperplanes instead of one to determine the separating boundary. These two (oriented) hyperplanes are computed directly using the training set S.



The Primal SVM QP Problem: the Separable Case

- $\boldsymbol{w}$ and b are determined by the quadratic programming (QP) problem:

$$\underset{w,b}{\text{minimize}} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2$$

subject to

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1, \ \text{ for } i = 1, \ldots, m.$$

- We can always separate the training points by mapping them to a higher dimensional space $\mathbb{R}^{n'}$ ($n' > n$), such that in $\mathbb{R}^{n'}$ the two sets $\{\phi(\boldsymbol{x}_i) \mid y_i = 1\}$ and $\{\phi(\boldsymbol{x}_i) \mid y_i = -1\}$ can be separated by hyperplanes.

- In practice, people may not want to fully separate the training set because of overfitting.

The Primal SVM QP Problem: the Inseparable Case

- To accommodate the separation errors on S, nonnegative slack variables (errors) ε_i are introduced:

$$\underset{w,b,\varepsilon}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + c\sum_i \varepsilon_i$$

  subject to (for $i = 1, \ldots, m$)

$$y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b) \geq 1 - \varepsilon_i \text{ and } \varepsilon_i \geq 0.$$

- If the training point $\phi(\boldsymbol{x}_i)$ is correctly separated by the soft-margin hyperplanes, then $\varepsilon_i = 0$.

- The coefficient c controls the trade-off between the size of the gap and how well S is separated.

The Dual SVM QP Problems

- In practice, the dual problems are computed instead:

$$\underset{\boldsymbol{\alpha}}{\text{maximize}} \quad \boldsymbol{\alpha}^T \mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^T H \boldsymbol{\alpha}$$

subject to

$$\begin{aligned} \boldsymbol{y}^T \boldsymbol{\alpha} &= 0, \\ \boldsymbol{\alpha} &\geq \mathbf{0}. \end{aligned}$$

$$\underset{\boldsymbol{\alpha}}{\text{maximize}} \quad \boldsymbol{\alpha}^T \mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^T H \boldsymbol{\alpha}$$

subject to

$$\begin{aligned} \boldsymbol{y}^T \boldsymbol{\alpha} &= 0, \\ \mathbf{0} \leq \boldsymbol{\alpha} &\leq \boldsymbol{c}. \end{aligned}$$

- The Hessian matrix $H_{ij} = y_i(\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j))y_j$ is a $m \times m$ symmetric positive semi-definite matrix.

The Positive Definite Kernel Functions

- Since only inner products are involved in the dual problems, a positive definite kernel function $k(\cdot, \cdot)$ is used instead of $\phi(\cdot)$, where

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j) \text{ and } H_{ij} = y_i k(\boldsymbol{x}_i, \boldsymbol{x}_j) y_j.$$

- Using kernel functions avoids the curse of dimensionality.

- Three positive definite kernels are used here:

linear:	$k(\boldsymbol{s}, \boldsymbol{t}) = \boldsymbol{s}^T \boldsymbol{t}$
poly(d):	$k(\boldsymbol{s}, \boldsymbol{t}) = (\boldsymbol{s}^T \boldsymbol{t} + 1)^d$
rbf(σ):	$k(\boldsymbol{s}, \boldsymbol{t}) = e^{-\|\boldsymbol{s}-\boldsymbol{t}\|_2^2/(2\sigma^2)}$

- For simplicity, we use $\boldsymbol{x}_i$ in place of $\phi(\boldsymbol{x}_i)$ in our later discussions.

Support Vectors

- At optimality,

$$\boldsymbol{w} = \sum_{i=1}^{m} y_i \alpha_i \boldsymbol{x}_i,$$

and if $0 < \alpha_i < c$ ($y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) = 1$) then

$$b = y_i - \boldsymbol{w}^T \boldsymbol{x}_i = y_i(1 - \boldsymbol{e}_i^T H \boldsymbol{\alpha}).$$

- $h_{\mathsf{SVM}}(\boldsymbol{x}) = \mathsf{sgn}(\boldsymbol{w}^T \boldsymbol{x} + b)$.

- $S_1 \cup S_2$ contain the support vectors (SVs).

$$\begin{cases} 0 < \alpha_i < c & \longleftrightarrow \boldsymbol{x}_i \in S_1,\ S_1 = \{\boldsymbol{x}_i |\ y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) = 1\ (\varepsilon_i = 0)\}; \\ \alpha_i = c & \longleftrightarrow \boldsymbol{x}_i \in S_2,\ S_2 = \{x_i |\ y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) = 1 - \varepsilon_i \text{ and } \varepsilon_i > 0\}; \\ \alpha_i = 0 & \longleftrightarrow \boldsymbol{x}_i \in S_3,\ S_3 = \{\boldsymbol{x}_i |\ y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) > 1\ (\varepsilon_i = 0)\}. \end{cases}$$

- SVs are the training points that determine the two separating hyperplanes through linear equations.

