

Introduction

ST420 Lecture 1

Richard Everitt

Practical considerations

- Lecturer: Richard Everitt
- Email: richard.everitt@warwick.ac.uk
- Classes:
 - video lectures on moodle;
 - weekly in-person "summary" lectures.
- Course materials:
 - slides for each lecture;
 - problem sheets.
- CATS: 15
- Assessment:
 - 80% April examination;
 - 20% from two assignments (10% each).

Getting help

- Use the discussion forum on moodle
 - please read the forum regularly!
- Send me an email.
- Come to my office hours (see moodle).

Recommended books

1. Chris Bishop, [Pattern Recognition and Machine Learning](#), Springer, 2006.
 2. Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, [An Introduction to Statistical Learning with Applications in R](#), Springer, 2013.
 3. Trevor Hastie, Robert Tibshirani and Jerome Friedman, [The Elements of Statistical Learning: Data Mining, Inference and Prediction](#), Springer, 2009.
- ISLR and ESL cover similar material. ISLR is an introductory textbook, whilst ESL covers more topics in greater depth. PRML discusses Bayesian approaches in more depth.

What will we cover?

- Supervised learning
 - Regression (linear, non-parametric, GLMs, GAMs)
 - Classification (linear discriminant analysis, logistic regression, neural networks, support vector machines)
 - Combining classifiers
- Statistical learning theory
 - Mathematical foundations of machine learning
- Big models and big data
 - Regularisation (ridge regression, lasso, Bayesian approaches)
 - Curse of dimensionality
 - Bayesian inference in high dimensions

Immediate plans

- Introduce supervised learning, using simple examples in regression and classification.
- Discuss issues such as the bias-variance trade-off.
- Then look at more advanced methods for regression and classification.

Buzz words

- Baron Schwartz (@xaprb) on twitter:
"When you're fundraising, it's AI
When you're hiring, it's ML
When you're implementing, it's linear regression"
- It is difficult to define the difference between:
 - statistics;
 - machine learning;
 - artificial intelligence;
 - data science.

What is statistics?

- From Wikipedia:
 - concerns the collection, organization, displaying, analysis, interpretation and presentation of data.
- History of the subject goes back over 1,000 years, being developed hand in hand with probability theory.

What are machine learning and AI?

- Terms from computer science.
- AI from Wikipedia:
 - "The field of AI research was born at a workshop at Dartmouth College in 1956, where the term Artificial Intelligence was coined by John McCarthy to distinguish the field from cybernetics."
- Machine learning:
 - Roughly speaking: having a computer program learn from experience.
 - Motivated by AI.
- Not necessarily statistical, but many (not all!) of the methods are reinventions of statistical techniques.
- Has pushed statistics in interesting new directions.

Artificial intelligence: AlphaGo

AlphaGo Official Trailer



What is data science?

- Could be seen as originating in John Tukey's 1962 paper [The Future of Data Analysis](#), which coined the term "data analysis".
 - different focus to mathematical statistics.
- Has become popular in recent years due to the large volume of data stored on hard drives.
- See David Donoho on [50 years of Data Science](#).

Machine learning overview

- AI is usually focussed on automatically **making decisions** based on data.
- Machine learning is usually focussed on **prediction**.
- Types of machine learning:
 - Supervised learning, with **labelled** data:
 - Regression;
 - Classification.
 - Unsupervised learning, with **unlabelled** data:
 - Clustering;
 - Dimensionality reduction;
 - Density estimation.
 - Semi-supervised learning, where some data are labelled and some are not.
 - Reinforcement learning: learning how to take actions to maximise some award.

Example: data

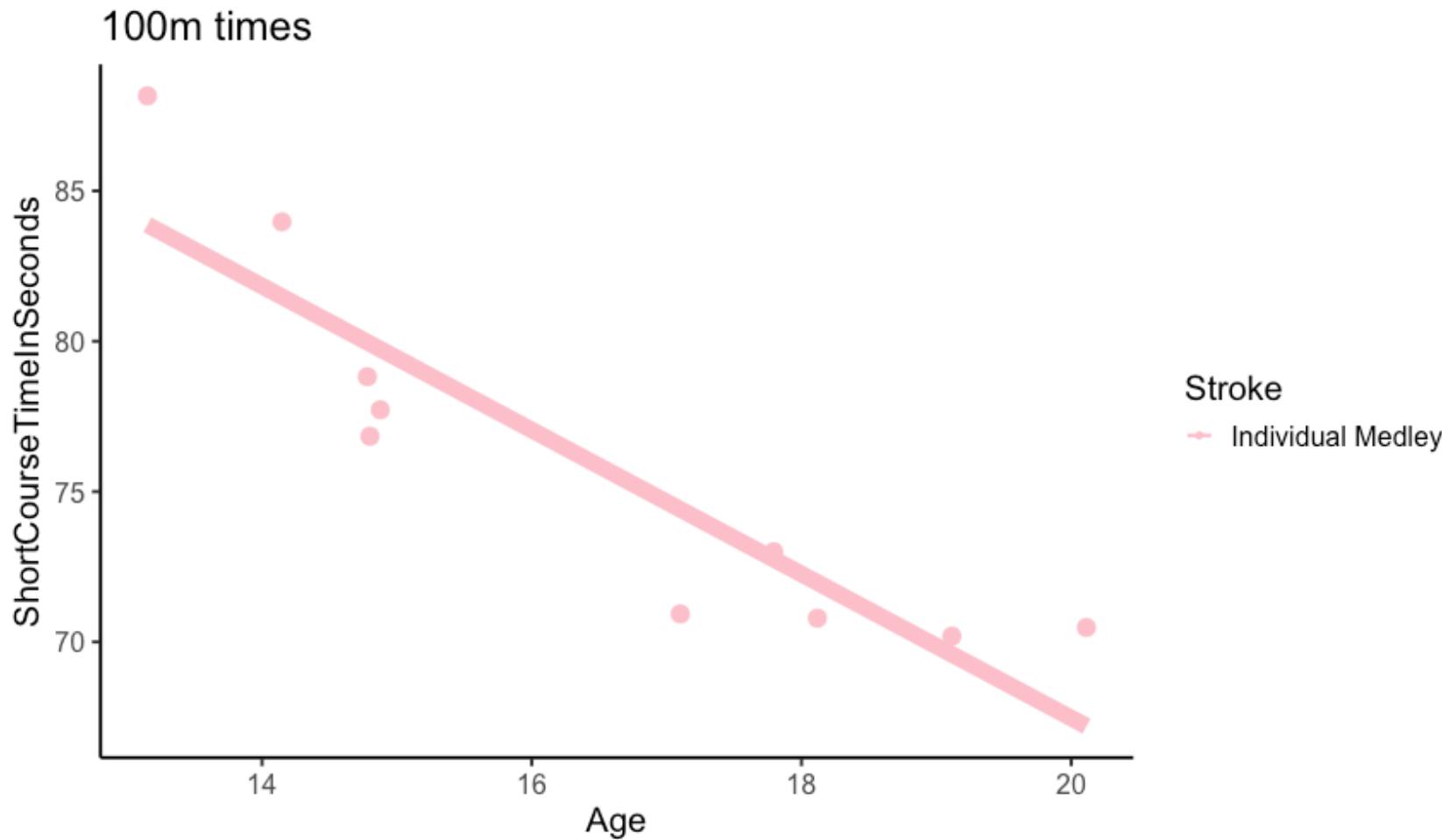
- Throughout the module, a dataset has n rows (individual instances) and p columns (variables).

...1	DATE	EVENT	VENUE	CLASS	STROKE	DISTANCE	TIME	POSITION	TOTAL	POOL	NOTES	IN LENGTH	RACE
1	1991-04-29	Tewkesbury Inter Club	Cascades, Tewkesbury Gala	12	Freestyle	50m	0.37.56	2	5	25m	NA		
2	1991-04-29	Tewkesbury Inter Club	Cascades, Tewkesbury Gala	12	Backstroke	25m	0.18.45	?	?	25m	Medley relay		
3	1991-04-29	Tewkesbury Inter Club	Cascades, Tewkesbury Gala	12	Freestyle	25m	0.16.67	?	?	25m	Freestyle relay		
4	1991-06-29	Tewkesbury Club	Cascades, Tewkesbury Championships	12	Backstroke	100m	1.41.30	?	?	25m	NA		

Supervised learning

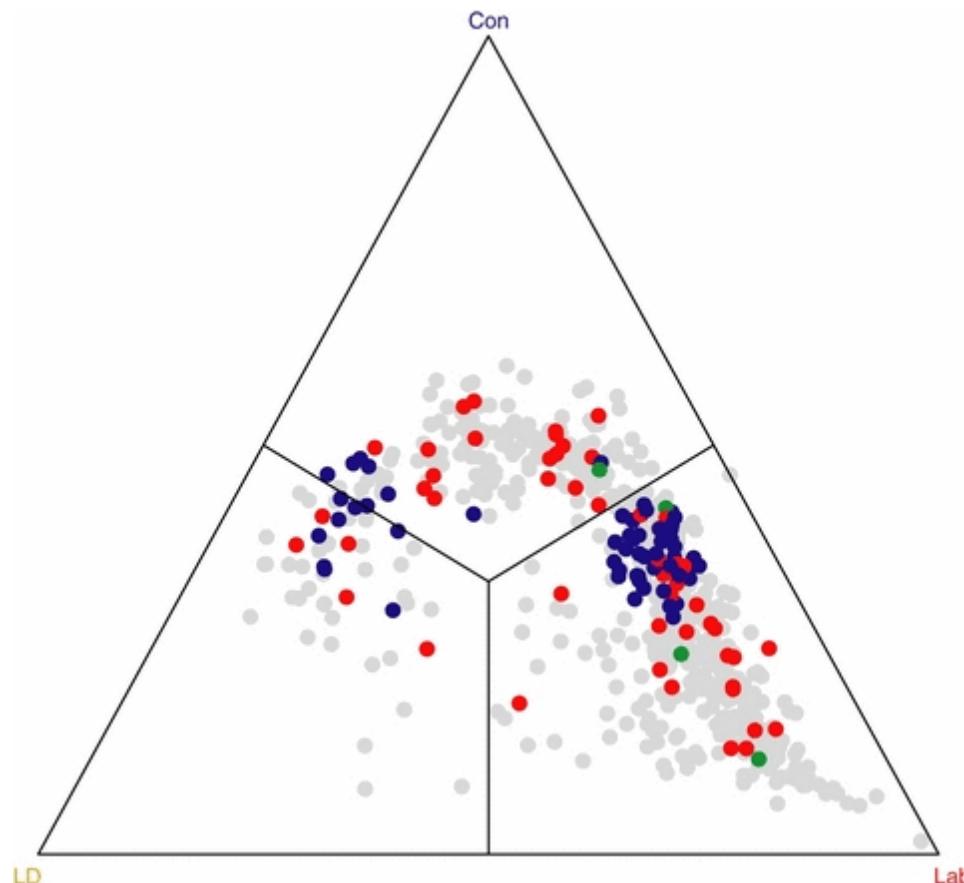
- Let $X = (X^{(1)}, \dots, X^{(p)})$ be a vector of predictors and Y be the response.
- The task in supervised learning is to estimate a function \hat{f} that predicts Y from X
 - usually based on observing n realisations of (X, Y) .
- Two flavours:
 - **Regression**, when Y is continuous;
 - **Classification**, when Y is discrete.

Regression: simple example

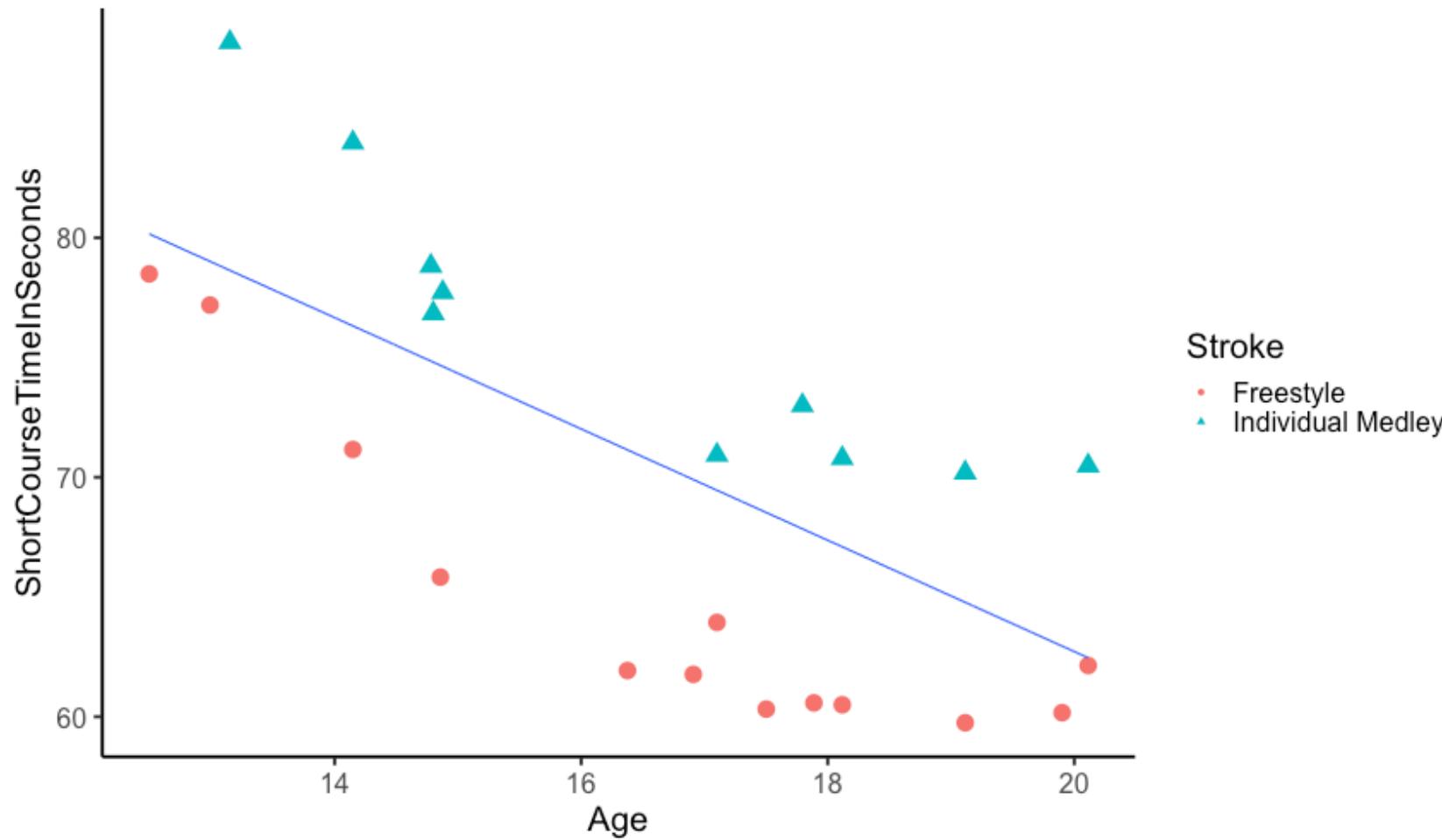


Regression: predicting election results

- Curtice, J and Firth, D (2008). Exit polling in a cold climate: The BBC/ITV experience in Britain in 2005. Vote shares of the three main parties in the UK in 2005.



Classification: simple example



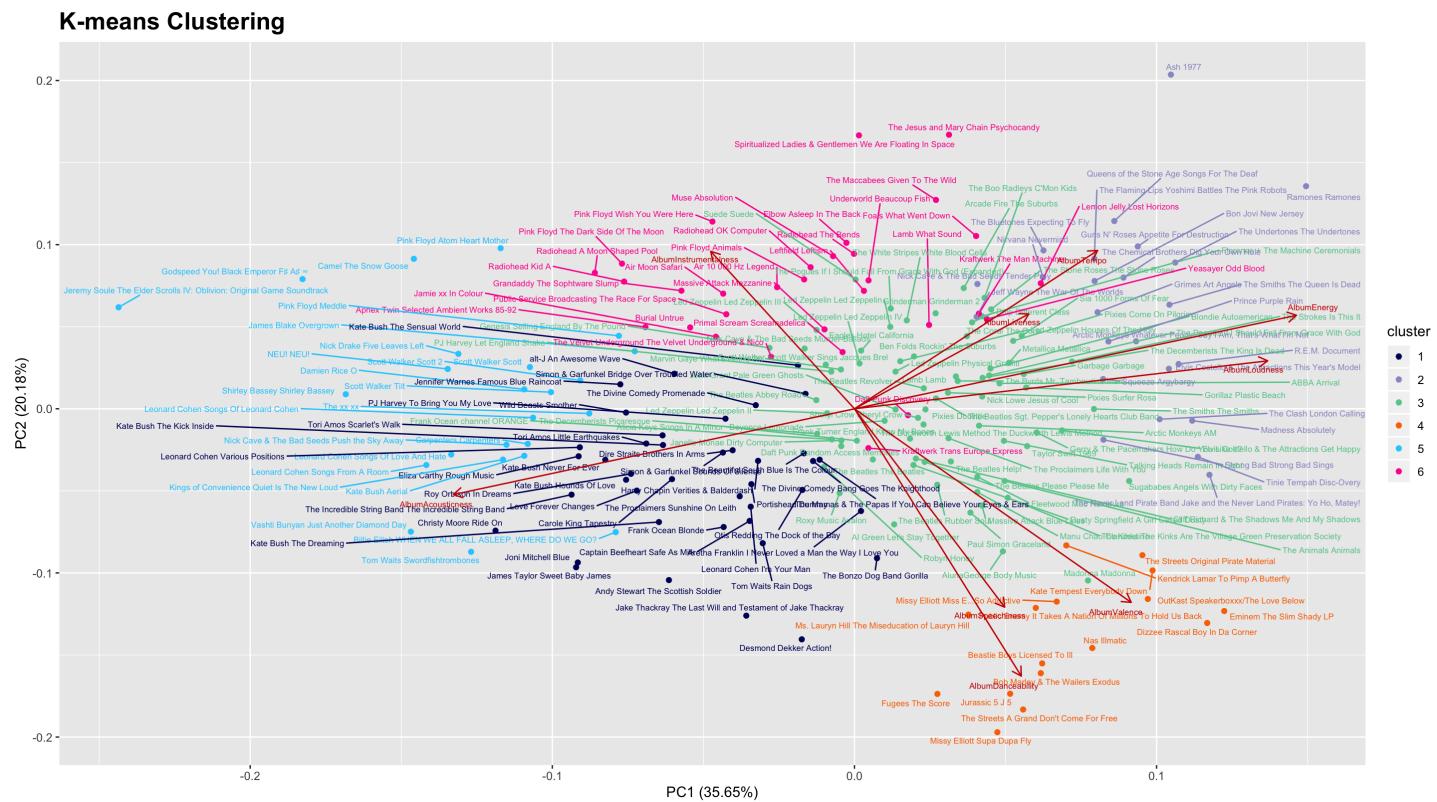
Classification: classifying people and cars

Video Analytics based Object Classification using Deep Learning



Aside: unsupervised learning

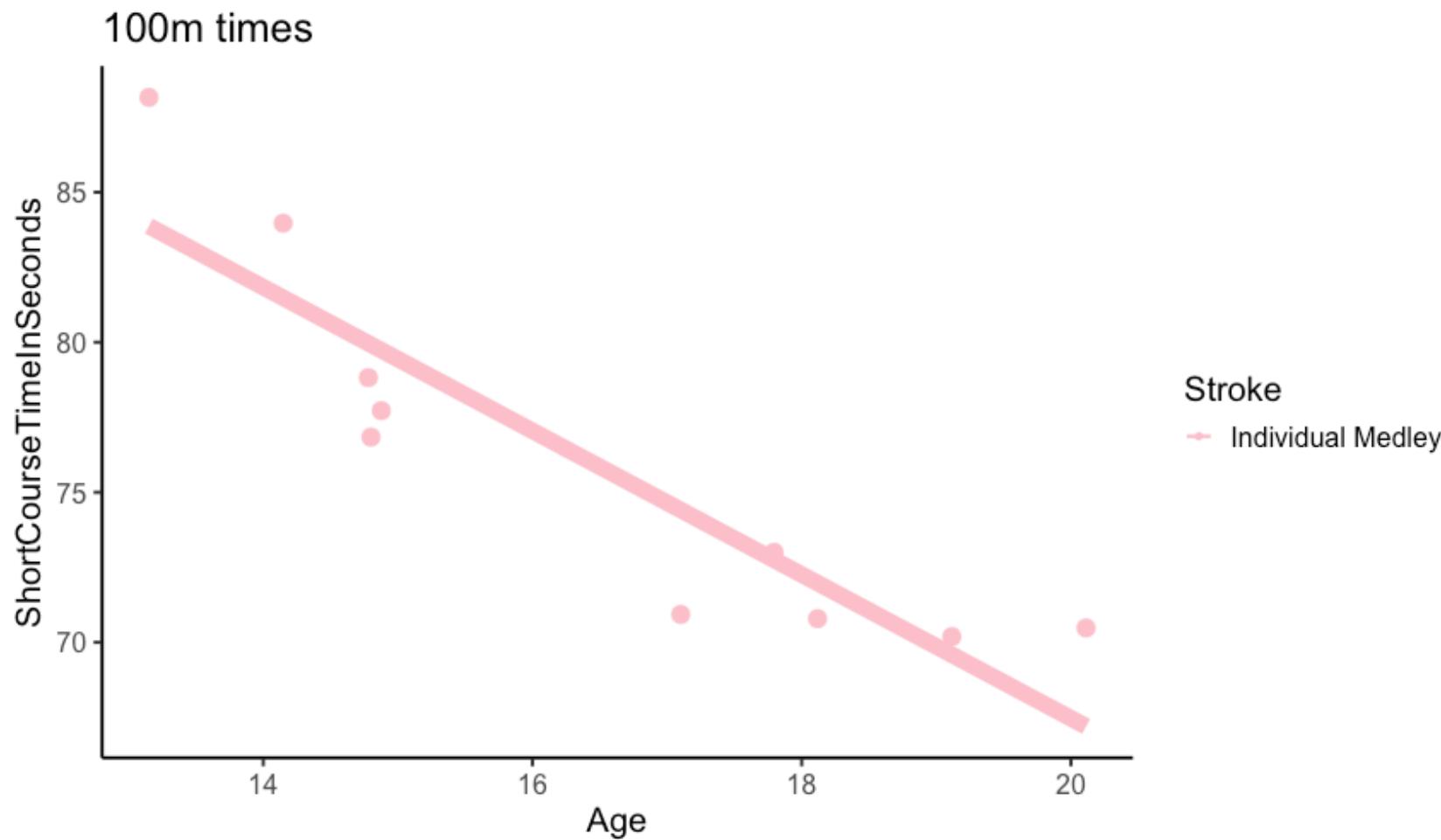
- Finding structure in "unlabelled" data (i.e. we only observe X).
 - Structure could mean clusters, or projecting onto a lower dimensional space that still captures the important features of the data.



Aside: semi-supervised learning

- We observe all X , some, but not all, of which have an associated Y .
- Usually the same goals as supervised learning, but where unlabelled data are used to improve predictions.
- Neither unsupervised nor semi-supervised learning are covered in this module.

Linear regression



Why estimate f ?

- Prediction
 - Suppose we have an estimate \hat{f} and a new data point X . We may predict a corresponding value of Y using
$$\hat{Y} = \hat{f}(X).$$
 - In this case \hat{f} is a "black box"; we are not concerned with the details of \hat{f} , only that it gives good predictions.
- Inference
 - We wish to understand the relationship between Y and X .
 - Which predictors are important?
 - What is the nature of the relationship between Y and the predictors?

How do we estimate f ?

- Choose a class of models to search through
 - For linear regression (where $p = 1$, $X = X^{(1)}$ and $X^{(1)}, Y \in \mathbb{R}$), we have the model
$$Y = \beta_0 + \beta_1 X^{(1)}.$$
 - Finding the estimate \hat{f} boils down to finding estimates $\hat{\beta}_0$ and $\hat{\beta}_1$.
- Search through this class of models to find \hat{f} .
 - To do this we need some criterion by which we judge some choices to be better than others.
 - For linear regression, the most common idea is to choose $\hat{\beta}_0$ and $\hat{\beta}_1$ using *least squares*.
 - That is, to look at the sum of the squared errors of the predictions, and to choose $\hat{\beta}_0$ and $\hat{\beta}_1$ such that this is minimised.
 - Can be done analytically in this situation, but this is not always the case.

Evaluation

- Our \hat{f} is constructed to explain the relationship between Y and X in the data we observe.
- Will it *generalise* to as yet unobserved data?
- To test this usually we hold back part of our data as *test* data. We use the rest, the *training* data, to find \hat{f} .
- We then assess the accuracy of \hat{f} by looking at its performance on the test data.

Immediate plan: extended version

- In lecture 2 we review/introduce some fundamental techniques
 - What are the models?
 - How do we fit them?
- We will begin to encounter some of the important issues in the module.
- What if linear regression/classification is not suitable?
 - How can we generalise these approaches?
 - If we do, what are the implications of making things more complicated?
 - Can we combine methods?
 - Can we say anything about the theory of these ideas?
- What if n or p are large?
- Bayesian methods.

Further reading

- ISLR chapter 1-2.
- ESL chapter 1-2.
- PRML chapter 1.

My first regression and classification

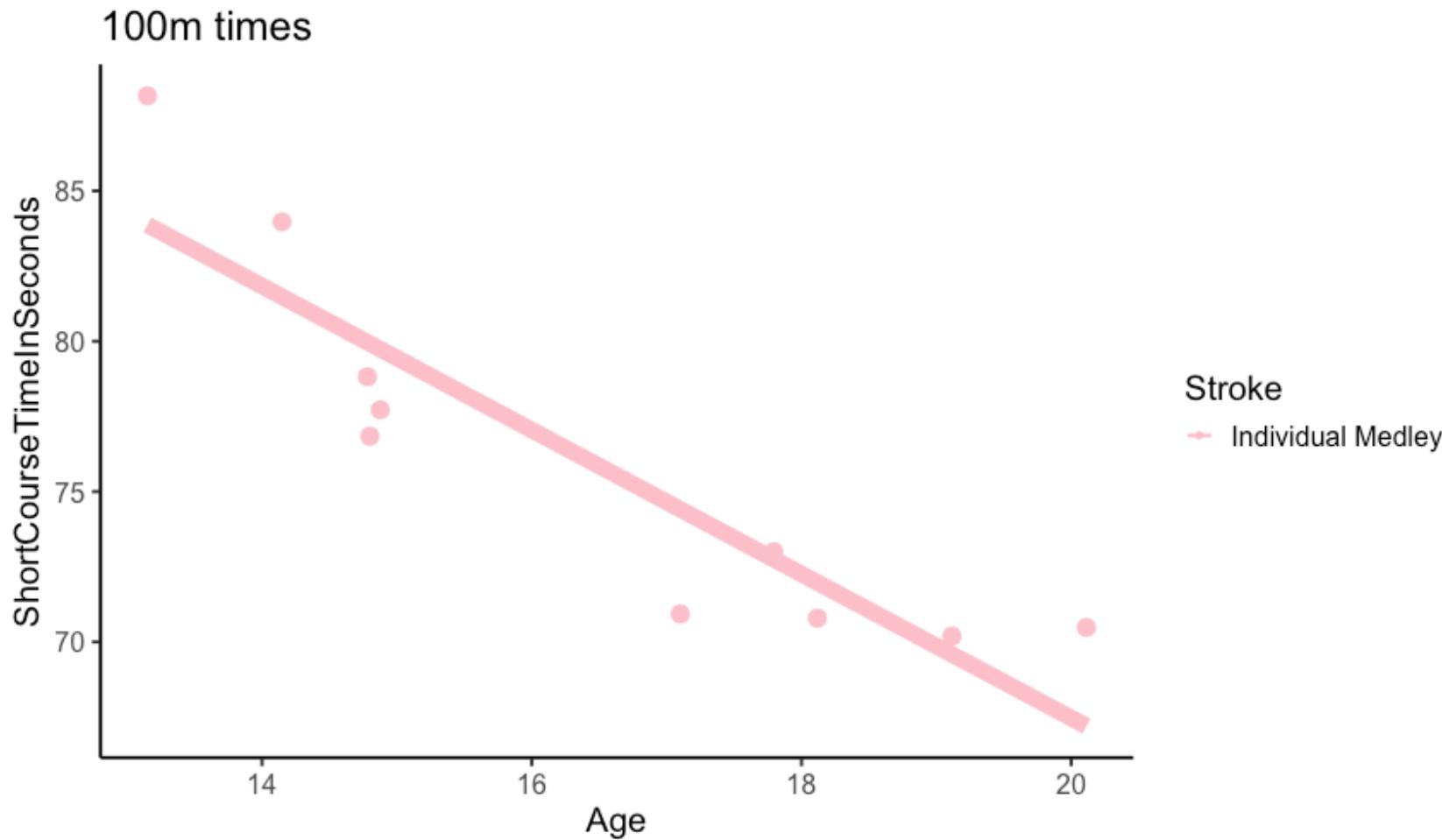
ST420 Lecture 2

Richard Everitt

Plan for today

- Go through some of the fundamentals of
 - linear regression;
 - linear classification.
- Introduce some other techniques and ideas at the end of the lecture.

Regression: simple example



How do we estimate f ?

- Choose a class of models to search through
 - For linear regression (where $p = 1$, $X = X^{(1)}$ and $X^{(1)}, Y \in \mathbb{R}$), we have the model
$$Y = \beta^0 + \beta^1 X^{(1)}.$$
 - Finding the estimate \hat{f} boils down to finding estimates $\hat{\beta}^0$ and $\hat{\beta}^1$.
- Search through this class of models to find \hat{f} .
 - To do this we need some criterion by which we judge some choices to be better than others.
 - For linear regression, the most common idea is to choose $\hat{\beta}^0$ and $\hat{\beta}^1$ using *least squares*.
 - That is, to look at the sum of the squared errors of the predictions, and to choose $\hat{\beta}^0$ and $\hat{\beta}^1$ such that this is minimised.
 - Can be done analytically in this situation, but this is not always the case.

Least squares

- Denote our observed data by $(x_1, y_1), \dots, (x_n, y_n)$
 - n observations of $(X^{(1)}, Y)$.
- We want our estimated model \hat{f} to be a close fit to the data.
 - We would like $\hat{y}_i = \hat{\beta}^0 + \hat{\beta}^1 x_i$ to be close to y_i for all i .
 - To do this, we need some measure of closeness.
- Let $e_i = \hat{y}_i - y_i$. This is known as the *i*th *residual*.
- Let the *residual sum of squares* (RSS) be

$$\text{RSS}(\hat{\beta}^0, \hat{\beta}^1) = \sum_{i=1}^n e_i^2.$$

- Fitting using least squares means choosing $\hat{\beta}^0, \hat{\beta}^1$ to minimise the RSS.

Least squares fit: univariate case

- The least squares fit is

$$\hat{\beta}^1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2},$$

$$\hat{\beta}^0 = \bar{y} - \hat{\beta}^1 \bar{x}$$

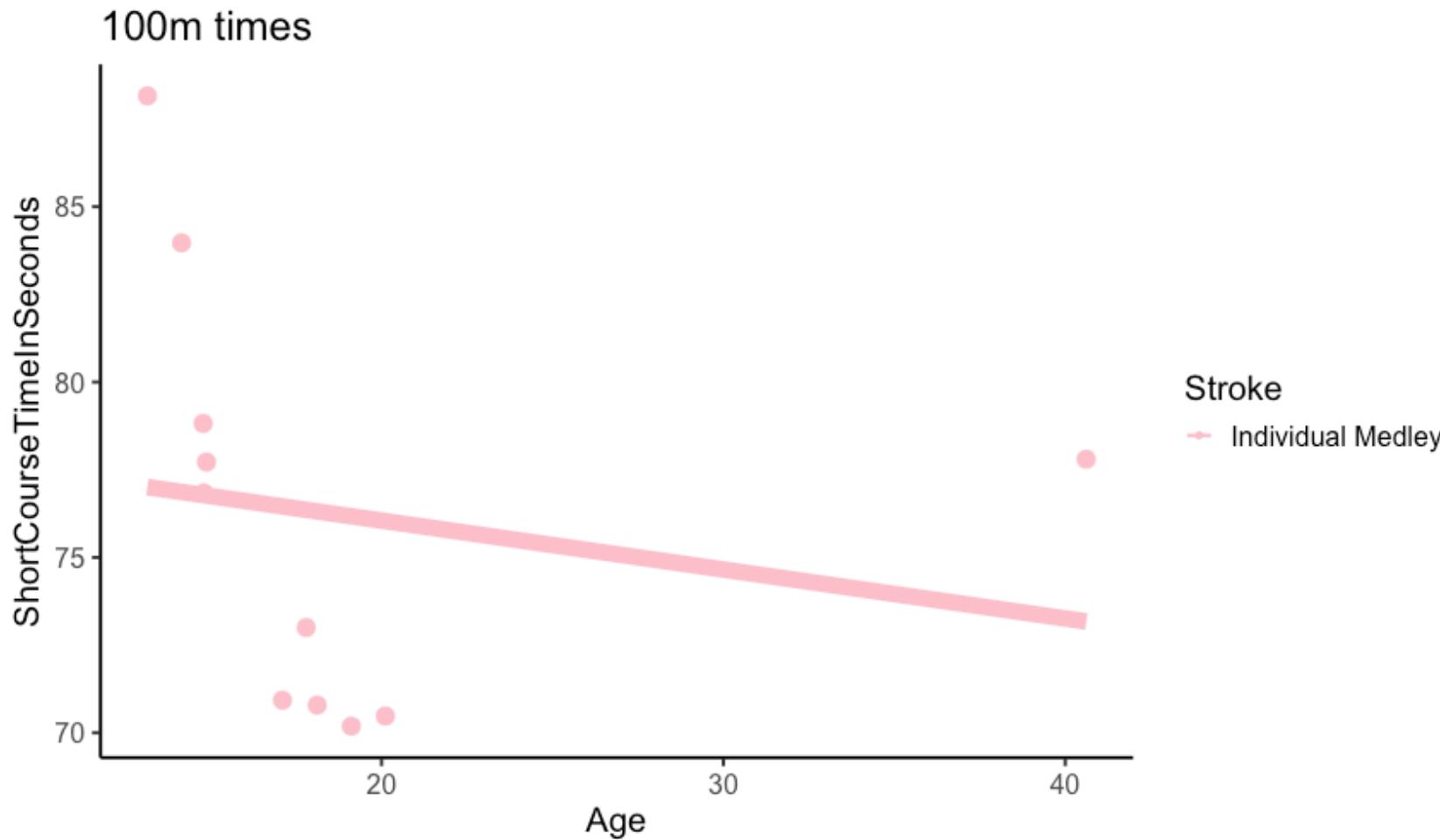
where

$$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

and

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i.$$

Regression: simple example continued



Polynomial regression

- Instead of $Y = \beta^0 + \beta^1 X^{(1)}$ use
 - $Y = \beta^0 + \beta^1 X^{(1)} + \beta^2 (X^{(1)})^2$
 - or $Y = \beta^0 + \beta^1 X + \beta^2 (X^{(1)})^2 + \beta^3 (X^{(1)})^3$
 - etc.
- Will this work?
 - Is it still possible to fit this model?
 - Will I fit the data better?
 - How many terms should I add?
 - Are there any problems with doing this?

How do I fit the model?

- Use the model

$$Y = \beta^0 + \sum_{j=1}^m (X^{(1)})^j \beta_j.$$

- The intercept term β^0 is sometimes (confusingly) called the "bias" in machine learning.
- Usually we write this model as

$$Y = X^T \beta,$$

where:

- $X^T = (1, (X^{(1)}), (X^{(1)})^2, \dots, (X^{(1)})^m);$
- $\beta = (\beta^0, \beta^1, \beta^2, \dots, \beta^m)^T.$
- Denote our observed data by $(x_1^{(1)}, y_1), \dots, (x_n^{(1)}, y_n)$
 - note slight additional precision compared to before (in using the superscript).

Least squares for polynomial regression

- Let $x_i^T = (1, (x_i^{(1)}), (x_i^{(1)})^2, \dots, (x_i^{(1)})^m)$.
- We wish to minimise

$$\text{RSS}(\beta) = \sum_{i=1}^n (y_i - x_i^T \beta)^2.$$

- Switching to matrix notation

$$\text{RSS}(\beta) = (y - \mathbf{X}\beta)^T (y - \mathbf{X}\beta)$$

where \mathbf{X} is an $n \times (m + 1)$ matrix known as the *design matrix* and $y = (y_1, \dots, y_n)^T$.

