Inf2b - Learning
Lecture 1: Introduction to Learning and Data

Hiroshi Shimodaira
(Credit: Iain Murray and Steve Renals)

Centre for Speech Technology Research (CSTR)
School of Informatics
University of Edinburgh

http://www.inf.ed.ac.uk/teaching/courses/inf2b/
https://piazza.com/ed.ac.uk/spring2020/infr08028
Office hours: Wednesdays at 14:00-15:00 in IF-3.04

Jan-Mar 2020

Welcome to Inf2b - Learning!

Today's Schedule:

1. Course structure
2. What is (machine) learning? (and why should you care?)
3. Administrative stuff
   - How to do well
4. Setting up a learning problem

(time allowing)

Course structure

http://www.inf.ed.ac.uk/teaching/courses/inf2b/

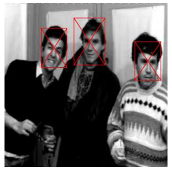
- 15+1 lectures (including review) - Tuesdays, Fridays
- Tutorials (starting in week 4)
- Drop-in labs for Learning (Tue 11:10-13:00, Wed 13:10-15:00)
- 1 assessed assignment (with drop-in labs)
  CW1 : 06/Mar. – 03/Apr.

Drop-in labs for Learning

- Tuesdays 11:10-13:00, Wednesdays 13:10-15:00 in AT-6.06
  Starting in Week 2. Both sessoins are the same.
- Worksheets available from the course webpage
- Purposes of lab sessions
  - Assistance in understanding basic algorithms and techniques of machine learning and data analysis
  - Assistance in programming with Matlab
  - Assistance in working on the assignment (CW1)
- Practice on machine learning using Matlab
  - Work on toy problems for the topics taught in the course
- Demonstrator: Teodora Georgescu (Tuedays), Riccardo Fiorista (Wednesdays)

Face detection

How would you detect a face?

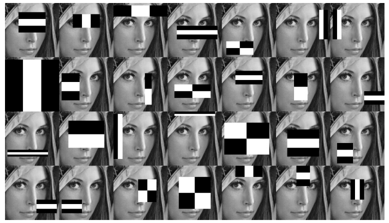


(R. Vaillant, C. Monrocq and Y. LeCun, 1994)

How does album software tag your friends?

http://demo.pittpatt.com/

Viola–Jones Face detection (2001)



- Face detector consists of linear combination of 'weak' classifiers that utilise five types of primitive features.
- The detector is trained on a training data set of a large number of positive and negative samples.
- Scan the input image with a sub-window (24 x 24 pixels) to detect a face.

Taken from: https://ahprojects.com/cvdazzle/
A nice demo: http://vimeo.com/12774628

Hiding from the machines (cameras)

The Viola-Jones face detector is fast, but has some drawbacks.

Taken from: https://ahprojects.com/cvdazzle/

Applications of machine learning

Within informatics:

- **Vision:** as we've seen. (eg1, eg2)
- **Graphics:** increasingly data driven
- **AI & Natural Language Processing (NLP):** text search/summarisation, speech recognition/synthesis, e.g. IBM Watson
- **Robotics:** vision, planning, control, . . .
- **Compilers:** learning how to optimise
  and beyond: data analysis across the sciences

Every day:

- Adverts / recommendations all over the web · · · Big Data
- Discounts in Tescos http://www.mathworks.co.uk/discovery/big-data-matlab.html
- Speech recognition and synthesis (e.g. Siri, Echo), Machine Translation, . . . with self-driving cars

Intro summary

- Fit numbers in a program to data (i.e. train machines on data)
- More robust than hand-fitted rules
- Can't approach humans at some tasks (e.g., vision)
- Machines make better predictions in many other cases

Attendance monitoring

- Attendance monitoring with Top Hat
  - Informatics 2B - Learning
  - Join code: **322890**

Private study

- $\sim$2 hours private study per lecture *in addition to tutorials & assignments*
- No required textbook for Inf2b There are notes and slides. See those for recommended books.
- Importance of maths skills (especially algebra)
  Why should you remember and get familiar with maths formulas for machine learning?
  - Good understanding of the ideas
  - Guessing reasonable output of the model
  - Identifying/spotting the problems (bugs) with the system implemented
- Importance of programming practice [with Matlab or Python] (attend the drop-in labs!)

Private study *(cont.)*

- Warning: Inf2b is NOT an easy course
- Inf2b requires a solid maths background:
  - Linear Algebra
  - Calculus
  - Probability
- Independent learning (self-directed learning) is essential.
  See the following page regarding differences between secondary-school and university in terms of learning style and what is expected from you as a student.
  https://www.birmingham.ac.uk/accessibility/transcripts/school-uni-differences.aspx
- For exam preparation, use not only notes, but also slides and tutorial sheets. NB: slides are not just the summaries of notes.

Maths skills

Useful webpage to check your maths:
http://www.mathsisfun.com/algebra

- Laws of exponents (Exponent rules)
  e.g. $x^m x^n = x^{m+n}$, $(x^m)^n = x^{mn}$
- Log and exponential
  e.g. $\log(x^n y^m) = n \log x + m \log y$, $e^{\ln x} = x$
- Quadratic equations and their solutions
  e.g. $ax^2 + bx + c = 0$, $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
- Vectors $\mathbf{v} = (v_1, v_2, \ldots, v_D)^T$
  - Notation: column/row vectors, transpose
  - Addition and subtraction eg. $\mathbf{u} + \mathbf{v}$
  - Dot product (inner product) $\mathbf{u} \cdot \mathbf{v} = \mathbf{u}^T \mathbf{v}$
- Equation of a straight line, linear equations

Maths skills (cont.)

- Matrices $A = (a_{ij})$, $A_{ij} = a_{ij}$
  - Addition, subtraction $A+B$, $A-B$
  - Multiplication $(AB)_{ij} = \sum_{k=1}^{d} a_{ik} b_{kj}$
  - Transpose $(ABC)^T = C^T B^T A^T$
  - Determinant $|A|$
  - Inverse $A^{-1}A = AA^{-1} = I$
  - Eigenvalues and eigenvectors
  - Vector spaces, subspaces, linear independence, basis and dimension, rank and nullity
  - Linear transformations $\mathbf{y} = A\mathbf{x}$

NB: See Section 4 of Learning Note No. 1 for the notation we use.

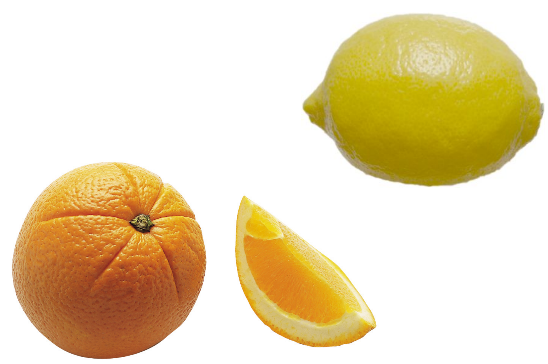
Two hours study this week?

- **Start to familiarise yourself with** MATLAB (or OCTAVE)
  Introductory worksheet on the course website
  Many others at the end of a web search
- **Learn Matlab** try the lab sheets for the 1st lab this week.
- **Love Python?** Learn `NumPy+SciPy+Matplotlib` (instead, or as well)
- **Vital skills:**
  - add, average, multiply vectors and matrices
  - plot data stored in vectors
  - save/read data to/from files

Classroom

- Have a look at the lecture note and slides in advance to the lecture.
- Have questions prepared to ask.

Classification of oranges and lemons



A two-dimensional space

Represent each sample as a point (w, h) in a 2D space

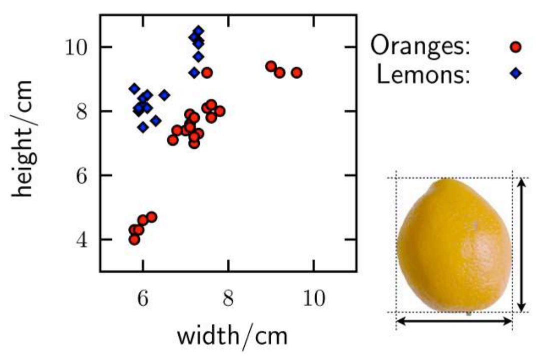
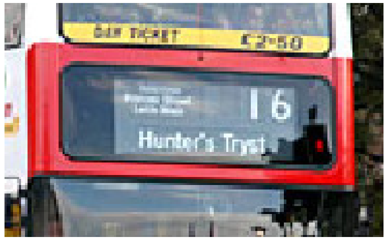
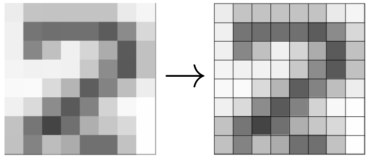


Photo image – pixels



Pixel image to a feature vector



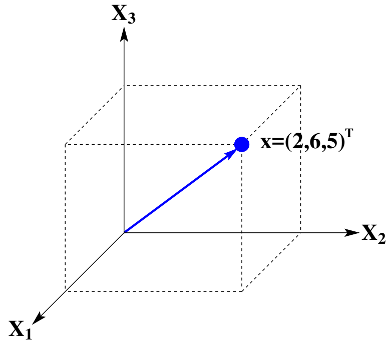
$$\longrightarrow \quad \longrightarrow \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{64} \end{pmatrix}$$

Turn each cell (pixel) into a number (somehow, see notes)
Unravel into a column vector, a **feature vector**
$\Rightarrow$ represented digit as point in $64D$

$$\mathbf{x} = (x_1, x_2, \ldots, x_{64})^T, \quad x_i \in \{0, \ldots, 127\} \text{ or } x_i \in \{0, 1\}$$

http://alex.seewald.at/digits/

Image data as a point in a vector space



Euclidean distance

Distance between $2D$ vectors: $\boldsymbol{u} = (u_1, u_2)^T$ and $\boldsymbol{v} = (v_1, v_2)^T$

$$r_2(\boldsymbol{u}, \boldsymbol{v}) = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2}$$

Distance between D-dimensional vectors: $\boldsymbol{u} = (u_1, \ldots, u_D)^T$ and $\boldsymbol{v} = (v_1, \ldots, v_D)^T$

$$r_2(\boldsymbol{u}, \boldsymbol{v}) = \sqrt{\sum_{k=1}^{D} (u_k - v_k)^2}$$

Measures similarities between feature vectors
i.e., similarities between digits, movies, sounds, galaxies, ...

Question

Have high-resolution scans of digits.

How many pixels should be sample?

What are pros and cons of:

2×2, 4×4, 16×16, or 100×100?

Example of image resolutions

Exercises in the lecture note 1

- Try the exercises in the lecture note 1.
- No solutions will be published.
- In case you're not sure if your answers are correct.
  - Discuss them with your classmates
  - Use the Inf2b-Learning discussion board on Piazza

Summary

- Self-study everyday.
- Drop-in labs for Learning starts in Week 2 (21st, 22nd Jan.)
  Try the worksheet before the lab.
- Tutorial starts in Week 4.
- Discussion forum in Piazza
- Office hours: Wednesdays at 14:00-15:00 (TBC) in IF-3.04

Inf2b - Learning

Lecture 2: Similarity and Reocommendation systems

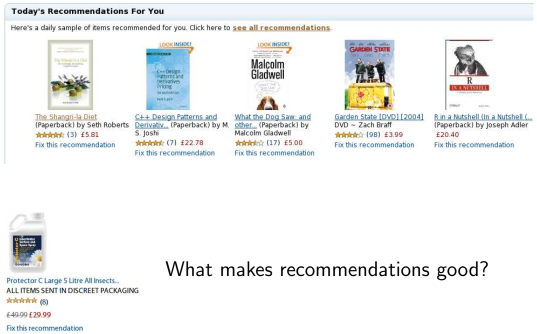
Hiroshi Shimodaira
(Credit: Iain Murray and Steve Renals)

Centre for Speech Technology Research (CSTR)
School of Informatics
University of Edinburgh

http://www.inf.ed.ac.uk/teaching/courses/inf2b/
https://piazza.com/ed.ac.uk/spring2020/infr08028
Office hours: Wednesdays at 14:00-15:00 in IF-3.04

Jan-Mar 2020

Recommender systems



What makes recommendations good?

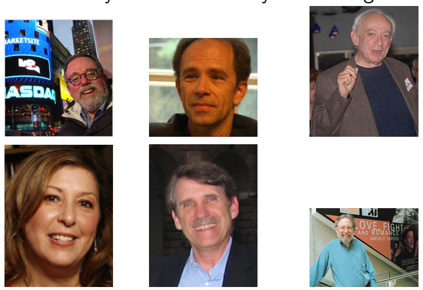
Today's schedule

1. Data and distances between entities
2. Similarity and recommendations
3. Normalisation, Pearson Correlation
4. Transposed problem

The Films in 2008

The Critics

David Denby Todd McCarthy Joe Morgenstern



Claudia Puig Peter Travers Kenneth Turan

Film review scores by critics – data

	Australia	*Body of Lies*	*Burn After*	*Hancock*	*Milk*	*Rev Road*
Denby	3	7	4	9	9	7
McCarthy	7	5	5	3	8	8
M'stern	7	5	5	0	8	4
Puig	5	6	8	5	9	8
Travers	5	8	8	8	10	9
Turan	7	7	8	4	7	8

Representation of data & notation:

$$X = \begin{pmatrix} 3 & 7 & 4 & 9 & 9 & 7 \\ 7 & 5 & 5 & 3 & 8 & 8 \\ 7 & 5 & 5 & 0 & 8 & 4 \\ 5 & 6 & 8 & 5 & 9 & 8 \\ 5 & 8 & 8 & 8 & 10 & 9 \\ 7 & 7 & 8 & 4 & 7 & 8 \end{pmatrix}$$

Score of movie m by critic c:

x_{cm}, $\mathrm{sc}_c(m)$

Score vector by critic c:

$\mathbf{x}_c = (x_{c1}, \ldots, x_{cM})^T$

aka feature vector

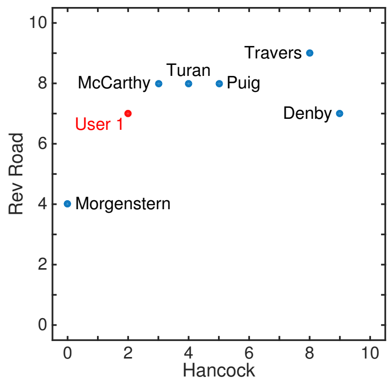
Problem definition

	Australia	*Body of Lies*	*Burn After*	*Hancock*	*Milk*	*Rev Road*
Denby	3	7	4	9	9	7
McCarthy	7	5	5	3	8	8
M'stern	7	5	5	0	8	4
Puig	5	6	8	5	9	8
Travers	5	8	8	8	10	9
Turan	7	7	8	4	7	8
User1	-	-	-	2	-	7
User2	-	6	9	-	-	6

Predict user's score $\hat{x}_{um}$ for unseen film m based on the film review scores by the critics. $\Rightarrow$ Film recommendation

(Fill the missing elements based on others)

A two-dimensional review space



Euclidean distance

Distance between 2D vectors: $\boldsymbol{u} = (u_1, u_2)^T$ and $\boldsymbol{v} = (v_1, v_2)^T$

$$r_2(\boldsymbol{u}, \boldsymbol{v}) = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2}$$

Distance between D-dimensional vectors: $\boldsymbol{u} = (u_1, \ldots, u_D)^T$ and $\boldsymbol{v} = (v_1, \ldots, v_D)^T$

$$r_2(\boldsymbol{u}, \boldsymbol{v}) = \sqrt{\sum_{k=1}^{D} (u_k - v_k)^2}$$

Measures similarities between feature vectors

i.e., similarities between digits, critics, movies, genes, ...

NB: $r_2(\,)$ denotes "2-norm", c.f. p-norm or L^p-norm. [Note 2]

cf. other distance measures, e.g. Hamming distance, city-block distance (L^1 norm).

Distances between critics

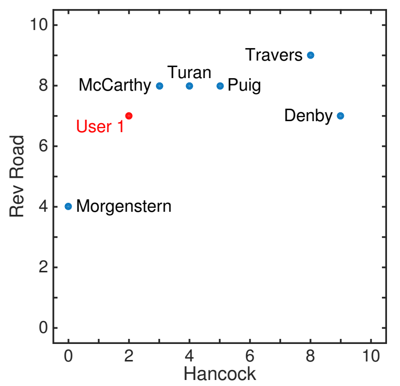
$$r_2(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{\sum_{m=1}^{M} (x_{im} - x_{jm})^2}$$

	Denby	McCarthy	M'stern	Puig	Travers	Turan
Denby		7.7	10.6	6.2	5.2	7.9
McCarthy	7.7		5.0	4.4	7.2	3.9
M'stern	10.6	5.0		7.5	10.7	6.8
Puig	6.2	4.4	7.5		3.9	3.2
Travers	5.2	7.2	10.7	3.9		5.6
Turan	7.9	3.9	6.8	**3.2**	5.6	

NB: Distances measured in a 6-dimensional space ($M = 6$)

The closest pair is Puig and Turan

2D distance between User1 and critics



$r_2(\text{User1}, \text{McCarthy})$

$$= \sqrt{(2-3)^2 + (7-8)^2}$$
$$= \sqrt{2}$$

$r_2(\text{User1}, \text{Turan})$

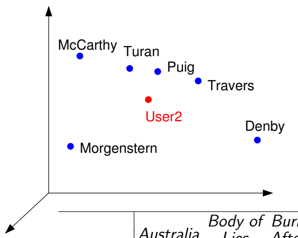
$$= \sqrt{(2-4)^2 + (7-8)^2}$$
$$= \sqrt{5}$$

Simple strategy 1 for film recommendation

- Find the closest critic, c^*, to User u,
- use x_{c^*m} for $\hat{x}_{um}$.

	Australia	*Body of Lies*	*Burn After*	*Hancock*	*Milk*	*Rev Road*
Denby	3	7	4	9	9	7
McCarthy	**7**	**5**	**5**	**3**	**8**	**8**
M'stern	7	5	5	0	8	4
Puig	5	6	8	5	9	8
Travers	5	8	8	8	10	9
Turan	7	7	8	4	7	8
User1	-	-	-	**2**	-	**7**
User2	-	6	9	-	-	6

Film recommendation for User2



	Australia	*Body of Lies*	*Burn After*	*Hancock*	*Milk*	*Rev Road*	r_2(critic, User2)
Denby	3	7	4	9	9	7	$\sqrt{27} \approx 5.2$
McCarthy	7	5	5	3	8	8	$\sqrt{21} \approx 4.6$
M'stern	7	5	5	0	8	4	$\sqrt{21} \approx 4.6$
Puig	5	6	8	5	9	8	$\sqrt{5} \approx 2.2$
Travers	5	8	8	8	10	9	$\sqrt{14} \approx 3.7$
Turan	7	7	8	4	7	8	$\sqrt{6} \approx 2.4$
User2	-	6	9	-	-	6	

Strategy 2

Consider not only the closest critic but also all the critics.

Option 1: The mean or average of critic scores for film m:

$$\hat{x}_{um} = \frac{1}{C}\sum_{c=1}^{C} x_{cm}$$

Option 2: Weighted average over critics:

Weight critic scores according to the *similarity* between the critic and user.

$$\hat{x}_{um} = \frac{1}{\sum_{c=1}^{C} \mathrm{sim}(\boldsymbol{x}_u, \boldsymbol{x}_c)} \sum_{c=1}^{C} \left(\mathrm{sim}(\boldsymbol{x}_u, \boldsymbol{x}_c) \cdot x_{cm}\right)$$

cf. Weighted arithmetic mean (weighted average) in maths:

$$\bar{x} = \frac{w_1 x_1 + w_2 x_2 + \cdots + w_n x_n}{w_1 + w_2 + \cdots w_n} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

NB: if every x_i has the same value, so does $\bar{x}$.

Similarity measures

There's a choice. For example:

$$\mathrm{sim}(\boldsymbol{u}, \boldsymbol{v}) = \frac{1}{1 + r_2(\boldsymbol{u}, \boldsymbol{v})}$$

Can now predict scores for User 2 (see notes)

Good measure?

- Consider distances 0, ∞, and in between.
- What if some critics rate more highly than others?
- What if some critics have a wider spread than others?
- What if not all critics have seen the same movies? (missing data problem)

Critic review score statistics

	Australia	*Body of Lies*	*Burn After*	*Hancock*	*Milk*	*Rev Road*	mean	std.
Denby	3	7	4	9	9	7	**6.5**	**2.5**
McCarthy	7	5	5	3	8	8	**6.0**	**2.0**
M'stern	7	5	5	0	8	4	**4.8**	**2.8**
Puig	5	6	8	5	9	8	**6.8**	**1.7**
Travers	5	8	8	8	10	9	**8.0**	**1.7**
Turan	7	7	8	4	7	8	**6.8**	**1.5**

Normalisation

Sample mean and **sample standard deviation** of critic c's scores:

$$\bar{x}_c = \frac{1}{M}\sum_{m=1}^{M} x_{cm}$$

$$s_c = \sqrt{\frac{1}{M-1}\sum_{m=1}^{M}(x_{cm} - \bar{x}_c)^2}$$

Different means and spreads make reviewers look different.

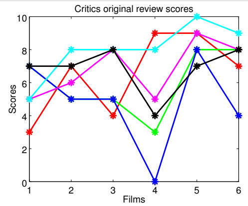
$\Rightarrow$ Create 'standardised score' with mean zero and st. dev. 1.

Standard score:

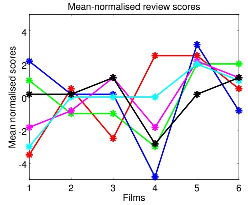
$$z_{cm} = \frac{x_{cm} - \bar{x}_c}{s_c}$$

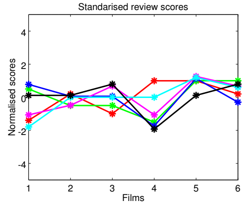
Many learning systems work better with standardised features / outputs

Normalisation of critics review scores



	Australia	*Body of Lies*	*Burn After*	*Hancock*	*Milk*	*Rev Road*
Denby	3	7	4	9	9	7
McCarthy	7	5	5	3	8	8
M'stern	7	5	5	0	8	4
Puig	5	6	8	5	9	8
Travers	5	8	8	8	10	9
Turan	7	7	8	4	7	8





Pearson correlation coefficient

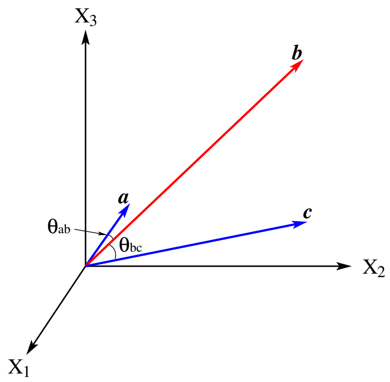
Estimate of 'correlation' between critics c and d:

$$r_{cd} = \frac{1}{M-1}\sum_{m=1}^{M} z_{cm}\, z_{dm}$$
$$= \frac{1}{M-1}\sum_{m=1}^{M}\left(\frac{x_{cm} - \bar{x}_c}{s_c}\right)\left(\frac{x_{dm} - \bar{x}_d}{s_d}\right).$$

- Based on standard scores
  (a shift and stretch of a reviewer's scale makes no difference – shift/scale invariant)
- $-1 \leq r_{cd} \leq 1$
- How r_{cd} can be used as a similarity measure?

Used in the mix by the winning netflix teams:
https://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pd

Pearson correlation coefficient *(cont.)*



And a trick: transpose your data matrix and run your code again. The result is sometimes interesting.

Transposed problem

Another strategy — based on distance between Movies

	Australia	*Body of Lies*	*Burn After*	*Hancock*	*Milk*	*Rev Road*
Australia		5.8	5.3	10.9	8.9	7.2
Body of Lies	5.8		3.7	6.6	5.9	4.0
Burn After	5.3	3.7		8.9	7.0	4.5
Hancock	10.9	6.6	8.9		10.9	8.4
Milk	8.9	5.9	7.0	10.9		4.8
Rev. Road	7.2	4.0	4.5	8.4	4.8	

Run the same code for distance between critics, simply **transpose the data matrix** first

Transpose of `data` in numpy is `data.T`, in Matlab/Octave it's `data'`

The Netflix million dollar prize

$C = 480,189$ users/critics

$M = 17,770$ movies

$C \times M$ matrix of ratings $\in \{1, 2, 3, 4, 5\}$

(ordinal values)

Full matrix $\sim$ 10 billion cells
$\sim$ 1% cells filled (100,480,507 ratings available)

References *(NE)*

- https://www.netflixprize.com
- https://doi.org/10.1109/MSPEC.2009.4907383
- https://doi.org/10.1109/MC.2009.263

Further reading *(NE)*

- J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez, Recommender systems survey, Knowledge-Based Systems, Volume 46, 2013, pp.109-132. https://doi.org/10.1016/j.knosys.2013.03.012
- Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, Guangquan Zhang, Recommender system application developments: A survey, Decision Support Systems, Volume 74, 2015, pp.12-32. https://doi.org/10.1016/j.dss.2015.03.008
- Shuai Zhang, Lina Yao, Aixin Sun, Yi Tay Deep Learning based Recommender System: A Survey and New Perspectives, ACM Computing Surveys (CSUR), February 2019, Article No.: 5. https://doi.org/10.1145/3285029

Quizzes

Q1: Give examples for $r_{cd} \approx -1,\ 0,$ and 1.

Q2: Show the Pearson correlation coefficient can be rewritten as

$$r_{cd} = \frac{\sum_{m=1}^{M}(x_{cm} - \bar{x}_c)(x_{dm} - \bar{x}_d)}{\sqrt{\sum_{m=1}^{M}(x_{cm} - \bar{x}_c)^2}\sqrt{\sum_{m=1}^{M}(x_{dm} - \bar{x}_d)^2}}$$

Q3: How the missing data of critics scores should be treated?

Q4: What if a user provides scores for a few films only?

Summary

- **Rating prediction:** fill in entries of a $C \times M$ matrix
- A row is a feature vector of a critic
- Guess cells based on weighted average of similar rows
- Similarity based on distance and Pearson correlation coef.
- Could transpose matrix and run same code!
- NB: we considered a very simple case only.
- Try the exercises in Note 2, and do programming in Lab 2.

Drop-in labs for Learning

- Lab1 on 21th at 11:10-13:00, 22nd Jan. at 13:10-15:00 in AT-6.06.

  "Similarity and recommender systems"
- Lab worksheet available from the course web page.
- Questions outside the lab hours:
  http://piazza.com/ed.ac.uk/spring2019/infr08009inf2blearning

Matlab/Octave version

```
c_scores = [
    3 7 4 9  9 7;
    7 5 5 3  8 8;
    7 5 5 0  8 4;
    5 6 8 5  9 8;
    5 8 8 8 10 9;
    7 7 8 4  7 8]; % CxM
u2_scores = [6 9 6];
u2_movies = [2 3 6]; % one-based indices

% The next line is complicated. See also next slide:
d2 = sum(bsxfun(@minus, c_scores(:,u2_movies), u2_scores).^2, 2)';
r2 = sqrt(d2);
sim = 1./(1 + r2); % 1xC
pred_scores = (sim * c_scores) / sum(sim) % 1xM = 1xC * CxM
```


Matlab/Octave square distances

Other ways to get square distances:

```
% The next line is like the Python, but not valid Matlab.
% Works in recent builds of Octave.
d2 = sum((c_scores(:,u2_movies) - u2_scores).^2, 2)';

% Old-school Matlab way to make sizes match:
d2 = sum((c_scores(:,u2_movies) - ...
       repmat(u2_scores, size(c_scores,1), 1)).^2, 2)';

% Sq. distance is common; I have a general routine at:
% homepages.inf.ed.ac.uk/imurray2/code/imurray-matlab/square_dist.m
d2 = square_dist(u2_scores', c_scores(:,u2_movies)');
```

Or you could write a for loop and do it as you might in Java. Worth doing to check your code.

NumPy programming example

```
from numpy import *

c_scores = array([
    [3, 7, 4, 9,  9, 7],
    [7, 5, 5, 3,  8, 8],
    [7, 5, 5, 0,  8, 4],
    [5, 6, 8, 5,  9, 8],
    [5, 8, 8, 8, 10, 9],
    [7, 7, 8, 4,  7, 8]]) # C,M
u2_scores = array([6, 9, 6])
u2_movies = array([1, 2, 5]) # zero-based indices

r2 = sqrt(sum((c_scores[:,u2_movies] - u2_scores)**2, 1).T) # C,
sim = 1/(1 + r2) # C,
pred_scores = dot(sim, c_scores) / sum(sim)
print(pred_scores)

# The predicted scores has predictions for all movies,
# including ones where we know the true rating from u2.
```


Inf2b - Learning

Lecture 3: Clustering and data visualisation

Hiroshi Shimodaira
(Credit: Iain Murray and Steve Renals)

Centre for Speech Technology Research (CSTR)
School of Informatics
University of Edinburgh
http://www.inf.ed.ac.uk/teaching/courses/inf2b/
https://piazza.com/ed.ac.uk/spring2020/infr08028
Office hours: Wednesdays at 14:00-15:00 in IF-3.04

Jan-Mar 2020

Today's Schedule

1. What is clustering
2. *K*-means clustering
3. Hierarchical clustering
4. Example – unmanned ground vehicle navigation
5. Dimensionality reduction with PCA and data visualisation
6. Summary

Clustering

- Clustering: partition a data set into meaningful or useful groups, based on distances between data points
- Clustering is an unsupervised process — the data items do not have class labels
- Why cluster?
  - Interpreting data: Analyse and describe a situation by automatically dividing a data set into groupings
  - Compressing data: Represent data vectors by their cluster index — vector quantisation

Clustering

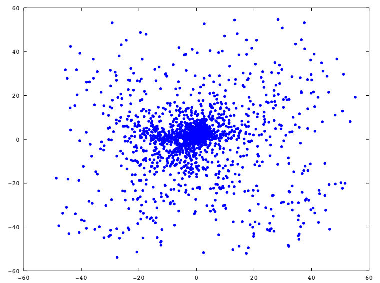
"Human brains are good at finding regularities in data. One way of expressing regularity is to put a set of objects into groups that are similar to each other. For example, biologists have found that most objects in the natural world fall into one of two categories: things that are brown and run away, and things that are green and don't run away. The first group they call animals, and the second, plants."

Recommended reading: David MacKay textbook, p284–

http://www.inference.phy.cam.ac.uk/mackay/itila/

Visualisation of film review users

MovieLens data set
(http://grouplens.org/datasets/movielens/)
$C \approx 1000$ users, $M \approx 1700$ movies

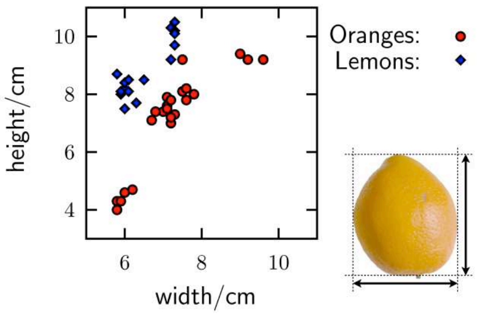


2D plot of users based on rating similarity

Application of clustering

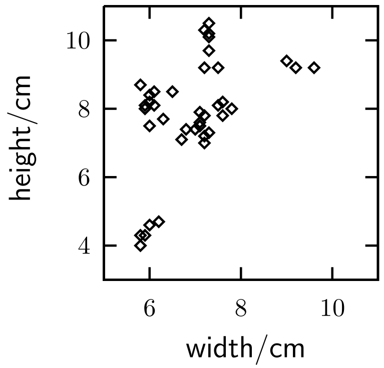
- Face clustering
  doi: 10.1109/CVPR.2013.450
  LHI-Animal-Face dataset
- Image segmentation
  http://dx.doi.org/10.1093/bioinformatics/btr246
- Document clustering
  Thesaurus generation
- Temporal Clustering of Human Behaviour
  http://www.f-zhou.com/tc.html

A two-dimensional space

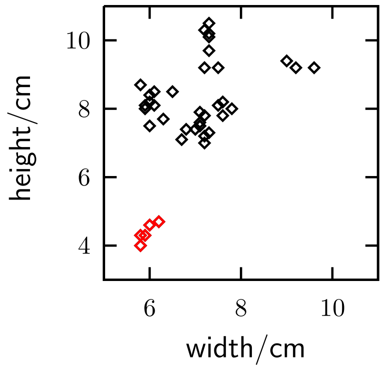


http://homepages.inf.ed.ac.uk/imurray2/teaching/oranges_and_lemons/

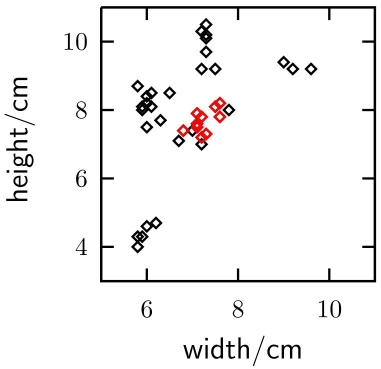
The Unsupervised data



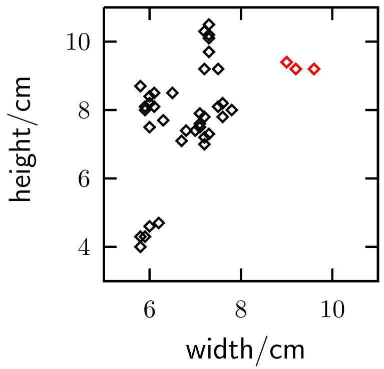
Manderins



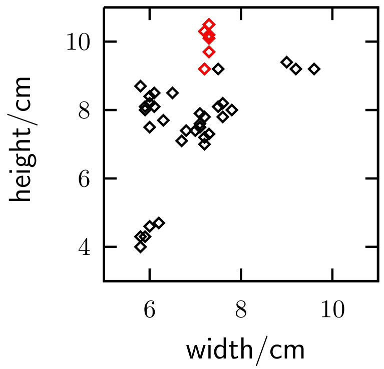
Navel oranges



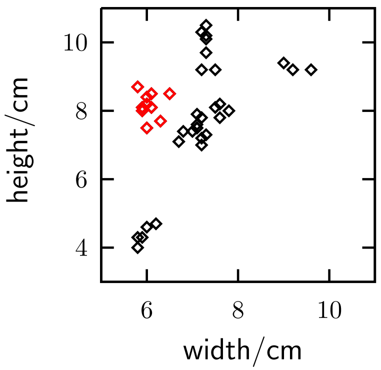
Spanish jumbo oranges



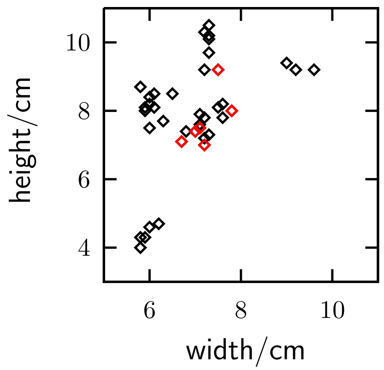
Belsan lemons



Some other lemons



"Selected seconds" oranges



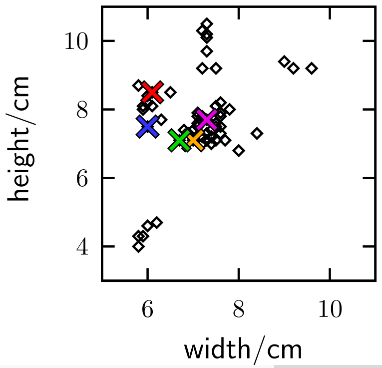
K-means clustering

A simple algorithm to find clusters:

1. Pick K random points as cluster centre positions
2. Assign each point to its nearest centre*
3. Move each centre to mean of its assigned points
4. If centres moved, `goto` 2.

