

Machine Learning, Machine Learning (extended)

1 – Introduction

Kashif Rajpoot

k.m.rajpoot@cs.bham.ac.uk

School of Computer Science
University of Birmingham

Outline

- What is machine learning?
- Applications
- Aims
- Learning outcomes
- Assessment
- Relevant texts
- Plagiarism
- Basics of machine learning
- What is the learning problem?
- Classes of learning
- Common terminology
- List of topics

What is machine learning?

- Algorithms that enable computers to learn from examples
- Algorithms learn from example observations of objects
 - Speech?
 - Image?
 - Health symptoms?
 - Stock price?
 - Personal shopping choice?

What is machine learning?

Given example observations of objects:

- Can we find similar objects?
- Can we make predictions about objects?
- Can we group the objects?

What is machine learning?

Fortune Teller



PREDICTIVE ANALYTICS



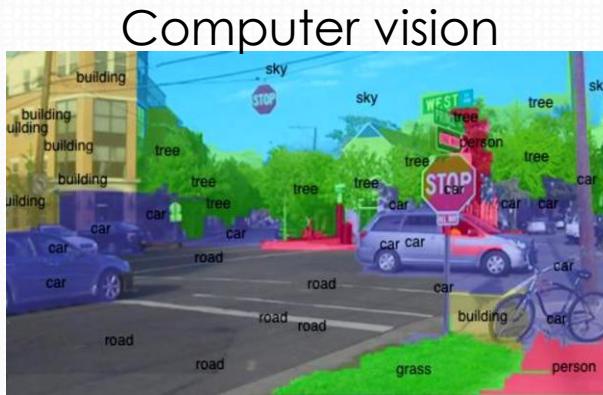
*"Why the change? Well, I could see
where the future was going..."*

Machine learning

- Variety of algorithms
 - Often the algorithm parameters need to be tuned
 - Each algorithm has its pros and cons
- Important to understand the algorithms
- We will discuss a small selection of algorithms..
 - ..but covering variety

Applications

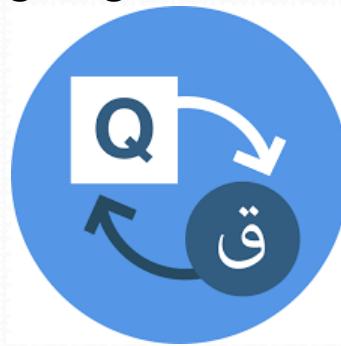
Speech recognition



Robot control



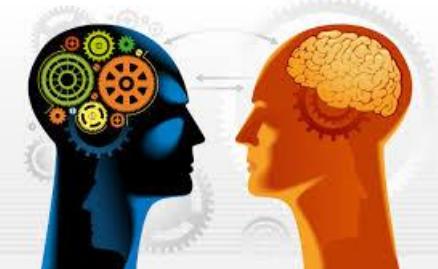
Language translation



Text or document classification (e.g. spam email)

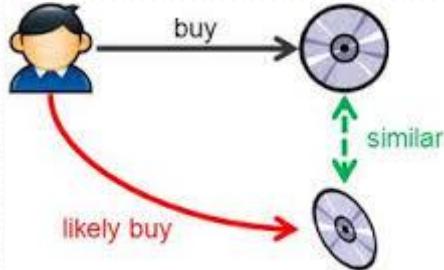


Natural language processing



Applications

Recommendation system



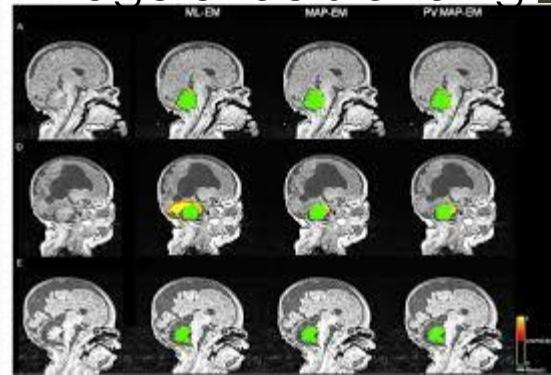
Game playing
(e.g. chess, go, backgammon)



Personal software assistant



Image understanding



Driverless cars



ML in industry

- Companies with lots of data, with a need to ‘understand’ the data, for which traditional algorithms don’t exist
- Google (in almost everything)
- Microsoft (e.g. personal assistant)
- Amazon (e.g. recommendation system)
- Facebook (e.g. friends suggestion, face tagging)
- Uber (e.g. driverless navigation)

Aims

1. Introduce the basic concepts and terminology of machine learning
2. Give an overview of the main approaches to machine learning
3. Show similarities and differences between different approaches
4. Present basic principles for the classification of approaches to machine learning
5. Give practical experience of applying machine learning algorithms to classification and data analysis problems
6. (ML extended only) Develop skills of literature surveying and critical thinking in an area of machine learning

Learning outcomes

On successful completion, the student should be able to:

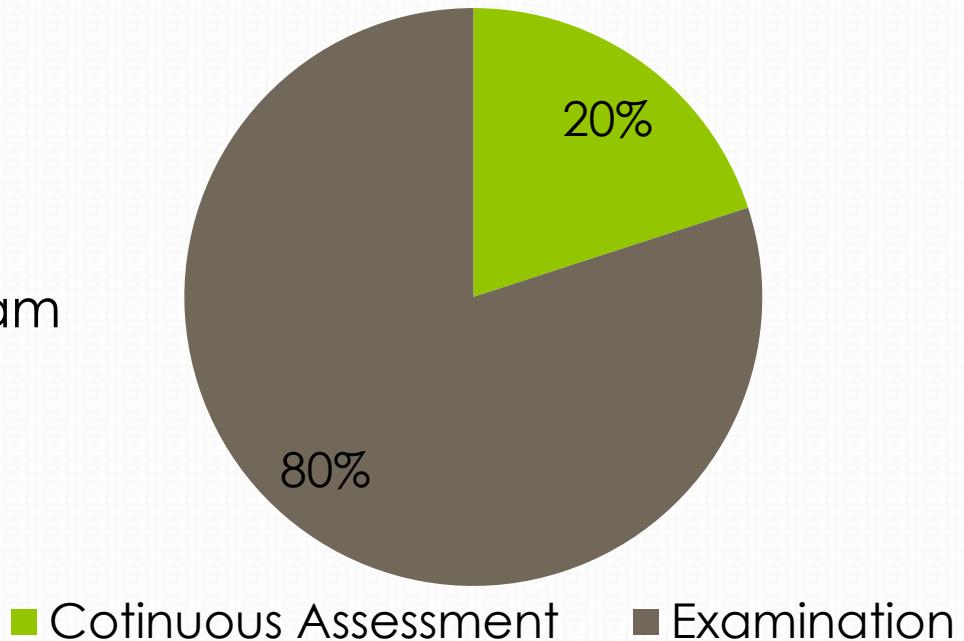
1. Demonstrate a knowledge and understanding of the main approaches to machine learning
2. Demonstrate the ability to apply the main approaches to unseen examples
3. Demonstrate an understanding of the differences, advantages and problems of the main approaches in machine learning
4. Demonstrate an understanding of the main limitations of current approaches to machine learning, and be able to discuss possible extensions to overcome these limitations
5. Demonstrate a practical understanding of the use of machine learning algorithms
6. (ML extended only) Survey and discuss the research literature in one subfield of machine learning

Module focus

- Understanding the fundamental principles
 - Commonly used algorithms
 - Common pitfalls
 - Categories of algorithms
- NOT a module on ML software packages

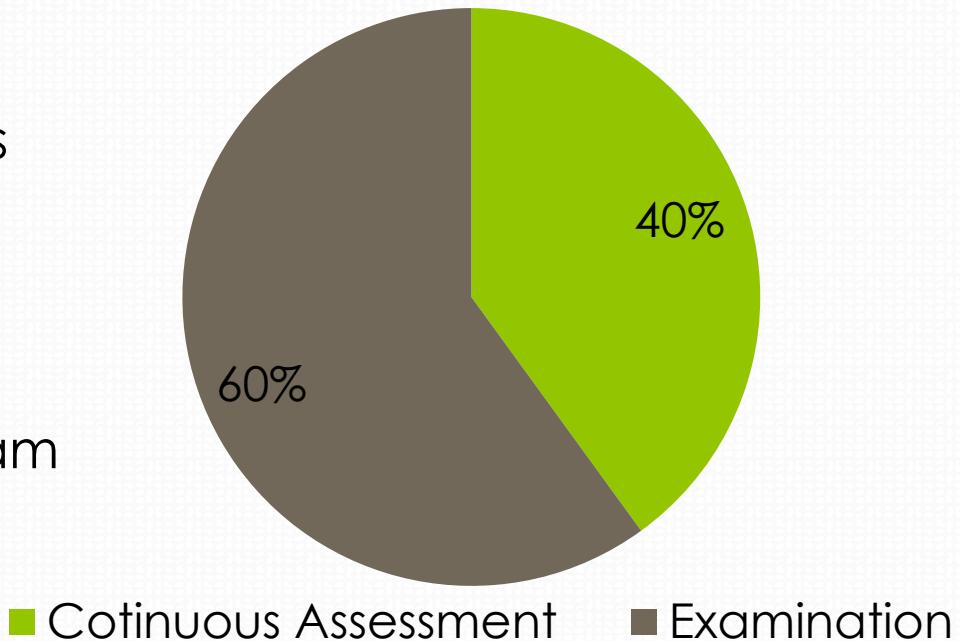
Assessment: machine learning

- Continuous assessment (20%)
 - Class test (20%)
- Examination (80%)
 - 90 minutes written exam
 - Closed-book and closed-notes exam



Assessment: machine learning (extended)

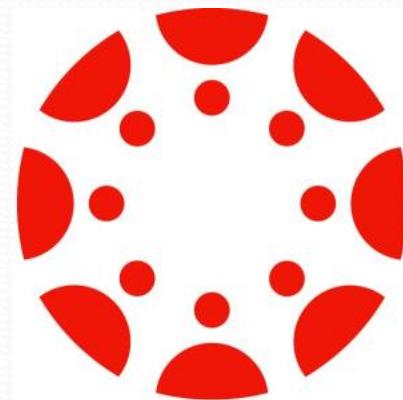
- Continuous assessment (40%)
 - Class test (20%)
 - Computer based tests ($2 \times 10\% = 20\%$)
- Examination (60%)
 - 90 minutes written exam
 - Closed-book and closed-notes exam



Assessment schedule (tentative)

Week	Test (ML)	Test (ML extended)
1		
2	Computer Based Test (ungraded) announced (6 th Oct)	Computer Based Test (ungraded) announced (6 th Oct)
3		
4	Computer Based Test (ungraded) DUE (20 th Oct)	Computer Based Test (ungraded) DUE (20 th Oct)
5	Online Test (ungraded) (27 th Oct)	Online Test (ungraded) (27 th Oct) Computer Based Test 1 (GRADED) announced (27 th Oct)
6		
7		Computer Based Test 1 (GRADED) DUE (10 th Nov)
8	Class Test (GRADED) (17 th Nov)	Class Test (GRADED) (17 th Nov) Computer Based Test 2 (GRADED) announced (17 th Nov)
9		
10		Computer Based Test 2 (GRADED) DUE (1 st Dec)
11		

Module website

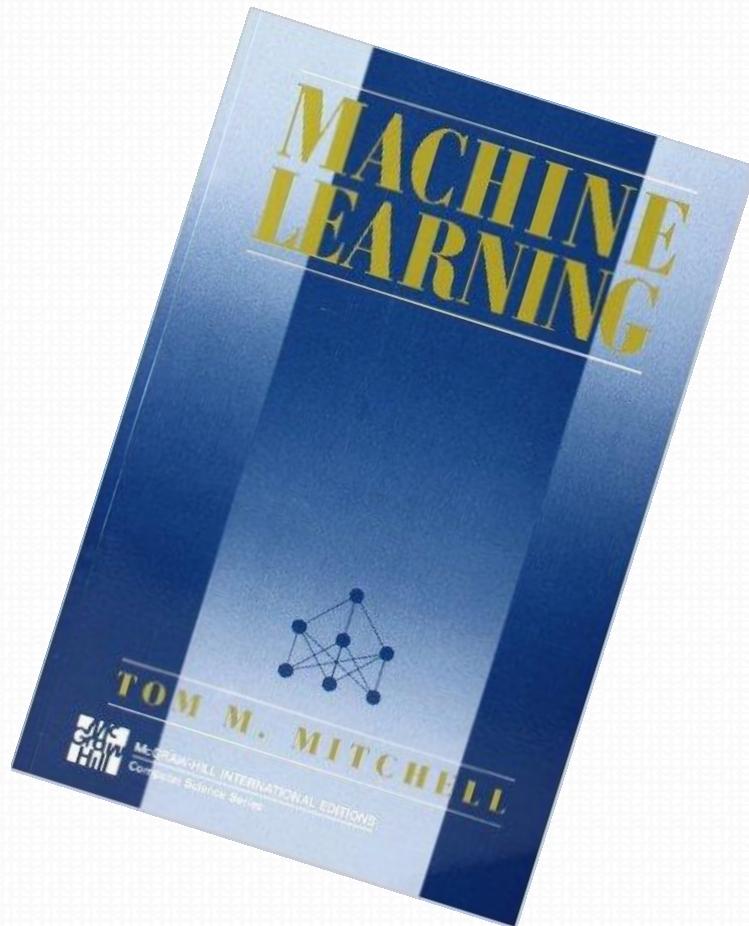
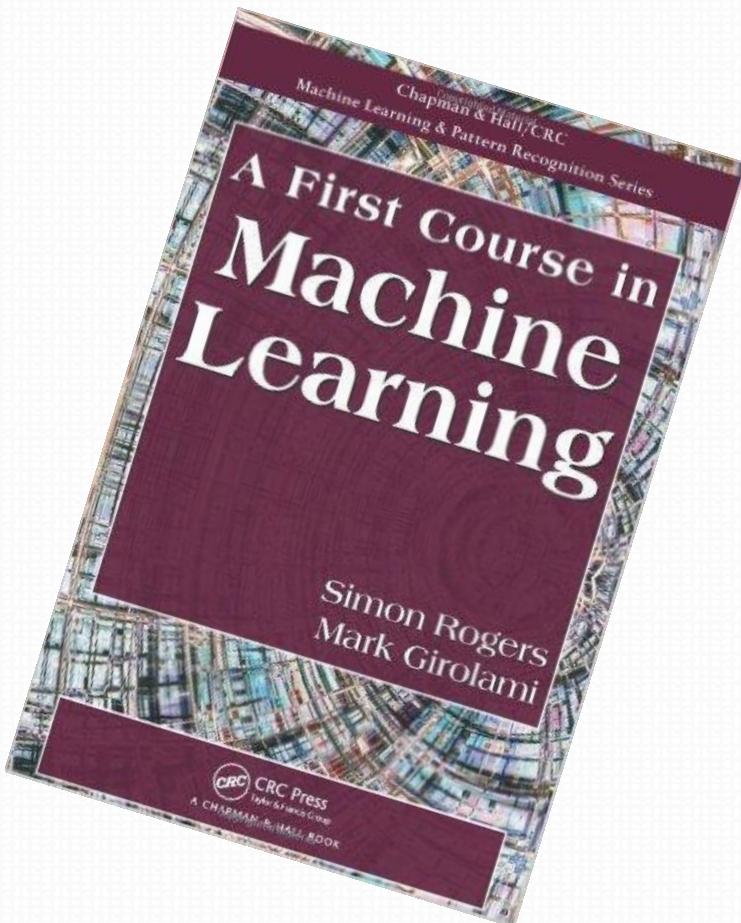


- Module Canvas page
 - <https://canvas.bham.ac.uk/courses/27269>
- Lecture slides will be uploaded weekly
- Announcements/discussions
- Computer based test submission
- Online class test

Office hours

- Tuesday 9.30am-11am
- Location: LG06d (lower ground floor)

Relevant texts



Plagiarism



- <https://intranet.birmingham.ac.uk/as/studentservices/conduct/plagiarism/index.aspx>
- <https://intranet.birmingham.ac.uk/as/studentservices/conduct/plagiarism/guidance-students.aspx>
- <http://www.birmingham.ac.uk/Documents/university/legal/plagiarism.pdf>

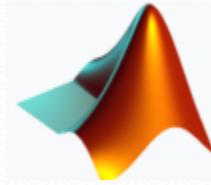
Pre-requisites

- Mathematical techniques for computer science (or equivalent)
- Introduction to AI (or equivalent)
- Math refresher material is available on Canvas
 - Linear algebra
 - Probability theory

Math refreshers

- Linear Algebra
 - Canvas
(https://canvas.bham.ac.uk/files/4348230/download?download_frd=1)
 - A First Course in Machine Learning (section 1.3)
- Probability theory
 - Canvas
(https://canvas.bham.ac.uk/files/4348231/download?download_frd=1)
 - A First Course in Machine Learning (sections 2.2 to 2.6)

MATLAB



- MATLAB is a very popular numerical computing environment
 - For computer bases tests, solution is required to be developed in MATLAB
 - Available for free through University's campus-wide license (<https://mysoftware.bham.ac.uk>)
- MATLAB basics (vectors, matrices, loops, plotting, etc)
 - <http://www.cyclismo.org/tutorial/matlab/>
 - <http://users.rowan.edu/~shreek/networks1/matlabintro.html>
- MATLAB primer (by Mathworks)
 - http://au.mathworks.com/help/pdf_doc/matlab/getstart.pdf

Basics of machine learning

What is the learning problem?

- Ability to improve performance (or to make accurate predictions) through experience to perform a task
 - Improve at task T, with respect to performance measure P, based on experience E
- Task?
- Performance measure?
- Experience?

What is the learning problem?

- Learning to play checkers
- Task T?
- Performance measure P?
- Experience E?



What is the learning problem?

- Learning to recognize handwritten words

- Task T?

- Performance measure P?

- Experience E?

Sincerely,
Albert

What is the learning problem?

- Learning to recognize faces
- Task T?
- Performance measure P?
- Experience E?



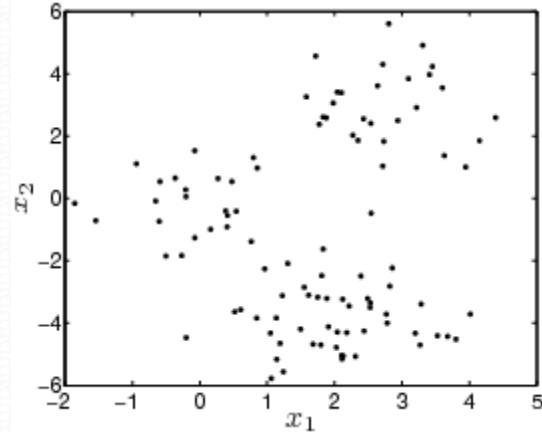
What is the learning problem?

- Learning to drive autonomously
- Task T?
- Performance measure P?
- Experience E?



What is the learning problem?

- Learning to find clusters in data
- Task T?
- Performance measure P?
- Experience E?



What is the learning problem?

- Learning to interpret image scene

- Task T?



1: art gallery



2: restaurant



3: computer room



4: biology laboratory



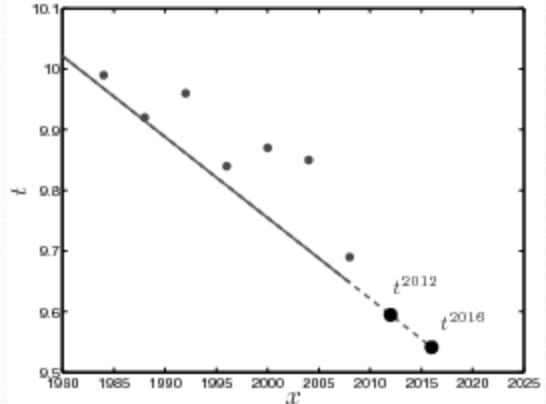
5: picnic area

- Performance measure P?

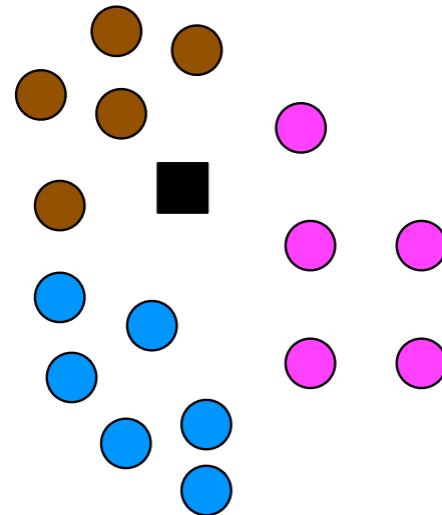
- Experience E?

Classes of learning

- **Regression:** learning a continuous function from a set of past examples
 - Predict a real value target for a future example
 - e.g. predict winning time in Olympic race

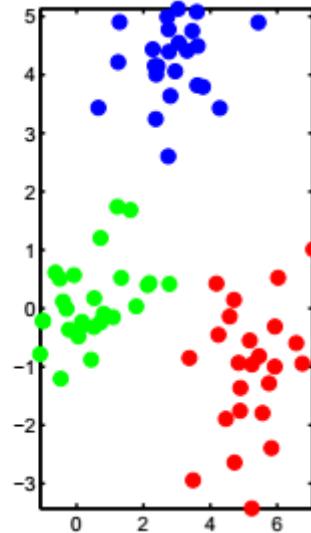
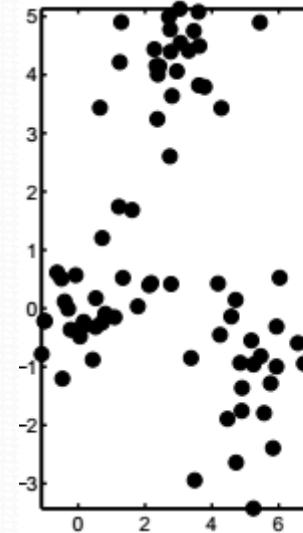


- **Classification:** Learning a function that can separate past examples of different types from one another
 - Assign a discrete target label/type for a future example
 - e.g. document classification

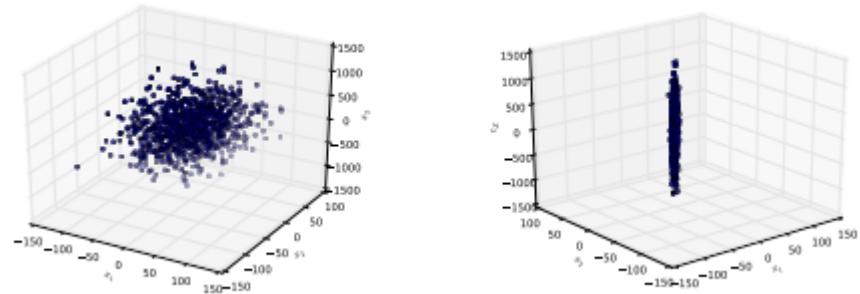


Classes of learning

- **Clustering:** partition examples into groups, each group having similar examples
 - e.g. brain regions with similar activation



- **Dimensionality reduction:** transform high-dimensional data into a lower-dimensional preserving representation
 - e.g. reducing unnecessary attributes



Training experience

- Direct or indirect feedback may be available
 - Chess game move
 - Digit recognition
 - Face recognition
- With or without a teacher
 - Examples (i.e. experience) with or without target labels
 - e.g. face recognition, data grouping
- Is the training experience representative of the performance goal?
 - e.g. digit recognition
 - How well the training examples distribution represent the true examples distribution?

Forms of machine learning

- **Supervised learning:** learner receives set of labelled examples (i.e. direct feedback) in order to learn to classify unseen examples
 - Classification, regression
- **Unsupervised learning:** learner receives set of unlabelled examples (i.e. no teacher) in order to learn to categorize unseen examples
 - Clustering
- **Dimensionality reduction:** transform high-dimensional data into a lower-dimensional preserving representation

ML: important questions

- How much training data is sufficient?
- What algorithms exist for learning general target functions from specific training examples?
- Can we transfer what is learned from one task to improve learning in other related tasks?
- What is the relationship between different learning algorithms, and which should be used when?

ML: important questions

- Can we build never ending learners?
- Can machine learning theories and algorithms help explain human learning?
- Can we design programming language containing machine learning primitives?

Common terminology

- *Examples*: items of data used for learning or evaluation
- *Features*: attributes that characterize an example
- *Labels*: values or categories assigned to examples
- *Task*: a prediction activity that the algorithm needs to learn
- *Performance*: measure of prediction accuracy of an algorithm
- *Experience*: past examples which can be used in learning
- *Training*: learning to predict from examples
- *Testing*: predicting previously unseen examples

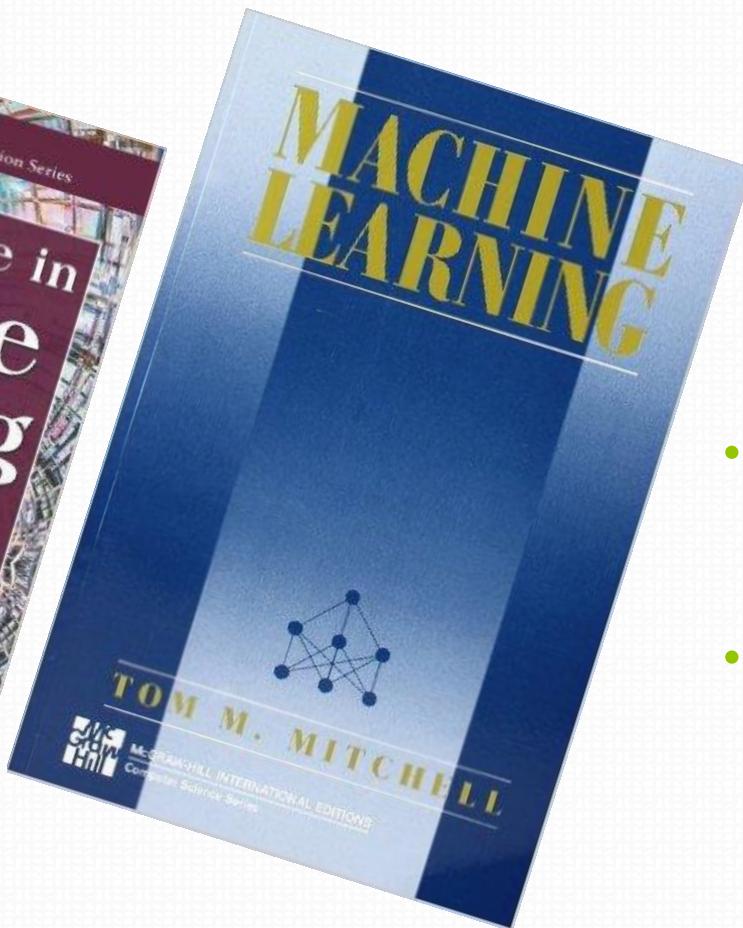
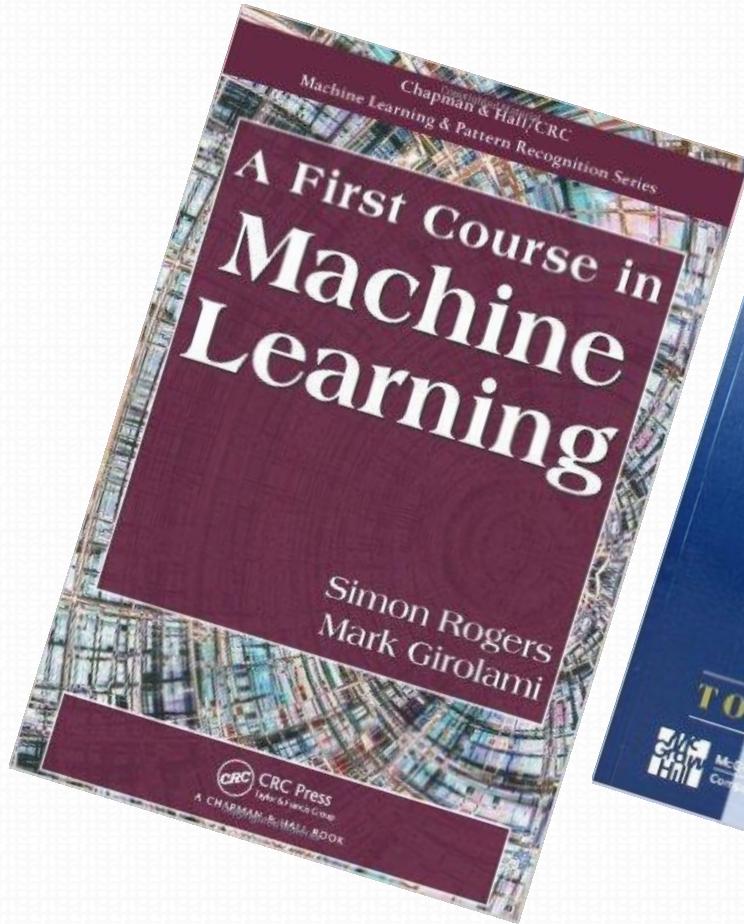
Common terminology

- *Cross validation*: distribute data into k-folds to train and evaluate algorithm performance
- *Training samples*: examples used to train algorithm
- *Validation samples*: examples used to tune algorithm parameters
- *Test samples*: examples used to evaluate algorithm
- *Loss function*: performance (loss) measure function
- *Learner function/model*: a function or model that is learnt to predict labels from features
- *Hypothesis set*: set of functions mapping features to labels

List of topics (tentative)

- Basics
- Supervised learning
 - Regression: linear modelling by least squares
 - Regression: linear modelling by maximum likelihood
 - Classification: Bayesian classification
 - Classification: instance-based classification
 - Classification: discriminative classification
- Unsupervised learning
 - Clustering: k-means clustering
 - Clustering: hierarchical clustering
 - Dimensionality reduction: principal component analysis
- Ensemble methods
 - Boosting
 - Random forests

C₃ R₁ E₂ D₂ I₁ T₁ S₁



Author's material
(Simon Rogers)

- Ata Kaban's material from previous years
- Various other sources for graphical illustration



Thank You

Machine Learning, Machine Learning (extended)

2 – Supervised Learning: Linear Modelling by Least Squares

Kashif Rajpoot

k.m.rajpoot@cs.bham.ac.uk

**School of Computer Science
University of Birmingham**

Outline

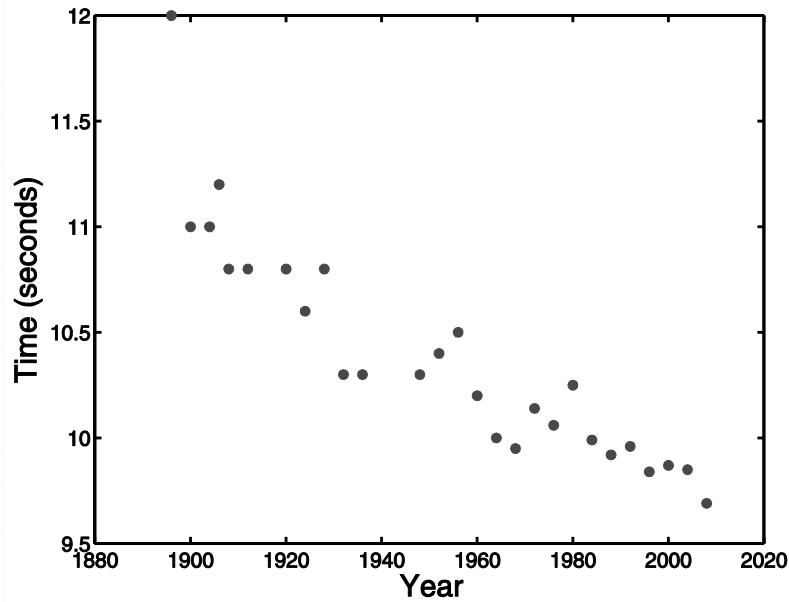
- Linear modelling
 - Least squares
 - Loss function
- Finding function's minimum
- Making predictions from model
- Linear modelling with vectors
- Non-linear response from a linear model
- Generalization and over-fitting
- Cross-validation
- Regularized least squares

Function modelling

- Determine a learner function/model
 - Learn relationship between attributes (i.e. features) and responses or targets (i.e. labels)
- Examples
 - Disease diagnosis
 - Image classification
 - Face recognition
 - Recommendation system
 - Speaker identification

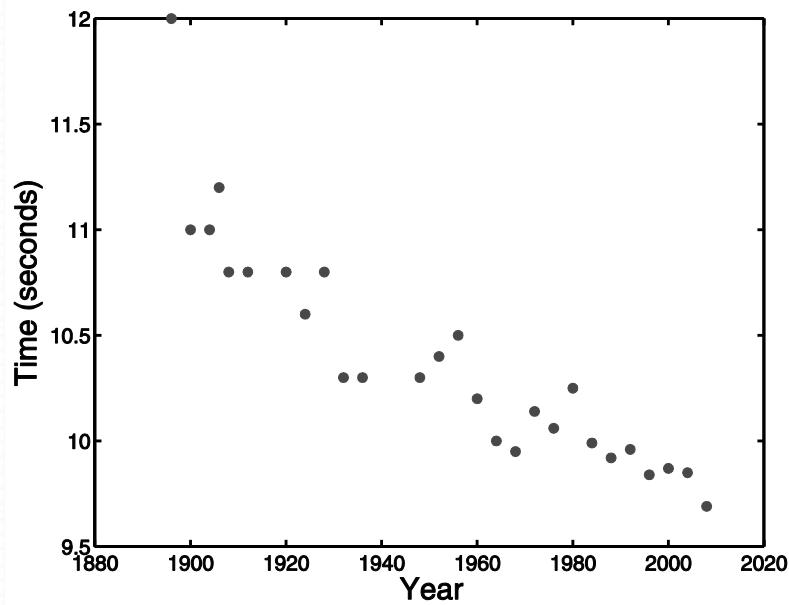
Linear modelling

- One of the most straightforward learning problems
 - Learn a linear function between attributes and responses
- Is there a functional dependence between Olympics year and 100m winning time?
 - Draw a line?
- Can we predict winning time for future games?



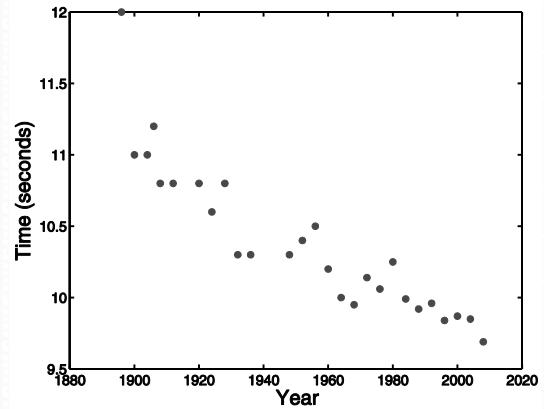
Linear modelling

- BIG assumptions
 - There is a relationship between Olympics year and winning time
 - This relationship is linear
 - This relationship will hold in future
- Are these good assumptions?



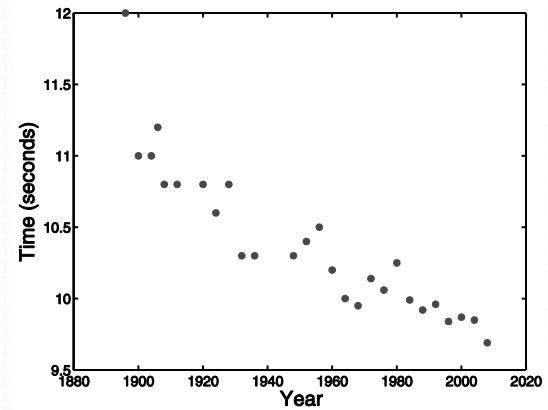
Linear modelling

- Learner model/function
 - Maps input attributes to output response
- Let's consider we can predict time $t = f(x)$
 - x ?
 - t ?
- Training samples
 - N attribute-response pairs $(x_1, t_1), (x_2, t_2), \dots (x_N, t_N)$



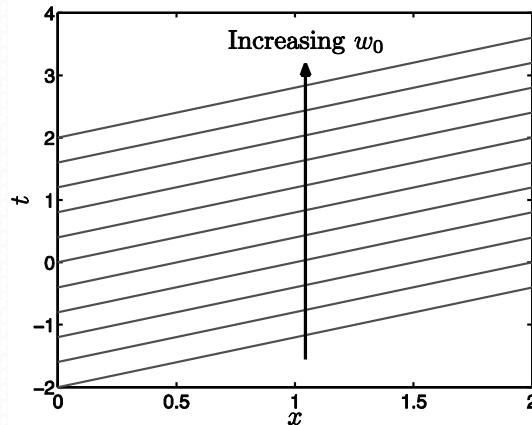
Linear modelling

- Let's consider we can predict time $t = f(x)$
 - Linear function: $f(x) = x$ and $f(x) = mx + c$
 - Non-linear function: $f(x) = \sin(x)$
- Can we model Olympic years and winning time with a linear function?
 - i.e. winning time drops by same amount every M years?
 - How about $t = f(x) = x$?
 - How about $t = f(x; w) = wx$?
 - How about $t = f(x; w_0, w_1) = w_0 + w_1 x$?

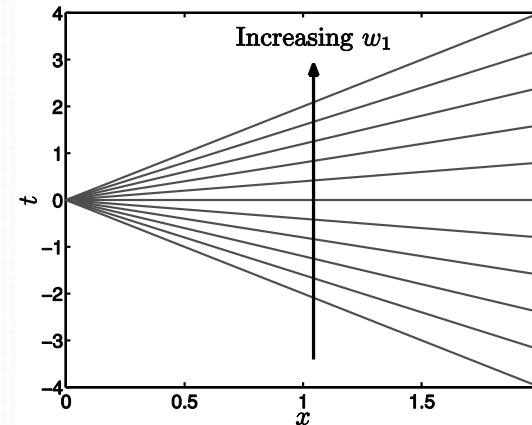


Linear modelling

$$t = f(x; w_0, w_1) = w_0 + w_1 x$$



(a) Increasing w_0 changes the point at which the line crosses the t axis



(b) Increasing w_1 changes the gradient of the line

- Learner function/model
 - Learn parameters w_0 and w_1 from past data $(x_1, t_1), (x_2, t_2), \dots (x_N, t_N)$ to predict future winning time t_{new} for a future year x_{new}

What's a good model?

- Generate a line that passes “as close as possible” to the past example points
- Loss function: loss between ‘ground truth’ t_n and model prediction $f(x_n; w_0, w_1)$

$$\mathcal{L}_n(t_n, f(x_n; w_0, w_1)) = (t_n - f(x_n; w_0, w_1))^2$$
$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^N \mathcal{L}_n(t_n, f(x_n; w_0, w_1))$$

- Can we use trial and error to find “best” w_0 and w_1 ?

What's a good model?

- Generate a line that passes “as close as possible” to the past example points
- Loss function: loss between ‘ground truth’ t_n and model prediction $f(x_n; w_0, w_1)$

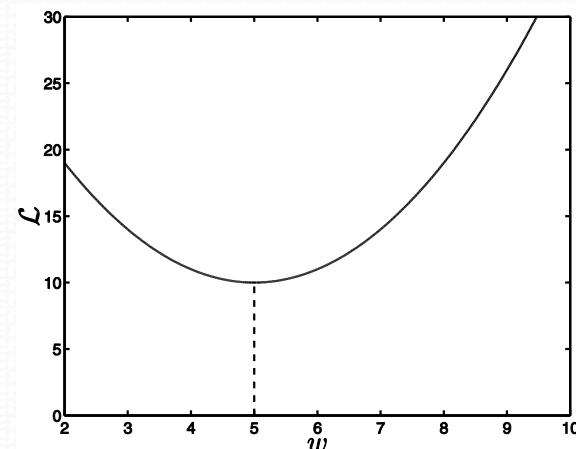
$$\mathcal{L}_n(t_n, f(x_n; w_0, w_1)) = (t_n - f(x_n; w_0, w_1))^2$$
$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^N \mathcal{L}_n(t_n, f(x_n; w_0, w_1))$$

- Find “best” w_0 and w_1 that reduces loss \mathcal{L}

$$\underset{w_0, w_1}{\operatorname{argmin}} \mathcal{L}$$

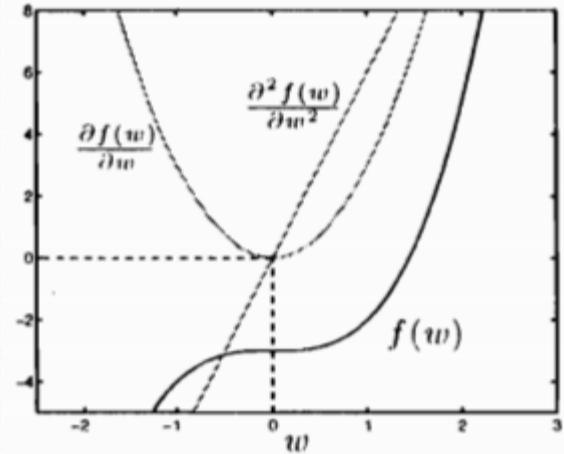
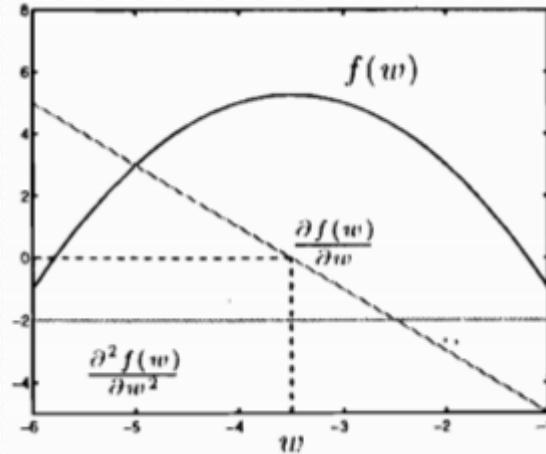
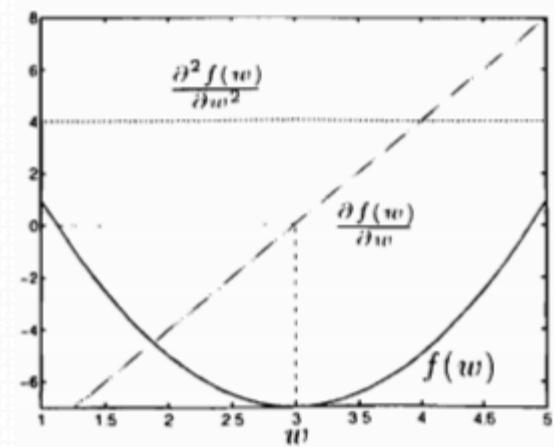
- Minimize least squares error

$$\underset{w_0, w_1}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^N (t_n - f(x_n; w_0, w_1))^2$$



Finding function's minimum

- A function's minimum or maximum can be determined where the 1st derivative is zero
 - Local minima
 - Local maxima
- At the function's minimum, 2nd derivative is positive



Linear modelling: least squares solution

- We are aiming to find a functional dependence relationship between Olympics year and winning time

$$f(x; w_0, w_1) = w_0 + w_1 x$$

- We can determine “best” w_0 and w_1 parameters (or weights) that minimize loss function \mathcal{L}

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^N (t_n - (w_0 + w_1 x_n))^2$$

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^N (w_1^2 x_n^2 + 2w_1 x_n (w_0 - t_n) + w_0^2 - 2w_0 t_n + t_n^2)$$

Finding function's minimum

- Partial derivatives, with respect to w_0 and w_1 , at minimum of \mathcal{L} must be 0

$$\frac{\partial \mathcal{L}}{\partial w_0} = \frac{\partial}{\partial w_0} \left[\frac{1}{N} \sum_{n=1}^N (w_1^2 x_n^2 + 2w_1 x_n (w_0 - t_n) + w_0^2 - 2w_0 t_n + t_n^2) \right]$$

$$\frac{\partial \mathcal{L}}{\partial w_0} = \frac{\partial}{\partial w_0} \left[w_0^2 + 2w_0 w_1 \frac{1}{N} \sum_{n=1}^N x_n - 2w_0 \frac{1}{N} \sum_{n=1}^N t_n \right]$$

$$\frac{\partial \mathcal{L}}{\partial w_0} = 2w_0 + 2w_1 \frac{1}{N} \left(\sum_{n=1}^N x_n \right) - \frac{2}{N} \left(\sum_{n=1}^N t_n \right) = 0$$

$$\hat{w}_0 = \bar{t} - w_1 \bar{x}$$

- w_1 ?

Finding function's minimum

- Partial derivatives, with respect to w_0 and w_1 , at minimum of \mathcal{L} must be 0

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial}{\partial w_1} \left[\frac{1}{N} \sum_{n=1}^N (w_1^2 x_n^2 + 2w_1 x_n (w_0 - t_n) + w_0^2 - 2w_0 t_n + t_n^2) \right]$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial}{\partial w_1} \left[w_1^2 \frac{1}{N} \sum_{n=1}^N x_n^2 + 2w_1 \frac{1}{N} \sum_{n=1}^N x_n (w_0 - t_n) \right]$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = 2w_1 \frac{1}{N} \left(\sum_{n=1}^N x_n^2 \right) + \frac{2}{N} \left(\sum_{n=1}^N x_n (w_0 - t_n) \right) = 0$$

- w_0 ?

Finding function's minimum

- Partial derivatives, with respect to w_0 and w_1 , at minimum of \mathcal{L} must be 0

$$\frac{\partial \mathcal{L}}{\partial w_1} = 2w_1 \frac{1}{N} \left(\sum_{n=1}^N x_n^2 \right) + \frac{2}{N} \left(\sum_{n=1}^N x_n (w_0 - t_n) \right) = 0$$

$$2w_1 \frac{1}{N} \left(\sum_{n=1}^N x_n^2 \right) + \frac{2}{N} \left(\sum_{n=1}^N x_n (\widehat{w}_0 - t_n) \right) = 0$$

$$\widehat{w}_0 = \bar{t} - w_1 \bar{x}$$

$$\widehat{w}_1 = \frac{\bar{x}t - \bar{x}\bar{t}}{\bar{x}^2 - (\bar{x})^2}$$

$$\widehat{w}_0 = \bar{t} - \widehat{w}_1 \bar{x}$$

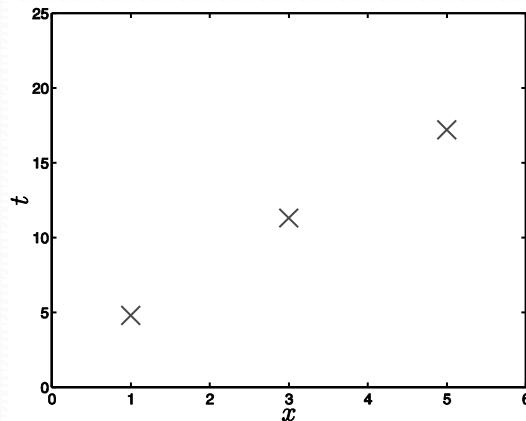
Linear modelling

$$\hat{w}_1 = \frac{\bar{x}t - \bar{x}\bar{t}}{\bar{x}^2 - (\bar{x})^2}$$

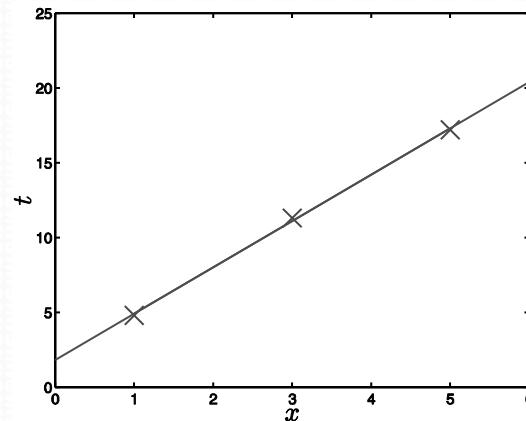
$$\hat{w}_0 = \bar{t} - \hat{w}_1 \bar{x}$$

- Compute model parameters from data

n	x_n	t_n	$x_n t_n$	x_n^2
1	1	4.8	4.8	1
2	3	11.3	33.9	9
3	5	17.2	86	25
$(1/N) \sum_{n=1}^N$	3	11.1	41.57	11.67



(a) The three synthetic data points described in Table 1.1



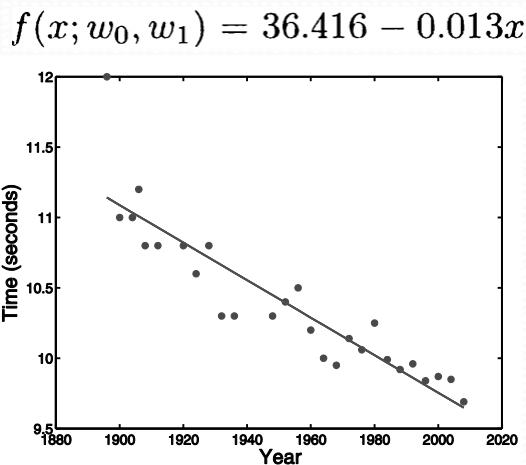
(b) The least squares fit defined by $f(x; w_0, w_1) = 1.8 + 3.1x$

Linear modelling

$$\hat{w}_1 = \frac{\bar{x}t - \bar{x}\bar{t}}{x^2 - (\bar{x})^2}$$

$$\hat{w}_0 = \bar{t} - \hat{w}_1 \bar{x}$$

- Compute model parameters from Olympics data



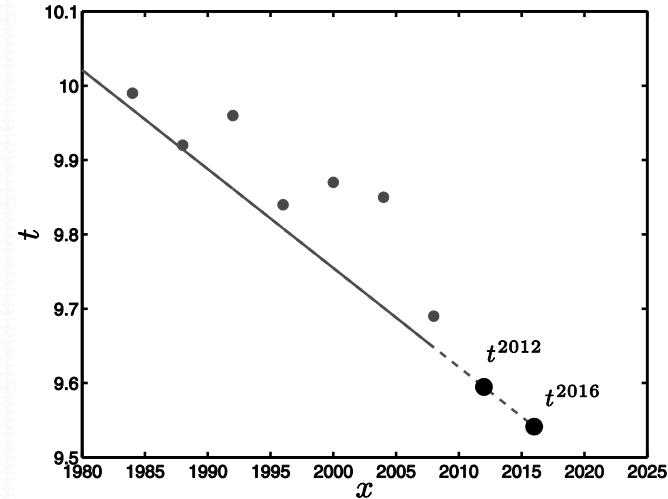
n	x_n	t_n	$x_n t_n$	x_n^2
1	1896	12.00	22752.0	3.5948×10^6
2	1900	11.00	20900.0	3.6100×10^6
3	1904	11.00	20944.0	3.6252×10^6
4	1906	11.20	21347.2	3.6328×10^6
5	1908	10.80	20606.4	3.6405×10^6
6	1912	10.80	20649.6	3.6557×10^6
7	1920	10.80	20736.0	3.6864×10^6
8	1924	10.60	20394.4	3.7018×10^6
9	1928	10.80	20822.4	3.7172×10^6
10	1932	10.30	19899.6	3.7326×10^6
11	1936	10.30	19940.8	3.7481×10^6
12	1948	10.30	20064.4	3.7947×10^6
13	1952	10.40	20300.8	3.8103×10^6
14	1956	10.50	20538.0	3.8259×10^6
15	1960	10.20	19992.0	3.8416×10^6
16	1964	10.00	19640.0	3.8573×10^6
17	1968	9.95	19581.6	3.8730×10^6
18	1972	10.14	19996.1	3.8888×10^6
19	1976	10.06	19878.6	3.9046×10^6
20	1980	10.25	20295.0	3.9204×10^6
21	1984	9.99	19820.2	3.9363×10^6
22	1988	9.92	19721.0	3.9521×10^6
23	1992	9.96	19840.3	3.9681×10^6
24	1996	9.84	19640.6	3.9840×10^6
25	2000	9.87	19740.0	4.0000×10^6
26	2004	9.85	19739.4	4.0160×10^6
27	2008	9.69	19457.5	4.0321×10^6
$(1/N) \sum_{n=1}^N$		1952.37	10.39	3.8130×10^6

Making predictions from model

- Predict winning time for years 2012 and 2016

$$f(x; w_0, w_1) = 36.416 - 0.013x$$

- Accurate & precise prediction?
 - Predicting past seen examples?
 - Predicting future unseen examples?
- Basis of prediction?

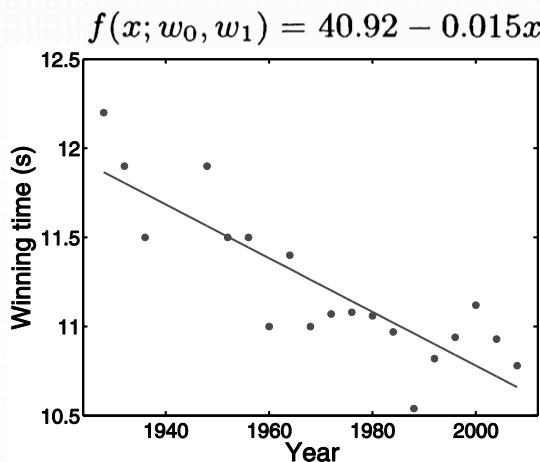


Making predictions from model

$$\hat{w}_1 = \frac{\bar{x}t - \bar{x}\bar{t}}{x^2 - (\bar{x})^2}$$

$$\hat{w}_0 = \bar{t} - \hat{w}_1 \bar{x}$$

- Compute model parameters from Olympics women's 100m data



n	x_n	t_n	$x_n t_n$	x_n^2
1	1928	12.20	23521.6	3.7172×10^6
2	1932	11.90	22990.8	3.7326×10^6
3	1936	11.50	22264.0	3.7481×10^6
4	1948	11.90	23181.2	3.7947×10^6
5	1952	11.50	22448.0	3.8103×10^6
6	1956	11.50	22494.0	3.8259×10^6
7	1960	11.00	21560.0	3.8416×10^6
8	1964	11.40	22389.6	3.8573×10^6
9	1968	11.00	21648.0	3.8730×10^6
10	1972	11.07	21830.0	3.8888×10^6
11	1976	11.08	21894.1	3.9046×10^6
12	1980	11.06	21898.8	3.9204×10^6
13	1984	10.97	21764.5	3.9363×10^6
14	1988	10.54	20953.5	3.9521×10^6
15	1992	10.82	21553.4	3.9681×10^6
16	1996	10.94	21836.2	3.9840×10^6
17	2000	11.12	22240.0	4.0000×10^6
18	2004	10.93	21903.7	4.0160×10^6
19	2008	10.78	21646.2	4.0321×10^6
$(1/N) \sum_{n=1}^N$		1970.74	11.22	3.8844×10^6

Making predictions from model

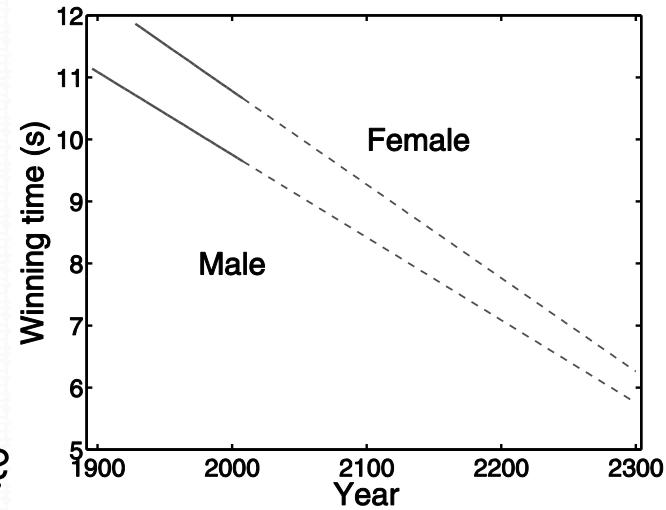
- Linear model from men's 100m data

$$f(x; w_0, w_1) = 36.416 - 0.013x$$

- Linear model from women's 100m data

$$f(x; w_0, w_1) = 40.92 - 0.015x$$

- Accurate & precise prediction?
 - Intersection of lines?
 - Distance between predicted and observed points?
 - Approaching zero?



Linear modelling

- What if we want to learn linear model from data having more than one attribute?
 - Olympics year (x)
 - Each athlete's personal best in lanes 1 to 8 (s_n)
- Linear model can then be represented as:

$$t = f(x, s_1, \dots, s_8; w_0, \dots, w_9)$$

$$\begin{aligned} t = & w_0 + w_1x + w_2s_1 + w_3s_2 + w_4s_3 \\ & + w_5s_4 + w_6s_5 + w_7s_6 + w_8s_7 + w_9s_8 \end{aligned}$$

- How to find the parameters w_n ?
 - Derive loss function \mathcal{L} ?
 - Take partial derivative with respect to each w_n ?

