中国科学院大学人工智能学院硕士课《模式识别》

支撑向量机与核方法

张煦尧(xyz@nlpr.ia.ac.cn)

2020年12月05日

University of Chinese Academy of Sciences

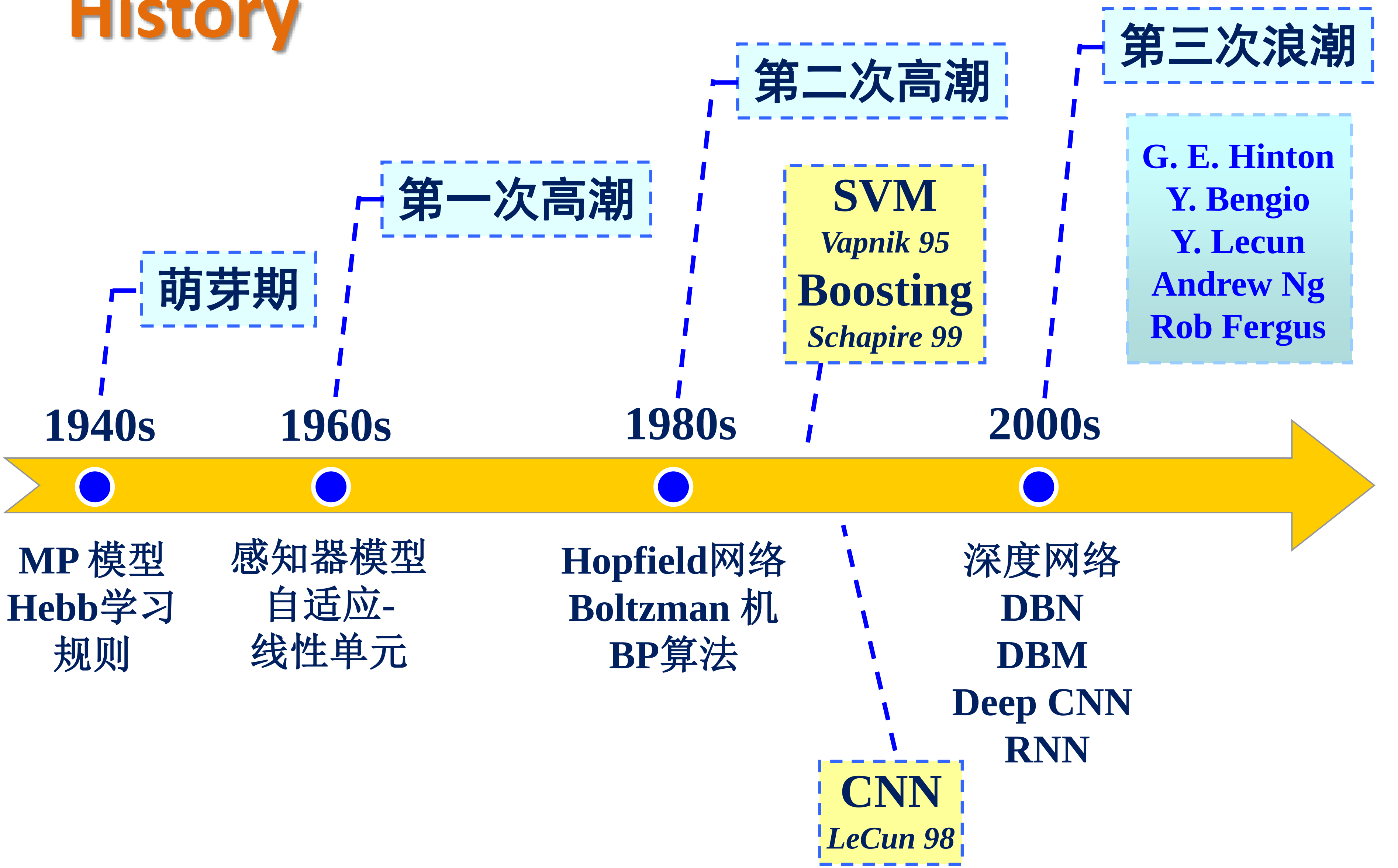
History

Boser, Guyon, Vapnik, A training algorithm for optimal margin classifiers. COLT 1992.

Cortes, Vapnik, Support-vector networks. Machine Learning, 1995.

- SVM is a classifier derived from statistical learning theory by Vapnik and Chervonenkis
- SVMs introduced by Boser, Guyon, Vapnik in COLT-92
- Initially popularized in the NIPS community, now an important and active field of all machine learning research.
- Special issues of Machine Learning Journal, and Journal of Machine Learning Research.

History



萌芽期
第一次高潮
第二次高潮
第三次浪潮
SVM
Vapnik 95
Boosting
Schapire 99
G. E. Hinton
Y. Bengio
Y. Lecun
Andrew Ng
Rob Fergus
1940s
1960s
1980s
2000s
MP 模型
Hebb学习
规则
感知器模型
自适应-
线性单元
Hopfield网络
Boltzman 机
BP算法
深度网络
DBN
DBM
Deep CNN
RNN
CNN
LeCun 98

SVM Theory

Structural Risk Minimization

- We want to get a low error rate on unseen data.
  - This is called "structural risk minimization"
  - Training error is "empirical risk minimization"

- It would be really helpful if we could get a guarantee:

  Test error rate <= train error rate + f(N, h, p)

  Where N = size of training set,

  h = measure of the model complexity,

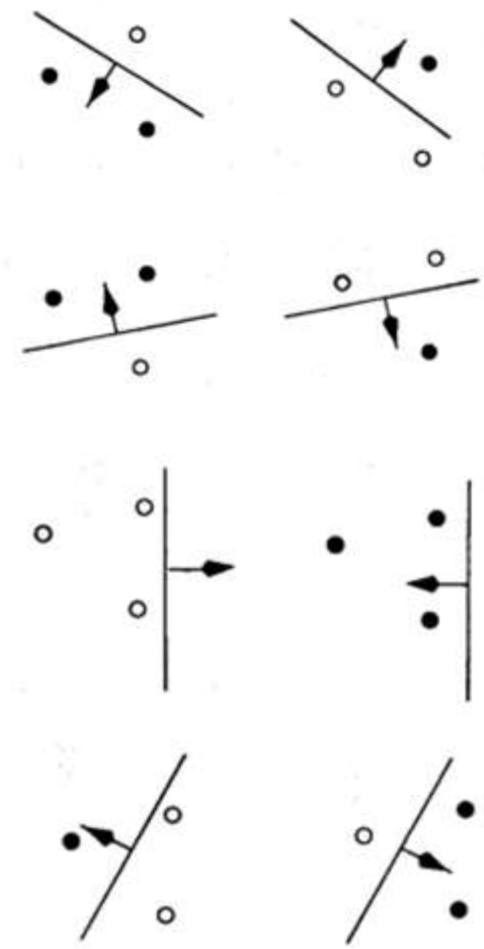
  p = the probability that this bound fails

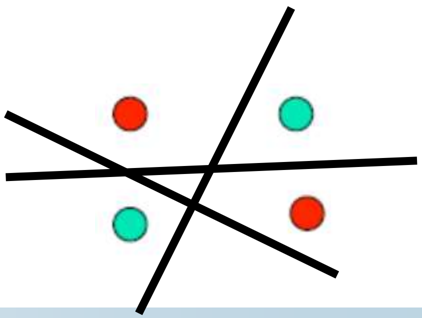
  We need p to allow for really unlucky test sets.

- Then we could choose the model complexity that minimizes the bound on the test error rate.

VC Dimension

- Pick n datapoints
- Assign labels of + or – to them at random
- If our model (e.g. a neural net with a certain number of hidden units) is powerful enough to learn any association of labels with the data, its too powerful!
- Characterize the power of a model class by asking how many datapoints it can “shatter”
- learn perfectly for all possible assignments of labels.
- This number of datapoints is called the Vapnik-Chervonenkis dimension.
- VC dimension for a 2D plane?
  – In 2-D, we can find a plane (i.e. a line) to deal with any labeling of three points. A 2-D hyperplane shatters 3 points





VC Dimension

- The VC dimension of a hyperplane in 2-D is 3.
  - In k dimensions it is k+1.

- Its just a coincidence that the VC dimension of a hyperplane is almost identical to the number of parameters it takes to define a hyperplane.

- A sine wave has infinite VC dimension and only 2 parameters!
  - By choosing the phase and period carefully we can shatter any random collection of one-dimensional datapoints

    - Let $x_i=10^i$ where i ranges from 1 to n. The classifier $y = \text{sign}\big(\sin(\alpha x)\big)$ can classify all x_i correctly for all possible combination of class labels on x_i

$$f(x) = b \sin(ax)$$

VC Dimension

- The VC-dimension of the nearest neighbor classifier is infinity, because no matter how many points you have, you get perfect classification on training data
  - 1-NN → K-NN: reduce VC dimension

- The higher the VC-dimension, the more flexible a classifier is

- VC-dimension, however, is a theoretical concept

- VC-dimension of most classifiers, in practice, is difficult to be computed exactly

- Qualitatively, if we think a classifier is flexible, it probably has a high VC-dimension

The Probabilistic Guarantee

$$E_{test} \leq E_{train} + \left(\frac{h + h\log(2N/h) - \log(p/4)}{N} \right)^{\frac{1}{2}}$$

where N = size of training set
h = VC dimension of the model class
p = upper bound on probability that this bound fails

Good generalization → large *N* and small *h*

So if we train models with different complexity, we should pick the one that minimizes this bound

Actually, this is only sensible if we think the bound is fairly tight, which it usually isn't. The theory provides insight, but in practice we still need some witchcraft. --- G. Hinton

Large Margin and VC Dimension

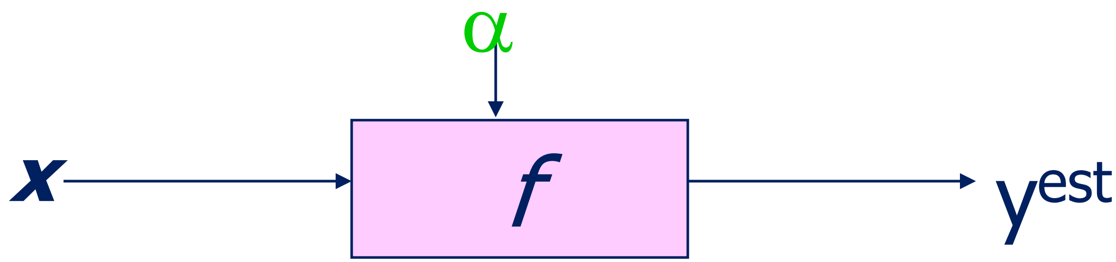
- If we use a large set of non-adaptive features, we can often make the two classes linearly separable.
  - But if we just fit any separating plane, it will not generalize well to new cases.
- If we fit the separating plane that maximizes the margin (the minimum distance to any of the data points), we will get much better generalization.
  - Intuitively, by maximizing the margin we are squeezing out all the surplus capacity that came from using a high-dimensional feature space.
- This can be justified by a whole lot of clever mathematics which shows that
  - large margin separators have lower VC dimension.
  - models with lower VC dimension have a smaller gap between the training and test error rates.

Some Philosophy

- **Simplest pattern classifier ? 最简单的模式分类器？**
  - Binary two-class problem
  - Linear classifier
- **Which linear classifier?**
  - The large margin one
- **Extension for nonlinear case**
  - Any function can be a linear function if transformed to a high-dimension space
  - Kernel method: the inner product in high-dimension space
  - Almost all kernel methods (including SVM) adopt the linear classifier as its initial study objective
- **Extension for multi-class case**
  - 1vsAll, 1vs1 …

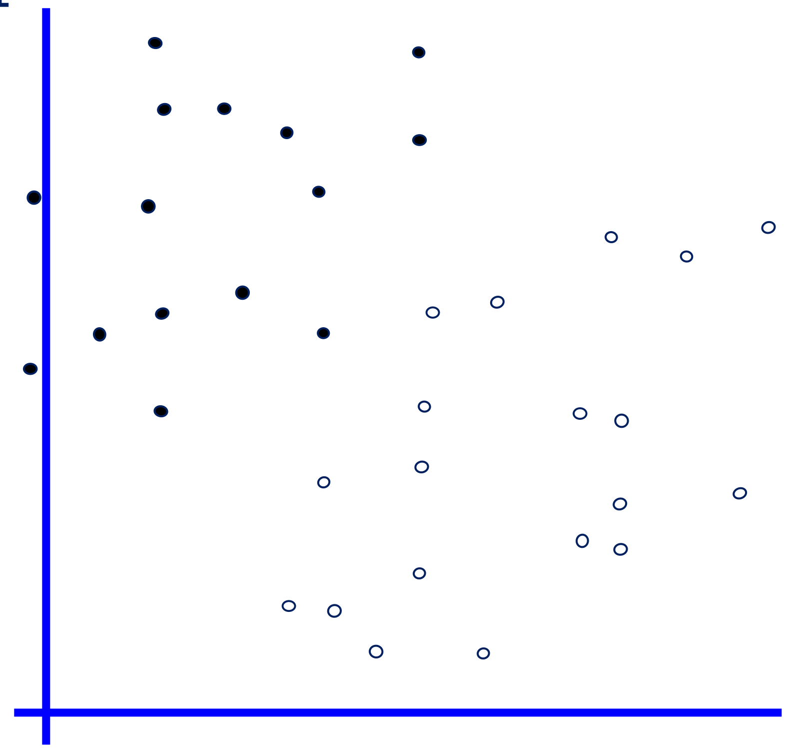
Hard-Margin SVM

Linear Classifiers



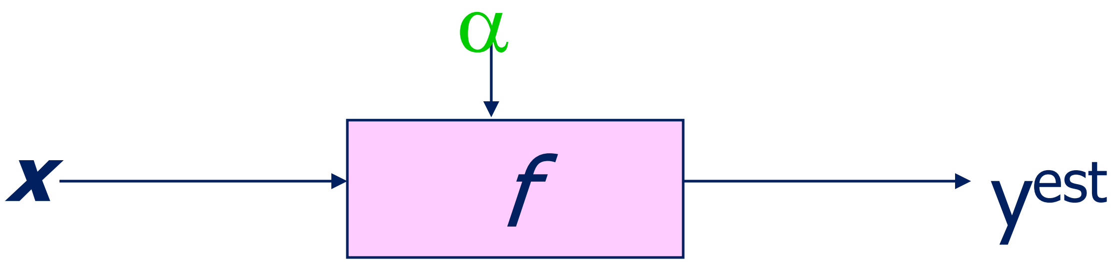
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}. \mathbf{x} + b)$

- • denotes +1
- ° denotes -1



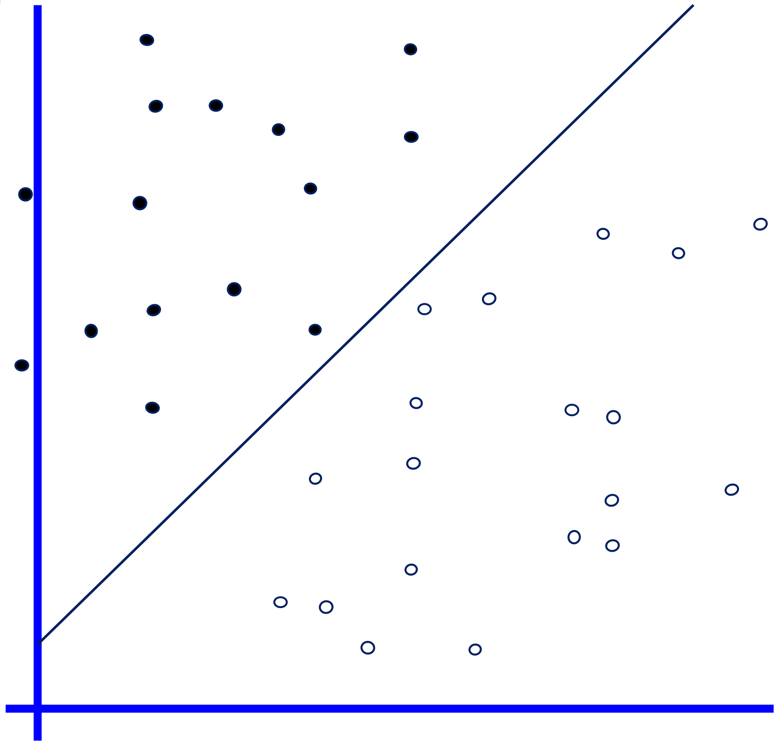
How would you classify this data?

