Lecture 3. Linear Regression

COMP90051 Statistical Machine Learning

Lecturer: Jean Honorio

THE UNIVERSITY OF
MELBOURNE

POSTERA CRESCAM LAUDE

This lecture

- Linear regression

  * Simple model (convenient maths at expense of flexibility)

  * Often needs less data, "interpretable", lifts to non-linear

  * Derivable under all Statistical Schools: Lect 2 case study

    - This week: Frequentist + Decision theory derivations

    Later in semester: Bayesian approach

  * Convenient optimisation: Training by "analytic" (exact) solution

- Basis expansion: Data transform for more expressive models

Linear Regression via Decision Theory

A warm-up example

Example: Predict humidity from temperature

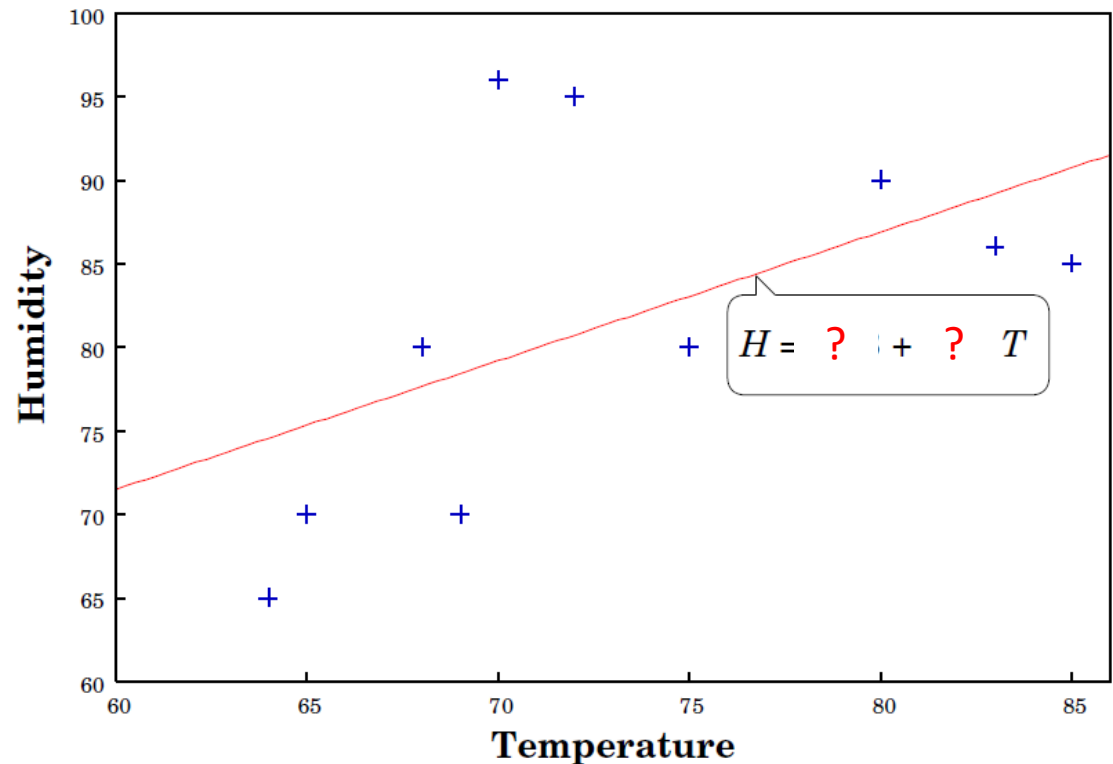| Temperature	Humidity
TRAINING DATA	
85	85
80	90
83	86
70	96
68	80
65	70
64	65
72	95
69	70
75	80
TEST DATA	
75	70

