Today’s Schedule

1. Discriminant functions (recap)
2. Decision boundary of linear discriminants
3. Discriminative training of linear discriminant functions
4. Perceptron
5. Perceptron structures and decision boundaries
Discriminant functions (recap)

\[ y_c(x) = \ln(P(x|C)P(C)) \]

\[ = -\frac{1}{2}(x - \mu_c)^T \Sigma_c^{-1}(x - \mu_c) - \frac{1}{2} \ln |\Sigma_c| + \ln P(c) \]

\[ = -\frac{1}{2}x^T \Sigma_c^{-1}x + \mu_c^T \Sigma_c^{-1}x - \frac{1}{2} \mu_c^T \Sigma_c^{-1} \mu_c - \frac{1}{2} \ln |\Sigma_c| + \ln P(c) \]
Linear discriminants for a 2-class problem

\[ y_1(x) = \mathbf{w}_1^T \mathbf{x} + w_{10} \]
\[ y_2(x) = \mathbf{w}_2^T \mathbf{x} + w_{20} \]

Combined discriminant function:

\[ y(x) = y_1(x) - y_2(x) = (\mathbf{w}_1 - \mathbf{w}_2)^T \mathbf{x} + (w_{10} - w_{20}) \]
\[ = \mathbf{w}^T \mathbf{x} + w_0 \]

Decision:

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

- Decision boundary:
  \[ y(x) = w^T x + w_0 = 0 \]

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Decision boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>line</td>
</tr>
<tr>
<td></td>
<td>( w_1 x_1 + w_2 x_2 + w_0 = 0 )</td>
</tr>
<tr>
<td>3</td>
<td>plane</td>
</tr>
<tr>
<td></td>
<td>( w_1 x_1 + w_2 x_2 + w_3 x_3 + w_0 = 0 )</td>
</tr>
<tr>
<td>d</td>
<td>hyperplane</td>
</tr>
<tr>
<td></td>
<td>( (\sum_{i=1}^{d} w_i x_i) + w_0 = 0 )</td>
</tr>
</tbody>
</table>

NB: \( w \) is a normal vector to the hyperplane
Decision boundary of linear discriminant (2D)

\[ y(x) = w_1x_1 + w_2x_2 + w_0 = 0 \]  
\[ (x_2 = -\frac{w_1}{w_2}x_1 - \frac{w_0}{w_2}, \text{ when } w_2 \neq 0) \]
Decision boundary of linear discriminant (3D)

\[ y(x) = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_0 = 0 \]
A discriminant for a two-class problem:

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

\[ = w^T x + w_0 \]
Error correction algorithm

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

where \( \dot{\mathbf{w}} = (w_0, \mathbf{w}^T)^T \), \( \dot{x} = (1, \mathbf{x}^T)^T \)

Let’s just use \( \mathbf{w} \) and \( \mathbf{x} \) to denote \( \dot{\mathbf{w}} \) and \( \dot{\mathbf{x}} \) from now on!

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

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

- Training set: \( D = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_N, t_N)\} \)
  where \( t_i \in \{0, 1\} : \text{target value} \)

- Update \( \mathbf{w} \) on error

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

NB: \( (\mathbf{w}^{(\text{new})})^T \mathbf{x}_i = \mathbf{w}^T \mathbf{x}_i + \eta (t_i - y(\mathbf{x}_i)) \|\mathbf{x}_i\|^2 \)
The Perceptron learning algorithm

Incremental (online) Perceptron algorithm:

\[
\text{for } i = 1, \ldots, N \\
\quad w \leftarrow w + \eta (t_i - y(x_i)) x_i
\]

Batch Perceptron algorithm:

\[
\begin{align*}
\nu_{sum} &= 0 \\
\text{for } i = 1, \ldots, N \\
\quad \nu_{sum} &= \nu_{sum} + (t_i - y(x_i)) x_i \\
\quad w &\leftarrow w + \eta \nu_{sum}
\end{align*}
\]

What about convergence?

The Perceptron learning algorithm terminate if training samples are linearly separable.
Geometry of Perceptron’s error correction

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

<table>
<thead>
<tr>
<th>( t_i - y(x_i) )</th>
<th>( y(x_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ w^T x = \| w \| \| x \| \cos \theta \]
Geometry of Perceptron’s error correction (cont.)

