Machine Learning

Yuh-Jye Lee

Lab of Data Science and Machine Intelligence
Dept. of Applied Math. at NCTU

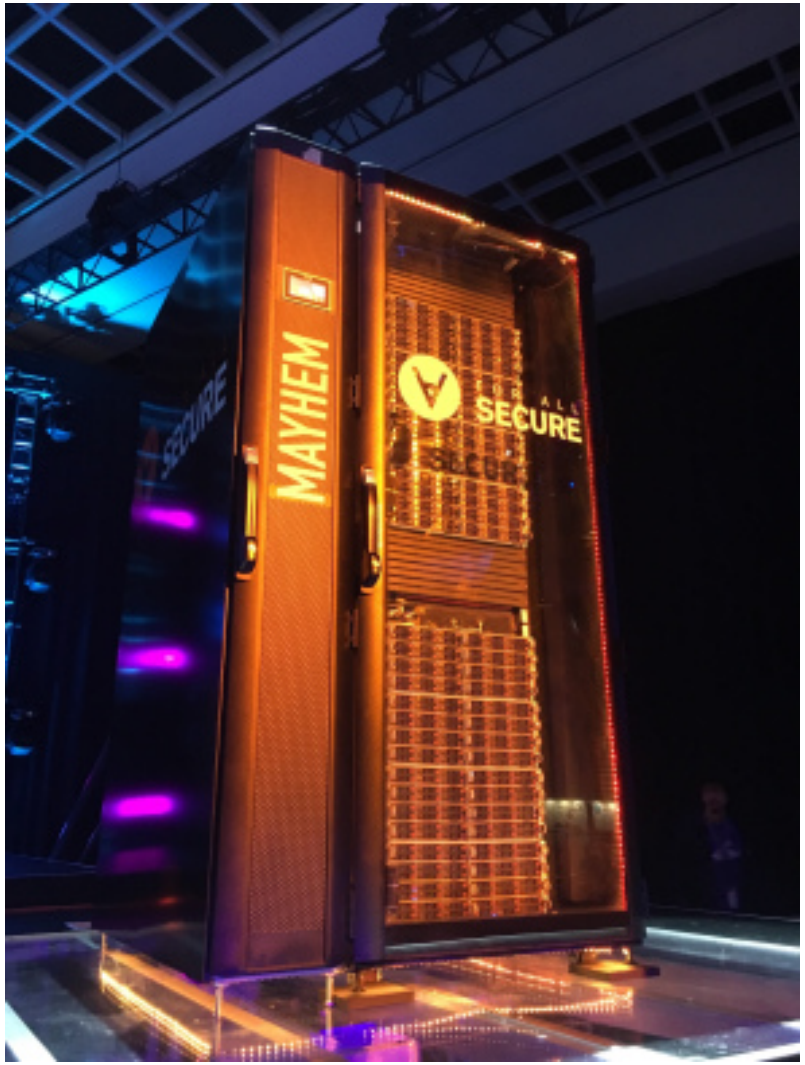
February 12, 2017

The Plan of My Lecture

- Focus on *Supervised Learning* mainly (30*minutes*)
  - Many examples
  - Basic Concept of Learning Theorey
- Will give you three basic algorithms (80*minutes*)
  - k-Nearest Neighbor
  - Naive Bayes Classifier
  - Online Perceptron Algorithm
- Brief Introduction to Optimization (90*minutes*)
- Support Vector Machines (90*minutes*)
- Evaluation and Closed Remarks (70*minutes*)

AlphaGo and Master

Mayhem Wins DARPA Cyber Grand Challenge



Supervised Learning Problems

- Assumption: training instances are drawn from an unknown but fixed probability distribution $P(\mathbf{x}, y)$ independently.
- Our learning task:
  - Given a training set $S = \{(\mathbf{x}^1, y_1), (\mathbf{x}^2, y_2), \ldots, (\mathbf{x}^\ell, y_\ell)\}$
  - We would like to construct a *rule*, $f(\mathbf{x})$ that can correctly predict the label y given unseen $\mathbf{x}$
  - If $f(\mathbf{x}) \neq y$ then we get some loss or penalty
  - For example: $\ell(f(\mathbf{x}), y) = \frac{1}{2}|f(\mathbf{x}) - y|$
- Learning examples: classification, regression and sequence labeling
  - If y is drawn from a finite set it will be a classification problem. The simplest case: $y \in \{-1, \ +1\}$ called *binary classification* problem
  - If y is a real number it becomes a regression problem
  - More general case, y can be a *vector* and each element is drawn from a finite set. This is the sequence labeling problem

Binary Classification Problem (A Fundamental Problem in Data Mining)

- Find a decision function (classifier) to discriminate two categories data set.
- Supervised learning in Machine Learning
  - Decision Tree, *Deep* Neural Network, k-NN and Support Vector Machines, etc.
- Discrimination Analysis in Statistics
  - Fisher Linear Discriminator
- Successful applications:
  - Cyber Security, Marketing, Bioinformatics, Fraud detection

Bankruptcy Prediction: Solvent vs. Bankrupt
A Binary Classification Problem

Deutsche Bank Dataset

- 40 financial indicators, (**x** part), from middle-market capitalization 422 firms in Benelux.
- 74 firms went *bankrupt* and 348 were *solvent*. (y part)
- The variables to be used in the model as explanatory inputs are 40 financial indicators such as: liquidity, profitability and solvency measurements.
- Machine Learning will identify the most important indicators

W. Härdle, Y.-J. Lee, D. Schäfer, Dorothea and Y.-R. Yeh, "Variable selection and oversampling in the use of smooth support vector machines for predicting the default risk of companies", Journal of Forecasting, vol 28, (6), p. 512 - 534, 2009

Binary Classification Problem

Given a training dataset

$$S = \{(\mathbf{x}^i, y_i) | \mathbf{x}^i \in \mathbb{R}^n, y_i \in \{-1, 1\}, i = 1, \ldots, \ell\}$$

$$\mathbf{x}^i \in A_+ \Leftrightarrow y_i = 1 \ \& \ \mathbf{x}^i \in A_- \Leftrightarrow y_i = -1$$

Main Goal:

$$\boxed{\text{Predict the unseen class label for new data}}$$

- Estimate a *posteriori probability* of class label

$$Pr(y = 1|\mathbf{x}) > Pr(y = -1|\mathbf{x}) \ \Rightarrow \mathbf{x} \in A_+$$

- Find a function $f : \mathbb{R}^n \to \mathbb{R}$ by learning from data

$$f(\mathbf{x}) \geq 0 \Rightarrow \mathbf{x} \in A_+ \text{ and } f(\mathbf{x}) < 0 \Rightarrow \mathbf{x} \in A_-$$

The simplest function is linear: $f(\mathbf{x}) = w^\top \mathbf{x} + b$

Goal of Learning Algorithms

- The early learning algorithms were designed to find such an accurate fit to the data.
  - At that time, the training set size is relative *small*
- A classifier is said to be *consistent* if it performed the correct classification of the training data.
  - Please note that it is **NOT** our learning purpose
- The ability of a classifier to correctly classify data *not in the training set* is known as its *generalization.*
- Bible code? 1994 Taipei Mayor election?
- Predict the real future *NOT fitting the data in your hand or predict the desired results*.

Three Fundamental Algorithms

- Naïve Bayes Classifier
  - Based on Bayes' Rule
- k-Nearest Neighbors Algorithm
  - Distance and Instances based algorithm
  - Lazy learning
- Online Perceptron Algorithm
  - Mistakes driven algorithm
  - The smallest unit of *Deep Neural Networks*

Conditional Probability

Definition

The conditional probability of an event A, given that an event B has occurred, is equal to

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

- Example
  Suppose that a fair die is tossed once. Find the probability of a 1 (event A), given an odd number was obtained (event B).

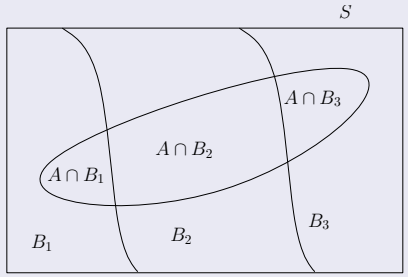
$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{1/6}{1/2} = \frac{1}{3}$$

- Restrict the sample space on the event B

Partition Theorem

Assume that $\{B_1, B_2, \ldots, B_k\}$ is a partition of S such that $P(B_i) > 0$, for $i = 1, 2, \ldots, k$. Then

$$P(A) = \sum_{i=1}^{k} P(A|B_i)P(B_i).$$



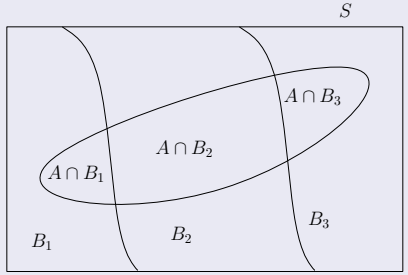
- Note that $\{B_1, B_2, \ldots, B_k\}$ is a partition of S if
  1. $S = B_1 \cup B_2 \cup \ldots \cup B_k$
  2. $B_i \cap B_j = \emptyset$ for $i \neq j$

Bayes' Rule

Bayes' Rule

Assume that $\{B_1, B_2, \ldots, B_k\}$ is a partition of S such that $P(B_i) > 0$, for $i = 1, 2, \ldots, k$. Then

$$P(B_j|A) = \frac{P(A|B_j)P(B_j)}{\sum_{i=1}^{k} P(A|B_i)P(B_i)}.$$



Naïve Bayes for Classification
Also Good for Multi-class Classification

- Estimate a *posteriori probability* of class label
- Let each *attribute* (variable) be a *random variable*. What is the probibility of

$$Pr(y = 1|\mathbf{x}) = Pr(y = 1|\mathbf{X}_1 = x_1, \mathbf{X}_2 = x_2, \ldots, \mathbf{X}_n = x_n)$$

- Naïve Bayes TWO not reasonable assumptions:
  - The importance of each attribute is *equal*
  - All attributes are *conditional probability independent* !

$$Pr(y = 1|\mathbf{x}) = \frac{1}{Pr(\mathbf{X} = \mathbf{x})} \prod_{i=1}^{n} Pr(y = 1|\mathbf{X}_i = x_i)$$

The Weather Data Example

Ian H. Witten & Eibe Frank, Data Mining

Outlook	Temperature	Humidity	Windy	Play(Label)
Sunny	Hot	High	False	-1
Sunny	Hot	High	True	-1
Overcast	Hot	High	False	+1
Rainy	Mild	High	False	+1
Rainy	Cool	Normal	False	+1
Rainy	Cool	Normal	True	-1
Overcast	Cool	Normal	True	+1
Sunny	Mild	High	False	-1
Sunny	Cool	Normal	False	+1
Rainy	Mild	Normal	False	+1
Sunny	Mild	Normal	True	+1
Overcast	Mild	High	True	+1
Overcast	Hot	Normal	False	+1
Rainy	Mild	High	True	-1

Probabilities for Weather Data Using Maximum Likelihood Estimation

Outlook			Temp.			Humidity			Windy			Play	
Play	Yes	No		Yes	No		Yes	No		Yes	No	Yes	No
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	T	3/9	3/5	9/14	5/14
Overcast	4/9	**0/5**	Mild	4/9	3/5	Normal	6/9	1/5	F	6/9	2/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

Likelihood of the two classes:

$$Pr(y = 1|sunny,\ cool,\ high,\ T) \propto \frac{2}{9} \cdot \frac{3}{9} \cdot \frac{3}{9} \cdot \frac{3}{9} \cdot \frac{9}{14}$$

$$Pr(y = -1|sunny,\ cool,\ high,\ T) \propto \frac{3}{5} \cdot \frac{1}{5} \cdot \frac{4}{5} \cdot \frac{3}{5} \cdot \frac{5}{14}$$

Zero-frequency Problem

- What if an attribute value does *NOT* occur with a class value?
  - The *posterior probability* will all be *zero*! No matter how likely the other attribute values are!
  - Laplace estimator will fix "zero-frequency", $\frac{k+\lambda}{n+a\lambda}$
- **Question:** Roll a dice 8 times. The outcomes are as: 2, 5, 6, 2, 1, 5, 3, 6. What is the probability for showing 4?

$$Pr(X=4) = \frac{0+\lambda}{8+6\lambda}, \quad Pr(X=5) = \frac{2+\lambda}{8+6\lambda}$$

Instance-based Learning: k-nearest neighbor algorithm

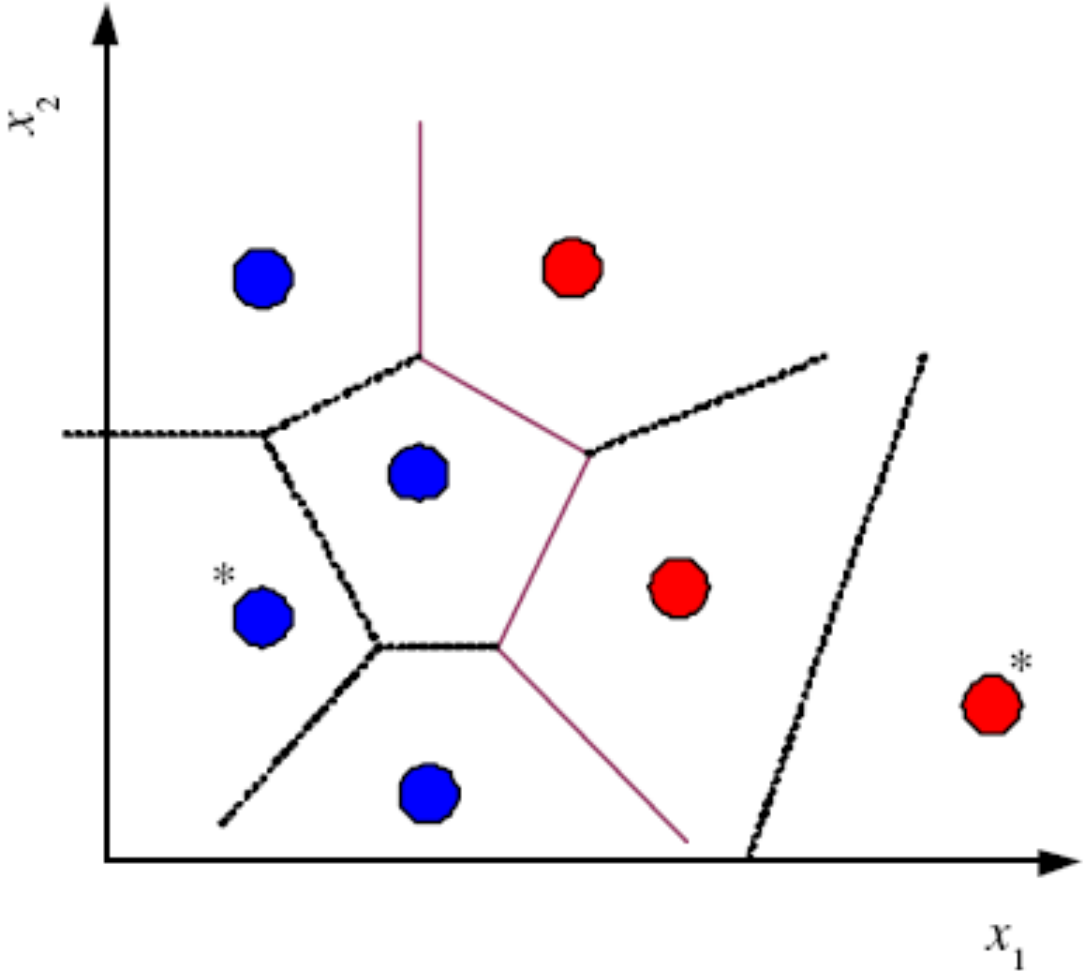
- Fundamental philosophy: Two instances that are *close to each other* or *similar to each other* they should share with the same label
- Also known as *memory-based learning* since what they do is store the training instances in a lookup table and *interpolate* from these.
- It requires memory of $\mathcal{O}(N)$
- Given an input similar ones should be found and finding them requires computation of $\mathcal{O}(N)$
- Such methods are also called *lazy learning* algorithms. Because they do NOT compute a model when they are given a training set but postpone the computation of the model until they are given a new test instance (query point)

k-Nearest Neighbors Classifier

- Given a query point $\mathbf{x}^o$, we find the k training points $\mathbf{x}^{(i)},\ i = 1, 2, \ldots, k$ *closest* in *distance* to x^o
- Then classify using *majority vote* among these k neighbors.
- Choose k as an odd number will avoid the tie. Ties are broken at random
- If all attributes (features) are real-valued, we can use Euclidean distance. That is $d(\mathbf{x}, \mathbf{x}^o) = \|\mathbf{x} - \mathbf{x}^o\|_2$
- If the attribute values are *discrete*, we can use *Hamming distance*, which counts the number of *nonmatching* attributes

$$d(\mathbf{x}, \mathbf{x}^o) = \sum_{j=1}^{n} \mathbf{1}(\mathbf{x}_j \neq \mathbf{x}_j^o)$$

