



UNIVERSITY OF
LIVERPOOL

SECOND SEMESTER EXAMINATIONS 2017/18

Data Mining and Visualisation

TIME ALLOWED : Two and a Half Hours

INSTRUCTIONS TO CANDIDATES

Answer **FOUR** questions.

If you attempt to answer more questions than the required number of questions, the marks awarded for the excess questions answered will be discarded (starting with your lowest mark).

Question 1 Consider the sentence

S = “I would love to go to London by train tomorrow or at least by bus next week”.

Answer the following questions.

- A. Given the stop word list [*to, at, by, the, or, would*], represent S as a bag-of-unigrams. **(2 marks)**

I, love, go, London, train, tomorrow, least, bus, next, week

- B. Generate all possible bigrams from S without removing the stop words. How many bigrams do you get? **(2 marks)**

16

- C. Generate all possible bigrams from S after removing the stop words from S . How many bigrams do you get? **(2 marks)**

9

- D. State one advantage of removing stop words in text mining tasks. **(2 marks)**

Reduction of the number of features, speed up, smaller model sizes.

- E. State one disadvantage of removing stop words in text mining tasks. **(2 marks)**

Removing features such as *to* or *would* will likely lose the information about respectively purpose/direction of an action or the modality of the text.

- F. What is meant by *part-of-speech* in text mining? **(2 marks)**

Part-of-speeches are nouns, verbs, adjectives, adverbs, etc. that indicate syntactic categories of words.

- G. Given the unigram feature set {London, train, bus, John, week, love, Liverpool}, represent S by a binary-valued feature vector \mathbf{s} **(2 marks)**

Let us assign indexes to the features as follows: London=0, train=1, bus=2, John=3, week=4, love=5, Liverpool=6. Then $\mathbf{s} = (1, 1, 1, 0, 1, 1, 0)^\top$.

- H. Compute the ℓ_2 norm of \mathbf{s} . **(2 marks)**

$\sqrt{5}$

- I. Compute the ℓ_1 norm of \mathbf{s} . **(2 marks)**

5

- J. Assume that the d -dimensional pre-trained word embedding for a word w is given by $v(w)$. Let us denote the set of unigrams computed in part (A) above by \mathcal{V} . Propose a method to create a d -dimensional embedding for the sentence s using \mathcal{V} and the word embeddings. **(3 marks)**

One approach to do this would be to compute the centroid of the word embeddings for the words in the sentence. Specifically, $\mathbf{s} = \frac{1}{|\mathcal{V}|} \sum_{w \in \mathcal{V}} v(w)$

- K. State a disadvantage of the sentence embedding method that you described in part (J). **(2 marks)**

The sentence embedding method described in part J ignores the ordering of the words in a sentence. Another disadvantage would be that it weighs all words equally when computing the sentence embedding.

- L. Propose a method to overcome the disadvantage that you described in part (K). **(2 marks)**

We can weigh each word using inverse document frequency (IDF) (or some other salience measure) before computing the centroid to incorporate the importance of a word when representing a sentence. To make the sentence embedding order-sensitive, we can use a recurrent neural network or use bigram (or higher-order n -grams) embeddings rather than using unigram embeddings.

Question 2 Consider a training dataset $\mathcal{D} = \{(\mathbf{x}_n, t_n)\}_{n=1}^4$, where $\mathbf{x}_n \in \mathbb{R}^2$ and $t_n \in \{-1, 1\}$. Here, $\mathbf{x}_1 = (0, 1)^\top$, $\mathbf{x}_2 = (-1, 0)^\top$, $\mathbf{x}_3 = (0, -1)^\top$, and $\mathbf{x}_4 = (1, 0)^\top$. We would like to train a binary Perceptron on \mathcal{D} parametrised by the weight vector $\mathbf{w} = (\alpha, \beta)^\top$ and bias b . Answer the following questions.

- A.** Show that if the labels are $t_1 = t_2 = 1$ and $t_3 = t_4 = -1$, then \mathcal{D} can be perfectly classified by the Perceptron with $\mathbf{w} = (-1, 1)$ and $b = 0$. **(4 marks)**

$$\begin{aligned}\mathbf{w}^\top \mathbf{x}_1 + 0 &= 1 \geq 0 \\ \mathbf{w}^\top \mathbf{x}_2 + 0 &= 1 \geq 0 \\ \mathbf{w}^\top \mathbf{x}_3 + 0 &= -1 < 0 \\ \mathbf{w}^\top \mathbf{x}_4 + 0 &= -1 < 0\end{aligned}$$

Therefore \mathcal{D} is perfectly classified by this Perceptron.

- B.** Now let us relabel \mathcal{D} such that $t_1 = t_4 = 1$ and $t_2 = t_3 = -1$. Initialising $\alpha = \beta = b = 0$ and visiting the training instances in \mathcal{D} once in the order $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$, and \mathbf{x}_4 , compute the final weight vector and the bias. **(4 marks)**

\mathbf{x}_1 is correctly classified so the weight vector does not change after observing \mathbf{x}_1 . However, \mathbf{x}_2 is misclassified and the weight vector is updated to $(0, 0)^\top - (-1, 0)^\top = (1, 0)^\top$ and the bias is updated to $b = -1$. Then, \mathbf{x}_3 and \mathbf{x}_4 are correctly classified by this weight vector and bias. Therefore, the final weight vector will be $(1, 0)$ and the bias will be $b = -1$.

- C.** Now let us relabel \mathcal{D} such that $t_1 = t_3 = 1$ and $t_2 = t_4 = -1$. Assuming that $b = 0$, write the conditions that must be satisfied by the activation scores for each of the four points in \mathcal{D} , if it is to be correctly classified by $\mathbf{w} = (\alpha, \beta)$. **(4 marks)**

$$\begin{aligned}(\alpha, \beta)^\top (0, 1) \geq 0 &\rightarrow \beta \geq 0 \\ (\alpha, \beta)^\top (-1, 0) < 0 &\rightarrow \alpha > 0 \\ (\alpha, \beta)^\top (0, -1) \geq 0 &\rightarrow \beta \leq 0 \\ (\alpha, \beta)^\top (1, 0) < 0 &\rightarrow \alpha < 0\end{aligned}$$

- D.** Using the inequalities you wrote in part (c) show that there does not exist a Perceptron that can linearly separate \mathcal{D} with a zero bias. **(2 marks)**

The second and fourth inequalities cannot be satisfied simultaneously by α . Therefore, $\mathbf{w} = (\alpha, \beta)$ does not exist.

- E.** Show that when $t_1 = t_3 = 1$ and $t_2 = t_4 = -1$, there does not exist a Perceptron even with $b \neq 0$. **(3 marks)**

The inequalities with the bias are as follows:

$$\begin{aligned}\beta + b &\geq 0 \\ -\alpha + b &< 0 \\ -\beta + b &\geq 0 \\ \alpha + b &< 0\end{aligned}$$

From these four inequalities we have $b \geq 0$ and $b < 0$, which cannot be satisfied simultaneously by b .

- F.** Given a feature vector $\mathbf{x} = (x_1, x_2)^\top$, let us consider a kernel ψ that maps $\mathbf{x} \in \mathbb{R}^2$ to $\mathbf{x}^* \in \mathbb{R}^4$ such that $\mathbf{x}^* = (x_1, x_2, x_1^2, x_2^2)^\top$. Compute the projections \mathbf{x}_1^* , \mathbf{x}_2^* , \mathbf{x}_3^* , and \mathbf{x}_4^* respectively of \mathbf{x}_1 , \mathbf{x}_2 , \mathbf{x}_3 , and \mathbf{x}_4 under ψ . (4 marks)

$$\begin{aligned}\mathbf{x}_1^* &= (0, 1, 0, 1)^\top \\ \mathbf{x}_2^* &= (-1, 0, 1, 0)^\top \\ \mathbf{x}_3^* &= (0, -1, 0, 1)^\top \\ \mathbf{x}_4^* &= (1, 0, 1, 1)^\top\end{aligned}$$

- G.** Is $\mathcal{D}^* = \{(\mathbf{x}_1^*, 1), (\mathbf{x}_2^*, -1), (\mathbf{x}_3^*, 1), (\mathbf{x}_4^*, -1)\}$ linearly separable? If yes, then give a weight vector and a bias term of a Perceptron that would correctly classify all four instances in \mathcal{D}^* . If no, then explain why \mathcal{D}^* is not linearly separable. (4 marks)

\mathcal{D}^* can be perfectly classified (therefore linearly separable) by $\mathbf{w}^* = (0, 0, 1, 0)^\top$ and $b^* = 0$.

Question 3 Consider five data points in \mathbb{R}^2 given by $\mathbf{x}_1 = (0, 0)^\top$, $\mathbf{x}_2 = (1, 0)^\top$, $\mathbf{x}_3 = (1, 1)^\top$, $\mathbf{x}_4 = (0, 1)^\top$, and $\mathbf{x}_5 = (-1, -1)^\top$. Answer the following questions about this dataset.

- A.** Let us assume that we clustered this dataset into two clusters $\mathcal{S}_1 = \{\mathbf{x}_4, \mathbf{x}_3\}$ and $\mathcal{S}_2 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_5\}$. Moreover, let us represent \mathcal{S}_1 and \mathcal{S}_2 by two 2-dimensional vectors respectively μ_1 and μ_2 . Write the within-cluster sum of squares objective, $f(\mathcal{S}_1, \mathcal{S}_2)$ for this clustering. **(3 marks)**

$$f(\mathcal{S}_1, \mathcal{S}_2) = \|\mathbf{x}_4 - \mu_1\|^2 + \|\mathbf{x}_3 - \mu_1\|^2 + \|\mathbf{x}_1 - \mu_1\|^2 + \|\mathbf{x}_2 - \mu_2\|^2 + \|\mathbf{x}_5 - \mu_2\|^2$$

- B.** Write μ_1 and μ_2 that would minimise $f(\mathcal{S}_1, \mathcal{S}_2)$. **(4 marks)**

$$\frac{\partial f}{\partial \mu_1} = -2(\mathbf{x}_4 - \mu_1) - 2(\mathbf{x}_3 - \mu_1) = 0$$

Gives, $\mu_1 = (\mathbf{x}_4 + \mathbf{x}_3)/2 = (0.5, 1.0)$. Likewise, we can get $\mu_2 = (\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_5)/3 = (0, -1/3)$.

- C.** Assuming that we initialised μ_1 and μ_2 such that $\mu_1 = (0.5, 0.5)^\top$ and $\mu_2 = (-2, -2)^\top$. Following the procedure of k -means clustering, assign the data points $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5$ to the two clusters \mathcal{S}_1 and \mathcal{S}_2 represented respectively by μ_1 and μ_2 . **(2 marks)**

$$\mathcal{S}_1 = \{\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1\}, \mathcal{S}_2 = \{\mathbf{x}_5\}$$

- D.** Compute the next values for μ_1 and μ_2 following the assignment done in part (3). **(2 marks)**
 $\mu_1 = (0.5, 0.5)^\top$ and $\mu_2 = (-1, -1)^\top$

- E.** Has the k -means clustering converged after the update in part (D)? Explain your answer. **(4 marks)**

There is a 0.5 probability that x_1 could be assigned to \mathcal{S}_2 . If that happens, the clustering continues and has not converged. However, with 0.5 probability x_1 can get assigned to \mathcal{S}_2 , in which case clusters do not change and has converged.

- F.** Consider the two clusters $\mathcal{S}_1 = \{R, R, B\}$ and $\mathcal{S}_2 = \{R, B\}$ consisting of red (R) and blue (B) colour balls. Compute the purity for this clustering. **(3 marks)**

$$(2+1)/5 = 0.6$$

- G.** Compute the rand index for the clustering described in part (F). **(4 marks)**

TP=1, FP=5, FN=3, and TN=3. Therefore, rand index is $(1+3) / (1+3+5+3) = 0.33$

- H.** Compute the precision, recall and F-score for the clustering described in part (F). **(3 marks)**
 precision = 1/6, recall = 1/4, and F = 1/5

Question 4

- A.** Consider flipping a coin C and a dice D at the same time and observing the outcomes. The events corresponding to the two sides of the coin are denoted by c_1 (head) and c_2 (tail), whereas those for the dice are denoted by $d_1, d_2, d_3, d_4, d_5, d_6$ respectively for the six sides of the dice. The probability of an event e is denoted by $p(e)$. The joint observations from 40 trials are summarised in Table 1. Answer the following questions about this experiment.

	d_1	d_2	d_3	d_4	d_5	d_6
c_1	2	4	3	5	2	4
c_2	3	3	4	2	4	4

Table 1: Frequency of the events observed in the experiment.

- (a) Compute $p(c_1)$, $p(c_2)$ and decide whether C is a biased coin or not. **(3 marks)**
 $p(c_1) = p(c_2) = 20/40 = 0.5$. Therefore, C is an unbiased coin.
- (b) Compute the mutual information between C and D . (You do not need to simplify the logarithms) **(4 marks)**

$$\begin{aligned}
 I(C, D) &= \sum_{i,j} p(c_i, d_j) \log \frac{p(c_i, d_j)}{p(c_i)p(d_j)} \\
 &= \frac{2}{40} \log \frac{2 * 40}{20 * 5} + \frac{4}{40} \log \frac{4 * 40}{20 * 7} + \frac{3}{40} \log \frac{3 * 40}{20 * 7} + \frac{5}{40} \log \frac{5 * 40}{20 * 7} \\
 &\quad + \frac{2}{40} \log \frac{2 * 40}{20 * 6} + \frac{2}{40} \log \frac{2 * 40}{20 * 6} + \frac{4}{40} \log \frac{4 * 40}{20 * 8} + \frac{3}{40} \log \frac{3 * 40}{20 * 5} \\
 &\quad + \frac{3}{40} \log \frac{3 * 40}{20 * 7} + \frac{2}{40} \log \frac{2 * 40}{20 * 7} + \frac{4}{40} \log \frac{2 * 40}{20 * 6} + \frac{4}{40} \log \frac{4 * 40}{20 * 8}
 \end{aligned}$$

- B.** Consider four product reviews r_1, r_2, r_3, r_4 represented in a three dimensional feature space consisting of the unigrams *awesome*, *awful* and *burger*. The frequency of each unigram in each review is shown in Table 2 and their sentiment labels (1 and -1 respectively denote positive and negative sentiment). Answer the following questions.

Review	awesome	awful	burger	label (t)
r_1	2	0	3	1
r_2	2	1	0	1
r_3	0	2	2	-1
r_4	2	3	1	-1

Table 2: A set of four reviews represented using three attributes.

- (a) Compute the marginal probabilities $p(\text{awesome})$, $p(\text{awful})$ and $p(\text{burger})$. **(3 marks)**
 $p(\text{awesome}) = p(\text{awful}) = p(\text{burger}) = 6/18$

- (b)** Compute the conditional probabilities $p(\text{awesome}|t = 1)$, $p(\text{awful}|t = 1)$ and $p(\text{burger}|t = 1)$. **(3 marks)**

$$p(\text{awesome}|t = 1) = 4/8, p(\text{awful}|t = 1) = 1/8 \text{ and } p(\text{burger}|t = 1) = 3/8$$

- (c)** Compute $p(t = 1)$ and $p(t = -1)$. **(2 marks)**

$$p(t = 1) = 1/2, p(t = -1) = 1/2$$

- (d)** Compute $p(t = 1|r_4)$ (You do not need to simplify the answer). **(4 marks)**

$$\begin{aligned} p(t = 1|r_4) &= \frac{p(r_4|t = 1)p(t = 1)}{p(r_4)} \\ p(r_4|t = 1) &= p(\text{awesome}|t = 1)^2 \times p(\text{awful}|t = 1)^3 \times p(\text{burger}|t = 1)^1 \\ p(r_4) &= p(\text{awesome})^2 \times p(\text{awful})^3 \times p(\text{burger})^1 \\ p(t = 1|r_4) &= \frac{(4/8)^2(1/8)^2(3/8)(1/2)}{(1/3)^2(1/3)^3(1/3)} \end{aligned}$$

- (e)** Apply Laplace smoothing for the occurrences of attributes in reviews shown in Table 2 and compute $p(t = 1|r_3)$ using the smoothed counts. (You do not need to simplify the answer) **(6 marks)**

Smoothed counts are shown in Table 3. Therefore, we have,

Review	awesome	awful	burger	label (t)
r_1	3	1	4	1
r_2	3	2	1	1
r_3	1	3	3	-1
r_4	3	4	2	-1

Table 3: Smoothed counts.

$$\begin{aligned} p(t = 1|r_3) &= \frac{p(r_3|t = 1)p(t = 1)}{p(r_3)} \\ &= \frac{(6/14)^1(3/14)^3(5/14)^3(1/2)}{(1/3)^1(1/3)^3(1/3)^3} \end{aligned}$$

2 marks are awarded for computing the smoothed counts and another 4 marks for correctly computing $p(t = 1|r_3)$. No penalties for not simplifying the answers.

Question 5 Consider the neural network with one hidden layer shown in Figure 1. Here, a two-dimensional input (represented by two features x_1, x_2) is multiplied by a weight matrix \mathbf{W} and subsequently a nonlinear activation of $\tanh(\theta) = \frac{\exp(\theta) - \exp(-\theta)}{\exp(\theta) + \exp(-\theta)}$ is applied. The outputs after applying the activation at the hidden layer are z_1 and z_2 , which are linearly weighted respectively by u_1 and u_2 to compute the prediction y . The output y is compared against the target output t for the instance $\mathbf{x} = (x_1, x_2)^\top$ to compute the loss given by

$$E(\mathbf{x}, t) = \frac{1}{2}(y - t)^2$$

Answer the following questions.

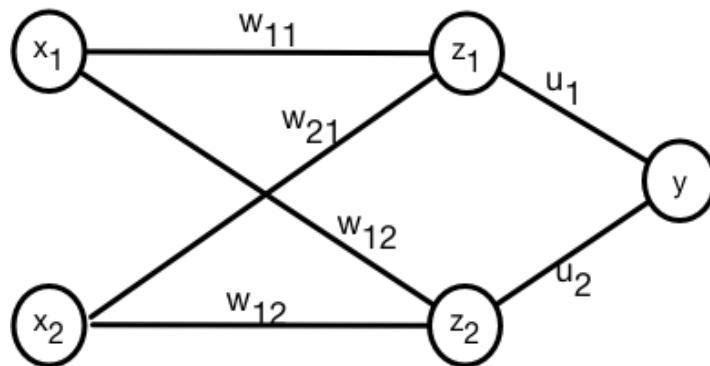


Figure 1: A neural network that takes two-dimensional feature vector and applies a \tanh activation in the hidden layer. The weight connecting nodes x_i and z_j is set to w_{ij} , whereas the weight connecting node z_i to the output node y is set to u_i .

- A. Express the output y in terms of z_1, u_1, z_2, u_2 . (2 marks)

$$y = z_1 u_1 + z_2 u_2$$

- B. Write z_1 using the input, weights in the first layer and the activation function. (2 marks)

$$z_1 = \tanh(x_1 w_{11} + x_2 w_{21})$$

- C. Write the loss gradient w.r.t. y . (2 marks)

$$\frac{\partial E}{\partial y} = (y - t)$$

- D. Show that

$$\frac{\partial z_1}{\partial w_{11}} = \left(1 - \tanh^2(x_1 w_{11} + x_2 w_{21})\right) x_1.$$

(4 marks)

This follows from the differentiation of $z_1 = \tanh(x_1 w_{11} + x_2 w_{21})$ w.r.t. w_{11} and applying $\tanh'(\theta) = 1 - \tanh^2(\theta)$

- E. Write $\frac{\partial E}{\partial w_{11}}$, the loss gradient w.r.t. w_{11} . (4 marks)

$$\frac{\partial E}{\partial w_{11}} = (y - t) u_1 x_1 \left(1 - \tanh^2(x_1 w_{11} + x_2 w_{21})\right)$$

F. Derive the stochastic gradient descent update rule for w_{11} . **(3 marks)**

$$w_{11}^{(k+1)} = w_{11}^{(k)} - \eta \frac{\partial E}{\partial w_{11}}. \text{ The loss gradient w.r.t to } w_{11} \text{ computed in part (E) must be substituted.}$$

G. Explain why it would be inappropriate to initialise the weights w_{ij} in the first layer to high numerical values. **(2 marks)**

This would increase the activation score that is input to the tanh and for high (positive or negative) scores, the gradient of the activation function will be close to zero. Therefore, the weights will not be updated after the initialisation. This is called the *saturation* of the activation.

H. State a solution that you can use to reduce the overfitting in a neural network. **(2 marks)**
 dropout, regularisation, early stopping, reduce the number of hidden layers.

I. Consider the ℓ_2 regularised version of the loss function given by

$$E(\mathbf{x}, t) = \frac{1}{2}(y - t)^2 + \lambda(w_{11}^2 + w_{12}^2 + w_{21}^2 + w_{22}^2) + \mu(u_1^2 + u_2^2),$$

where λ and μ are regularisation coefficients. Derive the update rule for w_{11} under this regularisation. **(4 marks)**

$$w_{11}^{(k+1)} = w_{11}^{(k)} - \eta \left((y - t)u_1x_1 \left(1 - \tanh^2(x_1w_{11} + x_2w_{21}) \right) \right) + 2\lambda w_{11}$$

COMP337+527: Data Mining and Visualization

2019-20
Shagufta Scanlon

Introduction to the Module

Acknowledgements

The slides are based on Professor Danushka Bollegala's slides, who taught this module until 2018-19.



Introduction

- University Teacher: Dr Shagufta Scanlon
- Office: G.14 Ashton Building (Ground Floor)
- Email: shagufta.scanlon@liverpool.ac.uk
- Research interests
 - Data Mining
 - Machine Learning
 - Natural Language Processing (NLP)

Course Material

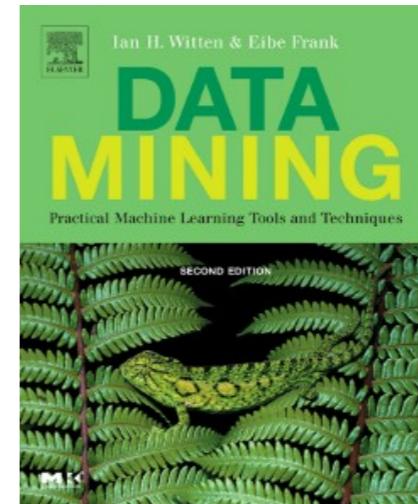
- Vital course pages – COMP337 and COMP527
- Vital – Announcements, module spec, lecture slides, stream lectures, lab material, assignments, relevant references, past exam papers, problem sets
- Discussion board (QA) on vital available
- Do not email me your questions. Instead post them on the discussion board so that others can also benefit from your QA

Evaluation

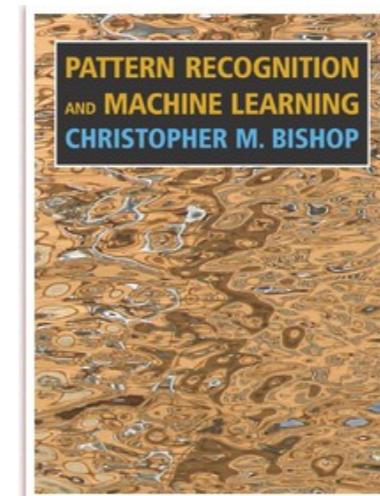
- 75% End of Year Exam
 - 2.5 hrs
 - short answers and/or essay type questions
 - Select 4 out of 5 questions
 - Past papers are available on vital
 - Some of the review questions might appear in the exam as well!
- 25% Continuous Assessment
 - Assignment 1: 12%
 - Assignment 2: 13%
 - Both assignments are programming oriented (in Python)
 - Attend lab sessions for Python+Data Mining (once a week)

References

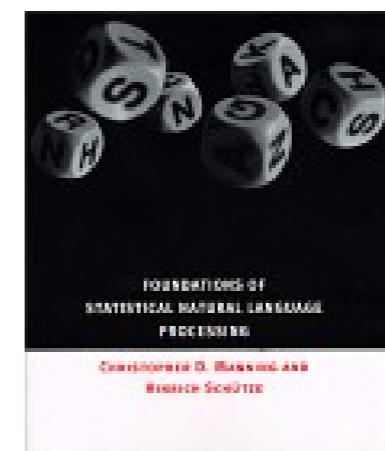
- Data Mining, *Witten*



- Pattern recognition and machine learning (PRML), *Bishop*



- Fundamentals of Statistical Natural Language Processing (FSNLP), *Manning*



Course Summary

- Data preprocessing (missing values, noisy data, scaling)
- Classification algorithms
 - Decision trees, Naive Bayes, k-NN, logistic regression, SVM
- Clustering algorithms
 - k-Means, k-Medoids, Hierarchical clustering
- Text Mining, Graph Mining, Information Retrieval
- Neural networks and Deep Learning
- Dimensionality reduction
- Visualization theory, t-SNE, embeddings
- Word embedding learning

Data Mining Introduction

Shagufta Scanlon



What is Data Mining?

- Various definitions
 - *The nontrivial extraction of implicit, previously unknown, and potentially useful information from data* (Piatetsky-Shapiro)
 - *...the automated or convenient extraction of patterns representing knowledge implicitly stored or captured in large databases, data warehouses, the Web, ... or data streams* (Han, page xxi)
 - *...the process of discovering patterns in data. The process must be automatic or (more usually) semiautomatic. The patterns discovered must be meaningful...* (Witten, page 5)

Applications of Text Mining

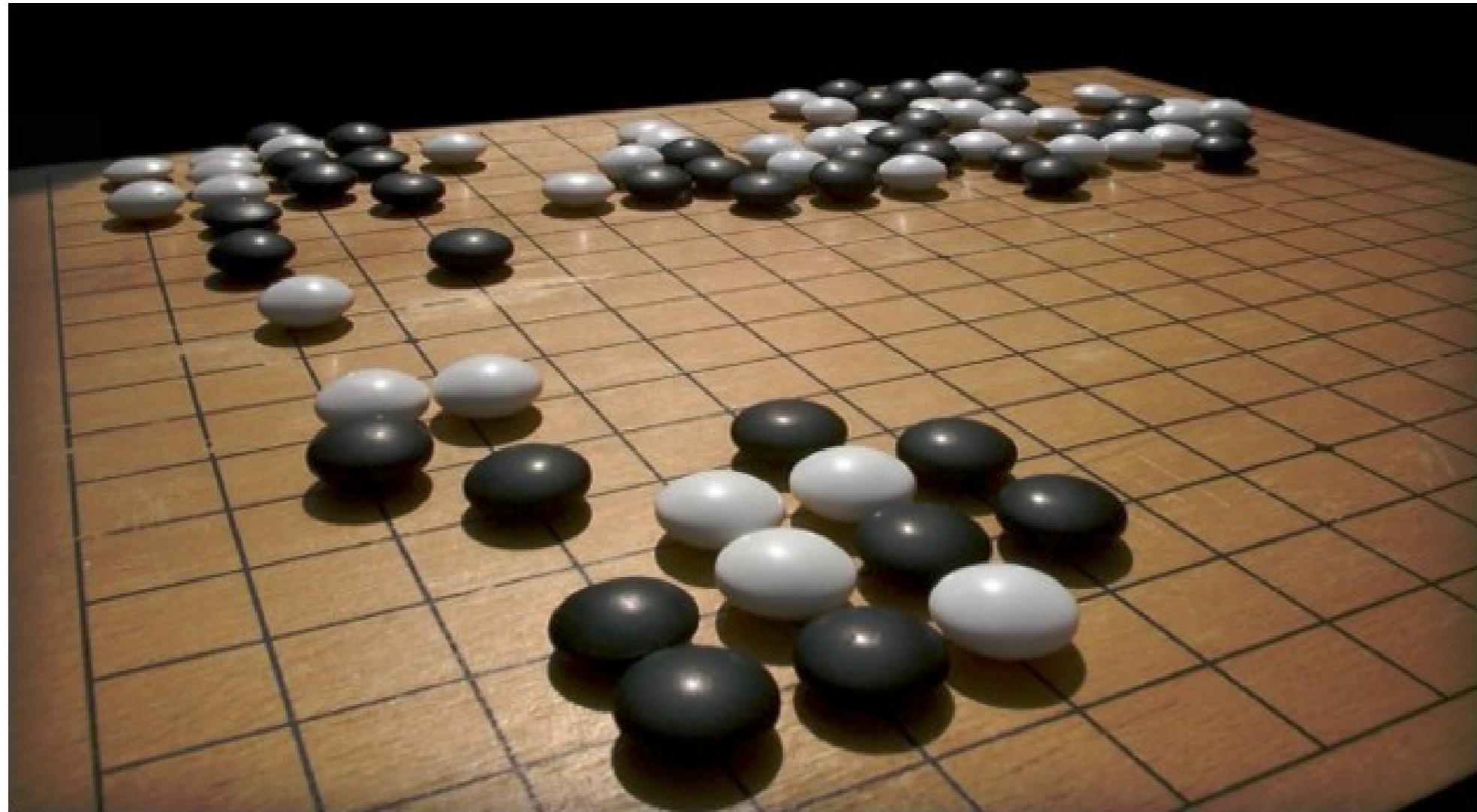


Computer program wins Jeopardy contest in 2011!

Applications of Deep Learning

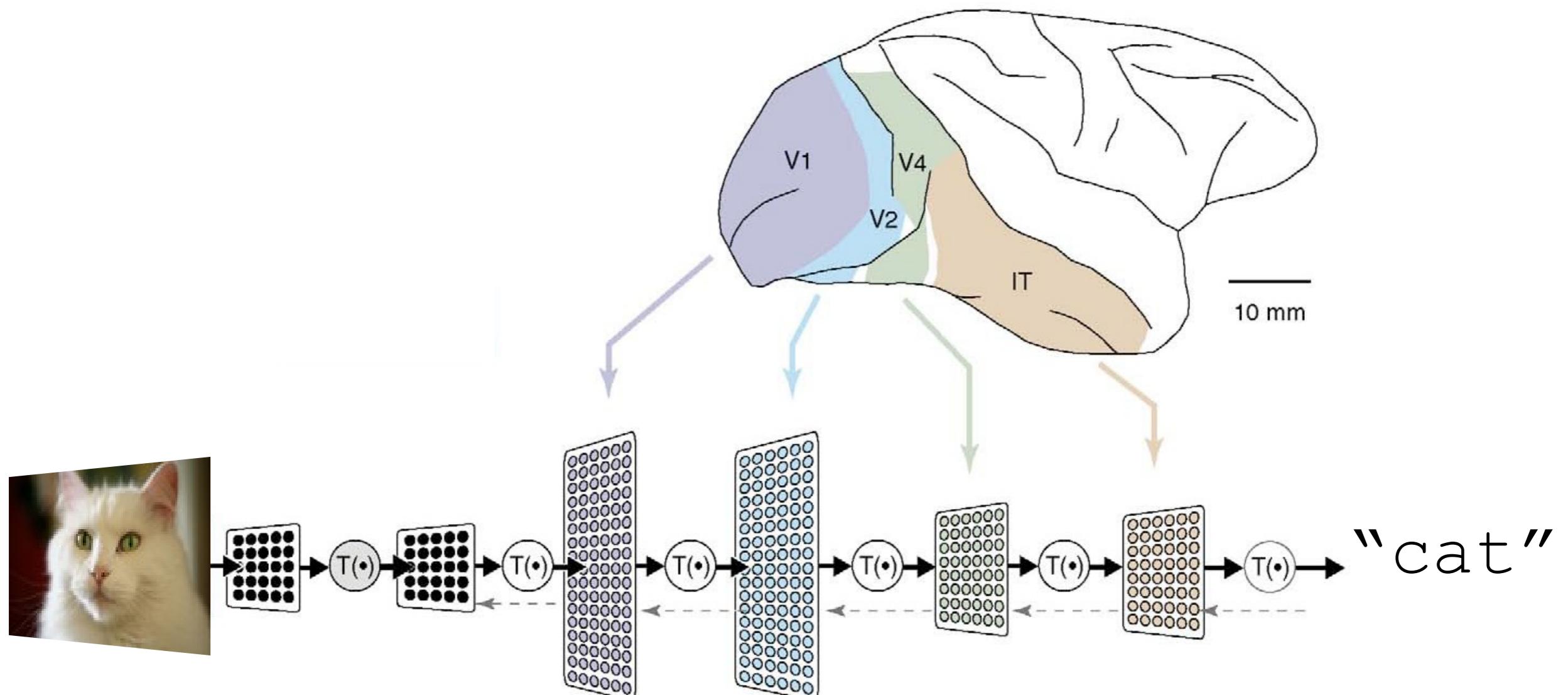
Google's DeepMind AI beats humans at the massively complex game Go

By Ryan Whitwam on January 27, 2016 at 4:00 pm | [11 Comments](#)



Google acquired the British

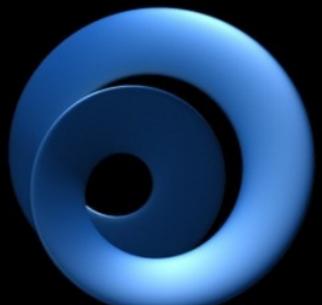
Deep Learning



An unsupervised neural network learns to recognize cats when trained using millions of you tube videos! (2012)



Deep Learning



DEEPMIND

WHO WE ARE

—
DEEPMIND IS PLEASED TO BE
PART OF **GOOGLE**.

Founded in 2011 by **Demis Hassabis, Shane Legg** and **Mustafa Suleyman**.

The team is based in London and was supported by some of the most iconic technology entrepreneurs and investors of the past decade.

Google acquires London-based AI (gaming) startup for USD 400M!

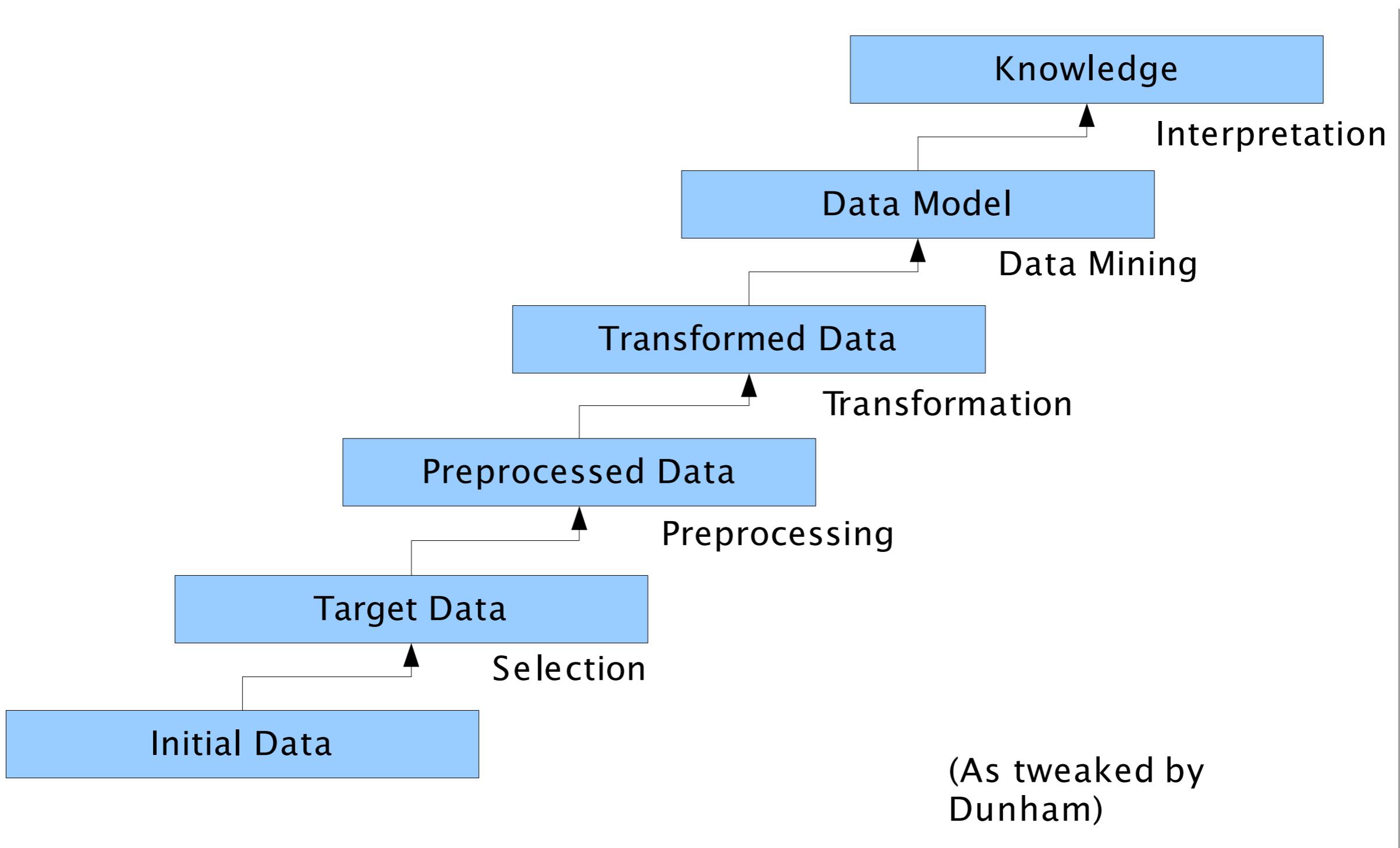
Industrial Interests

- Data Mining (DM)/ Machine Learning (ML)/ Natural Language Processing (NLP) experts are sought after by the CS industry
- Google research (Geoff Hinton/NN)
- Facebook AI research (Yann LeCun/Deep ML)
- Baidu (Andrew Ng)
- The ability to apply the algorithms we learn in this lecture (and their complex combinations) will greatly improve your employability in CS industries

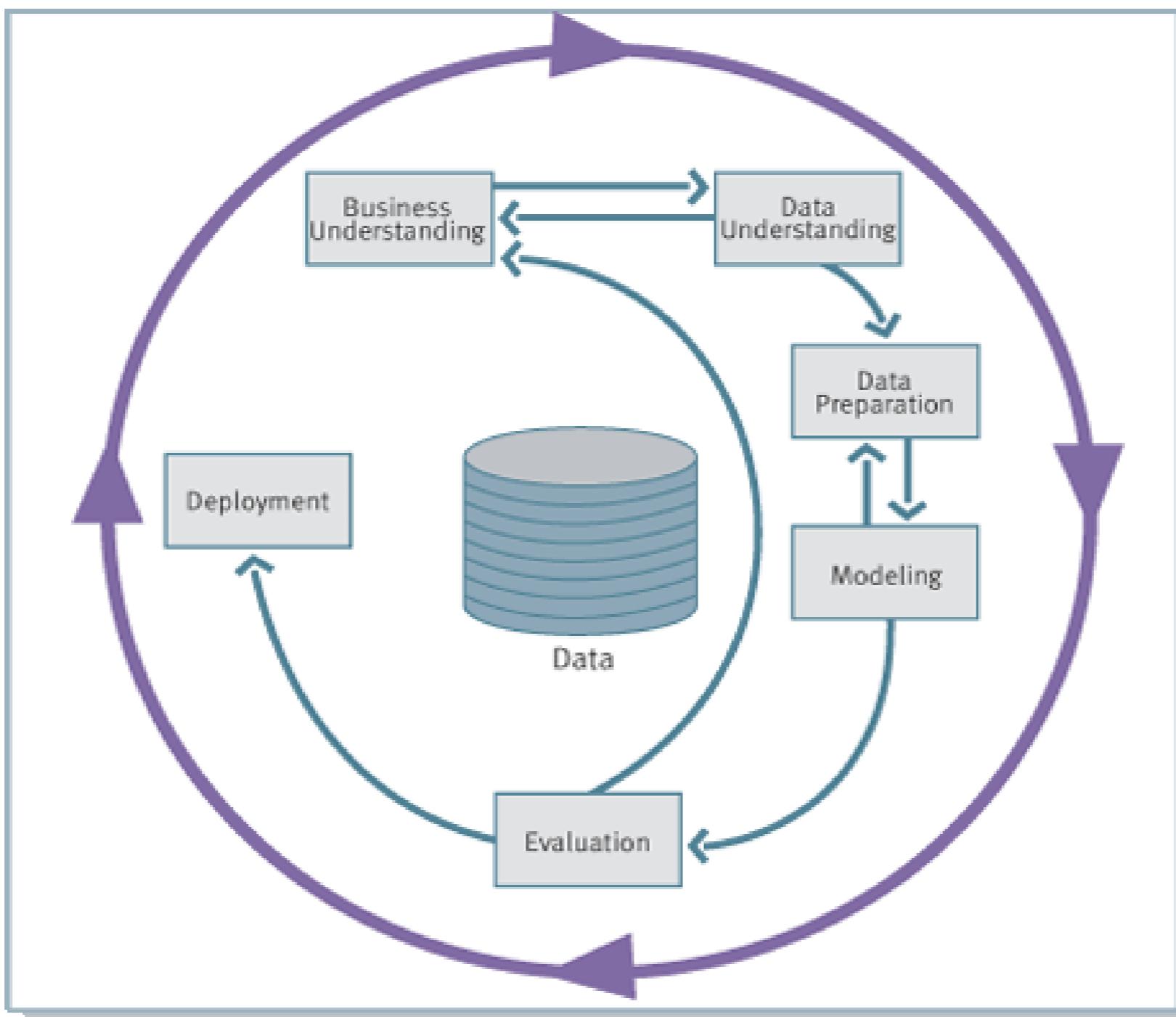
Academic Interests

- DM is an active research field.
- Top conferences
 - Knowledge Discovery and Data Mining (KDD) [<http://www.kdd.org/kdd2018/>]
 - Annual Conference of the Association for Computational Linguistics (ACL) [<http://acl2018.org/>]
 - International Word Wide Web Conference (WWW) [www2018.thewebconf.org]
 - International Conference on Machine Learning (ICML)
 - Neural and Information Processing (NIPS)
 - International Conference on Learning Representations (ICLR)

Piatetsky-Shapiro View



CRISP-DM View



Two Main Goals in DM

- Prediction
 - Build models that can predict future/unknown values of variables/patterns based on known data
 - Machine learning, Pattern recognition
- Description
 - Analyse given datasets to identify novel/interesting/useful patterns/rules/trends that can describe the dataset
 - clustering, pattern mining, associative rule mining

Broad Classification of Algorithms



Classification
Algorithms
(k-NN, Naive Bayes,
logistic regression,
SVM, Neural Networks,
Decision Trees)

Clustering Algorithms
(k-means, hierarchical
clustering)
visualization algorithms
(t-SNE, PCA)
Dimensionality reduction
(SVD, PCA)
Pattern/sequence mining

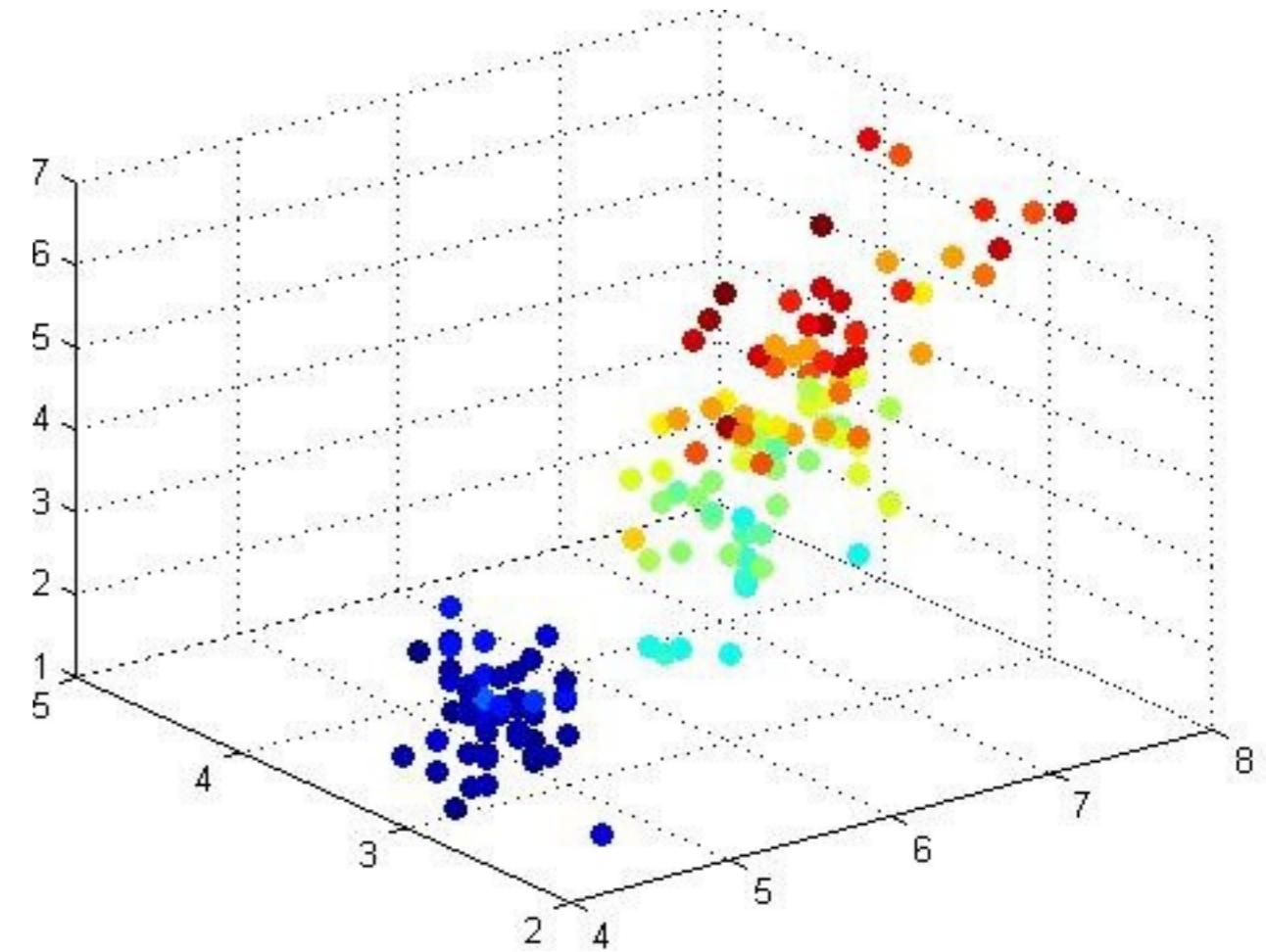
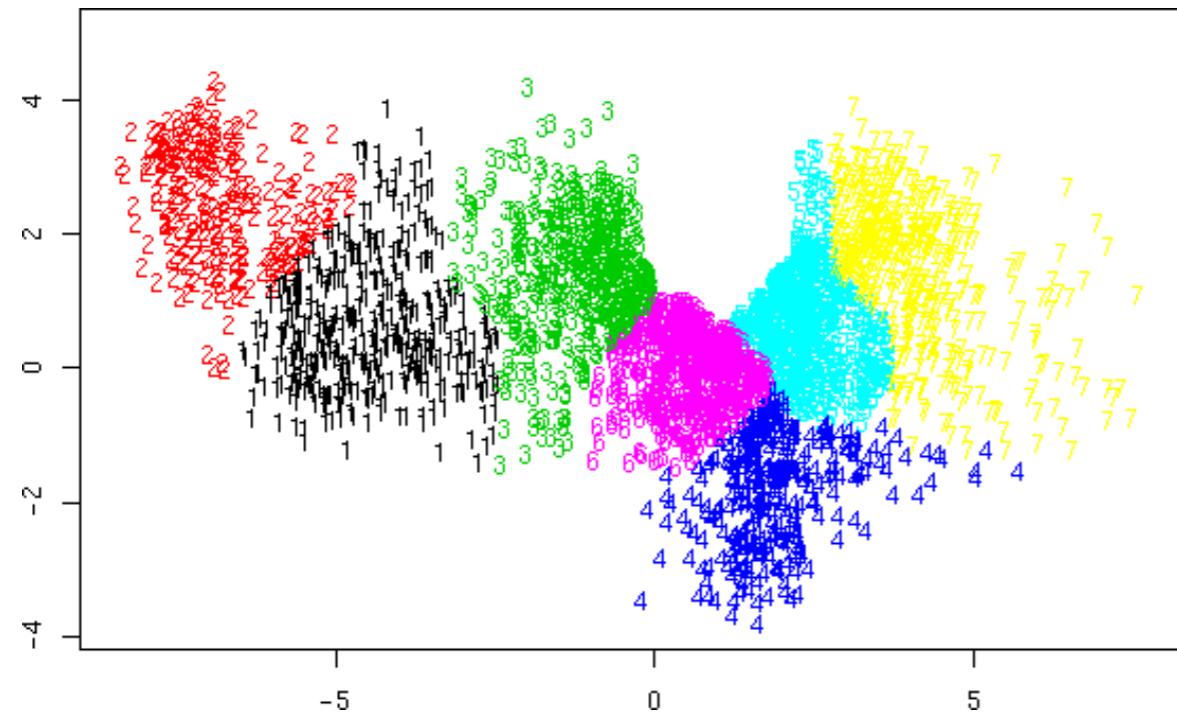
Classification

- Given a data point x , classify it into a set of **discrete classes**
- Example
 - Sentiment classification
 - *The movie was great* +1
 - *The food was cold and tasted bad* -1
 - Spam vs. non-spam email classification
- We want to learn a classifier $f(x)$ that predicts either -1 or +1.
We must learn function f that optimises some objective (e.g. number of misclassifications)
- A train dataset $\{x,y\}$ where $y \in \{-1,1\}$ is provided to learn the function f .
 - supervised learning

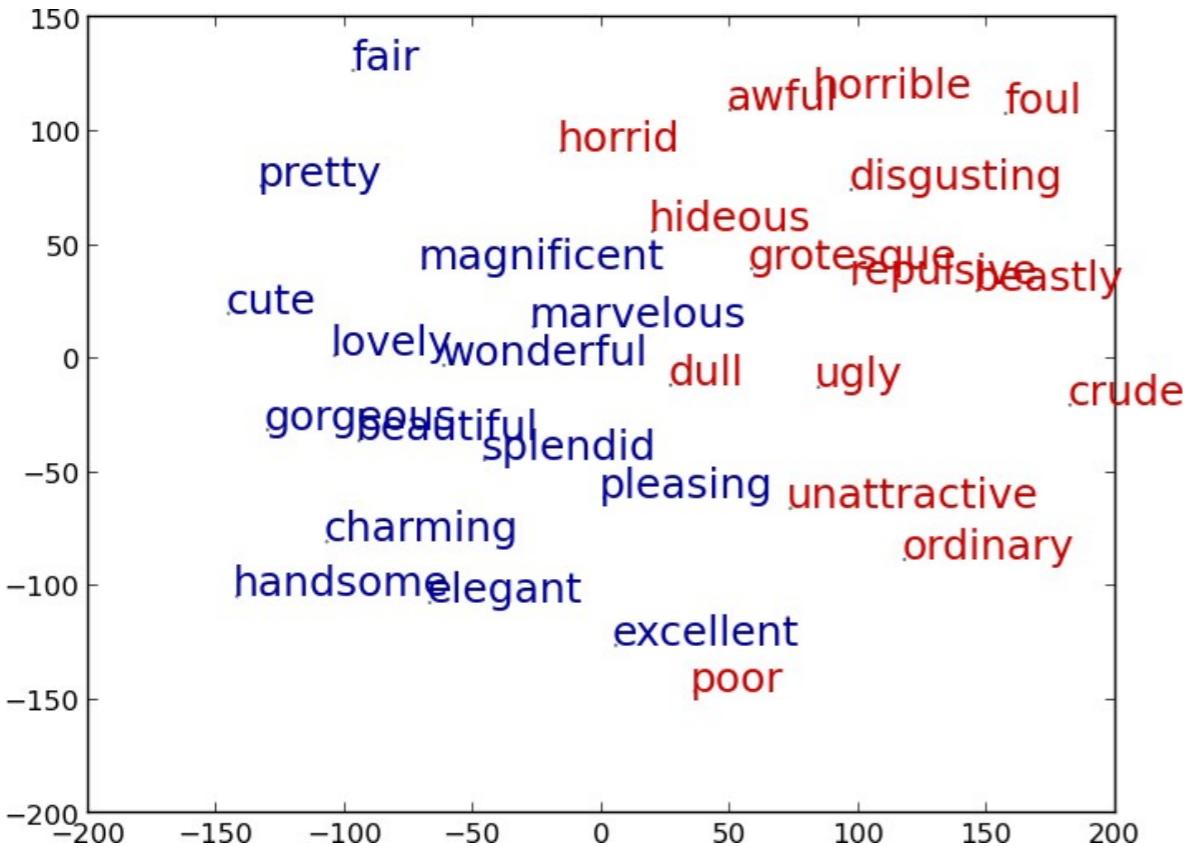
Clustering

- Given a dataset $\{x_1, x_2, \dots, x_n\}$ group the data points into k groups such that data points within the same group have some common attributes/similarities.
- Why we need clusters (groups)
 - If the dataset is large, we can select some representative samples from each cluster
 - Summarise the data, visualise the data

Cluster Visualization



Word Clusters



Yogatama+I4

words that express similar sentiments are grouped into the same cluster

Mathematical Preliminaries

COMP337 + 527 Data Mining and Visualisation

Linear Algebra

- In Data Mining, we will represent data points using a set of coordinates (corresponding to various attributes/features). This mathematical representation is compact and powerful enough to describe parallel processing methods.
- The branch of mathematics that concerns with such coordinated representations is called **linear algebra**
- Reference: Chapter 02 of the MML book
[<https://mml-book.github.io/book/chapter02.pdf>]

Vectors

- We will denote a vector \mathbf{x} in the n-dimensional real space by (lowercase bold fonts) $\mathbf{x} \in \mathbb{R}^n$
- We will use column vectors throughout this module (transposed by T when written as row vectors)
- e.g. $\mathbf{x} = (3.2, -9.1, 0.1)^T$
- A function can be seen as an infinite dimensional vector, where all function values are arranged as elements in the vector!

Matrices

- We obtain matrices by arranging a collection of vectors by columns or rows.
- We use uppercase bold fonts to denote matrices such as $\mathbf{M} \in \mathbb{R}^{n \times m}$
- When $n = m$ we say \mathbf{M} is square
- We denote the (i,j) element of \mathbf{M} by $M_{i,j}$
- If $M_{i,j} = M_{j,i}$ for all i and j , we say \mathbf{M} is symmetric.
Otherwise, \mathbf{M} is asymmetric
- If all elements in \mathbf{M} are real numbers, then we call \mathbf{M} to be a real matrix, otherwise a complex matrix

Vector arithmetic

- Given two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$ their *addition* is given by the vector $\mathbf{z} \in \mathbb{R}^N$ where i -th element z_i is given by $z_i = x_i + y_i$
- Their element-wise product (Hadamard product \otimes) is given by $z_i = x_i y_i$
- Their inner-product (dot product) is defined as

$$\mathbf{x}^\top \mathbf{y} = \sum_{i=1}^N x_i y_i$$

- Their outer-product ($\mathbf{x}\mathbf{y}^\top$) is defined as the matrix $\mathbf{M} \in \mathbb{R}^{N \times N}$ where $M_{i,j} = x_i y_j$

Quiz

- Given $\mathbf{x} = (1, 2, 3)^\top$ and $\mathbf{y} = (3, 2, 1)^\top$
 - Find $\mathbf{x} + \mathbf{y}$
 - Find $\mathbf{x} \otimes \mathbf{y}$
 - Find $\mathbf{x}^\top \mathbf{y}$
 - Find $\mathbf{x} \mathbf{y}^\top$

Matrix arithmetic

- Matrices of the same shape (number of rows and columns) can be added elementwise
 - $\mathbf{A} + \mathbf{B} = \mathbf{C}$ where $C_{i,j} = A_{i,j} + B_{i,j}$
- Matrices can be multiplied if the number of columns of the first matrix is equal to the number of rows of the second matrix

$$\mathbf{A} \in \mathbb{R}^{n \times m}, \mathbf{B} \in \mathbb{R}^{m \times p}$$

- $\mathbf{AB} = \mathbf{C}$ where $C_{i,j} = \sum_{k=1}^m A_{i,k}B_{k,j}$

Quiz

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, \mathbf{B} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 2 & 3 \\ -1 & 0 & 1 \end{pmatrix}$$

- Compute $\mathbf{A} + \mathbf{B}$
- Compute $\mathbf{B} + \mathbf{A}$
- Compute \mathbf{AB}
- Compute \mathbf{BA}
- Is matrix product commutative in general?

Transpose and Inverse

- The transpose of a matrix \mathbf{A} is denoted by \mathbf{A}^T and the (i,j) element of the transpose is $A_{j,i}$
 - $(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$
- The inverse of a square matrix \mathbf{A} is denoted by \mathbf{A}^{-1} and satisfies $\mathbf{AA}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$
 - Here, $\mathbf{I} \in \mathbb{R}^{n \times n}$ is the unit matrix (all diagonal elements are set to 1 and non-diagonal elements are set to 0)

Computing the inverse of a 2x2 matrix

- Compute the inverse of the following matrix

$$A = \begin{pmatrix} 1 & 2 \\ -2 & 1 \end{pmatrix}$$

Determinant of a matrix

- Determinant of a matrix **A** is denoted by $|A|$
- For a 2×2 matrix **A** its determinant is given by

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, |A| = ad - bc$$

Quiz: Matrix inversion

- Write the generalised form for the inverse of a 2x2 matrix using the matrix determinant.

Linear independence

- Let us consider a vector \mathbf{v} formed as the linearly-weighted sum of a set of vectors $\{\mathbf{x}_1, \dots, \mathbf{x}_K\}$ with respective coefficients $\lambda_1, \dots, \lambda_K$ as follows:

$$\mathbf{v} = \lambda_1 \mathbf{x}_1 + \dots + \lambda_K \mathbf{x}_K = \sum_{i=1}^K \lambda_i \mathbf{x}_i$$

- \mathbf{v} is called a linear combination of $\{\mathbf{x}_1, \dots, \mathbf{x}_K\}$
- The null vector $\mathbf{0}$ can always be represented as a linear combination of K vectors (Quiz: show this)
- We are interested in cases where we can represent a vector as the linear combination of non-zero coefficients.

Quiz: Linear independence

- Show that \mathbf{v} cannot be expressed as a linear combination of \mathbf{a} and \mathbf{b} , where

$$\mathbf{v} = (1, 2, -3, 4)^\top$$

$$\mathbf{a} = (1, 1, 0, 2)^\top$$

$$\mathbf{b} = (-1, -2, 1, 1)^\top$$

Rank

- The number of linear independent columns of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ ($m \leq n$) equals the number of linearly independent rows and is called the **rank** of \mathbf{A} is denoted by $\text{rank}(\mathbf{A})$
- $\text{rank}(\mathbf{A}) \leq \min(m, n) = m$
- If $\text{rank}(\mathbf{A}) = m$, then \mathbf{A} is said to be full-rank, otherwise rank deficit.
- Only full-rank square matrices are invertible.

Quiz:

- Find the ranks of the following matrices:

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \mathbf{B} = \begin{pmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{pmatrix}$$

Matrix trace

- The sum of diagonal elements is called the trace of the matrix. Specifically,

$$\text{tr}(\mathbf{A}) = \sum_i A_{i,i}$$

- Find $\text{tr}(\mathbf{A})$ for

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

Eigendecomposition

- Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a square matrix. Then $\lambda \in \mathbb{R}$ is an **eigenvalue** of \mathbf{A} and a nonzero $\mathbf{x} \in \mathbb{R}^n$ is the corresponding **eigenvector** of \mathbf{A} if
$$\mathbf{Ax} = \lambda\mathbf{x}$$
- We call this the *eigenvalue equation*
- an n -dimensional square matrix has exactly n eigenvectors and we can express \mathbf{A} using its eigenvectors as follows. This called the **eigendecomposition** of \mathbf{A} .

$$\mathbf{A} = \sum_{i=1}^n \lambda_i \mathbf{x}_i \mathbf{x}_i^\top$$

Quiz:

- Find the eigenvalues and the corresponding eigenvectors of \mathbf{A}

$$\mathbf{A} = \begin{pmatrix} 4 & 2 \\ 1 & 3 \end{pmatrix}$$

Vector Calculus

- This is also known as multivariate calculus, where we have functions of multiple variables (such as the dimensions in a vector) and we must compute partial or total derivatives w.r.t. the variables.
- All what you know from A/L calculus is still valid and can be used to derive the rules in vector calculus starting from the first principles.

Differentiation Rules

Product Rule: $(f(x)g(x))' = f'(x)g(x) + f(x)g'(x)$

Quotient Rule: $\left(\frac{f(x)}{g(x)}\right)' = \frac{f'(x)g(x) - f(x)g'(x)}{(g(x))^2}$

Sum Rule: $(f(x) + g(x))' = f'(x) + g'(x)$

Chain Rule: $(g(f(x)))' = (g \circ f)'(x) = g'(f(x))f'(x)$

Quiz: Using the chain rule compute the derivative of the function
 $h(x) = (2x + 1)^4$

Partial derivative

Definition 5.5 (Partial Derivative). For a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $\mathbf{x} \mapsto f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^n$ of n variables x_1, \dots, x_n we define the *partial derivatives* as

$$\begin{aligned}\frac{\partial f}{\partial x_1} &= \lim_{h \rightarrow 0} \frac{f(x_1 + h, x_2, \dots, x_n) - f(\mathbf{x})}{h} \\ &\vdots \\ \frac{\partial f}{\partial x_n} &= \lim_{h \rightarrow 0} \frac{f(x_1, \dots, x_{n-1}, x_n + h) - f(\mathbf{x})}{h}\end{aligned}\tag{5.39}$$

and collect them in the row vector

$$\nabla_{\mathbf{x}} f = \text{grad } f = \frac{df}{d\mathbf{x}} = \left[\frac{\partial f(\mathbf{x})}{\partial x_1} \quad \frac{\partial f(\mathbf{x})}{\partial x_2} \quad \dots \quad \frac{\partial f(\mathbf{x})}{\partial x_n} \right] \in \mathbb{R}^{1 \times n}, \tag{5.40}$$

where n is the number of variables and 1 is the dimension of the image/range/co-domain of f . Here, we defined the column vector $\mathbf{x} = [x_1, \dots, x_n]^T \in \mathbb{R}^n$. The row vector in (5.40) is called the *gradient* of f or the *Jacobian* and is the generalization of the derivative from Section 5.1.

Quiz: For $f(x,y) = (x+2y^3)^2$ compute $\partial f / \partial x$ and $\partial f / \partial y$

Chain rule for multivariate functions

Consider a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ of two variables x_1, x_2 . Furthermore, $x_1(t)$ and $x_2(t)$ are themselves functions of t . To compute the gradient of f with respect to t , we need to apply the chain rule (5.48) for multivariate functions as

$$\frac{df}{dt} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial x_1(t)}{\partial t} \\ \frac{\partial x_2(t)}{\partial t} \end{bmatrix} = \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial t} \quad (5.49)$$

where d denotes the gradient and ∂ partial derivatives.

Quiz: Consider $f(x_1, x_2) = x_1^2 + 2x_2$, where $x_1 = \sin(t)$ and $x_2 = \cos(t)$.
Find, df/dt .

Useful identities for computing gradients

$$\frac{\partial}{\partial \mathbf{X}} \mathbf{f}(\mathbf{X})^\top = \left(\frac{\partial \mathbf{f}(\mathbf{X})}{\partial \mathbf{X}} \right)^\top \quad (5.99)$$

$$\frac{\partial}{\partial \mathbf{X}} \text{tr}(\mathbf{f}(\mathbf{X})) = \text{tr} \left(\frac{\partial \mathbf{f}(\mathbf{X})}{\partial \mathbf{X}} \right) \quad (5.100)$$

$$\frac{\partial}{\partial \mathbf{X}} \det(\mathbf{f}(\mathbf{X})) = \det(\mathbf{f}(\mathbf{X})) \text{tr} \left(\mathbf{f}(\mathbf{X})^{-1} \frac{\partial \mathbf{f}(\mathbf{X})}{\partial \mathbf{X}} \right) \quad (5.101)$$

$$\frac{\partial}{\partial \mathbf{X}} \mathbf{f}(\mathbf{X})^{-1} = -\mathbf{f}(\mathbf{X})^{-1} \frac{\partial \mathbf{f}(\mathbf{X})}{\partial \mathbf{X}} \mathbf{f}(\mathbf{X})^{-1} \quad (5.102)$$

$$\frac{\partial \mathbf{a}^\top \mathbf{X}^{-1} \mathbf{b}}{\partial \mathbf{X}} = -(\mathbf{X}^{-1})^\top \mathbf{a} \mathbf{b}^\top (\mathbf{X}^{-1})^\top \quad (5.103)$$

$$\frac{\partial \mathbf{x}^\top \mathbf{a}}{\partial \mathbf{x}} = \mathbf{a}^\top \quad (5.104)$$

$$\frac{\partial \mathbf{a}^\top \mathbf{x}}{\partial \mathbf{x}} = \mathbf{a}^\top \quad (5.105)$$

$$\frac{\partial \mathbf{a}^\top \mathbf{X} \mathbf{b}}{\partial \mathbf{X}} = \mathbf{a} \mathbf{b}^\top \quad (5.106)$$

$$\frac{\partial \mathbf{x}^\top \mathbf{B} \mathbf{x}}{\partial \mathbf{x}} = \mathbf{x}^\top (\mathbf{B} + \mathbf{B}^\top) \quad (5.107)$$

$$\frac{\partial}{\partial \mathbf{s}} (\mathbf{x} - \mathbf{A}\mathbf{s})^\top \mathbf{W} (\mathbf{x} - \mathbf{A}\mathbf{s}) = -2(\mathbf{x} - \mathbf{A}\mathbf{s})^\top \mathbf{W} \mathbf{A} \quad \text{for symmetric } \mathbf{W}$$
$$(5.108)$$

Note: You do not need to memorise these but must be able to verify these by yourself.



UNIVERSITY OF
LIVERPOOL

RESIT EXAMINATIONS 2017/18

Data Mining and Visualisation

TIME ALLOWED : Two and a Half Hours

INSTRUCTIONS TO CANDIDATES

Answer **FOUR** questions.

If you attempt to answer more questions than the required number of questions, the marks awarded for the excess questions answered will be discarded (starting with your lowest mark).

Question 1 Consider a dataset \mathcal{D} of N instances, where each instance $x_i \in \mathcal{D}$ is represented by a three dimensional real-valued vector $\mathbf{x}_i = (x_{i1}, x_{i2}, x_{i3})^\top$. Moreover, a label $t_i \in \{-1, 1\}$ is assigned to \mathbf{x}_i . We would like to learn a binary classifier using \mathcal{D} . However, for some instances, we do not have x_{i3} values measured. Answer the following questions.

- A.** Explain what is meant by the *missing value problem* in data mining. **(2 marks)**

If some features are missing (not measured, unobserved) for some data points in a dataset, then this is called the missing value problem.

- B.** Compute the ℓ_2 norm of \mathbf{x}_i . **(2 marks)**

$$\|\mathbf{x}_i\|_2 = \sqrt{x_{i1}^2 + x_{i2}^2 + x_{i3}^2}$$

- C.** Write the ℓ_2 normalised version of \mathbf{x}_i . **(2 marks)**

$$\frac{\mathbf{x}_i}{\|\mathbf{x}_i\|_2}$$

- D.** Compute the means μ_1, μ_2, μ_3 and standard deviations $\sigma_1, \sigma_2, \sigma_3$ for the three features in \mathcal{D} . **(6 marks)**

$$\begin{aligned}\mu_1 &= \frac{1}{N} \sum_{n=1}^N x_{n1} \\ \sigma_1 &= \sqrt{\frac{\sum_{n=1}^N (x_{n1} - \mu_1)^2}{N-1}} \\ \mu_2 &= \frac{1}{N} \sum_{n=1}^N x_{n2} \\ \sigma_2 &= \sqrt{\frac{\sum_{n=1}^N (x_{n2} - \mu_2)^2}{N-1}} \\ \mu_3 &= \frac{1}{N} \sum_{n=1}^N x_{n3} \\ \sigma_3 &= \sqrt{\frac{\sum_{n=1}^N (x_{n3} - \mu_3)^2}{N-1}}\end{aligned}$$

- E.** Apply Gaussian scaling on \mathbf{x}_i . **(2 marks)**

$$\left(\frac{x_{i1} - \mu_1}{\sigma_1}, \frac{x_{i2} - \mu_2}{\sigma_2}, \frac{x_{i3} - \mu_3}{\sigma_3} \right)$$

- F.** Given that $\mu_3 = 0$ would it be problematic to replace missing values of x_{i3} to zero? Explain your answer. **(2 marks)**

If we replace missing x_{i3} values by zero we would not be able to distinguish among the instances for which x_{i3} was measured but turned out to be zero vs. instances where x_{i3} is missing.

- G.** As a solution to the missing value problem, we would like to predict x_{i3} using x_{i1} and x_{i2} using the linear relationship $\hat{x}_{i3} = ax_{i1} + bx_{i2} + c$, where $a, b, c \in \mathbb{R}$ are parameters that must be estimated from \mathcal{D} and \hat{x}_{i3} is the predicted value for x_{i3} . Write the squared loss for this prediction problem. **(3 marks)**

$$E(\mathcal{D}) = \sum_{i=1}^N (ax_{i1} + bx_{i2} + c - x_{i3})^2$$

- H.** Compute the gradient of the squared loss function w.r.t. a, b and c . **(3 marks)**

$$\frac{\partial E}{\partial a} = 2 \sum_{i=1}^N (ax_{i1} + bx_{i2} + c)x_{i1}$$

$$\frac{\partial E}{\partial b} = 2 \sum_{i=1}^N (ax_{i1} + bx_{i2} + c)x_{i2}$$

$$\frac{\partial E}{\partial c} = 2 \sum_{i=1}^N (ax_{i1} + bx_{i2} + c)$$

- I.** Write the update rules for a, b and c using stochastic gradient descent. **(3 marks)**

$$a^{(k+1)} = a^{(k)} - 2\eta \sum_{i=1}^N (ax_{i1} + bx_{i2} + c)x_{i1}$$

$$b^{(k+1)} = b^{(k)} - 2\eta \sum_{i=1}^N (ax_{i1} + bx_{i2} + c)x_{i2}$$

$$c^{(k+1)} = c^{(k)} - 2\eta \sum_{i=1}^N (ax_{i1} + bx_{i2} + c)$$

Question 2 We would like to use the Perceptron algorithm to learn a linear classifier $y = \mathbf{w}^\top \mathbf{x} + b$, defined by a weight vector $\mathbf{w} \in \mathbb{R}^d$ and a bias $b \in \mathbb{R}$ from a training dataset consisting of three instances, $\{(t_n, \mathbf{x}_n)\}_{n=1}^3$. Here, $\mathbf{x}_1 = (0, 0)^\top$, $\mathbf{x}_2 = (1, 1)^\top$ and $\mathbf{x}_3 = (-1, 1)^\top$, and the labels are $t_1 = 1$, $t_2 = -1$ and $t_3 = 1$. We predict an instance \mathbf{x} as positive if $\mathbf{w}^\top \mathbf{x} + b \geq 0$, and negative otherwise. The initial values of the weight vector and the bias are set respectively to $\mathbf{w}^{(0)} = (0, 0)^\top$ and $b = 0$. We visit the training instances in the order $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$. Answer the following questions.

- A. Plot the dataset in the two-dimensional space. (2 marks)

The three points form a triangle with x_1 at the origin and x_2 and x_3 mirroring each other on the y-axis.

- B. Write the perceptron update rule for a misclassified instance (t, \mathbf{x}) . (3 marks)

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + t\mathbf{x}$$

- C. What will be the values of the weight vector and the bias after observing the instance \mathbf{x}_1 . (3 marks)

$$y_1 = \mathbf{w}^{(0)^\top} \mathbf{x}_1 + b^{(0)} = 0. \text{ Therefore, this instance is classified correctly as positive.}$$

The weight vector and bias are not updated. $\mathbf{w}^{(1)} = (0, 0)^\top, b^{(1)} = 0$.

- D. What will be values of the weight vector and the bias after observing \mathbf{x}_2 . (4 marks)

$y_2 = 0$. *Therefore, x_2 will be incorrectly classified as positive. The weight vector and the bias will be update to $\mathbf{w}^{(2)} = (-1, -1)^\top, b^{(2)} = -1$*

- E. What will be the values of the weight vector and the bias after observing \mathbf{x}_3 . (4 marks)

$y_3 = (-1, 1)^\top (-1, -1)^\top - 1 = -1 < 0$. *Therefore, x_3 will be incorrectly classified as negative. The updated values will be $\mathbf{w}^{(3)} = (-2, 2)^\top, b^{(3)} = 0$.*

- F. Is the dataset consisting of x_1, x_2, x_3 linearly separable?. Justify your answer. (2 marks)

Yes. We have already found a Perceptron with a weight vector and a bias that would correctly classify all three instances in this dataset.

- G. Is it the case that a dataset consisting of three points is always linearly separable? If yes, explain your answer. If no, provide a counter example. (4 marks)

No. For example, if the three datapoints are on a straight line and the middle point has the opposite label than the other two points, then this dataset cannot be linearly separable.

- H. Explain a method that you can use to learn a Perceptron from a non-linearly separable dataset. (3 marks)

Apply a kernel method to project the dataset into a high dimensional feature space and learn a Perceptron in this high dimensional feature space.

Question 3 Consider the two sentences S_1 and S_2 given by:

$$S_1 = \text{I love cake with tea}$$

$$S_2 = \text{I drink beer with cake}$$

Answer the following questions.

- A.** Represent S_1 and S_2 respectively by feature vectors \mathbf{s}_1 and \mathbf{s}_2 , where elements correspond to the frequency of unigrams. (4 marks)

Let the unigram features be indexed as follows: I=0, love=1, cake=2, with=3, tea=4, drink=5, beer=6. Then we have $\mathbf{s}_1 = (1, 1, 1, 1, 1, 0, 0)^\top$ and $\mathbf{s}_2 = (1, 0, 1, 1, 0, 1, 1)^\top$.

- B.** Compute the ℓ_2 norms of \mathbf{s}_1 and \mathbf{s}_2 . (4 marks)

$$\|\mathbf{s}_1\|_2 = \sqrt{5}, \|\mathbf{s}_2\|_2 = \sqrt{5}$$

- C.** Compute the ℓ_1 norms of \mathbf{s}_1 and \mathbf{s}_2 . (4 marks)

$$\|\mathbf{s}_1\|_1 = 5, \|\mathbf{s}_2\|_1 = 5$$

- D.** Compute the cosine similarity between \mathbf{s}_1 and \mathbf{s}_2 . (2 marks)

$$\frac{\mathbf{s}_1^\top \mathbf{s}_2}{\|\mathbf{s}_1\|_2 \|\mathbf{s}_2\|_2} = 3/5$$

- E.** Compute the Manhattan distance between \mathbf{s}_1 and \mathbf{s}_2 . (2 marks)

$$|1 - 1| + |1 - 0| + |1 - 1| + |1 - 1| + |1 - 0| + |0 - 1| + |0 - 1| = 4$$

- F.** Assume that for all the unigrams u_i and bigrams $u_i u_{i+1}$ that appear in S_1 and S_2 we are given the marginal probabilities respectively $p(u_i)$ and $p(u_i u_{i+1})$. Compute the conditional probability of observing u_{i+1} given u_i . (2 marks)

$$\begin{aligned} p(u_{i+1}|u_i) &= \frac{p(u_{i+1}, u_i)}{p(u_i)} \\ p(u_{i+1}, u_i) &= p(u_{i+1}u_i) + p(u_iu_{i+1}) \\ p(u_{i+1}|u_i) &= \frac{p(u_{i+1}u_i) + p(u_iu_{i+1})}{p(u_i)} \end{aligned}$$

- G.** Using the Markov assumption, compute the likelihood $p(S_1)$ and $p(S_2)$. (4 marks)

$$p(S_1) = p(I)p(\text{love}|I)p(\text{cake}|\text{love})p(\text{with}|\text{love})p(\text{tea}|\text{with})$$

$$p(S_2) = p(I)p(\text{drink}|I)p(\text{beer}|\text{drink})p(\text{with}|\text{drink})p(\text{tea}|\text{with})$$

- H.** Explain how you can use the computation done in part (F) to evaluate whether S_2 is less common than S_1 in English texts written by native speakers. (3 marks)

Compare $p(S_1)$ and $p(S_2)$. If the likelihood of a sentence is small, then it is unlikely to be produced by a native speaker.

Question 4 Table 1 shows how four users u_1, u_2, u_3, u_4 purchased four items I_1, I_2, I_3, I_4 in an online shopping site over a period of one year. A cell value of 1 indicates that the user corresponding to the row has purchased the item corresponding to the column, and 0 otherwise. Answer the following questions.

	I_1	I_2	I_3	I_4
u_1	1	0	1	1
u_2	1	1	0	0
u_3	0	0	1	1
u_4	0	1	0	0

Table 1: A table showing four users u_1, u_2, u_3, u_4 who have purchased four items I_1, I_2, I_3, I_4 in an online shopping site over a period of one year.

- A.** Given that the users have been initially clustered into two clusters $S_1 = \{u_1, u_2\}$ and $S_2 = \{u_3, u_4\}$, compute the centroids for the two clusters respectively denoted by μ_1 and μ_2 . For this purpose, consider a user is represented by a vector over the items he or she has purchased in the past. **(2 marks)**

$$\mu_1 = (1, 0.5, 0.5, 0.5)^\top \text{ and } \mu_2 = (0, 0.5, 0.5, 0.5)^\top$$

- B.** Compute Euclidean distances between μ_1 and each of the four users. **(4 marks)**

$$d(\mathbf{u}_1, \mu_1) = \sqrt{0.75}$$

$$d(\mathbf{u}_2, \mu_1) = \sqrt{0.75}$$

$$d(\mathbf{u}_3, \mu_1) = \sqrt{1.75}$$

$$d(\mathbf{u}_4, \mu_1) = \sqrt{0.75}$$

- C.** Compute Euclidean distances between μ_2 and each of the four users. **(4 marks)**

$$d(\mathbf{u}_1, \mu_2) = \sqrt{1.75}$$

$$d(\mathbf{u}_2, \mu_2) = \sqrt{1.75}$$

$$d(\mathbf{u}_3, \mu_2) = \sqrt{0.75}$$

$$d(\mathbf{u}_4, \mu_2) = \sqrt{0.75}$$

- D.** Based on the distances computed in parts (B) and (C), determine the assignment of users to clusters for the next iteration. **(2 marks)**

Two possible assignments exist. $S_1 = \{u_1, u_2, u_4\}, S_2 = \{u_3\}$ or $S_1 = \{u_1, u_2\}, S_2 = \{u_3, u_4\}$

- E.** Let us denote the probability of a user purchasing an item I_j when he or she has purchased I_i by $p(I_j|I_i)$. From Table 1, compute $p(I_1|I_4)$, $p(I_2|I_4)$ and $p(I_3|I_4)$. **(3 marks)**

$$p(I_1|I_4) = 0.5, p(I_2|I_4) = 0 \text{ and } p(I_3|I_4) = 1$$

- F.** Based on your calculations in part (E), explain what is the best item to recommend to a user who has just purchased I_4 . **(2 marks)**

I_3 because $p(I_3|I_4) = 1$ and the user is likely to buy I_3 too, given that he/she has already purchased I_4

- G.** Represent the information shown in Table 1 by a bi-partite graph where the users and items are represented by vertices, and an undirected edge is formed between the vertices corresponding to u_i and I_j if and only if u_i has purchased I_j . **(4 marks)**

- H.** Consider a random walker moving along the edges of the graph you created in part (G), where the probability of moving from u_i to I_j is given by $\frac{1}{d(u_i)}$ and the probability of moving from I_j to u_i is given by $\frac{1}{d(I_j)}$. Here, $d(x)$ is the out-degree of the vertex x . Given that the random walker started from u_1 , compute the probability that the random walker will be in u_3 after two time steps. **(4 marks)**

$$p(u_1 \rightarrow I_3)p(I_3 \rightarrow u_3) + p(u_1 \rightarrow I_4)p(I_4 \rightarrow u_3) = 1/3$$

Question 5 Consider the three points $x_1 = (0, 1)$, $x_2 = (-1, 0)$ and $x_3 = (1, 0)$. We would like to project these three points onto a straight line using principle component analysis. Answer the following questions.

- A. Compute the total projection error if we project the three points onto the y -axis. **(3 marks)**

$$1+1=2$$

- B. Compute the total projection error if we project the three points onto the x -axis. **(3 marks)**

$$1$$

- C. Compute the mean \bar{x} of the three points. **(2 marks)**

$$(0, 1/3)$$

- D. Compute the covariance matrix for the three points. **(3 marks)**

Compute $\mathbf{x}_i - \bar{\mathbf{x}}$ and adding the outer product matrices gives $[[2, 0], [0, 2/3]]$.

- E. Compute the eigenvalues of the covariance computed in part (D). **(4 marks)**

Because the covariance matrix is diagonal we have $\lambda_1 = 2, \lambda_2 = 2/3$

- F. Compute the first principle component of the projection. **(3 marks)**

The eigenvector corresponding to λ_1 (the larger eigenvalue) is $(1, 0)$. This is the x -axis.

- G. Compute the second principle component of the projection. **(3 marks)**

The eigenvector corresponding to λ_2 (the smaller eigenvalue) is $(0, 1)$. This is the y -axis.

- H. Compute the total variance if we had projected the three points on to the first principle component. **(2 marks)**

$$\frac{(1-0)^2 + (-1-0)^2 + (0-0)^2}{3} = 2/3$$

- I. Compute the total variance if we had projected the three points on to the second principle component. **(2 marks)**

$$\frac{(1-0.5)^2 + (0-0.5)^2 + (0-0.5)^2}{3} = 0.25$$

Mathematical Preliminaries

COMP337 + 527 Data Mining and Visualisation

Linear Algebra

- In Data Mining, we will represent data points using a set of coordinates (corresponding to various attributes/features). This mathematical representation is compact and powerful enough to describe parallel processing methods.
- The branch of mathematics that concerns with such coordinated representations is called **linear algebra**
- Reference: Chapter 02 of the MML book
[<https://mml-book.github.io/book/chapter02.pdf>]

Vectors

- We will denote a vector \mathbf{x} in the n-dimensional real space by (lowercase bold fonts) $\mathbf{x} \in \mathbb{R}^n$
- We will use column vectors throughout this module (transposed by T when written as row vectors)
- e.g. $\mathbf{x} = (3.2, -9.1, 0.1)^T$
- A function can be seen as an infinite dimensional vector, where all function values are arranged as elements in the vector!

Matrices

- We obtain matrices by arranging a collection of vectors by columns or rows.
- We use uppercase bold fonts to denote matrices such as $\mathbf{M} \in \mathbb{R}^{n \times m}$
- When $n = m$ we say \mathbf{M} is square
- We denote the (i,j) element of \mathbf{M} by $M_{i,j}$
- If $M_{i,j} = M_{j,i}$ for all i and j , we say \mathbf{M} is symmetric.
Otherwise, \mathbf{M} is asymmetric
- If all elements in \mathbf{M} are real numbers, then we call \mathbf{M} to be a real matrix, otherwise a complex matrix

Vector arithmetic

- Given two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$ their *addition* is given by the vector $\mathbf{z} \in \mathbb{R}^N$ where i -th element z_i is given by $z_i = x_i + y_i$
- Their element-wise product (Hadamard product \otimes) is given by $z_i = x_i y_i$
- Their inner-product (dot product) is defined as

$$\mathbf{x}^\top \mathbf{y} = \sum_{i=1}^N x_i y_i$$

- Their outer-product ($\mathbf{x}\mathbf{y}^\top$) is defined as the matrix $\mathbf{M} \in \mathbb{R}^{N \times N}$ where $M_{i,j} = x_i y_j$

Quiz

- Given $\mathbf{x} = (1, 2, 3)^\top$ and $\mathbf{y} = (3, 2, 1)^\top$
 - Find $\mathbf{x} + \mathbf{y}$
 - Find $\mathbf{x} \otimes \mathbf{y}$
 - Find $\mathbf{x}^\top \mathbf{y}$
 - Find $\mathbf{x} \mathbf{y}^\top$

Matrix arithmetic

- Matrices of the same shape (number of rows and columns) can be added elementwise
 - $\mathbf{A} + \mathbf{B} = \mathbf{C}$ where $C_{i,j} = A_{i,j} + B_{i,j}$
- Matrices can be multiplied if the number of columns of the first matrix is equal to the number of rows of the second matrix

$$\mathbf{A} \in \mathbb{R}^{n \times m}, \mathbf{B} \in \mathbb{R}^{m \times p}$$

- $\mathbf{AB} = \mathbf{C}$ where $C_{i,j} = \sum_{k=1}^m A_{i,k}B_{k,j}$

Quiz

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, \mathbf{B} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 2 & 3 \\ -1 & 0 & 1 \end{pmatrix}$$

- Compute $\mathbf{A} + \mathbf{B}$
- Compute $\mathbf{B} + \mathbf{A}$
- Compute \mathbf{AB}
- Compute \mathbf{BA}
- Is matrix product commutative in general?

Transpose and Inverse

- The transpose of a matrix \mathbf{A} is denoted by \mathbf{A}^T and the (i,j) element of the transpose is $A_{j,i}$
 - $(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$
- The inverse of a square matrix \mathbf{A} is denoted by \mathbf{A}^{-1} and satisfies $\mathbf{AA}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$
 - Here, $\mathbf{I} \in \mathbb{R}^{n \times n}$ is the unit matrix (all diagonal elements are set to 1 and non-diagonal elements are set to 0)

Computing the inverse of a 2x2 matrix

- Compute the inverse of the following matrix

$$A = \begin{pmatrix} 1 & 2 \\ -2 & 1 \end{pmatrix}$$

Determinant of a matrix

- Determinant of a matrix **A** is denoted by $|A|$
- For a 2×2 matrix **A** its determinant is given by

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, |A| = ad - bc$$

Quiz: Matrix inversion

- Write the generalised form for the inverse of a 2x2 matrix using the matrix determinant.

Linear independence

- Let us consider a vector \mathbf{v} formed as the linearly-weighted sum of a set of vectors $\{\mathbf{x}_1, \dots, \mathbf{x}_K\}$ with respective coefficients $\lambda_1, \dots, \lambda_K$ as follows:

$$\mathbf{v} = \lambda_1 \mathbf{x}_1 + \dots + \lambda_K \mathbf{x}_K = \sum_{i=1}^K \lambda_i \mathbf{x}_i$$

- \mathbf{v} is called a linear combination of $\{\mathbf{x}_1, \dots, \mathbf{x}_K\}$
- The null vector $\mathbf{0}$ can always be represented as a linear combination of K vectors (Quiz: show this)
- We are interested in cases where we can represent a vector as the linear combination of non-zero coefficients.

Quiz: Linear independence

- Show that \mathbf{v} cannot be expressed as a linear combination of \mathbf{a} and \mathbf{b} , where

$$\mathbf{v} = (1, 2, -3, 4)^\top$$

$$\mathbf{a} = (1, 1, 0, 2)^\top$$

$$\mathbf{b} = (-1, -2, 1, 1)^\top$$

Rank

- The number of linear independent columns of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ ($m \leq n$) equals the number of linearly independent rows and is called the **rank** of \mathbf{A} is denoted by $\text{rank}(\mathbf{A})$
- $\text{rank}(\mathbf{A}) \leq \min(m, n) = m$
- If $\text{rank}(\mathbf{A}) = m$, then \mathbf{A} is said to be full-rank, otherwise rank deficit.
- Only full-rank square matrices are invertible.

Quiz:

- Find the ranks of the following matrices:

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \mathbf{B} = \begin{pmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{pmatrix}$$

Matrix trace

- The sum of diagonal elements is called the trace of the matrix. Specifically,

$$\text{tr}(\mathbf{A}) = \sum_i A_{i,i}$$

- Find $\text{tr}(\mathbf{A})$ for

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

Eigendecomposition

- Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a square matrix. Then $\lambda \in \mathbb{R}$ is an **eigenvalue** of \mathbf{A} and a nonzero $\mathbf{x} \in \mathbb{R}^n$ is the corresponding **eigenvector** of \mathbf{A} if
$$\mathbf{Ax} = \lambda\mathbf{x}$$
- We call this the *eigenvalue equation*
- an n -dimensional square matrix has exactly n eigenvectors and we can express \mathbf{A} using its eigenvectors as follows. This called the **eigendecomposition** of \mathbf{A} .

$$\mathbf{A} = \sum_{i=1}^n \lambda_i \mathbf{x}_i \mathbf{x}_i^\top$$

Quiz:

- Find the eigenvalues and the corresponding eigenvectors of \mathbf{A}

$$\mathbf{A} = \begin{pmatrix} 4 & 2 \\ 1 & 3 \end{pmatrix}$$

Vector Calculus

- This is also known as multivariate calculus, where we have functions of multiple variables (such as the dimensions in a vector) and we must compute partial or total derivatives w.r.t. the variables.
- All what you know from A/L calculus is still valid and can be used to derive the rules in vector calculus starting from the first principles.

