

Support Vector Machine Tutorial

Wu, Shih-Hung (Ph.D)

Dept. of CSIE, CYUT

References

- Book
 - Duda et al. “pattern classification”, Ch5
- Slides
 - Moore, Andrew (CMU)
 - <http://www.cs.cmu.edu/~awm/tutorials>
 - Lin, Chih-Jen (NTU)
 - <http://www.csie.ntu.edu.tw/~cjlin/talks.html>

5.11 Support Vector Machines

- Popular, easy-to-use, available
- Support Vector
- Data is mapped to a high dimension
- SVM training
- Example 2
 - SVM for the XOR Problem

Optimal hyperplane

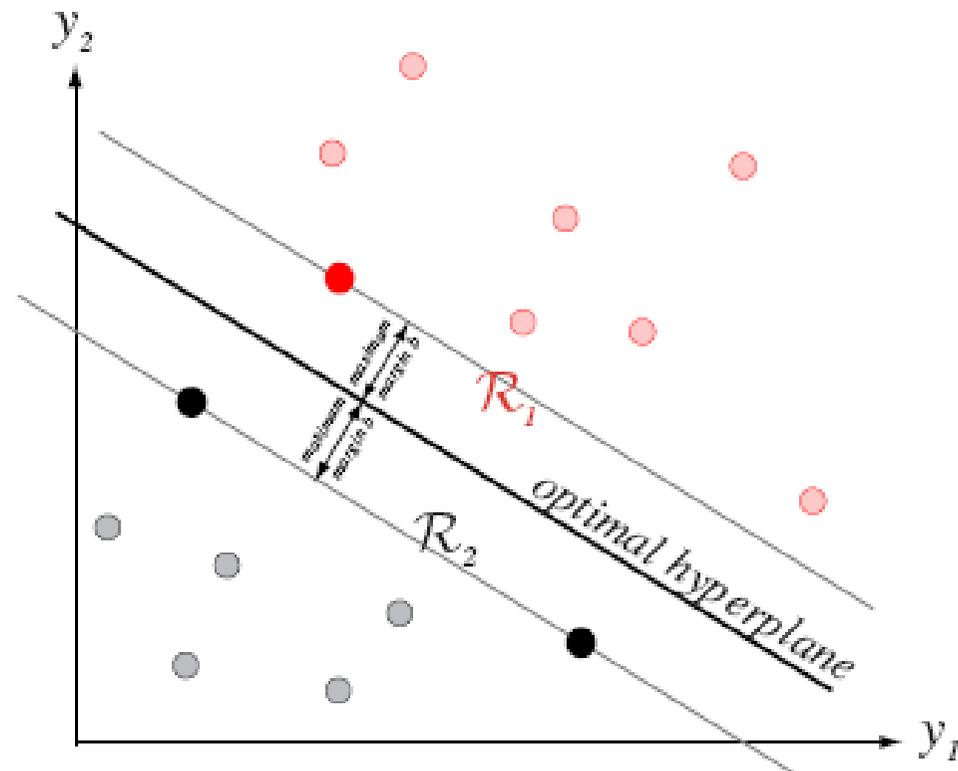
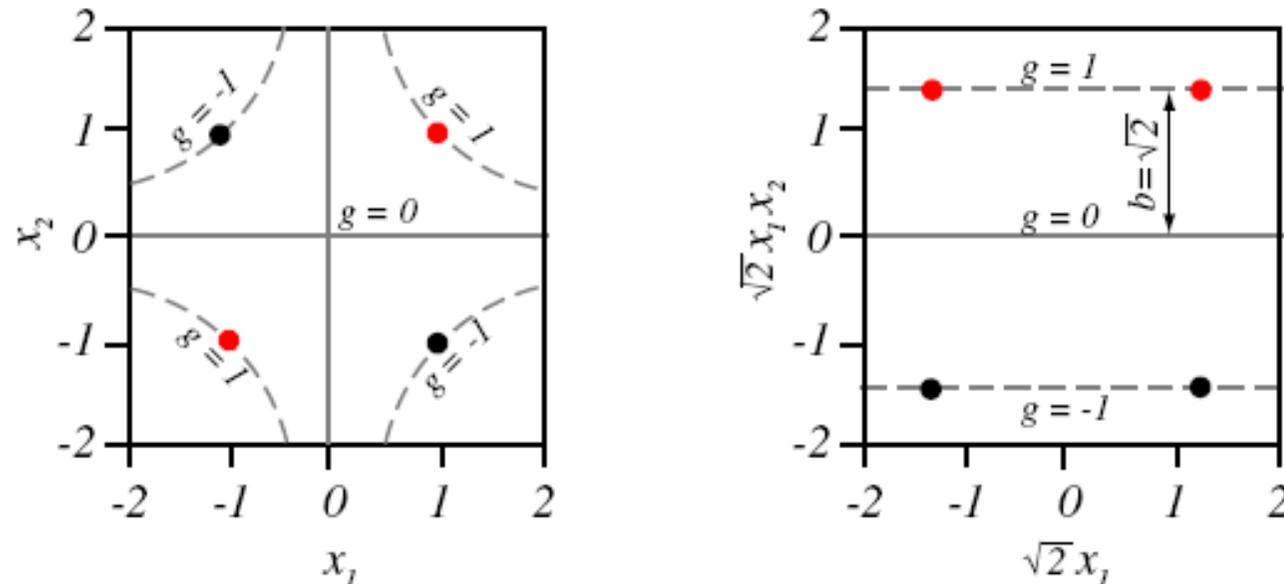


FIGURE 5.19. Training a support vector machine consists of finding the optimal hyperplane, that is, the one with the maximum distance from the nearest training patterns. The support vectors are those (nearest) patterns, a distance b from the hyperplane. The three support vectors are shown as solid dots. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Mapping to higher dimensional space

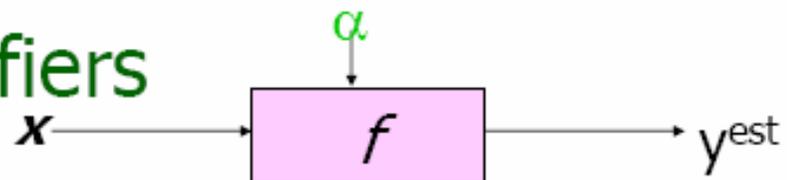


The XOR problem in the original $x_1 - x_2$ feature space is shown at the left; the two red patterns are in category ω_1 and the two black ones in ω_2 . These four training patterns \mathbf{x} are mapped to a six-dimensional space by $1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2$ and x_2^2 . In this space, the optimal hyperplane is found to be $g(x_1, x_2) = x_1x_2 = 0$ and the margin is $b = \sqrt{2}$. A two-dimensional projection of this space is shown at the right. The hyperplanes through the support vectors are $\sqrt{2}x_1x_2 = \pm 1$, and correspond to the hyperbolas $x_1x_2 = \pm 1$ in the original feature space, as shown.

SVM introduction

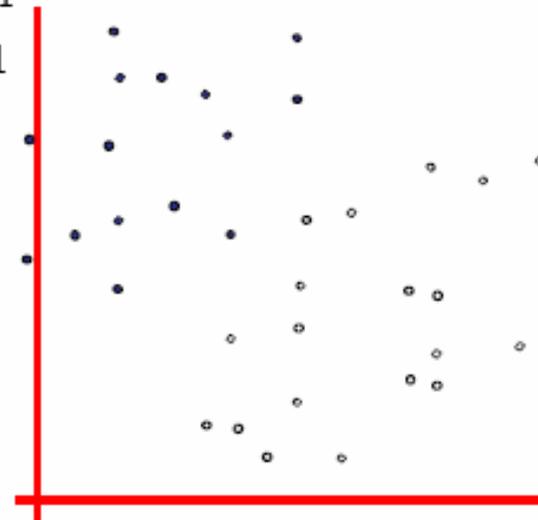
Example from Andrew Moor's
slides

Linear Classifiers



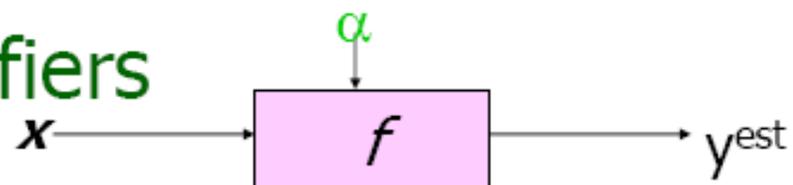
$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

- denotes +1
- denotes -1

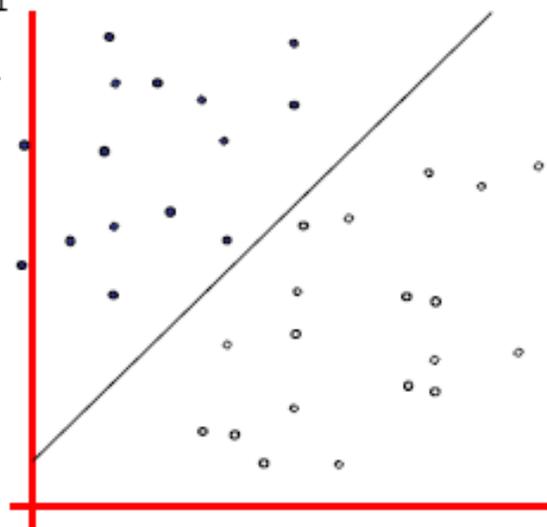


How would you classify this data?

