Today’s Schedule

1. Discriminant functions (recap)
2. Decision boundary of linear discriminants (recap)
3. Discriminative training of linear discriminants (Perceptron)
4. Structures and decision boundaries of Perceptron
5. LSE Training of linear discriminants
6. Appendix - calculus, gradient descent, linear regression

Discriminant functions (recap)

\[ y_k(x) = \ln \left( \frac{P(x|C_k)P(C_k)}{P(x|C_i)P(C_i)} \right) \]

\[ = -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) - \frac{1}{2} \ln |\Sigma_k| + \ln P(C_k) \]

\[ = \frac{1}{2}x^T \Sigma^{-1}x + \mu_k^T \Sigma^{-1}x - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k - \frac{1}{2} \ln |\Sigma_k| + \ln P(C_k) \]

Decision boundary of linear discriminants

\[ y(x) = w_1x_1 + w_2x_2 + w_0 = 0 \]

Decision boundary of linear discriminant (2D)

\[ y(x) = w_1x_1 + w_2x_2 + w_3x_3 + w_0 = 0 \]

Approach to linear discriminant functions

Generative models: \( p(x|C_k) \)

Discriminant function based on Bayes decision rule

\[ y_k(x) = \ln p(x|C_k) + \ln P(C_k) \]

↓ Gaussian pdf (model)

\[ y_k(x) = -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) - \frac{1}{2} \ln |\Sigma_k| + \ln P(C_k) \]

↓ Equal covariance assumption

\[ y_k(x) = w_k^T x + w_0 \]

↑ Why not estimating the decision boundary or \( P(C_k|x) \) directly?

Discriminative training / models

(Logistic regression, Perceptron / Neural network, SVM)

Training linear discriminant functions directly

A discriminant for a two-class problem:

\[ y(x) = y_1(x) - y_2(x) = (w_1 - w_2)^T x + (w_{01} - w_{02}) \]

\[ = w^T x + w_0 \]

\[ w = (w_1, w_2) \]

slope = \( w_1/w_2 \)

slope = -\( w_1/w_2 \)
**Perceptron error correction algorithm**

\[ a(x) = w^T x + w_0 = w^T \hat{x} \]

where \( \hat{x} = (x_0, w^T)^T, \hat{x} = (1, x)^T \)

Let's just use \( w \) and \( \hat{x} \) to denote \( w \) and \( \hat{x} \) from now on!

\[ y(x) = g(a(x)) = g(w^T \hat{x}) \quad \text{where} \quad g(a) = \begin{cases} 1, & \text{if } a \geq 0, \\ 0, & \text{if } a < 0 \end{cases} \]

\( g(a) \): activation / transfer function

- Training set: \( D = \{(x_1, t_1), \ldots, (x_N, t_N)\} \)
  where \( t_i \in \{0, 1\} : \text{ target value} \)

- Modify \( w \) if \( x_i \) was misclassified

\[ w^{(\text{new})} = w + \eta (t_i - y(x_i)) x_i \quad (0 < \eta < 1) \]

**Geometry of Perceptron’s error correction**

**The Perceptron learning algorithm**

Incremental (online) Perceptron algorithm:

for \( i = 1, \ldots, N \)

\[ w \leftarrow w + \eta (t_i - y(x_i)) x_i \]

Batch Perceptron algorithm:

\[ v_{\text{sum}} = 0 \]

for \( i = 1, \ldots, N \)

\[ v_{\text{sum}} = v_{\text{sum}} + (t_i - y(x_i)) x_i \]

\[ w \leftarrow w + \frac{v_{\text{sum}}}{N} \]

**Linearly separable vs linearly non-separable**

What about convergence?

The Perceptron learning algorithm terminates if training samples are linearly separable.