Differentiation Rules

Product Rule: $(f(x)g(x))' = f'(x)g(x) + f(x)g'(x)$

Quotient Rule: $\left(\frac{f(x)}{g(x)}\right)' = \frac{f'(x)g(x) - f(x)g'(x)}{(g(x))^2}$

Sum Rule: $(f(x) + g(x))' = f'(x) + g'(x)$

Chain Rule: $(g(f(x)))' = (g \circ f)'(x) = g'(f(x))f'(x)$

Quiz: Using the chain rule compute the derivative of the function
 $h(x) = (2x + 1)^4$

Partial derivative

Definition 5.5 (Partial Derivative). For a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $\mathbf{x} \mapsto f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^n$ of n variables x_1, \dots, x_n we define the *partial derivatives* as

$$\begin{aligned}\frac{\partial f}{\partial x_1} &= \lim_{h \rightarrow 0} \frac{f(x_1 + h, x_2, \dots, x_n) - f(\mathbf{x})}{h} \\ &\vdots \\ \frac{\partial f}{\partial x_n} &= \lim_{h \rightarrow 0} \frac{f(x_1, \dots, x_{n-1}, x_n + h) - f(\mathbf{x})}{h}\end{aligned}\tag{5.39}$$

and collect them in the row vector

$$\nabla_{\mathbf{x}} f = \text{grad } f = \frac{df}{d\mathbf{x}} = \left[\frac{\partial f(\mathbf{x})}{\partial x_1} \quad \frac{\partial f(\mathbf{x})}{\partial x_2} \quad \dots \quad \frac{\partial f(\mathbf{x})}{\partial x_n} \right] \in \mathbb{R}^{1 \times n}, \tag{5.40}$$

where n is the number of variables and 1 is the dimension of the image/range/co-domain of f . Here, we defined the column vector $\mathbf{x} = [x_1, \dots, x_n]^T \in \mathbb{R}^n$. The row vector in (5.40) is called the *gradient* of f or the *Jacobian* and is the generalization of the derivative from Section 5.1.

Quiz: For $f(x,y) = (x+2y^3)^2$ compute $\partial f / \partial x$ and $\partial f / \partial y$

Chain rule for multivariate functions

Consider a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ of two variables x_1, x_2 . Furthermore, $x_1(t)$ and $x_2(t)$ are themselves functions of t . To compute the gradient of f with respect to t , we need to apply the chain rule (5.48) for multivariate functions as

$$\frac{df}{dt} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial x_1(t)}{\partial t} \\ \frac{\partial x_2(t)}{\partial t} \end{bmatrix} = \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial t} \quad (5.49)$$

where d denotes the gradient and ∂ partial derivatives.

Quiz: Consider $f(x_1, x_2) = x_1^2 + 2x_2$, where $x_1 = \sin(t)$ and $x_2 = \cos(t)$.
Find, df/dt .

Useful identities for computing gradients

$$\frac{\partial}{\partial \mathbf{X}} \mathbf{f}(\mathbf{X})^\top = \left(\frac{\partial \mathbf{f}(\mathbf{X})}{\partial \mathbf{X}} \right)^\top \quad (5.99)$$

$$\frac{\partial}{\partial \mathbf{X}} \text{tr}(\mathbf{f}(\mathbf{X})) = \text{tr} \left(\frac{\partial \mathbf{f}(\mathbf{X})}{\partial \mathbf{X}} \right) \quad (5.100)$$

$$\frac{\partial}{\partial \mathbf{X}} \det(\mathbf{f}(\mathbf{X})) = \det(\mathbf{f}(\mathbf{X})) \text{tr} \left(\mathbf{f}(\mathbf{X})^{-1} \frac{\partial \mathbf{f}(\mathbf{X})}{\partial \mathbf{X}} \right) \quad (5.101)$$

$$\frac{\partial}{\partial \mathbf{X}} \mathbf{f}(\mathbf{X})^{-1} = -\mathbf{f}(\mathbf{X})^{-1} \frac{\partial \mathbf{f}(\mathbf{X})}{\partial \mathbf{X}} \mathbf{f}(\mathbf{X})^{-1} \quad (5.102)$$

$$\frac{\partial \mathbf{a}^\top \mathbf{X}^{-1} \mathbf{b}}{\partial \mathbf{X}} = -(\mathbf{X}^{-1})^\top \mathbf{a} \mathbf{b}^\top (\mathbf{X}^{-1})^\top \quad (5.103)$$

$$\frac{\partial \mathbf{x}^\top \mathbf{a}}{\partial \mathbf{x}} = \mathbf{a}^\top \quad (5.104)$$

$$\frac{\partial \mathbf{a}^\top \mathbf{x}}{\partial \mathbf{x}} = \mathbf{a}^\top \quad (5.105)$$

$$\frac{\partial \mathbf{a}^\top \mathbf{X} \mathbf{b}}{\partial \mathbf{X}} = \mathbf{a} \mathbf{b}^\top \quad (5.106)$$

$$\frac{\partial \mathbf{x}^\top \mathbf{B} \mathbf{x}}{\partial \mathbf{x}} = \mathbf{x}^\top (\mathbf{B} + \mathbf{B}^\top) \quad (5.107)$$

$$\frac{\partial}{\partial \mathbf{s}} (\mathbf{x} - \mathbf{A}\mathbf{s})^\top \mathbf{W} (\mathbf{x} - \mathbf{A}\mathbf{s}) = -2(\mathbf{x} - \mathbf{A}\mathbf{s})^\top \mathbf{W} \mathbf{A} \quad \text{for symmetric } \mathbf{W}$$
$$(5.108)$$

Note: You do not need to memorise these but must be able to verify these by yourself.



UNIVERSITY OF
LIVERPOOL

Second Semester Examinations 2016/17

Data Mining and Visualisation

TIME ALLOWED : Two and a Half Hours

INSTRUCTIONS TO CANDIDATES

Answer FOUR questions.

If you attempt to answer more questions than the required number of questions (in any section), the marks awarded for the excess questions answered will be discarded (starting with your lowest mark).

Question 1 Consider two sentences $S_1 = I \text{ like data mining}$ and $S_2 = I \text{ do not like data science}$. Answer the following questions about S_1 and S_2

- A. Write all unigrams in S_1 . **(2 marks)**

I, like, data, mining

- B. Write all bigrams in S_2 . **(2 marks)**

I+do, do+not, not+like, like+data, data+science

- C. What is meant by *stop words* in text mining? **(2 marks)**

Stop words are non-content features such as prepositions and articles. For example, *the, an, what, etc.*

- D. Assuming unigrams to be the feature space, represent S_1 and S_2 respectively by two vectors s_1 and s_2 , where the elements corresponds to the number times the corresponding unigram feature occurs in the sentence. **(4 marks)**

Let us assign indexes to the features as follows: I=0, like=1, data=2, mining=3, not=4, science=5, do=6. Then $s_1 = (1, 1, 1, 1, 0, 0, 0)^\top$ and $s_2 = (1, 1, 1, 0, 1, 1, 1)^\top$.

- E. Compute the inner-product between s_1 and s_2 . **(3 marks)**

$$s_1^\top s_2 = 3$$

- F. Compute the ℓ_2 norms of s_1 and s_2 . **(2 marks)**

$$\|s_1\|_2 = 2, \|s_2\|_2 = \sqrt{6}$$

- G. Compute the cosine similarity between the two sentences S_1 and S_2 . **(2 marks)**

$$\cos(S_1, S_2) = \frac{s_1^\top s_2}{\|s_1\|_2 \|s_2\|_2} = 3/2\sqrt{6}$$

- H. Compute the Manhattan distance between s_1 and s_2 . **(2 marks)**

$$|1 - 1| + |1 - 1| + |1 - 1| + |1 - 0| + |0 - 1| + |0 - 1| + |0 - 1| = 4$$

- I. Compute the Euclidean distance between s_1 and s_2 . **(2 marks)**

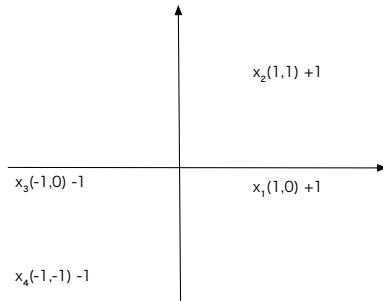
$$\sqrt{(1 - 1)^2 + (1 - 1)^2 + (1 - 1)^2 + (1 - 0)^2 + (0 - 1)^2 + (0 - 1)^2 + (0 - 1)^2} = 2$$

- J. Despite the two sentences are expressing opposite opinions, the cosine similarity measured in **G** gives a value greater than 0.5 indicating a high-degree of similarity. Suggest a solution to overcome this problem. **(4 marks)**

The issue here is that the negation indicator *not* matching (or not matching) between the sentences has equal contribution towards the cosine similarity as with any other feature. We can emphasise negation by assigning higher weights (larger than 1) to negation related features such as *not* to overcome this problem.

Question 2 Consider a training dataset $\{(\mathbf{x}_n, t_n)\}_{n=1}^4$, where $\mathbf{x}_n \in \mathbb{R}^2$ and $t_n \in \{-1, 1\}$. Here, $\mathbf{x}_1 = (1, 0)^\top$, $\mathbf{x}_2 = (1, 1)^\top$, $\mathbf{x}_3 = (-1, 0)^\top$, and $\mathbf{x}_4 = (-1, -1)^\top$. Moreover, the labels $t_1 = t_2 = 1$ and $t_3 = t_4 = -1$. Let us consider a support vector machine defined by a weight vector $\mathbf{w} = (\alpha, \beta)^\top$ and a bias term b . Here, $\alpha, \beta, b \in \mathbb{R}$.

- A. Plot the training dataset in two dimensional space. **(2 marks)**



- B. Explain what is meant by the inequality $t_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1$ with respect to each training data point. **(2 marks)**

This inequality means that each train data point must be classified correctly by the decision hyperplane \mathbf{w} with a margin of at least 1.

- C. Write down the four inequalities that must be satisfied by the four data points in the training dataset if they are to be correctly classified. **(4 marks)**

$$\begin{aligned}\alpha + b &\geq 1 \\ \alpha + \beta + b &\geq 1 \\ \alpha - b &\geq 1 \\ \alpha + \beta - b &\geq 1\end{aligned}$$

One mark is awarded for each inequality.

- D. Show that the maximisation of the margin corresponds to the minimisation of $\alpha^2 + \beta^2$. **(4 marks)**

Maximisation of the margin corresponds to the minimisation of the ℓ_2 norm of the vector \mathbf{w} , which is $\alpha^2 + \beta^2$.

- E. Using Lagrange multipliers for the four inequalities, write the objective function $L(\alpha, \beta, b, \boldsymbol{\lambda})$ where $\boldsymbol{\lambda}$ is a vector containing the four Lagrange multipliers given by $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3, \lambda_4)^\top$ each respectively for the inequalities corresponding to the four data points x_1, x_2, x_3, x_4 . **(2 marks)**

$$L(\alpha, \beta, b, \boldsymbol{\lambda}) = \frac{1}{2}(\alpha^2 + \beta^2) - \lambda_1(\alpha + b - 1) - \lambda_2(\alpha + \beta + b - 1) - \lambda_3(\alpha - b - 1) - \lambda_4(\alpha + \beta - b - 1)$$

F. Find the values of α, β, b by minimising the objective function defined in E. **(7 marks)**

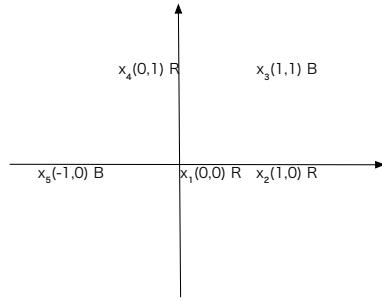
By setting the gradients of L w.r.t. α, β, b to zero, we obtain three equations and w.r.t. each Lagrangian multiplier another four equations. By solving these equations we obtain $\alpha = 1, \beta = 0, b = 0, \lambda_1 = \lambda_3 = 1/2, \lambda_2 = \lambda_4 = 0$. One mark is awarded for each variable.

G. Show that the decision hyperplane you obtained in F correctly classifies the four data points in the training dataset. **(4 marks)**

The weight vector is $(1, 0)$. Therefore, we have $y_1 = y_2 = 1 > 0$ and $y_3 = y_4 = -1 < 0$. One mark is awarded for each data point.

Question 3 Consider five data points in \mathbb{R}^2 given by $x_1 = (0, 0)^\top$, $x_2 = (1, 0)^\top$, $x_3 = (1, 1)^\top$, $x_4 = (0, 1)^\top$, and $x_5 = (-1, 0)^\top$. Here, data points x_1 , x_2 , and x_4 are labelled with the colour red, whereas data points x_3 and x_5 are labelled with the colour blue. Answer the following questions about this dataset.

- A. Plot this dataset in two-dimensional space. (3 marks)



- B. Assuming that we used k -means clustering to cluster this dataset into two clusters, and we set the initial cluster centres at x_2 and x_3 . Compute the clusters after the first assignment. (4 marks)

$$c_1 = \{3, 4\}, c_2 = \{1, 2, 5\}$$

- C. How many further iterations does it take for the k -means algorithms to cluster for this dataset. Justify your answer. (4 marks)

0 further iterations. The k -means clustering has converged because the points 4 and 3 are the closest to c_1 and points 5, 1, and 2 are the closest to c_2 . Therefore, the cluster centroids do not get updated any further. Answers without the justification will receive only 2 marks.

- D. Using the B-cubed method compute the precision for the final two clusters obtained at convergence. (4 marks)

$P(x_4) = P(x_3) = 1/2, P(x_1) = P(x_2) = 2/3, P(x_5) = 1/3$. Therefore, the overall precision will be the summation of individual precisions divided by the total number of instances (5) in the dataset, which is 8/15.

- E. Using the B-cubed method compute the recall for the final two clusters obtained at convergence. (4 marks)

$R(x_4) = 1/3, R(x_3) = 1/2, R(x_1) = R(x_2) = 2/3, R(x_5) = 1/2$. Therefore, the overall recall will be the summation of individual precisions divided by the total number of instances (5) in the dataset, which is 8/15.

- F. Using the precision and recall values, compute the F-score for the final two clusters obtained at convergence. (3 marks)

$$F = \frac{2PR}{P+R} = 8/15$$

G. Instead of creating two clusters, we would like to obtain three clusters via k -means where we use the data points 5, 1 and 2 as the centroids of the three initial clusters. Write down the final three clusters at convergence. **(3 marks)**

{5}, {1, 4}, {2, 3}

Question 4

- A.** Consider a biased coin that returns head with probability p . Let us assume that when this coin was flipped n times, we got $k(< n)$ heads. Answer the following questions about this event.

- (a) Compute the likelihood of observing $k(< n)$ heads when this coin was flipped for n times. **(2 marks)**

$$L = p^k(1-p)^{n-k}$$

- (b) Using the maximum likelihood estimation, compute the most likely value of p for this probabilistic event. **(4 marks)**

$\frac{\partial \log(L)}{\partial p} = \frac{\partial}{\partial p} k \log(p) + (n-k) \log(1-p) = 0$. This gives, $k/p - (n-k)/(1-p) = 0$, from which we obtain $p = k/n$.

- B.** Consider the dataset shown in Table 1 consisting of five reviews x_1, x_2, x_3, x_4, x_5 defined over three integer-valued attributes a_1, a_2, a_3 corresponding to the frequency of occurrences of a particular word in the document. The positive or negative sentiments label (t) of each review are denoted respectively by +1 and -1 in Table 1. Assuming that the three attributes to be mutually independent, and the probability of a review to be given by the product of the probabilities of individual attributes raised to their occurrences in the review, answer the following questions.

Review	a_1	a_2	a_3	label (t)
x_1	1	1	0	1
x_2	2	1	0	1
x_3	1	2	3	-1
x_4	0	1	1	-1
x_5	1	0	1	-1

Table 1: A set of five reviews represented using three attributes.

- (a) Compute the marginal probabilities $p(a_1), p(a_2)$ and $p(a_3)$. **(3 marks)**

$$p(a_1) = p(a_2) = p(a_3) = 1/3$$

- (b) Compute the conditional probabilities $p(a_1|t = 1), p(a_2|t = 1)$ and $p(a_3|t = 1)$. **(3 marks)**

$$p(a_1|t = 1) = 3/5, p(a_2|t = 1) = 2/5, \text{ and } p(a_3|t = 1) = 0$$

- (c) Assuming that the prior probabilities of the sentiment labels to be equal (i.e. $p(t = 1) = p(t = -1) = 0.5$), compute $p(t = -1|x_3)$, the probability of x_3 being negative. **(4 marks)**

$$p(t = 1|x_3) = \frac{p(x_3|t=1)}{p(x_3)}$$

However, note that

$$p(x_3|t = 1) = p(a_1|t = 1)^1 \times p(a_2|t = 1)^2 \times p(a_3|t = 1)^3 = 0$$

because $p(a_3|t = 1) = 0$. Therefore, $p(t = -1|x_3) = 1 - p(t = 1|x_3) = 1$.

Review	a_1	a_2	a_3	label (t)
x_1	2	2	1	1
x_2	3	2	1	1
x_3	2	3	4	-1
x_4	1	2	2	-1
x_5	2	1	2	-1

Table 2: Smoothed counts.

- (d) Apply Laplace smoothing for the occurrences of attributes in reviews shown in Table 1. Compute $p(t = 1|x_3)$ using the smoothed counts. **(4 marks)**

Smoothed counts are shown in Table 2. $p(a_1) = p(a_2) = p(a_3) = 10/30 = 1/3$, $p(a_1|t = 1) = 5/11$, $p(a_2|t = 1) = 4/11$, $p(a_3|t = 1) = 2/11$. Therefore, we have,

$$\begin{aligned} p(t = 1|x_3) &= \frac{p(x_3|t = 1)p(t = 1)}{p(x_3)} \\ &= \frac{p(a_1|t = 1)^2 p(a_2|t = 1)^3 p(a_3|t = 1)^4 p(t = 1)}{p(a_1)^2 p(a_2)^3 p(a_3)^4} \\ &= \frac{(5/11)^2 (4/11)^3 (2/11)^4 (1/2)}{(1/3)^2 (1/3)^3 (1/3)^4}. \end{aligned}$$

2 marks are awarded for computing the smoothed counts and another 2 marks for correctly computing $p(t = 1|x_3)$. No penalties for not simplifying the answers.

- (e) Assuming the reviews to be independent, compute the likelihood of this dataset using the smoothed counts. **(5 marks)**

Likelihood of the dataset is given by $p(x_1)p(x_2)p(x_3)p(x_4)p(x_5)$. By substituting $p(a_1), p(a_2), p(a_3)$ and aggregating the powers for each attributes we have, $p(a_1)^{10}p(a_2)^{10}p(a_3)^{10} = \frac{1}{3^{30}}$.

Question 5 We would like to project the four data points $x_1 = (0, 2)$, $x_2 = (-1, 0)$, $x_3 = (0, -2)$, $x_4 = (1, 0)$ shown in Figure 1 onto the $y = \tan(\theta)x$ line that passes through the origin $O = (0, 0)$ and has an angle $0 < \theta < \pi/2$ with the positive direction of the x -axis. The four corresponding projected points along the line segment are shown by A_1 , A_2 , A_3 and A_4 . Answer the following questions.

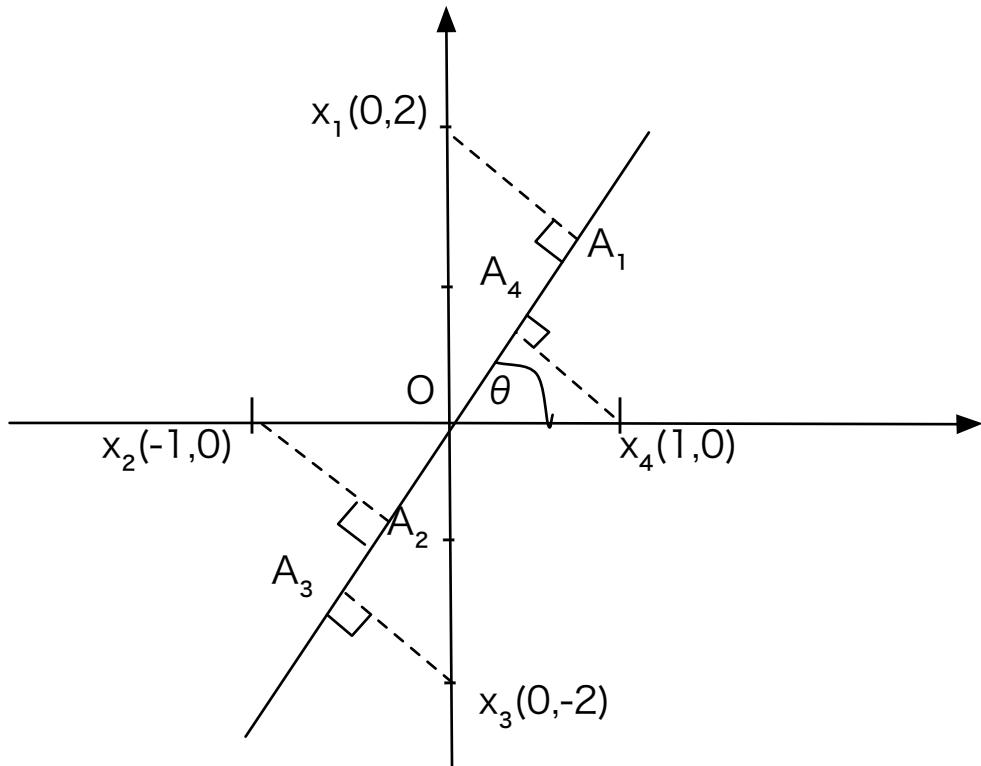


Figure 1: Four data points projected onto a straight line.

- A.** State two methods that can be used to project high dimensional data onto a lower dimensional space. **(2 marks)**

Principle Component Analysis, Singular Value Decomposition

- B.** Compute the perpendicular distances x_1A_1 , x_2A_2 , x_3A_3 , and x_4A_4 to the projection line $y = \tan(\theta)x$ respectively from the four points x_1 , x_2 , x_3 , and x_4 . **(4 marks)**

Let $d_1 = x_1A_1$, $d_2 = x_2A_2$, $d_3 = x_3A_3$, $d_4 = x_4A_4$. Then we have, $d_1 = 2\cos(\theta)$, $d_2 = \sin(\theta)$, $d_3 = 2\cos(\theta)$, $d_4 = \sin(\theta)$

- C.** Find the value of $\tan(\theta)$ for which the sum of squared projection errors is minimised. **(4 marks)**

The sum of squared projection error, $d_{sqd}^2 = d_1^2 + d_2^2 + d_3^2 + d_4^2 = 8\cos^2(\theta) + 2\sin^2(\theta)$. To find the minimum value of this error, we differentiate w.r.t. θ and set the result to zero.
 $\frac{\partial e}{\partial \theta} = -16\cos(\theta)\sin(\theta) + 4\sin(\theta)\cos(\theta) = 0$, which gives $\cos(\theta)\sin(\theta) = 0$. Therefore, $\theta = 0, \pi/2$. This corresponds to x and y axes respectively.

- D.** Compute the distances $e_1 = OA_1$, $e_2 = OA_2$, $e_3 = OA_3$ and $e_4 = OA_4$ to each of the projected points A_1, A_2, A_3, A_4 from the origin O . **(4 marks)**

$$e_1 = 2 \sin(\theta), e_2 = \cos(\theta), e_3 = 2 \sin(\theta), e_4 = \cos(\theta)$$

- E.** Compute the projected mean of the four data points. **(2 marks)**

Mean of the four points is $((0 - 1 + 0 + 1)/4, (2 + 0 - 2 + 0)/4) = (0, 0)$. Note that $(0, 0)$ gets projected to itself by $y = mx$. Therefore, the projected mean of the four data points is $(0, 0)$.

- F.** Compute the variance of the four data points on the line. **(4 marks)**

Because the mean is $(0, 0)$ and we have measured distances on the line from $(0, 0)$ already the variance is simply the sum of the squared distances (which is also the mean of the squared distances $\mathbb{E}[e^2]$ in this case) and can be computed as follows:

$$\mathbb{V}[e] = \mathbb{E}[e^2] = e_1^2 + e_2^2 + e_3^2 + e_4^2 = 8 \sin^2(\theta) + 2 \cos^2(\theta).$$

- G.** Compute the value of θ that maximises the variance of the projected points. **(5 marks)**

To find the optimal value of θ we set $\frac{\partial \mathbb{V}[e]}{\partial \theta}$ to zero, which gives,

$$\begin{aligned} 16 \sin(\theta) \cos(\theta) - 4 \cos(\theta) \sin(\theta) &= 0 \\ \sin(\theta) \cos(\theta) &= 0 \end{aligned}$$

The solutions are, $\theta = 0, \pi/2$. Therefore, maximising the sum of squared projection errors and minimising the variance of the projected points lead to the same optimal solution.



UNIVERSITY OF
LIVERPOOL

Resit Examinations 2016/17

Data Mining and Visualisation

TIME ALLOWED : Two and a Half Hours

INSTRUCTIONS TO CANDIDATES

Answer FOUR questions.

If you attempt to answer more questions than the required number of questions (in any section), the marks awarded for the excess questions answered will be discarded (starting with your lowest mark).

Question 1

- A.** State two techniques used in Support Vector Machines to classify linearly non-separable datasets. **(2 marks)**

Kernels, slack variables.

- (a)** Consider the quadratic kernel given by $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^\top \mathbf{y} + 1)^2$ for two vectors $\mathbf{x} = (x_1, x_2)^\top$ and $\mathbf{y} = (y_1, y_2)^\top$. Show that the quadratic kernel can be written as the inner-product between two 6-dimensional vectors, each containing information only from one of \mathbf{x} and \mathbf{y} . **(4 marks)**

$$\begin{aligned} k(\mathbf{x}, \mathbf{y}) &= (\mathbf{x}^\top \mathbf{y} + 1)^2 \\ &= (x_1 y_1 + x_2 y_2 + 1)^2 \\ &= x_1^2 y_1^2 + x_2^2 y_2^2 + 1 + 2x_1 y_1 + 2x_2 y_2 + 2x_1 x_2 y_1 y_2 \\ &= (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, 1)^\top (y_1^2, y_2^2, \sqrt{2}y_1, \sqrt{2}y_2, \sqrt{2}y_1 y_2, 1). \end{aligned}$$

As shown in the last step, the quadratic kernel can be represented as the inner-product between two 6-dimensional vectors each containing terms involving only \mathbf{x} or \mathbf{y} .

- (b)** Explain why a quadratic kernel might be able to learn a decision hyperplane for a non-linearly separable dataset. **(5 marks)**

As shown above, quadratic kernel is projecting 2 dimensional data to a high (6 dimensional) space. Therefore, the probability of finding a hyperplane in this high dimensional space that can linearly separate the dataset will increase. For example, in XOR non-linearity, the cross-product terms $x_1 x_2$ will be sufficient to find a hyperplane.

- B.** Consider three data points $\mathbf{x}_1 = (1, 0)$, $\mathbf{x}_2 = (3, -1)$, and $\mathbf{x}_3 = (3, 1)$. Here, \mathbf{x}_2 and \mathbf{x}_3 are labelled positively (+1), whereas \mathbf{x}_1 is negative (-1). Answer the following questions related to training a linear-kernel support vector machine on this dataset.

- (a)** Assuming that the decision hyperplane is defined by a weight vector $\mathbf{w} = (w_1, w_2)^\top$ and a bias term b , show that the prediction score y of a test instance $\mathbf{z} = (z_1, z_2)$ is given by $y = w_1 z_1 + w_2 z_2 + b$. **(3 marks)**

Prediction score is given by

$$y = \mathbf{w}^\top \mathbf{z} + b$$

. By substituting for \mathbf{x}, \mathbf{z} we derive the required result.

- (b)** Write the inequality constraints that must be satisfied by the three support vectors $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$. **(3 marks)**

$$\begin{aligned} -w_1 - b &\geq -1 \\ 3w_1 - w_2 + b &\geq 1 \\ 3w_1 + w_2 + b &\geq 1 \end{aligned}$$

- (c) By solving the inequality constraints you derived in (b), compute w_1, w_2 and b . **(3 marks)**

By solving the three simultaneous linear equations we obtain $w_1 = 1, w_2 = 0, b = -2$.

- (d) Plot the decision hyperplane alongside with the support vectors. **(3 marks)**

This will be a straight line parallel to the y-axis and passing through (2,0).

- (e) Predict the class label of a test data point (4, 1). **(2 marks)**

$y = 4 \times 1 - 2 = 2 > 0$. Therefore, this test instance is predicted as positive.

Question 2 We would like to use the Perceptron algorithm to learn a linear classifier $y = \mathbf{w}^\top \mathbf{x} + b$, defined by a weight vector $\mathbf{w} \in \mathbb{R}^d$ and a bias $b \in \mathbb{R}$ from a training dataset consisting of four instances, $\{(t_n, \mathbf{x}_n)\}_{n=1}^4$. Here, $\mathbf{x}_1 = (-1, 0)^\top$, $\mathbf{x}_2 = (1, 0)^\top$, $\mathbf{x}_3 = (1, 1)^\top$, and $\mathbf{x}_4 = (-1, 1)^\top$, and the labels are $t_1 = -1$, $t_2 = +1$, $t_3 = +1$, and $t_4 = -1$. We predict an instance \mathbf{x} as positive if $\mathbf{w}^\top \mathbf{x} + b > 0$, and negative otherwise. The initial values of the weight vector and the bias are set respectively to $\mathbf{w}^{(0)} = (0, 0)^\top$ and $b = 0$. We visit the training instances in the order $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$, and \mathbf{x}_4 . Answer the following questions.

- A. Plot the dataset in the two-dimensional space. (2 marks)

x₁ and x₂ will be on the x-axis and x₃ and x₄ will be parallel to this on line y = 1.

- B. Write the perceptron update rule for a misclassified instance (t, \mathbf{x}) . (3 marks)

$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + t\mathbf{x}$

- C. What will be the values of the weight vector and the bias after observing the instance \mathbf{x}_1 . (3 marks)

$y_1 = \mathbf{w}^{(0)^\top} \mathbf{x}_1 + b^{(0)} = 0$. Therefore, this instance is classified correctly as negative. The weight vector and bias are not updated. $w^{(1)} = (0, 0)^\top$, $b^{(1)} = 0$.

- D. What will be values of the weight vector and the bias after observing \mathbf{x}_2 . (4 marks)

$y_2 = 0$. Therefore, \mathbf{x}_2 will be incorrectly classified as negative. The weight vector and the bias will be update to $\mathbf{w}^{(2)} = (1, 0), b^{(2)} = 1$

- E. What will be the values of the weight vector and the bias after observing \mathbf{x}_3 . (4 marks)

$y_3 = (1, 0)^\top (1, 1) + 1 = 2 > 0$. Therefore, \mathbf{x}_3 will be correctly classified as positive, requiring no updates.

- F. What will be the values of the weight vector and bias after observing \mathbf{x}_4 . (4 marks)

$y_4 = (-1, 1)^\top (1, 0) + 1 = 0$. Therefore, \mathbf{x}_4 is correctly classified as negative, requiring no updates.

- G. If we re-assign the labels as $t_1 = -1, t_2 = +1, t_3 = -1, t_4 = 1$, show that there does not exist a weight vector $\mathbf{w} = (w_1, w_2)^\top$ and a bias b that can classify all four instances correctly. (5 marks)

First let us assume there exist such a weight vector and a bias. Then the following four inequalities must be satisfied by w_1, w_2, b .

$$-w_1 + b \leq 0 \quad (1)$$

$$w_1 + b > 0 \quad (2)$$

$$w_1 + w_2 + b \leq 0 \quad (3)$$

$$-w_1 + w_2 + b > 0 \quad (4)$$

From (1) and (2) we have $w_1 > 0$. However, from (3) and (4) we have $w_1 < 0$. There does not exist w_1 that can satisfy both constraints. Likewise, from (1) and (3) we have $2b + w_2 \leq 0$, whereas from (2) and (4) we have $w_2 + 2b > 0$. Again we cannot have w_2 and b that satisfy both those constraints. Therefore, there does not exist \mathbf{w} and b for this linearly non-separable dataset.

Question 3 Consider a two-dimensional dataset consisting of five instances $x_0 = (0, 0)$, $x_1 = (1, 1)$, $x_2 = (-1, 1)$, $x_3 = (-1, -1)$, and $x_4 = (1, -1)$. We would like to cluster this dataset into two clusters using the k -means clustering algorithm. Answer the following questions.

- A. Write the within cluster sum of squares objective function for a set of K clusters S_1, S_2, \dots, S_K . (3 marks)

$$\sum_{j=1}^K \sum_{x \in S_j} \|\mathbf{x} - \boldsymbol{\mu}_j\|^2$$

- B. Explain why it is important to randomly initialise the k -means algorithm and run for multiple times. (4 marks)

The within cluster sum of squares objective that is optimised by the k -means algorithm is a nonconvex function. Therefore, there is no guarantee that we will reach the minimum value on any single iteration started from an arbitrary initial set of cluster centres. The algorithm can in practice stuck in local minima. To overcome this issue, we must repeat the algorithm with different random initialisations and select the final set of clusters at convergence that correspond to the minimum value of the objective function.

- C. Let us assume the two initial cluster centres to be x_1 and x_2 . Find the two clusters produced by the k -means algorithm at convergence. (3 marks)

$\{x_2, x_3\}, \{x_0, x_1, x_4\}$

- D. Compute the value of the within cluster sum of squares objective for the set of clusters obtained in (C). (2 marks)

2.0 and 2.67. The total is 4.67

- E. Let us assume the two initial cluster centres to be x_0 and x_2 . Find the two clusters produced by the k -means algorithm at convergence. (3 marks)

$\{x_2\}, \{x_0, x_1, x_3, x_4\}$

- F. Compute the value of the within cluster sum of squares objective for the set of clusters obtained in (E). (2 marks)

0 and 5.5. The total is 5.5

- G. Let us assume the two initial cluster centres to be x_4 and x_2 . Find the two clusters produced by the k -means algorithm at convergence. (3 marks)

$\{x_2\}, \{x_0, x_1, x_3, x_4\}$

- H. Compute the value of the within cluster sum of squares objective for the set of clusters obtained in (G). (2 marks)

0 and 5.5. The total is 5.5

- I. Based on your results above, which set of clusters should we select? Justify your answer. **(3 marks)**

We must select the set of clusters that correspond to the minimum value of the within cluster sum of squares objective, which is $\{x_2, x_3\}, \{x_0, x_1, x_4\}$ (corresponding to the objective function value of 4.67)

Question 4 Let us assume that we must develop a sentiment classifier to predict sentiment of user reviews about products for an online e-commerce portal. Answer the following questions.

- A. Providing examples, explain what is meant by the term *stop word* in the context of text mining. **(3 marks)**

Stop words are non-content words such as functional words that can be safely removed from a document when representing the document using some features. For example, words such as a, an, the, is are considered as stop words.

- B. State a benefit of removing stop words when training a classifier. **(2 marks)**

This will reduce the number of features, thereby speeding up the training, testing times, and save memory when storing the train/test instances.

- C. Providing examples, explain what is meant by the term *part-of-speech tagging* in the context of text mining. **(3 marks)**

Part-of-speech (POS) tagging is the task of assigning words in a text parts of speeches such as nouns, verbs, adjectives, adverbs etc. The set of tags is pre-defined such as in the Penn POS tag set. For example, given the sentence, I like eating burgers. it will be assigned POS tags as follows: I/PRP like/VBP eating/VBG burgers/NNS

- D. Why would it be useful to use part-of-speech tags when training a sentiment classifier? **(2 marks)**

Sentiment is often expressed using adjectives. Therefore, knowing a particular feature (unigram) is an adjective can be useful clue to the sentiment classifier.

- E. Why would it be good to use bigrams in addition to unigrams when representing user reviews for training a sentiment classifier. **(4 marks)**

Negation indicators such as not will be handled as individual features under a unigram only feature representation. This can be problematic when training a sentiment classifier because we cannot accurately detect what has been negated in the document. On the other hand, bigrams will retain the negation indicators with the target of the negation. For example, not+great will be represented as a single bigram, indicating a negative sentiment.

- F. If the user reviews are rated from 1 to 5 stars in a discrete ordinal scale, where higher the value more positive the sentiment, how would you assign labels to the reviews such that you can train a binary sentiment classifier? **(3 marks)**

We could discretise the ratings. For example, we can assign a negative (-1) label to reviews rated either 1 or 2, whereas a positive (+1) label can be assigned to reviews rated either 4 or 5. We can ignore all reviews with rating 3 because they can be ambiguous with respect to the sentiment expressed.

- G. If our dataset contains identical copies (duplicate reviews), will it affect the performance of a naive Bayes classifier? Explain your answer. **(4 marks)**

Yes. Each feature in the duplicated reviews will be counted multiple times within each class. This will increase the likelihoods $p(w|t)$ of words, thereby affecting the posterior probability according to the Bayes' rule. Answers without the explanation will receive only 1 mark.

- H.** We would like to apply singular value decomposition to reduce the size of the feature vectors representing the reviews. Assuming that you do not have a separate validation dataset, how can you determine the optimal dimensionality for the singular value decomposition? **(4 marks)**

We can split the training dataset into two parts: a train part (80%) and a validation part (20%). We can then perform singular value decomposition-based dimensionality reduction under different dimensionalities and evaluate the sentiment classification accuracy of the trained binary sentiment classifier on the validation part of the dataset. We can then select the dimensionality that returns the best classification accuracy.

Question 5 Consider the multi-layer feedforward neural network shown in Figure 1. This neural network has 3 inputs x_1, x_2 and x_3 connected to a hidden layer consisting of two nodes h_1 and h_2 . The weight of the edge connecting x_i to h_j is w_{ji} . The two hidden nodes are connected to the output node o . The weight of the edge connecting the hidden node h_i to the output node o is u_i . The activation functions at hidden and output layers is set to sigmoid function defined as follows:

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

Moreover, squared error is used as the loss function at the output node, and is defined as,

$$E(o, t) = \frac{1}{2}(o - t)^2,$$

where t is the target output.

Answer the following questions.

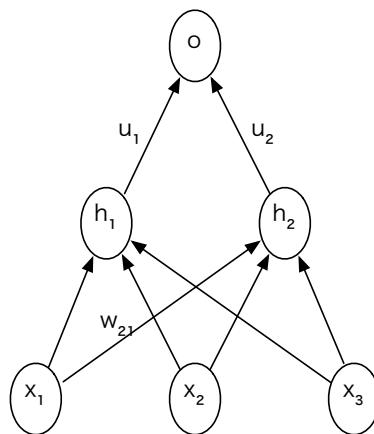


Figure 1: A multi-layer feedforward neural network.

- A. State one advantage and one disadvantage of using a single layer neural network vs. multi-layer neural network. **(2 marks)**

A single layer neural network can only classify linearly separable data, whereas a multi-layer neural network has the capability to handle non-linear data. However, the number of edge weights grow rapidly with the number of layers used in the neural network making it (a) time consuming to train, (b) require large labeled datasets, and (c) likely to overfit to the train data.

- B. Using the symbols defined in Figure 1, compute the activation at h_1 . **(3 marks)**

$$a(h_1) = \sigma(x_1w_{11} + x_2w_{12} + x_3w_{13})$$

- C. Compute the gradient of the loss with respect to the output o . **(3 marks)**

$$\frac{\partial E}{\partial o} = o - t$$

- D.** Compute the gradient of the loss w.r.t u_1 . **(3 marks)**

$$\begin{aligned}
 O &= \sigma(\sigma(h_1)u_1 + \sigma(h_2)u_2) \\
 \frac{\partial O}{\partial u_1} &= \sigma'(\sigma(h_1)u_1 + \sigma(h_2)u_2)\sigma(h_1) \\
 \therefore \frac{\partial E}{\partial u_1} &= \frac{\partial E}{\partial O} \frac{\partial O}{\partial u_1} = (o - t)\sigma'(\sigma(h_1)u_1 + \sigma(h_2)u_2)\sigma(h_1)
 \end{aligned}$$

Here, $h_1 = x_1w_{11} + x_2w_{12}$ and $h_2 = x_1w_{21} + x_2w_{22}$.

- E.** Compute the gradient of the loss with respect to w_{12} . **(4 marks)**

$$\begin{aligned}
 \frac{\partial O}{\partial h_1} &= \sigma'(\sigma(h_1)u_1 + \sigma(h_2)u_2)u_1 \\
 \frac{\partial h_1}{\partial w_{12}} &= \sigma'(x_1w_{11} + x_2w_{12} + x_3w_{13})x_2 \\
 \frac{\partial E}{\partial w_{12}} &= \frac{\partial E}{\partial h_1} \frac{\partial h_1}{\partial w_{12}} \\
 &= (o - t)\sigma'(\sigma(h_1)u_1 + \sigma(h_2)u_2)u_1\sigma'(x_1w_{11} + x_2w_{12} + x_3w_{13})x_2
 \end{aligned}$$

- F.** Using the Stochastic Gradient Descent rule, write the update rule for w_{12} . **(4 marks)**

$$w_{12}^{(k+1)} = w_{12}^{(k)} - \eta \frac{\partial E}{\partial w_{12}}$$

We can substitute for $\frac{\partial E}{\partial w_{12}}$ from part (E) above.

- G.** Using the update rule derived in (F), explain why we should scale the initial values of the weights such that the activation at the output node does not fall in the saturated regions of the logistic sigmoid function. **(3 marks)**

At the saturated regions of the logistic sigmoid we have either $\sigma(\theta) \rightarrow 0$ or $\sigma(\theta) \rightarrow 1$, for which the update becomes small (or zero). Therefore, the weight will get stuck in the initial values and will not get updated.

- H.** If we would like to learn a sparse neural network where most of the edge weights are set to zero, how can we modify the loss function. **(3 marks)**

We can add a regularisation term to the loss function. For obtaining sparse solutions, for example, we can add ℓ_1 regularisation terms for $\mathbf{w} = (w_{11}, w_{12}, w_{13}, w_{21}, w_{22}, w_{23})^\top$ and $\mathbf{u} = (u_1, u_2)^\top$. The modified loss function will be then,

$$E = \frac{1}{2}(o - z)^2 + \lambda_1 \|\mathbf{w}\|_1 + \lambda_2 \|\mathbf{u}\|_1$$

Data Representation



UNIVERSITY OF
LIVERPOOL

Types of data we encounter

- Categorical data
 - discrete categories (colour of a flower petal)
- Numerical data
 - integer values (number of petals in a flower)
 - real values (length of a petal)
- String/textual data
 - words in a document
- time series data / sequential data
 - continuous, chronological, flows in one direction

Things to consider

- What is the best data type to represent the value of a given variable?
- What type of data does a particular algorithm can handle?
- How can we convert one data type to a different data type?
- What is the scale/range of the data type that we are using and is it appropriate?

Case Study

- In what ways can we represent the following sentence?

The burger I ate was an awesome burger!

Case Study

- In what ways can we represent the following sentence?

The burger I ate was an awesome burger!

Method 1: By a list of words?

```
["the", "burger", "i", "ate", "was", "an", "awesome", "burger"]
```

Method 2: By the set of words?

```
{"the", "burger", "i", "ate", "was", "an", "awesome"}
```

Method 3: By a vector of word frequency?

```
("the":1, "burger":2, "i":1, "ate":1, "was":1, "an":1, "awesome":1)
```

Method 4: By a vector of letter frequency???

```
{'a': 3, ' ': 7, 'b': 2, 'e': 6, 'g': 2, 'i': 2, 'h': 1, 'm': 1, 'o': 1, 'n': 1, 's': 2, 'r': 4, 'u': 2, 't': 2, 'w': 1}
```

Data Representation

- Data representation is one of the first things we must do in data mining.
- What we can mine is largely determined by our data representation.
- There is no one best data representation method for all data mining tasks
- For example, unsuitable data representations will lead to poor classification performance irrespective of the classification algorithm.
- We represent a data point using a set of **features**

What is a feature?

- features are attributes of data points that we can use to represent the data points
- For example, “colour” can be a feature when representing a “flower”. The value (feature value) of the feature “colour” could be “red”.
- Features are central to data mining
- Features allow us to abstract data points and learn rules that can be used to predict things for unseen/future data points
 - If we learn the rule that
 - if colour== red then flower= rose
 - we can use this rule to classify not only flowers in our train dataset, but also all flowers including ones that we do not have in our train dataset
- Coming up of good features is an art!
 - feature-engineering
 - Recent work in machine learning such as deep learning focus on automatically discovering good features from data, without any human intervention

Categorical data

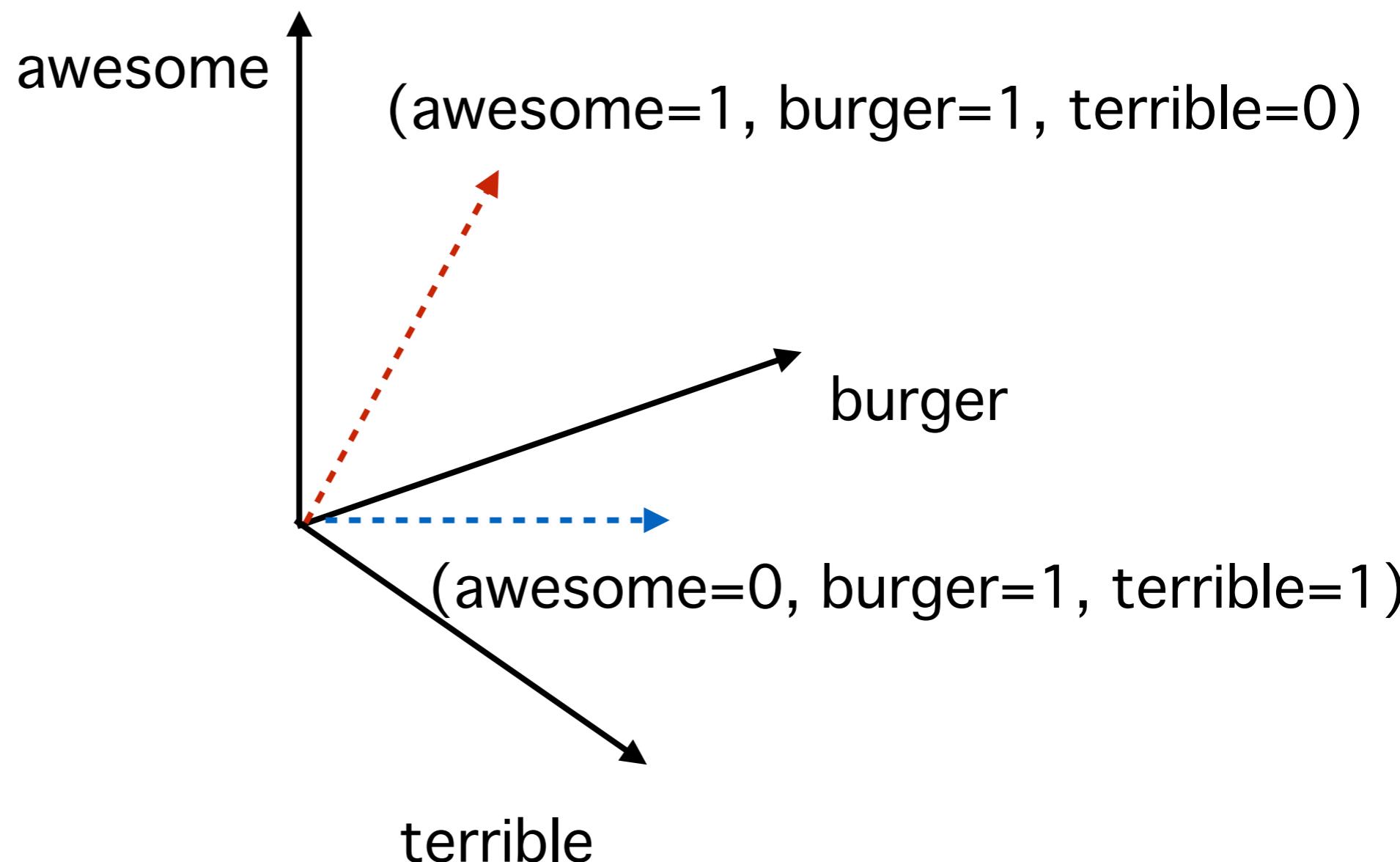
- Examples: colour of a flower
 - red, blue, green, ...
- Simple to represent and often consist of a small finite set of categories
- Easy to work when learning rules for data classification
 - if (colour == red) then flower = rose
- The number of categories need not necessarily be small
 - e.g. tags people use to label images such as names of people, locations, events or #tags is an open set

Challenges when using categorical data

- Algebraic comparisons are undefined!
 - rose(colour=red) vs. orchid(colour=yellow) ?
- No notion of feature correlation
 - “yellow” and “amber” are closely related colours. But there is no way we can guess this by looking at the data values
- Compare the above situations to
 - $\text{text}_1(\text{awesome}=4)$ vs. $\text{text}_2(\text{awesome}=0)$
 - if $\text{awesome} > 0$ then **POSITIVE_SENTIMENT**

Representing categorical data

- If you must represent categorical data, then you could do so by representing each category as a separate dimension of a vector.



Numerical data

- The number of possible values a variable can take can be infinite
 - all real values in range $[0,1]$
- There is a clearly defined ordering among the values (eg. $0.5 > 0.3$)
- Algebraic operations are well-defined
- “most” machine learning algorithms assume you have your data points represented in (for example) n-dimensional real space

$$\boldsymbol{x} \in \mathbb{R}^n$$

Challenges when handing numerical data

- Different features will be in different ranges.
 - $\text{height} \in [110, 230]$ (in centimetres)
 - $\text{weight} \in [40, 120]$ (in kilograms)
- If we learn some classification rule for body mass index (BMI), then we must consider the differences in ranges in each data dimension

Feature normalization

- There are various ways to normalize (scale) a numerical features into a common scale
- Method 1: [0, 1] scaling
 - compute the minimum and maximum value of a feature over the data points

$$\hat{x} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Method 2: Gaussian Normalization

- Compute the mean (μ) and the standard deviation (σ) for the feature and apply the following transformation

$$\hat{x} = \frac{x - \mu}{\sigma}$$

After this transformation each feature will have a zero mean and a unit variance. Therefore, it is “easier” to compare two features, ignoring their absolute scales.

Quiz

- Assume that the height of a student takes the following values
[$s_1=170\text{cm}$, $s_2=160\text{cm}$, $s_3=155\text{cm}$, $s_4=165\text{cm}$]
- Use method 1 ([0,1] normalization) to transform the four data points.
- Use method 2 (Gaussian normalization) to transform the four data point.



UNIVERSITY OF
LIVERPOOL

Second Semester Examinations 2015/16

Data Mining and Visualisation

TIME ALLOWED : Two and a Half Hours

INSTRUCTIONS TO CANDIDATES

Answer FOUR questions.

If you attempt to answer more questions than the required number of questions (in any section), the marks awarded for the excess questions answered will be discarded (starting with your lowest mark).

Question 1

- A. State the two main types of data mining models. **(2 marks)**

Predictive models and descriptive models. Each point will be assigned 1 mark.

- B. You are required to classify a given set of 100 flowers into three classes based on four attributes: sepal length, sepal width, petal length, and petal width. Answer the following questions about this experiment.

- (a) State two algorithms that you can use to learn a three-class classifier for this purpose. **(2 marks)**

logistic regression, SVM, perceptron, etc.

- (b) Assume that 20 out of the 100 flowers in your dataset do not have their petal length measured. John suggests that you use ignore petal length as an attribute and use the remaining three attributes during training. State a problem that you might encounter if you follow John's suggestion. **(3 marks)**

Petal length might be an important feature for this classification task. You might potentially loose accuracy of the classifier trained if you drop this attribute.

- (c) Instead of dropping petal length as an attribute, David suggests that you ignore the 20 flowers for which you do not have petal length measurements, and use the remaining 80 for training the classifier. State a problem that you might encounter if you follow David's suggestion. **(4 marks)**

You loose training instances if you follow this suggestion. We could overfit to the smaller training dataset if we were to fit many attributes on them, thereby potentially having poor test accuracy.

- (d) Instead of following John's or David's suggestions, Mary suggests that you set the missing petal length attribute values to zero, and use the entire 100 flowers for training the classifier. State a problem that you might encounter if you follow Mary's suggestion. **(4 marks)**

A flower cannot have a zero length petal. Therefore, you have effectively introduced noise into your train data set by replacing the missing values to zero. This might reduce the accuracy of the classifier because the replacement value of zero is inconsistent with the distribution of values for the petal length.

- (e) Instead of replacing the missing petal lengths by zero as suggested by Mary, propose a better value. **(4 marks)**

We could replace the missing petal lengths to the average value for the 80 flowers for which we have petal lengths.

- (f) Assuming that there is a high positive correlation between sepal length and petal length, how can you exploit this information to overcome the missing value problem of petal lengths? **(4 marks)**

We could learn a linear relationship between the two variables using a technique such as the linear regression and then use the learnt predictor to predict the missing values. We can then train a three-class classifier using this predicted data points as well as the original data points.

- (g) Without having access to a separate test dataset, how can we evaluate the accuracy of our three-class classifier? **(2 marks)**

We can set aside a portion of the train data as held out data, and evaluate using that portion.

Question 2 Assume that we are required to cluster a given set of 100 news articles according to the news topics mentioned in those articles. We tokenise each document into a set of unigrams and remove stop words using a pre-defined stop words list. We then count the frequency of occurrence of a word in a document, and represent the document using a feature vector where each dimension corresponds to a particular word. The feature values are set to the frequency of occurrence of the corresponding word in the document. We ℓ_2 normalise each feature vector. We then measure the distance between two documents using the Euclidean distance between the respective feature vectors. Finally, we use k -means clustering to generate the document clusters. Answer the following questions related to this document clustering task.

- A. Explain what is meant by a unigram as opposed to a bigram. **(2 marks)**

A unigram is a single word, whereas a bigram consists of consecutive two words.

- B. What is meant by *stop words* in text mining? **(2 marks)**

Stop words are non-content features such as prepositions and articles. For example, the, an, what, etc.

- C. Given a vector $x = (1, -1, 0)$, ℓ_2 normalise x . **(3 marks)**

(1/\sqrt{2}, -1/\sqrt{2}, 0)

- D. Why should we normalise the feature vectors representing documents before we compute Euclidean distance? **(3 marks)**

Without normalising longer documents we will have vectors with large feature values compared to shorter documents under the representation method described in the question.

- E. State a problem of using frequency of occurrence of a word in a document as its feature value? **(4 marks)**

Common words such as functional words are likely to have higher co-occurrence frequencies, although they do not convey much information about the topics discussed in the document.

- F. Propose a solution to overcome the problem you described in part E above. **(3 marks)**

Instead of using term frequency use tfidf as the feature value.

- G. Let us assume that we wanted to cluster the news articles into three clusters corresponding to political news, sports news, and foreign news. For this purpose let us assume that we ran k -means clustering with $k = 3$ but could not obtain three clusters covering the three categories as we wished. Propose a method to improve our chances of discovering clusters for the required categories using k -means. **(4 marks)**

Instead of randomly selecting the initial three documents (means), we can manually select three documents representing the three categories that we want to discover clusters for. We could further improve the probability of discovering the required categories by using centroids computed using multiple documents for each category as the initial cluster means.

- H.** Assuming that you are given a manually labeled set of news articles for the three categories mentioned in part **G**, explain a method to determine the optimal k value for the k -means clustering. **(4 marks)**

We can use the manually labeled news articles as the gold standard and compute the B-cubed F-score for a given set of clusters. During B-cubed evaluation we will consider only the labeled data and ignore the unlabeled data in a cluster. We will vary the value of k and select the value that produces the highest F-score.

Question 3 Consider a training dataset consisting of four instances $(\mathbf{x}_1, 1)$, $(\mathbf{x}_2, -1)$, $(\mathbf{x}_3, 1)$ and $(\mathbf{x}_4, -1)$ where $\mathbf{x}_1 = (1, 0)^\top$, $\mathbf{x}_2 = (0, 1)^\top$, $\mathbf{x}_3 = (-1, 0)^\top$, and $\mathbf{x}_4 = (0, -1)^\top$. Here, \mathbf{x}^\top denotes the transpose of vector \mathbf{x} . We would like to train a binary Perceptron to classify the four instances in this dataset. For this question ignore the bias term b in the Perceptron and answer the following.

- A. Write the perceptron update rule for a training instance (\mathbf{x}, y) which is misclassified by the current weight vector $\mathbf{w}^{(k)}$. **(2 marks)**

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + y\mathbf{x}$$

- B. Plot the four data points in the x-y plane. Is this dataset linearly separable? Justify your answer. **(3 marks)**

No, it is not. Answers that draw lines on the 2D plane for the extreme cases will receive full marks.

- C. Let us predict an instance \mathbf{x} to be positive if $\mathbf{w}^\top \mathbf{x} \geq 0$, and negative otherwise. Let us initialize $\mathbf{w} = (0, 0)^\top$. Compute the final value of the weight vector after presenting the training instances in the order $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$, and \mathbf{x}_4 . **(6 marks)**

When $\mathbf{w} = 0$, we have $\mathbf{w}^\top \mathbf{x}_1 = 0$. Hence, \mathbf{x}_1 is correctly predicted as positive. However, $\mathbf{w}^\top \mathbf{x}_2 = 0$, and \mathbf{x}_2 is misclassified. Therefore, using the update rule we will update the weight vector to $(0, 0) + (-1)(0, 1) = (0, -1)$. Next, for \mathbf{x}_3 we have $(0, -1)(-1, 0)^\top = 0$. Therefore, \mathbf{x}_3 is correctly classified. However, \mathbf{x}_4 will be misclassified because $(0, -1)^\top (0, -1) = 1 > 0$. Therefore, the final weight vector will be $(0, -1) + (-1)(0, -1) = (0, 0)$.

- D. If we continue training the perceptron in the same order as in part C for multiple iterations over the dataset, will the weight vector ever converge to a fixed solution? Explain your answer. **(4 marks)**

No it will not. As seen from part B, after the first iteration the weight vector returns to its initial value. The dataset is not linearly separable and this is a case where perceptron does not converge on such a dataset.

- E. If we present the four instances in the reverse order $\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1$, to the Perceptron, what would be the final value of weight vector at the end of the first iteration? **(6 marks)**

$(0, 0)(0, -1)^\top = 0$. Therefore, \mathbf{x}_4 is misclassified. The weight vector will be updated to $(0, 1)$. Next, $(0, 1)(-1, 0)^\top = 0$ and \mathbf{x}_3 is correctly classified. Next, $(0, 1)(0, 1)^\top = 1 > 0$ and \mathbf{x}_2 is misclassified. The updated weight vector will be $(0, 0)$. Finally, $(0, 0)(1, 0)^\top = 0$ and \mathbf{x}_1 is correctly classified. The final weight vector will be $(0, 0)$

- F. Now, let us re-assign the target labels for this dataset as follows $(\mathbf{x}_1, 1)$, $(\mathbf{x}_2, 1)$, $(\mathbf{x}_3, -1)$ and $(\mathbf{x}_4, -1)$. Can we use Perceptron algorithm to linearly classify this revised dataset? Justify your answer. **(4 marks)**

Yes. The dataset is linearly separable now and for a linearly separable dataset the perceptron will find a hyperplane that separates the two classes. Answers that either plots the data points in the 2D space or use some other method to show this will receive full marks.

Question 4 Assume that whether John would go to watch a football match depends on several factors such as the weather (being rainy, cloudy or sunny), temperature (being hot or cold), and traffic (high or less). Six past cases related to John's visits to football matches are shown in Table 1. Answer the following questions.

Table 1: John's past visits to football matches.

Weather	Temperature	Traffic	Go?
sunny	hot	less	yes
sunny	cold	high	no
cloudy	hot	high	no
rainy	cold	less	no
rainy	hot	less	yes
cloudy	cold	less	yes
sunny	hot	high	no
rainy	hot	high	no
cloudy	cold	high	no
sunny	cold	less	yes

- A. State three problems that are frequently observed in rule-based classifiers. **(6 marks)**
likely to overfit to the train data, can be time consuming when the dataset is large, too sensitive to the noise in the training data, cannot produce confidence scores. Each point will receive 2 marks.

- B. Using the dataset shown in Table 1, compute the coverage and the accuracy of the rule,

IF Weather = sunny THEN Go = Yes

(6 marks)

The rule covers 4 out of the 10 cases. Therefore, its coverage is 4/10. Out of those 4 matches, 2 cases have Go = YES. Therefore, the accuracy of the rule is 2/4. Correct answers for coverage will receive 3 marks and the correct answers for accuracy will receive 3 marks.

- C. Using Table 1 compute the conditional probabilities $P(Weather = sunny|Go = yes)$, $P(Weather = cloudy|Go = yes)$, and $P(Weather = rainy|Go = yes)$. **(6 marks)**
 $P(Weather = sunny|Go = yes) = 2/4$, $P(Weather = cloudy|Go = yes) = 1/3$, and $P(Weather = rainy|Go = yes) = 1/3$
- D. Use the Bayes' rule to compute $P(Weather = sunny|Go = yes)$. **(4 marks)**

$$\begin{aligned}
 & P(Weather = sunny|Go = yes) \\
 &= P(Weather = sunny|Go = yes)P(Go = yes)/P(Weather = sunny|Go = yes) \\
 &= (2/4) \times (4/10)/(4/10) = 0.5
 \end{aligned}$$

- E. Describe a method to overcome zero-probabilities when computing the likelihood of an event that can be decomposed into the product of a series of multiple independent events.

(3 marks)

Answers that describe Laplace smoothing or any other smoothing methods will receive full marks.

Question 5

- A.** Let us assume that we used some clustering algorithm to cluster a set 9 balls containing 2 red balls, 4 blue balls, and 3 green balls into 3 clusters as follows:

Cluster 1 = (red, red, blue)

Cluster 2 = (blue, blue, green)

Cluster 3 = (green, green, blue).

Answer the following questions about these clusters.

- (a) Following the majority labeling method determine the cluster labels. **(3 marks)**

Cluster 1 = red, Cluster 2 = blue, and Cluster 3 = green

- (b) Using the labels assigned in (a), compute the precision of red, blue and green classes. **(3 marks)**

Precision(red) = 2/3, Precision(blue) = 2/3, and Precision(green) = 2/3.

- (c) Using the labels assigned in (a), compute the recall of red, blue and green classes. **(3 marks)**

Recall(red) = 2/2, Recall(blue) = 2/4, and Recall(green) = 2/3.

- (d) Using the labels assigned in (a), compute the macro-averaged precision and macro-averaged recall. **(2 marks)**

Macro-averaged precision = $1/3 \times (2/3 + 2/3 + 2/3) = 2/3$. Macro-averaged recall = $1/3 \times (2/2 + 2/4 + 2/3) = 13/18$.

- (e) Using the labels assigned in (a), compute the micro-averaged precision and micro-averaged recall. **(4 marks)**

Let us denote the 2×2 contingency table (confusion matrix) for a label type in the following format: [(pred=true, actual=true), (pred=true, actual=false), (pred=false, actual=true), (pred=false, actual=false)]. Then, the contingency tables for the three label types will be, red = [2, 1, 0, 6], blue = [2, 1, 2, 4], green = [2, 1, 1, 5]. Adding the three tables element-wise we obtain, [6, 3, 3, 15]. Therefore, micro-averaged precision = 6/9, and micro-average recall = 6/9. One mark is awarded for computing the individual confusion matrices, one mark for adding them, one mark for computing precision, and one mark for computing recall.

- B.** Let us assume that we used a naive Bayes classifier for predicting whether a given email message is spam (positive class indicated by +1) or not (negative class indicated by -1). Table 2 shows the predicted probabilities $p(t = 1|x)$ for the positive class $t = 1$ for a given instance x , and the correct labels when evaluated on a test dataset containing 10 email messages. Answer the following questions about this classifier.

- (a) If we set the classification threshold to be 0.5 (i.e. if $p(t = 1|x) > 0.5$) then we predict x to be spam), compute the confusion matrix for this classifier. **(3 marks)**

Table 2: Predicted class probabilities and actual labels for 10 test instances.

$p(t = 1 x)$	actual label
0.79	1
0.83	-1
0.63	1
0.43	-1
0.32	1
0.23	-1
0.43	1
0.93	-1
0.83	1
0.75	-1

The predicted labels under 0.5 threshold will be as shown below.

$p(t = 1|x)$, actual label predicted label

0.79, 1 1, correct

0.83, -1 1, incorrect

0.63, 1 1, correct

0.43, -1 -1, correct

0.32, 1 -1, incorrect

0.23, -1 -1, correct

0.43, 1 -1, incorrect

0.93, -1 1, incorrect

0.83, 1 1, correct

0.75, -1 1, incorrect

Therefore, the confusion matrix (predicted vs. actual) will be [3, 3, 2, 2]. 1 mark for predicting the correct labels and 2 marks for computing the confusion matrix.

- (b) Compute the classification accuracy under the threshold 0.5. (2 marks)

$$\text{accuracy} = (3+2)/(3+3+2+2) = 0.5$$

- (c) What would be the accuracy if we increase the threshold to 0.7? (3 marks)

The predicted labels under 0.7 threshold will be as shown below.

$p(t = 1|x)$, actual label predicted label

0.79, 1 1, correct

0.83, -1 1, incorrect

0.63, 1 -1, correct

0.43, -1 -1, incorrect

0.32, 1 -1, incorrect

0.23, -1 -1, correct

0.43, 1 -1, incorrect

0.93, -1 1, incorrect

0.83, 1 1, correct

0.75, -1 1, incorrect

Therefore, the confusion matrix (predicted vs. actual) will be [2, 3, 3, 2].

Classification accuracy is 4/10 = 0.4.

- (d) Among the two spam classifiers obtained by setting the classification threshold to 0.5 and 0.7, which classifier would you prefer. Justify your answer. **(2 marks)**

Both classifiers have false positive rates of $3/5 = 0.6$. However, 0.5 threshold classifier marks 60% of emails as spam, whereas the 0.7 classifier marks only 50% of emails as spam. If you would like to avoid the case that accidentally a legitimate email being incorrectly classified as spam, you would prefer the classifier that marks lesser number of spam emails, which is the classifier with 0.7 threshold. An alternative answer would be to select the classifier with 0.5 threshold because it is more accurate. Both answers are considered correct if the relevant justification is provided. 1 mark for the answer and 1 mark for the justification.

Perceptron



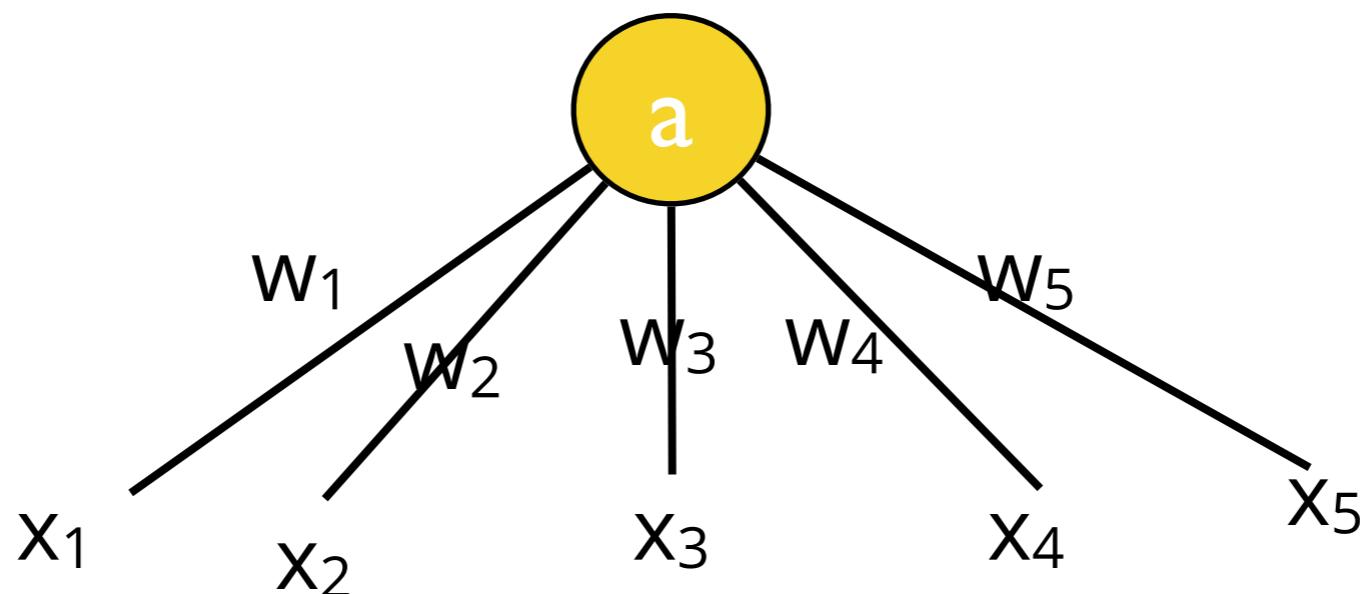
UNIVERSITY OF
LIVERPOOL

Bio-inspired model

- Perceptron is a bio-inspired algorithm that tries to mimic a single neuron
- We simply multiply each input (feature) by a weight and check whether this weighted sum (activation) is greater than a threshold.
- If so, then we “fire” the neuron (i.e. a decision is made based on the activation)

A single neuron

$$\text{activation (score)} = a = W_1X_1 + W_2X_2 + W_3X_3 + W_4X_4 + W_5X_5$$



```
if a > θ then  
    output = 1  
else  
    output = 0
```

If the activation is greater than a predefined threshold, then the neuron fires.

Bias

- Often we need to adjust a fixed *shift* from zero, if the “interesting” region happens to be far from the origin.
- We adjust the previous model by including a bias term b as follows

$$a = b + \sum_{i=1}^D w_d x_d$$

Notational trick

- By introducing a feature that is always ON (i.e. $x_0 = 1$ for all instances), we can squeeze the bias term b into the weight vector by setting $w_0 = b$

$$a = \sum_{i=0}^D w_d x_d = \mathbf{w}^\top \mathbf{x}$$

This is more “elegant” as we can write the activation as the inner-product between the weight vector and the feature vector. However, we should keep in mind that bias term still appears in the model.

Perceptron

- Consider only one training instance at a time
 - online learning
 - k-NN considers ALL instances (batch learning)
- Learn only if we make a mistake when we classify using the current weight vector.
Otherwise, we do not make adjustments to the weight vector
- Error-driven learning

Algorithm 5 PERCEPTRONTRAIN($\mathbf{D}, MaxIter$)

```
1:  $w_d \leftarrow 0$ , for all  $d = 1 \dots D$  // initialize weights
2:  $b \leftarrow 0$  // initialize bias
3: for  $iter = 1 \dots MaxIter$  do
4:   for all  $(x, y) \in \mathbf{D}$  do
5:      $a \leftarrow \sum_{d=1}^D w_d x_d + b$  // compute activation for this example
6:     if  $ya \leq 0$  then
7:        $w_d \leftarrow w_d + yx_d$ , for all  $d = 1 \dots D$  // update weights
8:        $b \leftarrow b + y$  // update bias
9:     end if
10:   end for
11: end for
12: return  $w_0, w_1, \dots, w_D, b$ 
```

Algorithm 6 PERCEPTRONTEST($w_0, w_1, \dots, w_D, b, \hat{x}$)

```
1:  $a \leftarrow \sum_{d=1}^D w_d \hat{x}_d + b$  // compute activation for the test example
2: return SIGN( $a$ )
```

slide credit: CIML (Daume III)

Detecting errors

- In Line 6 of PerceptronTrain code we have
 - $y_a \leq 0$
 - If the current instance is positive ($y = 1$), we should have a positive activation ($a > 0$) in order to have a correct prediction
 - If the current instance is negative ($y = -1$), we should have a negative activation ($a < 0$) in order to have a correct prediction
 - In both cases $y_a > 0$.
 - Therefore, if $y_a \leq 0$ then we have a misclassification

Update rule — Intuitive Explanation

- Perceptron update rule is
 - $\mathbf{w} = \mathbf{w} + y\mathbf{x}$
- If we incorrectly classify a positive instance as negative
 - We should have a higher (more positive) activation to avoid this
 - We should increase $\mathbf{w}^T \mathbf{x}$
 - Therefore, we should ADD the current instance to the weight vector
- If we incorrectly classify a negative instance as positive
 - We should have a lower (more negative) activation to avoid this
 - We should decrease $\mathbf{w}^T \mathbf{x}$
 - Therefore, we should DEDUCT the current instance from the weight vector

Update rule – Math Explanation

$$\begin{aligned} a' &= \sum_{d=1}^D w'_d x_d + b' \\ &= \sum_{d=1}^D (w_d + x_d) x_d + (b + 1) \\ &= \sum_{d=1}^D w_d x_d + b + \sum_{d=1}^D x_d x_d + 1 \\ &= a + \sum_{d=1}^D x_d^2 + 1 > a \end{aligned}$$

If the misclassified instance is a positive one, then after we update using $\mathbf{w} = \mathbf{w} + \mathbf{x}$, the new activation a' is greater than the old activation a .

Quiz 1

- Show that the analysis in the previous slide holds when $y = -1$ (i.e. we misclassified a negative instance)

$$a' = \sum_{d=1}^D w_d' x_d + b'$$

The misclassified instance is negative. Therefore, the update will be $w \leftarrow w - x_d$ and $b \leftarrow b - 1$

$$\begin{aligned}\therefore a' &= \sum_{d=1}^D (w_d - x_d) x_d + (b - 1) \\ &= \sum_{d=1}^D w_d x_d + b - \underbrace{\left(\sum_{d=1}^D x_d^2 + 1 \right)}_{> 0}\end{aligned}$$

$\therefore a' < a$ Therefore, the update will decrease the margin.

Things to remember

- There is no guarantee that we will correctly classify a misclassified instance in the next round.
- We have simply increased/decreased the activation but this adjustment might not be sufficient. We might need to do more aggressive adjustments
- There are algorithms that enforce such requirements explicitly such as the Passive Aggressive Classifier (not discussed here)

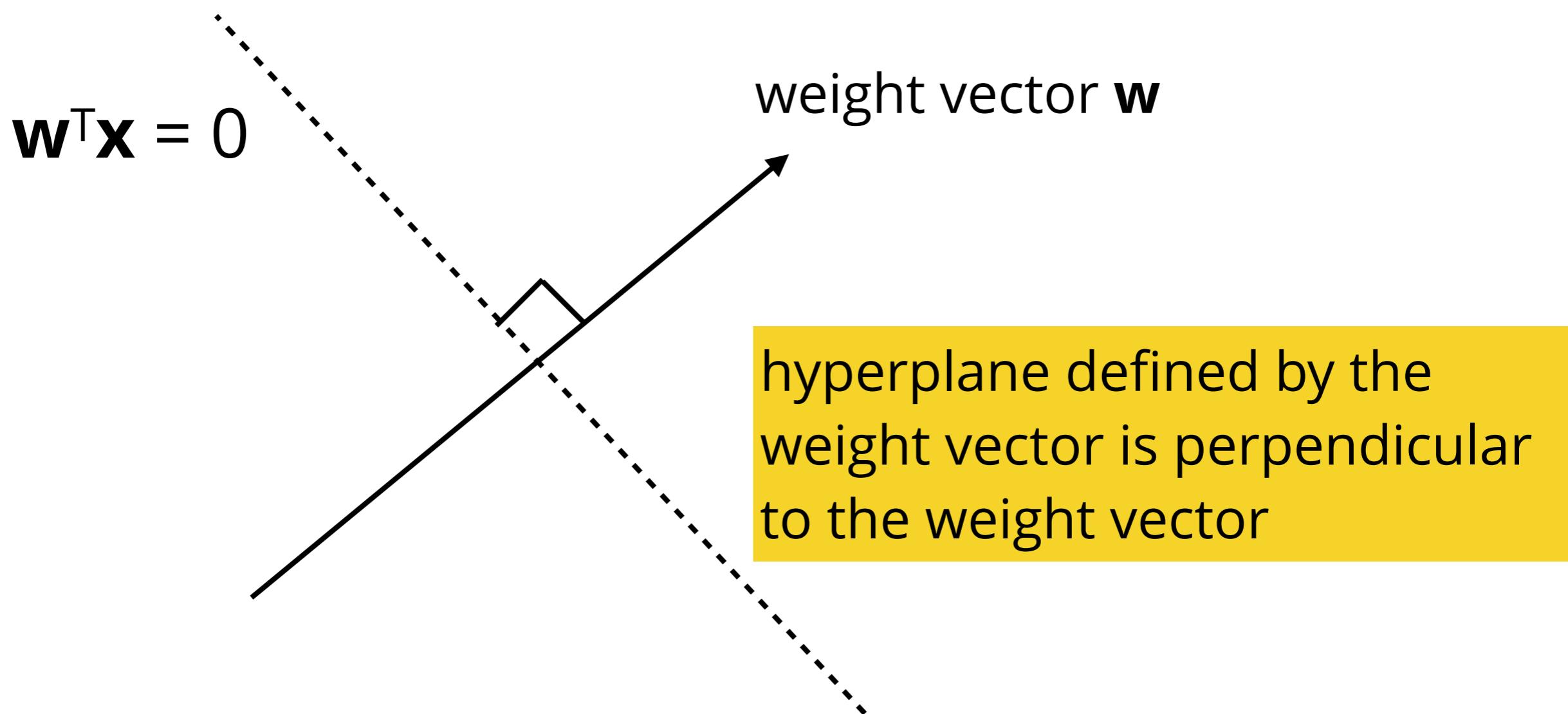
Ordering of instances

- Ordering training instances randomly within each iteration produces good results in practice
- Showing only all the positives first and all the negatives next is a bad idea

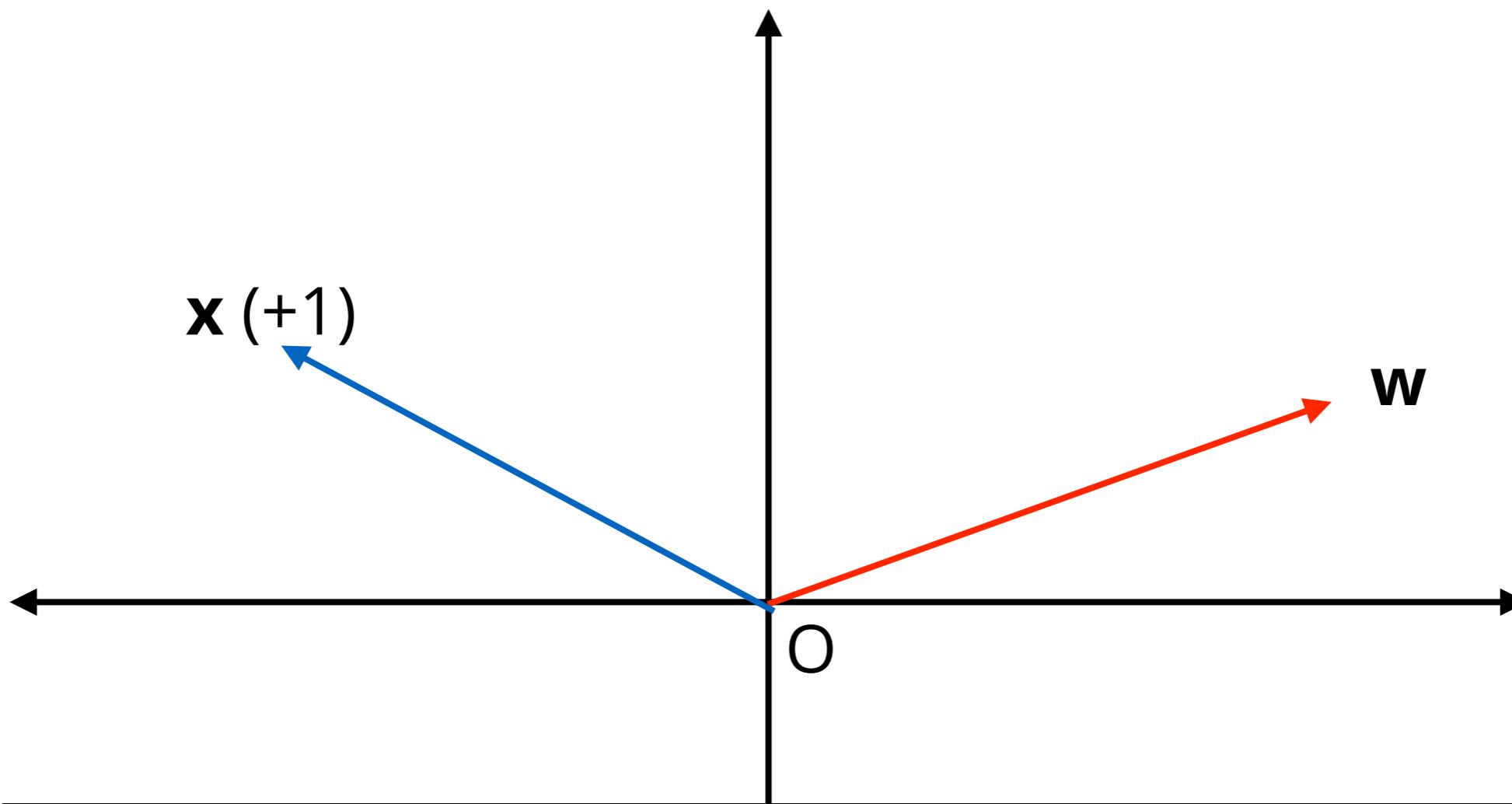
Hyperplane

- The decision in perceptron is made depending on $\mathbf{w}^T \mathbf{x} > 0$ or $\mathbf{w}^T \mathbf{x} \leq 0$
- Therefore, $\mathbf{w}^T \mathbf{x} = 0$ is the critical region (decision boundary)
- $\mathbf{w}^T \mathbf{x} = 0$ defines a hyperplane
- Example:
 - In 2D space we have $w_1x_1 + w_2x_2 = 0$ (ignoring the bias term), which is a straight line through the origin.
 - In N dimensional space this is an (N-1) dimensional hyperplane

Geometric Interpretation of Hyperplane

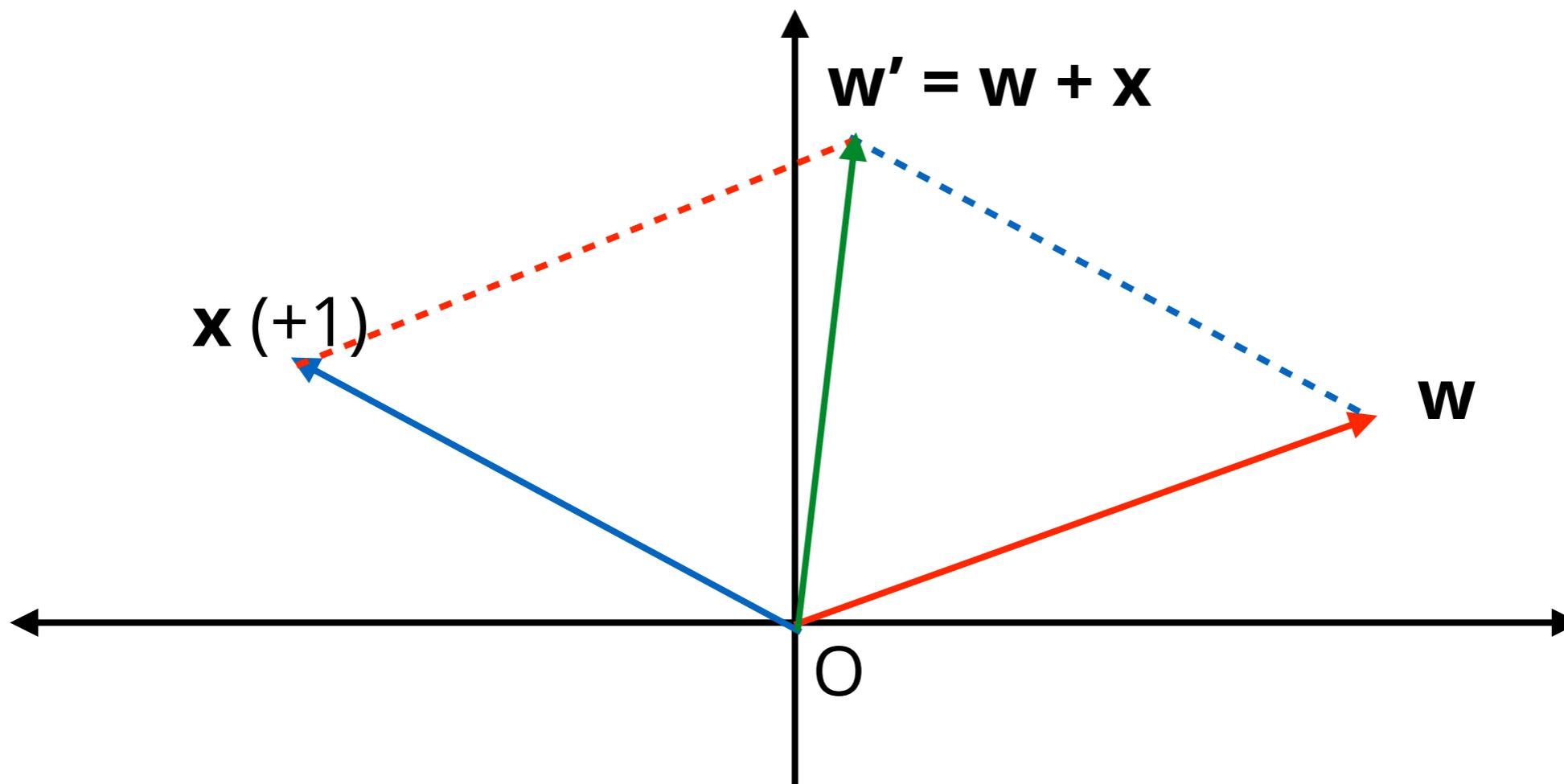


Geometric interpretation



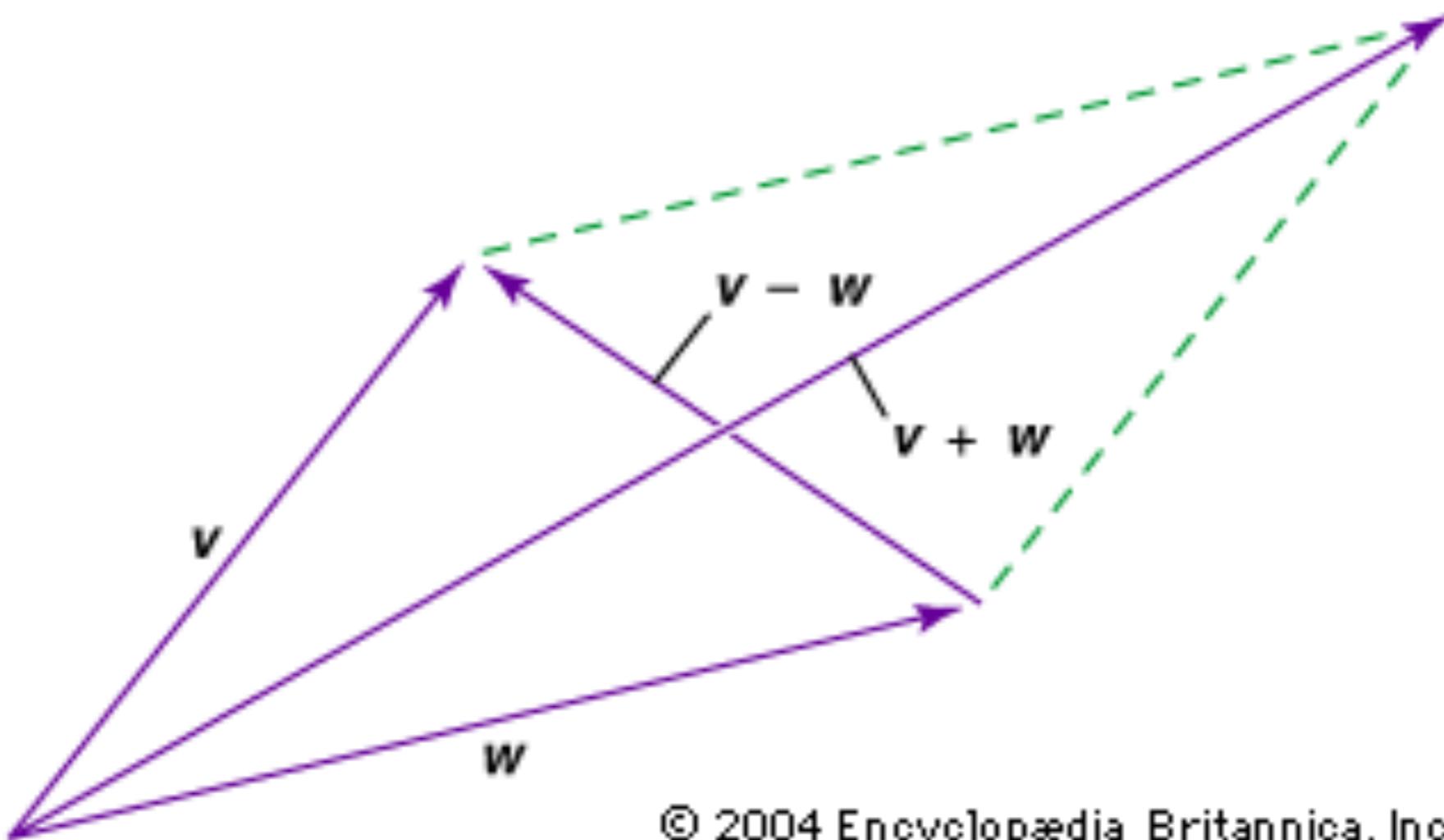
The angle between the current weight vector \mathbf{w} and the positive instance \mathbf{x} is greater than 90° . Therefore, $\mathbf{w}^T \mathbf{x} < 0$, and this instance is going to get misclassified as negative.