$H =$? + ? T
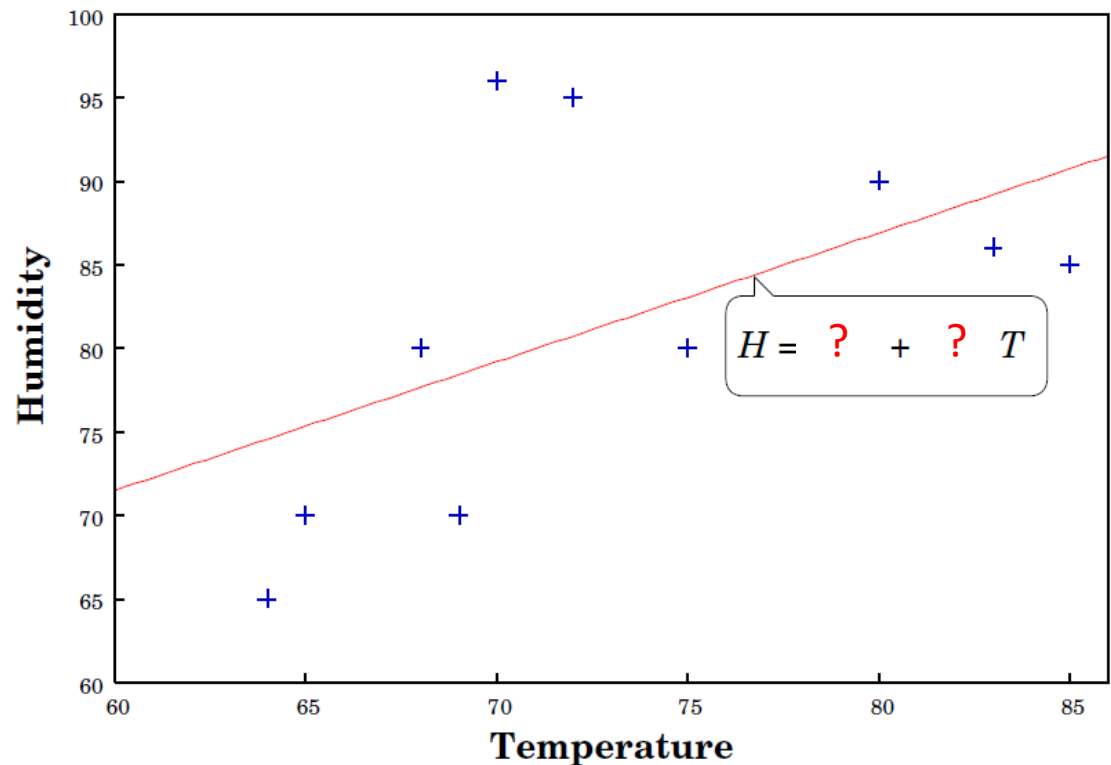
In regression, the task is to predict numeric response (*aka* dependent variable) from features (*aka* predictors or independent variables)

Assume a linear relation: $H = a + bT$

(H – humidity; T – temperature; a, b – parameters)

4

Example: Problem statement

- The model is
  $$H = a + bT$$

- Fitting the model $=$ finding "best" a, b values for data at hand

- Important criterion: minimise the sum of squared errors (*aka* residual sum of squares)

The scatter plot shows Humidity versus Temperature with a red fitted line. A callout reads: $H = \ ? \ + \ ? \ T$

Example: Minimise Sum Squared Errors

To find a, b that minimise $L = \sum_{i=1}^{10}\left(H_i - (a + b\,T_i)\right)^2$

set derivatives to zero:

$$\frac{\partial L}{\partial a} = -2\sum_{i=1}^{10}(H_i - a - b\,T_i) = 0$$

if we know b, then $\hat{a} = \frac{1}{10}\sum_{i=1}^{10}(H_i - b\,T_i)$

$$\frac{\partial L}{\partial b} = -2\sum_{i=1}^{10}(H_i - a - b\,T_i)T_i = 0$$

if we know a, then $\hat{b} = \frac{1}{\sum_{i=1}^{10} T_i^2}\sum_{i=1}^{10}(H_i - a)T_i$

