第9章：支撑向量机与核方法
Part 2

张煦尧(xyz@nlpr.ia.ac.cn)

2017年12月20日

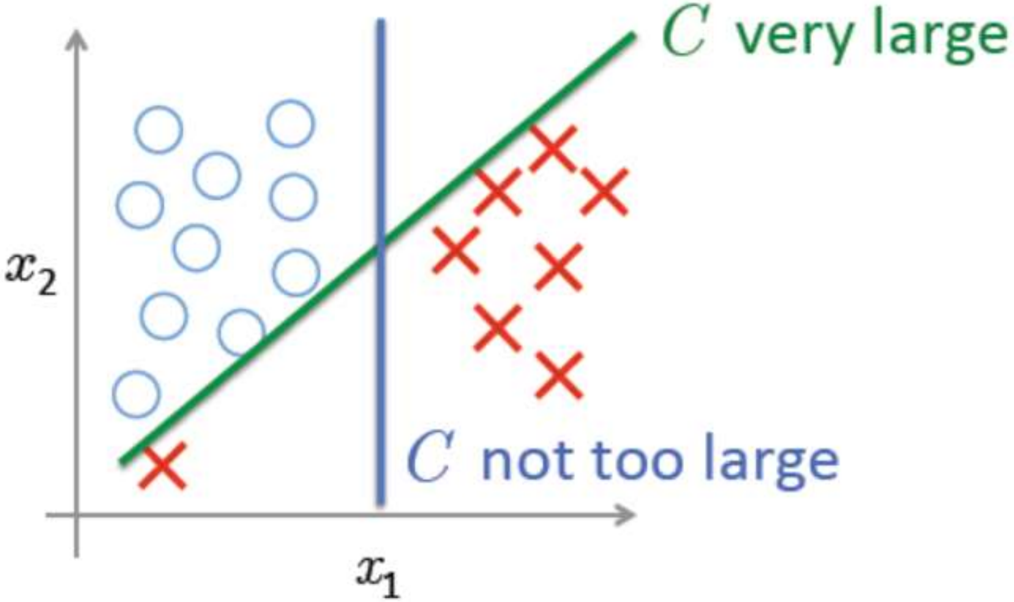
助教：何文浩(wenhao.he@nlpr.ia.ac.cn)
杨红明(hongming.yang@nlpr.ia.ac.cn)

中国科学院大学
University of Chinese Academy of Sciences

Model Selection

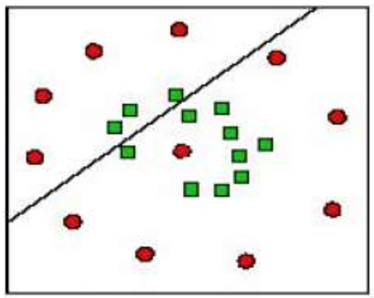
The "C" Problem

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



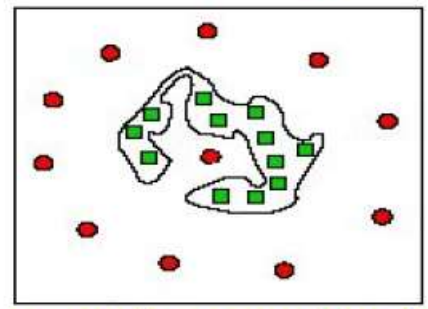
Overfitting and Underfitting

Under-Fitting

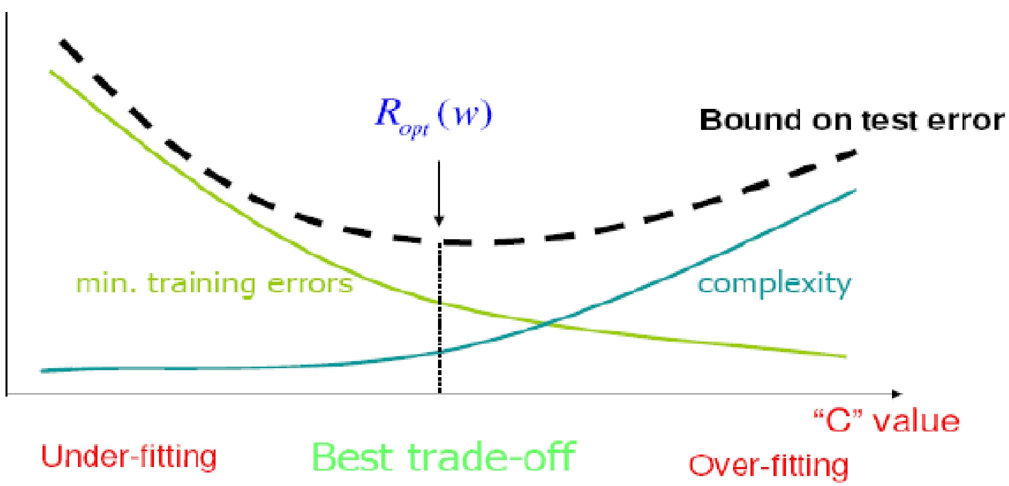


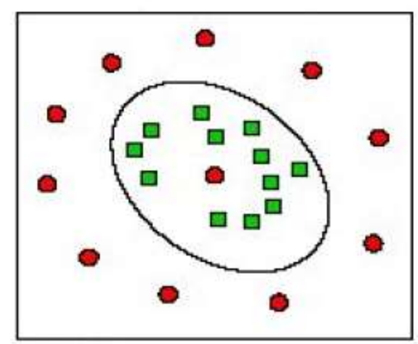
Too much simple!

Over-Fitting



Too much complicated!





Trade-Off

Model Selection (C and Kernel)

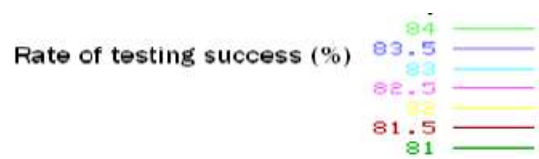
- In practice a Gaussian radial basis or a low degree polynomial kernel is a good start
- Checking which set of parameters (such as C or detla if we choose RBF kernel) are the most appropriate by Cross-Validation (K- fold):
  - divide randomly all the available training examples into K equal-sized subsets
  - 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

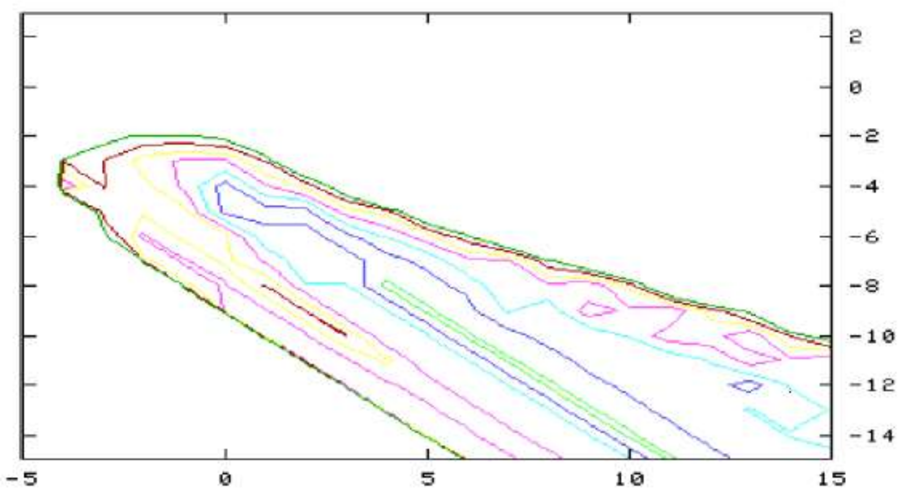
Model Selection Example

- 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

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

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





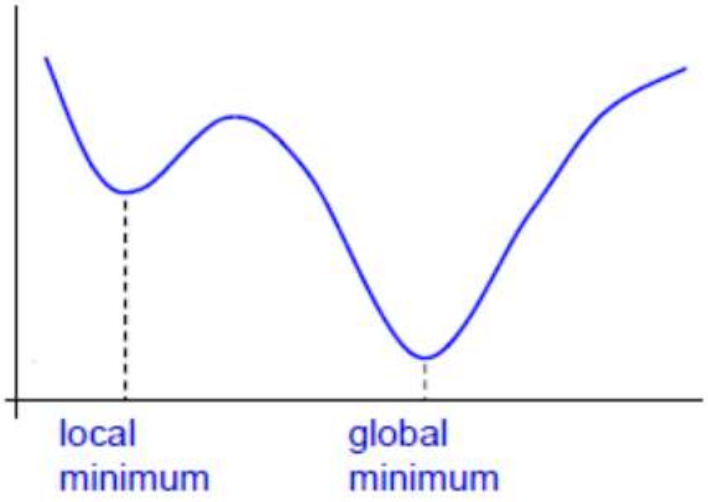
$C = 2^{5}$, $\sigma = 2^{-9}$

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

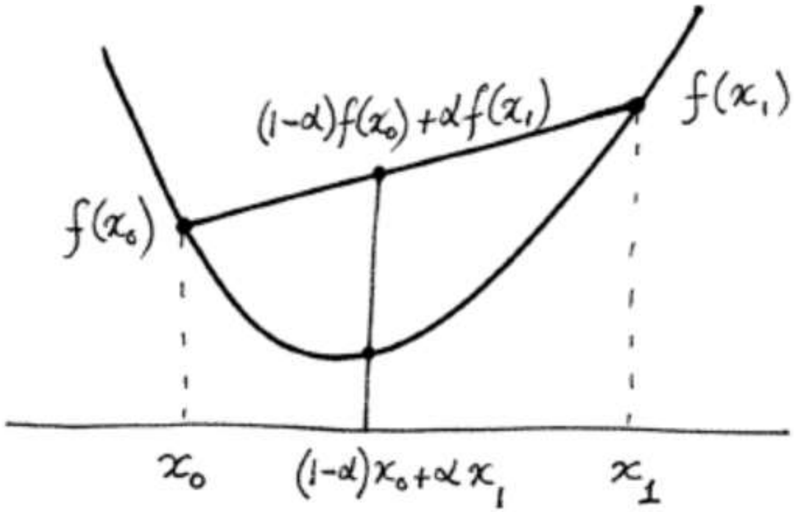
Convex Function

D – a domain in $\mathbb{R}^n$.

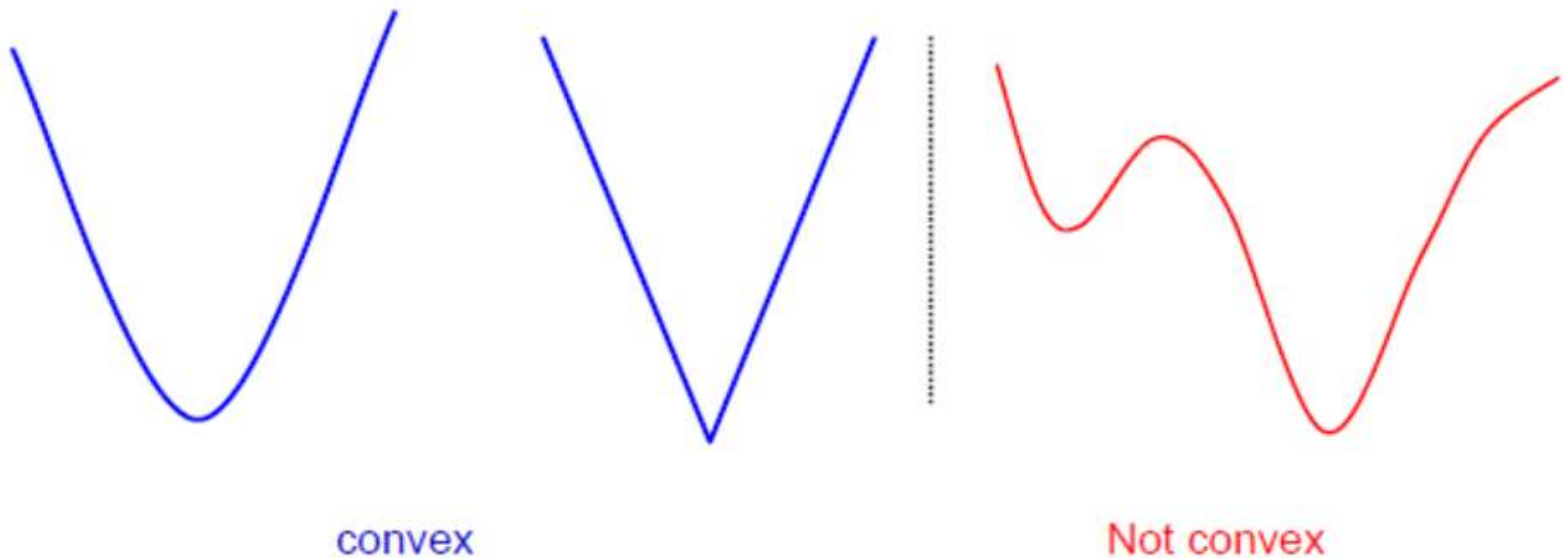
A convex function $f : D \to \mathbb{R}$ is one that satisfies, for any $\mathbf{x}_0$ and $\mathbf{x}_1$ in D:

$$f((1-\alpha)\mathbf{x}_0 + \alpha\mathbf{x}_1) \le (1-\alpha)f(\mathbf{x}_0) + \alpha f(\mathbf{x}_1) \quad .$$

Line joining $(\mathbf{x}_0, f(\mathbf{x}_0))$ and $(\mathbf{x}_1, f(\mathbf{x}_1))$ lies above the function graph.



Convex Function Examples



A non-negative sum of convex functions is convex

SVM is Convex

SVM

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

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
  - 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 (αi,αj) 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

    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

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

– Sequential Minimal Optimization (SMO)

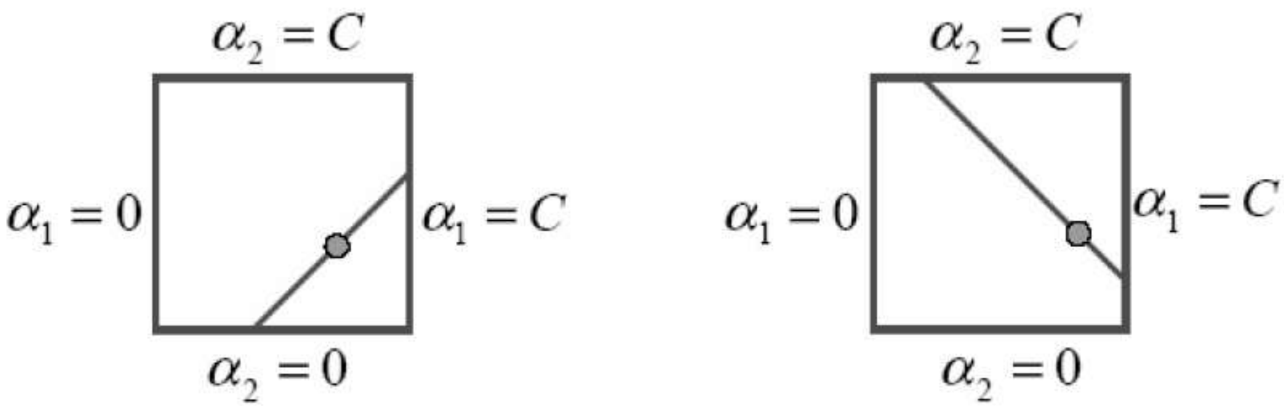
Each time, choose two samples violating the KKT condition and updating the weights until all the points satisfied KKT condition

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

SMO

Constraints on the Lagrange Multipliers

- Bound constraints $0 \leq \alpha_i \leq C$
- Linear equality constraint $\Sigma\, \alpha_i\, y_i = 0$



$y_1 \neq y_2 \Rightarrow \alpha_1 - \alpha_2 = \gamma$ $\qquad$ $y_1 = y_2 \Rightarrow \alpha_1 + \alpha_2 = \gamma$

SMO

$$\min_{\alpha_1\alpha_2} \quad \frac{1}{2}k_{11}{\alpha_1}^2 + \frac{1}{2}k_{22}{\alpha_2}^2 + y_1y_2k_{12}\alpha_1\alpha_2 - (\alpha_1 + \alpha_2) + constant$$

$$s.t. \begin{cases} y_1\alpha_1 + y_2\alpha_2 = -\sum_{i\neq 1,2} \alpha_i y_i = constant \\ 0 \le \alpha_1, \alpha_2 \le C \end{cases}$$

消去a1,不考虑0<=a2<=C约束

$${\alpha_2}^{new} = \alpha_2 + \frac{y_2(E_1 - E_2)}{\eta}$$

$$E_i = f_i - y_i$$

$$\eta = k(x_1, x_1) + k(x_2, x_2) - 2k(x_1, x_2)$$

再考虑0<=a2<=C约束

$${\alpha_2}^{new,clipped} = \begin{cases} H & if\ \ {\alpha_2}^{new} \ge H \\ L & if\ \ {\alpha_2}^{new} \le L \\ {\alpha_2}^{new} & else \end{cases}$$

$$\begin{cases} L = max(0, \alpha_2 - \alpha_1), H = min(C, C + \alpha_2 - \alpha_1) & if\ y_1 \neq y_2 \\ L = max(0, \alpha_2 + \alpha_1 - C), H = min(C, \alpha_2 + \alpha_1) & if\ y_1 = y_2 \end{cases}$$