Geometric interpretation



The new weight vector \mathbf{w}' is the addition of $\mathbf{w} + \mathbf{x}$ according to the perceptron update rule. It lies in between **x** and **w**. Notice that the angle between \mathbf{w}' and **x** is less than 90° . Therefore, **x** will be classified as positive by \mathbf{w}' .

Vector algebra revision



© 2004 Encyclopædia Britannica, Inc.

Quiz 2

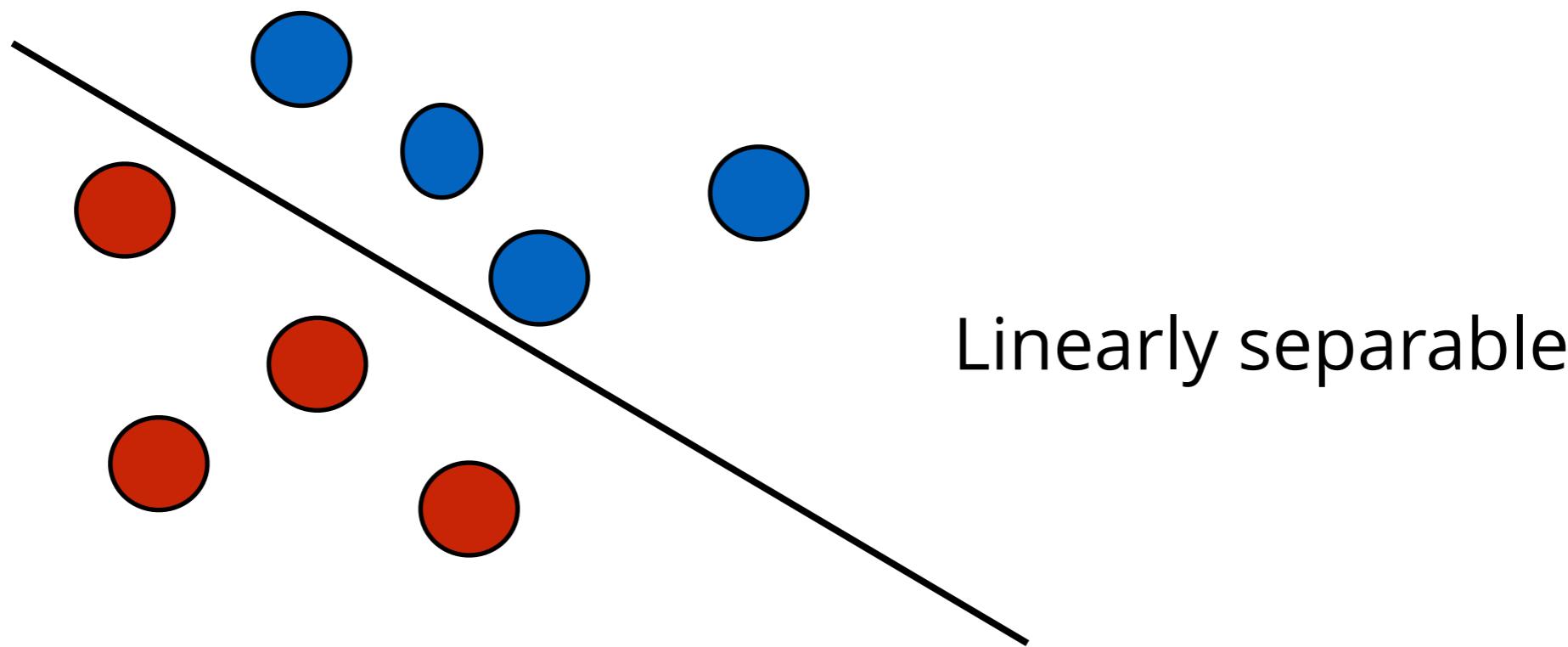
- Let $\mathbf{x} = (1, 0)^\top$ and $\mathbf{y} = (1, 1)^\top$. Compute $\mathbf{x}+\mathbf{y}$ and $\mathbf{x}-\mathbf{y}$ using the parallelogram approach described in the previous slide.

Quiz 3

- Provide a geometric interpretation for the update rule in Perceptron when a negative instance is mistaken to be positive.

Linear separability

- If a given set of positive and negative training instances can be separated into those two groups using a straight line (hyperplane), then we say that the dataset is *linearly separable*.

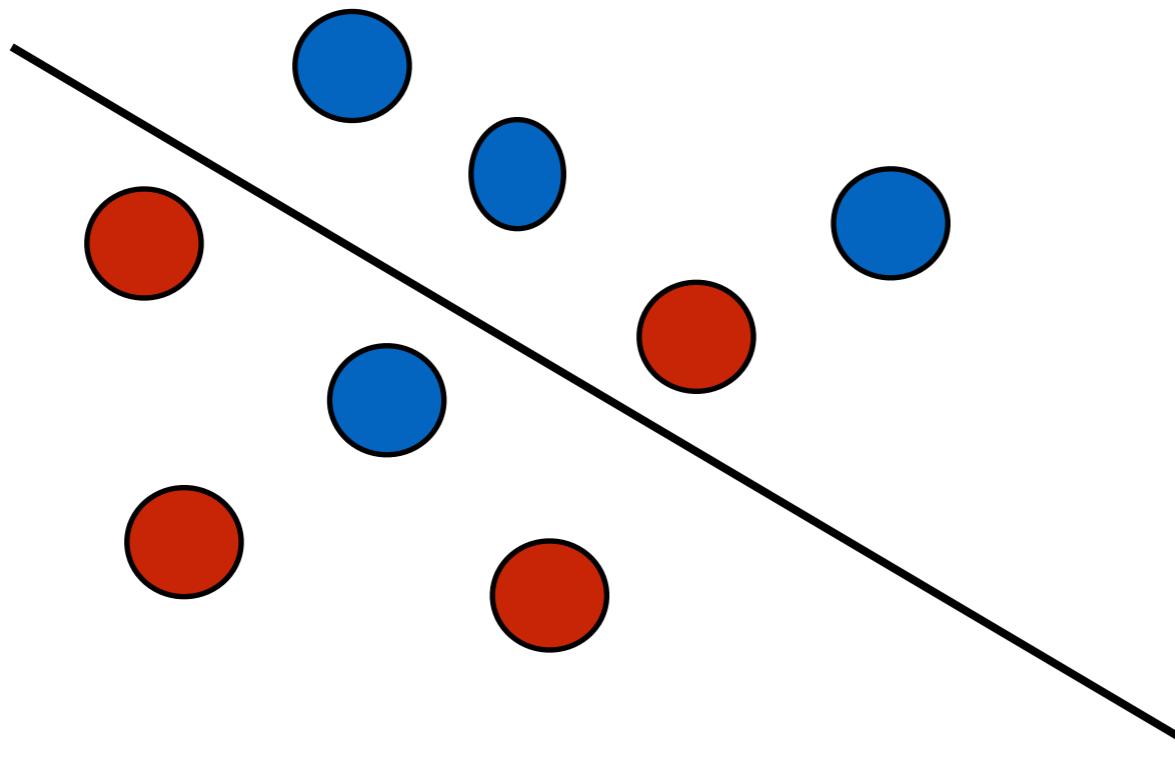


Remarks

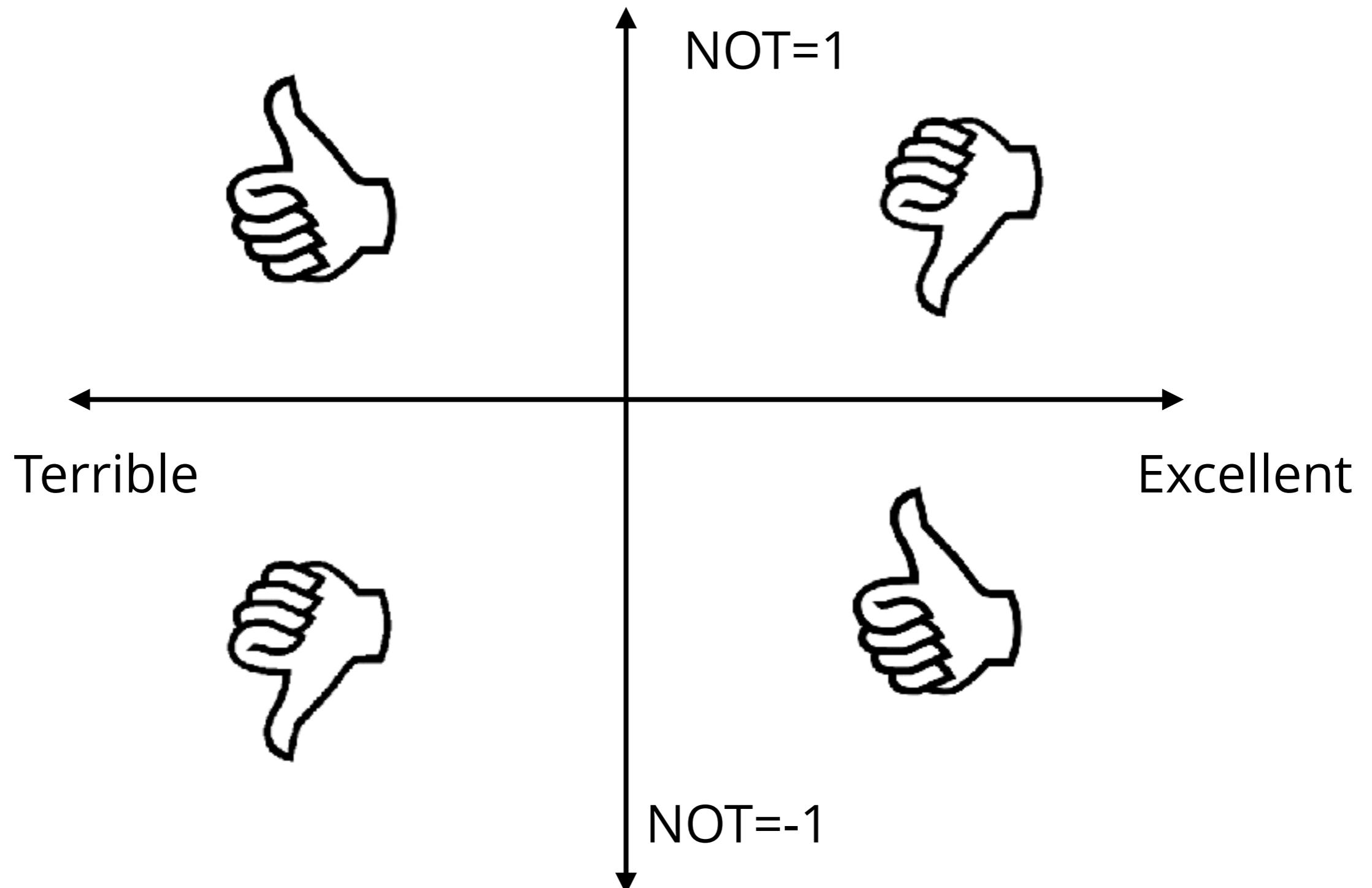
- When a dataset is linearly separable, there can exist more than one hyperplanes that separates the dataset into positive/negative groups.
- In other words, the hyperplane that linearly separates a linearly separable dataset might not be unique.
- However, (by definition) if a dataset is non-linearly separable, then there exist NO hyperplane that separates the dataset into positive/negative groups.

A non-linearly separable case

No matter how we draw straight lines, we cannot separate the red instances from the blue instances



Negation handling in Sentiment Classification



Mutually exclusive OR (XOR): $\text{XOR}(A, B) = 1$ only when one of the two inputs is 1.

Further Remarks

- When a dataset is linearly separable it can be proved that the perceptron will always find a separating hyperplane!
- The final weight vector returned by the Perceptron is more influenced by the final training instances it sees.
 - Take the average over all weight vectors during the training (averaged perceptron algorithm)



UNIVERSITY OF
LIVERPOOL

Resit Examinations 2015/16

Data Mining and Visualisation

TIME ALLOWED : Two and a Half Hours

INSTRUCTIONS TO CANDIDATES

Answer FOUR questions.

If you attempt to answer more questions than the required number of questions (in any section), the marks awarded for the excess questions answered will be discarded (starting with your lowest mark).

Question 1 Assume that you are given the dataset shown in Table 1 about heights (measured in centimeters) and weights (measured in kilograms) of a group of 5 students. Answer the following questions related to this dataset.

Table 1: Heights and weights of 10 students

Student no.	Height (cm)	Weight (kg)
1	169	60
2	171	70
3	150	80
4	180	75
5	150	60

- A. Heights and weights are in different ranges. Propose a method to scale both those values into the same [0,1] range. **(2 marks)**

$$\hat{x} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

- B. Using the method that you proposed in part A above scale the weights of the five students to [0,1] range **(5 marks)**

min = 60 and max = 80. Therefore, s₁ = 0, s₂ = 0.5, s₃ = 1.0, s₄ = 0.75, s₅ = 0.

- C. Assuming that there exists a strong correlation between the height and the weight of a person, propose a method to detect potentially incorrect weight measurements. **(4 marks)**

Because there is a strong correlation between the height and the weight of a person, we can learn a regression model such as a liner regression model. Next, we can use trained regression model to predict the weight of each student given that student's height. If the predicted value of the weight differs significantly from the measured value of the weight, then it is likely that the weight measurement was incorrect.

- D. Assume that you are planning to learn a naive Bayes classifier to predict whether a student is obese using their height and weight measurements. Will it be a problem if you accidentally took duplicate (repeated) measurements for some of the students? Explain your answer. **(4 marks)**

Yes. Duplicate data points will be a problem to the naive Bayes classifier because you would overestimate the counts for a paricular feature or a class.

- E. Assume that you are planning to learn a support vector machine to predict whether a student is obese using their height and weight measurements. Will it be a problem if you accidentally took duplicate (repeated) measurements for some of the students? Explain your answer. **(4 marks)**

No. Duplicate instances will be represented by the same point in the feature space. Therefore, duplicates will not affect the classification hyperplane.

- F. It turns out that 90% of the students are not obese. Why would classification accuracy be an inappropriate evaluation measure to evaluate the performance of the binary obese classifier we intend to learn from this dataset? **(3 marks)**

Because 90% of the students are not obese, by predicting non-obese for all the students (majority baseline) we will get a classification accuracy of 90%.

- G. Suggest a better classifier evaluation measure for the scenario discussed in part (F) above. **(3 marks)**

Area under the ROC curve (AUC).

Question 2 Assume we are given a training dataset consisting of four instances $(\mathbf{x}_1, +1)$, $(\mathbf{x}_2, +1)$, $(\mathbf{x}_3, -1)$, and $(\mathbf{x}_4, -1)$. Here, the $\mathbf{x}_1 = (1, 0)^\top$, $\mathbf{x}_2 = (0, 1)^\top$, $\mathbf{x}_3 = (-1, 0)^\top$, and $\mathbf{x}_4 = (0, -1)^\top$. We would like to train a binary k -nearest neighbour classifier from this dataset. Answer the following questions about this task.

- A. Compute the Euclidean distance between \mathbf{x}_1 and \mathbf{x}_2 . (3 marks)

$\sqrt{(1-0)^2 + (0-1)^2} = \sqrt{2}$. *1 mark will be awarded for writing the formula for Euclidean distance even if the computation is incorrect.*

- B. Compute the Manhattan distance between \mathbf{x}_1 and \mathbf{x}_2 . (3 marks)

$|1-0| + |0-1| = 2$. *1 mark will be awarded for writing the formula for Manhattan distance even if the computation is incorrect.*

- C. Using Euclidean distance as the distance measure and $k = 1$, classify an instance $\mathbf{x}^* = (0.5, 0)$. Explain your answer. (3 marks)

The Euclidean distances between \mathbf{x}^ and the four data points $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$, and \mathbf{x}_4 are respectively $0.5, \sqrt{5}/2, 1.5, \sqrt{5}/2$. Therefore, the label of the nearest neighbour of \mathbf{x}^* will be 1 . Therefore, \mathbf{x}^* will be classified as positive. The answers that do not show the reasoning will get only 1 mark.*

- D. Using Euclidean distance as the distance measure and $k = 3$, classify an instance $\mathbf{x}^* = (0.5, 0)$. Explain your answer. (3 marks)

The Euclidean distances between \mathbf{x}^ and the four data points $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$, and \mathbf{x}_4 are respectively $0.5, \sqrt{5}/2, 1.5, \sqrt{5}/2$. Therefore, the labels of the nearest neighbours of \mathbf{x}^* will be $1, 1, -1$. Therefore, \mathbf{x}^* will be classified as positive. The answers that do not show the reasoning will get only 1 mark.*

- E. Propose a strategy for determining the label of $\mathbf{x}^* = (0.5, 0)$ when $k = 2$. (4 marks)

The Euclidean distances between \mathbf{x}^ and the four data points $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$, and \mathbf{x}_4 are respectively $0.5, \sqrt{5}/2, 1.5, \sqrt{5}/2$. We have two sets of neighbours in this case $\{\mathbf{x}_1, \mathbf{x}_2\}$ or $\{\mathbf{x}_1, \mathbf{x}_4\}$. The first set of neighbours would predict positive whereas the second will be a split. A possible strategy will be to consider only the sets with a unique prediction and ignore splits (predicts positive). Alternative solutions would be to guess randomly.*

- F. What is the set of positively predicted data points by a 2-nearest neighbour classifier for this dataset? (3 marks)

This will be the set of data points in the first quadrant. $\{(\alpha, \beta) | \alpha > 0, \beta > 0\}$. Marks will not be penalised for not specifying the boundary conditions.

- G. Why would it be unwise to predict labels for test data using $k = 1$? (2 marks)

If the dataset is noisy and/or have outliers then the nearest neighbour only predictions are unreliable.

- H. If you were given a large labeled train dataset, propose a method to determine the best value of k for the k -nearest neighbour classifier. (4 marks)

Methods that splits the train dataset into held-out vs. train portions and perform parameter tuning, or cross-validation will receive full marks.

Question 3 In a typical information retrieval system the following steps are conducted to produce an inverted index. A document is first tokenised using space character as the delimiter. Next, using a pre-defined list, stop words are removed. Next, unigram tokens are selected from the remaining tokens in a document for creating an inverted index. The inverted index stores the list of ids of documents (posting list) in which a particular unigram occurs.

When a user enters a query, it is tokenised using the same procedure, stop words removed, and unigrams are extracted. Next, each unigram is searched against the created inverted index to obtain the corresponding posting lists. Finally, the intersection of all posting lists are returned to the user as the search result. Answer the following questions about this search engine.

- A. Explain what is meant by *stop word removal* in the context of text mining? **(2 marks)**

Remove non-content words such as articles using a pre-defined list of words.

Correct answers will receive 2 marks.

- B. Explain what is meant by a *unigram*. **(2 marks)**

A unigram is an individual token, e.g. cat.

- C. Let us assume that a user entered the query *data mining*. The posting list for the token *data* consists of two documents (denoted by their ids) $\{d_1, d_{10}\}$, and the token *mining* consists of three documents (denoted by their ids) $\{d_{10}, d_{12}, d_{15}\}$. What would be the result for the query *data mining*? Explain your answer. **(4 marks)**

This will be d_{10} because we are considering the intersection of the posting lists. Answers without explanations will receive only 2 marks.

- D. Using an example, explain how skip pointers can be used to speed up the computation of conjunctive (AND) queries. **(5 marks)**

We can put a pointer that indicates the value of the document id that we will encounter after a fixed number of skips. This pointer is called a skip pointer. For example, consider the two posting lists show below,

Brutus → 2[16] → 4 → 8 → 16[28] → 19 → 23 → 28

Caesar → 1[5] → 2 → 3 → 5[51] → 8 → 41 → 51

Here, the skip pointers are shown within square brackets. For example, when we are performing a linear search over the posting list for Caesar when we have matched up to 8 and must match 41 next, we know that we can skip the four documents after 16 and move straight up to 28 because we will not find 41 before we reach 28.

- E. State an advantage of performing stop word removal in an information retrieval system. **(3 marks)**

The posting lists for stop words such as the, an, at, etc. can be very large because those words occur virtually in all documents. Computing the intersections of long posting lists can be time consuming. By removing stop words prior to indexing the documents we can avoid this problem.

- F. The search engine described in this question does not take the word order into consideration. Propose a solution to overcome this problem. **(3 marks)**

We can use bigrams in addition to unigrams to represent both documents and queries. For example, data mining and mining data will be considered as different bigrams, hence different searchable units.

- G. Using examples describe what is meant by a *part-of-speech* in text mining? **(3 marks)**

Morphological categories of words such as nouns, verbs, adjectives, and adverbs are called part-of-speeches in text mining. An example of a noun would be cat. Answers that do not provide examples will receive only 2 marks.

- H. Explain the difference between static and dynamic ranking as used in information retrieval. **(3 marks)**

Static ranking methods such as the PageRank does not take into account the query when ranking the documents. On the other hand, dynamic ranking methods will consider both the query and the documents when ranking the documents.

Question 4 Consider the six transactions shown in Table 2 related to four items a , b , c , and d . We would like to find frequent itemsets from the database shown in Table 2. Answer the following questions.

Table 2: Itemsets for six transactions in a database.

Transaction ID	Itemset
T1	b,d
T2	a,b,c,d
T3	a,c
T4	c,d
T5	b,c,d
T6	a,b

- A. Using examples explain the difference between a substring and a subsequence. **(4 marks)**

Let us consider the string notebook. Substrings are continuous sequences of characters of a string. In this example, we have note, ote, tebo etc. as substrings. On the other hand, a subsequence does not have to be continuous and can have missing characters as long as the ordering (as given in the original string) is preserved. For example, we have n,t,k, n,o,b,k, n,o,o,k as subsequences. Answers that define the terms without examples will get a maximum of 2 marks.

- B. Why would it be unwise to generate all subsequences of strings in a database to find the most frequent subsequences? **(3 marks)**

The number of unique subsequences of a string with n unique letters is 2^n . Therefore, the number of candidate subsequences can grow very quickly, making it impossible to store the candidates in memory for finding the most frequent ones.

- C. What is meant by the apriori property in sequential pattern mining? **(3 marks)**

If S is a large itemset (with minimum support t), then any subset of S is also a large itemset (with the same minimum support t).

- D. Give that we are required to find itemsets with minimum frequency of 2, compute the minimum support for the database shown in Table 2. **(2 marks)**

Minimum support = minimum frequency / total transactions. Therefore, Minimum support = $2/6 = 0.33$.

- E. Under the minimum frequency of 2, state all large itemsets of length 1 for the database shown in Table 2. **(4 marks)**

There are four itemsets with frequency 2 and length 1. They are ((a), (b), (c), (d)). Each itemset will receive one mark.

- F. Under the minimum frequency of 2, state all large itemsets of length 2 for the database shown in Table 2. **(5 marks)**

There are 5 such itemsets as follows: ((a,b), (a,c), (b,c), (b,d), (c,d)). Note that (a,d) is small. Each itemset will receive 1 mark.

- G. Under the minimum frequency of 2, state all large itemsets of length 3 for the database shown in Table 2. **(2 marks)**

There is only one such itemset which is (b,c,d).

- H. Given a database containing a finite number of transactions, should the apriori algorithm always terminate? Explain your answer. **(2 marks)**

It will always terminate. During each iteration, the apriori algorithm grows the itemsets by adding a new item. In other words, the length of itemsets grow by 1 in each iteration. However, the total number of unique items will be finite given a finite set of transactions. Therefore, the apriori algorithm must terminate after a finite number of iterations. Correct answers without any justification will receive 1 mark. Additional 1 mark will be awarded for the justification.

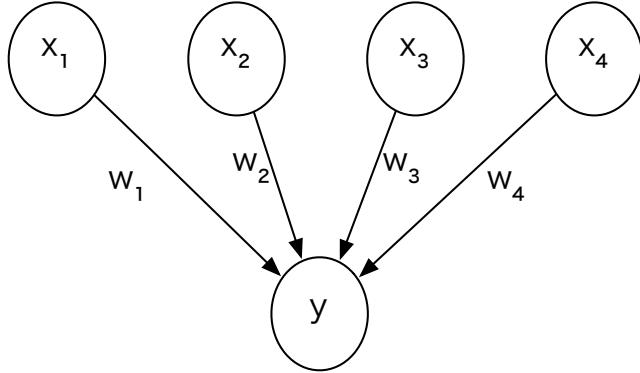


Figure 1: A single layer neural network.

Question 5 Figure 1 shows a neural network that receives 4 inputs x_1, x_2, x_3 , and x_4 , and multiplies inputs respectively by weights w_1, w_2, w_3 , and w_4 . The activation at the output node is y . Answer the following questions about this neural network.

- A.** State one advantage and one disadvantage of using a single layer neural network vs. multi-layer neural network. **(2 marks)**

A single layer neural network can only classify linearly separable data, whereas a multi-layer neural network has the capability to handle non-linear data. However, the number of edge weights grow rapidly with the number of layers used in the neural network making it (a) time consuming to train, (b) require large labeled datasets, and (c) likely to overfit to the train data.

- B.** Using the symbols specified in Figure 1, compute the activation at the output node y . **(4 marks)**

$$y = x_1w_1 + x_2w_2 + x_3w_3 + x_4w_4$$

- C.** Let us assume that the activation function at the output node is the logistic-sigmoid $\sigma(y)$ given by,

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

Compute the output value of the neural network. **(3 marks)**

$$\sigma(y) = \sigma(x_1w_1 + x_2w_2 + x_3w_3 + x_4w_4) = \frac{1}{1 + \exp(-x_1w_1 - x_2w_2 - x_3w_3 - x_4w_4)}$$

- D.** Let us assume the target label for this input to be t . Assuming squared loss, compute the loss associated with the prediction made by the neural network for the given input. **(2 marks)**

$$(t - \sigma(y))^2$$

- E.** Using the inputs x_1, x_2, x_3, x_4 and the target label t , we would like to learn the optimal values for the weights w_1, w_2, w_3 , and w_4 . Show that the partial derivative of the loss ℓ with

respect to w_1 is given by,

$$\frac{\partial \ell}{\partial w_1} = -2(t - \sigma(y))\sigma(y)(1 - \sigma(y))x_1.$$

If required, you may use the fact that,

$$\frac{\partial \sigma(y)}{\partial y} = \sigma(y)(1 - \sigma(y)).$$

(8 marks)

$$\begin{aligned}\frac{\partial \ell}{\partial w_1} &= \frac{\partial}{\partial w_1}(t - \sigma(y))^2 \\ &= -2(t - \sigma(y))\frac{\partial \sigma(y)}{\partial w_1} \\ &= -2(t - \sigma(y))\frac{\partial \sigma(y)}{\partial y} \frac{\partial y}{\partial w_1} \\ &= -2(t - \sigma(y))\sigma(y)(1 - \sigma(y))x_1\end{aligned}$$

- F.** We would like to use stochastic gradient descent with a fixed learning rate η for the optimisation. If the current value of the weight w_1 is denoted by $w_1^{(k)}$, then show that updated value $w_1^{(k+1)}$ of the weight w_1 is given by:

$$w_1^{(k+1)} = w_1^{(k)} + 2\eta(t - \sigma(y))\sigma(y)(1 - \sigma(y))x_1$$

(4 marks)

Substitute the loss gradient in the stochastic gradient descent update rule to obtain the update equation for w_1 .

- G.** Using the update equation given in part F, explain why we should scale the initial values of the weights such that the activation at the output node does not fall in the saturated regions of the logistic sigmoid function. **(2 marks)**

At the saturated regions of the logistic sigmoid we have either $\sigma(y) \rightarrow 0$ or $\sigma(y) \rightarrow 1$, for which the update becomes small (or zero). Therefore, the weight will get stuck in the initial values and will not get updated.

Data Mining Issues



UNIVERSITY OF
LIVERPOOL

Missing values

- What if we do not know the value of a particular feature of an instance?
 - eg. the height of the 10-th student is missing although other feature values are known for the 10-th student and all the other students in the dataset
- May be it is important to know this missing value because the height is an essential feature for our classification task such as obesity.

Handling missing values

- Ignore the training instance completely
 - might get away because of redundancy in the dataset. We might have another student with the same height as student no 10.
 - might not get away because student no 10 was the only student who had obesity. By ignoring student no 10, we loose all the information we had about the positive class.

Fill in values by hand

- Re-annotate the data or re-measure the instances with missing feature values
 - Reliable method to overcome the missing value problem
 - Might not be possible in practice because we no longer have access to the subjects.
 - Too slow (manual work)
 - Costly (manual work)
 - There might be lots of data points with missing feature values

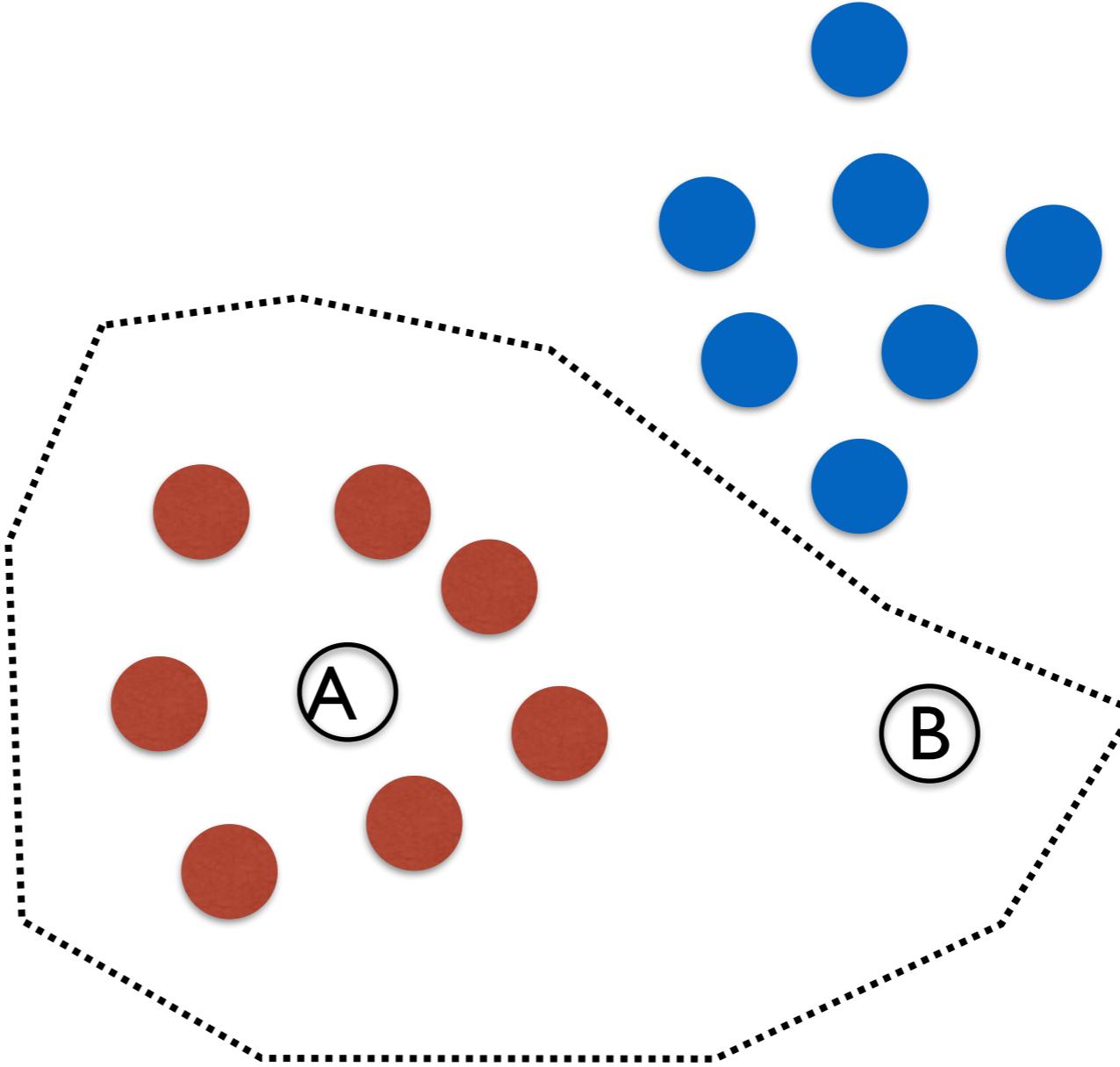
setting a “missingValue” constant

- We consider “missing” as another category for the feature and set some constant indicating that the value is missing such as “missingValue”.
- Not possible for numerical data. Does “0” mean the value was actually measured and turned out to be zero, or was it simply missing (possibly non-zero) in the dataset?
- Does not actually solve anything!

Replace with the mean

- Compute the mean of the available feature values for the entire training dataset and replace the missing values by this “sample” mean.
- This might be a good option if the data points with missing values are representative samples in the dataset.
- But if those data points are outliers this method is inaccurate

Outliers



It might be OK to say that point A is also red (the mean colour of the points around A) but it might not be accurate to say B is also red. B is an outlier belonging to neither red nor blue classes

Predicting missing values

- We can train a new classifier to first predict the missing values in data instances and then train a second classifier to predict the target class using all (original + missing values predicted) the data points.
- How is this possible?

Predicting missing values

$s_1=(\text{height}=170, \text{weight}=60)$

$s_2=(\text{height}=169, \text{weight}=60.1)$

$s_3=(\text{height}=171, \text{weight}=59.9)$

$s_4=(\text{height}=150, \text{weight}=40)$

$s_5=(\text{height}=155, \text{weight}=50)$

$s_6=(\text{height}=160, \text{weight}=55)$

$s_7=(\text{height}=168, \text{weight}=?)$

Guess the weight of s_7 ?

Accept missing values

- Just leave the data points with missing values as they are, and let the algorithm (eg. classifier) deal with the missing values in an appropriate manner
- The classifier might first try to come up with a rule to classify data without using features that have missing values. If it can do so with high accuracy, then we are fine. Nothing to worry about missing values.

Noisy values

- By “noisy data” we mean random errors scattered in the data
 - eg: due to inaccurate recording, data corruption
- Why is it a problem?
 - Overfitting:
 - If we assume the noisy values are correct values then we will learn classifiers to predict the noise as well. This could not be possible because it is “random” noise OR we might end up learning a classifier that learns “too much” from the train data and does not generalize to test data
 - This phenomenon is called “over-fitting” (fitting too much to the train data such that the model does not work well outside the give train dataset)
- Some noise will be very obvious:
 - data has incorrect type (string in numeric attribute)
 - data is very dissimilar to all other entries (10 in a feature otherwise in the range [0,1])
- Some incorrect values might not be obvious
 - eg. typing 0.52 instead of 0.25

Solutions to noisy values

- Manual inspection and removal
- Use clustering on the data to find instances or features that lie outside the main clusters (outlier detection) and remove them
- Use linear regression to determine the function, then remove those that lie from the predicted value
- Ignore all values that occur below a certain frequency threshold
 - effective for detecting misspellings in text
 - If noisy data points can be identified and removed, we can apply missing value techniques to fill the missing features.

Data normalization/canonicalization

- The same name can refer to different entities (namesake disambiguation problem)
 - Jim Clark (netscape ceo vs. F1 champion)
 - I had a mac (Apple Mac vs. McDonald burger)
- The same entity can be referred to by different names
 - Will Smith (fresh prince)
- How to normalize/resolve the different names to the proper entity?
 - word sense disambiguation problem

Redundant values

- What if we have multiple copies of the same data point?
 - eg. crawled data from mirror site
- Some machine learning algorithms can get adversely affected by duplicate data
 - Let us assume that seeing the word “advert” in an email is a good indicator that it is spam. We would like to compute the probability $P(\text{spam}|\text{advert})$
 - $P(\text{spam}|\text{advert}) = x / y$
 - x = no. of times we saw “advert” in spam emails
 - y = no of times we saw “advert” in an email
 - if we doubly count spam emails with the word “advert” then $P(\text{spam}|\text{advert}) = 2x / (x + y)$

Quiz

- Show that if we doubly count the spam emails that contain the word “advert” in our previous example, the probability $P(\text{spam}|\text{advert})$ becomes over-estimated.

Over-fitting vs. Under-fitting

- If a model M , trained on train data $D(\text{train})$ performs well on $D(\text{train})$, but poorly on a separate test dataset $D(\text{test})$, then it is likely that M is over-fitting to $D(\text{train})$
- Typically you will see 90-99% accuracy on $D(\text{train})$ and 40-60% accuracy on $D(\text{test})$ in the case of binary classification on balanced (equal no. of positive and negative) datasets
- This is because M has more than required parameters that it can “fit” to $D(\text{train})$ (too much flexibility), and it fits all of those on $D(\text{train})$, generalizing poorly to $D(\text{test})$
- Under-fitting is on the other hand is the situation where you get poor performance on $D(\text{train})$ because your model is not sufficiently “fitted” to the train data.

Solutions to under-fitting

- Learning has not converged
 - Let the training proceed for more iterations
- Your feature space is too small/inadequate
 - Implement more/better features
- Your train data is bad/noisy/missing values
 - Cleanse/re-annotate train data
- Your algorithm is not training well
 - select a different training algorithm

Solutions to Over-fitting

- Reduce the flexibility of your model
 - Regularization (we will discuss in detail when we learn logistic regression)
- Early stopping
 - Premature termination of training to prevent parameter overfitting
- Dropout
 - Randomly tune only half of your parameters on any given training instance
 - Known to be effective when training deep neural networks
 - We will discuss dropout in detail in our lecture on deep learning.

Slide 14 - Quiz: Linear independence

You need to prove that \mathbf{v} cannot be expressed as \mathbf{a} and \mathbf{b} and show the contradiction.

Using coefficients λ and μ , when you multiply these coefficients into \mathbf{a} and \mathbf{b} respectively, you should get \mathbf{v} .

$$\mathbf{v} = (1, 2, -3, 4)^T$$

$$\mathbf{a} = (1, 1, 0, 2)^T$$

$$\mathbf{b} = (-1, -2, 1, 1)^T$$

Use the coefficients and take the sum:

$$\mathbf{v} = \mathbf{a} + \mathbf{b}$$

$$1 = (1 \times \lambda) + (-1 \times \mu) = \lambda - \mu$$

$$2 = (1 \times \lambda) + (-2 \times \mu) = \lambda - 2\mu$$

$$-3 = (0 \times \lambda) + (1 \times \mu) = \mu$$

$$4 = (2 \times \lambda) + (1 \times \mu) = 2\lambda + \mu$$

$$\text{So, } \mu = -3$$

Now find λ , plug $\mu = -3$ into the above equations:

$$4 = 2\lambda + \mu$$

$$4 = 2\lambda + -3$$

$$3 + 4 = 2\lambda$$

$$7 = 2\lambda$$

$$\lambda = 7/2 = 3.5$$

$$\lambda = 3.5$$

Plug in values for λ and μ to check for contradictions:

$$2 = \lambda - 2\mu$$

$$2 = 3.5 - (2 \times -3) = 9.5$$

This is a contradiction since 9.5 does not equal 2. So \mathbf{v} cannot be expressed as a linear combination of \mathbf{a} and \mathbf{b} .

Slide 16 - Quiz

To find the ranks for these matrices **A** and **B**. Choose any row or column and try to make any elements 0.

Matrix A

$$\begin{array}{ccc} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{array}$$

3rd row are all 0's, so rank should be 2 since you can completely omit this row from matrix. So the number of linearly independent rows for matrix A is 2, so the rank is 2.

Matrix B

$$\begin{array}{ccc} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{array}$$

Eliminate elements in 1st column under 1st element.

Add 2 to all elements in 2nd row to get 0 for 1st element in row 2 and multiply by elements in row 1
 $r_2 + (2r_1)$:

$$-2 + (2 \times 1) = 0$$

$$-3 + (2 \times 2) = 1$$

$$1 + (2 \times 1) = 3$$

Subtract 3 from all elements in 3rd row to get 0 for 1st element in row 3 and multiply by elements in row 1

$r_3 - (3r_1)$:

$$3 - (3 \times 1) = 0$$

$$5 - (3 \times 2) = -1$$

$$0 - (3 \times 1) = -3$$

$$\begin{array}{ccc} 1 & 2 & 1 \\ 0 & 1 & 3 \\ 0 & -1 & -3 \end{array}$$

Eliminate elements in 2nd column under 2nd element.

Add 1 to get 0 for 2nd column under 2nd element and multiply by elements in row 2 (leave column 1 alone)

$r3 + (1 \times r2)$:

$$-1 + (1 \times 1) = 0$$

$$-3 + (1 \times 3) = 0$$

$$\begin{array}{ccc} 1 & 2 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 0 \end{array}$$

So row 3 has 0s, so rank of matrix is therefore 2.

Slide 19 - Quiz

Find the eigenvalues and the corresponding eigenvectors of A.

$$\begin{array}{cc} 4 & 2 \\ 1 & 3 \end{array}$$

Eigenvalues of A are the roots of the characteristic equation $\det(A - \lambda I) = 0$

$$\begin{aligned} (4 - \lambda) &\quad 2 \\ 1 &\quad (3 - \lambda) = 0 \\ \therefore (4 - \lambda) \times (3 - \lambda) - 2 \times 1 &= 0 \\ \therefore (12 - 7\lambda + \lambda^2) - 2 &= 0 \\ \therefore (\lambda^2 - 7\lambda + 10) &= 0 \\ \therefore (\lambda - 2)(\lambda - 5) &= 0 \\ \therefore (\lambda - 2) = 0 \text{ or } (\lambda - 5) &= 0 \end{aligned}$$

\therefore The eigenvalues of the matrix A are given by $\lambda=2, 5$

Using these values you can get the eigenvectors which should be:

$$[-1, 1]^T \text{ where } \lambda=2$$

$$[2, 1]^T \text{ where } \lambda=5$$

Lecture 2 – Mathematical Preliminaries – Quiz Solutions

Slide 6 - Quiz

Find $x + y$

Just add the corresponding dimensions:

$$1+3=4 \quad 2+2=4 \quad 3+1=4$$

Find $x \otimes y$

To find the element-wise product of x and y just multiply them together:

$$1 \times 3 = 3 \quad 2 \times 2 = 4 \quad 3 \times 1 = 3$$

Find $x^T y$

To find the inner product of x and y just multiply them together and then add them up:

$$1 \times 3 = 3 \quad 2 \times 2 = 4 \quad 3 \times 1 = 3$$

$$3 + 4 + 3 = 10$$

Find xy^T

Find the outer product of x and y :

$$1 \times 3 = 3 \quad 1 \times 2 = 2 \quad 1 \times 1 = 1$$

$$2 \times 3 = 6 \quad 2 \times 2 = 4 \quad 2 \times 1 = 2$$

$$3 \times 3 = 9 \quad 3 \times 2 = 6 \quad 3 \times 1 = 3$$

So now you have a 3 by 3 matrix for the outer product:

$$\begin{matrix} 3 & 2 & 1 \\ 6 & 4 & 2 \\ 9 & 6 & 3 \end{matrix}$$

Slide 8 - Quiz

Compute A+B:

Just add the corresponding elements together:

$$1+0=1 \quad 2+1=3 \quad 3+0=3 \quad 4+1=5 \quad 5+2=7 \quad 6+3=9 \quad 7+1=6 \quad 8+0=8 \quad 9+1=10$$

$$\begin{matrix} 1 & 3 & 3 \\ 5 & 7 & 9 \\ 6 & 8 & 10 \end{matrix}$$

Compute B+A:

This should be the same result as A+B above.

$$\begin{matrix} 1 & 3 & 3 \\ 5 & 7 & 9 \\ 6 & 8 & 10 \end{matrix}$$

Compute AB:

Multiply the 1st row in matrix A by the 1st column in matrix B (inner product between the 2 vectors), then to get the 1st element you add them up:

$$1\text{st element: } 1 \times 0 = 0 \quad 2 \times 1 = 2 \quad 3 \times -1 = -3 \quad 0 + 2 + -3 = -1$$

To get the 2nd element you multiply the 1st row in matrix A by the 2nd column in matrix B, then add:

$$2\text{nd element: } 1 \times 1 = 1 \quad 2 \times 2 = 4 \quad 3 \times 0 = 0 \quad 1 + 4 + 0 = 5$$

$$3\text{rd element: } 1 \times 0 = 0 \quad 2 \times 3 = 6 \quad 3 \times 1 = 3 \quad 0 + 6 + 3 = 9$$

Same again for the 2nd row in matrix A multiplied by the columns in matrix B:

$$1\text{st element: } 4 \times 0 = 0 \quad 5 \times 1 = 5 \quad 6 \times -1 = -6 \quad 0 + 5 + -6 = -1$$

$$2\text{nd element: } 4 \times 1 = 4 \quad 5 \times 2 = 10 \quad 6 \times 0 = 0 \quad 4 + 10 + 0 = 14$$

$$3\text{rd element: } 4 \times 0 = 0 \quad 5 \times 3 = 15 \quad 6 \times 1 = 6 \quad 0 + 15 + 6 = 21$$

3rd row of A multiplied by the 1st 2nd and 3rd columns of B:

$$1\text{st element: } 7 \times 0 = 0 \quad 8 \times 1 = 8 \quad 9 \times -1 = -9 \quad 0 + 8 + -9 = -1$$

$$2\text{nd element: } 7 \times 1 = 7 \quad 8 \times 2 = 16 \quad 9 \times 0 = 0 \quad 7 + 16 + 0 = 23$$

$$3\text{rd element: } 7 \times 0 = 0 \quad 8 \times 3 = 24 \quad 9 \times 1 = 9 \quad 0 + 24 + 9 = 33$$

So your matrix for AB should now be:

$$\begin{matrix} -1 & 5 & 9 \\ -1 & 14 & 21 \\ -1 & 23 & 33 \end{matrix}$$

Compute BA:

Multiply the 1st row in matrix B by the 1st column in matrix A and then add them up:

1st row 1st element:	0x1=0	1x4=4	0x7=0	0+4+0=4
1st row 2nd element:	0x2=0	1x5=5	0x8=0	0+5+0=5
1st row 3rd element:	0x3=0	1x6=6	0x9=0	0+6+0=6
2nd row 1st element:	1x1=1	2x4=8	3x7=21	1+8+21=30
2nd row 2nd element:	1x2=2	2x5=10	3x8=24	2+10+24=36
2nd row 3rd element:	1x3=3	2x6=12	3x9=27	3+12+27=42
3rd row 1st element:	-1x1=-1	0x4=0	1x7=7	-1+0+7=6
3rd row 2nd element:	-1x2=-2	0x5=0	1x8=8	-2+0+8=6
3rd row 3rd element:	-1x3=-3	0x6=0	1x9=9	-3+0+9=6

So, your matrix for BA should be:

$$\begin{matrix} 4 & 5 & 6 \\ 30 & 36 & 42 \\ 6 & 6 & 6 \end{matrix}$$

You can see the computations for AB and BA are not equal. So the matrix product is not commutative in general since they don't give you the same results.

Slide 10 - Computing the inverse of a 2x2 matrix and Slide 12 – Matrix Inversion

You basically multiply your 2 by 2 matrix by its inverse.

Matrix A

$$\begin{matrix} 1 & 2 \\ -2 & 1 \end{matrix}$$

Use this formula:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{\det \begin{pmatrix} a & b \\ c & d \end{pmatrix}} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

$$= \frac{1}{\det \begin{pmatrix} 1 & 2 \\ -2 & 1 \end{pmatrix}} \begin{pmatrix} 1 & -2 \\ -(-2) & 1 \end{pmatrix}$$

Find the matrix determinant according to formula:

$$\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$$

$$= 1 \cdot 1 - 2(-2) = 5$$

Multiply $\frac{1}{5}$ by your inverse matrix:

$$\frac{1}{5} \begin{pmatrix} 1 & -2 \\ -(-2) & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{5} & \frac{-2}{5} \\ \frac{2}{5} & \frac{1}{5} \end{pmatrix}$$

Note that you can check this answer is correct by performing matrix multiplication AA^{-1} (see slide 9 in lecture 2) and the result should be according to this:

$$\text{Identity Matrix } I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Remember it says all diagonal elements are set to 1 and non-diagonal elements are set to 0. So to check this we can multiply the matrix A by its inverse matrix A^{-1} which is what we already did above:

$$A = \begin{pmatrix} 1 & 2 \\ -2 & 1 \end{pmatrix}$$

$$A^{-1} = \begin{pmatrix} \frac{1}{5} & \frac{-2}{5} \\ \frac{2}{5} & \frac{1}{5} \end{pmatrix}$$

So, then if you multiply rows by columns and then add them together:

$$\text{1st row 1st element: } 1 \times \frac{1}{5} + 2 \times \frac{2}{5} \quad \text{1st row 2nd element: } 1 \times \frac{-2}{5} + 2 \times \frac{1}{5}$$

$$\text{2nd row 1st element: } -2 \times \frac{1}{5} + 1 \times \frac{2}{5} \quad \text{2nd row 2nd element: } -2 \times \frac{-2}{5} + 1 \times \frac{1}{5}$$

$$\frac{1}{5} + \frac{4}{5} \quad \frac{-2}{5} + \frac{2}{5}$$

$$\frac{-2}{5} + \frac{2}{5} \quad \frac{4}{5} + \frac{1}{5}$$

So then you have proved the answer to be correct since it's the same as the identity matrix:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Classifier Evaluation



We have a classifier/model/system...

- How **good** is it?
- Levels of **goodness**
 - Absolute goodness: When we run our trained model on the “wild” it does what we expect it to do
 - no way of knowing *before* we deploy the model and when we do know, too late!
 - Relative goodness: We have a small representative sample of test data
 - We compare the output produced by our classifier on this test dataset (gold standard) and measure how well it resembles the labels in the dataset.

Gold Standard

- A dataset that we use for evaluation purpose. Also known as test data.
- Each test instance in the test data have their correct labels annotated.
- Numerous measures exist (as we shortly see) to **compare** the predicted labels by the trained classifier and actual (**target**) labels in the test dataset.
- Never train on test data!!!

Confusion Matrix

	Actual YES(+)	Actual NO(-)
Predicted YES(+)	True Positives (TP)	False Positives (FP)
Predicted NO(-)	False Negatives (FN)	True Negatives (TN)

Definitions

- True Positive
 - We predicted as positive and it is indeed positive
- True Negative
 - We predicted as negative and it is indeed negative
- False Positive
 - We predicted as positive but it turns out to be negative
- False Negative
 - We predicted as negative but it turns out to be positive

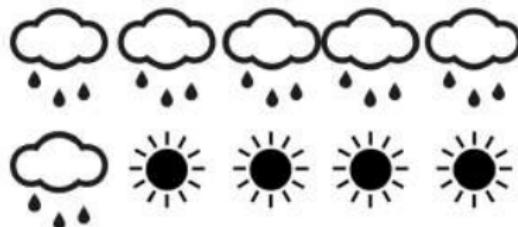
Detecting Cancer

- Let us assume we trained a classifier to detect cancer based on some features
- Predicting YES means we predict that the patient has cancer
- We predicted the patient as having cancer but further tests revealed that the patient does not have cancer.
 - False Positive
- We predicted the patient as not having cancer (so no further tests were done) but the patient died with cancer!
 - False Negative
- The moral of the story: FP and FN have very different importance in real-world data mining tasks.

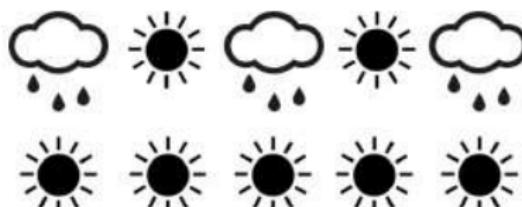
Evaluation Measures

- $(TP + TN) / (TP + TN + FP + FN)$
- Precision (P): Precision tells us about all the correct predictions out of total positive predictions
 - $P = TP / (TP + FP)$
- Recall (R): Recall tells us about how many instances were correctly classified out of all positive instances
 - $R = TP / (TP + FN)$
- F-score = $(2 \times \text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$

Quiz



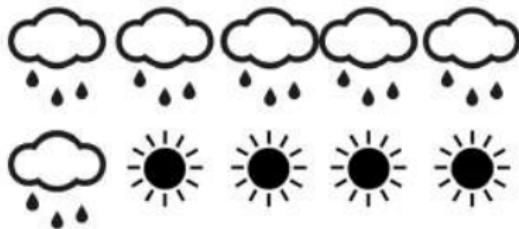
Actual Weather for 10 days



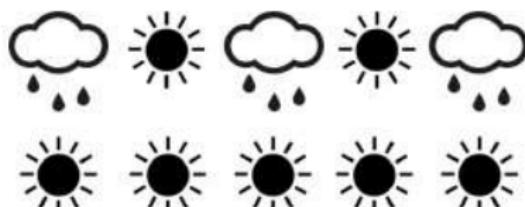
Model's prediction for 10 days

**Can you create a
confusion matrix and
compute Precision,
Recall and Accuracy?**

Quiz



Actual Weather for 10 days



Model's prediction for 10 days

	Actual (sunny)	Actual (rainy)
Predicted (sunny)	4 (TP)	3 (FP)
Predicted (rainy)	0 (FN)	3 (TN)

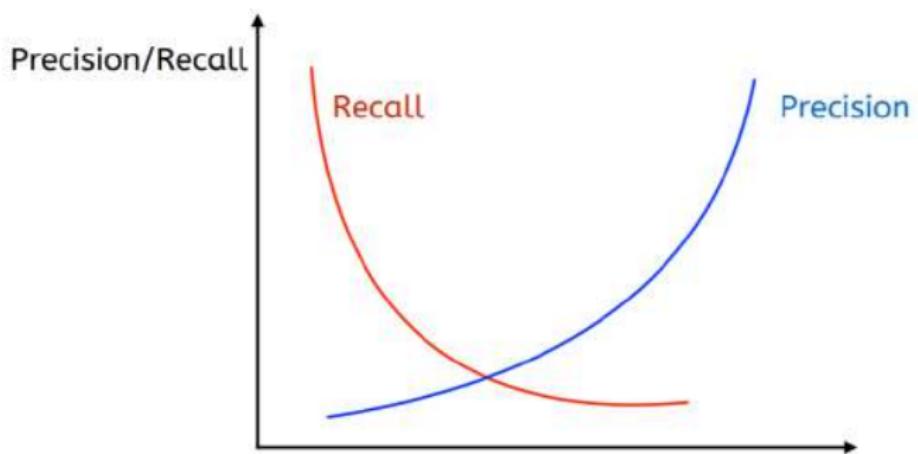
$$P = TP / (TP + FP) = 4 / 7 = 57\%$$

$$R = TP / (TP + FN) = 4 / 0 = 100\%$$

$$A = (TP + TN) / (TP + FP + FN + TN) = 70\%$$

$$F = (2 * 0.57 * 1) / (0.57 + 1) = 72\%$$

Precision-Recall Trade-off



By simply varying the threshold of our cancer detector we can get a high precision OR low recall system. There is a *trade-off*

8

Harmonic Mean

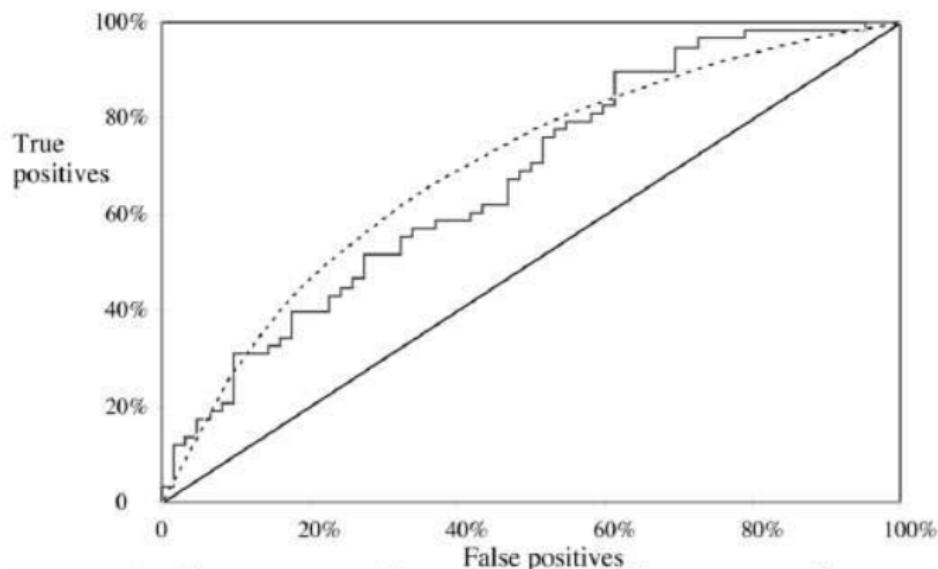
- Harmonic mean is a good way to compute an average when there is an inverse relation between two variables.
- F-score (F) is the harmonic mean between Precision (P) and Recall (R).

$$F = \frac{1}{\frac{\frac{1}{P} + \frac{1}{R}}{2}} = \frac{2PR}{P + R}$$

ROC Curves

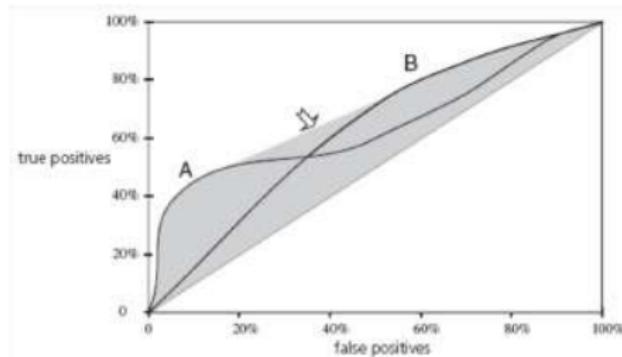
- Receiver Operating Characteristic
 - Originally a concept borrowed from signal processing
 - Tradeoff between hit rate and false alarm rate when trying to find real data in noisy channel.
- Plot true positive vertically, and false positives horizontally.
- The place to be is on the top left (high TP and low FP)
- Generate a series of models operating at various threshold values, measure TP and FP for each model to plot ROC curve
- We can generate a smooth ROC curve by the of cross-validation (discussed later) by generating a curve for each fold, and then averaging them.

ROC



Comparing Classifiers using ROC

- We can also plot two curves on the same chart, each generated from different classifiers. This lets us see at which point it is better to use one classifier rather than the other.
- By using both A and B classifiers with appropriate weightings, it is possible to get at points in between the two peaks



Evaluation measures for regression

- What if we are predicting real numbers instead of class labels?
 - This setting is called regression
- It is too harsh to consider non-exact matches as incorrect predictions
- For an instance x , our trained model predicts the output to be 1.001, but the actual value is 1.
 - Is this a correct prediction or an incorrect prediction
 - We are off by just 0.001

Evaluation Measures for Regression

Performance measure	Formula
mean-squared error	$\frac{(p_1 - a_1)^2 + \dots + (p_n - a_n)^2}{n}$
root mean-squared error	$\sqrt{\frac{(p_1 - a_1)^2 + \dots + (p_n - a_n)^2}{n}}$
mean absolute error	$\frac{ p_1 - a_1 + \dots + p_n - a_n }{n}$
relative squared error	$\frac{(p_1 - a_1)^2 + \dots + (p_n - a_n)^2}{(a_1 - \bar{a})^2 + \dots + (a_n - \bar{a})^2}, \text{ where } \bar{a} = \frac{1}{n} \sum_i a_i$
root relative squared error	$\sqrt{\frac{(p_1 - a_1)^2 + \dots + (p_n - a_n)^2}{(a_1 - \bar{a})^2 + \dots + (a_n - \bar{a})^2}}$
relative absolute error	$\frac{ p_1 - a_1 + \dots + p_n - a_n }{ a_1 - \bar{a} + \dots + a_n - \bar{a} }$
correlation coefficient	$\frac{S_{PA}}{\sqrt{S_P S_A}}, \text{ where } S_{PA} = \frac{\sum_i (p_i - \bar{p})(a_i - \bar{a})}{n-1},$ $S_P = \frac{\sum_i (p_i - \bar{p})^2}{n-1}, \text{ and } S_A = \frac{\sum_i (a_i - \bar{a})^2}{n-1}$

Root Mean Square Error

- A popular measure for evaluating numerical (real/ordinal) regression models is the *root mean square error* (RMSE) defined as follows:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

The diagram illustrates the components of the RMSE formula. The term $\frac{1}{n}$ is labeled 'mean'. The term $(y_i - \hat{y}_i)^2$ is labeled 'error'. The entire square root expression is labeled 'square'. A curved arrow labeled 'root' points to the square root symbol.

Robust Evaluation

Case Study

- We have a single train dataset. We do not have a separate test dataset. We want to evaluate a classification algorithm as **reliably** as possible using this train dataset. How should we go about this?

A Bad Idea

- Train using ALL the train data
- Evaluate on ALL the train data
- Likely to overfit!
 - We might get very impressive train performance but we have no idea how this classification model is going to perform on UNSEEN test data.
 - If the classifier simply REMEMBERS every instance in the train dataset, it will get 100% accuracy under this evaluation scheme
- Bad idea!

Cross-Validation

- We already discussed validation datasets under *how to set the hyper-parameters of a classifier* (see k-NN lecture notes)
- set aside a portion of **train** data for validation purposes

Cross-Validation (contd.)

- For example, use 1/5-th of train data for validation. This is called a **fold** of the dataset.
 - Train using the remaining 4/5-th and evaluate on the held-out 1/5th.
 - Repeat this process 5 times, each time selecting a different fold.
 - Evaluate the performance (using any evaluation measure we discussed so far) on the held-out data (1/5-th not used in training)
 - Take the average of the five numbers
 - This is called **5-fold cross-validation**

Cross-Validation (contd.)

- In N -fold cross-validation, we split the dataset into N equal partitions (folds), repeat the process N times, and take the average of the N evaluation measure values.

Issues

- Cross-validation is a trade-off between speed and the train dataset size
- Increase the number of folds:
 - More data to train from
 - Better estimates (averaging over many folds)
 - Slow (many re-trains)

Issues

- If we are performing cross-validation to set hyper-parameters, then it could be the case that a different value of the hyper-parameter is producing the best results in different folds! No clear winner to select
 - Select the best fold (max instead of avg)

Leave-One-Out Cross-Validation

- Leave only one of the train instances in each fold of validation
- train dataset size = $(N-1)$ instances
- validation dataset size = 1 instance
- Repeat the process N ($=$ dataset size) times
- Pros
 - We have lots of data to train from
- Cons
 - Very slow in practice because we have to re-train our classifier $N-1$ number of times

Clustering



Outline

- Why cluster data?
- Clustering as unsupervised learning
- Clustering algorithms
 - k-means, k-medoids
 - agglomerative clustering
 - Brown's clustering
 - Spectral clustering
- Cluster evaluation measures
 - Purity
 - Normalised Mutual Information
 - Rand Index
 - B-CUBED
 - Precision, Recall, F-score

Why cluster data?

- Data mining has two main objectives:
 - Prediction: classification, regression etc.
 - Description: pattern mining, rule extraction, visualisation, *clustering*
- Clustering is:
 - Unsupervised learning
 - no label data is required (consider classification algorithms we discussed so far in the lectures which are supervised algorithms)

Unsupervised Learning

- Supervised learning
 - labels for training instances are provided
- Unsupervised learning
 - no labels for training instances are provided
- Semi-Supervised learning
 - Both labeled and unlabeled training instances are provided
- What can we learn about training data if we do not have any labels?
 - The similarity and distribution of the features can still be learnt and this can be used to create rich feature spaces for supervised learning (if required)

Clustering: Example

Headlines

[More Headlines](#)

Coronavirus: Boris Johnson announces plan for 'delay' phase

Daily Mail · 1 hour ago



- **Coronavirus: Boris Johnson to hold emergency Cobra meeting**

BBC South East Wales · 7 hours ago

- **BREAKING: UK cases of coronavirus rise to 319**

Sky News · 6 hours ago

- **Coronavirus brings a reminder of the Iron law of politics**

Financial Times · 4 hours ago · Opinion

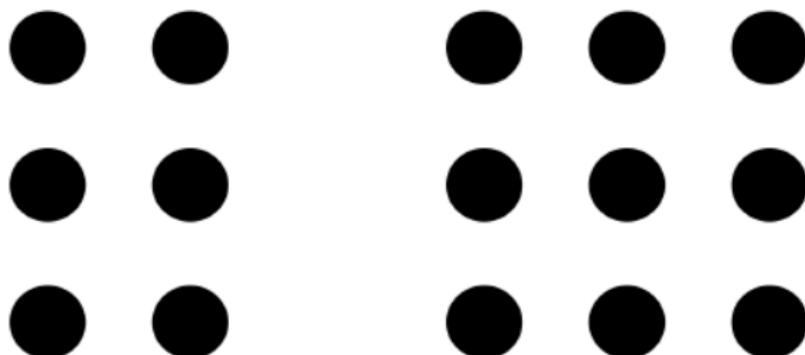
- **Nigel Farage: Yes, Protecting Us All from an Epidemic Should be Prioritized Over the Economy | Opinion**

Newsweek · 2 hours ago · Opinion

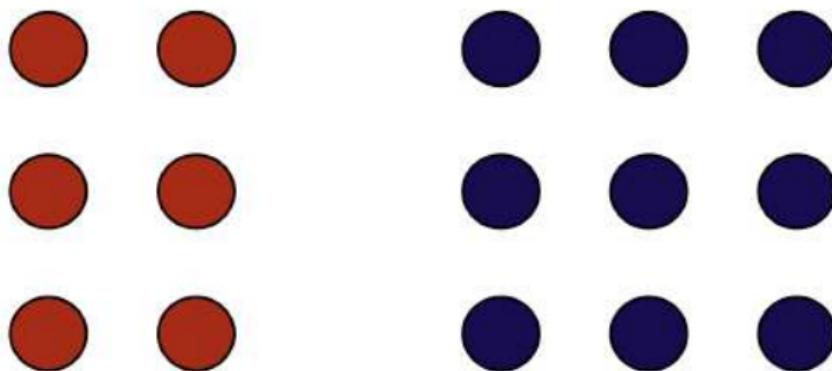
[View Full coverage](#)



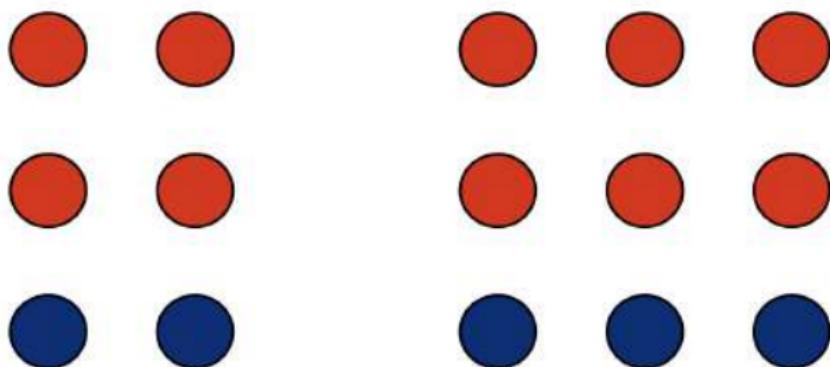
Quiz: Cluster the following data



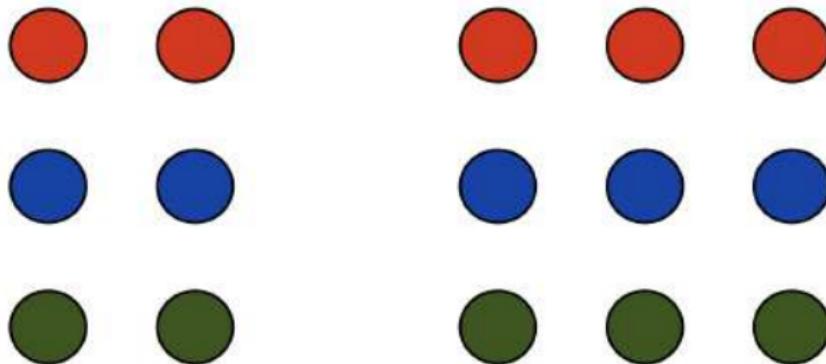
Quiz: Cluster the following data



Quiz: Cluster the following data



Quiz: Cluster the following data



How many clusters?

General Remarks

- A single dataset can be clustered into several ways
- There is no single right or wrong clustering
 - Simply different views of the same data
- how to measure the quality of clustering algorithm?
 - Two ways
 - Compare clusters produced by clustering algorithm against some reference (gold standard) set of clusters (**direct evaluation**)
 - Use the clusters for some other (eg. supervised learning) task and measure the difference in performance of the second task (**indirect evaluation**)

Clustering as Optimisation

- Given a dataset $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ of N instances represented as d dimensional real vectors ($\mathbf{x}_i \in \mathbb{R}^d$), partition these N instances into k clusters S_1, \dots, S_k such that some objective function $f(S_1, \dots, S_k)$ is minimised.
- Observations
 - k and f are given
 - f can be similarity between the clusters (good to create dissimilar clusters as much as possible), information gain, correlation and various other such goodness measures (heuristics)

Partitioning - k-means algorithm

$$\arg \min_{S_1, \dots, S_k} \sum_{i=1}^k \sum_{x_j \in S_i} \|x_j - \mu_i\|^2$$

We want to minimize the distance between data instances (x_j) and some cluster centres (μ_i)

$$f(S_1, \dots, S_k) = \sum_{i=1}^k \sum_{x_j \in S_i} \|x_j - \mu_i\|^2$$

This objective function is called the *within cluster sum of squares* (WCSS) objective

Partitioning - cluster centroids

$$\frac{\partial f(S_1, \dots, S_k)}{\partial \mu_i} = 0$$

$$\frac{\partial f(S_1, \dots, S_k)}{\partial \mu_i} = \sum_{\mathbf{x}_j \in S_i} 2(\mathbf{x}_j - \boldsymbol{\mu}_i)$$

$$\boldsymbol{\mu}_i = \frac{1}{|S_i|} \sum_{\mathbf{x}_j \in S_i} \mathbf{x}_j$$

Just compute the centroid (mean) of each cluster
and that will give you the cluster centers

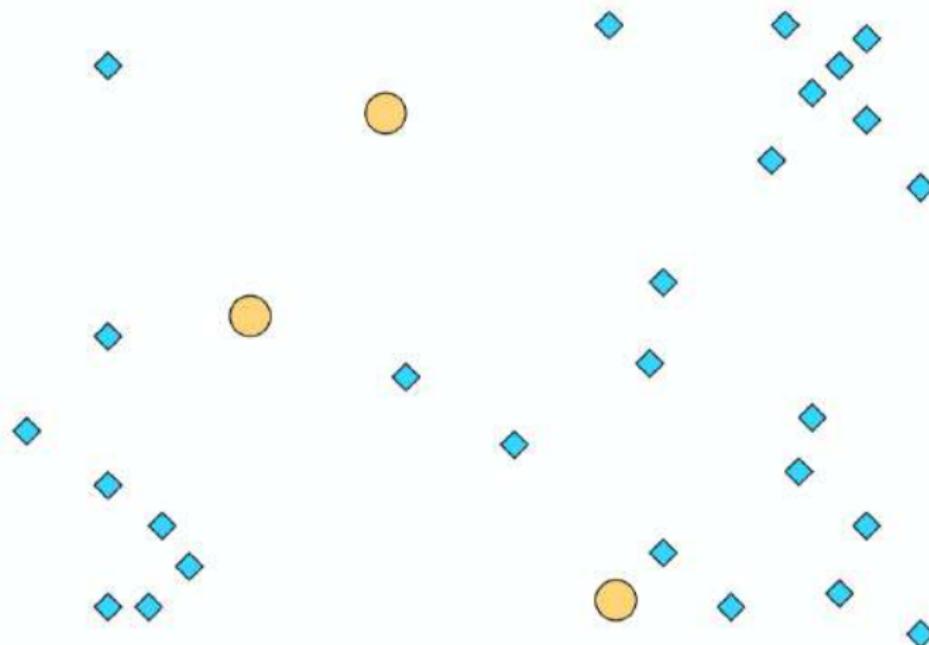
k-Means clustering

- Input

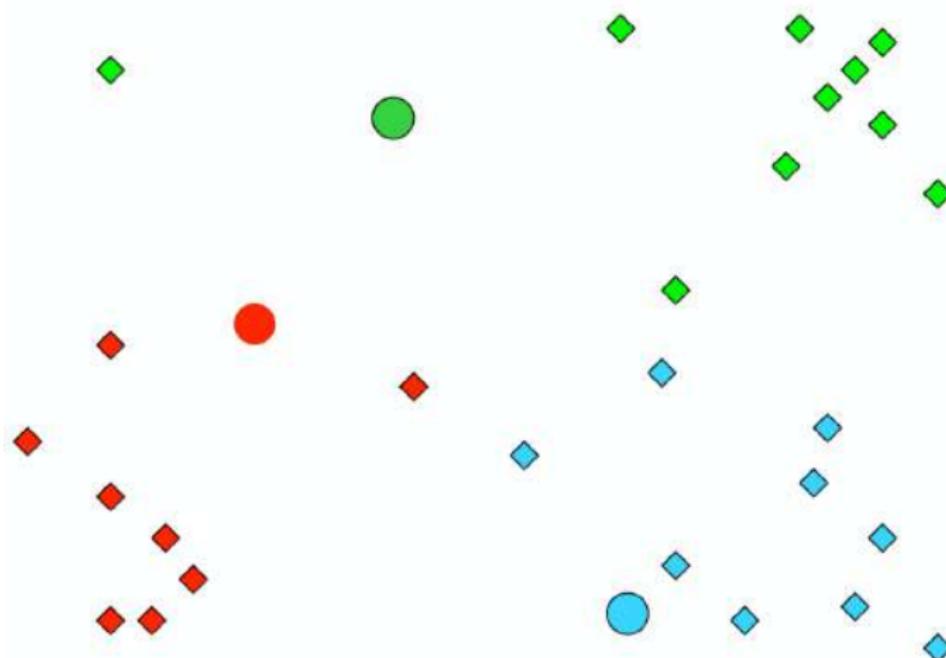
- The number of clusters k
 - Dataset $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ of N instances represented as d dimensional real vectors ($\mathbf{x}_i \in \mathbb{R}^d$)
- ➊ Set k instances from the dataset randomly (initial cluster means / centers)
 - ➋ Assign all other instances to the closest cluster centre
 - ➌ Compute the mean of each cluster
 - ➍ Until **convergence** repeat between steps 2 and 3

convergence = no instances have moved among clusters
(often after a fixed number of iterations specified by the user)

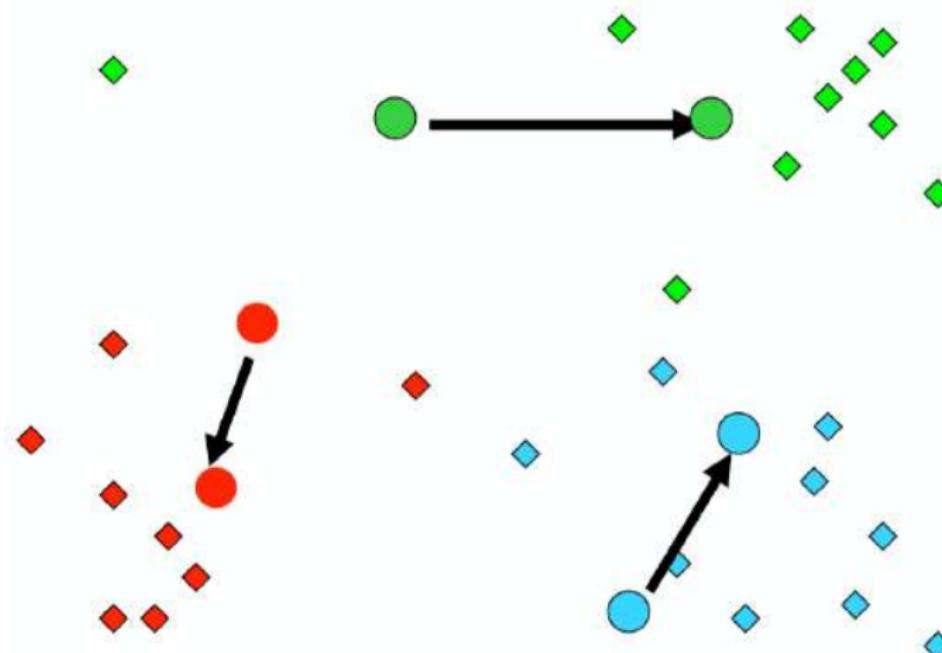
k-means clustering



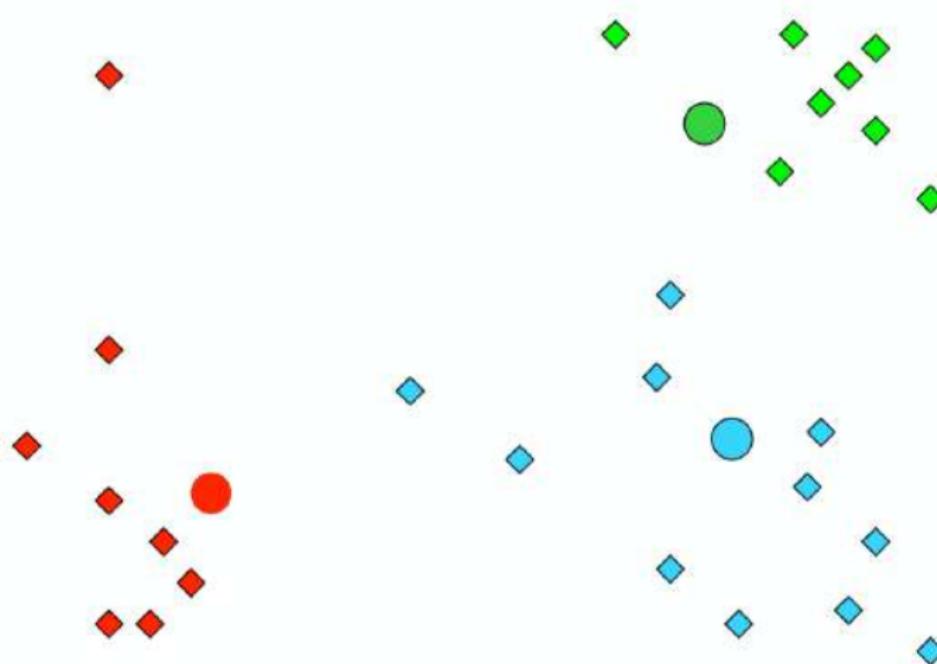
k-means clustering



k-means clustering



k-means clustering



Quiz: k-means clustering

- Given five data points: $\{(0, 0), (1, 0), (1, 1), (0, 1), (-1, 0)\}$
- Create two clusters $K = 2$: (c_1 and c_2)
- Choose $x_2 = (1, 0)$ and $x_3 = (1, 1)$ as initial centroids
- Use Euclidean Distance as the similarity metric

Quiz: k-means clustering (soln)

		$c_1 (1,0)$	$c_2 (1,1)$	Assignment
x_1	0,0	$\sqrt{(0-1)^2 + (0-0)^2} = \sqrt{1}$	$\sqrt{(0-1)^2 + (0-1)^2} = \sqrt{2}$	c_1
x_2	1,0	$\sqrt{(1-1)^2 + (0-0)^2} = \sqrt{0}$	$\sqrt{(1-1)^2 + (0-1)^2} = \sqrt{1}$	c_1
x_3	1,1	$\sqrt{(1-1)^2 + (1-0)^2} = \sqrt{1}$	$\sqrt{(1-1)^2 + (1-1)^2} = \sqrt{0}$	c_2
x_4	0,1	$\sqrt{(0-1)^2 + (1-0)^2} = \sqrt{2}$	$\sqrt{(0-1)^2 + (1-1)^2} = \sqrt{1}$	c_2
x_5	-1,0	$\sqrt{(-1-1)^2 + (0-0)^2} = \sqrt{4}$	$\sqrt{(-1-1)^2 + (0-1)^2} = \sqrt{5}$	c_1

- $c_1 = \{x_1, x_2, x_5\}; c_2 = \{x_3, x_4\}$
- $c_1 = \{(0, 0), (1, 0), (-1, 0)\}; c_2 = \{(1, 1), (0, 1)\}$
- $\mu_{c_1} = (0, 0); \mu_{c_2} = (0.5, 1)$
- computing clusters using new μ gives the same clusters

Evaluating Clustering - Purity

- Purity is an external evaluation criterion for cluster quality
- It gives the percentage of total number of items that were classified correctly.
- range [0, 1]

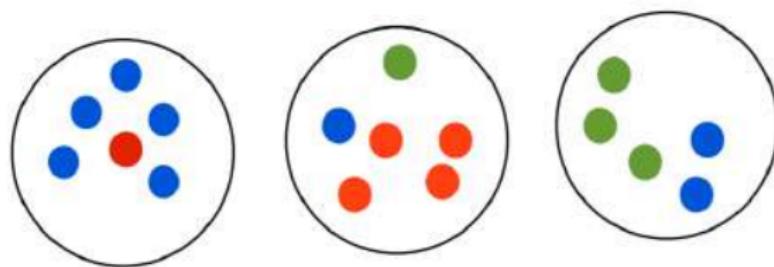
Evaluating Clustering - Purity

- Let us assume we have a set $\Omega = \{\omega_1, \dots, \omega_K\}$ clusters for a set of classes $C = \{c_1, \dots, c_j\}$.
- Purity measures the ratio of the items that are in the cluster with the same class of its own.

$$\text{purity}(\Omega, C) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j|$$

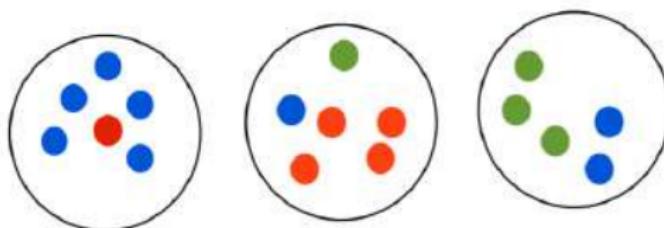
N is the number of items, ω_k is a cluster in Ω and c_j is the class which has the maximum count for cluster ω_k .

Purity



Quiz: Compute purity for this clustering.

Purity



Labels



$$\text{purity} = (5 + 4 + 3) / 17 = 12/17 = 0.71$$

Purity achieves its maximum value of 1 for singletons (each item is in a cluster containing only that single item)!

Obviously this is not good "clustering" and purity does not recognise this.

Purity

- bad clusterings have purity values close to 0
- perfect clustering has a purity of 1
- high purity is easy to achieve when the number of clusters is large
- particularly, purity is 1 if each item (singleton) gets its own cluster
- thus, purity cannot be used to trade off the quality of clustering against number of clusters.

Evaluating Clustering - NMI

- Let us assume we have a set $\Omega = \{\omega_1, \dots, \omega_K\}$ clusters for a set of classes $C = \{c_1, \dots, c_j\}$.
- Normalised Mutual Information (NMI) computes the ratio of information that we can know about the classes C given the clusters Ω to the averaged information that is contained in C and Ω .

Evaluating Clustering - NMI

- NMI is computed using:

$$\text{NMI}(\Omega, \mathcal{C}) = \frac{I(\Omega, \mathcal{C})}{[H(\Omega) + H(\mathcal{C})]/2}$$

where,

- Ω = set of clusters
- \mathcal{C} = set of classes
- $I(\Omega, \mathcal{C})$ = mutual information between Ω and \mathcal{C}
- $H(\cdot)$ = Entropy

Evaluating Clustering - NMI

- Mutual information $I(\Omega, \mathcal{C})$ is given by:

$$\begin{aligned} I(\Omega, \mathcal{C}) &= \sum_k \sum_j p(\omega_k \cap c_j) \log \left(\frac{p(\omega_k \cap c_j)}{p(\omega_k)p(c_j)} \right) \\ &= \sum_k \sum_j \frac{|\omega_k \cap c_j|}{N} \log \left(\frac{N|\omega_k \cap c_j|}{|\omega_k||c_j|} \right) \end{aligned}$$

where,

- $P(\omega_k)$ = probability of an object being in cluster ω_k
- $P(c_j)$ = probability of an object being in class c_j
- $P(\omega_k \cap c_j)$ = probability of an object being in the intersection of ω_k and c_j

NMI - Mutual Information

- mutual information measures the amount of information by which our knowledge about the classes increases when we are told about what clusters are.
- minimum of mutual information is 0
 - clustering is random w.r.t class
 - knowing an item in a cluster does not give any information about what classes could be
- mutual information achieves maximum for clustering that perfectly recreates the classes

NMI - Mutual Information

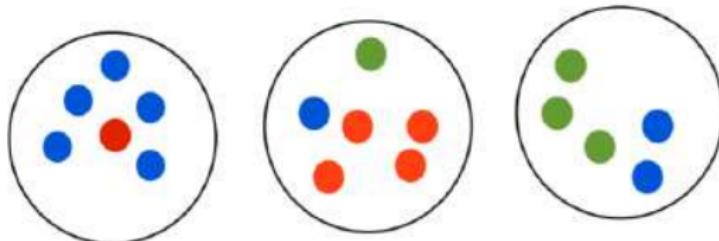
- clustering $K = N$ (one-document clusters) has maximum MI.
- thus MI has same problem of purity
- cannot penalise large cardinalities
- fewer clusters are better

Entropy

$$\text{NMI}(\Omega, \mathcal{C}) = \frac{I(\Omega, \mathcal{C})}{[H(\Omega) + H(\mathcal{C})]/2}$$

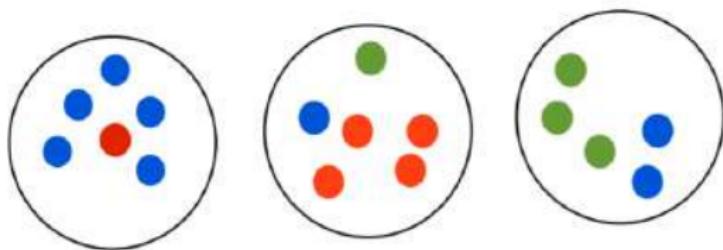
- the entropy function used in the denominator of NMI fixes this problem.
- entropy tends to increase with different number of clusters
- for example $H(\Omega)$ reaches its maximum $\log N$ for $K = N$, which ensures NMI is low for $K = N$
- NMI is always a number between 0 and 1

Calculating NMI for Clustering



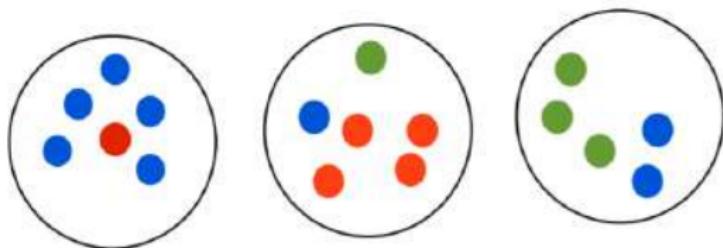
- Given $C = 3$ classes
- Let $c_1 = \text{Blue}$, $c_2 = \text{Red}$ and $c_3 = \text{Green}$
- Thus, $P(c_1) = \frac{8}{17}$, $P(c_2) = \frac{5}{17}$, $P(c_3) = \frac{4}{17}$
- Entropy of class: $H(C) = -\sum_{i=1}^3 P(c_j) \log P(c_j)$
 $= -[\frac{8}{17} \log \frac{8}{17} + \frac{5}{17} \log \frac{5}{17} + \frac{4}{17} \log \frac{4}{17}] = 1.055$

Calculating NMI for Clustering



- Likewise, $P(w_1) = \frac{6}{17}$, $P(w_2) = \frac{6}{17}$, $P(w_3) = \frac{5}{17}$
 $= -[\frac{6}{17} \log \frac{6}{17} + \frac{6}{17} \log \frac{6}{17} + \frac{5}{17} \log \frac{5}{17}] = 1.095$

Calculating NMI for Clustering



- $P(w_1 \cap c_1) = \frac{5}{17}, P(w_1 \cap c_2) = \frac{1}{17}, P(w_1 \cap c_3) = \frac{0}{17}$
- $P(w_2 \cap c_1) = \frac{1}{17}, P(w_2 \cap c_2) = \frac{4}{17}, P(w_2 \cap c_3) = \frac{1}{17}$
- $P(w_3 \cap c_1) = \frac{2}{17}, P(w_3 \cap c_2) = \frac{0}{17}, P(w_3 \cap c_3) = \frac{3}{17}$

Calculating NMI for Clustering

- Mutual information $I(\Omega, \mathcal{C})$ is given by:

$$\begin{aligned} I(\Omega, \mathcal{C}) &= \sum_k \sum_j p(\omega_k \cap c_j) \log \left(\frac{p(\omega_k \cap c_j)}{p(\omega_k)p(c_j)} \right) \\ &= \sum_k \sum_j \frac{|\omega_k \cap c_j|}{N} \log \left(\frac{N|\omega_k \cap c_j|}{|\omega_k||c_j|} \right) \end{aligned}$$

- substituting the values, we get $I(\Omega, \mathcal{C}) = 0.4496$
- Finally NMI is given by:

$$\text{NMI}(\Omega, \mathcal{C}) = \frac{I(\Omega, \mathcal{C})}{[H(\Omega) + H(\mathcal{C})]/2}$$

- thus, $NMI(\Omega, C) = \frac{0.4496}{1.055+1.095} = 0.4182$

NMI

- NMI is a good measure for determining the quality of clustering
- it is an external measure as we need class labels of instances to determine NMI
- Since it is normalised, NMI between clusterings having different number of clusters can be measured.

