Lecture 2

Linear Discriminants

M. Poel
G. Englebienne

University of Twente

Introduction
- Linear Discriminants

Linear Perceptron
- Motivation
- Training

Fisher's Linear Discriminant
- Optimising class separation

Probabilistic Models
- Generative Probabilistic Models
- Discriminative Probabilistic Models

Basis Functions

Introduction
- Linear Discriminants

Linear Perceptron
- Motivation
- Training

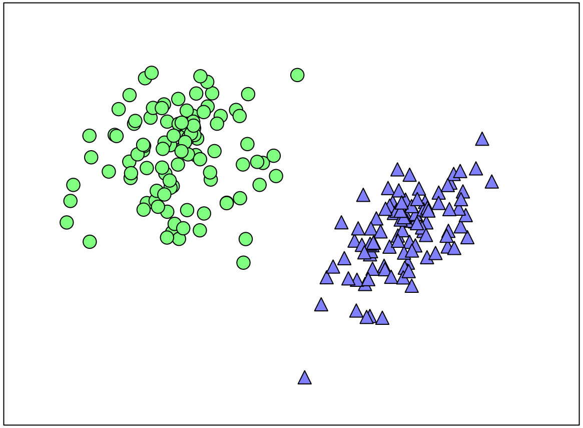
Fisher's Linear Discriminant
- Optimising class separation

Probabilistic Models
- Generative Probabilistic Models
- Discriminative Probabilistic Models

Basis Functions

Linear Discriminants

Example: Two Class problem



Linear Discriminants

Two-class classification problem:

- Obtain the class labels as a function of the features

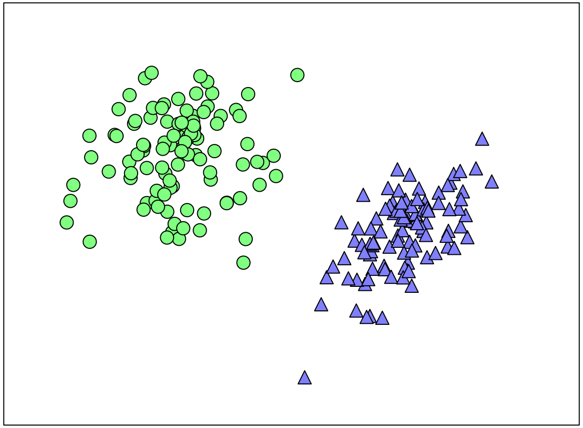
$$f(\mathbf{x}) = \begin{cases} \mathcal{C}_0 & \text{if } y(\mathbf{x}) > 0 \\ \mathcal{C}_1 & \text{if } y(\mathbf{x}) < 0 \end{cases}$$

- Discriminant: $y(\mathbf{x}) = 0$
- Linear classification: $y(\mathbf{x}) = 0$ is a linear function of $\mathbf{x}$

$$\begin{aligned} y(\mathbf{x}) &= w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \\ &= w_0 + \mathbf{w}^\top \mathbf{x} \end{aligned}$$