High-school optimisation:
- Write derivative
- Set to zero
- Solve for model
- (Check 2nd derivatives)

6

Example: Analytic solution

- We have two equations and two unknowns a, b

- Rewrite as a system of linear equations

$$\begin{pmatrix} 10 & \sum_{i=1}^{10} T_i \\ \sum_{i=1}^{10} T_i & \sum_{i=1}^{10} T_i^2 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{10} H_i \\ \sum_{i=1}^{10} T_i H_i \end{pmatrix}$$

- Analytic solution: $a = 25.3, b = 0.77$

- (Solve using `numpy.linalg.solve`)

More general decision rule

- Adopt a linear relationship between response $y \in \mathbb{R}$ and an instance with features $x_1, \ldots, x_m \in \mathbb{R}$

$$\hat{y} = w_0 + \sum_{i=1}^{m} x_i w_i$$

Here $w_0, \ldots, w_m \in \mathbb{R}$ denote weights (model parameters)

- Trick: add a dummy feature $x_0 = 1$ and use vector notation

$$\hat{y} = \sum_{i=0}^{m} x_i w_i = \boldsymbol{x}' \boldsymbol{w}$$

A lowercase symbol in **bold face** indicates a vector; $\boldsymbol{x}'$ denotes transpose

Mini Summary

- Linear regression
  - Simple, effective, "interpretable", basis for many approaches
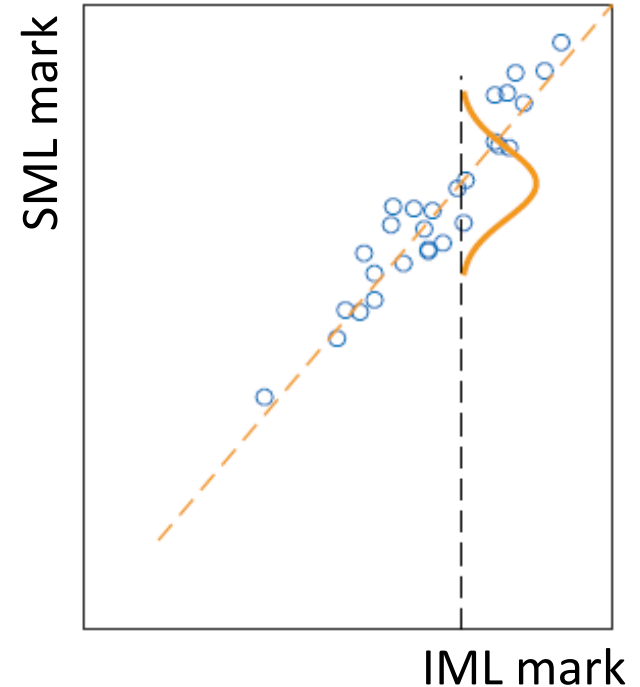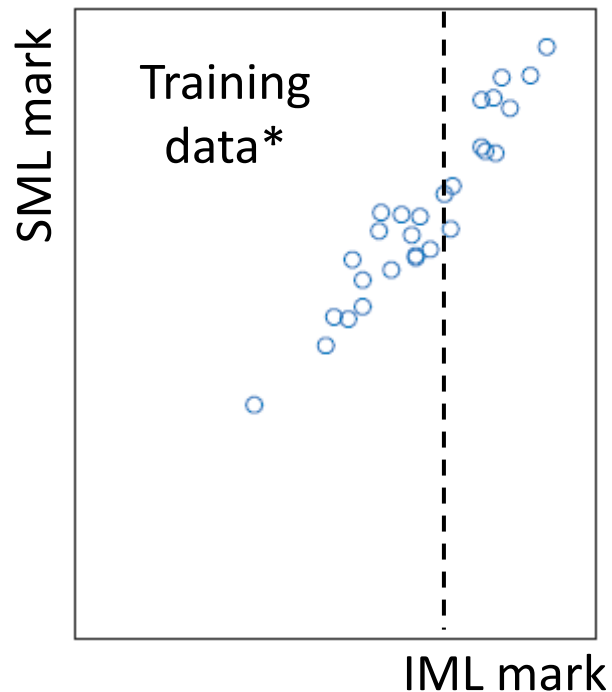 - Decision-theoretic frequentist derivation