Rand Index (RI)

- RI is a metric used to evaluate the quality of clustering technique
- RI measures the percentage of decisions that are correct
- decision - assigning a pair of data points to a cluster
- Total number of decisions: $\frac{N(N-1)}{2}$, where N is the total number of data points
- RI is given by: $\frac{TP+TN}{TP+FP+TN+FN}$

Rand Index (RI)

- TP = No. of item pairs that are in the same cluster and belong to the same class
- FP = No. of item pairs that are in the same cluster but belong to different classes
- TN = No. of item pairs that are in different clusters and belong to different classes
- FN = No. of item pairs that are in different clusters but belong to the same class

Rand Index (RI)

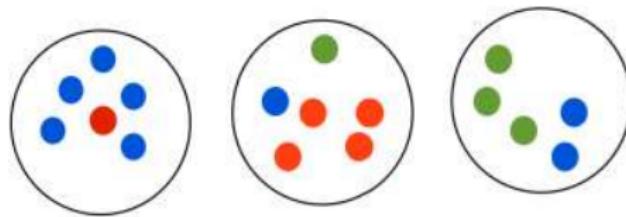
Contingency Table

contingency table	same cluster	different clusters
same class	TP	FN
different classes	FP	TN

$$RI = \frac{TP + TN}{TP + FP + TN + FN}$$

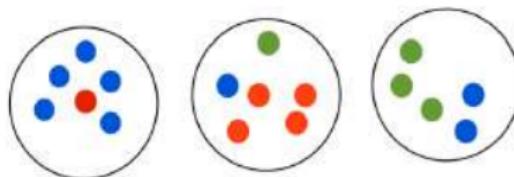
(accuracy of
the clustering)

Compute Rand Index (RI)



Quiz: Compute RI for this clustering.

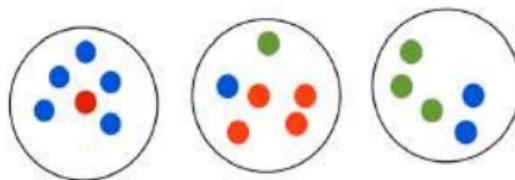
Compute Rand Index (RI)



- Three classes: blue, red, green
- Total items: 17
- Total number of pairs:

$$\frac{N(N - 1)}{2} = \frac{17(17 - 1)}{2} = 136 \quad (1)$$

Compute Rand Index (RI)



- To start, let us compute Total Positives = TP+FP

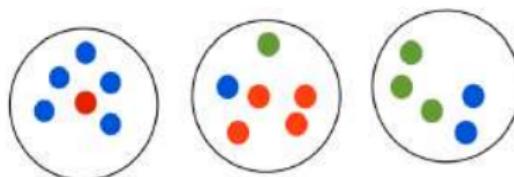
$$TP+FP = {}^6C_2 + {}^6C_2 + {}^5C_2$$

$${}^nC_r = \frac{n!}{r!(n-r)!}. \text{ Thus } {}^6C_2 = \frac{6!}{2!(6-2)!} = \frac{6 \times 5}{2} = 15;$$

$${}^5C_2 = \frac{5!}{2!(5-2)!} = \frac{5 \times 4}{2} = 10$$

$$TP + FP = 15 + 15 + 10 = 40$$

Compute Rand Index (RI)



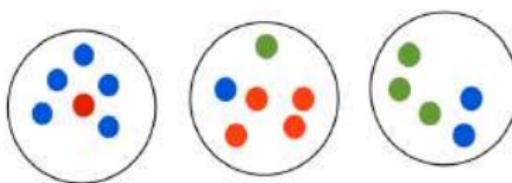
- To start, let us compute TP

$$TP = {}^5C_2 + {}^4C_2 + {}^3C_2 + {}^2C_2$$

$$TP = 10 + 6 + 3 + 1 = \mathbf{20}$$

- Thus, $FP = 40 - 20 = \mathbf{20}$

Compute Rand Index (RI)



- let us calculate negatives!

$$\text{Total Negatives} = \text{Total Pairs} - \text{Total Positives} (\text{TP} + \text{FP})$$

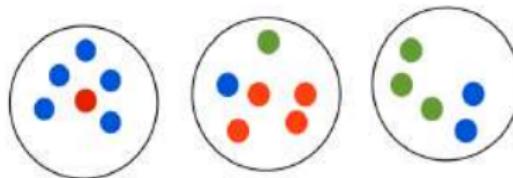
$$\text{Total Negatives} = 136 - 40 = 96$$

- FN is calculated by looking at pairs that should be grouped together but are not!

$$\text{FN} = [(3 \times 5) + (1 \times 2)] + (1 \times 4) + (1 \times 3) = \mathbf{24}$$

- TN = Total Negatives - FN = 96 - 24 = **72**

Compute Rand Index (RI)



		same cluster	different clusters
same class	20	24	
different classes	20	72	

$$\begin{aligned} \text{RI} &= (20+72) / (20+24+20+72) \\ &= 0.676 \end{aligned}$$

Evaluating Clustering - P/R/F

- We can use Precision (P), Recall (R), and F-measure (F) at to evaluate the accuracy of a clustering.
- For this purpose we must first create the contingency table as we did for RI and then compute P, R, F as follows

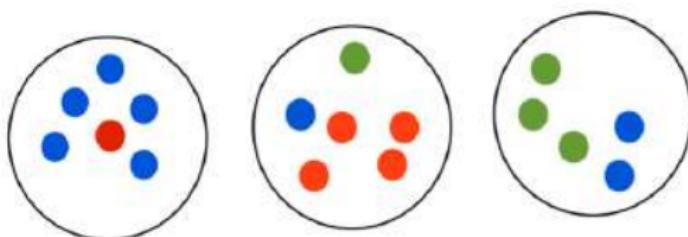
$$P = TP / (TP + FP)$$

$$R = TP / (TP + FN)$$

$$F = 2PR / (P + R)$$

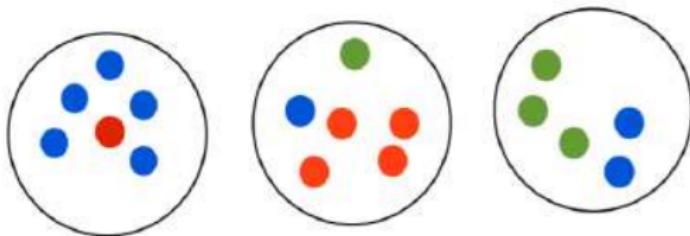
Ref: <https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html>

Evaluating Clustering - P/R/F



Quiz: Compute P/R/F for this clustering.

Evaluating Clustering - P/R/F



	same cluster	different clusters
same class	TP=20	FN=24
different classes	FP=20	TN=72

$$P = TP / (TP + FP) = 20 / (20+20) = 0.5$$

$$R = TP / (TP + FN) = 20 / (20 + 24) = 0.45$$

$$F = 2PR / (P + R) = 0.47$$

B-CUBED Measure

- Proposed in (Bagga B. Baldwin = B³)
 - A. Bagga and B. Baldwin. Entity-based cross document coreference resolution using the vector space model, In Proc. of 36th COLING-ACL, pages 79--85, 1998.
 - We would like to evaluate clustering without labelling any clusters.

$$\text{precision}(x) = \frac{\text{No. of items in } C(x) \text{ with } A(x)}{\text{No. of items in } C(x)}$$

$$\text{recall}(x) = \frac{\text{No. of items in } C(x) \text{ with } A(x)}{\text{Total no. of items with } A(x)}$$

$C(x)$: The ID of the cluster that x belongs to

$A(x)$: label of x

B-CUBED Measure

- Compute the average over all the items (instances) that appear in all clusters (N)

$$\text{Precision} = \frac{1}{N} \sum_{p \in \text{DataSet}} \text{Precision}(p)$$

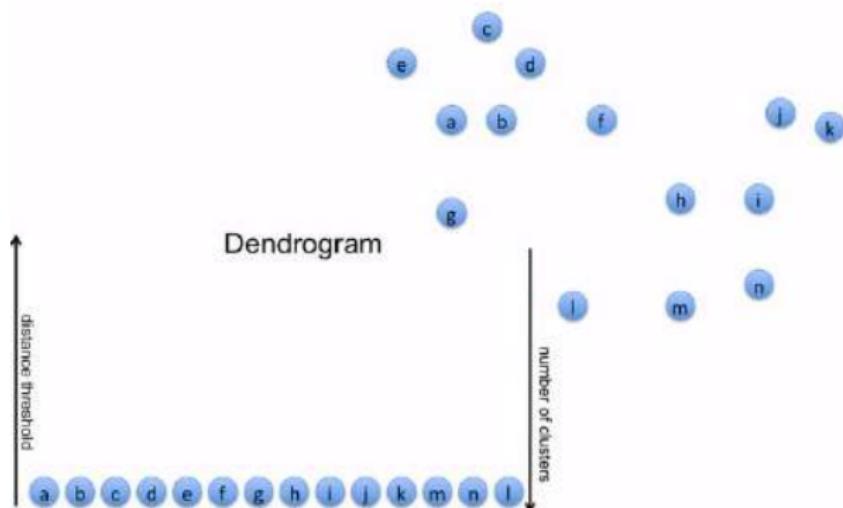
$$\text{Recall} = \frac{1}{N} \sum_{p \in \text{DataSet}} \text{Recall}(p)$$

$$F\text{-Score} = \frac{1}{N} \sum_{p \in \text{DataSet}} F(p)$$

Hierarchical Clustering

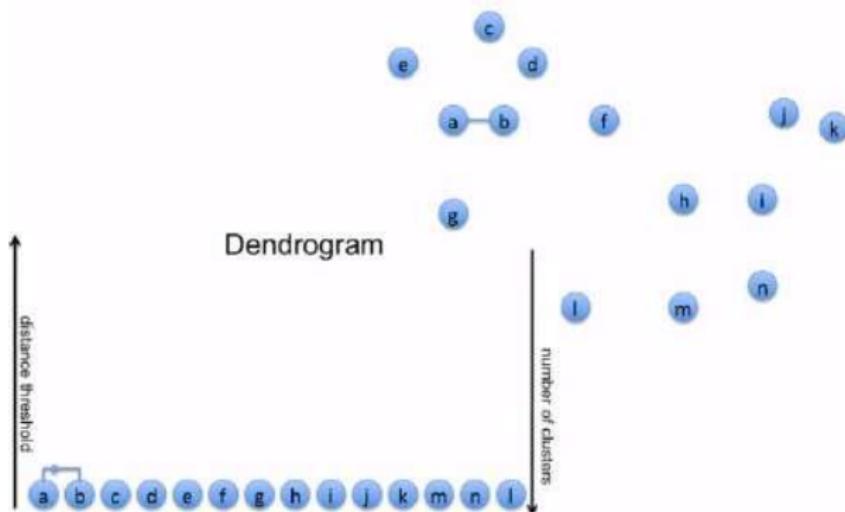
- Sometimes we might want to organise the data into a hierarchy of subsuming concepts for visualisation (abstraction) purposes
- Two methods exists
 - Conglomerative clustering
 - Start from one big cluster with all data instances and repeatedly partition it
 - Top-down approach
 - Agglomerative clustering
 - Start singletons (clusters with exactly one instance) and iteratively merge the most *similar* two clusters
 - Bottom-up approach
 - computationally more efficient ($O(\log n)$ merges required)

Hierarchical Clustering (example)



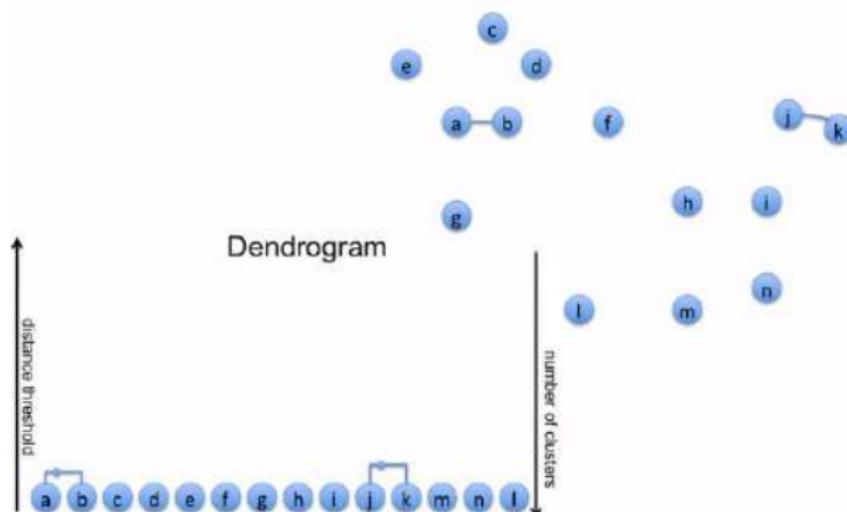
Source: Victor Lavrenko

Hierarchical Clustering (example)



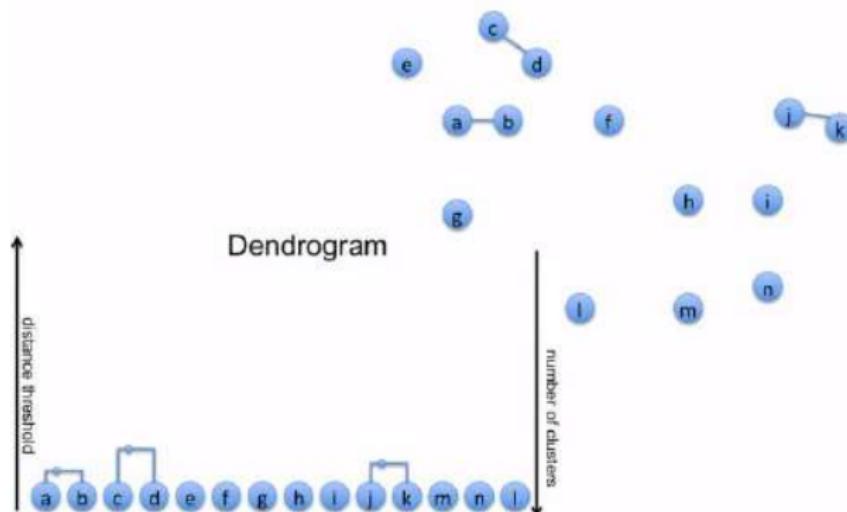
Source: Victor Lavrenko

Hierarchical Clustering (example)



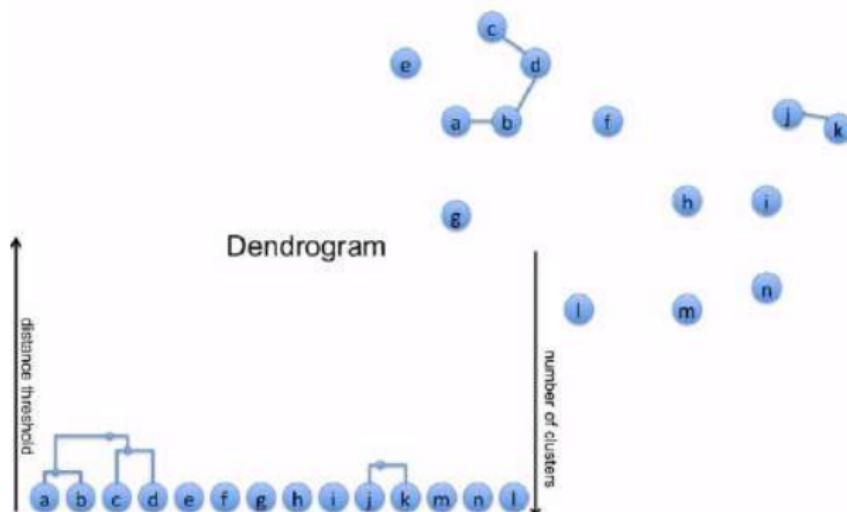
Source: Victor Lavrenko

Hierarchical Clustering (example)



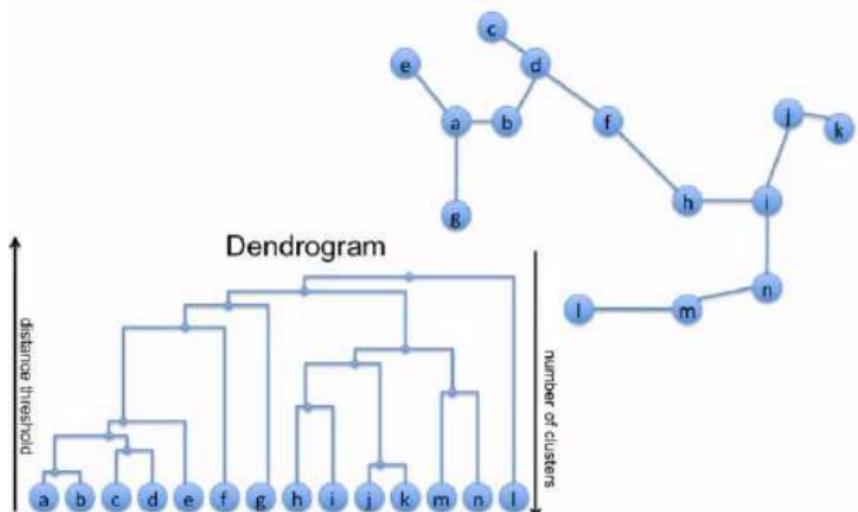
Source: Victor Lavrenko

Hierarchical Clustering (example)



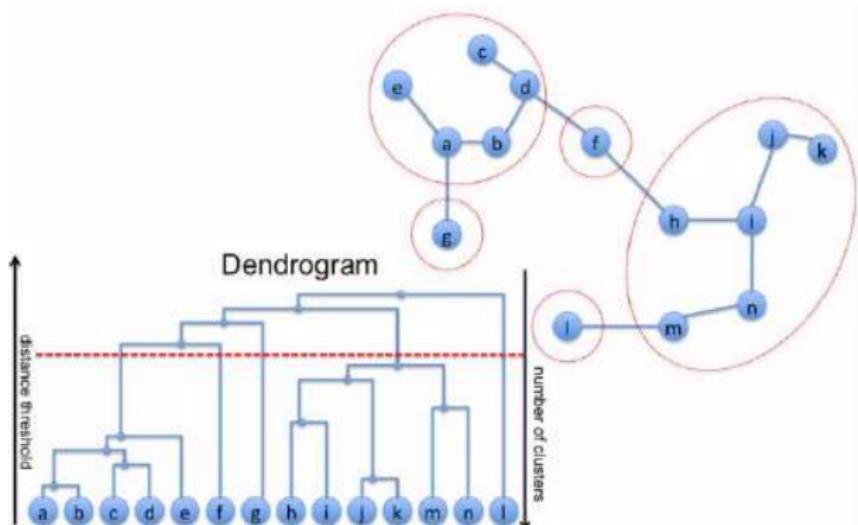
Source: Victor Lavrenko

Hierarchical Clustering (example)



Source: Victor Lavrenko

Hierarchical Clustering (example)



Source: Victor Lavrenko

Data Clustering

Danushka Bollegala



Outline

- Why cluster data?
- Clustering as unsupervised learning
- Clustering algorithms
 - k-means, k-medoids
 - agglomerative clustering
 - Brown's clustering
 - Spectral clustering
- Cluster evaluation measures
 - Purity
 - Normalised Mutual Information
 - Rand Index
 - B-CUBED
 - Precision, Recall, F-score
- Supervised clustering

We look only at topics shown in red here

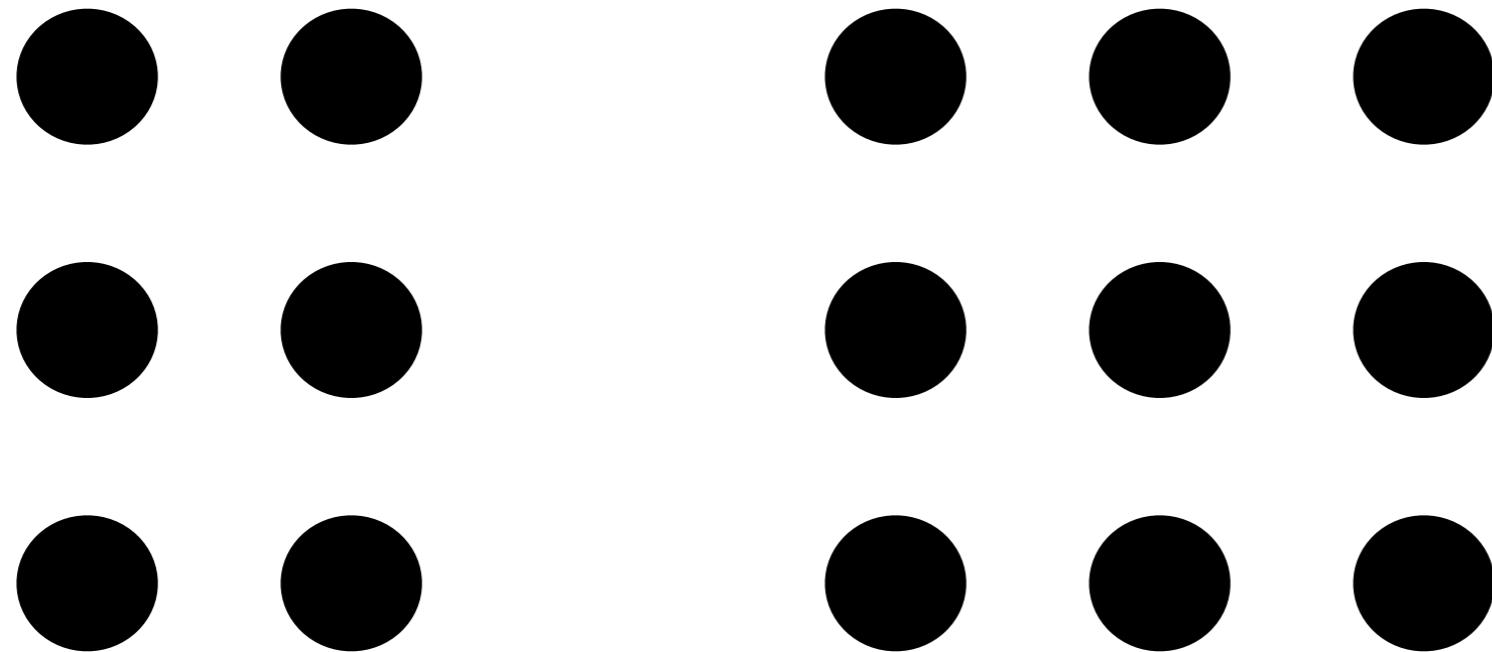
Why cluster data?

- Data Mining has two main objectives
 - Prediction: classification, regression etc.
 - Description: pattern mining, rule extraction, visualisation *clustering*
- Clustering is:
 - Unsupervised learning
 - no label data is required (consider classification algorithms we discussed so far in the lecture which are supervised algorithms)
 - Generalisation / Abstraction of concepts
 - Topic detection
 - Visualisation
 - Outlier detection

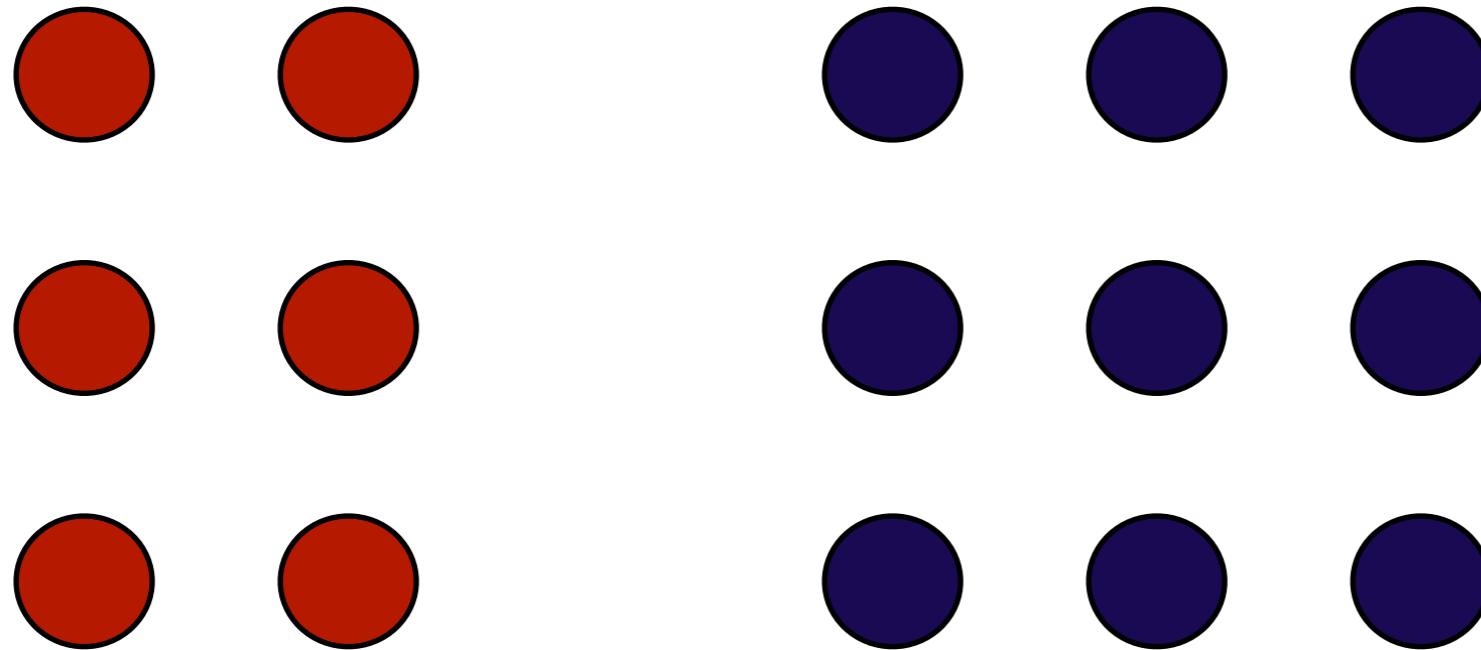
Unsupervised Learning

- Supervised learning
 - labels for training instances are provided
- Unsupervised learning
 - No labels for training instances are provided
- Semi-supervised learning
 - Both labeled and unlabeled training instances are provided
- What can we learn about training data if we do not have any labels?
 - The similarity and distribution of the features can still be learnt and this can be used to create rich feature spaces for supervised learning (if required)

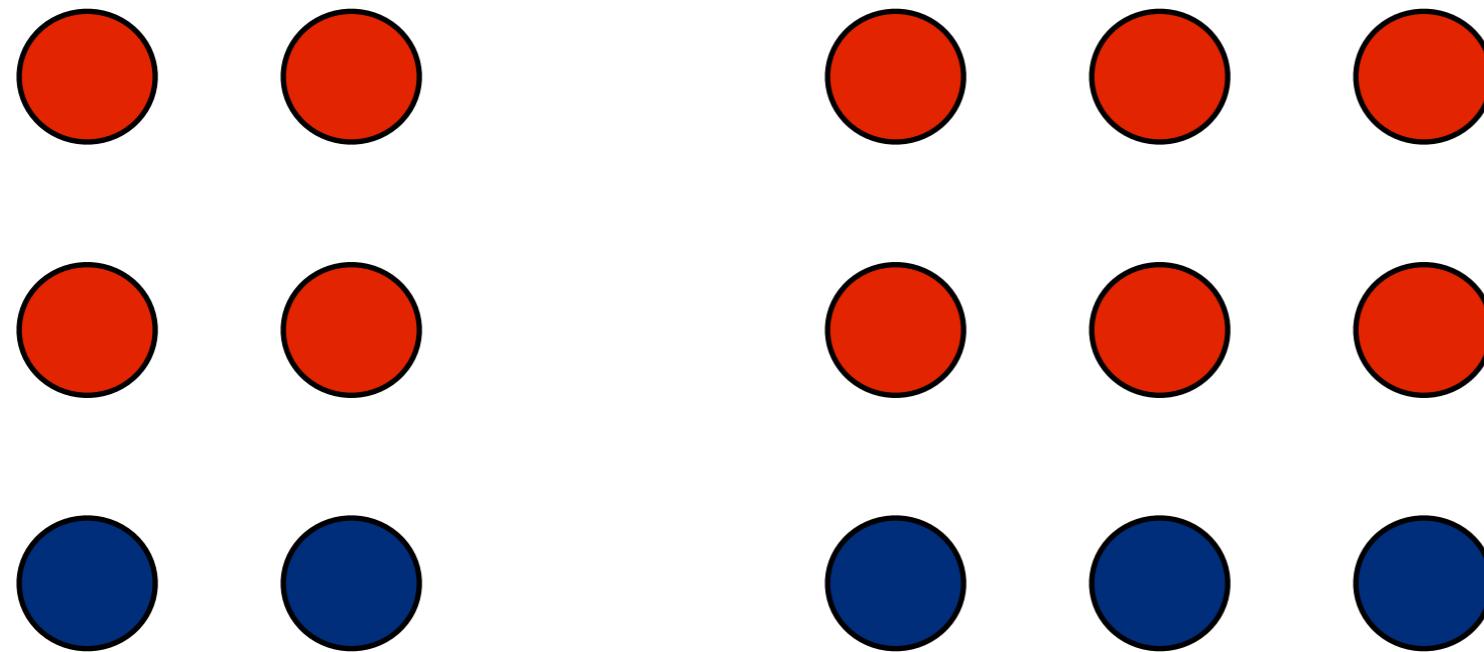
Quiz: Cluster the Following Data



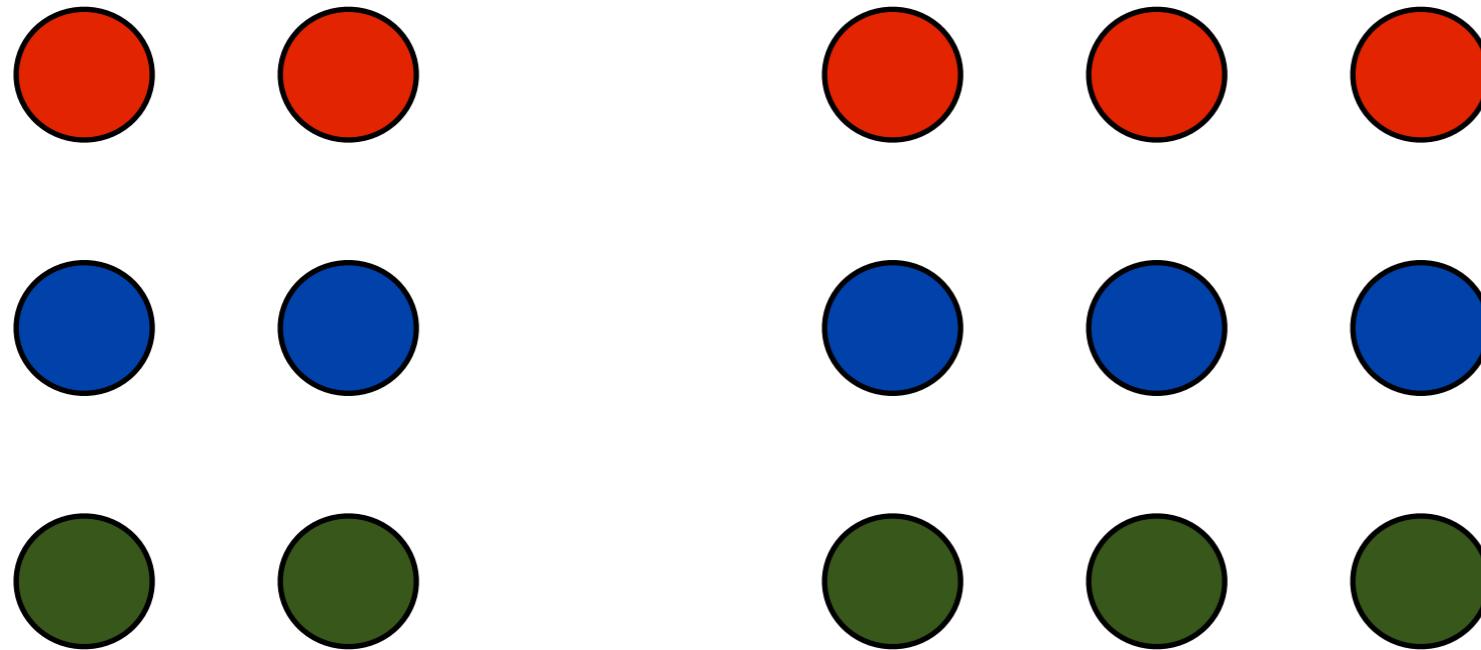
Quiz: Cluster the Following Data



Quiz: Cluster the Following Data



Quiz: Cluster the Following Data



How many clusters?

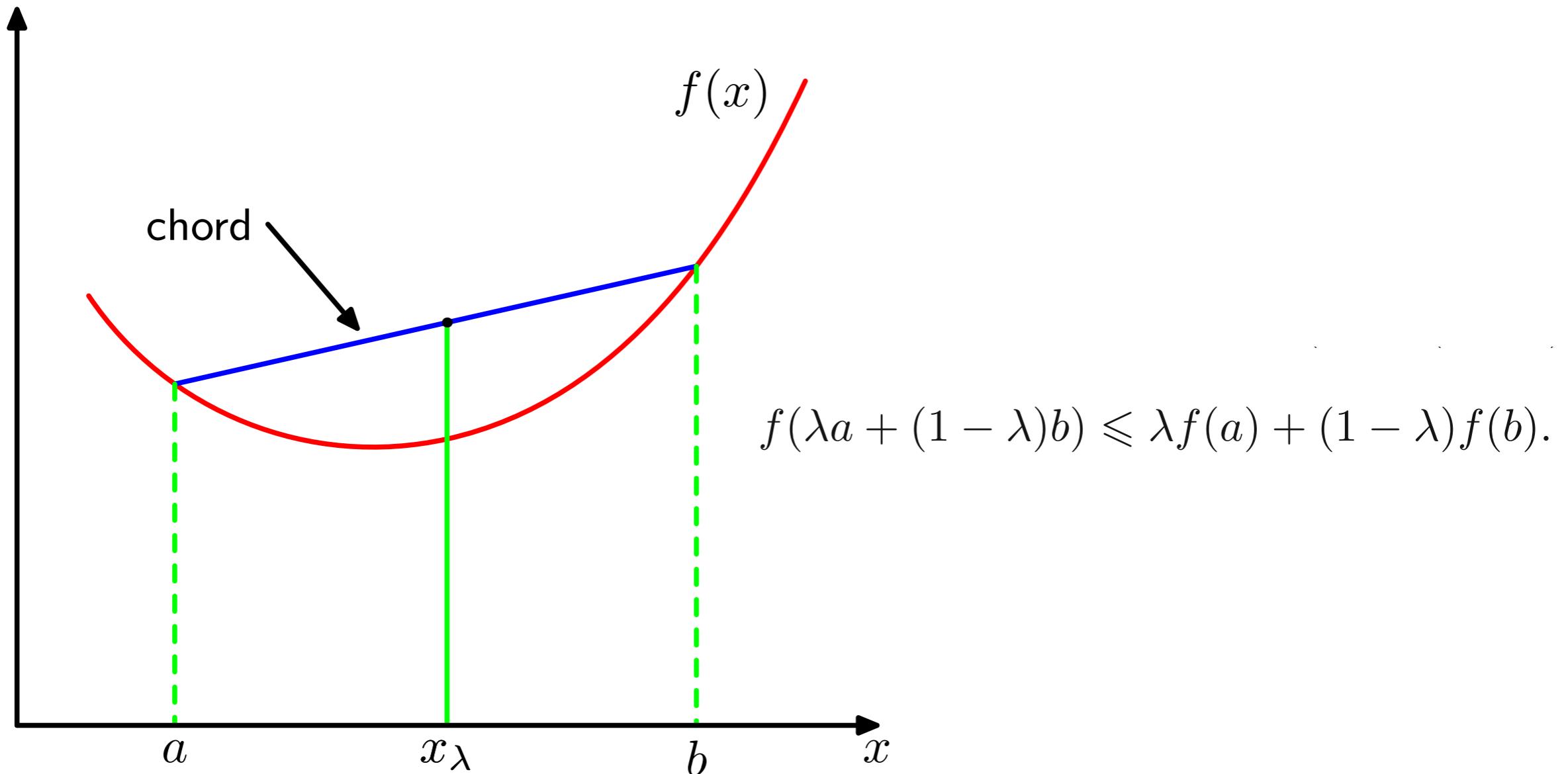
General Remarks

- A single dataset can be clustered into several ways
- There is no single right or wrong clustering
 - Simply different views on the same data
- If so how can we measure the quality of a clustering algorithm?
 - Two ways
 - Compare the clusters produced by a clustering algorithm against some reference (gold standard) set of clusters (**direct evaluation**)
 - Use the clusters as features for some other (eg. supervised learning) task and measure the difference in the performance of the second task (**indirect evaluation**)

Clustering as Optimisation

- Given a dataset $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ of N instances represented as d dimensional real vectors ($\mathbf{x}_i \in \mathbb{R}^d$), partition these N instances into k clusters S_1, \dots, S_k such that some objective function $f(S_1, \dots, S_k)$ is minimised.
- Observations
 - k and f are given
 - f can be the similarity between the clusters (good to create dissimilar clusters as much as possible), information gain, correlation and various other such *goodness* measures (heuristics)
 - Often clustering is an NP hard and a non-convex problem
 - <http://rangevoting.org/VattaniKmeansNPC.pdf>
 - approximations, relaxations are required in practice

Convex Functions



Clustering Algorithms

- Partitioning
 - Construct k partitions and iteratively update the partitions
 - k-Means, k-Medoids
- Hierarchical
 - Create a hierarchy of clusters (dendrogram)
 - Agglomerative clustering (bottom-up)
 - Conglomerative clustering (top-down)
- Graph-based clustering
 - Graph-cut algorithms (Spectral Clustering)
- Model-based clustering
 - Mixture of Gaussians
- Other types: Non-parametric Bayesian (Latent Dirichlet Allocation), Expectation Maximisation (EM) algorithm, and many more ...

k-Means Derivation

$$\arg \min_{S_1, \dots, S_k} \sum_{i=1}^k \sum_{x_j \in S_i} \|x_j - \mu_i\|^2$$

We want to minimize the distance between data instances (x_j) and some cluster centres (μ_i)

$$f(S_1, \dots, S_k) = \sum_{i=1}^k \sum_{x_j \in S_i} \|x_j - \mu_i\|^2$$

This objective function is called the *within cluster sum of squares* (WCSS) objective

$$\frac{\partial f(S_1, \dots, S_k)}{\partial \mu_i} = 0$$

$$\frac{\partial f(S_1, \dots, S_k)}{\partial \mu_i} = \sum_{\mathbf{x}_j \in S_i} 2(\mathbf{x}_j - \boldsymbol{\mu}_i)$$

$$\boldsymbol{\mu}_i = \frac{1}{|S_i|} \sum_{\mathbf{x}_j \in S_i} \mathbf{x}_j$$

Just compute the centroid (mean) of each cluster
and that will give you the cluster centers

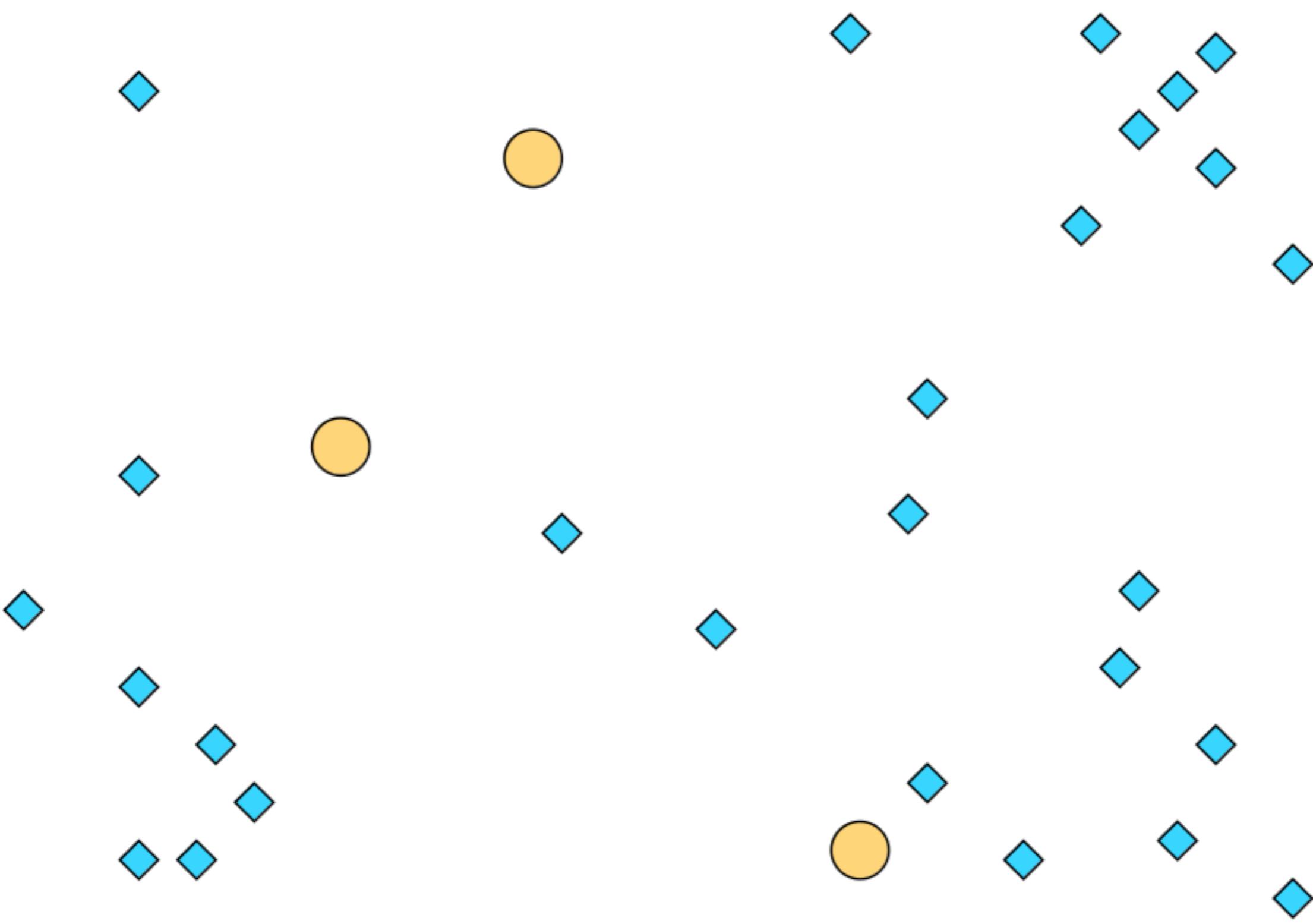
k-Means Clustering

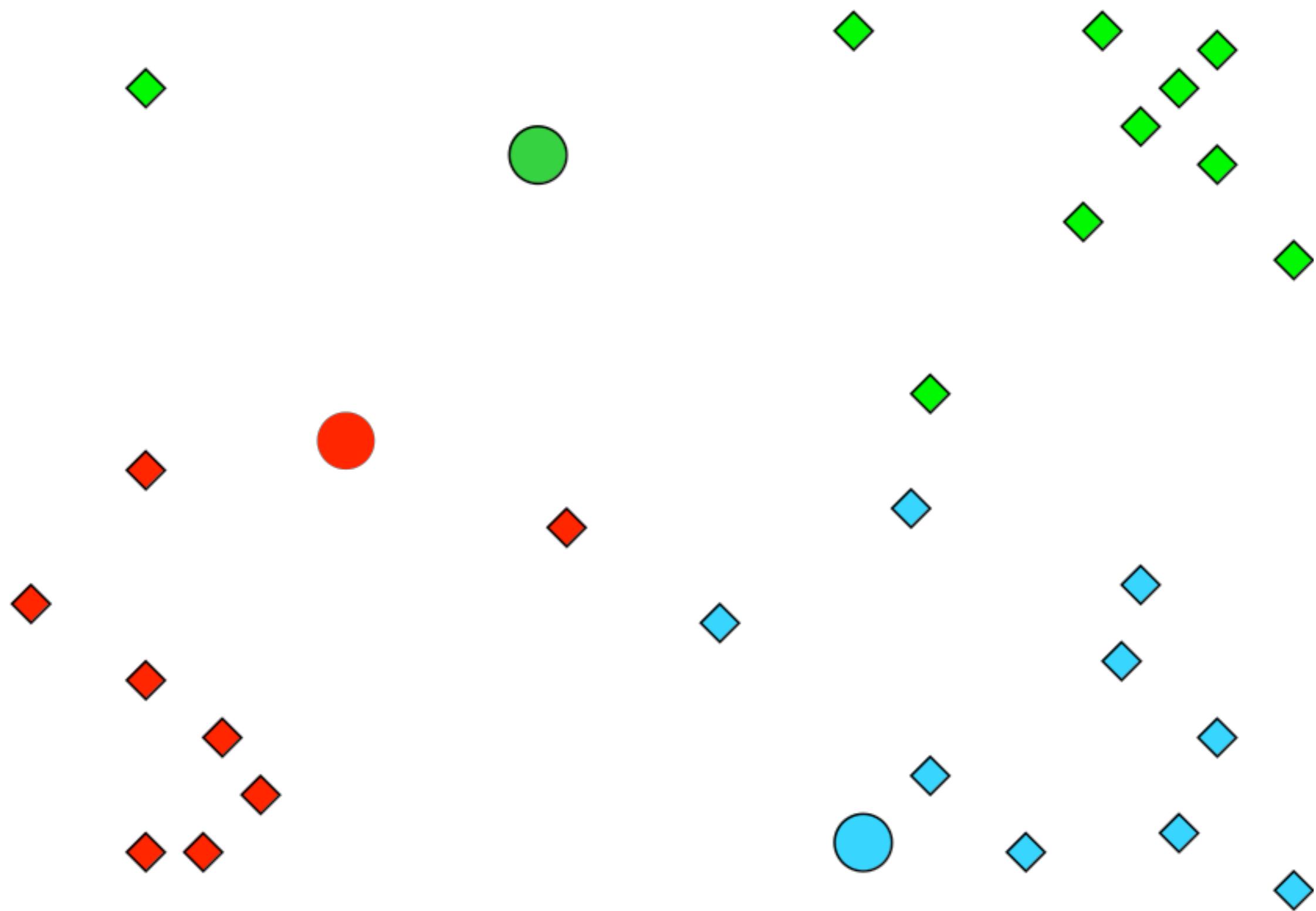
- INPUT
 - The number of clusters k
 - Dataset $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ of N instances represented as d dimensional real vectors ($\mathbf{x}_i \in \mathbb{R}^d$)

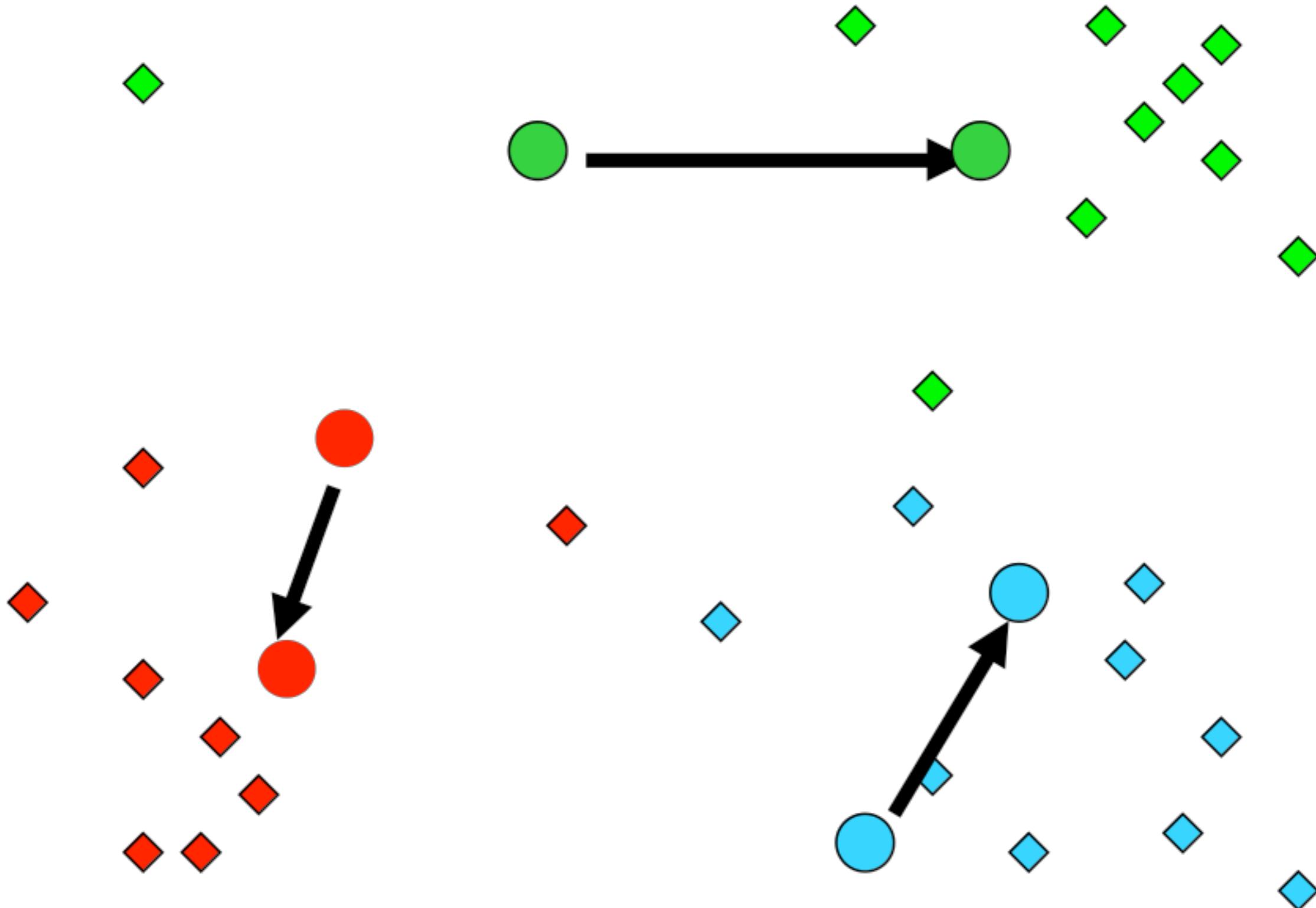
1. Set k instances from the dataset randomly. (initial cluster means/ centers)
2. Assign all other instances to the closest cluster centre.
3. Compute the mean of each cluster
4. until **convergence** repeat between steps 2 and 3

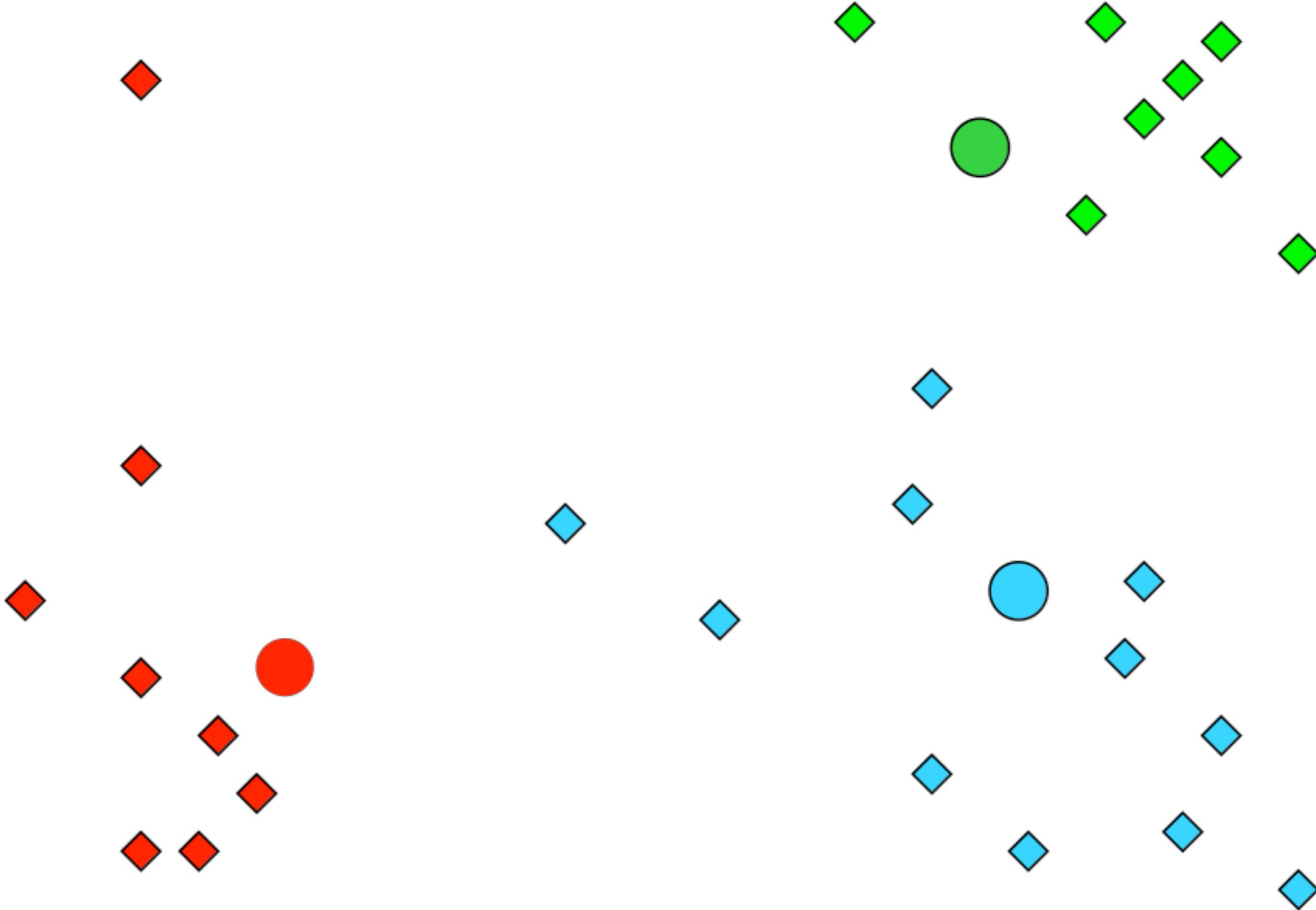
convergence = no instances have moved among clusters

(often after a fixed number of iterations specified by the user)









Issues with k-Means

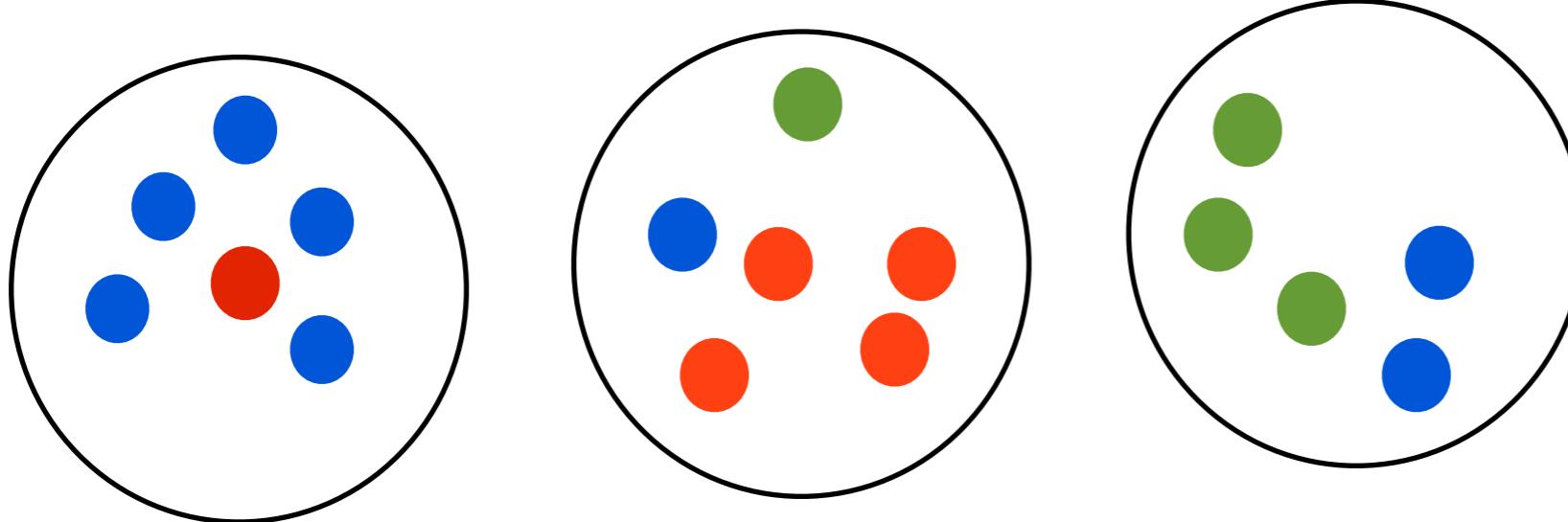
- Results can vary depending on the initial random choices
- Can get trapped in a local minimum that isn't the global optimal solution
 - Repeat the clustering procedure multiple times with different initialisations and select the *best* final clustering
 - *best?* according to what? many heuristics exist.
 - smallest number of iterations before convergence
 - largest total distance between the final cluster means
 - Outliers have a larger effect on the mean value, hence cluster centre and the cluster
 - cluster centres (means) are not actual instances in the cluster
 - We could pick actual instances as initial cluster centroids.

Evaluating Clustering — Purity

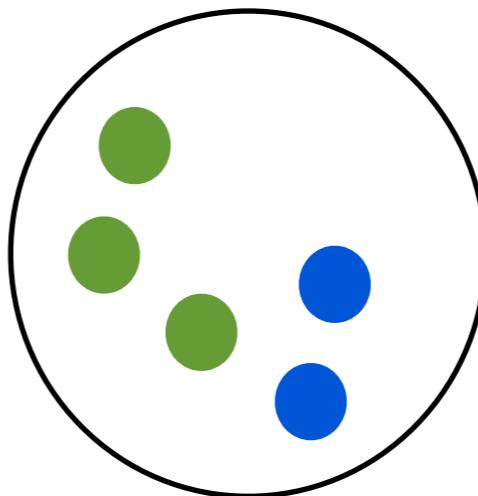
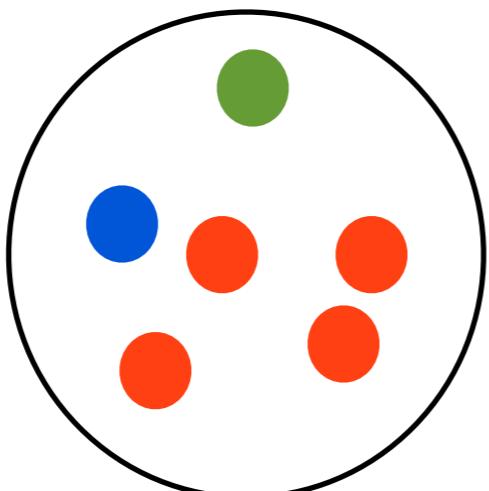
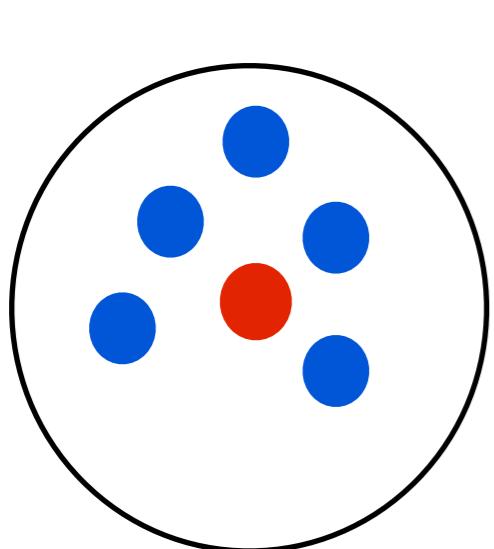
- Let us assume that we have a set $\Omega = \{\omega_1, \dots, \omega_K\}$ clusters for a set of classes $C = \{c_1, \dots, c_J\}$
- Purity measures the ratio of the items that are in the cluster with the same class as its own.

$$\text{purity}(\Omega, C) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j|$$

- Here, N is the total number of items.



Quiz: Compute purity for this clustering.



Labels



$$\text{purity} = (5 + 4 + 3) / 17 = 12/17 = 0.71$$

Purity achieves its maximum value of 1 for singletons (each item is in a cluster containing only that single item)! Obviously this is not good “clustering” and purity does not recognise this.

Evaluating Clustering — NMI

- Let us assume that we have a set $\Omega = \{\omega_1, \dots, \omega_K\}$ clusters for a set of classes $C = \{c_1, \dots, c_J\}$
- Normalised Mutual Information (NMI) computes the ratio of information that we can know about the classes C given the clusters Ω to the averaged information that is contained in C and Ω .

$$\text{NMI}(\Omega, C) = \frac{I(\Omega, C)}{[H(\Omega) + H(C)]/2}$$

$$\begin{aligned} I(\Omega, C) &= \sum_k \sum_j p(\omega_k \cap c_j) \log \left(\frac{p(\omega_k \cap c_j)}{p(\omega_k)p(c_j)} \right) \\ &= \sum_k \sum_j \frac{|\omega_k \cap c_j|}{N} \log \left(\frac{N|\omega_k \cap c_j|}{|\omega_k||c_j|} \right) \end{aligned}$$

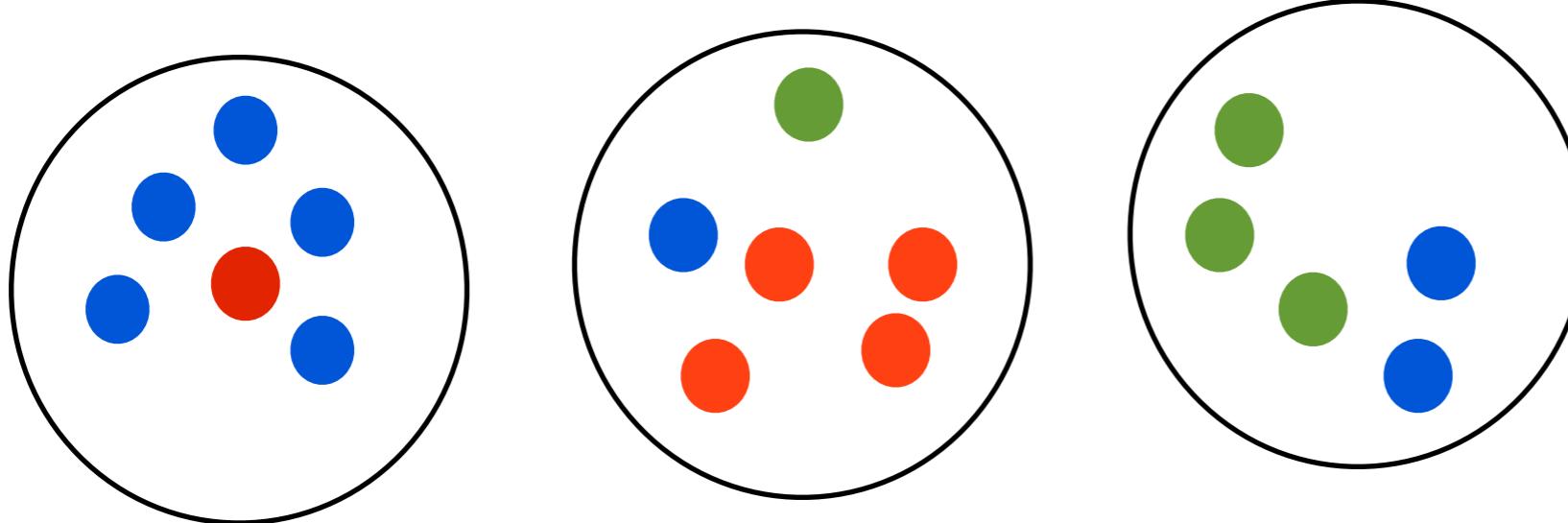
Mutual Information (MI)

$$\begin{aligned} H(\Omega) &= - \sum_k p(\omega_k) \log p(\omega_k) \\ &= - \sum_k \frac{|\omega_k|}{N} \log \frac{|\omega_k|}{N} \end{aligned}$$

Entropy

Why we do we normalise by the average?

- $I(X,Y) \leq [H(X) + H(Y)]/2$
- Proof (sketch):
 - $I(X,Y) = H[X] - H[X|Y] = H[Y] - H[Y|X]$
 - Add those two and use the fact that (conditional) entropy is nonnegative
 - $H[X|Y] + H[Y|X] \geq 0$



Quiz: Compute NMI for this clustering.

Let $C_1 = \text{Blue}$, $C_2 = \text{Red}$ and $C_3 = \text{Green}$.

$$P(C_1) = \frac{8}{17}, \quad P(C_2) = \frac{5}{17}, \quad P(C_3) = \frac{4}{17}.$$

$$\begin{aligned} H(C) &= - \sum_{j=1}^3 P(C_j) \log P(C_j) \\ &= - \left[\frac{8}{17} \log \frac{8}{17} + \frac{5}{17} \log \frac{5}{17} + \frac{4}{17} \log \frac{4}{17} \right] = 1.055 \end{aligned}$$

Likewise,

$$P(\omega_1) = \frac{6}{17}, \quad P(\omega_2) = \frac{6}{17}, \quad P(\omega_3) = \frac{5}{17}$$

$$H(\omega) = - \left[\frac{6}{17} \log \frac{6}{17} + \frac{6}{17} \log \frac{6}{17} + \frac{5}{17} \log \frac{5}{17} \right] = 1.095$$

$$P(\omega_1 \cap C_1) = \frac{5}{17} \quad P(\omega_1 \cap C_2) = \frac{1}{17} \quad P(\omega_1 \cap C_3) = \frac{0}{17}$$

$$P(\omega_2 \cap C_1) = \frac{1}{17} \quad P(\omega_2 \cap C_2) = \frac{4}{17} \quad P(\omega_2 \cap C_3) = \frac{1}{17}$$

$$P(\omega_3 \cap C_1) = \frac{2}{17} \quad P(\omega_3 \cap C_2) = \frac{0}{17} \quad P(\omega_3 \cap C_3) = \frac{3}{17}$$

$$I(\omega, C) = \sum_{k=1}^2 \sum_{j=1}^3 P(\omega_k \cap C_j) \log \frac{P(\omega_k \cap C_j)}{P(\omega_k) P(C_j)} = 0.4496$$

$$\therefore NMI(\omega, C) = \frac{I(\omega, C)}{(H(\omega) + H(C))/2} = \frac{0.4496}{(1.055 + 1.095)/2} = 0.4182$$

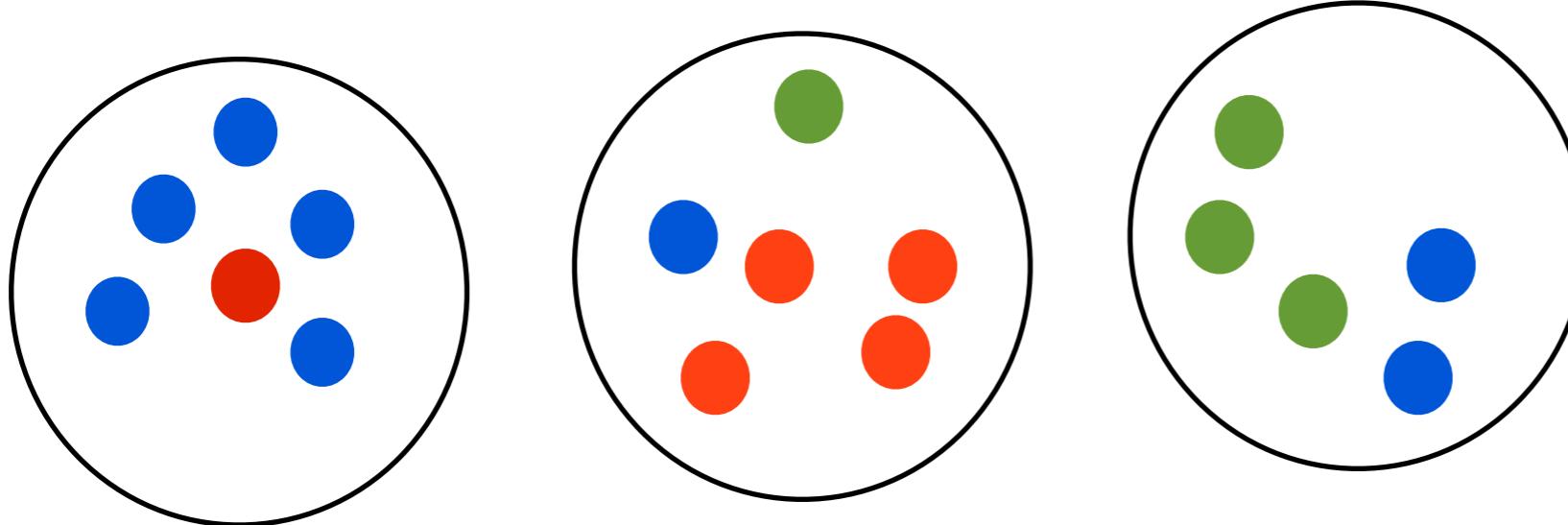
Evaluating Clustering — Rand Index (RI)

- Build a contingency table considering pairs of items in each cluster
 - Positive = same cluster
 - Negative = different clusters
 - True = same class
 - False = different classes
- TP = No. of item pairs that are in the same cluster and belong to the same class
- FP = No. of item pairs that are in the same cluster but belong to different classes
- TN = No. of item pairs that are in different clusters and belong to different classes
- FN = No. of item pairs that are in different clusters but belong to the same class
-

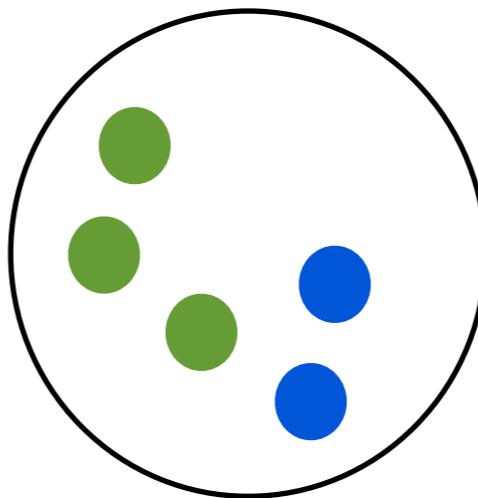
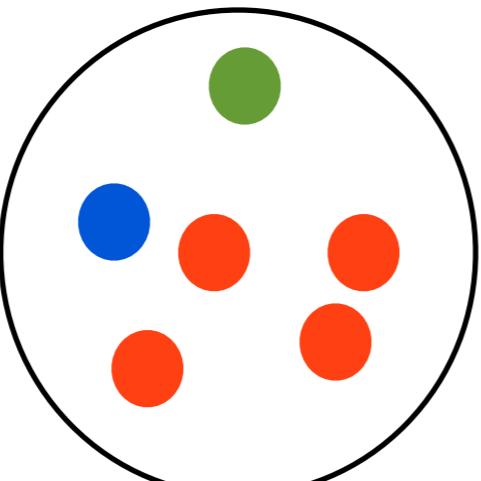
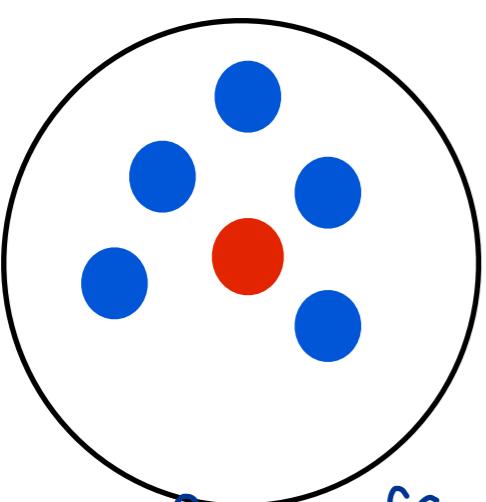
contingency table		same cluster	different clusters
same class	TP	FN	
different classes	FP	TN	

$$RI = \frac{TP + TN}{TP + FP + TN + FN}$$

(accuracy of
the clustering)



Quiz: Compute RI for this clustering.



$$TP + FP = {}^6C_2 + {}^6C_2 + {}^5C_2 = 15 + 15 + 10 = 40.$$

$$\left[{}^nC_r = \frac{n!}{r!(n-r)!} \right]$$

$${}^6C_2 = \frac{6!}{2!4!} = \frac{6 \times 5}{2} = 15 \quad {}^5C_2 = \frac{5!}{2!3!} = \frac{5 \times 4}{2} = 10.$$

$$TP = {}^5C_2 + {}^4C_2 + {}^3C_2 + {}^2C_2 = 10 + 6 + 3 + 1 = 20$$

$${}^4C_2 = \frac{4!}{2!2!} = \frac{4 \times 3}{2} = 6$$

$$\therefore FP = 40 - 20 = 20.$$

$$\begin{aligned} FN &= (5 \times 1) + (1 \times 4) + (5 \times 2) + (1 \times 2) + (1 \times 3) \\ &= 5 + 4 + 10 + 2 + 3 = 24 \end{aligned}$$

$$TN + FN = (5+1)(1+1+4) + (5+1)(3+2) + (1+1+4)(3+2)$$

$$= 6 \times 6 + 6 \times 5 + 6 \times 5 = 36 + 30 + 30 = 96.$$

$$\therefore TN = 96 - 24 = 72$$

	same cluster	different clusters
same class	20	24
different classes	20	72

$$\begin{aligned} RI &= (20+72) / (20+24+20+72) \\ &= 0.676 \end{aligned}$$

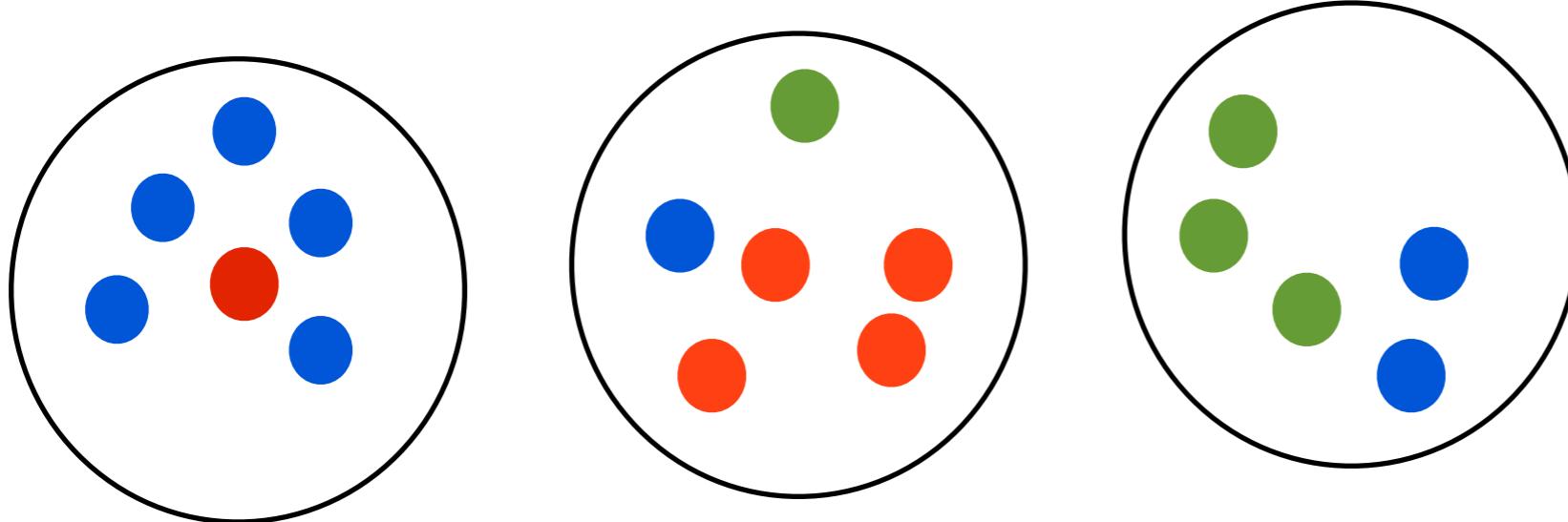
Evaluating Clustering — P/R/F

- We can use Precision (P), Recall (R), and F-measure (F) at to evaluate the accuracy of a clustering.
- For this purpose we must first create the contingency table as we did for RI and then compute P, R, F as follows

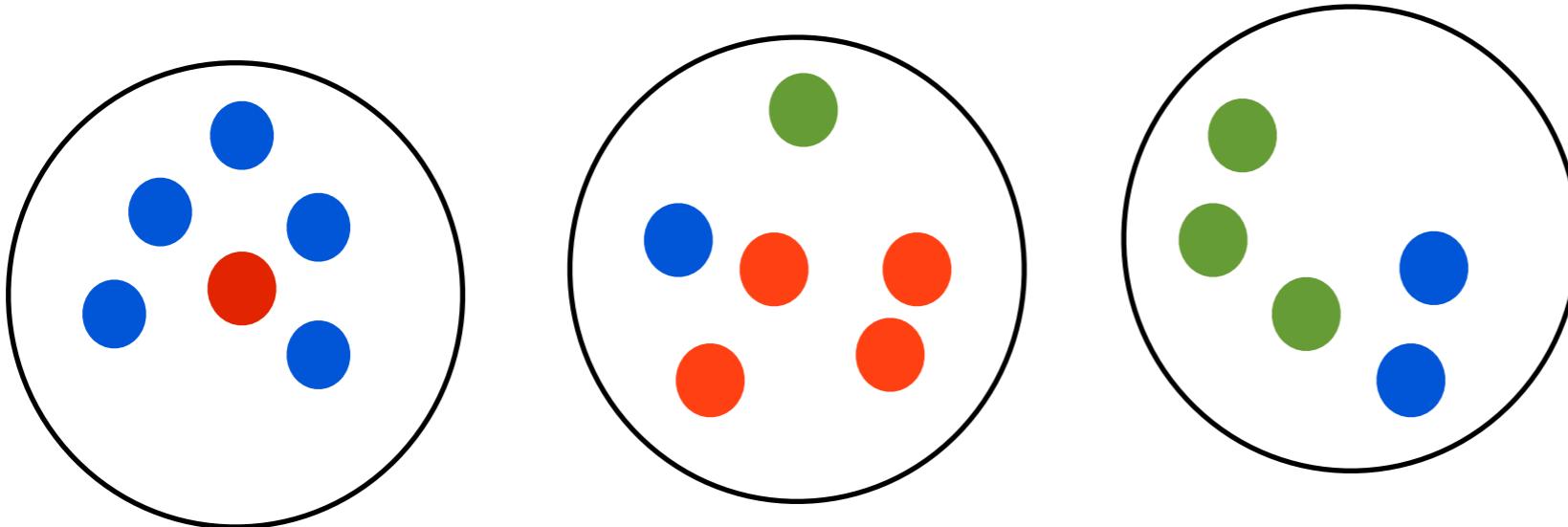
$$P = TP / (TP + FP)$$

$$R = TP / (TP + FN)$$

$$F = 2PR / (P + R)$$



Quiz: Compute P/R/F for this clustering.



	same cluster	different clusters
same class	TP=20	FN=24
different classes	FP=20	TN=72

$$P = TP / (TP + FP) = 20 / (20+20) = 0.5$$

$$R = TP / (TP + FN) = 20 / (20 + 24) = 0.45$$

$$F = 2PR / (P + R) = 0.47$$

B-CUBED Measure

- Proposed in (Bagga B. Baldwin = B³)
 - A. Bagga and B. Baldwin. Entity-based cross document coreference resolution using the vector space model, In Proc. of 36th COLING-ACL, pages 79–85, 1998.
- We would like to evaluate clustering without labelling any clusters.

$$\text{precision}(x) = \frac{\text{No. of items in } C(x) \text{ with } A(x)}{\text{No. of items in } C(x)}$$

$$\text{recall}(x) = \frac{\text{No. of items in } C(x) \text{ with } A(x)}{\text{Total no. of items with } A(x)}$$

$C(x)$: The ID of the cluster that x belongs to

$A(x)$: label of x

B-CUBED Measure

- Compute the average over all the items (instances) that appear in all clusters (N)

$$\text{Precision} = \frac{1}{N} \sum_{p \in \text{DataSet}} \text{Precision}(p)$$

$$\text{Recall} = \frac{1}{N} \sum_{p \in \text{DataSet}} \text{Recall}(p)$$

$$F\text{-Score} = \frac{1}{N} \sum_{p \in \text{DataSet}} F(p)$$

Hierarchical Clustering

- Sometimes we might want to organise the data into a hierarchy of subsuming concepts for visualisation (abstraction) purposes
- Two methods exists
 - Conglomerative clustering
 - Start from one big cluster with all data instances and repeatedly partition it
 - Top-down approach
 - Agglomerative clustering
 - Start singletons (clusters with exactly one instance) and iteratively merge the most *similar* two clusters
 - Bottom-up approach
 - computationally more efficient ($O(\log n)$ merges required)

Merging two clusters

- Single linkage
 - Distance between two clusters A and B is the smallest distance between any instance $a \in A$ and $b \in B$

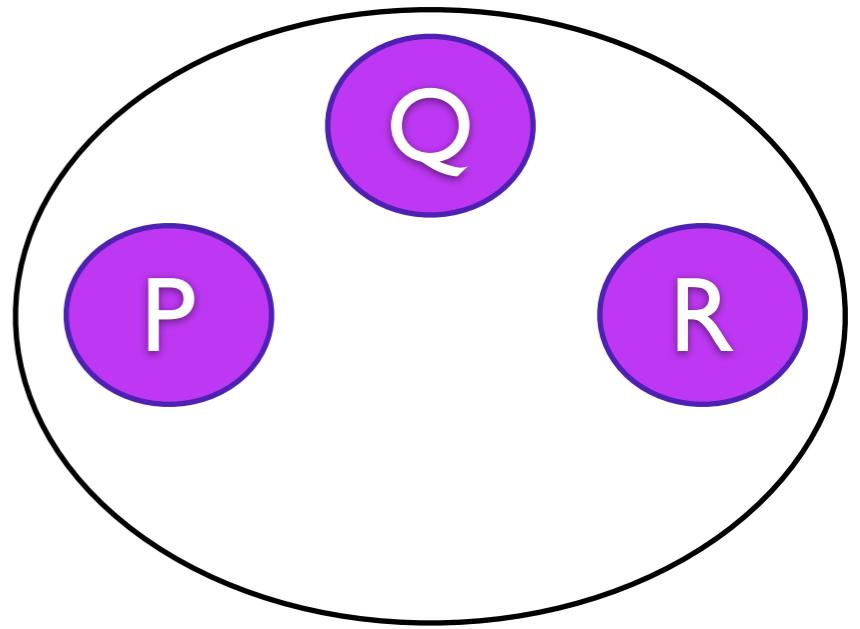
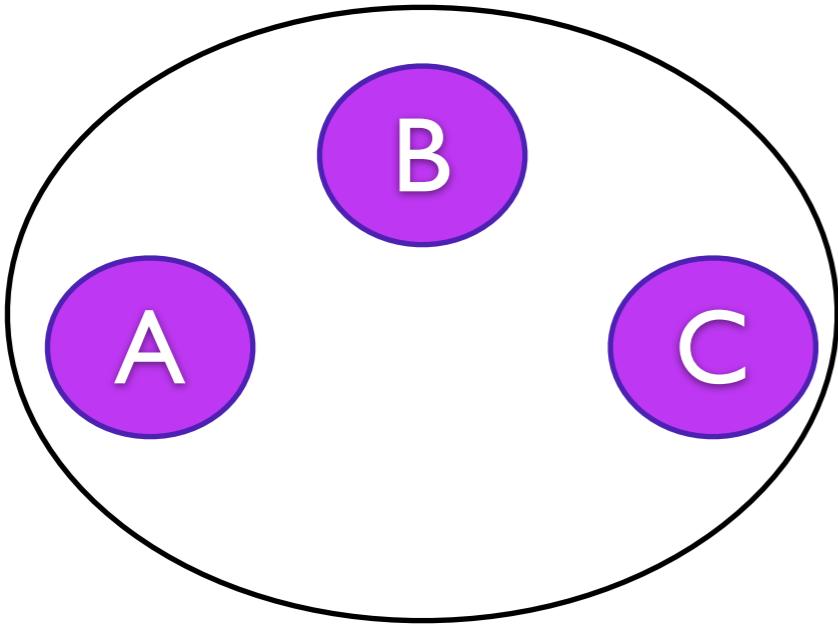
$$D(\mathcal{A}, \mathcal{B}) = \min_{a \in \mathcal{A}, b \in \mathcal{B}} dist(a, b)$$

- Complete linkage
 - Distance between two clusters A and B is the largest distance between any instance $a \in A$ and $b \in B$

$$D(\mathcal{A}, \mathcal{B}) = \max_{a \in \mathcal{A}, b \in \mathcal{B}} dist(a, b)$$

- Average linkage (Group-Average)
 - Average of all the pairs selected from each cluster

$$D(\mathcal{A}, \mathcal{B}) = \frac{1}{|\mathcal{A}||\mathcal{B}|} \sum_{a \in \mathcal{A}, b \in \mathcal{B}} dist(a, b)$$

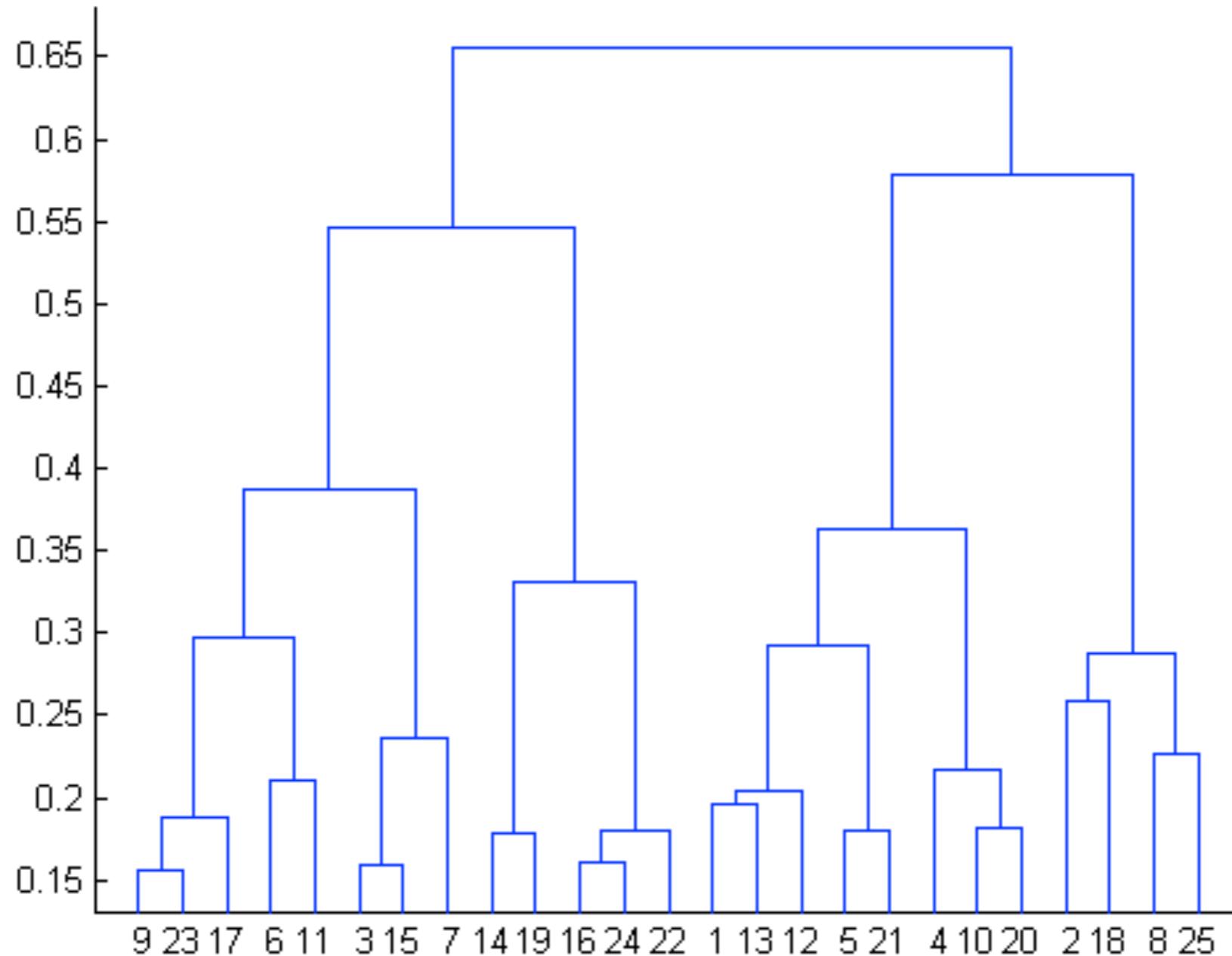


Quiz: Let us assume that in the 2D space there are two clusters $\{A, B, C\}$ and $\{P, Q, R\}$. Which of the distances correspond to the single link and complete link distances between the shown clusters?

