Agenda

✧ Linear Discriminant Functions (LDF)

✧ Multi-class problems
  ✧ Linear machine
  ✧ Completely Linearly Separation
  ✧ Pairwise Linearly Separation

✧ Linear Discriminant Function Design
  ✧ Least Mean Squared Error Method
  ✧ Sum of Squared Error Method
  ✧ Ho-Kashyap Method
  ✧ Probabilistic Methods
Linear Discriminant Functions (LDF)

- **Definition:**
  - LDF is a function that is a linear combination of the components of $x$
    
    $$g(x) = w^T x + w_0$$

  - where $w$ is the weight vector and $w_0$ the bias, or threshold weight.

- **A two-category classifier with a discriminant function of the above form uses the following rule:**
  - Decide $w_1$ if $g(x) > 0$ and $w_2$ if $g(x) < 0$
  - Decide $w_1$ if $w^T x > -w_0$ and $w_2$ otherwise
  - The value $g(x)$ of the function for a certain point $x$
    - is called functional margin
  - If $g(x) = 0$ then $x$ can be assigned to either class
    - The equation $g(x) = 0$ defines the decision surface that
      - separates points assigned to the category $w_1$ from points assigned to the category $w_2$
    - When $g(x)$ is linear, the decision surface is a hyperplane.
Linear Discriminant Functions

- In conclusion, a linear discriminant function divides the feature space by a hyperplane decision surface.
- Decision boundary $g(x)=0$ corresponds to $(d-1)$-dimensional hyperplane in $d$-dimensional $x$-space.
- The orientation of the surface is determined by the normal vector $w$ and the location of the surface is determined by the bias.
  - We can view Fisher method (LDA) as a linear discriminant function, too.
Multi-class problems

✧ Suppose we have an n-classes classification problem, and we want to separate them with linear discriminant functions
  ✧ Do you have any idea about how to use discriminant function in this case
    ✧ We have many ways to do this.

✧ Using linear discriminant function in multi-class problems
  ✧ Linear machines (one versus one)
  ✧ Completely linearly separation (one versus the rest)
  ✧ Pairwise linearly separation

✧ We introduce above methods through illustrative examples in next slides.
Case 1: Linear Machine

✧ Suppose a 3-class classification problem with the following discriminant functions:

\[ g_1(x) = -x_1 + x_2 \]
\[ g_2(x) = x_1 + x_2 - 1 \]
\[ g_3(x) = -x_2 \]

and use the following rule for classification (linear machine rule):

\[ x \in C_i \iff g_i(x) > g_j(x) ; \forall j \neq i \]

How do these classes partition space?
Case 1: Linear Machine

- Each class partition can be obtained through solving two equations.
- The result:
More on Linear Machines

✧ In some texts, it is called one versus one (one against one).

✧ How many functions we need for n classes? (n)

✧ The decision regions for linear machine are convex and this restriction limits the flexibility of the classifier.
Case 2: Completely Linearly Separation

Suppose a 3-class classification problem with the following discriminant functions:

\[ g_1(x) = -x_1 + x_2 \]
\[ g_2(x) = x_1 + x_2 - 5 \]
\[ g_3(x) = -x_2 + 1 \]

and use the following rule for classification (completely linearly separation rule):

\[ \text{if } g_i(x) > 0 \Rightarrow x \in C_i \text{ and if } g_i(x) < 0 \Rightarrow x \notin C_i \]

How these classes partition the space? Determine the undecided sub-spaces.
Case 2: Completely Linearly Separation

✧ Each class partition can be obtained through solving three equations.
✧ The result:
More on Completely Linearly Separation

✧ In some texts, it is called one versus the rest (one against all).
✧ If we have n classes, what is the number of needed functions? (n)
✧ Are the decision regions convex?
✧ Compare the undecided sub-spaces in two cases.
Case 3: Pairwise Linearly Separation