- We wish to find β where

$$\frac{d}{d\beta} \text{RSS}(\beta) = 0,$$

Least squares continued

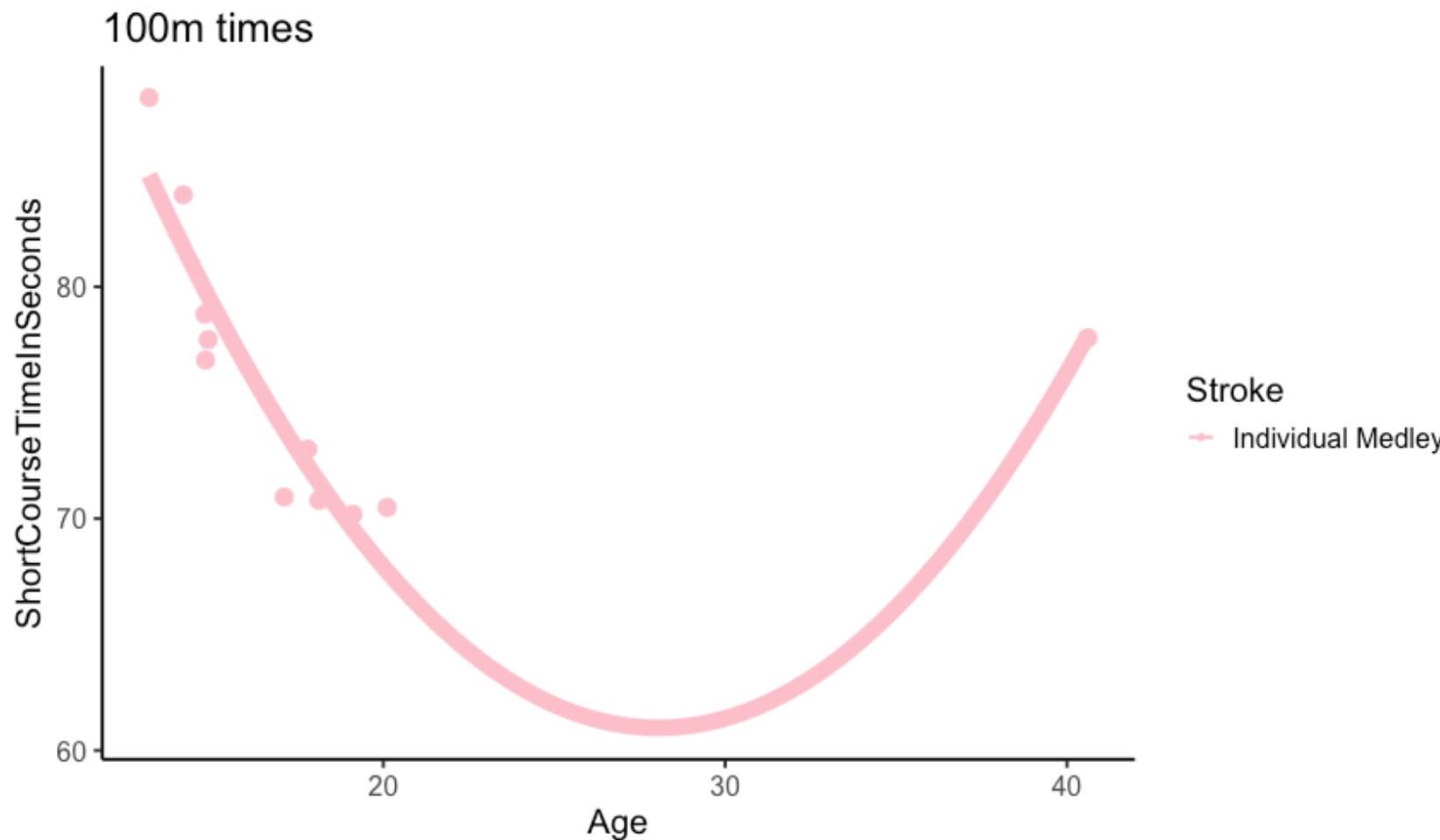
- Differentiating with respect to β we obtain

$$-2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) = 0.$$

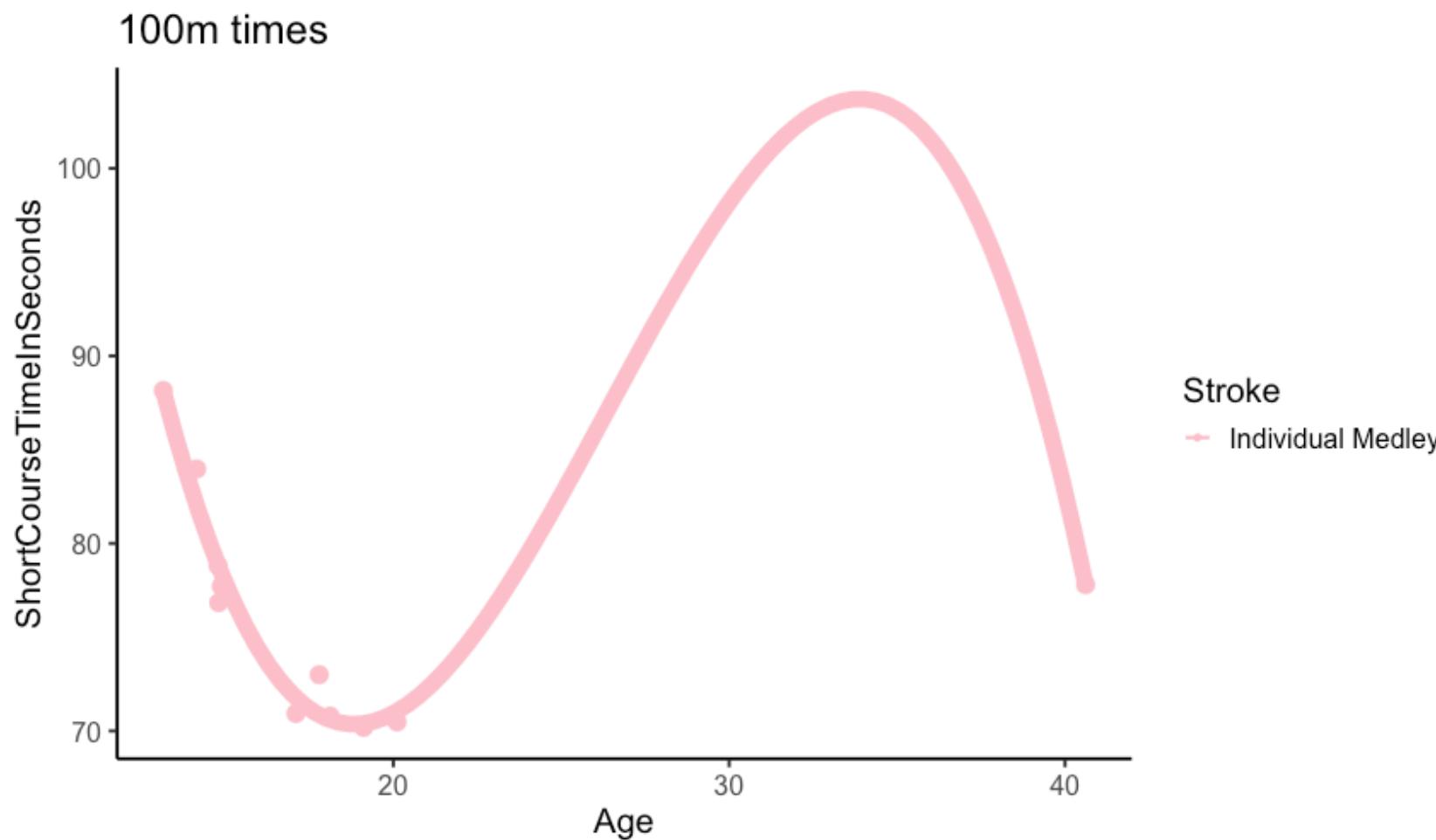
- If $\mathbf{X}^T\mathbf{X}$ is non-singular, then the unique solution is given by

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}.$$

Polynomial regression with quadratic term



Polynomial regression with quadratic and cubic terms



Extension: change of basis

- We are using the vector $X^T = (1, X^{(1)}, (X^{(1)})^2, \dots, (X^{(1)})^m)$ as a basis for our estimate \hat{f} .
- Note that we can fit the model in the same way for any basis

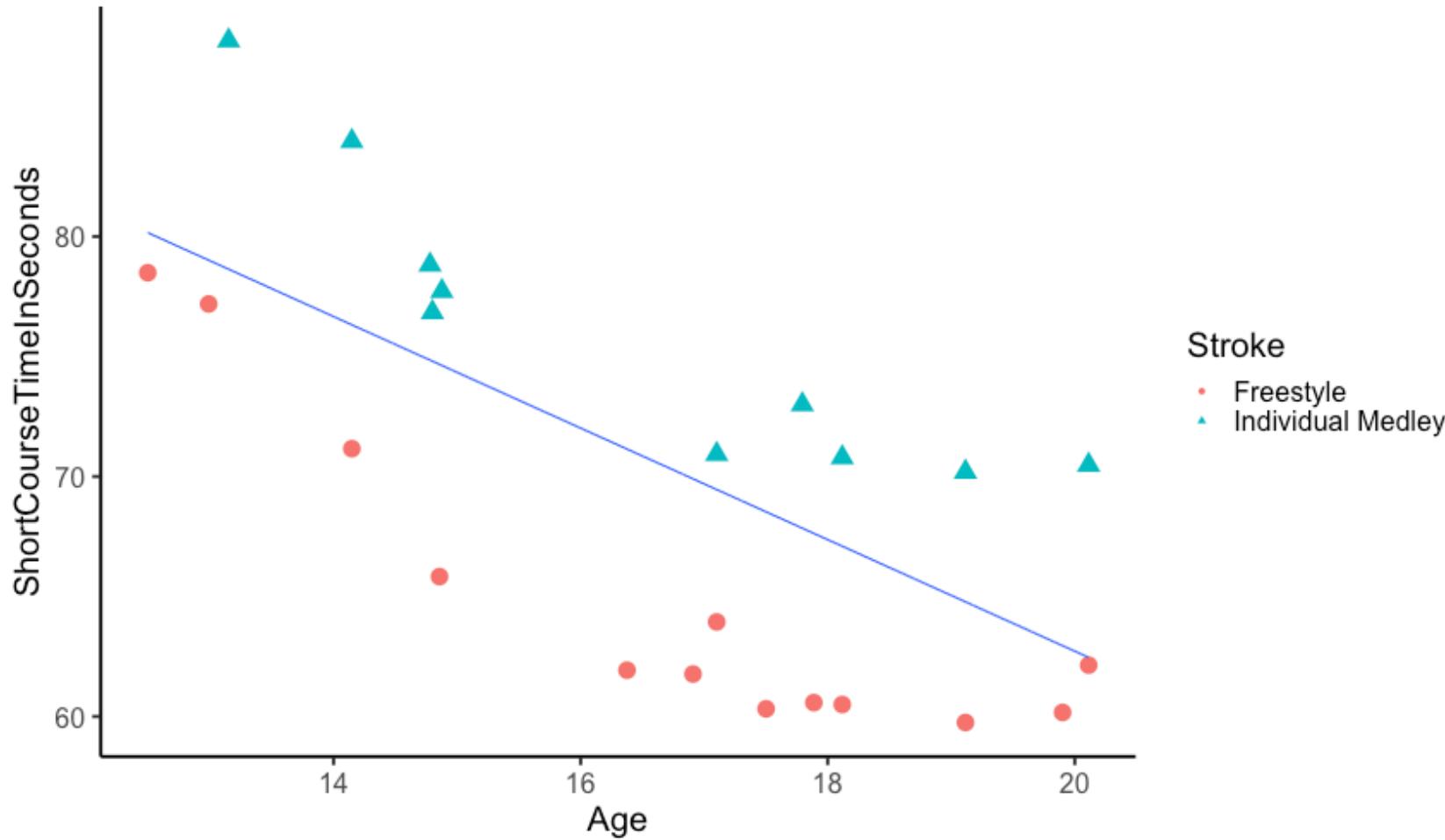
$$X^T = (1, (\phi^1(X^{(1)})), (\phi^2(X^{(1)})), \dots, (\phi^m(X^{(1)}))).$$

- Common choices of basis:
 - nonlinear functions to transform the predictors $\phi^j(x) = \log(x)$, \sqrt{x} , etc;
 - piecewise constant $\phi^j(x) = I(L_m \leq x \leq U_m)$;
 - "Gaussian" $\phi^j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2s^2}\right)$;
 - sigmoidal $\phi^j(x) = \sigma\left(\frac{(x-\mu_j)}{s}\right)$ where $\sigma(a) = \left(\frac{1}{1+\exp(-a)}\right)$;
 - Fourier;
 - wavelets.

Extensions: multiple regression

- We can do multiple regression where $X^T = (1, X^{(1)}, X^{(2)}, \dots, X^{(p)})$.
- In general we can use any function of our predictors
 - e.g. $X^T = (1, X^{(1)}, X^{(2)}, X^{(1)}X^{(2)})$ uses the *interaction* term $X^{(1)}X^{(2)}$.
- Notation: we will use p for the number of predictors, and m for the number of basis functions.

Linear classification



Perceptron

- Separate classes with a linear combination of predictors: idea dates back to Rosenblatt (1958).
- Let Y be a discrete variable that can take values in $\{+1, -1\}$
 - classification with 2 classes.
- Let $Y = f(X)$ where

$$f(X) = \begin{cases} +1 & \text{if } X^T \beta \geq 0 \\ -1 & \text{if } X^T \beta < 0 \end{cases}.$$

- When X^T contains the intercept term (as previously), and is of length $m + 1$, $X^T \beta$ is the equation of a hyperplane in m -dimensions
 - special case: when $m = p = 2$, as in the figure in the previous slide, $X^T \beta$ is a line, where $X^T = (1, X^{(1)}, X^{(2)})$.
- We will discuss fitting such a model later in the module.

Extension: change of basis

- Just as for regression, we can use a different basis. i.e. use

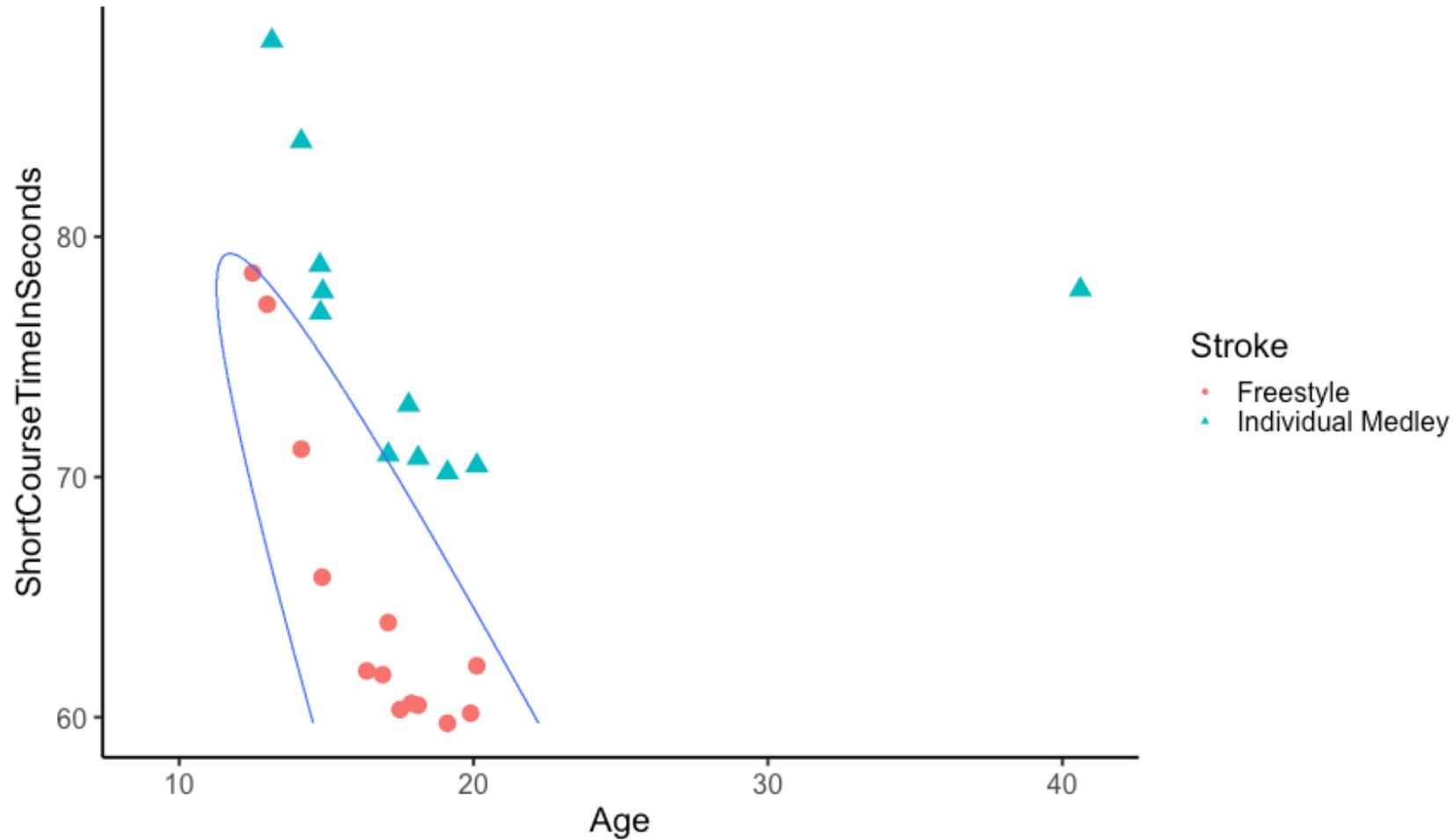
$$X^T = (1, (\phi^1(X^{(1)})), (\phi^2(X^{(1)})), \dots, (\phi^m(X^{(1)})))$$

then once more use

$$f(X) = \begin{cases} +1 & \text{if } X^T \beta > 0 \\ -1 & \text{if } X^T \beta \leq 0 \end{cases}.$$

- In our 2-d example, we use $X^T = (1, X^{(1)}, X^{(2)}, X^{(1)}X^{(2)}, (X^{(1)})^2, (X^{(2)})^2)$
 - linear functions in the augmented space map down to quadratic functions in the original space
 - hence we obtain a quadratic decision boundary.

Classification using quadratic curve



Review

- These simple techniques can take us quite a long way.
- In many situations, more sophisticated methods do not improve performance.
- These are "parametric" choices for f :
 - define a model;
 - estimate a vector of $m + 1$ parameters
- What about non-parametric approaches?
 - Do not make explicit assumptions about the functional form of f .
 - Try to fit the data points without f being too wiggly.

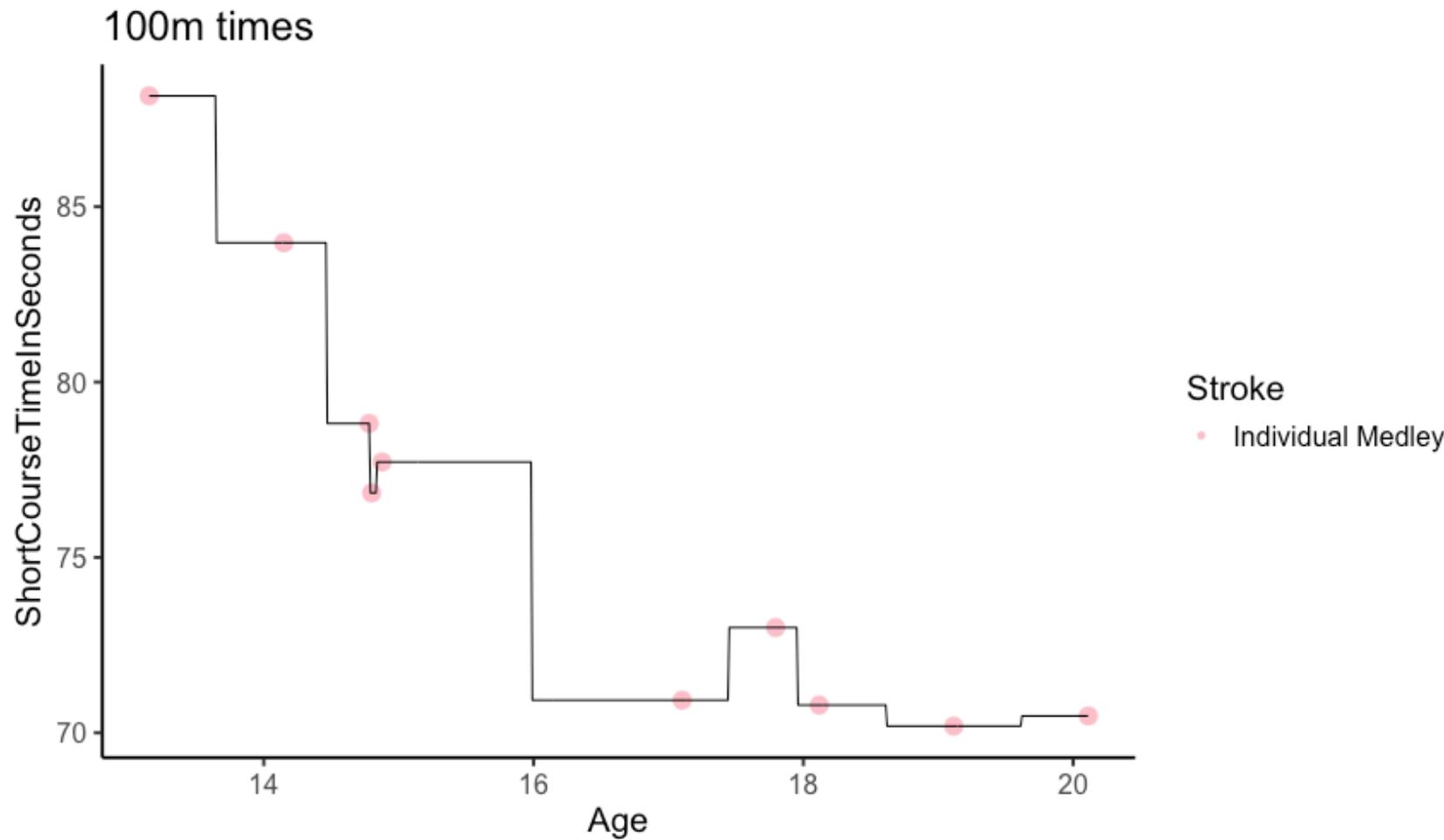
k -nearest neighbour regression

- Idea: predicted value is the average of the y values corresponding to nearby x points.
- The estimated k -nearest neighbour regression is

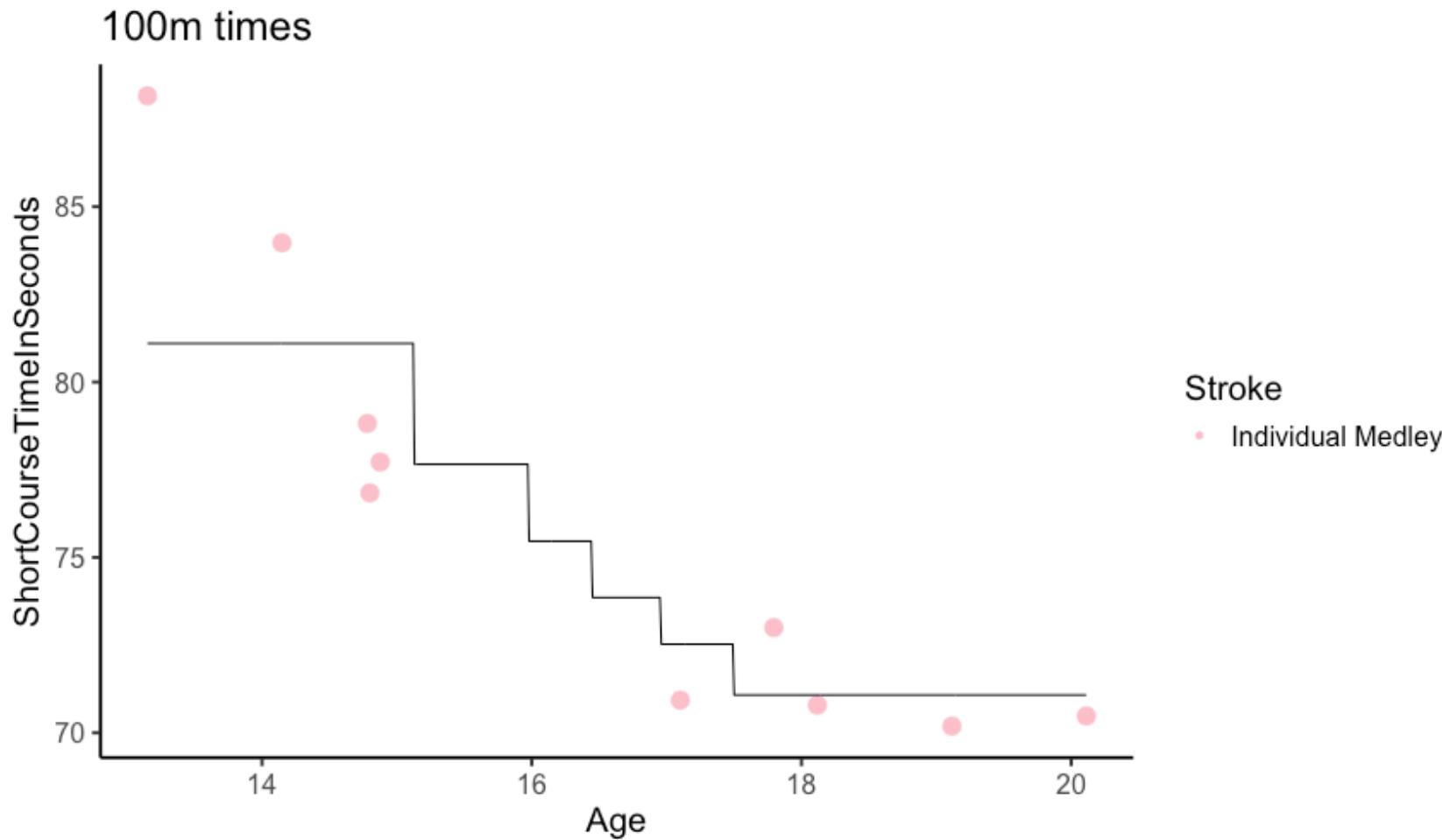
$$\hat{Y} = \frac{1}{k} \sum_{\{l|x_{(l)} \in N_k(x)\}} y_{(l)}$$

where $N_k(x)$ is the neighbourhood of x defined by the k closest (according to some metric) points in the training data.

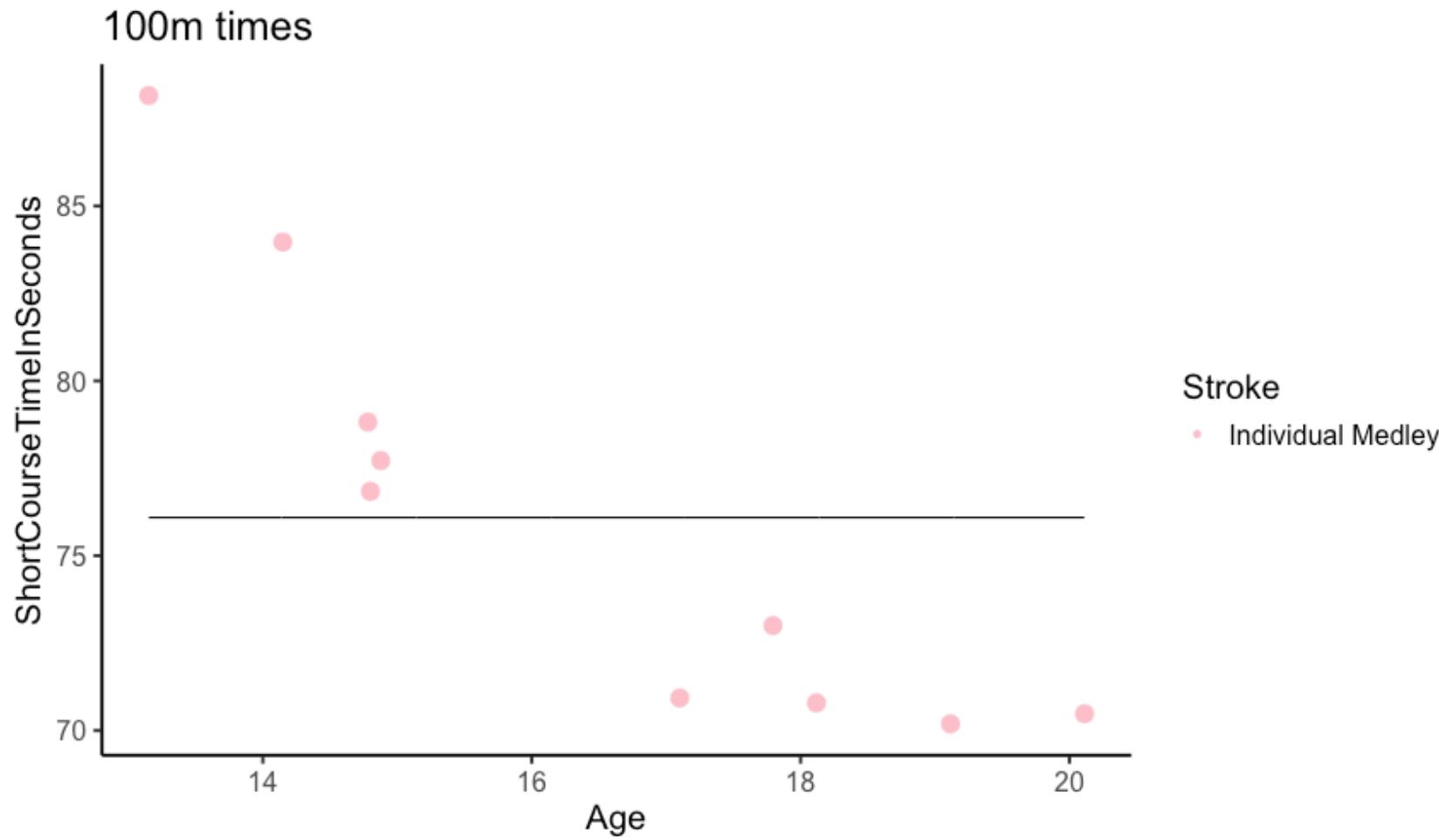
1-nearest neighbour regression



5-nearest neighbour regression



10-nearest neighbour regression



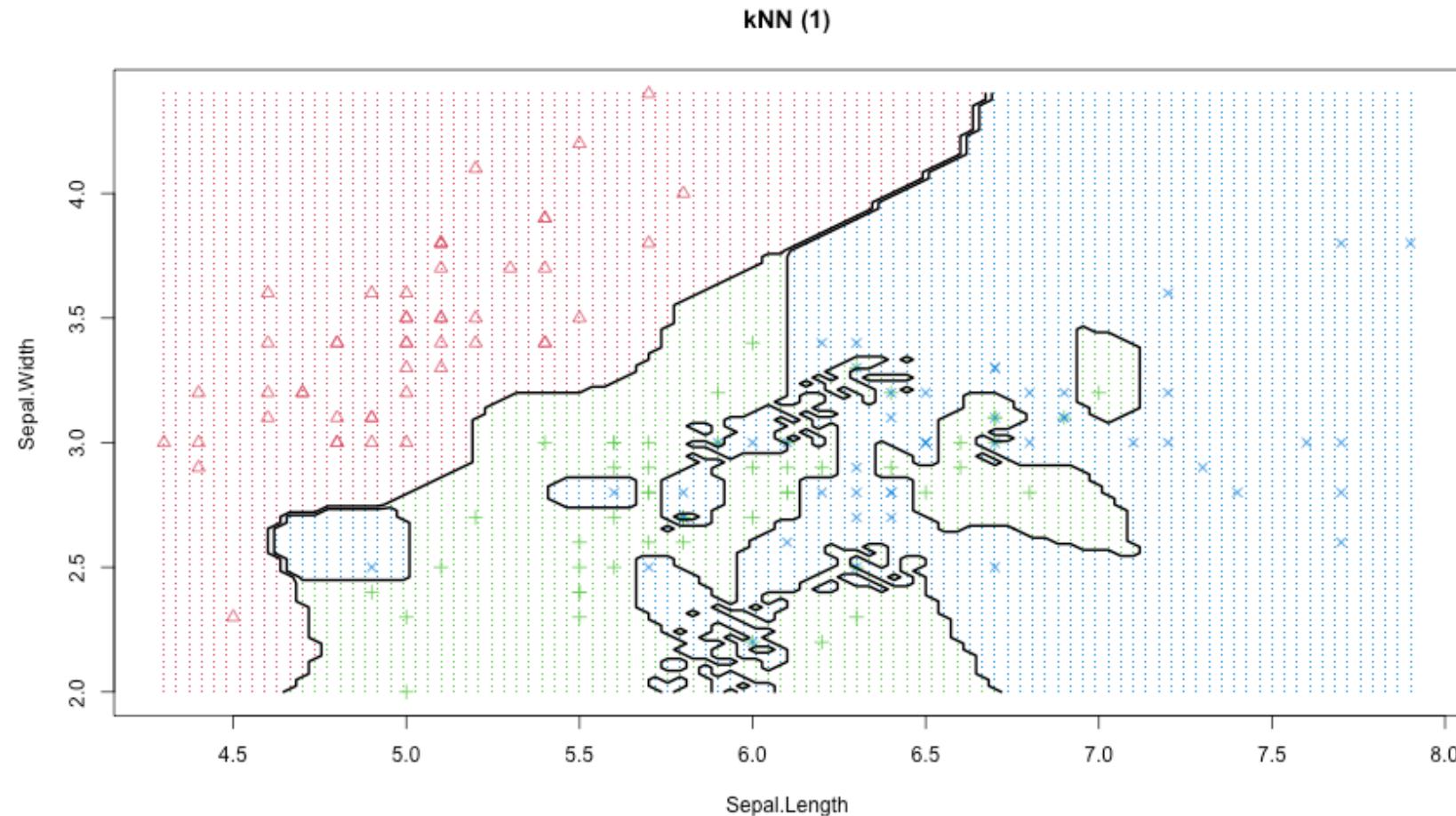
k -nearest neighbour classification

- Idea: predicted value most common class of the y values corresponding to nearby x points.
- The estimated k -nearest neighbour classification is

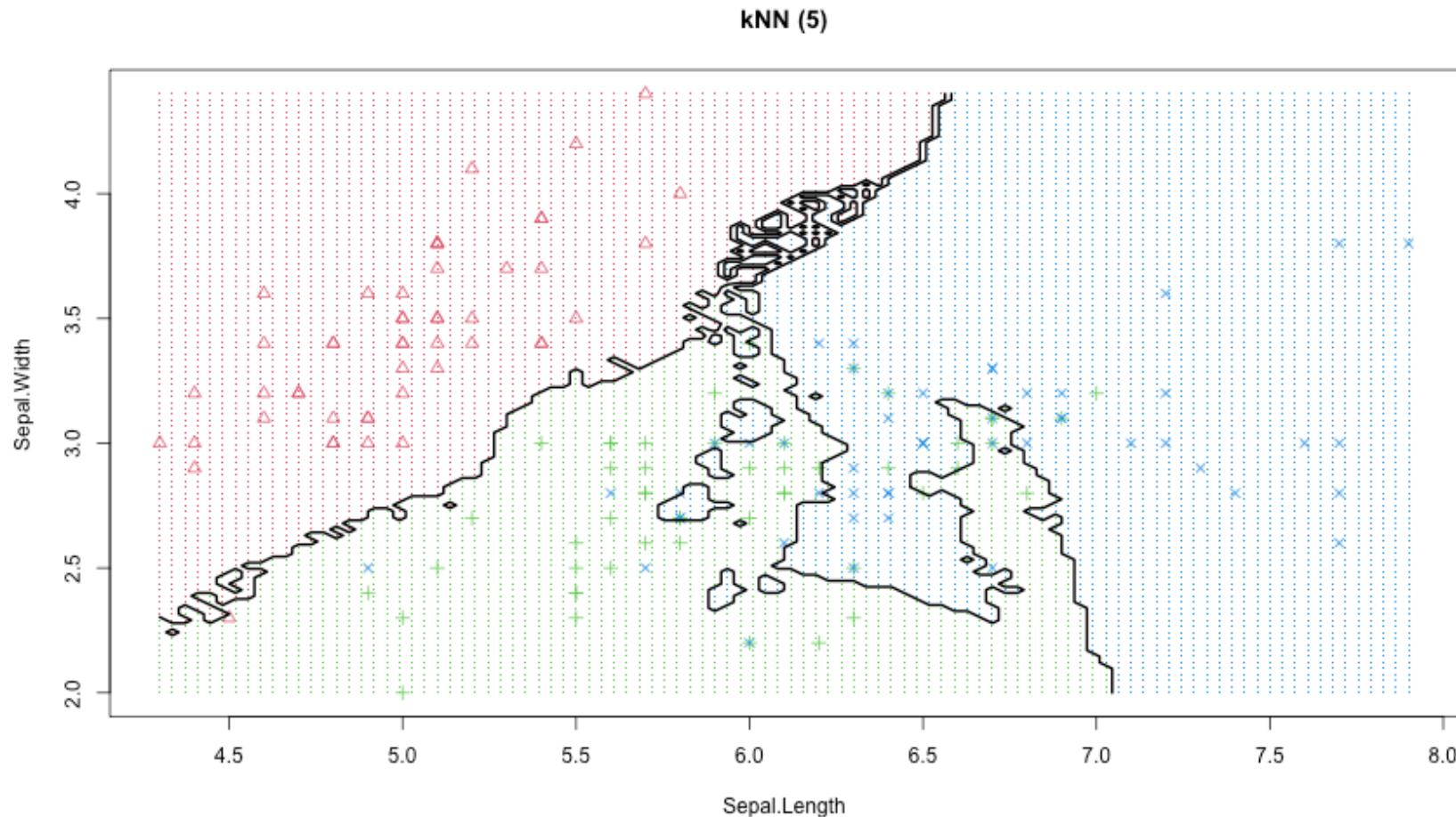
$$\hat{Y} = \text{mode}_{\{l | x_{(l)} \in N_k(x)\}} \{y_{(l)}\}$$

where $N_k(x)$ is the neighbourhood of x defined by the k closest (according to some metric) points in the training data.

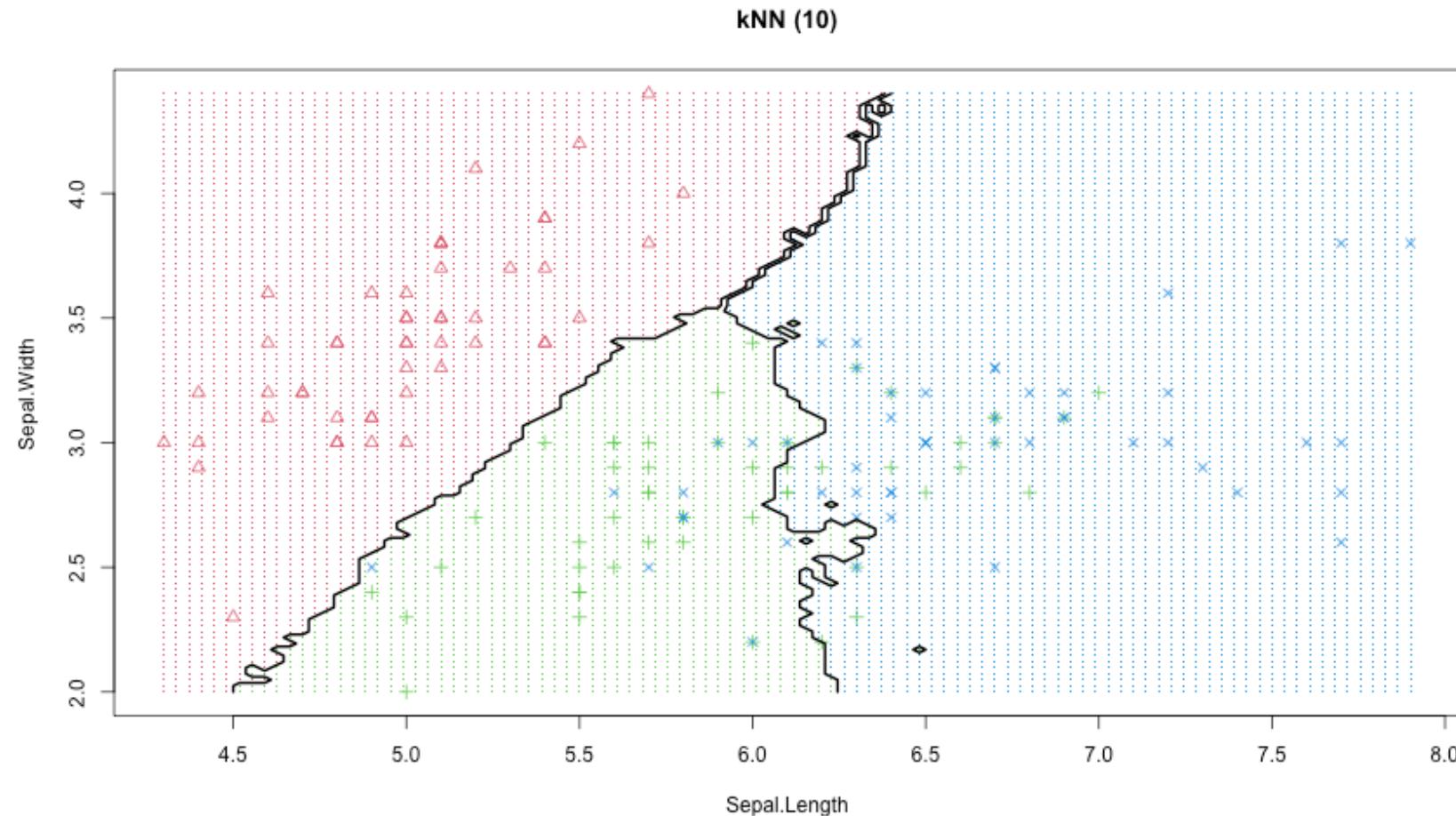
1-nearest neighbour classifier



5-nearest neighbour classifier



10-nearest neighbour classifier



Features of nearest neighbour

- Advantages:
 - simple and easy to implement;
 - does not make any assumptions about the structure of data;
 - theoretical guarantees (see later)...
- Disadvantages:
 - how to choose k ?
 - how to choose an appropriate metric?
 - to make predictions we need to use the whole training set;
 - computationally expensive to make a prediction, $O(np)$.

