

# 支撑向量机与核方法

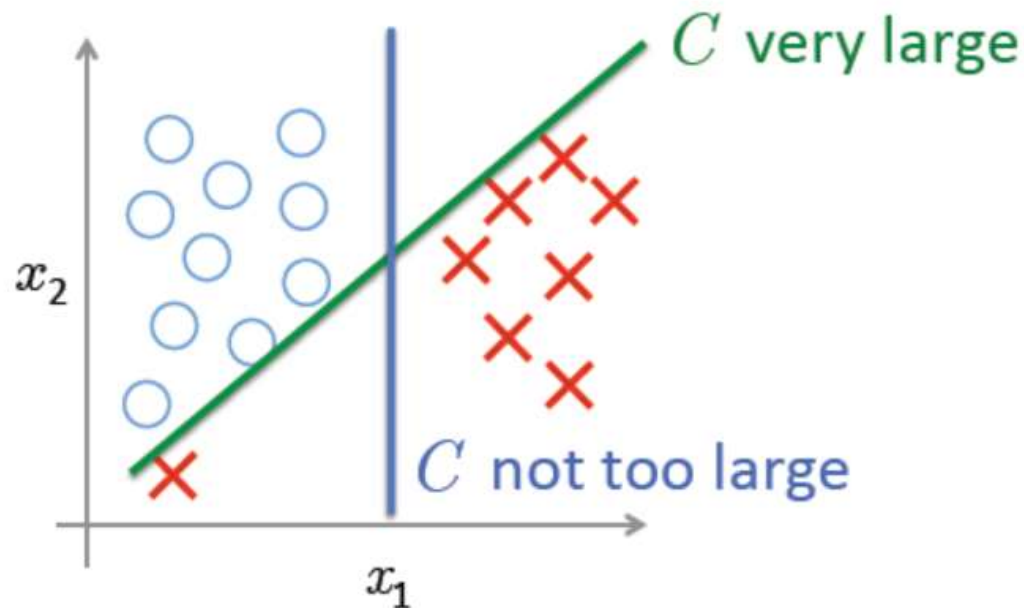
张煦尧([xyz@nlpr.ia.ac.cn](mailto:xyz@nlpr.ia.ac.cn))

2020年12月09日

# Model Selection

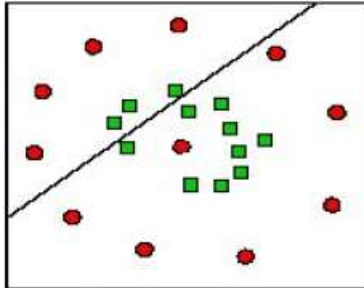
# The “C” Problem

- “C” plays a major role in controlling “overfitting”
- Finding the “Right” value for “C” is one of the major problems of SVM



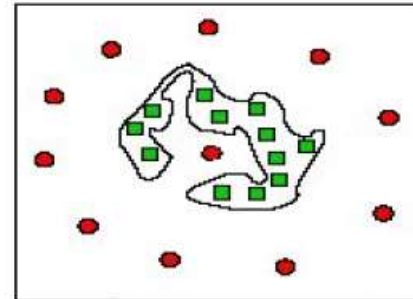
# Overfitting and Underfitting

Under-Fitting

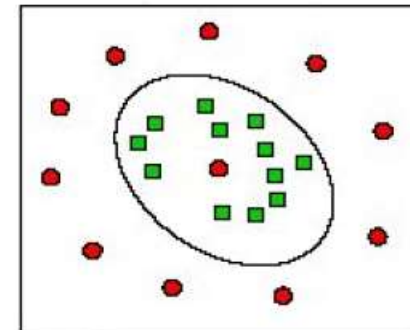
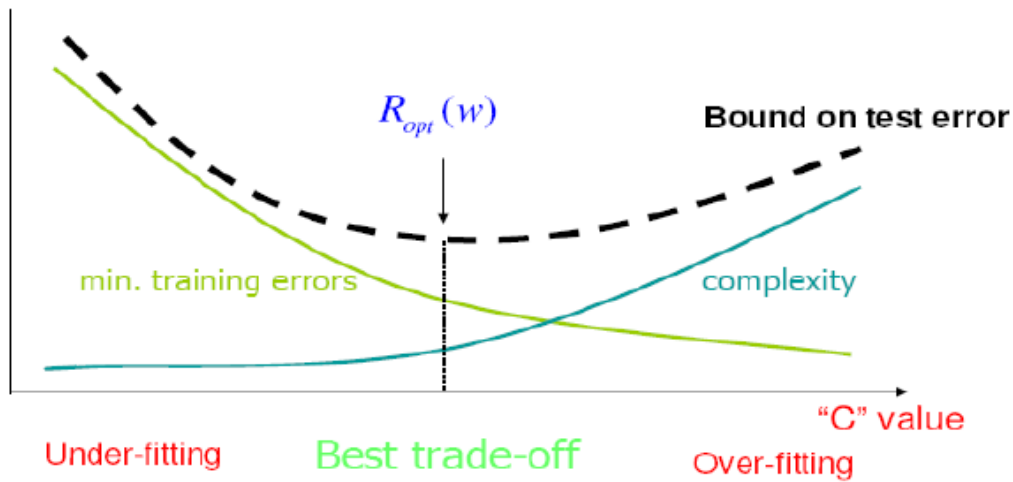


Too much simple!

Over-Fitting



Too much complicated!



Trade-Off

# Model Selection (C and Kernel)

- In practice a Gaussian radial basis or a low degree polynomial kernel is a good start
- Checking which set of parameters (such as C or  $\gamma$  if we choose RBF kernel) are the most appropriate by Cross-Validation (K- fold):
  - divide randomly all the available training examples into K equal-sized subsets
  - use all but one subset to train the SVM with the chosen parameters
  - use the held out subset to measure classification error
  - repeat above two steps for each subset
  - average the results to get an estimate of the generalization error of the SVM classifier

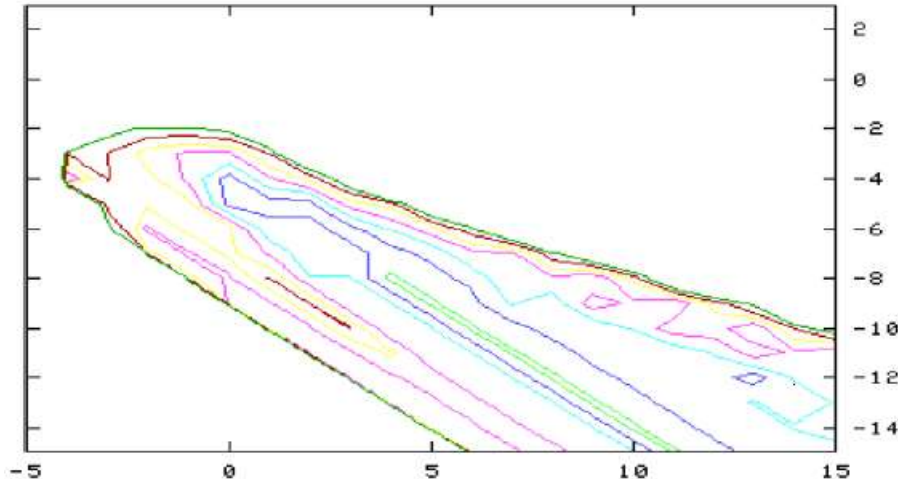
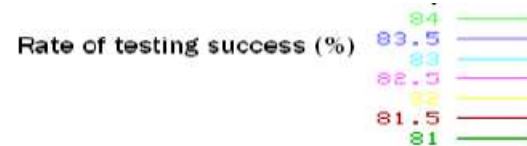
Why cross-validation?

# Model Selection Example

- This example is provided in the LIBSVM guide. In this example they are searching the “best” values for “C” and for an RBF Kernel for a given training using the model selection procedure we saw above

$$C = 2^{-5}, 2^{-4}, \dots, 2^{15}$$

$$\sigma = 2^{-15}, 2^{-14}, \dots, 2^3$$

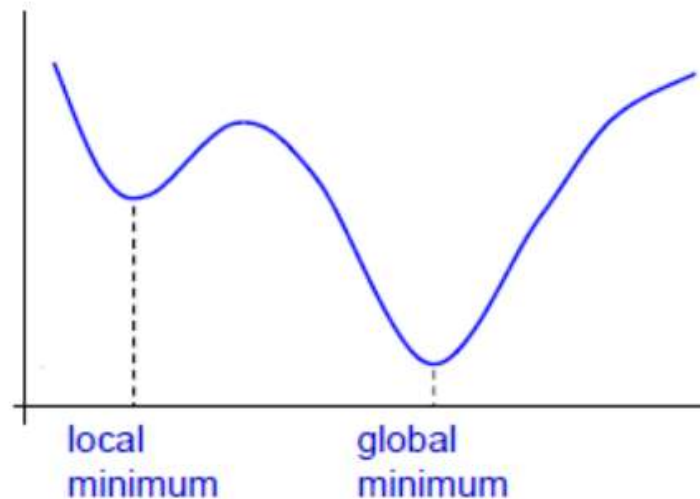


$C = 2^5, \sigma = 2^{-9}$   
*is a good choice*

# Optimization

# Local and Global Optimal

- Unique solution?
- Depend on initialization?



- If the cost function is convex, then a locally optimal point is globally optimal



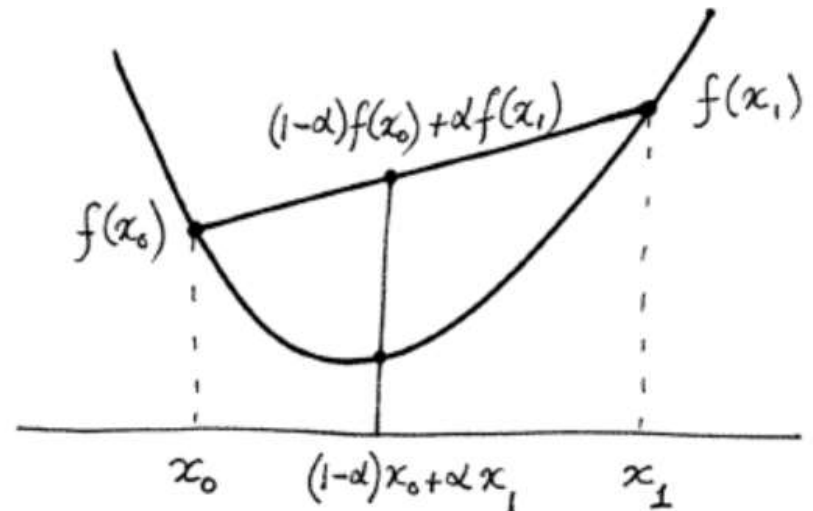
# Convex Function

$D$  – a domain in  $\mathbb{R}^n$ .

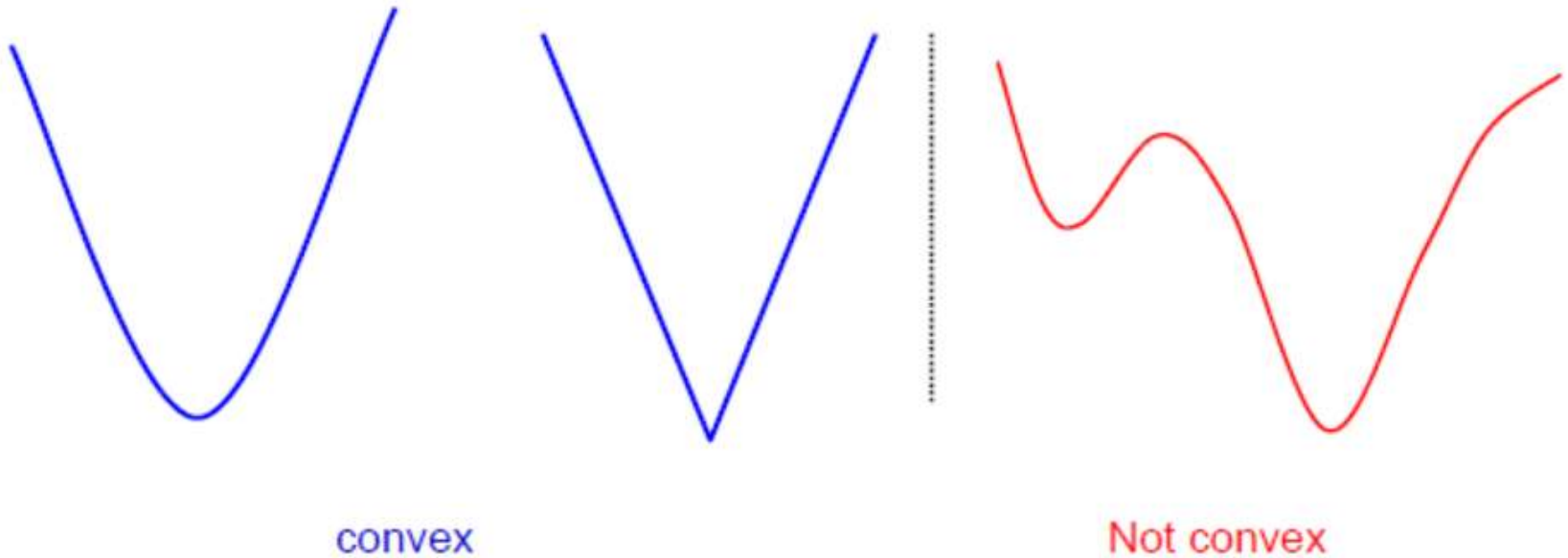
A **convex function**  $f : D \rightarrow \mathbb{R}$  is one that satisfies, for any  $x_0$  and  $x_1$  in  $D$ :

$$f((1 - \alpha)x_0 + \alpha x_1) \leq (1 - \alpha)f(x_0) + \alpha f(x_1) .$$

Line joining  $(x_0, f(x_0))$   
and  $(x_1, f(x_1))$  lies  
above the function graph.



# Convex Function Examples



A non-negative sum of convex functions is convex

SVM is convex. Why?

# SVM is Convex



SVM

$$\min_{\mathbf{w} \in \mathbb{R}^d} C \sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i)) + \|\mathbf{w}\|^2$$

# Quadratic Programming

- Many approaches have been proposed

<http://www.numerical.rl.ac.uk/qp/qp.html>

## *A QUADRATIC PROGRAMMING PAGE*

**Nick Gould and Philippe Toint**

This page contains a list of information and links related to the wonderful world of quadratic programming. If you have anything you would like to add, please send us a [message](#).

- **Quadratic programming codes:**

- [BQPD](#) from Roger Fletcher
- [CPLEX](#) Barrier/QP solver
- [CPLEX](#) Simplex/QP solver
- [CPLEX](#) Mixed-integer QP solver
- The [Xpress-MP](#) Newton-barrier QP solver from FICO
- [HOPDM](#) from Jacek Gondzio and Anna Altman
- [LINDO](#)
- The packages CQP, DQP, QP, QPC, QPB and QPA from [GALAHAD](#)
- The packages VE02, VE09, VE17, HSL\_VE12 and HSL\_VE19 from [HSL](#) (formerly known as the Harwell Subroutine Library)
- [LOQO](#) from Bob Vanderbei
- [MINQ](#) for convex general and non-convex bound constrained problems, in Matlab, by Arnold Neumaier
- A [method](#) for nonconvex quadratic programming by Gennadij Bulanov for Windows' users
- [CirCut](#) for finding approximate solutions to certain binary quadratic programs, including the Max-Cut and the Max-Bisection problems, by Yin Zhang
- The subroutines E04NCF, E04NFF, E04NKF, H02CBF and H02CEF from the [NAG fortran](#) library
- The package nag\_qp\_sol from the [NAG fl90](#) library

# Quadratic Programming

- Most are “interior-point” methods 内点法
  - Start with an initial solution that can violate the constraints
  - Improve this solution by optimizing the objective function and/or reducing the amount of constraint violation
- For SVM, sequential minimal optimization (SMO) seems to be the most popular
  - A QP with two variables is trivial to solve
  - Each iteration of SMO picks a pair of  $(\alpha_i, \alpha_j)$  and solve the QP with these two variables; repeat until convergence
- In practice, we can just regard the QP solver as a “blackbox” without bothering how it works
  - considering BP in deep learning

# Speed up training SVM

- Hint 1: Only SVs will influence the decision boundary

## —Chunk strategy 组块策略

Training SVM only on part data and keep remaining the SVs and repeating training by adding new data sequentially until SVs don't change

# Speed up training SVM

- Hint 2: When the optimum is achieved, KKT conditions are satisfied

$$\begin{cases} \alpha_i = 0 & \Rightarrow y_i f(x_i) \geq 1 \Rightarrow \text{Samples outside the boundary} \\ 0 < \alpha_i < C & \Rightarrow y_i f(x_i) = 1 \Rightarrow \text{Samples on the boundary} \\ \alpha_i = C & \Rightarrow y_i f(x_i) \leq 1 \Rightarrow \text{Samples within the boundary} \end{cases}$$

## — Sequential Minimal Optimization (SMO)

Each time, choose two samples violating the KKT condition and updating the weights until all the points satisfied KKT condition

J. C. Platt, Fast Training of Support Vector Machines using Sequential Minimal Optimization, Advances in kernel methods: support vector learning, 1999.

