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)

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

Decision boundary of linear discriminants

\[ y(x) = w^T x + w_0 = 0 \]

Dimension | Decision boundary
---|---
2 | line \( w_1x_1 + w_2x_2 + w_0 = 0 \)
3 | plane \( w_1x_1 + w_2x_2 + w_3x_3 + w_0 = 0 \)
\( D \) | hyperplane \( \sum_{i=1}^{D} w_ix_i + w_0 = 0 \)

NB: \( w \) is a normal vector to the hyperplane

Decision boundary of linear discriminant (2D)

\[ y(x) = w_1x_1 + w_2x_2 + w_0 = 0 \quad (x_2 = -\frac{w_1}{w_2}x_1 - \frac{w_0}{w_2} \text{ when } w_2 \neq 0) \]

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)
- Equal covariance assumption
- 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:

\[ y(x) = y_1(x) - y_2(x) = (w_1 - w_2)^T x + (w_{10} - w_{20}) \]

\[ = w^T x + w_0 \]

Linear discriminants for a 2-class problem

\[ y_1(x) = w_1^T x + w_{10} \]
\[ y_2(x) = w_2^T x + w_{20} \]

Combined discriminant function:

\[ y(x) = y_1(x) - y_2(x) = (w_1 - w_2)^T x + (w_{10} - w_{20}) = w^T x + w_0 \]

Decision:

\[ C = \begin{cases} 1, & \text{if } y(x) \geq 0, \\ 2, & \text{if } y(x) < 0 \end{cases} \]

Decision boundary of linear discriminant (3D)

\[ y(x) = w_1x_1 + w_2x_2 + w_3x_3 + w_0 = 0 \]
Perceptron error correction algorithm

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

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

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

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

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

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

- Training set: \( D = \{ (x_1, t_1), \ldots, (x_N, t_N) \} \)

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

- \( w^{(\text{new})} \leftarrow w + \eta (t_i - y(x_i)) x_i \) \((0 < \eta < 1)\)

- Learning rate \( \eta \)

\[ (w^{(\text{new})})^T x_i = w^T x_i + \eta (t_i - y(x_i)) \|x_i\|^2 \]

**Geometry of Perceptron’s error correction**

\[ y(x_i) = g(w^T x_i) \]

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

\[ t_i - y(x_i) \]

\[ y(x_i) = 0 \quad 1 \]

\[ t_i = 0 \quad 1 \quad 0 \]

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

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

\[ w \leftarrow w + \eta V_{\text{sum}} \]

**Linearly separable vs linearly non-separable**

(a-1) Linearly separable

(a-2) 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'

1957 Frank Rosenblatt: 'Perceptron'

**Character recognition by Perceptron**

**Perceptron structures and decision boundaries**

\[ y(x) = g(a(x)) \]

\[ g(a) = \begin{cases} 1, & a \geq 0 \\ 0, & a < 0 \end{cases} \]

\[ w = (w_0, w_1, \ldots, w_D)^T \]

\[ x = (1, x_1, \ldots, x_D)^T \]

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

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

- **NOT**
  - \( x \rightarrow y \)
  - \( 0 \rightarrow 1 \)
  - \( 1 \rightarrow 0 \)

- **OR**
  - \( x \lor y \)
  - \( 0 \lor 0 \rightarrow 0 \)
  - \( 0 \lor 1 \rightarrow 1 \)
  - \( 1 \lor 0 \rightarrow 1 \)
  - \( 1 \lor 1 \rightarrow 1 \)

- **NAND**
  - \( \neg (x \land y) \)
  - \( \neg (0 \land 0) \rightarrow 1 \)
  - \( \neg (0 \land 1) \rightarrow 1 \)
  - \( \neg (1 \land 0) \rightarrow 1 \)
  - \( \neg (1 \land 1) \rightarrow 1 \)

- **XOR**
  - \( x \oplus y \)
  - \( 0 \rightarrow 1 \)
  - \( 1 \rightarrow 0 \)

**Question:** find the weights for each function

---

**Perceptron structures and decision boundaries (cont.)**

- **Training with least squares**
  - Squared error function:
    \[ E(w) = \frac{1}{2} \sum_{n=1}^{N} (w^T x_n - t_n)^2 \]

  - Optimisation problem:
    \[ \min_w E(w) \]

  - One way to solve this is to apply gradient descent (steepest descent):
    \[ w \leftarrow w - \eta \nabla_w E(w) \]

    where \( \eta \): step size (a small positive constant)

    \[ \nabla_w E(w) = \left( \frac{\partial E}{\partial w_0}, \ldots, \frac{\partial E}{\partial w_D} \right)^T \]

  - Partial derivatives of functions of more than one variable
    \[ \frac{df}{dx} = \lim_{\epsilon \to 0} \frac{f(x + \epsilon) - f(x)}{\epsilon} \]

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

    - 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., \( f(x,y) = y^3x^2 \), \( \frac{df}{dx} = 2y^3x \)

