Machine Learning & Pattern Recognition

SONG Xuemeng

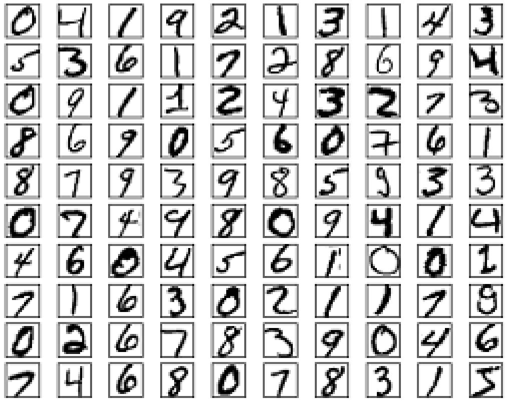
sxmustc@gmail.com

http://xuemeng.bitcron.com/

Support Vector Machines

A Brief History of SVM

- SVM is related to statistical learning theory [3].
- SVM is first introduced in 1992 [1].
- Success in handwritten digit recognition
  - 1.1% test error rate for SVM. The same as that of a carefully constructed neural network, LeNet 4[2].



[1] B.E. Boser et al. A Training Algorithm for Optimal Margin Classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory 5 144-152, Pittsburgh, 1992.
[2] L. Bottou et al. Comparison of classifier methods: a case study in handwritten digit recognition. Proceedings of the 12th IAPR International Conference on Pattern Recognition, vol. 2, pp. 77-82.
[3] V. Vapnik. The Nature of Statistical Learning Theory. 2nd edition, Springer, 1999

SVM: Brief History

1963 Margin (Vapnik & Lerner)

1964 Margin (Vapnik and Chervonenkis, 1964)

1964 RBF Kernels (Aizerman)

1965 Optimization formulation (Mangasarian)

1971 Kernels (Kimeldorf annd Wahba)

1992-1994 SVMs (Vapnik et al)

1996 – present Rapid growth, numerous apps

1996 – present Extensions to other problems

A Brief History of SVM

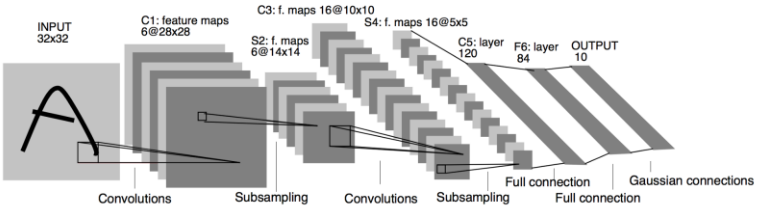
- Vapnik born in the Soviet Union (1936)
- Master: mathematics, the Uzbek State University (1958)
- Ph.D: statistics at the Institute of Control Sciences, Moscow (1964)
- Worked at the Institute of Control Sciences (until 1990)
- Then joined AT&T Bell Labs (1991)
- While at AT&T, Vapnik and colleagues developed the SVM (1995)
- Inducted into U.S. National Academy of Engineering (2006)
- Joined Facebook AI Research (2014)

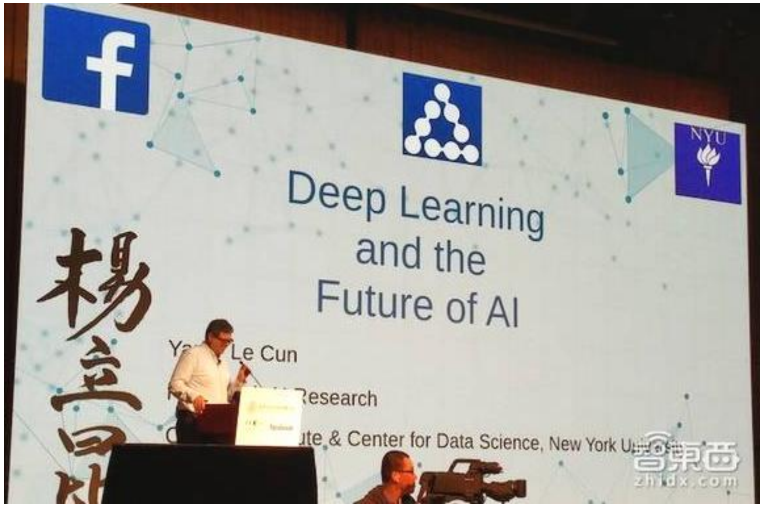
Yann LeCun shared Facebook AI Research's photo.
November 25, 2014 ·

Welcome to Facebook AI Research, Vladimir!

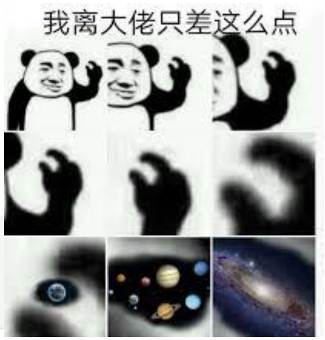
A Brief History of SVM

- Yann LeCun: born in France (1960)
- PhD: Computer Science, Université Pierre et Marie Curie (1987)
- Joined AT&T Bell Labs (1988), where he developed Convolutional Neural Networks
- Joined New York University (2003)
- Join the Facebook AI Research as the first director (2013)
- Inducted into U.S. National Academy of Engineering (2017)

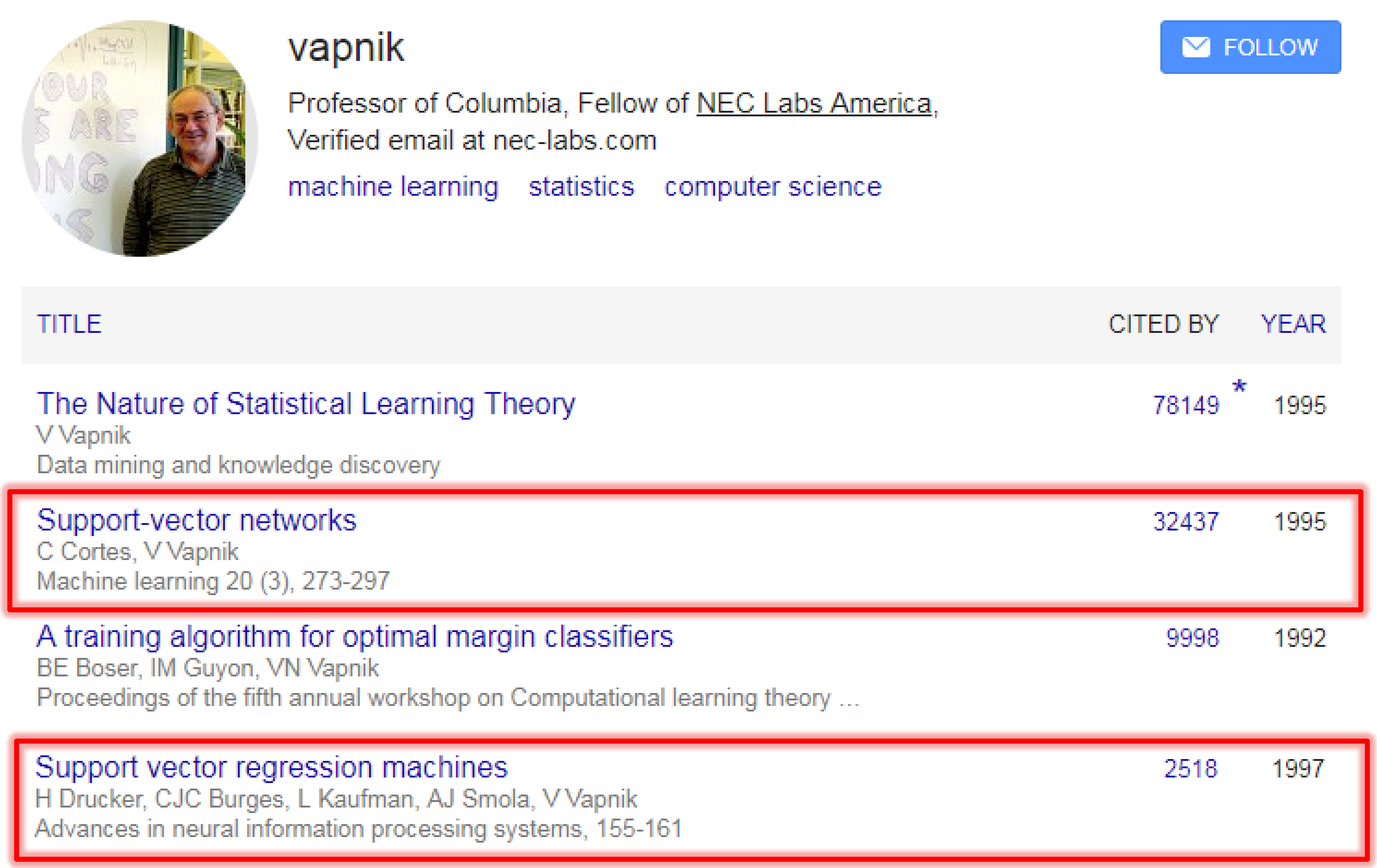



NYU
Deep Learning and the Future of AI
Le Cun
Research
te & Center for Data Science, New York U
智东西
zhidx.com

A Brief History of SVM



Google Scholar



vapnik

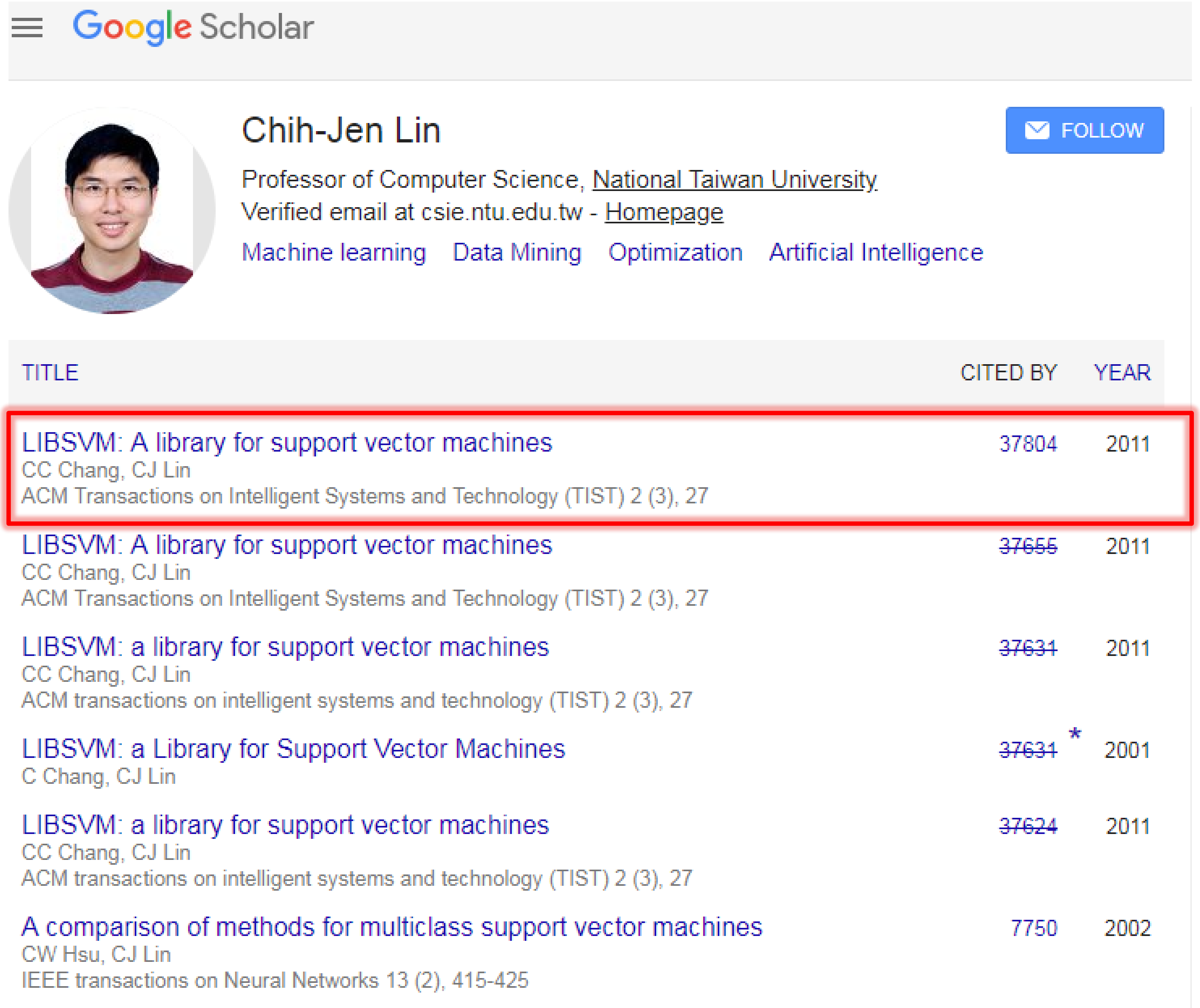
Professor of Columbia, Fellow of NEC Labs America, Verified email at nec-labs.com

machine learning statistics computer science

TITLE	CITED BY	YEAR
The Nature of Statistical Learning Theory V Vapnik Data mining and knowledge discovery	78149 *	1995
Support-vector networks C Cortes, V Vapnik Machine learning 20 (3), 273-297	32437	1995
A training algorithm for optimal margin classifiers BE Boser, IM Guyon, VN Vapnik Proceedings of the fifth annual workshop on Computational learning theory ...	9998	1992
Support vector regression machines H Drucker, CJC Burges, L Kaufman, AJ Smola, V Vapnik Advances in neural information processing systems, 155-161	2518	1997

By February 2017, overall, his publications have been cited close to 180,000 times!

A Brief History of SVM



Chih-Jen Lin

Professor of Computer Science, National Taiwan University
Verified email at csie.ntu.edu.tw - Homepage

Machine learning   Data Mining   Optimization   Artificial Intelligence

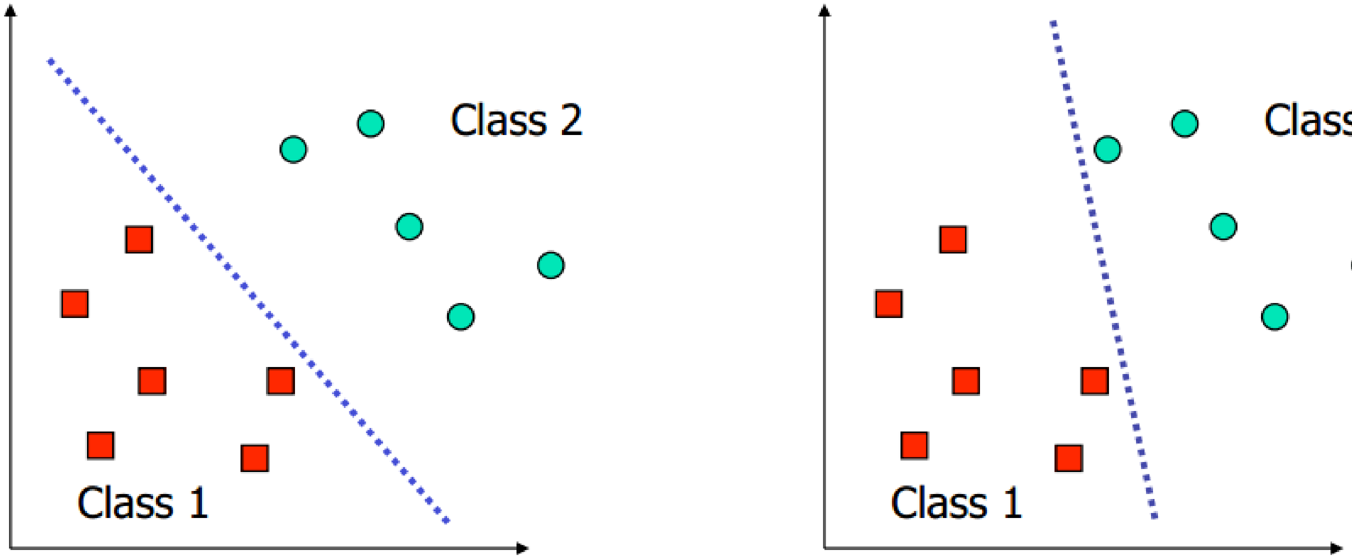
TITLE	CITED BY	YEAR
LIBSVM: A library for support vector machines CC Chang, CJ Lin ACM Transactions on Intelligent Systems and Technology (TIST) 2 (3), 27	37804	2011
LIBSVM: A library for support vector machines CC Chang, CJ Lin ACM Transactions on Intelligent Systems and Technology (TIST) 2 (3), 27	~~37655~~	2011
LIBSVM: a library for support vector machines CC Chang, CJ Lin ACM transactions on intelligent systems and technology (TIST) 2 (3), 27	~~37631~~	2011
LIBSVM: a Library for Support Vector Machines C Chang, CJ Lin	~~37631~~ *	2001
LIBSVM: a library for support vector machines CC Chang, CJ Lin ACM transactions on intelligent systems and technology (TIST) 2 (3), 27	~~37624~~	2011
A comparison of methods for multiclass support vector machines CW Hsu, CJ Lin IEEE transactions on Neural Networks 13 (2), 415-425	7750	2002

Support Vector Machines

What Is a Good Decision Boundary?

What Is a Good Decision Boundary?

- Consider a binary, linearly separable classification problem.
- $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$: our data set and $y_i \in \{1, -1\}$: the class label of $\mathbf{x}_i$.
- Many decision boundaries!
- Are all decision boundaries equally good?



Examples of Bad Decision Boundaries

Preliminary

- **Consider a line l_1:**

$$y = ax + b$$

Preliminary

- **Consider a line l_1:**

$$y = ax + b \quad \underset{y \to x_2}{\overset{x \to x_1}{\Longrightarrow}} ax_1 + (-1)x_2 + b = 0$$

Preliminary

- **Consider a line l_1:**

$$y = ax + b \quad \overset{x \to x_1}{\underset{y \to x_2}{\Longrightarrow}} ax_1 + (-1)x_2 + b = 0 \Rightarrow [a, -1]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b = 0$$

Preliminary

- **Consider a line l_1:**

$$y = ax + b \underset{y \to x_2}{\overset{x \to x_1}{\Longrightarrow}} ax_1 + (-1)x_2 + b = 0 \Rightarrow [a, -1]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b = 0$$

- **Vector representation:**

$$\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b = 0 \qquad \boldsymbol{w} = [w_1\ w_2]^{\boldsymbol{T}} \quad \boldsymbol{x} = [x_1\ x_2]^{\boldsymbol{T}}$$

$$y = ax + b \qquad \boldsymbol{w} = [a, -1]^{\boldsymbol{T}}$$

Preliminary

- **Consider a line l_1:**

$$y = ax + b \quad \overset{x \to x_1}{\underset{y \to x_2}{\Longrightarrow}} ax_1 + (-1)x_2 + b = 0 \Rightarrow [a, -1]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b = 0$$

- **Vector representation:**

$$\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b = 0 \qquad \boldsymbol{w} = [w_1\ w_2]^{\boldsymbol{T}} \quad \boldsymbol{x} = [x_1\ x_2]^{\boldsymbol{T}}$$

$$y = ax + b \qquad \boldsymbol{w} = [a, -1]^{\boldsymbol{T}}$$

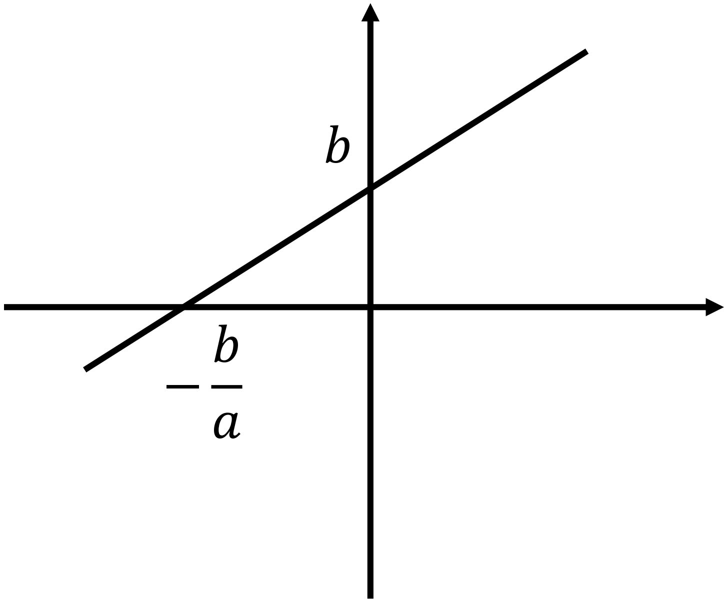
What is the meaning of $\boldsymbol{w}$?

Preliminary

- **Consider a line l_1:**

$$y = ax + b \qquad \boldsymbol{w} = [a, -1]^{\boldsymbol{T}}$$

- Consider $\boldsymbol{\beta} = [1, a]^T$, $\boldsymbol{\beta}$ should be ??? to the line l_1.

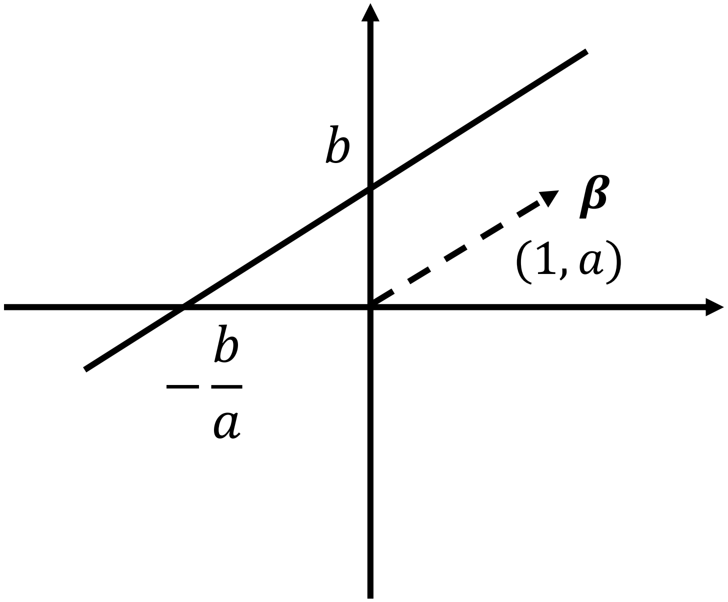


Preliminary

- **Consider a line l_1:**

$$y = ax + b \qquad\qquad \boldsymbol{w} = [a, -1]^{\boldsymbol{T}}$$

- Consider $\boldsymbol{\beta} = [1, a]^T$, $\boldsymbol{\beta}$ should be parallel to the line l_1.