# Geometry of SVM



# Another Interpretation of SVM

Hard-margin SVM

$$\begin{aligned} \min_{\alpha} \quad & \left\| \sum_{I_+} \alpha_i x_i - \sum_{I_-} \alpha_i x_i \right\|^2 \\ \text{s.t.} \quad & \alpha_i \geq 0 \quad \forall i \\ & \sum_{I_+} \alpha_i = 1 \\ & \sum_{I_-} \alpha_i = 1 \end{aligned}$$

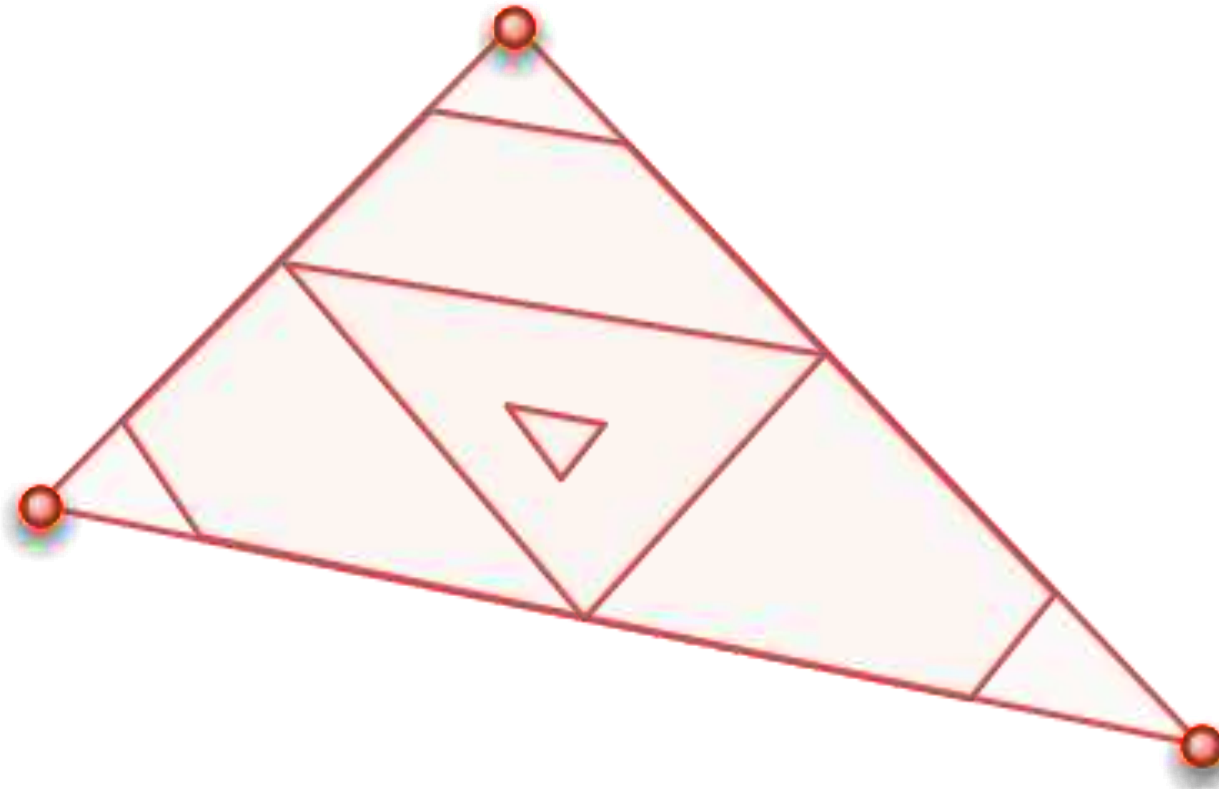
Soft-margin SVM

$$\begin{aligned} \min_{\alpha} \quad & \left\| \sum_{I_+} \alpha_i x_i - \sum_{I_-} \alpha_i x_i \right\|^2 \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq \mu \quad \forall i \\ & \sum_{I_+} \alpha_i = 1 \\ & \sum_{I_-} \alpha_i = 1 \end{aligned}$$

Geometric Interpretation:

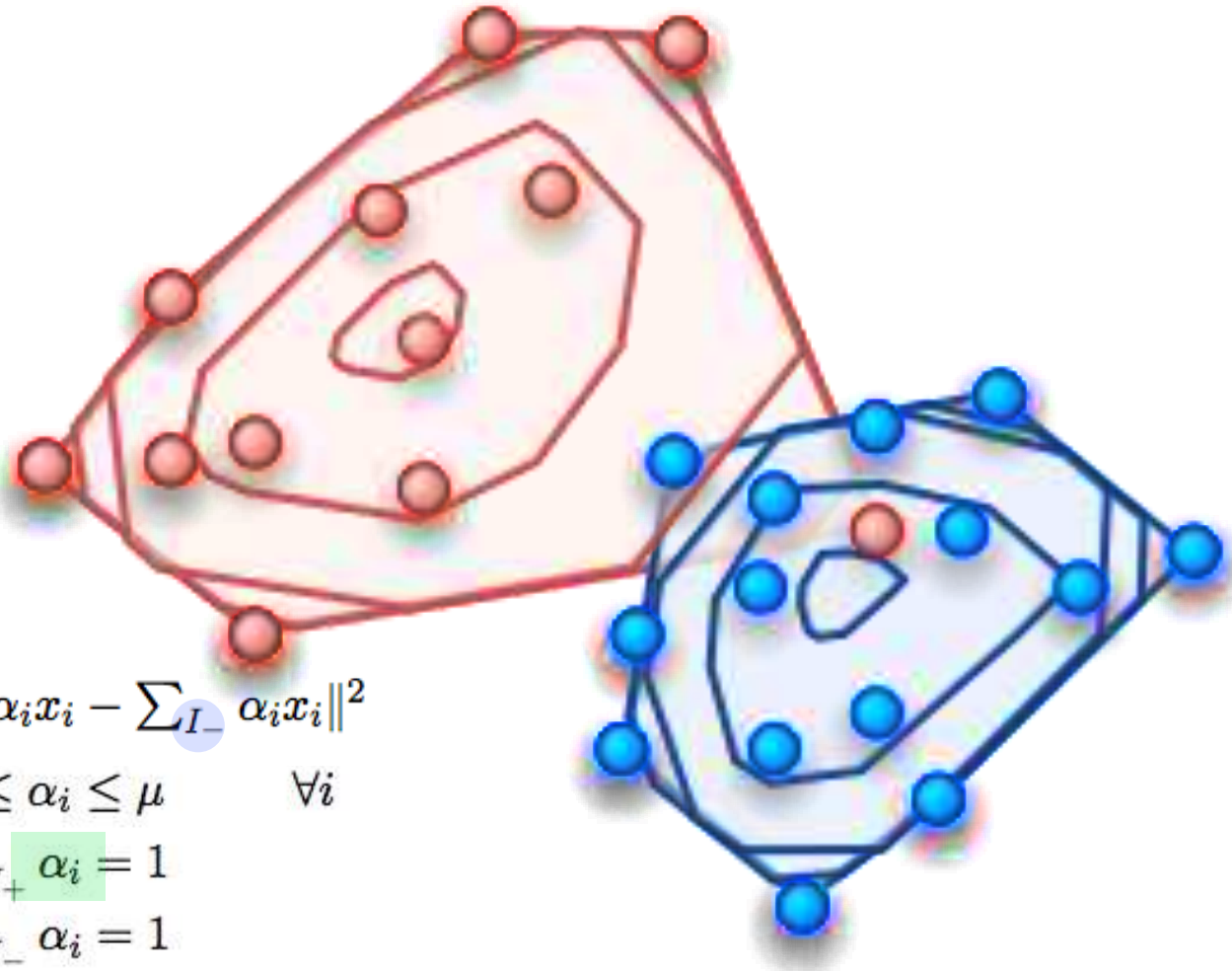
# Reduced Convex Hulls

$$RCH(X, \mu) := \left\{ \sum_i \alpha_i x_i \mid 0 \leq \alpha_i \leq \mu \ \forall i, \ \sum_i \alpha_i = 1 \right\}$$



# Geometric Interpretation:

## Distance between reduced convex hulls



$$\begin{aligned} \min_{\alpha} \quad & \left\| \sum_{I_+} \alpha_i x_i - \sum_{I_-} \alpha_i x_i \right\|^2 \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq \mu \quad \forall i \\ & \sum_{I_+} \alpha_i = 1 \\ & \sum_{I_-} \alpha_i = 1 \end{aligned}$$

# Multi-Class SVM

# Multi-class Extension

- SVMs can only handle two-class outputs (i.e. a categorical output variable with arity 2).
- What can be done?
- Answer: with output arity  $N$ , learn  $N$  SVM's
  - SVM 1 learns “Output==1” vs “Output != 1”
  - SVM 2 learns “Output==2” vs “Output != 2”
  - :
  - SVM  $N$  learns “Output== $N$ ” vs “Output !=  $N$ ”
- Then to predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

# Multi-class Extension

Two Methods for Multi-Class SVM:

- Multiple Binary Problem
  - one-vs-all
  - one-vs-one
  - DAGSVM
  - half-vs-half
  - LatticeSVM
- Single Machine
  - consider all classes at once, result in a much larger optimization problem in one step.

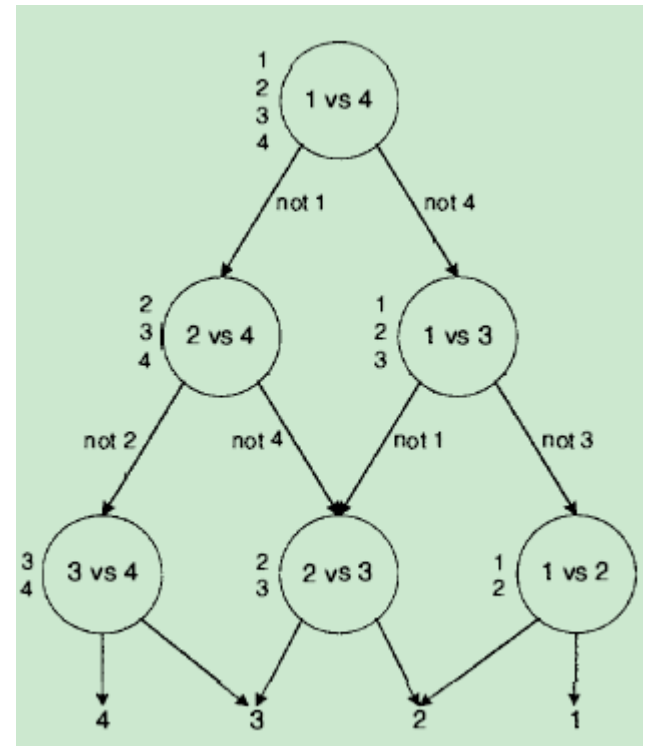
# Multiple Binary Problem

- one VS all (1-v-a)
  - $C$   $N$ -variable quadratic programming problems are solved
- one VS one (1-v-1)
  - $C(C-1)/2$  quadratic programming problems where each of them has about  $2N/C$  variables.

# DAGSVM

J.C. Platt et al, Large Margin DAGs for Multiclass Classification, *NIPS 2000*.

- Directed Acyclic Graph 有向无环图
- Rooted Binary DAG is used
- The choice of the class order in the list is arbitrary.





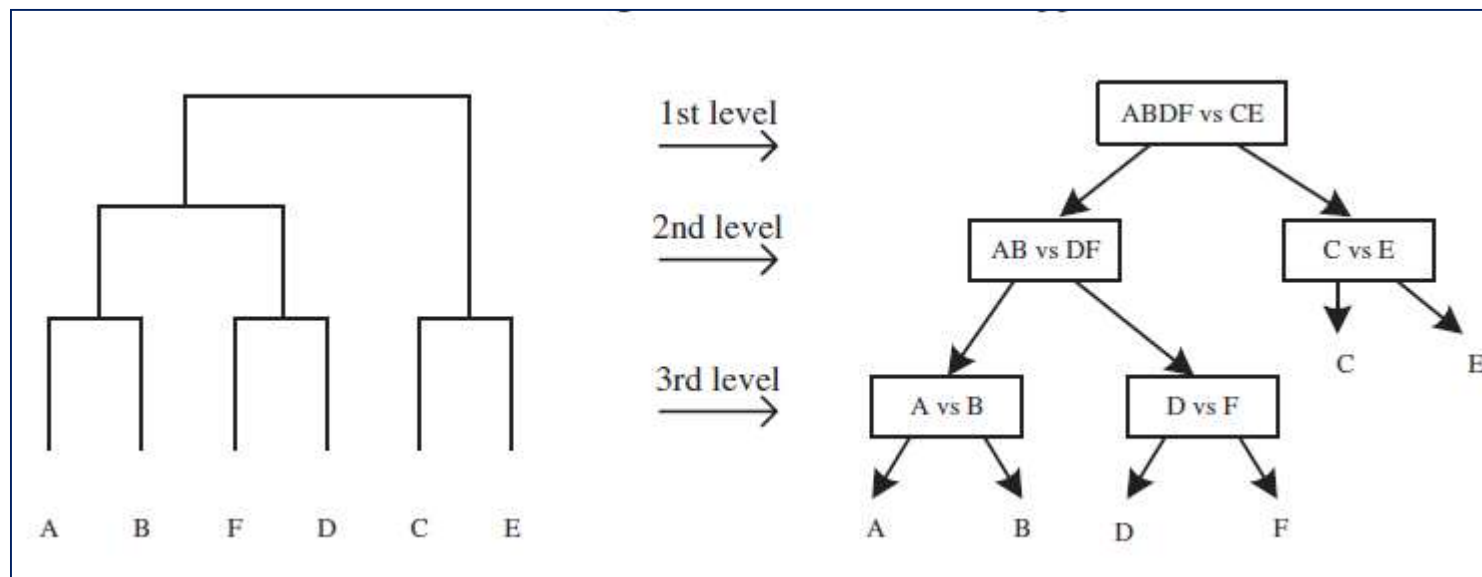
# DAGSVM

- Algorithm:
  1. Create a list of class labels  $L=(1,2,...,M)$  (arbitrary order)
  2. Evaluates the sample with the binary SVM that corresponds to the first and last element in list  $L$ , the loser class index is eliminated from the list.
  3. After  $M-1$  evaluations, the last label is the answer.

# Half-vs-Half

H. Lei, V. Govindaraju, Half-Against-Half Multi-Class Support Vector Machines, *MCS 2005*.

V. Vural, J.G. Dy, A Hierarchical Method for Multi-Class Support Vector Machines, *ICML 2004*.



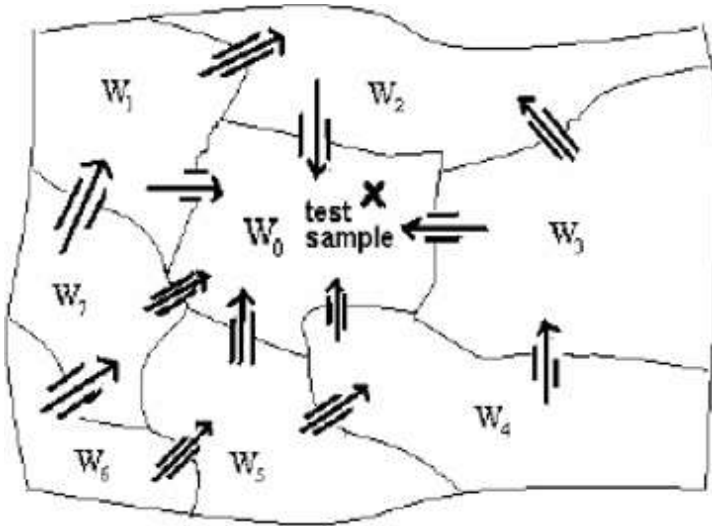
# Half-vs-Half

- How to divide the data into two subsets hierarchically?
  - ✓ Two subsets have the largest margin.
  - ✓ Two subsets have minimum number of SVs
  - ✓ k-means division
  - ✓ Spherical shells
  - ✓ Balanced subsets
  - ✓ Hierarchy clustering
- Testing speed: worst case:  $M-1$  evaluations
  - 1-v-1:  $M(M-1)/2$
  - 1-v-a:  $M$
  - DAGSVM:  $M-1$

# LatticeSVM

Z. Liu, L. Jin, LatticeSVM—A New Method for Multi-Class Support Vector Machines, *IJCNN 2008*.

- At most  $CK$  binary problem to be solved.  $K$  is the number of neighbor classes for each class.
- Methods to Obtain Neighbor Classes:
  - sample-sample distance
  - center-center distance
  - sample-center distance



Algorithm:

1. Choose a initial class  $i$  randomly, search its neighbor classes  $j$ .
2. While( $SVM_{ij}(x) = j$ )
  - move to the neighbor class  $j$
  - search its neighbor classes.
3. Classifiy using the last class index.

Note: for nonseparable cases, there will be cycle path, simply select the one with smaller index.

# Single Machine 1

- A more natural way to solve multi-class problems is to construct a decision function by considering all classes at once. Result in a much larger optimization problem in one step.
- Notation:

$$\{x_i, y_i\}_{i=1}^N, x_i \in R^d, y_i \in \{1, \dots, M\}$$

$$z_i = \phi(x_i), k(x_i, x_j) = \langle z_i, z_j \rangle$$

$$f_m(x) = w_m^T z = \sum_{i=1}^N \alpha_i^m k(x_i, x), m = 1, \dots, M$$

$$x \in \arg \max_m f_m(x)$$

$$\begin{aligned} & \min_{\{w_m\} \{\xi_i^m\}} \frac{1}{2} \sum_m \|w_m\|^2 + C \sum_i \sum_{m \neq y_i} \xi_i^m \\ & s.t. \begin{cases} w_{y_i}^T z_i - w_m^T z_i \geq 1 - \xi_i^m \\ \xi_i^m \geq 0 \end{cases}, \forall i, \forall m \neq y_i \end{aligned}$$

# Single Machine 1

J. Weston, C. Watkins, Multi-Class Support Vector Machines, *Technical Report 1998*.

$$f_m(x) = w_m^T z = \sum_{i=1}^N \alpha_i^m k(x_i, x), m = 1, \dots, M$$

- V.S.

$$f_m(x) = \sum_{i: y_i = m} \alpha_i k(x_i, x)$$

- Just solve a simple linear program:

$$\begin{aligned} \min \quad & \sum_{i=1}^N \alpha_i + C \sum_{i=1}^N \sum_{m \neq y_i} \xi_i^m \\ \text{s.t.} \quad & \begin{cases} f_{y_i}(x_i) - f_m(x_i) \geq 1 - \xi_i^m \\ \alpha_i \geq 0, \xi_i^m \geq 0 \end{cases}, \forall i, \forall m \neq y_i \end{aligned}$$

# Single Machine 2

K. Crammer, Y. Singer, On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines, *JMLR* 2001.

$$\begin{aligned} \min_{\{w_m\} \{\xi_i\}} \quad & \frac{1}{2} \sum_m \|w_m\|^2 + C \sum_i \xi_i \\ \text{s.t.} \quad & \begin{cases} w_{y_i}^T z_i - w_m^T z_i \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases}, \forall i, \forall m \neq y_i \end{aligned}$$

$$\begin{aligned} \xi_i &= \text{hinge}(w_{y_i}^T z_i - \max_{m \neq y_i} w_m^T z_i) \\ &= \max_{m \neq y_i} \xi_i^m \end{aligned}$$

- Compare the correct label to all the other labels with  $N(M-1)$  slack variables.
- The correct label is only compared to the highest similarity-score among the rest of the labels, with  $N$  slack variables.

# Comparison

C.-W. Hsu, C.-J. Lin, A Comparison of Methods for Multiclass Support Vector Machines, *IEEE Trans. Neural Networks*, 2002.

- 1-v-1 and DAG methods are more suitable for practical use in large scale case.
- Single Machine considering all class at once will lead to fewer support vectors.