Comparing linear regression with nearest neighbour

- Imagine we fit the regression using different subsets of the data
 - this is a thought experiment to try to understand the dependence of the regression fit on the available training data.
- Linear regression:
 - regression line will not change too much over the subsets;
 - if the linear assumption was wrong, we will always have a bad fit.
- Nearest neighbour:
 - regression line will depend heavily on subsets;
 - only assumption made is that f is locally constant, so is more robust to different situations.

Final thoughts

- Which approach is more interpretable?
- Which approach will work better when:
 - n gets bigger?
 - p gets bigger?
- Next few lectures:
 - formalise an approach to fitting models;
 - give a flavour of statistical learning theory.

Further reading

- ISLR chapter 2.
- ESL chapter 2.
- PRML chapter 1.

Loss, risk and estimation

ST420 Lecture 3

Richard Everitt

Supervised learning

- Recall that our aim is to find an estimate $\hat{f} : \mathcal{X} \rightarrow \mathcal{Y}$
 - given predictors $X \in \mathcal{X}$ we wish to make accurate predictions of $Y \in \mathcal{Y}$ by using $\hat{f}(X)$
 - we find an estimate \hat{f} from training data $\mathcal{T} = \{(X_i, Y_i)\}_{i=1}^n$.
- In this lecture, we will introduce a means by which we can measure the accuracy of the prediction
 - we use a *loss function*
 - building on this idea allows us to formalise some of the ideas we have introduced.

Loss and risk

- Let X and Y be random variables (or vectors) that respectively have outcomes in the spaces \mathcal{X} and \mathcal{Y} .
- Let $f \in \mathcal{F}, f : \mathcal{X} \rightarrow \mathcal{Y}$ be a function from \mathcal{X} to \mathcal{Y} .
- A loss function is any function

$$L : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$$

- we will use it to measure the difference between values of Y and predictions $f(X)$.
- The *risk* $R : \mathcal{F} \rightarrow \mathbb{R}_{\geq 0}$ associated with loss function L and function $f : \mathcal{X} \rightarrow \mathcal{Y}$ is given by

$$R(f) = \mathbb{E}_{X,Y} [L(Y, f(X))].$$

Empirical risk

- Suppose we observe training data $\mathcal{T} = \{(X_i, Y_i)\}_{i=1}^n$, a random sample from the joint distribution of the variables X and Y .
- Then the *empirical risk* \hat{R} is an estimate of R based on the sample \mathcal{T}

$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^n L(Y_i, f(X_i)).$$

- R is a deterministic function, but \hat{R} is random since it depends on a random sample.
- The Law of Large Numbers tells us that, for any f , as $n \rightarrow \infty$,

$$\hat{R}(f) \rightarrow R(f).$$

Empirical risk minimisation (ERM)

- Loss measures how close predictions of a function f are to outcomes.
- Risk gives the average of the loss over the joint distribution of X and Y .
- So, risk gives us some measure of how good a predictor a function f is
 - called *expected prediction error* (EPE) in ESL.
- Suppose that we consider some class of functions $\bar{\mathcal{F}}$ to restrict our attention to
 - then we might want to find the f that minimises the risk

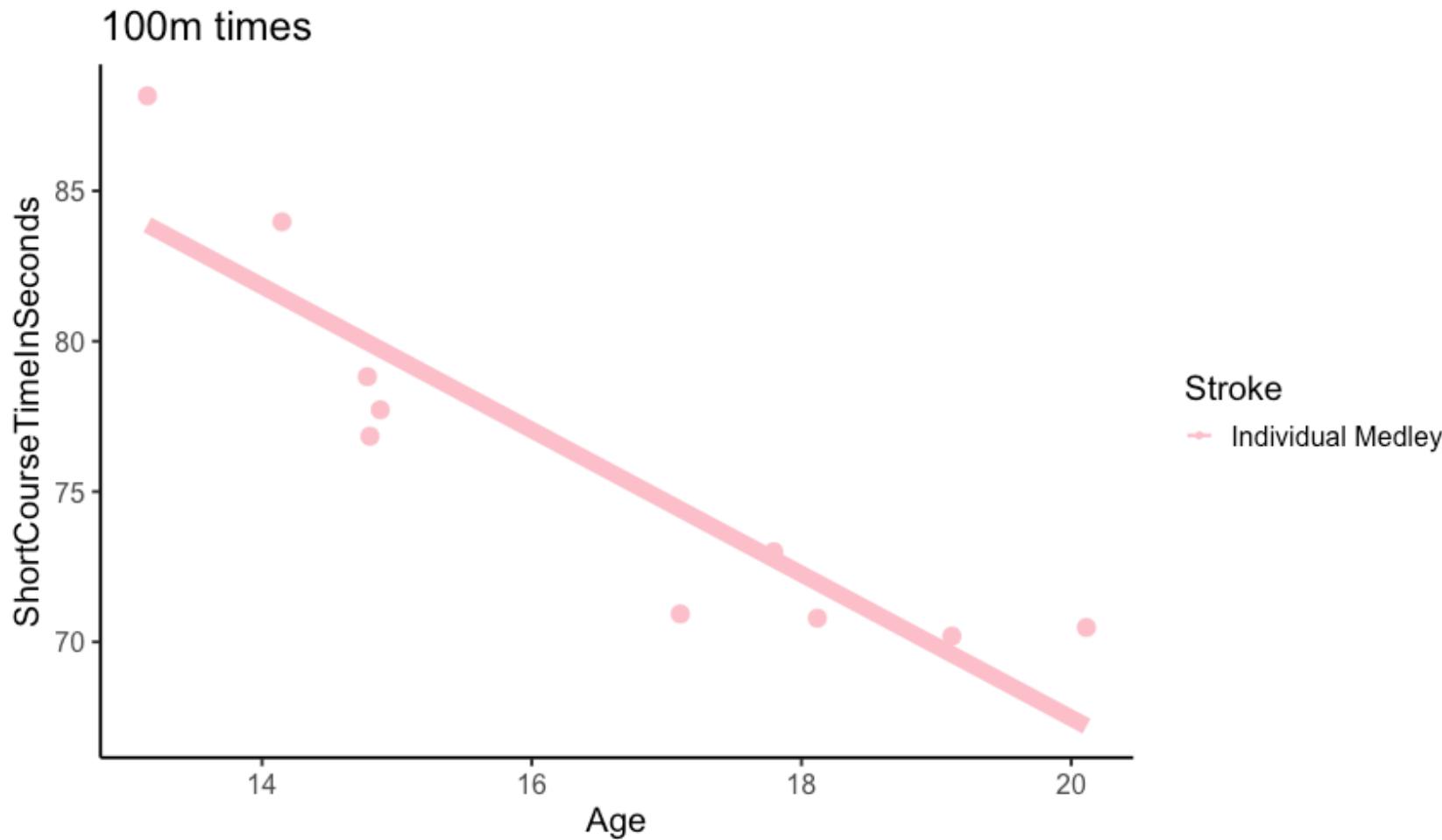
$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} R(f)$$

- in practice $R(f)$ is not available, so instead we minimise the empirical risk

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \hat{R}(f)$$

- using $\hat{R}(f)$ instead of $R(f)$ has important implications that we will discuss later.

Regression: simple example



ERM for least squares linear regression

- **Choice of functions.** Let $\bar{\mathcal{F}}$ be functions of the form $f(X) = X^T \beta$ (as outlined in lecture 1).
- **Choice of loss function.** Let

$$L(Y, f(X)) = (Y - f(X))^2.$$

- **ERM.** We wish to find β (which fully determine f) that satisfy

$$\arg \min_{\beta \in \mathbb{R}^{p+1}} \frac{1}{n} \sum_{i=1}^n (Y_i - X_i^T \beta)^2.$$

- Note that this is the same as minimising the residual sum of squares (the term $1/n$ is the only difference from the expression in the previous lecture).
- Least squares is a special case of ERM.

Examples of loss functions for regression

- Squared (L_2) error loss

$$L(Y, f(X)) = (Y - f(X))^2.$$

- Absolute (L_1) error loss

$$L(Y, f(X)) = |Y - f(X)|.$$

- L_p error loss for any $p > 0$

$$L(Y, f(X)) = |Y - f(X)|^p.$$

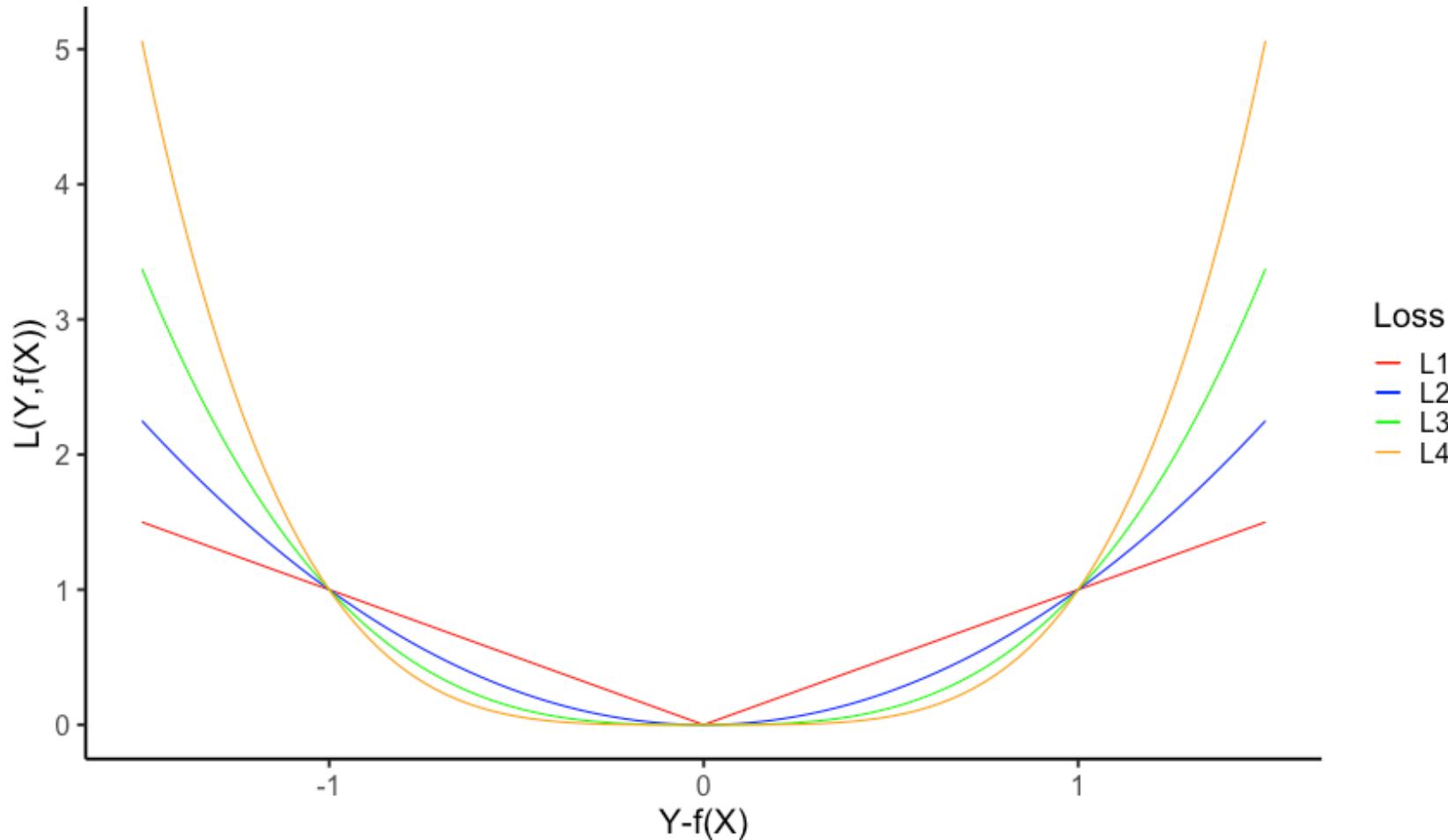
- Log-likelihood loss

$$L(Y, f(X)) = -2 \log l(Y \mid X, \theta)$$

where l is some distribution of Y given X and θ .

- the same as squared loss when doing regression under the assumption of Gaussian noise.

Sketch of loss functions for regression



Loss functions for classification: setup

- For simplicity here, we restrict ourselves to the case of two classes, where Y can take values in $\{-1, 1\}$.
- Recall the perceptron: $Y = f(X)$ where

$$f(X) = \begin{cases} +1 & \text{if } X^T \beta \geq 0 \\ -1 & \text{if } X^T \beta < 0 \end{cases}.$$

- Here let $g(X) \in \mathbb{R}$ to be the function that is mapped to the discrete value $f(X)$, e.g. the $g(X) = X^T \beta$ in the perceptron case.
- f depends on g , thus we still use the notation $L(Y, f(X))$
 - we are omitting this dependence in our notation (just as we are when we omit the dependence of f on β when using linear regression).

Examples of loss functions for classification

- Zero-one loss

$$L(Y, f(X)) = I(Yg(X) \leq 0).$$

- Hinge loss

$$L(Y, f(X)) = \max\{1 - Yg(X), 0\}.$$

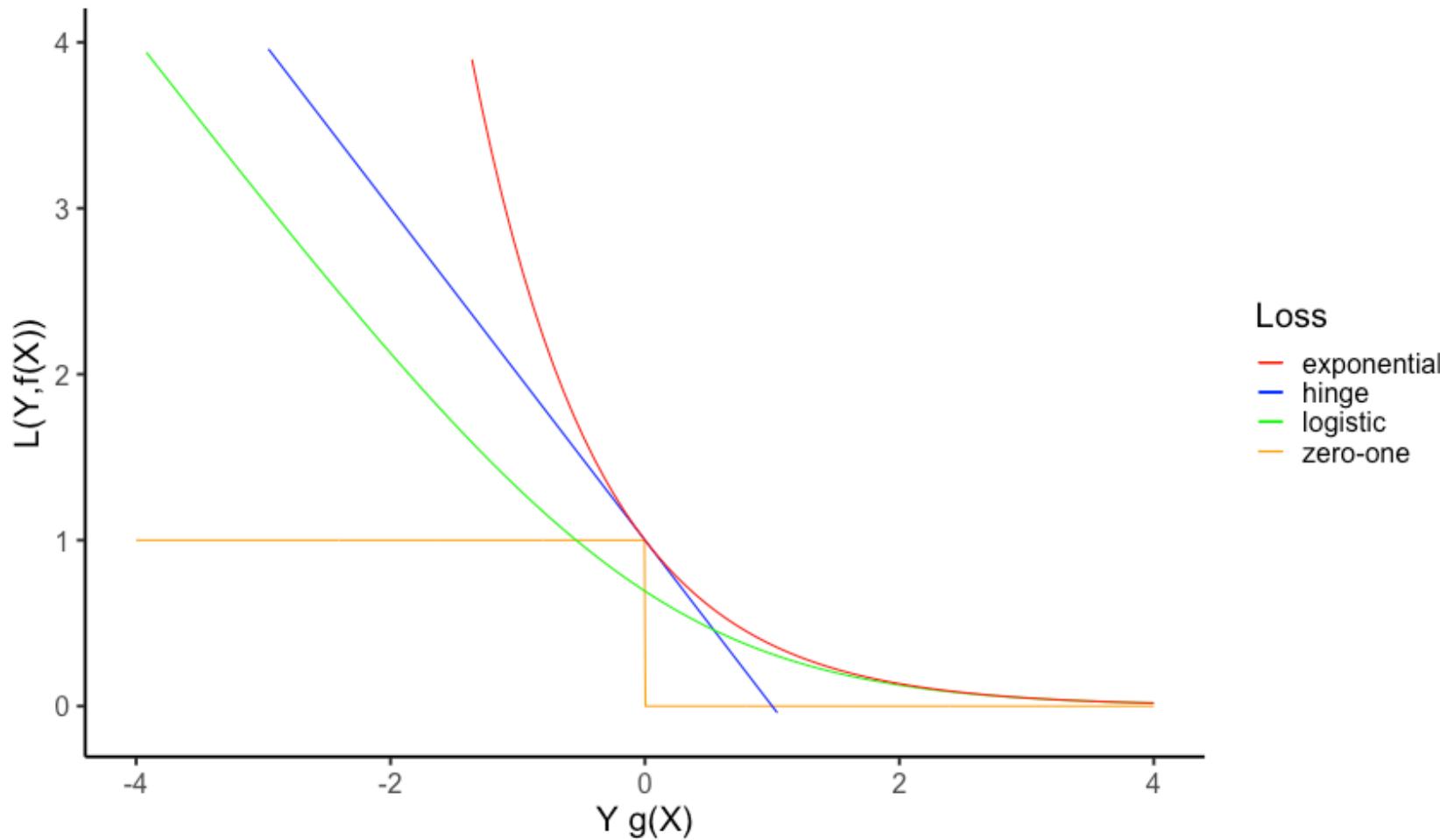
- Logistic loss $p > 0$

$$L(Y, f(X)) = \log(1 + \exp(-Yg(X))).$$

- Exponential loss

$$L(Y, f(X)) = \exp(-Yg(X)).$$

Sketch of loss functions for classification



Convex loss functions

- Many of the loss functions are convex, which turns out to be useful!
- We now recap some important definitions and properties.
- **Convex set.** A subset X of a vector space is convex if for all $x, y \in X$ and $\lambda \in [0, 1]$,
$$(1 - \lambda)x + \lambda y \in X.$$
- **Convex and strictly convex functions.** Let X be a convex subset of a vector space and let $f : X \rightarrow \mathbb{R}$ be a function. Then:

- f is convex if $x, y \in X$ and $\lambda \in [0, 1]$,

$$f((1 - \lambda)x + \lambda y) \leq (1 - \lambda)f(x) + \lambda f(y).$$

- f is strictly convex if $x \neq y \in X$ and $\lambda \in (0, 1)$,

$$f((1 - \lambda)x + \lambda y) < (1 - \lambda)f(x) + \lambda f(y).$$

Minima

- **Global minimum.** A global minimum of a function $f : X \rightarrow \mathbb{R}$ is a point $x_0 \in X$ such that $\forall x \in X f(x_0) \leq f(x)$.
- **Local minimum.** Let X be a metric space. A function $f : X \rightarrow \mathbb{R}$ has a local minimum x_0 if there exists some $\epsilon > 0$ such that for all x within a distance of ϵ of x_0 , $f(x_0) \leq f(x)$.
- **Strict minima.** Global and local minima are said to be strict if the inequalities $f(x_0) \leq f(x)$ in the previous definitions are changed to $f(x_0) < f(x)$.
- A point is a strict global minimum if and only if it is the unique global minimum.

Properties of convex functions

- A local minimum of a convex function is also a global minimum.
- A global minimum of a strictly convex function is a unique global minimum.
- Recall that a twice-differentiable function is convex on a convex domain if and only if:
 - (function of one variable) its second derivative is everywhere ≥ 0 ;
 - (function of > 1 variable) its second derivative is everywhere positive semi-definite.
- If a function's second derivative is everywhere > 0 (or everywhere positive definite for multiple variables), then function is strictly convex
 - but the converse does not hold.

Convex loss functions

- Notice that all of the functions we have given (with the exception of L_p where $0 < p < 1$, and zero-one loss) are strictly convex
 - as a function of $Y - f(X)$ for regression
 - as a function of $Yg(X)$ for classification.
- There are useful implications when we use regression functions of the form $f(X) = X^T \beta$ or $g(X) = X^T \beta$ in classification (for $\beta \in \mathbb{R}^m$).
- Consider the classification case, and let ψ be a convex differentiable function and let $L(Y, f(X)) = \psi(Yg(X))$.
- We will use that a matrix M is positive semi-definite if $z^T M z \geq 0$ for every non-zero column vector z
 - or positive definite if $z^T M z > 0$.

Convex loss functions and global minima

- In this case the empirical risk is

$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^n \psi(y_i x_i^T \beta).$$

- Its first differential is

$$\frac{\partial \hat{R}(f)}{\partial \beta} = \frac{1}{n} \sum_{i=1}^n \psi'(y_i x_i^T \beta) y_i x_i.$$

- Its second differential is

$$\frac{\partial^2 \hat{R}(f)}{\partial \beta \partial \beta^T} = \frac{1}{n} \sum_{i=1}^n \psi''(y_i x_i^T \beta) y_i^2 x_i x_i^T.$$

Convex loss functions

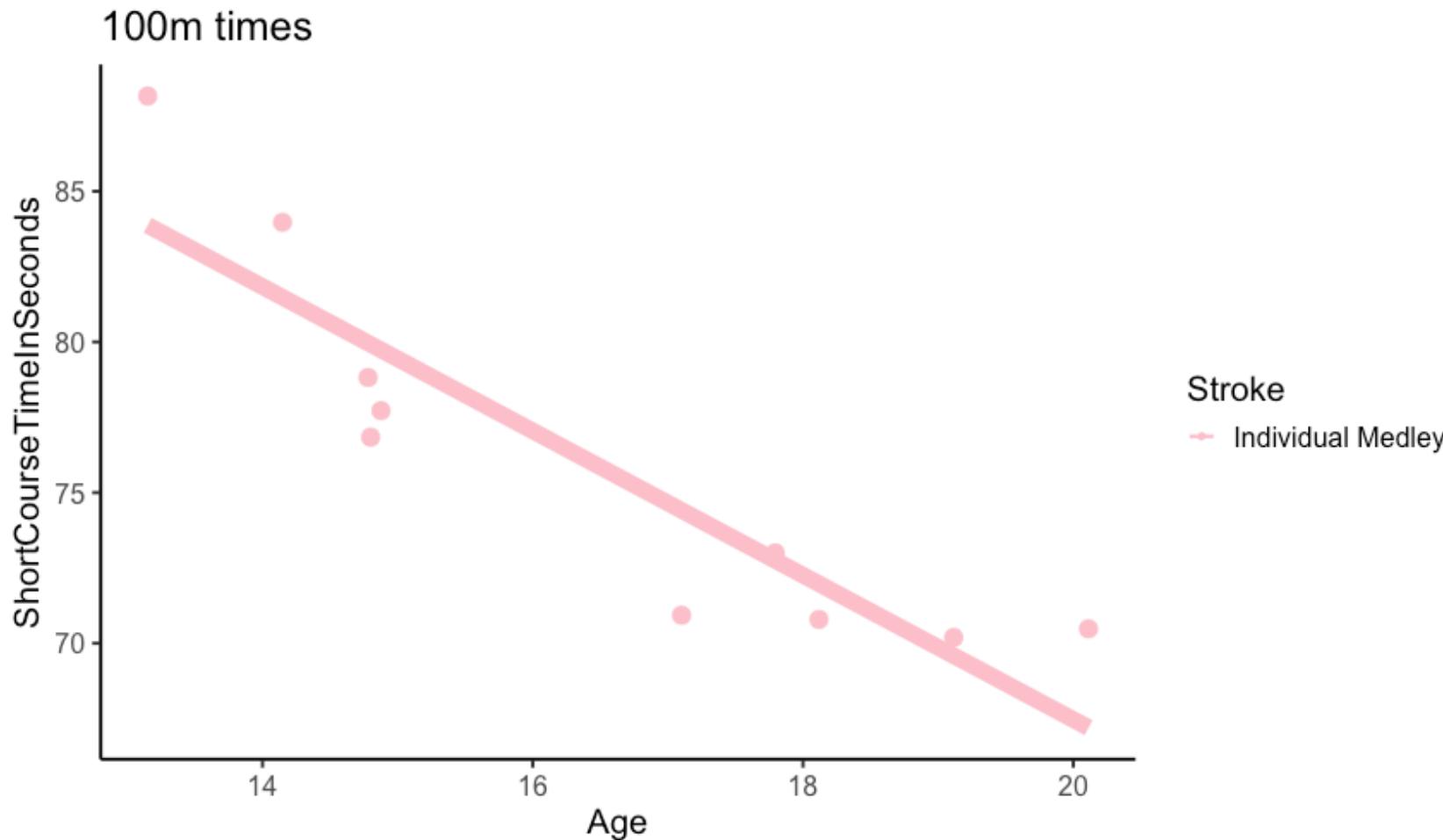
- Now

$$z^T \frac{\partial^2 \hat{R}(f)}{\partial \beta \partial \beta^T} z = \frac{1}{n} \sum_{i=1}^n \psi''(y_i x_i^T \beta) (y_i z^T x_i)^2,$$

which is ≥ 0 since ψ is convex (and hence $\psi''(x) \geq 0$).

- Therefore the second differential is positive semi-definite
 - thus \hat{R} is a convex function of β
 - therefore a local minimum of \hat{R} is a **global minimum**.
- Often it will be the case that finding this minimum cannot be done analytically
 - we often use a gradient-based search algorithm in practice.

Where does the prediction error come from?



Decomposing the prediction error

- Let's focus on regression with squared error loss.
- What is the expected error in a prediction in Y given X ?

$$\begin{aligned}\mathbb{E}_Y[(Y - f(X))^2 | X] &= \mathbb{E}_Y[Y^2 | X] - 2f(X)\mathbb{E}_Y[Y | X] + f(X)^2 \\&= \mathbb{E}_Y[Y^2 | X] - (\mathbb{E}_Y[Y | X])^2 + (\mathbb{E}_Y[Y | X])^2 - 2f(X)\mathbb{E}_Y[Y | X] + f(X)^2 \\&= \underbrace{\mathbb{V}_Y[Y | X]}_{\text{irreducible}} + (\mathbb{E}_Y[Y | X] - f(X))^2\end{aligned}$$

- The *irreducible error* is due to the randomness in Y that can't be captured by a function
 - we can never predict perfectly, unless there is a deterministic relationship between X and Y .

Optimal f and optimal risk

- We have

$$R(f) = \mathbb{E}_X \left[\mathbb{E}_Y \left[(Y - f(X))^2 \mid X \right] \right]$$

to see this (exercise if you need to), start with the expression for the risk, rewrite the expectation as integrals, factorise the joint distribution of X and Y as the product of the conditional of $Y \mid X$ and the marginal on X , and rearrange.

- So

$$R(f) = \mathbb{E}_X \left[\mathbb{V}_Y [Y \mid X] \right] + \mathbb{E}_X \left[(\mathbb{E}_Y [Y \mid X] - f(X))^2 \right].$$

- Note that both terms here are positive
 - makes it easy to see the choice of f that gives the optimal risk.

Optimal f and optimal risk

- We obtain the optimal risk when

$$f(X) = \mathbb{E}_Y[Y | X],$$

and the optimal risk is

$$\mathbb{E}_X[\mathbb{V}_Y[Y | X]].$$

- Remarks:
 - the optimal risk is usually non-zero;
 - this choice of f is known as the *regression function*;
 - we do not know the true conditional distribution for $Y | X$, so we cannot obtain $\mathbb{E}_Y[Y | X]$ analytically.

Revisiting k -nearest neighbour

- Recall

$$\hat{Y} = \hat{f}(x) = \frac{1}{k} \sum_{\{l | X_{(l)} \in N_k(x)\}} Y_{(l)}$$

where $N_k(x)$ is the neighbourhood of x defined by the k closest (according to some metric) points in the training data.

- This method is trying to directly approximate $\mathbb{E}_Y[Y | X = x]$. The approximations are:
 - the approximation is an empirical average over the training data;
 - instead of conditioning on a point x , we are conditioning on a neighbourhood close to the point
 - under mild regularity conditions on the joint distribution of X and Y , we have that for any x , $\hat{f}(x) \rightarrow \mathbb{E}_Y[Y | X = x]$ as $n, k \rightarrow \infty$ with $k/n \rightarrow 0$.
- Will this always be the best method?

Optimal f and optimal risk: classification case

- Now let Y take values in $\{1, \dots, C\}$.
- Again consider the conditional

$$\mathbb{E}_Y [L(Y, f(X)) \mid X] = \sum_{c=1}^C L(Y = c, f(X)) P(Y = c \mid X).$$

- Use zero-one loss, where $L(Y = c, f(X)) = 1$ if $f(X) \neq c$ and 0 otherwise.
- We obtain

$$\sum_{c=1}^C L(Y = c, f(X)) P(Y = c \mid X) = \sum_{c=1}^C I(f(X) \neq c) P(Y = c \mid X) = 1 - P(Y = f(X) \mid X)$$

since we are adding up all of the probabilities apart from $P(Y = f(X) \mid X)$.

Optimal f and optimal risk: classification case

- As for the regression case,

$$R(f) = \mathbb{E}_X \left[\mathbb{E}_Y [L(Y, f(X)) \mid X] \right].$$

- For zero-one loss we have

$$R(f) = \mathbb{E}_X [1 - P(Y = f(X) \mid X)].$$

- To minimise $R(f)$ we simply need to find f that minimises $1 - P(Y = f(X) \mid X)$.
- This is the f that maximises $P(Y = f(X) \mid X)$.
- Thus we must use the f that chooses Y to be the most probable class under the conditional distribution $P(Y \mid X)$ (which, as before, is unknown to us).
- This is called the *Bayes classifier*, and its error rate is called the *Bayes risk*.

Revisiting k -nearest neighbour

- Recall

$$\hat{Y} = \text{mode}_{\{l | X_{(l)} \in N_k(x)\}} \{ Y_{(l)} \}$$

where $N_k(X)$ is the neighbourhood of x defined by the k closest (according to some metric) points in the training data.

- The set $\{Y_{(l)} \mid l \in \{X_{(l)} \in N_k(x)\}\}$ is an approximate draw from $P(Y \mid X = x)$. The approximations are:
 - the approximation is an empirical average over the training data;
 - instead of conditioning on a point X , we are conditioning on a neighbourhood close to the point.
- The mode of this set is approximately the value of Y for which $P(Y \mid X = x)$ is maximised.
- Will this always be the best method?

Summary

- Risk, defined through a loss function, gives us a means for assessing the predictions of models.
- Using the training data, we can estimate risk for different choices of f within $\bar{\mathcal{F}}$, and this gives us a criterion for choosing the model (ERM),
- We show that the optimal f :
 - for regression, when using squared error loss, is $\mathbb{E}_Y[Y \mid X]$;
 - for classification, when using zero-one loss, is $P(Y \mid X)$.
- k -nearest neighbour is a simple method for approximating these optimal f s.

A problem with ERM

- Can we use ERM to choose between a linear model and k -nearest neighbour?
 - No! We will in most situations choose k -nearest neighbour, since it gives us a function closer to the training data.
- Can we use ERM to choose the most appropriate value of k for k -nearest neighbour?
 - No! k will be chosen to be as small as possible.
- Is ERM doing the most useful thing? If not, why not?
 - Find out next time in ST420.

Further reading

- ISLR chapter 2.
- ESL chapter 2.
- PRML chapter 1.

Excess risk and the bias-variance tradeoff

ST420 Lecture 4

Richard Everitt

Loss, risk, etc

- X and Y are random variables, and P is their joint distribution (which is unknown).
- $f \in \mathcal{F}$ is a function from \mathcal{X} to \mathcal{Y} .
- Loss:

$$L : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$$

gives a distance $L(Y, f(X))$ between the real Y and predictions $f(X)$.

- Risk:

$$R(f) = \mathbb{E}_{X,Y} [L(Y, f(X))]$$

is the expected loss over P .

Optimal function

- Denote the optimal $f \in \mathcal{F}$, i.e. the function achieves the minimum risk according to the chosen loss, by f^* .
- Regression with squared error loss: this is given by

$$f^*(X) = \mathbb{E}_Y[Y | X].$$

- Classification with zero-one loss: this is given by

$$f^*(X) = \arg \max_Y P(Y | X).$$

ERM for estimation

- Training data \mathcal{T} is a random sample $\{(X_i, Y_i)\}_{i=1}^n$
- Empirical risk: estimate of risk based on training data

$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^n L(Y_i, f(X_i)).$$

- Consider a restricted space of functions $\bar{\mathcal{F}}$.
- Choose the f that minimises the empirical risk: call it \hat{f} .

Questions about using ERM

- Does \hat{f} give a similar risk to f^* ?
 - If not, what is the reason for the discrepancy?
 - Let \bar{f}^* be the optimal f in $\bar{\mathcal{F}}$. Does \hat{f} give a similar risk to \bar{f}^* ?
- There may be problems when using ERM to choose models of different complexity within $\bar{\mathcal{F}}$
 - it will always choose the one that fits the training data most closely
 - but this \hat{f} may not generalise to new data
 - what should we do?

Excess risk

- We wish to understand the errors arising from fitting a model $\hat{f} \in \bar{\mathcal{F}}$ using ERM, compared to the optimal situation.
- We study the *excess risk*

$$R(\hat{f}) - R(f^*).$$

- The training data \mathcal{T} is a random sample
 - \hat{f} depends on \mathcal{T} , so it is also random
 - the excess risk depends on \hat{f} so it is also random.

Expected excess risk

- It is easier to study the expectation of the excess risk across different training datasets

$$\mathbb{E}_{\mathcal{T}} \left[R(\hat{f}) - R(f^*) \right].$$

- The issue with ERM is that our fit may be too tailored to one particular dataset, and may not generalise.
- By looking at the expectation over \mathcal{T} , we will see what happens across possible training datasets.

Expected excess risk

- In the case of squared error loss, we have that the excess risk is given by

$$R(\hat{f}) - R(f^*) = \mathbb{E}_X \left[(\hat{f}(X) - f^*(X))^2 \right]$$

(exercise).

- Then

$$\begin{aligned} \mathbb{E}_{\mathcal{T}} [R(\hat{f}) - R(f^*)] &= \mathbb{E}_{\mathcal{T}} \left[\mathbb{E}_X \left[(\hat{f}(X) - f^*(X))^2 \right] \right] \\ &= \mathbb{E}_X \left[\mathbb{E}_{\mathcal{T}} \left[(\hat{f}(X) - f^*(X))^2 \right] \right] \end{aligned}$$

Expected excess risk

- Now

$$\begin{aligned}\mathbb{E}_{\mathcal{T}} \left[(\hat{f}(X) - f^*(X))^2 \right] &= \mathbb{E}_{\mathcal{T}} \left[\hat{f}(X)^2 - 2\hat{f}(X)f^*(X) + f^*(X)^2 \right] \\ &= \mathbb{E}_{\mathcal{T}} \left[\hat{f}(X)^2 \right] - 2f^*(X)\mathbb{E}_{\mathcal{T}} \left[\hat{f}(X) \right] + f^*(X)^2 \\ &= \mathbb{V}_{\mathcal{T}} \left[\hat{f}(X) \right] + \mathbb{E}_{\mathcal{T}} \left[\hat{f}(X) \right]^2 - 2f^*(X)\mathbb{E}_{\mathcal{T}} \left[\hat{f}(X) \right] + f^*(X)^2 \\ &= \mathbb{V}_{\mathcal{T}} \left[\hat{f}(X) \right] + \left(\mathbb{E}_{\mathcal{T}} \left[\hat{f}(X) \right] - f^*(X) \right)^2.\end{aligned}$$

Bias

- $\left(\mathbb{E}_{\mathcal{T}}[\hat{f}(X)] - f^*(X) \right)^2$ is the squared bias.
- It tells us how much error we expect due to the trained model differing from the optimal regression.
- This will depend on how restrictive our class of models $\bar{\mathcal{F}}$ is: our model *complexity*. Recall from lecture 2:
 - linear regression may have a large bias if the linear assumption is not appropriate;
 - k -nearest neighbour has a smaller bias when k is small, and larger bias when k is large.
- A large bias occurs when our model *underfits* the data.

Variance

- $\mathbb{V}_{\mathcal{T}}[\hat{f}(X)]$ gives the variance of the fitted function across different data training sets.
- It tells us how much error we expect due to having estimated the function from a training set.
- Again this depends on model complexity. Recall again from lecture 2:
 - linear regression usually has a low variance compared to k -nearest neighbour;
 - k -nearest neighbour has a larger variance when k is small, and smaller variance when k is large.
- A large variance occurs when our model *overfits* the training data.

Example: k -nearest neighbour

- Look at the regression case where $Y = f(X) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$
- Simple to understand this model by looking at $\mathbb{E}_Y \mathbb{E}_{\mathcal{T}} [L(Y, \hat{f}(x_0)) \mid x_0]$ for a fixed x_0 in the case of squared error loss.

$$\mathbb{E}_Y \mathbb{E}_{\mathcal{T}} [L(Y, \hat{f}(x_0)) \mid x_0] = \sigma_\epsilon^2 + \left[f(x_0) - \frac{1}{k} \sum_{l=1}^k f(x_{(l)}) \right]^2 + \sigma_\epsilon^2/k.$$

- σ_ϵ^2 is the irreducible error
- $f(x_0) - \frac{1}{k} \sum_{l=1}^k f(x_{(l)})$ is the bias, which increases with k (since we are including $x_{(l)}$ far away from x_0)
- σ_ϵ^2/k is the variance (recall this is the expression for the variance of a sample mean).

