Lecture 3. Linear Regression. Optimisation.

COMP90051 Statistical Machine Learning

Semester 2, 2018
Lecturer: Ben Rubinstein

THE UNIVERSITY OF
MELBOURNE

This lecture

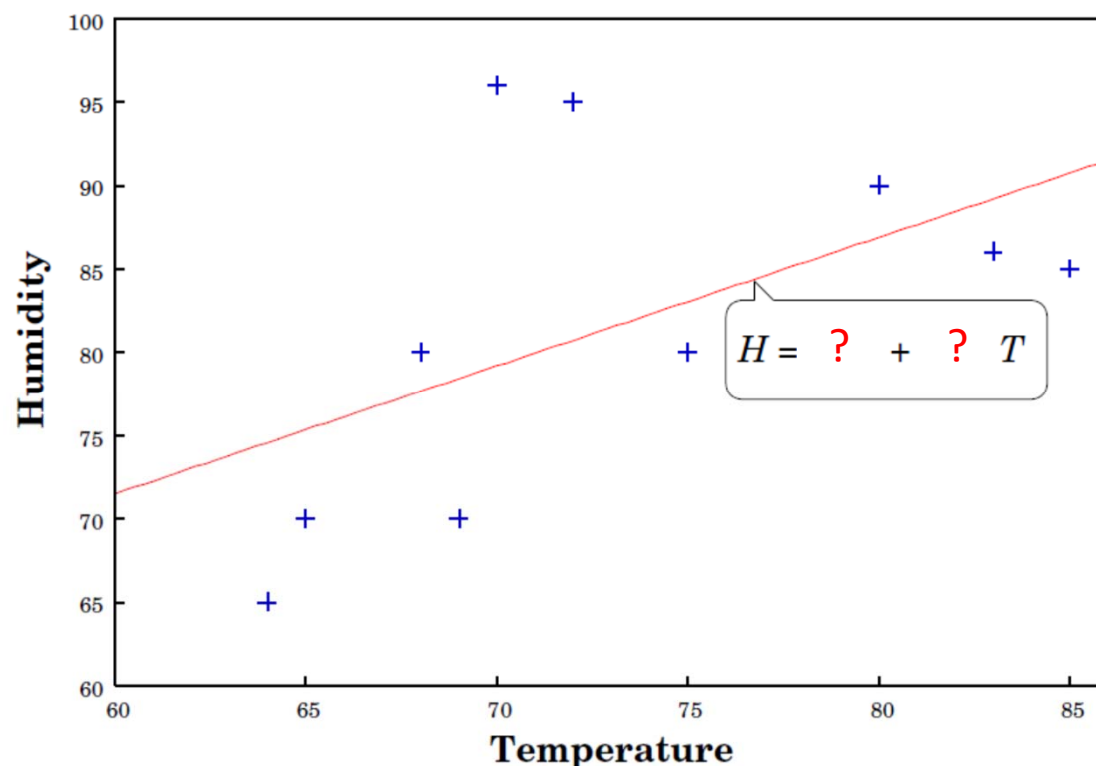- ## Linear regression

  - ∗ 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

- ## Optimisation for ML

  - ∗ Analytic solutions

  - ∗ Gradient descent

  - ∗ Convexity

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

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

$H = ? + ? T$
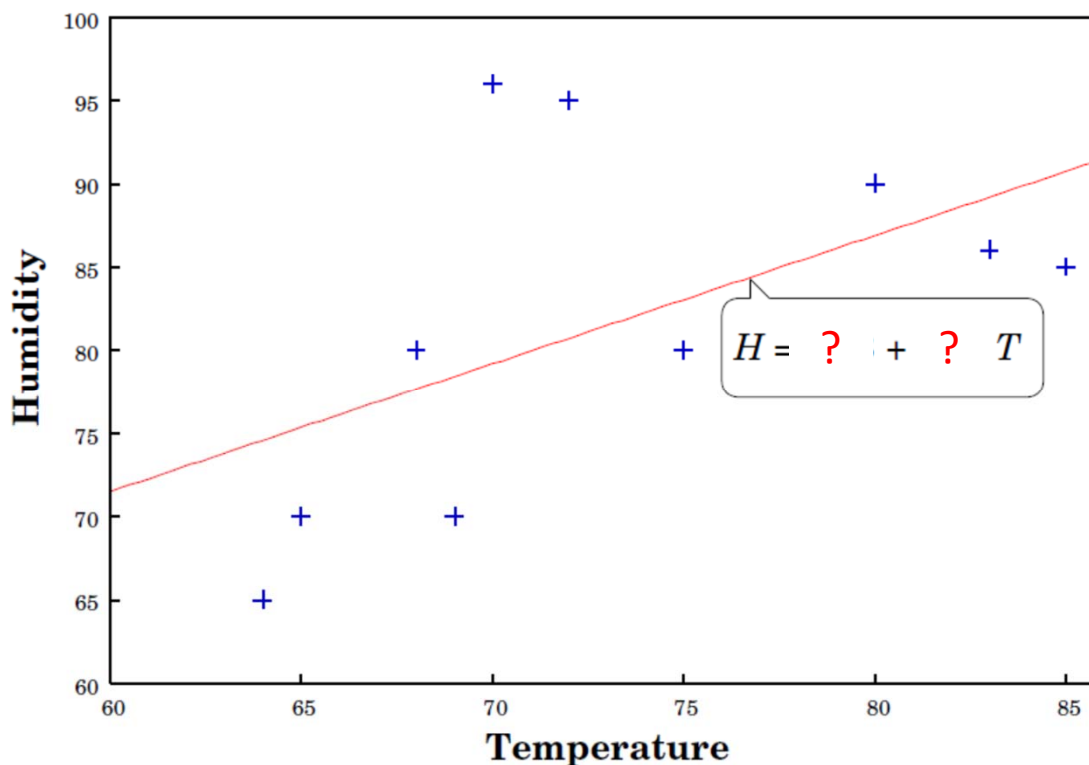
Example: Minimise Sum Squared Errors

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