# Locally Linear SVM

# Motivation

- Linear SVM: fast and simple
  - drawback: many data are not linearly separable
- Kernel SVM: flexible
  - drawback: large number of SVs
- Solution: Locally Linear SVM

L. Ladicky, P.H.S. Torr, Locally Linear Support Vector Machines, ICML 2011.

# Local Coding for Manifold Learning

- $\mathbf{v}$ : anchor point

$$\mathbf{x} \approx \sum_{\mathbf{v} \in C} \gamma_{\mathbf{v}}(\mathbf{x}) \mathbf{v}$$

$$\sum_{\mathbf{v} \in C} \gamma_{\mathbf{v}}(\mathbf{x}) = 1$$

- We can approximate a function by local coding

$$f(\mathbf{x}) \approx \sum_{\mathbf{v} \in C} \gamma_{\mathbf{v}}(\mathbf{x}) f(\mathbf{v})$$

# Locally Linear SVM

- SVM  $H(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$

- Local SVM  $H(\mathbf{x}) = \mathbf{w}(\mathbf{x})^T \mathbf{x} + b(\mathbf{x})$

$$\begin{aligned} H(\mathbf{x}) &= \sum_{i=1}^n \sum_{\mathbf{v} \in C} \gamma_{\mathbf{v}}(\mathbf{x}) w_i(\mathbf{v}) x_i + \sum_{\mathbf{v} \in C} \gamma_{\mathbf{v}}(\mathbf{x}) b(\mathbf{v}) \\ &= \sum_{\mathbf{v} \in C} \gamma_{\mathbf{v}}(\mathbf{x}) \left( \sum_{i=1}^n w_i(\mathbf{v}) x_i + b(\mathbf{v}) \right). \end{aligned} \quad (7)$$

# Locally Linear SVM

- $m$ : the number of anchor point
- $n$ : the dimensionality
- $W$ :  $m \times n$  matrix, each row is a linear classify

$$H(\mathbf{x}) = \gamma(\mathbf{x})^T \mathbf{W} \mathbf{x} + \gamma(\mathbf{x})^T \mathbf{b}$$

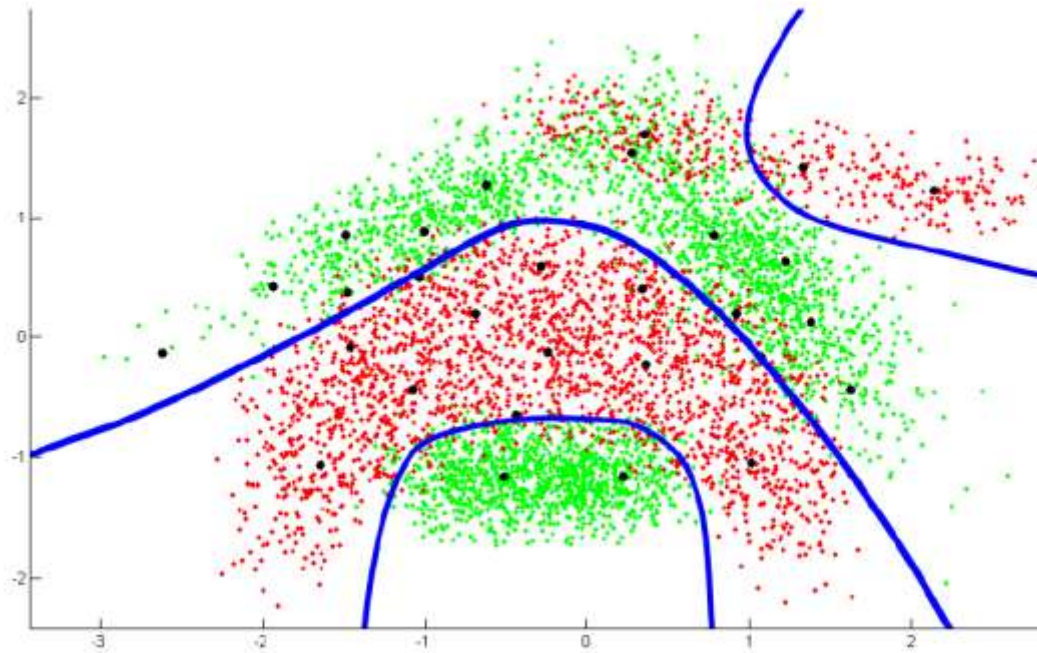
$$\arg \min_{\mathbf{W}, \mathbf{b}} \frac{\lambda}{2} \|\mathbf{W}\|^2 + \frac{1}{|S|} \sum_{k \in S} \max(0, 1 - y_k H_{\mathbf{W}, \mathbf{b}}(\mathbf{x}_k)),$$

- ✓ Bilinear SVM
- ✓ Transform the feature from  $n$  to  $nm$
- ✓ Weighted sum of linear SVMs for each anchor point
- ✓ More general than standard linear SVM
- ✓ More general than linear SVM over local coding

# Locally Linear SVM

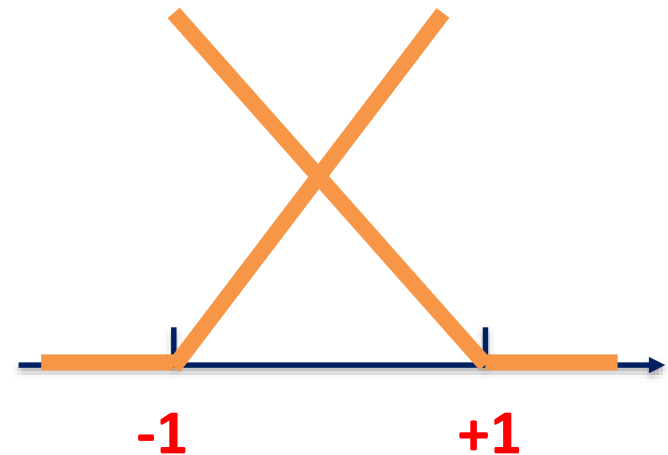
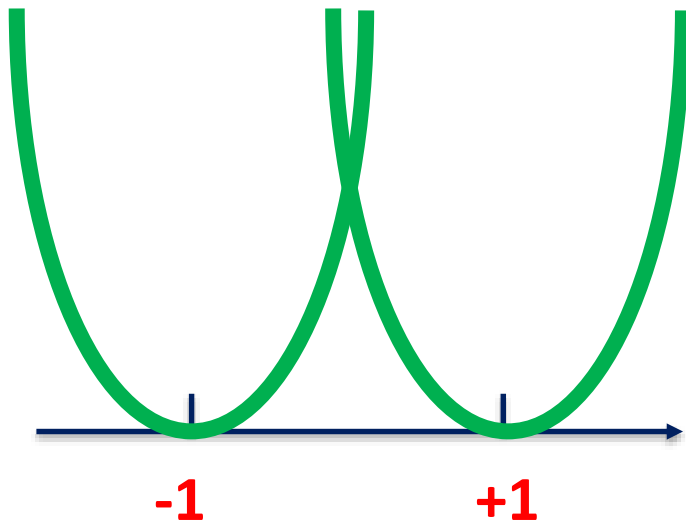
K-means clustering  $\rightarrow$  anchor points

Coefficients of the local coding  $\rightarrow$  inverse Euclidian distance based weighting from nearest neighbors.



# Least Squares Extension of SVM

# Least Square Loss vs Hinge Loss





# Linear Regression

- Data set  $\{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^m$
- Destination set  $\{\mathbf{y}_i\}_{i=1}^n \subset \mathbb{R}^c$
- Linear regression can be defined as:

$$\min_{\mathbf{W}, \mathbf{b}} \sum_{i=1}^n \left\| \mathbf{W}^T \mathbf{x}_i + \mathbf{b} - \mathbf{y}_i \right\|_2^2 + \lambda \left\| \mathbf{W} \right\|_F^2$$

$$\mathbf{W} \in \mathbb{R}^{m \times c} \quad \mathbf{b} \in \mathbb{R}^c$$

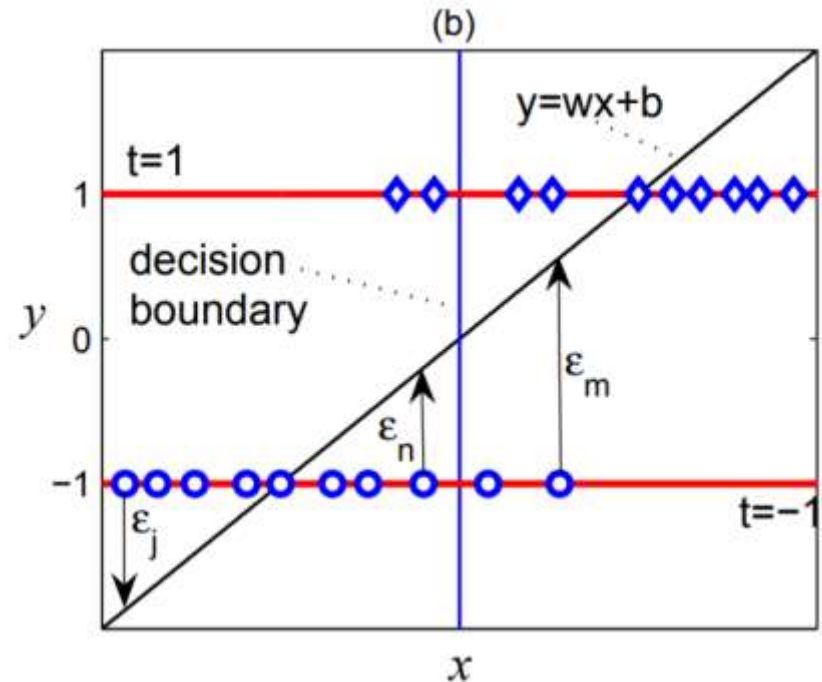
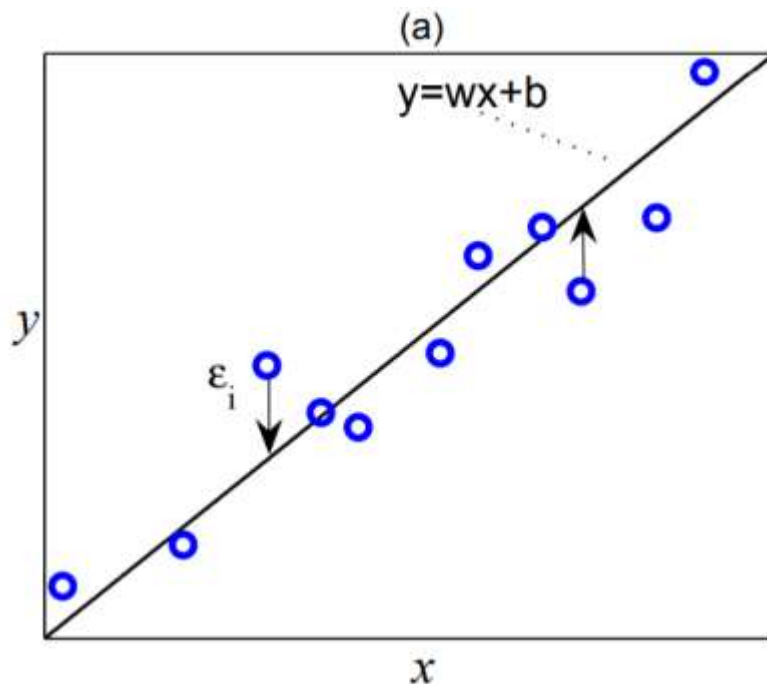
- Vector L2 norm  $\| \cdot \|_2$
- Matrix Frobenius norm  $\| \cdot \|_F$
- **The simplest (multi-class) classifier !**

# Different Loss Functions

- Data  $\{x_i\}_{i=1}^N \quad \{t_i\}_{i=1}^N \quad t_i \in \{1, -1\}$
- Classifier  $\hat{t} = \text{sign}(w^\top \phi(x) + b).$
- Margin variable  $z = ty$
- **Large  $z \rightarrow$  small loss**
- Loss Functions:
  - 0-1 loss  $l_{mer}(z_i) = \|(-z_i)_+\|_0$
  - Hinge loss  $l_{hinge} = \{(1 - z_i)_+\}^p \quad (p=1 \text{ or } 2)$
  - Logistic regression  $l_{log} = \log_2[1 + \exp(-z_i)]$
  - Least squares  $l_{ls} = (1 - z_i)^2$
  - Adaboost  $l_{exp} = \exp(-z_i).$

# Regression vs Classification

- For regression, data pairs are located closely nearby the prediction function
- For classification, data pairs are clamped onto two pattern-tracks (the red axis-aligned lines), as a result, the residues (arrow line) have much larger magnitudes.



# Stagewise Least Square

- Use a least square form within each stage to approximate a bounded monotonic nonconvex loss

$$l^{(k)}(z_i) = (z_i - \tau_i^{(k)})^2$$

$$z_i^{(k)} = t_i y_i^{(k)} \quad y_i^{(k)} = x_i^T w^{(k)} + b^{(k)}$$

$$\begin{aligned} \tau_i^{(0)} &= 1, \\ \tau_i^{(k+1)} &= z_i^{(k)} + S(1 - z_i^{(k)}), \end{aligned}$$

$$S(v) = \max(0, \min(1, v)).$$

# Stagewise Least Square

$$\begin{aligned}\tau_i^{(0)} &= 1, \\ \tau_i^{(k+1)} &= z_i^{(k)} + S(1 - z_i^{(k)}),\end{aligned}$$

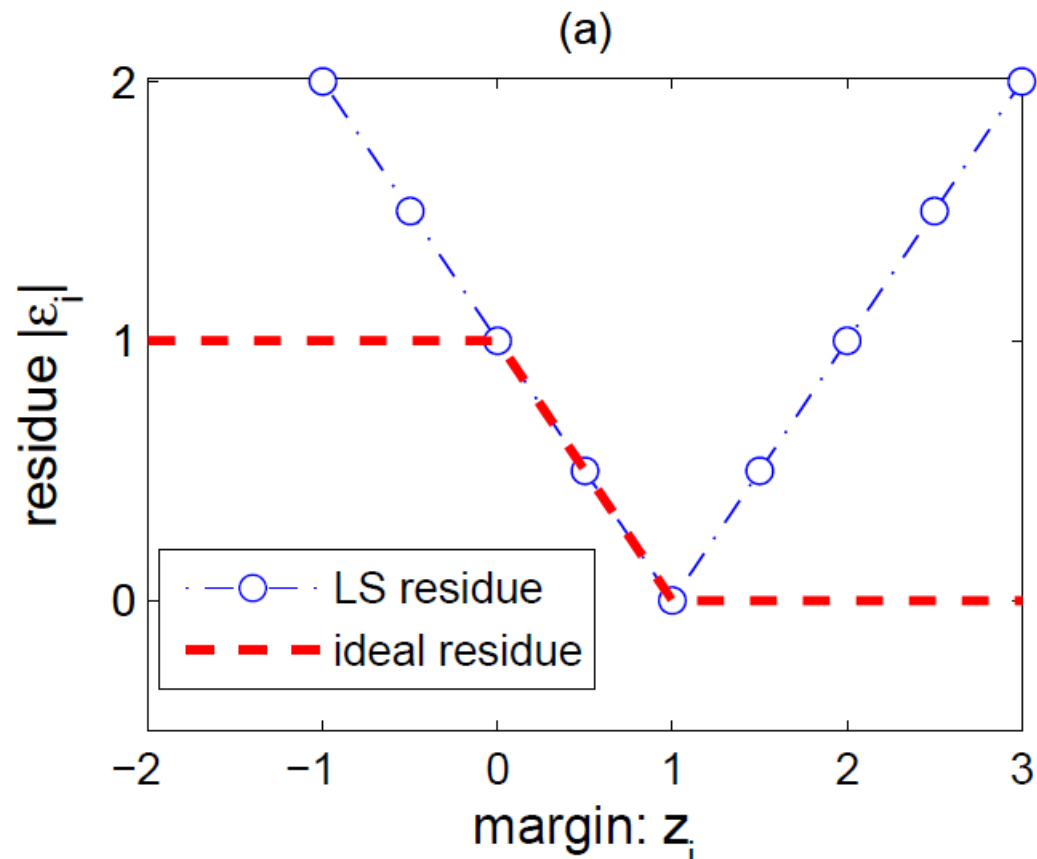
- For  $z > 1$  (correctly classified with margin  $> 1$ )
  - Target is  $z$
  - Place less emphasis on such patterns
  - Loss = 0
- For  $z < 0$  (misclassified)
  - Target is  $z + 1$
  - Penalize misclassification
  - Loss = 1
- For  $0 < z < 1$  (correctly classified with insufficient margin)
  - Target is 1
  - Encourage increasing margin (normal least squares)
  - Loss =  $(z-1)^2$

# Stagewise Least Square

$$\tau = \begin{cases} z, & z > 1 \\ z + 1, & z < 0 \\ 1, & 0 \leq z \leq 1 \end{cases}$$

