Lecture 3. Linear Regression

COMP90051 Statistical Machine Learning

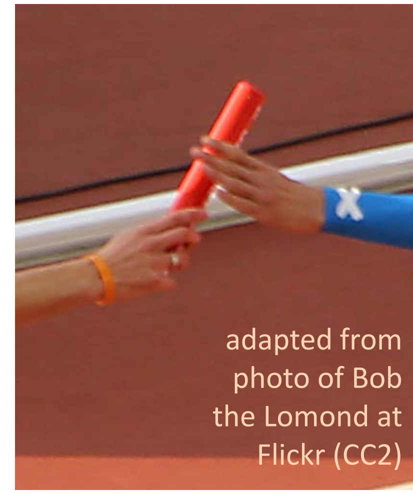
Semester 2, 2017
Lecturer: Andrey Kan

THE UNIVERSITY OF MELBOURNE

Copyright: University of Melbourne

Weeks 2 to 8 inclusive

- Lecturer: Andrey Kan
  - MS: Moscow, PhD: Melbourne
  - Email: andrey.kan@unimelb.edu.au
  - Office hour: after Thursday's lecture
- Past: software engineer
  - Optimising compilers
  - Multimedia framework
- Present: postdoc
  - Medical image analysis
  - Computational biology
- Future: applied scientist
  - Machine Learning
  - *Going to Seattle at the end of September!*


adapted from photo of Bob the Lomond at Flickr (CC2)

This lecture

- Linear regression
  - Worked example and the model
  - Regression as a probabilistic model
- Regularisation
  - Irrelevant features and an ill-posed problem
  - Regulariser as a prior

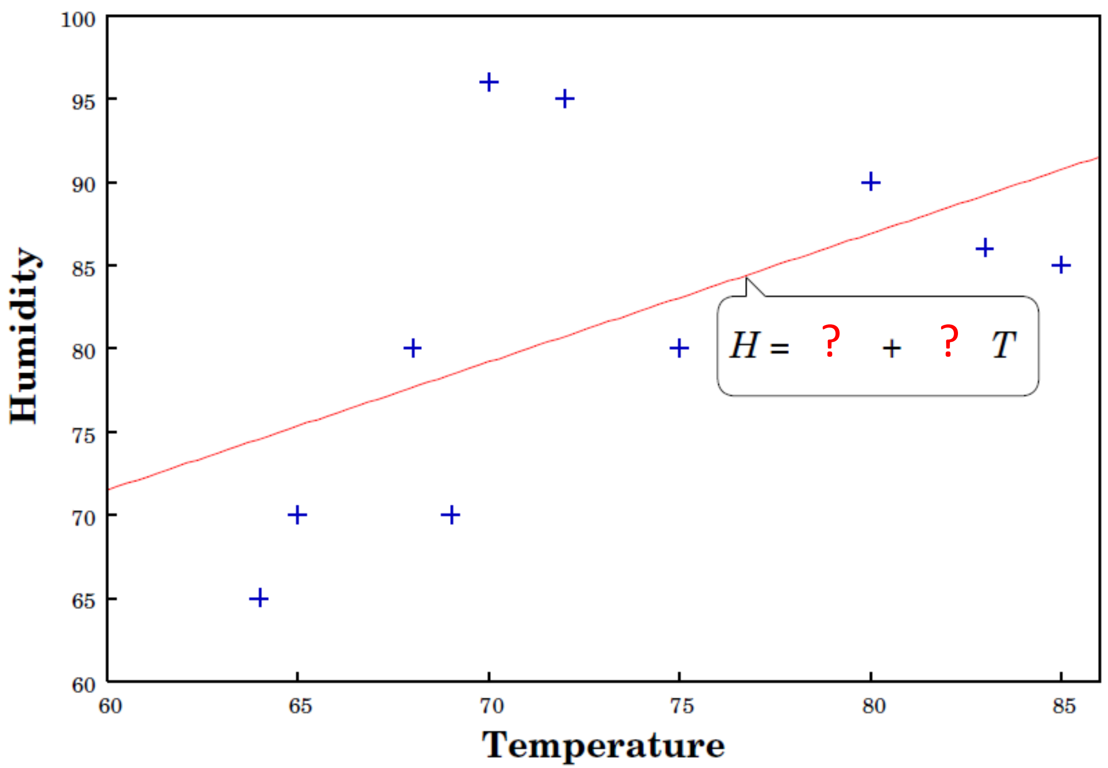
Linear Regression Model

A simple model that tends to require less data and be easier to interpret.