set derivatives to zero:

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

> **Basic calculus:**
> - Write derivative
> - Set to zero
> - Solve for model
>
> Will cover again later

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

Can we be more systematic?

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` or sim.)

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 $w_1, \ldots, w_m \in \mathbb{R}$ 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   8
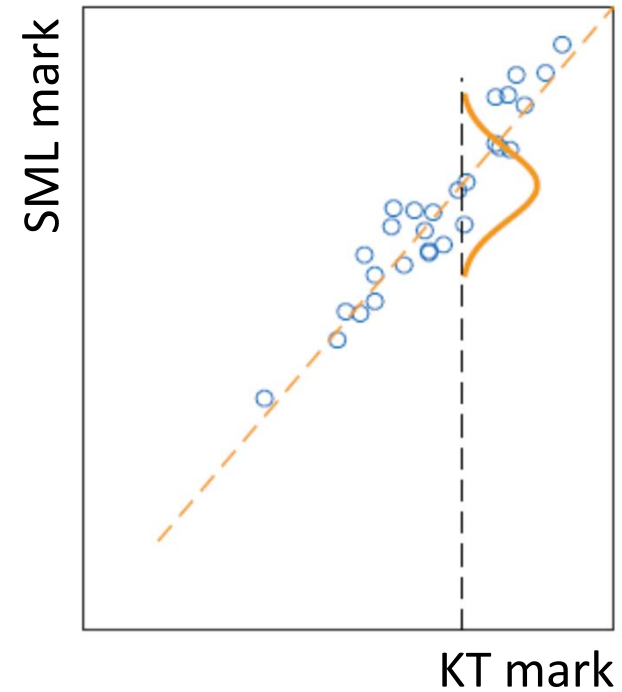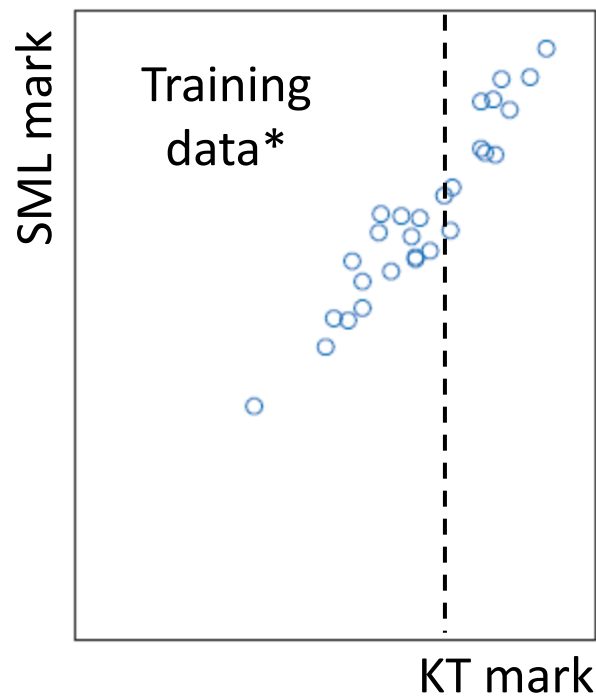
Linear Regression via Frequentist Probabilistic Model

