Predictive Learning from Data

LECTURE SET 9-1

Margin-Based Methods and Support Vector Machines

Cherkassky, Vladimir, and Filip M. Mulier. *Learning from data: concepts, theory, and methods*. John Wiley & Sons, 2007.

Source: Dr. Vladimir Cherkassky (revised by Dr. Hsiang-Han Chen)

OUTLINE

- **Motivation**
- Margin-based loss
- SVM for classification
- SVM classification: examples
- Support Vector Regression
- SVM and regularization
- Summary

Note: presentation follows Ch. 9 (Cherkassky & Mulier, 2007)

Motivation for Nonlinear Methods

1. Predictive learning algorithm is proposed using ‘reasonable’ heuristic arguments.

   reasonable ~ statistical or biological
2. Empirical validation + improvement
3. Statistical explanation (why it really works)

Examples: CART, MARS, neural nets, AdaBoost

In contrast, SVM methodology was originally introduced and motivated in VC-theory.

SVM: Brief History

1963 Margin (Vapnik & Lerner)

1964 Margin (Vapnik and Chervonenkis, 1964)

1964 RBF Kernels (Aizerman)

1965 Optimization formulation (Mangasarian)

1971 Kernels (Kimeldorf annd Wahba)

1992-1994 SVMs (Vapnik et al)

1996 – present Rapid growth, numerous apps

1996 – present Extensions to advanced settings

Growing Popularity of SVMs

- **GOOGLE search on SVM**
  → 89.8 mln results
- **GOOGLE search on Kernel Methods**
  → 154.0 mln results
- **GOOGLE search on Classification And Regression Trees**
  → 32.2 mln results
- **GOOGLE search on Multilayer Perceptrons**
  → 13.2 mln results

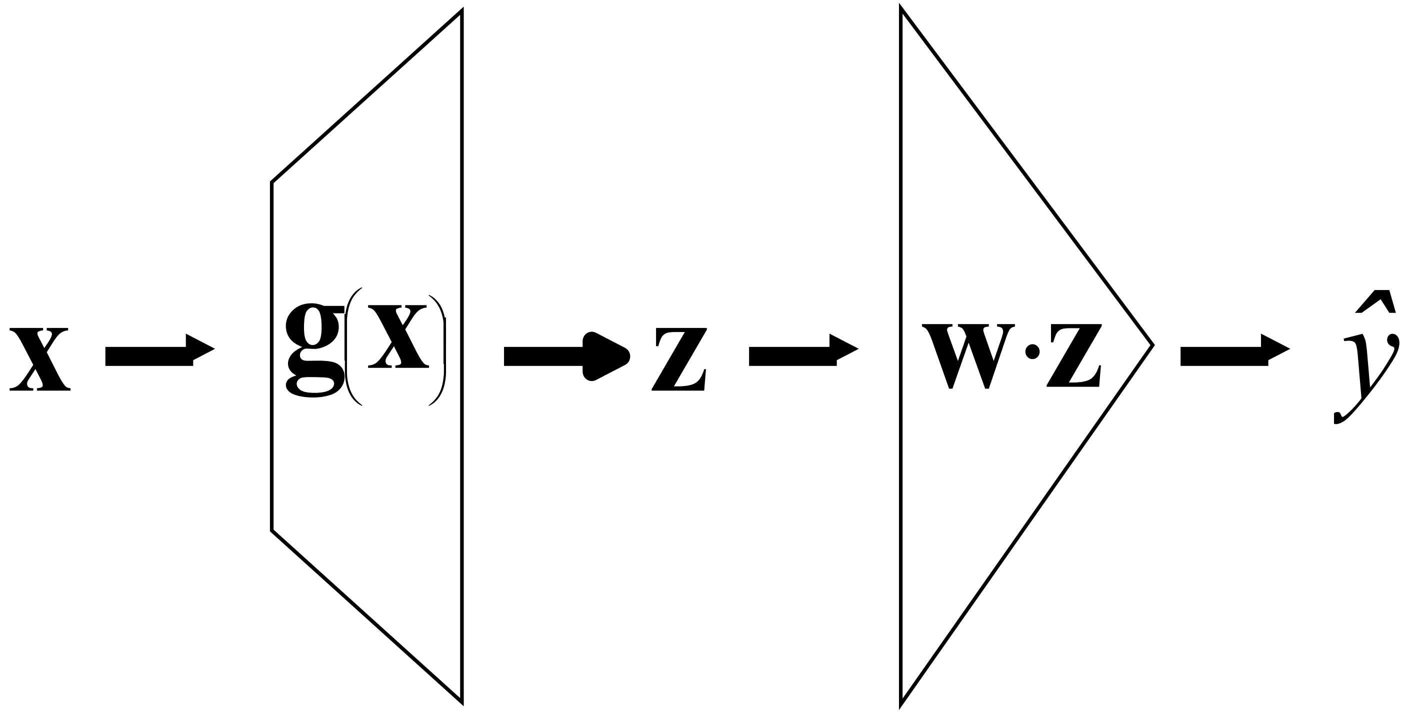
BUT plenty of conceptual misunderstanding

MOTIVATION for SVM

- **Recall 'conventional' methods:**
  - model complexity ~ dimensionality (# features)
  - nonlinear methods → multiple minima
  - hard to control complexity
- **'Good' learning method:**
  (a) tractable optimization formulation
  (b) tractable complexity control(1-2 parameters)
  (c) flexible nonlinear parameterization
- Properties (a), (b) hold for 'linear methods'
- **SVM solution approach**

SVM Approach

- Linear approximation in **Z**-space using special adaptive loss function
- Complexity is independent of dimensionality (via the number of support vectors)



SVM Approach

SVM combines four distinct concepts:

1. **New implementation of the SRM inductive principle**: a special structure indexed by adaptive margin-based loss function.
2. **Mapping of inputs onto a high-dimensional space using a set of nonlinear basis functions defined a priori.**
3. **Linear functions with constraints on complexity.**
4. **Duality theory of optimization**

OUTLINE

- Motivation
- **Margin-based loss**
  - Example: binary classification
  - Philosophical motivation
  - Loss functions for classification
  - VC-theoretical motivation
- SVM for classification
- SVM classification: examples
- Support Vector Regression
- SVM and regularization
- Summary

Importance of Reward/ Punishment

Loss function (problem-specific) can be used
to control model complexity (generalization)

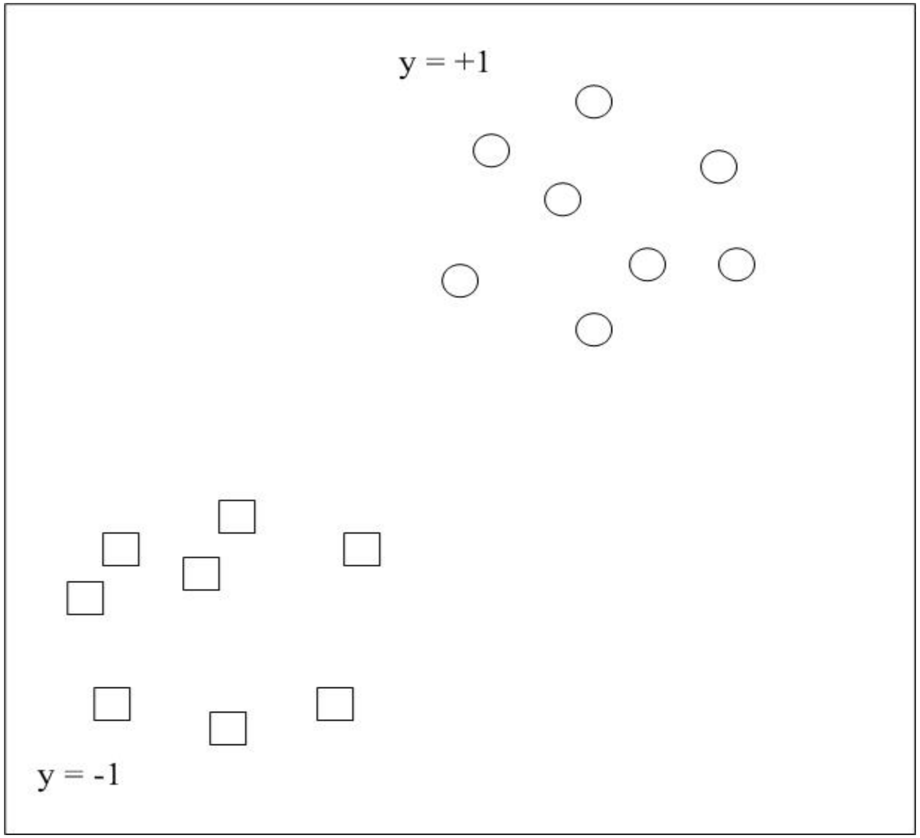
Main Idea

- Model complexity is controlled by *special loss function* used for fitting training data
- Such **empirical loss functions** may be **different** from the loss functions used for learning problem setting
- Such loss functions are **adaptive**, i.e. can adapt their complexity to given data set
- Different loss functions for different learning problems (classification, regression etc.)
- Model complexity (VC-dim.) is controlled **independently** of the number of features

Example: binary classification

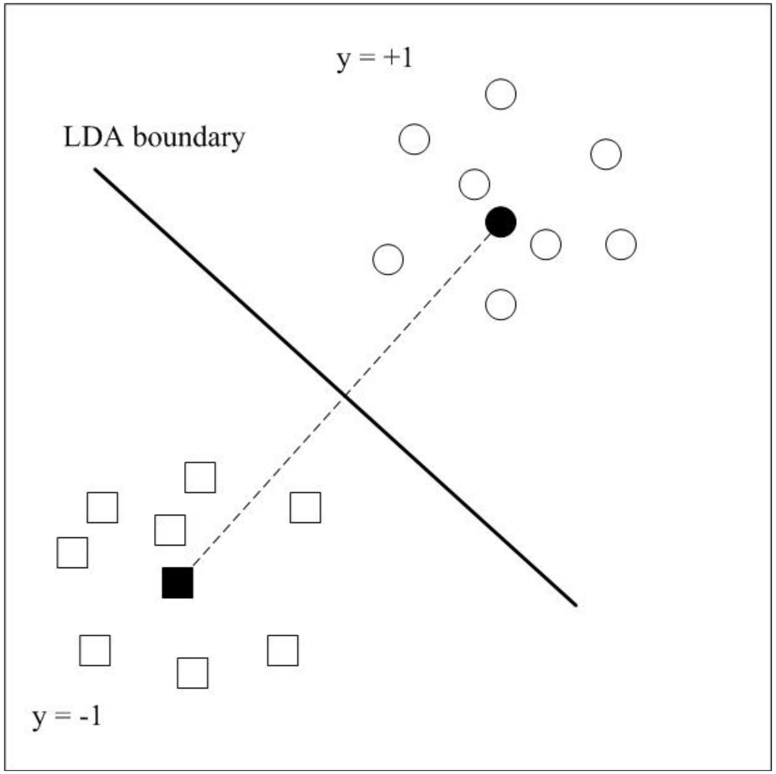
Given: Linearly separable data

How to construct good linear decision boundary ?

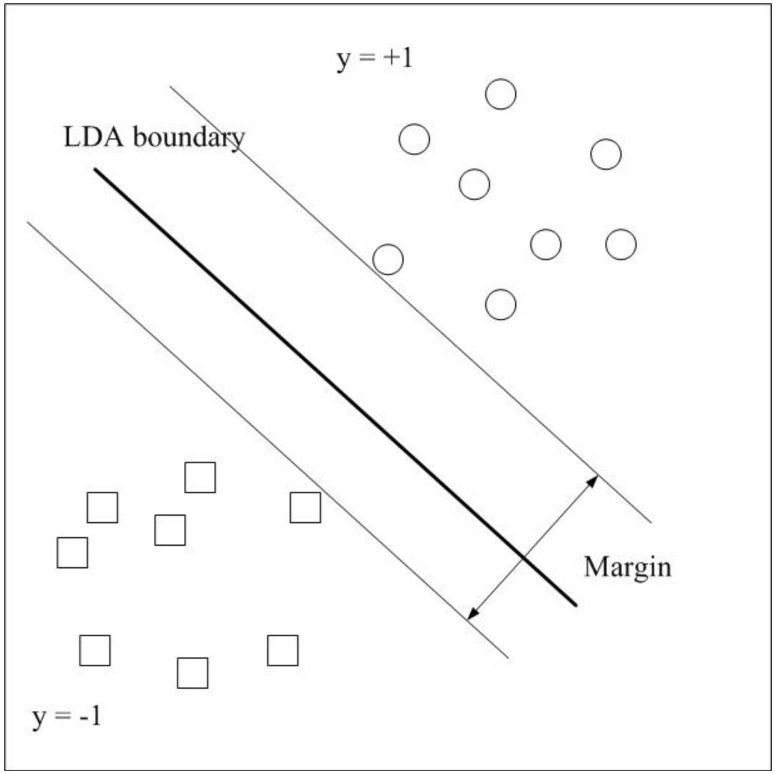


Linear Discriminant Analysis

LDA solution

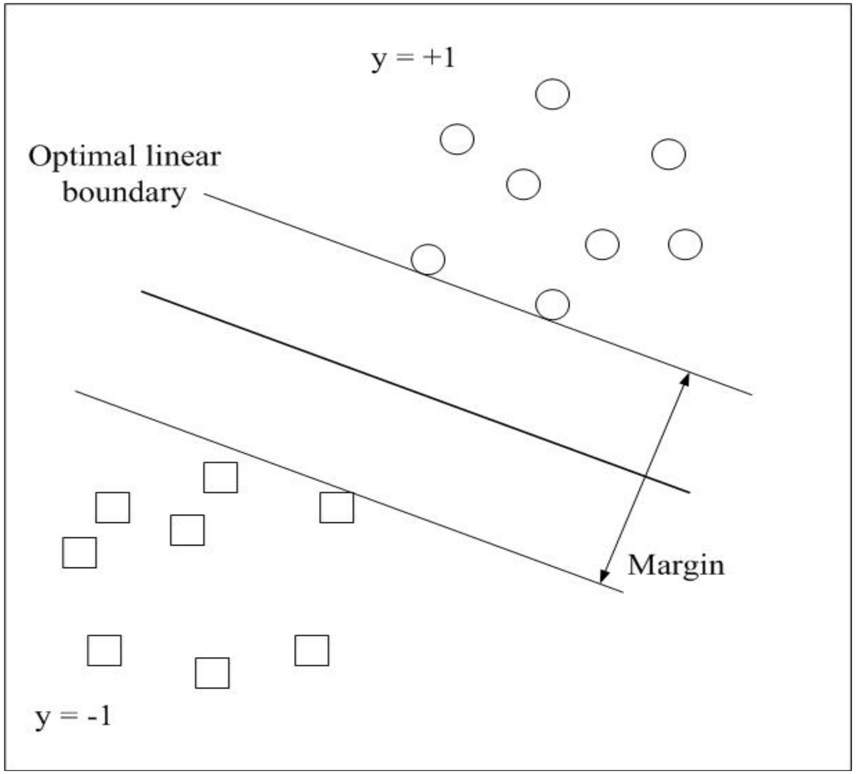


Separation margin



Largest-margin solution

- Many solutions explain the data well (zero error)
  All solutions ~ the same linear parameterization
  Larger margin ~ more confidence (falsifiability)



$$M = 2\Delta$$

Motivation: philosophical

- **Classical view:** good model explains the data + low complexity

→ Occam's razor (complexity ~ # parameters)

- **VC theory:** good model explains the data + low VC-dimension

  ~ VC-falsifiability: good model explains the data + has large falsifiability

The idea: falsifiability ~ *empirical loss function*

Adaptive loss functions

- Both goals (explanation + falsifiability) can encoded into empirical loss function where