Next:

Frequentist derivation; Solution/training approach

Linear Regression
via Frequentist Probabilistic Model
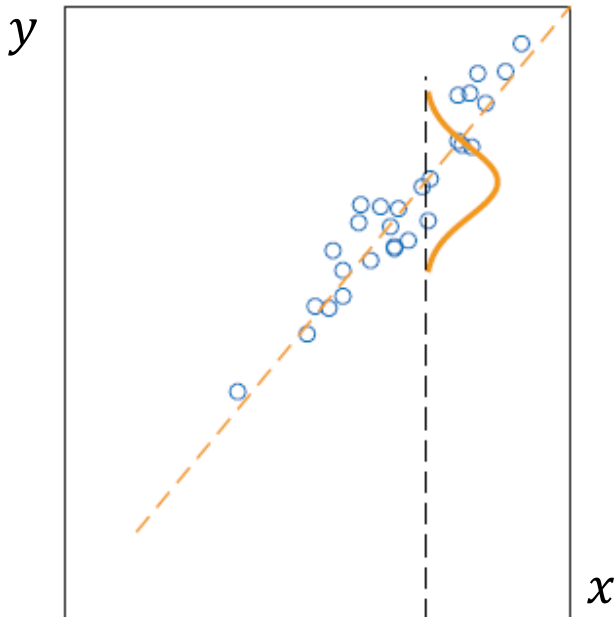
Max-Likelihood Estimation

Data is noisy!

Example: predict mark for Statistical Machine Learning (SML)
from mark for Intro ML (IML aka KT)

Regression as a probabilistic model

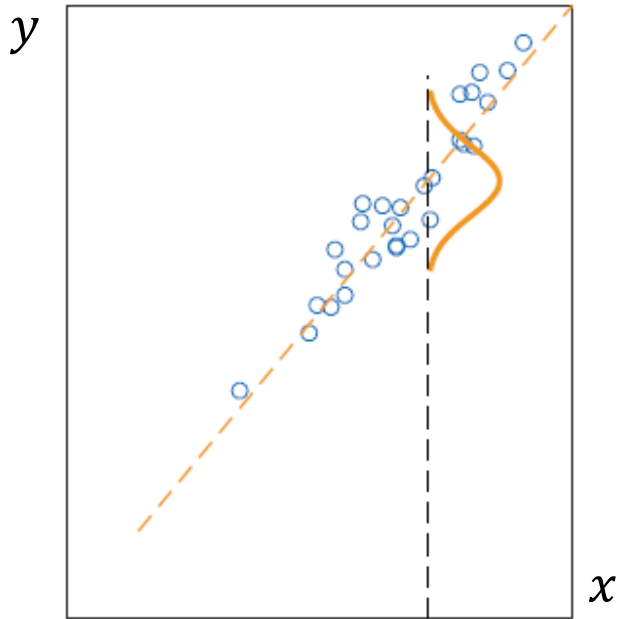- Assume a probabilistic model:
  $$y = \boldsymbol{x}'\boldsymbol{w} + \varepsilon$$
  * Here $\boldsymbol{x}, y$ and ε are r.v.'s
  * Variable ε encodes noise

- Next, assume Gaussian noise (indep. of $\boldsymbol{x}$): $\varepsilon \sim \mathcal{N}(0, \sigma^2)$
  thus: $y \sim \mathcal{N}(\boldsymbol{x}'\boldsymbol{w}, \sigma^2)$

- Recall that $\mathcal{N}(z; \mu, \sigma^2) \equiv \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right)$

- Therefore

$$p_{\boldsymbol{w},\sigma^2}(y|\boldsymbol{x}) = \mathcal{N}(y; \boldsymbol{x}'\boldsymbol{w}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - \boldsymbol{x}'\boldsymbol{w})^2}{2\sigma^2}\right)$$

this is a squared error!