**Background of Perceptron**

- 1940s: Warren McCulloch and Walter Pitts: 'threshold logic'
- Donald Hebb: 'Hebbian learning'
- 1957: Frank Rosenblatt: 'Perceptron'

**Character recognition by Perceptron**

\[ y(x) = g(a(x)) = g(w^T x) \]

where \( g(a) = \begin{cases} 1, & \text{if } a \geq 0, \\ 0, & \text{if } a < 0 \end{cases} \)

**Perceptron structures and decision boundaries**

NB: A one node/neuron constructs a decision boundary, which splits the input space into two regions.
Perceptron as a logical function

<table>
<thead>
<tr>
<th>NOT</th>
<th>OR</th>
<th>NAND</th>
<th>XOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>x₀</td>
<td>x₁</td>
<td>y₀</td>
<td>y₁</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0 0</td>
<td>1 0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0 1</td>
<td>0 0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0 1</td>
<td>0 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1 0</td>
<td>1 0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1 1</td>
<td>1 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1 1</td>
<td>1 0</td>
</tr>
</tbody>
</table>

Perceptron structures and decision boundaries (cont.)

Question: find the weights for each function

0 2 1 xxx
ΣΣ ΣΣ Σ
Σ
Σ
Σ

Problems with the Perceptron learning algorithm

- No training algorithms for multi-layer Perceptron
- Non-convergence for linearly non-separable data
- Weights \( \mathbf{w} \) are adjusted for misclassified data only (correctly classified data are not considered at all)

\[ E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} (\mathbf{w}^T \mathbf{x}_n - t_n)^2 \]

Training with least squares

Derivatives of functions of one variable

\[ \frac{df}{dx} = f'(x) = \lim_{\epsilon \to 0} \frac{f(x + \epsilon) - f(x)}{\epsilon} \]

e.g., \( f(x) = 4x^3, f'(x) = 12x^2 \)

Partial derivatives of functions of more than one variable

\[ \frac{df}{dx} = \lim_{\epsilon \to 0} \frac{f(x + \epsilon, y) - f(x, y)}{\epsilon} \]

e.g., \( f(x, y) = y^3x^2, \frac{df}{dx} = 2y^3x \)
Consider $f(x)$, where $x = (x_1, \ldots, x_D)^T$.

**Notation:** all partial derivatives put in a vector:

$$\nabla_x f(x) = \left( \frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_D} \right)^T$$

**Example:** $f(x) = x_1^2 x_2^2$

$$\nabla_x f(x) = \begin{pmatrix} 3x_1^2 x_2^2 \\ 2x_1 x_2^2 \end{pmatrix}$$

**Fact:** $f(x)$ changes most quickly in direction $\nabla_x f(x)$

---

**Linear regression (one variable) least squares line fitting**

- **Training set:** $D = \{(x_n, t_n)\}_{n=1}^N$
- **Linear regression:** $t_n = ax_n + b$
- **Objective function:** $E = \sum_{n=1}^N (t_n - (ax_n + b))^2$
- **Optimisation problem:** $\min_{a,b} E$

**Partial derivatives:**

$$\frac{\partial E}{\partial a} = 2 \sum_{n=1}^N (t_n - (ax_n + b))(-x_n)$$

$$= 2a \sum_{n=1}^N x_n^2 + 2b \sum_{n=1}^N x_n - 2 \sum_{n=1}^N t_n x_n$$

$$\frac{\partial E}{\partial b} = -2 \sum_{n=1}^N (t_n - (ax_n + b))$$

$$= 2a \sum_{n=1}^N x_n + 2b \sum_{n=1}^N 1 - 2 \sum_{n=1}^N t_n$$

---

**Linear regression (multiple variables)**

- **Training set:** $D = \{(x_n, t_n)\}_{n=1}^N$, where $x_n = (1, x_1, \ldots, x_D)^T$
- **Linear regression:** $t_n = x^T_n w$
- **Objective function:** $E = \sum_{n=1}^N (t_n - x^T_n w)^2$
- **Optimisation problem:** $\min_{a,b} E$

---

**Summary**

- Training discriminant functions directly (discriminative training)
- Perceptron training algorithm
  - Perceptron error correction algorithm
  - Least squares error + gradient descent algorithm
- Linearly separable vs linearly non-separable
- Perceptron structures and decision boundaries
- See Notes 11 for a Perceptron with multiple output nodes