From Nearest to Linear Discriminant Functions

Biometrics
CSE 190
Lecture 10

Announcements

• HW2 due today
• Literature review due next Tuesday.
• Most of last two lecture was on the blackboard.
• Extra class Tomorrow, 2:00-3:00, CSE4140
• No class next Thursday, 5/15

Non-Parametric Density Estimation

• Given a collection of $n$ samples, estimate the probability density.
  - Parzen Windows
  - K-th nearest neighbor

• Main ideas:
  1. As number of samples ($n$) approaches infinity, estimated density should approach true density
  2. Approximated density should be "reasonable" for finite $n$.

K-th Nearest Neighbor Density Estimation

![K-th Nearest Neighbor Density Estimation](image)

---

Pattern Classification, Ch4 (Part 1)

– The $k$ – nearest-neighbor rule

• Goal: Classify $x$ by assigning it the label most frequently represented among the $k$ nearest samples and use a voting scheme
The nearest-neighbor rule

- Let \( D_n = \{x_1, x_2, \ldots, x_n\} \) be a set of \( n \) labeled prototypes
- Let \( x' \in D_n \) be the closest prototype to a test point \( x \) then the nearest-neighbor rule for classifying \( x \) is to assign it the label associated with \( x' \)
- The nearest-neighbor rule leads to an error rate greater than the minimum possible: the Bayes rate
- If the number of prototype is large (unlimited), the error rate of the nearest-neighbor classifier is never worse than twice the Bayes rate (it can be demonstrated!
- If \( n \to \infty \), it is always possible to find \( x' \) sufficiently close so that:
  \[ P(\omega_i | x') = P(\omega_i | x) \]

Whitening Transform

See blackboard

Linear Discriminant Functions (Sections 5.1-5.2)

- Definition
  A linear discriminant function is a linear combination of the components of \( x \)
  \[ g(x) = w^T x + w_0 \quad (1) \]
  where \( w \) is the weight vector and \( w_0 \) the bias
- A two-category classifier with a discriminant function of the form (1) uses the following rule:
  Decide \( \omega_1 \) if \( g(x) > 0 \) and \( \omega_2 \) if \( g(x) < 0 \)
  \[ \leftrightarrow \text{ Decide } \omega_1 \text{ if } w^T x > -w_0 \text{ and } \omega_2 \text{ otherwise} \]
  If \( g(x) = 0 \) \( \Rightarrow x \) is assigned to either class
• The equation \( g(x) = 0 \) defines the decision surface that separates points assigned to the category \( \omega_1 \) from points assigned to the category \( \omega_2 \).

• When \( g(x) \) is linear, the decision surface is a hyperplane.

• Algebraic measure of the distance from \( x \) to the hyperplane.

The multi-category case

• We define \( c \) linear discriminant functions

\[
g_i(x) = w_i^T x + w_{i0} \quad i = 1, \ldots, c
\]

and assign \( x \) to \( \omega_i \) if \( g_i(x) > g_j(x) \) \( \forall j \neq i \); in case of ties, the classification is undefined.

• In this case, the classifier is a "linear machine".

• A linear machine divides the feature space into \( c \) decision regions, with \( g_i(x) \) being the largest discriminant if \( x \) is in the region \( R_i \).

• For a two contiguous regions \( R_i \) and \( R_j \), the boundary that separates them is a portion of hyperplane \( H_{ij} \) defined by:

\[
(g_i(x) = g_j(x)) \iff (w_i - w_j)^T x + (w_{i0} - w_{j0}) = 0
\]

• \( w_i - w_j \) is normal to \( H_{ij} \) and

\[
d(x, H_{ij}) = \frac{g_i - g_j}{\|w_i - w_j\|}
\]

It is easy to show that the decision regions for a linear machine are convex, this restriction limits the flexibility and accuracy of the classifier.

Perceptron

Linear, threshold units

\[
\phi(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_1 x_1 + \cdots + w_n x_n > \theta \\ -1 & \text{otherwise.} \end{cases}
\]
The threshold can be easily forced to 0 by introducing an additional weight input $W_0 = \theta$.

$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1x_1 + \cdots + w_nx_n > 0 \\ -1 & \text{otherwise} \end{cases}$$

How powerful is a perceptron?

Threshold = 0

Inverter

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Boolean AND

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Boolean OR

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Boolean XOR

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Concept Space & Linear Separability

Linear Separability

$w_1 x_1 + w_2 x_2 \geq \theta$ for positive $o$ (AND)

$w_1 x_1 + w_2 x_2 \leq \theta$ for negative $o$ (XOR)

Training Perceptron

Perceptron Training Rule

Given training data of pairs $<\text{input features, output}>$

$$w'_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \eta (t - o) x_i$$

Where:

- $t$ is target value
- $o$ is perceptron output
- $\Delta$ is small constant (e.g., $\Delta = 0.1$)
- $\eta$ is learning rate

Converges if...

- training data linearly separable
- step size $\eta$ sufficiently small
- no "hidden" units

Gradient Descent

Learn $w_i$’s that minimize squared error

$$E[\overline{w}] = \frac{1}{2} \sum_{j=0}^{n} (t_j - o_j)^2$$

$T_j$: Training label

$O_j$: Output of net with training input
**Gradient Descent**
- To find the best direction in the feature space, we compute the gradient of $E$ with respect to each of the components of $\hat{w}$:
  \[ \nabla E(\hat{w}) = \left[ \frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_n} \right] \]
- This vector specifies the direction that produces the steepest increase in $E$.
- We want to modify $\hat{w}$ in the direction of $-\nabla E(\hat{w})$.
- Where:
  \[ \Delta \hat{w} = -R \nabla E(\hat{w}) \]

**Batch Learning**
- Initialize each $w_i$ to a small random value.
- Repeat until termination:
  \[ \Delta w_i = 0 \]
- For each training example $d$ do:
  \[ o_d = \sigma(\sum_i w_i x_{i,d}) \]
  \[ \Delta w_i \leftarrow \Delta w_i + \eta (y_d - o_d) o_d (1-o_d) x_{i,d} \]
  \[ w_i \leftarrow w_i + \Delta w_i \]

**Increasing Expressiveness: Multi-Layer Neural Networks**

### Boolean XOR

<table>
<thead>
<tr>
<th>Input 1</th>
<th>Input 2</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

#### 2-layer Neural Net

**Multi-Layer Neural Network**
- Multi-layer networks can represent arbitrary functions, but building effective learning methods for such networks was thought to be difficult.
- Networks are composed of an input layer, hidden layer(s), and output layer. Activation is feed-forward from input to output.

**Basic Unit in Multi-Layer Neural Network**
- **Linear Unit**: $o_j = \hat{w} \cdot \hat{x}$ multiple layers of linear functions produce linear functions. We want to represent nonlinear functions.
- **Threshold units**: $o_j = \text{sgn}(\hat{w} \cdot \hat{x} - T)$ are not differentiable, hence unsuitable for gradient descent.
- Use a non-linear, differentiable output function such as the sigmoid (or logistic) function:
  \[ o_j = \frac{1}{1 + e^{-(w \cdot x - T)}} \]
Model Neuron (Logistic)

- Use a non-linear, differentiable output function such as the sigmoid or logistic function.

\[ y_j = \frac{1}{1 + e^{-(\text{net}_j - T_j)}} \]

- Net input to a unit is defined as: \( \text{net}_j = \sum w_k x_i \)

- Output of a unit is defined as: \( O_j = \frac{1}{1 + e^{-(\text{net}_j - T_j)}} \)