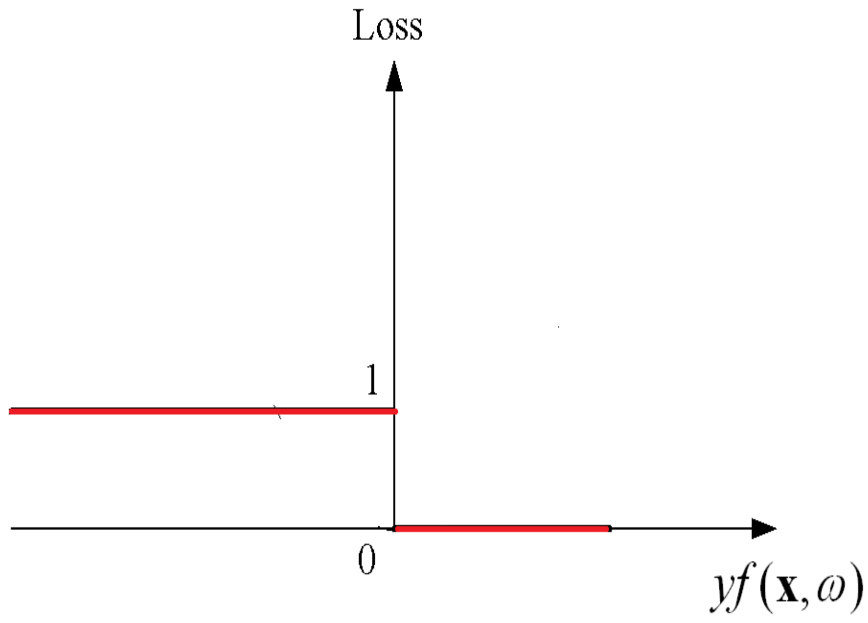
  - (large) portion of the data has zero loss

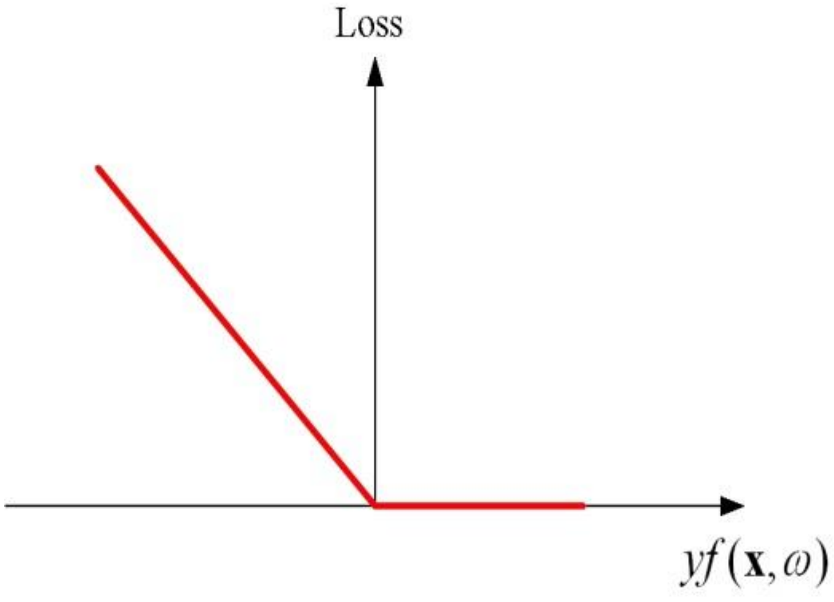
  - the rest of the data has non-zero loss (uncertainty). i.e. this data falsifies the model
- The trade-off (between the two goals) is adaptively controlled → adaptive loss function
- Example of such adaptive loss function for classification problem is shown next

Loss Functions for Classification

- Decision rule $D(\mathbf{x}) = sign\left(f(\mathbf{x},\omega)\right)$
- Quantity $yf(\mathbf{x},\omega)$ is analogous to residuals in regression
- Common loss functions: 0/1 loss and linear loss
- Properties of a good loss function?

  ~ continuous + convex + robust

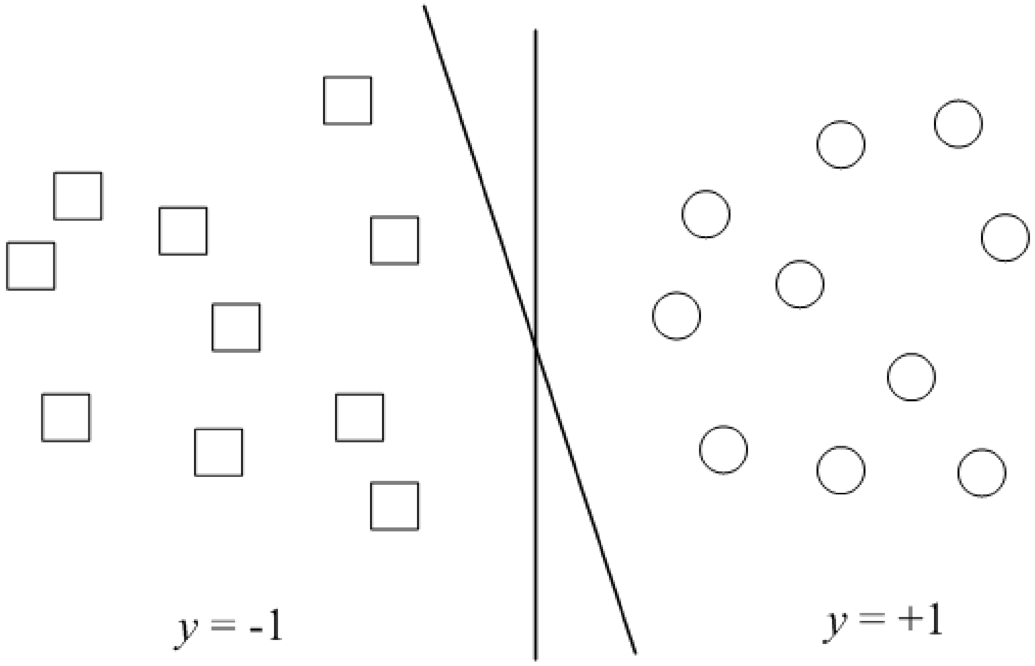




Motivation for margin-based loss

- **Given:** Linearly separable data

How to construct linear decision boundary?

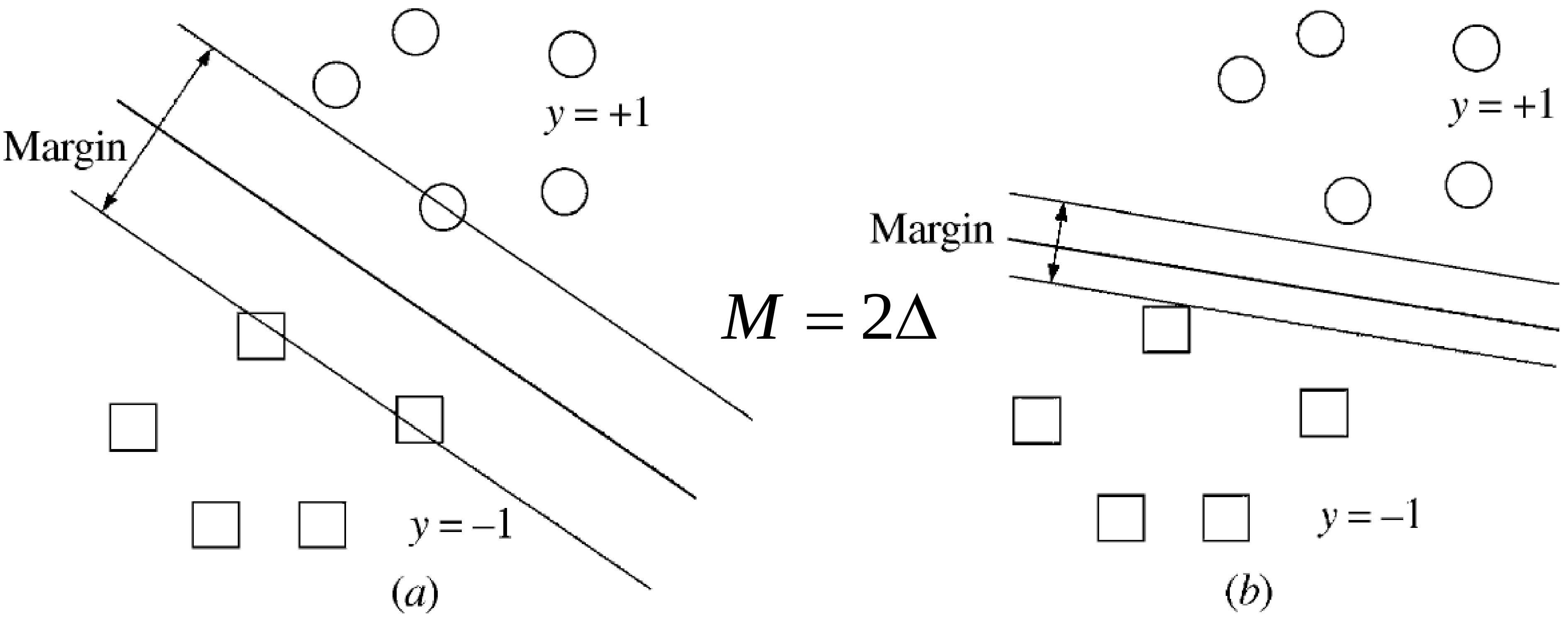


(a) Many linear decision boundaries (that have no errors)

Largest-margin solution

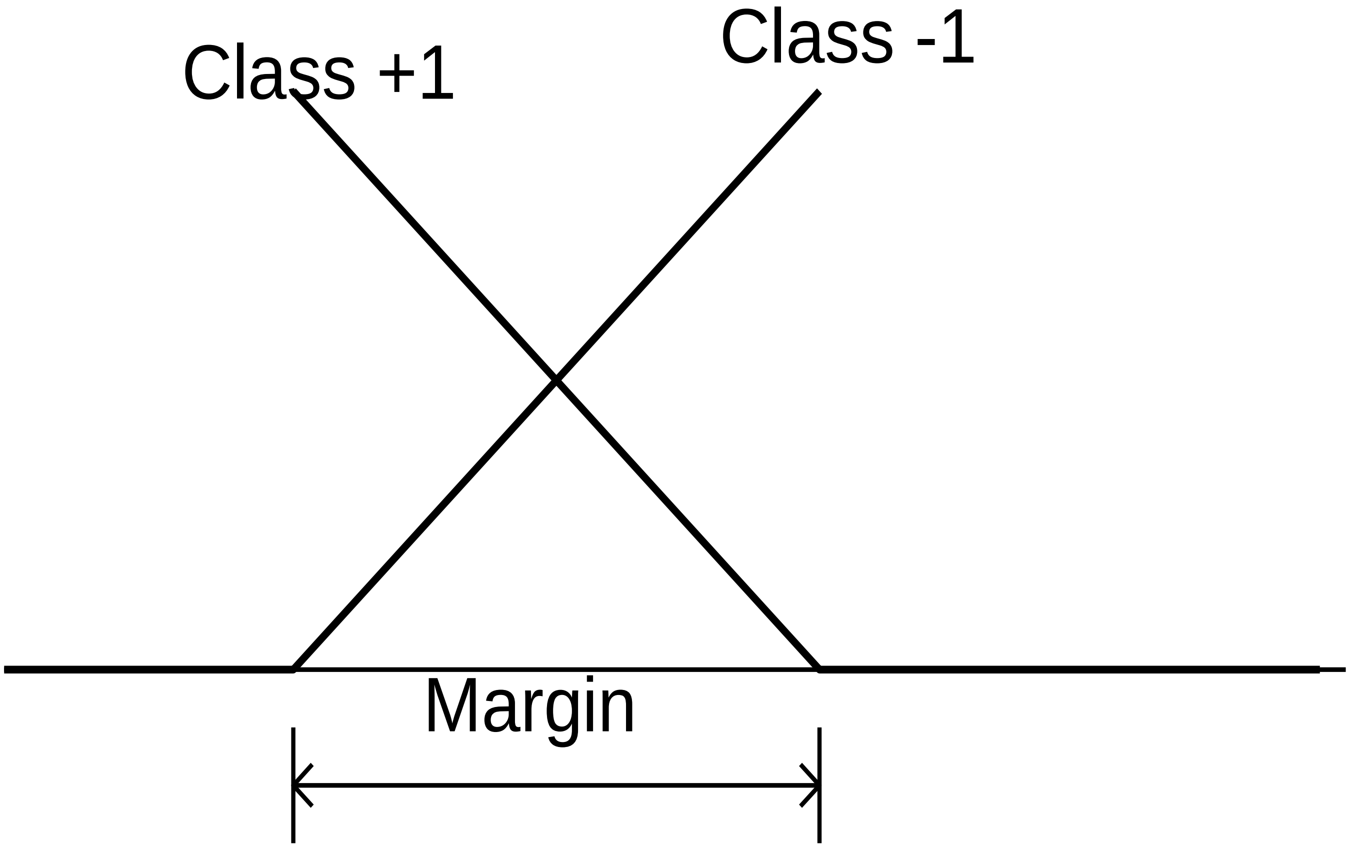
- Good decision boundary :

explains the data + has large falsifiability (i.e., confidence)



$$M = 2\Delta$$

Margin-based loss for classification: margin size is adapted to training data



$$M = 2\Delta$$

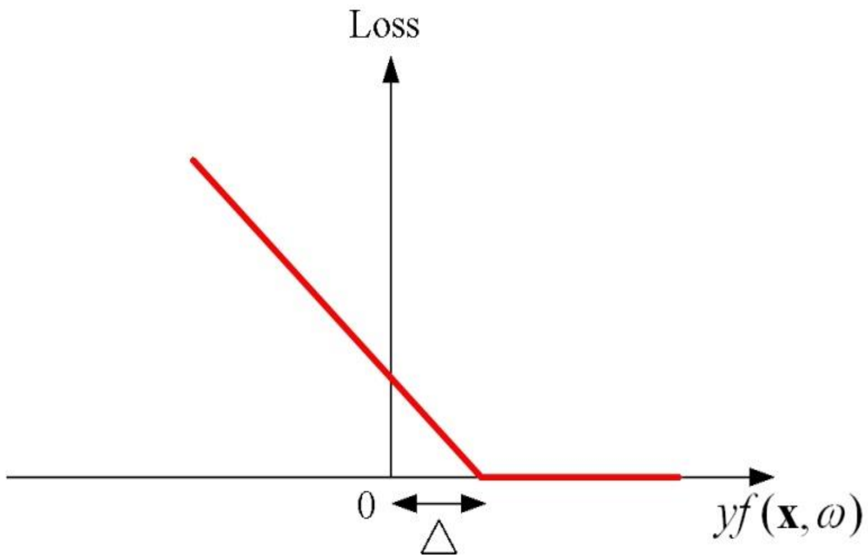
$$y \cdot f(\mathbf{x}, \omega)$$

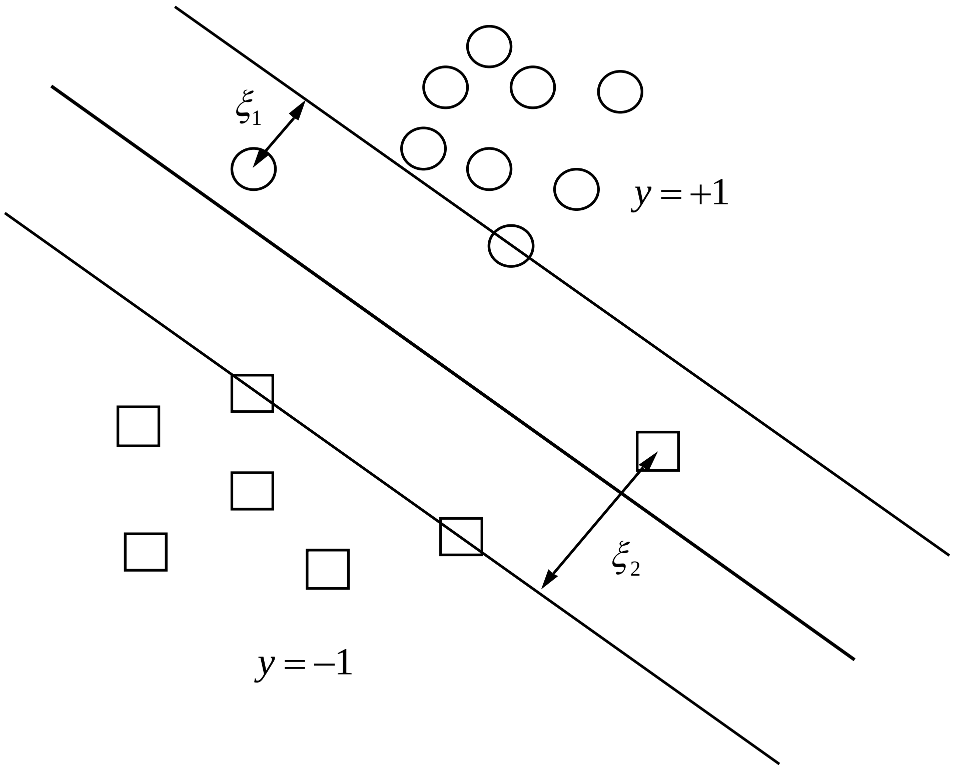
$$L_{\Delta}(y, f(\mathbf{x}, \omega)) = \max\left(\Delta - yf(\mathbf{x}, \omega), 0\right)$$

Margin-based loss for classification

SVM loss or hinge loss $L_\Delta(y, f(\mathbf{x},\omega)) = \max\left(\Delta - yf(\mathbf{x},\omega), 0\right)$

Minimization of *slack variables* $\xi_i = \Delta - y_i f(\mathbf{x}_i, \omega)$





Keep samples away from the decision boundary.