Linear Classifiers



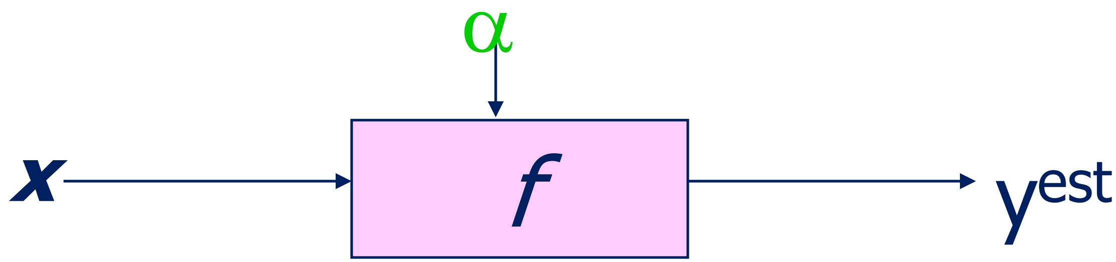
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} + b)$

- • denotes +1
- ° denotes -1



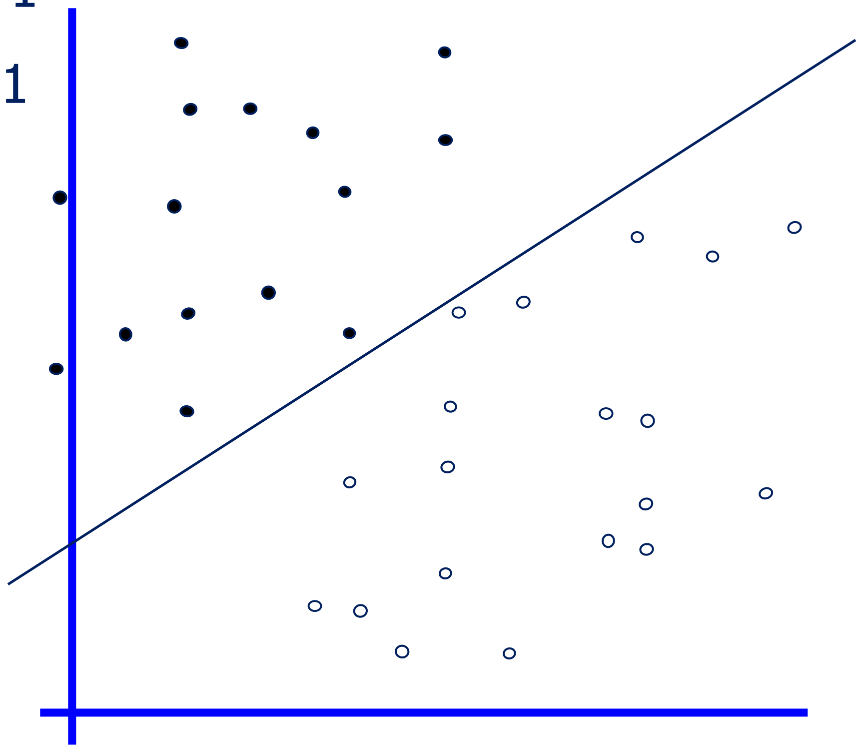
How would you classify this data?

Linear Classifiers



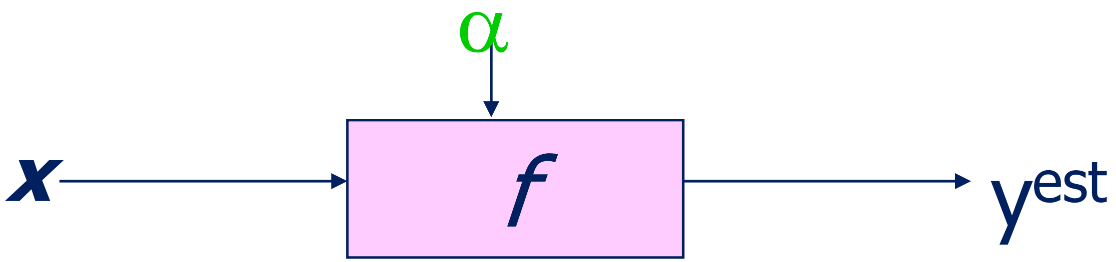
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} + b)$

- • denotes +1
- ° denotes -1



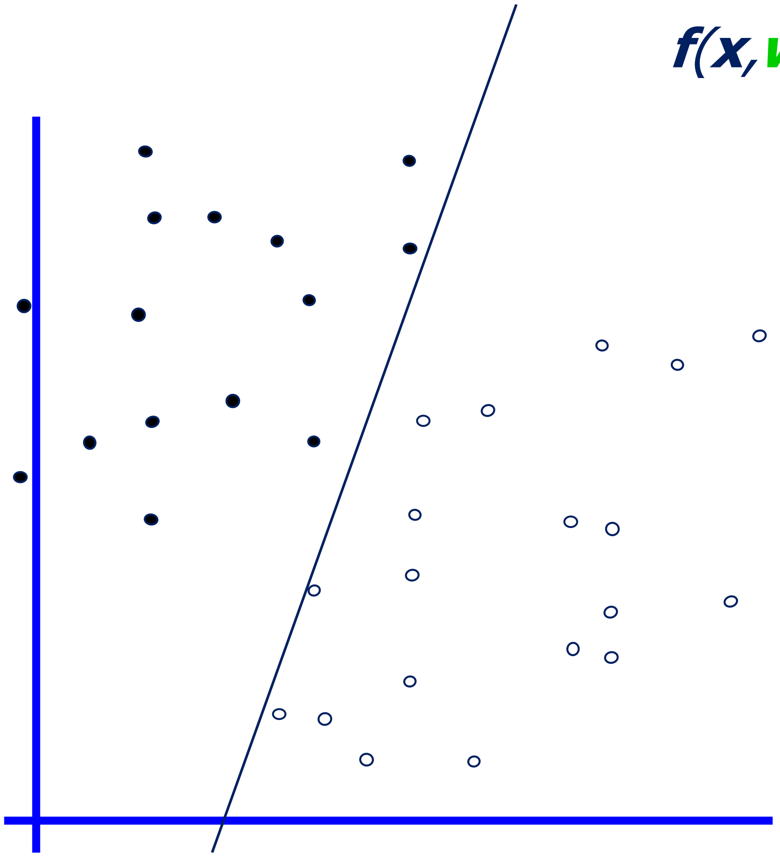
How would you classify this data?

Linear Classifiers



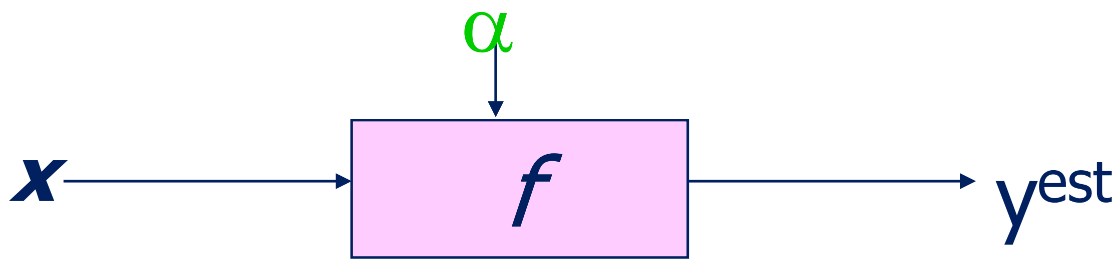
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} + b)$

- ● denotes +1
- ○ denotes -1

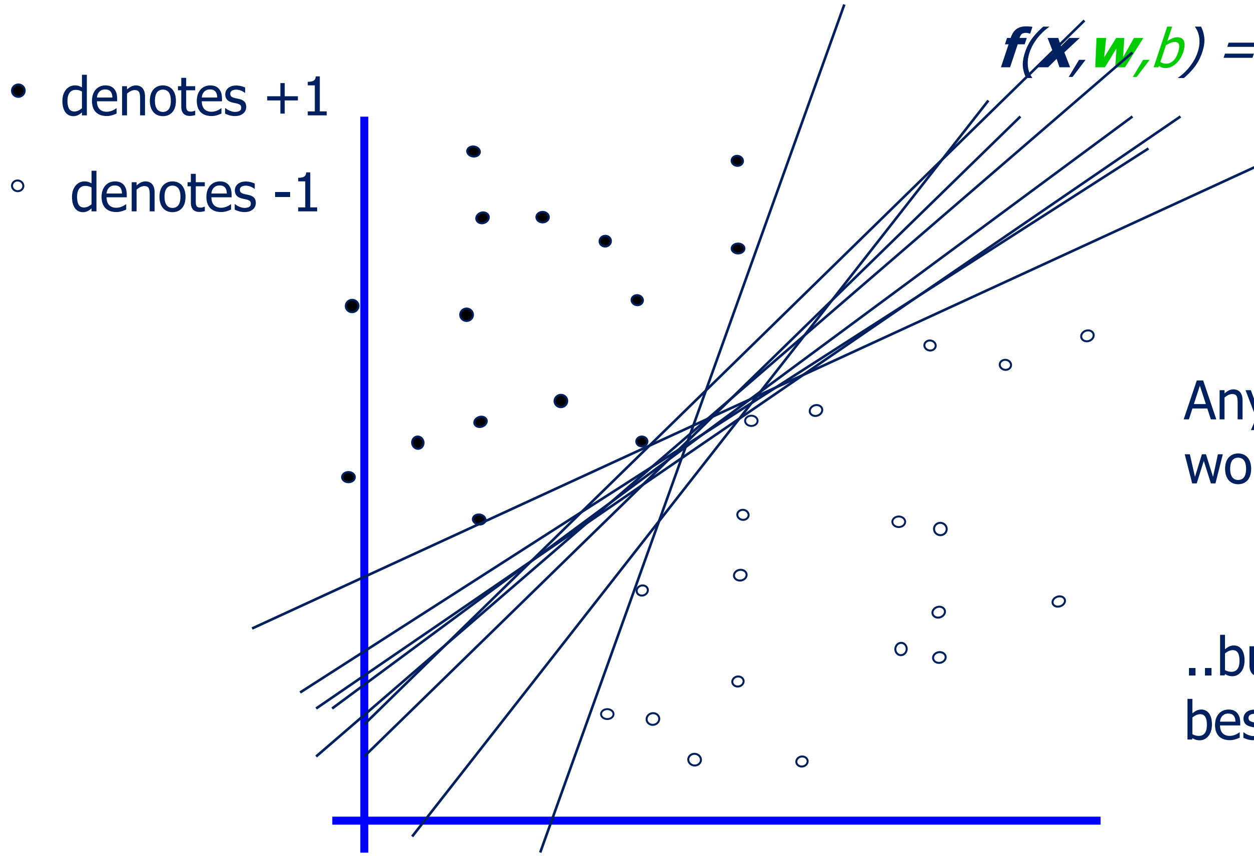


How would you classify this data?

Linear Classifiers



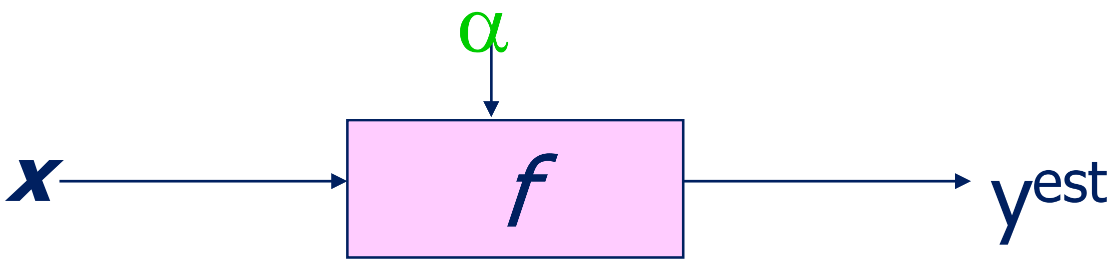
$$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} + b)$$



Any of these would be fine..

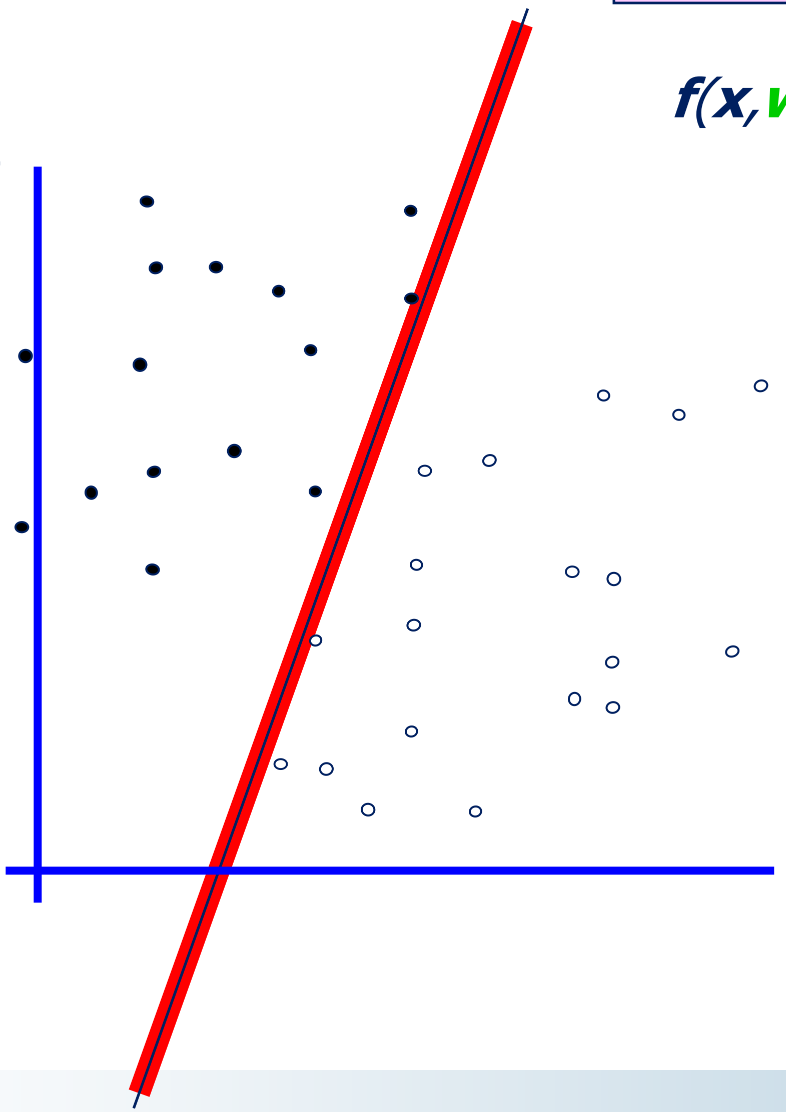
..but which is best?

Classifier Margin



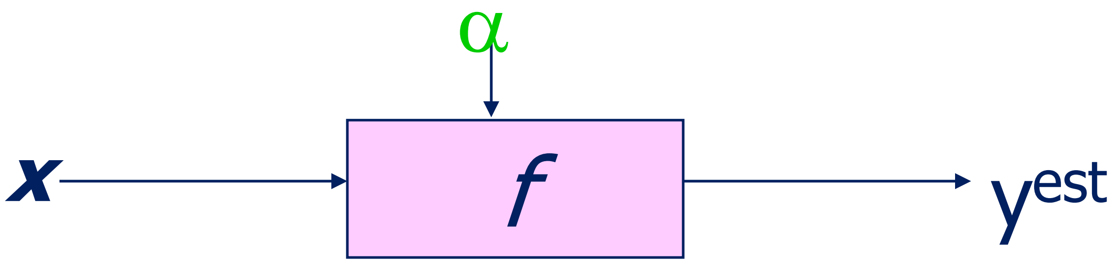
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} + b)$