Max Likelihood Estimation
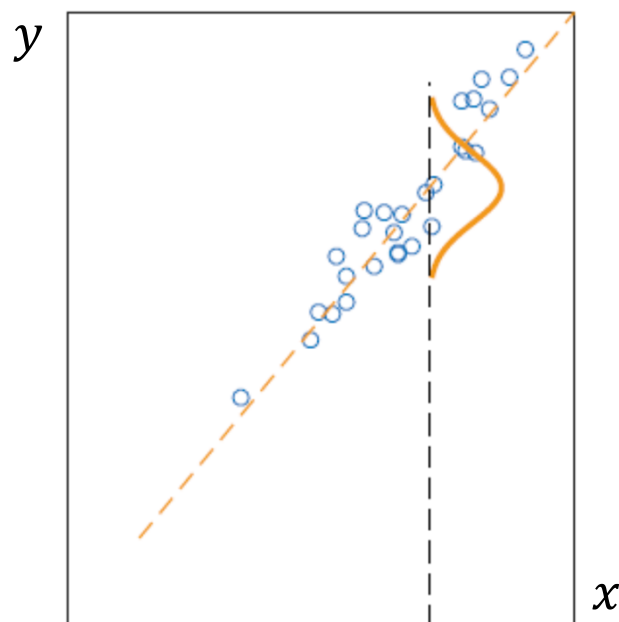
Data is noisy!

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

Regression as a probabilistic model

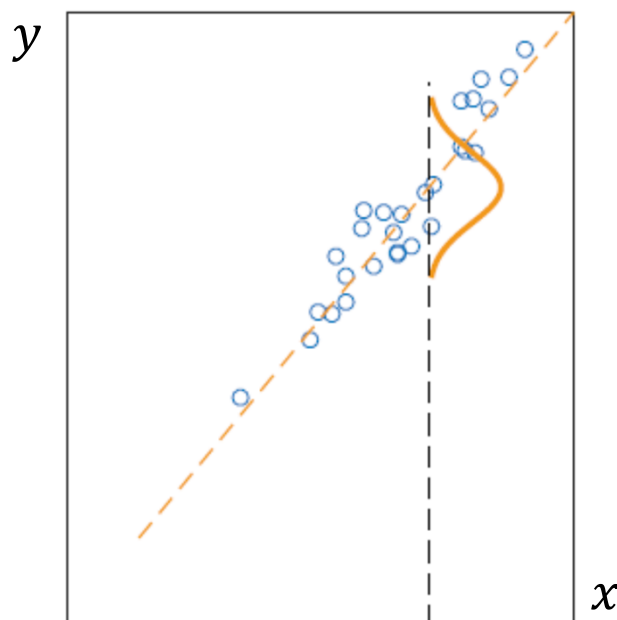- Assume a probabilistic model: $Y = X'w + \varepsilon$
  - ∗ Here X, Y and ε are r.v.'s
  - ∗ Variable ε encodes noise

- Next, assume Gaussian noise (indep. of X): $\varepsilon \sim \mathcal{N}(0, \sigma^2)$

- 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 (see L2.20)

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

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

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 (why?)

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

the sum of squared errors!

