Lecture 3. Linear Regression

COMP90051 Statistical Machine Learning

Lecturer: Jean Honorio

THE UNIVERSITY OF MELBOURNE

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

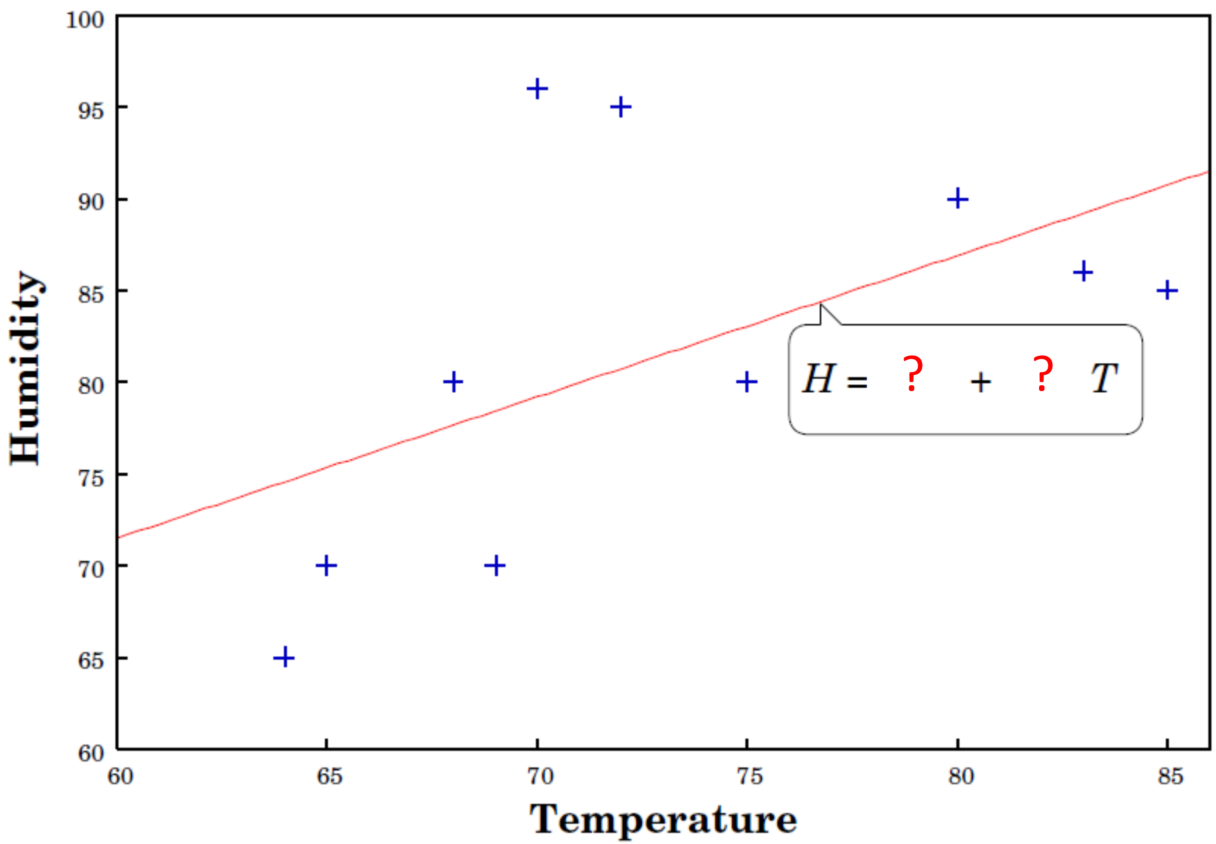
- Basis expansion: Data transform for more expressive models

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



In regression, the task is to predict numeric response (*aka* dependent variable) from features (*aka* predictors or independent variables)