1-Nearest Neighbor Decision Boundary (Voronoi)



Distance Measure

- Using different distance measurements will give very different results in k-NN algorithm.
- Be careful when you compute the distance
- We might need to *normalize* the scale between different attributes. For example, yearly income vs. daily spend
- Typically we first standardize each of the attributes to have mean zero and variance 1

$$\hat{\mathbf{x}}_j = \frac{\mathbf{x}_j - \mu_j}{\sigma_j}$$

Learning Distance Measure

- Finding a distance function $d(\mathbf{x}^i, \mathbf{x}^j)$ such that if $\mathbf{x}^i$ and $\mathbf{x}^j$ are belong to the *class* the distance is *small* and if they are belong to the *different classes* the distance is large.
- Euclidean distance: $\|\mathbf{x}^i - \mathbf{x}^j\|_2^2 = (\mathbf{x}^i - \mathbf{x}^j)^\top (\mathbf{x}^i - \mathbf{x}^j)$
- Mahalanobis distance: $d(\mathbf{x}^i, \mathbf{x}^j) = (\mathbf{x}^i - \mathbf{x}^j)^\top M (\mathbf{x}^i - \mathbf{x}^j)$ where M is a positive semi-definited matrix.

$$(\mathbf{x}^i - \mathbf{x}^j)^\top M (\mathbf{x}^i - \mathbf{x}^j) = (\mathbf{x}^i - \mathbf{x}^j)^\top L^\top L (\mathbf{x}^i - \mathbf{x}^j)$$

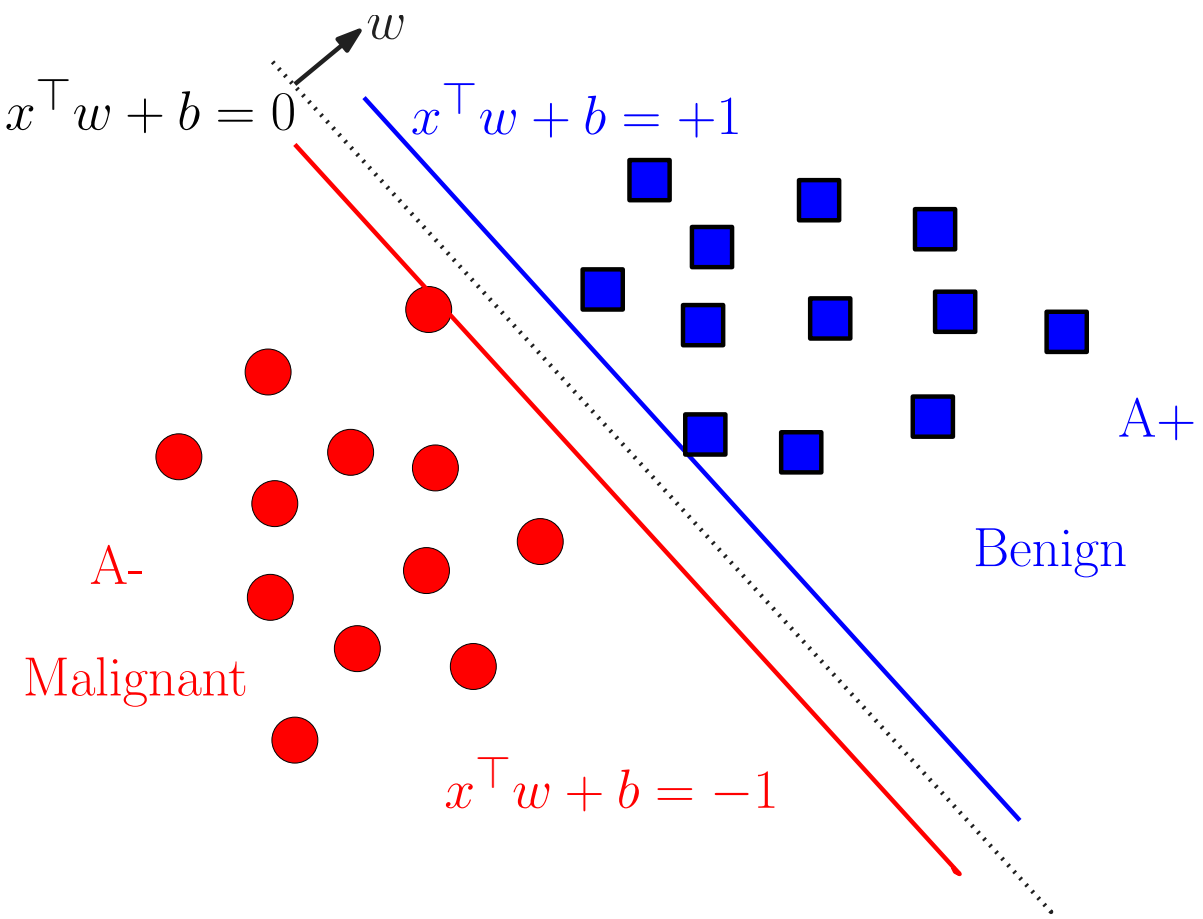
$$= (L\mathbf{x}^i - L\mathbf{x}^j)^\top (L\mathbf{x}^i - L\mathbf{x}^j)$$

- The matrix L can be with the size $k \times n$ and $k << n$

Linear Learning Machines

- The simplest function is linear: $f(\mathbf{x}) = w^\top \mathbf{x} + b$
- Finding this simplest function via an on-line and mistake-driven procedure
- Update the weight vector and bias when there is a misclassified point

Binary Classification Problem
Linearly Separable Case



Online Learning

Definition of online learning

Given a set of new training data,

- Online learner can update its model without reading old data while improving its performance.
- In contrast, off-line learner must combine old and new data and start the learning all over again, otherwise the performance will suffer.

- Online is considered as a solution of large learning tasks
- Usually require several passes (or *epochs*) through the training instances
- Need to keep all instances unless we only run the algorithm one single pass

Perceptron Algorithm (Primal Form) Rosenblatt, 1956

- Given a training dataset S, and initial weight vector $w^0 = \mathbf{0}$ and the bias $b_0 = 0$

  Repeat:

  for $i = 1$ *to* ℓ

  if $y_i(\langle w^k \cdot \mathbf{x}^i \rangle + b_k) \leq 0$ *then*

  $w^{k+1} \leftarrow w^k + \eta y_i \mathbf{x}^i$

  $b_{k+1} \leftarrow b_k + \eta y_i R^2$

  $$\boxed{R = \max_{1 \leq i \leq \ell} \|\mathbf{x}^i\|}$$

  $k \leftarrow k + 1$

  end if

  Until no mistakes made within the for loop

  Return: $k, (w^k, b_k)$.

- What is k ?

$y_i(\langle w^{k+1} \cdot \mathbf{x}^i \rangle + b_{k+1}) > y_i(\langle w^k \cdot \mathbf{x}^i \rangle) + b_k$?
$w^{k+1} \longleftarrow w^k + \eta y_i \mathbf{x}^i$ and $b_{k+1} \longleftarrow b_k + \eta y_i R^2$

$$
\begin{aligned}
y_i(\langle w^{k+1} \cdot \mathbf{x}^i \rangle + b_{k+1}) &= y_i(\langle (w^k + \eta y_i \mathbf{x}^i) \cdot \mathbf{x}^i \rangle + b_k + \eta y_i R^2) \\
&= y_i(\langle w^k \cdot \mathbf{x}^i \rangle + b_k) + y_i(\eta y_i(\langle \mathbf{x}^i \cdot \mathbf{x}^i \rangle + R^2)) \\
&= y_i(\langle w^k \cdot \mathbf{x}^i \rangle + b_k) + \eta(\langle \mathbf{x}^i \cdot \mathbf{x}^i \rangle + R^2)
\end{aligned}
$$

$$\boxed{R = \max_{1 \le i \le \ell} \|\mathbf{x}^i\|}$$

Perceptron Algorithm Stop in Finite Steps

Theorem(Novikoff)
Let S be a non-trivial training set, and let

$$R = \max_{1 \leq i \leq \ell} \|\mathbf{x}^i\|$$

Suppose that there exists a vector w_{opt} *such that* $\|w_{opt}\| = 1$ and

$$y_i(\langle w_{opt} \cdot \mathbf{x}^i \rangle + b_{opt}) \geq \gamma \text{ for } 1 \leq i \leq \ell.$$

Then the number of mistakes made by the on-line perceptron algorithm on S is almost $(\frac{2R}{\gamma})^2$.

Perceptron Algorithm (Dual Form)

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

Given a linearly separable training set S and $\alpha = 0$, $\alpha \in \mathbb{R}^\ell$, $b = 0$, $R = \max_{1 \le i \le \ell} \|\mathbf{x}^i\|$.

Repeat: *for* $i = 1$ *to* ℓ

if $y_i(\sum_{j=1}^{\ell} \alpha_j y_j \langle \mathbf{x}^j \cdot \mathbf{x}^i \rangle + b) \le 0$ *then*

$\alpha_i \leftarrow \alpha_i + 1$; $b \leftarrow b + y_i R^2$

end if

end for

Until no mistakes made within the for loop return: (α, b)

What We Got in the Dual Form of Perceptron Algorithm?

- The number of updates equals: $\sum_{i=1}^{\ell} \alpha_i = \|\alpha\|_1 \leq (\frac{2R}{\gamma})^2$
- $\alpha_i > 0$ implies that the training point $(\mathbf{x}^i, y_i)$ has been misclassified in the training process at least once.
- $\alpha_i = 0$ implies that removing the training point $(\mathbf{x}^i, y_i)$ will not affect the final results.
- The training data only appear in the algorithm through the entries of the Gram matrix, $G \in \mathbb{R}^{\ell \times \ell}$ which is defined below:

$$G_{ij} = \langle \mathbf{x}^i, \mathbf{x}^j \rangle$$

The **key idea** of *kernel trick* in SVMs and all *kernel methods*

Outline

You Have Learned (Unconstrained) Optimization in Your High School

Let $f(x) = ax^2 + bx + c$, $a \neq 0$, $x^* = -\frac{b}{2a}$

Case 1 : $f''(x^*) = 2a > 0 \Rightarrow x^* \in \arg\min_{x \in \mathbb{R}} f(x)$

Case 2 : $f''(x^*) = 2a < 0 \Rightarrow x^* \in \arg\max_{x \in \mathbb{R}} f(x)$

For minimization problem (Case I),

- $f'(x^*) = 0$ is called the first order optimality condition.
- $f''(x^*) > 0$ is the second order optimality condition.

Gradient and Hessian

- Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function. The gradient of function f at a point $x \in \mathbb{R}^n$ is defined as

$$\nabla f(x) = [\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \ldots, \frac{\partial f(x)}{\partial x_n}] \in \mathbb{R}^n$$

- If $f : \mathbb{R}^n \to \mathbb{R}$ is a twice differentiable function. The Hessian matrix of f at a point $x \in \mathbb{R}^n$ is defined as

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \in \mathbb{R}^{n \times n}$$

Example of Gradient and Hessian

$$
\begin{aligned}
f(x) &= x_1^2 + x_2^2 - 2x_1 + 4x_2 \\
&= \frac{1}{2}\begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -2 & 4 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}
\end{aligned}
$$

$$
\nabla f(x) = \begin{bmatrix} 2x_1 - 2 & 2x_2 + 4 \end{bmatrix}, \nabla^2 f(x) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}
$$

By letting $\nabla f(x) = 0$, we have $x^* = \begin{bmatrix} 1 \\ -2 \end{bmatrix} \in \arg\min_{x \in \mathbb{R}^2} f(x)$

Quadratic Functions (Standard Form)
$f(x) = \frac{1}{2}x^\top Hx + p^\top x$

Let $f : \mathbb{R}^n \to \mathbb{R}$ and $f(x) = \frac{1}{2}x^\top Hx + p^\top x$
where $H \in \mathbb{R}^{n\times n}$ is a symmetric matrix and $p \in \mathbb{R}^n$
then

$$\nabla f(x) = Hx + p$$

$$\nabla^2 f(x) = H \text{ (Hessian)}$$

Note: If H is positive definite, then $x^* = -H^{-1}p$ is the unique solution of $\min f(x)$.

Least-squares Problem

$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$

$$
\begin{aligned}
f(x) &= (Ax - b)^\top (Ax - b) \\
&= x^\top A^\top A x - 2b^\top A x + b^\top b \\
\nabla f(x) &= 2A^\top A x - 2A^\top b \\
\nabla^2 f(x) &= 2A^\top A \\
x^* &= (A^\top A)^{-1} A^\top b \in \arg\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2
\end{aligned}
$$

If $A^\top A$ is nonsingular matrix $\Rightarrow$ P.D.

Note : x^* is an analytical solution.