Returning to excess risk

- We would like the excess risk $R(\hat{f}) - R(f^*)$ to be small.
- Consistency results that describe the convergence in probability of $R(\hat{f})$ to $R(f^*)$.
- *Strong universal consistency* holds when

$$R(\hat{f}) - R(f^*) \rightarrow 0,$$

as $n \rightarrow \infty$ with probability 1, regardless of the distribution P .

- Under conditions on the regularity of f , holds for k -nearest neighbour with $n, k \rightarrow \infty$ with $k/n \rightarrow 0$ (but not if k is fixed).
- For many classes of functions we will not have consistency, but we can look at bounds for the excess risk.

Decomposing excess risk

- Decomposing excess risk:

$$R(\hat{f}) - R(f^*) = \underbrace{\left(R(\hat{f}) - R(\bar{f}^*) \right)}_{\text{estimation error}} + \underbrace{\left(R(\bar{f}^*) - R(f^*) \right)}_{\text{approximation error}}.$$

- *Estimation error* compares the risk of the best possible f - call it \bar{f}^* - in our choice of function space to the risk of our estimate obtained using ERM.
- *Approximation error* compares the risk of the best possible f to the best f in our choice of function space.
- For a fixed $\bar{\mathcal{F}}$, the approximation error is deterministic: it simply tells us about the effect of the choice $\bar{\mathcal{F}}$.
- Approximation error is usually difficult to estimate. For there to be any chance of it converging to 0 (when $f^* \notin \bar{\mathcal{F}}$), we need $\bar{\mathcal{F}}$ to change with n .

Estimation error

- The estimation error $R(\hat{f}) - R(\bar{f}^*)$ is random since it depends on \hat{f} .
- Much of learning theory aims to bound the estimation error (so that we can bound the excess risk).
- It can be decomposed as follows

$$R(\hat{f}) - R(\bar{f}^*) = \underbrace{\left(R(\hat{f}) - \hat{R}(\hat{f}) \right)}_{(1)} + \underbrace{\left(\hat{R}(\hat{f}) - \hat{R}(\bar{f}^*) \right)}_{(2)} + \underbrace{\left(\hat{R}(\bar{f}^*) - R(\bar{f}^*) \right)}_{(3)}.$$

- We consider the terms one at a time.

Estimation error decomposition (2)

- Term (2) is $\hat{R}(\hat{f}) - \hat{R}(\bar{f}^*)$.
- The difference between the empirical risk for the ERM estimated function \hat{f} and for the $\bar{\mathcal{F}}$ optimal function \bar{f}^* .
- We know that $\hat{R}(\hat{f}) \leq \hat{R}(\bar{f}^*)$ by definition of \hat{f} , so (2) ≤ 0 .
- Thus

$$R(\hat{f}) - R(\bar{f}^*) \leq \underbrace{\left(R(\hat{f}) - \hat{R}(\hat{f}) \right)}_{(1)} + \underbrace{\left(\hat{R}(\bar{f}^*) - R(\bar{f}^*) \right)}_{(3)}.$$

Estimation error decomposition (3)

- Term (3) is $\hat{R}(\bar{f}^*) - R(\bar{f}^*)$.
- The difference between the true and empirical risk for the $\bar{\mathcal{F}}$ optimal function \bar{f}^* .

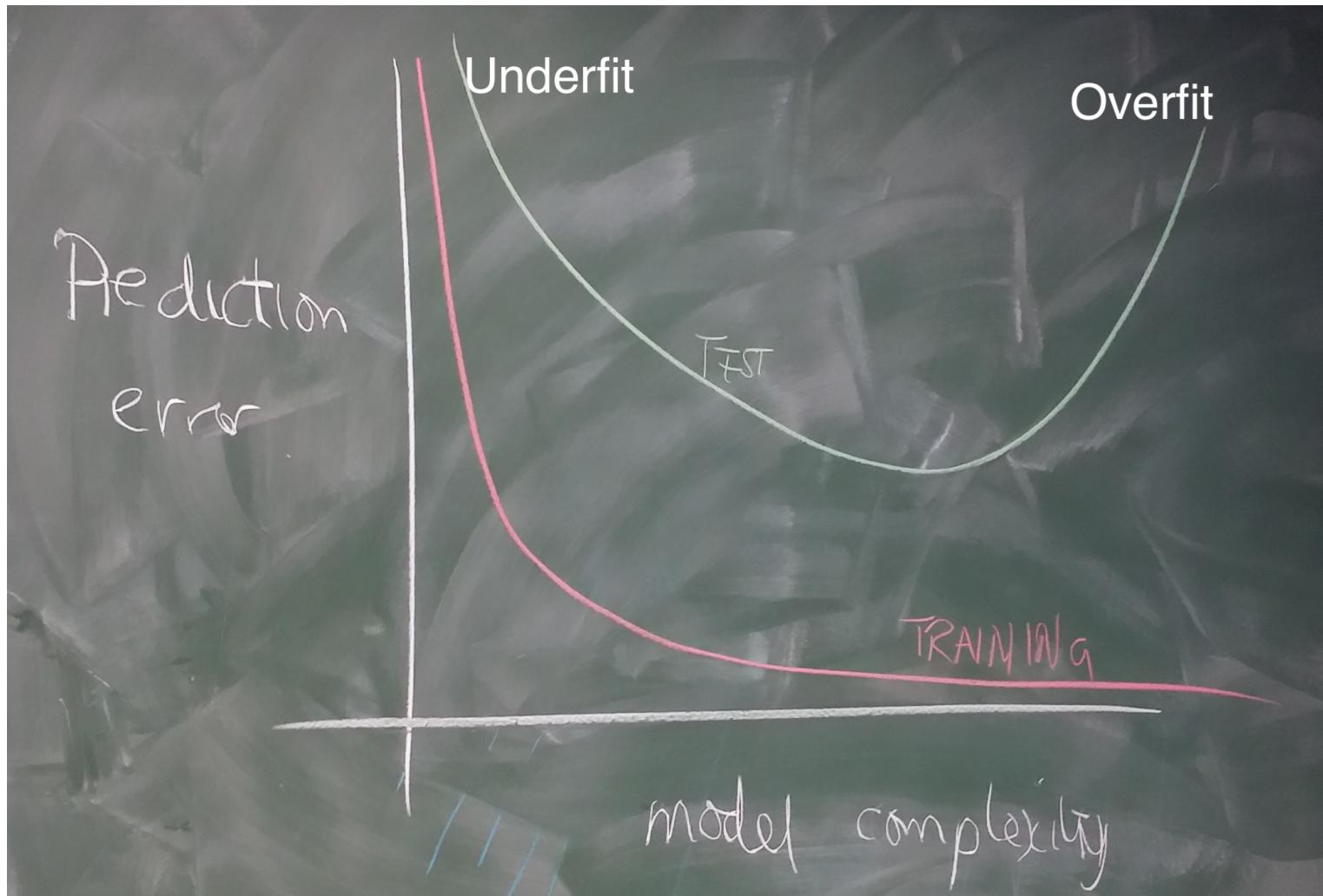
Estimation error decomposition (1)

- Term (1) is $R(\hat{f}) - \hat{R}(\hat{f})$.
- This is the difference between the true and empirical risk for ERM estimated function \hat{f} .
 - $\hat{R}(\hat{f})$ is the training error of \hat{f}
 - $R(\hat{f})$ is essentially the test error of \hat{f}
 - $R(\hat{f}) - \hat{R}(\hat{f})$ is the *generalisation error* of \hat{f} .
 - can be thought of as the extent to which we overfit to the training data
 - usually positive.
- It is random, due to the dependence on the training data.
- Bounds for this are studied later in the module.

Generalisation to test data

- Two important factors:
 - complexity of the model
 - size of the data.
- Restrictive → complex models
 - underfit → overfit
 - more interpretable → less interpretable (when used for inference).
- Small $n \rightarrow$ large n
 - less overfitting.
- As n increases, we can fit more complex models.

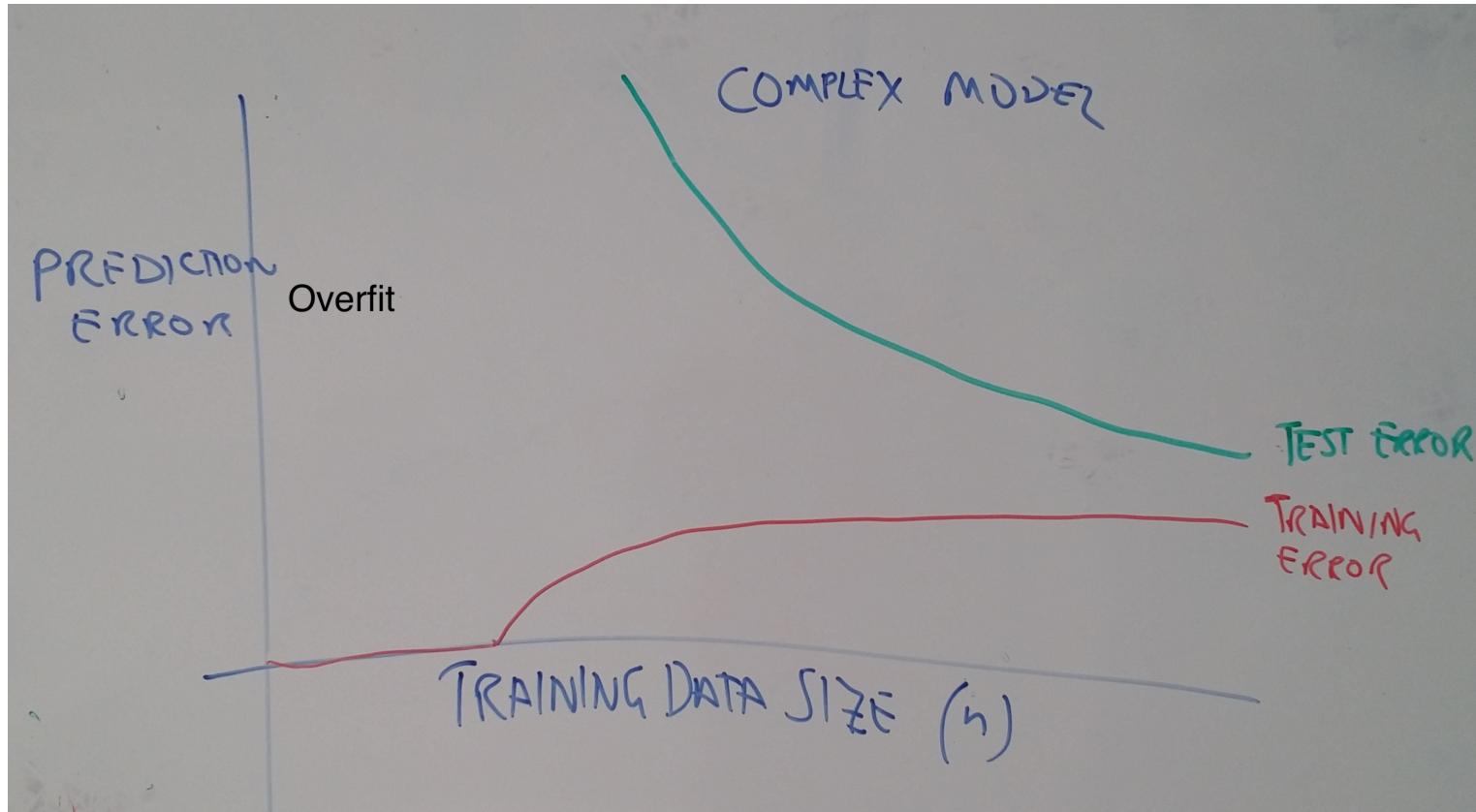
Prediction error against complexity



Prediction error against training data size (simple model)



Prediction error against training data size (complex model)



In-sample error

Optimism and expected optimism

- Define the *optimism* to be

$$\text{op}(\hat{f}) = R^{\text{in}}(\hat{f}) - \hat{R}(\hat{f})$$

- not quite the same as the excess risk (uses in-sample error instead of risk).
- Difficult to estimate, but can sometimes estimate the *expected optimism*

$$\omega(\hat{f}) = \mathbb{E}_Y [\text{op}(\hat{f})]$$

- here the expectation is over training sets as with expected excess risk, but recall that only Y is random in R^{in} , thus the notation \mathbb{E}_Y .

Properties of expected optimism

- Quite generally, it can be shown that

$$\omega = \frac{2}{n} \sum_{i=1}^n \text{Cov}(\hat{y}_i, y_i)$$

(true for squared error and zero-one loss, for example).

- For a linear model when $Y = f(X) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$

$$\omega = 2 \frac{d}{n} \sigma_\epsilon^2.$$

Model comparison methods

- C_p statistic, when d parameters are fit using squared error loss

$$C_p = \hat{R}(f) + 2 \frac{d}{n} \hat{\sigma}_\epsilon^2$$

where $\hat{\sigma}_\epsilon^2$ is an estimate of the MSE of a low-bias model.

- Akaike information criterion (AIC), when d parameters are fit using a log-likelihood loss

$$\text{AIC} = -\frac{2}{N} \sum_{i=1}^n p(y_i \mid \hat{\theta}) + 2 \frac{d}{n} .$$

- For a Gaussian model with variance σ_ϵ^2 , these are equal.

Summary

- Our prediction error (risk) is made up of three terms:
 - irreducible error
 - bias
 - variance.
- Excess risk quantifies how good our predictions are compared to the optimal situation.
- The excess risk does not usually go to zero, but we can bound it.
- The estimation error is the important quantity to bound in order to bound the excess risk.
- A bound on the generalisation error will bound the estimation error
 - links to our intuition built up in previous lectures (hopefully!).

Next

- In the next few lectures we will explore methods for classification.
- We will return to learning theory in a couple of weeks.

Further reading

- ISLR chapter 2.
- ESL chapter 2, chapter 7.
- PRML chapter 1.

Generative classification

ST420 Lecture 5

Richard Everitt

Recap

- We introduced some simple regression and classification techniques.
- We formalised:
 - assessing the accuracy of a model;
 - the process of fitting models;
 - ideas such as a fitted model generalising beyond the training data.

Classification

- X and Y are random variables, and P is their joint distribution (which is unknown).
- Recall that our aim is to find an estimate $\hat{f} : \mathcal{X} \rightarrow \mathcal{Y}$
 - given predictors $X \in \mathcal{X}$ we wish to make accurate predictions of $Y \in \mathcal{Y}$ by using $\hat{f}(X)$
 - we find an estimate \hat{f} from training data $\mathcal{T} = \{(X_i, Y_i)\}_{i=1}^n$.
- Plan
 - Introduce some of the most important methods for classification.
 - Return to the theory later.

Probabilistic approaches to classification

- Probabilistic classification methods give a probability distribution over the different possible classes.
- Suppose we wish to model the joint distribution $P(X, Y)$. There are two approaches we could use:
 - **Generative.** Choose a model for the factorisation

$$P(X, Y | \theta) = P_{X|Y}(X | Y, \theta_{X|Y})P_Y(Y | \theta_Y).$$

- **Discriminative.** We choose a model for the factorisation

$$P(X, Y | \theta) = P_{Y|X}(Y | X, \theta_{Y|X})P_X(X | \theta_X).$$

- Either approach is valid.

Generative approaches

- What is the model doing?
- $P_Y(Y | \theta_Y)$ can be thought of as a prior distribution on the model.
- $P_{X|Y}(X | Y, \theta_{X|Y})$ can be thought of as a model for the data within each class. We consider the case where X is continuous, and let $p_{X|Y}$ be the corresponding density.
- Use the formula

$$P_{Y|X}(Y = y | X = x, \theta_Y, \theta_{X|Y}) = \frac{p_{x|Y}(x | Y = y, \theta_{X|Y})P_Y(Y = y | \theta_Y)}{\sum_{\bar{y} \in \mathcal{Y}} p_{X|Y}(X | Y = \bar{y}, \theta_{X|Y})P_Y(Y = \bar{y} | \theta_Y)}$$

to find the probability of each class given $X = x$. Note that the denominator is normalising the distribution. Here we are simply finding the numerator for each class (\bar{y} is indexing each class), and dividing by the sum of all of these values.

- To train the classifier, we need to find estimates of the parameters θ_Y and $\theta_{X|Y}$.

Advantages and disadvantages

- Advantages
 - A natural way to construct a model for the data - i.e. in being able to use a model for the data within each class.
- Disadvantages
 - Although it is natural to specify the model in this way, the approach will only be effective if the model for the data is chosen well.
 - We needed to use the formula on the previous slide to obtain our classifier $P(Y | X)$, whereas this is estimated directly in the discriminative case. Whilst the parameters $\theta_Y, \theta_{X|Y}$ will be estimated so as to provide a good fit to the data, this is not quite the same as ensuring the optimal performance of the classifier.

Discriminative approaches

- We are directly modelling the distribution $P_{Y|X}(Y | X, \theta_{Y|X})$ that we need for classification.
- The distribution $P_X(X | \theta_X)$ is not needed.
- To train the classifier, we need to find parameters $\theta_{Y|X}$.

Advantages and disadvantages

- Advantages
 - We are directly modelling the classification problem, rather than the distribution of the data within each class. This might be an advantage when the distribution of the data for each class is complicated, but it is actually quite easy to discriminate between two classes.
- Disadvantages
 - This class of approaches can be more like a “black-box” - it may be difficult to interpret why they make the classification decisions that they do.

Linear discriminant analysis

- Use a generative approach

$$P(X, Y | \theta) = P_{X|Y}(X | Y, \theta_{X|Y})P_Y(Y | \theta_Y).$$

- Let P_Y be a discrete distribution on classes: it gives the probability of seeing each class
 - the parameter vector θ_Y gives the probability of each class Y .
- Let $P_{X|Y}$ be a (multivariate) Gaussian distribution, with the same (co)variance for each class.
 - $\theta_{X|Y}$ is the mean of the Gaussian for class Y , and the class-independent (co)variance.
- Training the model involves estimating θ_Y and $\theta_{X|Y}$ from training data.

Learning (estimation): θ_Y

- Usually these parameters are estimated using maximum likelihood (although they do not have to be).
- We give the solution when there are two classes, so that Y can take values in $\{0, 1\}$
- In this case θ_Y is a single parameter: the probability of being in class 1. This is given by $\pi_1 = \frac{n_1}{n}$, where n_1 is the number of data points in the training data that are in class 1.

Learning (estimation): means

- The mean for each class y is given by the sample mean of the points in that class.
- Let $I_y = \{1 \leq j \leq n \mid Y_j = y\}$ be the indices of the data labelled as class y (note that $n_y = |I_y|$).
- Then

$$\mu_y = \frac{1}{n_y} \sum_{j \in I_y} X_j.$$

Learning (estimation): covariance

- This is the weighted average of the sample covariance of the points in each class y , each weighted by n_y/n .
- For each class we have

$$\Sigma_y = \frac{1}{n_y} \sum_{j \in I_y} (X_j - \mu_y)^2,$$

and the covariance we use is

$$\Sigma = \sum_{y \in \mathcal{Y}} \frac{n_y}{n} \Sigma_y.$$

Example

- Let the points $(1, 2, 3)$ be labelled with class 1, and the points $(3, 5, 7, 9)$ be labelled with class 0.
- Part 1: train the classifier
 - estimate the parameters using maximum likelihood (i.e. the equations above)
- Part 2: classify a new point
 - classify the point $x = 3$.

Train the classifier

- We have

$$\pi_1 = \frac{3}{7} \quad \pi_0 = 1 - \pi_1 = \frac{4}{7}$$

$$\mu_1 = \frac{1 + 2 + 3}{3} = 2 \quad \mu_0 = \frac{3 + 5 + 7 + 9}{4} = 6$$

$$\Sigma_1 = \frac{(-1)^2 + 0^2 + 1^2}{3} = \frac{2}{3} \quad \Sigma_0 = \frac{(-3)^2 + (-1)^2 + 1^2 + 3^2}{4} = 5$$

$$\Sigma = \frac{3}{7} \times \frac{2}{3} + \frac{4}{7} \times 5 = 3.142857.$$

Classify the point

- Find the distribution $P_{Y|X}(Y = 1 | X = 3, \theta_Y, \theta_{X|Y})$. (From hereon we omit dependence on the parameters for simplicity.)
- To do this find

$$\frac{p_{X|Y}(x = 3 | Y = 1)P_Y(Y = 1)}{\sum_{\bar{y} \in \mathcal{Y}} p_{X|Y}(x = 3 | \bar{y})P_y(Y = \bar{y})},$$

and likewise for $Y = 0$.

- We have

$$p_{X|Y}(X = 3 | Y = 1)P_Y(Y = 1) = \frac{1}{\sqrt{2\pi\Sigma}} \exp\left(-\frac{(3 - \mu_1)^2}{2\Sigma}\right) \times \frac{3}{7}$$

$$p_{X|Y}(X = 3 | Y = 0)P_Y(Y = 0) = \frac{1}{\sqrt{2\pi\Sigma}} \exp\left(-\frac{(3 - \mu_0)^2}{2\Sigma}\right) \times \frac{4}{7}$$

Classify the point

- Note that the $\frac{1}{\sqrt{2\pi\Sigma}}$ term and the $1/7$ are the same in both cases, so we only need to calculate

$$3 \exp\left(-\frac{(3 - \mu_1)^2}{2\Sigma}\right) \quad 4 \exp\left(-\frac{(3 - \mu_0)^2}{2\Sigma}\right).$$

- Substituting in μ_1, μ_0, Σ , we obtain

$$2.558756 \quad 0.9554968.$$

- Normalising this distribution, we obtain

$$P_{Y|X}(Y = 1 | X = 3) = 0.73$$

$$P_{Y|X}(Y = 0 | X = 3) = 0.27$$

to 2 dp.

Decision boundary

- To make classifications using a probabilistic classifier, for a test point x we find the class

$$\arg \max_{y \in \mathcal{Y}} P_{Y|X}(Y = y \mid X = x).$$

- The set of points in \mathcal{X} that mark the change in the classification decision are the decision boundary
 - these are the points where the classification decision is ambiguous.
- For the binary classification problem, this is the set
$$\{x \in \mathcal{X} \mid P_{Y|X}(Y = 0 \mid X = x) = P_{Y|X}(Y = 1 \mid X = x)\}.$$
- For LDA, what does the decision boundary look like?
 - univariate case first;
 - then multivariate case.

Decision boundary in LDA

- We require $P_{Y|X}(Y = 0 | X = x) = P_{Y|X}(Y = 1 | X = x)$, or alternatively (dropping some subscripts to ease the notation)

$$\log \left[\frac{p(x | Y = 1)P(Y = 1)}{p(x | Y = 0)P(Y = 0)} \right] = 0.$$

- Substituting in the Gaussian densities we obtain

$$\log \left[\frac{\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu_1)^2}{2\sigma^2}\right)}{\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu_0)^2}{2\sigma^2}\right)} \right] + \log \left[\frac{P(Y = 1)}{P(Y = 0)} \right] = 0$$
$$\log \left[\exp\left(\frac{(x - \mu_0)^2}{2\sigma^2} - \frac{(x - \mu_1)^2}{2\sigma^2}\right) \right] + \log \left[\frac{P(Y = 1)}{P(Y = 0)} \right] = 0$$

Decision boundary in LDA

$$\frac{x^2 - 2x\mu_0 + \mu_0^2 - x^2 + 2x\mu_1 - \mu_1^2}{2\sigma^2} + \log \left[\frac{P(Y=1)}{P(Y=0)} \right] = 0$$

$$\left(\frac{\mu_1 - \mu_0}{\sigma^2} \right)x + \frac{\mu_0^2 - \mu_1^2}{2\sigma^2} + \log \left[\frac{P(Y=1)}{P(Y=0)} \right] = 0$$

- Therefore the decision boundary in LDA is linear in x , which we can solve to obtain a single point in this case.

Decision boundary in LDA

- In the multivariate case, substituting in the Gaussian densities we obtain

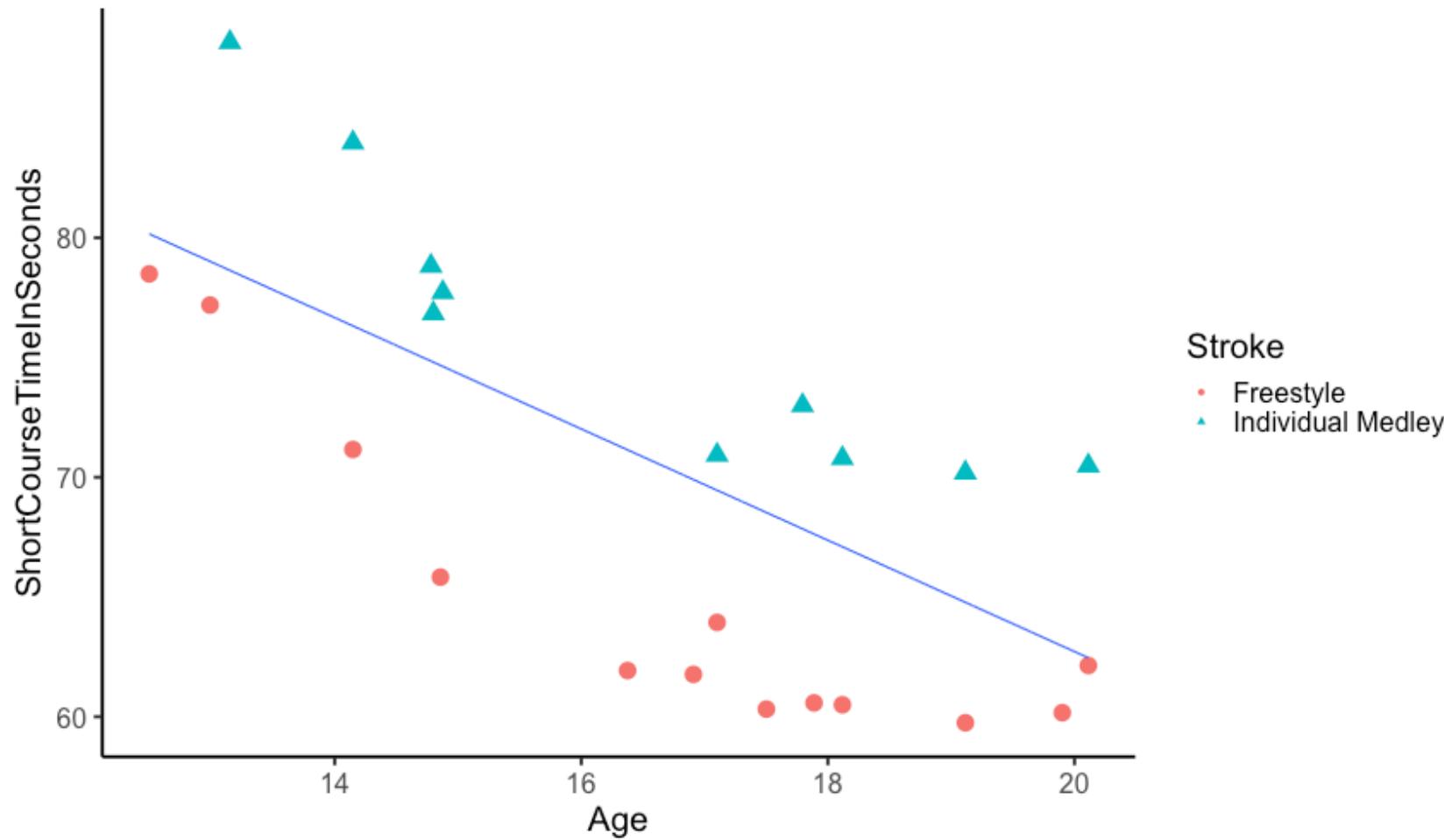
$$\begin{aligned} & \log \left[\frac{\frac{1}{\sqrt{(2\pi)^k |\Sigma_1|}} \exp \left(-\frac{(x-\mu_1)^T \Sigma_1^{-1} (x-\mu_1)}{2} \right)}{\frac{1}{\sqrt{(2\pi)^k |\Sigma_1|}} \exp \left(-\frac{(x-\mu_0)^T \Sigma_1^{-1} (x-\mu_0)}{2} \right)} \right] + \log \left[\frac{p(Y=1)}{p(Y=0)} \right] = \\ & \log \left[\exp \left(\frac{(x-\mu_0)^T \Sigma_1^{-1} (x-\mu_0)}{2} - \frac{(x-\mu_1)^T \Sigma_1^{-1} (x-\mu_1)}{2} \right) \right] + \log \left[\frac{p(Y=1)}{p(Y=0)} \right] = \\ & \frac{x^T \Sigma_1^{-1} x - 2x^T \Sigma_1^{-1} \mu_0 + \mu_0^T \Sigma_1^{-1} \mu_0 - x^T \Sigma_1^{-1} x + 2x^T \Sigma_1^{-1} \mu_1 - \mu_1^T \Sigma_1^{-1} \mu_1}{2} + \log \left[\frac{p(Y=1)}{p(Y=0)} \right] = \end{aligned}$$

Decision boundary in LDA

$$(\Sigma_1^{-1}(\mu_1 - \mu_0))^T x + \frac{\mu_0^T \Sigma_1^{-1} \mu_0 - \mu_1^T \Sigma_1^{-1} \mu_1}{2} + \log \left[\frac{p(Y=1)}{p(Y=0)} \right] = 0$$

- Therefore the decision boundary in LDA is linear: it is defined by a hyperplane.

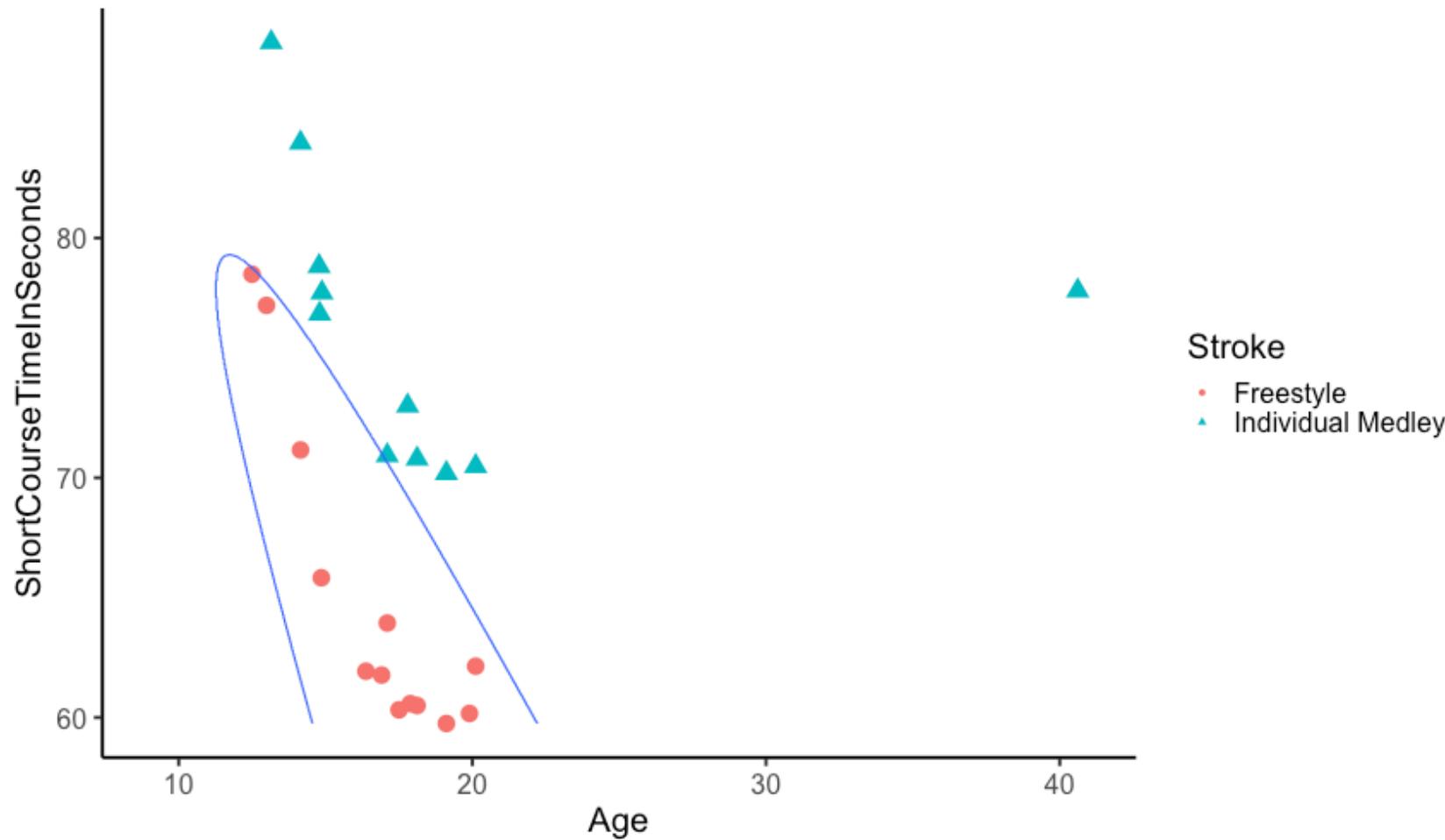
LDA classification



Quadratic discriminant analysis

- Exactly the same idea as LDA, but using different covariance matrices for each within-class Gaussian distribution.
- This results in a quadratic decision boundary.

QDA classification



Decision boundary in QDA

- We require

$$\log \left[\frac{p(x | Y = 1)P(Y = 1)}{p(x | Y = 0)P(Y = 0)} \right] = 0.$$

- Substituting in the Gaussian densities we obtain

$$\log \left[\frac{\frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right)}{\frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right)} \right] + \log \left[\frac{P(Y = 1)}{P(Y = 0)} \right] = 0$$
$$\log \left[\exp\left(\frac{(x - \mu_0)^2}{2\sigma_0^2} - \frac{(x - \mu_1)^2}{2\sigma_1^2}\right) + \log\left(\frac{\sigma_0}{\sigma_1}\right) \right] + \log \left[\frac{P(Y = 1)}{P(Y = 0)} \right] = 0$$

Decision boundary in QDA

$$\frac{x^2 - 2x\mu_0 + \mu_0^2}{2\sigma_0^2} - \frac{x^2 - 2x\mu_1 + \mu_1^2}{2\sigma_1^2} + \log\left(\frac{\sigma_0}{\sigma_1}\right) + \log\left[\frac{P(Y=1)}{P(Y=0)}\right] = 0$$

$$\left(\frac{1}{2\sigma_0^2} - \frac{1}{2\sigma_1^2}\right)x^2 + \left(\frac{\mu_1}{\sigma_1^2} - \frac{\mu_0}{\sigma_0^2}\right)x + \left(\frac{\mu_0^2}{2\sigma_0^2} - \frac{\mu_1^2}{2\sigma_1^2}\right) + \log\left[\frac{\sigma_0}{\sigma_1} \frac{P(Y=1)}{P(Y=0)}\right] = 0$$

- Therefore the decision boundary in QDA is quadratic: we can solve this to find up to two solutions, so that the boundary is defined by two points.
- The multivariate case will be on the problem sheet.

Summary

- Introduced generative and discriminative classification.
- Studied the generative case
 - modelling the within class densities as Gaussian leads to LDA and QDA.
- Next time:
 - logistic regression.

Logistic regression

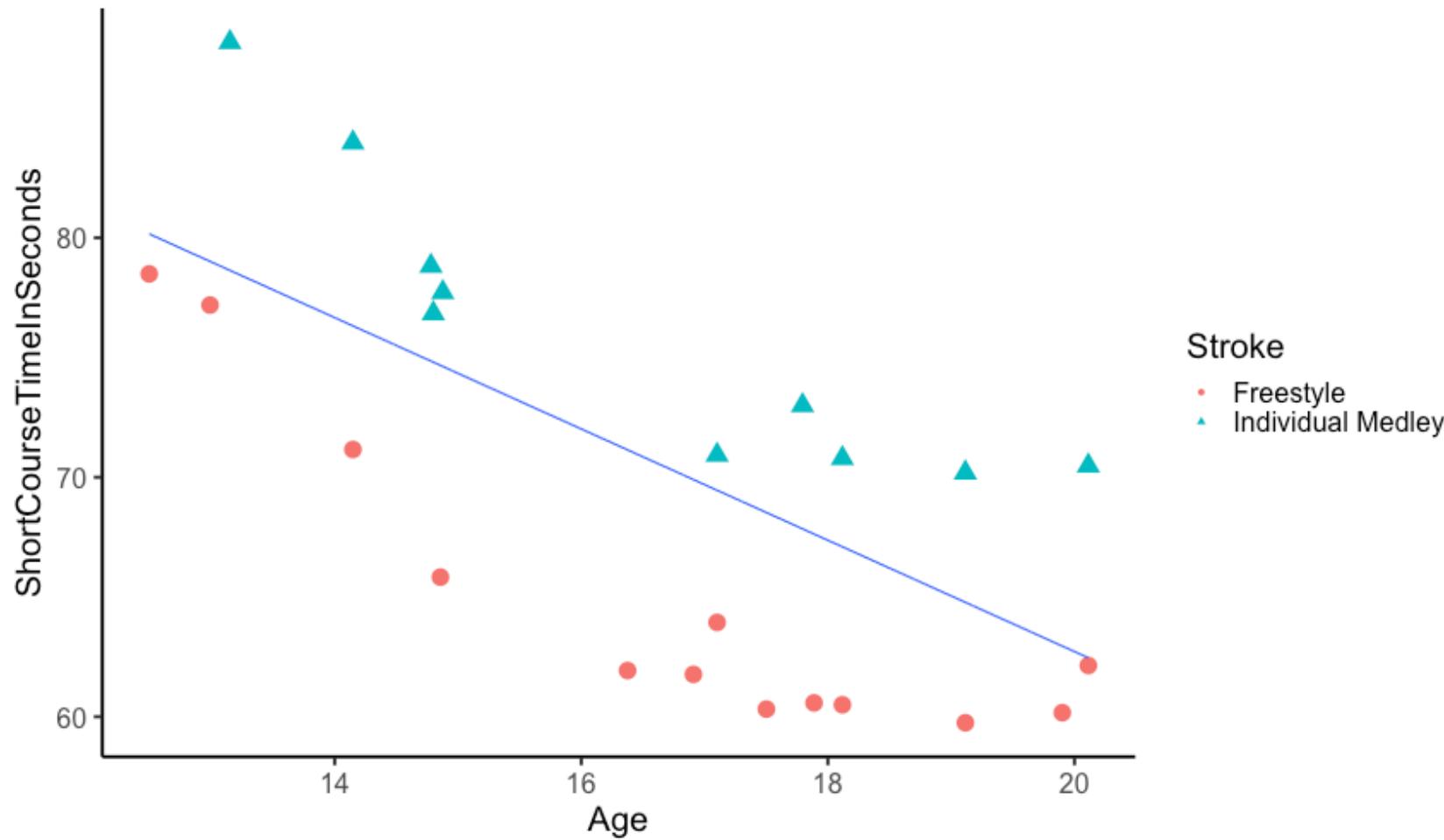
ST420 Lecture 6

Richard Everitt

Recap

- Introduced generative and discriminative classification.
- Generative classification:
 - very useful when we can accurately model the within-class distributions of X ;
 - gives a linear decision boundary when the within-class distributions are modelled as Gaussian with a common (co)variance (LDA);
 - gives a quadratic decision boundary when the within-class distributions are modelled as Gaussian with independent (co)variances (QDA).
- Discriminative classification:
 - directly models $Y|X$.
- Why not directly construct $Y|X$ using a linear decision boundary?