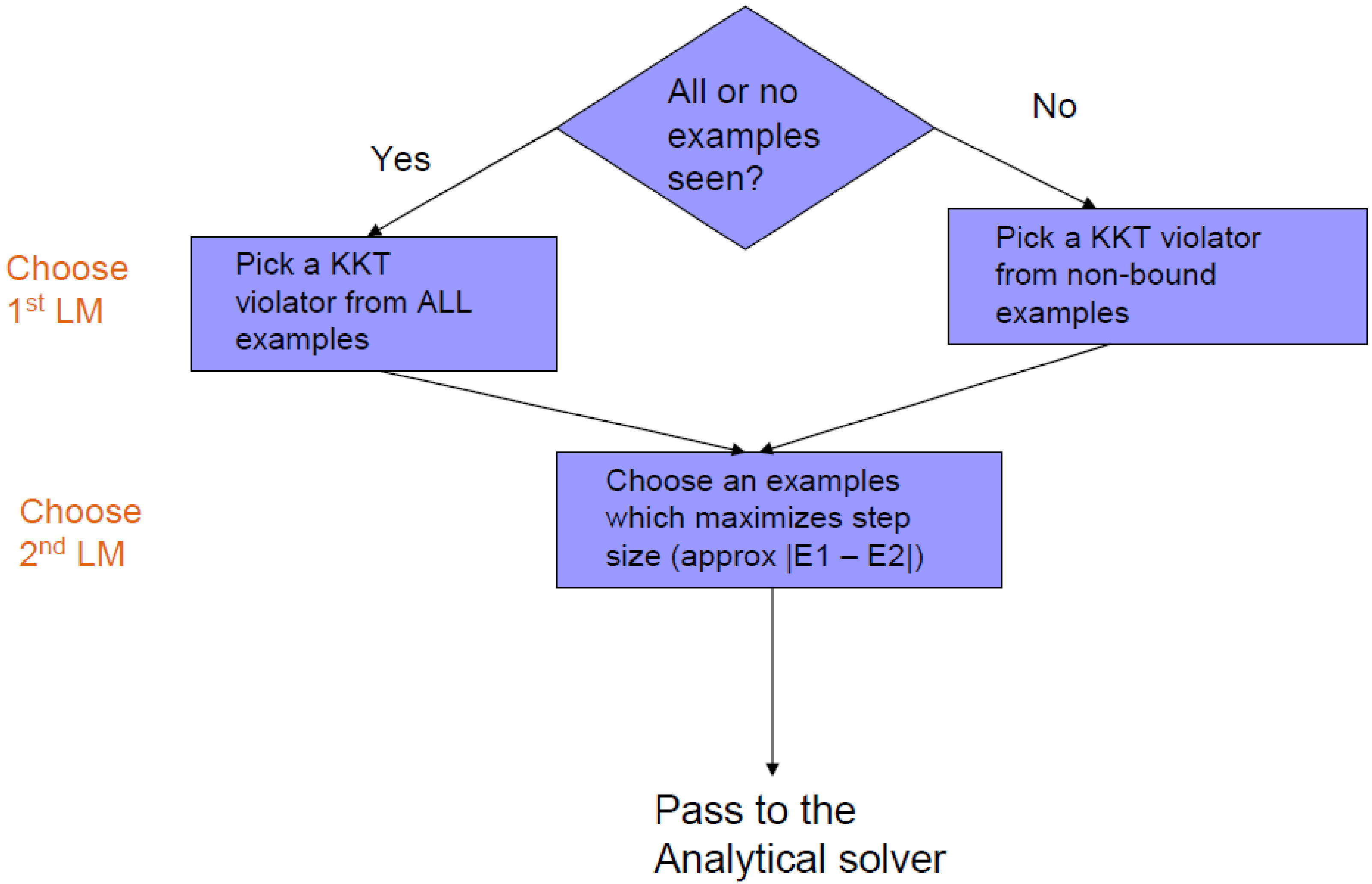
更新a1

$${\alpha_1}^{new} = \alpha_1 + y_1y_2(\alpha_2 - {\alpha_2}^{new,clipped})$$

SMO

- **选取两个Langrange乘子变量的启发式规则：**
- **第一个Langrange乘子：**
  - 任何违反KKT条件的乘子(或样本)都是合法的第一个Langrange乘子。
  - 第一个Langrange乘子的选择构成SMO算法的外层循环。
  - 外层循环分两种情况：
    - The outer loop makes repeated passes over the non-bound examples ($\alpha_i = 0\ or\ C$) until all of the non-bound examples obey the KKT conditions within ε (e.g. 10^{-3})
    - The outer loop then goes back and iterates over the entire training set.
    - The outer loop keeps alternating between **single passes over the entire training set** and **multiple passes over the non-bound subset** until the entire training set obeys the KKT condition within ε, whereupon the algorithm terminates.
- **第二个Langrange乘子的选择：**
  - 与第一个乘子的结合，应该使第二个乘子的迭代步长较大,第二个乘子的迭代步长大致正比于$|E_1 - E_2|$,因此应选择第二个乘子最大化$|E_1 - E_2|$，即当E_1为正时，最小化E_2，当E_1为负时，最大化E_2.
- 很多文献改进方法关于如何选择两个乘子变量
  - Maximal violating pair
  - Second order information

SMO



Sequential Minimal Optimization

Geometry of SVM

Another Interpretation of SVM

Hard-margin SVM

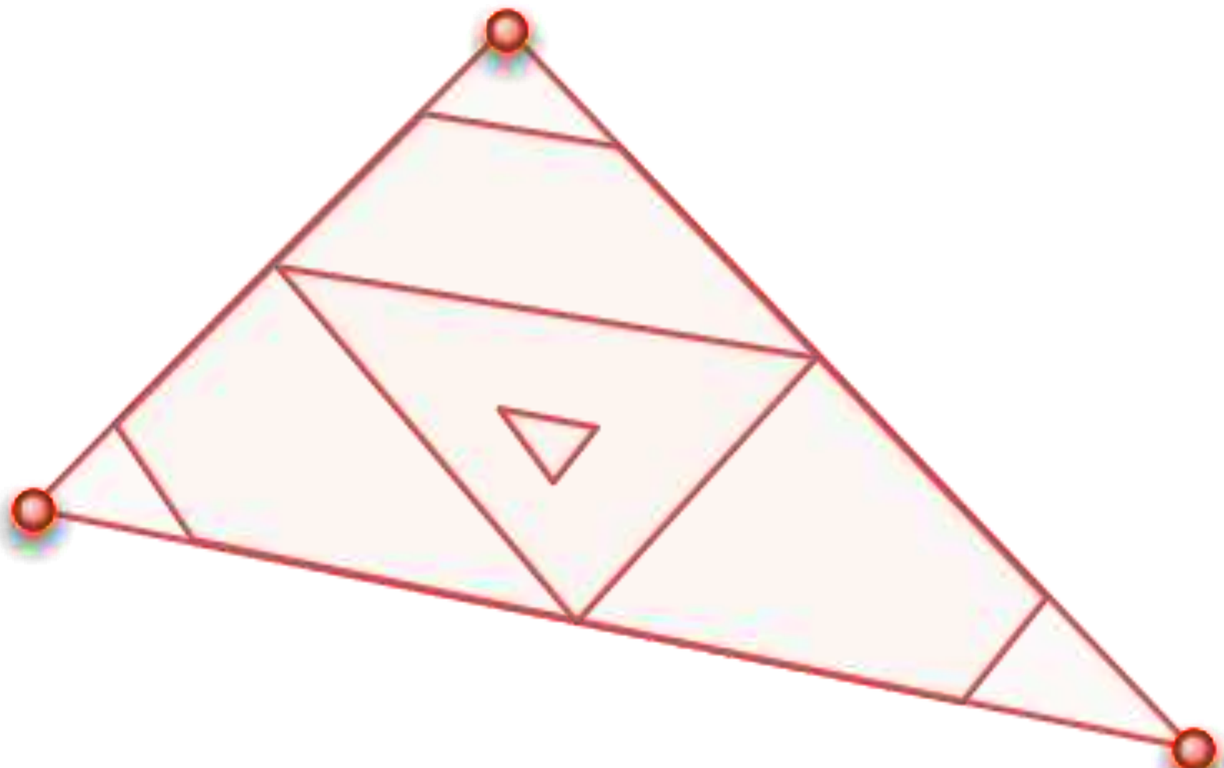
$$\begin{aligned} \min_{\alpha} \quad & \|\textstyle\sum_{I_+} \alpha_i x_i - \sum_{I_-} \alpha_i x_i\|^2 \\ s.t. \quad & \alpha_i \geq 0 \qquad \forall i \\ & \textstyle\sum_{I_+} \alpha_i = 1 \\ & \textstyle\sum_{I_-} \alpha_i = 1 \end{aligned}$$

Soft-margin SVM

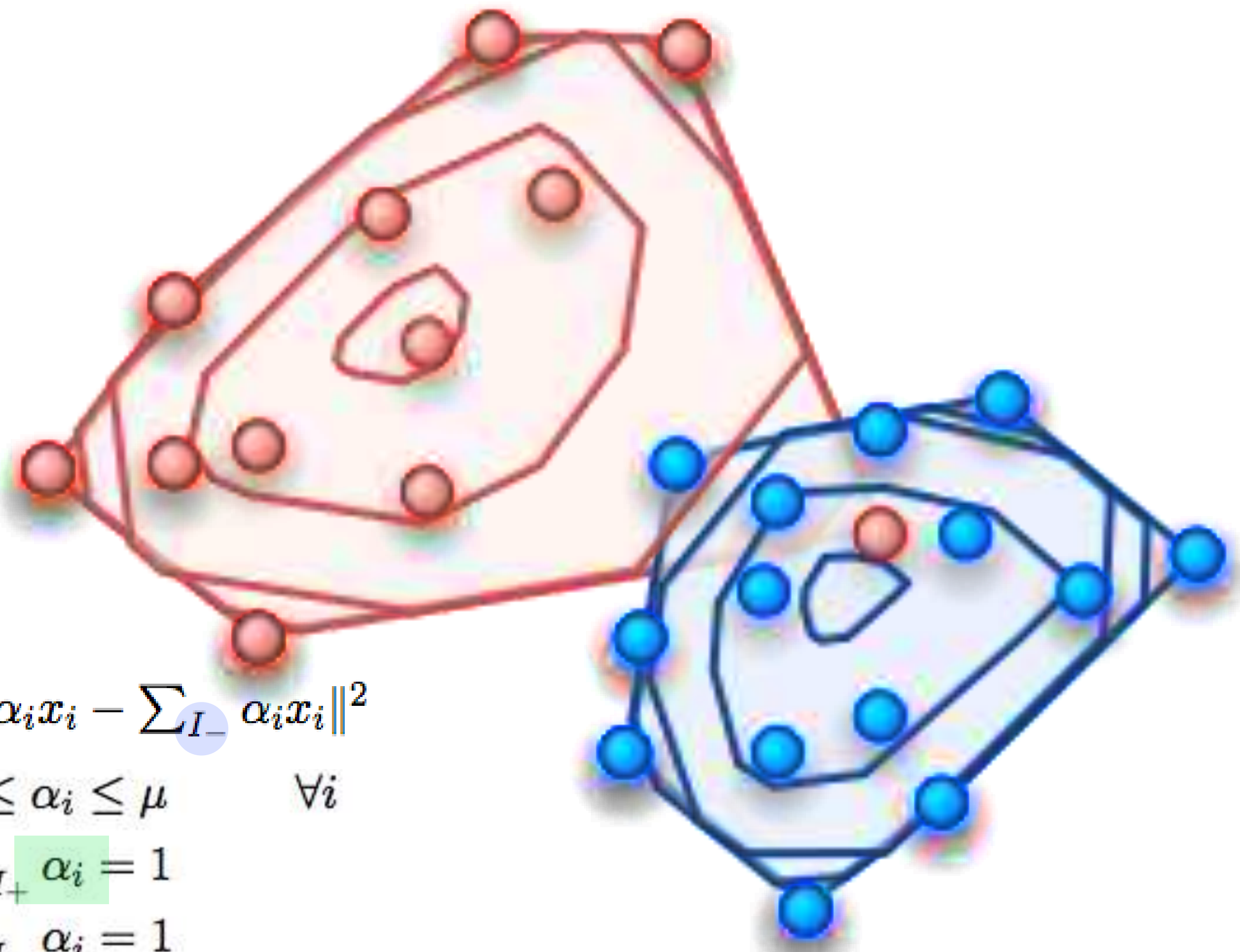
$$\begin{aligned} \min_{\alpha} \quad & \|\textstyle\sum_{I_+} \alpha_i x_i - \sum_{I_-} \alpha_i x_i\|^2 \\ s.t. \quad & 0 \leq \alpha_i \leq \mu \qquad \forall i \\ & \textstyle\sum_{I_+} \alpha_i = 1 \\ & \textstyle\sum_{I_-} \alpha_i = 1 \end{aligned}$$

Reduced Convex Hulls

$$RCH(X,\mu) := \left\{ \sum_i \alpha_i x_i \,\middle|\, 0 \le \alpha_i \le \mu \;\; \forall i, \;\; \sum_i \alpha_i = 1 \right\}$$



Distance between reduced convex hulls



$$\min_{\alpha} \quad \| \textstyle\sum_{I_+} \alpha_i x_i - \sum_{I_-} \alpha_i x_i \|^2$$

$$\begin{aligned} s.t. \quad & 0 \leq \alpha_i \leq \mu \qquad \forall i \\ & \textstyle\sum_{I_+} \alpha_i = 1 \\ & \textstyle\sum_{I_-} \alpha_i = 1 \end{aligned}$$

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

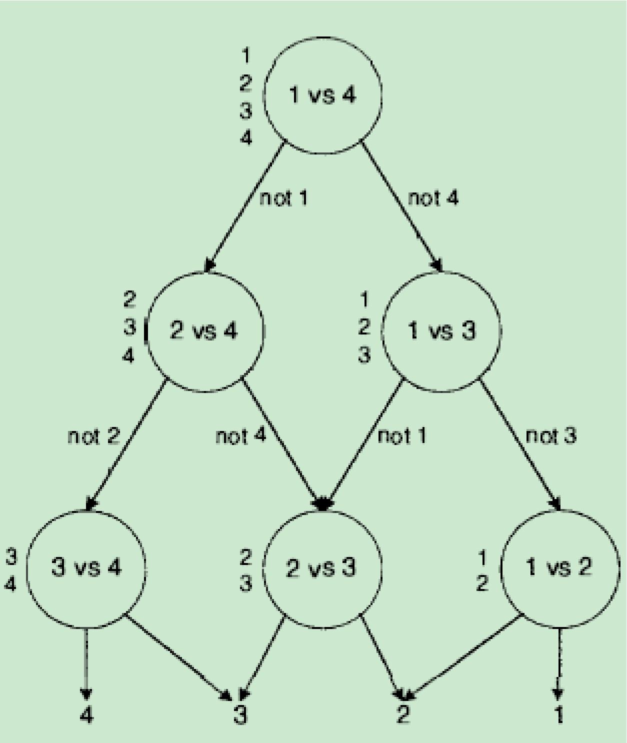
- one VS all (1-v-a)
  - *C* *N*-variable quadratic programming problems are solved

- 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

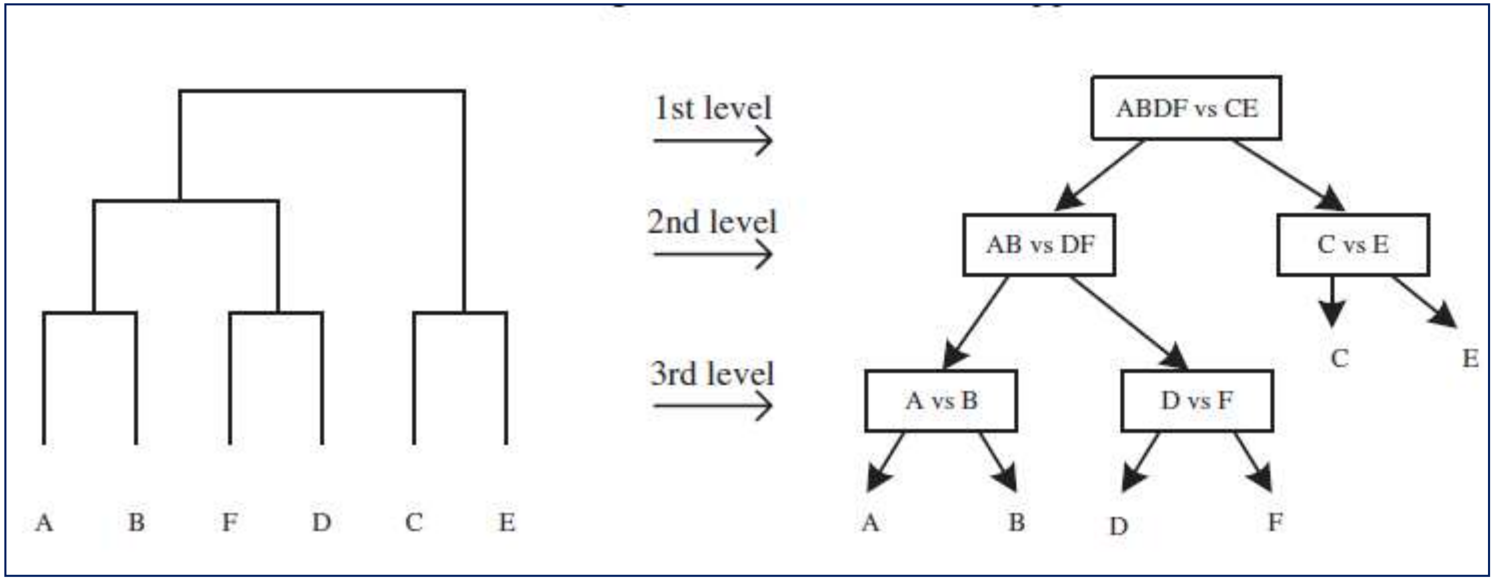
- Algorithm:

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*.