- • denotes +1
- ° denotes -1

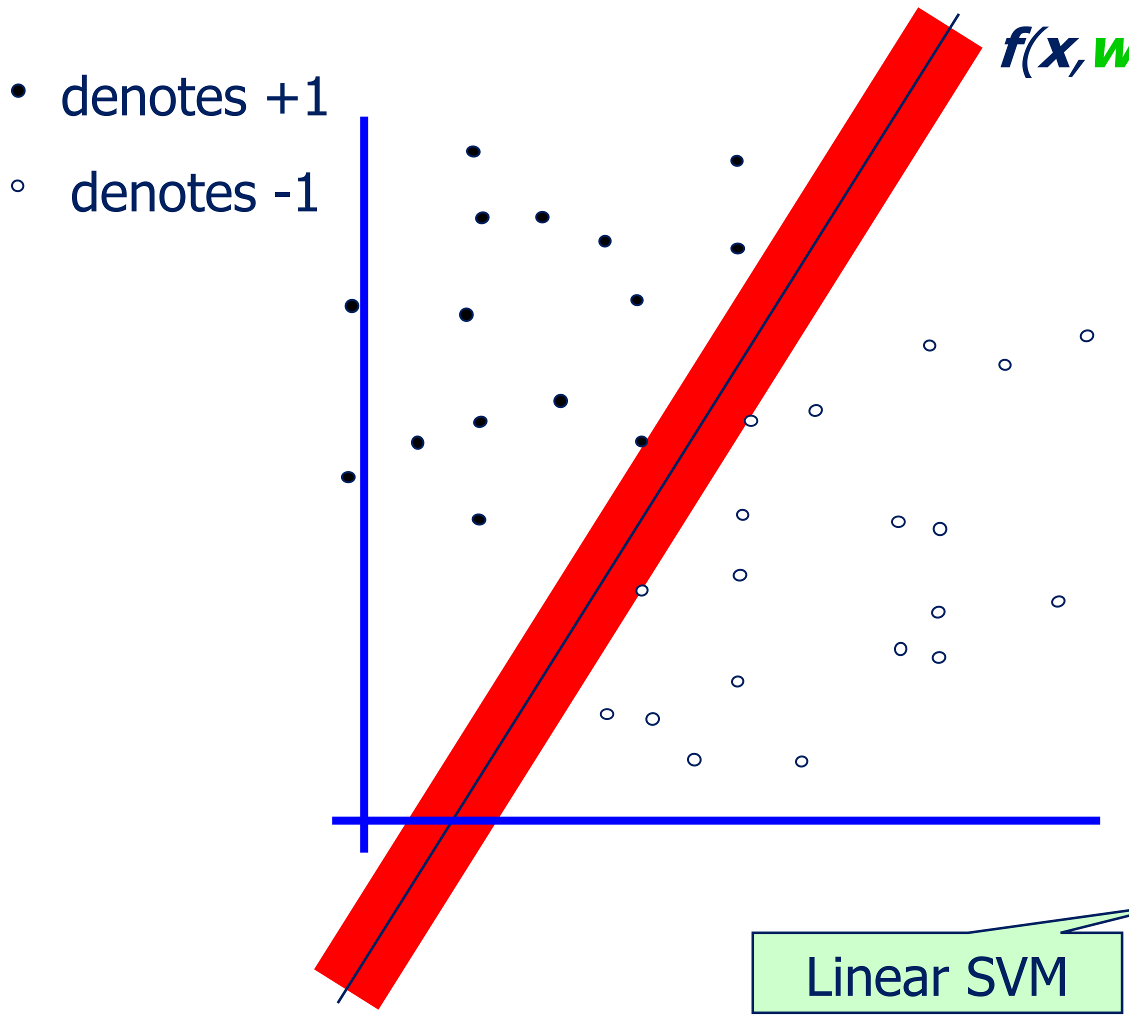


Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin



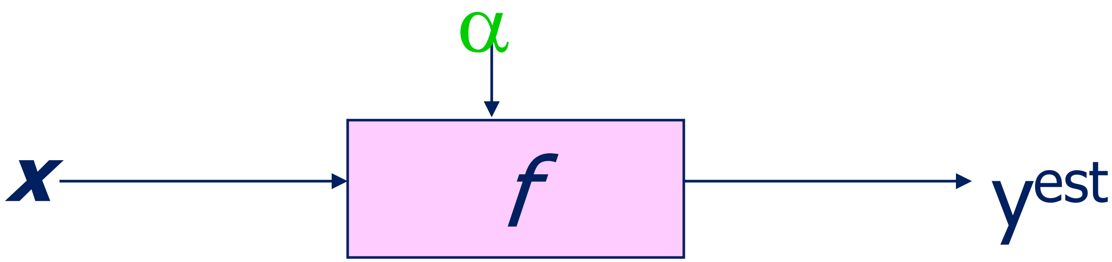
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{x} + b)$



The maximum margin linear classifier is the linear classifier with the maximum margin.

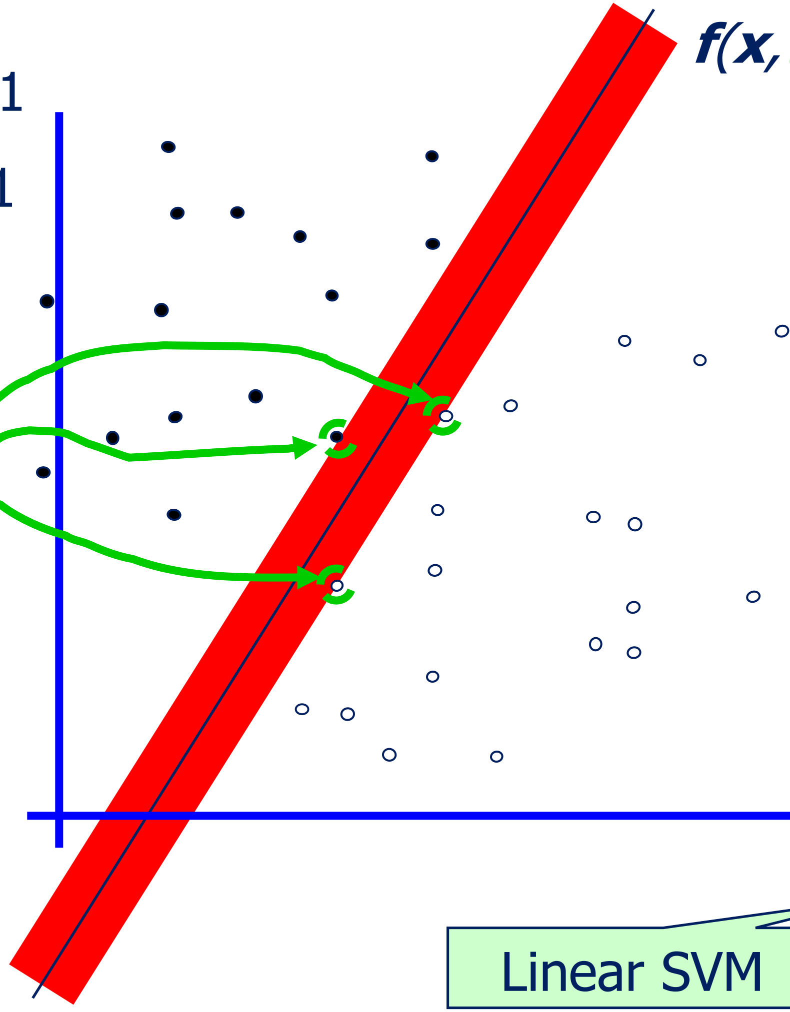
This is the simplest kind of SVM (Called an LSVM)

Maximum Margin



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

- • denotes +1
- ◦ denotes -1



Support Vectors are those datapoints that the margin pushes up against

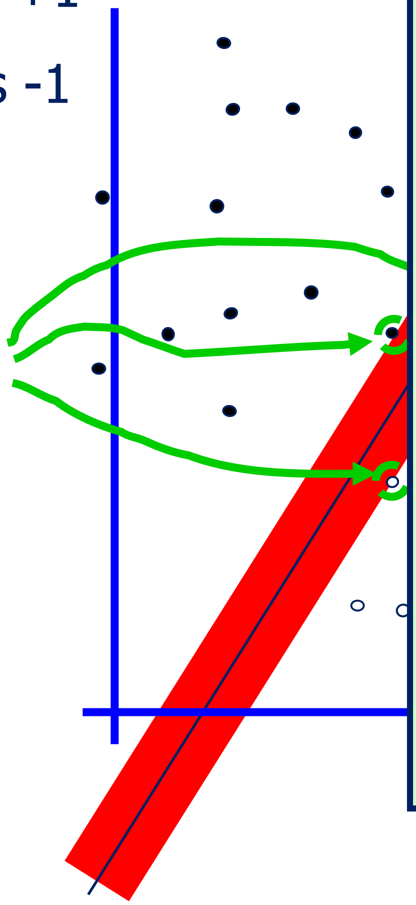
The maximum margin linear classifier is the linear classifier with the maximum margin.

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

Linear SVM

Why Maximum Margin?

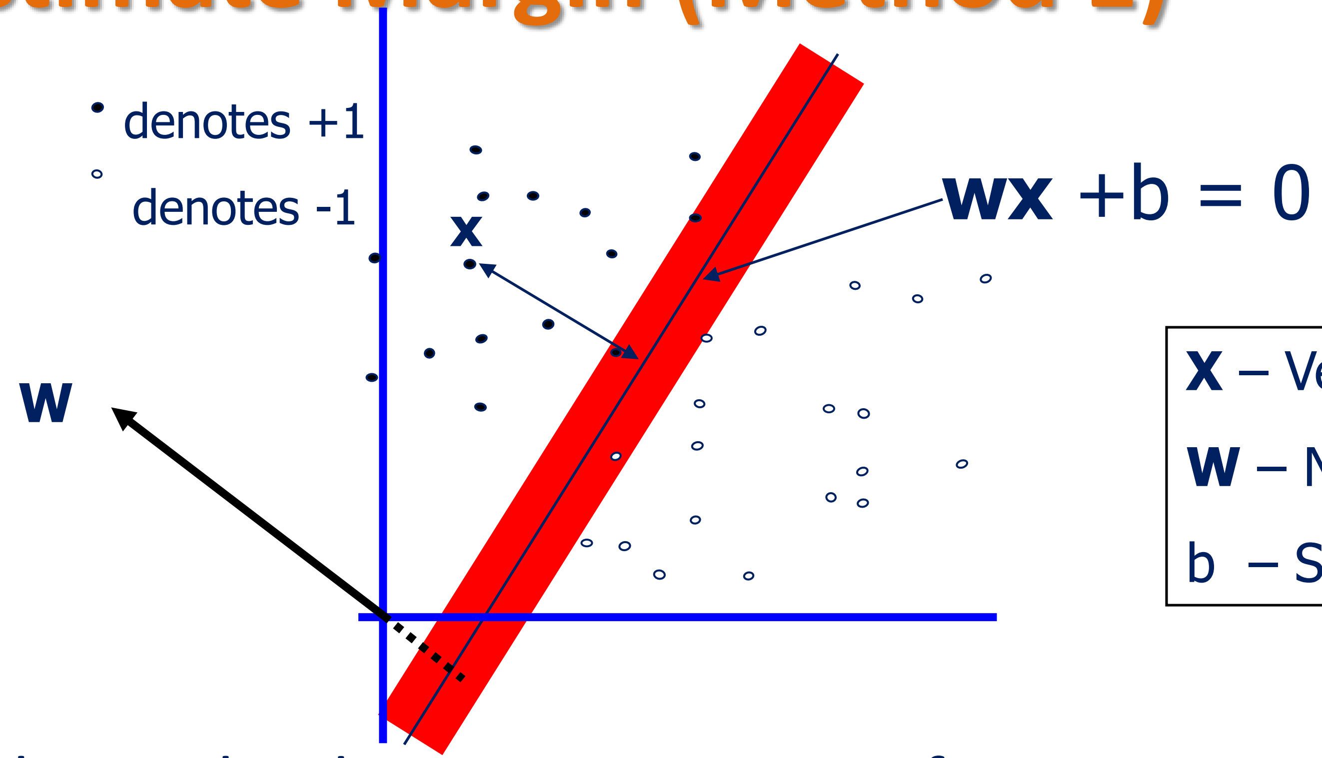
- • denotes +1
- ◦ denotes -1



Support Vectors are those datapoints that the margin pushes up against

1. **Intuitively this feels safest.**
2. **Empirically it works very well.**
3. If we've made a small error in the location of the boundary (it's been jolted in its perpendicular direction) this gives us least chance of causing a misclassification.
4. LOOCV is easy since the model is immune to removal of any non-support-vector datapoints.
5. There's some theory (using VC dimension) that this is a good thing.

Estimate Margin (Method 1)



x – Vector

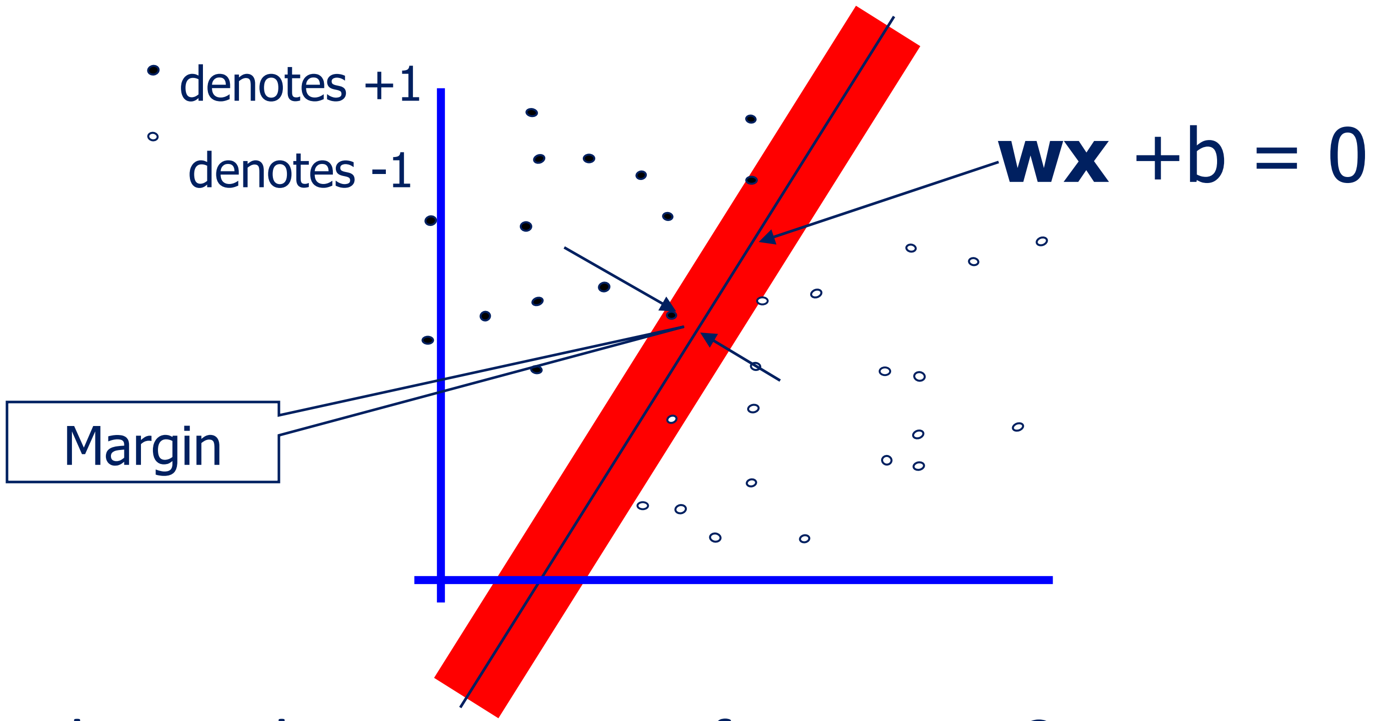
w – Normal Vector

b – Scale Value

- What is the distance expression for a point **x** to a line **wx**+b= 0?

$$d(\mathbf{x}) = \frac{|\mathbf{x}\cdot\mathbf{w}+b|}{\sqrt{\|\mathbf{w}\|_2^2}} = \frac{|\mathbf{x}\cdot\mathbf{w}+b|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

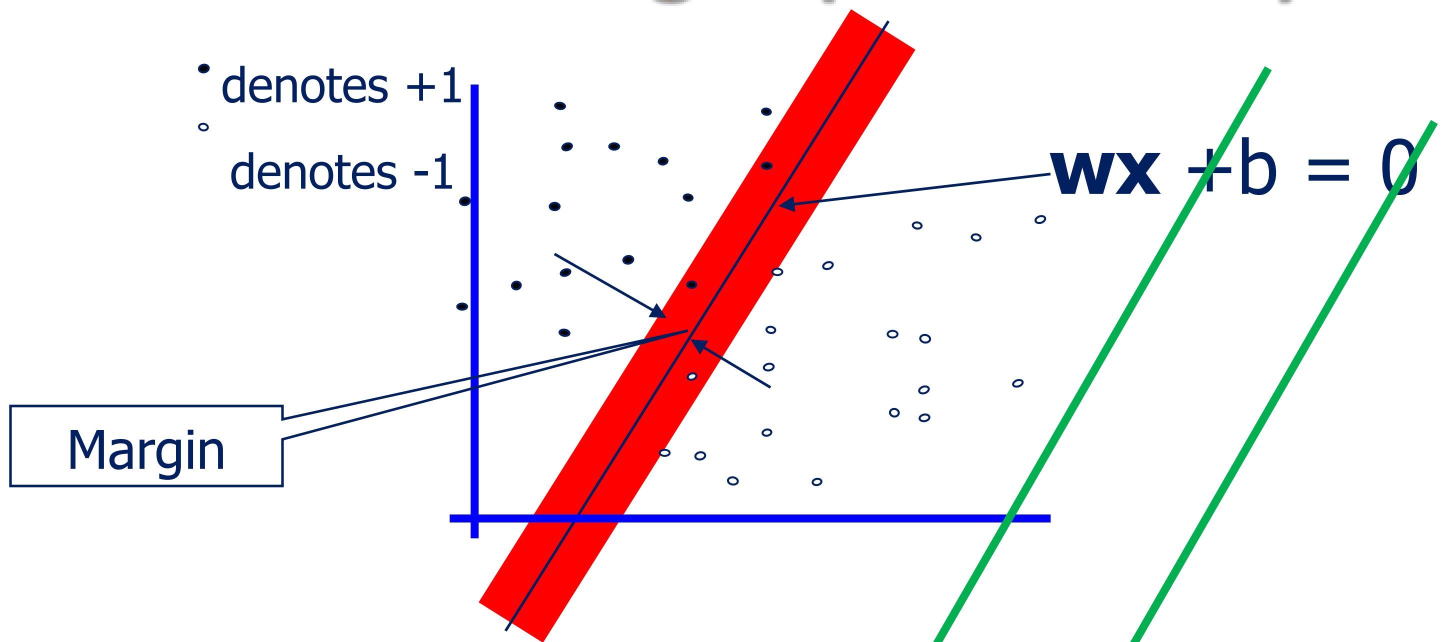
Estimate Margin (Method 1)