How to Solve an Unconstrained MP

- Get an initial point and iteratively decrease the obj. function value.
- Stop once the stopping criteria is satisfied.
- Steep decent might not be a good choice.
- Newtons method is highly recommended.
  - Local and quadratic convergent algorithm.
  - Need to choose a good step size to guarantee global convergence.

The First Order Taylor Expansion

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function

$$f(x+d) = f(x) + \nabla f(x)^\top d + \alpha(x,d)\|d\|,$$

where

$$\lim_{d \to 0} \alpha(x,d) = 0$$

If $\nabla f(x)^\top d < 0$ and d is small enough then $f(x+d) < f(x)$.

We call d is a descent direction.

Steep Descent with Exact Line Search

Start with any $x^0 \in \mathbb{R}^n$. Having x^i, stop if $\nabla f(x^i) = 0$.
Else compute x^{i+1} as follows:

1. Steep descent direction: $d^i = -\nabla f(x^i)$
2. Exact line search: Choose a stepsize such that

$$\frac{df(x^i + \lambda d^i)}{d\lambda} = f'(x^i + \lambda d^i) = 0$$

3. Updating: $x^{i+1} = x^i + \lambda d^i$

MATLAB Code for Steep Descent with Exact Line Search (Quadratic Function Only)

function $[x, f\_value, iter] = grdlines(Q, p, x0, esp)$
%
% min $0.5 * x^\top Qx + p^\top x$
% *Solving unconstrained minimization via*
% *steep descent with exact line search*
%

```
flag = 1;
iter = 0;
while flag > esp
    grad = Qx0+p;
    temp1 = grad'*grad;
    if temp1 < 10^-12
        flag = esp;
    else
        stepsize = temp1/(grad'*Q*grad);
        x1 = x0 - stepsize*grad;
        flag = norm(x1-x0);
        x0 = x1;
    end;
        iter = iter + 1;
end;
x = x0;
f_value = 0.5*x'*Q*x+p'*x;
```

The Key Idea of Newton's Method

Let $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a twice differentiable function

$$f(x+d) = f(x) + \nabla f(x)^\top d + \frac{1}{2} d^\top \nabla^2 f(x) d + \beta(x,d) \parallel d \parallel$$

where $\lim_{d \to 0} \beta(x,d) = 0$

At i^{th} iteration, use a quadratic function to approximate

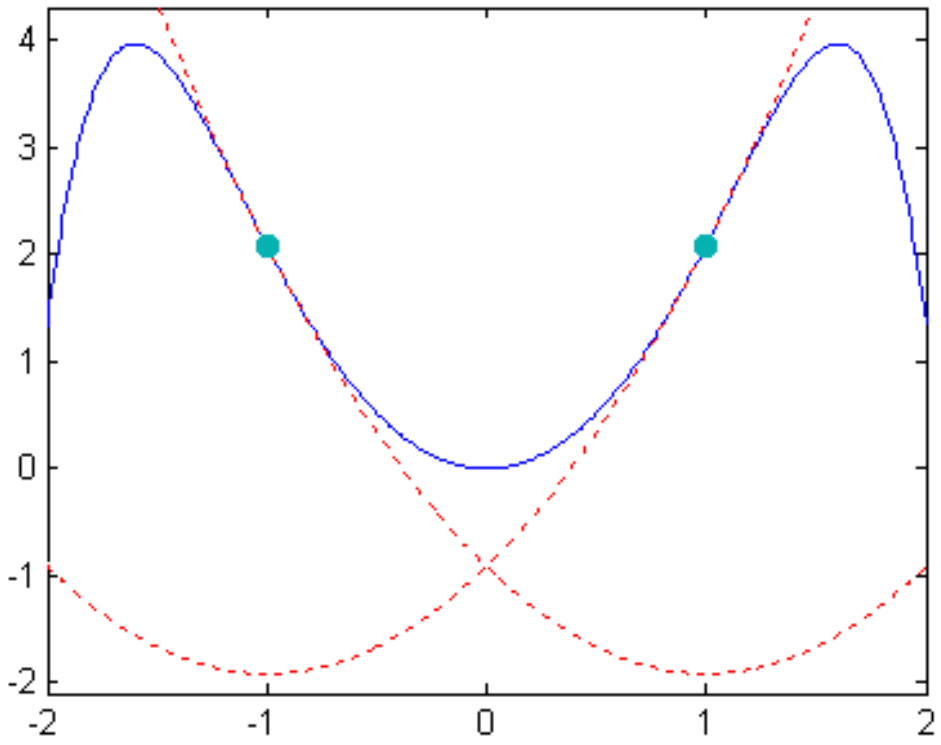
$$f(x) \approx f(x^i) + \nabla f(x^i)(x - x^i) + \frac{1}{2}(x - x^i)^\top \nabla^2 f(x^i)(x - x^i)$$

$x^{i+1} = \arg\min \tilde{f}(x)$

Newton's Method

Start with $x^0 \in \mathbb{R}^n$. Having x^i ,stop if $\nabla f(x^i) = 0$
Else compute x^{i+1} as follows:

1. Newton direction: $\nabla^2 f(x^i) d^i = -\nabla f(x^i)$
   Have to solve a system of linear equations here!
2. Updating: $x^{i+1} = x^i + d^i$
   - Converge only when x^0 is close to x^* enough.



$f(x) = \frac{1}{6}x^6 + \frac{1}{4}x^4 + 2x^2$
$g(x) = f(x^i) + f'(x^i)(x - x^i) + \frac{1}{2}f''(x^i)(x - x^i)^2$
It can not converge to the optimal solution.

People of ACM: David Blei, (Sept. 9, 2014)

The recipient of the 2013 ACM- Infosys Foundation Award in the Computing Sciences, he is joining Columbia University this fall as a Professor of Statistics and Computer Science, and will become a member of Columbia's Institute for Data Sciences and Engineering.

What is the most important recent innovation in machine learning?

[A]: One of the main recent innovations in ML research has been that we (the ML community) can now scale up our algorithms to massive data, and I think that this has fueled the modern renaissance of ML ideas in industry. The main idea is called *stochastic optimization*, which is an adaptation of an *old algorithm invented by statisticians in the 1950s*.

What is the most important recent innovation in machine learning?

[A]: *In short, many machine learning problems can be boiled down to trying to find parameters that maximize (or minimize) a function*. A common way to do this is "gradient ascent," iteratively following the steepest direction to climb a function to its top. This technique requires repeatedly calculating the steepest direction, and the problem is that this calculation can be expensive. *Stochastic optimization* lets us use *cheaper approximate calculations*. It has transformed modern machine learning.

Gradient Descent: Batch Learning

- For an optimization problem

$$\min f(\mathbf{w}) = \min r(\mathbf{w}) + \frac{1}{\ell}\sum_{i=1}^{\ell} \ell(\mathbf{w}; (\mathbf{x}^i, y_i))$$

- GD tries to find a direction and the learning rate decreasing the objective function value.

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \nabla f(\mathbf{w}^t)$$

where η is the learning rate, $-\nabla f(\mathbf{w}^t)$ is the steepest direction

$$\nabla f(\mathbf{w}^t) = \nabla r(\mathbf{w}^t) + \frac{1}{\ell}\sum_{i=1}^{\ell} \nabla \ell(\mathbf{w}^t; (\mathbf{x}^i, y_i))$$

- When ℓ is large, computing $\sum_{i=1}^{\ell} \nabla \ell(\mathbf{w}^t; (\mathbf{x}^i, y_i))$ may cost much time.

Stochastic Gradient Descent: Online Learning

- In GD, we compute the gradient using the entire training set.
- In *stochastic gradient descent*(SGD), we use

$$\nabla \ell(\mathbf{w}^t; (\mathbf{x}^t, y_t)) \quad \text{instead of} \quad \frac{1}{\ell} \sum_{i=1}^{\ell} \nabla \ell(\mathbf{w}^t; (\mathbf{x}^i, y_i))$$

- So the gradient of $f(\mathbf{w}^t)$

$$\nabla f(\mathbf{w}^t) = \nabla r(\mathbf{w}^t) + \nabla \ell(\mathbf{w}^t; (\mathbf{x}^t, y_t))$$

- SGD computes the gradient using only one instance.
- In experiment, SGD is significantly faster than GD when ℓ is large.

Online Perceptron Algorithm [Rosenblatt, 1956]

- The Perceptron is considered as a SGD method. The underlying optimization problem of the algorithm

$$\min_{(\mathbf{w},b)\in\mathbb{R}^{n+1}} \quad \sum_{i=1}^{\ell}(-y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b))_+$$

- In the linearly separable case, the Perceptron alg. will be terminated in finite steps no matter what learning rate is chosen
- In the nonseparable case, how to decide the appropriate learning rate that will make the least mistake is very difficult
- Learning rate can be a nonnegative number. More general case, it can be a positive definite matrix

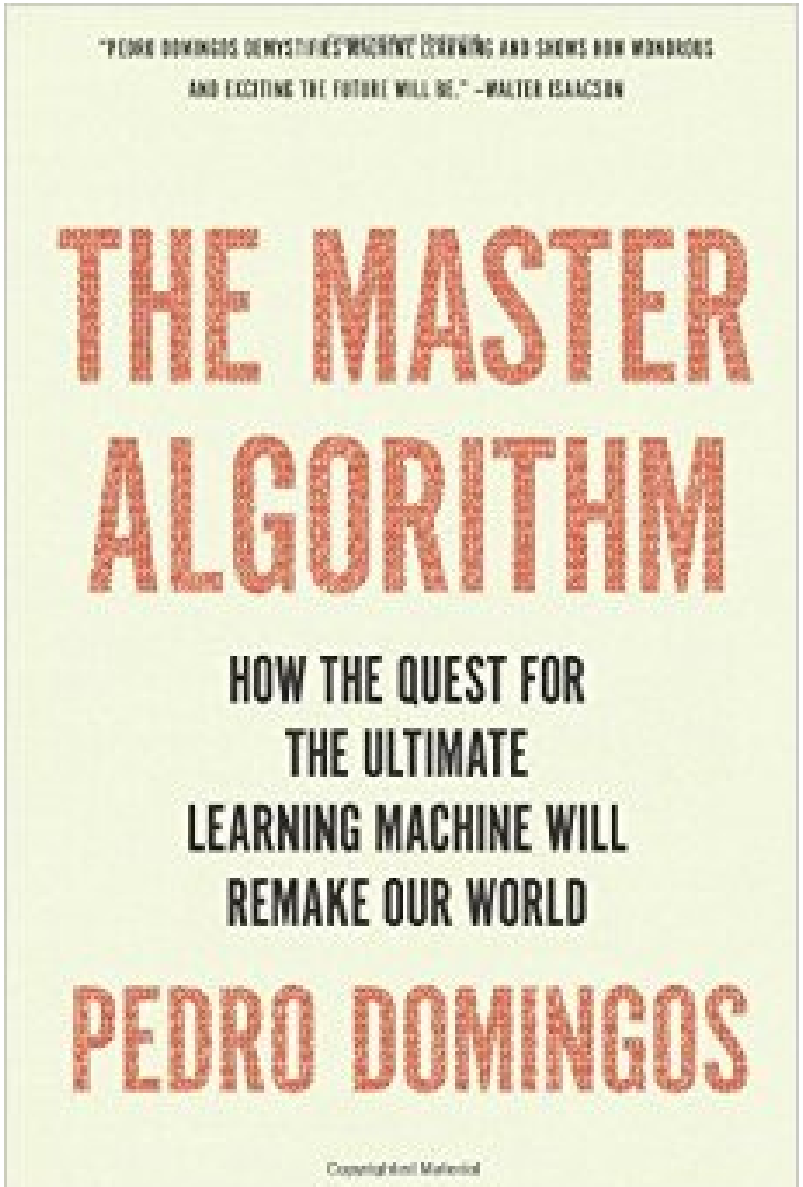
What is Machine Learning?

Representation + Optimization + Evaluation

Pedro Domingos, A few useful things to know about machine learning, Communications of the ACM, Vol. 55 Issue 10, 78-87, October 2012

The most important reading assignment in my Machine Learning and Data Science and Machine Intelligence Lab at NCTU

The Master Algorithm



The Master Algorithm

Expected Risk vs. Empirical Risk

- Assumption: training instances are drawn from an unknown but fixed probability distribution $P(\mathbf{x}, y)$ independently.
- Ideally, we would like to have the *optimal rule* f^* that minimizes the *Expected Risk*: $E(f) = \int \ell(f(\mathbf{x}), y) dP(\mathbf{x}, y)$ among all functions
- Unfortunately, we can not do it. $P(\mathbf{x}, y)$ is unknown and we have to restrict ourselves in a certain *hypothesis space*, $\mathcal{F}$
- How about compute $f_\ell^* \in \mathcal{F}$ that minimizes the *Empirical Risk*: $E_\ell(f) = \frac{1}{\ell} \sum_i \ell(f(\mathbf{x}^i), y_i)$
- Only minimizing the empirical risk will be in danger of *overfitting*

Approximation Optimization Approach

- Most of learning algorithms can be formulated as an optimization problem

Approximation Optimization Approach

- Most of learning algorithms can be formulated as an optimization problem
- The objective function consists of two parts: $E_\ell(f)+$ controls on VC-error bound

Approximation Optimization Approach

- Most of learning algorithms can be formulated as an optimization problem
- The objective function consists of two parts: $E_\ell(f)+$ controls on VC-error bound
- Controlling the VC-error bound will avoid the *overfitting* risk

Approximation Optimization Approach

- Most of learning algorithms can be formulated as an optimization problem
- The objective function consists of two parts: $E_\ell(f)+$ controls on VC-error bound
- Controlling the VC-error bound will avoid the *overfitting* risk
- It can be achieved via adding the *regularization* term into the objective function

Approximation Optimization Approach

- Most of learning algorithms can be formulated as an optimization problem
- The objective function consists of two parts: $E_\ell(f)+$ controls on VC-error bound
- Controlling the VC-error bound will avoid the *overfitting* risk
- It can be achieved via adding the *regularization* term into the objective function
- Note that: We have made lots of approximations when formulate a learning task as an optimization problem

Approximation Optimization Approach

- Most of learning algorithms can be formulated as an optimization problem
- The objective function consists of two parts: $E_\ell(f)+$ controls on VC-error bound
- Controlling the VC-error bound will avoid the *overfitting* risk
- It can be achieved via adding the *regularization* term into the objective function
- Note that: We have made lots of approximations when formulate a learning task as an optimization problem
  - Why bother to find the optimal solution for the problem?
  - One could stop the optimization iteration before its convergence

Constrained Optimization Problem

Problem setting: Given function f, g_i, $i = 1, ..., k$ and h_j, $j = 1, ..., m$, defined on a domain $\Omega \subseteq \mathbb{R}^n$,

$$\begin{aligned} \min_{x \in \Omega} \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad \forall i \\ & h_j(x) = 0, \quad \forall j \end{aligned}$$

where $f(x)$ is called the objective function and $g(x) \leq 0$, $h(x) = 0$ are called constrains.