It offers mathematical convenience at the expense of flexibility.

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



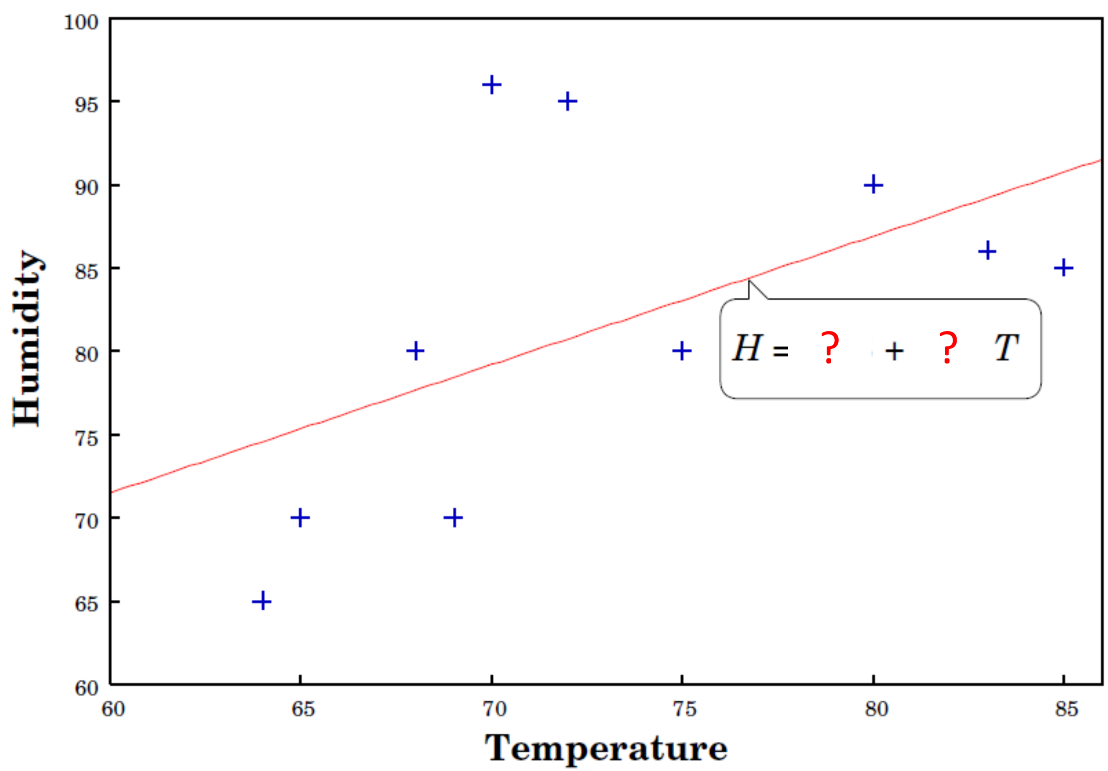
In regression, the task is to predict numeric response (*aka* dependent variable) from features (*aka* predictors or independent variables)

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



Example: Finding parameter values

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

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

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

Basic calculus:
- Write derivative
- Set to zero
- Solve for model

Coordinate descent: guess a, solve for b; solve for a; repeat gives linear regression $a = 25.3, b = 0.77$ (requires many iterations!)