- What is the expression for margin?

$$\text{margin} \equiv \arg\min_{\mathbf{x}\in D} d(\mathbf{x}) = \arg\min_{\mathbf{x}\in D} \frac{|\mathbf{x}\cdot\mathbf{w}+b|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

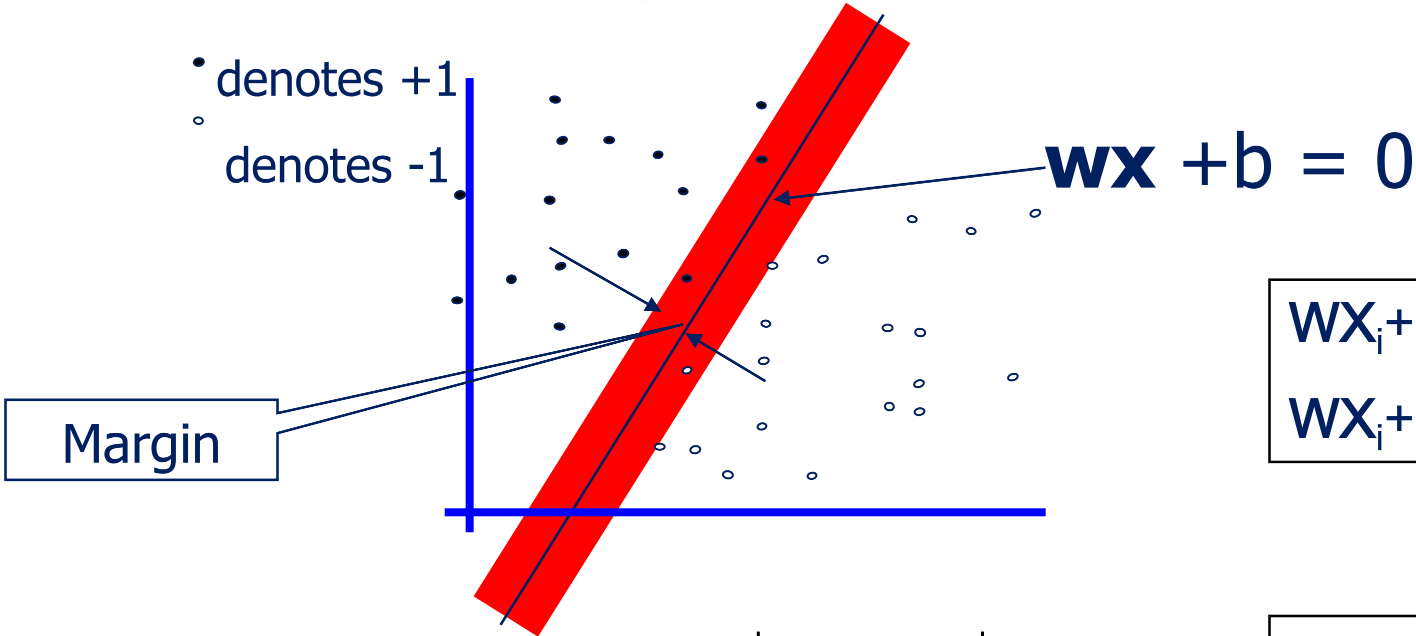
Estimate Margin (Method 1)



$$\begin{aligned} &\underset{\mathbf{w},b}{\text{argmax}}\ \ \text{margin}(\mathbf{w}, b, D) \\ &= \underset{\mathbf{w},b}{\text{argmax}}\ \underset{\mathbf{x}_i \in D}{\arg\min}\ d(\mathbf{x}_i) \\ &= \underset{\mathbf{w},b}{\text{argmax}}\ \underset{\mathbf{x}_i \in D}{\arg\min}\ \frac{\left|b + \mathbf{x}_i \cdot \mathbf{w}\right|}{\sqrt{\sum_{i=1}^{d} w_i^2}} \end{aligned}$$

Is this all?

Estimate Margin (Method 1)



$\mathbf{w}\mathbf{x}_i$+b≥0 iff y_i=1

$\mathbf{w}\mathbf{x}_i$+b≤0 iff y_i=-1

⇩

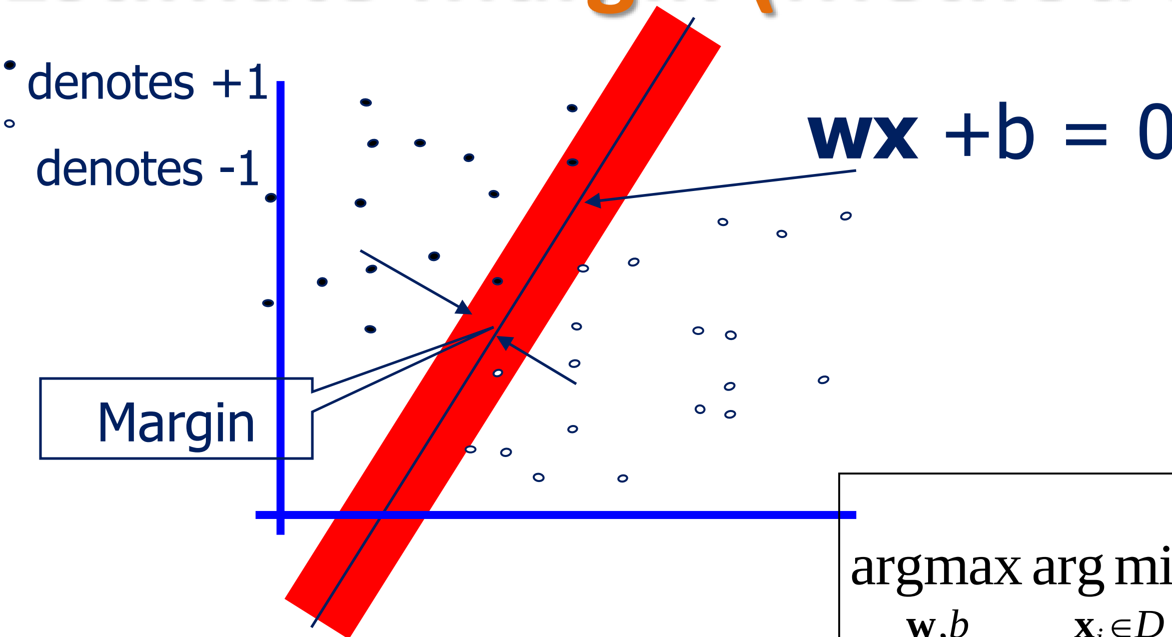
$y_i(\mathbf{w}\mathbf{x}_i+b) \geq 0$

$$\underset{\mathbf{w},b}{\operatorname{argmax}} \underset{\mathbf{x}_i \in D}{\arg\min} \frac{\left|b + \mathbf{x}_i \cdot \mathbf{w}\right|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

$$\text{subject to } \forall \mathbf{x}_i \in D : y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 0$$

- Min-max problem → game problem

Estimate Margin (Method 1)



$\mathbf{w}\mathbf{x}_i + b \geq 0$ iff $y_i = 1$

$\mathbf{w}\mathbf{x}_i + b \leq 0$ iff $y_i = -1$

$$\Downarrow$$

$y_i(\mathbf{w}\mathbf{x}_i + b) \geq 0$

$$\underset{\mathbf{w},b}{\operatorname{argmax}} \underset{\mathbf{x}_i \in D}{\arg\min} \frac{|b + \mathbf{x}_i \cdot \mathbf{w}|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

$$\text{subject to } \forall \mathbf{x}_i \in D : y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b \right) \geq 0$$

Strategy:

$$\forall \mathbf{x}_i \in D : \ |b + \mathbf{x}_i \cdot \mathbf{w}| \geq 1$$

$$\Downarrow$$

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{d} w_i^2$$

$$\text{subject to } \forall \mathbf{x}_i \in D : y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b \right) \geq 1$$

Estimate Margin (Method 1)

- How does it come ?

$$\underset{\mathbf{w},b}{\text{argmax}}\ \underset{\mathbf{x}_i \in D}{\arg\min} \frac{\left|b + \mathbf{x}_i \cdot \mathbf{w}\right|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

subject to $\forall \mathbf{x}_i \in D: y_i\left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 0$

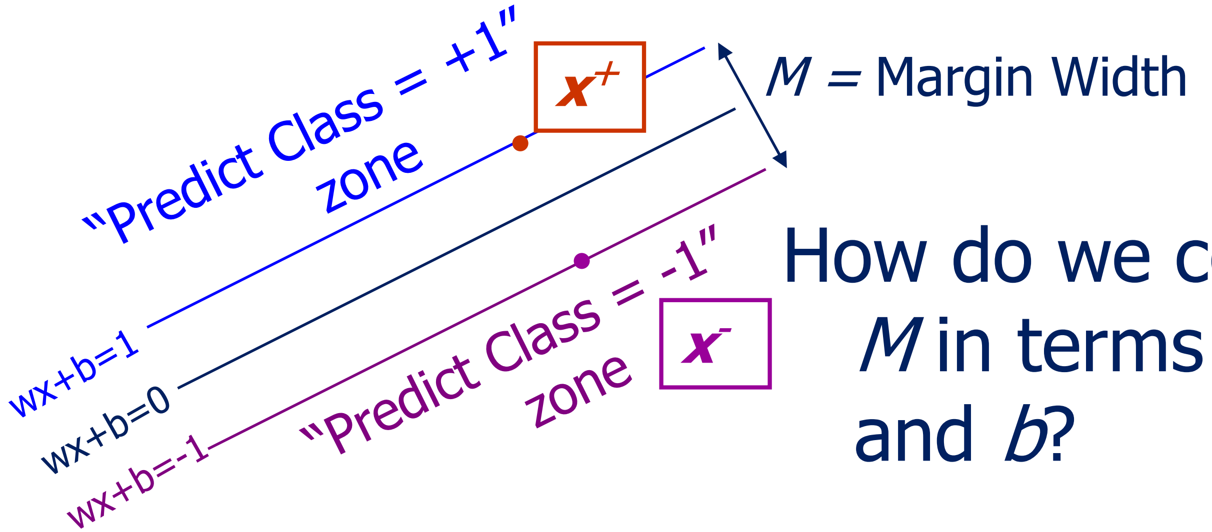
$$\forall \mathbf{x}_i \in D: \ \left|b + \mathbf{x}_i \cdot \mathbf{w}\right| \geq 1$$

$$\underset{\mathbf{w},b}{\text{argmin}} \sum_{i=1}^{d} w_i^2$$

subject to $\forall \mathbf{x}_i \in D: y_i\left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 1$

$$\underset{\mathbf{x}_i \in D}{\arg\min} \frac{\left|b + \mathbf{x}_i \cdot \mathbf{w}\right|}{\sqrt{\sum_{i=1}^{d} w_i^2}} \geq \underset{\mathbf{x}_i \in D}{\arg\min} \frac{1}{\sqrt{\sum_{i=1}^{d} w_i^2}} = \frac{1}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

Estimate Margin (Method 2)



How do we compute M in terms of $\boldsymbol{w}$ and b?

- Plus-plane = $\{ \boldsymbol{x} : \boldsymbol{w} . \boldsymbol{x} + b = +1 \}$
- Minus-plane = $\{ \boldsymbol{x} : \boldsymbol{w} . \boldsymbol{x} + b = -1 \}$
- What is the distance between these two planes?

Estimate Margin (Method 2)

- Margin can also be defined as distance between two parallel lines.

Given 2 parallel lines with equations

$$ax + by + c_1 = 0$$

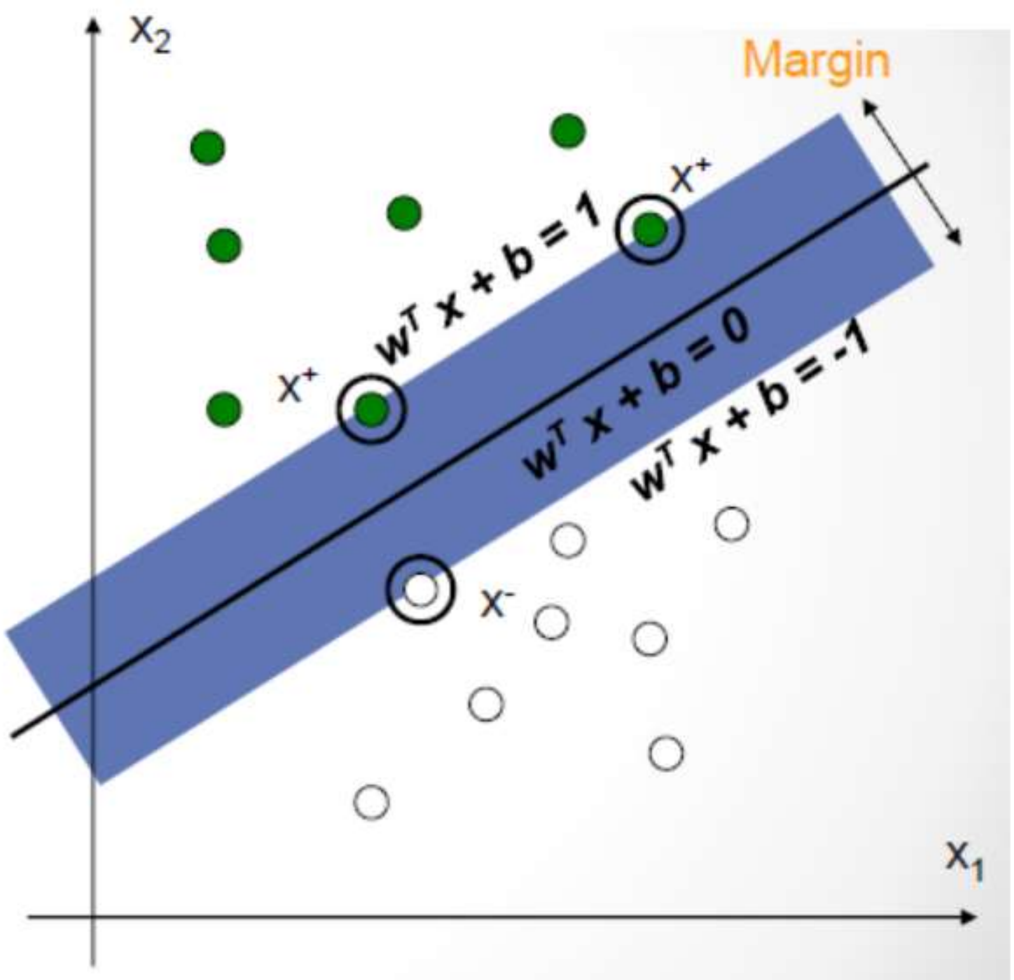
and

$$ax + by + c_2 = 0$$

the distance between them is given by:

$$d = \frac{|c_2 - c_1|}{\sqrt{a^2 + b^2}}$$

$$\frac{2}{\|\mathbf{w}\|}$$



Maximize Margin

$$\text{maximize} \quad \frac{2}{\|\mathbf{w}\|}$$

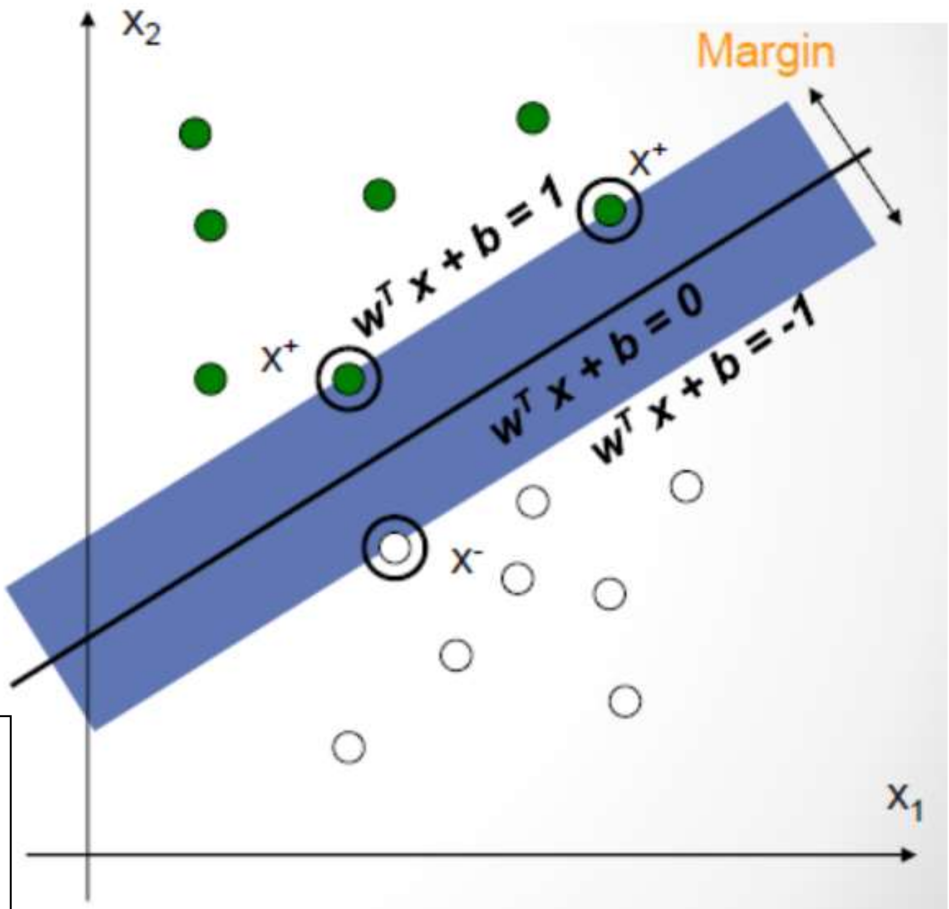
such that

For $y_i = +1, \quad \mathbf{w}^T \mathbf{x}_i + b \geq 1$

For $y_i = -1, \quad \mathbf{w}^T \mathbf{x}_i + b \leq -1$

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{d} w_i^2$$

$$\text{subject to} \quad \forall \mathbf{x}_i \in D : y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b \right) \geq 1$$