Example

$$\begin{aligned} \min \quad & f(x) = 2x_1^2 + x_2^2 + 3x_3^2 \\ \text{s.t.} \quad & 2x_1 - 3x_2 + 4x_3 = 49 \end{aligned}$$

<sol>

$$L(x, \beta) = f(x) + \beta(2x_1 - 3x_2 + 4x_3 - 49),\ \beta \in \mathbb{R}$$

$$\frac{\partial}{\partial x_1} L(x, \beta) = 0 \quad \Rightarrow \quad 4x_1 + 2\beta = 0$$

$$\frac{\partial}{\partial x_2} L(x, \beta) = 0 \quad \Rightarrow \quad 2x_2 - 3\beta = 0$$

$$\frac{\partial}{\partial x_3} L(x, \beta) = 0 \quad \Rightarrow \quad 6x_3 + 4\beta = 0$$

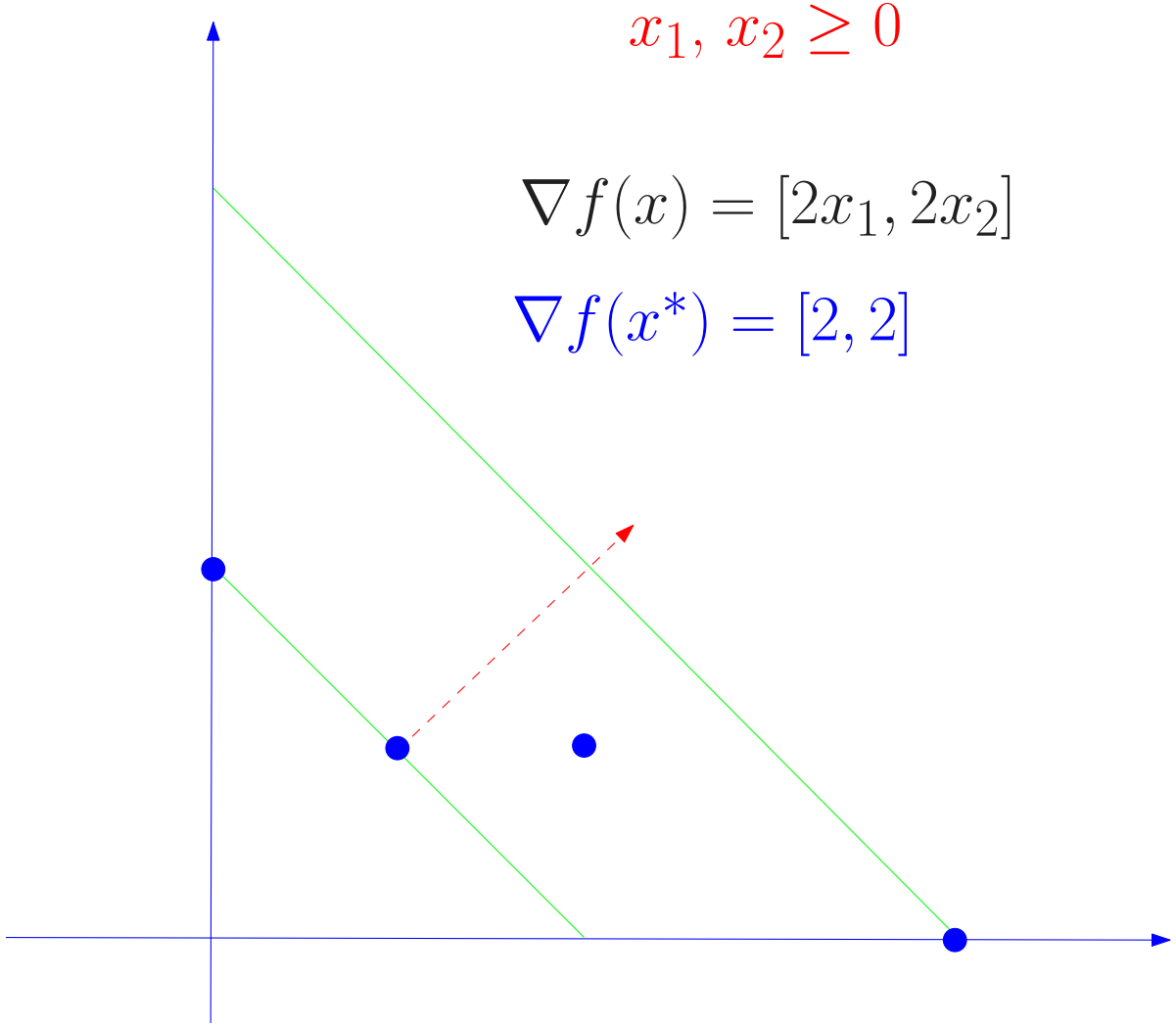
$$2x_1 - 3x_2 + 4x_3 - 49 = 0 \Rightarrow \beta = -6$$

$$\Rightarrow x_1 = 3,\ x_2 = -9,\ x_3 = 4$$

$$\min_{x \in R^2} x_1^2 + x_2^2 \quad \begin{aligned} x_1 + x_2 &\leq 4 \\ -x_1 - x_2 &\leq -2 \\ x_1,\, x_2 &\geq 0 \end{aligned}$$

$$\nabla f(x) = [2x_1, 2x_2]$$

$$\nabla f(x^*) = [2, 2]$$



Definitions and Notation

- Feasible region:

$$\mathcal{F} = \{x \in \Omega \mid g(x) \leq 0, h(x) = 0\}$$

where $g(x) = \begin{bmatrix} g_1(x) \\ \vdots \\ g_k(x) \end{bmatrix}$ and $h(x) = \begin{bmatrix} h_1(x) \\ \vdots \\ h_m(x) \end{bmatrix}$

- A solution of the optimization problem is a point $x^* \in \mathcal{F}$ such that $\nexists x \in \mathcal{F}$ for which $f(x) < f(x^*)$ and x^* is called a global minimum.

Definitions and Notation

- A point $\bar{x} \in \mathcal{F}$ is called a local minimum of the optimization problem if $\exists \varepsilon > 0$ such that

$$f(x) \geq f(\bar{x}), \qquad \forall x \in \mathcal{F} \text{ and } \|x - \bar{x}\| < \varepsilon$$

- At the solution x^*, an inequality constraint $g_i(x)$ is said to be active if $g_i(x^*) = 0$, otherwise it is called an inactive constraint.
- $g_i(x) \leq 0 \Leftrightarrow g_i(x) + \xi_i = 0$, $\xi_i \geq 0$ where ξ_i is called the slack variable

Definitions and Notation

- Remove an inactive constraint in an optimization problem will NOT affect the optimal solution
  - Very useful feature in SVM
- If $\mathcal{F} = \mathbb{R}^n$ then the problem is called unconstrained minimization problem
  - Least square problem is in this category
  - SSVM formulation is in this category
  - Difficult to find the global minimum without convexity assumption

The Most Important Concepts in Optimization(minimization)

- A point is said to be an *optimal solution* of a unconstrained minimization if there exists no decent direction $\Longrightarrow \nabla f(x^*) = 0$
- A point is said to be an optimal solution of a constrained minimization if there exists no feasible decent direction $\Longrightarrow$ KKT conditions
  - There might exist decent direction but move along this direction will leave out the feasible region

Minimum Principle

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex and differentiable function $\mathcal{F} \subseteq \mathbb{R}^n$ be the feasible region.

$$x^* \in \arg\min_{x \in \mathcal{F}} f(x) \iff \nabla f(x^*)(x - x^*) \geq 0 \quad \forall x \in \mathcal{F}$$

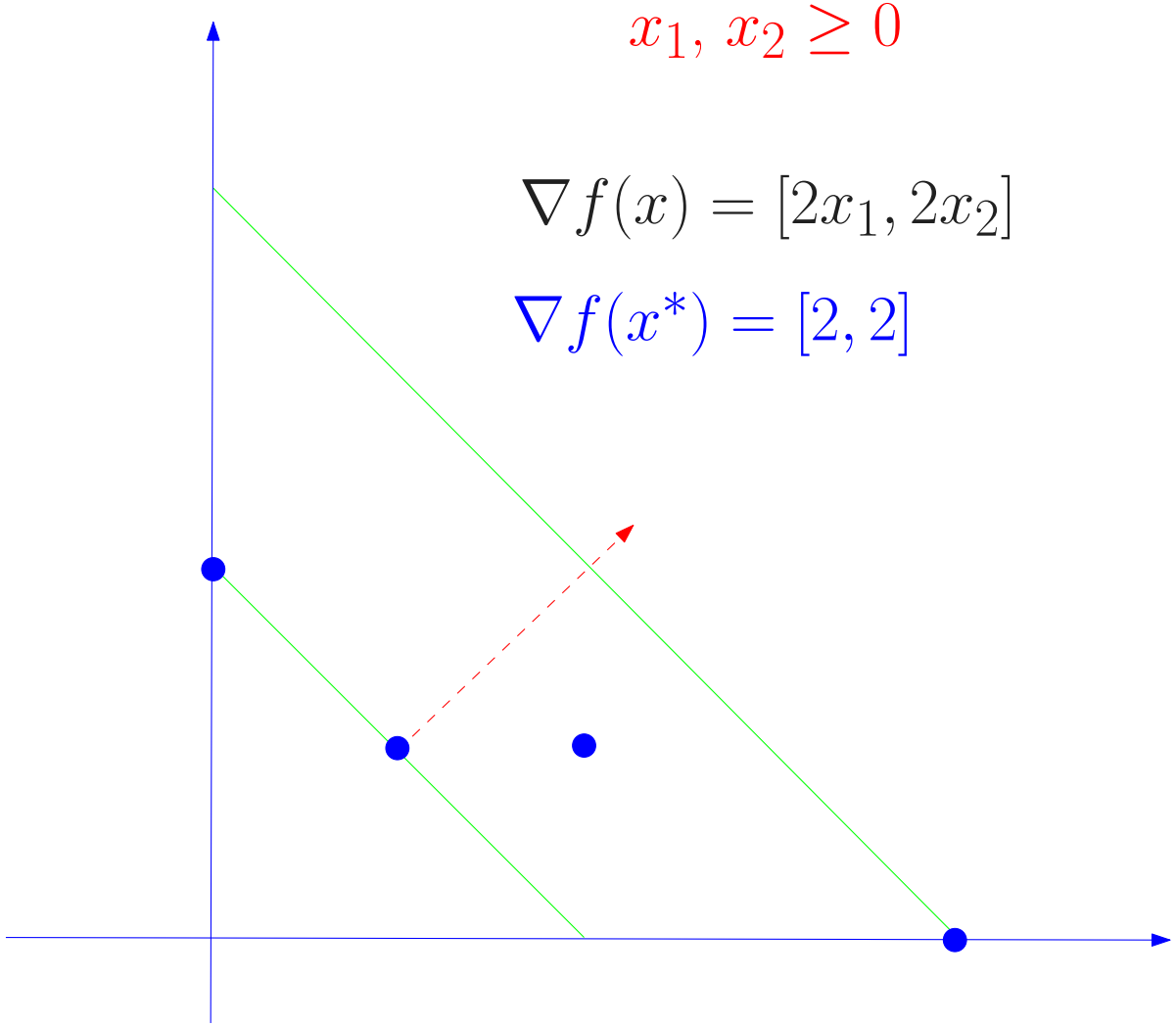
Example:

$$\min (x-1)^2 \quad \text{s.t.} \quad a \leq x \leq b$$

$$\min_{x \in R^2} x_1^2 + x_2^2 \quad \begin{aligned} x_1 + x_2 &\leq 4 \\ -x_1 - x_2 &\leq -2 \\ x_1,\, x_2 &\geq 0 \end{aligned}$$

$$\nabla f(x) = [2x_1, 2x_2]$$

$$\nabla f(x^*) = [2, 2]$$



Linear Programming Problem

- An optimization problem in which the objective function and all constraints are linear functions is called a linear programming problem

$$
\begin{array}{lll}
\text{(LP)} & \min & p^\top x \\
 & \text{s.t.} & Ax \leq b \\
 & & Cx = d \\
 & & L \leq x \leq U
\end{array}
$$

Linear Programming Solver in MATLAB

X=LINPROG(f,A,b) attempts to solve the linear programming problem:

$$\min_{x} \quad f'*x \quad \text{subject to: } A*x <= b$$

X=LINPROG(f,A,b,Aeq,beq) solves the problem above while additionally satisfying the equality constraints Aeq*x = beq.

X=LINPROG(f,A,b,Aeq,beq,LB,UB) defines a set of lower and upper bounds on the design variables, X, so that the solution is in the range LB <= X <= UB.
Use empty matrices for LB and UB if no bounds exist. Set LB(i) = -Inf if X(i) is unbounded below; set UB(i) = Inf if X(i) is unbounded above.

Linear Programming Solver in MATLAB

X=LINPROG(f,A,b,Aeq,beq,LB,UB,X0) sets the starting point to X0. This option is only available with the active-set algorithm. The default interior point algorithm will ignore any non-empty starting point.

You can type "help linprog" in MATLAB to get more information!

L_1-Approximation: $\min_{x \in \mathbb{R}^n} \|Ax - b\|_1$

$$\|z\|_1 = \sum_{i=1}^{m} |z_i|$$

$$\min_{x,s} \mathbf{1}^\top s \quad \text{s.t. } -s \leq Ax - b \leq s$$

Or

$$\min_{x,s} \sum_{i=1}^{m} s_i \quad \text{s.t. } -s_i \leq A_i x - b_i \leq s_i \quad \forall i$$

$$\min_{x,s} \begin{bmatrix} 0 & \cdots & 0 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} x \\ s \end{bmatrix}$$

$$\text{s.t. } \begin{bmatrix} A & -I \\ -A & -I \end{bmatrix}_{2m \times (n+m)} \begin{bmatrix} x \\ s \end{bmatrix} \leq \begin{bmatrix} b \\ -b \end{bmatrix}$$

Chebyshev Approximation: $\min_{x \in \mathbb{R}^n} \|Ax - b\|_\infty$

$$\|z\|_\infty = \max_{1 \le i \le m} |z_i|$$

$$\begin{aligned} \min_{x,\gamma} \quad & \gamma \\ \text{s.t.} \quad & -\mathbf{1}\gamma \le Ax - b \le \mathbf{1}\gamma \end{aligned}$$

$$\begin{aligned} \min_{x,s} \quad & \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \gamma \end{bmatrix} \\ \text{s.t.} \quad & \begin{bmatrix} A & -\mathbf{1} \\ -A & -\mathbf{1} \end{bmatrix}_{2m \times (n+1)} \begin{bmatrix} x \\ \gamma \end{bmatrix} \le \begin{bmatrix} b \\ -b \end{bmatrix} \end{aligned}$$

Quadratic Programming Problem

- If the objective function is convex quadratic while the constraints are all linear then the problem is called convex quadratic programming problem

$$
\begin{array}{lll}
\text{(QP)} & \min & \frac{1}{2}x^\top Qx + p^\top x \\
 & \text{s.t.} & Ax \leq b \\
 & & Cx = d \\
 & & L \leq x \leq U
\end{array}
$$

Quadratic Programming Solver in MATLAB

X=QUADPROG(H,f,A,b) attempts to solve the quadratic programming problem:

$$\min_{x} \quad 0.5*x'*H*x+f'*x \quad \text{subject to: } A*x <= b$$

X=QUADPROG(H,f,A,b,Aeq,beq) solves the problem above while additionally satisfying the equality constraints Aeq*x=beq.