Margin based complexity control

- Large degree of falsifiability is achieved by
  - large margin (for classification)
- Explanation of training data is achieved by
  - minimization of slack variables
- For linear classifiers:
  larger margin → smaller VC-dimension

$$\Delta_1 > \Delta_2 > \Delta_3 > \ldots \sim \qquad h_1 < h_2 < h_3 < \ldots$$

VC-theoretical Motivation

- SVM model complexity (~ VC-dimension)

- New implementation of SRM using VC-bounds

- Adaptive margin loss functions for other types of learning problems:

  - for regression

  - for single-class learning

Complexity of Δ-margin hyperplanes

- If data samples belong to a sphere of radius *R*, then the set of Δ-margin hyperplanes has VC dimension bounded by

$$h \le \min(R^2/\Delta^2, d) + 1$$

- For large margin hyperplanes, VC-dimension controlled **independent** of dimensionality *d*.

SVM Model Complexity

- Two ways to control model complexity

  -via model parameterization $f(\mathbf{x}, \omega)$

  using fixed loss function: $L(y, f(\mathbf{x}, \omega))$

  -via adaptive loss function: $L_{\Delta}(y, f(\mathbf{x}, \omega))$

  using fixed (linear) parameterization $f(\mathbf{x}, \omega) = (\mathbf{w} \cdot \mathbf{x}) + b$

- ~ Two types of SRM structures

VC Generalization Bound and SRM

- **Classification:** the following bound holds with probability of $1-\eta$ for all approximating functions

$$R(\omega) < R_{emp}(\omega) + \Phi\left(R_{emp}(\omega), n/h, -\ln\eta/n\right)$$

- **Two general strategies for implementing SRM**:

1. Keep Φ fixed and minimize $R_{emp}(\omega)$

   (most statistical and neural network methods)

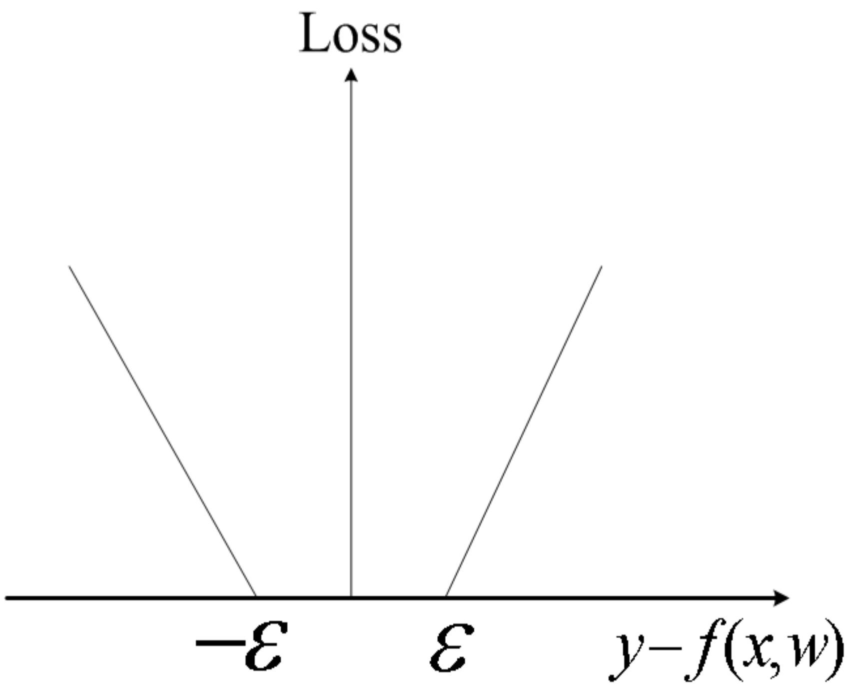
2. Keep $R_{emp}(\omega)$ fixed (small) and minimize Φ

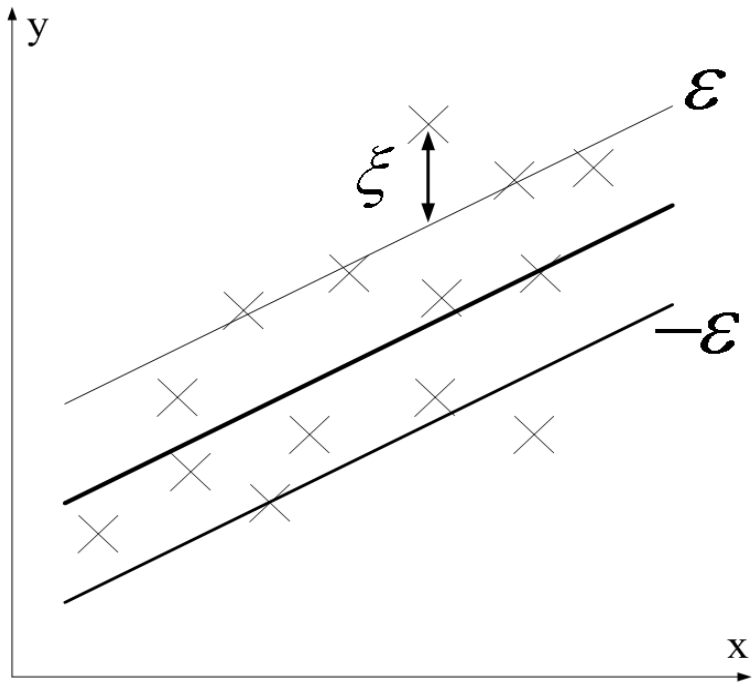
   larger margin → smaller VC-dimension

- Equivalence classes $F_0, F_1, \ldots, F_N$ on a set of possible models:

  - for each class F_k select the *largest-margin* hyperplane

Epsilon loss for regression



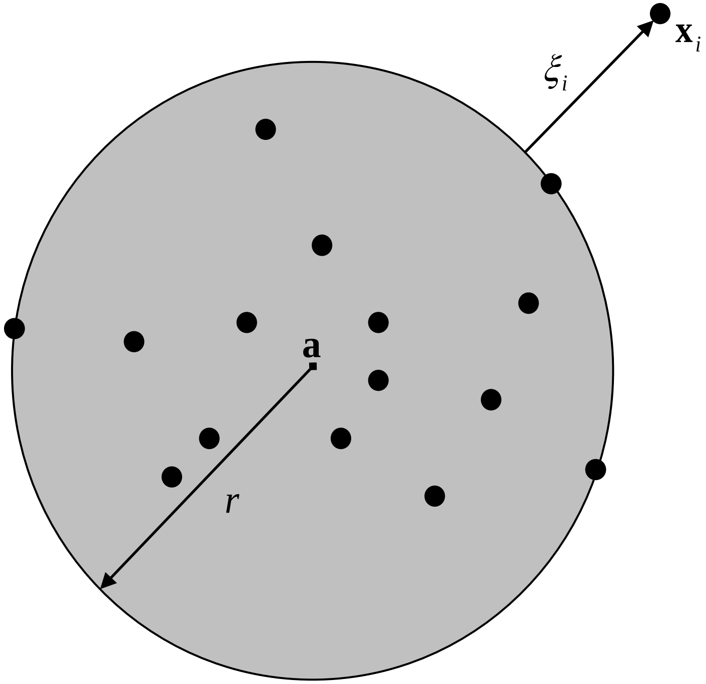


$$L_\varepsilon(y, f(\mathbf{x}, \omega)) = \max\left(|\, y - f(\mathbf{x}, \omega)\,| - \varepsilon, 0\right)$$

Margin based complexity control

- Large degree of falsifiability is achieved by
  - large margin (classification)
  - small epsilon (regression)
- Margin-based methods control complexity independent of problem dimensionality
- The same idea can be used for other learning settings

Single Class Learning



Boundary is specified by a hypersphere with center ***a*** and radius *r.*
An optimal model minimizes the volume of the sphere *and* the total distance of the data points outside the sphere

$$L_r(f(\mathbf{x},\omega)) = \max\left(|\,\mathbf{x}-\mathbf{a})\,| - r, 0\right)$$

ε

Margin-based loss: summary

- Classification: $L_\Delta(y, f(\mathbf{x},\omega)) = \max(\Delta - yf(\mathbf{x},\omega),0)$

  falsifiability controlled by margin Δ

- Regression: $L_\varepsilon(y, f(\mathbf{x},\omega)) = \max(|\, y - f(\mathbf{x},\omega)\,| - \varepsilon,0)$

  falsifiability controlled by ε

- Single class learning: $L_r(f(\mathbf{x},\omega)) = \max(|\,\mathbf{x} - \mathbf{a})\,| - r,0)$

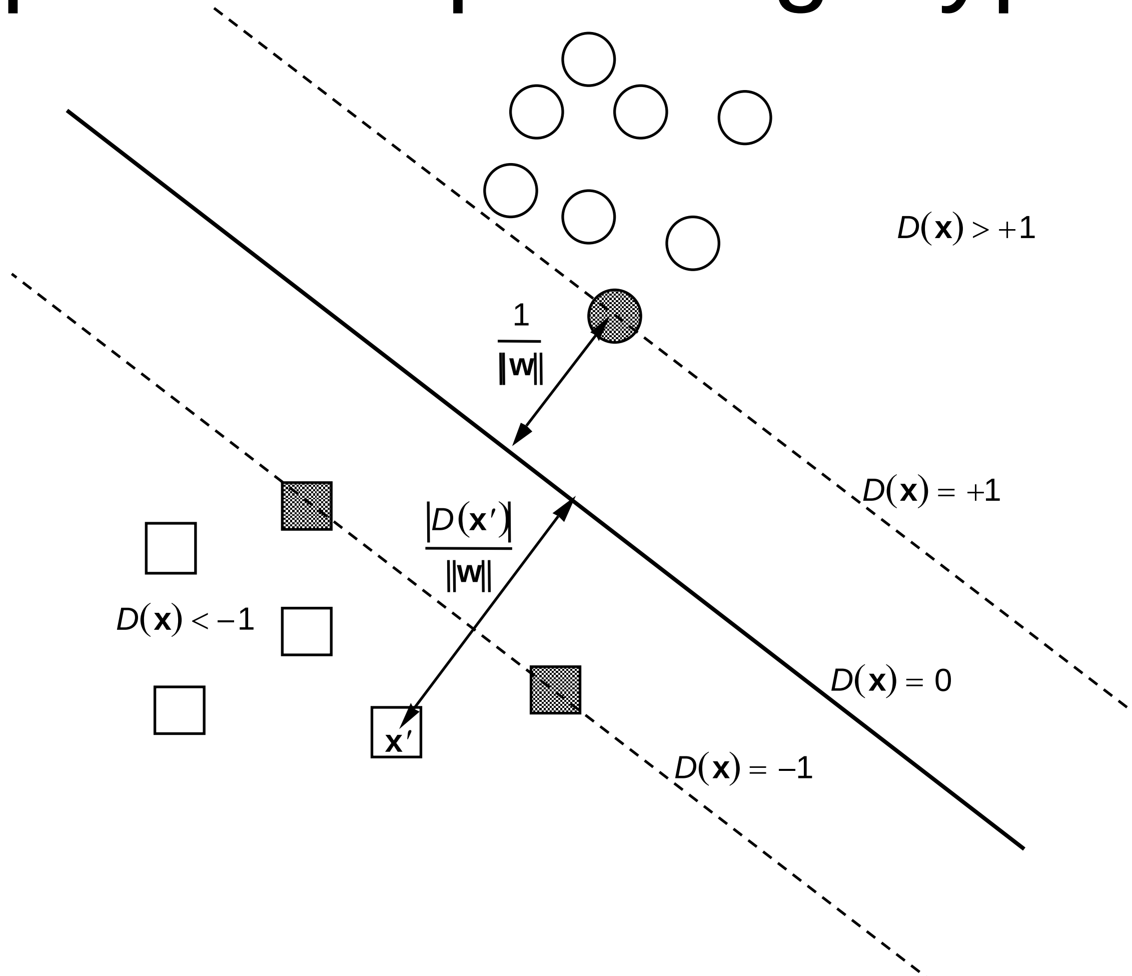
  falsifiability controlled by radius *r*

NOTE: the same interpretation/ motivation for margin-based loss for *different types* of learning problems.

OUTLINE

- Motivation
- Margin-based loss
- **SVM for classification**
  - Linear SVM classifier
  - Inner product kernels
  - Nonlinear SVM classifier
- SVM classification: examples
- Support Vector Regression
- SVM and regularization
- Summary

Optimal Separating Hyperplane



Distance btwn hyperplane and sample $|D(\mathbf{x}')|/\|\mathbf{w}\|$

→ Margin $\Delta = 1/\|\mathbf{w}\|$ **Shaded points are SVs**

Optimization Formulation

- Given training data $(\mathbf{x}_i, y_i) \quad i = 1, ..., n$
- Find parameters $\mathbf{w}, b$ of linear hyperplane

$$D(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{x}) + b$$

that minimize $\eta(\mathbf{w}) = 0.5 \cdot \|\mathbf{w}\|^2$

under constraints $y_i[(\mathbf{w} \cdot \mathbf{x}_i) + b] \geq 1$ correct prediction where y and pre

- Quadratic optimization with linear constraints

tractable for moderate dimensions *d*

- For large dimensions use **dual formulation:**

- scales with *n* rather than *d*

- uses *only dot products*

Optimal Separating Hyperplane

- Given training data $(\mathbf{x}_i, y_i)$ $i = 1,...,n$
- Find parameters $\mathbf{w}, b$ of linear hyperplane

$$D(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{x}) + b$$

- The separating hyperplane satisfy the following constraints

$$\begin{aligned} (\mathbf{w} \cdot \mathbf{x}_i) + b &\geq +\Delta \qquad if\ y_i = +1, \\ (\mathbf{w} \cdot \mathbf{x}_i) + b &\leq -\Delta \qquad if\ y_i = -1, \end{aligned} \qquad i = 1, \ldots, n$$

- Given in terms of one compact equation,

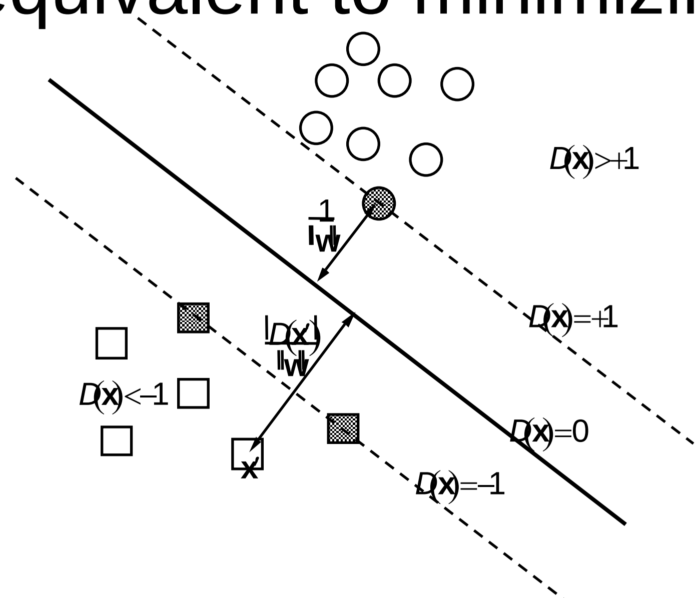
$$y_i[(\mathbf{w} \cdot \mathbf{x}_i) + b] \geq \Delta, \quad i = 1, \ldots, n$$

Optimal Separating Hyperplane

- All training patterns are at least Δ away from the decision boundary and so obey the inequality.

$$\frac{y_i[(\mathbf{w} \cdot \mathbf{x}_i) + b]}{\| \mathbf{w} \|} \geq \Delta, \qquad i = 1, \ldots, n$$

- This inequality implies that maximizing the margin Δ is equivalent to minimizing $\| \mathbf{w} \|$.