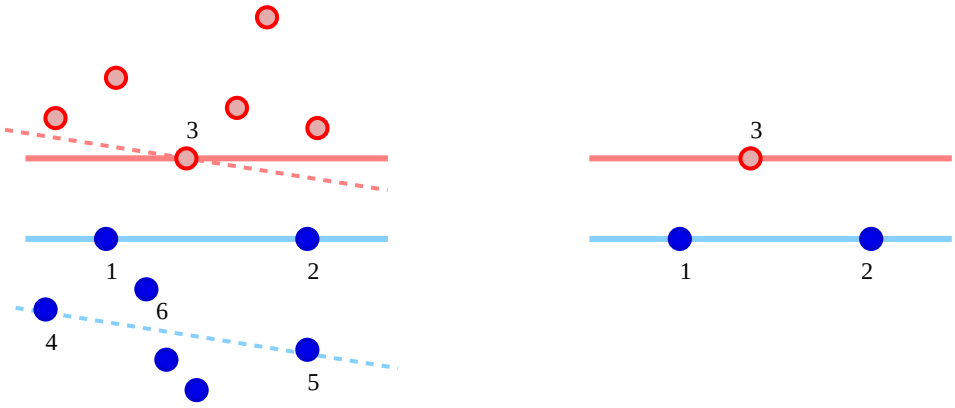
- The solution to the dual problem is sparse.

Solving the Dual SVM QP Problem Efficiently

- Solving the dual problem is computationally challenging.
- The decomposition strategy.
- A projected conjugate gradient algorithm.
- Speed considerations.
- Numerical experiments.
- Conclusion and future work.

The Decomposition Strategy

- The sparsity property.
- An observation from the primal problem point of view:



- How to apply this strategy to solve the dual problem efficiently?
  1. Constructing a subproblem.
  2. Solving the subproblems.
  3. The optimality conditions.
- A projected conjugate gradient algorithm.

Applying the Decomposition Strategy

- Projecting the dual QP problem onto a subspace.
  - Let us restrict $\boldsymbol{\alpha}$ in the affine space determined by $\boldsymbol{\alpha}_0$ and P. That is, $\boldsymbol{\alpha} = \boldsymbol{\alpha}_0 + P\hat{\boldsymbol{\alpha}}$.
  - The projected dual problem is

$$\underset{\hat{\boldsymbol{\alpha}}}{\text{maximize}} \;\; \hat{F}(\hat{\boldsymbol{\alpha}}) \equiv \hat{\boldsymbol{\alpha}}^T P^T \boldsymbol{r}_0 - \frac{1}{2}\hat{\boldsymbol{\alpha}}^T P^T H P \hat{\boldsymbol{\alpha}}$$

    subject to

$$\begin{aligned} \boldsymbol{y}^T P\hat{\boldsymbol{\alpha}} &= 0, \\ \mathbf{0} \le \boldsymbol{\alpha}_0 + P\hat{\boldsymbol{\alpha}} &\le \boldsymbol{c}, \end{aligned}$$

    where $\boldsymbol{r}_0$ is the gradient of $F(\boldsymbol{\alpha})$ at $\boldsymbol{\alpha}_0$.
  - P is a general matrix so far. It can be specified directly or by its orthogonal complement Q, where $P^T Q = 0$.

- How to determine P or Q, and how to solve each subproblem?

Constructing a Subproblem

- $\boldsymbol{\alpha} = \boldsymbol{\alpha}_0 + P\hat{\boldsymbol{\alpha}}$.

$P = E_p$ and $Q = E_q$ $(m = p + q)$
$$\underset{\hat{\boldsymbol{\alpha}}}{\text{maximize}} \;\; \hat{F}(\hat{\boldsymbol{\alpha}}) \equiv \hat{\boldsymbol{\alpha}}^T \hat{\boldsymbol{r}}_0 - \tfrac{1}{2}\hat{\boldsymbol{\alpha}}^T \hat{H} \hat{\boldsymbol{\alpha}}$$ subject to $$\hat{\boldsymbol{y}}^T \hat{\boldsymbol{\alpha}} = 0,$$ $$\mathbf{0} \le \hat{\boldsymbol{\alpha}}_0 + \hat{\boldsymbol{\alpha}} \le \boldsymbol{c},$$ where $\hat{\boldsymbol{r}_0} = P^T \boldsymbol{r}_0$, $\hat{\boldsymbol{y}} = P^T \boldsymbol{y}$, $\hat{H} = P^T H P$, etc.

