Lecture Notes on

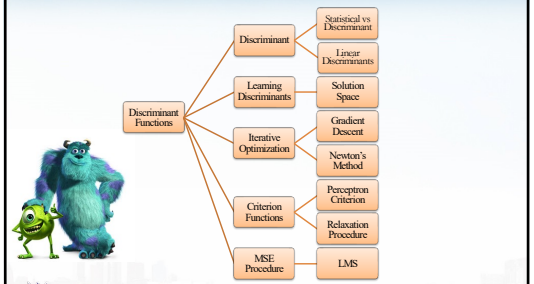
Pattern Recognition

Session 07(A): Discriminant Functions (I)

Gang Li
School of Information Technology
Deakin University, VIC 3125, Australia

1

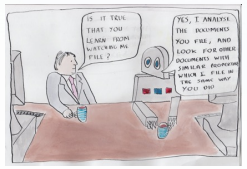
Road map



2

Classification

- Classification
- By Regression?



3

Classification

- **Classification**
  - Learn a mapping from observations to discrete labels given a set of training set
  - Example:
    - Spam Classification
      - Observations are emails
      - Labels are {**spam**, **not-spam**} (Binary Classification)
      - Given a set of labeled emails, we want to predict whether a new email is spam or not-spam

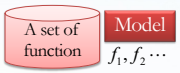
4

Classification

- **Classification**
  - Learn a mapping from observations to discrete labels given a set of training set
  - Example:
    - Click-through Rate Prediction
      - Observations are user-ad-publisher triples
      - Labels are {**not-click**, **click**} (Binary Classification)
      - Given a set of labeled observations, we want to predict whether a new user-ad-publisher triple will result in a click

5

ML Framework

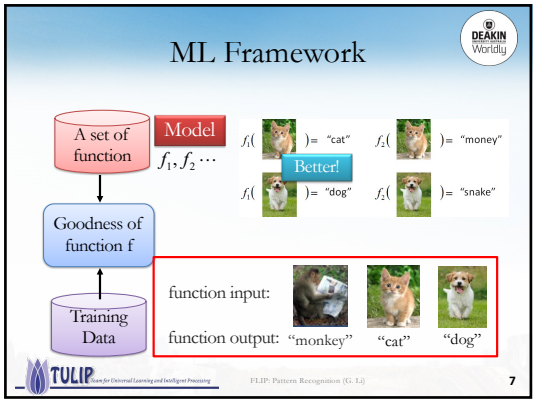


$f_1($  $)=$ "cat" $f_2($  $)=$ "money"

$f_1($  $)=$ "dog" $f_2($  $)=$ "snake"

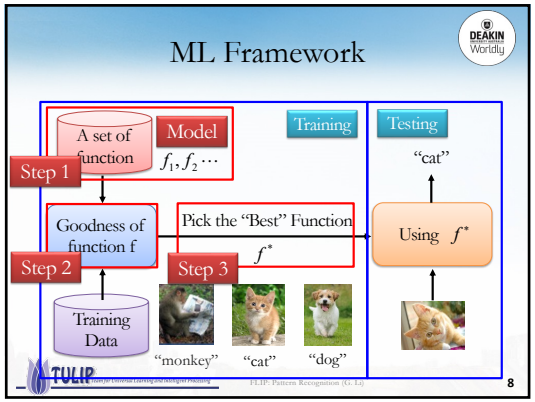
6

ML Framework



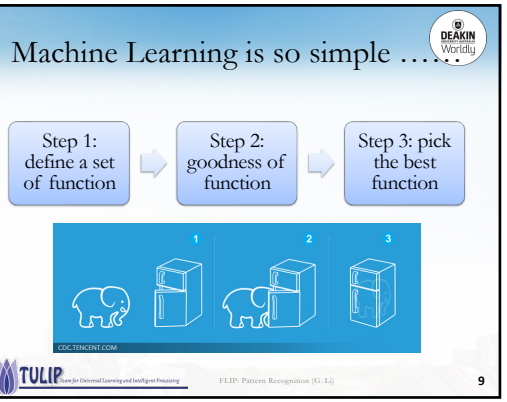
7

ML Framework



8

Machine Learning is so simple ……



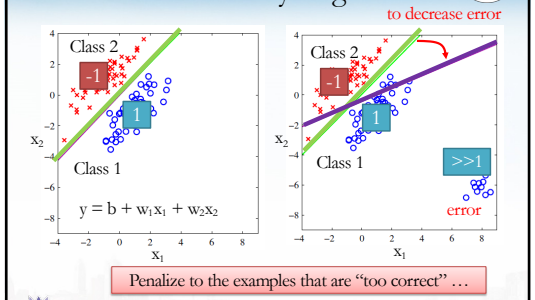
9

Classification by Regression?

- Classification as Regression?
  - Binary classification as example
    - Training:
      - class 1 means the target is 1; class 2 means the target is -1
    - Testing:
      - closer to 1 → class 1; closer to -1 → class 2
      - Or if > 0 then class 1, else if <0 then class 2
  - Multiple class:
    - Class 1 means the target is 1; Class 2 means the target is 2; Class 3 means the target is 3 …… problematic

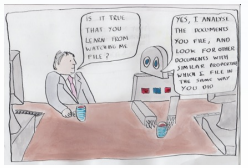
10

Classification by Regression?



11

Generative vs. Discriminative



- 3 Steps
- Probability-based Method
- From Generative to Discriminative Model

12

Three Steps

- Function Set (Generative Model):

$$f(X) = \begin{cases} class\ 1 & if \quad P(C_1|x) > 0.5 \\ class\ 2 & else\ if\ P(C_1|x) \le 0.5 \end{cases} \qquad P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

- Goodness of a function:
  - The parameters of the distribution that maximizing the likelihood (the probability of generating data)
- Find the best function:
  - MLE
  - Bayesian Estimation

13

Generative v.s. Discriminative

- Example

Training Data: Class 1 | Class 2 X 4 | Class 2 X 4 | Class 2 X 4

Testing Data: Class 1? Class 2?