Quadratic Programming

Find $\arg\max_{\mathbf{u}} \quad c + \mathbf{d}^T\mathbf{u} + \frac{\mathbf{u}^T R\mathbf{u}}{2}$ Quadratic criterion

Subject to

$$\left.\begin{aligned}
a_{11}u_1 + a_{12}u_2 + \ldots + a_{1m}u_m &\le b_1 \\
a_{21}u_1 + a_{22}u_2 + \ldots + a_{2m}u_m &\le b_2 \\
&\vdots \\
a_{n1}u_1 + a_{n2}u_2 + \ldots + a_{nm}u_m &\le b_n
\end{aligned}\right\}$$

n additional linear inequality constraints

And subject to

$$\left.\begin{aligned}
a_{(n+1)1}u_1 + a_{(n+1)2}u_2 + \ldots + a_{(n+1)m}u_m &= b_{(n+1)} \\
a_{(n+2)1}u_1 + a_{(n+2)2}u_2 + \ldots + a_{(n+2)m}u_m &= b_{(n+2)} \\
&\vdots \\
a_{(n+e)1}u_1 + a_{(n+e)2}u_2 + \ldots + a_{(n+e)m}u_m &= b_{(n+e)}
\end{aligned}\right\}$$

e additional linear equality constraints

Quadratic Programming for Linear SVM

$$\{\vec{w}^*, b^*\} = \min_{\vec{w},b} \sum_i w_i^2$$

subject to $y_i\left(\vec{w}\cdot\vec{x}_i + b\right) \geq 1$ for all training data $(\vec{x}_i, y_i)$

$$\{\vec{w}^*, b^*\} = \underset{\vec{w},b}{\operatorname{argmax}} \left\{0 + \vec{0}\cdot\vec{w} - \vec{w}^T \mathbf{I_n} \vec{w}\right\}$$

$$\left.\begin{array}{l} y_1\left(\vec{w}\cdot\vec{x}_1 + b\right) \geq 1 \\ y_2\left(\vec{w}\cdot\vec{x}_2 + b\right) \geq 1 \\ \ldots. \\ y_N\left(\vec{w}\cdot\vec{x}_N + b\right) \geq 1 \end{array}\right\} \text{inequality constraints}$$

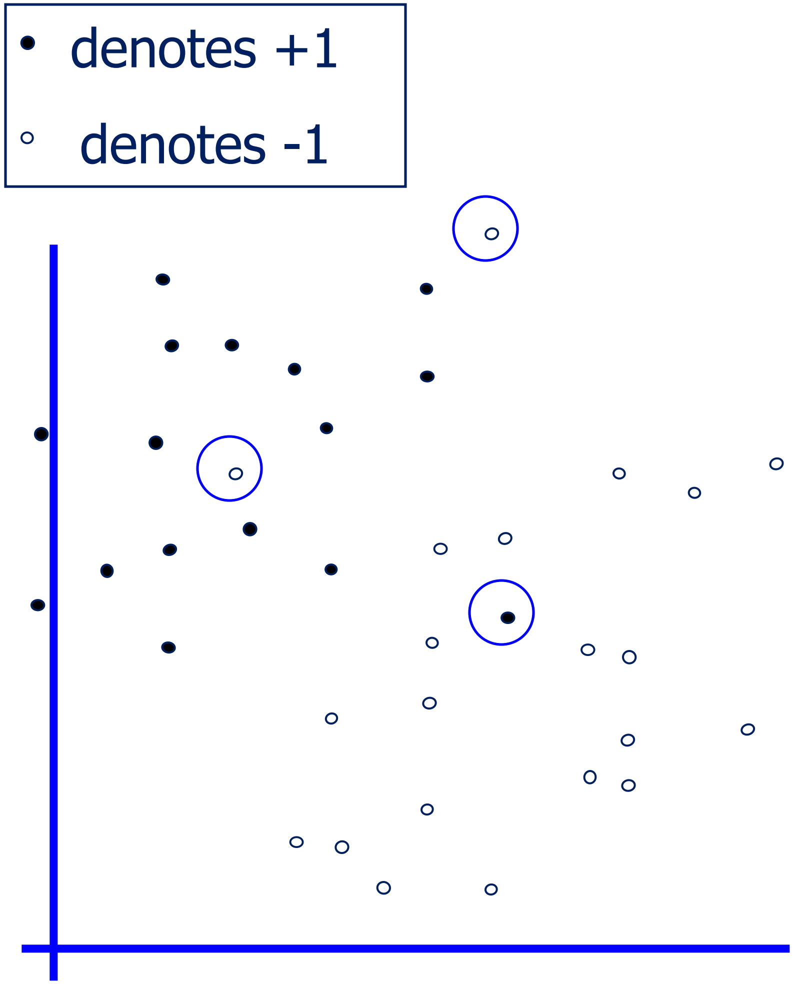
Soft-Margin SVM

Noisy Data

Not linear separable

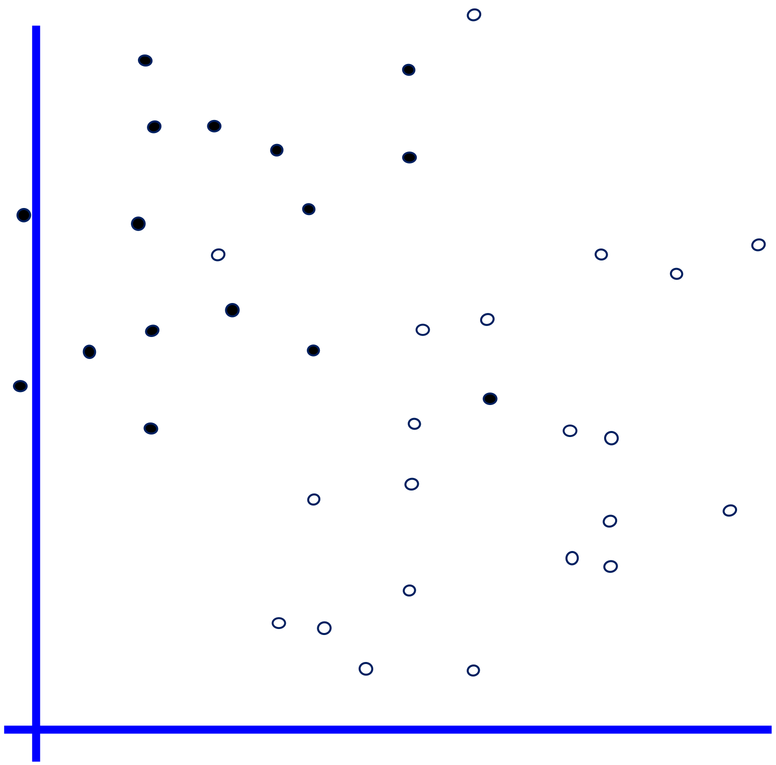
This is going to be a problem!

What should we do?



Noisy Data

- • denotes +1
- ○ denotes -1



This is going to be a problem!

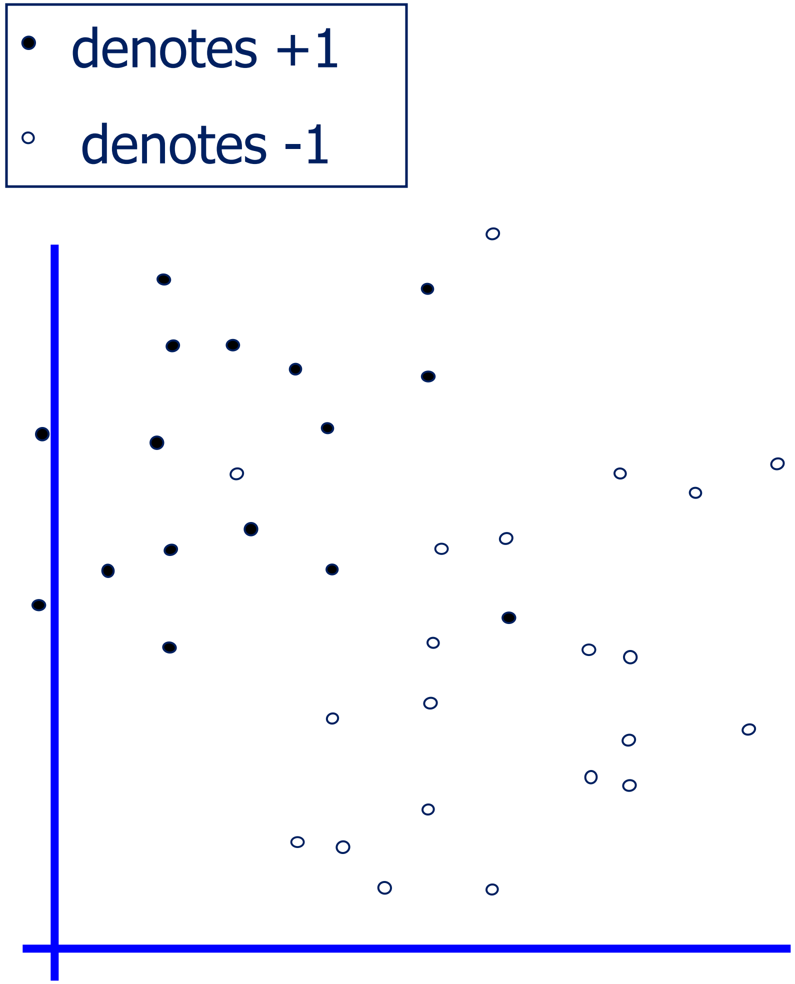
What should we do?

Idea 1:

Find minimum ***w.w***, while minimizing number of training set errors.

Problemette: Two things to minimize makes for an ill-defined optimization

Noisy Data



This is going to be a problem!

What should we do?

Idea 1.1:

Minimize

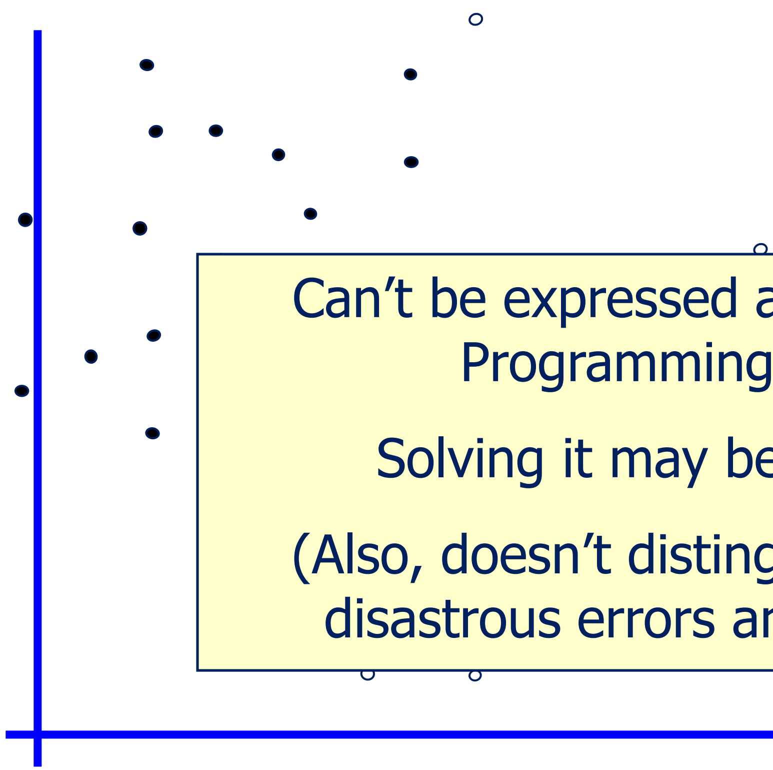
$\boldsymbol{w.w} + C$ *(#train errors)*

Tradeoff parameter

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

Noisy Data

- • denotes +1
- ◦ denotes -1



This is going to be a problem!

What should we do?

Idea 1.1:

Minimize

$\boldsymbol{w.w} + C\,(\#train\ errors)$

Tradeoff parameter

Can't be expressed as a Quadratic Programming problem.

Solving it may be too slow.

(Also, doesn't distinguish between disastrous errors and near misses)

So… any other ideas?

us reject this

you guess what

Noisy Data

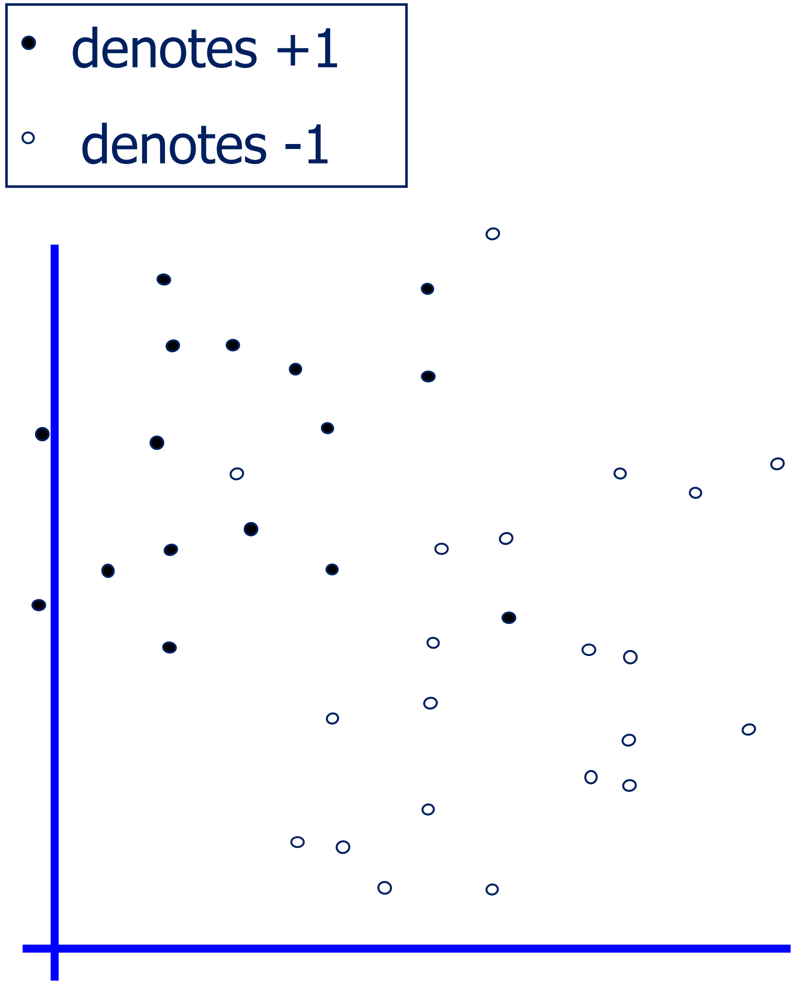
This is going to be a problem!

What should we do?

