Neural Network Theory and Applications: Lecture Four

Bao-Liang Lu（吕宝粮）
Dept. of Computer Science & Engineering
Shanghai Jiao Tong University
Office: 3-431 Dianxin Bld, Minhang
Tel: 3420-5422
bllu@sjtu.edu.cn

Outline of Lecture Four

- Extreme Learning Machine (ELM)
- Vapnik–Chervonenkis (VC) dimension
- Support Vector Machine
- Performance evaluation Index

Vapnik–Chervonenkis (VC) dimension

Measures of Complexity

- "Complexity" is a measure of a set of classifiers, not any specific (fixed) classifier
- Many possible measures
  - degrees of freedom
  - description length
  - Vapnik-Chervonenkis dimension

  etc.
- There are many reasons for introducing a measure of complexity
  - generalization error guarantees
  - selection among competing families of classifiers

Shattering a Set of Instances

Definition: a *dichotomy* of a set *S* is a partition of *S* into two disjoint subsets. （二分）

Definition: a set of instances *S* is *shattered* by hypothesis space *H* if and only if for every dichotomy of S there exists some hypothesis in *H* consistent with this dichotomy. （打散）

打散：如果存在一个有h个样本的样本集能够被一个函数集中的函数按照所有可能的2^h种形式分为两类，则称函数集能够将样本数为h的样本集打散；

VC维：如果函数集能够打散h个样本的样本集，而不能打散h+1个样本的样本集，则称函数集的VC维为h。

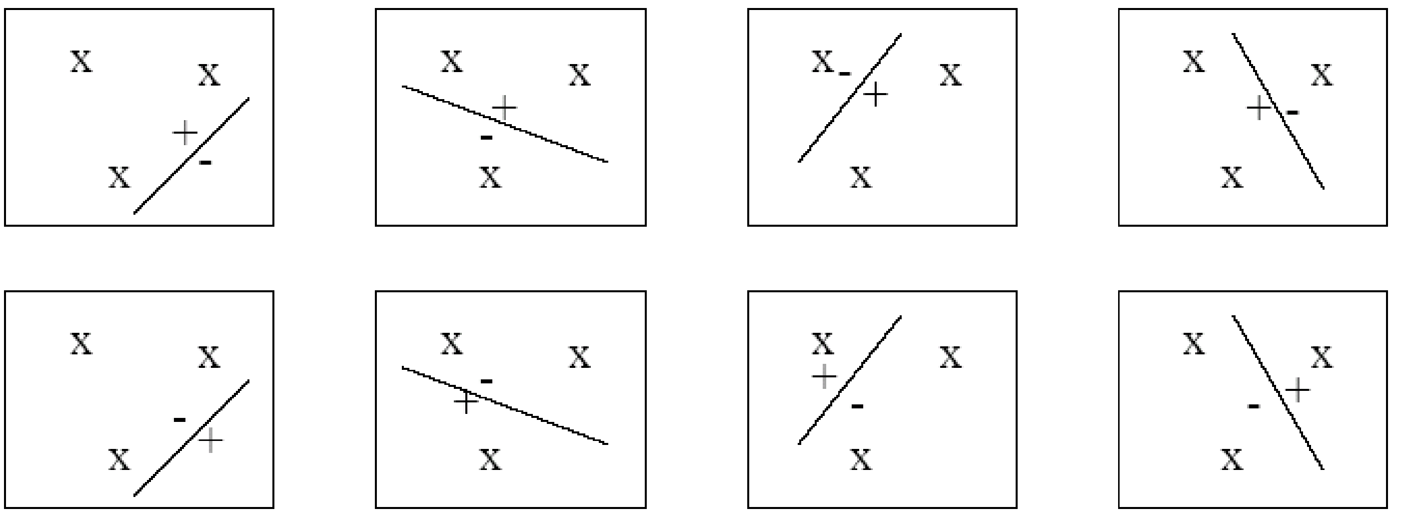
VC–dimension: Shattering

- A set of classifiers F *shatters* n points $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ if

$$[h(\mathbf{x}_1)\ h(\mathbf{x}_2)\ \dots\ h(\mathbf{x}_n)], \quad h \in F$$

generates all 2^n distinct labelings.

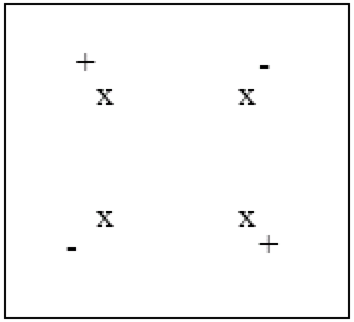
- Example: linear decision boundaries shatter (any) 3 points in 2D



but not any 4 points...

VC–dimension: Shattering–1

- We cannot shatter 4 points in 2D with linear separators
  For example, the following labeling



  cannot be produced with any linear separator

- More generally: the set of all d-dimensional linear separators can shatter exactly $d + 1$ points

VC–dimension

- *The VC-dimension d_{VC} of a set of classifiers F is the largest number of points that F can shatter*
- This is a combinatorial concept and doesn't depend on what type of classifier we use, only how "flexible" the set of classifiers is

Example: Let F be a set of classifiers defined in terms of linear combinations of m **fixed** basis functions

$$h(\mathbf{x}) = \text{sign}\,(\,w_0 + w_1\phi_1(\mathbf{x}) + \ldots + w_m\phi_m(\mathbf{x})\,)$$

d_{VC} is at most $m+1$ regardless of the form of the fixed basis functions.

Bounds on the VC dimension of NNs

- The VC dimension of feedforward network with a threshold function is

  $$O(W \log W)$$

- The VC dimension of feedforward network with a sigmoid function is

  $$O(W^2)$$

Outline of Lecture Four

- Extreme Learning Machine (ELM)
- Vapnik-Chervonenkis (VC) dimension
- Support Vector Machine
- Performance evaluation Index

Support Vector Machine

Introduction

- What are benefits SV learning?
  - Based on simple idea
  - High performance in practical applications
- Characteristics of SV method
  - Can dealing with complex nonlinear problems (pattern recognition, regression, feature extraction)
  - But working with a simple linear algorithm (by the use of kernels)

Empirical Risk

- We want to estimate a function using training data

$$T = (\{X_i, d_i)\}_{i=1}^{N} \rightarrow F(X, W)$$

- Loss between desired response and actual response

$$L(d, F(X,W)) = (d - F(X,W))^2$$

- Expected risk （风险泛函）

$$R(W) = \frac{1}{2}\int L(d, F(X;W))\, dF_{X,D}(X,d)$$

- Empirical risk （经验风险泛函）

$$R_{emp}(W) = \frac{1}{N}\sum_{i=1}^{N} L(d_i, F(X_i, W))$$

Empirical risk minimization principle

- The true expected risk is approximated by empirical risk

$$R_{emp}(W) = \frac{1}{N}\sum_{i=1}^{N} L(d_i, F(X_i, W))$$

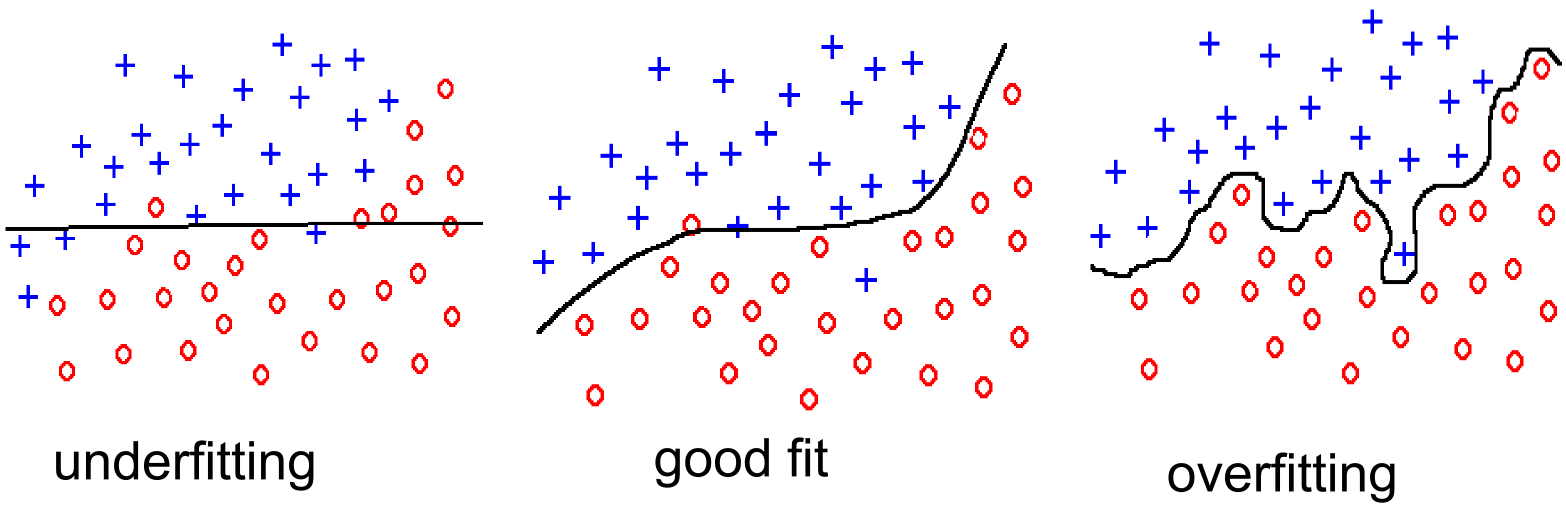
- The learning based on the empirical minimization principle is defined as

$$W^* = \arg\min_{W} \; R_{\text{emp}}(W)$$

Examples of algorithms: Perceptron, Back-propagation, etc.

Overfitting and underfitting

- Problem: How rich class of classifications *F(X,W) to use*



- Problem of generalization: A small empirical risk $R_{emp}(W)$ does not imply small true expected risk $R\ (W)$

Structural Risk Minimization

- Statistical learning theory : Vapnik & Chervonenkis
- An upper bound on the expected risk of a classification rule

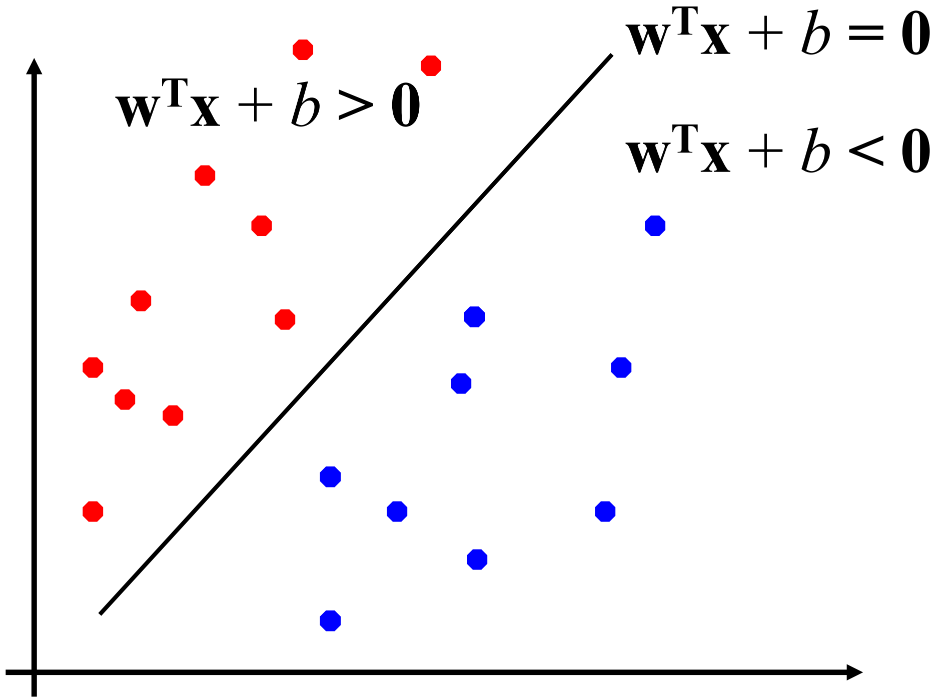
$$R(W) \le R_{emp}(W) + \sqrt{\frac{h[\log(2N/h)+1]-\log(\alpha)}{N}}$$

where N is the number of training data, h is VC-dimension of class of functions.

- SRM Principle: to find a network structure such that decreasing the VC dimension occurs at the expense of the smallest possible increase in training error

Perceptron Revisited: Linear Separators

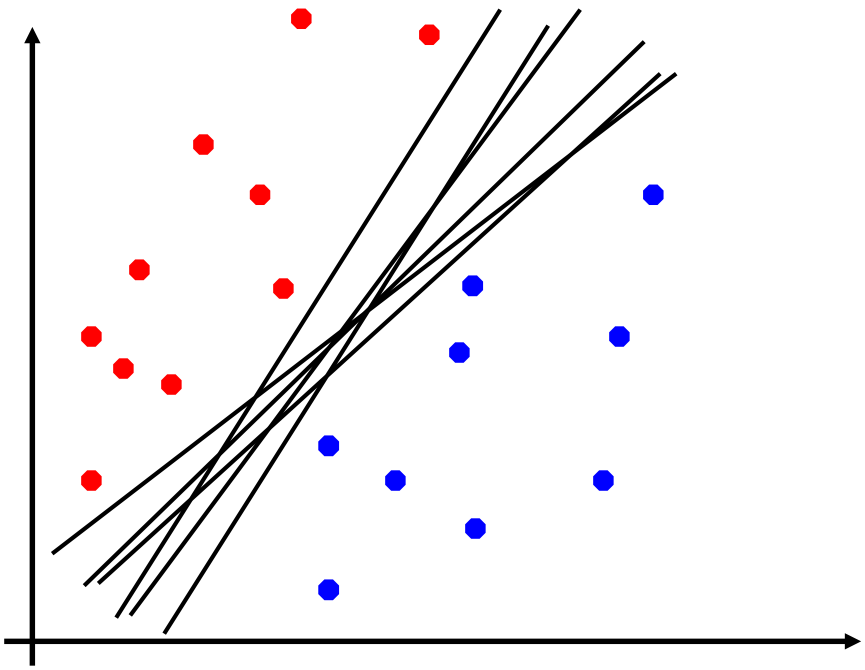
- Binary classification can be viewed as the task of separating classes in feature space:



$$f(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b)$$

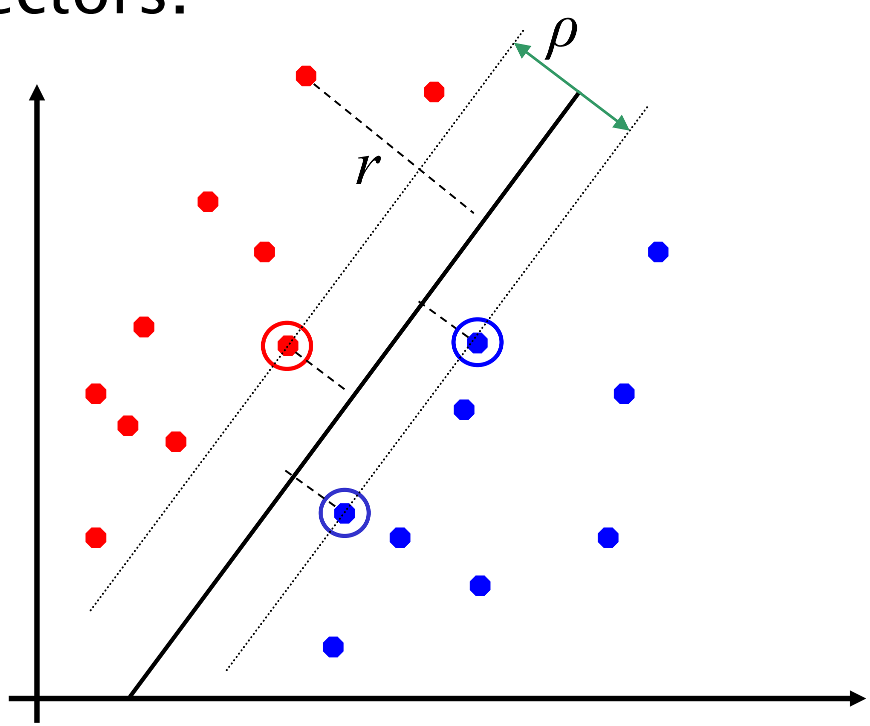
Linear Separators

- Which of the linear separators is optimal?



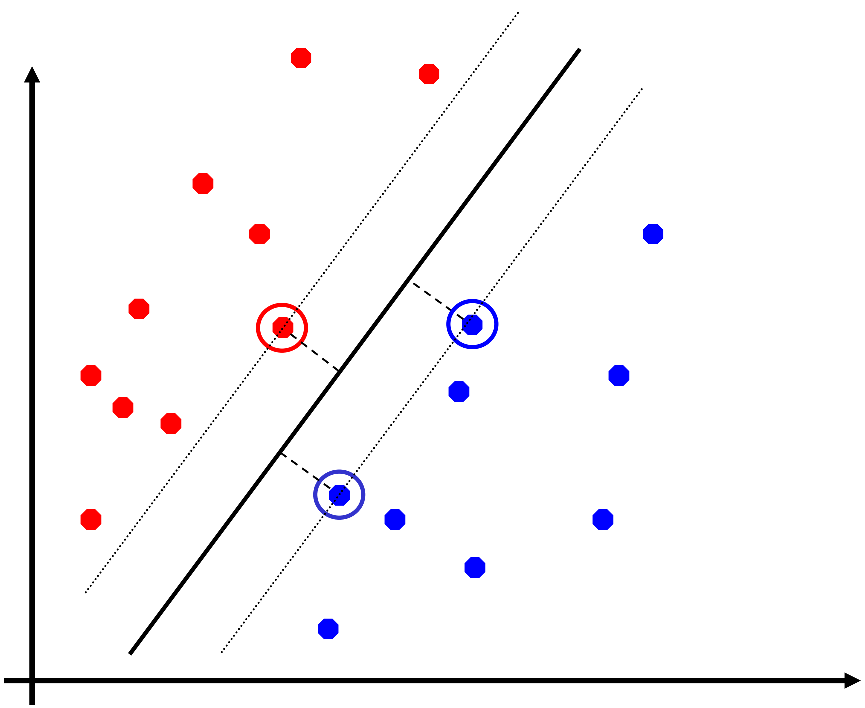
Classification Margin

- Distance from example $\mathbf{x}_i$ to the separator is $r = \dfrac{\mathbf{w}^T\mathbf{x}_i + b}{\|\mathbf{w}\|}$
- Examples closest to the hyperplane are ***support vectors***.
- *Margin* ρ of the separator is the distance between support vectors.