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

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

<table>
<thead>
<tr>
<th>( t_i - y(x_i) )</th>
<th>( y(x_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ w^T x = \| w \| \| x \| \cos \theta \]

\[ w^{(\text{new})} \leftarrow w + \eta (t_i - y(x_i)) x_i \quad (0 < \eta < 1) \]
Geometry of Perceptron’s error correction (cont.)

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

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

\[
\begin{array}{c|cc}
 t_i - y(x_i) & y(x_i) \\
\hline
 0 & 0 & 1 \\
 1 & 0 & -1 \\
\end{array}
\]

\[ w^T x = \| w \| \| x \| \cos \theta \]
Background of Perceptron

(1940s) Warren McCulloch and Walter Pitts: 'threshold logic'
Donald Hebb: 'Hebbian learning'

1957 Frank Rosenblatt: 'Perceptron'

(a) function unit
Character recognition by Perceptron

\( (1,1) \)

\( i \)

\( j \)

\( H \)

\( (W,H) \)

\( W \)

\( \Sigma \)
Linearly separable vs linearly non-separable

(a-1) Linearly separable

(a-2) Linearly non-separable
Problems with the Perceptron learning algorithm

- Non-convergence for linearly 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

- 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 const.)
  \[ \nabla_w E(w) = \left( \frac{\partial E}{\partial w_0}, \ldots, \frac{\partial E}{\partial w_d} \right)^T \]
Training with least squares (cont.)

\[
\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{n=1}^{N} (w^T x_n - t_n)^2
\]

\[
= \sum_{n=1}^{N} (w^T x_n - t_n) \frac{\partial}{\partial w_i} w^T x_n
\]

\[
= \sum_{n=1}^{N} (w^T x_n - t_n) x_{ni}
\]

- Trainable in linearly non-separable case
- Not robust (sensitive) against erroneous data (outliers) far away from the boundary
- More or less a linear discriminant
Perceptron structures and decision boundaries

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

\[ w = (w_0, w_1, \ldots, w_d)^T \]
\[ x = (1, x_1, \ldots, x_d)^T \]

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

\[ a(x) = 1 - x_1 + x_2 \]
\[ = w_0 + w_1 x_1 + w_2 x_2 \]
\[ w_0 = 1, w_1 = -1, w_2 = 1 \]

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>EXOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$y$</td>
<td>$x_1$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Question: find the weights for each function
Perceptron structures and decision boundaries (cont.)
Perceptron structures and decision boundaries (cont.)
Perceptron structures and decision boundaries (cont.)

![Diagram of Perceptron structures and decision boundaries](image)

\[ \sum \]

\[ x_0 \quad x_1 \quad x_2 \]

\[ X_1 \]

\[ X_2 \]
Perceptron structures and decision boundaries (cont.)
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
Appendix – derivatives

Derivatives of functions of one variable

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

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

Partial derivatives of functions of more than one variable

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

e.g., \( f(x, y) = y^3x^2, \quad \frac{\partial f}{\partial x} = 2y^3x \)
### Derivative rules

<table>
<thead>
<tr>
<th>Function</th>
<th>Derivative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>$c$</td>
</tr>
<tr>
<td></td>
<td>$x$</td>
</tr>
<tr>
<td>Power</td>
<td>$x^n$</td>
</tr>
<tr>
<td></td>
<td>$\frac{1}{x}$</td>
</tr>
<tr>
<td></td>
<td>$\sqrt{x}$</td>
</tr>
<tr>
<td>Exponential</td>
<td>$e^x$</td>
</tr>
<tr>
<td>Logarithms</td>
<td>$\ln(x)$</td>
</tr>
<tr>
<td>Sum rule</td>
<td>$f(x) + g(x)$</td>
</tr>
<tr>
<td>Product rule</td>
<td>$f(x)g(x)$</td>
</tr>
<tr>
<td>Reciprocal rule</td>
<td>$\frac{1}{f(x)}$</td>
</tr>
<tr>
<td></td>
<td>$\frac{f(x)}{g(x)}$</td>
</tr>
<tr>
<td>Chain rule</td>
<td>$f(g(x))$</td>
</tr>
<tr>
<td>$z = f(y), y = g(x)$</td>
<td>$\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx}$</td>
</tr>
</tbody>
</table>
Consider \( f(x) \), where \( x = (x_1, \ldots, x_d)^T \)

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

\[
\nabla_x f(x) = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_d} \right)^T
\]

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

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

**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**

1. Set an initial value $x_0$ and set $t = 0$
2. If $\|\nabla_x f(x_t)\| \approx 0$, then stop. Otherwise, do the following.
3. $x_{t+1} = x_t - \eta \nabla_x f(x_t)$ for $\eta > 0$
4. $t = t + 1$, and go to step 2.