Group-Average Agglomerative Clustering

- INPUT:
 - A set of N data instances $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, Number of clusters k
 - Initialise
 - Create singletons $S_i = \{\mathbf{x}_i\}$ for $i = 1, \dots, N$
 - Repeat until only we are left with one cluster
 - Merge the two clusters S_i and S_j with the minimum distance (cf. maximum similarity)
 - $$D(S_i, S_j) = \frac{1}{|S_i||S_j|} \sum_{a \in S_i, b \in S_j} dist(a, b)$$

Dendrogram



Clustering



Outline

- Why cluster data?
- Clustering as unsupervised learning
- Clustering algorithms
 - k-means, k-medoids
 - agglomerative clustering
 - Brown's clustering
 - Spectral clustering
- Cluster evaluation measures
 - Purity
 - Normalised Mutual Information
 - Rand Index
 - B-CUBED
 - Precision, Recall, F-score

Why cluster data?

- Data mining has two main objectives:
 - Prediction: classification, regression etc.
 - Description: pattern mining, rule extraction, visualisation, *clustering*
- Clustering is:
 - Unsupervised learning
 - no label data is required (consider classification algorithms we discussed so far in the lectures which are supervised algorithms)

Unsupervised Learning

- Supervised learning
 - labels for training instances are provided
- Unsupervised learning
 - no labels for training instances are provided
- Semi-Supervised learning
 - Both labeled and unlabeled training instances are provided
- What can we learn about training data if we do not have any labels?
 - The similarity and distribution of the features can still be learnt and this can be used to create rich feature spaces for supervised learning (if required)

Clustering: Example

Headlines

[More Headlines](#)

Coronavirus: Boris Johnson announces plan for 'delay' phase

Daily Mail · 1 hour ago

- [Coronavirus: Boris Johnson to hold emergency Cobra meeting](#)

BBC South East Wales · 7 hours ago

- [BREAKING: UK cases of coronavirus rise to 319](#)

Sky News · 6 hours ago

- [Coronavirus brings a reminder of the iron law of politics](#)

Financial Times · 4 hours ago · Opinion

- [Nigel Farage: Yes, Protecting Us All from an Epidemic Should be Prioritized Over the Economy | Opinion](#)

Newsweek · 2 hours ago · Opinion

 [View Full coverage](#)



General Remarks

- A single dataset can be clustered into several ways
- There is no single right or wrong clustering
 - Simply different views of the same data
- how to measure the quality of clustering algorithm?
 - Two ways
 - Compare clusters produced by clustering algorithm against some reference (gold standard) set of clusters (**direct evaluation**)
 - Use the clusters for some other (eg. supervised learning) task and measure the difference in performance of the second task (**indirect evaluation**)

Clustering as Optimisation

- Given a dataset $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ of N instances represented as d dimensional real vectors ($\mathbf{x}_i \in \mathbb{R}^d$), partition these N instances into k clusters S_1, \dots, S_k such that some objective function $f(S_1, \dots, S_k)$ is minimised.
- Observations
 - k and f are given
 - f can be similarity between the clusters (good to create dissimilar clusters as much as possible), information gain, correlation and various other such goodness measures (heuristics)

Partitioning - k-means algorithm

$$\arg \min_{S_1, \dots, S_k} \sum_{i=1}^k \sum_{x_j \in S_i} \|x_j - \mu_i\|^2$$

We want to minimize the distance between data instances (x_j) and some cluster centres (μ_i)

$$f(S_1, \dots, S_k) = \sum_{i=1}^k \sum_{x_j \in S_i} \|x_j - \mu_i\|^2$$

This objective function is called the *within cluster sum of squares* (WCSS) objective

Partitioning - cluster centroids

$$\mu_i = \frac{1}{|S_i|} \sum_{x_j \in S_i} x_j$$

Just compute the centroid (mean) of each cluster
and that will give you the cluster centers

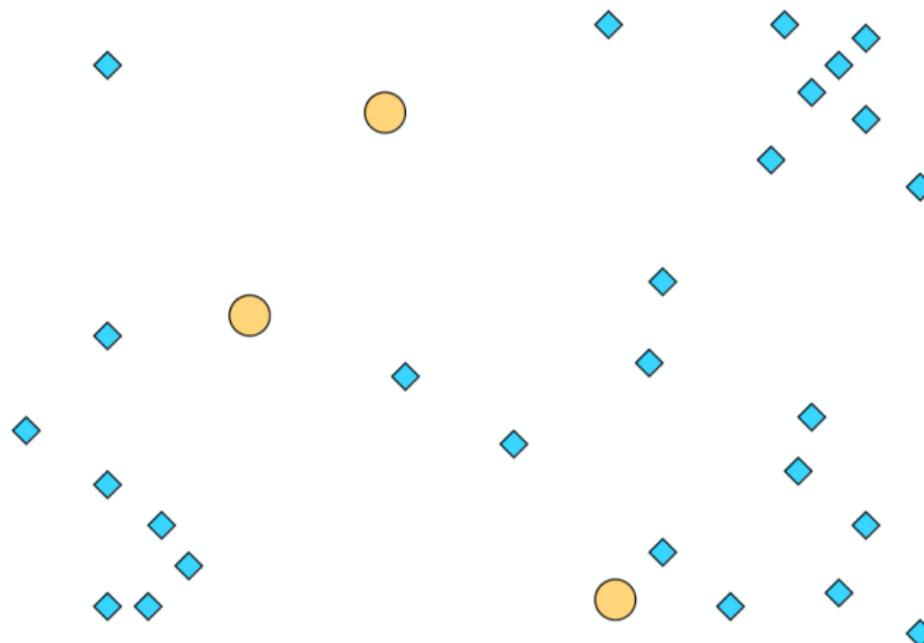
k-Means clustering

- Input

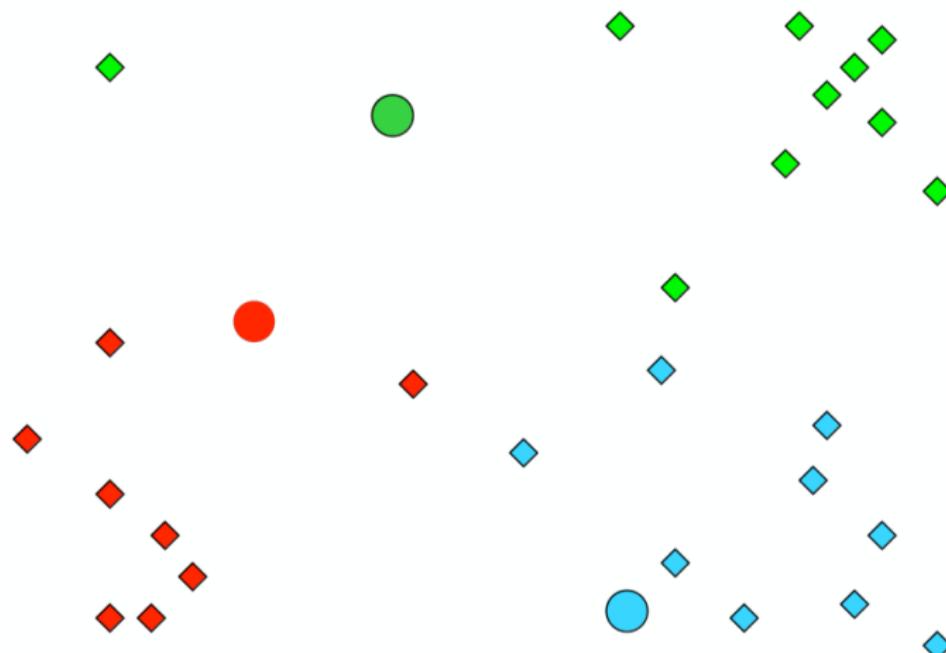
- The number of clusters k
- Dataset $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ of N instances represented as d dimensional real vectors ($\mathbf{x}_i \in \mathbb{R}^d$)
- 1 Set k instances from the dataset randomly (initial cluster means / centers)
- 2 Assign all other instances to the closest cluster centre
- 3 Compute the mean of each cluster
- 4 Until **convergence** repeat between steps 2 and 3

convergence = no instances have moved among clusters
(often after a fixed number of iterations specified by the user)

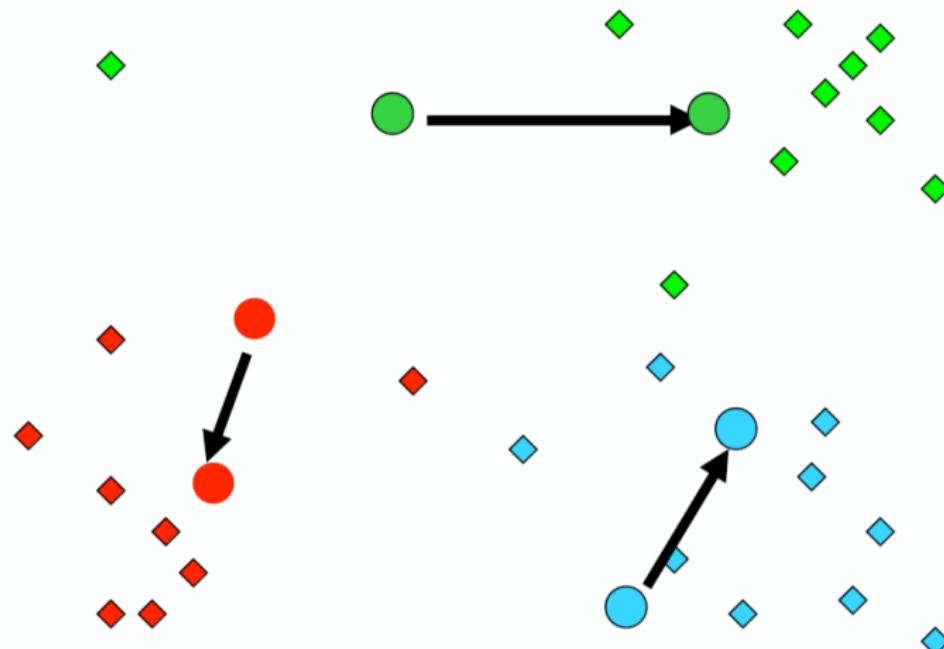
k-means clustering



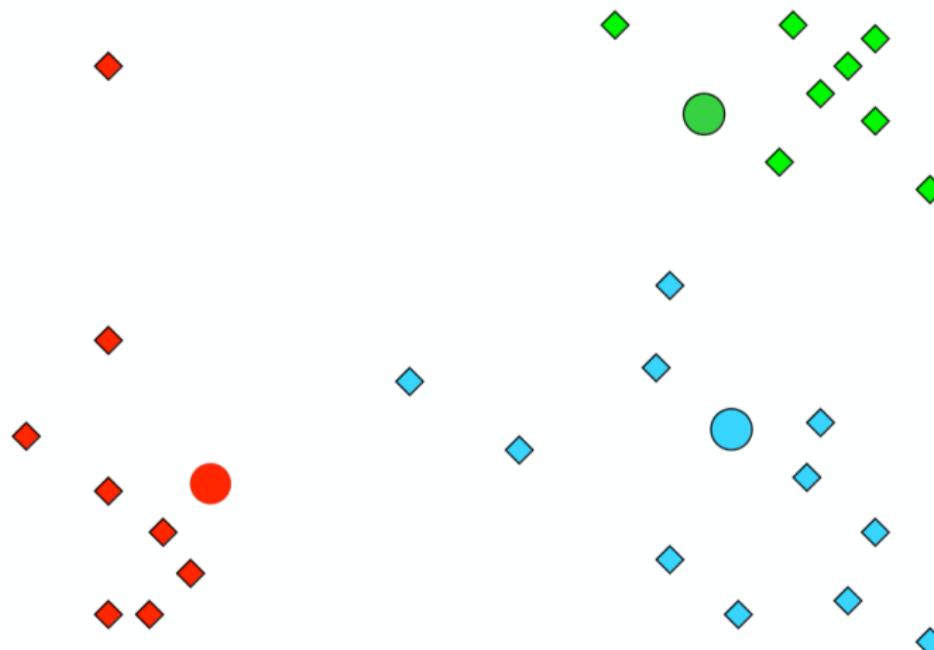
k-means clustering



k-means clustering



k-means clustering



k-means clustering (example)

		$c_1 (1,0)$	$c_2 (1,1)$	Assignment
x_1	0,0	$\sqrt{(0-1)^2 + (0-0)^2} = \sqrt{1}$	$\sqrt{(0-1)^2 + (0-1)^2} = \sqrt{2}$	c_1
x_2	1,0	$\sqrt{(1-1)^2 + (0-0)^2} = \sqrt{0}$	$\sqrt{(1-1)^2 + (0-1)^2} = \sqrt{1}$	c_1
x_3	1,1	$\sqrt{(1-1)^2 + (1-0)^2} = \sqrt{1}$	$\sqrt{(1-1)^2 + (1-1)^2} = \sqrt{0}$	c_2
x_4	0,1	$\sqrt{(0-1)^2 + (1-0)^2} = \sqrt{2}$	$\sqrt{(0-1)^2 + (1-1)^2} = \sqrt{1}$	c_2
x_5	-1,0	$\sqrt{(-1-1)^2 + (0-0)^2} = \sqrt{4}$	$\sqrt{(-1-1)^2 + (0-1)^2} = \sqrt{5}$	c_1

- $c_1 = \{x_1, x_2, x_5\}; c_2 = \{x_3, x_4\}$
- $c_1 = \{(0, 0), (1, 0), (-1, 0)\}; c_2 = \{(1, 1), (0, 1)\}$
- $\mu_{c_1} = (0, 0); \mu_{c_2} = (0.5, 1)$
- computing clusters using new μ gives the same clusters

Decision Trees

Danushka Bollegala



Rule-based Classifiers

- In rule-based learning, the idea is to *learn* a rule from train data in the form IF X THEN Y (or a combination of nested conditions) that explains when Y would be TRUE
- Example
 - IF forecast=rainy THEN play=NO
 - If the weather forecast says it is going to rain, then we will not have a match tomorrow.

Pros/Cons

- Advantages of Rule-based Learners
 - Rules are often easier to understand
 - Compare this with the perceptron's weight vector we studied last week
 - We can handle features (aka attributes) that have categorical values (e.g. forecast=sunny/rainy) without requiring those to be mapped to numerical values
- Disadvantages of Rule-based Learners
 - Tend to overfit to the train data more easily than some of the other classification algorithms (k-NN, SVMs)
 - Rules can get very complicated when we have millions of features
 - Consider learning a rule for sentiment classification from texts
 - Old-school machine learning :-) but there are some modern versions with good performance such as Random Forest Classifiers.

Dataset

outlook	temperature	humidity	windy	play?
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Task

- We would like to learn a rule that predicts when we would play
 - IF THEN play=yes
- Requirements
 - The rule must be accurate (it should correctly predict the training instances as much as possible)
 - The rule must be simple
 - Occam's razor

Occam's Razor

- If something can be explained by two hypotheses A and B, and if A is simpler than B, then we should *prefer* A over B
- Razor?
 - chops off complicated explanations such as the hypothesis B above
- Why we would care?
 - Complex rules (lots of IF conditions) are likely overfit to the train data, which is bad.



William of Ockham
(1287-1347 Surrey UK)

Dataset Properties

- We have 14 training instances
- We have four features
 - $\text{outlook} \in \{\text{sunny}, \text{overcast}, \text{rainy}\}$
 - $\text{temp} \in \{\text{hot}, \text{mild}, \text{cool}\}$
 - $\text{humidity} \in \{\text{high}, \text{normal}\}$
 - $\text{windy} \in \{\text{FALSE}, \text{TRUE}\}$
- Target prediction
 - $\text{play} \in \{\text{yes}, \text{no}\}$
 - A binary classification task

Tree Learning Algorithm

create an empty tree T

select a feature A from the set of features

create branches in T for each value v of A

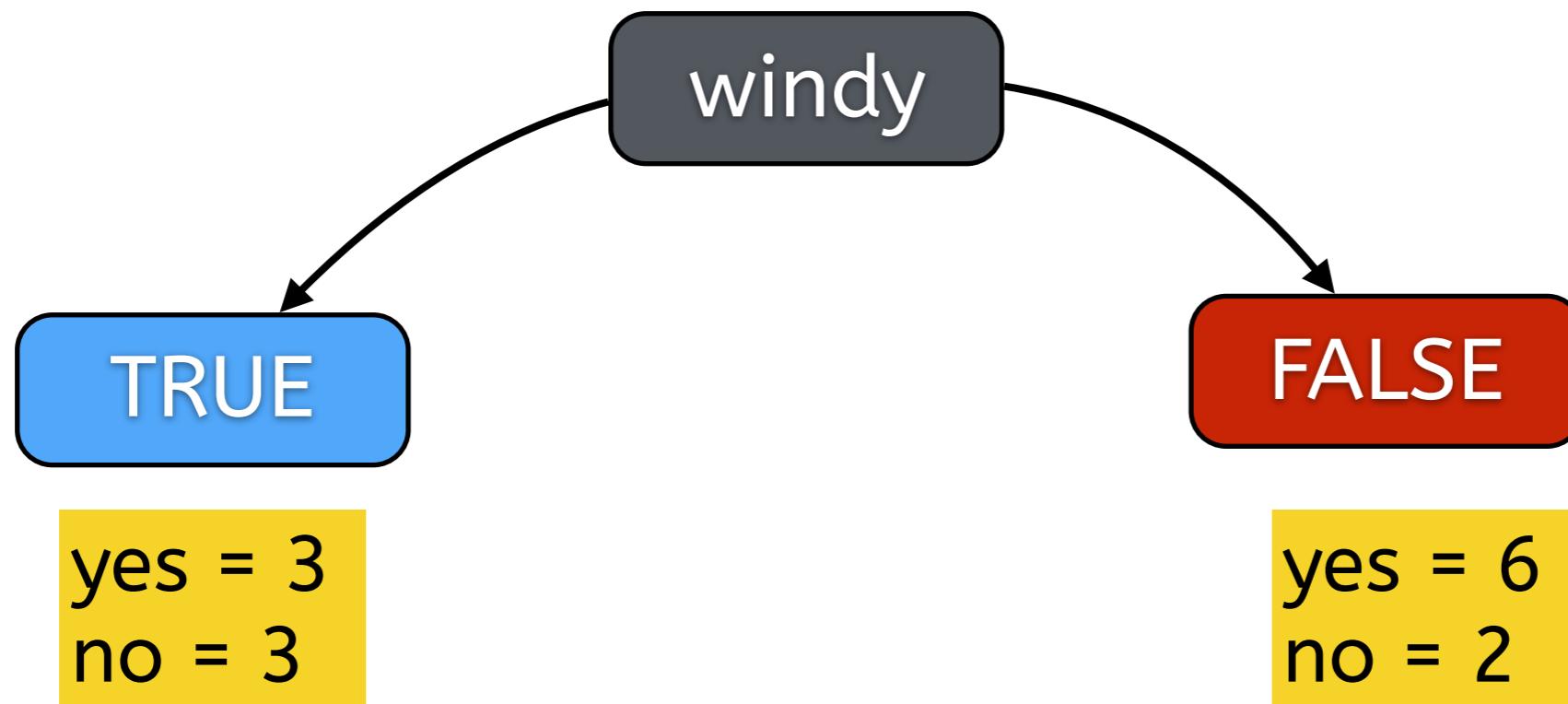
for each branch

recurse with instances where $A = v$

add tree as branch node

How to select a feature to branch?

- Lets make a random guessing algorithm and see how we can improve it
- Let us assume that we selected “windy” as the first feature (selected randomly among the four features in the dataset)



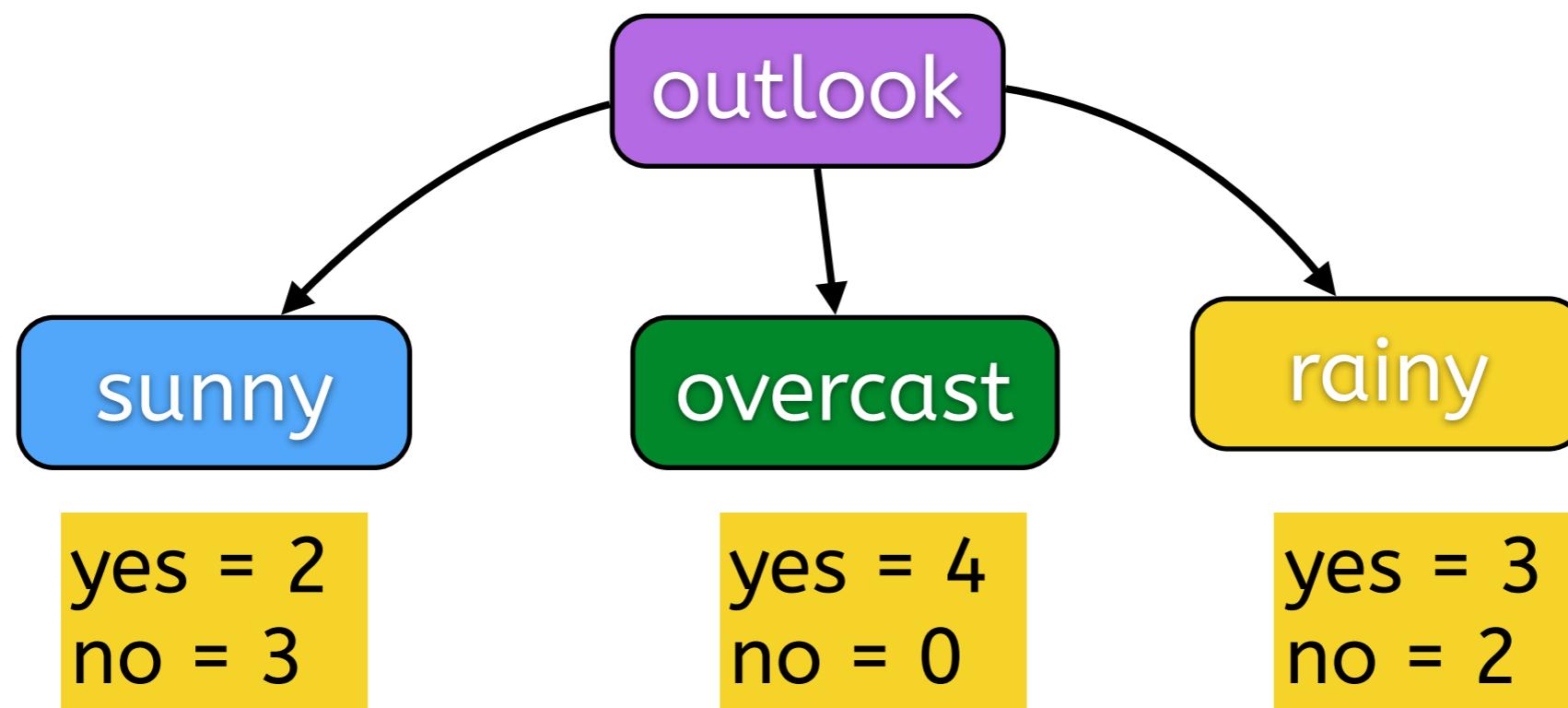
IF windy is TRUE or FALSE, we cannot identify a situation where play will always be yes or no.
Can we find a better feature that splits yes/no “purely”?

Dataset

outlook	temperature	humidity	windy	play?
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Lets make a second random guess

- Let us assume we selected “outlook” as the first feature (selected randomly among the four features in the dataset)



Dataset

outlook	temperature	humidity	windy	play?
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	TRUE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Rule we can learn

- IF outlook = overcast THEN play = yes
- This rule matches 4 out of 14 instances
 - coverage = $4/14 = 0.28$
- When it matches, it is perfect!
 - accuracy = $4/4 = 1.0$
- But what about the $14-4 = 10$ instances for which this rule does not match?
 - This rule cannot be used to classify those 10 instances.
 - We need to look into the other features as well
- Features that give us “pure” splits are better because we can “divide and conquer” using such features more efficiently!

Can we express “purity” empirically?

- (yes=4, no=0) is a more pure split than (yes=2, no=2). Can we make this intuition empirical?
- There are numerous measures that would evaluate how disproportionate a split is. One very popular such measure is the **entropy**

Entropy

- Is a measure of “surprise”
- How surprised will you be if you hear that you got a red ball when you randomly picked a ball from a bag that has 3 red balls and 3 blue balls vs. when you pick a ball from a bag that has 5 red balls and 1 blue ball?
- A measure of the amount of information that we get when we hear someone picked a red ball from the type of bags we discussed above

Entropy— Definition

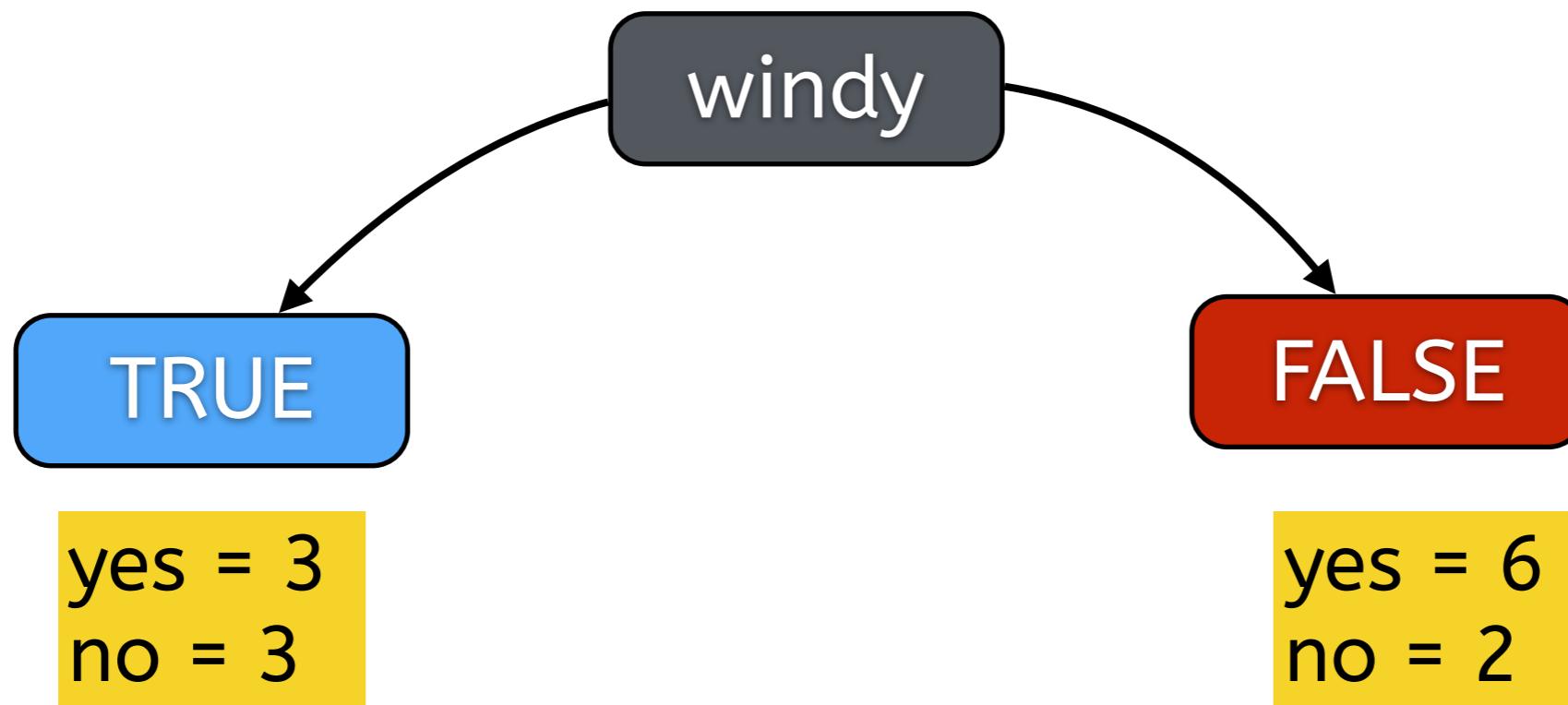
$$H(p) = - \sum_{i=1}^n p_i \log_2 p_i$$

- red balls = 3 and blue balls = 3
 - $p(\text{red}) = 3/6 = 0.5$
 - $p(\text{blue}) = 3/6 = 0.5$
 - $H(p) = - (0.5 \times \log(0.5) + 0.5 \times \log(0.5)) = 1$
 - $\log(0.5) = -1$ for base 2 logarithm)
 - We get 1 byte of information (the maximum in this case) when we hear some one picked a red ball from this bag because anything is possible (50-50 chance) and we have no idea before hand what the outcome of this random draw would be.

Back to the example

- Play=no for 5 out of the 14 instances in our dataset.
- Therefore, the original entropy (before we do any branching) is:
- $-5/14 * \log(5/14) - 9/14 * \log(9/14) = 0.9402$

If we select “windy” as the first feature for branching



$$\begin{aligned} & -\frac{3}{6} \log(\frac{3}{6}) - \frac{3}{6} \log(\frac{3}{6}) \\ &= 1 \end{aligned}$$

$$\begin{aligned} & -\frac{6}{8} \log(\frac{6}{8}) - \frac{2}{8} \log(\frac{2}{8}) \\ &= 0.811 \end{aligned}$$

We have 6 instances for $\text{windy}=\text{TRUE}$ and 8 instances for $\text{windy}=\text{FALSE}$. Therefore, the expected entropy is:

$$6/14 * 1 + 8/14 * 0.811 = 0.8919$$

$$\text{Information Gain (IG)} = 0.9402 - 0.8919 = 0.0483$$

Information Gain (IG)

- Information gain (IG) when using a feature f to branch the tree is defined as the **difference between** the amount of entropy we have **before we branch** using f and **after we branch** using f .
- We would prefer f that would yield the maximum information gain
 - If we can gain more information about the dataset using a particular feature f then we will select that feature first for branching
 - A greedy algorithm but works well in practice

Computing Entropy

```
import math

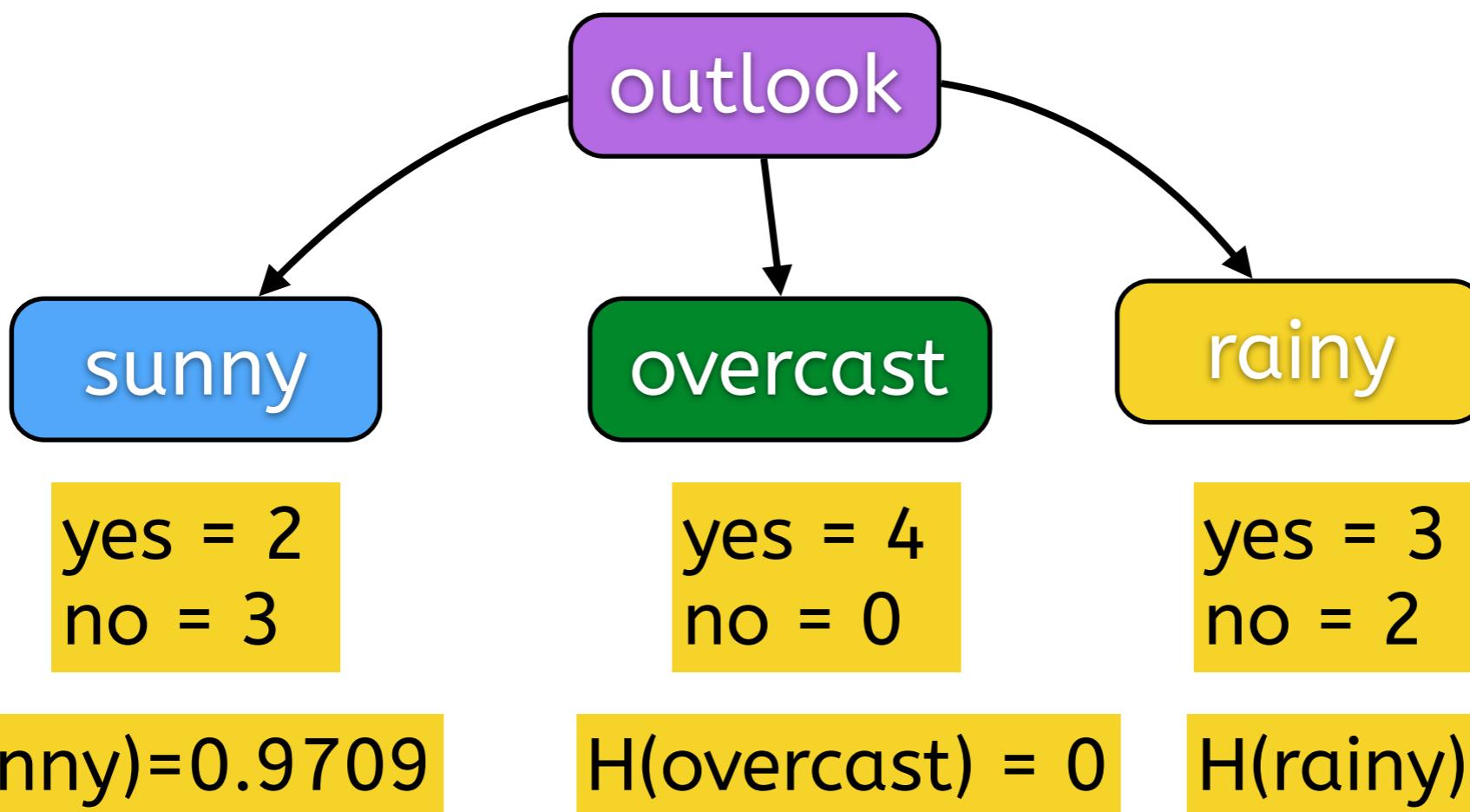
def log2(x):
    return math.log(x) / math.log(2)

def entropy(x, y):
    t = float(x + y)
    e = (x / t) * log2(x / t) + (y / t) * log2(y / t)
    return -e

if __name__ == "__main__":
    print entropy(2,6)
```

On the other hand...

- If we select “outlook” as the first feature



$$\begin{aligned}\text{Expected entropy} &= 5/14 * H(\text{sunny}) + 4/14 * H(\text{overcast}) + 5/14 * H(\text{rainy}) \\ &= 0.6935\end{aligned}$$

$$\text{Information Gain} = 0.9402 - 0.6935 = 0.2467$$

Dataset

outlook	temperature	humidity	windy	play?
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	TRUE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Dataset

outlook	temperature	humidity	windy	play?
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	TRUE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Computing Information Gain (IG)

```
import math

def log2(x):
    return 0 if x == 0 else math.log(x) / math.log(2)

def entropy(x, y):
    t = float(x + y)
    e = (x / t) * log2(x / t) + (y / t) * log2(y / t)
    return -e

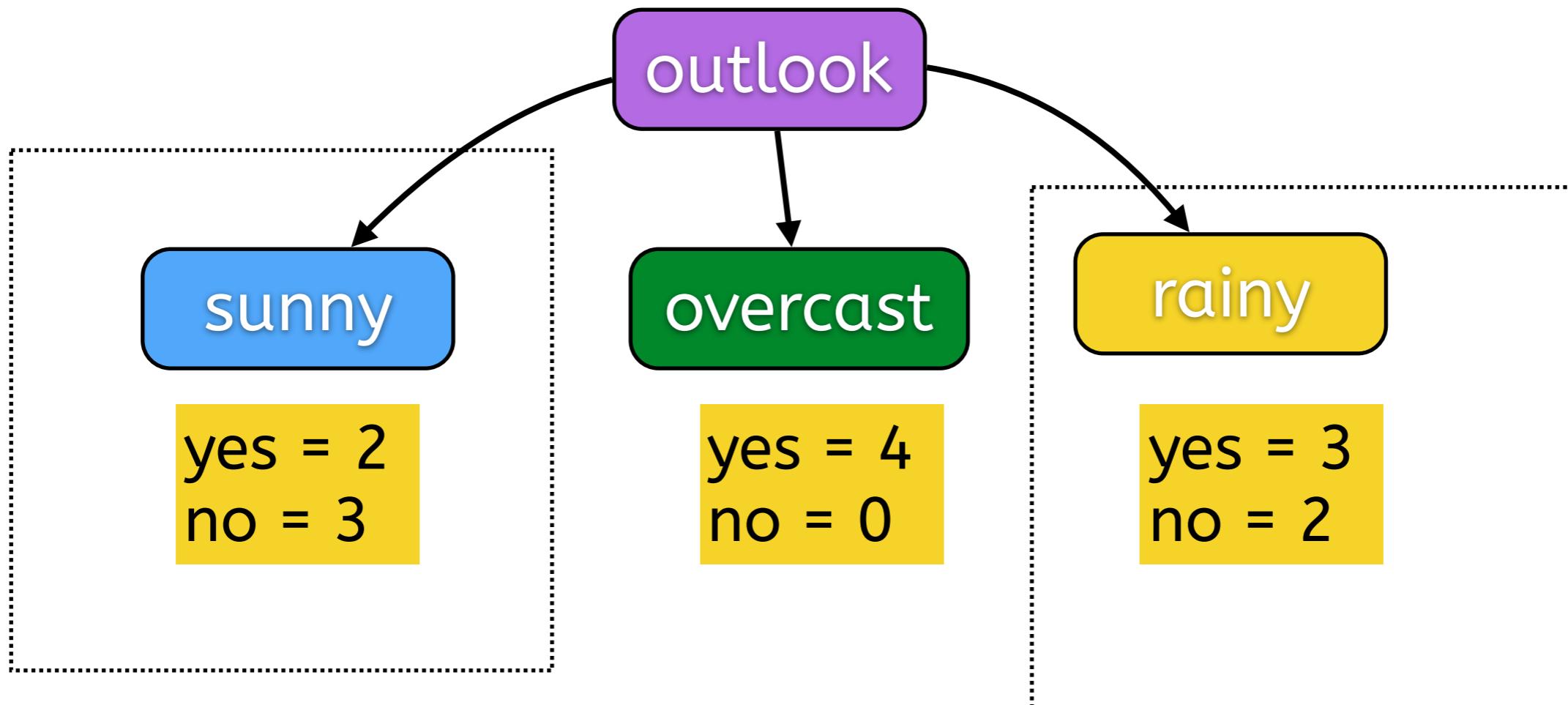
def IG(L):
    (xold, yold, tot) = (0, 0, 0)
    for (x, y) in L:
        xold += x
        yold += y
        tot += x + y
    original = entropy(xold, yold)
    evals = [entropy(x,y) for (x,y) in L]
    newval = 0
    for i in range(len(L)):
        newval += (float(sum(L[i]))) / float(tot)) * evals[i]
    return original - newval

if __name__ == "__main__":
    print "outlook =", IG([(3,2), (0,4), (2,3)])
    print "windy =", IG([(2,6), (3,3)])
    print "temp =", IG([(2,2), (2,4), (1,3)])
    print "humidity =", IG([(4,3), (1,6)])
```

Computing information gains

- $IG(\text{windy}) = 0.0483$
- $IG(\text{outlook}) = 0.2467$
- $IG(\text{temp}) = 0.029$
- $IG(\text{humidity}) = 0.1518$
- outlook is the clear winner here. By selecting outlook, we maximize the information gain

First branching



We need to further branch the “sunny” and “rainy” sub-trees.

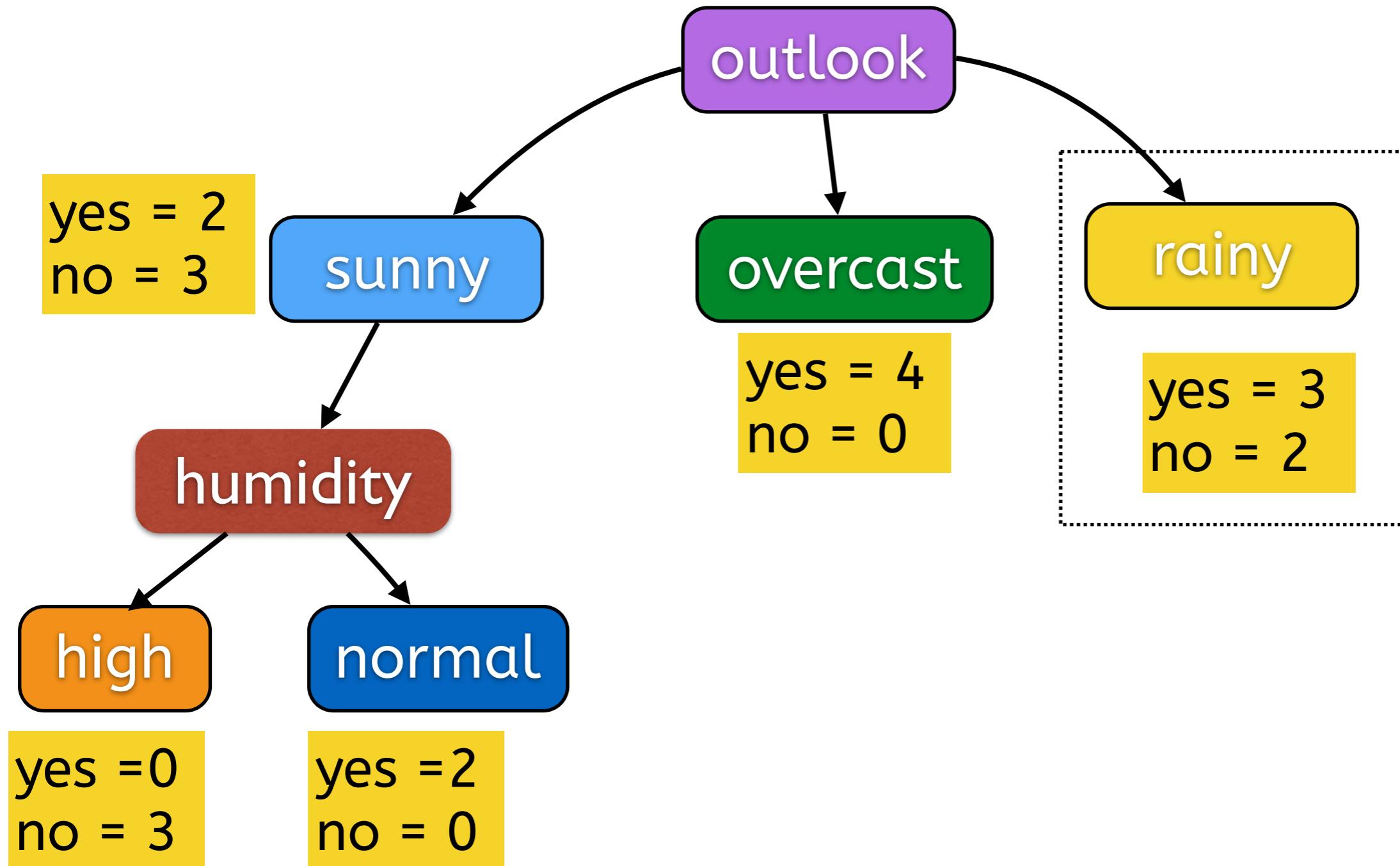
outlook=sunny dataset

outlook	temperature	humidity	windy	play?
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
sunny	mild	normal	TRUE	yes

No need for math here

- IF outlook=sunny AND humidity=high THEN play=no
- But lets do the math anyway...
 - $IG(\text{humidity}) = 0.9709$
 - $IG(\text{temp}) = 0.5709$
 - $IG(\text{windy}) = 0.0199$
 - “humidity” wins!

The Decision Tree so far...



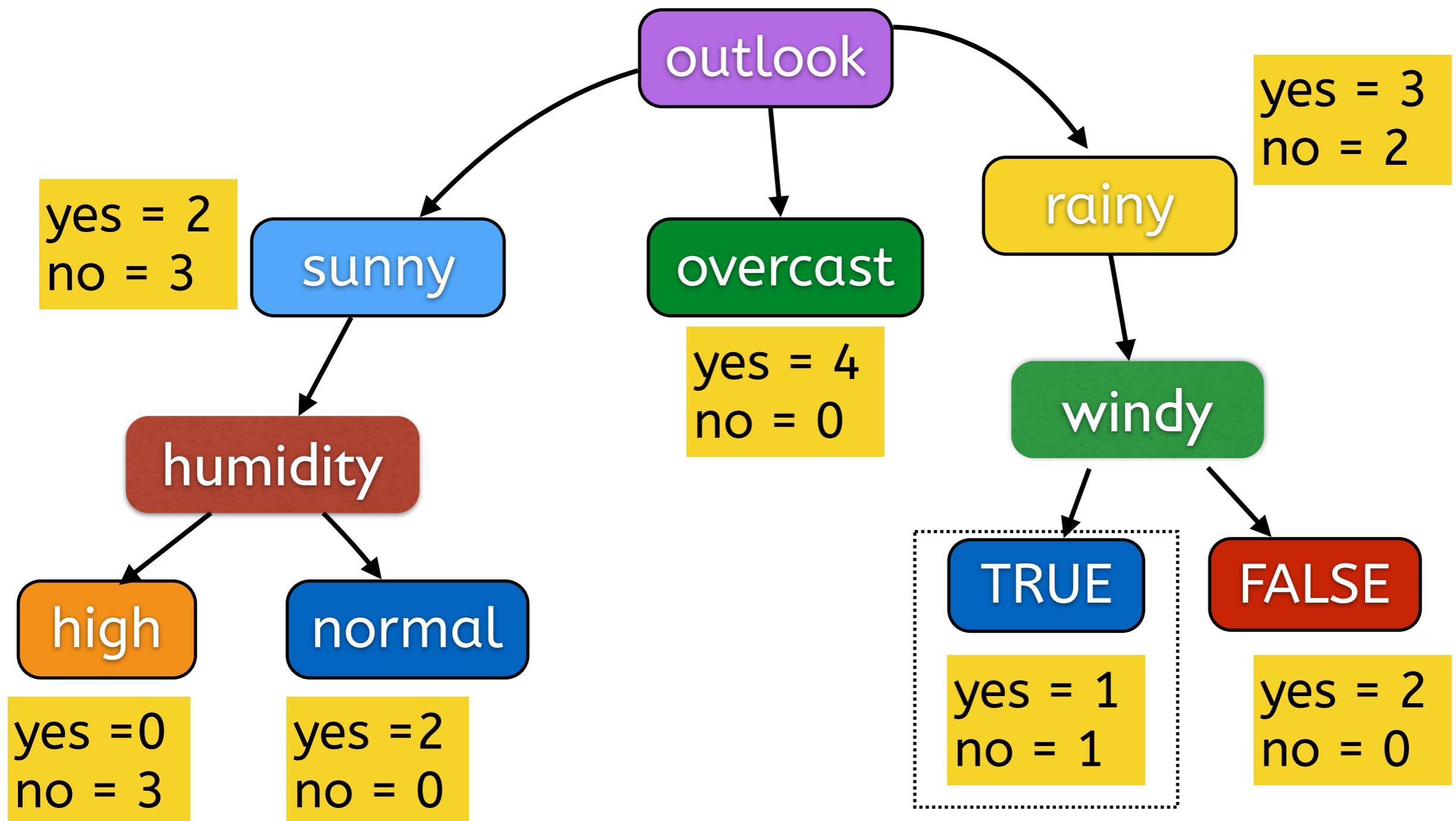
outlook=rainy dataset

outlook	temperature	humidity	windy	play?
rainy	mild	high	FALSE	yes
rainy	cool	normal	TRUE	yes
rainy	cool	normal	TRUE	no
rainy	mild	normal	FALSE	yes
rainy	mild	high	TRUE	no

Computing Information Gains

- $IG(\text{windy}) = 0.4199$
- $IG(\text{temp}) = 0.0199$
- $IG(\text{humidity}) = 0.0199$
- The largest IG is due to “windy”

Final Decision Tree



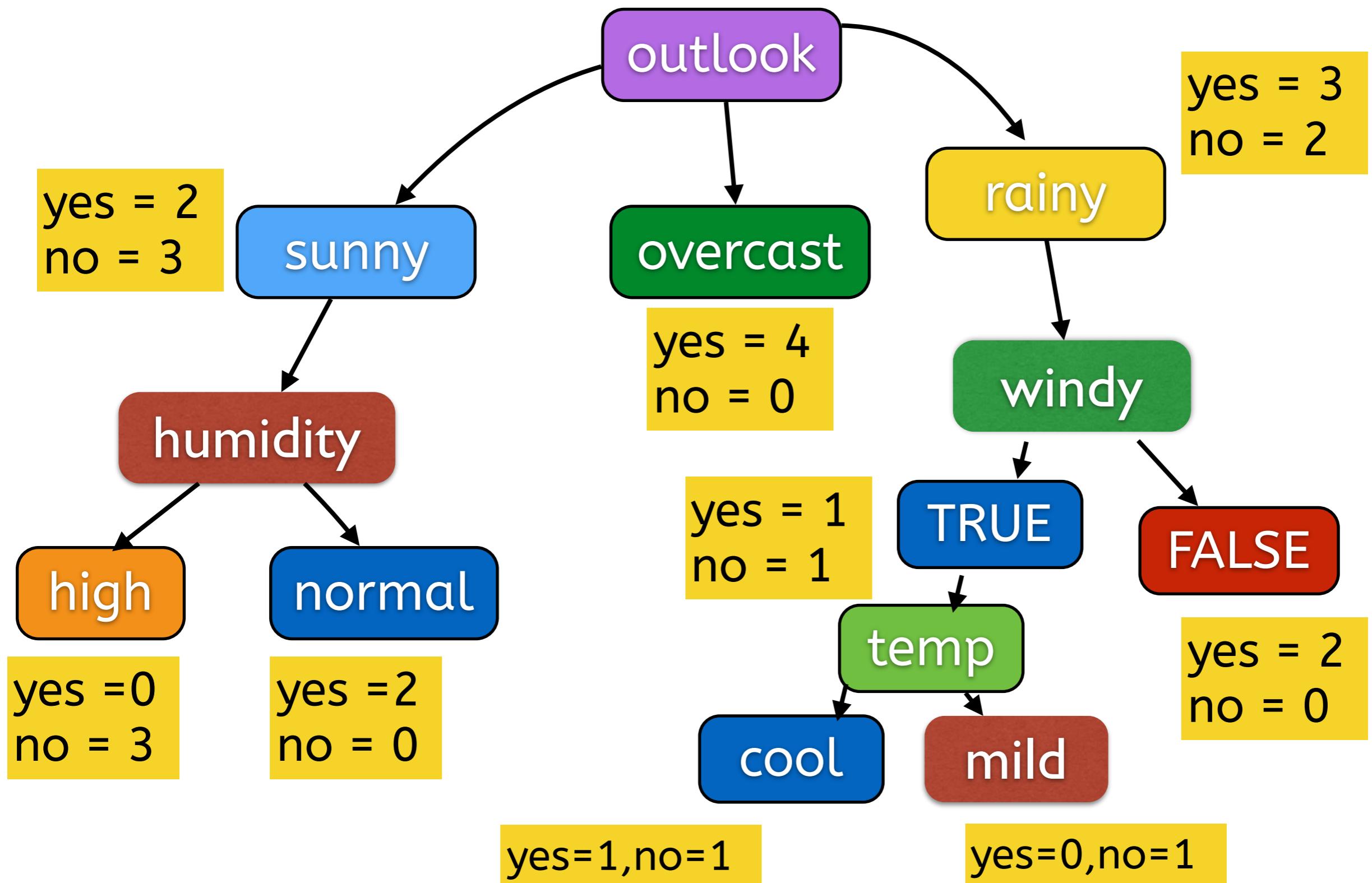
outlook=rainy AND windy=TRUE dataset

outlook	temperature	humidity	windy	play?
rainy	cool	normal	TRUE	yes
rainy	cool	normal	TRUE	no
rainy	mild	high	TRUE	no

Computing Information Gains

- $IG(\text{temp}) = 0.2516$
- $IG(\text{humidity}) = 0.2516$
- No clear winner here. It is a random guess between temp vs. humidity
- Lets select temp

Final Decision Tree



`outlook=rainy AND windy=TRUE AND temp = cool`

outlook	temperature	humidity	windy	play?
rainy	cool	normal	TRUE	yes
rainy	cool	normal	TRUE	no

Final feature selection

- We are only left with humidity
- But humidity is always “normal” for the remaining instances and we have one “yes” and one “no” for play.
- The dataset cannot be further branched
- All remaining instances (two in total) must be included as exceptions (remembered) to the rule

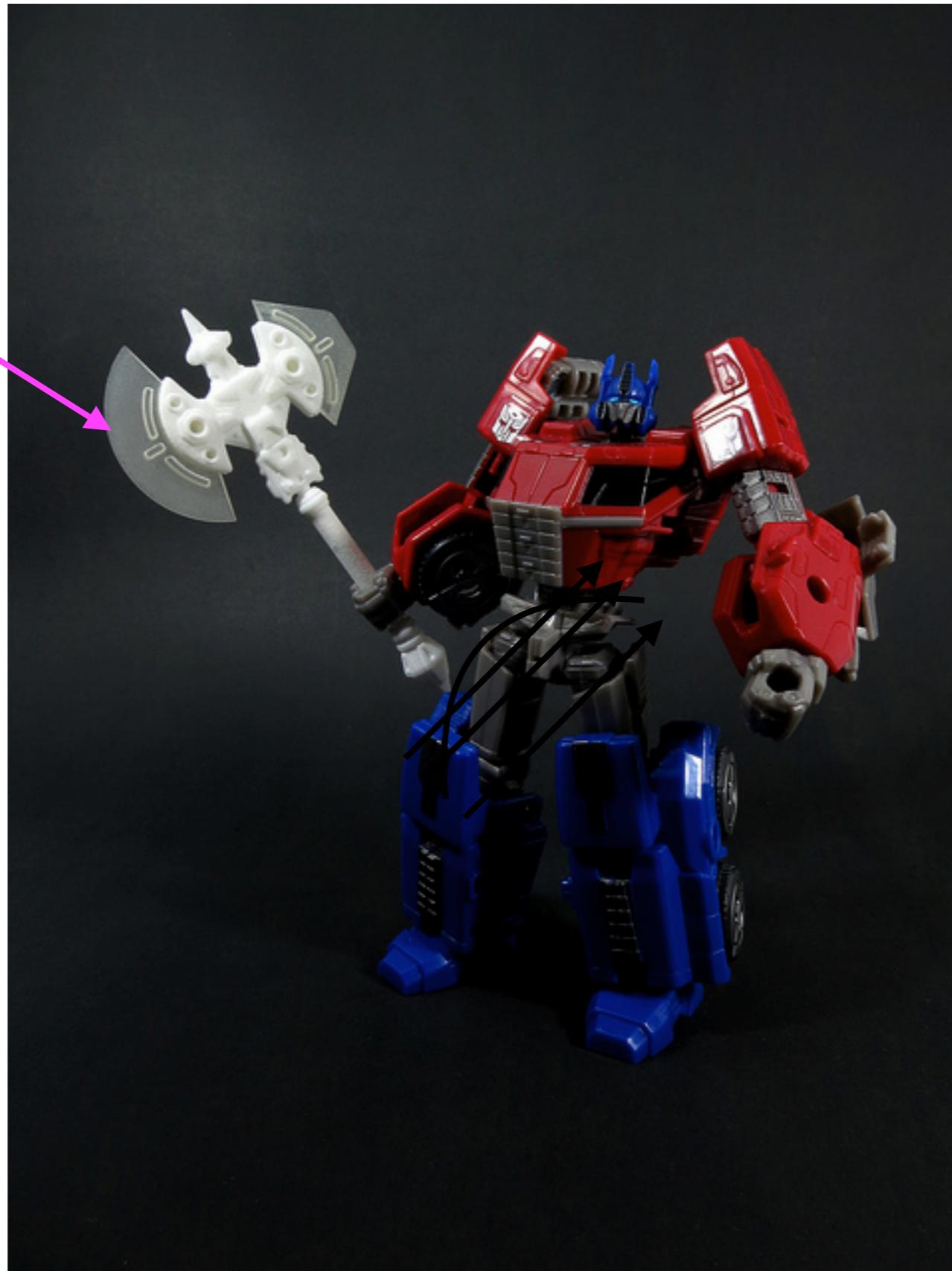
Final Rule

- IF (outlook = overcast) OR
- ((outlook = sunny) AND (humidity = normal))
OR
- ((outlook = rainy) AND (windy = FALSE))
 - THEN play = yes
 - ELSE play = no

ID3 Algorithm

- The algorithm that we just learnt is called the ID3 algorithm (Iterative Dichotomizer)
- Proposed by John Ross Quinlan
- J. R. Quinlan, *Induction of Decision Trees*,
Machine Learning, pp. 81-106, vol. 1, 1986.

Dichtomizer



Decision Tree Issues

- Ordering of Attribute Splits
 - As seen, we need to build the tree picking the best attribute to split first. (greedy)
- Numeric/Missing
 - Dividing numeric data is more complicated. We need to perform some form of a “binnig” (i.e. discretization) as a pre-processing step.
- Tree structure
 - A balanced tree with the fewest levels is preferable.
- Stopping criteria
 - When should we stop (to avoid overfitting)
- Pruning
 - It may be beneficial (speed/over-fitting) to prune the tree once created. We can do this after we have created the tree or while creating the tree (incrementally)

Quiz

- Learn a decision tree that predicts whether a car is fast from the following dataset.

Model	Engine	SC/Turbo	Weight	Fuel Eco	Fast
Prius	small	no	average	good	no
Civic	small	no	light	average	no
WRX STI	small	yes	average	bad	yes
M3	medium	no	heavy	bad	yes
RS4	large	no	average	bad	yes
GTI	medium	no	light	bad	no
XJR	large	yes	heavy	bad	no
S500	large	no	heavy	bad	no
911	medium	yes	light	bad	yes
Corvette	large	no	average	bad	yes
Insight	small	no	light	good	no
RSX	small	no	average	average	no
IS350	medium	no	heavy	bad	no
MR2	small	yes	average	average	no
E320	medium	no	heavy	bad	no

Information Retrieval

Danushka Bollegala

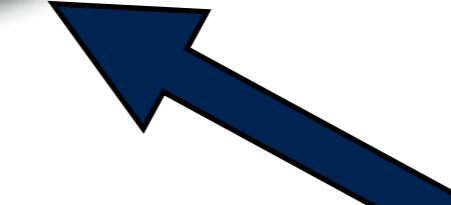
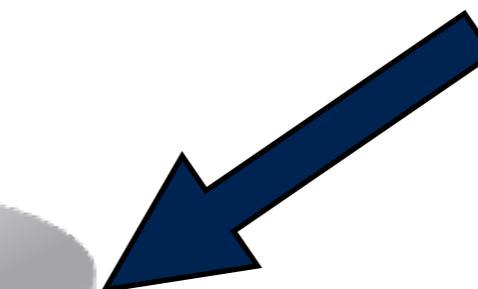
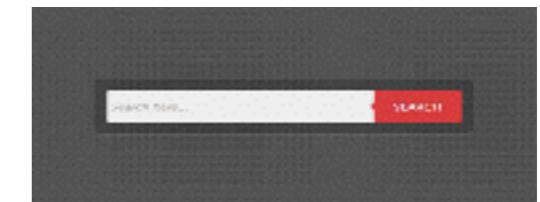


Anatomy of a Search Engine

Document
Indexing



Query Processing



Results
Ranking



Document Processing

- Format detection
 - Plain text, PDF, PPT, ...
- Text extraction
 - Convert to plain text
- Language detection
- Tokenisation
- Indexing

Language Detection

- How to detect the language of a document?
 - Check for specific characters such as kanji characters for Chinese, and Hiragana/Katakana for Japanese, Hangul for Korean etc.
- Not always perfect
 - Some documents might contain multiple languages
 - e.g. Japanese web site that teaches English
 - Character-based statistical approaches are used
 - If we get the language wrong, we will use a wrong tokeniser

Tokenisation

- If we do not tokenise properly, then we cannot search for those terms!
- Tokens vs. Words
 - Token refers to a single unit of text that we can search for
 - *the burger i ate was an awesome burger*
 - tokens = the, burger, i, ate, was, an, awesome, burger
 - Tokens might not necessarily be words in English
 - Repeating tokens are counted separately
 - note the two *burger* tokens in the previous example

Is tokenisation simple?

- Simple! Just split at spaces to get tokens
 - `s.split()`
- What about the following?
 - Dr. D. T. Bollegala
 - Should we consider *Dr.*, *.*, *D.*, *.*, *T.*, *.*, *Bollelala* or *Dr.*, *D.*, *T.*, *Bollelala*?
- Japanese and Chinese languages do not use spaces at all!
 - 私は学校に行きました.
 - Tokens: 私/は/学校/に/行きました/.

Indexing

- Search engines create an *inverted index* from tokens to documents for efficient retrieval
- Similar to the indexes you find at the end of a text book
 - *Support Vector Machines* p. 13, p. 56, p. 124
- Index is a large table between tokens and unique document ids

Inverted Index

$D_1 = I$ went to school

$D_2 =$ The school was closed

$D_3 =$ Tomorrow is a holiday

I

went

school

tomorrow

holiday

closed

to

D_1

D_1

D_1 D_2

D_3

D_3

D_2

D_1

The list of document IDs for a particular token is called a *posting list*

Notes

- We can assign integer IDs to documents so that we can sort the posting lists.
- By doing so we can quickly answer AND queries.
- We can assign integer ids to terms (tokens) as well. This will save space.
- We only need to store one entry for a word in a document. (multiple entries can be ignored, *bag-of-words* model).
- We need to store the length of a posting list as meta data.
- AND queries
 - Start with the shorter posting list.
 - Find the document ids in the shorter list in the longer list.

Example

Query = *Brutus AND Calpurnia*

Results = 2, 31

Brutus → [1 | 2 | 4 | 11 | 31 | 45 | 173 | 174]

Caesar → [1 | 2 | 4 | 5 | 6 | 16 | 57 | 132 | ...]

Calpurnia → [2 | 31 | 54 | 101]

:

Dictionary

Postings

Start with the posting list for Calpurnia (shorter list), and check for 2, 31. No point checking any further than 173. We will not find 54 and 101 in Brutus.

AND query processing

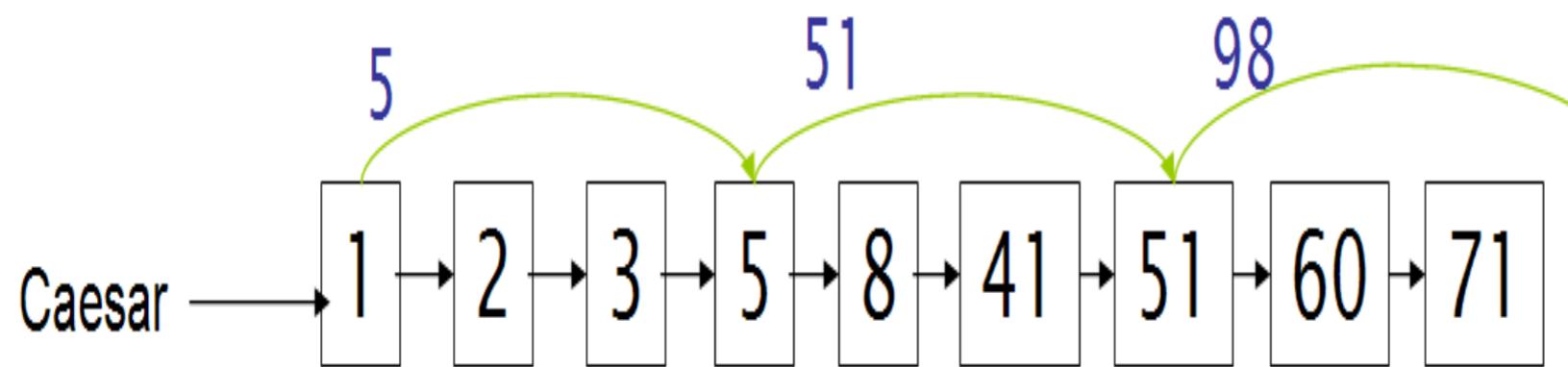
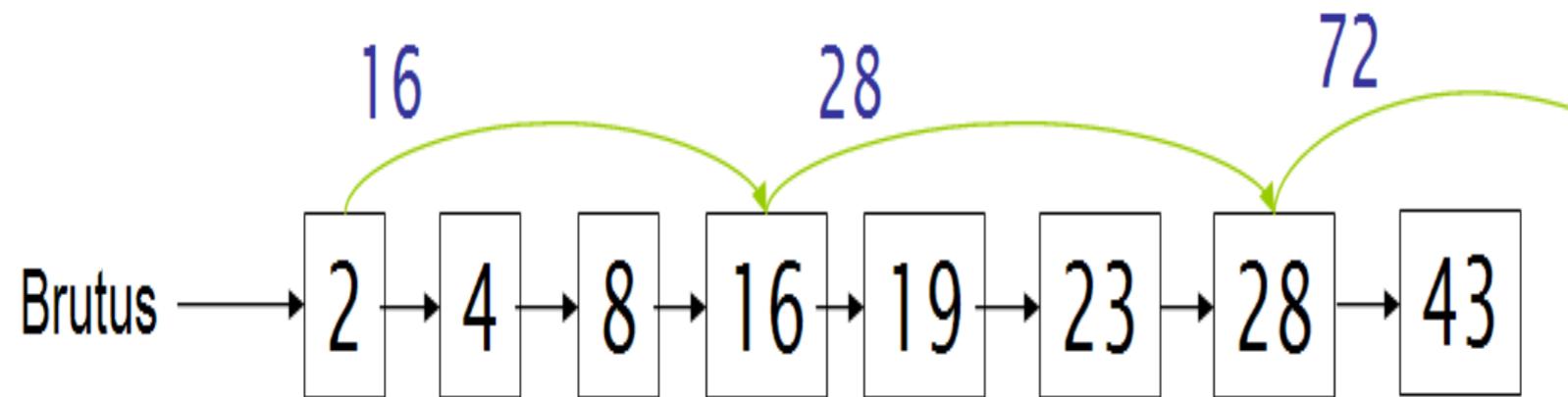
$\text{INTERSECT}(p_1, p_2)$

```
1  answer  $\leftarrow \langle \rangle$ 
2  while  $p_1 \neq \text{NIL}$  and  $p_2 \neq \text{NIL}$ 
3  do if  $\text{docID}(p_1) = \text{docID}(p_2)$ 
4      then  $\text{ADD}(\text{answer}, \text{docID}(p_1))$ 
5           $p_1 \leftarrow \text{next}(p_1)$ 
6           $p_2 \leftarrow \text{next}(p_2)$ 
7      else if  $\text{docID}(p_1) < \text{docID}(p_2)$ 
8          then  $p_1 \leftarrow \text{next}(p_1)$ 
9          else  $p_2 \leftarrow \text{next}(p_2)$ 
10 return answer
```

Skip pointers

- The previous version of answering AND queries is inefficient.
 - $O(n+m)$ if the length of the two posting lists are n and m .
- We can add *skip pointers* to speed up the search.
- If the value to be searched for is larger than the skip pointer then we can directly skip over all the values under the skip pointer.
- How to find skip points? (square root heuristic)
- Trade-off:
 - number of skips vs. skip range

Example



After matching up to 8, and when we want to match 41 next, we note that at 16 we have a skip of 28. This means that we will not observe 41 during this skip range. We can skip over 19 and 23, and resume the search process from 28. We cannot skip to 72 because 41 is in between 28 and 72.

Disk access vs. Memory access

- Disk seek time = 5ms
- disk read per 1b = 2×10^{-8} s
- memory read per 1b = 10^{-9} s
- Reading from memory is faster compared to disk.
- We need to read in blocks (ca. 64kb) when we read from the disk because of the seek overhead.
- Main memory is limited (10~100GB) vs. disk space (1~10TB)

Blocked Sort-Based Indexing

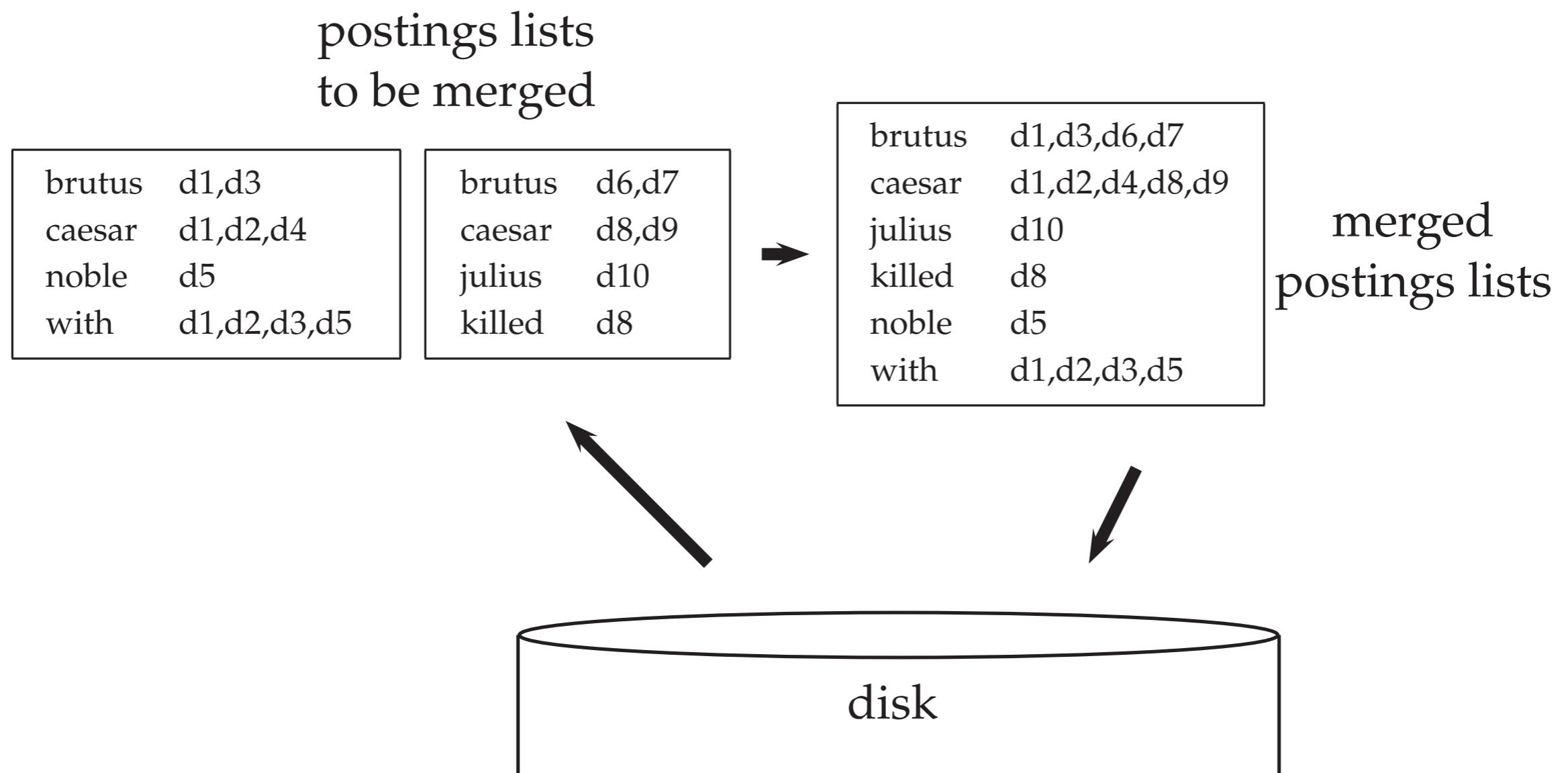
- BSBI (Blocked Sort-based Indexing)
 - Segment the document collection into parts of equal size
 - Sort the termID-docID pairs for each block in memory
 - Store intermediate sorted results on disk
 - Merge all intermediate results into the final index

BSBI

BSBINDEXCONSTRUCTION()

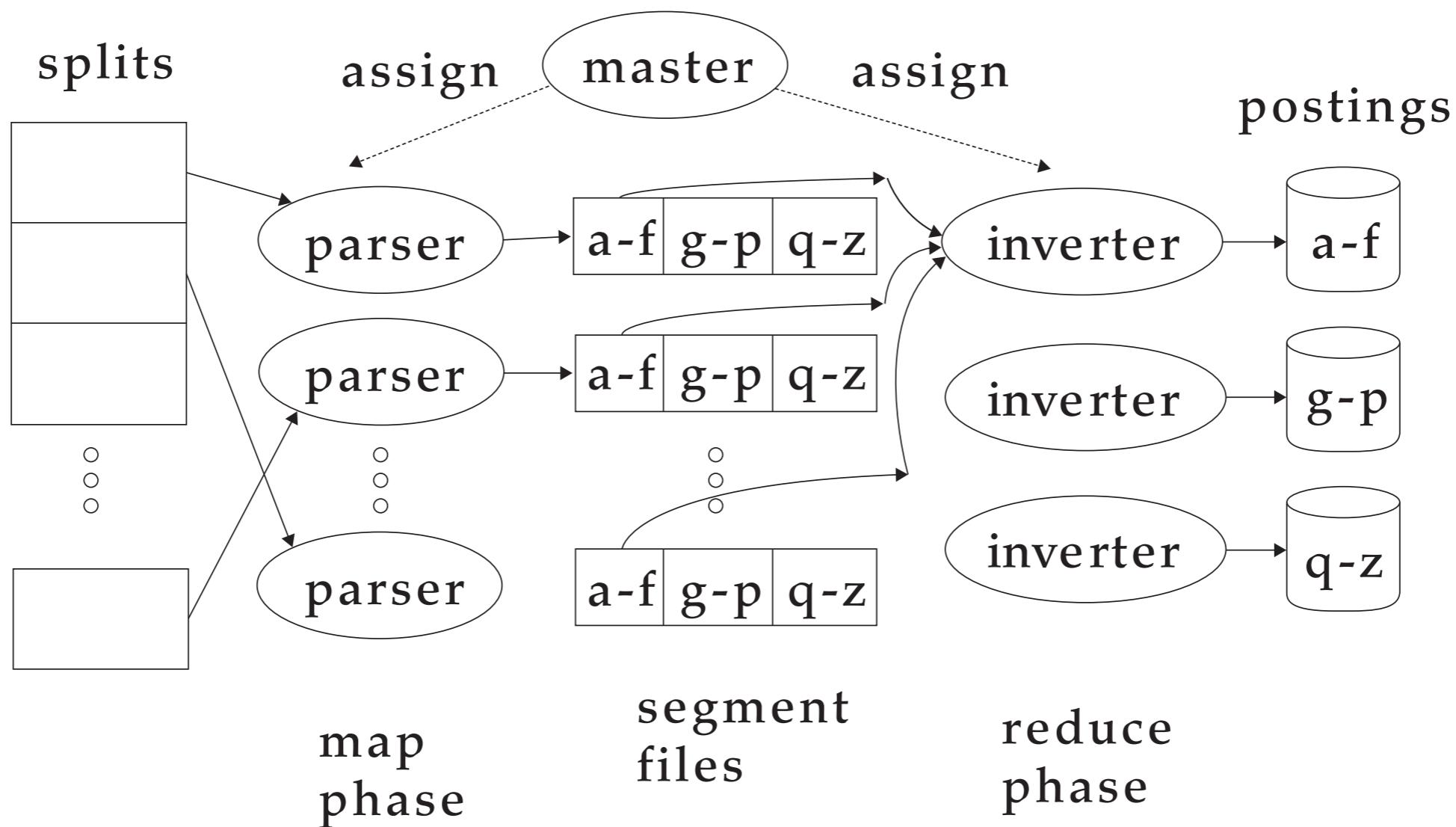
- 1 $n \leftarrow 0$
- 2 **while** (all documents have not been processed)
- 3 **do** $n \leftarrow n + 1$
- 4 $block \leftarrow \text{PARSENEXTBLOCK}()$
- 5 BSBI-INVERT($block$)
- 6 WRITEBLOCKTODISK($block, f_n$)
- 7 MERGEBLOCKS($f_1, \dots, f_n; f_{\text{merged}}$)

Example: BSBI



Web scale indexing

- Most large documents collections (e.g. Web) result in indexes that cannot be stored in a single machine
- Distributed indexing methods are required
- Methods based on MapReduce are used.



Ranking

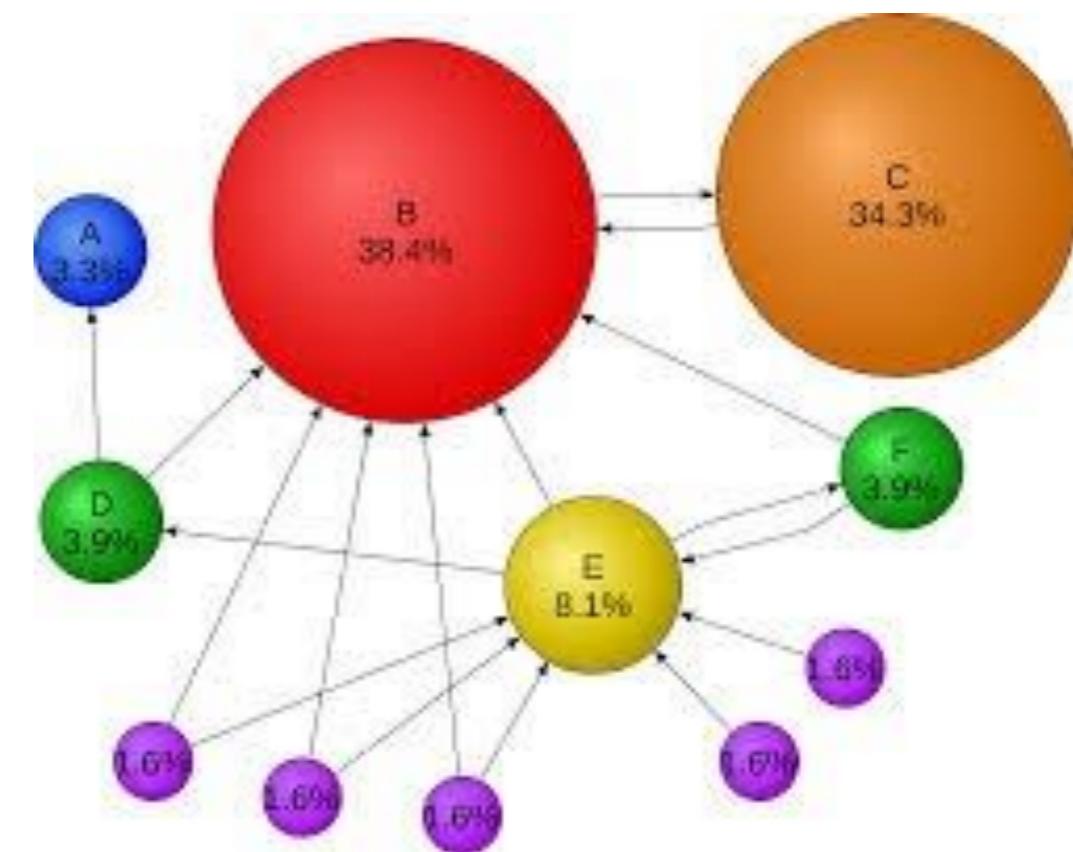
- Often there are hundreds of documents that contain a particular query
- We must rank the search results according to their *relevance* to a query
- There are numerous factors that need to be considered when computing $f(q,d)$, the relevance of a document d to a query q .

Static Ranking

- The ranking of a document independent of the query
 - PageRank is a famous example of a static ranking algorithm

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

Discussed later in our
Graph Mining lecture



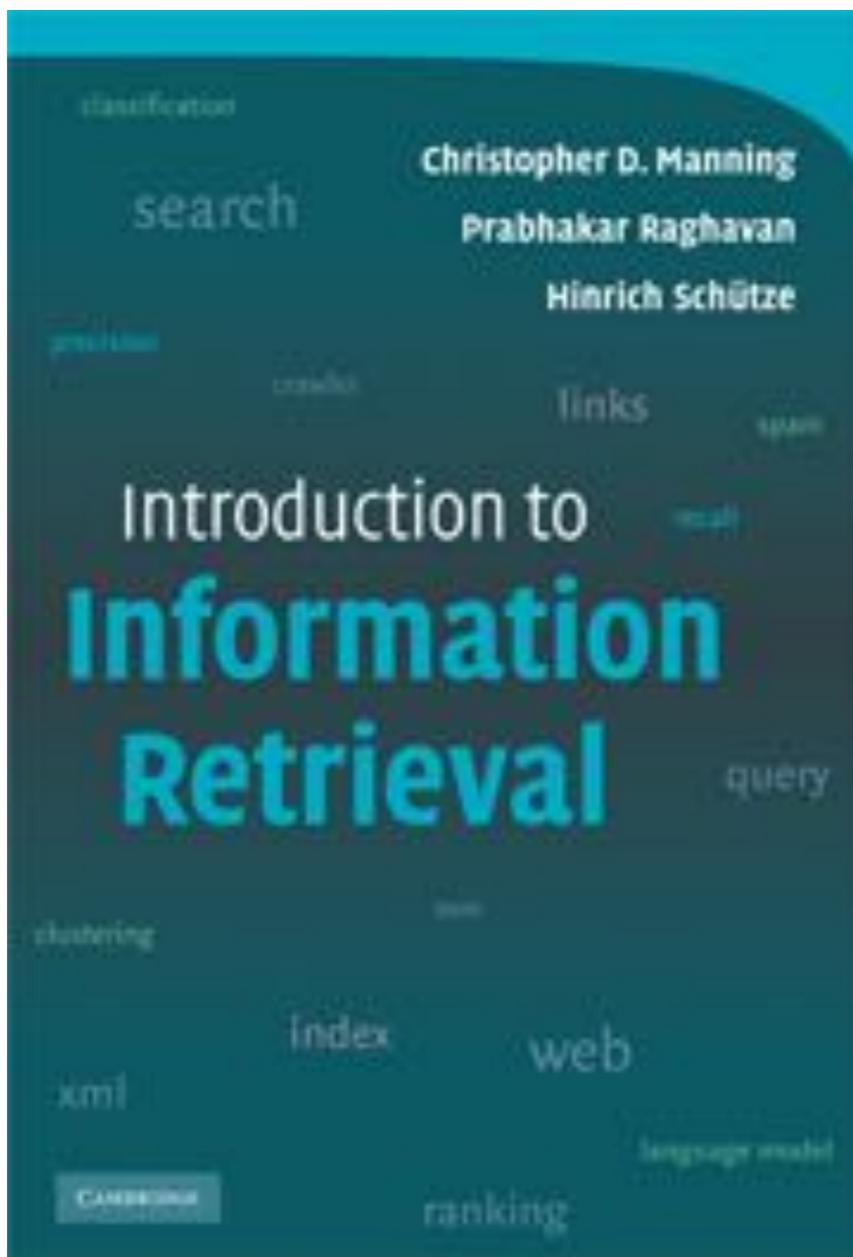
Dynamic Ranking

- The rank of a document depends on the query
- Features
 - term frequency
 - PageRank
 - novelty of the document
 - position of the query within the document
 - title, anchor text, links, etc.
- $f(q,d)$ is computed as the linear combination of numerous features that indicate relevance
 - $f(q,d) = \mathbf{w}^\top \boldsymbol{\Psi}(q,d)$

How to learn the relevance weights?

- Clickthrough data are collected by the search engines
- Assume that we entered a query q and obtained a ranked list of documents d_1, d_2, d_3 .
- If we skip d_1 and clicked on d_2 , then the search engine creates a training instance indicating that $f(q,d_2) > f(q,d_1)$
- Billions of people are using search engines and clicking on documents, giving a large and cheap training dataset to learn the relevance function f .

References



PDF available here.

<http://www-nlp.stanford.edu/IR-book/>

k-NN Classifier

Classification vs. Regression

- In classification
 - We are given a dataset $\{x, y\}$ where each instance x must be classified into a pre-defined finite (and often small) **unordered set** of classes y . We are required to learn a function $f(x)$ that predicts the **class** y , given x .
 - Example
 - In binary classification $f(x) = \text{sgn}(ax + b) \in [-1, +1]$
- In regression
 - We are given a dataset $\{x, y\}$ where each instance x is associated with a **real number** y , and we are required to learn a function $f(x)$ that predicts the **value** of y , given x .
 - Example
 - In linear regression $f(x) = ax + b$
 - Note that in “ordinal regression” we have integer values instead of real-values. There is a clear total ordering among the target values. Therefore, ordinal regression is “regression” and not “classification”.

Classification Algorithms

- There are loads of them...
 - k-NN, Perceptron, Naive Bayes, logistic regression, SVM, neural networks, decision trees, random forests,...
- When you learn new classification algorithms look for two main points
 - What is the **loss function** that the classifier minimizes?
 - What **optimization method** does it use for the minimization of the loss function?

World's simplest classifier!