Maximum Margin Classification

- Maximizing the margin is good according to intuition and **probably approximately correct** (PAC) theory.
- Implies that only support vectors matter; other training examples are ignorable.



Non-linear Programming Problem

Consider a problem involving both equality and inequality constraints

$$\text{minimize } f(x)$$

$$\text{subject to } \; h_1(x) = 0, \dots, h_m(x) = 0,$$

$$g_1(x) \leq 0, \dots, g_r(x) \leq 0,$$

where f, h_i, g_j are continuously differentiable functions from R^n to R.

Linear SVM Mathematically

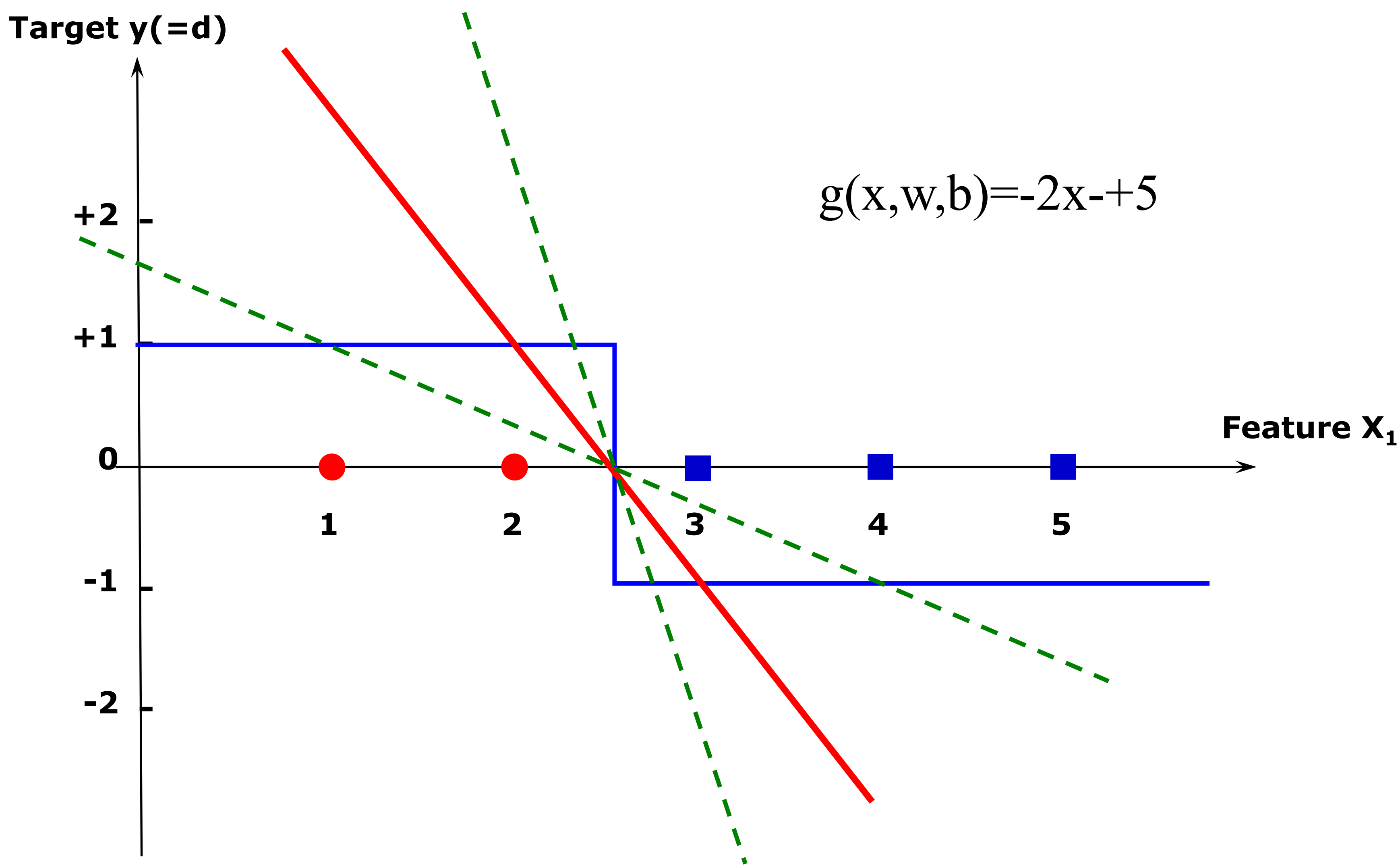
- Let training set $\{(\mathbf{x}_i, y_i)\}_{i=1..n}$, $\mathbf{x}_i \in \mathbf{R}^d$, $y_i \in \{-1, 1\}$ be separated by a hyperplane with margin ρ. Then for each training example $(\mathbf{x}_i, y_i)$:

$$\begin{aligned} &\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b \leq -\rho/2 \quad \text{if } y_i = -1 \\ &\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b \geq \rho/2 \quad \text{if } y_i = 1 \end{aligned} \quad \Leftrightarrow \quad y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) \geq \rho/2$$

- For every support vector $\mathbf{x}_s$ the above inequality is an equality. After rescaling $\mathbf{w}$ and b by $\rho/2$ in the equality, we obtain that distance between each $\mathbf{x}_s$ and the hyperplane is

$$r = \frac{y_s(\mathbf{w}^T\mathbf{x}_s + b)}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

- Then the margin can be expressed through (rescaled) $\mathbf{w}$ and b as: $\rho = 2r = \dfrac{2}{\|\mathbf{w}\|}$



Target y(=d)
g(x,w,b)=-2x-+5
+2
+1
0
-1
-2
Feature X1
1
2
3
4
5

Linear SVMs Mathematically (cont.)

- Then we can formulate the *quadratic optimization problem:*

Find $\mathbf{w}$ and b such that

$$\rho = \frac{2}{\|\mathbf{w}\|} \quad \text{is maximized}$$

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $\quad y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) \geq 1$

Which can be reformulated as:

Find $\mathbf{w}$ and b such that

$$\mathbf{\Phi}(\mathbf{w}) = ||\mathbf{w}||^2 = \mathbf{w}^{\mathrm{T}}\mathbf{w} \quad \text{is minimized}$$

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $\quad y_i\,(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) \geq 1$

Solving the Optimization Problem

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}^T\mathbf{w}$ is minimized
and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i (\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

- Need to optimize a quadratic function subject to linear constraints.
- Quadratic optimization problems are a well-known class of mathematical programming problems for which several (non-trivial) algorithms exist.
- The solution involves constructing a dual problem where a Lagrange multiplier αi is associated with every inequality constraint in the primal (original) problem:

Find $\alpha_1 \ldots \alpha_n$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

The Optimization Problem Solution

- Given a solution α1…αn to the dual problem, solution to the primal is:

$$\mathbf{w} = \Sigma \alpha_i y_i \mathbf{x}_i \qquad b = y_k - \Sigma \alpha_i y_i \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_k \quad \text{for any } \alpha_k > 0$$

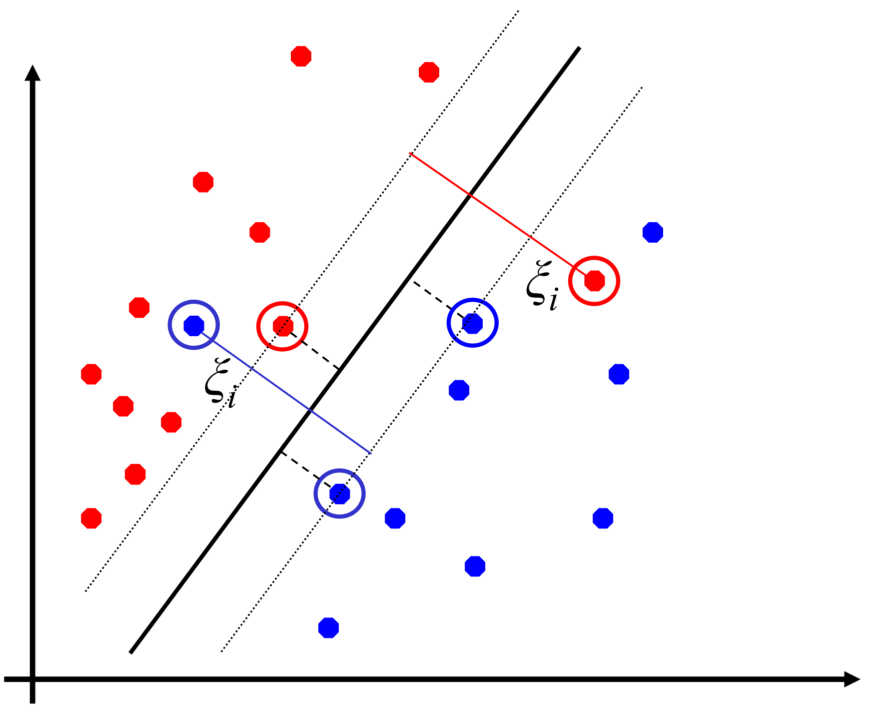
- Each non-zero αi indicates that corresponding xi is a support vector.
- Then the classifying function is (note that we don’t need **w** explicitly):

$$f(\mathbf{x}) = \Sigma \alpha_i y_i \mathbf{x}_i^{\mathrm{T}} \mathbf{x} + b$$

- Notice that it relies on an inner product between the test point x and the support vectors xi
- Also keep in mind that solving the optimization problem involved computing the inner products $x_i^T x_j$ between all training points.

Soft Margin Classification

- What if the training set is not linearly separable?
- Slack variables ξi can be added to allow misclassification of difficult or noisy examples, resulting margin called soft.



Soft Margin Classification Mathematically

- The old formulation:

> Find $\mathbf{w}$ and b such that
> $\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{w}$ is minimized
> and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $\quad y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) \geq 1$

- Modified formulation incorporates slack variables:

> Find $\mathbf{w}$ and b such that
> $\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{w} + C\Sigma\xi_i$ is minimized
> and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $\quad y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$

- Parameter C can be viewed as a way to control overfitting: it "trades off" the relative importance of maximizing the margin and fitting the training data.

Soft Margin Classification – Solution

- Dual problem is identical to separable case (would not be identical if the 2-norm penalty for slack variables CΣξi2 was used in primal objective, we would need additional Lagrange multipliers for slack variables):

Find $\alpha_1 \ldots \alpha_N$ such that

$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j$ is maximized and

(1) $\Sigma\alpha_i y_i = 0$

(2) $0 \leq \alpha_i \leq C$ for all α_i

- Again, xi with non-zero αi will be support vectors.
- Solution to the dual problem is:

$\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x}_i$

$b = y_k(1 - \xi_k) - \Sigma\alpha_i y_i \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_k$ for any k s.t. $\alpha_k > 0$

Again, we don't need to compute **w** explicitly for classification:

$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^{\mathbf{T}}\mathbf{x} + b$

SVM Boundaries with different C

Find $\mathbf{w}$ and b such that

$\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{w} + C\Sigma\xi_i$ is minimized

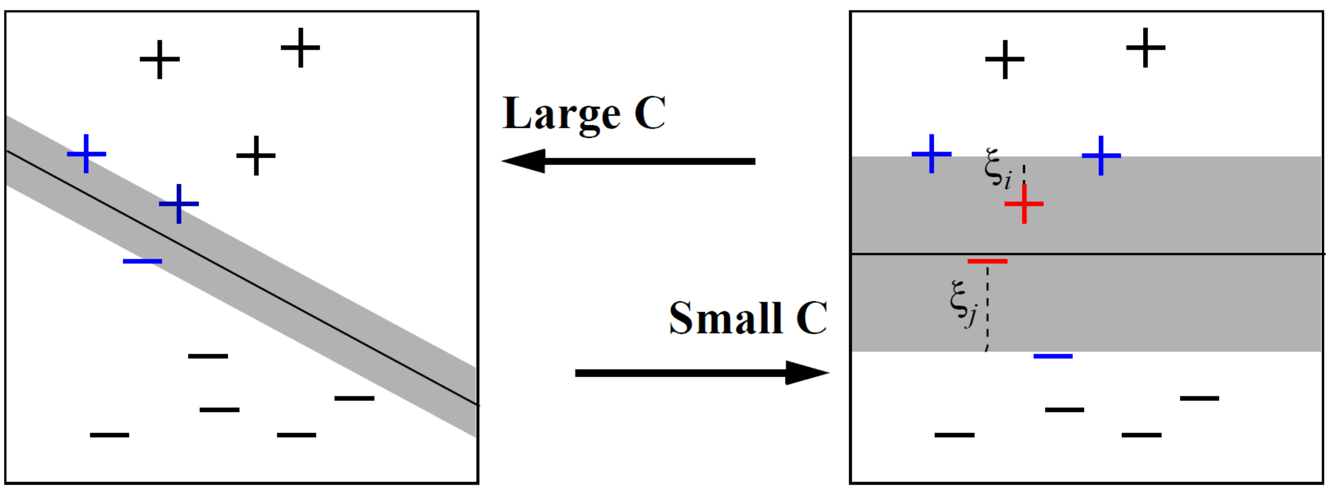
and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) \geq 1 - \xi_i$, $\xi_i \geq 0$

Find $\alpha_1 \ldots \alpha_N$ such that

$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j$ is maximized and

(1) $\Sigma\alpha_i y_i = 0$

(2) $0 \leq \alpha_i \leq C$ for all α_i



Theoretical Justification for Maximum Margins

- Vapnik has proved the following:

  The class of optimal linear separators has VC dimension h bounded from above as

  $$\boxed{h \le \min\left\{\left\lceil \frac{D^2}{\rho^2} \right\rceil, m_0 \right\} + 1}$$

  where ρ is the margin, D is the diameter of the smallest sphere that can enclose all of the training examples, and m0 is the dimensionality.
- Intuitively, this implies that regardless of dimensionality m0 we can minimize the VC dimension by maximizing the margin ρ.
- Thus, complexity of the classifier is kept small regardless of dimensionality.

Linear SVMs: Overview

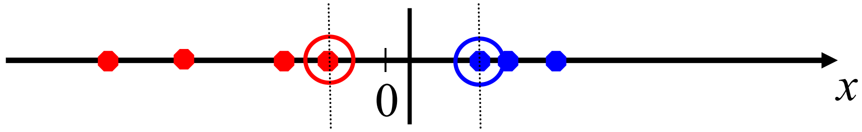
- The classifier is a separating hyperplane.
- Most "important" training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points xi are support vectors with non-zero Lagrangian multipliers αi.
- Both in the dual formulation of the problem and in the solution training points appear only inside inner products:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \boxed{\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

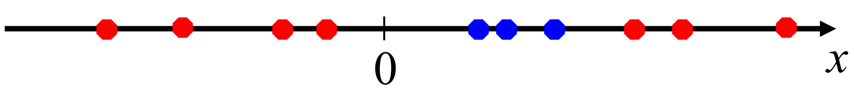
$$f(\mathbf{x}) = \Sigma\alpha_i y_i \boxed{\mathbf{x}_i^{\mathrm{T}}\mathbf{x}} + b$$

Non-linear SVMs

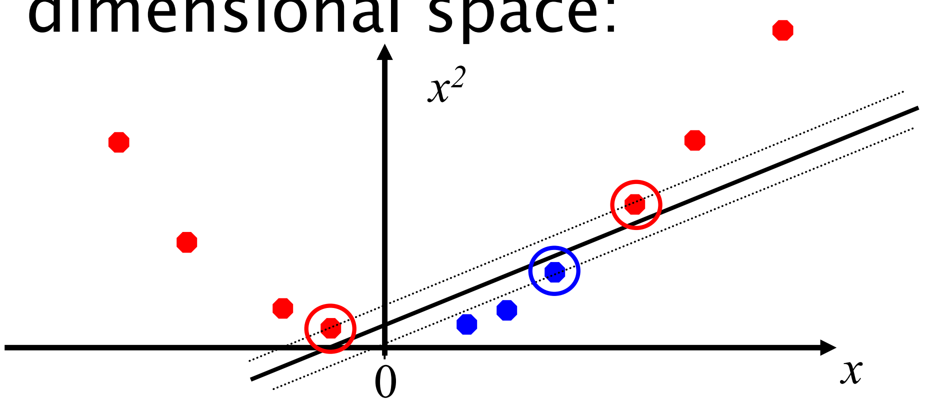
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