Linear modelling with vectors

- Let's consider our model

$$t = w_0 + w_1 x$$

- Let's write parameters and attributes in vector form:

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \quad \mathbf{x}_n = \begin{bmatrix} 1 \\ x_n \end{bmatrix}$$

- The model in equivalent vector notation:

$$t = f(x_n; w_0, w_1) = \mathbf{w}^\top \mathbf{x}_n = w_0 + w_1 x_n$$

- Thus

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^N (t_n - (w_0 + w_1 x_n))^2$$

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix}$$

becomes

$$\begin{aligned} \mathcal{L} &= \frac{1}{N} \sum_{n=1}^N (t_n - \mathbf{w}^\top \mathbf{x}_n)^2 = \frac{1}{N} (\mathbf{t} - \mathbf{X}\mathbf{w})^\top (\mathbf{t} - \mathbf{X}\mathbf{w}) \\ &= \frac{1}{N} (\mathbf{X}\mathbf{w} - \mathbf{t})^\top (\mathbf{X}\mathbf{w} - \mathbf{t}) \end{aligned}$$

Linear modelling with vectors

- Finding loss function's minimum

$$\mathcal{L} = \frac{1}{N}(\mathbf{X}\mathbf{w} - \mathbf{t})^\top(\mathbf{X}\mathbf{w} - \mathbf{t})$$

$$\mathcal{L} = \frac{1}{N}\mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w} - \frac{2}{N}\mathbf{w}^\top\mathbf{X}^\top\mathbf{t} + \frac{1}{N}\mathbf{t}^\top\mathbf{t}$$

- Obtain partial derivatives

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_0} \\ \frac{\partial \mathcal{L}}{\partial w_1} \end{bmatrix}$$

Linear modelling with vectors

- Obtain partial derivatives of loss function \mathcal{L}

$$\mathcal{L} = \frac{1}{N}(\mathbf{w}^\top \mathbf{X}^\top \mathbf{X} \mathbf{w} - 2\mathbf{w}^\top \mathbf{X}^\top \mathbf{t} + \mathbf{t}^\top \mathbf{t})$$

Exercise 1.3

$$\frac{1}{N} \mathbf{w}^\top \mathbf{X}^\top \mathbf{X} \mathbf{w} = w_0^2 \frac{1}{N} \left(\sum_{n=1}^N X_{n0}^2 \right) + 2w_0 w_1 \frac{1}{N} \left(\sum_{n=1}^N X_{n0} X_{n1} \right) + w_1^2 \frac{1}{N} \left(\sum_{n=1}^N X_{n1}^2 \right)$$

$$2\mathbf{w}^\top \mathbf{X}^\top \mathbf{t} = 2w_0 \frac{1}{N} \left(\sum_{n=1}^N X_{n0} t_n \right) + 2w_1 \frac{1}{N} \left(\sum_{n=1}^N X_{n1} t_n \right)$$

- Considering $X_{n0} = 1$ and $X_{n1} = x_n$, we get:

$$\mathcal{L} = w_0^2 + 2w_0 w_1 \frac{1}{N} \left(\sum_{n=1}^N x_n \right) + w_1^2 \frac{1}{N} \left(\sum_{n=1}^N x_n^2 \right) - 2w_0 \frac{1}{N} \left(\sum_{n=1}^N t_n \right) - 2w_1 \frac{1}{N} \left(\sum_{n=1}^N x_n t_n \right)$$

$t^\top t$ omitted as it's independent of w_0 or w_1

Linear modelling with vectors

- Obtain partial derivatives of loss function \mathcal{L}

$$\frac{\partial \mathcal{L}}{\partial w_0} = 2w_0 + 2w_1\bar{x} - 2\bar{t}$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = 2w_0\bar{x} + 2w_1\overline{\bar{x}^2} - 2\bar{x}\bar{t}$$

which is equivalent to our earlier derivation (slides 13 & 14):

$$\frac{\partial \mathcal{L}}{\partial w_0} = 2w_0 + 2w_1 \frac{1}{N} \left(\sum_{n=1}^N x_n \right) - \frac{2}{N} \left(\sum_{n=1}^N t_n \right)$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = 2w_1 \frac{1}{N} \left(\sum_{n=1}^N x_n^2 \right) + \frac{2}{N} \left(\sum_{n=1}^N x_n(w_0 - t_n) \right)$$

Linear modelling with vectors

- Obtain partial derivatives of loss function \mathcal{L} , with vector notation:

$$\mathcal{L} = \frac{1}{N} \mathbf{w}^\top \mathbf{X}^\top \mathbf{X} \mathbf{w} - \frac{2}{N} \mathbf{w}^\top \mathbf{X}^\top \mathbf{t} + \frac{1}{N} \mathbf{t}^\top \mathbf{t}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{2}{N} \mathbf{X}^\top \mathbf{X} \mathbf{w} - \frac{2}{N} \mathbf{X}^\top \mathbf{t} = 0$$

$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{t}$$

$f(\mathbf{w})$	$\frac{\partial f}{\partial \mathbf{w}}$
$\mathbf{w}^\top \mathbf{x}$	\mathbf{x}
$\mathbf{x}^\top \mathbf{w}$	\mathbf{x}
$\mathbf{w}^\top \mathbf{w}$	$2\mathbf{w}$
$\mathbf{w}^\top \mathbf{Cw}$	$2\mathbf{Cw}$

- To predict the winning time t_{new} for a new vector of attributes \mathbf{x}_{new} :

$$t_{new} = \hat{\mathbf{w}}^\top \mathbf{x}_{new}$$

Nonlinear response from a linear model

- Linear model is far too simplistic – it predicts winning time of -3.5 seconds in year 3000
- Linear model $f(x; \mathbf{w}) = w_0 + w_1 x$ is linear in both parameters (\mathbf{w}) and data (x)
 - Linearity in parameters is useful to obtain analytical solution we have seen earlier
- Nonlinearity in data

$$\mathbf{x}_n = \begin{bmatrix} 1 \\ x_n \\ x_n^2 \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 \end{bmatrix}$$

and determining parameters: $\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$

Nonlinear response from a linear model

- The model now becomes:

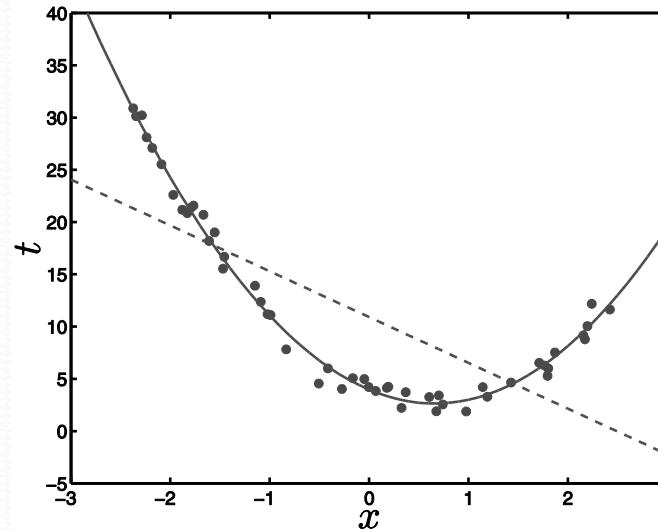
$$f(x; \mathbf{w}) = \mathbf{w}^\top \mathbf{x} = w_0 + w_1 x + w_2 x^2$$

which is linear in parameters (\mathbf{w})

- The parameters $\hat{\mathbf{w}}$ can be determined like before:

$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{t}$$

- Quadratic data
 - Quadratic fit
 - Linear fit



Nonlinear response from a linear model

- In general, data can be represented as a “polynomial” function of any order:

$$\mathbf{X} = \begin{bmatrix} x_1^0 & x_1^1 & x_1^2 & \cdots & x_1^K \\ x_2^0 & x_2^1 & x_2^2 & \cdots & x_2^K \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ x_N^0 & x_N^1 & x_N^2 & \cdots & x_N^K \end{bmatrix}$$

such that the model is: $f(x; \mathbf{w}) = \sum_{k=0}^K w_k x^k$

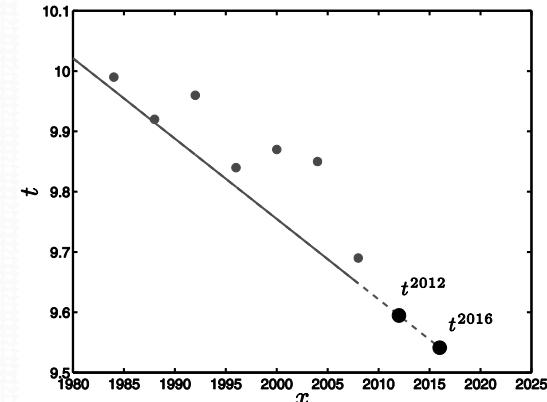
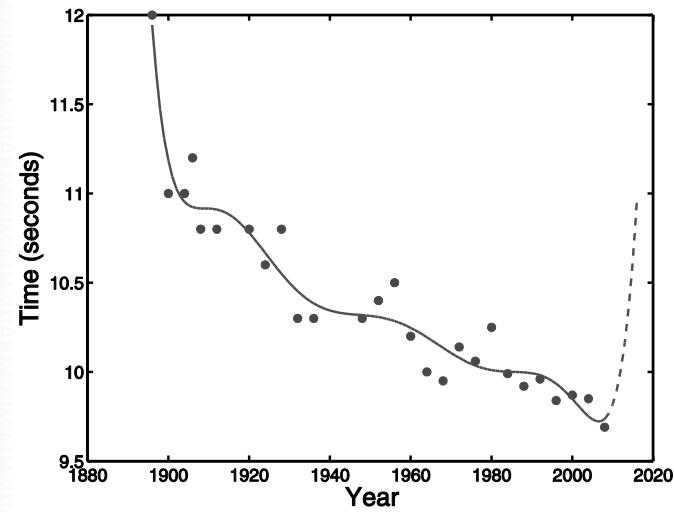
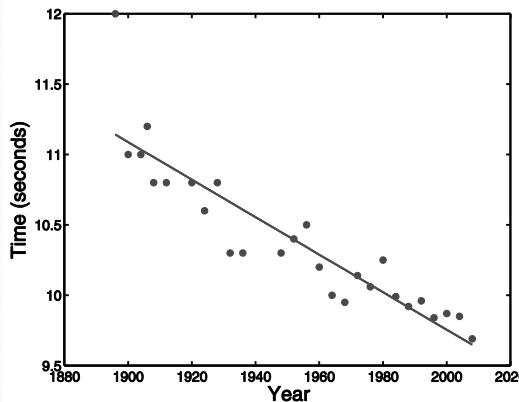
while the parameters can be computed as before:

$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{t}$$

Nonlinear response from a linear model

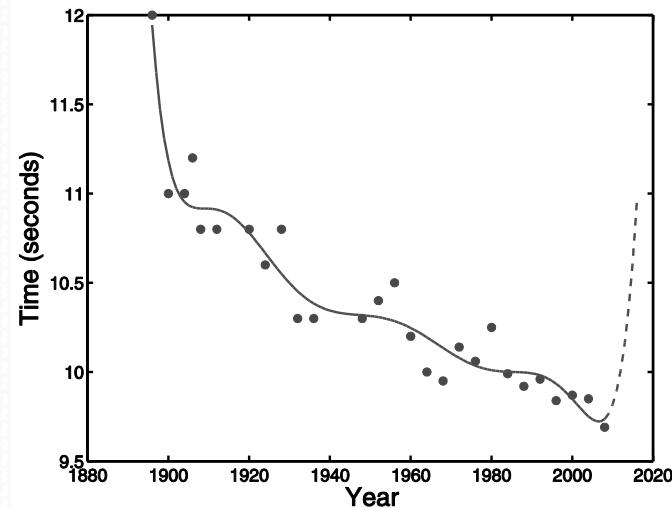
- 8th order polynomial fit on Olympic men's 100m data
- Model selection: which model is better?
 - What is “better”?
 - $\mathcal{L}^8 = 0.459$, $\mathcal{L}^1 = 1.358$
- Fitting vs prediction

$$f(x; w_0, w_1) = 36.416 - 0.013x$$



Generalization and over-fitting

- Learning aims to build model from past examples in order to predict future examples
 - What is a “good” model?
 - A good model should generalize beyond *training* examples – i.e. minimize loss on “unseen” data
 - Do we have “unseen” data available?
- Over-fitting
 - Model very closely fits to the training data (observed data)
 - Model does not generalize well to “unseen” data
- Model complexity
 - More complex models may have poor generalization

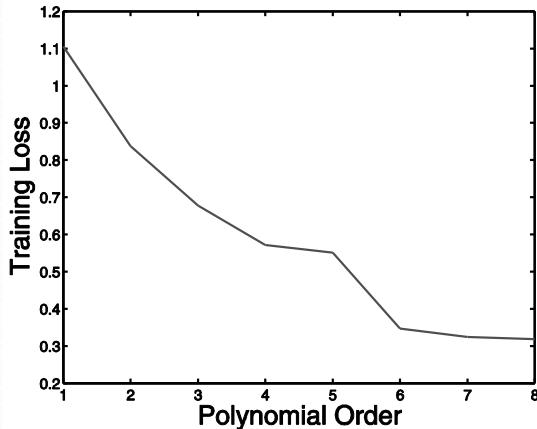


Validation data

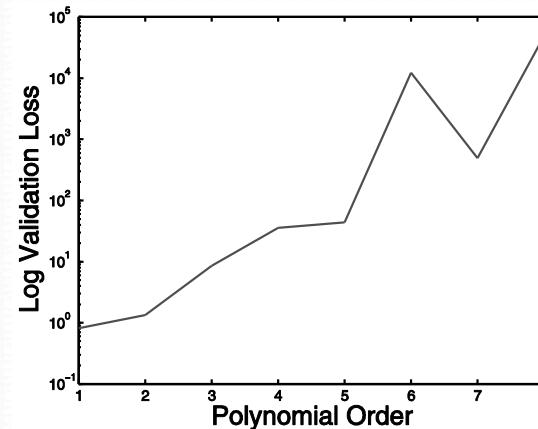
- Do we have “unseen” data available?
 - Validation data can be used to validate the predictive performance of a model
- Where to get validation data from?
 - Retain a “proportion” from the available data
 - For example: (i) train the model on pre-1980 Olympics data, (ii) ask the model to predict post-1980 winning time, and (iii) compute the validation loss on this “unseen” data.
 - Lowest validation loss could help select “optimal” model

Generalization and over-fitting

- Monotonic decrease in training loss
- Validation loss
- Model complexity
- What is a “good” model?



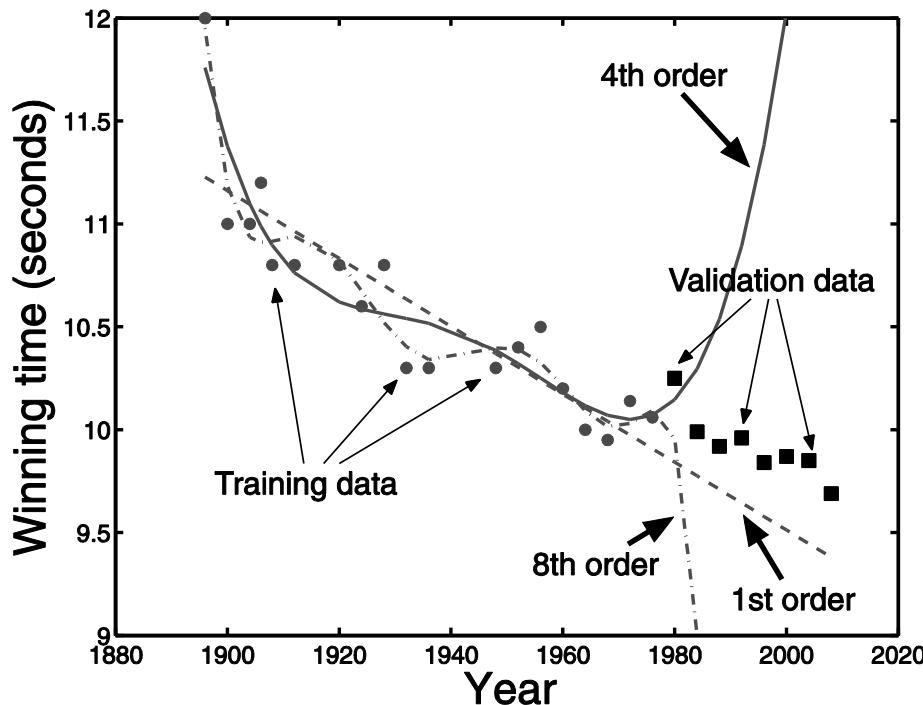
(a) Training loss for the Olympics men’s 100 m data



(b) Log validation loss for the Olympics men’s 100 m data. When using the squared loss, this is also known as the squared predictive error and measures how close the predicted values are to the true values. Note that the log loss is plotted as the value increases so rapidly

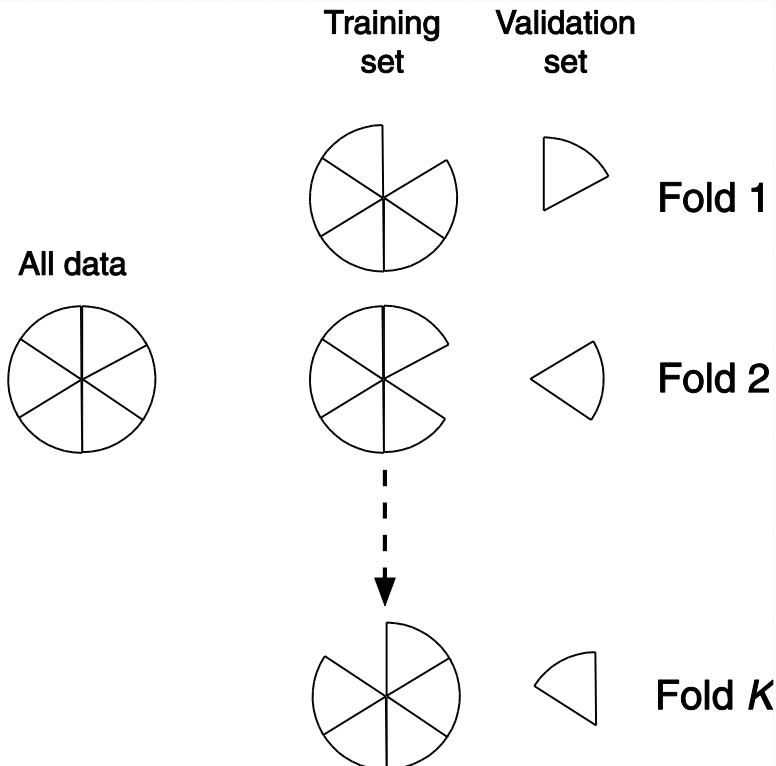
Generalization and over-fitting

- Model complexity
- What is a “good” model?



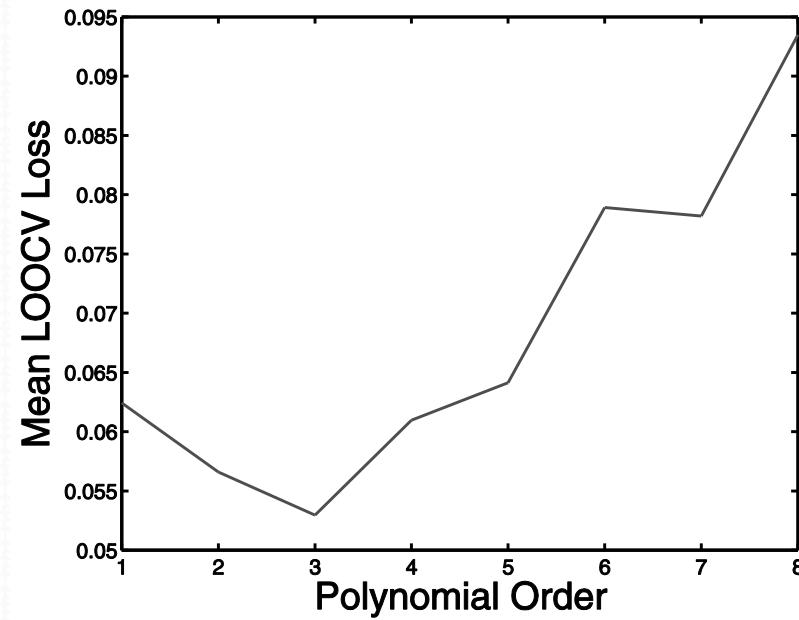
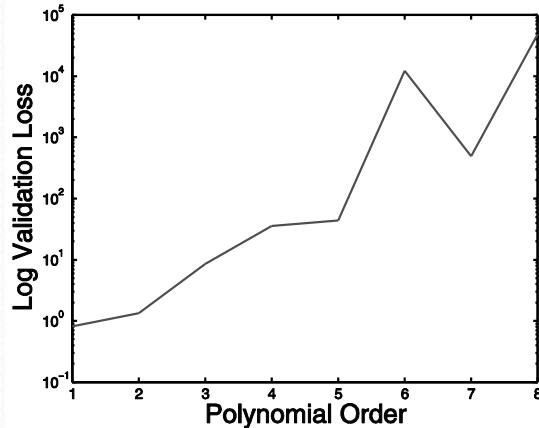
Validation data

- Validation is biased towards choice of data in validation set, particularly if data is small
- K-fold cross-validation



Cross-validation

- Average K-fold cross validation loss
- Leave one out (LOO) cross validation (LOOCV)
 - Extreme case of K fold cross validation where K = N
 - LOOCV loss $\mathcal{L}^{LOOCV} = \frac{1}{N} \sum_{n=1}^N (t_n - \hat{\mathbf{w}}_{-n}^T \mathbf{x}_n)^2$
- Cross-validation loss vs validation loss
- Model selection is difficult



Regularized least squares

- In model learning, the objective is to ensure good generalization and prevent over-fitting (i.e. avoid model complexity)
 - Regularization is a way of achieving this objective
- Let's take a very simple model:

$$f(x; \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + w_5 x^5$$

by considering that $\mathbf{w} = [0, 0, \dots, 0]^T$

- Now assign non-zero parameter values one-by-one to each component of \mathbf{w} , this gradually increases model complexity
- The model complexity (thus possibly over-fitting) increases with an increase in \mathbf{w} magnitude

Regularized least squares

- The increase in model complexity can be “controlled” by “controlling” \mathbf{w}

$$\sum_i w_i^2$$

or

$$\mathbf{w}^T \mathbf{w}$$

- Thus, a model should aim to minimize loss while simultaneously penalizing over-complexity

$$\mathcal{L}' = \mathcal{L} + \lambda \mathbf{w}^T \mathbf{w}$$

$$\mathcal{L}' = \frac{1}{N} \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} - \frac{2}{N} \mathbf{w}^T \mathbf{X}^T \mathbf{t} + \frac{1}{N} \mathbf{t}^T \mathbf{t} + \lambda \mathbf{w}^T \mathbf{w}$$

Regularized least squares

- Finding loss function's minimum

$$\mathcal{L}' = \frac{1}{N} \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} - \frac{2}{N} \mathbf{w}^T \mathbf{X}^T \mathbf{t} + \frac{1}{N} \mathbf{t}^T \mathbf{t} + \lambda \mathbf{w}^T \mathbf{w}$$

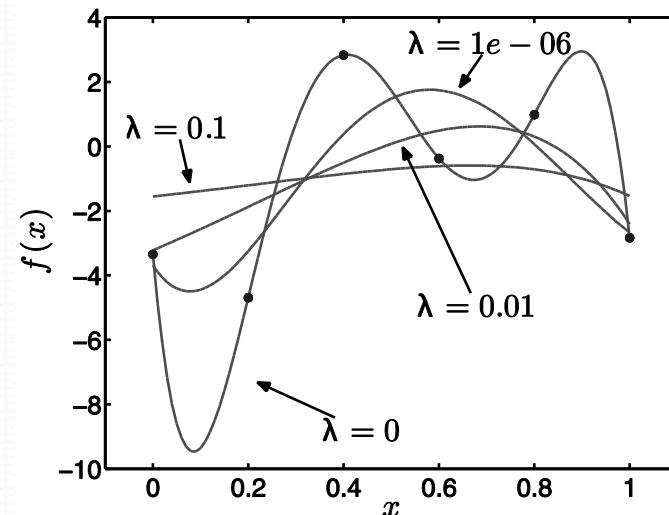
$$\frac{\partial \mathcal{L}'}{\partial \mathbf{w}} = \frac{2}{N} \mathbf{X}^T \mathbf{X} \mathbf{w} - \frac{2}{N} \mathbf{X}^T \mathbf{t} + 2\lambda \mathbf{w} = 0$$

- The parameters for a regularized model of the data can be obtained as:

$$\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X} + N\lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{t}$$

- How to determine λ ?
- Over-fitting/generalization trade-off

$f(\mathbf{w})$	$\frac{\partial f}{\partial \mathbf{w}}$
$\mathbf{w}^T \mathbf{x}$	\mathbf{x}
$\mathbf{x}^T \mathbf{w}$	\mathbf{x}
$\mathbf{w}^T \mathbf{w}$	$2\mathbf{w}$
$\mathbf{w}^T \mathbf{C} \mathbf{w}$	$2\mathbf{C}\mathbf{w}$



Summary

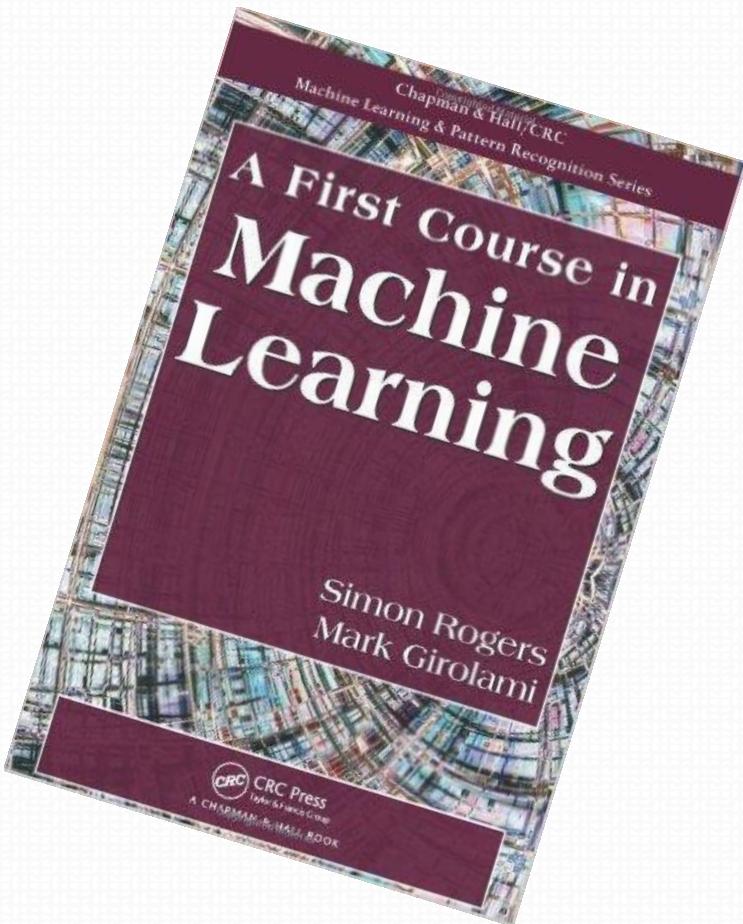
- Linear model can learn to make predictions
- Loss estimated from training samples isn't reliable
- Generalization is desirable
- Model complexity typically leads to overfitting
- Cross-validation error is reflective of generalization
- Non-linear predictions can be estimated from linear model

Exercise (ungraded)

- Book (FCML) – exercise 1.1
- Book (FCML) – exercise 1.2
- Book (FCML) – exercise 1.3
- Book (FCML) – exercise 1.6 (use vector notation and MATLAB)
- Book (FCML) – exercise 1.7
- Book (FCML) – exercise 1.8
- Book (FCML) – exercise 1.10

C, R, E, D₂, I₁, T₁, S₁

₃ ₂ ₁ ₁



Author's material
(Simon Rogers)



Thank You

Machine Learning, Machine Learning (extended)

3 - Supervised Learning:
Linear Modelling by Maximum Likelihood
Kashif Rajpoot

k.m.rajpoot@cs.bham.ac.uk

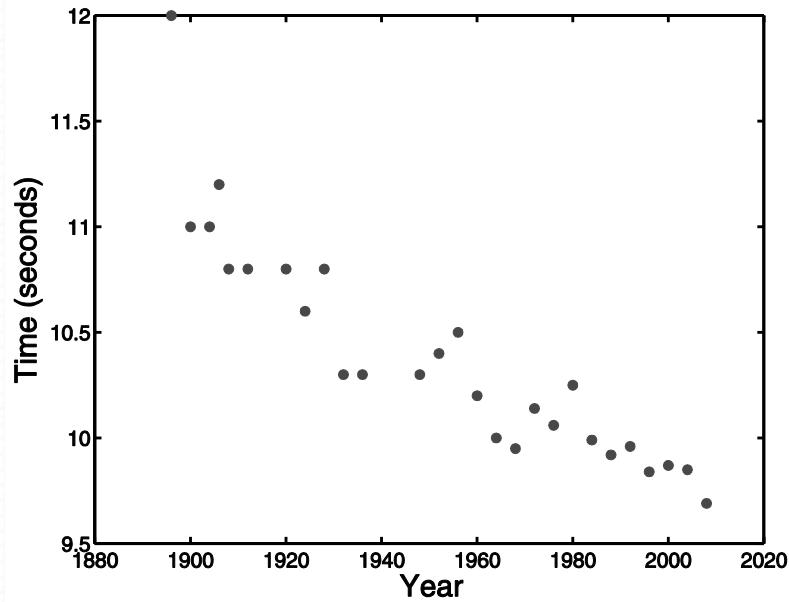
School of Computer Science
University of Birmingham

Outline

- Linear modelling
- Error = noise
- Thinking generatively
- Likelihood
- Maximum likelihood
- Model complexity
- Bias-variance tradeoff

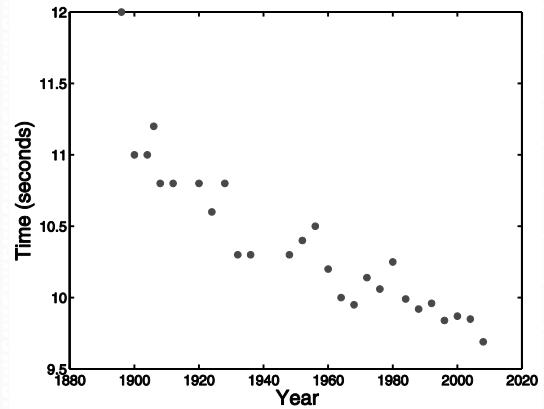
Linear modelling

- One of the most straightforward learning problems
 - Learn a linear function between attributes and responses
- Is there a functional dependence between Olympics year and 100m winning time?
 - Draw a line?
- Can we predict winning time for future games?



Linear modelling

- Learner model/function
 - Maps input attributes to output response
- Let's consider we can predict time $t = f(x)$
 - x ?
 - t ?
- Training samples
 - N attribute-response pairs $(x_1, t_1), (x_2, t_2), \dots (x_N, t_N)$



Linear modelling

- Linear modelling by minimizing loss

$$t_n = \hat{w}_0 + \hat{w}_1 x_n$$

where the model parameters are estimated from Olympics data

$$\hat{w}_1 = \frac{\bar{x}t - \bar{x}\bar{t}}{\bar{x^2} - (\bar{x})^2}$$

$$\hat{w}_0 = \bar{t} - \hat{w}_1 \bar{x}$$

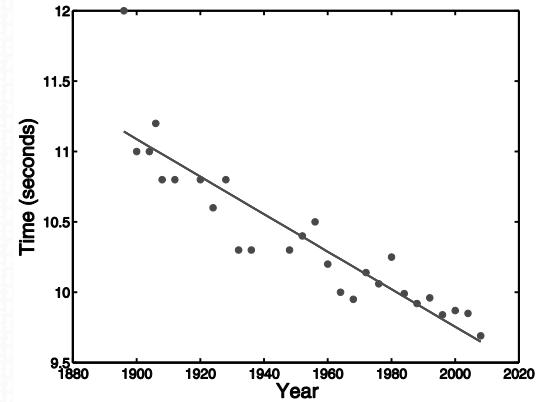
- With the vector notation

$$t_n = \hat{\mathbf{w}}^T \mathbf{x}_n$$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \text{ and } \mathbf{x}_n = \begin{bmatrix} 1 \\ x_n \end{bmatrix}$$

while model parameters are estimated as: $\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t}$

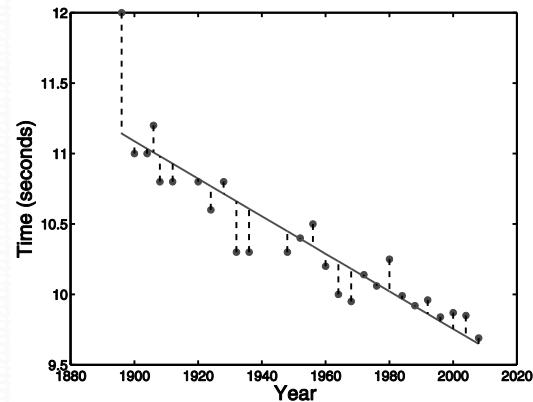
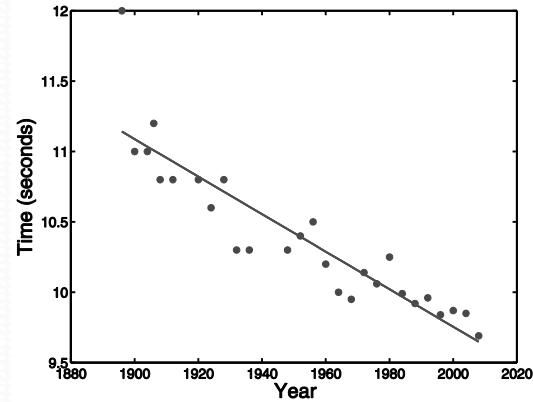
$$f(x; w_0, w_1) = 36.416 - 0.013x$$



$$\mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix}$$

Linear modelling

- Linear modelling by minimizing loss
 - Model shows to capture trend in data observations
 - Model fails to explain each data observation correctly (i.e. error)
- Let's recall our assumptions
 - There is a relationship between Olympics year and winning time
 - This relationship is linear
 - This relationship will hold in future
- Are these good assumptions?
- Still, ignoring the error is not right
 - Let's consider error as noise and model it

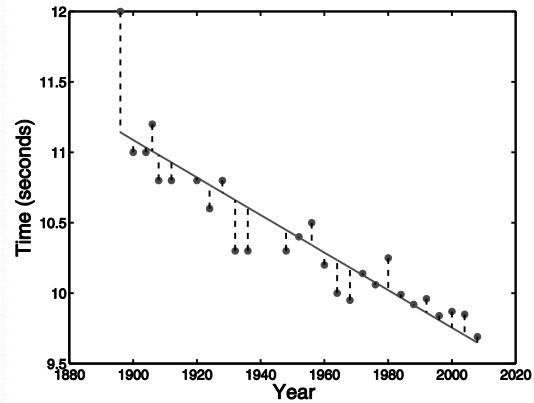
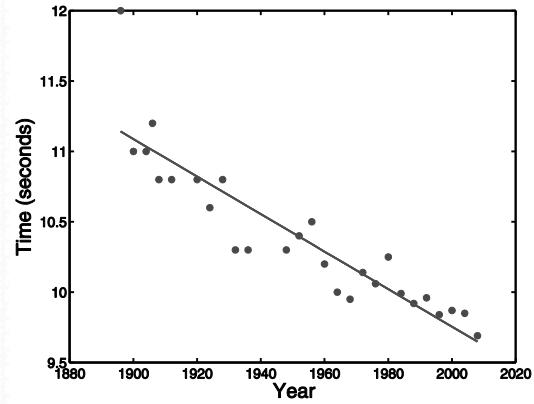


Linear modelling

- Approaches to linear modelling
 - Minimize the loss: minimize the squared error between model predictions and training observations
 - Maximize the likelihood: maximize the likelihood of match between model predictions and training observations
 - Build a model to generate data
 - Explicitly model the noise (i.e. the error between model and observations)

Thinking generatively

- Let's recall that our assumptions are weak
 - The process generating this data is very complex
- Still, can we try to build a model that can generate such data?
- Generative modelling: build a model to generate data that *looks like* data observations
 - $t_n = \mathbf{w}^T \mathbf{x}_n$
 - Error?
 - Modelling error



Refresher: Probability

- Random variable (X, Y)
- Probability of tossing a coin (head=1, tail=0)
 - $P(X = 0) = 0.5$
 - $P(X = 1) = 0.5$

Refresher: Conditional probability

- When the outcome of an event is affected (i.e. conditioned) by the outcome of another event
- For example: we toss a coin and then tell the result
 - $P(X = 1)$: probability of coin landing head
 - $P(X = 0)$: probability of coin landing tail
 - $P(Y = 1)$: probability of telling coin landed head
 - $P(Y = 0)$: probability of telling coin landed tail
- $P(Y = y|X = x)$: probability of telling coin landed y , given that coin has landed x

Refresher: Conditional probability

- If we always tell the true outcome:

- $P(Y = 1|X = 1) = ?$
- $P(Y = 1|X = 1) = 1$
- $P(Y = 0|X = 0) = ?$
- $P(Y = 0|X = 0) = 1$
- $P(Y = 0|X = 1) = ?$
- $P(Y = 0|X = 1) = 0$
- $P(Y = 1|X = 0) = ?$
- $P(Y = 1|X = 0) = 0$

Refresher: Conditional probability

- If we tell the true head outcome only 80% times:
 - $P(Y = 1|X = 1) = ?$
 - $P(Y = 1|X = 1) = 0.8$
 - $P(Y = 0|X = 0) = ?$
 - $P(Y = 0|X = 0) = 1$
 - $P(Y = 0|X = 1) = ?$
 - $P(Y = 0|X = 1) = 0.2$
 - $P(Y = 1|X = 0) = ?$
 - $P(Y = 1|X = 0) = 0$

Refresher: Joint probability

- What is the probability that the coin lands heads and we say heads?
 - This is joint probability (i.e. probability of two or more variables)
 - $P(Y = y, X = x)$
- In case of no dependence between variables:
 - $P(Y = y, X = x) = P(Y = y)P(X = x)$
- In case of dependence between variables:
 - $P(Y = y, X = x) = P(Y = y|X = x)P(X = x)$, or
 - $P(Y = y, X = x) = P(X = x|Y = y)P(Y = y)$

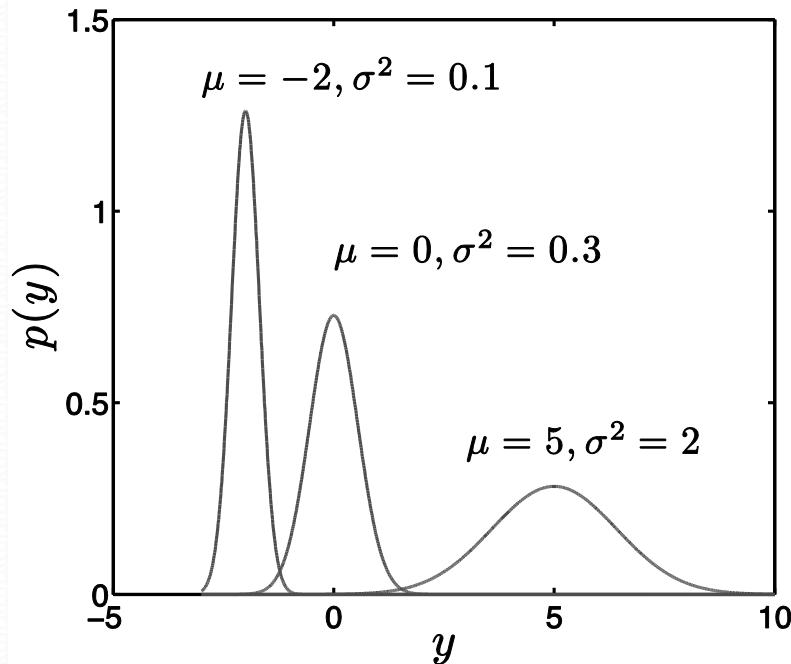
Refresher: Joint probability

- What is the probability that the coin lands heads and we say heads?
 - $P(Y = 1, X = 1) = P(Y = 1|X = 1)P(X = 1) = ?$
 - $P(Y = 1, X = 1) = P(Y = 1|X = 1)P(X = 1) = 0.8 \times 0.5 = 0.4$
- Other joint probabilities
 - $P(Y = 0, X = 1) = P(Y = 0|X = 1)P(X = 1) = ?$
 - $P(Y = 0, X = 1) = P(Y = 0|X = 1)P(X = 1) = 0.2 \times 0.5 = 0.1$
 - $P(Y = 1, X = 0) = P(Y = 1|X = 0)P(X = 0) = ?$
 - $P(Y = 1, X = 0) = P(Y = 1|X = 0)P(X = 0) = 0 \times 0.5 = 0$
 - $P(Y = 0, X = 0) = P(Y = 0|X = 0)P(X = 0) = ?$
 - $P(Y = 0, X = 0) = P(Y = 0|X = 0)P(X = 0) = 1 \times 0.5 = 0.5$

Refresher: Gaussian pdf

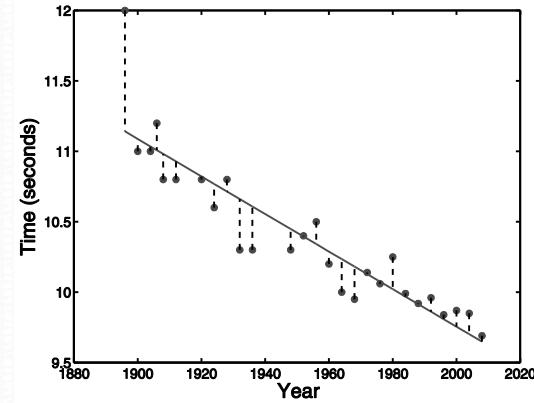
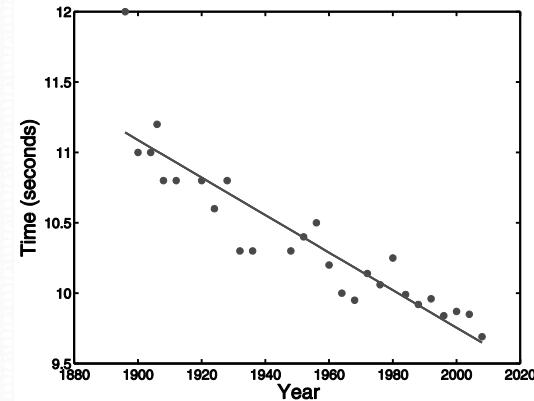
- Gaussian (or normal) probability density function

$$p(y|\mu, \sigma^2) = \mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(y - \mu)^2\right\}$$



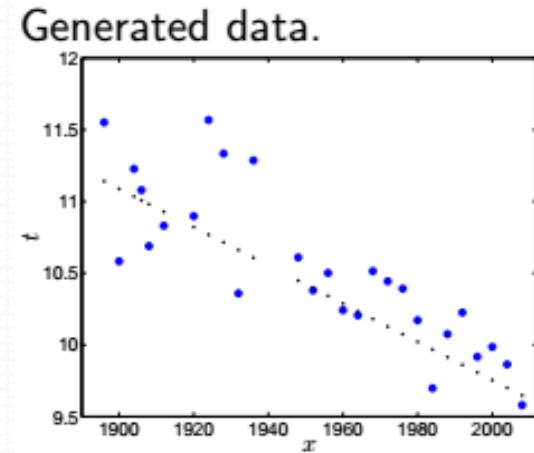
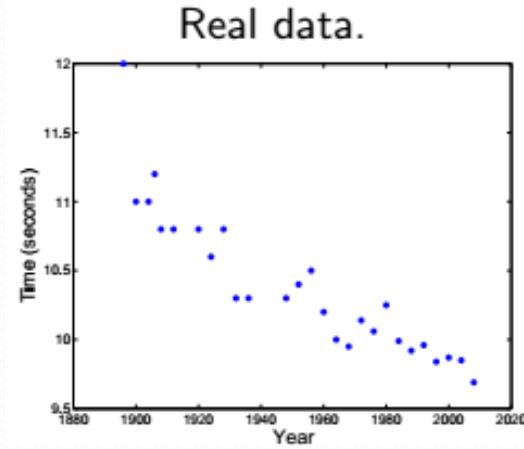
Thinking generatively

- Generative modelling: build a model to generate data that looks like data observations
 - $t_n = \mathbf{w}^T \mathbf{x}_n$
 - Error?
- Generative model with noise considerations:
 - $t_n = \mathbf{w}^T \mathbf{x}_n + \varepsilon_n$
 - Additive noise?
- Noise (ε_n)
 - Is it a random variable?
 - Is it discrete or continuous?
 - What pdf for ε_n ?



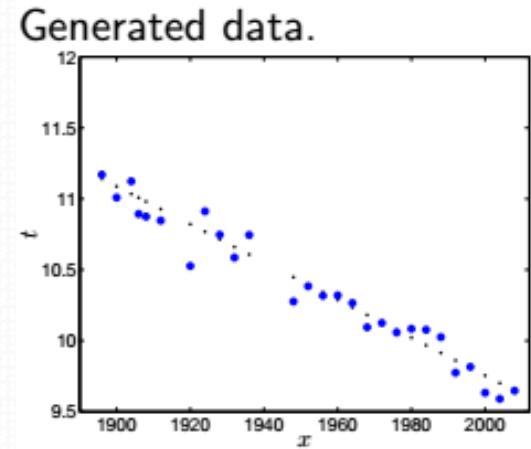
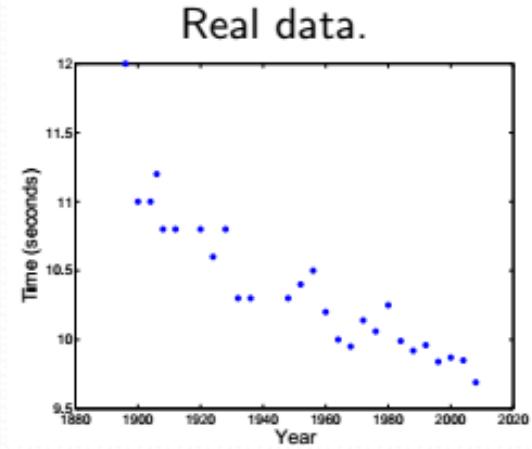
Thinking generatively

- Probability density of ε_n
 $p(\varepsilon_1, \varepsilon_2, \dots, \varepsilon_N) = \prod_{n=1}^N p(\varepsilon_n)$
i.e. each ε_n is independent
- Let's use a Gaussian density for
 $p(\varepsilon_n) = \mathcal{N}(\mu, \sigma^2) = \mathcal{N}(0, 0.05)$
- Generative model with noise considerations:
 - $t_n = \mathbf{w}^T \mathbf{x}_n + \varepsilon_n$
- Deterministic component (i.e. trend)
- Random component (i.e. noise)



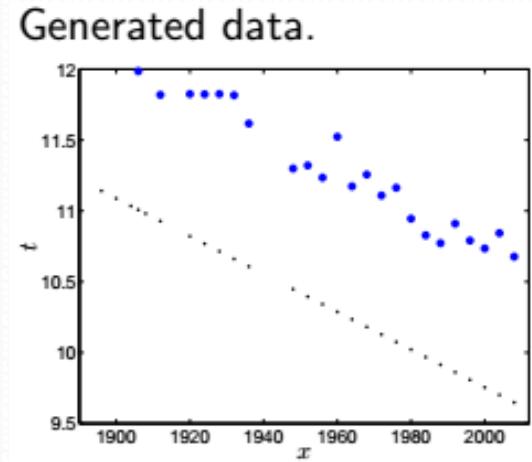
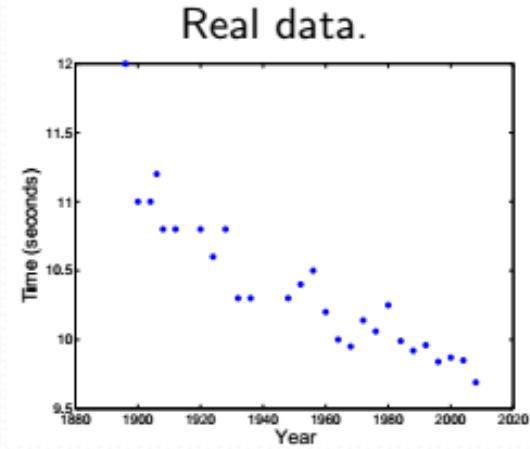
Thinking generatively

- Probability density of ε_n
 $p(\varepsilon_1, \varepsilon_2, \dots, \varepsilon_N) = \prod_{n=1}^N p(\varepsilon_n)$
i.e. each ε_n is independent
- Let's use a Gaussian density for
 $p(\varepsilon_n) = \mathcal{N}(\mu, \sigma^2) = \mathcal{N}(0, 0.01)$
- Generative model with noise considerations:
 - $t_n = \mathbf{w}^T \mathbf{x}_n + \varepsilon_n$
- Deterministic component (i.e. trend)
- Random component (i.e. noise)



Thinking generatively

- Probability density of ε_n
 $p(\varepsilon_1, \varepsilon_2, \dots, \varepsilon_N) = \prod_{n=1}^N p(\varepsilon_n)$
i.e. each ε_n is independent
- Let's use a Gaussian density for
 $p(\varepsilon_n) = \mathcal{N}(\mu, \sigma^2) = \mathcal{N}(1, 0.01)$
- Generative model with noise considerations:
 - $t_n = \mathbf{w}^T \mathbf{x}_n + \varepsilon_n$
- Deterministic component (i.e. trend)
- Random component (i.e. noise)



Thinking generatively

- Generative model with noise considerations:
 - $t_n = \mathbf{w}^T \mathbf{x}_n + \varepsilon_n$
 - $t_n = f(\mathbf{x}_n; \mathbf{w}) + \varepsilon_n$, where $p(\varepsilon_n) = \mathcal{N}(0, \sigma^2)$
 - Model is now determined not only by \mathbf{w} but also σ^2
- t_n can now be considered a random variable itself, due to addition of ε_n
 - i.e. for a given \mathbf{x}_n , t_n is not a single fixed value but rather is drawn out from a pdf
 - Thus finding \mathbf{w} and σ^2 by minimizing loss (with least squares approach) is not possible

Thinking generatively

- The probability density of t_n is:

$$p(t_n | \mathbf{x}_n, \mathbf{w}, \sigma^2) = \mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2)$$

- Let's recall $p(\varepsilon_n) = \mathcal{N}(0, \sigma^2)$

$$y = a + z$$

$$p(z) = \mathcal{N}(m, s)$$

$$p(y) = \mathcal{N}(m + a, s)$$

- Note that $\mathbf{w}^T \mathbf{x}_n$ determines the mean (i.e. trend) and σ^2 determines variance (i.e. noise)

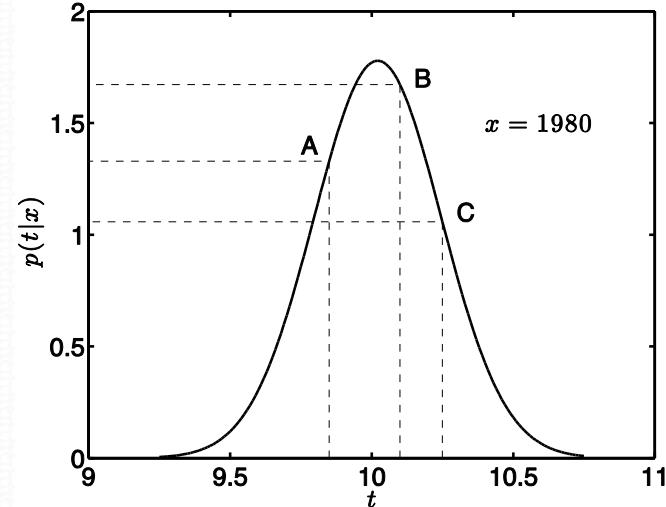
Thinking generatively

- Let's use this generative model to look at pdf for year 1980

$$p(t_n | \mathbf{x}_n = [1, 1980]^T, \mathbf{w} = [36.416, -0.0133]^T, \sigma^2 = 0.05)$$

This generates a Gaussian pdf $\mathcal{N}(\mathbf{w}^T \mathbf{x}_n = 10.02, \sigma^2 = 0.05)$

- Height of the curve corresponds to how likely it is to observe a particular t
- A, B, or C – more likely?
- Likelihood at $t_n = 10.25$?
- Can we determine \mathbf{w} and σ^2 that maximize likelihood of observed data t_n with generated data?