Linear Classifiers



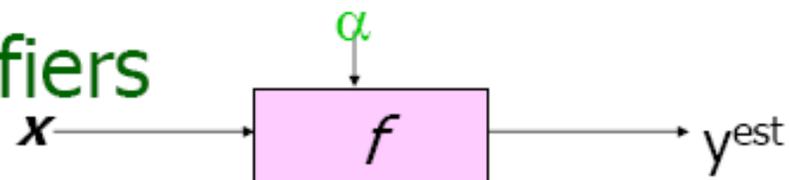
- denotes +1
- denotes -1



$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

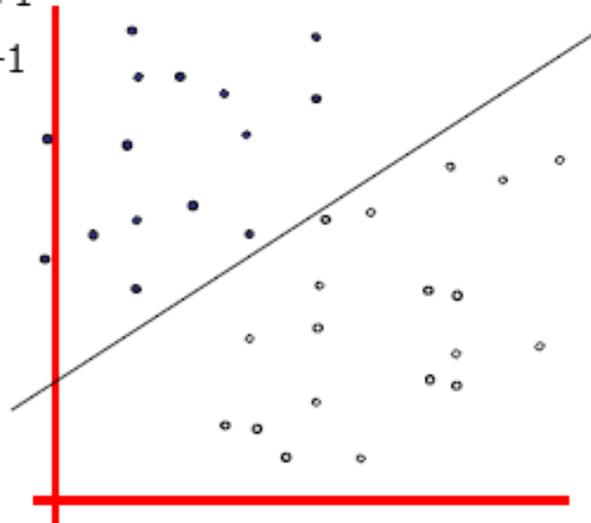
How would you classify this data?

Linear Classifiers



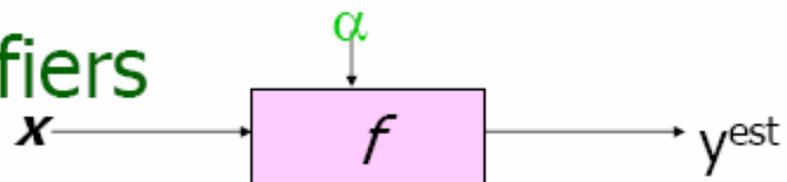
$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

- denotes +1
- denotes -1

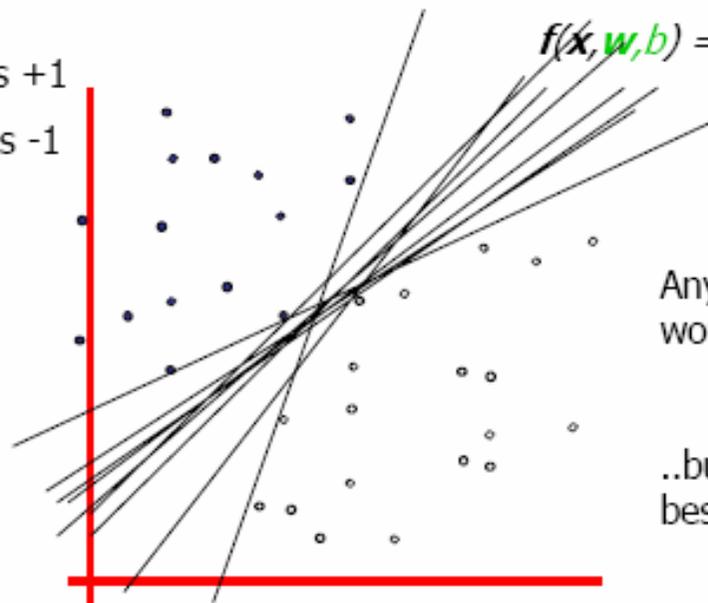


How would you classify this data?

Linear Classifiers



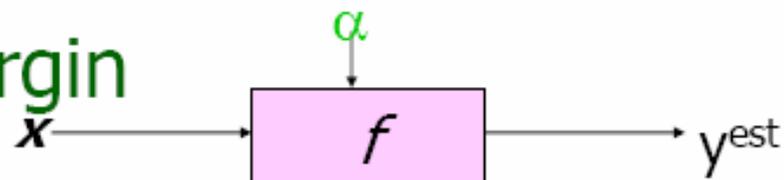
- denotes +1
- denotes -1



Any of these
would be fine..

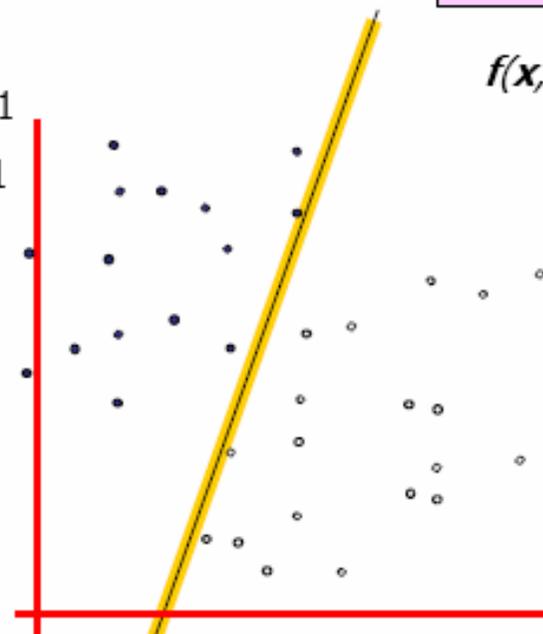
..but which is
best?

Classifier Margin



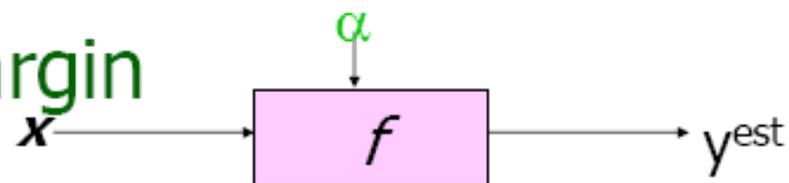
$$f(x, w, b) = \text{sign}(w \cdot x - b)$$

- denotes +1
- denotes -1

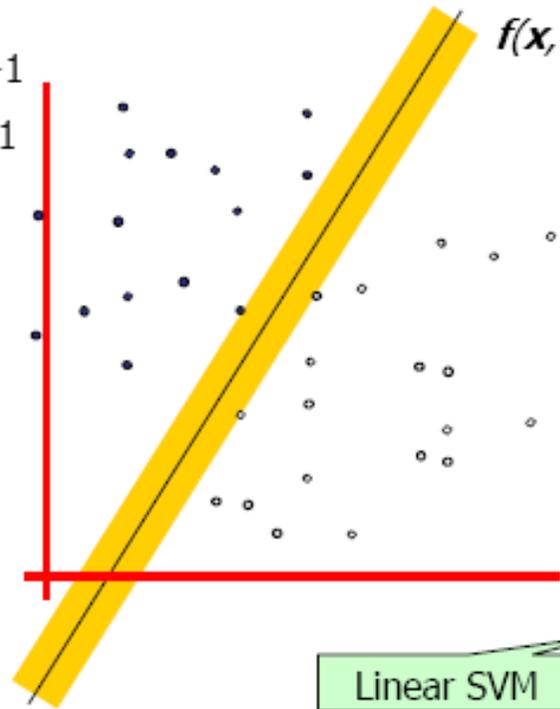


Define the **margin** of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin



- denotes +1
- denotes -1



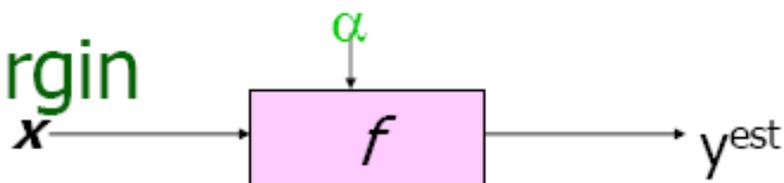
$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

The **maximum margin linear classifier** is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

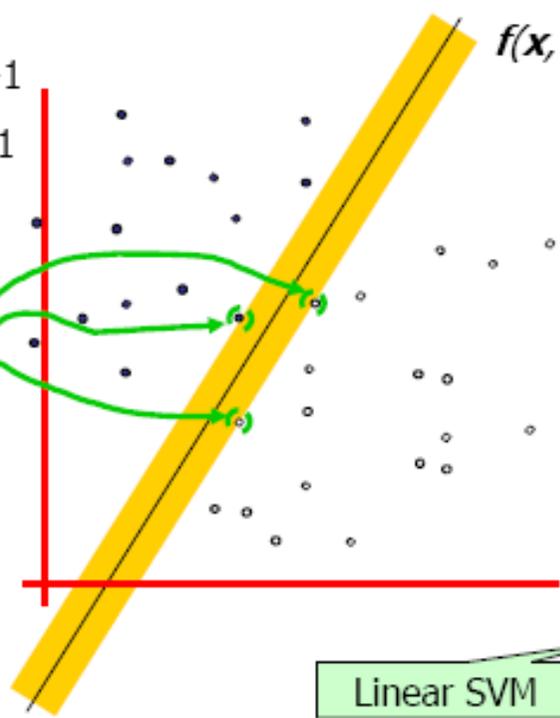
Linear SVM

Maximum Margin



- denotes +1
- denotes -1

Support Vectors
are those
datapoints that
the margin
pushes up
against



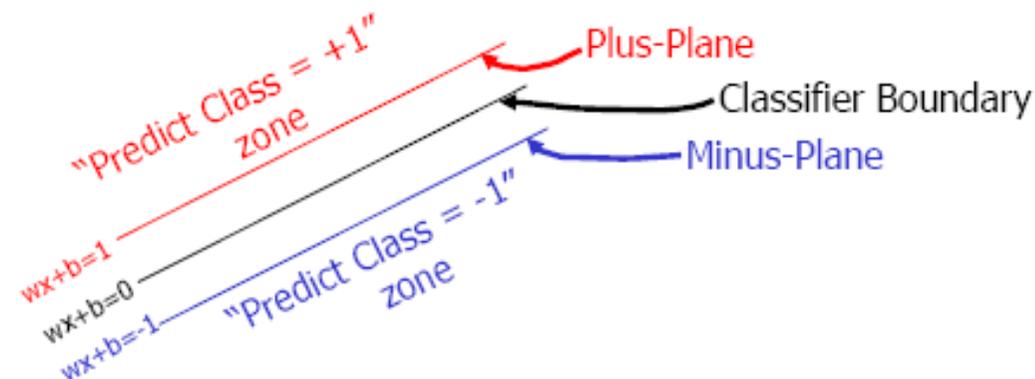
$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \cdot \mathbf{x} - b)$$