$P = I - \frac{\hat{\boldsymbol{y}}\hat{\boldsymbol{y}}^T}{p}$, $Q = \hat{\boldsymbol{y}}$ and $\hat{\boldsymbol{\alpha}} = P\hat{\boldsymbol{\alpha}}$
$$\underset{\hat{\boldsymbol{\alpha}}}{\text{maximize}} \;\; \hat{F}(\hat{\boldsymbol{\alpha}}) \equiv \hat{\boldsymbol{\alpha}}^T P^T \hat{\boldsymbol{r}}_0 - \tfrac{1}{2}\hat{\boldsymbol{\alpha}}^T P^T \hat{H} P \hat{\boldsymbol{\alpha}}$$ subject to $$\mathbf{0} \le \hat{\boldsymbol{\alpha}_0} + P\hat{\boldsymbol{\alpha}} \le \boldsymbol{c}.$$

- $\hat{H}$ is a $p \times p$ matrix ($p \ll m$).

Solving the Subproblems by the CG Method

- If $\mathbf{0} \leq \hat{\boldsymbol{\alpha}}_0 + P\hat{\boldsymbol{\alpha}} \leq \boldsymbol{c}$ never becomes active, we have

$$P^T \hat{H} P \hat{\boldsymbol{\alpha}} = P^T \hat{\boldsymbol{r}}_0,$$

or equivalently

$$P \hat{H} P \hat{\boldsymbol{\alpha}} = P \hat{\boldsymbol{r}}_0 \ (P \text{ is symmetric}).$$

The CG iteration	The simplified version
$\hat{\boldsymbol{\alpha}}^{(0)} = 0, \ \boldsymbol{\gamma}^{(0)} = P\hat{\boldsymbol{r}}_0,$ $\boldsymbol{\rho}^{(0)} = \boldsymbol{\gamma}^{(0)}$ $\lambda = \frac{{\boldsymbol{\gamma}^{(l-1)}}^T \boldsymbol{\gamma}^{(l-1)}}{{\boldsymbol{\rho}^{(l-1)}}^T P \hat{H} P \boldsymbol{\rho}^{(l-1)}}$ $\hat{\boldsymbol{\alpha}}^{(l)} = \hat{\boldsymbol{\alpha}}^{(l-1)} + \lambda \boldsymbol{\rho}^{(l-1)}$ $\boldsymbol{\gamma}^{(l)} = \boldsymbol{\gamma}^{(l-1)} - \lambda P \hat{H} P \boldsymbol{\rho}^{(l-1)}$ $\mu = \frac{{\boldsymbol{\gamma}^{(l)}}^T \boldsymbol{\gamma}^{(l)}}{{\boldsymbol{\gamma}^{(l-1)}}^T \boldsymbol{\gamma}^{(l-1)}}$ $\boldsymbol{\rho}^{(l)} = \boldsymbol{\gamma}^{(l)} + \mu \boldsymbol{\rho}^{(l-1)}$	$\hat{\boldsymbol{\alpha}}^{(0)} = 0, \ \boldsymbol{\gamma}^{(0)} = P\hat{\boldsymbol{r}}_0,$ $\boldsymbol{\rho}^{(0)} = \boldsymbol{\gamma}^{(0)}$ $\lambda = \frac{{\boldsymbol{\gamma}^{(l-1)}}^T \boldsymbol{\gamma}^{(l-1)}}{{\boldsymbol{\rho}^{(l-1)}}^T \hat{H} \boldsymbol{\rho}^{(l-1)}}$ $\hat{\boldsymbol{\alpha}}^{(l)} = \hat{\boldsymbol{\alpha}}^{(l-1)} + \lambda \boldsymbol{\rho}^{(l-1)}$ $\boldsymbol{\gamma}^{(l)} = \boldsymbol{\gamma}^{(l-1)} - \lambda P \hat{H} \boldsymbol{\rho}^{(l-1)}$ $\mu = \frac{{\boldsymbol{\gamma}^{(l)}}^T \boldsymbol{\gamma}^{(l)}}{{\boldsymbol{\gamma}^{(l-1)}}^T \boldsymbol{\gamma}^{(l-1)}}$ $\boldsymbol{\rho}^{(l)} = \boldsymbol{\gamma}^{(l)} + \mu \boldsymbol{\rho}^{(l-1)}$

- How about $0 \leq (\hat{\boldsymbol{\alpha}}_0)_i + (P\hat{\boldsymbol{\alpha}})_i \leq c$ becomes active during the iteration?

The Optimality Conditions

- An equivalent problem:

$$Q = [E_q, \boldsymbol{y}] \text{ and } P = I - Q(Q^TQ)^{-1}Q^T.$$

- Define

$$\boldsymbol{d} = P\boldsymbol{r}$$

and

$$\boldsymbol{\beta} = -[I_{q\times q}, \mathbf{0}](Q^TQ)^{-1}Q^T\boldsymbol{r},$$

where $\boldsymbol{r} = \mathbf{1} - H\boldsymbol{\alpha}$ and P is a $m \times m$ matrix.