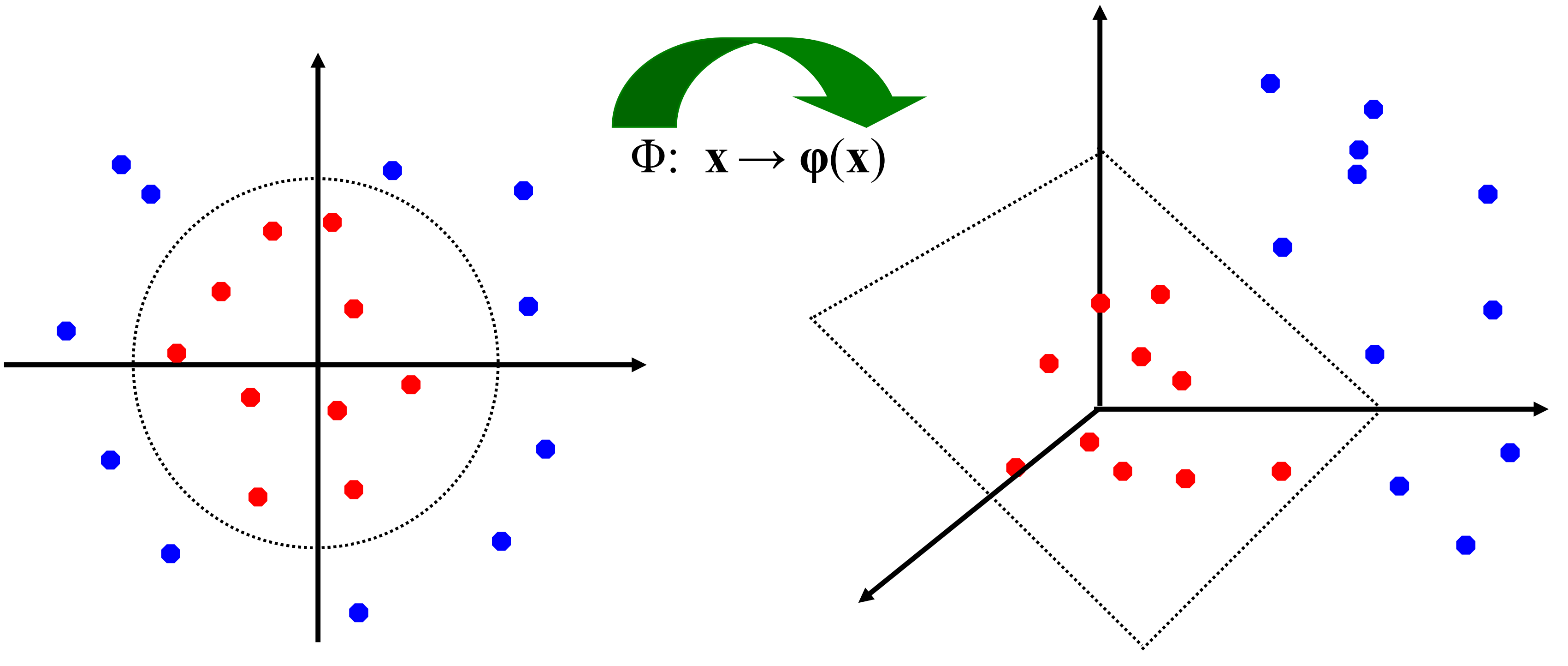


- How about… mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



The "Kernel Trick"

- The linear classifier relies on inner product between vectors $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T \boldsymbol{x}_j$
- If every datapoint is mapped into high-dimensional space via some transformation $\Phi: \boldsymbol{x} \rightarrow \varphi(\boldsymbol{x})$, the inner product becomes:
  $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x}_j)$
- A kernel function is a function that is eqiuvalent to an inner product in some feature space.

Example:

2-dimensional vectors $\boldsymbol{x} = [x_1 \;\; x_2]$; let $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1 + \boldsymbol{x}_i^T \boldsymbol{x}_j)^2$,

Need to show that $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x}_j)$:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1 + \boldsymbol{x}_i^T \boldsymbol{x}_j)^2 = 1 + x_{i1}^2 x_{j1}^2 + 2x_{i1}x_{j1}x_{i2}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$$
$$= [1 \;\; x_{i1}^2 \;\; \sqrt{2x_{i1}x_{i2}} \;\; x_{i2}^2 \;\; \sqrt{2x_{i1}} \;\; \sqrt{2x_{i2}}]^{\mathrm{T}} [1 \;\; x_{j1}^2 \;\; \sqrt{2x_{j1}x_{j2}} \;\; x_{j2}^2 \;\; \sqrt{2x_{j1}} \;\; \sqrt{2x_{j2}}]$$
$$= \varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x}_j),$$

where $\varphi(\boldsymbol{x}) = [1 \;\; x_1^2 \;\; \sqrt{2x_1x_2} \;\; x_2^2 \;\; \sqrt{2x_1} \;\; \sqrt{2x_2}]$

Thus, a kernel function implicitly maps data to a high-dimensional space (without the need to compute each $\varphi(\boldsymbol{x})$ explicitly).

What Functions are Kernels?

- For some functions $K(x_i, x_j)$ checking that $K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$ can be cumbersome.
- Mercer's theorem:
  Every semi-positive definite symmetric function is a kernel
- Semi-positive definite symmetric functions correspond to a semi-positive definite symmetric Gram matrix:

K=

$K(\mathbf{x}_1,\mathbf{x}_1)$	$K(\mathbf{x}_1,\mathbf{x}_2)$	$K(\mathbf{x}_1,\mathbf{x}_3)$	$\ldots$	$K(\mathbf{x}_1,\mathbf{x}_n)$
$K(\mathbf{x}_2,\mathbf{x}_1)$	$K(\mathbf{x}_2,\mathbf{x}_2)$	$K(\mathbf{x}_2,\mathbf{x}_3)$		$K(\mathbf{x}_2,\mathbf{x}_n)$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
$K(\mathbf{x}_n,\mathbf{x}_1)$	$K(\mathbf{x}_n,\mathbf{x}_2)$	$K(\mathbf{x}_n,\mathbf{x}_3)$	$\ldots$	$K(\mathbf{x}_n,\mathbf{x}_n)$

Examples of Kernel Functions

- **Linear kernel**

$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}')$$

- **Polynomial kernel**

$$K(\mathbf{x}, \mathbf{x}') = \left(1 + (\mathbf{x}^T \mathbf{x}')\right)^p$$

where $p = 2, 3, \ldots$. To get the feature vectors we concatenate all p^{th} order polynomial terms of the components of $\mathbf{x}$ (weighted appropriately)

- **Radial basis kernel**

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|^2\right)$$

In this case the feature space consists of functions and results in a *non-parametric* classifier.

Non-linear SVMs Mathematically

- Dual problem formulation:

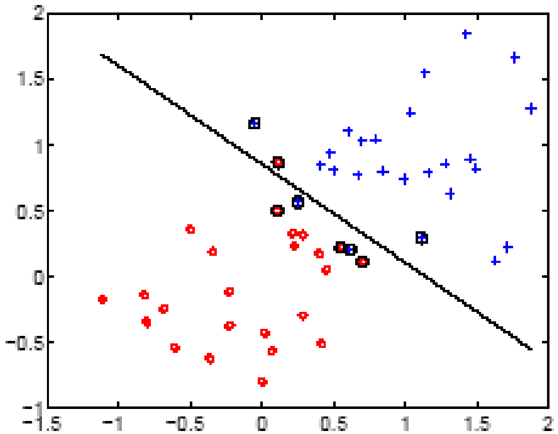
Find $\alpha_1 \ldots \alpha_n$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - ½\Sigma\Sigma\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

- The solution is:

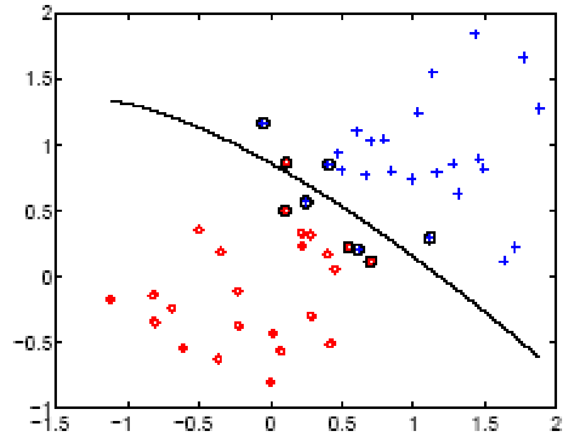
$f(\mathbf{x}) = \Sigma\alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b$

- Optimization techniques for finding α_i' s remain the same!

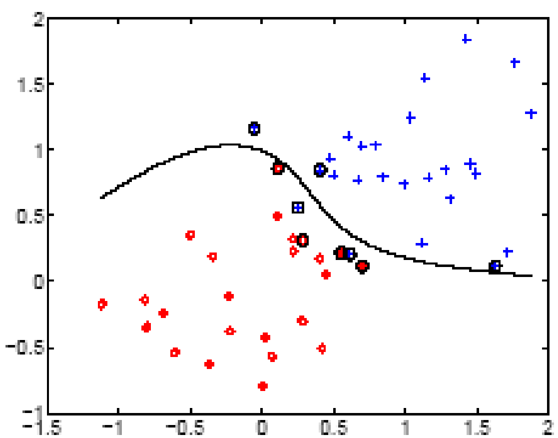
SVM Examples



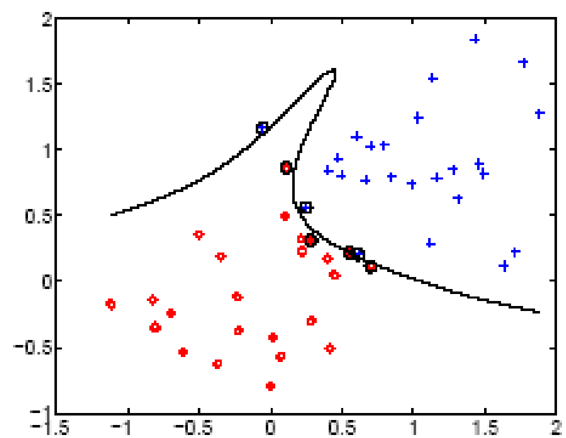
linear



2^{nd} order polynomial



4^{th} order polynomial



8^{th} order polynomial

Key Points

- Learning depends only on dot products of sample pairs.
- Exclusive reliance on dot products enables approach to non-linearly separable problems.
- The classifier depends only on the support vectors, not on all the training points.
- Max margin lowers hypothesis variance.
- The optimal classifier is defined uniquely–there are no "local maxima" in the search space
- Polynomial in number of data points and dimensionality

Limitations of traditional Methods

- Some of the two-class problems may fall into a load imbalance situation because the size of each class may be very imbalance in some problems. (eg. Forest CoverType).
- Using the one-versus-one, some of the two-class problems may still be too large to learn.

Min-Max Modular SVM

- Dividing a K-class problem into K(K-1)/2 two-class problems.
- These two-class problems can be further be decomposed into a number of relatively smaller and simplifier subproblems.
- These subproblems are independent from each other in learning phase, so they can be easily trained in a parallel way.

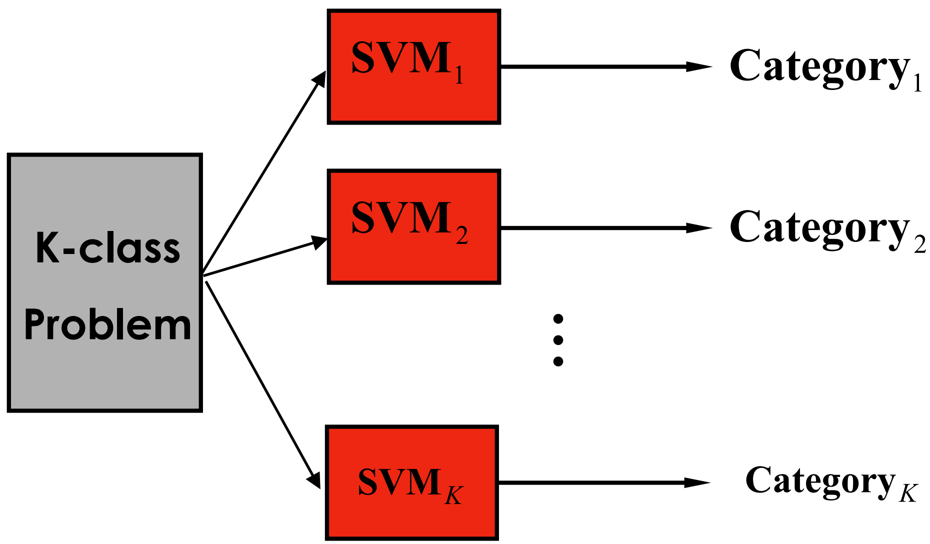
SVMs for Multi-class Classification Problems

Three task decomposition methods:

- One-versus-rest
- One-versus-one
- Part-versus-part

One-Versus-Rest method

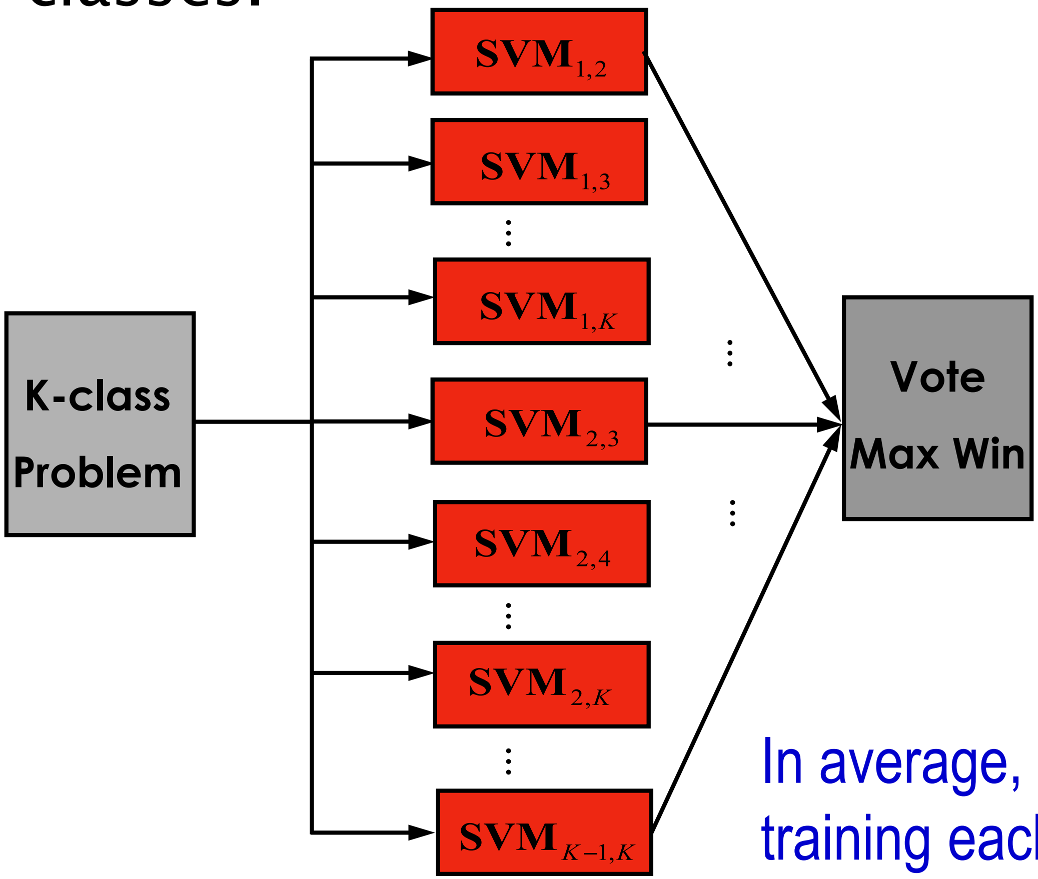
- This method requires one classifier per category. The ith SVM will be trained with all of the examples in the ith class with positive labels, and all other examples with negative labels.



The Number of training data for each classifier is *N*

One–Versus–One Method

- This method constructs K(K–1)/2 classifiers where each one is trained on data from two out of K classes.