Likelihood

- For each input-response pair (\mathbf{x}_n, t_n) , we have a Gaussian likelihood: $p(t_n | \mathbf{x}_n, \mathbf{w}, \sigma^2)$
- Maximize the likelihood of matching all observed responses (t_1, t_2, \dots, t_N) conditioned on the observed data $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$ and model parameters (\mathbf{w}, σ^2)
$$p(t_1, t_2, \dots, t_N | \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N, \mathbf{w}, \sigma^2) = p(\mathbf{t} | \mathbf{X}, \mathbf{w}, \sigma^2)$$
- Let's recall that $p(\varepsilon_1, \varepsilon_2, \dots, \varepsilon_N) = \prod_{n=1}^N p(\varepsilon_n)$
i.e. each ε_n is independent
- Thus, likelihood can be estimated as:

$$L = p(\mathbf{t} | \mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{n=1}^N \mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2)$$

Maximum likelihood

- Likelihood estimate

$$L = p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{n=1}^N \mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2)$$

- How to find \mathbf{w} and σ^2 ?
- Find “best” \mathbf{w} and σ^2 that maximize likelihood L

$$\underset{\mathbf{w}, \sigma^2}{\operatorname{argmax}} L$$

- It’s mathematically convenient if we, instead, maximize the log of likelihood L

$$\underset{\mathbf{w}, \sigma^2}{\operatorname{argmax}} \log(L)$$

- Model parameters (\mathbf{w}, σ^2) that maximize log likelihood ($\log(L)$) also maximize likelihood (L)

Maximum likelihood

- Likelihood estimate

$$L = p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{n=1}^N \mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2)$$

- Log likelihood estimate

$$\log(L) = \log \left[\prod_{n=1}^N \mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2) \right] = \sum_{n=1}^N \log[\mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2)]$$

- Let's recall

$$\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left\{ -\frac{1}{2\sigma^2} (y - \mu)^2 \right\}$$

so

$$\mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left\{ -\frac{1}{2\sigma^2} (t_n - \mathbf{w}^T \mathbf{x}_n)^2 \right\}$$

Maximum likelihood

$$\log(L) = \sum_{n=1}^N \log[\mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2)]$$

- Considering that

$$\mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(t_n - \mathbf{w}^T \mathbf{x}_n)^2\right\}$$

we get

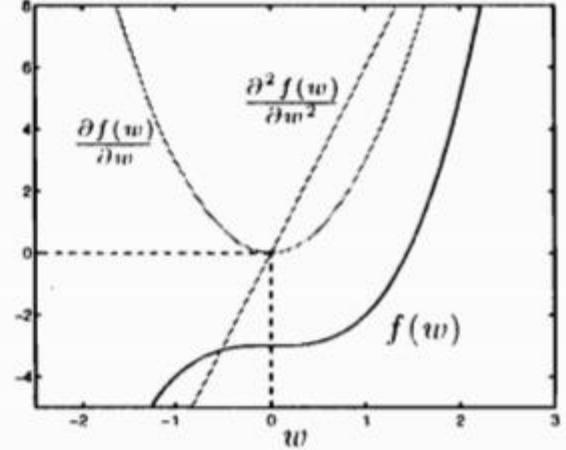
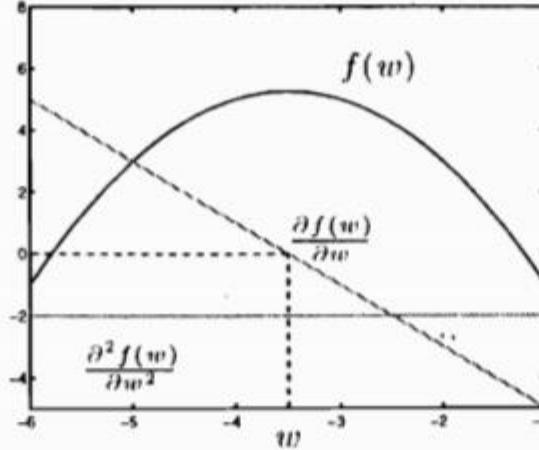
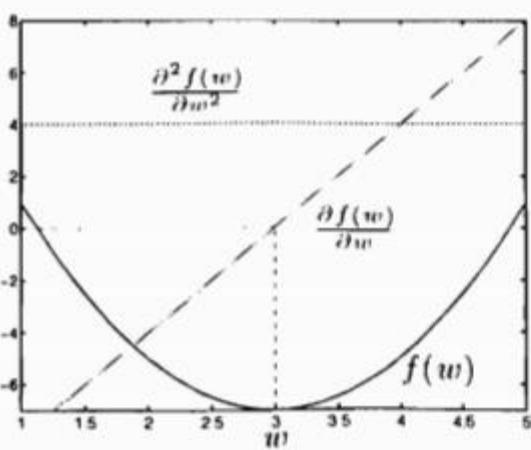
$$\log(L) = \sum_{n=1}^N \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(t_n - \mathbf{w}^T \mathbf{x}_n)^2\right\}\right]$$

which can be simplified to:

$$\log(L) = -\frac{N}{2} \log(2\pi) - N \log(\sigma) - \frac{1}{2\sigma^2} \sum_{n=1}^N (t_n - \mathbf{w}^T \mathbf{x}_n)^2$$

Finding function's maximum

- A function's minimum or maximum can be determined where the 1st derivative is zero
 - Local maxima
 - Local minima



Maximum likelihood

- Finding function's maximum

$$\frac{\partial \log(L)}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} \left[-\frac{N}{2} \log(2\pi) - N \log(\sigma) - \frac{1}{2\sigma^2} \sum_{n=1}^N (t_n - \mathbf{w}^T \mathbf{x}_n)^2 \right]$$

$$\frac{\partial \log(L)}{\partial \mathbf{w}} = \frac{1}{\sigma^2} \sum_{n=1}^N \mathbf{x}_n (t_n - \mathbf{x}_n^T \mathbf{w})$$

$$\mathbf{x}_n^T \mathbf{w} = \mathbf{w}^T \mathbf{x}_n$$

$$\frac{\partial \log(L)}{\partial \mathbf{w}} = \frac{1}{\sigma^2} \sum_{n=1}^N (\mathbf{x}_n t_n - \mathbf{x}_n \mathbf{x}_n^T \mathbf{w}) = \mathbf{0}$$

$$\frac{\partial \log(L)}{\partial \mathbf{w}} = \frac{1}{\sigma^2} (\mathbf{X}^T \mathbf{t} - \mathbf{X}^T \mathbf{X} \mathbf{w}) = \mathbf{0}$$

$$\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t}$$

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix}$$

- It's exactly the same solution as from least squares
 - Minimizing squared loss = maximizing likelihood

Maximum likelihood

- Finding function's maximum

$$\frac{\partial \log(L)}{\partial \sigma} = \frac{\partial}{\partial \sigma} \left[-\frac{N}{2} \log(2\pi) - N \log(\sigma) - \frac{1}{2\sigma^2} \sum_{n=1}^N (t_n - \mathbf{w}^T \mathbf{x}_n)^2 \right]$$

$$\frac{\partial \log(L)}{\partial \sigma} = -\frac{N}{\sigma} + \frac{1}{\sigma^3} \sum_{n=1}^N (t_n - \mathbf{x}_n^T \hat{\mathbf{w}})^2 = 0$$

$$\mathbf{x}_n^T \mathbf{w} = \mathbf{w}^T \mathbf{x}_n$$

$$\widehat{\sigma^2} = \frac{1}{N} \sum_{n=1}^N (t_n - \mathbf{x}_n^T \hat{\mathbf{w}})^2$$

$$\widehat{\sigma^2} = \frac{1}{N} (\mathbf{t} - \mathbf{X} \hat{\mathbf{w}})^T (\mathbf{t} - \mathbf{X} \hat{\mathbf{w}})$$

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix}$$

Model complexity

$$\log(L) = -\frac{N}{2} \log(2\pi) - N \log(\sigma) - \frac{1}{2\sigma^2} \sum_{n=1}^N (t_n - \mathbf{w}^T \mathbf{x}_n)^2$$

- Consider that:

$$\widehat{\sigma^2} = \frac{1}{N} \sum_{n=1}^N (t_n - \mathbf{x}_n^T \widehat{\mathbf{w}})^2$$

so the log likelihood estimate at maximum is:

$$\log(L) = -\frac{N}{2} \log(2\pi) - N \log(\sigma) - \frac{1}{2\widehat{\sigma^2}} N \widehat{\sigma^2}$$

$$\log(L) = -\frac{N}{2} (1 + \log(2\pi)) - \frac{N}{2} \log(\widehat{\sigma^2})$$

Model complexity

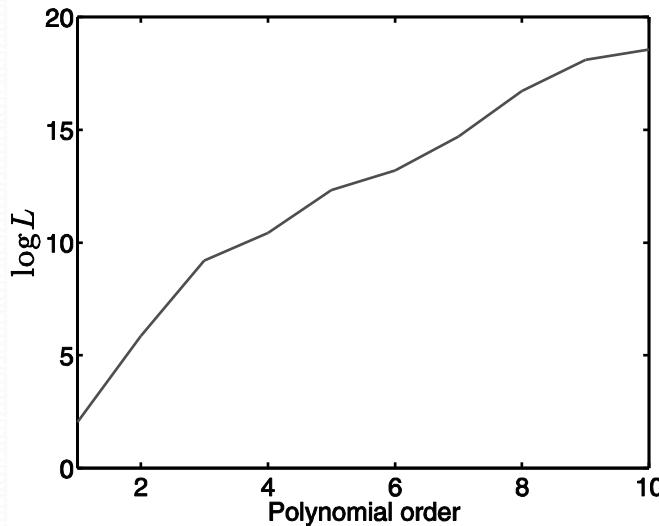
- Log likelihood estimate at maximum:

$$\log(L) = -\frac{N}{2}(1 + \log(2\pi)) - \frac{N}{2}\log(\widehat{\sigma^2})$$

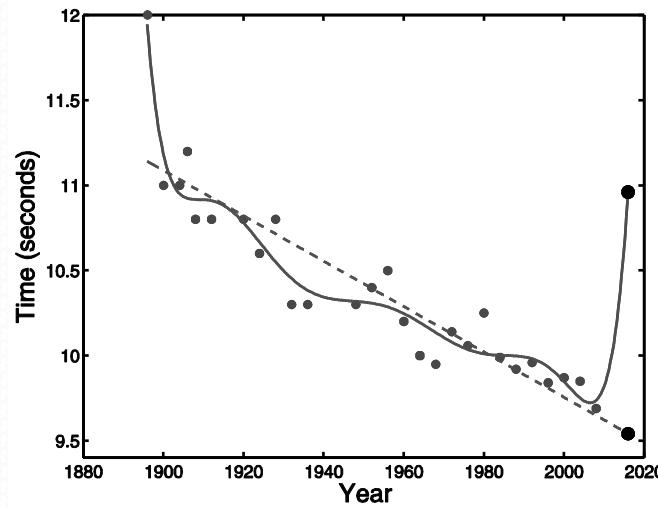
- Decrease in $\widehat{\sigma^2}$ will result in increase in log likelihood
- Note that $\mathbf{w}^T \mathbf{x}_n$ determines the mean (i.e. trend) and σ^2 determines variance (i.e. noise)
- Decrease in σ^2 will result in those values of model parameters \mathbf{w} that capture observations closely
 - i.e. over-fitting and poor generalization

Model complexity

- Modelling Olympics data again..
- Higher model complexity results in maximum likelihood
 - Generalization and over-fitting tradeoff



(a) Increase in log likelihood as the polynomial order increases



(b) 1st and 8th order polynomial functions fitted to the Olympics men's 100 m data. Large dark circles correspond to predictions for the 2016 Olympics

Bias-variance tradeoff

- Generalization and over-fitting tradeoff
- Error between predicted values and observed values can be considered to consist of two components – bias and model variance:

$$\bar{\mathcal{M}} = \mathcal{B}^2 + \mathcal{V}$$

- \mathcal{B}^2 – bias: systematic mismatch between our model and the actual process that generated data
 - Decrease in bias \mathcal{B}^2 to control $\bar{\mathcal{M}}$
 - Complex models lead to decrease in bias
- \mathcal{V} – variance: more complex model has higher variance

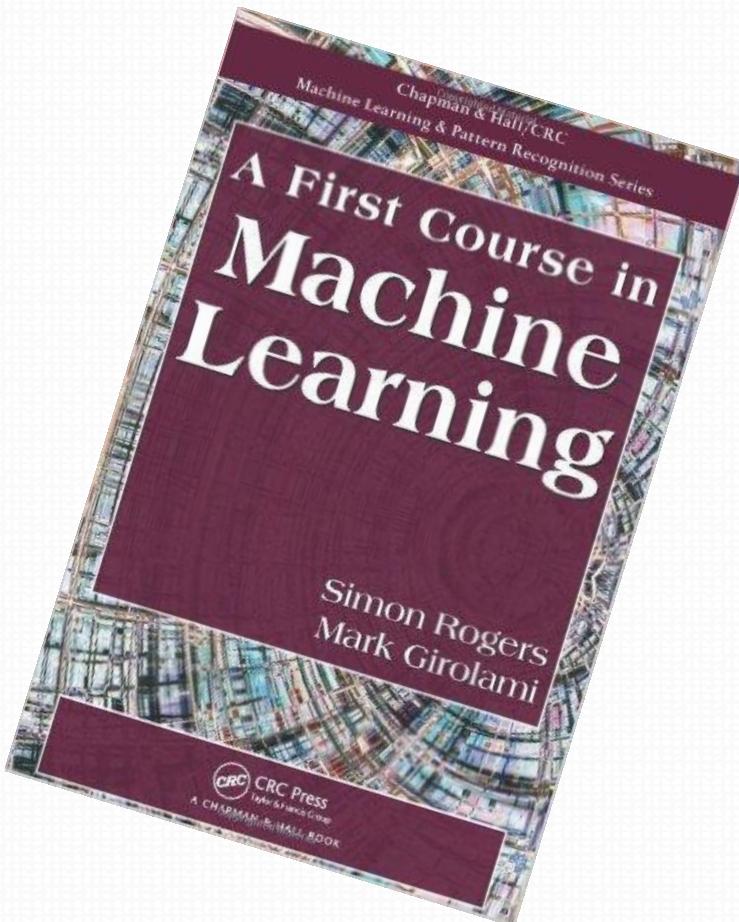
Summary

- Explicit modelling of error as noise
- Noise modelling as Gaussian random variable
- Likelihood of model predictions and observed data
- Maximizing the likelihood
- Generalization and over-fitting tradeoff

Exercise (ungraded)

- Book (FCML) – exercise 2.1
- Book (FCML) – exercise 2.2 (refresher: probability)
- Book (FCML) – exercise 2.5 (refresher: probability)
- Book (FCML) – MATLAB code – olymplike.m
- Book (FCML) – MATLAB code – genolymp.m

CREDITS



Author's material
(Simon Rogers)



Thank You

Machine Learning, Machine Learning (extended)

**4 – Supervised Learning:
Bayesian Classification**

Kashif Rajpoot

k.m.rajpoot@cs.bham.ac.uk

**School of Computer Science
University of Birmingham**

Outline

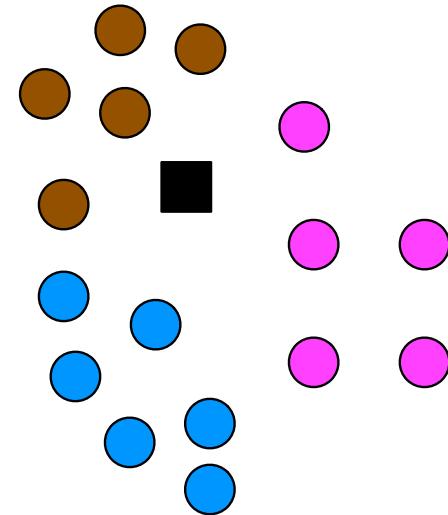
- Supervised learning
- Classification
 - Probabilistic vs non-probabilistic
 - Generative vs discriminative
- Refresher: probability
- Bayesian classification
- Naïve Bayes classification
- Gaussian classification

Supervised learning

- Regression
 - Minimised loss (e.g. least squares)
 - Maximum likelihood
- Classification
 - Generative (e.g. Bayesian)
 - Instance-based (e.g. k-NN)
 - Discriminative (e.g. SVM)

Classification

- A set of N objects with attributes (usually vector) \mathbf{x}_n
- Each object has an associated target label t_n
- Binary classification
 $t_n \in \{0,1\}$ or $t_n \in \{-1,1\}$
- Multi-class classification
 $t_n \in \{1,2,\dots,C\}$
- Classifier learns from $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ and t_1, t_2, \dots, t_N so that it can later classify \mathbf{x}_{new}



Probabilistic vs non-probabilistic classification

- Probabilistic classifiers produce a probability of class membership

$$P(t_{\text{new}} = k | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$$

$$P(t_{\text{new}} = 1 | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$$

$$P(t_{\text{new}} = 0 | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$$

- Non-probabilistic classifiers produce a hard assignment

$$t_{\text{new}} = 1 \text{ or } t_{\text{new}} = 0$$

Probabilistic vs non-probabilistic classification

- Probabilities provide us with more information
 - $P(t_{new} = 1) = 0.6$ is more useful than $t_{new} = 1$
 - Confidence level
- Particularly important where cost of misclassification is high and imbalanced
 - Diagnosis: telling a diseased person they are healthy is much worse than telling a healthy person they are diseased

Generative vs discriminative classification

- Generative classifiers generate a model for each class, based on training samples available
 - Data in each class can be seen as generated by some model
 - For new test samples, they assign these samples to the class that suits best (e.g. by probability measure)
- In contrast, discriminative classifiers attempt to explicitly define the decision boundary that separates the classes
 - Intuitively, these methods are for binary class problems but can be extended to multi-class problems

Bayesian classifier

- A classifier built on Bayes rule
 - Builds a probabilistic model of the data, embedding prior knowledge
 - Allows us to extract prior knowledge from observed data
- Generative approach
 - Builds a model from training objects
 - Any new objects can be classified based on the probabilistic model specification

Refresher: probability

- Conditional probability
- Joint probability
- Marginal probability
- Bayes rule

Refresher: Conditional probability

- When the outcome of an event is affected (i.e. conditioned) by the outcome of another event
- For example: we toss a coin and then tell the result
 - $P(X = 1)$: probability of coin landing head
 - $P(X = 0)$: probability of coin landing tail
 - $P(Y = 1)$: probability of telling coin landed head
 - $P(Y = 0)$: probability of telling coin landed tail
- $P(Y = y|X = x)$: probability of telling coin landed y , given that coin has landed x

Refresher: Conditional probability

- If we always tell the true outcome:

- $P(Y = 1|X = 1) = ?$
- $P(Y = 1|X = 1) = 1$
- $P(Y = 0|X = 0) = ?$
- $P(Y = 0|X = 0) = 1$
- $P(Y = 0|X = 1) = ?$
- $P(Y = 0|X = 1) = 0$
- $P(Y = 1|X = 0) = ?$
- $P(Y = 1|X = 0) = 0$

Refresher: Conditional probability

- If we tell the true head outcome only 80% times:
 - $P(Y = 1|X = 1) = ?$
 - $P(Y = 1|X = 1) = 0.8$
 - $P(Y = 0|X = 0) = ?$
 - $P(Y = 0|X = 0) = 1$
 - $P(Y = 0|X = 1) = ?$
 - $P(Y = 0|X = 1) = 0.2$
 - $P(Y = 1|X = 0) = ?$
 - $P(Y = 1|X = 0) = 0$

Refresher: Joint probability

- What is the probability that the coin lands heads and we say heads?
 - This is joint probability (i.e. probability of two or more variables)
 - $P(Y = y, X = x)$
- In case of no dependence between variables:
 - $P(Y = y, X = x) = P(Y = y)P(X = x)$
- In case of dependence between variables:
 - $P(Y = y, X = x) = P(Y = y|X = x)P(X = x)$, or
 - $P(Y = y, X = x) = P(X = x|Y = y)P(Y = y)$

Refresher: Joint probability

- What is the probability that the coin lands heads and we say heads?
 - $P(Y = 1, X = 1) = P(Y = 1|X = 1)P(X = 1) = ?$
 - $P(Y = 1, X = 1) = P(Y = 1|X = 1)P(X = 1) = 0.8 \times 0.5 = 0.4$
- Other joint probabilities
 - $P(Y = 0, X = 1) = P(Y = 0|X = 1)P(X = 1) = ?$
 - $P(Y = 0, X = 1) = P(Y = 0|X = 1)P(X = 1) = 0.2 \times 0.5 = 0.1$
 - $P(Y = 1, X = 0) = P(Y = 1|X = 0)P(X = 0) = ?$
 - $P(Y = 1, X = 0) = P(Y = 1|X = 0)P(X = 0) = 0 \times 0.5 = 0$
 - $P(Y = 0, X = 0) = P(Y = 0|X = 0)P(X = 0) = ?$
 - $P(Y = 0, X = 0) = P(Y = 0|X = 0)P(X = 0) = 1 \times 0.5 = 0.5$

Refresher: Marginal probability

- What is the probability of telling that the coin landed heads, or telling that the coin landed tails?
 - $P(Y = 1) = ?$
 - $P(Y = 0) = ?$
 - Where is X ?
- X has been marginalized from the joint distribution $P(Y = y, X = x)$
 - $P(Y = y) = \sum_x P(Y = y, X = x)$
 - For coin toss example:
 - $P(Y = y) = P(Y = y, X = 0) + P(Y = y, X = 1)$
- $P(Y = 1) = P(Y = 1, X = 0) + P(Y = 1, X = 1)$
- $P(Y = 1) = 0 + 0.4 = 0.4$
- $P(Y = 0) = ?$

Refresher: Bayes rule

- Joint probability can be estimated as (assuming dependence between variables):
 - $P(Y = y, X = x) = P(Y = y|X = x)P(X = x)$, or
 - $P(Y = y, X = x) = P(X = x|Y = y)P(Y = y)$
- By equating the right hand sides:
 - $P(X = x|Y = y)P(Y = y) = P(Y = y|X = x)P(X = x)$, so:

$$P(X = x|Y = y) = \frac{P(Y = y|X = x)P(X = x)}{P(Y = y)}$$

- For coin toss example, this is finding the probability that the coin landed in a particular way given what was said about the outcome

Refresher: Bayes rule

- What is the probability of coin landing head if it was told as head?
 - $P(X = 1|Y = 1) = ?$
 - $P(X = 1|Y = 1) = \frac{P(Y = 1|X = 1)P(X=1)}{P(Y=1)} = \frac{0.8 \times 0.5}{0.4} = 1$
- What is the probability of coin landing tail if it was told as tail?
 - $P(X = 0|Y = 0) = ?$
 - $P(X = 0|Y = 0) = \frac{P(Y = 0|X = 0)P(X=0)}{P(Y=0)} = \frac{1 \times 0.5}{0.6} = 0.83$
- What is the probability of coin landing head if it was told as tail?
 - $P(X = 1|Y = 0) = ?$
- What is the probability of coin landing tail if it was told as head?
 - $P(X = 0|Y = 1) = ?$

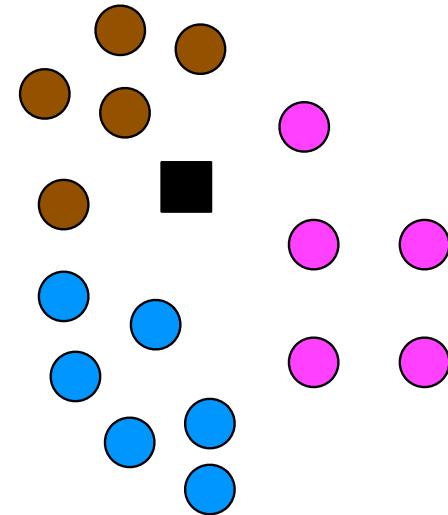
Refresher: Bayes rule

$$P(X = x|Y = y) = \frac{P(Y = y|X = x)P(X = x)}{P(Y = y)}$$

- $P(X = x)$ denotes **prior belief**: prior probability of event x occurring, before seeing any data for prediction
- $P(Y = y|X = x)$ denotes **class-conditional likelihood**: probability of the observed data y occurring, given that event x has occurred
- $P(Y = y)$ denotes **data evidence**: marginal probability of observed data y
 - $P(Y = y) = \sum_x P(Y = y, X = x) = \sum_x P(Y = y|X = x)P(X = x)$
- $P(X = x|Y = y)$ denotes **posterior probability**: probability of event x occurring after seeing the new data y

Classification

- A set of N objects with attributes (usually vector) \mathbf{x}_n
- Each object has an associated target label t_n
- Binary classification
 $t_n \in \{0,1\}$ or $t_n \in \{-1,1\}$
- Multi-class classification
 $t_n \in \{1,2,\dots,C\}$
- Classifier learns from $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ and t_1, t_2, \dots, t_N so that it can later classify \mathbf{x}_{new}



Classification

- Let's begin with a simpler problem formulation
- A set of N objects with discrete-valued scalar attribute x_n
- Each object x_n has an associated target label t_n
- Classifier learns from x_1, x_2, \dots, x_N and t_1, t_2, \dots, t_N so that it can later classify x_{new}
- Let's turn Bayes rule in to a classifier that can predict t_{new} for unseen data x_{new}

Bayesian classification

$$P(t_{new} = c|x_{new}) = \frac{P(x_{new}|t_{new} = c)P(t_{new} = c)}{P(x_{new})}$$

What is x_{new} ?

- $P(t_{new} = c)$ - **prior belief**: prior probability of label c occurring, before seeing any data for prediction
- $P(x_{new}|t_{new} = c)$ - **class-conditional likelihood**: probability of the data x_{new} occurring, given that label c is true
- $P(x_{new})$ - **data evidence**: marginal probability of data x_{new}
 - $P(x_{new}) = \sum_{c=1}^C P(x_{new}, t_{new} = c) = \sum_{c=1}^C P(x_{new}|t_{new} = c)P(t_{new} = c)$
- $P(t_{new} = c|x_{new})$ - **posterior probability**: probability of label c occurring after seeing the data x_{new}

Bayesian classification

$$P(t_{new} = c | x_{new}) = \frac{P(x_{new} | t_{new} = c) P(t_{new} = c)}{P(x_{new})}$$

can be written as:

$$P(c | x_{new}) = \frac{P(x_{new} | c) P(c)}{P(x_{new})}$$

- The Bayesian classifier estimates probability for all candidate class labels $c \in \{1, 2, \dots, C\}$
 - In this context, each class label c can also be termed as *candidate hypothesis* whose probability is estimated
 - Let's call set of all target labels or candidate hypotheses as *hypotheses space* $H = \{1, 2, \dots, C\}$

Bayesian classification

- Bayesian classifier aims to assign target label $c \in H$ that has maximum posterior probability

$$\underset{c \in H}{\operatorname{argmax}} P(c|x_{new})$$

$$\underset{c \in H}{\operatorname{argmax}} \frac{P(x_{new}|c)P(c)}{P(x_{new})}$$

$$\underset{c \in H}{\operatorname{argmax}} P(x_{new}|c)P(c)$$

- Note that $P(x_{new})$ is independent of target label c
 - i.e. it is same for all target labels, thus can be omitted
- This is called **maximum a posterior** hypothesis

Bayesian classification

- $P(x_{new}|c)$ denotes *class-conditional distribution*, specific to class c , evaluated at x_{new}
 - We need a class-conditional distribution for each class c , at each discrete-valued scalar attribute x_{new}
 - This distribution can be estimated from training data
- $P(c)$ denotes *prior probability* that can be set as:
 - Uniform prior: $P(c) = \frac{1}{C}$ i.e. each class is equally probable
 - Class size prior: $P(c) = \frac{N_c}{N}$ i.e. class prior as per its frequency in training observations

Bayesian classification

- In cases where the prior is uniform for all labels

$$\underset{c \in H}{\operatorname{argmax}} P(x_{new}|c)P(c)$$

becomes

$$\underset{c \in H}{\operatorname{argmax}} P(x_{new}|c)$$

- This is called **maximum likelihood** hypothesis

Bayesian classification

- **Maximum a posterior** (MAP) hypothesis
 - $\underset{c \in H}{\operatorname{argmax}} P(x_{new}|c)P(c)$
- **Maximum likelihood** (ML) hypothesis
 - $\underset{c \in H}{\operatorname{argmax}} P(x_{new}|c)$
- Typically, MAP estimate is used for Bayesian classification since it is flexible regarding prior use

Bayesian classification: Example

- Cancer diagnosis problem
 - A patient takes a lab test and the result comes back positive for cancer.
 - It is known that the test returns a correct positive result in only 98% of the cases and a correct negative result in only 97% of the cases.
 - Furthermore, only 0.008 (i.e. 0.8%) of the entire population has cancer.
1. What is the probability that this patient has cancer?
 2. What is the probability that this patient does not have cancer?
 3. What is the diagnosis?

Bayesian classification: Example

- Cancer diagnosis problem
 - A patient takes a lab test and the result comes back positive for cancer.
 - It is known that the test returns a correct positive result in only 98% of the cases and a correct negative result in only 97% of the cases.
 - Furthermore, only 0.008 (i.e. 0.8%) of the entire population has cancer.
- $P(cancer|+) = P(+/cancer)P(cancer) = ?$
- $P(\neg cancer|+) = P(?)|?)P(?) = ?$

Bayesian classification

- In the discussion (and example) so far, the data has only one discrete-valued attribute

$$\underset{c \in H}{\operatorname{argmax}} P(x_{new}|c)P(c)$$

- What if data has several discrete-valued attributes?
- Classification problem addressed was:
 - Classifier learns from x_1, x_2, \dots, x_N and t_1, t_2, \dots, t_N so that it can later classify x_{new}
- Now the classification problem becomes:
 - Classifier learns from $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ and t_1, t_2, \dots, t_N so that it can later classify \mathbf{x}_{new}

Naïve Bayes assumption

- Instead of $\underset{c \in H}{\operatorname{argmax}} P(x_{new}|c)P(c)$, we need to estimate:
$$\underset{c \in H}{\operatorname{argmax}} P(\mathbf{x}_{new}|c)P(c) = \underset{c \in H}{\operatorname{argmax}} P(x_{new}^1, x_{new}^2, \dots, x_{new}^d | c)P(c)$$
- Recall that $P(\mathbf{x}_{new}|c)$ denotes *class-conditional distribution*, specific to class c , evaluated at \mathbf{x}_{new}
 - It is extremely difficult to fit class-conditional distribution for each class c , for a high dimensional data
- Naïve Bayes assumption: attributes that describe data are conditionally independent given a hypothesis

$$P(\mathbf{x}_{new}|c) = \prod_{i=1}^d P(x_{new}^i|c)$$

- It is a simplifying assumption, obviously it may be violated in reality
- In spite of that, it works well in practice

Naïve Bayes classification

- Naïve Bayes classifier: uses the Naïve Bayes assumption and estimates the maximum a posterior (MAP) hypothesis

$$\underset{c \in H}{\operatorname{argmax}} P(\mathbf{x}_{new} | c) P(c) = \underset{c \in H}{\operatorname{argmax}} \prod_{i=1}^d p(x_{new}^i | c) P(c)$$

- Note that $p(x_{new}^i | c)$ denotes probability for the i^{th} attribute value
- Each attribute has discrete values (in discussion so far)
 - Continuous attribute values to be discussed later...
- Very simple but practical classification algorithm
- Successful applications:
 - Medical diagnosis
 - Text classification

Naïve Bayes classification: Learning to diagnose

- Given the patients data (symptoms and diagnosis):

chills	runny nose	headache	fever	Flu?
Y	N	Mild	Y	N
Y	Y	No	N	Y
Y	N	Strong	Y	Y
N	Y	Mild	Y	Y
N	N	No	N	N
N	Y	Strong	Y	Y
N	Y	Strong	N	N
Y	Y	Mild	Y	Y

- What is the probability for the following?

Y	N	Mild	N	?
---	---	------	---	---

Naïve Bayes classification: Learning to diagnose

- What is the probability for the following?

chills	runny nose	headache	fever	Flu?
Y	N	Mild	N	?

- $P(\text{flu} = Y | \mathbf{x}_{\text{new}}) = P(\mathbf{x}_{\text{new}} | \text{flu} = Y)P(\text{flu} = Y) = ?$
- Using Naïve Bayes classifier:
 - $P(\text{flu} = Y | \mathbf{x}_{\text{new}}) =$
 $P(\text{chills} = Y | \text{flu} = Y)$
 $P(\text{runny nose} = N | \text{flu} = Y)$
 $P(\text{headache} = \text{Mild} | \text{flu} = Y)$
 $P(\text{fever} = N | \text{flu} = Y)$
 $P(\text{flu} = Y) = ?$
 - $P(\text{flu} = N | \mathbf{x}_{\text{new}}) = P(\mathbf{x}_{\text{new}} | \text{flu} = N)P(\text{flu} = N) = ?$

Naïve Bayes classification: Learning to classify

- Given the tennis playing data:

outlook	temperature	humidity	wind	Play tennis?
sunny	hot	high	weak	N
sunny	hot	high	strong	N
overcast	hot	high	weak	Y
rain	mild	high	weak	Y
rain	cool	normal	weak	Y
rain	cool	normal	strong	N
overcast	cool	normal	strong	Y
sunny	mild	high	weak	N
sunny	cool	normal	weak	Y
rain	mild	normal	weak	Y
sunny	mild	normal	strong	Y
overcast	mild	high	strong	Y
overcast	hot	normal	weak	Y
rain	mild	high	strong	N

Naïve Bayes classification: Learning to classify

- Classify the new data:

outlook	temperature	humidity	wind	Play tennis?
sunny	cool	high	strong	?

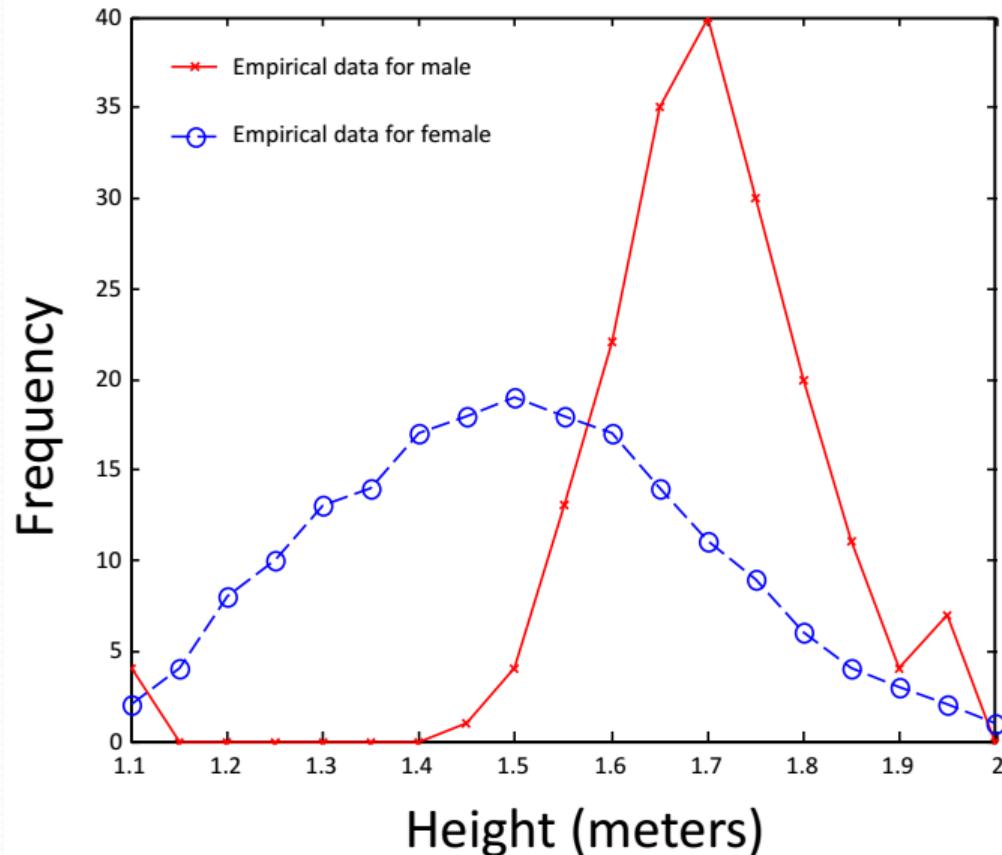
- $P(play = Y | \mathbf{x}_{new}) = P(\mathbf{x}_{new} | play = Y)P(play = Y) = ?$
- $P(play = N | \mathbf{x}_{new}) = P(\mathbf{x}_{new} | play = N)P(play = N) = ?$

Gaussian classification

- In the discussion (and examples) so far, the data has discrete-valued attributes (e.g. 'sunny', 'Y')
 - What if the attributes are real-valued?
- For discrete-valued attributes, we used probability distributions to estimate attribute value relation with class label
- For real-valued attributes, we need to use probability density function (pdf) to estimate attribute value relation with class label
- Gaussian classifier: Bayes or Naïve Bayes classifier that utilizes Gaussian pdf

Gaussian classification: Learning to predict gender

- Given heights of 190 male and 190 females, can a classifier learn to predict gender?
- Attribute?
 - Height
- Class labels?
 - $t_{male} = 1$
 - $t_{female} = 0$
- $c \in \{0,1\}$



Gaussian classification: Learning to predict gender

- Let's recall maximum a posterior estimate from Bayes rule:

$$p(c|x_{new}) = p(x_{new}|c)p(c)$$

- We need class prior:
 - $p(c = 1) = ?$
 - $p(c = 0) = ?$
- We will also need class-conditional likelihood:
 - $p(x_{new}|c = 1)$: probability that a male has height x_{new}
 - $p(x_{new}|c = 0)$: probability that a female has height x_{new}
- Posterior
 - $p(c = 1|x_{new})$: probability that height x_{new} is a male
 - $p(c = 0|x_{new})$: probability that height x_{new} is a female

Gaussian classification: Learning to predict gender

- Class-conditional likelihood $p(x_{new}|c)$
 - Class-conditional distribution can be modelled as a Gaussian pdf, for each class

- Univariate Gaussian pdf

$$p(x_{new}|\mu_c, \sigma_c^2) = \frac{1}{\sqrt{2\pi\sigma_c^2}} \exp\left\{-\frac{(x_{new}-\mu_c)^2}{2\sigma_c^2}\right\}$$

- How to estimate μ_c and σ_c^2 ?

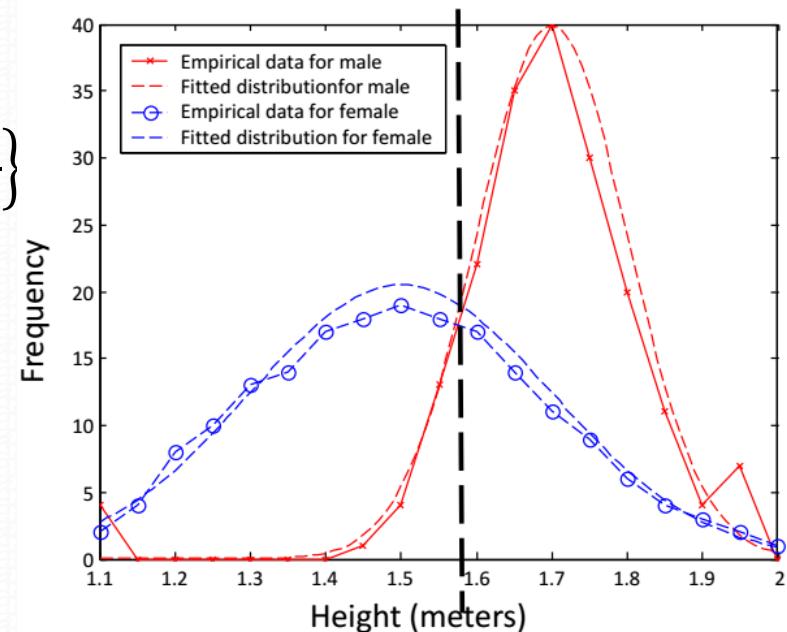
$$\mu_c = \frac{1}{N_c} \sum_{n=1}^{N_c} x_n$$

$$\sigma_c^2 = \frac{1}{N_c} \sum_{n=1}^{N_c} (x_n - \mu_c)^2$$

- We will have a separate Gaussian pdf for each class

- μ_0 and σ_0^2 : mean and variance for female class

- μ_1 and σ_1^2 : mean and variance for male class



Gaussian classification: Learning to predict gender

- By estimating class-conditional likelihood and class prior, posterior can be estimated:

$$p(c|x_{new}) = p(x_{new}|c)p(c)$$

- Since $p(x_{new}|c) = p(x_{new}|\mu_c, \sigma_c^2)$, we get:

$$p(c|x_{new}) = p(x_{new}|\mu_c, \sigma_c^2)p(c)$$

- Recall that

$$p(x_{new}|\mu_c, \sigma_c^2) = \frac{1}{\sqrt{2\pi\sigma_c^2}} \exp\left\{-\frac{(x_{new} - \mu_c)^2}{2\sigma_c^2}\right\}$$

- Thus, the posterior estimate for prediction is:

$$p(c|x_{new}) = \frac{1}{\sqrt{2\pi\sigma_c^2}} \exp\left\{-\frac{(x_{new} - \mu_c)^2}{2\sigma_c^2}\right\} p(c)$$

Gaussian classification

- In the Gaussian classifier discussion so far, the data has only one attribute
 - i.e. one-dimensional input x_n
 - For such case, univariate Gaussian pdf was sufficient to estimate class-conditional likelihood
- What if the data has multiple attributes?
 - i.e. d-dimensional input $\mathbf{x}_n = \{x_n^1, x_n^2, \dots, x_n^d\}$
 - Multivariate Gaussian pdf will be needed for estimation of class-conditional likelihood

$$p(\mathbf{x}_{new} | \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}_c|}} \exp \left\{ -\frac{1}{2} (\mathbf{x}_{new} - \boldsymbol{\mu}_c)^T \boldsymbol{\Sigma}_c^{-1} (\mathbf{x}_{new} - \boldsymbol{\mu}_c) \right\}$$

Gaussian classification

- Multivariate Gaussian pdf for class-conditional likelihood

$$p(\mathbf{x}_{new} | \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}_c|}} \exp \left\{ -\frac{1}{2} (\mathbf{x}_{new} - \boldsymbol{\mu}_c)^T \boldsymbol{\Sigma}_c^{-1} (\mathbf{x}_{new} - \boldsymbol{\mu}_c) \right\}$$

where

$$\boldsymbol{\mu}_c = \frac{1}{N_c} \sum_{n=1}^{N_c} \mathbf{x}_n$$

$$\boldsymbol{\Sigma}_c = \frac{1}{N_c} \sum_{n=1}^{N_c} (\mathbf{x}_n - \boldsymbol{\mu}_c)(\mathbf{x}_n - \boldsymbol{\mu}_c)^T$$

What is the difference between \mathbf{x}_n and \mathbf{x}_{new} ?

Gaussian classification

- By estimating class-conditional likelihood and class prior, posterior can be estimated:

$$p(c|x_{new}) = p(x_{new}|c)p(c)$$

- Since $p(x_{new}|c) = p(x_{new}|\mu_c, \Sigma_c)$, we get:

$$p(c|x_{new}) = p(x_{new}|\mu_c, \Sigma_c)p(c)$$

- Recall that

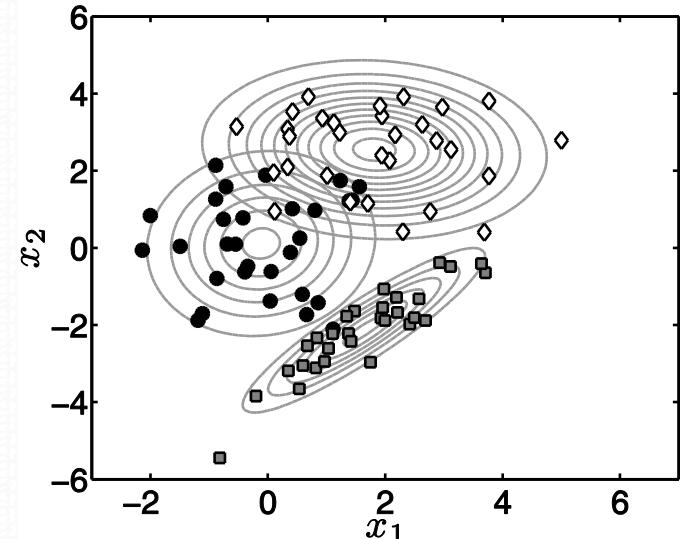
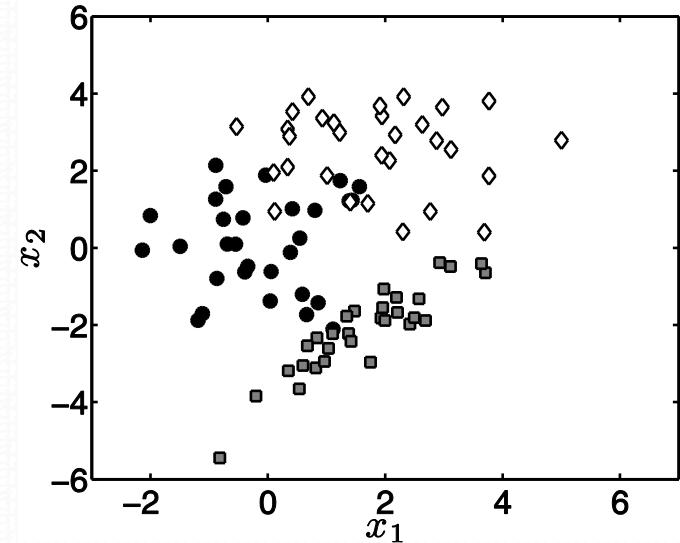
$$p(x_{new}|\mu_c, \Sigma_c) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_c|}} \exp \left\{ -\frac{1}{2} (x_{new} - \mu_c)^T \Sigma_c^{-1} (x_{new} - \mu_c) \right\}$$

- Thus, the posterior estimate for prediction is:

$$p(c|x_{new}) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_c|}} \exp \left\{ -\frac{1}{2} (x_{new} - \mu_c)^T \Sigma_c^{-1} (x_{new} - \mu_c) \right\} p(c)$$

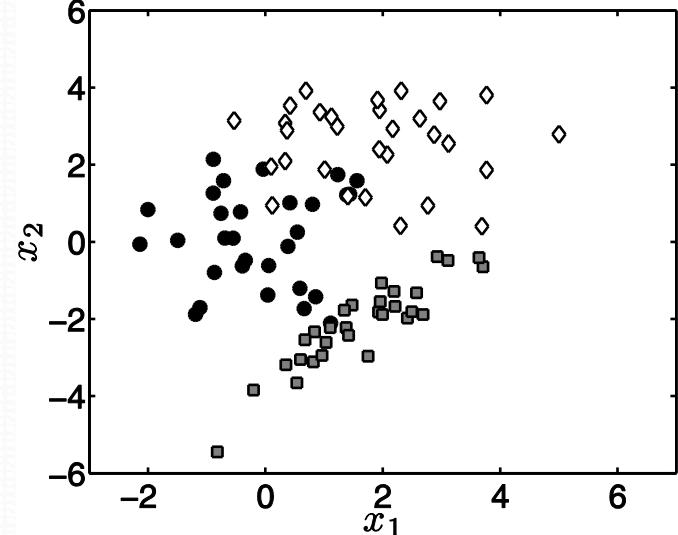
Gaussian classification

- Example: three class 2d data (30 samples each)
 - 1: black circles
 - 2: white diamonds
 - 3: grey squares
- Class conditional prior
$$p(\mathbf{x}_{new} | \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}_c|}} \exp \left\{ -\frac{1}{2} (\mathbf{x}_{new} - \boldsymbol{\mu}_c)^T \boldsymbol{\Sigma}_c^{-1} (\mathbf{x}_{new} - \boldsymbol{\mu}_c) \right\}$$
- Density contours



Gaussian classification: Making predictions

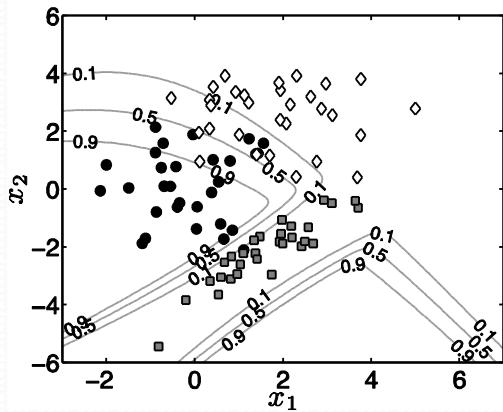
- Example: three class 2d data (30 samples each)
 - 1: black circles
 - 2: white diamonds
 - 3: grey squares
- Posterior $p(c|\mathbf{x}_{new})$ for $\mathbf{x}_{new} = [2,0]^T$



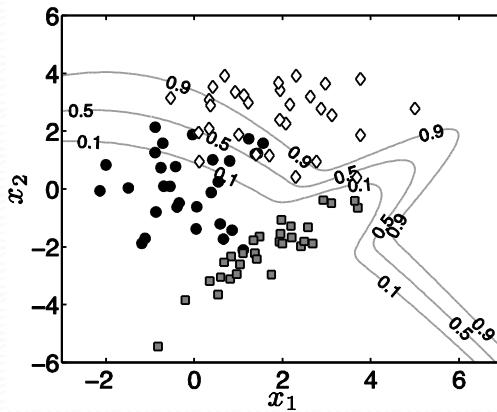
c	$p(\mathbf{x}_{new} \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$	$p(c)$	$p(c \mathbf{x}_{new})$ $= p(\mathbf{x}_{new} \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)p(c)$	$P(c \mathbf{x}_{new})$
1	0.0138	1/3	0.0046	0.6890
2	0.0061	1/3	0.0020	0.3024
3	0.0002	1/3	0.0001	0.0087

Gaussian classification: Making predictions

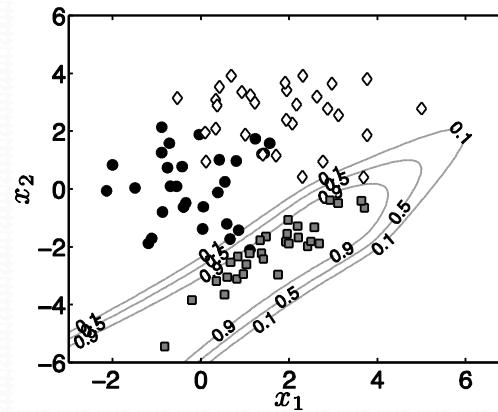
- By evaluating the Gaussian classifier on a grid of many \mathbf{x}_{new} values, we can estimate and draw the classification probability contours



(a) $P(T_{\text{new}} = 1 | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$



(b) $P(T_{\text{new}} = 2 | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$



(c) $P(T_{\text{new}} = 3 | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$

- The steepness of density contours for class 3 partly explains the odd behaviour in (a) and (b)

Gaussian classification

- How many parameters to estimate multivariate Gaussian pdf with 2d data, for each class?
 - $\boldsymbol{\mu}_c = \frac{1}{N_c} \sum_{n=1}^{N_c} \mathbf{x}_n$
 - $\boldsymbol{\Sigma}_c = \frac{1}{N_c} \sum_{n=1}^{N_c} (\mathbf{x}_n - \boldsymbol{\mu}_c)(\mathbf{x}_n - \boldsymbol{\mu}_c)^T$
 - Five parameters: two for $\boldsymbol{\mu}_c$ and three for $\boldsymbol{\Sigma}_c$
- In general, for D dimensional data,
we need to estimate $D + D + \frac{D(D-1)}{2}$ parameters
 - For 10-dimensional data, 65 parameters need to be estimated for each class
 - What if the training data size is small (e.g. 30 samples)?

Gaussian classification: Naïve Bayes assumption

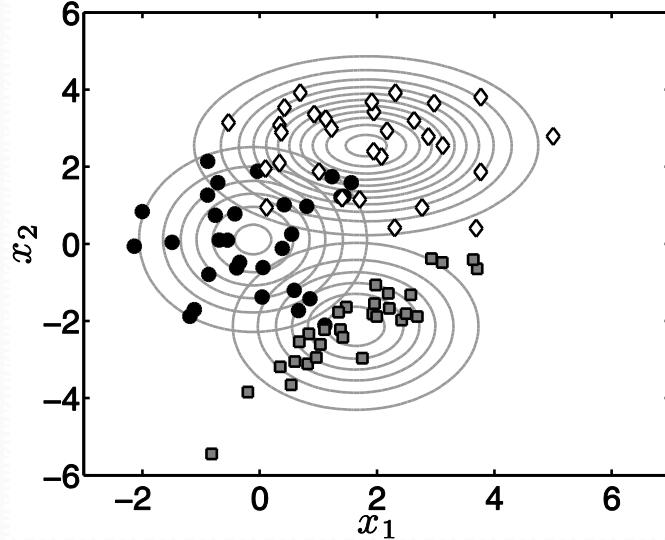
- To “overcome” lack of data, Naïve Bayes assumption can be utilized
 - i.e. each attribute is assumed to be independent
- Class-conditional multivariate distribution is factorized into product of D univariate distributions, for each class

$$p(\mathbf{x}_{new} | c) = \prod_{d=1}^D p(x_{new}^d | c) = \prod_{d=1}^D p(x_{new}^d | \mu_c, \sigma_c^2)$$

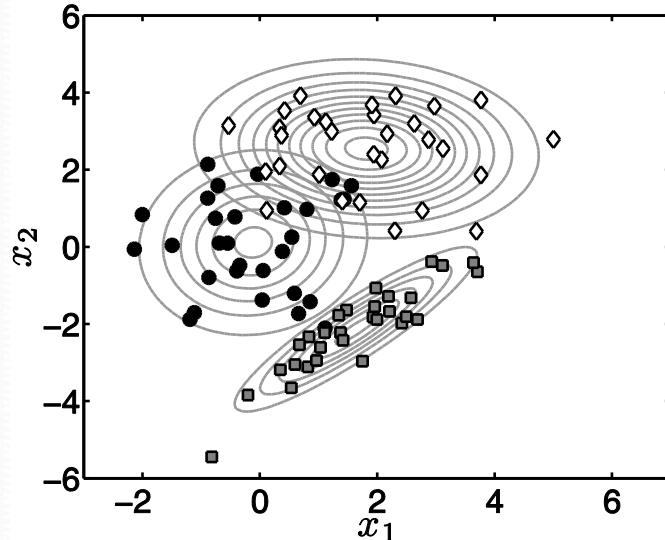
- Each univariate distribution relies on only two parameters: mean μ_c and variance σ_c^2
 - Thus, only $2 * D$ parameters need to be estimated with Naïve Bayes assumption

Gaussian classification: Naïve Bayes assumption

- Density contours,
with Naïve Bayes assumption



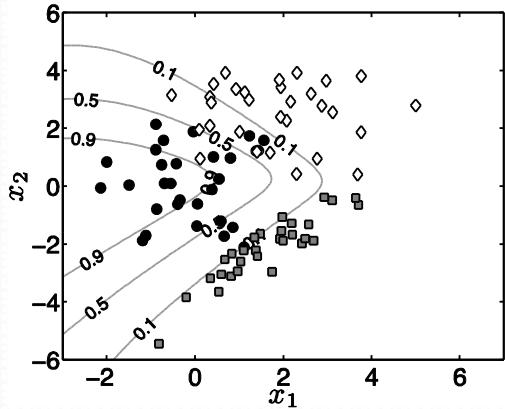
- Density contours,
without Naïve Bayes
assumption



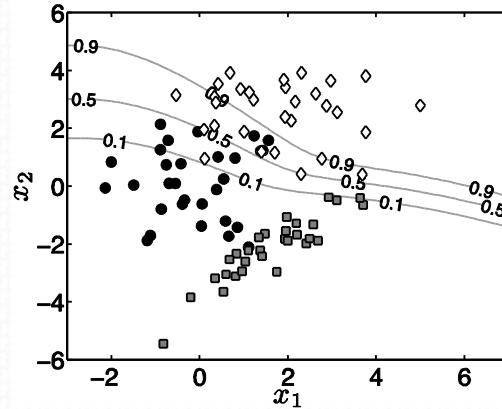
Gaussian classification: Naïve Bayes assumption

- Classification probability contours

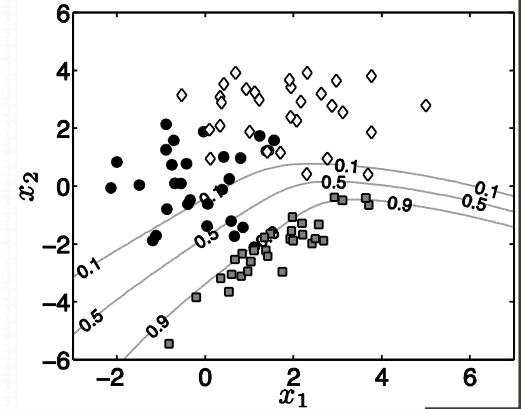
with
Naïve
Bayes
assump
tion



(a) $P(T_{\text{new}} = 1 | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$

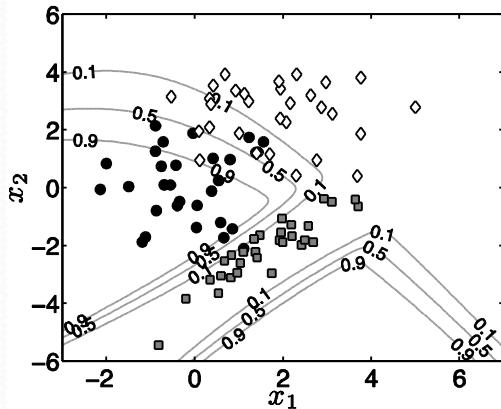


(b) $P(T_{\text{new}} = 2 | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$

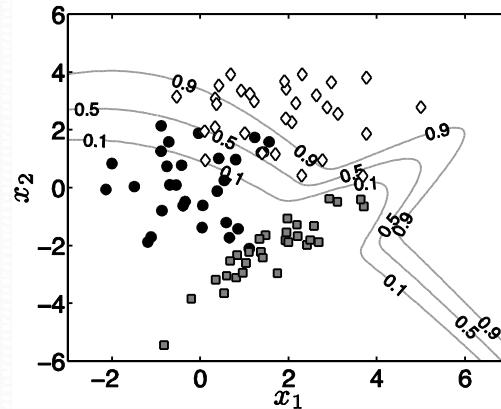


(c) $P(T_{\text{new}} = 3 | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$

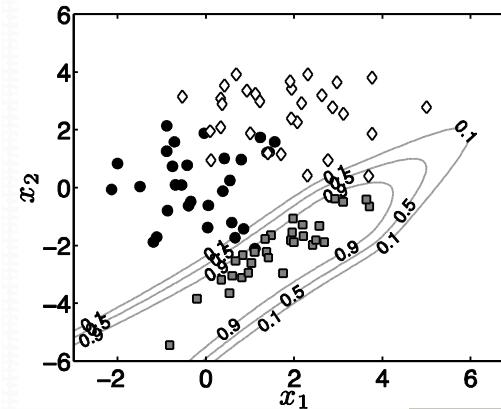
without
Naïve
Bayes
assump
tion



(a) $P(T_{\text{new}} = 1 | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$



(b) $P(T_{\text{new}} = 2 | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$



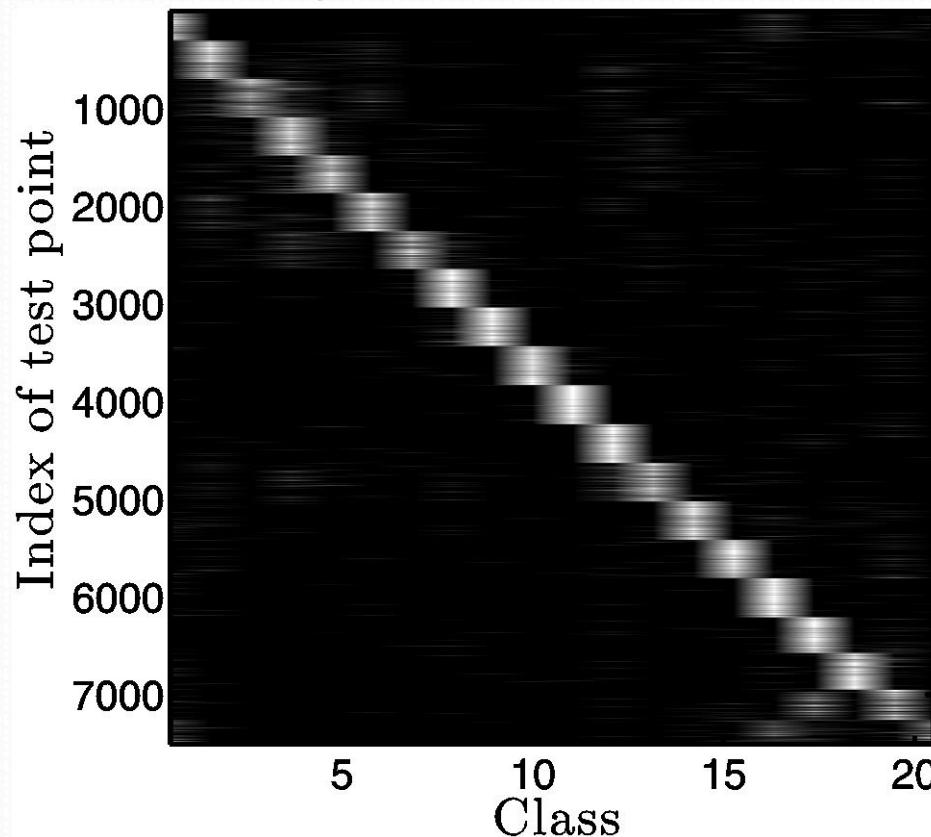
(c) $P(T_{\text{new}} = 3 | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t})$

Gaussian classification: Learning to classify text

- Automatic text classification
 - 20 newsgroups dataset consisting of 20,000 documents (covering sports, religion, computing, etc)
 - Each document is a post to one of 20 newsgroups
- Learn to classify a new document to one of these 20 newsgroups
- What are the attributes?
 - Words?
- How to encode the document as a vector of numerical values for classification?
 - Bag-of-words
 - Word frequency
- With or without Naïve Bayes assumption?