- α is optimal iff

$$\boldsymbol{d} = 0 \text{ and } \boldsymbol{\beta} \geq 0.$$

- Since $(Q^TQ)^{-1} = \begin{bmatrix} I + \frac{E_q^T\boldsymbol{y}\boldsymbol{y}^TE_q}{p} & \frac{-E_q^T\boldsymbol{y}}{p} \\ \frac{-\boldsymbol{y}^TE_q}{p} & \frac{1}{p} \end{bmatrix}$, it is easy to compute d and $\boldsymbol{\beta}$.

$$\boldsymbol{\beta} = E_q^T\boldsymbol{r} + \frac{E_q^T\boldsymbol{y}((E_q^T\boldsymbol{y})E_q^T\boldsymbol{r} - \boldsymbol{y}^T\boldsymbol{r})}{p}.$$

- Each time when active constraints are to be relaxed, we start with the one that has the most negative β_i.

A Projected Conjugate Gradient Algorithm

- A summary of our algorithm:

  $\boldsymbol{\alpha} = 0$;
  initialize E_p;
  while $\boldsymbol{d} \neq 0$ or $\boldsymbol{\beta} < 0$
    at most k steps of the CG iterations for the subproblem
    update $\boldsymbol{\alpha}$ and $\boldsymbol{r}$:

    $$\boldsymbol{\alpha} = \boldsymbol{\alpha} + E_p(I - \frac{\boldsymbol{y}\boldsymbol{y}^T}{p})\hat{\boldsymbol{\alpha}};$$

    $$\boldsymbol{r} = \boldsymbol{r} - HE_p(I - \frac{\boldsymbol{y}\boldsymbol{y}^T}{p})\hat{\boldsymbol{\alpha}};$$

    relax at most l active constraints with the most negative β_i;
    update E_p;
    compute the columns of H corresponding to the relaxed constraints;
    update HE_p and $\hat{H}$;
  end

- Memory requirement: (H is not precomputed!)

  $$X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m] \text{ and } HE_p.$$

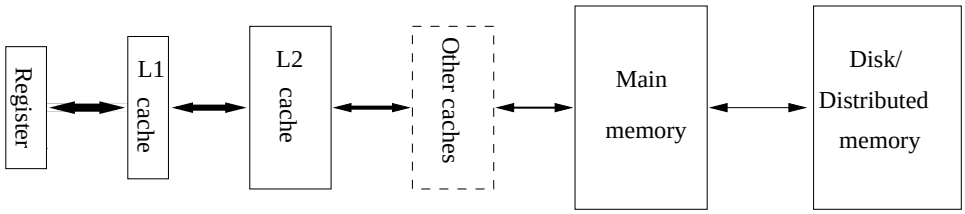
- Flops:
  Solving a sequence of smaller problems is cheaper when the solution is sparse.

- Two parameters: k and l.

- How to initialize E_p?

Speed Considerations:
Being Adaptive to the Memory Hierarchy

- A model of the modern computer memory hierarchy:



level	*clock cycles*
register	1
L1 cache	$2-3$
L2 cache	$6-12$
near main memory	$6-100$
far main memory	$100-250$
distributed main memory	$O(100)$
message-passing	$O(1000)-O(10000)$

- Linear algebra operations: level $1-3$.

- ATLAS BLAS.

- Thanks to MATLAB's inclusion of ATLAS BLAS.

Speed Considerations: Setting k and l Adaptively

- The spectrum of H.
  - The size of $S_1 \cup S_2$.
  - the convergence of the CG method.
- k is used to control the steps of the CG iterations.
  - There is no need to compute every subproblem exactly.
  - The rank deficiency of H.
- l is used to control the number of active constraints relaxed at each time.
  - When l active constraints are relaxed, $[\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_l}]^T X$ is computed to generate the corresponding columns of H.
  - If l is set large, then there is more chance of some previously relaxed constraints becoming active again.
- k and l is set to be $\widehat{k}$, the estimated number of the leading eigenvalues of H.
  - In our algorithm, $\widehat{k}$ is set to be the number of iterations for the CG method to reduce the residual norm to a scale of 0.01 for the initial subproblem.

Numerical Experiments

- The two training sets:

Training sets	Digit17	Face
Size of the training set	13007	31022
The positive training points	6742	2901
The negative training points	6265	28121
The dimension of the training points	784	361
The format of the training points	sparse	dense
Separability	easier	harder

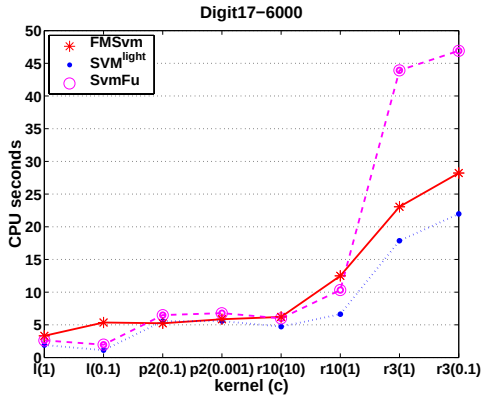
- Comparing against SVM^{light} (C) and SvmFu (C++).