Optimal Separating Hyperplane

- Therefore, the form can be formally stated

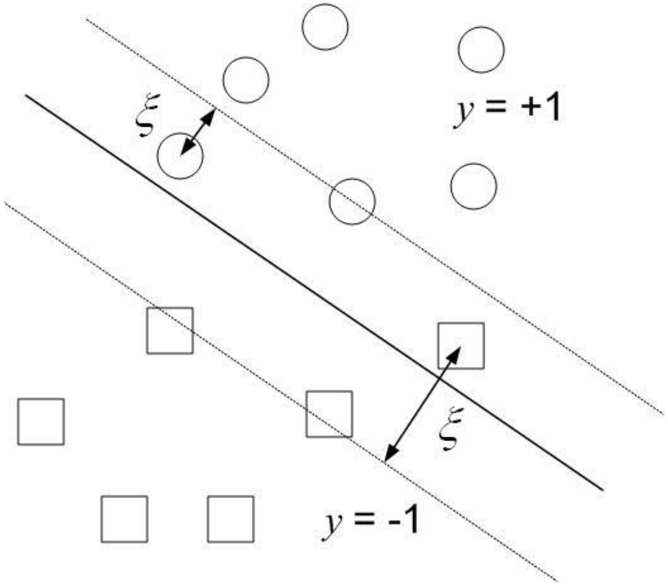
$$\text{Objective Function}: \min_{\beta,b} \left\{ \frac{||\beta^2||}{2} \right\}$$

$$\text{s.t Linear Constraint}: y_i(\beta^T x_i + b) \geq 1 \, , \forall x_i \in D$$

- Considering the non-separable case

$$\text{Objective Function}: \min_{\beta,b,\xi_i} \left\{ \frac{||\beta^2||}{2} + C \sum_{i=1}^{n} (\xi_i)^k \right\}$$

$$\text{s.t Linear Constraint}: y_i(\beta^T x_i + b) \geq 1 - \xi_i, \text{where } \xi_i \geq 0$$



How to solve the constraint optimization problems?

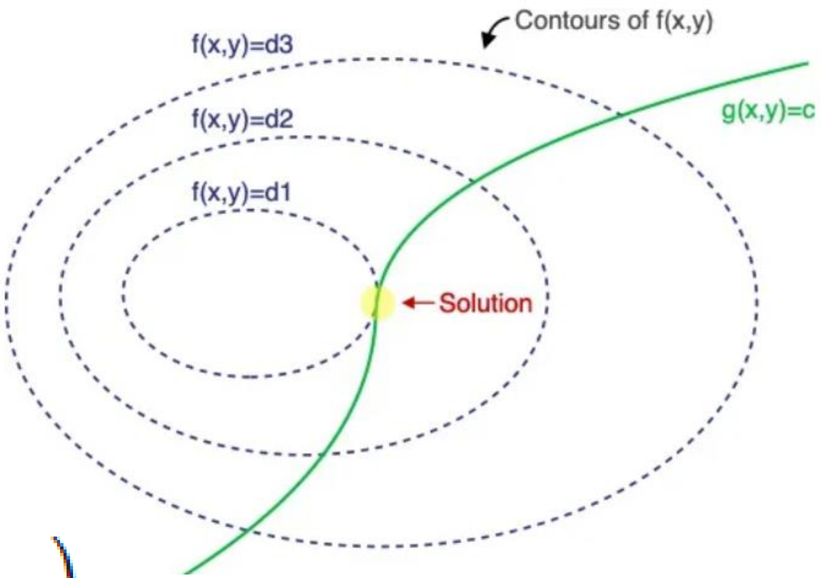
https://www.adeveloperdiary.com/data-science/machine-learning/support-vector-machines-for-beginners-duality-problem/

Lagrange Multiplier

- Solving the constraint optimization

$$\max_{x,y} f(x,y)$$

$$\text{such that } g(x,y) = c$$



- Introduce a new variable α as Lagrange Multiplier => Lagrangian function.

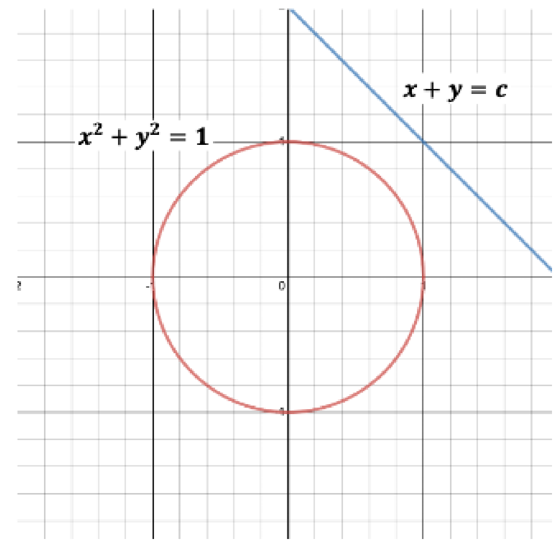
$$L(x,y,\alpha) = f(x,y) - \alpha\Big(g(x,y) - c\Big)$$

- The solution is to find (x,y,α), so that $\Delta L = 0$.

https://www.adeveloperdiary.com/data-science/machine-learning/support-vector-machines-for-beginners-duality-problem/

Lagrange Multiplier - Example

- Find $$\max_{x,y} x + y$$ $$\text{s.t. } x^2 + y^2 = 1$$



- Consider it's Lagrangian

$$L(x,y,\alpha) = f(x,y) - \alpha\Big(g(x,y) - c\Big)$$
$$= x + y - \alpha(x^2 + y^2 - 1)$$

- Take the derivatives

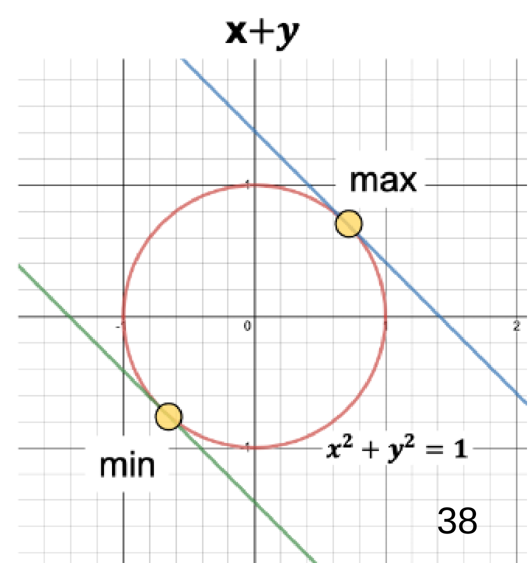
$$\delta_x L(x,y,\alpha) = 1 - 2\alpha x = 0$$
$$x = \frac{1}{2\alpha}$$

$$\delta_y L(x,y,\alpha) = 1 - 2\alpha y = 0$$
$$y = \frac{1}{2\alpha}$$

$$\delta_\alpha L(x,y,\alpha) = x^2 + y^2 - 1 = 0$$
$$\frac{1}{2\alpha}^2 + \frac{1}{2\alpha}^2 - 1 = 0$$

$$\Rightarrow$$

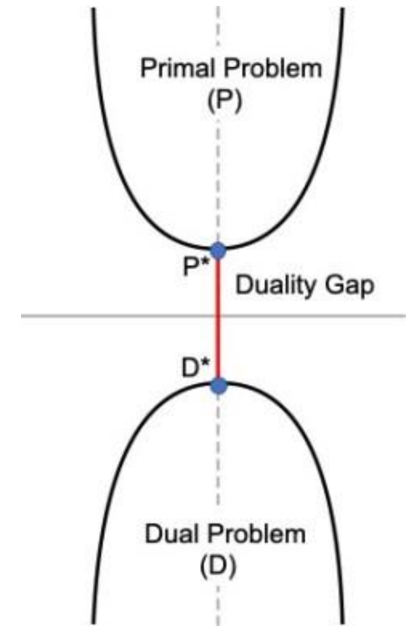
$$\alpha = \pm\frac{1}{\sqrt{2}} \qquad x = \frac{1}{\sqrt{2}}$$
$$y = \frac{1}{\sqrt{2}}$$

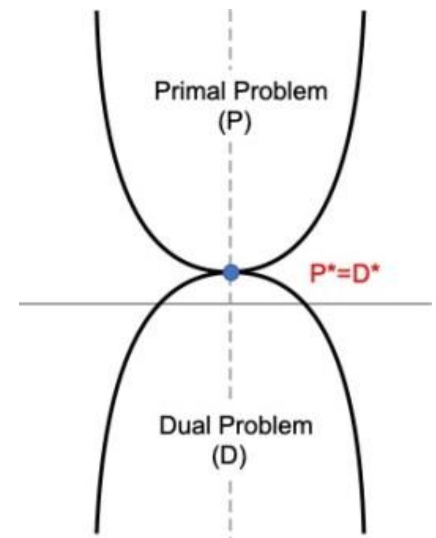


https://www.adeveloperdiary.com/data-science/machine-learning/support-vector-machines-for-beginners-duality-problem/

Duality

- Primal Problem is something that we want to minimize.
- Dual Problem is something we want to maximize.
- The solution to the dual problem provides a lower bound to the solution of the primal (minimization) problem.
- There are some conditions named KKT (Karush–Kuhn–Tucker) condition, needs to hold in order to have P*=D*.





https://www.adeveloperdiary.com/data-science/machine-learning/support-vector-machines-for-beginners-duality-problem/

Optimal Separating Hyperplane

- Back to the original problem

$$\text{Objective Function}: \min_{\beta,b}\left\{\frac{||\beta^2||}{2}\right\}$$

$$\text{s.t Linear Constraint}: y_i(\beta^T x_i + b) \geq 1 \text{ , } \forall x_i \in D$$

- Rewrite the constrain to fit KKT conditions:

$$\text{s.t Linear Constraint}: 1 - y_i(\beta^T x_i + b) \leq 0 \text{ , } \forall x_i \in D \longrightarrow 2$$

- The Lagrangian can defined as following,

$$\begin{aligned} \min L &= \frac{||\beta^2||}{2} + \sum_{i=1}^{n} \alpha_i (1 - y_i(\beta^T x_i + b)) \\ &= \frac{||\beta^2||}{2} - \sum_{i=1}^{n} \alpha_i (y_i(\beta^T x_i + b) - 1) \end{aligned}$$

$$\begin{array}{lr} 1 & \alpha_i g_i(x) = 0 \text{ , } \forall i = 1..n \\ 2 & g_i(x) \leq 0 \text{ , } \forall i = 1..n \\ 3 & \alpha_i \geq 0 \text{ , } \forall i = 1..n \\ 4 & \delta_{x_d} L = 0 \text{ , } \forall d = 1..D \\ 5 & \delta_{\lambda_j} L = 0 \text{ , } \forall j = 1..m \end{array}$$

Strong Duality

KKT Conditions

https://www.adeveloperdiary.com/data-science/machine-learning/support-vector-machines-for-beginners-duality-problem/

Optimal Separating Hyperplane

- Taking derivatives and fit KKT conditions

$$\alpha_i(1 - y_i(\beta^T x_i + b)) = 0 \longrightarrow ① ③$$
$$\text{and } \alpha_i \geq 0$$

$$\delta_\beta L = \beta - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \quad \beta = \sum_{i=1}^{n} \alpha_i y_i x_i \quad \text{and } \delta_b L = \sum_{i=1}^{n} \alpha_i y_i = 0 \longrightarrow ④ ⑤$$

- Plugging these we get the Dual Lagrangian Objective Function,

$$L_{dual} = \frac{||\beta^2||}{2} - \sum_{i=1}^{n} \alpha_i (y_i(\beta^T x_i + b) - 1)$$
$$= \frac{1}{2}\beta^T\beta - \sum_{i=1}^{n} \alpha_i y_i \beta^T x_i - \sum_{i=1}^{n} \alpha_i y_i b + \sum_{i=1}^{n} \alpha_i$$
$$= \frac{1}{2}\beta^T\beta - \beta^T\Big(\sum_{i=1}^{n} \alpha_i y_i x_i\Big) - b\Big(\sum_{i=1}^{n} \alpha_i y_i\Big) + \sum_{i=1}^{n} \alpha_i$$
$$= \frac{1}{2}\beta^T\beta - \beta^T\Big(\beta\Big) - b\Big(0\Big) + \sum_{i=1}^{n} \alpha_i$$
$$= -\frac{1}{2}\beta^T\beta + \sum_{i=1}^{n} \alpha_i$$
$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

- *L* should be minimized w.r.t *β* and *b*, and should be maximized w.r.t α_i

https://www.adeveloperdiary.com/data-science/machine-learning/support-vector-machines-for-beginners-duality-problem/

Optimal Separating Hyperplane

- Hence, instead of minimizing the Primal Problem, we can now maximize the Dual Problem.
- So we can write the following,

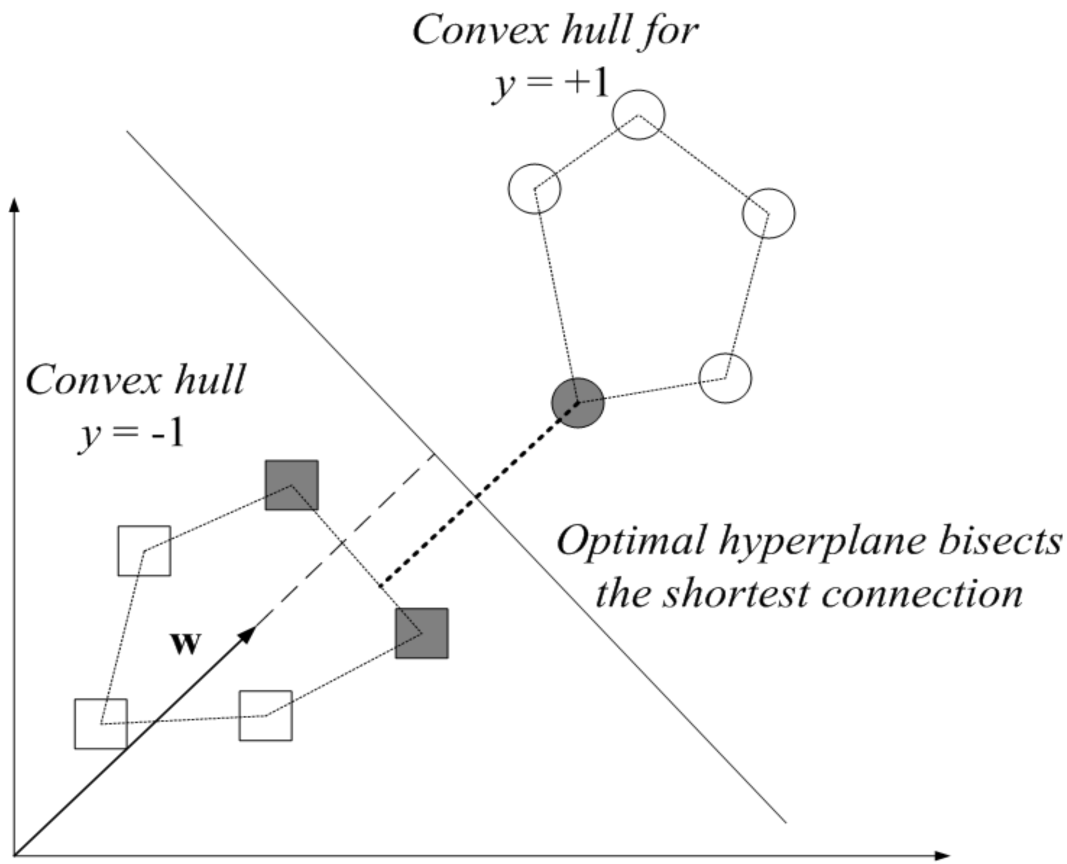
$$\textbf{Objective Function: } \max_{\alpha} L_{dual} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$\textbf{Linear Constraints: } \alpha_i \geq 0, \forall i \in D, \textbf{ and } \sum_{i=1}^{n} \alpha_i y_i = 0$$

- It is a Convex Quadratic programming problem.
- Therefore, it can be solved using standard optimization techniques!