LDA classification



Perceptron

- Separate classes with a linear combination of predictors: idea dates back to Rosenblatt (1958).
- Let Y be a discrete variable that can take values in $\{1, 0\}$
 - classification with 2 classes;
 - note change in notation from lecture 2, so that we fit in with the standard for logistic regression.
- Let $Y = f(X)$ where

$$f(X) = \begin{cases} 1 & \text{if } X^T \beta \geq 0 \\ 0 & \text{if } X^T \beta < 0 \end{cases}.$$

- When X^T contains the intercept term (as previously), and is of length $m + 1$, $X^T \beta$ is the equation of a hyperplane in m -dimensions
 - special case: when $m = 2$, as in the figure in the previous slide, $X^T \beta$ is a line, where $X^T = (1, X^{(1)}, X^{(2)})$.

Deterministic and probabilistic classification and regression

- *Probabilistic classifiers* were discussed in the previous lecture
 - they use a model that describes (directly or indirectly) $P_{Y|X}(Y | X, \beta)$.
- The perceptron is a *deterministic classifier*
 - a deterministic function maps X to Y .
- The analogous situation exists in regression:
 - in "deterministic regression" we simply consider the deterministic function f that maps X to Y ;
 - in "probabilistic regression" we have a full description of the distribution of Y given X (i.e. usually in practice this describes the distribution of points around the "line of best fit").

Perceptron

- Rewrite as a probabilistic classifier to view it in the same framework as logistic regression
 - would like $P_{Y|X}(Y | X, \beta)$
 - rather than mapping X to Y , we would like a distribution on Y given X .
- Rewriting the equation

$$f(X) = \sigma(X^T \beta)$$

where

$$\sigma(a) = \begin{cases} 1 & \text{if } a \geq 0 \\ 0 & \text{if } a < 0 \end{cases}.$$

- Use

$$P_{Y|X}(Y | X, \beta) = \begin{cases} \sigma(X^T \beta) & \text{if } Y = 1 \\ \sigma(-(X^T \beta)) & \text{if } Y = 0 \end{cases}.$$

Log-odds

- The perceptron maps $X^T \beta$ to $\{0, 1\}$.
 - How can we map $X^T \beta$ sensibly to a probability in $[0, 1]$?
- An event having odds $o = o : 1$ (for $o \in \mathbb{R}_{\geq 0}$) of happening means the same as there being a probability of $o/(o + 1)$ that the event happens.
- In our case the event will be the probability of being in class 1 for a binary classification problem.
- We will be dealing with the log of the odds, known as the log odds: this lies in \mathbb{R} .
- There is a bijective function that maps the log odds to probabilities.

Logistic sigmoid and logit

- The logistic sigmoid function maps the log odds to a probability. $\sigma : \mathbb{R} \rightarrow [0, 1]$

$$\sigma(a) = \frac{1}{1 + \exp(-a)}.$$

- Its inverse, the logit function, maps a probability to the corresponding log odds.
 $\text{logit} : [0, 1] \rightarrow \mathbb{R}$

$$\text{logit}(p) = \log\left(\frac{p}{1 - p}\right).$$

- Note that $\sigma(0) = 0.5$
 - intuition: log odds of 0 is and odds of 1:1, which is clearly what we should get in giving a probability of a half.

Logistic regression

- A model for binary classification

$$P_{Y|X}(Y | X, \beta) = \begin{cases} \sigma(X^T \beta) & \text{if } Y = 1 \\ \sigma(- (X^T \beta)) & \text{if } Y = 0 \end{cases}.$$

- Left as an exercise if you want to check that these probabilities sum to 1...
 - follows from the property $\sigma(-a) = 1 - \sigma(a)$ (which you can also show as an exercise).
- What is the decision boundary? We have
$$\sigma(X^T \beta) = 0.5 \quad \Leftrightarrow \quad X^T \beta = 0.$$
- A hyperplane, as found in LDA. Let's re-examine LDA...

Reorganising LDA

$$\begin{aligned} p(Y = 1 \mid x) &= \frac{p(x \mid Y = 1)p(Y = 1)}{p(x \mid Y = 0)p(Y = 0) + p(x \mid Y = 1)p(Y = 1)} \\ &= \frac{\frac{p(x|Y=1)p(Y=1)}{p(x|Y=1)p(Y=1)}}{\frac{p(x|Y=0)p(Y=0)}{p(x|Y=1)p(Y=1)} + \frac{p(x|Y=1)p(Y=1)}{p(x|Y=1)p(Y=1)}} \\ &= \frac{1}{\exp\left(\log\left[\frac{p(x|Y=0)p(Y=0)}{p(x|Y=1)p(Y=1)}\right]\right) + 1} \\ &= \frac{1}{1 + \exp\left(-\log\left[\frac{p(x|Y=1)p(Y=1)}{p(x|Y=0)p(Y=0)}\right]\right)}. \end{aligned}$$

Relationship between LDA and logistic regression

- For LDA, where the covariance Σ is chosen to be the same for both Gaussian distributions, we end up with

$$\sigma(X^T \beta^1 + \beta^0) = 0.5,$$

where

$$\beta^1 = \Sigma^{-1}(\mu_1 - \mu_0)$$

$$\beta^0 = -\frac{1}{2}\mu_1^T \Sigma^{-1} \mu_1 + \frac{1}{2}\mu_0^T \Sigma^{-1} \mu_0 + \log \frac{P_Y(Y=1)}{P_Y(Y=0)}.$$

- For clarity here, we separated out the intercept from the other values of β .

Relationship between LDA and logistic regression

- Similarity
 - both have a linear decision boundary.
- Difference
 - the parameters that determine the linear decision boundary are fit in a different way.
- Which to use (from ESL p129):
 - "generally felt that logistic regression is a safer, more robust bet than the LDA model, relying on fewer assumptions. It is our experience that the models give very similar results, even when LDA is used inappropriately, such as with qualitative predictors."

Fitting logistic regression: loss function

- We will use the loss function induced by our choice of likelihood.
- To ease the notation, we relabel class 0 to be class -1 .
- The likelihood is

$$P_{Y|X}(Y | X, \beta) = \begin{cases} \frac{1}{1 + \exp(-X^T \beta)} & \text{if } Y = +1 \\ \frac{1}{1 + \exp(X^T \beta)} & \text{if } Y = -1 \end{cases} .$$

- With this notation, we can write the likelihood in one expression

$$P_{Y|X}(Y | X, \beta) = \frac{1}{1 + \exp(-YX^T \beta)} .$$

- We will discuss the general f case
 - so far we have used $f(X) = X^T \beta$.

Fitting logistic regression: loss function

- Take the loss function to be $-\log(P_{Y|X})$.
- We have

$$\begin{aligned} L(Y, f(X)) &= -\log(P_{Y|X}) \\ &= -\log\left(\frac{1}{1 + \exp(-Yg(X))}\right) \\ &= \log(1 + \exp(-Yg(X))). \end{aligned}$$

- What is the function g^* that minimises this risk?
- It is

$$g^*(X) = \log\left(\frac{P_{Y|X}(Y = +1 | X)}{P_{Y|X}(Y = -1 | X)}\right).$$

- P here is the true underlying distribution, not our model.

Fitting logistic regression: ERM

- The empirical risk function is given by

$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^n \log(1 + \exp(-Y_i g(X_i))),$$

where we use the usual notation for the training data.

- Recall lecture 3...
- The loss function $L(Y, f(X)) = \log(1 + \exp(-Yg(X)))$ is convex
 - thus, when choosing $g(X) = X^T \beta$, \hat{R} is a convex function of β
 - therefore a local minimum of \hat{R} is a **global minimum**.
- We can thus set up a gradient search method for finding the optimal β
 - details ESL p120.

Generalised linear models

- Logistic regression is a special case of a more general framework.
- Generalised linear models (GLMs):
 - model $\mathbb{E}_Y[Y | X] = h^{-1}(X^T \beta)$
 - $X^T \beta$ known as the *linear predictor*
 - h known as the *link function*
 - the variance $\mathbb{V}_Y[Y | X]$ is typically chosen to be a function of the mean, so $\mathbb{V}_Y[Y | X] = V(h^{-1}(X^T \beta))$.
- In logistic regression we use a Bernoulli distribution for Y
 - its expectation is the probability of "success" (class 1 in our case)
 - the link function is the logit.

Non-linear decision boundaries

- Use a different set of features
 - falls within existing $X^T \beta$ framework (see lecture 2).
- What if we choose a very flexible class of functions?

Summary

- Focussing on discriminative classification.
- Revisited the perceptron, and built on these ideas to introduce logistic regression
 - a popular probabilistic classification technique.
- Showed the relationship between logistic regression and LDA.
- Looked at fitting LDA, and extensions.
- What if we don't know what set of features to use?
 - Next time!

Neural networks

ST420 Lecture 7

Richard Everitt

Recap

- Discriminative classification
 - directly models $Y \mid X$.
- Logistic regression
 - gives a decision boundary of $X^T \beta = 0$
 - within this framework, we can use a set of features that allows us to construct decision boundaries with interesting shapes.
- How to construct the set of features?
- One approach is described in this lecture.

Inspiration for neural networks

- Our brain is a very good classifier (when it has received lots of training data).
- Brains are built of neurons.
- Neurons:
 - take input signals from other neurons, some of which we may think of as positive, and some negative;
 - essentially find a weighted sum of these signals;
 - “fire” (send a signal to the next neurons) if the weighted sum passes some threshold.
- Rough idea:
 - to make an artificial brain, out of artificial neurons, and use this to perform classification tasks.

Neurons

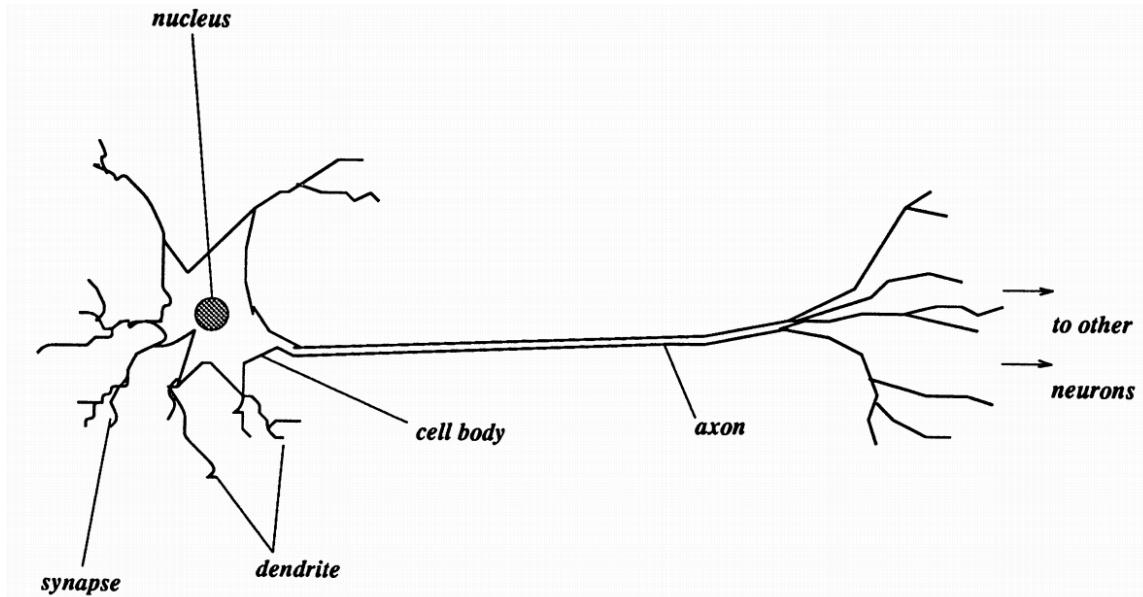
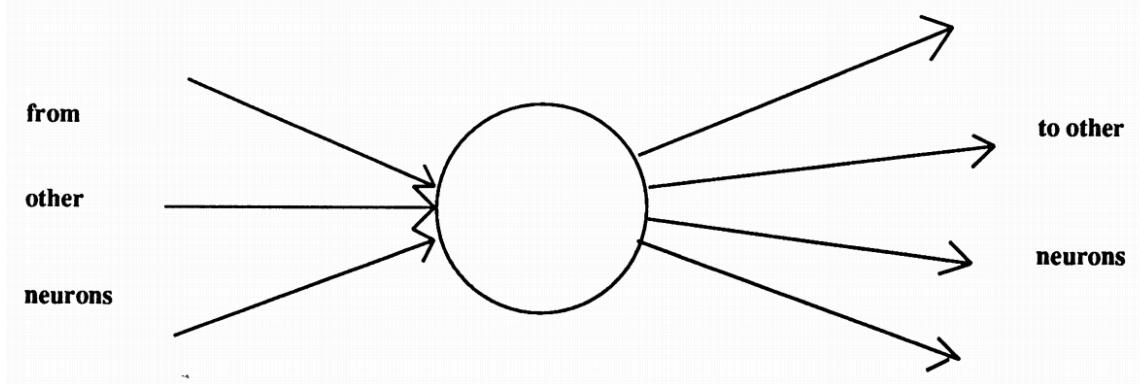


FIG. 2a. Schematic diagram of real neuron.



Neural networks and statistics

Perceptron

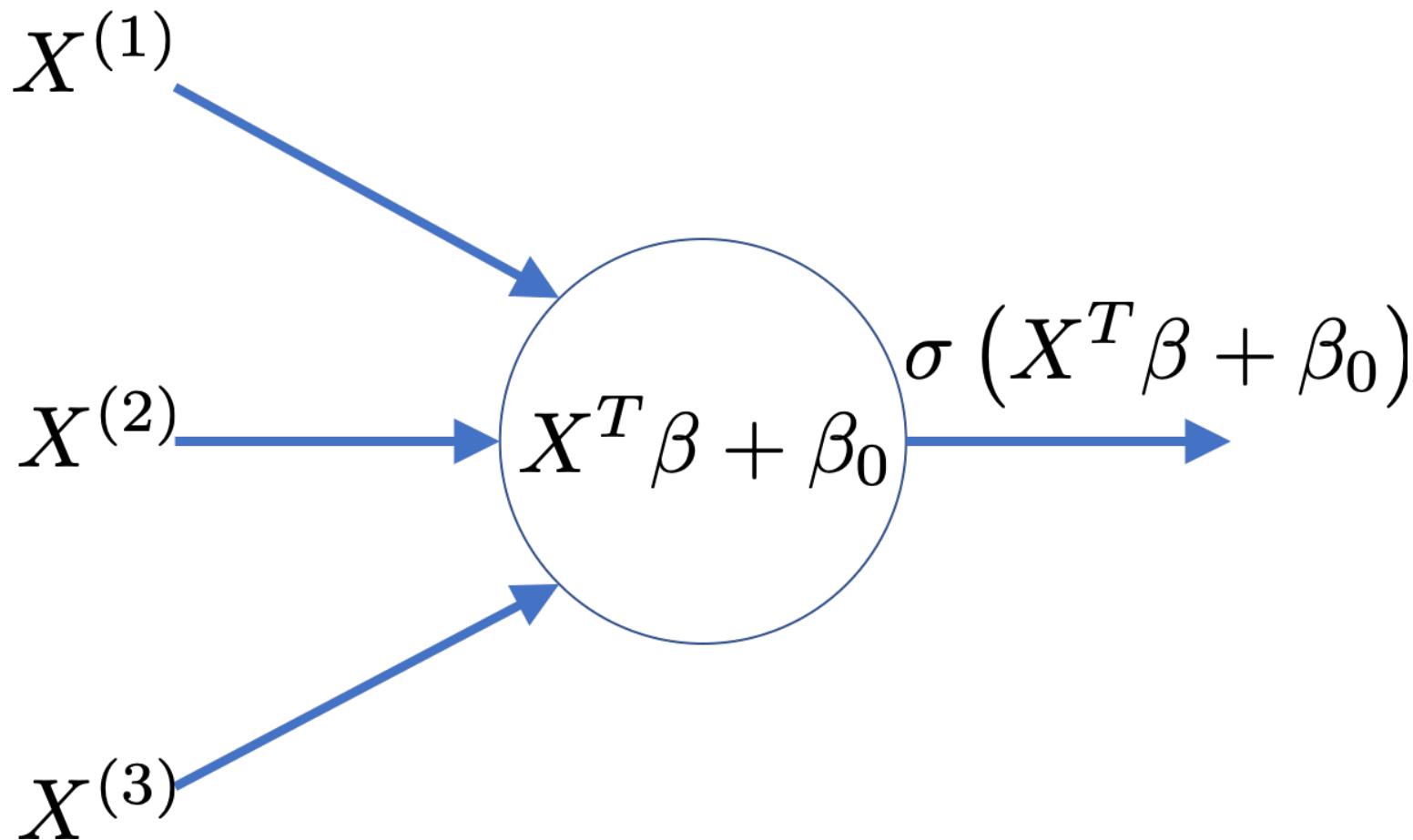
- A model of a neuron.
- Let Y be a discrete variable that can take values in $\{+1, -1\}$
 - classification with 2 classes.
- We write $X^T \beta + \beta^0$ in place of $X^T \beta$
 - for neural networks this will simplify the presentation.
- Use

$$P_{Y|X}(Y = 1 | X, \beta) = \sigma(X^T \beta + \beta^0)$$

where

$$\sigma(a) = \begin{cases} 1 & \text{if } a \geq 0 \\ 0 & \text{if } a < 0 \end{cases}.$$

Illustration of a perceptron



Logistic regression

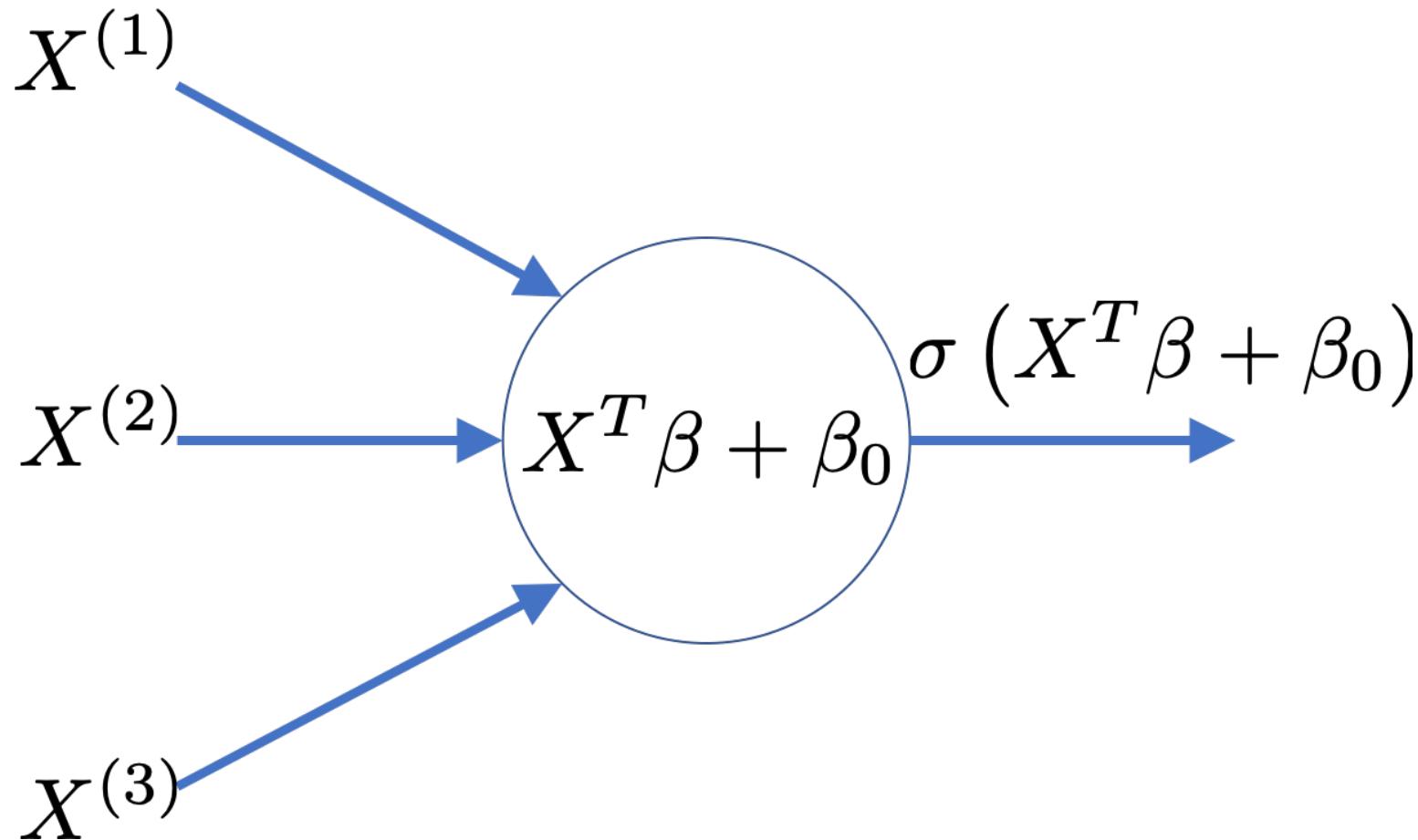
- The same structure as the perceptron.
- Use

$$P_{Y|X}(Y = 1 \mid X, \beta) = \sigma(X^T \beta + \beta^0)$$

where

$$\sigma(a) = \frac{1}{1 + \exp(-a)} .$$

Illustration of logistic regression



Moving to a non-linear decision boundary

- Our decision boundary is determined by the equation $X^T \beta + \beta^0 = 0$.
- We could obtain different decision boundaries by using some set of features

$$X^T = ((\phi^1(X)), (\phi^2(X)), \dots, (\phi^m(X))).$$

- Common choices of features:
 - nonlinear functions to transform the predictors $\phi^j(x) = \log(x)$, \sqrt{x} , etc;
 - piecewise constant $\phi^j(x) = I(L_m \leq x \leq U_m)$;
 - "Gaussian" $\phi^j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2s^2}\right)$;
 - sigmoidal $\phi^j(x) = \sigma\left(\frac{(x-\mu_j)}{s}\right)$ where $\sigma(a) = \left(\frac{1}{1+\exp(-a)}\right)$;
 - Fourier;
 - wavelets.

Neural networks: the key idea of the model

- Let each feature $\phi^i(X)$ itself be some linear combination of features.
- Construct starting from the *input layer* consisting of the input variables $X^{(1)}, \dots, X^{(p)}$
 - let $m = p$ (recall that we use p for the number of X variables, and m used for the number of features).
- Neural networks consist of a number L of layers. The variables at each layer:
 - are functions of those in the layer below
 - are used to construct features for the variables in the layer above.
- Let there be m_l variables in each layer (with $m_0 = m$).
- Let $A_l^{(k)}$ be the k th variable at the l th layer
 - each A variable is known as an *activation*.

Neural networks: description of the model

- 0th layer (input layer)
 - let $A_0^{(k)} = X^{(k)}$.
- All layers
 - let $Z_l^{(k)} = \sigma_l(A_l^{(k)})$
 - σ_l is known as the *activation function* at layer l
 - let $Z_l = (Z_l^{(1)}, \dots, Z_l^{(m_l)})$
 - usually σ_0 is the identity in layer 0
 - usually $\sigma_1 = \dots = \sigma_L$.

Neural networks: description of the model

- l th layer where $1 \leq l \leq L$

$$A_l^{(k)} = Z_{l-1}^T \beta_l^{(k)} + \beta_l^{(k,0)}$$

- $\beta_l^{(k)}$ is a vector of *weights* of length m_l
- $\beta_l^{(k,0)}$ is the "bias".

Neural networks: output layer for regression

- The output layer (layer L) depends on the nature of Y .
- Regression
 - the usual choice is that there is one output variable Y , which has the same status as a single Z variable in that layer

$$Y = \sigma_L(Z_{L-1}^T \beta_L + \beta_L^0)$$

- usually σ_L is the identity
- σ_L may also be other choices commonly used for a GLM.

Neural networks: output layer for binary classification

- Binary classification
 - one output variable Y , with

$$Y = \sigma_L(Z_{L-1}^T \beta_L + \beta_L^0)$$

where σ_L is the logistic function

- thus for $L = 1$ we have logistic regression.

Neural networks: output layer for multi-class classification

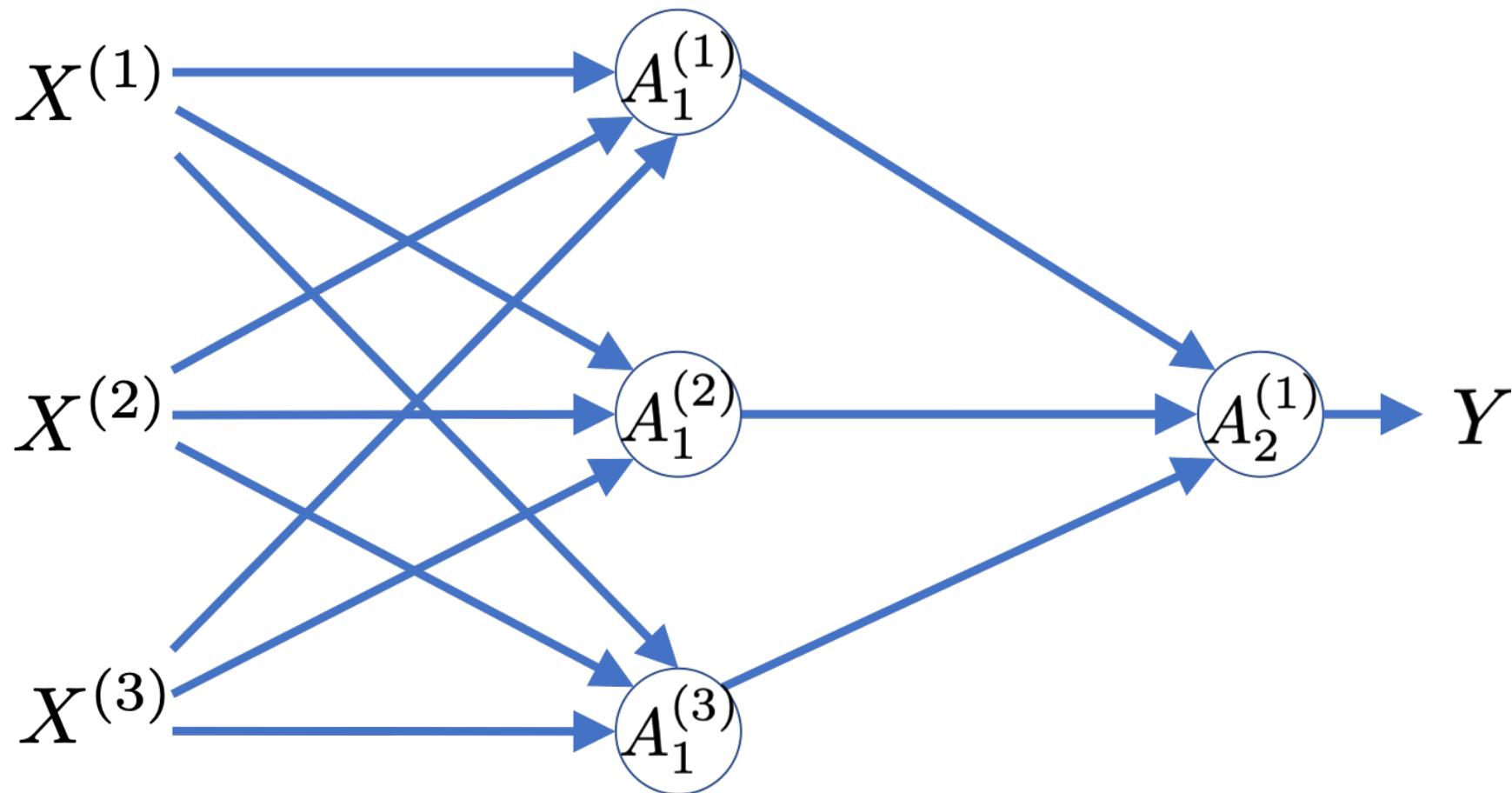
- Multi-class classification with C classes
 - C continuous output variables $Y^{(k)}$, each modelling an unnormalised log probability of being in each class

$$Y^{(k)} = \sigma_L \left(Z_{L-1}^T \beta_L^{(k)} + \beta_L^{(k,0)} \right)$$

- use the *soft-max* function to obtain a discrete output variable Y with K outcomes
 - the probability of state k is given by

$$\frac{\exp(Y^{(k)})}{\sum_{k=1}^C \exp(Y^{(k)})}.$$

Illustration of a neural network



Deep and wide networks

- Deep neural networks
 - L is large.
- Wide neural networks
 - m_l is large.
- Why make these values large?
 - gives a very flexible non-parametric representation for functions.
- Number of parameters can be large!
 - we have many values to fit
 - we might expect to overfit.

Tensorflow playground

Tinker With a **Neural Network** in Your Browser.
Don't Worry, You Can't Break It. We Promise.

(<https://git>

Epoch	Learning rate	Activation	Regularization	Regularization rate	Problem type
000,000	0.03	Tanh	None	0	Classification

DATA

Which dataset do you want to use?

Ratio of training

FEATURES

Which properties do you want to feed in?

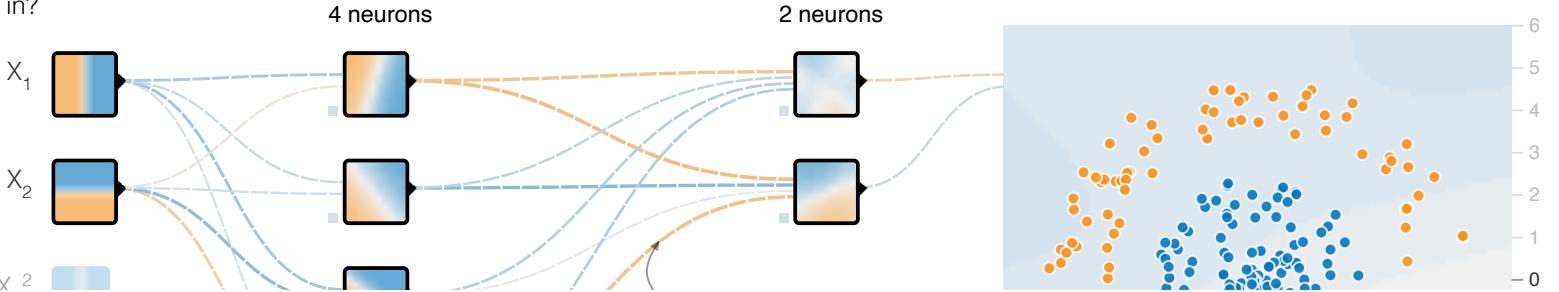
x_1
 x_2
 x_3

2 HIDDEN LAYERS

4 neurons
2 neurons

OUTPUT

Test loss 0.500
Training loss 0.507



Fitting neural networks

- Loss function in the regression case: squared error loss

$$L(Y, f(X)) = \mathbb{E}_P \left[(Y - f(X))^2 \right].$$

- Loss function in the binary $\{0, 1\}$ classification case: cross-entropy (negative log-likelihood) loss

$$L(Y, f(X)) = \mathbb{E}_P \left[Y \log(f(X)) + (1 - Y) \log(1 - f(X)) \right].$$

- Fitted using gradient descent
 - uses the chain rule for differentiation (iterated across layers) to calculate the gradient
 - algorithm is known as back-propagation.
- Not convex
 - risk surface as a function of β is complicated
 - possibly ridges, saddle-points, local minima.

Excess risk

- The excess risk can be large
 - usually since the generalisation error can be large
 - because we have such a flexible model we are likely to overfit.
- However, deep learning can be successful when trained on very large datasets.
- What if we do not have enough data to offset the model complexity?
- How might we combat overfitting?

Combatting overfitting I: ERM with a restrictive $\bar{\mathcal{F}}$

- Use empirical risk minimisation (ERM)

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \hat{R}(f)$$

to fit a model in some restrictive family of functions $\bar{\mathcal{F}}$

- for example, restrict our model to $L = 1$, logistic regression.
- The problem is that we are likely to underfit, without knowing in advance how to choose $\bar{\mathcal{F}}$.

Combatting overfitting II: structural risk minimisation

- Structural risk minimisation (SRM) is the idea of choosing from an infinite sequence of models $\bar{\mathcal{F}}_1, \bar{\mathcal{F}}_2, \dots$, of increasing size, using the empirical risk, and an added penalty for the complexity of the model, i.e.

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}_d, d \in \mathbb{N}} \hat{R}(f) + \text{pen}(m, n).$$

- What is $\text{pen}(m, n)$?
 - penalises models as the number of parameters m gets larger
 - can be offset by n being large.
- Ideally we would choose $\text{pen}(m, n)$ to be the generalisation error
 - not available directly, but we can bound the generalisation error, and use this bound as $\text{pen}(m, n)$ (see later in the module).

Combatting overfitting III: regularisation

- Choose a flexible class $\bar{\mathcal{F}}$.
- Modify ERM by using a *regularising* term.
- For example

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \hat{R}(f) + \lambda \|f\|^2$$

where $\|\cdot\|$ is some norm.

- Called *weight decay* in neural networks.
- We have a free parameter λ , the *regularisation parameter*
 - aim to choose the right trade-off between fit and complexity.
- How choose λ ?
 - usually cross-validation (see later in the module).

Combatting overfitting: other approaches

- Bayesian approach
 - using a prior distribution to regularise.
- Bagging/boosting
 - combining multiple neural networks.
- Neural network specific approaches
 - sharing weights (so that different parts of the network are doing the same operation)
 - local connectivity (each hidden unit is connected to only a small patch of variables in the layer below)
 - dropout (randomly omit variables).

Summary

- Introduced neural networks as an extension to logistic regression.
- Discussed methods for approaching the problem of overfitting.
- Next lecture we will take a step back, and consider the problems of "big models" and "big data".

Further reading

- ESL chapter 11.
- PRML chapter 5.

Big data and big models

ST420 Lecture 8

Richard Everitt

Introducing big data

- Big data is a term that has become popular in the past 10 years.
- It refers to the collection, storage and analysis of large digital data sets.
- Examples:
 - data from social media (twitter/facebook/etc)
 - large Hadron collider (150 million sensors delivering data 40 million times per second)
 - [100,000 genomes project](#) (100,000 genomes, each 3 billion bases long)
 - Atacama Large Millimeter Array (advanced telescope, generates 2Tb of data a day).

Challenges

- Big data brings with it a number of challenges
 - volume (size of the dataset)
 - velocity (the rate at which the data arrives)
 - data quality (may contain errors, missing entries or require some other pre-processing)
 - sampling frame (data is often collected, rather than coming from a designed experiment).
- In this module we focus on the issues relating to volume
 - after a short description of some of the issues in genomics.

Genomic data

- A *genome* is the genetic material of an organism
 - usually consists of deoxyribonucleic acid (DNA), a polymer inside cells
 - made up of a sequence of nucleotides (smaller molecules) that encode information
 - nucleotides can be one of four types (cytosine [C], guanine [G], adenine [A] or thymine [T]).
- The representation of a genome we use for analysis is a string made up of these 4 letters ("bases")
 - e.g. GATATGACTA
 - the human genome consists of 3 billion of these bases, over (usually) 23 chromosome pairs (it is split into different pieces).

Whole genome sequencing

- In order for us to analyse it, somehow the information in the DNA molecule needs to be translated into a string
 - this is achieved using "DNA sequencing".
- DNA sequencing:
 - does **not** give a perfect transcription of the nucleotides in chromosome;
 - provides imperfect transcriptions of smaller sections of the chromosome;
 - these smaller chunks of sequence data overlap.
- Much effort is devoted to joining these smaller chunks of sequence together, using the overlaps to resolve errors
 - this can be a very hard problem
 - a research field of its own.

Variation between individuals

- Differences between corresponding positions in the genome give rise to differences between individuals.
- Consider the following two sequences from two different individuals
 - GATATGACTA
 - GATATTACTA
- The difference at the 6th position is the only interesting position in terms of studying the differences between the individuals
 - this is called a *single-nucleotide polymorphism* (SNP, pronounced "snip")
 - approximately 10 million in the human genome

Further complication

- What did I mean by "corresponding" on the previous slide?
- For most of two genomes, we can "align" them, such that most of the bases in this alignment are the same.
- However, this alignment is not possible for the whole genome
 - in species such as some bacteria, it is not possible to align large sections (thousands of bases) of the genome.
- Take-home message:
 - we might like our sequence data to be in the form of a $n \times p$ matrix (individuals in the rows, and bases in the columns)...
 - but it takes lot of effort to get this point!

Why study big data?

- Because it's there?
- In some cases we hope that we will be able to find out things that we could not before we had the data.
- In others, we hope that we can use models that are more complicated than was possible previously
 - "big models".

This lecture

- The focus is on the impact of supervised learning when there are a large number n of data points and/or when the dimension p of \mathcal{X} is large (a large number of predictors).
- Computational issues:
 - what is the impact of large n and/or p on the CPU time and memory requirements when using supervised learning?
- Algorithmic issues (affects computational time, but worth separate consideration):
 - for cases where the solution cannot be found analytically and an algorithm is used to fit the model
 - what is the impact of large n and/or p on the performance of algorithms that fit the models?
- Statistical:
 - what statistical issues arise (or are solved!) when n and/or p are large?

Notation

- Recall that we have used:
 - p for the number of variables in the data (or, sometimes, the number of predictors)
 - m for the number of parameters of the model (recall polynomial regression).
- For a standard linear regression we will have one parameter for each predictor, so $p = m$.
 - in every case $m \geq p$ since we have at least one parameter for each predictor (if we omit a predictor, we will simply think of this as using a smaller p).
- The number of parameters is usually the important factor, but we will usually use p to follow the convention in the literature.

Computational issues: CPU time

- When n and/or p are large we need to consider the effect on the computational time the methods will require.
- Even if methods scale linearly with n and p , if n and/or p are large enough we may run into computational problems.
- Many of the methods we have studied are more expensive than this, for example:
 - training linear regression is $O(p^2n + p^3)$ if all of the required matrix multiplications and inversions are performed in the most naive way
 - training SVMs uses the dual form, thus is $O(n^2p + n^3)$
 - training LDA is $O(p^2n + p^3)$.

