# Classification

Cédric Archambeau

cedrica@amazon.com

Peyresq Summer School
France, July 2016

# Overview

1. Classification (2.5 hours)
2. Clustering (1.5 hours)
3. Practical sessions (1 hour)

# Overview

1. Classification (2.5 hours)
2. Clustering (1.5 hours)
3. Practical sessions (1 hour)

## LEARNING GOALS

- Understand what is a classification problem and when it can be applied.
- Being able to reason about new models and derive learning algorithms.

# Overview

1. Classification (2.5 hours)
2. Clustering (1.5 hours)
3. Practical sessions (1 hour)

## LEARNING GOALS

- Understand what is a classification problem and when it can be applied.
- Being able to reason about new models and derive learning algorithms.
- **Being able to learn more by yourself!**

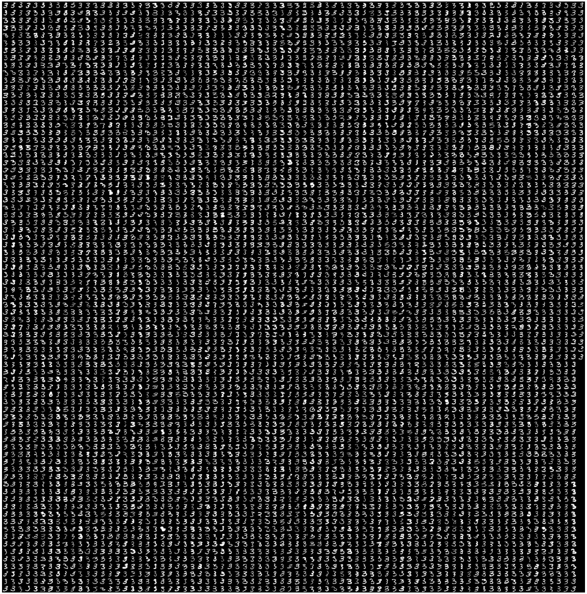
# Outline


1. What is classification?
2. Decision theory
3. Generative classifiers
4. Discriminative classifiers
5. Summary
6. Exercises

# Outline


1. What is classification?
2. Decision theory
3. Generative classifiers
4. Discriminative classifiers
5. Summary
6. Exercises

# A first example: digit classification



MNIST handwritten digits

# A first example: digit classification

MNIST handwritten digit sample

# A first example: digit classification

MNIST handwritten digit sample

Given an image, can we predict which digit it is (i.e., which label it has)?

# A first example: digit classification

MNIST handwritten digit sample

Given an image, can we predict which digit it is (i.e., which label it has)?

Pre-processed data set of handwritten digits: $\mathcal{D} = \{(\boldsymbol{x}_i, t_i) | i = 1, \ldots, n\}$.

- http://yann.lecun.com/exdb/mnist

# A first example: digit classification

MNIST handwritten digit sample

Given an image, can we predict which digit it is (i.e., which label it has)?

Pre-processed data set of handwritten digits: $\mathcal{D} = \{(\boldsymbol{x}_i, t_i) | i = 1, \ldots, n\}$.

- `http://yann.lecun.com/exdb/mnist`
- Instance or *data point* $i$ consists in a $28 \times 28$ bitmap image $\boldsymbol{x}_i$ and a label $t_i \in \{0, \ldots, 9\}$.

# A first example: digit classification

MNIST handwritten digit sample

Given an image, can we predict which digit it is (i.e., which label it has)?

Pre-processed data set of handwritten digits: $\mathcal{D} = \{(\boldsymbol{x}_i, t_i) | i = 1, \ldots, n\}$.

- http://yann.lecun.com/exdb/mnist
- Instance or *data point* $i$ consists in a $28 \times 28$ bitmap image $\boldsymbol{x}_i$ and a label $t_i \in \{0, \ldots, 9\}$.
- Each image is represented as a 784-dimensional vector of pixels, quantized to $\{0, \ldots, 255\}$.

# A first example: digit classification

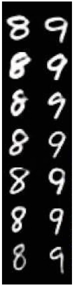
MNIST handwritten digit sample

Can we learn $f : \boldsymbol{x} \mapsto f(\boldsymbol{x}) = t$?

Pre-processed data set of handwritten digits: $\mathcal{D} = \{(\boldsymbol{x}_i, t_i) | i = 1, \ldots, n\}$.

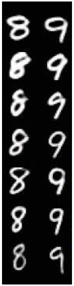
- `http://yann.lecun.com/exdb/mnist`
- Instance or *data point* $i$ consists in a $28 \times 28$ bitmap image $\boldsymbol{x}_i$ and a label $t_i \in \{0, \ldots, 9\}$.
- Each image is represented as a 784-dimensional vector of pixels, quantized to $\{0, \ldots, 255\}$.

# A first example: digit classification



We would like to distinguish digit 8 from digit 9:

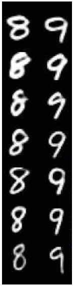
# A first example: digit classification



We would like to distinguish digit 8 from digit 9:

$$f : \mathbb{R}^d \to \{-1, +1\}$$

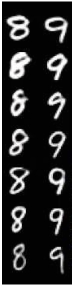
# A first example: digit classification




We would like to distinguish digit 8 from digit 9:

$$f : \mathbb{R}^d \to \{-1, +1\} \text{ or } f : \mathbb{R}^d \to \{0, 1\}.$$

# A first example: digit classification

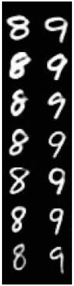


We would like to distinguish digit 8 from digit 9:

$$f : \mathbb{R}^d \to \{-1, +1\} \text{ or } f : \mathbb{R}^d \to \{0, 1\}.$$

This is a binary classification problem!

# A first example: digit classification



We would like to distinguish digit 8 from digit 9:

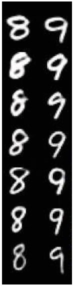
$$f : \mathbb{R}^d \to \{-1, +1\} \text{ or } f : \mathbb{R}^d \to \{0, 1\}.$$

This is a binary classification problem!

- Why not a look-up table?

$$f_{\mathrm{LU}}(\boldsymbol{x}) = \begin{cases} t_i & \text{if } \boldsymbol{x} = \boldsymbol{x}_i, i \in \{1, \dots, n\}, \\ \text{Don't know} & \text{if } \boldsymbol{x} \neq \boldsymbol{x}_i, i \in \{1, \dots, n\}. \end{cases}$$

# A first example: digit classification



We would like to distinguish digit 8 from digit 9:

$$f : \mathbb{R}^d \to \{-1, +1\} \text{ or } f : \mathbb{R}^d \to \{0, 1\}.$$

This is a binary classification problem!

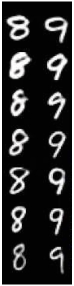
- Why not a look-up table?

$$f_{\mathrm{LU}}(\boldsymbol{x}) = \begin{cases} t_i & \text{if } \boldsymbol{x} = \boldsymbol{x}_i, i \in \{1, \ldots, n\}, \\ \text{Don't know} & \text{if } \boldsymbol{x} \neq \boldsymbol{x}_i, i \in \{1, \ldots, n\}. \end{cases}$$

- Why not nearest neighbours?

$$f_{\mathrm{NN}}(\boldsymbol{x}) = t_i \iff \|\boldsymbol{x} - \boldsymbol{x}_i\| \leqslant \|\boldsymbol{x} - \boldsymbol{x}_j\|, j \in \{1, \ldots, n\}.$$

# A first example: digit classification



We would like to distinguish digit 8 from digit 9:

$$f(\boldsymbol{x}) = \mathrm{sign}(y(\boldsymbol{x})) = \begin{cases} +1 & \text{if } y(\boldsymbol{x}) > 0, \\ -1 & \text{if } y(\boldsymbol{x}) < 0. \end{cases}$$

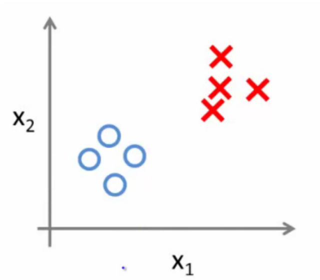
- Why not a look-up table?

$$f_{\mathrm{LU}}(\boldsymbol{x}) = \begin{cases} t_i & \text{if } \boldsymbol{x} = \boldsymbol{x}_i, i \in \{1, \ldots, n\}, \\ \text{Don't know} & \text{if } \boldsymbol{x} \neq \boldsymbol{x}_i, i \in \{1, \ldots, n\}. \end{cases}$$

- Why not nearest neighbours?

$$f_{\mathrm{NN}}(\boldsymbol{x}) = t_i \quad \Longleftrightarrow \quad \|\boldsymbol{x} - \boldsymbol{x}_i\| \leqslant \|\boldsymbol{x} - \boldsymbol{x}_j\|, j \in \{1, \ldots, n\}.$$

# Linear discriminant function (aka linear classifier)

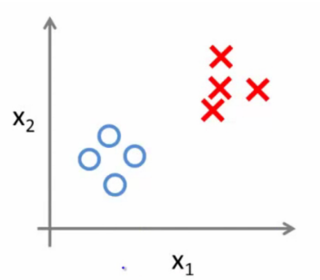



We assume the instances can be separated by a linear subspace (or hyperplane):

$$y(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b, \quad f_{\mathrm{LIN}}(\boldsymbol{x}) = \mathrm{sign}(y(\boldsymbol{x})),$$

where $\boldsymbol{w} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$, $b \in \mathbb{R}$.

# Linear discriminant function (aka linear classifier)



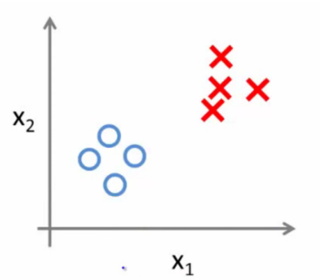

We assume the instances can be separated by a linear subspace (or hyperplane):

$$y(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b, \quad f_{\mathrm{LIN}}(\boldsymbol{x}) = \mathrm{sign}(y(\boldsymbol{x})),$$

where $\boldsymbol{w} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$, $b \in \mathbb{R}$.

- The decision boundary is the set $\{\boldsymbol{x} : y(\boldsymbol{x}) = 0\}$.

# Linear discriminant function (aka linear classifier)




We assume the instances can be separated by a linear subspace (or hyperplane):

$$y(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b, \quad f_{\mathrm{LIN}}(\boldsymbol{x}) = \mathrm{sign}(y(\boldsymbol{x})),$$

where $\boldsymbol{w} \in \mathbb{R}^d \backslash \{\mathbf{0}\}$, $b \in \mathbb{R}$.

- The decision boundary is the set $\{\boldsymbol{x} : y(\boldsymbol{x}) = 0\}$.
- Learning is to find $\boldsymbol{w}$ and $b$ such that $\forall i : f_{\mathrm{LIN}}(\boldsymbol{x}_i) \approx t_i$.

# Relation to nearest neighbour classification?

$$
\begin{aligned}
& \|\boldsymbol{x}_* - \boldsymbol{w}_{+1}\| < \|\boldsymbol{x}_* - \boldsymbol{w}_{-1}\| \quad \Leftrightarrow \quad \|\boldsymbol{x}_* - \boldsymbol{w}_{+1}\|^2 < \|\boldsymbol{x}_* - \boldsymbol{w}_{-1}\|^2 \\
\Leftrightarrow \; & \|\boldsymbol{x}_*\|^2 - 2\boldsymbol{w}_{+1}^T \boldsymbol{x}_* + \|\boldsymbol{w}_{+1}\|^2 < \|\boldsymbol{x}_*\|^2 - 2\boldsymbol{w}_{-1}^T \boldsymbol{x}_* + \|\boldsymbol{w}_{-1}\|^2 \\
\Leftrightarrow \; & \boldsymbol{w}_{+1}^T \boldsymbol{x}_* - \|\boldsymbol{w}_{+1}\|^2/2 > \boldsymbol{w}_{-1}^T \boldsymbol{x}_* - \|\boldsymbol{w}_{-1}\|^2/2 \\
\Leftrightarrow \; & (\boldsymbol{w}_{+1} - \boldsymbol{w}_{-1})^T \boldsymbol{x}_* + \frac{1}{2}\left(\|\boldsymbol{w}_{-1}\|^2 - \|\boldsymbol{w}_{+1}\|^2\right) > 0.
\end{aligned}
$$

# How can we picture a linear discriminant function?

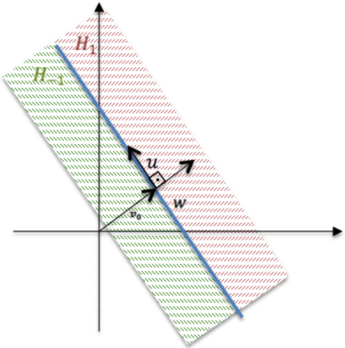



Figure 2.6: Separating hyperplane in $\mathbb{R}^2$. The decision boundary (blue) is defined by the normal vector $\boldsymbol{w}$ and an offset $b \in \mathbb{R}$. $\boldsymbol{v}_0$ is a point on the hyperplane, obtained by orthogonal projection of the origin. The plane separates $\mathbb{R}^2$ into two halfspaces $\mathcal{H}_{+1}$ $(\boldsymbol{w}^T\boldsymbol{x} + b > 0)$ and $\mathcal{H}_{-1}$ $(\boldsymbol{w}^T\boldsymbol{x} + b < 0)$, the decision regions of the corresponding linear discriminant.