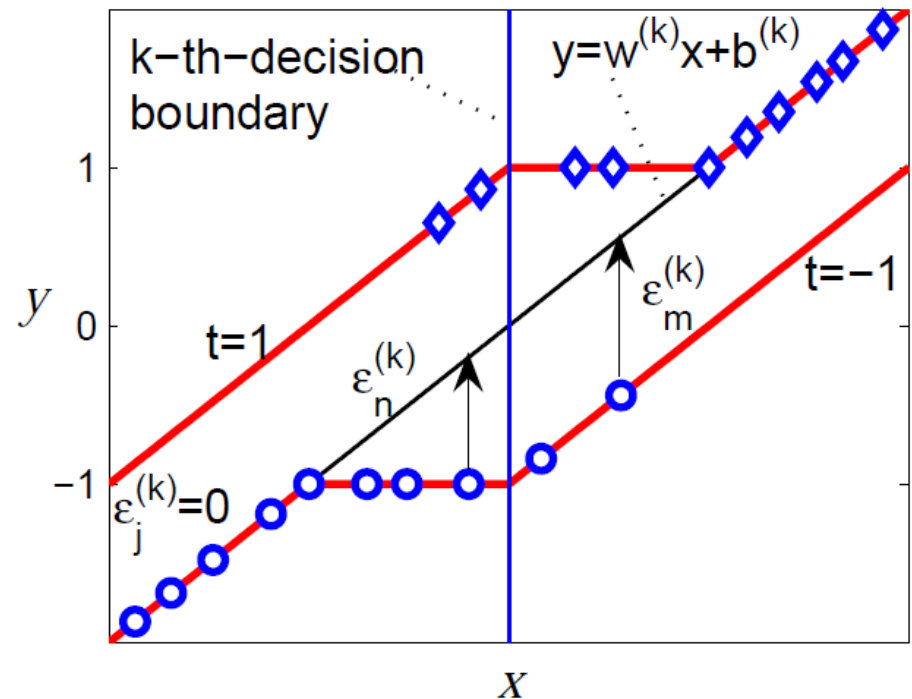
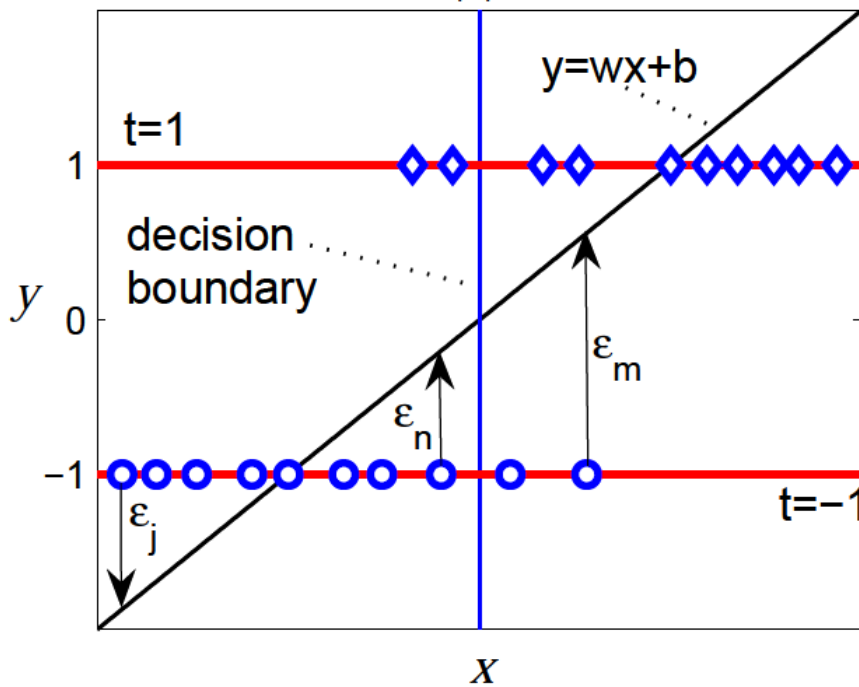
$$\text{loss} = (z - \tau)^2 = \begin{cases} 0, & z > 1 \\ 1, & z < 0 \\ (z - 1)^2, & 0 \leq z \leq 1 \end{cases}$$

# Stagewise Least Square



# Stagewise Least Square

$$\tau = \begin{cases} z, & z > 1 \\ z + 1, & z < 0 \\ 1, & 0 \leq z \leq 1 \end{cases}$$



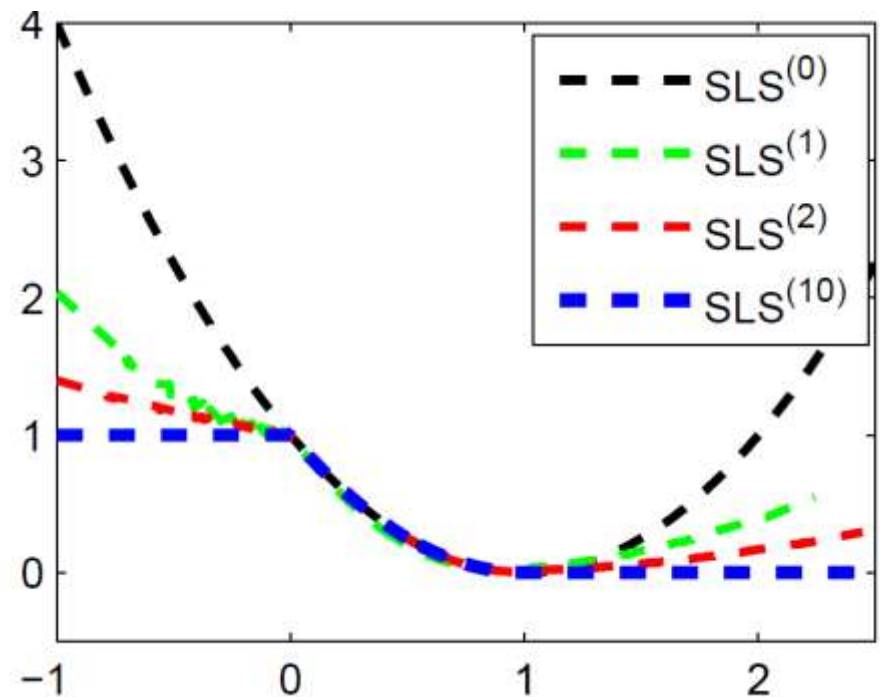
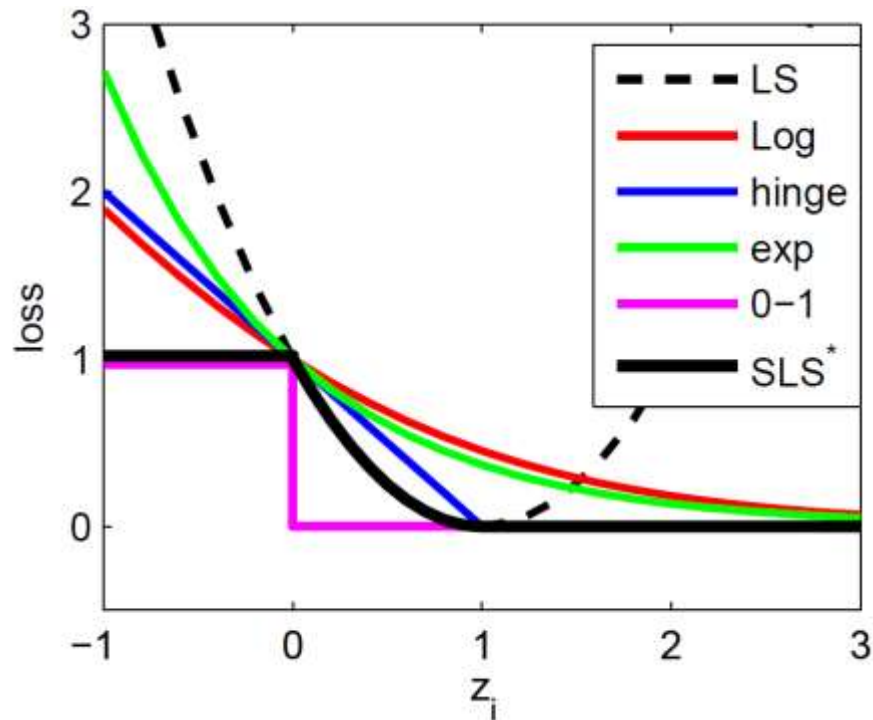


# Stagewise Least Square

- Convexity: it is convex at each stage
- Efficiency & Simplicity: it is in an LS style within each stage, thus it is easy to be implemented (merely involves solving linear systems)
- Sparseness: it naturally results in sparser kernel classifiers with better scalability
- Effectiveness & Robustness: it approximates a bounded loss function in a stage-wise manner

# Stagewise Least Square

$$l_{SLS^*}^{(k+1)} = \begin{cases} 0, & \text{if } z_i > 1 \\ (1 - z_i)^2, & \text{if } z_i \in [0, 1] \\ 1, & \text{else} \end{cases}$$



# Stagewise Least Square

$$\min \frac{1}{2} \bar{w}^\top \bar{w} + C \frac{1}{2} \sum_{i=1}^N (\bar{w}^\top \bar{x}_i - t_i \tau_i)^2.$$

$$\bar{w} = (\bar{X}^\top \bar{X} + C^{-1} I)^{-1} \bar{X}^\top \Gamma.$$

---

**Algorithm 1** Linear SLS-based Classifier (LSLS)

---

-**Input:** Data matrix  $X$ , label vector  $t$ ,  $C$ ,  $k_{max}$

-**Output:** A linear classifier:  $\text{sign}(w^\top x + b)$

$G = (\bar{X}^\top \bar{X} + C^{-1} I)^{-1} \bar{X}^\top$  (if  $D \gg N$ , use Eq.(3.9))

**for**  $k=1$  **to**  $k_{max}$

$\bar{w}^{(k)} = G \times \Gamma^{(k)}$

**end for**

$$\bar{w} = \bar{w}^{(k_{max})}$$

---

# Discriminative LSR

- Data  $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$   $y_i \in \{1, 2, \dots, c\}$
- Class  $\mathbf{f}_j = [0, \dots, 0, 1, 0, \dots, 0]^T \in \mathbb{R}^c$
- Training sample fitting

$$\mathbf{f}_{y_i} \approx \mathbf{W}^T \mathbf{x}_i + \mathbf{t}, \quad i = 1, 2, \dots, n$$

- Let

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times m}$$

$$\mathbf{Y} = [\mathbf{f}_{y_1}, \mathbf{f}_{y_2}, \dots, \mathbf{f}_{y_n}]^T \in \mathbb{R}^{n \times c}$$

- Training sample fitting

$$\mathbf{XW} + \mathbf{e}_n \mathbf{t}^T \approx \mathbf{Y}$$

Shiming Xiang et al. Discriminative Least Squares Regression for Multiclass Classification and Feature Selection, IEEE Transactions on Neural Network and Learning Systems (T-NNLS), 2012.

# Discriminative LSR

$$\mathbf{XW} + \mathbf{e}_n \mathbf{t}^T \approx \mathbf{Y}$$

- Each column of  $\mathbf{Y} \rightarrow$  binary regression with +1/0
- How to enlarge the margin?

- Define a new matrix  $\mathbf{B} \in \mathbb{R}^{n \times c}$

$$B_{ij} = \begin{cases} +1, & \text{if } y_i = j \\ -1, & \text{otherwise.} \end{cases}$$

- Each element in  $\mathbf{B}$  corresponds to a dragging direction
- Target re-definition  $\mathbf{M} \in \mathbb{R}^{n \times c}$

$$\mathbf{XW} + \mathbf{e}_n \mathbf{t}^T - (\mathbf{Y} + \mathbf{B} \odot \mathbf{M}) \approx \mathbf{0}$$

# Discriminative LSR

- Optimize three parts of parameters

$$\begin{array}{ll} \min_{\mathbf{W}, \mathbf{t}, \mathbf{M}} & ||\mathbf{XW} + \mathbf{e}_n \mathbf{t}^T - \mathbf{Y} - \mathbf{B} \odot \mathbf{M}||_F^2 + \lambda ||\mathbf{W}||_F^2 \\ \text{s.t.} & \mathbf{M} \geq \mathbf{0} \end{array}$$

- OPT1:
  - Fix  $\mathbf{M}$  and solve  $\mathbf{W}, \mathbf{t} \rightarrow$  unconstrained convex QP  $\rightarrow$  解析解
- OPT2:
  - Fix  $\mathbf{W}, \mathbf{t}$  and solve  $\mathbf{M} \rightarrow$  very simple elementwise solution

# DLSR: OPT1: Fix M

$$\begin{aligned} \min_{\mathbf{W}, \mathbf{t}, \mathbf{M}} \quad & \|\mathbf{X}\mathbf{W} + \mathbf{e}_n \mathbf{t}^T - \mathbf{Y} - \mathbf{B} \odot \mathbf{M}\|_F^2 + \lambda \|\mathbf{W}\|_F^2 \\ \text{s.t.} \quad & \mathbf{M} \geq \mathbf{0} \end{aligned}$$

- Fix M, and let  $\mathbf{R} = \mathbf{Y} + \mathbf{B} \odot \mathbf{M} \in \mathbb{R}^{n \times c}$
- We have:

$$\mathbf{W} = (\mathbf{X}^T \mathbf{H} \mathbf{X} + \lambda \mathbf{I}_m)^{-1} \mathbf{X}^T \mathbf{H} \mathbf{R}$$

$$\mathbf{t} = \frac{(\mathbf{R}^T \mathbf{e}_n - \mathbf{W}^T \mathbf{X}^T \mathbf{e}_n)}{n}$$

# DLSR: OPT1: Fix M

$$\begin{aligned} \min_{\mathbf{W}, \mathbf{t}, \mathbf{M}} \quad & \|\mathbf{XW} + \mathbf{e}_n \mathbf{t}^T - \mathbf{Y} - \mathbf{B} \odot \mathbf{M}\|_F^2 + \lambda \|\mathbf{W}\|_F^2 \\ \text{s.t.} \quad & \mathbf{M} \geq \mathbf{0} \end{aligned}$$

$$\begin{aligned} \frac{\partial g(\mathbf{W}, \mathbf{t})}{\partial \mathbf{t}} = \mathbf{0} &\Rightarrow \mathbf{W}^T \mathbf{X}^T \mathbf{e}_n + \mathbf{t} \mathbf{e}_n^T \mathbf{e}_n - \mathbf{R}^T \mathbf{e}_n = \mathbf{0} \\ &\Rightarrow \mathbf{t} = \frac{(\mathbf{R}^T \mathbf{e}_n - \mathbf{W}^T \mathbf{X}^T \mathbf{e}_n)}{n}. \end{aligned}$$

$$\begin{aligned} \frac{\partial g(\mathbf{W}, \mathbf{t})}{\partial \mathbf{W}} &= \mathbf{0} \\ &\Rightarrow \mathbf{X}^T \left( \mathbf{XW} + \frac{1}{n} \mathbf{e}_n \mathbf{e}_n^T \mathbf{R} - \frac{1}{n} \mathbf{e}_n \mathbf{e}_n^T \mathbf{XW} - \mathbf{R} \right) + \lambda \mathbf{W} = \mathbf{0} \\ &\Rightarrow \mathbf{X}^T \left( \mathbf{I}_n - \frac{1}{n} \mathbf{e}_n \mathbf{e}_n^T \right) \mathbf{XW} - \mathbf{X}^T \left( \mathbf{I}_n - \frac{1}{n} \mathbf{e}_n \mathbf{e}_n^T \right) \mathbf{R} + \lambda \mathbf{W} = \mathbf{0} \\ &\Rightarrow \mathbf{W} = (\mathbf{X}^T \mathbf{H} \mathbf{X} + \lambda \mathbf{I}_m)^{-1} \mathbf{X}^T \mathbf{H} \mathbf{R}. \end{aligned}$$



## DLSR: OPT2: Fix $\mathbf{W}$ $\mathbf{t}$

$$\begin{aligned} \min_{\mathbf{W}, \mathbf{t}, \mathbf{M}} \quad & ||\mathbf{XW} + \mathbf{e}_n \mathbf{t}^T - \mathbf{Y} - \mathbf{B} \odot \mathbf{M}||_F^2 + \lambda ||\mathbf{W}||_F^2 \\ \text{s.t.} \quad & \mathbf{M} \geq \mathbf{0} \end{aligned}$$

$$\mathbf{P} = \mathbf{XW} + \mathbf{e}_n \mathbf{t}^T - \mathbf{Y}$$

$$\min_{\mathbf{M}} \quad ||\mathbf{P} - \mathbf{B} \odot \mathbf{M}||_F^2, \quad \text{s.t.} \quad \mathbf{M} \geq \mathbf{0}.$$

$$\min_{M_{ij}} \quad (P_{ij} - B_{ij} M_{ij})^2, \quad \text{s.t.} \quad M_{ij} \geq 0$$

$$M_{ij} = \max(B_{ij} P_{ij}, 0).$$

$$\mathbf{M} = \max(\mathbf{B} \odot \mathbf{P}, \mathbf{0}).$$

# Algorithm of DLSR

$$\begin{aligned} \min_{\mathbf{W}, \mathbf{t}, \mathbf{M}} \quad & \|\mathbf{X}\mathbf{W} + \mathbf{e}_n \mathbf{t}^T - \mathbf{Y} - \mathbf{B} \odot \mathbf{M}\|_F^2 + \lambda \|\mathbf{W}\|_F^2 \\ \text{s.t.} \quad & \mathbf{M} \geq \mathbf{0} \end{aligned}$$

---

## Algorithm 1 *DLSR*

---

**Input:**  $n$  data points  $\{\mathbf{x}_i\}_{i=1}^n$  in  $\mathbb{R}^m$ , and their corresponding class labels  $\{y_i\}_{i=1}^n \subset \{1, 2, \dots, c\}$ ; parameter  $\lambda$  in (7); and maximum number of iterations  $T$ .