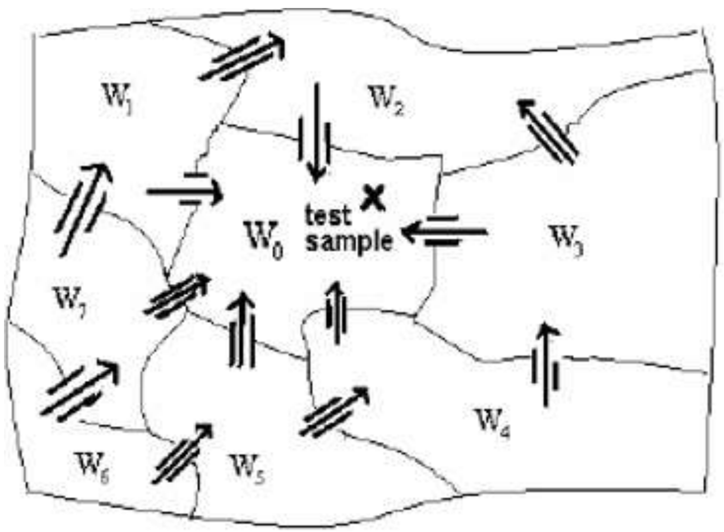
Half-vs-Half

- How to divide the data into two subsets hierarchically?
  - ✓ Two subsets have the largest margin.
  - ✓ Two sbusets 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 neighor 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 neighor classes *j*.
2. While(*SVM_ij(x) = j*)

   move to the neighor class *j*

   search its neighor 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:

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

$$z_i = \phi(x_i),\ k(x_i, x_j) =< z_i, z_j >$$

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

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

$$\min_{\{w_m\}\{\xi_i^m\}} \frac{1}{2} \sum_m \|w_m\|^2 + C \sum_i \sum_{m \neq y_i} \xi_i^m$$

$$s.t. \begin{cases} w_{y_i}^T z_i - w_m^T z_i \geq 1 - \xi_i^m \\ \xi_i^m \geq 0 \end{cases}, \forall i, \forall m \neq y_i$$

Single Machine 1

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

- V.S.

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

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

- Just solve a simple linear program:

$$\min \sum_{i=1}^{N} \alpha_i + C \sum_{i=1}^{N} \sum_{m \neq y_i} \xi_i^m$$

$$s.t. \begin{cases} f_{y_i}(x_i) - f_m(x_i) \geq 1 - \xi_i^m \\ \alpha_i \geq 0, \xi_i^m \geq 0 \end{cases} , \forall i, \forall m \neq y_i$$

Single Machine 2

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

$$\min_{\{w_m\}\{\xi_i\}} \frac{1}{2}\sum_m \|w_m\|^2 + C\sum_i \boxed{\xi_i}$$

$$s.t. \begin{cases} w_{y_i}^T z_i - w_m^T z_i \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} , \forall i, \forall m \neq y_i$$

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

- 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

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

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

Locally Linear SVM

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

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

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

✓ Bilinear SVM

✓ Transform the feature from *n* to *nm*

✓ Weighted sum of linear SVMs for each anchor point

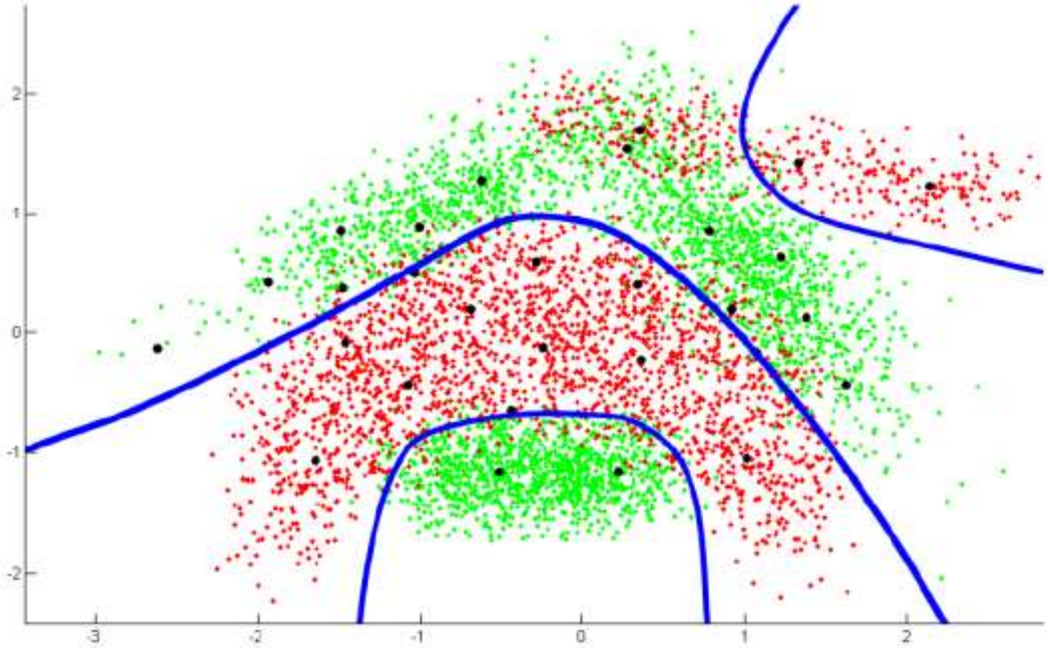
✓ More general than standard linear SVM

✓ More general than linear SVM over local coding

Locally Linear SVM

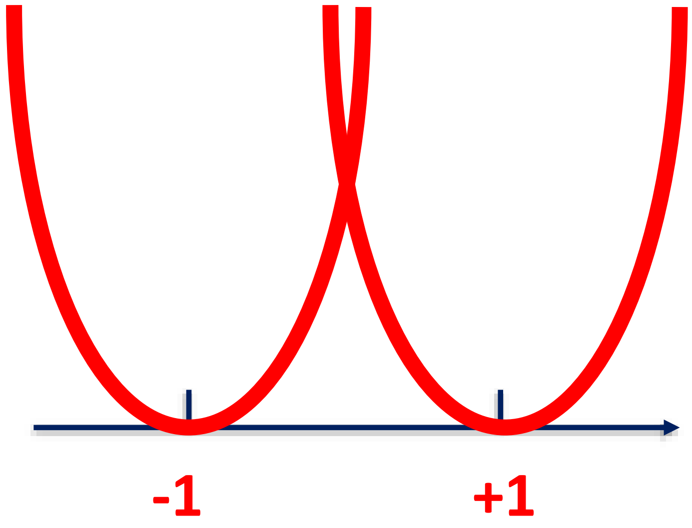
K-means clustering → anchor points

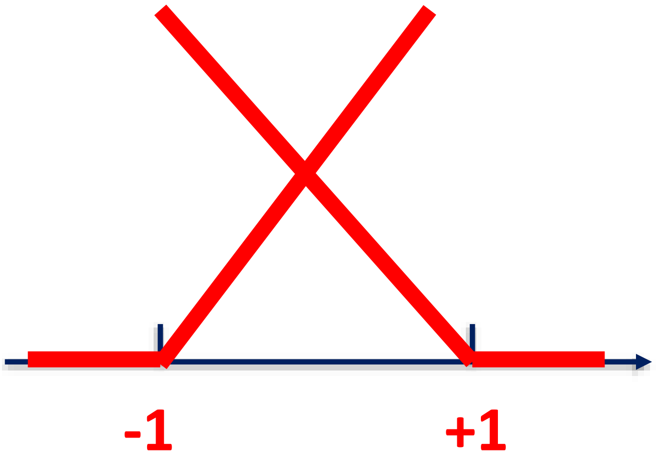
Coefficients of the local coding → inverse Euclidian distance based weighting from nearest neighbors.



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:

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

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

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

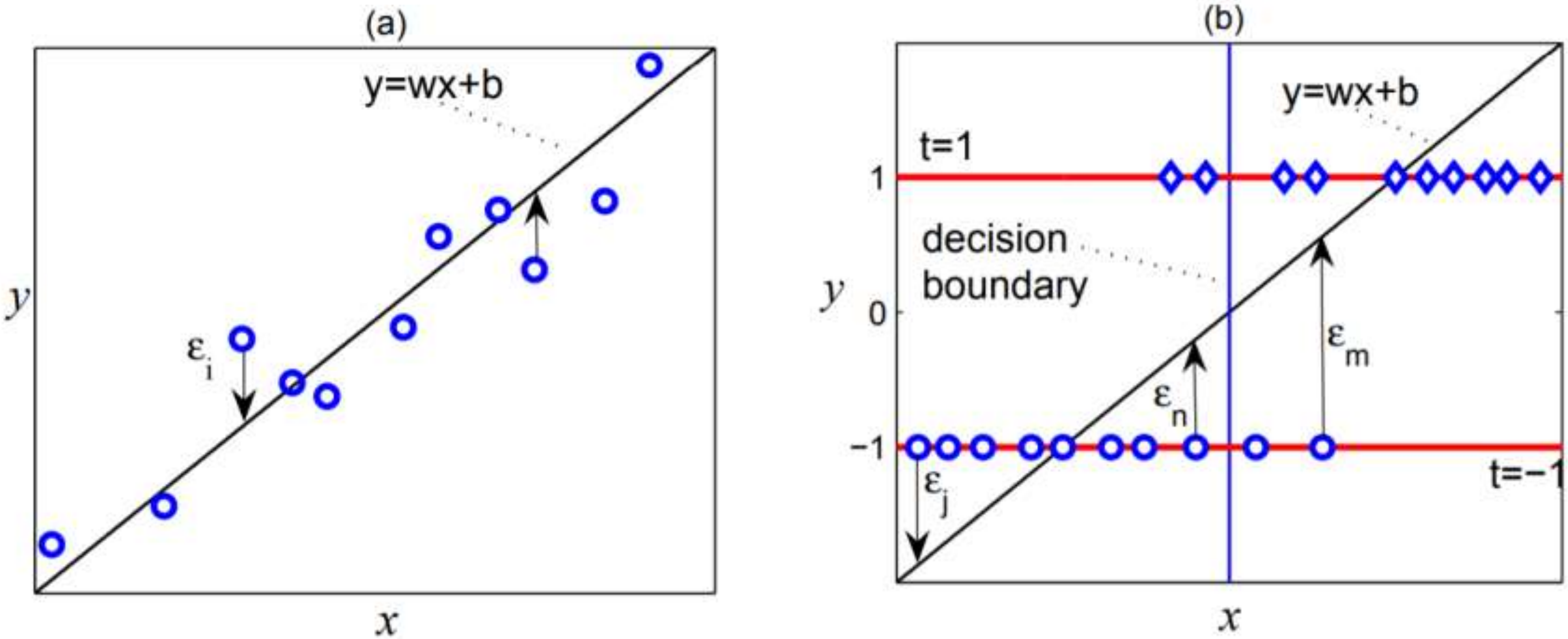
Different Loss Functions

- Data $\{x_i\}_{i=1}^N$ $\{t_i\}_{i=1}^N$ $t_i \in \{1, -1\}$
- Classifier $\widehat{t} = \text{sign}(w^\top \phi(x) + b).$
- Margin variable $z = ty$
- **Large z → small loss**

- Loss Functions:
  - 0-1 loss $l_{mer}(z_i) = ||(\text{-}z_i)_+||_0$
  - Hinge loss $l_{hinge} = \{(1 - z_i)_+\}^p$ $(p\text{=}1 \text{ or } 2)$
  - Logistic regression $l_{log}\text{=}\log_2[1\text{+}\exp(\text{-} z_i)]$
  - Least squares $l_{ls}\text{=}(1\text{-}z_i)^2$
  - Adaboost $l_{exp} = \exp(\text{-}z_i).$

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

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

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

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

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

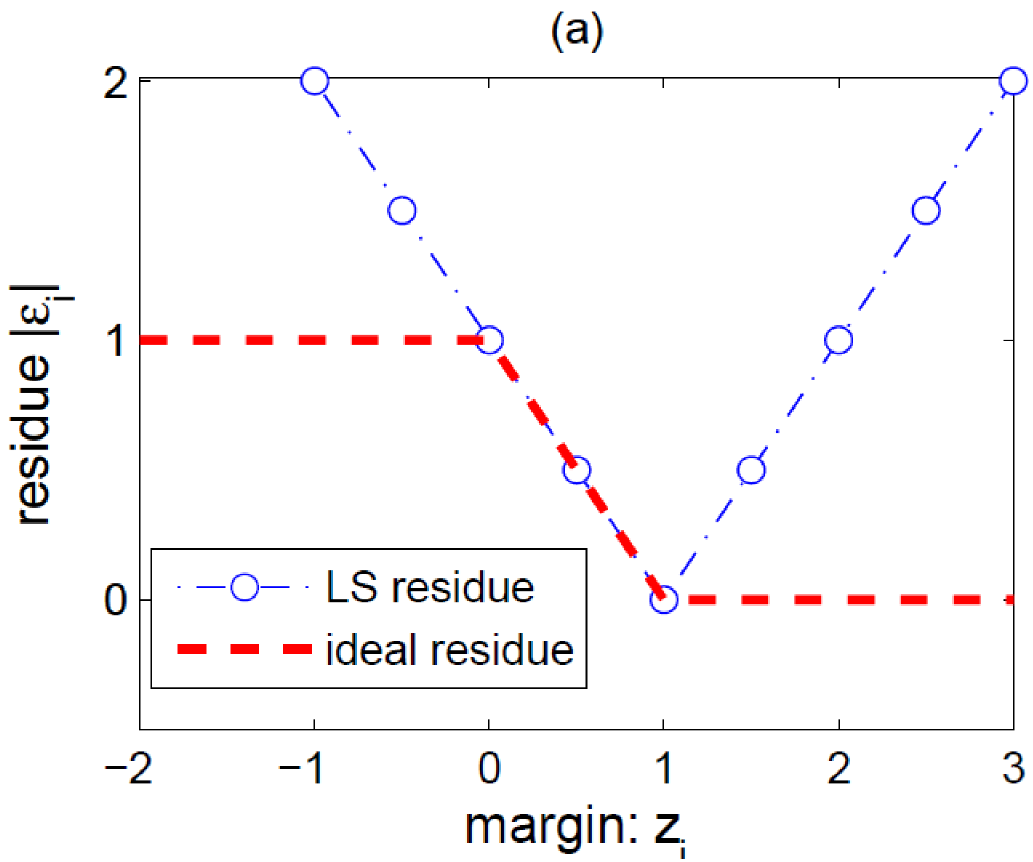
S.-H. Yang and B.-G. Hu, A stagewise least square loss function for classification, in Proc. SIAM Int. Conf. Data Mining (SDM), 2008.