# How can we picture a linear discriminant function?

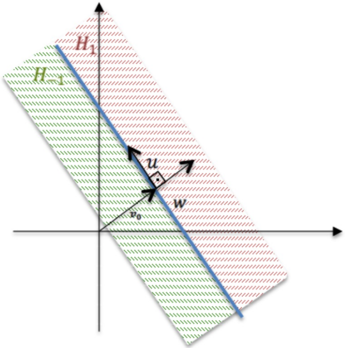



Figure 2.6: Separating hyperplane in $\mathbb{R}^2$. The decision boundary (blue) is defined by the normal vector $\boldsymbol{w}$ and an offset $b \in \mathbb{R}$. $\boldsymbol{v}_0$ is a point on the hyperplane, obtained by orthogonal projection of the origin. The plane separates $\mathbb{R}^2$ into two halfspaces $\mathcal{H}_{+1}$ $(\boldsymbol{w}^T\boldsymbol{x} + b > 0)$ and $\mathcal{H}_{-1}$ $(\boldsymbol{w}^T\boldsymbol{x} + b < 0)$, the decision regions of the corresponding linear discriminant.

- Vector $\boldsymbol{w}$ is orthogonal to any vector $\boldsymbol{u}$ in the hyperplane: $\boldsymbol{w}^\top \boldsymbol{u} = 0$.

# How can we picture a linear discriminant function?

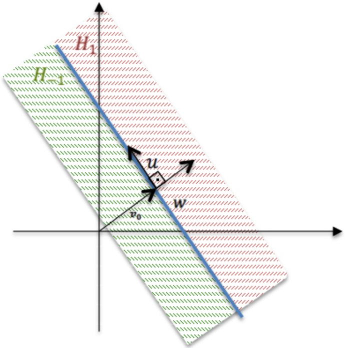



Figure 2.6: Separating hyperplane in $\mathbb{R}^2$. The decision boundary (blue) is defined by the normal vector $\boldsymbol{w}$ and an offset $b \in \mathbb{R}$. $\boldsymbol{v}_0$ is a point on the hyperplane, obtained by orthogonal projection of the origin. The plane separates $\mathbb{R}^2$ into two halfspaces $\mathcal{H}_{+1}$ ($\boldsymbol{w}^T\boldsymbol{x} + b > 0$) and $\mathcal{H}_{-1}$ ($\boldsymbol{w}^T\boldsymbol{x} + b < 0$), the decision regions of the corresponding linear discriminant.

- Vector $\boldsymbol{w}$ is orthogonal to any vector $\boldsymbol{u}$ in the hyperplane: $\boldsymbol{w}^\top \boldsymbol{u} = 0$.
- Offset vector $\boldsymbol{v}_0 = -(b/\|\boldsymbol{w}\|^2)\boldsymbol{w}$ is the projection of the origin.

# How can we picture a linear discriminant function?

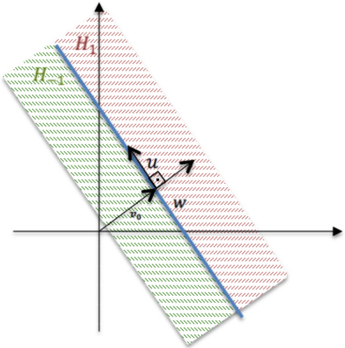
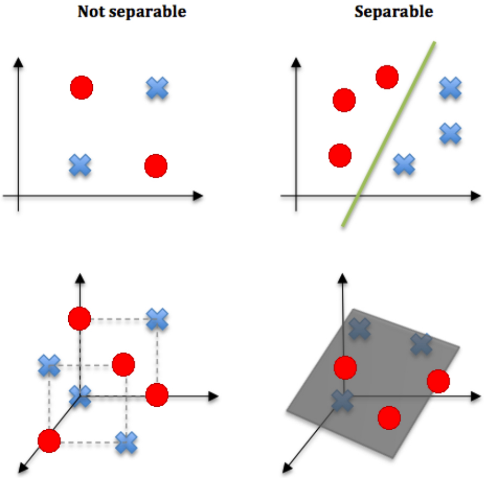



Figure 2.6: Separating hyperplane in $\mathbb{R}^2$. The decision boundary (blue) is defined by the normal vector $\boldsymbol{w}$ and an offset $b \in \mathbb{R}$. $\boldsymbol{v}_0$ is a point on the hyperplane, obtained by orthogonal projection of the origin. The plane separates $\mathbb{R}^2$ into two halfspaces $\mathcal{H}_{+1}$ ($\boldsymbol{w}^T\boldsymbol{x} + b > 0$) and $\mathcal{H}_{-1}$ ($\boldsymbol{w}^T\boldsymbol{x} + b < 0$), the decision regions of the corresponding linear discriminant.

- Vector $\boldsymbol{w}$ is orthogonal to any vector $\boldsymbol{u}$ in the hyperplane: $\boldsymbol{w}^\top \boldsymbol{u} = 0$.
- Offset vector $\boldsymbol{v}_0 = -(b/\|\boldsymbol{w}\|^2)\boldsymbol{w}$ is the projection of the origin.
- We restrict weight vectors to be unit norm: $\{\boldsymbol{w} : \|\boldsymbol{w}\| = 1\}$.

# Can instances always be separated?

# Can instances always be separated?



# Feature maps

- Representing digits as vectors of pixels is arbitrary. Perhaps a transformation would be beneficial?

# Feature maps

- Representing digits as vectors of pixels is arbitrary. Perhaps a transformation would be beneficial?
- Let $\phi : \mathbb{R}^d \to \mathbb{R}^p : \boldsymbol{x} \mapsto \phi(\boldsymbol{x})$. The feature function $\phi(\boldsymbol{x})$ defines a mapping of input space into a feature space.

# Feature maps

- Representing digits as vectors of pixels is arbitrary. Perhaps a transformation would be beneficial?
- Let $\phi : \mathbb{R}^d \to \mathbb{R}^p : \boldsymbol{x} \mapsto \phi(\boldsymbol{x})$. The feature function $\phi(\boldsymbol{x})$ defines a mapping of input space into a feature space.
- We generalise linear classifiers by learning them in the feature space:

$$y(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x}) + b, \qquad f_{\mathrm{LIN}}(\boldsymbol{x}) = \mathrm{sign}(y(\boldsymbol{x})).$$

# Feature maps

- Representing digits as vectors of pixels is arbitrary. Perhaps a transformation would be beneficial?
- Let $\phi : \mathbb{R}^d \to \mathbb{R}^p : \boldsymbol{x} \mapsto \phi(\boldsymbol{x})$. The feature function $\phi(\boldsymbol{x})$ defines a mapping of input space into a feature space.
- We generalise linear classifiers by learning them in the feature space:

$$y(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x}) + b, \qquad f_{\mathrm{LIN}}(\boldsymbol{x}) = \mathrm{sign}(y(\boldsymbol{x})).$$

Example: linear classifier with quadratic features

$$\phi(\boldsymbol{x}) = \begin{bmatrix} x_1 \\ \vdots \\ x_d \\ x_1 x_1 \\ \vdots \\ x_1 x_d \\ x_2 x_2 \\ \vdots \\ x_2 x_d \\ \vdots \\ x_d x_d \end{bmatrix} \in \mathbb{R}^{d(d+3)/2}. \qquad y(x) = \boldsymbol{w}^\top \phi(\boldsymbol{x}) + b = \sum_j w_j x_j + \sum_j \sum_{k \leqslant j} w_{jk} x_j x_k + b$$

# How do we estimate $\boldsymbol{w}$?

Perceptron (Rosenblatt, '62):

- Instance $i$ is correctly classified if $\boldsymbol{w}^\top \phi(\boldsymbol{x}_i) t_i > 0$.

# How do we estimate $\boldsymbol{w}$?

Perceptron (Rosenblatt, '62):

- Instance $i$ is correctly classified if $\boldsymbol{w}^\top \phi(\boldsymbol{x}_i) t_i > 0$.
- Perceptron criterion:

$$E_{\mathrm{P}}(\boldsymbol{w}) = -\sum_{i \in \mathcal{M}} \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) t_i,$$

where $\mathcal{M} = \{i : \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) t_i < 0\}$.

# How do we estimate $\boldsymbol{w}$?

Perceptron (Rosenblatt, '62):

- Instance $i$ is correctly classified if $\boldsymbol{w}^\top \phi(\boldsymbol{x}_i) t_i > 0$.
- Perceptron criterion:

$$E_{\mathrm{P}}(\boldsymbol{w}) = -\sum_{i \in \mathcal{M}} \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) t_i,$$

  where $\mathcal{M} = \{i : \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) t_i < 0\}$.
- The error $E_{\mathrm{P}}(\boldsymbol{w})$ can be minimised by applying to following rule:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \phi(\boldsymbol{x}_i) t_i.$$

# How do we estimate $\boldsymbol{w}$?

Perceptron (Rosenblatt, '62):

- Instance $i$ is correctly classified if $\boldsymbol{w}^\top \phi(\boldsymbol{x}_i) t_i > 0$.
- Perceptron criterion:

$$E_{\mathrm{P}}(\boldsymbol{w}) = -\sum_{i \in \mathcal{M}} \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) t_i,$$

  where $\mathcal{M} = \{i : \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) t_i < 0\}$.
- The error $E_{\mathrm{P}}(\boldsymbol{w})$ can be minimised by applying to following rule:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \phi(\boldsymbol{x}_i) t_i.$$

- Why does this algorithm work?

# How do we estimate $\boldsymbol{w}$?

Perceptron (Rosenblatt, '62):

- Instance $i$ is correctly classified if $\boldsymbol{w}^\top \phi(\boldsymbol{x}_i) t_i > 0$.
- Perceptron criterion:

$$E_{\mathrm{P}}(\boldsymbol{w}) = -\sum_{i \in \mathcal{M}} \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) t_i,$$

  where $\mathcal{M} = \{i : \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) t_i < 0\}$.
- The error $E_{\mathrm{P}}(\boldsymbol{w})$ can be minimised by applying to following rule:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \phi(\boldsymbol{x}_i) t_i.$$

- Why does this algorithm work?
  $t_i \phi(\boldsymbol{x}_i)^\top (\boldsymbol{w} + t_i \phi(\boldsymbol{x}_i)) = t_i \phi(\boldsymbol{x}_i)^\top \boldsymbol{w} + \|\phi(\boldsymbol{x}_i)\|^2 > t_i \phi(\boldsymbol{x}_i)^\top \boldsymbol{w}$.

# How do we estimate $\boldsymbol{w}$?

Perceptron (Rosenblatt, '62):

- Instance $i$ is correctly classified if $\boldsymbol{w}^\top \phi(\boldsymbol{x}_i) t_i > 0$.
- Perceptron criterion:

$$E_{\mathrm{P}}(\boldsymbol{w}) = -\sum_{i \in \mathcal{M}} \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) t_i,$$

  where $\mathcal{M} = \{i : \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) t_i < 0\}$.
- The error $E_{\mathrm{P}}(\boldsymbol{w})$ can be minimised by applying to following rule:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \phi(\boldsymbol{x}_i) t_i.$$

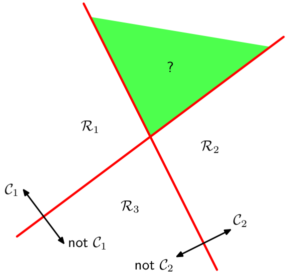
- Why does this algorithm work?
  $t_i \phi(\boldsymbol{x}_i)^\top (\boldsymbol{w} + t_i \phi(\boldsymbol{x}_i)) = t_i \phi(\boldsymbol{x}_i)^\top \boldsymbol{w} + \|\phi(\boldsymbol{x}_i)\|^2 > t_i \phi(\boldsymbol{x}_i)^\top \boldsymbol{w}$.
- If the data is not linearly separable, then the perceptron will not converge :-(

# How should we handle multi-class problems?

- By combining binary classifiers?

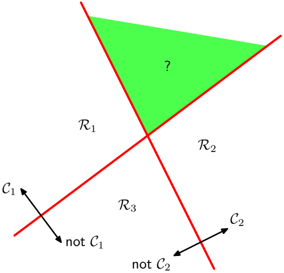
# How should we handle multi-class problems?