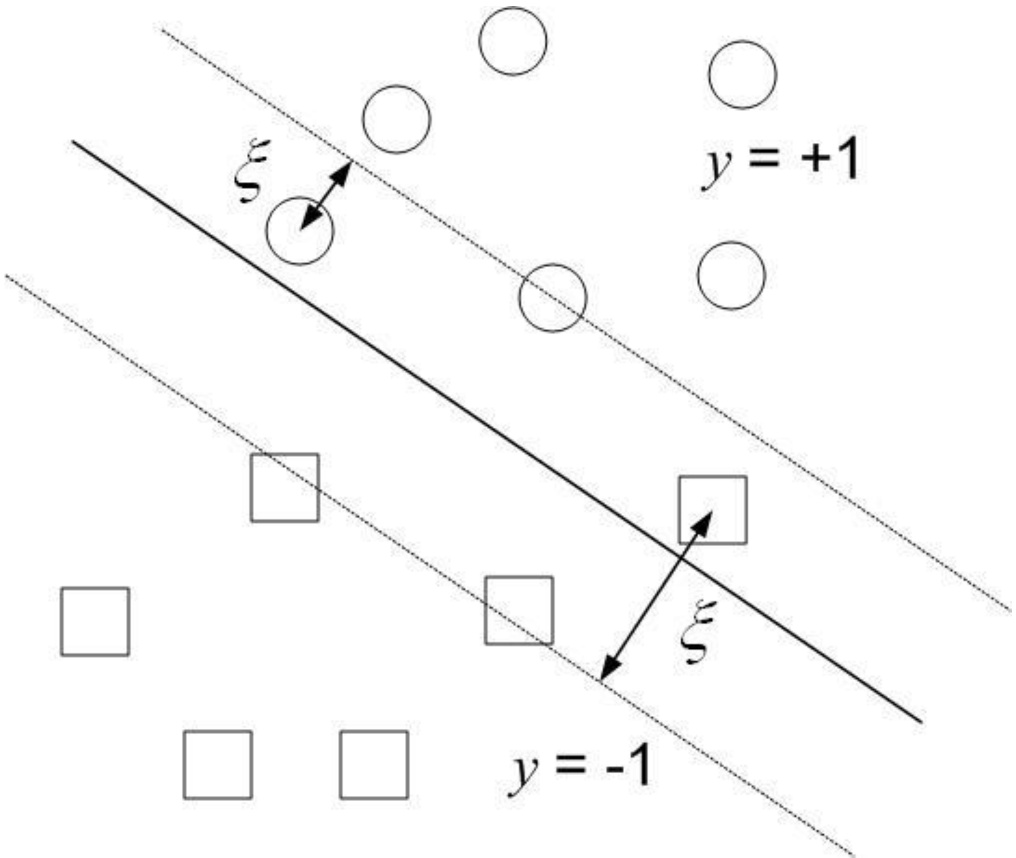
https://www.adeveloperdiary.com/data-science/machine-learning/support-vector-machines-for-beginners-duality-problem/

Convex Hull Interpretation of Dual



Find convex hulls for each class. The closest points to an optimal hyperplane are support vectors

Classification: non-separable data

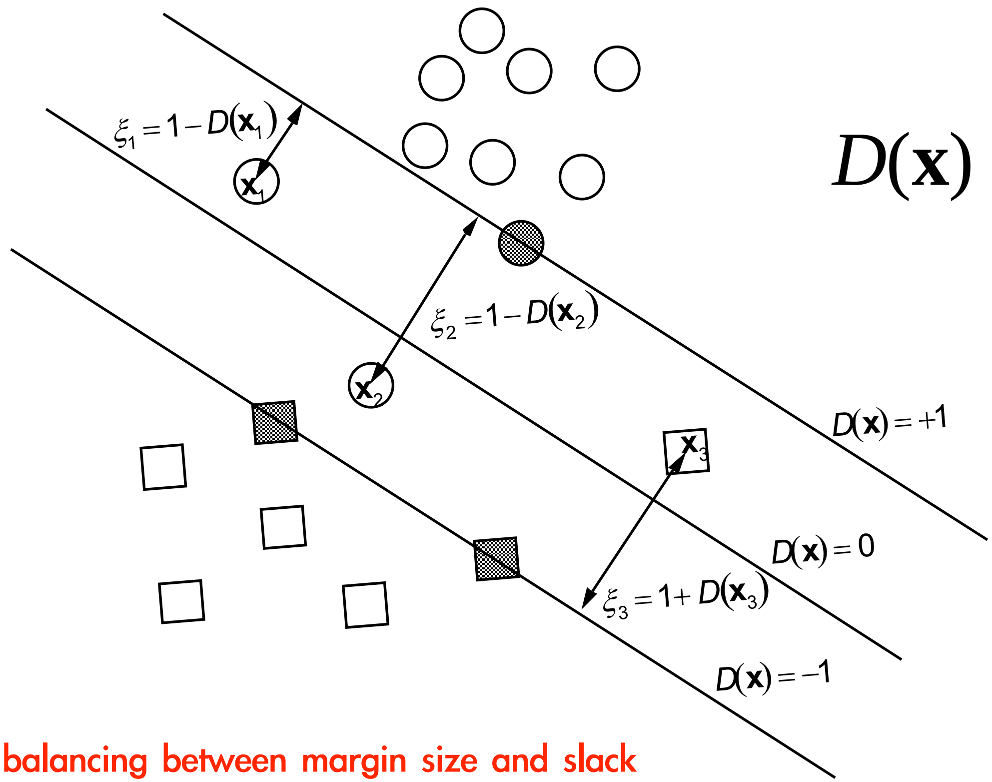


$slack\_variables$

$\xi = \Delta - yf(\mathbf{x}, \omega)$

$$L_{\Delta}(y, f(\mathbf{x}, \omega)) = \max\left(\Delta - yf(\mathbf{x}, \omega), 0\right)$$

SVM for non-separable data



$$D(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{x}) + b$$

for balancing between margin size and slack

Minimize $$\frac{C}{n}\sum_{i=1}^{n}\xi_i + \frac{1}{2}\|\mathbf{w}\|^2 \to \min$$

under constraints $y_i\left[(\mathbf{w} \cdot \mathbf{x}_i) + b\right] \geq 1 - \xi_i$

SVM Dual Formulation

- Given training data $(\mathbf{x}_i, y_i) \quad i = 1, \ldots, n$
- Find parameters α_i^*, b^* of an opt. hyperplane

$$D(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i^* y_i (\mathbf{x} \cdot \mathbf{x}_i) + b^*$$

as a solution to maximization problem

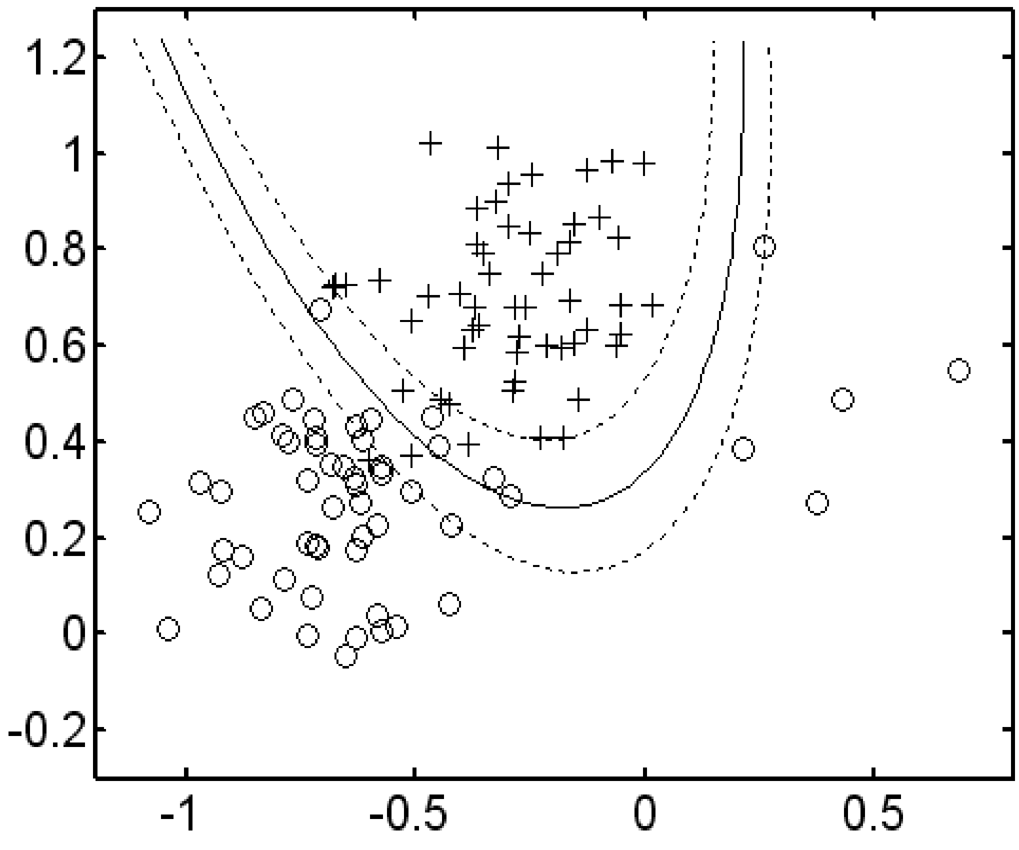
$$\mathrm{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \rightarrow \max$$

under constraints $\sum_{i=1}^{n} y_i \alpha_i = 0, \quad 0 \le \alpha_i \le C/n$

- *Note:* data samples with nonzero α_i^* are SVs
  free/unbounded $0 < \alpha_i < C/n$ and *bounded* SVs
- Formulation requires only inner products $(\mathbf{x} \cdot \mathbf{x}')$

Nonlinear Decision Boundary

- Fixed (linear) parameterization is too rigid
- Nonlinear curved margin may yield **larger margin** (falsifiability) *and* **lower error**



Nonlinear Mapping via Kernels

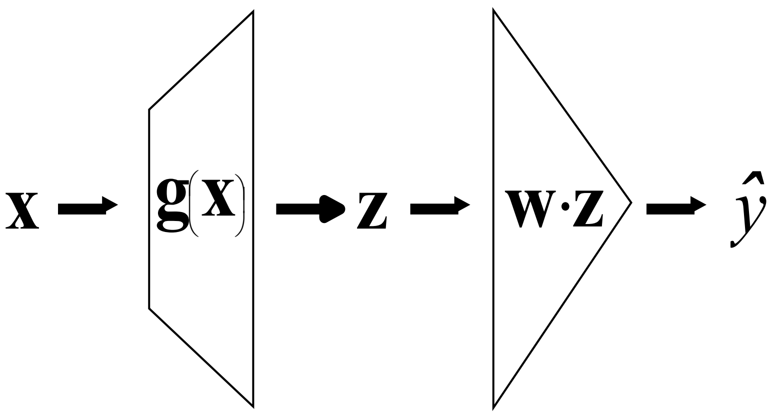
Nonlinear *f(**x**,w)* + margin-based loss = SVM

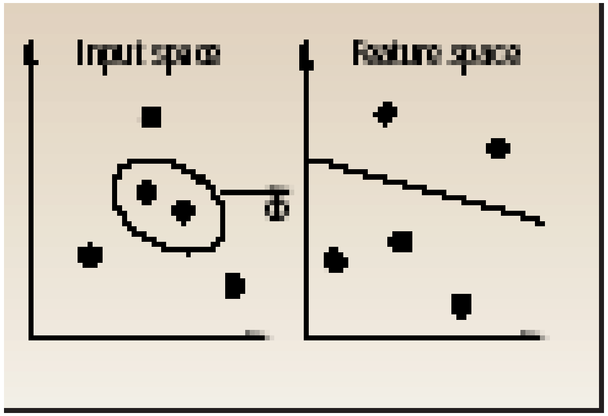
- Nonlinear mapping to feature **z** space, i.e.

$$\mathbf{x} \sim (x_1, x_2) \rightarrow \mathbf{z} \sim (1, x_1, x_2, x_1 x_2, x_1^2, x_2^2)$$

- Linear in z-space=nonlinear in x-space
- BUT $(\mathbf{z}\cdot\mathbf{z}') = H(\mathbf{x},\mathbf{x}')$ ~ kernel trick

→ Compute dot product via kernel analytically





SVM Formulation (with kernels)

- Replacing $(\mathbf{z}\cdot\mathbf{z}')\rightarrow H(\mathbf{x},\mathbf{x}')$ leads to:
- Find parameters α_i^*, b^* of an optimal hyperplane $D(\mathbf{x})=\sum_{i=1}^{n}\alpha_i^* y_i H(\mathbf{x}_i,\mathbf{x})+b^*$

  as a solution to maximization problem

$$\mathrm{L}(\alpha)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j H(\mathbf{x}_i,\mathbf{x}_j)\rightarrow \max$$

  under constraints $\sum_{i=1}^{n} y_i\alpha_i=0,\quad 0\le\alpha_i\le C/n$
- *Given:* the training data $(\mathbf{x}_i, y_i)\quad i=1,\ldots,n$

  an inner product kernel $H(\mathbf{x},\mathbf{x}')$

  regularization parameter *C*

Examples of Kernels

Kernel $H(\mathbf{x}, \mathbf{x}')$ is a symmetric function satisfying general (Mercer's) conditions

Examples of kernels for different mappings **x→z**

- Polynomials of degree *q*

$$H(\mathbf{x}, \mathbf{x}') = [(\mathbf{x} \cdot \mathbf{x}') + 1]^q$$

- RBF kernel

$$H(\mathbf{x}, \mathbf{x}') = \exp\left\{ -\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{\sigma^2} \right\}$$

- Neural Networks $\quad H(\mathbf{x}, \mathbf{x}') = \tanh[v(\mathbf{x} \cdot \mathbf{x}') + a]$

  for given parameters $\quad v, a$

Automatic selection of the number of hidden units (SV's)

More on Kernels

- The kernel matrix has all info (data + kernel)

  H(1,1) H(1,2)…….H(1,n)

  H(2,1) H(2,2)…….H(2,n)

  ………………………….

  H(n,1) H(n,2)…….H(n,n)

- Kernel defines a distance in some feature space (aka kernel-induced feature space)
- Kernels can incorporate apriori knowledge
- Kernels can be defined over complex structures (trees, sequences, sets etc)

Kernel Terminology

- The term kernel is used in 3 contexts:

  - nonparametric density estimation

  - equivalent kernel representation for linear least squares regression

  - SVM kernels

- SVMs are often called kernel methods
- Kernel trick can be used with *any classical linear method*, to yield a nonlinear method:

  For example, ***ridge regression*** + ***kernel*** → LS SVM

Support Vectors

- SV's ~ training samples with non-zero loss
- SV's are samples that falsify the model
- The model depends only on SVs

→ SV's ~ robust characterization of the data

WSJ Feb 27, 2004:

About 40% of us (Americans) will vote for a Democrat, even if the candidate is Genghis Khan. About 40% will vote for a Republican, even if the candidate is Attila the Han. This means that the election is left in the hands of one-fifth of the voters.