Gaussian classification: Learning to classify text

- ~11,000 documents used as training dataset
- ~7,000 documents used as testing dataset
- Each test document is assigned a class, which has highest probability out of the 20 probability estimates
 - 78% classification accuracy



Summary

- Generative approach to classification
- Bayes rule for classification
- Naïve Bayes classifier is a simple but effective classifier for data having several attributes
- Class-conditional likelihood
- Class prior
- Maximum a posteriori Naïve Bayes estimate

Exercise (ungraded)

- ML – Tom Mitchell: Exercise 6.1
 - Consider the example application (disease diagnosis) of Bayes rule (on slides 27 & 28). Suppose the doctor decides to order a second lab test for the same patient, and suppose the second test returns a positive result as well. What are the posterior probabilities of cancer and \neg cancer following these two tests? Assume that the two tests are independent?

Exercise (ungraded)

- Given training data, train a Gaussian classifier (without Naïve Bayes assumption)

Attribute 1	Attribute 2	class
-4	1	1
-5	2	1
-3	3	1
-2.5	4.5	1
-4	5	1
3	1	2
3.5	0	2
4	0.5	2
4	-1	2
3.5	-1	2

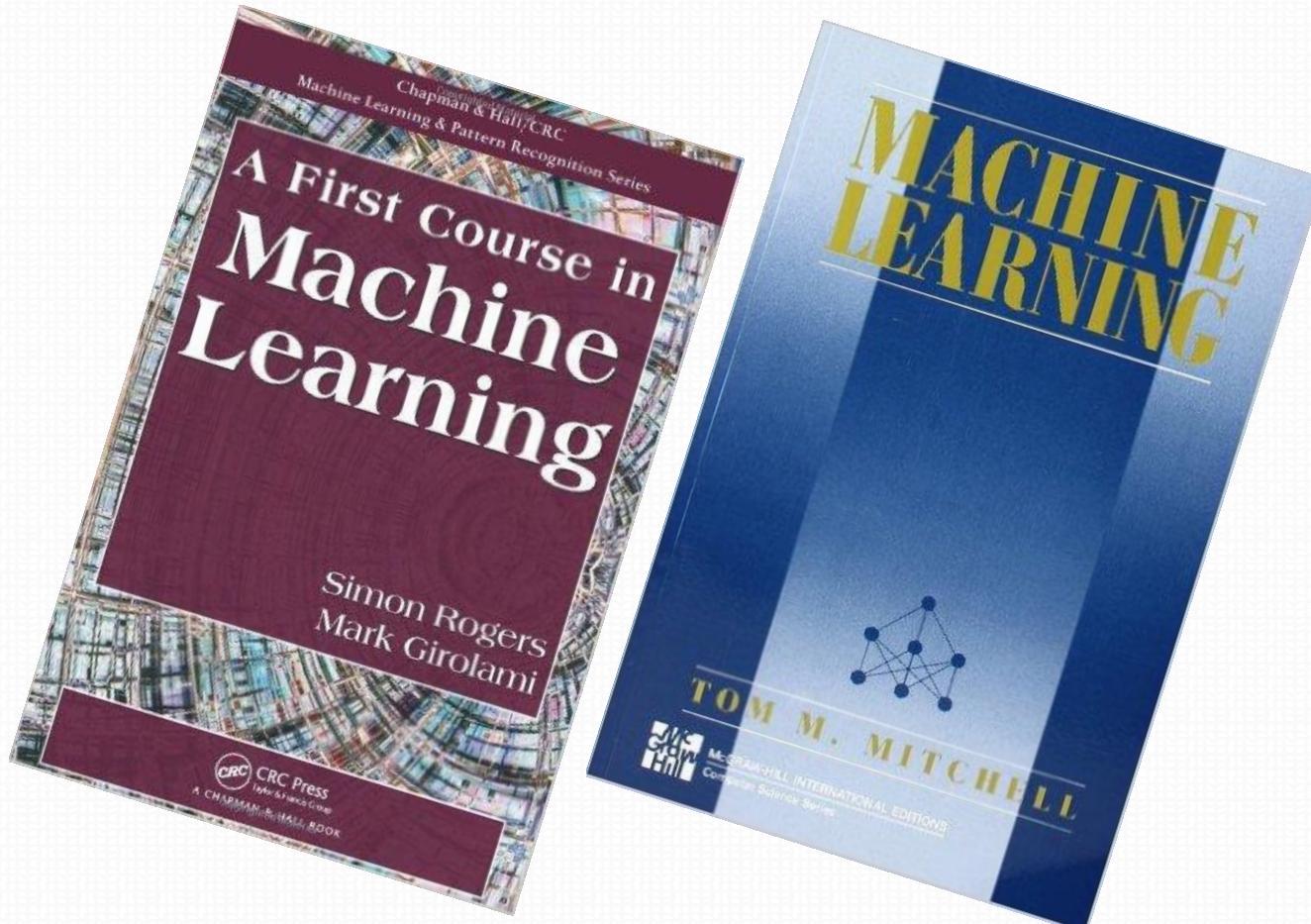
- Predict a new sample by estimating posterior using this Gaussian classifier

Attribute 1	Attribute 2	class
-2	2	?

Exercise (ungraded)

- Try MATLAB code – plotcc.m (from FCML book website)
- Try MATLAB code – bayesclass.m (from FCML book website)

C₃ R₁ E₂ D₂ I₁ T₁ S₁



Author's material
(Simon Rogers)

- Ata Kaban's material from previous years



Thank You

Machine Learning, Machine Learning (extended)

5 – Supervised Learning: Evaluation Metrics for Classification

Kashif Rajpoot

k.m.rajpoot@cs.bham.ac.uk

**School of Computer Science
University of Birmingham**

Outline

- Performance evaluation
- 0/1 loss
- Classification accuracy
- Confusion matrix
 - True positive
 - True negative
 - False positive
 - False negative
- Sensitivity
- Specificity
- ROC analysis
 - Area under curve
- Hold-out validation
- Cross-validation
 - K-fold
 - Leave-one-out
- Repeated cross-validation

Performance evaluation

- How to assess a classification algorithm?
- How to choose?
 - Classification algorithm?
 - Algorithm parameters?

0/1 loss

- 0/1 loss: proportion of times classifier is wrong
- Consider a set of classifier label predictions t_1, t_2, \dots, t_N and *ground truth* target labels $t_1^*, t_2^*, \dots, t_N^*$
- Mean 0/1 loss can be computed as:
 - $\frac{1}{N} \sum_{n=1}^N \delta(t_n \neq t_n^*)$
- For a particular test sample prediction t_n :
 - $\delta(t_n \neq t_n^*) = 1$
 - $\delta(t_n = t_n^*) = 0$
- The lower the 0/1 loss, the better
- Advantages
 - Simple
 - Suitable for binary or multi-class classification

0/1 loss

- Disadvantage
 - Suffers from class imbalance
- Imagine we're building a classifier to detect a rare disease
 - Consider only 1% of population is diseased
 - $t = 1$, for diseased
 - $t = 0$, for healthy
- What if algorithm always predicts $t = 0$?
- Accuracy will be 99%, but the classification algorithm is rubbish

Classification accuracy

- A slight variant of 0/1 loss, computed as percentage accuracy
- Consider a set of classifier label predictions t_1, t_2, \dots, t_N and ground truth target labels $t_1^*, t_2^*, \dots, t_N^*$
- Mean classification accuracy can be computed as:
 - $100 * \frac{1}{N} \sum_{n=1}^N \delta(t_n = t_n^*)$
- For a particular test sample prediction t_n :
 - $\delta(t_n \neq t_n^*) = 0$
 - $\delta(t_n = t_n^*) = 1$
- The higher the classification accuracy, the better
- Disadvantage
 - Suffers from class imbalance, similar to 0/1 loss

Confusion matrix (CM)

- Shows the simple count of correctly (and wrongly) classified samples by the classifier
- Consider a binary classification problem with 20 diseased and 80 healthy test samples

		Actual class label		Total
		Diseased	Healthy	
Predicted class label	Diseased	15	4	19
	Healthy	5	76	81
Total		20	80	

- Variety of metrics can be estimated from CM
- Suitable for binary and multi-class classification
 - Particularly useful for multi-class problems
 - Specifically indicates problems with an individual class

Confusion matrix

- Example: classify ~7000 test documents in 20 classes (newsgroups data)
 - Too similar classes?
 - Need more data?

Predicted class	True class																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	242	3	3	0	1	0	0	1	0	4	2	0	2	10	4	7	1	12	7	47
2	0	296	33	8	8	42	9	1	1	0	0	4	18	7	8	2	0	1	1	3
3	0	6	209	15	9	8	4	0	0	0	0	1	0	1	0	1	0	0	0	0
4	0	12	60	303	36	12	46	2	0	1	0	1	28	3	0	0	0	0	0	0
5	0	8	10	22	277	2	21	0	0	1	0	2	7	0	0	1	1	0	0	0
6	1	21	30	2	2	304	0	1	0	3	0	1	3	0	1	2	0	0	1	0
7	0	1	0	5	5	1	235	5	1	2	0	1	1	0	0	0	1	0	0	0
8	0	3	1	6	4	0	31	356	25	3	1	0	9	4	0	0	2	2	1	0
9	0	2	2	0	1	2	5	4	353	1	0	0	2	0	1	0	1	1	0	1
10	0	0	2	0	1	1	0	2	2	348	4	0	0	1	0	0	1	1	1	0
11	1	0	1	1	0	0	1	0	0	16	382	0	1	0	1	0	1	1	0	0
12	1	16	16	5	4	10	3	1	1	2	0	360	45	0	4	1	3	4	3	1
13	1	4	1	24	16	0	9	5	1	2	0	3	260	3	4	0	0	0	0	0
14	2	3	4	0	8	0	2	0	1	0	2	2	6	324	4	1	1	0	3	3
15	3	7	4	1	2	3	3	2	0	0	1	0	4	3	336	0	2	0	7	5
16	39	4	5	0	0	1	3	1	1	3	2	2	5	17	4	376	3	7	2	68
17	4	0	0	0	3	1	1	5	4	1	0	9	0	3	1	3	325	3	95	19
18	7	1	0	0	0	1	3	1	2	2	1	0	2	6	2	1	2	325	4	5
19	7	2	9	0	6	2	5	8	5	8	4	8	0	10	21	1	16	19	185	7
20	10	0	1	0	0	0	1	0	0	0	0	1	0	1	1	2	4	0	1	92

Confusion matrix (CM)

		Actual class label		Total
		Diseased	Healthy	
Predicted class label	Diseased	TP	FP	TP+FP
	Healthy	FN	TN	FN+TN
Total		TP+FN	FP+TN	

- True positive (TP): diseased samples are classified as diseased
 - Number of test samples with $t_n^* = 1$ that are classified as $t_n = 1$
- True negative (TN): healthy samples are classified as healthy
 - Number of test samples with $t_n^* = 0$ that are classified as $t_n = 0$
- False positive (FP): healthy samples are classified as diseased
 - Number of test samples with $t_n^* = 0$ that are classified as $t_n = 1$
- False negative (FN): diseased samples are classified as healthy
 - Number of test samples with $t_n^* = 1$ that are classified as $t_n = 0$

Sensitivity and Specificity

- Sensitivity: proportion of diseased samples that are classified as diseased
 - The higher, the better

$$Sen = \frac{TP}{TP + FN} = \frac{TP}{All\ Positive}$$

- Specificity: the proportion of healthy samples that are classified as healthy
 - The higher, the better

$$Spec = \frac{TN}{TN + FP} = \frac{TN}{All\ Negative}$$

Sensitivity and Specificity

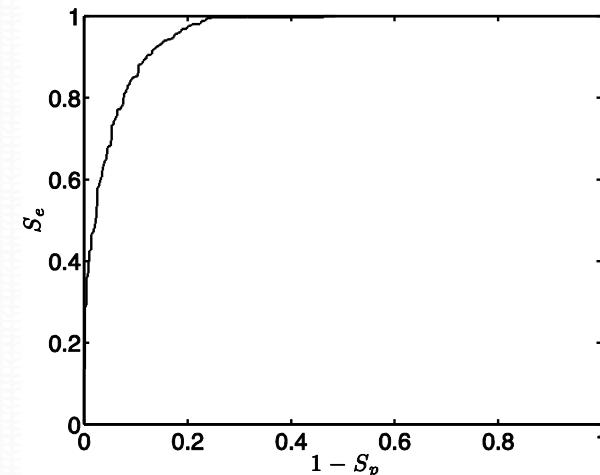
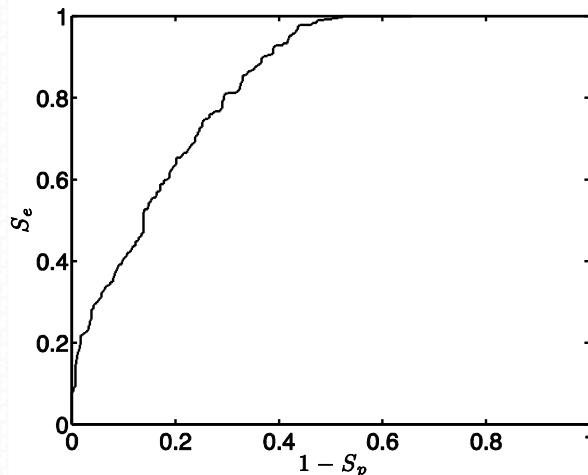
- Consider only 1% of population is diseased
- Let's say:
 - $t = 1$, for diseased
 - $t = 0$, for healthy
- What if algorithm always predicts $t = 0$?
 - $Sen = ?$
 - $Sen = 0$
 - $Spec = ?$
 - $Spec = 1$
- We would like both Sen and $Spec$ to be as high as possible
 - Often, increasing one will decrease the other
- In a disease diagnosis system:
 - We can probably tolerate a decrease in specificity (healthy samples classified as diseased), if it provides an increase in sensitivity (diseased samples classified as healthy)

Performance evaluation

- Let's recall that often classification algorithms provide real-valued output which is then thresholded to assign a classification label
- Bayesian classifier
 - $P(t_{new} = 1 | \mathbf{x}_{new}, \mathbf{X}, t)$
- SVM
 - $t_{new} = sign(\sum_{n=1}^N t_n \alpha_n k(\mathbf{x}_n, \mathbf{x}_{new}) + b)$
- How about perturbing this threshold value?

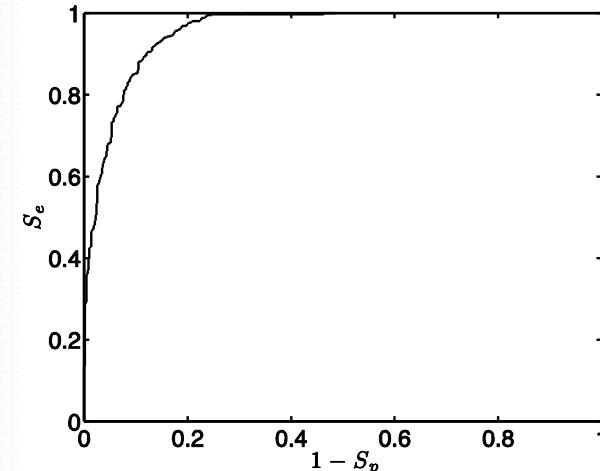
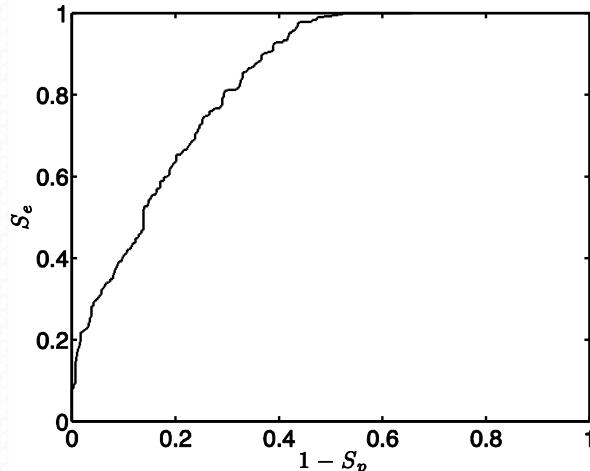
ROC analysis

- Threshold is changed to evaluate classifier's performance
- Receiver operating characteristic (ROC) curve
 - Sensitivity (Sen) is plotted against the complementary specificity ($1 - Spec$)
 - Every point on the curve reflects classifier's performance at a particular threshold value



ROC analysis

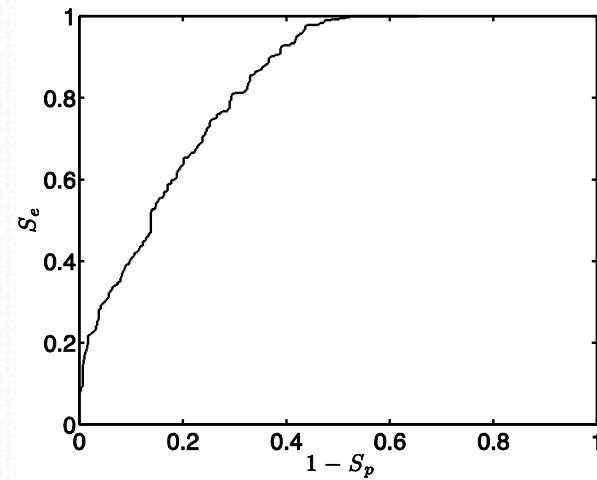
- We would like both Sen and $Spec$ to be as high as possible
 - i.e. $Sen = 1, Spec = 1, 1 - Spec = 0$
- Bottom left: every sample is classified as healthy (0)
- Top right: every sample is classified as diseased (1)
- Ideal: get the curve to the top left corner
 - Perfect classification: $Sen = 1, Spec = 1$



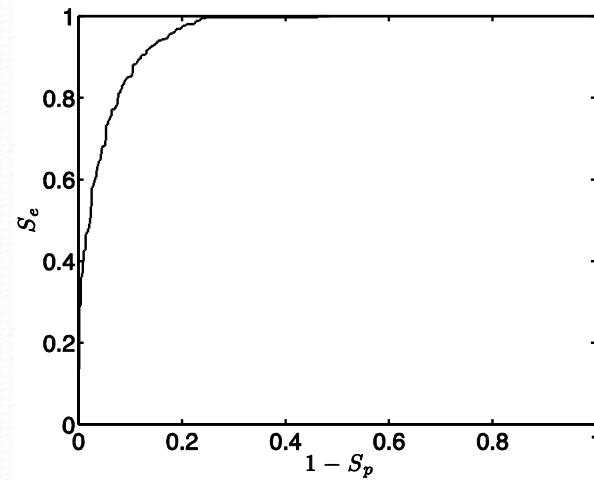
ROC analysis

- Area under curve (AUC)
 - Quantify performance by estimating AUC
 - The higher, the better
- AUC is a better measure than 0/1 loss or classification accuracy
 - Considers class imbalance in data

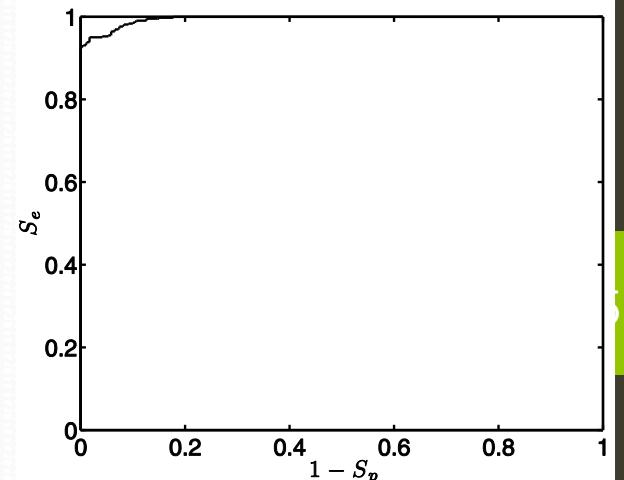
AUC= 0.8348



AUC= 0.9551



AUC= 0.9936



ROC analysis

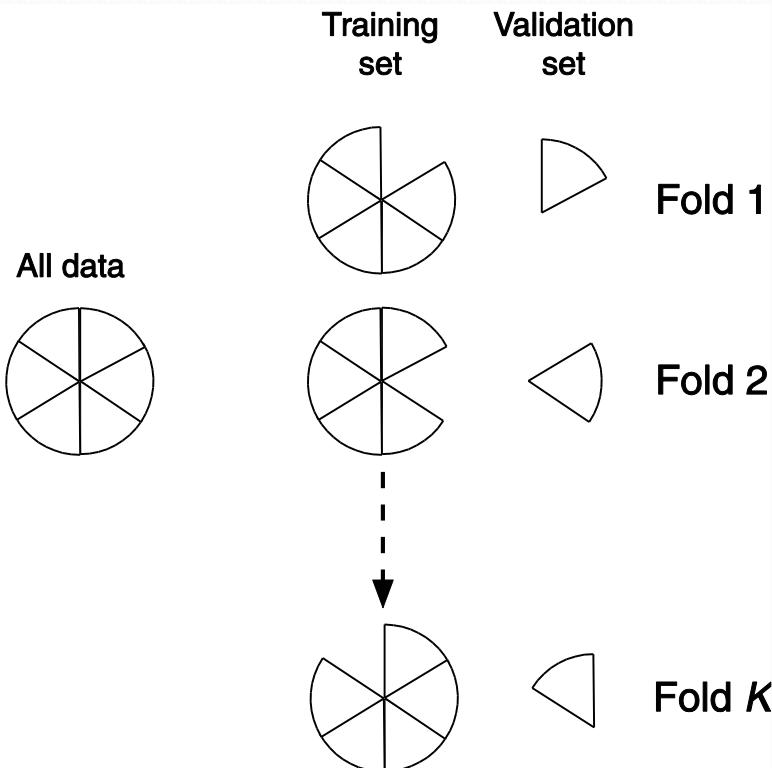
- Multi-class classification
 - Not naturally suitable
- One-against-all classification
- For example: for a 3 class problem, we will generate 3 ROC analyses
 - Each one looks at the binary classification results for class c against the rest

Hold-out validation

- We have earlier discussed the issues of overfitting and generalization
 - Ideally, we want a classifier that can generalize well i.e. on unseen data
 - Where to get unseen data?
- Hold-out validation
 - Partition observed data into training and testing portions
 - For example: 80-20, or 50-50 split
- Disadvantages
 - Reduce training data
 - Representativeness i.e. validation is biased towards choice of data in validation set, particularly if data is small

Cross-validation

- K-fold cross-validation
 - Hold out K^{th} portion/fold for testing
 - Repeat the classifier training and testing for each fold
- Compute average error from all folds
- Leave-one-out cross-validation (LOOCV)
 - Extreme case of K-fold cross-validation
 - Computationally exhausting



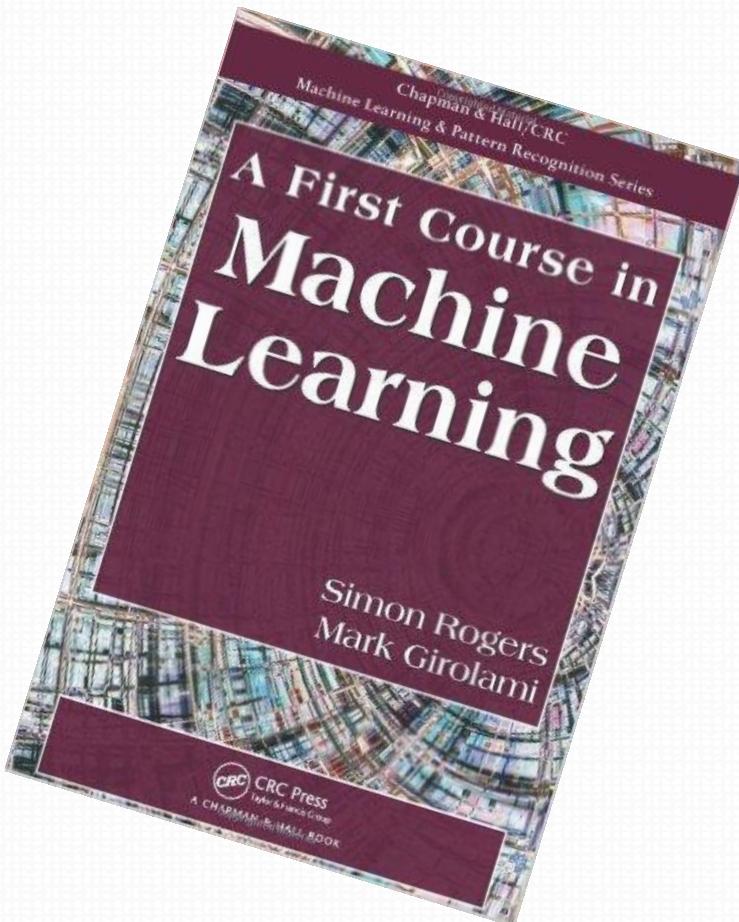
Repeated cross-validation

- The hold-out validation or k-fold cross-validation can be made more reliable by repeating it several times
 - With random selection of training and testing datasets
- Accuracies obtained from various repeats are averaged to indicate overall performance
- Computationally exhaustive

Summary

- 0/1 loss
- Classification accuracy
- Confusion matrix
- Sensitivity
- Specificity
- ROC analysis
- Assessing binary-class classification performance
- Assessing multi-class classification performance
- Cross-validation

CREDITS



Author's material
(Simon Rogers)



Thank You

Machine Learning, Machine Learning (extended)

6 – Supervised Learning: Instance-based Classification

Kashif Rajpoot

k.m.rajpoot@cs.bham.ac.uk

**School of Computer Science
University of Birmingham**

Outline

- Supervised learning
- Instance based learning
- k-nearest neighbour (kNN) classification
- Distance measure
- Difficulties with kNN
- How to choose k ?
- Distance-weighted kNN

Supervised learning

- Regression
 - Minimised loss (e.g. least squares)
 - Maximum likelihood
- Classification
 - Generative (e.g. Bayesian)
 - Instance-based (e.g. k-NN)
 - Discriminative (e.g. SVM)

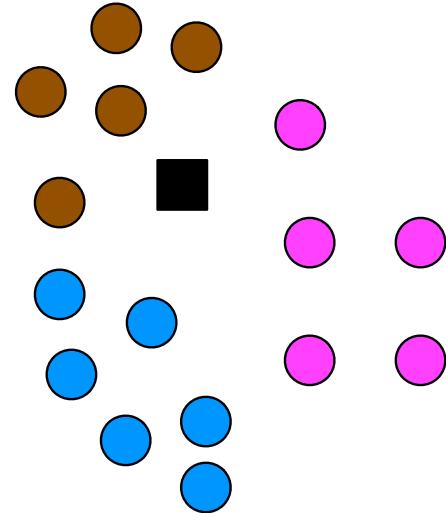
Classification

- A set of N objects with attributes \mathbf{x}_n
- Each object has an associated target label t_n
- Binary classification

$$t_n \in \{0,1\} \text{ or } t_n \in \{-1,1\}$$

- Multi-class classification

$$t_n \in \{1,2,\dots,C\}$$



- Classifier learns from $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ and t_1, t_2, \dots, t_N so that it can later classify \mathbf{x}_{new}

Instance-based supervised learning

- There is no ‘training’ involved
 - Stores all ‘training’ samples
- Non-probabilistic classification
 - Look at nearest instances from the past examples to determine target labels (discrete or real-valued) of new unseen samples
 - k-nearest neighbour (kNN) classification
- Suitable for binary or multi-class classification

kNN classification

- Store all N ‘training’ samples represented by attributes $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ and labels t_1, t_2, \dots, t_N
- Choose parameter k
 - i.e. number of nearest neighbours to consider for assigning target label to a new test sample \mathbf{x}_{new}
- For discrete labels, classify a new test sample \mathbf{x}_{new} :
 - Find k nearest (closest) training samples/instances
 - Find the most common class (label) out of these k neighbours
 - Assign this class to \mathbf{x}_{new}
 - In case of a tie, assign randomly from tied classes

kNN classification

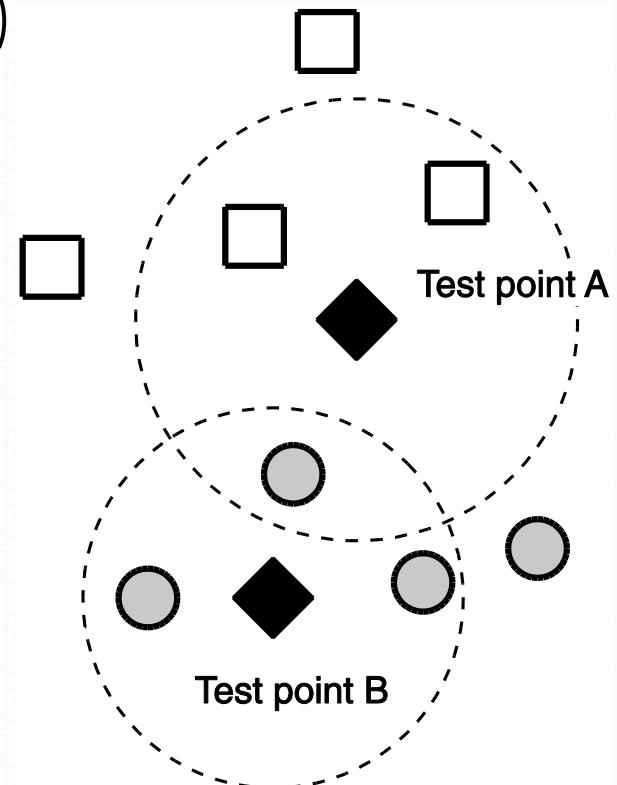
- Training algorithm
 - Store all training instances ($\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ and t_1, t_2, \dots, t_N) as *training_examples*
- Classification algorithm
 - Given a test instance \mathbf{x}_{new} to be classified:
 - Let $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k$ denote the k nearest instances to \mathbf{x}_{new} from the *training_examples*
 - Let t_1, t_2, \dots, t_k denote the labels of k nearest instances to \mathbf{x}_{new}
 - Assign target label t_{new} to \mathbf{x}_{new} as below:

$$t_{new} = \operatorname{argmax}_{c \in \{1, 2, \dots, C\}} \sum_{i=1}^k \delta(c, t_i)$$

where $\delta(c, t_i) = 1$ if $c = t_i$, or $\delta(c, t_i) = 0$ otherwise

kNN classification

- For discrete labels, classify a new test sample x_{new} as below:
 - Find k nearest (closest) training samples/instances
 - Find the most common class (label) out of these k neighbours
 - Assign this class to x_{new}
 - In case of a tie, assign randomly from tied classes
- $k = 3$



kNN classification

- For real-valued target labels, classify a new test sample x_{new} as below:

- Find k nearest (closest) training samples/instances
 - Find the average of k nearest target labels

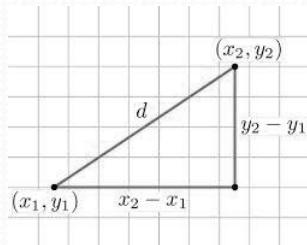
$$t_{new} = \frac{1}{k} \sum_{i=1}^k t_i$$

- Assign this target label to x_{new}
- Regression?

Distance measure

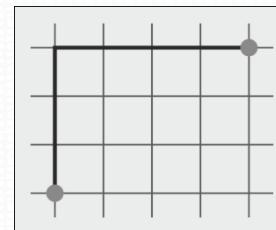
- To determine the nearest neighbours, a distance measure is required
 - kNN algorithm is flexible to use any distance measure
 - Thus, kNN can be used for any data type (images, text, audio, etc.) for which distance measure is available

Euclidean
distance



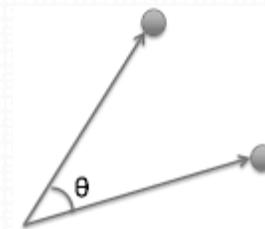
$$D_e(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^d (\mathbf{p}_i - \mathbf{q}_i)^2}$$

Manhattan
distance



$$D_m(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^d |\mathbf{p}_i - \mathbf{q}_i|$$

Cosine angle

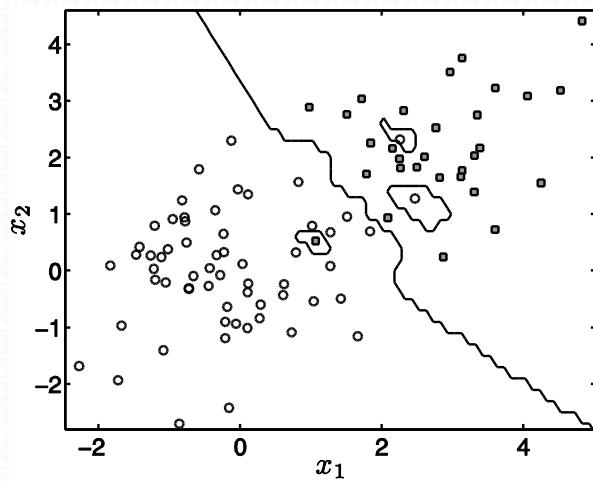


$$D_c(\mathbf{p}, \mathbf{q}) = \cos^{-1} \left(\frac{\sum_{i=1}^d \mathbf{p}_i \mathbf{q}_i}{\sqrt{\sum_{i=1}^d \mathbf{p}_i^2} \sqrt{\sum_{i=1}^d \mathbf{q}_i^2}} \right) / \pi$$

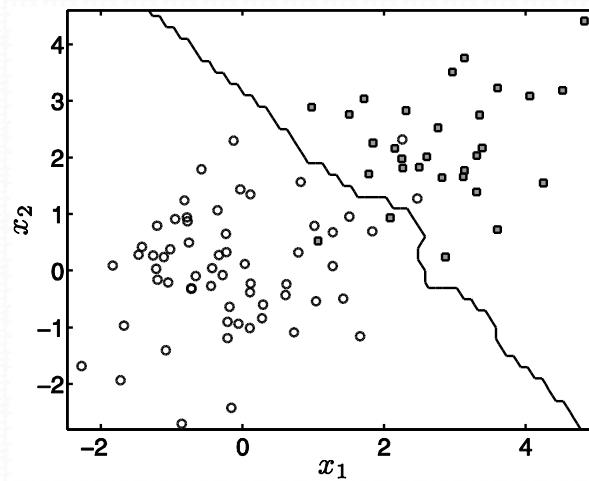
[10]

Role of k

- Line indicates decision boundary
- Noisy data



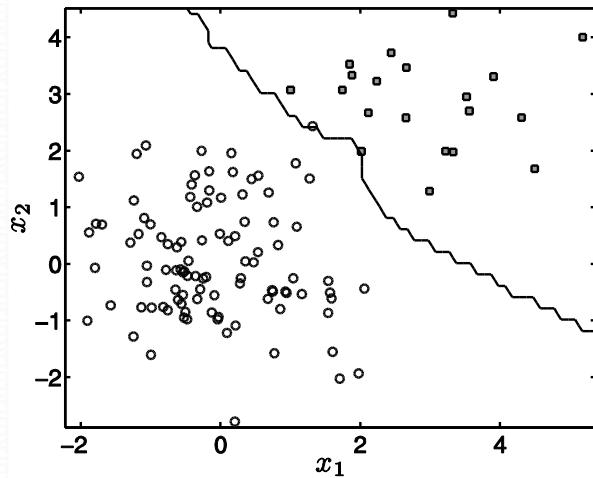
(a) Decision boundary when $K = 1$



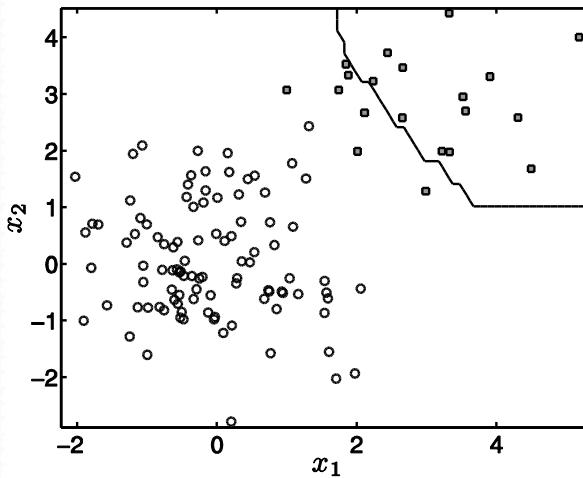
(b) Decision boundary when $K = 5$

Role of k

- Binary classification
 - 50 training samples vs 20 training samples
- Suitable value of k regularizes boundary and thus avoids over-fitting



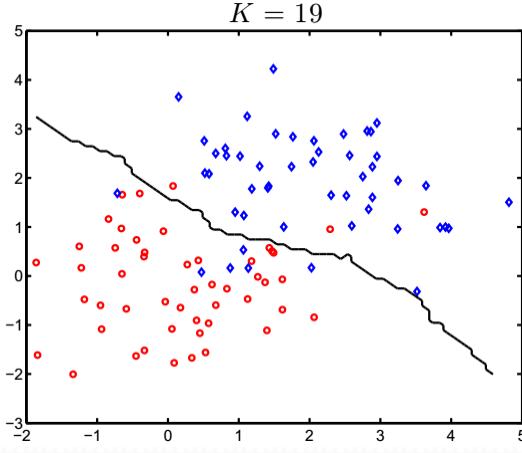
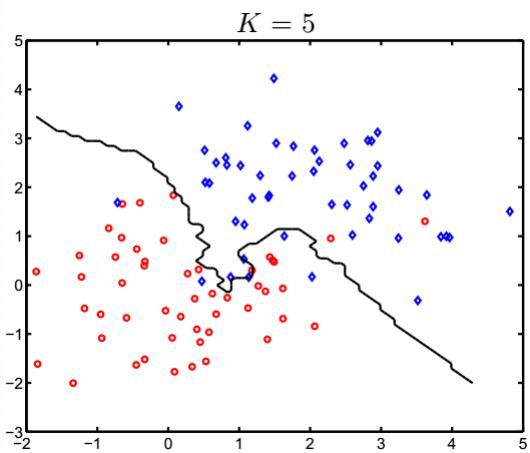
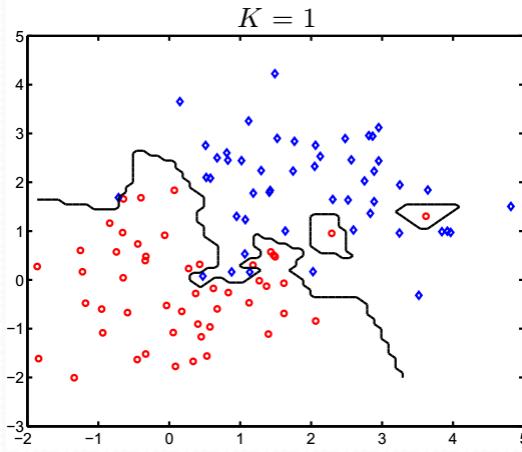
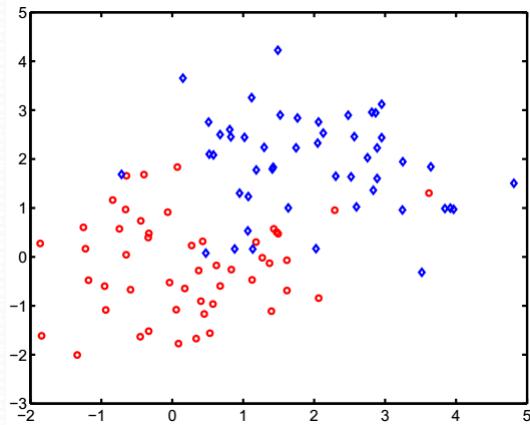
(a) Decision boundary when $K = 5$



(b) Decision boundary when $K = 39$

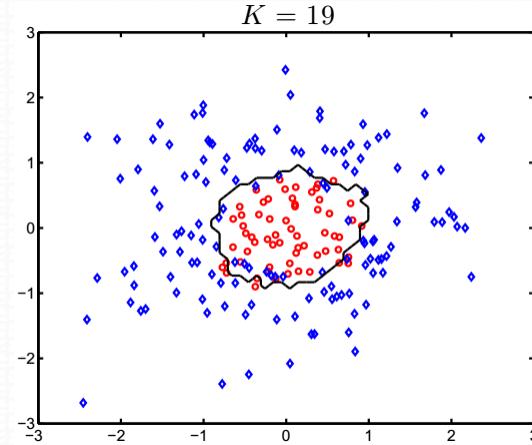
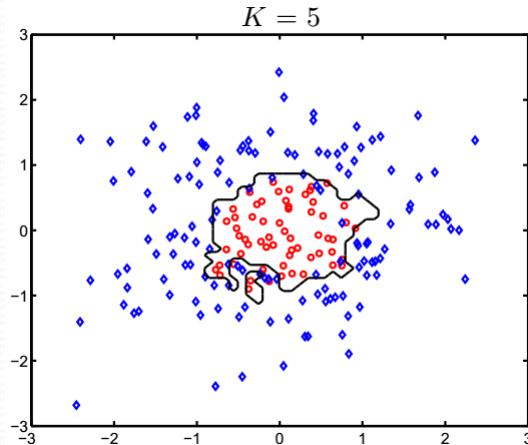
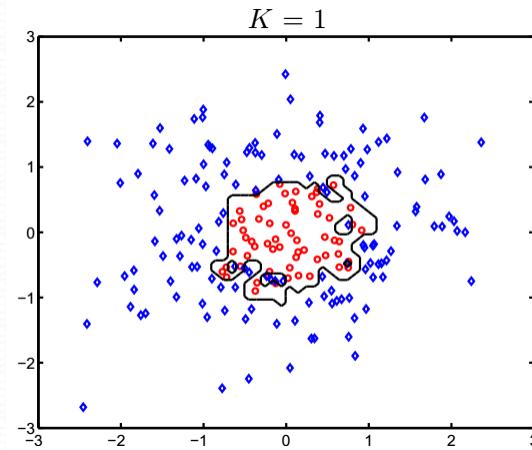
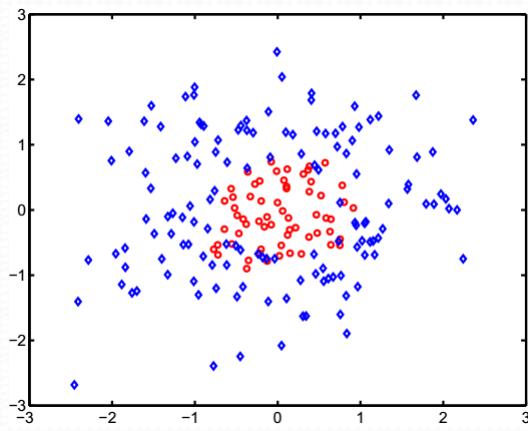
Role of k

- Boundary regularization



Role of k

- Boundary regularization

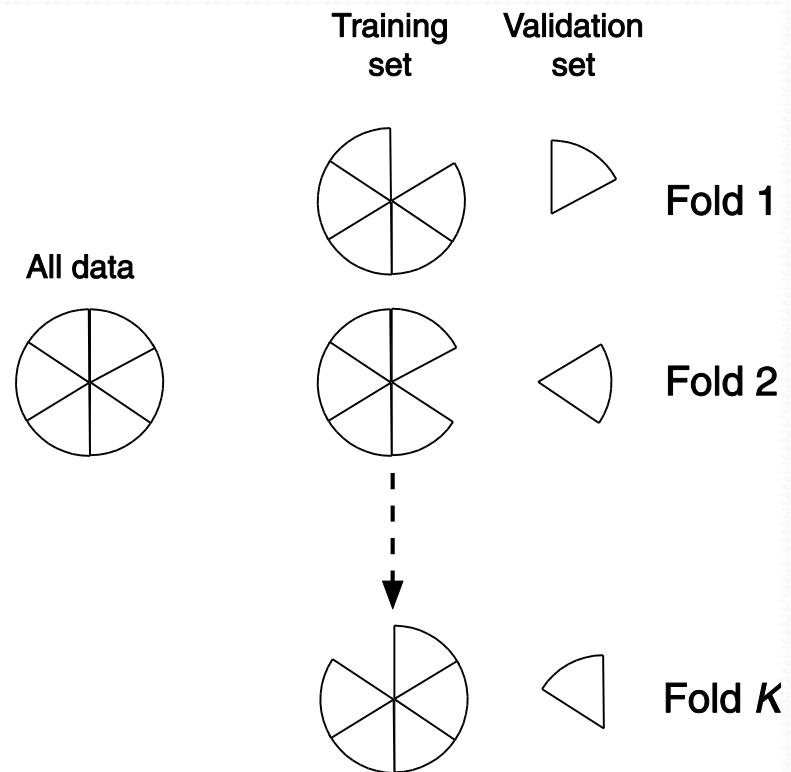


Difficulties with kNN

- Class imbalance : uneven number of objects in each class
 - Small classes will ‘disappear’ with increase in k value
 - Example: 5 training samples from class 1 and 100 training samples from class 2
 - For $k \geq 11$, class 1 ‘disappears’
- How do we choose k ?
 - Depends on data
 - Cross-validation
- Computational cost for classification can be high
 - Computation takes place at the time of classification
 - Calculates the distance of a test sample from all training samples
- There may be irrelevant attributes amongst the attributes

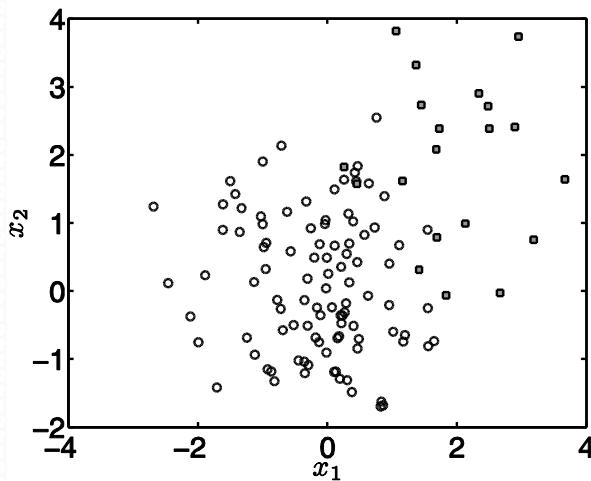
How to choose k ?

- Cross-validation
 - Split the data
 - Use some for training, some for testing
 - Need a measure of ‘goodness’ or ‘accuracy’ or ‘error’
 - Determine percentage misclassification error
 - Find k that minimises misclassification error

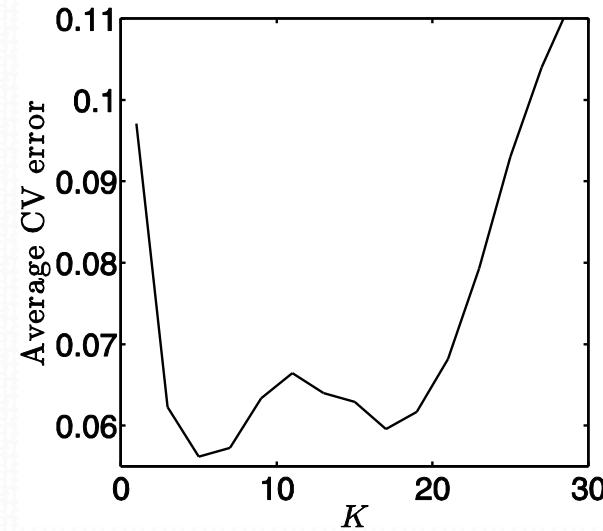


How to choose k ?

- Cross-validation
 - 10 fold CV repeated 100 times to remove the effect of cross-validation partitioning bias



(a) Binary classification dataset. Note the class imbalance: the grey squares class has fewer members than the white circles



(b) Average cross-validation error as K is increased

Characteristics of instance-based learning

- Instance-based learner is a *lazy learner*
 - It does all the work when the test sample is presented
- In contrast, *eager learner* builds a parameterized model from training samples, in advance of being presented with a test sample
- Instance-based learner produces *local* approximation to the target function
 - Finds a different approximation with each test instance

When to consider kNN?

- Not too many attributes (about 20)
- Lots of training data
- Advantages
 - Training is super fast
 - It can learn complex target functions
 - Doesn't loose information

Distance-weighted kNN

- How about weighting nearest neighbours more strongly than the farthest neighbours in determining the target label t_{new} ?
- Classification algorithm
 - Given a test instance \mathbf{x}_{new} to be classified:
 - Let $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k$ denote the k nearest instances to \mathbf{x}_{new} from the *training_examples*
 - Let t_1, t_2, \dots, t_k denote the labels of k nearest instances to \mathbf{x}_{new}
 - Assign target label t_{new} to \mathbf{x}_{new} as below:

$$t_{new} = \operatorname{argmax}_{c \in C} \sum_{i=1}^k w_i \delta(c, t_i)$$

$$\text{where } w_i = \frac{1}{D(\mathbf{x}_i, \mathbf{x}_{new})}$$

Distance-weighted kNN

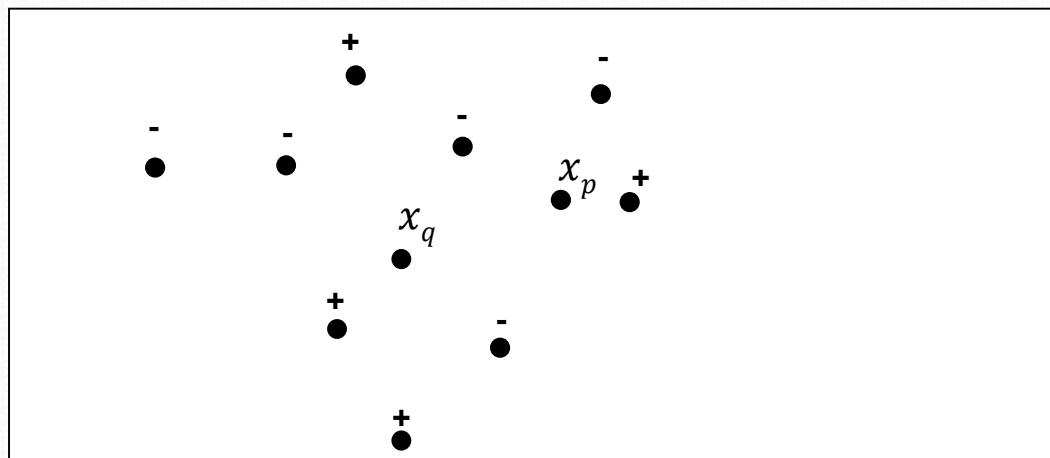
- This opens up the possibility to use all training samples as neighbours rather than only k neighbours
- As a result, the classifier becomes a global function approximation method
 - In contrast, typical kNN is a local function approximation method

Summary

- Instance-based classification
- Fast ‘learning’
- Simple setup
- Distance measure choice is flexible
- k is the only parameter that needs tuning
- Class imbalance in data needs consideration

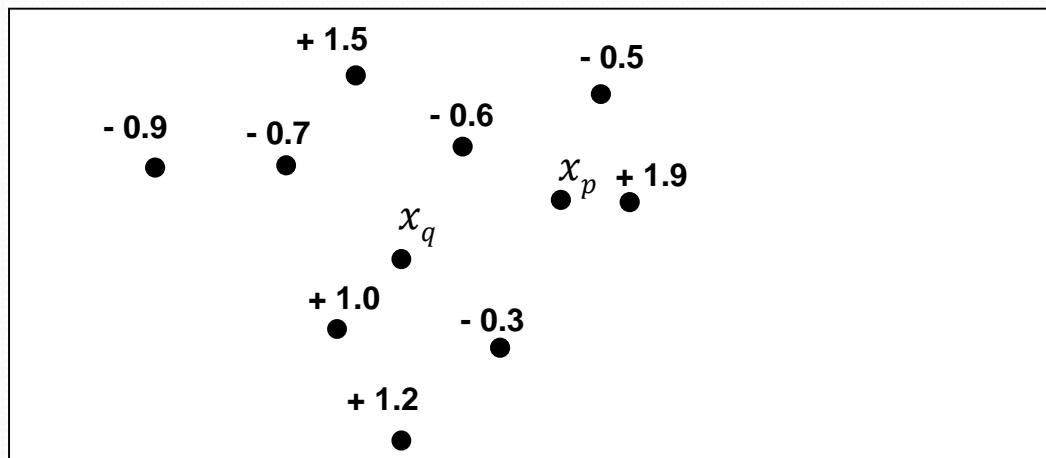
Exercise (ungraded)

- Assume a Boolean target function (i.e. binary classifier) and a two dimensional instance space. Determine how the kNN would classify the test instances x_p and x_q for $k = 1$, $k = 3$ and $k = 5$.



Exercise (ungraded)

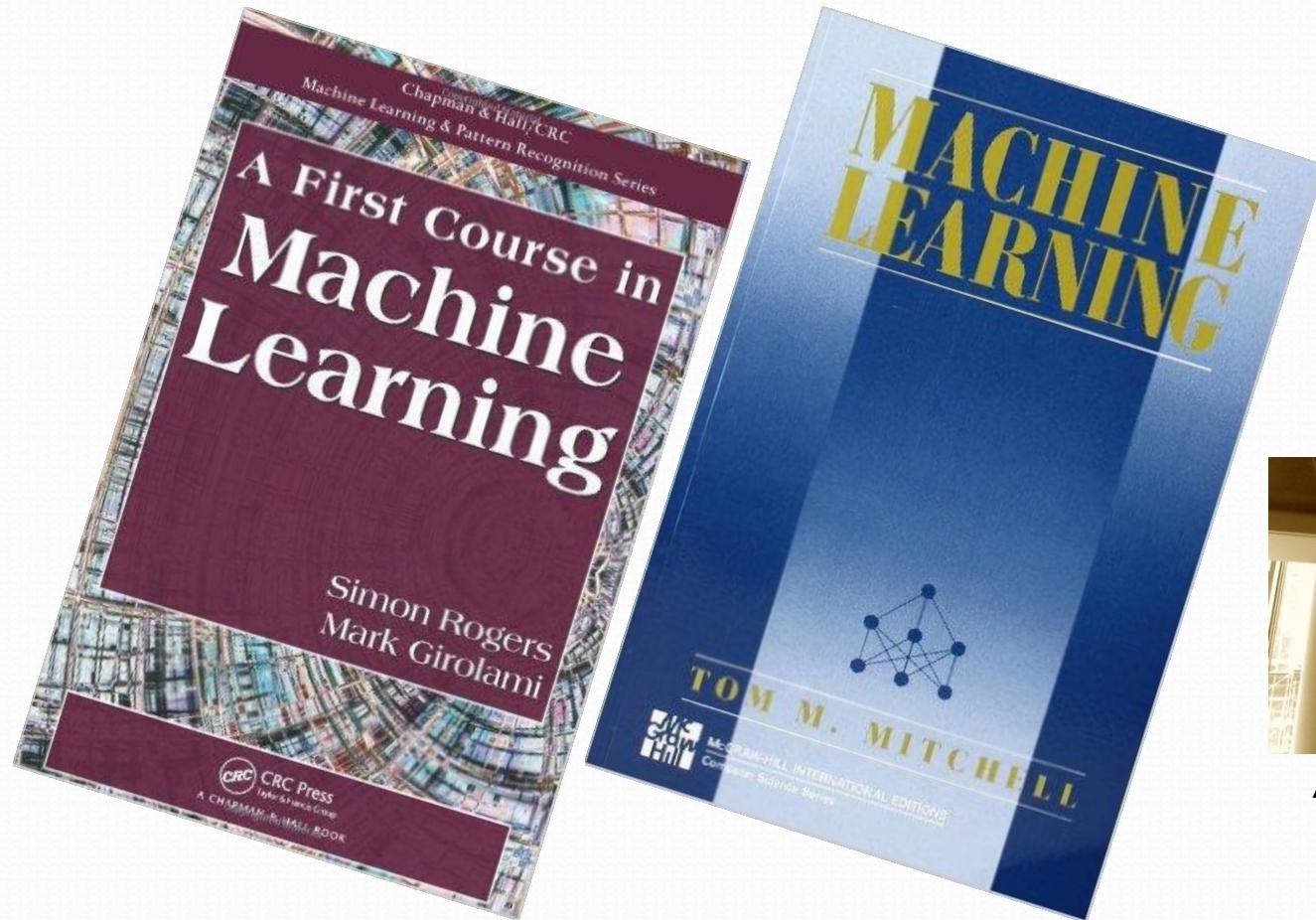
- In the diagram below, the numbers refer to the values taken by a real-valued target function. Calculate the values predicted for the target function at the test points x_p and x_q by kNN, with $k = 1$, $k = 3$, and $k = 5$.