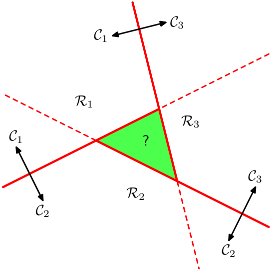
- By combining binary classifiers?



# How should we handle multi-class problems?

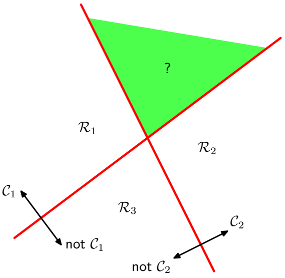
- By combining binary classifiers?

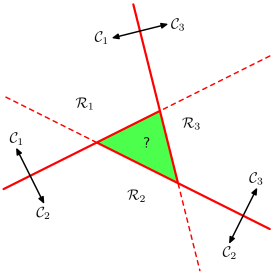


# How should we handle multi-class problems?

- By combining binary classifiers?

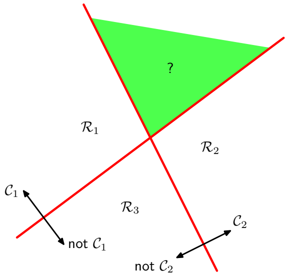


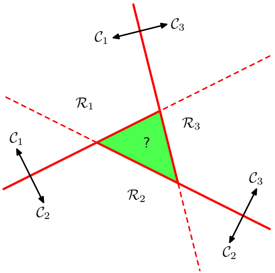




- Are there other ways?

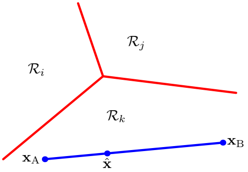
# How should we handle multi-class problems?

- By combining binary classifiers?




- Are there other ways?




$$y_k(\boldsymbol{x}) = \boldsymbol{w}_k^\top \phi(\boldsymbol{x}) + b_k, \qquad f(\boldsymbol{x}) = \arg\max_k \{y_1(\boldsymbol{x}), \ldots, y_m(\boldsymbol{x})\}.$$

# Examples of classification problems

# Examples of classification problems

- Spam detection

# Examples of classification problems

- Spam detection
- Fraud detection

# Examples of classification problems

- Spam detection
- Fraud detection
- Document categorisation

# Examples of classification problems

- Spam detection
- Fraud detection
- Document categorisation
- Sentiment analysis

# Examples of classification problems

- Spam detection
- Fraud detection
- Document categorisation
- Sentiment analysis
- Face recognition

# Examples of classification problems

- Spam detection
- Fraud detection
- Document categorisation
- Sentiment analysis
- Face recognition
- Object categorisation

# Examples of classification problems

- Spam detection
- Fraud detection
- Document categorisation
- Sentiment analysis
- Face recognition
- Object categorisation
- ...

# Outline


1. What is classification?
2. Decision theory
3. Generative classifiers
4. Discriminative classifiers
5. Summary
6. Exercises

# Bayes' rule

$$P(t|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|t)P(t)}{p(\boldsymbol{x})}, \qquad p(\boldsymbol{x}) = \sum_t p(\boldsymbol{x}|t)P(t).$$

# Bayes' rule

$$P(t|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|t)P(t)}{p(\boldsymbol{x})}, \qquad p(\boldsymbol{x}) = \sum_t p(\boldsymbol{x}|t)P(t).$$

- $P(t)$ is the class prior

# Bayes' rule

$$P(t|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|t)P(t)}{p(\boldsymbol{x})}, \qquad p(\boldsymbol{x}) = \sum_t p(\boldsymbol{x}|t)P(t).$$

- $P(t)$ is the class prior
- $P(t|\boldsymbol{x})$ is the class posterior

# Bayes' rule

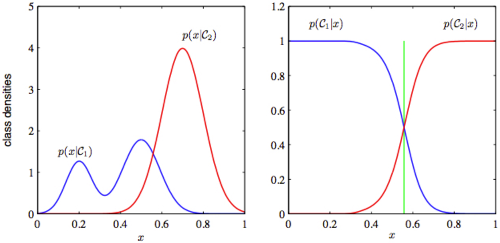
$$P(t|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|t)P(t)}{p(\boldsymbol{x})}, \qquad p(\boldsymbol{x}) = \sum_t p(\boldsymbol{x}|t)P(t).$$

- $P(t)$ is the class prior
- $P(t|\boldsymbol{x})$ is the class posterior
- $p(\boldsymbol{x}|t)$ is the class-conditional density (or likelihood)

# Bayes' rule

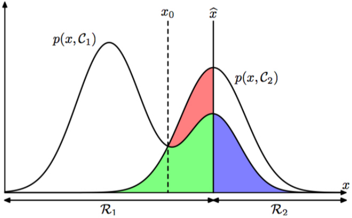
$$P(t|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|t)P(t)}{p(\boldsymbol{x})}, \qquad p(\boldsymbol{x}) = \sum_t p(\boldsymbol{x}|t)P(t).$$

- $P(t)$ is the class prior
- $P(t|\boldsymbol{x})$ is the class posterior
- $p(\boldsymbol{x}|t)$ is the class-conditional density (or likelihood)



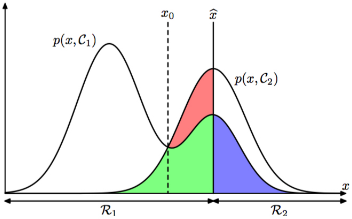

$$P(C_1) = P(t = -1),$$
$$P(C_1|x) = P(t = -1|x),$$
$$p(x, C_1) = p(x, t = -1).$$

# Minimising the classification error




$$
\begin{aligned}
p(x, C_1) &= p(x, t = -1) \\
&= p(x|t = -1)P(t = -1).
\end{aligned}
$$

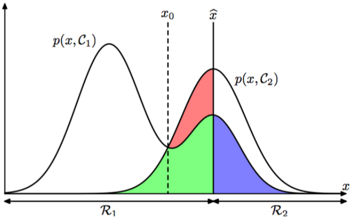
# Minimising the classification error




$$
\begin{aligned}
p(x, C_1) &= p(x, t = -1) \\
&= p(x|t = -1)P(t = -1).
\end{aligned}
$$

- Let $\hat{x}$ be the decision threshold: $\mathcal{R}_1 = \{x : x < \hat{x}\}$ and $\mathcal{R}_2 = \{x : x > \hat{x}\}$.

# Minimising the classification error



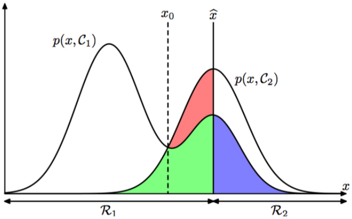

$$
\begin{aligned}
p(x, C_1) &= p(x, t = -1) \\
&= p(x|t = -1)P(t = -1).
\end{aligned}
$$

- Let $\hat{x}$ be the decision threshold: $\mathcal{R}_1 = \{x : x < \hat{x}\}$ and $\mathcal{R}_2 = \{x : x > \hat{x}\}$.
- The misclassification rate is the combined coloured areas:

$$P(\mathrm{error}) = P(x \in \mathcal{R}_1, C_2) + P(x \in \mathcal{R}_2, C_1),$$

where $P(x \in \mathcal{R}_1, C_2) = \int_{x \in \mathcal{R}_1} p(x, C_2)dx$.

# Minimising the classification error



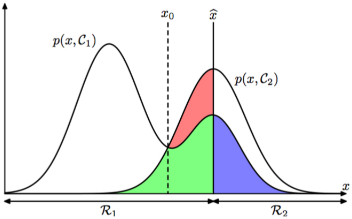

$$p(x, C_1) = p(x, t = -1)$$
$$= p(x|t = -1)P(t = -1).$$

- Let $\hat{x}$ be the decision threshold: $\mathcal{R}_1 = \{x : x < \hat{x}\}$ and $\mathcal{R}_2 = \{x : x > \hat{x}\}$.
- The misclassification rate is the combined coloured areas:

$$P(\mathrm{error}) = P(x \in \mathcal{R}_1, C_2) + P(x \in \mathcal{R}_2, C_1),$$

  where $P(x \in \mathcal{R}_1, C_2) = \int_{x \in \mathcal{R}_1} p(x, C_2) dx$.
- False positives: blue area. False negatives: green+red area.

# Minimising the classification error




$$p(x, C_1) = p(x, t = -1)$$
$$= p(x|t = -1)P(t = -1).$$

- Let $\hat{x}$ be the decision threshold: $\mathcal{R}_1 = \{x : x < \hat{x}\}$ and $\mathcal{R}_2 = \{x : x > \hat{x}\}$.
- The misclassification rate is the combined coloured areas:

$$P(\text{error}) = P(x \in \mathcal{R}_1, C_2) + P(x \in \mathcal{R}_2, C_1),$$

  where $P(x \in \mathcal{R}_1, C_2) = \int_{x \in \mathcal{R}_1} p(x, C_2) dx$.
- False positives: blue area. False negatives: green+red area.
- Bayes error at $x = x_0$: blue+green area.

# Example of a Bayes optimal classifier

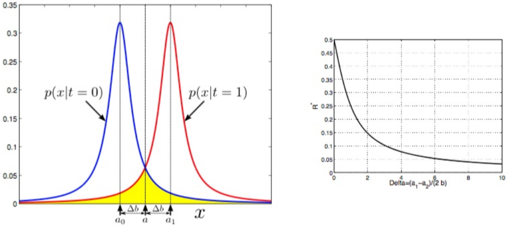
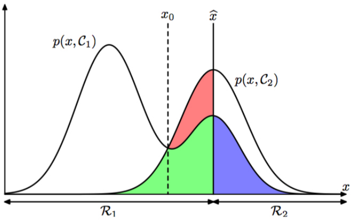


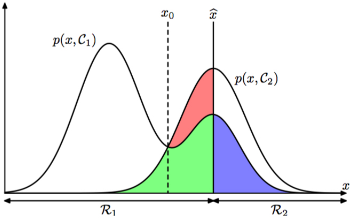

Figure 5.5: Bayes-optimal classifier and Bayes error for two class-conditional Cauchy distributions, centered at $a_0$ and $a_1$. The optimal rule thresholds at the midpoint $a = (a_0 + a_1)/2$. Since the class prior is $P(t = 0) = P(t = 1) = 1/2$, the Bayes error $R^*$ is twice the yellow area. Right plot show $R^*$ as function of separation parameter $\Delta$. The slow decay of $R^*$ is due to the very heavy tails of the Cauchy distributions.

# Precision and recall




| | $t = 1$ | $t = -1$ |
|---|---|---|
| $x > x_0$ | $TP$ | $FP$ |
| $x < x_0$ | $FN$ | $TN$ |

# Precision and recall



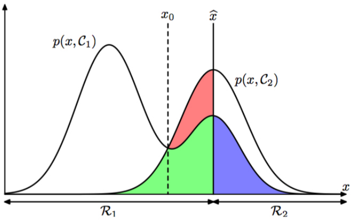

| | $t = 1$ | $t = -1$ |
|---|---|---|
| $x > x_0$ | $TP$ | $FP$ |
| $x < x_0$ | $FN$ | $TN$ |

- The precision is the proportion of positives in the instances classified as being positive:

$$P(x) = \frac{TP(x)}{TP(x) + FP(x)},$$

where $TP$ are the true positives and $FP$ the false positives.

# Precision and recall




| | $t = 1$ | $t = -1$ |
|---|---|---|
| $x > x_0$ | $TP$ | $FP$ |
| $x < x_0$ | $FN$ | $TN$ |

- The precision is the proportion of positives in the instances classified as being positive:

$$P(x) = \frac{TP(x)}{TP(x) + FP(x)},$$

where $TP$ are the true positives and $FP$ the false positives.

- The recall is the proportion of correctly classified positives:

$$R(x) = \frac{TP(x)}{TP(x) + FN(x)},$$

where $FN$ are the false negatives.

# Should we always minimise the misclassification rate?

# Should we always minimise the misclassification rate?

In a hospital, a tissue sample is taken from a patient, giving rise to an input vector $\boldsymbol{x}$. A classifier $f(\boldsymbol{x})$ is to predict whether the patient has cancer ($t = 1$) or not ($t = -1$). Is the cost of predicting that the patient has cancer while he/she has not the same, as predicting that the patient has not contracted cancer while he/she has the disease?

# Should we always minimise the misclassification rate?

In a hospital, a tissue sample is taken from a patient, giving rise to an input vector $\boldsymbol{x}$. A classifier $f(\boldsymbol{x})$ is to predict whether the patient has cancer ($t = 1$) or not ($t = -1$). Is the cost of predicting that the patient has cancer while he/she has not the same, as predicting that the patient has not contracted cancer while he/she has the disease?