Example: Analytic solution

- Can we do better? 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: $a = 25.3, b = 0.77$
- (Can solve using `numpy.linalg.solve` or equivalent)

Linear regression model

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

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

Here $w_1, \dots, w_m \in \mathbb{R}$ denote weights (model parameters)

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

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

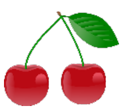
A lowercase symbol in **bold face** indicates a vector; $\boldsymbol{x}'$ denotes transpose

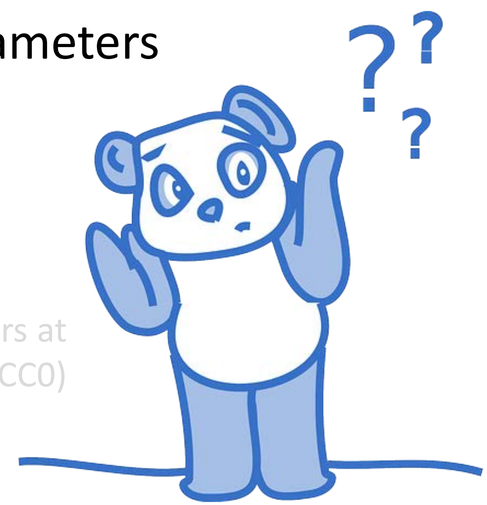
Checkpoint

- Consider a dataset $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)\}$ and some parameter vector $\boldsymbol{w}$ (same dimensionality as $\boldsymbol{x}_i$). Which of the following statements is necessarily true


Each y_i can be expressed as $(\boldsymbol{x}_i)'\boldsymbol{w}$


Given $\boldsymbol{w}$, it is always possible to compute the sum of squared errors for this dataset

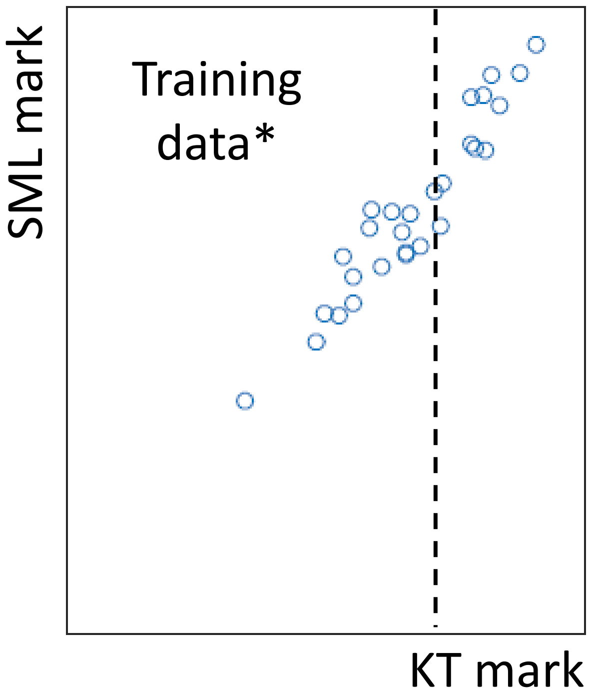

Linear regression model for this data has n parameters

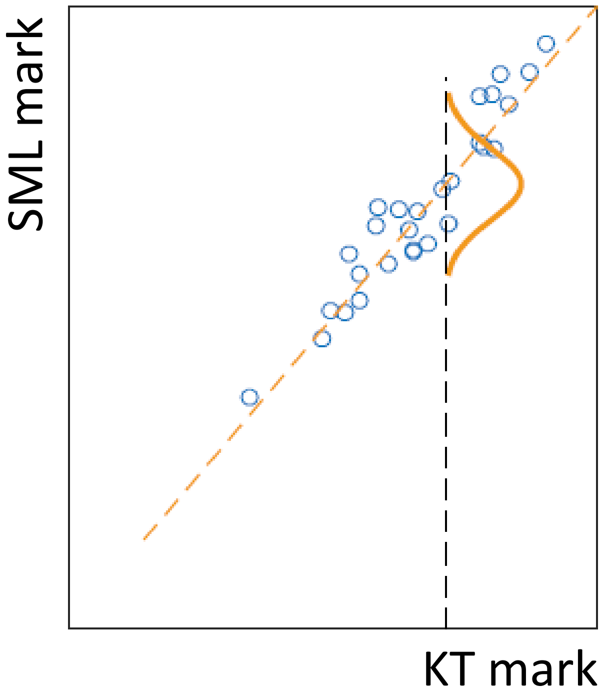


art: OpenClipartVectors at pixabay.com (CC0)

Data is noisy!