Exercise (ungraded)

- Try MATLAB code – knnexample.m (from FCML book website)
- Try MATLAB code – knncv.m (from FCML book website)

C₃ R₁ E₂ D₂ I₁ T₁ S₁



Author's material
(Simon Rogers)



Thank You

Machine Learning, Machine Learning (extended)

7 – Unsupervised Learning: Clustering

Kashif Rajpoot

k.m.rajpoot@cs.bham.ac.uk

**School of Computer Science
University of Birmingham**

Outline

- Supervised vs unsupervised learning
- Clustering
- Similarity measure
- K-means clustering
- Kernelized k-means clustering
- Hierarchical agglomerative clustering

Supervised vs unsupervised learning

- So far, we have looked at supervised learning
- Supervised learning
 - The algorithm learns from $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ and t_1, t_2, \dots, t_N so that it can later classify \mathbf{x}_{new}
 - The availability of t_n makes learning a supervised task
- What if we only have \mathbf{x}_n , but not t_n ?
 - Unsupervised learning
- Clustering: create a grouping of objects
 - Each group containing “similar” objects

Clustering: examples

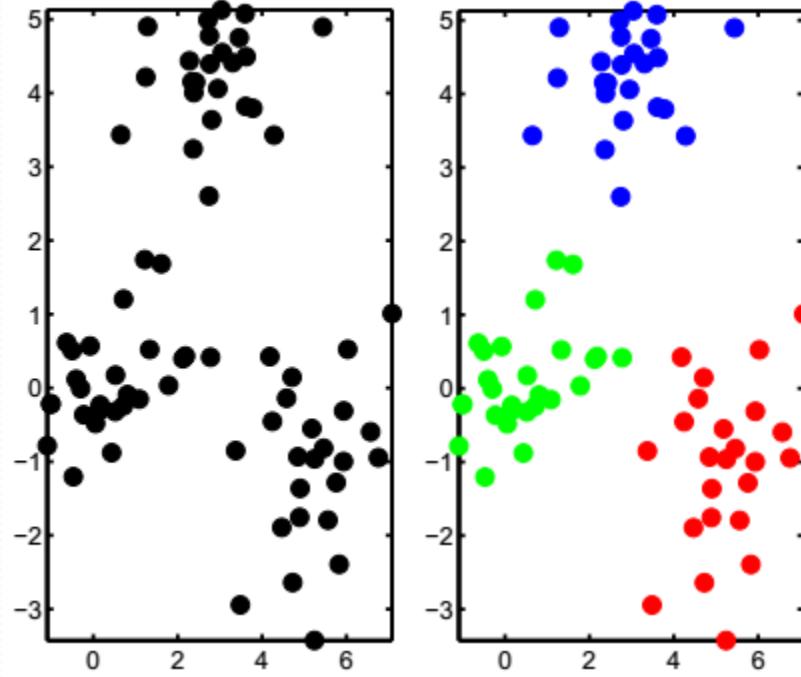
- Shopping recommendation system
 - Recommend products likely to be purchased by a customer
- Grouping people from social network
 - Based on user activity
- Brain region network analysis
 - Determine “similar” brain regions that “activate” together or “rest” together

Clustering

- Clustering: partition data into groups such that each group contains “similar” objects
 - Increase intra-group similarity
 - Similar objects should belong to same cluster

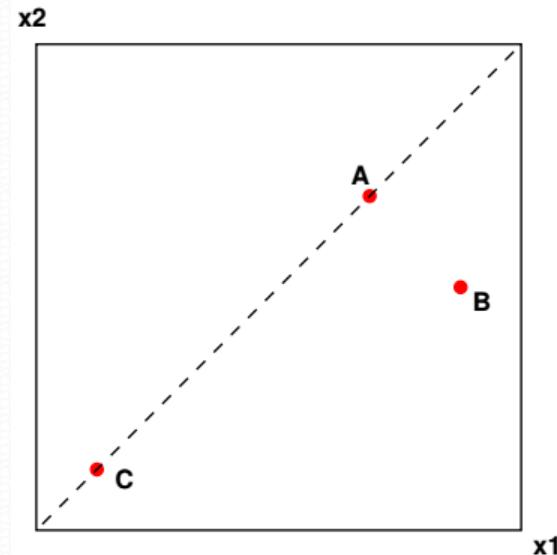
- How do we measure “similarity” between objects?
 - “closeness”

$$\begin{aligned} D_{ij} &= (\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j) \\ &= (x_i^{(1)} - x_j^{(1)})^2 + (x_i^{(2)} - x_j^{(2)})^2 \\ &= \sum_{d=1}^M (x_i^{(d)} - x_j^{(d)})^2 \end{aligned}$$



Clustering: Similarity measure

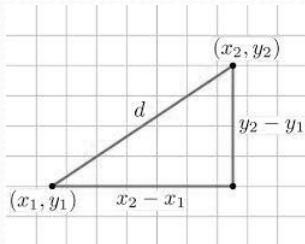
- Clustering: partition data into groups such that each group contains “similar” objects
 - Increase intra-group similarity
 - Similar objects should belong to same cluster
- How do we measure “similarity” between objects?
 - “closeness”
- Which points are “closest” or “most similar”?
- Choice of similarity measure is dependent on the nature of data and your application



Clustering: Similarity measure

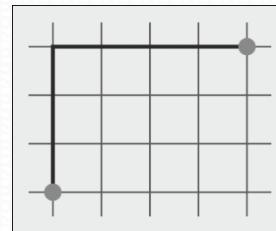
- How to estimate the distance (i.e. similarity) between two objects/points \mathbf{p} and \mathbf{q} ?
 - Various measures can be used
- K-means algorithm is flexible to use any distance measure

Euclidean distance



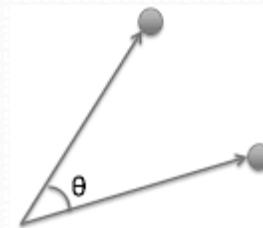
$$D_e(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{d=1}^M (p^{(d)} - q^{(d)})^2}$$

Manhattan distance



$$D_m(\mathbf{p}, \mathbf{q}) = \sum_{d=1}^M |p^{(d)} - q^{(d)}|$$

Cosine angle

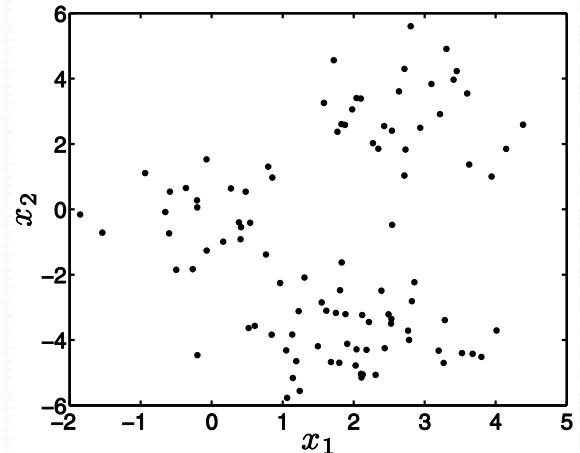


$$D_c(\mathbf{p}, \mathbf{q}) = \cos^{-1} \left(\frac{\sum_{d=1}^M p^{(d)} q^{(d)}}{\sqrt{\sum_{d=1}^M p^{(d)}^2} \sqrt{\sum_{d=1}^M q^{(d)}^2}} \right) / \pi$$

K-means clustering

- Consider that there are total K clusters
- In 2D, each cluster k is represented by a cluster centre (i.e. mean)

$$\boldsymbol{\mu}_k = [\mu_k^{(1)}, \mu_k^{(2)}]^T$$



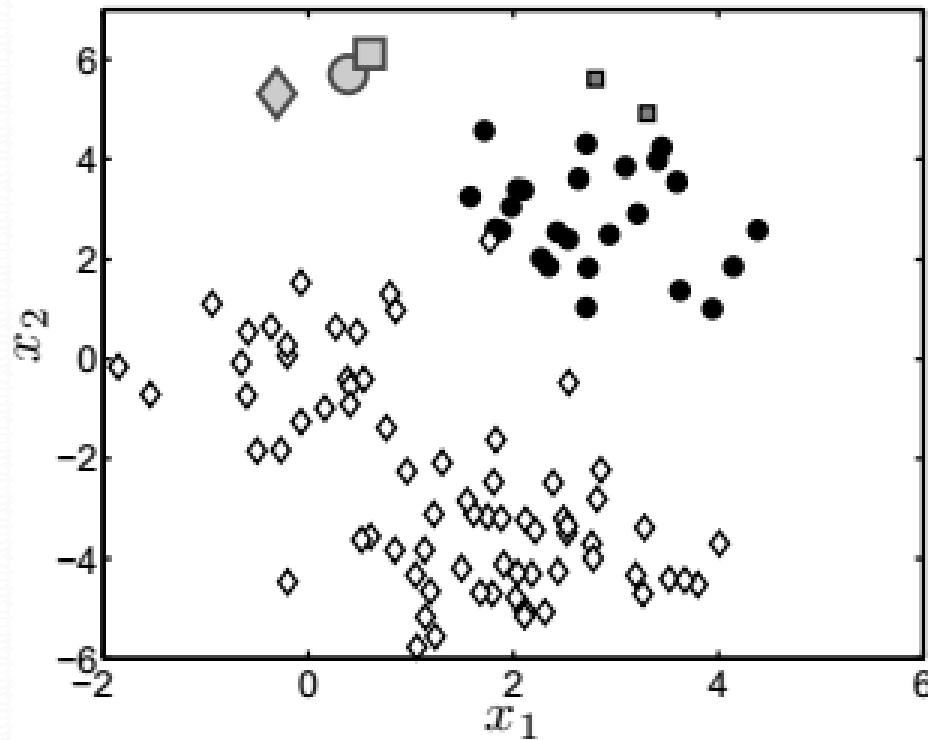
- Each object \mathbf{x}_n is assigned to its closest cluster k on the basis of its distance to cluster mean $\boldsymbol{\mu}_k$
- Distance to cluster k can be estimated in various ways
 - For example: squared Euclidean distance
- $D_{nk} = (\mathbf{x}_n - \boldsymbol{\mu}_k)^T(\mathbf{x}_n - \boldsymbol{\mu}_k)$
- No analytical solution for finding the closest cluster mean $\boldsymbol{\mu}_k$ for all objects
 - Iterative solution

K-means clustering

- Iterative algorithm: Method 1
 1. Randomly initialize cluster means $\mu_1, \mu_2, \dots, \mu_K$
 2. Compute distances $D_{n1}, D_{n2}, \dots, D_{nK}$ for each object x_n to all K cluster means
 3. Assign the object x_n to cluster k with lowest distance D_{nk}
 - i.e. assign x_n to its closest cluster k with mean μ_k
 4. Update each cluster mean μ_k , to represent the mean of newly updated cluster
$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N_k} x_n$$
where N_k is the number of objects in k^{th} cluster
 5. Stop if assignments don't change, or reached max number of iterations, else jump to step 2

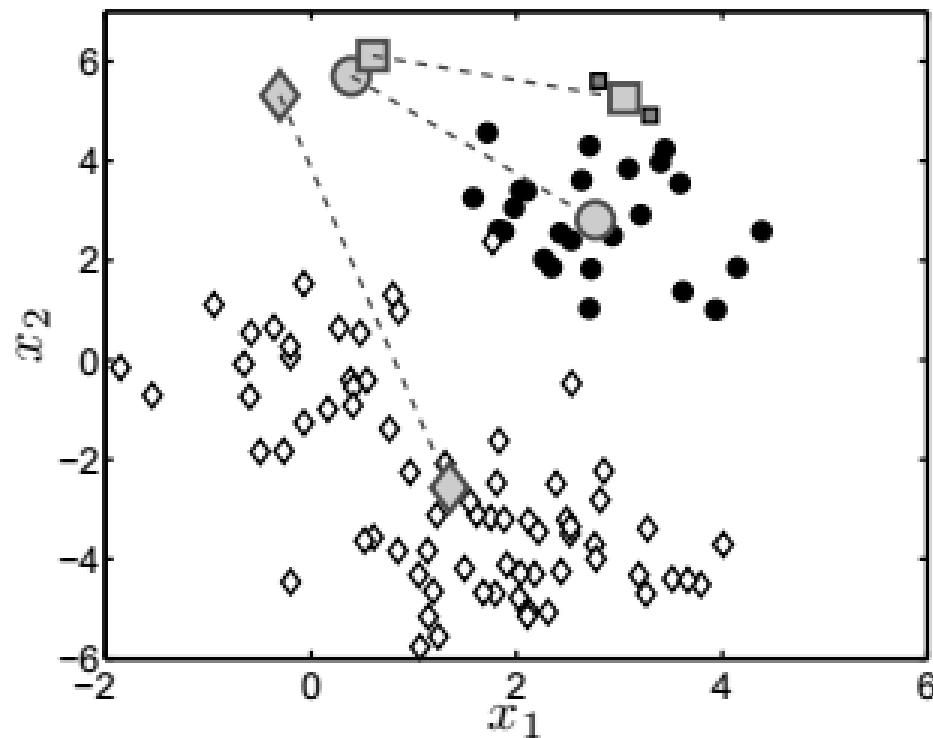
K-means clustering

- Randomly initialize cluster means $\mu_1, \mu_2, \dots, \mu_k$
- Assign each x_n to its closest cluster with mean μ_k



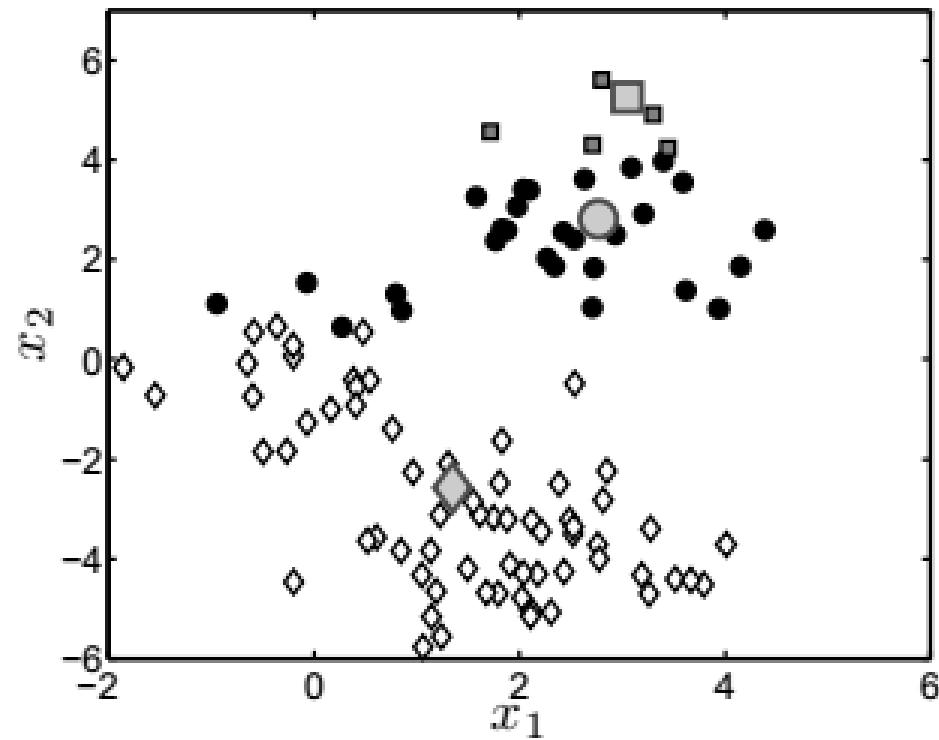
K-means clustering

- Update means $\mu_k = \frac{1}{N_k} \sum_{n=1}^{N_k} x_n$



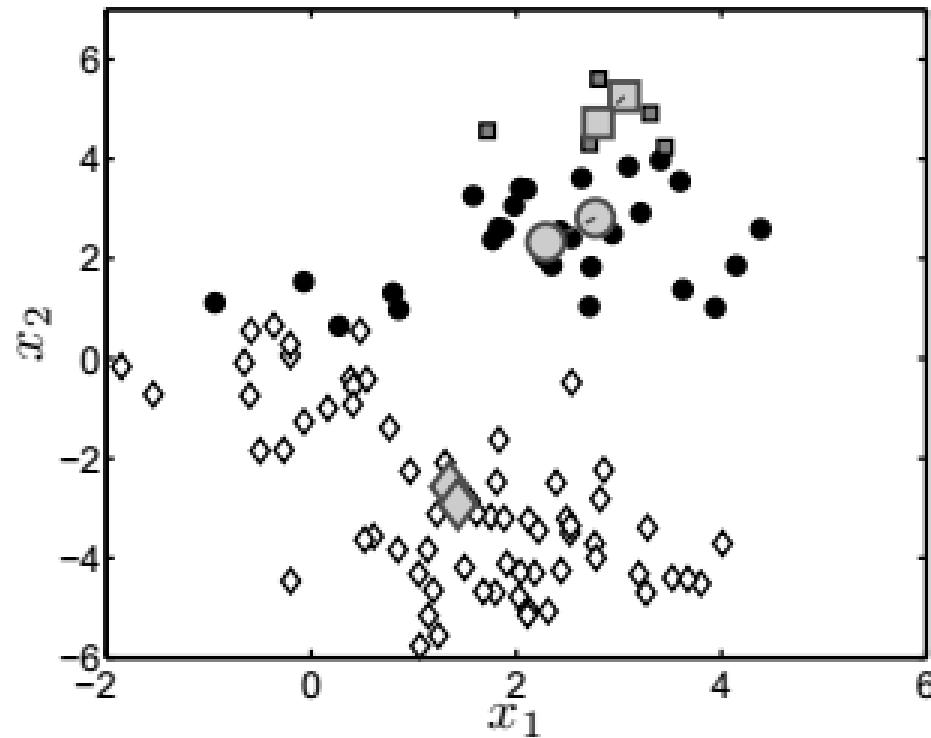
K-means clustering

- Assign each x_n to its closest cluster with mean μ_k



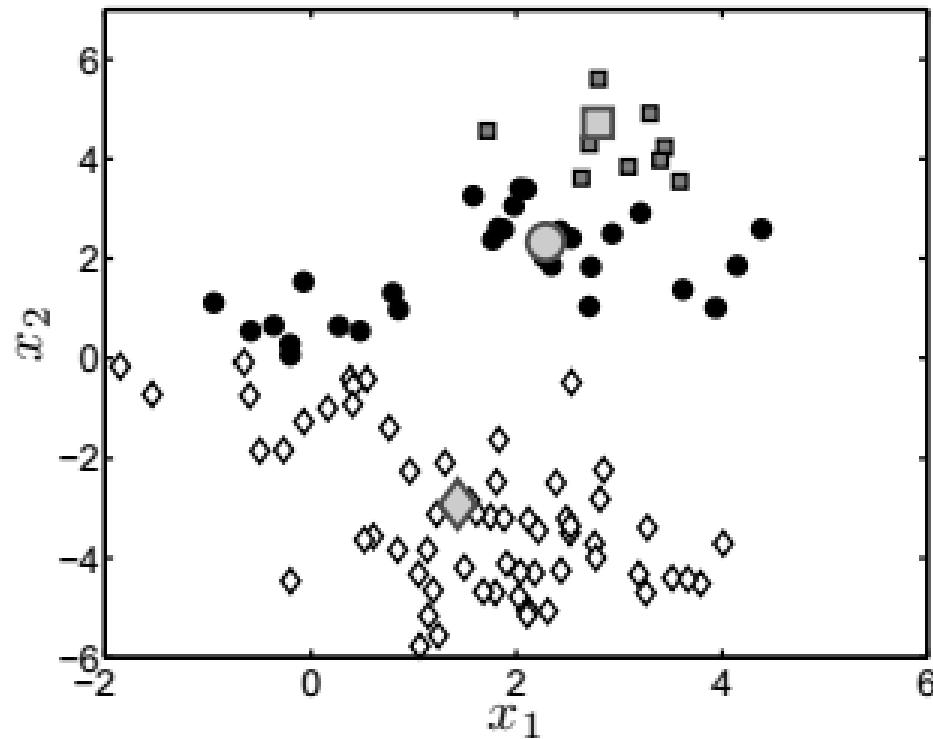
K-means clustering

- Update means $\mu_k = \frac{1}{N_k} \sum_{n=1}^{N_k} x_n$



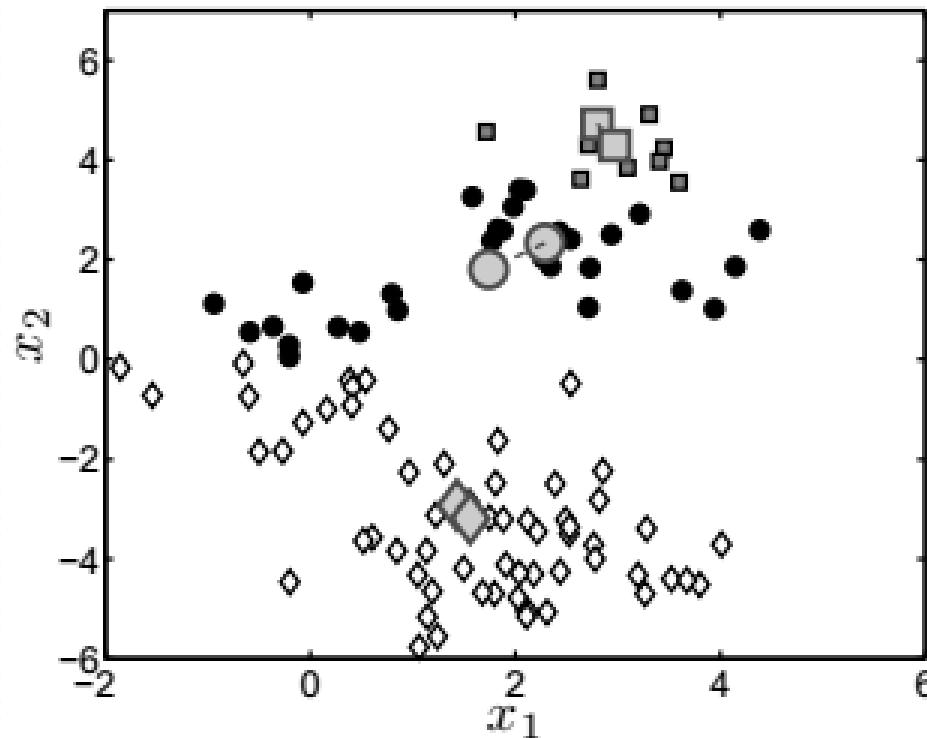
K-means clustering

- Assign each x_n to its closest cluster with mean μ_k



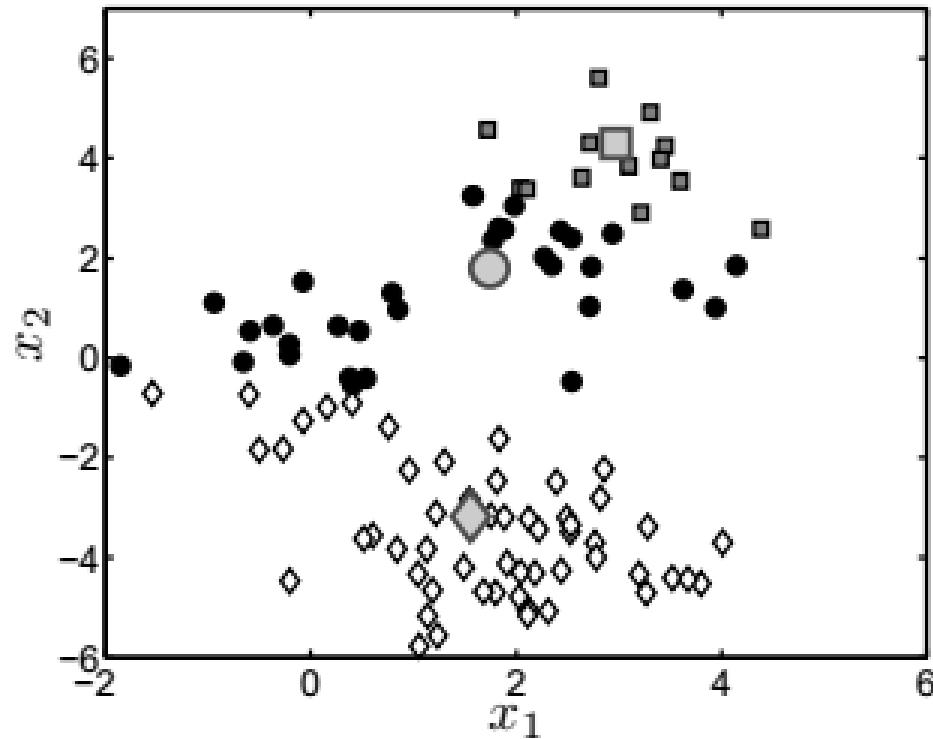
K-means clustering

- Update means $\mu_k = \frac{1}{N_k} \sum_{n=1}^{N_k} x_n$



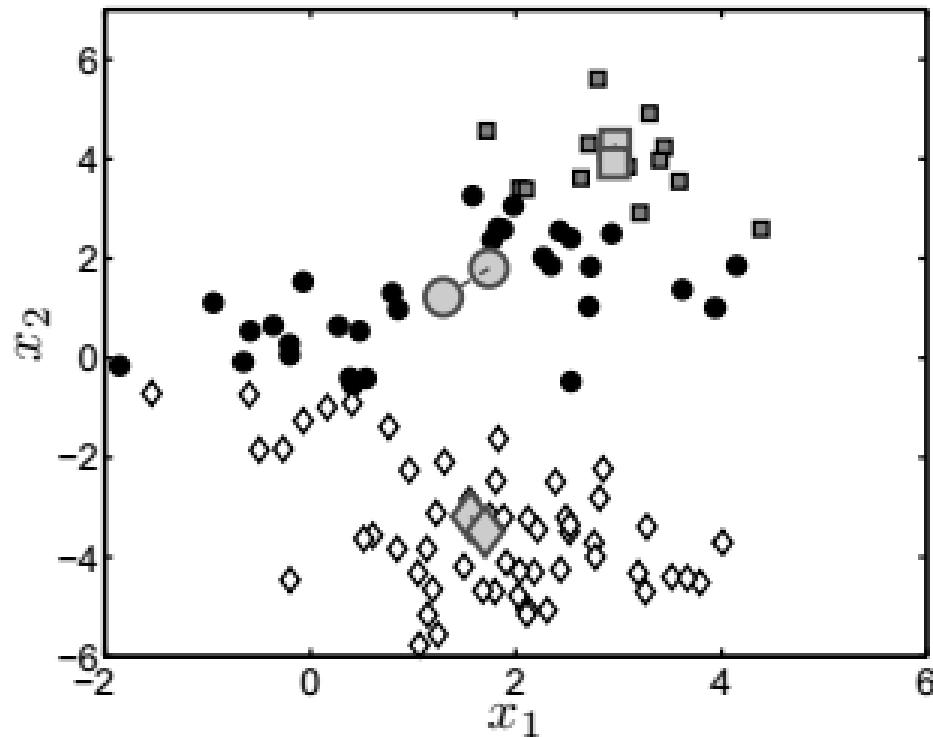
K-means clustering

- Assign each x_n to its closest cluster with mean μ_k



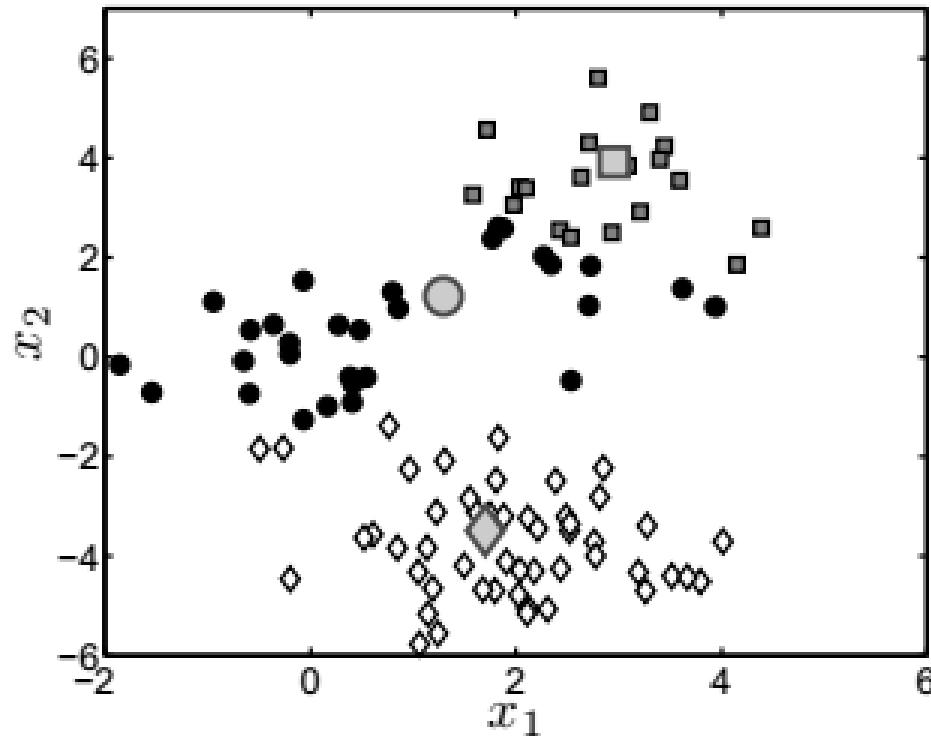
K-means clustering

- Update means $\mu_k = \frac{1}{N_k} \sum_{n=1}^{N_k} x_n$



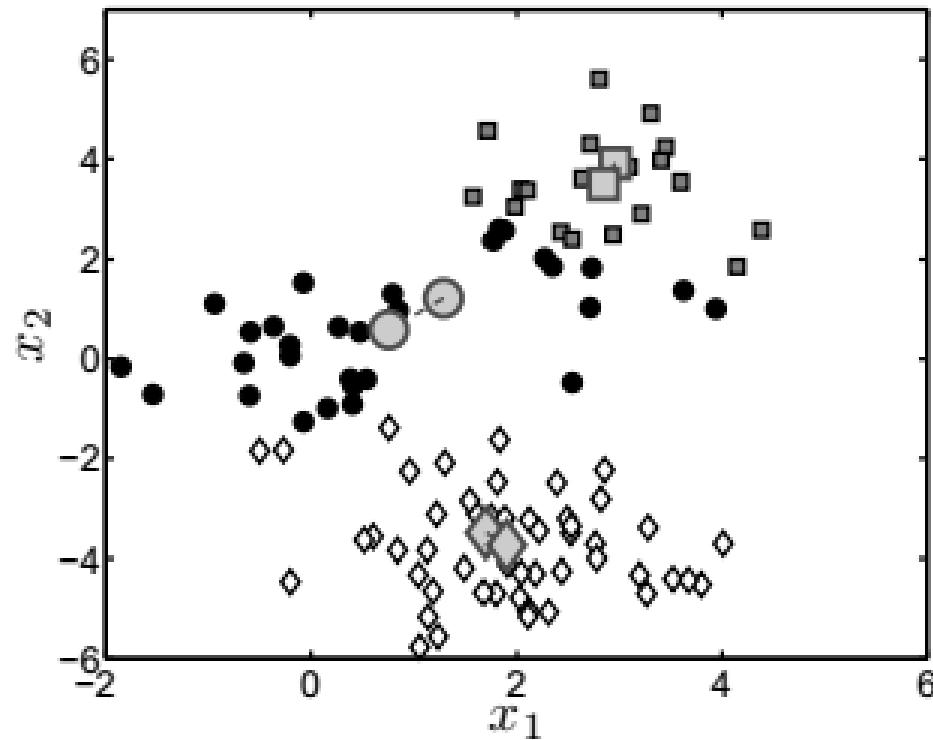
K-means clustering

- Assign each x_n to its closest cluster with mean μ_k



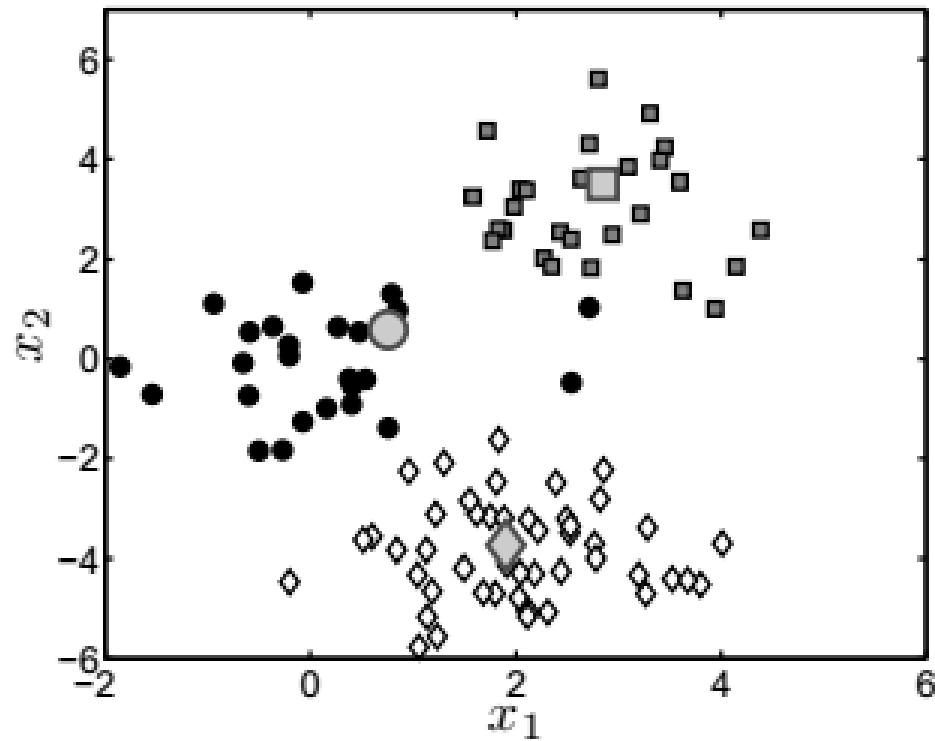
K-means clustering

- Update means $\mu_k = \frac{1}{N_k} \sum_{n=1}^{N_k} x_n$



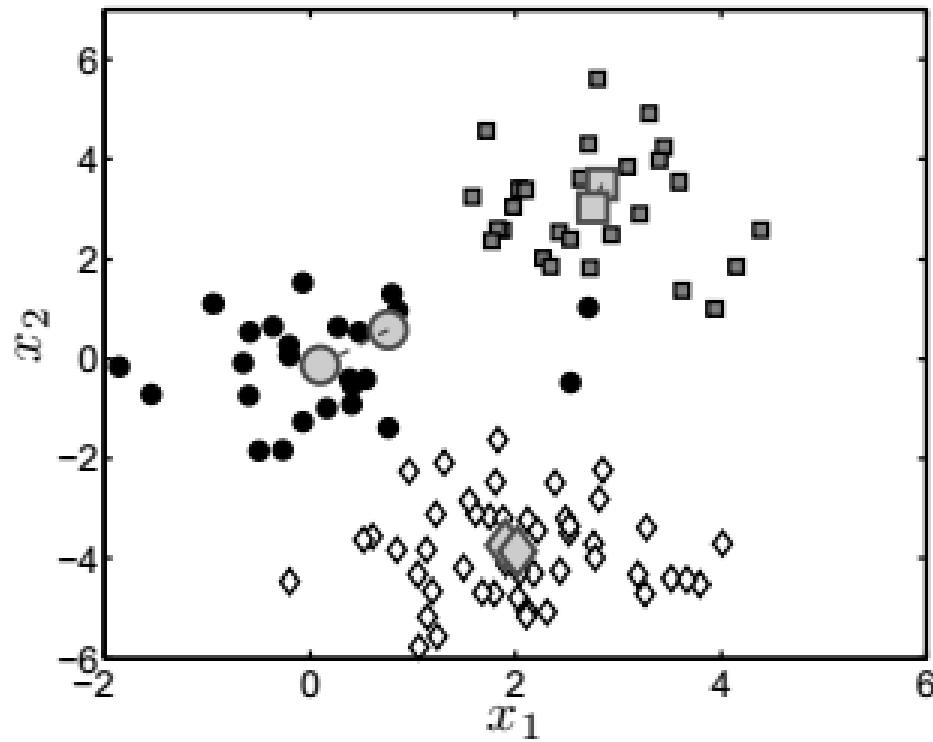
K-means clustering

- Assign each x_n to its closest cluster with mean μ_k



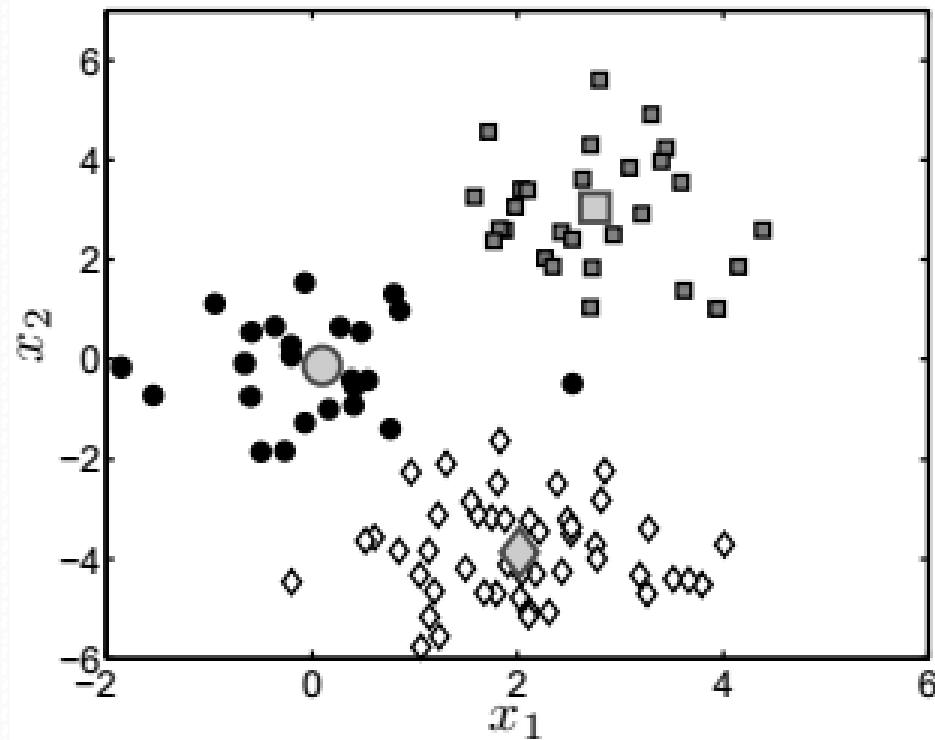
K-means clustering

- Update means $\mu_k = \frac{1}{N_k} \sum_{n=1}^{N_k} x_n$



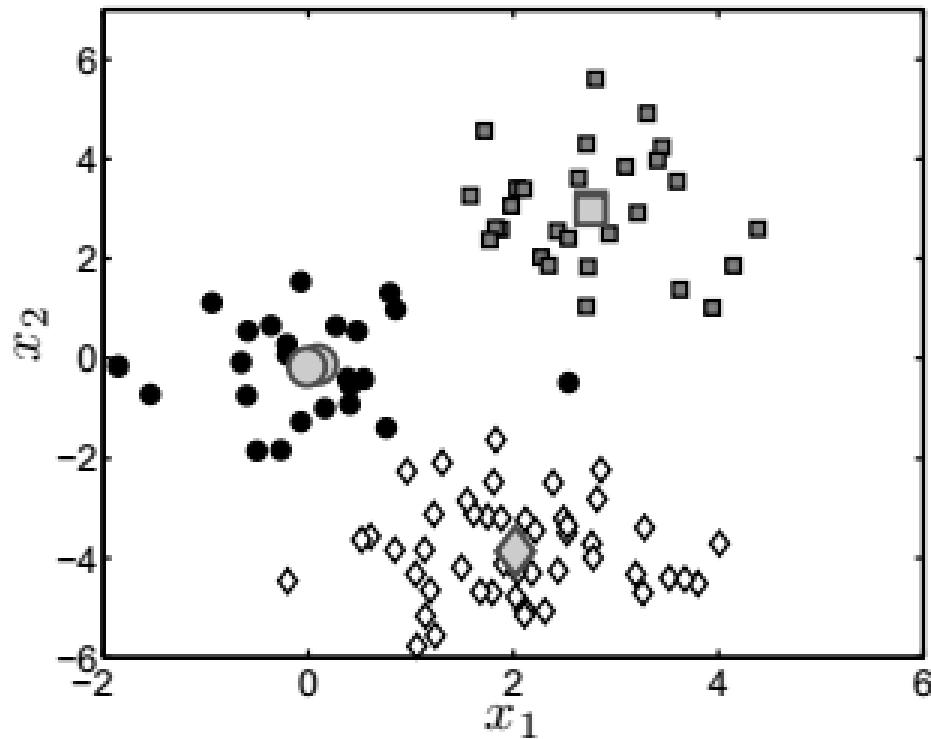
K-means clustering

- Assign each x_n to its closest cluster with mean μ_k



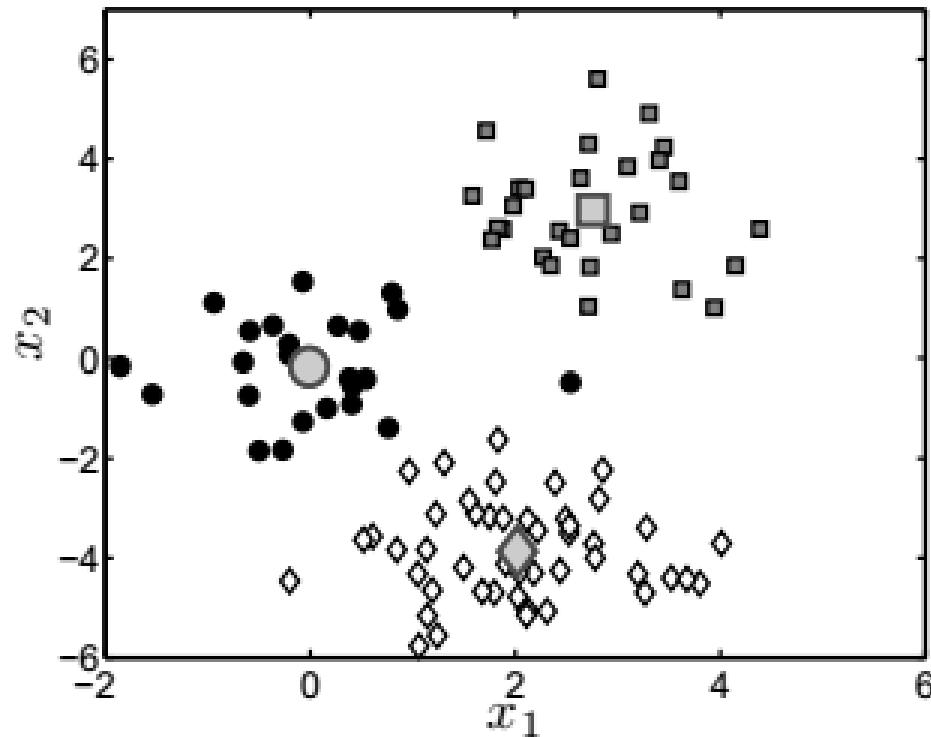
K-means clustering

- Update means $\mu_k = \frac{1}{N_k} \sum_{n=1}^{N_k} x_n$



K-means clustering

- Solution at convergence

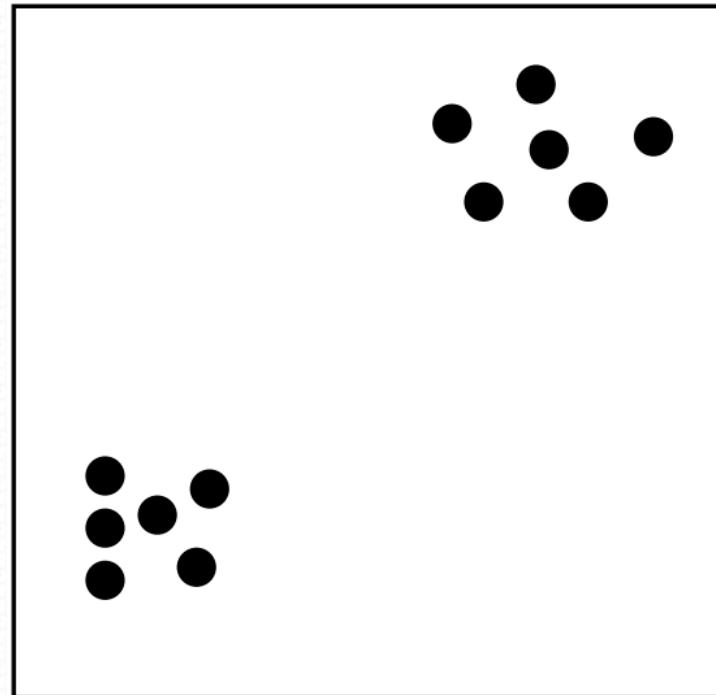


K-means clustering

- Iterative algorithm: Method 2
 1. Randomly assign each object \mathbf{x}_n to one of K clusters
 2. Compute cluster means $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K$ for each cluster k , to represent the mean of newly updated cluster
$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N_k} \mathbf{x}_n$$
where N_k is the number of objects in k^{th} cluster
 3. Compute distances $D_{n1}, D_{n2}, \dots, D_{nK}$ for each object \mathbf{x}_n to all K cluster means
 4. Assign the object \mathbf{x}_n to cluster k with lowest distance D_{nk}
 - i.e. assign \mathbf{x}_n to its closest cluster k with mean $\boldsymbol{\mu}_k$
 5. Stop iterations if assignments don't change, or maximum number of iterations reached, else jump to step 2

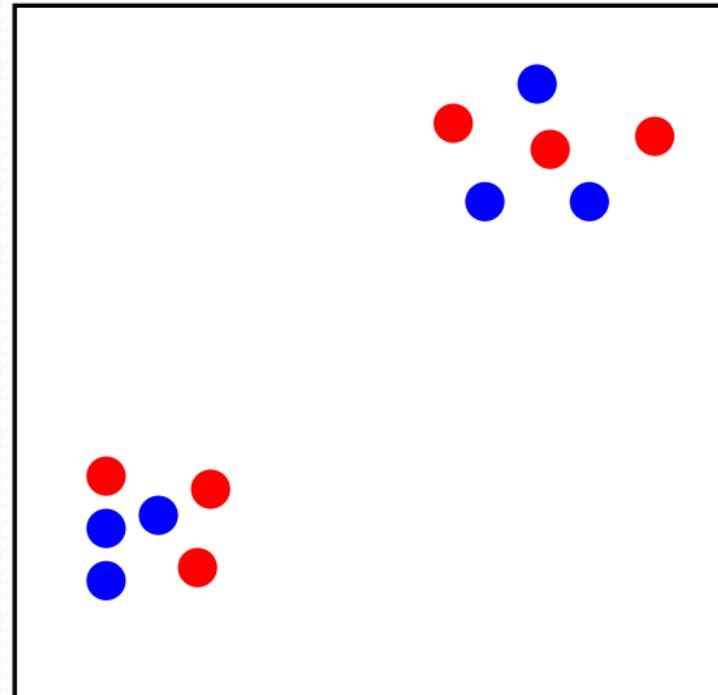
K-means clustering

- Cluster data points with Method 2



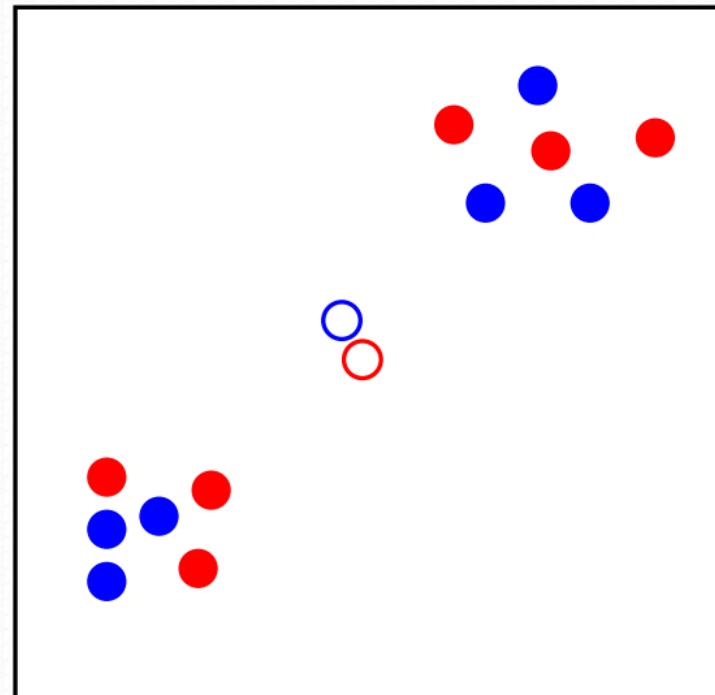
K-means clustering

- Randomly assign each object x_n to one of K clusters



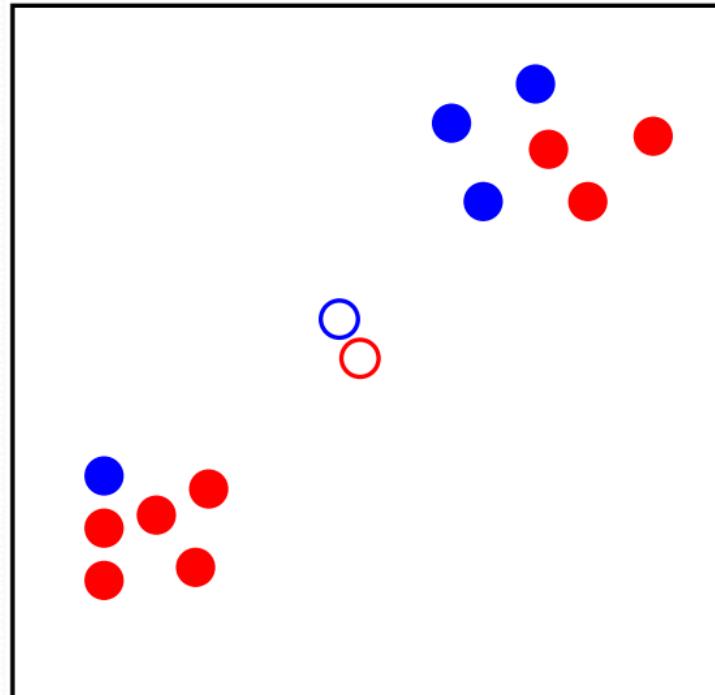
K-means clustering

- Update means $\mu_k = \frac{1}{N_k} \sum_{n=1}^{N_k} x_n$



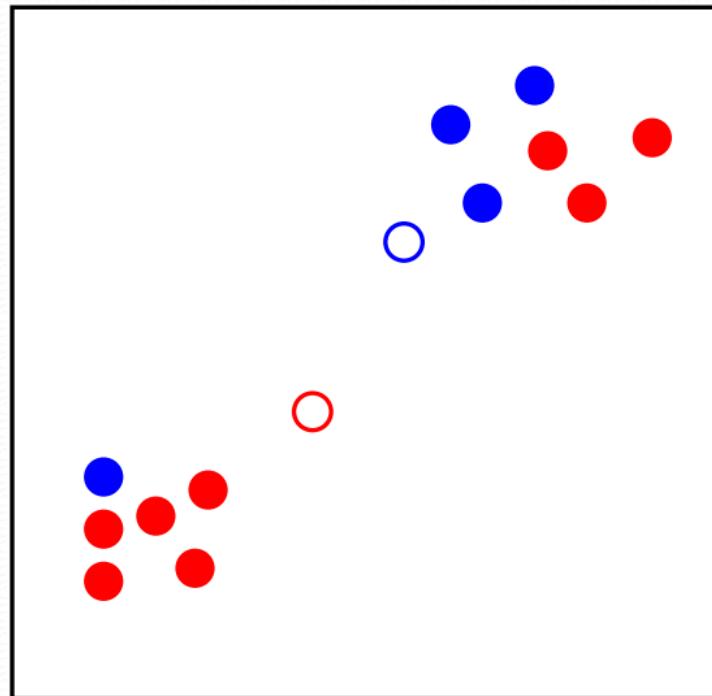
K-means clustering

- Assign each x_n to its closest cluster with mean μ_k



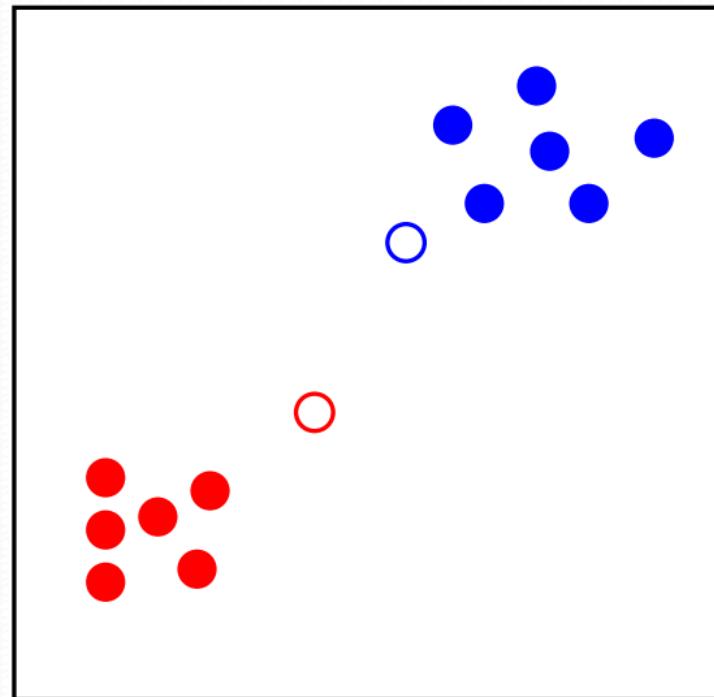
K-means clustering

- Update means $\mu_k = \frac{1}{N_k} \sum_{n=1}^{N_k} x_n$



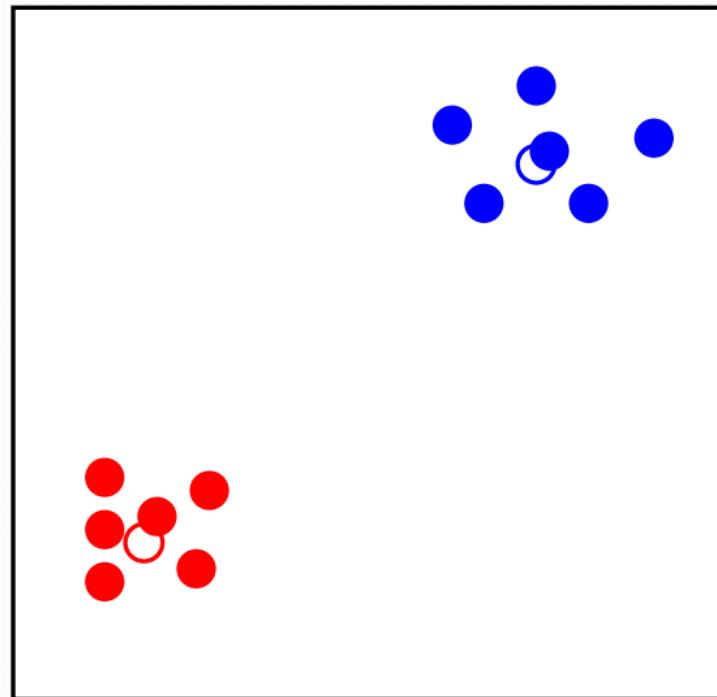
K-means clustering

- Assign each x_n to its closest cluster with mean μ_k



K-means clustering

- Update means $\mu_k = \frac{1}{N_k} \sum_{n=1}^{N_k} x_n$



K-means clustering

- The k-means clustering algorithm converges to a local minimum of the following *intra-group distance estimate*:

$$\mathcal{D} = \sum_{k=1}^K \sum_{n=1}^{N_k} (\mathbf{x}_n - \boldsymbol{\mu}_k)^T (\mathbf{x}_n - \boldsymbol{\mu}_k)$$

- i.e. total distance between all the objects and their respective cluster means
- Global optimum vs local optimum
 - Depends on random initialization
- Repeat runs of k-means clustering, pick one with lowest \mathcal{D}

How to choose K?