The **maximum margin linear classifier** is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

Linear SVM

Specifying a line and margin



- Plus-plane = $\{ \mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = +1 \}$
- Minus-plane = $\{ \mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = -1 \}$

Classify as.. **+1** if $\mathbf{w} \cdot \mathbf{x} + b \geq 1$

-1 if $\mathbf{w} \cdot \mathbf{x} + b \leq -1$

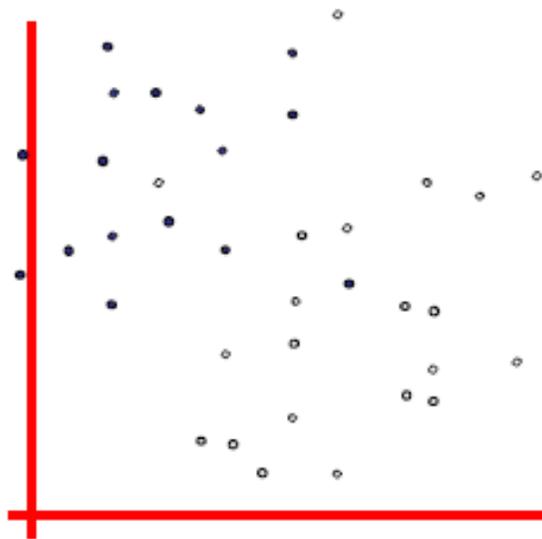
Universe
explodes if $-1 < \mathbf{w} \cdot \mathbf{x} + b < 1$

How to deal with Noisy Data?

Uh-oh!

This is going to be a problem!
What should we do?

- denotes +1
- denotes -1



Idea 1.1:

Minimize

$$w \cdot w + C (\#train\ errors)$$

Tradeoff parameter

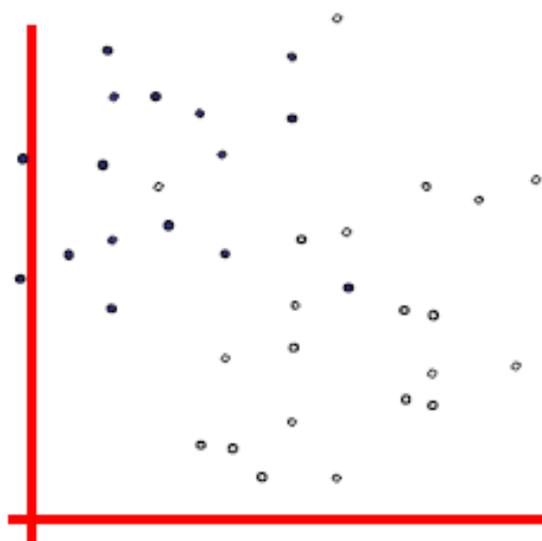
There's a serious practical problem that's about to make us reject this approach. Can you guess what it is?

Uh-oh!

This is going to be a problem!

What should we do?

- denotes +1
- denotes -1

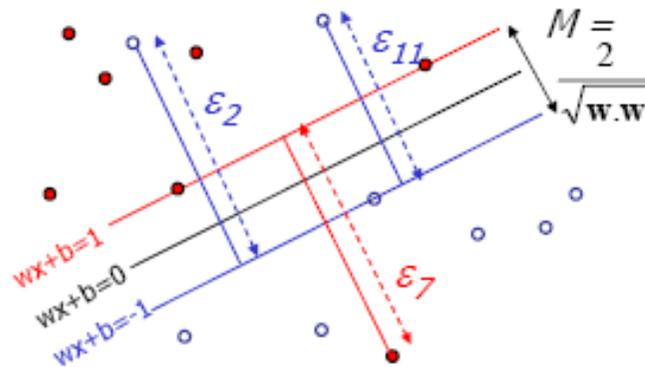


Idea 2.0:

Minimize

$w \cdot w + C$ (distance of error points to their correct place)

Learning Maximum Margin with Noise



- Given guess of \mathbf{w} , b we can
- Compute sum of distances of points to their correct zones
 - Compute the margin width
- Assume R datapoints, each (\mathbf{x}_k, y_k) where $y_k = +/- 1$

What should our quadratic optimization criterion be?

Minimize
$$\frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_{k=1}^R \varepsilon_k$$

How many constraints will we have? R

What should they be?

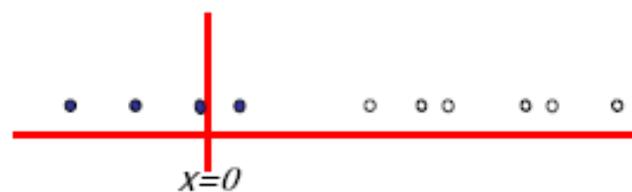
$$\mathbf{w} \cdot \mathbf{x}_k + b \geq 1 - \varepsilon_k \text{ if } y_k = 1$$

$$\mathbf{w} \cdot \mathbf{x}_k + b \leq -1 + \varepsilon_k \text{ if } y_k = -1$$

Mapping to a higher Dimensional
space

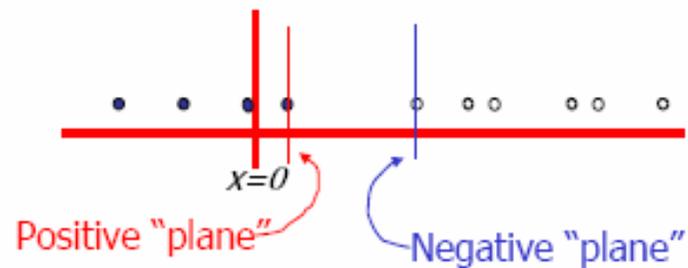
Suppose we're in 1-dimension

What would
SVMs do with
this data?



Suppose we're in 1-dimension

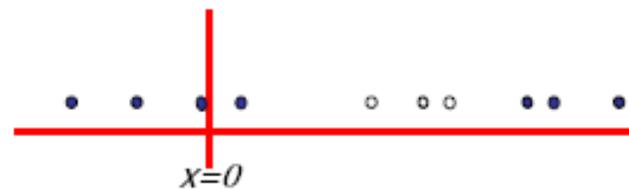
Not a big surprise



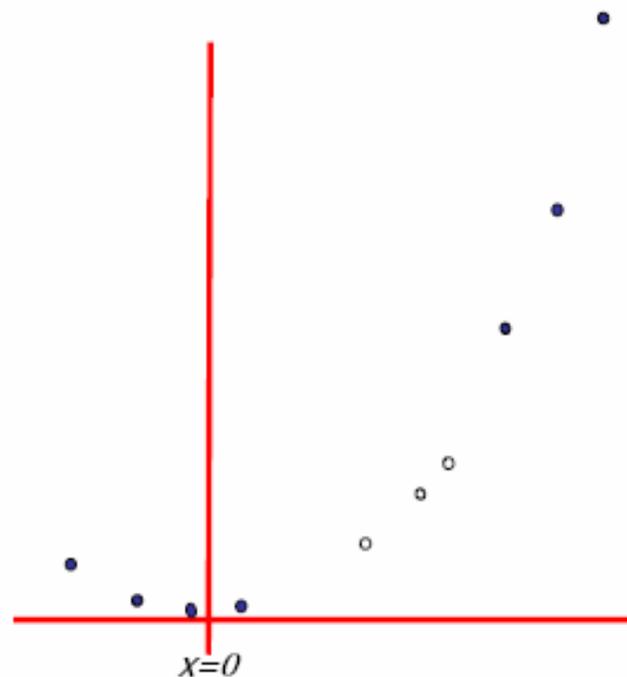
Harder 1-dimensional dataset

That's wiped the smirk off SVM's face.

What can be done about this?



Harder 1-dimensional dataset

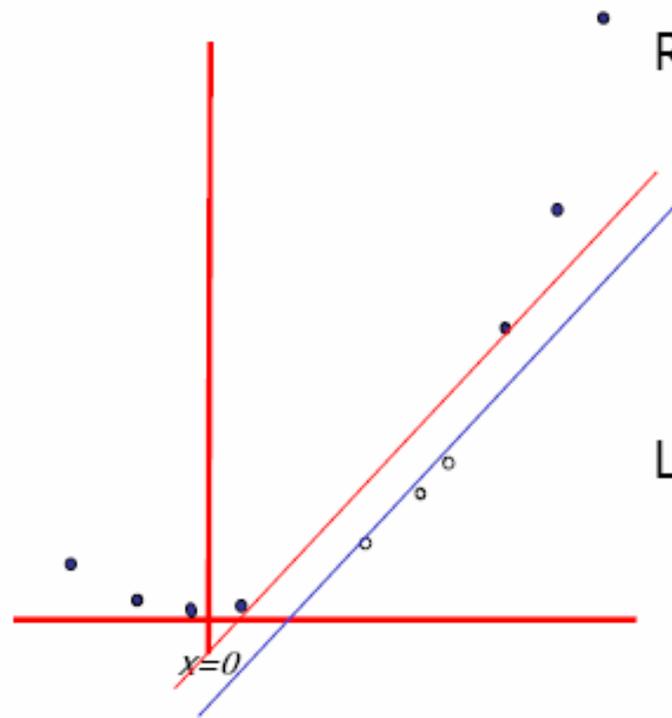


Remember how
permitting non-
linear basis
functions made
linear regression
so much nicer?

Let's permit them
here too

$$\mathbf{z}_k = (x_k, x_k^2)$$

Harder 1-dimensional dataset



Remember how
permitting non-
linear basis
functions made
linear regression
so much nicer?