Parametric probabilistic model

- Using simplified notation, discriminative model is:

$$p(y|\boldsymbol{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - \boldsymbol{x}'\boldsymbol{w})^2}{2\sigma^2}\right)$$

- Unknown parameters: $\boldsymbol{w}, \sigma^2$

- Given observed data $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)\}$, we want to find parameter values that "best" explain the data

- Maximum-likelihood estimation: choose parameter values that maximise the probability of observed data

13

Maximum likelihood estimation

- Assuming independence of data points, the probability of data is

$$p(y_1, \ldots, y_n | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \prod_{i=1}^{n} p(y_i | \boldsymbol{x}_i)$$

- For $p(y_i | \boldsymbol{x}_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \boldsymbol{x}_i'\boldsymbol{w})^2}{2\sigma^2}\right)$

- "Log trick": Instead of maximising this quantity, we can maximise its logarithm

$$\sum_{i=1}^{n} \log p(y_i | \boldsymbol{x}_i) = -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i'\boldsymbol{w})^2 + C$$

the sum of squared errors!

here C doesn't depend on $\boldsymbol{w}$ (it's a constant)

- <u>Under this model</u>, maximising log-likelihood as a function of $\boldsymbol{w}$ is equivalent to minimising the sum of squared errors

Method of least squares

- Training data: $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$. Note bold face in $\boldsymbol{x}_i$

- For convenience, place instances in rows (so attributes go in columns), representing training data as an $n \times m$ matrix $\boldsymbol{X}$, and n vector $\boldsymbol{y}$

- Probabilistic model/decision rule assumes $\boldsymbol{y} \approx \boldsymbol{Xw}$

- To find $\boldsymbol{w}$, minimise the sum of squared errors

$$L = \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i' \boldsymbol{w})^2$$

$$= \|\boldsymbol{y} - \boldsymbol{Xw}\|^2$$

$$= (\boldsymbol{y} - \boldsymbol{Xw})'(\boldsymbol{y} - \boldsymbol{Xw}) \qquad \text{since } \|\boldsymbol{u}\|^2 = \boldsymbol{u}'\boldsymbol{u}$$

$$= \boldsymbol{y}'\boldsymbol{y} - 2\boldsymbol{y}'\boldsymbol{Xw} + \boldsymbol{w}'\boldsymbol{X}'\boldsymbol{Xw}$$

In this slide: UPPERCASE symbol in bold face means matrix; $\boldsymbol{X}'$ denotes transpose

Method of least squares

- To find $\boldsymbol{w}$, minimise the sum of squared errors
$$L = \boldsymbol{y}'\boldsymbol{y} - 2\boldsymbol{y}'\boldsymbol{X}\boldsymbol{w} + \boldsymbol{w}'\boldsymbol{X}'\boldsymbol{X}\boldsymbol{w}$$

- Setting gradient to zero
$$\nabla L = \left[\frac{\partial L}{\partial w_1}, \dots, \frac{\partial L}{\partial w_m}\right]' = -2\boldsymbol{X}'\boldsymbol{y} + 2\boldsymbol{X}'\boldsymbol{X}\boldsymbol{w} = \boldsymbol{0}$$

- Solving for $\boldsymbol{w}$ yields
$$\widehat{\boldsymbol{w}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}$$

Analytic solution:
- Write gradient
- Set to zero
- Solve for model

  * This system of equations called the normal equations
  * System is well defined only if the inverse exists

In this slide: UPPERCASE symbol in bold face means matrix; $\boldsymbol{X}'$ denotes transpose; $\boldsymbol{X}^{-1}$ denotes matrix inverse

Bayesian derivation?