Let us define a loss function, which assigns a unique loss to every decision we could take:

$$\begin{array}{cc} & \begin{array}{cc} \text{cancer} & \text{normal} \end{array} \\ \begin{array}{c} \text{cancer} \\ \text{normal} \end{array} & \begin{pmatrix} 0 & 1000 \\ 1 & 0 \end{pmatrix} \end{array}$$

# Should we always minimise the misclassification rate?

In a hospital, a tissue sample is taken from a patient, giving rise to an input vector $\boldsymbol{x}$. A classifier $f(\boldsymbol{x})$ is to predict whether the patient has cancer ($t = 1$) or not ($t = -1$). Is the cost of predicting that the patient has cancer while he/she has not the same, as predicting that the patient has not contracted cancer while he/she has the disease?

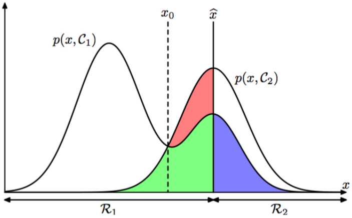
Let us define a loss function, which assigns a unique loss to every decision we could take:

$$\begin{array}{c c} & \begin{array}{cc} \text{cancer} & \text{normal} \end{array} \\ \begin{array}{c} \text{cancer} \\ \text{normal} \end{array} & \begin{pmatrix} 0 & 1000 \\ 1 & 0 \end{pmatrix} \end{array}$$

The expected loss is given by

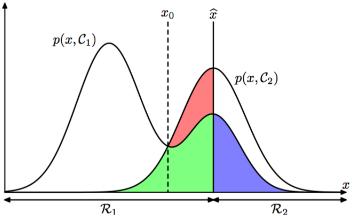
$$\mathbb{E}(L) = \sum_k \sum_l \int_{x \in \mathcal{R}_l} L_{kl}\, p(x, C_k) dx.$$

# How can we make a trade-off?




| | $t = 1$ | $t = -1$ |
|---|---|---|
| $x > \hat{x}$ | $TP$ | $FP$ |
| $x < \hat{x}$ | $FN$ | $TN$ |
| | $P$ | $N$ |

# How can we make a trade-off?

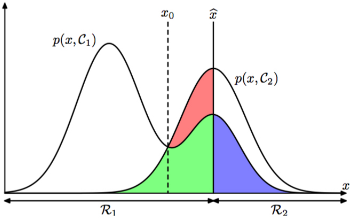



| | $t = 1$ | $t = -1$ |
|---|---|---|
| $x > \hat{x}$ | $TP$ | $FP$ |
| $x < \hat{x}$ | $FN$ | $TN$ |
| | $P$ | $N$ |

- We defined the decision threshold of a linear classifier as follows:

$$x_0 = \{x : f_{\mathrm{LIN}}(x) = 0\}$$

# How can we make a trade-off?




| | $t = 1$ | $t = -1$ |
|---|---|---|
| $x > \hat{x}$ | $TP$ | $FP$ |
| $x < \hat{x}$ | $FN$ | $TN$ |
| | $P$ | $N$ |

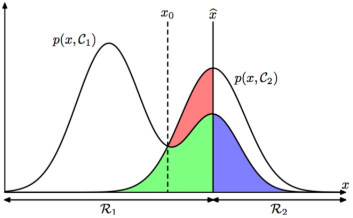
- We defined the decision threshold of a linear classifier as follows:

$$x_0 = \{x : f_{\mathrm{LIN}}(x) = 0\}$$

- We can decide to threshold at any another score $\alpha$:

$$x_\alpha = \{x : f(x) = \alpha\}$$

# How can we make a trade-off?




| | $t = 1$ | $t = -1$ |
|---|---|---|
| $x > \hat{x}$ | $TP$ | $FP$ |
| $x < \hat{x}$ | $FN$ | $TN$ |
| | $P$ | $N$ |

- We defined the decision threshold of a linear classifier as follows:

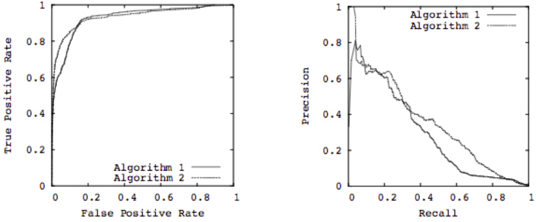
$$x_0 = \{x : f_{\mathrm{LIN}}(x) = 0\}$$

- We can decide to threshold at any another score $\alpha$:

$$x_\alpha = \{x : f(x) = \alpha\}$$

- What is the effect of chosing $\hat{x}$?

# Area under the curve (AUC)

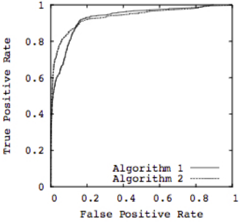



(a) Comparison in ROC space

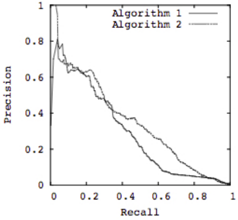
(b) Comparison in PR space

- AUC enables us to compare classifiers irrespective of the decision threshold.

# Area under the curve (AUC)



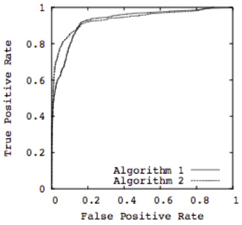

(a) Comparison in ROC space



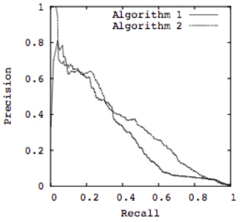

(b) Comparison in PR space

- AUC enables us to compare classifiers irrespective of the decision threshold.
- Receiver-operating characteristic (ROC):
  - Monotonic
  - AUC $\approx$ probability of scoring a positive higher than a negative.

# Area under the curve (AUC)




(a) Comparison in ROC space




(b) Comparison in PR space

- AUC enables us to compare classifiers irrespective of the decision threshold.
- Receiver-operating characteristic (ROC):
  - Monotonic
  - AUC $\approx$ probability of scoring a positive higher than a negative.
- Precision-recall:
  - Non-monotonic
  - One to one mapping with ROC

# How do we measure the performance in multi-class classification?

- Confusion matrix:

| | $t=1$ | $t=2$ | $t=3$ |
|---|---|---|---|
| $f(\boldsymbol{x})=1$ | $c_{11}$ | $c_{12}$ | $c_{13}$ |
| $f(\boldsymbol{x})=2$ | $c_{21}$ | $c_{22}$ | $c_{23}$ |
| $f(\boldsymbol{x})=3$ | $c_{31}$ | $c_{32}$ | $c_{33}$ |
| | $P_1$ | $P_2$ | $P_3$ |

# How do we measure the performance in multi-class classification?

- Confusion matrix:

| | $t=1$ | $t=2$ | $t=3$ |
|---|---|---|---|
| $f(\boldsymbol{x})=1$ | $c_{11}$ | $c_{12}$ | $c_{13}$ |
| $f(\boldsymbol{x})=2$ | $c_{21}$ | $c_{22}$ | $c_{23}$ |
| $f(\boldsymbol{x})=3$ | $c_{31}$ | $c_{32}$ | $c_{33}$ |
| | $P_1$ | $P_2$ | $P_3$ |

- What are the expressions of precision and recall?

# Outline

1. What is classification?
2. Decision theory
3. Generative classifiers
4. Discriminative classifiers
5. Summary
6. Exercises

# Generative classifiers

Consider the data set $\mathcal{D} = \{(\boldsymbol{x}_i, t_i) | i = 1, \ldots, n\}$. We are interested in the **posterior class probability**:

$$P(t = k|\boldsymbol{x}) = \frac{\overbrace{p(\boldsymbol{x}|t = k)}^{\text{class}-\text{conditional density}} \overbrace{P(t = k)}^{\text{class prior}}}{p(\boldsymbol{x})}, \qquad k = \{1, \ldots, m\}.$$

# Generative classifiers

Consider the data set $\mathcal{D} = \{(\boldsymbol{x}_i, t_i) | i = 1, \ldots, n\}$. We are interested in the **posterior class probability**:

$$P(t = k|\boldsymbol{x}) = \frac{\overbrace{p(\boldsymbol{x}|t = k)}^{\text{class-conditional density}} \overbrace{P(t = k)}^{\text{class prior}}}{p(\boldsymbol{x})}, \qquad k = \{1, \ldots, m\}.$$

- Prior: $P(t = k) = \pi_k$.
- Continuous features: $p(\boldsymbol{x}|t = k) = \text{Gaussian}\,(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$.

# Generative classifiers

Consider the data set $\mathcal{D} = \{(\boldsymbol{x}_i, t_i) | i = 1, \ldots, n\}$. We are interested in the **posterior class probability**:

$$P(t = k | \boldsymbol{x}) = \frac{\overbrace{p(\boldsymbol{x} | t = k)}^{\text{class-conditional density}} \; \overbrace{P(t = k)}^{\text{class prior}}}{p(\boldsymbol{x})}, \qquad k = \{1, \ldots, m\}.$$

- Prior: $P(t = k) = \pi_k$.
- Continuous features: $p(\boldsymbol{x} | t = k) = \mathrm{Gaussian}\,(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$.

How can we learn the parameters $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{m}$?

# Generative classifiers

Consider the data set $\mathcal{D} = \{(\boldsymbol{x}_i, t_i) | i = 1, \dots, n\}$. We are interested in the **posterior class probability**:

$$P(t = k|\boldsymbol{x}) = \frac{\overbrace{p(\boldsymbol{x}|t = k)}^{\text{class-conditional density}} \quad \overbrace{P(t = k)}^{\text{class prior}}}{p(\boldsymbol{x})}, \qquad k = \{1, \dots, m\}.$$

- Prior: $P(t = k) = \pi_k$.
- Continuous features: $p(\boldsymbol{x}|t = k) = \mathrm{Gaussian}\left(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)$.

How can we learn the parameters $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^m$?

$$\arg\max_{\boldsymbol{\theta}} \; \ln \prod_i p(t_i|\boldsymbol{\theta}) \quad \text{(maximum likelihood estimation)}$$

where $p(t_i|\boldsymbol{\theta}) = \mathrm{Categorical}\left(\pi_1 \mathrm{Gaussian}\left(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1\right), \dots, \pi_m \mathrm{Gaussian}\left(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\right)\right)$.

# Generative classifiers

Consider the data set $\mathcal{D} = \{(\boldsymbol{x}_i, t_i) | i = 1, \ldots, n\}$. We are interested in the **posterior class probability**:

$$P(t = k|\boldsymbol{x}) = \frac{\overbrace{p(\boldsymbol{x}|t = k)}^{\text{class−conditional density}} \overbrace{P(t = k)}^{\text{class prior}}}{p(\boldsymbol{x})}, \qquad k = \{1, \ldots, m\}.$$

- Prior: $P(t = k) = \pi_k$.
- Continuous features: $p(\boldsymbol{x}|t = k) = \mathrm{Gaussian}\left(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)$.
- Discrete features: $p(\boldsymbol{x}|t = k) = \mathrm{Multinomial}\left(\boldsymbol{\mu}_k\right)$.

How can we learn the parameters $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{m}$?

$$\arg\max_{\boldsymbol{\theta}} \ln \prod_i p(t_i|\boldsymbol{\theta}) \quad \text{(maximum likelihood estimation)}$$

where $p(t_i|\boldsymbol{\theta}) = \mathrm{Categorical}\left(\pi_1 \mathrm{Gaussian}\left(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1\right), \ldots, \pi_m \mathrm{Gaussian}\left(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\right)\right)$.
where $p(t_i|\boldsymbol{\theta}) = \mathrm{Categorical}\left(\pi_1 \mathrm{Multinomial}\left(\boldsymbol{\mu}_1\right), \ldots, \pi_m \mathrm{Multinomial}\left(\boldsymbol{\mu}_m\right)\right)$.

# Definitions

Multivariate Gaussian probability density:

$$\boldsymbol{x} \sim \mathrm{Gaussian}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}.$$

Multinomial probability distribution:

$$\boldsymbol{x} \sim \mathrm{Multinomial}\left(\boldsymbol{\mu}\right) = \frac{(\sum_j x_j)!}{\prod_j x_j!} \prod_{j=1}^{d} \mu_j^{x_j}.$$

Categorical probability distribution:

$$t \sim \mathrm{Categorical}\left(\boldsymbol{p}\right) = \prod_{k=1}^{m} p_k^{\delta_k(t)},$$

where $\delta_z(\cdot)$ is the kronecker delta centred at $z$.