- SVM Generalization ~ data compression

New insights provided by SVM

- Why linear classifiers can *generalize*?

$$h \le \min(R^2/\Delta^2, d) + 1$$

  (1) Margin is *large* (relative to *R*)

  (2) % of SV's is *small*

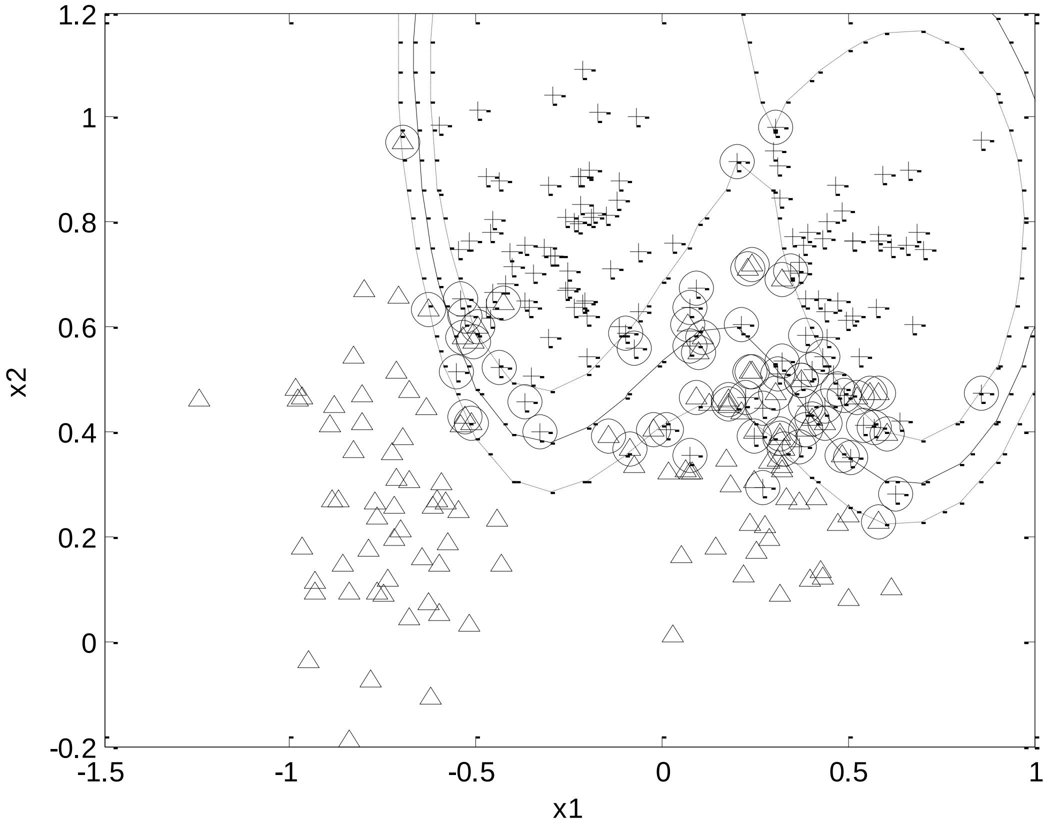
  (3) ratio *d/n* is *small*

- SVM offers an effective way to control complexity (via margin + kernel selection) i.e. implementing (1) or (2) or both
- Requires common-sense parameter tuning

OUTLINE

- Motivation
- Margin-based loss
- SVM for classification
- **SVM classification: examples**
- Support Vector Regression
- SVM and Regularization
- Summary

RBF SVM for Ripley's Data Set



- No. of Training samples = 250
- No. of Test samples = 1,000
- Model selection via 10-fold cross-validation
- **Test error 9.8%**

Details of Model Selection

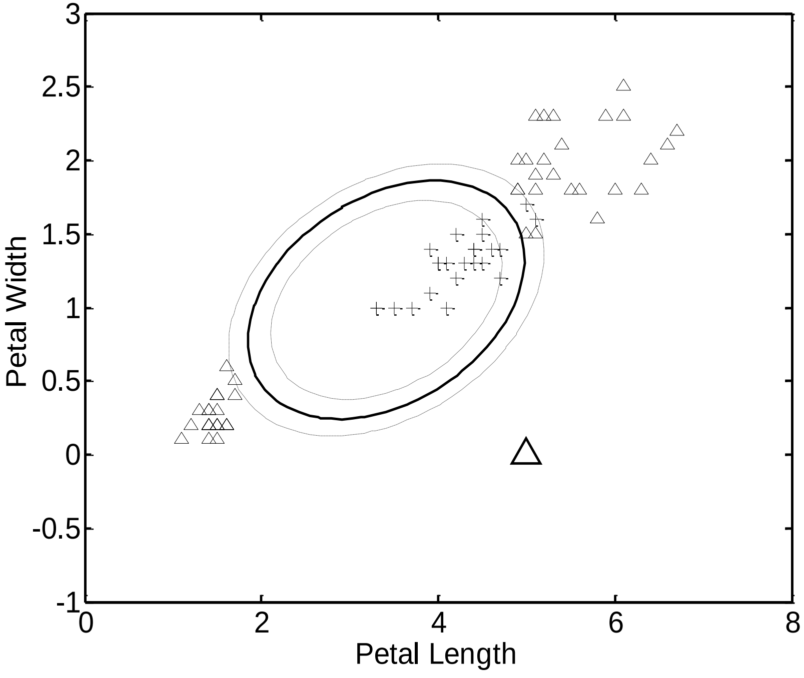
- RBF kernel $K(\mathbf{u},\mathbf{v}) = \exp\left(-\gamma \left|\mathbf{u}-\mathbf{v}\right|^2\right)$
- Two tuning parameters: *C* and *gamma (RBF parameter)*
- Optimal values selected via 10-fold cross-validation
- *Note* **log scale** for parameter values

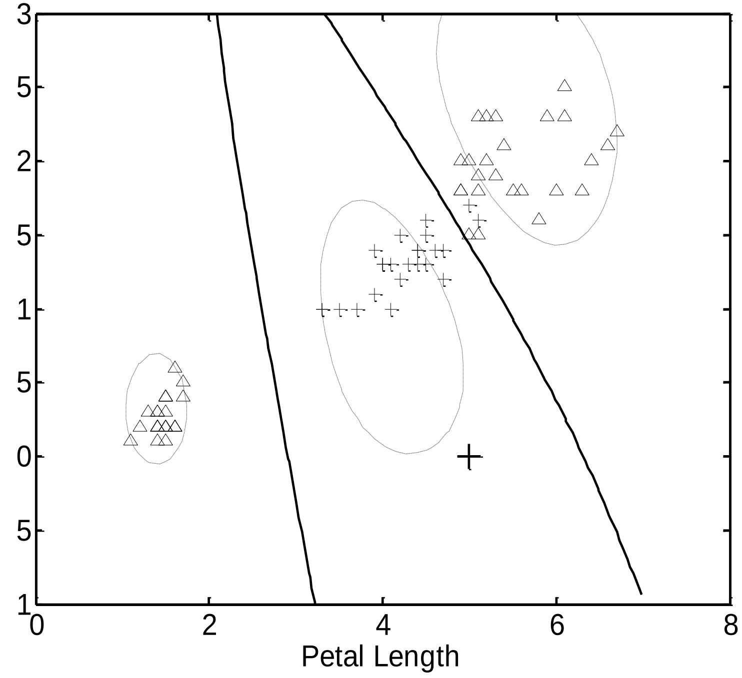
gamma	C= 0.1	C= 1	C= 10	C= 100	**C= 1000**	C= 10000
$=2^{-3}$	98.4%	23.6%	18.8%	20.4%	18.4%	14.4%
$=2^{-2}$	51.6%	22%	20%	20%	16%	14%
$=2^{-1}$	33.2%	19.6%	18.8%	15.6%	13.6%	14.8%
$=2^{0}$	28%	18%	16.4%	14%	**12.8%**	15.6%
$=2^{1}$	20.8%	16.4%	14%	12.8%	16%	17.2%
$=2^{2}$	19.2%	14.4%	13.6%	15.6%	15.6%	16%
$=2^{3}$	15.6%	14%	15.6%	16.4%	18.4%	18.4%

IRIS Data Set

- Modified Iris data set:
  - two input variables, petal length and petal width
  - two classes: *iris versicolor* (+) vs *not_versicolor* (Δ)

- Two SVM models: **polynomial kernel** and **RBF kernel**





Many challenging applications

- Mimic human recognition capabilities
  - high-dimensional data
  - content-based
  - context-dependent
- Example: read the sentence

  Sceitnitss osbevred: it is nt inptrant how lteters are msspled isnide the word. It is ipmoratnt that the fisrt and lsat letetrs do not chngae, tehn the txet is itneprted corrcetly
- SVM is suitable for sparse high-dimensional representations

Example SVM Applications

- Handwritten digit recognition
- Face detection in unrestricted images
- Text/ document classification
- Image classification and retrieval
- …….

Handwritten Digit Recognition (mid-90's)

- Data set:

  postal images (zip-code), segmented, cropped;

  ~ 7K training samples, and 2K test samples
- Data encoding:

  16x16 pixel image → 256-dim. vector
- Original motivation: Compare SVM with custom MLP network (LeNet) designed for this application
- Multi-class problem: one-vs-all approach

  → 10 SVM classifiers (one per each digit)

Digit Recognition Results

- Summary
  - prediction accuracy better than custom NN's
  - accuracy does not depend on the kernel type
  - 100 – 400 support vectors per class (digit)
- More details

Type of kernel	**No. of Support Vectors**	**Error%**
Polynomial	274	4.0
RBF	291	4.1
Neural Network	254	4.2

- ~ 80-90% of SV's coincide (for different kernels)
- Reduced-set SVM (Burges, 1996) ~ 15 per class

Face detection in images (Osuna, et al, 1997)

- **Goal**: Detect and locate face(s) in an image
- **Training data**: face and non-face 19X19 pixel images obtained from real data
- **Test**: Set A 313 high-quality images with 1 face

  Set B 23 mixed quality images (multiple faces)
- **Preprocessing**: illumination gradient correction, histogram equalization
- **Face Search Strategy**: divide an image into overlapping sub-images, at multiple scales, and classify each sub-image using SVM
- **Multiple scales** of the original image are used

  → 5 million sub-images (for Set A)

Face detection: results

- SVM Classifier:
  2-nd order polynomial

- Performance Results

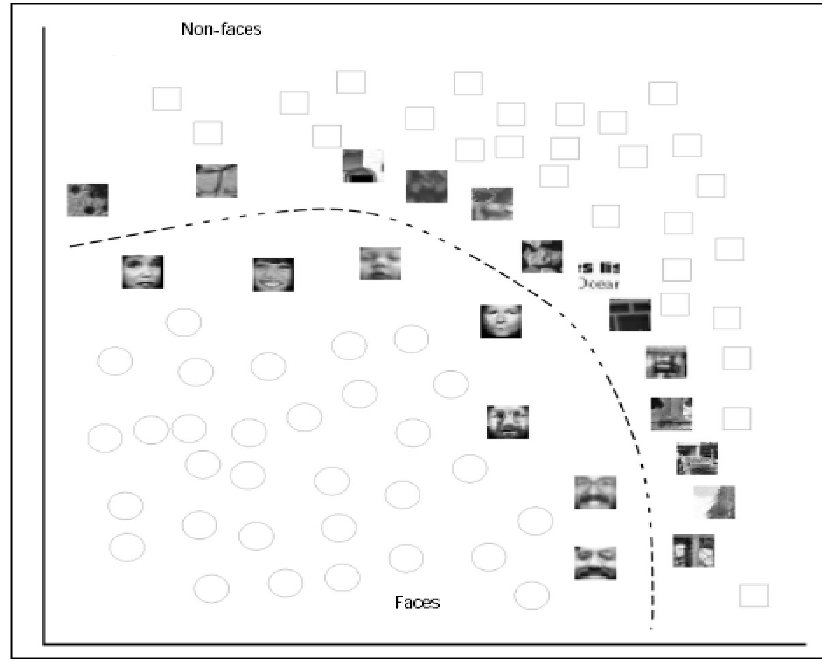


Figure 6. Geometrical interpretation of how the SVM separates the face and nonface classes. The patterns are real support vectors obtained after training the system. Notice the small number of total support vectors and the fact that a higher proportion of them correspond to nonfaces.

	Detection rate	False Alarm
Test Set A	97%	4%
Test Set B	74%	20%

Document Classification (Joachims, 1998)

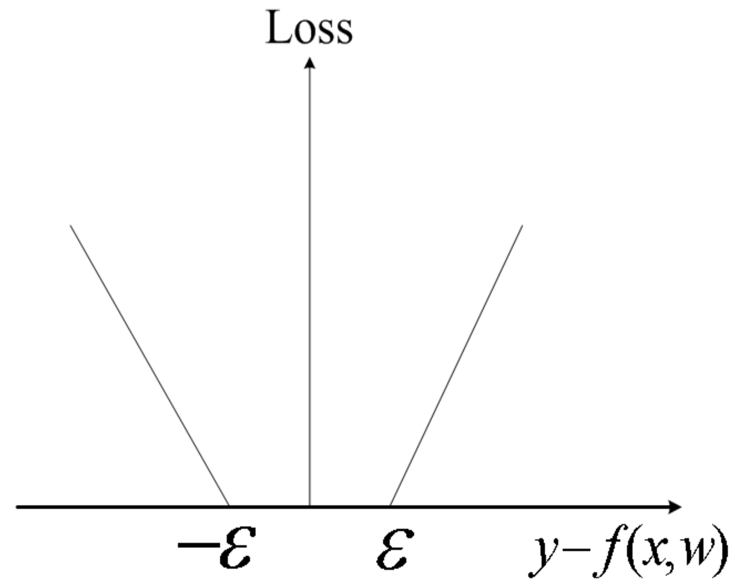
- **The Problem:** Classification of text documents in large data bases, for text indexing and retrieval
- **Traditional approach**: human categorization (i.e. via feature selection) – relies on a good indexing scheme. This is time-consuming and costly
- **Predictive Learning Approach (SVM):** construct a classifier using all possible features (words)
- **Document/ Text Representation**:

  individual words = input features (possibly weighted)
- SVM performance:
  - Very promising (~ 90% accuracy vs 80% by other classifiers)
  - Most problems are linearly separable → use linear SVM

OUTLINE

- Introduction and motivation
- Margin-based loss
- SVM for classification
- SVM classification: examples
- **Support Vector Regression**
- SVM and regularization
- Summary

Linear SVM regression



Assume linear parameterization

$$f(\mathbf{x},\omega) = \mathbf{w} \cdot \mathbf{x} + b$$

SVM regression functional:

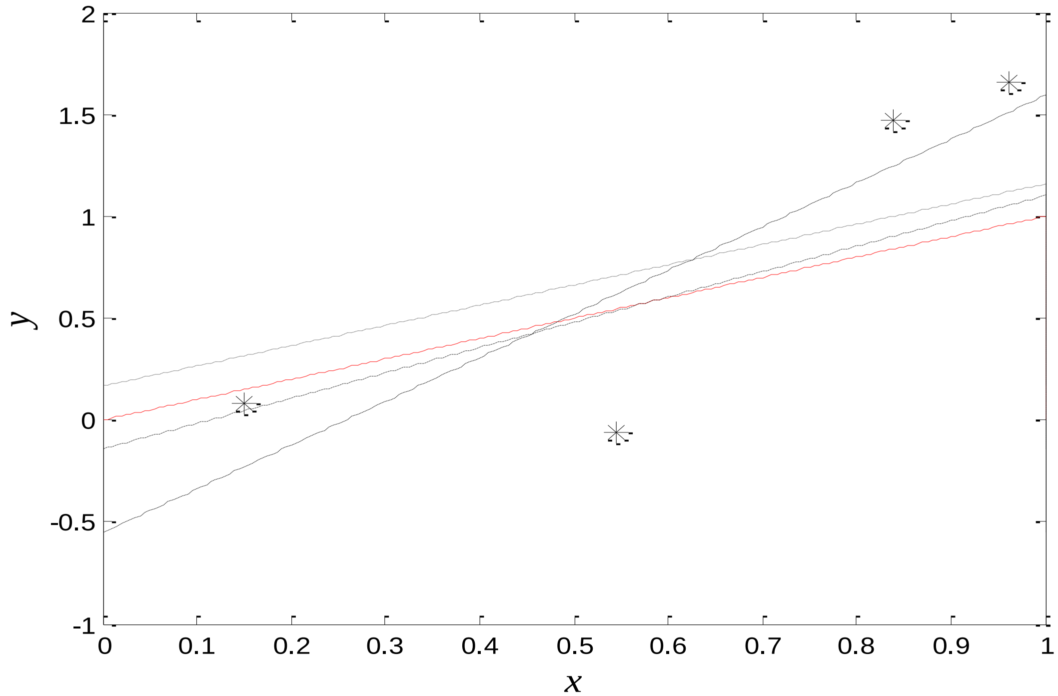
$$R_{SVM}(\mathbf{w}, b, \mathbf{Z}_n) = \frac{1}{2}\|\mathbf{w}\|^2 + C \cdot R_{emp}(\omega, \mathbf{Z}_n) \rightarrow \min$$

where $$R_{emp}(\omega, \mathbf{Z}_n) = \frac{1}{n}\sum_{i=1}^{n} L_\varepsilon(y_i, f(\mathbf{x}_i, \omega))$$

$$L_\varepsilon(y, f(\mathbf{x},\omega)) = \max\left(| y - f(\mathbf{x},\omega) | - \varepsilon, 0\right)$$