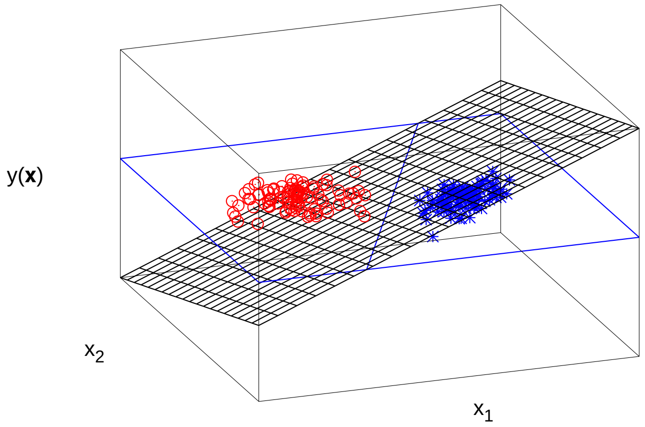
Linear Discriminants

Example



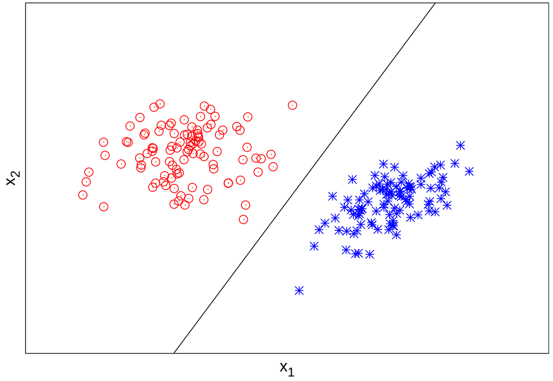
Linear Discriminants

Example



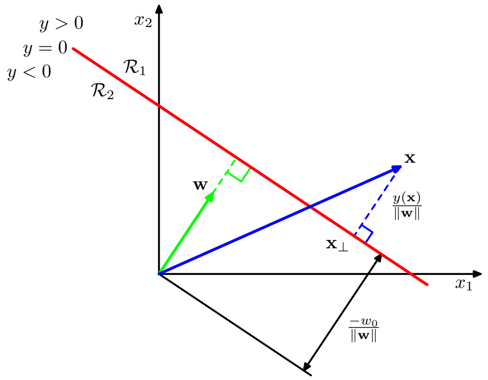
Linear Discriminants

Example



Representation of the linear discriminant

- $y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0$



- Slope is determined by **w**, offset from origin by w_0

Representation

- For convenience
$$\tilde{\mathbf{w}}^\top = (w_0, w_1, \cdots w_D)$$
$$\tilde{\mathbf{x}}^\top = (1, x_1, \cdots, x_D)$$
so that $y(\mathbf{x}) = \tilde{\mathbf{w}}^\top \tilde{\mathbf{x}}$
- Hyperplane equation becomes $\tilde{\mathbf{w}}^\top \tilde{\mathbf{x}} = 0$
- Generalisation to k classes:
  - Combine multiple 2-class classifiers
  - Beware of ambiguous regions

Question

How do we determine $\mathbf{w}$?

Representation

- For convenience
$$\tilde{\mathbf{w}}^\top = (w_0, w_1, \cdots w_D)$$
$$\tilde{\mathbf{x}}^\top = (1, x_1, \cdots, x_D)$$
so that $y(\mathbf{x}) = \tilde{\mathbf{w}}^\top \tilde{\mathbf{x}}$
- Hyperplane equation becomes $\tilde{\mathbf{w}}^\top \tilde{\mathbf{x}} = 0$
- Generalisation to k classes:
    - Combine multiple 2-class classifiers
    - Beware of ambiguous regions

Question

How do we determine $\mathbf{w}$?

Learning the optimal discriminant

- We have a classification function $y(\mathbf{x}; \mathbf{w})$
  - Modify $\mathbf{x}$: classify a new datapoint
  - Modify $\mathbf{w}$: change the discriminant
- Define a measure of the "goodness" of the classifier
  - Evaluate the classifier
  - Optimise the classifier

So what's a good classifier?

- Minimise misclassified elements in training set
- Maximise separation of the most ambiguous elements
- Maximise data probability
- ...

Learning the optimal discriminant

- We have a classification function $y(\mathbf{x}; \mathbf{w})$
  - Modify $\mathbf{x}$: classify a new datapoint
  - Modify $\mathbf{w}$: change the discriminant
- Define a measure of the "goodness" of the classifier
  - Evaluate the classifier
  - Optimise the classifier

So what's a good classifier?