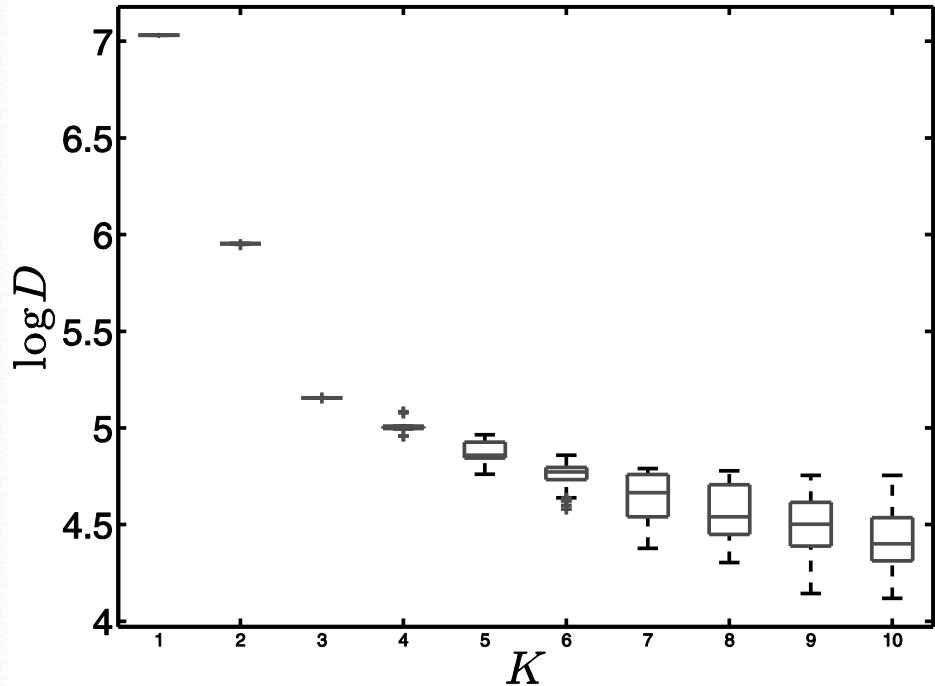
- Model selection

- 50 repeat runs

- \mathcal{D} decreases with increasing K
 - Why?

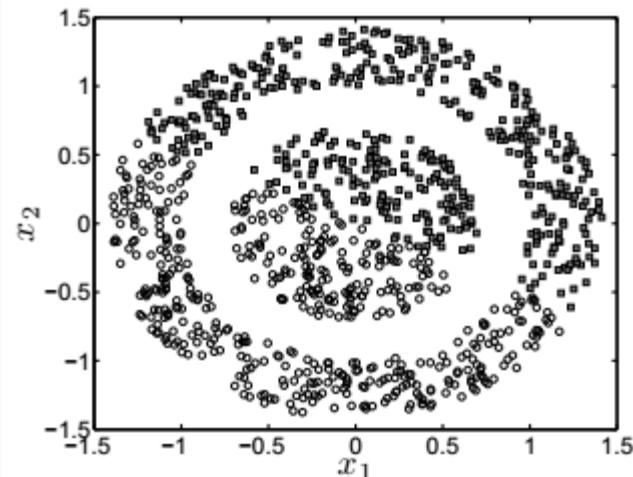
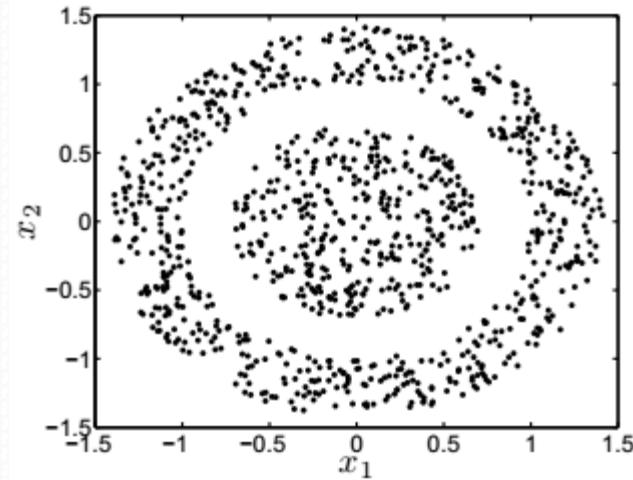
- $N = K$?

- K often depends on application/problem/data



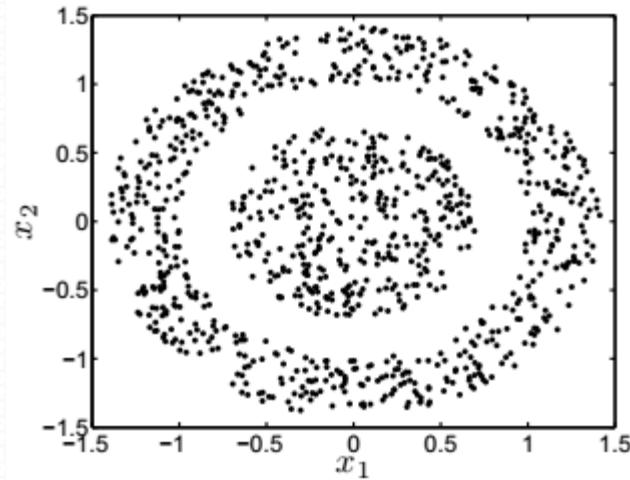
K-means limitations

- K-means assumes clear cluster structure
 - What if it's lacking?
- Cluster means?
- Do objects conform to the similarity concept that K-means clustering heavily relies on?
- Outer cluster can't be represented by a single cluster mean



Data transformation

- How about transforming data in to a new space where it can be separable?
 - $\mathbf{x} \rightarrow \phi(\mathbf{x})$
- For this example, consider $\phi(\mathbf{x}) = x_n^{(1)^2} + x_n^{(2)^2}$?
- In practice, there is a very neat trick which doesn't even require the explicit data transformation
 - Kernel trick
- Kernel function (a kind of similarity measure)
 - A function that is equivalent to the dot product of vectors in the transformed space
 - $k(\mathbf{x}_m, \mathbf{x}_n) = \phi(\mathbf{x}_m)^T \phi(\mathbf{x}_n)$



Kernel function

- There is a number of off-the-shelf kernels that have been shown to work well
- Linear kernel
 - $k(\mathbf{x}_m, \mathbf{x}_n) = \mathbf{x}_m^T \mathbf{x}_n$
- Gaussian kernel
 - $k(\mathbf{x}_m, \mathbf{x}_n) = \exp\{-\gamma(\mathbf{x}_m - \mathbf{x}_n)^T(\mathbf{x}_m - \mathbf{x}_n)\}$
 - $k(\mathbf{x}_m, \mathbf{x}_n) = \exp\{-\gamma\|\mathbf{x}_m - \mathbf{x}_n\|^2\}$
- Polynomial kernel
 - $k(\mathbf{x}_m, \mathbf{x}_n) = (\mathbf{x}_m^T \mathbf{x}_n + c)^\beta$

Kernelized k-means

- Can we kernelize k-means?
- *Kernel function*
 - A function that corresponds to inner product of vectors in some other transformed space
- As long as an algorithm has data appearing only in inner products in model learning, kernels can be used

$$k(\mathbf{x}_m, \mathbf{x}_n) = \phi(\mathbf{x}_m)^T \phi(\mathbf{x}_n)$$

Kernelized k-means

- Distance estimate (squared Euclidean)

$$D_{nk} = (\mathbf{x}_n - \boldsymbol{\mu}_k)^T (\mathbf{x}_n - \boldsymbol{\mu}_k)$$

- Cluster mean calculation

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N_k} \mathbf{x}_n$$

- Distance can be estimated as:

$$D_{nk} = \left(\mathbf{x}_n - \frac{1}{N_k} \sum_{m=1}^{N_k} \mathbf{x}_m \right)^T \left(\mathbf{x}_n - \frac{1}{N_k} \sum_{m=1}^{N_k} \mathbf{x}_m \right)$$

$$D_{nk} = \mathbf{x}_n^T \mathbf{x}_n - \frac{2}{N_k} \sum_{m=1}^{N_k} \mathbf{x}_m^T \mathbf{x}_n + \left(\frac{1}{N_k} \right)^2 \sum_{m=1}^{N_k} \sum_{l=1}^{N_k} \mathbf{x}_m^T \mathbf{x}_l$$

Kernelized k-means

- Distance can be estimated as:

$$D_{nk} = \mathbf{x}_n^T \mathbf{x}_n - \frac{2}{N_k} \sum_{m=1}^{N_k} \mathbf{x}_m^T \mathbf{x}_n + \left(\frac{1}{N_k} \right)^2 \sum_{m=1}^{N_k} \sum_{l=1}^{N_k} \mathbf{x}_m^T \mathbf{x}_l$$

- With kernel trick, dot products could be replaced:

$$D_{nk}$$

$$= k(\mathbf{x}_n, \mathbf{x}_n) - \frac{2}{N_k} \sum_{m=1}^{N_k} k(\mathbf{x}_m, \mathbf{x}_n) + \left(\frac{1}{N_k} \right)^2 \sum_{m=1}^{N_k} \sum_{l=1}^{N_k} k(\mathbf{x}_m, \mathbf{x}_l)$$

- Let's recall μ_k will be

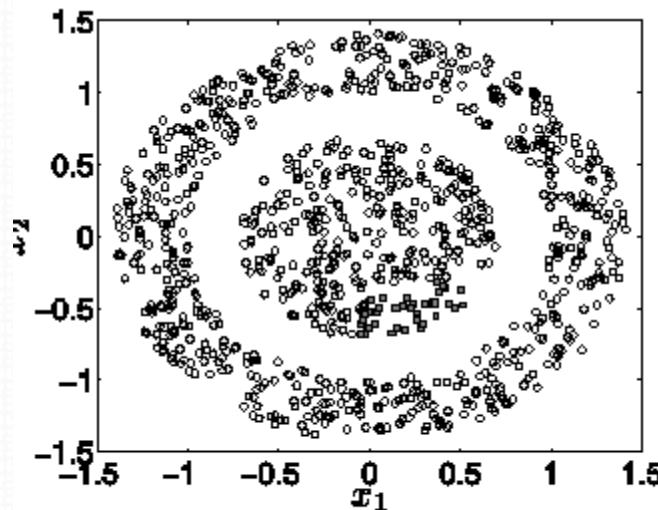
$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N_k} \phi(\mathbf{x}_n)$$

but what's $\phi(\mathbf{x}_n)$?

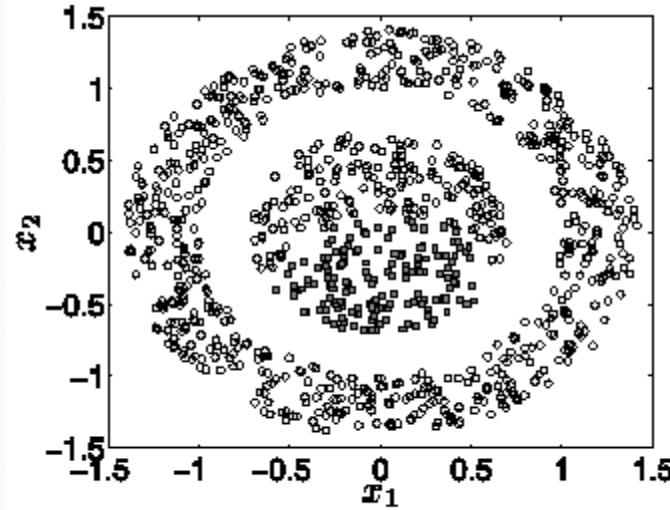
Kernelized k-means

- Algorithm:
 1. Randomly assign each object \mathbf{x}_n to one of K clusters
 2. Compute distances $D_{n1}, D_{n2}, \dots, D_{nK}$ for each object \mathbf{x}_n , using the kernel trick
$$D_{nk} = k(\mathbf{x}_n, \mathbf{x}_n) - \frac{2}{N_k} \sum_{m=1}^{N_k} k(\mathbf{x}_m, \mathbf{x}_n) + \left(\frac{1}{N_k} \right)^2 \sum_{m=1}^{N_k} \sum_{l=1}^{N_k} k(\mathbf{x}_m, \mathbf{x}_l)$$
 3. Assign the object \mathbf{x}_n to cluster k with lowest distance D_{nk}
 - i.e. assign \mathbf{x}_n to its closest cluster
 4. Stop iterations if assignments don't change, or maximum number of iterations reached, else jump to step 2

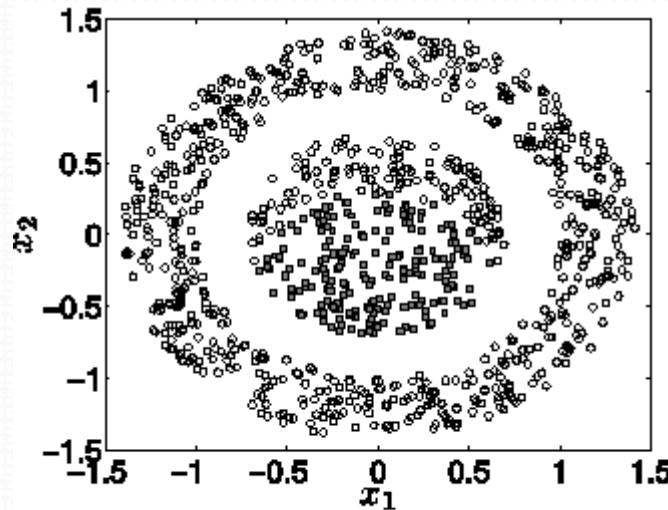
Kernelized k-means



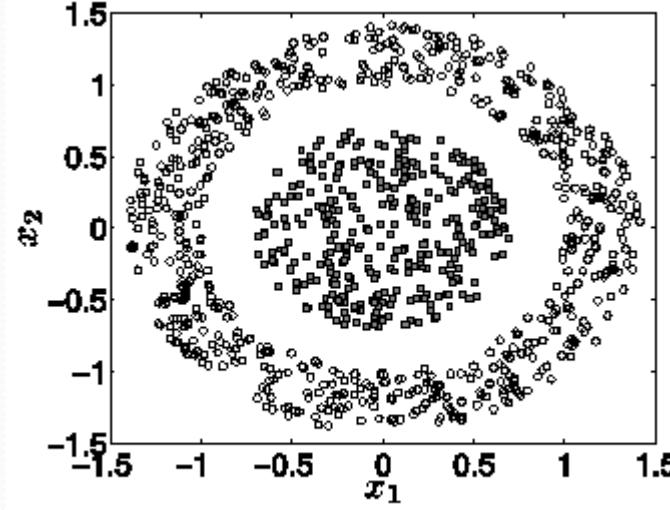
(a) Kernel K-means after one iteration



(b) After five iterations



(c) After 10 iterations



(d) At convergence (30 iterations)

Kernelized k-means

- Brings flexibility to simple k-means algorithm
- Need to set additional kernel parameters
- Sensitive to initialization
 - Often, it's good to run standard k-means algorithm before kernelized k-means for decent initialization

K-means: limitations

- Sensitive to initialization
- Computationally expensive
- Often needs repeated runs
- Makes hard assignments (i.e. each object can belong only to one cluster)
 - Soft assignment (each object has a probability of belonging to each cluster)

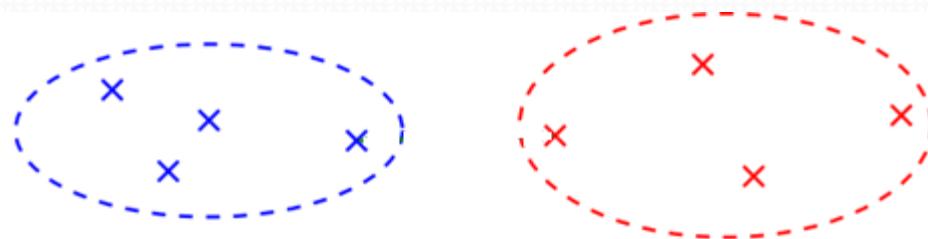
Hierarchical clustering

- Bottom-up (agglomerative) approach
 - Each object starts as a singleton cluster
 - Two most similar clusters are merged iteratively
 - Stop when all objects are in same cluster
- Top-down (divisive) approach
 - All objects belong to same cluster
 - “outsider” objects from least cohesive cluster are removed iteratively
 - Stop when each object is a singleton cluster

Clustering: similarity measure

- We know how to estimate similarity (or distance between pair of objects)
- How do we measure similarity between a pair of clusters (R and S)?

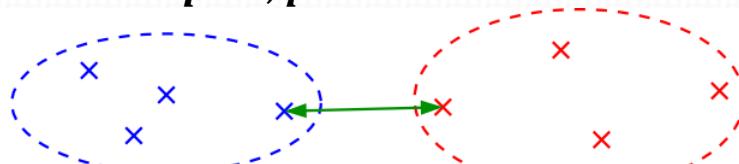
- Linkage
 - Min/single link
 - Max/complete link
 - Average link



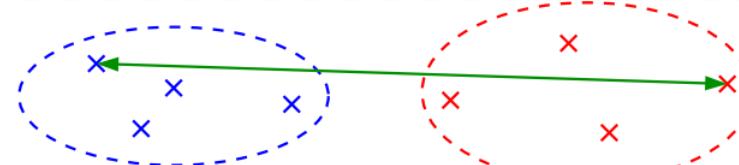
Clustering: similarity measure

- Min/single link:
 - Clusters can get very large
- Max/complete link:
 - Results in small, round clusters
- Average link:
 - Compromise between min and max links

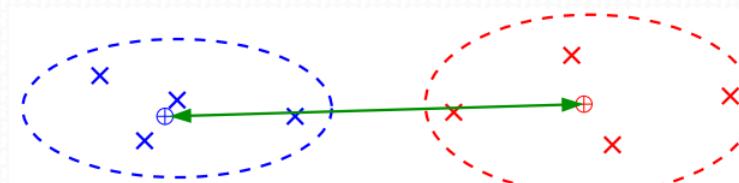
$$D(R, S) = \min_{p \in R, q \in S} D(p, q)$$



$$D(R, S) = \max_{p \in R, q \in S} D(p, q)$$



$$D(R, S) = \frac{1}{|R||S|} \sum_{p \in R, q \in S} D(p, q)$$

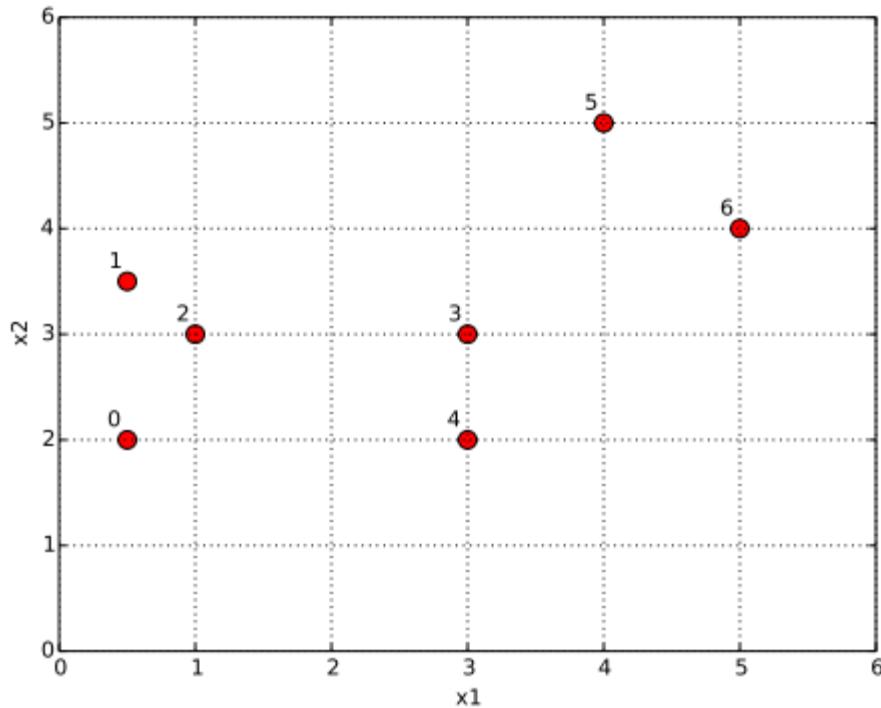


Hierarchical clustering

- Hierarchical agglomerative clustering (HAC) is a bottom-up clustering strategy
 - Each object is a singleton cluster
 - Successively merge closest pair of clusters together, till all clusters merge
1. Consider each object as a singleton cluster
 2. Compute distance between all pairs of clusters
 3. Remove the closest pair of clusters from the data, and replace them as a single cluster
 4. If not all objects belong to same single cluster, jump to step 2

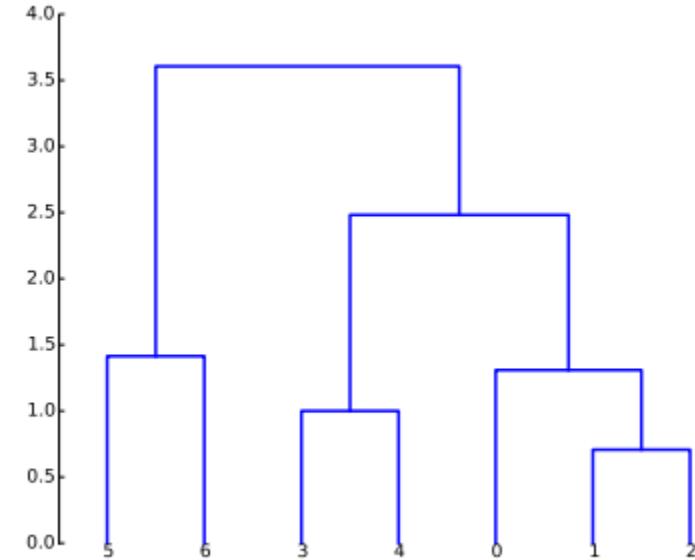
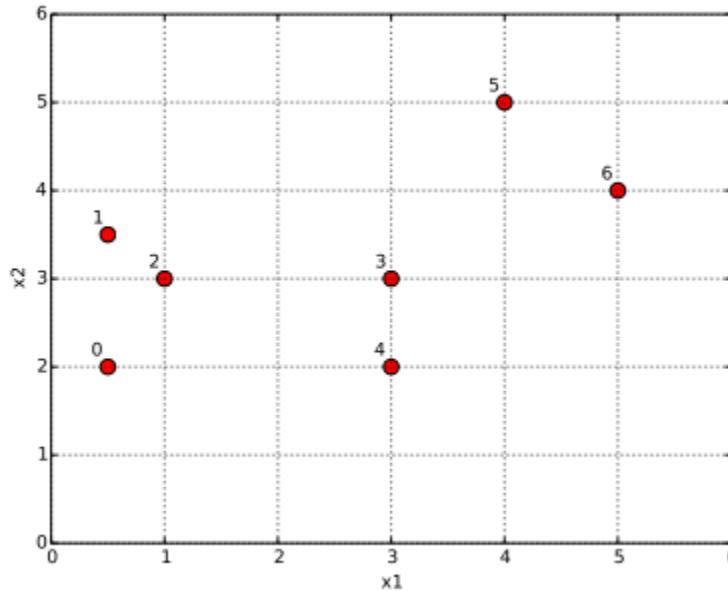
Hierarchical clustering

- Hierarchically cluster



Hierarchical clustering

- The results of hierarchical clustering can be visualized as a dendrogram where the height of each “junction” in the dendrogram represents the distance between the pair of clusters
- Hierarchical clustering generates all number of clusters
 - Cut horizontally across the dendrogram



Flat clustering vs hierarchical clustering

- Flat clustering (K-means) results in a single partitioning of objects
 - Hierarchical clustering can produce different partitioning results depending on the level-of-resolution we are looking at
-
- Flat clustering needs to know K in advance
 - Hierarchical clustering doesn't need to know K

Flat clustering vs hierarchical clustering

- Flat clustering is usually more efficient (run-time) – if we know K
 - $N \times K$ distance computations at each iteration
- Hierarchical clustering can be slow (has to make several merge/split decisions)
 - Choosing 2 closest objects from N objects requires
$$\frac{N!}{2!(N-2)!} \Rightarrow N * (N - 1)/2$$
 calculations just for first pairing
- No clear consensus on which of the two produces better clustering
 - Results depend upon nature of data

Summary

- Two common clustering methods
 - By associating an object with its nearest cluster mean
 - By successively merging smaller clusters to hierarchically form larger clusters
- Free choices
 - How many clusters?
 - What distance metric?
 - What linkage method?
- Optimal combination is often found by trial-and-error
- Kernelized k-means

Exercise (ungraded)

- Use K-means algorithm and Euclidean distance metric to cluster the following 2-dimensional objects in to 3 clusters
 - O1 (2,10), O2 (2,5), O3 (8,4), O4 (5,8), O5 (7,5), O6 (6,4), O7 (1,2), O8 (4,9)
- Suppose that initial cluster centres are O1, O4, and O7. Run the K-means algorithm steps for 1 iteration and show:
 - The clusters (i.e. objects belonging to each cluster)
 - The centres of new clusters
 - Draw a 10x10 grid with all the 8 objects and show the clusters and centres after the first iteration

Exercise (ungraded)

- Use min/single link to perform agglomerative clustering by showing the dendrogram
 - The data is described by the distance matrix

	A	B	C	D
A	0	1	4	5
B		0	2	6
C			0	3
D				0

- Use max/complete link to perform agglomerative clustering by showing the dendrogram
 - The data is described by the distance matrix
- Note: the height of each “junction” in the dendrogram represents the distance between the pair of clusters

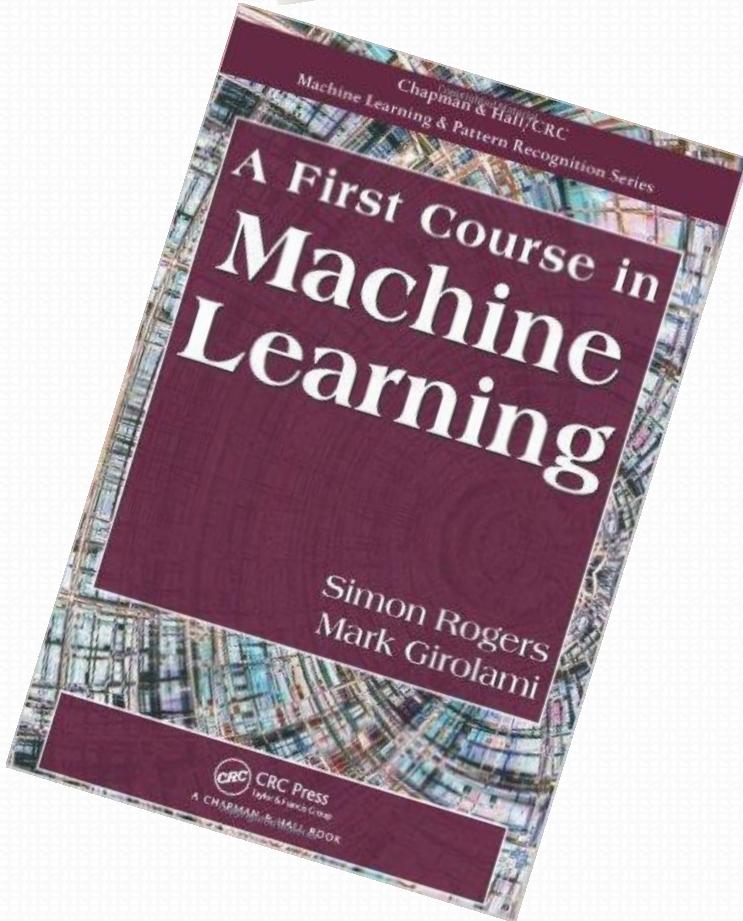
Exercise (ungraded)

- Use single and complete link agglomerative clustering to cluster the following 8 objects by showing the dendograms
 - O1 (2,10), O2 (2,5), O3 (8,4), O4 (5,8), O5 (7,5), O6 (6,4), O7 (1,2), O8 (4,9)

Exercise (ungraded)

- Try MATLAB code - kmeansexample.m (from FCML book website)
- Try MATLAB code - kmeansK.m (from FCML book website)
- Try MATLAB code - kernelkmeans.m (from FCML book website)
- Try MATLAB code – kmeans_cluster.m (from Canvas)
- Try MATLAB code – run_kmeans.m (from Canvas)

C, R, E, D, I, T, S,



Author's material
(Simon Rogers)



Iain Styles



Thank You

Machine Learning, Machine Learning (extended)

8 – Supervised Learning: Discriminative Classification

Kashif Rajpoot

k.m.rajpoot@cs.bham.ac.uk

**School of Computer Science
University of Birmingham**

Outline

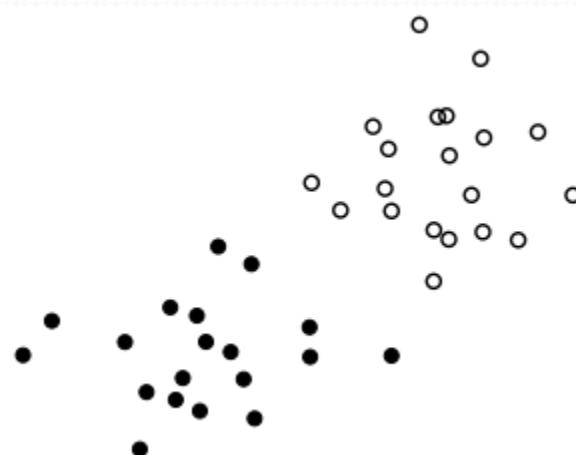
- Supervised learning
- Discriminative classification
 - Decision boundary
 - The margin
- Maximizing the margin
- Making predictions
- Support vectors
- Hard margin
- Soft margin
- Non-linear decision boundary
- Kernel trick

Supervised learning

- Regression
 - Minimised loss (e.g. least squares)
 - Maximum likelihood
- Classification
 - Generative (e.g. Bayesian)
 - Instance-based (e.g. k-NN)
 - Discriminative (e.g. SVM)

Classification

- A set of N objects with attributes (usually vector) \mathbf{x}_n
- Each object has an associated target label t_n
- Binary classification
 $t_n \in \{0,1\}$ or $t_n \in \{-1,1\}$
- Multi-class classification
 $t_n \in \{1,2,\dots,C\}$

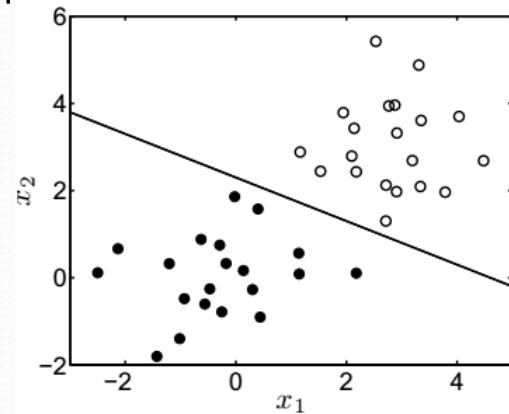
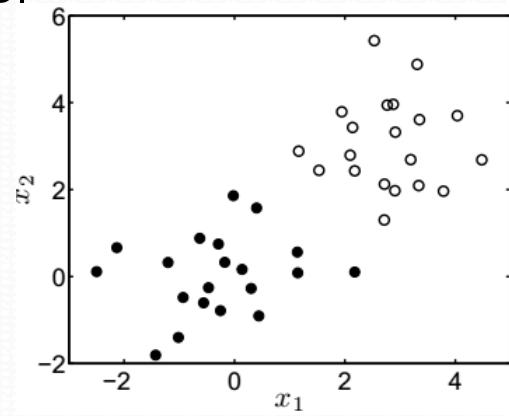


Generative vs discriminative classification

- Generative classifiers generate a model for each class, based on training samples available
 - Data in each class can be seen as generated by some model
 - For new test samples, they assign these samples to the class that suits best (e.g. by probability measure)
- In contrast, discriminative classifiers attempt to explicitly define the decision boundary that separates the classes
 - Intuitively, these methods are for binary class problems but can be extended to multi-class problems

Support vector machines

- Let's consider a 2-d example where a model needs to learn classification
- Let's consider model as a linear decision boundary (i.e. straight line) that separates the two classes
- SVM is a binary classifier that learns a linear decision boundary from attributes $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ and target labels t_1, t_2, \dots, t_N , with $t_n \in \{-1, 1\}$
- In n-d, SVM is a discriminant classifier that determines a *linear hyperplane*
- SVM is a very popular classifier in bioinformatics, medical imaging, digit classification, and various other areas

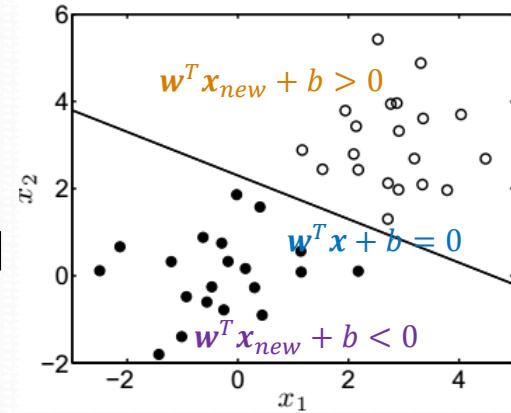
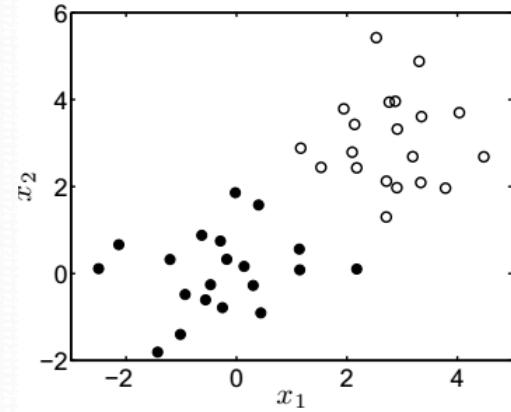


Line: refresher

- What's the equation of a straight line?
 - $y = mx + c \Rightarrow \mathbf{w}^T \mathbf{x} + b = 0?$
 - $\mathbf{w}?$ $b?$
 - $y = -\frac{1}{4}x + 3 \Rightarrow \mathbf{w}^T \mathbf{x} + b = 0$
 - $\mathbf{w}?$ $b?$
- For what points: $\mathbf{w}^T \mathbf{x} + b > 0?$
- For what points: $\mathbf{w}^T \mathbf{x} + b < 0?$
- For what points: $\mathbf{w}^T \mathbf{x} + b = 1?$
- For what points: $\mathbf{w}^T \mathbf{x} + b = -1?$
- Relation of \mathbf{w} to the straight line $\mathbf{w}^T \mathbf{x} + b = 0?$
 - For example: consider $y = 2x$

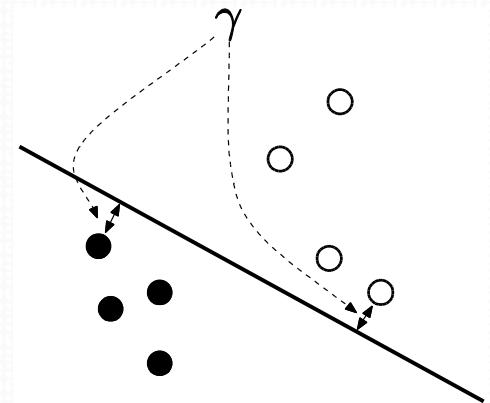
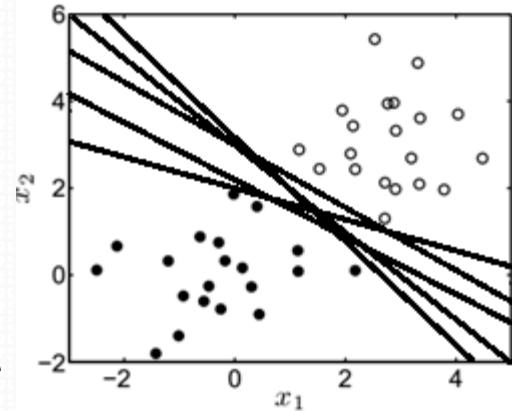
Decision boundary

- Linear decision boundary can be represented as a straight line
 - $\mathbf{w}^T \mathbf{x} + b = 0$
- To classify a new test sample \mathbf{x}_{new} :
 - $t_{new} = 1$: if $\mathbf{w}^T \mathbf{x}_{new} + b > 0$
 - $t_{new} = -1$: if $\mathbf{w}^T \mathbf{x}_{new} + b < 0$
- The decision function (prediction) becomes:
 - $t_{new} = sign(\mathbf{w}^T \mathbf{x}_{new} + b)$
- Decision boundary is determined by \mathbf{w} and b
 - How to choose \mathbf{w} and b ?



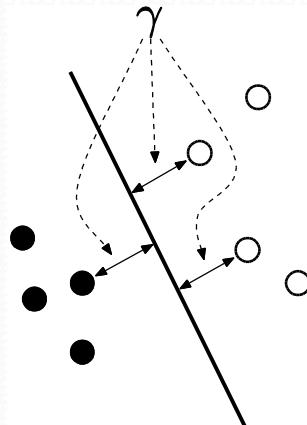
The margin

- Given linearly separable two class data, there are *infinite* number of straight lines that can separate it
 - Which should we choose?
- According to *learning theory*, a decision boundary that maximizes the margin of the boundary to the training set is the one that will minimize the generalization error
 - Margin: perpendicular distance (γ) between the boundary and closest training points of each class
- SVM finds the decision boundary that maximizes the margin
- How to choose w and b ?
 - Optimize the margin γ

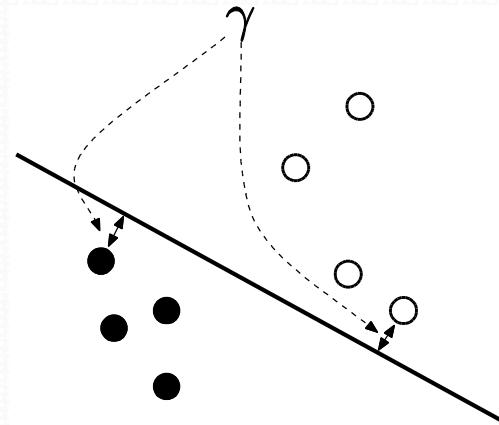


Maximizing the margin

- Maximize the *perpendicular distance* from the decision boundary to the closest points on each side
 - Maximum margin decision boundary better reflects the data characteristics than non-optimal boundary
 - Maximum margin decision boundary classifier generalizes well on unseen test data



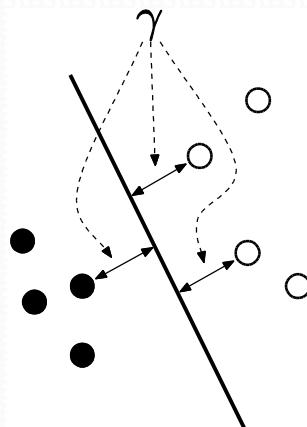
(a) The decision boundary that maximises the margin



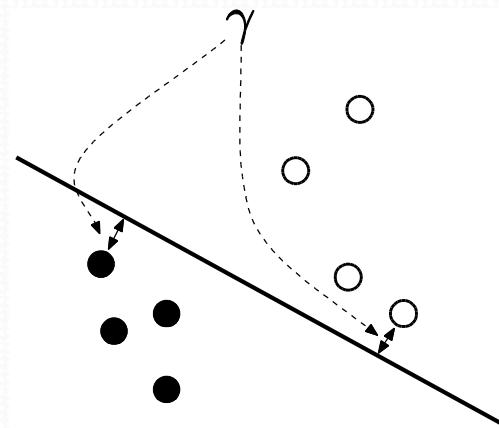
(b) A non-optimal decision boundary

Maximizing the margin

- From all possible linear decision boundaries, the one that maximizes the margin on the training set will minimize the generalization error
 - subject to have seen “enough” training samples and assuming that data isn’t “noisy”



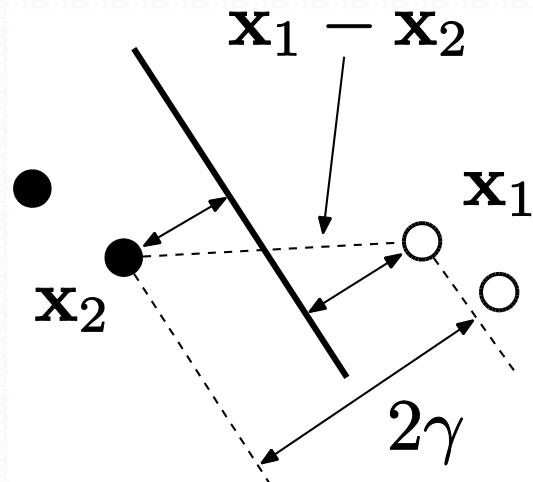
(a) The decision boundary that maximises the margin



(b) A non-optimal decision boundary

Maximizing the margin

- Let's consider two closest points to the boundary: \mathbf{x}_1 and \mathbf{x}_2
- Double margin (2γ) is equal to the component of vector joining \mathbf{x}_1 and \mathbf{x}_2 in the direction perpendicular to the boundary
 - $\mathbf{x}_1 - \mathbf{x}_2$ is the vector joining \mathbf{x}_1 and \mathbf{x}_2
 - $\mathbf{w}/\|\mathbf{w}\|$ is the direction perpendicular to the boundary
 - Thus $2\gamma = \frac{1}{\|\mathbf{w}\|} \mathbf{w}^T (\mathbf{x}_1 - \mathbf{x}_2)$



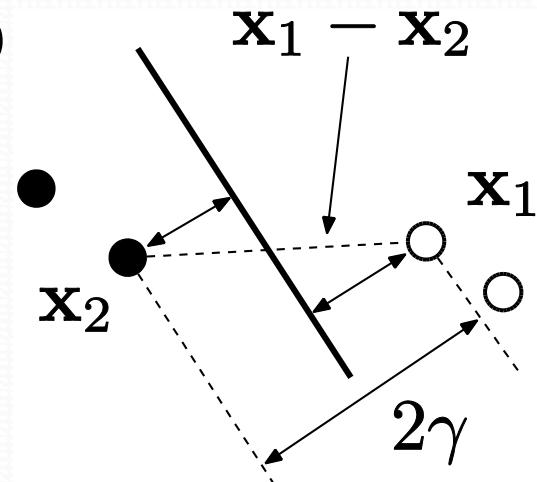
Maximizing the margin

- Double margin can be estimated as:

$$2\gamma = \frac{1}{\|\mathbf{w}\|} \mathbf{w}^T (\mathbf{x}_1 - \mathbf{x}_2)$$

- Decision function $t_{new} = \text{sign}(\mathbf{w}^T \mathbf{x}_{new} + b)$ is invariant to scaling its argument by a positive constant λ

- $\text{sign}(\mathbf{w}^T \mathbf{x}_{new} + b) = \text{sign}(\lambda \mathbf{w}^T \mathbf{x}_{new} + \lambda b)$

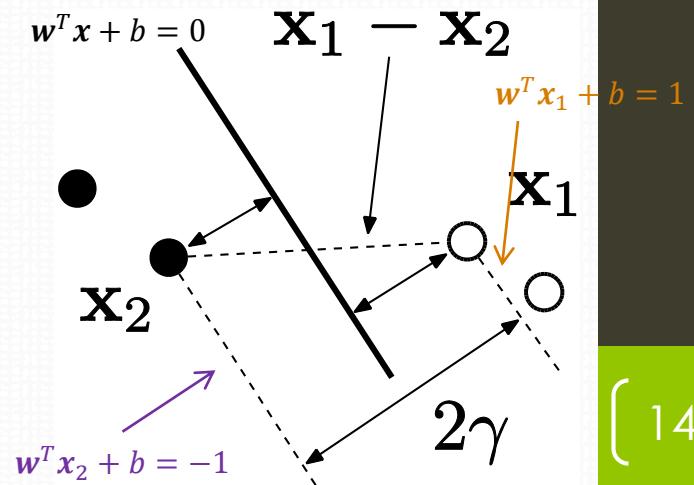


- Let's set the scale such that:

- $\mathbf{w}^T \mathbf{x}_1 + b = 1$
- $\mathbf{w}^T \mathbf{x}_2 + b = -1$

Maximizing the margin

- Considering:
 - $\mathbf{w}^T \mathbf{x}_1 + b = 1$ (line parallel to decision boundary)
 - $\mathbf{w}^T \mathbf{x}_2 + b = -1$ (line parallel to decision boundary)
- By subtracting the above two equations:
 - $(\mathbf{w}^T \mathbf{x}_1 + b) - (\mathbf{w}^T \mathbf{x}_2 + b) = 1 - (-1)$
 - $\mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2) = 2$
- Thus:
 - $2\gamma = \frac{1}{\|\mathbf{w}\|} \mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2) = \frac{1}{\|\mathbf{w}\|} 2$
 - $\gamma = \frac{1}{\|\mathbf{w}\|}$



Maximizing the margin

- SVM maximizes $2\gamma = \frac{2}{\|\mathbf{w}\|}$
 - Equivalent to minimize $\frac{\|\mathbf{w}\|}{2}$
 - Equivalent to minimize (due to mathematical simplicity)
$$\frac{1}{2} \|\mathbf{w}\|^2 = \frac{1}{2} \mathbf{w}^T \mathbf{w}$$
- There are constraints (to prevent training samples falling in margin band):
 - For \mathbf{x}_n with $t_n = 1$: $\mathbf{w}^T \mathbf{x}_n + b \geq 1$
 - For \mathbf{x}_n with $t_n = -1$: $\mathbf{w}^T \mathbf{x}_n + b \leq -1$
- This can be expressed as:
 - $t_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1$
- This is why using $t_n \in \{-1, 1\}$ is beneficial over using $t_n \in \{0, 1\}$

Maximizing the margin

- SVM optimization problem is:

$$\underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2} \mathbf{w}^T \mathbf{w}$$

subject to constraint $t_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1$

- By the use of Lagrange multipliers (α_n), the constraints can be expressed in the minimization function (beyond our module scope):

$$\underset{\mathbf{w}, \alpha}{\operatorname{argmin}} \mathcal{L} = \underset{\mathbf{w}, \alpha}{\operatorname{argmin}} \left[\frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{n=1}^N \alpha_n (t_n(\mathbf{w}^T \mathbf{x}_n + b) - 1) \right]$$

subject to $\alpha_n \geq 0$

Maximizing the margin

- To find the minimum of minimization function \mathcal{L} , take the 1st derivative with respect to \mathbf{w} and b and set to 0:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{n=1}^N \alpha_n t_n \mathbf{x}_n = 0$$

$$\mathbf{w} = \sum_{n=1}^N \alpha_n t_n \mathbf{x}_n$$

and

$$\frac{\partial \mathcal{L}}{\partial b} = - \sum_{n=1}^N \alpha_n t_n = 0$$

$$\sum_{n=1}^N \alpha_n t_n = 0$$

- α_n ?

Let's recall..

$f(\mathbf{w})$	$\frac{\partial f}{\partial \mathbf{w}}$
$\mathbf{w}^\top \mathbf{x}$	\mathbf{x}
$\mathbf{x}^\top \mathbf{w}$	\mathbf{x}
$\mathbf{w}^\top \mathbf{w}$	$2\mathbf{w}$
$\mathbf{w}^\top \mathbf{Cw}$	$2\mathbf{Cw}$

Maximizing the margin

- By substituting $\mathbf{w} = \sum_{n=1}^N \alpha_n t_n \mathbf{x}_n$ in the SVM minimization function:

$$\underset{\mathbf{w}, \alpha}{\operatorname{argmin}} \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{n=1}^N \alpha_n (t_n (\mathbf{w}^T \mathbf{x}_n + b) - 1)$$

we obtain:

$$\underset{\alpha}{\operatorname{argmin}} \frac{1}{2} \left(\sum_{m=1}^N \alpha_m t_m \mathbf{x}_m^T \right) \left(\sum_{n=1}^N \alpha_n t_n \mathbf{x}_n \right) - \sum_{n=1}^N \alpha_n \left(t_n \left(\sum_{m=1}^N \alpha_m t_m \mathbf{x}_m^T \mathbf{x}_n + b \right) - 1 \right)$$

$$\underset{\alpha}{\operatorname{argmin}} \frac{1}{2} \sum_{m=1}^N \sum_{n=1}^N \alpha_m \alpha_n t_m t_n \mathbf{x}_m^T \mathbf{x}_n - \sum_{m=1}^N \sum_{n=1}^N \alpha_m \alpha_n t_m t_n \mathbf{x}_m^T \mathbf{x}_n - \sum_{n=1}^N \alpha_n t_n b + \sum_{n=1}^N \alpha_n$$

$$\underset{\alpha}{\operatorname{argmin}} \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{m=1}^N \sum_{n=1}^N \alpha_m \alpha_n t_m t_n \mathbf{x}_m^T \mathbf{x}_n$$

$$\boxed{\sum_{n=1}^N \alpha_n t_n = 0}$$

Maximizing the margin

- SVM optimization function becomes:

$$\underset{\alpha}{\operatorname{argmin}} \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \alpha_m \alpha_n t_m t_n \mathbf{x}_m^T \mathbf{x}_n$$

subject to

$$\alpha_n \geq 0$$

and

$$\sum_{n=1}^N \alpha_n t_n = 0$$

- Where are \mathbf{w} and b in optimization?
- This is a standard quadratic programming optimization problem (beyond our module scope) which can be solved numerically in MATLAB (quadprog)
 - There is no analytical solution

Making predictions

- Let's recall that target label t_{new} for a new test sample \mathbf{x}_{new} can be predicted as:

$$t_{new} = \text{sign}(\mathbf{w}^T \mathbf{x}_{new} + b)$$

- Do we know \mathbf{w} and b ?
- Let's recall $\mathbf{w} = \sum_{n=1}^N \alpha_n t_n \mathbf{x}_n$, so we get:

$$t_{new} = \text{sign} \left(\sum_{n=1}^N \alpha_n t_n \mathbf{x}_n^T \mathbf{x}_{new} + b \right)$$

- Do we know b ?

Making predictions

- To find b , consider the closest point \mathbf{x}_n to a new test sample \mathbf{x}_{new} , for which we already know:

$$\mathbf{w}^T \mathbf{x}_n + b = \pm 1 = t_n \quad \text{or} \quad t_n (\mathbf{w}^T \mathbf{x}_n + b) = 1$$

- Let's recall $\mathbf{w} = \sum_{n=1}^N \alpha_n t_n \mathbf{x}_n$, so we get:

$$t_n \left(\sum_{m=1}^N \alpha_m t_m \mathbf{x}_m^T \mathbf{x}_n + b \right) = 1$$

$$\sum_{m=1}^N \alpha_m t_m \mathbf{x}_m^T \mathbf{x}_n + b = \frac{1}{t_n}$$

$$b = t_n - \sum_{m=1}^N \alpha_m t_m \mathbf{x}_m^T \mathbf{x}_n$$

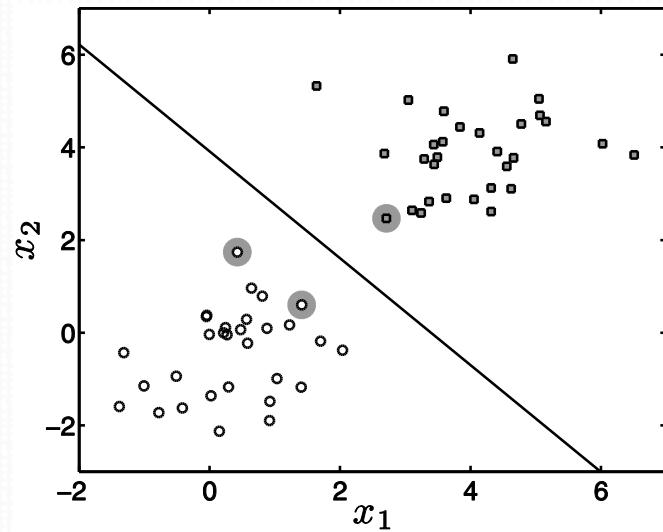
$$\boxed{\frac{1}{t_n} = t_n}$$

- Thus, we can predict t_{new} :

$$t_{new} = sign \left(\sum_{n=1}^N \alpha_n t_n \mathbf{x}_n^T \mathbf{x}_{new} + b \right)$$

Support vectors

- Support vector?
- Support vectors are the training samples closest to the maximum margin decision boundary
 - These vectors “support” the decision boundary
- Maximizing the margin determines the boundary
 - Margin is defined by support vectors
 - Can we discard non-support vectors?



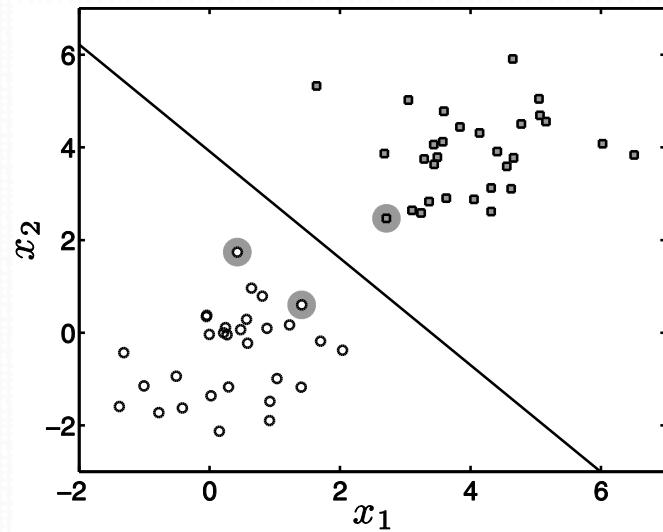
Support vectors

- Can we discard non-support vectors?