Computational issues: CPU time

- In implementations there are sometimes short-cuts that can improve on these results.

Computational complexity of machine learning algorithms

APRIL 16, 2018

⌚ Reading time ~7 minutes

What is complexity? Good question. I should have addressed it right away. It is a notion which is often addressed in algorithmic classes, but not in machine learning classes... Simply put, say you have a model. Training it on n points takes x minutes. Now what happens if you train it on kn points? If the training time is now less than the training time is linear. Sometimes it is more. The new training time may be $12x$. In this case the

Computational issues: memory

- When n and/or p are large, we may not be able to load the data into RAM in order to perform the required calculations.
- In extreme cases we may not even be able to store all of the data on one hard drive!
- We need to consider the possibility that data is stored on different computers.

Computational issues: a few ideas

- A number of these problems can need to be addressed on the computational side
 - using parallel computing
 - GPUs
 - fast communication between compute nodes.
- There is also a statistical side to this work
 - subsampling the complete dataset, so that we do not need to use samples of size n
 - fusing results from multiple sources (maybe different compute nodes)
 - these ideas need careful justification.

Algorithmic issues

- When fitting some models, there is no analytic solution.
- Instead an algorithm must be run in order to fit the model
 - e.g. stochastic gradient descent.
- Each iteration of such an algorithm will have a particular computational complexity (which will involve n and probably also p)
 - but we also need to consider the ability of the method to explore a space of dimension p .

Algorithmic issues

- This issue becomes particularly interesting when using Bayesian statistics.
- In this case we often need to run a Markov chain Monte Carlo (MCMC) algorithm for exploring the space.
- Studying MCMC algorithms with large n and/or p will be discussed towards the end the module, after we have motivated and described the use of Bayesian statistics.
- An interesting part of this topic is that our intuition about probability distributions in high-dimensional spaces is often wrong.

Statistical issues: estimation and approximation error

- Leaving aside the computational challenges, what are the statistical challenges and opportunities for large n and/or p ?
- Later in the module, we will see that:
 - estimation error $R(\hat{f}) - R(\bar{f}^*)$ usually decreases as n increases, since we get closer to fitting the optimal function within the class;
 - approximation error $R(\bar{f}^*) - R(f^*)$ usually decreases as p increases, since the class of functions grows more flexible, and more able to fit the optimal function.
- However:
 - estimation error $R(\hat{f}) - R(\bar{f}^*)$ usually increases as p increases, since the model overfits (we could compensate for this by increasing n).

Statistical issues: effect of increasing p

- Aside from computational considerations, increasing n helps us train a supervised regression algorithm.
- However, increasing p can be problematic, even putting aside any computational issues or algorithmic issues
 - see the next two lectures.
- Later in the module we will see other issues that arise for large p
 - the "curse of dimensionality".
- This lecture: the multiple testing problem.

Multiple testing: example

- Consider an artificial example that has $p = n$ (taken from David Firth's lecture notes).
- Let's generate random data, 101 variables for a sample of size 100.
- Let's make all the variables iid $N(0,1)$, independently of one another.

```
set.seed(2017)

mydata <- matrix(rnorm(100 * 101), 100, 101)
## 100 rows, 101 columns

mydata <- as.data.frame(mydata)
## a data frame with variables V1,...,V101
```

Multiple testing: example

- Now let us see if we can find evidence of a relationship between `v101` and any of the other 100 variables.
- (We know that, in reality, there is no such relationship!)
- We will compute the simple linear least-squares regression of `v101` on each of the other 100 variables in turn, and look at the estimated slopes and significance tests.

Multiple testing: example

```
mymodel <- lm(V101 ~ 1, data = mydata)
## The empty model, with just an intercept -- as a starting point

## Now let's test all 100 simple regressions that have V101 as
## response variable
add1(mymodel, names(mydata)[1:100], test = "F")
```

```
## Single term additions
##
## Model:
## V101 ~ 1
##          Df Sum of Sq      RSS      AIC F value    Pr(>F)
##                 100.376  2.3749
## V1           1   0.0011 100.374  4.3737   0.0011  0.973444
## V2           1   0.0487 100.327  4.3264   0.0475  0.827843
## V3           1   0.0261 100.349  4.3488   0.0255  0.873421
## V4           1   0.8987  99.477  3.4755   0.8854  0.349044
## ...         1   0.5000  99.500  3.5000   0.5000  0.449000
```

Multiple testing: example

Some of those variables look very promising. For example, V54:

```
library(dplyr)
library(ggplot2)
myplot <- mydata %>% ggplot(aes(x = V54, y = V101)) + geom_point()
myplot + geom_smooth(method = "lm")
```