Stagewise Least Square

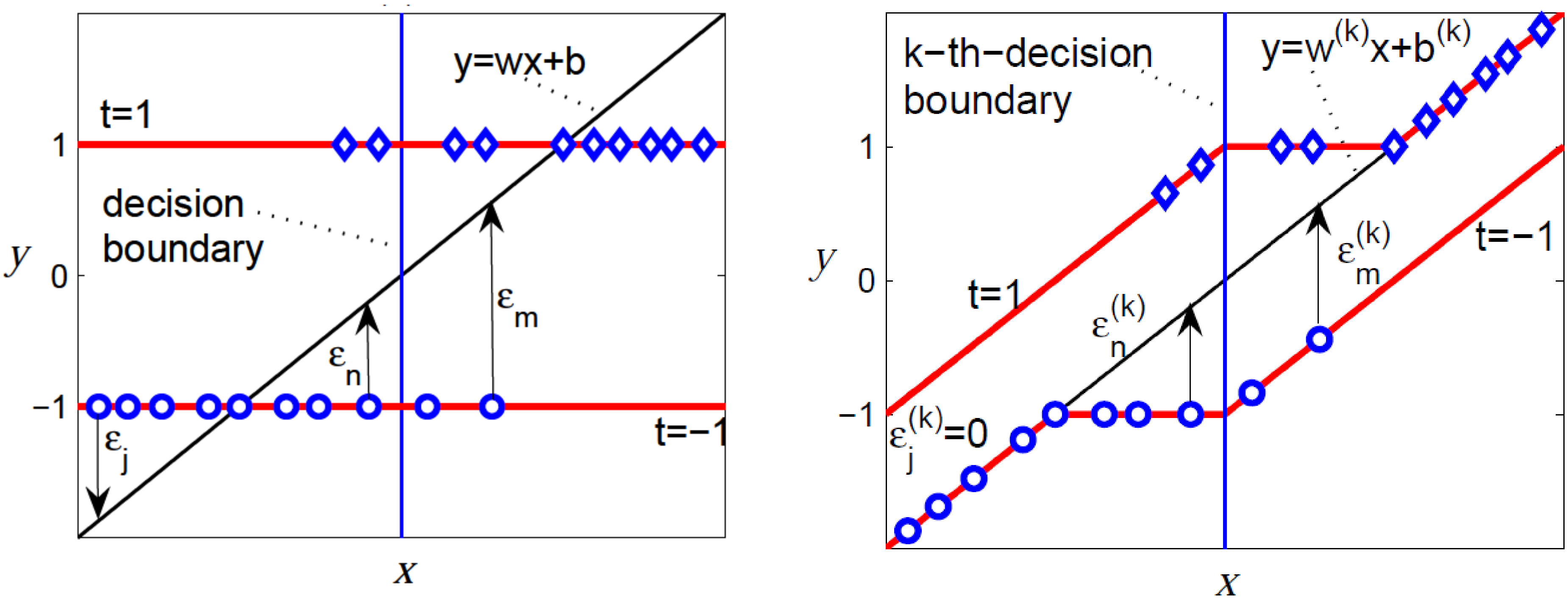
$$\boxed{\begin{aligned}\tau_i^{(0)} &= 1,\\ \tau_i^{(k+1)} &= z_i^{(k)} + S(1 - z_i^{(k)}),\end{aligned}}$$

- 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



Stagewise Least Square

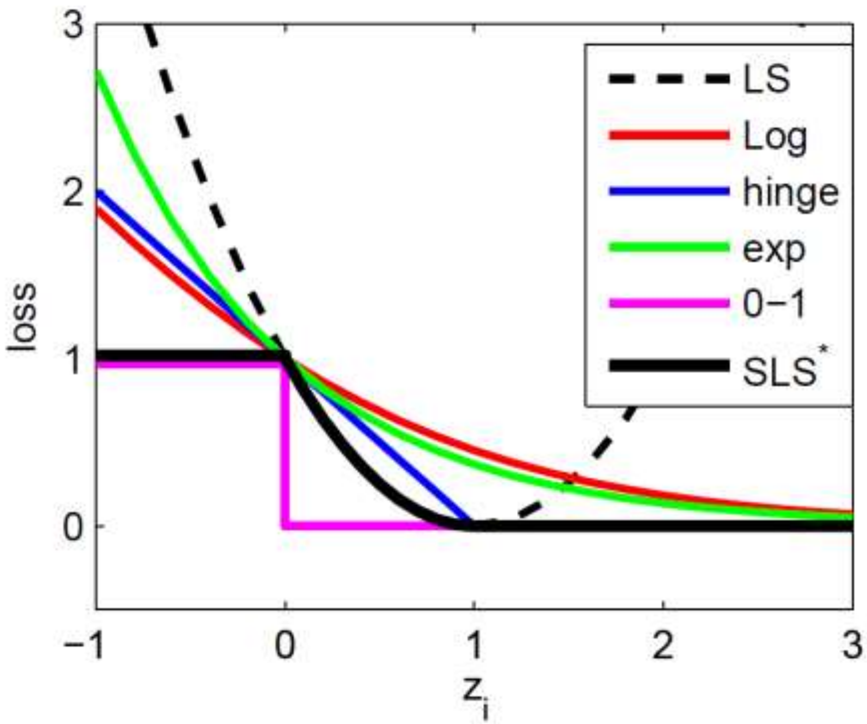


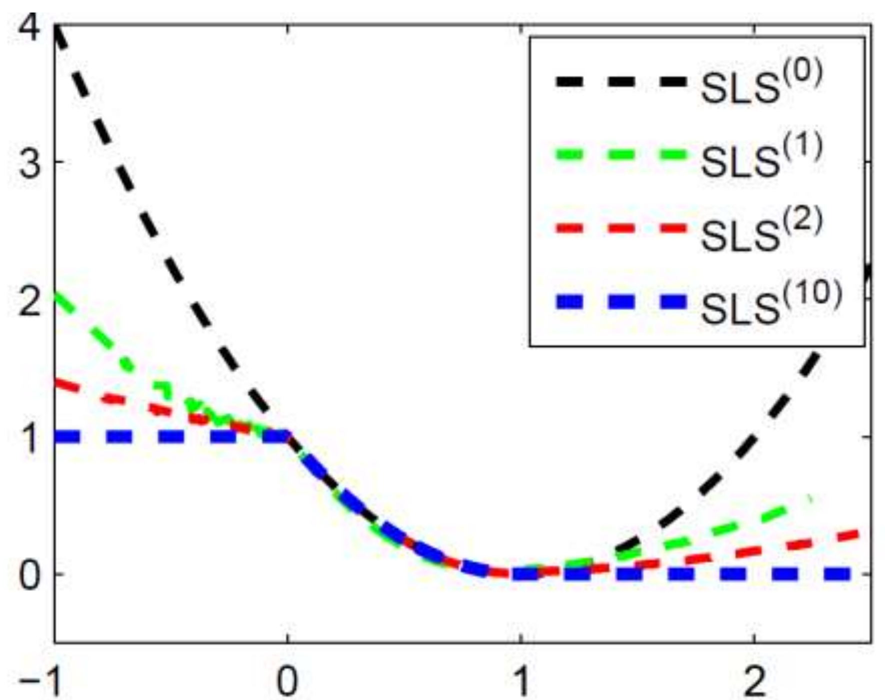
Stagewise Least Square

- 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}
$\quad \boxed{\bar{w}^{(k)} = G \times \Gamma^{(k)}}$
end for
$\bar{w} = \bar{w}^{(k_{max})}$

Kernelization

$$\begin{aligned} \min \quad & \tfrac{1}{2}\bar{w}^T\bar{w} + C\tfrac{1}{2}\textstyle\sum_{i=1}^{N}\varepsilon_i^2 \\ s.t. \quad & y_i - t_i\tau_i = \varepsilon_i \end{aligned}$$

$$w = \sum_{i=1}^{N} \alpha_i \phi(x_i)$$

$$b = \sum_{i=1}^{N} \alpha_i$$

$$\alpha_i = C\varepsilon_i$$

$$\varepsilon_i = y_i - t_i\tau_i^{(k)}.$$

$$\begin{pmatrix} -1 & 1_N^T \\ 1_N & K + C^{-1}I \end{pmatrix} \begin{pmatrix} b \\ \alpha \end{pmatrix} = \begin{pmatrix} 0 \\ \Gamma \end{pmatrix}$$

$$y(x) = \sum_{i=1}^{N} \alpha_i k(x, x_i) + b.$$

Discriminative LSR

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

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

- Let

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

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

- Training sample fitting

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

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 Y → 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 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

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

- OPT1:
  – Fix M and solve W, t → unconstrained convex QP→解析解
- OPT2:
  – Fix W, t and solve M → very simple elementwise solution

DLSR: OPT1: Fix M

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

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

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

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

DLSR: OPT1: Fix M

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

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

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

$$\frac{\partial g(\mathbf{W}, \mathbf{t})}{\partial \mathbf{W}} = \mathbf{0}$$

$$\Rightarrow \mathbf{X}^T\left(\mathbf{X}\mathbf{W} + \frac{1}{n}\mathbf{e}_n\mathbf{e}_n^T\mathbf{R} - \frac{1}{n}\mathbf{e}_n\mathbf{e}_n^T\mathbf{X}\mathbf{W} - \mathbf{R}\right) + \lambda\mathbf{W} = \mathbf{0}$$

$$\Rightarrow \mathbf{X}^T\left(\mathbf{I}_n - \frac{1}{n}\mathbf{e}_n\mathbf{e}_n^T\right)\mathbf{X}\mathbf{W} - \mathbf{X}^T\left(\mathbf{I}_n - \frac{1}{n}\mathbf{e}_n\mathbf{e}_n^T\right)\mathbf{R} + \lambda\mathbf{W} = \mathbf{0}$$

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

DLSR: OPT2: Fix W t

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

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

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

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

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

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

Algorithm of DLSR

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

Algorithm 1 *DLSR*

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

```
Allocate M, W, W_0, t, and t_0.
M = 0, W_0 = 0, and t_0 = 0.
Construct X and Y in (4), and B according to (5).
Let U = (X^T H X + λ I_m)^{-1} X^T H.
Let k = 1.
while k < T do
  R = Y + B ⊙ M.
  W = UR,    t = (1/n) R^T e_n − (1/n) W^T X^T e_n.
  P = XW + e_n t^T − Y.
  M = max(B ⊙ P, 0).
  if (||W − W_0||_F^2 + ||t − t_0||_2^2) < 10^{-4}, then
    Stop.
  end if
  W_0 = W,   t_0 = t,   k = k + 1.
end while
Output W and t.
```

Re-think DLSR

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

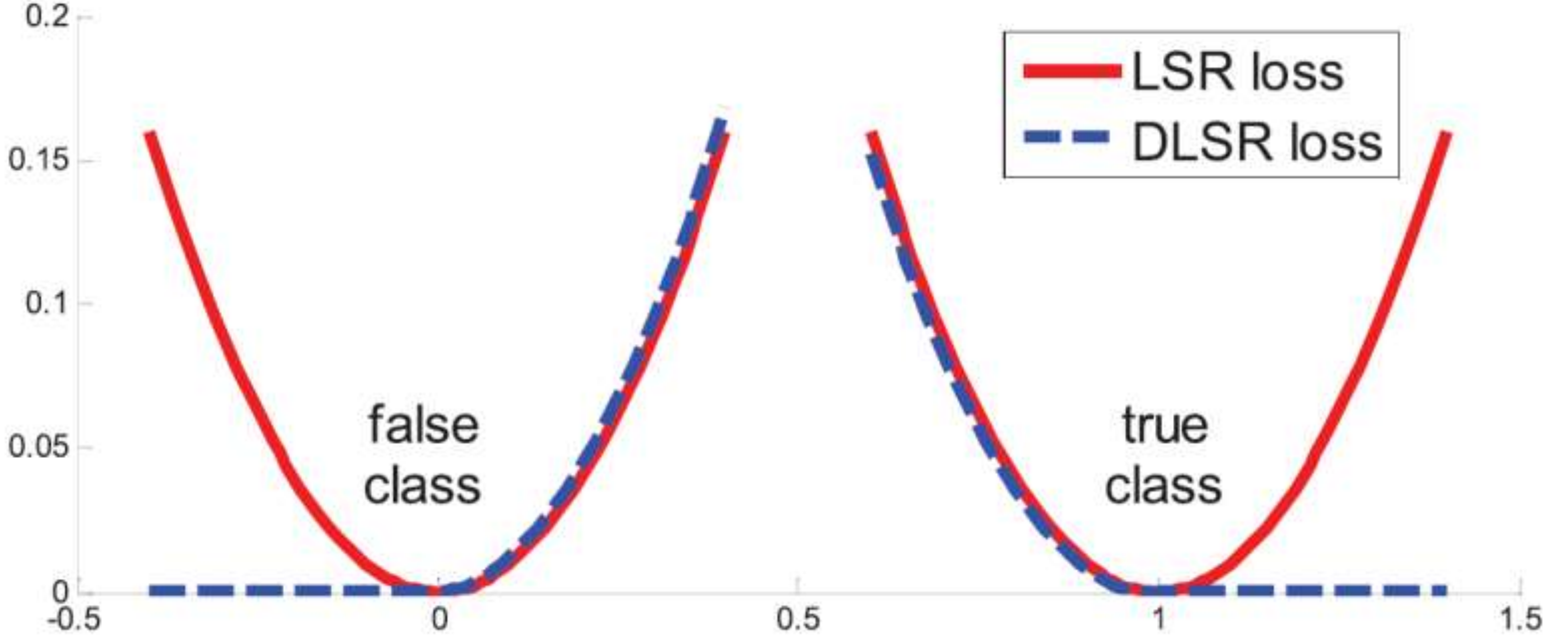
$$\mathbf{M} \geq \mathbf{0}$$

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

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

- 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

$\mathbf{y}$	regression result	margin≥ 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 $\mathbf{T} \in \mathbb{R}^{n \times c}$

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

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

ReLSR: OPT1: Regression

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

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

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

ReLSR: OPT2: Retargeting

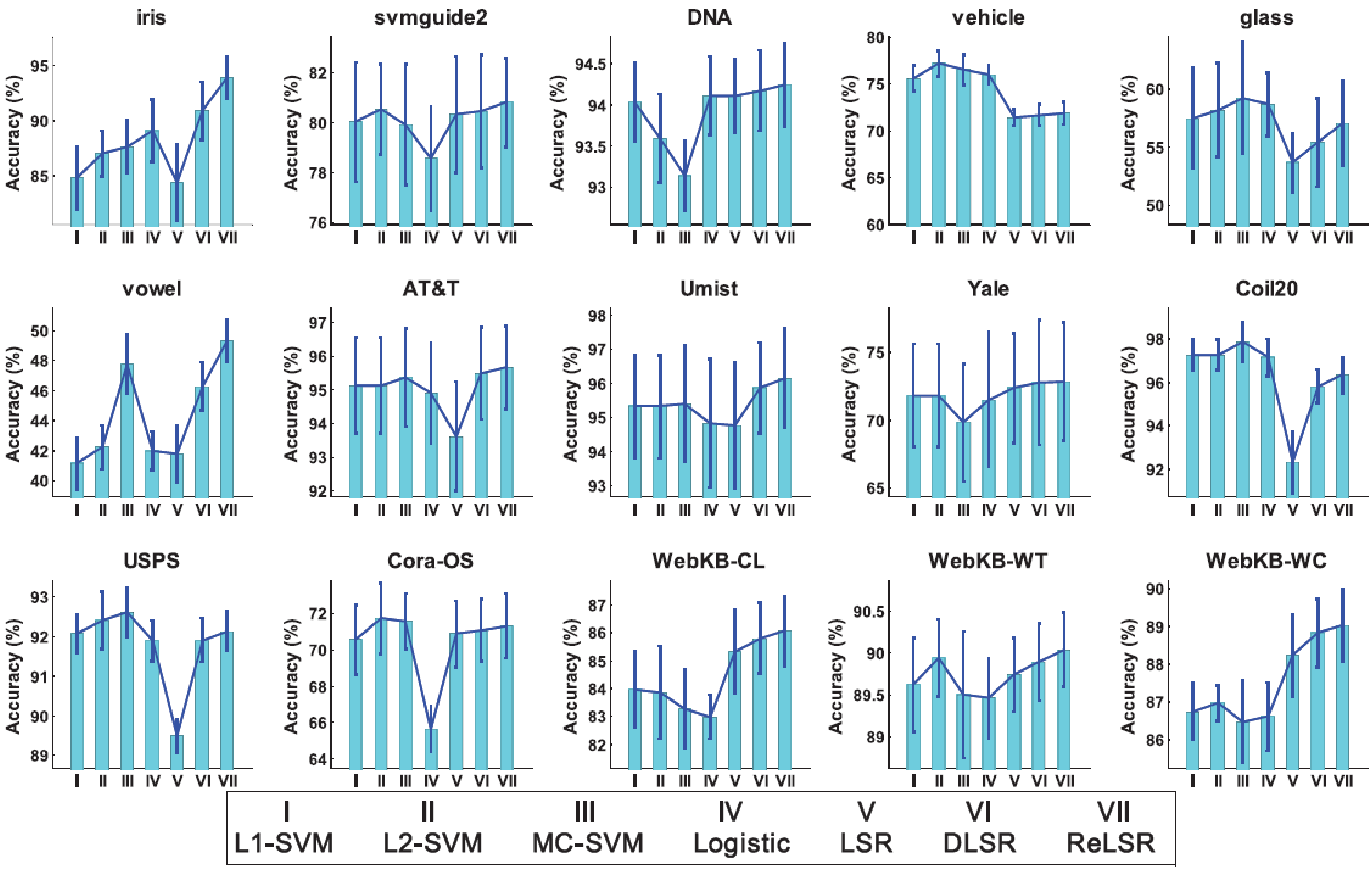
$$\boxed{\begin{aligned} &\min_{\mathbf{W},\mathbf{b},\mathbf{T}} \|\mathbf{X}\mathbf{W} + \mathbf{e}_n\mathbf{b}^\top - \mathbf{T}\|_F^2 + \beta\|\mathbf{W}\|_F^2 \\ &\text{s.t.} \quad T_{i,y_i} - \max_{j \neq y_i} T_{i,j} \geq 1, \quad i = 1, 2, \ldots, n. \end{aligned}}$$

$$\text{Retargeting:} \quad \min_{\mathbf{T}} \|\mathbf{X}\mathbf{W} + \mathbf{e}_n\mathbf{b}^\top - \mathbf{T}\|_F^2 = \|\mathbf{R} - \mathbf{T}\|_F^2$$

$$\text{s.t.} \quad T_{i,y_i} - \max_{j \neq y_i} T_{i,j} \geq 1, \quad i = 1, 2, \ldots, n.$$