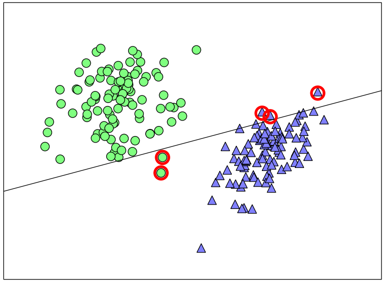
- Minimise misclassified elements in training set
- Maximise separation of the most ambiguous elements
- Maximise data probability
- ...

Learning the optimal discriminant

- We have a classification function $y(\mathbf{x}; \mathbf{w})$
  - Modify $\mathbf{x}$: classify a new datapoint
  - Modify $\mathbf{w}$: change the discriminant
- Define a measure of the "goodness" of the classifier
  - Evaluate the classifier
  - Optimise the classifier

So what's a good classifier?

- Minimise misclassified elements in training set
- Maximise separation of the most ambiguous elements
- Maximise data probability
- ...

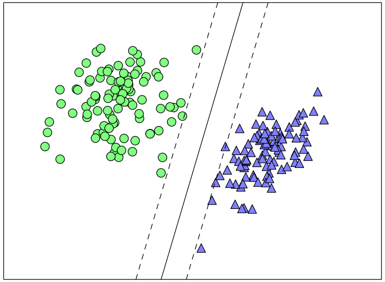


Learning the optimal discriminant

- We have a classification function $y(\mathbf{x}; \mathbf{w})$
  - Modify $\mathbf{x}$: classify a new datapoint
  - Modify $\mathbf{w}$: change the discriminant
- Define a measure of the "goodness" of the classifier
  - Evaluate the classifier
  - Optimise the classifier

So what's a good classifier?

- Minimise misclassified elements in training set
- Maximise separation of the most ambiguous elements
- Maximise data probability
- ...

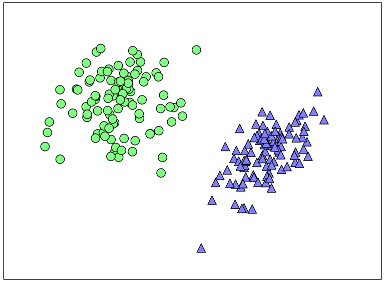


Learning the optimal discriminant

- We have a classification function $y(\mathbf{x}; \mathbf{w})$
  - Modify $\mathbf{x}$: classify a new datapoint
  - Modify $\mathbf{w}$: change the discriminant
- Define a measure of the "goodness" of the classifier
  - Evaluate the classifier
  - Optimise the classifier

So what's a good classifier?

- Minimise misclassified elements in training set
- Maximise separation of the most ambiguous elements
- Maximise data probability
- ...



Introduction
- Linear Discriminants

Linear Perceptron
- Motivation
- Training

Fisher's Linear Discriminant
- Optimising class separation

Probabilistic Models
- Generative Probabilistic Models
- Discriminative Probabilistic Models

Basis Functions

Linear Perceptron

- Inspired by the brain
- Model of neurons:
  - Binary signals (transmit / don't transmit)
  - Multiple inputs, one output
  - Send a signal to the output if inputs are sufficiently activated
- Activation:

$$f(\mathbf{x};\mathbf{w}) = h(\mathbf{w}^\top \mathbf{x}), \text{ where } h(a) = \begin{cases} +1 & \text{if } a > 0 \\ -1 & \text{otherwise} \end{cases}$$

  is a (non-linear) step function.
- Training data set $\{\mathbf{x}_n, t_n\}$ uses the target coding scheme:

$$t_n = \begin{cases} +1 & \text{if } \mathbf{x}_n \text{ belongs to } \mathcal{C}_1 \\ -1 & \text{otherwise} \end{cases}$$

Training the Linear Perceptron

- Classification is correct iff:
  - $t_n > 0$ and $\mathbf{w}^\top \mathbf{x} > 0$ (because then $f(\mathbf{w}^\top \mathbf{x}) > 0$), or
  - $t_n < 0$ and $\mathbf{w}^\top \mathbf{x} < 0$