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

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

Non-linear Continuous Optimisation

Brief summary of a few basic optimisation methods vital for ML

Optimisation formulations in ML

- Training = Fitting = Parameter estimation

- Typical formulation

$$\widehat{\boldsymbol{\theta}} \in \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmin}} L(data, \boldsymbol{\theta})$$

  * argmin because we want a minimiser not the minimum
    - Note: argmin can return a set (minimiser not always unique!)
  * Θ denotes a model family (including constraints)
  * L denotes some objective function to be optimised
    - E.g. MLE: (conditional) likelihood
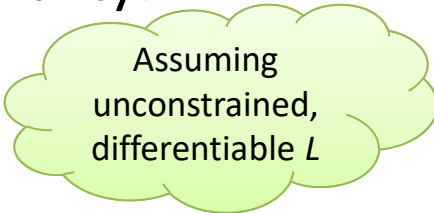   - E.g. Decision theory: (regularised) empirical risk

15

Two solution approaches

- Analytic (*aka* closed form) solution
  - * Known only in limited number of cases
  - * Use 1st-order necessary condition for optimality:

  $$\frac{\partial L}{\partial \theta_1} = \cdots = \frac{\partial L}{\partial \theta_p} = 0$$
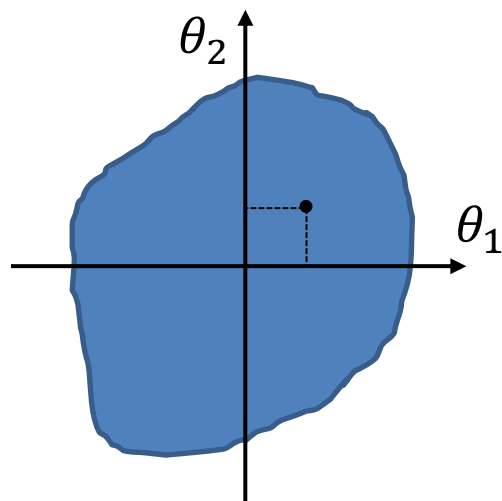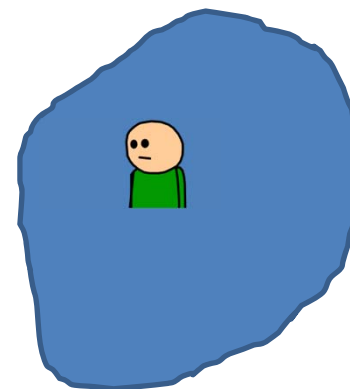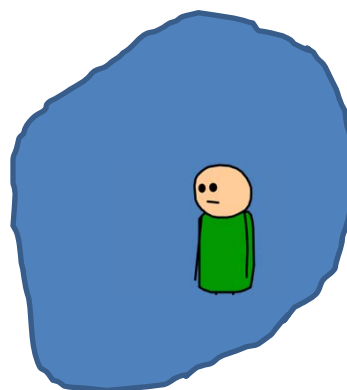
  Assuming unconstrained, differentiable *L*

- Approximate iterative solution
  1. Initialisation: choose starting guess $\boldsymbol{\theta}^{(1)}$, set $i = 1$
  2. Update: $\boldsymbol{\theta}^{(i+1)} \leftarrow SomeRule[\boldsymbol{\theta}^{(i)}]$, set $i \leftarrow i + 1$
  3. Termination: decide whether to Stop
  4. Go to Step 2
 5. Stop: return $\widehat{\boldsymbol{\theta}} \approx \boldsymbol{\theta}^{(i)}$

Finding the deepest point
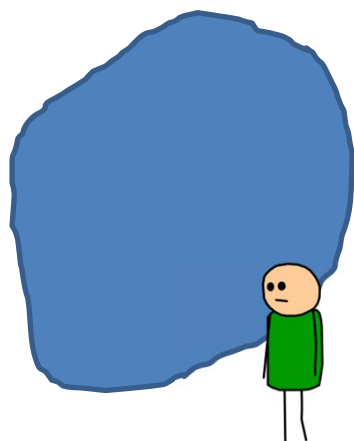
- In this example, a model has 2 parameters