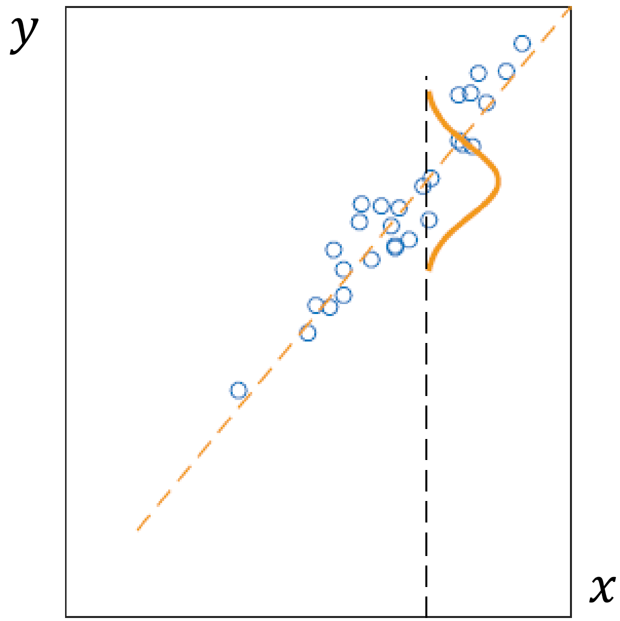
Example: predict mark for Statistical Machine Learning (SML) from mark for Knowledge Technologies (KT)





* synthetic data :)

Regression as a probabilistic model



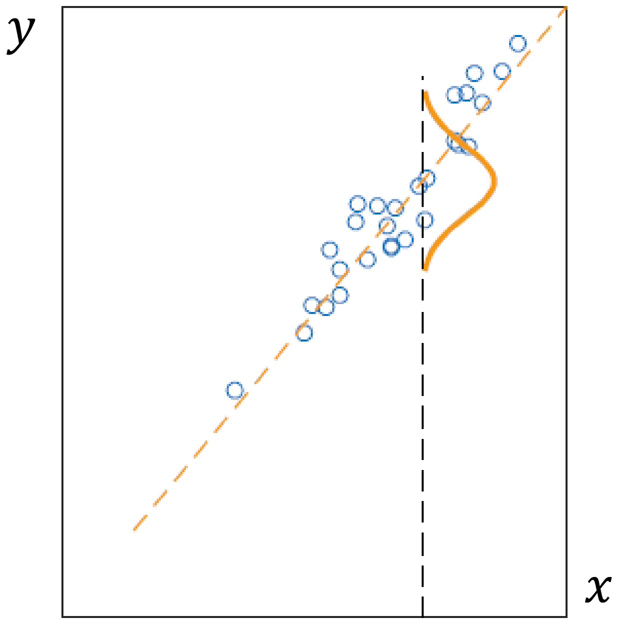
- Assume a probabilistic model: $\mathcal{Y} = \boldsymbol{x}'\boldsymbol{w} + \varepsilon$
  * Here $\mathcal{Y}$ and ε are random variables
  * Variable ε encodes noise
- Next, assume normally distributed noise: $\varepsilon \sim \mathcal{N}(0, \sigma^2)$

this is a squared error!