Let's permit them
here too

$$\mathbf{z}_k = (x_k, x_k^2)$$

Common SVM basis functions

$\mathbf{z}_k =$ (polynomial terms of \mathbf{x}_k of degree 1 to q)

$\mathbf{z}_k =$ (radial basis functions of \mathbf{x}_k)

$$z_k[j] = \varphi_j(\mathbf{x}_k) = \text{KernelFn}\left(\frac{\|\mathbf{x}_k - \mathbf{c}_j\|}{KW}\right)$$

$\mathbf{z}_k =$ (sigmoid functions of \mathbf{x}_k)

This is sensible.

Is that the end of the story?

No...there's one more trick!

Quadratic Basis Functions

$$\Phi(\mathbf{x}) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ \sqrt{2}x_m \\ x_1^2 \\ x_2^2 \\ \vdots \\ x_m^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_1x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \sqrt{2}x_2x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \vdots \\ \sqrt{2}x_{m-1}x_m \end{pmatrix}$$

Constant Term

Linear Terms

Pure Quadratic Terms

Quadratic Cross-Terms

Number of terms (assuming m input dimensions) = $(m+2)\text{-choose-}2$
= $(m+2)(m+1)/2$
= (as near as makes no difference) $m^2/2$

You may be wondering what those $\sqrt{2}$'s are doing.

- You should be happy that they do no harm
- You'll find out why they're there soon.

SVM tool

Example from Lin, Chih-Jen's
slides

Outline

- Support vector classification
- Two practical example
- Support vector regression
- Discussion and conclusions

Data Classification

- Given training data in different classes (labels **known**)
Predict test data (labels **unknown**)
- Examples
 - Handwritten digits recognition
 - Spam filtering
 - Text classification
 - Prediction of signal peptide in human secretory proteins
- Training and testing

- Methods:
 - Nearest Neighbor
 - Neural Networks
 - Decision Tree
- Support vector machines: a new method
- Becoming more and more popular

Why Support Vector Machines

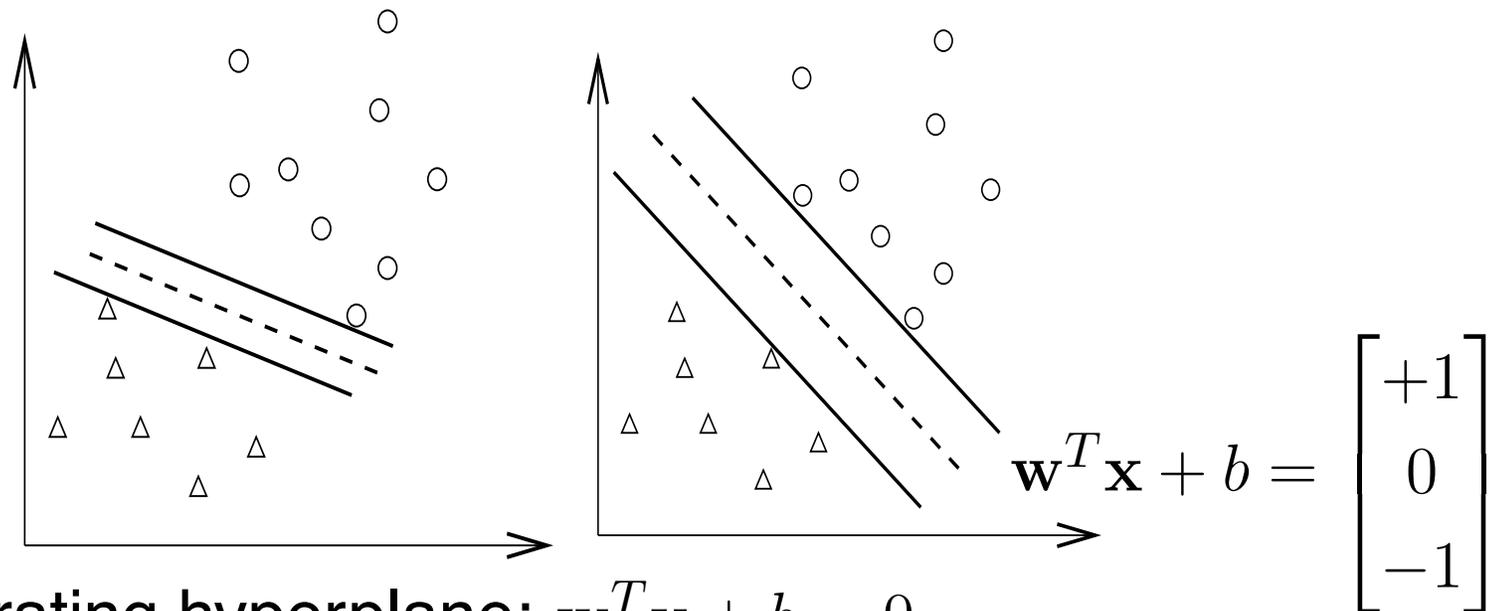
- Existing methods:
Nearest neighbor, Neural networks, decision trees.
- SVM: a new one
- In my opinion, **after careful data pre-processing**
Appropriately use NN or SVM \Rightarrow similar accuracy
- But, **users may not use them properly**
- The chance of SVM
 - Easier for users to appropriately use it
 - The ambition: replacing NN **on some applications**

Support Vector Classification

- **Training** vectors : $\mathbf{x}_i, i = 1, \dots, l$
- Consider a simple case with **two classes**:
Define a vector y

$$y_i = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ in class 1} \\ -1 & \text{if } \mathbf{x}_i \text{ in class 2,} \end{cases}$$

- A hyperplane which separates all data



● A separating hyperplane: $\mathbf{w}^T \mathbf{x} + b = 0$

$$(\mathbf{w}^T \mathbf{x}_i) + b > 0 \quad \text{if } y_i = 1$$

$$(\mathbf{w}^T \mathbf{x}_i) + b < 0 \quad \text{if } y_i = -1$$

- Decision function $f(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$, \mathbf{x} : test data
Variables: \mathbf{w} and b : Need to know coefficients of a plane

Many possible choices of \mathbf{w} and b

- Select \mathbf{w}, b with the **maximal margin**.

Maximal distance between $\mathbf{w}^T \mathbf{x} + b = \pm 1$

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

- Distance between $\mathbf{w}^T \mathbf{x} + b = 1$ and -1 :

$$2/\|\mathbf{w}\| = 2/\sqrt{\mathbf{w}^T \mathbf{w}}$$

- $\max 2/\|\mathbf{w}\| \equiv \min \mathbf{w}^T \mathbf{w}/2$

$$\min_{\mathbf{w}, b} \frac{1}{2} \mathbf{w}^T \mathbf{w}$$

subject to $y_i((\mathbf{w}^T \mathbf{x}_i) + b) \geq 1,$
 $i = 1, \dots, l.$

Higher Dimensional Feature Spaces

- Earlier we tried to find a linear separating hyperplane
Data may not be linear separable
- Non-separable case: **allow training errors**

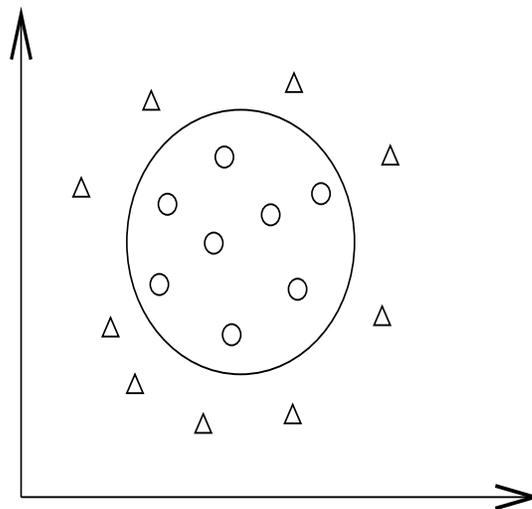
$$\min_{\mathbf{w}, b, \xi} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^l \xi_i$$

$$y_i((\mathbf{w}^T \mathbf{x}_i) + b) \geq 1 - \xi_i,$$

$$\xi_i \geq 0, \quad i = 1, \dots, l$$

- $\xi_i > 1$, \mathbf{x}_i **not on the correct side** of the separating plane
- C : **large** penalty parameter, **most ξ_i are zero**

- Nonlinear case: **linear separable in other spaces ?**



- Higher dimensional (maybe infinite) feature space

$$\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \dots).$$

- Example: $\mathbf{x} \in R^3, \phi(\mathbf{x}) \in R^{10}$

$$\phi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_3, x_1^2, x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3)$$

- A standard problem [Cortes and Vapnik, 1995]:

$$\begin{aligned} & \min_{\mathbf{w}, b, \xi} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^l \xi_i \\ & \text{subject to} \quad y_i (\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \dots, l. \end{aligned}$$

Finding the Decision Function

- \mathbf{w} : a vector in a high dimensional space \Rightarrow maybe **infinite** variables
- The **dual** problem

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2} \boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - \mathbf{e}^T \boldsymbol{\alpha} \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, i = 1, \dots, l \\ & \mathbf{y}^T \boldsymbol{\alpha} = 0, \end{aligned}$$

where $Q_{ij} = y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ and $\mathbf{e} = [1, \dots, 1]^T$

$$\mathbf{w} = \sum_{i=1}^l \alpha_i y_i \phi(\mathbf{x}_i)$$

- **Primal and dual**: optimization theory. Not trivial.
Infinite dimensional programming.
- A **finite** problem:
#variables = #training data
- $Q_{ij} = y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ needs a **closed** form
Efficient calculation of **high dimensional inner products**
Kernel trick, $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

- **Example:** $\mathbf{x}_i \in R^3, \phi(\mathbf{x}_i) \in R^{10}$

$$\phi(\mathbf{x}_i) = (1, \sqrt{2}(x_i)_1, \sqrt{2}(x_i)_2, \sqrt{2}(x_i)_3, (x_i)_1^2, (x_i)_2^2, (x_i)_3^2, \sqrt{2}(x_i)_1(x_i)_2, \sqrt{2}(x_i)_1(x_i)_3, \sqrt{2}(x_i)_2(x_i)_3)$$

Then $\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$.