Multiple testing: example

```
my_significant_model <- lm(V101 ~ V54, data = mydata)
summary(my_significant_model)
```

```
##
## Call:
## lm(formula = V101 ~ V54, data = mydata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.59834 -0.58019 -0.02763  0.62173  2.19275
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 0.11607   0.09609   1.208  0.23001  
## V54         0.29259   0.08915   3.282  0.00143 ** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Multiple testing: example

- An unscrupulous scientist, desperate to publish, might forget to mention that they had looked at 100 regressions in order to find this highly 'significant' one.
- **That would simply be incorrect.** The calculation of significance needs to take account of the whole testing procedure, which in this case involved 100 separate significance tests.
- This is a "voodoo correlation".

Aside: type I and type II error

- An easy way to remember which is which is to recall the story of the "Boy Who Cried Wolf".
- The null hypothesis is that there is no wolf.
- The story is in two parts:
 1. To begin with, the villagers believe there is a wolf when there really is no wolf - this is a **type I error**.
 2. In the end, the villagers believe there is no wolf when there really is a wolf - this is a **type II error**.

Multiple testing: family-wise error rate

- A "false rejection" or "false positive" is a type I error.
- Suppose we perform many tests.
- Let A_j be the event that test j is falsely rejected. This is our significance level α .
- The *family-wise error rate* (FWER) is the probability of at least one false rejection
 - i.e. a result such as we found in the example.
- For p independent tests, the FWER is

$$1 - (1 - \alpha)^p.$$

- Positive dependence reduces the FWER.

Multiple testing: Bonferroni correction

- Idea: reduce the FWER to α by reducing the significance level on every test.
- The *Bonferroni correction* chooses the significance level for every test to be α/p
 - guarantees that the FWER is $\leq \alpha$.
- This follows by using the union bound.
- It is often too conservative (in that it too often fails to detect significant results), especially when:
 - there is positive dependence between the test statistics;
 - or p is large.

Multiple testing: false discovery rate

- Idea: rather than look at the probability of at least one false rejection, look at the expected proportion of false rejections out of all rejections
 - this is known as the *false discovery rate* (FDR).
- What is the difference?
 - when looking at the FWER, we are trying to control for the number of false rejections in **all of the hypothesis tests**
 - when looking at the FDR, we are trying to control for the number of false rejections in **only the cases where the null hypothesis is rejected.**

Multiple testing: Benjamini-Hochberg correction

- Idea: introduce a procedure to keep the FDR to $\leq \alpha$
 - the reason as to why this works is more complicated, and we will not cover it.
- The *Benjamini-Hochberg correction* uses the following procedure:
 1. Fix the false discovery rate α and let $\rho_1 \leq \dots \leq \rho_p$ denote the ordered p-values.
 2. Let
$$M = \max \left\{ j \mid \rho_j < \alpha \frac{j}{p} \right\}.$$
 3. Reject all hypotheses whose p-value $\rho_j \leq \rho_M$.
- Typically the FDR is chosen to be higher than might be used as a significance level, e.g. it might be chosen to be 10 or 15%.

Further reading

- ESL chapter 18.

Regularisation I

ST420 Lecture 9

Richard Everitt

Recap

- Neural networks, and other flexible classification and regression techniques, can overfit and thus fail to generalise from training to test data
 - the generalisation error $R(\hat{f}) - \hat{R}(\hat{f})$ may be large, which leads to a large excess risk.
- There are several ways of combatting overfitting
 - this lecture is about *regularisation*, also known as *shrinkage*.

Combatting overfitting III: regularisation

- Choose a flexible class $\bar{\mathcal{F}}$.
- Modify ERM by using a *regularising* term.
- For example

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \hat{R}(f) + \lambda \|f\|^2$$

where $\|\cdot\|$ is some norm.

- Called *weight decay* in neural networks.
- We have a free parameter λ , the *regularisation parameter*
 - aim to choose the right trade-off between fit and complexity.
- How choose λ ?
 - usually cross-validation (see later in the module).

Using a penalty

- Hand-wavy description of the problem
 - for regression/classifications with an $X^T\beta$ form (linear regression, GLMs, neural networks, etc)
 - if the model class is very flexible, usually m will be large, we have a large number of β parameters and the freedom to choose them so that we get a very small empirical risk
 - we have the freedom to choose β to get a very close fit to the training data.
- Idea
 - when any β^j is small (or zero in the extreme case), the corresponding $\phi^j(X)$ will make only a small (zero) contribution to the regression/classification
 - so instead of minimising the empirical risk, we minimise the empirical risk plus some penalty for the β being large.

Setup

- Let $\bar{\mathcal{F}}$ be such that $f(X) = X^T \beta + \beta^0$
 - as in the lecture on neural networks, we separate out the "intercept" or "bias" term
 - recall that this is **not** the bias in our predictions - it is simply the terminology used in machine learning.

Ridge regression

- Recall that ERM minimises the empirical risk

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \hat{R}(f).$$

- In *ridge regression* we instead use

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \hat{R}(f) + \lambda \|\beta\|_2^2,$$

where $\|\beta\|_2 = \left(\sum_{j=1}^m (\beta^j)^2 \right)^{1/2}$ and $\lambda \geq 0$.

- Also known as *Tikhonov regularisation*, after Andrey Tikhonov who first formulated this idea
 - ridge regression was a rediscovery of the technique.

Ridge regression: remarks

- We have added a penalty that favours small values of β
 - introduced bias, but reduced variance
 - should help us avoid overfitting.
- Choice of λ is important
 - controls the amount of shrinkage
 - when $\lambda = 0$ the regularisation term does nothing (no additional bias, same variance as before)
 - as $\lambda \rightarrow \infty$, the β shrink to 0.

Ridge regression: remarks

- Note the intercept term is not usually included in the regularisation term
 - would make the results depend on the choice of origin for Y .
- All predictors should be scaled before using ridge regression
 - use

$$\tilde{X}_i^{(j)} = \frac{X_i^{(j)}}{\sqrt{\frac{1}{n} \sum_{i=1}^n (X_i^{(j)} - \bar{X}^{(j)})^2}}$$

$$\text{where } \bar{X}^{(j)} = \frac{1}{n} \sum_{i=1}^n X_i^{(j)}$$

- otherwise the scales of the predictors affect the shrinkage.
- In neural network, the penalty would apply to all weights in the network.

Alternative idea: best subset selection

- Rather than shrinking the β values, make a "hard" decision about which of the m variables to include.
- Known as *Best Subset Selection*.
- Usually implemented using a variation on the following idea
 - find the best model (using ERM) when using only one predictor (i.e. try using each predictor on its own, and choose the one that minimises the empirical risk)
 - do the same for all models with 2 predictors, then 3, etc
 - use some approach that penalises model complexity (e.g. AIC, etc) to choose the model complexity.
- Problem: there are 2^m models
 - often cannot search through exhaustively
 - search algorithms are usually greedy and ad hoc.

Lasso

- Ridge regression uses

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \hat{R}(f) + \lambda \|\beta\|_2^2,$$

where $\|\beta\|_2 = \left(\sum_{j=1}^m (\beta^j)^2 \right)^{1/2}$.

- The *lasso* instead uses

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \hat{R}(f) + \lambda \|\beta\|_1,$$

where $\|\beta\|_1 = \sum_{j=1}^m |\beta^j|$.

Beyond the lasso

- In general we could use

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \hat{R}(f) + \lambda \|\beta\|_r^r,$$

where $\|\beta\|_r = \left(\sum_{j=1}^m |\beta_j|^r \right)^{1/r}$.

- Ridge regression is $r = 2$.
- Lasso is $r = 1$.
- We could consider values of $r \geq 0$
 - for $r < 1$, $\|\cdot\|_r$ is not a norm, but we can still use it.

Alternative view

- It can be shown that the following solutions are equivalent...

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \hat{R}(f) + \lambda \|\beta\|_r^r,$$

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \hat{R}(f) \quad \text{subject to } \left(\sum_{j=1}^m |\beta^j|^r \right) \leq s(\lambda).$$

- For each choice of λ there is a number $s(\lambda)$ such that the second formulation is equivalent to the first.
- With this view we can easily see why scaling the predictors makes a difference
 - without the penalty, under least squares, scaling predictor j by a constant c simply results in its corresponding estimate of β^j being scaled by $1/c$
 - with the penalty, such a scaling on one β^j impacts $\left(\sum_{j=1}^m |\beta^j|^r \right)$.

Common framework

- The previous slide shows that ridge and lasso are special cases of a general framework.
- We can also see best subset selection in this way:

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \hat{R}(f) \quad \text{subject to } \sum_{j=1}^m I(\beta^j \neq 0) \leq s.$$

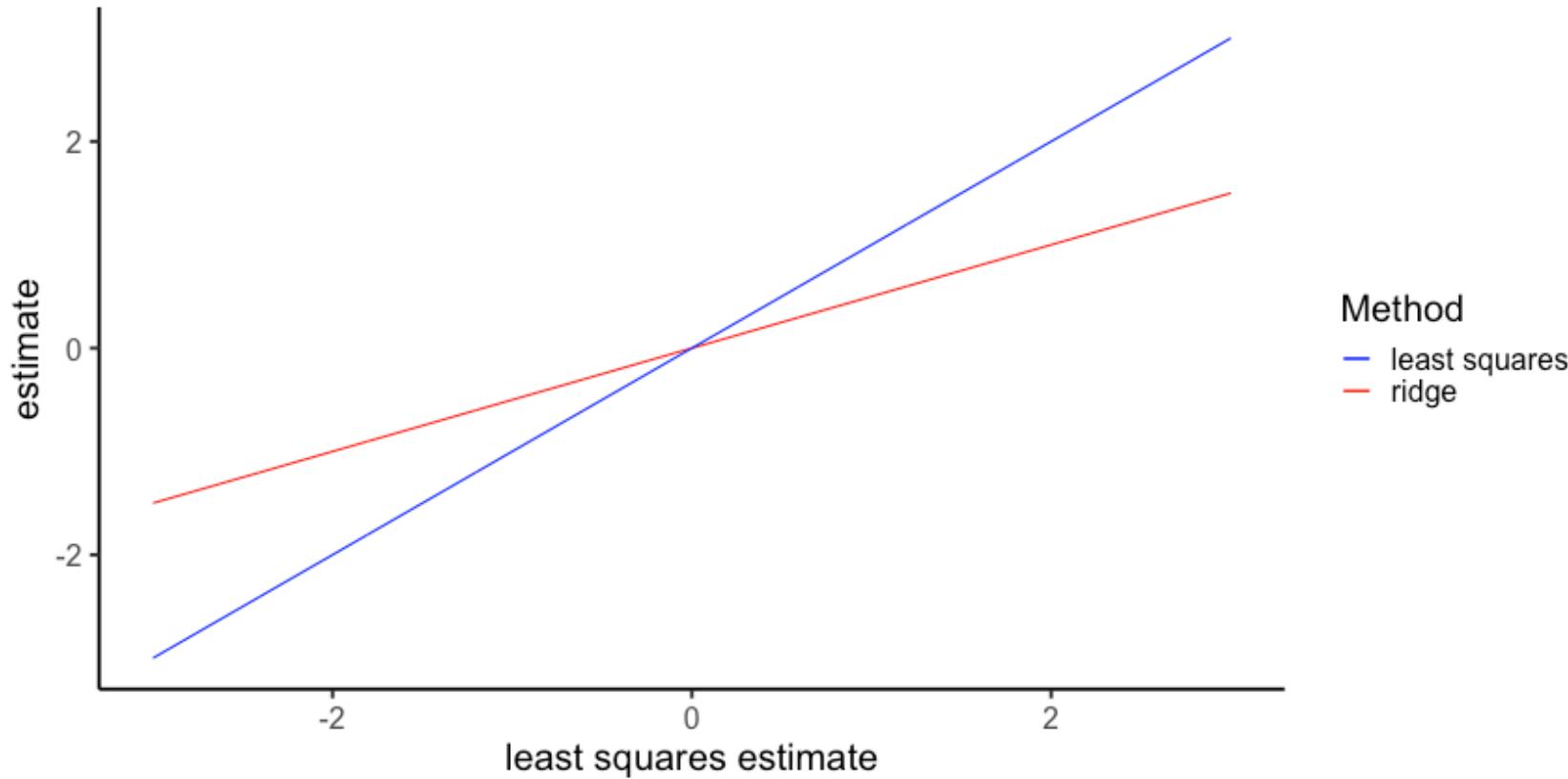
- These techniques have a lot in common
 - all shrink some/all coefficients to 0.
- Which to use?
 - depends on the situation
 - let's take a look at what the practical effect of the constraints are on the coefficients.

Nature of the shrinkage

- Let \mathbf{X} be an orthonormal design matrix
 - unrealistic, but will help us understand
 - more generally, we get approximately the same behaviour.
- In this case, solutions to the three techniques can be found analytically. Each method transforms the least squares estimate $\hat{\beta}^j$:
 - ridge regression does a proportional shrinkage;
 - lasso translates each coefficient by a constant factor, truncating at zero (*soft thresholding*);
 - best subset selection sets to zero variables with coefficients smaller than the s th largest (*hard-thresholding*).

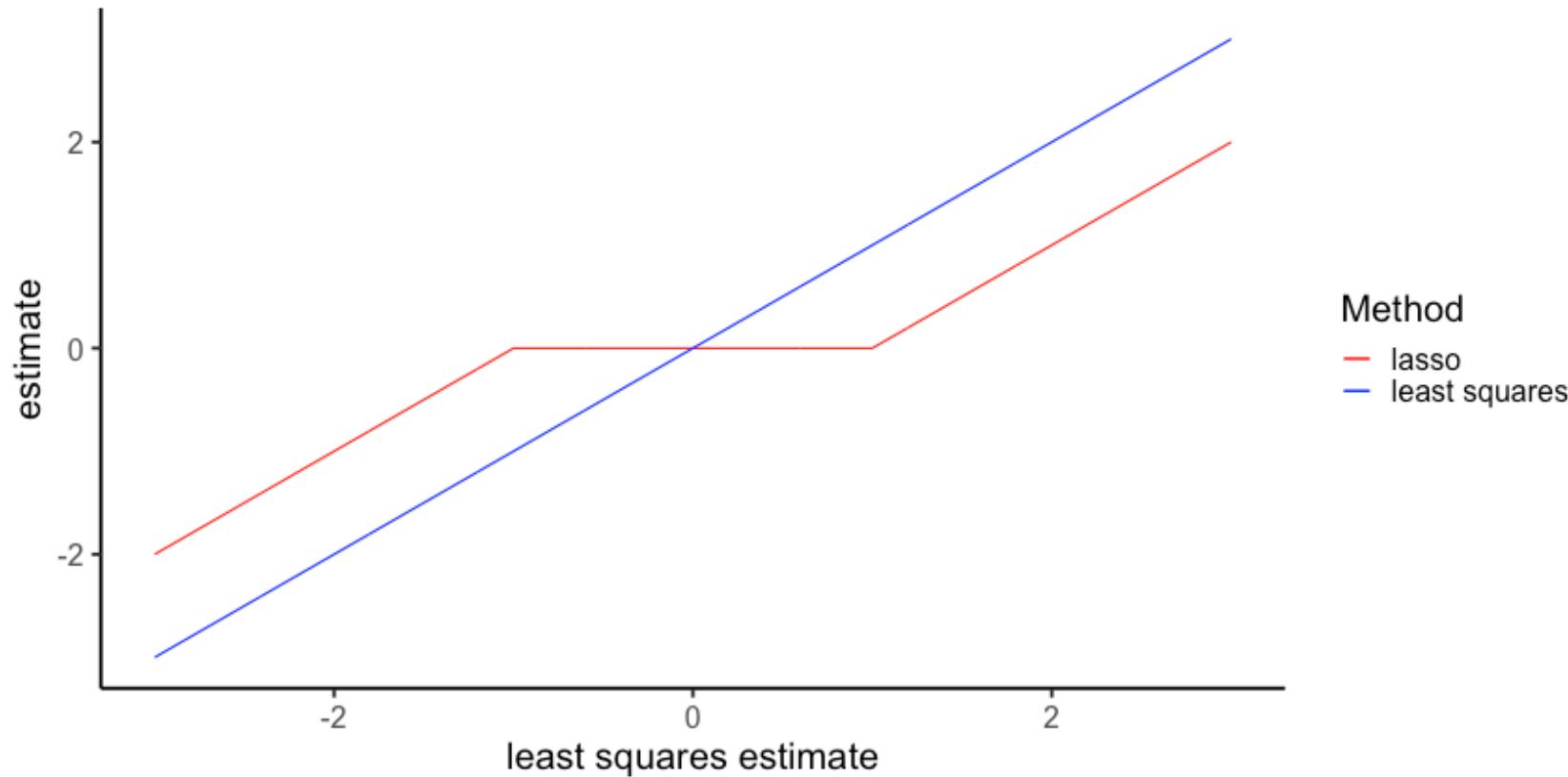
Ridge regression

$$\hat{\beta}^j / (1 + \lambda)$$



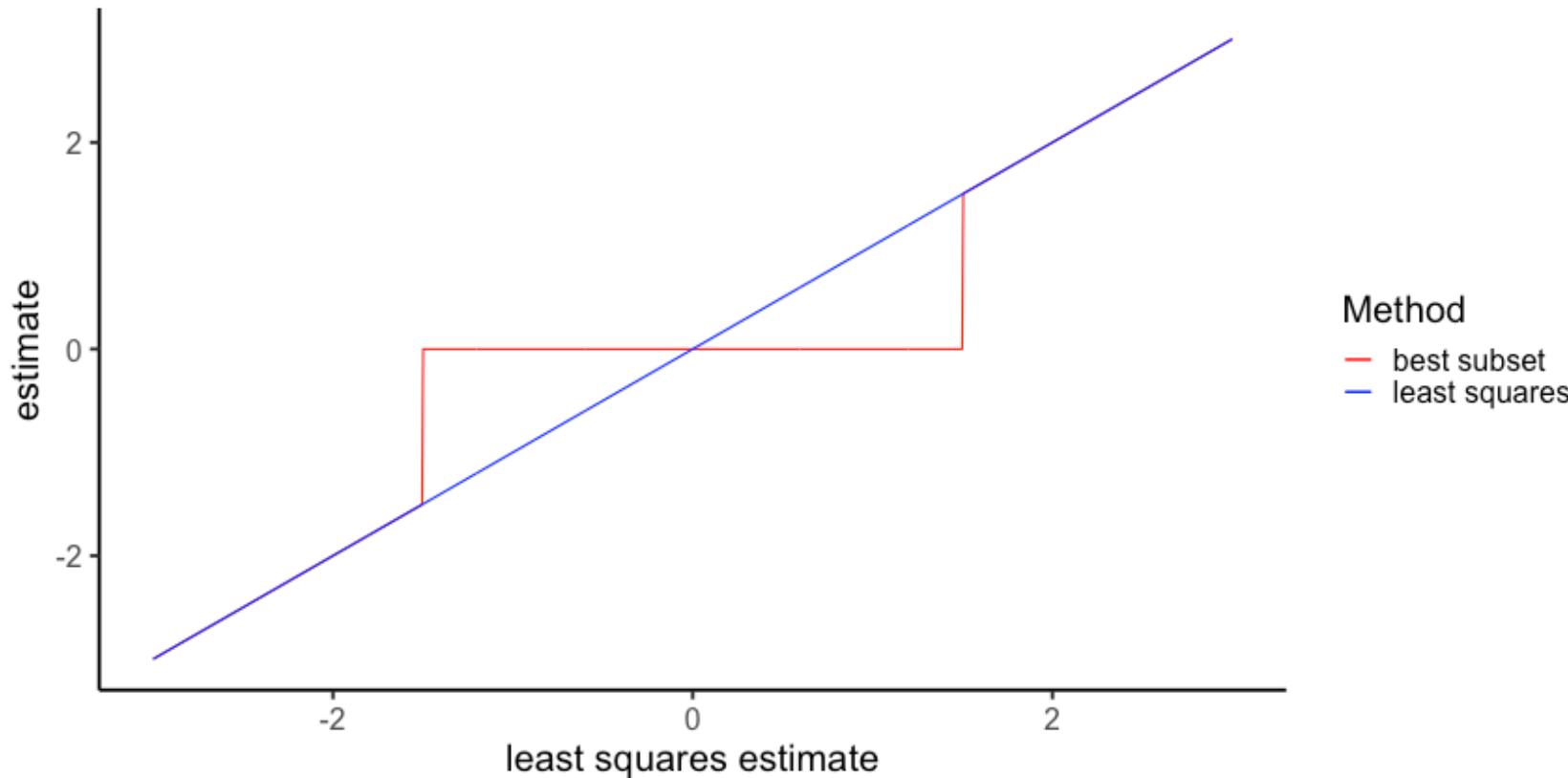
Lasso

$$\text{sign}(\hat{\beta}^j) \left(|\hat{\beta}^j| - \lambda \right)_+$$

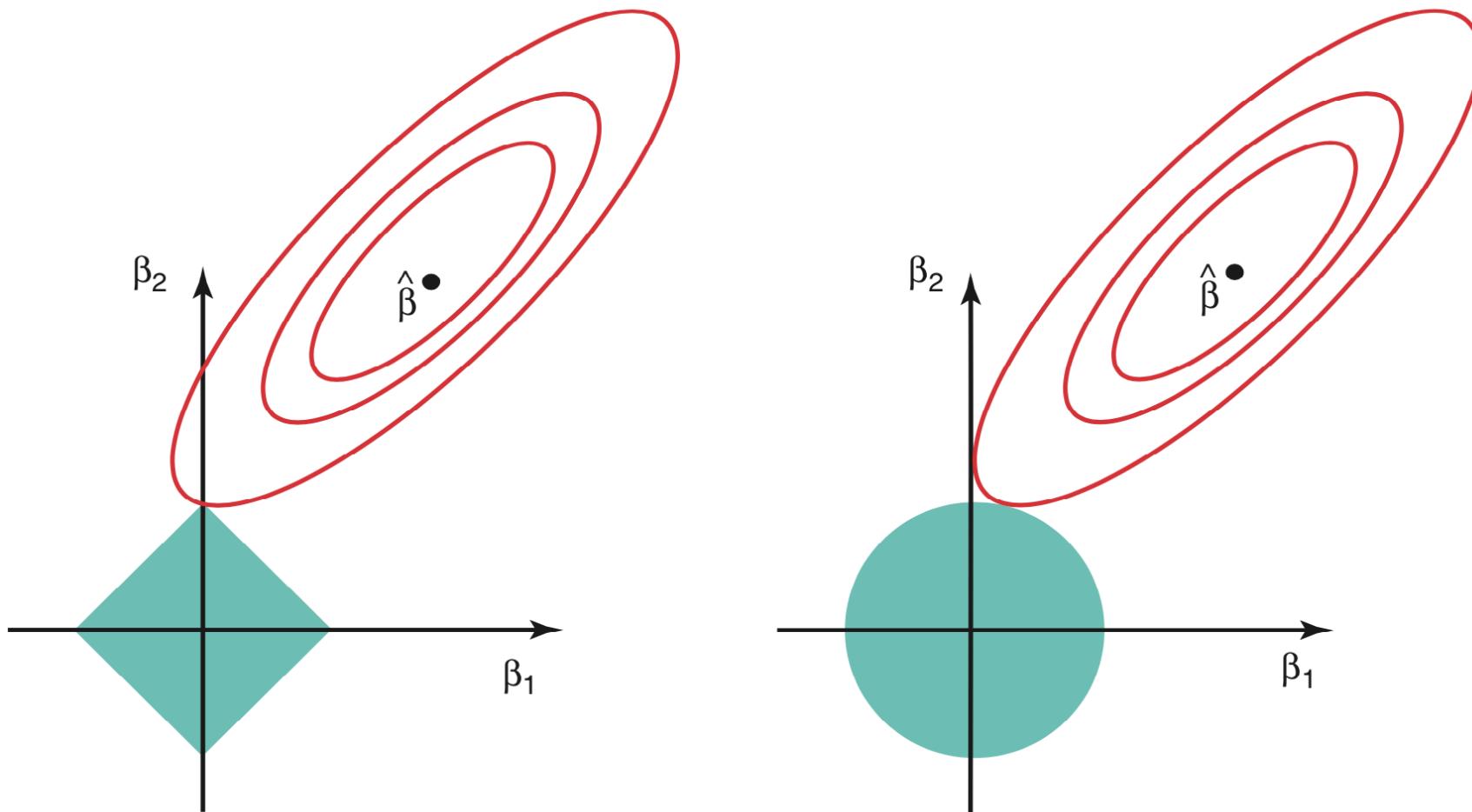


Best subset selection

$$\hat{\beta}^j I\left(\left|\hat{\beta}^j\right| \geq \left|\hat{\beta}^{s\text{th largest}}\right|\right)$$



Why does the lasso do variable selection?



- From ISLR p222.

Example: logistic regression

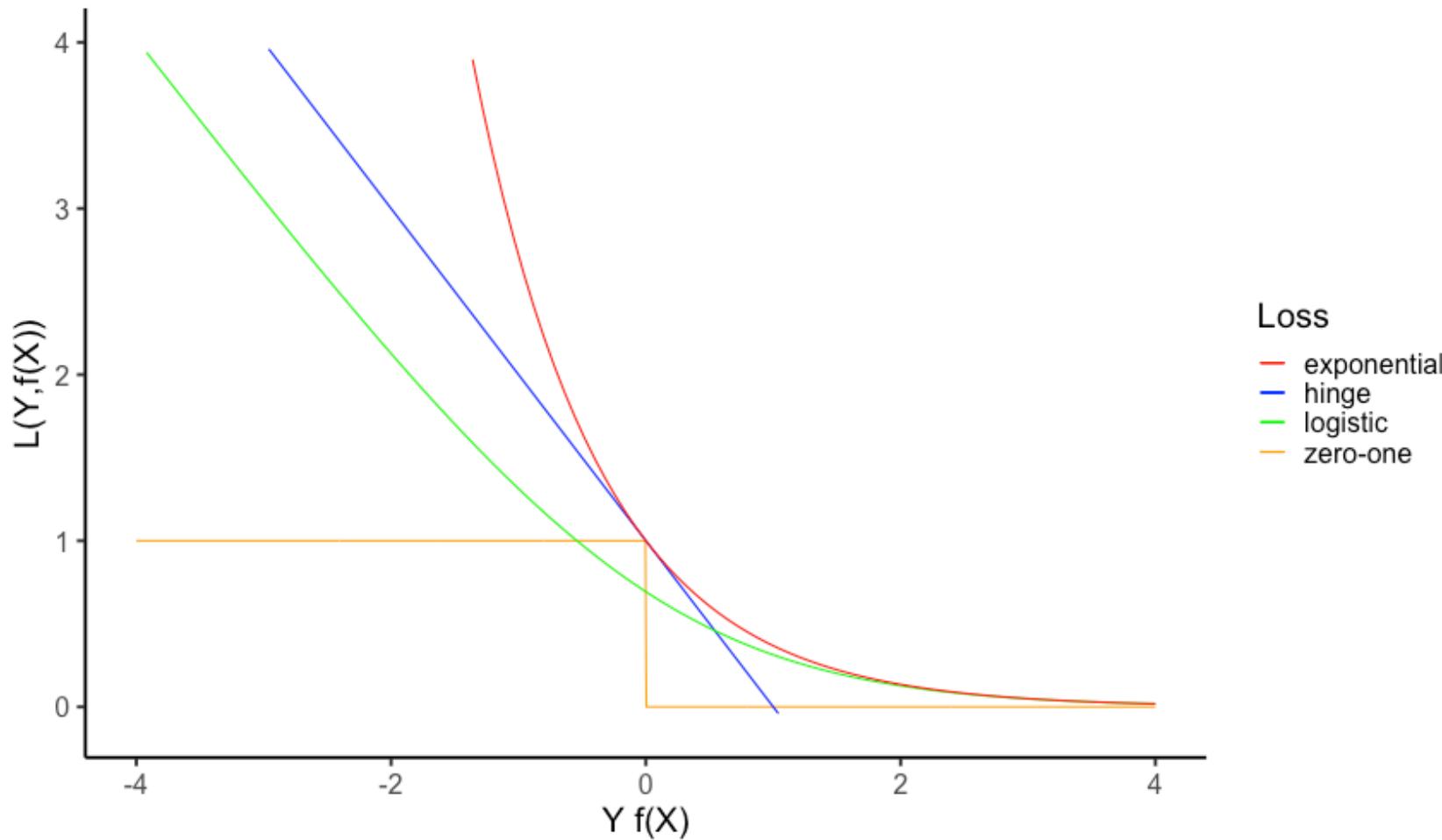
- We used a loss function of $L(Y, f(X)) = \log(1 + \exp(-Yf(X)))$.
- The empirical risk function is given by

$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^n \log(1 + \exp(-Y_i f(X_i))),$$

where we use the usual notation for the training data.

- The loss function $L(Y, f(X)) = \log(1 + \exp(-Yf(X)))$ is convex
 - so there is a **global minimum**.

Sketch of loss functions for classification



Example: logistic regression

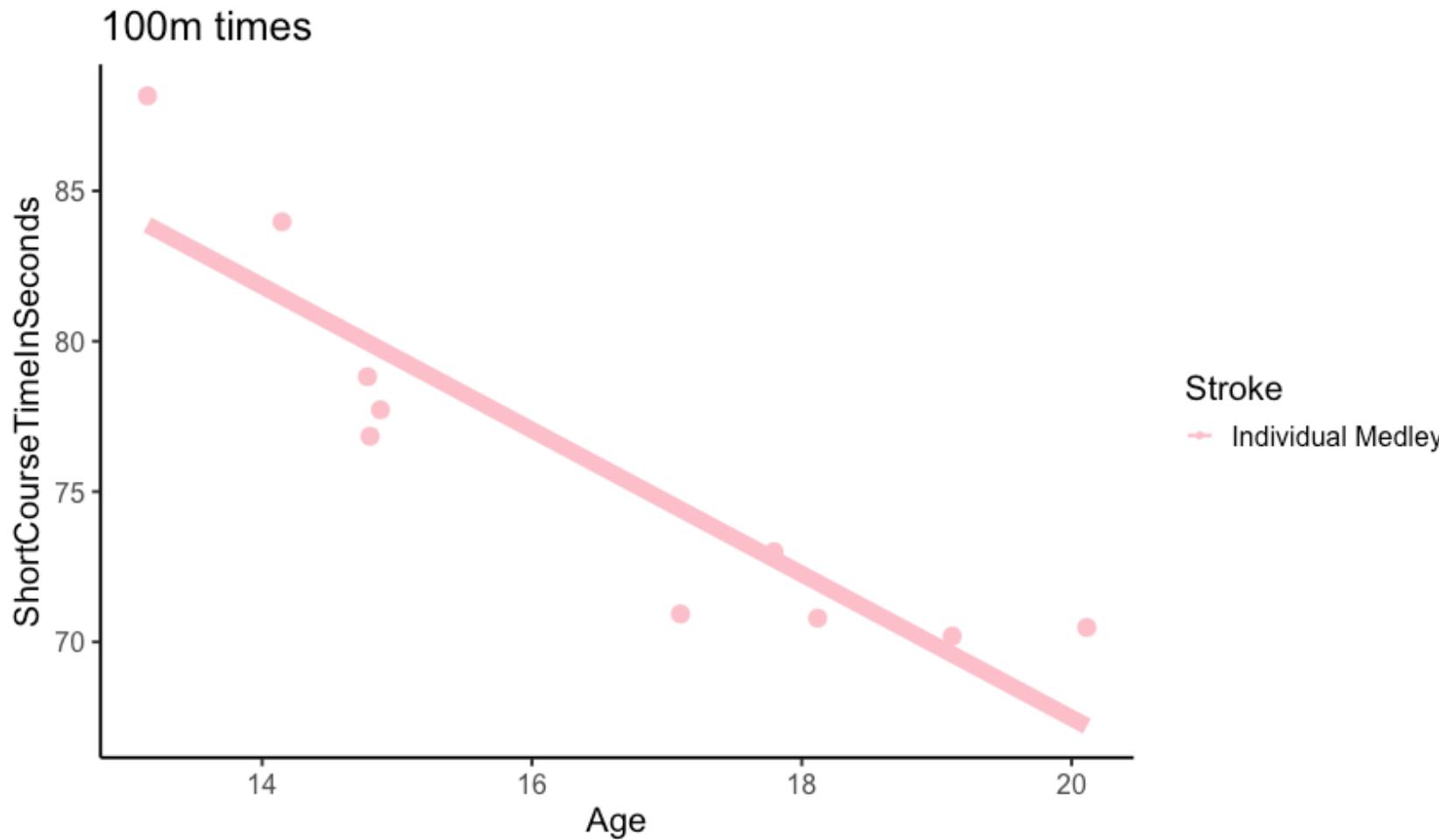
- In the case of training data that is *linearly separable* (a hyperplane can be used to perfectly separate the classes in the training data), the minimum is global but not unique
 - when $X^T \beta = 0$, also $X^T k\beta = 0$ - means that scaling β results in the same decision boundary;
 - also there are many hyperplanes that would result in the same classification.
- For logistic loss
 - the risk is minimised by taking $f(X)$ to infinity
 - when $f(X) = X^T \beta$, this means that we would take β to infinity (the scaling k that minimises the risk goes to infinity)
 - the result is that the logistic function tends towards the Heaviside step function - the activation function used in the perceptron.

Example: logistic regression

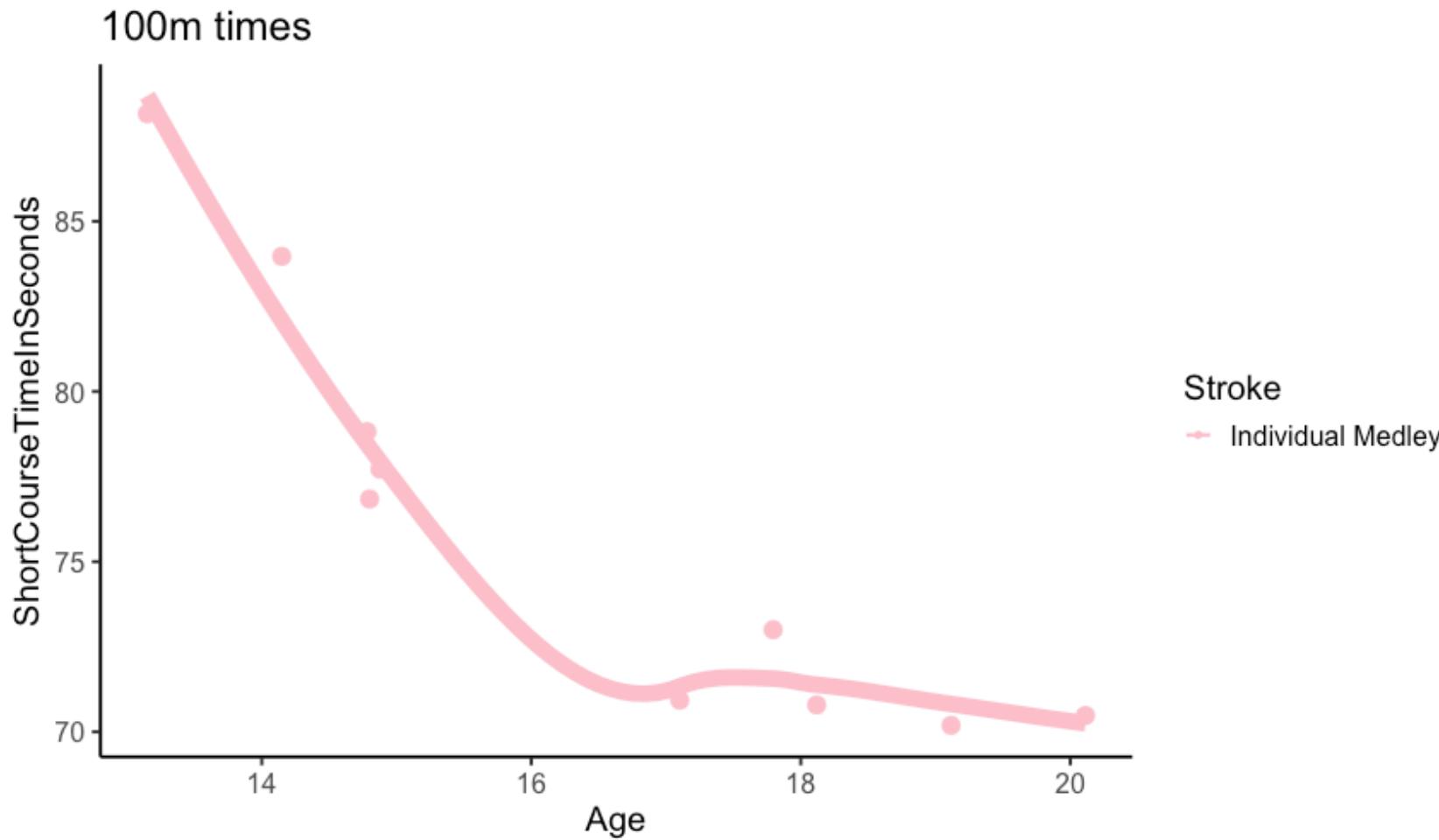
- In practice, we use regularisation to prevent this behaviour

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \frac{1}{n} \sum_{i=1}^n \log(1 + \exp(-Y_i f(X_i))) + \lambda \|\beta\|_2^2.$$

Example: non-parametric regression



Example: non-parametric regression



Example: non-parametric regression

- *Smoothing splines* are a method for non-parametric regression from \mathbb{R} to \mathbb{R} .
- We would like to fit the data with some general f , but we do not want the function to be too "wiggly"
 - for a general class of functions f will tend to be wiggly if fitted using ERM (it will closely fit the training data).
- Smoothing splines choose f using ERM with a regularisation penalty, over the space of all functions $f : \mathbb{R} \rightarrow \mathbb{R}$ - recall we call this \mathcal{F} .
- A smoothing spline minimises

$$\arg \min_{f \in \mathcal{F}} \hat{R}(f) + \lambda \int_x (f''(x))^2 dx.$$

- Note that the integral in the penalising term grows when f is more wiggly.

Example: non-parametric regression

- Remarkably
 - there is a unique function minimizing this criterion
 - this unique function takes a simple form.
- A natural cubic spline fit has the following properties
 - cubic functions in between the observed X
 - linear outside of the observed X
 - continuous and have continuous first and second derivatives at the observed X
 - minimises a particular regularised empirical risk.
- The cubic spline fit is the unique function minimising

$$\arg \min_{f \in \mathcal{F}} \hat{R}(f) + \lambda \int_x (f''(x))^2 dx.$$

Summary

- Regularisation constrains the fit of a flexible model
 - can avoid overfitting, reduces variance.
- Most common examples are ridge regression and the lasso
 - best subset selection can also be seen in this framework.
- Selecting the tuning parameter λ is left until a later lecture.

Further reading

- ISLR chapter 6.
- ESL chapter 3.
- PRML chapter 3.

Regularisation II

ST420 Lecture 10

Richard Everitt

Statistical issues: estimation and approximation error

- Leaving aside the computational challenges, what are the statistical challenges and opportunities for large n and/or p ?
- Later in the module, we will see that:
 - estimation error $R(\hat{f}) - R(\bar{f}^*)$ usually decreases as n increases, since we get closer to fitting the optimal function within the class;
 - approximation error $R(\bar{f}^*) - R(f^*)$ usually decreases as p increases, since the class of functions grows more flexible, and more able to fit the optimal function.
- However:
 - estimation error $R(\hat{f}) - R(\bar{f}^*)$ usually increases as p increases, since the model is more difficult to fit (we could compensate for this by increasing n).

Multiple linear regression

- What happens when $p + 1 = n$, and we fit one single regression/classification model?
 - not anything to do with multiple testing any more - thinking about the fit and ability to generalise of the fitted model
 - we might expect the situation to be different to the single variable case, since we are jointly fitting $p + 1$ variables.
- Consider regression with squared error loss.
- Recall the expected excess risk (lecture 4)
 - the expected prediction error of our fitted model minus the optimal prediction error
 - optimal prediction error in this case is $R(f^*) = \mathbb{E}_X [\mathbb{V}_Y [Y | X]]$
 - and we have

$$\mathbb{E}_{\mathcal{T}} [R(\hat{f}) - R(f^*)] = \underbrace{\mathbb{V}_{\mathcal{T}} [\hat{f}(X)]}_{\text{variance}} + \underbrace{\left(\mathbb{E}_{\mathcal{T}} [\hat{f}(X)] - f^*(X) \right)^2}_{\text{squared bias}}.$$

Multiple linear regression

- The model is $Y = X^T \beta + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$.
- Let \mathbf{X} be the $n \times (p + 1)$ matrix known as the *design matrix*.
- The least squares estimator is

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T Y.$$

- Suppose the true relationship between X and Y is linear and the noise $\mathbb{V}_Y[Y | X]$ is homoscedastic (does not depend on X)
 - then this solution is unbiased
 - it has minimum variance amongst all unbiased solutions (the Gauss–Markov theorem).

Multiple linear regression $p + 1 \leq n$

- In the case of $p \leq n$, the variance of $X^T \hat{\beta}$ is approximately

$$\sigma^2 \frac{p+1}{n}$$

- very important result!
- The expected excess risk is therefore approximately $\sigma^2 \frac{p+1}{n}$.
- The expected risk is therefore approximately $\sigma^2 \left(1 + \frac{p+1}{n}\right)$.
- Fixed a fixed n , think about the influence of increasing the number of predictors
 - we add σ^2/n to the expected risk for each one.
- If we do not include a predictor on which there is a dependence, we will get a bias
 - but for every predictor we include, we increase the variance (no matter whether the predictor is useful for the model or not).

Multiple linear regression $p + 1 = n$

- Suppose $n = 1$
 - then we can fit a regression that fits through this point simply using the intercept, $p = 0$.
- Suppose $n = 2$
 - then we can fit a regression that fits through both points using only one predictor, $p = 1$.
- Suppose $n = 3$
 - then we can fit a regression that fits through all three points using two predictors, $p = 2$.
- In general, if $p + 1 = n$
 - we can choose a fit that passes through all of the points.
- Will this fit be useful?
 - if there is any noise in the data, we know that the model will be fitting to the noise.

Multiple linear regression $p + 1 > n$

- Recall the least squares estimator is

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T Y.$$

- We need $\mathbf{X}^T \mathbf{X}$ to be invertible
 - however it is singular for $p + 1 > n$
 - therefore we cannot use the least squares fit.
- Idea for possible solutions
 - use dimension reduction on the predictors
 - use regularisation or variable selection to reduce the effect of, or omit, some predictors.

Aside: dimension reduction

- Recall *principal components analysis* (PCA)
 - can be used for finding lower dimensional projections of the data that explain most of the variation.
- *Partial least squares* (PLS) is an approach for performing PCA within a regression
 - the projection is chosen such that the predictors are weighted by the strength of their effect on the response.
- However, PLS has been found not to work well on problems where $p + 1 > n$
 - "a large number of noisy features can contaminate the predictions" (ESL p680).
- A technique called "supervised principal components" can be successful
 - but the key to this is that it involves an initial step that filters out noisy features
 - PLS can also be modified to do this.

Regularisation revisited

- Consider the case of $p + 1 > n$, where $\mathbf{X}^T \mathbf{X}$ is singular.
- We cannot use the least squares estimator

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T Y.$$

- Idea: make the $\mathbf{X}^T \mathbf{X}$ term non-singular using

$$(\mathbf{X}^T \mathbf{X} + \lambda I_{p+1})^{-1} \mathbf{X}^T Y,$$

for some λ

- this is the ridge estimator, that results from finding

$$\arg \min_{\beta} \left[(Y - \mathbf{X}\beta)^T (Y - \mathbf{X}\beta) + \lambda \beta^T \beta \right].$$

Regularisation revisited

- Behaviour of ridge
 - using such a regularisation would shrink the coefficients
 - introduces a bias, but lowers the variance
 - for the case $p + 1 \leq n$ (when the least squares estimator exists), may have a lower error than least squares
 - a larger λ is needed the larger p is.

Ridge, lasso, subset selection

- Recall the previous lecture.
- Ridge (shrinks coefficients):

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \hat{R}(f) \quad \text{subject to } \left(\sum_{j=1}^p (\beta^j)^2 \right) \leq s(\lambda).$$

- Lasso (soft-thresholding):

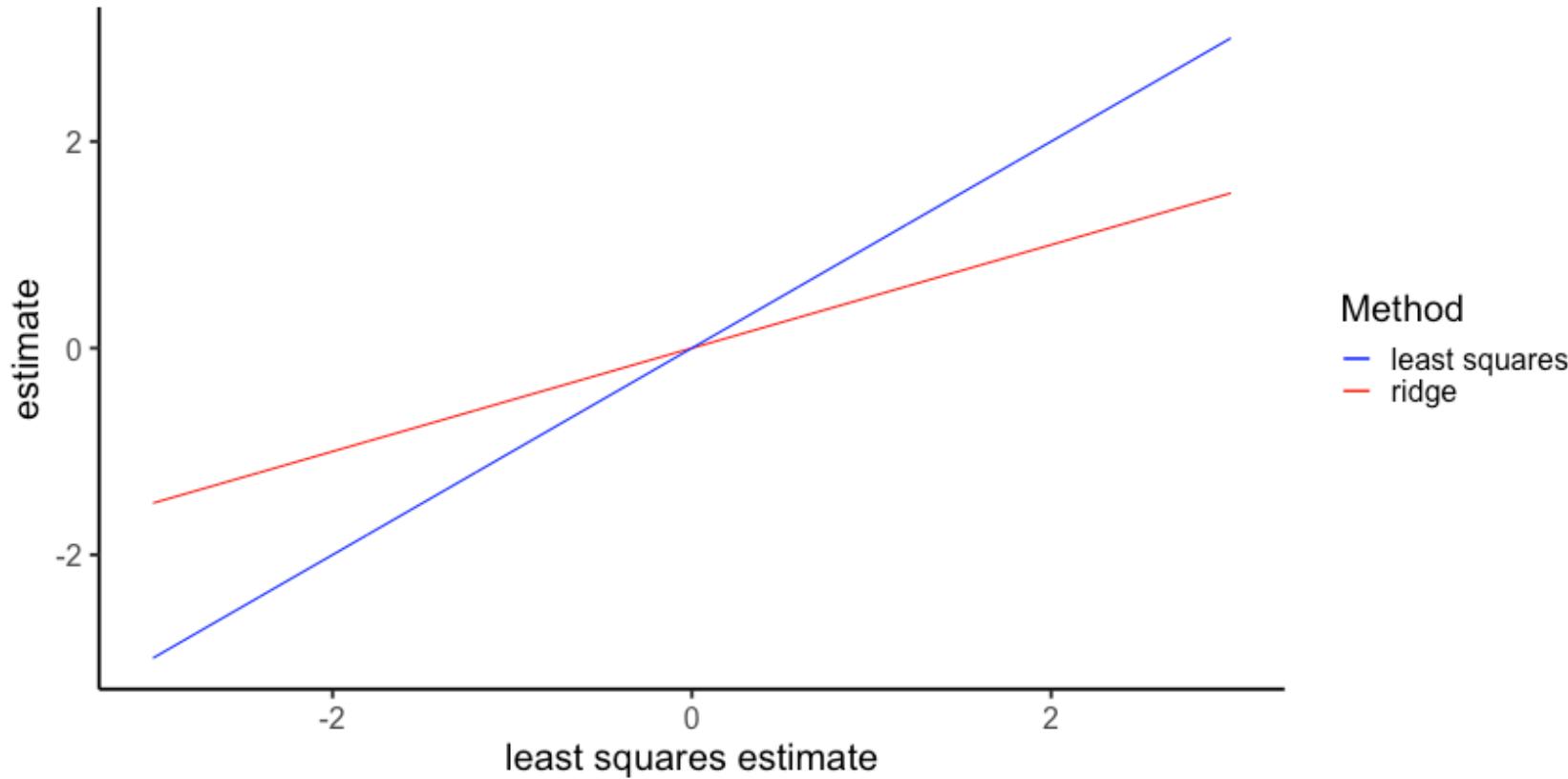
$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \hat{R}(f) \quad \text{subject to } \left(\sum_{j=1}^p |\beta^j| \right) \leq s(\lambda).$$

- Best subset selection (hard-thresholding):

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \hat{R}(f) \quad \text{subject to } \sum_{j=1}^p I(\beta^j \neq 0) \leq s.$$

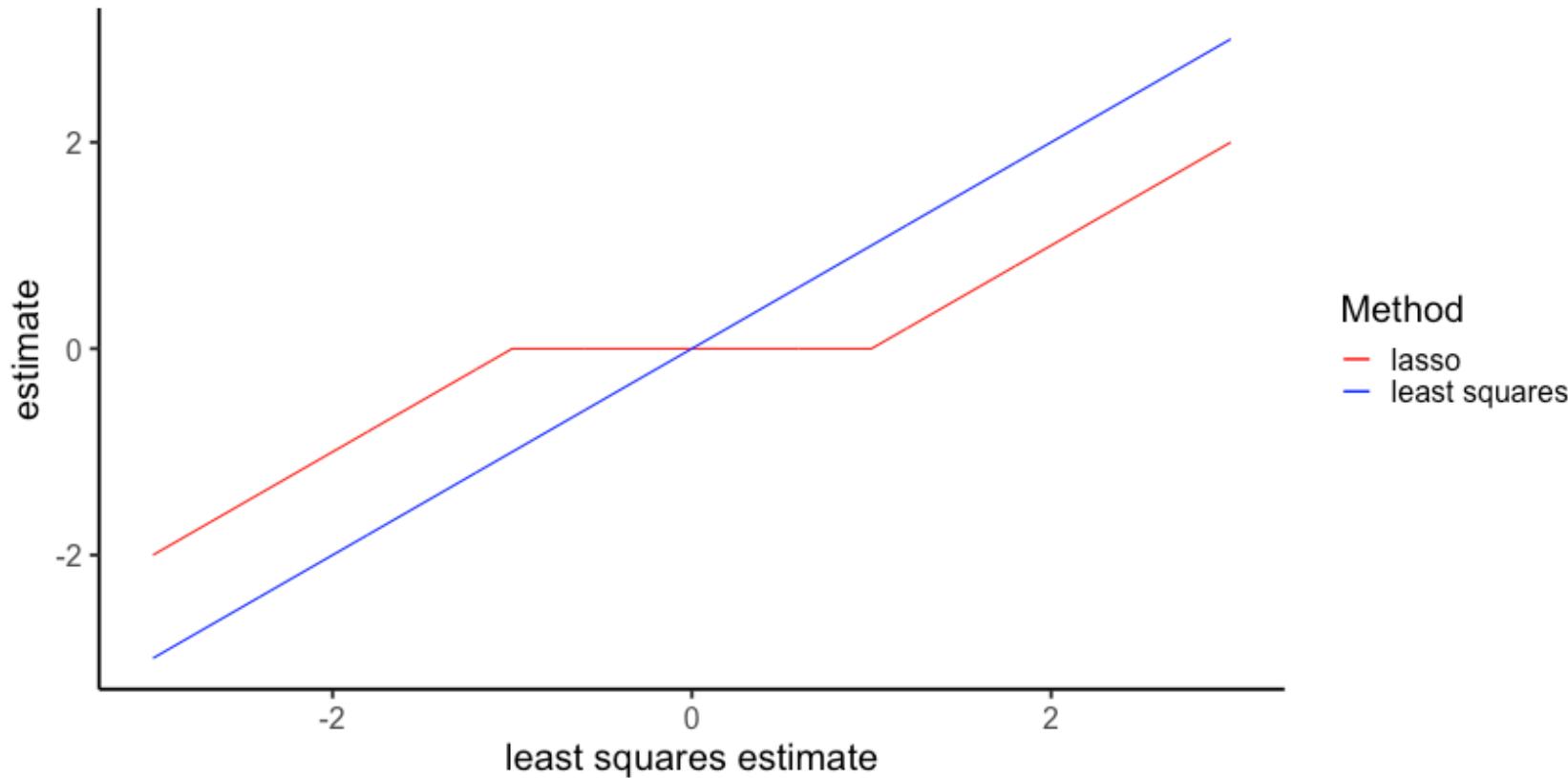
Ridge regression

$$\hat{\beta}^j / (1 + \lambda)$$



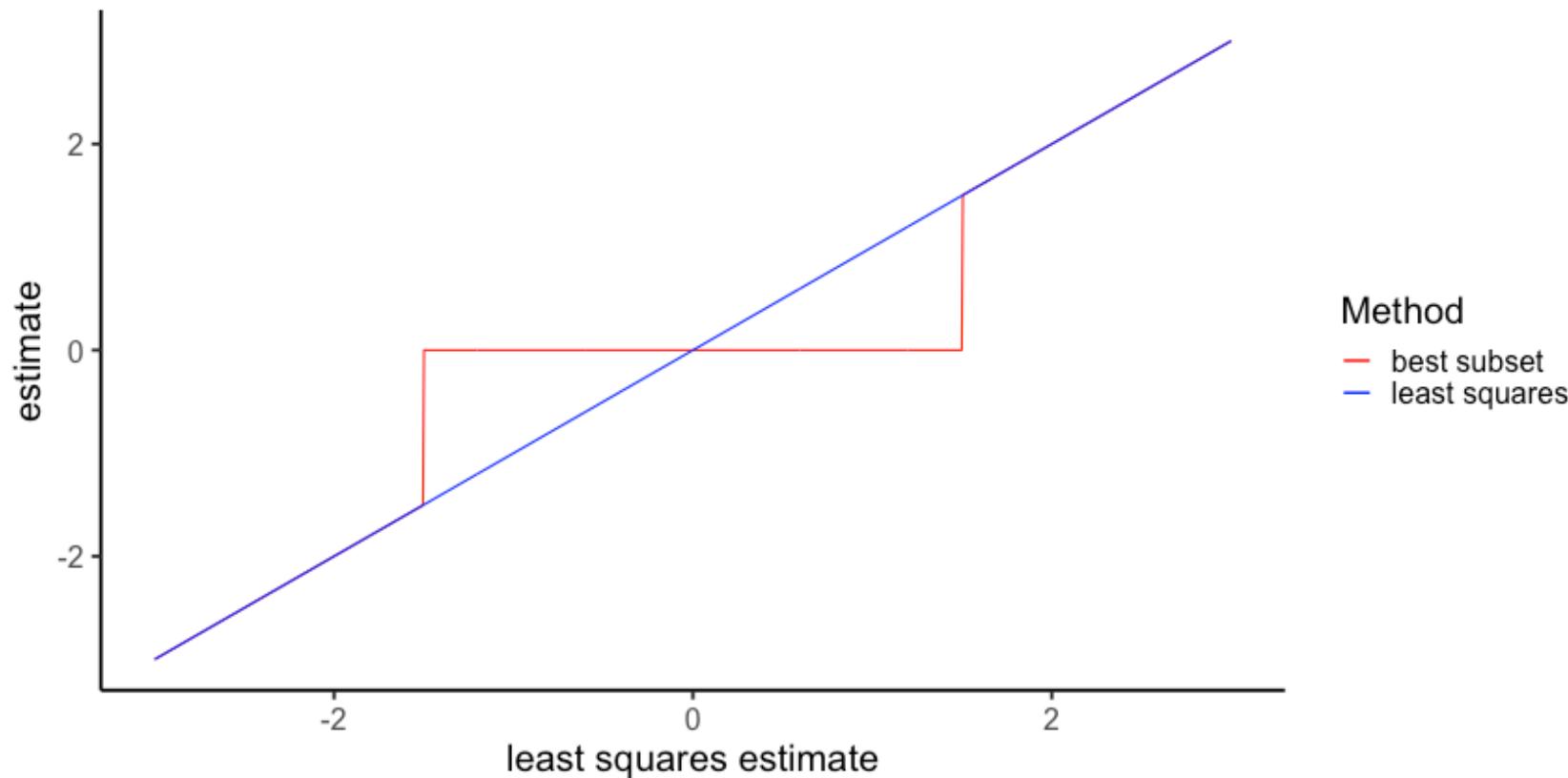
Lasso

$$\text{sign}(\hat{\beta}^j) \left(|\hat{\beta}^j| - \lambda \right)_+$$



Best subset selection

$$\hat{\beta}^j I\left(\left|\hat{\beta}^j\right| \geq \left|\hat{\beta}^{s\text{th largest}}\right|\right)$$



Variable selection

- Ridge regression simultaneously shrinks all coefficients
 - does not completely address the problem posed by large p , since we shrink the predictors containing the "signal" as well as the noise
 - we might prefer to perform variable selection in this case.
- Both the lasso and best subset selection perform some form of variable selection
 - recall that this was the idea that helped supervised principal components.
- Suppose that not all of the predictors affect the response, and there is a linear relationship between the response and all of the other predictors
 - a good variable selection technique may be able to reduce the variance (through omitting predictors), without introducing bias (by only omitting the appropriate predictors).

Lasso for large p problems

- The lasso performs variable selection, and has computational advantages over best subset selection (previous lecture).
- However, when faced with dependent predictors that all have a strong relationship with the response, the lasso does not know which to choose
 - occurs in genomics, where different sites in the genome can be dependent.
- In this case, we prefer the behaviour of ridge regression, which will reduce variance without losing much bias by simultaneously shrinking all of the dependent predictors.
- Recall that ridge and lasso are special cases of the general case of minimising

$$\arg \min_{f \in \bar{\mathcal{F}}} \hat{R}(f) + \lambda \|\beta\|_r^r$$

for $r \geq 0$.

- Can we use $1 < r < 2$ and obtain the useful properties of both?

Elastic net

- The proposal of using $1 < r < 2$ does not work in the desired fashion
 - the method loses the variable selection property of some predictors becoming precisely zero.
- An alternative proposal is to use the *elastic net* penalty, resulting in, for the linear regression case,

$$\arg \min_{\beta} (Y - \mathbf{X}\beta)^T (Y - \mathbf{X}\beta) + \lambda \sum_{j=1}^p [\alpha (\beta^j)^2 + (1 - \alpha) |\beta^j|]$$

for $0 \leq \alpha \leq 1$.

Elastic net

- Can still perform variable selection
 - since the region to which the parameters are constrained still has "sharp corners" (see figure in previous lecture)
 - can also allow more than n variables to be selected, whereas the lasso allows at most n .
- Also performs shrinkage in the manner of ridge regression (ESL p661).

The Dantzig selector

- Candes and Tao (2007) introduce a method that has impressive theoretical properties for the case of $p + 1 > n$
 - "Even though n may be much smaller than p , our estimator achieves a loss within a logarithmic factor of the ideal mean squared error one would achieve with an oracle which would supply perfect information about which coordinates are nonzero, and which were above the noise level."
- This sounds like it is achieving exactly what we need!
 - we can use the Dantzig selector to do very effective variable selection, which reduces the variance but does not introduce much bias.

The Dantzig selector

- The lasso uses

$$\arg \min_{\beta} \left[(Y - \mathbf{X}\beta)^T (Y - \mathbf{X}\beta) \right] \quad \text{subject to } \sum_{j=1}^p |\beta^j| \leq s(\lambda).$$

- The Dantzig selector uses

$$\arg \min_{\beta} \|\mathbf{X}^T (Y - \mathbf{X}\beta)\|_{l_\infty} \quad \text{subject to } \sum_{j=1}^p |\beta^j| \leq s(\lambda),$$

where $\|\cdot\|_{l_\infty}$ is the maximum absolute value of the components of the vector.

The Dantzig selector

- In the discussion of Candes and Tao (2007), Efron, Hastie and Tibshirani compare empirically the Dantzig selection and the lasso.
- The lasso outperforms (or is the equal of) the Dantzig selector in all cases in the discussion.
- Similar theoretical guarantees now exist for the lasso.
- The Dantzig selector is used relatively rarely compared to the lasso.

Summary

- We have discussed the case of linear regression when p is large
 - similar reasoning applies to linear classification.
- Next lectures:
 - support vector classifiers and support vector machines.

Further reading

- ISLR chapter 6.
- ESL chapters 3 and 18.
- PRML chapter 3.

Support vector classifiers

ST420 Lecture 11

Richard Everitt

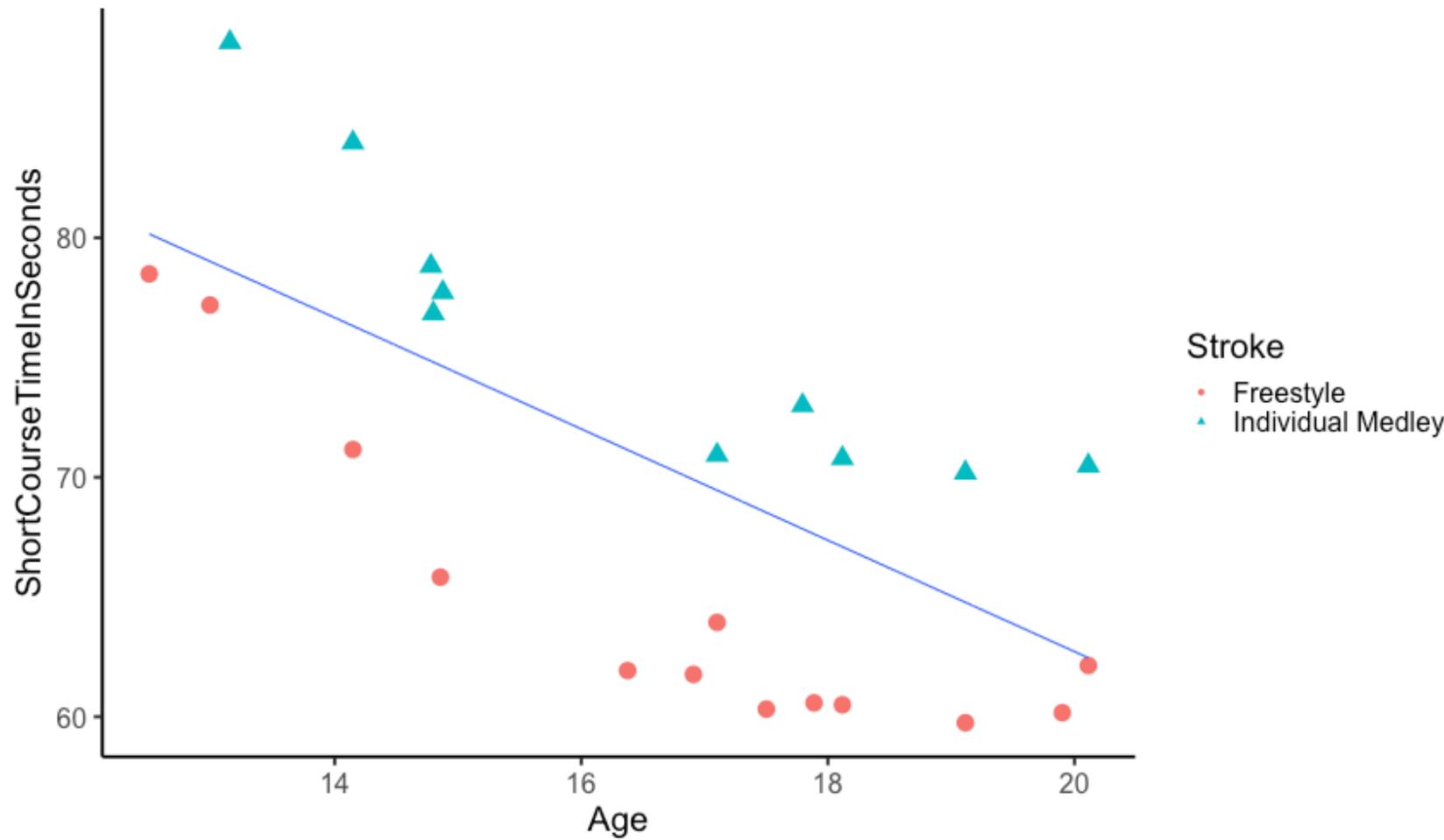
Recap

- We have been introducing different methods for classification.
- We wanted to use a flexible set of basis functions, without overfitting
 - we looked at neural networks to learn basis functions adaptively
 - we introduced regularisation for avoiding overfitting.
- Are there other approaches?

Support vector machines

- *Support vector machines* (SVMs) were probably the most "trendy" classification technique until the renewed interest in neural networks.
- Based on using a linear approach to separating classes, with two innovations:
 - the idea of using a "margin" in order to define a unique separating hyperplane (dates back to the work of Vapnik and Chervonenkis in the 1960s);
 - the use of kernels to implicitly use a complicated set of basis functions on which to base the classification (Vapnik 1992).
- We will spend a lecture each on these ideas.

LDA classification



Perceptron

- Let Y be a discrete variable that can take values in $\{+1, -1\}$
 - classification with 2 classes.
- We write $X^T \beta + \beta^0$ in place of $X^T \beta$.
- Use

$$P_{Y|X}(Y = 1 \mid X, \beta) = \sigma(X^T \beta + \beta^0)$$

where

$$\sigma(a) = \begin{cases} 1 & \text{if } a \geq 0 \\ 0 & \text{if } a < 0 \end{cases}.$$

- Classes separated by a hyperplane
 - this hyperplane is not unique if the classes are linearly separable in the training data...
 - but different hyperplanes will result in different risks.

Maximal margin classifier: idea

- Consider the linearly separable case.
- Define the *margin* to be the smallest distance between the decision boundary and any of the training samples.
- The idea of the *maximal margin classifier* is to choose the hyperplane that maximises the margin.
- Why?
 - originally motivated by statistical learning theory (later in the module)
 - intuitively, we might expect this idea to give us the best chance of generalising.

Maximal margin classifier: definition

- Recall
 - the perpendicular distance of a point X from a hyperplane $X^T \beta + \beta^0 = 0$ is given by $|X^T \beta + \beta^0| / \|\beta\|_2$.
- Thus the distance of a correctly classified point X_i to the decision boundary is

$$\frac{Y_i(X_i^T \beta + \beta^0)}{\|\beta\|_2}.$$

- The margin is the distance of the boundary to the closest point from the data set
 - we wish to optimize the parameters β and β^0 to maximize this distance
 - thus the maximum margin solution is given by

$$\arg \max_{\beta, \beta^0} \left\{ \frac{1}{\|\beta\|_2} \min_i (Y_i(X_i^T \beta + \beta^0)) \right\}.$$

Maximal margin classifier: definition

- Recall that we can rescale β and β^0 in $X^T\beta + \beta^0 = 0$ by any constant, and still obtain the same decision boundary.
- Thus we can set $Y_i(X_i^T\beta + \beta^0) = 1$ for the point that is closest to the boundary
- Therefore the problem reduces to finding

$$\arg \max_{\beta, \beta^0} \left\{ \frac{1}{\|\beta\|_2} \right\} \quad \text{subject to} \quad Y_i(X_i^T\beta + \beta^0) \geq 1 \quad \forall i = 1 : N.$$

- Equivalently

$$\arg \min_{\beta, \beta^0} \left\{ \frac{1}{2} \|\beta\|_2^2 \right\} \quad \text{subject to} \quad Y_i(X_i^T\beta + \beta^0) \geq 1 \quad \forall i = 1 : N,$$

where the factor of $1/2$ was added here for later convenience and using that the β that maximises $\|\beta\|_2$ also maximises $\|\beta\|_2^2$.

Maximal margin classifier: properties

- By definition $Y_i(X_i^T \beta + \beta^0) = 1$ will hold for at least one point
 - for these cases the constraint is said to be *active*.
- For all other points, we have $Y_i(X_i^T \beta + \beta^0) > 1$
 - for these cases the constraint is said to be *inactive*.
- Equivalent to the following optimisation of empirical risk with penalisation
 - use loss function of

$$L(Y, f(X)) = \begin{cases} 0 & \text{if } Y(X^T \beta + \beta^0) \geq 1 \\ \infty & \text{if } Y(X^T \beta + \beta^0) < 1 \end{cases}.$$

Maximal margin classifier: properties

- Fitted using

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \underbrace{\frac{1}{n} \sum_{i=1}^n L(Y_i, f(X_i))}_{\text{loss due to misclassifications}} + \underbrace{\frac{1}{2} \|\beta\|_2^2}_{\text{regularisation}} .$$

- The regularisation is in the form of maximising the margin.
- When we perform the optimisation, we find that the solution is determined completely by points that lie on the margin
 - these points are known as the *support vectors*.

Support vector classifier: idea

- By construction, the solution does not exist outside of the linearly separable case.
- We now introduce some flexibility, and allow overlapping classes
 - rather than using ∞ to penalise cases of incorrect classification, we allow misclassifications ("margin errors") to be possible.
- The result is known as the *support vector* or *soft margin* classifier.

Support vector classifier: idea

- The idea is to trade off margin errors with maximising the margin
 - use a different loss function to allow some margin errors
 - keep the same regularisation term to maximise the margin.
- We might try the loss function

$$L(Y, f(X)) = I(Y(X^T \beta + \beta^0) < 0)$$

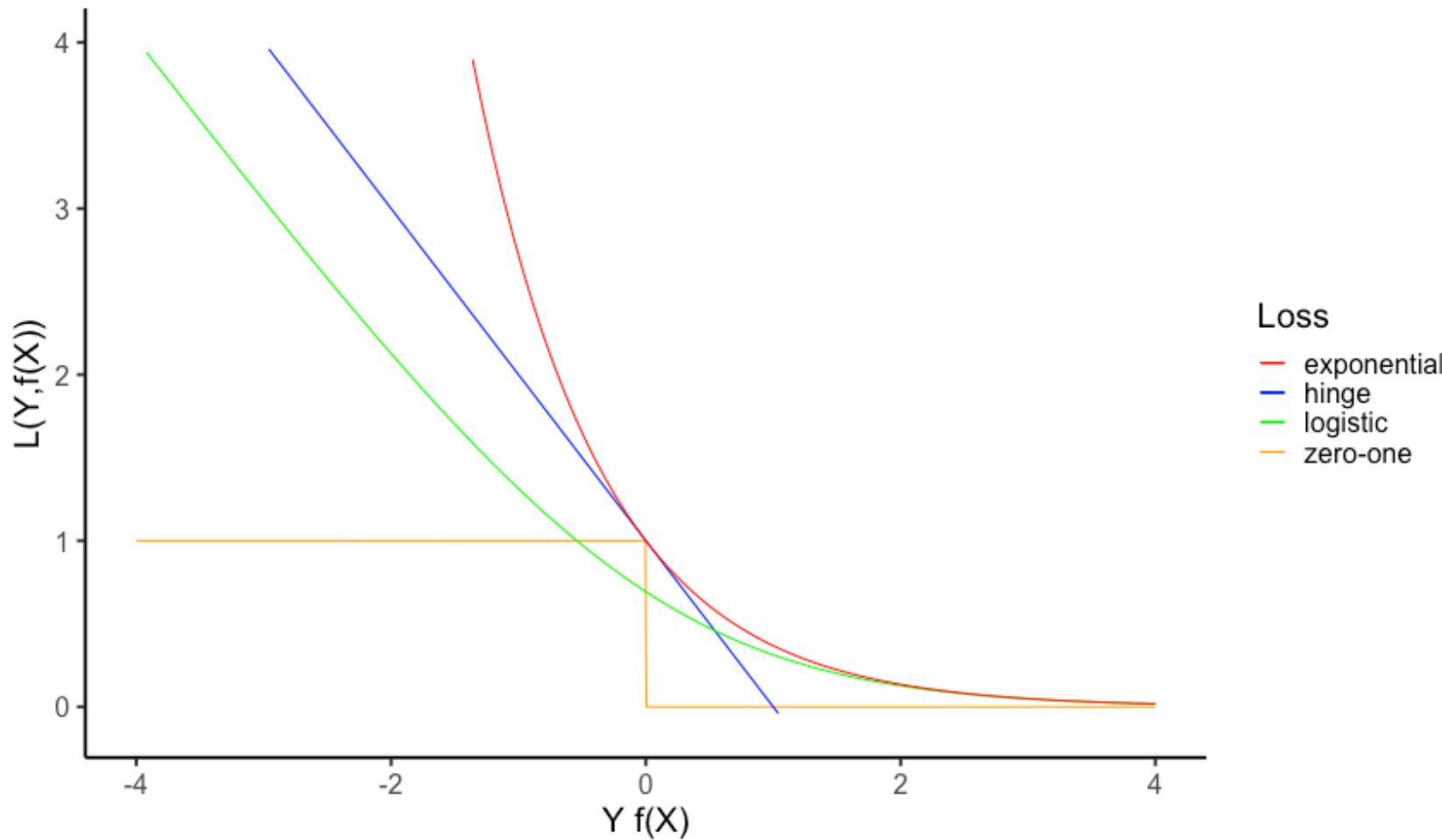
- recall that this is zero-one loss
- penalises misclassifications.

- Or

$$L(Y, f(X)) = I(Y(X^T \beta + \beta^0) < 1)$$

- penalises margin errors.
- Both result in an awkward optimisation problem.

Sketch of loss functions for classification



Support vector classifier: definition

- Instead uses the *hinge* loss function

$$(Y, f(X)) = \max\{1 - Yg(X), 0\}$$

- a convex upper bound to zero-one loss.

- Fit using

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \frac{1}{n} \sum_{i=1}^n \max\{1 - Y_i g(X_i), 0\} + \lambda \|\beta\|_2^2.$$

- This is more commonly seen in the form of

$$\arg \min_{\beta, \beta^0} \left\{ C \sum_{i=1}^n \xi_i + \frac{1}{2} \|\beta\|_2^2 \right\} \quad \text{subject to} \quad \xi_i \geq 0, \quad Y_i (X_i^T \beta + \beta^0) \geq 1 - \xi_i \quad \forall i = 1 : n$$

Interpreting the ξ_i

- The ξ_i variables allow some "slack" in the constraints.
- $\xi_i = 0$ corresponds to the previous case, of points being correctly classified.
- $0 < \xi_i \leq 1$ for the points that lie inside the margin, but on the correct side of the decision boundary.
- $\xi_i > 1$ for the points that are on the wrong side of the decision boundary.
- $\sum_{i=1}^n \xi_i$ is an upper bound on the number of misclassified points.
- The constant C controls the trade-off between minimizing training errors and controlling model complexity (as λ does in the usual regularisation equation).

Comparing to logistic regression

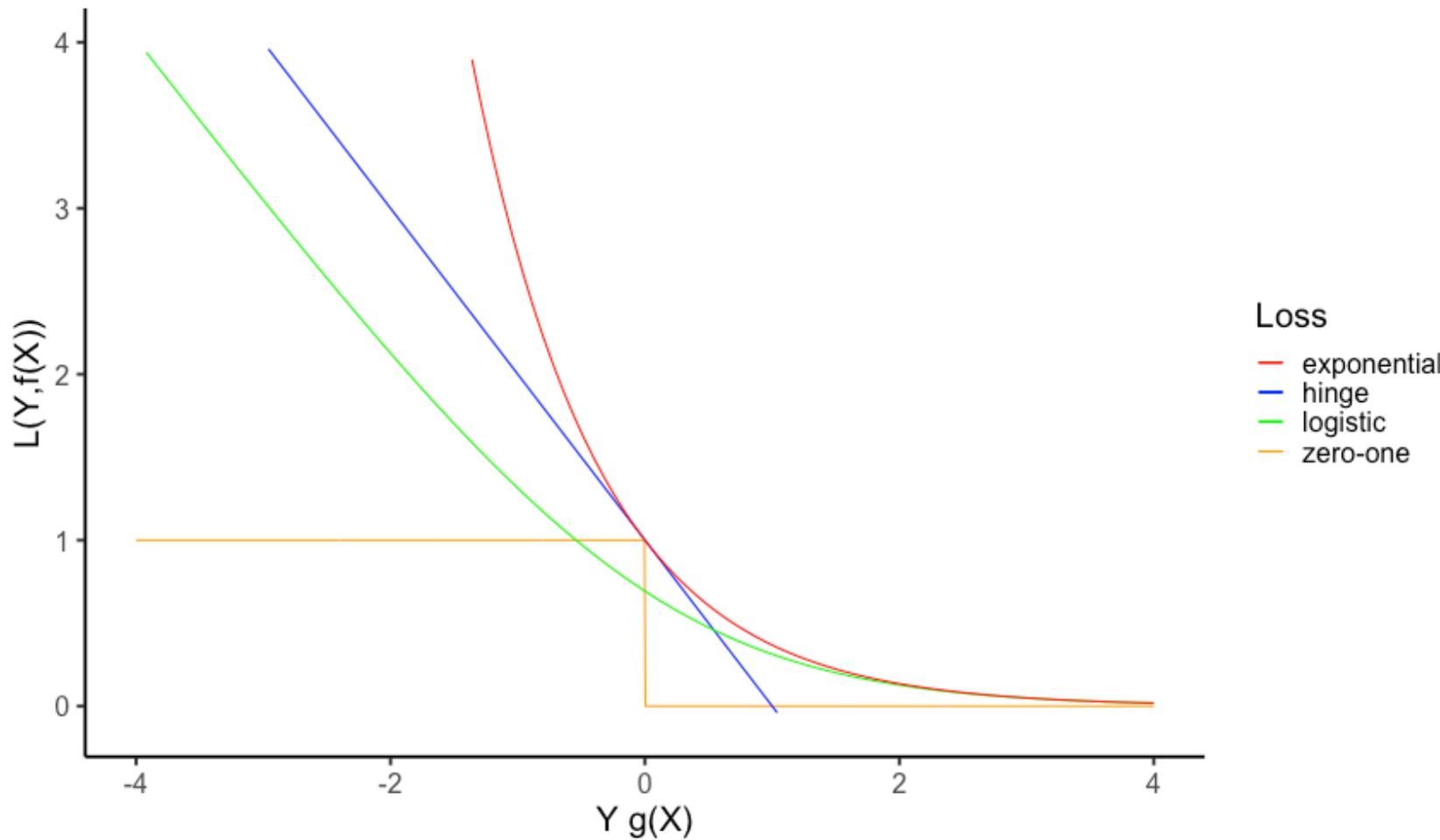
- Logistic regression
 - used logistic loss function of $L(Y, f(X)) = \log(1 + \exp(-Yg(X)))$
 - fitted using

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \frac{1}{n} \sum_{i=1}^n \log(1 + \exp(-Y_i g(X_i))) + \lambda \|\beta\|_2^2.$$

- Support vector classifier
 - uses *hinge* loss function $L(Y, f(X)) = \max\{1 - Yg(X), 0\}$
 - fitted using

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \frac{1}{n} \sum_{i=1}^n \max\{1 - Y_i g(X_i), 0\} + \lambda \|\beta\|_2^2.$$