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

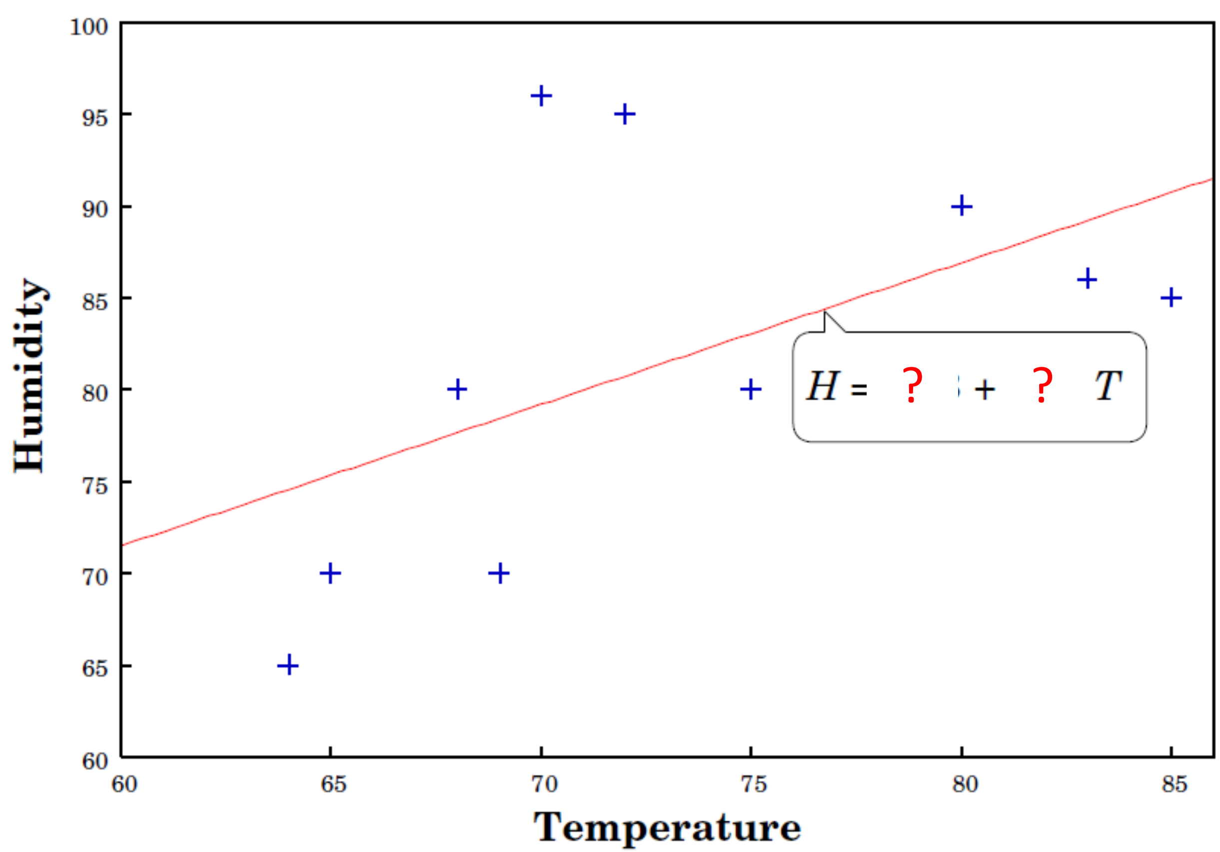
(H – humidity; T – temperature; a, b – parameters)

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)

RSS

absolute errors



Example: Minimise Sum Squared Errors

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

objective function

建立一个函数，不停地试探 a、b的值，找到一个最低点

set derivatives to zero:

partial

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

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

two linear equations

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

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`)

两未知数 两方程

More general decision rule

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

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

Here $w_0, \dots, w_m \in \mathbb{R}$ 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

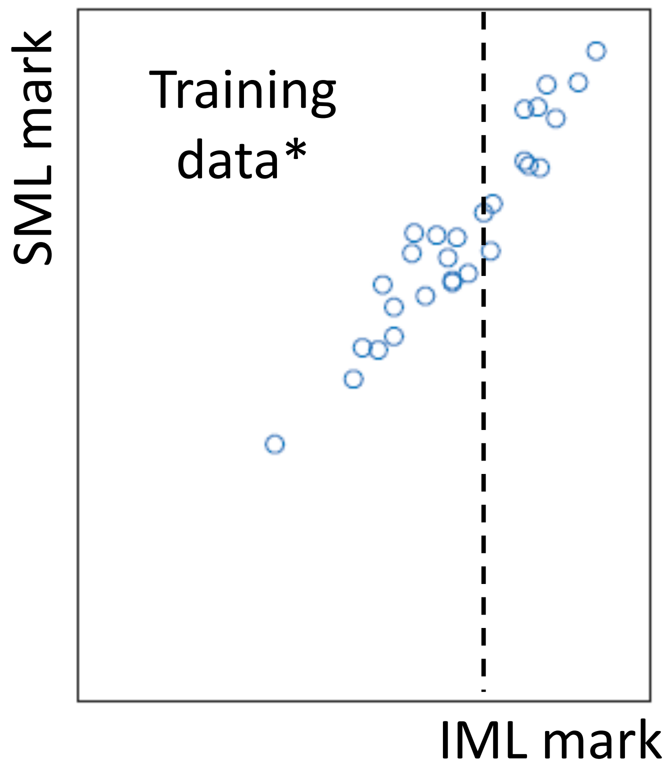
Linear Regression via Frequentist Probabilistic Model