- Each location corresponds to a particular combination of parameter values

- Depth indicates objective value (e.g. loss) of that candidate model on data

Coordinate descent

- Suppose $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_K]'$

1. Choose $\boldsymbol{\theta}^{(1)}$ and some T

2. For i from 1 to T*
   1. $\boldsymbol{\theta}^{(i+1)} \leftarrow \boldsymbol{\theta}^{(i)}$
   2. For j from 1 to K
      1. Fix components of $\boldsymbol{\theta}^{(i+1)}$, except j-th component
      2. Find $\hat{\theta}_j^{(i+1)}$ that minimises $L\left(\theta_j^{(i+1)}\right)$
      3. Update j-th component of $\boldsymbol{\theta}^{(i+1)}$

3. Return $\widehat{\boldsymbol{\theta}} \approx \boldsymbol{\theta}^{(i)}$
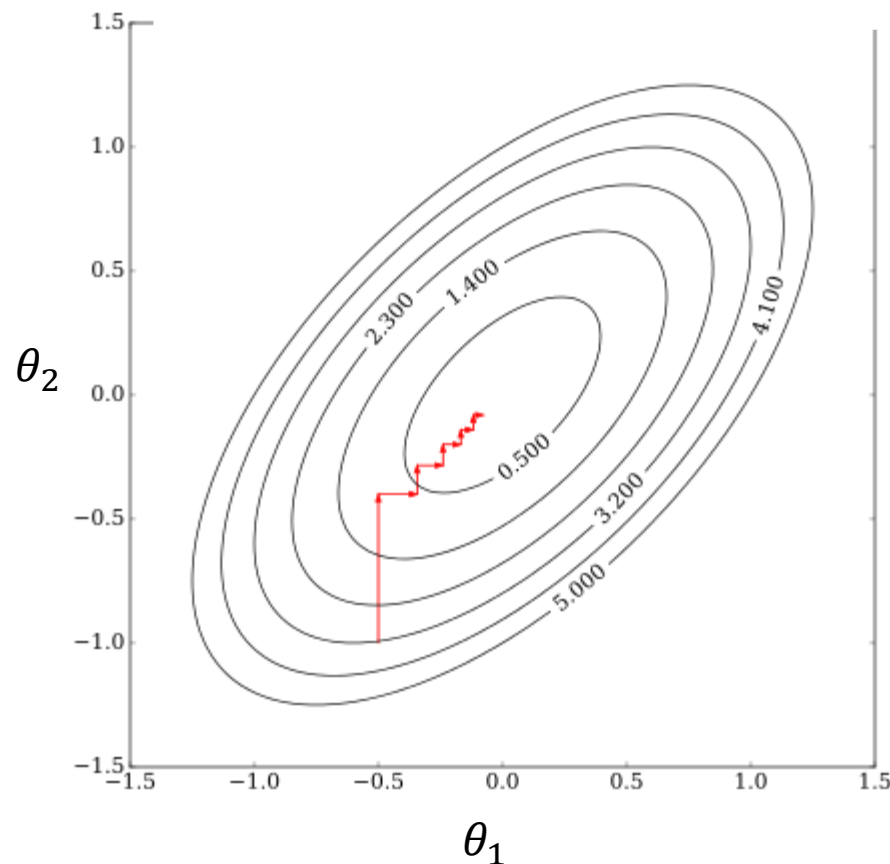
*Other stopping criteria can be used

Reminder: The gradient

- Gradient at $\boldsymbol{\theta}$ defined as $\left[\frac{\partial L}{\partial \theta_1}, \ldots, \frac{\partial L}{\partial \theta_p}\right]'$ evaluated at $\boldsymbol{\theta}$

- The gradient points to the direction of maximal change of $L(\boldsymbol{\theta})$ when departing from point $\boldsymbol{\theta}$