X=QUADPROG(H,f,A,b,Aeq,beq,LB,UB) defines a set of lower and upper bounds on the design variables, X, so that the solution is in the range LB <= X <= UB.
Use empty matrices for LB and UB if no bounds exist. Set LB(i) = -Inf if X(i) is unbounded below; set UB(i) = Inf if X(i) is unbounded above.

Quadratic Programming Solver in MATLAB

X=QUADPROG(H,f,A,b,Aeq,beq,LB,UB,X0) sets the starting point to X0.

You can type "help quadprog" in MATLAB to get more information!

Standard Support Vector Machine

$$\min_{w,b,\xi_A,\xi_B} C(\mathbf{1}^\top \xi_A + \mathbf{1}^\top \xi_B) + \frac{1}{2}\|w\|_2^2$$

$$\begin{aligned}
&(Aw + \mathbf{1}b) + \xi_A \geq \mathbf{1} \\
&(Bw + \mathbf{1}b) - \xi_B \leq -\mathbf{1} \\
&\xi_A \geq 0, \xi_B \geq 0
\end{aligned}$$

Farkas' Lemma

For any matrix $A \in \mathbb{R}^{m \times n}$ and any vector $b \in \mathbb{R}^n$, either

$$Ax \leq 0, \quad b^\top x > 0 \quad \text{has a solution}$$

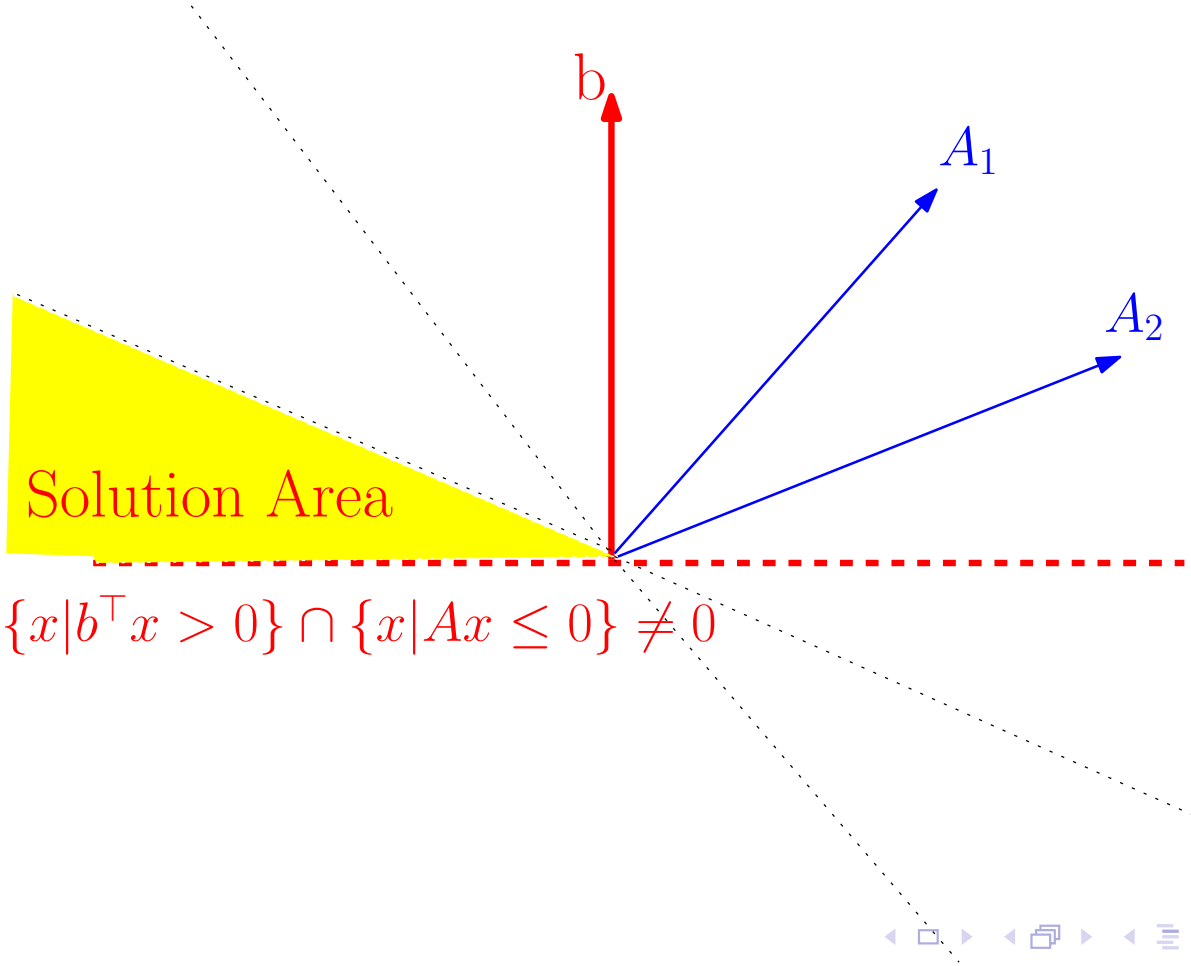
or

$$A^\top \alpha = b, \quad \alpha \geq 0 \quad \text{has a solution}$$

but never both.

Farkas' Lemma
$Ax \leq \mathbf{0}$, $b^\top x > 0$ has a solution

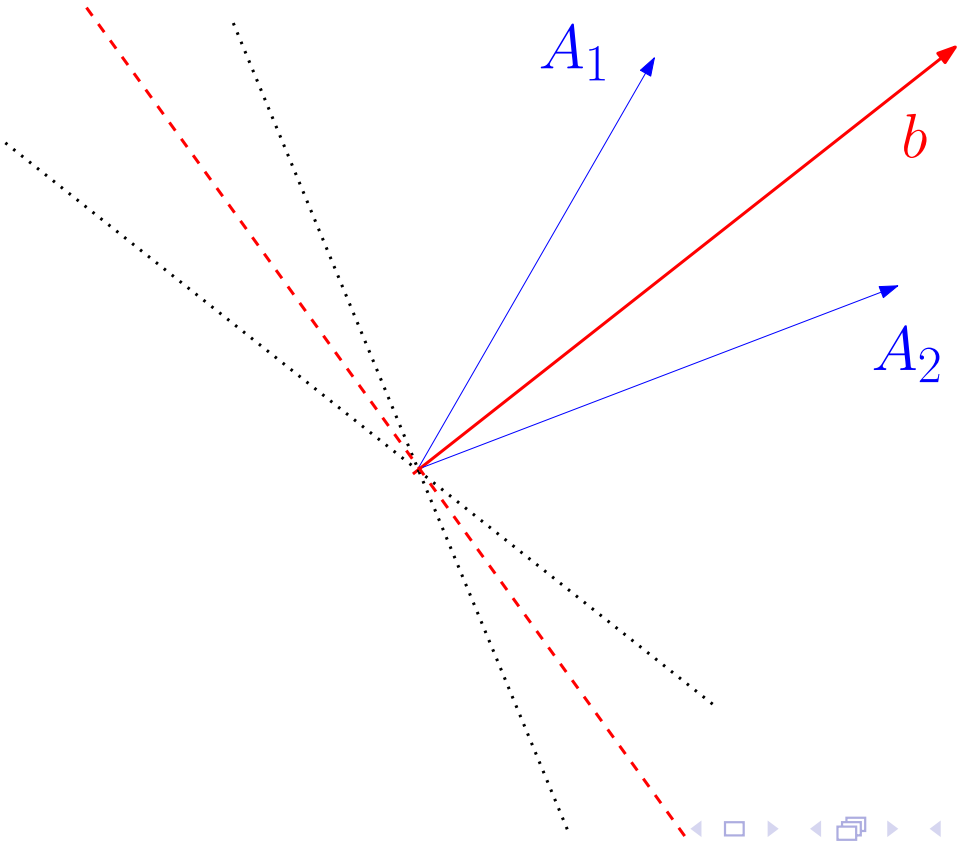
b is NOT in the cone generated by A_1 and A_2



Farkas' Lemma
$A^\top \alpha = b$, $\alpha \geq 0$ has a solution

b is in the cone generated by A_1 and A_2

$$\{x | b^\top > 0\} \cap \{x | Ax \leq 0\} = \emptyset$$



Minimization Problem *vs.* Kuhn-Tucker Stationary-point Problem

MP:

$$\begin{aligned} \min_{x \in \Omega} \quad & f(x) \\ s.t. \quad & g(x) \leq 0 \end{aligned}$$

KTSP:

$$\begin{aligned} \text{Find} \quad & \bar{x} \in \Omega, \ \bar{\alpha} \in \mathbb{R}^m \ \text{ such that} \\ & \nabla f(\bar{x}) + \bar{\alpha}^\top \nabla g(\bar{x}) = 0 \\ & \bar{\alpha}^\top g(\bar{x}) = 0 \\ & g(\bar{x}) \leq 0 \\ & \bar{\alpha} \geq 0 \end{aligned}$$

Lagrangian Function
$\mathcal{L}(x, \alpha) = f(x) + \alpha^\top g(x)$

Let $\mathcal{L}(x, \alpha) = f(x) + \alpha^\top g(x)$ and $\alpha \geq 0$

- If $f(x)$, $g(x)$ are convex the $\mathcal{L}(x, \alpha)$ is convex.
- For a fixed $\alpha \geq 0$, if $\bar{x} \in \arg\min\{\mathcal{L}(x, \alpha) | x \in \mathbb{R}^n\}$ then
$$\left.\frac{\partial \mathcal{L}(x, \alpha)}{\partial x}\right|_{x=\bar{x}} = \nabla f(\bar{x}) + \alpha^\top \nabla g(\bar{x}) = 0$$
- Above result is a sufficient condition if $\mathcal{L}(x, \alpha)$ is convex.

KTSP with Equality Constraints? (Assume $h(x) = 0$ are linear functions)

$$h(x) = 0 \;\Leftrightarrow\; h(x) \leq 0 \text{ and } -h(x) \leq 0$$

KTSP:

$$\begin{aligned}
\text{Find} \quad & \bar{x} \in \Omega, \bar{\alpha} \in \mathbb{R}^k, \bar{\beta}_+, \bar{\beta}_- \in \mathbb{R}^m \text{ such that} \\
& \nabla f(\bar{x}) + \bar{\alpha}^\top \nabla g(\bar{x}) + (\bar{\beta}_+ - \bar{\beta}_-)^\top \nabla h(\bar{x}) = 0 \\
& \bar{\alpha}^\top g(\bar{x}) = 0,\; (\bar{\beta}_+)^\top h(\bar{x}) = 0,\; (\bar{\beta}_-)^\top (-h(\bar{x})) = 0 \\
& g(\bar{x}) \leq 0,\; h(\bar{x}) = 0 \\
& \bar{\alpha} \geq 0,\; \bar{\beta}_+,\; \bar{\beta}_- \geq 0
\end{aligned}$$

KTSP with Equality Constraints

KTSP:

$$
\begin{aligned}
\text{Find} \quad & \bar{x} \in \Omega, \bar{\alpha} \in \mathbb{R}^k, \bar{\beta} \in \mathbb{R}^m \text{ such that} \\
& \nabla f(\bar{x}) + \bar{\alpha}^\top \nabla g(\bar{x}) + \bar{\beta} \nabla h(\bar{x}) = 0 \\
& \bar{\alpha}^\top g(\bar{x}) = 0, \; g(\bar{x}) \leq 0, \; h(\bar{x}) = 0 \\
& \bar{\alpha} \geq 0
\end{aligned}
$$

- Let $\bar{\beta} = \bar{\beta}_+ - \bar{\beta}_-$ and $\bar{\beta}_+, \bar{\beta}_- \geq 0$ then $\bar{\beta}$ is free variable

Generalized Lagrangian Function $\mathcal{L}(x, \alpha, \beta) = f(x) + \alpha^\top g(x) + \beta^\top h(x)$

Let $\mathcal{L}(x, \alpha, \beta) = f(x) + \alpha^\top g(x) + \beta^\top h(x)$ and $\alpha \geq 0$

- If $f(x)$, $g(x)$ are convex and $h(x)$ is linear then $\mathcal{L}(x, \alpha, \beta)$ is convex.
- For fixed $\alpha \geq 0$, if $\bar{x} \in \arg\min\{\mathcal{L}(x, \alpha, \beta) | x \in \mathbb{R}^n\}$ then

$$\frac{\partial \mathcal{L}(x, \alpha, \beta)}{\partial x}\Big|_{x=\bar{x}} = \nabla f(\bar{x}) + \alpha^\top \nabla g(\bar{x}) + \beta^\top \nabla h(\bar{x}) = 0$$

- Above result is a sufficient condition if $\mathcal{L}(x, \alpha, \beta)$ is convex.