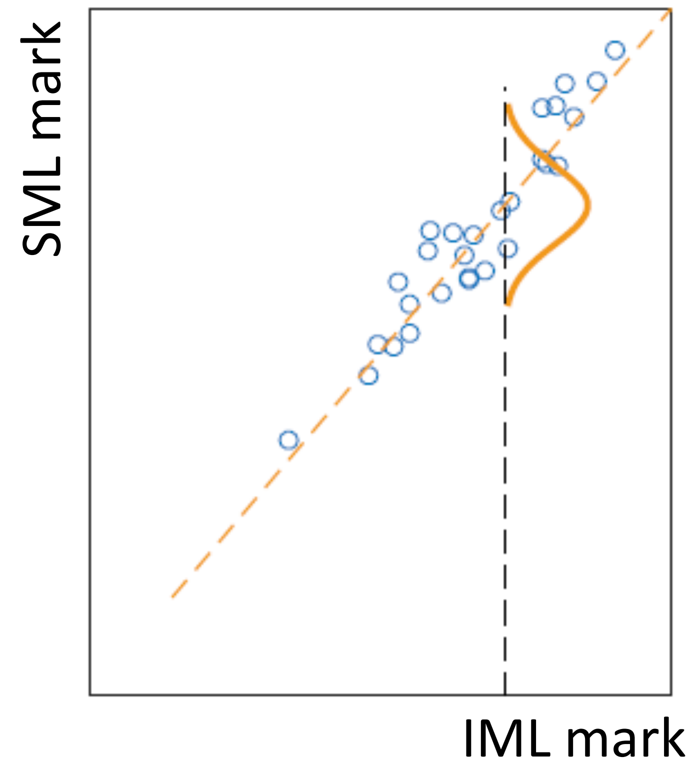
Max-Likelihood Estimation

Data is noisy!

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

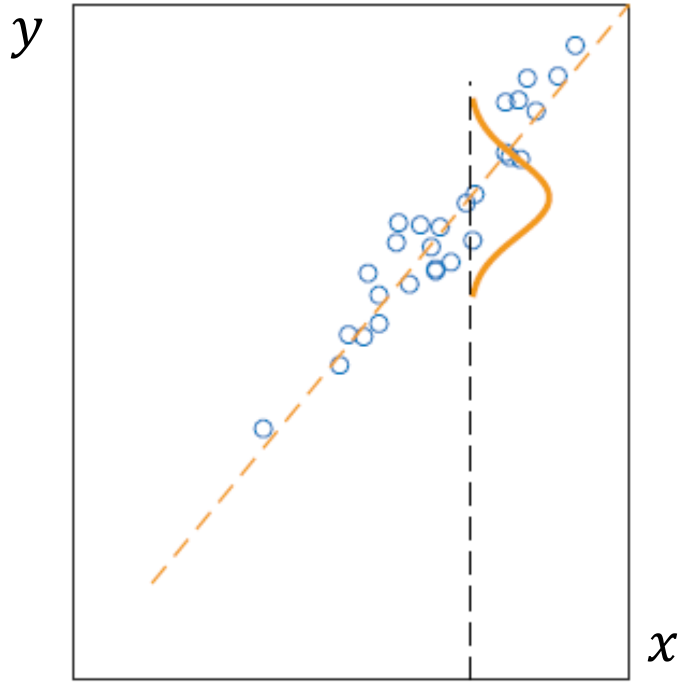
use input to predict output





* synthetic data :)