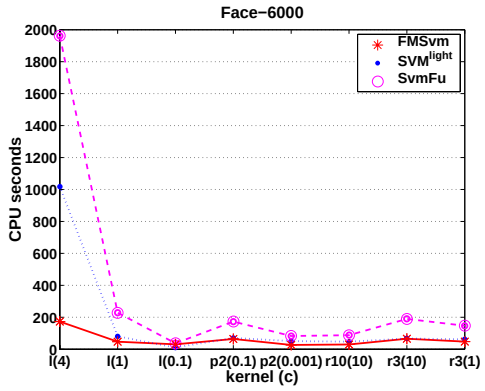
- The choices of $k(s,t)$ and c:

Training sets: Digit17-6000 and Digit17			
l(1)	l(0.1)	p2(0.1)	p2(0.001)
r10(10)	r10(1)	r3(1)	r3(0.1)
Training sets: Face-6000, Face-13901 and Face			
l(4)	l(1)	l(0.1)	p2(1)
p2(0.001)	r10(10)	r3(10)	r3(1)

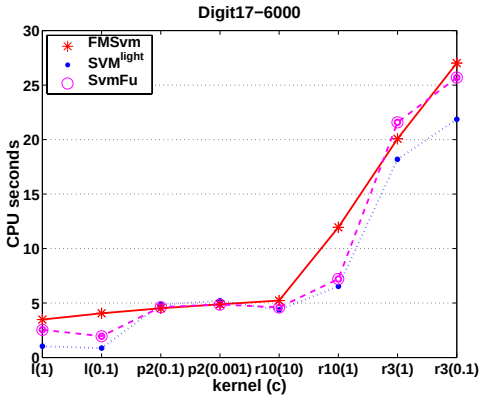
Numerical Experiments

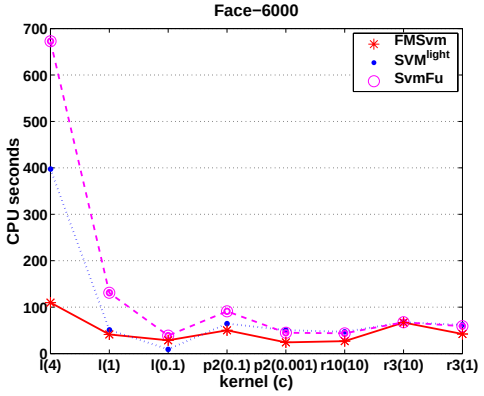
Using Default Settings





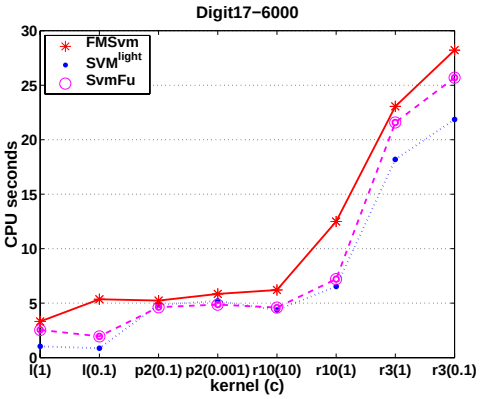
Using Estimated Optimal Settings

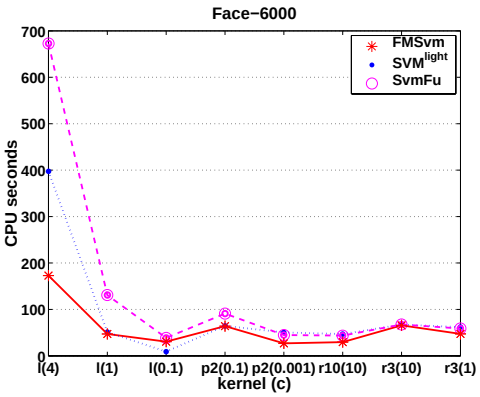


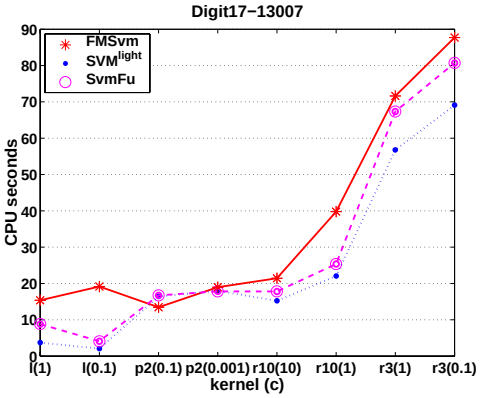


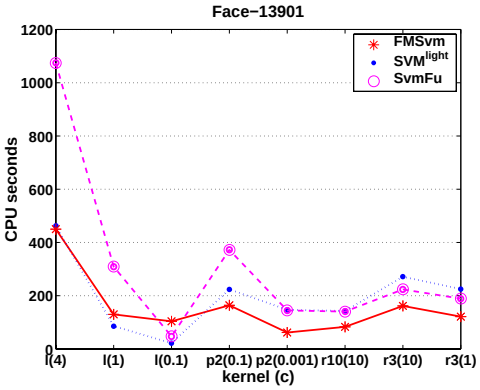
Numerical Experiments

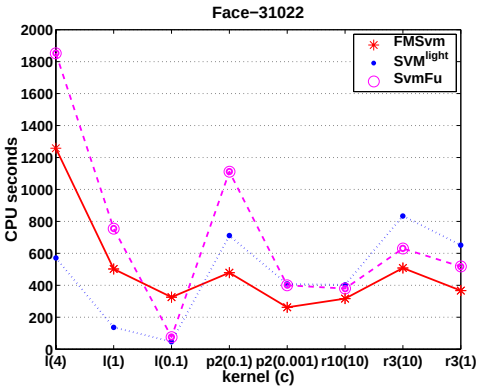
Comparison of FMSvm with SVM^{light} and SvmFu with the estimated optimal parameter settings











Conclusion and Future Work

- Conclusion:
  1. efficient;
  2. easy to use;
  3. highly portable.

- Future work:
  1. Try FMSvm on more training problems and platforms.
  2. Improve FMSvm’s performance and understand better its convergence behavior.

Applying SVMs to Financial Markets

- Identifying problems:
  - A mathematical model is available, but only an approximation.
  - Decisions are made based on experience.
- Constructing training sets:
  - Domain knowledges.
  - Data are from different categories.
  - Scaling the training data.
- Using SVMs to predict the movement of Dow Jones Industrials Average Index (DJIA):
  - The geometric Brownian motion solution.
  - The SVM solution.
  - The comparison between the two solutions.
- Conclusion and future work.

The Geometric Brownian Motion Model

- The weak form of the market-efficiency hypothesis.

- The equations:

$$dS = \mu S dt + \sigma S dz.$$

  By Itô's lemma, we have

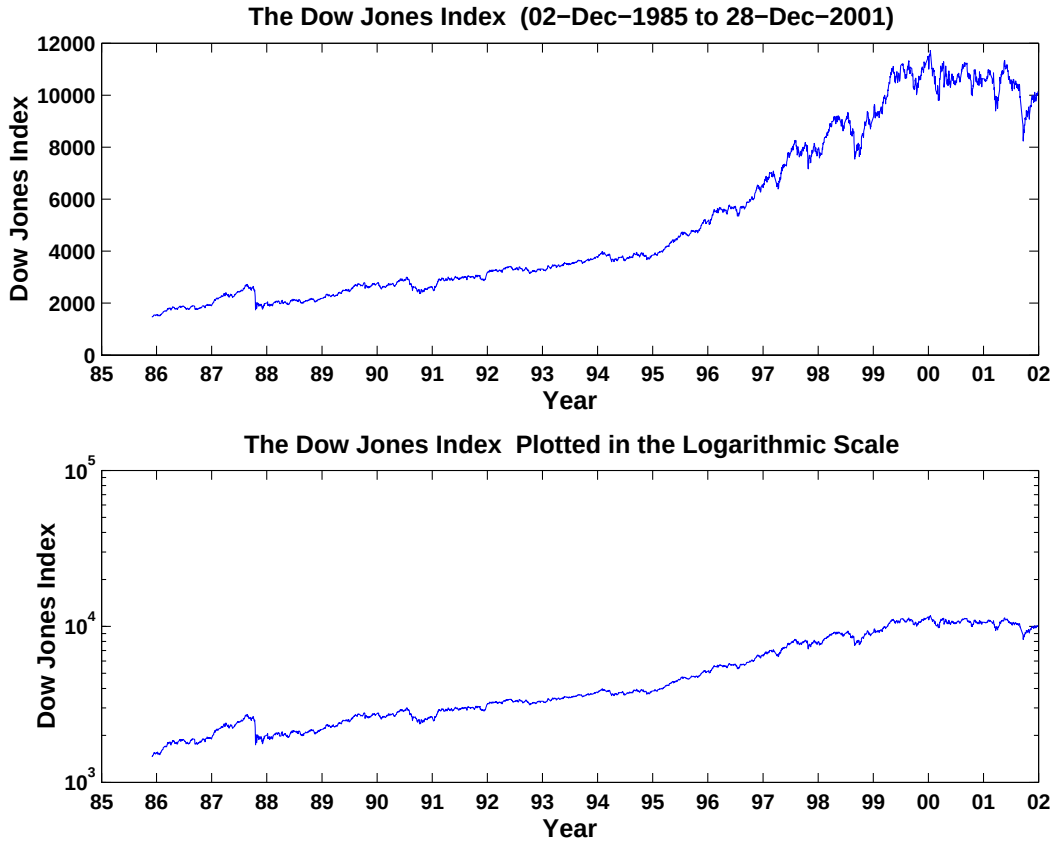
$$d \ln S = (\mu - \frac{\sigma^2}{2})dt + \sigma dz.$$