# Binary classification

$$P(t=1|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|t=1)P(t=1)}{\sum_k p(\boldsymbol{x}|t=k)P(t=k)}$$

# Binary classification

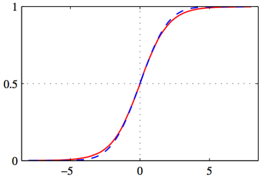
$$
\begin{aligned}
P(t=1|\boldsymbol{x}) &= \frac{p(\boldsymbol{x}|t=1)P(t=1)}{\sum_k p(\boldsymbol{x}|t=k)P(t=k)} \\
&= \frac{1}{1+\frac{p(\boldsymbol{x}|t=-1)P(t=-1)}{p(\boldsymbol{x}|t=1)P(t=1)}}
\end{aligned}
$$

# Binary classification

$$
\begin{aligned}
P(t=1|\boldsymbol{x}) &= \frac{p(\boldsymbol{x}|t=1)P(t=1)}{\sum_k p(\boldsymbol{x}|t=k)P(t=k)} \\
&= \frac{1}{1+\frac{p(\boldsymbol{x}|t=-1)P(t=-1)}{p(\boldsymbol{x}|t=1)P(t=1)}} \\
&= \frac{1}{1+e^{-\ln\frac{p(\boldsymbol{x}|t=1)P(t=1)}{p(\boldsymbol{x}|t=-1)P(t=-1)}}}
\end{aligned}
$$

# Binary classification

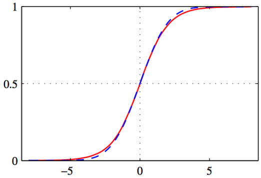
$$P(t=1|\mathbf{x}) = \frac{p(\mathbf{x}|t=1)P(t=1)}{\sum_k p(\mathbf{x}|t=k)P(t=k)}$$
$$= \frac{1}{1+\frac{p(\mathbf{x}|t=-1)P(t=-1)}{p(\mathbf{x}|t=1)P(t=1)}}$$
$$= \frac{1}{1+e^{-\ln\frac{p(\mathbf{x}|t=1)P(t=1)}{p(\mathbf{x}|t=-1)P(t=-1)}}}$$




Sigmoid: $\sigma(z) = \frac{1}{1+\exp(-z)}$.

# Binary classification

$$P(t=1|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|t=1)P(t=1)}{\sum_k p(\boldsymbol{x}|t=k)P(t=k)}$$
$$= \frac{1}{1+\frac{p(\boldsymbol{x}|t=-1)P(t=-1)}{p(\boldsymbol{x}|t=1)P(t=1)}}$$
$$= \frac{1}{1+e^{-\ln\frac{p(\boldsymbol{x}|t=1)P(t=1)}{p(\boldsymbol{x}|t=-1)P(t=-1)}}}$$




Sigmoid: $\sigma(z) = \frac{1}{1+\exp(-z)}$.

Let $\alpha \in [0,1]$. The classifier is defined as follows:

$$f(\boldsymbol{x}) = \begin{cases} +1 & \text{if } P(t=1|\boldsymbol{x}) > \alpha, \\ -1 & \text{if } P(t=1|\boldsymbol{x}) < \alpha. \end{cases}$$

# Binary classification with continuous features

- The classifier is defined by the following priors and class-conditionals:

$$\begin{aligned} P(t=1) &= \pi, & p(\boldsymbol{x}|t=1) &= \mathrm{Gaussian}\left(\boldsymbol{\mu}_{+1}, \boldsymbol{\Sigma}\right), \\ P(t=-1) &= 1-\pi, & p(\boldsymbol{x}|t=-1) &= \mathrm{Gaussian}\left(\boldsymbol{\mu}_{-1}, \boldsymbol{\Sigma}\right). \end{aligned}$$

where the classes are assumed to share the **same** covariance.

# Binary classification with continuous features

- The classifier is defined by the following priors and class-conditionals:

$$
\begin{aligned}
P(t=1) &= \pi, & p(\boldsymbol{x}|t=1) &= \mathrm{Gaussian}\left(\boldsymbol{\mu}_{+1}, \boldsymbol{\Sigma}\right), \\
P(t=-1) &= 1-\pi, & p(\boldsymbol{x}|t=-1) &= \mathrm{Gaussian}\left(\boldsymbol{\mu}_{-1}, \boldsymbol{\Sigma}\right).
\end{aligned}
$$

  where the classes are assumed to share the **same** covariance.
- The log-likelihood is given by

$$
\begin{aligned}
\ln p(\boldsymbol{t}|\boldsymbol{\theta}) = {} & \sum_i \delta_{+1}(t_i)\left(\ln \pi + \ln \mathrm{Gaussian}\left(\boldsymbol{\mu}_{+1}, \boldsymbol{\Sigma}\right)\right) \\
& + \sum_i \delta_{-1}(t_i)\left(\ln(1-\pi) + \ln \mathrm{Gaussian}\left(\boldsymbol{\mu}_{-1}, \boldsymbol{\Sigma}\right)\right).
\end{aligned}
$$

# Binary classification with continuous features

- The classifier is defined by the following priors and class-conditionals:

$$
\begin{aligned}
P(t=1) &= \pi, & p(\boldsymbol{x}|t=1) &= \mathrm{Gaussian}\left(\boldsymbol{\mu}_{+1}, \boldsymbol{\Sigma}\right), \\
P(t=-1) &= 1-\pi, & p(\boldsymbol{x}|t=-1) &= \mathrm{Gaussian}\left(\boldsymbol{\mu}_{-1}, \boldsymbol{\Sigma}\right).
\end{aligned}
$$

  where the classes are assumed to share the **same** covariance.
- The log-likelihood is given by

$$
\begin{aligned}
\ln p(\boldsymbol{t}|\boldsymbol{\theta}) = &\sum_i \delta_{+1}(t_i)\left(\ln \pi + \ln \mathrm{Gaussian}\left(\boldsymbol{\mu}_{+1}, \boldsymbol{\Sigma}\right)\right) \\
&+ \sum_i \delta_{-1}(t_i)\left(\ln(1-\pi) + \ln \mathrm{Gaussian}\left(\boldsymbol{\mu}_{-1}, \boldsymbol{\Sigma}\right)\right).
\end{aligned}
$$

- Maximum likelihood solution:

  $\pi = \frac{\sum_i \delta_{+1}(t_i)}{n}$ ,

# Binary classification with continuous features

- The classifier is defined by the following priors and class-conditionals:

$$
\begin{aligned}
P(t=1) &= \pi, & p(\boldsymbol{x}|t=1) &= \mathrm{Gaussian}\left(\boldsymbol{\mu}_{+1}, \boldsymbol{\Sigma}\right), \\
P(t=-1) &= 1-\pi, & p(\boldsymbol{x}|t=-1) &= \mathrm{Gaussian}\left(\boldsymbol{\mu}_{-1}, \boldsymbol{\Sigma}\right).
\end{aligned}
$$

  where the classes are assumed to share the **same** covariance.

- The log-likelihood is given by

$$
\begin{aligned}
\ln p(\boldsymbol{t}|\boldsymbol{\theta}) = & \sum_i \delta_{+1}(t_i)\left(\ln \pi + \ln \mathrm{Gaussian}\left(\boldsymbol{\mu}_{+1}, \boldsymbol{\Sigma}\right)\right) \\
& + \sum_i \delta_{-1}(t_i)\left(\ln(1-\pi) + \ln \mathrm{Gaussian}\left(\boldsymbol{\mu}_{-1}, \boldsymbol{\Sigma}\right)\right).
\end{aligned}
$$

- Maximum likelihood solution:

  $\pi = \frac{\sum_i \delta_{+1}(t_i)}{n}$ , $\boldsymbol{\mu}_{+1} = \frac{1}{n_{+1}} \sum_{i=1}^{N} \delta_{+1}(t_i)\boldsymbol{x}_i$ ,

# Binary classification with continuous features

- The classifier is defined by the following priors and class-conditionals:

$$\begin{aligned} P(t=1) &= \pi, & p(\mathbf{x}|t=1) &= \mathrm{Gaussian}\left(\boldsymbol{\mu}_{+1}, \boldsymbol{\Sigma}\right), \\ P(t=-1) &= 1-\pi, & p(\mathbf{x}|t=-1) &= \mathrm{Gaussian}\left(\boldsymbol{\mu}_{-1}, \boldsymbol{\Sigma}\right). \end{aligned}$$

  where the classes are assumed to share the **same** covariance.

- The log-likelihood is given by

$$\begin{aligned} \ln p(\mathbf{t}|\boldsymbol{\theta}) = &\sum_i \delta_{+1}(t_i)\left(\ln \pi + \ln \mathrm{Gaussian}\left(\boldsymbol{\mu}_{+1}, \boldsymbol{\Sigma}\right)\right) \\ &+ \sum_i \delta_{-1}(t_i)\left(\ln(1-\pi) + \ln \mathrm{Gaussian}\left(\boldsymbol{\mu}_{-1}, \boldsymbol{\Sigma}\right)\right). \end{aligned}$$

- Maximum likelihood solution:

  $\pi = \frac{\sum_i \delta_{+1}(t_i)}{n}$ , $\boldsymbol{\mu}_{+1} = \frac{1}{n_{+1}} \sum_{i=1}^{N} \delta_{+1}(t_i)\mathbf{x}_i$ , $\boldsymbol{\mu}_{-1} = \frac{1}{n_{-1}} \sum_{i=1}^{N} \delta_{-1}(t_i)\mathbf{x}_i$ ,

# Binary classification with continuous features

- The classifier is defined by the following priors and class-conditionals:

$$P(t=1)=\pi, \qquad p(\boldsymbol{x}|t=1)=\mathrm{Gaussian}\left(\boldsymbol{\mu}_{+1},\boldsymbol{\Sigma}\right),$$
$$P(t=-1)=1-\pi, \qquad p(\boldsymbol{x}|t=-1)=\mathrm{Gaussian}\left(\boldsymbol{\mu}_{-1},\boldsymbol{\Sigma}\right).$$

  where the classes are assumed to share the **same** covariance.

- The log-likelihood is given by

$$\begin{aligned}\ln p(\boldsymbol{t}|\boldsymbol{\theta}) &= \sum_i \delta_{+1}(t_i)\left(\ln\pi+\ln\mathrm{Gaussian}\left(\boldsymbol{\mu}_{+1},\boldsymbol{\Sigma}\right)\right)\\ &\quad+\sum_i \delta_{-1}(t_i)\left(\ln(1-\pi)+\ln\mathrm{Gaussian}\left(\boldsymbol{\mu}_{-1},\boldsymbol{\Sigma}\right)\right).\end{aligned}$$

- Maximum likelihood solution:

  $\pi=\frac{\sum_i \delta_{+1}(t_i)}{n}$ , $\boldsymbol{\mu}_{+1}=\frac{1}{n_{+1}}\sum_{i=1}^N \delta_{+1}(t_i)\boldsymbol{x}_i$ , $\boldsymbol{\mu}_{-1}=\frac{1}{n_{-1}}\sum_{i=1}^N \delta_{-1}(t_i)\boldsymbol{x}_i$ ,

  $\boldsymbol{\Sigma}=\frac{n_{+1}}{n}\boldsymbol{S}_{+1}+\frac{n_{-1}}{n}\boldsymbol{S}_{-1}$,

# Binary classification with continuous features

- The classifier is defined by the following priors and class-conditionals:

$$P(t=1) = \pi, \qquad p(\boldsymbol{x}|t=1) = \mathrm{Gaussian}\left(\boldsymbol{\mu}_{+1}, \boldsymbol{\Sigma}\right),$$
$$P(t=-1) = 1-\pi, \qquad p(\boldsymbol{x}|t=-1) = \mathrm{Gaussian}\left(\boldsymbol{\mu}_{-1}, \boldsymbol{\Sigma}\right).$$

  where the classes are assumed to share the **same** covariance.

- The log-likelihood is given by

$$\ln p(\boldsymbol{t}|\boldsymbol{\theta}) = \sum_i \delta_{+1}(t_i)\left(\ln \pi + \ln \mathrm{Gaussian}\left(\boldsymbol{\mu}_{+1}, \boldsymbol{\Sigma}\right)\right) + \sum_i \delta_{-1}(t_i)\left(\ln(1-\pi) + \ln \mathrm{Gaussian}\left(\boldsymbol{\mu}_{-1}, \boldsymbol{\Sigma}\right)\right).$$

- Maximum likelihood solution:

  $\pi = \frac{\sum_i \delta_{+1}(t_i)}{n}$ , $\boldsymbol{\mu}_{+1} = \frac{1}{n_{+1}} \sum_{i=1}^N \delta_{+1}(t_i) \boldsymbol{x}_i$ , $\boldsymbol{\mu}_{-1} = \frac{1}{n_{-1}} \sum_{i=1}^N \delta_{-1}(t_i) \boldsymbol{x}_i$ ,

  $\boldsymbol{\Sigma} = \frac{n_{+1}}{n} \boldsymbol{S}_{+1} + \frac{n_{-1}}{n} \boldsymbol{S}_{-1}$, $\boldsymbol{S}_{\pm 1} = \frac{1}{n_{\pm 1}} \sum_{i=1}^N \delta_{\pm 1}(t_i)(\boldsymbol{x}_i - \boldsymbol{\mu}_{\pm 1})(\boldsymbol{x}_i - \boldsymbol{\mu}_{\pm 1})^\top$ .