* In the unlikely event of a tie, break tie in some way. For example, assign to the centre with smallest index in memory.

K-means clustering



Evaluation of clustering

- One way to measure the quality of a k-means clustering solution is by a *sum-squared error function*, i.e. the sum of squared distances of each point from its cluster centre.
- Let $z_{kn} = 1$ if the point $\mathbf{x}_n$ belongs to cluster k and $z_{kn} = 0$ otherwise. Then:

$$E = \sum_{k=1}^{K}\sum_{n=1}^{N} z_{kn} \|\mathbf{x}_n - \mathbf{m}_k\|^2$$

$\mathbf{x}_n = (x_{n1}, \ldots, x_{nD})^T$
$\mathbf{m}_k = (m_{k1}, \ldots, m_{kD})^T$
$\|\cdot\|$: Euclidean (L^2) norm

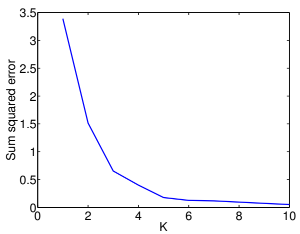
where $\mathbf{m}_k$ is the centre of cluster k.
- Sum-squared error is related to the variance — thus performing k-means clustering to minimise E is sometimes called minimum variance clustering.
- This is a within-cluster error function — it does not include a between clusters term

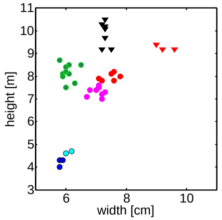
Theory of K-means clustering

- If assignments don't change, algorithm terminates.
- **Can assignments cycle, never terminating?**
- **Convergence proof technique:** find a *Lyapunov function* $\mathcal{L}$, that is bounded below and cannot increase.
  $\mathcal{L}$ = sum of square distances between points and centres
  NB: $E^{(t+1)} \leq E^{(t)}$
- **K-means is an optimisation algorithm** for $\mathcal{L}$. *Local optima* are found, i.e. there is no guarantee of finding global optimum. Running multiple times and using the solution with best $\mathcal{L}$ is common.

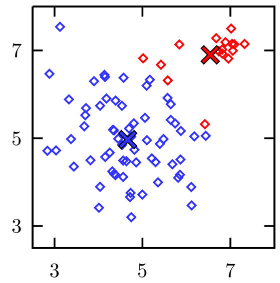
How to decide K?

- The sum-squared error decreases as K increases ($E \to 0$ as $K \to N$)
- We need another measure?!



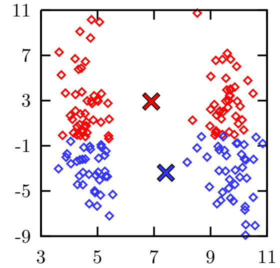


Failures of K-means (e.g. 1)



Large clouds pull small clusters off-centre

Failures of K-means (e.g. 2)



Distance needs to be measured sensibly.

Clustering clustering methods (NE)

- K-means clustering is not the only method for clustering data
- See: http://en.wikipedia.org/wiki/Cluster_analysis

Hierarchical clustering (NE)

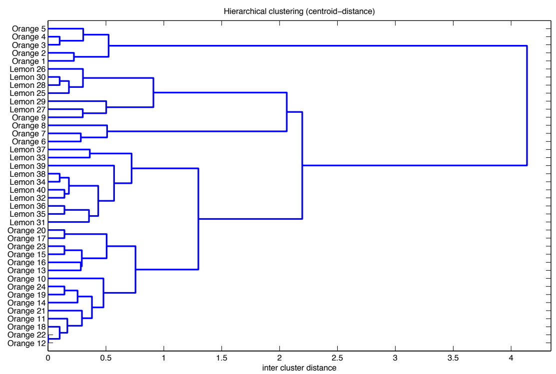
Form a 'dendrogram' / binary tree with data at leaves

Bottom-up / Agglomerative:
- Repeatedly merge closest groups of points
- Often works well. Expensive: $O(N^3)$

Top-down / Divisive:
- Recursively split groups into two (e.g. with k-means)
- Early choices might be bad.
- Much cheaper! $\sim O(N^2)$ or $O(N^2 \log N)$

More detail:
Pattern Classification (2nd ed.), Duda, Hart, Stork. §10.9

Bottom-up clustering of the lemon/orange data

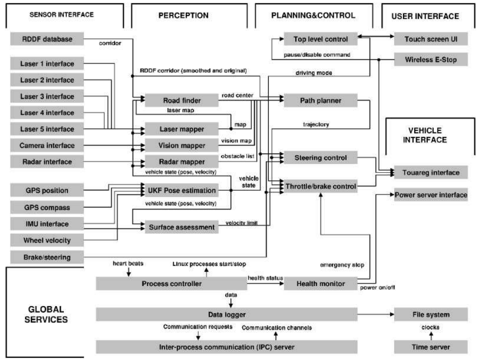


Stanley

Stanford Racing Team; DARPA 2005 challenge

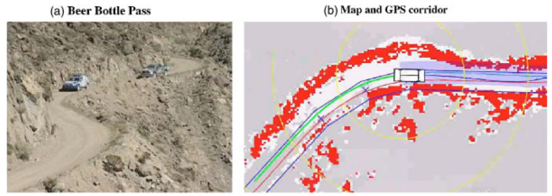
http://robots.stanford.edu/talks/stanley/

Inside Stanley



Stanley figures from Thrun et al., J. Field Robotics 23(9):661, 2006.

Perception and intelligence

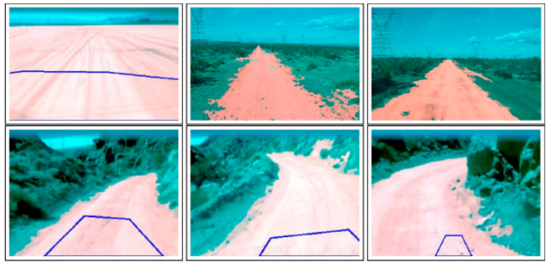


It would look pretty stupid to run off the road, just because the trip planner said so.

How to stay on the road?

Classifying road seems hard. Colours and textures change: road appearance in one place may match ditches elsewhere.

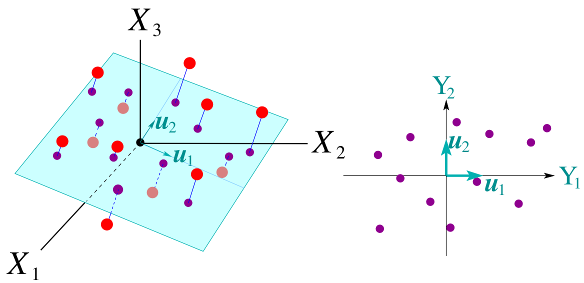
Clustering to stay on the road



Stanley used a Gaussian mixture model. "Souped up *k*-means."
The cluster just in front is road (unless we already failed).

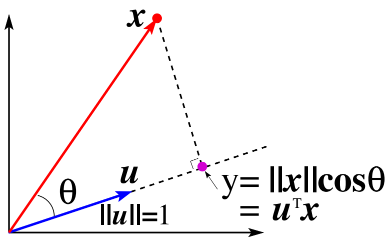
Dimensionality reduction and data visualisation

- High-dimensional data are difficult to understand and visualise.
- Consider dimensionality reduction of data for visualisation

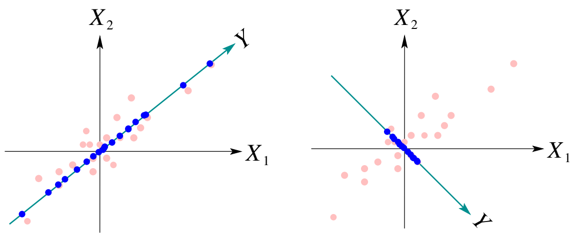


Project each sample in 3D onto a 2D plane

Orthogonal projection of data onto an axis



Optimal projection of 2D data onto 1D



- Mapping 2D to 1D: $y_n = \mathbf{u}^T \mathbf{x}_n = u_1 x_{n1} + u_2 x_{n2}$
- Optimal mapping: $\max_{\mathbf{u}} \mathrm{Var}(y)$

  $\mathrm{Var}(y) = \frac{1}{N-1}\sum_{n=1}^{N}(y_n - \bar{y})^2$
- cf. least squares fitting (linear regression)

Principal Component Analysis (PCA)

- Mapping D-dimensional data to a *principal component axis* $\mathbf{u} = (u_1, \ldots, u_D)^T$ that maximises $\mathrm{Var}(y)$:

  $y_n = \mathbf{u}^T \mathbf{x}_n = u_1 x_{n1} + \cdots + u_D x_{nD}$ NB: $\|\mathbf{u}\| = 1$
- $\mathbf{u}$ is given as the eigenvector with the largest eigenvalue of the covariance matrix, S:

$$S = \frac{1}{N-1}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T, \quad \bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

- Eigen values λ_i and eigenvectors $\mathbf{p}_i$ of S:

  $S\,\mathbf{p}_i = \lambda_i\,\mathbf{p}_i, \quad i = 1, \ldots, D$

  If $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_D$, then $\mathbf{u} = \mathbf{p}_1$, and $\mathrm{Var}(y) = \lambda_1$

NB: $\mathbf{p}_i^T \mathbf{p}_j = 0$, i.e. $\mathbf{p}_i \perp \mathbf{p}_j$ for $i \neq j$
$\mathbf{p}_i$ is normally normalised so that $\|\mathbf{p}_i\| = 1$.

Covariance matrix

$$S = \begin{pmatrix} s_{11} & \cdots & s_{1D} \\ \vdots & \ddots & \vdots \\ s_{D1} & \cdots & s_{DD} \end{pmatrix} \quad \cdots \text{ D-by-D symmetric matrix}$$

- In scalar representation:

$$s_{ij} = \frac{1}{N-1}\sum_{n=1}^{N}(x_{ni}-\bar{x}_i)(x_{nj}-\bar{x}_j), \qquad \bar{x}_i = \frac{1}{N}\sum_{n=1}^{N} x_{ni}$$

- Relation with Pearson's correlation coefficient:

$$\begin{aligned} r_{ij} &= \frac{1}{N-1}\sum_{n=1}^{N}\left(\frac{x_{ni}-\bar{x}_i}{s_i}\right)\left(\frac{x_{nj}-\bar{x}_j}{s_j}\right) \\ &= \frac{1}{s_i s_j}\frac{1}{N-1}\sum_{n=1}^{N}(x_{ni}-\bar{x}_i)(x_{nj}-\bar{x}_j) \\ &= \frac{s_{ij}}{\sqrt{s_{ii}s_{jj}}} \qquad \text{cf: } s_i = \sqrt{s_{ii}} = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(x_{ni}-\bar{x}_i)^2} \end{aligned}$$

Principal Component Analysis (PCA) *(cont.)*

- Let $\mathbf{v} = \mathbf{p}_2$, i.e. the eigenvector for the second largest eiven values, λ_2
- Map $\mathbf{x}_n$ on to the axis by $\mathbf{v}$:

$$z_n = \mathbf{v}^T\mathbf{x}_n = v_1 x_{n1} + \cdots + v_D x_{nD}$$

- Point $(y_n, z_n)^T$ in $\mathcal{R}^2$ is the projection of $\mathbf{x}_n \in \mathcal{R}^D$ on the 2D plane spanned by $\mathbf{u}$ and $\mathbf{v}$.

$$\mathrm{Var}(y) = \lambda_1, \quad \mathrm{Var}(z) = \lambda_2$$

- Can be generalised to a mapping from $\mathcal{R}^D$ to $\mathcal{R}^\ell$ using $\{\mathbf{p}_1, \ldots, \mathbf{p}_\ell\}$, where $\ell < D$.
- NB: Dimensionality reduction may involve loss of information. Some informaation will be lost if

$$\frac{\sum_{i=1}^{\ell}\lambda_i}{\sum_{i=1}^{D}\lambda_i} < 1$$

PCA on the film review toy data

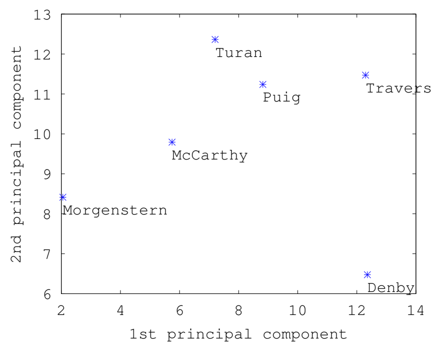
	Australia	*Body of Lies*	*Burn After*	*Hancock*	*Milk*	*Rev Road*
Denby	3	7	4	9	9	7
McCarthy	7	5	5	3	8	8
M'stern	7	5	5	0	8	4
Puig	5	6	8	5	9	8
Travers	5	8	8	8	10	9
Turan	7	7	8	4	7	8

$$S = \begin{pmatrix} 2.66 & -1.07 & 0.53 & -4.67 & -1.20 & -0.67 \\ -1.07 & 1.47 & 1.07 & 3.27 & 0.60 & 1.27 \\ 0.53 & 1.07 & 3.47 & 0.67 & 0.20 & 1.87 \\ -4.67 & 3.27 & 0.67 & 10.97 & 2.30 & 3.67 \\ -1.20 & 0.60 & 0.20 & 2.30 & 1.10 & 0.60 \\ -0.67 & 1.27 & 1.87 & 3.67 & 0.60 & 3.07 \end{pmatrix} \quad P = \begin{pmatrix} -0.341 & 0.345 & 0.326 & -0.180 & 0.603 & -0.512 \\ 0.255 & 0.151 & -0.240 & -0.548 & 0.496 & 0.554 \\ 0.101 & 0.786 & -0.503 & 0.028 & -0.280 & -0.198 \\ 0.827 & -0.154 & 0.096 & -0.182 & 0.025 & -0.450 \\ 0.181 & -0.065 & -0.341 & 0.733 & 0.556 & 0.015 \\ 0.304 & 0.461 & 0.676 & 0.309 & -0.047 & 0.375 \end{pmatrix}$$

$$Q = \begin{pmatrix} 15.8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4.85 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.13 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.634 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.288 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

where $P = (\mathbf{p}_1, \ldots, \mathbf{p}_6)$ and $(Q)_{ii} = \lambda_i$ for $i = 1, \ldots, 6$

PCA on the film review toy data (*cont.*)



Dimensionality reduction $D \to \ell$ by PCA

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_\ell \end{pmatrix} = \begin{pmatrix} \mathbf{p}_1^T\mathbf{x} \\ \mathbf{p}_2^T\mathbf{x} \\ \vdots \\ \mathbf{p}_\ell^T\mathbf{x} \end{pmatrix} = \begin{pmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \vdots \\ \mathbf{p}_\ell^T \end{pmatrix}\mathbf{x}$$

where $\{\mathbf{p}_i\}_{i=1}^{\ell}$ are the eigenvectors for the ℓ largest eigenvalues of S. The above can be rewritten as

$$\mathbf{y} = A^T\mathbf{x} \qquad \cdots \text{ linear transformation from } R^D \text{ to } R^\ell$$

$$\mathbf{y} = (y_1, \ldots, y_\ell)^T : \quad \ell\text{-dimensional vector}$$
$$A = (\mathbf{p}_1, \ldots, \mathbf{p}_\ell) : \quad D \times \ell \text{ matrix}$$

In many applications, we normalise data before PCA, e.g. $\mathbf{y} = A^T(\mathbf{x} - \bar{\mathbf{x}})$.

Summary

- **Clustering**
  K-means for minimising 'cluster variance'
  Review notes, *not just slides*
  [other methods exist: hierarchical, top-down and bottom-up]
- **Unsupervised learning**
  Spot structure in unlabelled data
  Combine with knowledge of task
- **Principal Component Analysis (PCA)**
  Find principal component axes for dimensionality reduction and visualisation
- Try implementing the algorithms! (Lab 3 in Week 4)

Further reading *(NE)*

- Rui Xu, D. Wunsch, "Survey of clustering algorithms," in IEEE Transactions on Neural Networks, vol. 16, no. 3, pp. 645-678, May 2005.
  https://doi.org/10.1109/TNN.2005.845141
- Dongkuan Xu, Yingjie Tian, "A Comprehensive Survey of Clustering Algorithms," Annals of Data Science, 2015, Volume 2, Number 2, Page 165.
  https://doi.org/10.1007/s40745-015-0040-1
- C. Bishop, "Pattern Recognition and Machine Learning," Chapter 12.1 (PCA).
  https://www.microsoft.com/en-us/research/people/cmbishop/prml-book/
- C.O.S. Sorzano, J. Vargas, A. Pascual Montano, "A survey of dimensionality reduction techniques," 2014.
  https://arxiv.org/abs/1403.2877

Quizes

Q1: Find computational complexity of k-means algorithm

Q2: For k-means clustering, discuss possible methods for mitigating the local minimum problem.

Q3: Discuss possible problems with k-means clustering and solutions when the variances of data (i.e. s_i, $i = 1, \ldots, D$) are much different from each other.

Q4: For k-means clustering, show $E^{(t+1)} \le E^{(t)}$. *(NE)*

Q5: At page 37, show $\mathbf{y} = \mathbf{u}^T\mathbf{x}$.

Q6: At page 39, show $\mathrm{Var}(y) = \lambda_1$, where λ_1 is the largest eigenvalue of S. *(NE)*

Q7: The first principal component axis is sometimes confused with the line of least squares fitting (or regression line). Explain the difference.

Inf2b - Learning

Lecture 4: Classification and nearest neighbours

Hiroshi Shimodaira
(Credit: Iain Murray and Steve Renals)

Centre for Speech Technology Research (CSTR)
School of Informatics
University of Edinburgh

http://www.inf.ed.ac.uk/teaching/courses/inf2b/
https://piazza.com/ed.ac.uk/spring2020/infr08028
Office hours: Wednesdays at 14:00-15:00 in IF-3.04

Jan-Mar 2020

Today's topics

1. Classification
2. Nearest neighbour classification
3. Decision boundary
4. Tips on pre-processing data
5. Generalisation and over-fitting

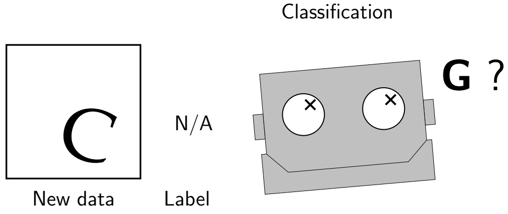
Types of learning problems

Data	System: input	System: output	Type of problem	Type of learning
$\mathbf{x}$	$\{\mathbf{x}\}$	groups (subsets)	clustering	unsupervised learning
$(\mathbf{x}, y)$	$\mathbf{x}$	y: discrete category	classification	supervised learning
$(\mathbf{x}, y)$	$\mathbf{x}$	y: continuous value	regression	supervised learning

where $\mathbf{x} = (x_1, \ldots, x_D)^T$: feature vector
y : target vector or scalar

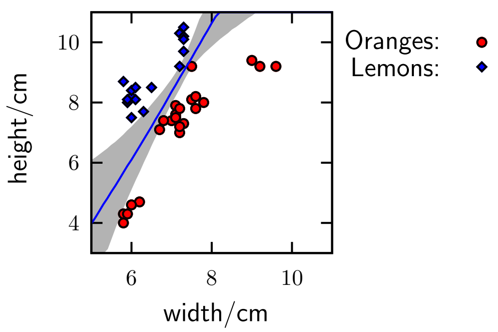
Supervised learning

Test mode



Goal of training: develop a classifier of good generalisation

Supervised learning



Classification

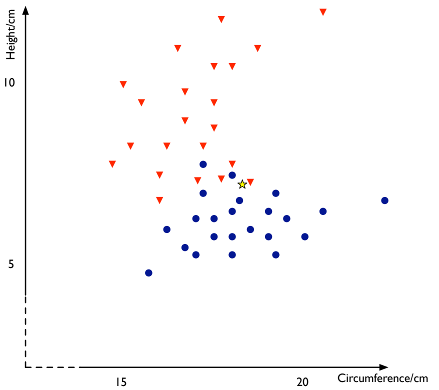
- The data has a feature vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_D)^T$ and a label $c \in \{1, \ldots, C\}$
- Training set: A set of N feature vectors and their labels $(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_N, c_N)$
- Use a learning algorithm to train a classifier from a training set
- Test set: a set of feature vectors to which the classifier must assign labels – used for evaluation. (NB: training and test sets should be mutually exclusive)
- Error function: how accurate is the classifier? One option is to count the number of misclassifications:

$$\text{Error rate} = \frac{\#\text{ of misclassified samples}}{\#\text{ of test samples}}$$

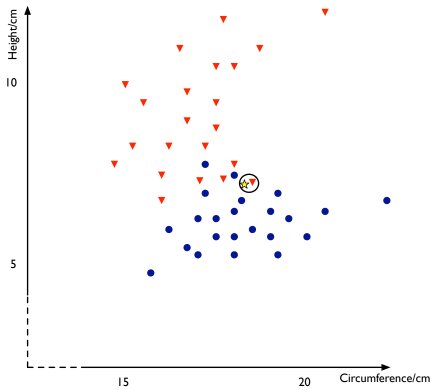
Nearest-neighbour classifier

- Nearest neighbour classification: label a test example to have the label of the closest training example
- K-nearest neighbour (K-NN) classification: find the K closest points in the training set to the test example; classify using a majority vote of the K class labels
- Training a K-nearest neighbour classifier is simple! — Just store the training set
- Classifying a test example requires finding the K closest training examples
  - This is computationally demanding if the training set is large — potentially need to compute the Euclidean distance between the test example and every training example
  - Data structures such as the kD-tree can make finding nearest neighbours much more efficient (in the average case)

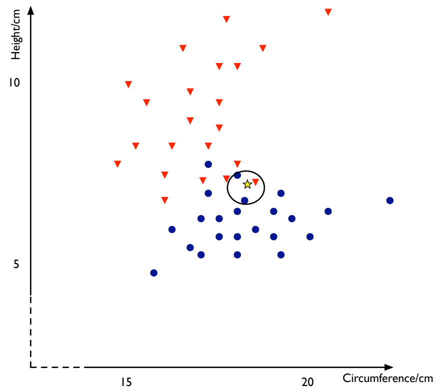
Classifying test data with K-nearest neighbours



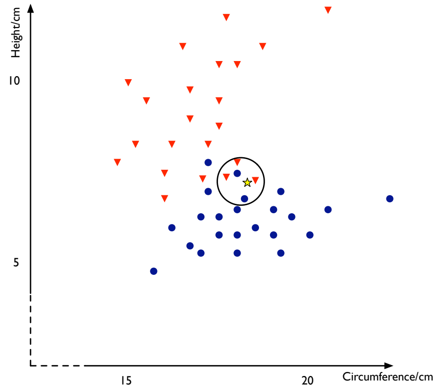
1-nearest neighbour



3-nearest neighbour



5-nearest neighbour



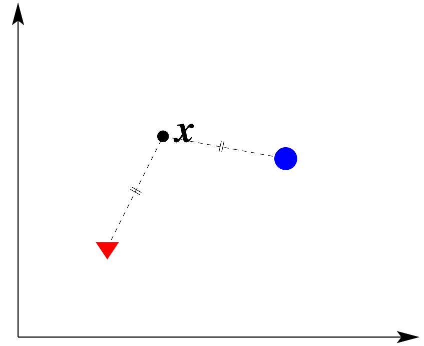
K-NN classification algorithm

For each test example $\mathbf{z} \in Z$:

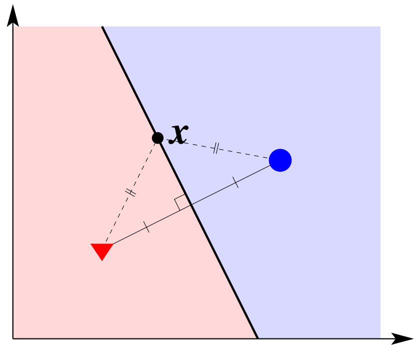
- Compute the distance $r(\mathbf{z}, \mathbf{x})$ between $\mathbf{z}$ and each training example $(\mathbf{x}, c) \in X$
- Select $U_k(\mathbf{z}) \subseteq X$, the set of the k nearest training examples to $\mathbf{z}$
- Decide the class of $\mathbf{z}$ by the majority voting:

$$c(\mathbf{z}) = \underset{j \in \{1,\dots,C\}}{\arg\max} \sum_{(\mathbf{x},c) \in U_k(\mathbf{z})} \delta_{j\,c}$$

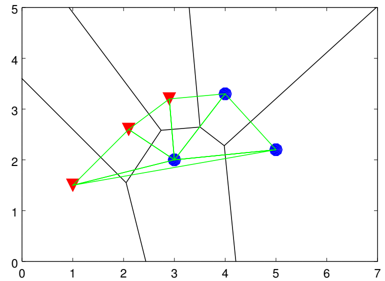
Geometry of nearest neighbour



Geometry of nearest neighbour – decision boundary and decision regions

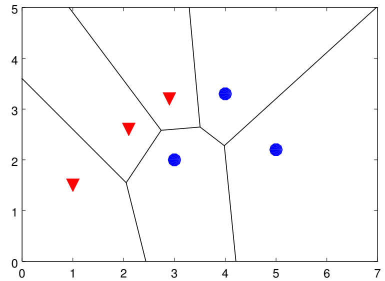


Geometry of nearest neighbour



Delaunay triangulation

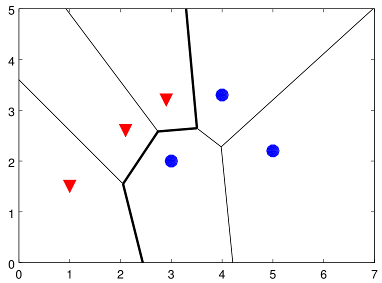
Voronoi tessellation



Voronoi diagram

Decision boundary

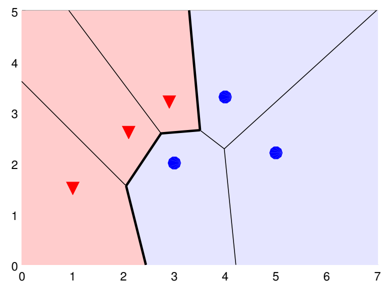
Decision boundary: boundary (surface) that partitions the vector space into subsets of different classes.



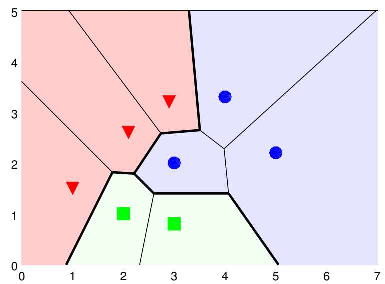
K-NN classification forms *piecewise-linear decision boundary*.

Decision regions

Decision regions: regions separated by the decision boundaries

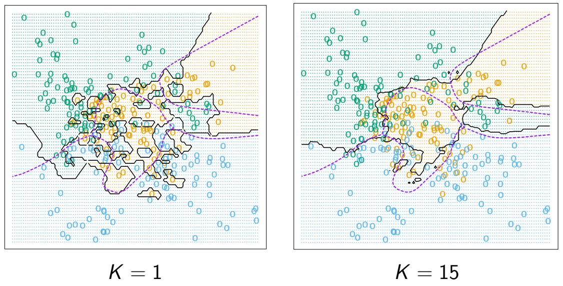


Decision boundaries for $C = 3$



What K should we use?

An example where a large K reduces noise

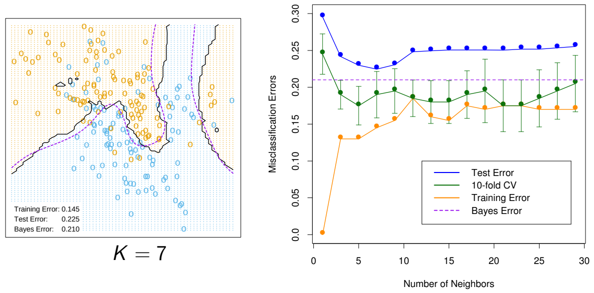


(Black curve: KNN decision boundary, broken purple curve: the Bayes decision boundary

The Elements of Statistical Learning (2nd Ed.)
Hastie, Tibshirani, Friedman. §13.3 p463–

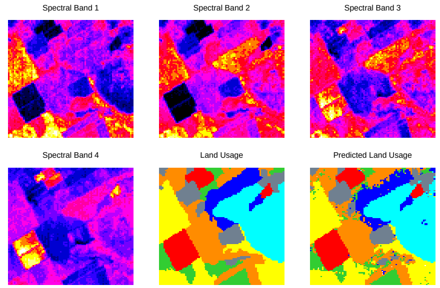
http://statweb.stanford.edu/~tibs/ElemStatLearn/

Learning curves



The Elements of Statistical Learning (2nd Ed.)
Hastie, Tibshirani, Friedman. §13.3 p463–

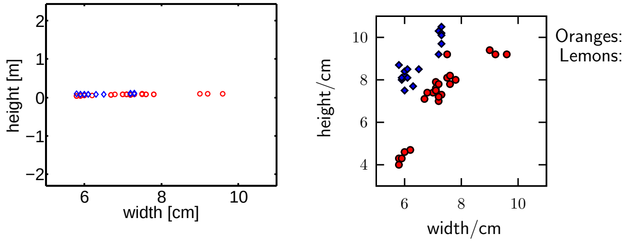
LANDSAT Application



Predict land-usage from satellite data
KNN applied to 9 pixel patch in 4 spectral bands, with $K = 5$

The Elements of Statistical Learning (2nd Ed.)
Hastie, Tibshirani, Friedman. §13.3 p463–

Tips on pre-processing data



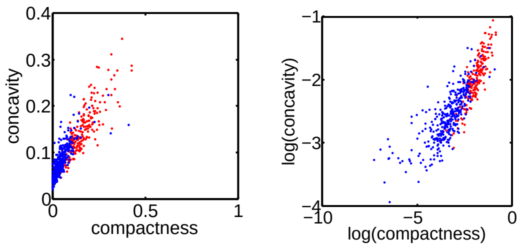
different units same unit

$\Rightarrow$ Standardise features unless understand units

Tips on pre-processing data

Wisconsin Diagnostic Breast Cancer (WDBC) data set

http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

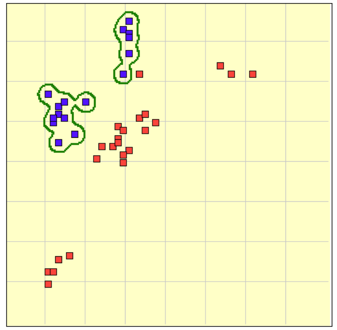


Linear scale Log scale

$\Rightarrow$ Consider transformation, e.g. log-transform.

Generalisation and over-fitting

How reasonable is this decision boundary?



Poor generalisation: stories

In a competition:
http://blog.kaggle.com/2012/07/06/
the-dangers-of-overfitting-psychopathy-post-mortem/

Classic stories:
http://neil.fraser.name/writing/tank/

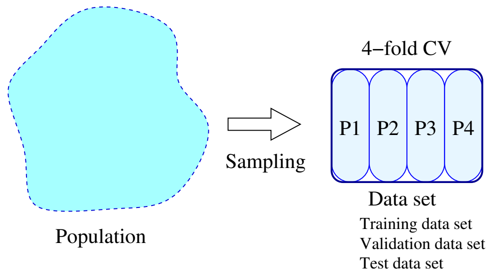
http://www.j-paine.org/dobbs/neural_net_urban_legends.html

How reliable is the error rate?

- Error rate on training data set:
  can be $\sim$ 0%
  $\Rightarrow$ useless to estimate generalisation error
- Error rate on a test data set (exclusive to the training set)
  - How large should the data set be?
  - How should it be collected?

  Cross validation is used to estimate generalisation error
  (swapping test and training data sets)
  - k-fold cross validation (k-fold CV)
    (2-fold CV is sometimes called 'holdout method')
  - leave-one-out cross validation (LOO CV)

Cross validation



Summary

- **Classification with similarity based methods**
  - Represent items as feature vectors
  - Compute distances to other items and sort
  - Assign a class label to the feature vector
  - k-NN: an example-based approach that classifies a test point based on the classes of the closet training samples
  - Larger k results in a smoother solution
  - Decision boundaries/regions, Voronoi diagram
- **Generalisation**
  - Overfitting: tuning a classifier to closely to the training set can reduce accuracy on the test set
  - Compare methods on held out data (validation set)
  - Estimate final performance on *really* new data (test set)

Further reading (NE)

- L. Jiang, Z. Cai, D. Wang, S. Jiang, "Survey of Improving K-Nearest-Neighbor for Classification," Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)
- M.R. Abbasifard, B. Ghahremani, H. Naderi, "A Survey on Nearest Neighbor Search Methods," International Journal of Computer Applications (0975 – 8887), Vol.95, No.25, June 2014.
- Hand-Drawn Voronoi Diagrams
- Roberto Tamassia, "Introduction to Voronoi Diagrams," Lecture notes of C.S. 252, Computational Geometry, University of Brown, 1993.
- Steven Fortune, "A sweepline algorithm for Voronoi diagrams," Algorithmica 2, 153 (1987).

Labs

04th, 05th Feb.	Lab-3	K-means clustering and PCA
11th, 12th Feb.	Lab-4	K-NN classification

Inf2b - Learning

Lecture 5: Introduction to statistical pattern recognition and Optimisation

Hiroshi Shimodaira
(Credit: Iain Murray and Steve Renals)

Centre for Speech Technology Research (CSTR)
School of Informatics
University of Edinburgh

http://www.inf.ed.ac.uk/teaching/courses/inf2b/
https://piazza.com/ed.ac.uk/spring2020/infr08028
Office hours: Wednesdays at 14:00-15:00 in IF-3.04

Jan-Mar 2020

Today's Schedule

1. Probability (review)
2. What is Bayes' theorem for?
3. Bayes decision rule
4. More about probability
5. Optimisation problems

Motivation for probability

In some applications we need to:

- Communicate uncertainty
- Use prior knowledge
- Deal with missing data

  (we cannot easily measure similarity)

Warming up

- Throwing two dices
  - Probability of $\{1, 1\}$?
  - Probability of $\{2, 5\}$?
- Drawing two cards from a deck of cards
  - Probability of $\{\text{Club}, \text{Spade}\}$?
  - Probability of $\{\text{Club}, \text{Club}\}$?

Warming up (*cont.*)

- Probability that a student in Informatics has eyeglasses?
- Probability that you live more than 90 years?
- When a real dice is thrown, is the probability of getting $\{1\}$ $\frac{1}{6}$?