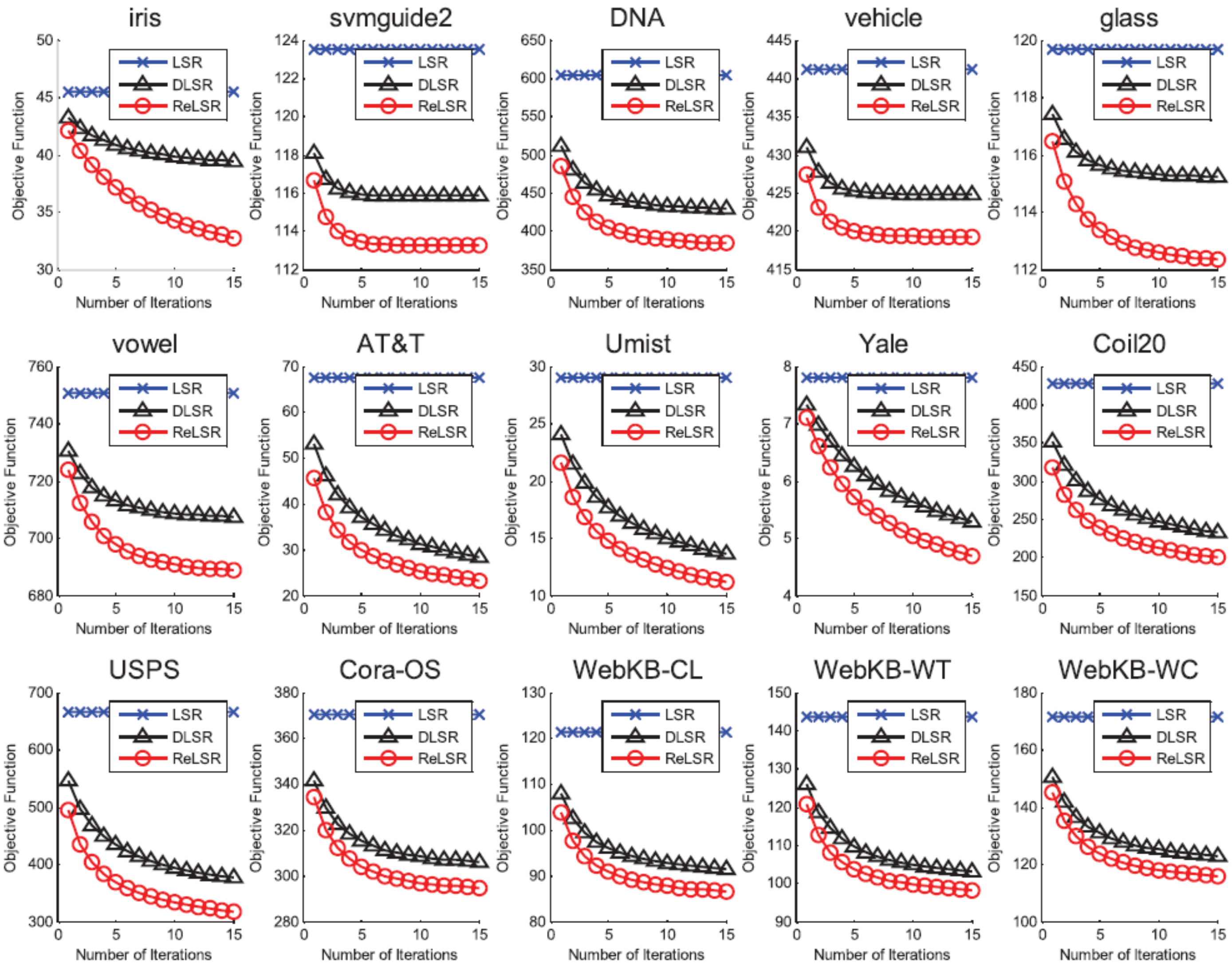
$$\min_{\mathbf{t}} \|\mathbf{r} - \mathbf{t}\|_2^2 = \sum_{i=1}^{c} (r_i - t_i)^2 \quad \text{s.t.} \quad t_k - \max_{i \neq k} t_i \geq 1.$$

Comparison



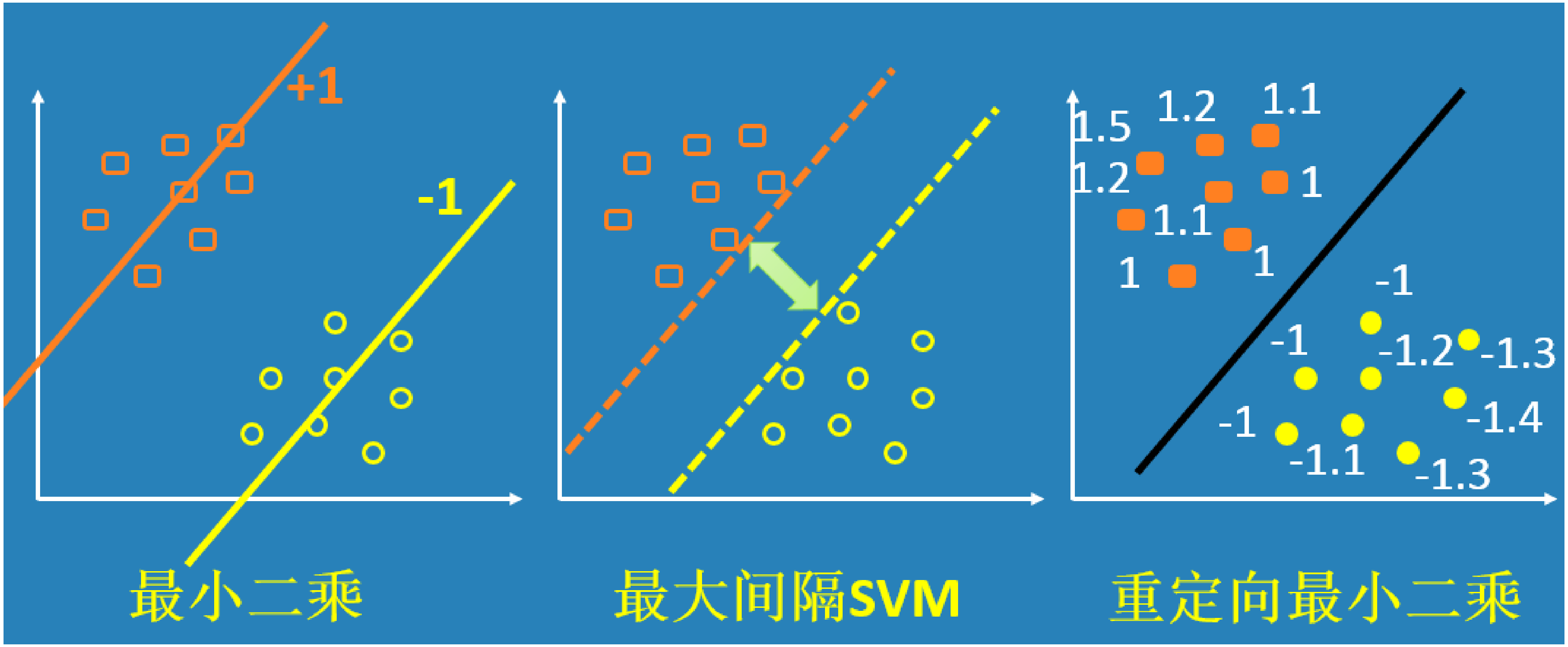
iris
svmguide2
DNA
vehicle
glass
vowel
AT&T
Umist
Yale
Coil20
USPS
Cora-OS
WebKB-CL
WebKB-WT
WebKB-WC
Accuracy (%)
I II III IV V VI VII
I L1-SVM
II L2-SVM
III MC-SVM
IV Logistic
V LSR
VI DLSR
VII ReLSR

Comparison



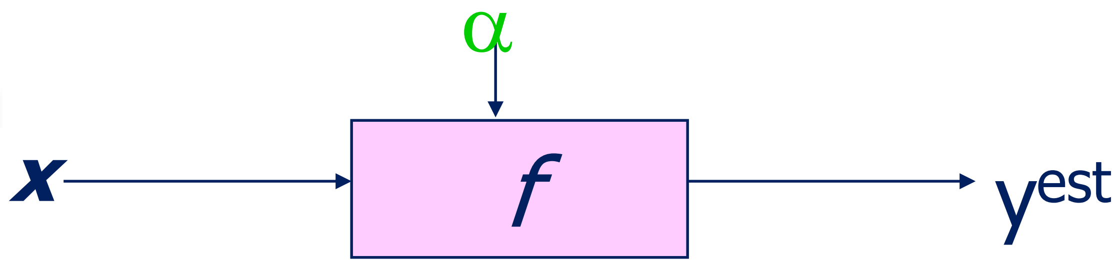
iris
svmguide2
DNA
vehicle
glass
vowel
AT&T
Umist
Yale
Coil20
USPS
Cora-OS
WebKB-CL
WebKB-WT
WebKB-WC
LSR
DLSR
ReLSR
Objective Function
Number of Iterations

From Regression to Classification

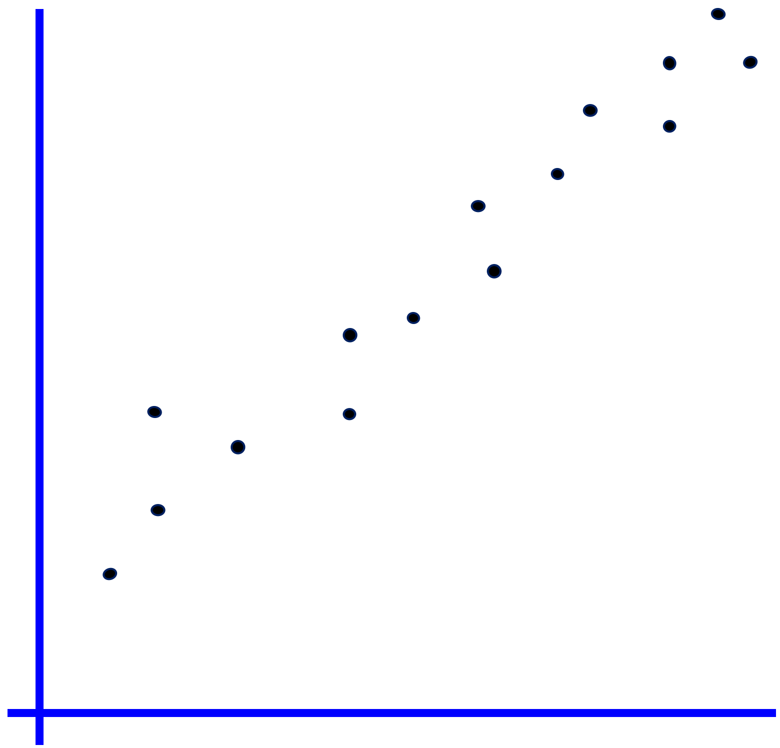


Support Vector Regression

Linear Regression

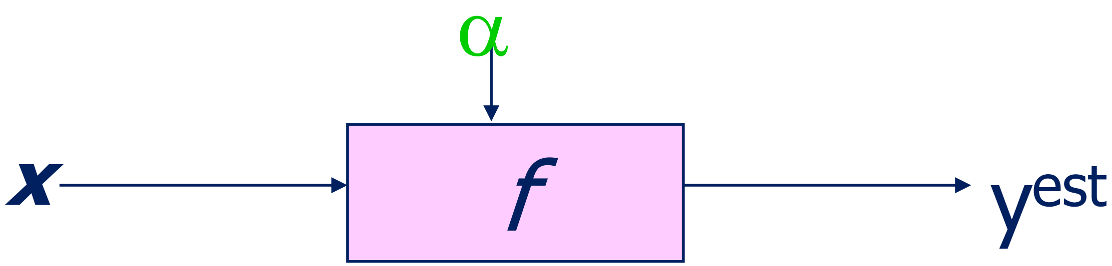


$$f(\boldsymbol{x}, \boldsymbol{w}, b) = \boldsymbol{w}.\, \boldsymbol{x} + b$$

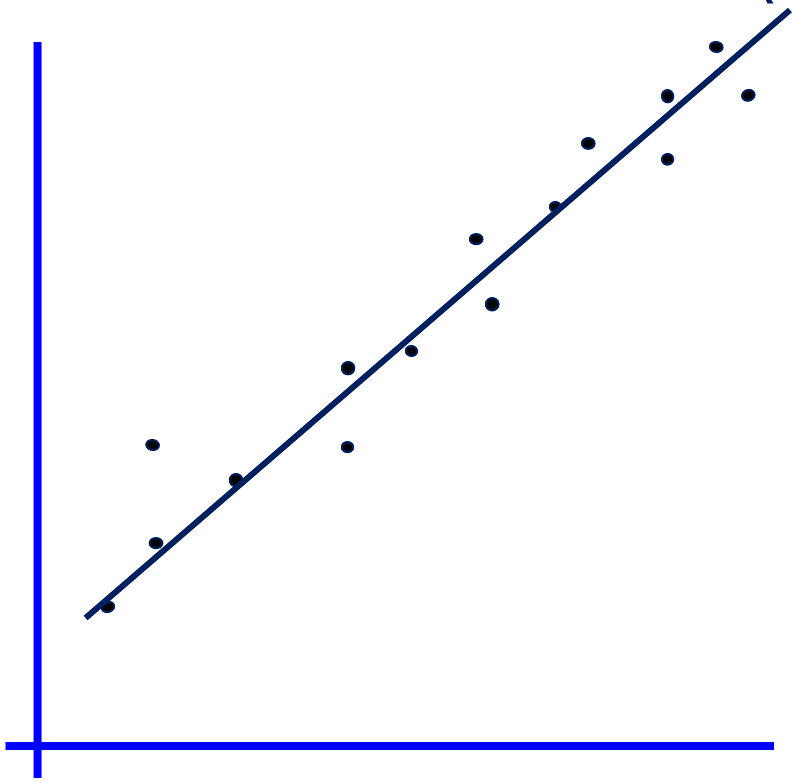


How would you fit this data?

Linear Regression

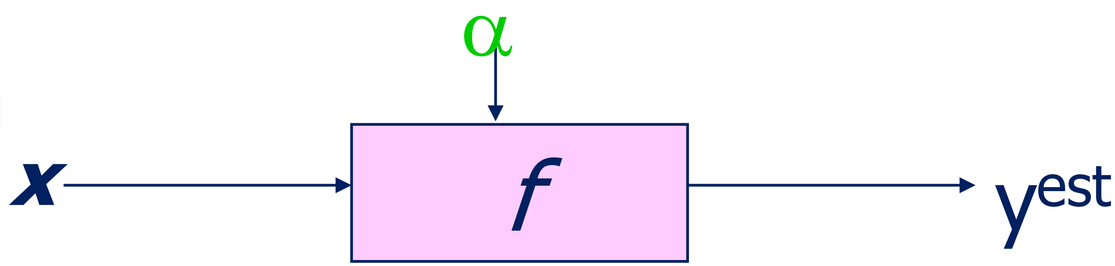


$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{w},b) = \boldsymbol{w}.\ \boldsymbol{x} + b$

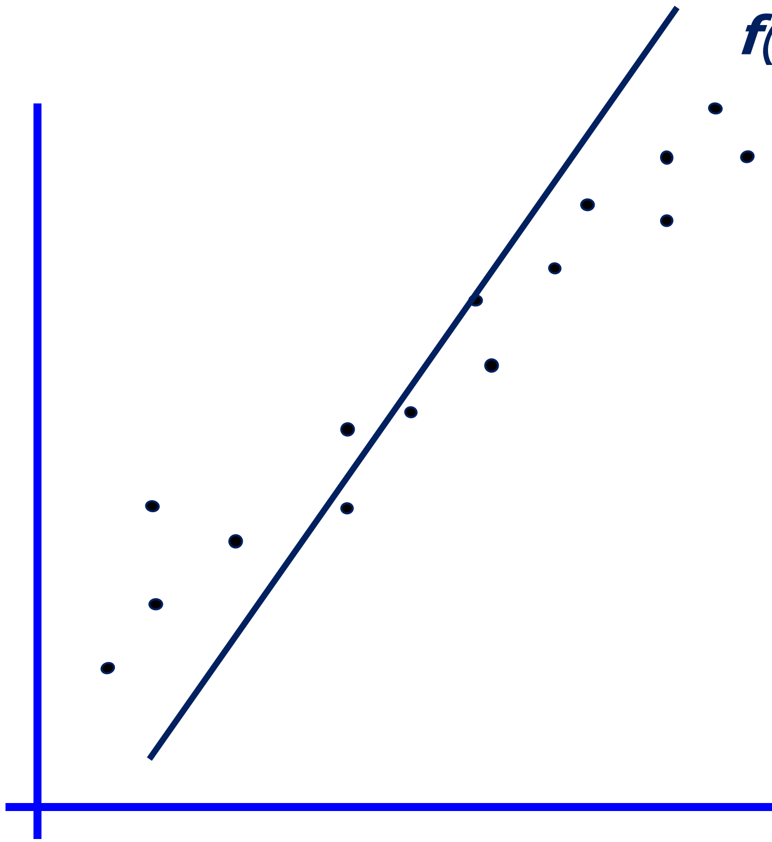


How would you fit this data?