Suppose a 3-class classification problem with the following discriminant functions:

\[
g_{12}(x) = -x_1 - x_2 + 5 \\
g_{13}(x) = -x_1 + 3 \\
g_{23}(x) = -x_1 + x_2 \quad \quad g_{ij}(x) = -g_{ji}(x)
\]

and use the following rule for classification:

\[
x \in C_i \iff \forall j \neq i \ g_{ij}(x) > 0
\]

How these classes partition the space? Determine the undecided sub-spaces.
Case 3: Pairwise Linearly Separation

✧ Each class partition can be obtained through solving two equation.
✧ The result:
More on Pairwise Linearly Separation

✧ If we have \( n \) classes, what is the number of needed functions? \( (c(n,2)) \)

✧ Are the decision regions convex?

\[
\forall j \neq i \quad g_i(y) \geq g_j(y) \text{ and } g_i(z) \geq g_j(z) \iff g_i(\alpha y + (1-\alpha)z) \geq g_j(\alpha y + (1-\alpha)z)
\]

✧ Definition: A region \( R_i \) is convex iff \( \forall y, z \in R_i \Rightarrow \alpha y + (1-\alpha)z \in R_i \)
Linear Discriminant Functions

✧ Main problem
  ✧ How to create the discriminant functions for each class (how obtain w)?

✧ Many methods exist for this purpose, such as:

✧ Error Minimization Methods
  ✧ Least Mean Squared Error Method → will be discussed in next slides
  ✧ Sum of Squared Error Method → will be discussed in next slides
  ✧ Ho-Kashyap Method → will be discussed in next slides

✧ Fisher Linear Discriminant Method → discussed in lecture 3

✧ Perceptorn Method → will be discussed in lecture 9

✧ Probabilistic Methods → discussed in lecture 6

✧ etc.
Augmented Space

- Consider a linear discriminant function \( g(x) \) in feature space \( x \)
  \[
g(x) = w^T x + w_0
\]

- We can use augmented space to get rid of \( w_0 \) in new augmented feature space \( y \) which map each data point \( x \) in to new augmented space according:
  \[
y = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} = \begin{bmatrix} 1 \\ x \end{bmatrix}
\]

- So the new discriminant function will be:
  \[
g(y) = a^T y ; \quad a = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} = \begin{bmatrix} w_0 \\ w \end{bmatrix}
\]
Inverted Space

✧ In classification with discriminant functions in augmented space we seek for a line so that:

\[ a' y > 0 \quad \text{for positive class} \]
\[ a' y < 0 \quad \text{for negative class} \]

✧ If we invert the training data of negative class then for all data point we must have

\[ a' y > 0 \]

✧ So we can easily look for \( a' \), given training data \( y \) which is in the form of:

\[
y = \begin{bmatrix}
1 & x_{11} & \cdots & x_{1d} \\
1 & x_{21} & \cdots & x_{2d} \\
\vdots & \vdots & \ddots & \vdots \\
-1 & x_{n-11} & \cdots & x_{n-1d} \\
-1 & x_{n1} & \cdots & x_{nd}
\end{bmatrix}
\]
Least Mean Squared Error

- We want to choose the $W$ that minimizes the mean-squared-error criterion function:

$$J(w) = E\left[|y - g(x)|^2\right] = E[(y - w^t x)^2]$$

$$\hat{w} = \text{arg min}_w J(w)$$

$$\frac{\partial J(w)}{\partial w} = 2E[x(y - w^t x)] = 2E[(xy - xw^t x)] = 2E[(xy - xx^t w)] = 2E xy - E[xx^t] w = 0$$

$$w = E[xx^t]^{-1} E xy = R_x^{-1} E xy$$

$$R_x = E[xx^t] = \begin{bmatrix} E[x_1x_1] & \ldots & E[x_1x_n] \\ E[x_2x_1] & \ldots & E[x_2x_n] \\ \vdots & \vdots & \vdots \\ E[x_nx_1] & \ldots & E[x_nx_n] \end{bmatrix}, \quad E[xy] = \begin{bmatrix} x_1y \\ x_2y \\ \vdots \\ x_ny \end{bmatrix}$$