- Later in the semester: return of linear regression

- Fully Bayesian, with a posterior:
  - Bayesian linear regression

- Bayesian (MAP) point estimate of weight vector:
  - Adds a penalty term to sum of squared losses
  - Equivalent to L_2 "regularisation" to be covered next week
  - Called: ridge regression

Mini Summary

- Linear regression
  - * Simple, effective, "interpretable", basis for many approaches
  - * Probabilistic frequentist derivation
  - * Solution by normal equations

 Later in semester: Bayesian approaches

Next: Basis expansion for non-linear regression

Basis Expansion

Extending the utility of models via
data transformation

Basis expansion for linear regression

- Real data is likely to be non-linear

- What if we still wanted to use a linear regression?

  * Simple, easy to understand, computationally efficient, etc.

- How to marry non-linear data to a linear method?
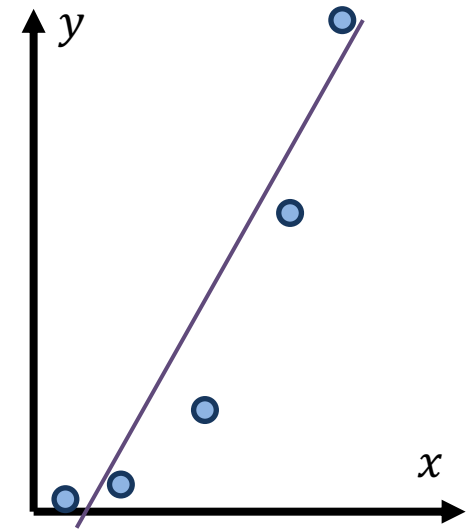
If you can't beat'em, join'em

Transform the data

- The trick is to transform the data: Map data into another features space, s.t. data is linear in that space

- Denote this transformation $\varphi: \mathbb{R}^m \to \mathbb{R}^k$. If $\boldsymbol{x}$ is the original set of features, $\varphi(\boldsymbol{x})$ denotes new feature set

- <u>Example</u>: suppose there is just one feature x, and the data is scattered around a parabola rather than a straight line
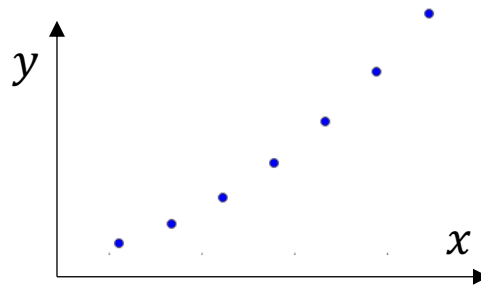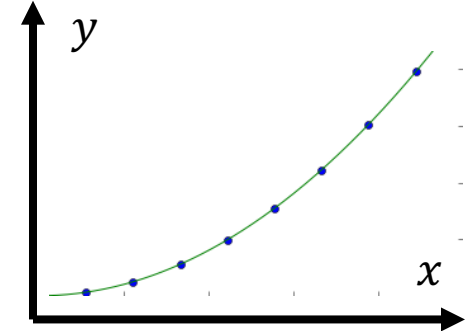
Example: Polynomial regression

- Define
  * $\varphi_1(x) = x$
  * $\varphi_2(x) = x^2$

- Next, apply linear regression to φ_1, φ_2

$$y = w_0 + w_1 \varphi_1(x) + w_2 \varphi_2(x) = w_0 + w_1 x + w_2 x^2$$

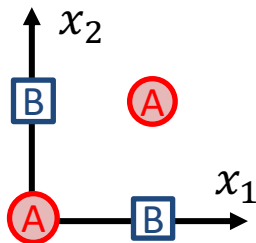 and here you have **quadratic regression**

- More generally, obtain **polynomial regression** if the new set of attributes are powers of x