Theoretical probability vs. Empirical probability
aka:
relative frequency
experimental probability
for a sample set drawn from a population

Rules of Probability

Random variables	Events/values
X	$\{x_1, x_2, \ldots, x_L\}$
Y	$\{y_1, y_2, \ldots, y_M\}$

Product Rule:

$$P(Y=y_j, X=x_i) = P(Y=y_j \mid X=x_i)\, P(X=x_i)$$
$$= P(X=x_i \mid Y=y_j)\, P(Y=y_j)$$

Abbreviation:

$$P(Y, X) = P(Y \mid X)\, P(X)$$
$$= P(X \mid Y)\, P(Y)$$

X and Y are *independent* iff:

$$P(X, Y) = P(X)\, P(Y)$$

$$P(X|Y) = P(X), \quad P(Y|X) = P(Y)$$

Rules of Probability (*cont.*)

Sum Rule:

$$P(X=x_i) = \sum_{j=1}^{M} P(X=x_i, Y=y_j)$$

Abbreviation:

$$P(X) = \sum_{Y} P(X, Y)$$

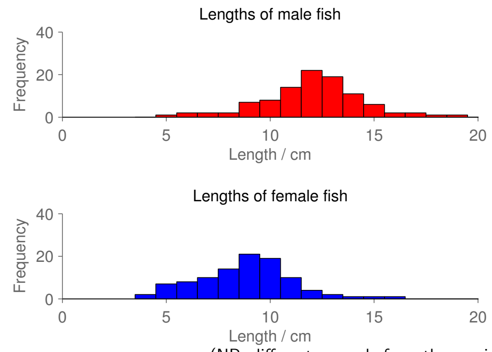
RHS: *Mariginalisation* of the joint probability over Y.
LHS: *Marginal probability* of X.

Application:

$$P(X) = \sum_{Y} P(X \mid Y)\, P(Y)$$

Example: determining the sex of fish

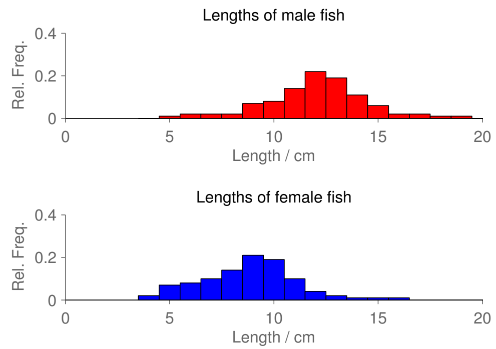
Histograms of fish lengths ($N_F = N_M = 100$)



(NB: different example from the one in Note 5.)

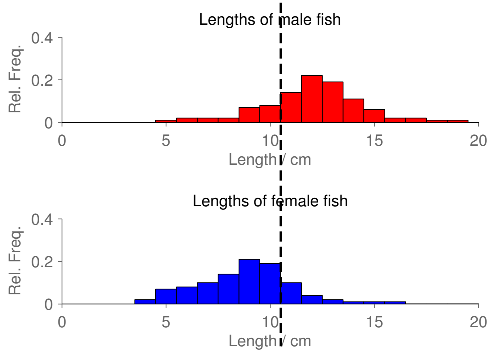
Example: determining the sex of fish

Relative frequencies of fish length



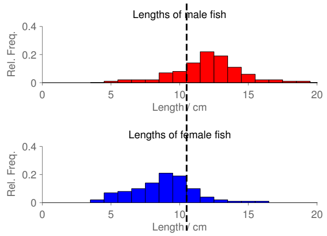
Example: determining the sex of fish

Possible decision boundary



Fish questions

- How to classify 4 cm, or 19 cm fish?
- How to classify 10 cm fish?

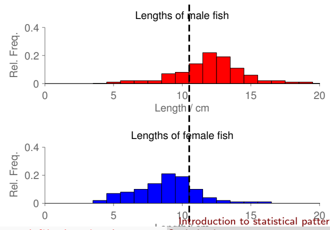


Fish questions

Relative frequency of male fish length:	$P(x \mid \mathrm{M})$
Relative frequency of female fish length:	$P(x \mid \mathrm{F})$

Given a fish length, **x**, is it sensible to decide as follows?

If $P(x \mid \mathrm{M}) > P(x \mid \mathrm{F}) \;\Rightarrow$ male fish
If $P(x \mid \mathrm{M}) < P(x \mid \mathrm{F}) \;\Rightarrow$ female fish



Fish questions (*cont.*)

How to obtain $P(Y \mid x)$? (where $Y = \{F, M\}$)

- The product rule:
$$P(Y, x) = P(Y \mid x)\, P(x) = P(x \mid Y)\, P(Y)$$
- Posterior probabilities:
$$P(Y \mid x) = \frac{P(x \mid Y)\, P(Y)}{P(x)} \propto P(x \mid Y)\, P(Y)$$

i.e.

$$P(M \mid x) = \frac{P(x \mid M)\, P(M)}{P(x)} \propto P(x \mid M)\, P(M)$$

$$P(F \mid x) = \frac{P(x \mid F)\, P(F)}{P(x)} \propto P(x \mid F)\, P(F)$$

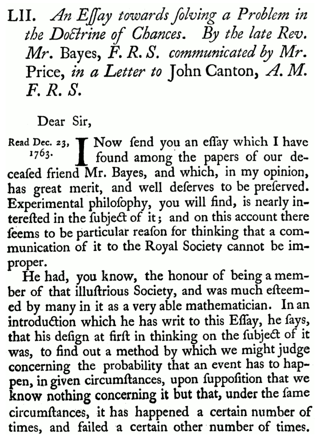
Bayes' Theorem

$$P(H \mid E) = \frac{P(E \mid H)\, P(H)}{P(E)}$$

Thomas Bayes (?) (1702? – 1761)

http://www.york.ac.uk/depts/maths/histstat/bayespic.htm

c.f. Bayesian inference, Bayesian



LII. *An Eſſay towards ſolving a Problem in the Doctrine of Chances. By the late Rev. Mr.* Bayes, *F. R. S. communicated by Mr.* Price, *in a Letter to* John Canton, *A. M. F. R. S.*

Dear Sir,

Read Dec. 23, 1763.

I Now ſend you an eſſay which I have found among the papers of our deceaſed friend Mr. Bayes, and which, in my opinion, has great merit, and well deſerves to be preſerved. Experimental philoſophy, you will find, is nearly intereſted in the ſubject of it; and on this account there ſeems to be particular reaſon for thinking that a communication of it to the Royal Society cannot be improper.

He had, you know, the honour of being a member of that illuſtrious Society, and was much eſteemed by many in it as a very able mathematician. In an introduction which he has writ to this Eſſay, he ſays, that his deſign at firſt in thinking on the ſubject of it was, to find out a method by which we might judge concerning the probability that an event has to happen, in given circumſtances, upon ſuppoſition that we know nothing concerning it but that, under the ſame circumſtances, it has happened a certain number of times, and failed a certain other number of times.

'Bayesian' philosophy refs

Non-examinable!

Bayes' paper:
http://www.jstor.org/stable/105741
http://dx.doi.org/10.1093/biomet/45.3-4.296 (re-typeset)

Cox's paper:
http://dx.doi.org/10.1119/1.1990764
http://dx.doi.org/10.1016/S0888-613X(03)00051-3 modern commentary

MacKay textbook, amongst many others

Bayes decision rule

Class $C = \{1, \ldots, K\}$; C_k to denote $C = k$; input features $X = \mathbf{x}$

Choose the most probable class: (maximum posterior class)

$$k_{\max} = \arg\max_{k \in C} P(C_k \mid \mathbf{x}) = \arg\max_{k} P(\mathbf{x} \mid C_k)\, P(C_k)$$

where

$$\overbrace{P(C_k \mid \mathbf{x})}^{\text{posterior}} = \frac{\overbrace{P(\mathbf{x} \mid C_k)}^{\text{likelihood}}\ \overbrace{P(C_k)}^{\text{prior}}}{P(\mathbf{x})} = \frac{P(\mathbf{x} \mid C_k)\, P(C_k)}{\sum_{j=1}^{K} P(\mathbf{x} \mid C_j)\, P(C_j)}$$

$\Rightarrow$

- It is known this decision rule gives minimum error rate. (We will discuss this in Lecture 10)
- Also called
  - Minimum error (misclassification) rate classification (PRML C. M. Bishop (2006) Section 1.5)
  - Maximum posterior probability (MAP) decision rule

Inferring labels for $x = 11$

- **Equal prior probabilities:**

$$P(M \mid x=11) = \frac{P(x=11 \mid M)\, P(M)}{P(x=11)}$$
$$= \frac{P(x=11 \mid M) P(M)}{P(x=11 \mid M)\, P(M) + P(x=11 \mid F) P(F)}$$
$$= \frac{0.14 \cdot 0.5}{0.14 \cdot 0.5 + 0.10 \cdot 0.5} = \frac{0.14}{0.24} = 0.58\dot{3}$$

$$P(F \mid x=11) = \frac{P(x=11 \mid F)\, P(F)}{P(x=11 \mid M)\, P(M) + P(x=11 \mid F)\, P(F)}$$
$$= \frac{0.10 \cdot 0.5}{0.14 \cdot 0.5 + 0.10 \cdot 0.5} = \frac{0.10}{0.24} = 0.41\dot{6}$$

$\rightarrow$ classify it as male

NB: For classification, no need to calculate $P(x = 11)$.

Inferring labels for $x = 11$ (*cont.*)

- **Equal prior probabilities:**

$$\frac{P(M \mid x=11)}{P(F \mid x=11)} = \frac{P(x=11 \mid M)\, P(M)}{P(x=11 \mid F)\, P(F)} = \frac{0.14 \cdot 0.5}{0.10 \cdot 0.5} = 1.4$$

Classify it as male:

- **Twice as many females as males:** (i.e., $P(M) = 1/3$, $P(F) = 2/3$)

$$\frac{P(M \mid x=11)}{P(F \mid x=11)} = \frac{P(x=11 \mid M)\, P(M)}{P(x=11 \mid F)\, P(F)} = \frac{0.14 \cdot 1/3}{0.10 \cdot 2/3} = 0.7$$

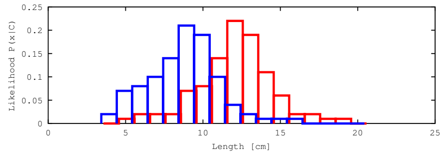
Classify it as female

Likelihood vs posterior probability

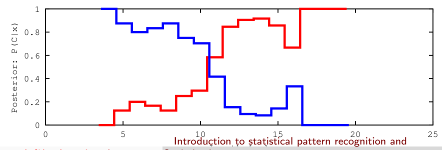
$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)} = \frac{P(\mathbf{x} \mid C_k)\, P(C_k)}{\sum_{j=1}^{K} P(\mathbf{x} \mid C_j)\, P(C_j)}$$

$$P(M) : P(F) = 1 : 1$$

$P(x|C_k)$



$P(C_k|x)$

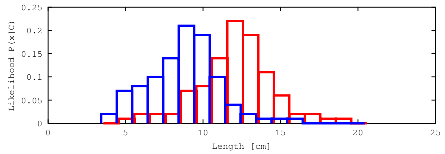


Likelihood vs posterior probability (*cont.*)

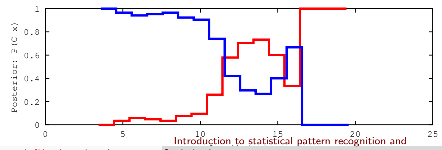
$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)} = \frac{P(\mathbf{x} \mid C_k)\, P(C_k)}{\sum_{j=1}^{K} P(\mathbf{x} \mid C_j)\, P(C_j)}$$

$$P(M) : P(F) = 1 : 4$$

$P(x|C_k)$



$P(C_k|x)$



Some more questions

- Assume $P(M) = P(F) = 0.5$
  1. What is the value of $P(M \mid X{=}4)$?
  2. What is the value of $P(F \mid X{=}18)$?
  3. You observe data point $x{=}22$. To which class should it be assigned?
- Discuss how you could improve classification performance.
  - What if we increase the number of histogram bins?
  - What if we increase the number of samples?
  - What if we measure not only fish length but also weight? (How can we estimate probabilities?)
- It seems that we can estimate $P(C|x)$ directly from data, right?

More about probability

- Conditional probability of three variables

$$P(X, Y \mid Z) = \frac{P(Y, Z \mid X)\, P(X)}{P(Z)}$$

$$P(X \mid Y, Z) = \frac{P(Z \mid Y, X)\, P(X \mid Y)}{P(Z \mid Y)}$$

- Chain rule

$$P(X_1, X_2, \ldots, X_N) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)\cdots \cdots P(X_N|X_1, \ldots, X_{N-1})$$

Prove!

Independence vs zero correlation

- Independence vs Pearson correlation coefficient $\rho = 0$

  If X and Y are independent, $\rho_{XY} = 0$.

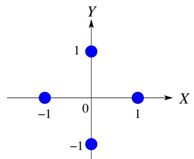
  The converse is not true.

  See https://en.wikipedia.org/wiki/Correlation_and_dependence

E.g. $(X, Y) = (-1, 0), (0, -1), (0, 1), (1, 0)$, each of which occurs with a probability of $\frac{1}{4}$.

$P(X{=}-1)\, P(Y{=}0) = 1/4 \cdot 1/2 = 1/8$
$P(X{=}0)\, P(Y{=}-1) = 1/2 \cdot 1/4 = 1/8$
$P(X{=}0)\, P(Y{=}1) = 1/2 \cdot 1/4 = 1/8$
$P(X{=}1)\, P(Y{=}0) = 1/4 \cdot 1/2 = 1/8$

$\rho_{XY} = 0$, but $P(X, Y) \neq P(X)\, P(Y)$
i.e., not independent



Optimisation problems we've seen

- Bayes decision rule (MAP decision rule)
  $$k_{\max} = \arg\max_{k \in C} P(C_k \,|\, \mathbf{x})$$
- K-NN classification
  $$c(\mathbf{z}) = \arg\max_{j \in \{1,\ldots,C\}} \sum_{(\mathbf{x},c) \in U_k(\mathbf{z})} \delta_{jc}$$
  where $U_k(\mathbf{z})$ is the set of k nearest training examples to $\mathbf{z}$.
- K-means clustering
  $$\min_{\{\mathbf{m}_k\}_1^K} E$$
  $$\text{where } E = \frac{1}{N}\sum_{k=1}^{K}\sum_{n=1}^{N} z_{kn} \|\mathbf{x}_n - \mathbf{m}_k\|^2$$
- Dimensionality reduction to 2D with PCA
  $$\max_{\mathbf{u},\mathbf{v}} \mathrm{Var}(y) + \mathrm{Var}(z)$$
  $$\text{subject to } \|\mathbf{u}\| = 1, \|\mathbf{v}\| = 1, \mathbf{u} \perp \mathbf{v}$$

Optimisation problems : other examples

- Find the shortest path between Edinburgh and London
- Find the cheapest flights from Edinburgh to Tokyo
- For UG4 projects, find the optimal allocation of supervisors and students under given constraints (e.g. no supervisors can take more than five students.)

Types of optimisation problems

- Continuous vs Discrete optimisation
- Unconstrained vs Constrained optimisation

https://neos-guide.org/optimization-tree
https://en.wikipedia.org/wiki/Optimization_problem

Continuous & unconstrained optimisation problems

Minimisation of *objective function*

$$\min_{\mathbf{x}} f(\mathbf{x}) \qquad \text{where } \mathbf{x} \in \mathcal{R}^D,\ f : \mathcal{R}^D \to \mathcal{R}$$

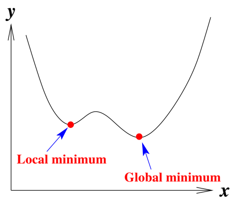
Optimal solution, $\mathbf{x}^* : f(\mathbf{x}^*) \le f(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{R}^D$, satisfies †

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = 0, \text{ for } i = 1, \ldots, D$$

Vector rerpresentation:

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_D}\right)^T = \mathbf{0}$$

where $\mathbf{0} = (0, \ldots, 0)^T$



† This is not a sufficient condition, but a necessary condition.

Optimisation of a quadratic function of one variable

Optimisation problem:

$$\min_{x} f(x)$$
$$f(x) = ax^2 + bx + c, \quad a > 0$$

- Approach 1:
  $$ax^2 + bx + c = a\left(x + \frac{b}{2a}\right)^2 - \frac{b^2}{4a} + c$$
- Approach 2:
  $$\frac{\mathrm{d}f(x)}{\mathrm{d}x} = 2ax + b = 0$$
- Solution: $x = -\dfrac{b}{2a}$

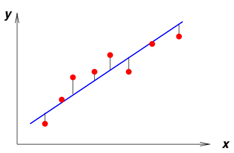
Optimisation of a quadratic function of two variables

- Optimisation problem:
  $$\min_{\{x,y\}} g(x,y)$$
  $$g(x,y) = ax^2 + by^2 + cxy + dx + ey + f$$
  where $a > 0,\ b > 0,\ c^2 < 4ab$

$\rightarrow$
$$\frac{\partial g}{\partial x} = 2ax + cy + d = 0$$
$$\frac{\partial g}{\partial y} = 2by + cx + e = 0$$
$$\begin{pmatrix} 2a & c \\ c & 2b \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -d \\ -e \end{pmatrix}$$

Least square error line fitting

- Optimisation problem
  $$\min_{a,b} \frac{1}{N}\sum_{n=1}^{N} (\hat{y}_n - y_n)^2$$
  $$\hat{y}_n = ax_n + b$$



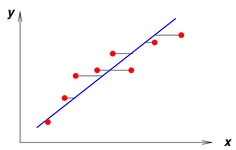
$\rightarrow$
$$\frac{\partial E}{\partial a} = \frac{2}{N}\sum_{n=1}^{N}(ax_n + b - y_n)x_n = 0$$
$$\frac{\partial E}{\partial b} = \frac{2}{N}\sum_{n=1}^{N}(ax_n + b - y_n) = 0$$

$\Rightarrow$ See the lecture note for details.

Least square error line fitting (cont.)

Exercise:

- Optimisation problem
  $$\min_{c,d} \frac{1}{N}\sum_{n=1}^{N} (\hat{x}_n - x_n)^2$$
  $$\hat{x}_n = c\,y_n + d$$



Find the solution

Iterative optimisation

Many optimisation problems do not have a closed-form solution! (e.g. K-means clustering)

Iterative optimisation method

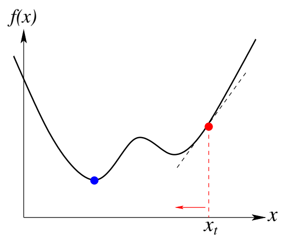
- Step 1: Choose an initial point $\mathbf{x}_0$, and make $t = 0$.
- Step 2: Choose $\mathbf{x}_{t+1}$ based on an update formula for $\mathbf{x}_t$.
- Step 3: $t \leftarrow t + 1$ and go to step 2 unless stopping criterion is met.

Example of iterative optimisation methods

- Gradient descent
  $$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \,\nabla f(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_t} \qquad \text{where } \eta > 0$$
- Conjugate gradient method
- Newton's method

Gradient descent

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_t} \qquad \text{where } \eta > 0$$



Things to consider:

- Choice of η (i.e. learning parameter)
- Local-minimum problem

Summary

- *Bayes' theorem* for statistical pattern classification
- Posterior is proportional to prior times likelihood
- $P(x)$ can be obtained with *marginalisation* of $P(x|C)P(C)$
- *Bayes decision rule* achieves minimum error rate classification
- Discuss possible difficulties of applying the Bayes' decision rule to real problems
- Pattern recognition as optimisation problem
- Most of optimisation problem does not have a closed-form solution $\rightarrow$ Iterative optimisation method
- Check the examples in slides, and try the exercises in Note 5.

Mid-course feedback

Your Learn course webpage
$\rightarrow$ (on the left black tab) Have Your Say
$\rightarrow$ Mid-course feedback

Inf2b - Learning

Lecture 6: Naive Bayes

Hiroshi Shimodaira
(Credit: Iain Murray and Steve Renals)

Centre for Speech Technology Research (CSTR)
School of Informatics
University of Edinburgh

http://www.inf.ed.ac.uk/teaching/courses/inf2b/
https://piazza.com/ed.ac.uk/spring2020/infr08028
Office hours: Wednesdays at 14:00-15:00 in IF-3.04

Jan-Mar 2020

Today's Schedule

1. Bayes decision rule review
2. The curse of dimensionality
3. Naive Bayes
4. Text classification using Naive Bayes (introduction)

Bayes decision rule (recap)

Class $C = \{1, \ldots, K\}$; C_k to denote $C = k$; input features $X = \mathbf{x}$

Most probable class: (maximum posterior class)

$$k_{\max} = \arg\max_{k \in C} P(C_k \,|\, \mathbf{x}) = \arg\max_k \frac{P(\mathbf{x} \,|\, C_k)\; P(C_k)}{\sum_{j=1}^{K} P(\mathbf{x}|C_j)P(C_j)}$$
$$= \arg\max_k \; P(\mathbf{x}|C_k)P(C_k)$$

where $P(C_k \,|\, \mathbf{x})$: posterior
$P(\mathbf{x} \,|\, C_k)$: likelihood
$P(C_k)$: prior

$\Rightarrow$ Minimum error (misclassification) rate classification

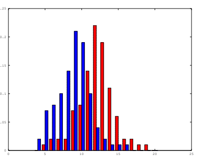
(PRML C. M. Bishop (2006) Section 1.5)

Fish classification (revisited)

Bayesian class estimation:

$$P(C_k \,|\, x) = \frac{P(x \,|\, C_k)\, P(C_k)}{P(x)} \propto P(x \,|\, C_k)\, P(C_k)$$

Estimating the terms: (Non-Bayesian)



Priors: $P(C = M) \approx \dfrac{N_M}{N_M + N_F}, \ldots$

Likelihoods: $P(x \,|\, C = M) \approx \dfrac{n_M(x)}{N_M}, \ldots$

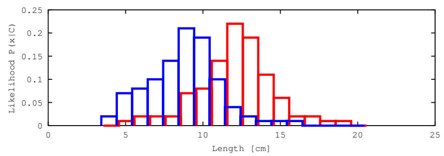
NB: These *approximations* work well only if we have enough data

Fish classification (revisited)

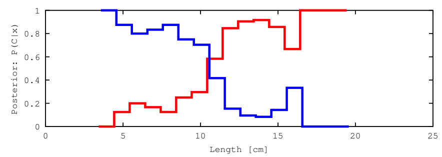
$$P(C_k \,|\, x) = \frac{P(x \,|\, C_k)\, P(C_k)}{P(x)}$$

$$P(M) : P(F) = 1 : 1$$

$P(x \,|\, C_k)$



$P(C_k \,|\, x)$

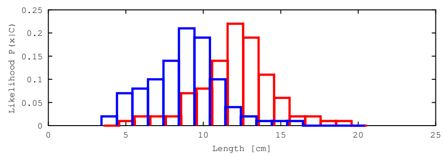


Fish classification (revisited)

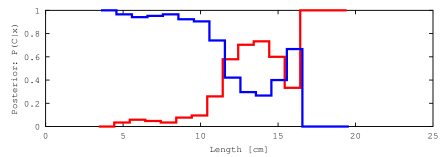
$$P(C_k \,|\, x) = \frac{P(x \,|\, C_k)\, P(C_k)}{P(x)}$$

$$P(M) : P(F) = 1 : 4$$

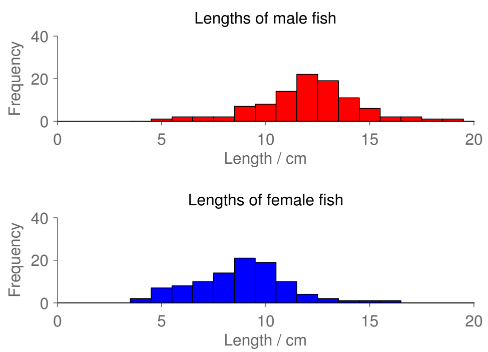
$P(x \,|\, C_k)$



$P(C_k \,|\, x)$



How can we improve the fish classification?



More features!?

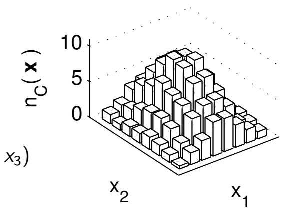
$$P(\mathbf{x} \mid C_k) \approx \frac{n_{C_k}(x_1, \ldots, x_D)}{N_{C_k}}$$

1D histogram: $n_{C_k}(x_1)$

2D histogram: $n_{C_k}(x_1, x_2)$

3D cube of numbers: $n_{C_k}(x_1, x_2, x_3)$

$\vdots$



100 binary variables, 2^{100} settings (the universe is $\approx 2^{98}$ picoseconds old)

In high dimensions almost all $n_{C_k}(x_1, \ldots, x_D)$ are zero

$\Rightarrow$ Bellman's "curse of dimensionality"

Avoiding the Curse of Dimensionality

Apply the chain rule?

$$\begin{aligned} P(\mathbf{x} \mid C_k) &= P(x_1, x_2, \ldots, x_D \mid C_k) \\ &= P(x_1|C_k)\, P(x_2|x_1, C_k)\, P(x_3|x_2, x_1, C_k)\, P(x_4|x_3, x_2, x_1, C_k) \cdots \\ &\quad \cdots P(x_{d-1}|x_{d-2}, \ldots, x_1, C_k)\, P(x_D|x_{D-1}, \ldots, x_1, C_k) \end{aligned}$$

Solution: assume structure in $P(x \mid C_k)$

For example,

- Assume x_{d+1} depends on x_d only
  $P(\mathbf{x} \mid C_k) \approx P(x_1|C_k) P(x_2|x_1, C_k) P(x_3|x_2, C_k) \cdots P(x_D|x_{D-1}, C_k)$
- Assume $x \in \mathcal{R}^D$ distributes in a low dimensional vector space
  - Dimensionality reduction by PCA (Principal Component Analysis) / KL-transform

Avoiding the Curse of Dimensionality (cont.)

- Apply smoothing windows (e.g. Parzen windows)
- Apply a probability distribution model (e.g. Normal dist.)
- Assume $x_1, \ldots, x_D$ are conditionally independent given class

  $\Rightarrow$ Naive Bayes rule/model/assumption
  (or *idiot Bayes rule*)

$$\begin{aligned} P(x_1, x_2, \ldots, x_D|C_k) &= P(x_1|C_k)\, P(x_2|C_k) \cdots P(x_D|C_k) \\ &= \prod_{d=1}^{D} P(x_d|C_k) \end{aligned}$$

- *Is it reasonable?*
  Often not, of course!
  Although it can still be *useful*.

Example - game played depending on the weather

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

$$P(Play \mid O, T, H, W) = \frac{P(O, T, H, W \mid Play)\, P(Play)}{P(O, T, H, W)}$$

Weather data - how to calculate probabilities?

$$P(Play \mid O, T, H, W) = \frac{P(O, T, H, W \mid Play)\, P(Play)}{P(O, T, H, W)}$$

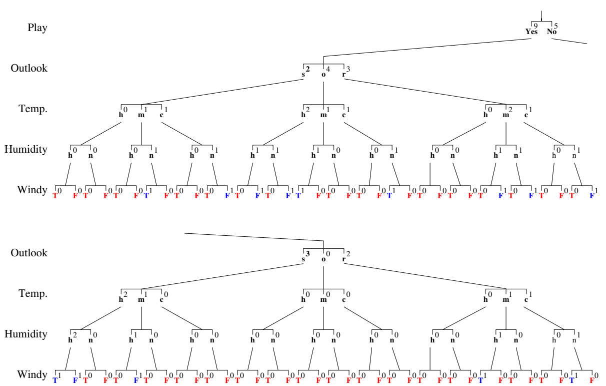
If we use histograms for this 4D data: $n_{Play}(O, T, H, W)$

$$\begin{array}{ccccccc} \text{Outlook} & & \text{Temp.} & & \text{Humidity} & & \text{Windy} \\ \begin{bmatrix} \text{sunny} \\ \text{overcast} \\ \text{rainy} \end{bmatrix} & \times & \begin{bmatrix} \text{hot} \\ \text{mild} \\ \text{cool} \end{bmatrix} & \times & \begin{bmatrix} \text{high} \\ \text{normal} \end{bmatrix} & \times & \begin{bmatrix} \text{true} \\ \text{false} \end{bmatrix} \end{array}$$

\# of bins in the histogram $= 3 \times 3 \times 2 \times 2 = 36$

\# of samples available $= 9$ for play:yes, 5 for play:no

Weather data - tree representation



Applying Naive Bayes

$$\begin{aligned} P(Play \mid O, T, H, W) &= \frac{P(O, T, H, W \mid Play)\, P(Play)}{P(O, T, H, W)} \\ &\propto P(O, T, H, W \mid Play)\, P(Play) \end{aligned}$$

Applying the Naive Bayes rule,

$$P(O, T, H, W \mid Play) = P(O|Play)\, P(T|Play)\, P(H|Play)\, P(W|Play)$$

Weather data summary

Counts

Outlook	Y	N	Temperature	Y	N	Humidity	Y	N	Windy	Y	N	Play Y	Play N
sunny	2	3	hot	2	2	high	3	4	f	6	2	9	5
overc	4	0	mild	4	2	norm	6	1	t	3	3		
rainy	3	2	cool	3	1								

Relative frequencies $P(x|Play = Y), P(x|Play = N)$

Outlook	Y	N	Temperature	Y	N	Humidity	Y	N	Windy	Y	N	Play P(Y)	Play P(N)
s	2/9	3/5	h	2/9	2/5	h	3/9	4/5	f	6/9	2/5	9/14	5/14
o	4/9	0/5	m	4/9	2/5	n	6/9	1/5	t	3/9	3/5		
r	3/9	2/5	c	3/9	1/5								

Test example

	Outlook	Temp.	Humidity	Windy	Play
$\mathbf{x} =$	(sunny	cool	high	true)	?

Weather data summary (*Ver.2*)

Counts

Play	Outlook			Temp.			Humidity		Windy	
	sunny	overc	rainy	hot	mild	cool	high	norm	False	True
Yes 9	2	4	3	2	4	3	3	6	6	3
No 5	3	0	2	2	2	1	4	1	2	3

Relative frequencies $P(x|Play)$

Play	Outlook			Temp.			Humidity		Windy	
$P(Play)$	sunny	overc	rainy	hot	mild	cool	high	norm	False	True
Y 9/14	2/9	4/9	3/9	2/9	4/9	3/9	3/9	6/9	6/9	3/9
N 5/14	3/5	0/5	2/5	2/5	2/5	1/5	4/5	1/5	2/5	3/5

Test example

	Outlook	Temp.	Humidity	Windy	Play
$\mathbf{x} =$	(sunny	cool	high	true)	?

Applying Naive Bayes

Posterior prob. of "play" given x = (sunny, cool, humid, windy)

$$P(\text{Play}\,|\,\mathbf{x}) \propto P(\mathbf{x}\,|\,\text{Play})\,P(\text{Play})$$

$$P(\text{Play}=Y\,|\,\mathbf{x}) \propto P(O=s|Y)\,P(T=c|Y)\,P(H=h|Y)\,P(W=t|Y)\,P(Y)$$
$$\propto \frac{2}{9}\cdot\frac{3}{9}\cdot\frac{3}{9}\cdot\frac{3}{9}\cdot\frac{9}{14} \approx 0.0053$$

$$P(\text{Play}=N\,|\,\mathbf{x}) \propto P(O=s|N)\,P(T=c|N)\,P(H=h|N)\,P(W=t|N)\,P(N)$$
$$\propto \frac{3}{5}\cdot\frac{1}{5}\cdot\frac{4}{5}\cdot\frac{3}{5}\cdot\frac{5}{14} \approx 0.0206$$

Exercise: find the odds of play, $P(\text{play}=Y\,|\,\mathbf{x})/P(\text{play}=N\,|\,\mathbf{x})$ (answer in notes)

Naive Bayes properties

Easy and cheap:
Record counts, convert to frequencies, score each class by multiplying prior and likelihood terms

$$P(C_k\,|\,x) \propto \left(\prod_{d=1}^{D} P(x_d|C_k)\right) P(C_k)$$

Statistically viable:
Simple count-based estimates work in 1D

Often overconfident:
Treats dependent evidence as independent

Another approach for the weather example

- What about applying k-NN?
- Data representation (by quantification)

$$X = \begin{array}{c} \begin{array}{ccccc} O & T & H & W & P \end{array} \\ \begin{pmatrix} 3 & 3 & 2 & 0 & 0 \\ 3 & 3 & 2 & 1 & 0 \\ 2 & 3 & 2 & 0 & 1 \\ 1 & 2 & 2 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 2 & 1 & 1 & 1 & 1 \\ 3 & 2 & 2 & 0 & 0 \\ 3 & 1 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 & 1 \\ 3 & 2 & 1 & 1 & 1 \\ 2 & 2 & 2 & 1 & 1 \\ 2 & 3 & 1 & 0 & 1 \\ 1 & 2 & 2 & 1 & 0 \end{pmatrix} \end{array}$$

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

Outlook	sunny	3
	overc	2
	rainy	1
Temp.	hot	3
	mild	2
	cold	1
Humid.	high	2
	norm	1
Windy	True	1
	False	0
Play	Yes	1
	No	0

Another approach for the weather example (*cont.*)

- k-NN
- Sorted distance between $X(:,1\!:\!4)$ and $\mathbf{x}$

rank	dist.	idx	label
1	1.41	(7)	Y
2	1.41	(8)	N
3	1.41	(9)	Y
4	1.41	(11)	Y
5	1.41	(12)	Y
6	2.00	(2)	N
7	2.24	(1)	N
8	2.24	(6)	N
9	2.24	(14)	N
10	2.45	(3)	Y
11	2.45	(4)	Y
12	2.45	(5)	Y
13	2.65	(10)	Y
14	2.65	(13)	Y

rank	dist.	idx	label
1	1.41	(8)	N
2	1.41	(12)	Y
3	2.00	(2)	N
4	2.24	(1)	N
5	2.24	(7)	Y
6	2.24	(9)	Y
7	2.24	(11)	Y
8	2.24	(14)	N
9	2.45	(3)	Y
10	2.45	(4)	Y
11	2.83	(6)	N
12	3.00	(5)	Y
13	3.16	(10)	Y
14	3.16	(13)	Y

where the values for Humidity were doubled.

Another approach for the weather example (*cont.*)

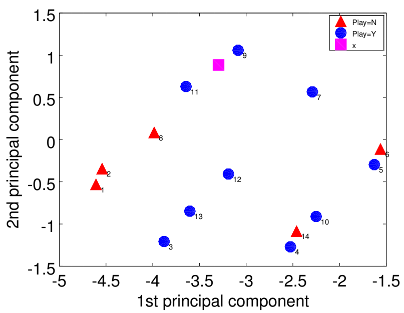
- Correlation matrix for (O, T, H, W, P)

	O	T	H	W	P
O	1.00000	0.33541	0.16903	0.00000	−0.17638
T	0.33541	1.00000	0.56695	−0.19094	−0.19720
H	0.16903	0.56695	1.00000	0.00000	−0.44721
W	0.00000	−0.19094	0.00000	1.00000	−0.25820
P	−0.17638	−0.19720	−0.44721	−0.25820	1.00000

NB: Humidity has the largest (negative) correlation with Play.

Another approach for the weather example (*cont.*)

- Dimensionality reduction by PCA



Exercise (past exam question)

The table gives a small dataset. Tick marks indicate which movies 3 children (marked c) and 4 adults (marked a) have watched. The final two rows give the movies watched by two users of the system of unknown age.

type	m_1	m_2	m_3	m_4
c	✓			
c	✓		✓	
c	✓			
a				
a		✓		
a			✓	
a	✓	✓	✓	✓
y_1	✓		✓	
y_2			✓	

Apply maximum likelihood estimation of the priors and likelihoods to this data, using the naive Bayes assumption for the likelihoods. Hence find the odds that the test user y_i is child: $P(y_i = c|\text{data})/P(y_i = a|\text{data})$ for $i = 1, 2$. State the MAP classification of each user.