  That is,

$$\ln \frac{S_T}{S_0} \sim N((\mu - \frac{\sigma^2}{2})T, \sigma^2 T).$$

- We want to predict on Thursday (after the market is closed) whether or not the closing index of DJIA on Friday is higher than its opening index on Monday.

The Dow Jones Industrial Average Index



The Geometric Brownian Motion Solution

- The geometric Brownian motion model:

$$\frac{dS}{S} = \mu dt + \sigma dz.$$

- Its discrete-time version:

$$\frac{\Delta S}{S} = \mu \Delta t + \sigma \epsilon \sqrt{\Delta t}.$$

$$\frac{\Delta S}{S} \sim N(\mu \Delta t, \sigma^2 \Delta t).$$

- Stock price on Friday: $(R = \frac{S_{\mathsf{Fr}} - S_{\mathsf{Th}}}{S_{\mathsf{Th}}} \sim N(\mu, \sigma^2))$

$$\begin{aligned} S_{\mathsf{Fr}} &= (1 + R) S_{\mathsf{Th}} \\ &= (1 + \mu + \sigma\epsilon) S_{\mathsf{Th}}. \end{aligned}$$

- We want to know $S_{\mathsf{Fr}} \gtrless S_{\mathsf{Mo}}$.

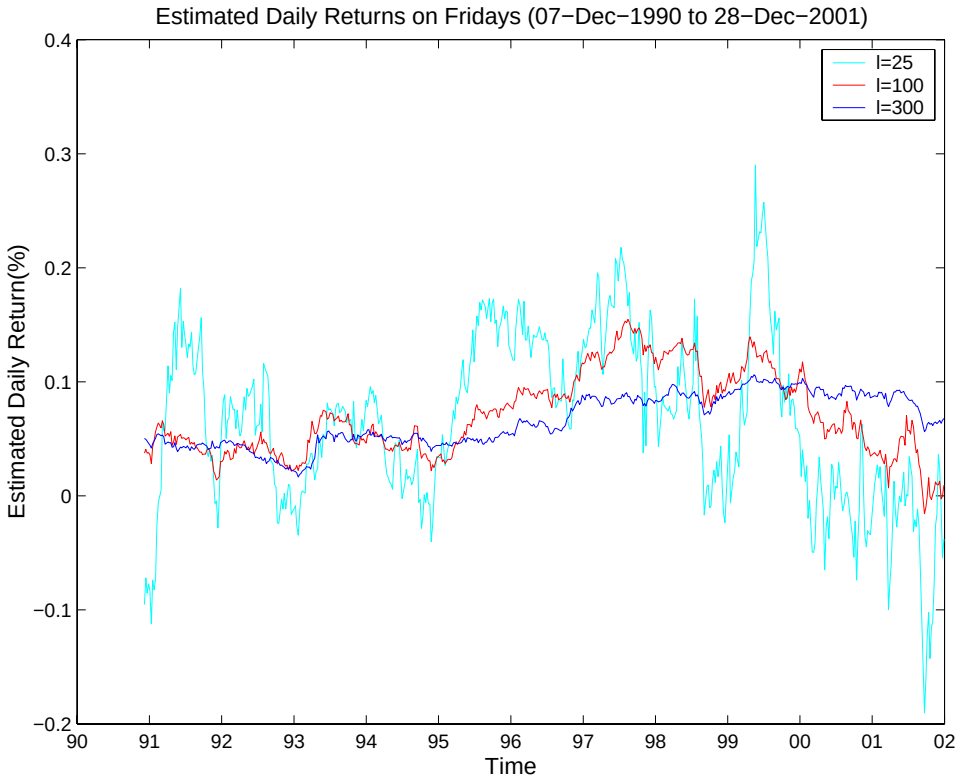
$$\epsilon \underset{<}{\geq} \frac{S_{\mathsf{Mo}}/S_{\mathsf{Th}} - 1 - \mu}{\sigma} = \xi.$$

- Since $\begin{cases} \Pr(\epsilon > \xi) > \Pr(\epsilon < \xi) \text{ if } \xi < 0 \\ \Pr(\epsilon > \xi) < \Pr(\epsilon < \xi) \text{ if } \xi > 0, \end{cases}$

$$h_{\mathsf{GBM}}(\xi) = \mathsf{sgn}(-\xi).$$

The Geometric Brownian Motion Solution

- Estimating μ and σ: only μ matters.