- At the optimum, non-support vectors will have zero α_n

$$t_{new} = \text{sign}(\sum_{n=1}^N \alpha_n t_n \mathbf{x}_n^T \mathbf{x}_{new} + b)$$

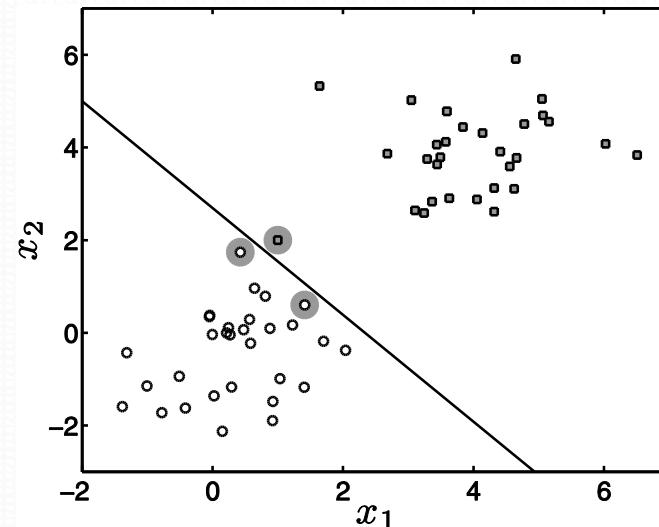
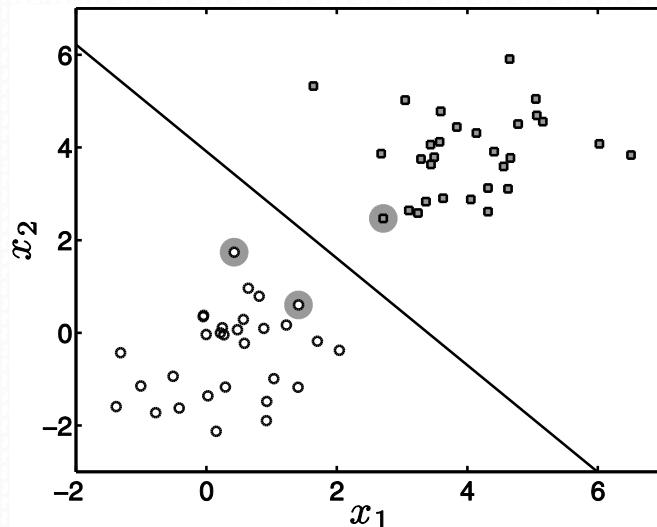
- We get a sparse solution
 - Decision boundary is a function of a small subset (i.e. support vectors)



- How does this compare to kNN classification?
 - kNN finds distance to all objects and finds k closest ones

Support vectors

- At times, a sparse solution can result in problems
- Why does this happen?
 - Hard margin: decision boundary is completely determined by training samples: $t_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1$
 - All training samples need to reside on correct side of decision boundary
- Soft margin: permit some points to lie within margin band or even on the wrong side of boundary



Soft margin

- Permit some training points to lie within margin band or even on the wrong side of boundary
 - Relax the constraint $t_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1$ to
$$t_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1 - \xi_n, \text{ subject to } \xi_n \geq 0$$
- The optimization problem becomes:

$$\underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{n=1}^N \xi_n$$

subject to $t_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1 - \xi_n$ and $\xi_n \geq 0$

- Parameter C controls to what extent the algorithm will permit the training samples to reside within margin band or on the wrong side of the boundary

Soft margin

- With use of Lagrange multipliers and similar math work as before, the optimization problem becomes:

$$\underset{\alpha}{\operatorname{argmax}} \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \alpha_m \alpha_n t_m t_n \mathbf{x}_m^T \mathbf{x}_n$$

subject to

$$0 \leq \alpha_n \leq C$$

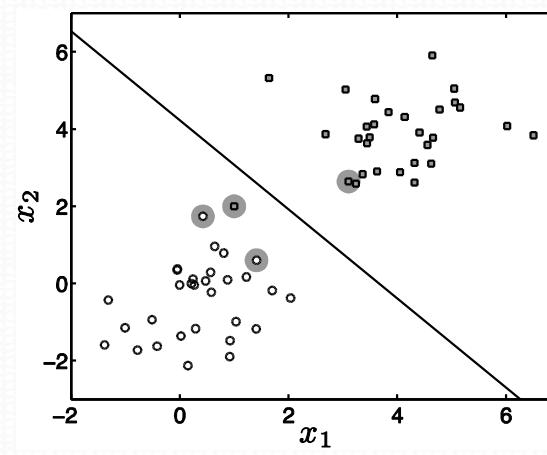
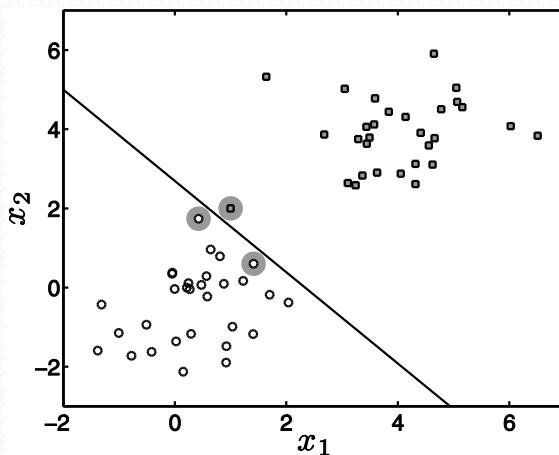
and

$$\sum_{n=1}^N \alpha_n t_n = 0$$

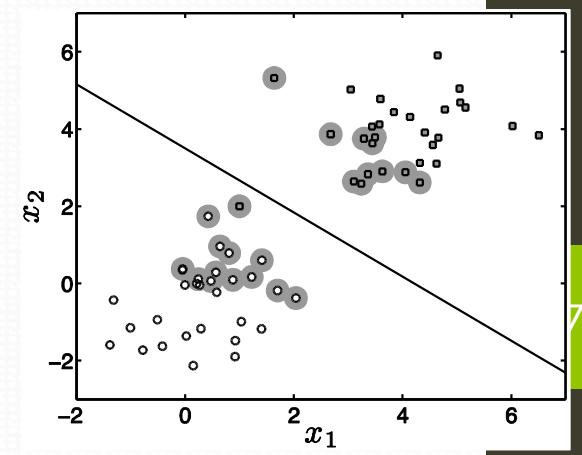
- Note that the only difference, compared to hard margin classifier, is an upper bound C on α_n

Soft margin

- For the stray support vector: $\alpha_n = 5.45$
- Setting C can bring a change in decision boundary
 - Some other α_n will have to become non-zero to bring change in the decision boundary
- With decreasing C , maximum potential influence of each training point is reduced
 - More training points become involved in the decision function
- How to choose C ?
 - Cross-validation



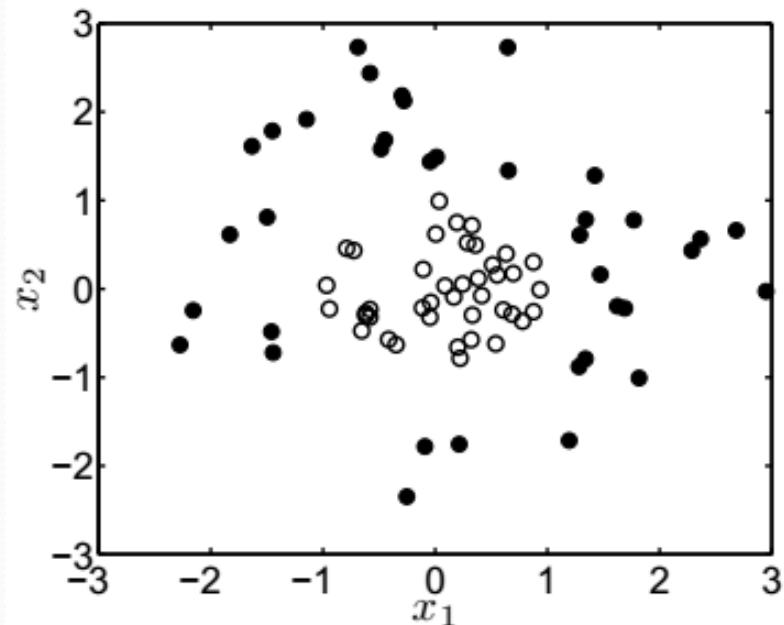
(a) $C = 1$



(b) $C = 0.01$

Non-linear decision boundary

- SVM can determine linear decision boundary
 - What if data is not linearly separable?
 - Can soft margin help?

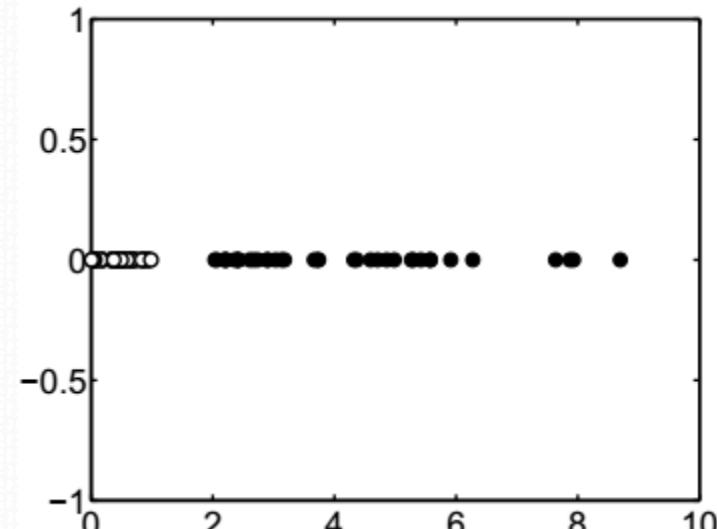
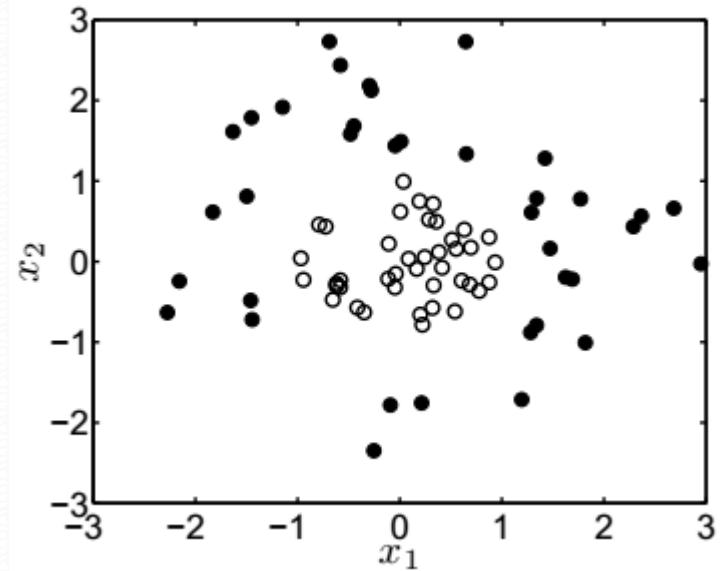


Non-linear decision boundary

- How about transforming data in to a new space where it can be separable?
 - $\mathbf{x} \rightarrow \phi(\mathbf{x})$

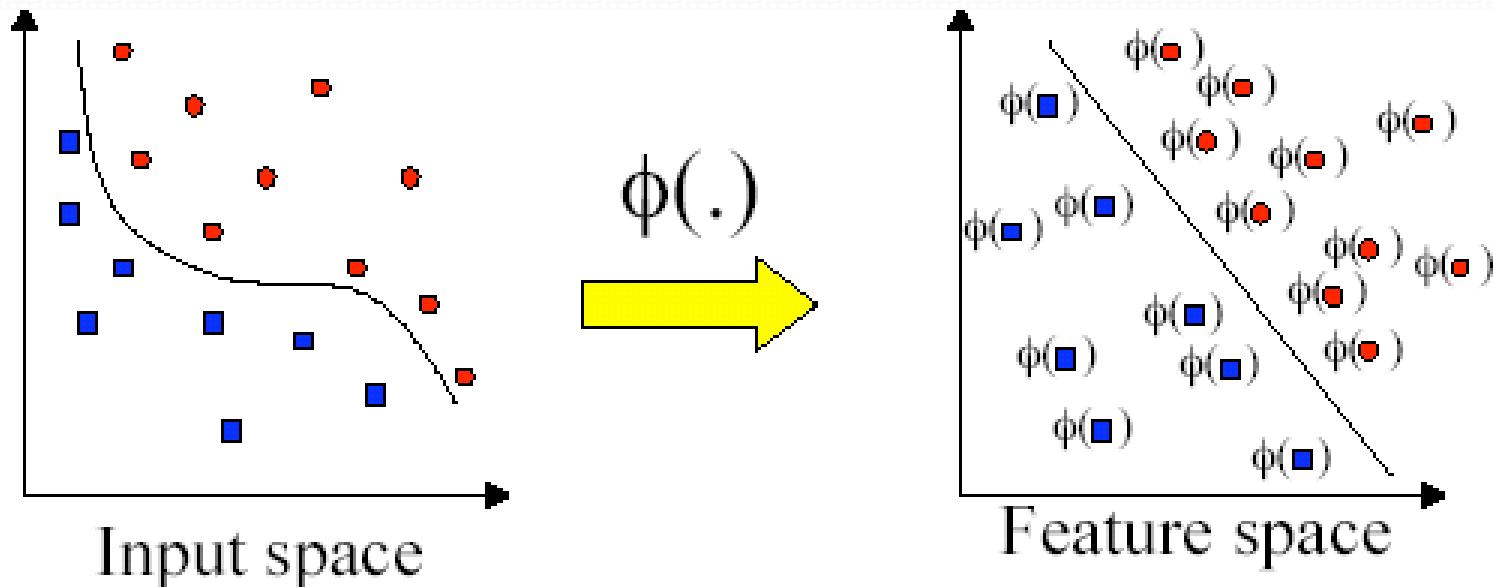
- For this example, consider:

$$\phi(\mathbf{x}_n) = x_{n1}^2 + x_{n2}^2$$



Non-linear decision boundary

- Transform (or project) the data in to a space where it's linearly separable



Non-linear decision boundary

- SVM optimization function:

$$\underset{\alpha}{argmax} \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \alpha_m \alpha_n t_m t_n \mathbf{x}_m^T \mathbf{x}_n$$

- Use $\phi(\mathbf{x}_n)$ instead of \mathbf{x}_n in optimization:

$$\underset{\alpha}{argmax} \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \alpha_m \alpha_n t_m t_n \phi(\mathbf{x}_m)^T \phi(\mathbf{x}_n)$$

and in prediction:

$$t_{new} = sign \left(\sum_{n=1}^N \alpha_n t_n \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_{new}) + b \right)$$

- Note that the data \mathbf{x}_m , \mathbf{x}_n , and \mathbf{x}_{new} always appear within dot product
- After the transformation, the dot product is calculated in the new space

Kernel trick

- Let's consider:
$$\phi(\mathbf{x}_m)^T \phi(\mathbf{x}_n) = (x_{m1}^2 + x_{m2}^2)(x_{n1}^2 + x_{n2}^2) = k(\mathbf{x}_m, \mathbf{x}_n)$$
- Dot product in the transformed space can be considered as a function of the original space
- *Kernel function*: a function that is equivalent to the dot product of vectors in the transformed $\phi(\dots)$ space
 - $\phi(\mathbf{x}_m)^T \phi(\mathbf{x}_n) = k(\mathbf{x}_m, \mathbf{x}_n)$
- *Kernel trick*: very neat trick which doesn't even require the explicit data transformation

Kernel function

- There are a number of off-the-shelf kernels that have been shown to work well
 - Think of kernel as a similarity metric
- Linear kernel
 - $k(\mathbf{x}_m, \mathbf{x}_n) = \mathbf{x}_m^T \mathbf{x}_n$
- Gaussian kernel
 - $k(\mathbf{x}_m, \mathbf{x}_n) = \exp\{-\beta(\mathbf{x}_m - \mathbf{x}_n)^T(\mathbf{x}_m - \mathbf{x}_n)\} = \exp\{-\beta\|\mathbf{x}_m - \mathbf{x}_n\|^2\}$
- Polynomial kernel
 - $k(\mathbf{x}_m, \mathbf{x}_n) = (\mathbf{x}_m^T \mathbf{x}_n + c)^\beta$
- $k(\mathbf{x}_m, \mathbf{x}_n)$ corresponds to $\phi(\mathbf{x}_m)^T \phi(\mathbf{x}_n)$ for some transformation $\phi(\mathbf{x}_n)$
 - Don't even need to know what is $\phi(\mathbf{x}_n)$

Non-linear decision boundary

- Use $k(\mathbf{x}_m, \mathbf{x}_n)$ instead of $\phi(\mathbf{x}_m)^T \phi(\mathbf{x}_n)$ in optimization function:

$$\underset{\alpha}{argmax} \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \alpha_m \alpha_n t_m t_n k(\mathbf{x}_m, \mathbf{x}_n)$$

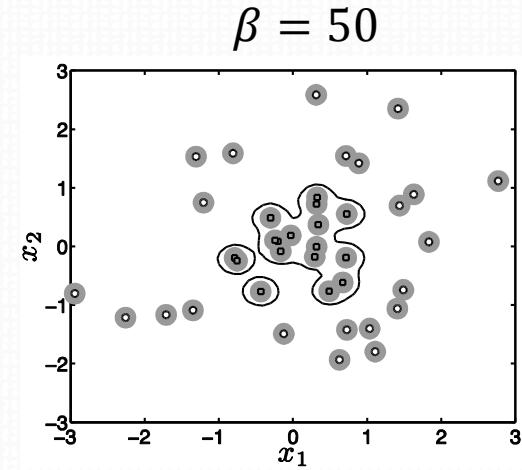
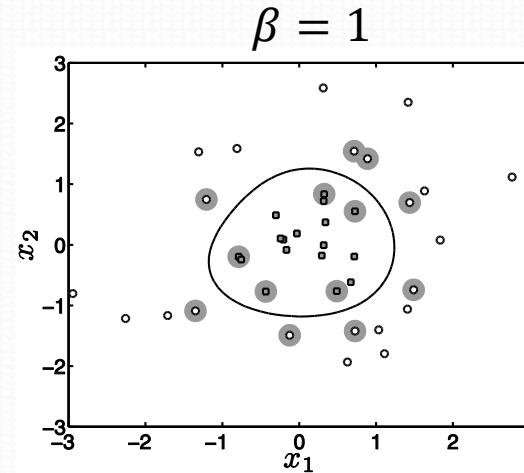
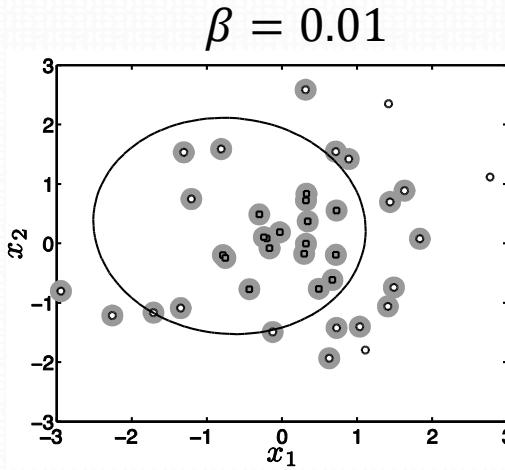
and in prediction function:

$$t_{new} = sign \left(\sum_{n=1}^N \alpha_n t_n k(\mathbf{x}_n, \mathbf{x}_{new}) + b \right)$$

- SVM is still finding linear boundaries...
 - ...but in some other space

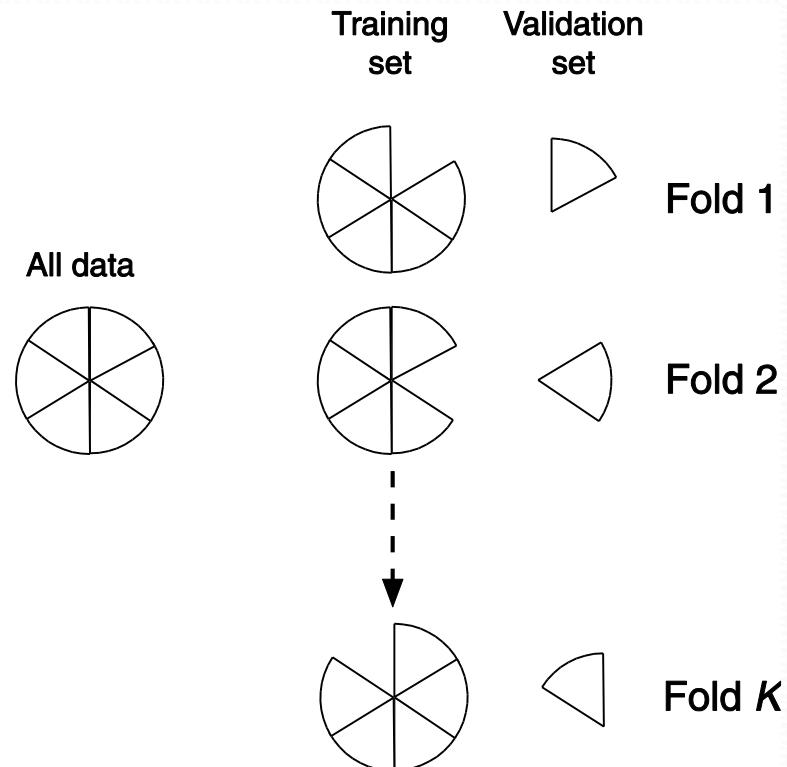
Non-linear decision boundary

- Non-linear data classification with SVM using Gaussian kernel
 - $C = 10$
- β controls the model complexity
 - Very small $\beta \Rightarrow$ too simple (under-fitting)
 - Very large $\beta \Rightarrow$ too complex (over-fitting)
- Non-sparse model for too small or too large β



Parameter selection

- How to choose C and β ?
 - Parameter choice is data dependent
- Cross-validation
 - Search over C and β
- Extra computational burden



Kernelizing other algorithms

- Other algorithms can be kernelized
 - As long as they have data appearing only in inner products in model learning and prediction
- Simple algorithms can learn complex decision boundaries
 - We have seen its usefulness in k-means clustering
- kNN requires distance between each training sample \mathbf{x}_n and test sample \mathbf{x}_{new}
 - The distance can be written as: $(\mathbf{x}_n - \mathbf{x}_{new})^T(\mathbf{x}_n - \mathbf{x}_{new})$
 - Can we kernelize kNN classifier?

SVM multi-class classification

- How to use a binary classifier (e.g. SVM) for multi-class classification?
- Consider that we have C number of classes
 - C not to be confused with C parameter for soft margin (they're different!!)
- There are two common strategies
 - One-vs-all
 - One-vs-one

SVM multi-class classification

- One-vs-all
 - This is the most frequently used option
 - Train C distinct binary classifiers, each classifier learning one-vs-rest (one: +1, rest: -1)
 - For classification:

$$t_{new} = \arg \max_{c \in 1 \dots C} f_c(x_{new})$$

where f_c is the classifier function that predicts confidence score for label c

- This approach creates class imbalance problem
 - i.e. one class has many more samples than the other one

SVM multi-class classification

- One-vs-one
 - Train $C(C - 1)/2$ distinct binary classifiers, each classifier learning one-vs-one (+1 vs -1)
 - For classification, all classifiers are used and t_{new} is assigned according to maximum voting
 - Addresses class imbalance problem in data
 - Some test samples may receive same number of votes for multiple classes

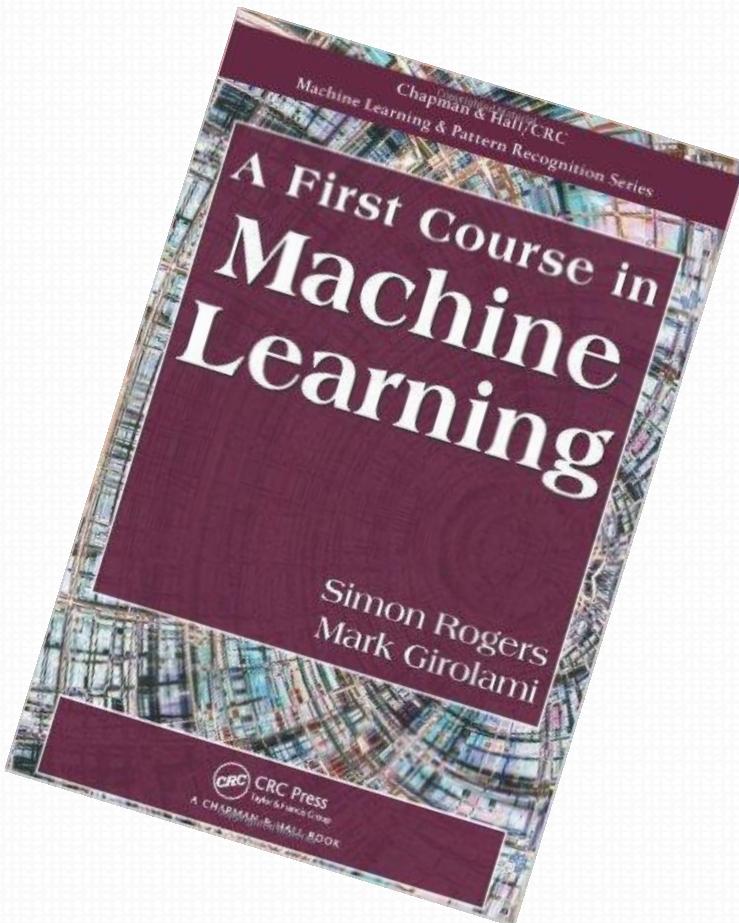
Summary

- Discriminative classification
- SVM: non-probabilistic linear binary classifier
 - Margin maximization
- Support vectors
- Hard margin vs soft margin
- Non-linear boundary learning with kernel trick
- Multi-class classification with a binary classifier

Exercise (ungraded)

- Try MATLAB code - svmhard.m (from FCML book website)
 - Requires quadprog from Optimization Toolbox
- Try MATLAB code - svmssoft.m (from FCML book website)
 - Requires quadprog from Optimization Toolbox
- Try MATLAB code - svmgauss.m (from FCML book website)
 - Requires quadprog from Optimization Toolbox

C, R, E, D, I, T, S,



Author's material
(Simon Rogers)



Ata Kaban



Thank You

Machine Learning, Machine Learning (extended)

9 – Unsupervised Learning: Dimensionality Reduction

Kashif Rajpoot

k.m.rajpoot@cs.bham.ac.uk

**School of Computer Science
University of Birmingham**

Outline

- High dimensionality
- Curse of dimensionality
- Dimensionality reduction
- Projection
- Preserving ‘interesting’ characteristics in data
- Principal component analysis (PCA)

High dimensionality

- Dimensionality
 - Number of attributes (i.e. features) in the data
- Technological advances lead to increasingly high dimensional data sets
 - Tens to hundreds to thousands...
 - Mass spectrometry, brain functional imaging, genomics, hyperspectral imaging, financial analysis
- High dimensional data => expected to give “more” information (features) about an object?
 - Not always, it could actually cause “curse of dimensionality”

Curse of dimensionality

- Computational burden
 - For example: consider clustering 3000 samples of 2000-dimensional data with k-means algorithm in 10 classes?
- Visualization
 - Typically, we can visualize data only up to 3d/4d
 - What about higher-dimensional data?
- Parameter estimation
 - Higher dimensions need estimation of higher number of parameters (e.g. regression, classification)

Dimensionality reduction

- Dimensionality reduction aims to avoid the ‘curse of dimensionality’ by reducing the attributes/dimensions/features

1. Feature selection

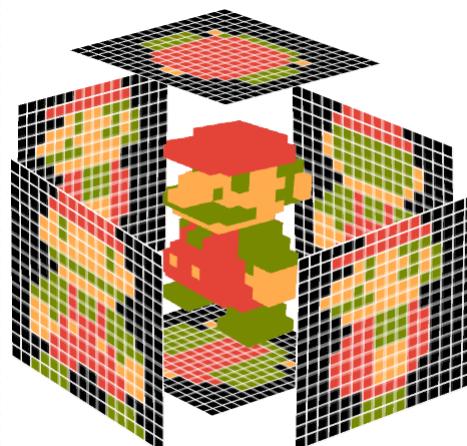
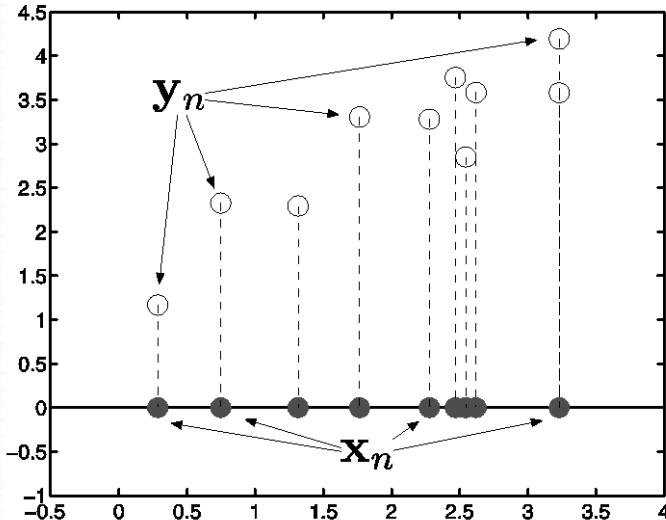
- Sequential feature selection (one of the simplest methods for feature selection)
 - Gradually add (remove) a feature to include (exclude)
- Determine feature scoring/importance by cross-validation

2. Subspace projection

- Make new features by combining (i.e. linearly or non-linearly) the old features
- Represent the data in fewer number of dimensions but ‘preserving’ the ‘interesting’ characteristics of data

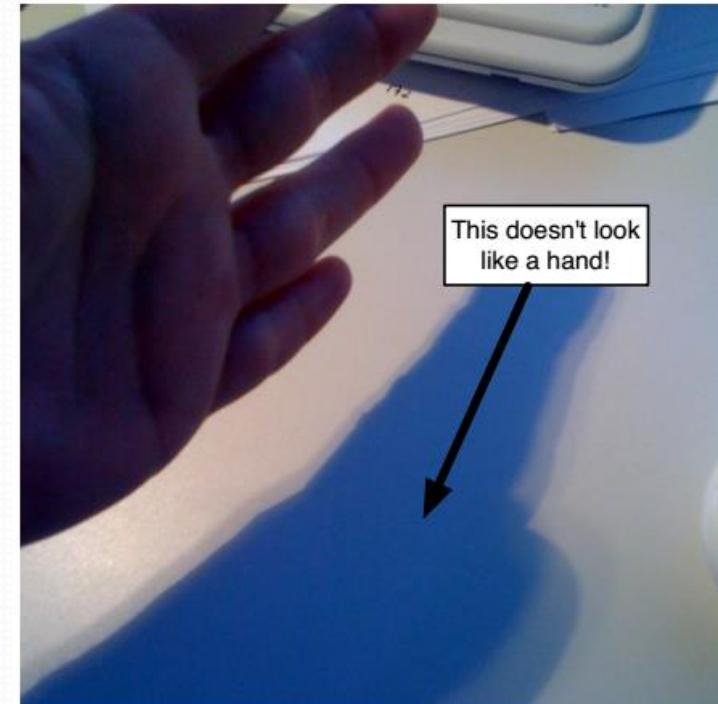
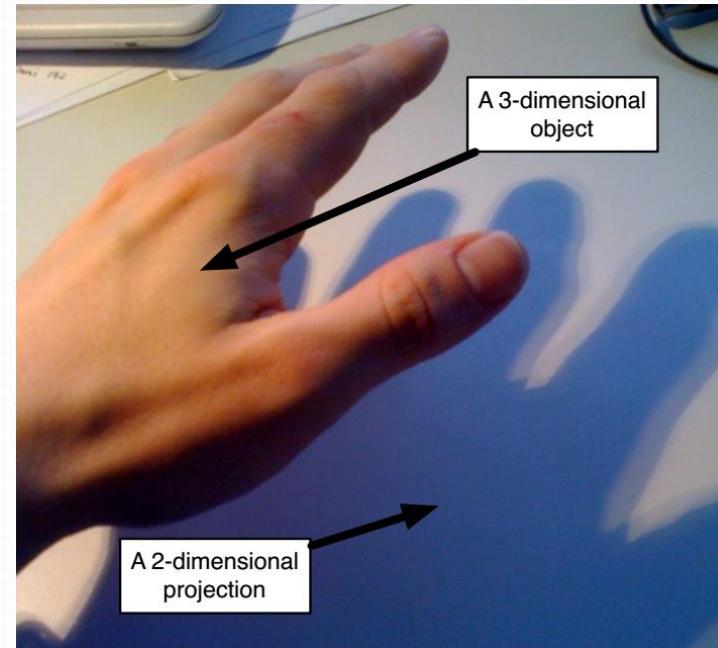
Projection

- Represent the data in fewer number of features but ‘preserving’ the ‘interesting’ characteristics of data?
- 3D world to 2D image projection



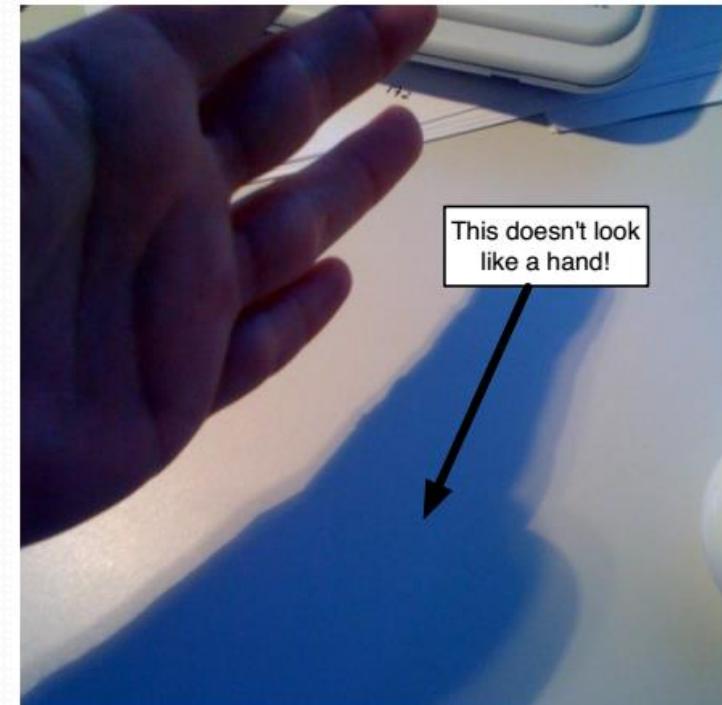
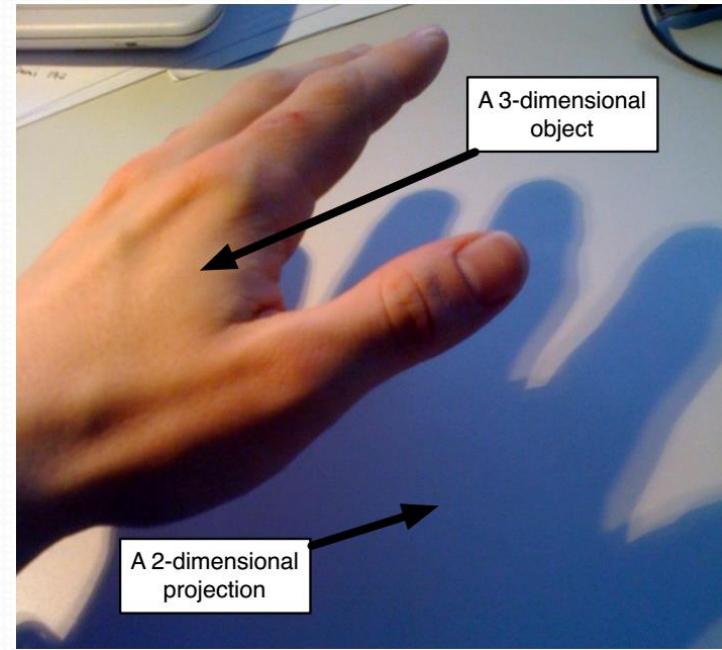
Projection

- Project M -dimensional data \mathbf{Y} containing N samples into a lower D -dimensional representation \mathbf{X} ($D \ll M$)
- $\mathbf{X} = \mathbf{Y}\mathbf{W}$
 - \mathbf{Y} is $N \times M$
 - \mathbf{W} is $M \times D$
 - \mathbf{X} is $N \times D$ (i.e. D -dimensional)
- What is \mathbf{W} ?
 - Defines the projection
 - Changing \mathbf{W} is like changing where the light is coming from or rotating the hand
- \mathbf{Y} is hand, \mathbf{X} is shadow



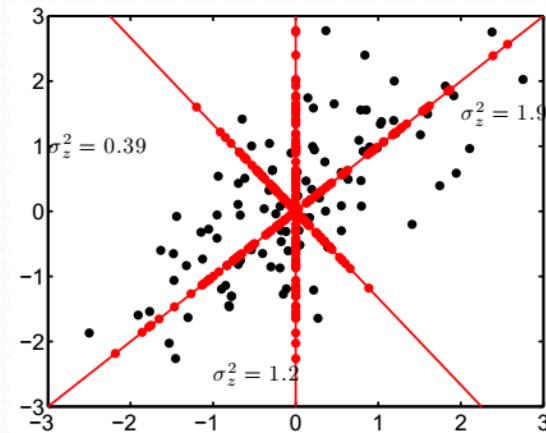
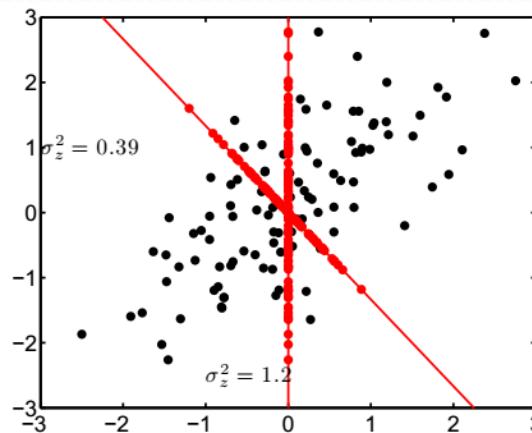
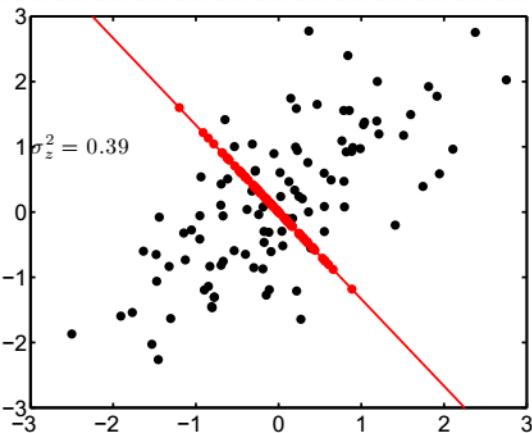
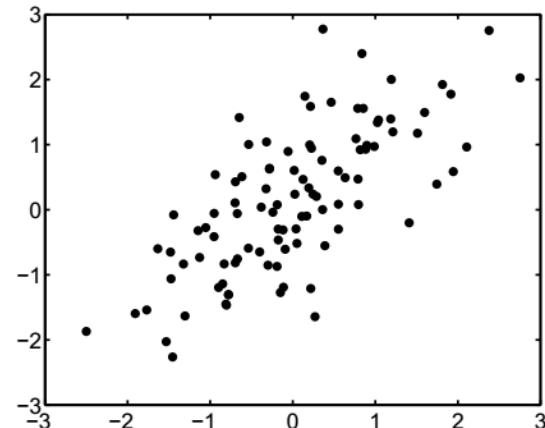
Projection

- Different \mathbf{W} will result in different projections
- How to choose \mathbf{W} ?
 - Not all projections will represent our data ‘well’
- We should choose a \mathbf{W} that preserves the ‘interesting’ data characteristics
 - such that $D \ll M$
 - M is the actual data’s dimensionality
 - D is the projected data’s dimensionality



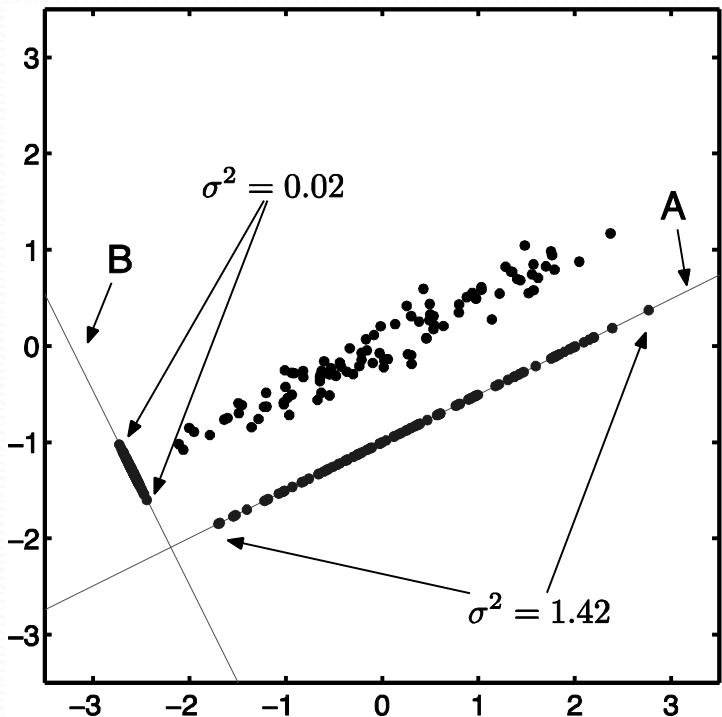
Preserve ‘interesting’ data characteristics

- Project 2-d data into 1-d?
- Pick some arbitrary w
- Project the data onto it
- The position on the line is our 1-d representation
- Compute the variance on the line

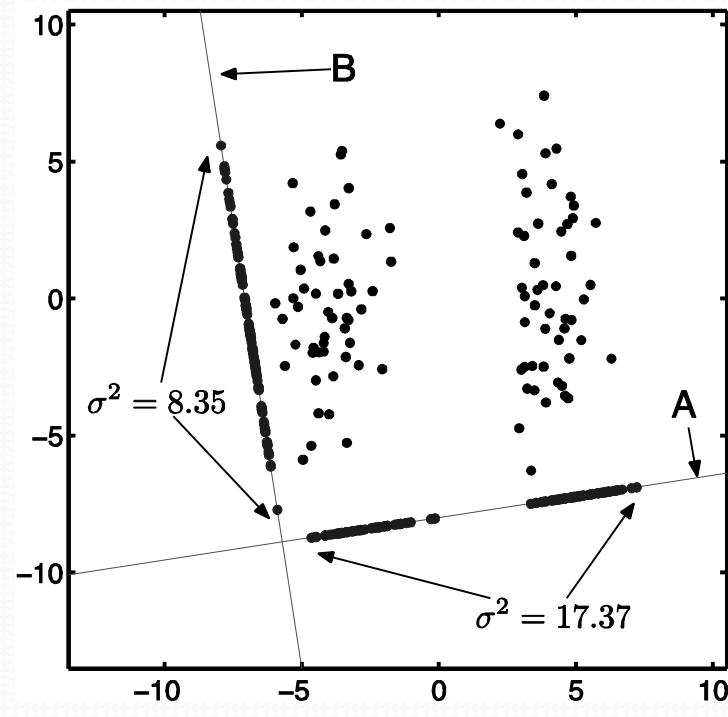


Preserve ‘interesting’ data characteristics

- Represent the data in fewer number of features but ‘preserving’ the ‘interesting’ characteristics of data?
 - High variance = highly “interesting” characteristics



(a) Data from a single, elongated Gaussian



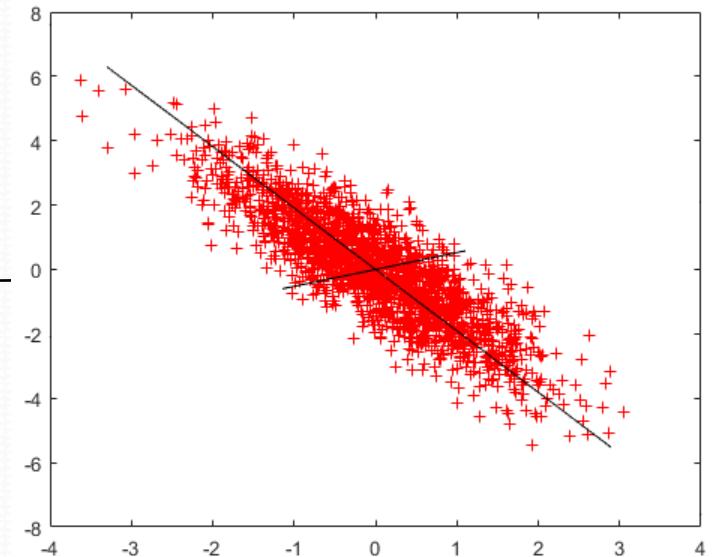
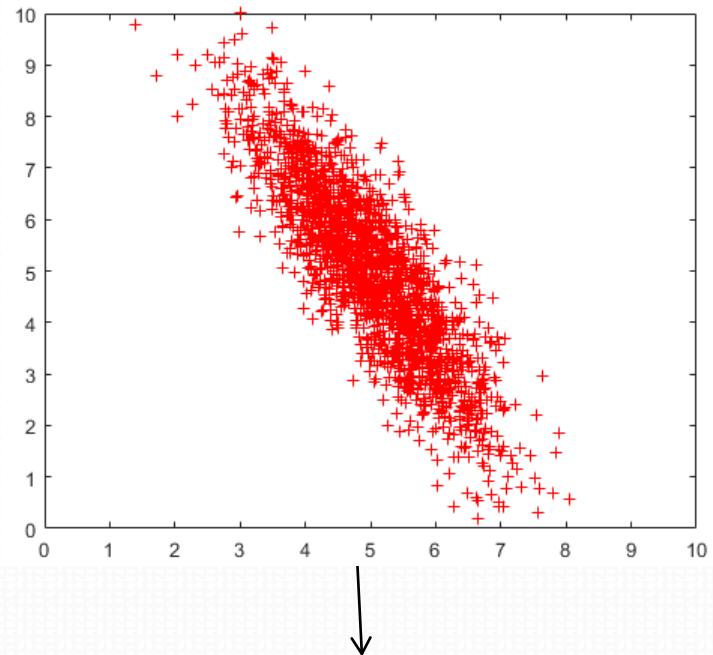
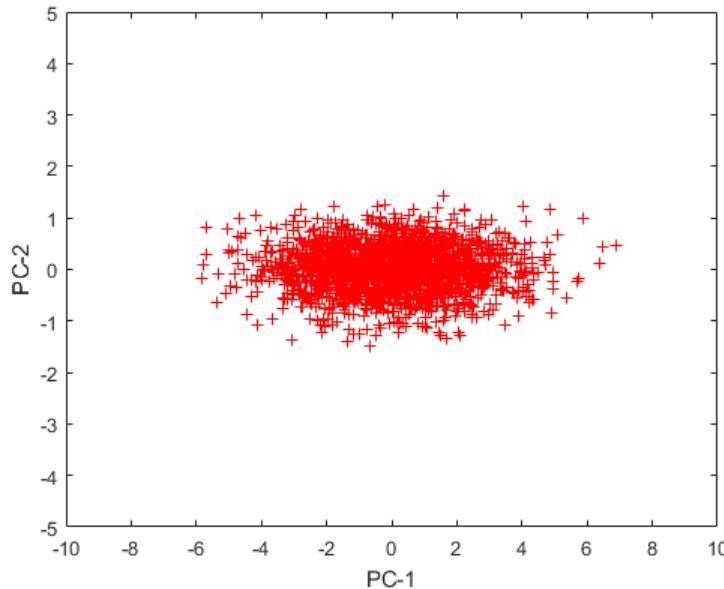
(b) Data from two Gaussians

Principal component analysis (PCA)

- PCA chooses \mathbf{W} such that it transforms M -dimensional data \mathbf{Y} into D -dimensional representation \mathbf{X} with $D \ll M$
 - It preserves dimensions with high variance ('interesting' characteristics)
 - It removes dimensions which are redundant (i.e. not much 'interesting' – having low variance)
- Find the columns of \mathbf{W} one at a time
 - \mathbf{w}_d as the d^{th} column of \mathbf{W}
 - Each $M \times 1$ column defines a new dimension

PCA

- PCA determines the dominant modes of variation from within the data and then projects data onto this ‘natural’ coordinate system
- Matches the coordinate system to the shape of the data



PCA

- Consider \mathbf{w}_d as a new dimension, then the data projection in this dimension is computed as:

$$\mathbf{x}_d = \mathbf{Y}\mathbf{w}_d$$

- PCA chooses \mathbf{w}_d that maximizes the variance of \mathbf{x}_d

- $\sigma_d^2 = \frac{1}{N} \sum_{n=1}^N (x_d^{(n)} - \mu_d)^2$ where $\mu_d = \frac{1}{N} \sum_{n=1}^N x_d^{(n)}$

and $x_d^{(n)}$ denotes the n^{th} sample value for d^{th} attribute

- Each new column of \mathbf{W} is found such that it maximizes variance and is orthogonal (perpendicular) to the previous columns

PCA: process

- Search for $\mathbf{W} = [\mathbf{w}_1 \ \mathbf{w}_2 \ \dots \ \mathbf{w}_D]$?
- Fortunately, analytical solution exists
- \mathbf{W} are the eigenvectors of the covariance matrix Σ of \mathbf{Y}
 - The covariance matrix Σ describes the way different attributes (i.e. features or variables) co-vary

PCA: process

- Covariance matrix Σ describes the way attributes co-vary
 - $\Sigma_{i,j}$ denotes the covariance of i^{th} and j^{th} attributes of \mathbf{Y}

$$\Sigma_{i,j} = \frac{1}{N} \sum_{n=1}^N (y_i^{(n)} - \mu_i)(y_j^{(n)} - \mu_j)$$

where μ_i and μ_j denote the mean of i^{th} and j^{th} attributes, respectively, while $y_i^{(n)}$ and $y_j^{(n)}$ denote the n^{th} sample value for i^{th} and j^{th} attributes, respectively.

- The data \mathbf{Y} is mean subtracted (along each dimension) to translate the data to the centre of coordinate system
 - Each row is an object, each column is a dimension
 - $\tilde{\mathbf{Y}} = \mathbf{Y} - \bar{\mathbf{Y}}$, where each column of $\bar{\mathbf{Y}}$ is mean value μ across that particular attribute
- The covariance matrix can then be computed as:

$$\Sigma = \frac{1}{N} (\mathbf{Y} - \bar{\mathbf{Y}})^T (\mathbf{Y} - \bar{\mathbf{Y}}) = \frac{1}{N} \tilde{\mathbf{Y}}^T \tilde{\mathbf{Y}}$$

PCA: process

- PCA finds new coordinate vectors

$$\mathbf{W} = [\mathbf{w}_1 \ \mathbf{w}_2 \ \dots \ \mathbf{w}_D]$$

that align with data shape

- $\mathbf{w}_i^T \mathbf{w}_j = 0, \forall i \neq j$ new dimensions are orthogonal
- PCA looks to find the direction \mathbf{w} that maximizes variance in that direction

$$\mathcal{F} = \underset{\mathbf{w}}{\operatorname{argmax}} \ var(\tilde{\mathbf{Y}}\mathbf{w})$$

- $var(\tilde{\mathbf{Y}}\mathbf{w}) = (\tilde{\mathbf{Y}}\mathbf{w})^T \tilde{\mathbf{Y}}\mathbf{w} = \mathbf{w}^T \tilde{\mathbf{Y}}^T \tilde{\mathbf{Y}}\mathbf{w} = \mathbf{w}^T \Sigma \mathbf{w}$

- So

$$\mathcal{F} = \underset{\mathbf{w}}{\operatorname{argmax}} \mathbf{w}^T \Sigma \mathbf{w}$$

subject to $\mathbf{w}^T \mathbf{w} = \mathbf{1}$

(we could keep increasing \mathbf{w} to maximise \mathcal{F} , hence the need to constrain $\mathbf{w}^T \mathbf{w} = \mathbf{1}$)

PCA: process

$$\mathcal{F} = \underset{\mathbf{w}}{\operatorname{argmax}} \mathbf{w}^T \Sigma \mathbf{w}$$

subject to $\mathbf{w}^T \mathbf{w} = 1$

- Using the Lagrange multiplier “trick” (beyond our module scope):

$$\mathcal{F} = \underset{\mathbf{w}}{\operatorname{argmax}} (\mathbf{w}^T \Sigma \mathbf{w} - \lambda(\mathbf{w}^T \mathbf{w} - 1))$$

- To maximize, differentiate with respect to \mathbf{w} and set to 0:

$$\frac{\partial \mathcal{F}}{\partial \mathbf{w}} = 2\Sigma \mathbf{w} - 2\lambda \mathbf{w} = 0$$

$$\Sigma \mathbf{w} = \lambda \mathbf{w}$$

- This can be solved with eigenvalue/eigenvector method (beyond our module scope)

Aside: Eigenvector

- We obtained solution for \mathbf{w} :

$$\Sigma \mathbf{w} = \lambda \mathbf{w}$$

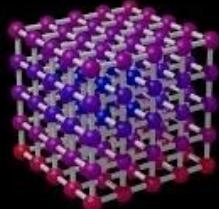
- Note that the multiplication of matrix Σ with vector \mathbf{w} changes it only by a scalar factor
- Such a vector \mathbf{w} is called eigenvector
 - λ is the eigenvalue, the scale by which eigenvector \mathbf{w} changes
- The eigenvector is a ‘special’ vector which has nice properties to become a transformation vector

Aside: Eigenvector

What is eigenvector?

<https://www.youtube.com/watch?v=ue3yoeZvt8E>

$$\mathbf{A}\vec{v} = \lambda\vec{v}$$



Introduction to eigenvalues
and eigenvectors

<https://www.youtube.com/watch?v=PhfbEr2btGQ>

EIGENVALUES AND
EIGENVECTORS INTRO

 KHAN ACADEMY

PCA: process

- The eigenvector and eigenvalue solution of covariance matrix Σ will provide M eigenvectors $[\mathbf{w}_1 \mathbf{w}_2 \dots \mathbf{w}_M]$ and M eigenvalues $[\lambda_1 \lambda_2 \dots \lambda_M]$:

$$\Sigma \mathbf{w}_i = \lambda_i \mathbf{w}_i$$

- Which \mathbf{w} has highest variance?
- Multiplying both sides of above equation by \mathbf{w}_i^T :

$$\mathbf{w}_i^T \Sigma \mathbf{w}_i = \lambda_i \mathbf{w}_i^T \mathbf{w}_i = \lambda_i$$

i.e. λ_i denotes variance along \mathbf{w}_i dimension since $\mathbf{w}_i^T \Sigma \mathbf{w}_i = \text{var}(\tilde{\mathbf{Y}} \mathbf{w}_i)$

- The principal components of data \mathbf{Y} are the eigenvectors $[\mathbf{w}_1 \mathbf{w}_2 \dots \mathbf{w}_M]$ of covariance matrix Σ , ordered by eigenvalues $[\lambda_1 \lambda_2 \dots \lambda_M]$

PCA: process

- Having found the principal components (PCs)

$$\mathbf{W} = [\mathbf{w}_1 \ \mathbf{w}_2 \ \dots \ \mathbf{w}_M]$$

data \mathbf{Y} can now be transformed to these new dimensions:

$$\mathbf{X} = \tilde{\mathbf{Y}}\mathbf{W}$$

where $\tilde{\mathbf{Y}}$ is the mean-subtracted data

- \mathbf{W} is the projection (aka *loadings*)
- \mathbf{X} is the projected data (aka *scores*)

PCA: algorithmic workflow

1. Form the $N \times M$ zero-mean matrix, by subtracting mean
 - $\tilde{\mathbf{Y}} = \mathbf{Y} - \bar{\mathbf{Y}}$, where each column of $\bar{\mathbf{Y}}$ is mean value across that particular attribute

2. Calculate the $M \times M$ covariance matrix Σ

$$\Sigma = \frac{1}{N} \tilde{\mathbf{Y}}^T \tilde{\mathbf{Y}}$$

3. Calculate the M eigenvalues ($\lambda_1, \lambda_2, \dots, \lambda_M$) and eigenvectors ($\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_M$) of Σ
4. Sort the eigenvalues and eigenvectors from largest to smallest by eigenvalue
5. Choose D eigenvectors corresponding to highest eigenvalues
6. Compute the scores \mathbf{X} (i.e. projections) by projecting data $\tilde{\mathbf{Y}}$ on to new coordinates \mathbf{W} (i.e. PCs or eigenvectors)

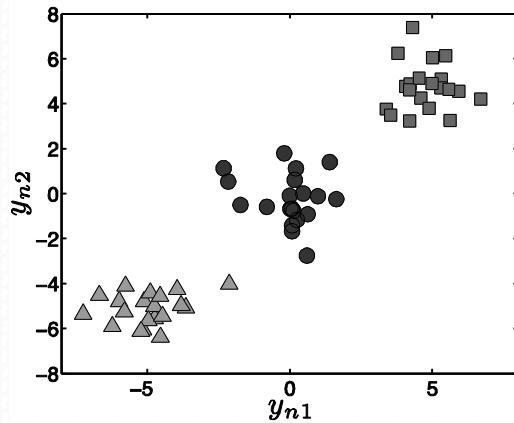
$$\mathbf{X} = \tilde{\mathbf{Y}}\mathbf{W}$$

How to choose D ?