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

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

Parametric probabilistic model



- Using simplified notation, our model is

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

- Note that parameters are now $\boldsymbol{w}, \sigma^2$

- 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

Maximum likelihood estimation

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

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

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

- "Log trick": Instead of maximising this quantity, maximise the logarithm

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

the sum of squared errors!

here K doesn't depend on $\boldsymbol{w}$

- 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 $n \times (m+1)$ matrix $\boldsymbol{X}$, and $n \times 1$ vector $\boldsymbol{y}$
- The model assumes $\boldsymbol{y} \approx \boldsymbol{X}\boldsymbol{w}$
- To find $\boldsymbol{w}$, minimise the sum of squared errors

$$L = \sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{m} X_{ij} w_j \right)^2$$

- Setting derivative to zero and solving for $\boldsymbol{w}$ yields

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

  - This is a system of equations called the normal equations
  - This system is defined only if the inverse exists

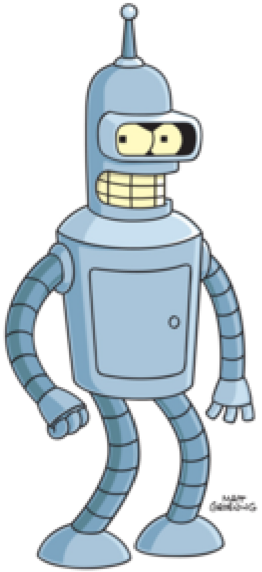
Basic calculus:
- Write derivative
- Set to zero
- Solve for model

An UPPERCASE symbol in bold face indicates a matrix; $\boldsymbol{X}'$ denotes transpose; $\boldsymbol{X}^{-1}$ denotes matrix inverse

Heads up!

- This subject covers a large variety of computational methods
- But there will be several recurring topics, common threads run throughout the entire course
- These topics reflect fundamental aspects of machine learning

- Basis expansion, representation
- Optimisation, loss functions
- Regularisation, overfitting



Regularisation

Process of introducing additional information in order to solve an ill-posed problem or to prevent overfitting (*Wikipedia*)

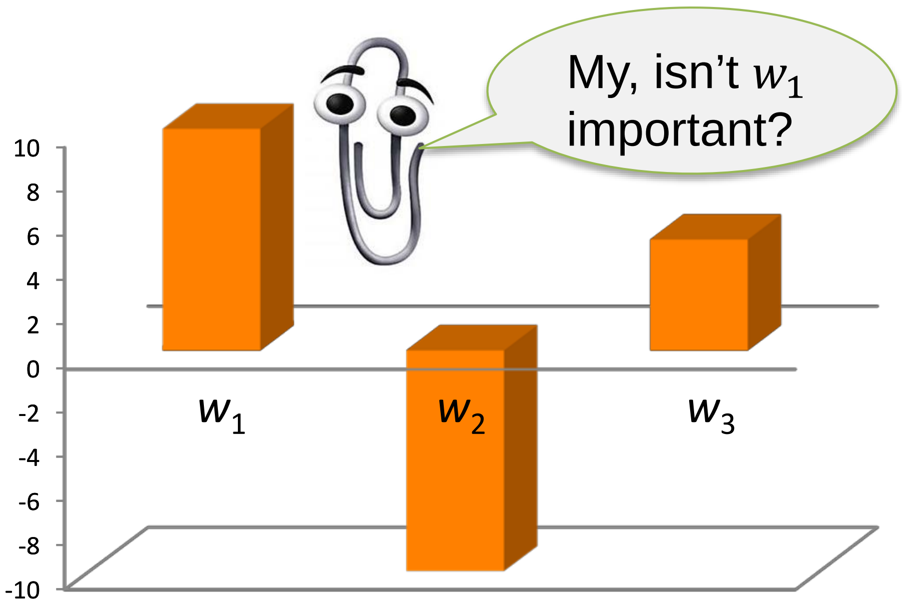
Regularisation

- Major technique, common in Machine Learning
- Addresses one or more of the following related problems
  * Avoid ill-conditioning
  * Introduce prior knowledge
  * Constrain modelling
- This is achieved by augmenting the objective function
- In this lecture: we cover the first two aspects. We will cover more of regularisation throughout the subject

Example 1: Irrelevant features

- Linear model on three features, first two same
  - $\boldsymbol{X}$ is matrix a for $n = 4$ instances (rows)
  - First two columns of $\boldsymbol{X}$ identical
  - Feature 2 is **irrelevant** (or feature 1)
  - Model: $y = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_0$

3	3	7
6	6	9
21	21	79
34	34	2



- Effect of perturbations on model predictions?
  - Add Δ to w_1
  - Subtract Δ from w_2