Example: linear classification

- Example binary classification problem: Dataset not linearly separable

- Define transformation as

$$\varphi_i(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{z}_i\|, \text{ where } \boldsymbol{z}_i \text{ some pre-defined constants}$$

- Choose $\boldsymbol{z}_1 = [0,0]'$, $\boldsymbol{z}_2 = [0,1]'$, $\boldsymbol{z}_3 = [1,0]'$, $\boldsymbol{z}_4 = [1,1]'$

there exist weights that make
new data separable, e.g.:

The transformed data is linearly separable!

w_1	w_2	w_3	w_4
1	0	0	1

x_1	x_2	y
0	0	Class A
0	1	Class B
1	0	Class B
1	1	Class A

φ_1	φ_2	φ_3	φ_4
0	1	1	$\sqrt{2}$
1	0	$\sqrt{2}$	1
1	$\sqrt{2}$	0	1
$\sqrt{2}$	1	1	0

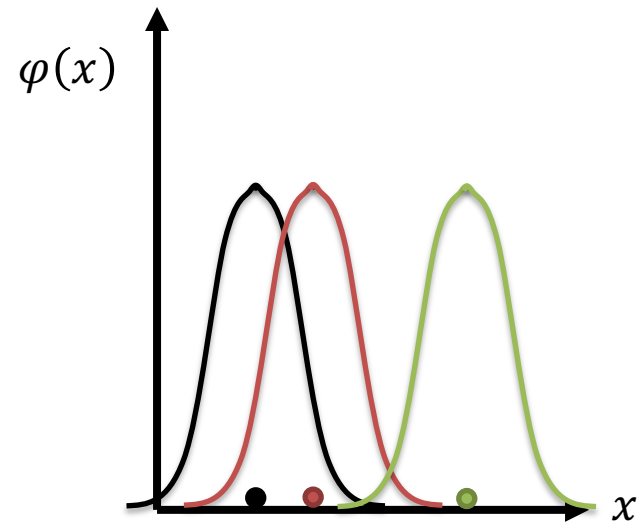| $\boldsymbol{\varphi}'\boldsymbol{w}$	y
$\sqrt{2}$	Class A
2	Class B
2	Class B
$\sqrt{2}$	Class A

Radial basis functions

- Previous example: motivated by approximation theory where sums of RBFs approx. functions

- A radial basis function is a function of the form $\varphi(\boldsymbol{x}) = \psi(\|\boldsymbol{x} - \boldsymbol{z}\|)$, where $\boldsymbol{z}$ is a constant

- Examples:

- $\varphi(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{z}\|$

- $\varphi(\boldsymbol{x}) = \exp\left(-\frac{1}{\sigma}\|\boldsymbol{x} - \boldsymbol{z}\|^2\right)$

Challenges of basis expansion

- Basis expansion can significantly increase the utility of methods, especially, linear methods

- In the above examples, one limitation is that the transformation needs to be defined beforehand
  * Need to choose the size of the new feature set
  * If using RBFs, need to choose z_i

- Regarding z_i, one can choose uniformly spaced points, or cluster training data and use cluster centroids

- Another popular idea is to use training data $z_i \equiv x_i$
  * E.g., $\varphi_i(x) = \psi(\|x - x_i\|)$
  * However, for large datasets, this results in a large number of features $\rightarrow$ computational hurdle

Further directions

- There are several avenues for taking the idea of basis expansion to the next level
  - ✳ Will be covered later in this subject

- One idea is to *learn* the transformation φ from data
  - ✳ E.g., Artificial Neural Networks

- Another powerful extension is the use of the kernel trick
  - ✳ "Kernelised" methods, e.g., kernelised perceptron

- Finally, in sparse kernel machines, training depends only on a few data points
  - ✳ E.g., SVM

Mini Summary

- Basis expansion

  * Extending model expressiveness via data transformation
  * Examples for linear and logistic regression

Next time:

First/second-order iteration optimisation;

Logistic regression - linear probabilistic model for classification.