- Problem: stops at a local minimum (difficult to find a global maximum).
Linear regression (one variable) least squares line fitting

- Training set: \( D = \{(x_n, t_n)\}_{n=1}^{N} \)
- Linear regression: \( \hat{t}_n = ax_n + b \)
- Objective function: \( E = \sum_{n=1}^{N} (t_i - (ax_i + b))^2 \)
- Optimisation problem: \( \min_{a, b} E \)
- Partial derivatives:
  \[
  \frac{\partial E}{\partial a} = 2 \sum_{n=1}^{N} ((t_i - (ax_i + b)) (-x_i)) \\
  = 2a \sum_{n=1}^{N} x_i^2 + 2b \sum_{n=1}^{N} x_i - 2 \sum_{n=1}^{N} t_i x_i
  \]
  \[
  \frac{\partial E}{\partial b} = -2 \sum_{n=1}^{N} ((t_i - (ax_i + b))) \\
  = 2a \sum_{n=1}^{N} x_i + 2b \sum_{n=1}^{N} 1 - 2 \sum_{n=1}^{N} t_i
  \]
Letting $\frac{\partial E}{\partial a} = 0$ and $\frac{\partial E}{\partial b} = 0$

\[
\begin{pmatrix}
\sum_{n=1}^{N} x_i^2 & \sum_{n=1}^{N} x_i \\
\sum_{n=1}^{N} x_i & \sum_{n=1}^{N} 1
\end{pmatrix}
\begin{pmatrix}
a \\
b
\end{pmatrix}
= 
\begin{pmatrix}
\sum_{n=1}^{N} t_i x_i \\
\sum_{n=1}^{N} t_i
\end{pmatrix}
\]
Linear regression (multiple variables)

Training set:
\[ D = \{(\mathbf{x}_n, t_n)\}_{n=1}^N, \text{ where } \mathbf{x}_n = (1, x_1, \ldots, x_d)^T \]

Linear regression:
\[ \hat{t}_n = \mathbf{w}^T \mathbf{x}_n \]

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

Optimisation problem:
\[ \min_{a, b} E \]

Elements of Statistical Learning (2nd Ed.) © Hastie, Tibshirani & Friedman 2009
Linear regression (multiple variables) (cont.)

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

- Partial derivatives:
  \[
  \frac{\partial E}{\partial w_i} = -2 \sum_{n=1}^{N} (t_n - \mathbf{w}^T \mathbf{x}_n) x_{ni}
  \]

- Vector/matrix representation:
  \[
  \mathbf{X} = \begin{bmatrix}
  \mathbf{x}_1^T \\
  \vdots \\
  \mathbf{x}_N^T
  \end{bmatrix}
  = \begin{bmatrix}
  x_{10}, \ldots, x_{1d} \\
  \vdots \\
  x_{N0}, \ldots, x_{Nd}
  \end{bmatrix},
  \mathbf{T} = \begin{bmatrix}
  t_1 \\
  \vdots \\
  t_N
  \end{bmatrix}
  \]
  \[
  E = (\mathbf{T} - \mathbf{Xw})^T (\mathbf{T} - \mathbf{Xw})
  \]
  \[
  \frac{\partial E}{\partial \mathbf{w}} = -2\mathbf{X}^T (\mathbf{T} - \mathbf{Xw})
  \]

- Letting \( \frac{\partial E}{\partial \mathbf{w}} = \mathbf{0} \) \( \Rightarrow \mathbf{X}^T (\mathbf{T} - \mathbf{Xw}) = \mathbf{0} \)
  \[
  \mathbf{X}^T \mathbf{Xw} = \mathbf{X}^T \mathbf{T}
  \]
  \[
  \mathbf{W} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{T}
  \]
  \( \cdots \) analytic solution if the inverse exists