How about Naïve Bayes?

$$P(x|C_i) = P(x_1|C_i)P(x_2|C_i)$$

14

Generative v.s. Discriminative

- Example

Training Data: Class 1 | Class 2 X 4 | Class 2 X 4 | Class 2 X 4

$$P(C_1) = \frac{1}{13} \quad P(x_1 = 1|C_1) = 1 \quad P(x_2 = 1|C_1) = 1$$
$$P(C_2) = \frac{12}{13} \quad P(x_1 = 1|C_2) = \frac{1}{3} \quad P(x_2 = 1|C_2) = \frac{1}{3}$$

15

Generative v.s. Discriminative

- Example

Training Data: Class 1 | Class 2 X 4 | Class 2 X 4 | Class 2 X 4

Testing Data

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)} = \frac{1}{1 + \frac{P(x|C_2)P(C_2)}{P(x|C_1)P(C_1)}} = \frac{1}{1 + exp(-z)} = \sigma(z)$$

<0.5

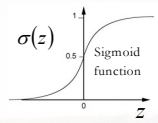
$1\times1 \quad \frac{1}{13} \quad \frac{1}{3}\times\frac{1}{3} \quad \frac{12}{13}$

16

Posterior Probability

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)} = \frac{1}{1 + \frac{P(x|C_2)P(C_2)}{P(x|C_1)P(C_1)}} = \frac{1}{1 + exp(-z)} = \sigma(z)$$

$$z = ln\frac{P(x|C_1)P(C_1)}{P(x|C_2)P(C_2)}$$



17

Posterior Probability

$$P(C_1|x) = \sigma(z) \quad \text{sigmoid} \quad z = ln\frac{P(x|C_1)P(C_1)}{P(x|C_2)P(C_2)}$$

$$z = ln\frac{P(x|C_1)}{P(x|C_2)} + ln\boxed{\frac{P(C_1)}{P(C_2)}} \Rightarrow \frac{\frac{N_1}{N_1+N_2}}{\frac{N_2}{N_1+N_2}} = \frac{N_1}{N_2}$$

$$P(x|C_1) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\Sigma^1|^{1/2}}exp\left\{-\frac{1}{2}(x-\mu^1)^T(\Sigma^1)^{-1}(x-\mu^1)\right\}$$

$$P(x|C_2) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\Sigma^2|^{1/2}}exp\left\{-\frac{1}{2}(x-\mu^2)^T(\Sigma^2)^{-1}(x-\mu^2)\right\}$$

18

Generative vs Discriminative Model

$$z = ln\frac{P(x|C_1)}{P(x|C_2)} + ln\frac{P(C_1)}{P(C_2)}$$

$$ln\frac{\frac{1}{(2\pi)^{n/2}}\frac{1}{|\Sigma^1|^{1/2}}exp\left\{-\frac{1}{2}(x-\mu^1)^T(\Sigma^1)^{-1}(x-\mu^1)\right\}}{\frac{1}{(2\pi)^{n/2}}\frac{1}{|\Sigma^2|^{1/2}}exp\left\{-\frac{1}{2}(x-\mu^2)^T(\Sigma^2)^{-1}(x-\mu^2)\right\}}$$

$$= ln\frac{|\Sigma^2|^{1/2}}{|\Sigma^1|^{1/2}} - \frac{1}{2}[(x-\mu^1)^T(\Sigma^1)^{-1}(x-\mu^1) - (x-\mu^2)^T(\Sigma^2)^{-1}(x-\mu^2)]$$

19

Generative vs Discriminative Model

$$z = ln\frac{P(x|C_1)}{P(x|C_2)} + ln\frac{P(C_1)}{P(C_2)}$$

$$(x-\mu^1)^T(\Sigma^1)^{-1}(x-\mu^1)$$
$$= x^T(\Sigma^1)^{-1}x - x^T(\Sigma^1)^{-1}\mu^1 - (\mu^1)^T(\Sigma^1)^{-1}x + (\mu^1)^T(\Sigma^1)^{-1}\mu^1$$
$$= x^T(\Sigma^1)^{-1}x - 2(\mu^1)^T(\Sigma^1)^{-1}x + (\mu^1)^T(\Sigma^1)^{-1}\mu^1$$

$$(x-\mu^2)^T(\Sigma^2)^{-1}(x-\mu^2)$$
$$= x^T(\Sigma^2)^{-1}x - 2(\mu^2)^T(\Sigma^2)^{-1}x + (\mu^2)^T(\Sigma^2)^{-1}\mu^2$$

$$z = ln\frac{|\Sigma^2|^{1/2}}{|\Sigma^1|^{1/2}} - \frac{1}{2}x^T(\Sigma^1)^{-1}x + (\mu^1)^T(\Sigma^1)^{-1}x - \frac{1}{2}(\mu^1)^T(\Sigma^1)^{-1}\mu^1 + \frac{1}{2}x^T(\Sigma^2)^{-1}x - (\mu^2)^T(\Sigma^2)^{-1}x + \frac{1}{2}(\mu^2)^T(\Sigma^2)^{-1}\mu^2 + ln\frac{N_1}{N_2}$$

20

Generative vs Discriminative Model

$$z = ln\frac{|\Sigma^2|^{1/2}}{|\Sigma^1|^{1/2}} - \frac{1}{2}x^T(\Sigma^1)^{-1}x + (\mu^1)^T(\Sigma^1)^{-1}x - \frac{1}{2}(\mu^1)^T(\Sigma^1)^{-1}\mu^1 + \frac{1}{2}x^T(\Sigma^2)^{-1}x - (\mu^2)^T(\Sigma^2)^{-1}x + \frac{1}{2}(\mu^2)^T(\Sigma^2)^{-1}\mu^2 + ln\frac{N_1}{N_2}$$

$\Sigma_1 = \Sigma_2 = \Sigma$

$$z = \underbrace{(\mu^1-\mu^2)^T\Sigma^{-1}}_{\boldsymbol{w^T}}x - \underbrace{\frac{1}{2}(\mu^1)^T\Sigma^{-1}\mu^1 + \frac{1}{2}(\mu^2)^T\Sigma^{-1}\mu^2 + ln\frac{N_1}{N_2}}_{\mathrm{b}}$$

$P(C_1|x) = \sigma(w \cdot x + b)$ How about directly find $\boldsymbol{w}$ and b?

In generative model, we estimate $N_1, N_2, \mu^1, \mu^2, \Sigma$

Then we have $\boldsymbol{w}$ and b

21

Generative vs. Discriminative

$$P(C_1|x) = \sigma(w \cdot x + b)$$

directly find $\boldsymbol{w}$ and b

Find $\mu^1, \mu^2, \Sigma^{-1}$

$$w^T = (\mu^1-\mu^2)^T\Sigma^{-1}$$

$$b = -\frac{1}{2}(\mu^1)^T(\Sigma^1)^{-1}\mu^1 + \frac{1}{2}(\mu^2)^T(\Sigma^2)^{-1}\mu^2 + ln\frac{N_1}{N_2}$$

Will we obtain the same set of w and b?

The same model (function set), but different function may be selected by the same training data.