# What is the form of the decision boundary?

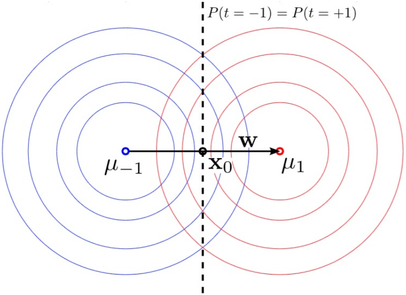
Plugging the priors and the class-conditionals in the posterior leads to

$$P(t=1|\boldsymbol{x}) = \sigma\left((\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1})^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + -\frac{1}{2}\boldsymbol{\mu}_{+1}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_{+1} + \frac{1}{2}\boldsymbol{\mu}_{-1}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_{-1} + \ln\frac{\pi}{1-\pi}\right)$$

# What is the form of the decision boundary?

Plugging the priors and the class-conditionals in the posterior leads to

$$
\begin{aligned}
P(t=1|\boldsymbol{x}) &= \sigma\left((\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1})^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} + -\frac{1}{2}\boldsymbol{\mu}_{+1}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_{+1} + \frac{1}{2}\boldsymbol{\mu}_{-1}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_{-1} + \ln\frac{\pi}{1-\pi}\right) \\
&= \sigma\left(\boldsymbol{w}^\top\boldsymbol{x} + b\right).
\end{aligned}
$$



# Binary classification with discrete features

- The classifier is defined by the following priors and class-conditionals:

$$P(t=1) = \pi, \qquad p(\boldsymbol{x}|t=1) = \mathrm{Multinomial}\left(\boldsymbol{\mu}_{+1}\right),$$
$$P(t=-1) = 1-\pi, \qquad p(\boldsymbol{x}|t=-1) = \mathrm{Multinomial}\left(\boldsymbol{\mu}_{-1}\right).$$

where the features are assumed to be **independent** given the class.

# Binary classification with discrete features

- The classifier is defined by the following priors and class-conditionals:

$$P(t=1)=\pi, \qquad p(\boldsymbol{x}|t=1)=\mathrm{Multinomial}\left(\boldsymbol{\mu}_{+1}\right),$$
$$P(t=-1)=1-\pi, \qquad p(\boldsymbol{x}|t=-1)=\mathrm{Multinomial}\left(\boldsymbol{\mu}_{-1}\right).$$

  where the features are assumed to be **independent** given the class.

- The log-likelihood is given by

$$\begin{aligned}\ln p(\boldsymbol{t}|\boldsymbol{\theta}) = &\sum_i \delta_{+1}(t_i)\left(\ln\pi + \ln\mathrm{Multinomial}\left(\boldsymbol{\mu}_{+1}\right)\right)\\ &+ \sum_i \delta_{-1}(t_i)\left(\ln(1-\pi) + \ln\mathrm{Multinomial}\left(\boldsymbol{\mu}_{-1}\right)\right).\end{aligned}$$

# Binary classification with discrete features

- The classifier is defined by the following priors and class-conditionals:

$$P(t = 1) = \pi, \qquad p(\boldsymbol{x}|t = 1) = \mathrm{Multinomial}\left(\boldsymbol{\mu}_{+1}\right),$$
$$P(t = -1) = 1 - \pi, \qquad p(\boldsymbol{x}|t = -1) = \mathrm{Multinomial}\left(\boldsymbol{\mu}_{-1}\right).$$

  where the features are assumed to be **independent** given the class.

- The log-likelihood is given by

$$\begin{aligned} \ln p(\boldsymbol{t}|\boldsymbol{\theta}) = &\sum_i \delta_{+1}(t_i)\left(\ln \pi + \ln \mathrm{Multinomial}\left(\boldsymbol{\mu}_{+1}\right)\right) \\ &+ \sum_i \delta_{-1}(t_i)\left(\ln(1-\pi) + \ln \mathrm{Multinomial}\left(\boldsymbol{\mu}_{-1}\right)\right). \end{aligned}$$

- Maximum likelihood solution:

  $\pi = \frac{\sum_i \sum_j \delta_{+1}(t_i) x_{ij}}{\sum_i \sum_j x_{ij}}$ ,

# Binary classification with discrete features

- The classifier is defined by the following priors and class-conditionals:

$$P(t=1)=\pi, \qquad p(\boldsymbol{x}|t=1)=\mathrm{Multinomial}\left(\boldsymbol{\mu}_{+1}\right),$$
$$P(t=-1)=1-\pi, \qquad p(\boldsymbol{x}|t=-1)=\mathrm{Multinomial}\left(\boldsymbol{\mu}_{-1}\right).$$

  where the features are assumed to be **independent** given the class.

- The log-likelihood is given by

$$\begin{aligned}\ln p(\boldsymbol{t}|\boldsymbol{\theta}) &= \sum_i \delta_{+1}(t_i)\left(\ln\pi + \ln \mathrm{Multinomial}\left(\boldsymbol{\mu}_{+1}\right)\right) \\ &+ \sum_i \delta_{-1}(t_i)\left(\ln(1-\pi) + \ln \mathrm{Multinomial}\left(\boldsymbol{\mu}_{-1}\right)\right).\end{aligned}$$

- Maximum likelihood solution:

  $\pi = \frac{\sum_i \sum_j \delta_{+1}(t_i) x_{ij}}{\sum_i \sum_j x_{ij}}$ ,

  $\boldsymbol{\mu}_{+1} = \frac{\sum_i \delta_{+1}(t_i)\boldsymbol{x}_i}{\sum_i \sum_j \delta_{+1}(t_i) x_{ij}}$ ,

# Binary classification with discrete features

- The classifier is defined by the following priors and class-conditionals:

$$P(t=1) = \pi, \qquad p(\boldsymbol{x}|t=1) = \mathrm{Multinomial}\left(\boldsymbol{\mu}_{+1}\right),$$
$$P(t=-1) = 1-\pi, \qquad p(\boldsymbol{x}|t=-1) = \mathrm{Multinomial}\left(\boldsymbol{\mu}_{-1}\right).$$

  where the features are assumed to be **independent** given the class.

- The log-likelihood is given by

$$\ln p(\boldsymbol{t}|\boldsymbol{\theta}) = \sum_i \delta_{+1}(t_i)\left(\ln\pi + \ln \mathrm{Multinomial}\left(\boldsymbol{\mu}_{+1}\right)\right) + \sum_i \delta_{-1}(t_i)\left(\ln(1-\pi) + \ln \mathrm{Multinomial}\left(\boldsymbol{\mu}_{-1}\right)\right).$$

- Maximum likelihood solution:

  $\pi = \dfrac{\sum_i \sum_j \delta_{+1}(t_i) x_{ij}}{\sum_i \sum_j x_{ij}}$ ,

  $\boldsymbol{\mu}_{+1} = \dfrac{\sum_i \delta_{+1}(t_i)\boldsymbol{x}_i}{\sum_i \sum_j \delta_{+1}(t_i) x_{ij}}$ , $\boldsymbol{\mu}_{-1} \dfrac{\sum_i \delta_{-1}(t_i)\boldsymbol{x}_i}{\sum_i \sum_j \delta_{-1}(t_i) x_{ij}}$ .

# What is the form of the decision boundary?

Plugging the priors and the class-conditionals in the posterior leads to

$$P(t = 1|\boldsymbol{x}) = \sigma\left((\ln \boldsymbol{\mu}_{+1} - \ln \boldsymbol{\mu}_{-1})^\top \boldsymbol{x} + \ln \frac{\pi}{1-\pi}\right)$$

# What is the form of the decision boundary?

Plugging the priors and the class-conditionals in the posterior leads to

$$
\begin{aligned}
P(t = 1|\boldsymbol{x}) &= \sigma\left( (\ln \boldsymbol{\mu}_{+1} - \ln \boldsymbol{\mu}_{-1})^\top \boldsymbol{x} + \ln \frac{\pi}{1-\pi} \right) \\
&= \sigma\left( \boldsymbol{w}^\top \boldsymbol{x} + b \right).
\end{aligned}
$$

# What is the form of the decision boundary?

Plugging the priors and the class-conditionals in the posterior leads to

$$\begin{aligned} P(t=1|\boldsymbol{x}) &= \sigma\left( (\ln \boldsymbol{\mu}_{+1} - \ln \boldsymbol{\mu}_{-1})^\top \boldsymbol{x} + \ln \frac{\pi}{1-\pi} \right) \\ &= \sigma\left( \boldsymbol{w}^\top \boldsymbol{x} + b \right). \end{aligned}$$

What if an entry of $\boldsymbol{\mu}_{\pm 1}$ is zero? When can this occur?

# What is the form of the decision boundary?

Plugging the priors and the class-conditionals in the posterior leads to

$$
\begin{aligned}
P(t=1|\boldsymbol{x}) &= \sigma\left((\ln \boldsymbol{\mu}_{+1} - \ln \boldsymbol{\mu}_{-1})^\top \boldsymbol{x} + \ln \frac{\pi}{1-\pi}\right) \\
&= \sigma\left(\boldsymbol{w}^\top \boldsymbol{x} + b\right).
\end{aligned}
$$

What if an entry of $\boldsymbol{\mu}_{\pm 1}$ is zero? When can this occur?

$$
\arg\max_{\boldsymbol{\theta}} \ \ln \prod_i p(t_i|\boldsymbol{\theta}) + \ln p(\boldsymbol{\theta}) \quad \text{(maximum a posteriori estimation)}
$$

where $p(\boldsymbol{\theta}) = \mathrm{Dirichlet}\,(\alpha \mathbf{1})$:

# What is the form of the decision boundary?

Plugging the priors and the class-conditionals in the posterior leads to

$$\begin{aligned} P(t=1|\boldsymbol{x}) &= \sigma\left((\ln \boldsymbol{\mu}_{+1} - \ln \boldsymbol{\mu}_{-1})^\top \boldsymbol{x} + \ln \frac{\pi}{1-\pi}\right) \\ &= \sigma\left(\boldsymbol{w}^\top \boldsymbol{x} + b\right). \end{aligned}$$

What if an entry of $\boldsymbol{\mu}_{\pm 1}$ is zero? When can this occur?

$$\arg\max_{\boldsymbol{\theta}} \; \ln \prod_i p(t_i|\boldsymbol{\theta}) + \ln p(\boldsymbol{\theta}) \quad \text{(maximum a posteriori estimation)}$$

where $p(\boldsymbol{\theta}) = \mathrm{Dirichlet}\,(\alpha \mathbf{1})$:

- Parameter $\alpha$ can be interpreted as a pseudo-count. $(\star)$
- Adding a prior is equivalent to regularisation.

# Naive Bayes classifier

- Generative classifier making the simplifying assumption that features are independent given the class:

$$P(t = k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|t = k)P(t = k)}{p(\boldsymbol{x})} \approx \frac{\prod_{j=1}^{d} p(x_j|t = k)P(t = k)}{p(\boldsymbol{x})}.$$

# Naive Bayes classifier

- Generative classifier making the simplifying assumption that features are independent given the class:

$$P(t = k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|t = k)P(t = k)}{p(\boldsymbol{x})} \approx \frac{\prod_{j=1}^{d} p(x_j|t = k)P(t = k)}{p(\boldsymbol{x})}.$$

- Number of parameters scales linearly with the number of features!

# Naive Bayes classifier

- Generative classifier making the simplifying assumption that features are independent given the class:

$$P(t=k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|t=k)P(t=k)}{p(\boldsymbol{x})} \approx \frac{\prod_{j=1}^{d} p(x_j|t=k)P(t=k)}{p(\boldsymbol{x})}.$$

- Number of parameters scales linearly with the number of features!

REUTERS

You are here: Home > News > Science > Article

Go to a Section: U.S. International Business Markets Politics Entertainment Technology Sports Oddly Enough

**Extreme conditions create rare Antarctic clouds**

Tue Aug 1, 2006 3:20am ET

Email This Article | Print This Article | Reprints

[-] Text [+]

SYDNEY (Reuters) - Rare, mother-of-pearl colored clouds caused by extreme weather conditions above Antarctica are a possible indication of global warming, Australian scientists said on Tuesday.