- The test set: December 7, 1990 to December 28, 2001.

- The performance: 96 errors out 511 trials. The accuracy of h_{GBM} on this test set is 81.21%.

The SVM Solution

- $h_{\mathrm{SVM}}(\boldsymbol{x}) = \mathrm{sgn}(\boldsymbol{w}^T\boldsymbol{x} + b)$, where "experience" is incorporated in $\boldsymbol{w}$ and b.
- Our first training set: July 5, 1954 to November 30,1990.

  $\boldsymbol{x} = [341.3\ 343.6\ 339.4\ 340.9\ 23300\ \ldots\ 340.4\ 343.0\ 339.2\ 341.1\ 30000]^T$
- Scaling the raw training vectors:

  $\boldsymbol{x}_s = [1\ 1.007\ 0.994\ 0.999\ 1\ \ldots\ 0.997\ 1.005\ 0.994\ 0.999\ 1.288]^T$
- Results: (There are totally 1704 training vectors, and the test set contains 511 vectors.)

poly(4)							
c	0.2	0.5	0.8	1	1.2	1.4	1.6
Errors	106	99	96	96	96	98	97
rbf(1)							
c	20	40	60	80	100	120	140
Errors	109	101	101	98	96	96	99

The SVM Solution

- Our intuition is that if there is any available information correlated to the movement of DJIA, then adding it to the training set may improve the performance.

- We add the daily federal funds rate to the training set. We choose this rate simply because it is a daily rate.

- Again, we use the relative values with respect to the Monday rate in the training vectors, instead of real rates.

- Results:

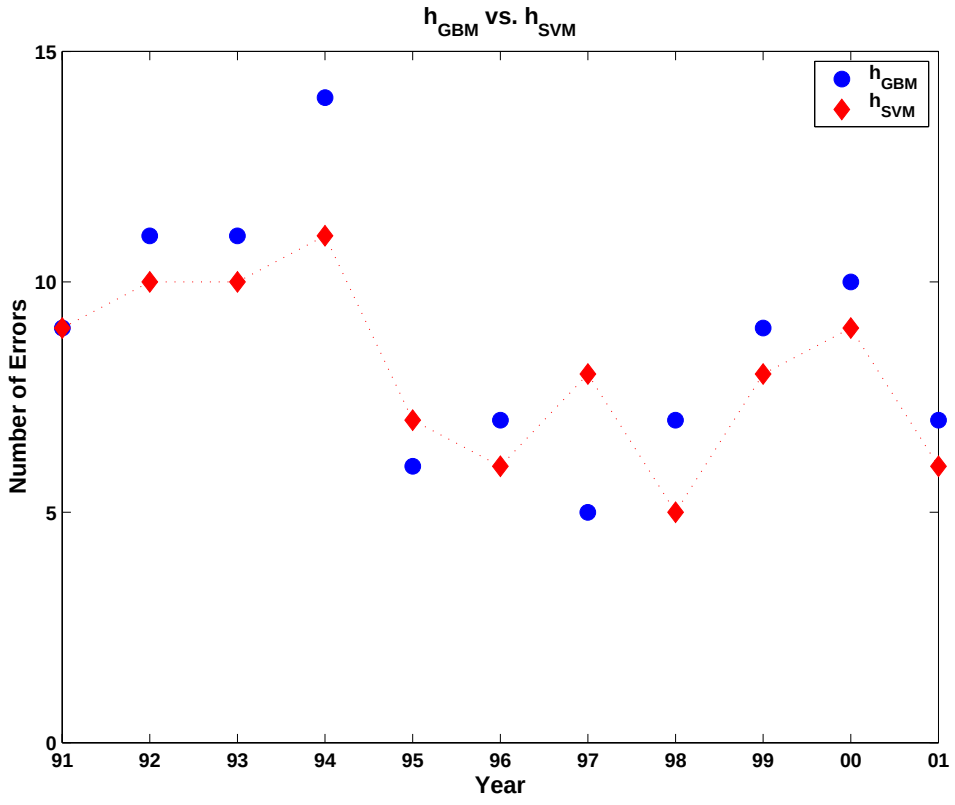
poly(3)								
c	0.05	1	1.5	2.5	10	20	30	35
Errors	103	102	96	96	94	93	89	89

- When $c = 30$, it takes around 2.6 hours (on `newton.mit.edu`) to train the SVM comparing with the scale of seconds when using the old training set.

The Geometric Brownian Motion Model VS. SVMs

- The Geometric Brownian Motion model is only an approximation.

- The semi-strong form of the market-efficiency hypothesis.



Conclusions and Future Work

- Conclusions:
  1. SVMs can learn well from time series data.
  2. There are potentials to achieve better performance using SVMs.

- Future work:
  1. We want to find good applications of SVMs (need domain knowledge).
  2. How to construct a training set based on data from different categories is also an interesting problem.