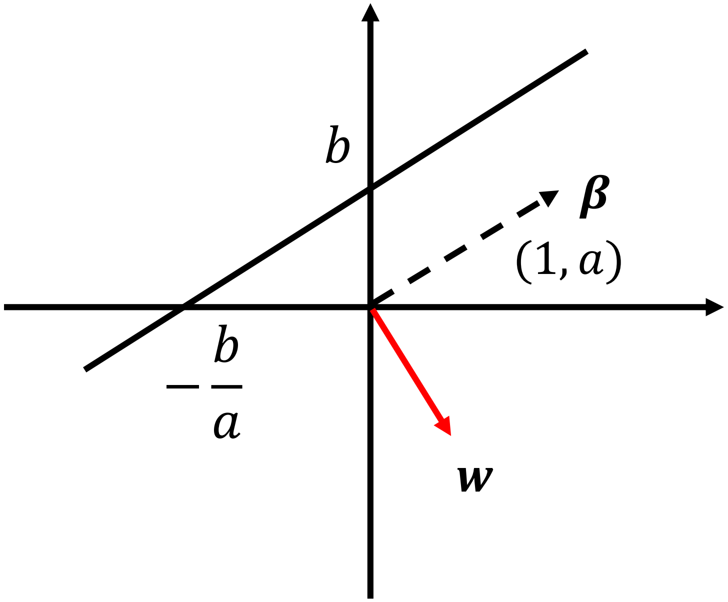


Preliminary

- **Consider a line l_1:**

$$y = ax + b \qquad \boldsymbol{w} = [a, -1]^{\boldsymbol{T}}$$

- Consider $\boldsymbol{\beta} = [1, a]^T$, $\boldsymbol{\beta}$ should be parallel to the line l_1.



- We found that $\boldsymbol{w^T\beta} = 0 \;\rightarrow\; \boldsymbol{\beta} \perp \boldsymbol{w}.$
- Vector $\mathbf{w}$ is perpendicular to the line l_1.

Preliminary

- Given a point (x_0, y_0), the distance from the point to the line $Ax + By + C = 0$:

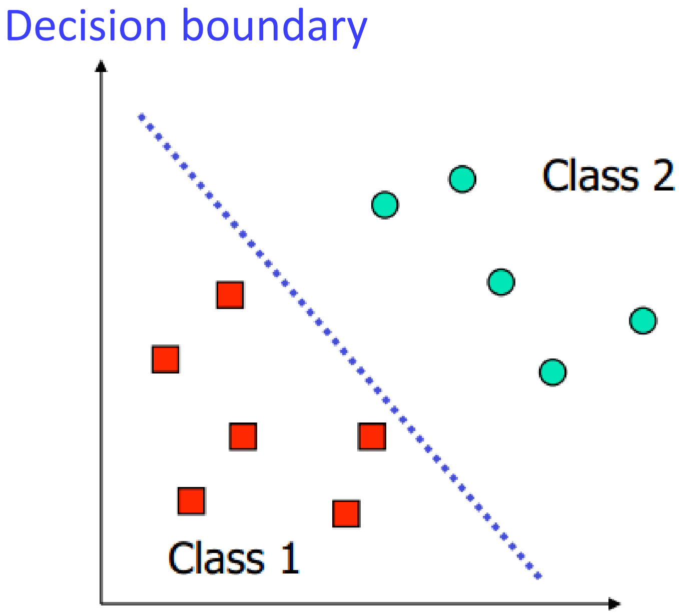
$$distance = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}}$$

- Given a point $\boldsymbol{x_i}$, the distance from the point to the line $\boldsymbol{w^T x} + b = 0$:

$$distance = \frac{\left|\boldsymbol{w^T x} + b\right|}{||\boldsymbol{w}||}$$

What Is a Good Decision Boundary?

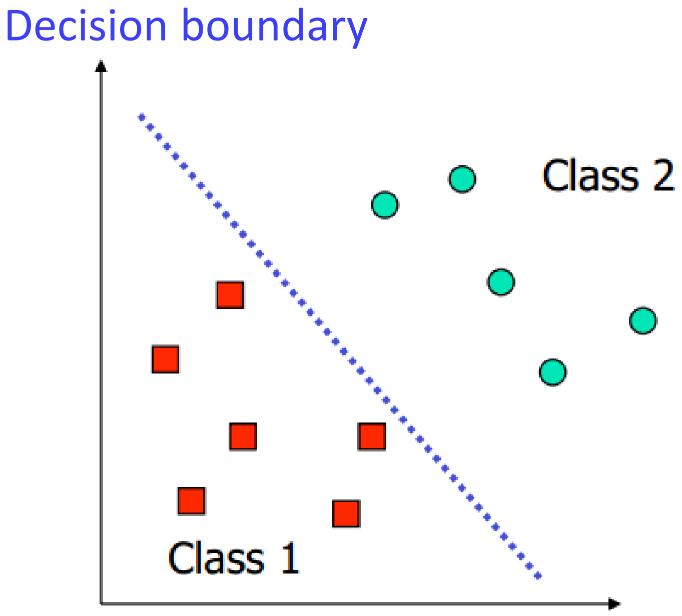
- Find the hyperplane (i.e., decision boundary) linearly separating our classes.
- Our boundary will have equation: $\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b = 0$



- Above the decision boundary should have label 1.
- i.e., for any $\boldsymbol{x_i}$ s.t. $\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b > 0$, then $y_i = 1$.

What Is a Good Decision Boundary?

- Find the hyperplane (i.e., decision boundary) linearly separating our classes.
- Our boundary will have equation: $\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b = 0$

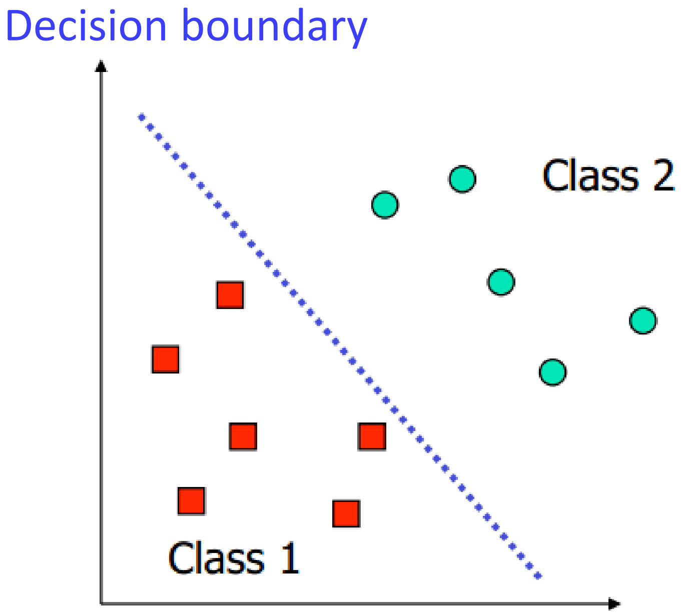


- Above the decision boundary should have label 1.
- i.e., for any $\boldsymbol{x_i}$ s.t. $\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b > 0$, then $y_i = 1$.

- Below the decision boundary should have label -1.
- i.e., for any $\mathbf{x_i}$ s.t. $\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b < 0$, then $y_i = -1$.

What Is a Good Decision Boundary?

- Find the hyperplane (i.e., decision boundary) linearly separating our classes.
- Our boundary will have equation: $\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b = 0$



- Above the decision boundary should have label 1.
- i.e., for any $\boldsymbol{x_i}$ s.t. $\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b > 0$, then $y_i = 1$.

- Below the decision boundary should have label -1.
- i.e., for any $\mathbf{x_i}$ s.t. $\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b < 0$, then $y_i = -1$.

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

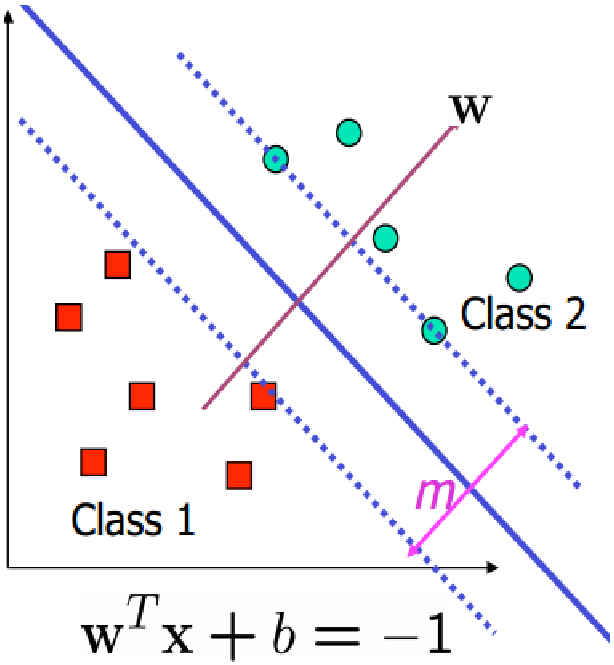
What Is a Good Decision Boundary?

- Moreover, we hope the hyperplane lies in the middle

$$\begin{cases} (\boldsymbol{w}^T\boldsymbol{x}+b)/\|\boldsymbol{w}\| \geq \frac{m}{2} & \forall\ y_i = 1 \\ (\boldsymbol{w}^T\boldsymbol{x}+b)/\|\boldsymbol{w}\| \leq -\frac{m}{2} & \forall\ y_i = -1 \end{cases}$$

$$distance = \frac{|\boldsymbol{w}^T\boldsymbol{x}+b|}{\|\boldsymbol{w}\|}$$

m is the margin



m is the margin

What Is a Good Decision Boundary?

- Moreover, we hope the hyperplane lies in the middle

$$\begin{cases} (\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b)/\|\boldsymbol{w}\| \geq \frac{m}{2} & \forall\, y_i = 1 \\ (\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b)/\|\boldsymbol{w}\| \leq -\frac{m}{2} & \forall\, y_i = -1 \end{cases}$$

$distance = \frac{|\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}+b|}{\|\boldsymbol{w}\|}$

m is the margin

- Can be re-written as

$$\begin{cases} \boldsymbol{w}_p{}^{\boldsymbol{T}}\boldsymbol{x} + b_p \geq 1 & \forall\, y_i = 1 \\ \boldsymbol{w}_p{}^{\boldsymbol{T}}\boldsymbol{x} + b_p \leq -1 & \forall\, y_i = -1 \end{cases}$$

$$\boldsymbol{w}_p = \frac{2\boldsymbol{w}}{\|\boldsymbol{w}\|m} \qquad b_p = \frac{2b}{\|\boldsymbol{w}\|m}$$

What Is a Good Decision Boundary?

- Moreover, we hope the hyperplane lies in the middle

$$\begin{cases} (\boldsymbol{w}^T\boldsymbol{x}+b)/\|\boldsymbol{w}\| \geq \frac{m}{2} & \forall\, y_i = 1 \\ (\boldsymbol{w}^T\boldsymbol{x}+b)/\|\boldsymbol{w}\| \leq -\frac{m}{2} & \forall\, y_i = -1 \end{cases}$$

$distance = \frac{|\boldsymbol{w}^T\boldsymbol{x}+b|}{\|\boldsymbol{w}\|}$

m is the margin

- Can be re-written as

$$\begin{cases} \boldsymbol{w_p}^T\boldsymbol{x}+b_p \geq 1 & \forall\, y_i = 1 \\ \boldsymbol{w_p}^T\boldsymbol{x}+b_p \leq -1 & \forall\, y_i = -1 \end{cases}$$

$$\boldsymbol{w}_p = \frac{2\boldsymbol{w}}{\|\boldsymbol{w}\|m} \qquad b_p = \frac{2b}{\|\boldsymbol{w}\|m}$$

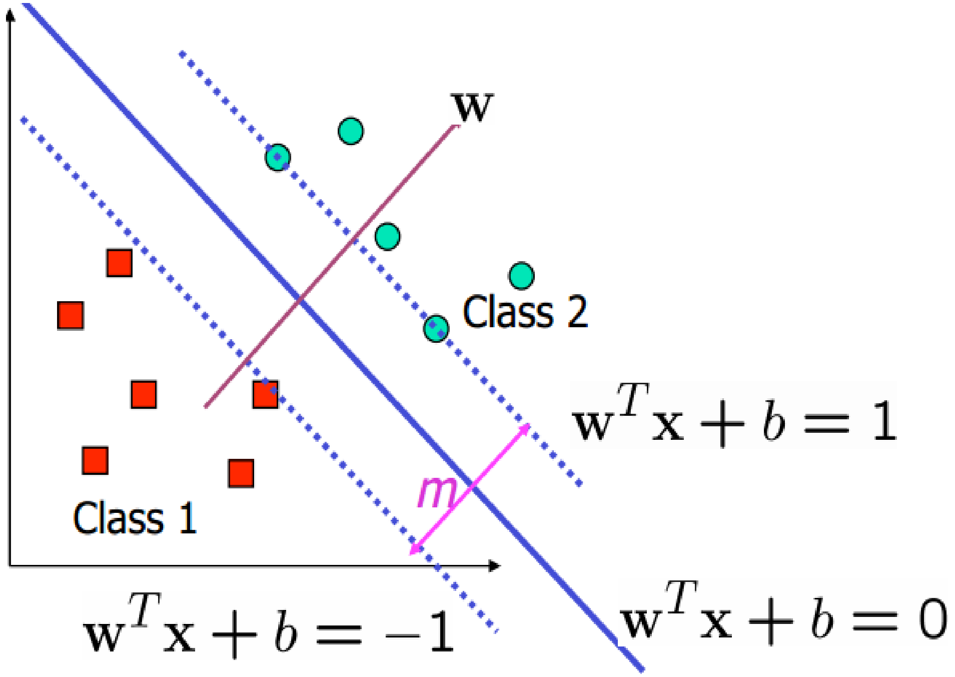
- Interestingly, we found that

$\boldsymbol{w_p}^T\boldsymbol{x}+b_p = 0$ and $\boldsymbol{w}^T\boldsymbol{x}+b = 0$ is the same hyperplane.

What Is a Good Decision Boundary?

- Therefore,

$$\begin{cases} \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b \geq 1 & \forall\, y_i = 1 \\ \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b \leq -1 & \forall\, y_i = -1 \end{cases} \Longrightarrow y_i(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_i + b) \geq 1$$



$\mathbf{w}$
Class 2
Class 1
m
$\mathrm{w}^T\mathrm{x} + b = 1$
$\mathrm{w}^T\mathrm{x} + b = 0$
$\mathrm{w}^T\mathrm{x} + b = -1$

Large-margin Decision Boundary

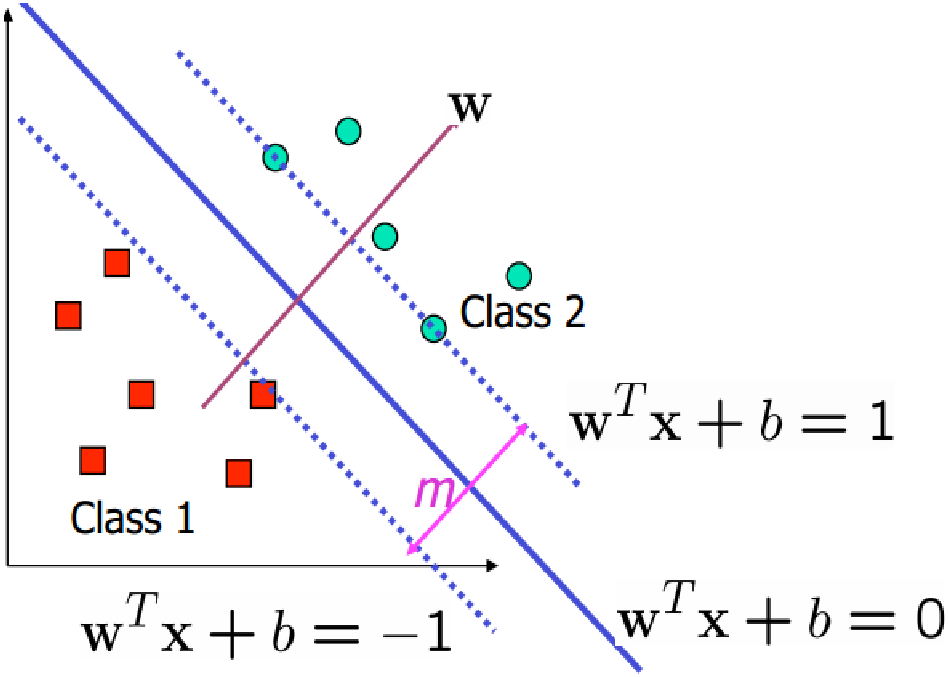
- The decision boundary should be as far away from the data of both classes as possible
  - We should maximize the margin, m

- For the supported vectors,

$$\mathit{Distance} = \left|\boldsymbol{w}^T \boldsymbol{x}_i + b\right| / \|\boldsymbol{w}\|$$

$$= 1/\|\boldsymbol{w}\|$$

$$m = 2/\|\boldsymbol{w}\|$$


w
Class 2
$\mathbf{w}^T\mathbf{x} + b = 1$
m
Class 1
$\mathbf{w}^T\mathbf{x} + b = -1$
$\mathbf{w}^T\mathbf{x} + b = 0$

Optimization Problem

- The decision boundary can be found by solving the following constraint optimization problem

$$\max_{\boldsymbol{w}} 2/||\boldsymbol{w}||$$

$$s.t.\ y_i(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_i + b) \geq 1, i = 1,2, \ldots, n$$

Optimization Problem

- The decision boundary can be found by solving the following constraint optimization problem

$$\max_{\boldsymbol{w}} 2/\|\boldsymbol{w}\|$$

$$s.t.\ y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1, i = 1,2,\dots,n$$

- To solve the problem efficiently, we transformed it into a form:

$$\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$s.t.\ y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1, i = 1,2,\dots,n$$

- The above is an optimization problem with a convex quadratic objective and only linear constraints.

Large-margin Decision Boundary

- However, here we will turn to the Lagrange duality.

- The dual form will allow us to use kernels to get optimal margin classifiers to work efficiently in very high dimensional spaces.

- The dual form will allow us to derive an efficient algorithm to sole the optimization problem.

Lagrange Duality

Consider a problem of the following form:

$$\min_{w} f(\boldsymbol{w})$$

$$\text{s.t. } h_i(\boldsymbol{w}) = 0, i = 1, \dots, l.$$

Lagrange multiplier method:

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\beta}) = f(\boldsymbol{w}) + \sum_{i=1}^{l} \beta_i h_i(\boldsymbol{w})$$

β_i's are the Lagrange multipliers.

No constraint now.

Set the partial derivatives to zero:

$$\frac{\partial \mathcal{L}(\boldsymbol{w}, \boldsymbol{\beta})}{\partial w_i} = 0 \qquad \frac{\partial \mathcal{L}(\boldsymbol{w}, \boldsymbol{\beta})}{\partial \beta_i} = 0$$

Lagrange Duality

Consider the following primal optimization problem:

$$\min_{w} f(\boldsymbol{w})$$

$$\text{s.t. } g_i(\boldsymbol{w}) \leq 0, i = 1, \ldots, k$$

$$h_i(\boldsymbol{w}) = 0, i = 1, \ldots, l.$$

Generalized Lagrangian

α_i's and β_i's are the Lagrange multipliers.

$\alpha_i \geq 0$

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\boldsymbol{w}) + \sum_{i=1}^{k} \alpha_i g_i(\boldsymbol{w}) + \sum_{i=1}^{l} \beta_i h_i(\boldsymbol{w})$$

Lagrange Duality

Consider the following primal optimization problem:

$$\min_{w} f(\boldsymbol{w})$$
$$\text{s.t. } g_i(\boldsymbol{w}) \leq 0, i = 1, \dots, k$$
$$h_i(\boldsymbol{w}) = 0, i = 1, \dots, l.$$

Generalized Lagrangian

α_i's and β_i's are the Lagrange multipliers.

$\alpha_i \geq 0$

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\boldsymbol{w}) + \sum_{i=1}^{k} \alpha_i g_i(\boldsymbol{w}) + \sum_{i=1}^{l} \beta_i h_i(\boldsymbol{w})$$

Consider the quantity:

$$\theta_{\mathcal{P}}(\boldsymbol{w}) = \max_{\boldsymbol{\alpha}, \boldsymbol{\beta}, \alpha_i \geq 0} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

If $\boldsymbol{w}$ is given and violates any primal constraint (i.e., $g_i(\boldsymbol{w}) > 0$ or $h_i(\boldsymbol{w}) \neq 0$ for some i), then what happens? $\theta_{\mathcal{P}}(\boldsymbol{w}) =$?

Lagrange Duality

If $\boldsymbol{w}$ is given and violates ay primal constraint (i.e., $g_i(\boldsymbol{w}) > 0$ or $h_i(\boldsymbol{w}) \neq 0$ for some i),

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\boldsymbol{w}) + \sum_{i=1}^{k} \alpha_i g_i(\boldsymbol{w}) + \sum_{i=1}^{l} \beta_i h_i(\boldsymbol{w})$$

$$\theta_{\mathcal{P}}(\boldsymbol{w}) = \max_{\boldsymbol{\alpha}, \boldsymbol{\beta}, \alpha_i \geq 0} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \infty$$

Lagrange Duality

If $\boldsymbol{w}$ is given and violates ay primal constraint (i.e., $g_i(\boldsymbol{w}) > 0$ or $h_i(\boldsymbol{w}) \neq 0$ for some i),

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\boldsymbol{w}) + \sum_{i=1}^{k} \alpha_i g_i(\boldsymbol{w}) + \sum_{i=1}^{l} \beta_i h_i(\boldsymbol{w})$$

$$\theta_{\mathcal{P}}(\boldsymbol{w}) = \max_{\boldsymbol{\alpha}, \boldsymbol{\beta}, \alpha_i \geq 0} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \infty$$

Therefore, if the constraints are indeed satisfied for a given $\boldsymbol{w}$, then

$$\theta_{\mathcal{P}}(\boldsymbol{w}) = f(\boldsymbol{w})$$

Lagrange Duality

If $\boldsymbol{w}$ is given and violates ay primal constraint (i.e., $g_i(\boldsymbol{w}) > 0$ or $h_i(\boldsymbol{w}) \neq 0$ for some i),

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\boldsymbol{w}) + \sum_{i=1}^{k} \alpha_i g_i(\boldsymbol{w}) + \sum_{i=1}^{l} \beta_i h_i(\boldsymbol{w})$$

$$\theta_{\mathcal{P}}(\boldsymbol{w}) = \max_{\boldsymbol{\alpha}, \boldsymbol{\beta}, \alpha_i \geq 0} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \infty$$

Therefore, if the constraints are indeed satisfied fo n

$$\theta_{\mathcal{P}}(\boldsymbol{w}) = f(\boldsymbol{w})$$

WHY?