Idea 2.0:

Minimize

$\boldsymbol{w.w}$ + *C (distance of error points to their correct place)*



SVM for Noisy Data

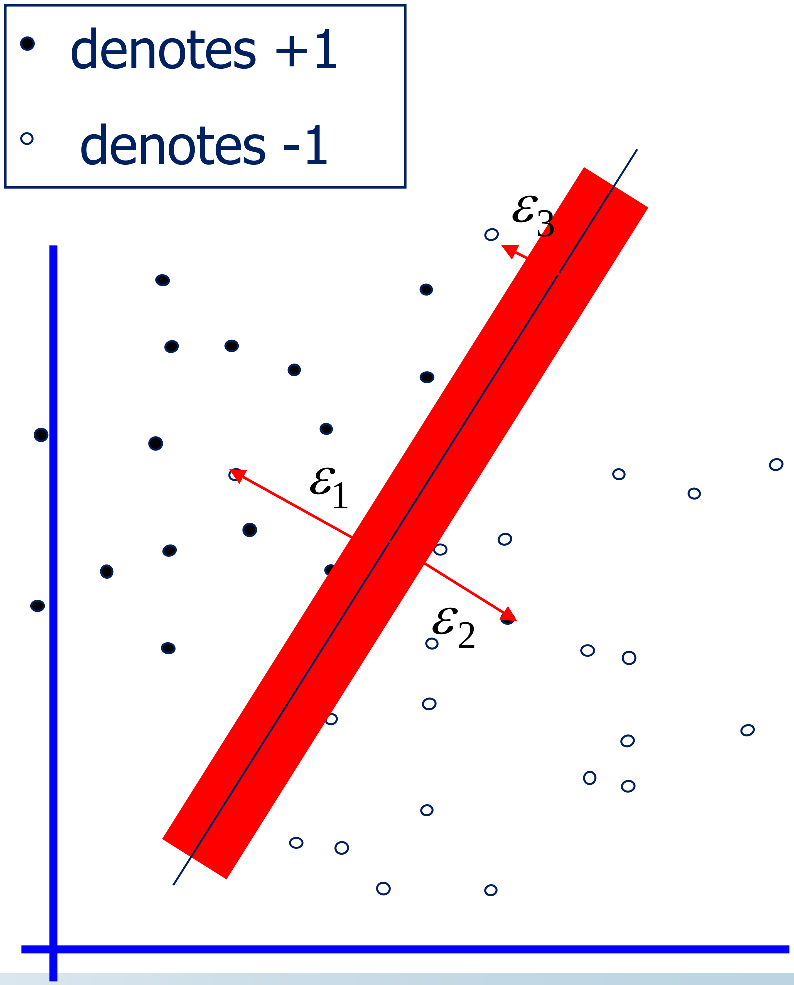
$$\{\vec{w}^*, b^*\} = \min_{\vec{w},b} \sum_{i=1}^{d} w_i^2 + c\sum_{j=1}^{N} \varepsilon_j$$

$$y_1\left(\vec{w}\cdot\vec{x}_1 + b\right) \geq 1 - \varepsilon_1$$

$$y_2\left(\vec{w}\cdot\vec{x}_2 + b\right) \geq 1 - \varepsilon_2$$

$$\ldots$$

$$y_N\left(\vec{w}\cdot\vec{x}_N + b\right) \geq 1 - \varepsilon_N$$



- Any problem with the above formulism?

What happens when $\varepsilon_i < 0$?

SVM for Noisy Data

$$\{\vec{w}^*, b^*\} = \min_{\vec{w},b} \sum_{i=1}^{d} w_i^2 + c\sum_{j=1}^{N} \varepsilon_j$$

$$y_1\left(\vec{w}\cdot\vec{x}_1 + b\right) \geq 1-\varepsilon_1, \boxed{\varepsilon_1 \geq 0}$$

$$y_2\left(\vec{w}\cdot\vec{x}_2 + b\right) \geq 1-\varepsilon_2, \boxed{\varepsilon_2 \geq 0}$$

$$\ldots$$

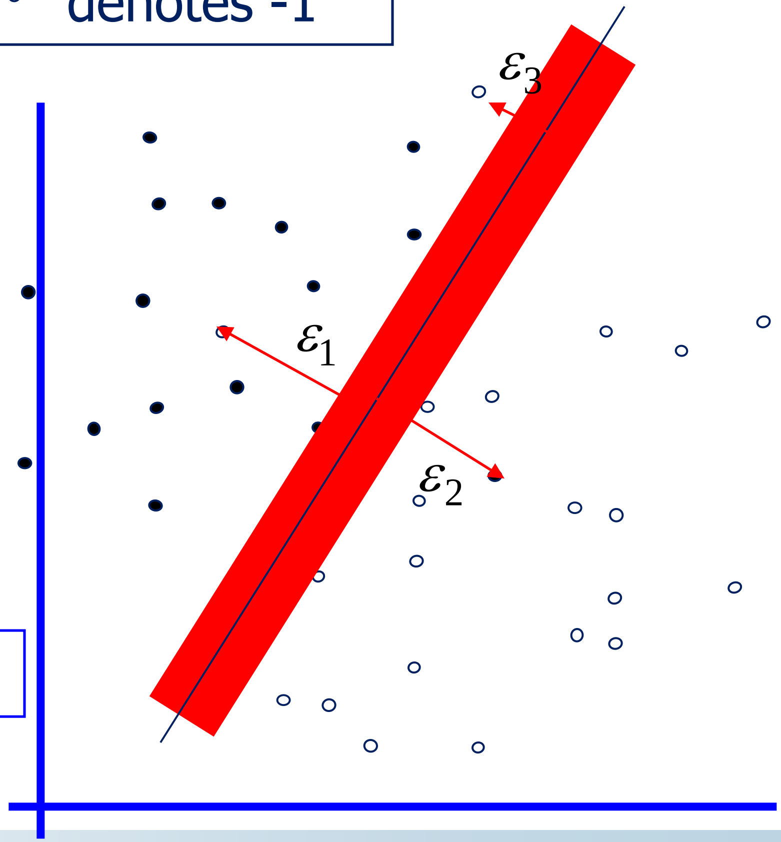
$$y_N\left(\vec{w}\cdot\vec{x}_N + b\right) \geq 1-\varepsilon_N, \boxed{\varepsilon_N \geq 0}$$

- Balance the trade off between margin and classification errors

Describe the Theory

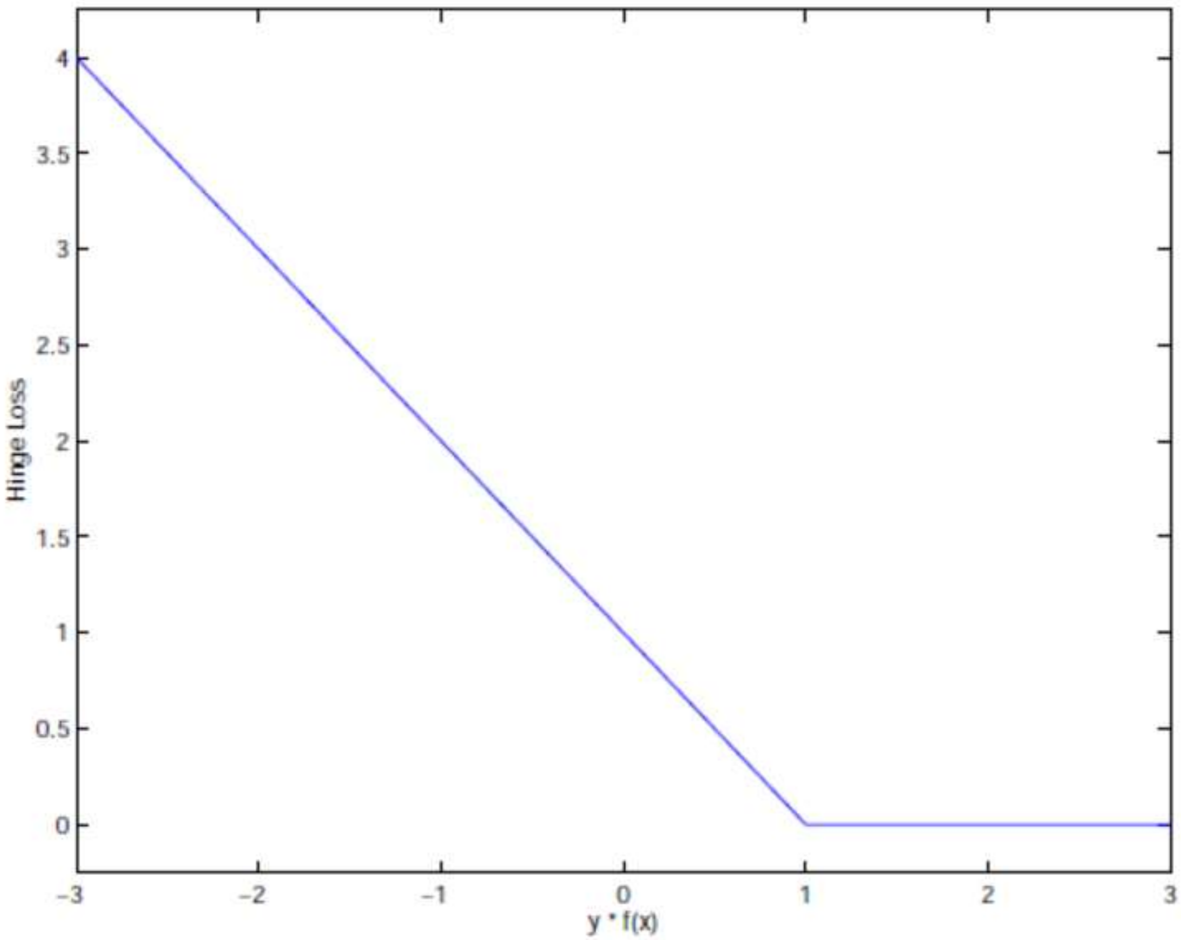
Describe the Mistake

- denotes +1
- denotes -1

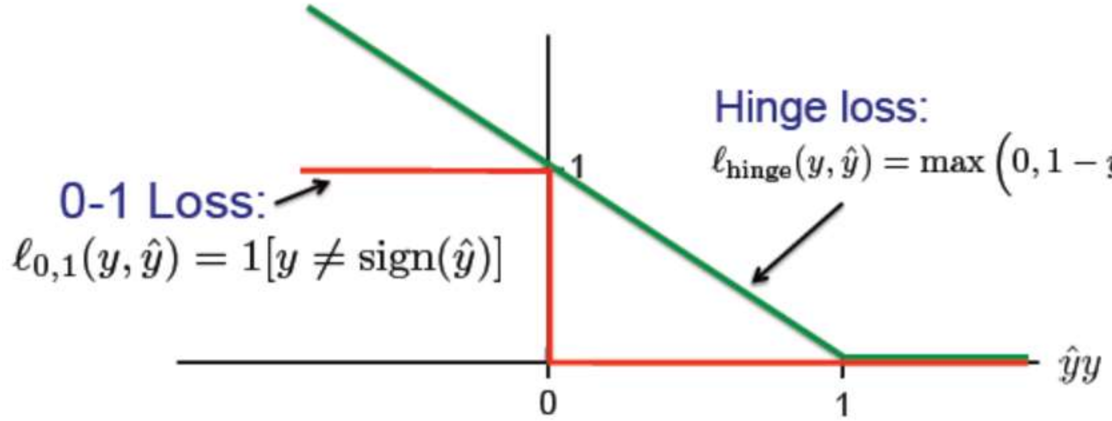


Hinge Loss

$$hinge(x) = max(1 - x, 0)$$



Hinge Loss



Hinge loss upper bounds 0/1 loss!

It is the tightest *convex* upper bound on the 0/1 loss

- The SVM is a Tikhonov regularization problem, using the hinge loss:

$$\operatorname*{argmin}_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (1 - y_i f(x_i))_+ + \lambda \|f\|_{\mathcal{H}}^2.$$

Conclusion so far

- Which linear classifier → large margin
- How to calculate margin → $\frac{2}{\|\mathbf{w}\|}$
- Hard margin SVM → QP
- Not linear separable → soft-margin SVM
- Still QP
- L2 norm regularization + hinge loss

Dual Problem

Lagrange Multipliers

$$Minimize\ f(x)$$

$$subject\ to\quad \begin{cases} a(x) \geq 0 \\ b(x) \leq 0 \\ c(x) = 0 \end{cases}$$

$$L(x, \alpha) = f(x) - \alpha_1 a(x) - \alpha_2 b(x) - \alpha_3 c(x)$$

$$\begin{cases} \alpha_1 \geq 0 \\ \alpha_2 \leq 0 \\ \alpha_3\ is\ unconstrained \end{cases}$$

We can recover the primal problem by maximizing the Lagrangian with respect to the Lagrange multipliers

$$\max_{\alpha} L(x, \alpha) = \begin{cases} f(x), & if \begin{cases} a(x) \geq 0 \\ b(x) \leq 0 \\ c(x) = 0 \end{cases} \\ +\infty, & otherwise \end{cases}$$

So, the Primal problem can be changed into Dual problem

$$\min_{x} \boxed{\underbrace{\max_{\alpha} L(x, \alpha)}_{Primal(x)}} = \max_{\alpha} \boxed{\underbrace{\min_{x} L(x, \alpha)}_{Dual(\alpha)}}$$

Karush-Kuhn-Tucker conditions

- For a local minimum

$$
\begin{cases}
\mathit{Stationarity} \quad \nabla f(x^*) - \alpha_1 \nabla a(x^*) - \alpha_2 \nabla b(x^*) - \alpha_3 \nabla c(x^*) = 0 \\
\mathit{Primal\ feasibility} \quad \begin{cases} a(x^*) \geq 0 \\ b(x^*) \leq 0 \\ c(x^*) = 0 \end{cases} \\
\mathit{Dual\ feasibility} \quad \begin{cases} \alpha_1 \geq 0 \\ \alpha_2 \leq 0 \\ \alpha_3 \text{ is unconstrained} \end{cases} \\
\mathit{Complementary\ slackness} \quad \begin{cases} \alpha_1 a(x^*) = 0 \\ \alpha_2 b(x^*) = 0 \\ \alpha_3 c(x^*) = 0 \end{cases}
\end{cases}
$$

Lagrange Transformation

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to } 1-y_i(\mathbf{w}^T\mathbf{x}_i+b) \le 0 \qquad \text{for } i = 1, \ldots, n$$

- The Lagrangian is

Lagrangian multipliers

$$\mathcal{L} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \alpha_i \left(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\right)$$

- Setting the gradient of ***L*** w.r.t. **w** and b to zero, we have

$$\mathbf{w} + \sum_{i=1}^{n} \alpha_i(-y_i)\mathbf{x}_i = \mathbf{0} \quad \Rightarrow \quad \boxed{\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i}$$

$$\boxed{\sum_{i=1}^{n} \alpha_i y_i = 0}$$

$\alpha_i \geq 0$

Dual Problem

- We can transform the problem to its dual

Dot product of X

$$\max.\ W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0,\ \sum_{i=1}^{n} \alpha_i y_i = 0$$

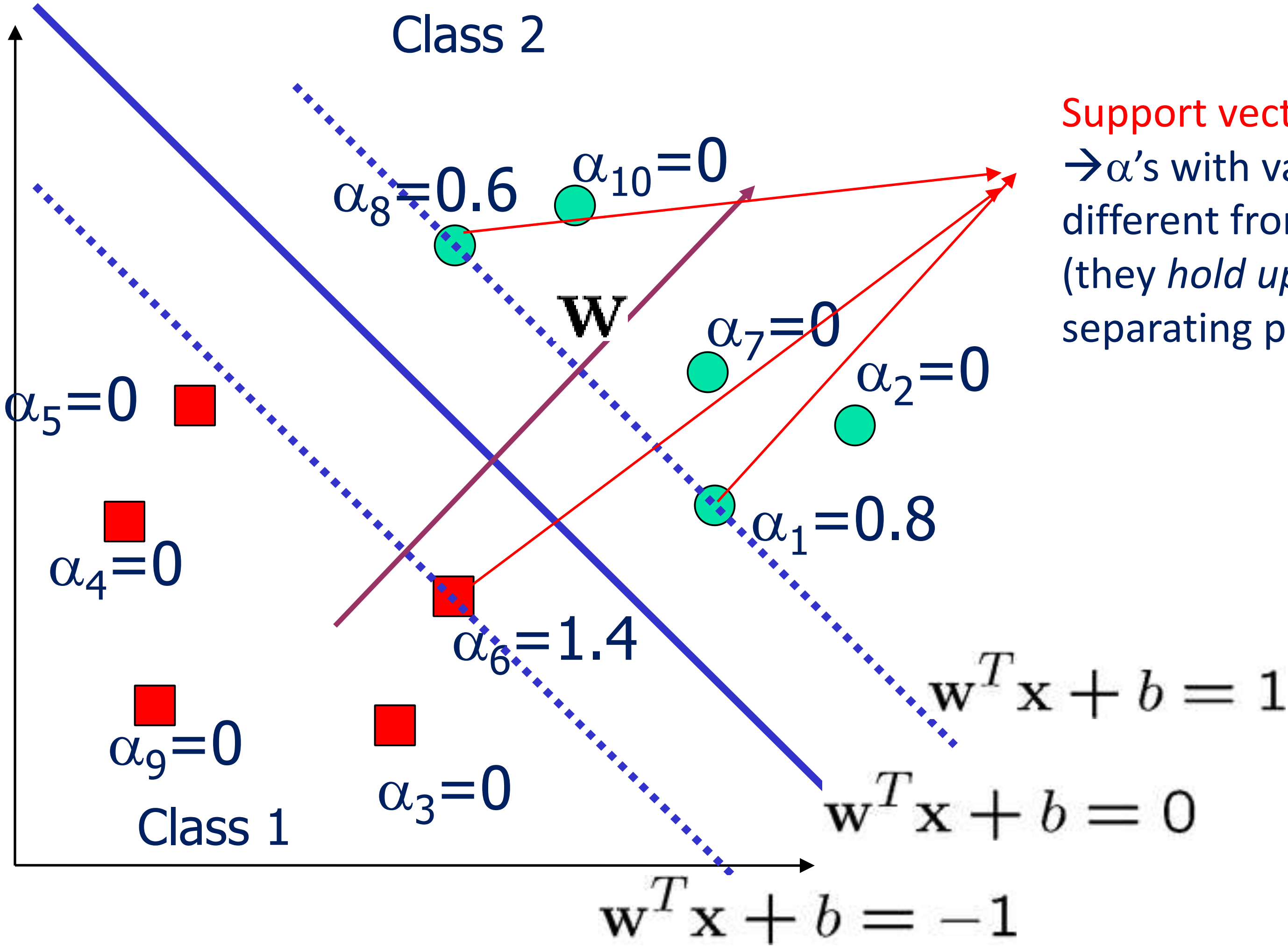
α's → New variables (Lagrangian multipliers)