Identifying Spam

Spam?

I got your contact information from your country's information directory during my desperate search for someone who can assist me secretly and confidentially in relocating and managing some family fortunes.

Identifying Spam

Spam?

Dear Dr. Steve Renals, The proof for your article, Combining Spectral Representations for Large-Vocabulary Continuous Speech Recognition, is ready for your review. Please access your proof via the user ID and password provided below. Kindly log in to the website within 48 HOURS of receiving this message so that we may expedite the publication process.

Identifying Spam

Spam?

Congratulations to you as we bring to your notice, the results of the First Category draws of THE HOLLAND CASINO LOTTO PROMO INT. We are happy to inform you that you have emerged a winner under the First Category, which is part of our promotional draws.

Identifying Spam

Question

How can we identify an email as spam automatically?

Text classification: classify email messages as spam or non-spam (ham), based on the words they contain

With the Bayes decision rule,

$$P(\text{Spam}|\mathbf{x}_1, \ldots, \mathbf{x}_L) \propto P(\mathbf{x}_1, \ldots, \mathbf{x}_L|\text{Spam})P(\text{Spam})$$

Using the naiave Bayes assumption,

$$P(\mathbf{x}_1, \ldots, \mathbf{x}_L|\text{Spam}) = P(\mathbf{x}_1|\text{Spam}) \cdots P(\mathbf{x}_L|\text{Spam})$$

Summary

- The curse of dimensionality
- Approximation by the Naive Bayes rule
- Example: classifying multidimensional data using Naive Bayes
- Next lecture: Text classification using Naive Bayes

Inf2b - Learning

Lecture 7: Text Classification using Naive Bayes

Hiroshi Shimodaira
(Credit: Iain Murray and Steve Renals)

Centre for Speech Technology Research (CSTR)
School of Informatics
University of Edinburgh

http://www.inf.ed.ac.uk/teaching/courses/inf2b/
https://piazza.com/ed.ac.uk/spring2020/infr08028
Office hours: Wednesdays at 14:00-15:00 in IF-3.04

Jan-Mar 2020

Today's Schedule

1. Text classification
2. Bag-of-words models
3. Multinomial document model
4. Bernoulli document model
5. Generative models
6. Zero Probability Problem

Identifying Spam

Spam?

I got your contact information from your country's information directory during my desperate search for someone who can assist me secretly and confidentially in relocating and managing some family fortunes.

Identifying Spam

Spam?

Dear Dr. Steve Renals, The proof for your article, Combining Spectral Representations for Large-Vocabulary Continuous Speech Recognition, is ready for your review. Please access your proof via the user ID and password provided below. Kindly log in to the website within 48 HOURS of receiving this message so that we may expedite the publication process.

Identifying Spam

Spam?

Congratulations to you as we bring to your notice, the results of the First Category draws of THE HOLLAND CASINO LOTTO PROMO INT. We are happy to inform you that you have emerged a winner under the First Category, which is part of our promotional draws.

Text Classification using Bayes Theorem

- Document $\mathcal{D}$, with a fixed set of classes $C = \{1, \ldots, K\}$
- Classify $\mathcal{D}$ as the class with the highest posterior probability:

$$k_{\max} = \arg\max_k P(C_k \mid \mathcal{D}) = \arg\max_k \frac{P(\mathcal{D} \mid C_k)\, P(C_k)}{P(\mathcal{D})}$$
$$= \arg\max_k P(\mathcal{D} \mid C_k)\, P(C_k)$$

- How do we represent $\mathcal{D}$?
- How do we estimate $P(\mathcal{D} \mid C_k)$ and $P(C_k)$?

How do we represent $\mathcal{D}$?

- A sequence of words: $\mathcal{D} = (X_1, X_2, \ldots, X_n)$
  computational very expensive, difficult to train
- A set of words (Bag-of-Words)
  - Ignore the position of the word
  - Ignore the order of the word
  - Consider the words in pre-defined vocabulary V ($D = |V|$)

Multinomial document model a document is represented by an integer feature vector, whose elements indicate frequency of corresponding word in the document

$$\mathbf{x} = (x_1, \ldots, x_D) \qquad x_i \in \mathcal{N}_0$$

Bernoulli document model a document is represented by a binary feature vector, whose elements indicate absence or presence of corresponding word in the document

$$\mathbf{b} = (b_1, \ldots, b_D) \qquad b_i \in \{0, 1\}$$

BoW models: Bernoulli vs. Multinomial

Document $\mathcal{D}$: "Congratulations to you as we bring to your notice, the results of the First Category draws of THE HOLLAND CASINO LOTTO PROMO INT. We are happy to inform you that you have emerged a winner under the First Category, which is part of our promotional draws."

Term ($w_t \in V$)	Multinomial ($x_t \in \mathcal{N}_0$) $\mathbf{x} = (x_t)$	Bernoulli ($b_t \in \{0,1\}$) $\mathbf{b} = (b_t)$
bring	1	1
can	0	0
casino	1	1
category	2	1
congratulations	1	1
draws	2	1
first	2	1
lotto	1	1
the	4	1
true	0	0
winner	1	1
you	3	1
$D = 12$	$\mathbf{x} = (1, 0, 1, 2, \ldots, 1, 3)$	$\mathbf{b} = (1, 0, 1, 1, \ldots, 1, 1)$

Notation for document model

- Training documents:

Class	Documents
C_1	$\mathcal{D}_1^{(1)} \ldots \mathcal{D}_i^{(1)} \ldots \mathcal{D}_{N_1}^{(1)}$
$\vdots$	$\vdots$
C_K	$\mathcal{D}_1^{(K)} \ldots \mathcal{D}_i^{(K)} \ldots \mathcal{D}_{N_K}^{(K)}$

- Flattened representation of training data:

Documents	$\mathcal{D}_1$	$\ldots$	$\mathcal{D}_i$	$\ldots$	$\mathcal{D}_N$
Class indicator	z_{1k}	$\ldots$	z_{ik}	$\ldots$	z_{Nk}

where $N = N_1 + \cdots + N_K$,

$$z_{ik} = \begin{cases} 1 & \text{if } \mathcal{D}_i \text{ belongs to class } C_k \\ 0 & \text{otherwise} \end{cases}$$

- Test document : $\mathcal{D}$

Discrete probability distributions - review

Bernoulli distribution

Eg: Tossing a biased coin ($P(H) = p$), the probability of $k = \{0, 1\}$ 0:Tail, 1:Head is

$$P(k) = kp + (1-k)(1-p) = p^k (1-p)^{1-k}$$

Binomial distribution

Eg: Tossing a biased coin n times, the probability of observing Head k times is

$$P(k) = \binom{n}{k} p^k (1-p)^{n-k}. \qquad \binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Multinomial distribution

Eg: Tossing a biased dice n times, the probability of $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5, x_6)$, where x_i is the number of occurrences for face i, is

$$P(\mathbf{x}) = \frac{n!}{x_1! \cdots x_6!} p_1^{x_1} p_2^{x_2} p_3^{x_3} p_4^{x_4} p_5^{x_5} p_6^{x_6}.$$

Classification with multinomial document model

Assume a test document $\mathcal{D}$ is given as a sequence of words:

$(o_1, o_2, \ldots, o_n) \qquad o_i \in V = \{w_1, \ldots, w_D\}$

Feature vector: $\mathbf{x} = (x_1, \ldots, x_D)$ $\cdots$ word frequencies, $\sum_{t=1}^{D} x_t = n$

Document likelihood with multinomial distribution:

$$P(\mathbf{x} \mid C_k) = \frac{n!}{\prod_{t=1}^{D} x_t!} \prod_{t=1}^{D} P(w_t \mid C_k)^{x_t} \qquad \text{NB: } P^0 = 1 \ (P > 0)$$

For classification, we can omit irrelevant term, so that:

$$P(\mathbf{x} \mid C_k) \propto \prod_{t=1}^{D} P(w_t \mid C_k)^{x_t} = P(o_1|C_k)\, P(o_2|C_k) \cdots P(o_n|C_k)$$

$$P(C_k \mid \mathbf{x}) \propto P(C_k) \prod_{i=1}^{n} P(o_i \mid C_k)$$

Training of multinomial document model

Features: $\mathbf{x} = (x_1, \ldots, x_D)$: *word frequencies* in a doc.
Training data set

Class	Docs	Feature vectors
C_1	$\mathcal{D}_1^{(1)}, \vdots, \mathcal{D}_{N_1}^{(1)}$	$\begin{pmatrix} \mathbf{x}_1^{(1)} \\ \vdots \\ \mathbf{x}_{N_1}^{(1)} \end{pmatrix} = \begin{pmatrix} x_{11}^{(1)} & \ldots & x_{1D}^{(1)} \\ \vdots & & \vdots \\ x_{N_1 1}^{(1)} & \ldots & x_{N_1 D}^{(1)} \end{pmatrix}$
		$n_1(w_1), \ldots, n_1(w_D)$
	$\hat{P}(C_1) = N_1/N$	$\hat{P}(w_t \mid C_1)$: $n_1(w_1)/S_1, \ldots, n_1(w_D)/S_1$
C_k	$\mathcal{D}_1^{(k)}, \vdots, \mathcal{D}_{N_k}^{(k)}$	$\begin{pmatrix} \mathbf{x}_1^{(k)} \\ \vdots \\ \mathbf{x}_{N_k}^{(k)} \end{pmatrix} = \begin{pmatrix} x_{11}^{(k)} & \ldots & x_{1D}^{(k)} \\ \vdots & & \\ x_{N_k 1}^{(k)} & \ldots & x_{N_k D}^{(k)} \end{pmatrix}$
		$n_k(w_1), \ldots, n_k(w_D)$
	$\hat{P}(C_k) = N_k/N$	$\hat{P}(w_t \mid C_k)$: $n_k(w_1)/S_k \ldots, n_k(w_D)/S_k$

$S_k = \sum_{t=1}^{D} n_k(w_t)$

Multinomial doc. model – example

See Note 7!

Classification with Bernoulli document model

A test document $\mathcal{D}$ with feature vector $\mathbf{b} = (b_1, \ldots, b_D)$

Document likelihood with (multivariate) Bernoulli distribution:

$$P(\mathbf{b} \mid C_k) = \prod_{t=1}^{D} P(b_t|C_k) = \prod_{t=1}^{D} [b_t P(w_t|C_k) + (1-b_t)(1-P(w_t|C_k))]$$
$$= \prod_{t=1}^{D} P(w_t \mid C_k)^{b_t} (1-P(w_t|C_k))^{(1-b_t)}$$

$$\hat{P}(w_t|C_k) = \frac{n_k(w_t)}{N_k}$$

(fraction of class k docs with word w_t)

In Classification,

$$P(C_k \mid \mathbf{b}) \propto P(C_k)\, P(\mathbf{b} \mid C_k)$$

Training of Bernoulli document model

Features: $\mathbf{b} = (b_1, \dots, b_D)$: $D = |V|$, i.e. vocabulary *binary vector* of word occurrences in a document

Training data set

Class	Docs	Feature vectors
C_1	$\mathcal{D}_1^{(1)}, \dots, \mathcal{D}_{N_1}^{(1)}$	$\begin{pmatrix} \mathbf{b}_1^{(1)} \\ \vdots \\ \mathbf{b}_{N_1}^{(1)} \end{pmatrix} = \begin{pmatrix} b_{11}^{(1)} & \dots & b_{1D}^{(1)} \\ \vdots & & \vdots \\ b_{N_1 1}^{(1)} & \dots & b_{N_1 D}^{(1)} \end{pmatrix}$
		$n_1(w_1), \dots, n_1(w_D)$
	$\hat{P}(C_1) = N_1/N$	$\hat{P}(w_t \vert C_1)$: $n_1(w_1)/N_1, \dots, n_1(w_D)/N_1$
C_k	$\mathcal{D}_1^{(k)}, \dots, \mathcal{D}_{N_k}^{(k)}$	$\begin{pmatrix} \mathbf{b}_1^{(k)} \\ \vdots \\ \mathbf{b}_{N_k}^{(k)} \end{pmatrix} = \begin{pmatrix} b_{11}^{(k)} & \dots & b_{1D}^{(k)} \\ \vdots & & \vdots \\ b_{N_1 1}^{(k)} & \dots & b_{N_1 D}^{(k)} \end{pmatrix}$
		$n_k(w_1), \dots, n_k(w_D)$
	$\hat{P}(C_k) = N_k/N$	$\hat{P}(w_t \vert C_k)$: $n_k(w_1)/N_k, \dots, n_k(w_D)/N_k$

Bernoulli doc. model – example

Classify documents as Sports (S) or Informatics (I)

Vocabulary V:

w_1 = *goal*
w_2 = *tutor*
w_3 = *variance*
w_4 = *speed*
w_5 = *drink*
w_6 = *defence*
w_7 = *performance*
w_8 = *field*

$D = |V| = 8$

Bernoulli doc. model – example *(cont.)*

Training data: (rows give documents, columns word presence)

$$\mathbf{B}^{\text{Sport}} = \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 0&0&1&0&1&1&0&0 \\ 0&1&0&1&0&1&1&0 \\ 1&0&0&1&0&1&0&1 \\ 1&0&0&0&1&0&1&1 \\ 0&0&1&1&0&0&1&1 \end{pmatrix}$$

$$\mathbf{B}^{\text{Inf}} = \begin{pmatrix} 0&1&1&0&0&0&1&0 \\ 1&1&0&1&0&0&1&1 \\ 0&1&1&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&1&0&1&0&1&0 \end{pmatrix}$$

Estimating priors and likelihoods:

$$P(S) = 6/11, \quad P(I) = 5/11$$
$$(P(w_t|S)) = (\ 3/6 \quad 1/6 \quad 2/6 \quad 3/6 \quad 3/6 \quad 4/6 \quad 4/6 \quad 4/6\ )$$
$$(P(w_t|I)) = (\ 1/5 \quad 3/5 \quad 3/5 \quad 1/5 \quad 1/5 \quad 1/5 \quad 3/5 \quad 1/5\ )$$

Bernoulli doc. model – example *(cont.)*

Test documents: $\boldsymbol{b}_1 = [\ 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1\ ]$

Priors, Likelihoods: $P(S) = 6/11, \quad P(I) = 5/11$

$$(P(w_t|S)) = (\ 3/6 \quad 1/6 \quad 2/6 \quad 3/6 \quad 3/6 \quad 4/6 \quad 4/6 \quad 4/6\ )$$
$$(P(w_t|I)) = (\ 1/5 \quad 3/5 \quad 3/5 \quad 1/5 \quad 1/5 \quad 1/5 \quad 3/5 \quad 1/5\ )$$

Posterior probabilities:

$$P(S\,|\,\boldsymbol{b}_1) \propto P(S)\prod_{t=1}^{8}[b_{1t}P(w_t\,|\,S) + (1-b_{1t})(1-P(w_t\,|\,S))]$$
$$\propto \frac{6}{11}\left(\frac{1}{2}\times\frac{5}{6}\times\frac{2}{3}\times\frac{1}{2}\times\frac{1}{2}\times\frac{2}{3}\times\frac{1}{3}\times\frac{2}{3}\right) = \frac{5}{891} = 5.6\times10^{-3}$$

$$P(I\,|\,\boldsymbol{b}_1) \propto P(I)\prod_{t=1}^{8}[b_{1t}P(w_t\,|\,I) + (1-b_{1t})(1-P(w_t\,|\,I))]$$
$$\propto \frac{5}{11}\left(\frac{1}{5}\times\frac{2}{5}\times\frac{2}{5}\times\frac{1}{5}\times\frac{1}{5}\times\frac{1}{5}\times\frac{2}{5}\times\frac{1}{5}\right) = \frac{8}{859375} = 9.3\times10^{-6}$$

$\Rightarrow$ Classify this document as S.

Summary of the document models

Class	Doc	Multinomial doc. model feature vectors	Bernoulli doc. model feature vectors
C_k	$\mathcal{D}_1^{(k)}, \dots, \mathcal{D}_{N_k}^{(k)}$	$\begin{pmatrix} \mathbf{x}_1^{(k)} \\ \vdots \\ \mathbf{x}_{N_k}^{(k)} \end{pmatrix} = \begin{pmatrix} x_{11}^{(k)} & \dots & x_{1D}^{(k)} \\ \vdots & & \vdots \\ x_{N_1 1}^{(k)} & \dots & x_{1D}^{(k)} \end{pmatrix}$	$\begin{pmatrix} \mathbf{b}_1^{(k)} \\ \vdots \\ \mathbf{b}_{N_k}^{(k)} \end{pmatrix} = \begin{pmatrix} b_{11}^{(k)} & \dots & b_{1D}^{(k)} \\ \vdots & & \vdots \\ b_{N_1 1}^{(k)} & \dots & b_{1D}^{(k)} \end{pmatrix}$
$\hat{P}(C_k) = \frac{N_k}{N}$		$n_k(w_1), \dots, n_k(w_D)$	$n_k(w_1), \dots, n_k(w_D)$
$\hat{P}(w_t \vert C_k)$:		$\frac{n_k(w_1)}{S_k}, \dots, \frac{n_k(w_D)}{S_k}$	$\frac{n_k(w_1)}{N_k}, \dots, \frac{n_k(w_D)}{N_k}$

$S_k = \sum_{t=1}^{D} n_k(w_t)$

$$P(\boldsymbol{x}\,|\,C_k) \propto \prod_{t=1}^{D} P(w_t|C_k)^{x_t} = \prod_{i=1}^{n} P(o_i|C_k)$$

$$P(\boldsymbol{b}\,|\,C_k) = \prod_{t=1}^{D}[b_t P(w_t|C_k) + (1-b_t)(1-P(w_t|C_k))]$$

Question

What's the approximate value of:

$P(\text{"the"} \mid C)$

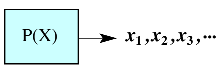
(a) in the Bernoulli model

(b) in the multinomial model?

Common words, 'stop words', are often removed from feature vectors.

Generative models

- Models that generate observable data randomly based on a distribution



- Examples
  - Coin tossing models

Coin	Generated data sequence
Fair coin ($P(H)=P(T)=0.5$)	$H, T, T, H, T, H, H, T, \dots$
Unfair coin ($P(H)=0.7, P(T)=0.3$)	$T, H, H, H, H, H, T, H, \dots$

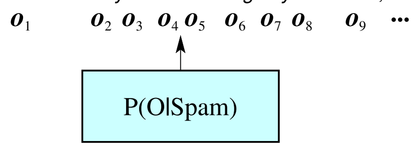
  - Dice throwing models

Dice	Generated data sequence
Unbiased dice ($P(X) = 1/6,\ X \in \{1, \dots, 6\}$)	$2, 4, 3, 5, 3, 6, 5, 5, 4, 6, \dots$
Biased dice ($(P(X)) = (0.1, 0.1, 0.1, 0.1, 0.2, 0.4)$)	$6, 6, 5, 5, 6, 1, 2, 6, 6, 6, \dots$

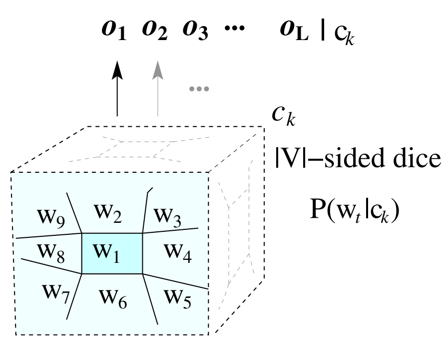
Generative models *(cont.)*

- Spam mail generator

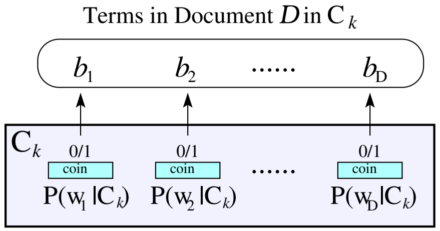
Congratulations to you as we bring to your notice, ...



Generative model — Multinomial document model



Generative model — Bernoulli document model



Word relative-frequencies of spam emails

of spam emails: 169

to	0.0395032	from	0.00664282	http	0.00369482
the	0.0383633	content	0.00644629	money	0.00345898
you	0.0267285	have	0.0059353	by	0.00338037
of	0.0257851	bank	0.0059353	or	0.00330176
and	0.0252349	usd	0.00581738	name	0.00322314
your	0.0222476	on	0.00554223	funds	0.00322314
in	0.0200857	we	0.00542432	was	0.00318384
i	0.0198892	it	0.00518848	type	0.00318384
this	0.0145828	are	0.00507056	s	0.00318384
a	0.0138752	transfer	0.00479541	0a	0.00314453
my	0.0132463	our	0.0047561	if	0.00310522
for	0.0132463	com	0.00467749	1	0.00310522
is	0.0112024	am	0.00467749	can	0.00306592
3d	0.0108879	account	0.00455957	payment	0.002948
with	0.00915845	unlocked	0.00424512	message	0.002948
will	0.00876538	20	0.0041665	address	0.00286938
that	0.00849023	email	0.00404858	us	0.00283008
as	0.00797925	please	0.00385205	his	0.00279077
me	0.00766479	not	0.00377344	contact	0.00279077
be	0.00703589	all	0.00377344	has	0.00271216

Generated word sequence example

of kin good your the part of with and atm to new from which projects has the transfer my how 3d and with united in in o beneficiary that died pathak id efforts has to studies have my as can you the 3d you your with transfer will your a your m and the your i is ve country user nokia the this for i value banking an click confirm world i it me my country is 2010 very below i and now until html of position http here of mail following there be while the by for your willing

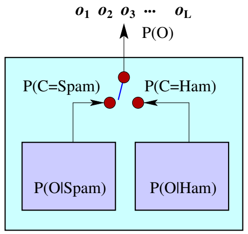
Generative models for classification

Model for classification

$$P(C_k \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid C_k)\, P(C_k)}{P(\mathbf{x})} \propto P(\mathbf{x} \mid C_k)\, P(C_k)$$

Model for observation $\cdots$ generative model

$$P(\mathbf{x}) = \sum_{k=1}^{K} P(\mathbf{x}|C_k)P(C_k)$$



Smoothing in multinomial document model

- Zero probability problem

$$P(\mathbf{x} \mid C_k) \propto \prod_{t=1}^{D} P(w_t \mid C_k)^{x_t} = 0 \text{ if } \exists j : P(w_j|C_k) = 0$$

$$P(w_t \mid C_k) = \frac{\sum_{i=1}^{N} x_{it} z_{ik}}{\sum_{t'=1}^{|V|} \sum_{i=1}^{N} x_{it'} z_{ik}} = \frac{n_k(w_t)}{\sum_{t'=1}^{D} n_k(w_{t'})}$$

- Smoothing – a 'trick' to avoid zero counts:

$$P(w_t \mid C_k) = \frac{1 + \sum_{i=1}^{N} x_{it} z_{ik}}{|V| + \sum_{t'=1}^{|V|} \sum_{i=1}^{N} x_{it'} z_{ik}} = \frac{1 + n_k(w_t)}{D + \sum_{t'=1}^{D} n_k(w_{t'})}$$

  Known as *Laplace's rule of succession* or *add one smoothing*.

Multinomial vs Bernoulli doc. models

	Multinomial	Bernoulli
Generative model	draw a words from a multinomial distribution	draw a document from a multi-dimensional Bernoulli distribution
Document representation	Vector of frequencies	Binary vector
Multiple occurrences	Taken into account	Ignored
Document length	Longer docs OK	Best for short docs
Feature vector dimension	Longer OK	Shorter
Behaviour with "the"	$P(\text{"the"}\|C_k) \approx 0.05$	$P(\text{"the"}\|C_k) \approx 1.0$
Non-occurring words in test doc	do not affect likelihood	affect likelihood

Multinomial vs Bernoulli doc. models *(cont.)*

Fig. 1 in A. McCallum and K.Nigam, "A Comparison of Event Models for Naive Bayes Text Classification", AAAI Workshop on Learning for Text Categorization, 1998

Document pre-processing

- Stop-word removal
  Remove pre-defined common words that are not specific or discriminatory to the different classes.
- Stemming
  Reduce different forms of the same word into a single word (base/root form)
- Feature selection
  e.g. choose words based on the mutual information

Exercise 1

Use the Bernoulli model and the Naive Bayes assumption for the following.
Consider the vocabulary $V = \{\texttt{apple}, \texttt{banana}, \texttt{computer}\}$. We have two classes of documents F (fruit) and E (electronics). There are four training documents in class F; they are listed below in terms of the number of occurrences of each word from V in each document:

- apple(2); banana(1); computer(0)
- apple(0); banana(1); computer(0)
- apple(3); banana(2); computer(1)
- apple(1); banana(0); computer(0)

There are also four training documents in class E:

- apple(2); banana(0); computer(0)
- apple(0); banana(0); computer(1)
- apple(3); banana(1); computer(2)
- apple(0); banana(0); computer(1)

Exercise 1 *(cont.)*

1. Write the training data as a matrix for each class, where each row corresponds to a training document.
2. Estimate the prior probabilities from the training data
3. For each class (F and E) and for each word (apple, banana and computer) estimate the likelihood of the word given the class.
4. Consider two test documents:
   - `apple`(1); `banana`(0); `computer`(0)
   - `apple`(1); `banana`(1); `computer`(0)

   For each test document, estimate the posterior probabilities of each class given the document, and hence classify the document.

Exercise 2

Use the Multinomial model and the Naive Bayes assumption for the following.
Consider the vocabulary $V = \{\texttt{fish}, \texttt{chip}, \texttt{circuit}\}$. We have two classes of documents F (food) and E (electronics). There are four training documents in class F; they are listed below;

- `fish chip fish`
- `chip`
- `circuit fish chip`
- `fish fish`

There are also four training documents in class E:

- `circuit circuit`
- `chip circuit`
- `chip chip`
- `circuit`

Exercise 2 *(cont.)*

1. Estimate the parameters of a multinomial model for the two document classes, using add-one smoothing.
2. Consider two test documents:
   - `fish chip`
   - `chip circuit chip circuit fish chip circuit`

   Classify each of the test documents by (approximately) estimating the posterior probability of each class
3. With reference to the test documents in the previous question, explain why a process such as add-one smoothing is used when estimating the parameters of a multinomial model.

Exercise 3

Consider two writers, Baker and Clark, who were twins, and who published four and six children's books, respectively. The following table shows the frequencies of four words, **wizard**, **river**, **star**, and **warp**, with respect to the first page of each book, and the information whether the book was a bestseller or not.

	Words				
Author	wizard	river	star	warp	Bestseller
Baker	1	1	1	0	No
Baker	1	1	0	1	No
Baker	1	1	1	1	yes
Baker	1	1	0	0	No
Clark	0	1	0	1	No
Clark	0	0	2	1	No
Clark	0	2	1	2	Yes
Clark	1	1	1	2	No
Clark	0	1	2	2	Yes
Clark	0	1	2	1	Yes

Two unpublished book drafts, Doc 1 and Doc 2, were found after the death of the writers, but it's not clear which of them wrote the documents.

Exercise 3 *(cont.)*

1. Without having any information about Doc 1 and Doc 2, decide the most probable author of each document in terms of minimum classification error, and justify your decision.
2. The same analysis of word frequencies was carried out for Doc 1 and Doc 2, whose result is shown below. Using the Naive Bayes classification with the multinomial document model without smoothing, find the author of each document.

	wizard	river	start	warp
Doc 1	2	1	1	0
Doc 2	1	1	2	1

3. In addition to modifications to the vocabulary, discuss two possible methods for improving the classification performance.
4. Another document, Doc 3, was found later, and a publisher is considering its publication. Assuming the Naive Bayes classification with the multinomial document model with no smoothing, without identifying the author, predict whether Doc 3 is likely to be a bestseller or not based on the word frequency table for Doc 3 shown below.

	wizard	river	start	warp
Doc 3	0	1	1	2

5. Using the same situations as in part (d) except that we now know the author of Doc 3 was Baker, predict whether Doc 3 is likely to be a bestseller or not.

Summary

- Our first 'real' application of Naive Bayes
- Two BoW models for documents: Multinomial and Bernoulli
- Generative models
- Smoothing (Add-one/Laplace smoothing)
- Good reference:
  C. Manning, P. Raghavan and H. Schütze, Introduction to Information Retrieval, University Press. 2008.
  See Chapter 13 Text classification & Naive Bayes
- **As always:**
  be able to implement, describe, compare and contrast (see Lecture Note)

Inf2b - Learning

Lecture 8: Real-valued distributions and Gaussians

Hiroshi Shimodaira
(Credit: Iain Murray and Steve Renals)

Centre for Speech Technology Research (CSTR)
School of Informatics
University of Edinburgh

http://www.inf.ed.ac.uk/teaching/courses/inf2b/
https://piazza.com/ed.ac.uk/spring2020/infr08028
Office hours: Wednesdays at 14:00-15:00 in IF-3.04

Jan-Mar 2020

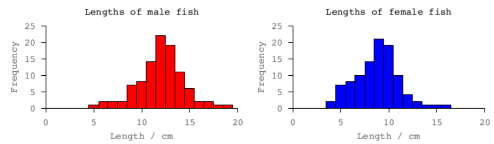
Today's Schedule

Real-valued distributions and Gaussians

1. Continuous random variables
2. The Gaussian distribution (one-dimensional)
3. Maximum likelihood estimation
4. The multidimensional Gaussian distribution

Discrete to continuous random variables

Fish example again:

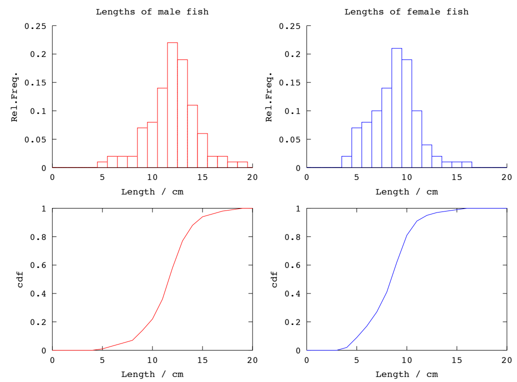


$$c^* = \arg\max_c P(c|x) = \arg\max_c \frac{P(x|c)P(c)}{P(x)} = \arg\max_c P(x|c)P(c)$$

- What if the number of bins $\to \infty$? (i.e. the width of bin $\to 0$)
- $P(X = x|C)$ will be almost 0 everywhere!
- We instead consider a cumulative distribution function (cdf) with a continuous random variable:

  $F(x) = P(X \leq x)$

Cumulative distribution functions graphed



Cumulative distribution function properties

Cumulative distribution functions have the following properties:

1. $F(-\infty) = 0$;
2. $F(\infty) = 1$;
3. If $a \leq b$ then $F(a) \leq F(b)$.

To obtain the probability of falling in an interval we can do the following:

$$P(a < X \leq b) = P(X \leq b) - P(X \leq a) = F(b) - F(a)$$

Probability density function (pdf)

- The rate of change of the cdf gives us the probability density function (pdf) , $p(x)$:

$$p(x) = \frac{d}{dx}F(x) = F'(x)$$

$$F(x) = \int_{-\infty}^{x} p(x)\, dx$$

- $p(x)$ is **not** the probability that X has value x. But the pdf is proportional to the probability that X lies in a small interval $[x, x + dx]$.
- Notation: p for pdf, P for probability

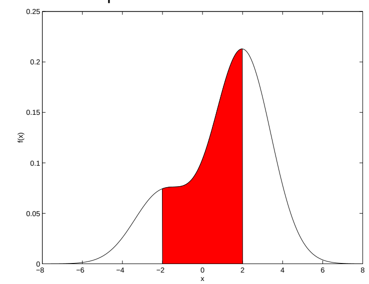
pdf and cdf

The probability that the random variable lies in interval (a, b) is given by:

$$P(a < X \leq b) = F(b) - F(a) = \int_{-\infty}^{b} p(x)\, dx - \int_{-\infty}^{a} p(x)\, dx = \int_{a}^{b} p(x)\, dx$$

pdf and cdf

The probability that the random variable lies in interval (a, b) is the area under the pdf between a and b:



The Gaussian distribution

- The Gaussian (or Normal) distribution is the most common (and easily analysed) continuous distribution
- It is also a reasonable model in many situations (the famous "bell curve")
- If a (scalar) variable has a Gaussian distribution, then it has a probability density function with this form:

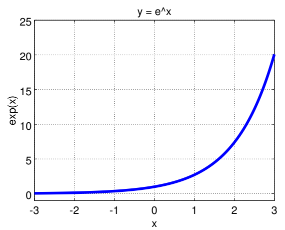
$$p(x\,|\,\mu, \sigma^2) = N(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$$

NB: $\exp(f(x)) = e^{f(x)}$

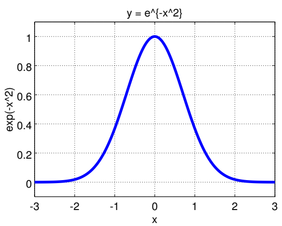
- The Gaussian is described by two parameters:
  - the mean μ (location)
  - the variance σ^2 (dispersion)

Natural exponential function

$y = e^x = \exp(x)$

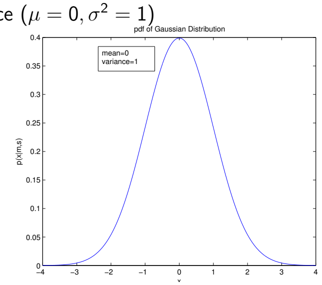


$y = \exp(-x^2)$



Plot of Gaussian distribution

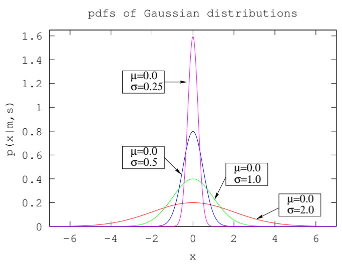
- Gaussians have the same shape, with the location controlled by the mean, and the spread controlled by the variance
- One-dimensional Gaussian with zero mean and unit variance ($\mu = 0, \sigma^2 = 1$)



Another plot of a Gaussian

Properties of the Gaussian distribution

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



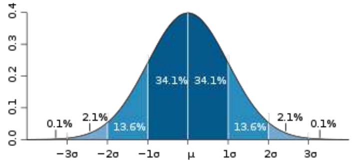
$$\int_{-\infty}^{\infty} N(x;\mu,\sigma^2)dx = 1$$

$$\lim_{\sigma\to 0} N(x;\mu,\sigma^2) = \delta(x-\mu)$$

(Dirac delta function)

Facts about the Gaussian distribution

- A Gaussian can be used to describe approximately any random variable that tends to cluster around the mean
- Concentration:
  - About 68% of values drawn from a normal distribution are within one SD away from the mean
  - About 95% are within two SDs
  - About 99.7% lie within three SDs of the mean



Central Limit Theorem

- Under certain conditions, the sum of a large number of random variables will have approximately normal distribution.
- Several other distributions are well approximated by the Normal distribution:
  - Binomial $B(n,p)$, when n is large and p is not too close to 1 or 0
  - Poisson $P_o(\lambda)$ when λ is large
  - Other distributions including chi-squared and Student's T
- The Wikipedia entry on the Gaussian distribution is good

Parameter estimation form data

- Estimate the mean and variance parameters of a Gaussian from data $\{x_1, x_2, \ldots, x_N\}$
- Sample mean and sample variance (unbiased) estimates:

$$\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

$$\hat{\sigma}^2 = \frac{1}{N-1}\sum_{n=1}^{N}(x_n-\hat{\mu})^2$$

- Maximum likelihood estimates (MLE):

$$\hat{\mu}_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

$$\hat{\sigma}^2_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}(x_n-\hat{\mu}_{\mathrm{ML}})^2$$

Example: Gaussians

A pattern recognition problem has two classes, S and T. Some observations are available for each class:

Class S	10	8	10	10	11	11
Class T	12	9	15	10	13	13

The mean and variance of each pdf are estimated with MLE.