- Given a training dataset D_{train} of N instances $\{\mathbf{x}, y\}$, simply “remember” the entire N instance in the memory (or hard-disk)
 - memory-based learning
- When we want to classify a test (previously unseen, not in D_{train}) instance \mathbf{x}^* , we would simply check whether \mathbf{x}^* is in D_{train}
 - if $\mathbf{x}^* \in D_{\text{train}}$ the return the label of \mathbf{x}^*
 - otherwise make a random guess

Quiz 1

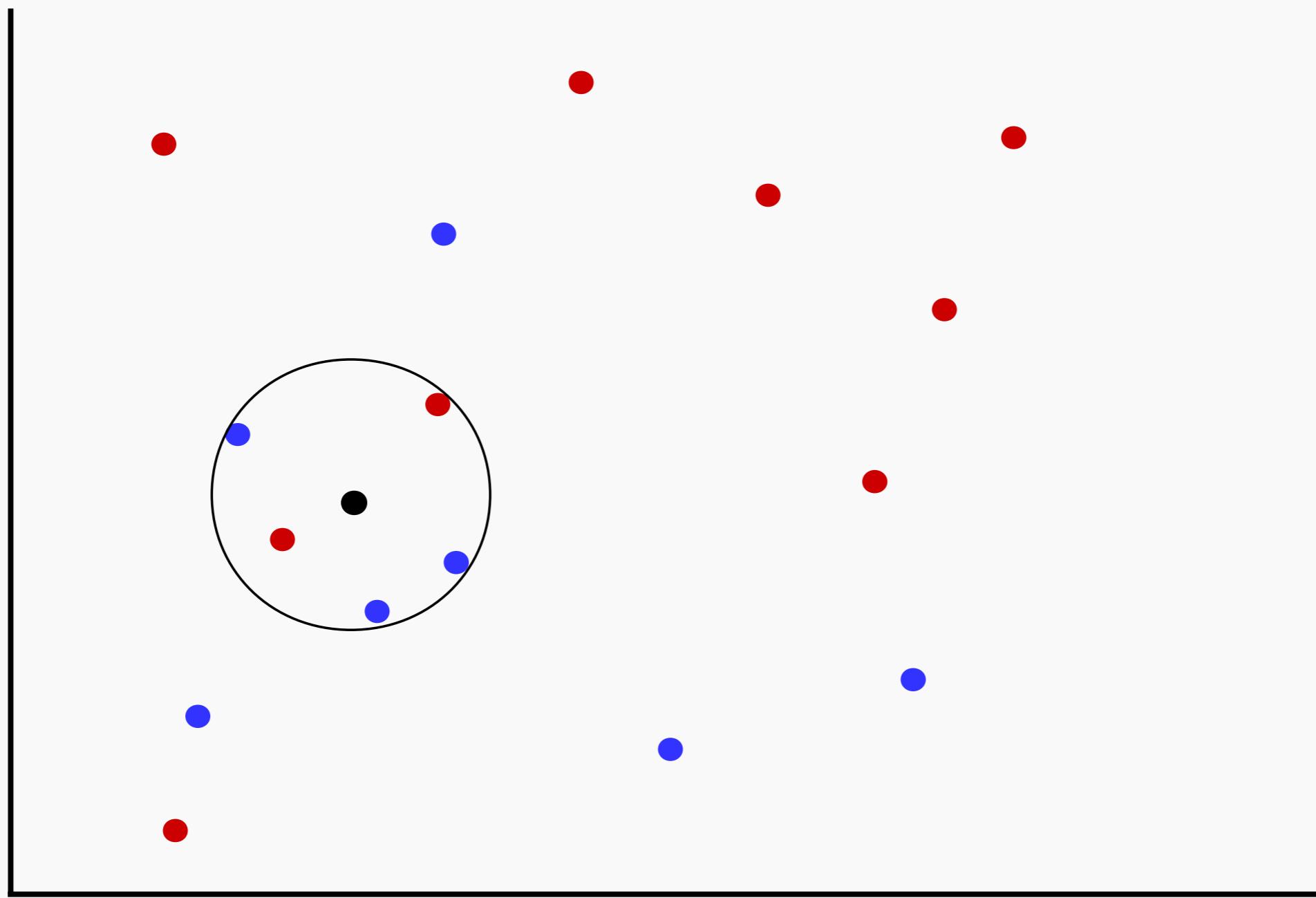
- What is the problem with the memory-based learner described in the previous slide?

k-Nearest Neighbour Classifier

1. Find the k closest (nearest) instances (neighbours) to the test instance
2. Find the majority label among those nearest neighbours
3. The majority label in the nearest neighbours is predicted to the test instance

Example

5-NN - find 5 closest to black point, 3 blue and 2 red, so predict blue



Measuring similarity/distance

- Various distance/similarity measures can be used
- Cosine similarity

$$\text{CosSim}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

$$\mathbf{x}^\top \mathbf{y} = \sum_{i=1}^D x^{(i)} y^{(i)}$$
$$\|\mathbf{x}\| = \sqrt{\sum_{i=1}^D x^{(i)2}}$$

Here, $x^{(i)}$ denotes the i -th element of the vector \mathbf{x}

Quiz 2

- Let $\mathbf{x} = (1, 0, -1)^\top$ and $\mathbf{y} = (-1, 1, 0)^\top$. Compute the cosine similarity between the two vectors \mathbf{x} and \mathbf{y} .

Euclidean Distance

- Euclidean distance between two vectors \mathbf{x} and \mathbf{y} is defined as follows

$$\text{EucDist}(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^\top (\mathbf{x} - \mathbf{y})} = \sqrt{\sum_{i=1}^D (x^{(i)} - y^{(i)})^2}$$

Quiz 3

- Let $\mathbf{x} = (1, 0, -1)^\top$ and $\mathbf{y} = (-1, 1, 0)^\top$. Compute the Euclidean distance between the two vectors \mathbf{x} and \mathbf{y} .

Manhattan Distance

- Manhattan (city block) distance between two vectors \mathbf{x} and \mathbf{y} is defined as follows

$$\text{ManDist}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_1 = \sum_{i=1}^D |x^{(i)} - y^{(i)}|$$

Here, $\|\mathbf{x}\|_1$ denotes L1 norm of the vector \mathbf{x}

Manhattan



Quiz 3

- Let $\mathbf{x} = (1, 0, -1)^\top$ and $\mathbf{y} = (-1, 1, 0)^\top$. Compute the Manhattan distance between the two vectors \mathbf{x} and \mathbf{y} .

Vector norms

- “norm” is a mathematical concept that expresses the “size/length” of a measure
- Popular vector norms in data mining are as follows

- L1 norm

$$\|\mathbf{x}\|_1 = \sum_{i=1}^D |x^{(i)}|$$

- L2 norm

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^D x^{(i)2}}$$

- L0 norm

$$\|\mathbf{x}\|_0 = \text{no. of non-zero elements in } \mathbf{x}$$

- L ∞ norm

$$\|\mathbf{x}\|_\infty = \max_i(x^{(1)}, x^{(2)}, \dots, x^{(D)})$$

Quiz 4

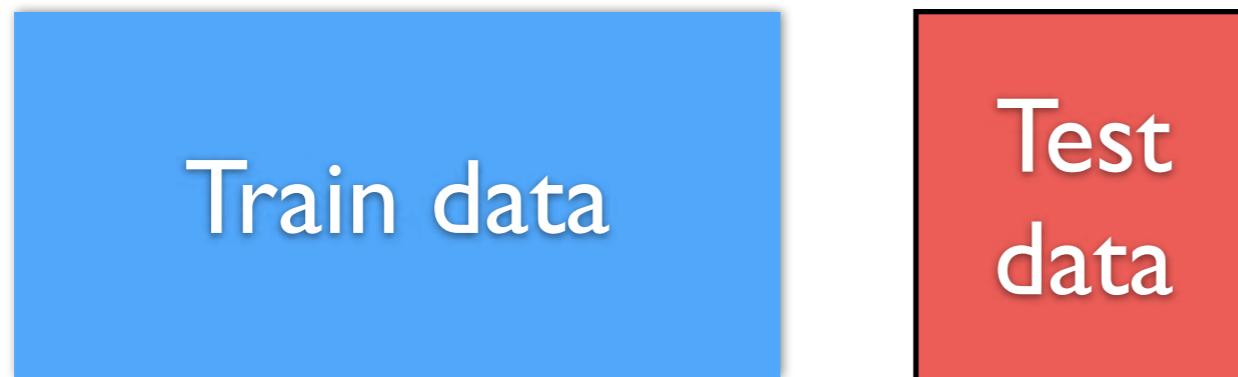
- Give the vector $\mathbf{x} = (1, 0, -1, 2)^\top$, compute the following $\|\mathbf{x}\|_1, \|\mathbf{x}\|_2, \|\mathbf{x}\|_0, \|\mathbf{x}\|_\infty$

k-NN classification

- Select odd k values to avoid ties
- What value to use for k?
 - Depends on the dataset size. Large and diverse datasets need a higher k, whereas a high k for small datasets might cross out of the class boundaries
 - Calculate accuracy on a validation set for increasing values of k, and use the best value on the test data

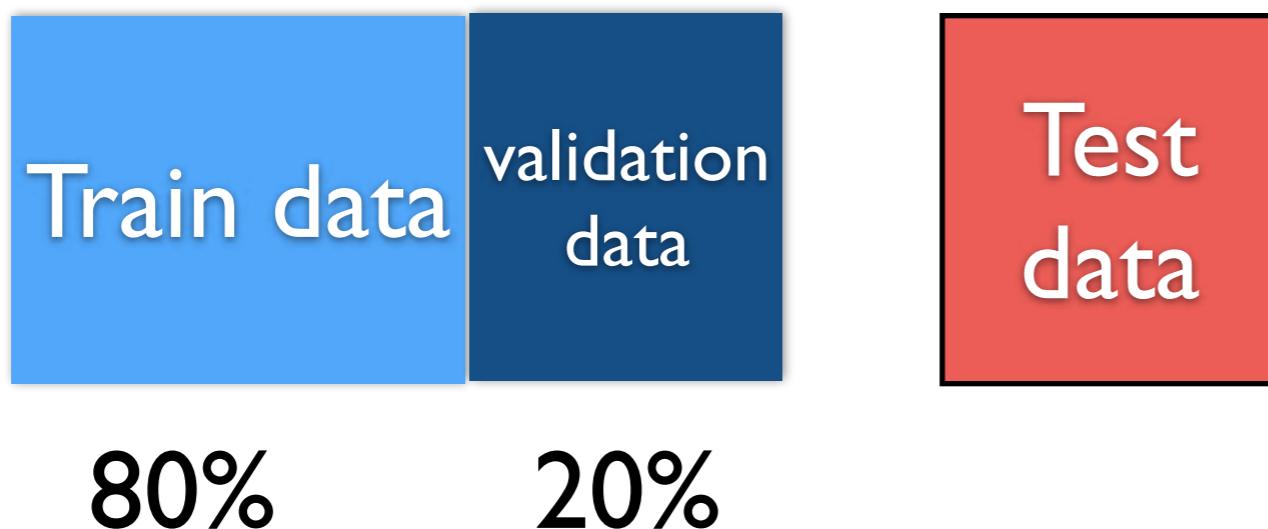
Train/Test/Validation datasets

- Validation data
 - Set aside (hold-out) a fraction of train data for validation purposes.
 - Hyper-parameters are often tuned using validation data
 - Hyper-parameter is a parameter that is NOT learnt during training, but is set BEFORE running the training algorithm. (e.g. k in k-NN is a hyper-parameter)
 - Using validation data to set hyper-parameters reduce overfitting
 - Of course, you CANNOT use test data for any tuning except for testing.



Train/Test/Validation datasets

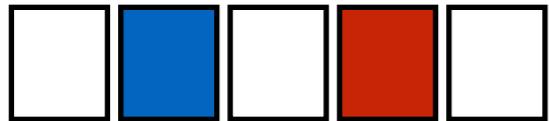
- Validation data
 - Set aside (hold-out) a fraction of train data for validation purposes.
 - Hyper-parameters are often tuned using validation data
 - Hyper-parameter is a parameter that is NOT learnt during training, but is set BEFORE running the training algorithm. (e.g. k in k-NN is a hyper-parameter)
 - Using validation data to set hyper-parameters reduce overfitting
 - Of course, you CANNOT use test data for any tuning except for testing.



Implementation considerations

- During train time
 - nothing to do
- During test time
 - Classification can be very slow when finding the k nearest neighbours.
 - In a trivial implementation it could require N number of comparisons, where N is the size of the train dataset
 - By using indexing we can speed up this look up process

Indexing trick in k-NN



Let us assume that the test instance has a red feature and a blue feature

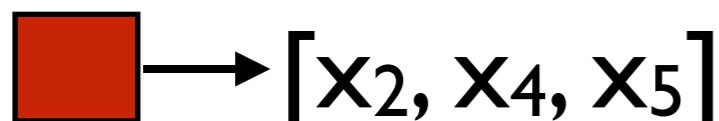
We would like to find train instances that are similar to this test instance

Obviously, only train instances that have at least a red or a blue feature will have non-zero similarity with this test instance

Let us create an index that lists which train instances have which color features



We only need to measure similarity between these train instances



Exploits the sparseness in feature vectors

Logistic Regression



Binary Classification

- Given an instance \mathbf{x} , we must classify it to either positive (1) or negative (0) class.
 - We can use $\{1, -1\}$ instead of $\{1, 0\}$
 - (but we will use 1, 0 as it simplifies the notation in subsequent derivations)
- Binary classification can be seen as learning a function f such that $f(\mathbf{x})$ returns either 1 or 0 indicating the predicted class.

Some terms in Machine Learning

- Training dataset with N instances
 - $\{(x_1, t_1), \dots, (x_N, t_N)\}$ This can also be written as $\{(x_n, t_n)\}_{n=1}^N$
- Target label (class)
 - t : The class labels in the training dataset
 - Annotated by humans (supervised learning)
- Predicted label
 - Labels predicted by our model $f(x)$

From Naive Bayes to Logistic Regression

- Recall Naive Bayes Classifier
 - Predict:

$$\hat{Y} = \operatorname{argmax}_y P(\mathbf{X} | Y) = \operatorname{argmax}_y P(\mathbf{X} | Y)P(Y)$$

- We use independence assumption:

$$P(\mathbf{X} | Y) = P(X_1, X_2, \dots, X_m | Y)P(Y) = \prod_{i=1}^m P(X_i | Y)$$

- Can we model $P(Y | \mathbf{X})$ directly?

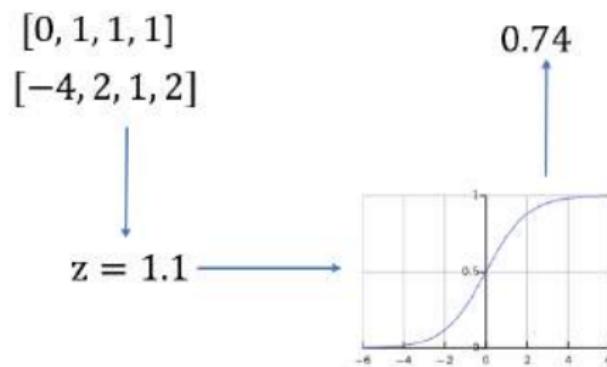
From Naive Bayes to Logistic Regression

- Can we model $P(Y | \mathbf{X})$ directly?
 - We use logistic regression model to achieve this
 - Given X, \mathbf{W} we compute $P(Y = 1 | X)$

$$\begin{matrix} [0, 1, 1, 1] \\ [-4, 2, 1, 2] \end{matrix} \xrightarrow{\hspace{10em}} 0.74$$

From Naive Bayes to Logistic Regression

- Can we model $P(Y | \mathbf{X})$ directly?
 - We use logistic regression model to achieve this
 - Given X, \mathbf{W} we compute $P(Y = 1 | X)$



Generative vs. Discriminative Classifier



Generative

- goal of understanding how dogs look and cats look
- model can 'generate' (draw) a dog
- given a test image, choose the closest model as 'label'

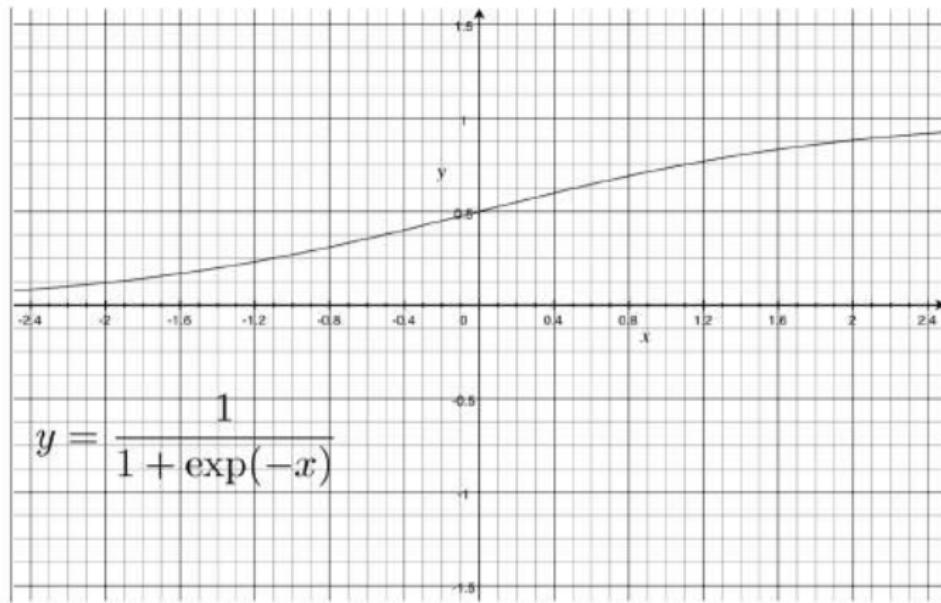
Discriminative

- trying to learn to distinguish the classes (not learning much about them)
- If one feature neatly separates the classes, the model is satisfied.
- e.g., dogs wearing collars and cat aren't

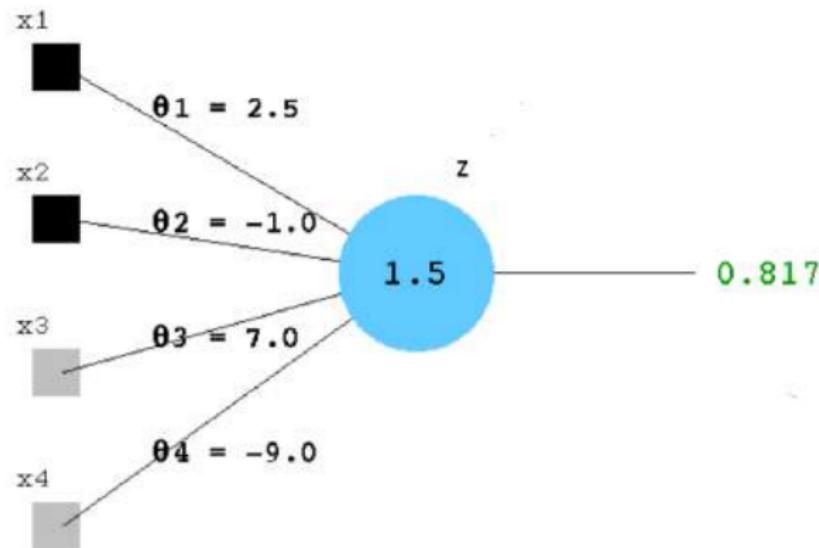
Logistic Regression

- is not a *regression* model
- is a *classification* model
- is a **Discriminative** classifier and not a **Generative** classifier
- is the basis of many advanced machine learning methods
 - neural networks, deep learning, conditional random fields,
...
- try to fit a logistic Sigmoid function to predict the class labels

Logistic Sigmoid Function



Logistic Regression Example



How to find Parameters of the Model

- In logistic regression, given \mathbf{x}, \mathbf{w} we compute $P(Y = y | \mathbf{x})$
 - How to compute \mathbf{w} ?
-
- Maximum Likelihood Estimate/Principle (MLE): parameter estimation method to find parameters of the model
 - The parameter values are found such that the values maximise the *likelihood* of making the observations given the parameters.

MLE Example

- Consider a bag containing 3 balls.
- Each ball is either red or blue (we have no other information)
- number of blue balls (θ) might be 0, 1, 2, or 3
- we are allowed to choose 4 balls at random with replacement.
- the random variables X_1, X_2, X_3 and X_4 are defined as follows:

$$X_i = \begin{cases} 1 & \text{if the } i\text{th chosen ball is blue} \\ 0 & \text{if the } i\text{th chosen ball is red} \end{cases}$$

MLE Example

- X_i 's are i.i.d and $X_i \sim Bernoulli\left(\frac{\theta}{3}\right)$.
- let us say the following values for X_i 's are observed:

$$x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 1$$

- Given the above:
 - For each possible value of θ , can we compute the probability of the observed sample
 - Can we find the value of θ for which the probability of the observed sample is the largest?

MLE Example

- Since $X_i \sim \text{Bernoulli}\left(\frac{\theta}{3}\right)$, we have:

$$P_{X_i}(x) = \begin{cases} \frac{\theta}{3} & \text{for } x = 1 \\ 1 - \frac{\theta}{3} & \text{for } x = 0 \end{cases}$$

- Since X_i 's are independent, the joint PMF of X_1, X_2, X_3 and X_4 , can be written as:

$$P_{X_1 X_2 X_3 X_4}(x_1, x_2, x_3, x_4) = P_{X_1}(x_1)P_{X_2}(x_2)P_{X_3}(x_3)P_{X_4}(x_4)$$

MLE Example

- Thus:

$$\begin{aligned} P_{X_1 X_2 X_3 X_4}(1, 0, 1, 1) &= \frac{\theta}{3} \cdot \left(1 - \frac{\theta}{3}\right) \cdot \frac{\theta}{3} \cdot \frac{\theta}{3} \\ &= \left(\frac{\theta}{3}\right)^3 \left(1 - \frac{\theta}{3}\right). \end{aligned}$$

θ	$P_{X_1 X_2 X_3 X_4}(1, 0, 1, 1; \theta)$
0	0
1	0.0247
2	0.0988
3	0

Log Likelihood

- Parameters of logistic regression model are chosen using maximum likelihood estimation (MLE) method.
- There are two steps:
 - 1 write the log-likelihood function (define cross entropy function)
 - 2 find the values of w that maximise the log-likelihood function

Logistic Regression Model

- for an individual training instance (\mathbf{x}_i, t_i) :

$$P(t_i = 1 \mid \mathbf{x}_i) = y_i = \sigma(\mathbf{w}^\top \mathbf{x}_i) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x}_i)}$$

$$P(t_i = 0 \mid \mathbf{x}_i) = 1 - y_i$$

- using the equation form of the PMF of Bernoulli distribution, where $t_n \in \{0, 1\}$, the probability of a single instance can be written as:

$$P(t_i = t \mid \mathbf{x}_i) = y_i^{t_i} \{1 - y_i\}^{(1-t_i)}$$

Likelihood

- for a dataset $\{x_n, t_n\}$, where $t_n \in \{0, 1\}$, with $n = 1, \dots, N$, the likelihood function can be written as:

$$L(\mathbf{w}) = P(\mathbf{t} | \mathbf{w}) = \prod_{n=1}^N y_n^{t_n} \{1 - y_n\}^{(1-t_n)}$$

where $\mathbf{t} = (t_1, \dots, t_N)^\top$ and $y_n = p(t_n = 1 | x_n)$

Cross Entropy

- define a negative logarithm of the likelihood to get *cross entropy* error function $E(w)$:

$$L(\mathbf{w}) = \prod_{n=1}^N y_n^{t_n} \{1 - y_n\}^{(1-t_n)}$$

$$LL(\mathbf{w}) = E(\mathbf{w}) = - \sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}$$

Derivation of the Gradient

- differentiating $E(w)$ w.r.t w , we get the gradient $\nabla E(w)$:

$$E(\mathbf{w}) = - \sum_{n=1}^N \{ t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \}$$

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N \left\{ t_n \frac{1}{y_n} \frac{\partial y_n}{\partial w} + (1 - t_n) \frac{1}{1 - y_n} \left(-\frac{\partial y_n}{\partial w} \right) \right\}$$

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N \left\{ \frac{t_n}{y_n} - \frac{1 - t_n}{1 - y_n} \right\} \left(\frac{\partial y_n}{\partial w} \right)$$

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N \left\{ \frac{t_n(1 - y_n) - (1 - t_n)y_n}{y_n(1 - y_n)} \right\} \left(\frac{\partial y_n}{\partial w} \right)$$

Derivation of the Gradient

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N \left\{ \frac{t_n(1 - y_n) - (1 - t_n)y_n}{y_n(1 - y_n)} \right\} \left(\frac{\partial y_n}{\partial w} \right)$$

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N \left\{ \frac{(t_n - y_n)}{y_n(1 - y_n)} \right\} \left(\frac{\partial y_n}{\partial w} \right) \quad (1)$$

Derivation of the Gradient

- differentiating y_n w.r.t w :

$$\frac{\partial y_n}{\partial w} = \frac{\partial}{\partial w} \left\{ \frac{1}{1+\exp^{-wx_n}} \right\}$$

$$\frac{\partial y_n}{\partial w} = \frac{(1+\exp^{-wx_n}) \frac{\partial(1)}{\partial w} - 1 \cdot \frac{\partial}{\partial w}(1+\exp^{-wx_n})}{(1+\exp^{-wx_n})^2}$$

$$\frac{\partial y_n}{\partial w} = \frac{-(\exp^{-wx_n})(-x_n)}{(1+\exp^{-wx_n})^2}$$

$$\frac{\partial y_n}{\partial w} = \frac{1}{1+\exp^{-wx_n}} \frac{\exp^{-wx_n} x_n}{1+\exp^{-wx_n}}$$

Derivation of the Gradient

- differentiating y_n w.r.t \mathbf{w} :

$$\frac{\partial y_n}{\partial w} = \frac{1}{1+\exp^{-wx_n}} \frac{\exp^{-wx_n} x_n}{1+\exp^{-wx_n}}$$

$$\frac{\partial y_n}{\partial w} = \frac{1}{1+\exp^{-wx_n}} \frac{(1+\exp^{-wx_n})-1}{1+\exp^{-wx_n}} x_n$$

$$\frac{\partial y_n}{\partial w} = \frac{1}{1+\exp^{-wx_n}} \left(\frac{1+\exp^{-wx_n}}{1+\exp^{-wx_n}} - \frac{1}{1+\exp^{-wx_n}} \right) x_n$$

$$\frac{\partial y_n}{\partial w} = \frac{1}{1+\exp^{-wx_n}} \left(1 - \frac{1}{1+\exp^{-wx_n}} \right) x_n$$

$$\frac{\partial y_n}{\partial w} = y_n(1 - y_n)x_n \quad (2)$$

Derivation of the Gradient

- Substituting (2) in (1), we get:

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N \left\{ \frac{(t_n - y_n)}{y_n(1 - y_n)} \right\} y_n(1 - y_n) x_n$$

- Simplifying further:

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N (t_n - y_n) x_n$$

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (y_n - t_n) x_n \quad (3)$$

Updating the weight vector

- Generic update rule

$$\mathbf{w}^{(r+1)} = \mathbf{w}^{(r)} - \eta \nabla E(\mathbf{w})$$

- Update rule with cross-entropy error function

$$\mathbf{w}^{(r+1)} = \mathbf{w}^{(r)} - \eta (y_n - t_n) \mathbf{x}_n$$

- the contribution of the gradient from a data point n is given by the error $(y_n - t_n)$ times the feature vector \mathbf{x}_n

Logistic Regression Algorithm

- Given a set of training instances $\{(x_1, t_1), \dots, (x_N, t_N)\}$, learning rate (η) and iterations T :
- Initialise weight vector $w = 0$
- For j in $1, \dots, T$
 - for n in $1, \dots, N$
 - if $\text{pred}(x_i) \neq t_i$ #misclassification
 - $w^{(r+1)} = w^{(r)} - \eta(y_n - t_n)x_n$
- Return the final weight vector w

Prediction Function (*pred*)

- Given the weight vector \mathbf{w} , returns the class label for an instance \mathbf{x}
 - if $\mathbf{w}^T \mathbf{x} > 0$:
 - predicted label = +1 # positive class
 - else:
 - predicted label = 0 # negative class

Online vs. Batch

- Online vs. Batch Logistic Regression
 - The algorithm we discussed in the previous slides is an *online algorithm* because it considers only one instance at a time and updates the weight vector
 - Referred to as the Stochastic Gradient Descent (SGD) update
 - In the batch version, we will compute the cross-entropy error over the *entire* training dataset and then update the weight vector
 - Popular optimisation algorithm for the batch learning of logistic regression is the Limited Memory BFGS (L-BFGS) algorithm
 - Batch version is slow compared to the SGD version. But shows slightly improved accuracies in many cases
 - SGD version can require multiple iterations over the dataset before it converges (if ever)
 - SGD is a technique that is frequently used with large scale machine learning tasks (even when the objective function is non-convex)

Regularisation

- Regularisation
 - Reducing overfitting in a model by constraining it (reducing the complexity/no. of parameters)
 - For classifiers that use a weight vector, regularisation can be done by minimising the norm (length) of the weight vector.
 - Several popular regularisation methods exist
 - L2 regularisation (ridge regression or Tikhonov regularisation)
 - L1 regularisation (Lasso regression)
 - L1+L2 regularisation (mixed regularisation)

L2 Regularisation

- Let us denote the Loss of classifying a dataset D using a model represented by a weight vector \mathbf{w} by $L(D, \mathbf{w})$ and we would like to impose L2 regularisation on \mathbf{w} .
- The overall objective to minimise can then be written as follows (here λ is called the regularisation coefficient and is set via cross-validation)

$$J(D, \mathbf{w}) = L(D, \mathbf{w}) + \lambda \|\mathbf{w}\|_2^2$$

- The gradient of the overall objective simply becomes the addition of the loss-gradient and the scaled weight vector \mathbf{w} .

$$\frac{\partial J(D, \mathbf{w})}{\partial \mathbf{w}} = \frac{\partial L(D, \mathbf{w})}{\partial \mathbf{w}} + 2\lambda \mathbf{w}$$

Examples

- Note that SGD update for minimising a loss multiplies the loss gradient by a negative learning rate (η). Therefore, the L2 regularised update rules will have a $-2\eta\lambda\mathbf{w}$ term as shown in the following examples
- L2 regularised Perceptron update (for a misclassified instance we do)

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + t\mathbf{x} - 2\lambda\mathbf{w}^{(k)}$$

- L2 regularised logistic regression

$$\begin{aligned}\mathbf{w}^{(k+1)} &= \mathbf{w}^{(k)} - \eta((y - t)\mathbf{x} + 2\lambda\mathbf{w}^{(k)}) \\ &= (1 - 2\lambda\eta)\mathbf{w}^{(k)} - \eta(y - t)\mathbf{x}\end{aligned}$$

How to set λ

- Split your training dataset into training and validation parts (eg. 80%-20%)
- Try different values for λ (typically in the logarithmic scale). Train a different classification model for each λ and select the value that gives the best performance (eg. accuracy) on the validation data.
- $\lambda = 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 0, 10^1, 10^2, 10^3, 10^4, 10^5$

References

- Bishop (Pattern Recognition and Machine Learning)
Section 4.3.2
- Software
 - Scikit-learn (Python)
 - https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Logistic Regression



Binary Classification

- Given an instance \mathbf{x} , we must classify it to either positive (1) or negative (0) class.
 - We can use $\{1, -1\}$ instead of $\{1, 0\}$
 - (but we will use 1, 0 as it simplifies the notation in subsequent derivations)
- Binary classification can be seen as learning a function f such that $f(\mathbf{x})$ returns either 1 or 0 indicating the predicted class.

Some terms in Machine Learning

- Training dataset with N instances
 - $\{(x_1, t_1), \dots, (x_N, t_N)\}$ This can also be written as $\{(x_n, t_n)\}_{n=1}^N$
- Target label (class)
 - t : The class labels in the training dataset
 - Annotated by humans (supervised learning)
- Predicted label
 - Labels predicted by our model $f(x)$

From Naive Bayes to Logistic Regression

- Recall Naive Bayes Classifier
 - Predict:

$$\hat{Y} = \operatorname{argmax}_y P(\mathbf{X} | Y) = \operatorname{argmax}_y P(\mathbf{X} | Y)P(Y)$$

- We use independence assumption:

$$P(\mathbf{X} | Y) = P(X_1, X_2, \dots, X_m | Y)P(Y) = \prod_{i=1}^m P(X_i | Y)$$

- Can we model $P(Y | \mathbf{X})$ directly?

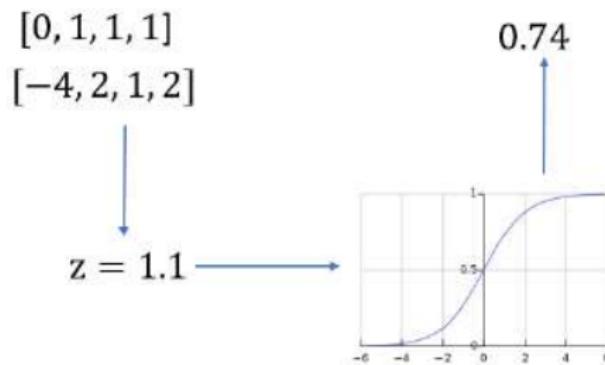
From Naive Bayes to Logistic Regression

- Can we model $P(Y | \mathbf{X})$ directly?
 - We use logistic regression model to achieve this
 - Given X, \mathbf{W} we compute $P(Y = 1 | X)$

$$\begin{matrix} [0, 1, 1, 1] \\ [-4, 2, 1, 2] \end{matrix} \xrightarrow{\hspace{10em}} 0.74$$

From Naive Bayes to Logistic Regression

- Can we model $P(Y | \mathbf{X})$ directly?
 - We use logistic regression model to achieve this
 - Given X, \mathbf{W} we compute $P(Y = 1 | X)$



Generative vs. Discriminative Classifier



Generative

- goal of understanding how dogs look and cats look
- model can 'generate' (draw) a dog
- given a test image, choose the closest model as 'label'

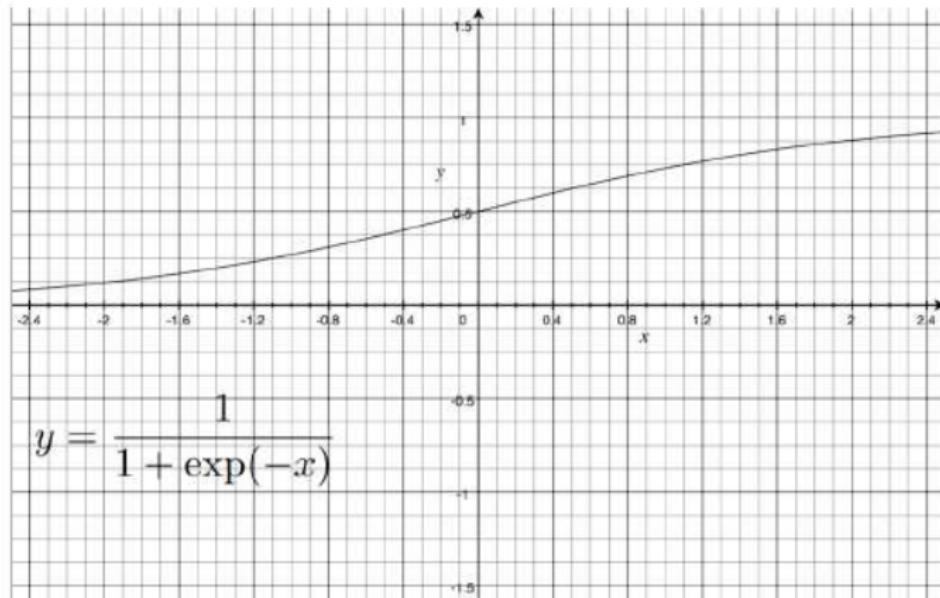
Discriminative

- trying to learn to distinguish the classes (not learning much about them)
- If one feature neatly separates the classes, the model is satisfied.
- e.g., dogs wearing collars and cat aren't

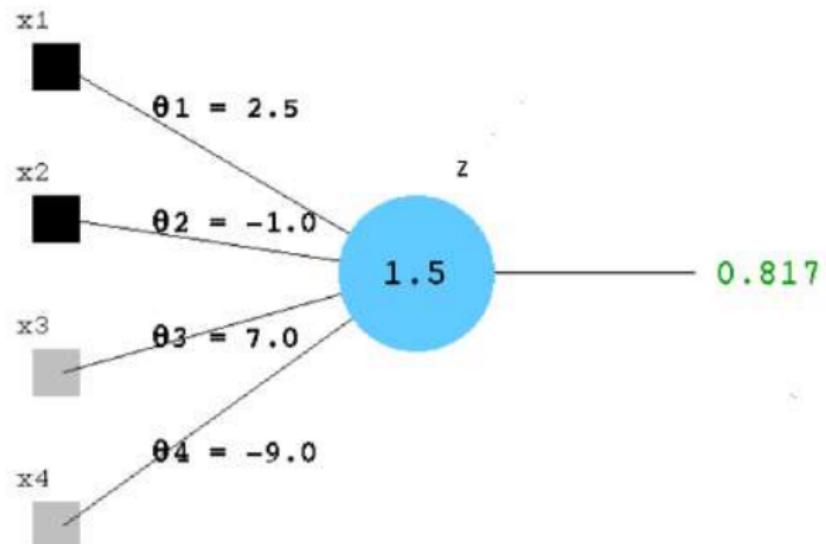
Logistic Regression

- is not a *regression* model
- is a *classification* model
- is a **Discriminative** classifier and not a **Generative** classifier
- is the basis of many advanced machine learning methods
 - neural networks, deep learning, conditional random fields,
...
- try to fit a logistic Sigmoid function to predict the class labels

Logistic Sigmoid Function



Logistic Regression Example



How to find Parameters of the Model

- In logistic regression, given \mathbf{x}, \mathbf{w} we compute $P(Y = y | \mathbf{x})$
 - How to compute \mathbf{w} ?
-
- Maximum Likelihood Estimate/Principle (MLE): parameter estimation method to find parameters of the model
 - The parameter values are found such that the values maximise the *likelihood* of making the observations given the parameters.

MLE Example

- Consider a bag containing 3 balls.
- Each ball is either red or blue (we have no other information)
- number of blue balls (θ) might be 0, 1, 2, or 3
- we are allowed to choose 4 balls at random with replacement.
- the random variables X_1, X_2, X_3 and X_4 are defined as follows:

$$X_i = \begin{cases} 1 & \text{if the } i\text{th chosen ball is blue} \\ 0 & \text{if the } i\text{th chosen ball is red} \end{cases}$$

MLE Example

- X_i 's are i.i.d and $X_i \sim Bernoulli\left(\frac{\theta}{3}\right)$.
- let us say the following values for X_i 's are observed:

$$x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 1$$

- Given the above:
 - For each possible value of θ , can we compute the probability of the observed sample
 - Can we find the value of θ for which the probability of the observed sample is the largest?

MLE Example

- Since $X_i \sim \text{Bernoulli}\left(\frac{\theta}{3}\right)$, we have:

$$P_{X_i}(x) = \begin{cases} \frac{\theta}{3} & \text{for } x = 1 \\ 1 - \frac{\theta}{3} & \text{for } x = 0 \end{cases}$$

- Since X_i 's are independent, the joint PMF of X_1, X_2, X_3 and X_4 , can be written as:

$$P_{X_1 X_2 X_3 X_4}(x_1, x_2, x_3, x_4) = P_{X_1}(x_1)P_{X_2}(x_2)P_{X_3}(x_3)P_{X_4}(x_4)$$

MLE Example

- Thus:

$$\begin{aligned} P_{X_1 X_2 X_3 X_4}(1, 0, 1, 1) &= \frac{\theta}{3} \cdot \left(1 - \frac{\theta}{3}\right) \cdot \frac{\theta}{3} \cdot \frac{\theta}{3} \\ &= \left(\frac{\theta}{3}\right)^3 \left(1 - \frac{\theta}{3}\right). \end{aligned}$$

θ	$P_{X_1 X_2 X_3 X_4}(1, 0, 1, 1; \theta)$
0	0
1	0.0247
2	0.0988
3	0

Log Likelihood

- Parameters of logistic regression model are chosen using maximum likelihood estimation (MLE) method.
- There are two steps:
 - 1 write the log-likelihood function (define cross entropy function)
 - 2 find the values of w that maximise the log-likelihood function

Logistic Regression Model

- for an individual training instance (\mathbf{x}_i, t_i) :

$$P(t_i = 1 \mid \mathbf{x}_i) = y_i = \sigma(\mathbf{w}^\top \mathbf{x}_i) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x}_i)}$$

$$P(t_i = 0 \mid \mathbf{x}_i) = 1 - y_i$$

- using the equation form of the PMF of Bernoulli distribution, where $t_n \in \{0, 1\}$, the probability of a single instance can be written as:

$$P(t_i = t \mid \mathbf{x}_i) = y_i^{t_i} \{1 - y_i\}^{(1-t_i)}$$

Likelihood

- for a dataset $\{x_n, t_n\}$, where $t_n \in \{0, 1\}$, with $n = 1, \dots, N$, the likelihood function can be written as:

$$L(\mathbf{w}) = P(\mathbf{t} | \mathbf{w}) = \prod_{n=1}^N y_n^{t_n} \{1 - y_n\}^{(1-t_n)}$$

where $\mathbf{t} = (t_1, \dots, t_N)^\top$ and $y_n = p(t_n = 1 | x_n)$

Cross Entropy

- define a negative logarithm of the likelihood to get *cross entropy* error function $E(w)$:

$$L(\mathbf{w}) = \prod_{n=1}^N y_n^{t_n} \{1 - y_n\}^{(1-t_n)}$$

$$LL(\mathbf{w}) = E(\mathbf{w}) = - \sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}$$

Derivation of the Gradient

- differentiating $E(w)$ w.r.t w , we get the gradient $\nabla E(w)$:

$$E(\mathbf{w}) = - \sum_{n=1}^N \{ t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \}$$

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N \left\{ t_n \frac{1}{y_n} \frac{\partial y_n}{\partial w} + (1 - t_n) \frac{1}{1 - y_n} \left(-\frac{\partial y_n}{\partial w} \right) \right\}$$

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N \left\{ \frac{t_n}{y_n} - \frac{1 - t_n}{1 - y_n} \right\} \left(\frac{\partial y_n}{\partial w} \right)$$

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N \left\{ \frac{t_n(1 - y_n) - (1 - t_n)y_n}{y_n(1 - y_n)} \right\} \left(\frac{\partial y_n}{\partial w} \right)$$

Derivation of the Gradient

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N \left\{ \frac{t_n(1-y_n) - (1-t_n)y_n}{y_n(1-y_n)} \right\} \left(\frac{\partial y_n}{\partial w} \right)$$

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N \left\{ \frac{(t_n - y_n)}{y_n(1-y_n)} \right\} \left(\frac{\partial y_n}{\partial w} \right) \quad (1)$$

Derivation of the Gradient

- differentiating y_n w.r.t w :

$$\frac{\partial y_n}{\partial w} = \frac{\partial}{\partial w} \left\{ \frac{1}{1+\exp^{-wx_n}} \right\}$$

$$\frac{\partial y_n}{\partial w} = \frac{(1+\exp^{-wx_n}) \frac{\partial(1)}{\partial w} - 1 \cdot \frac{\partial}{\partial w}(1+\exp^{-wx_n})}{(1+\exp^{-wx_n})^2}$$

$$\frac{\partial y_n}{\partial w} = \frac{-(\exp^{-wx_n})(-x_n)}{(1+\exp^{-wx_n})^2}$$

$$\frac{\partial y_n}{\partial w} = \frac{1}{1+\exp^{-wx_n}} \frac{\exp^{-wx_n} x_n}{1+\exp^{-wx_n}}$$

Derivation of the Gradient

- differentiating y_n w.r.t \mathbf{w} :

$$\frac{\partial y_n}{\partial w} = \frac{1}{1+\exp^{-wx_n}} \frac{\exp^{-wx_n} x_n}{1+\exp^{-wx_n}}$$

$$\frac{\partial y_n}{\partial w} = \frac{1}{1+\exp^{-wx_n}} \frac{(1+\exp^{-wx_n})-1}{1+\exp^{-wx_n}} x_n$$

$$\frac{\partial y_n}{\partial w} = \frac{1}{1+\exp^{-wx_n}} \left(\frac{1+\exp^{-wx_n}}{1+\exp^{-wx_n}} - \frac{1}{1+\exp^{-wx_n}} \right) x_n$$

$$\frac{\partial y_n}{\partial w} = \frac{1}{1+\exp^{-wx_n}} \left(1 - \frac{1}{1+\exp^{-wx_n}} \right) x_n$$

$$\frac{\partial y_n}{\partial w} = y_n(1 - y_n)x_n \quad (2)$$

Derivation of the Gradient

- Substituting (2) in (1), we get:

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N \left\{ \frac{(t_n - y_n)}{y_n(1 - y_n)} \right\} y_n(1 - y_n) x_n$$

- Simplifying further:

$$\nabla E(\mathbf{w}) = - \sum_{n=1}^N (t_n - y_n) x_n$$

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (y_n - t_n) x_n \quad (3)$$

Updating the weight vector

- Generic update rule

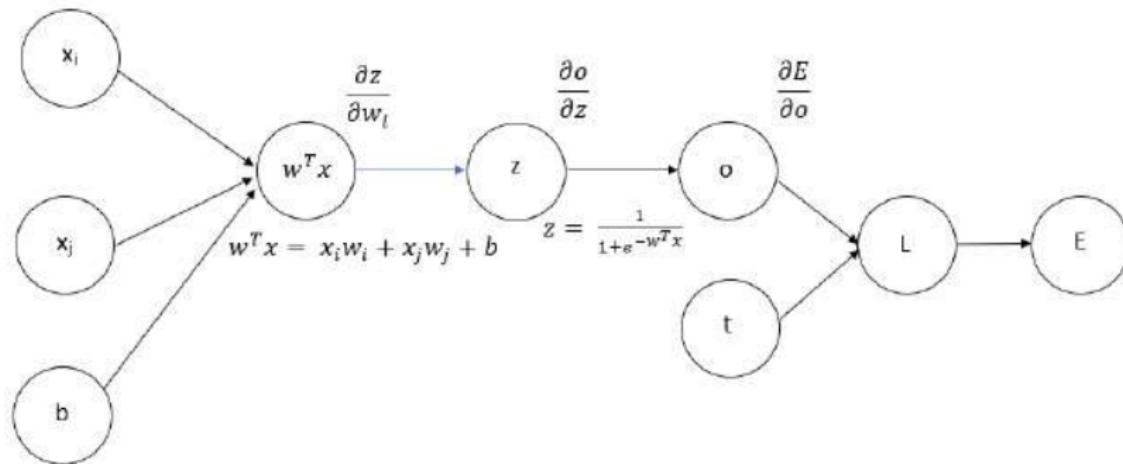
$$\mathbf{w}^{(r+1)} = \mathbf{w}^{(r)} - \eta \nabla E(\mathbf{w})$$

- Update rule with cross-entropy error function

$$\mathbf{w}^{(r+1)} = \mathbf{w}^{(r)} - \eta (y_n - t_n) \mathbf{x}_n$$

- the contribution of the gradient from a data point n is given by the error $(y_n - t_n)$ times the feature vector \mathbf{x}_n

Updating Weight Vector



/

Logistic Regression Algorithm

- Given a set of training instances $\{(x_1, t_1), \dots, (x_N, t_N)\}$, learning rate (η) and iterations T :
- Initialise weight vector $w = 0$
- For j in $1, \dots, T$
 - for n in $1, \dots, N$
 - if $\text{pred}(x_i) \neq t_i$ #misclassification
 - $w^{(r+1)} = w^{(r)} - \eta(y_n - t_n)x_n$
- Return the final weight vector w

Prediction Function (*pred*)

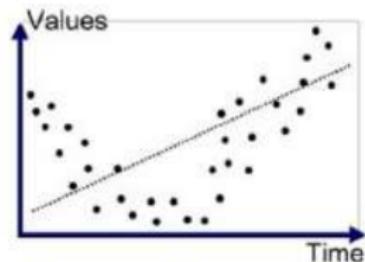
- Given the weight vector \mathbf{w} , returns the class label for an instance \mathbf{x}
 - if $\mathbf{w}^T \mathbf{x} > 0$:
 - predicted label = +1 # positive class
 - else:
 - predicted label = 0 # negative class

Online vs. Batch

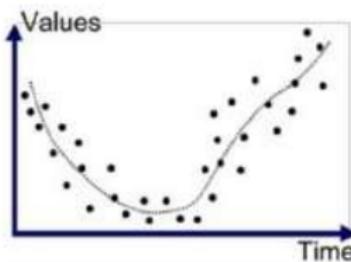
- Online vs. Batch Logistic Regression
 - The algorithm we discussed in the previous slides is an *online algorithm* because it considers only one instance at a time and updates the weight vector
 - Referred to as the Stochastic Gradient Descent (SGD) update
 - In the batch version, we will compute the cross-entropy error over the *entire* training dataset and then update the weight vector
 - Popular optimisation algorithm for the batch learning of logistic regression is the Limited Memory BFGS (L-BFGS) algorithm
- Batch version is slow compared to the SGD version. But shows slightly improved accuracies in many cases
- SGD version can require multiple iterations over the dataset before it converges (if ever)
- SGD is a technique that is frequently used with large scale machine learning tasks (even when the objective function is non-convex)

Regularisation

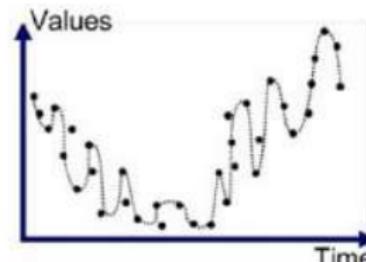
Overfitting



Underfitted



Good Fit/Robust



Overfitted

Regularisation

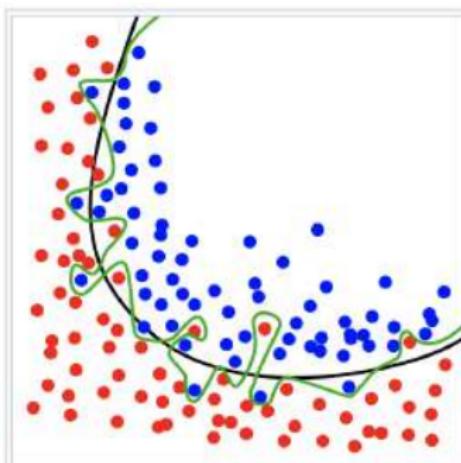


Figure 1. The green line represents an overfitted model and the black line represents a regularized model. While the green line best follows the training data, it is too dependent on that data and it is likely to have a higher error rate on new unseen data, compared to the black line.

How to reduce overfitting?

Without adding any regularisation

- Train on more data
- Data augmentation
- Early stopping

stop training when validation error starts increasing or validation error stops improving

Regularisation

- Regularisation
 - Reducing overfitting in a model by constraining it (reducing the complexity/no. of parameters)
 - For classifiers that use a weight vector, regularisation can be done by minimising the norm (length) of the weight vector.
 - Several popular regularisation methods exist
 - L2 regularisation (ridge regression or Tikhonov regularisation)
 - L1 regularisation (Lasso regression)
 - L1+L2 regularisation (mixed regularisation)

L2 Regularisation

- Let us denote the Loss of classifying a dataset D using a model represented by a weight vector \mathbf{w} by $L(D, \mathbf{w})$ and we would like to impose L2 regularisation on \mathbf{w} .
- The overall objective to minimise can then be written as follows (here λ is called the regularisation coefficient and is set via cross-validation)

$$J(D, \mathbf{w}) = L(D, \mathbf{w}) + \lambda \|\mathbf{w}\|_2^2$$

- The gradient of the overall objective simply becomes the addition of the loss-gradient and the scaled weight vector \mathbf{w} .

$$\frac{\partial J(D, \mathbf{w})}{\partial \mathbf{w}} = \frac{\partial L(D, \mathbf{w})}{\partial \mathbf{w}} + 2\lambda \mathbf{w}$$

Examples

- Note that SGD update for minimising a loss multiplies the loss gradient by a negative learning rate (η). Therefore, the L2 regularised update rules will have a $-2\eta\lambda\mathbf{w}$ term as shown in the following examples
- L2 regularised Perceptron update (for a misclassified instance we do)

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + t\mathbf{x} - 2\lambda\mathbf{w}^{(k)}$$

- L2 regularised logistic regression

$$\begin{aligned}\mathbf{w}^{(k+1)} &= \mathbf{w}^{(k)} - \eta((y - t)\mathbf{x} + 2\lambda\mathbf{w}^{(k)}) \\ &= (1 - 2\lambda\eta)\mathbf{w}^{(k)} - \eta(y - t)\mathbf{x}\end{aligned}$$

How to set λ

- Split your training dataset into training and validation parts (eg. 80%-20%)
- Try different values for λ (typically in the logarithmic scale). Train a different classification model for each λ and select the value that gives the best performance (eg. accuracy) on the validation data.
- $\lambda = 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 0, 10^1, 10^2, 10^3, 10^4, 10^5$

References

- Bishop (Pattern Recognition and Machine Learning)
Section 4.3.2
- Software
 - Scikit-learn (Python)
 - https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Naive Bayes Classifier

Danushka Bollegala



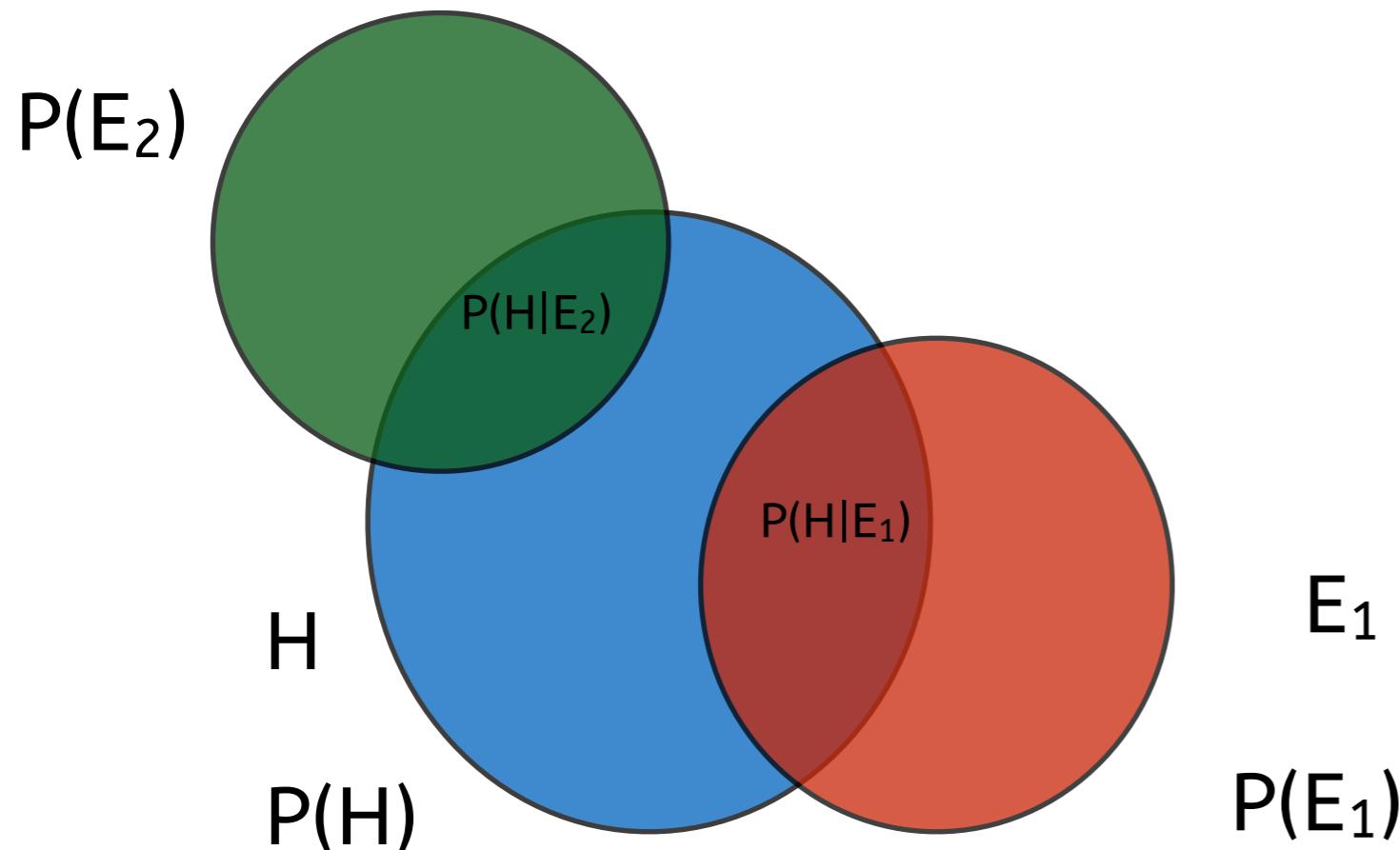
Bayes Rule

- The probability of hypothesis H, given evidence E
 - $P(H | E) = P(E | H)P(H)/P(E)$
- Terminology
 - $P(E)$: Marginal probability of the evidence E
 - $P(H)$: Prior probability of hypothesis H
 - $P(E | H)$: Likelihood of the evidence given hypothesis
 - $P(H | E)$: Posterior probability of the hypothesis H

Example

- Meningitis causes a stiff neck 50% of the time. Meningitis occurs 1/50000 and stiff neck occurs 1/20. Compute the probability of meningitis, given that the patient has a stiff neck?
- $H = \text{meningitis}$, $E = \text{stiff neck}$
- $P(H) = 1/50000$, $P(E) = 1/20$, $P(E|H) = 0.5$
- From Bayes' rule we have
 - $P(H|E) = P(E|H)P(H)/P(E) = 0.0002$

Intuition/Derivation



H can be related to several evidences E_1, E_2, \dots, E_n

$$P(H) = P(E_1)P(H|E_1) + P(E_2)P(H|E_2) + \dots + P(E_n)P(H|E_n)$$

By the definition of conditional probability we have

$$P(H|E) = P(H,E)/P(E)$$

$$P(E|H) = P(H,E)/P(H)$$

Dividing one from the other we get

$$P(H|E) = P(E|H)P(H)/P(E)$$

Bayes Rule — Proportional Form

- Often the evidence is given ($P(E)$ is fixed) and we need to select from a set of hypothesis h_1, h_2, \dots, h_k
- In such cases we can simplify the formula to
 - $P(H | E) \propto P(E | H)P(H)$
 - posterior \propto likelihood \times prior
- At least remember this form!

Naive Bayes

- Let us assume a particular hypothesis H depends on several evidences E_1, E_2, \dots, E_n
- From Bayes rule we have
 - $P(H | E_1, E_2, \dots, E_n) \propto P(E_1, E_2, \dots, E_n | H)P(H)$
- Let us further assume that given the hypothesis H , the evidences are **mutually exclusive**
 - Then we can decompose the likelihood term
 - $P(H | E_1, E_2, \dots, E_n) \propto P(E_1 | H) \dots P(E_n | H)P(H)$
- This independence assumption is what make naive bayes so naive!

Independent Events

- Joint probability of independent events
 - $P(A,B) = P(A | B)P(B)$ This holds for ANY two random events A and B, irrespective of whether they are independent or not.
 - But if A is independent of B, then B's occurrence has no consequence on A
 - $P(A | B) = P(A)$
- Therefore, when A and B are independent
 - $P(A,B) = P(A)P(B)$

Being naive makes life easy

- Let $H = \text{engine-does-not-start}$, and evidences $A = \text{weak-battery}$ and $B = \text{no-gas}$
- $P(H | A, B) = P(A, B | H)P(H)/P(A, B)$
- We must estimate the likelihood $P(A, B | H)$
- If A and B are mutually independent given H
 - $P(A, B | H) = P(A | H)P(B | H)$
 - $P(A | H)$ can be estimated by finding how many cars had engine not working because of a weak battery
 - $P(B | H)$ can be estimated by finding how many cars had engines not working because of no gas
 - On the other hand, if we tried to estimate $P(A, B | H)$ directly then we need to find how many cars had engines not working due to a weak battery and no gas. Such cases could be rare making our estimate of $P(A, B | H)$ unreliable or zero (in the worst case).
- Making the independence assumption makes estimates possible in practice.

Predicting whether play=yes

Outlook		Temperature		Humidity		Windy		Play		
	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No
Sunny	2	3	Hot	2	2	High	3	4	False	6 2
Overcast	4	0	Mild	4	2	Normal	6	1	True	3 3
Rainy	3	2	Cool	3	1					
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9 2/5
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9 3/5
Rainy	3/9	2/5	Cool	3/9	1/5					

To play or not to play

Given a test instance

$$x = (\text{outlook}=\text{sunny}, \text{temp}=\text{cool}, \text{humidity}=\text{high}, \text{windy}=\text{TRUE})$$

$$\begin{aligned} P(\text{play}=\text{yes}|x) &\propto P(x|\text{play}=\text{yes})P(\text{play}=\text{yes}) \\ &= P(\text{outlook}=\text{sunny}|\text{play}=\text{yes}) \times P(\text{temp}=\text{cool}|\text{play}=\text{yes}) \times \\ &\quad P(\text{humidity}=\text{high}|\text{play}=\text{yes}) \times P(\text{windy}=\text{TRUE}|\text{play}=\text{yes}) \times P(\text{play}=\text{yes}) \\ &= 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.00529 \end{aligned}$$

$$\begin{aligned} P(\text{play}=\text{no}|x) &\propto P(x|\text{play}=\text{no})P(\text{play}=\text{no}) \\ &= P(\text{outlook}=\text{sunny}|\text{play}=\text{no}) \times P(\text{temp}=\text{cool}|\text{play}=\text{no}) \times \\ &\quad P(\text{humidity}=\text{high}|\text{play}=\text{no}) \times P(\text{windy}=\text{TRUE}|\text{play}=\text{no}) \times P(\text{play}=\text{no}) \\ &= 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0020 \end{aligned}$$

Therefore play=yes.

Computing probabilities

- Note that
 - $P(\text{play=yes} | x) \approx 0.0052$
 - $P(\text{play=no} | x) \approx 0.0020$
- How can we compute the actual probabilities?
- Note that
 - $P(\text{play=yes} | x) + P(\text{play=no} | x) = 1$
- Therefore,
 - $P(\text{play=yes} | x) = 0.0052 / (0.0052 + 0.0020) = 0.72$

Sometimes it is too naive...

- Naive Bayes' assumption that the features are independent given the hypothesis is sometimes too naive to be true.
- The probability of Liverpool winning a football match is not independent of the probability for each member of the team scoring a goal.
- However, as we saw in a previous slide, it gives us a method to estimate the joint distribution of a set of random variables without getting into data sparseness issues.
- The linear classifiers we studied in the module so far such as the perceptron are also making such assumptions about the feature independence (the activation score is a linearly weighted sum after all)
- $\log(P(A,B|H)) = \log(P(A|H) \times P(B|H)) = \log(P(A|H)) + \log(P(B|H))$

Zero probabilities

- Issue: If a feature value does not co-occur with a class value, then the probability generated for it will be 0.
- Eg. Given outlook=overcast, the probability of play=no is 0/5. The other features will be ignored as the final result will be multiplied by 0.
- This is bad for our 4 feature dataset, but terrible for (say) a 1000 feature dataset.
- In text classification, we often encounter situations where a feature does not occur in a particular class.

Laplace Smoothing

- We can “borrow” some probabilities from high probability features and distribute them among zero probability features to avoid having feature with zero probabilities
- This is called *smoothing*
- There are numerous smoothing techniques based on different policies. As long as the total probability mass remains unchanged any policy of probability reassignment is valid.
- A popular method is called **Laplace smoothing**
 - Add 1 to all counts to avoid zeros!
 - $P(w) = (\text{count}(w) + 1) / (N + |\mathcal{V}|)$
 - $\text{count}(w)$: the actual count (before smoothing) of a word w
 - N : corpus size in words (total number of counts of all words) $\sum_{w \in \mathcal{V}} \text{count}(w)$
 - $|\mathcal{V}|$: vocabulary size (how many different words do we have)

Quiz

- Let $\text{count(cat)}=9$, $\text{count(dog)}=0$, and $\text{count(rabbit)} = 1$. Compute the probabilities $P(\text{cat})$, $P(\text{dog})$, and $P(\text{rabbit})$.
- Now smooth the above probabilities using Laplace smoothing.

Document Classification

- A document can be represented using a bag of words (features such as unigrams and bigrams). We could represent a document by a vector where each element corresponds to the total frequency of a feature in the document.
- $D = \text{"the burger i ate was an awesome burger"}$
- $v(D) = \{\text{the:1}, \text{burger:2}, \text{i:1}, \text{ate:1}, \text{was:1}, \text{an:1}, \text{awesome:1}\}$
- Assuming the features to be independent (the *naive* assumption) we can compute the likelihood (probability) of this document D , $p(D)$ as follows
- $p(D) = p(\text{the})^1 p(\text{burger})^2 p(\text{ate})^1 p(\text{was})^1 p(\text{an})^1 p(\text{awesome})^1$
- If a word w occurs n times in D , then the term corresponding to $p(w)$ appears n times in the product. Therefore, we have $p(w)^n$ in the likelihood computation above.

Document Classification

- The Bayesian model is often used to classify documents as it deals well with a huge number of features simultaneously.
- But we might know how many times a word occurs in the document (vector representation in the previous slide)
- This leads to Multinomial Naive Bayes
- Assumptions:
 - Probability of a word occurring in a document is independent of its location within the document.
 - The document length is not related to the class.

Multinomial Distribution

- This is an extension of the Binomial distribution for more than two classes
- Binomial distribution
 - What is the probability that when I flip a coin n times I will get k number of heads (H) and (n-k) number of tails (T)?

$$p(H = k, T = n - k) = \frac{n!}{k!(n - k)!} p^k (1 - p)^{(n - k)}$$

- Multinomial distribution
 - Lets flip a dice instead of a coin. There are six outcomes.

$$p(1 = a, 2 = b, \dots, 6 = f) = \frac{n!}{a!b!\dots f!} p(1)^a p(2)^b \dots p(6)^f$$

$$n = a + b + c + d + e + f$$

Document Classification

$$P(\mathbf{x}|y) = N! \prod_{w \in D} \frac{p(w|y)^{h(w,D)}}{h(w,D)!}$$

- N = number of words in the document \mathbf{x}
- $p(w|y)$ = probability that the word w occurs in class y
- $h(w,D)$ = total occurrences of the word w in document D

Classifying “burger” sentiment

- Let us assume that we would like to classify whether the following document is positive (1) or negative (-1) in sentiment.
- $D = \text{"the burger i ate was an awesome burger"}$
- Further assume we see these words in positive and negative classes as follows. To make our computations easier let us assume that we removed “the”, “i”, “was”, “an” as stop words.

word	+1	-1
burger	3	2
ate	3	2
awesome	4	1

Computing class conditional probabilities

word	+1	-1	$p(w +1)$	$p(w -1)$
burger	3	2	3/10	2/5
ate	3	2	3/10	2/5
awesome	4	1	4/10	1/5

Quiz

- Using the probabilities in the table in slide 21 and multinomial naive Bayes formula in slide 19, compute $P(D|+1)$ and $P(D|-1)$

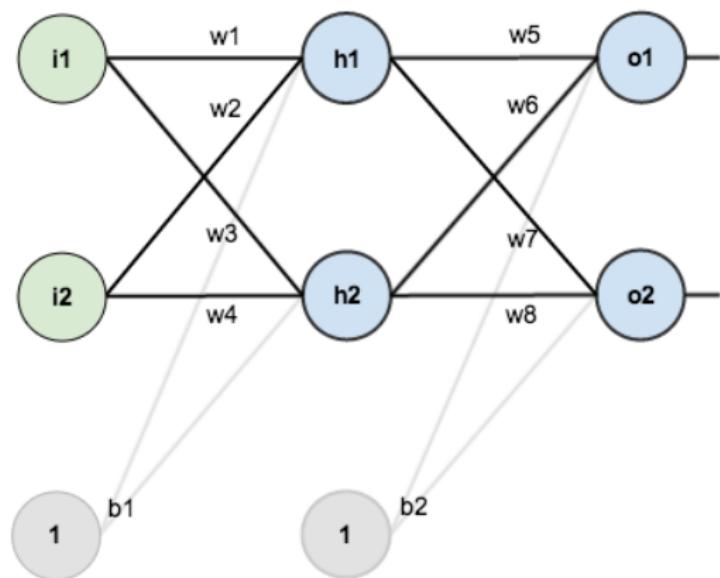
Quiz

- Assuming that both positive and negative classes have equal probability
 - $p(+1) = p(-1) = 0.5$
- Compute $P(+1 | D)$ and $P(-1 | D)$ for the document in slide 22.
- Is this document D positive or negative?

Backpropagation: Numerical Example

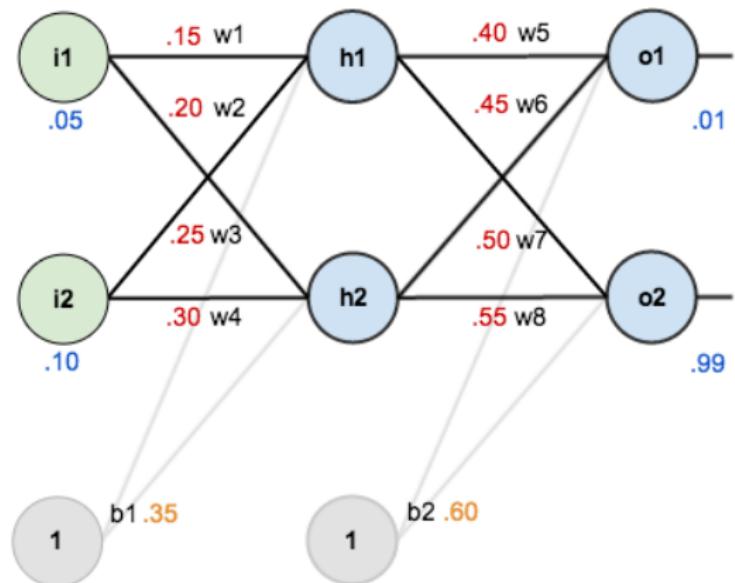


Backpropagation: Numerical example



source: <https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/>

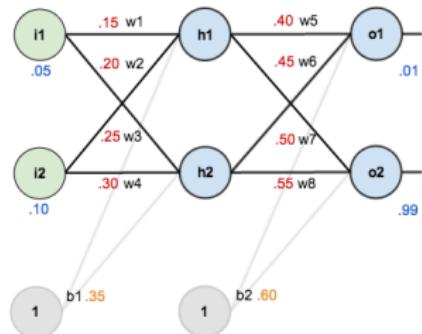
Backpropagation: Numerical example



source: <https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/>

Backpropagation: Numerical example

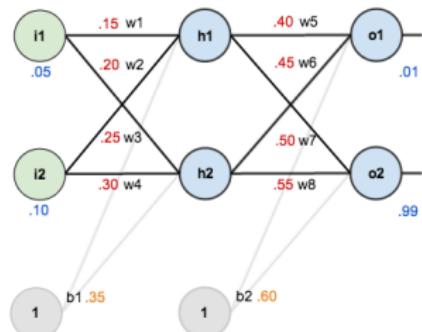
- two inputs
- two hidden neurons
- two output neurons
- parameters of the model:
 - Weight matrices W_1 , W_2 ,
 - bias b_1 , b_2



Backpropagation: Forward Pass

hidden layer neurons

- at $\mathbf{h} \in \mathbb{R}^2$, we compute:
 - $net_h = W_1^T \mathbf{x}$
 - $out_h = \sigma(W_1^T \mathbf{x})$ (activation)
- $net_{h1} = w_1 \times i_1 + w_2 \times i_2 + b_1 \times 1$
- $net_{h1} = 0.15 \times 0.005 + 0.2 \times 0.1 + 0.35 \times 1 = 0.3775$
- $out_{h1} = \sigma(net_{h1}) = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.59326992$
- Similarly we get $out_{h2} = 0.596884378$



Backpropagation: Forward Pass

output layer neurons

- at $\mathbf{o} \in \mathbb{R}^2$, we compute:

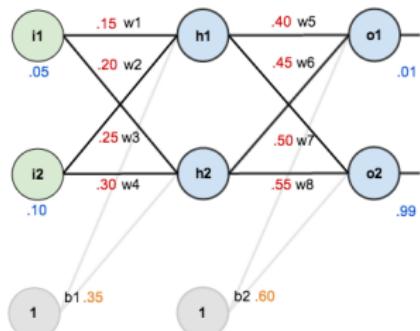
- $net_o = W_2^T \mathbf{h}$
- $out_o = \sigma(W_2^T \mathbf{h})$

- $net_{o1} = w_5 \times out_{h1} + w_6 \times out_{h2} + b_2 \times 1$

- $out_{o1} = 0.4 \times 0.5932 + 0.45 \times 0.5968 + 0.6 \times 1 = 1.1059$

- $out_{o1} = \sigma(net_{o1}) = \frac{1}{1+e^{-net_{o1}}} = \frac{1}{1+e^{-1.1059}} = 0.7513$

- Similarly we get $out_{o2} = 0.7729$



Backpropagation: Total Error

- $E_{total} = \sum \frac{1}{2}(target - output)^2$

- error at $o1$:

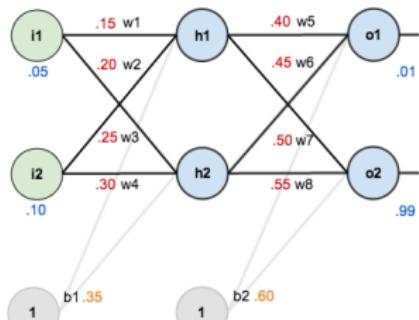
$$E_{o1} = \frac{1}{2}(target_{o1} - output_{o1})^2$$

$$E_{o1} = \frac{1}{2}(0.01 - 0.7513)^2 = 0.2748$$

- similarly $E_{o2} = 0.02356$

- Total Error: $E_{total} = E_{o1} + E_{o2}$

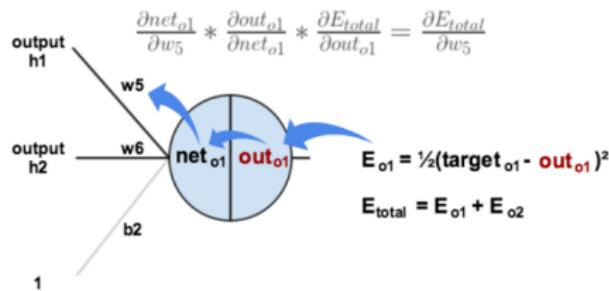
$$E_{total} = 0.2748 + 0.0235 = 0.2983$$



Backpropagation: Backwards Pass

- update weights so that they cause actual output to be closer to target output
- specifically, we will consider $\frac{\partial E_{total}}{\partial w_5}$ (change in w_5 that affects total error):

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial w_5}$$



1

Backpropagation: Backwards Pass

1. compute: $\frac{\partial E_{total}}{\partial out_{o1}}$

- $E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 \times \frac{1}{2}(target_{o1} - out_{o1})^{2-1} \times (-1) + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1})$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(0.01 - 0.7513) = 0.7413$$

Backpropagation: Backwards Pass

2. compute: $\frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}}$

$$\text{out}_{o1} = \sigma(\text{net}_{o1}) = \frac{1}{1+e^{-\text{net}_{o1}}}$$

$$\frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} = \text{out}_{o1}(1 - \text{out}_{o1}) \text{ (see logistic regression lecture notes)}$$

$$\frac{\partial \text{out}_{o1}}{\partial \text{net}_{o1}} = 0.7513(1 - 0.7513) = 0.1868$$

3. compute: $\frac{\partial \text{net}_{o1}}{\partial w_5}$

$$\text{net}_{o1} = w_5 \times \text{out}_{h1} + w_6 \times \text{out}_{h2} + b_2 \times 1$$

$$\frac{\partial \text{net}_{o1}}{\partial w_5} = \text{out}_{h1} = 0.5932$$

Backpropagation: Backwards Pass

Putting it together:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.7413 \times 0.1868 \times 0.5932 = 0.0821$$

- To decrease the error, we subtract $\frac{\partial E_{total}}{\partial w_5}$ from current weight:

$$w_5^+ = w_5 - \eta \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 \times 0.0821 = 0.3589$$

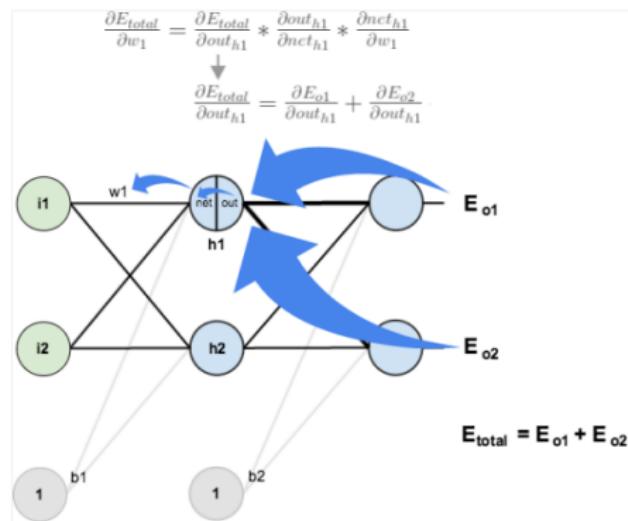
Similarly:

$$w_6^+ = 0.4086; w_7^+ = 0.51130; w_8^+ = 0.5613$$

Backpropagation: Backwards Pass

Hidden Layer:

- compute w_1, w_2, w_3, w_4 : $\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} \times \frac{\partial out_{h1}}{\partial net_{h1}} \times \frac{\partial net_{h1}}{\partial w_1}$



Backpropagation: Backwards Pass

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial out_{h1}}$$

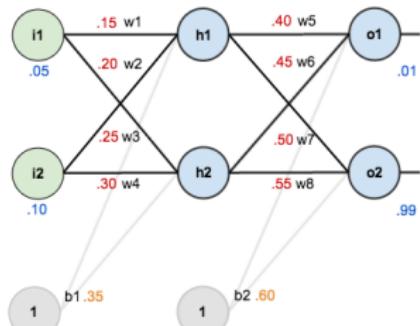
1. compute $\frac{\partial E_{o1}}{\partial out_{o1}}$

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2$$

$$\frac{\partial E_{o1}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.7513) = 0.7413$$

2. compute $\frac{\partial out_{o1}}{\partial net_{o1}}$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.7513(1 - 0.7513) = 0.1868$$



Backpropagation: Backwards Pass

3. compute: $\frac{\partial net_{o1}}{\partial out_{h1}}$

$$net_{o1} = w_5 \times out_{h1} + w_6 \times out_{h2} + b_2 \times 1$$

$$\frac{\partial net_{o1}}{\partial out_{h1}} = w_5 = 0.40$$

Substituting for $\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial out_{h1}}$, we get

$$\frac{\partial E_{o1}}{\partial out_{h1}} = 0.7413 \times 0.1868 \times 0.40 = 0.0553$$

Following the same process: $\frac{\partial E_{o2}}{\partial out_{h1}} = -0.0190$

Thus, $\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}} = 0.0553 + (-0.0190) = 0.0363$

Backpropagation: Backwards Pass

- We had: $\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} \times \frac{\partial out_{h1}}{\partial net_{h1}} \times \frac{\partial net_{h1}}{\partial w_1}$
- We computed $\frac{\partial E_{total}}{\partial out_{h1}}$, we need to compute: $\frac{\partial out_{h1}}{\partial net_{h1}}; \frac{\partial net_{h1}}{\partial w_1}$

$$out_{h1} = \sigma(net_{h1}) = \frac{1}{1+e^{-net_{h1}}}$$

$$\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1})$$

$$\frac{\partial out_{h1}}{\partial net_{h1}} = 0.5932(1 - 0.5932) = 0.2413$$

Backpropagation: Backwards Pass

- We had: $\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} \times \frac{\partial out_{h1}}{\partial net_{h1}} \times \frac{\partial net_{h1}}{\partial w_1}$
- We computed $\frac{\partial E_{total}}{\partial out_{h1}}$; $\frac{\partial out_{h1}}{\partial net_{h1}}$, we need to compute: $\frac{\partial net_{h1}}{\partial w_1}$

$$net_{h1} = w_1 \times i_1 + w_2 \times i_2 + b_1 \times 1$$

$$\frac{\partial net_{h1}}{\partial w_1} = i_1 = 0.05$$

- Putting it together:

$$\frac{\partial E_{total}}{\partial w_1} = 0.0363 \times 0.2413 \times 0.05 = 0.00043$$

- update w_1 :

$$w_1^+ = w_1 - \eta \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 \times 0.000438 = 0.1497$$

Similarly: $w_2^+ = 0.1995$; $w_3^+ = 0.2497$; $w_4^+ = 0.2995$

Backpropagation: Backwards Pass

- with inputs 0.05 and 0.1 - initial error 0.298371109
- after first round of backpropagation - error: 0.291027924
- after repeating the process 10,000 times - error: 0.0000351085
- the output neurons generates:
 - 0.015912196 (vs 0.01 target)
 - 0.984065734 (vs 0.99 target)

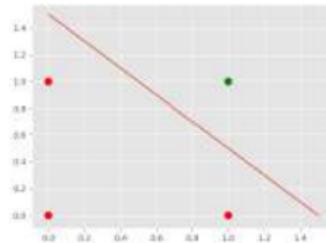
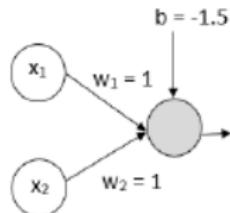
Neural Networks



Perceptron: Linearly separable data

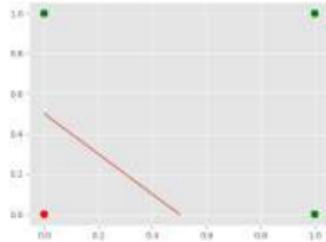
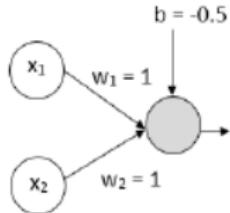
Boolean AND

Input x_1	Input x_2	Output
0	0	0
0	1	0
1	0	0
1	1	1



Boolean OR

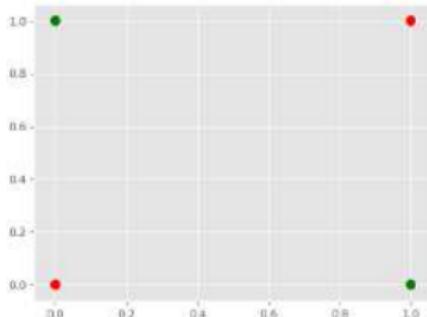
Input x_1	Input x_2	Output
0	0	0
0	1	1
1	0	1
1	1	1



Perceptron: Linearly inseparable data

Boolean XOR

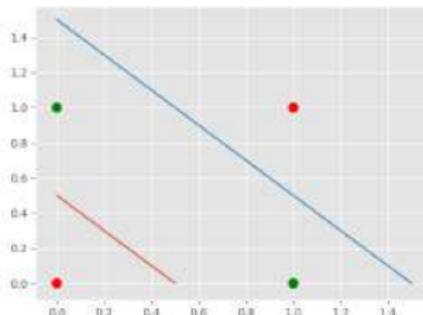
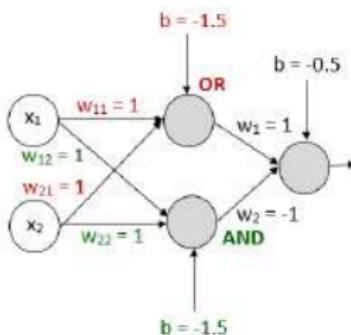
Input x_1	Input x_2	Output
0	0	0
0	1	1
1	0	1
1	1	0



Perceptron: Linearly inseparable data

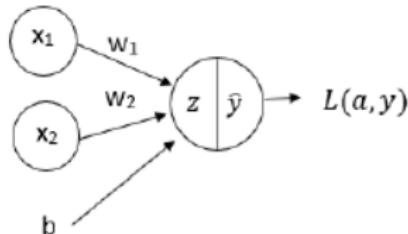
Boolean XOR

Input x_1	Input x_2	Output
0	0	0
0	1	1
1	0	1
1	1	0



Logistic Regression to Neural Network

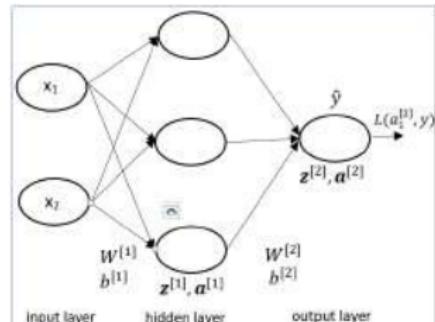
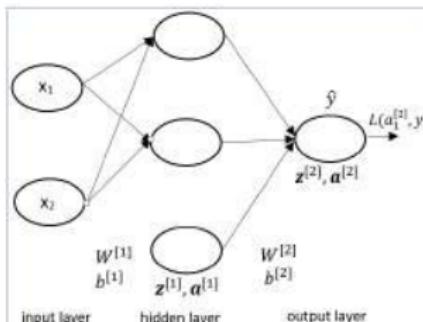
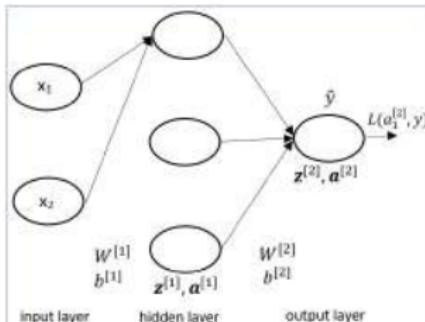
Logistic regression (sigmoid neuron)



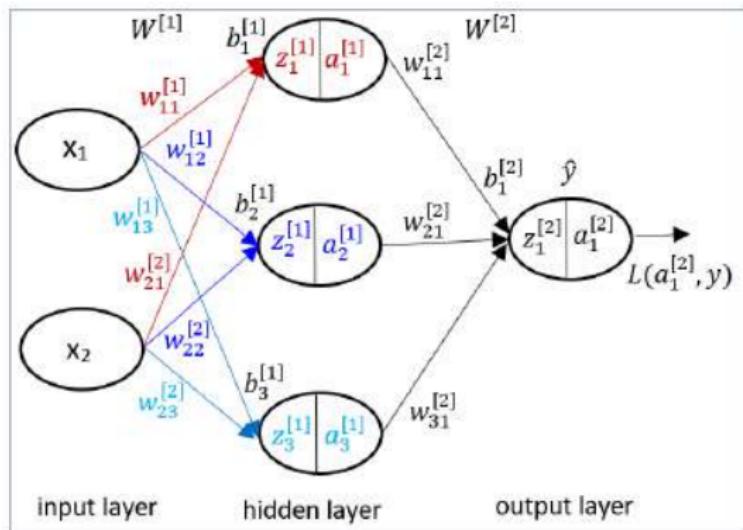
$$z = w_1x_1 + w_2x_2 + b$$

$$\hat{y} = a = \sigma(z)$$

Stack multiple sigmoid neurons to create Neural network



Neural Network: Representations

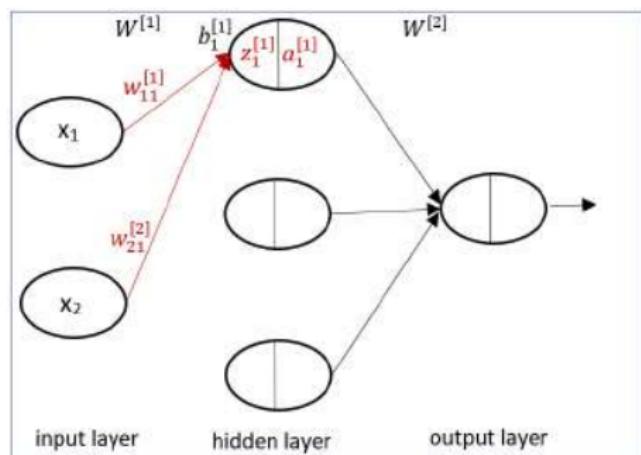


Neural Network: Forward Pass

Compute output at first hidden unit:

$$z_1^{[1]} = w_{11}^{[1]}x_1 + w_{21}^{[1]}x_2 + b_1^{[1]}$$

$$a_1^{[1]} = \sigma(z_1^{[1]})$$

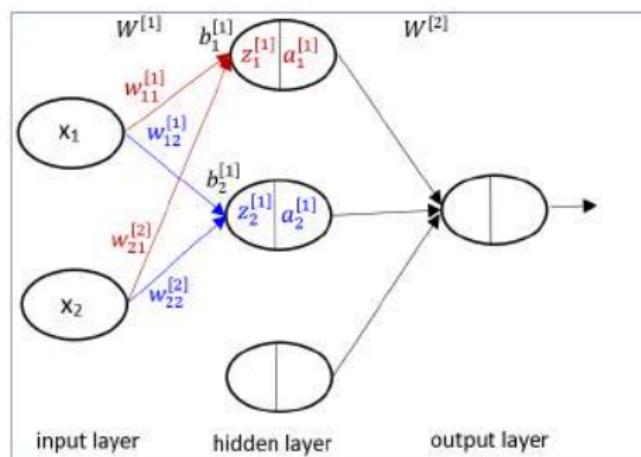


Neural Network: Forward Pass

Compute output at second hidden unit:

$$z_2^{[1]} = w_{12}^{[1]}x_1 + w_{22}^{[1]}x_2 + b_2^{[1]}$$

$$a_2^{[1]} = \sigma(z_2^{[1]})$$

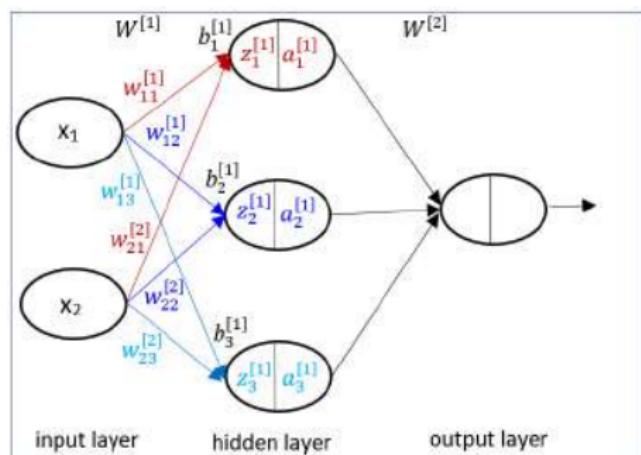


Neural Network: Forward Pass

Compute output at third hidden unit:

$$z_3^{[1]} = w_{13}^{[1]}x_1 + w_{23}^{[1]}x_2 + b_3^{[1]}$$

$$a_3^{[1]} = \sigma(z_3^{[1]})$$

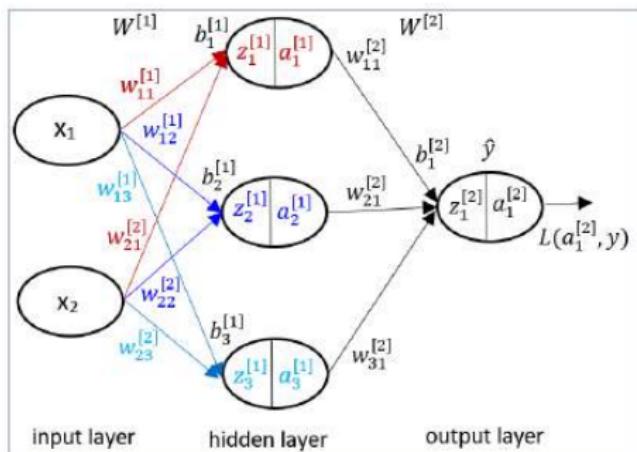


Neural Network: Forward Pass

Compute output at the output layer:

$$z_1^{[2]} = w_{11}^{[2]} a_1^{[1]} + w_{21}^{[2]} a_2^{[1]} + w_{31}^{[2]} a_3^{[1]} + b_1^{[2]}$$

$$a_1^{[2]} = \sigma(z_1^{[2]}) = \hat{y}$$



Neural Network: Forward Pass

Putting everything together:

$$z_1^{[1]} = w_{11}^{[1]}x_1 + w_{21}^{[1]}x_2 + b_1^{[1]}$$

$$a_1^{[1]} = \sigma(z_1^{[1]})$$

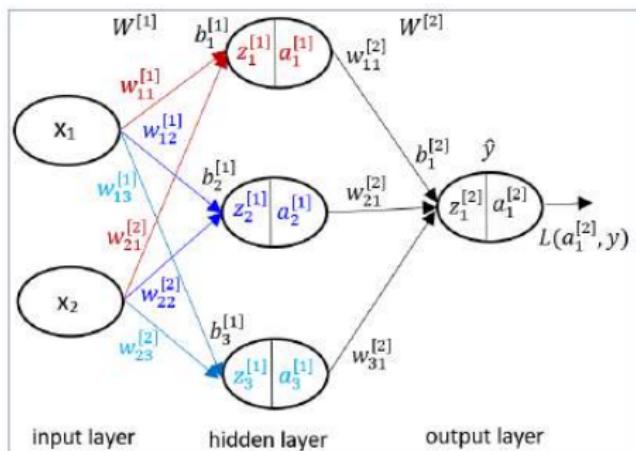
$$z_2^{[1]} = w_{12}^{[1]}x_1 + w_{22}^{[1]}x_2 + b_2^{[1]}$$

$$a_2^{[1]} = \sigma(z_2^{[1]})$$

$$z_3^{[1]} = w_{13}^{[1]}x_1 + w_{23}^{[1]}x_2 + b_3^{[1]}$$

$$a_3^{[1]} = \sigma(z_3^{[1]})$$

$$z_1^{[2]} = w_{11}^{[2]}a_1^{[1]} + w_{21}^{[2]}a_2^{[1]} + w_{31}^{[2]}a_3^{[1]} + b_1^{[2]}; a_1^{[2]} = \sigma(z_1^{[2]}) = \hat{y}$$