Lagrange Duality

If $\boldsymbol{w}$ is given and violates ay primal constraint (i.e., $g_i(\boldsymbol{w}) > 0$ or $h_i(\boldsymbol{w}) \neq 0$ for some i),

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\boldsymbol{w}) + \sum_{i=1}^{k} \alpha_i g_i(\boldsymbol{w}) + \sum_{i=1}^{l} \beta_i h_i(\boldsymbol{w})$$

$$\theta_{\mathcal{P}}(\boldsymbol{w}) = \max_{\boldsymbol{\alpha}, \boldsymbol{\beta}, \alpha_i \geq 0} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \infty$$

Therefore, if the constraints are indeed satisfied fo n

$$\theta_{\mathcal{P}}(\boldsymbol{w}) = f(\boldsymbol{w})$$

WHY?

Consequently…

$$\theta_{\mathcal{P}}(\boldsymbol{w}) = \begin{cases} f(\boldsymbol{w}) & \text{if } \boldsymbol{w} \text{ satisfies primal constraints} \\ \infty & \text{otherwise.} \end{cases}$$

Lagrange Duality

Consequently…

$$\min_{\boldsymbol{w}} f(\boldsymbol{w})$$

$$\text{s.t. } g_i(\boldsymbol{w}) \le 0, i = 1, \dots, k$$

$$h_i(\boldsymbol{w}) = 0, i = 1, \dots, l.$$

$$\Rightarrow \quad \min_{\boldsymbol{w}} \theta_{\mathcal{P}}(\boldsymbol{w}) = \min_{\boldsymbol{w}} \max_{\boldsymbol{\alpha},\boldsymbol{\beta},\alpha_i \ge 0} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

How to optimize it? **DIFFICULT!**

- It is hard to explicitly express the objective function $\theta_{\mathcal{P}}(\boldsymbol{w})$.
- Thus it is hard to calculate the derivative with respect with $\boldsymbol{w}$.

Lagrange Duality

Primal optimization problem

$$\min_{\boldsymbol{w}} \theta_{\mathcal{P}}(\boldsymbol{w}) = \min_{\boldsymbol{w}} \max_{\boldsymbol{\alpha},\boldsymbol{\beta},\alpha_i \geq 0} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

Let us look at a slightly different problem. We define:

$$\theta_{\mathcal{D}}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \min_{w} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

$\mathcal{D}$ refers to “dual”.

We can now pose the dual optimization problem:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\beta},\alpha_i \geq 0} \theta_{\mathcal{D}}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \max_{\boldsymbol{\alpha},\boldsymbol{\beta},\alpha_i \geq 0} \min_{w} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

How are the primal and the dual problems related?

Lagrange Duality

How are the primal and the dual problems related?

$$\boldsymbol{d}^* = \max_{\boldsymbol{\alpha},\boldsymbol{\beta},\alpha_i \geq 0} \min_{w} \mathcal{L}(\boldsymbol{w},\boldsymbol{\alpha},\boldsymbol{\beta}) \leq \min_{w} \max_{\boldsymbol{\alpha},\boldsymbol{\beta},\alpha_i \geq 0} \mathcal{L}(\boldsymbol{w},\boldsymbol{\alpha},\boldsymbol{\beta}) = \boldsymbol{p}^*$$

“max min” is always less than or equal to the “min max”

Under certain conditions, we will have $\boldsymbol{d}^* = \boldsymbol{p}^*$.

Theorem 1

Condition: Suppose f and the g_i's are convex, and the h_i's are affine. Suppose further that there exists some $\boldsymbol{w}$ so that $g_i(\boldsymbol{w}) < 0$ for all i (strictly feasible).

- There must exist $\boldsymbol{w}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*$ so that $\boldsymbol{w}^*$ is the solution to the primal problem, $\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*$ are the solution to the dual problem, i.e., $\boldsymbol{d}^* = \boldsymbol{p}^* = \mathcal{L}(\boldsymbol{w}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$.
- $\boldsymbol{w}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*$ satisfy the Karush-Kuhn-Tucker (KKT) conditions.

$$\frac{\partial \mathcal{L}(\boldsymbol{w}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)}{\partial w_i} = 0 \qquad i = 1, \dots, n$$

$$\frac{\partial \mathcal{L}(\boldsymbol{w}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)}{\partial \beta_i} = 0 \qquad i = 1, \dots, l$$

$$\boxed{\alpha_i{}^* g_i(\boldsymbol{w}^*) = 0 \qquad i = 1, \dots, k}$$

$$g_i(\boldsymbol{w}^*) \le 0 \qquad i = 1, \dots, k$$

$$\alpha_i{}^* \ge 0 \qquad i = 1, \dots, k$$

We always have either $\alpha_i{}^* = 0$ or $g_i(\boldsymbol{w}^*) = 0$.

If some $\boldsymbol{w}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*$ satisfy the KKT conditions, then it is also a solution to the primal and dual problems.

Large-margin Decision Boundary

- Optimization problem

$$\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 = \frac{1}{2}\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w}$$

$$s.t.\ y_i(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_i + b) \geq 1, i = 1,2, \dots, n$$

- The Lagrangian

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = f(\boldsymbol{w}) + \sum_{i=1}^{n} \alpha_i(1 - y_i(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_i + b))$$

- Taking the partial derivative

$$\frac{\partial \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha})}{\partial \boldsymbol{w}} = \boldsymbol{w} + \sum_{i=1}^{n} -\alpha_i y_i \boldsymbol{x}_i = 0 \quad \Rightarrow \quad \boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i$$

$$\frac{\partial \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha})}{\partial b} = \sum_{i=1}^{n} -\alpha_i y_i = 0 \quad \Rightarrow \quad 0 = \sum_{i=1}^{n} \alpha_i y_i$$

Large-margin Decision Boundary

- Optimization problem

$$\boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i \; 0 = \sum_{i=1}^{n} \alpha_i y_i$$

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = f(\boldsymbol{w}) + \sum_{i=1}^{n} \alpha_i (1 - y_i(\boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{x}_i + b))$$

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

$$\mathcal{L}(\boldsymbol{w}^*, \boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i{}^{\boldsymbol{T}} \boldsymbol{x}_j$$

Large-margin Decision Boundary

- Optimization problem

$$\mathcal{L}(\boldsymbol{w}^*, \boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j {\boldsymbol{x}_i}^{\boldsymbol{T}} \boldsymbol{x}_j$$

- Dual optimization problem: $\underset{\boldsymbol{\alpha},\boldsymbol{\beta},\alpha_i \geq 0}{max}\ \theta_{\mathcal{D}}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \underset{\boldsymbol{\alpha},\boldsymbol{\beta},\alpha_i \geq 0}{max}\ \underset{w}{min}\ \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$

$$\underset{\boldsymbol{\alpha}}{max} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j {\boldsymbol{x}_i}^{\boldsymbol{T}} \boldsymbol{x}_j$$

$$s.t.\ \alpha_i \geq 0\boldsymbol{,}\ i = 1, \ldots, n$$

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

How to optimize?

Coordinate Ascent

- Consider trying to solve the unconstrained optimization problem

$$\underset{\boldsymbol{\alpha}}{max}\, W(\alpha_1, \alpha_2, \dots, \alpha_l)$$

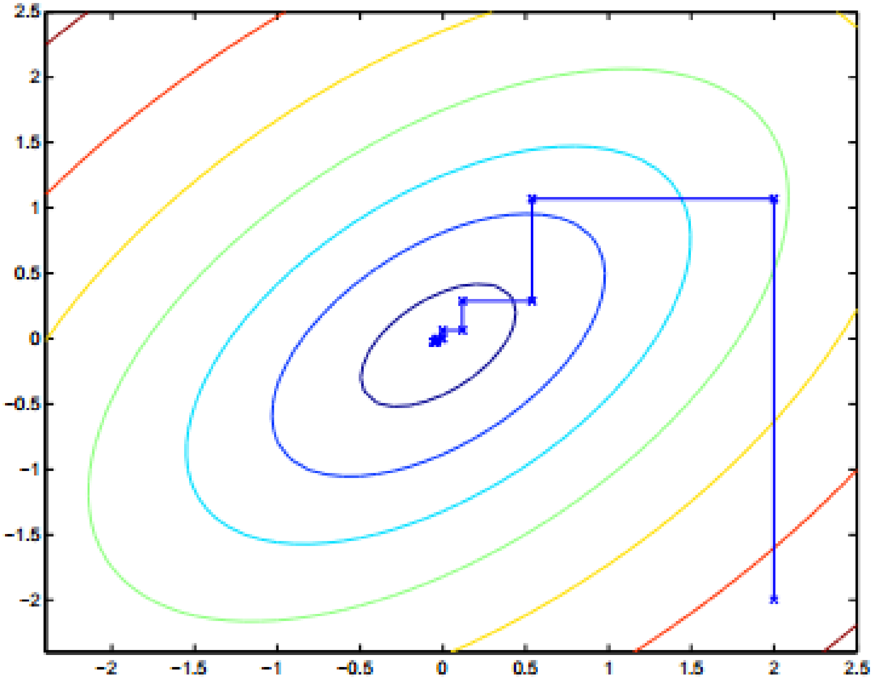
- Coordinate Ascent

```
Loop until convergence:{
        For $i = 1, \dots l${
            $\alpha_i := \underset{\hat{\alpha}_i}{argmax}\, W(\alpha_1, \dots, \alpha_{i-1}, \hat{\alpha}_i, \alpha_{i+1}, \dots \alpha_l)$
        }
}
```

In the innermost loop of this algorithm, we will hold all the variables except for some α_i fixed, and re-optimize W with respect to just the parameter α_i.

Coordinate Ascent

- The ellipses are the contours of the objective function.
- Coordinate ascent was initialized at (2, -2).
- The path that it took on its way to the global maximum is plotted.
- **Note**: Coordinate ascent takes a step that's parallel to one of the axes, since only one variable is being optimized at a time.



Sequential Minimal Optimization

- Dual optimization problem:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j {\boldsymbol{x}_i}^{\boldsymbol{T}} \boldsymbol{x}_j$$

$$s.t.\ \alpha_i \geq 0,\ i = 1, \dots, n$$

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

- Let's say we have a set of α_i's that satisfy the constraints.
- Suppose we hold $\alpha_2, \dots, \alpha_n$ fixed, can we take a coordinate ascent step and optimize the function with respect to α_1?
- **NO!!!** $\sum_{i=1}^{n} \alpha_i y_i = 0$ $\alpha_1 = -y_1 \sum_{i=2}^{n} \alpha_i y_i$

Sequential Minimal Optimization

- We must update at least two of α_i's simultaneously.
- SMO

```
Repeat until convergence:{
    1. Select some pair $\alpha_i$ and $\alpha_j$ to update next.
    2. Re-optimize $W(\boldsymbol{\alpha})$ with respect to $\alpha_i$ and $\alpha_j$, while holding
       all the other $\alpha_k$'s $(k \neq i, j)$ fixed.
}
```

- SMO is efficient as that the update to α_i and α_j can be computed very efficiently.

Deriving The Efficient Update

- Suppose we have a set of α_i's that satisfy the constraints.
- And we decided to hold $\alpha_3, \ldots, \alpha_n$ fixed, and optimize the objective function with respect to α_1 and α_2.
- Based on the constraint, we have

$$\alpha_1 y_1 + \alpha_2 y_2 = -\sum_{i=3}^{n} \alpha_i y_i = \zeta \quad \text{Constant}$$

$$W(\alpha_1, \alpha_2, \ldots, \alpha_n) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i{}^{\boldsymbol{T}} \boldsymbol{x}_j$$

$$W(\alpha_1, \alpha_2, \ldots, \alpha_n) = W(y_1(\zeta - \alpha_2 y_2), \alpha_2, \ldots, \alpha_n)$$

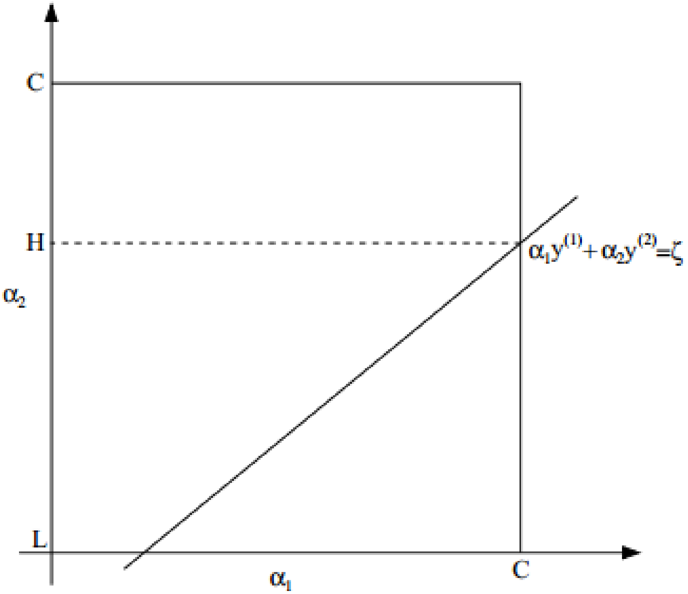
- This is some quadratic function with α_2.

Deriving The Efficient Update

- If we ignore the box constraint $(L \leq \alpha_2 \leq H)$, then we can easily maximize the quadratic function. Let $\alpha_2^{\text{new,unclipped}}$ denote the resulting value of α_2.
- Then we have $\alpha_2^{new} = \begin{cases} H & \text{if } \alpha_2^{new,unclipped} > H \\ \alpha_2^{new,unclipped} & \text{if } L \leq \alpha_2^{new,unclipped} \leq H \\ L & \text{if } \alpha_2^{new,unclipped} < L \end{cases}$

- Once we have α_2^{new}, we can obtain the α_1^{new} with $\alpha_1 y_1 + \alpha_2 y_2 = \zeta$

- Now we have obtained the solution of $\boldsymbol{\alpha}$, how to get $\boldsymbol{w}^*$ **and** b^*?



How To Get $\boldsymbol{w}^*$?

- Remember that we have the following constraint by taking the partial derivative

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

$$\frac{\partial\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha})}{\partial\boldsymbol{w}} = \boldsymbol{w} + \sum_{i=1}^{n} -\alpha_i y_i \boldsymbol{x}_i = 0$$

$$\frac{\partial\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha})}{\partial b} = \sum_{i=1}^{n} -\alpha_i y_i = 0$$

$$\boxed{\begin{aligned} \boldsymbol{w} &= \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i \\ 0 &= \sum_{i=1}^{n} \alpha_i y_i \end{aligned}}$$

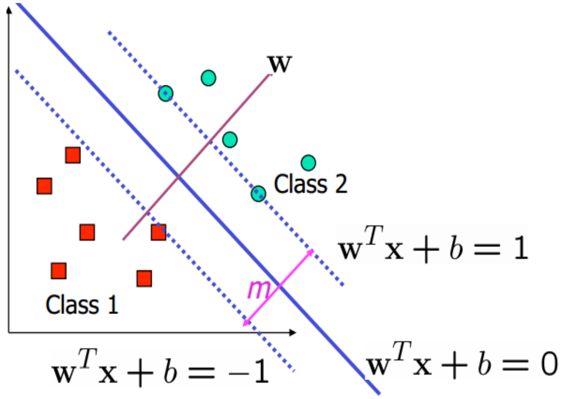
$$\boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i = \sum_{i \in S} \alpha_i y_i \boldsymbol{x}_i$$

How To Get b^*?

- In practice, we can derive b^* as follows,

1. Note that given $\boldsymbol{w}^*$

$$b^* = -\frac{\max_{i:y_i=-1} \boldsymbol{w}^{*\boldsymbol{T}}\boldsymbol{x}_i + \min_{i:y_i=1} \boldsymbol{w}^{*\boldsymbol{T}}\boldsymbol{x}_i}{2}$$



w
Class 2
$\mathrm{w}^T\mathrm{x} + b = 1$
m
Class 1
$\mathrm{w}^T\mathrm{x} + b = -1$
$\mathrm{w}^T\mathrm{x} + b = 0$

2. Note that given a supported vector, we have $y_s f(\boldsymbol{x}_s) = 1$

$$y_s\left(\left(\sum_{i\in S}\alpha_i y_i \boldsymbol{x}_i{}^{\boldsymbol{T}}\right)\boldsymbol{x}_s + b\right) = 1$$

where $S = \{i|\alpha_i > 0, i = 1,2,\ldots,n\}$ is the set of index of supported vectors.

$$b^* = \frac{1}{|S|}\sum_{i\in S}\left(\frac{1}{y_s} - \sum_{i\in S}\alpha_i y_i \boldsymbol{x}_i{}^{\boldsymbol{T}}\boldsymbol{x}_i\right)$$

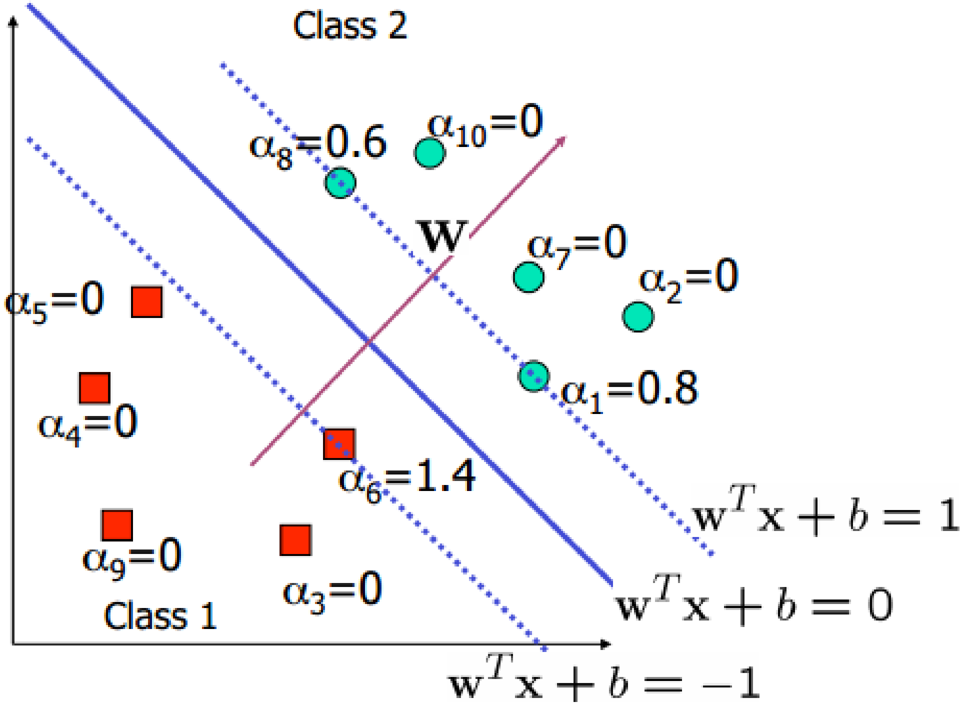
Characteristics of The Solution

- Many of the α_i's are zero (why?)
  - $\boldsymbol{w}$ is a linear combination of a small number of data points.

$$\boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i = \sum_{i \in S} \alpha_i y_i \boldsymbol{x}_i$$

- Supported vectors (SV):
  - $\boldsymbol{x}_i$ with a non-zero α_i

- The decision boundary is determined only by the SV.

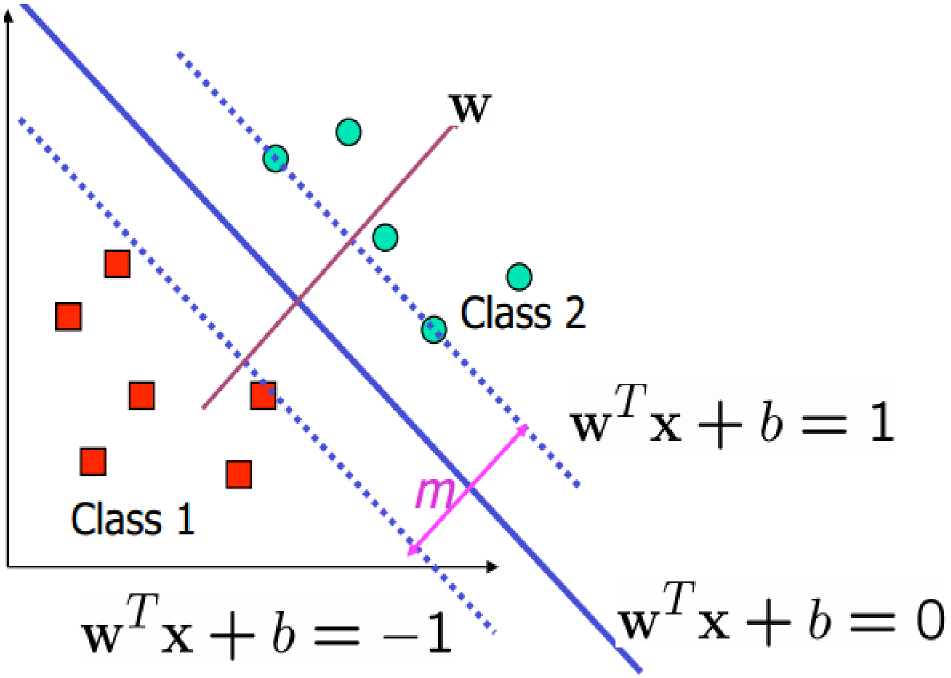


Test Phase

Once we have trained a Support Vector Machine, how can we use it?