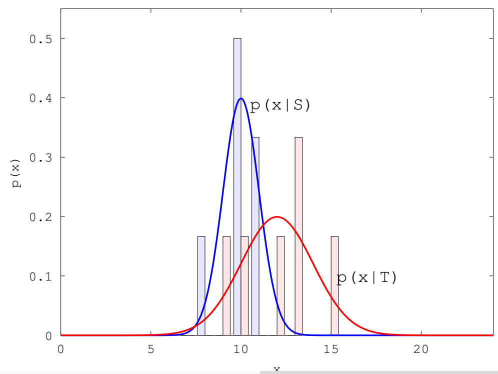
S : mean = 10; variance = 1
T : mean = 12; variance = 4

$$p(x|S) = \frac{1}{\sqrt{2\pi\cdot 1}}\exp\left(-\frac{(x-10)^2}{2\cdot 1}\right)$$

$$p(x|T) = \frac{1}{\sqrt{2\pi\cdot 4}}\exp\left(-\frac{(x-12)^2}{2\cdot 4}\right)$$

Example: Gaussians (cont.)

Sketch the pdf for each class. cf. the histograms



Parameter estimation as an optimisation problem

- Given an observation (training) set of N samples: $\mathcal{D} = \{x_1, x_2, \ldots, x_N\}$
- How can we estimate the mean μ and variance σ^2 of the population?
- Define the problem as an optimisation problem

  Maximum Likelihood (ML) estimation:

$$\max_{\mu,\sigma^2} p(\mathcal{D}\,|\,\mu,\sigma^2)$$

NB: ML is just a one criterion for parameter estimation

ML estimation of a univariate Gaussian pdf

Assumption:

Samples $\mathcal{D} = \{x_n\}_{n=1}^{N}$ are drawn independently from the same distribution (i.i.d.)

Likelihood:

$$\begin{aligned} p(\mathcal{D}\,|\,\mu,\sigma^2) &= p(x_1,\ldots,x_N\,|\,\mu,\sigma^2) \\ &= p(x_1\,|\,\mu,\sigma^2)\cdots p(x_N\,|\,\mu,\sigma^2) = \prod_{n=1}^{N} p(x_n\,|\,\mu,\sigma^2) \\ &= L(\mu,\sigma^2\,|\,\mathcal{D}) \end{aligned}$$

Optimisation problem:

Find such parameters μ and σ^2 that maximise the likelihood:

$$\max_{\mu,\sigma^2} L(\mu,\sigma^2\,|\,\mathcal{D})$$

ML estimation of a univariate Gaussian pdf (cont.)

The log likelihood: NB: the natural log (ln) is assumed

$$\begin{aligned} LL(\mu,\sigma^2\,|\,\mathcal{D}) = \ln L(\mu,\sigma^2\,|\,\mathcal{D}) &= \ln \prod_{n=1}^{N} p(x_n\,|\,\mu,\sigma^2) \\ &= \sum_{n=1}^{N} \ln p(x_n\,|\,\mu,\sigma^2) \\ &= \sum_{n=1}^{N} \ln\left(\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-(x_n-\mu)^2}{2\sigma^2}\right)\right) \\ &= -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln\left(\sigma^2\right) - \sum_{n=1}^{N}\frac{(x_n-\mu)^2}{2\sigma^2} \end{aligned}$$

ML estimation of a univariate Gaussian pdf *(cont.)*

$$LL(\mu, \sigma^2 \,|\, \mathcal{D}) = -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln\left(\sigma^2\right) - \sum_{n=1}^{N}\frac{(x_n-\mu)^2}{2\sigma^2}$$

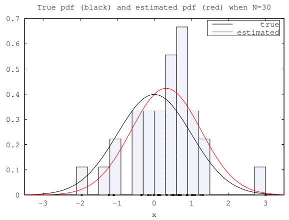
$$\frac{\partial LL(\mu, \sigma^2 \,|\, \mathcal{D})}{\partial \mu} = 2\sum_{n=1}^{N}\frac{x_n-\mu}{2\sigma^2} = 0$$

$$\Rightarrow \hat{\mu} = \frac{1}{N}\sum_{n=1}^{N}x_n$$

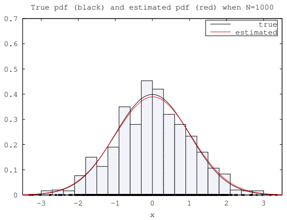
$$\frac{\partial LL(\hat{\mu}, \sigma^2 \,|\, \mathcal{D})}{\partial \sigma^2} = -\frac{N}{2}\frac{1}{\sigma^2} + \sum_{n=1}^{N}\frac{(x_n-\hat{\mu})^2}{2(\sigma^2)^2} = 0$$

$$\Rightarrow \sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n-\hat{\mu})^2$$

Examples of parameter estimation with MLE



$N = 30$



$N = 1000$

The multidimensional Gaussian distribution

- The D-dimensional vector $\mathbf{x} = (x_1, \ldots, x_D)^T$ is multivariate Gaussian if it has a probability density function of the following form:

  $$p(\mathbf{x} \,|\, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right).$$

  The pdf is parameterised by the mean vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_D)^T$ and the covariance matrix $\boldsymbol{\Sigma} = (\sigma_{ij})$.
- The 1-dimensional Gaussian is a special case of this pdf
- The argument to the exponential $\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})$ is referred to as a *quadratic form*.

Covariance matrix

- The mean vector $\boldsymbol{\mu}$ is the expectation of $\mathbf{x}$:
  $$\boldsymbol{\mu} = E[\mathbf{x}]$$
- The covariance matrix $\boldsymbol{\Sigma}$ is the expectation of the deviation of $\mathbf{x}$ from the mean:
  $$\boldsymbol{\Sigma} = E[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T]$$
- $\boldsymbol{\Sigma}$ is a $D \times D$ symmetric matrix: $\boldsymbol{\Sigma}^T = \boldsymbol{\Sigma}$
  $$\sigma_{ij} = E[(x_i-\mu_i)(x_j-\mu_j)] = E[(x_j-\mu_j)(x_i-\mu_i)] = \sigma_{ji}\,.$$
- The sign of the covariance σ_{ij} helps to determine the relationship between two components:
  - If x_j is large when x_i is large, then $(x_j-\mu_j)(x_i-\mu_i)$ will tend to be positive;
  - If x_j is small when x_i is large, then $(x_j-\mu_j)(x_i-\mu_i)$ will tend to be negative.

Covariance matrix *(cont.)*

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \cdots & \cdots & \cdots & \sigma_{1D} \\ \sigma_{21} & \sigma_{22} & \cdots & \cdots & \cdots & \sigma_{2D} \\ \vdots & \vdots & \ddots & & & \vdots \\ \vdots & \vdots & & \sigma_{ii} & & \vdots \\ \vdots & \vdots & & & \ddots & \vdots \\ \sigma_{D1} & \sigma_{D2} & \cdots & \cdots & \cdots & \sigma_{DD} \end{pmatrix}$$

- $\sigma_i^2 = \sigma_{ii}$
- $|\boldsymbol{\Sigma}| = \det(\boldsymbol{\Sigma})$: determinant
  e.g. for $D = 2$,
  $$|\boldsymbol{\Sigma}| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = a \times d - b \times c$$
- See dimensionality reduction with PCA in Lecture Slides (3).

Parameter estimation

Maximum likelihood estimation (MLE):

$$\boldsymbol{\mu} = E[\mathbf{x}]$$

$$\hat{\boldsymbol{\mu}}_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

$$\boldsymbol{\Sigma} = E[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T]$$

$$\hat{\boldsymbol{\Sigma}}_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n-\hat{\boldsymbol{\mu}}_{\mathrm{ML}})(\mathbf{x}_n-\hat{\boldsymbol{\mu}}_{\mathrm{ML}})^T$$

Correlation matrix

The covariance matrix is not scale-independent: Define the correlation matrix R of correlation coefficients ρ_{ij}:

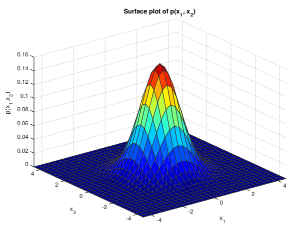
$$R = (\rho_{ij})$$

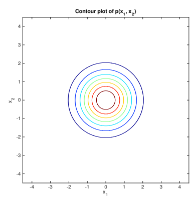
$$\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}}$$

$$\rho_{ii} = 1$$

- Scale-independent (ie independent of the measurement units) and location-independent, ie:
  $$\rho(x_i, x_j) = \rho(ax_i + b, cx_j + d) \qquad \text{for } a > 0, c > 0$$
- The correlation coefficient satisfies $-1 \leq \rho \leq 1$, and
  $$\rho(x, y) = +1 \quad \text{if } y = ax + b \quad a > 0$$
  $$\rho(x, y) = -1 \quad \text{if } y = ax + b \quad a < 0$$

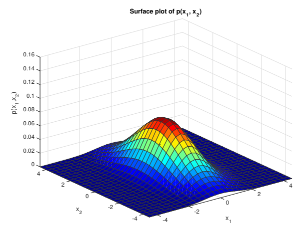
Spherical Gaussian

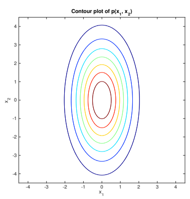




$$\boldsymbol{\mu} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad R = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

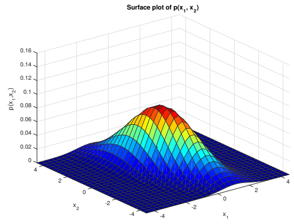
2-D Gaussian with a diagonal covariance matrix

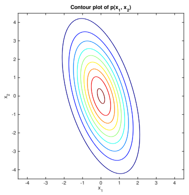




$$\boldsymbol{\mu} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix} \qquad R = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

2-D Gaussian with a full covariance matrix



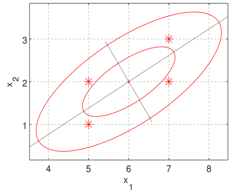


$$\boldsymbol{\mu} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & -1 \\ -1 & 4 \end{pmatrix} \quad R = \begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix}$$

Example of parameter estimation of a 2D Gaussian

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n, \qquad \hat{\boldsymbol{\Sigma}} = \frac{1}{N}\sum_{n=1}^{N} (\mathbf{x}_n - \hat{\boldsymbol{\mu}})(\mathbf{x}_n - \hat{\boldsymbol{\mu}})^T$$

$$\mathbf{x}: \begin{pmatrix} 5 \\ 1 \end{pmatrix}, \begin{pmatrix} 5 \\ 2 \end{pmatrix}, \begin{pmatrix} 7 \\ 2 \end{pmatrix}, \begin{pmatrix} 7 \\ 3 \end{pmatrix}$$



$$\boldsymbol{\mu} = \frac{1}{4}\left\{ \begin{bmatrix} 5 \\ 1 \end{bmatrix} + \begin{bmatrix} 5 \\ 2 \end{bmatrix} + \begin{bmatrix} 7 \\ 2 \end{bmatrix} + \begin{bmatrix} 7 \\ 3 \end{bmatrix} \right\} = \begin{bmatrix} 6 \\ 2 \end{bmatrix}$$

$$\mathbf{x}_n - \boldsymbol{\mu}: \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\boldsymbol{\Sigma} = \frac{1}{4}\left\{ \begin{bmatrix} -1 \\ -1 \end{bmatrix}[-1,-1] + \begin{bmatrix} -1 \\ 0 \end{bmatrix}[-1,0] + \begin{bmatrix} 1 \\ 0 \end{bmatrix}[1,0] + \begin{bmatrix} 1 \\ 1 \end{bmatrix}[1,1] \right\} = \begin{pmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

Example *(cont.)*

$$\hat{\mu}_i = \frac{1}{N}\sum_{n=1}^{N} x_{ni}, \qquad \hat{\sigma}_{ij} = \frac{1}{N}\sum_{n=1}^{N} (x_{ni} - \hat{\mu}_i)(x_{nj} - \hat{\mu}_j)$$

$$\mathbf{x}: \begin{pmatrix} 5 \\ 1 \end{pmatrix}, \begin{pmatrix} 5 \\ 2 \end{pmatrix}, \begin{pmatrix} 7 \\ 2 \end{pmatrix}, \begin{pmatrix} 7 \\ 3 \end{pmatrix}$$

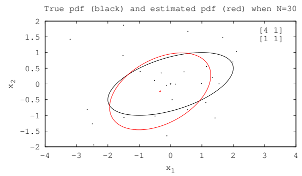
$$\mu_1 = \tfrac{1}{4}(5+5+7+7) = 6$$
$$\mu_2 = \tfrac{1}{4}(1+2+2+3) = 2$$

$$\mathbf{x} - \boldsymbol{\mu}: \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

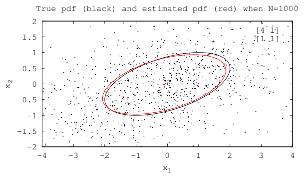
$$\boldsymbol{\Sigma}: \quad \sigma_{11} = \tfrac{1}{4}((-1)^2 + (-1)^2 + 1^2 + 1^2) = 1$$
$$\sigma_{12} = \tfrac{1}{4}((-1)\cdot(-1) + (-1)\cdot 0 + 1\cdot 0 + 1\cdot 1) = \tfrac{1}{2}$$
$$\sigma_{22} = \tfrac{1}{4}((-1)^2 + 0^2 + 0^2 + 1^2) = \tfrac{1}{2}$$

Practical issues

Parameter estimation of multivariate Gaussian distribution can be difficult.



$N = 30$



$N = 1000$

Exercise

- Try Q3, Q4, Q5 in Tutorial 3
- Try Q3 in Tutorial 4
- Try Q4 in Tutorial 4, and
  - Find $\boldsymbol{\Sigma}_i^{-1}$ for $i = 1, 2$.
  - Find $|\boldsymbol{\Sigma}_i|$ for $i = 1, 2$.
  - Find the correlation matrix for each class.
  - What the covariance matrix and pdf will be if the naive Bayes assumption is applied?

Exercise *(cont.)*

Additional to Q3 in Tutorial 4:

The sample variance (σ^2_{ML}) is the maximum likelihood estimate for the variance parameter of a one-dimensional Gaussian. Consider the log likelihood of a set of N data points $x_1, \ldots, x_N$ being generated by a Gaussian with the mean μ and variance σ^2.

$$L = \ln p(\{x_1, \ldots, x_N\} \,|\, \mu, \sigma^2) = -\frac{1}{2}\sum_{n=1}^{N}\left(\frac{(x_n - \mu)^2}{\sigma^2} + \ln \sigma^2 + \ln(2\pi) \right)$$

Assuming that the mean μ is know, show that that maximum likelihood estimate for the variance is indeed the sample variance.

Summary

Gaussians

- Continuous random variable: cumulative distribution function and probability density function
- Univariate Gaussian pdf:
  $$p(x \,|\, \mu, \sigma^2) = N(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2} \right)$$
- Multivariate Gaussian pdf:
  $$p(\mathbf{x} \,|\, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right)$$
- Estimate parameters (mean and covariance matrix) using maximum likelihood estimation
- Try Lab-6 (next week)

Inf2b - Learning

Lecture 9: Classification with Gaussians

Hiroshi Shimodaira
(Credit: Iain Murray and Steve Renals)

Centre for Speech Technology Research (CSTR)
School of Informatics
University of Edinburgh

http://www.inf.ed.ac.uk/teaching/courses/inf2b/
https://piazza.com/ed.ac.uk/spring2020/infr08028
Office hours: Wednesdays at 14:00-15:00 in IF-3.04

Jan-Mar 2020

Today's Schedule

Classification with Gaussians

The multidimensional Gaussian distribution

- The D-dimensional vector $\mathbf{x} = (x_1, \ldots, x_D)^T$ is multivariate Gaussian if it has a probability density function of the following form:

$$p(\mathbf{x} \mid \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right).$$

  The pdf is parameterised by the mean vector $\boldsymbol{\mu}$ and the covariance matrix Σ.
- The 1-dimensional Gaussian is a special case of this pdf
- The argument to the exponential $\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})$ is referred to as a *quadratic form*, and it is always *non-negative*.

Covariance matrix

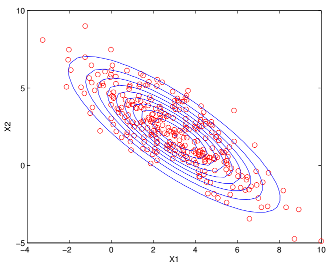
Covariance matrix (with ML estimation):

$$\Sigma = \begin{pmatrix} \sigma_{11} & \cdots & \sigma_{1D} \\ \vdots & \ddots & \vdots \\ \sigma_{D1} & \cdots & \sigma_{DD} \end{pmatrix} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu})(\mathbf{x}_n - \boldsymbol{\mu})^T$$

where $\mathbf{x}_n = (x_{n1}, \ldots, x_{nD})^T$, $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_D)^T$

- Symmetric : $\Sigma^T = \Sigma$, and $(\Sigma^{-1})^T = \Sigma^{-1}$
- Semi-positive definite: $\mathbf{x}^T\Sigma\,\mathbf{x} \geq 0$, and $\mathbf{x}^T\Sigma^{-1}\mathbf{x} \geq 0$
- cf: sample covariance matrix, which uses $\frac{1}{N-1}$.

Maximum likelihood fit to a Gaussian



Tips on calculating covariance matrices

MATLAB is optimised for matrix/vector operations

$$\underset{(D\times D)}{\Sigma} = \frac{1}{N}\sum_{n=1}^{N}\underset{(D\times 1)}{(\mathbf{x}_n - \boldsymbol{\mu})}\underset{(1\times D)}{(\mathbf{x}_n - \boldsymbol{\mu})^T}$$

$$= \frac{1}{N}\underset{(D\times N)}{(\mathbf{x}_1 - \boldsymbol{\mu}, \ldots, \mathbf{x}_N - \boldsymbol{\mu})}\underset{(N\times D)}{\begin{pmatrix} \mathbf{x}_1^T - \boldsymbol{\mu}^T \\ \vdots \\ \mathbf{x}_N^T - \boldsymbol{\mu}^T \end{pmatrix}}$$

$$= \frac{1}{N}\underset{(D\times N)}{(X - M_N)^T}\underset{(N\times D)}{(X - M_N)}$$

$$\underset{(N\times D)}{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} = \begin{bmatrix} x_{11}, & \ldots, & x_{1D} \\ \vdots & & \vdots \\ x_{N1}, & \ldots, & x_{ND} \end{bmatrix}, \quad \underset{(N\times D)}{M_N} = \begin{bmatrix} M \\ \vdots \\ M \end{bmatrix} = \begin{bmatrix} \mu_1, & \ldots, & \mu_D \\ \vdots & & \vdots \\ \mu_1, & \ldots, & \mu_D \end{bmatrix}$$

$$\underset{(1\times D)}{M} = \boldsymbol{\mu}^T = [\ \mu_1,\ \ldots,\ \mu_D\ ], \qquad = \frac{1}{N}1_{NN}X$$

Properties of covariance matrix

$$\Sigma = V\,D\,V^T$$

$$= \begin{pmatrix} v_{11} & \cdots & v_{1D} \\ \vdots & \ddots & \vdots \\ v_{D1} & \cdots & v_{DD} \end{pmatrix}\begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_D \end{pmatrix}\begin{pmatrix} v_{11} & \cdots & v_{1D} \\ \vdots & \ddots & \vdots \\ v_{D1} & \cdots & v_{DD} \end{pmatrix}^T$$

$$= (\mathbf{v}_1, \ldots, \mathbf{v}_D)\ \mathrm{Diag}(\lambda_1, \ldots, \lambda_D)\ (\mathbf{v}_1, \ldots, \mathbf{v}_D)^T$$

- $\mathbf{v}_i$: eigen vector, λ_i : eigen value

  $\Sigma\,\mathbf{v}_i = \lambda_i\,\mathbf{v}_i$
- $\lambda_i \geq 0$, $\quad \|\mathbf{v}_i\| = 1$
- $|\Sigma| = \prod_{i=1}^{D}\lambda_i$
- $\sum_{i=1}^{D}\sigma_{ii} = \sum_{i=1}^{D}\lambda_i$

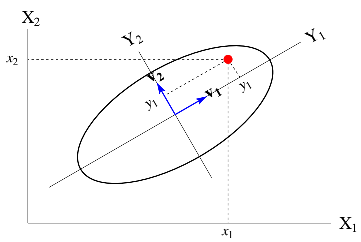
Properties of covariance matrix

- $\mathrm{rank}(\Sigma)$
  - the number of linearly independent columns (or rows)
  - the number of bases (i.e. the dimension of the column space)

$$\mathrm{rank}(\Sigma) = D \quad\rightarrow\quad \forall_i\ :\ \lambda_i > 0$$
$$\forall_{i\neq j}\ :\ \mathbf{v}_i \perp \mathbf{v}_j$$
$$|\Sigma| > 0$$

$$\mathrm{rank}(\Sigma) < D \quad\rightarrow\quad \exists_i\ :\ \lambda_i = 0$$
$$\exists_{(i,j)}\ :\ \rho(x_i, x_j) = 1$$
$$|\Sigma| = 0$$

Geometry of covariance matrix



Sort eigen values: $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_D$

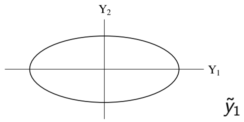
$\mathbf{v}_1$: eigen vector of λ_1

$\mathbf{v}_2$: eigen vector of λ_2

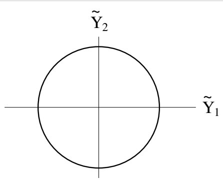
$y_1 = \mathbf{v}_1^T\mathbf{x}$, $\quad \mathrm{Var}(y_1) = \lambda_1$

$y_2 = \mathbf{v}_2^T\mathbf{x}$, $\quad \mathrm{Var}(y_2) = \lambda_2$

Geometry of covariance matrix



$\longrightarrow$



$\tilde{y}_1 = y_1/\sqrt{\lambda_1}$

$\tilde{y}_2 = y_2/\sqrt{\lambda_2}$

$$(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}) = (\tilde{\mathbf{y}}-\tilde{\mathbf{u}})^T(\tilde{\mathbf{y}}-\tilde{\mathbf{u}}) = ||\tilde{\mathbf{y}}-\tilde{\mathbf{u}}||^2 \quad (\dagger)$$

$$\text{where}\quad \tilde{\mathbf{u}} = \left(\frac{\mathbf{v}_1}{\sqrt{\lambda_1}}, \frac{\mathbf{v}_2}{\sqrt{\lambda_2}}\right)^T\boldsymbol{\mu} = \left(\frac{\mathbf{v}_1^T\boldsymbol{\mu}}{\sqrt{\lambda_1}}, \frac{\mathbf{v}_2^T\boldsymbol{\mu}}{\sqrt{\lambda_2}}\right)^T$$

Problems with the estimation of covariance matrix

- $|\Sigma| \rightarrow 0$ when
  - N is not large enough (when compared with D)

    NB: $|\Sigma| = 0$ for $N \leq D$
  - There is high dependence (correlation) among variables (e.g. $\rho(x_i, x_j) \approx 1$)
- Σ^{-1} becomes unstable when $|\Sigma|$ is small.
- Solutions?
  - Share Σ among classes ($\Rightarrow$linear discriminant functions)
  - Assume independence among variables $\Rightarrow$ a diagonal covariance matrix rather than a 'full' covariance matrix.
  - Reduce the dimensionality by transforming the data into a low-dimensional vector space (e.g. PCA).
  - Another regularisation:
    - Add a small positive number to the diagonal elements

      $\Sigma \leftarrow \Sigma + \epsilon I$

Shared covariance matrix among classes

- How to estimate the shared covariance:

  $\Sigma_k = \Sigma$ for all $k = 1, \ldots, K$

$$\Sigma = \frac{1}{K}\sum_{k=1}^{K} \Sigma_k$$

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

- Why is the following not good?

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

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

Covariance matrix when naive Bayes is assumed

$$\Sigma = \begin{pmatrix} \sigma_{11} & & 0 \\ & \ddots & \\ 0 & & \sigma_{DD} \end{pmatrix}, \qquad \sigma_{ij} = 0 \text{ for } i \neq j$$

$$p(\mathbf{x} \mid \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right)$$

$$= p(x_1|\mu_1, \sigma_{11}) \cdots p(x_D|\mu_D, \sigma_{DD})$$

$$= \prod_{i=1}^{D} \left\{ \frac{1}{\sqrt{2\pi\sigma_{ii}}} \exp\left(\frac{-(x_i - \mu_i)^2}{2\sigma_{ii}}\right) \right\}$$

Bayes theorem and probability densities

- Rules for probability densities are similar to those for probabilities:

$$p(x, y) = p(x|y)\, p(y)$$

$$p(x) = \int p(x, y)\, dy$$

- We may mix probabilities of discrete variables and probability densities of continuous variables:

$$p(x, Z) = p(x|Z)\, P(Z)$$

- Bayes' theorem for continuous data x and class C:

$$P(C|x) = \frac{p(x|C)\, P(C)}{p(x)}$$

$$P(C|x) \propto p(x|C)\, P(C)$$

Bayes theorem and univariate Gaussians

- If $p(x|C)$ is Gaussian with mean μ and variance σ^2:

$$P(C|x) \propto p(x|C)\, P(C) = N(x; \mu, \sigma^2)\, P(C)$$

$$\propto \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) P(C)$$

- The log likelihood $LL(x|C)$ is:

$$LL(x \mid \mu, \sigma^2) = \ln p(x \mid \mu, \sigma^2)$$

$$= \frac{1}{2}\left(-\ln(2\pi) - \ln\sigma^2 - \frac{(x-\mu)^2}{\sigma^2}\right)$$

- The log posterior probability $\ln P(C|x)$ is:

$$\ln P(C \mid x) \propto LL(x \mid C) + \ln P(C)$$

$$\propto \frac{1}{2}\left(-\ln(2\pi) - \ln\sigma^2 - \frac{(x-\mu)^2}{\sigma^2}\right) + \ln P(C)$$

Log probability ratio (log odds)

For a classification problem of two classes: C_1 and C_2,

$$\ln \frac{P(C_1|x)}{P(C_2|x)} = \ln P(C_1|x) - \ln P(C_2|x)$$

$$= -\frac{1}{2}\left(\frac{(x-\mu_1)^2}{\sigma_1^2} - \frac{(x-\mu_2)^2}{\sigma_2^2} + \ln\sigma_1^2 - \ln\sigma_2^2\right)$$

$$+ \ln P(C_1) - \ln P(C_2)$$

$$\ln P(C_1|x) - \ln P(C_2|x) > 0 \quad \Rightarrow \quad C_1$$

$$\ln P(C_1|x) - \ln P(C_2|x) < 0 \quad \Rightarrow \quad C_2$$

Example: 1-dimensional Gaussian classifier

- Two classes, S and T, with some observations:

Class S	10	8	10	10	11	11
Class T	12	9	15	10	13	13

- Assume that each class may be modelled by a Gaussian. The estimated mean and variance of each pdf with the maximum likelihood (ML) estimation are given as follows:

$$\mu(S) = 10 \quad \sigma^2(S) = 1$$
$$\mu(T) = 12 \quad \sigma^2(T) = 4$$

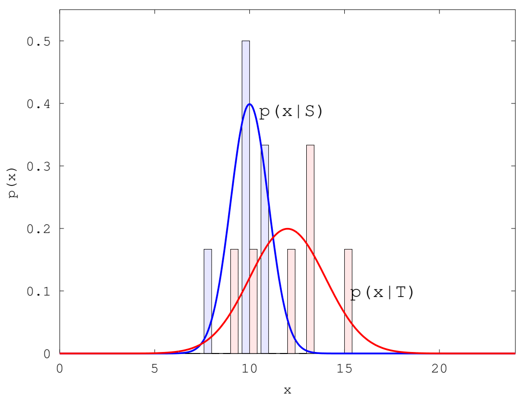
- The following unlabelled data points are available:

  $x_1 = 10, \quad x_2 = 11, \quad x_3 = 6$

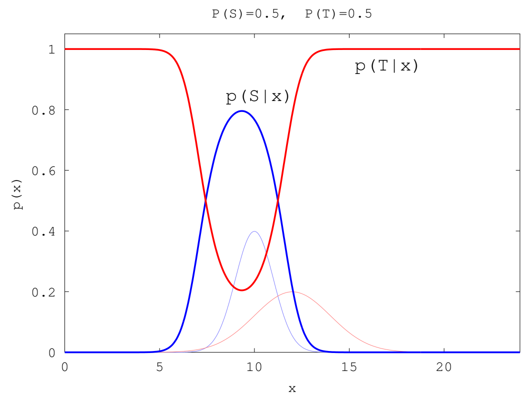
  To which class should each of the data points be assigned?

  Assume the two classes have equal prior probabilities.

Gaussian pdfs for S and T vs histograms



Posterior probabilities



Example: 1-dimensional Gaussian classifier *(cont.)*

- Take the log odds (posterior probability ratios):

$$\ln \frac{P(S|X=x)}{P(T|X=x)} = -\frac{1}{2}\left(\frac{(x-\mu_S)^2}{\sigma_S^2} - \frac{(x-\mu_T)^2}{\sigma_T^2} + \ln\sigma_S^2 - \ln\sigma_T^2\right)$$

$$+ \ln P(S) - \ln P(T)$$

- In the example the priors are equal, so:

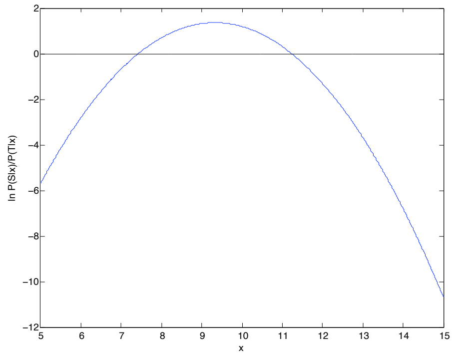
$$\ln \frac{P(S|X=x)}{P(T|X=x)} = -\frac{1}{2}\left(\frac{(x-\mu_S)^2}{\sigma_S^2} - \frac{(x-\mu_T)^2}{\sigma_T^2} + \ln\sigma_S^2 - \ln\sigma_T^2\right)$$

$$= -\frac{1}{2}\left((x-10)^2 - \frac{(x-12)^2}{4} - \ln 4\right)$$

- If log odds are less than 0 assign to T, otherwise assign to S.

