582631 – 4 credits

Introduction to Machine Learning

Lecturer: Jyrki Kivinen
Assistant: Yuan Zou

Department of Computer Science
University of Helsinki

based on material created by Patrik Hoyer and others

29 October–6 December 2013

Introduction

- What is machine learning? Motivation & examples
  - Definition
  - Relation to other fields
  - Examples
- Course outline and related courses
- Practical details of the course
  - Lectures
  - Exercises
  - Exam
  - Grading

What is machine learning?

- Definition:

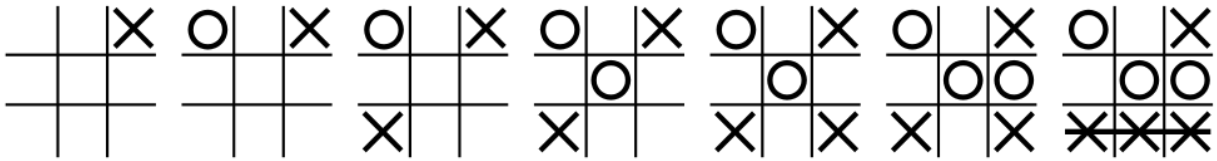
  machine = computer, computer program (in this course)

  learning = improving performance on a given task, based on experience / examples

- In other words
  - instead of the programmer writing explicit rules for how to solve a given problem, the programmer instructs the computer how to learn from examples
  - in many cases the computer program can even become better at the task than the programmer is!

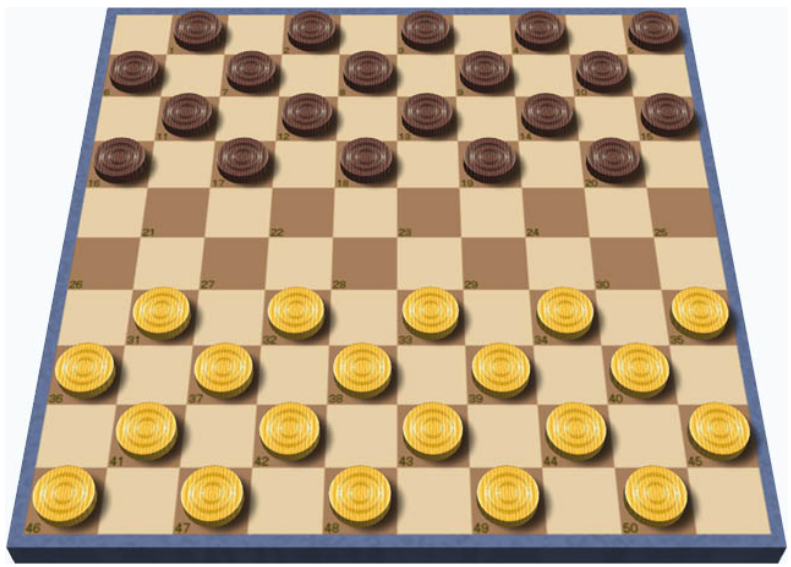
Example 1:

- How to program the computer to play tic-tac-toe?



- Option A: The programmer writes explicit rules, e.g. 'if the opponent has two in a row, and the third is free, stop it by placing your mark there', etc (lots of work, difficult, not at all scalable!)
- Option B: Go through the game tree, choose optimally (for non-trivial games, must be combined with some heuristics to restrict tree size)
- Option C: Let the computer try out various strategies by playing against itself and others, and noting which strategies lead to winning and which to losing (='machine learning')

- Arthur Samuel (50's and 60's):
  - Computer program that learns to play checkers
  - Program plays against itself thousands of times, learns which positions are good and which are bad (i.e. which lead to winning and which to losing)
  - The computer program eventually becomes much better than the programmer.