Test Phase

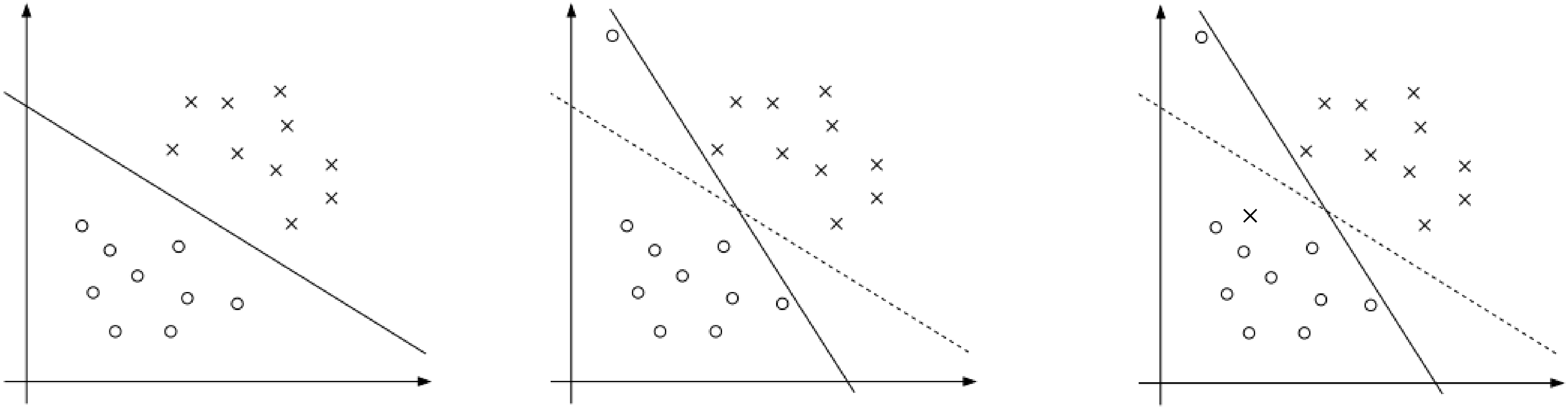
- We simply determine on which side of the decision boundary a given test sample $\boldsymbol{x}$ lies and assign the corresponding class label. i.e. we take the class of $\boldsymbol{x}$ to be $sgn(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b)$
- Note: $\boldsymbol{w}$ need not to be formed explicitly



w
Class 2
$\mathrm{w}^T\mathrm{x} + b = 1$
m
Class 1
$\mathrm{w}^T\mathrm{x} + b = -1$
$\mathrm{w}^T\mathrm{x} + b = 0$

Regularization and The Non-separable Case

- In some cases (due to the outliers), it is not clear that finding a separating hyperplane is exactly what we'd want to do.
- Figure (a) shows an optimal margin classifier, and when a single outlier is added in the upper-left region (Figure b), it causes the decision boundary to make a dramatic swing, and the resulting classifier has a much smaller margin (sensitive to outliers).



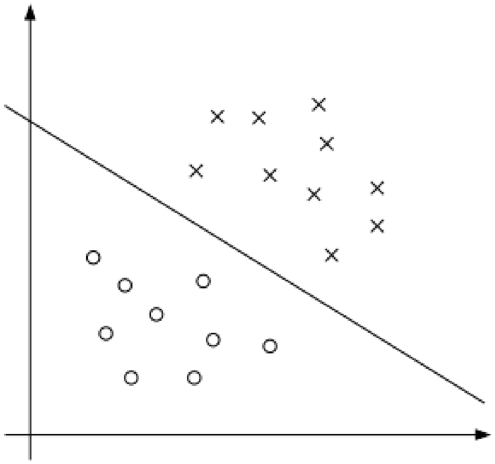
(a) Linearly separable

(b) Linearly separable with outliers

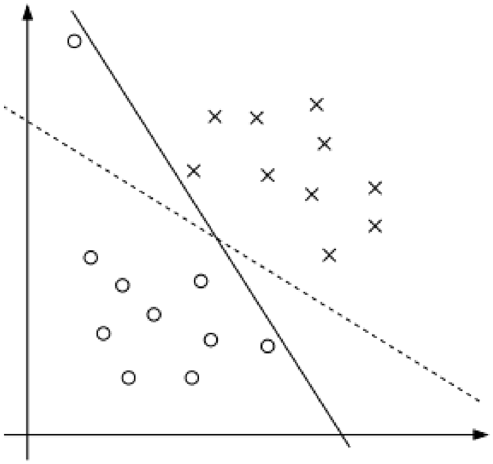
(c) Non-linearly separable

Regularization and The Non-separable Case

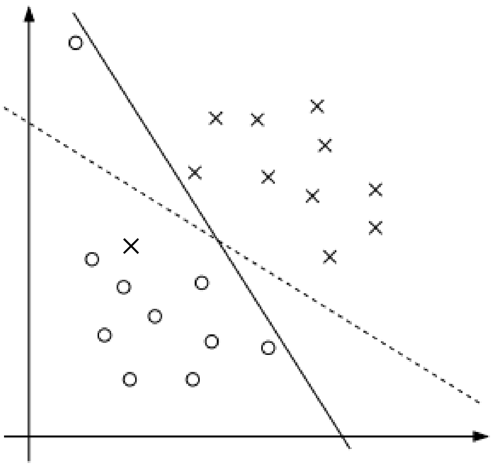
- In some cases (due to the outliers), it is not clear that finding a separating hyperplane is exactly what we'd want to do.
- In some cases (Figure c), the data cannot be perfectly linearly separable.



(a) Linearly separable



(b) Linearly separable with outliers



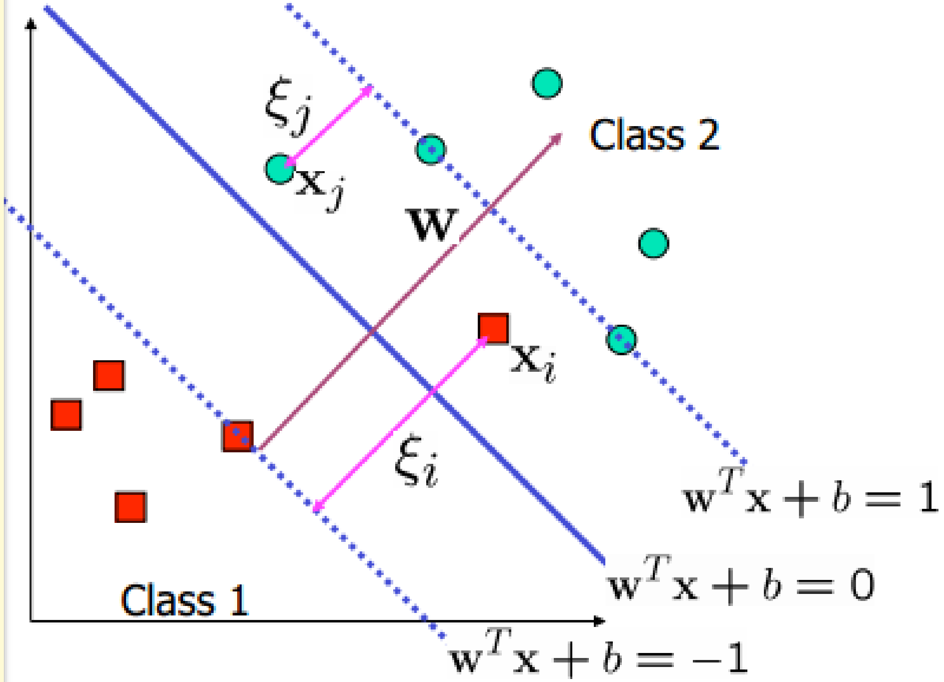
(c) Non-linearly separable

Regularization and The Non-separable Case

To make the algorithm work for non-linearly separable datasets as well as be less sensitive to outliers, we introduce the positive slack variables ξ_i in constraints (allow "error" ξ_i in classification):

$$\begin{cases} \boldsymbol{w}^T\boldsymbol{x}_i + b \geq \;\; 1 - \xi_i & y_i = 1 \\ \boldsymbol{w}^T\boldsymbol{x}_i + b \leq -1 + \xi_i\,, & y_i = -1 \\ \qquad \xi_i \geq 0 & \forall\, i \end{cases}$$

- $\xi_i = 0$: no error for $\boldsymbol{x}_i$.
- For an error to occur, the corresponding $\xi_i$must exceed 1, so $\sum_i \xi_i$ is an upper bound on the number of training errors.


ξ_j
$\mathbf{x}_j$
Class 2
$\mathbf{w}$
$\mathbf{x}_i$
ξ_i
$\mathrm{w}^T\mathrm{x} + b = 1$
$\mathrm{w}^T\mathrm{x} + b = 0$
Class 1
$\mathrm{w}^T\mathrm{x} + b = -1$

Regularization and The Non-separable Case

A natural way to assign an extra cost for errors as follow,

$$\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\left(\sum_i \xi_i\right)^k$$

- C is a parameter to be chosen by the user, a larger C refers to assigning a higher penalty to errors.
- For simplicity, we set k=1.

- We reformulate our optimization (l_1 regularization) as follows,

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

$$s.t.\ y_i(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_i + b) \geq 1 - \xi_i, i = 1,2, \ldots, n$$

$$\xi_i \geq 0, i = 1,2, \ldots, n$$

Regularization and The Non-separable Case

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

$$s.t.\ y_i(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_i + b) \geq 1 - \xi_i, i = 1,2,\dots,n$$

$$\xi_i \geq 0, i = 1,2,\dots,n$$

- Examples are permitted to have margin less than 1
  - If an example has margin 1 – ξ_i (with ξ_i> 0), we pay a cost of the objective function being increased by $C\xi_i$.

- C controls the relative weighting between the twin goals
  - Making the $\|\boldsymbol{w}\|^2$ small (makes the margin large)
  - Ensuring that most examples have margin at least 1.

Regularization and The Non-separable Case

- As before, we can form the Lagrangian,

$$\mathcal{L}(\boldsymbol{w}, b, \xi, \boldsymbol{\alpha}, \boldsymbol{r}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[y_i\left(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_i + b\right) - 1 + \xi_i\right] - \sum_{i=1}^{n} r_i\xi_i$$

α_i’s and r_i’s are our Lagrange multipliers (constrained to be $\geq\ 0$)

Regularization and The Non-separable Case

- As before, we can form the Lagrangian,

$$\mathcal{L}(\boldsymbol{w}, b, \xi, \boldsymbol{\alpha}, \boldsymbol{r}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[y_i\left(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_i + b\right) - 1 + \xi_i\right] - \sum_{i=1}^{n} r_i\xi_i$$

α_i's and r_i's are our Lagrange multipliers (constrained to be $\geq$ 0)

- Setting the derivatives with respect to $\boldsymbol{w}$ and b to zero;

$$\boldsymbol{w} = \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i = \sum_{i \in S}\alpha_i y_i \boldsymbol{x}_i \qquad 0 = \sum_{i=1}^{n}\alpha_i y_i$$

Regularization and The Non-separable Case

- As before, we can form the Lagrangian,

$$\mathcal{L}(\boldsymbol{w}, b, \xi, \boldsymbol{\alpha}, \boldsymbol{r}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[y_i\left(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_i + b\right) - 1 + \xi_i\right] - \sum_{i=1}^{n} r_i\xi_i$$

α_i's and r_i's are our Lagrange multipliers (constrained to be $\geq$ 0)

- Setting the derivatives with respect to $\boldsymbol{w}$ and b to zero;

$$\boldsymbol{w} = \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i = \sum_{i\in S}\alpha_i y_i \boldsymbol{x}_i \qquad\qquad 0 = \sum_{i=1}^{n}\alpha_i y_i$$

- Then the dual problem,

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j {\boldsymbol{x}_i}^{\boldsymbol{T}}\boldsymbol{x}_j$$

$$s.t.\ 0 \leq \alpha_i \leq C, i = 1, \ldots, n$$

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

Regularization and The Non-separable Case

- As before, we can form the Lagrangian,

$$\mathcal{L}(\boldsymbol{w}, b, \xi, \boldsymbol{\alpha}, \boldsymbol{r}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[y_i\left(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_i + b\right) - 1 + \xi_i\right] - \sum_{i=1}^{n} r_i\xi_i$$

α_i's and r_i's are our Lagrange multipliers (constrained to be $\geq\ 0$)

- Setting the derivatives with respect to $\boldsymbol{w}$ and b to zero;

$$\boldsymbol{w} = \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i = \sum_{i\in S}\alpha_i y_i \boldsymbol{x}_i \qquad 0 = \sum_{i=1}^{n}\alpha_i y_i$$

- Then the dual problem,

$$\max_{\boldsymbol{\alpha}} \textstyle\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i{}^{\boldsymbol{T}}\boldsymbol{x}_j$$

$$s.t.\ 0 \leq \alpha_i \leq C,\ i = 1, \ldots, n$$

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

Similar to the linear separable case, except that there is an upper bound C on α_i.

Extension to Non-linear Decision Boundary

- So far, we have only considered the linear decision boundary.
- How to generalize it to become nonlinear?

Extension to Non-linear Decision Boundary

- So far, we have only considered the linear decision boundary.
- How to generalize it to become nonlinear?
- **KEY**: transform $\boldsymbol{x}_i$ to a higher dimensional space to "make life easier"
  - Input space: the space the point $\boldsymbol{x}_i$'s are located
  - Feature space: the space of $\phi(\boldsymbol{x}_i)$ after transformation

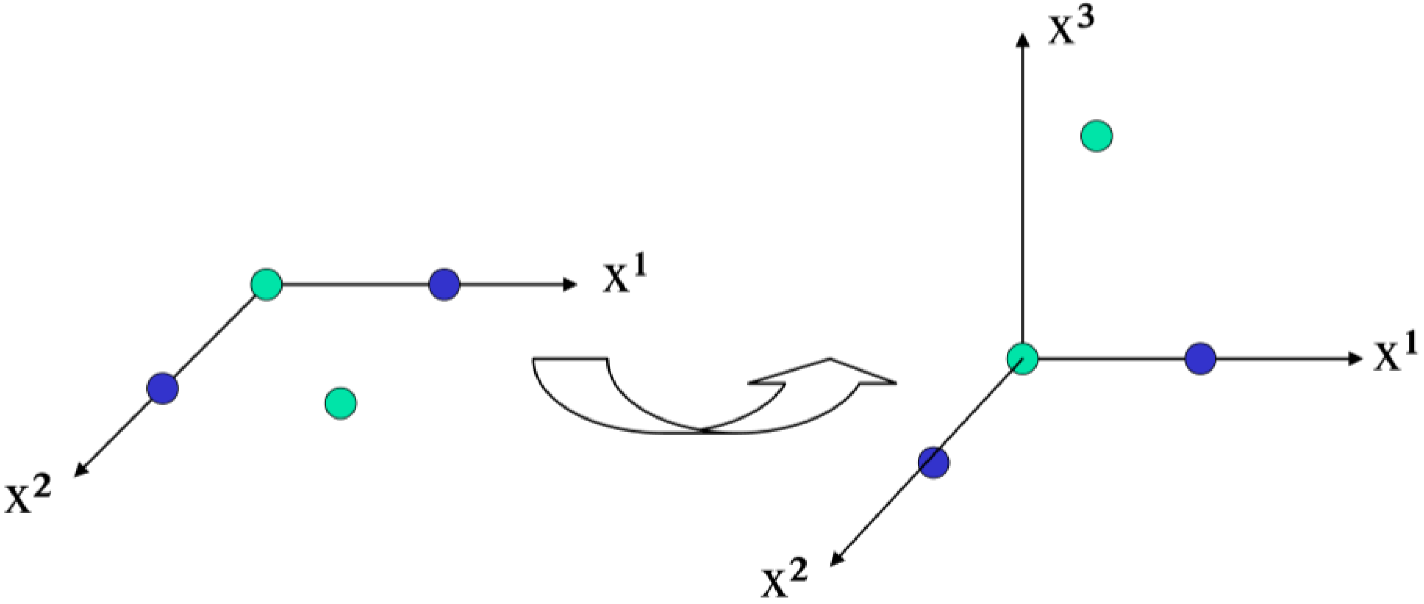
Extension to Non-linear Decision Boundary

- So far, we have only considered the linear decision boundary.
- How to generalize it to become nonlinear?
- **KEY**: transform $\boldsymbol{x}_i$ to a higher dimensional space to "make life easier"
  - Input space: the space the point $\boldsymbol{x}_i$'s are located
  - Feature space: the space of $\phi(\boldsymbol{x}_i)$ after transformation

- Why transform?
  - Linear operation in the feature space is equivalent to non-linear operation in input space.
  - Classification can be easier with a proper transformation. In the XOR problem, for example, adding a new feature of $\boldsymbol{x}_1 * \boldsymbol{x}_2$ make the problem linearly separable.

Extension to Non-linear Decision Boundary

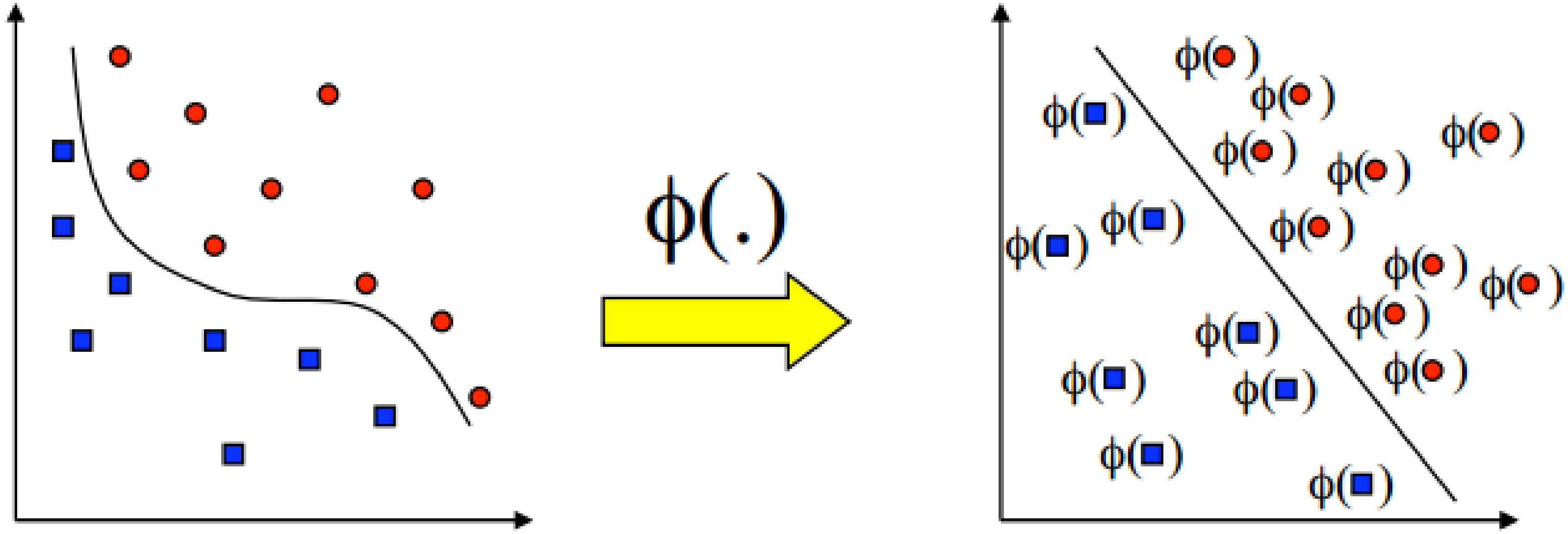
- Linear models cannot learn the XOR function
  - $f([0,1], w) = 1, f([1,0], w) = 1, f([1,1], w) = 0$, and $f([0,0], w) = 0$.
  - $f([0,1,0], w) = 1, f([1,0,0], w) = 1, f([1,1,1], w) = 0$, and $f([0,0,0], w) = 0$.



Original input space **Transformed feature space**

Transforming The Data

- Rather than applying SVM using the original input space $\boldsymbol{x}$, we instead want to learn using some feature space $\phi(\boldsymbol{x})$
- To do so, we simply need to go over our previous SVM algorithm, and replace x everywhere in it with $\phi(\boldsymbol{x})$.



Input space

Feature space

Transforming The Data

- Rather than applying SVM using the original input space $\boldsymbol{x}$, we instead want to learn using some feature space $\phi(\boldsymbol{x})$
- To do so, we simply need to go over our previous SVM algorithm, and replace x everywhere in it with $\phi(\boldsymbol{x})$.

Train SVM:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i{}^{\boldsymbol{T}} \boldsymbol{x}_j$$

$$s.t.\ 0 \leq \alpha_i \leq C\boldsymbol{,}\ i = 1, \dots, n$$

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

Test SVM:

$$\boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i = \sum_{i \in S} \alpha_i y_i \boldsymbol{x}_i$$

$$f(\boldsymbol{x}) = sign\left(\boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{x} + b\right)$$

$$= sign\left(\left(\sum_{i \in S} \alpha_i y_i \boldsymbol{x}_i{}^{\boldsymbol{T}}\, \boldsymbol{x}\right) + b\right)$$

Transforming The Data

- Rather than applying SVM using the original input space $\boldsymbol{x}$, we instead want to learn using some feature space $\phi(\boldsymbol{x})$
- To do so, we simply need to go over our previous SVM algorithm, and replace x everywhere in it with $\phi(\boldsymbol{x})$.

Train SVM:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j \boldsymbol{x}_i{}^{\boldsymbol{T}} \boldsymbol{x}_j \longrightarrow \boxed{\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)}$$

$$s.t.\ 0 \le \alpha_i \le C\boldsymbol{,}\ i = 1, \dots, n$$

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

Test SVM:

$$\boldsymbol{w}^* = \textstyle\sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i = \sum_{i\in S} \alpha_i y_i \boldsymbol{x}_i$$

$$f(\boldsymbol{x}) = sign\left(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b\right)$$

$$= sign\left(\left(\textstyle\sum_{i\in S} \alpha_i y_i \boldsymbol{x}_i{}^{\boldsymbol{T}}\, \boldsymbol{x}\right) + b\right) \longrightarrow \boxed{\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x})}$$

Transforming The Data

- The data points only appear as inner product
- As long as we can calculate the inner product $\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$ in the feature space, we do not need the mapping $\phi(\boldsymbol{x}_i)$ explicitly!

Kernel trick comes to rescue

Transforming The Data

The Kernel Trick

- Define the kernel function K by

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^{\boldsymbol{T}}\phi(\boldsymbol{x}_j)$$

Train SVM:

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

$$s.t.\ 0 \le \alpha_i \le C, i = 1, \dots, n$$

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

Test SVM:

$$\boldsymbol{w}^* = \textstyle\sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i = \sum_{i\in S} \alpha_i y_i \boldsymbol{x}_i$$

$$f(\boldsymbol{x}) = sgn(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b)$$

$$= sgn\left(\left(\textstyle\sum_{i\in S} \alpha_i y_i \boldsymbol{x}_i{}^{\boldsymbol{T}}\boldsymbol{x}\right) + b\right) \qquad \boxed{K(\boldsymbol{x}_i, \boldsymbol{x})}$$

The Kernel Trick

Interesting: $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ may be very inexpensive to calculate, even though $\phi(\boldsymbol{x}_i)$ itself may be very expensive to calculate (it can be an extremely high dimensional vector).