  Therefore, classification is correct if

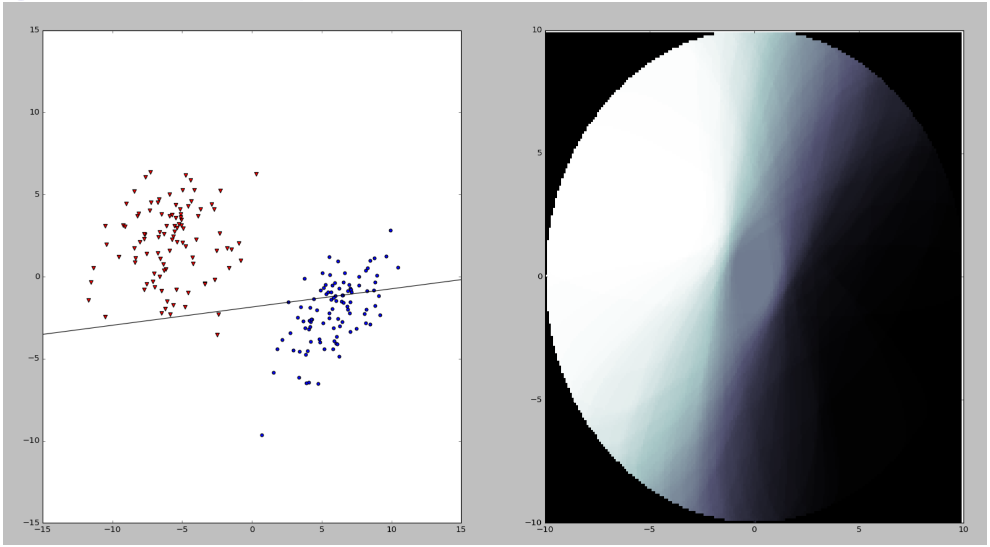
$$\mathbf{w}^\top \mathbf{x}_n t_n > 0$$

- We want to minimise the misclassification rate

$$E(\mathbf{w}) = -\sum_{n \in \mathcal{M}} \mathbf{w}^\top \mathbf{x}_n t_n$$

where $\mathcal{M} = \{\mathbf{x}_i | \mathbf{w}^\top \mathbf{x}_i t_i \leqslant 0\}$ is the set of misclassified samples.

Training the Linear Perceptron



Perceptron Training Algorithm

- Update $\mathbf{w}$ by stochastic gradient descent:

$$\begin{aligned}\mathbf{w}^{(i)} &= \mathbf{w}^{(i-1)} - \eta \nabla E(\mathbf{w}^{(i-1)}) \\ &= \mathbf{w}^{(i-1)} + \eta \mathbf{x}_n t_n\end{aligned}$$

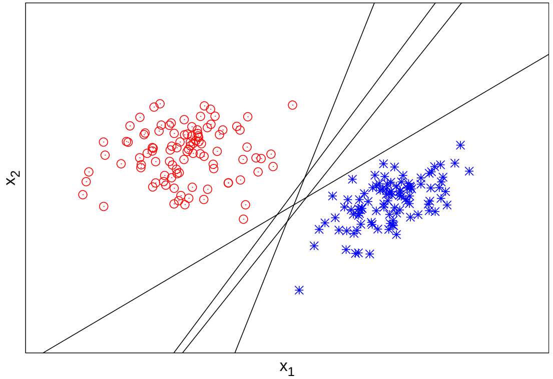
- Training algorithm: cycle through the training patterns and if misclassified, update $\mathbf{w}$
- This was revolutionary, because it seemed plausible that neurons could really work like this
- Perceptron Convergence Theorem: If the training set is linearly separable, the algorithm is guaranteed to find a solution in a finite number of steps.

Problems with perceptrons

- Training is slow, and does not generalise easily to more than 2 classes
- As long as the algorithm hasn't converged, there is no way to know whether the problem is not linearly separable, or just slow to converge.
- Because the set of misclassified training examples changes at each weight update, the algorithm is not guaranteed to reduce the overall error at each step. If the data is not linearly separable, no particularly good solution may be found.
- In general, many solutions exist and the final solution depends on the initial conditions, not on some optimality criterion.