- This is a convex quadratic programming (QP) problem
  - Global maximum of α_i can always be found

  → well established tools for solving this optimization problem
- **Note:**

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

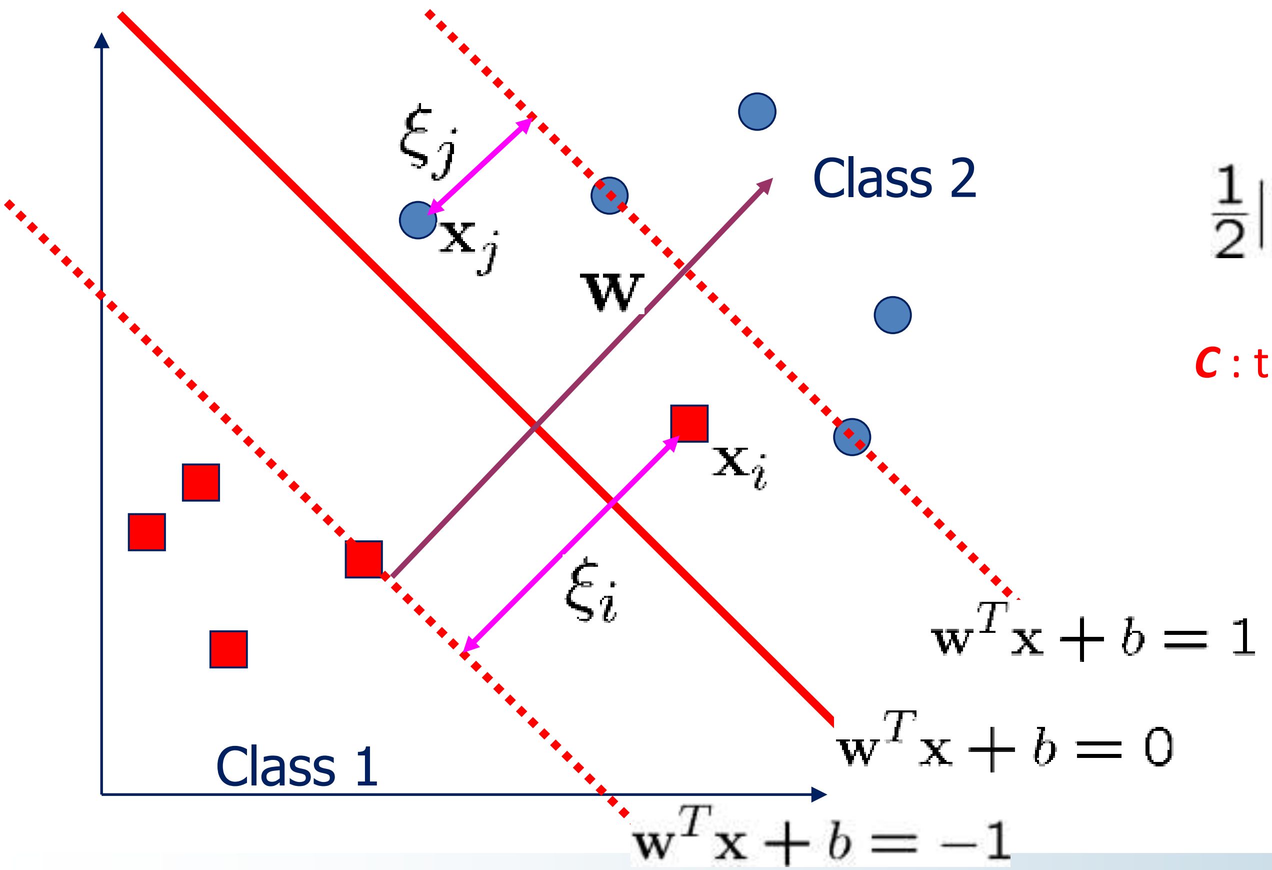
Support Vectors



Support vectors
→α's with values different from zero (they *hold up* the separating plane)!

Soft-Margin Case

- We allow "error" ξ_i in classification; it is based on the output of the discriminant function $\mathbf{w}^T\mathbf{x}$+b
- ξ_i approximates the number of misclassified samples



New objective function:

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

C : tradeoff parameter between error and margin; chosen by the user; large C means a higher penalty to errors

Soft-Margin Case

- Lagrangian Problem

$$L = \frac{1}{2}\|w\|^2 + \ C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i(y_i(w^T x_i + b) - 1 + \xi_i) - \sum_{i=1}^{n}\gamma_i\xi_i$$

$\alpha_i \geq 0 \quad \gamma_i \geq 0$

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_i \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_i \alpha_i y_i = 0$$

$$\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow \alpha_i + \gamma_i = C \Rightarrow 0 \leq \alpha_i \leq C$$

Dual Problem for Soft-Margin SVM

- The dual of the problem is

$$\max.\ W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \boxed{\mathbf{x}_i^T \mathbf{x}_j}$$

$$\text{subject to } \boxed{C \geq \alpha_i \geq 0,} \sum_{i=1}^{n} \alpha_i y_i = 0$$

- **w** is also recovered as $\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$
- The only difference with the linear separable case is that there is an upper bound *C* on α_i
- Once again, a QP solver can be used to find α_i efficiently!!!

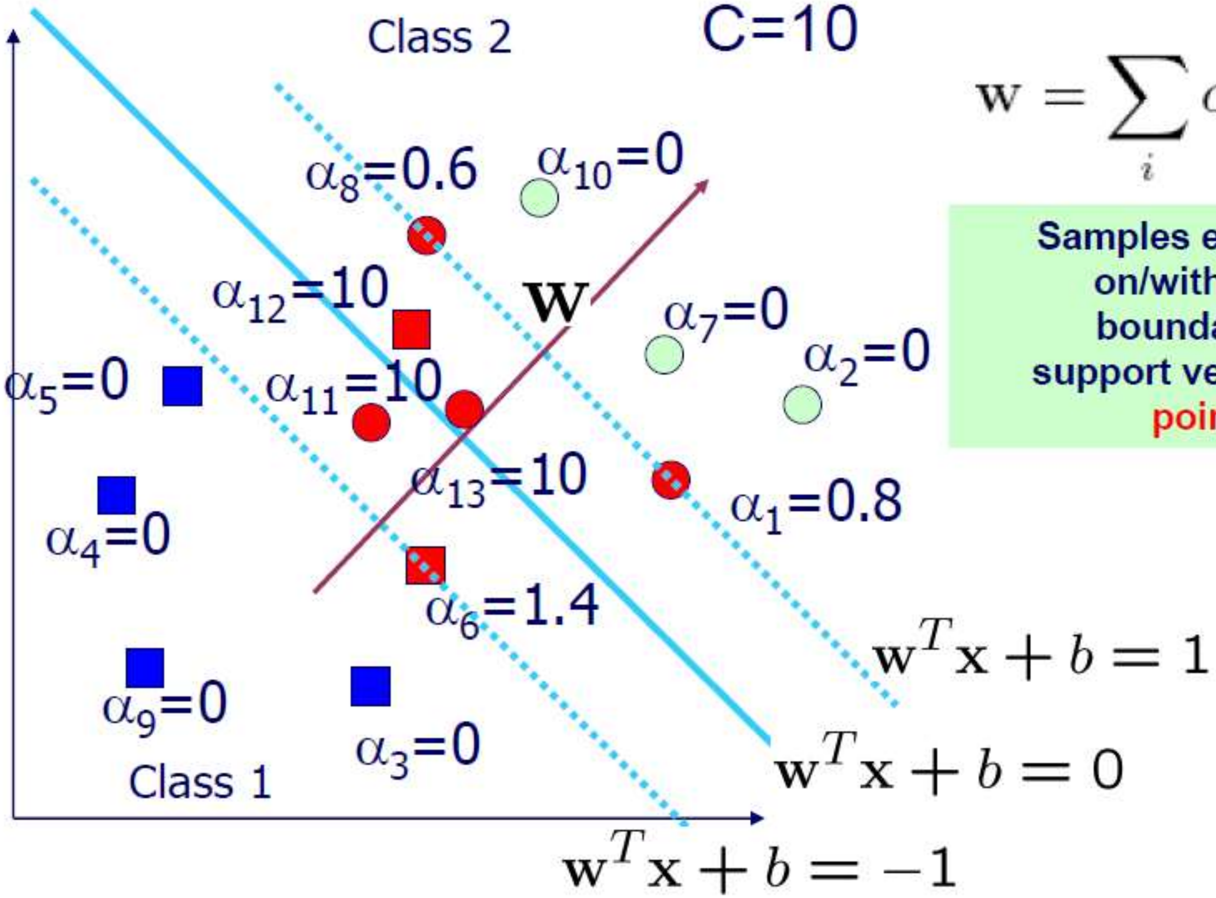
KKT conditions

- When optimum is achieved, KKT conditions are satisfied

$$\begin{cases} \alpha_i(y_i f(x_i) - 1 + \xi_i) = 0 \\ \gamma_i \xi_i = 0 \\ \alpha_i + \gamma_i = C \Rightarrow 0 \leq \alpha_i \leq C \end{cases}$$

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

Support Vectors



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

Samples exactly on/within the boundary are support vectors (red points)

Finding the bias b

- Find the bias *b* based on

$$0 < \alpha_i < C \quad \Rightarrow \quad y_i f(\mathrm{x}_i) = 1$$

$$f(\mathbf{z}) = \sum_{j=1}^{s} \alpha_j y_j \mathbf{x}_j^T \mathbf{z} + b$$

$$\Downarrow$$

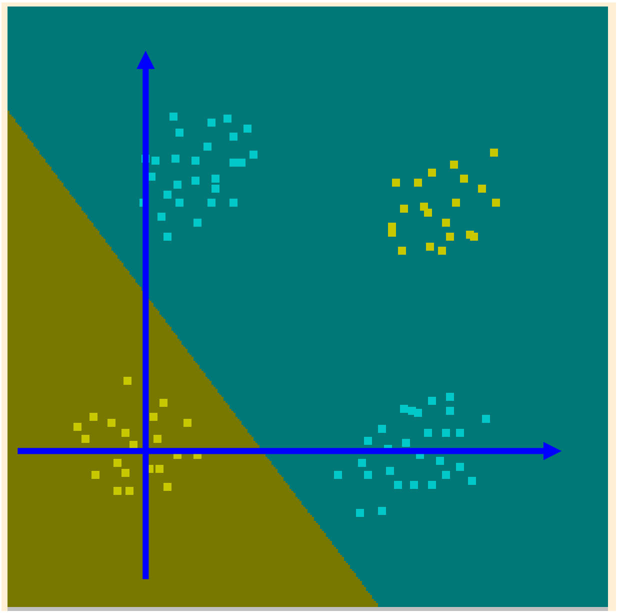
$$\boxed{b = 1 - \frac{1}{|i : 0 < \alpha_i < C|} \sum_{i:0<\alpha_i<C} \sum_{j=1}^{s} \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i}$$

$$b = y_i - \sum_{j=1}^{s} \alpha_j y_j x_j^T x_i \qquad \forall 0 < \alpha_i < C$$

Kernel Methods

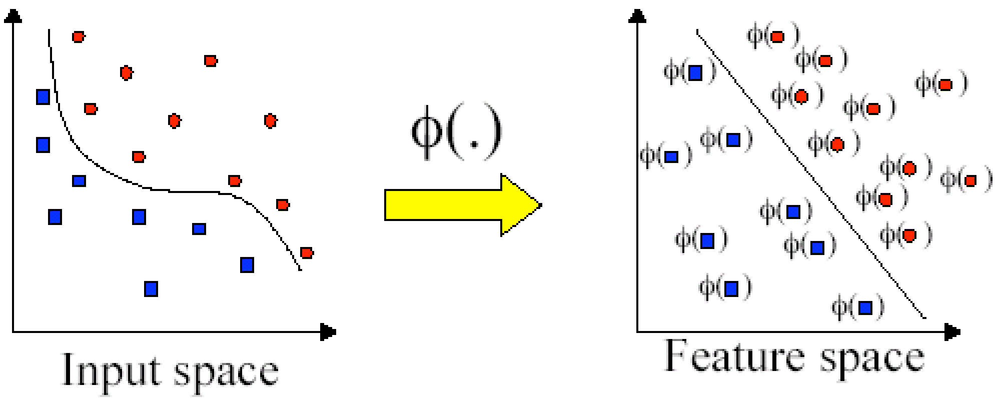
Feature Transformation ?

- The problem is non-linear
- Find some trick to transform the input
- Linear separable after Feature Transformation
- What Features should we use ?



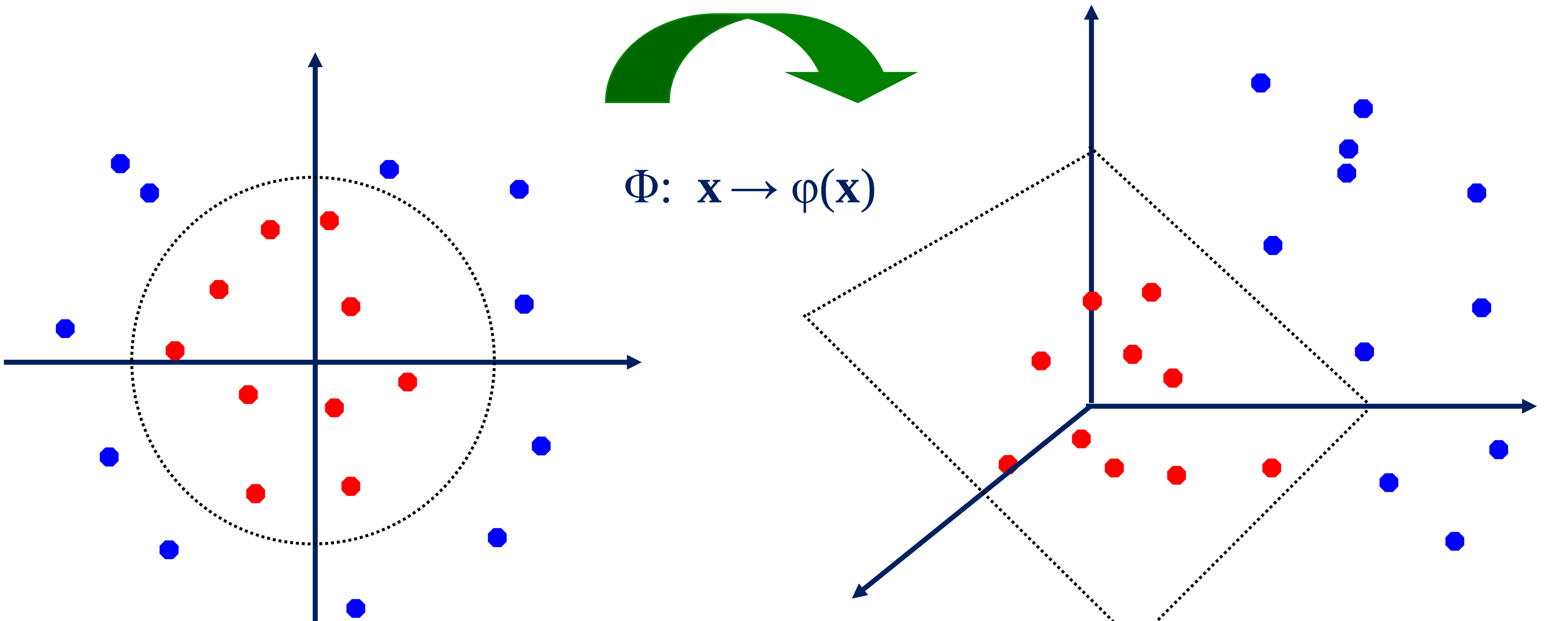
XOR Problem

Basic Idea :



Non-linear SVMs: Feature spaces

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



$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$$

Kernel Trick

Recall:

$$\text{maximize} \quad \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=j=1}^{N} \alpha_i \alpha_j y_i y_j x_i x_j$$

$$\text{subject to} \quad C \geq \alpha_i \geq 0, \sum_{i=1}^{N} \alpha_i y_i = 0$$

Note that data only appears as dot products

Since data is only represented as dot products, we need not do the mapping explicitly.

Introduce a Kernel Function *K* such that:

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

Kernel function – a function that can be applied to pairs of input data to evaluate dot products in some corresponding feature space

Example Transform

- Consider the following transformation

$$\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$
$$\phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1y_2)$$

- Define the kernel function *K* (**x**,**y**) as

$$\begin{aligned} \langle \phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}), \phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) \rangle &= (1 + x_1y_1 + x_2y_2)^2 \\ &= K(\mathbf{x}, \mathbf{y}) \end{aligned}$$

$$K(\mathbf{x}, \mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

- The inner product ϕ(.)ϕ(.) can be computed by *K* without going through the map ϕ(.) explicitly!!!