- 1: Allocate  $\mathbf{M}$ ,  $\mathbf{W}$ ,  $\mathbf{W}_0$ ,  $\mathbf{t}$ , and  $\mathbf{t}_0$ .
  - 2:  $\mathbf{M} = \mathbf{0}$ ,  $\mathbf{W}_0 = \mathbf{0}$ , and  $\mathbf{t}_0 = \mathbf{0}$ .
  - 3: Construct  $\mathbf{X}$  and  $\mathbf{Y}$  in (4), and  $\mathbf{B}$  according to (5).
  - 4: Let  $\mathbf{U} = (\mathbf{X}^T \mathbf{H} \mathbf{X} + \lambda \mathbf{I}_m)^{-1} \mathbf{X}^T \mathbf{H}$ .
  - 5: Let  $k = 1$ .
  - 6: **while**  $k < T$  **do**
  - 7:    $\mathbf{R} = \mathbf{Y} + \mathbf{B} \odot \mathbf{M}$ .
  - 8:    $\mathbf{W} = \mathbf{U}\mathbf{R}$ ,    $\mathbf{t} = \frac{1}{n} \mathbf{R}^T \mathbf{e}_n - \frac{1}{n} \mathbf{W}^T \mathbf{X}^T \mathbf{e}_n$ .
  - 9:    $\mathbf{P} = \mathbf{X}\mathbf{W} + \mathbf{e}_n \mathbf{t}^T - \mathbf{Y}$ .
  - 10:    $\mathbf{M} = \max(\mathbf{B} \odot \mathbf{P}, \mathbf{0})$ .
  - 11:   **if**  $(\|\mathbf{W} - \mathbf{W}_0\|_F^2 + \|\mathbf{t} - \mathbf{t}_0\|_2^2) < 10^{-4}$ , **then**
  - 12:     Stop.
  - 13:   **end if**
  - 14:    $\mathbf{W}_0 = \mathbf{W}$ ,    $\mathbf{t}_0 = \mathbf{t}$ ,    $k = k + 1$ .
  - 15: **end while**
  - 16: Output  $\mathbf{W}$  and  $\mathbf{t}$ .
-

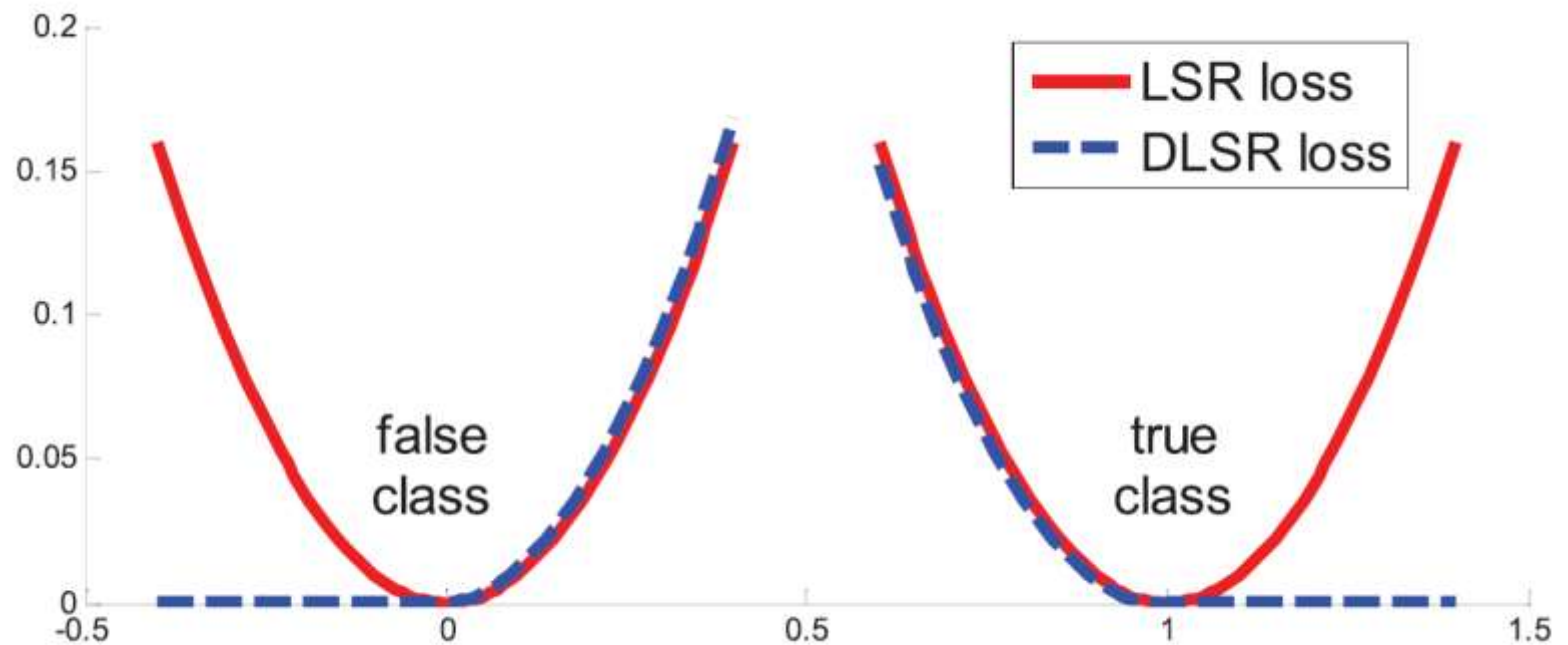
# Re-think DLSR

$$\begin{aligned} \mathbf{W} &\in \mathbb{R}^{m \times c} \\ \mathbf{M} &\geq \mathbf{0} \end{aligned} \quad B_{ij} = \begin{cases} +1, & \text{if } y_i = j \\ -1, & \text{otherwise.} \end{cases}$$

$$\begin{aligned} \min_{\mathbf{W}, \mathbf{t}, \mathbf{M}} \quad & \|\mathbf{XW} + \mathbf{e}_n \mathbf{t}^T - \mathbf{Y} - \mathbf{B} \odot \mathbf{M}\|_F^2 + \lambda \|\mathbf{W}\|_F^2 \\ \text{s.t.} \quad & \mathbf{M} \geq \mathbf{0} \end{aligned}$$

- Another formalization of **one-vs-all SVM** with **squared hinge loss** !

# Re-think DLSR



# Retargeted LSR

- LSR: use **exact 0-1** as target
- DLSR: use **relaxed 0-1** as target
- ReLSR: use **any (learned) number** as target
  - Totally learn the regression targets from data
  - With flexible large margin constraints

| y         | regression result | margin $\geq 1$ | LSR       | DLSR                             | ReLSR                                     |
|-----------|-------------------|-----------------|-----------|----------------------------------|---|
| [1, 0, 0] | [1.5, 0, 0]       | Yes             | loss=0.25 | loss=0.00 target=[1.5, 0, 0]     | loss=0.00 target=[1.5, 0, 0]              |
| [1, 0, 0] | [1, -0.5, -0.5]   | Yes             | loss=0.50 | loss=0.00 target=[1, -0.5, -0.5] | loss=0.00 target=[1, -0.5, -0.5]          |
| [0, 1, 0] | [0.5, 1.5, 0.5]   | Yes             | loss=0.75 | loss=0.50 target=[0, 1.5, 0]     | loss=0.00 target=[0.5, 1.5, 0.5]          |
| [0, 1, 0] | [-0.5, 1.5, 0.5]  | Yes             | loss=0.75 | loss=0.25 target=[-0.5, 1.5, 0]  | loss=0.00 target=[-0.5, 1.5, 0.5]         |
| [0, 0, 1] | [0.2, 0.2, 0.8]   | No              | loss=0.12 | loss=0.12 target=[0, 0, 1]       | loss=0.11 target=[0.0667, 0.0667, 1.0667] |
| [0, 0, 1] | [-0.2, 0.2, 0.6]  | No              | loss=0.24 | loss=0.20 target=[-0.2, 0, 1]    | loss=0.18 target=[-0.2, -0.1, 0.9]        |

Xu-Yao Zhang et al. Retargeted Least Squares Regression Algorithm. IEEE Transactions on Neural Network and Learning Systems (T-NNLS), 2015.

# Retargeted LSR

- Target Matrix  $\mathbf{T} \in \mathbb{R}^{n \times c}$

$$\begin{aligned} \min_{\mathbf{W}, \mathbf{b}, \mathbf{T}} \quad & \|\mathbf{XW} + \mathbf{e}_n \mathbf{b}^\top - \mathbf{T}\|_F^2 + \beta \|\mathbf{W}\|_F^2 \\ \text{s.t.} \quad & T_{i, y_i} - \max_{j \neq y_i} T_{i, j} \geq 1, \quad i = 1, 2, \dots, n. \end{aligned}$$

- The constraint flexibility:  $\text{LSR} < \text{DLSR} < \text{ReLSR}$
- LSR/DLSR  $\rightarrow$  decomposed into  $c$  independent sub-problems (one-against-rest)
- Because of  $\mathbf{T}$ , ReLSR is a single and compact machine for multiclass classification

# ReLSR: OPT1: Regression

$$\begin{aligned} \min_{\mathbf{W}, \mathbf{b}, \mathbf{T}} \quad & \|\mathbf{XW} + \mathbf{e}_n \mathbf{b}^\top - \mathbf{T}\|_F^2 + \beta \|\mathbf{W}\|_F^2 \\ \text{s.t.} \quad & T_{i, y_i} - \max_{j \neq y_i} T_{i, j} \geq 1, \quad i = 1, 2, \dots, n. \end{aligned}$$

Regression:  $\min_{\mathbf{W}, \mathbf{b}} \|\mathbf{XW} + \mathbf{e}_n \mathbf{b}^\top - \mathbf{T}\|_F^2 + \beta \|\mathbf{W}\|_F^2$

$$\mathbf{W} = (\mathbf{X}^\top \mathbf{H} \mathbf{X} + \beta \mathbf{I}_d)^{-1} \mathbf{X}^\top \mathbf{H} \mathbf{T}, \quad \mathbf{b} = \frac{(\mathbf{T} - \mathbf{XW})^\top \mathbf{e}_n}{n}$$

# ReLSR: OPT2: Retargeting

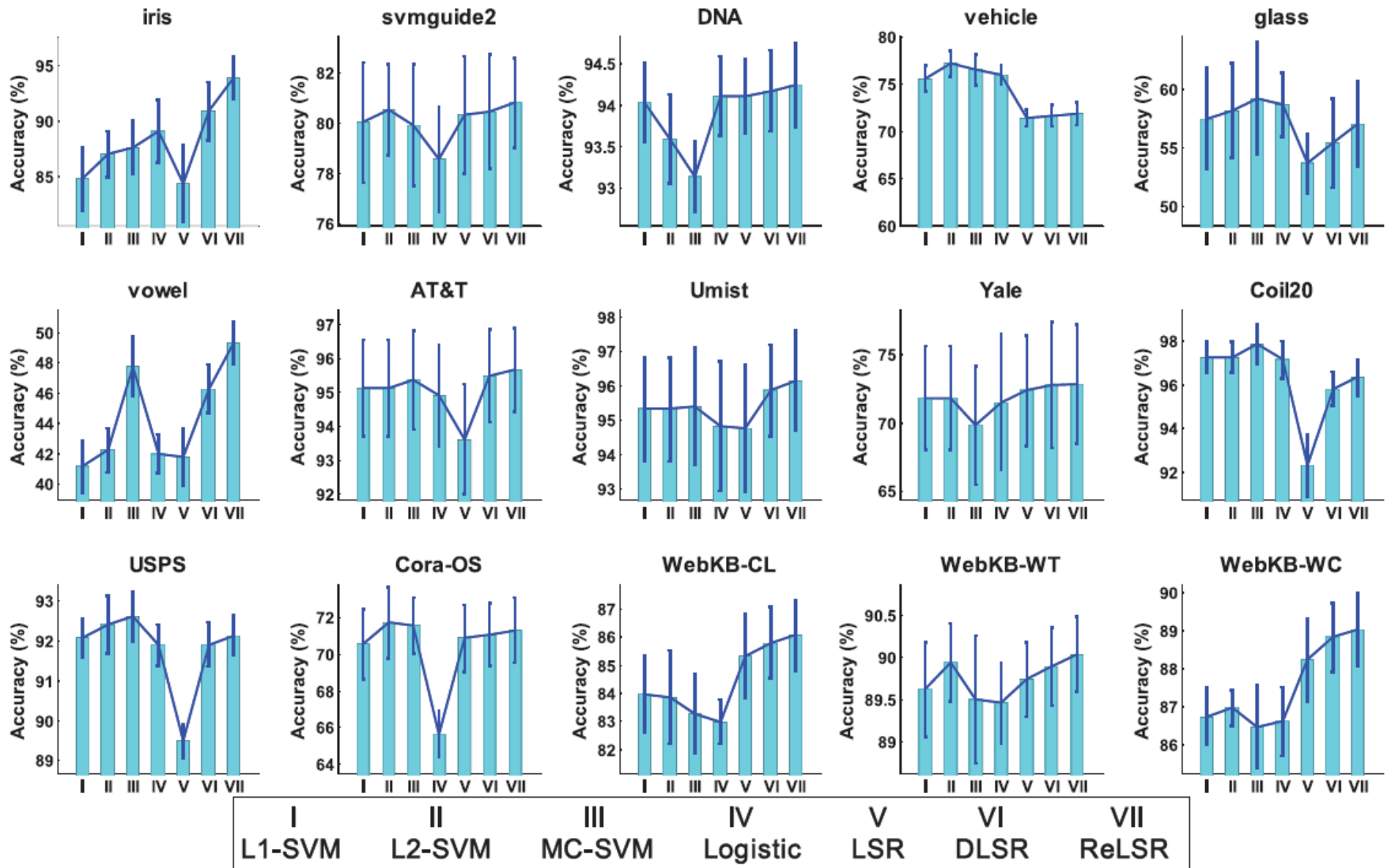
$$\begin{aligned} \min_{\mathbf{W}, \mathbf{b}, \mathbf{T}} \quad & \|\mathbf{X}\mathbf{W} + \mathbf{e}_n \mathbf{b}^\top - \mathbf{T}\|_F^2 + \beta \|\mathbf{W}\|_F^2 \\ \text{s.t.} \quad & T_{i, y_i} - \max_{j \neq y_i} T_{i, j} \geq 1, \quad i = 1, 2, \dots, n. \end{aligned}$$

Retargeting: 
$$\begin{aligned} \min_{\mathbf{T}} \quad & \|\mathbf{X}\mathbf{W} + \mathbf{e}_n \mathbf{b}^\top - \mathbf{T}\|_F^2 = \|\mathbf{R} - \mathbf{T}\|_F^2 \\ \text{s.t.} \quad & T_{i, y_i} - \max_{j \neq y_i} T_{i, j} \geq 1, \quad i = 1, 2, \dots, n. \end{aligned}$$

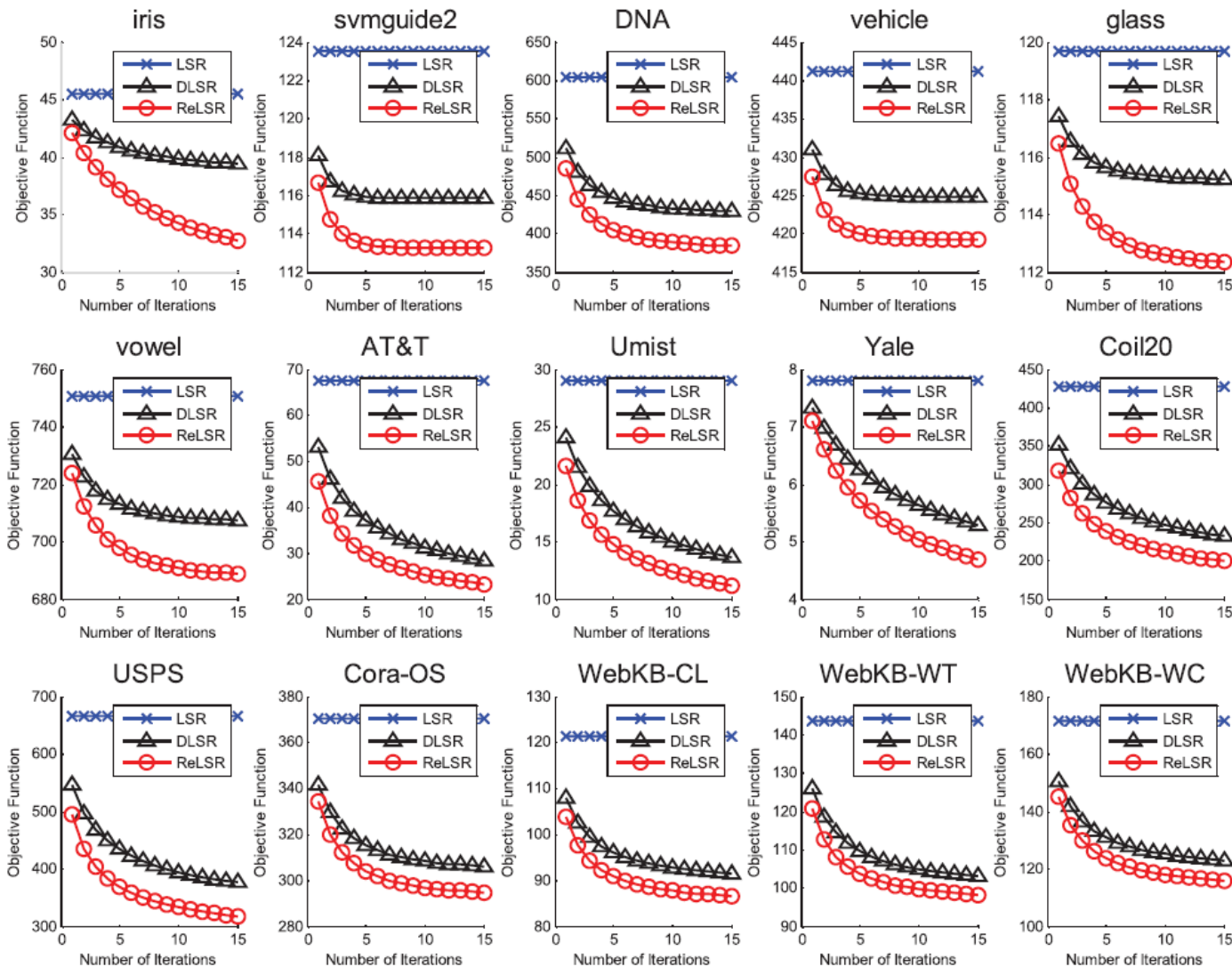
$$\min_{\mathbf{t}} \|\mathbf{r} - \mathbf{t}\|_2^2 = \sum_{i=1}^c (r_i - t_i)^2 \quad \text{s.t.} \quad t_k - \max_{i \neq k} t_i \geq 1.$$