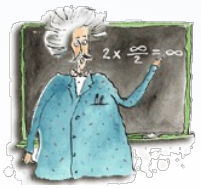
22

Generative vs. Discriminative

- Usually, people believe discriminative model is better
- Benefit of generative model
  - With the assumption of probability distribution
    - less training data is needed
    - more robust to the noise
  - Priors and class-dependent probabilities can be estimated from different sources.

23

Discriminant Approach



- Discriminant Approach
- Discriminant Function
- Linear Discriminant Function

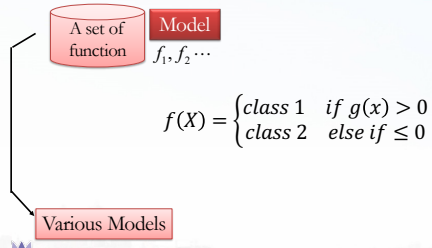
24

Statistical vs Discriminant Approach

- ***Statistical*** approaches find the decision boundary by first estimating the probability distribution of the patterns belonging to each class.
- In the ***discriminant*** approach, the decision boundary is constructed explicitly without assuming a probability distribution.

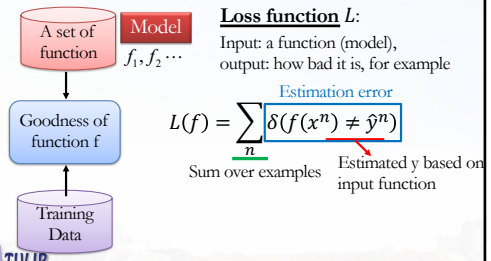
25

Step 1: Model



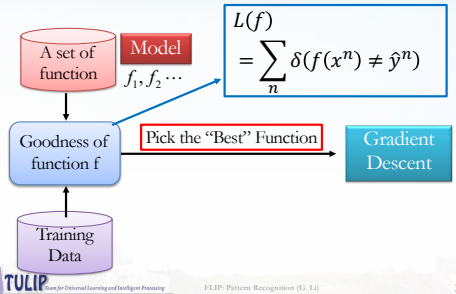
26

Step 2: Goodness of Function



27

Step 3: Best Function



28

Discriminant Approach

- Specify ***parametric form*** of the decision boundary (e.g., linear or quadratic) .
- Find the ***best*** decision boundary of the specified form using a set of training examples x_k.
  - This is performed by ***minimizing*** a criterion function (e.g., "training error" or "sample risk"):

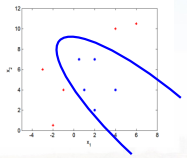
$$J(w) = \frac{1}{n}\sum_{k=1}^{n}[z_k - g(x_k, w)]^2$$

correct class predicted class

29

Discriminant Functions: two-categories case

- Decide w_1 if $g(\mathbf{x}) > 0$ and w_2 if $g(\mathbf{x}) < 0$
  - If $g(\mathbf{x})=0$, then $\mathbf{x}$ lies on the decision boundary and can be assigned to either class.
- Classification is viewed as finding a decision boundary that separates the data belonging to different classes.



30

Linear Discriminant Functions: two-categories case

- A linear discriminant function has the following form:

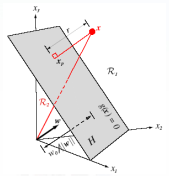
$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0 = \sum_{i=1}^{d} w_i x_i + w_0$$

- The decision boundary, given by g(**x**)=0, is a ***hyperplane*** where the orientation of the hyperplane is determined by **w** and its location by w_0.
  - **w** is the normal to the hyperplane
  - If w_0=0, the hyperplane passes through the origin

31

Geometric Interpretation of g(x)

- g(x) provides ***an algebraic measure of the distance*** of x from the hyperplane.



x can be expressed as follows:

$$\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

- *direction of r*

$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0 = \mathbf{w}^t\left(\mathbf{x}_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|}\right) + w_0$$

$$= \mathbf{w}^t\mathbf{x}_p + r\frac{\mathbf{w}^t\mathbf{w}}{\|\mathbf{w}\|} + w_0 = r\|\mathbf{w}\|$$

32

Geometric Interpretation of g(x)

- Therefore, the distance of x from the hyperplane is given by:

$$r = \frac{g(\mathbf{x})}{\|\mathbf{w}\|}$$

- w_0 determines the distance of the hyperplane from the origin:
  - setting x=0: $r = \frac{w_0}{\|\mathbf{w}\|}$

33

Linear Discriminant Functions: multi-category case

- There are several ways to devise multi-category classifiers using linear discriminant functions:
  1. One against the rest (i.e., c-1 two-class problems)

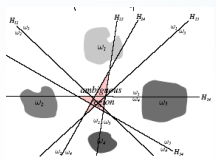
problem
- ambiguous regions

34

Linear Discriminant Functions: multi-category case

- There are several ways to devise multi-category classifiers using linear discriminant functions:
  2. One against another (i.e., c(c-1)/2 pairs of classes)

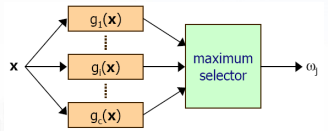
problem
- ambiguous regions



35

Linear Discriminant Functions: multi-category case

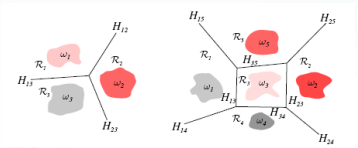
- To avoid the problem of ambiguous regions:
  - Define c linear discriminant functions
  - Assign **x** to ω_i if $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$.
- The resulting classifier is called a ***linear machine***



36

Linear Discriminant Functions: multi-category case

- A linear machine divides the feature space in c ***convex*** decisions regions.
  - If x is in region R_i, the $g_i(x)$ is the largest.



- c(c-1)/2 pairs of regions but typically less decision boundaries

37

Linear Discriminant Functions: multi-category case

- The boundary between two regions R_i and R_j is a portion of the hyperplane H_{ij} given by:

$$g_i(\mathbf{x}) = g_j(\mathbf{x}) \quad or \quad g_i(\mathbf{x}) - g_j(\mathbf{x}) = 0$$
$$or \quad (\mathbf{w}_i - \mathbf{w}_j)^t \mathbf{x} + (w_{i0} - w_{j0}) = 0$$

- $(w_i - w_j)$ is normal to H_{ij} and the signed distance from x to H_{ij} is

$$r = \frac{g_i(\mathbf{x}) - g_j(\mathbf{x})}{\| \mathbf{w}_i - \mathbf{w}_j \|}$$

38

Higher Order Discriminant Functions

- Can produce more complicated decision boundaries than linear discriminant functions.