Example1: Suppose $\boldsymbol{x_i} \in \mathbb{R}^h$, and consider $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i{}^{\boldsymbol{T}} \boldsymbol{x}_j)^2$

We can also write this as

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(\sum_{p=1}^{h} \boldsymbol{x}_{ip}\boldsymbol{x}_{jp}\right)\left(\sum_{q=1}^{h} \boldsymbol{x}_{iq}\boldsymbol{x}_{jq}\right) = \sum_{p=1}^{h}\sum_{q=1}^{h} \boldsymbol{x}_{ip}\boldsymbol{x}_{iq}\,\boldsymbol{x}_{jp}\boldsymbol{x}_{jq} = \sum_{p,q=1}^{h} (\boldsymbol{x}_{ip}\boldsymbol{x}_{iq})(\boldsymbol{x}_{jp}\boldsymbol{x}_{jq})$$

Let $h = 3$ (feature dimension), and $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^{\boldsymbol{T}} \phi(\boldsymbol{x}_j)$,

Then $\phi(\boldsymbol{x}_i) = ?$

The Kernel Trick

More interesting: $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ may be very inexpensive to calculate, even though $\phi(\boldsymbol{x}_i)$ itself may be very expensive to calculate (it can be an extremely high dimensional vector).

Example1: Consider $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = ({\boldsymbol{x}_i}^{\boldsymbol{T}} \boldsymbol{x}_j)^2 = \sum_{p,q=1}^{h} (\boldsymbol{x}_{ip}\boldsymbol{x}_{iq})(\boldsymbol{x}_{jp}\boldsymbol{x}_{jq})$

Let $h = 3$ (feature dimension), and we have $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^{\boldsymbol{T}} \phi(\boldsymbol{x}_j)$,

$$\phi(\boldsymbol{x}_i) = \begin{bmatrix} x_{i1}x_{i1} \\ x_{i1}x_{i2} \\ x_{i1}x_{i3} \\ x_{i2}x_{i1} \\ x_{i2}x_{i2} \\ x_{i2}x_{i3} \\ x_{i3}x_{i1} \\ x_{i3}x_{i2} \\ x_{i3}x_{i3} \end{bmatrix}$$

- Calculating $\phi(\boldsymbol{x}_i)$ requires $\mathcal{O}(h^2)$
- Calculating $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ takes only $\mathcal{O}(h)$

The Kernel Trick

More interesting: $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ may be very inexpensive to calculate, even though $\phi(\boldsymbol{x}_i)$ itself may be very expensive to calculate (it can be an extremely high dimensional vector).

Example2: Consider $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i{}^{\boldsymbol{T}}\boldsymbol{x}_j + c)^2$

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{p,q=1}^{h} (\boldsymbol{x}_{ip}\boldsymbol{x}_{iq})(\boldsymbol{x}_{jp}\boldsymbol{x}_{jq}) + \sum_{p=1}^{h} (\sqrt{2c}\boldsymbol{x}_{ip})(\sqrt{2c}\boldsymbol{x}_{jp}) + c^2$$

Still set $h = 3$

$$\phi(\boldsymbol{x}_i) = [x_{i1}x_{i1}, x_{i1}x_{i2}, x_{i1}x_{i3}, x_{i2}x_{i1}, x_{i2}x_{i2}, x_{i2}x_{i3}, x_{i3}x_{i1}, x_{i3}x_{i2}, x_{i3}x_{i3}, \sqrt{2c}x_{i1}, \sqrt{2c}x_{i2}, \sqrt{2c}x_{i3}, c]^T$$

In fact, we never need to explicitly represent feature vectors in this very high dimensional feature space.

A Slightly Different View

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^{\boldsymbol{T}} \phi(\boldsymbol{x}_j)$$

A Slightly Different View

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^{\boldsymbol{T}} \phi(\boldsymbol{x}_j)$$

Intuition:

- If $\phi(\boldsymbol{x}_i)$ and $\phi(\boldsymbol{x}_j)$ are close, we might want $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to be large.
- If $\phi(\boldsymbol{x}_i)$ and $\phi(\boldsymbol{x}_j)$ are far apart (nearly orthogonal), we might want $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to be small.

A Slightly Different View

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^{\boldsymbol{T}} \phi(\boldsymbol{x}_j)$$

Intuition:

- If $\phi(\boldsymbol{x}_i)$ and $\phi(\boldsymbol{x}_j)$ are close, we might want $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to be large.
- If $\phi(\boldsymbol{x}_i)$ and $\phi(\boldsymbol{x}_j)$ are far apart (nearly orthogonal), we might want $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to be small.

□ We can think of $K(x_i, x_j)$ as a measurement of how similar are $\phi(x_i)$ and $\phi(x_j)$.

A Slightly Different View

Now you need to come up with some function $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ that you think might be a reasonable measure of how similar $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are.

A Slightly Different View

Now you need to come up with some function $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ that you think might be a reasonable measure of how similar $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are.

Q: How about this one ? $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right)$

A Slightly Different View

Now you need to come up with some function $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ that you think might be a reasonable measure of how similar $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are.

Q: How about this one ? $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right)$

A: YES!

This is the Gaussian kernel, which corresponds to an infinite dimensional feature mapping ϕ.

Gaussian Kernel

Gaussian kernel: an infinite dimensional feature mapping ϕ.

$$K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \exp\left(-\frac{\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{2\sigma^2}\right)$$

Gaussian Kernel

Gaussian kernel: an infinite dimensional feature mapping ϕ.

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right) \quad \Leftrightarrow \quad K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right), \gamma > 0$$

Gaussian Kernel

Gaussian kernel: an infinite dimensional feature mapping ϕ.

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right) \Leftrightarrow K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right), \gamma > 0$$

Let x be a scalar (1-D), $\gamma = 1$, $K(x_i, x_j) = \exp\left(-(x_i - x_j)^2\right)$

$$K(x_i, x_j) = \exp\left(-(x_i - x_j)^2\right)$$

$$=$$

$$=$$

$$= \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

Gaussian Kernel

Gaussian kernel: an infinite dimensional feature mapping ϕ.

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right) \Leftrightarrow K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right), \gamma > 0$$

Let x be a scalar (1-D),$\gamma = 1$, $K(x_i, x_j) = \exp\left(-(x_i - x_j)^2\right)$

$$K(x_i, x_j) = \exp\left(-(x_i - x_j)^2\right) = \exp(-(x_i)^2) \exp\left(-(x_j)^2\right) \exp\left((2x_i x_j)\right)$$

$$=$$

$$=$$

$$= \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

Gaussian Kernel

$$e^x = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \frac{1}{5!}x^5 + \frac{1}{6!}x^6 + \cdots\cdots$$

Gaussian kernel: an infinite dimensional feature mapping ϕ.

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right) \quad \Leftrightarrow \quad K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right), \gamma > 0$$

Let x be a scalar (1-D),$\gamma = 1$, $K(x_i, x_j) = \exp\left(-(x_i - x_j)^2\right)$

$$K(x_i, x_j) = \exp\left(-(x_i - x_j)^2\right) = \exp(-(x_i)^2) \exp\left(-(x_j)^2\right) \exp\left((2x_i x_j)\right)$$

$$= \exp(-(x_i)^2) \exp\left(-(x_j)^2\right) \sum_{k=0}^{\infty} \frac{(2x_i x_j)^k}{k!}$$

$$= \sum_{k=0}^{\infty} \exp(-(x_i)^2) \exp\left(-(x_j)^2\right) \sqrt{\frac{2^k}{k!}}(x_i)^k \sqrt{\frac{2^k}{k!}}(x_j)^k$$

$$= \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

Gaussian Kernel

Gaussian kernel: an infinite dimensional feature mapping ϕ.

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right) \quad \Leftrightarrow \quad K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right), \gamma > 0$$

Let x be a scalar (1-D),$\gamma = 1$, $K(x_i, x_j) = \exp\left(-(x_i - x_j)^2\right)$

$$K(x_i, x_j) = \exp\left(-(x_i - x_j)^2\right) = \exp(-(x_i)^2) \exp\left(-(x_j)^2\right) \exp\left((2x_i x_j)\right)$$

$$= \exp(-(x_i)^2) \exp\left(-(x_j)^2\right) \sum_{k=0}^{\infty} \frac{(2x_i x_j)^k}{k!}$$

$$= \sum_{k=0}^{\infty} \exp(-(x_i)^2) \exp\left(-(x_j)^2\right) \sqrt{\frac{2^k}{k!}}(x_i)^k \sqrt{\frac{2^k}{k!}}(x_j)^k$$

$$= \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

with infinite dimensional $\phi(\boldsymbol{x}_i) = \exp(-(x_i)^2)\left[1, \sqrt{\frac{2}{1!}}x_i, \sqrt{\frac{2^2}{2!}}(x_i)^2, \ldots\right]^T$

Gaussian SVM: Achieve large margin in infinite-dim space.

Gaussian Kernel

Gaussian kernel: an infinite dimensional feature mapping ϕ.

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right) \quad \Leftrightarrow \quad K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right), \gamma > 0$$

Testing $f(\boldsymbol{x}) = sign(\boldsymbol{w}^T\boldsymbol{x} + b) = sign\left(\sum_{i \in S} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b\right)$

$$= sign\left(\sum_{i \in S} \alpha_i y_i exp(-\gamma\|\boldsymbol{x} - \boldsymbol{x}_i\|^2) + b\right)$$

Gaussian Kernel

Gaussian kernel: an infinite dimensional feature mapping ϕ.

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right) \iff K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right), \gamma > 0$$

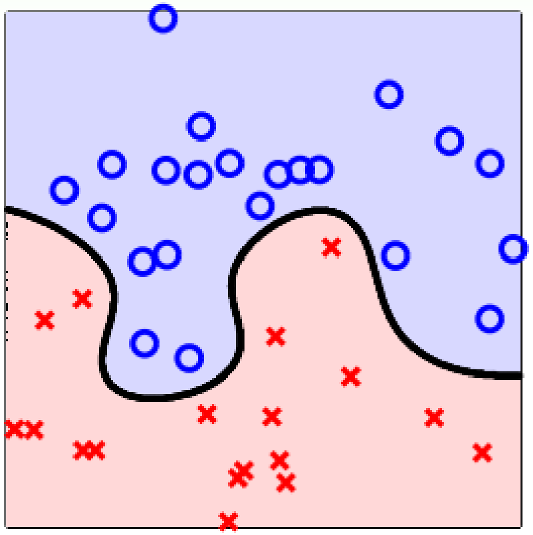
Testing $f(\boldsymbol{x}) = sign(\boldsymbol{w}^T\boldsymbol{x} + b) = sign\left(\sum_{i \in S} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b\right)$

$$= sign\left(\sum_{i \in S} \alpha_i y_i exp(-\gamma\|\boldsymbol{x} - \boldsymbol{x}_i\|^2) + b\right)$$

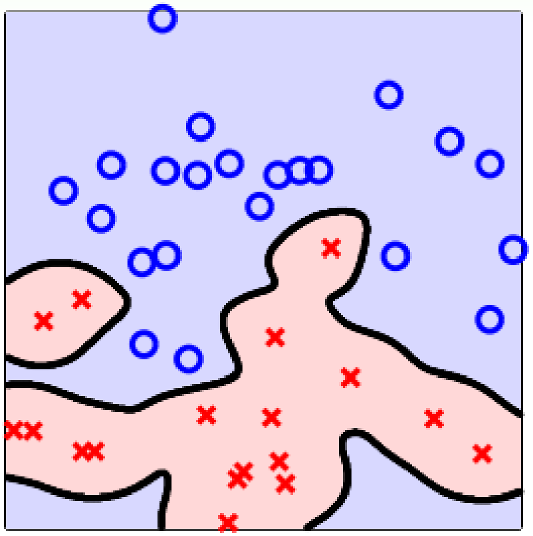
Gaussian SVM: find α_i to combine Gaussians centered at SVs $\boldsymbol{x}_i$
Also called *Radial Basis Function* (RBF) kernel.

Gaussian Kernel

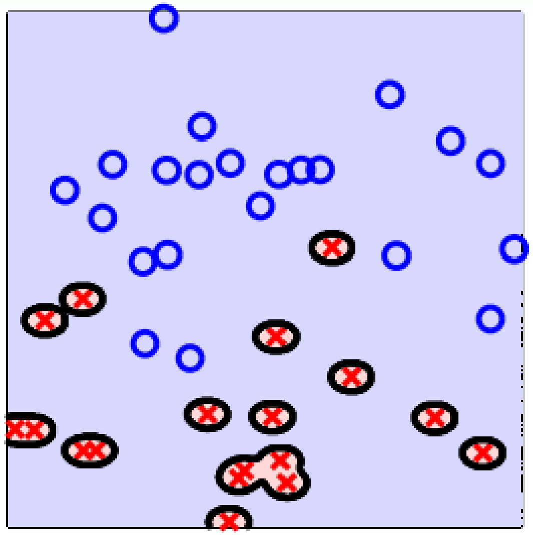
- large $\gamma \Rightarrow$ sharp Gaussians $\Rightarrow$ 'overfit'?
- **Warning: SVM can still overfit :-(**



$\exp(-1\|\mathbf{x}-\mathbf{x}'\|^2)$



$\exp(-10\|\mathbf{x}-\mathbf{x}'\|^2)$



$\exp(-100\|\mathbf{x}-\mathbf{x}'\|^2)$

Gaussian SVM: need careful selection of γ

A Slightly Different View

Now you need to come up with some function $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ that you think might be a reasonable measure of how similar $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are.

Q: How about this one ? $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right)$

A: YES!

This is the Gaussian kernel, which corresponds to an infinite dimensional feature mapping ϕ.

But more broadly, given some function K, how can we tell if it's a valid kernel?

I.e., can we tell if there is some feature mapping ϕ so that $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^{\boldsymbol{T}} \phi(\boldsymbol{x}_j)$ for all $\boldsymbol{x}_i, \boldsymbol{x}_j$.

Condition for a Valid Kernel

- Given the training dataset $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n\}$.
- Let $K_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ be the (i, j)-entry of $K \in \mathbb{R}^{n \times n}$.
- K is called the *Kernel matrix*.
- If K is a valid kernel, then ??

Condition for a Valid Kernel

- Given the training dataset $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$.
- Let $K_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ be the (i, j)-entry of $K \in \mathbb{R}^{n\times n}$.
- K is called the *Kernel matrix*.
- If K is a valid kernel, then $K_{ij} = \phi(\boldsymbol{x}_i)^{\boldsymbol{T}}\phi(\boldsymbol{x}_j) =$ ${}_{ji}$.
- K is symmetric. K is positive semi-definite.

WHY?

Condition for a Valid Kernel

- Given the training dataset $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$.
- Let $K_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ be the (i, j)-entry of $K \in \mathbb{R}^{n\times n}$.
- K is called the *Kernel matrix*.
- If K is a valid kernel, then $K_{ij} = \phi(\boldsymbol{x}_i)^{\boldsymbol{T}}\phi(\boldsymbol{x}_j) =$ ${}_{ji}$.
- K is symmetric. K is positive semi-definite.

WHY?

$$
\begin{aligned}
z^T K z &= \sum_i \sum_j z_i K_{ij} z_j = \sum_i \sum_j z_i \phi(\boldsymbol{x}_i)^{\boldsymbol{T}} \phi(\boldsymbol{x}_j) z_j \\
&= \sum_i \sum_j z_i \sum_k \phi_k(\boldsymbol{x}_i)\phi_k(\boldsymbol{x}_j)\, z_j = \sum_i \sum_j \sum_k z_i \phi_k(\boldsymbol{x}_i)\phi_k(\boldsymbol{x}_j) z_j \\
&= \sum_k \left(\sum_i z_i \phi_k(\boldsymbol{x}_i)\right)^2 \geq 0
\end{aligned}
$$

ϕ_k: the k-th coordinate of vector ϕ.

K is a valid kernel. ⟺ K is positive semi-definite.

Condition for a Valid Kernel

Theorem (Mecer). Let $K \in \mathbb{R}^d \times \mathbb{R}^d \longmapsto \mathbb{R}$ be given. Then for K to be a valid (Mercer) kernel, it is necessary and sufficient that for any $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}, (n < \infty)$, the corresponding kernel matrix is symmetric positive semi-definite.

Given a function K, apart from trying to find a mapping ϕ for it, this theorem gives another way of testing if it is a valid kernel.

Condition for a Valid Kernel

Theorem (Mecer). Let $K \in \mathbb{R}^d \times \mathbb{R}^d \longmapsto \mathbb{R}$ be given. Then for K to be a valid (Mercer) kernel, it is necessary and sufficient that for any $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n\}, (n < \infty)$, the corresponding kernel matrix is symmetric positive semi-definite.

Given a function K, apart from trying to find a mapping ϕ for it, this theorem gives another way of testing if it is a valid kernel.

Q: Which of the following is not a valid kernel? (Hint: consider two 1-dimensional vectors $x_1 = 1, x_2 = -1$)

A: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (-1 + \boldsymbol{x}_i{}^T \boldsymbol{x}_j)^2$

B: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (0 + \boldsymbol{x}_i{}^T \boldsymbol{x}_j)^2$

C: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1 + \boldsymbol{x}_i{}^T \boldsymbol{x}_j)^2$

D: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (-1 - \boldsymbol{x}_i{}^T \boldsymbol{x}_j)^2$

Condition for a Valid Kernel

Theorem (Mecer). Let $K \in \mathbb{R}^d \times \mathbb{R}^d \longmapsto \mathbb{R}$ be given. Then for K to be a valid (Mercer) kernel, it is necessary and sufficient that for any $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n\}, (n < \infty)$, the corresponding kernel matrix is symmetric positive semi-definite.

Given a function K, apart from trying to find a mapping ϕ for it, this theorem gives another way of testing if it is a valid kernel.

Q: Which of the following is not a valid kernel? (Hint: consider two 1-dimensional vectors $x_1 = 1, x_2 = -1$)

A: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (-1 + \boldsymbol{x}_i{}^T \boldsymbol{x}_j)^2$

B: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (0 + \boldsymbol{x}_i{}^T \boldsymbol{x}_j)^2$

C: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1 + \boldsymbol{x}_i{}^T \boldsymbol{x}_j)^2$

D: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (-1 - \boldsymbol{x}_i{}^T \boldsymbol{x}_j)^2$

Kernel Examples

- Polynomial kernel with degree $d \geq 1, \gamma > 0, c \geq 0$

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\gamma \boldsymbol{x}_i{}^{\boldsymbol{T}} \boldsymbol{x}_j + c)^d$$

$\alpha = d = 1$ → linear kernel

Q: Which of the following transform can be used to derive the 2nd polynomial kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\gamma \boldsymbol{x}_i{}^{\boldsymbol{T}} \boldsymbol{x}_j + c)^2$?

A: $\phi(\boldsymbol{x}) = \left[1, \sqrt{2\gamma}x_1, \ldots, \sqrt{2\gamma}x_h, \gamma x_1^2, \ldots, \gamma x_h^2\right]^T$

B: $\phi(\boldsymbol{x}) = \left[c, \sqrt{2\gamma}x_1, \ldots, \sqrt{2\gamma}x_h, x_1^2, \ldots, x_h^2\right]^T$

C: $\phi(\boldsymbol{x}) = \left[c, \sqrt{2\gamma c}x_1, \ldots, \sqrt{2\gamma c}x_h, x_1^2, \ldots, x_h^2\right]^T$

D: $\phi(\boldsymbol{x}) = \left[c, \sqrt{2\gamma c}x_1, \ldots, \sqrt{2\gamma c}x_h, \gamma x_1^2, \ldots, \gamma x_h^2\right]^T$

Kernel Examples

- Polynomial kernel with degree $d \geq 1, \gamma > 0, c \geq 0$

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\gamma \boldsymbol{x}_i{}^{\boldsymbol{T}} \boldsymbol{x}_j + c)^d$$

$\alpha = d = 1 \rightarrow$ linear kernel

Q: Which of the following transform can be used to derive the 2nd polynomial kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\gamma \boldsymbol{x}_i{}^{\boldsymbol{T}} \boldsymbol{x}_j + c)^2$?

A: $\phi(\boldsymbol{x}) = \left[1, \sqrt{2\gamma} x_1, \ldots, \sqrt{2\gamma} x_h, \gamma x_1^2, \ldots, \gamma x_h^2\right]^T$

B: $\phi(\boldsymbol{x}) = \left[c, \sqrt{2\gamma} x_1, \ldots, \sqrt{2\gamma} x_h, x_1^2, \ldots, x_h^2\right]^T$

C: $\phi(\boldsymbol{x}) = \left[c, \sqrt{2\gamma c} x_1, \ldots, \sqrt{2\gamma c} x_h, x_1^2, \ldots, x_h^2\right]^T$

D: $\phi(\boldsymbol{x}) = \left[c, \sqrt{2\gamma c} x_1, \ldots, \sqrt{2\gamma c} x_h, \gamma x_1^2, \ldots, \gamma x_h^2\right]^T$

Common Kernel

- Linear kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i{}^{\boldsymbol{T}} \boldsymbol{x}_j$
  - ✓ Safe, fast, and explainable (w and SVs)
  - □ Restricted (Not always separable)

Common Kernel

- Linear kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i{}^{\boldsymbol{T}} \boldsymbol{x}_j$
  - ✓ Safe, fast, and explainable (w and SVs)
  - ☐ Restricted (Not always separable)
- Polynomial kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\gamma \boldsymbol{x}_i{}^{\boldsymbol{T}} \boldsymbol{x}_j + c)^d, d \geq 1, \gamma > 0, c \geq 0$
  - ✓ $\alpha = d = 1 \rightarrow$ linear kernel
  - ✓ Less restricted than linear
  - ☐ Numerical difficulty for large d
  - ☐ Three parameters, difficult to select

Common Kernel

- Linear kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i{}^{\boldsymbol{T}} \boldsymbol{x}_j$
  - ✓ Safe, fast, and explainable (w and SVs)
  - □ Restricted (Not always separable)
- Polynomial kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\gamma \boldsymbol{x}_i{}^{\boldsymbol{T}} \boldsymbol{x}_j + c)^d, d \geq 1, \gamma > 0, c \geq 0$
  - ✓ $\alpha = d = 1 \rightarrow$ linear kernel
  - ✓ Less restricted than linear
  - □ Numerical difficulty for large d
  - □ Three parameters, difficult to select
- Radial basis function kernel (Gaussian kernel) with $\gamma > 0$
  - ✓ More powerful than linear/polynomial
  - ✓ One parameter, easier to select
  - □ Mysterious, slower, maybe too powerful.
  - □ One of most popular but shall be used with care

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\gamma \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2\right)$$

Common Kernel

- Linear kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i{}^{\boldsymbol{T}} \boldsymbol{x}_j$
  - ✓ Safe, fast, and explainable (w and SVs)
  - □ Restricted (Not always separable)
- Polynomial kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\gamma \boldsymbol{x}_i{}^{\boldsymbol{T}} \boldsymbol{x}_j + c)^d, d \geq 1, \gamma > 0, c \geq 0$
  - ✓ $\alpha = d = 1 \rightarrow$ linear kernel
  - ✓ Less restricted than linear
  - □ Numerical difficulty for large d
  - □ Three parameters, difficult to select
- Radial basis function kernel (Gaussian kernel) with $\gamma > 0$
  - ✓ More powerful than linear/polynomial

In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try.