- We can also use the gradient descent rule for updating $w$ instead of analytical solving.
Sum of Squared Error

✧ SSE uses the sum of squared error as objective function

✧ Also known as Pseudo inverse matrix method

\[ J(w) = \|Xw - b\|^2 \]
\[
\frac{\partial}{\partial w} J(w) = 2w^tX^tX - 2b^tX = 0
\]
\[ w = \frac{(X^tX)^{-1}X^tb}{X^t b} \]
**Sum of Squared Error**

✧ **Example**

✧ Find the SSE boundary for the given data points, $c_1 : [(1,2),(2,0)]$ and $c_2 : [(3,1),(2,3)]$

$$w^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$x = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 0 \\ -1 & -3 & -1 \\ -1 & -2 & -3 \end{bmatrix}$

$\Rightarrow X^T = (X'X)^{-1}X' = \begin{bmatrix} 5/4 & 13/12 & 3/4 & 7/12 \\ -1/2 & -1/6 & -1/2 & -1/6 \\ -1 & -1/3 & 0 & -1/3 \end{bmatrix}$

assuming $y = [1 \ 1 \ 1]^T \Rightarrow w = X^T y = [11/3 \ -4/3 \ -2/3]$

$\Rightarrow g(x) = \frac{11}{3} - \frac{4}{3}x_1 - \frac{2}{3}x_2$
Ho-Kashyap Method

✧ The main limitation of the SSE is lack of guarantees that a separating hyperplane will be found in the linearly separable case
  ✧ The SSE rule tries to minimize \( \|w'x - b\|^2 \)
  ✧ Finding a separating hyperplane depends on how suitably the outputs \( b \) are selected
✧ If the two classes are linearly separable, there must exist vectors \( w \) and \( b \) such that
  \( w^T x = b > 0 \)
  ✧ if \( b \) were known, to compute the separating hyperplane, the SSE solution will be \( w = x \cdot b \)
  ✧ Nevertheless, since \( b \) is unknown, one must solve the equation for both \( w \) and \( b \)
✧ A possible algorithm is the Ho-Kashyap procedure:
  1. Find the target values \( b \) with gradient descent
  2. Compute the weight vector \( w \) from the SSE solution
  3. Repeat 1 and 2 until convergence
Ho-Kashyap Method

✧ g(x) > 0 can be rewrite as g(x)=b; b>0
  ✧ How we can determine b?

✧ Objective function in this case is \( J(w,b) = \| w^t x - b \|^2 \)

✧ Ho-Kashyap method offers an iterative method for obtaining w and b, using following steps:
  ✧ Keep b constant and optimize J relative to w (using obtained b from last step)
    ✧ Using previous method we have:
      \[ w(t+1) = x^{-1} b(t) \]

  ✧ Keep w constant and optimize J relative to b (using obtained w from last step)
    ✧ The objective is to minimize
      \[ \frac{\partial J}{\partial w} = -2 w^t x - b \]
    ✧ Using Gradient descent method we have:
      \[ b(t+1) = b(t) + \eta \| 2 w^t(t) x - b \| \]
    ✧ To hold the constraint b>0, we set (xw-b) in this rule to zero if it becomes negative, then the rule will be:
      \[ b(t+1) = b(t) + \eta \left( w^t(t) x - b + \left| w^t(t) x - b \right| \right) \]
Probabilistic Methods

✧ Maximum likelihood
  ✧ $g_i(x) = P(x|w_i)$

✧ Bayesian Classifier
  ✧ $g_i(x) = P(w_i|x)$
    ✧ $g_i(x) = P(x|w_i) P(w_i)$
    ✧ $g_i(x) = \ln P(x|w_i) + \ln P(w_i)$

✧ Expected Loss (Conditional Risk)
  ✧ $g_i(x) = -R(a_i|x)$
Any Question

End of Lecture 7

Thank you!

Spring 2013

http://ce.sharif.edu/courses/91-92/2/ce725-1/