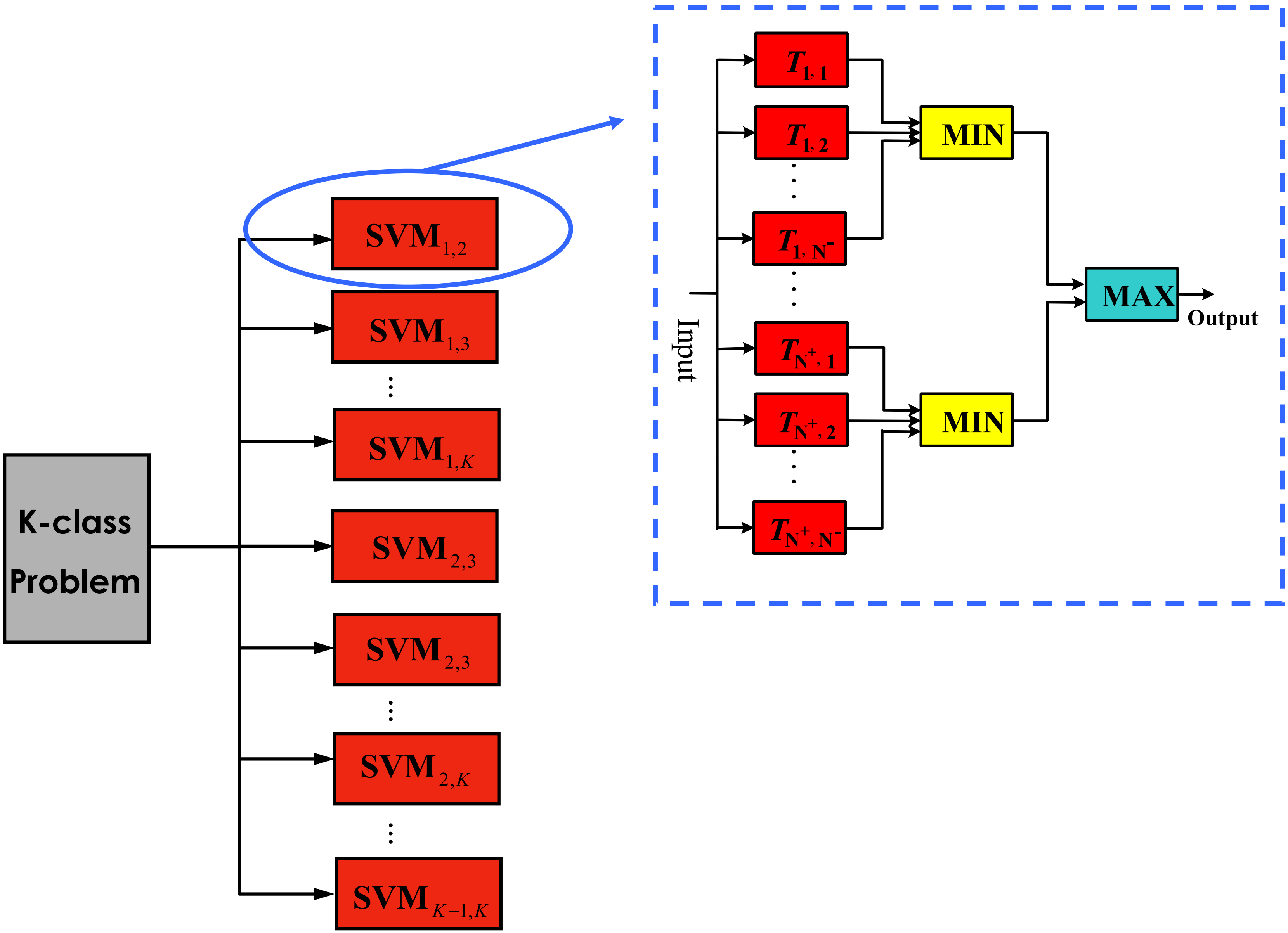
In average, number of data for training each classifier is ***2N/K***

Limitations of traditional Methods

- Some of the two-class problems may fall into a load imbalance situation for the size of each class may be very imbalance in some problems.
- Using the one-versus-one, some of the two-class problems may still be too large to learn.

Part-versus-part

- Part-vs-part: Any two-class problem can be further decomposed into a number of two-class sub-problems as small as needed.



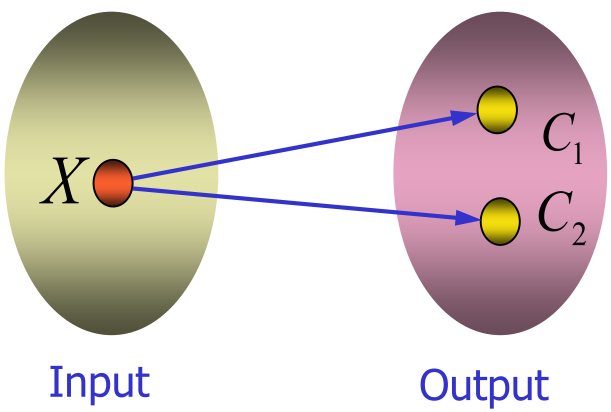
K-class
Problem
SVM 1,2
SVM 1,3
SVM 1,K
SVM 2,3
SVM 2,3
SVM 2,K
SVM K−1,K
Input
T 1, 1
T 1, 2
T 1, N-
T N+, 1
T N+, 2
T N+, N-
MIN
MIN
MAX
Output

Advantages of part-versus-part method

- A large-scale two-class problem can be divided into a number of relatively smaller two-class problems
- A serious imbalance two-class problem can be divided into a number of balance two-class problems
- Massively parallel learning can be easily implemented

What is a multi-label problem ?

- For a given training input *x*, there are *n* (n>1) labels, *yi* ($i=1,\ldots,n$), corresponding to the training input *x*
- Multi-label problems can not be directly solved by using conventional learning frameworks because a one-to-many mapping should be created



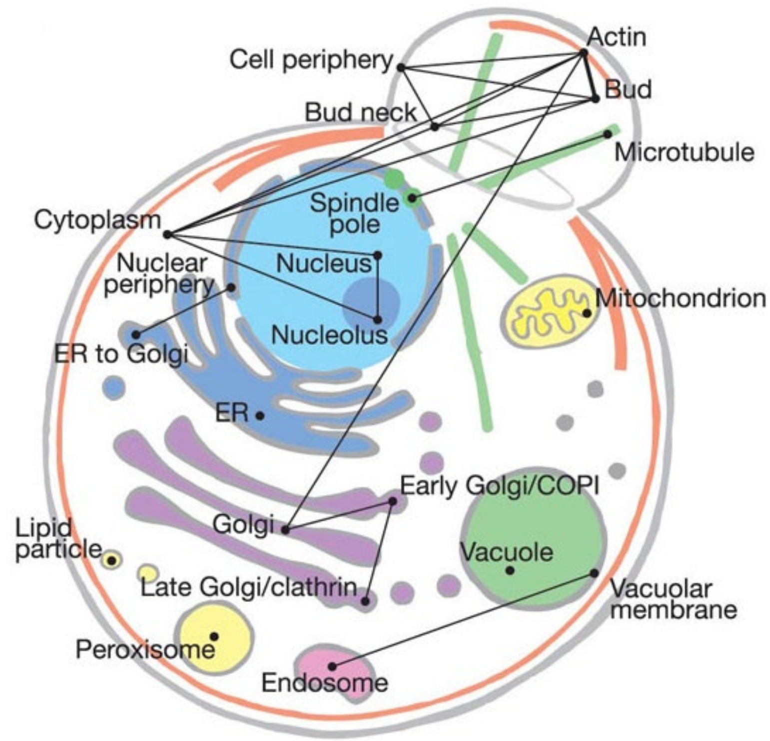
Multi-label problems DO Exit !

- Text categorization

  There are 1.7 labels for each document in average at Yomiuri News corpus

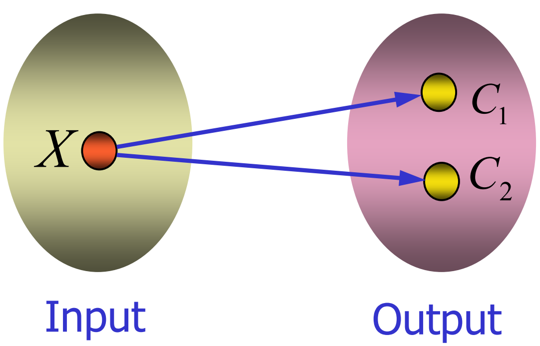
- Subcellular localization of protein subsequence

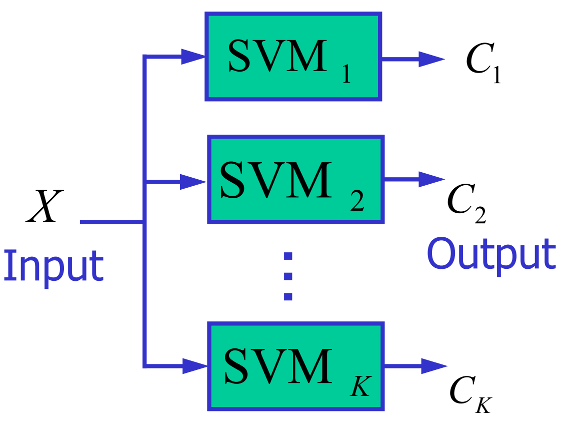
  One protein sequence has at most 5 locations in budding yeast



Existing Method for Multi-Label Problem

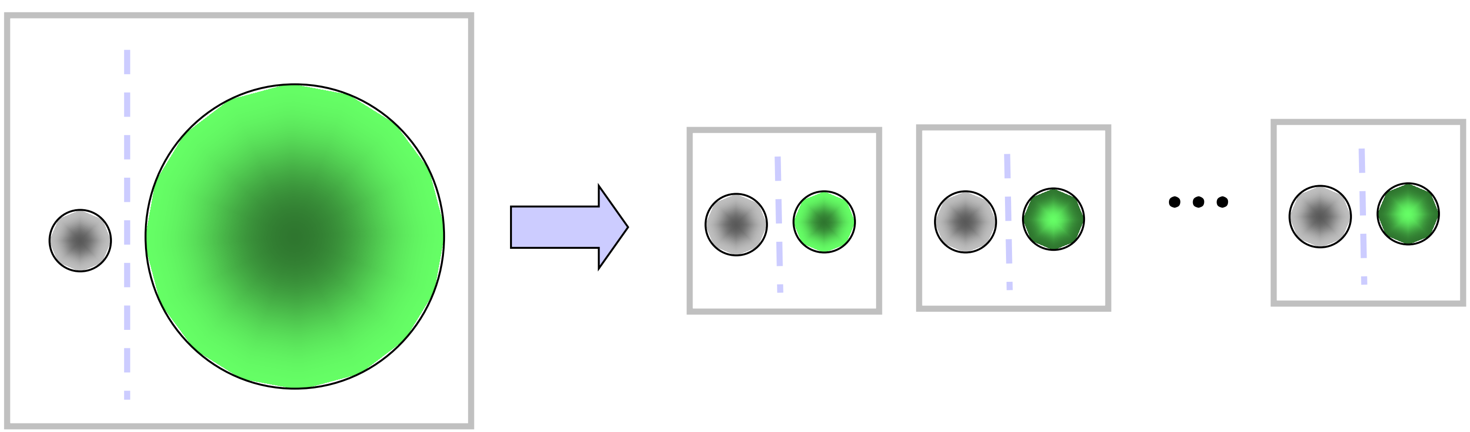
- Divide a K-class multi-label problem into K two-class problems using one-versus-rest method.
- Shortcoming: each of the two class problems will be a serious imbalance and large-scale one.





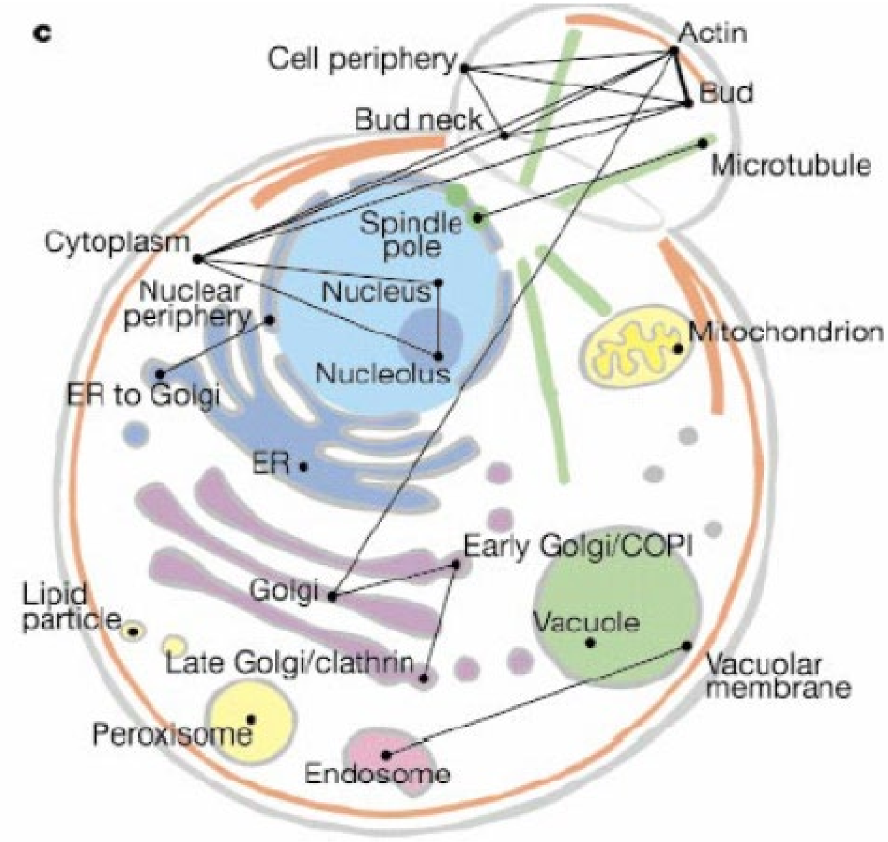
Part–versus–part method

- Divide a K–class multi–label problem into K two–class problems using one–versus–rest method
- Divide each of the imbalance or large–scale two–class problems into a number of relatively more balance and smaller two–class subproblems.



Protein Subcellular Localization

- The function of a protein is closely correlated with its subcellular location.
- Since more and more protein sequences enter into public database, extracting the sequence information for predicting protein subcellular location becomes very important.
- Multi-location problem: One protein sequence has at most 5 locations in yeast cells.



酵母数据集分布

亚细胞位置	序列个数	亚细胞位置	序列个数
Actin	29	Lipid particle	19
Bud	23	Microtubule	20
Bud neck	60	Mitochondrion	494
Cell periphery	106	Nuclear periphery	59
Cytoplasm	1576	Nucleolus	157
Early Golgi	51	Nucleus	1333
Endosome	43	Peroxisome	20
ER	272	Punctate composite	123
ER to Golgi	6	Spindle pole	58
Golgi	40	Vacuolar membrane	54
Late Golgi	37	vacuole	129
总标号数		4709	
蛋白质总数		3555	

Experimental Result

- 22-label classification Problem
  - One-vs-rest
  - Build 22 SVM classifiers corresponding to 22 subcellular locations.
- Divide big class to smaller parts
  - Module =50, 100,500,1000
- 10-fold cross-validation

实验结果

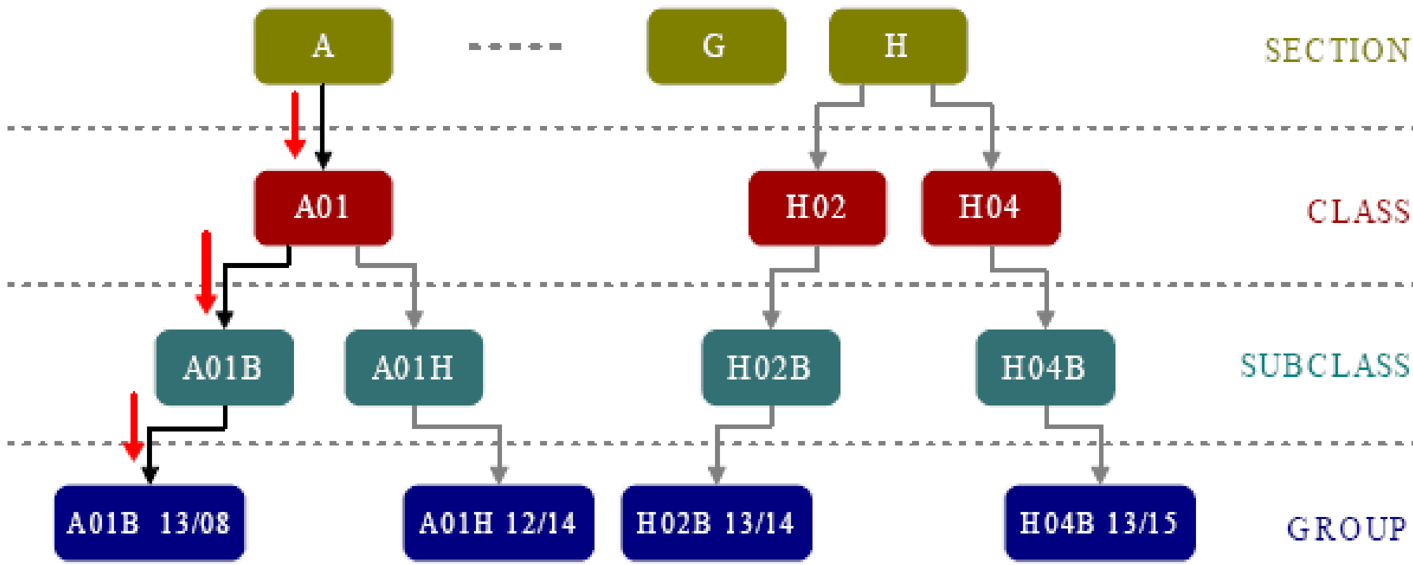
分类器	总体准确率(%)	位置准确率(%)	宏观平均(%)	微观平均(%)
M^3-SVM(50)	74.3	**59.4**	54.0	68.8
M^3-SVM(100)	**74.4**	58.7	55.5	69.7
M^3-SVM(500)	73.4	53.6	**56.7**	73
M^3-SVM(1000)	71.3	52.1	56.4	**73.5**
SVM	69.3	49.1	55.6	73.8
NN	67	48.3	49.5	67.5

分类器	串行运行时间 （秒）	最大模块运行时间 （秒）
SVM	33.3	5.7
M^3SVM(50)	36.3	<0.1
M^3SVM(100)	26.3	<0.1
M^3-SVM(500)	23.5	0.6
M^3SVM(1000)	25.2	2.3

An example of Japanese patent

	PATENT-JA-UPA-1998-000001
<Bibliography>	
[publication date]	(43)【公開日】平成１０年（１９９８）１月６日
[title of invenction]	(54)【発明の名称】土壌改良方法とその作業機
...	...
<Abstract>	
[purpose]	【課題】心土破砕、特に雪上心土破砕作業の際に積雪...
[solution]	【解決手段】心土破砕を行うために用いるサブソイウの...
...	...
<Claims>	
[claim1]	【請求項１】サブソイウ作業機を用いて心土破砕作業...
[claim2]	【請求項２】サブソイウ作業機において、そのナイフ...
...	...
<Description>	
[technique field]	【発明の属する技術分野】本発明は、土壌改良方法とそ...
[prior art]	【従来の技術】圃場の表面がまだ積雪に覆われている状...
[problem to be solved]	【発明が解決しようとする課題】心土破砕は通常春先に...
[means of solving problems]	【課題を解決するための手段】述のような目的達成す...
[effects of invention]	【発明の効果】以上の説明から明らかなように、本発明...
...	...
< Explanation of Drawing >	
[figure1]	【図１】本発明を施す圃場断面図である。
...	...