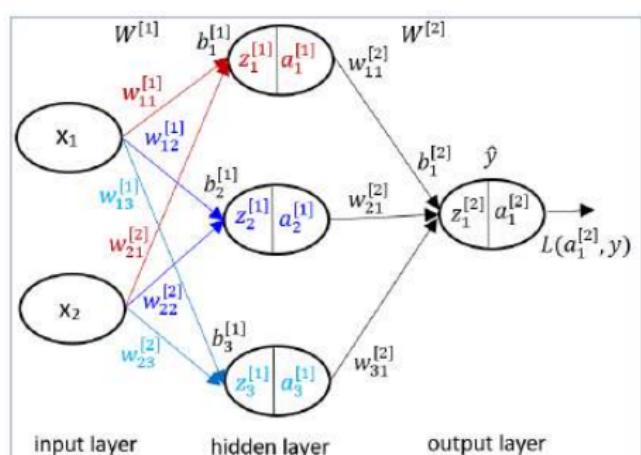
NN: outputs at different layers

$$\mathbf{z}^{[1]} = W^{[1]T} \mathbf{x} + b^{[1]}$$

$$\mathbf{a}^{[1]} = \sigma(\mathbf{z}^{[1]})$$

$$\mathbf{z}^{[2]} = W^{[2]T} \mathbf{a}^{[1]} + b^{[2]}$$

$$\hat{y} = \mathbf{a}^{[2]} = \sigma(\mathbf{z}^{[2]})$$



Neural Network: Vector & Matrix Form

$$\mathbf{z}^{[1]} = W^{[1]T} \mathbf{x} + b^{[1]}$$

$$\mathbf{z}^{[1]} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \end{bmatrix}_{3 \times 1} = \begin{bmatrix} w_{11}^{[1]} & w_{21}^{[1]} \\ w_{12}^{[1]} & w_{22}^{[1]} \\ w_{13}^{[1]} & w_{23}^{[1]} \end{bmatrix}_{3 \times 2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{2 \times 1} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \end{bmatrix}_{3 \times 1}$$

$$\mathbf{a}^{[1]} = \sigma(\mathbf{z}^{[1]}) = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \end{bmatrix}_{3 \times 1}$$

Neural Network: Vector & Matrix Form

$$\mathbf{z}^{[2]} = W^{[2]T} \mathbf{a}^{[1]} + b^{[2]}$$

$$\mathbf{z}^{[2]} = \left[z_1^{[2]} \right]_{1 \times 1} = \begin{bmatrix} w_{11}^{[2]} & w_{21}^{[2]} & w_{31}^{[2]} \end{bmatrix}_{1 \times 3} \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \end{bmatrix}_{3 \times 1} + b^{[2]}$$

$$\mathbf{a}^{[2]} = \sigma(\mathbf{z}^{[2]}) = \hat{y} \text{ (scalar)}$$

NN: single training example

$$\mathbf{z}^{[1]} = W^{[1]T} \mathbf{x} + b^{[1]}$$

$$\mathbf{a}^{[1]} = \sigma(\mathbf{z}^{[1]})$$

$$\mathbf{z}^{[2]} = W^{[2]T} \mathbf{a}^{[1]} + b^{[2]}$$

$$\hat{y} = \mathbf{a}^{[2]} = \sigma(\mathbf{z}^{[2]})$$

NN: m training examples (2-dim)

$$Z^{[1]}_{3 \times m} = W^{[1]T}_{3 \times 2} X_{2 \times m} + b^{[1]}$$

$$A^{[1]}_{3 \times m} = \sigma(Z^{[1]}_{3 \times m})$$

$$Z^{[2]}_{1 \times m} = W^{[2]T}_{1 \times 3} A^{[1]}_{3 \times m} + b^{[2]}$$

$$A^{[2]}_{1 \times m} = \sigma(Z^{[2]})$$

$$X = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(m)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(m)} \end{bmatrix}_{2 \times m}$$

$$Z^{[1]} = \begin{bmatrix} z_1^{1} & \dots & z_1^{[1](m)} \\ z_2^{1} & \dots & z_2^{[1](m)} \\ z_3^{1} & \dots & z_3^{[1](m)} \end{bmatrix}_{3 \times m}$$

NN: for m training examples (d-dim)

- **to generalise:**

- m training examples
- d -dimensional features
- h hidden units

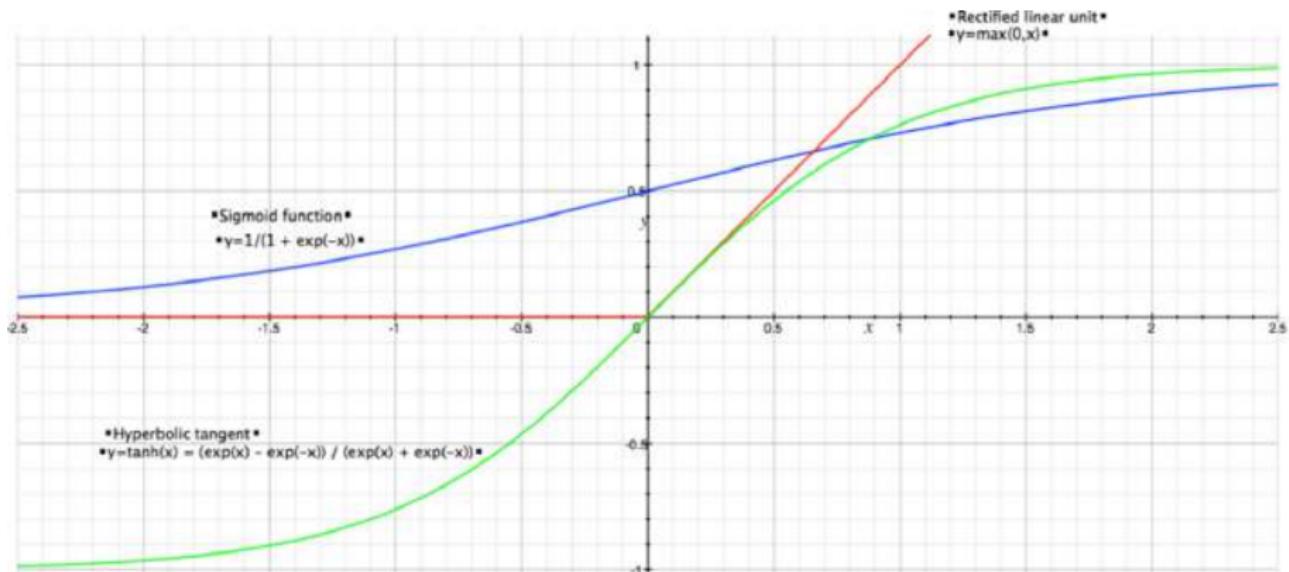
$$Z^{[1]}_{h \times m} = W^{[1]T}_{h \times d} X_{d \times m} + b^{[1]}$$

$$A^{[1]}_{h \times m} = \sigma(Z^{[1]}_{h \times m})$$

$$Z^{[2]}_{1 \times m} = W^{[2]T}_{1 \times h} A^{[1]}_{h \times m} + b^{[2]}$$

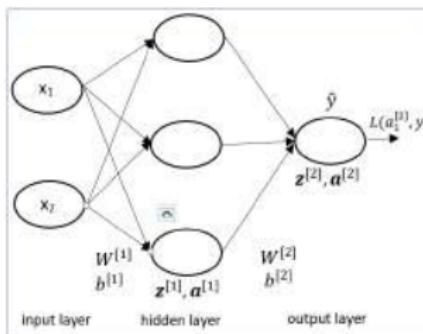
$$A^{[2]}_{1 \times m} = \sigma(Z^{[2]})$$

Activation Functions



NN: k -classes

- so far, we had NN one output unit, computed $P(y = 1 | \mathbf{x})$
- used sigmoid to classify, useful for binary classification
- what if we have k outputs (e.g., sentiment analysis)
 - 2 classes (positive, negative) vs.
 - 5 classes (strongly negative, weakly negative, neutral, weakly positive, strongly positive)



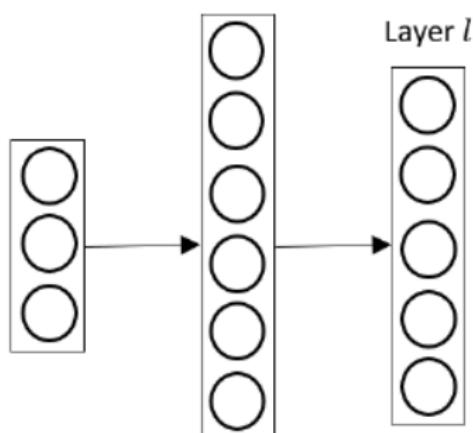
NN: softmax activation

- k -classes $\rightarrow k$ -output nodes \rightarrow we output K -length vector

$$\mathbf{z}^{[l]}_{5 \times 1} = W^{[l]} \mathbf{a}^{[l-1]} + b^{[l]}$$

$$\mathbf{a}^{[l]}_{5 \times 1},$$

where $a_i = \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}}$, for $i = 1, \dots, K$



NN: softmax - Example

$$\mathbf{z}^{[l]} = \begin{bmatrix} 7 \\ 5 \\ 2 \\ -1 \\ 3 \end{bmatrix}; \mathbf{e}^{\mathbf{z}^{[l]}} = \begin{bmatrix} e^7 \\ e^5 \\ e^2 \\ e^{-1} \\ e^3 \end{bmatrix} = \begin{bmatrix} 1096.63 \\ 148.4 \\ 7.4 \\ 0.4 \\ 20.1 \end{bmatrix}$$

let $\sum_{k=1}^K e^{z_k} = 1272.93$

$$\text{where } a_1 = \frac{e^{z_1}}{\sum_{k=1}^K e^{z_k}} = \frac{1096.63}{1272.93}$$

$$\text{where } a_1 = 0.86$$

Following this:

$$\mathbf{a}^{[l]} = \begin{bmatrix} 0.8615 \\ 0.1165 \\ 0.0058 \\ 0.0003 \\ 0.0157 \end{bmatrix}$$

NN: Backpropagation

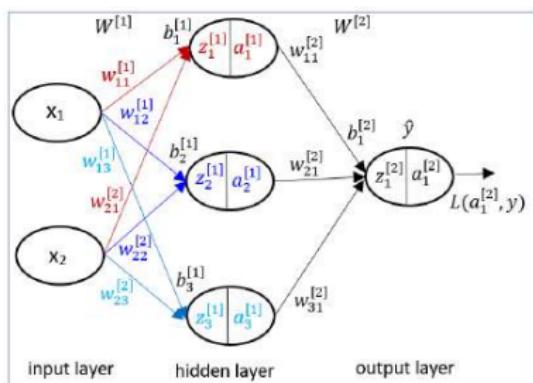
- we have computed *loss* or *error* function at the final output layer.

$$\mathbf{z}^{[1]} = W^{[1]} \mathbf{x} + b^{[1]}$$

$$\mathbf{a}^{[1]} = \sigma(\mathbf{z}^{[1]})$$

$$\mathbf{z}^{[2]} = W^{[2]} \mathbf{a}^{[1]} + b^{[2]}$$

$$\hat{y} = \mathbf{a}^{[2]} = \sigma(\mathbf{z}^{[2]})$$



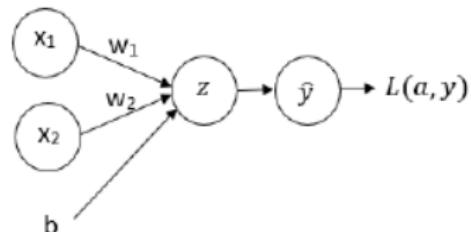
- update parameters $W^{[1]}, W^{[2]}, b^{[1]}, b^{[2]}$

Logistic Regression: Compute Gradients

Forward Pass

$$z = w_1x_1 + w_2x_2 + b$$

$$\hat{y} = a = \sigma(z)$$



$$L = L(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$$

Backpropagate Errors

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial a} \frac{\partial a}{\partial z} \frac{\partial z}{\partial w_1}$$

$$\frac{\partial L}{\partial w_1} = -\frac{y}{a} + \frac{1-y}{1-a} = \frac{a-y}{a(1-a)}; \frac{\partial a}{\partial z} = a(1-a); \frac{\partial z}{\partial w_1} = x_1$$

$$\frac{\partial L}{\partial w_1} = (a - y)x_1; \frac{\partial L}{\partial w_2} = (a - y)x_2; \frac{\partial L}{\partial b} = a - y$$

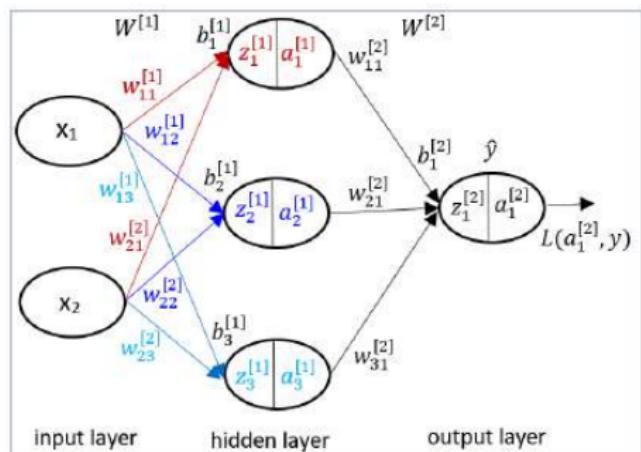
NN: Backpropagation

$$\mathbf{z}^{[1]} = W^{[1]} \mathbf{x} + b^{[1]}$$

$$\mathbf{a}^{[1]} = \sigma(\mathbf{z}^{[1]})$$

$$\mathbf{z}^{[2]} = W^{[2]} \mathbf{a}^{[1]} + b^{[2]}$$

$$\hat{y} = \mathbf{a}^{[2]} = \sigma(\mathbf{z}^{[2]})$$



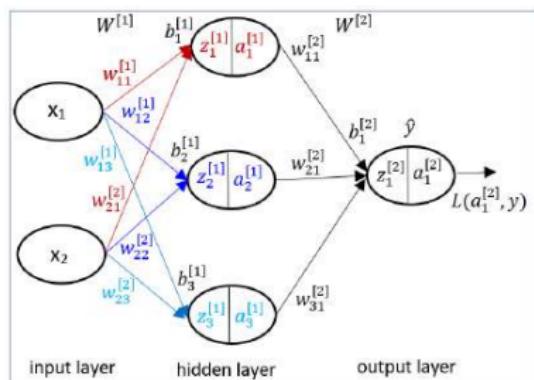
- update parameters $W^{[1]}, W^{[2]}, b^{[1]}, b^{[2]}$

NN: Backpropagation

update parameters $W^{[2]}$; compute $\frac{\partial L}{\partial W^{[2]}}$

$$\frac{\partial L}{\partial W^{[2]}} = \frac{\partial L}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial W^{[2]}}$$

(chain rule: $\frac{\partial y}{\partial x} = \frac{\partial y}{\partial z} \frac{\partial z}{\partial x}$)



NN: Backpropagation

1. compute: $\frac{\partial L}{\partial \mathbf{a}^{[2]}}$

$$L = L(\mathbf{a}^{[2]}, y) = -(y \log(\mathbf{a}^{[2]}) + (1 - y) \log(1 - \mathbf{a}^{[2]}))$$

(using log-likelihood of cross entropy function)

$$\frac{\partial L}{\partial \mathbf{a}^{[2]}} = -(y \times \frac{1}{\mathbf{a}^{[2]}} + (1 - y) \times \frac{1}{1 - \mathbf{a}^{[2]}}) \times (-1)$$

$$\frac{\partial L}{\partial \mathbf{a}^{[2]}} = -\left(\frac{y}{\mathbf{a}^{[2]}} - \frac{1 - y}{1 - \mathbf{a}^{[2]}} \right)$$

$$\frac{\partial L}{\partial \mathbf{a}^{[2]}} = \frac{\mathbf{a}^{[2]} - y}{\mathbf{a}^{[2]}(1 - \mathbf{a}^{[2]})}$$

NN: Backpropagation

2. compute: $\frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}}$

$$\mathbf{a}^{[2]} = \frac{1}{1 + e^{-(\mathbf{z}^{[2]})}}$$

$$\frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}} = \mathbf{a}^{[2]}(1 - \mathbf{a}^{[2]})$$

NN: Backpropagation

3. compute: $\frac{\partial \mathbf{z}^{[2]}}{\partial W^{[2]}}$

$$\mathbf{z}^{[2]} = W^{[2]}\mathbf{a}^{[1]} + b^{[2]}$$

$$\frac{\partial \mathbf{z}^{[2]}}{\partial W^{[2]}} = \mathbf{a}^{[1]}$$

We have now computed: $\frac{\partial L}{\partial \mathbf{a}^{[2]}}$; $\frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}}$; $\frac{\partial \mathbf{z}^{[2]}}{\partial W^{[2]}}$

$$\frac{\partial L}{\partial W^{[2]}} = \frac{\partial L}{\partial \mathbf{a}^{[2]}} \frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}} \frac{\partial \mathbf{z}^{[2]}}{\partial W^{[2]}}$$

$$\frac{\partial L}{\partial W^{[2]}} = \frac{\mathbf{a}^{[2]} - y}{\mathbf{a}^{[2]}(1 - \mathbf{a}^{[2]})} \mathbf{a}^{[2]}(1 - \mathbf{a}^{[2]})\mathbf{a}^{[1]}$$

$$\frac{\partial L}{\partial W^{[2]}} = (\mathbf{a}^{[2]} - y)\mathbf{a}^{[1]}$$

NN: Backpropagation

update parameters $b^{[2]}$; compute $\frac{\partial L}{\partial b^{[2]}}$

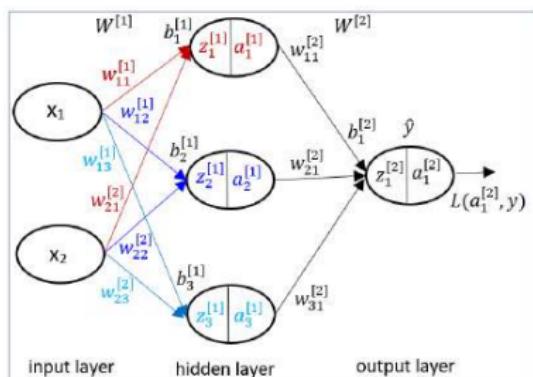
$$\frac{\partial L}{\partial b^{[2]}} = \frac{\partial L}{\partial \mathbf{a}^{[2]}} \frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}} \frac{\partial \mathbf{z}^{[2]}}{\partial b^{[2]}}$$

we know: $\frac{\partial L}{\partial \mathbf{a}^{[2]}}$; $\frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}}$

$$\mathbf{z}^{[2]} = W^{[2]} \mathbf{a}^{[1]} + b^{[2]}$$

$$\frac{\partial \mathbf{z}^{[2]}}{\partial b^{[2]}} = 1$$

$$\text{Thus } \frac{\partial L}{\partial b^{[2]}} = (\mathbf{a}^{[2]} - y)$$



NN: Backpropagation

update parameters $W^{[1]}$; compute $\frac{\partial L}{\partial W^{[1]}}$

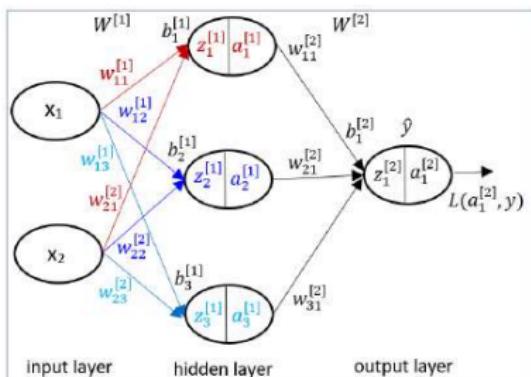
$$\frac{\partial L}{\partial W^{[1]}} = \frac{\partial L}{\partial \mathbf{a}^{[2]}} \frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}} \frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{a}^{[1]}} \frac{\partial \mathbf{a}^{[1]}}{\partial \mathbf{z}^{[1]}} \frac{\partial \mathbf{z}^{[1]}}{\partial W^{[1]}}$$

we know: $\frac{\partial L}{\partial \mathbf{a}^{[2]}} \frac{\partial \mathbf{a}^{[2]}}{\partial \mathbf{z}^{[2]}} = (\mathbf{a}^{[2]} - y)$

compute $\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{a}^{[1]}}$

$$\mathbf{z}^{[2]} = W^{[2]} \mathbf{a}^{[1]} + b^{[2]}$$

$$\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{a}^{[1]}} = W^{[2]}$$



NN: Backpropagation

compute: $\frac{\partial \mathbf{a}^{[1]}}{\partial \mathbf{z}^{[1]}}$

$$\mathbf{a}^{[1]} = \frac{1}{1 + e^{-(\mathbf{z}^{[1]})}}$$

$$\frac{\partial \mathbf{a}^{[1]}}{\partial \mathbf{z}^{[1]}} = g^{[1]}'(\mathbf{z}^{[1]}) = \mathbf{a}^{[1]}(1 - \mathbf{a}^{[1]})$$

Finally: $\mathbf{z}^{[1]} = W^{[1]}\mathbf{x} + b^{[1]}$

$$\frac{\partial \mathbf{z}^{[1]}}{\partial W^{[1]}} = \mathbf{x}$$

NN: Backpropagation

Putting everything together:

$$\frac{\partial L}{\partial W^{[1]}} = \underbrace{\frac{\partial L}{\partial \mathbf{a}^{[2]}}}_{\mathbf{a}^{[2]} - y} \underbrace{\frac{\partial \mathbf{z}^{[2]}}{\partial \mathbf{a}^{[1]}}}_{W^{[2]}} \underbrace{\frac{\partial \mathbf{a}^{[1]}}{\partial \mathbf{z}^{[1]}}}_{g^{[1]}'(\mathbf{z}^{[1]})} \underbrace{\frac{\partial \mathbf{z}^{[1]}}{\partial W^{[1]}}}_{\mathbf{x}}$$

$$\underbrace{\frac{\partial L}{\partial W^{[1]}}}_{3 \times 2} = \underbrace{\mathbf{a}^{[2]} - y}_{1 \times 1} \underbrace{W^{[2]}}_{3 \times 1} \underbrace{g^{[1]}'(\mathbf{z}^{[1]})}_{3 \times 1} \underbrace{\mathbf{x}}_{2 \times 1}$$

$$\underbrace{\frac{\partial L}{\partial W^{[1]}}}_{3 \times 2} = \underbrace{W^{[2] T}}_{3 \times 1} \circ \underbrace{g^{[1]}'(\mathbf{z}^{[1]})}_{3 \times 1} \underbrace{\mathbf{a}^{[2]} - y}_{1 \times 1} \underbrace{\mathbf{x}^T}_{1 \times 2}$$

NN: Gradient Descent

for any single layer ℓ , the update rule is defined by:

$$W^{[\ell]} = W^{[\ell]} - \alpha \frac{\partial J}{\partial W^{[\ell]}}$$

where $J = \frac{1}{m} \sum_{i=1}^m \mathcal{L}^{(i)}$, where $\mathcal{L}^{(i)}$ is the loss for the single example.

In this assignment, we will measure the cosine similarity between two given sentences using numpy. Let us assume the two sentences are:

In [2] :

```
A = "I love data mining"  
B = "I hate data mining"
```

First thing we need to do is to import numpy

In [3] :

```
import numpy
```

We will get the list of words in each sentence by calling the split() method in strings. lower() will convert the sentence into lowercase.

In [4] :

```
wordsA = A.lower().split()  
wordsB = B.lower().split()
```

In [5] :

```
print wordsA  
print wordsB
```

```
['i', 'love', 'data', 'mining']  
['i', 'hate', 'data', 'mining']
```

Before we represent each sentence using a numpy array, we must know the dimensionality of the feature space (i.e. the total number of unique features in) all instances. We can use the set class for this purpose.

In [6] :

```
vocab = set(wordsA)  
vocab = vocab.union(set(wordsB))
```

Lets print all the features in the vocabulary vocab.

In [7] :

```
print vocab
```

```
set(['mining', 'love', 'i', 'hate', 'data'])
```

We see that vocab contains all the features in all sentences. However, we need to have a one-to-one mapping between features and their ids in the feature space. Each feature must be associated with a single dimension in the feature space. This can be done easily by converting the vocab into a list. In a list we have a one-to-one mapping between each element and its position in the list, starting from 0.

In [8] :

```
vocab = list(vocab)
```

You can see the list of unique features as follows:

In [9] :

```
print vocab
```

```
['mining', 'love', 'i', 'hate', 'data']
```

Now lets declare two 1D-arrays (which will act as feature vectors) for representing the two sentences.

In [10] :

```
vA = numpy.zeros(len(vocab), dtype=float)  
vB = numpy.zeros(len(vocab), dtype=float)
```

`numpy.zeros(shape, data_type)` returns arrays of the specified shape and size, filled with zeros. In our case we need 1D arrays filled with float type zeros. `dtype` is a keyword indicating the data type to store in the array. Unlike python lists where we could store any data type, with `numpy.arrays` we can store only values for one data type. This makes `numpy.arrays` efficient in terms of both speed and memory.

We will now go through the list of features for the first sentence and increment the corresponding feature value in the vector.

In [11] :

```
for w in wordsA:  
    i = vocab.index(w)  
    vA[i] += 1
```

Lets print this vector.

In [12] :

```
print vA
```

```
[ 1.  1.  1.  0.  1.]
```

Take a moment to check this result. The first feature in the vocabulary is "mining". This features occurs in the first sentence. Therefore, its value is set to 1. Go through all the features in the first sentence and confirm that this is true.

We can do the same procedure to populate the vector for the second sentence as follows:

In [13] :

```
for w in wordsB:  
    i = vocab.index(w)  
    vB[i] += 1
```

In [14] :

```
print vB
```

```
[ 1.  0.  1.  1.  1.]
```

Again check that the vector is correctly populated for the second sentence.

Next, we will compute the cosine similarity between the two vectors. The cosine similarity between two vectors x and y is defined as follows:

```
cos(x,y) = numpy.dot(x,y) / (numpy.sqrt(numpy.dot(x,x)) * numpy.sqrt(numpy.dot(y,y)))
```

In [16] :

```
cos = numpy.dot(vA, vB) / (numpy.sqrt(numpy.dot(vA,vA)) * numpy.sqrt(numpy.dot(vB,vB)))
```

In [17] :

```
print cos
```

0.75

Therefore, the cosine similarity between the two sentences is 0.75. Here, `numpy.dot` computes the inner-product between two vectors, and `numpy.sqrt` computes the square root.

Of course, this is not the only way to compute cosine similarity. Numpy provides the linear algebra routines such as norm of a vector in the `numpy.linalg` package. If you use this, you do not need to call the `numpy.sqrt`. For example,

In [18] :

```
print numpy.dot(vA, vB) / (numpy.linalg.norm(vA) * numpy.linalg.norm(vB))
```

0.75

As you can see you get the same result but the code looks better.

Perceptron

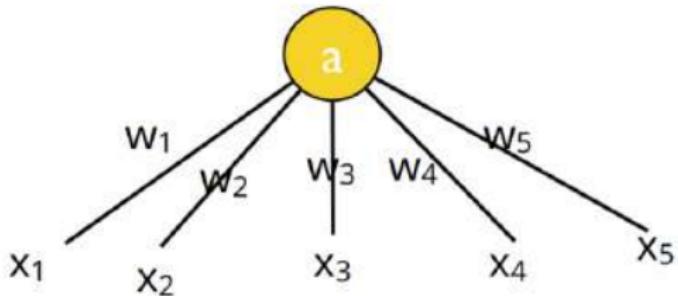


Bio-inspired model

- Perceptron is a bio-inspired algorithm that tries to mimic a single neuron.
- We simply multiply each input (feature) by a weight and check whether this weighted sum (activation) is greater than a threshold.
- If so, then we “fire” the neuron (i.e., a decision is made based on the activation).

A Single neuron

$$\text{activation (score)} = a = W_1X_1 + W_2X_2 + W_3X_3 + W_4X_4 + W_5X_5$$



```
if a > θ then  
    output = 1  
else  
    output = 0
```

If the activation is greater than a predefined threshold, then the neuron fires.

Bias

- Often we need to adjust a fixed *shift* from zero, if the “interesting” region happens to be far from the origin.
- We adjust the previous model by including a bias term b as follows:

$$a = b + \sum_{i=1}^D w_i x_i$$

Notational trick

- By introducing a feature that is always ON (i.e., $X_0 = 1$ for all instances), we can squeeze the bias term b into the weight vector by setting $w_0 = b$

$$a = \sum_{i=1}^D w_i x_i = \mathbf{w}^\top \mathbf{x}$$

This is more “elegant” as we can write the activation as the inner-product between weight vector and feature vector. However, we should keep in mind that bias term still appears in the model

Perceptron

- Consider only one training instance at a time
 - online learning
 - k-NN considers ALL instances (batch learning)
- Learn only if we make a mistake when we classify using the current weight vector. Otherwise, we do not make adjustments to the weight vector
 - Error-driven learning

Perceptron

Algorithm 5 PERCEPTRONTRAIN(\mathbf{D} , $MaxIter$)

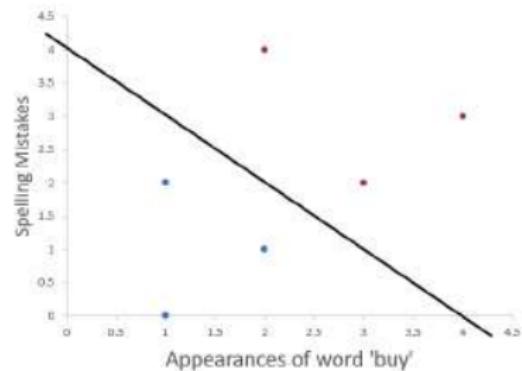
```
1:  $w_d \leftarrow 0$ , for all  $d = 1 \dots D$                                 // initialize weights
2:  $b \leftarrow 0$                                                        // initialize bias
3: for  $iter = 1 \dots MaxIter$  do
4:   for all  $(x, y) \in \mathbf{D}$  do
5:      $a \leftarrow \sum_{d=1}^D w_d x_d + b$                                 // compute activation for this example
6:     if  $ya \leq 0$  then
7:        $w_d \leftarrow w_d + yx_d$ , for all  $d = 1 \dots D$                 // update weights
8:        $b \leftarrow b + y$                                               // update bias
9:     end if
10:   end for
11: end for
12: return  $w_0, w_1, \dots, w_D, b$ 
```

Algorithm 6 PERCEPTRONTEST($w_0, w_1, \dots, w_D, b, \hat{x}$)

```
1:  $a \leftarrow \sum_{d=1}^D w_d \hat{x}_d + b$                                 // compute activation for the test example
2: return SIGN( $a$ )
```

Example

Spelling Mistakes	Appearances of word 'buy'	Spam or Ham
x_1	x_2	y
3	2	-1
2	1	1
4	3	-1
1	0	1
1	2	1
2	4	-1



Update Weights

	x_1	x_2	y	w_1	w_2	b	a	ya
1	3	2	-1	0	0	0	0	0

- 1 initialise weight vector w_d ($w_1 = w_2 = 0$) and bias ($b = 0$)
- 2 compute activation :

$$a = \sum_{i=0}^D w_d x_d + b \implies w_1 \cdot x_1 + w_2 \cdot x_2 + b \implies a = 0, ya = 0$$

- 3 since $ya \leq 0$, update weight vector and bias

$$w_d \leftarrow w_d + yx_d, \text{ thus } w_1 = w_1 - y \cdot x_1 = -3$$

- 4 similarly, we get $w_2 = -2$ and $b = b + y = -1$

Update Weights

	x_1	x_2	y	w_1	w_2	b	a	ya
1	3	2	-1	-3	-2	-1	0	0
2	2	1	1	-1	-1	0	-9	-9
3	4	3	-1				-7	7

Second training example:

- ① we use updated weights from first training example
- ② we get $a = -9$ and $ya = -9$ (because $y = 1$)
- ③ since $ya \leq 0$, update parameters

Third training example:

- ① we use updated weights from the second training example
- ② we get $a = -7$ and $ya = 7$ (because $y = -1$)
- ③ since $ya > 0$, we don't update and go to next instance.

Detected Errors

	x_1	x_2	y	w_1	w_2	b	a	ya
1	3	2	-1	-3	-2	-1	0	0
2	2	1	1	-1	-1	0	-9	-9
3	4	3	-1				-7	7

- In Line 6 of Perceptron Train code, we have:
 - $ya \leq 0$
 - if the current instance is positive ($y = +1$), we need a positive activation ($a > 0$) to have a correct prediction
 - if the current instance is negative ($y = -1$), we need a negative activation ($a < 0$) to have a correct prediction
 - in both cases $ya > 0$
 - therefore, if $ya \leq 0$, we have a misclassification

Update Rule - Intuitive Explanation

	x_1	x_2	y	w_1	w_2	b	a	ya
1	3	2	-1	-3	-2	-1	0	0
2	2	1	1	-1	-1	0	-9	-9
3	4	3	-1				-7	7

- Perceptron update rule is
 - $\mathbf{w} = \mathbf{w} + y\mathbf{x}$
- If we incorrectly classify a positive instance as negative
 - We should have a higher (more positive) activation to avoid this
 - We should increase $\mathbf{w}^T \mathbf{x}$
 - Therefore, we should ADD the current instance to the weight vector

Update Rule - Intuitive Explanation

	x_1	x_2	y	w_1	w_2	b	a	ya
1	3	2	-1	-3	-2	-1	0	0
2	2	1	1	-1	-1	0	-9	-9
3	4	3	-1				-7	7

- If we incorrectly classify a negative instance as positive
 - We should have a lower (more negative) activation to avoid this
 - We should decrease $\mathbf{w}^T \mathbf{x}$
 - Therefore, we should DEDUCT the current instance from the weight vector

Update rule - Math Explanation

- Let's look at a misclassified positive example ($y_n = +1$)
 - Perceptron (wrongly) thinks $\mathbf{w}_{old}^\top \mathbf{x}_n + b_{old} < 0$
- Updates would be:
 - $\mathbf{w}_{new} = \mathbf{w}_{old} + y_n \mathbf{x}_n = \mathbf{w}_{old} + \mathbf{x}_n$ (since $y_n = +1$)
 - $b_{new} = b_{old} + y_n = b_{old} + 1$
$$\begin{aligned}\mathbf{w}_{new}^\top \mathbf{x}_n + b_{new} &= (\mathbf{w}_{old} + \mathbf{x}_n)^\top \mathbf{x}_n + b_{old} + 1 \\ &= (\mathbf{w}_{old}^\top \mathbf{x}_n + b_{old}) + \mathbf{x}_n^\top \mathbf{x}_n + 1\end{aligned}$$
- Thus $\mathbf{w}_{new}^\top \mathbf{x}_n + b_{new}$ is **less negative** than $\mathbf{w}_{old}^\top \mathbf{x}_n + b_{old}$
 - we are making ourselves more correct on this example.

Update rule - Math Explanation

- Let's look at a misclassified negative example ($y_n = -1$)
 - Perceptron (wrongly) thinks $\mathbf{w}_{old}^\top \mathbf{x}_n + b_{old} > 0$
- Updates would be:
 - $\mathbf{w}_{new} = \mathbf{w}_{old} + y_n \mathbf{x}_n = \mathbf{w}_{old} - \mathbf{x}_n$ (since $y_n = -1$)
 - $b_{new} = b_{old} + y_n = b_{old} - 1$
$$\begin{aligned}\mathbf{w}_{new}^\top \mathbf{x}_n + b_{new} &= (\mathbf{w}_{old} - \mathbf{x}_n)^\top \mathbf{x}_n + b_{old} - 1 \\ &= (\mathbf{w}_{old}^\top \mathbf{x}_n + b_{old}) - \mathbf{x}_n^\top \mathbf{x}_n - 1\end{aligned}$$
- Thus $\mathbf{w}_{new}^\top \mathbf{x}_n + b_{new}$ is less positive than $\mathbf{w}_{old}^\top \mathbf{x}_n + b_{old}$
 - we are making ourselves more correct on this example.

Things to Remember

- There is no guarantee that we will correctly classify a misclassified instance in the next round.
- We have simply increased/decreased the activation but this adjustment might not be sufficient. We might need to do more aggressive adjustments
- There are algorithms that enforce such requirements explicitly such as the Passive Aggressive Classifier (not discussed here)

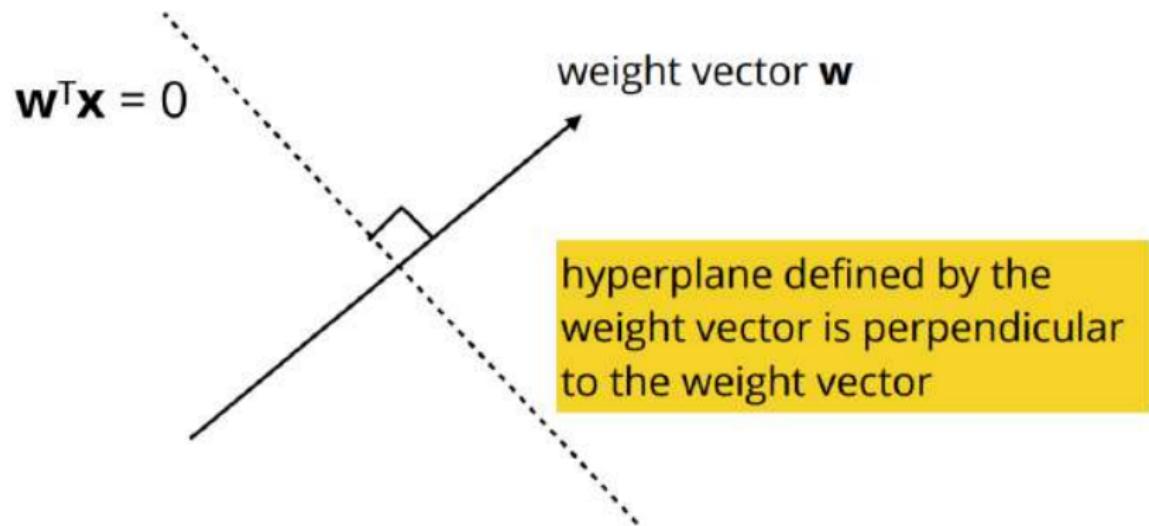
Ordering of Instances

- Ordering training instances randomly within each iteration produces good results in practice.
- Showing only all the positives first and all the negatives next is a bad idea.

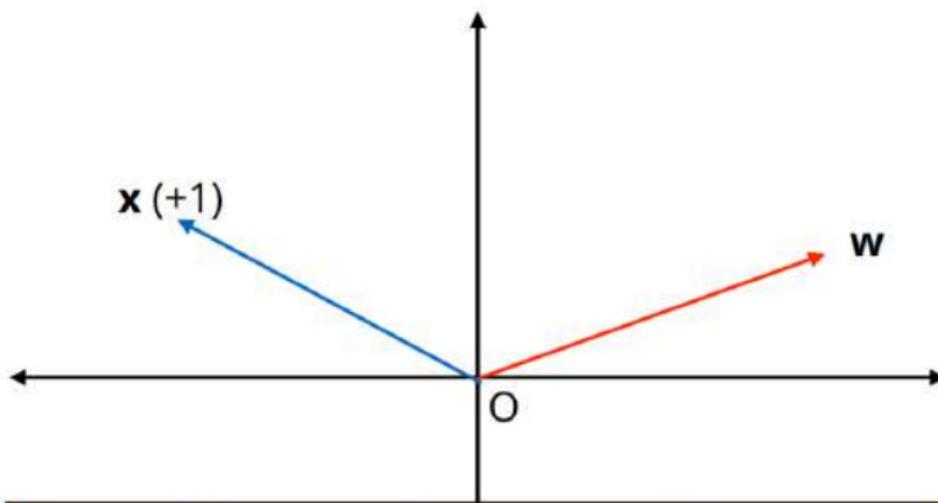
Hyperplane

- The decision in perceptron is made depending on $\mathbf{w}^T \mathbf{x} > 0$ or $\mathbf{w}^T \mathbf{x} \leq 0$
- Therefore, $\mathbf{w}^T \mathbf{x} = 0$ is the critical region (decision boundary)
- $\mathbf{w}^T \mathbf{x} = 0$ defines a hyperplane
- Example:
 - In 2D space we have $w_1x_1 + w_2x_2 = 0$ (ignoring the bias term), which is a straight line through the origin.
 - In N dimensional space this is an (N-1) dimensional hyperplane

Geometric Interpretation of Hyperplane

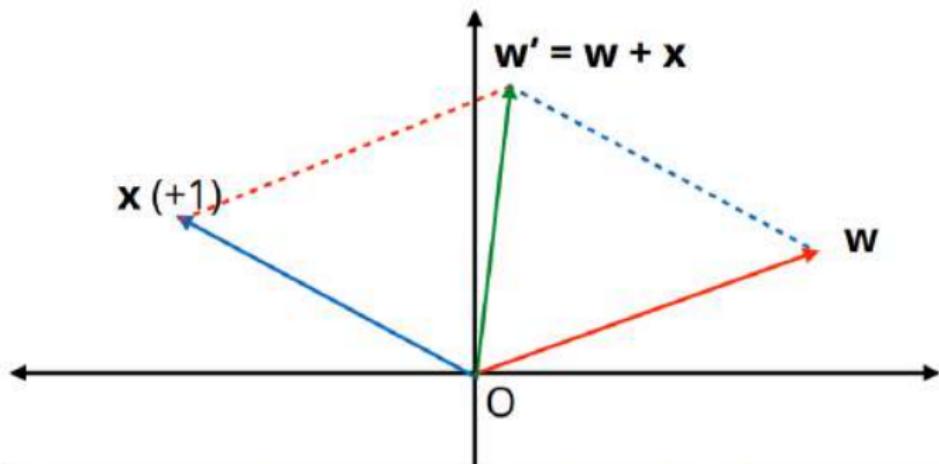


Geometric Interpretation



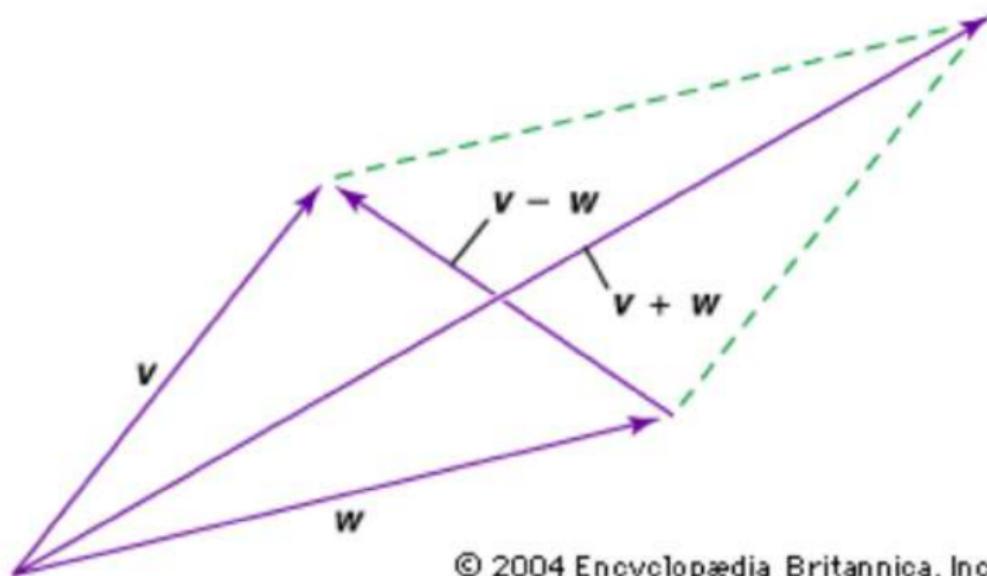
The angle between the current weight vector **w** and the positive instance **x** is greater than 90° . Therefore, $\mathbf{w}^T \mathbf{x} < 0$, and this instance is going to get misclassified as negative.

Geometric Interpretation



The new weight vector w' is the addition of $w + x$ according to the perceptron update rule. It lies in between x and w . Notice that the angle between w' and x is less than 90° . Therefore, x will be classified as positive by w' .

Vector algebra revision



© 2004 Encyclopaedia Britannica, Inc.

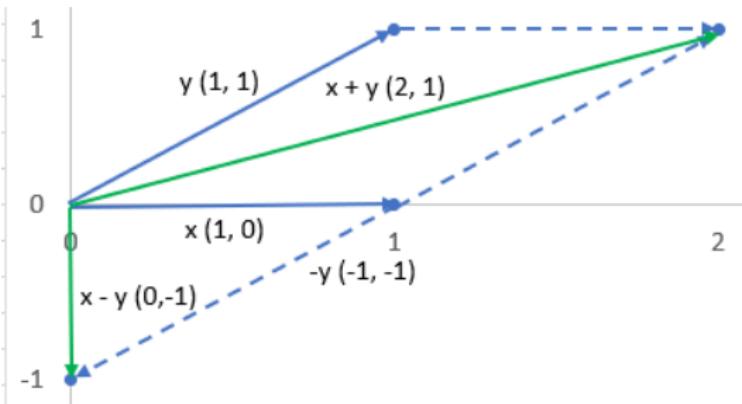
Quiz 1

- Let $\mathbf{x} = (1, 0)^\top$ and $\mathbf{y} = (1, 1)^\top$. compute $\mathbf{x} + \mathbf{y}$ and $\mathbf{x} - \mathbf{y}$ using the parallelogram approach described in the previous slide.

Quiz 1

- Let $\mathbf{x} = (1, 0)^\top$ and $\mathbf{y} = (1, 1)^\top$. compute $\mathbf{x} + \mathbf{y}$ and $\mathbf{x} - \mathbf{y}$ using the parallelogram approach described in the previous slide.

$$\mathbf{x} + \mathbf{y} = (2, 1); \mathbf{x} - \mathbf{y} = (0, -1)$$

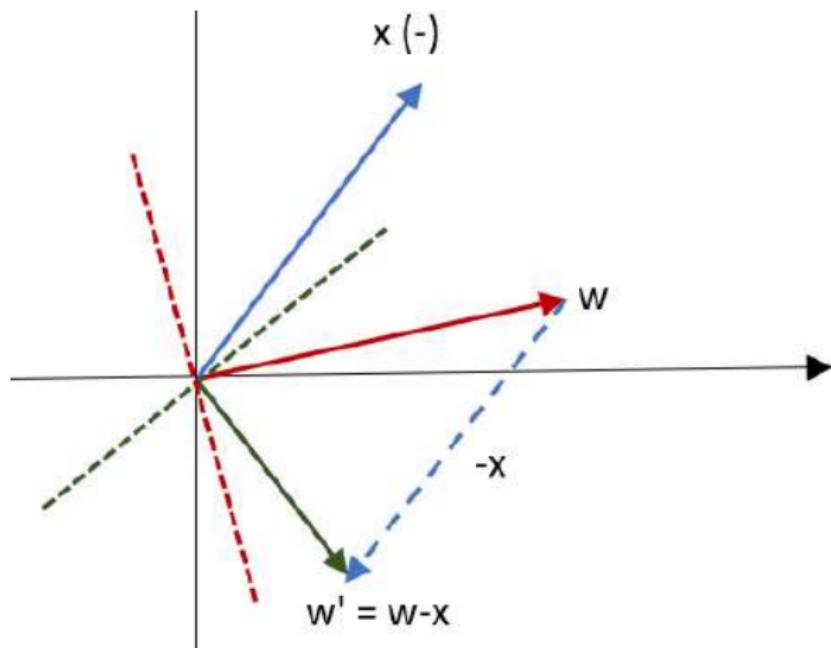


Quiz 2

- Provide a geometric interpretation for the update rule in Perceptron when a negative instance is mistaken to be positive.

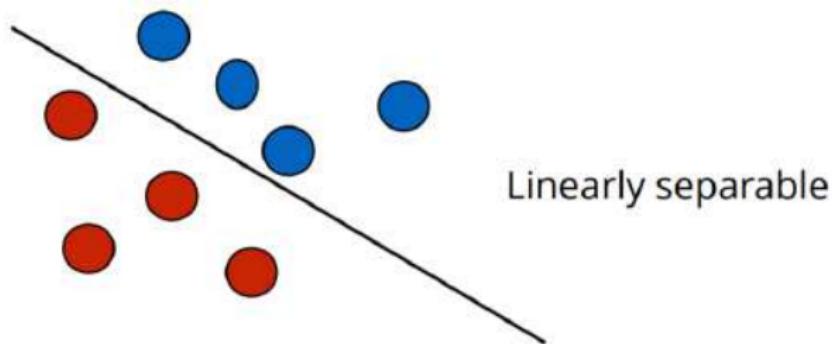
Quiz 2

- Provide a geometric interpretation for the update rule in Perceptron when a negative instance is mistaken to be positive.



Linear separability

- If a given set of positive and negative training instances can be separated into those two groups using a straight line (hyperplane), then we say that the dataset is *linearly separable*.

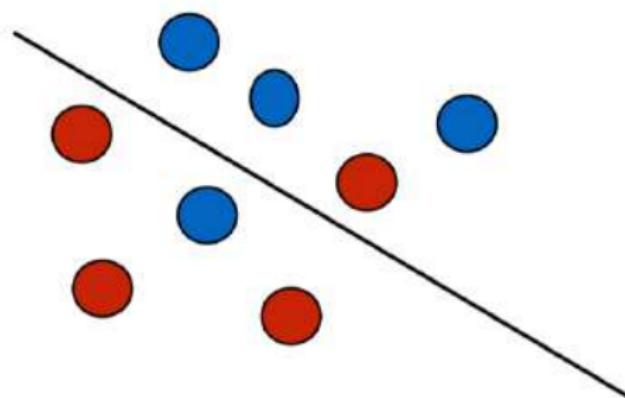


Remarks

- When a dataset is linearly separable, there can exist more than one hyperplanes that separates the dataset into positive/negative groups.
- In other words, the hyperplane that linearly separates a linearly separable dataset might not be unique.
- However, (by definition) if a dataset is nonlinearly separable, then there exist NO hyperplane that separates the dataset into positive/negative groups.

A non-linearly separable case

No matter how we draw straight lines, we cannot separate the red instances from the blue instances



Further Remarks

- When a dataset is linearly separable it can be proved that the Perceptron will always find a separating hyperplane!
- The final weight vector returned by the Perceptron is more influenced by the final training instances it sees.
 - Take the average over all weight vectors during the training (averaged Perceptron algorithm)

COMP337+527

Data Mining and Visualisation

Problem Set 0

Shagufta Scanlon

Question 1 Consider two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$ defined as $\mathbf{x} = (1, 2, -1)^\top$ and $\mathbf{y} = (-1, 0, 1)^\top$. Answer the following questions about these two vectors.

- A. Compute the length (ℓ_2 norm) of \mathbf{x} and \mathbf{y} . **(4 marks)**

$$\|\mathbf{x}\|_2 = \sqrt{1+4+1} = \sqrt{6} \text{ and } \|\mathbf{y}\|_2 = \sqrt{1+0+1} = \sqrt{2}$$
- B. Compute the inner product between \mathbf{x} and \mathbf{y} . **(2 marks)**

$$\mathbf{x}^\top \mathbf{y} = -1 + 0 + -1 = -2$$
- C. Compute the cosine of the angle between the two vectors \mathbf{x} and \mathbf{y} . **(4 marks)**

The definition of cosine similarity is $\frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}$. Therefore, the required value will be $-2/\sqrt{12}$.

- D. Compute the Euclidean distance between the end points corresponding to the two vectors \mathbf{x} and \mathbf{y} . **(4 marks)**

$$\text{The definition of the Euclidean distance is } \sqrt{\sum_i (x_i - y_i)^2}.$$

Therefore, we get $\sqrt{4+4+4} = 2\sqrt{2}$
- E. For any two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ such that $\|\mathbf{x}\|_2 = \|\mathbf{y}\|_2 = 1$ show that the following relationship holds between their cosine similarity $\cos(\mathbf{x}, \mathbf{y})$ and their Euclidean distance $\text{Euc}(\mathbf{x}, \mathbf{y})$. **(6 marks)**

$$\text{Euc}(\mathbf{x}, \mathbf{y})^2 = 2(1 - \cos(\mathbf{x}, \mathbf{y}))$$

$$\begin{aligned}
 \text{Euc}(\mathbf{x}, \mathbf{y})^2 &= (\mathbf{x} - \mathbf{y})^\top (\mathbf{x} - \mathbf{y}) \\
 &= \mathbf{x}^\top \mathbf{x} + \mathbf{y}^\top \mathbf{y} - 2\mathbf{x}^\top \mathbf{y} \\
 &= 1 + 1 - 2\cos(\mathbf{x}, \mathbf{y}) \\
 &= 2(1 - \cos(\mathbf{x}, \mathbf{y})) \quad \square
 \end{aligned}$$

Question 2 Consider a matrix $\mathbf{A} \in \mathbb{R}^{2 \times 2}$ defined as follows:

$$\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

Answer the following questions related to \mathbf{A} .

- A. Compute the transpose \mathbf{A}^\top . (2 marks)

For a matrix $\mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, $\mathbf{A}^\top = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$. Therefore,
we have

$$\mathbf{A}^\top = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

- B. Compute the determinant $\det(\mathbf{A})$. (2 marks)

$$\det(\mathbf{A}) = ac - bd = 2 \times 2 - 1 \times 1 = 3$$

- C. Compute the inverse \mathbf{A}^{-1} . (4 marks)

$$\mathbf{A}^{-1} = \frac{1}{\det \mathbf{A}} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

From which is follows,

$$\mathbf{A}^{-1} = \begin{pmatrix} 2/3 & -1/3 \\ -1/3 & 2/3 \end{pmatrix}.$$

- D. Compute the eigenvalues and eigenvectors of \mathbf{A} . (6 marks)

Eigenvector \mathbf{x} corresponding to the eigenvalue λ satisfies the equation $\mathbf{Ax} = \lambda\mathbf{x}$. From which it follows that $(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \mathbf{0}$. Therefore, $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$. In this case, we get $\det \begin{pmatrix} 2 - \lambda & 1 \\ 1 & 2 - \lambda \end{pmatrix} = 0$. Solving this second-order polynomial equation we get $\lambda = 1, 3$, which are the eigenvalues. Substituting these values separately in the eigenvalue equation we get the eigenvectors corresponding $\lambda = 1$ and $\lambda = 3$ to be respectively $(1, -1)^\top$ and $(1, 1)^\top$, subjected to a scaling factor.

Document Classification

$$P(x|y) = N! \prod_{w \in D} \frac{p(w|y)^{h(w,D)}}{h(w,D)!}$$

- N = number of words in the document x
- $p(w|y)$ = probability that the word w occurs in class y
- $h(w,D)$ = total occurrences of the word w in document D

Classifying “burger” sentiment

- Let us assume that we would like to classify whether the following document is positive (1) or negative (-1) in sentiment.
- $D = \text{"the burger i ate was an awesome burger"}$
- Further assume that we encounter these words in positive and negative classes as follows. To make our computations easier let us assume that we removed “the”, “i”, “was”, “an” as stop words.

word	+1	-1
burger	3	2
ate	3	2
awesome	4	1

Computing class conditional probabilities

word	+1	-1	$p(w +1)$	$p(w -1)$
burger	3	2	3/10	2/5
ate	3	2	3/10	2/5
awesome	4	1	4/10	1/5

Quiz

- Using the probabilities in the table in slide 21 and multinomial naive Bayes formula in slide 19, compute $P(D|+1)$ and $P(D|-1)$

$$P(D|+1) = P(\text{burger} | +1)^2 p(\text{ate} | +1)^1 p(\text{awesome} | +1)^1 \frac{4!}{2! 1! 1!}$$

$$= \left(\frac{3}{10}\right)^2 \left(\frac{3}{10}\right)^1 \left(\frac{4}{10}\right)^1 \approx \frac{24}{2}$$

$$= (0.3)^3 \times 0.4 \times 12 \quad \text{---①}$$

$$P(D|-1) = P(\text{burger} | -1)^2 p(\text{ate} | -1)^1 p(\text{awesome} | -1)^1 \frac{4!}{2! 1! 1!}$$

$$= \left(\frac{2}{5}\right)^2 \left(\frac{2}{5}\right)^1 \left(\frac{1}{5}\right)^1 \times 12$$

$$= (0.4)^3 \times 0.2 \times 12 \quad \text{---②}$$

Quiz

- Assuming that both positive and negative classes have equal probability
 - $p(+1) = p(-1) = 0.5$
- Compute $P(+1|D)$ and $P(-1|D)$ for the document in slide 22.
- Is this document D positive or negative?

$$P(+1|D) = \frac{p(+1)p(+1)}{p(D)}$$

$$\begin{aligned} p(D) &= p(\text{burger})^2 p(\text{ate})^1 p(\text{awesome})^1 \frac{4!}{2!1!1!} \\ &= \left(\frac{5}{15}\right)^2 \left(\frac{5}{15}\right)^1 \left(\frac{5}{15}\right)^1 12 \end{aligned}$$

$$\therefore P(+1|D) = \frac{(0.3)^3 \times 0.4 \times 0.5}{(1/2)^4}$$

$$P(-1|D) = \frac{p(D|-1)p(-1)}{p(D)} = \frac{(0.4)^3 \times 0.2 \times 0.5}{(1/2)^4}$$

$$\frac{P(+1|D)}{P(-1|D)} = \left(\frac{0.3}{0.4}\right)^3 \frac{0.4}{0.2} = \frac{81}{32} > 1 \quad \therefore \underline{\underline{D \text{ is positive}}}.$$

Dimensionality Reduction

COMP 527 Data Mining
Danushka Bollegala

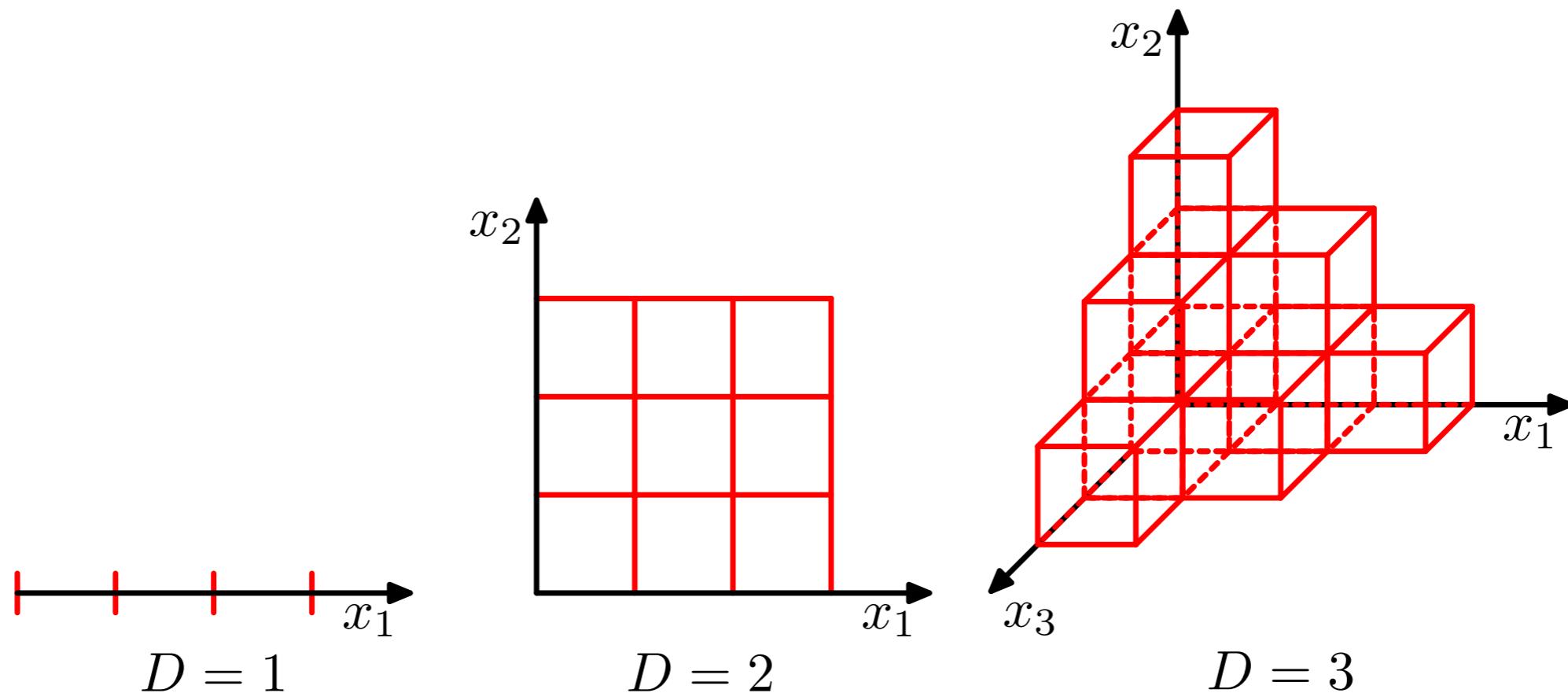


Outline

- Problems with high dimensional data
- Dimensionality Reduction Methods
 - Singular Value Decomposition (SVD)
 - Principal Component Analysis (PCA)

Problems in High Dimensions

- Curse of dimensionality
 - We need exponentially large number of data points to cover a high dimensional space

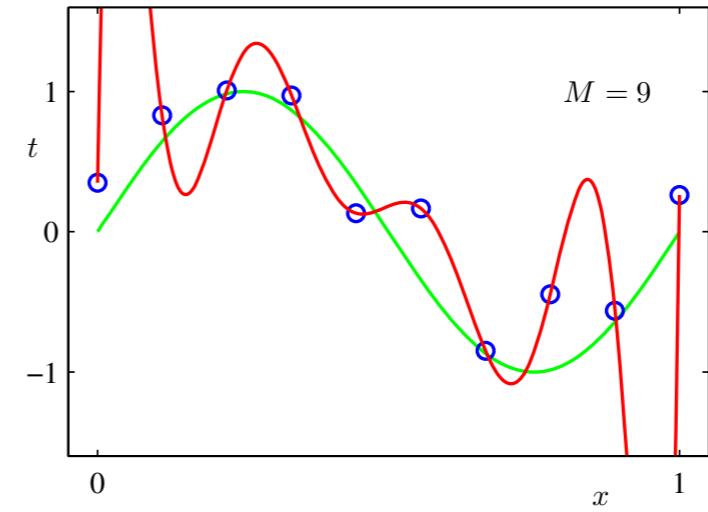
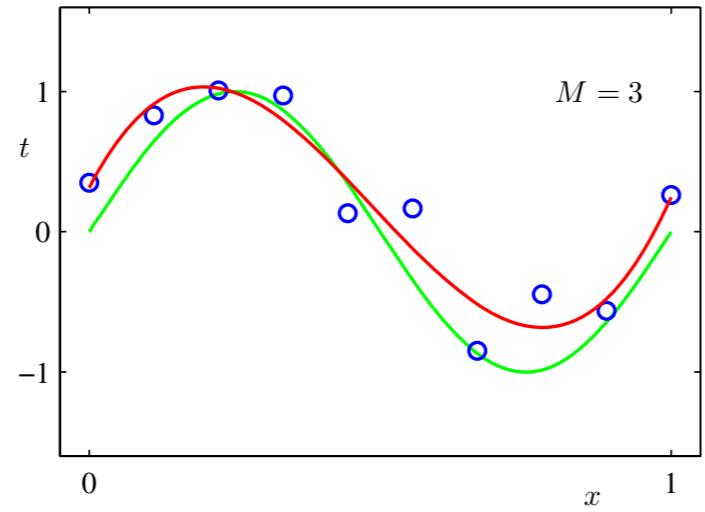
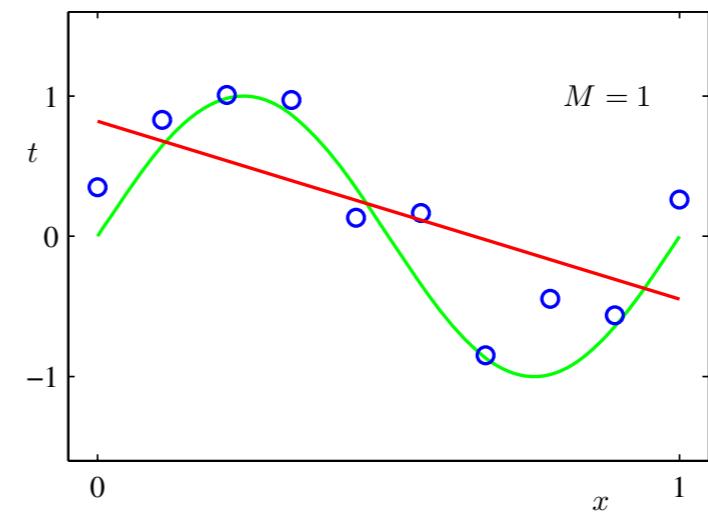
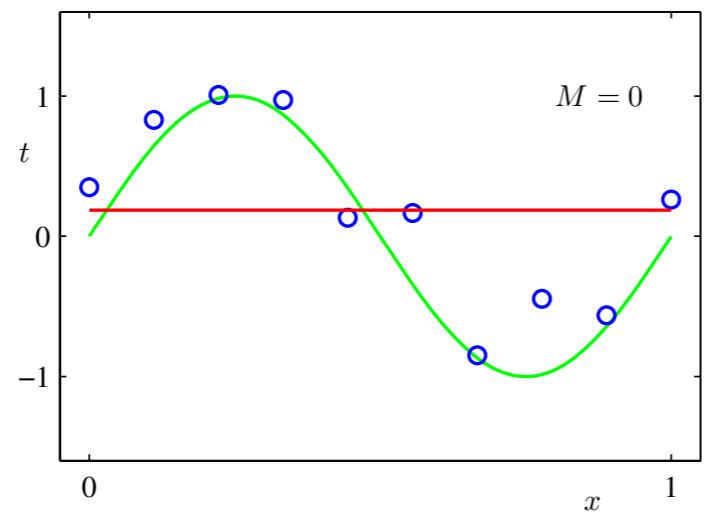


Problems in High Dimensions

- Data Sparseness
 - Although we have a large feature space (lots of dimensions to the data), we only observe a small number of non-zero features in any instance
 - This was the case for texts (in particular with the bag-of-words model)

Problems in High Dimensions

- Overfitting
 - Given n train data points, we can come up with an n -dimensional (n -th order) polynomial that passes through all those data points.
 - But it is very unlikely that it will fit well for the test data points



Problems in High Dimensions

- Time consuming (time complexity is large)
 - Consider computing cosine similarity between two n dimensional vectors, when n increases.
- Memory issues (space complexity is large)
 - Storing high dimensional dense vectors can be problematic when
 - the dimensionality of the vectors is large
 - there are lots of vectors (instances) to store

Solution

- Dimensionality Reduction
 - Try to project the original vectors to a lower dimensional space L
- What constraints do we have
 - Try to minimise the error due to the projection
 - If X and Y are neighbours in the original space, then they must also be neighbours in the projected space
 - Try to retain salient/important/principal dimensions as much as possible and remove the non-salient/unimportant/auxiliary dimensions as much as possible

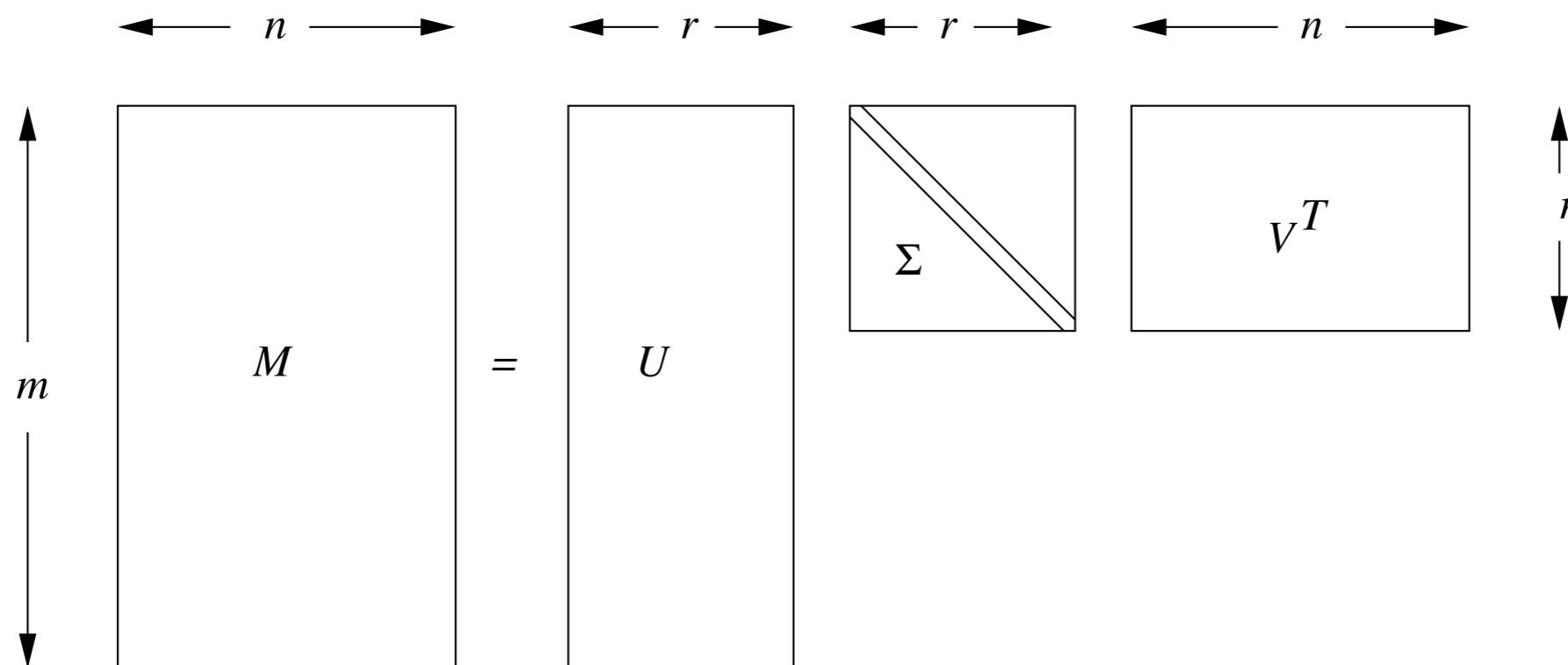
Eigenvalue Decomposition

- Linear Algebra Revision
 - Eigenvalues and Eigenvectors of a Square Matrix
 - $Ax = \lambda x$
 - x is the eigenvector of A corresponding to the eigenvalue λ
- Compute the eigenvalues and eigenvectors of the following matrix

$$\begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}$$

Singular Value Decomposition

- Eigenvalue decomposition can be performed only for square matrices.
- Singular Value Decomposition (SVD) is a operation that can be applied to any matrix
 - $M = U\Sigma V^T$
 - U and V are unitary (perpendicular) matrices, Σ is a diagonal matrix (singular values of M are diagonal elements of Σ).
 - Columns of U and V are perpendicular. $U^T U = I$ and $V^T V = I$.



SVD?



Snayperskaya Vintovka sistem'y Dragunova obraz'tsa
(Dragunov Sniper Rifle or SVD)

Example

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} .14 & 0 \\ .42 & 0 \\ .56 & 0 \\ .70 & 0 \\ 0 & .60 \\ 0 & .75 \\ 0 & .30 \end{bmatrix} \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix} \begin{bmatrix} .58 & .58 & .58 & 0 & 0 \\ 0 & 0 & .71 & .71 & .71 \end{bmatrix}$$

M U Σ V^T

To perform SVD in python (scipy) use `scipy.linalg.svd`
<http://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.svd.html>

Dimensionality Reduction with SVD

- Procedure
 - Perform SVD on M . Retain the top- k (largest) singular values in Σ and set the remainder to zero.
 - Let us denote the diagonal matrix produced by the previous step by Σ_k
 - The k -dimensional approximation (projection) M_k of M is then given by
 - $M_k = U \Sigma_k V^T$

Reason

- The k -dimensional matrix M_k that minimises the Frobenius norm $\|M - M_k\|$ is given by the matrix M_k computed as described in the previous slide
- Frobenius norm
 - Extension of the vector L2 norm to matrices
 - Frobenius norm of a matrix M is given by

$$\sqrt{\sum_{ij} M_{ij}^2}$$

Proof

- By performing SVD on M, let

- $M = PQR^T$

$$m_{ij} = \sum_k \sum_\ell p_{ik} q_{k\ell} r_{\ell j} \quad \|M\|^2 = \sum_i \sum_j (m_{ij})^2 = \sum_i \sum_j \left(\sum_k \sum_\ell p_{ik} q_{k\ell} r_{\ell j} \right)^2$$

$$\left(\sum_k \sum_\ell p_{ik} q_{k\ell} r_{\ell j} \right)^2 = \sum_k \sum_\ell \sum_m \sum_n p_{ik} q_{k\ell} r_{\ell j} p_{in} q_{nm} r_{mj}$$

$$\|M\|^2 = \sum_i \sum_j \sum_k \sum_\ell \sum_n \sum_m p_{ik} q_{k\ell} r_{\ell j} p_{in} q_{nm} r_{mj}$$

$$\|M\|^2 = \sum_i \sum_j \sum_k \sum_n p_{ik} q_{kk} r_{kj} p_{in} q_{nn} r_{nj}$$

$$\|M\|^2 = \sum_j \sum_k q_{kk} r_{kj} q_{kk} r_{kj}$$

Much easier proof exists if you use
the trace of a matrix and its properties

$$\|M\|^2 = \sum_k (q_{kk})^2$$

Proof using Trace

$$\mathbf{M} = \mathbf{PQR}^\top$$

$$\|\mathbf{M}\|_2^2 = \text{tr}(\mathbf{M}^\top \mathbf{M})$$

$$\|\mathbf{M}\|_2^2 = \text{tr}(\mathbf{RQP}^\top \mathbf{PQR}^\top)$$

$$\|\mathbf{M}\|_2^2 = \text{tr}(\mathbf{RQQR}^\top)$$

$$\|\mathbf{M}\|_2^2 = \text{tr}(\mathbf{RQ}^2 \mathbf{R}^\top)$$

$$\|\mathbf{M}\|_2^2 = \text{tr}(\mathbf{R}^\top \mathbf{RQ}^2)$$

$$\|\mathbf{M}\|_2^2 = \text{tr}(\mathbf{Q}^2)$$

SVD and Approximation Error

$$\mathbf{M} = \mathbf{P} \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_k & \\ & & & \sigma_{k+1} \\ & & & & \ddots \\ & & & & & \sigma_n \end{bmatrix} \mathbf{R}^\top$$

If \mathbf{M}_k is the matrix with $(k+1)$ and above singular values set to zero
(k -th rank approximation)

$$\mathbf{M}_k = \mathbf{P} \begin{bmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_k & & \\ & & & 0 & \\ & & & & \ddots \\ & & & & & 0 \end{bmatrix} \mathbf{R}^\top$$

SVD and Approximation error

- Then, the approximation error, $\|M - M_k\|^2$ becomes

$$\|M - M_k\|^2 = \text{tr} \left(P \begin{bmatrix} 0 & & & \\ & \ddots & & \\ & & 0 & \\ & & & \sigma_{k+1} \\ & & & & \ddots \\ & & & & & \sigma_n \end{bmatrix} R^\top \right)$$

$$\|M - M_k\|^2 = \sigma_{k+1}^2 + \dots + \sigma_n^2$$

If we want to minimize this error, then we must select the largest singular values for the first $1 \dots k$ positions!

Applications of SVD

- Latent Semantic Analysis
 - words vs. document matrix
 - Find similar words (query expansion)
 - Find similar documents (similarity search)
- Recommendation Systems
 - users vs. items/products
 - Recommend similar products to users

Two uses of SVD

- SVD for dimensionality reduction
 - Compute $M = U\Sigma V^T$
 - Get the largest k singular values from Σ to construct a diagonal matrix Σ_k
 - Get the corresponding left singular vectors from U to construct a matrix U_k
 - Reduce the number of columns of M to k to construct the matrix M_k
 - $M_k = U_k \Sigma_k$

Two uses of SVD

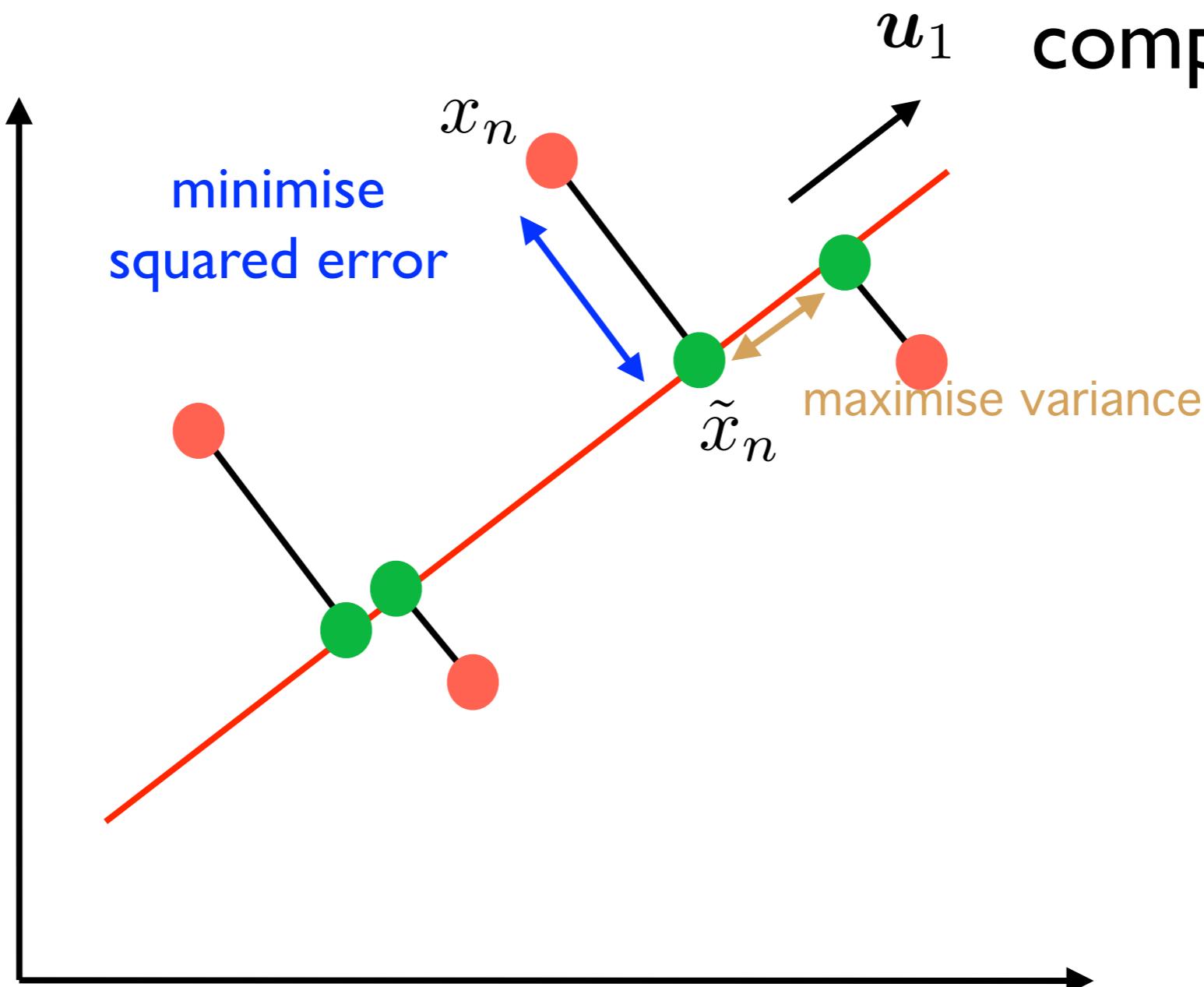
- SVD to increase the density of a matrix
 - Compute $M = U\Sigma V^T$
 - Get the largest k singular values from Σ to construct a diagonal matrix Σ_k
 - Get the corresponding left singular vectors from U to construct a matrix U_k
 - Get the corresponding right singular vectors from V to construct a matrix V_k
 - Reproduce a dense version of M , M_k
 - $M_k = U_k \Sigma_k V_k^T$
 - Lesser number of non-zero values in M_k
 - However, we end up with negative values in M_k even though M is a matrix with all non-negative values!

Principal Component Analysis

- We would like to **project** our **high dimensional** data points to a **low dimensional** space by **preserving the geometric properties** in the original space as much as possible
- Two ways to do this:
 - Maximise the variance of the projected data
 - Linear projection that minimises the average projection cost (e.g. sum of squared Euclidean distance between original and projected points)
- PCA is also known as the Karhunen-Loève Transform

Idea

First principal component



Maximum Variance Formulation

- Problem
 - Given D dimensional N data points $\{x_n\}$, where $n=1, \dots, N$, we must project those into an $M < D$ dimensional space
 - M is given
- Let us consider the case $M=1$ (one-dimensional projection)
- The projection direction is given by the unit vector u_1
 - $u_1^T u_1 = 1$
 - $\tilde{x}_n = u_1^T x_n$

Maximum Variance Formulation

- Mean of the data points

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$$

- Variance of the projected data

$$\frac{1}{N} \sum_{n=1}^N (\mathbf{u}_1^\top \mathbf{x}_n - \mathbf{u}_1^\top \bar{\mathbf{x}})^2 = \mathbf{u}_1^\top \mathbf{S} \mathbf{u}_1$$

*An unbiased estimator of variance uses $(N-1)$ instead of N .
But this does not matter because we are only interested in the maximisation of the variance.

- \mathbf{S} is the covariance matrix given by

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^\top$$

Maximum Variance Formulation

- We must maximize the variance subjected to the normalization constraint on \mathbf{u}_1

$$L(\mathbf{u}_1, \lambda_1) = \mathbf{u}_1^\top \mathbf{S} \mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^\top \mathbf{u}_1)$$

Lagrange multiplier method

$$\frac{\partial L(\mathbf{u}_1, \lambda_1)}{\partial \mathbf{u}_1} = 0 \quad \rightarrow \quad \mathbf{S} \mathbf{u}_1 = \lambda_1 \mathbf{u}_1$$

$$\mathbf{u}_1^\top \mathbf{S} \mathbf{u}_1 = \lambda_1$$

This is variance!

\mathbf{u}_1 is the eigenvector of \mathbf{S} that corresponds to the largest eigenvalue of \mathbf{S}

PCA Algorithm

- INPUT

- D dimensional N data points $\{x_n\}$, where $n=1,...,N$
- Dimensionality M

- Procedure

- Compute the covariance matrix S for the dataset
- Compute the first M eigenvectors of S
- return the computed eigenvectors

A Word on Complexity

- Eigenvalue decomposition of a $D \times D$ matrix is $O(D^3)$
- However, we only need the largest M eigenvectors of S
- This can be computed efficiently using truncated methods such as the power-iteration method in $O(MD^2)$
- Reference
 - Golub & Van Loan, Matrix Computations, John Hopkins University Press, 1996.

Computational Remarks (1/2)

- The following methods for computing PCA are equivalent (i.e. gives the same principal components)
 1. Compute the sum of squared Euclidean distance between original and projected points and **minimise** it.
 2. Compute the pairwise squared Euclidean distance between projected points and **maximise** it.
 3. Compute the variance of the projected points on the projection line and **maximise** it.

Computational Remarks (2/2)

- Let us show that (2) and (3) are equivalent

$$\begin{aligned}(3) &= \sum_{i=1}^N (\mathbf{x}_i^\top \mathbf{u} - \bar{\mathbf{x}}^\top \mathbf{u})^2 \\&= \sum_{i=1}^N \left(\left(\mathbf{x}_i - \frac{1}{N} \sum_{j=1}^N \mathbf{x}_j \right)^\top \mathbf{u} \right)^2 \\&= \sum_{i=1}^N \sum_{j=1}^N ((\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{u})^2 \\&= \sum_{i=1}^N \sum_{j=1}^N (\mathbf{x}_i^\top \mathbf{u} - \mathbf{x}_j^\top \mathbf{u})^2 \\&= (2)\end{aligned}$$

Using (2) for computing PCA is usually a bad idea when the dataset is large because the pairwise combinations grow quadratically with the number of data points in the dataset. But could be helpful for small examples because we do not have to compute the mean.

References

- Mining of Massive Datasets
 - <http://infolab.stanford.edu/~ullman/mmds.html#original>
 - Chapter 11 on Dimensionality Reduction (SVD)
- Pattern Recognition and Machine Learning
 - Section 1.1 (overfitting)
 - Section 1.4 (curse of dimensionality)
 - Page 561 onwards: PCA

Text Mining

Department of Computer Science
University of Liverpool

March 2020

Overview

1 Introduction

- Some examples
- Definition and Challenges
- Steps in text mining

2 Preprocessing

- Tokenisation
- Stemming
- Methods for Stopword Removal - Stopword List
- Sentence Segmentation

3 Part-Of-Speech (POS) Tagging

- Rule-based Methods

4 References

Simple Question: Why do dogs howl at the moon?

https://www.google.com/search?q=why+do+dogs+howl+at+the+moon&lz=1C1CHBF_en-GBGB823GB823&oq=w

Google why do dogs howl at the moon

All Images Maps Videos News More Settings Tools

About 7,230,000 results (0.62 seconds)

They **do** tend to **howl** more as darkness is falling, or morning is coming. That might be why people think they're **howling at the moon**. They also tip their heads back, adding to that impression. **Dogs** still have some wolf behavior, so some **howl**.