- **Popular methods:** $K(\mathbf{x}_i, \mathbf{x}_j) =$

$$e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}, \text{ (Radial Basis Function)}$$

$$(\mathbf{x}_i^T \mathbf{x}_j / a + b)^d \text{ (Polynomial kernel)}$$

Kernel Tricks

- Kernel: $K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y})$

- **No need** to explicitly know $\phi(\mathbf{x})$

- Common kernels $K(\mathbf{x}_i, \mathbf{x}_j) =$

$$e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}, \text{ (Radial Basis Function)}$$

$$(\mathbf{x}_i^T \mathbf{x}_j / a + b)^d \text{ (Polynomial kernel)}$$

- They can be inner product in **infinite** dimensional space

- Assume $x \in R^1$ and $\gamma > 0$.

$$\begin{aligned}
e^{-\gamma\|x_i-x_j\|^2} &= e^{-\gamma(x_i-x_j)^2} = e^{-\gamma x_i^2 + 2\gamma x_i x_j - \gamma x_j^2} \\
&= e^{-\gamma x_i^2 - \gamma x_j^2} \left(1 + \frac{2\gamma x_i x_j}{1!} + \frac{(2\gamma x_i x_j)^2}{2!} + \frac{(2\gamma x_i x_j)^3}{3!} + \dots \right) \\
&= e^{-\gamma x_i^2 - \gamma x_j^2} \left(1 \cdot 1 + \sqrt{\frac{2\gamma}{1!}} x_i \cdot \sqrt{\frac{2\gamma}{1!}} x_j + \sqrt{\frac{(2\gamma)^2}{2!}} x_i^2 \cdot \sqrt{\frac{(2\gamma)^2}{2!}} x_j^2 \right. \\
&\quad \left. + \sqrt{\frac{(2\gamma)^3}{3!}} x_i^3 \cdot \sqrt{\frac{(2\gamma)^3}{3!}} x_j^3 + \dots \right) \\
&= \phi(x_i)^T \phi(x_j),
\end{aligned}$$

where

$$\phi(x) = e^{-\gamma x^2} \left[1, \sqrt{\frac{2\gamma}{1!}} x, \sqrt{\frac{(2\gamma)^2}{2!}} x^2, \sqrt{\frac{(2\gamma)^3}{3!}} x^3, \dots \right]^T.$$

Decision function

- \mathbf{w} : maybe an **infinite** vector
- At optimum

$$\mathbf{w} = \sum_{i=1}^l \alpha_i y_i \phi(\mathbf{x}_i)$$

- Decision function

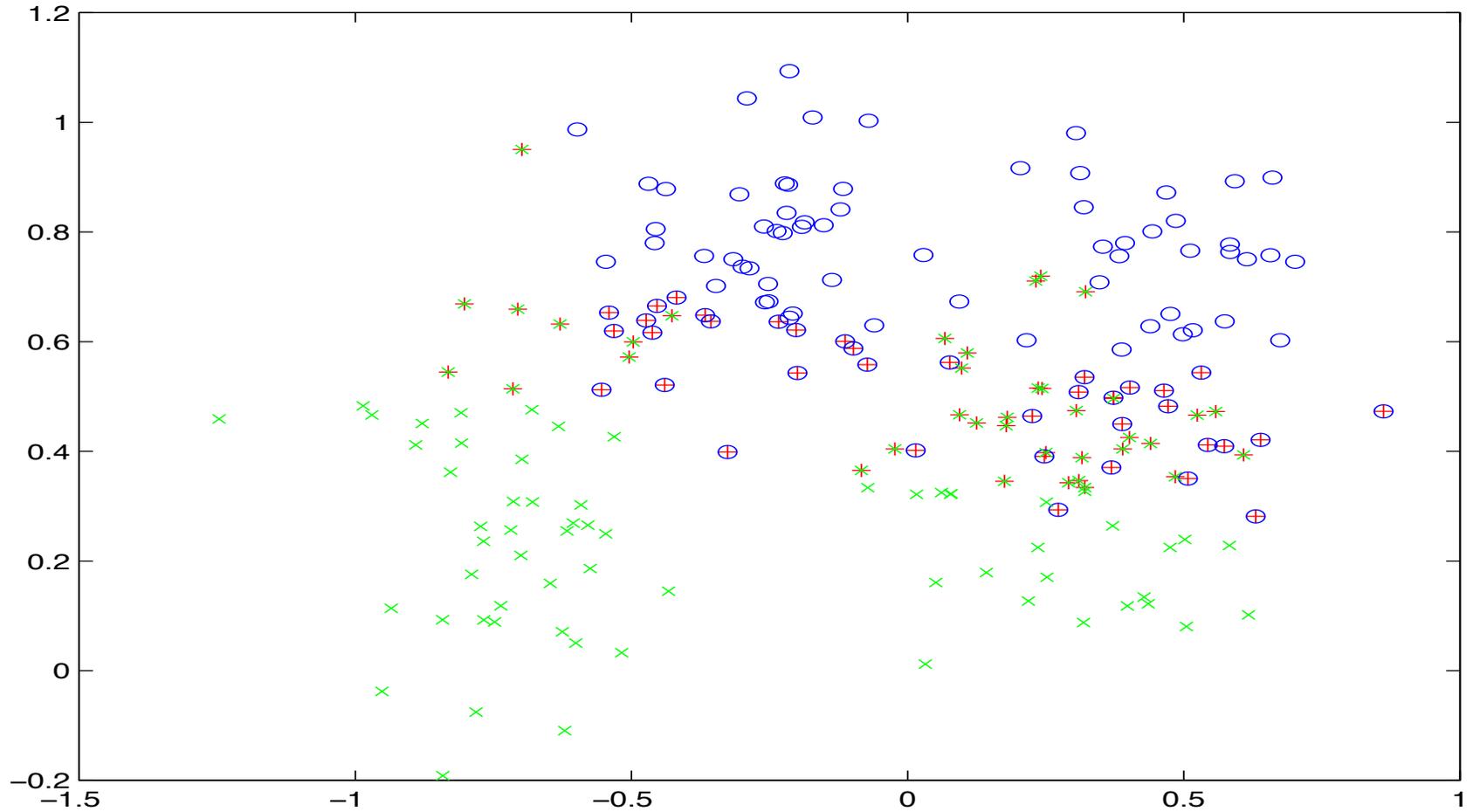
$$\begin{aligned} & \mathbf{w}^T \phi(\mathbf{x}) + b \\ &= \sum_{i=1}^l \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + b \\ &= \sum_{i=1}^l \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \end{aligned}$$

No need to have \mathbf{w}

- > 0 : 1st class, < 0 : 2nd class
- Only $\phi(\mathbf{x}_i)$ of $\alpha_i > 0$ used

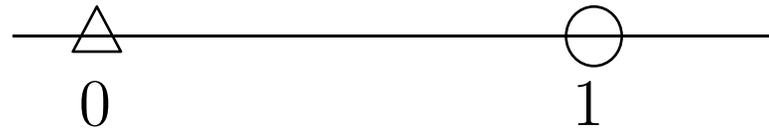
$\alpha_i > 0 \Rightarrow$ support vectors

Support Vectors: More Important Data



A Toy Example

- Two training data in R^1 :



- What is the separating hyperplane ?

Primal Problem

• $\mathbf{x}_1 = 0, \mathbf{x}_2 = 1$ with $\mathbf{y} = [-1, 1]^T$.

• Primal problem

$$\min_{w,b} \quad \frac{1}{2}w^2$$

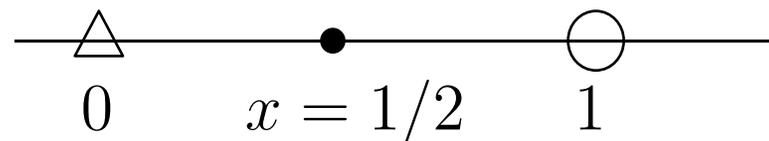
$$\text{subject to} \quad w \cdot 1 + b \geq 1, \quad (1)$$

$$-1(w \cdot 0 + b) \geq 1. \quad (2)$$

- $-b \geq 1$ and $w \geq 1 - b \geq 2$.
- For any (w, b) satisfying two inequality constraints

$$w \geq 2$$

- We are minimizing $\frac{1}{2}w^2$
The smallest possibility is $w = 2$.
- $(w, b) = (2, -1)$ is the optimal solution.
- The separating hyperplane $2x - 1 = 0$
In the middle of the two training data:



Dual Problem

- Formula derived before

$$\min_{\alpha \in R^l} \quad \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) - \sum_{i=1}^l \alpha_i$$

subject to $\alpha_i \geq 0, i = 1, \dots, l$, and $\sum_{i=1}^l \alpha_i y_i = 0$.

- Get the objective function

$$\mathbf{x}_1^T \mathbf{x}_1 = 0, \mathbf{x}_1^T \mathbf{x}_2 = 0$$

$$\mathbf{x}_2^T \mathbf{x}_1 = 0, \mathbf{x}_2^T \mathbf{x}_2 = 1$$

• Objective function

$$\begin{aligned} & \frac{1}{2}\alpha_1^2 - (\alpha_1 + \alpha_2) \\ &= \frac{1}{2} \begin{bmatrix} \alpha_1 & \alpha_2 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} - \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}. \end{aligned}$$

• Constraints

$$\alpha_1 - \alpha_2 = 0, 0 \leq \alpha_1, 0 \leq \alpha_2.$$

- $\alpha_2 = \alpha_1$ to the objective function,

$$\frac{1}{2}\alpha_1^2 - 2\alpha_2$$

- Smallest value at $\alpha_1 = 2$.