Log odds

Test samples: $x_1 = 10$, $x_2 = 11$, $x_3 = 6$



Example: unequal priors

- Now, assume $P(S) = 0.3$, $P(T) = 0.7$. Including this prior information, to which class should each of the above test data points, x_1, x_2, x_3, be assigned?
- Again compute the log odds:

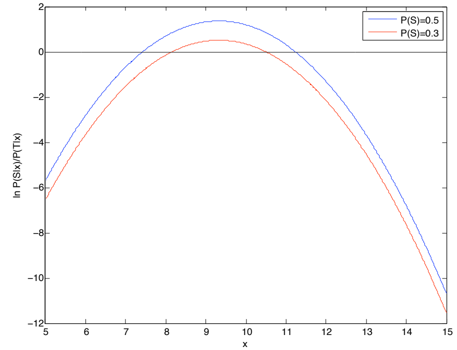
$$\ln \frac{P(S|X=x)}{P(T|X=x)} = -\frac{1}{2}\left(\frac{(x-\mu_s)^2}{\sigma_S^2} - \frac{(x-\mu_T)^2}{\sigma_T^2} + \ln \sigma_S^2 - \ln \sigma_T^2\right) + \ln P(S) - \ln P(T)$$

$$= -\frac{1}{2}\left((x-10)^2 - \frac{(x-12)^2}{4} - \ln 4\right) + \ln P(S) - \ln P(T)$$

$$= -\frac{1}{2}\left((x-10)^2 - \frac{(x-12)^2}{4} - \ln 4\right) + \ln(3/7)$$

Log odds

Test samples: $x_1 = 10$, $x_2 = 11$, $x_3 = 6$



Multivariate Gaussian classifier

- Multivariate Gaussian (in D dimensions):

$$p(\mathbf{x}\,|\,\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

- Log likelihood:

$$LL(\mathbf{x}\,|\,\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{D}{2}\ln(2\pi) - \frac{1}{2}\ln|\boldsymbol{\Sigma}| - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})$$

- Posterior probability: $p(C|\mathbf{x}) \propto p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})P(C)$
- Log posterior probability:

$$\ln P(C\,|\,\mathbf{x}) \propto -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) - \frac{1}{2}\ln|\boldsymbol{\Sigma}| + \ln P(C) + \text{const.}$$

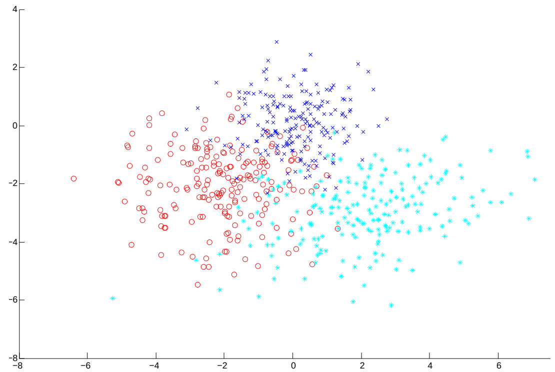
- Try Q4 of Tutorial 4

Example

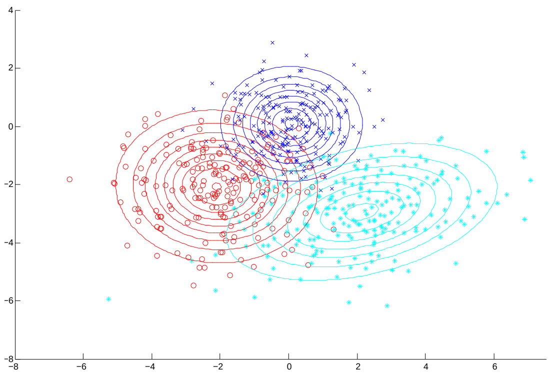
- 2-dimensional data from three classes (A, B, C).
- The classes have equal prior probabilities.
- 200 points in each class
- Load into Matlab ($n \times 2$ matrices, each row is a data point) and display using a scatter plot:

```
xa = load('trainA.dat');
xb = load('trainB.dat');
xc = load('trainC.dat');
hold on;
scatter(xa(:, 1), xa(:,2), 'r', 'o');
scatter(xb(:, 1), xb(:,2), 'b', 'x');
scatter(xc(:, 1), xc(:,2), 'c', '*');
```

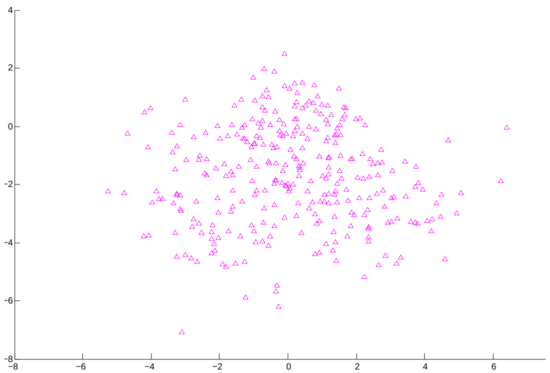

Training data



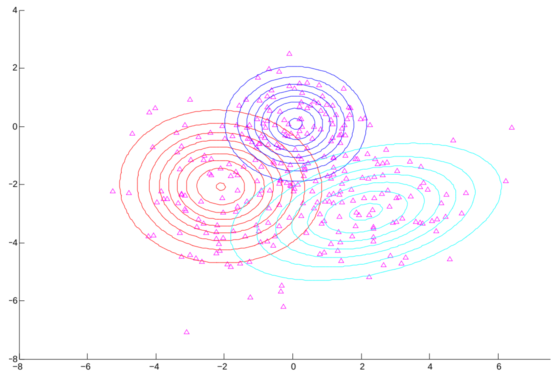
Gaussians estimated from training data



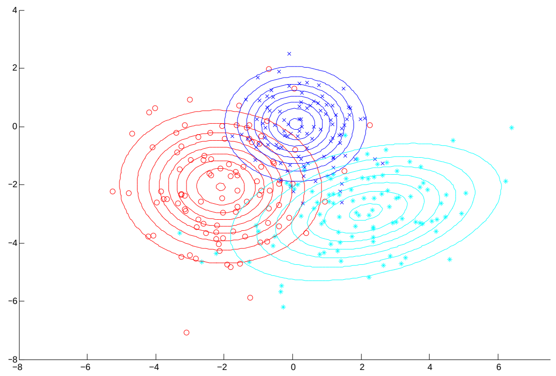
Testing data



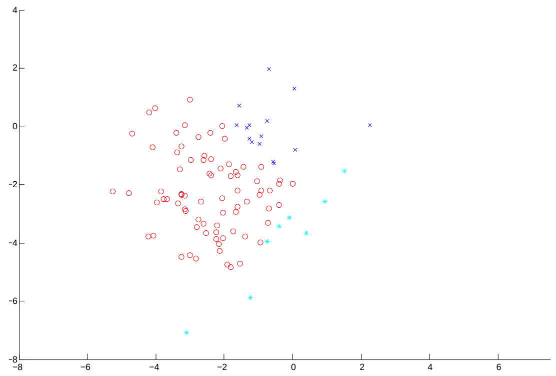
Testing data — with estimated class distributions



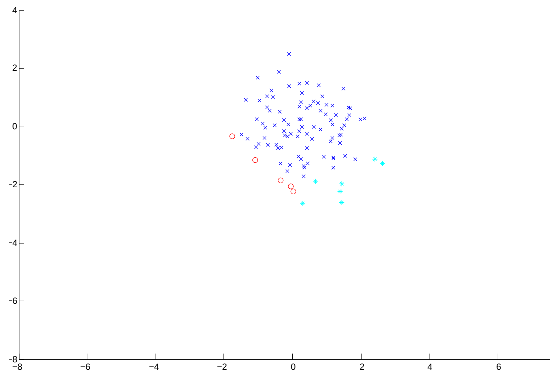
Testing data — with true class indicated



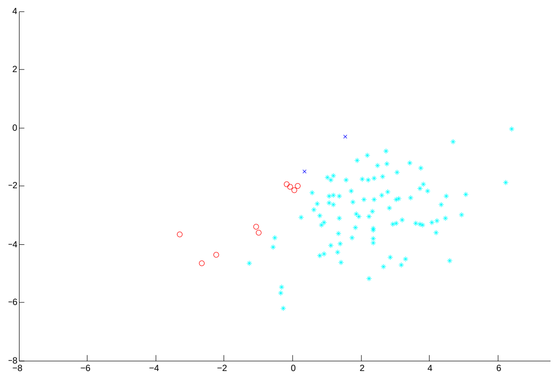
Classifying test data from class A



Classifying test data from class B



Classifying test data from class C



Result

- Analyse the result by percent correct, and in more detail with a confusion matrix
  - Columns of a confusion matrix correspond to the predicted classes (classifier outputs)
  - Rows correspond to the actual (true) class labels
  - Element (r, c) is the number of patterns from true class r that were classified as class c
  - Total number of correctly classified patterns is obtained by summing the numbers on the leading diagonal
- Confusion matrix in this case

		Predicted class		
Test Data		A	B	C
Actual	A	77	15	8
class	B	5	88	7
	C	9	2	89

- Overall proportion of test patterns correctly classified is

$$(77 + 88 + 89)/300 = 254/300 = 0.85$$

Performance measures

- Accuracy (correct classification rate) $= 1 -$ error rate
- Confusion matrix
- Precision, Recall
- F-measure (F1 score)

$$F_1 = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- Receiver operating characteristic (ROC)

NB: measures shown in grey are non-examinable

Example: Classifying spoken vowels

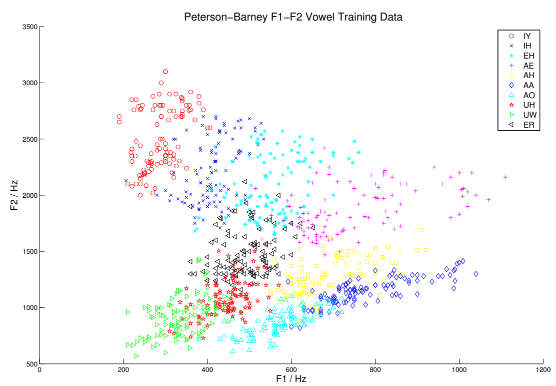
- 10 Spoken vowels in American English
- Vowels can be characterised by formant frequencies — resonances of vocal tract
  - there are usually three or four identifiable formants
  - first two formants written as F1 and F2
- Peterson-Barney data — recordings of spoken vowels by American men, women, and children
  - two examples of each vowel per person
  - for this example, data split into training and test sets
  - children's data not used in this example
  - different speakers in training and test sets
- (see `http://en.wikipedia.org/wiki/Vowel` for more)
- Classify the data using a Gaussian classifier
- Assume equal priors

The data

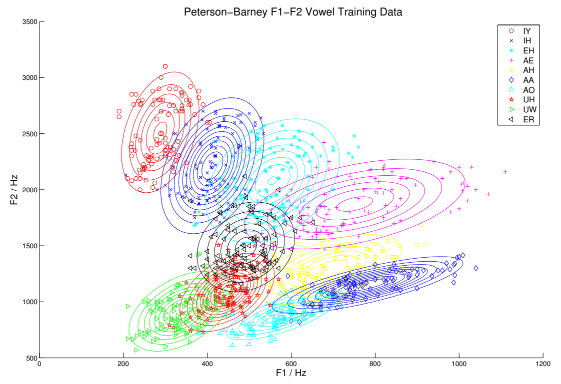
Ten steady-state vowels, frequencies of F1 and F2 at their centre:

- IY — "bee"
- IH — "big"
- EH — "red"
- AE — "at"
- AH — "honey"
- AA — "heart"
- AO — "frost"
- UH — "could"
- UW — "you"
- ER — "bird"

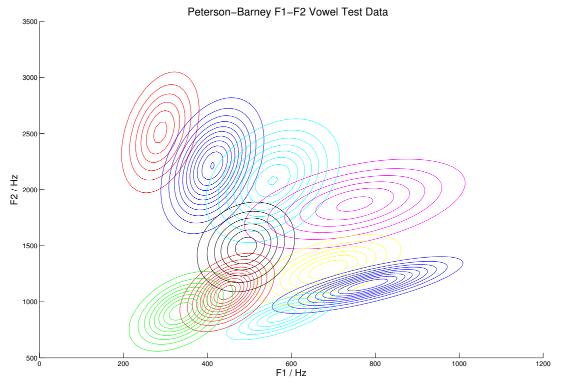
Vowel data — 10 classes



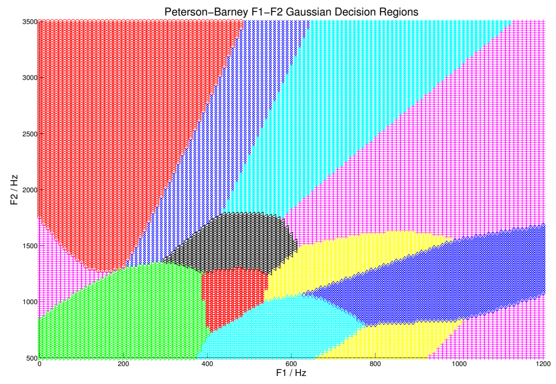
Data and Gaussians for each class



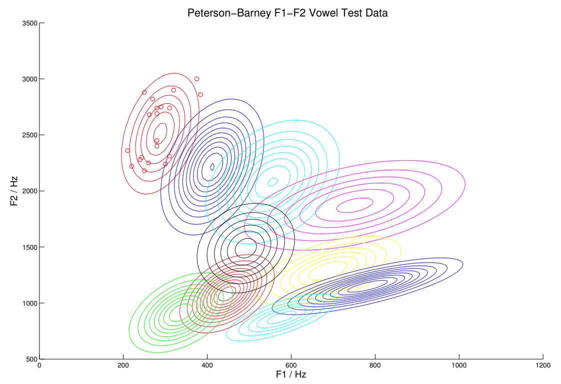
Gaussians for each class



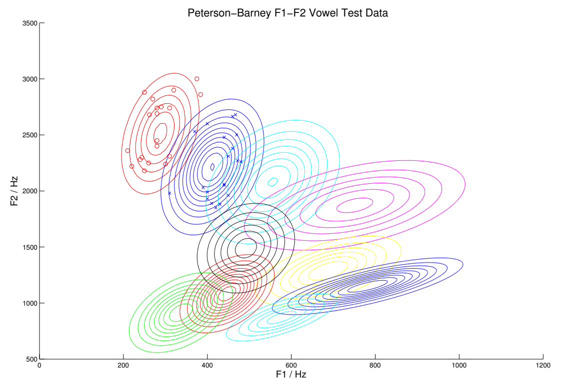
Decision Regions



Test data for class 1 (IY)



Test data for class 2 (IY)



Confusion matrix

	Predicted class										
	IY	IH	EH	AE	AH	AA	AO	UH	UW	ER	% corr.
IY	20	0	0	0	0	0	0	0	0	0	100
IH	0	20	0	0	0	0	0	0	0	0	100
EH	0	0	15	1	0	0	0	0	0	4	75
AE	0	0	3	16	1	0	0	0	0	0	80
AH	0	0	0	0	18	2	0	0	0	0	90
AA	0	0	0	0	2	17	1	0	0	0	85
AO	0	0	0	0	0	4	16	0	0	0	80
UH	0	0	0	0	2	0	0	18	0	0	90
UW	0	0	0	0	0	0	0	5	15	0	75
ER	0	0	0	0	0	0	0	2	0	18	90

Total: 86.5% correct

Exercise

1. Consider estimating a covariance matrix Σ from a data set. Discuss what we could say about the data for the following situations:
   - Σ is almost diagonal (i.e. $\sigma_{ij} \approx 0$ for $i \neq j$).
   - $|\Sigma| \approx 0$.
2. Give examples of data for each situation above.
3. Discuss the minimum number of training samples required to have a covariance matrix that is invertible, i.e. $|\Sigma| \neq 0$. (Hint: think $D = 1$ first, and $D = 2$, and so on.)

Summary

- Covariance matrix
- Using Bayes' theorem with pdfs
- Log probability ratio (log odds)
- The Gaussian classifier: 1-dimensional and multi-dimensional
- Classification examples
- Evaluation measures. Confusion matrix

Familiarise yourself with vector/matrix operations, using pens and papers! (as well as computers)

Inf2b - Learning

Lecture 10: Discriminant functions

Hiroshi Shimodaira
(Credit: Iain Murray and Steve Renals)

Centre for Speech Technology Research (CSTR)
School of Informatics
University of Edinburgh

http://www.inf.ed.ac.uk/teaching/courses/inf2b/
https://piazza.com/ed.ac.uk/spring2020/infr08028
Office hours: Wednesdays at 14:00-15:00 in IF-3.04

Jan-Mar 2020

Today's Schedule

1. Decision Regions
2. Decision Boundaries for minimum error rate classification
3. Discriminant Functions

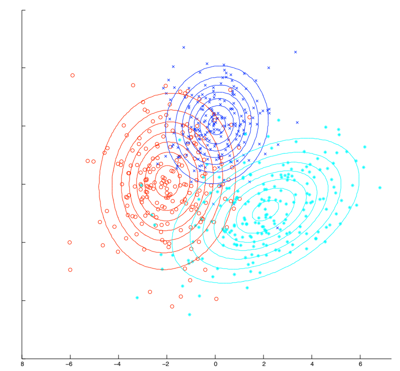
Decision regions

- Recall Bayes' Rule:

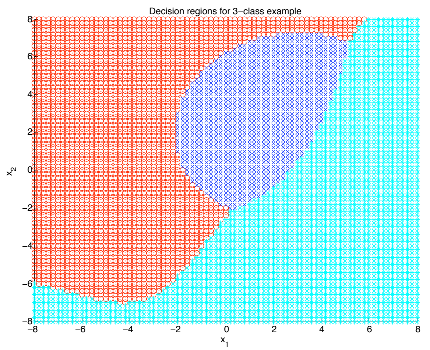
$$P(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)P(C_k)}{p(\mathbf{x})}$$

- Given an unseen point $\mathbf{x}$, we assign to the class for which $P(C_k|x)$ is largest. $(k^* = \arg\max_k P(C_k|x))$
- Thus $\mathbf{x}$-space (the input space) may be regarded as being divided into decision regions $\mathcal{R}_k$ such that a point falling in $\mathcal{R}_k$ is assigned to class C_k.
- Decision region $\mathcal{R}_k$ need not be contiguous, but may consist of several disjoint regions each associated with class C_k.
- The boundaries between these regions are called decision boundaries. (Recall the examples of decision boundaries by k-NN classification in Chapter 4)

Gaussians estimated from data

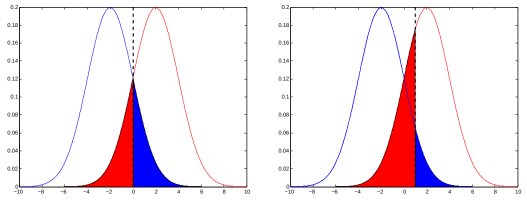


Decision Regions



Placement of decision boundaries

- Consider a 1-dimensional feature space (x) and two classes C_1 and C_2.
- How to place the decision boundary to minimise the probability of misclassification (based on $p(x, C_k)$)?



$\longleftarrow \mathcal{R}_1 \longrightarrow | \longleftarrow \mathcal{R}_2 \longrightarrow$ $\quad$ $\longleftarrow \mathcal{R}_1 \longrightarrow | \longleftarrow \mathcal{R}_2 \longrightarrow$

Decision regions and misclassification

Confusion matrix

In\Out	C_1	C_2
C_1	N_{11}	N_{12}
C_2	N_{21}	N_{22}

$\Rightarrow$

Normalised version

In\Out	C_1	C_2	
C_1	P_{11}	P_{12}	$P_{11}+P_{12}=1$
C_2	P_{21}	P_{22}	$P_{21}+P_{22}=1$

$$P_{11} = P(x \in \mathcal{R}_1|C_1) = \frac{N_{11}}{N_1}, \quad P_{12} = P(x \in \mathcal{R}_2|C_1) = \frac{N_{12}}{N_1}$$

$$P_{21} = P(x \in \mathcal{R}_1|C_2) = \frac{N_{21}}{N_2}, \quad P_{22} = P(x \in \mathcal{R}_2|C_2) = \frac{N_{22}}{N_2}$$

$$N_1 = N_{11}+N_{12}, \ N_2 = N_{21}+N_{22}, \ P(C_1) = \frac{N_1}{N_1+N_2}, \ P(C_2) = \frac{N_2}{N_1+N_2}$$

$$P(\text{correct}) = \frac{N_{11}+N_{22}}{N_1+N_2} = P_{11}\,P(C_1) + P_{22}\,P(C_2)$$

$$P(\text{error}) = \frac{N_{12}+N_{21}}{N_1+N_2} = P_{12}\,P(C_1) + P_{21}\,P(C_2)$$

$$= \int_{\mathcal{R}_2} p(x|C_1)\,P(C_1)\,\mathrm{d}x + \int_{\mathcal{R}_1} p(x|C_2)\,P(C_2)\,\mathrm{d}x$$

Minimising probability of misclassification

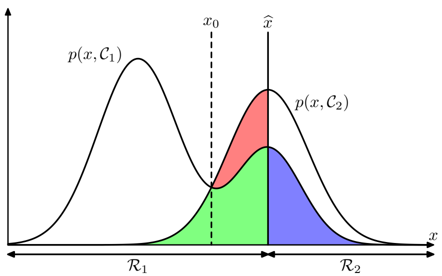
$$P(\text{error}|\mathcal{R}_1, \mathcal{R}_2) = \int_{\mathcal{R}_2} p(x\,|\,C_1)\,P(C_1)\,\mathrm{d}x + \int_{\mathcal{R}_1} p(x\,|\,C_2)\,P(C_2)\,\mathrm{d}x$$

- If there is $x_e \in \mathcal{R}_2$ such that $p(x_e|C_1)P(C_1) > p(x_e|C_2)P(C_2)$, letting $\mathcal{R}_2^* = \mathcal{R}_2 - \{x_e\}$ and $\mathcal{R}_1^* = \mathcal{R}_1 + \{x_e\}$ gives

$$P(\text{error}|\mathcal{R}_1^*, \mathcal{R}_2^*) < P(\text{error}|\mathcal{R}_1, \mathcal{R}_2)$$

- $P(\text{error})$ is minimised by assigning each point to the class with the maximum posterior probability (Bayes decision rule / MAP decision rule / minimum error rate classification).
- This justification for the maximum posterior probability may be extended to D-dimensional feature vectors and K classes

Minimising probability of misclassification *(cont.)*



After Fig. 1.24, C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
$\hat{x}$ denotes the current decision boundary, which causes error shown in red, green, and blue regions. The error is minimised by locating the boundary at x_o.

Discriminant functions

- We can express a classification rule in terms of a discriminant function $y_k(\mathbf{x})$ for each class, such that x is assigned to class C_k if:

$$y_k(\mathbf{x}) > y_\ell(\mathbf{x}) \quad \forall\, \ell \neq k$$

- If we assign $\mathbf{x}$ to class C with the highest posterior probability $P(C|\mathbf{x})$, then the log posterior probability forms a suitable discriminant function:

$$y_k(\mathbf{x}) = \ln p(\mathbf{x}\,|\,C_k) + \ln P(C_k)$$

- Decision boundaries between C_k and C_ℓ are defined when the discriminant functions are equal: $y_k(\mathbf{x}) = y_\ell(\mathbf{x})$
- Decision boundaries are not changed by monotonic transformations (such as taking the log) of the discriminant functions.

Discriminant functions for Gaussian pdfs

- What is the form of the discriminant function when using a Gaussian pdf?

$$p(\mathbf{x}\,|\,\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right)$$

- If the discriminant function is the log posterior probability:

$$y_k(\boldsymbol{x}) = \ln p(\boldsymbol{x}|C_k) + \ln P(C_k)$$

- Then, substituting in the log probability of a Gaussian and dropping constant terms we obtain:

$$y_k(\boldsymbol{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_k| + \ln P(C_k)$$

- This function is quadratic in $\boldsymbol{x}$

Discriminant functions for Gaussian pdfs *(cont.)*

- To see if the function is really quadratic in $\boldsymbol{x}$,

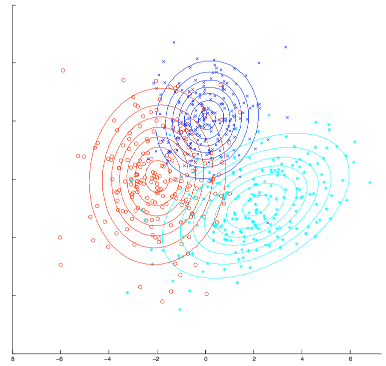
$$\begin{aligned}
&(\mathbf{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) \\
&= \boldsymbol{x}^T\boldsymbol{\Sigma}_k^{-1}\boldsymbol{x} - \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}_k^{-1}\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{\Sigma}_k^{-1}\boldsymbol{\mu}_k + \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}_k^{-1}\boldsymbol{\mu}_k \\
&= \boldsymbol{x}^T\boldsymbol{\Sigma}_k^{-1}\boldsymbol{x} - 2\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}_k^{-1}\boldsymbol{x} + \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}_k^{-1}\boldsymbol{\mu}_k
\end{aligned}$$

- In 2-D case, let $\boldsymbol{\Sigma}_k^{-1} = A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$,

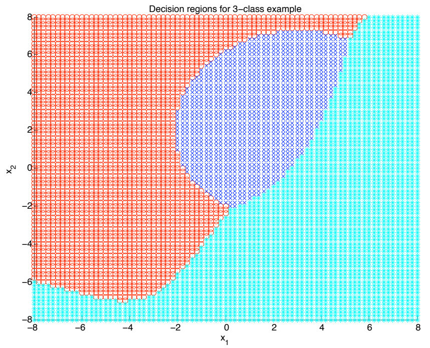
$$\begin{aligned}
\boldsymbol{x}^T\boldsymbol{\Sigma}_k^{-1}\boldsymbol{x} &= \boldsymbol{x}^T A\,\boldsymbol{x} \\
&= \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\
&= a_{11}x_1^2 + (a_{12}+a_{21})x_1x_2 + a_{22}x_2^2
\end{aligned}$$

See Note 10 for details.

Gaussians estimated from training data



Decision Regions



Gaussians with equal covariance

$$\begin{aligned}
y_k(\mathbf{x}) &= -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_k| + \ln P(C_k) \\
&= -\frac{1}{2}(\boldsymbol{x}^T\boldsymbol{\Sigma}_k^{-1}\boldsymbol{x} - 2\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}_k^{-1}\mathbf{x} + \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}_k^{-1}\boldsymbol{\mu}_k) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_k| + \ln P(C_k)
\end{aligned}$$

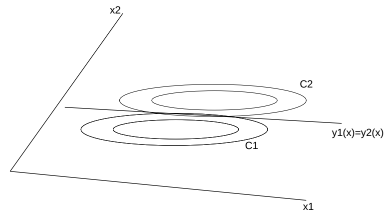
- Consider the special case in which the Gaussian pdfs for each class all share the same class-independent covariance matrix: $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}, \quad \forall\, C_k$

$$\begin{aligned}
y_k(\mathbf{x}) &= (\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1})\,\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln P(C_k) \\
&= \mathbf{w}_k^T\mathbf{x} + w_{k0} \qquad = w_{kD}x_D + \cdots + w_{k1}x_1 + w_{k0}
\end{aligned}$$

where $\mathbf{w}_k^T = \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}$, $\quad w_{k0} = -\frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln P(C_k)$

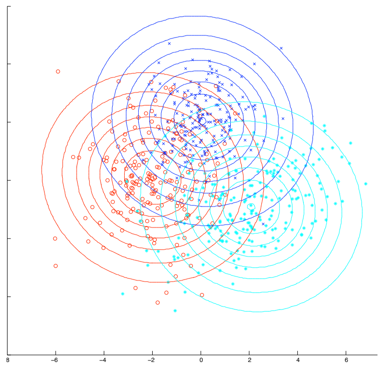
- This is called a linear discriminant function, as it is a linear function of $\mathbf{x}$.

Gaussians with equal covariance *(cont.)*

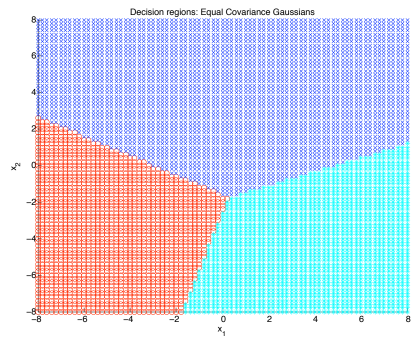


- In two dimensions the boundary is a line
- In three dimensions it is a plane
- In D dimensions it is a hyperplane (i.e. $\{\mathbf{x} \mid \boldsymbol{w}_k^T\mathbf{x} + w_{k0} = 0\}$)

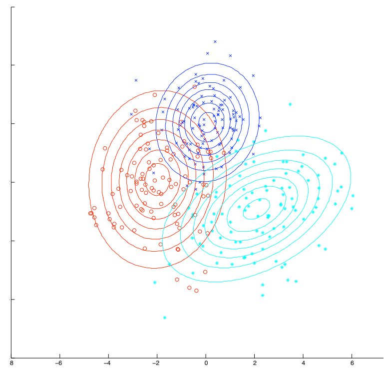
Gaussians estimated from the data: Σ shared



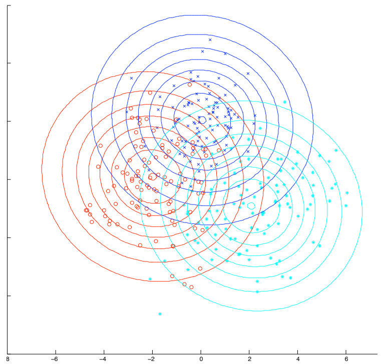
Decision Regions: Σ shared



Testing data (Non-equal covariance)



Testing data (Equal covariance)



Results

- Non-equal covariance Gaussians

		Predicted class		
Test Data		A	B	C
Actual	A	77	15	8
class	B	5	88	7
	C	9	2	89

Fraction correct: $(77 + 88 + 89)/300 = 254/300 \approx 0.85$.

- Equal covariance Gaussians

		Predicted class		
Test Data		A	B	C
Actual	A	80	14	6
class	B	10	90	0
	C	8	6	86

Fraction correct: $(80 + 90 + 86)/300 = 256/300 \approx 0.85$.

Spherical Gaussians with Equal Covariance

- Spherical Gaussians: $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}$

$$\Rightarrow \quad |\boldsymbol{\Sigma}| = \sigma^{2D}, \quad \boldsymbol{\Sigma}^{-1} = \frac{1}{\sigma^2}\mathbf{I}$$

$$y_k(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_k| + \ln P(C_k)$$

$$= -\frac{1}{2\sigma^2}(\mathbf{x}-\boldsymbol{\mu}_k)^T(\mathbf{x}-\boldsymbol{\mu}_k) - \frac{1}{2}\ln\sigma^{2D} + \ln P(C_k)$$

$$y_k(\mathbf{x}) = -\frac{1}{2\sigma^2}\|\mathbf{x}-\boldsymbol{\mu}_k\|^2 + \ln P(C_k)$$

- If equal prior probabilities are assumed,

$$y_k(\mathbf{x}) = -\|\mathbf{x}-\boldsymbol{\mu}_k\|^2$$

The decision rule: "assign a test data to the class whose mean is closest".

The class means ($\boldsymbol{\mu}_k$) may be regarded as class templates or prototypes.

Two-class linear discriminants

- For a two class problem, the log odds can be used as a single discriminant function:

$$y(\mathbf{x}) = \ln\frac{P(C_1|\mathbf{x})}{P(C_2|\mathbf{x})} = \ln\frac{p(\mathbf{x}|C_1)P(C_1)}{p(\mathbf{x}|C_2)P(C_2)}$$

$$= \ln p(\mathbf{x}|C_1) - \ln p(\mathbf{x}|C_2) + \ln P(C_1) - \ln P(C_2)$$

- If the pdf is a Gaussian with the shared covariance matrix, we have a linear discriminant:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

$\mathbf{w}$ and w_0 are functions of $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma, P(C_1)$, and $P(C_2)$.

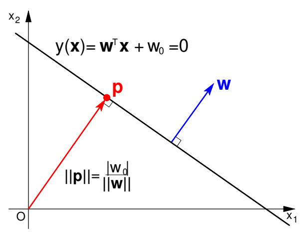
- $\mathbf{w}$ is a normal vector to the decision boundary.

Let $\mathbf{a}$ and $\mathbf{b}$ be two points on the decision boundary

$$\mathbf{w}^T\mathbf{a} + w_0 = \mathbf{w}^T\mathbf{b} + w_0 = 0 \quad \Rightarrow \quad \mathbf{w}^T(\mathbf{a}-\mathbf{b}) = 0$$

$$i.e. \;\; \mathbf{w} \perp (\mathbf{a}-\mathbf{b})$$

Geometry of a two-class linear discriminant



- $\mathbf{w}$ is normal to the decision boundary (hyperplane), $\mathbf{w}^T\mathbf{x} + w_0 = 0$.
- If $\mathbf{p}$ is the point on the hyperplane closest to the origin, then the normal distance from the hyperplane to the origin is given by:

$$\|p\| = \frac{\mathbf{w}^T\mathbf{p}}{\|\mathbf{w}\|} = \frac{|w_0|}{\|\mathbf{w}\|}$$

$$0 = \mathbf{w}^T\mathbf{p} + w_0$$
$$= \|\mathbf{w}\|\|\mathbf{p}\|\cos 0 + w_0$$
$$= \|\mathbf{w}\|\|\mathbf{p}\| \pm w_0$$

Exercise

1. Considering a classification problem of two classes, where each class is modelled with a D-dimensional Gaussian distribution. Derive the formula for the decision boundary, and show that it is quadratic in $\mathbf{x}$.
2. Considering a classification problem of two classes, whose discriminant function takes the form, $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$.
   - Confirm that the decision boundary is a straight line when $D = 2$.
   - Confirm that the weight vector $\mathbf{w}$ is a normal vector to the decision boundary.
3. Try Lab-7 on Classification with Gaussians

Summary

- Obtaining decision boundaries from probability models and a decision rule
- Minimising the probability of error
- Discriminant functions and Gaussian pdfs
- Linear discriminants and Gaussians with equal covariance
- In next lectures, we will see discriminant functions trained with different criteria.

Inf2b - Learning

Lecture 11: Single layer Neural Networks (1)

Hiroshi Shimodaira
(Credit: Iain Murray and Steve Renals)

Centre for Speech Technology Research (CSTR)
School of Informatics
University of Edinburgh

http://www.inf.ed.ac.uk/teaching/courses/inf2b/
https://piazza.com/ed.ac.uk/spring2020/infr08028
Office hours: Wednesdays at 14:00-15:00 in IF-3.04

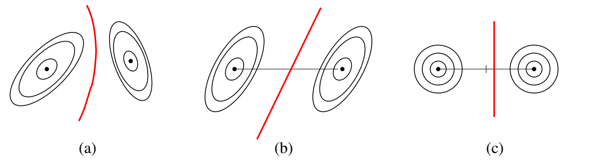
Jan-Mar 2020

Today's Schedule

1. Discriminant functions (recap)
2. Decision boundary of linear discriminants (recap)
3. Discriminative training of linear discriminans (Perceptron)
4. Structures and decision boundaries of Perceptron
5. LSE Training of linear discriminants
6. Appendix - calculus, gradient descent, linear regression

Discriminant functions (recap)

$$y_k(x) = \ln\left(P(x|C)P(C_k)\right)$$
$$= -\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) - \frac{1}{2}\ln|\Sigma_k| + \ln P(C_k)$$
$$= -\frac{1}{2}x^T\Sigma_k^{-1}x + \mu_k^T\Sigma_k^{-1}x - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\ln|\Sigma_k| + \ln P(C_k)$$



Linear discriminants for a 2-class problem

$$y_1(x) = w_1^T x + w_{10}$$
$$y_2(x) = w_2^T x + w_{20}$$

Combined discriminant function:

$$y(x) = y_1(x) - y_2(x) = (w_1 - w_2)^T x + (w_{10} - w_{20})$$
$$= w^T x + w_0$$

Decision:

$$C = \begin{cases} 1, & \text{if } y(x) \geq 0, \\ 2, & \text{if } y(x) < 0 \end{cases}$$

Decision boundary of linear discriminants

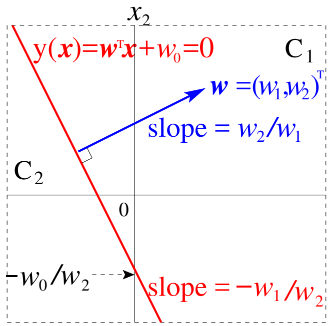
- Decision boundary:
  $y(x) = w^T x + w_0 = 0$

Dimension	Decision boundary	
2	line	$w_1x_1 + w_2x_2 + w_0 = 0$
3	plane	$w_1x_1 + w_2x_2 + w_3x_3 + w_0 = 0$
D	hyperplane	$(\sum_{i=1}^{D} w_ix_i) + w_0 = 0$

NB: w is a normal vector to the hyperplane

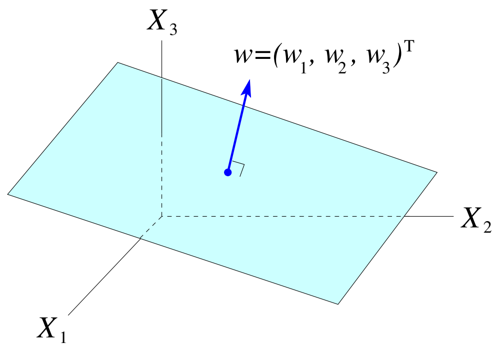
Decision boundary of linear discriminant (2D)

$y(x) = w_1x_1 + w_2x_2 + w_0 = 0 \quad \left(x_2 = -\frac{w_1}{w_2}x_1 - \frac{w_0}{w_2}, \text{ when } w_2 \neq 0\right)$



Decision boundary of linear discriminant (3D)

$y(x) = w_1x_1 + w_2x_2 + w_3x_3 + w_0 = 0$



Approach to linear discminant functions

Generative models : $p(\mathbf{x}|C_k)$

Discriminant function based on Bayes decision rule

$$y_k(\mathbf{x}) = \ln p(\mathbf{x}|C_k) + \ln P(C_k)$$

$\downarrow$ Gaussian pdf (model)

$$y_k(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\mu_k)^T\Sigma_k^{-1}(\mathbf{x}-\mu_k) - \frac{1}{2}\ln|\Sigma_k| + \ln P(C_k)$$

$\downarrow$ Equal covariance assumption

$$y_k(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

$\uparrow$ Why not estimating the decision boundary or $P(C_k|\mathbf{x})$ directly?