Known as nacreous clouds, the spectacular formations showing delicate wisps of colors were photographed in the sky over an Australian meteorological base at Mawson Station on July 25.

Figure 6.8: The Reuters RCV1 collection is a set of 800,000 documents (news articles), with about 200 words per document on average. After standard pre-processing (stop word removal), its dictionary (set of distinct words) is roughly of size 400,000. A common machine learning problem associated with this data is to classify documents into groups (for example: politics, business, sports, science, movies), which are often organized in a hierarchical fashion.

# Naive Bayes classifier

- Generative classifier making the simplifying assumption that features are independent given the class:

$$P(t=k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|t=k)P(t=k)}{p(\boldsymbol{x})} \approx \frac{\prod_{j=1}^{d} p(x_j|t=k)P(t=k)}{p(\boldsymbol{x})}.$$

- Number of parameters scales linearly with the number of features!


REUTERS

You are here: Home > News > Science > Article

Go to a Section: U.S. International Business Markets Politics Entertainment Technology Sports Oddly Enough

**Extreme conditions create rare Antarctic clouds**

Tue Aug 1, 2006 3:20am ET

Email This Article | Print This Article | Reprints

[-] Text [+]

SYDNEY (Reuters) - Rare, mother-of-pearl colored clouds caused by extreme weather conditions above Antarctica are a possible indication of global warming, Australian scientists said on Tuesday.

Known as nacreous clouds, the spectacular formations showing delicate wisps of colors were photographed in the sky over an Australian meteorological base at Mawson Station on July 25.

Figure 6.8: The Reuters RCV1 collection is a set of 800,000 documents (news articles), with about 200 words per document on average. After standard pre-processing (stop word removal), its dictionary (set of distinct words) is roughly of size 400,000. A common machine learning problem associated with this data is to classify documents into groups (for example: politics, business, sports, science, movies), which are often organized in a hierarchical fashion.

Document categorisation:

$$P(t=k) = \pi_k,$$
$$p(\boldsymbol{x}|t=k) = \text{Multinomial}(\boldsymbol{\mu}_k).$$

# Naive Bayes classifier

- Generative classifier making the simplifying assumption that features are independent given the class:

$$P(t=k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|t=k)P(t=k)}{p(\boldsymbol{x})} \approx \frac{\prod_{j=1}^{d} p(x_j|t=k)P(t=k)}{p(\boldsymbol{x})}.$$

- Number of parameters scales linearly with the number of features!


Figure 6.8: The Reuters RCV1 collection is a set of 800,000 documents (news articles), with about 200 words per document on average. After standard pre-processing (stop word removal), its dictionary (set of distinct words) is roughly of size 400,000. A common machine learning problem associated with this data is to classify documents into groups (for example: politics, business, sports, science, movies), which are often organized in a hierarchical fashion.

Document categorisation:

$$P(t=k) = \pi_k,$$
$$p(\boldsymbol{x}|t=k) = \mathrm{Multinomial}\left(\boldsymbol{\mu}_k\right).$$

- Category/theme/topic $k$ is modelled by a discrete distribution $\boldsymbol{\mu}_k$ over the vocabulary of size $d$.

# Naive Bayes classifier

- Generative classifier making the simplifying assumption that features are independent given the class:

$$P(t = k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|t = k)P(t = k)}{p(\boldsymbol{x})} \approx \frac{\prod_{j=1}^{d} p(x_j|t = k)P(t = k)}{p(\boldsymbol{x})}.$$

- Number of parameters scales linearly with the number of features!


REUTERS

You are here: Home > News > Science > Article

Go to a Section: U.S. International Business Markets Politics Entertainment Technology Sports Oddly Enough

**Extreme conditions create rare Antarctic clouds**

Tue Aug 1, 2006 3:20am ET

Email This Article | Print This Article | Reprints

[-] Text [+]

SYDNEY (Reuters) - Rare, mother-of-pearl colored clouds caused by extreme weather conditions above Antarctica are a possible indication of global warming, Australian scientists said on Tuesday.

Known as nacreous clouds, the spectacular formations showing delicate wisps of colors were photographed in the sky over an Australian meteorological base at Mawson Station on July 25.

Figure 6.8: The Reuters RCV1 collection is a set of 800,000 documents (news articles), with about 200 words per document on average. After standard pre-processing (stop word removal), its dictionary (set of distinct words) is roughly of size 400,000. A common machine learning problem associated with this data is to classify documents into groups (for example: politics, business, sports, science, movies), which are often organized in a hierarchical fashion.

Document categorisation:

$$P(t = k) = \pi_k,$$
$$p(\boldsymbol{x}|t = k) = \text{Multinomial}(\boldsymbol{\mu}_k).$$

- Category/theme/topic $k$ is modelled by a discrete distribution $\boldsymbol{\mu}_k$ over the vocabulary of size $d$.
- $\boldsymbol{x}_i$ represents document $i$; it contains the word counts.

# Maximum likelihood (ML) estimation: summary

The likelihood is the joint probability of observing i.i.d. data:

$$\ell(\boldsymbol{\theta}; \boldsymbol{t}) = \ln p(\boldsymbol{t}; \boldsymbol{\theta}) = \ln \prod_{i=1}^{n} p(t_i; \boldsymbol{\theta}).$$

# Maximum likelihood (ML) estimation: summary

The likelihood is the joint probability of observing i.i.d. data:

$$\ell(\boldsymbol{\theta}; \boldsymbol{t}) = \ln p(\boldsymbol{t}; \boldsymbol{\theta}) = \ln \prod_{i=1}^{n} p(t_i; \boldsymbol{\theta}).$$

The goal is to find the parameters that maximise the log-likelihood function:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; \boldsymbol{t}).$$

- ML leads to a point estimate of $\boldsymbol{\theta}$ and is asymptotically consistent.
- The likelihood is unbounded, so ML estimator can overfit!

# Maximum a posteriori (MAP) estimation: summary

Penalise unreasonable values ($\sim$ regularisation) by imposing a prior distribution on the parameters:

$$p(\boldsymbol{\theta}|\boldsymbol{X}) \propto p(\boldsymbol{X}|\boldsymbol{\theta})p(\boldsymbol{\theta}).$$

# Maximum a posteriori (MAP) estimation: summary

Penalise unreasonable values ($\sim$ regularisation) by imposing a prior distribution on the parameters:

$$p(\boldsymbol{\theta}|\boldsymbol{X}) \propto p(\boldsymbol{X}|\boldsymbol{\theta})p(\boldsymbol{\theta}).$$

The goal is to maximise the penalised log-likelihood :

$$\ell_{\mathrm{MAP}}(\boldsymbol{\theta}; \boldsymbol{t}) = \ell(\boldsymbol{\theta}; \boldsymbol{t}) + \ln p(\boldsymbol{\theta}).$$

- MAP leads to a point estimate of $\boldsymbol{\theta}$ and asymptotically agrees with ML estimate.
- MAP is not invariant under reparametrisation!

# Outline


1. What is classification?
2. Decision theory
3. Generative classifiers
4. Discriminative classifiers
5. Summary
6. Exercises

# Generative versus discriminative classification

$$f(\boldsymbol{x}) = \begin{cases} +1 & \text{if } P(t = 1|\boldsymbol{x}) > \alpha, \\ -1 & \text{if } P(t = 1|\boldsymbol{x}) < \alpha. \end{cases}$$

# Generative versus discriminative classification

$$f(\boldsymbol{x}) = \begin{cases} +1 & \text{if } P(t=1|\boldsymbol{x}) > \alpha, \\ -1 & \text{if } P(t=1|\boldsymbol{x}) < \alpha. \end{cases}$$

- Generative classifiers:

$$P(t=k|\boldsymbol{x}) \propto p(\boldsymbol{x}|t=k)P(t=k)$$

  - Require explicit class-conditionals
  - Take a linear form in specific cases

# Generative versus discriminative classification

$$f(\boldsymbol{x}) = \begin{cases} +1 & \text{if } P(t=1|\boldsymbol{x}) > \alpha, \\ -1 & \text{if } P(t=1|\boldsymbol{x}) < \alpha. \end{cases}$$

- Generative classifiers:

$$P(t=k|\boldsymbol{x}) \propto p(\boldsymbol{x}|t=k)P(t=k)$$

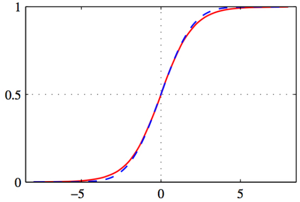
  - Require explicit class-conditionals
  - Take a linear form in specific cases

- Discriminative classifier:

$$P(t=k|\boldsymbol{x}) = \sigma(y(\boldsymbol{x};\boldsymbol{\theta}))$$

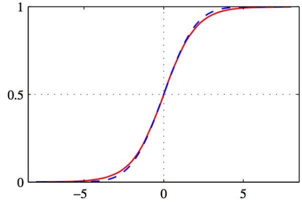
  - Does not rely on class-conditionals
  - Less parameters to learn (or optimise)
  - Easy to change the feature map $\phi(\boldsymbol{x})$

# (Binary) logistic regression




- Linear discriminant: $y(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x}) + b$ .

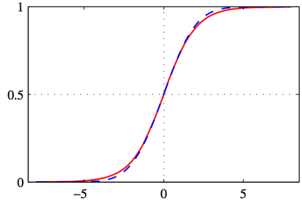
# (Binary) logistic regression




- Linear discriminant: $y(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x}) + b$ .
- Logistic link:

$$P(t = +1|\boldsymbol{x}) = \sigma(y(\boldsymbol{x})) = \frac{1}{1 + \exp\left(-y(\boldsymbol{x})\right)}.$$

# (Binary) logistic regression



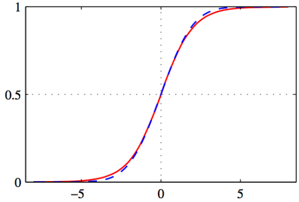

- Linear discriminant: $y(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x}) + b$ .
- Logistic link:

$$P(t = +1|\boldsymbol{x}) = \sigma(y(\boldsymbol{x})) = \frac{1}{1 + \exp\left(-y(\boldsymbol{x})\right)}.$$

- Conditional likelihood:

$$t|\boldsymbol{x} \sim \mathrm{Bernoulli}\left(\sigma\left(y(\boldsymbol{x})\right)\right) = \sigma\left(y(\boldsymbol{x})\right)^{\delta_{+1}(t)}\left(1 - \sigma\left(y(\boldsymbol{x})\right)\right)^{\delta_{-1}(t)}.$$

# (Binary) logistic regression




- Linear discriminant: $y(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x}) + b$ .
- Logistic link:

$$P(t = +1|\boldsymbol{x}) = \sigma(y(\boldsymbol{x})) = \frac{1}{1 + \exp\left(-y(\boldsymbol{x})\right)}.$$

- Conditional likelihood:

$$t|\boldsymbol{x} \sim \mathrm{Bernoulli}\left(\sigma\left(y(\boldsymbol{x})\right)\right) = \sigma\left(y(\boldsymbol{x})\right)^{\delta_{+1}(t)}\left(1 - \sigma\left(y(\boldsymbol{x})\right)\right)^{\delta_{-1}(t)}.$$

- Alternative formulation: $P(t|\boldsymbol{x}) = \sigma\left(ty(\boldsymbol{x})\right)$ . $(\star)$

# How do we learn $\boldsymbol{w}$?

$$\ln p(\boldsymbol{t}|\boldsymbol{x};\boldsymbol{w}) = \sum_i \ln \mathrm{Bernoulli}\left(\sigma(y(\boldsymbol{x}_i))\right).$$

# How do we learn $\boldsymbol{w}$?

$$\ln p(\boldsymbol{t}|\boldsymbol{x};\boldsymbol{w}) = \sum_i \ln \mathrm{Bernoulli}\left(\sigma(y(\boldsymbol{x}_i))\right).$$

1. Iterative reweighted least squares (IRLS):

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \underbrace{\left(\boldsymbol{\Phi}^\top \boldsymbol{R}\boldsymbol{\Phi}\right)^{-1}}_{=\boldsymbol{H}^{-1}} \underbrace{\boldsymbol{\Phi}^\top(\boldsymbol{\sigma} - \boldsymbol{t})}_{=\nabla_{\boldsymbol{w}} \ln p(\boldsymbol{t}|\boldsymbol{w})}$$

where $\boldsymbol{\Phi} = (\phi(\boldsymbol{x}_1)^\top, \ldots, \phi(\boldsymbol{x}_n)^\top)^\top$, $R_{ii} = \sigma(y(\boldsymbol{x}_i))\,(1 - \sigma(y(\boldsymbol{x}_i)))$, $\boldsymbol{\sigma} = (\sigma(y(\boldsymbol{x}_1)), \ldots, \sigma(y(\boldsymbol{x}_n)))^\top$ and $\boldsymbol{t} = (t_1, \ldots, t_n)^\top$.