Lagrangian Dual Problem

$$\begin{aligned} \max_{\alpha,\beta} \min_{x \in \Omega} \quad & \mathcal{L}(x, \alpha, \beta) \\ s.t. \quad & \alpha \geq 0 \end{aligned}$$

Lagrangian Dual Problem

$$\begin{aligned} \max_{\alpha,\beta} \min_{x\in\Omega} \quad & \mathcal{L}(x,\alpha,\beta) \\ s.t. \quad & \alpha \geq 0 \end{aligned}$$

$$\Updownarrow$$

$$\begin{aligned} \max_{\alpha,\beta} \quad & \theta(\alpha,\beta) \\ s.t. \quad & \alpha \geq 0 \\ \text{where} \quad & \theta(\alpha,\beta) = \inf_{x\in\Omega} \mathcal{L}(x,\alpha,\beta) \end{aligned}$$

Weak Duality Theorem

Let $\bar{x} \in \Omega$ be a feasible solution of the primal problem and (α, β) a feasible sulution of the *dual* problem. then $f(\bar{x}) \geq \theta(\alpha, \beta)$

$$\theta(\alpha, \beta) = \inf_{x \in \Omega} \mathcal{L}(x, \alpha, \beta) \leq \mathcal{L}(\tilde{x}, \alpha, \beta)$$

Corollary:

$$\sup\{\theta(\alpha, \beta) | \alpha \geq 0\} \leq \inf\{f(x) | g(x) \leq 0,\ h(x) = 0\}$$

Weak Duality Theorem

Corollary

If $f(x^*) = \theta(\alpha^*, \beta^*)$ where $\alpha^* \geq \mathbf{0}$ and $g(x^*) \leq \mathbf{0}$, $h(x^*) = \mathbf{0}$,then x^* and (α^*, β^*) solve the *primal* and *dual* problem respectively. In this case,

$$\mathbf{0} \leq \alpha \perp g(x) \leq \mathbf{0}$$

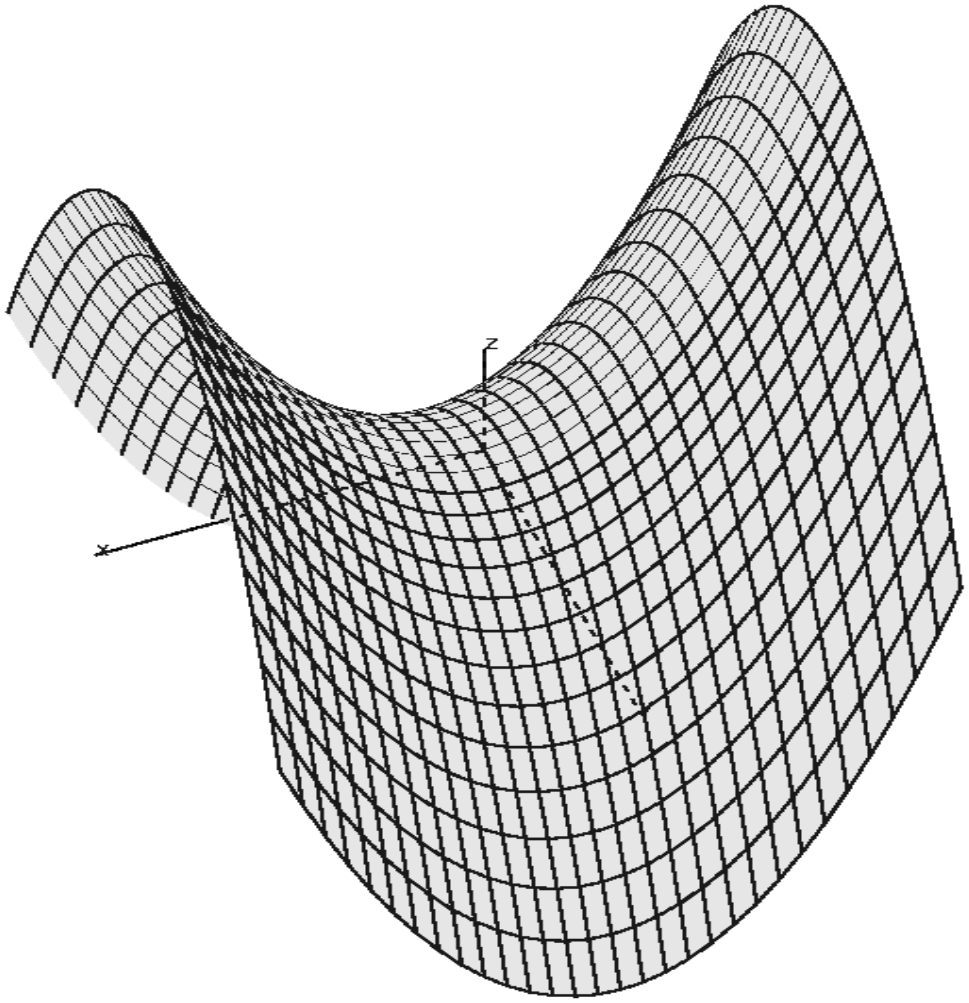
Saddle Point of Lagrangian

Let $x^* \in \Omega$,$\alpha^* \geq \mathbf{0}$, $\beta^* \in \mathbb{R}^m$ satisfying

$$\mathcal{L}(x^*, \alpha, \beta) \leq \mathcal{L}(x^*, \alpha^*, \beta^*) \leq \mathcal{L}(x, \alpha^*, \beta^*) \text{ , } \forall x \in \Omega \text{ , } \alpha \geq \mathbf{0}$$

Then (x^*, α^*, β^*) is called The saddle point of the Lagrangian function

Saddle Point of $f(x, y) = x^2 - y^2$



Dual Problem of Linear Program

$$\textit{Primal}\ \text{LP} \qquad \min_{x \in \mathbb{R}^n} \quad p^\top x \qquad \text{subject to} \quad Ax \geq b\ ,\ x \geq \mathbf{0}$$

$$\textit{Dual}\ \text{LP} \qquad \max_{\alpha \in \mathbb{R}^m} \quad b^\top \alpha \qquad \text{subject to} \quad A^\top \alpha \leq p\ ,\ \alpha \geq \mathbf{0}$$

- All duality theorems hold and work perfectly!

Lagrangian Function of Primal LP

$\mathcal{L}(x, \alpha) = p^\top x + \alpha_1^\top (b - Ax) + \alpha_2^\top (-x)$

$$\max_{\alpha_1, \alpha_2 \geq \mathbf{0}} \min_{x \in \mathbb{R}^n} \mathcal{L}(x, \alpha_1, \alpha 2)$$

$$\Updownarrow$$

$$\begin{aligned} \max_{\alpha_1, \alpha_2 \geq \mathbf{0}} \quad & p^\top x + \alpha_1^\top (b - Ax) + \alpha_2^\top (-x) \\ \text{subject to} \quad & p - A^\top \alpha_1 - \alpha_2 = \mathbf{0} \\ & (\nabla_x \mathcal{L}(x, \alpha_1, \alpha_2) = \mathbf{0}) \end{aligned}$$

Application of LP Duality
LSQ – NormalEquation Always Has a Solution

For any matrix $A \in \mathbb{R}^{mxn}$ and any vector $b \in \mathbb{R}^m$,
consider $\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2$

$$x^* \in \arg\min\{\|Ax - b\|_2^2\} \Leftrightarrow A^\top Ax^* = A^\top b$$

Claim : $A^\top Ax = A^\top b$ always has a solution.

Dual Problem of Strictly Convex Quadratic Program

Primal QP

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \frac{1}{2} x^\top Q x + p^\top x \\ \text{s.t.} \quad & Ax \leq b \end{aligned}$$

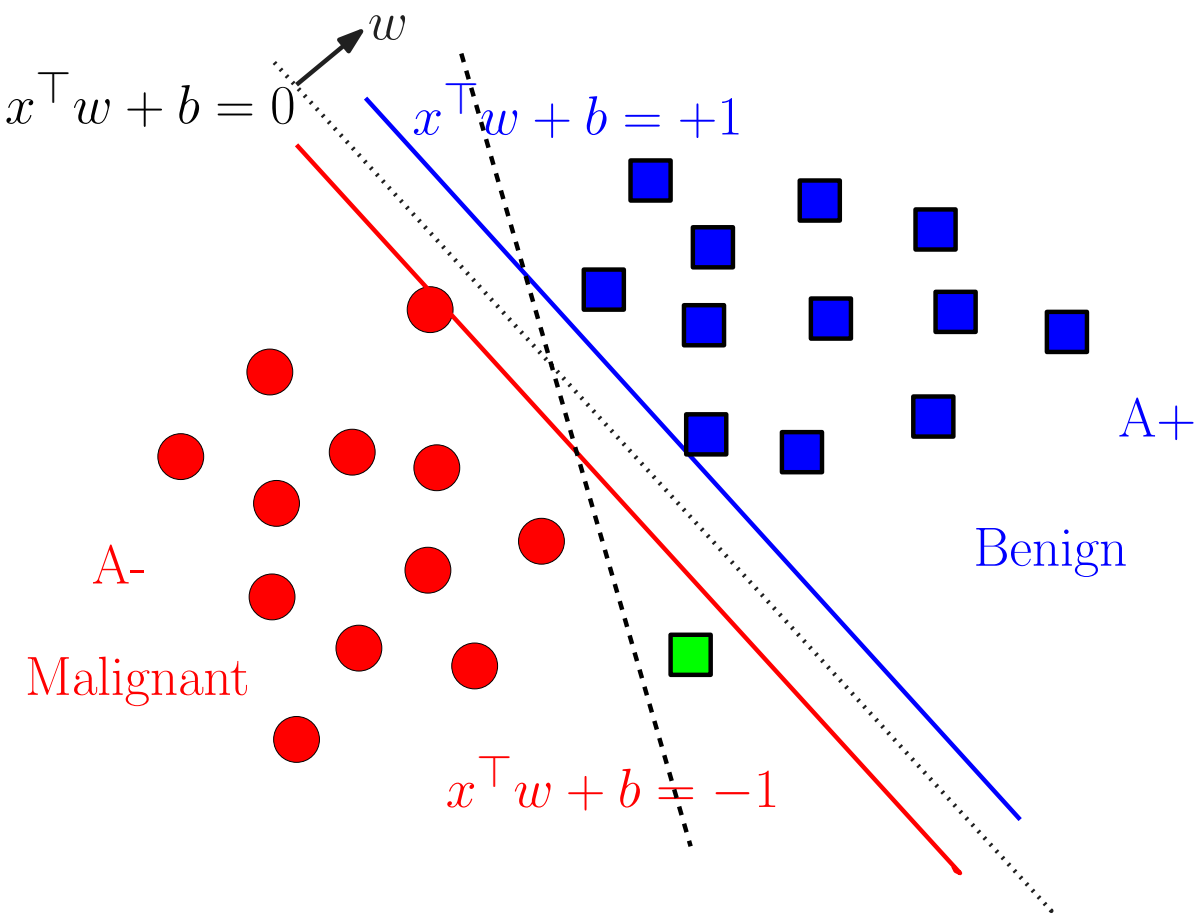
With *strictlyconvex* assumption, we have

Dual QP

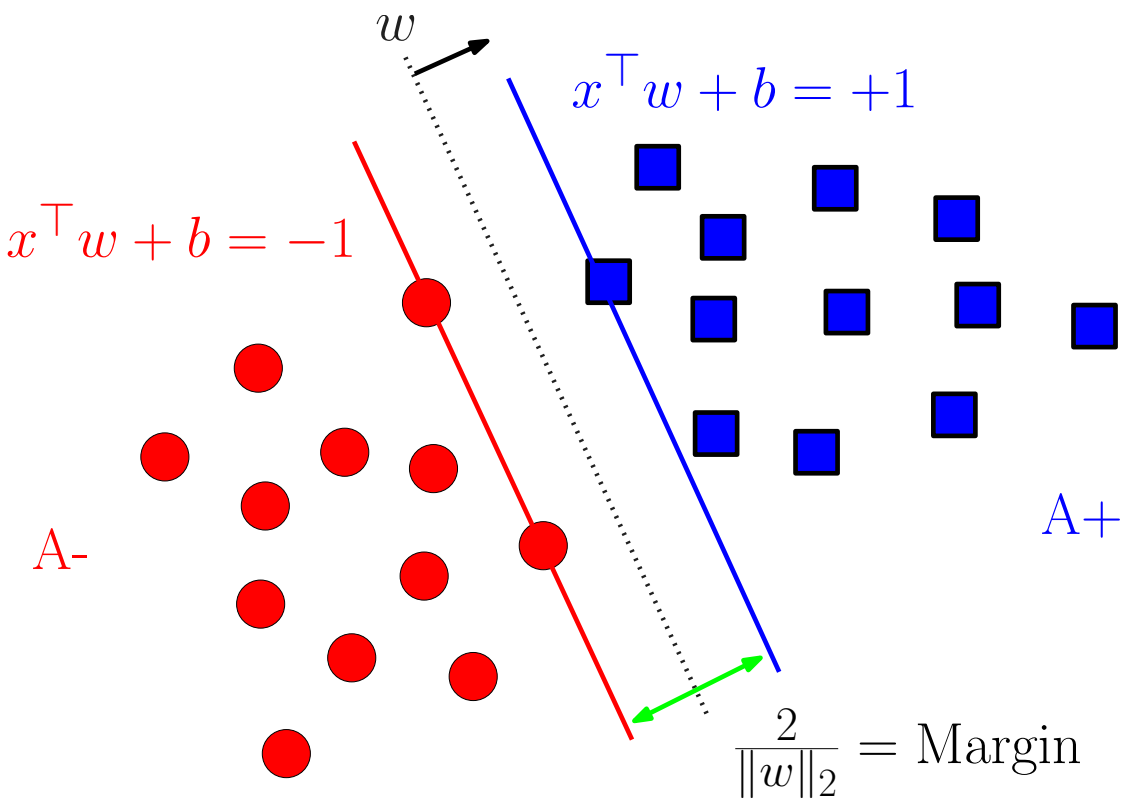
$$\begin{aligned} \max \quad & -\frac{1}{2}(p^\top + \alpha^\top A) Q^{-1} (A^\top \alpha + p) - \alpha^\top b \\ \text{s.t.} \quad & \alpha \geq \mathbf{0} \end{aligned}$$

Outline

Binary Classification Problem
Linearly Separable Case



Support Vector Machines
Maximizing the Margin between Bounding Planes



Why Use Support Vector Machines? Powerful tools for Data Mining

- SVM classifier is an optimally defined surface
- SVMs have a good geometric interpretation
- SVMs can be generated very efficiently
- Can be extended from linear to nonlinear case
  - Typically nonlinear in the input space
  - Linear in a higher dimensional "feature space"
  - Implicitly defined by a kernel function
- Have a sound theoretical foundation
  - Based on Statistical Learning Theory

Why We Maximize the Margin? (Based on Statistical Learning Theory)

- The Structural Risk Minimization (SRM):
  - The expected risk will be less than or equal to empirical risk (training error)+ VC (error) bound
- $\|w\|_2 \propto$ *VC bound*
- min *VC bound* $\Leftrightarrow$ min $\frac{1}{2}\|w\|_2^2 \Leftrightarrow$ max *Margin*

Summary the Notations

Let $S = \{(x^1, y_1), (x^2, y_2), \ldots, (x^\ell, y_\ell)$ be a training dataset and represented by matrices

$$A = \begin{bmatrix} (x^1)^\top \\ (x^2)^\top \\ \vdots \\ (x^\ell)^\top \end{bmatrix} \in \mathbb{R}^{\ell \times n}, D = \begin{bmatrix} y_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & y_\ell \end{bmatrix} \in \mathbb{R}^{\ell \times \ell}$$

$A_i w + b \geq +1, \ for \ D_{ii} = +1$
$A_i w + b \leq -1, \ for \ D_{ii} = -1$, equivalent to $D(Aw + \mathbf{1}b) \geq \mathbf{1}$, where $\mathbf{1} = [1, 1, \ldots, 1]^\top \in \mathbb{R}^\ell$

Support Vector Classification (Linearly Separable Case, Primal)

The hyperplane (w, b) is determined by solving the minimization problem:

$$\min_{(w,b)\in\mathbb{R}^{n+1}} \frac{1}{2}\|w\|_2^2$$

$$D(Aw + \mathbf{1}b) \geq \mathbf{1},$$

It realizes the maximal margin hyperplane with geometric margin

$$\gamma = \frac{1}{\|w\|_2}$$

Support Vector Classification (Linearly Separable Case, Dual Form)