Problems with perceptrons

Example



Introduction

Linear Discriminants

Linear Perceptron

Motivation

Training

Fisher's Linear Discriminant

Optimising class separation

Probabilistic Models

Generative Probabilistic Models

Discriminative Probabilistic Models

Basis Functions

Fisher's Linear Discriminant

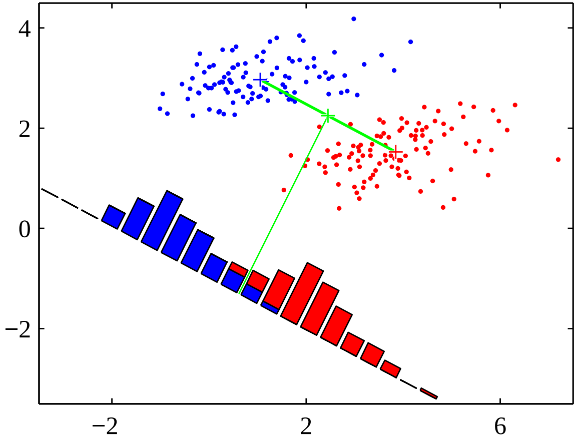
- Linear classification can be seen as dimensionality reduction:

$$y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$$

  projects the D-dimensional vector $\mathbf{x}$ to 1 dimension
- This results in loss of information; classes which are easily separable in D dimensions may strongly overlap in 1 dimension.
- Determine $\mathbf{w}$ so as to obtain a projection that maximises the class separation.

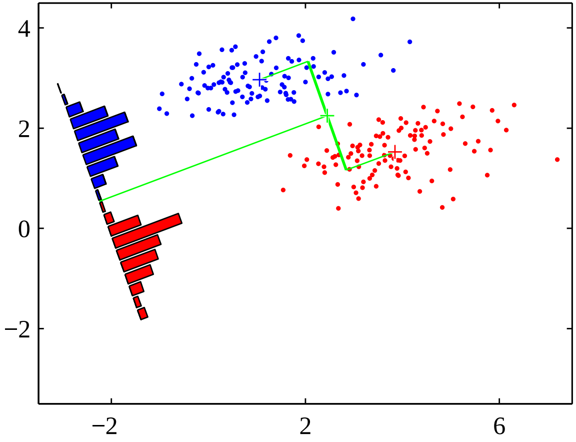
Measuring class separation

- Maximise distance between the projected class means
- Minimise variance within the projected classes



Measuring class separation

- Maximise distance between the projected class means
- Minimise variance within the projected classes



Optimising class separation

- The class means are given by:

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n$$

- Projected on $\mathbf{w}$, this gives: $m_k = \mathbf{w}^\top \mathbf{m}_k$, so (for a 2-class problem) we want to maximise $(m_2 - m_1)^2$
- Simultaneously we want to minimise the variance within the projected classes:

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (\mathbf{w}^\top \mathbf{x} - m_k)^2$$

- Fisher's discriminant combines these as follows:

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

Introduction
    Linear Discriminants

Linear Perceptron
    Motivation
    Training

Fisher's Linear Discriminant
    Optimising class separation

Probabilistic Models
    Generative Probabilistic Models
    Discriminative Probabilistic Models

Basis Functions

Probabilistic Models

- It is often useful to know more than the predicted class:
  - Confidence in the prediction
  - Capacity to meaningfully combine prediction and other information
- We are interested in the probability of the class given a datapoint, $p(\mathcal{C}|\mathbf{x})\ldots$
- But we can get other information:
  - How likely is the datapoint (do we have a good model for that datapoint?)
  - We can deal with missing data/dimensions