- Shorthand notation

  * $\nabla L \overset{\text{def}}{=} \left[\frac{\partial L}{\partial \theta_1}, \ldots, \frac{\partial L}{\partial \theta_p}\right]'$ computed at point $\boldsymbol{\theta}$
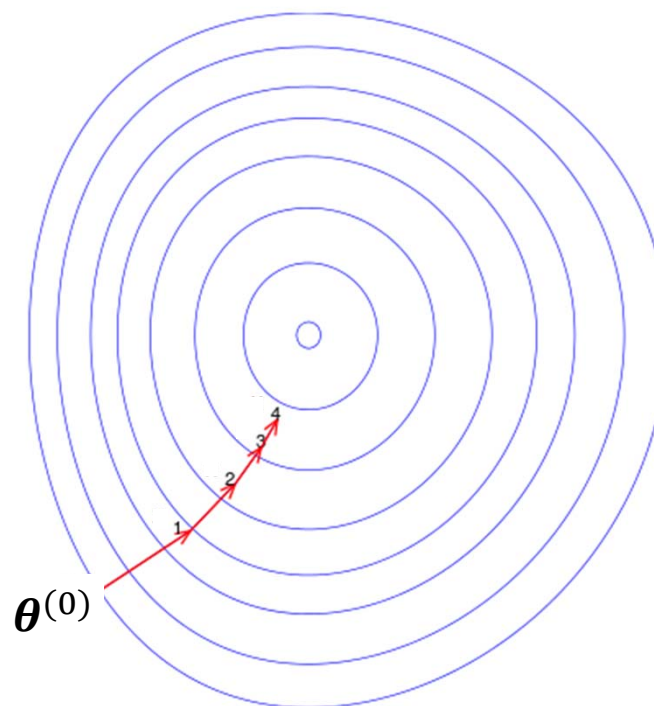
  * Here ∇ is the "nabla" symbol

Harps, p. 984.

Gradient descent

1. Choose $\boldsymbol{\theta}^{(1)}$ and some T

2. For i from 1 to T*
   1. $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - \eta \boldsymbol{\nabla} L(\boldsymbol{\theta}^{(i)})$

3. Return $\widehat{\boldsymbol{\theta}} \approx \boldsymbol{\theta}^{(i)}$

- Note: η is dynamically updated in each step

- Variants: Stochastic GD, Mini batches, Momentum, AdaGrad, ….

Assuming L is differentiable

$\boldsymbol{\theta}^{(0)}$

*Other stopping criteria can be used

Convex objective functions

- 'Bowl shaped' functions

- Informally: if line segment between any two points on graph of function lies above or on graph

- Formally $f: D \rightarrow \mathbf{R}$ is convex
  if $\forall \boldsymbol{a}, \boldsymbol{b} \in D, t \in [0,1]$:
  $$f(t\boldsymbol{a} + (1 - t)\boldsymbol{b}) \leq tf(\boldsymbol{a}) + (1 - t)f(\boldsymbol{b})$$
  Strictly convex if inequality is strict (<)

- Gradient descent on (strictly) convex function guaranteed to find a (unique) global minimum!

21
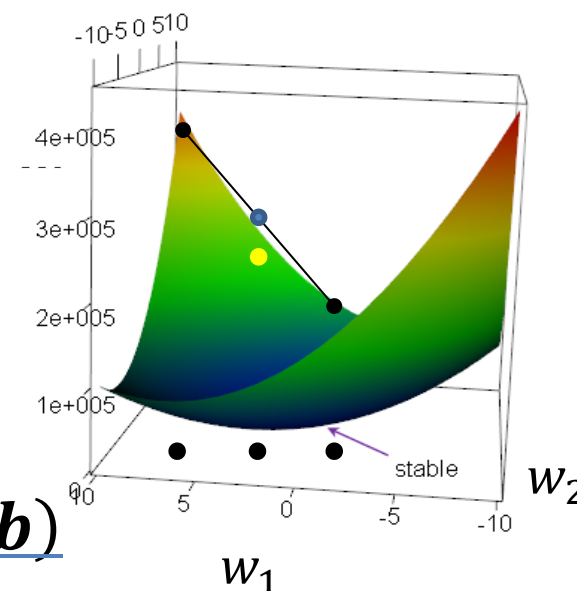
L_1 and L_2 norms

- 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
  * Often component of objectives or stopping criteria in optimisation-for-ML

- More specifically, we will often use the L_2 norm (*aka* Euclidean distance)