The dual problem of previous MP:

$$\max_{\alpha \in R^\ell} \quad \mathbf{1}^\top \alpha - \frac{1}{2}\alpha^\top DAA^\top D\alpha$$

subject to

$$\mathbf{1}^\top D\alpha = 0, \alpha \geq \mathbf{0}$$

Applying the KKT optimality conditions, we have $w = A^\top D\alpha$. But where is b ?
Don't forget

$$\mathbf{0} \leq \alpha \perp D(Aw + \mathbf{1}b) - \mathbf{1} \geq \mathbf{0}$$

Dual Representation of SVM

(Key of Kernel Methods: $w = A^\top D\alpha^* = \sum_{i=1}^{\ell} y_i \alpha_i^* A_i^\top$)

The hypothesis is determined by (α^*, b^*)

$$\begin{aligned} h(x) &= sgn(\langle x \cdot A^\top D\alpha^* \rangle + b^*) \\ &= sgn(\sum_{i=1}^{\ell} y_i \alpha_i^* \langle x^i \cdot x \rangle + b^*) \\ &= sgn(\sum_{\alpha_i^* > 0} y_i \alpha_i^* \langle x^i \cdot x \rangle + b^*) \end{aligned}$$

Remember : $A_i^\top = x_i$

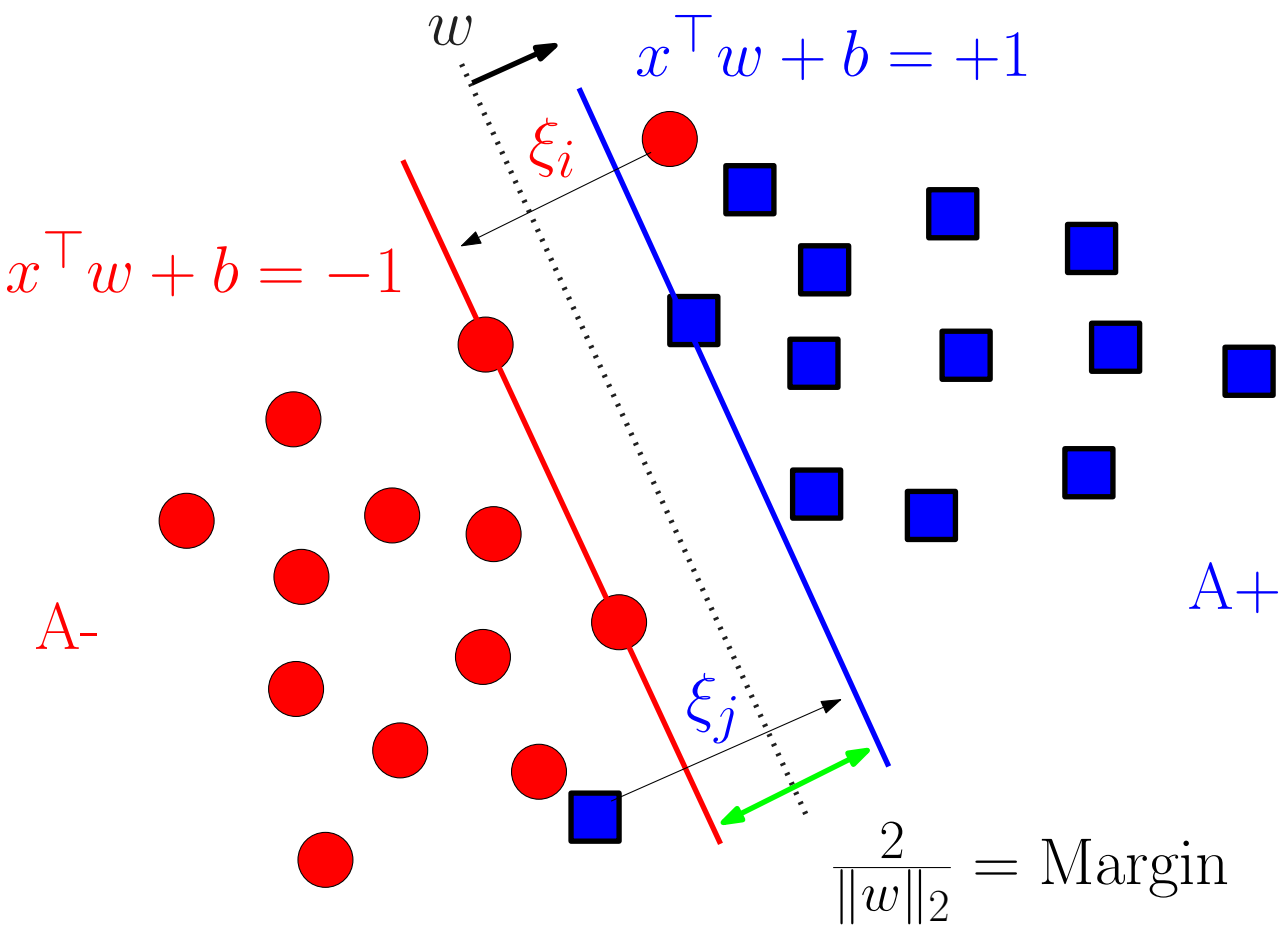
Soft Margin SVM (Nonseparable Case)

- If data are not linearly separable
  - Primal problem is infeasible
  - Dual problem is unbounded above
- Introduce the slack variable for each training point

$$y_i(w^\top x^i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i$$

- The inequality system is always feasible e.g.

$$w = \mathbf{0}, \quad b = 0, \quad \xi = \mathbf{1}$$


w
$x^\top w + b = +1$
ξ_i
$x^\top w + b = -1$
A-
A+
ξ_j
$\frac{2}{\|w\|_2} = \text{Margin}$

Robust Linear Programming
Preliminary Approach to SVM

$$\begin{aligned} \min_{w,b,\xi} \quad & \mathbf{1}^\top \xi \\ \text{s.t.} \quad & D(Aw + \mathbf{1}b) + \xi \geq \mathbf{1} \quad (LP) \\ & \xi \geq \mathbf{0} \end{aligned}$$

where ξ is nonnegative slack(*error*) vector

- The term $\mathbf{1}^\top \xi$, 1-norm measure of *error* vector, is called the *training error*
- For the linearly separable case, at solution of(LP): $\xi = \mathbf{0}$

Support Vector Machine Formulations (Two Different Measures of Training Error)

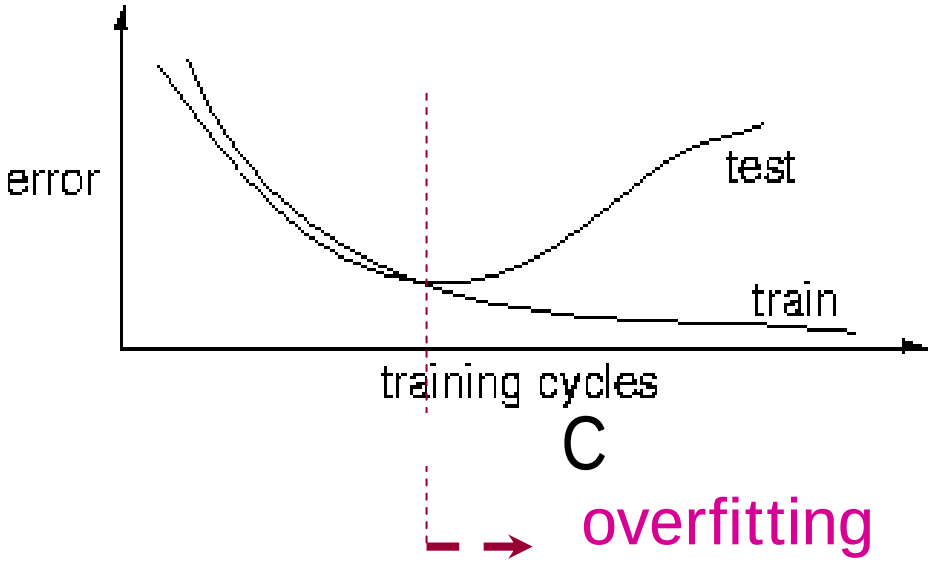
2-Norm Soft Margin:

$$\min_{(w,b,\xi)\in\mathbb{R}^{n+1+\ell}} \quad \frac{1}{2}\|w\|_2^2 + \frac{C}{2}\|\xi\|_2^2$$
$$D(Aw + \mathbf{1}b) + \xi \geq \mathbf{1}$$

1-Norm Soft Margin (Conventional SVM)

$$\min_{(w,b,\xi)\in\mathbb{R}^{n+1+\ell}} \quad \frac{1}{2}\|w\|_2^2 + C\mathbf{1}^\top\xi$$
$$D(Aw + \mathbf{1}b) + \xi \geq \mathbf{1}$$
$$\xi \geq \mathbf{0}$$

Tuning Procedure
How to determine C ?



The final value of parameter is one with the maximum testing set correctness!

1-Norm SVM (Different Measure of Margin)

1-Norm SVM:

$$\min_{(w,b,\xi)\in\mathbb{R}^{n+1+\ell}} \quad \| w \|_1 + C\mathbf{1}^\top \xi$$
$$D(Aw + \mathbf{1}b) + \xi \geq \mathbf{1}$$
$$\xi \geq \mathbf{0}$$

Equivalent to:

$$\min_{(s,w,b,\xi)\in\mathbb{R}^{2n+1+\ell}} \quad \mathbf{1}s + C\mathbf{1}^\top \xi$$
$$D(Aw + \mathbf{1}b) + \xi \geq \mathbf{1}$$
$$-s \leq w \leq s$$
$$\xi \geq \mathbf{0}$$

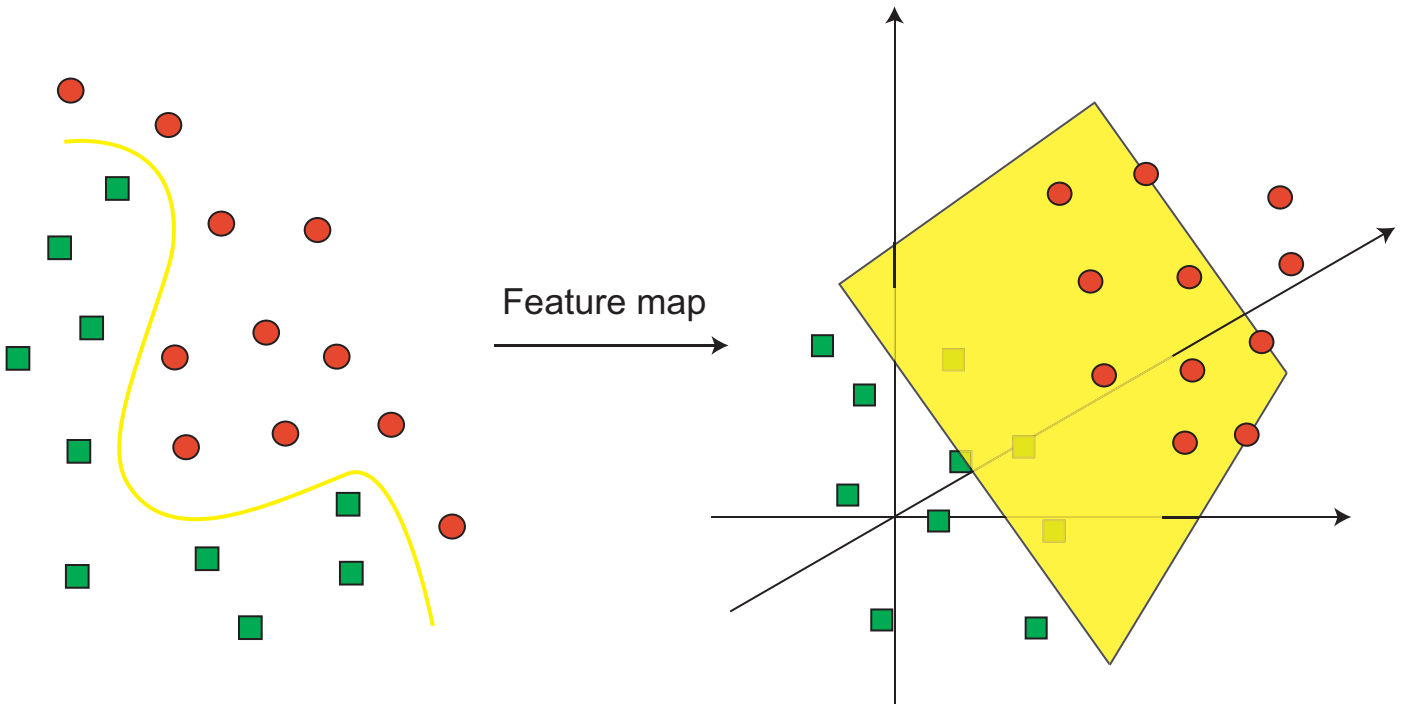
Good for feature selection and similar to the LASSO

Two-spiral Dataset (94 white Dots & 94 Red Dots)

Learning in Feature Space (Could Simplify the Classification Task)

- Learning in a high dimensional space could degrade generalization performance
  - This phenomenon is called *curse of dimensionality*
- By using a *kernel function*, that represents the inner product of training example in feature space, we never need to explicitly know the nonlinear map
  - Even do not know the dimensionality of feature space
- There is no free lunch
  - Deal with a huge and dense kernel matrix
    - Reduced kernel can avoid this difficulty

$$X \overset{\Phi}{\dashrightarrow} F$$



nonlinear pattern in data space

approximate linear pattern in feature space

Linear Machine in Feature Space

Let $\phi : X \longrightarrow F$ be a nonlinear map from the input space to some feature space

The classifier will be in the form(*primal*):

$$f(x) = (\sum_{j=1}^{?} w_j \phi_j(x)) + b$$

Make it in the *dual* form:

$$f(x) = (\sum_{i=1}^{\ell} \alpha_i y_i \langle \phi(x^i) \cdot \phi(x) \rangle) + b$$

Kernel:Represent Inner Product in Feature Space

Definition: A kernel is a function $K : X \times X \longrightarrow \mathbb{R}$ such that *for all* $x, z \in X$

$$K(x, z) = \langle \phi(x) \cdot \phi(z) \rangle$$

where $\phi : X \longrightarrow F$
The classifier will become:

$$f(x) = (\sum_{i=1}^{\ell} \alpha_i y_i K(x^i, x)) + b$$

A Simple Example of Kernel
Polynomial Kernel of Degree 2: K(x,z)=$\langle x, z\rangle^2$

Let $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \in \mathbb{R}^2$ and the nonlinear map

$\phi : \mathbb{R}^2 \longmapsto \mathbb{R}^3$ defined by $\phi(x) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{bmatrix}$.

Then $\langle \phi(x), \phi(z)\rangle = \langle x, z\rangle^2 = K(x, z)$

- There are many other nonlinear maps, $\psi(x)$, that satisfy the relation: $\langle \psi(x), \psi(z)\rangle = \langle x, z\rangle^2 = K(x, z)$

Power of the Kernel Technique

Consider a nonlinear map $\phi : \mathbb{R}^n \longmapsto \mathbb{R}^p$ that consists of distinct features of all the *monomials* of degree d.