Structure vs distribution

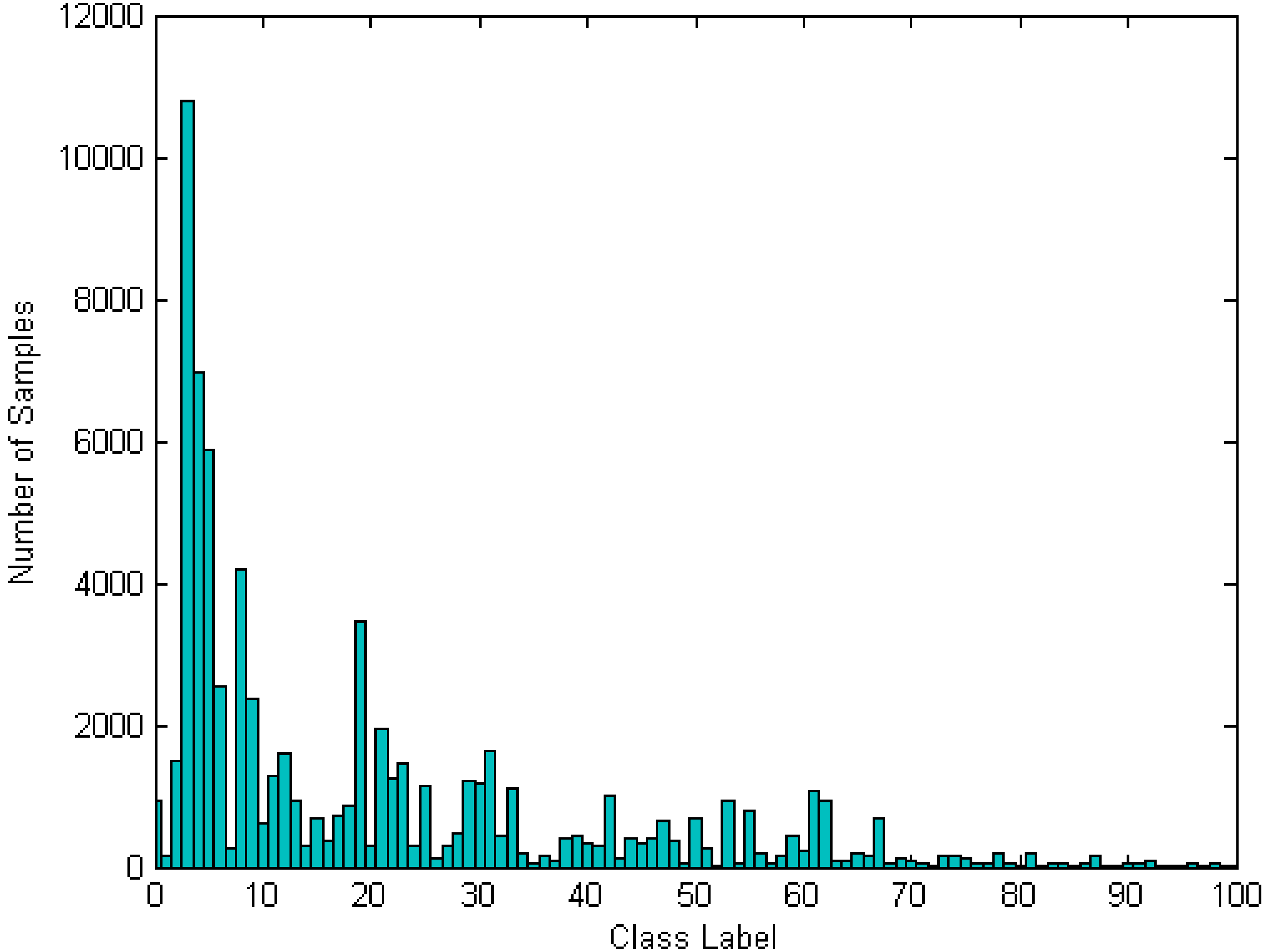


		Section	Class	Subclass	Group	Subgroup
No. Classes		8	120	630	7002	57913
No. Labels	Max	6	16	24	35	91
	Avg	1.3	1.5	1.7	2.2	2.7
No. Data	Max	857587	354104	176973	97008	23944
	Min	50540	38	1	1	1

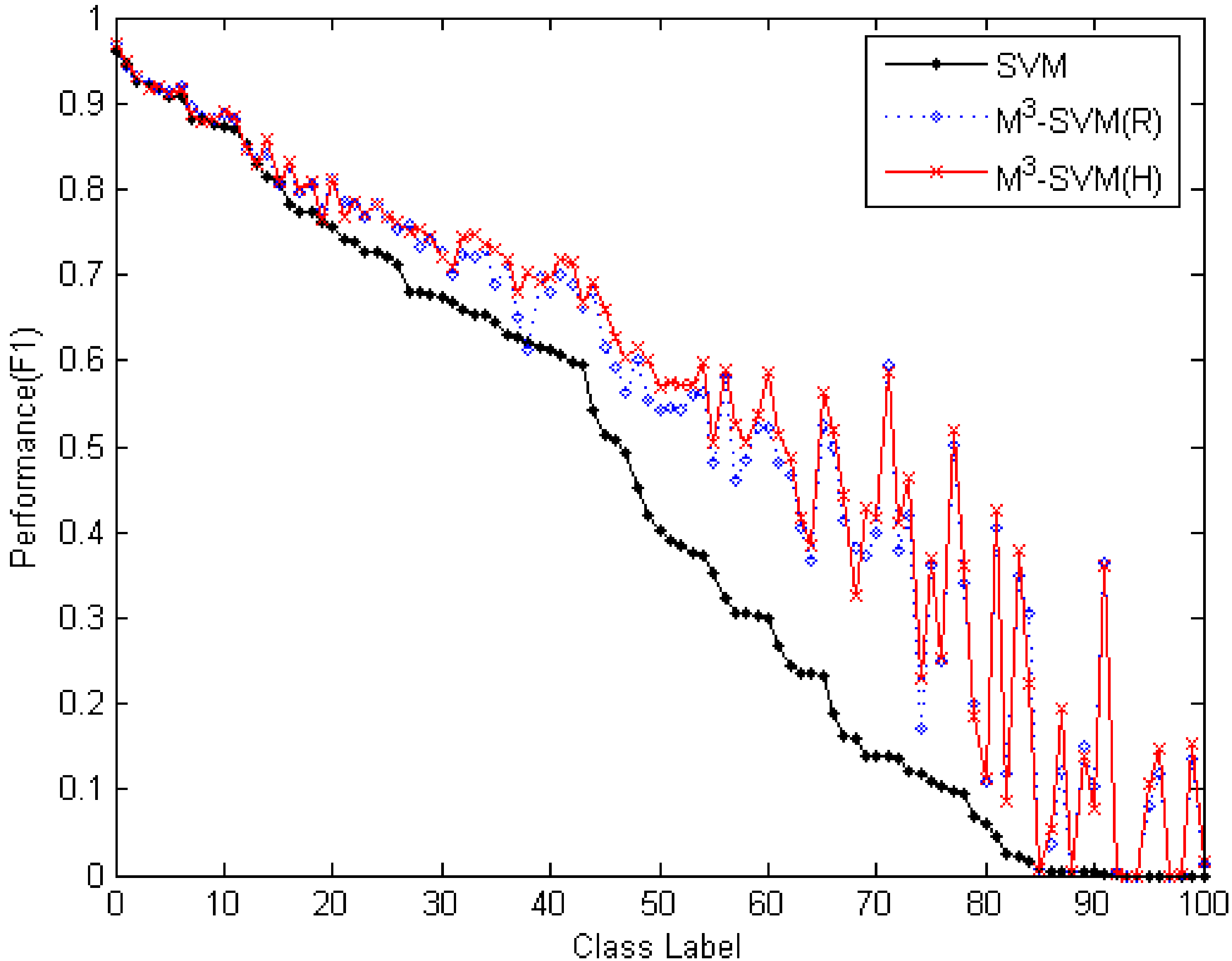
Text Categorization

(F. Y. Liu & B. L. Lu, 2005)

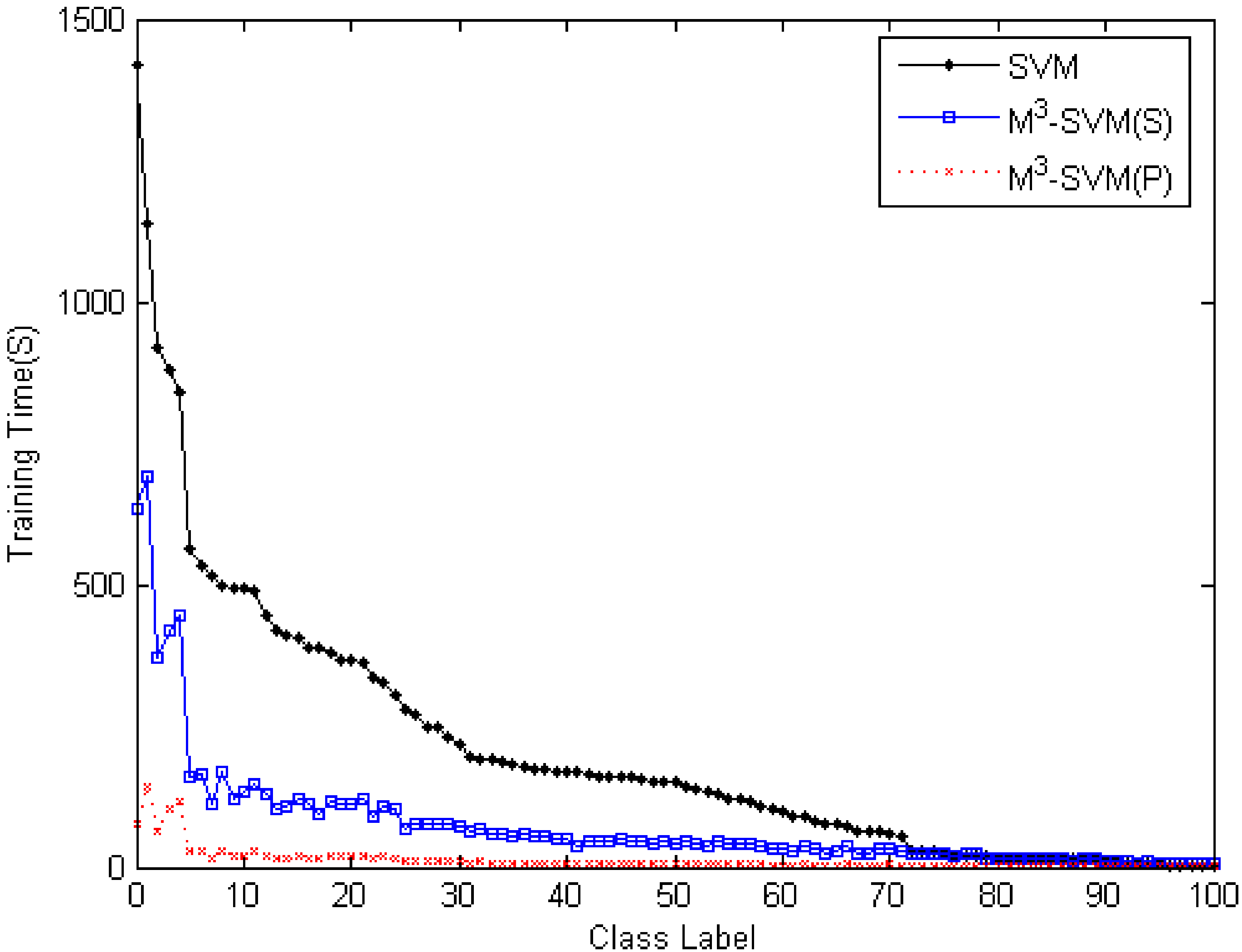
RCV1-V2: Data Distribution



RCV1–V2: Generalization Performance



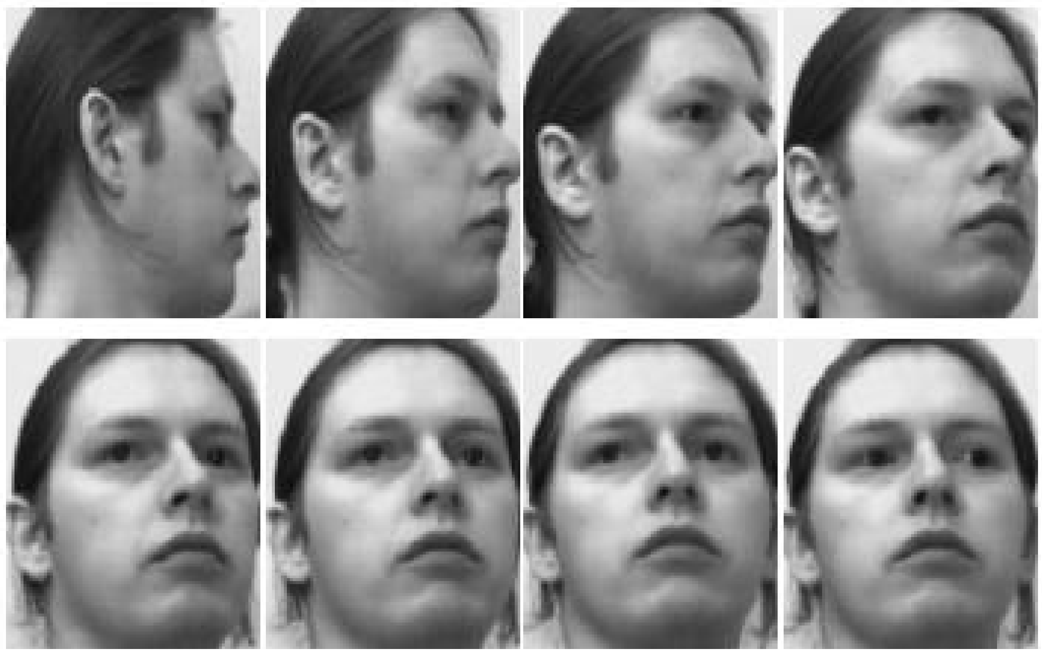
RCV1-V2: Training Times

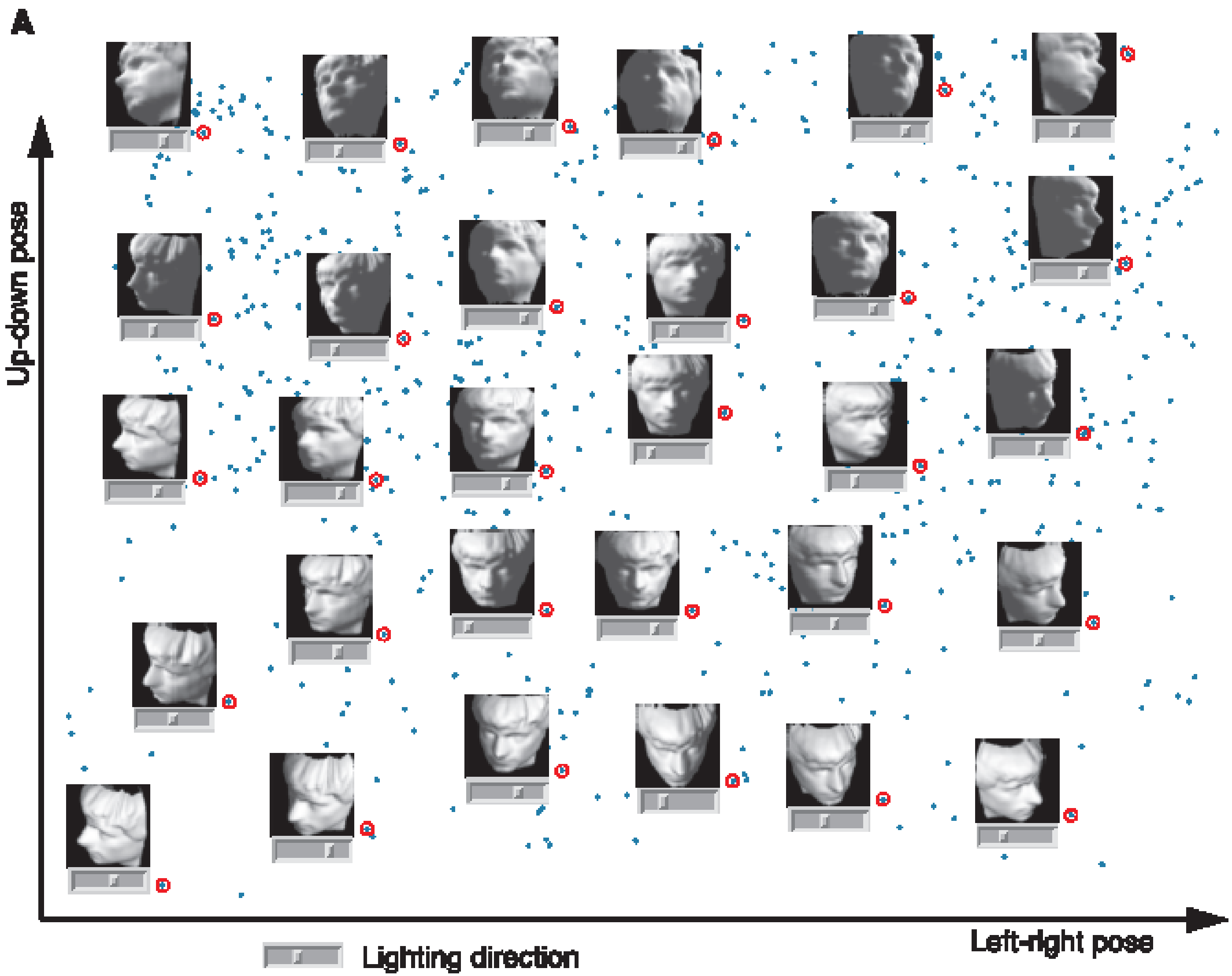


Gender Recognition

(H. C. Lian & B. L. Lu, 2005)