Linear Regression

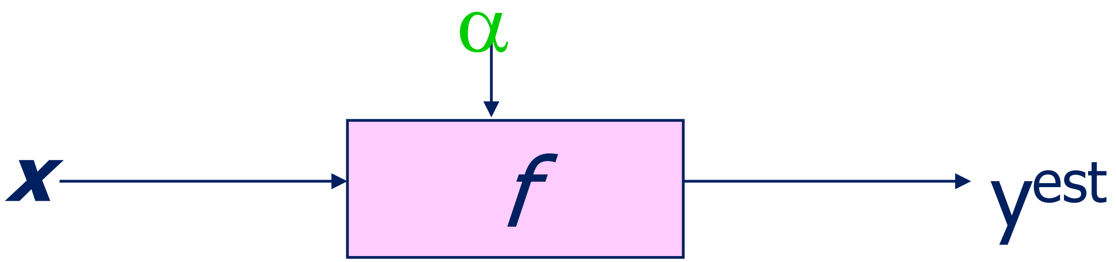


$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{w},b) = \boldsymbol{w}.\ \boldsymbol{x} + b$

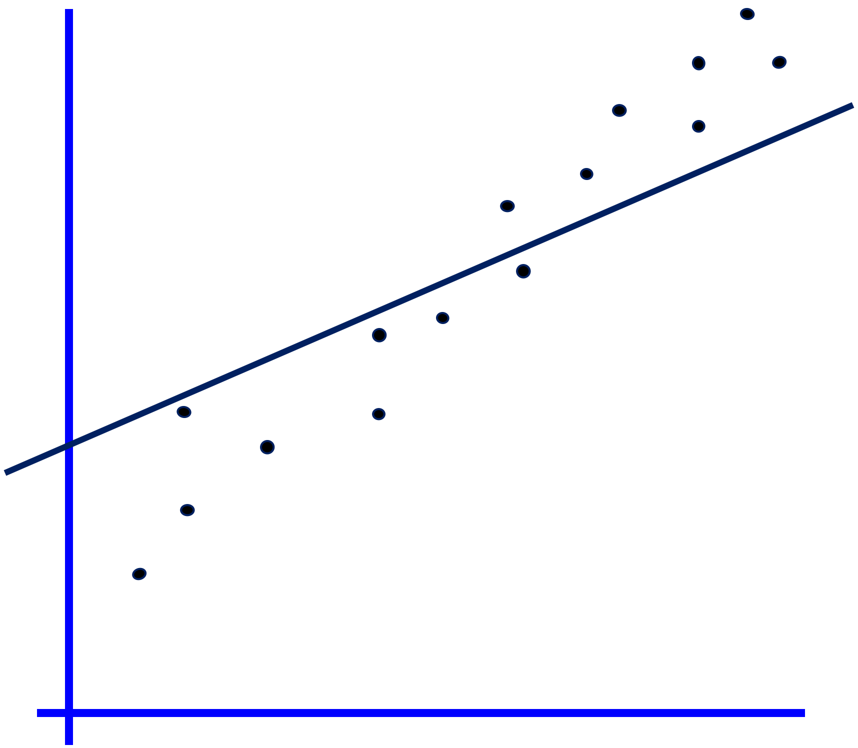


How would you fit this data?

Linear Regression

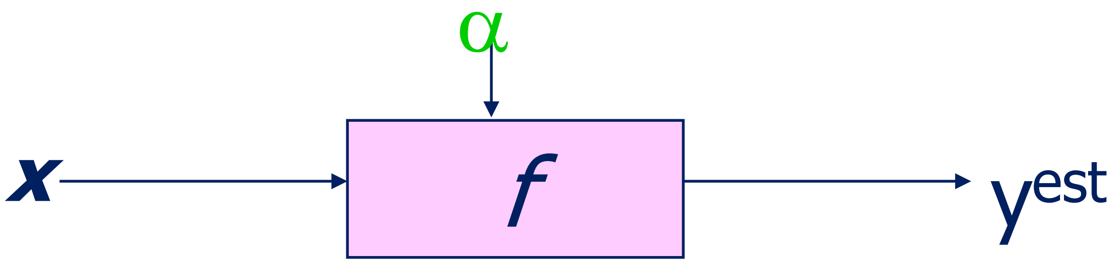


$$f(\mathbf{x},\mathbf{w},b) = \mathbf{w}.\ \mathbf{x} + b$$

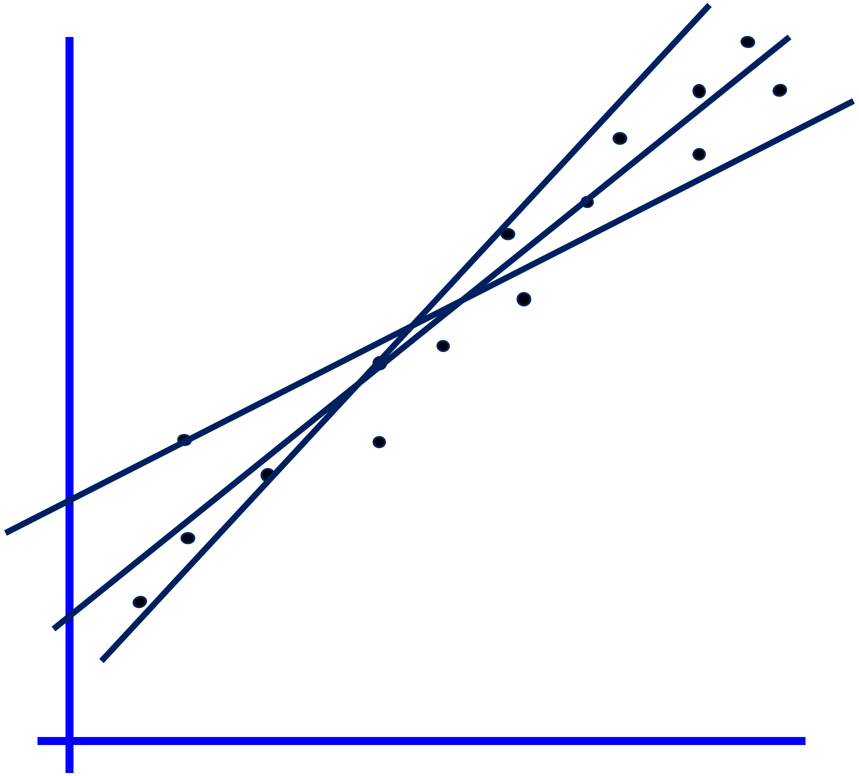


How would you fit this data?

Linear Regression



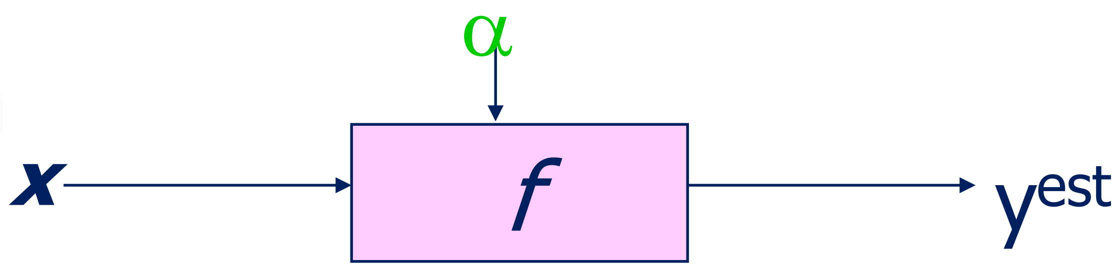
$$f(\mathbf{x},\mathbf{w},b) = \mathbf{w}.\ \mathbf{x} + b$$



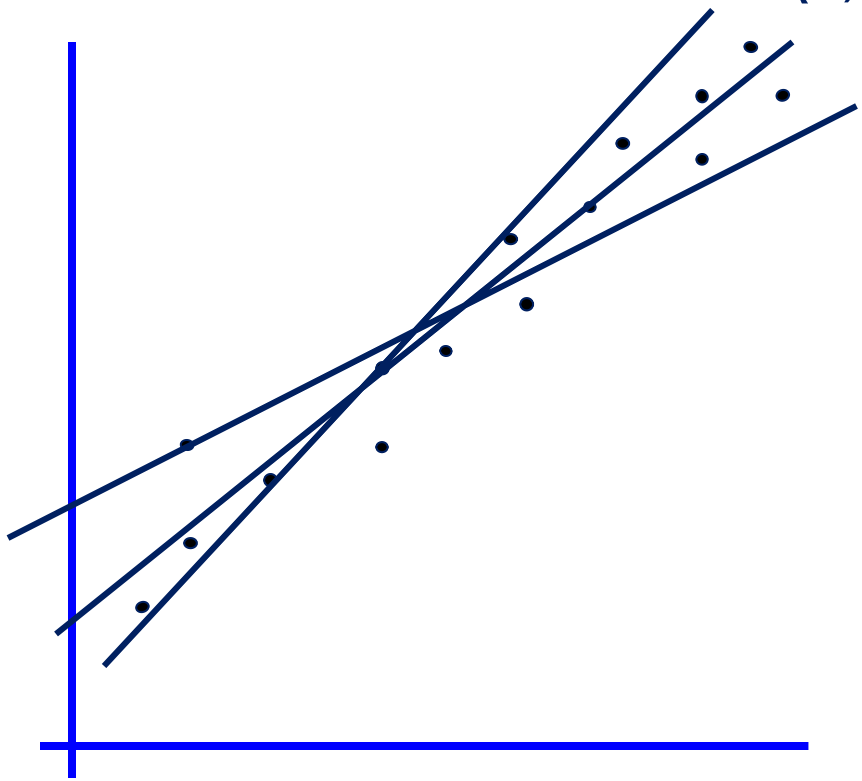
Any of these would be fine..

..but which is best?

Linear Regression

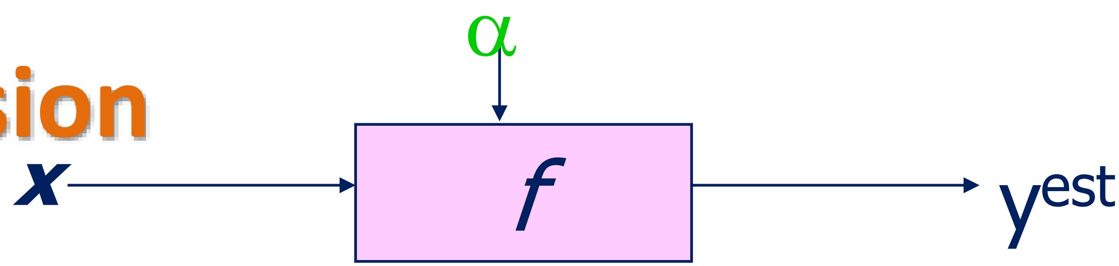


$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{w},b) = \boldsymbol{w}.\,\boldsymbol{x} + b$$

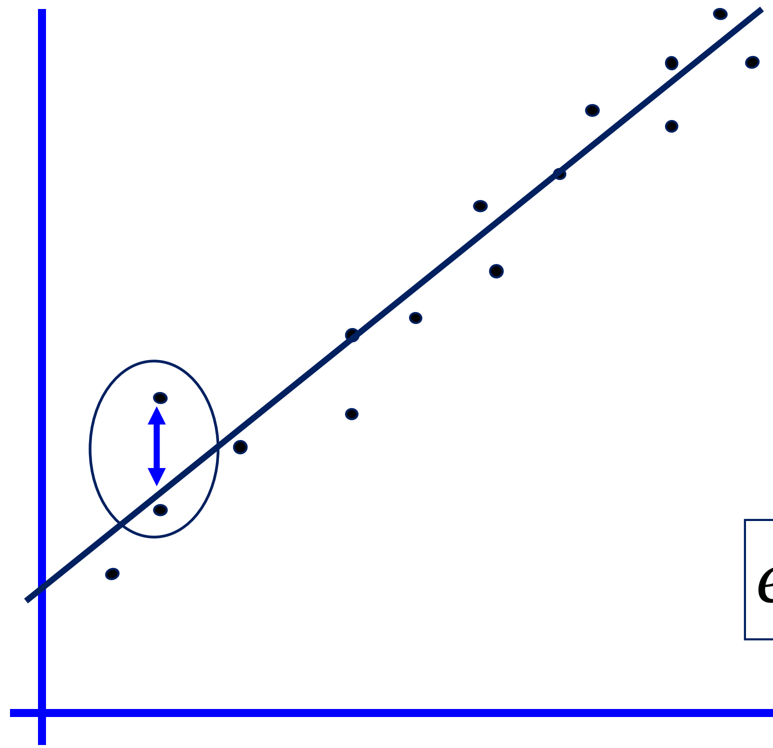


How to define the fitting error of a linear regression ?

Linear Regression



$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{w},b) = \boldsymbol{w}.\ \boldsymbol{x} + b$

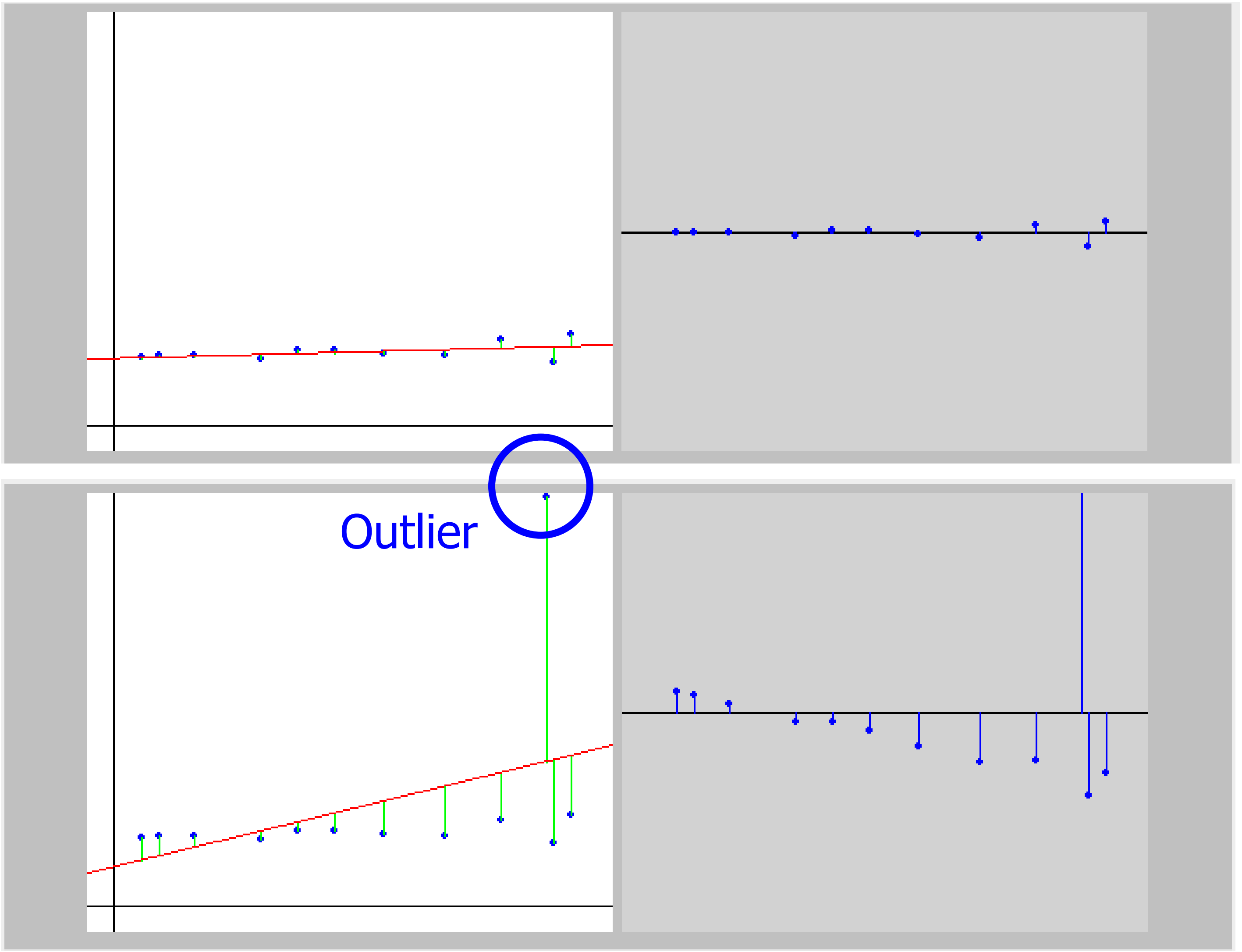


How to define the fitting error of a linear regression ?