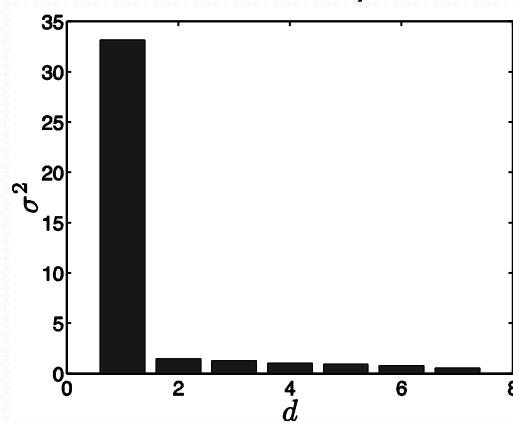
- We get M eigenvectors from the $M \times M$ covariance matrix
 - i.e. M new dimensions
- How to choose D dimensions (such that $D \ll M$)?
 - Application domain knowledge
 - Visualization
 - Computational burden
 - ‘interesting’ structure (i.e. defined by variance)
 - Post-processing results
- Total variation ($\sum_{d=1}^M \lambda_d$)
 - Percentage variation preserved by D eigenvectors can be estimated as: $[\sum_{d=1}^D \lambda_d / \sum_{d=1}^M \lambda_d] * 100$

PCA: example

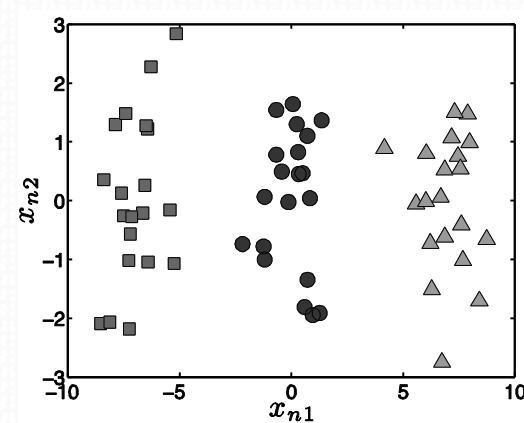
- Consider this 2D data for 3-classes
 - Five additional dimensions were added with random values $\mathcal{N}(0,1)$
 - Perform PCA and dimensionality reduction



(a) First two dimensions of the data objects \mathbf{y}_n



(b) The seven eigenvalues (variances of the projected dimensions)

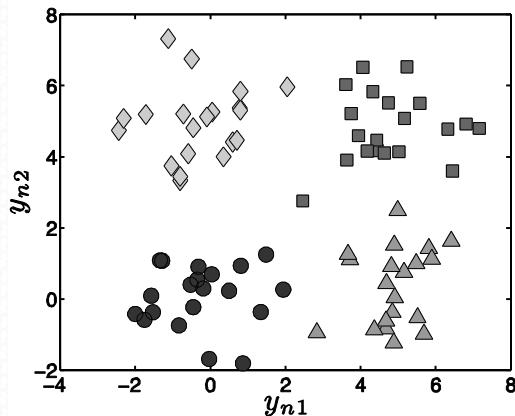


(c) The data projected onto the first two principal components

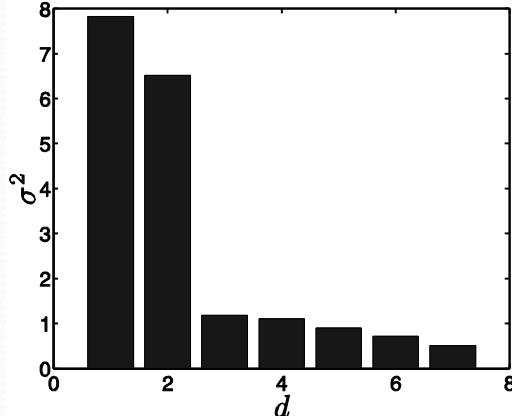
- It determines the dominant modes of variation from within the data and then projects data onto this 'natural' coordinate system

PCA: example

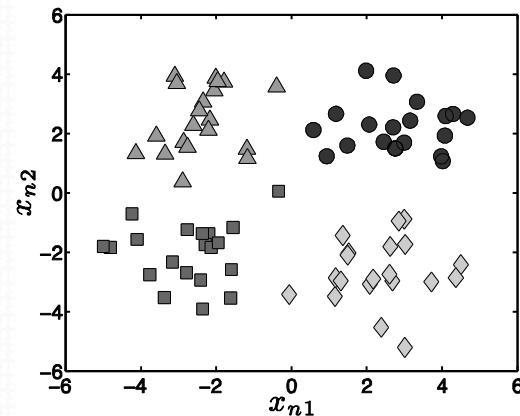
- Consider this 2D data for 4-classes
 - Five additional dimensions were added with random values $\mathcal{N}(0,1)$
 - Perform PCA and dimensionality reduction



(a) First two dimensions of the data objects y_n



(b) The seven eigenvalues (variances of the projected dimensions)



(c) The data projected onto the first two principal components

- It determines the dominant modes of variation from within the data and then projects data onto this 'natural' coordinate system

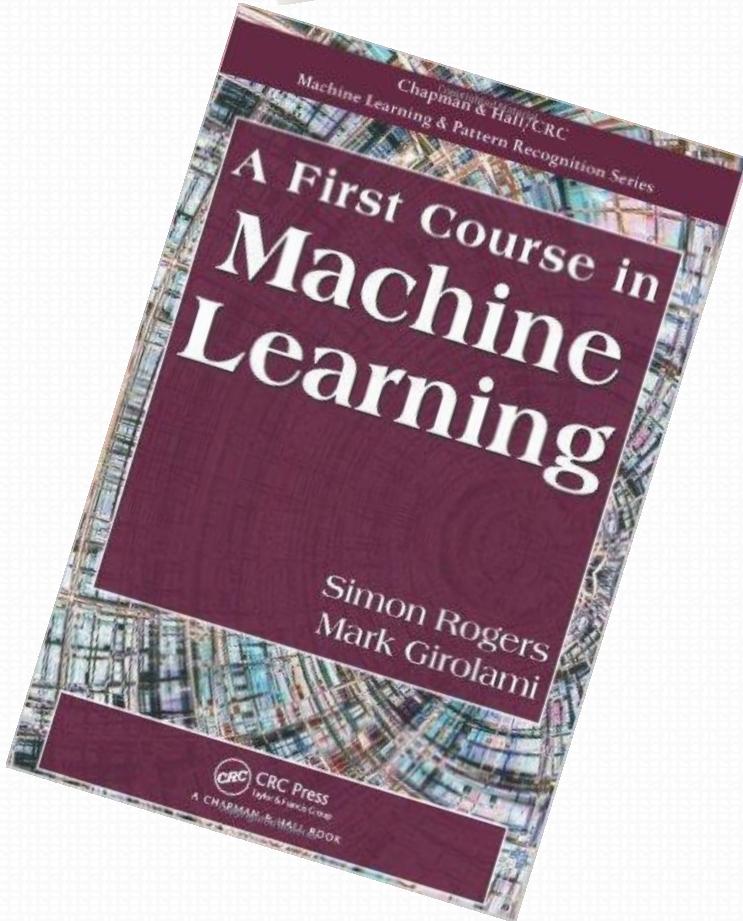
Summary

- “curse of dimensionality” with high-dimensional data can pose various problems
- Data projection to preserve ‘interesting’ characteristics within data
 - Avoid “curse of dimensionality”
- Dimensionality reduction is a form of unsupervised learning
- Dimensionality reduction is often used as a pre-processing step before classification or clustering
 - The ‘success’ of dimensionality reduction can be evaluated by subsequent processing operation

Exercise (ungraded)

- Experiment with MATLAB code – pcaexample.m
(from FCML book website)
- Experiment with MATLAB code – pcaexample2.m
(from FCML book website)
- Experiment with MATLAB code –
rgbeyepcaexample.m (from Canvas)

CREDITS



Author's material
(Simon Rogers)



Ata Kaban



Iain Styles



Thank You

Machine Learning, Machine Learning (extended)

**10 – Supervised Learning:
Ensemble Methods**

Kashif Rajpoot

k.m.rajpoot@cs.bham.ac.uk

**School of Computer Science
University of Birmingham**

Outline

- Ensemble methods
 - Boosting
 - Bagging
- Decision tree
- Random forests

Ensemble methods

- Combining ‘weak classifiers’ in order to produce a ‘strong classifier’
 - “two heads are better than one”
- **Boosting**: train a new classifier focusing on training samples misclassified by an earlier classifier
 - Weak classifier: any classifier better than a random guess
 - AdaBoost
- **Bagging (bootstrap aggregation)**: generate new training data as a random subset of original data and train a new classifier on this subset
 - Weak classifier: a decision tree classifier
 - Random forests

Modified from Randomized Forests for Visual Recognition

Jamie Shotton Tae-Kyun Kim Björn Stenger



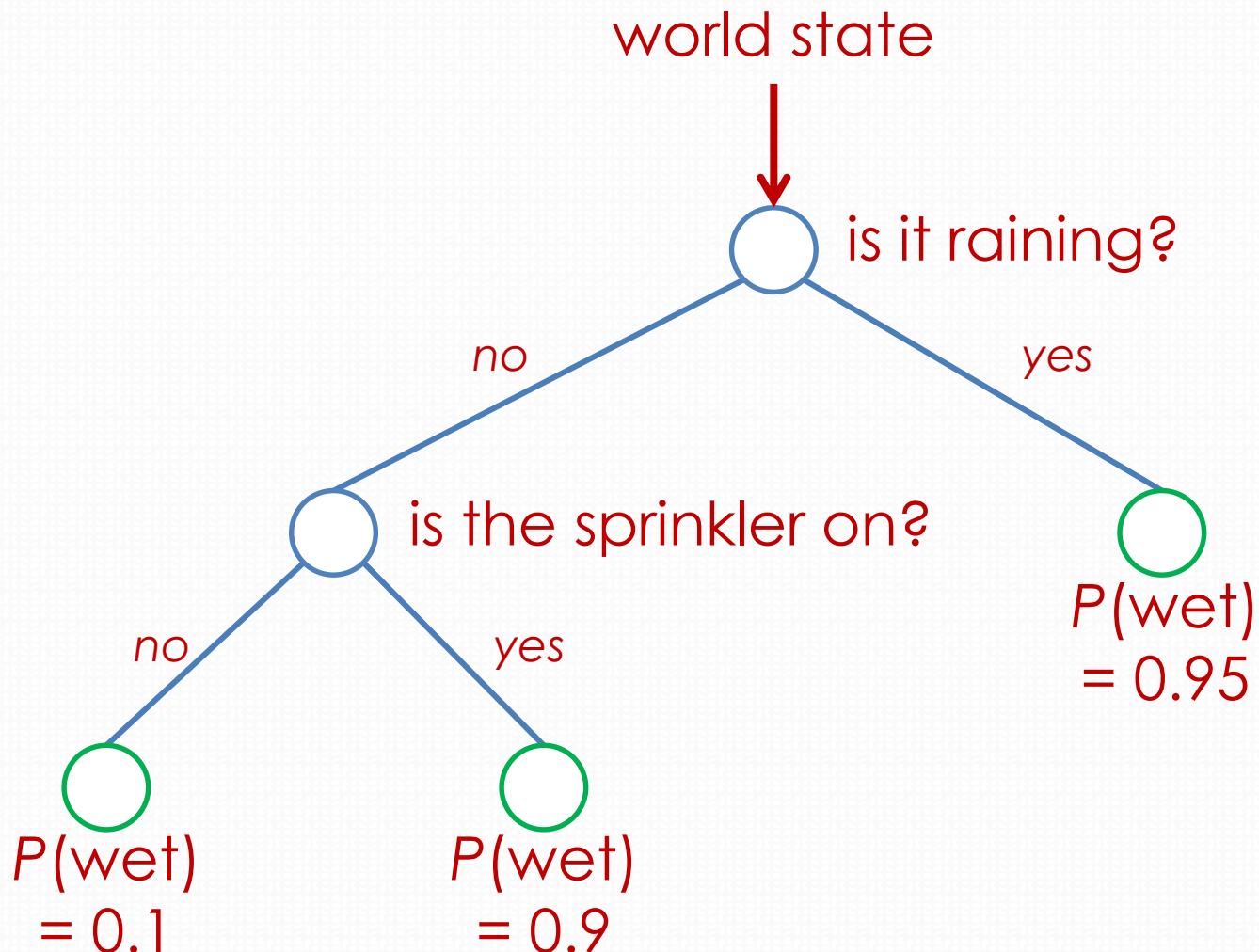
ICCV 2009, Kyoto, Japan

Randomized decision forests

- Very fast tool for classification
- Good generalization through randomized training
- Inherently multi-class
- Simple training / testing algorithms

“Randomized Decision Forests” = “Randomized Forests” =
“Random Forests™”

Basics: is the grass wet?



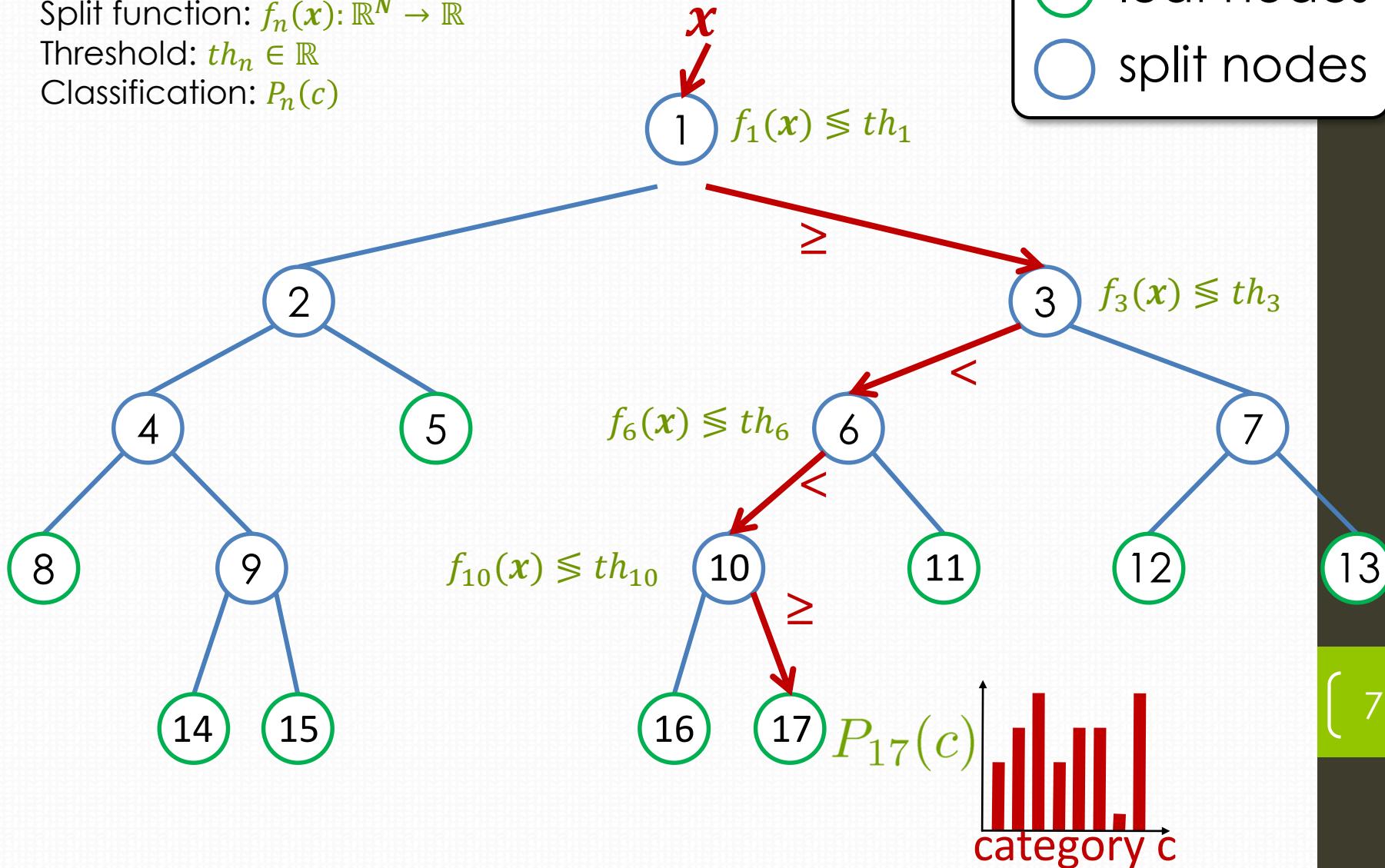
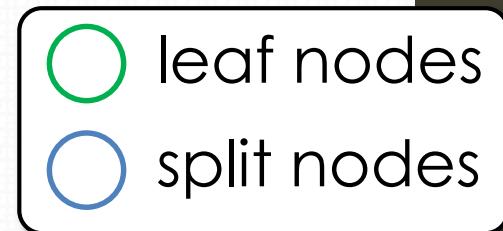
Basics: binary decision tree

Feature/attribute vector: $x \in \mathbb{R}^N$

Split function: $f_n(x): \mathbb{R}^N \rightarrow \mathbb{R}$

Threshold: $th_n \in \mathbb{R}$

Classification: $P_n(c)$

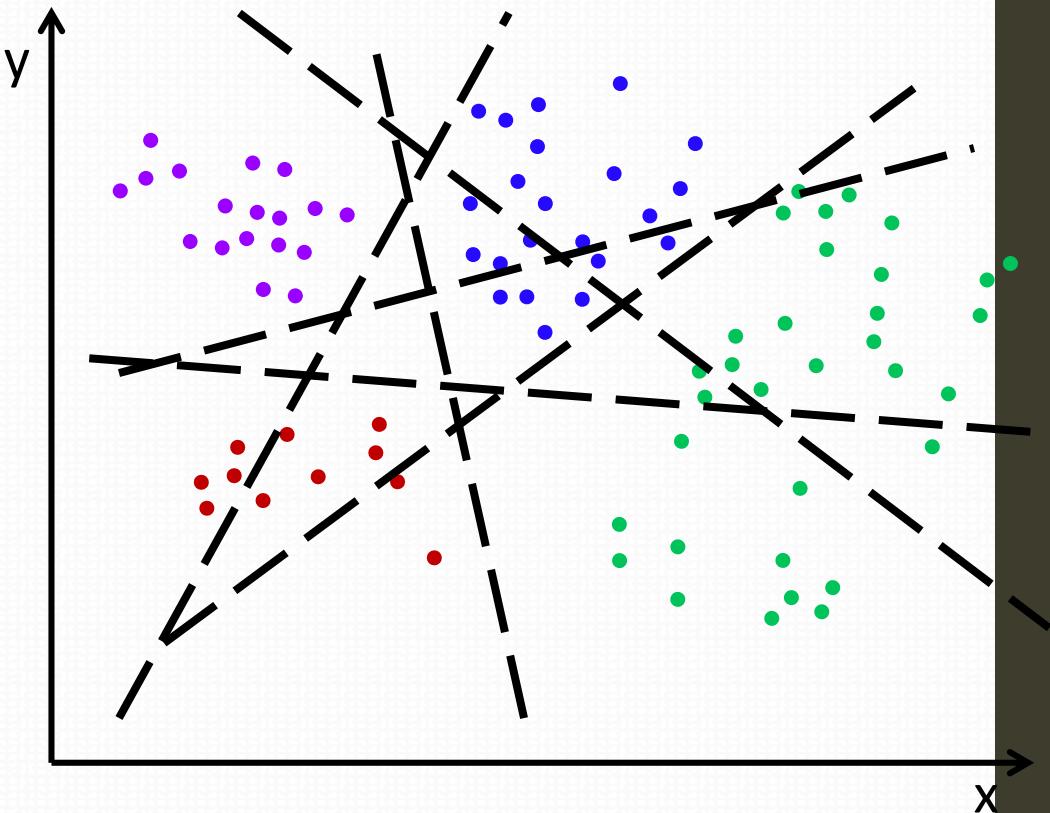


Decision tree classification: pseudo-code

```
double[] ClassifyDT(node, x)
    if node.IsSplitNode then
        if node.f(x) >= node.th then
            return ClassifyDT(node.right, x)
        else
            return ClassifyDT(node.left, x)
        end
    else
        return node.P
    end
end
```

Learning example

- Try several lines, 'chosen at random'
- Keep line that best separates data
 - Maximize information gain
- Recurse

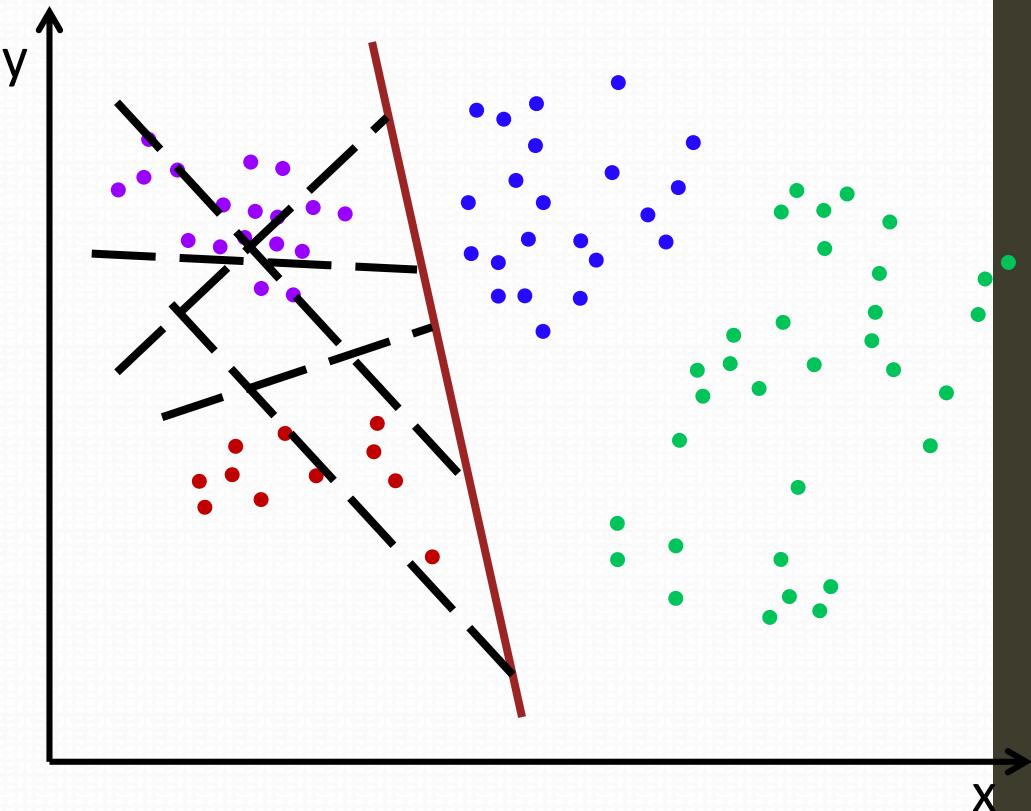


- feature vectors are x, y coordinates:
- split functions are lines with parameters a, b :
- threshold determines intercepts:
- four classes: purple, blue, red, green

$$\begin{aligned} \mathbf{x} &= [x, y]^T \\ f_n(\mathbf{x}) &= ax + by \\ th_n \end{aligned}$$

Learning example

- Try several lines, 'chosen at random'
- Keep line that best separates data
 - Maximize information gain
- Recurse

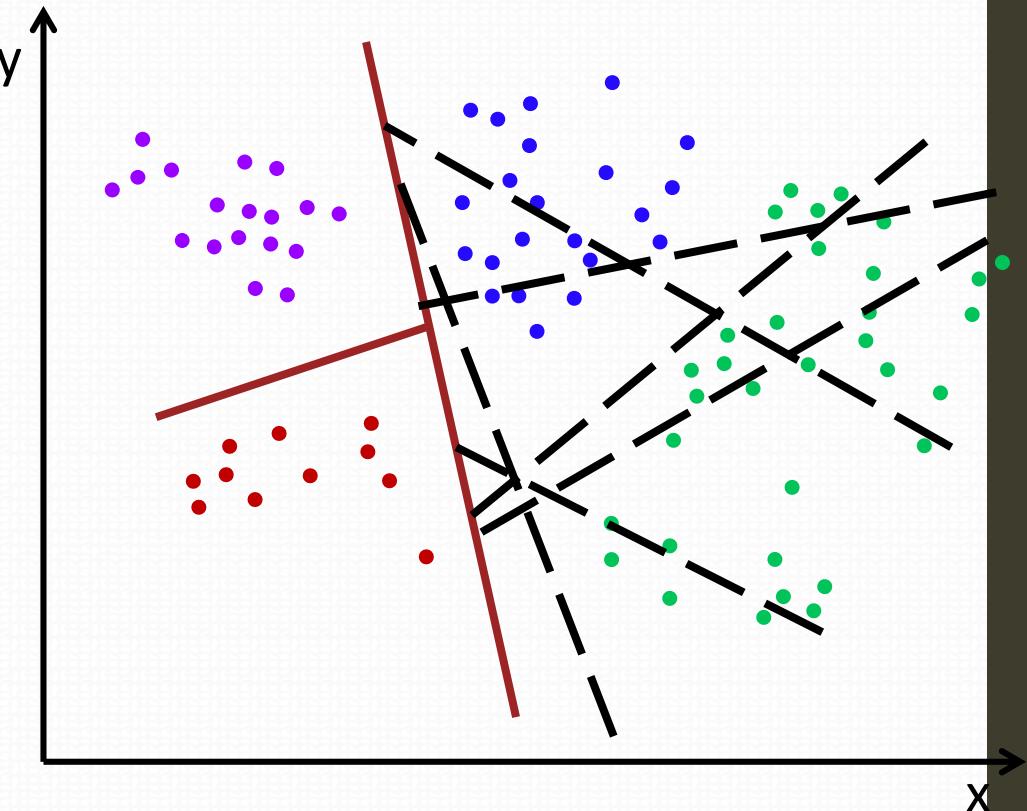


- feature vectors are x, y coordinates:
- split functions are lines with parameters a, b :
- threshold determines intercepts:
- four classes: purple, blue, red, green

$$\begin{aligned} \mathbf{x} &= [x, y]^T \\ f_n(\mathbf{x}) &= ax + by \\ th_n \end{aligned}$$

Learning example

- Try several lines, 'chosen at random'
- Keep line that best separates data
 - Maximize information gain
- Recurse

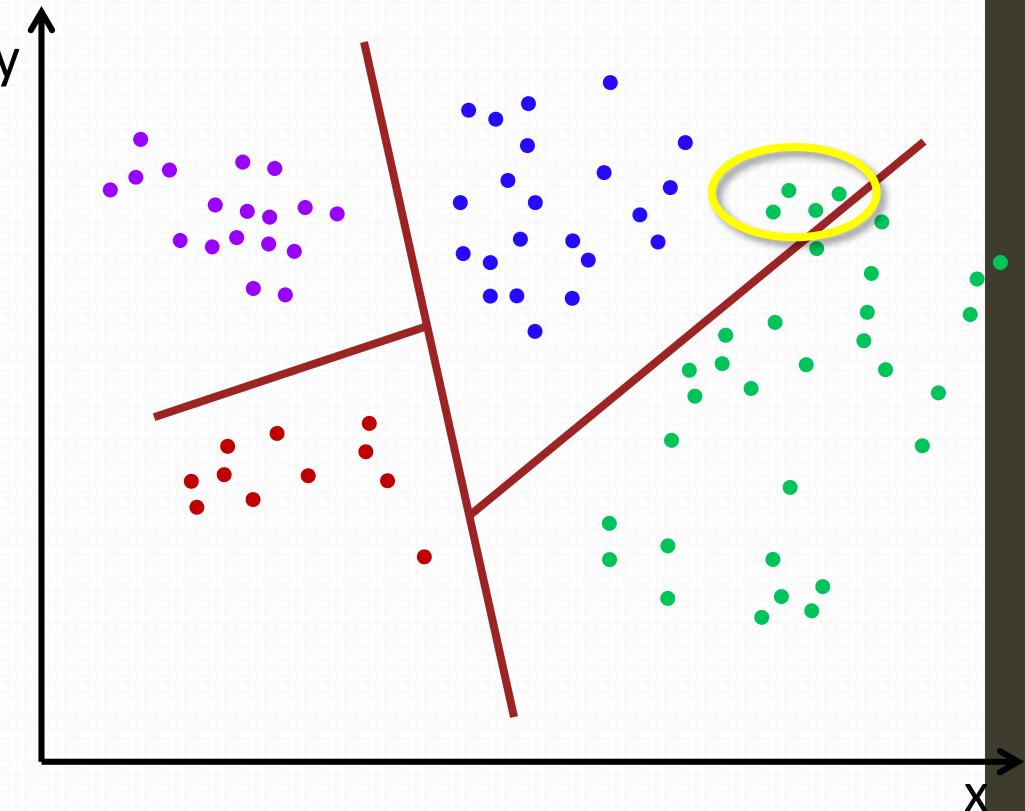


- feature vectors are x, y coordinates:
- split functions are lines with parameters a, b :
- threshold determines intercepts:
- four classes: purple, blue, red, green

$$\begin{aligned} \mathbf{x} &= [x, y]^T \\ f_n(\mathbf{x}) &= ax + by \\ th_n \end{aligned}$$

Learning example

- Try several lines, 'chosen at random'
- Keep line that best separates data
 - Maximize information gain
- Recurse



- feature vectors are x, y coordinates:
- split functions are lines with parameters a, b :
- threshold determines intercepts:
- four classes: purple, blue, red, green

$$\begin{aligned} \mathbf{x} &= [x, y]^T \\ f_n(\mathbf{x}) &= ax + by \\ th_n \end{aligned}$$

Randomized learning

- Randomness in:
 - Bagging: randomly select the subset of data at the root of a tree
 - Randomly select the features at a tree node
 - “feature = attribute” in machine learning
 - Randomly select the threshold value

Randomized learning

- Randomly select X_n examples as a subset from X examples
- Recursively split X_n examples at node n

left split $\hookrightarrow X_l = \{x_i \in X_n | f(x_i) < th\}$ threshold

right split $\hookrightarrow X_r = \{x_i \in X_n | f(x_i) \geq th\}$

function of example i 's feature vector

Randomized learning

- Features $f(\mathbf{x})$ chosen at random from feature pool F
- Threshold th chosen at random in range
 - $th \in (\min(f(\mathbf{x})), \max(f(\mathbf{x})))$
- Choose f and th to maximize an objective function (e.g. information gain, Gini index)
 - Estimates whether it's “good” to distribute data further

Randomized learning

- $P_n(c)$ is the histogram (i.e. count) of example labels of class c which reached node n
- For example, at a leaf node, if 200 training example reach and there are 3 classes with following count, then the $P_n(c)$ is estimated as:

	$c = 1$	$c = 2$	$c = 3$
Count of examples	12	134	54
$P_n(c)$	12/200	134/200	54/200
$P_n(c)$	0.06	0.67	0.27

Implementation details

- How many features and thresholds to try?
 - just one = “extremely randomized”
 - few → fast training, may under-fit
 - many → slower training, may over-fit
- When to stop growing the tree?
 - maximum depth
 - minimum information gain

Decision tree learning: pseudo-code

```
TreeNode LearnDT(X)
repeat featureTests times
    let f = RndFeature()
    let r = EvaluateFeatureResponses(X, f)

    repeat threshTests times
        let th = RndThreshold(r)
        let (X_l, X_r) = Split(X, r, th)

        let gain = InfoGain(X_l, X_r)
        if gain is best then remember f, th, X_l, X_r
    end
end

if best gain is sufficient
    return SplitNode(f, th, LearnDT(X_l), LearnDT(X_r))
else
    return LeafNode(HistogramExamples(X_s))
end
end
```

Binary decision tree: summary

- Fast greedy training algorithm
 - can search infinite pool of features
 - heterogeneous pool of features
- Fast testing algorithm
- Needs careful choice of hyper-parameters
 - maximum depth
 - number of features and thresholds to try

Information gain and entropy

- Information gain: gain in information (i.e purity of data according to class labels) by split of data from parent to child nodes in the tree

- $$IG(f_n) = E(\text{parent}) - \frac{|X_l|}{|X_n|} * E(\text{left}) - \frac{|X_r|}{|X_n|} * E(\text{right})$$

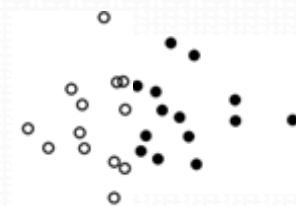
where $|X_n|$ denotes number of data samples at node n

- Entropy: measure of disorder (or impurity) in a bunch of data samples

- $$E = - \sum_{c=1}^C P_c \log_2(P_c)$$



Low entropy



High entropy

Information gain and entropy

- $E = -\sum_{c=1}^C P_c \log_2(P_c) = ?$
 - $E = -P_{Flu=Y} \log_2(P_{Flu=Y}) - P_{Flu=N} \log_2(P_{Flu=N}) = ?$
 - $E = -\frac{5}{8} \log_2 \left(\frac{5}{8}\right) - \frac{3}{8} \log_2 \left(\frac{3}{8}\right) = 0.9544$

chills	runny nose	headache	fever	Flu?
Y	N	Mild	Y	N
Y	Y	No	N	Y
Y	N	Strong	Y	Y
N	Y	Mild	Y	Y
N	N	No	N	N
N	Y	Strong	Y	Y
N	Y	Strong	N	N
Y	Y	Mild	Y	Y

Information gain and entropy

$$E(left) = ?$$

$$E(chills = Y) = -P_{Flu=Y} \log_2(P_{Flu=Y}) - P_{Flu=N} \log_2(P_{Flu=N}) = ?$$

$$E(chills = Y) = -\frac{3}{4} \log_2 \left(\frac{3}{4}\right) - \frac{1}{4} \log_2 \left(\frac{1}{4}\right) = 0.8113$$

- $IG(chills) = E(parent) - \frac{|X_l|}{|X_n|} * E(left) - \frac{|X_r|}{|X_n|} * E(right) = ?$
- $IG(chills) = 0.9544 - 0.5 * 0.8113 - 0.5 * 1 = 0.0488$

$$E(right) = ?$$

$$E(chills = N) = -P_{Flu=Y} \log_2(P_{Flu=Y}) - P_{Flu=N} \log_2(P_{Flu=N}) = ?$$

$$E(chills = N) = -\frac{2}{4} \log_2 \left(\frac{2}{4}\right) - \frac{2}{4} \log_2 \left(\frac{2}{4}\right) = 1$$

$$\frac{|X_l|}{|X_n|} = ?$$

$$\frac{|X_{chills=Y}|}{|X_n|} = ?$$

$$\frac{|X_{chills=Y}|}{|X_n|} = \frac{4}{8}$$

$$\frac{|X_r|}{|X_n|} = ?$$

$$\frac{|X_{chills=N}|}{|X_n|} = ?$$

$$\frac{|X_{chills=N}|}{|X_n|} = \frac{4}{8}$$

chills	runny nose	headache	fever	Flu?
Y	N	Mild	Y	N
Y	Y	No	N	Y
Y	N	Strong	Y	Y
N	Y	Mild	Y	Y
N	N	No	N	N
N	Y	Strong	Y	Y
N	Y	Strong	N	N
Y	Y	Mild	Y	Y

Information gain and entropy

$$E(left) = ?$$

$$E(runny = Y) = -P_{Flu=Y} \log_2(P_{Flu=Y}) - P_{Flu=N} \log_2(P_{Flu=N}) = ?$$

$$E(runny = Y) = -\frac{4}{5} \log_2 \left(\frac{4}{5}\right) - \frac{1}{5} \log_2 \left(\frac{1}{5}\right) = 0.7219$$

- $IG(runny) = E(parent) - \frac{|X_l|}{|X_n|} * E(left) - \frac{|X_r|}{|X_n|} * E(right) = ?$
- $IG(runny) = 0.9544 - 0.625 * 0.7219 - 0.375 * 0.9183 = 0.1589$

$$E(right) = ?$$

$$E(runny = N) = -P_{Flu=Y} \log_2(P_{Flu=Y}) - P_{Flu=N} \log_2(P_{Flu=N}) = ?$$

$$E(runny = N) = -\frac{1}{3} \log_2 \left(\frac{1}{3}\right) - \frac{2}{3} \log_2 \left(\frac{2}{3}\right) = 0.9183$$

$$\frac{|X_l|}{|X_n|} = ?$$

$$\frac{|X_{runny=Y}|}{|X_n|} = ?$$

$$\frac{|X_{runny=Y}|}{|X_n|} = \frac{5}{8}$$

$$\frac{|X_r|}{|X_n|} = ?$$

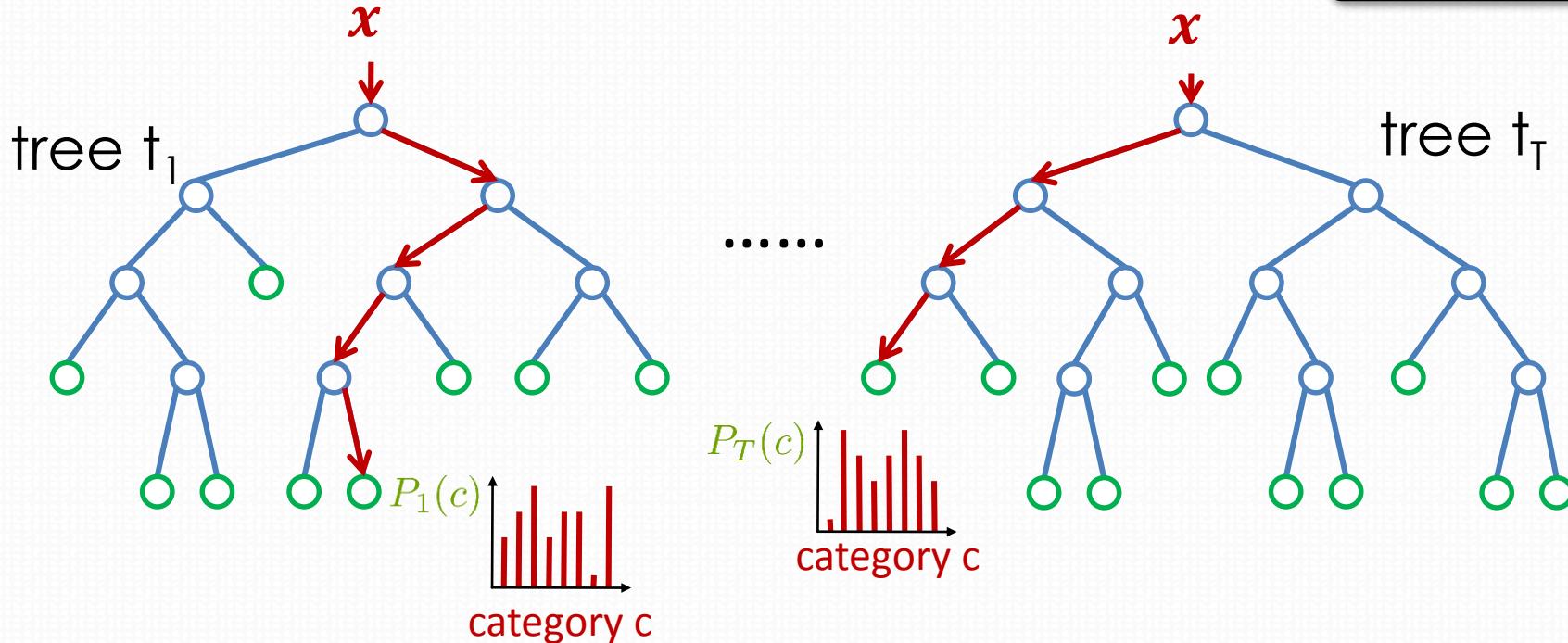
$$\frac{|X_{runny=N}|}{|X_n|} = ?$$

$$\frac{|X_{runny=N}|}{|X_n|} = \frac{3}{8}$$

chills	runny nose	headache	fever	Flu?
Y	N	Mild	Y	N
Y	Y	No	N	Y
Y	N	Strong	Y	Y
N	Y	Mild	Y	Y
N	N	No	N	N
N	Y	Strong	Y	Y
N	Y	Strong	N	N
Y	Y	Mild	Y	Y

A forest of trees

- Forest is ensemble of several decision trees



- Classification

$$\cdot P(c|x) = \frac{1}{T} \sum_{tr=1}^T P_{tr}(c|x)$$

Decision forests: pseudo-code

```
double[] ClassifyDF(forest, x)
    // allocate memory
    let P = double[forest.CountClasses]

    // loop over trees in forest
    for tr = 1 to forest.CountTrees
        let P' = ClassifyDT(forest.Tree[tr], x)
        P = P + P' // sum distributions
    end

    // normalise
    P = P / forest.CountTrees
end
```

Learning a forest

- Divide training examples into T subsets
 - $X_{tr} \subseteq X$
 - improves generalization
 - reduces memory requirements & training time
- Subsets are chosen at random
- Subsets can have overlap (and usually do)
- Train each decision tree tr on subset X_{tr}

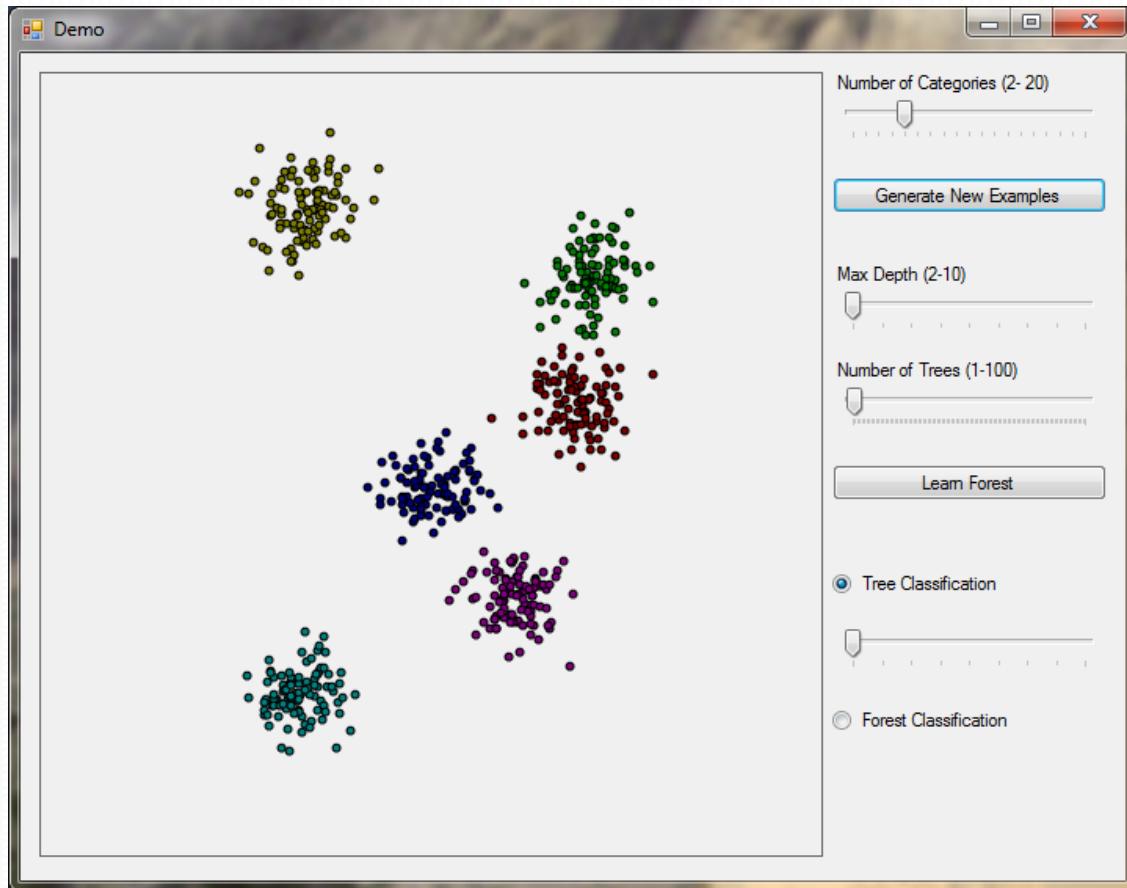
Learning a forest: pseudo-code

```
Forest LearnDF(countTrees, X)
    // allocate memory
    let forest = Forest(countTrees)

    // loop over trees in forest
    for tr = 1 to countTrees
        let X_tr = RandomSplit(X)
        forest[tr] = LearnDT(X_tr)
    end

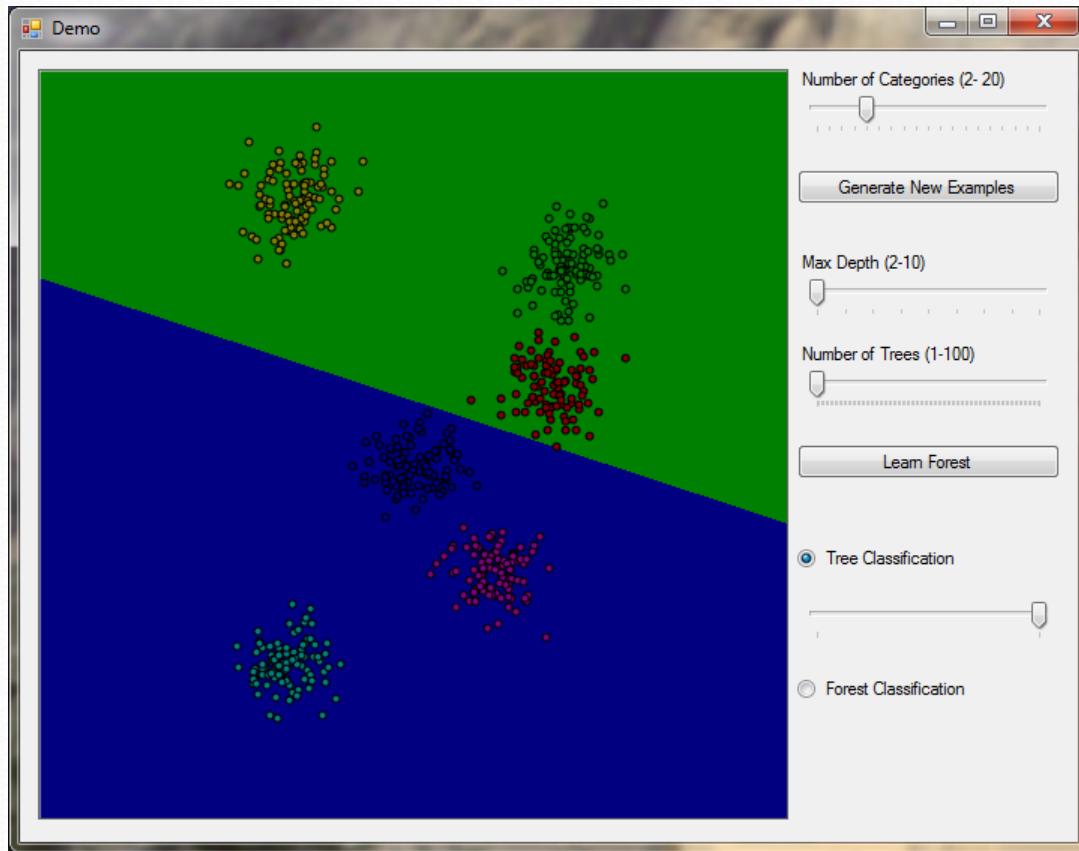
    // return forest object
    return forest
end
```

Random forest: demo



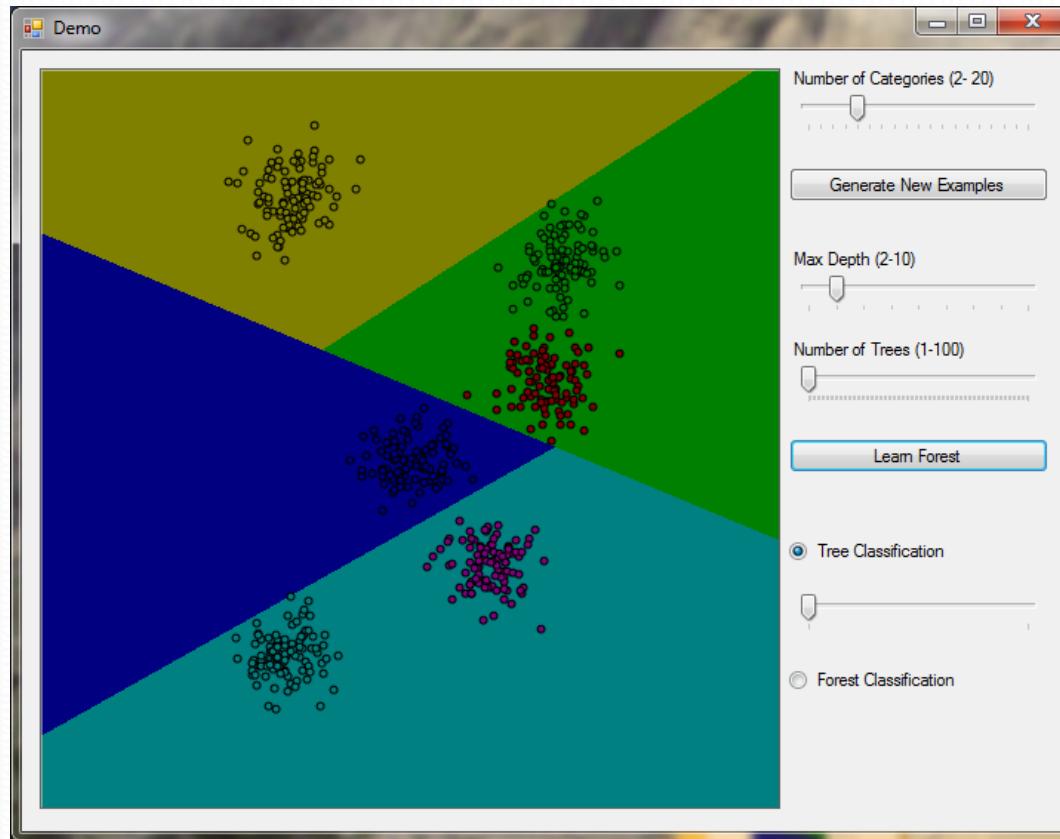
6 classes in a 2 dimensional feature space.
Split functions are lines in this space.

Random forest: demo



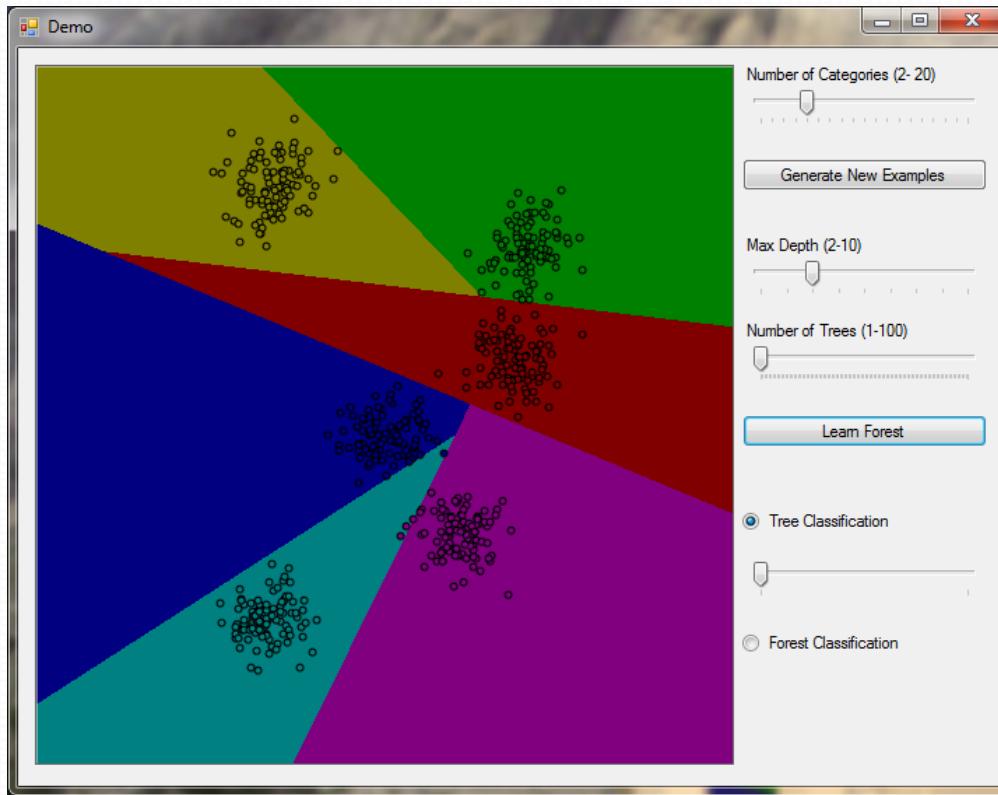
With a depth 2 tree, you cannot separate all six classes.

Random forest: demo



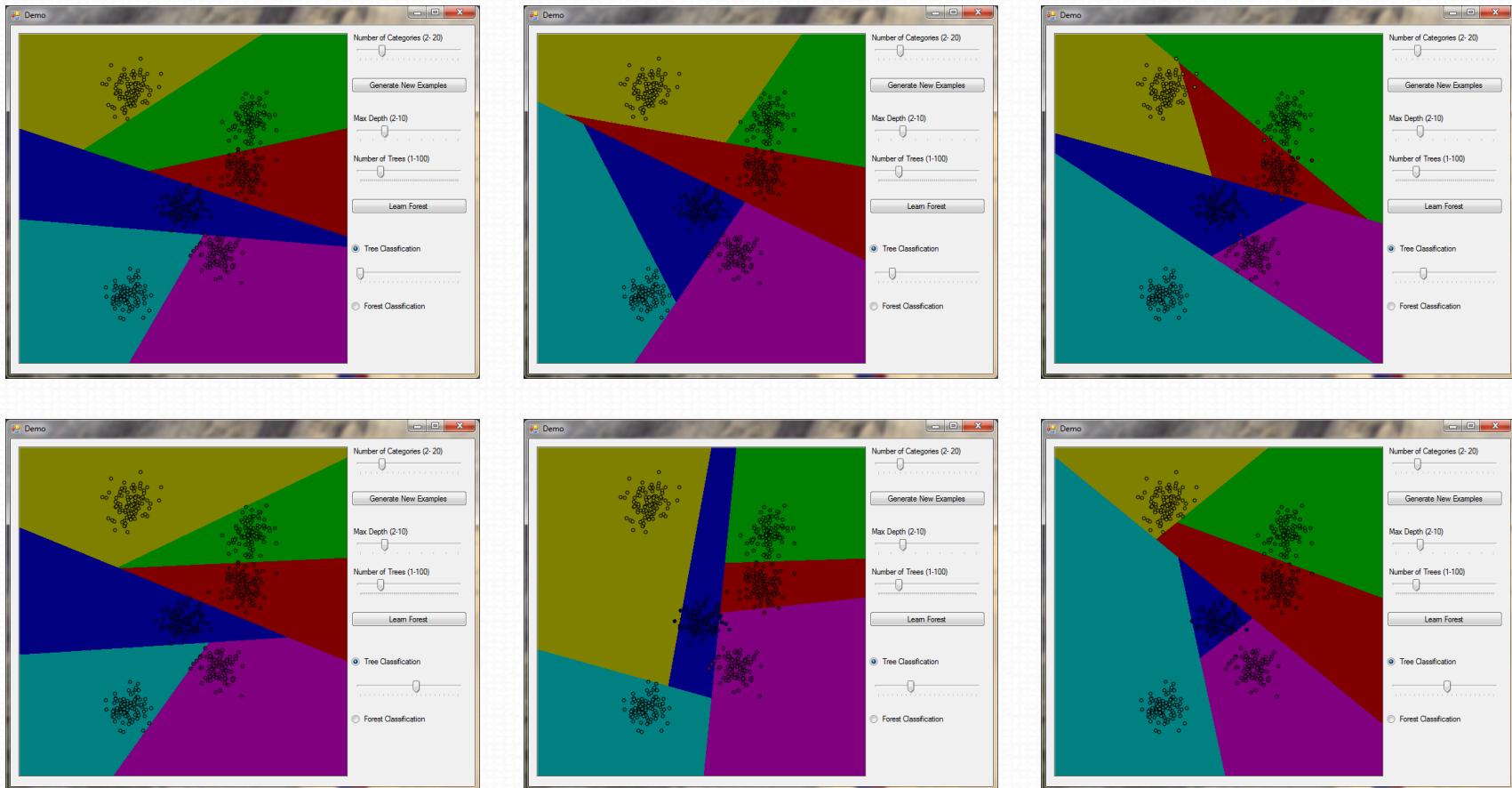
With a depth 3 tree, you are doing better, but still cannot separate all six classes.

Random forest: demo



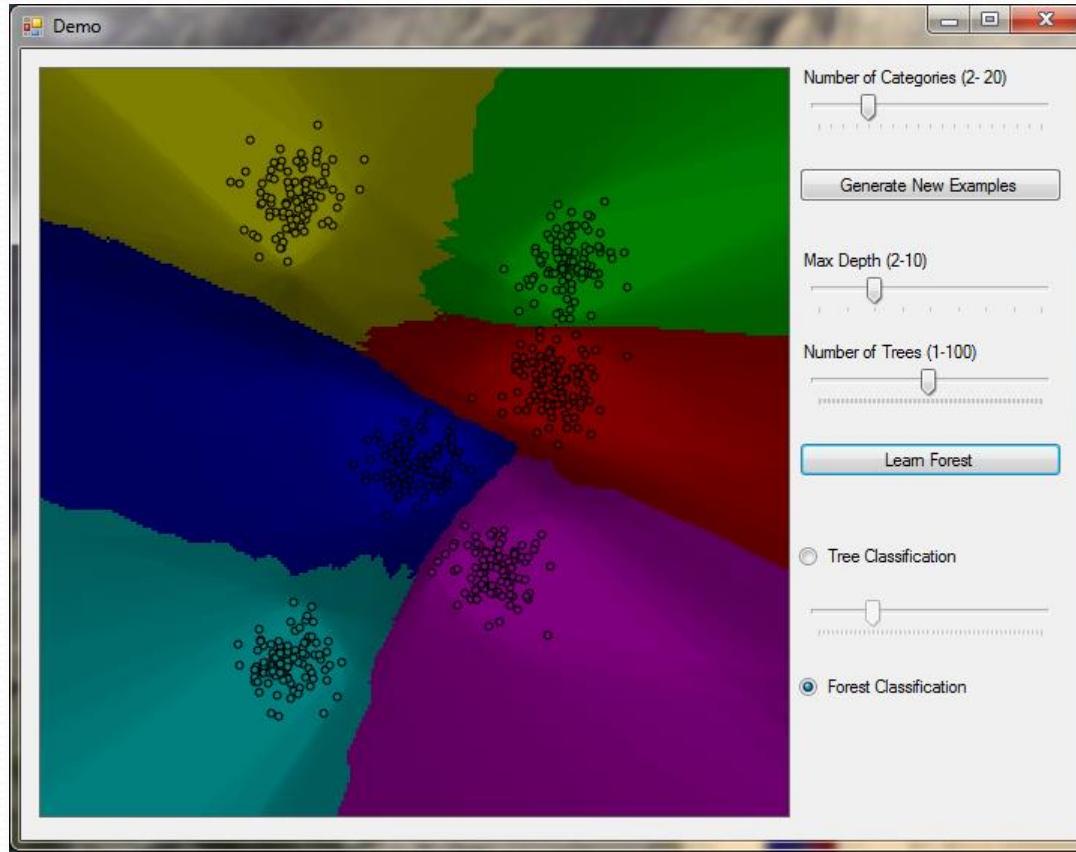
With a depth 4 tree, you now have at least as many leaf nodes as classes, and so are able to classify most examples correctly.

Random forest: demo



Different trees within a forest can give rise to very different decision boundaries, none of which is particularly good on its own.

Random forest: demo



But averaging together many trees in a forest can result in decision boundaries that look very sensible, and are even quite close to the max margin classifier.

Summary

- Very fast classification algorithm
- Accuracy comparable with other classifiers
- Simple to implement

Further reading/References

- ICCV'2009 Tutorial
 - <http://jamie.shotton.org/work/presentations/ICCV2009Tutorial1Part1.pptx>
- Random Forests for Regression and Classification; by Adele Cutler
 - <http://www.math.usu.edu/adele/RandomForests/Ovronnaz.pdf>
- Machine Learning; by Tom Mitchell (Chapter 3)
- Pattern Classification; By Duda, Hart, Stork (Chapter 8)



Thank You