α_2 as well

- If smallest value < 0
clipped to 0

Let Us Try A Practical Example

- A problem from astroparticle physics

```
1.0 1:2.617300e+01 2:5.886700e+01 3:-1.894697e-01 4:1.251225e+02
1.0 1:5.707397e+01 2:2.214040e+02 3:8.607959e-02 4:1.229114e+02
1.0 1:1.725900e+01 2:1.734360e+02 3:-1.298053e-01 4:1.250318e+02
1.0 1:2.177940e+01 2:1.249531e+02 3:1.538853e-01 4:1.527150e+02
1.0 1:9.133997e+01 2:2.935699e+02 3:1.423918e-01 4:1.605402e+02
1.0 1:5.537500e+01 2:1.792220e+02 3:1.654953e-01 4:1.112273e+02
1.0 1:2.956200e+01 2:1.913570e+02 3:9.901439e-02 4:1.034076e+02
```

- Training and testing sets available: 3,089 and 4,000
- Data format is an issue

SVM software: LIBSVM

- <http://www.csie.ntu.edu.tw/~cjlin/libsvm>
- Now one of the most used SVM software
- Installation
- On Unix:
Download zip file and make
- On Windows:
 - Download zip file and make
 - `c:nmake -f Makefile.win`
 - Windows binaries included in the package

Usage of LIBSVM

● Training

Usage: `svm-train [options] training_set_file`

options:

`-s svm_type` : set type of SVM (default 0)

0 -- C-SVC

1 -- nu-SVC

2 -- one-class SVM

3 -- epsilon-SVR

4 -- nu-SVR

`-t kernel_type` : set type of kernel function

● Testing

Usage: `svm-predict test_file model_file output`

Training and Testing

- Training

```
$/svm-train train.1  
.....*  
optimization finished, #iter = 6131  
nu = 0.606144  
obj = -1061.528899, rho = -0.495258  
nSV = 3053, nBSV = 724  
Total nSV = 3053
```

- Testing

```
$/svm-predict test.1 train.1.model  
test.1.predict  
Accuracy = 66.925% (2677/4000)
```

What does this Output Mean

- obj: the optimal objective value of the dual SVM
- rho: $-b$ in the decision function
- nSV and nBSV: number of support vectors and bounded support vectors
(i.e., $\alpha_i = C$).
- nu-svm is a somewhat equivalent form of C-SVM where C is replaced by ν .

Why this Fails

- After training, nearly 100% support vectors
- Training and testing accuracy **different**

```
$/svm-predict train.1 train.1.model o
Accuracy = 99.7734% (3082/3089)
```

- Most kernel elements:

$$K_{ij} \begin{cases} = 1 & \text{if } i = j, \\ \rightarrow 0 & \text{if } i \neq j. \end{cases}$$

Data Scaling

- Without scaling
Attributes in **greater numeric ranges may dominate**
- Example:

	height	sex
x_1	150	F
x_2	180	M
x_3	185	M

and

$$y_1 = 0, y_2 = 1, y_3 = 1.$$

- The separating hyperplane

Δ
 \mathbf{x}_1

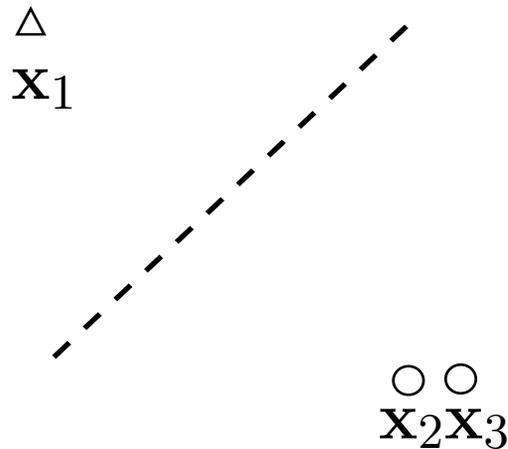
0 0
 $\mathbf{x}_2 \mathbf{x}_3$

- Decision strongly depends on the first attribute
- What if the second is more important

- Linearly scale the first to $[0, 1]$ by:

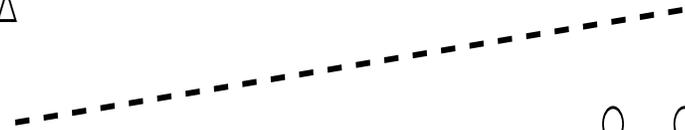
$$\frac{\text{1st attribute} - 150}{185 - 150},$$

- New points and separating hyperplane



- Transformed to the original space,

$x_1 \Delta$



$0 \quad 0$
 $x_2 \quad x_3$

- The second attribute plays a role

After Data Scaling

- A common mistake

```
$/svm-scale -1 -1 -u 1 train.1 > train.1.scale  
$/svm-scale -1 -1 -u 1 test.1 > test.1.scale
```

- Same factor on training and testing

```
$/svm-scale -s rangel train.1 > train.1.scale
```

```
$/svm-scale -r rangel test.1 > test.1.scale
```

```
$/svm-train train.1.scale
```

```
$/svm-predict test.1.scale train.1.scale.model  
test.1.predict
```

→ Accuracy = 96.15%

- We store the scaling factor used in training and apply them for testing set

More on Training

- Train scaled data and then prediction

```
$/svm-train train.1.scale
```

```
$/svm-predict test.1.scale train.1.scale.model  
test.1.predict
```

→ Accuracy = 96.15%

- Training accuracy now is

```
$/svm-predict train.1.scale train.1.scale.model
```

```
Accuracy = 96.439% (2979/3089) (classification)
```

- Default parameter

- $C = 1, \gamma = 0.25$

Different Parameters

- If we use $C = 20, \gamma = 400$

```
$/svm-train -c 20 -g 400 train.1.scale  
./svm-predict train.1.scale train.1.scale.mod  
Accuracy = 100% (3089/3089) (classification)
```

- 100% training accuracy but

```
$/svm-predict test.1.scale train.1.scale.mod  
Accuracy = 82.7% (3308/4000) (classification)
```

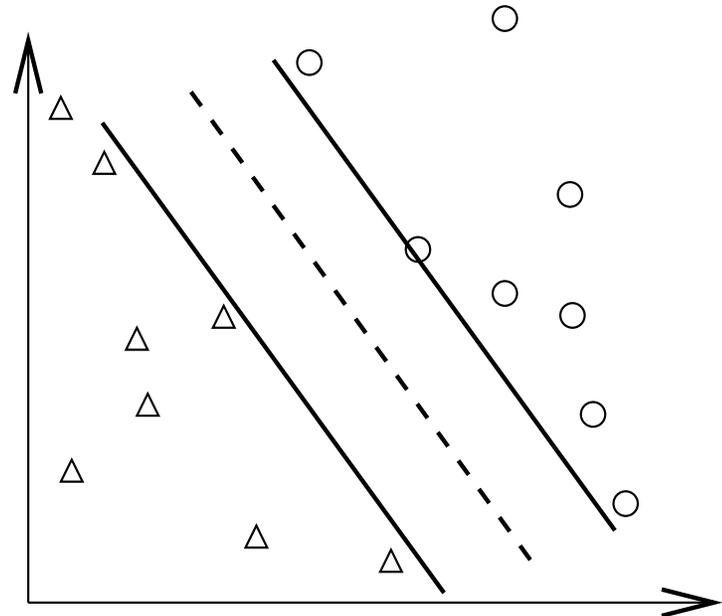
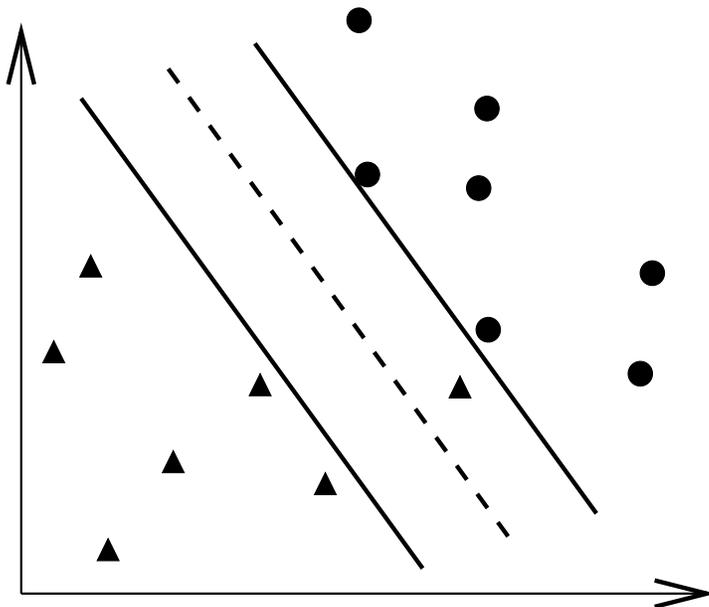
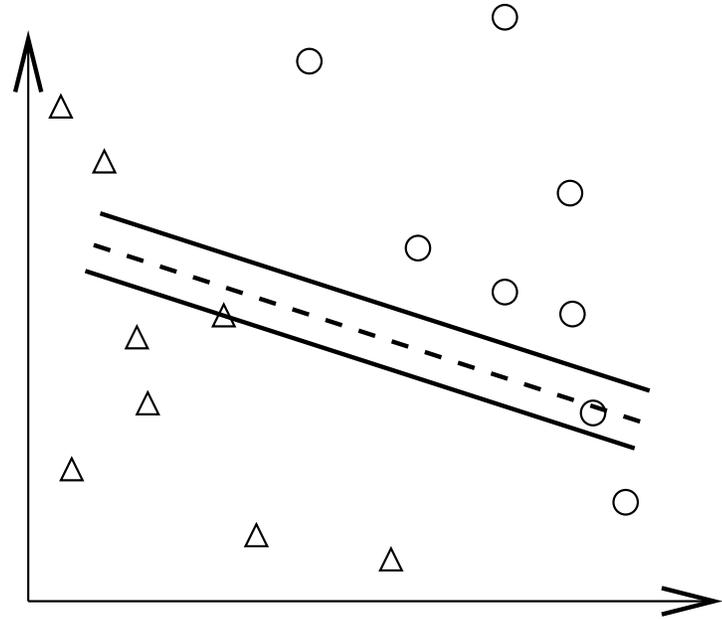
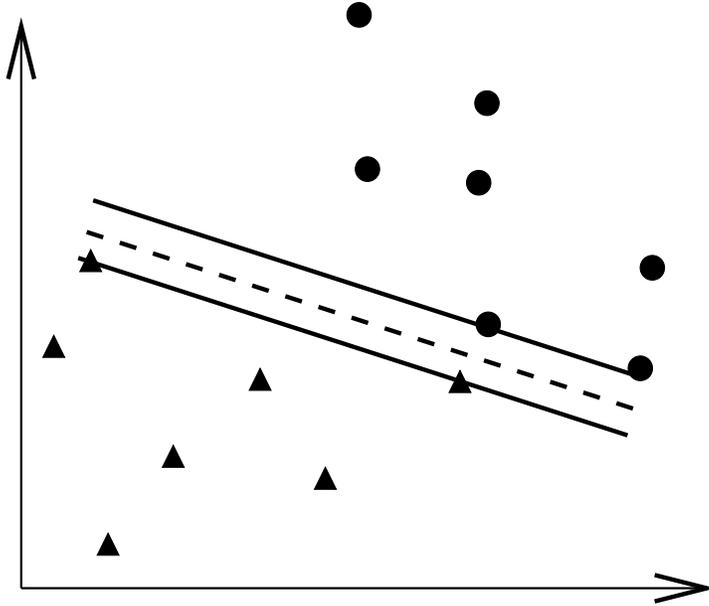
- Very bad test accuracy