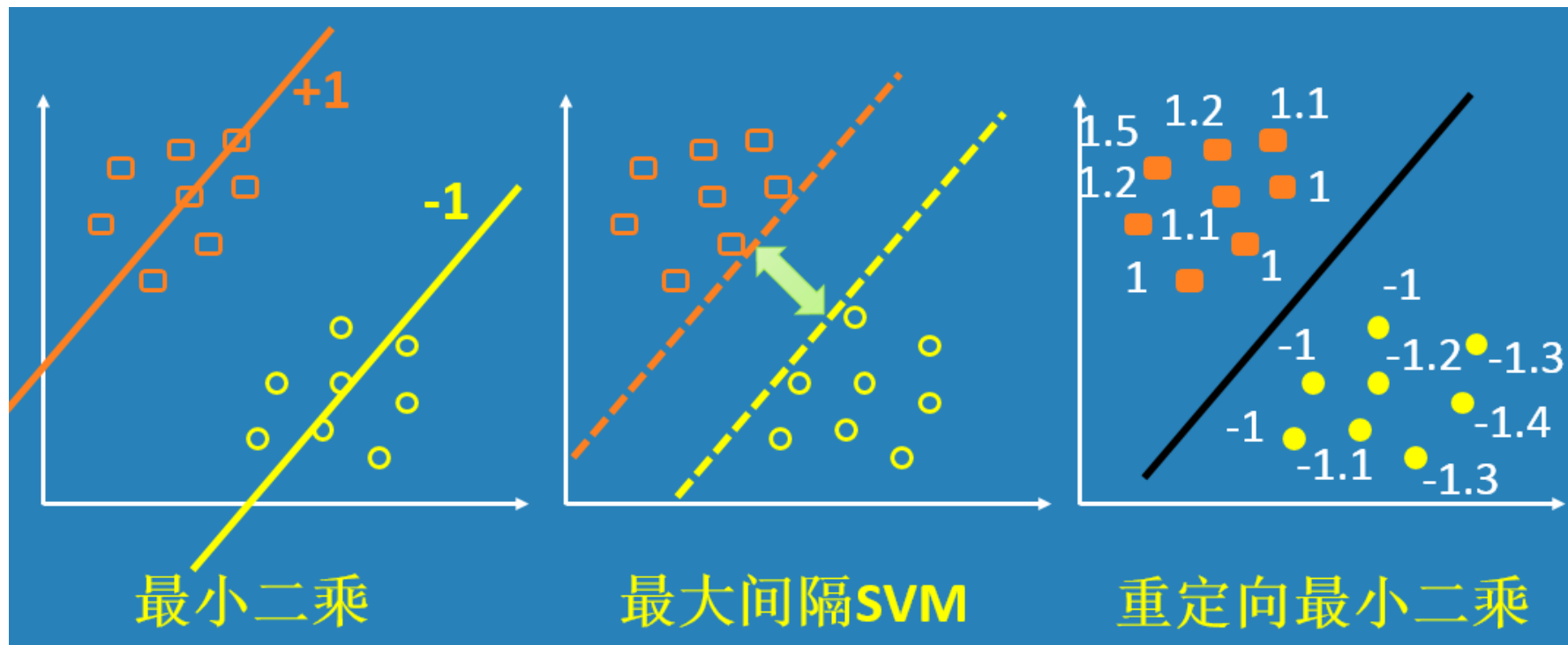
# Comparison



# Comparison

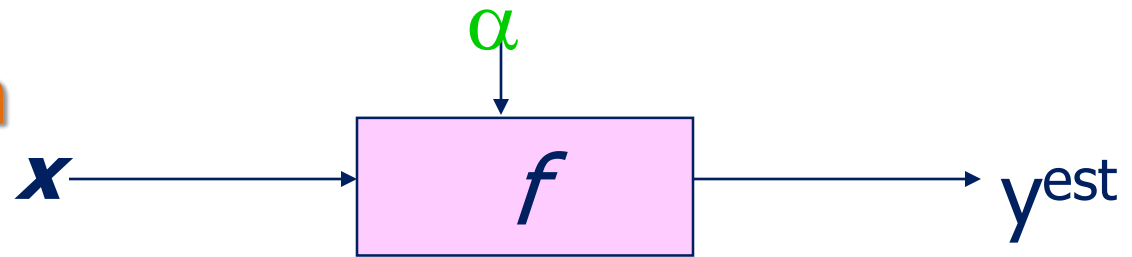


# From Regression to Classification

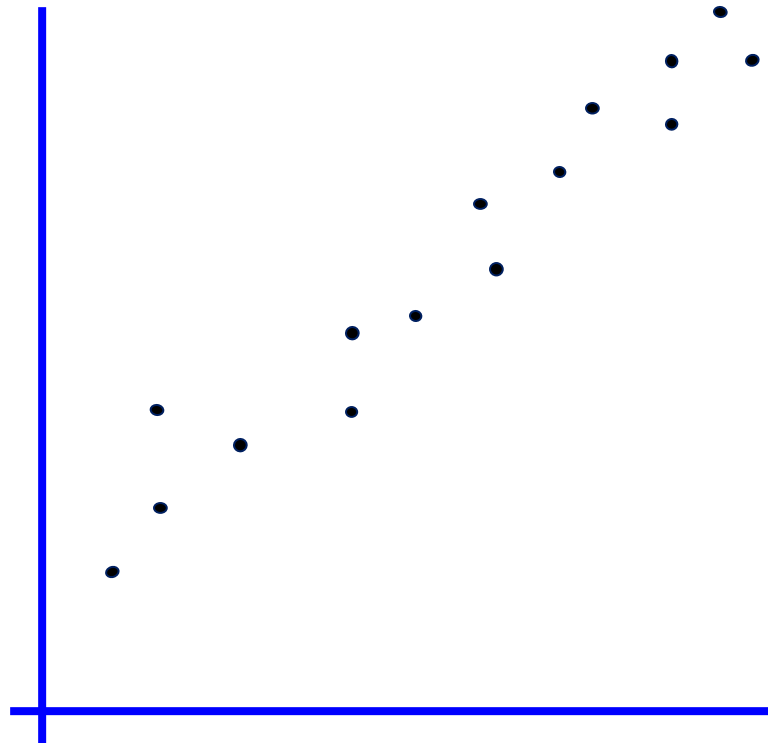


# Support Vector Regression

# Linear Regression

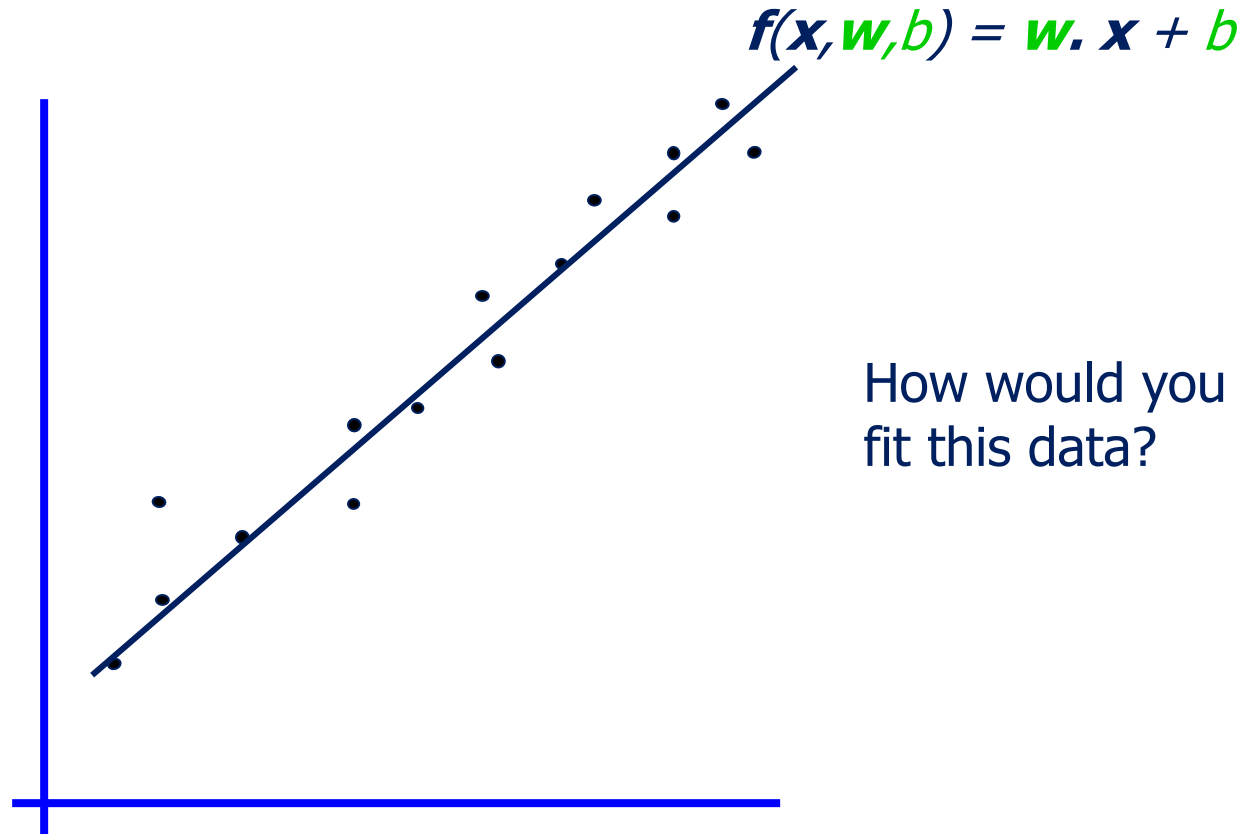
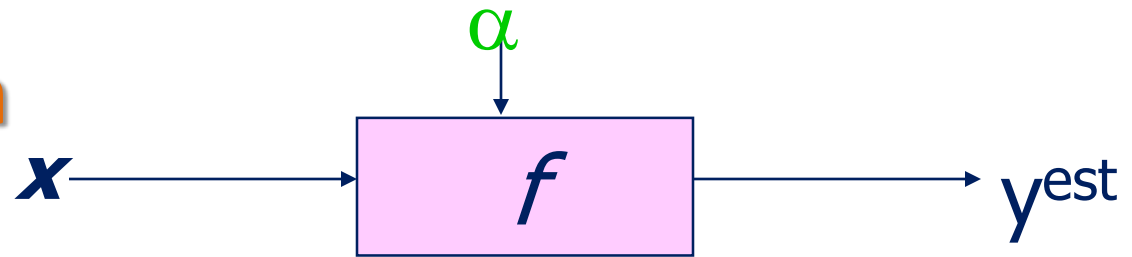


$$f(\mathbf{x}, \mathbf{w}, b) = \mathbf{w} \cdot \mathbf{x} + b$$

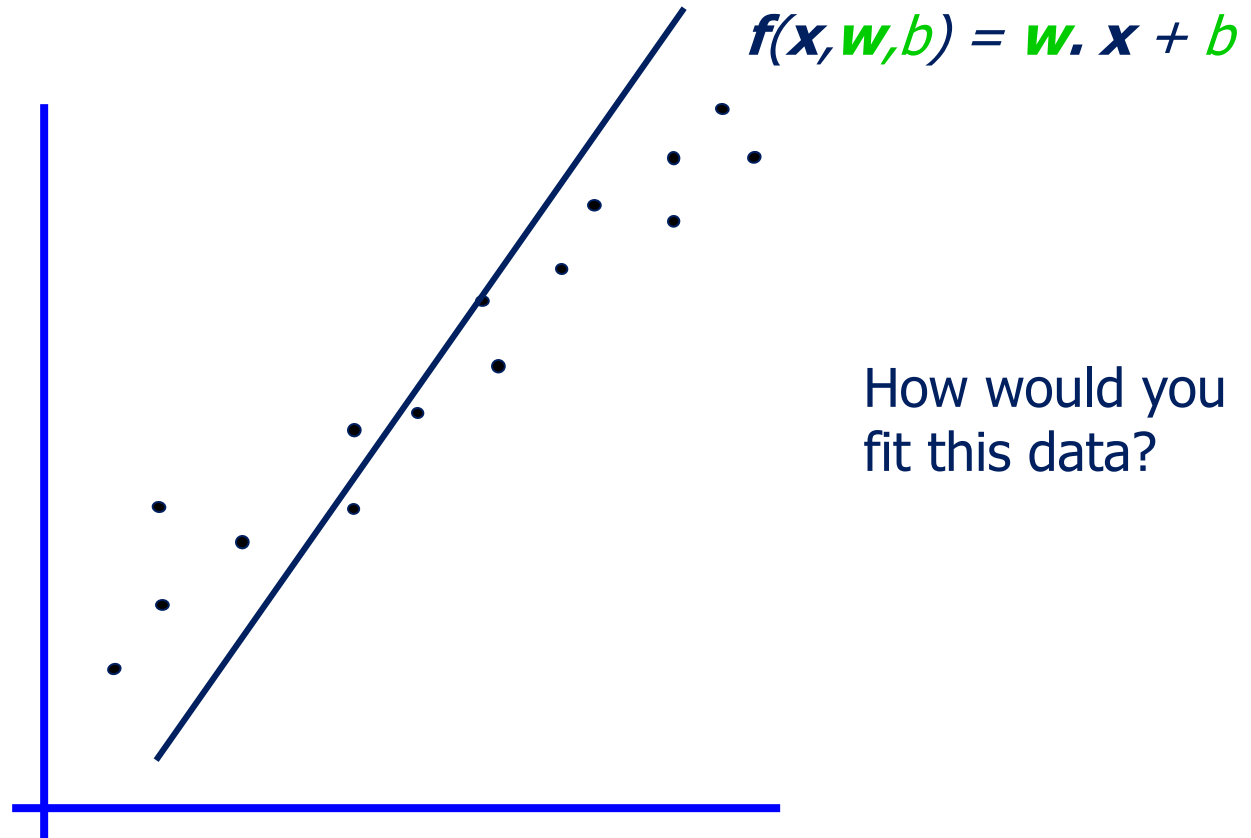
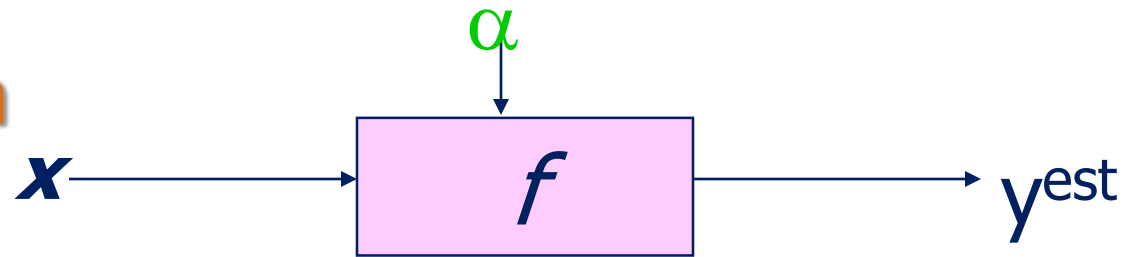


How would you fit this data?

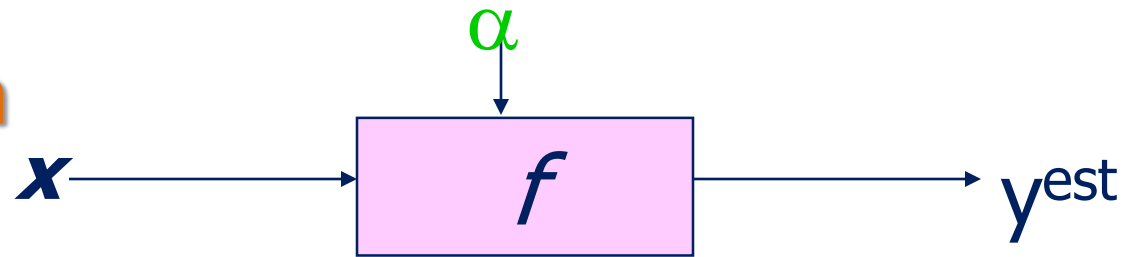
# Linear Regression



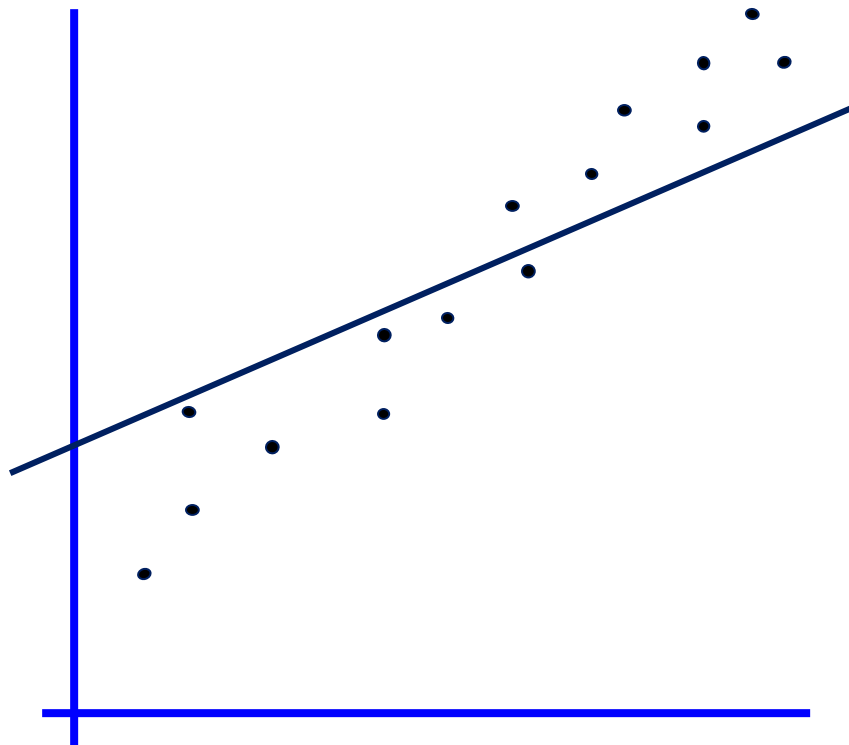
# Linear Regression



# Linear Regression



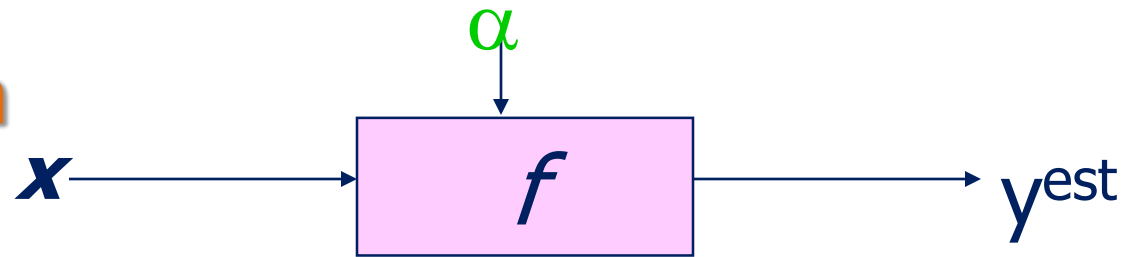
$$f(x, w, b) = w \cdot x + b$$



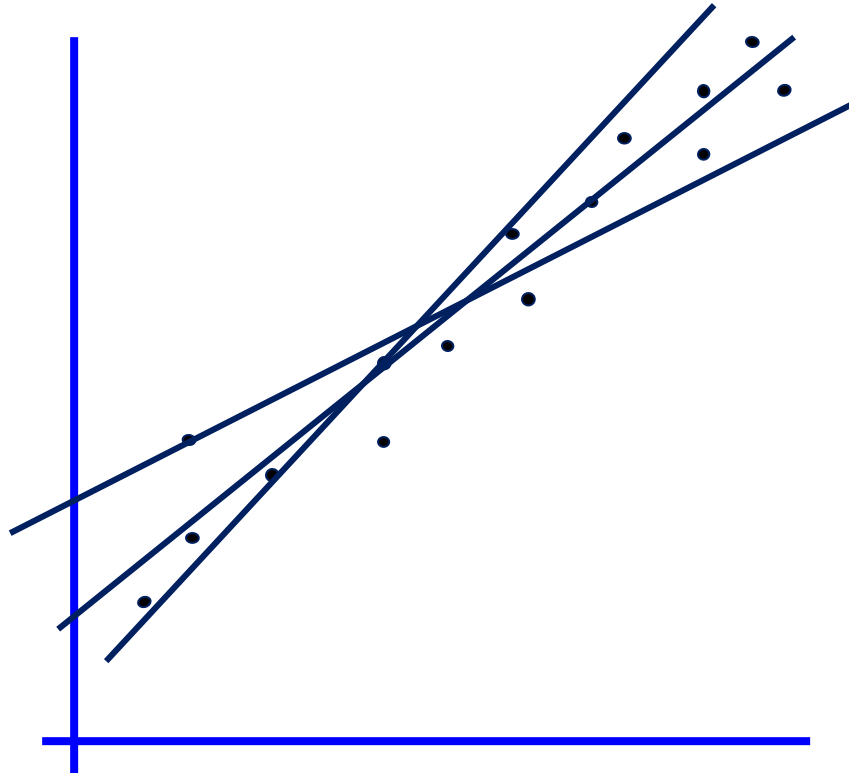
How would you fit this data?