Examples of Kernel Function

- Polynomial kernel with degree *d*

$$K(\mathrm{x},\mathrm{y}) = (\mathrm{x}^T\mathrm{y}+1)^d$$

- Radial basis function kernel with width σ

$$K(\mathrm{x},\mathrm{y}) = \exp(-||\mathrm{x}-\mathrm{y}||^2/(2\sigma^2))$$

  – Closely related to radial basis function neural networks
- Sigmoid with parameter κ and θ

$$K(\mathrm{x},\mathrm{y}) = \tanh(\kappa\mathrm{x}^T\mathrm{y}+\theta)$$

  – It does not satisfy the Mercer condition on all κ and θ
- Research on different kernel functions in different applications is very active

Modification Due to Kernel Function

- Change **all inner products to kernel functions**
- For training,

Original

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \boxed{\mathbf{x}_i^T \mathbf{x}_j}$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

With kernel function

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \boxed{K(\mathbf{x}_i, \mathbf{x}_j)}$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

Modification Due to Kernel Function

- For testing, the new data ***z*** is classified as ***class 1 if f>=0*** and as ***class 2 if f<0***

Original

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

$$f = \mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \boxed{\mathbf{x}_{t_j}^T \mathbf{z}} + b$$

With kernel function

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \phi(\mathbf{x}_{t_j})$$

$$f = \langle \mathbf{w}, \phi(\mathbf{z}) \rangle + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \boxed{K(\mathbf{x}_{t_j}, \mathbf{z})} + b$$

Modification Due to Kernel Function

- Find the bias ***b***

Original

$$\boxed{b = 1 - \frac{1}{|i : 0 < \alpha_i < C|} \sum_{i:0<\alpha_i<C} \sum_{j=1}^{s} \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i}$$

With kernel function

$$\boxed{b = 1 - \frac{1}{|i : 0 < \alpha_i < C|} \sum_{i:0<\alpha_i<C} \sum_{j=1}^{s} \alpha_j y_j k(\mathbf{x}_j, \mathbf{x}_i)}$$

Example

- Suppose we have 5 1D data points
  - $x_1=1$, $x_2=2$, $x_3=4$, $x_4=5$, $x_5=6$, with 1, 2, 6 as class 1 and 4, 5 as class 2 $\Rightarrow$ $y_1=1$, $y_2=1$, $y_3=-1$, $y_4=-1$, $y_5=1$
- We use the polynomial kernel of degree 2
  - $K(x,y) = (xy+1)^2$
  - C is set to 100
- We first find α_i (i=1, …, 5) by

$$\text{max.} \quad \sum_{i=1}^{5} \alpha_i - \frac{1}{2} \sum_{i=1}^{5} \sum_{j=1}^{5} \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

$$\text{subject to } 100 \geq \alpha_i \geq 0, \sum_{i=1}^{5} \alpha_i y_i = 0$$

Example

- By using a QP solver, we get

  $\alpha_1=0$, $\alpha_2=2.5$, $\alpha_3=0$, $\alpha_4=7.333$, $\alpha_5=4.833$

  – Verify that the constraints are indeed satisfied

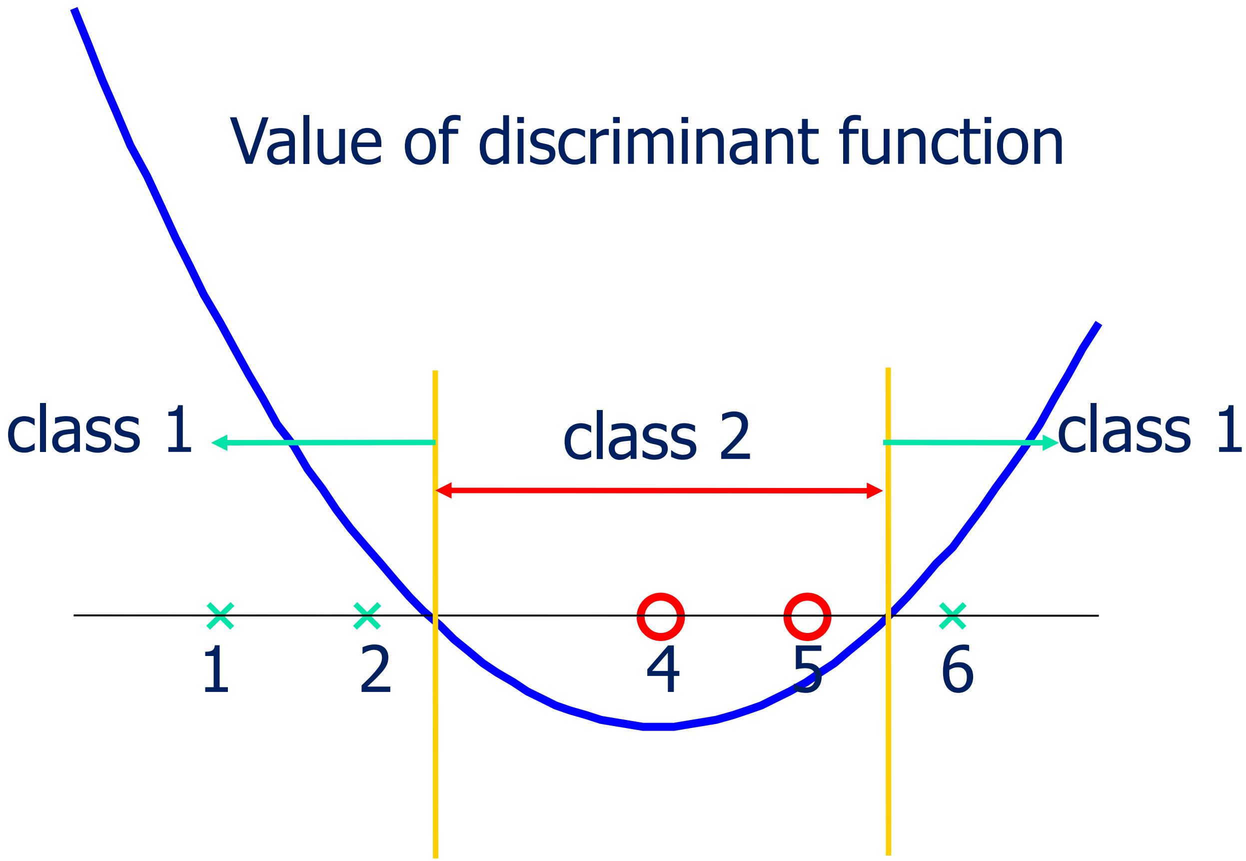
  – The support vectors are $\{x_2=2, x_4=5, x_5=6\}$
- The discriminant function is

$$
\begin{aligned}
f(y) &= 2.5(1)(2y+1)^2 + 7.333(-1)(5y+1)^2 + 4.833(1)(6y+1)^2 + b \\
&= 0.6667x^2 - 5.333x + b
\end{aligned}
$$

- *b* is recovered by solving $f(2)=1$ or by $f(5)=-1$ or by $f(6)=1$, as x_2, x_4, x_5 lie on $y_i(\mathbf{w}^T\phi(z)+b)=1$ and all give $b=9$

$$\Rightarrow \quad f(y) = 0.6667x^2 - 5.333x + 9$$

Example

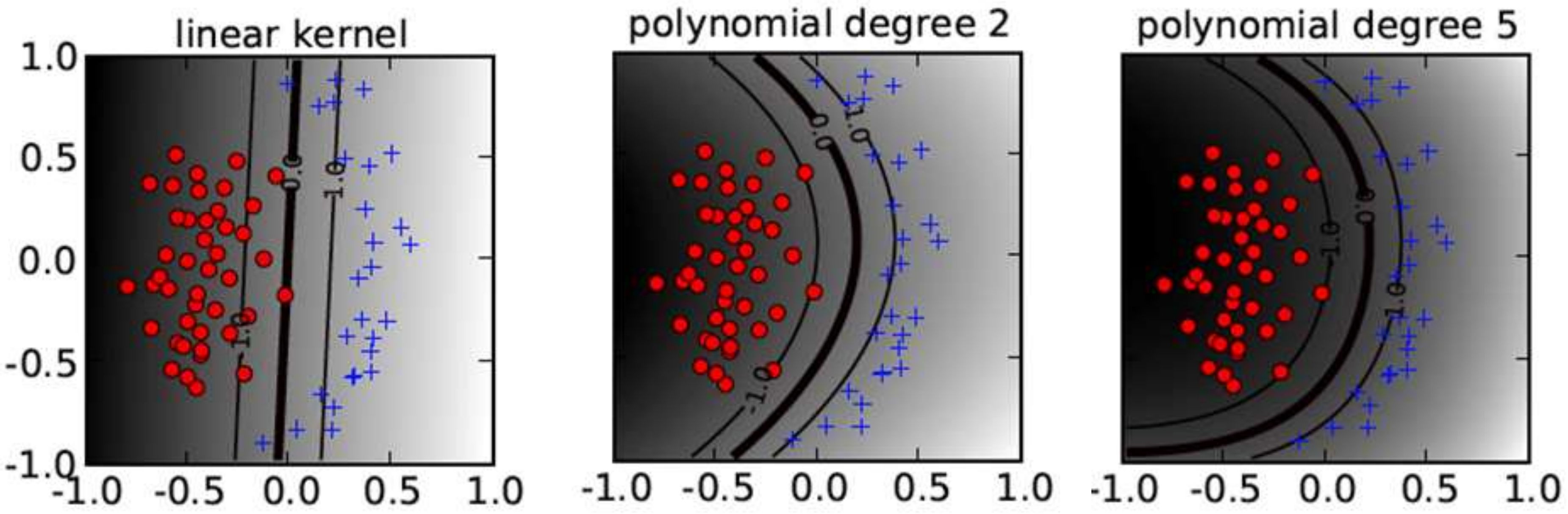


Value of discriminant function
class 1
class 2
class 1
1
2
4
5
6

Choosing Kernel Functions

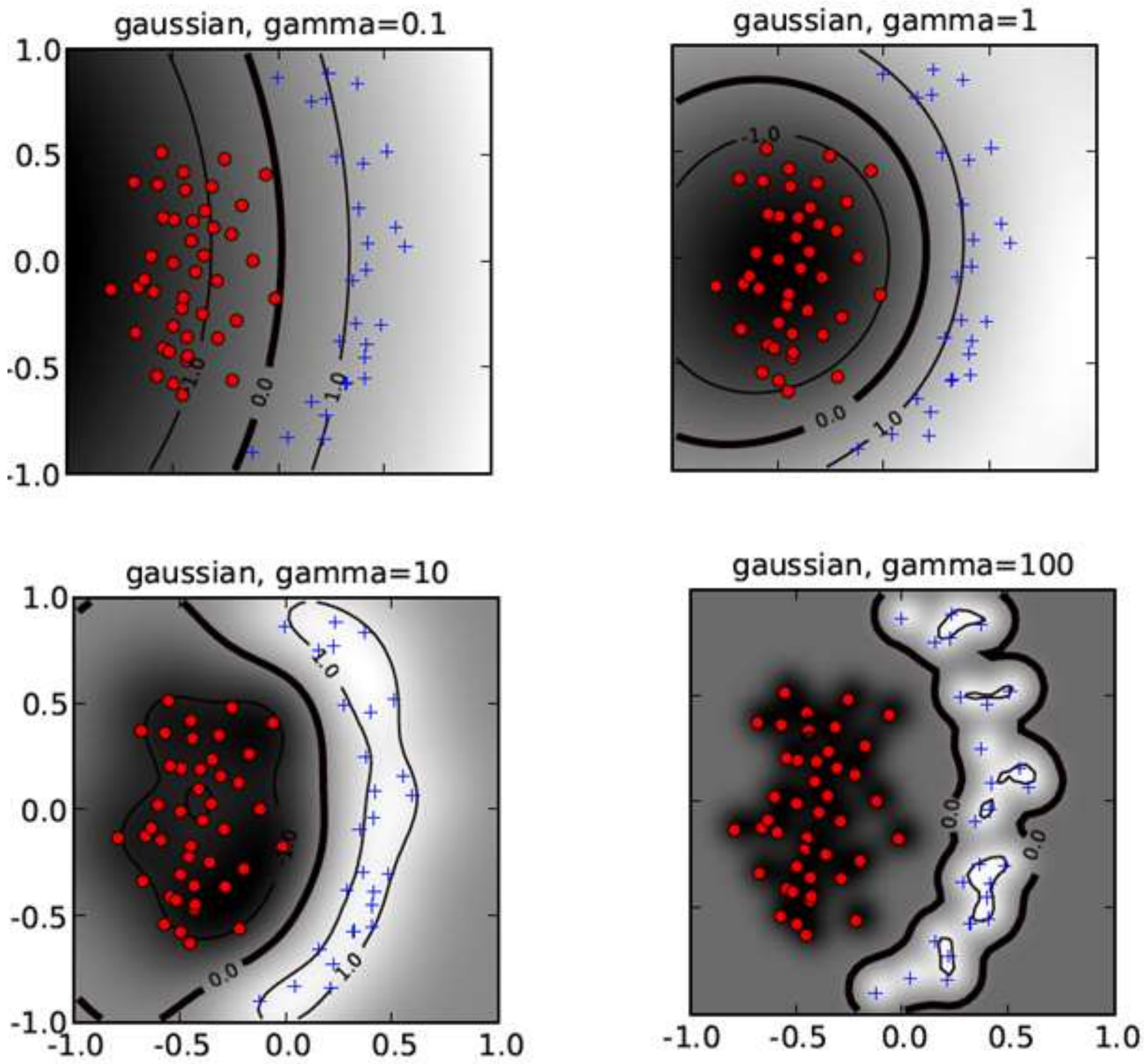
- Probably the most tricky part of using SVM.
- The kernel function is important because it creates the kernel matrix, which summarizes all the data
- Many principles have been proposed (diffusion kernel, Fisher kernel, string kernel, …)
- There is even research to estimate the kernel matrix from available information
- Multiple Kernel Learning
- In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try
- Note that SVM with RBF kernel is closely related to RBF neural networks, with the centers of the radial basis functions automatically chosen for SVM

Polynomial Kernel



RBF Kernel

$$K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/(2\sigma^2))$$



Conclusion and Discussion

Steps in SVM

- Prepare data matrix $\{(x_i,y_i)\}$
- Select a Kernel function
- Select the error parameter *C*
- “Train” the system (to find all α_i and b)
- New data can be classified using α_i and Support Vectors

Weakness

- Training (Testing) is quite slow compared to ANN
  - Because of Constrained Quadratic Programming
- Essentially a binary classifier
  - However, there are some tricks to evade this.
- Very sensitive to noise. Why?
  - A few off data points can completely throw off the algorithm
- Biggest Drawback: The choice of Kernel function.
  - There is no “set-in-stone” theory for choosing a kernel function for any given problem (still in research...)
  - Once a kernel function is chosen, there is only ONE modifiable parameter, the error penalty C.

Strengths

- Training is relatively easy
  - don’t have to deal with local minimum like in ANN
  - SVM solution is always global and unique
- Unlike ANN, doesn’t suffer from “curse of dimensionality”
  - How? Why? We have infinite dimensions?!
  - Maximum Margin Constraint: DOT-PRODUCTS!
- Less prone to overfitting
- Simple, easy to understand geometric interpretation.
  - No large networks to mess around with.

Kernelize Logistic Regression (LR)

$$p(y \mid \vec{x}) = \frac{1}{1 + \exp(-y\vec{x} \cdot \vec{w})}$$

$$l_{reg}(\vec{\alpha}) = \sum_{i=1}^{N} \log \frac{1}{1 + \exp(-y\vec{x} \cdot \vec{w})} - c\sum_{k=1}^{N} w_k^2$$

What is the difference between SVM and LR?

- SVM: L2 norm regularization + hinge loss
- LR: L2 norm regularization + sigmoid probability (entropy loss)

Merits and drawbacks?

How can we introduce the nonlinearity into the logistic regression?

Kernelize Logistic Regression

$$\vec{x} \to \vec{\phi}(\vec{x}),\ \vec{w} = \sum_{i=1}^{N} \alpha_i \vec{\phi}(\vec{x}_i)$$

$$K(\vec{w}, \vec{x}) = \sum_{i=1}^{N} \alpha_i K(\vec{x}_i, \vec{x})$$

$$p(y \mid \vec{x}) = \frac{1}{1 + \exp(-yK(\vec{x}, \vec{w}))} = \frac{1}{1 + \exp\left(-y\sum_{i=1}^{N} \alpha_i K(\vec{x}_i, \vec{x})\right)}$$

$$l_{reg}(\vec{\alpha}) = \sum_{i=1}^{N} \log \frac{1}{1 + \exp\left(-y_i \sum_{j=1}^{N} \alpha_j K(\vec{x}_j, \vec{x}_i)\right)} - c\sum_{i,j=1}^{N} \alpha_i \alpha_j K(\vec{x}_i, \vec{x}_j)$$

- Representation Theorem
- **Kernelization** of many algorithms (PCA, LDA …)
  - From linear to non-linear without changing the algorithm

Thank you!

Q&A