- Overfitting happens

Overfitting and Underfitting

- When training and predicting a data, we should
 - Avoid **underfitting**: small training error
 - Avoid **overfitting**: small testing error

● and ▲: training; ○ and △: testing



Overfitting

- In theory
 - You can easily achieve 100% training accuracy
- This is useless
- Surprisingly
 - Many application papers did this

Parameter Selection

- Is very important
- Now parameters are C , kernel parameters
- Example:

$$\gamma \text{ of } e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}$$

$$a, b, d \text{ of } (\mathbf{x}_i^T \mathbf{x}_j / a + b)^d$$

- How to select them ?
So performance better ?

Performance Evaluation

- Training errors not important; only test errors count
- l training data, $\mathbf{x}_i \in R^n, y_i \in \{+1, -1\}, i = 1, \dots, l$, a learning machine:

$$x \rightarrow f(\mathbf{x}, \alpha), f(\mathbf{x}, \alpha) = 1 \text{ or } -1.$$

Different α : different machines

- The expected test error (generalized error)

$$R(\alpha) = \int \frac{1}{2} |y - f(\mathbf{x}, \alpha)| dP(\mathbf{x}, y)$$

y : class of \mathbf{x} (i.e. 1 or -1)

- $P(\mathbf{x}, y)$ unknown, empirical risk (training error):

$$R_{emp}(\alpha) = \frac{1}{2l} \sum_{i=1}^l |y_i - f(\mathbf{x}_i, \alpha)|$$

- $\frac{1}{2}|y_i - f(\mathbf{x}_i, \alpha)|$: loss, choose $0 \leq \eta \leq 1$, with probability at least $1 - \eta$:

$$R(\alpha) \leq R_{emp}(\alpha) + \text{another term}$$

- A good pattern recognition method:
minimize both terms at the same time
- $R_{emp}(\alpha) \rightarrow 0$
another term \rightarrow large

Performance Evaluation (Cont.)

- In practice
 - Available data \Rightarrow training and validation
- Train the training
- Test the validation
- k -fold cross validation:
 - Data randomly separated to k groups.
 - Each time $k - 1$ as training and one as testing

CV and Test Accuracy

- If we select parameters so that CV is the highest,
 - Does CV represent future test accuracy ?
 - Slightly different
- If we have enough parameters, we can achieve 100% CV as well
 - e.g. more parameters than # of training data
 - But test accuracy may be different
- So
 - Available data with class labels
 - ⇒ training, validation, testing

- Using CV on training + validation
- Predict testing with the best parameters from CV

A Simple Procedure

1. Conduct simple **scaling** on the data
 2. Consider **RBF** kernel $K(x, y) = e^{-\gamma\|x-y\|^2}$
 3. Use cross-validation to find the **best parameter** C and γ
 4. Use the best C and γ to **train the whole** training set
 5. Test
- Best C and γ by training $k - 1$ and **the whole** ?

In theory, a **minor** difference

No problem in practice

Parameter Selection Procedure in LIBSVM

- grid search + CV

```
./grid.py train.1 train.1.scale
```

```
[local] -1 -7 85.1408 (best c=0.5, g=0.0078125, rate=85.1408)
```

```
[local] 5 -7 95.4354 (best c=32.0, g=0.0078125, rate=95.4354)
```

```
.
```

```
.
```

```
.
```

- grid.py: a python script in the python directory of LIBSVM

● Easy parallelization on a cluster

```
$/grid.py train.1 train.1.scale
```

```
[linux1] -1 -7 85.1408 (best c=0.5, g=0.0078125, rate=85.1408)
```

```
[linux7] 5 -7 95.4354 (best c=32.0, g=0.0078125, rate=95.4354)
```

```
.  
. .  
. .
```

Parallel Parameter Selection

- Specify machine names in grid.py

```
telnet_workers = []  
ssh_workers = ['linux1', 'linux1', 'linux2',  
              'linux3']  
nr_local_worker = 1
```

linux1: more powerful or two CPUs

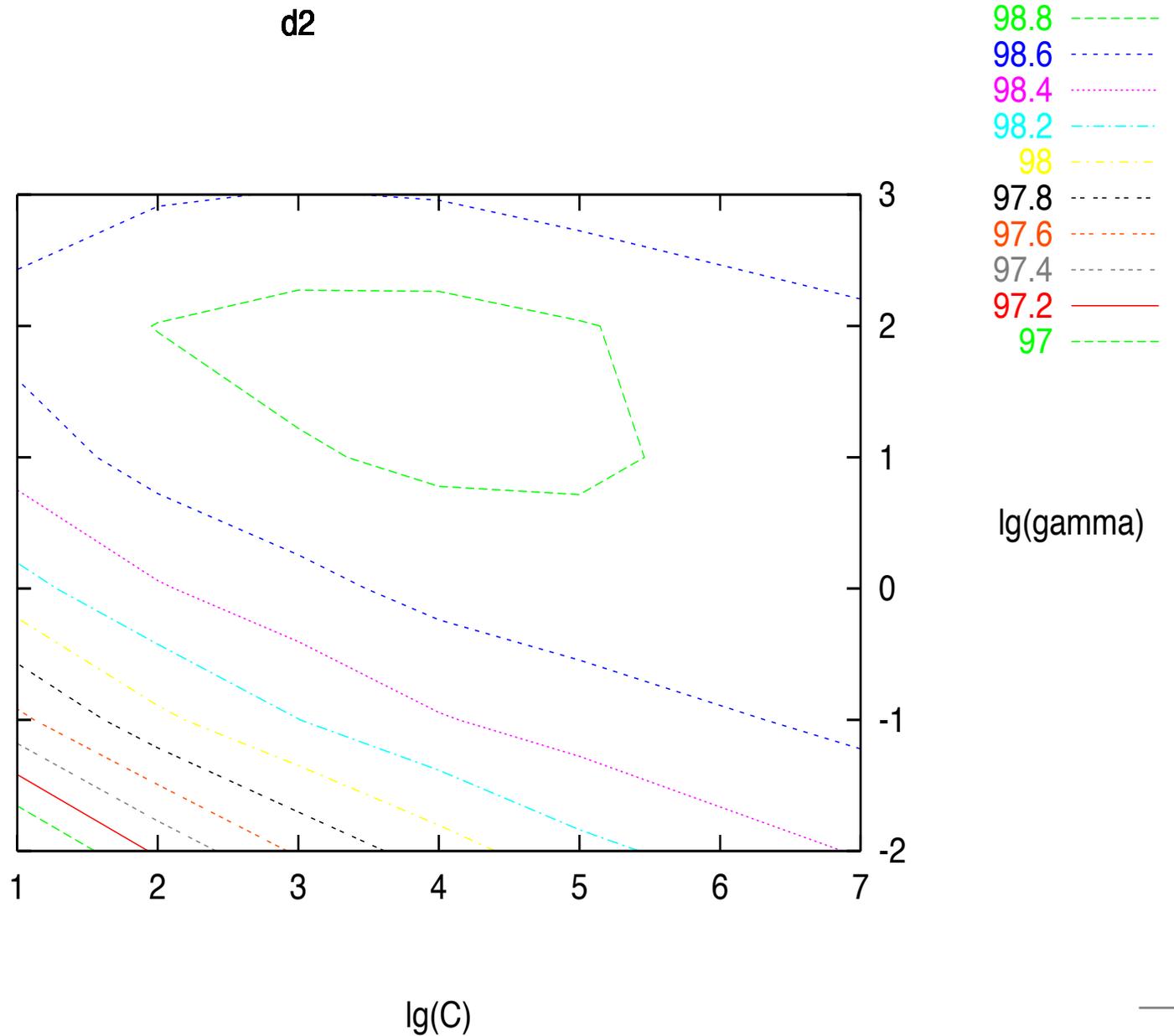
- A simple centralized control

Load balancing not a problem

- We can use other tools

Too simple so not consider them

Contour of Parameter Selection



Simple script in LIBSVM

- easy.py: a script for dummies

```
$python easy.py train.1 test.1  
Scaling training data...  
Cross validation...  
Best c=2.0, g=2.0  
Training...  
Scaling testing data...  
Testing...  
Accuracy = 96.875% (3875/4000)
```