# How do we learn $\boldsymbol{w}$?

$$\ln p(\boldsymbol{t}|\boldsymbol{x};\boldsymbol{w}) = \sum_i \ln \mathrm{Bernoulli}\left(\sigma(y(\boldsymbol{x}_i))\right).$$

1. Iterative reweighted least squares (IRLS):

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \underbrace{\left(\boldsymbol{\Phi}^\top \boldsymbol{R}\boldsymbol{\Phi}\right)^{-1}}_{=\boldsymbol{H}^{-1}} \underbrace{\boldsymbol{\Phi}^\top(\boldsymbol{\sigma} - \boldsymbol{t})}_{=\nabla_{\boldsymbol{w}} \ln p(\boldsymbol{t}|\boldsymbol{w})}$$

where $\boldsymbol{\Phi} = (\phi(\boldsymbol{x}_1)^\top, \ldots, \phi(\boldsymbol{x}_n)^\top)^\top$, $R_{ii} = \sigma(y(\boldsymbol{x}_i))\,(1 - \sigma(y(\boldsymbol{x}_i)))$, $\boldsymbol{\sigma} = (\sigma(y(\boldsymbol{x}_1)), \ldots, \sigma(y(\boldsymbol{x}_n)))^\top$ and $\boldsymbol{t} = (t_1, \ldots, t_n)^\top$.

- Instantiation of Newton-Raphson
- Objective is convex!

# How do we learn $\boldsymbol{w}$?

$$\ln p(\boldsymbol{t}|\boldsymbol{x};\boldsymbol{w}) = \sum_i \ln \mathrm{Bernoulli}\left(\sigma(y(\boldsymbol{x}_i))\right).$$

1. Iterative reweighted least squares (IRLS):

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \underbrace{\left(\boldsymbol{\Phi}^\top \boldsymbol{R}\boldsymbol{\Phi}\right)^{-1}}_{=\boldsymbol{H}^{-1}} \underbrace{\boldsymbol{\Phi}^\top(\boldsymbol{\sigma} - \boldsymbol{t})}_{=\nabla_{\boldsymbol{w}} \ln p(\boldsymbol{t}|\boldsymbol{w})}$$

   where $\boldsymbol{\Phi} = (\phi(\boldsymbol{x}_1)^\top, \ldots, \phi(\boldsymbol{x}_n)^\top)^\top$, $R_{ii} = \sigma(y(\boldsymbol{x}_i))\,(1 - \sigma(y(\boldsymbol{x}_i)))$, $\boldsymbol{\sigma} = (\sigma(y(\boldsymbol{x}_1)), \ldots, \sigma(y(\boldsymbol{x}_n)))^\top$ and $\boldsymbol{t} = (t_1, \ldots, t_n)^\top$.
   - Instantiation of Newton-Raphson
   - Objective is convex!
2. Alternatives include gradient descent

# How do we learn $\boldsymbol{w}$?

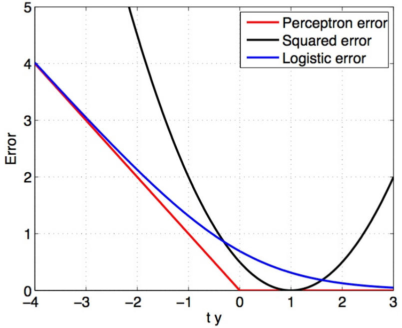
$$\ln p(\boldsymbol{t}|\boldsymbol{x};\boldsymbol{w}) = \sum_i \ln \mathrm{Bernoulli}\left(\sigma(y(\boldsymbol{x}_i))\right).$$

1. Iterative reweighted least squares (IRLS):

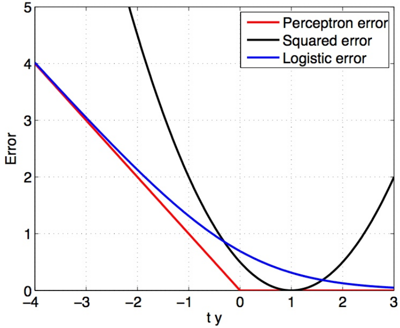
$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \underbrace{\left(\boldsymbol{\Phi}^\top \boldsymbol{R}\boldsymbol{\Phi}\right)^{-1}}_{=\boldsymbol{H}^{-1}} \underbrace{\boldsymbol{\Phi}^\top(\boldsymbol{\sigma} - \boldsymbol{t})}_{=\nabla_{\boldsymbol{w}} \ln p(\boldsymbol{t}|\boldsymbol{w})}$$

   where $\boldsymbol{\Phi} = (\phi(\boldsymbol{x}_1)^\top, \ldots, \phi(\boldsymbol{x}_n)^\top)^\top$, $R_{ii} = \sigma(y(\boldsymbol{x}_i))\,(1 - \sigma(y(\boldsymbol{x}_i)))$, $\boldsymbol{\sigma} = (\sigma(y(\boldsymbol{x}_1)), \ldots, \sigma(y(\boldsymbol{x}_n)))^\top$ and $\boldsymbol{t} = (t_1, \ldots, t_n)^\top$.
   - Instantiation of Newton-Raphson
   - Objective is convex!
2. Alternatives include gradient descent and stochastic gradient descent ($\star$)

# Classification losses



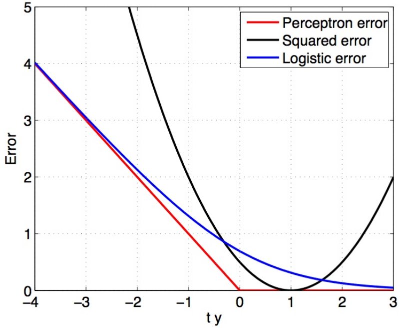
# Classification losses




- Perceptron loss:

$$E(\mathbf{x}_i) = \begin{cases} 0 & \text{if } t_i y(\mathbf{x}_i) > 0, \\ t_i y(\mathbf{x}_i) & \text{if } t_i y(\mathbf{x}_i) < 0. \end{cases}$$

# Classification losses



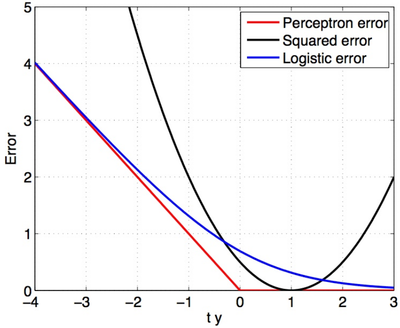

- Perceptron loss:

$$E(\boldsymbol{x}_i) = \begin{cases} 0 & \text{if } t_i y(\boldsymbol{x}_i) > 0, \\ t_i y(\boldsymbol{x}_i) & \text{if } t_i y(\boldsymbol{x}_i) < 0. \end{cases}$$

- Logistic loss:

$$E(\boldsymbol{x}_i) = \ln\left(1 + e^{-t_i y(\boldsymbol{x}_i)}\right).$$

# Classification losses




- Perceptron loss:

$$E(\boldsymbol{x}_i) = \begin{cases} 0 & \text{if } t_i y(\boldsymbol{x}_i) > 0, \\ t_i y(\boldsymbol{x}_i) & \text{if } t_i y(\boldsymbol{x}_i) < 0. \end{cases}$$

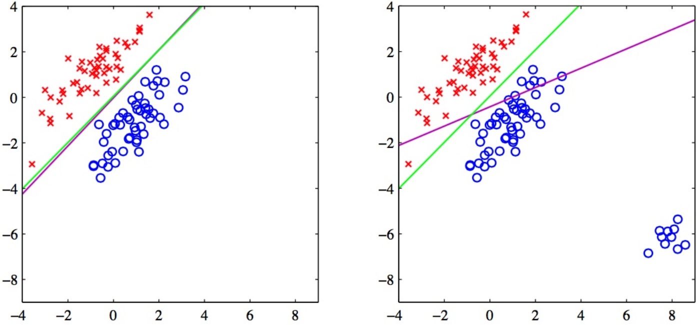
- Logistic loss:

$$E(\boldsymbol{x}_i) = \ln\left(1 + e^{-t_i y(\boldsymbol{x}_i)}\right).$$

- Squared error:

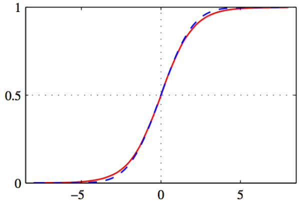
$$E(x_i) = \frac{1}{2}\left(t_i y(\boldsymbol{x_i}) - 1\right)^2.$$

# Is the squared error suitable for classification?




(Green: perceptron. Magenta: squared error.)

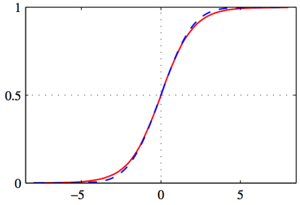
# Other link functions?




- Probit regression:

$$\Phi(y(\boldsymbol{x})) = \int_{-\inf}^{y(\boldsymbol{x})} \text{Gaussian}\,(0,1)\, dz, \quad y(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b.$$

# Other link functions?




- Probit regression:

$$\Phi(y(\boldsymbol{x})) = \int_{-\text{inf}}^{y(\boldsymbol{x})} \text{Gaussian}\,(0,1)\, dz, \quad y(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b.$$

- Latent variable view:

$$t|z \sim I(tz > 0), \qquad z \sim \text{Gaussian}\,(\boldsymbol{y}(\boldsymbol{x}), 1)\,.$$

# Multinomial logistic regression

- Linear discriminant:

$$y_k(\boldsymbol{x}) = \boldsymbol{w}_k^\top \phi(\boldsymbol{x}) + b_k, \quad k \in \{1, \ldots, m\}.$$

# Multinomial logistic regression

- Linear discriminant:

$$y_k(\boldsymbol{x}) = \boldsymbol{w}_k^\top \phi(\boldsymbol{x}) + b_k, \quad k \in \{1, \ldots, m\}.$$

- Softmax:

$$P(t = k|x) = \frac{\exp\left(y(\boldsymbol{x}_k)\right)}{\sum_l \exp\left(y(\boldsymbol{x}_l)\right)}.$$

# Multinomial logistic regression

- Linear discriminant:

$$y_k(\boldsymbol{x}) = \boldsymbol{w}_k^\top \phi(\boldsymbol{x}) + b_k, \quad k \in \{1, \ldots, m\}.$$

- Softmax:

$$P(t = k|x) = \frac{\exp(y(\boldsymbol{x}_k))}{\sum_l \exp(y(\boldsymbol{x}_l))}.$$

- Conditional likelihood:

$$t|\boldsymbol{x} \sim \mathrm{Categorical}(\boldsymbol{\mu}),$$

where $\mu_k = P(t = k|x)$.

# Outline


1. What is classification?
2. Decision theory
3. Generative classifiers
4. Discriminative classifiers
5. Summary
6. Exercises

# Summary

- Linear classifiers:
  - Perceptron
  - Naive Bayes
  - (Multi-nomial) logistic regression

# Summary

- Linear classifiers:
  - Perceptron
  - Naive Bayes
  - (Multi-nomial) logistic regression
- Linear classifier can be non-linear!

# Summary

- Linear classifiers:
  - Perceptron
  - Naive Bayes
  - (Multi-nomial) logistic regression
- Linear classifier can be non-linear!
- Techniques to learn the parameters

# Summary

- Linear classifiers:
  - Perceptron
  - Naive Bayes
  - (Multi-nomial) logistic regression
- Linear classifier can be non-linear!
- Techniques to learn the parameters
- Trade-offs when making decisions

# Outline


1. What is classification?
2. Decision theory
3. Generative classifiers
4. Discriminative classifiers
5. Summary
6. Exercises

# Exercise 1

Can you propose a Naive Bayes classifier with continuous features? Derive the maximum likelihood estimates of the parameters.

# Exercise 2

Derive the update equations of a generative classifier with discrete binary features.

# Exercise 3

What is the form of the decision boundary for a binary classifier with Gaussian features with different covariance matrices?

# Exercise 3$^\star$

What are the expressions of the precision and the recall in the multi-class case?

# References


J. H. Albert, and S. Chib (1993): *Bayesian Analysis of Binary and Polychotomous Response Data*. Journal of the American Statistical Association 88 (422): 669?679.

C. Bishop: *Pattern Recognition and Machine Learning*. Springer, 2006.

J. Davis and M. Goadrich: *The Relationship Between Precision-Recall and ROC Curves*. ICML 2006.

Y. Ng and M. Jordan: *On Discriminative vs. Geneartive classifiers: A comparison of Logistic Regression and Naive Bayes*. NIPS 2001.

M. Seeger: *Pattern Classification and Machine Learning*. Lecture notes EPFL, 2012.