Then $p = \binom{n+d-1}{d}$.

$x_1^3 x_2^1 x_3^4 x_4^4 \Longrightarrow$ x o o o x o x o o o o x o o o o

For example: n=11, d=10, p=92378

- Is it necessary? We only need to know $\langle \phi(x), \phi(z) \rangle$!
- This can be achieved $K(x, z) = \langle x, z \rangle^d$

Kernel Technique
Based on Mercer's Condition(1909)

- The value of kernel function represents the inner product of two training points in feature space
- Kernel function merge two steps
  1. map input data from input space to feature space (might be infinite dim.)
  2. do inner product in the feature space

Example of Kernel
$K(A, B) : \mathbb{R}^{\ell \times n} \times \mathbb{R}^{n \times \tilde{\ell}} \longmapsto R^{\ell \times \tilde{\ell}}$

$A \in \mathbb{R}^{\ell \times n}, a \in \mathbb{R}^{\ell}, \mu \in \mathbb{R}$, d is an integer:

- Polynomial Kernel:
  - $(AA^\top + \mu aa^\top)^d_\bullet$ (Linear Kernel$AA^\top$: $\mu = 0, d = 1$)
- Gaussian (Radial Basis) Kernel:
  - $K(A, A^\top)_{ij} = e^{-\mu \|A_i - A_j\|_2^2}, \quad i, j = 1, ..., m$
- The ij-entry of $K(A, A^\top)$ represents the "similarity" of data points A_i and A_j

Nonlinear Support Vector Machine (Applying the Kernel Trick)

1-Norm Soft Margin Linear SVM:

$$\max_{\alpha \in \mathbb{R}^\ell} \ \mathbf{1}^\top \alpha - \frac{1}{2}\alpha^\top DAA^\top D\alpha \ \ s.t. \ \mathbf{1}^\top D\alpha = 0, \ \ \mathbf{0} \leq \alpha \leq C\mathbf{1}$$

- Applying the kernel trick and running linear SVM in the feature space without knowing the nonlinear mapping

1-Norm Soft Margin Nonlinear SVM:

$$\max_{\alpha \in \mathbb{R}^\ell} \mathbf{1}^\top \alpha - \frac{1}{2}\alpha^\top DK(A, A^\top)D\alpha$$
$$s.t. \ \mathbf{1}^\top D\alpha = 0, \ \ \mathbf{0} \leq \alpha \leq C\mathbf{1}$$

- All you need to do is replacing $AA^\top$ by $K(A, A^\top)$

1-Norm SVM (Different Measure of Margin)

1-Norm SVM:

$$\min_{(w,b,\xi)\in\mathbb{R}^{n+1+\ell}} \quad \| w \|_1 + C\mathbf{1}^\top \xi$$
$$D(Aw + \mathbf{1}b) + \xi \geq \mathbf{1}$$
$$\xi \geq \mathbf{0}$$

Equivalent to:

$$\min_{(s,w,b,\xi)\in\mathbb{R}^{2n+1+\ell}} \quad \mathbf{1}s + C\mathbf{1}^\top \xi$$
$$D(Aw + \mathbf{1}b) + \xi \geq \mathbf{1}$$
$$-s \leq w \leq s$$
$$\xi \geq \mathbf{0}$$

Good for feature selection and similar to the LASSO

Outline

How to Evaluated What's been learned
Cost is not sensitive

- Measure the performance of a classifier in terms of error rate or accuracy

$$\textit{Error rate} = \frac{\textit{Number of misclassified point}}{\textit{Total number of data point}}$$

Main Goal: Predict the unseen class label for new data

- We have to asses a classifier's error rate on a set that play no rule in the learning class
- Split the data instances in hand into two parts:
  1. Training set: for learning the classifier.
  2. Testing set: for evaluating the classifier.

k-fold Stratified Cross Validation
Minimize the usage of the data in hands

- Split the data into k approximately equal partitions.
- Each in turn is used for testing while the remainder is used for training.
- The labels $(+/-)$ in the training and testing sets should be in about right proportion.
  - Doing the random splitting in the positive class and negative class respectively will guarantee it.
  - This procedure is called stratification.
- Leave-one-out cross-validation if $k = \#$ of data point.
  - No random sampling is involved but nonstratified.

How to Compare Two Classifier? Testing Hypothesis:Paired t-test

- We compare two leaving algorithm by comparing the average error rate over several cross-validations.
- Assume the same cross-validation split can be used for both methods

$$H_0 : \bar{d} = 0 \text{ v.s } H_1 : \bar{d} \neq 0$$

where $\bar{d} = \frac{1}{k}\sum_{i=1}^{k} d_i$ and $d_i = x_i - y_i$

- The t-statistic:

$$t = \frac{\bar{d}}{\sqrt{\sigma_d^2/k}}$$

How to Evaluate What's Been Learned? When cost is sensitive

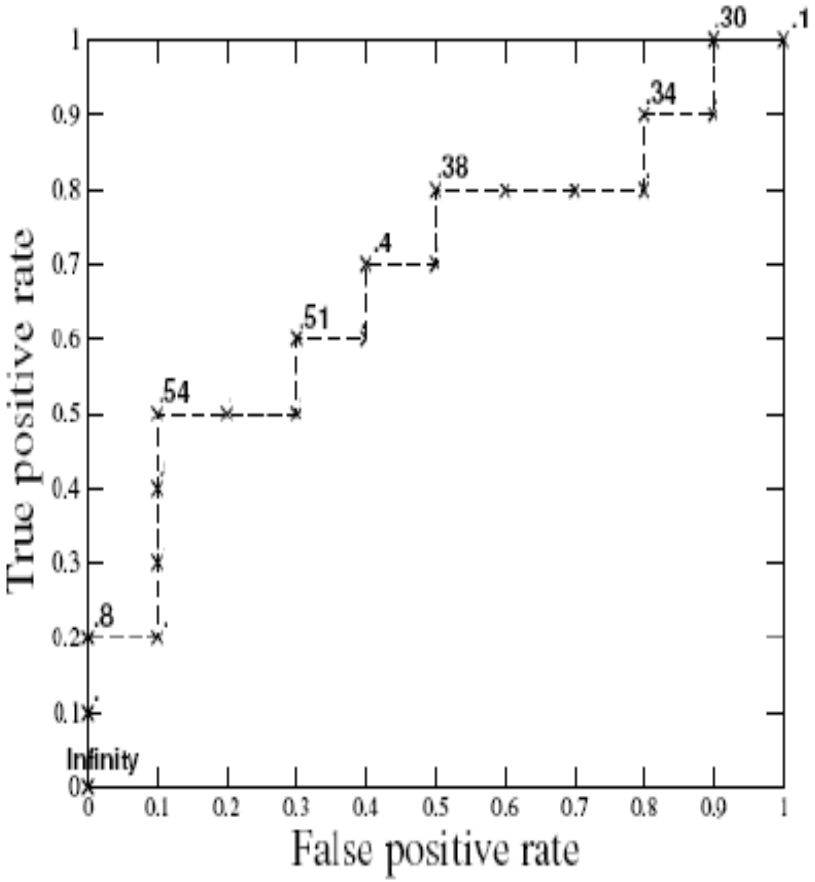
- Two types error will occur: False Positive(FP) & False Negative(FN)
- For binary classification problem, the results can be summarized in a 2×2 confusion matrix.

	Predicted Class	
Actual Class	True Pos. (TP)	False Neg. (FN)
	False Pos. (FP)	True Neg. (FN)

ROC Curve
Receiver Operating Characteristic Curve

- An evaluation method for learning models.
- What it concerns about is the Ranking of instances made by the learning model.
- A Ranking means that we sort the instances w.r.t the probability of being a positive instance from high to low.
- ROC curve plots the true positive rate (TPr) as a function of the false positive rate (FPr).

An example of ROC Curve



Inst ID	Class	Score
1	P	0.51
2	P	0.8
3	P	0.3
4	P	0.55
5	P	0.4
6	P	0.34
7	P	0.9
8	P	0.54
9	P	0.38
10	P	0.6
11	N	0.35
12	N	0.52
13	N	0.36
14	N	0.37
15	N	0.7
16	N	0.1
17	N	0.39
18	N	0.53
19	N	0.5
20	N	0.33

Sort →

Inst ID	Class	Score
7	P	0.9
2	P	0.8
15	N	0.7
10	P	0.6
4	P	0.55
8	P	0.54
18	N	0.53
12	N	0.52
1	P	0.51
19	N	0.5
5	P	0.4
17	N	0.39
9	P	0.38
14	N	0.37
13	N	0.36
11	N	0.35
6	P	0.34
20	N	0.33
3	P	0.3
16	N	0.1

Using ROC to Compare Two Methods

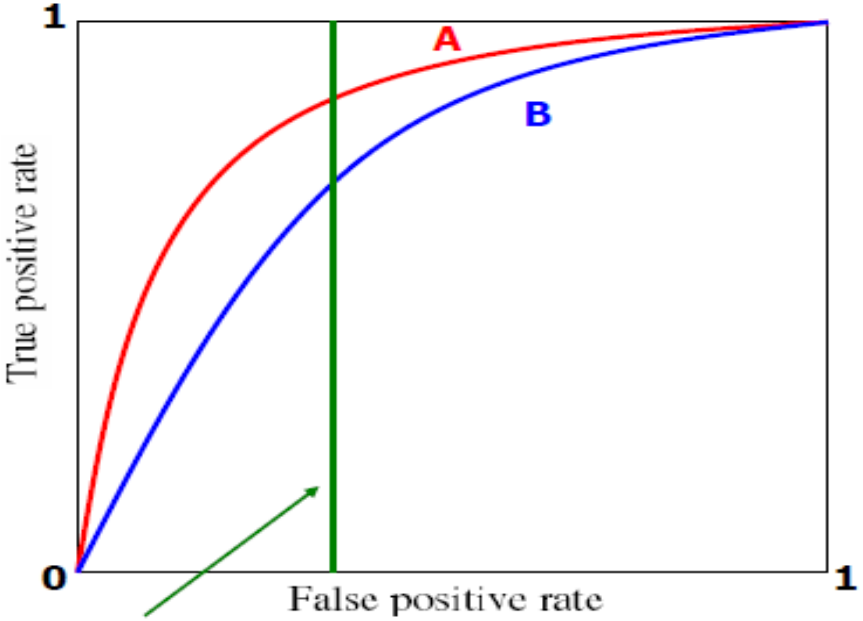
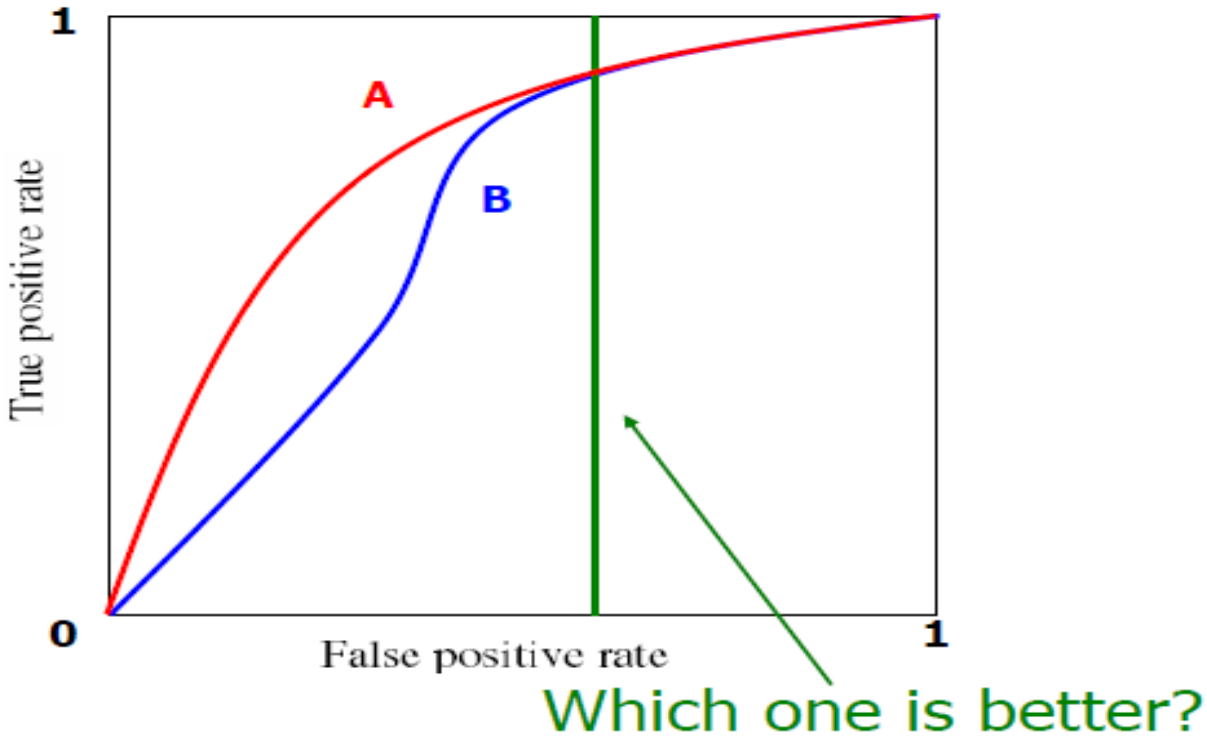


Figure: Under the same FP rate, method A is better than B.

Using ROC to Compare Two Methods



Area under the Curve (AUC)

- An index of ROC curve with range from 0 to 1.
- An AUC value of 1 corresponds to a perfect Ranking (all positive instances are ranked high than all negative instance).
- A simple formula for calculating AUC:

$$AUC = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} I_{f(x_i)>f(x_j)}}{m}$$

where m: number of positive instances.
n: number of negative instances.

Performance Measures in Information Retrieval (IR)

- An IR system, such as Google, for given a query (keywords search) will try to retrieve all relevant documents in a corpus.
  - Documents returned that are NOT relevant: FP.
  - The relevant documents that are NOT return: FN.
- Performance measures in IR, Recall & Precision.

$$Recall = \frac{TP}{TP + TN}$$

and

$$Precision = \frac{TP}{TP + FP}$$

Balance the Trade-off between Recall and Precision

- Two extreme cases:
  1. Return only document with 100% confidence then precision=1 but recall will be very small.
  2. Return all documents in the corpus then recall=1 but precision will be very small.
- F-measure balances this trade-off:

$$F - measure = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

Reference

- C. J. C Burges. "A Tutorial on Support Vector Machines for Pattern Recognition", Data Mining and Knowledge Discovery, Vol. 2, No. 2, (1998) 121-167.
- N. Cristianini and J. Shawe-Taylor. "An Introduction to Support Vector Machines", Cambridge University Press,(2000).