...identical predictions

...no interpretability

Problems with irrelevant features

- In our example, suppose $[\hat{w}_0, \hat{w}_1, \hat{w}_2, \hat{w}_3]$ is the "best solution"
- But for arbitrary δ solution $[\hat{w}_0, \hat{w}_1 + \delta, \hat{w}_2 - \delta, \hat{w}_3]'$ will lead to the same predictions and to the same sum of squared errors
- The solution is not unique
- One problem is lack of interpretability
- A more serious problem is that the finding the best parameters becomes an ill-posed problem

Irrelevant (co-linear) features in general

- X-treme case: features complete clones
- For linear models, more generally
  - Feature $\boldsymbol{X}_{\cdot j}$ is irrelevant if
  - $\boldsymbol{X}_{\cdot j}$ is a linear combination of other columns

$$\boldsymbol{X}_{\cdot j} = \sum_{l \neq j} \alpha_l \boldsymbol{X}_{\cdot l}$$

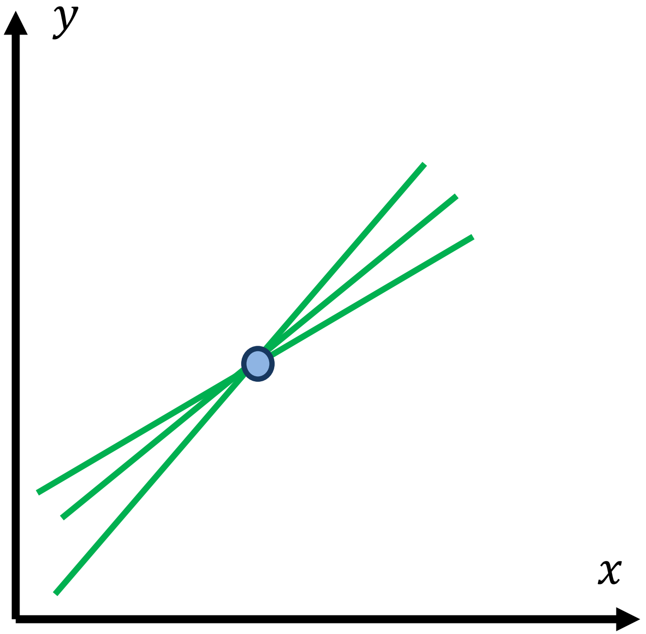
… for some constants α_l

- Even *near*-irrelevance can be problematic
- Not just a pathological x-treme; *easy to happen!*

$\boldsymbol{X}_{\cdot j}$ denotes the j-th column of X

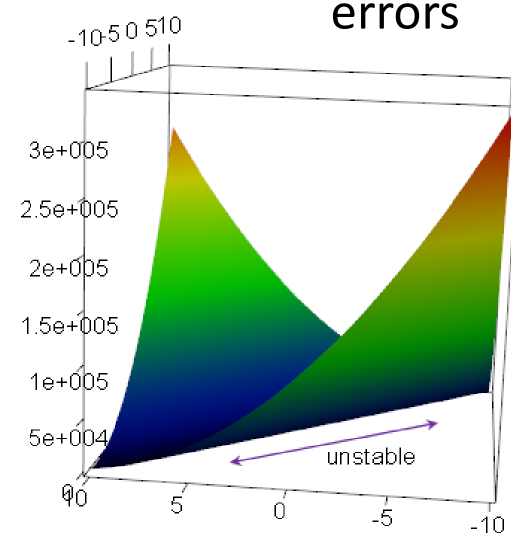
Example 2: Lack of data

- Model is more complex than data
- Extreme example:
  - Model has two parameters (slope and intercept)
  - Only one data point
- Underdetermined system