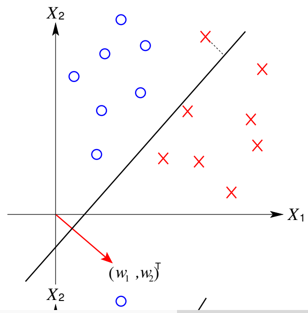
Discriminative training / models

(Logistic regression, Percepton / Neural network, SVM)

Training linear discriminant functions directly

- A discriminant for a two-class problem:

$$y(x) = y_1(x) - y_2(x) = (w_1 - w_2)^T x + (w_{10} - w_{20})$$
$$= w^T x + w_0$$



Perceptron error correction algorithm

$$a(\dot{x}) = w^T x + w_0 = \dot{w}^T\dot{x}$$

where $\dot{w} = (w_0, w^T)^T$, $\dot{x} = (1, x^T)^T$

Let's just use w and x to denote $\dot{w}$ and $\dot{x}$ from now on!

$$y(x) = g(\,a(x)\,) = g(\,w^T x\,) \qquad \text{where } g(a) = \begin{cases} 1, & \text{if } a \geq 0, \\ 0, & \text{if } a < 0 \end{cases}$$

$g(a)$: activation / transfer function

- Training set : $\mathcal{D} = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_N, t_N)\}$
  where $t_i \in \{0, 1\}$: target value
- Modify w if $\mathbf{x}_i$ was misclassified

$$w^{(\text{new})} \leftarrow w + \eta\,(t_i - y(\mathbf{x}_i))\,\mathbf{x}_i \qquad (0 < \eta < 1)$$

learning rate

NB:

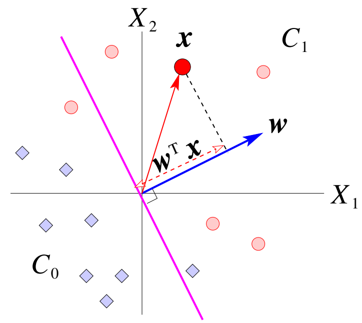
$$(w^{(\text{new})})^T\mathbf{x}_i = w^T\mathbf{x}_i + \eta\,(t_i - y(\mathbf{x}_i))\,\|\mathbf{x}_i\|^2$$

Geometry of Perceptron's error correction

$$y(\mathbf{x}_i) = g(w^T\mathbf{x}_i)$$
$$w^{(\text{new})} \leftarrow w + \eta\,(t_i - y(\mathbf{x}_i))\,\mathbf{x}_i \qquad (0 < \eta < 1)$$

$t_i - y(\mathbf{x}_i)$		$y(\mathbf{x}_i)$ 0	$y(\mathbf{x}_i)$ 1
t_i	0	0	-1
	1	1	0

$w^T x = \|w\|\|x\|\cos\theta$

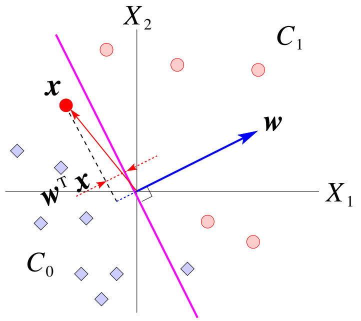


Geometry of Perceptron's error correction *(cont.)*

$$y(\mathbf{x}_i) = g(\boldsymbol{w}^T \mathbf{x}_i)$$
$$\boldsymbol{w}^{(\text{new})} \leftarrow \boldsymbol{w} + \eta\,(t_i - y(\mathbf{x}_i))\,\mathbf{x}_i \qquad (0 < \eta < 1)$$

$t_i - y(\mathbf{x}_i)$		$y(\mathbf{x}_i) = 0$	$y(\mathbf{x}_i) = 1$
t_i	0	0	-1
	1	1	0

$\boldsymbol{w}^T\boldsymbol{x} = \|\boldsymbol{w}\|\|\boldsymbol{x}\| \cos\theta$

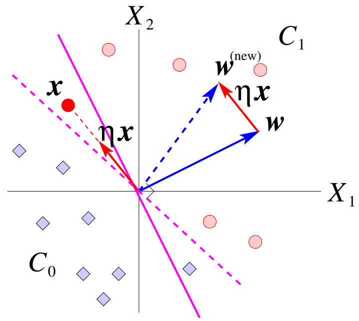


Geometry of Perceptron's error correction *(cont.)*

$$y(\mathbf{x}_i) = g(\boldsymbol{w}^T \mathbf{x}_i)$$
$$\boldsymbol{w}^{(\text{new})} \leftarrow \boldsymbol{w} + \eta\,(t_i - y(\mathbf{x}_i))\,\mathbf{x}_i \qquad (0 < \eta < 1)$$

$t_i - y(\mathbf{x}_i)$		$y(\mathbf{x}_i) = 0$	$y(\mathbf{x}_i) = 1$
t_i	0	0	-1
	1	1	0

$\boldsymbol{w}^T\boldsymbol{x} = \|\boldsymbol{w}\|\|\boldsymbol{x}\| \cos\theta$



The Perceptron learning algorithm

Incremental (online) Perceptron algorithm:

for $i = 1, \ldots, N$
  $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta\,(t_i - y(\mathbf{x}_i))\,\mathbf{x}_i$

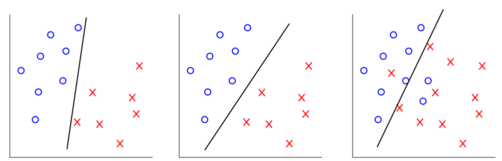
Batch Perceptron algorithm:

$\boldsymbol{v}_{sum} = \mathbf{0}$
for $i = 1, \ldots, N$
  $\boldsymbol{v}_{sum} = \boldsymbol{v}_{sum} + (t_i - y(\mathbf{x}_i))\,\mathbf{x}_i$
$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta\,\boldsymbol{v}_{sum}$

What about convergence?

The Perceptron learning algorithm terminates if training samples are linearly separable.

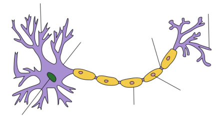
Linearly separable vs linearly non-separable



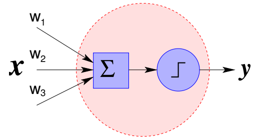
(a–1) (a–2) (b)

Linearly separable Linearly non–separable

Background of Perceptron



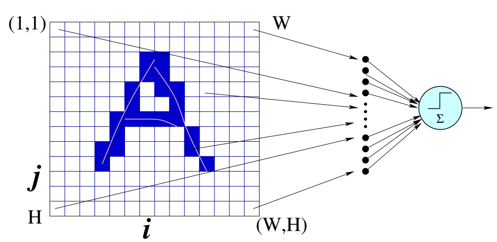
(https://en.wikipedia.org/wiki/File:Neuron_Hand-tuned.svg)



(a) function unit

1940s Warren McCulloch and Walter Pitts : 'threshold logic'
Donald Hebb : 'Hebbian learning'
1957 Frank Rosenblatt : 'Perceptron'

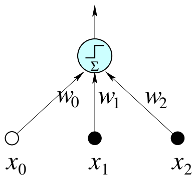
Character recognition by Perceptron

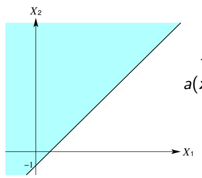


Perceptron structures and decision boundaries

$$y(\boldsymbol{x}) = g(a(\boldsymbol{x})) \qquad \boldsymbol{w} = (w_0, w_1, \ldots, w_D)^T$$
$$= g(\,\boldsymbol{w}^T\boldsymbol{x}\,) \qquad \boldsymbol{x} = (1, x_1, \ldots, x_D)^T$$

where $g(a) = \begin{cases} 1, & \text{if } a \geq 0, \\ 0, & \text{if } a < 0 \end{cases}$





$x_2 \geq x_1 - 1$

$a(\boldsymbol{x}) = 1 - x_1 + x_2$
$\quad = w_0 + w_1 x_1 + w_2 x_2$

$w_0 = 1,\ w_1 = -1,\ w_2 = 1$

NB: A one node/neuron constructs a decision boundary, which splits the input space into two regions

Perceptron as a logical function

NOT

x_1	y
0	1
1	0

OR

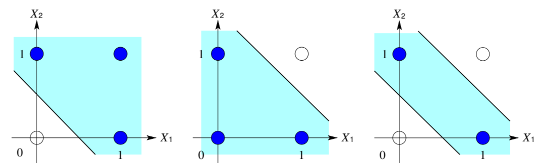
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

NAND

x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0

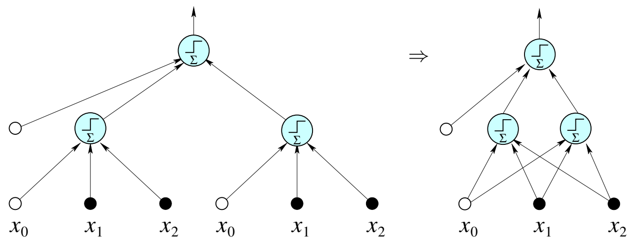
XOR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

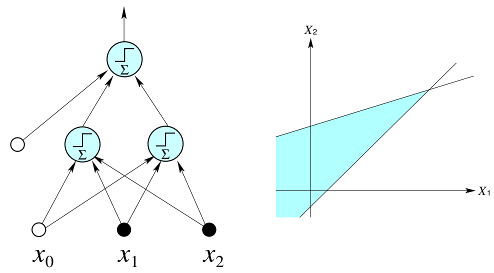


Question: find the weights for each function

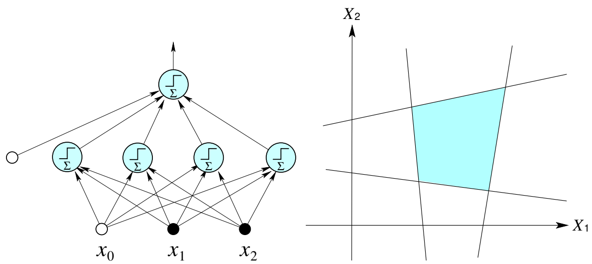
Perceptron structures and decision boundaries *(cont.)*



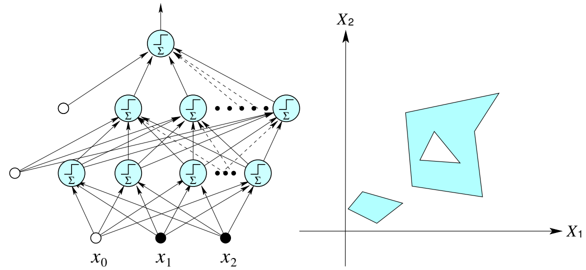
Perceptron structures and decision boundaries (cont.)



Perceptron structures and decision boundaries (cont.)



Perceptron structures and decision boundaries (cont.)



Problems with the Perceptron learning algorithm

- No training algorithms for multi-layer Percepton
- Non-convergence for linearly non-separable data
- Weights $\boldsymbol{w}$ are adjusted for misclassified data only (correctly classified data are not considered at all)

$\Rightarrow$

- Consider not only mis-classification (on train data), but also the optimality of decision boundary
  - Least squares error training
  - Large margin classifiers (e.g. SVM)

Training with least squares

- Squared error function:

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\boldsymbol{w}^T\mathbf{x}_n - t_n\right)^2$$

- Optimisation problem:

$$\min_{\boldsymbol{w}} E(\boldsymbol{w})$$

- One way to solve this is to apply gradient descent (steepest descent):

$$\boldsymbol{w} \;\leftarrow\; \boldsymbol{w} - \eta\, \nabla_{\boldsymbol{w}} E(\boldsymbol{w})$$

where η : step size (a small positive const.)

$$\nabla_{\boldsymbol{w}} E(\boldsymbol{w}) = \left(\frac{\partial E}{\partial w_0}, \ldots \frac{\partial E}{\partial w_D}\right)^T$$

Training with least squares (cont.)

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{n=1}^{N}\left(\boldsymbol{w}^T\mathbf{x}_n - t_n\right)^2$$

$$= \sum_{n=1}^{N}\left(\boldsymbol{w}^T\mathbf{x}_n - t_n\right)\frac{\partial}{\partial w_i}\boldsymbol{w}^T\mathbf{x}_n$$

$$= \sum_{n=1}^{N}\left(\boldsymbol{w}^T\mathbf{x}_n - t_n\right) x_{ni}$$

- Trainable in linearly non-separable case
- Not robust (sensitive) against errornous data (outliers) far away from the boundary
- More or less a linear discriminant

Appendix – derivatives

- Derivatives of functions of one variable

$$\frac{\mathrm{d}f}{\mathrm{d}x} = f'(x) = \lim_{\epsilon\to 0}\frac{f(x+\epsilon) - f(x)}{\epsilon}$$

e.g., $f(x) = 4x^3$, $f'(x) = 12x^2$

- Partial derivatives of functions of more than one variable

$$\frac{\partial f}{\partial x} = \lim_{\epsilon\to 0}\frac{f(x+\epsilon, y) - f(x,y)}{\epsilon}$$

e.g., $f(x,y) = y^3x^2$, $\frac{\partial f}{\partial x} = 2y^3x$

Derivative rules

	Function	Derivative
Constant	c	0
	x	1
Power	x^n	nx^{n-1}
	$\frac{1}{x}$	$-\frac{1}{x^2}$
	$\sqrt{x}$	$\frac{1}{2}x^{-\frac{1}{2}}$
Exponential	e^x	e^x
Logarithms	$\ln(x)$	$\frac{1}{x}$
Sum rule	$f(x) + g(x)$	$f'(x) + g'(x)$
Product rule	$f(x)g(x)$	$f'(x)g(x) + f(x)g'(x)$
Reciprocal rule	$\frac{1}{f(x)}$	$-\frac{f'(x)}{f^2(x)}$
	$\frac{f(x)}{g(x)}$	$-\frac{f'(x)g(x)-f(x)g'(x)}{g^2(x)}$
Chain rule	$f(g(x))$	$f'(g(x))g'(x)$
	$z = f(y), y = g(x)$	$\frac{\mathrm{d}z}{\mathrm{d}x} = \frac{\mathrm{d}z}{\mathrm{d}y}\frac{\mathrm{d}y}{\mathrm{d}x}$

Vectors of derivatives

Consider $f(\boldsymbol{x})$, where $\boldsymbol{x} = (x_1, \ldots, x_D)^T$

Notation: all partial derivatives put in a vector:

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_D}\right)^T$$

Example: $f(\boldsymbol{x}) = x_1^3 x_2^2$

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \begin{pmatrix} 3x_1^2x_2^2 \\ 2x_1^3x_2 \end{pmatrix}$$

Fact: $f(\boldsymbol{x})$ changes most quickly in direction $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$

Gradient descent (steepest descent)

- First order optimisation algorithm using $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$
- Optimisation problem: $\min_{\boldsymbol{x}} f(\boldsymbol{x})$
- Useful when analytic solutions (closed forms) are not available or difficult to find
- Algorithm
  1. Set an initial value $\boldsymbol{x}_0$ and set $t = 0$
  2. If $\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_t)\| \simeq 0$, then stop. Otherwise, do the following.
  3. $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_t)$ for $\eta > 0$
  4. $t = t+1$, and go to step 2.
- Problem: stops at a local minimum (difficult to find a global maximum).

Linear regression (one variable) least squares line fitting

- Training set: $\mathcal{D} = \{(x_n, t_n)\}_{n=1}^N$
- Linear regression: $\hat{t}_n = a x_n + b$
- Objective function: $E = \sum_{n=1}^N (t_i - (a x_i + b))^2$
- Optimisation problem: $\min_{a,b} E$
- Partial derivatives:

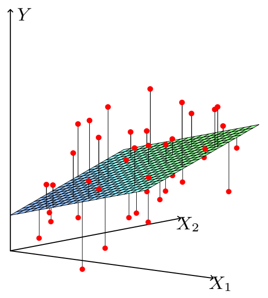
$$\frac{\partial E}{\partial a} = 2\sum_{n=1}^N ((t_i - (a x_i + b))(-x_i)$$
$$= 2a\sum_{n=1}^N x_i^2 + 2b\sum_{n=1}^N x_i - 2\sum_{n=1}^N t_i x_i$$
$$\frac{\partial E}{\partial b} = -2\sum_{n=1}^N ((t_i - (a x_i + b))$$
$$= 2a\sum_{n=1}^N x_i + 2b\sum_{n=1}^N 1 - 2\sum_{n=1}^N t_i$$

Linear regression (one variable) (cont.)

Letting $\frac{\partial E}{\partial a} = 0$ and $\frac{\partial E}{\partial b} = 0$

$$\begin{pmatrix} \sum_{n=1}^N x_i^2 & \sum_{n=1}^N x_i \\ \sum_{n=1}^N x_i & \sum_{n=1}^N 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{n=1}^N t_i x_i \\ \sum_{n=1}^N t_i \end{pmatrix}$$

Linear regression (multiple variables)



- Training set: $\mathcal{D} = \{(\boldsymbol{x}_n, t_n)\}_{n=1}^N$, where $\boldsymbol{x}_n = (1, x_1, \ldots, x_D)^T$
- Linear regression: $\hat{t}_n = \boldsymbol{w}^T \boldsymbol{x}_n$
- Objective function: $E = \sum_{n=1}^N (t_n - \boldsymbol{w}^T \boldsymbol{x}_n)^2$
- Optimisation problem: $\min_{a,b} E$

Elements of Statistical Learning (2nd Ed.) © Hastie, Tibshirani & Friedman 2009

Linear regression (multiple variables) (cont.)

- $E = \sum_{n=1}^N (t_n - \boldsymbol{w}^T \boldsymbol{x}_n)^2$
- Partial derivatives: $\frac{\partial E}{\partial w_i} = -2\sum_{n=1}^N (t_n - \boldsymbol{w}^T \boldsymbol{x}_n) x_{ni}$
- Vector/matrix representation *(NE)* :

$$X = \begin{bmatrix} x_1^T \\ \vdots \\ x_N^T \end{bmatrix} = \begin{bmatrix} x_{10}, & \ldots, & x_{1d} \\ \vdots & & \vdots \\ x_{N0}, & \ldots, & x_{Nd} \end{bmatrix}, \; T = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix}$$

$$E = (T - X\boldsymbol{w})^T (T - X\boldsymbol{w})$$

$$\frac{\partial E}{\partial \boldsymbol{w}} = -2X^T(T - XW)$$

Letting $\frac{\partial E}{\partial \boldsymbol{w}} = \boldsymbol{0} \quad \Rightarrow X^T(T - XW) = \boldsymbol{0}$

$X^T X W = X^T T$

$W = (X^T X)^{-1} X^T T$ $\cdots$ analytic solution if the inverse exists

Summary

- Training discriminant functions directly (discriminative training)
- Perceptron training algorithm
  - Perceptron error correction algorithm
  - Least squares error + gradient descent algorithm
- Linearly separable vs linearly non-separable
- Perceptron structures and decision boundaries
- See Notes 11 for a Perceptron with multiple output nodes

Inf2b - Learning

Lectures 12,13: Single layer Neural Networks (2,3)

Hiroshi Shimodaira
(Credit: Iain Murray and Steve Renals)

Centre for Speech Technology Research (CSTR)
School of Informatics
University of Edinburgh

http://www.inf.ed.ac.uk/teaching/courses/inf2b/
https://piazza.com/ed.ac.uk/spring2020/infr08028
Office hours: Wednesdays at 14:00-15:00 in IF-3.04

Jan-Mar 2020

Today's Schedule

1. Perceptron (recap)
2. Problems with Perceptron
3. Extensions of Perceptron
4. Training of a single-layer neural network

Perceptron (recap)

- Input-to-output function

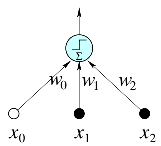
$$a(\dot{\mathbf{x}}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 = \dot{\mathbf{w}}^T \dot{\mathbf{x}}$$
where $\dot{\mathbf{w}} = (w_0, \boldsymbol{w}^T)^T$, $\dot{\mathbf{x}} = (1, \boldsymbol{x}^T)^T$

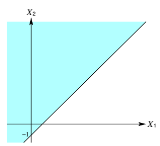
$x_0 = 1$

$$y(\dot{\mathbf{x}}) = g(\,a(\dot{\mathbf{x}})\,) = g(\,\dot{\mathbf{w}}^T \dot{\mathbf{x}}\,)$$

where $g(a) = \begin{cases} 1, & \text{if } a \geq 0, \\ 0, & \text{if } a < 0 \end{cases}$

$g(a)$: activation/transfer function





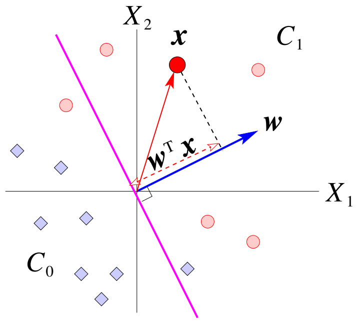
$x_2 \geq x_1 - 1$
$a(\boldsymbol{x}) = 1 - x_1 + x_2$
$= w_0 + w_1 x_1 + w_2 x_2$
$w_0 = 1, w_1 = -1, w_2 = 1$

Geometry of Perceptron's error correction

$$y(\mathbf{x}_i) = g(\boldsymbol{w}^T \mathbf{x}_i)$$
$$\boldsymbol{w}^{(\text{new})} \leftarrow \boldsymbol{w} + \eta\,(t_i - y(\mathbf{x}_i))\,\mathbf{x}_i \qquad (0 < \eta < 1)$$

$t_i - y(\mathbf{x}_i)$		$y(\mathbf{x}_i)$ = 0	$y(\mathbf{x}_i)$ = 1
t_i	0	0	-1
	1	1	0

$\boldsymbol{w}^T \boldsymbol{x} = \|\boldsymbol{w}\| \|\boldsymbol{x}\| \cos\theta$

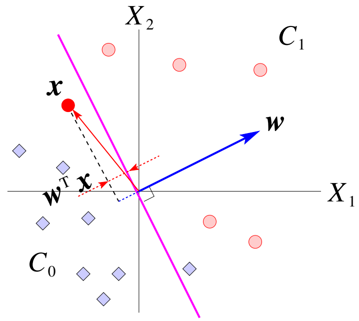


Geometry of Perceptron's error correction (cont.)

$$y(\mathbf{x}_i) = g(\boldsymbol{w}^T \mathbf{x}_i)$$
$$\boldsymbol{w}^{(\text{new})} \leftarrow \boldsymbol{w} + \eta\,(t_i - y(\mathbf{x}_i))\,\mathbf{x}_i \qquad (0 < \eta < 1)$$

$t_i - y(\mathbf{x}_i)$		$y(\mathbf{x}_i)$ = 0	$y(\mathbf{x}_i)$ = 1
t_i	0	0	-1
	1	1	0

$\boldsymbol{w}^T \boldsymbol{x} = \|\boldsymbol{w}\| \|\boldsymbol{x}\| \cos\theta$

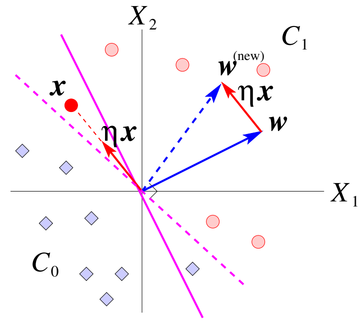


Geometry of Perceptron's error correction (cont.)

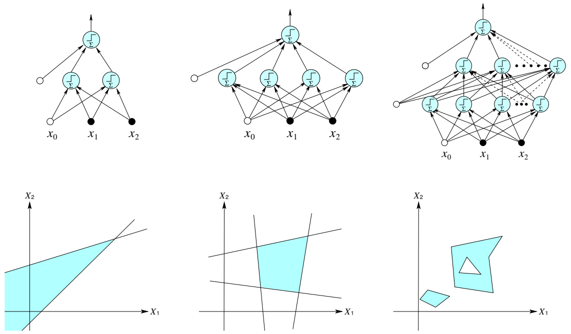
$$y(\mathbf{x}_i) = g(\boldsymbol{w}^T \mathbf{x}_i)$$
$$\boldsymbol{w}^{(\text{new})} \leftarrow \boldsymbol{w} + \eta\,(t_i - y(\mathbf{x}_i))\,\mathbf{x}_i \qquad (0 < \eta < 1)$$

$t_i - y(\mathbf{x}_i)$		$y(\mathbf{x}_i)$ = 0	$y(\mathbf{x}_i)$ = 1
t_i	0	0	-1
	1	1	0

$\boldsymbol{w}^T \boldsymbol{x} = \|\boldsymbol{w}\| \|\boldsymbol{x}\| \cos\theta$



Perceptron structures and decision boundaries

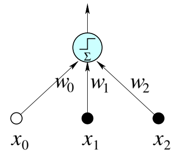


Question: Find the weights for each network

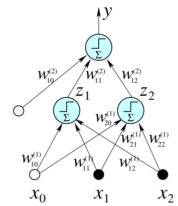
Limitations of Perceptron

- Single-layer perceptron is just a linear classifier (Marvin Minsky and Seymour Papert, 1969)
- Multi-layer perceptron can form complex decision boundaries (piecewise-linear), but it is hard to train
- Training does not stop if data are linearly non-separable
- Weights $\boldsymbol{w}$ are adjusted for misclassified data only (correctly classified data are not considered at all)

A limitation of Perceptron

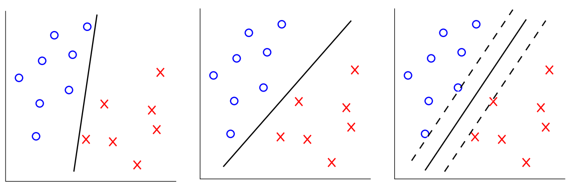


$$y = g(\boldsymbol{w}^T \boldsymbol{x})$$



$$z_1 = g(\mathbf{w}_1^{(1)\,T}\boldsymbol{x}) = g(w_{11}^{(1)}x_1 + w_{12}^{(1)}x_2 + w_{10}^{(1)})$$
$$z_2 = g(\mathbf{w}_2^{(1)\,T}\boldsymbol{x}) = g(w_{21}^{(1)}x_1 + w_{22}^{(1)}x_2 + w_{20}^{(1)})$$
$$y = g(\mathbf{w}^{(2)\,T}\boldsymbol{z}) = g(w_{11}^{(2)}z_1 + w_{12}^{(2)}z_2 + w_{10}^{(2)})$$

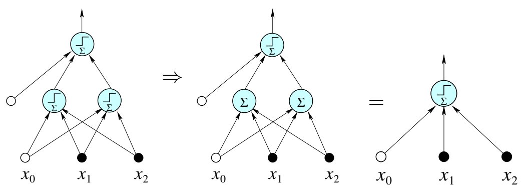
Choices of decision boundaries



(a) (b) (c)

How can we resolve the problem of training?

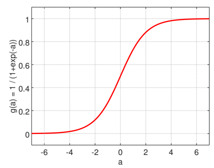
- Use the least squares error criterion for training
$$E_2(\boldsymbol{w}) = \sum_{n=1}^{N} (y_n - t_n)^2$$
- Replace $g(\,)$ with a differentiable function
  What about removing $g(\,)$ in the hidden layer?
$$z_i = g(\mathbf{w}_i^{(1)\,T}\boldsymbol{x}) \quad \Rightarrow \quad z_i = \mathbf{w}_i^{(1)\,T}\boldsymbol{x}$$



Question: Show networks with linear hidden nodes reduce to single-layer networks

How can we resolve the problem of training? (cont.)

- Replace $g(\,)$ with a differentiable non-linear function
  e.g., Logistic sigmoid function:
$$g(a) = \frac{1}{1+e^{-a}} = \frac{1}{1+\exp(-a)}$$



Mapping: $(-\infty, +\infty) \rightarrow (0, 1)$

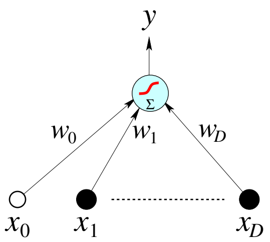
$$\frac{d}{da}g(a) = g'(a) = g(a)\,(1 - g(a))$$

Single Layer Neural Network

Assume a single-layer neural network with a single output node with a logistic sigmoid function:

$$y(\mathbf{x}) = g(\,\mathbf{w}^T\mathbf{x}\,) = g\left(\sum_{i=0}^{D} w_i x_i\right)$$

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



Single Layer Neural Network *(cont.)*

- Training set : $\mathcal{D} = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_N, t_N)\}$
  where $t_i \in \{0,1\}$
- Error function:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\,y_n - t_n\,)^2$$
$$= \frac{1}{2}\sum_{n=1}^{N}\left(g(\mathbf{w}^T\mathbf{x}_n) - t_n\right)^2$$
$$= \frac{1}{2}\sum_{n=1}^{N}\left(g\left(\sum_{i=0}^{D} w_i x_{ni}\right) - t_n\right)^2$$

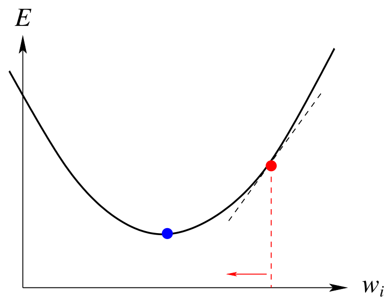
- Definition of the training problem as an optimisation problem
  $\min_{\mathbf{w}} E(\mathbf{w})$

Training of single layer neural network

- Optimisation problem: $\min_{\mathbf{w}} E(\mathbf{w})$
- No analytic solution
- Employ an iterative method (requires initial values)
  e.g. Gradient descent (steepest descent), Newton's method, Conjugate gradient methods
- Gradient descent
  (scalar rep.)
  $$w_i^{(\text{new})} \leftarrow w_i - \eta\frac{\partial}{\partial w_i}E(\mathbf{w}), \qquad (\eta > 0)$$
  (vector rep.)
  $$\mathbf{w}^{(\text{new})} \leftarrow \mathbf{w} - \eta\,\nabla_{\mathbf{w}}E(\mathbf{w}), \qquad (\eta > 0)$$
- Online/stochastic gradient descent (cf. Batch training)
  Update the weights one pattern at a time. (See Note 11)

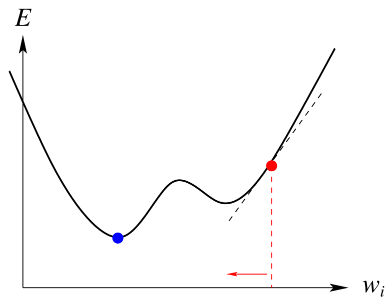
Gradient descent

$$w_i^{(\text{new})} \leftarrow w_i - \eta\frac{\partial}{\partial w_i}E(\mathbf{w}), \qquad (\eta > 0)$$



Local minimum problem with the gradient descent

$$w_i^{(\text{new})} \leftarrow w_i - \eta\frac{\partial}{\partial w_i}E(\mathbf{w}), \qquad (\eta > 0)$$



Training of the single-layer neural network

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\,y_n - t_n\,)^2 = \frac{1}{2}\sum_{n=1}^{N}\left(g\left(\sum_{i=0}^{D} w_i x_{ni}\right) - t_n\right)^2$$

where $y_n = g(a_n)$, $\quad a_n = \sum_{i=0}^{D} w_i x_{ni}$, $\quad \frac{\partial\, a_n}{\partial\, w_i} = x_{ni}$

$$\frac{\partial\, E(\mathbf{w})}{\partial\, w_i} = \frac{\partial\, E(\mathbf{w})}{\partial\, y_n}\frac{\partial\, y_n}{\partial\, a_n}\frac{\partial\, a_n}{\partial\, w_i}$$
$$= \sum_{n=1}^{N}(y_n - t_n)\frac{\partial\, g(a_n)}{\partial\, a_n}\frac{\partial\, a_n}{\partial\, w_i}$$
$$= \sum_{n=1}^{N}(y_n - t_n)\,g'(a_n)\,x_{ni}$$
$$= \sum_{n=1}^{N}(y_n - t_n)\,g(a_n)\,(1 - g(a_n))\,x_{ni}$$

Another training criterion – cross-entropy error

- Training problem with the mean squared error (MSE) criterion with the sigmoid function

$$E_{\text{MSE}}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\,y_n - t_n\,)^2\,, \qquad y_n = g(a_n)$$

$$\frac{\partial\, E_{\text{MSE}}(\mathbf{w})}{\partial\, w_i} = \sum_{n=1}^{N}(y_n - t_n)\,g'(a_n)\,x_{ni}\,, \qquad g'(a) = g(a)(1-g(a))$$

For such a that $g(a) \approx 0$ or 1, $g'(a) \approx 0$.