  - □ One of most popular but shall be used with care

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\gamma \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2\right)$$

Remarks for Kernel

- The idea of kernels has significantly broader applicability than SVMs.
- If you have any learning algorithm that you can write in terms of only inner products between input vectors ${\boldsymbol{x}_i}^T \boldsymbol{x}_j$
- Then by replacing this with $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, where K is a kernel
- You can "magically" allow your algorithm to work efficiently in the high dimensional feature space corresponding to K.

- Standard linear algorithms can be generalized to its nonlinear version by going to the feature space
  - **Kernel** PCA
  - **kernel** independent component analysis (ICA)
  - **kernel** canonical correlation analysis (CCA)
  - **kernel** k-means

Modification Due to Kernel Function

- Change all inner products to kernel functions,

Train

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i{}^{T} \boldsymbol{x}_j$$

$$s.t.\ 0 \leq \alpha_i \leq C,\ i = 1, \dots, n$$

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

Test

$$f = \boldsymbol{w}^{T}\boldsymbol{x} + b$$

$$= \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i{}^{T} \boldsymbol{x} + b$$

Train

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$s.t.\ 0 \leq \alpha_i \leq C,\ i = 1, \dots, n$$

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

Test

$$f = \boldsymbol{w}^{T}\phi(\boldsymbol{x}) + b$$

$$= \sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b$$

More on Kernel Functions

- For training, since SVM only requires the value of $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, there is no restriction of the form of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$
  - $\boldsymbol{x}_i$ can be a sequence or a tree, instead of a feature vector
- $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is just a similarity measure comparing $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$

More on Kernel Functions

- For training, since SVM only requires the value of $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, there is no restriction of the form of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$
  - $\boldsymbol{x}_i$ can be a sequence or a tree, instead of a feature vector
- $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is just a similarity measure comparing $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$

- For testing, $\boldsymbol{x}$ is classified as class 1 if $f > 0$, and class 2 if $f < 0$

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \phi(\boldsymbol{x}_i) \ = \sum_{i \in S} \alpha_i y_i \phi(\boldsymbol{x}_i)$$

$$f = \boldsymbol{w}^{\boldsymbol{T}} \phi(\boldsymbol{x}) + b = \sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b = \sum_{i \in S} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b$$

More on Kernel Functions

- For training, since SVM only requires the value of $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, there is no restriction of the form of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$
  - $\boldsymbol{x}_i$ can be a sequence or a tree, instead of a feature vector
- $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is just a similarity measure comparing $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$

- For testing, $\boldsymbol{x}$ is classified as class 1 if $f > 0$, and class 2 if $f < 0$

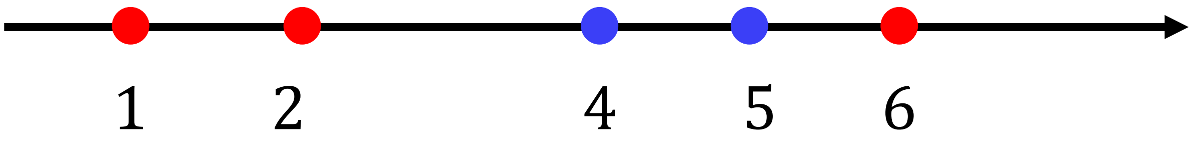
$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \phi(\boldsymbol{x}_i) \ = \sum_{i \in S} \alpha_i y_i \phi(\boldsymbol{x}_i)$$

$$f = \boldsymbol{w}^{\boldsymbol{T}} \phi(\boldsymbol{x}) + b = \sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b = \sum_{i \in S} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b$$

- For testing, the discriminant function essentially is a weighted sum of the similarity between $\boldsymbol{x}$ and the support vectors

Example

- Suppose we have 5 1-D data points
  - $x_1 = 1, x_2 = 2, x_3 = 4, x_4 = 5, x_5 = 6$
  - Labels $y_1 = 1, y_2 = 1, y_3 = -1, y_4 = -1, y_5 = 1$
- Which kernel do you want to use?



Example

- Suppose we have 5 1-D data points
  - $x_1 = 1, x_2 = 2, x_3 = 4, x_4 = 5, x_5 = 6$
  - Labels $y_1 = 1, y_2 = 1, y_3 = -1, y_4 = -1, y_5 = 1$
- We use the polynomial kernel of degree 2
  - $K(x_i, x_j) \; = (x_i x_j + 1)^2$
  - $C = 100$

Example

- Suppose we have 5 1-D data points
  - $x_1 = 1, x_2 = 2, x_3 = 4, x_4 = 5, x_5 = 6$
  - Labels $y_1 = 1, y_2 = 1, y_3 = -1, y_4 = -1, y_5 = 1$
- We use the polynomial kernel of degree 2
  - $K(x_i, x_j) \;= (x_i x_j + 1)^2$
  - $C = 100$

- We first find $\alpha_i (i = 1, \dots, 5)$ by

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{5} \alpha_i - \frac{1}{2} \sum_{i=1}^{5} \sum_{j=1}^{5} \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

$$s.t.\ 0 \leq \alpha_i \leq 100, i = 1, \dots, 5$$

$$\textstyle\sum_{i=1}^{5} \alpha_i y_i = 0$$

Example

- Using a QP solver, we get
  - $\alpha_1 = 0, \alpha_2 = 2.5, \alpha_3 = 0, \alpha_4 = 7.333, \alpha_5 = 4.833$
  - Note that the constraints are indeed satisfied
  - The support vectors are ???

Example

- Using a QP solver, we get
  - $\alpha_1 = 0, \alpha_2 = 2.5, \alpha_3 = 0, \alpha_4 = 7.333, \alpha_5 = 4.833$
  - Note that the constraints are indeed satisfied
  - The support vectors are $\{x_2 = 2, x_4 = 5, x_5 = 6\}$

Example

- Using a QP solver, we get
  - $\alpha_1 = 0, \alpha_2 = 2.5, \alpha_3 = 0, \alpha_4 = 7.333, \alpha_5 = 4.833$
  - Note that the constraints are indeed satisfied
  - The support vectors are $\{x_2 = 2, x_4 = 5, x_5 = 6\}$

- The discriminant function: $f = \boldsymbol{w}^{\boldsymbol{T}} \phi(\boldsymbol{x}) + b = \sum_{i \in S} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b$

Example

- Using a QP solver, we get
  - $\alpha_1 = 0, \alpha_2 = 2.5, \alpha_3 = 0, \alpha_4 = 7.333, \alpha_5 = 4.833$
  - Note that the constraints are indeed satisfied
  - The support vectors are $\{x_2 = 2, x_4 = 5, x_5 = 6\}$

- The discriminant function: $f = \boldsymbol{w}^{\boldsymbol{T}} \phi(\boldsymbol{x}) + b = \sum_{i \in S} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b$

$$f = 2.5 * 1 * (2x + 1)^2 + 7.333 * (-1) * (5x + 1)^2 + \overset{\alpha_5}{\underline{4.833}} * \overset{y_5}{\underline{1}} * \overset{K(x, x_5)}{\underline{(6x + 1)^2}} + b$$

Example

- Using a QP solver, we get
  - $\alpha_1 = 0, \alpha_2 = 2.5, \alpha_3 = 0, \alpha_4 = 7.333, \alpha_5 = 4.833$
  - Note that the constraints are indeed satisfied
  - The support vectors are $\{x_2 = 2, x_4 = 5, x_5 = 6\}$

- The discriminant function: $f = \boldsymbol{w}^{\boldsymbol{T}} \phi(\boldsymbol{x}) + b = \sum_{i \in S} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b$

$$f = 2.5 * 1 * (2x+1)^2 + 7.333 * (-1) * (5x+1)^2 + \overset{\alpha_5}{\underline{4.833}} * \overset{y_5}{\underline{1}} * \overset{K(x, x_5)}{\underline{(6x+1)^2}} + b$$

- b is recovered by solving $f(2) = 1$ or $f(5) = -1$ or by $f(6) = 1$.
- As x_2, x_5, x_4 lie on the line $\phi(\boldsymbol{w})^T \phi(\boldsymbol{x}) + b = \pm 1$(support vectors).
- All three give $b = 9$

Example

- Using a QP solver, we get
  - $\alpha_1 = 0, \alpha_2 = 2.5, \alpha_3 = 0, \alpha_4 = 7.333, \alpha_5 = 4.833$
  - Note that the constraints are indeed satisfied
  - The support vectors are $\{x_2 = 2, x_4 = 5, x_5 = 6\}$

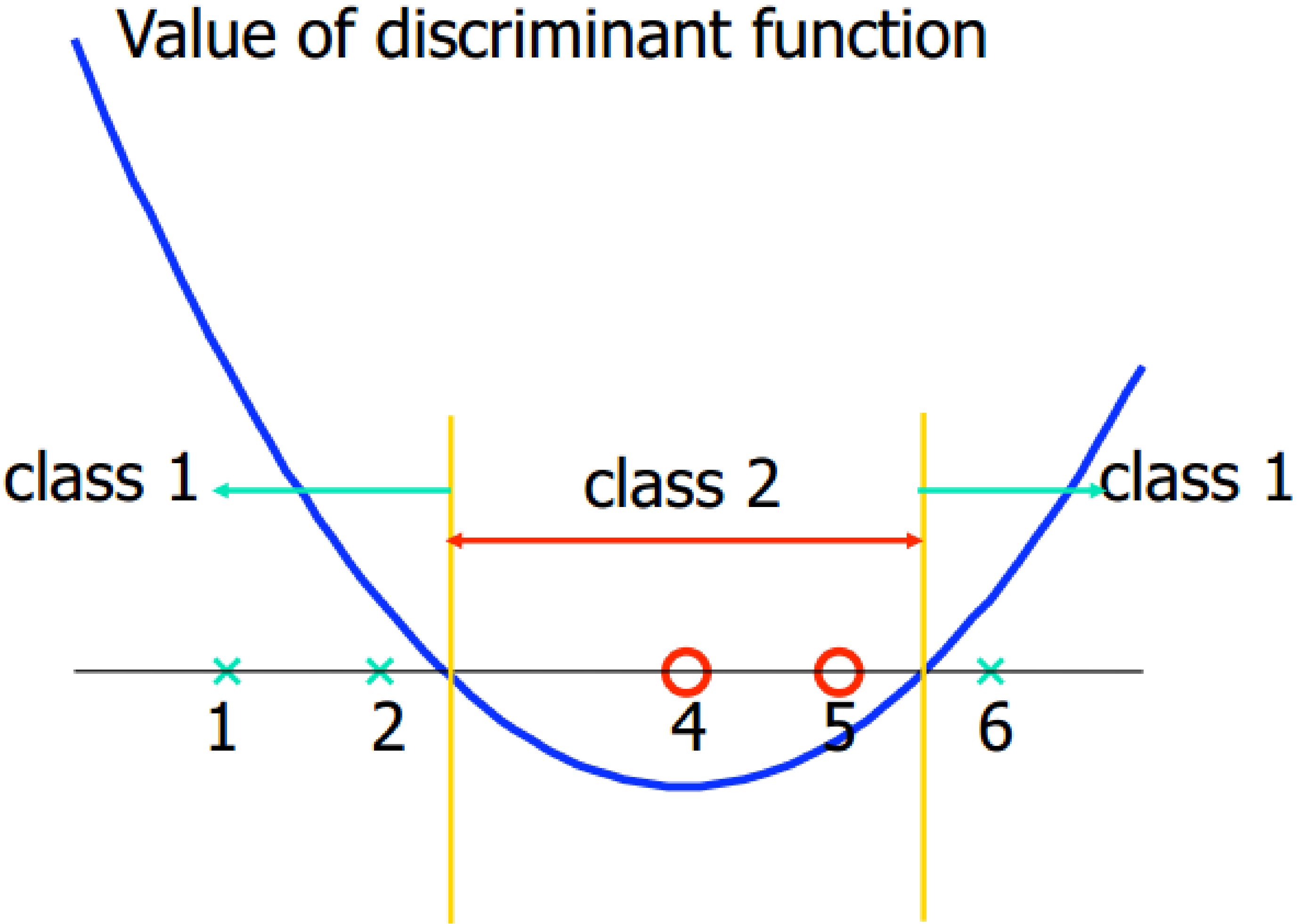
- The discriminant function: $f = \boldsymbol{w}^{\boldsymbol{T}}\phi(\boldsymbol{x}) + b = \sum_{i \in S} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b$

$$f = 2.5 * 1 * (2x+1)^2 + 7.333 * (-1) * (5x+1)^2 + \overset{\alpha_5}{\underline{4.833}} * \overset{y_5}{\underline{1}} * \overset{K(x, x_5)}{\underline{(6x+1)^2}} + b$$

- b is recovered by solving $f(2) = 1$ or $f(5) = -1$ or by $f(6) = 1$.
- As x_2, x_5, x_4 lie on the line $\phi(\boldsymbol{w})^T\phi(\boldsymbol{x}) + b = \pm 1$(support vectors).
- All three give $b = 9$

❖ $\boldsymbol{f(x) = 0.6667x^2 - 5.333x + 9}$

Example



- ❖ $f(x) = 0.6667x^2 - 5.333x + 9$

Why SVM Work?

- The feature space is often very high dimensional. Why don’t we have the curse of dimensionality?
- A classifier in a high-dimensional space has many parameters and is usually hard to estimate.

Why SVM Work?

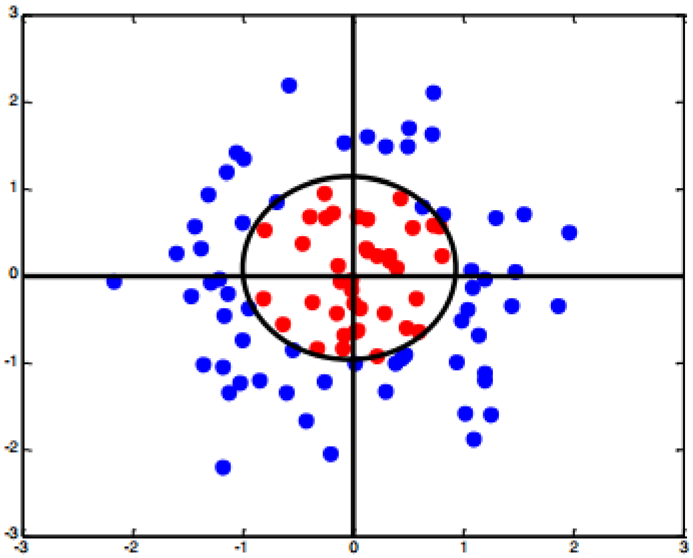
- The feature space is often very high dimensional. Why don’t we have the curse of dimensionality?
- A classifier in a high-dimensional space has many parameters and is usually hard to estimate.

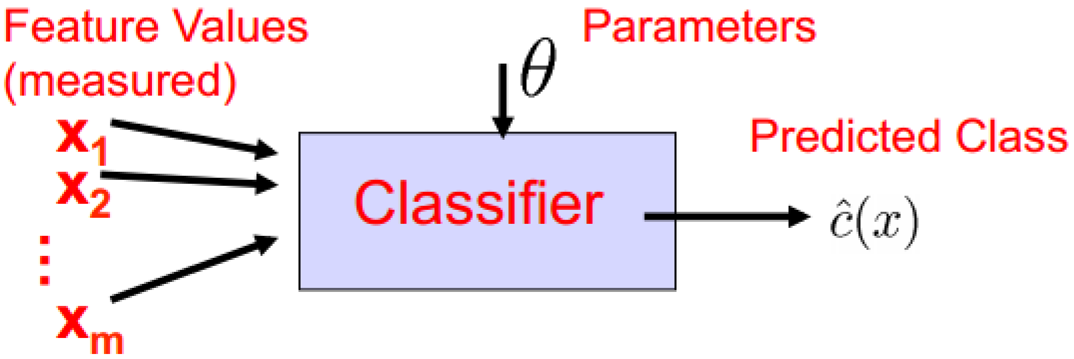
- Vapnik argues that the fundamental problem is not the number of parameters to be estimated.
- Rather, the problem is the capacity/flexibility of a classifier.
- Typically, a classifier with many parameters is very flexible, but there are also exceptions.

Now we have to introduce the VC dimension…

Learners and Complexity

Different learners have different power (capacity).





Feature Values (measured)

x_1

x_2

$\vdots$

x_m

Parameters

θ

Classifier

Predicted Class

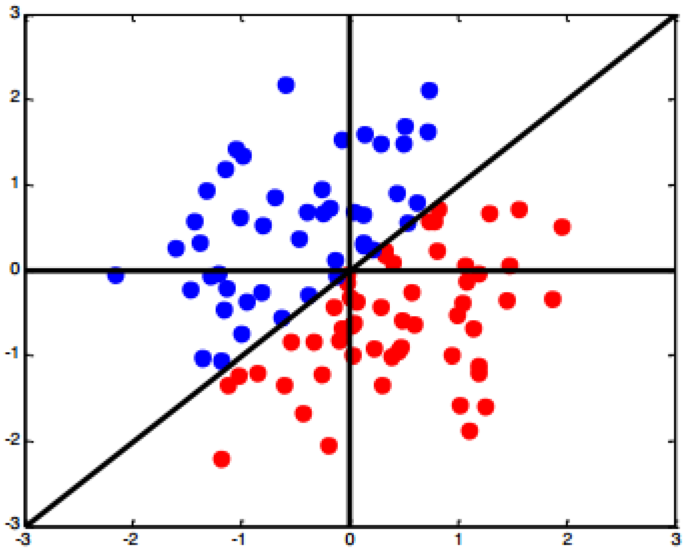
$\hat{c}(x)$

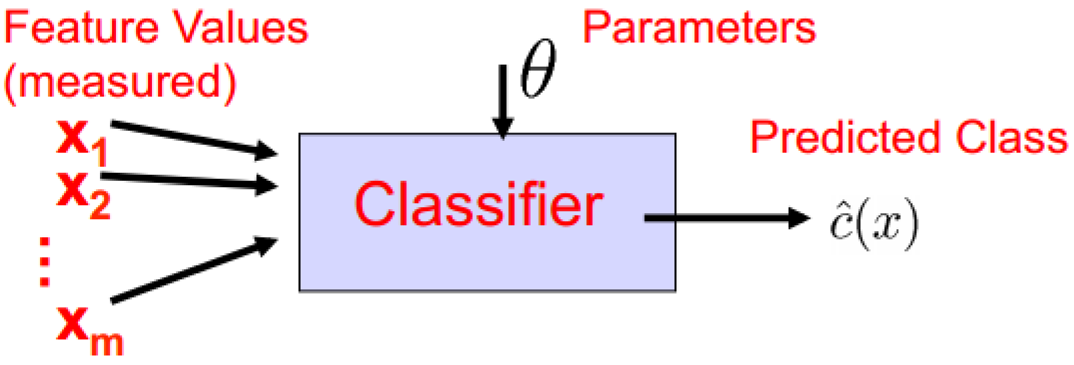
Example:

$$\hat{c}(x) = \text{sign}(x^T x - \theta_0)$$

Learners and Complexity

Different learners have different power (capacity).





Feature Values (measured)

x_1

x_2

⋮

x_m

Parameters

θ

Classifier

Predicted Class

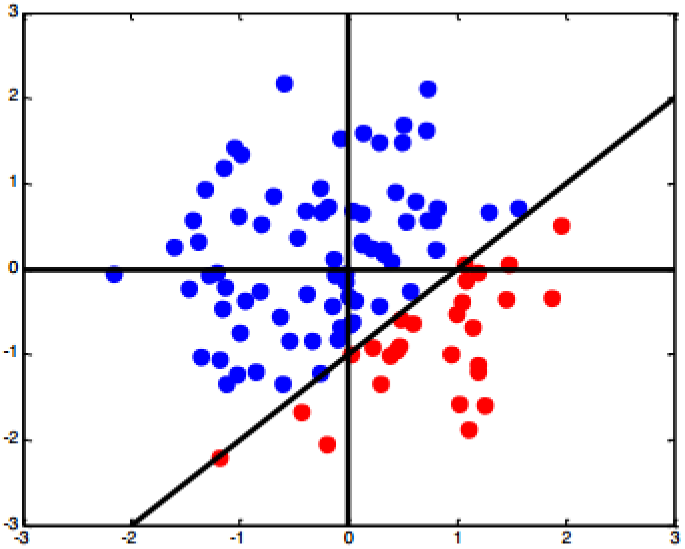
$\hat{c}(x)$

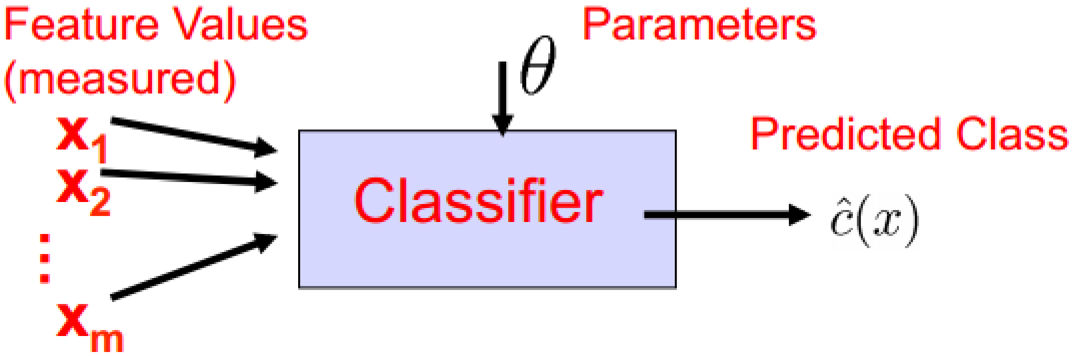
Example:

$$\hat{c}(x) = \mathrm{sign}(\theta_1 x_1 + \theta_2 x_2)$$

Learners and Complexity

Different learners have different power (capacity).