Ill-posed problems

- In both examples, finding the best parameters becomes an ill-posed problem
- This means that the problem solution is not defined
  - In our case w_1 and w_2 cannot be uniquely identified
- Remember the normal equations solution $\hat{\boldsymbol{w}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}$
- With irrelevant features, $\boldsymbol{X}'\boldsymbol{X}$ has no inverse
- The system of linear equations has more unknowns than equations



convex, but not strictly convex

Side note: L1 and L2 norms

- Throughout the course we will often encounter norms that are functions $\mathbb{R}^n \to \mathbb{R}$ of a particular form
  - Intuitively, norms measure lengths of vectors in some sense
- More specifically, we will often use the L2 norm (*aka* Euclidean distance)

$$\|\boldsymbol{a}\| = \|\boldsymbol{a}\|_2 \equiv \sqrt{a_1^2 + \cdots + a_n^2}$$

- And also the L1 norm (*aka* absolute norm or Manhattan distance)

$$\|\boldsymbol{a}\|_1 \equiv |a_1| + \cdots + |a_n|$$

- For example, the sum of squared errors is a squared norm

$$L = \sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{m} X_{ij} w_j \right)^2 = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2$$

Re-conditioning the problem

- Regularisation: introduce an additional condition into the system
- The original problem is to minimise $\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$
- The regularised problem is to minimise

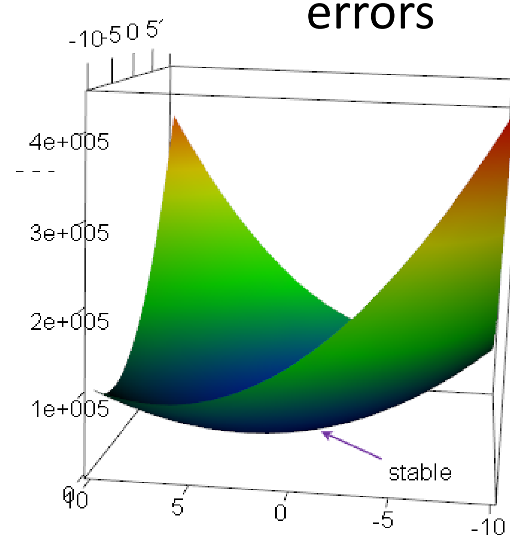
$$\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2 \text{ for } \lambda > 0$$

- The solution is now

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}'\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}'\boldsymbol{y}$$

- This formation is called ridge regression
  - Turns the ridge into a peak

sum of squared errors



w_2

w_1

strictly convex

Regulariser as a prior

- Without regularisation model parameters are found based entirely on the information contained in the training set $\boldsymbol{X}$
- Regularisation essentially means introducing additional information
- Recall our probabilistic model $\mathcal{Y} = \boldsymbol{x}'\boldsymbol{w} + \varepsilon$
    * Here $\mathcal{Y}$ and ε are random variables, where ε denotes noise
- Now suppose that $\boldsymbol{w}$ is also a random variable (denoted as $\mathcal{W}$) with a normal prior distribution

$$\mathcal{W} \sim \mathcal{N}(0, \lambda^2)$$

Regulariser as a prior

- Now suppose that $\boldsymbol{w}$ is also a random variable (denoted as $\mathcal{W}$) with a normal prior distribution
$$\mathcal{W} \sim \mathcal{N}(0, \lambda^2)$$
- Prior = our initial expectations before seeing data
- In the above prior, we expect small weights and that no one feature dominates
  * Is this always appropriate? Consider data centring and scaling
- We could encode much more elaborate problem knowledge

Computing posterior using Bayes rule

- The prior is then used to compute the posterior

posterior | likelihood | prior

$$p(\boldsymbol{w}|\boldsymbol{X},\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{y}|\boldsymbol{X})}$$

marginal likelihood

- Instead of maximum likelihood (MLE), take ***maximum a posteriori*** estimate (MAP)

- Apply log trick, so that

$$\log(posterior) = \log(likelihood) + \log(prior) \cancel{- \log(marg)}$$

- Arrive at the problem of minimising

$$\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2$$

this term doesn't affect optimisation

This lecture

- Linear regression
  - Worked example and the model
  - Regression as a probabilistic model
- Regularisation
  - Irrelevant features and an ill-posed problem
  - Regulariser as a prior