$$\boxed{err_i = (w.x_i - b - y_i)^2}$$

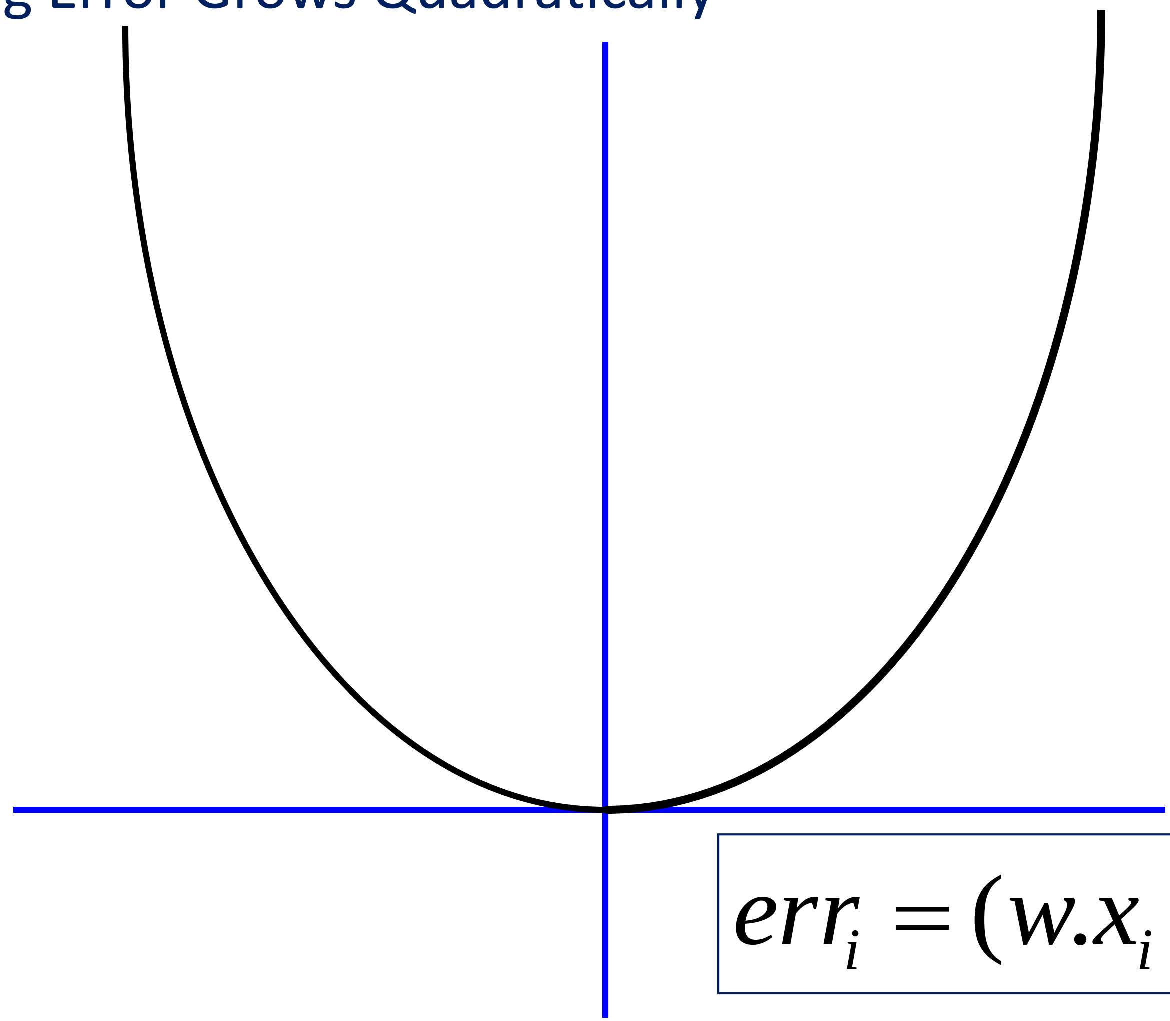
Squared-Loss

Sensitive to Outliers



Why?

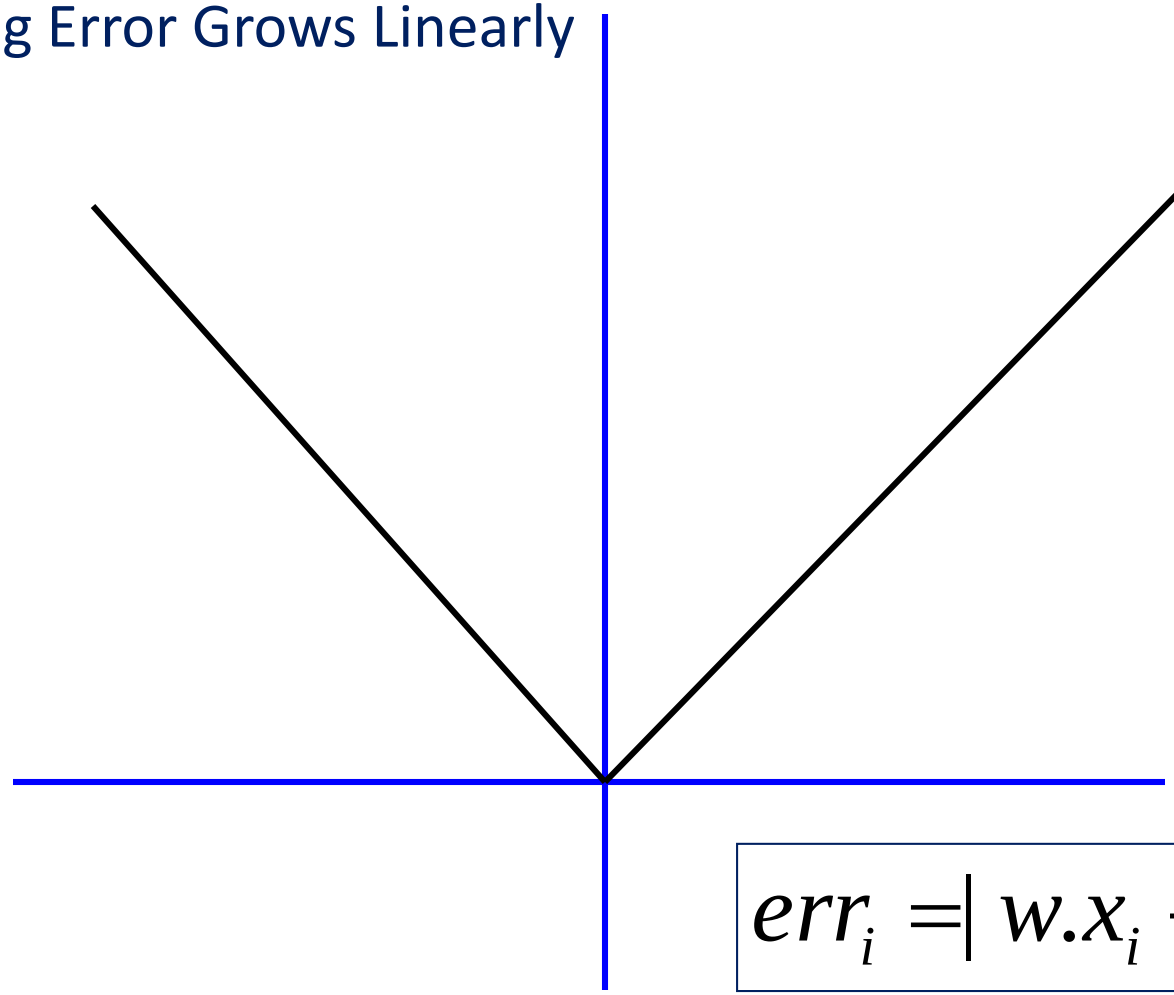
- Squared-Loss Function
  - Fitting Error Grows Quadratically



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

How about Linear-Loss ?

- Linear-Loss Function
  - Fitting Error Grows Linearly

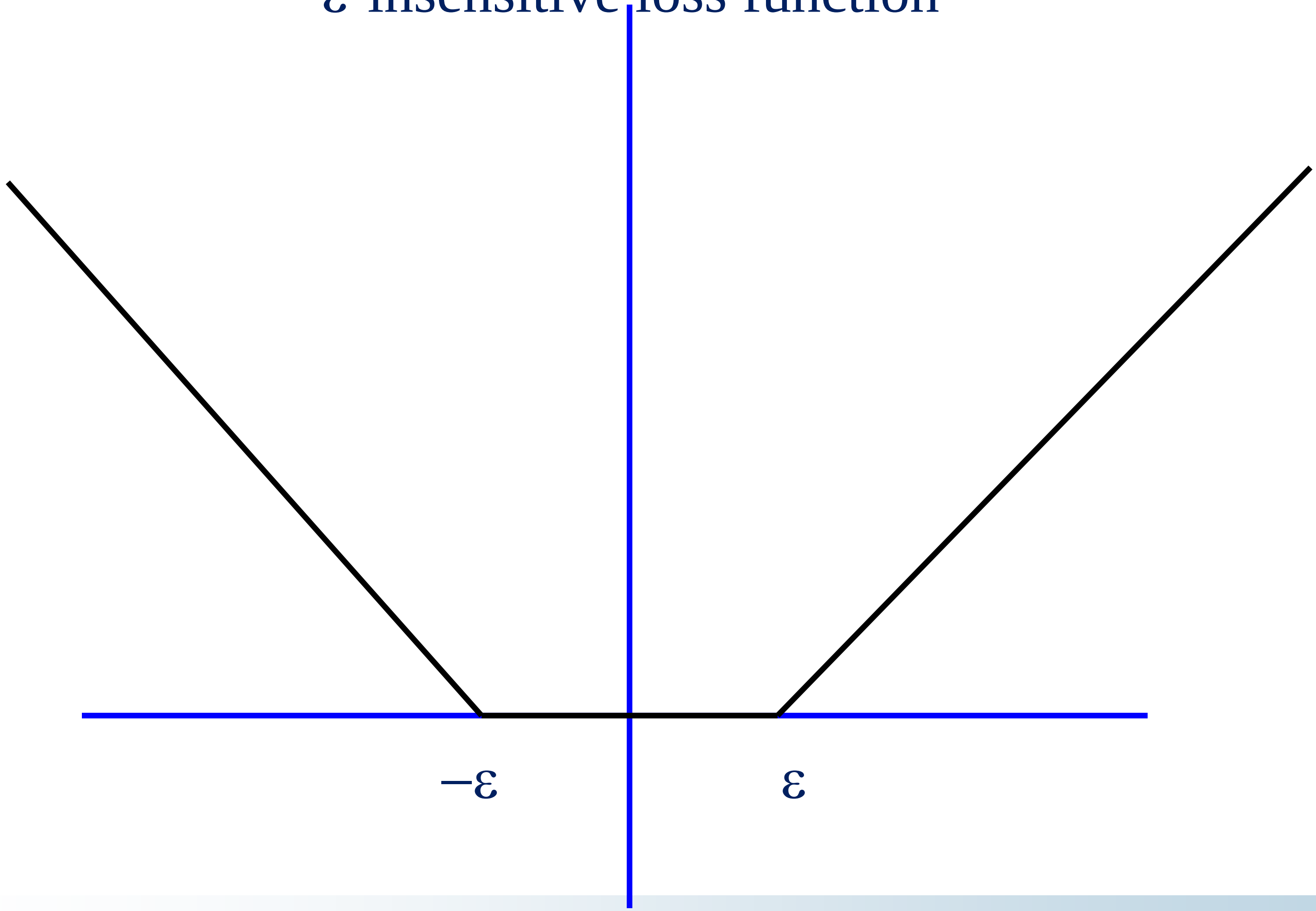


$$err_i = | w.x_i - b - y_i |$$

Actually

- SVR uses the Loss Function below

ε-insensitive loss function



Epsilon Support Vector Regression (ε-SVR)

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

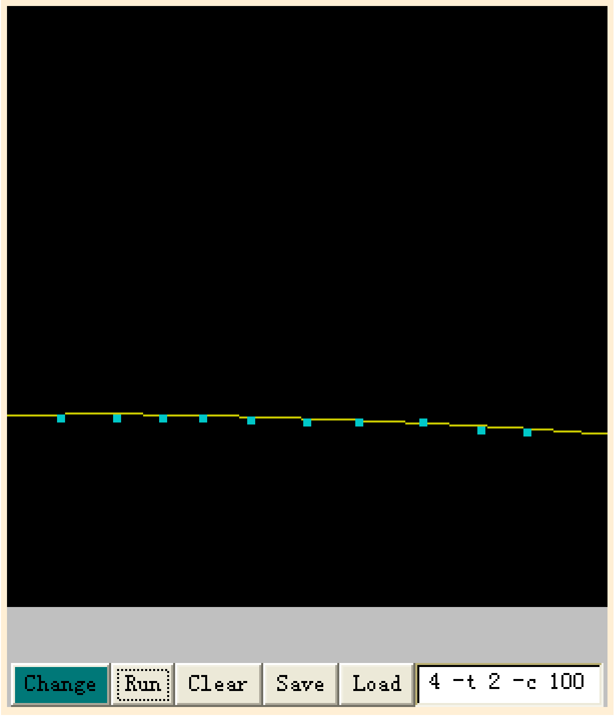
$$\text{Min } \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$

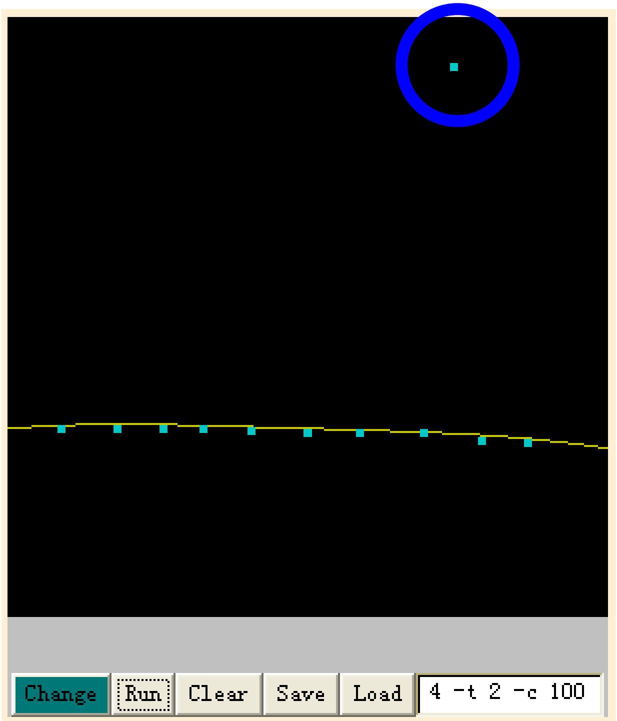
$$\text{subject to } \begin{cases} u_i - \mathbf{w}^T\mathbf{x}_i - b \le \epsilon + \xi_i \\ \mathbf{w}^T\mathbf{x}_i + b - u_i \le \epsilon + \xi_i^* \\ \xi_i \ge 0, \xi_i^* \ge 0 \end{cases}$$

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

Online Demo

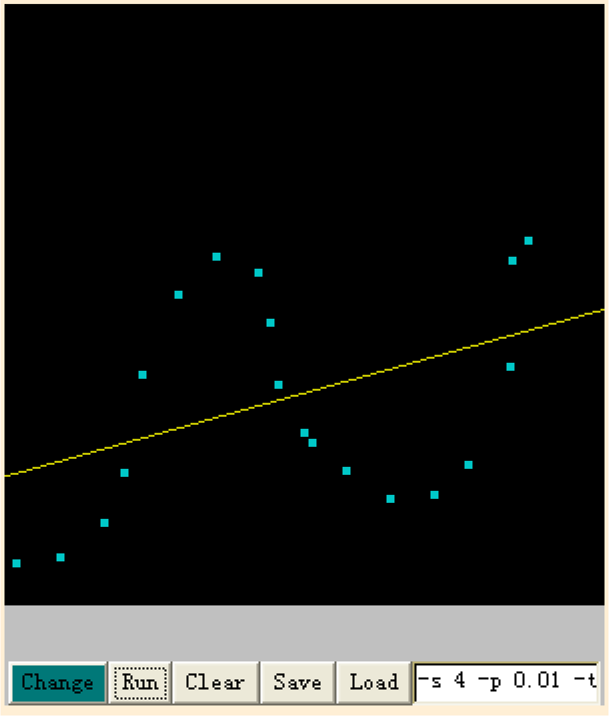
- Less Sensitive to Outlier

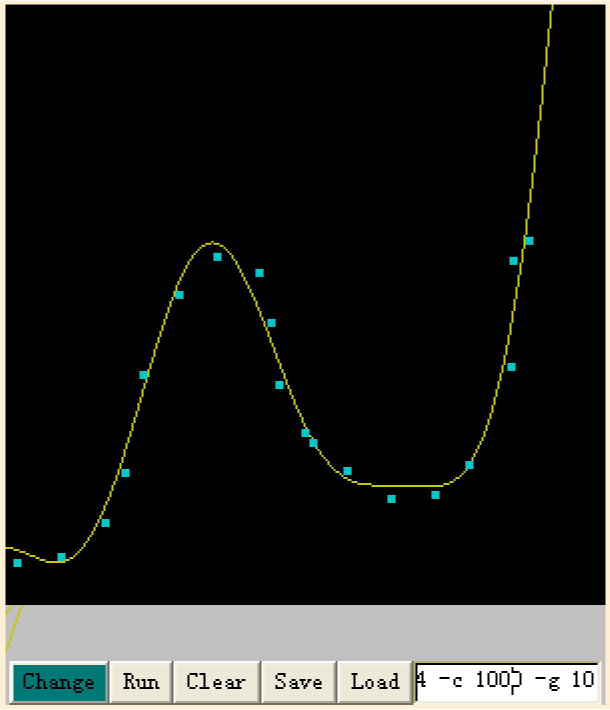




Again, Extend to Non-Linear Case

- Similar with SVM





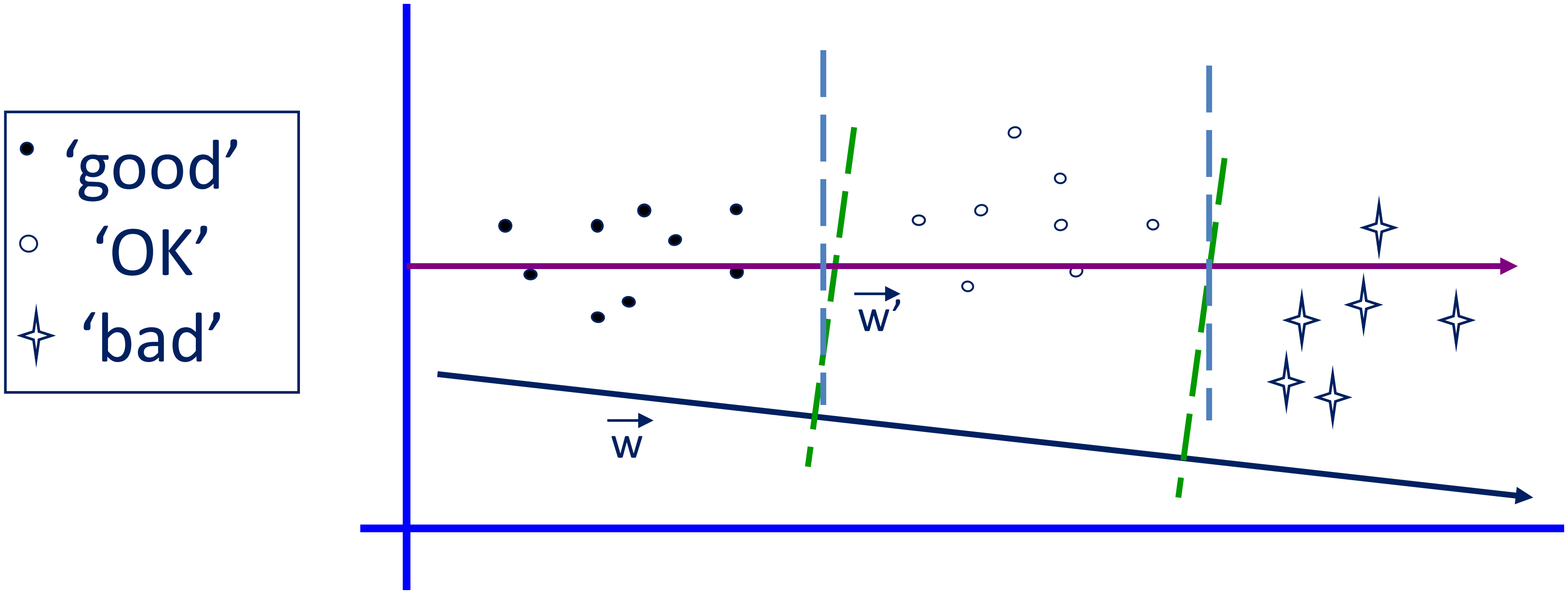
SVM for Ranking

Ranking Problem

- Consider a problem of ranking essays
  - Three ranking categories: good, ok, bad
  - Given a input document, predict its ranking category
- How should we formulate this problem?
- A simple multiple class solution
  - Each ranking category is a independent class
- But, there is something missing here …