Loss Functions for Regression

- Small data set (4 samples): $y = x + \delta \qquad \delta \sim N(0, 0.36)$
- Input samples uniform in [0, 1]
- Loss functions: squared, least-modulus, SVM $\varepsilon = 0.6$
- Estimates shown: dotted, dashed, and dashed-dotted



Comparison of Loss Functions (cont'd)

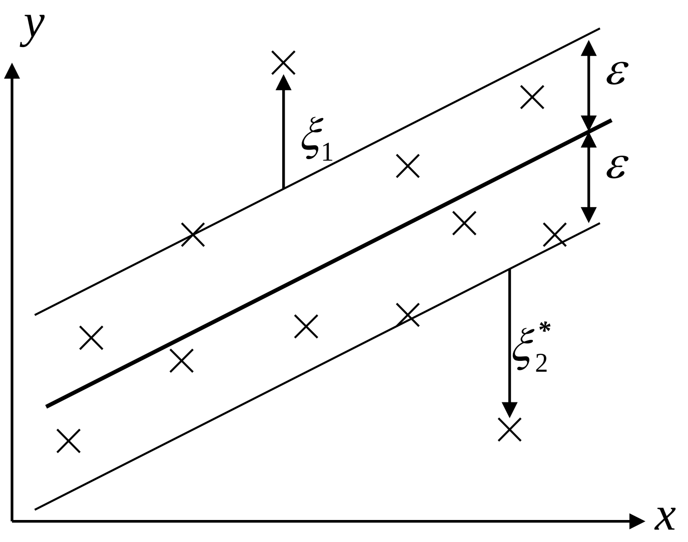
- Small data set (4 samples): $y = x + \delta$ $\delta \sim N(0, 0.36)$
- Input samples uniform in [0, 1]
- MSE test error for 5 realizations of training data:

	Squared loss	*Least modulus loss*	*SVM loss with epsilon=0.6*
1	0.024	0.134	0.067
2	0.128	0.075	0.063
3	0.920	0.274	0.041
4	0.035	0.053	0.032
5	0.111	0.027	0.005
Mean	0.244	0.113	0.042
St. Deviation	0.381	0.099	0.025

Direct Optimization Formulation

Given training data

$$\left(\mathbf{x}_i, y_i\right) \qquad i = 1,\dots,n$$



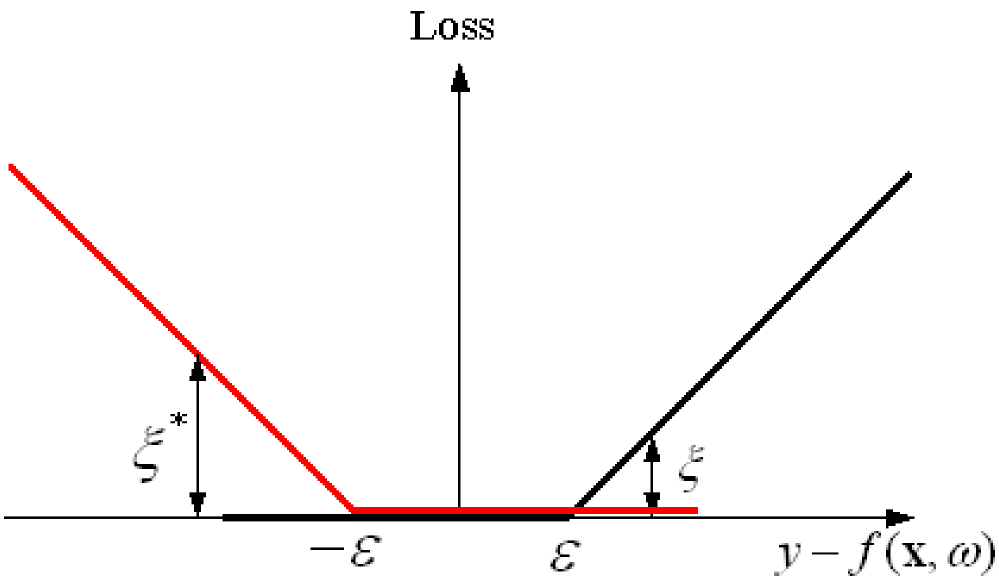
Minimize

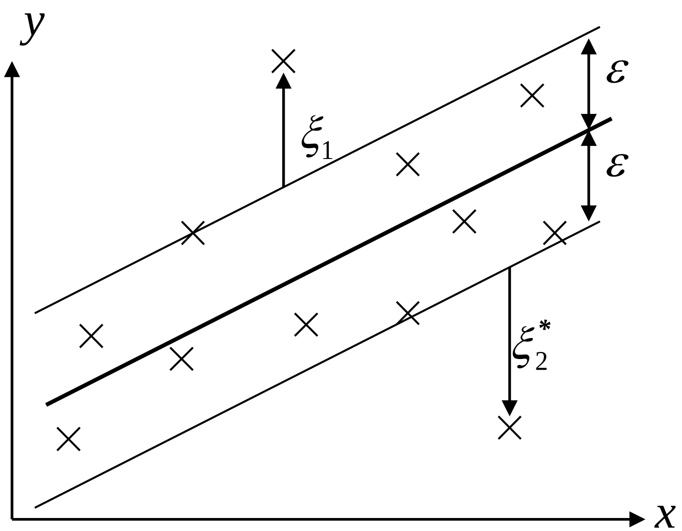
$$\frac{1}{2}(\mathbf{w}\cdot\mathbf{w}) + \frac{C}{n}\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$

Under constraints

$$\begin{cases} y_i - (\mathbf{w}\cdot\mathbf{x}_i) - b \le \varepsilon + \xi_i \\ (\mathbf{w}\cdot\mathbf{x}_i) + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0, i = 1,\dots,n \end{cases}$$

Connection to Hinge Loss





ε-insensitive loss ~ sum of two hinge loss functions with margin borders at +/- ε

Dual Formulation for SVM Regression

Given training data $\left(\mathbf{x}_i, y_i\right) \quad i = 1,\ldots,n$

And the values of ε, C

Find coefficients $\alpha_i^*, \beta_i^*, i = 1,\ldots,n$ which maximize

$$\mathrm{L}(\alpha_i, \beta_i) = -\varepsilon \sum_{i=1}^{n} (\alpha_i + \beta_i) + \sum_{i=1}^{n} y_i (\alpha_i - \beta_i) - \frac{1}{2} \sum_{i,j=1}^{n} (\alpha_i - \beta_i)(\alpha_j - \beta_j)(\mathbf{x}_i \cdot \mathbf{x}_j)$$

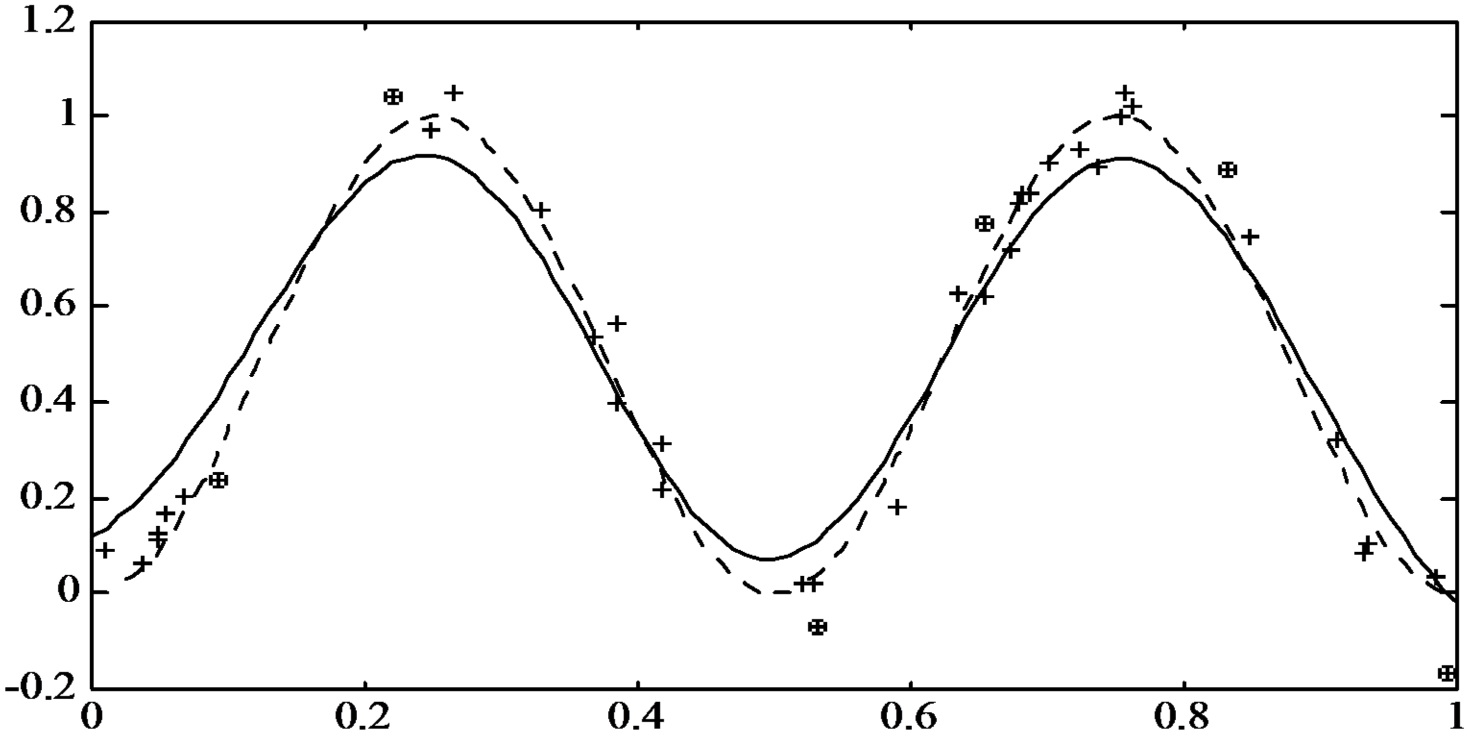
Under constraints

$$\sum_{i=1}^{n} \alpha_i = \sum_{i=1}^{n} \beta_i$$

$$0 \le \alpha_i \le C/n$$

$$0 \le \beta_i \le C/n, i = 1,\ldots,n$$

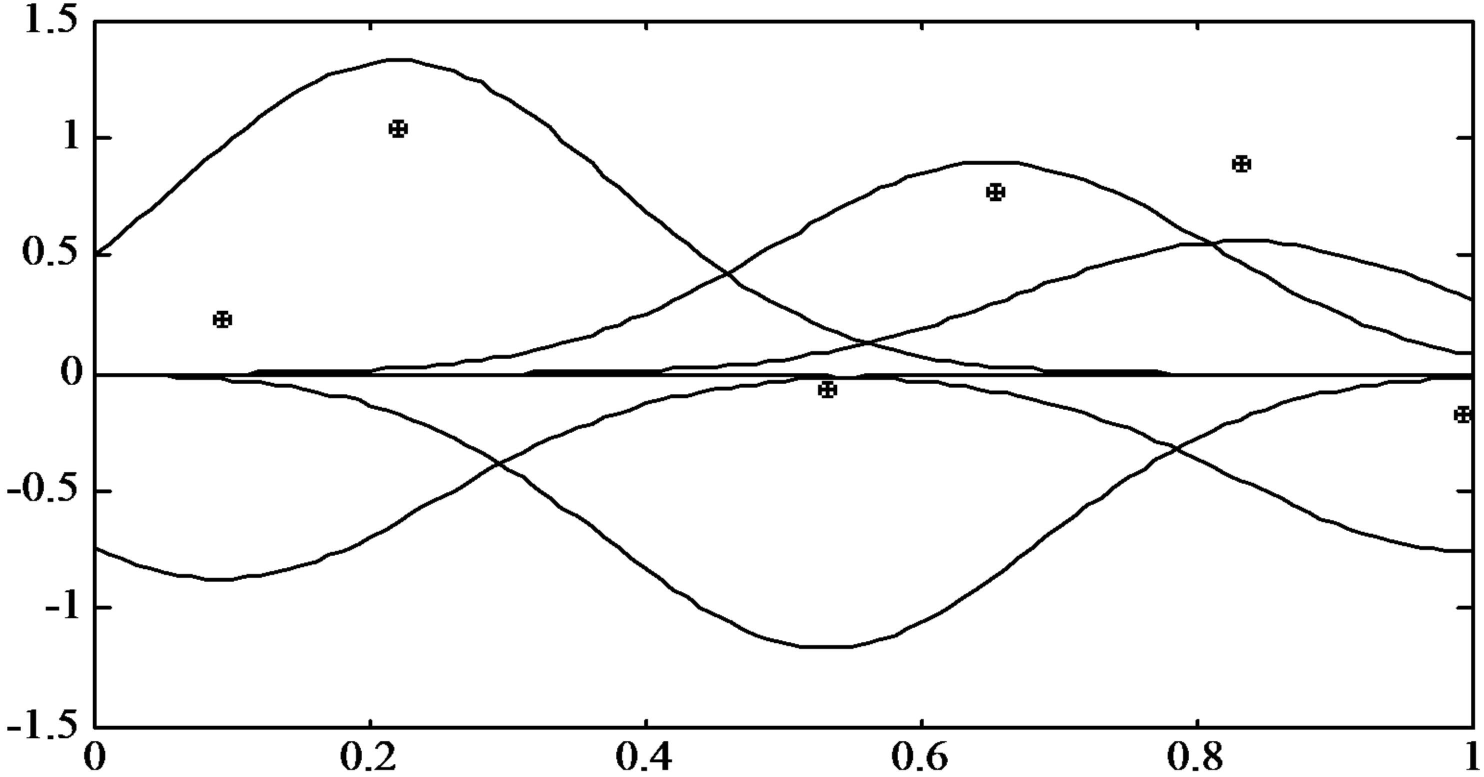
Example: regression using RBF kernel



RBF estimate (dashed line) using $\varepsilon = 0.16, C = 2000$

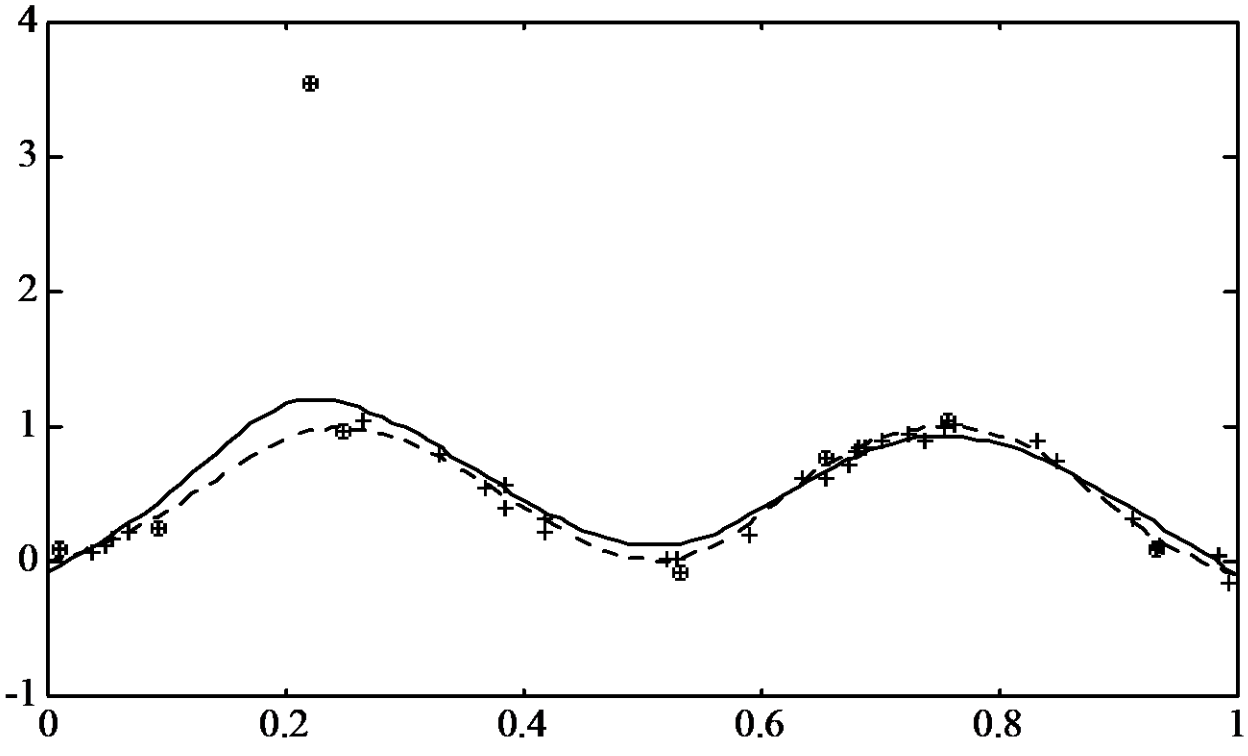
SVM model uses only 6 SV's (out of the 40 points)