Regression as a probabilistic model



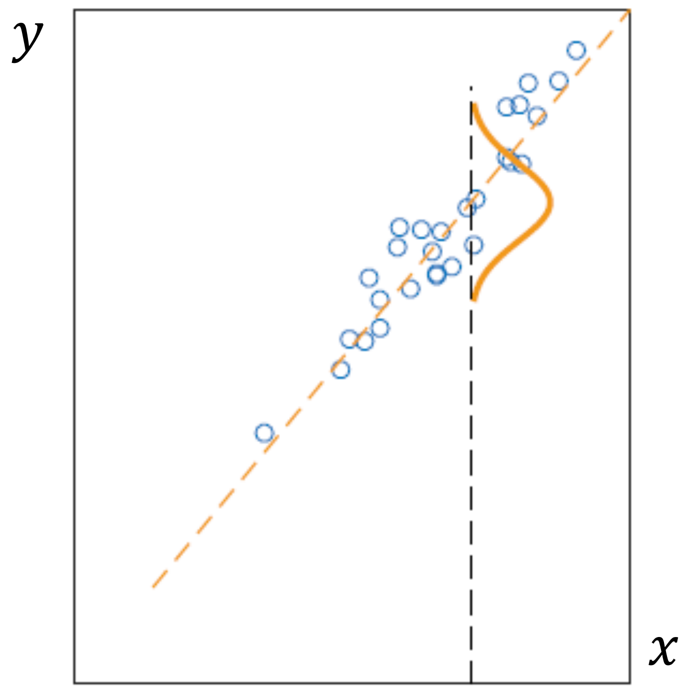
- Assume a probabilistic model: $y = \boldsymbol{x}'\boldsymbol{w} + \varepsilon$
  - Here $\boldsymbol{x}, y$ and ε are r.v.'s
  - Variable ε encodes noise
- Next, assume Gaussian noise (indep. of $\boldsymbol{x}$): $\varepsilon \sim \mathcal{N}(0, \sigma^2)$
  thus: $y \sim \mathcal{N}(\boldsymbol{x}'\boldsymbol{w}, \sigma^2)$

this is a squared error!

- 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

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

Parametric probabilistic model



- Using simplified notation, discriminative model is: 判别式模型

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

条件概率

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

- 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

Maximum likelihood estimation

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

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

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

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

连乘变加

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

the sum of squared errors!

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

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

Method of least squares

- Training data: $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)\}$. 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 $\boldsymbol{y} \approx \boldsymbol{X}\boldsymbol{w}$
- To find $\boldsymbol{w}$, minimise the sum of squared errors

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

$$= \|\underset{n \times n}{\boldsymbol{y}} - \underset{n \times m}{\boldsymbol{X}}\boldsymbol{w}\|^2$$

$$= (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})'(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) \quad \text{since } \|\boldsymbol{u}\|^2 = \boldsymbol{u}'\boldsymbol{u}$$

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

matrix

quadratic form

$A_{2\times3}$ $B_{2\times3}$ $3\times2\times2\times3 = 3\times3$

$A'B = B'A$ $B'A$ $3\times2\times2\times3 = 3\times3$

$w'X'y = y'Xw$

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

Method of least squares

二次型导数: $\frac{\partial(w'Aw)}{\partial w} = (A+A')w$ (A是矩阵, w是向量) $A=A'$

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

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

$\frac{\partial(a'Xw)}{\partial w} = X'a$ a 向量 X矩阵 w向量

$y'y$

- Setting gradient to zero

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

- Solving for $\boldsymbol{w}$ yields

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y} \quad = m\times 1$$

$m\times m \times m\times n \times n\times 1$

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

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

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

$L = w' x' X w$

$$\frac{\partial (w'Aw)}{\partial} = (A + A')w$$ 若A是对称的 $\Rightarrow A = A' \Rightarrow 2Aw$

$$\frac{\partial \; w'(x'x)w}{\partial w} = (x'x + x'x)w = 2x'Xw$$

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

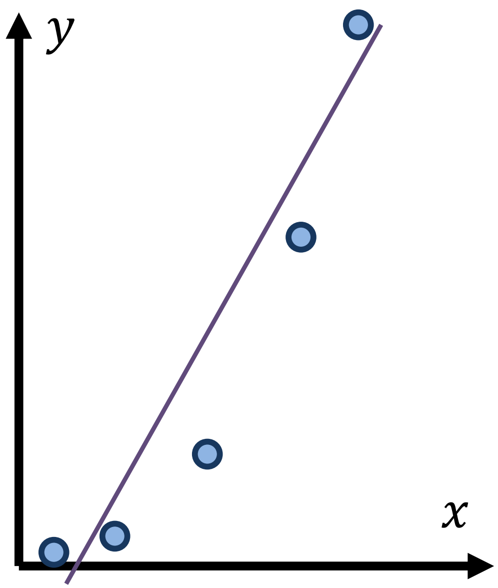
Next: Basis expansion for non-linear regression