Linear discriminant: $g(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + w_0 = \sum_{i=1}^{d} w_i x_i + w_0$

Quadratic discriminant: obtained by adding terms corresponding to products of pairs of components of x

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=1}^{d} x_i x_j w_{ij}$$

Polynomial discriminant: obtained by adding terms such as $x_i x_j x_k w_{ijk}$.

39

Generalized discriminants

- Generalized linear discriminant functions

$$g(\mathbf{x}) = \sum_{i=1}^{\hat{d}} a_i y_i(\mathbf{x}) \quad or \quad g(\mathbf{x}) = \mathbf{a}^t \mathbf{y}$$

  - a is a $\hat{d}$-dimensional weight vector
  - φ functions $y_i(\mathbf{x})$ map points from the d-dimensional $\mathbf{x}$-space to the $\hat{d}$-dimensional $\mathbf{y}$-space
    - usually, $\hat{d} >> d$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_d \end{bmatrix} \Rightarrow \begin{bmatrix} y_1(\mathbf{x}) \\ y_2(\mathbf{x}) \\ \dots \\ y_{\hat{d}}(\mathbf{x}) \end{bmatrix}$$

40

Generalized discriminants

- Generalized linear discriminant functions

$$g(\mathbf{x}) = \sum_{i=1}^{\hat{d}} a_i y_i(\mathbf{x}) \quad or \quad g(\mathbf{x}) = \mathbf{a}^t \mathbf{y}$$

  - The resulting discriminant function is ***not*** linear in x, but it ***is*** linear in y.
  - The generalized discriminant separates points in the transformed space by a ***hyperplane*** passing through the ***origin***.

41

Generalized discriminants

- Example

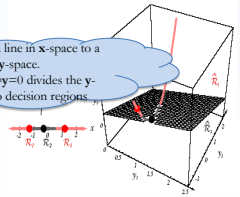
$g(x) = -1 + x + 2x^2$

$$\mathbf{a} = \begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix}$$

- The corresponding decision regions R_1, R_2 in the **x**-space are **not** simply connected!

- g(x) maps a line in **x**-space to a parabola in **y**-space.
- The plane **a**·**y**=0 divides the **y**-space in two decision regions

$$y = \begin{bmatrix} y_1(x) \\ y_2(x) \\ y_3(x) \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ x^2 \end{bmatrix}$$



$g(x) > 0 \;\; if \;\; x < -1 \; or \; x > 0.5$

42

Learning Discriminants

- Linear Separable
- Solution Space

43

Learning: two-category, linearly separable case

- Task: Given a linear discriminant function, **learn** the weights **w** using a set of *n* labeled samples $\mathbf{x}_i$
  - i.e., each $\mathbf{x}_i$ has a class label ω_1 or ω_2

$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0 = \sum_{i=1}^{d} w_i x_i + w_0$$

44

Augmented feature/weight vectors

- Increasing dimensionality simplifies things:
  - Find a single weight vector **a** in **y**-space.
  - Decision hyperplane, given by g(**y**)=0, in **y**-space passes through the origin (but could be at any position in **x**-space).

$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0 = \sum_{i=1}^{d} w_i x_i + x_0 w_0 = \sum_{i=0}^{d} w_i x_i = \mathbf{a}^t\mathbf{y}$$

$$(i.e.,\ x_0 = 1)$$

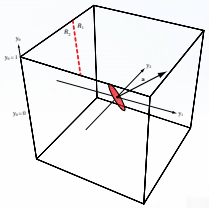
$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_d \end{bmatrix} \Rightarrow \mathbf{a} = \begin{bmatrix} w_0 \\ w_1 \\ \dots \\ w_d \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_d \end{bmatrix} \Rightarrow \mathbf{y} = \begin{bmatrix} 1 \\ x_1 \\ \dots \\ x_d \end{bmatrix}$$

- dimensionality: d → (d+1)

45

Classification in augmented space

- Given a discriminant function $g(\mathbf{x})=\mathbf{a}^t\mathbf{y}$, the classification rule is:
  - If $\mathbf{a}^t\mathbf{y}_i>0$ assign $\mathbf{y}_i$ to ω_1
  - else if $\mathbf{a}^t\mathbf{y}_i<0$ assign $\mathbf{y}_i$ to ω



46

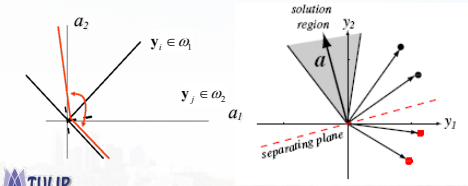
Learning: two-category, linearly separable case

- Given a discriminant function $g(\mathbf{x})=\mathbf{a}^t\mathbf{y}$, the goal is to **learn** the weights **a** using a set of *n* labeled samples $\mathbf{y}_i$
  - Every training sample $\mathbf{y}_i$ places a ***constraint*** on the weight vector **a**.
    - $\mathbf{a}^t\mathbf{y}=0$ defines a hyperplane in solution space having **y** as a normal vector.
  - Given *n* examples, the solution **a** must lie on the ***intersection*** of *n* half-spaces

47

Effect of training examples on solution

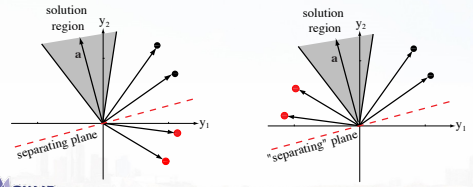
- Solution can be visualized either in the ***solution*** space or the ***feature*** space
  - solution space (a_1, a_2)
  - feature space (y_1, y_2)



48

Normalized Version

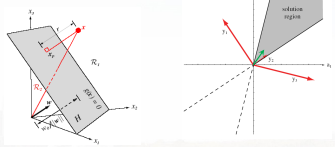
- If $\mathbf{y}_i$ in ω_2, replace $\mathbf{y}_i$ by $-\mathbf{y}_i$
- Find $\mathbf{a}$ such that: $\mathbf{a}^t\mathbf{y}_i > 0$



49

Solution Uniqueness/Constraints

- Solution vector is usually not unique; we can impose certain constraints, e.g.:
  - **Method 1**: find unit-length weight vector that maximizes the minimum distance from the samples to the separating plane.



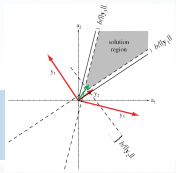
50

Solution Uniqueness/Constraints

- Solution vector is usually not unique; we can impose certain constraints, e.g.:
  - **Method 2**: find the minimum-length weight vector satisfying the constraint below where b is a positive constant called margin.

$$g(\mathbf{x}) = \mathbf{a}^t\mathbf{y}_i \geq b > 0$$

Main objective:
- move solution to the center of the feasible region as this solution is more likely to classify new test samples correctly.