Feature Values (measured)

x_1

x_2

$\vdots$

x_m

Parameters

θ

Classifier

Predicted Class

$\hat{c}(x)$

Example:

$$\hat{c}(x) = \text{sign}(\theta_1 x_1 + \theta_2 x_2 + \theta_0)$$

Learners and Complexity

Trade-off:

- More power = more complex systems, might overfit
- Less power = will not overfit, but may not find the “best” learner

How can we quantify representation power?

Learners and Complexity

Trade-off:

- More power = more complex systems, might overfit
- Less power = will not overfit, but may not find the “best” learner

How can we quantify representation power?

One solution is VC (Vapnik-Chervonenkis) dimension

Learners and Complexity

Assume that our training data are i.i.d. from some distribution $p(x)$

Risk $$R(\theta) = \mathrm{TestError} = \mathbb{E}[\delta(c \neq \hat{c}(x\,;\,\theta))]$$

Empirical risk $$R^{\mathrm{emp}}(\theta) = \mathrm{TrainError} = \frac{1}{N}\sum_i \delta(c^{(i)} \neq \hat{c}(x^{(i)}\,;\,\theta))$$

How are these related?

Learners and Complexity

Assume that our training data are i.i.d. from some distribution $p(x)$

Risk $$R(\theta) = \text{TestError} = \mathbb{E}[\delta(c \neq \hat{c}(x\,;\,\theta))]$$

Empirical risk $$R^{\text{emp}}(\theta) = \text{TrainError} = \frac{1}{N}\sum_i \delta(c^{(i)} \neq \hat{c}(x^{(i)}\,;\,\theta))$$

How are these related?

- Under-fitting domain:
- Over-fitting domain:

Learners and Complexity

Assume that our training data are i.i.d. from some distribution $p(x)$

Risk $R(\theta) = \text{TestError} = \mathbb{E}[\delta(c \neq \hat{c}(x\,;\,\theta))]$

Empirical risk $R^{\text{emp}}(\theta) = \text{TrainError} = \frac{1}{N}\sum_i \delta(c^{(i)} \neq \hat{c}(x^{(i)}\,;\,\theta))$

How are these related?

- Under-fitting domain: pretty similar…
- Over-fitting domain: test error might be lots worse!

VC Dimension and Risk

Given some classifier, let H be its VC dimension

- Represents “capacity” of classifier

$$R(\theta) = \mathrm{TestError} = \mathbb{E}[\delta(c \neq \hat{c}(x\,;\,\theta))]$$

$$R^{\mathrm{emp}}(\theta) = \mathrm{TrainError} = \frac{1}{N}\sum_i \delta(c^{(i)} \neq \hat{c}(x^{(i)}\,;\,\theta))$$

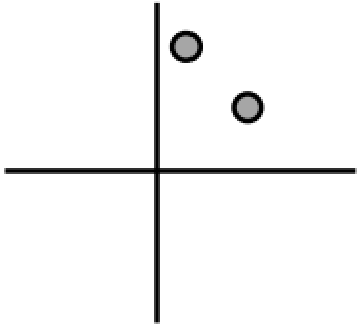
With high probability $(1-\eta)$, Vapnik showed

$$\mathrm{TestError} \leq \mathrm{TrainError} + \sqrt{\frac{H\log(2N/H) + H - \log(\eta/4)}{N}}$$

Shattering

- We say a classifier $f(x)$ can shatter points $x_1, x_2, \ldots x_N$ iff for all $y_1, \ldots y_N$, $f(x)$ can achieve zero error on the training data (x_1, y_1), (x_2, y_2),…, (x_N, y_N)
(i.e.,) there exists some θ that gets zero error.

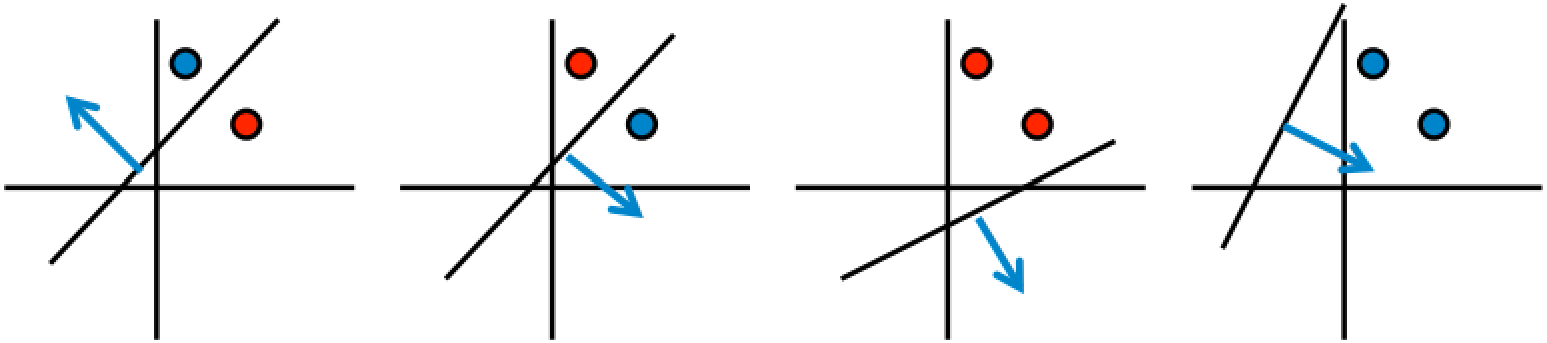
- Can $f(x; \theta) = sign(\theta x^T)$ shatter theses points?



Shattering

- We say a classifier $f(x)$ can shatter points $x_1, x_2, \ldots x_N$ iff for all $y_1, \ldots y_N$, $f(x)$ can achieve zero error on the training data (x_1, y_1), (x_2, y_2),…, (x_N, y_N)
  (i.e.,) there exists some θ that gets zero error.

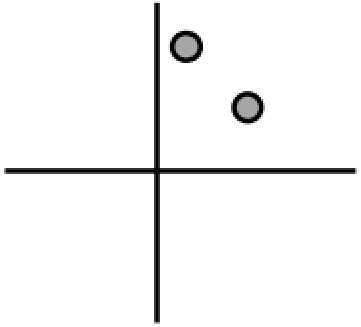
- Can $f(x;\theta) = sign(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$ shatter theses points?
- Yes: there are 4 possible training sets.



Shattering

- We say a classifier $f(x)$ can shatter points $x_1, x_2, \ldots x_N$ iff for all $y_1, \ldots y_N$, $f(x)$ can achieve zero error on the training data (x_1, y_1), (x_2, y_2),…, (x_N, y_N)
  (i.e.,) there exists some θ that gets zero error.

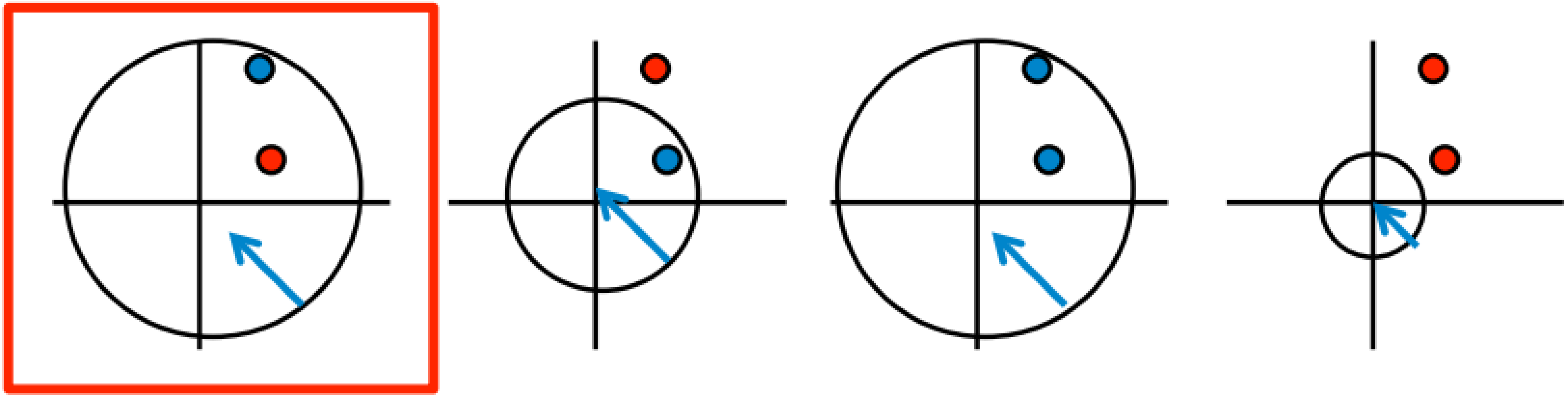
- Can $f(x; \theta) = sign(x^T x + \theta)$ shatter theses points?



Shattering

- We say a classifier $f(x)$ can shatter points $x_1, x_2, \dots x_N$ iff for all $y_1, \dots y_N, f(x)$ can achieve zero error on the training data (x_1, y_1), (x_2, y_2),…, (x_N, y_N)
  (i.e.,) there exists some θ that gets zero error.

- Can $f(x; \theta) = sign(x^T x + \theta)$ shatter theses points?



VC Dimension

- The VC dimension is defined as the maximum number of points that can be arranged so that $f(x)$ can shatter them.

VC Dimension

- The VC dimension is defined as the maximum number of points that can be arranged so that $f(x)$ can shatter them.

- **Shattering**:
  - I pick H points and place them at $x_1, x_2, \dots, x_H$
  - You challenge me with 2^H possible labeling $y_1, y_2, \dots, y_H$, $y_i \in \{-1,1\}$
  - VC dimension is the maximum number of points that I can place so that a $f(x;\theta)$ can correctly classify for the arbitrary labeling $y_1, y_2, \dots, y_H$.

VC Dimension

- The VC dimension is defined as the maximum number of points that can be arranged so that $f(x)$ can shatter them.

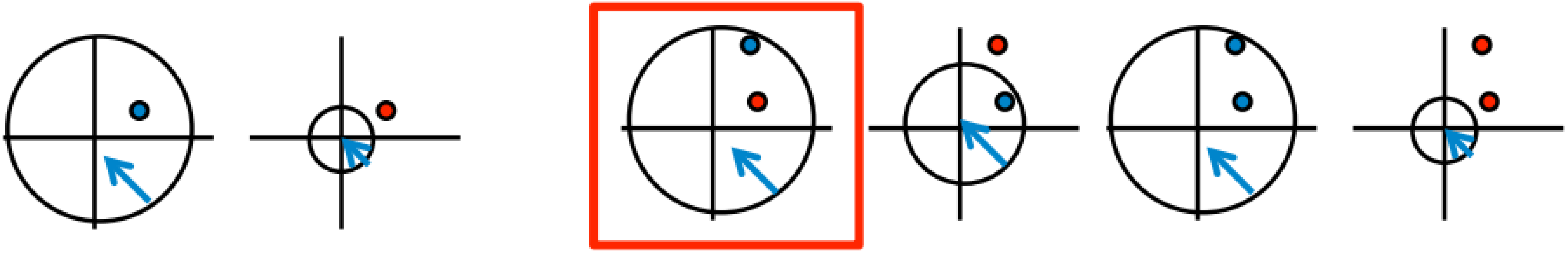
- **Shattering**:
  - I pick H points and place them at $x_1, x_2, \dots, x_H$
  - You challenge me with 2^H possible labeling $y_1, y_2, \dots, y_H$, $y_i \in \{-1,1\}$
  - VC dimension is the maximum number of points that I can place so that a $f(x;\theta)$ can correctly classify for the arbitrary labeling $y_1, y_2, \dots, y_H$.

A family of classifiers is said to have *infinite* VC dimension if it can shatter H points, no matter how large H.

VC Dimension

- The VC dimension is defined as the maximum number of points that can be arranged so that $f(x)$ can shatter them.

- Example: what is the VC dimension of $f(x;\theta) = sign(x^T x + \theta)$?
- VC dim=1: can arrange one point, cannot arrange two.



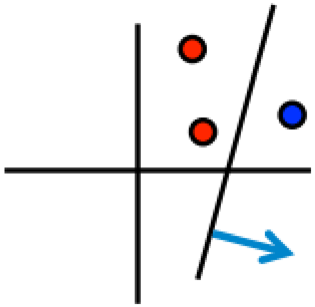
VC Dimension

- Example: what is the VC dimension of the two dimensional line, $f(x;\theta) = sign(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$?

VC Dimension

- Example: what is the VC dimension of the two dimensional line, $f(x;\theta) = sign(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$?

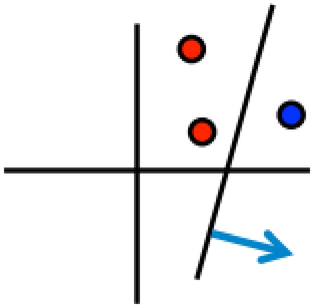
- VC dim>=3?



VC Dimension

- Example: what is the VC dimension of the two dimensional line, $f(x;\theta) = sign(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$?

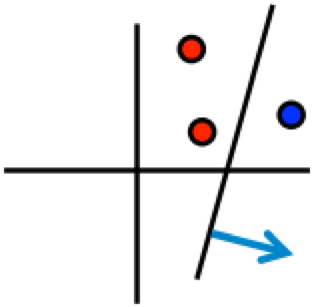
- VC dim>=3? Yes



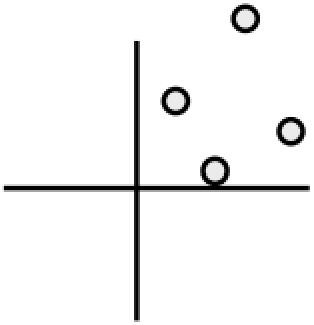
VC Dimension

- Example: what is the VC dimension of the two dimensional line, $f(x;\theta) = sign(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$?

- VC dim>=3? Yes



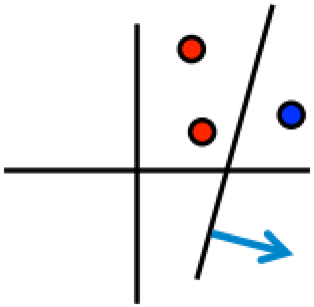
- VC dim>=4?



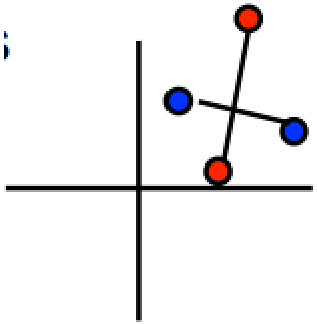
VC Dimension

- Example: what is the VC dimension of the two dimensional line, $f(x;\theta) = sign(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$?

- VC dim>=3? Yes



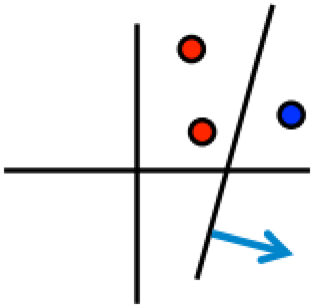
- VC dim>=4? No…



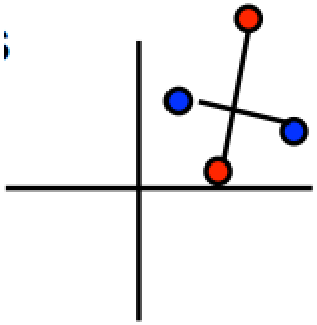
VC Dimension

- Example: what is the VC dimension of the two dimensional line, $f(x;\theta) = sign(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$?

- VC dim>=3? Yes



- VC dim>=4? No…



★ For a general, linear classifier in d dimensions with a constant term: VC dim = d+1.

VC Dimension

- The VC dimension is defined as the maximum number of points that can be shattered by $f(x)$.
- VC dimension measures the “power” of the learner
- The higher the VC-dimension, the more flexible the classifier is.

- Note that if the VC dimension is H, then there exists at least one set of H points that can be shattered, but in general it will not be true that every set of H points can be shattered.

- Does not necessarily equal the number of parameters!
  - Can define a classifier with one parameter but lots of power?

VC Dimension

Consider the one-parameter family of functions, defined as,

$$f(x, \alpha) = sign(\sin(\alpha x))$$

You choose some number H, the task is to find H points that can be shattered. I choose them to be:

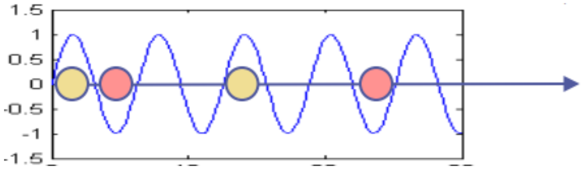
$$x_i = 10^{-i}, i = 1, \dots H$$

You specify any labels you like: $y_1, y_2, \dots, y_H,\ \ y_i \in \{-1,1\}$

Then $f(\alpha)$ gives this labeling if I choose α to be

$$\alpha = \pi\left(1 + \sum_{i=1}^{h} \frac{(1-y_i)10^i}{2}\right)$$

Thus the VC dimension of this machine is infinite.



VC Dimension

Consider the one-parameter family of functions, defined as,

$$f(x,\alpha) = sign(\sin(\alpha x))$$

You choose some number H, the task is to find H points that can be shattered. I choose them to be:

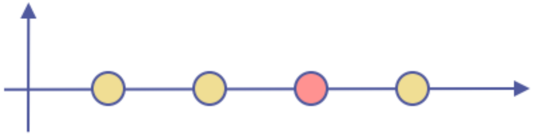
$$x_i = 10^{-i}, i = 1, \dots H$$

You specify any labels you like: $y_1, y_2, \dots, y_H,\ \ y_i \in \{-1,1\}$

Then $f(\alpha)$ gives this labeling if I choose α to be

$$\alpha = \pi\left(1 + \sum_{i=1}^{h} \frac{(1-y_i)10^i}{2}\right)$$

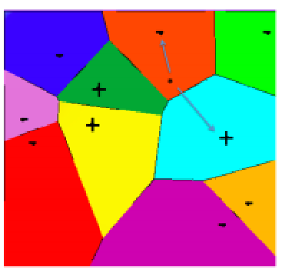
But, if I choose 4 equally spaced x's then cannot shatter



VC Dimension

- VC-dimension, however, is a theoretical concept.
- Difficult to be computed exactly in practice.
  - Qualitatively, if a classifier is flexible, it probably has a high VC-dimension.

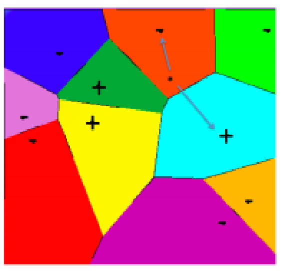
- Consider the nearest neighbor classification algorithm
  - Input a query example x
  - Finding training data x_i in $\{x_1, \ldots, x_N\}$ closest to x
  - Predict label for x as y_i



- The VC dimension of the nearest neighbor classifier is ???

VC Dimension

- VC-dimension, however, is a theoretical concept.
- Difficult to be computed exactly in practice.
  - Qualitatively, if a classifier is flexible, it probably has a high VC-dimension.

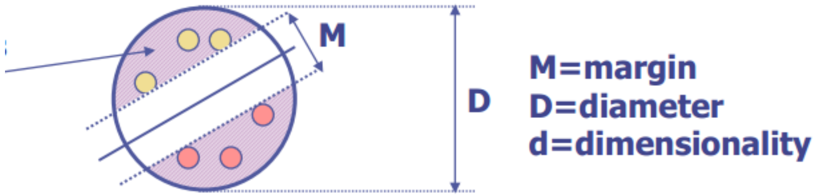


- Consider the nearest neighbor classification algorithm
  - Input a query example x
  - Finding training data x_i in $\{x_1, \ldots, x_N\}$ closest to x
  - Predict label for x as y_i
- The VC dimension of the nearest neighbor classifier is infinity, because no matter how many points you have, you can get perfect classification on training data.
- But still works well in practice
- $H = \infty \nRightarrow$ poor performance $H = \mathit{low} \Rightarrow$ good performance

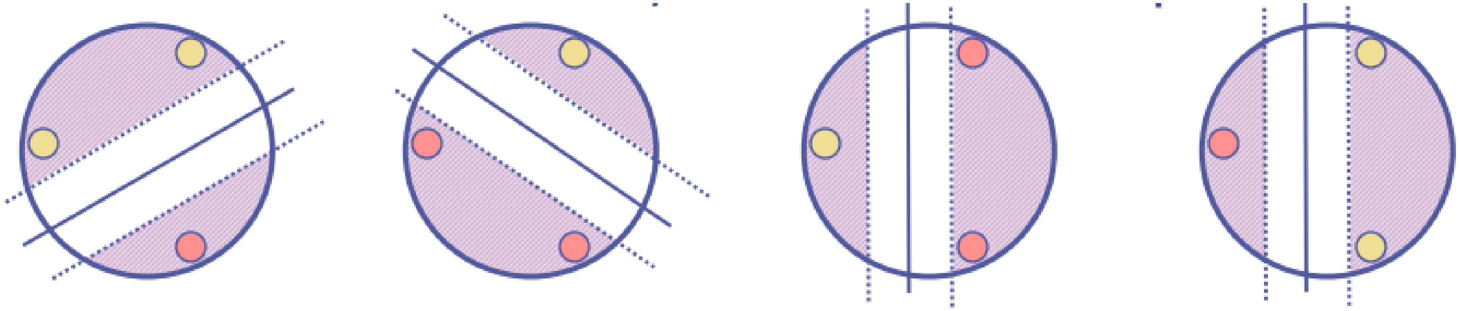
VC Dimension & Large Margins

- Linear classifiers are too big a function class, since $H = d + 1$
- Can reduce VC dimension if we restrict them
- Constrain linear classifiers to data living inside a sphere
- Gap-Tolerant classifiers: a linear classifier whose activity is constrained to a sphere & outside a margin.