- **Appendix – derivatives**
  - Derivatives of functions of one variable
    \[ \frac{df}{dx} = \lim_{\epsilon \to 0} \frac{f(x + \epsilon) - f(x)}{\epsilon} \]

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

- **Problems with the Perceptron learning algorithm**
  - No training algorithms for multi-layer Perceptron
  - Non-convergence for locally non-separable data
  - Weights \( w \) are adjusted for misclassified data only (correctly classified data are not considered at all)

  \[ \Rightarrow \]

  - Consider not only mis-classification (on train data), but also the optimality of decision boundary
    - Least squares error training
    - Large margin classifiers (e.g. SVM)

---

**Training with least squares (cont.)**

- Trainable in linearly non-separable case
  - Not robust (sensitive) against erroneous data (outliers) far away from the boundary
  - More or less a linear discriminant
### Derivative rules

<table>
<thead>
<tr>
<th>Function</th>
<th>Derivative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>$c$</td>
</tr>
<tr>
<td>Power</td>
<td>$x^a$</td>
</tr>
<tr>
<td>$1/x$</td>
<td>$x^{-1}$</td>
</tr>
<tr>
<td>$\sqrt{x}$</td>
<td>$x^{1/2}$</td>
</tr>
<tr>
<td>Exponential</td>
<td>$e^x$</td>
</tr>
<tr>
<td>Logarithm</td>
<td>$\ln(x)$</td>
</tr>
</tbody>
</table>

### Vectors of derivatives

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:** $f(x) = x_1^2 x_2^3$

$$\nabla_x f(x) = \left( 2x_1 x_2^3, 3x_1^2 x_2^2 \right)$$

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

### Gradient descent (steepest descent)

- **First order optimisation algorithm using $\nabla_x f(x)$**
- **Optimisation problem:** $\min_x f(x)$
- **Useful when analytic solutions (closed forms) are not available or difficult to find**
- **Algorithm**
  - Set an initial value $x_0$ and set $t = 0$
  - If $\|\nabla_x f(x_t)\| \leq \epsilon$, then stop. Otherwise, do the following.
  - $x_{t+1} = x_t - \eta \nabla_x f(x_t)$ for $\eta > 0$
  - $t = t + 1$, and go to step 2.
- **Problem:** stops at a local minimum (difficult to find a global maximum).

### Linear regression (one variable)

- **Training set:** $D = \{(x_n, t_n)\}_{n=1}^N$
- **Linear regression:** $f_n = a x_n + b$
- **Objective function:** $E = \sum_{n=1}^N (t_n - (a x_n + b))^2$
- **Optimisation problem:** $\min_{a,b} E$
  - **Partial derivatives:**
    $$\frac{\partial E}{\partial a} = 2 \sum_{n=1}^N (t_n - (a x_n + b)) \cdot (-x_n)$$
    $$= 2a \sum_{n=1}^N x_n^2 + 2b \sum_{n=1}^N x_n - 2 \sum_{n=1}^N t_n x_n$$
    $$\frac{\partial E}{\partial b} = -2 \sum_{n=1}^N (t_n - (a x_n + b))$$
    $$= 2a \sum_{n=1}^N x_n + 2b \sum_{n=1}^N 1 - 2 \sum_{n=1}^N t_n$$
  - **Letting** $\frac{\partial E}{\partial a} = 0$ and $\frac{\partial E}{\partial b} = 0$,
    $$\begin{pmatrix} \sum_{n=1}^N x_n^2 & \sum_{n=1}^N x_n \\ \sum_{n=1}^N x_n & \sum_{n=1}^N 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{n=1}^N t_n x_n \\ \sum_{n=1}^N t_n \end{pmatrix}$$

### Linear regression (multiple variables)

- **Training set:** $D = \{(x_n, t_n)\}_{n=1}^N$, where $x_n = (1, x_{n1}, \ldots, x_{nd})^T$
- **Linear regression:** $f_n = w^T x_n$
- **Objective function:** $E = \sum_{n=1}^N (t_n - w^T x_n)^2$
- **Optimisation problem:** $\min_{a,b} E$

### Linear regression (one variable) (cont.)

### Linear regression (multiple variables) (cont.)

- $E = \sum_{n=1}^N (t_n - w^T x_n)^2$
- **Partial derivatives:**
  $$\frac{\partial E}{\partial w_0} = -2 \sum_{n=1}^N (t_n - w^T x_n) x_{n0}$$
- **Vector/matrix representation ($\partial E/\partial w$):**
  $$X = \begin{pmatrix} x_{10} & \cdots & x_{1d} \\ \vdots & \ddots & \vdots \\ x_{N0} & \cdots & x_{Nd} \end{pmatrix}$$
  $$T = \begin{pmatrix} t_1 \\ \vdots \\ t_N \end{pmatrix}$$
  $$E = (T - XW)^T (T - XW)$$
  $$\frac{\partial E}{\partial w} = -2X^T (T - XW)$$
- **Letting** $\frac{\partial E}{\partial w} = 0$,
  $$X^T X W = X^T T$$
  $$W = (X^T X)^{-1} X^T T$$

### 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**