51

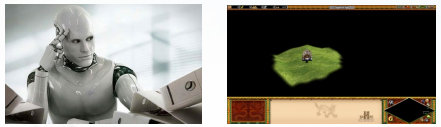
Iterative Optimization

- Gradient Descent
- Why it works?
- Newton's Method

52

Gradient Descent

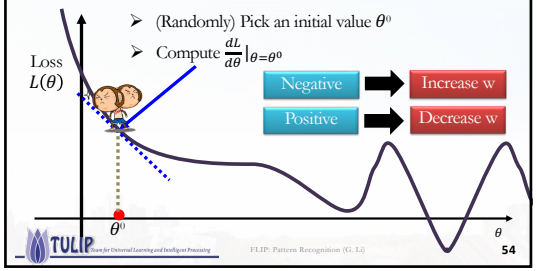
- Gradient Descent is the "learning" of machines in deep learning ……
  - Even AlphaGo using this approach.



53

Gradient Descent

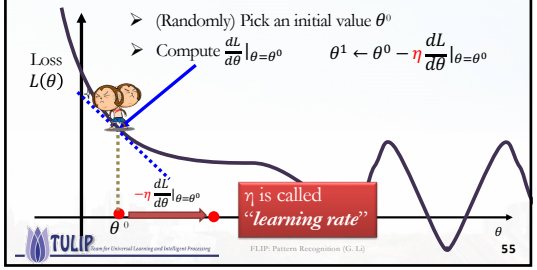
- Loss function $L(\theta)$ with one parameter θ: $\theta^* = arg\min_\theta L(\theta)$
  - (Randomly) Pick an initial value θ^0
  - Compute $\frac{dL}{d\theta}|_{\theta=\theta^0}$



54

Gradient Descent

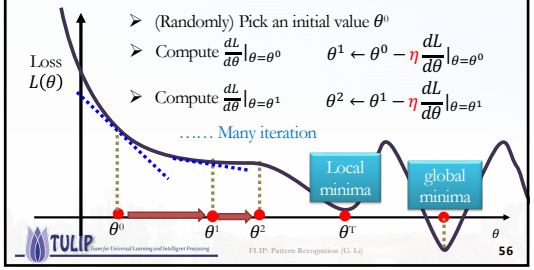
- Loss function $L(\theta)$ with one parameter θ: $\theta^* = arg\min_\theta L(\theta)$
  - (Randomly) Pick an initial value θ^0
  - Compute $\frac{dL}{d\theta}|_{\theta=\theta^0}$ $\theta^1 \leftarrow \theta^0 - \eta \frac{dL}{d\theta}|_{\theta=\theta^0}$



55

Gradient Descent

- Loss function $L(\theta)$ with one parameter θ: $\theta^* = arg\min_\theta L(\theta)$
  - (Randomly) Pick an initial value θ^0
  - Compute $\frac{dL}{d\theta}|_{\theta=\theta^0}$ $\theta^1 \leftarrow \theta^0 - \eta \frac{dL}{d\theta}|_{\theta=\theta^0}$
  - Compute $\frac{dL}{d\theta}|_{\theta=\theta^1}$ $\theta^2 \leftarrow \theta^1 - \eta \frac{dL}{d\theta}|_{\theta=\theta^1}$
  - …… Many iteration



56

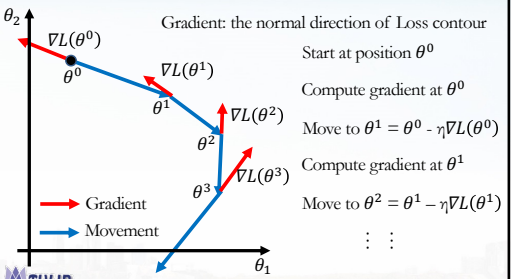
Gradient Descent

- How about 2 parameters? $\theta_1, \theta_2 = arg\min_{\theta_1, \theta_2} L(\theta_1, \theta_2)$
  - Randomly start at $\theta^0 = \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \end{bmatrix}$
  - Iteratively Compute $\nabla L(\theta) = \begin{bmatrix} \partial L(\theta_1)/\partial\theta_1 \\ \partial L(\theta_2)/\partial\theta_2 \end{bmatrix}$

$$\theta^1 = \theta^0 - \eta\nabla L(\theta^0) \qquad \begin{bmatrix} \theta_1^1 \\ \theta_2^1 \end{bmatrix} = \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \end{bmatrix} - \eta \begin{bmatrix} \partial L(\theta_1^0)/\partial\theta_1 \\ \partial L(\theta_2^0)/\partial\theta_2 \end{bmatrix}$$

57

Gradient Descent



Gradient: the normal direction of Loss contour

Start at position θ^0

Compute gradient at θ^0

Move to $\theta^1 = \theta^0 - \eta\nabla L(\theta^0)$

Compute gradient at θ^1

Move to $\theta^2 = \theta^1 - \eta\nabla L(\theta^1)$

⋮ ⋮

58

Regression by Gradient Descent

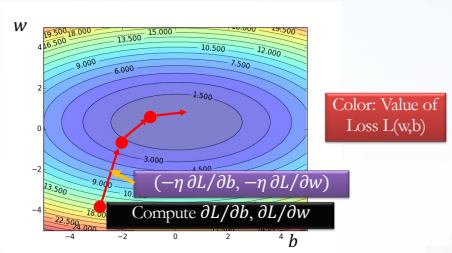
- $L(w,b) = \sum_{i=1}^{n}(y_i - f(x_i))^2$, with $y = wx + b$
- $L(w,b) = \sum_{i=1}^{n}(y_i - (b + wx_i))^2$
- Take the partial derivatives

$$\frac{\partial L}{\partial w} = \sum_{i=1}^{n} 2(y_i - (b + wx_i))(-x_i)$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n} 2(y_i - (b + wx_i))$$

59

Regression by Gradient Descent



60

Gradient Descent

- When solving:

$$\theta^* = \arg\min_{\theta} L(\theta) \quad \text{by gradient descent}$$

- Each time we update the parameters, we obtain θ that makes $L(\theta)$ smaller.

$$L(\theta^0) > L(\theta^1) > L(\theta^2) > \cdots$$

Is this statement correct?

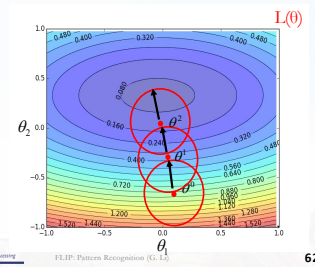
61

Formal Derivation

- Suppose that θ has two variables $\{\theta_1, \theta_2\}$

Given a point, we can easily find the point with the smallest value nearby.