Multi-view Face Recognition

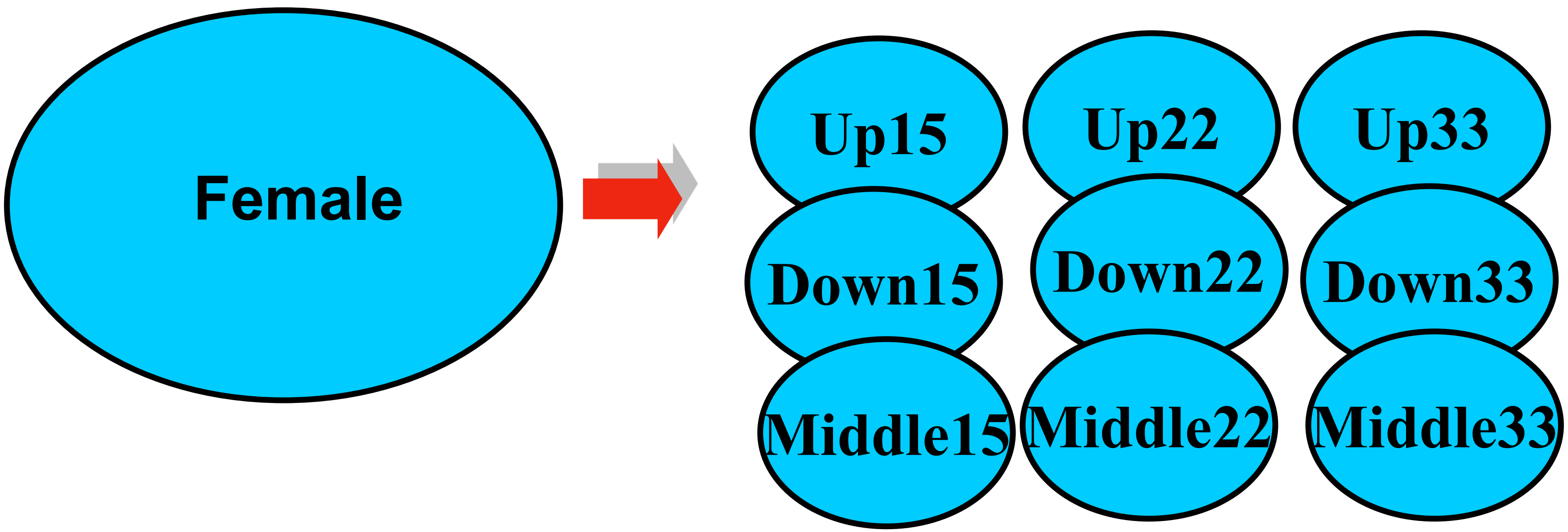


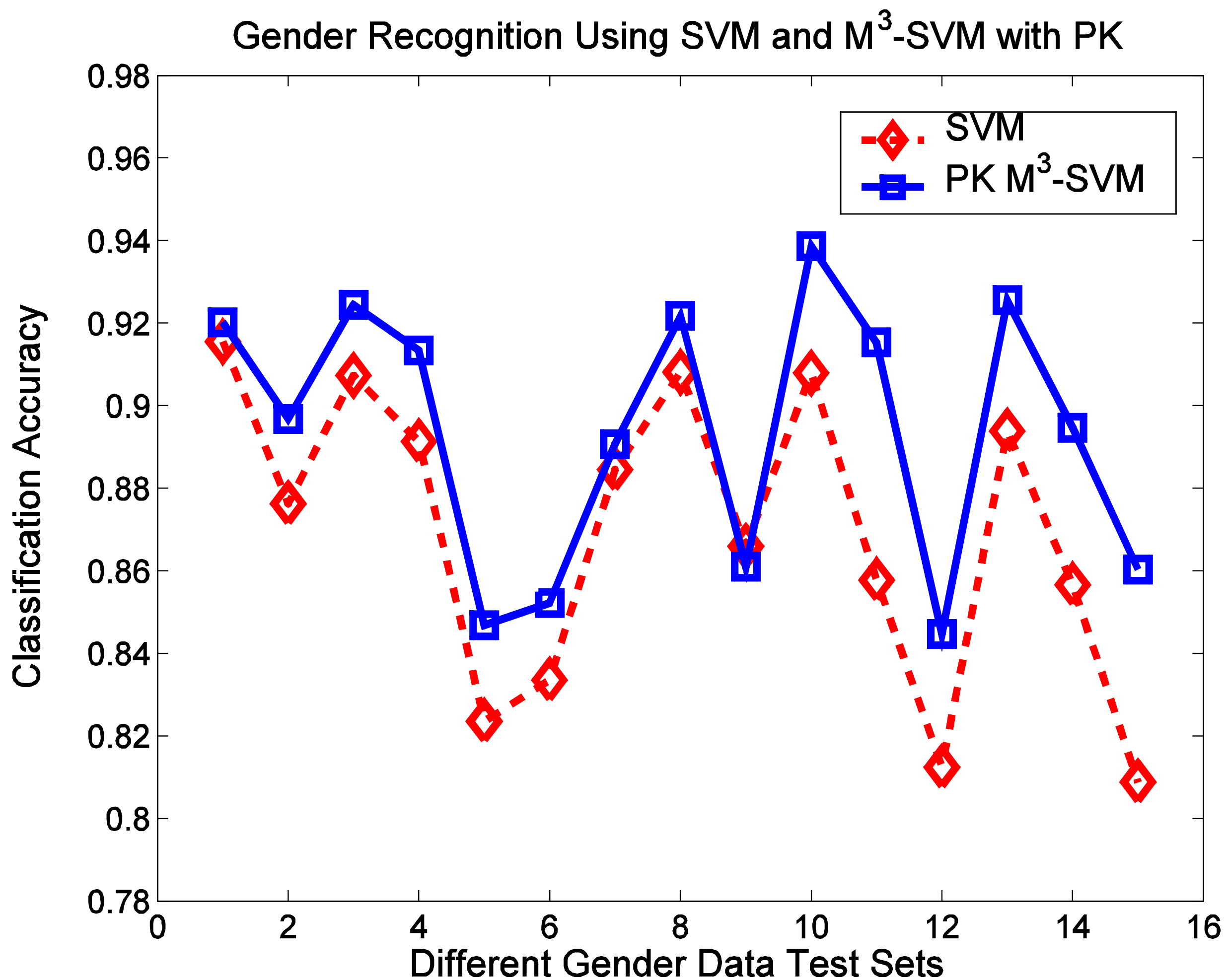


A
Up-down pose
Lighting direction
Left-right pose

Task Decomposition

View information is used for task decomposition



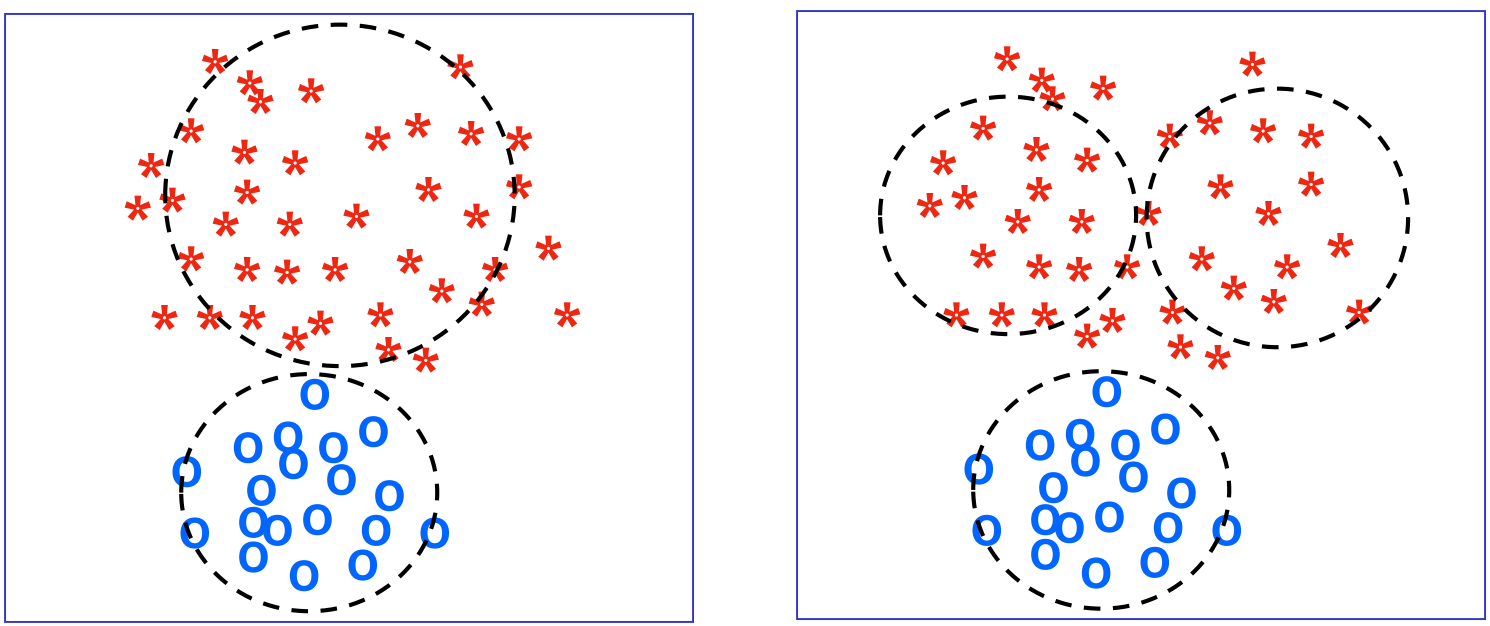

Gender Recognition Using SVM and M3-SVM with PK
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
0.78
Classification Accuracy
SVM
PK M3-SVM
0
2
4
6
8
10
12
14
16
Different Gender Data Test Sets

Gender Recognition Using a SVM With Equal Clustering

(J. Luo & B. L. Lu, 2005)

Equal Clustering

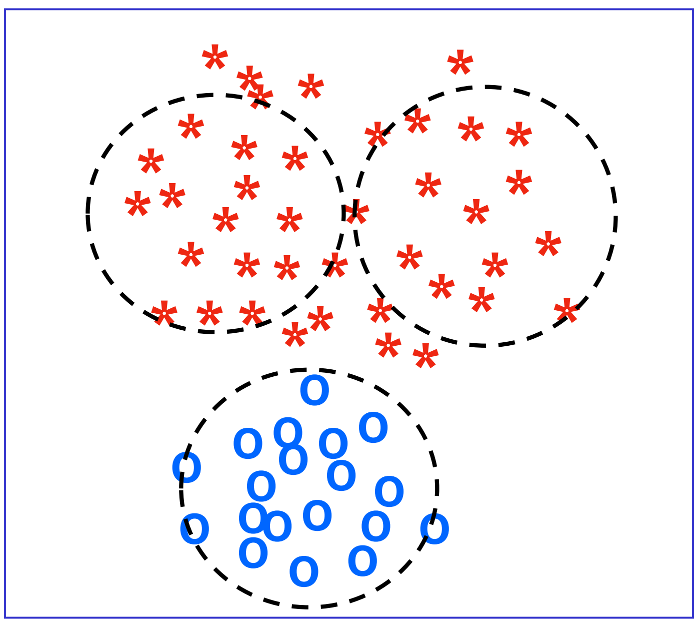
- Based on the algorithm "GeoClust" (Choudhury, Nair and Keane, 2002)
- To generate spatially localized clusters that contain (nearly) equal number of samples to keep load balance.



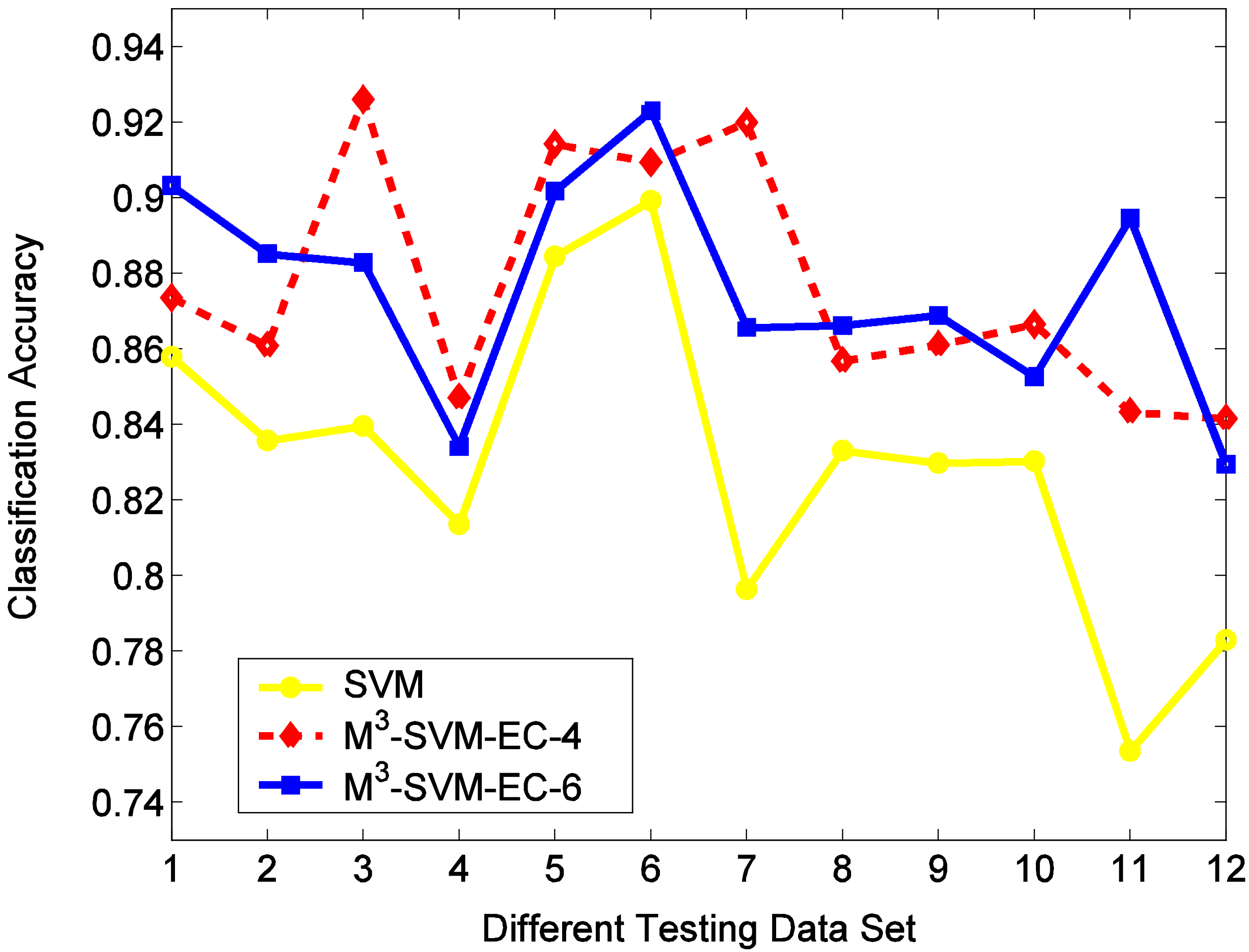
Basic Idea of Equal Clustering

- Solve an unconstrained nonlinear programming problem as follows:

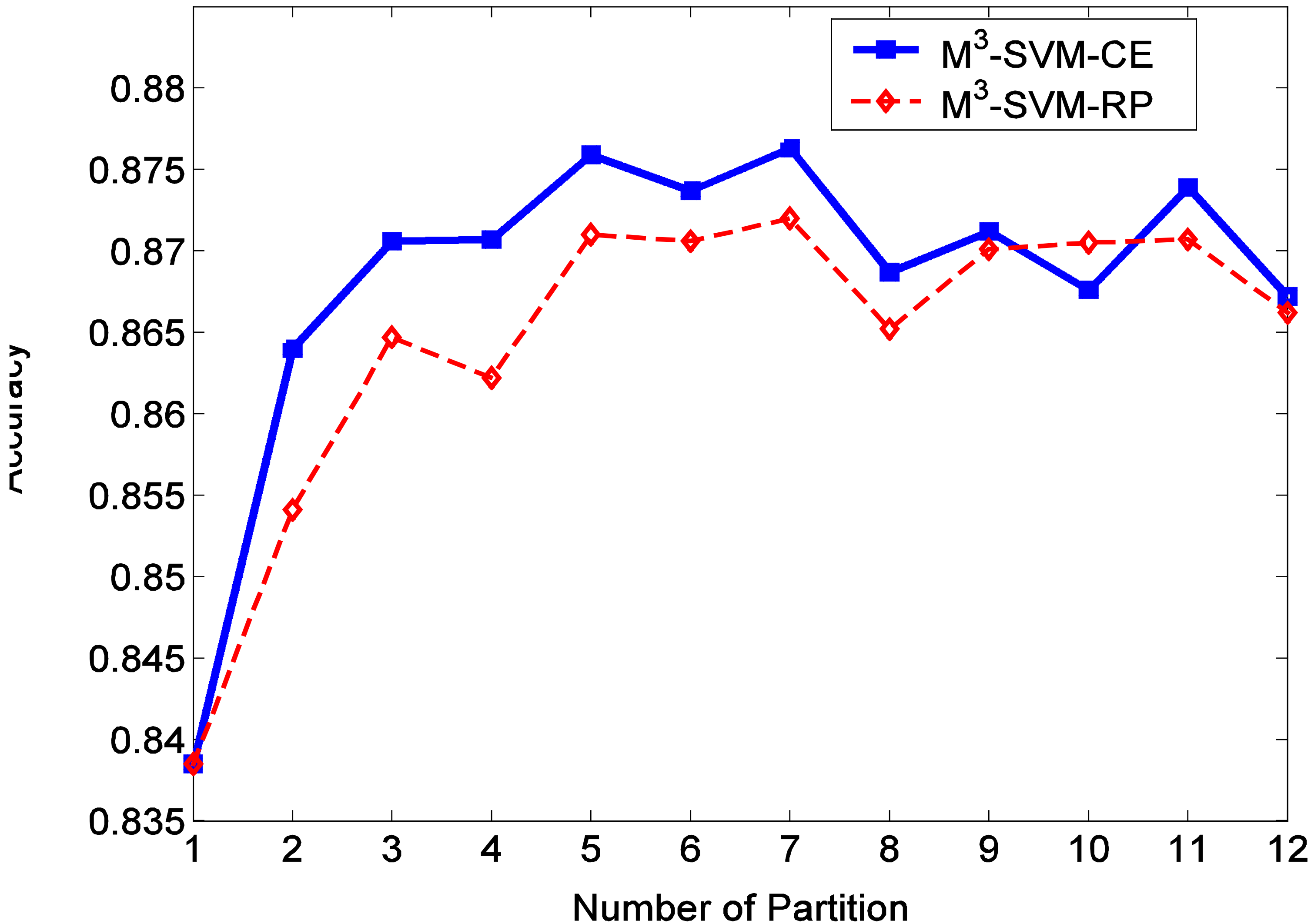
$$\underset{c_1, c_2, \ldots, c_m}{\text{Minimize}} \; h = \max_{i=1}^{m} \left| W_i - \overline{W} \right|$$



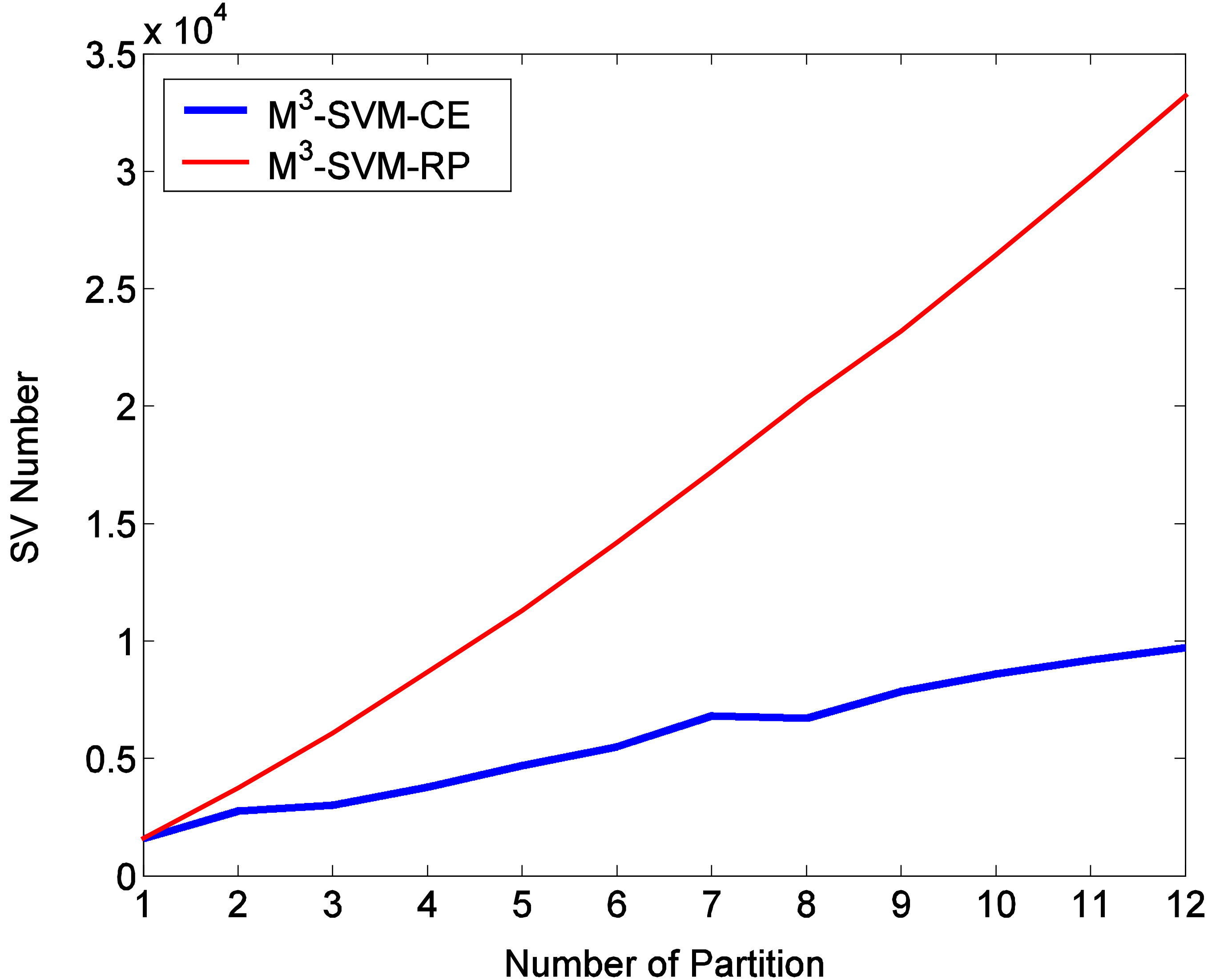
Gender Estimation on Peal dataset



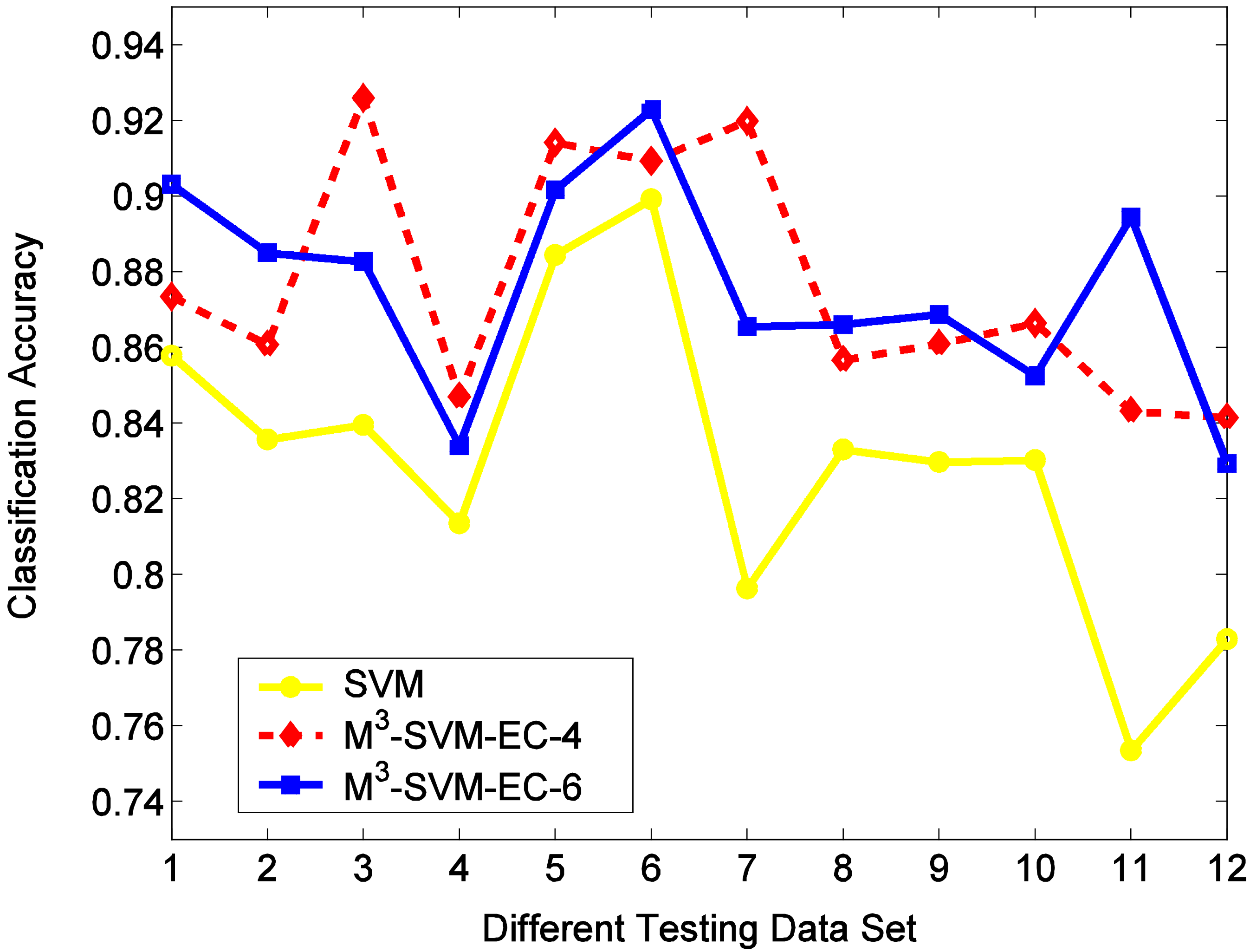
Comparison of Generalization Accuracy



Comparison of Number of SVs



Results of Gender Recognition



SVM software packages

- LibSVM
  - Http://www.csie.ntu.edu.tw/~cjlin/libsvm/
  - Chih-Chung Chang and Chin-Jen Lin
- SVM^{light}
  - http://svmlight.joachims.org/
  - Thorsten Joachims

LibSVM

- Various language versions
  - C++, C#, java, MatLab, etc.
  - Recommend C++ version
- The source code is readable
- The interface is clear

LibSVM

- Two executable files
  - Train.exe
    - Compiled by svm.cpp, svm.h and svm-train.c
  - Test.exe
    - Complied by svm.cpp, svm.h and svm-predict.c

LibSVM

- Description of svmtrain.exe
  - “one versus one” is implemented a solution to multi-class problem
  - Several frequently used parameters
    - -s : svm type (0 for classification)
    - -t : kernel type (2 for RBF kernel)
    - -g : gamma value
    - -c : panelized cost
    - e. g.,

    svmtrain -s 0 -t 2 -g 0.5 –c 2 train_file model_file

LibSVM

- Description of svmpredict.exe
  - e. g.,
    svmpredict test_file model_file result_file

LibSVM

- If you want to directly modify the source code and do your homework…
  - The source code has several interface functions. You can write codes to call these functions.
    - svm_train(), svm_predict_values(), svm_save_model(),…
  - Not recommended unless you have strong understanding to SVMs

Outline of Lecture Four

- Extreme Learning Machine (ELM)
- Vapnik-Chervonenkis (VC) dimension
- Support Vector Machine
- Performance evaluation Index

Performance evaluation Index

混淆矩阵(Confusion Matrix)

	预测正类	预测负类
正类	TP	FN
负类	FP	TN

TP (True Positive)：样本属于正类，预测结果为正类

FP (False Positive)：样本属于负类，预测结果为正类

FN (False Negative)：样本属于正类，预测结果为负类

TN (True Negative)：样本属于负类，预测结果为负类

常用的评价指标：精度

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

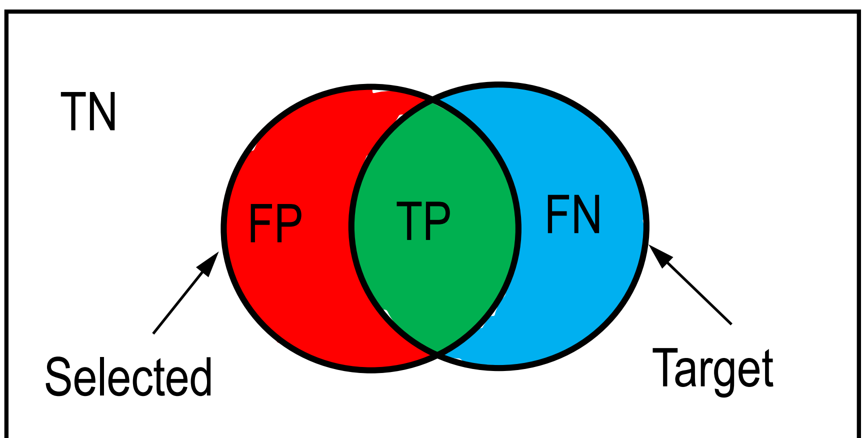
使用精度度量分类器的局限性

- 考虑一个2类问题：
  - 第一类有9990个测试样本
  - 第二类只有10个测试样本
- 如果分类器把全部测试样本都分为第一类，其精度为9990/10000=99%！
- 显然，这里凸显出精度的局限性。因为，它未能全面地衡量分类器对第二类的分类性能。

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

精确率、召回率、F1度量

- 真正（True positive, TP); 假负（False negative, FN）
- 假正（False positive, FP）; 真负（True negative, TN）
- 真正率（True positive rate, TPR):TPR=TP/(TP+FN)
- 假正率（False positive rate, FPR)：FPR=FP/(FP+TN)
- 精确率（Precision)：p=TP/(TP+FP)
- 召回率（Recall)：r=TP/(TP+FN)
- F_1度量：F1=2r*p/(r+p)

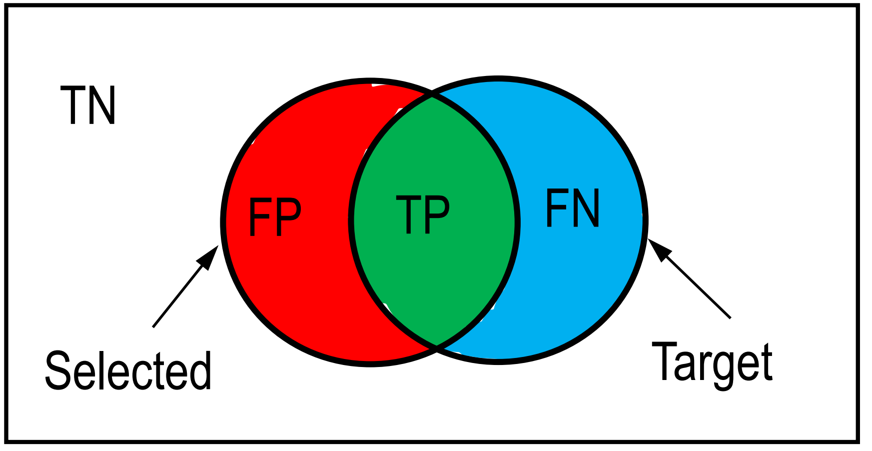


	预测正类	预测负类
正类	TP	FN
负类	FP	TN

Macro、Mirco-p、r、F1

p: 查准率
r: 查全率

Macro-p
Macro-r
Macro-F1
先计算p、r，再求平均

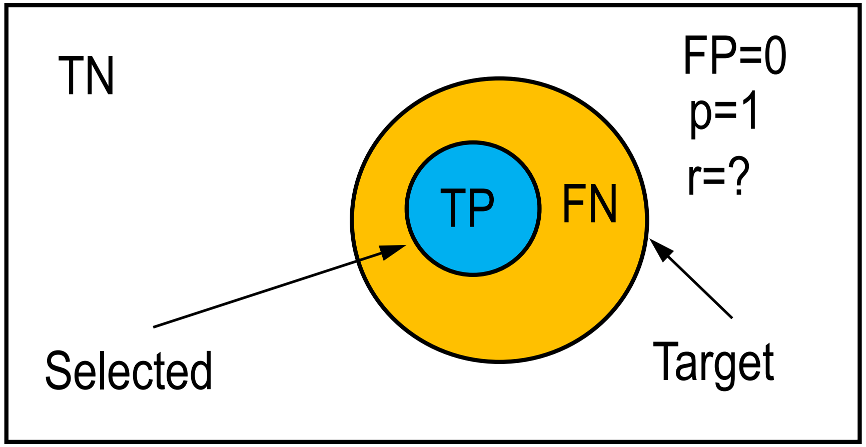


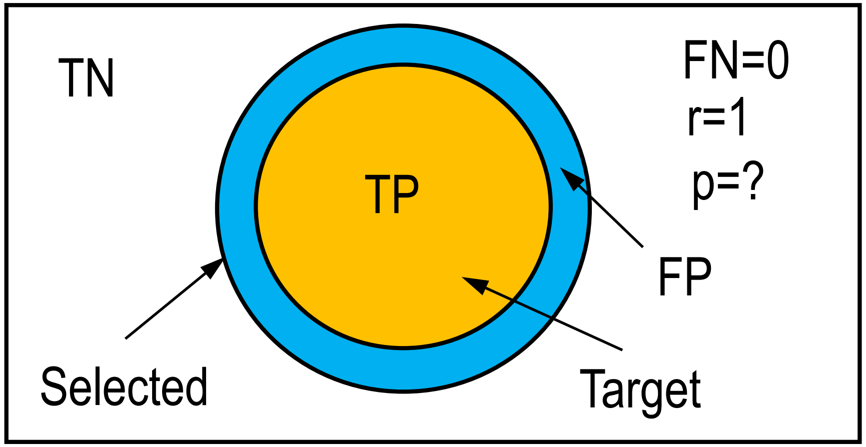
Micro-p
Micro-r
Micro-F1
先求TP、FP、TN、FN
平均，再求p、r、F1