Only count errors in shaded region
Elsewhere have $L(x, y, \theta) = 0$

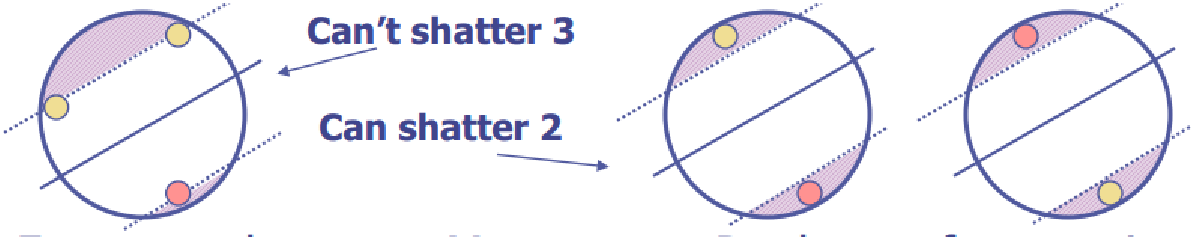


If M is small relative to D, can still shatter 3 points:

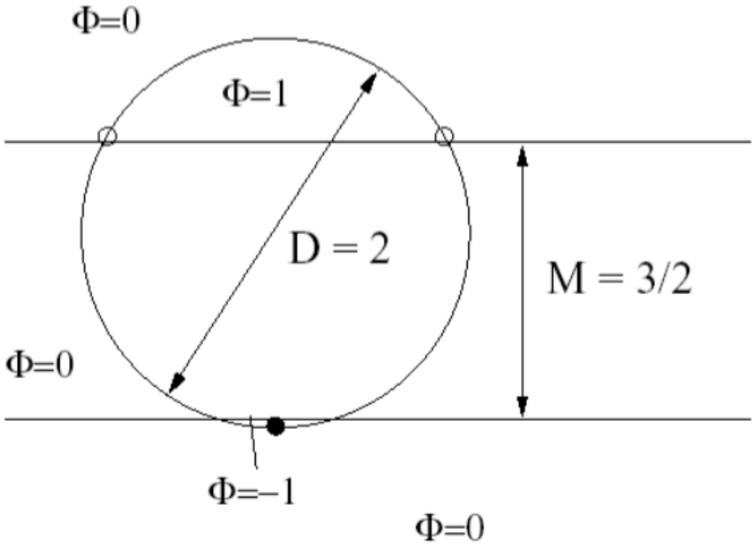


VC Dimension & Large Margins

- But as M grows relative to D, can only shatter 2 points.

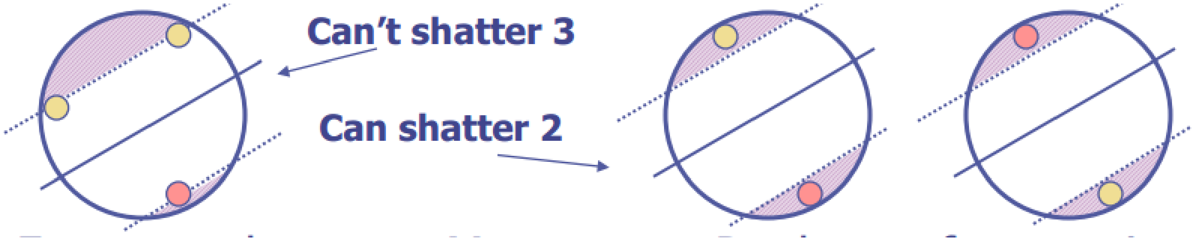


- Assume D=2, M=3/2,



VC Dimension & Large Margins

- But as M grows relative to D, can only shatter 2 points.

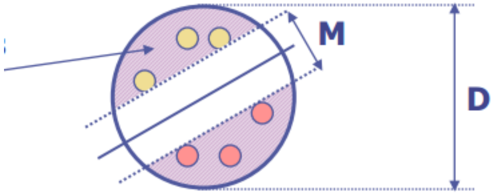


- For hyperplanes, as M grows vs. D, shatter fewer points!

VC dimension H decreases while M grows, the general formula:

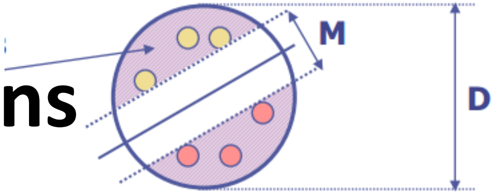
$$H \leq min\left\{\left[\frac{D^2}{M^2}\right], d\right\} + 1$$

$[x]$ refers to the least integer greater than or equal to x.



M=margin
D=diameter
d=dimensionality

VC Dimension & Large Margins



M=margin
D=diameter
d=dimensionality

VC dimension H decreases while M grows, the general formula:

$$H \leq min\left\{\left\lceil\frac{D^2}{M^2}\right\rceil, d\right\} + 1$$

$\lceil x \rceil$ refers to the least integer greater than or equal to x.

- Before, (general linear classifier) just had $H = d + 1$
- Now we have a smaller H
- If data is anywhere, D is infinite and back to $H = d + 1$
- Typically real data is bounded (by sphere), D is fixed
- Maximizing M reduces H, improving risk bound
- **Note:** $R(\theta)$ does not count errors in margin or outside sphere.

Structural Risk Minimization (SRM)

- We should find a classifier that minimizes the sum of the training error (empirical risk) and a term that is a function of the flexibility of the classifier (model complexity)

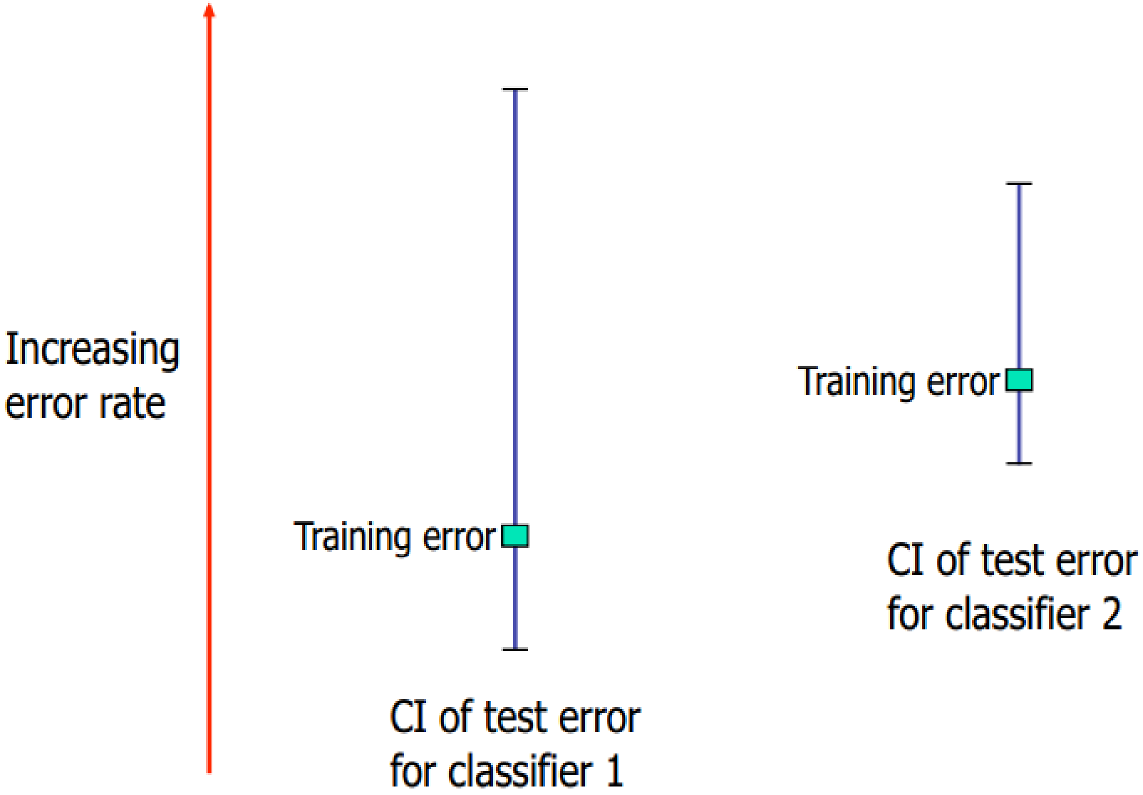
With high probability $(1-\eta)$, Vapnik showed

$$\mathrm{TestError} \leq \mathrm{TrainError} + \sqrt{\frac{H\log(2N/H) + H - \log(\eta/4)}{N}}$$

- Recall the concept of confidence interval
  - E.g., we are 99% confident that the population mean lies in the 99% confidence interval estimated from a sample.

Structural Risk Minimization (SRM)

- We can also construct a confidence interval (CI) for the generalization error.
- SRM prefers classifier 2 although it has a higher training error, because the upper limit of CI is smaller.



Structural Risk Minimization (SRM)

- SVM (Large margin classifier) can be viewed as a SRM
  - $\frac{1}{2}||\boldsymbol{w}||^2$: shrinks the parameters towards zero to avoid overfitting; related to the VC-dimension of the resulting classifier;
  - $\sum_{i=1}^{n} \xi_i$: the training error.

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

$$s.t.\ y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \geq 1 - \xi_i, i = 1,2, \dots, n$$

$$\xi_i \geq 0, i = 1,2, \dots, n$$

Summary: Steps for Classification

- Prepare the data matrix
- Select the kernel function to use
- Select the parameter of the kernel function and the value of C
  - You can use the values suggested by the SVM software, or you can set apart a validation set to determine the values of the parameter
- Execute the training algorithm and obtain the α_i
- Unseen data can be classified using the α_i and the support vectors

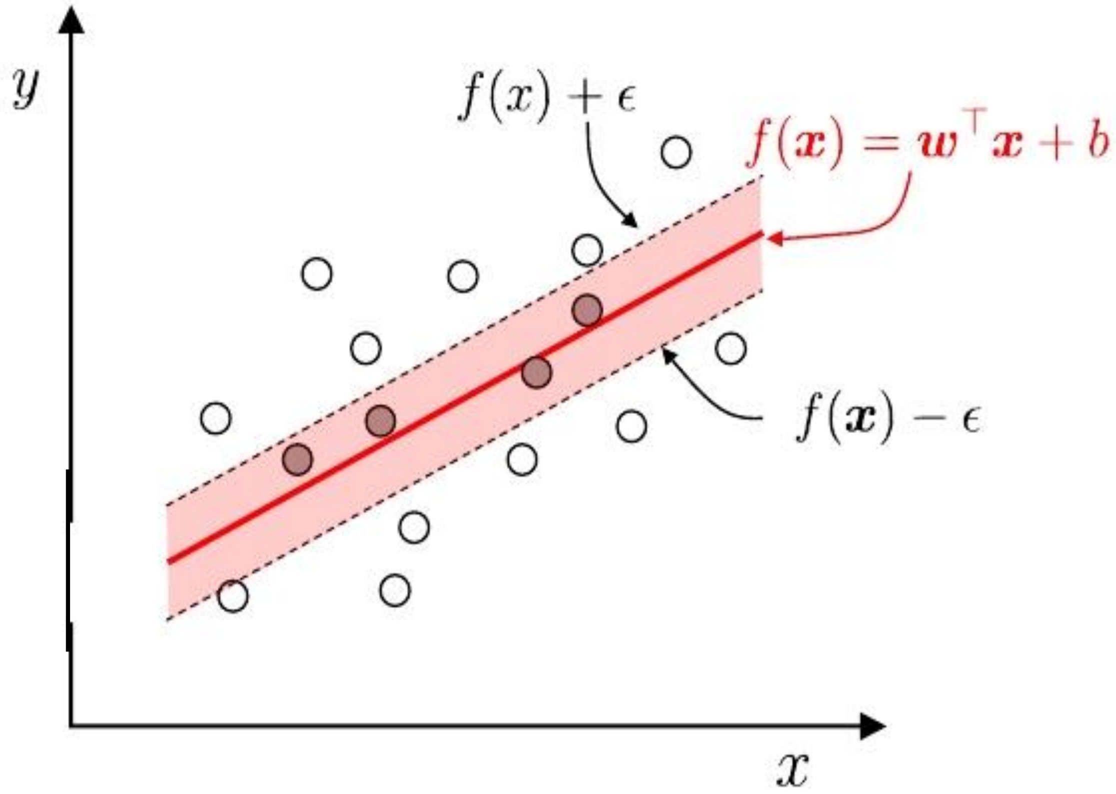
Strengths & Weaknesses of SVM

- **Strengths**
  - Training is relatively easy
    - No local optimal
  - It scales relatively well to high dimensional data
  - Tradeoff between classifier complexity and error can be controlled explicitly
  - Non-traditional data like strings and trees can be used as input to SVM, instead of feature vectors

- **Weaknesses**
  - Need to choose a "good" kernel function

Epsilon Support Vector Regression (ε-SVR)

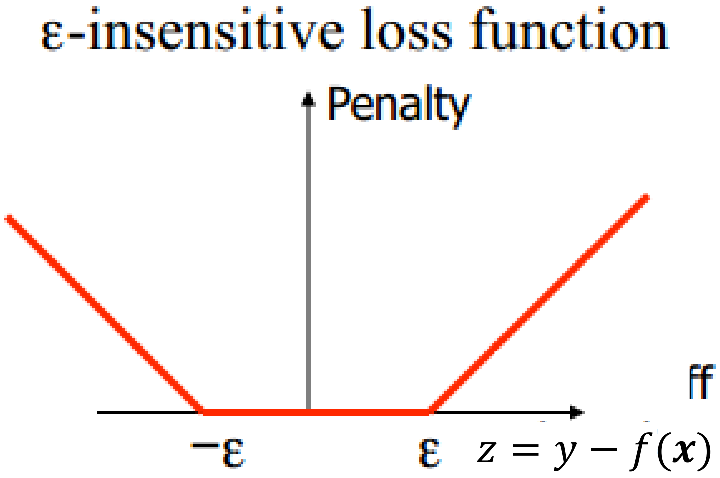
- Linear regression in feature space
- Unlike in least square regression, the error function is ε-insensitive loss function
  - Intuitively, mistake less than ε is ignored

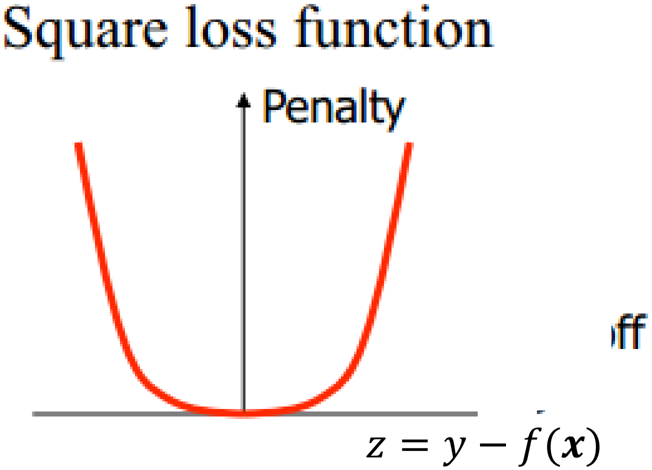


y
$f(x) + \epsilon$
$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$
$f(\boldsymbol{x}) - \epsilon$
x

Epsilon Support Vector Regression (ε-SVR)

- Linear regression in feature space
- Unlike in least square regression, the error function is ε-insensitive loss function
  - Intuitively, mistake less than ε is ignored





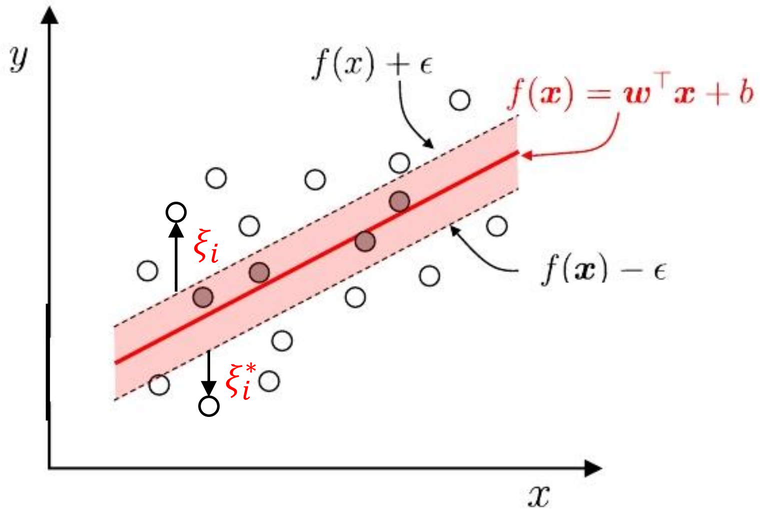
$$l(z) = \begin{cases} |z| - \varepsilon & \text{If } |z| \geq \varepsilon, \\ 0 & \text{otherwise} \end{cases}$$

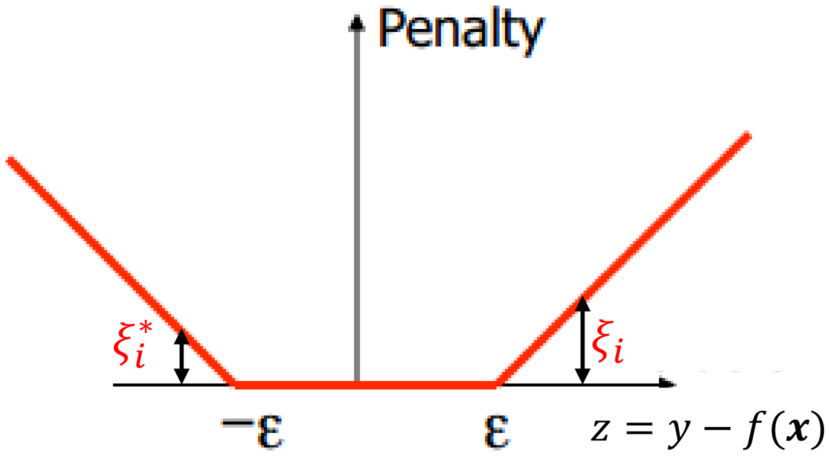
$$l(z) = z^2$$

Epsilon Support Vector Regression (ε-SVR)

- Given a data set $\{x_1, x_2, \ldots, x_n\}$ with target values $\{y_1, y_2, \ldots, y_n\}$.
- The optimization problem of ε-SVR,

$$\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \qquad \text{s.t.} \begin{cases} y_i - (\boldsymbol{w}^T\boldsymbol{x}_i + b) \le \varepsilon + \xi_i \\ (\boldsymbol{w}^T\boldsymbol{x}_i + b) - y_i \le \varepsilon + \xi_i^* \\ \xi_i^* \ge 0, \xi_i \ge 0 \end{cases}$$


y
$f(x) + \epsilon$
$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$
ξ_i
$f(\boldsymbol{x}) - \epsilon$
ξ_i^*
x



Epsilon Support Vector Regression (ε-SVR)

- C is a parameter to control the amount of influence of the error
- $\frac{1}{2}||\boldsymbol{w}||^2$ controlls the complexity of the regression function
- After training (solving the QP), we get values of α_i and α_i^*, which are both zero if $\boldsymbol{x}_i$ does not contribute to the error function
- For a new data $\boldsymbol{x}$,

$$f(\boldsymbol{x}) = \sum_{j=1}^{s} (\alpha_j - \alpha_j^*) K(\boldsymbol{x}_j, \boldsymbol{x}) + b$$

Discussion

- What is the **VC Dimension** of an **SVM** with **RBF kernel?**

- What if every training point becomes a support vector?

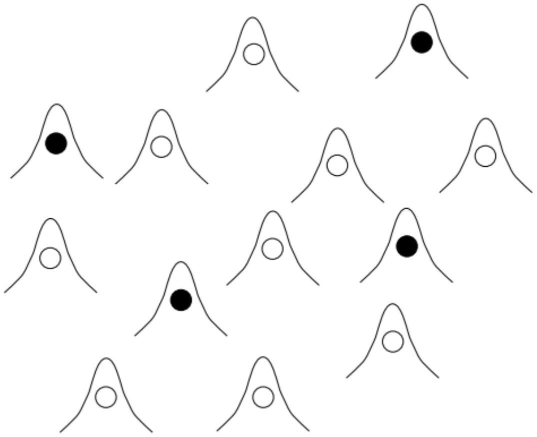
$$\text{SVs: } \alpha_i > 0, y_i(\textstyle\sum_{i\in S} \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i) + b)) = 1$$

- Gaussian RBF SVMs of sufficiently small width (large γ) can classify an arbitrarily large number of training points correctly, and thus have infinite VC dimension.

$$K(\boldsymbol{x}, \boldsymbol{x}_i) = exp(-\gamma\|\boldsymbol{x} - \boldsymbol{x}_i\|^2)$$

$$\gamma \to \infty, K\left(x_i, x_j\right) = 0, K(x_i, x_i) = 1$$

$$f(\boldsymbol{x}) = sign\left(\sum_{i\in S} \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i) + b\right)$$



Conclusion

- Support Vectors
- Lagrangian Dual
- KKT conditions
- Sequential Minimal Optimization
- Coordinate Ascent
- Two key points: maximize the margin and the kernel trick.
- Linear kernel, Polynomial kernel, RBF (Gaussian) kernel.
- VC dimensions and Structural Risk Minimization (SRM)

Many SVM implementations are available on the web for you to try on your data set!

Homework

1. Please derive the dual problem for the following objective function.

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

$$s.t.\ y_i(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_i + b) \geq 1 - \xi_i, i = 1,2,\dots,n$$

$$\xi_i \geq 0, i = 1,2,\dots,n$$

The answer should be:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j \boldsymbol{x}_i{}^{\boldsymbol{T}}\boldsymbol{x}_j$$

$$s.t.\ 0 \leq \alpha_i \leq C\boldsymbol{,}\ i = 1,\dots,n$$

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

Homework

2. Please prove that for the following objective function,

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

$$s.t.\ y_i(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_i + b) \geq 1 - \xi_i, i = 1,2, \dots, n$$

$$\xi_i \geq 0, i = 1,2, \dots, n$$

We have

$$\alpha_i = 0 \Rightarrow y_i(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_i + b)\ \geq 1$$

$$\alpha_i = C \Rightarrow y_i\left(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_i + b\right) \leq 1$$

$$0 < \alpha_i < C \Rightarrow y_i\left(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_i + b\right)\ = 1$$