Why?



62

Taylor Series

- **Taylor series**: Let h(x) be any function infinitely differentiable around x = x_0.

$$h(x) = \sum_{k=0}^{\infty} \frac{h^{(k)}(x_0)}{k!}(x - x_0)^k$$

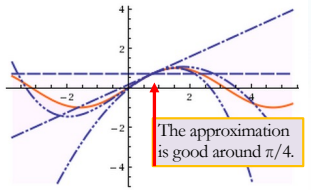
$$= h(x_0) + h'(x_0)(x - x_0) + \frac{h''(x_0)}{2!}(x - x_0)^2 + \ldots$$

When x is close to x_0 $\Rightarrow$ $h(x) \approx h(x_0) + h'(x_0)(x - x_0)$

63

Taylor Series

$$\sin(x) = \frac{1}{\sqrt{2}} + \frac{x-\frac{\pi}{4}}{\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^2}{2\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^3}{6\sqrt{2}} + \frac{\left(x-\frac{\pi}{4}\right)^4}{24\sqrt{2}} + \frac{\left(x-\frac{\pi}{4}\right)^5}{120\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^6}{720\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^7}{5040\sqrt{2}} + \frac{\left(x-\frac{\pi}{4}\right)^8}{40\,320\sqrt{2}} + \frac{\left(x-\frac{\pi}{4}\right)^9}{362\,880\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^{10}}{3\,628\,800\sqrt{2}} + \ldots$$



64

Multivariable Taylor Series

$$h(x,y) = h(x_0,y_0) + \frac{\partial h(x_0,y_0)}{\partial x}(x - x_0) + \frac{\partial h(x_0,y_0)}{\partial y}(y - y_0)$$

+ something related to $(x\text{-}x_0)^2$ and $(y\text{-}y_0)^2$ + ……

When x and y is close to x_0 and y_0

$$h(x,y) \approx h(x_0,y_0) + \frac{\partial h(x_0,y_0)}{\partial x}(x - x_0) + \frac{\partial h(x_0,y_0)}{\partial y}(y - y_0)$$

65

Back to Formal Derivation

Based on Taylor Series:

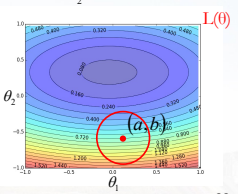
- If the red circle is ***small enough***, in the red circle

$$L(\theta) \approx L(a,b) + \frac{\partial L(a,b)}{\partial \theta_1}(\theta_1 - a) + \frac{\partial L(a,b)}{\partial \theta_2}(\theta_2 - b)$$

$$s = L(a,b)$$

$$u = \frac{\partial L(a,b)}{\partial \theta_1}, v = \frac{\partial L(a,b)}{\partial \theta_2}$$

$$L(\theta) \approx s + u(\theta_1 - a) + v(\theta_2 - b)$$



66

Back to Formal Derivation

Based on Taylor Series:

- If the red circle is ***small enough***, in the red circle

$$L(\theta) \approx s + u(\theta_1 - a) + v(\theta_2 - b)$$

constant

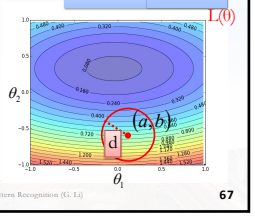
$s = L(a,b)$

$u = \frac{\partial L(a,b)}{\partial \theta_1}, v = \frac{\partial L(a,b)}{\partial \theta_2}$

Find θ_1 and θ_2 in the red circle ***minimizing*** $L(\theta)$

$$(\theta_1 - a)^2 + (\theta_2 - b)^2 \le d^2$$

Simple, right?



67

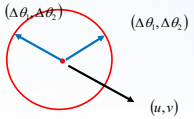
Gradient descent – two variables

Red Circle: (If the radius is small)

$$L(\theta) \approx s + u(\theta_1 - a) + v(\theta_2 - b)$$

Find θ_1 and θ_2 in the red circle ***minimizing*** $L(\theta)$

$$\frac{(\theta_1 - a)^2}{\Delta\theta_1} + \frac{(\theta_2 - b)^2}{\Delta\theta_2} \le d^2$$



To minimize $L(\theta)$

$$\begin{bmatrix}\Delta\theta_1 \\ \Delta\theta_2\end{bmatrix} = -\eta\begin{bmatrix}u \\ v\end{bmatrix} \Rightarrow \begin{bmatrix}\theta_1 \\ \theta_2\end{bmatrix} = \begin{bmatrix}a \\ b\end{bmatrix} - \eta\begin{bmatrix}u \\ v\end{bmatrix}$$

68

Back to Formal Derivation

Based on Taylor Series:

- If the red circle is ***small enough***, in the red circle

$$L(\theta) \approx s + u(\theta_1 - a) + v(\theta_2 - b)$$

$s = L(a,b)$

$u = \frac{\partial L(a,b)}{\partial \theta_1}, v = \frac{\partial L(a,b)}{\partial \theta_2}$

Find θ_1 and θ_2 yielding the smallest value of $L(\theta)$ in the circle

$$\begin{bmatrix}\theta_1 \\ \theta_2\end{bmatrix} = \begin{bmatrix}a \\ b\end{bmatrix} - \eta\begin{bmatrix}u \\ v\end{bmatrix} = \begin{bmatrix}a \\ b\end{bmatrix} - \eta\begin{bmatrix}\frac{\partial L(a,b)}{\partial \theta_1} \\ \frac{\partial L(a,b)}{\partial \theta_2}\end{bmatrix}$$

This is gradient descent.

Not satisfied if the red circle (learning rate) is not small enough

You can consider the second order term, e.g. Newton's method.

69

Newton's method

- Using second order approximation
  - Make sure $L(\theta)$ reaches the lowest point along the decreasing direction

Descent (1st derivatives): ∇L *Hessian (2nd derivatives):* H

$$L(\theta) \approx L(\theta_0) + L'(\theta_0)(\theta - \theta_0) + \frac{L''(\theta_0)}{2}(\theta - \theta_0)^2$$

optimum learning rate

- If $L(\theta)$ is quadratic, then H is constant which implies that the learning rate is constant!