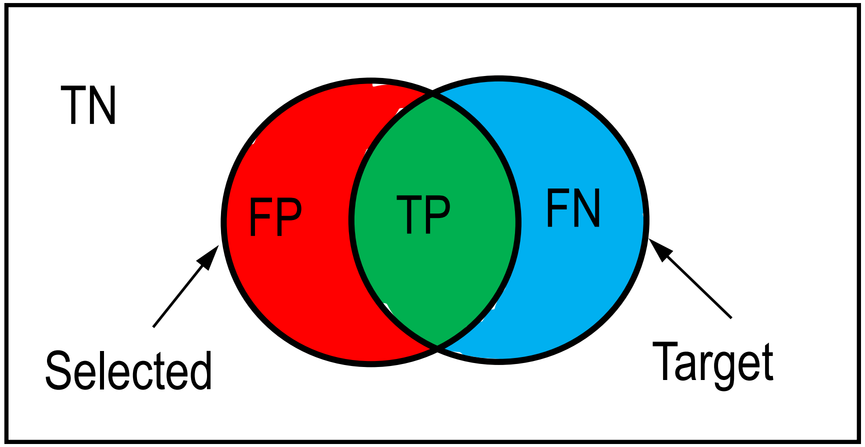
精确率、召回率、F1度量

- 精确率（Precision）：p=TP/(TP+FP)
- 召回率（Recall）：r=TP/(TP+FN)
- F1度量：F1=2*r*p/(r+p)

	预测正类	预测负类
正类	TP	FN
负类	FP	TN





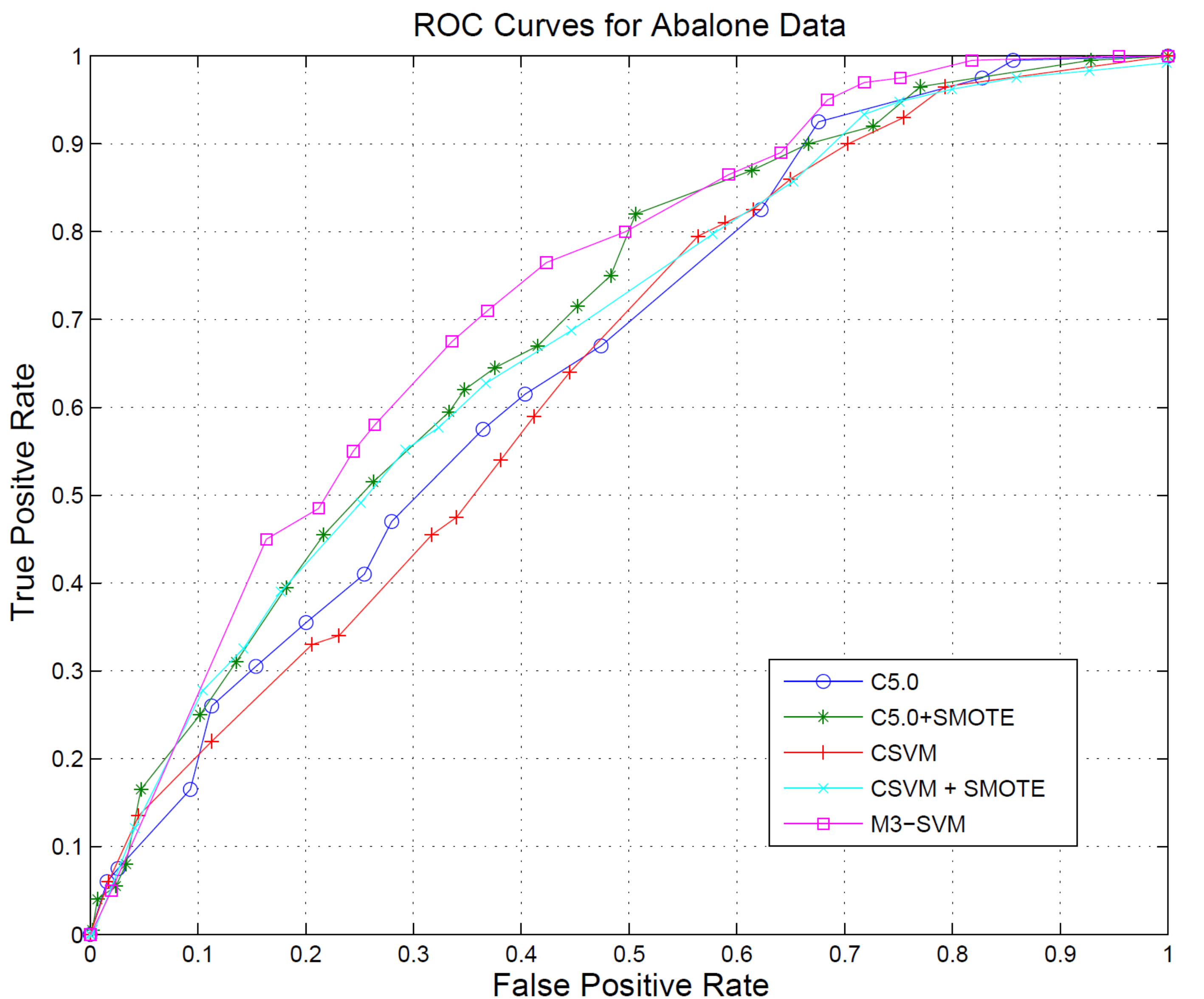


ROC (Receiver Operating Characteristic) 曲线

ROC (Receiver Operating Characteristic)曲线

- Developed in 1950s for signal detection theory to analyze noisy signals
  - Characterize the trade-off between positive hits and false alarms
- ROC curve plots TPR (on the y-axis) against FPR (on the x-axis)
- Performance of each classifier represented as a point on the ROC curve
  - Changing the threshold of algorithm, sample distribution or cost matrix changes the location of the point
- AUC：**Area under ROC Curve**

使用ROC曲线比较 分类算法



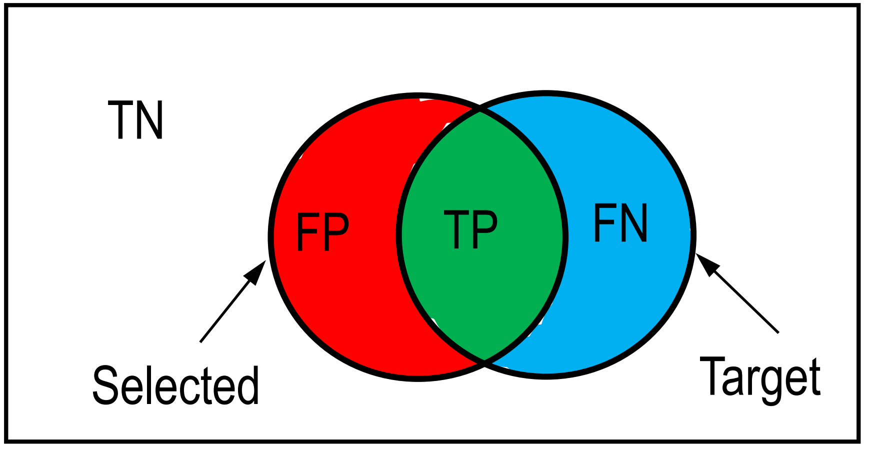
评估指标

- 真正类率（召回率）

TPR (True Positive Rate)：

$$TPR = \frac{TP}{TP + FN}$$

	预测正类	预测负类
正类	TP	FN
负类	FP	TN



- 假正类率

FPR (False Positive Rate)：

$$FPR = \frac{FP}{FP + TN}$$

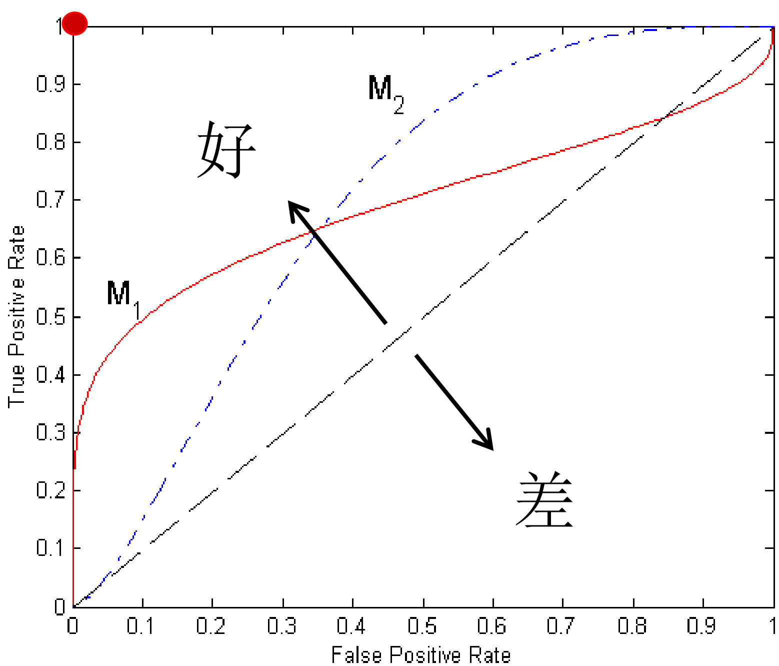
- **TPR越大，分类效果越好。而FPR越大，分类效果越差。**

ROC 曲线的几个关键点

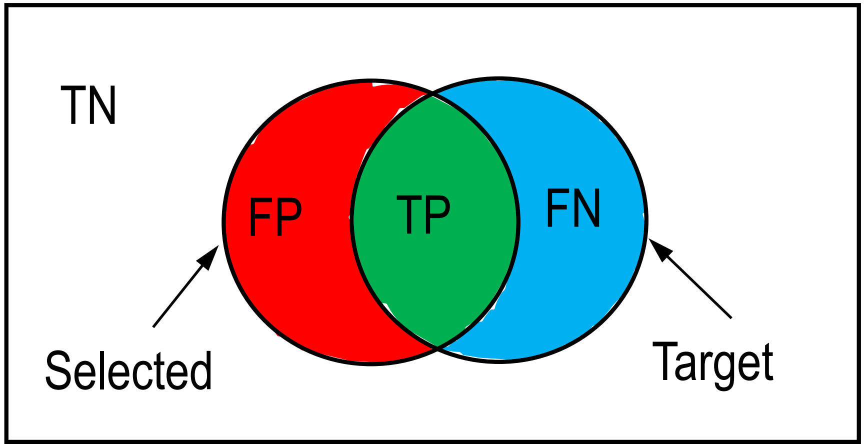
- (TPR=0, FPR=0): 把每个输入都预测为负类;
- (TPR=1, FPR=1): 把每个输入都预测为正类
- (TPR=1, FPR=0): 理想模型
- 主对角线: 随机猜测

（随机猜测是指以固定的概率 *p* 把输入分为正类）

	预测正类	预测负类
正类	TP	FN
负类	FP	TN



TPR=TP/(TP+FN)
FPR=FP/(FP+TN)



第 4卷第 2期 智 能 系 统 学 报 Vol 4 № 2
2009年 4月 CAA I Transactions on Intelligent Systems Apr 2009

不平衡分类问题研究综述

叶志飞[1], 文益民[2],吕宝粮[1,3]
(1. 上海交通大学 计算机科学与工程系,上海 200240; 2. 湖南工业职业技术学院 信息工程系,湖南 长沙 410208;
3. 上海交通大学 智能计算与智能系统教育部微软重点实验室,上海 200240)

摘 要:实际的分类问题往往都是不平衡分类问题,采用传统的分类方法,难以得到满意的分类效果. 为此,十多年来,人们相继提出了各种解决方案. 对国内外不平衡分类问题的研究做了比较详细地综述,讨论了数据不平衡性引发的问题,介绍了目前几种主要的解决方案. 通过仿真实验,比较了具有代表性的重采样法、代价敏感学习、训练集划分以及分类器集成在 3个实际的不平衡数据集上的分类性能,发现训练集划分和分类器集成方法能较好地处理不平衡数据集,给出了针对不平衡分类问题的分类器评测指标和将来的工作.

关键词:机器学习;不平衡模式分类;重采样;代价敏感学习;训练集划分;分类器集成;分类器性能评测

使用AUC指标比较分类算法

表 6.2　5种方法在三个数据上的结果

数据	方法	TP%	TN%	B-ACC%	AUC
Rooftop	C5.0	78.5	80.2	79.9	87.43
	CSVM	80.3	**81.8**	81.1	87.98
	C5.0 + SMOTE	79.9	80.1	80.0	88.22
	CSVM + SMOTE	81.3	80.4	80.9	87.87
	M3-SVM	**81.6**	81.4	**81.5**	**89.28**
Park	C5.0	82.6	85.8	84.2	90.39
	CSVM	84.9	85.5	85.2	93.93
	C5.0 + SMOTE	84.3	83.8	84.2	90.96
	CSVM + SMOTE	85.4	85.1	85.3	94.10
	M3-SVM	**87.2**	**87.7**	**87.5**	**94.54**
Abalone	C5.0	61.5	59.6	60.6	66.84
	CSVM	59.0	58.8	58.9	64.25
	C5.0 + SMOTE	64.5	62.4	63.5	69.53
	CSVM +SMOTE	62.7	63.3	63.0	68.00
	M3-SVM	**67.5**	**66.4**	**67.0**	**72.67**

二类分类器预测过程

- 一般二类分类器在预测时会计算一个评估函数

  $$f(x;w)$$

  其中，x为待预测样本的特征向量，w为已训练的参数。对于线性分类器有 $f(x;w)=x\cdot w$ 。

- 预测结果如下输出：
  - 若 $f(x;w)>0$ ，分类器输出正类
  - 若 $f(x;w)<0$ ，分类器输出负类

二类分类器预测过程

- 通过引入阈值 t，改变分类器预测时对正负类的倾向：
  - 若 $f(x;w) > t$，分类器输出正类。
  - 若 $f(x;w) < t$，分类器输出负类。
- 阈值t增大，分类器预测结果偏向负类。
- 阈值t减小，分类器预测结果偏向正类。

ROC曲线的绘制

- ROC曲线以FPR为横轴，TPR为纵轴。
- 设置不同的阈值t，分类器预测结果有不同的FPR值和TPR值。

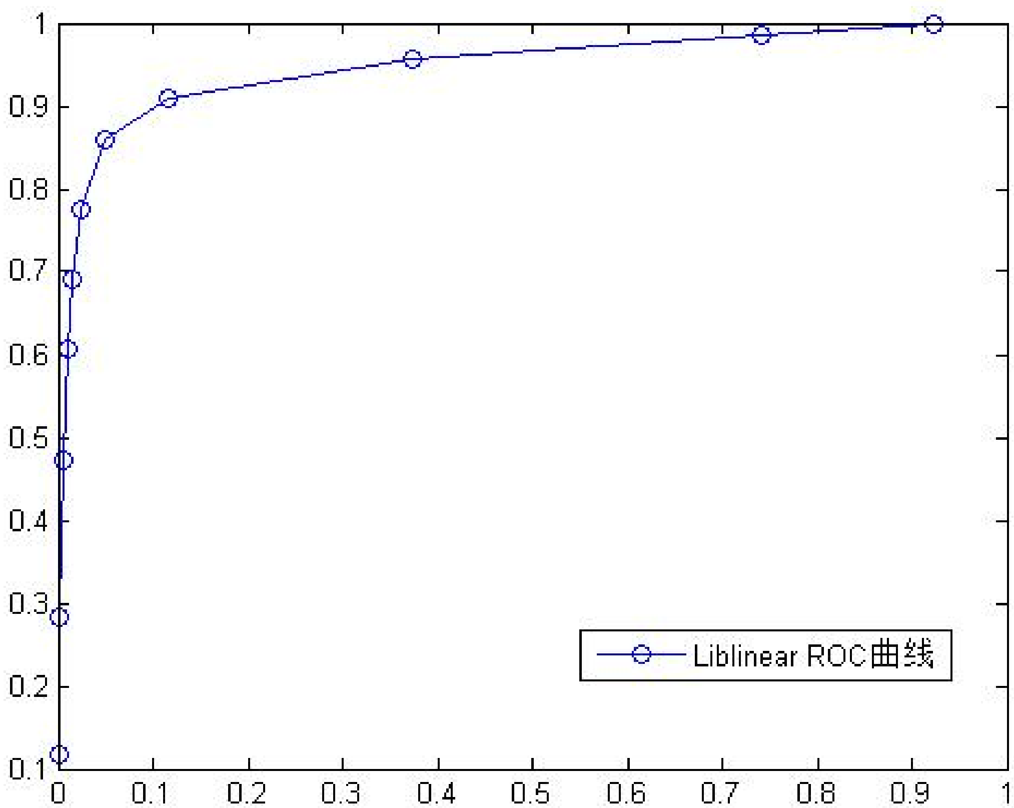
例：Liblinear的ROC曲线

- 分别以-8,-4,-2,-1,-0.5,0,0.5,1,2,4,8为阈值，使用Liblinear进行预测，结果如下表：

t	-8	-4	-2	-1	-0.5	0	0.5	1	2	4	8
FPR	0.001	0.002	0.005	0.010	0.014	0.025	0.049	0.115	0.373	0.741	0.924
TPR	0.117	0.282	0.472	0.606	0.691	0.774	0.859	0.910	0.956	0.986	0.998

例：Liblinear的ROC曲线

- 使用画图工具将表格中的数据绘成图片。如下图：



ROC曲线与分类效果

- ROC曲线下方面积越大，分类效果越好。下图是**Liblinear的ROC曲线与M3-Liblinear的ROC曲线。**