# Linear Regression



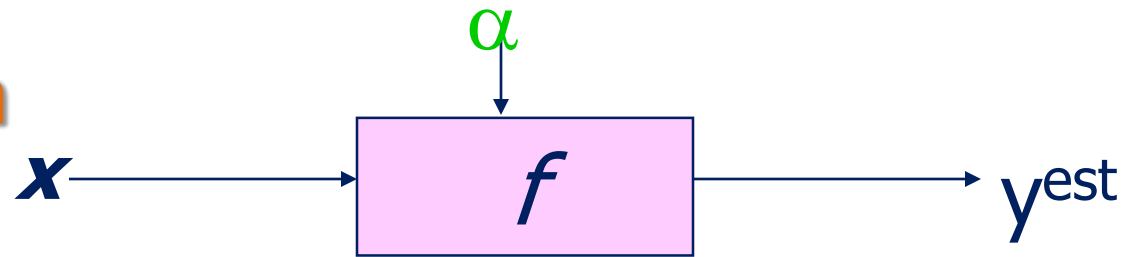
$$f(\mathbf{x}, \mathbf{w}, b) = \mathbf{w} \cdot \mathbf{x} + b$$



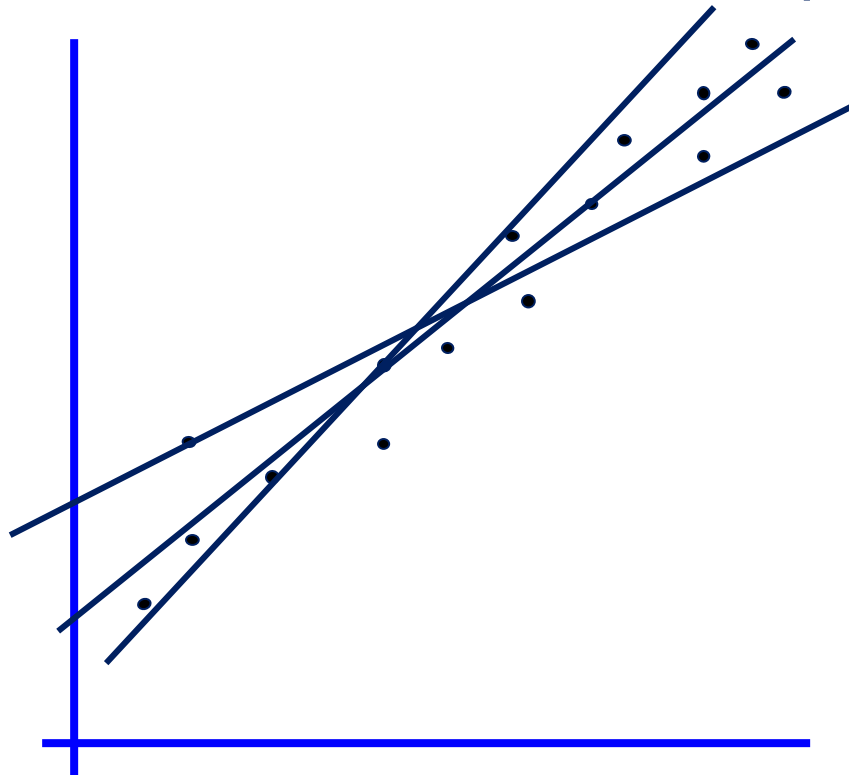
Any of these  
would be fine..

..but which is  
best?

# Linear Regression

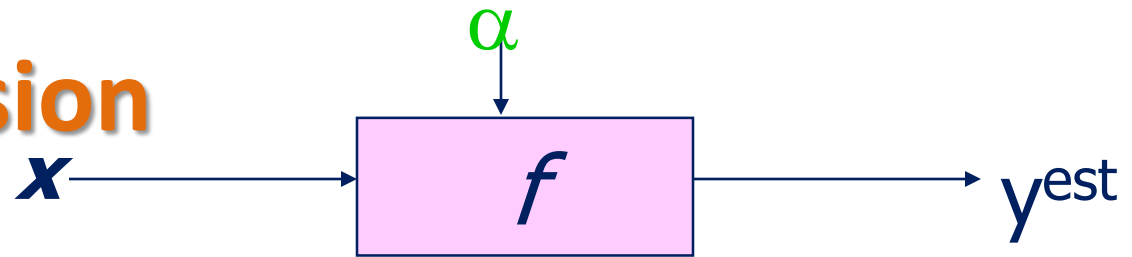


$$f(\mathbf{x}, \mathbf{w}, b) = \mathbf{w} \cdot \mathbf{x} + b$$

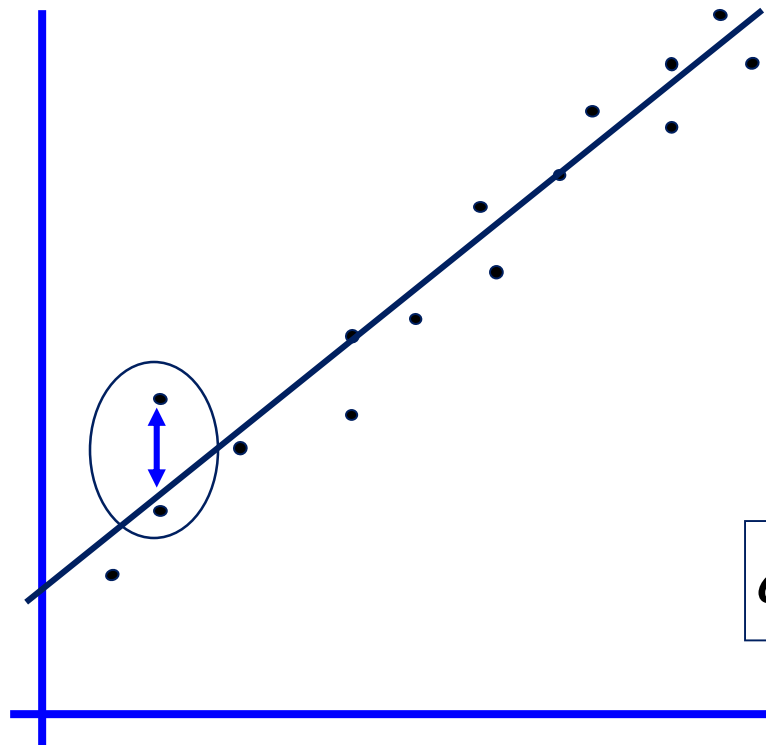


How to define the  
fitting error of a  
linear regression ?

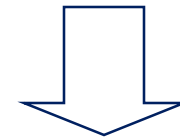
# Linear Regression



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



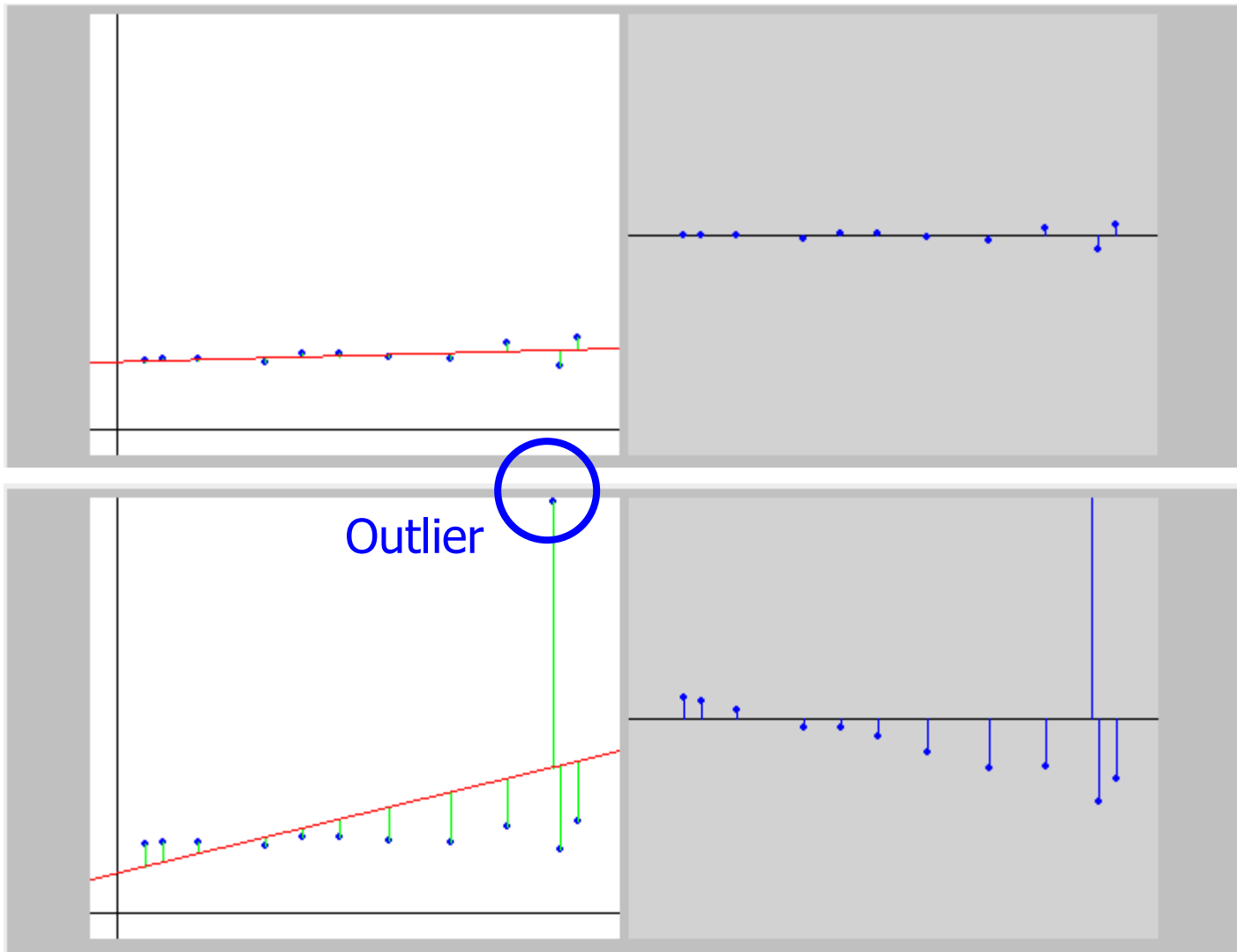
How to define the  
fitting error of a  
linear regression ?



$$err_i = (w \cdot x_i - b - y_i)^2$$

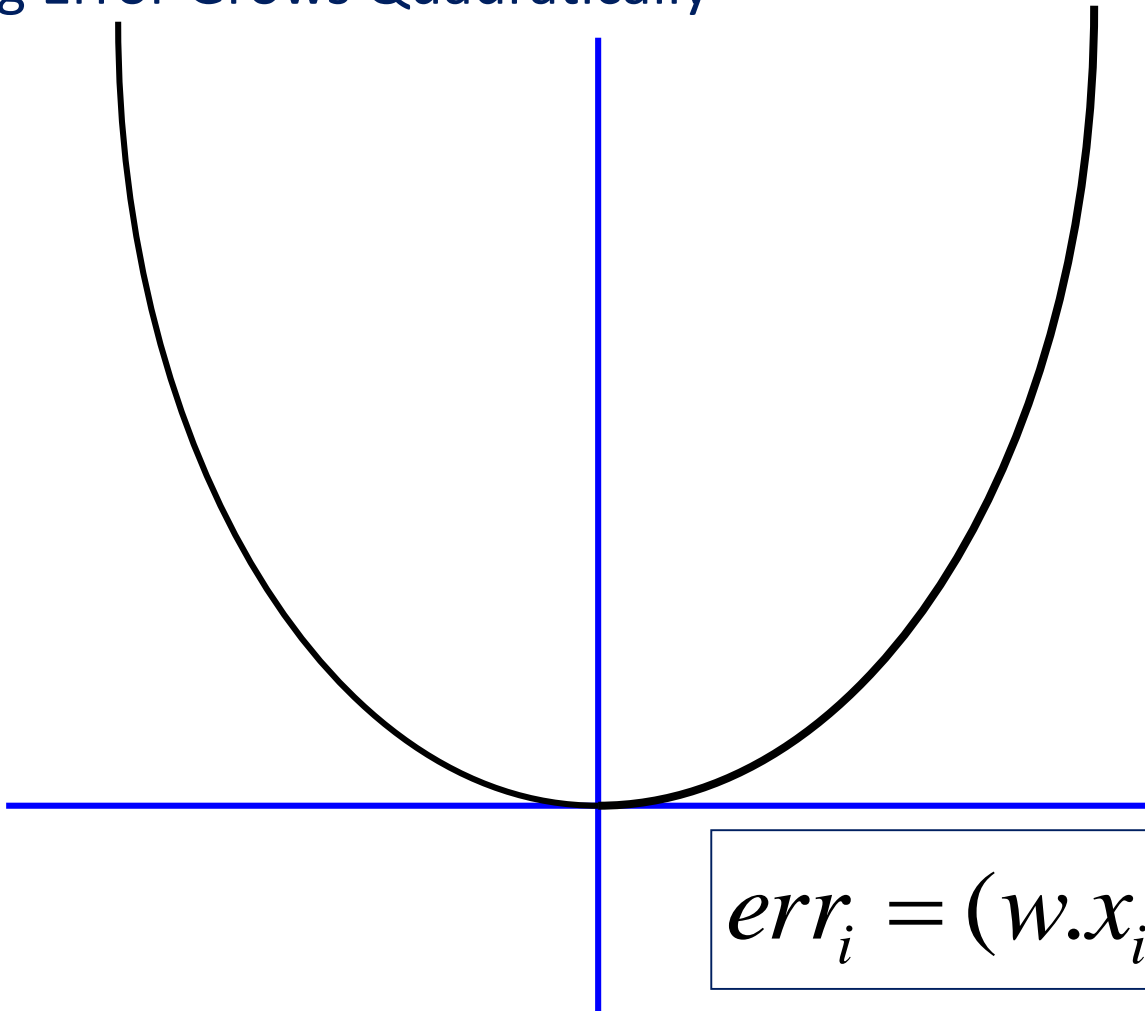
Squared-Loss

# Sensitive to Outliers



# Why?

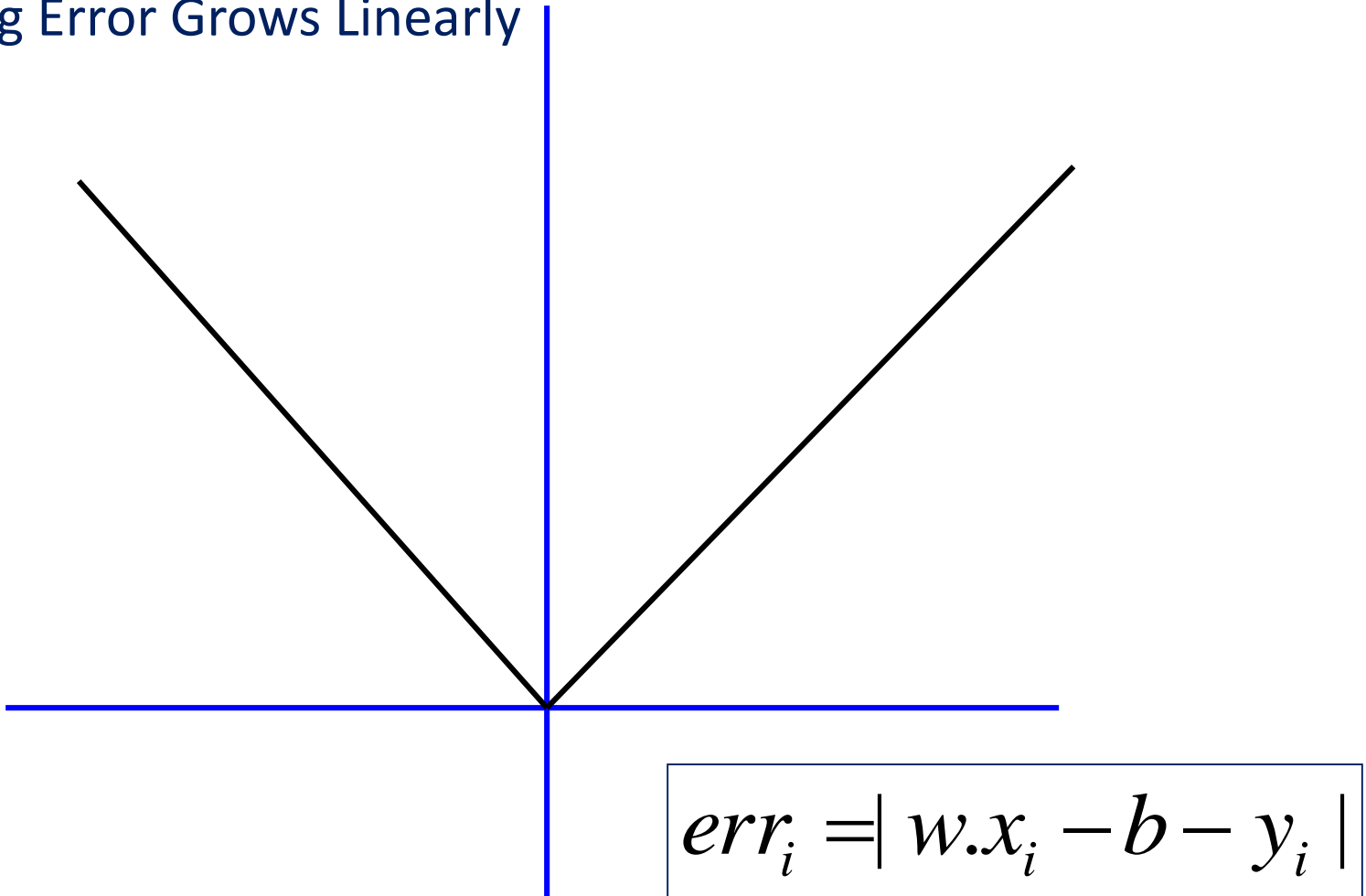
- Squared-Loss Function
  - Fitting Error Grows Quadratically