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

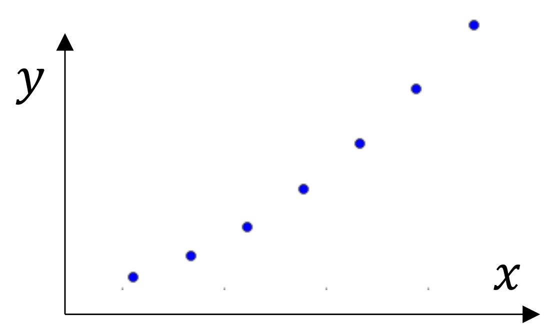
- The trick is to transform the data: Map data into another features space, s.t. data is linear in that space

非线性 $\xrightarrow{\text{转高维}}$ 线性   transform higher dimension → lower dimension or lower → higher

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

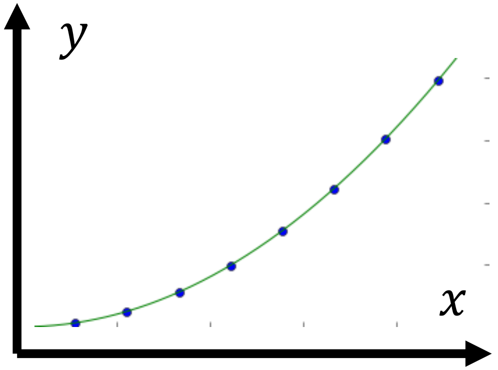
- Example: suppose there is just one feature x, and the data is scattered around a parabola rather than a straight line

$m=1: x \Rightarrow k=2,: x^2+x$



Example: Polynomial regression

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



- Next, apply linear regression to φ_1, φ_2

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

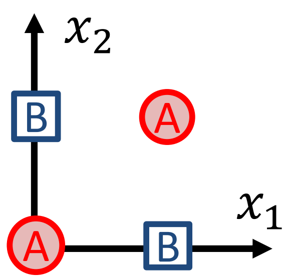
and here you have **quadratic regression**

- 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

  $\varphi_i(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{z}_i\|$, where $\boldsymbol{z}_i$ some pre-defined constants
- Choose $\boldsymbol{z}_1 = [0,0]'$, $\boldsymbol{z}_2 = [0,1]'$, $\boldsymbol{z}_3 = [1,0]'$, $\boldsymbol{z}_4 = [1,1]'$



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

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

w_1	w_2	w_3	w_4
1	0	0	1

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

The transformed data is linearly separable!

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

Radial basis functions

原始特征空间 (不可线性分) ⇒ 转换到 (线性可分) 高维空间

$(X^TX)^{-1}X^Ty$

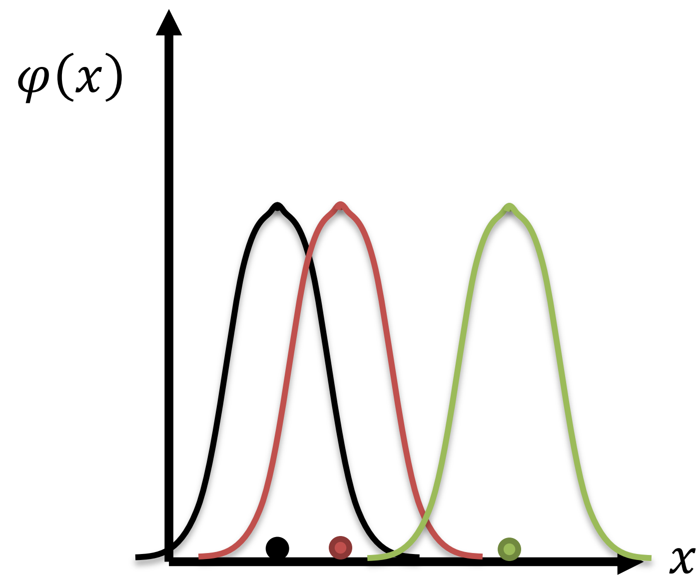
- Previous example: motivated by approximation theory where sums of RBFs approx. functions

- 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:

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

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



not linear separable ⇒ 转换到高维 可分 (separable)

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

  rule of transformation

  z_1, z_2, z_3

  什么样的方式是好的transformation
  - Need to choose the size of the new feature set
  - If using RBFs, need to choose $\boldsymbol{z}_i$
- Regarding $\boldsymbol{z}_i$, one can choose uniformly spaced points, or cluster training data and use cluster centroids
- Another popular idea is to use training data $\boldsymbol{z}_i \equiv \boldsymbol{x}_i$
  - E.g., $\varphi_i(\boldsymbol{x}) = \psi(\|\boldsymbol{x} - \boldsymbol{x}_i\|)$
  - However, for large datasets, this results in a large number of features → 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

project data onto higher infinite dimension separable without storing data into the computer.

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