70

Newton's method

- We will try to update θ to minimize $L(\theta)$

$$L(\theta) \approx L(\theta_0) + L'(\theta_0)(\theta - \theta_0) + \frac{L''(\theta_0)}{2}(\theta - \theta_0)^2$$

- Take the first derivative:

$$\frac{\partial L}{\partial \theta} = L'(\theta_0) + L''(\theta_0)(\theta - \theta_0) = 0$$

- Then we have: $(\theta - \theta_0) = -\frac{L'(\theta_0)}{L''(\theta_0)}$ hence the update rule:

$$\theta = \theta_0 - \frac{L'(\theta_0)}{L''(\theta_0)}, \text{ which is } \theta = \theta_0 - H^{-1}(\theta_0)\nabla L(\theta_0)$$

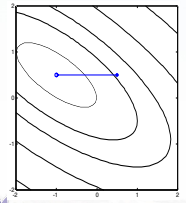
optimum learning rate

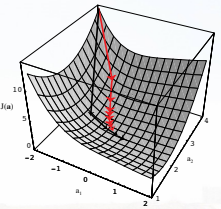
- If $L(\theta)$ is quadratic, then H is constant which implies that the learning rate is constant!

71

Newton's method

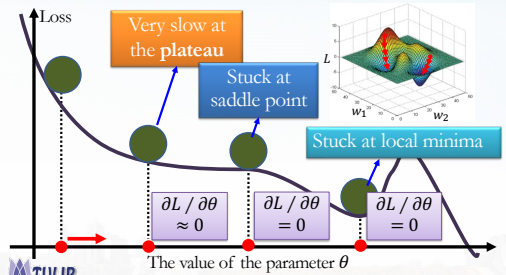
- If $L(\theta)$ is quadratic, Newton's method converges in **one step**!





72

Limitations of Gradient Descent



73

Iterative Optimization Methods

- Gradient Descent
- Newton's Method

74

Iterative Optimization

- Define a loss function $J(a)$ (based on training samples) that is minimized if a is a solution vector.
- Minimize $J(a)$ iteratively:

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\mathbf{p}_k$$

$\mathbf{a}(k) \xrightarrow{\eta(k)\mathbf{p}_k} \mathbf{a}(k+1)$

$\mathbf{p}_k$ • search direction

$\eta(k)$ • learning rate

75

Gradient Descent

- Gradient descent rule:

$$\mathbf{p}_k = -\nabla J(\mathbf{a}(k))$$

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\mathbf{p}_k$$

$\mathbf{a}(k) \xrightarrow{\eta(k)\mathbf{p}_k} \mathbf{a}(k+1)$

Algorithm 1 (Basic gradient descent)

1 **begin initialize** a, criterion θ, $\eta(\cdot)$, $k = 0$
2 **do** $k \leftarrow k + 1$
3 $\mathbf{a} \leftarrow \mathbf{a} - \eta(k)\nabla J(\mathbf{a})$
4 **until** $\eta(k)\nabla J(\mathbf{a}) < \theta$
5 **return** a
6 **end**

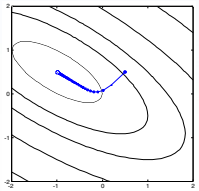
$\mathbf{p}_k$ • search direction

$\eta(k)$ • learning rate

76

Gradient Descent

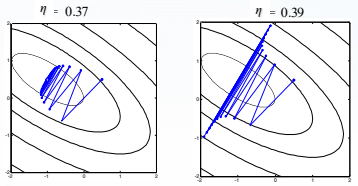
- solution space α



77

Gradient Descent

- What is the effect of the learning rate?



- slow but converges to solution
- fast by overshoots solution

78

Gradient Descent

- How to choose the learning rate $\eta(k)$?
  - Make sure $J(a(k+1))$ reaches the lowest point along the decreasing direction

Taylor series approximation

$$J(\mathbf{a}) \simeq J(\mathbf{a}(k)) + \nabla J^t(\mathbf{a} - \mathbf{a}(k)) + \frac{1}{2}(\mathbf{a} - \mathbf{a}(k))^t \mathbf{H} (\mathbf{a} - \mathbf{a}(k)),$$

Hessian (2nd derivatives)

optimum learning rate

- If $J(\mathbf{a})$ is quadratic, then H is constant which implies that the learning rate is constant!

79

Newton's method

- Newton's descent rule:

$$\mathbf{p}_k = -H^{-1}\nabla J(\mathbf{a}(k))$$

requires inverting H

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\mathbf{p}_k$$

$\mathbf{a}(k)$ —$\eta(k)\mathbf{p}_k$→ $\mathbf{a}(k+1)$

Algorithm 2 (Newton descent)

```
begin initialize a, criterion θ
    do
        a ← a − H^{-1}∇J(a)
    until H^{-1}∇J(a) < θ
    return a
end
```

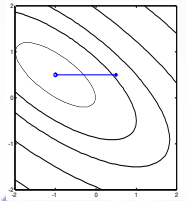
$\mathbf{p}_k$ • search direction

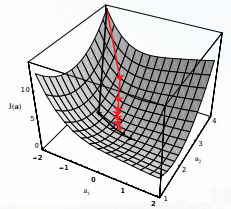
$\eta(k)$ • learning rate

80

Newton's method

- If $J(\mathbf{a})$ is quadratic, Newton's method converges in **one step**!





81

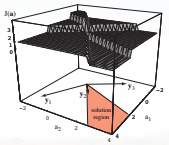
Criterion Function

- Perceptron Criterion Function
- Relaxation Procedure

82

Criterion Function

- Constructing a ***criterion function*** for solving the linear inequalities $\mathbf{a}^t\mathbf{y}_i > 0$.
  - The simplest choice is let J(a) be the number of samples misclassified by a.
    - This leads to a piecewise constant, not suitable for gradient search



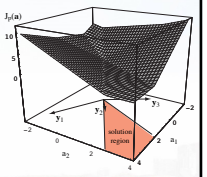
83

Perceptron Criterion Function

- Constructing a ***criterion function*** for solving the linear inequalities $\mathbf{a}^t\mathbf{y}_i > 0$.
  - **Perceptron Criterion Function**:

$$J_p(\mathbf{a}) = \sum_{\mathbf{y} \in Y(\mathbf{a})} (-\mathbf{a}^t\mathbf{y})$$

- where $Y(\mathbf{a})$ is the set of samples misclassified by $\mathbf{a}$.
- If $Y(\mathbf{a})$ is empty, $J_p(\mathbf{a})=0$; otherwise, $J_p(\mathbf{a})>0$ (normalized version)



84

Perceptron rule (Batch Correction)

- The gradient of $J_p(\mathbf{a})$ is:

$$\nabla J_p = \sum_{\mathbf{y}\in Y(\mathbf{a})} (-\mathbf{y})$$

- The perceptron update rule is obtained using gradient descent:

$$\mathbf{a}(k+1) = \mathbf{a}(k) - \eta(k)\nabla J(\mathbf{a}(k))$$

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k) \sum_{\mathbf{y}\in Y(\mathbf{a})} \mathbf{y}$$

Algorithm 3 (Batch Perceptron)

```
1 begin initialize a, η(·), criterion θ, k = 0
2     do k ← k + 1
3         a ← a + η(k) Σ_{y∈Y_k} y
4     until η(k) Σ_{y∈Y_k} y < θ
5     return a
6 end
```

• consider all examples misclassified

85

Perceptron rule (Single-Sample Correction)

- Fix the learning rate $\eta(k) = 1$
- Modify the weight vector upon each misclassified single sample
- The misclassified sample will be input infinitely often
- The perceptron update rule is obtained using gradient descent:

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k) y$$

86

Perceptron rule (Single-Sample Correction)

- **Perceptron Convergence Theorem:**
  - If training samples are linearly separable, then the sequence of weight vectors will terminate at a solution vector in a finite number of steps.