Generative Probabilistic Models

- Generative Probabilistic Models learn the distribution of the data, $p(\mathbf{x}|\mathcal{C})$
- This can be used for classification, using Bayes' rule:

$$p(\mathcal{C}_i|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_i)p(\mathcal{C}_i)}{\sum_k p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}$$

- If we learn the correct distributions, this is optimal: as the number of data points $\rightarrow \infty$, any other classifier will perform at most as well
- For certain distributions, the generative probabilistic model results in linear classification

Generative Probabilistic Models

- Generative Probabilistic Models learn the distribution of the data, $p(\mathbf{x}|\mathcal{C})$
- This can be used for classification, using Bayes' rule:

$$p(\mathcal{C}_i|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_i)p(\mathcal{C}_i)}{\sum_k p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}$$

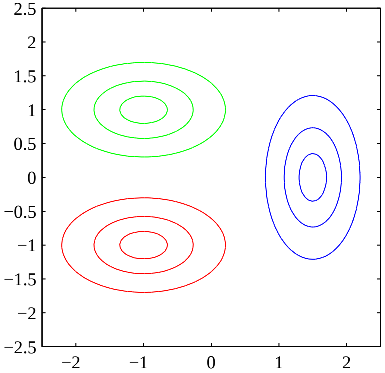
- If we learn the correct distributions, this is optimal: as the number of data points $\rightarrow \infty$, any other classifier will perform at most as well
- For certain distributions, the generative probabilistic model results in linear classification

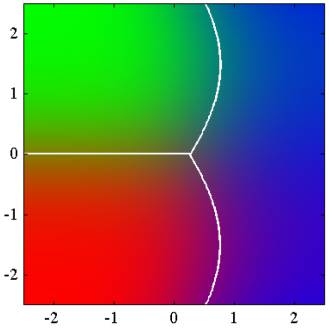
Bayes' rule: some terminology

$$\underbrace{p(\mathcal{C}_i|\mathbf{x})}_{\text{posterior}} = \underbrace{\frac{\overbrace{p(\mathbf{x}|\mathcal{C}_i)}^{\text{likelihood}}\ \overbrace{p(\mathcal{C}_i)}^{\text{prior}}}{\sum_k p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}}_{\text{evidence}}$$

Linear Discriminants with Normal densities

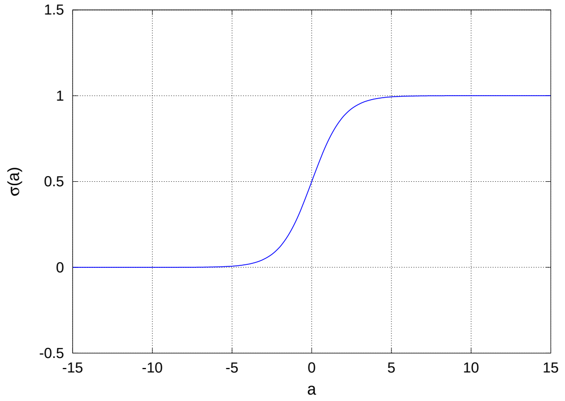
Normal distributions with $\mathbf{\Sigma}_k = \mathbf{\Sigma}_l$





Discriminative Probabilistic Models

The posterior distribution $p(\mathcal{C}_k|\mathbf{x})$ for a two-class problem can be written as a sigmoid $\sigma(a) = \frac{1}{1+e^{-a}}$:



Discriminative Probabilistic Models

The posterior distribution $p(\mathcal{C}_k|\mathbf{x})$ for a two-class problem can be written as a sigmoid $\sigma(a) = \frac{1}{1+e^{-a}}$:

$$\begin{aligned} p(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \frac{1}{1 + \frac{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}} \\ &= \sigma(a) \text{ where } a = -\ln \frac{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)} \end{aligned}$$

For k classes, the posterior is the softmax function

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

Generative vs. Discriminative

Generative models learn the distribution of the data:

- You can sample from them and generate new data
- Use Bayes' rule to obtain posterior probabilities
- Classification is optimal *if the true distributions are known*