- Cross-entropy error *(NE)*

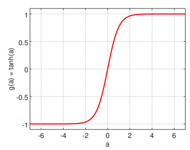
$$E_{\text{H}}(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}\{\,t_n \ln y_n + (1-t_n)\ln(1-y_n)\,\}$$

It can be shown that:

$$\frac{\partial\, E_{\text{H}}(\mathbf{w})}{\partial\, w_i} = \frac{1}{N}\sum_{n=1}^{N}(y_n - t_n)\,x_{ni}$$

Other activation functions *(NE)*

- Tanh



$$g(a) = \tanh(a) = \frac{1 - e^{-2a}}{1 + e^{-2a}}$$

  - Mapping $(-\infty, +\infty) \rightarrow (-1, 1)$
  - 0 (zero) centred $\rightarrow$ faster convergence than sigmoid
- ReLU (Rectified Linear Unit)
  $g(a) = \max(0, a)$
  - Several times faster than tanh.
  - 'Dying ReLU' problem – a unit of outputting 0 always

Exercise

1. Show networks with linear nodes in all hidden layers reduce to single-layer networks.
2. Prove that the derivative of the logistic sigmoid function $g(a)$ is given as $g'(a) = g(a)\,(1 - g(a))$, and sketch the graph of it.
3. Explain about the learning rate η for the gradient descent method.
4. Explain the problem with the training of a neural network with the MSE criterion when the sigmoid function is used as the activation function.
5. *(NE)* Prove that the partial derivative of the cross-entropy error is given as
$$\frac{\partial\, E_{\text{H}}(\mathbf{w})}{\partial\, w_i} = \frac{1}{N}\sum_{n=1}^{N}(y_n - t_n)\,x_{ni}\,.$$

Summary

- Limitations of Perceptron
- Solutions to the problems
- Neural network with differentiable non-linear functions (e.g. logistic sigmoid function)
- Training of the network with the gradient descent algorithm
- Considered only a single-layer network with a single-output node
- A very good reference: http://neuralnetworksanddeeplearning.com/

Inf2b - Learning

Lecture 14: Multi-layer neural networks (1)

Hiroshi Shimodaira
(Credit: Iain Murray and Steve Renals)

Centre for Speech Technology Research (CSTR)
School of Informatics
University of Edinburgh
http://www.inf.ed.ac.uk/teaching/courses/inf2b/
https://piazza.com/ed.ac.uk/spring2020/infr08028
Office hours: Wednesdays at 14:00-15:00 in IF-3.04

Jan-Mar 2020

Today's Schedule

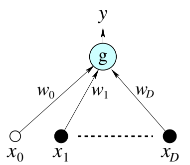
1. Single-layer network with a single output node (recap)
2. Single-layer network with multiple output nodes
3. Multi-layer neural network
4. Activation functions

Single-layer network with a single output node (recap)

- Activation function:

$$y = g(a) = g\Big(\sum_{i=0}^{D} w_i x_i\Big)$$

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



- Training set : $\mathcal{D} = \{(\mathbf{x}_n, t_n)\}_{n=1}^{N}$

  where $t_n \in \{0, 1\}$
- Error function:

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N} (\, y_n - t_n \,)^2$$

- Optimisation problem (training)

$$\min_{\boldsymbol{w}} E(\boldsymbol{w})$$

Training of single layer neural network

- Optimisation problem: $\min_{\boldsymbol{w}} E(\boldsymbol{w})$
- No analytic solution (no closed form)
- Employ an iterative method (requires initial values) e.g. Gradient descent (steepest descent), Newton's method, Conjugate gradient methods
- Gradient descent

  (scalar rep.)

$$w_i^{(\text{new})} \leftarrow w_i - \eta \frac{\partial}{\partial w_i} E(\boldsymbol{w}), \qquad (\eta > 0)$$

  (vector rep.)

$$\boldsymbol{w}^{(\text{new})} \leftarrow \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} E(\boldsymbol{w}), \qquad (\eta > 0)$$

- Online/stochastic gradient descent (cf. Batch training) Update the weights one pattern at a time. (See Note 11)

Training of the single-layer neural network

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N} (\, y_n - t_n \,)^2 \;=\; \frac{1}{2}\sum_{n=1}^{N} (\, g(a_n) - t_n \,)^2$$

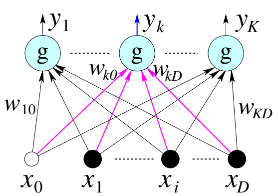
where $y_n = g(a_n), \quad a_n = \sum_{i=0}^{D} w_i x_{ni}, \quad \frac{\partial\, a_n}{\partial\, w_i} = x_{ni}$

$$\frac{\partial\, E(\boldsymbol{w})}{\partial\, w_i} = \frac{\partial\, E(\boldsymbol{w})}{\partial\, y_n}\frac{\partial\, y_n}{\partial\, a_n}\frac{\partial\, a_n}{\partial\, w_i}$$

$$= \sum_{n=1}^{N} (y_n - t_n)\, \frac{\partial\, g(a_n)}{\partial\, a_n}\frac{\partial\, a_n}{\partial\, w_i}$$

$$= \sum_{n=1}^{N} (y_n - t_n)\, g'(a_n)\, x_{ni}$$

Single-layer network with multiple output nodes



- K output nodes: $y_1, \ldots, y_K$.
- For $\mathbf{x}_n = (x_{n0}, \ldots, x_{nD})^T$,

$$y_{nk} \;=\; g\Big(\sum_{i=0}^{D} w_{ki}\, x_{ni}\Big) = g(a_{nk})$$

$$a_{nk} \;=\; \sum_{i=0}^{D} w_{ki}\, x_{ni}$$

Single-layer network with multiple output nodes

- Training set : $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{t}_1), \ldots, (\mathbf{x}_N, \mathbf{t}_N)\}$

  where $\mathbf{t}_n = (t_{n1}, \ldots, t_{nK})$ and $t_{nk} \in \{0, 1\}$
- Error function:

$$E(w) = \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{y}_n - \mathbf{t}_n\|^2 \;=\; \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K} (y_{nk} - t_{nk})^2$$

$$= \sum_{n=1}^{N} E_n, \quad \text{where } E_n = \frac{1}{2}\sum_{k=1}^{K} (y_{nk} - t_{nk})^2$$

- Training by the gradient descent:

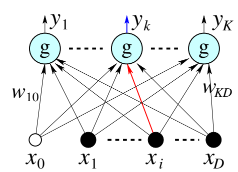
$$w_{ki} \leftarrow w_{ki} - \eta \frac{\partial E}{\partial w_{ki}}, \qquad (\eta > 0)$$

The derivatives of the error function (single-layer)

$$E_n = \frac{1}{2}\sum_{k=1}^{K} (y_{nk} - t_{nk})^2$$

$$y_{nk} \;=\; g(a_{nk})$$

$$a_{nk} \;=\; \sum_{i=0}^{D} w_{ki} x_{ni}$$



$$\frac{\partial E_n}{\partial w_{ki}} = \frac{\partial E_n}{\partial y_{nk}}\frac{\partial y_{nk}}{\partial a_{nk}}\frac{\partial a_{nk}}{\partial w_{ki}}$$

$$= (y_{nk} - t_{nk})\, g'(a_{nk})\, x_{ni}$$

Multi-layer neural networks

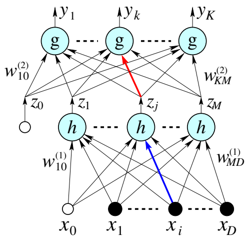
Multi-layer perceptron (MLP)

- Hidden-to-output weights:

$$w_{kj}^{(2)} \leftarrow w_{kj}^{(2)} - \eta \frac{\partial E}{\partial w_{kj}^{(2)}}$$

- Input-to-hidden weights:

$$w_{ji}^{(1)} \leftarrow w_{ji}^{(1)} - \eta \frac{\partial E}{\partial w_{ji}^{(1)}}$$



Training of MLP

1940s Warren McCulloch and Walter Pitts : 'threshold logic'
Donald Hebb : 'Hebbian learning'
1957 Frank Rosenblatt : 'Perceptron'
1969 Marvin Minsky and Seymour Papert : limitations of neural networks
1980 Kunihiro Fukushima: 'Neocognitoron'
1986 D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors" (1974, Paul Werbos)

The derivatives of the error function (two-layers)

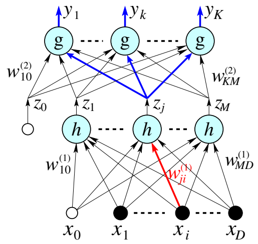
$$E_n = \frac{1}{2}\sum_{k=1}^{K}(y_{nk}-t_{nk})^2$$

$$y_{nk} = g(a_{nk}), \quad a_{nk} = \sum_{j=1}^{M} w_{kj}^{(2)} z_{nj}$$

$$z_{nj} = h(b_{nj}), \quad b_{nj} = \sum_{i=0}^{D} w_{ji}^{(1)} x_{ni}$$

$$\frac{\partial E_n}{\partial w_{kj}^{(2)}} = \frac{\partial E_n}{\partial y_{nk}} \frac{\partial y_{nk}}{\partial a_{nk}} \frac{\partial a_{nk}}{\partial w_{kj}^{(2)}} = (y_{nk}-t_{nk})\, g'(a_{nk})\, z_{nj}$$

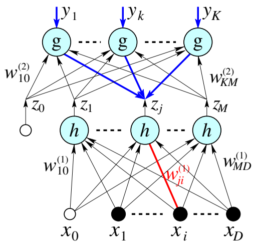
$$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \frac{\partial E_n}{\partial z_{nj}} \frac{\partial z_{nj}}{\partial b_{nj}} \frac{\partial b_{nj}}{\partial w_{ji}^{(1)}} = \left(\sum_{k=1}^{K}(y_{nk}-t_{nk})\frac{\partial y_{nk}}{\partial z_{nj}}\right) h'(b_{nj})\, x_{ni} = \left(\sum_{k=1}^{K}(y_{nk}-t_{nk}) g'(a_{nk}) w_{kj}^{(2)}\right) h'(b_{nj})\, x_{ni}$$



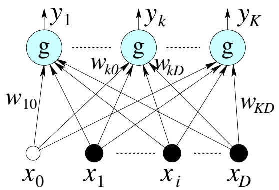
Error back propagation

$$\frac{\partial E_n}{\partial w_{kj}^{(2)}} = \frac{\partial E_n}{\partial y_{nk}} \frac{\partial y_{nk}}{\partial a_{nk}} \frac{\partial a_{nk}}{\partial w_{kj}^{(2)}} = (y_{nk}-t_{nk})\, g'(a_{nk})\, z_{nj} = \delta_{nk}^{(2)} z_{nj}, \quad \delta_{nk}^{(2)} = \frac{\partial E_n}{\partial a_{nk}}$$

$$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \frac{\partial E_n}{\partial z_{nj}} \frac{\partial z_{nj}}{\partial b_{nj}} \frac{\partial b_{nj}}{\partial w_{ji}^{(1)}} = \left(\sum_{k=1}^{K}(y_{nk}-t_{nk}) g'(a_{nk}) w_{kj}^{(2)}\right) h'(b_{nj})\, x_{ni} = \left(\sum_{k=1}^{K}\delta_{nk}^{(2)} w_{kj}^{(2)}\right) h'(b_{nj})\, x_{ni}$$



Notes on Activation functions

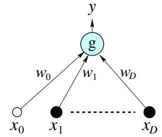


- Interpretation of output values
- Normalisation of the output values
- Other activation functions

Output of logistic sigmoid activation function

- Consider a single-layer network with a single output node logistic sigmoid activation function:

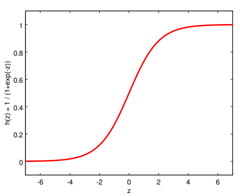
$$y = g(a) = \frac{1}{1+\exp(-a)} = g\left(\sum_{i=0}^{D} w_i x_i\right) = \frac{1}{1+\exp\left(-\sum_{i=0}^{D} w_i x_i\right)}$$



- Consider a two class problem, with classes C_1 and C_2. The posterior probability of C_1:

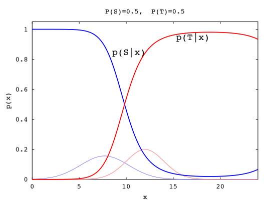
$$P(C_1|\mathbf{x}) = \frac{p(\mathbf{x}|C_1)\,P(C_1)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|C_1)\,P(C_1)}{p(\mathbf{x}|C_1)\,P(C_1) + p(\mathbf{x}|C_2)\,P(C_2)} = \frac{1}{1+\frac{p(\mathbf{x}|C_2)\,P(C_2)}{p(\mathbf{x}|C_1)\,P(C_1)}} = \frac{1}{1+\exp\left(-\ln\frac{p(\mathbf{x}|C_1)\,P(C_1)}{p(\mathbf{x}|C_2)\,P(C_2)}\right)}$$

Approximation of posterior probabilities



Logistic sigmoid function

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



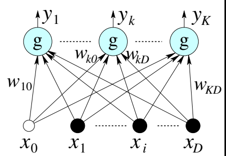
Posterior probabilities of two classes with Gaussian distributions:

Normalisation of output nodes

- Outputs with sigmoid activation funtion:

$$\sum_{k=1}^{K} y_k \neq 1$$

$$y_k = g(a_k) = \frac{1}{1+\exp(-a_k)}, \quad a_k = \sum_{i=0}^{D} w_{ki} x_i$$



- Softmax activation function for $g()$:

$$y_k = \frac{\exp(a_k)}{\sum_{\ell=1}^{K}\exp(a_\ell)}$$

- Properties of the softmax function
  - (i) $0 \leq y_k \leq 1$
  - (ii) $\sum_{k=1}^{K} y_k = 1$
  - (iii) differentiable
  - (iv) $y_k \approx P(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)P(C_k)}{\sum_{\ell=1}^{K} p(\mathbf{x}|C_\ell)P(C_\ell)}$

Some questions on activation functions

- Is the logistic sigmoid function necessary for single-layer single-output-node network?
  - No, in terms of classification. (we can replace it with $g(a) = a$)
- What benefits are there in using the logistic sigmoid function?

Summary

- Training of single-layer network
- Training of multi-layer network with 'error back propagation'
- Activation functions
  - Approximation of posterior probabilities
    - Sigmoid function (for single output node)
    - Softmax function (for multiple output nodes)
- A very good reference: http://neuralnetworksanddeeplearning.com/

Inf2b - Learning

Lecture 15: Multi-layer neural networks (2)

Hiroshi Shimodaira
(Credit: Iain Murray and Steve Renals)

Centre for Speech Technology Research (CSTR)
School of Informatics
University of Edinburgh

http://www.inf.ed.ac.uk/teaching/courses/inf2b/
https://piazza.com/ed.ac.uk/spring2020/infr08028
Office hours: Wednesdays at 14:00-15:00 in IF-3.04

Jan-Mar 2020

Today's Schedule

1. Training of neural networks (recap)
2. Activation functions
3. Experimental comparison of different classifiers
4. Overfitting and generalisation
5. Deep Neural Networks

Training of neural networks (recap)

- Optimisation problem (training):

$$\min_{\boldsymbol{w}} E(\boldsymbol{w}) = \min_{\boldsymbol{w}} \frac{1}{2}\sum_{n=1}^{N}(y_n - t_n)^2$$

- No analytic solution (no closed form)
- Employ an iterative method (requires initial values) e.g. Gradient descent (steepest descent), Newton's method, Conjugate gradient methods
- Gradient descent

$$w_i^{(\mathrm{new})} \leftarrow w_i - \eta \frac{\partial}{\partial w_i} E(\boldsymbol{w}), \qquad (\eta > 0)$$

Training of the single-layer neural network (recap)

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}(y_n - t_n)^2 = \frac{1}{2}\sum_{n=1}^{N}(g(a_n) - t_n)^2$$

$$\text{where} \quad y_n = g(a_n), \quad a_n = \sum_{i=0}^{D} w_i x_{ni}, \quad \frac{\partial a_n}{\partial w_i} = x_{ni}$$

$$\frac{\partial E(\boldsymbol{w})}{\partial w_i} = \frac{\partial E(\boldsymbol{w})}{\partial y_n}\frac{\partial y_n}{\partial a_n}\frac{\partial a_n}{\partial w_i}$$

$$= \sum_{n=1}^{N}(y_n - t_n)\frac{\partial g(a_n)}{\partial a_n}\frac{\partial a_n}{\partial w_i}$$

$$= \sum_{n=1}^{N}(y_n) - t_n)\, g'(a_n)\, x_{ni}$$

Multi-layer neural networks (recap)

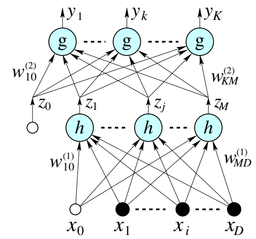
Multi-layer perceptron (MLP)

- Hidden-to-output weights:

$$w_{kj}^{(2)} \leftarrow w_{kj}^{(2)} - \eta \frac{\partial E}{\partial w_{kj}^{(2)}}$$

- Input-to-hidden weights:

$$w_{ji}^{(1)} \leftarrow w_{ji}^{(1)} - \eta \frac{\partial E}{\partial w_{ji}^{(1)}}$$



The derivatives of the error function (two-layers) (recap)

$$E_n = \frac{1}{2}\sum_{k=1}^{K}(y_{nk} - t_{nk})^2$$

$$y_{nk} = g(a_{nk}), \quad a_{nk} = \sum_{j=1}^{M} w_{kj}^{(2)} z_{nj}$$

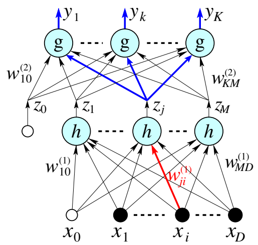
$$z_{nj} = h(b_{nj}), \quad b_{nj} = \sum_{i=0}^{D} w_{ji}^{(1)} x_{ni}$$

$$\frac{\partial E_n}{\partial w_{kj}^{(2)}} = \frac{\partial E_n}{\partial y_{nk}}\frac{\partial y_{nk}}{\partial a_{nk}}\frac{\partial a_{nk}}{\partial w_{kj}^{(2)}}$$

$$= (y_{nk} - t_{nk})\, g'(a_{nk})\, z_{nj}$$

$$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \frac{\partial E_n}{\partial z_{nj}}\frac{\partial z_{nj}}{\partial b_{nj}}\frac{\partial b_{nj}}{\partial w_{ji}^{(1)}} = \left(\sum_{k=1}^{K}(y_{nk} - t_{nk})\frac{\partial y_{nk}}{\partial z_{nj}}\right) h'(b_{nj})\, x_{ni}$$

$$= \left(\sum_{k=1}^{K}(y_{nk} - t_{nk}) g'(a_{nk}) w_{kj}^{(2)}\right) h'(b_{nj})\, x_{ni}$$



Error back propagation (recap)

$$\frac{\partial E_n}{\partial w_{kj}^{(2)}} = \frac{\partial E_n}{\partial y_{nk}}\frac{\partial y_{nk}}{\partial a_{nk}}\frac{\partial a_{nk}}{\partial w_{kj}^{(2)}}$$

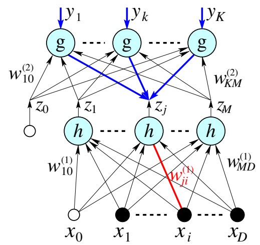
$$= (y_{nk} - t_{nk})\, g'(a_{nk})\, z_{nj}$$

$$= \delta_{nk}^{(2)}\, z_{nj}, \quad \delta_{nk}^{(2)} = \frac{\partial E_n}{\partial a_{nk}}$$

$$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \frac{\partial E_n}{\partial z_{nj}}\frac{\partial z_{nj}}{\partial b_{nj}}\frac{\partial b_{nj}}{\partial w_{ji}^{(1)}}$$

$$= \left(\sum_{k=1}^{K}(y_{nk} - t_{nk}) g'(a_{nk}) w_{kj}^{(2)}\right) h'(b_{nj})\, x_{ni}$$

$$= \left(\sum_{k=1}^{K}\delta_{nk}^{(2)} w_{kj}^{(2)}\right) h'(b_{nj})\, x_{ni}$$



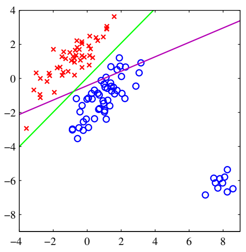
Some questions on activation functions

- Is the logistic sigmoid function necessary for single-layer single-output-node network?
  - No, in terms of classification.
    We can replace it with $g(a) = a$. However, decision boundaries can be different. (NB: A linear decision boundary ($a = 0.5$) is formed in either case.)
- What benefits are there in using the logistic sigmoid function in the case above?
  - The output can be regarded as a posterior probability.
  - Compared with a linear output node ($g(a) = a$), 'logistic regression' normally forms a more robust decision boundary against noise.

Logistic sigmoid vs a linear output node

Binary classification problem with the least squares error (LSE):

$$g(a) = \frac{1}{1+\exp(-a)} \quad \text{vs} \quad g(a) = a$$



(after Fig 4.4b in PRML C. M. Bishop (2006))

Implementations of gradient descent

$$E(w) = \frac{1}{2}\sum_{n=1}^{N} ||\mathbf{y}_n - \mathbf{t}_n||^2 = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}(y_{nk}-t_{nk})^2$$

$$= \sum_{n=1}^{N} E_n, \quad \text{where } E_n = \frac{1}{2}\sum_{k=1}^{K}(y_{nk}-t_{nk})^2$$

- Batch gradient descent:

$$w_{ki} \leftarrow w_{ki} - \eta \frac{\partial E}{\partial w_{ki}}$$

- Incremental (online) gradient descent:
  Update weights for each $\mathbf{x}_n$

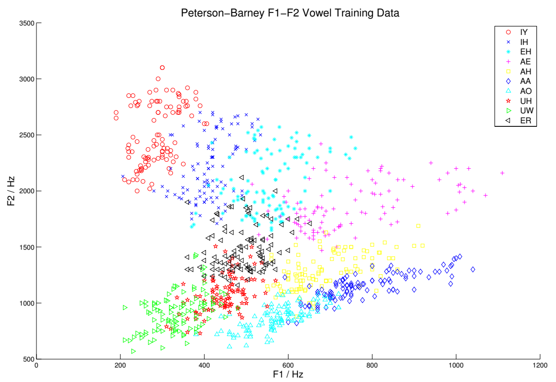
$$w_{ki} \leftarrow w_{ki} - \eta \frac{\partial E_n}{\partial w_{ki}}$$

- Stochastic gradient descent: c.f. Batch/Mini-batch training
  Update weights for randomly chosen $\boldsymbol{x}$.

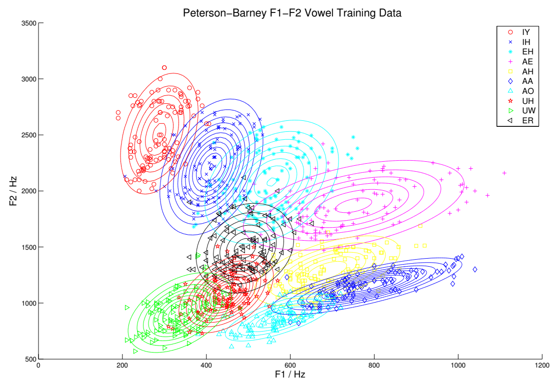
Experimental comparison

- Task: spoken vowel classification
- Classifiers:
  - Gaussian classifier
  - Single layer network (SLN)
  - Multi-layer perceptron (MLP)

Classifying spoken vowels (lecture 09) — Training data



Gaussian for each class



Details of the classifiers

- **Gaussian classifier**: (2-dimensional) Gaussian for each class. Training involves estimating mean vector and covariance matrix for each class, assume equal priors. (50 parameters)
- **Single layer network**: 2 inputs, 10 outputs. Iterative training of weight matrix. (30 parameters)
- **MLP**: two inputs, 25 hidden units, 10 outputs. Trained by gradient descent (backprop). (335 parameters)
- For SLN and MLP normalise feature vectors to mean=0 and sd=1:

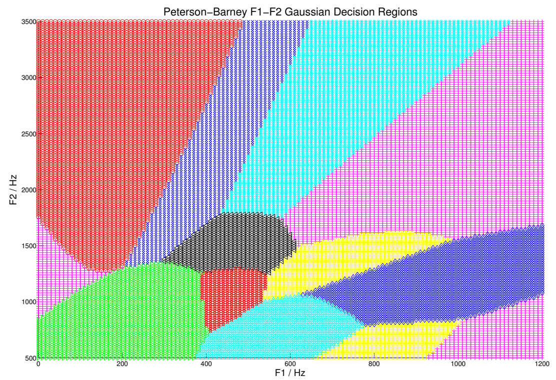
$$z_{ni} = \frac{x_i^n - mi}{s_i}$$

  m_i is sample mean of feature i computed from the training set, s_i is standard deviation.

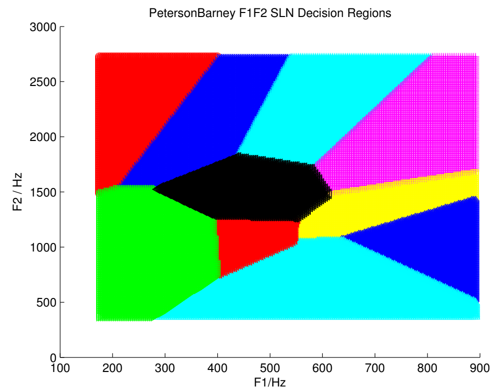
Results

Gaussian classifier:	86.5% correct
Single layer network:	85.5% correct
MLP:	86.5% correct

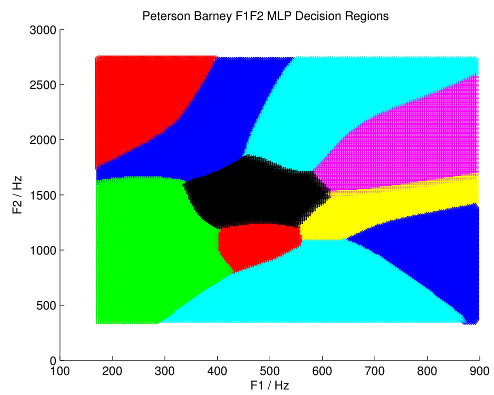
Decision Regions: Gaussian classifier



Decision Regions: Single-layer perceptron



Decision Regions: Multi-layer perceptron



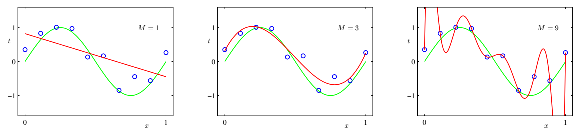
Problems with multi-layer neural networks

- Still difficult to train
  - Computationally very expensive (e.g. weeks of training)
  - Slow convergence ('vanishing gradients')
  - Difficult to find the optimal network topology
- Poor generalisation (under some conditions)
  - Very good performance on the training set
  - Poor performance on the test set

Overfitting and generalisation

Example of curve fitting by a polynomial function:

$$y(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{k=0}^{M} w_k x^k$$



(after Fig 1.4 in PRML C. M. Bishop (2006))

- cf. memorising the training data

Generalisation in neural networks

- How many hidden units (or, how many weights) do we need?
- Optimising training set performance does not necessarily optimise test set performance
  - Network too "flexible": Too many weights compared with the number of training examples
  - Network not flexible enough: Not enough weights (hidden units) to represent the desired mapping
- **Generalisation Error**: The predicted error on unseen data. How can the generalisation error be estimated?
  - Training error?
  $$E_{\text{train}} = \frac{1}{2} \sum_{\text{trainingset}} \sum_{k=1}^{K} (y_k - t_k)^2$$
  - Cross-validation error?
  $$E_{\text{xval}} = \frac{1}{2} \sum_{\text{validationset}} \sum_{k=1}^{K} (y_k - t_k)^2$$

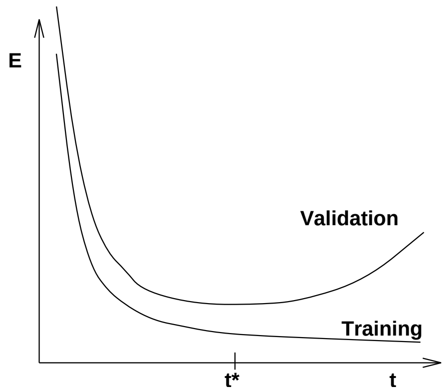
Overtraining in neural networks (†)

- Overtraining (overfitting) corresponds to a network function too closely fit to the training set (too much flexibility)
- Undertraining corresponds to a network function not well fit to the training set (too little flexibility)
- Solutions
  - If possible increasing both network complexity in line with the training set size
  - Use prior information to constrain the network function Control the flexibility: **Structural Stabilisation**
  - Control the effective flexibility: **early stopping** and **regularisation**

Early stopping (†)

- Use validation set to decide when to stop training
- Training-set error monotonically decreases as training progresses
- Validation-set error will reach a minimum then start to increase
- "Effective Flexibility" increases as training progresses
- Network has an increasing number of "effective degrees of freedom" as training progresses
- Network weights become more tuned to training data
- Very effective — used in many practical applications such as speech recognition and optical character recognition

Early stopping



Regularisation — Penalising complexity (†)

- Original error function
$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N} ||\mathbf{y}_n - \mathbf{t}_n||^2$$
- Regularised error function
$$\tilde{E}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N} ||\mathbf{y}_n - \mathbf{t}_n||^2 + \frac{\beta}{2}\sum_{\ell} ||\boldsymbol{w}||^2$$

Ability of neural networks (†)

- Universal approximation thorem
  - "Univariate function and a set of affine functionals can uniformly approximate any continuous function of n real variables with support in the unit hypercube; only mild conditions are imposed on the univariate function. " (G. Cybenko (1989))

    $\longrightarrow$

    A single-output node nerural network with a single hidden layer with a finite neurons can approximate continuous functions.
  - K. Hornik (1990) doi:10.1016/0893-6080(91)90009-T
  - N. Guliyev, V. Ismailov (2018) 10.31219/osf.io/xgnw8

Problems with multi-layer neural networks

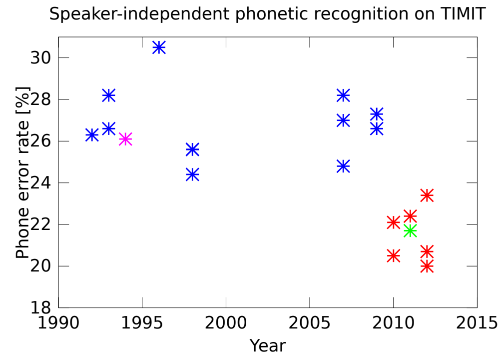
- Still difficult to train
  - Computationally very expensive (e.g. weeks of training)
  - Slow convergence ('vanishing gradients')
  - Difficult to find the optimal network topology
- Poor generalisation (under some conditions)
  - Very good performance on the training set
  - Poor performance on the test set

Breakthrough (†)

1957 Frank Rosenblatt : 'Perceptron'
1986 D. Rumelhart, G. Hinton, and R. Williams: 'Backpropagation'
2006 G. Hinton etal (U. Toronto)
"Reducing the dimensionality of data with neural networks", Science.
2009 J. Schmidhuber (Swiss AI Lab IDSIA)
Winner at ICDAR2009 handwriting recognition competition
2011- many papers from U.Toronto, Microsoft, IBM, Google, ...

- What's the ideas?
  - Pretraining
    - A single layer of feature detectors $\rightarrow$ Stack it to form several hidden layers
  - Fine-tuning, dropout
  - GPU
  - Convolutional network (CNN), Long short-term memory (LSTM)
  - Rectified linear unit (ReLU)

Breakthrough (†)



Summary

- Error back propagation training
- Logistic sigmoid vs linear node
- Decision boundaries
- Overfitting vs generalisation
- (Feed-forward network vs RNN)
- A very good reference:
  http://neuralnetworksanddeeplearning.com/

Inf2b - Learning

Lecture 16: Review

Hiroshi Shimodaira
(Credit: Iain Murray and Steve Renals)

Centre for Speech Technology Research (CSTR)
School of Informatics
University of Edinburgh
http://www.inf.ed.ac.uk/teaching/courses/inf2b/
https://piazza.com/ed.ac.uk/spring2020/infr08028
Office hours: Wednesdays at 14:00-15:00 in IF-3.04

Jan-Mar 2020

Today's Schedule

1. Topic revision
2. Maths formulae to remember
3. Methods/derivations to understand
4. Exam technique

Topics dealt within the course

- Distance and similarity measures (Pearson correlation coef.)
- Clustering (K-means clustering)
- Dimensionality reduction (covariance matrix, PCA)
- Classification
  - K-NN classification
  - Naive Bayes
  - Gaussian classifiers (MLE, discriminant functions)
  - Neural networks (Perceptron error correction algorithm, sum-of-squares error cost function, gradient descent, EBP)
- Statistical pattern recognition theories
  - Bayes theorem, and Bayes decision rule
  - Probability distributions and parameter estimation
    - Bernoulli distribution / Multinomial distribution
    - Gaussian distribution
  - Discriminant functions
  - Decision boundaries/regions (minimum error rate classification)
  - Evaluation measures and methods
- Optimisation problems

Maths formulae to remember

- Euclidean distance:

$$r_2(\boldsymbol{x}, \boldsymbol{y}) = ||\boldsymbol{x} - \boldsymbol{y}|| = \sqrt{\sum_{i=1}^{D}(x_i - y_i)^2}$$

cf. $\mathrm{sim}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{1+r_2(\boldsymbol{x},\boldsymbol{y})}$ as a similarity measure

- Pearson correlation coefficient:

$$\rho(x, y) = \frac{1}{N-1}\sum_{n=1}^{N}\frac{(x_n - \mu_x)}{\sigma_x}\frac{(y_n - \mu_y)}{\sigma_y}$$

- Bayes Theorem

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

$$P(C_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|C_k)P(C_k)}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{x}|C_k)P(C_k)}{\sum_{k=1}^{K} p(\boldsymbol{x}|C_k)P(C_k)}$$

Maths formulae to remember (cont.)

- Bayes decision rule (cf. MAP decision rule)

$$k^* = \arg\max_k P(C_k \,|\, \boldsymbol{x}) = \arg\max_k P(\boldsymbol{x}|C_k)P(C_k)$$

- Naive Bayes for document classification
  (vocabulary: $V = \{w_1, \ldots, w_{|V|}\}$, test document: $D = (o_1, \ldots, o_L)$)
  - Likelihood by Bernoulli document model

$$P(\boldsymbol{b}|C_k) = \prod_{t=1}^{|V|}[b_t P(w_t \,|\, C_k) + (1-b_t)(1-P(w_t \,|\, C_k))]$$

  - Likelihood by Multinomial document model

$$p(\boldsymbol{x}|C_k) \propto \prod_{t=1}^{|V|} P(w_t|C_k)^{x_t} = \prod_{i=1}^{L} P(o_i|C_k)$$

Maths formulae to remember (cont.)

- Univariate Gaussian pdf:
$$p(x\,|\,\mu,\sigma^2) = N(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$$
- Multivariate Gaussian pdf:
$$p(\mathbf{x}\,|\,\boldsymbol{\mu},\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$
Parameter estimation from samples:
$$\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n\,,\qquad \hat{\Sigma} = \frac{1}{N-1}\sum_{n=1}^{N}(\mathbf{x}_n-\hat{\mu})(\mathbf{x}_n-\hat{\mu})^T$$
NB: N in case of MLE
- Correlation coefficient:
$$\rho(x_i,x_j) = \rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}}\,,\qquad \Sigma = (\sigma_{ij})$$

Maths formulae to remember (cont.)

- Logistic sigmoid function:
$$y = g(a) = \frac{1}{1+\exp(-a)}$$
$$g'(a) = g(a)(1-g(a))$$
- Softmax activation function (for multiple output nodes):
$$y_k = \frac{\exp(a_k)}{\sum_{\ell=1}^{K}\exp(a_\ell)}$$
- and basic maths rules (e.g. differentiation)

Methods/derivations to understand (non exhaustive)

- Clustering and classification
- Discriminant functions of Gaussian Bayes classifiers
- Learning as an optimisation problem
  - Maximum likelihood estimation
  - Gradient descent and back propagation algorithm (neural networks) for minimising the sum-of-squares error

NB: Learning is a difficult problem by nature — generalisation from a limited amount of training samples.
→ need to assume some structures (constraints):
  - Probability distributions
  - Naive Bayes
  - Diagonal covariance matrix rather than a full covariance for each class, shared covariance matrix among classes, regularisation.
  - Dimensionality reduction and feature selection *(NE)*

Machine learning as optimisation problems

- Euclidean-distance based classification
$k^* = \arg\min_k \|\boldsymbol{x} - \mathbf{r}_k\|$
- K-means clustering
$\min_{\{z_{kn}\}} \sum_{k=1}^{K}\sum_{n=1}^{N} z_{kn}\,\|\mathbf{x}_n - \mathbf{m}_k\|^2$
- Dimensionality reduction to 2D with PCA
$\max_{\mathbf{u},\mathbf{v}} \mathrm{Var}(y) + \mathrm{Var}(z)$
subject to $\|\mathbf{u}\|=1, \|\mathbf{v}\|=1, \mathbf{u}\perp\mathbf{v}$
- Bayes decision rule
$k^* = \arg\max_k P(C_k\,|\,\mathbf{x}) = \arg\max_k P(\mathbf{x}|C_k)P(C_k)$
- Maximum likelihood parameter estimation
$\max_{\boldsymbol{\mu},\boldsymbol{\Sigma}} L(\boldsymbol{\mu},\boldsymbol{\Sigma}|\mathcal{D})$
- Least squares error training of neural networks
$\min_{\boldsymbol{w}} \frac{1}{2}\sum_{n=1}^{N}\|\mathbf{y}_n - \mathbf{t}_n\|^2$

Exam revision

Look at lecture notes, slides, tutorials, coursework, and past papers.

Early exam papers: many (useful) multiple choice Qs
- No longer the exam format
- Syllabus has changed slightly

Recent exam papers since 2008/09
- Answer two questions from section A (ADS) and two questions from section B (Learning).
- Closed-book exam.
- Calculators may be used (approved ones only).
- Solutions are available only for 2008/09, 2009/10, 2013/14 (no plans of releasing those of missing years)
- NB: errors in some solutions, e.g. 5 (c) of 2008/09: square root is not taken in computing standard deviations.

Well prepared for the exam of 120 minutes
60 minutes/section, 30 minutes/question

Exam revision (cont.)

Don't overfit!
Anything that appears in the notes, slides, tutorial sheets, or coursework is examinable, unless marked non-examinable, extra topics, or (†)

Don't trust unofficial solutions

Inf2b Revision Meeting
- Date: TBC (in late April)
- Send me questions/requests that you want me to discuss at the meeting.

Time in the exam

- Half an hour per question (minus time to pick questions)
- Don't panic!
- Go for easy marks first
- Don't spend a long time on any small part
- Don't scrawl - you might lose marks if the marker cannot read/understand
- Know the standard stuff:
there's not time to work everything out from scratch

Calculators may be used in the examination: The School of Informatics does not provide calculators for use in exams. If the use of a calculator is permitted in an exam, it's your responsibility to bring an approved calculator to the exam.

End-of-course feedback:

Thanks!