Sketch of loss functions for classification



Discussion

- Hinge loss is exactly zero for observations on the correct side of the margin
 - gives a sparse solution.
- Logistic loss not exactly zero anywhere
 - but it is very small for observations that are far from the decision boundary.
- Often give similar results
 - when the classes are well separated, support vector classifiers tend to perform better than logistic regression;
 - for overlapping classes, logistic regression can perform better.

Support vector classifier: properties

- What is the function f^* that minimises the risk under hinge loss?
- It is

$$f^*(X) = \text{sign}\left(P_{Y|X}(Y = +1 \mid X) - 1/2\right).$$

- When we perform the optimisation, we find that points that lie outside of the margin do not determine the parameters of the hyperplane
 - points that lie on or inside the margin are known as the *support vectors*.

Summary

- Support vector machines use two innovations:
 - the idea of using a "margin" in order to define a unique separating hyperplane;
 - the use of kernels to implicitly use a complicated set of basis functions on which to base the classification.
- We have examined the first of these
 - this was motivated by learning theory
 - we have seen that it corresponds to using ERM with penalisation, when using the hinge loss function
 - this is not so different from logistic regression.
- The second idea allows non-linear decision boundaries
 - we will examine it next time.

Further reading

- ISLR chapter 9.
- ESL chapter 12.
- PRML chapter 7.

Kernel methods

ST420 Lecture 12

Richard Everitt

Recap

- We have been looking at different methods for classification.
- One strand of thought has been about moving beyond linear approaches by using a flexible set of features.
- Last lecture we looked at the maximal marginal classifier and the support vector classifier
 - we saw both models in the loss/risk framework.
- This lecture we turn the support vector classifier into a support vector machine.

Support vector classifier

- Let $g(X) = X^T \beta + \beta^0$, where this equation defines the decision boundary.
- Fit using

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \frac{1}{n} \sum_{i=1}^n \max\{1 - Y_i g(X_i), 0\} + \lambda \|\beta\|_2^2.$$

- Let

$$\phi(X)^T = (\phi^1(X), \phi^2(X), \dots, \phi^m(X)),$$

where in contrast to previous lectures we are making explicit the use of features.

- How should we choose the ϕ^j ?

Returning to ridge regression

- In *ridge regression* we use

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \hat{R}(f) + \lambda \|\beta\|_2^2,$$

where $\|\beta\|_2^2 = \sum_{j=1}^m (\beta^j)^2$ and $\lambda \geq 0$.

- When $f(X) = \phi(X)^T \beta$, the estimate can be found analytically. Let Φ be the design matrix with dimension $n \times m$.
- Then the ridge regression estimate of the parameters is given by

$$\hat{\beta} = (\Phi^T \Phi + \lambda \mathbf{1}_m)^{-1} \Phi Y.$$

- We are going to rewrite the solution, under squared error loss, in a different form.

Ridge regression: dual form

- First, we make superficial changes to the objective function

$$\hat{f} = \arg \min_{\beta} \frac{1}{2} \sum_{i=1}^n (\beta^T \phi(X_i) - Y_i)^2 + \frac{\lambda}{2} \beta^T \beta.$$

- Now, differentiating with respect to β , and solving for β , we find

$$\begin{aligned}\beta &= -\frac{1}{\lambda} \sum_{i=1}^n (\beta^T \phi(X_i) - Y_i) \phi(X_i) \\ &= \sum_{i=1}^n a_i \phi(X_i) \\ &= \Phi^T a,\end{aligned}$$

where Φ is the design matrix, and $a = (a_1, \dots, a_n)^T$ with $a_i = -\frac{1}{\lambda} (\beta^T \phi(X_i) - Y_i)$.

Ridge regression: dual form

- The plan is to reformulate ridge with least squares in terms of a , rather than β .
- We obtain the *dual representation*

$$\hat{f} = \arg \min_a \frac{1}{2} a^T \Phi \Phi^T \Phi \Phi^T a - a^T \Phi \Phi^T Y + \frac{1}{2} Y^T Y + \frac{\lambda}{2} a^T \Phi \Phi^T a.$$

- The product $K = \Phi \Phi^T$ is known as the *Gram matrix*, whose elements are

$$K_{i_1, i_2} = \phi(X_{i_1})^T \phi(X_{i_2}) = k(X_{i_1}, X_{i_2}).$$

- $k(X_{i_1}, X_{i_2})$ is known as the *kernel function*.

Ridge regression: dual form

- Setting the gradient with respect to a to zero and solving, we obtain

$$\hat{a} = (K + \lambda \mathbf{1}_n)^{-1} Y.$$

- Compare with

$$\hat{\beta} = (\Phi^T \Phi + \lambda \mathbf{1}_m)^{-1} \Phi^T Y.$$

- It looks as though, when $n > m$, we have made things more complicated...
 - previously we needed to invert an $m \times m$ matrix
 - in the dual form, we need to invert an $n \times n$ matrix.
- The advantage is that the solution is in terms of the kernel k , rather than explicitly in terms of the ϕ .

Using kernels

- This is not the only case where we can write our solution in terms of k
 - we can do the same in any other situation where the solution can be written in terms of the inner product $\phi(X_{i_1})^T \phi(X_{i_2})$.
- Support vector classifiers are one case
 - the solution is not analytic, but there exists a dual form of the constrained optimisation that is written solely in terms of $k(X_{i_1}, X_{i_2})$
 - this is known as a *support vector machine*.
- The same is true of principal component analysis (PCA)
 - the resultant method is called *kernel PCA*.

What is the point of all of this?

- What is the advantage of not using ϕ explicitly?
- The reason is that we can choose (and then use) k without ever making reference to ϕ .
- Choices of k will result in the ϕ being defined implicitly
 - it turns out that some choices of k implicitly use vectors of ϕ that are of very high or even infinite dimension!
- Simply through a choice of k , which we need to evaluate at all pairs of data points, we can use a very flexible set of features for our regression or classification method.
- This seems a bit abstract, until we see an example.

Examples of kernels

- First example: $k(X_1, X_2) = (X_1^T X_2)^2$.
- Let's take a look at what features we would be using if this was our kernel, in the case of $X = (X^{(1)}, X^{(2)})$

$$\begin{aligned} k(X_1, X_2) &= (X_1^T X_2)^2 \\ &= \left(X_1^{(1)} X_2^{(1)} + X_1^{(2)} X_2^{(2)} \right)^2 \\ &= \left(X_1^{(1)} \right)^2 \left(X_2^{(1)} \right)^2 + 2X_1^{(1)} X_2^{(1)} X_1^{(2)} X_2^{(2)} + \left(X_1^{(2)} \right)^2 \left(X_2^{(2)} \right)^2 \\ &= \left(\left(X_1^{(1)} \right)^2, \sqrt{2}X_1^{(1)} X_2^{(2)}, \left(X_1^{(2)} \right)^2 \right) \left(\left(X_2^{(1)} \right)^2, \sqrt{2}X_1^{(1)} X_2^{(2)}, \left(X_2^{(2)} \right)^2 \right)^T. \end{aligned}$$

- We obtain

$$\phi(X) = \left(\left(X^{(1)} \right)^2, \sqrt{2}X^{(1)} X^{(2)}, \left(X^{(2)} \right)^2 \right)^T.$$

Examples of kernels

- In the first example, we ended up with all terms of degree 2.
- What if we also want the lower order terms?
- Second example: $k(X_1, X_2) = (X_1^T X_2 + c)^2$ for some $c > 0$.
- This will also give us constant and linear terms.

Examples of kernels

- How do we get higher order terms?
- Third example: $k(X_1, X_2) = (X_1^T X_2 + c)^m$ for some $c > 0$.
- This will give us all terms up to degree m .

Questions

- How much freedom do we have in our choice of k ?
- If there are rules (mathematical necessities) that govern possible choices, are there easy ways of checking that we satisfy them?
- What was that you were saying about (implicitly) using an infinite number of ϕ ?

Setup

- We start with fundamental notation:
 - \mathcal{X} is the domain of the predictors
 - $f : \mathcal{X} \rightarrow \mathbb{R}$ is a function that we will use for prediction (where we have taken our usual \mathcal{Y} to be \mathbb{R})
 - \mathcal{F} is a space of functions from \mathcal{X} to \mathbb{R} .
- We are going to define a function f via using a vector of features ϕ
 - the function will be a linear combination of the feature vector, in the case of a finite and infinite number of features respectively

$$f(x) = \sum_{j=1}^{\infty} a^j \phi^j(x) \quad f(x) = \sum_{j=1}^m a^j \phi^j(x)$$

- this means that each function $f : \mathcal{X} \rightarrow \mathbb{R}, f \in \mathcal{F}$ can be represented by the a^j parameters.

Plan

- Define what a Hilbert space is
 - will be needed in what follows.
- Define what a kernel is
 - it is defined to be a function that can be used as in our motivation above.
- Define what a reproducing kernel Hilbert space is
 - constructing functions using kernels
 - has the nice property that we will end up being able to implicitly use an infinite feature space.

Inner product space

- Let \mathcal{F} be a vector space over \mathbb{R} .
- \mathcal{F} is an *inner product space* if an inner product is defined on it.
- An inner product on \mathcal{F} is a function $\langle \cdot, \cdot \rangle_{\mathcal{F}} : \mathcal{F} \times \mathcal{F} \rightarrow \mathbb{R}$ that satisfies the following properties for all $f, g, h \in \mathcal{F}$ and $c \in \mathbb{R}$.
 1. Symmetry. $\langle f, g \rangle = \langle g, f \rangle$.
 2. Linearity. $\langle cf, g \rangle = c\langle f, g \rangle$ and $\langle f + g, h \rangle = \langle f, h \rangle + \langle g, h \rangle$.
 3. Positive definite. $\langle f, f \rangle > 0$ for $f \in \mathcal{F} \setminus \{0\}$.
- An inner product provides us with a means for geometrical constructions such as angles, orthogonality, length.
- "Length" can be given by the norm $\|f\| = \sqrt{\langle f, f \rangle}$.

Hilbert space

- A *Hilbert space* is an inner product space that is complete:
 - any Cauchy sequence converges with respect to the norm $\|\cdot\|$ to an element in the space.
- In what follows, as suggested by the notation, our space of functions \mathcal{F} will be a Hilbert space, containing functions from \mathcal{X} to \mathbb{R} .
- Recall that \mathcal{X} is a non-empty domain on which the functions are defined.
- ϕ maps values of $x \in \mathcal{X}$ to features: ϕ defines our *feature map*.

Kernel

- $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a *kernel* if there exists a Hilbert "feature" space \mathcal{F} and a "feature" map $\phi : \mathcal{X} \rightarrow \mathcal{F}$ such that for all $x, z \in \mathcal{X}$,
- $$k(x, z) = \langle \phi(x), \phi(z) \rangle_{\mathcal{F}}.$$
- We are simply saying that the kernel corresponds to the inner product of some feature vectors in some function space
 - recall that we were going to replace with kernels with inner products that cropped up in algorithms.
 - One can show that a kernel is always positive definite
 - equivalent to the Gram matrix $K(x_{i_1}, x_{i_2})$ being positive definite for all possible choices of the set $\{x_i\}_{i=1}^n$.

Reproducing kernel Hilbert space (RKHS)

- First we describe the construction we want to achieve, then we will make the definition that gives this construction.
- We will use a space in which the evaluation of a function is given by linearly combining evaluations of features - in the case of an infinite and finite number of features respectively

$$f(x) = \sum_{j=1}^{\infty} a^j \phi^j(x) \quad f(x) = \sum_{j=1}^m a^j \phi^j(x)$$

- this means that each function $f \in \mathcal{F}$ can be represented by the a^j parameters
- the parameter vector (a^1, a^2, \dots) defines the function $f(\cdot)$, which we evaluate at x by using the linear combination.
- The surprising/difficult part is that the feature map $\phi(x)$ is also a function in \mathcal{F} .

Reproducing kernel Hilbert space (RKHS)

- What do we mean by the feature map $\phi(x)$ being a function in \mathcal{F} ?
 - we already know that ϕ is a function that maps $x \in \mathcal{X}$ to the feature space
 - but $\phi(x)$ also defines the parameters a^j of a function in \mathcal{F} .
- How?
 - For some "fixed" $z \in \mathcal{X}$, take $a^j(z) = \phi^j(z)$.
 - z is acting as an index for the function.
 - This vector $(a^1(z), a^2(z), \dots)$ defines a function $k(\cdot, z) : \mathcal{X} \rightarrow \mathbb{R}$.
 - We can evaluate this function at x using, in the infinite and finite forms respectively,

$$k(x, z) = \sum_{j=1}^{\infty} a^j(z) \phi^j(x) \quad k(x, z) = \sum_{j=1}^m a^j(z) \phi^j(x).$$

Reproducing kernel Hilbert space (RKHS)

- On the previous slide we wrote $a^j(z)$ in place of $\phi^j(z)$ to emphasise the parameterisation we are using for members of \mathcal{F} .
- If we now use $\phi^j(z)$ instead, in the infinite case we obtain

$$k(x, z) = \sum_{j=1}^{\infty} \phi^j(z)\phi^j(x) = \sum_{j=1}^{\infty} \phi^j(x)\phi^j(z) = k(z, x).$$

- We have ended up with the function $k(x, z)$ being the inner product of $\phi(x)$ and $\phi(z)$.
- Now we know how the construction works, we make the definition.

Reproducing kernel Hilbert space (RKHS)

- Let \mathcal{F} be a Hilbert space of functions $f : \mathcal{X} \rightarrow \mathbb{R}$.
- A function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is called a *reproducing kernel* if
 1. The feature map of every point is a function in \mathcal{F} : $\forall x \in \mathcal{X}, k(\cdot, x) \in \mathcal{F}$.
 2. The reproducing property: $\forall x \in \mathcal{X}, \forall f \in \mathcal{F}, \langle f, k(\cdot, x) \rangle_{\mathcal{F}} = f(x)$.

Reproducing kernel Hilbert space (RKHS)

- A particular case gives us $\forall x, z \in \mathcal{X}$,

$$k(z, x) = \langle k(\cdot, z), k(\cdot, x) \rangle_{\mathcal{F}}.$$

- If \mathcal{F} has a reproducing kernel, it is called a *reproducing kernel Hilbert space* (RKHS).
- It is easy to check that a reproducing kernel is a kernel
 - so the reproducing kernel is positive definite.
- It turns out that:
 - for every positive definite k , there exists a unique RKHS
 - a RKHS can be written as the span of $k(\cdot, x) = \phi(x) \quad \forall x \in \mathcal{X}$.

Valid kernels

- A necessary and sufficient condition for there to exist a unique RKHS with reproducing kernel k is that k is positive definite
 - the Moore-Aronszajn theorem.
- Also, k is positive definite if and only if k is a kernel (i.e. has a representation as an inner product between features, as defined on a previous slide)
 - this representation may not be unique.
- The following three properties are equivalent
 - k is positive definite
 - k is a kernel
 - k is a reproducing kernel.

Valid kernels

- We can build kernels from other kernels, e.g.
 - if k is a kernel on \mathcal{X} , then for $c > 0$, ck is also a kernel on \mathcal{X}
 - if k_1 and k_2 are kernels on \mathcal{X} , then $k_1 + k_2$ is also a kernel on \mathcal{X}
 - if k_1 and k_2 are kernels on \mathcal{X} , then k_1k_2 is also a kernel on \mathcal{X}
 - many more.
- These properties are enough to show that our second and third examples of kernels were valid kernels (after having shown directly that the first was).

Representer theorem

- Suppose we wish to find

$$\hat{f} = \arg \min_{f \in \mathcal{F}} \hat{R}(f) + \Gamma(\|f\|_{\mathcal{F}}^2)$$

where \mathcal{F} is a RKHS and Γ is some increasing function.

- It turns out that the solution takes the form

$$\hat{f} = \sum_{i=1}^n \alpha_i k(\cdot, X_i).$$

- Although \mathcal{F} may be infinite, the solution consists of a finite number of terms
 - see the example of smoothing splines earlier in the module.

Summary

- The "kernel trick" is the idea of replacing an inner product with a kernel, which implicitly uses a potentially high or infinite dimensional feature space.
- It is used in support vector machines, but is not exclusive to this method.
- Over the past lectures, we introduced many of the important models and methods used in machine learning.
- In the next few lectures we will look at the role played by Bayesian approaches.

Further reading

- ISLR chapter 9.
- ESL chapter 12.
- PRML chapters 6 and 7.

Bayesian statistics

ST420 Lecture 13

Richard Everitt

This lecture

- We will introduce Bayesian methods, which are the focus for the next few lectures.
- Begin from regularisation in regression
 - we will see that Bayesian statistics provides us with a general way of doing regularisation.
- Review some of the main ideas of Bayesian statistics.
- Begin to describe some of the main challenges.

Empirical risk minimisation and likelihoods

- ERM. We wish to find β (which fully determine f) that satisfy

$$\arg \min_{\beta \in \mathbb{R}^{p+1}} \frac{1}{n} \sum_{i=1}^n L(Y_i, f(X_i)).$$

- Recall

- squared (L_2) error loss

$$L(Y, f(X)) = (Y - f(X))^2.$$

- log-likelihood loss

$$L(Y, f(X)) = -2 \log l(Y | X, \theta)$$

where l is some distribution of Y given X and θ .

- What is the loss function doing?

- giving a score that is minimised when choosing "good" values of θ .

Empirical risk minimisation and likelihoods

- Squared error loss
 - choose θ values that give rise to small prediction errors of some function f that predicts Y from X .
- Log-likelihood loss
 - choose θ values that make the data most likely under some model for the distribution of $Y | X, \theta$.
- The main difference is that in the latter case we have a model for the distribution $l(Y | X, \theta)$, rather than just a function.
- Previously we asserted that using these two loss functions gave rise to the same results in the case of linear regression with Gaussian noise.

Empirical risk minimisation and likelihoods

- Choose $l(Y | X, \theta)$ to be Gaussian with mean $X^T \beta$ and variance σ^2 .
- Finding the maximum likelihood using this model gives the same result as minimising the empirical risk under log-likelihood loss.
- Then

$$\begin{aligned} -2 \log l(Y | X, \theta) &= -2 \log \prod_{i=1}^n \frac{1}{\sigma \sqrt{2\pi}} \exp\left(-\frac{(Y_i - X_i^T \beta)^2}{2\sigma^2}\right) \\ &= -2 \sum_{i=1}^n -\frac{1}{2} \log[2\pi\sigma^2] - \frac{1}{2} \left(\frac{(Y_i - X_i^T \beta)^2}{\sigma^2} \right) \\ &= n \log[2\pi\sigma^2] + \frac{1}{\sigma^2} \sum_{i=1}^n (Y_i - X_i^T \beta)^2. \end{aligned}$$

- Notice the factor of two in the log-likelihood loss cancels.

Using the likelihood

- Maximising the likelihood is the same as minimising risk under log-likelihood loss, which is the same as minimising squared error loss.
- From the point of view we have been following for much of the module:
 - the likelihood is simply a way of arriving at a suitable loss function;
 - it is very general way of doing so, since we simply need to define a model $l(Y | X, \theta)$.
- Thinking a little more widely, there are also additional benefits of the likelihood view...
 1. One might argue it gives a little more insight into our approach than the loss function, e.g. logistic regression is usually presented as a model $l(Y | X, \beta)$, rather than as minimising the risk under logistic loss.
 2. It gives us a "probabilistic" approach that describes the predictive distribution $l(Y_{\text{pred}} | X_{\text{pred}}, \hat{\beta})$ of a predicted response Y_{pred} given some new input X_{pred} . Previously we have only considered point estimates in regression.

Using Bayes: MAP estimates

- Recall that Bayesian statistics uses a likelihood and also a prior distribution $p(\beta)$ on β
 - we will recap on the detail of this a little later in the lecture.
- The *maximum a posteriori* (MAP) estimate is the value of β that maximises

$$l(Y | X, \beta)p(\beta),$$

or equivalently

$$\log l(Y | X, \beta) + \log p(\beta),$$

- We now need to choose $p(\beta)$ as well as $l(Y | X, \beta)$
 - but this gives us a different class of estimators of β .
- Consider the same model before for $l(Y | X, \beta)$, and choose $p(\beta^j)$ to be independent normal distributions for $j = 1, \dots, p$ with zero mean and variance τ^2 .

Using Bayes: MAP estimates

- Continue with our example

$$\begin{aligned}-2 \log l(Y|X,\theta) - 2 \log p(\beta) &= n \log[2\pi\sigma^2] + \frac{1}{\sigma^2} \sum_{i=1}^n (Y_i - X_i^T \beta)^2 - 2 \left[-\frac{1}{2} \log[2\pi\tau^2] - \frac{1}{2} \left(\frac{\beta^T \beta}{\tau^2} \right) \right] \\&= n \log[2\pi\sigma^2] + \frac{1}{\sigma^2} \sum_{i=1}^n (Y_i - X_i^T \beta)^2 + \log[2\pi\tau^2] + \left(\frac{\beta^T \beta}{\tau^2} \right) \\&= C + \frac{1}{\sigma^2} \left(\sum_{i=1}^n (Y_i - X_i^T \beta)^2 + \frac{\sigma^2}{\tau^2} \beta^T \beta \right).\end{aligned}$$

- Note that the quantity in the brackets is the same as that used in ridge regression with squared error loss, and for a particular λ
 - the ridge regression solution is the same as the MAP estimate with this choice of prior and likelihood.

Using Bayes: MAP estimates

- Consider the same model before for $l(Y | X, \beta)$, and choose $p(\beta^j)$ to be independent Laplace distributions for $j = 1, \dots, p$

$$p(\beta^j) = \frac{1}{2\tau} \exp(-|\beta|/\tau).$$

- The MAP estimate under this prior is the same as the lasso estimator, with $\lambda = \sigma^2/\tau$.
- Note that in both cases, we are assuming $\beta_0 = 0$
 - to include β_0 we could use a uniform improper prior on β_0 , i.e. $p(\beta_0) \propto 1$.
- We have seen that the prior distribution in Bayesian approaches can be used to provide regularisation..
- We will see that using other priors to produce different regularisation is possible, including doing variable selection.

Advantages of Bayesian methods

- We saw that likelihoods could be interpreted as a loss function, but that they also offer more than this.
- The same is true of Bayesian methods.
- Since we are using a likelihood, we still have the two advantages of the likelihood-based approaches:
 1. Interpretability.
 2. Existence of a predictive distribution.
- Using Bayesian methods, we also have the following advantages:
 1. Regularisation approaches that have an interpretation as prior distributions.
 2. Model averaging can be used for variance reduction.
- It can be worth it! (see ESL section 11.9).

Disadvantages of Bayesian statistics

- We can (arguably) achieve most of the above advantages through using frequentist approaches, as we have been doing so far in the module.
- Bayesian inference (which we need to do in order to fit the models) is typically much more difficult than the optimisation problems we have needed to solve so far in the module.
- Using Bayesian methods for big data problems is at the forefront of current research, and sometimes is not easy.

Example of Bayesian inference



Bayesian statistics

- Let:
 - θ some unknown parameter
 - y be observed data
 - $p(\theta)$ be a *prior* distribution on θ
 - $l(y | \theta)$ be a model for data y given parameter θ .
- $p(\theta)$ and $l(y | \theta)$ completely specify a joint distribution on θ and y
 - Bayesian inference makes use of standard properties of joint distributions to perform inference.
- In Bayesian statistics, we wish to find the posterior distribution $\pi(\theta | y)$.

Bayesian statistics

- We use Bayes' theorem

$$\pi(\theta | y) = \frac{p(\theta)l(y | \theta)}{p(y)}.$$

$$p(y) = \int_{\theta} p(\theta)l(y | \theta)d\theta$$

is known as the *marginal likelihood*, or *evidence*.

- This is simply the result of
 - specifying a joint distribution using $p(\theta)$ and $l(y | \theta)$
 - using a property of joint distributions.
- We may then find statistics of the posterior, such as the posterior expected value of θ

$$\mathbb{E}[\theta | y] = \int_{\theta} \theta \pi(\theta | y) d\theta.$$

Bayesian statistics

- Suppose that we have another unobserved variable ψ that is used in specifying the model $l(y | \theta, \psi)$ for the data
 - then we need to specify a joint prior $p(\theta, \psi)$ on θ and ψ
 - if we believe θ and ψ to be independent (before observing the data), we might use

$$p(\theta, \psi) = p(\theta)p(\psi)$$

- or if a sensible prior for ψ depends on θ , we might use

$$p(\theta, \psi) = p(\theta)p(\psi | \theta).$$

Bayesian statistics

- Suppose we are interested in the marginal posterior $\pi(\theta \mid y)$.
- We could find this through using Bayes' theorem to find the joint posterior on θ and ψ

$$\pi(\theta, \psi \mid y) = \frac{p(\theta, \psi)l(y \mid \theta, \psi)}{p(y)},$$

then marginalising over ψ

$$\pi(\theta \mid y) = \int_{\psi} \pi(\theta, \psi \mid y) d\psi.$$

Bayesian computation

- On the previous slides, we did not mention that the integrals are usually not analytically tractable!
- The marginal likelihood

$$p(y) = \int_{\theta} p(\theta)l(y | \theta)d\theta$$

cannot usually be found analytically

- the exception is when a "conjugate prior" is chosen for θ , but this is not possible except for a few situations
- this means we cannot usually analytically find $\pi(\theta | y)$ or any of its properties (such as expectation).

Bayesian computation

- When we have a joint posterior, it is usually not possible to marginalise it analytically, e.g.

$$\int_{\psi} \pi(\theta, \psi \mid y) d\psi$$

is not usually available.

- "Bayesian computation" refers to the approximation or estimation of quantities (usually integrals) in Bayesian inference.

Bayesian computation

- Before the 1990s, Bayesian inference was not particularly popular, due to the lack of methods that were capable of estimating or approximating these integrals
 - a particularly challenging case arises when there are a large number of unknowns - in this case the integrals are over a high-dimensional space.
- After this, Monte Carlo became popular for estimating the required integrals
 - a development that became possible due to the availability of desktop computers.
- A key idea is to use points simulated from $\pi(\theta | y)$ to approximate expectations of functions.
- However
 - Monte Carlo is challenging with big data and big models (those with lots of parameters)
 - the topic for next week.
- The remainder of this lecture discusses some of the implications of working in the Bayesian framework.

Predictive distribution: likelihood case

- Suppose that $\hat{\theta}$ is the maximum likelihood estimate of the parameters of a supervised learning model, estimated using likelihood $l(Y | X, \theta)$ on some training data \mathcal{T} .
- Then $l(\tilde{Y} | X, \hat{\theta})$ defines a predictive distribution
 - given a new data point (vector of predictors) $X = \tilde{X}$, $l(\tilde{Y} | \tilde{X}, \hat{\theta})$ defines a distribution over possible values of \tilde{Y} given the fitted model, and the predictors x
 - for example, in the linear/Gaussian regression case, $\hat{\theta} = (\hat{\beta}, \hat{\sigma}^2)$ and the predictive distribution is
$$\mathcal{N}(\tilde{X}^T \hat{\beta}, \hat{\sigma}^2).$$
- We can express some uncertainty about our prediction.

Predictive distribution: Bayesian case

- Desired distribution (where \mathcal{T} is training data)

$$\tilde{l}(\tilde{Y} \mid \tilde{X}, \mathcal{T}).$$

- We have

$$\tilde{l}(\tilde{Y} \mid \tilde{X}, \mathcal{T}) = \int_{\theta} l(\tilde{Y} \mid \tilde{X}, \theta) \pi(\theta \mid \mathcal{T}) d\theta.$$

- If we have points $\{\theta_i\}_{i=1}^N$ simulated from $\pi(\theta \mid y)$, we can approximate the predictive distribution using

$$\frac{1}{N} \sum_{i=1}^N l(\tilde{Y} \mid \tilde{X}, \theta_i).$$

- Compare to the likelihood case

- we are averaging over many values of θ , rather than just using one.

Implications of using Bayes

- Compare the Bayesian predictive distribution to the likelihood case
 - we are averaging over many values of θ , rather than just using one.
- This averaging idea, which is a consequence of the Bayesian approach, can mitigate overfitting.
- Note that all quantities that were conditioned on $\hat{\theta}$ in the frequentist case, now depend on the random θ from the posterior $\pi(\theta | \mathcal{T})$
 - previously we would have considered how the sampling distribution of $\hat{\theta}$ affected quantities conditioned on $\hat{\theta}$
 - now the randomness comes from the posterior $\pi(\theta | \mathcal{T})$.

Implications of using Bayes

- One nice consequence of the Bayesian approach is that
 - $\pi(\theta | \mathcal{T})$ is available (as long as we can draw points from it)
 - whereas the sampling distribution of $\hat{\theta}$ is not usually available.

Bayesian model comparison

- Bayesian statistics has a built-in method for model comparison.
- Simply treat the model M as a random variable, with prior distribution $p(M)$.
- The relative posterior probability of two models M_1 and M_2 is known as the *posterior odds* and is given by

$$\frac{\pi(M_1 \mid y)}{\pi(M_2 \mid y)} = \underbrace{\frac{p(M_1)}{p(M_2)}}_{\text{prior odds}} \cdot \underbrace{\frac{l(y \mid M_1)}{l(y \mid M_2)}}_{\text{Bayes factor}}.$$

- Where does $l(y \mid M)$ come from?
 - this is the evidence for model M

$$l(y \mid M) = \int_{\theta_M} p(\theta_M) l(y \mid \theta_M) d\theta_M.$$

Next time

- Elaborate on how Monte Carlo is used for Bayesian computation.
- Then we will see these ideas in action when looking at Bayesian variable selection.

Monte Carlo methods

ST420 Lecture 14

Richard Everitt

Recap

- We have introduced Bayesian methods
 - the computation of integrals is the key to implementing Bayesian methods.
- In this lecture we introduce Monte Carlo methods for estimating the required integrals.
- There will not be many real examples in this lecture
 - these will be found in the next two lectures, where we will look at variable selection using Bayesian methods.

Recall

- We wish to estimate expectations under the posterior distribution $\pi(\theta | y)$, such as

$$\mathbb{E}[\theta | y] = \int_{\theta} \theta \pi(\theta | y) d\theta.$$

- We need to estimate such integrals even though we do not know the denominator in Bayes' theorem, the marginal likelihood $p(y)$
 - recall that the marginal likelihood is also of interest in its own right.
- Idea:
 - use Monte Carlo integration
 - turns the problem of integration into one of simulation.

Monte Carlo

- A commonly used idea is to use a *Monte Carlo* approximation to estimate an integral
 - let θ be a random variable with distribution π
 - if $(\theta_1, \dots, \theta_N)$ is an i.i.d. sample from π
 - then

$$\frac{1}{N} \sum_{i=1}^N h(\theta_i)$$

is an unbiased estimate of

$$\mathbb{E}[h(\theta)]$$

where h is some function (which will be the identity if all we need is the expectation, or the squared distance to the mean if we wish to estimate the variance).

- law of large numbers guarantees convergence as $N \rightarrow \infty$.

How can we simulate?

- When doing Bayesian inference we have an expression for the unnormalised posterior distribution $p(\theta)l(y | \theta)$.
- *Monte Carlo methods* are techniques for simulating from distributions.
- There are several population methods:
 - rejection sampling
 - importance sampling
 - sequential Monte Carlo
 - Markov chain Monte Carlo.

Markov chain Monte Carlo

- We would like to draw points from $\pi(\theta \mid y)$.
- Recall that Markov chains are sequences of random variables where each variable is conditionally independent of its predecessors in the sequence, given the previous variable.
- A Markov chain is defined by its transition distribution, $P(\theta_{t+1} \mid \theta_t)$.
- Some Markov chains have a limiting distribution.
 - under certain conditions

$$\lim_{N \rightarrow \infty} \|P^N(\cdot \mid \theta_0) - \pi(\cdot)\|_{\text{TV}} = 0,$$

where $\|\cdot\|_{\text{TV}}$ is total variation norm (definition not needed for this module).

- Idea:
 - construct a Markov chain that has $\pi(\theta \mid y)$ as its limiting distribution.

Metropolis-Hastings

- The following algorithm has limiting distribution $\pi(\theta \mid y)$, starting at some initial point θ_0 .
-

Algorithm 1: Metropolis-Hastings

For $t = 1 : N$

- Simulate $\theta^* \sim q(\cdot \mid \theta_{t-1})$
- Simulate $u \sim \mathcal{U}[0, 1]$
- if $u < \min\left\{1, \frac{l(y \mid \theta^*) p(\theta^*) q(\theta_{t-1} \mid \theta^*)}{l(y \mid \theta_{t-1}) p(\theta_{t-1}) q(\theta^* \mid \theta_{t-1})}\right\}$
 - $\theta_t = \theta^*$
 - else
 - $\theta_t = \theta_{t-1}$

Metropolis-Hastings

Metropolis-Hastings example.