Example: Engine Misfire Detection

Problem Description

- First problem of IJCNN Challenge 2001, data from Ford
- Given time series length $T = 50,000$
- The k th data

$$x_1(k), x_2(k), x_3(k), x_4(k), x_5(k), y(k)$$

- $y(k) = \pm 1$: output, affected **only** by $x_1(k), \dots, x_4(k)$
- $x_5(k) = 1$, k th data considered for evaluating accuracy
- 50,000 training data, 100,000 testing data (in two sets)

- Past and future information may affect $y(k)$
- $x_1(k)$: periodically nine 0s, one 1, nine 0s, one 1, and so on.
- Example:

0.000000	-0.999991	0.169769	0.000000	1.000000
0.000000	-0.659538	0.169769	0.000292	1.000000
0.000000	-0.660738	0.169128	-0.020372	1.000000
1.000000	-0.660307	0.169128	0.007305	1.000000
0.000000	-0.660159	0.169525	0.002519	1.000000
0.000000	-0.659091	0.169525	0.018198	1.000000
0.000000	-0.660532	0.169525	-0.024526	1.000000
0.000000	-0.659798	0.169525	0.012458	1.000000

- $x_4(k)$ more important

Background: Engine Misfire Detection

- How engine works
 - Air-fuel mixture injected to cylinder
 - intact, compression, combustion, exhaustion
- Engine misfire: a substantial fraction of a cylinder's air-fuel mixture fails to ignite
- Frequent misfires: pollutants and costly replacement
- On-board detection:
 - Engine crankshaft rotational dynamics with a position sensor
- Training data: from some **expensive** experimental environment

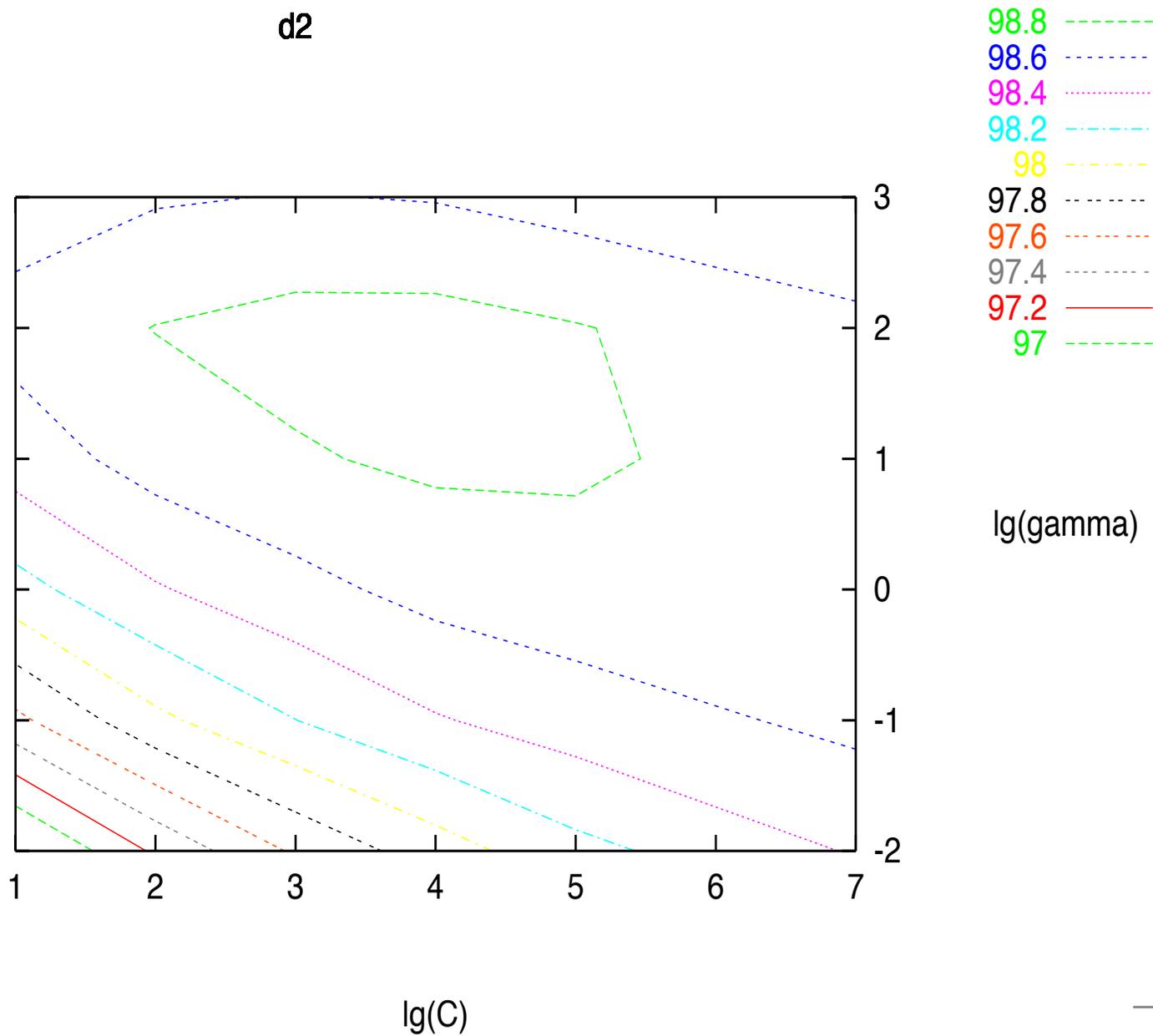
Encoding Schemes

- For SVM: each data is a vector
- $x_1(k)$: periodically nine 0s, one 1, nine 0s, one 1, ...
 - 10 binary attributes
 $x_1(k - 5), \dots, x_1(k + 4)$ for the k th data
 - $x_1(k)$: an integer in 1 to 10
 - Which one is better
 - **We think 10 binaries better for SVM**
- $x_4(k)$ more important
 - **Including $x_4(k - 5), \dots, x_4(k + 4)$ for the k th data**
- Each training data: 22 attributes

Training SVM

- Selecting parameters; generating a good model for prediction
- RBF kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}$
- Two parameters: γ and C
- Five-fold cross validation on 50,000 data
Data randomly separated to five groups.
Each time four as training and one as testing
- Use $C = 2^4$, $\gamma = 2^2$ and train 50,000 data for the **final model**

d2



- Test set 1: 656 errors, Test set 2: 637 errors
- About 3000 support vectors of 50,000 training data
A good case for SVM
- This is just the outline. There are other details.
- **It is essential to do model selection.**

Dual Problems for Other Formulas

- So we think that for any optimization problem Lagrangian dual exists
- This is wrong
- Remember we calculate

$$\min \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^l \alpha_i [y_i (\mathbf{w}^T \phi(\mathbf{x}_i)) - 1]$$

by

$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0.$$

- Note that

$$f'(x) = 0 \Leftrightarrow x \text{ minimum}$$

is wrong

- Example

$$f(x) = x^3, x = 0 \text{ not minimum}$$

- This function must satisfy certain conditions
- Some papers wrongly derived the dual of their new formulations without checking conditions

- $[2, 2]^T$ satisfies constraints $0 \leq \alpha_1$ and $0 \leq \alpha_2$

It is optimal

- Primal-dual relation

$$\begin{aligned} w &= y_1 \alpha_1 x_1 + y_2 \alpha_2 x_2 \\ &= 1 \cdot 2 \cdot 1 + (-1) \cdot 2 \cdot 0 \\ &= 2 \end{aligned}$$

- The **same** as solving the primal

Multi-class Classification

- k classes
- One-against-all: Train k binary SVMs:

1st class vs. $(2 - k)$ th class
2nd class vs. $(1, 3 - k)$ th class
⋮

- k decision functions

$$(\mathbf{w}^1)^T \phi(\mathbf{x}) + b_1$$

⋮

$$(\mathbf{w}^k)^T \phi(\mathbf{x}) + b_k$$

- Select the index with the largest $(\mathbf{w}^j)^T \phi(\mathbf{x}) + b_j$

Multi-class Classification (Cont.)

- One-against-one: train $k(k - 1)/2$ binary SVMs
 $(1, 2), (1, 3), \dots, (1, k), (2, 3), (2, 4), \dots, (k - 1, k)$
Select the one with **the largest vote**
- This is the method used by LIBSVM
- Try a 4-class problem
6 binary SVMs

```
$libsvm-2.5/svm-train bsvm-2.05/vehicle.scale
optimization finished, #iter = 173
obj = -142.552559, rho = 0.748453
nSV = 194, nBSV = 183
optimization finished, #iter = 330
obj = -149.912202, rho = -0.786410
nSV = 227, nBSV = 217
optimization finished, #iter = 169
obj = -139.655613, rho = 0.998277
nSV = 186, nBSV = 177
optimization finished, #iter = 268
obj = -185.161735, rho = -0.674739
nSV = 253, nBSV = 244
optimization finished, #iter = 477
obj = -378.264371, rho = 0.177314
nSV = 405, nBSV = 394
optimization finished, #iter = 337
obj = -186.182860, rho = 1.104943
nSV = 261, nBSV = 247
Total nSV = 739
```

- There are many other methods

A comparison in [Hsu and Lin, 2002]

- For a software

We select one which is **generally good** but not **always the best**

- Finally I chose 1 vs. 1

Similar accuracy to others

Shortest training

A bit longer on testing than 1 vs. all

Why Shorter Training Time

- 1 vs. 1

$k(k - 1)/2$ problems, each $2l/k$ data on average

- 1 vs. all

k problems, each l data

- If solving the optimization problem:
polynomial of the size with degree d

- Their complexities

$$\frac{k(k - 1)}{2} O\left(\left(\frac{2l}{k}\right)^d\right) \text{ vs. } kO(l^d)$$

Conclusions

- Dealing with data is interesting especially if you get good accuracy
- Some basic understandings are essential when applying methods
e.g. the importance of validation
- No method is the best for all data
Deep understanding of one or two methods very helpful