Example: decomposition of RBF model



Weighted sum of 6 RBF kernel fcts gives the SVM model

Robustness of SVM regression



Solid curve: SVM estimate with the outlier
Dashed curve: SVM estimate without the outlier

OUTLINE

- Motivation
- Margin-based loss
- SVM for classification
- SVM classification: examples
- Support Vector Regression
- **SVM and regularization**
- Summary

SVM and Regularization

- **SVM →** implements risk minimization (SRM) approach ~ system imitation
- **Regularization →** for ill-posed function interpolation problems ~ system identification
- But their risk functionals 'look similar'

$$R_{SVM}(\mathbf{w}, b) = C\sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i, \omega)) + \frac{1}{2}\|\mathbf{w}\|^2$$

$$R_{reg}(\mathbf{w}, b) = \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i, \omega))^2 + \lambda\|\mathbf{w}\|^2$$

→Many claims:

SVM = special case of regularization

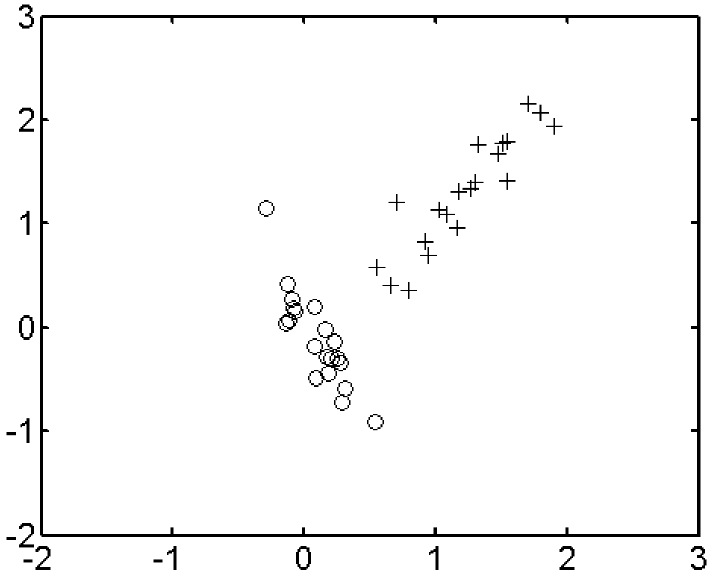
SVM and Regularization (cont'd)

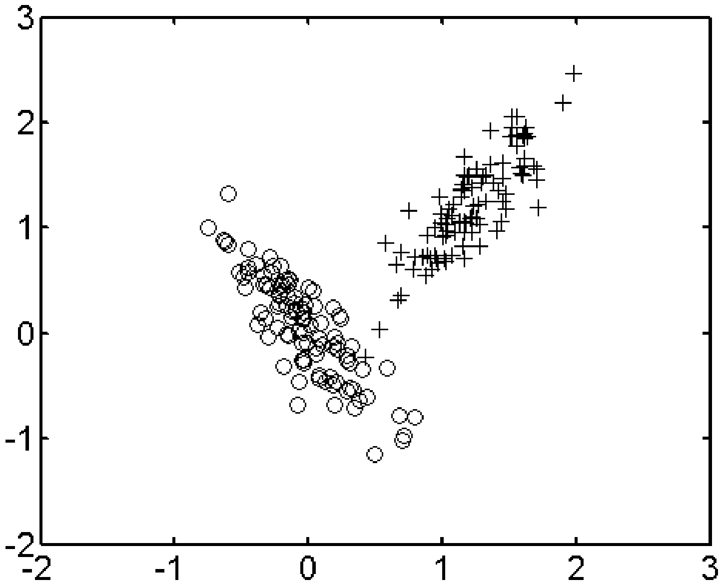
- Regularization/Penalization was developed under function approximation setting where only an asymptotic theory can be developed
- Well-known and widely used for low-dimensional problems since 1960's (splines)

  Ripley (1996): Since splines are so useful in one dimension, they might appear to be the obvious methods in more. In fact, they appear to be restricted and little used.
- The role of margin-based loss

Comparison for Classification

- Linear SVM vs Penalized LDA – comparison is fair
- Data Sets: small (20 samples per class)
  large (100 samples per class)





Comparison results: classification

- Small sample size:

  Linear SVM yields 0.5% - 1.1% error rate

  Penalized LDA yields 2.8% - 3% error

- Large sample size:

  Linear SVM yields 0.4% - 1.1% error rate

  Penalized LDA yields 1.1% - 2.2% error

- Conclusion: margin based complexity control is better than regularization

Comparison for regression

- Linear SVM vs linear ridge regression (RR)

Note: Linear SVM has 2 tunable parameters

- Sparse Data Set:

30 noisy samples, using target function

$$t(\mathbf{x}) = 2x_1 + x_2 + 0 \cdot x_3 + 0 \cdot x_4 + 0 \cdot x_5 \qquad \mathbf{x} \in [0,1]^5$$

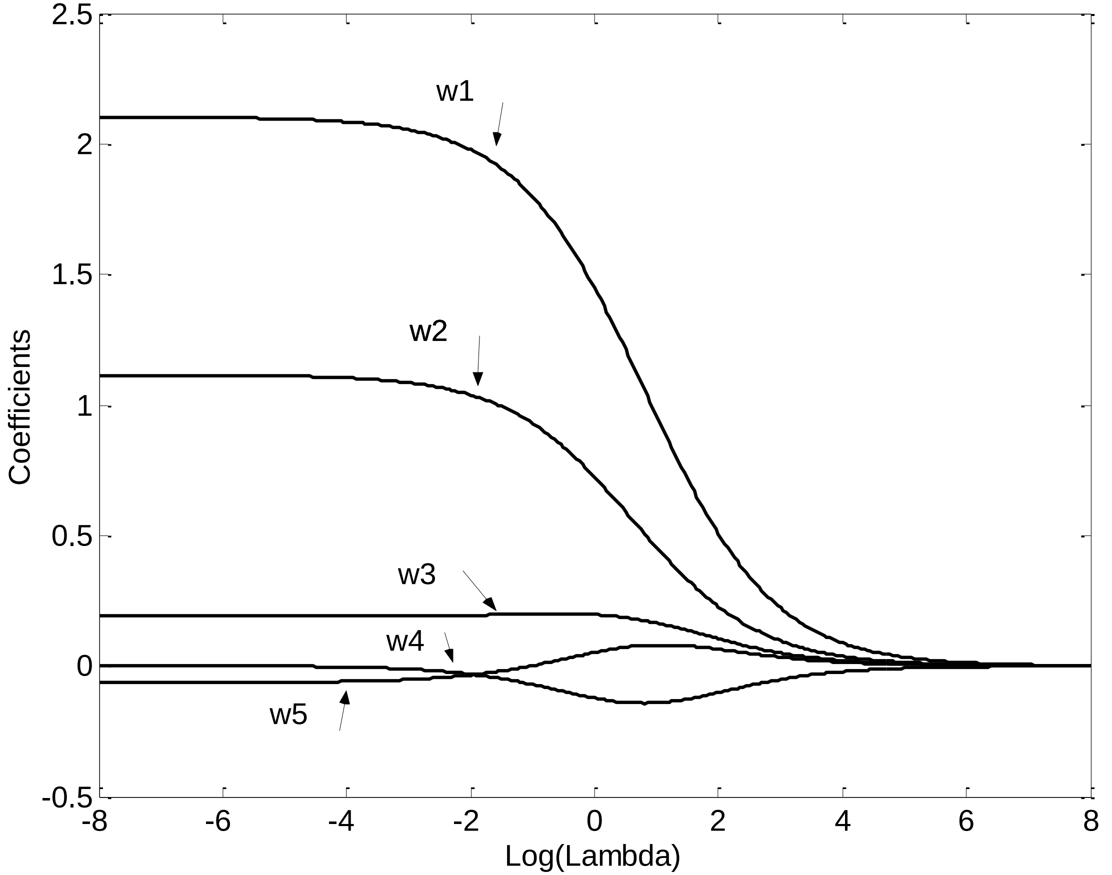
corrupted with gaussian noise with $\sigma = 0.2$

- **Complexity Control:**

  - for RR vary regularization parameter
  - for SVM ~ epsilon and C parameters

Complexity control for ridge regression

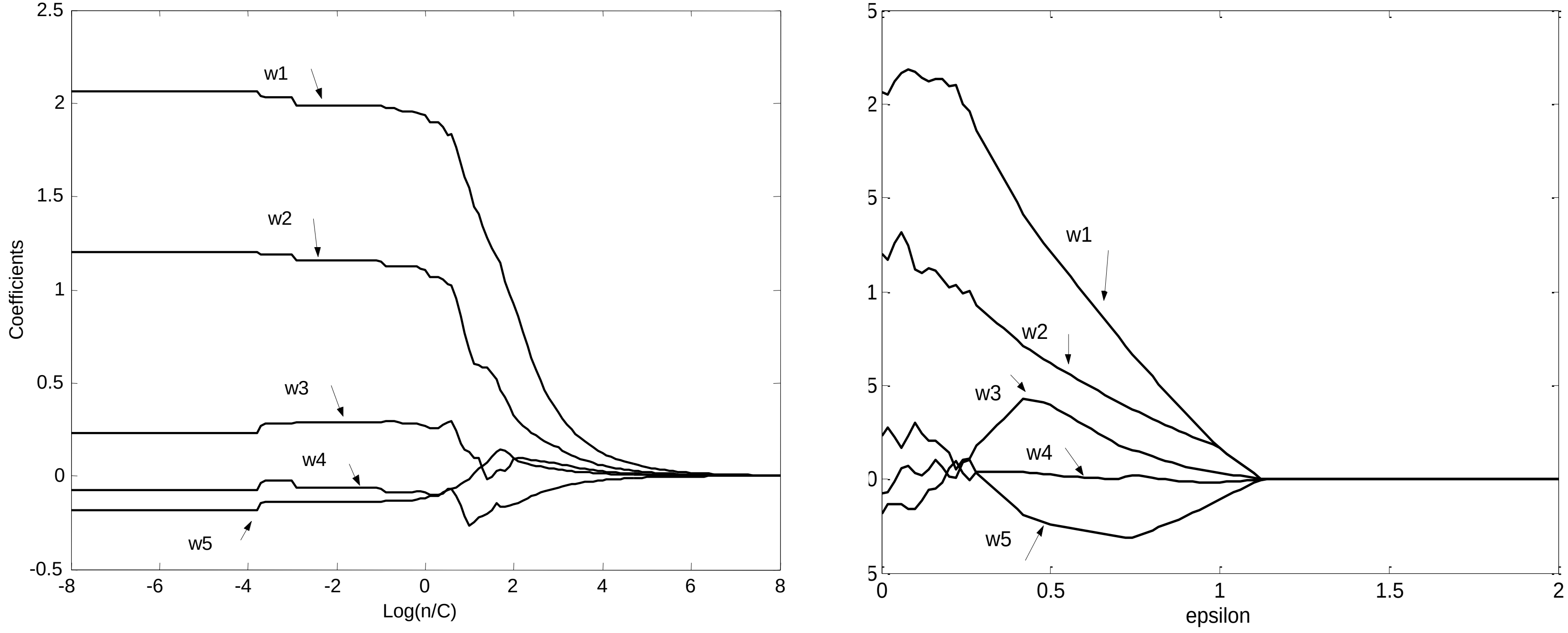
- Coefficient shrinkage for ridge regression



Complexity control for SVM

- Coefficient shrinkage for SVM:

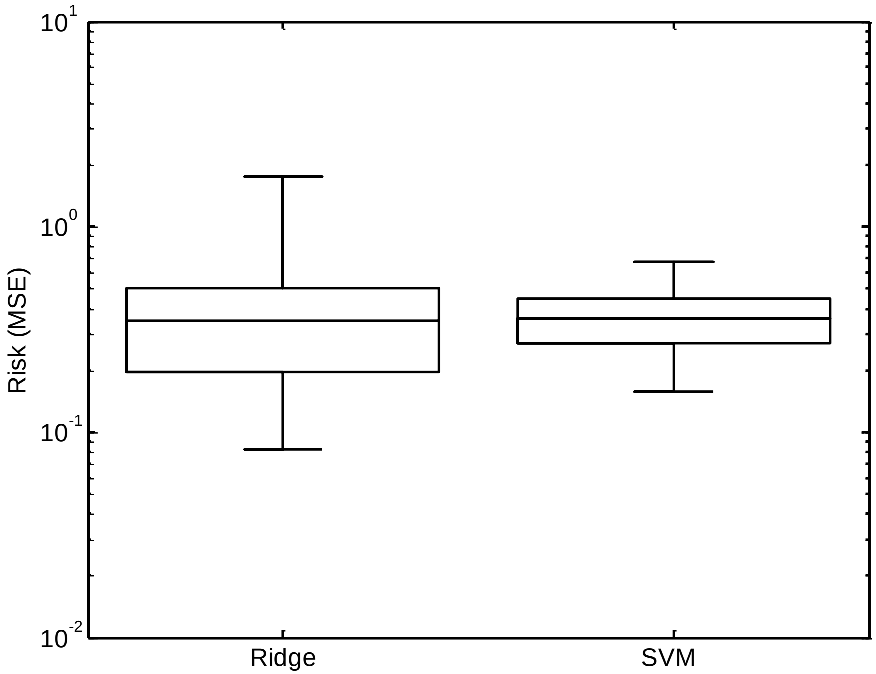
(a)Vary C (epsilon=0) (b)Vary epsilon(C=large)



Comparison: ridge regression vs SVM

- Sparse setting: n=10, noise $\sigma = 0.2$
- Ridge regression: λ chosen by cross-validation
- SV Regression: $\varepsilon = \sigma = 0.2$

  C selected by cross-validation
- Ave Risk (100 realizations): 0.44 (RR) vs 0.37 (SVM)



Summary

- SVM solves the learning problem directly → different SVM formulations
- Margin-based loss: robust + performs complexity control
- Implementation of SRM: new type of structure
- Nonlinear feature selection (~ SV's): incorporated into model estimation