Discriminative models directly optimise $p(\mathcal{C}_k|\mathbf{x}) = \sigma(a)$

- In the case of Normal distributions $p(\mathbf{x}|\mathcal{C}_i)$ with shared covariances, a is a linear function of $\mathbf{x}$
- However this can be relaxed: a is a linear function of $\mathbf{x}$ for many distributions of $\mathbf{x}$
- Directly optimising $p(\mathcal{C}_k|\mathbf{x})$ typically requires fewer parameters to be adapted

Logistic Regression

If we write the posterior probability as:

$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x})$$
$$p(\mathcal{C}_2|\mathbf{x}) = 1 - p(\mathcal{C}_1|\mathbf{x})$$

How do we obtain **w**?

- Maximum Likelihood — Maximise $p(\mathcal{C}_i|\mathbf{x})$ for all $\mathcal{C}_i$
- No closed-form solution: iterative algorithm
  - Gradient descent (useful fact: $\frac{d}{da}\sigma(a) = \sigma(a)(1 - \sigma(a))$)
  - Newton-Raphson method: Iterative Reweighted Least Squares
- We will have a closer look at logistic regression in the lab session.

Introduction
Linear Discriminants

Linear Perceptron
Motivation
Training

Fisher's Linear Discriminant
Optimising class separation

Probabilistic Models
Generative Probabilistic Models
Discriminative Probabilistic Models

Basis Functions

Non-linear Basis Functions

Linear models are by definition restricted: it is easy to find datasets that are not linearly separable.
However Linear classifiers have quite appealing characteristics:

- Easy to train and use
- Few adjustable parameters $\rightarrow$ less prone to overfitting

We can keep the advantages of the classifiers and extend them to non-linear problems by transforming the features:

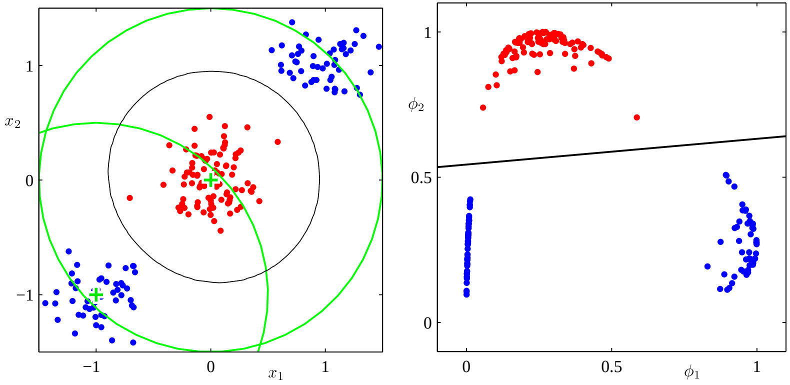
$$\mathbf{x} \rightarrow \phi(\mathbf{x}) \tag{1}$$

$$\mathbf{w}^\top \mathbf{x} \rightarrow \mathbf{w}^\top \phi(\mathbf{x}) \tag{2}$$

Using non-linear basis functions (and sometimes projecting the data into a higher number of dimensions) we can make complex data linearly separable.

Non-Linear Basis Functions

Transforming features x_1 and x_2 using "Gaussian" basis functions: $\phi_1 = f(\mathbf{x} - \boldsymbol{\mu}_1)$, where $\boldsymbol{\mu}_1 = (-1, -1)^\top$ and $\phi_2 = f(\mathbf{x} - \boldsymbol{\mu}_2)$, where $\boldsymbol{\mu}_2 = (0, 0)^\top$



Wrap-up

Today, we have seen:

▶ Linear discriminants	(Bishop, p. 181–182)
▶ Perceptron	(Bishop, p. 192–196)
▶ Fisher's linear discriminant	(Bishop, p. 186–189)
▶ Probabilistic motivation	(Bishop, p. 196–201)
▶ Discriminative linear models	(Bishop, p. 205–206)