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

- And also the L_1 norm (*aka* absolute norm or Manhattan distance)

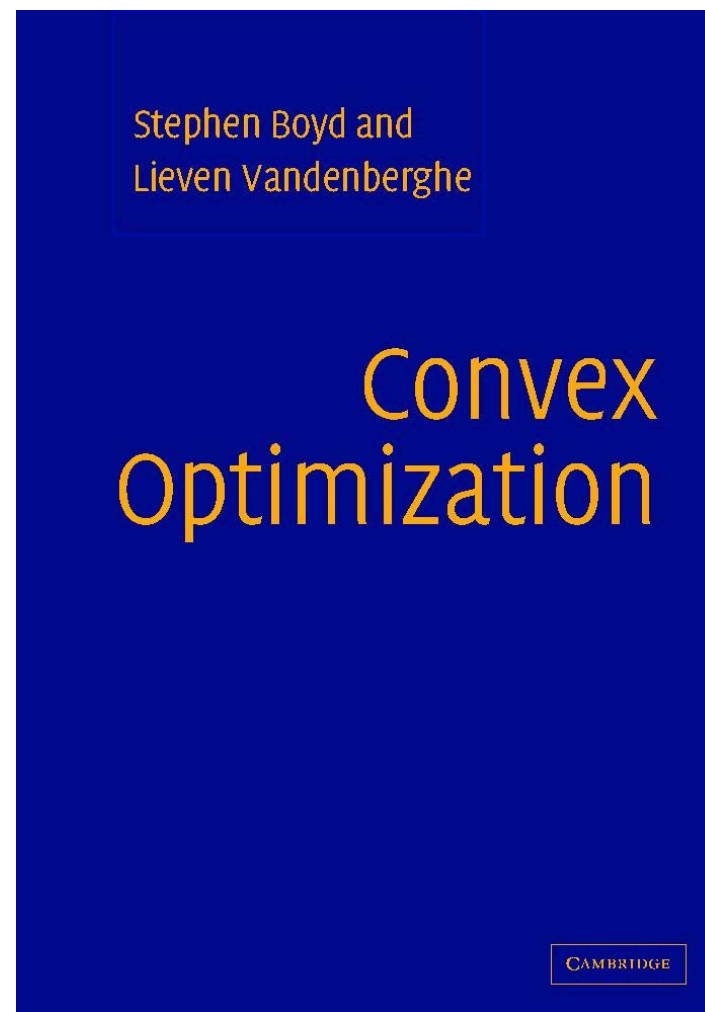$$\|\boldsymbol{a}\|_1 \equiv |a_1| + \cdots + |a_n|$$

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

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

22

…And much much more

- What if you have constraints?

- What about speed of convergence?

- Is there anything faster than gradient descent (yes, but can be expensive)

- Do you really need differentiable objectives? (no)

- Are there more tricks? (hell yeah!)

Stephen Boyd and Lieven Vandenberghe

Convex Optimization

CAMBRIDGE

Free at http://web.stanford.edu/~boyd/cvxbook/

One we've seen: Log trick

- Instead of optimising $L(\theta)$, try convenient $\log L(\theta)$

- Why are we allowed to do this?

- Strictly monotonic function: $a > b \implies f(a) > f(b)$
  - ∗ Example: log function!

- **Lemma**: Consider any objective function $L(\theta)$ and any strictly monotonic f. θ^* is an optimiser of $L(\theta)$ if and only if it is an optimiser of $f(L(\theta))$.
  - ∗ Proof: Try it at home for fun!

Linear Regression Optimisation

For either MLE/decision-theoretic derivations

Method of least squares

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

- 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 $n \times (m+1)$ 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

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

- Setting derivative to zero and solving for $\boldsymbol{w}$ yields
$$\widehat{\boldsymbol{w}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}$$

* 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   26

Summary

- ## Linear regression

  * Probabilistic frequentist derivation

  * Decision-theoretic frequentist derivation

- ## Optimisation for ML

Workshop #2: DIY lin. regression, Bayesian coin flipping

Next time: logistic regression - a linear model for classification; basis expansion for non-linear extensions