Why do dogs or wolves howl at the moon when it ... - UCSB Science Line
sciencline.ucsb.edu/getkey.php?key=340

About this result Feedback

People also ask

Why do dogs and wolves howl at the moon? ▾

When a dog howls does it mean death? ▾

What does it mean when a dog is howling? ▾

Why does my dog bark at the moon? ▾

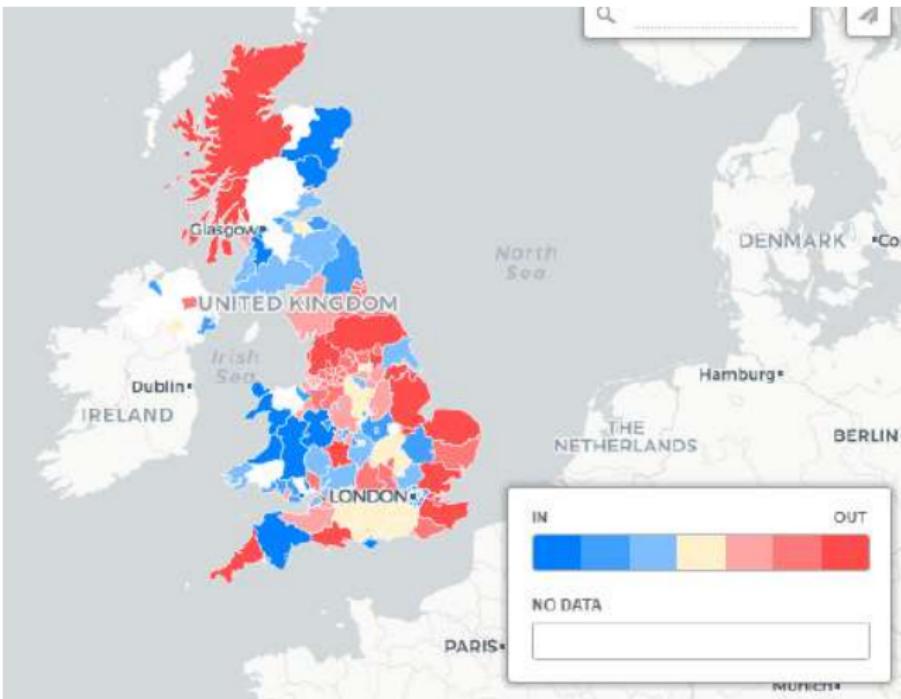
Feedback

Why Do Dogs Howl At The Moon? - Dogtime

<https://dogtime.com/dog-health/dog...22207-why-do-dogs-howl-at-the-moon> ▾

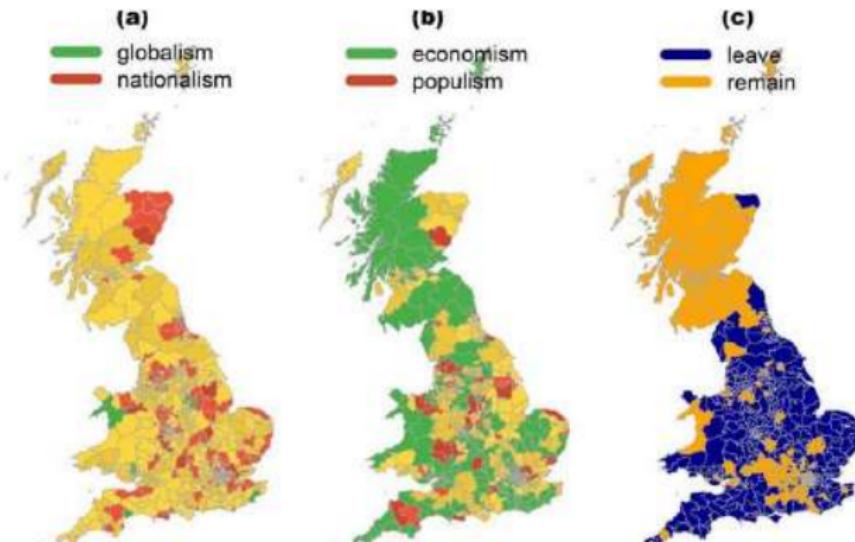
Wolves are the ancestors of our indoor pups, and they're known for howling at the moon. ... Wolves are nocturnal, and they need to communicate, so they howl at night. They also throw their heads back

Text Mining Around Us - Sentiment Analysis



source: <https://www.jellyfish.co.uk/news-and-views/update-eu-referendum-campaigns-seem-to-be-causing-little-impact>

Text Mining Around Us - Opinion Mining



Color-coded heat map of UK parliamentary constituencies (see legend). In graphics (a) and (b), green is used for constituencies showing majority economic and globalist sentiment, and red is used for constituencies showing majority populist and nationalist sentiment. Yellow is the result of adding green to red, with these constituencies somewhere in the middle of the scales. Graph(c) shows voting patterns in the referendum. Credit: Dr. Marco Bastos and Dr. Dan Mercea

source: <https://phys.org/news/2018-04-brexit-debate-twitter-driven-economic.html>

Text Mining Around Us - Movie Recommendation Systems

Google movie recommendations 2018

All News Images Shopping Videos More Settings Tools

About 547,000,000 results (0.52 seconds)

According to IMDb

View 2+ more

The Guilty Mission: Impossible - Fallout Searching A Star Is Born Spider-Man: Into the Spider-Verse Blindspotting You Were Never Really Here

- The Guilty (2018) R | 85 min | Crime, Drama, Thriller ...
- Mission: Impossible - Fallout (2018) PG-13 | 147 min | Action, Adventure, Thriller ...
- Searching (III) (2018) ...
- A Star Is Born (2018) ...
- Spider-Man: Into the Spider-Verse (2018) ...
- Blindsight (2018) ...
- You Were Never Really Here (2017) ...
- A Quiet Place (2018) ...

More items... • 16 Jan 2018

Top 50 Best Films of 2018 - IMDb
<https://www.imdb.com/list/ls021105452/>

About this result Feedback

Navigation icons: back, forward, search, etc.

Text Mining Around Us - Document Summarization

All News Images Books Videos More Settings Tools

About 465,000,000 results (0.41 seconds)

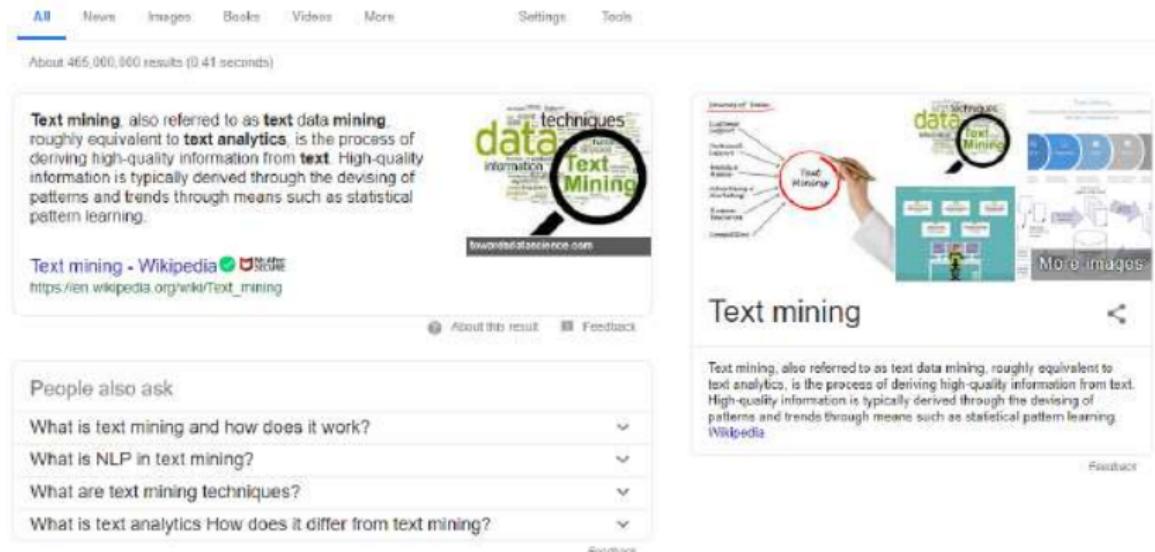
Text mining, also referred to as **text data mining**, roughly equivalent to **text analytics**, is the process of deriving high-quality information from **text**. High-quality information is typically derived through the devising of patterns and trends through means such as statistical pattern learning.

Text mining - Wikipedia  
https://en.wikipedia.org/wiki/Text_mining

People also ask

- What is text mining and how does it work?
- What is NLP in text mining?
- What are text mining techniques?
- What is text analytics How does it differ from text mining?

Feedback



The screenshot shows a search results page for "Text mining". At the top, there are navigation links for All, News, Images, Books, Videos, More, Settings, and Tools. Below that, a search bar shows the query "Text mining". The main content area displays the first result, which is a snippet from Wikipedia. The snippet defines text mining as a process of deriving high-quality information from text using statistical pattern learning. It includes a link to the full Wikipedia article. To the right of the snippet, there is a large image featuring a hand holding a magnifying glass over a cloud of words like "data", "information", "Text Mining", and "techniques". Below the image, there are several smaller thumbnail previews of other search results. At the bottom of the snippet, there are "About this result" and "Feedback" buttons. On the left side of the snippet, there is a "People also ask" section with four expandable questions: "What is text mining and how does it work?", "What is NLP in text mining?", "What are text mining techniques?", and "What is text analytics How does it differ from text mining?". Each question has a "Feedback" button below it. The overall layout is typical of a Google search result page.

Text Mining - Definition and Challenges

- Text mining

- process of extracting interesting and non-trivial patterns or knowledge from unstructured text documents [Tan et al., 1999].
- *a.k.a* text data mining [Hearst, 1997],
- knowledge discovery from textual databases [Feldman and Dagan, 1995]
- text analytics - application to solve business problems

Text Mining - Challenges

- Unorganized form of data
 - semi-structured or unstructured
- Deriving semantics from content
 - ambiguities at different levels - lexical, syntactic, semantic and pragmatic
 - Text has multiple interpretations
Teacher Strikes Idle Kids
Violinist linked to JAL crash blossoms
 - Word sense ambiguity
Red Tape Holds Up New Bridges
- Non-standard English
 - language in Tweets
 - SOO PROUD of what U accomp.

Text Mining - Challenges

- New Words

- 850 new words added dictionary at Merriam-Webster.com in 2018
- Cryptocurrency
- Chiweenie - a cross between a Chihuahua and a dachshund
- Dumpster fire - a disastrous event

- Idioms

- dark horse; get cold feet

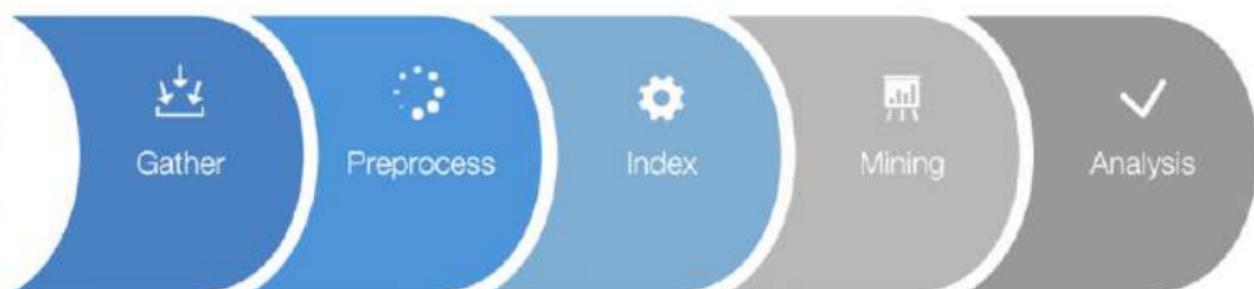
- Combining information from multi-lingual texts

- Integrate domain knowledge

Steps in Text Mining

Text Mining

Text mining involves a series of activities to be performed in order to efficiently mine the information. These activities are:



Data assemble from
difference resources

Data preparation
and transformation

Quick access and
search stored data

Algorithm, inference and
information extraction

User analysis,
Navigation

source: <http://openminted.eu/text-mining-101/>

Text Mining - Preprocessing Steps

- Tokenisation
- Stemming
- Stopword Removal
- Sentence Segmentation

Tokenisation

- Process of splitting text into words
- What is a word?

string of contiguous alphanumeric characters with space on either side;
may include hyphens and apostrophes, but no other punctuation
marks [Kučera and Francis, 1967].

- Useful clue - space or tab (English)

Tokenisation - Problems

- Periods
 - usually helps if we remove them
 - but useful to retain in certain cases such as \$22.50; Ed.,
- hyphenation
 - useful to retain in some cases e.g., state-of-the-art
 - better to remove in other cases e.g., gold-import ban, 50-year-old
- Single apostrophes
 - useful to remove them e.g., *isn't*, *didn't*
- space may not be a useful clue all the time
- sometimes we want to use words separated by space as 'single' word
- For example:
 - San Francisco
 - University of Liverpool
 - Danushka Bollegala

Regular Expressions for Tokenisation

- Regular Expressions Cheatsheet

REGEX	NOTE	EXAMPLE	EXPLANATION
\s	white space	\d\s\d	digit space digit
\S	not white space	\d\S\d	digit non-whitespace digit
\d	digit	\d\d\d-\d\d-\d\d\d	SSN
\D	not digit	\D\D\D	three non-digits
\w	word character (letter, number, or _)	\w\w\w	three word chars
\W	not a word character	\W\W\W	three non-word chars
[...]	any included character	[a-z0-9#]	any char that is a thru z, 0 thru 9, or #
[^...]	no included character	[^xyz]	any char but x, y, or z
*	zero or more	\w*	zero or more words chars
+	one or more	\d+	integer
?	zero or one	\d\d\d-?\d\d-?\d\d\d	SSN with dashes being optional
	or	\w \d	word or digit character

Regular Expressions for Tokenisation

```
1 raw = """'When I'M a Duchess,' she said to herself, (not in a very hopeful tone  
2         though), 'I won't have any pepper in my kitchen AT ALL. Soup does very  
3         well without--Maybe it's always pepper that makes people hot-tempered,'..."""  
4  
5 import re  
6 print(re.split(r'\t', raw))  
7 ["'When", "I'M", 'a', "Duchess,", "'she", 'said', 'to', 'herself,', '(not', 'in', 'a',  
8 'very', 'hopeful', 'tone\n\t\t', "','though', "'I", "won't", 'have', 'any',  
9 'pepper', 'in', 'my', 'kitchen', 'AT', 'ALL', 'Soup', 'does', 'very\n\t\t', ',',  
10 'well', 'without--Maybe', "it's", 'always', 'pepper', 'that', 'makes', 'people',  
11 'hot-tempered,..."]  
12  
13 print(re.split(r'[ \t\n]+', raw))  
14 ["'When", "I'M", 'a', "Duchess,", "'she", 'said', 'to', 'herself,', '(not', 'in', 'a',  
15 'very', 'hopeful', 'tone', 'though', ',', "'I", "won't", 'have', 'any', 'pepper', 'in',  
16 'my', 'kitchen', 'AT', 'ALL', 'Soup', 'does', 'very', 'well', 'without--Maybe',  
17 "it's", 'always', 'pepper', 'that', 'makes', 'people', "hot-tempered,..."]  
18  
19 print(re.findall(r"\w+ (?:[-']\w+)*|[.-().+|\$]\w*", raw))  
20 [("'", 'When', 'I', "'", 'M', 'a', 'Duchess', "'", "'she", 'said', 'to',  
21 'herself', ',', '(', 'not', 'in', 'a', 'very', 'hopeful', 'tone', 'though',  
22 ')', ',', "'I", "won'", "'t", 'have', 'any', 'pepper', 'in', 'my',  
23 'kitchen', 'AT', 'ALL', 'Soup', 'does', 'very', 'well', 'without', '--',  
24 'Maybe', "it", "'", 's', 'always', 'pepper', 'that', 'makes', 'people',  
25 'hot', '--', 'tempered', ',', "'", '...']
```

Stanford Parser for Tokenisation

```
1 raw = """'When I'M a Duchess,' she said to herself, (not in a very hopeful tone  
2     though), 'I won't have any pepper in my kitchen AT ALL. Soup does very  
3     well without--Maybe it's always pepper that makes people hot-tempered, '..."""  
4  
5 path_to_parser_jar = 'lib/stanford-parser.jar'  
6 path_to_models_jar = 'lib/stanford-parser-3.5.1-models.jar'  
7  
8 # POS Tagger  
9 from nltk.tokenize.stanford import StanfordTokenizer  
10 tokenizer = StanfordTokenizer(path_to_parser_jar)  
11  
12 tokenized_text = tokenizer.tokenize(raw)  
13 print tokenized_text  
14  
15 [u'', u'When', u'I', u'M', u'a', u'Duchess', u',', u'"', u'she', u'said', u'to',  
16 u'herself', u',', u'-LRB-', u'not', u'in', u'a', u'very', u'hopeful', u'tone', u'though',  
17 u'-RRB-', u',', u'", u'I', u'wo', u'n't', u'have', u'any', u'pepper', u'in', u'my',  
18 u'kitchen', u'AT', u'ALL', u'.', u'Soup', u'does', u'very', u'well', u'without', u'--',  
19 u'Maybe', u'it', u's', u'always', u'pepper', u'that', u'makes', u'people', u'hot-tempered',  
20 u',', u'"', u'...']
```

Tokenisation

- Tokenisation turns out to be more difficult than one expects
- No single solution works well
- Decide what counts as a token depending on the application domain

SPACY (<https://spacy.io/>)

- SPACY - a relatively new package for “Industrial strength NLP in Python”.
- Developed by Matt Honnibal at Explosion AI
- Designed with applied data scientist in mind
- SPACY supports:
 - Tokenisation
 - Lemmatisation
 - Part-of-speech tagging
 - Entity recognition
 - Dependency parsing
 - Sentence recognition
 - Word-to-vector transformations

SPACY - Feature Comparison

	SPACY	SYNTAXNET	NLTK	CORENLP
Programming language	Python	C++	Python	Java
Neural network models	✓	✓	✗	✓
Integrated word vectors	✓	✗	✗	✗
Multi-language support	✓	✓	✓	✓
Tokenization	✓	✓	✓	✓
Part-of-speech tagging	✓	✓	✓	✓
Sentence segmentation	✓	✓	✓	✓
Dependency parsing	✓	✓	✗	✓
Entity recognition	✓	✗	✓	✓
Coreference resolution	✗	✗	✗	✓

source: <https://spacy.io/usage/facts-figures>

SPACY - Benchmarks

SYSTEM	YEAR	LANGUAGE	ACCURACY	SPEED (WPS)
spaCy v2.x	2017	Python / Cython	92.6	n/a <small>?</small>
spaCy v1.x	2015	Python / Cython	91.8	13,963
ClearNLP	2015	Java	91.7	10,271
CoreNLP	2015	Java	89.6	8,602
MATE	2015	Java	92.5	550
Turbo	2015	C++	92.4	349

source: <https://spacy.io/usage/facts-figures>

SPACY - Detailed Speed Comparison

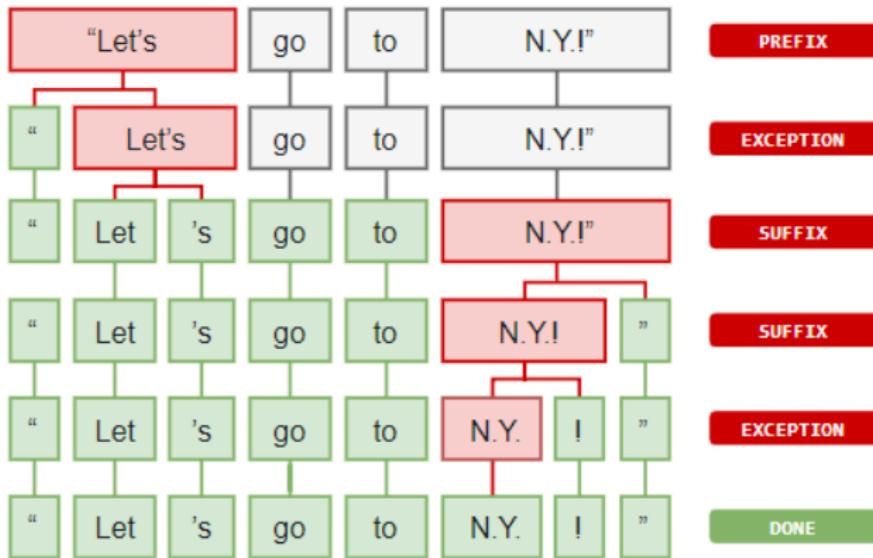
SYSTEM	ABSOLUTE (MS PER DOC)			RELATIVE (TO SPACY)		
	TOKENIZE	TAG	PARSE	TOKENIZE	TAG	PARSE
spaCy	0.2ms	1ms	19ms	1x	1x	1x
CoreNLP	0.18ms	10ms	49ms	0.9x	10x	2.6x
ZPar	1ms	8ms	850ms	5x	8x	44.7x
NLTK	4ms	443ms	n/a	20x	443x	n/a

source: <https://spacy.io/usage/facts-figures>

Tokenization in SPACY

- Tokenizes text into words, punctuations and so on.
- Applies rules specific to each language
- Step 1: Split raw text based on whitespace characters (`text.split(' ')`)
- Step 2: Processes each substring from left to right and performs two checks:
 - Does the substring match a tokenizer exception rule
 - e.g., “don’t” ==> no whitespace ==> but split into two tokens “do” and “nt
 - “U.K.” ==> remain as one token

Tokenization in SPACY



source: <https://spacy.io/usage/spacy-101>

Tokenization in SPACY

The screenshot shows a Jupyter Notebook cell with the following content:

```
import spacy

nlp = spacy.load('en_core_web_sm')
doc = nlp(u'Apple is looking at buying U.K. startup for $1 billion')

for token in doc:
    print(token.text, token.lemma_, token.pos_, token.tag_, token.dep_,
          token.shape_, token.is_alpha, token.is_stop)
```

At the bottom left of the cell is a blue "RUN" button.

source: <https://spacy.io/usage/spacy-101>

Tokenization in SPACY

TEXT	LEMMA	POS	TAG	DEP	SHAPE	ALPHA	STOP
Apple	apple	PROPN	NNP	nsubj	Xxxxx	True	False
is	be	VERB	VBD	aux	xx	True	True
looking	look	VERB	VBG	ROOT	xxxx	True	False
at	at	ADP	IN	prep	xx	True	True
buying	buy	VERB	VBG	pcomp	xxxx	True	False
U.K.	u.k.	PROPN	NNP	compound	X.X.	False	False
startup	startup	NOUN	NN	dobj	xxxx	True	False
for	for	ADP	IN	prep	xxx	True	True
\$	\$	SYM	\$	quantmod	\$	False	False
1	1	NUM	CD	compound	d	False	False
billion	billion	NUM	CD	pobj	xxxx	True	False

source: <https://spacy.io/usage/spacy-101>

Stemming

- Removal of inflectional ending from words (strip off any affixes)
 - connections, connecting, connect, connected → connect
- Problems
 - Can conflate semantically different words
 - *Gallery* and *gall* may both be stemmed to *gall*
- Lemmatization: a further step to ensure that the resulting form is a word present in a dictionary

Regular Expressions for Stemming

```
1 import re
2 print re.findall(r'^(.*)ing|ly|ed|ions|ies|ive|es|s|ment$', 'processing')
3 [('process', 'ing')]
4
5 import re
6 print re.findall(r'^(.*)ing|ly|ed|ions|ies|ive|es|s|ment$', 'processes')
7 [('processe', 's')]
8
```

- note that the star operator is “greedy”
- the `.*` part of expression tries to consume as much as the input as possible
- for non-greedy version of the star operator = `*?`

```
9 import re
10 print re.findall(r'^(.*)?ing|ly|ed|ions|ies|ive|es|s|ment$', 'processes')
11 [('process', 'es')]
12
13
```

Regular Expressions for Stemming

```
78 import nltk, re
79
80 def stem(word):
81     regexp = r'^(.*)ing|ly|ed|ions|ies|ive|es|s|ment)$'
82     stem, suffix = re.findall(regexp, word)[0]
83     return stem
84
85 raw = """DENNIS: Listen, strange women lying in ponds distributing swords
86         is no basis for a system of government. Supreme executive power derives from
87         a mandate from the masses, not from some farcical aquatic ceremony."""
88
89 tokens = nltk.word_tokenize(raw)
90 print [stem(t) for t in tokens]
91
92 ['DENNIS', ':', 'Listen', ',', 'strange', 'women', 'ly', 'in', 'pond', 'distribut', 'sword',
93 'i', 'no', 'basi', 'for', 'a', 'system', 'of', 'govern', ',', 'Supreme', 'execut', 'power',
94 'deriv', 'from', 'a', 'mandate', 'from', 'the', 'mass', ',', 'not', 'from', 'some',
95 'farcical', 'aquatic', 'ceremony', '.']
```

- Problems

- RE removes 's' from 'ponds', but also from 'is' and 'basis'
- produces some non-words like 'distribut', 'deriv'

NLTK Stemmers

- NLTK provides several off-the-shelf stemmers
- Porter and Lancaster stemmers have their own rules for stripping affixes

```
1 import nltk, re
2
3 raw = """DENNIS: Listen, strange women lying in ponds distributing swords
4      is no basis for a system of government. Supreme executive power derives from
5      a mandate from the masses, not from some farcical aquatic ceremony."""
6
7 porter = nltk.PorterStemmer()
8 lancaster = nltk.LancasterStemmer()
9 tokens = nltk.word_tokenize(raw)
10
11 print [porter.stem(t) for t in tokens]
12 ['denni', ':', 'listen', ',', 'strang', 'wom', 'lie', 'in', 'pond', 'distribut',
13 'sword', 'is', 'no', 'bas', 'for', 'a', 'system', 'of', 'govern', ',', 'suprem',
14 'execut', 'power', 'deriv', 'from', 'a', 'mandat', 'from', 'the', 'mass', ',', 'not',
15 'from', 'some', 'farcic', 'aquat', 'ceremeoni', '.']
16
17 print [lancaster.stem(t) for t in tokens]
18 ['den', ':', 'list', ',', 'strange', 'wom', 'lying', 'in', 'pond', 'distribut', 'sword',
19 'is', 'no', 'bas', 'for', 'a', 'system', 'of', 'govern', ',', 'suprem', 'execut', 'pow',
20 'der', 'from', 'a', 'mand', 'from', 'the', 'mass', ',', 'not', 'from', 'som', 'farc',
21 'aqua', 'ceremeony', '.']
```

Is stemming useful?

- Provides some improvement for IR performance (especially for smaller documents).
- Very useful for some queries, but on an average does not help much.
- Since improvement is very minimal, often IR engines does not use stemming.

Stopword Removal

- Removal of high frequency words
- Most common words such as articles, prepositions, and pronouns etc. does not help in identifying meaning

a	an	and	are	as	at	be	by	for	from
has	he	in	is	it	its	of	on	that	the
to	was	were	will	with					

Figure: A stop list of 25 semantically non-selective words which are common in Reuters-RCV1

Methods for stopword removal - Zipf's law

- frequency of a word is inversely proportional to its rank in the frequency table
- remove most frequent words

Zipf's law

- frequency of a word is inversely proportional to its rank in the frequency table
- i.e., frequency of the word decreases sharply with the increase in rank
- implies a small number of words appear very often and large number rarely occur
- remove most frequent words

Mutual Information

- supervised method that computes mutual information between a given term and a document class
- low mutual information suggests low discrimination power of the term and hence should be removed
- compute $A(t, c)$, expected *mutual information* (MI) of term t and class c .
- Formally, MI is calculated using:

$$I(U; C) = \sum_{e_t \in \{1, 0\}} \sum_{e_c \in \{1, 0\}} P(U = e_t, C = e_c) \log_2 \frac{P(U = e_t, C = e_c)}{P(U = e_t)P(C = e_c)},$$

where U is a random variable and takes values $e_t = 1$ (the document contains term t) and $e_t = 0$ (the document does not contain term t) where C is a random variable and takes values $e_c = 1$ (the document is in class c) and $e_c = 0$ (the document is not in class c)

Mutual Information

- For MLEs of probabilities:

$$I(U; C) = \frac{\frac{N_{11}}{N} \log_2 \frac{NN_{11}}{N_{1.}N_{.1}} + \frac{N_{01}}{N} \log_2 \frac{NN_{01}}{N_{0.}N_{.1}}}{\frac{N_{10}}{N} \log_2 \frac{NN_{10}}{N_{1.}N_{.0}} + \frac{N_{00}}{N} \log_2 \frac{NN_{00}}{N_{0.}N_{.0}}}$$

N s are counts of documents in different categories. Example: class 'poultry' and the term 'export'

	$e_c = e_{\text{poultry}} = 1$	$e_c = e_{\text{poultry}} = 0$
$e_t = e_{\text{export}} = 1$	$N_{11} = 49$	$N_{10} = 27,652$
$e_t = e_{\text{export}} = 0$	$N_{01} = 141$	$N_{00} = 774,106$

Mutual Information

	$e_c = e_{poultry} = 1$	$e_c = e_{poultry} = 0$
$e_t = e_{export} = 1$	$N_{11} = 49$	$N_{10} = 27,652$
$e_t = e_{export} = 0$	$N_{01} = 141$	$N_{00} = 774,106$

$$\begin{aligned} I(U; C) &= \frac{49}{801,948} \log_2 \frac{801,948 \cdot 49}{(49+27,652)(49+141)} \\ &\quad + \frac{141}{801,948} \log_2 \frac{801,948 \cdot 141}{(141+774,106)(49+141)} \\ &\quad + \frac{27,652}{801,948} \log_2 \frac{801,948 \cdot 27,652}{(49+27,652)(27,652+774,106)} \\ &\quad + \frac{774,106}{801,948} \log_2 \frac{801,948 \cdot 774,106}{(141+774,106)(27,652+774,106)} \\ &\approx 0.0001105 \end{aligned}$$

Mutual Information

UK		China		poultry	
london	0.1925	china	0.0997	poultry	0.0013
uk	0.0755	chinese	0.0523	meat	0.0008
british	0.0596	beijing	0.0444	chicken	0.0006
stg	0.0555	yuan	0.0344	agriculture	0.0005
britain	0.0469	shanghai	0.0292	avian	0.0004
plc	0.0357	hong	0.0198	broiler	0.0003
england	0.0238	kong	0.0195	veterinary	0.0003
pence	0.0212	xinhua	0.0155	birds	0.0003
pounds	0.0149	province	0.0117	inspection	0.0003
english	0.0126	taiwan	0.0108	pathogenic	0.0003
coffee		elections		sports	
coffee	0.0111	election	0.0519	soccer	0.0681
bags	0.0042	elections	0.0342	cup	0.0515
growers	0.0025	polls	0.0339	match	0.0441
kg	0.0019	voters	0.0315	matches	0.0408
colombia	0.0018	party	0.0303	played	0.0388
brazil	0.0016	vote	0.0299	league	0.0386
export	0.0014	poll	0.0225	beat	0.0301
exporters	0.0013	candidate	0.0202	game	0.0299
exports	0.0013	campaign	0.0202	games	0.0284
crop	0.0012	democratic	0.0198	team	0.0264

Figure: Features with high information scores for six Reuters-RCV1 classes

Mutual Information

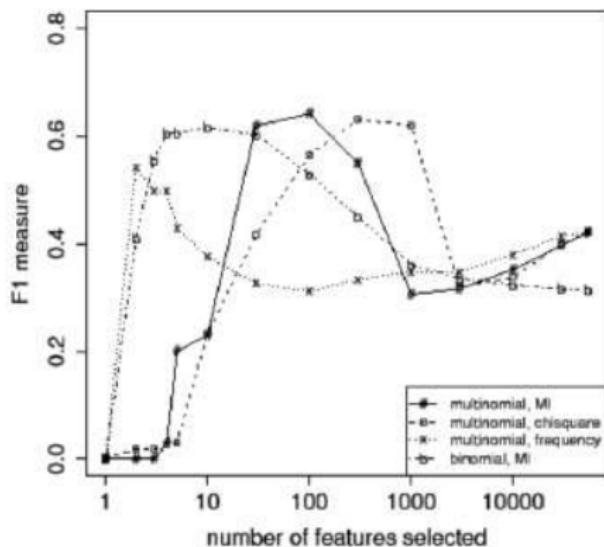


Figure: Effect of feature set size on accuracy

Sentence Segmentation

- Divide text into sentences
- Involves identifying **sentence boundaries** between words in different sentences
- *a.k.a* sentence boundary detection, sentence boundary disambiguation, sentence boundary recognition
- Useful and necessary for various NLP tasks such as
 - sentiment analysis
 - relation extraction
 - question answering systems
 - knowledge extraction

Sentence boundary detection algorithms

- Heuristic methods
- Statistical classification trees [Riley, 1989]
 - probability of a word occurring before or after a boundary, case and length of words
- Neural Networks [Palmer and Hearst, 1997]
 - POS distribution of preceding and following words
- Maximum entropy model [Mikheev 1998]

Sentence Segmentation - Using SPACY

The screenshot shows a Jupyter Notebook cell with the title "Editable code example (experimental)" and version "v2.0.18 · Python 3 · via Binder". The code imports Spacy, loads the "en_core_web_sm" model, processes a multi-sentence string, and prints each sentence. The output below the cell shows the two sentences separated by a new line.

```
import spacy

nlp = spacy.load('en_core_web_sm')
doc = nlp(u"This is a sentence. This is another sentence.")
for sent in doc.sents:
    print(sent.text)
```

This is a sentence.
This is another sentence.

Sentence Segmentation - Using SPACY

The screenshot shows a Jupyter Notebook cell with the title "Editable code example (experimental)" and version "v2.0.18 · Python 3 · via Binder". The code uses spaCy to process a sentence and then adds a custom pipe to handle ellipsis boundaries.

```
import spacy

text = u"This is a sentence... hello...and another sentence."

nlp = spacy.load('en_core_web_sm')
doc = nlp(text)
print('Before:', [sent.text for sent in doc.sents])

def set_custom_boundaries(doc):
    for token in doc[:-1]:
        if token.text == '...':
            doc[token.i+1].is_sent_start = True
    return doc

nlp.add_pipe(set_custom_boundaries, before='parser')
doc = nlp(text)
print('After:', [sent.text for sent in doc.sents])
```

Below the code cell is a "RUN" button. The output section displays the results:

```
Before: ['this is a sentence...', 'hello...', 'and another sentence.']
After: ['this is a sentence...', 'hello...', 'and another sentence.']
```

Part-of-Speech Tagging (POS)

- Task of tagging POS tags (Nouns, Verbs, Adjectives, Adverbs, ...) for words
- POS tags provide lot of information about a word
 - knowing whether a word is **noun** or **verb** gives information about neighbouring words
 - nouns are preceded by determiners; adjectives and verbs by nouns
 - useful for Named entity recognition; Machine Translation; Parsing; Word sense disambiguation
- Given a word, we assume it can belong to only one of the POS tags.
- POS Tagging problem
 - Given a sentence $S = w_1 w_2 \dots w_n$ consisting of n words, determine the corresponding tag sequence $P = P_1 P_2 \dots P_n$

POS Tagging - Challenges

- Words often have more than one POS: e.g., back
 - *The back door* = adjective (JJ)
 - *On my back* = noun (NN)
 - *Win the voters back* = adverb (RB)
 - *Promised to back the bill* = verb (VB)

POS Tagging - Tagset

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	<i>and, but, or</i>	SYM	symbol	+%, &
CD	cardinal number	<i>one, two</i>	TO	"to"	<i>to</i>
DT	determiner	<i>a, the</i>	UH	interjection	<i>ah, oops</i>
EX	existential 'there'	<i>there</i>	VB	verb base form	<i>eat</i>
FW	foreign word	<i>mea culpa</i>	VBD	verb past tense	<i>ate</i>
IN	preposition/sub-conj	<i>of, in, by</i>	VBG	verb gerund	<i>eating</i>
JJ	adjective	<i>yellow</i>	VBN	verb past participle	<i>eaten</i>
JJR	adj., comparative	<i>bigger</i>	VBP	verb non-3sg pres	<i>eat</i>
JJS	adj., superlative	<i>wildest</i>	VBZ	verb 3sg pres	<i>eats</i>
LS	list item marker	<i>1, 2, One</i>	WDT	wh-determiner	<i>which, that</i>
MD	modal	<i>can, should</i>	WP	wh-pronoun	<i>what, who</i>
NN	noun, sing. or mass	<i>llama</i>	WPS	possessive wh-	<i>whose</i>
NNS	noun, plural	<i>llamas</i>	WRB	wh-adverb	<i>how, where</i>
NNP	proper noun, sing.	<i>IBM</i>	\$	dollar sign	\$
NNPS	proper noun, plural	<i>Carolinas</i>	#	pound sign	#
PDT	predeterminer	<i>all, both</i>	"	left quote	' or "
POS	possessive ending	<i>'s</i>	"	right quote	' or "
PRP	personal pronoun	<i>I, you, he</i>	(left parenthesis	[, (, {, <
PRP\$	possessive pronoun	<i>your, one's</i>)	right parenthesis],), }, >
RB	adverb	<i>quickly, never</i>	,	comma	,
RBR	adverb, comparative	<i>faster</i>	.	sentence-final punc	. ! ?
RBS	adverb, superlative	<i>fastest</i>	:	mid-sentence punc	: ; ... --
RP	particle	<i>up, off</i>			

Figure: Penn Treebank POS Tags

POS Tagging - Brown Corpus

- **Brown Corpus** - standard corpus used for POS tagging task
- first text corpus of American English
- published in 1963-1964 by Francis and Kucera
- consists of 1 million words (500 samples of 2000+ words each)
- Brown corpus is PoS tagged with Penn TreeBank tagset.
- ≈ 11% of the word types are ambiguous with regard to POS
- ≈ 40% of the word tokens are ambiguous
- ambiguity for common words. e.g. **that**
 - I know **that** he is honest = preposition (IN)
 - Yes, **that** play was nice = determiner (DT)
 - You can't to **that** far = adverb (RB)

Automatic POS Tagging

- Symbolic
 - Rule-based
 - Transformation-based
- Probabilistic
 - Hidden Markov Models
 - Maximum Entropy Markov Models
 - Conditional Random Fields

Automatic POS Tagging - Brill Tagger

- An example of Transformation-Based Learning
 - Basic idea: do a quick job first (using frequency), then revise it using contextual rules.
 - Painting metaphor from the readings
- Very popular (freely available, works fairly well)
- A supervised method: requires a tagged corpus

Automatic POS Tagging - Brill Tagger

- Start with simple (less accurate) rules...learn better ones from tagged corpus
 - Tag each word initially with most likely POS
 - Examine set of **transformations** to see which improves tagging decisions compared to tagged corpus
 - Re-tag corpus using best transformation
 - Repeat until, e.g., performance doesn't improve
 - Result: tagging procedure (ordered list of transformations) which can be applied to new, untagged text

Automatic POS Tagging: Brill Tagger - Example

- Examples:
 - They are expected to race tomorrow.
 - The race for outer space.
- Tagging algorithm:
 1. Tag all uses of “race” as NN (most likely tag in the Brown corpus)
 - They are expected to race/NN tomorrow
 - the race/NN for outer space
 2. Use a transformation rule to replace the tag NN with VB for all uses of “race” preceded by the tag TO:
 - They are expected to race/VB tomorrow
 - the race/NN for outer space

Automatic POS Tagging: Brill Tagger - Sample Final Rules

Rules:

NN -> NNP if the tag of words i+1...i+2 is 'NNP'
NN -> VB if the tag of the preceding word is 'TO'
NN -> VBD if the tag of the following word is 'DT'
NN -> VBD if the tag of the preceding word is 'NNS'
NN -> JJ if the tag of the preceding word is 'DT', and the tag of the following word is 'NN'
NN -> NNP if the tag of the preceding word is 'NN', and the tag of the following word is ','
NN -> NNP if the tag of words i+1...i+2 is 'NNP'
NN -> IN if the tag of the preceding word is '..'
NNP -> NN if the tag of words i-3...i-1 is 'JJ'
NN -> JJ if the tag of the following word is 'JJ'
NN -> VBP if the tag of the preceding word is 'PRP'
WDT -> IN if the tag of the following word is 'DT'
NN -> JJ if the tag of the preceding word is 'IN', and the tag of the following word is 'NN'
NN -> VBN if the tag of the preceding word is 'VBP'
VBD -> VB if the tag of the preceding word is 'MD'
NN -> JJ if the tag of the preceding word is 'CC', and the tag of the following word is 'NN'



Knowledge discovery in textual databases (kdt).



Texttiling: Segmenting text into multi-paragraph subtopic passages.



Computational analysis of present-day American English.



Text mining: The state of the art and the challenges.

output a

Word Representations



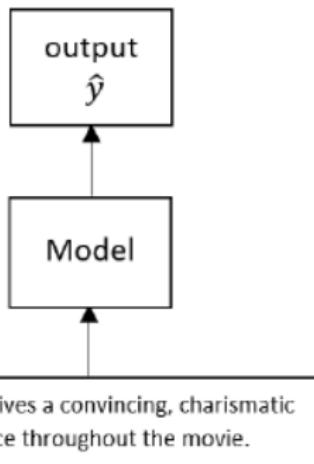
UNIVERSITY OF
LIVERPOOL

Introduction

The actor gives a convincing, charismatic performance throughout the movie.

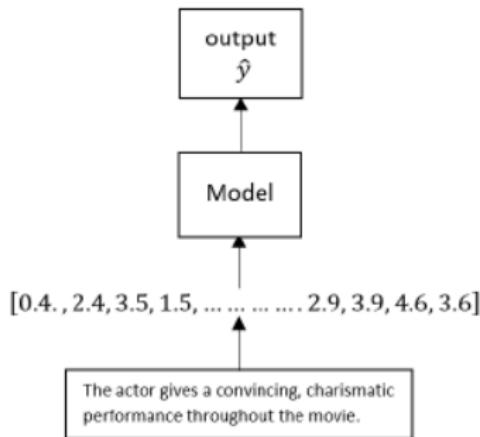
- Imagine we are given a set of words (sentence, document etc.) and we are interested in learning some function using it (e.g., $\hat{y} = \text{sentiment}(\text{words})$)

Introduction



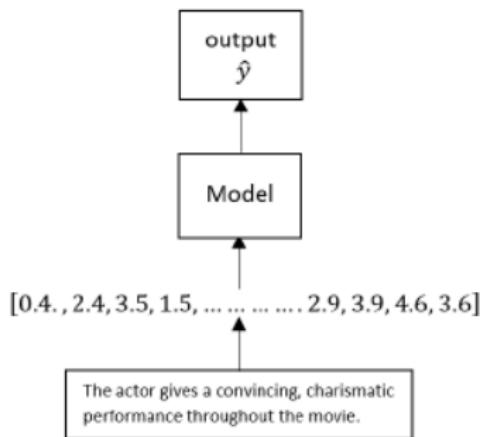
- Imagine we are given a set of words (sentence, document etc.) and we are interested in learning some function using it (e.g., $\hat{y} = \text{sentiment}(\text{words})$)
- Goal would be to apply a model and learn the function $\hat{y} = f(x)$

Introduction



- Imagine we are given a set of words (sentence, document etc.) and we are interested in learning some function using it (e.g., $\hat{y} = \text{sentiment}(\text{words})$)
- Goal would be to apply a model and learn the function $\hat{y} = f(x)$
- convert each input word to a vector x (mathematical representation)

Introduction



- Imagine we are given a set of words (sentence, document etc.) and we are interested in learning some function using it (e.g., $\hat{y} = \text{sentiment}(\text{words})$)
- Goal would be to apply a model and learn the function $\hat{y} = f(x)$
- convert each input word to a vector x (mathematical representation)
- **How can we represent words using vectors?**

One-hot vectors (simple representation)

Corpus

- The actor gives a convincing performance.
- The movie is very powerful.
- The movie was awful.

$\mathcal{V} = [\text{the, actor, gives, a, convincing performance, movie, is, very, powerful, was, awful}]$

- Given a corpus, consider the set \mathcal{V} of all unique words in the input (all sentences or documents)
- \mathcal{V} is called the **vocabulary** of the corpus
- Each word in \mathcal{V} needs a representation.

one-hot vector representation

the : [1 0 0 0 0 0 0 0 0 0 0 0] dimension = \mathbb{R}^{12} , $\mathcal{V} = 12$
actor : [0 1 0 0 0 0 0 0 0 0 0 0]
awful : [0 0 0 0 0 0 0 0 0 0 0 1]

One-hot vector representation - problems

```
cat : [1 0 0 0 0 0 0 0 0 0 0 0]  
dog : [0 0 0 0 1 0 0 0 0 0 0 0]  
car : [0 0 0 0 0 0 0 0 0 0 0 1]
```

- representation size increases significantly with the increase in the size of the corpus. (e.g., 50K for PTB, 13M for Google 1T corpus)

One-hot vector representation - problems

```
cat : [1  0  0  0  0  0  0  0  0  0  0  0]
dog : [0  0  0  0  1  0  0  0  0  0  0  0]
car : [0  0  0  0  0  0  0  0  0  0  0  1]
```

- representation size increases significantly with the increase in the size of the corpus. (e.g., 50K for PTB, 13M for Google 1T corpus)
- results in sparse representation

One-hot vector representation - problems

```
cat : [1 0 0 0 0 0 0 0 0 0 0 0]
dog : [0 0 0 0 1 0 0 0 0 0 0 0]
car : [0 0 0 0 0 0 0 0 0 0 0 1]
```

- representation size increases significantly with the increase in the size of the corpus. (e.g., 50K for PTB, 13M for Google 1T corpus)
- results in sparse representation
- each vector is equidistant from every other vector.

$$\text{euclid_dist}(\text{cat}, \text{dog}) = \sqrt{2}; \text{euclid_dist}(\text{cat}, \text{car}) = \sqrt{2}$$
$$\text{cosine_sim}(\text{cat}, \text{dog}) = 0; \text{cosine_sim}(\text{cat}, \text{car}) = 0$$

- ideally, *cat* and *dog* should be represented closer compared to *cat* and *car*

Why similarity between words matter?

- if we can learn:

“fast” is similar to “rapid”

“tall” is similar to “height”

- useful for **question answering task**

Question: “How tall is Mount Everest?”

Candidate answer: “The official height of Mount Everest is 29029 feet”.

Why similarity between words matter?

Plagiarism Detection

MAINFRAMES

Mainframes are primarily referred to large computers with rapid, advanced processing capabilities that can execute and perform tasks equivalent to many Personal Computers (PCs) machines networked together. It is characterized with high quantity Random Access Memory (RAM), very large secondary storage devices, and high-speed processors to cater for the needs of the computers under its service.

Consisting of advanced components, mainframes have the capability of running multiple large applications required by many and most enterprises and organizations. This is one of its advantages. Mainframes are also suitable to cater for those applications (programs) or files that are of very high

MAINFRAMES

Mainframes usually are referred those computers with fast, advanced processing capabilities that could perform by itself tasks that may require a lot of Personal Computers (PC) Machines. Usually mainframes would have lots of RAMs, very large secondary storage devices, and very fast processors to cater for the needs of those computers under its service.

Due to the advanced components mainframes have, these computers have the capability of running multiple large applications required by most enterprises, which is one of its advantage. Mainframes are also suitable to cater for those applications or files that are of very large demand

Do words have meanings?

Not really.

They just borrow meanings from their neighbours

Distributional Hypothesis



J. R. Firth

*“You shall know a word by
the company it keeps”*

Image credit: www.odit.org

Quiz

- X is a device that is easy to carry around, you can speak using X, watch the Internet. It was manufactured by Apple. What could be X?
 - a dog
 - an airplane
 - an iPhone
 - a banana

But is that really true?

- Don't dictionaries define the meanings of words?
 - Dictionaries define the meanings of words using other words, in a recursive manner.
- Distributional hypothesis provides us with a practical method to learn the meanings of words using text corpora
- Distributional semantic representations have been successfully used in numerous NLP tasks reporting state-of-the-art performances.

Therefore, it must be correct.

Two approaches ...

- **Distributional** Semantic Representations
 - Use the set of words that co-occur with X to represent the meaning of X
 - Sparse and high-dimensional
 - Classical approach for semantic representations
- **Distributed** Semantic Representations
 - Learn representations that can accurately predict the words that appear in the same context as X.
 - Limited dimensionality(10~1000)
 - Low dimensional and dense
 - Deep learning (to be precise representation learning) methods have been used
 - A more recent/modern approach

Two approaches ...

- **Distributional Semantic Representations**
 - Use the set of words that co-occur with X to represent the meaning of X
 - Sparse and high-dimensional
 - Classical approach for semantic representations
- **Distributed Semantic Representations**
 - Learn representations that can accurately predict the words that appear in the same context as X.
 - Limited dimensionality(10~1000)
 - Low dimensional and dense
 - Deep learning (to be precise representation learning) methods have been used
 - A more recent/modern approach

Use co-occurring words to represent meaning

Example sentences for “apple”:

- S_2 : Apples are red
- S_2 : Red apples are delicious
- S_3 : Apples are produced in Washington

Use co-occurring words to represent meaning

Example sentences for “apple”:

- S_2 : Apples are red
- S_2 : Red apples are delicious
- S_3 : Apples are produced in Washington

Representation for “apple”:

apple = [(red,2), (delicious,1),
(Washington,1), (produce,1)]

Use co-occurring words to represent meaning

Example sentences for “apple”:

- S_2 : Apples are red
- S_2 : Red apples are delicious
- S_3 : Apples are produced in Washington

Representation for “apple”:

apple = [(red,2), (delicious,1),
(Washington,1), (produce,1)]

Example sentences for “oranges”:

- S_4 : Oranges are yellow
- S_5 : Oranges are delicious
- S_6 : Oranges are produced in California

Use co-occurring words to represent meaning

Example sentences for “apple”:

- S_2 : Apples are red
- S_2 : Red apples are delicious
- S_3 : Apples are produced in Washington

Representation for “apple”:

apple = [(red,2), (delicious,1),
(Washington,1), (produce,1)]

Example sentences for “oranges”:

- S_4 : Oranges are yellow
- S_5 : Oranges are delicious
- S_6 : Oranges are produced in California

Representation for “oranges”:

oranges = [(yellow,1),
(delicious,1), (California,1),
(produce,1)]

Use co-occurring words to represent meaning

- We have:

`apple = [(red,2), (delicious,1), (Washington,1), (produce,1)]`

`oranges = [(yellow,1), (delicious,1), (California,1), (produce,1)]`

- Similarity between two words can be measured using overlapping features/attributes in their respective semantic representations

Jaccard Coefficient = $| \text{apple AND orange} | / | \text{apple OR orange} |$
 $\text{sim(apple,orange)} = 2/6 = 0.3333$

Co-occurrence Matrix

- We can arrange the semantic representations
- We learn for all the words in a corpus as rows in a co-occurrence matrix
 - rows = semantic representation of words
 - various features/words that co-occur with words

[1.5]	red	1	delicious	1	Washington	produce	1	yellow	0	California	0
apple	2	1	1	1	0	0	0	1	0	0	1
orange	0	1	0	1	1	1	1	1	1	1	0

Issues of large co-occurrences

- How reliable are large co-occurrences?
- Consider Google hits (no. of pages) for,
 - (car, automobile) = 11,300,000
 - (car, apple) = 49,000,000
- apples are more similar to cars than automobiles ???
- We need proper weighting for co-occurrences (in particular when some words are very common)

Co-occurrence Weighting Measures

- combines scores of individual words, $h(x)$, $h(y)$ and co-occurrences between two words $h(x, y)$
 - weighting function = $f(h(x), h(y), h(x, y))$
- pointwise mutual information (PMI) (Church & Hanks, 1989)
 - Do words x and y co-occur more than if they were independent
 - $PMI(x, y) = \log\left(\frac{p(x,y)}{p(x)p(y)}\right) = \log\left(\frac{h(x,y)/N}{(h(x)/N) \times (h(y)/N)}\right)$

Positive Pointwise Mutual Information

- PMI ranges from $-\infty$ to $+\infty$
- negative values are problematic
 - Unreliable without enormous corpora
 - consider w_1 and w_2 whose probability is each 10^{-6}
 - hard to be sure $p(w_1, w_2)$ is significantly different than 10^{-12}
 - replace negative PMI values by 0
- Positive PMI (PPMI) between word1 and word2:
$$PMI(x, y) = \max\left(\log_2 \frac{h(x,y)/N}{(h(x)/N) \times (h(y)/N)}, 0\right)$$

PPMI - Example

Example contexts: 20 words (Brown corpus)

- equal amount of sugar, a sliced lemon, a tablespoonful of **apricot** preserve or jam, a pinch each of clove and nutmeg,
- on board for their enjoyment. Cautiously she sampled her first **pineapple** and another fruit whose taste she likened to that of
- of a recursive type well suited to programming on the **digital** computer. In finding the optimal R-stage policy from that of
- substantially affect commerce, for the purpose of gathering data and **information** necessary for the study authorised in the first section of this

PPMI - Example

Co-occurrence Matrix

	material	computer	data	pinch	result	sugar
apricot	0	0	0	1	0	1
pineapple	0	0	0	1	0	1
digital	0	2	1	0	1	0
information	0	1	6	0	4	0

- two words are considered to be similar if their context vectors are similar

PPMI - Example

Co-occurrence Matrix

	material	computer	data	pinch	result	sugar	$c(x)$
apricot	0	0	0	1	0	1	2
pineapple	0	0	0	1	0	1	2
digital	0	2	1	0	1	0	4
information	0	1	6	0	4	0	11
$c(y)$	0	3	7	2	5	2	

$c = \text{count}$

$$PMI(x, y) = \log\left(\frac{p(x,y)}{p(x)p(y)}\right) = \log\left(\frac{h(x,y)/N}{(h(x)/N) \times (h(y)/N)}\right)$$

$N = \text{sum of all words in all contexts} = 19$

$h(x, y) = h(x=\text{information}, y=\text{data}) = 6; h(x, y)/N = 6/19 = 0.32$
(x is the word, y is the context)

$h(x) = h(x=\text{information}) = 11; h(x)/N = 11/19 = 0.58$

$h(y) = h(y=\text{data}) = 7; h(y)/N = 7/19 = 0.37$

PPMI - Example

$$PMI(x, y) = \log\left(\frac{p(x,y)}{p(x)p(y)}\right) = \log\left(\frac{h(x,y)/N}{(h(x)/N) \times (h(y)/N)}\right)$$

N = sum of all words in all contexts = 19

$h(x, y) = h(x=\text{information}, y=\text{data}) = 6; h(x, y)/N = 6/19 = 0.32$
(x is the word, y is the context)

$h(x) = h(x=\text{information}) = 11; h(x)/N = 11/19 = 0.58$
 $h(y) = h(y=\text{data}) = 7; h(y)/N = 7/19 = 0.37$

Co-occurrence Matrix

	$h(x, y)$					$h(x)$
	computer	data	pinch	result	sugar	
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
$h(y)$	0.16	0.37	0.11	0.26	0.11	

PPMI - Example

	$h(x, y)$					
	computer	data	pinch	result	sugar	$h(x)$
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
$h(y)$	0.16	0.37	0.11	0.26	0.11	

$$PMI(x, y) = \log\left(\frac{p(x, y)}{p(x)p(y)}\right) = \log\left(\frac{h(x, y)/N}{(h(x)/N) \times (h(y)/N)}\right)$$

$$PMI(\text{information}, \text{ data}) = \log_2(0.32/(0.37 \times 0.58)) = 0.57$$

$$h(x, y)$$

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

PPMI - Example - add-2 smoothing

	without smoothing				
	computer	data	pinch	result	sugar
apricot	0	0	1	0	1
pineapple	0	0	1	0	1
digital	2	1	0	1	0
information	1	6	0	4	0

	with add-2 smoothing				
	computer	data	pinch	result	sugar
apricot	2	2	3	2	3
pineapple	2	2	3	2	3
digital	4	3	2	3	2
information	3	8	2	6	2

	$h(x, y)$ with add-2 smoothing					
	computer	data	pinch	result	sugar	$h(x)$
apricot	0.03	0.03	0.05	0.03	0.05	0.20
pineapple	0.03	0.03	0.05	0.03	0.05	0.20
digital	0.07	0.05	0.03	0.10	0.03	0.36
information	0.05	0.14	0.03	0.10	0.03	0.36
$h(y)$	0.19	0.25	0.17	0.22	0.17	

PPMI - Example - add-2 smoothing

$h(x, y)$ without smoothing

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

$h(x, y)$ with add-2 smoothing

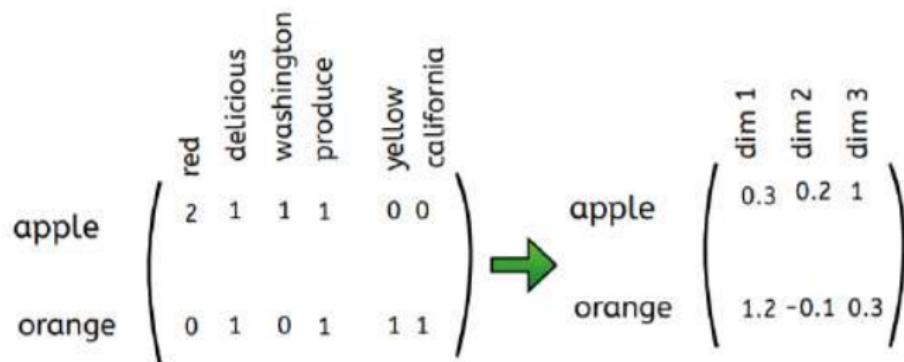
	computer	data	pinch	result	sugar
apricot	0.00	0.00	0.56	0.00	0.56
pineapple	0.00	0.00	0.56	0.00	0.56
digital	0.62	0.00	0.00	0.00	0.00
information	0.00	0.57	0.00	0.37	0.00

Issues of zero co-occurrences

- Some words never co-occur even in very large corpora.
- If words x and y do not co-occur, then
 - x and y might not be related OR
 - it could be that our corpus was too small and we did not observe their co-occurrences.
- If we have co-occurrence-based semantic representations that have many zeros, then we will have many zero similarity scores, which is not good.
- How can we reduce the number of zeros?

Dimensionality reduction / Low-dimensional projection

- We can reduce the number of features (columns) thereby collapsing similar dimensions.
- This process will reduce the number of zeros.



Note that the number of words (rows) does not change. Therefore, we have a representation for all the words that appear in the original co-occurrence matrix.

Dimensionality reduction methods

- Singular Value Decomposition (SVD)
 - $A = UDV^T$, use U or UD as the lower-dimensional projection
- Principal Component Analysis (PCA)
 - See the lecture on dimensionality reduction
- non-negative matrix factorization (NMF)
 - $A = WH$
- There are many libraries that implement the above (and many more)
 - In Python: numpy, scipy, sklearn

Selecting context

- How to select the “context”?
 - sentence-level co-occurrences
 - proximity window (n words before/after)
 - Words that are connected via dependency relations

Two approaches ...

- **Distributional** Semantic Representations
 - Use the set of words that co-occur with X to represent the meaning of X
 - Sparse and high-dimensional
 - Classical approach for semantic representations
- **Distributed** Semantic Representations
 - Learn representations that can accurately predict the words that appear in the same context as X.
 - Limited dimensionality(10~1000)
 - Low dimensional and dense
 - Deep learning (to be precise representation learning) methods have been used
 - A more recent/modern approach

Continuous bag-of-word and skip-gram

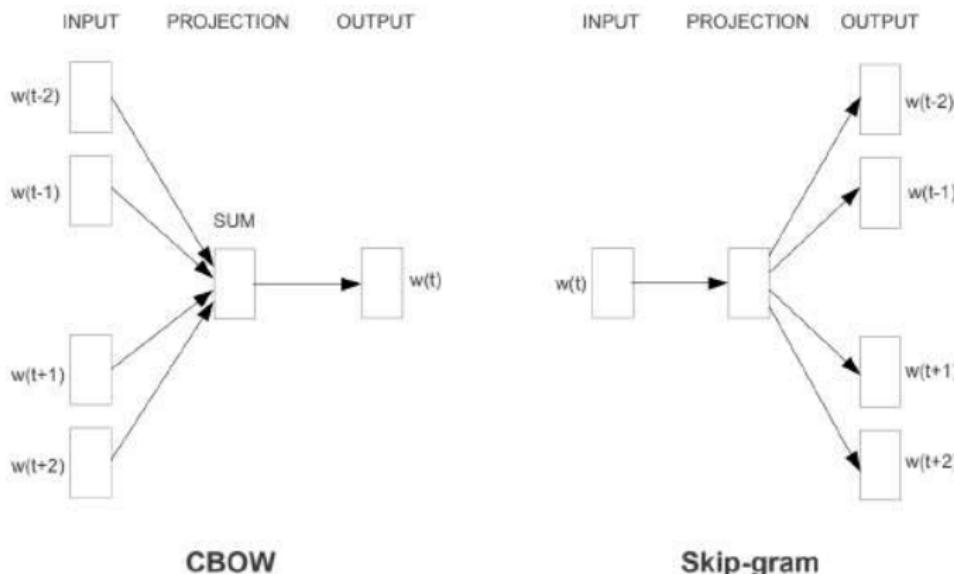
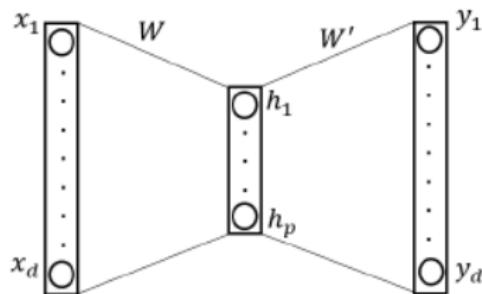


Figure: Source: Efficient Estimation of Word Representations in Vector Space, Tomas Mikolov et al., 2013

CBOW Model

- given context (window of words), predict words



- input: $\mathbf{x} \in \mathbb{R}^d$, $d = |\mathcal{V}|$, size of the vocabulary
- hidden layer: $\mathbf{h} = W^T \mathbf{x}$
- output: $\mathbf{u} = W'^T \mathbf{h}$
 $\mathbf{y} = \phi(\mathbf{u})$; ϕ = activation function

$$y_i = \frac{\exp(v_w'^T v_c)}{\sum_{w=1}^{\mathcal{V}} \exp(v_w'^T v_c)}$$

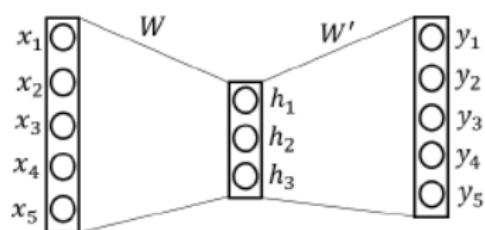
- parameters: $\{v_c \in W, v_w \in W'\}$

Figure: Architecture of CBOW model

CBOW Model with one word in context

Toy corpus:

- The dog ran
- The cat screamed



$$\mathcal{V} = [\text{the, dog, ran, can, screamed}]$$

One-hot vector representation:

the	[1, 0, 0, 0, 0]
dog	[0, 1, 0, 0, 0]
ran	[0, 0, 1, 0, 0]
cat	[0, 0, 0, 1, 0]
screamed	[0, 0, 0, 0, 1]

CBOW Model with one word in context

- learn relation between *cat* and *screamed*, compute $P(\text{word} \mid \text{context}) = P(w \mid c)$
- input word would be *cat*
- hidden vector $\mathbf{h} = W^T \mathbf{x}$

$$\mathbf{h}_{3 \times 1} = \begin{bmatrix} [1.5]h_1 \\ h_2 \\ h_3 \end{bmatrix}_{3 \times 1} = \begin{bmatrix} [1.5]w_{11} & w_{21} & w_{31} & w_{41} & w_{51} \\ w_{12} & w_{22} & w_{32} & w_{42} & w_{52} \\ w_{13} & w_{23} & w_{33} & w_{43} & w_{53} \end{bmatrix}_{3 \times 5} \begin{bmatrix} [1.5]0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}_{5 \times 1}$$

$$\mathbf{h}_{3 \times 1} = \begin{bmatrix} [1.5]w_{11} \times 0 + w_{21} \times 0 + w_{31} \times 0 + w_{41} \times 1 + w_{51} \times 0 + \\ w_{12} \times 0 + w_{22} \times 0 + w_{32} \times 0 + w_{42} \times 1 + w_{52} \times 0 + \\ w_{13} \times 0 + w_{23} \times 0 + w_{33} \times 0 + w_{43} \times 1 + w_{53} \times 0 + \end{bmatrix}_{3 \times 1} = \begin{bmatrix} [1.5]w_{41} \\ w_{42} \\ w_{43} \end{bmatrix}$$

CBOW Model with one word in context

Remember, we are computing $\mathbf{h} = W^T \mathbf{x}$ (hidden representation of context)

$$\begin{bmatrix} [1.5]w_{11} \times 0 + w_{21} \times 0 + w_{31} \times 0 + w_{41} \times 1 + w_{51} \times 0 + \\ w_{12} \times 0 + w_{22} \times 0 + w_{32} \times 0 + w_{42} \times 1 + w_{52} \times 0 + \\ w_{13} \times 0 + w_{23} \times 0 + w_{33} \times 0 + w_{43} \times 1 + w_{53} \times 0 + \end{bmatrix} = \\ \begin{bmatrix} [1.5]w_{41} \\ w_{42} \\ w_{43} \end{bmatrix}$$

- note, the fourth column of W^T is \mathbf{h}
- note the fourth column of W^T is the fourth row of W
- in the toy example, $W^T \mathbf{x}$ is the 4th row of W
- to generalise, $W^T \mathbf{x}$ is the k th row of W

let us denote

$$h = W^T \mathbf{x} = v_c \quad (1)$$

$$W = \begin{bmatrix} [1.5]w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \\ w_{51} & w_{52} & w_{53} \end{bmatrix}$$

v_c can also be regarded as the context vector in W

CBOW Model with one word in context

from hidden to output layer

we compute $\mathbf{u} = W'^T \mathbf{h}$

$$\mathbf{u} = \begin{bmatrix} [1.5]w'_{11} & w'_{21} & w'_{31} \\ w'_{12} & w'_{22} & w'_{32} \\ w'_{13} & w'_{23} & w'_{33} \\ w'_{14} & w'_{24} & w'_{34} \\ w'_{15} & w'_{25} & w'_{35} \end{bmatrix} \begin{bmatrix} [1.5]h_1 \\ h_2 \\ h_3 \end{bmatrix}$$

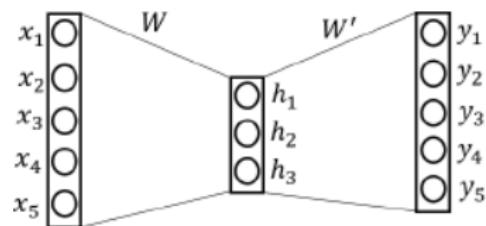
$$u_i = {W}_{:,i}'^T \mathbf{h}$$

${W}_{:,i}'^T$ is the i th row of W'^T , which is the i th column in W'

let us denote

$${W}_{:,i}'^T = v_w \quad (2)$$

v_w is also the word vector in W'



$$W' = \begin{bmatrix} [1.5]w'_{11} & w'_{12} & w'_{13} & w'_{14} & w'_{15} \\ w'_{21} & w'_{22} & w'_{23} & w'_{24} & w'_{25} \\ w'_{31} & w'_{32} & w'_{33} & w'_{34} & w'_{35} \end{bmatrix}$$

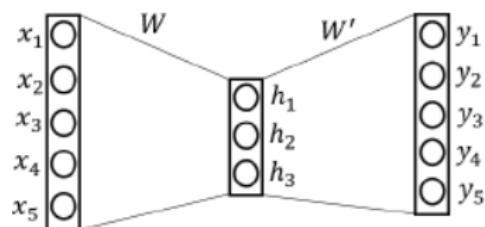
CBOW Model with one word in context

- $y_i = \phi(u_i)$
- softmax activation is used to compute probability for each y_i

$$y_i = \frac{\exp(u_i)}{\sum_{i=1}^V \exp(u_i)}$$

$$y_i = \frac{\exp(W_i^T h)}{\sum_{i=1}^V \exp(W_i^T h)}$$

- from (1) and (2): $y_i = \frac{\exp(v_w^T v_c)}{\sum_{w=1}^V \exp(v_w^T v_c)}$
- To train the network, we learn v_w, v_c
- objective function: since we predict probabilities at each y_i , we can use maximum likelihood to maximise the likelihood of observing the data



CBOW Model with one word in context

- maximise log likelihood of the model

parameters: $\theta = \{v_c, v_w\}$

likelihood: $L(\theta) = \prod_{w \in V} P(w | c; \theta)$

$L = \log(L(\theta)) = \sum_{w \in V} \log P(w | c; \theta)$

$$L = \sum_{w \in V} \log \frac{\exp(v_w^T v_c)}{\sum_{w=1}^V \exp(v_w^T v_c)}$$

$$L = \sum_{w \in V} v_w^T v_c - \log \sum_{w=1}^V \exp(v_w^T v_c)$$

- compute $\frac{\partial L}{\partial v_w}, \frac{\partial L}{\partial v_c}$ to maximise the likelihood
- consider $\frac{\partial L}{\partial v_w} = v_c - \frac{1}{\sum_{w=1}^V \exp(v_w^T v_c)} \exp(v_w^T v_c) v_c$
- $\frac{\partial L}{\partial v_w} = v_c - P(w | c) v_c$
- $\frac{\partial L}{\partial v_w} = v_c(1 - P(w | c))$

CBOW Model with one word in context

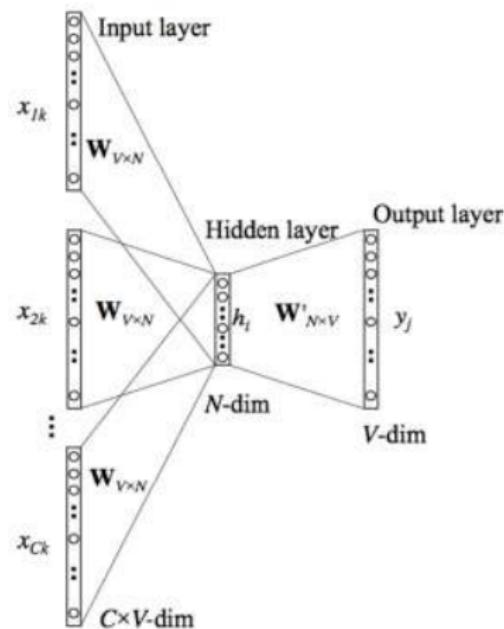
- update gradients using gradient descent.

$$v_w = v_w - \eta \frac{\partial L}{\partial v_w}$$

$$v_w = v_w - \eta [v_c(1 - P(w | c))]$$

- similarly $\frac{\partial L}{\partial v_c}$ can be computed to update v_c .

CBOW Model- multiple context words



Continuous bag-of-word and skip-gram

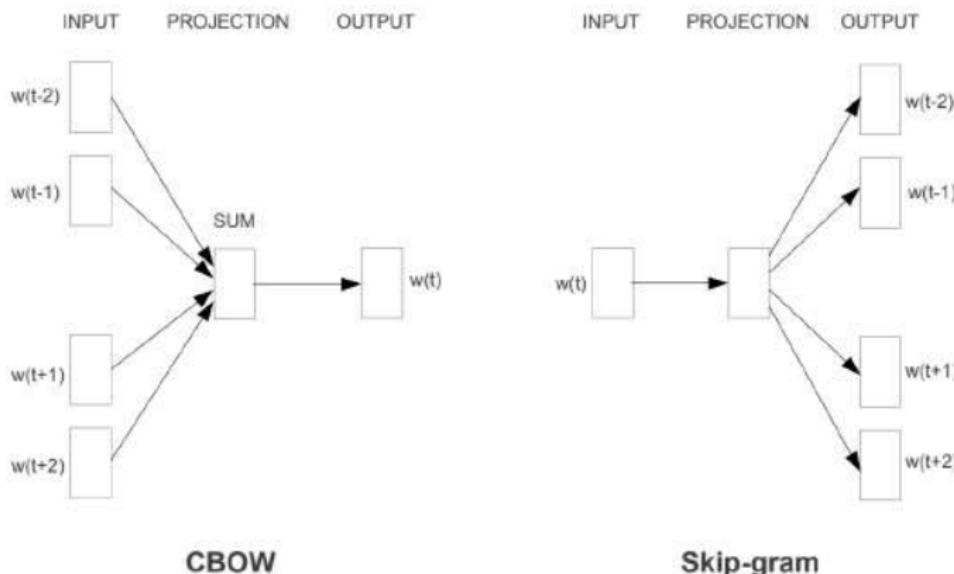
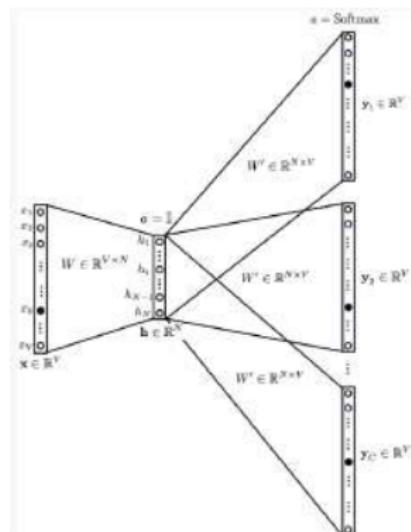


Figure: Source: Efficient Estimation of Word Representations in Vector Space, Tomas Mikolov et al., 2013

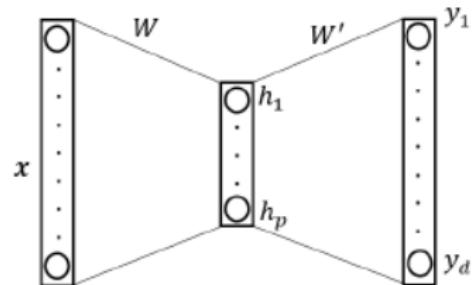
Skip-gram model

- architecture similar to CBOW model
- given input word (center word) predict context words
- CBOW model: one set of output probabilities; Skip-gram model: c set of output probabilities.
- output: $y_i = \frac{\exp(v_w^T v_c)}{\sum_{w=1}^V \exp(v_w^T v_c)}$
- computationally expensive to compute softmax function ($10^5 - 10^7$ terms).
- solution: negative sampling



Negative Sampling

- turn unsupervised learning into supervised learning.
- create two sets of word pairs:
 - \mathcal{D} : set of all word pairs (w, c) present in the text
 - \mathcal{D}' : set of all word pairs (w, c) that are not present in the text
- we can now define:
 - $z = 1$ if $(w, c) \in \mathcal{D}$
 - $z = 0$ if $(w, c) \in \mathcal{D}'$

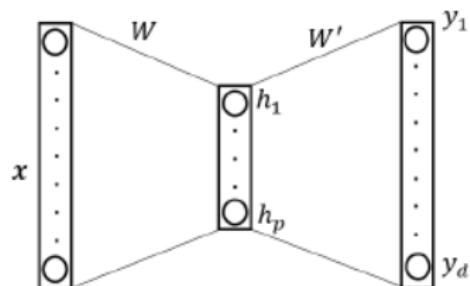


Negative Sampling

- instead, we can compute probabilities using logistic regression
 - $P(z = 1 | w, c)$ is the probability that (w, c) is in \mathcal{D}
 - $P(z = 0 | w, c)$ is the probability that (w, c) is in \mathcal{D}'
- we have turned the task into classification problem
- we can apply logistic function to compute probabilities

thus,

$$P(z = 1 | w, c) = \sigma(v_c^T v_w) = \frac{1}{1 + e^{-v_c^T v_w}}$$



Negative Sampling

- if, $P(z = 1 | w, c) = \frac{1}{1 + e^{-v_c^T v_w}}$
- then $P(z = 0 | w, c) = 1 - \frac{1}{1 + e^{-v_c^T v_w}}$

$$P(z = 0 | w, c) = \frac{e^{-v_c^T v_w}}{1 + e^{-v_c^T v_w}}$$

$$P(z = 0 | w, c) = \frac{1}{1 + e^{v_c^T v_w}} = \sigma(-v_c^T v_w)$$

- probabilities can be written in a generalised form:

$$P(z | w, c; \theta) = \left(\frac{1}{1 + e^{-v_c^T v_w}} \right)^z \left(\frac{1}{1 + e^{v_c^T v_w}} \right)^{(1-z)}, \text{ where } \theta = \{v_c, v_w\}$$

Negative Sampling

- probabilities can be written in a generalised form:

$$P(z \mid w, c; \theta) = \left(\frac{1}{1+e^{-v_c^T v_w}} \right)^z \left(\frac{1}{1+e^{v_c^T v_w}} \right)^{(1-z)}, \text{ where } \theta = \{v_c, v_w\}$$

- to find v_c, v_w , we can use maximum likelihood for all (w, c) pairs in $\mathcal{D}, \mathcal{D}'$

$$\text{thus: } L(\theta) = \prod_{(w,c) \in \mathcal{D}, \mathcal{D}'} \left(\left(\frac{1}{1+e^{-v_c^T v_w}} \right)^z \left(\frac{1}{1+e^{v_c^T v_w}} \right)^{(1-z)} \right)$$

- to find the log likelihood:

$$\log(L(\theta)) = \sum_{(w,c) \in \mathcal{D}, \mathcal{D}'} z \log \left(\frac{1}{1+e^{-v_c^T v_w}} \right) + (1 - z) \log \left(\frac{1}{1+e^{v_c^T v_w}} \right)$$

Negative Sampling

- to further simplify:

$$\log(L(\theta)) = \sum_{(w,c) \in \mathcal{D}} \log \left(\frac{1}{1+e^{-v_c^T v_w}} \right) + \sum_{(w,c) \in \mathcal{D}'} \log \left(\frac{1}{1+e^{v_c^T v_w}} \right)$$

$$\log(L(\theta)) = \sum_{(w,c) \in \mathcal{D}} \log \sigma(v_c^T v_w) + \sum_{(w,c) \in \mathcal{D}'} \log \sigma(-v_c^T v_w)$$

- loss function :

$$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^k \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right]$$

Hierarchical Softmax

- provides computationally efficient approximation of the full softmax
- main advantage is that instead of evaluating W output nodes to obtain probability distribution, we evaluate about $\log_2(W)$
- the depth of the binary tree depends on the vocabulary
- the path to any given word in the vocabulary is known
- length to reach any word is $\log_2(\mathcal{V})$



Figure: Balanced Binary Tree

Hierarchical Softmax

- We can arrange the words using:
 - random order
 - WordNet based rules
 - frequency

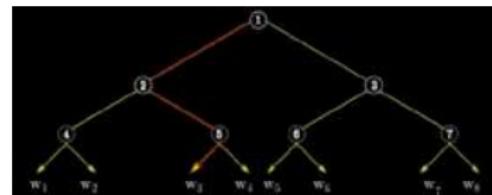
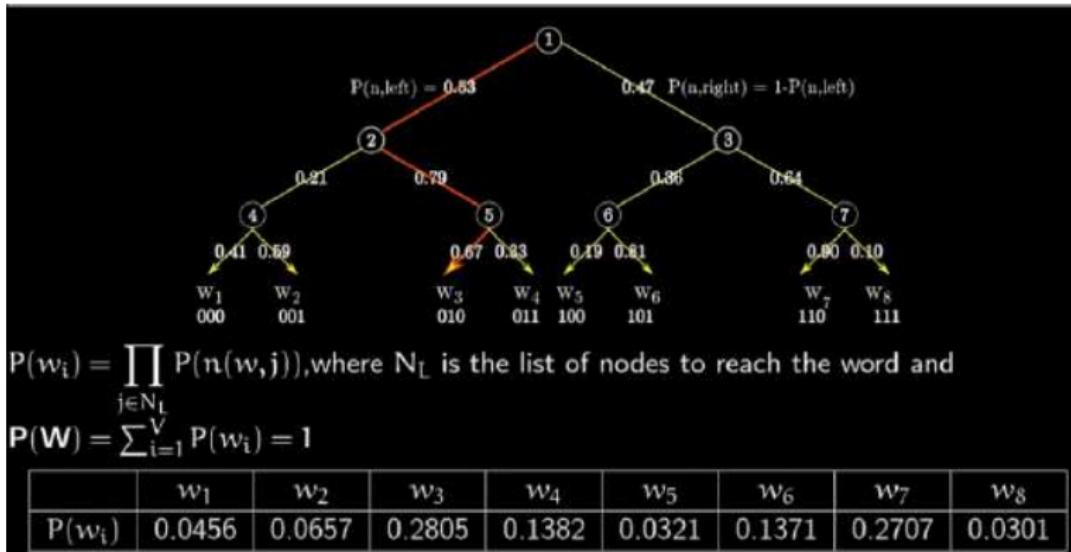


Figure: Balanced Binary Tree

Hierarchical Softmax

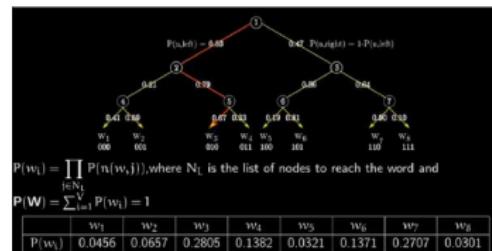
- provides a well-defined multinomial distribution among all words



Hierarchical Softmax

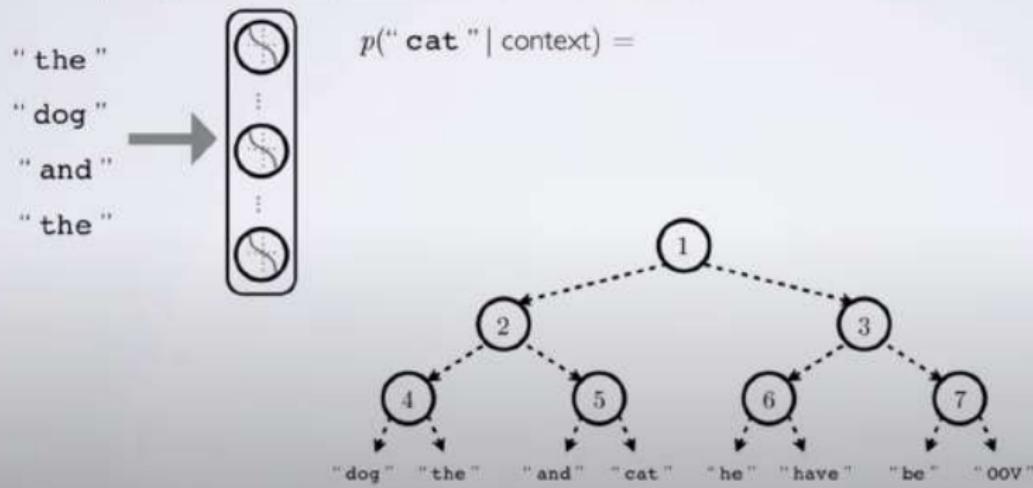
Advantages

- a flat hierarchy can be decomposed into a binary tree
- The path to each word (leaf) is unique
- Each intermediate node provides the relative probabilities of its child nodes
- every leaf represents the probability of the word



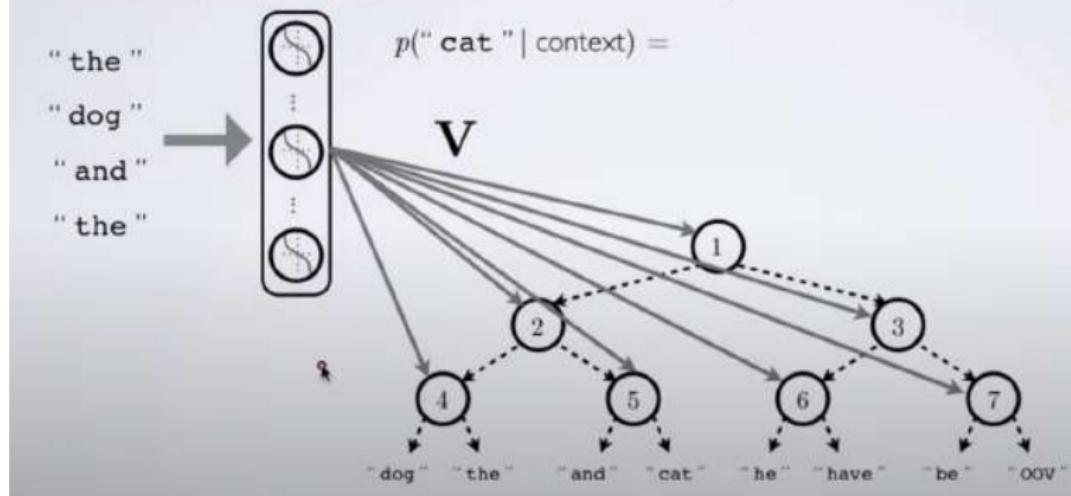
Hierarchical Softmax

- Example: ["the", "dog", "and", "the", "cat"]



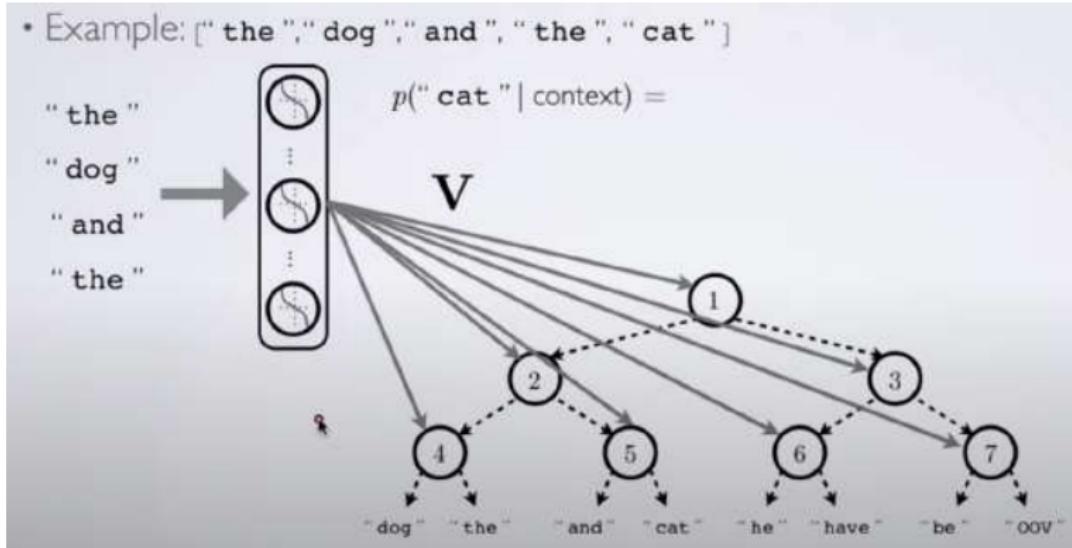
Hierarchical Softmax

- Example: ["the", "dog", "and", "the", "cat"]



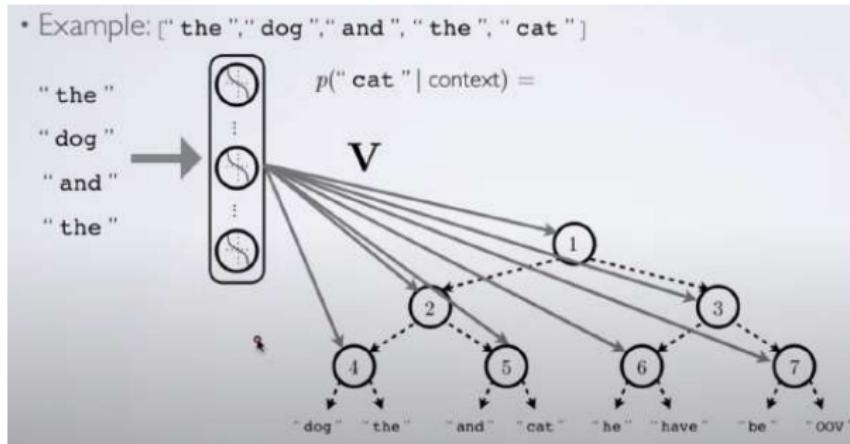
Hierarchical Softmax

- $P(\text{cat} \mid \text{context}) = p(\text{branch left at 1} \mid \text{context})p(\mid \text{branch right at 2} \parallel \text{context})p(\mid \text{branch right at 5} \parallel \text{context})$



Hierarchical Softmax

- How to compute probabilities at inner nodes?
 - probability at a node to its right child node $\sigma(v_w \cdot h)$
 - probability at a node to its left child node $1 - \sigma(v_w \cdot h)$



Hierarchical Softmax

$$\text{item } P(\text{cat} \mid \text{context}) = (1 - \sigma(v_w \mathbf{h}))(\sigma(v_w \mathbf{h}))(\sigma(v_w \mathbf{h}))$$

- Example: ["the", "dog", "and", "the", "cat"]