$$err_i = (w.x_i - b - y_i)^2$$

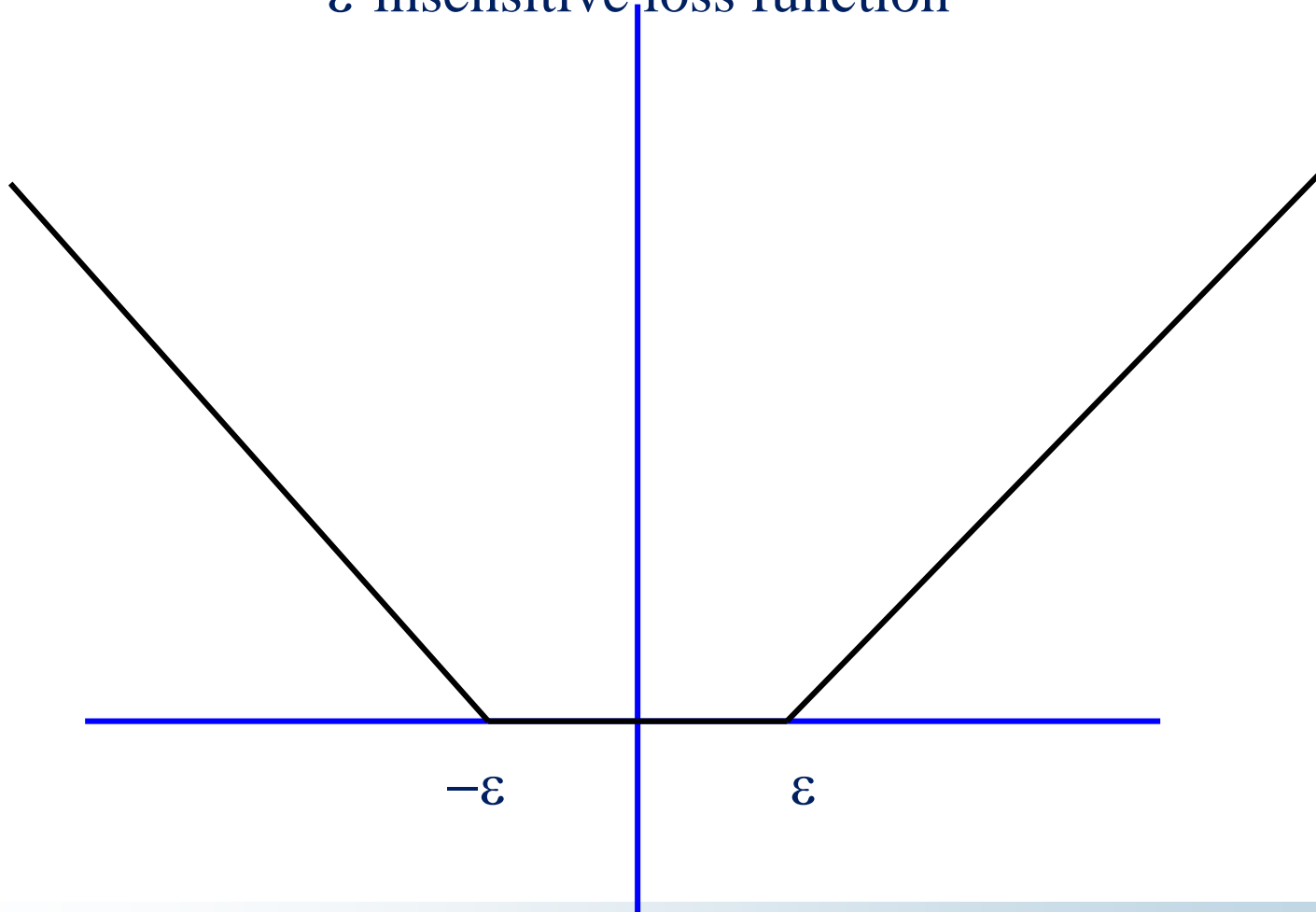
# How about Linear-Loss ?

- Linear-Loss Function
  - Fitting Error Grows Linearly



# Actually

- SVR uses the Loss Function below  
 $\epsilon$ -insensitive loss function



# Epsilon Support Vector Regression ( $\epsilon$ -SVR)

- Given: a data set  $\{x_1, \dots, x_n\}$  with target values  $\{u_1, \dots, u_n\}$ , we want to do  $\epsilon$ -SVR
- The optimization problem is

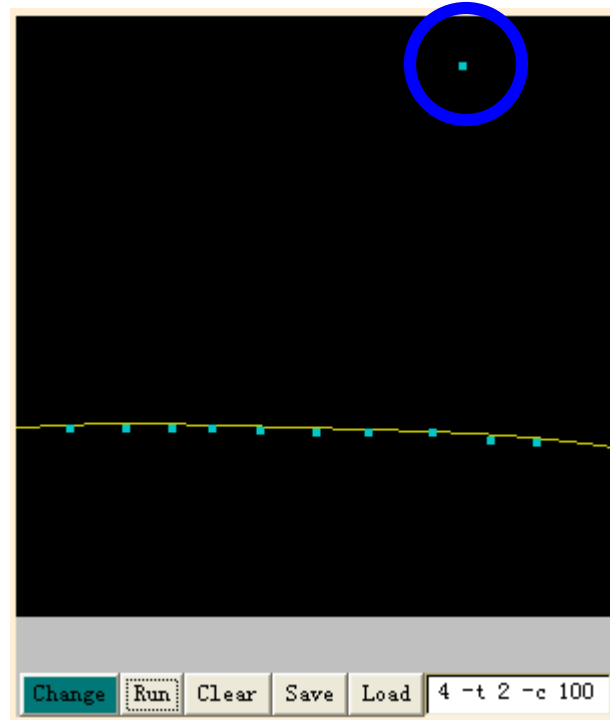
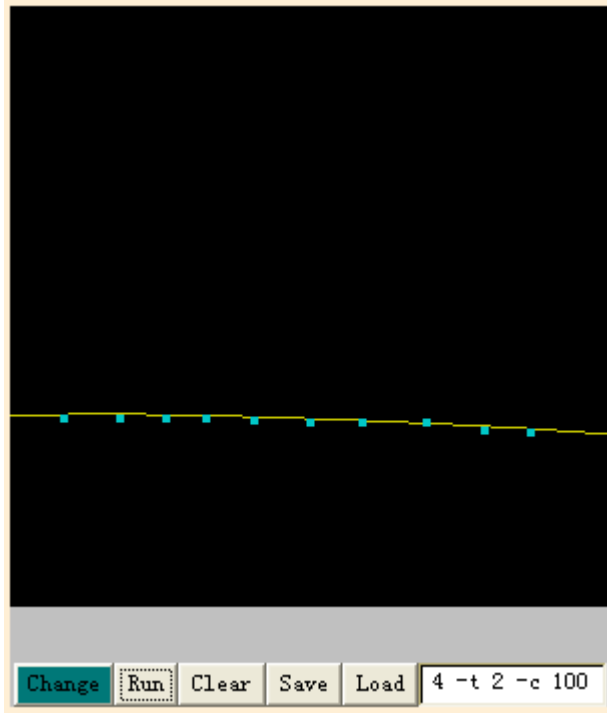
$$\begin{aligned} & \text{Min } \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n (\xi_i + \xi_i^*) \\ & \text{subject to } \begin{cases} u_i - w^T x_i - b \leq \epsilon + \xi_i \\ w^T x_i + b - u_i \leq \epsilon + \xi_i^* \\ \xi_i \geq 0, \xi_i^* \geq 0 \end{cases} \end{aligned}$$

- Similar to SVM, this can be solved as a quadratic programming problem



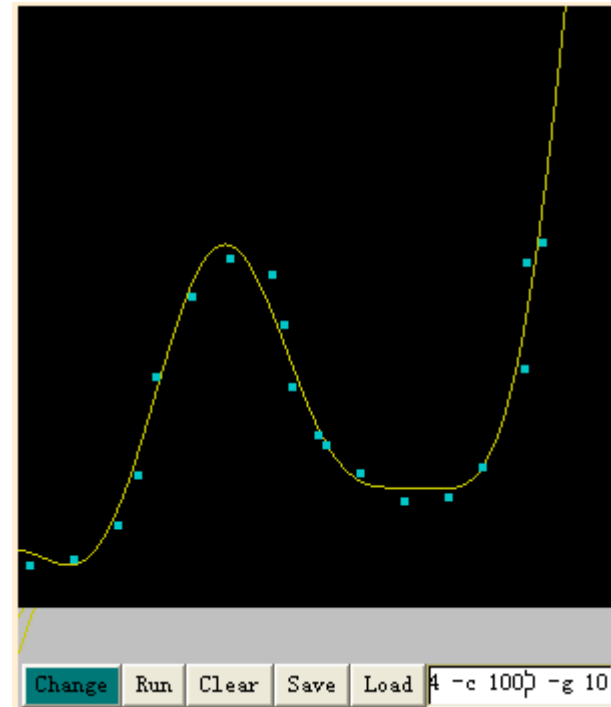
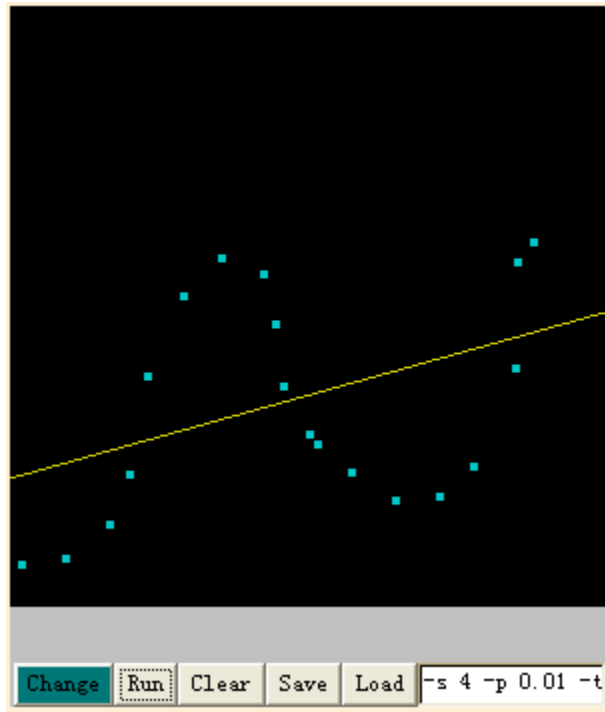
# Online Demo

- Less Sensitive to Outlier



# Again, Extend to Non-Linear Case

- Similar with SVM



# Large Margin Nearest Neighbor (LMNN)

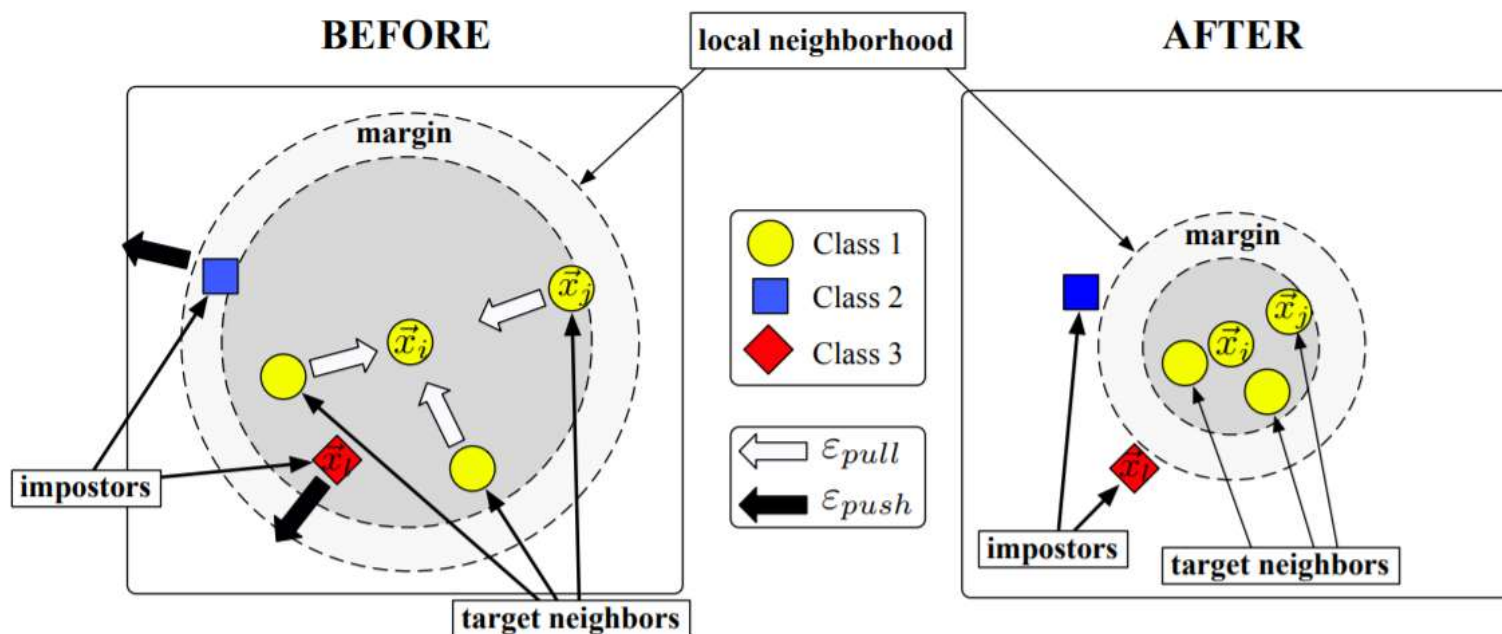
# LMNN

[PDF] Distance Metric Learning for Large Margin Nearest Neighbor ...

[citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.157.9469&rep...](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.157.9469&rep...) ▼ 翻译此页

作者：KQ Weinberger - 2009 - 被引用次数：1691 - 相关文章

Journal of Machine Learning Research 10 (2009) 207-244. Submitted 12/07; Revised 9/08; Published 2/09. Distance Metric Learning for **Large Margin. Nearest Neighbor** Classification. Kilian Q. Weinberger. KILIAN@YAHOO-INC.COM. Yahoo! Research. 2821 Mission College Blvd. Santa Clara, CA 9505. Lawrence K. Saul.



# LMNN

- Penalize large distances between each input and its target neighbors

$$\epsilon_{\text{pull}}(\mathbf{L}) = \sum_{j \rightsquigarrow i} \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2.$$

- Penalize small distances between differently labeled examples

$y_{il} = 1$  if and only if  $y_i = y_l$ , and  $y_{il} = 0$  otherwise.

$$\epsilon_{\text{push}}(\mathbf{L}) = \sum_{i, j \rightsquigarrow i} \sum_l (1 - y_{il}) \left[ 1 + \|\mathbf{L}(\vec{x}_i - \vec{x}_j)\|^2 - \|\mathbf{L}(\vec{x}_i - \vec{x}_l)\|^2 \right]_+$$

- The total loss  $\epsilon(\mathbf{L}) = (1 - \mu) \epsilon_{\text{pull}}(\mathbf{L}) + \mu \epsilon_{\text{push}}(\mathbf{L}).$

- Objective

- KNN always belong to the same class
- Examples from different classes are separated by a large margin

# LMNN

- **Target neighbors**: selected before learning. Each instance has exactly  $k$  different target neighbors, which all share the same class label
- **Impostors**: another data point (nearest neighbors) with a different class label
- **Objective**: minimize the number of impostors for all data instances

$$d(\vec{x}_i, \vec{x}_j) = (\vec{x}_i - \vec{x}_j)^\top \mathbf{M}(\vec{x}_i - \vec{x}_j).$$

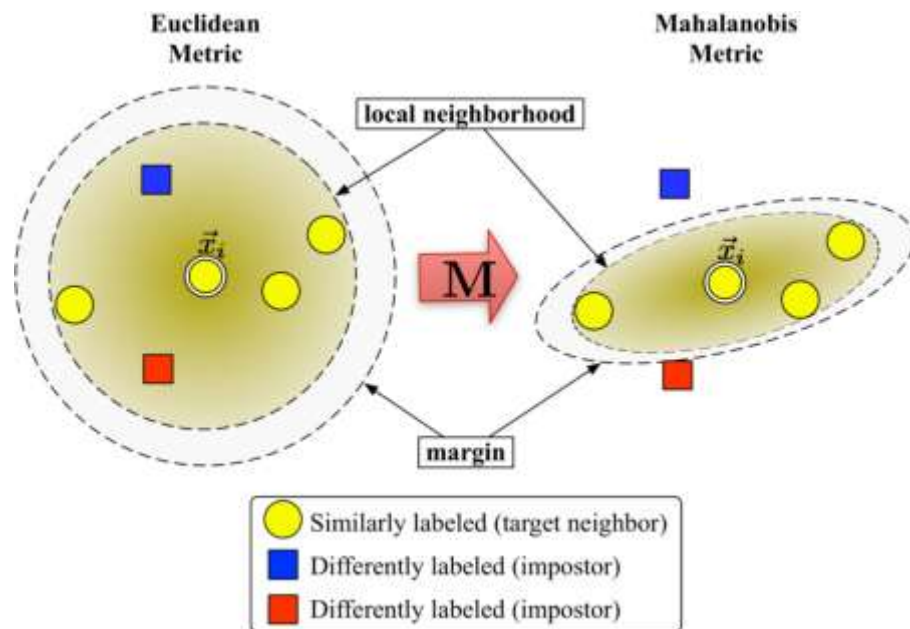
$$\min_{\mathbf{M}} \sum_{i,j \in N_i} d(\vec{x}_i, \vec{x}_j) + \sum_{i,j,l} \xi_{ijl}$$

$$\forall_{i,j \in N_i, l, y_l \neq y_i}$$

$$d(\vec{x}_i, \vec{x}_j) + 1 \leq d(\vec{x}_i, \vec{x}_l) + \xi_{ijl}$$

$$\xi_{ijl} \geq 0$$

$$\mathbf{M} \succeq 0$$



# Conclusion

- Structural Risk Minimization → VC Dimension
- Large Margin → Low VC Dimension
- Hard Margin SVM → Soft Margin SVM
- Dual Problem → Kernel Method
- Model Selection → Tradeoff C and Kernel
- Optimization → SMO
- Geometry of SVM → Reduced Convex Hull
- Multi-Class SVM → Binary or Single-Machine
- Locally Linear SVM → Nonlinear Classifier without Kernel
- Least Squares Extensions → Another Perspective
- Large Margin for Regression, Ranking, Nearest Neighbors
- Ideas of: large margin, hinge loss, kernel method, are widely used in different PR&ML tasks

***Thank You!***  
***Q&A***