- We miss the ordinal relationship between classes !

Ordinal Regression



- Which choice is better?
- How could we formulate this problem?

Ordinal Regression

- What are the two decision boundaries?

$$\mathbf{w}\cdot\mathbf{x}+b_1=0 \text{ and } \mathbf{w}\cdot\mathbf{x}+b_2=0$$

- What is the margin for ordinal regression?

$$\text{margin}_1(\mathbf{w},b_1) \equiv \min_{\mathbf{x}\in D_g\cup D_o} d(\mathbf{x}) = \min_{\mathbf{x}\in D_g\cup D_o} \frac{|\mathbf{x}\cdot\mathbf{w}+b_1|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

$$\text{margin}_2(\mathbf{w},b_2) \equiv \min_{\mathbf{x}\in D_o\cup D_b} d(\mathbf{x}) = \min_{\mathbf{x}\in D_o\cup D_b} \frac{|\mathbf{x}\cdot\mathbf{w}+b_2|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

$$\text{margin}(\mathbf{w},b_1,b_2)=\min(\text{margin}_1(\mathbf{w},b_1),\text{margin}_2(\mathbf{w},b_2))$$

- Maximize margin $\{\mathbf{w}^*,b_1^*,b_2^*\}=\arg\max_{\mathbf{w},b_1,b_2} \text{margin}(\mathbf{w},b_1,b_2)$

Ordinal Regression

$$
\begin{aligned}
\{\mathbf{w}^*, b_1^*, b_2^*\} &= \arg\max_{\mathbf{w}, b_1, b_2} \ \text{margin}(\mathbf{w}, b_1, b_2) \\
&= \arg\max_{\mathbf{w}, b_1, b_2} \ \min(\text{margin}_1(\mathbf{w}, b_1), \text{margin}_2(\mathbf{w}, b_2)) \\
&= \arg\max_{\mathbf{w}, b_1, b_2} \ \min\left(\min_{\mathbf{x} \in D_g \cup D_o} \frac{|\mathbf{x} \cdot \mathbf{w} + b_1|}{\sqrt{\sum_{i=1}^{d} w_i^2}}, \min_{\mathbf{x} \in D_o \cup D_b} \frac{|\mathbf{x} \cdot \mathbf{w} + b_2|}{\sqrt{\sum_{i=1}^{d} w_i^2}} \right)
\end{aligned}
$$

subject to

$$\forall \mathbf{x}_i \in D_g : \mathbf{x}_i \cdot \mathbf{w} + b_1 > 0$$

$$\forall \mathbf{x}_i \in D_o : \mathbf{x}_i \cdot \mathbf{w} + b_1 < 0, \mathbf{x}_i \cdot \mathbf{w} + b_2 > 0$$

$$\forall \mathbf{x}_i \in D_b : \mathbf{x}_i \cdot \mathbf{w} + b_2 < 0$$

- How do we solve this monster ?

Ordinal Regression

- The same old trick
- To remove the scaling invariance, set

$$\forall \mathbf{x}_i \in D_g \cup D_o : \ |b_1 + \mathbf{x}_i \cdot \mathbf{w}| \geq 1$$

$$\forall \mathbf{x}_i \in D_o \cup D_b : \ |b_2 + \mathbf{x}_i \cdot \mathbf{w}| \geq 1$$

- Now the problem is simplified as:

$$\{\mathbf{w}^*, b_1^*, b_2^*\} = \underset{\mathbf{w}, b_1, b_2}{\arg\min} \ \sum_{i=1}^{d} w_i^2$$

subject to

$$\forall \mathbf{x}_i \in D_g : \mathbf{x}_i \cdot \mathbf{w} + b_1 \geq 1$$

$$\forall \mathbf{x}_i \in D_o : \mathbf{x}_i \cdot \mathbf{w} + b_1 \leq -1, \mathbf{x}_i \cdot \mathbf{w} + b_2 \geq 1$$

$$\forall \mathbf{x}_i \in D_b : \mathbf{x}_i \cdot \mathbf{w} + b_2 \leq -1$$

Ordinal Regression

- Noisy case

$$\{\mathbf{w}^*, b_1^*, b_2^*\} = \underset{\mathbf{w}, b_1, b_2}{\arg\min} \sum_{i=1}^{d} w_i^2 + c \sum_{x_i \in D_g} \varepsilon_i^+ + c \sum_{x_i \in D_o} \left(\varepsilon_i^+ + \varepsilon_i^-\right) + c \sum_{x_i \in D_b} \varepsilon_i^-$$

subject to

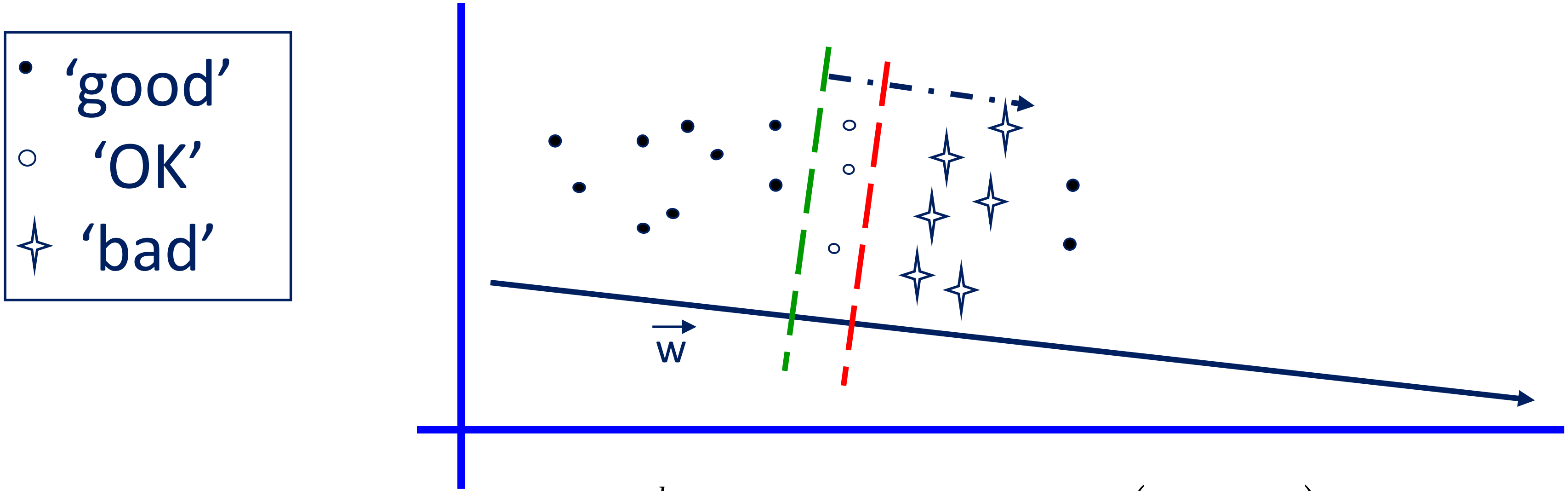
$$\forall \mathbf{x}_i \in D_g : \mathbf{x}_i \cdot \mathbf{w} + b_1 \geq 1 - \varepsilon_i^+, \varepsilon_i^+ \geq 0$$

$$\forall \mathbf{x}_i \in D_o : \mathbf{x}_i \cdot \mathbf{w} + b_1 \leq -1 + \varepsilon_i^-, \mathbf{x}_i \cdot \mathbf{w} + b_2 \geq 1 - \varepsilon_i^+, \varepsilon_i^+ \geq 0, \varepsilon_i^- \geq 0$$

$$\forall \mathbf{x}_i \in D_b : \mathbf{x}_i \cdot \mathbf{w} + b_2 \leq -1 + \varepsilon_i^-, \varepsilon_i^- \geq 0$$

- Is this sufficient enough?

Ordinal Regression



$$\{\mathbf{w}^*, b_1^*, b_2^*\} = \underset{\mathbf{w}, b_1, b_2}{\arg\min} \ \sum_{i=1}^{d} w_i^2 + c \sum_{x_i \in D_g} \varepsilon_i^+ + c \sum_{x_i \in D_o} \left(\varepsilon_i^+ + \varepsilon_i^-\right) + c \sum_{x_i \in D_b} \varepsilon_i^-$$

subject to

$$\forall \mathbf{x}_i \in D_g : \mathbf{x}_i \cdot \mathbf{w} + b_1 \geq 1 - \varepsilon_i^+, \varepsilon_i^+ \geq 0$$

$$\forall \mathbf{x}_i \in D_o : \mathbf{x}_i \cdot \mathbf{w} + b_1 \leq -1 + \varepsilon_i^-, \mathbf{x}_i \cdot \mathbf{w} + b_2 \geq 1 - \varepsilon_i^+, \varepsilon_i^+ \geq 0, \varepsilon_i^- \geq 0$$

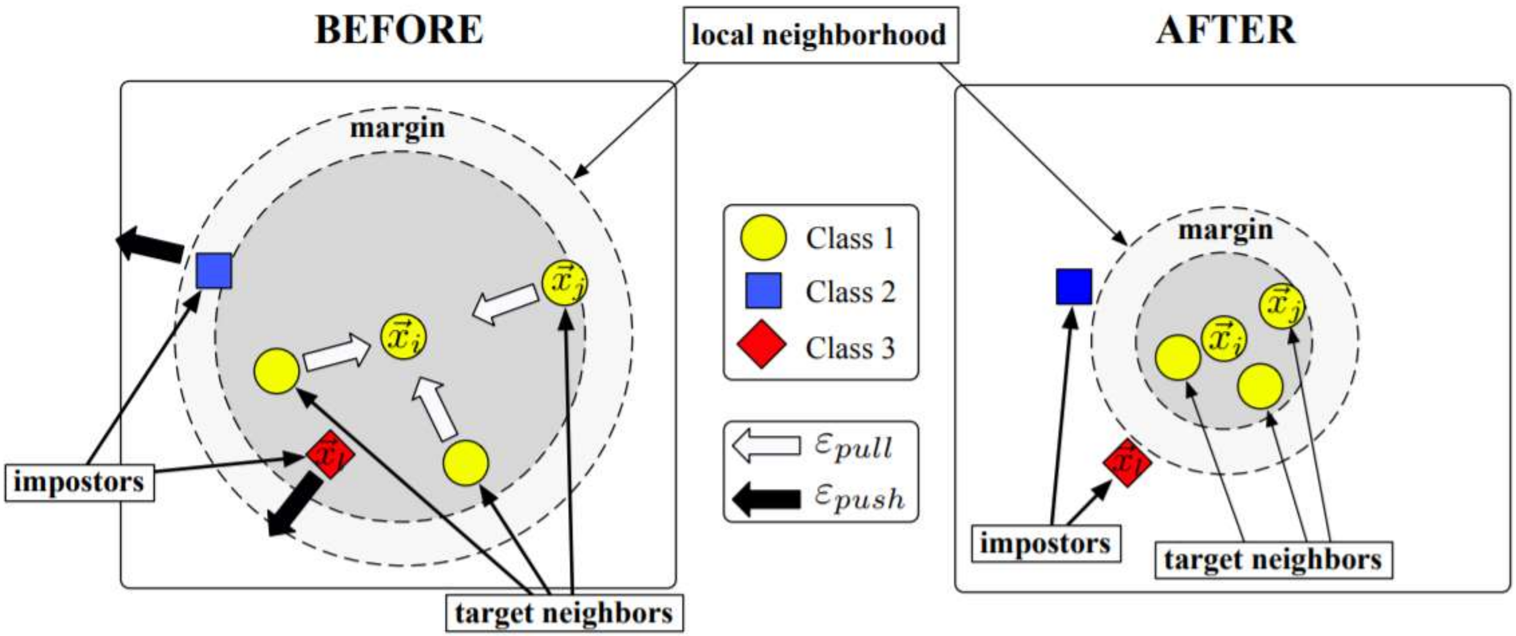
$$\forall \mathbf{x}_i \in D_b : \mathbf{x}_i \cdot \mathbf{w} + b_2 \leq -1 + \varepsilon_i^-, \varepsilon_i^- \geq 0$$

$$\boxed{b_1 \leq b_2}$$

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... ▾ 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

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

- Penalize small distances between differently labeled examples

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

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

- The total loss $\varepsilon(\mathbf{L}) = (1-\mu)\,\varepsilon_{pull}(\mathbf{L}) + \mu\varepsilon_{push}(\mathbf{L}).$
- 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

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

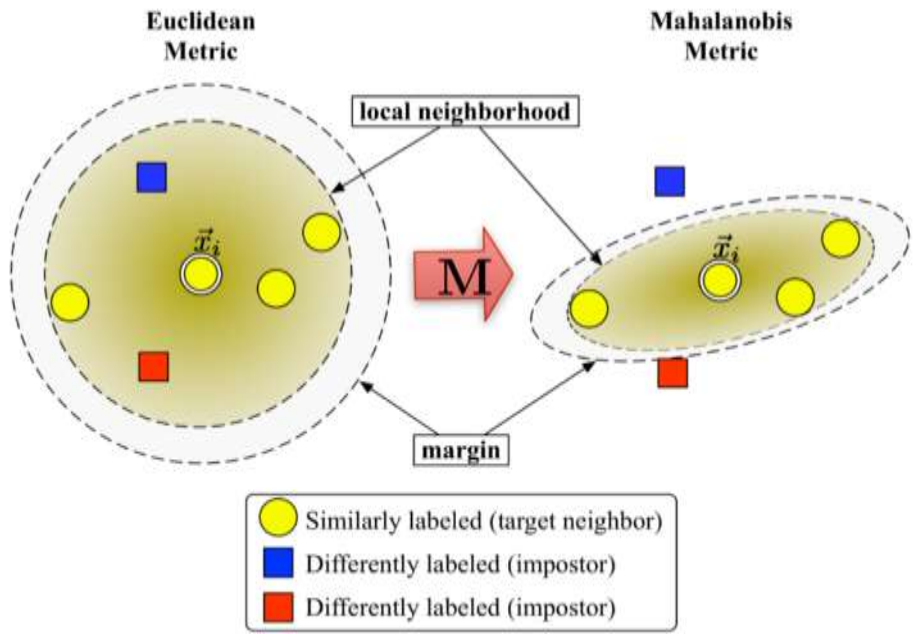
$$\min_{\mathbf{M}} \sum_{i,j \in N_i} d(\vec{x}_i, \vec{x}_j) + \sum_{i,j,l} \xi_{ijl}$$

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

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

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

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



References

- G. Hinton, CSC 2515, Support Vector Machines, 2008.
- Andrew W. Moore, Support Vector Machines, 2001.
- Carla P. Gomes, CS 4700: Foundations of Artificial Intelligence.
- Martin Jaggi, Geometry of Support Vector Machines, 2008.

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
- To be continued ......

Thank You!
Q&A