Algorithm 4 (Fixed-increment single-sample Perceptron)

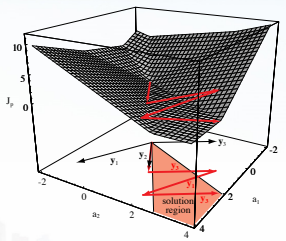
```
1 begin initialize a, k = 0
2     do k ← (k + 1) mod n
3         if y_k is misclassified by a then a ← a + y_k
4     until all patterns properly classified
5     return a
6 end
```

η(k)=1

• consider one example at a time

87

Perceptron rule



88

Perceptron rule (Margins)

- Generalizations:
  - Variable learning rate and a margin

Algorithm 5 (Variable increment Perceptron with margin)

```
1 begin initialize a, criterion θ, margin b, η(·), k = 0
2     do k ← k + 1
3         if a^t y_k + b < 0 then a ← a + η(k) y_k
4     until a^t y_k + b ≤ 0 for all k
5     return a
6 end
```


89

Relaxation Procedures

- Note that different criterion functions exist
- One possible choice is:

$$J_q(a) = \sum_{y\in Y} (a^t y)^2$$

  - Where Y is again the set of the training samples that are misclassified by **a**
  - However, there are two problems with this criterion
    - The function is too smooth and can converge to a=0
    - Jq is dominated by training samples with large magnitude

90

Relaxation Procedures

- A modified version that avoids two problems is

$$J_r(a) = \frac{1}{2}\sum_{y\in Y}\frac{(a^t y - b)^2}{\|y\|^2}$$

  - Here Y is the set of samples for which its gradient is given by:

$$\nabla J_r = \sum_{y\in Y}\frac{(a^t y - b)}{\|y\|^2} y$$

91

Relaxation Procedures

Algorithm 8 (Batch relaxation with margin)

```
begin initialize a, η(·), k = 0
    do k ← k + 1
        Y_k = {}
        j = 0
        do j ← j + 1
            if y_j is misclassified then Append y_j to Y_k
        until j = n
        a ← a + η(k) Σ_{y∈Y_k} (b − a^t y)/‖y‖² y
    until Y_k = {}
return a
end
```


92

Relaxation Procedures

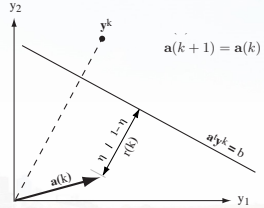
Algorithm 9 (Single-sample relaxation with margin)

```
begin initialize a, η(·), k = 0
    do k ← k + 1
        if y_k is misclassified then a ← a + η(k) (b − a^t y_k)/‖y_k‖² y_k
        until all patterns properly classified
    return a
end
```


93

Relaxation Procedures

- Since y/|y| is the unit normal vector for the hyperplane. The correction calls for a(k) to be moved a certain fraction of η of the distance from a(k) to the hyperplane

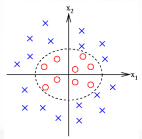


$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta\frac{b - \mathbf{a}^t(k)\mathbf{y}^k}{\|\mathbf{y}^k\|^2}\mathbf{y}^k$$

94

Nonseparable Behavior

- Perceptron and relaxation procedures are all error-correction procedure.
  - Their success on separable problems is largely due to its relentless search for an error-free solution.
  - For non-separable problems, correction may never converge.



95

MSE Procedure

- MSE and Discrimants
- Pseudoinverse

96

Minimum Squared Error Procedures

- Define a criterion function for all samples, rather than the misclassified samples
- Try to find the vector such that $\mathbf{a}^t\mathbf{y}_i > \mathbf{b}_i$
  - Where $\mathbf{b}_i$ is any positive value

$$\begin{pmatrix} Y_{10} & Y_{11} & \cdots & Y_{1d} \\ Y_{20} & Y_{21} & \cdots & Y_{2d} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ Y_{n0} & Y_{n1} & \cdots & Y_{nd} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

97

Minimum Squared Error Procedures

- Minimum squared error and pseudoinverse
  - The problem is to find a weight vector $\mathbf{a}$ satisfying $\mathbf{Ya}=\mathbf{b}$
  - If we have more equations than unknowns, $\mathbf{a}$ is over-determined.
  - We want to choose the one that minimizes the sum-of-squared-error criterion function

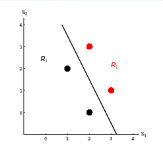
$$J_s(\mathbf{a}) = \|\mathbf{Ya} - \mathbf{b}\|^2 = \sum_{i=1}^{n} (\mathbf{a}^t\mathbf{y}_i - b_i)^2.$$

98

Minimum Squared Error Procedures

- Pseudoinverse

$$\begin{aligned} \mathbf{a} &= (\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t\mathbf{b} \\ &= \mathbf{Y}^\dagger\mathbf{b}, \end{aligned}$$

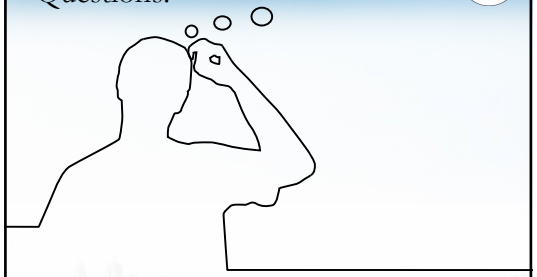


Algorithm 10 (LMS)

```
1 begin initialize a, b, criterion θ, η(·), k = 0
2       do k ← k + 1
3          a ← a + η(k)(b_k − a^t y^k)y^k
4    until η(k)(b_k − a^t y^k)y^k < θ
5    return a
6 end
```


99

Questions?



101