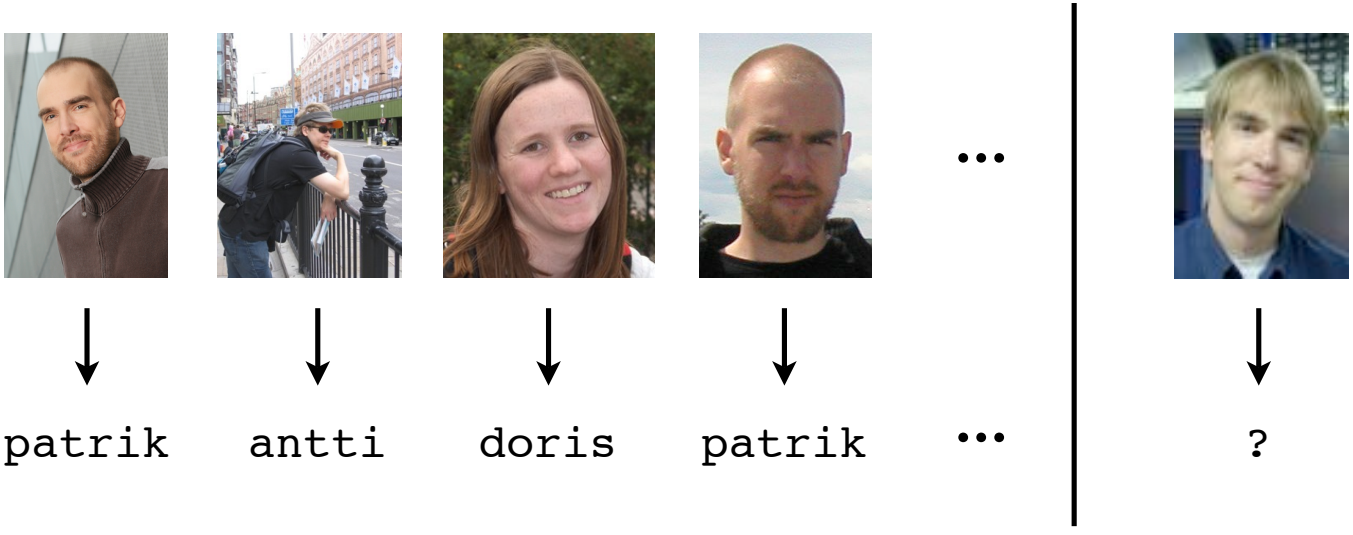
Example 2: spam filter

- ~~Programmer writes rules: "If it contains 'viagra' then it is spam."~~ (difficult, not user-adaptive)
- The user marks which mails are spam, which are legit, and the computer learns itself what words are predictive

From: medshop@spam.com Subject: viagra cheap meds...	spam
From: my.professor@helsinki.fi Subject: important information here's how to ace the test...	non-spam
⋮	⋮
From: mike@example.org Subject: you need to see this how to win $1,000,000...	?

Example 3: face recognition

- Face recognition is hot (facebook, apple; security; . . .)
- ~~Programmer writes rules: "If short dark hair, big nose, then it is Mikko"~~ (impossible! how do we judge the size of the nose?!)
- The computer is shown many `(image, name)` example pairs, and the computer learns which features of the images are predictive (difficult, but not impossible)



Problem setup

- One definition of machine learning: A computer program improves its performance on a given task with experience (i.e. examples, data).
- So we need to separate
  - Task: What is the problem that the program is solving?
  - Performance measure: How is the performance of the program (when solving the given task) evaluated?
  - Experience: What is the data (examples) that the program is using to improve its performance?

Related scientific disciplines (1)

- Artificial Intelligence (AI)
  - Machine learning can be seen as 'one approach' towards implementing 'intelligent' machines (or at least machines that behave in a seemingly intelligent way).
- Artificial neural networks, computational neuroscience
  - Inspired by and trying to mimic the function of biological brains, in order to make computers that learn from experience. Modern machine learning really grew out of the neural networks boom in the 1980's and early 1990's.
- Pattern recognition
  - Recognizing objects and identifying people in controlled or uncontrolled settings, from images, audio, etc. Such tasks typically require machine learning techniques.

Availability of data

- These days it is very easy to
  - collect data (sensors are cheap, much information digital)
  - store data (hard drives are big and cheap)
  - transmit data (essentially free on the internet).
- The result? *Everybody* is collecting large quantities of data.
  - Businesses: shops (market-basket data), search engines (web pages and user queries), financial sector (stocks, bonds, currencies etc), manufacturing (sensors of all kinds), social networking sites (facebook, twitter), anybody with a web server (hits, user activity)
  - Science: genomes sequenced, gene expression data, experiments in high-energy physics, images of remote galaxies, global ecosystem monitoring data, drug research and development, public health data
- But how to benefit from it? Analysis is becoming key!

Related scientific disciplines (2)

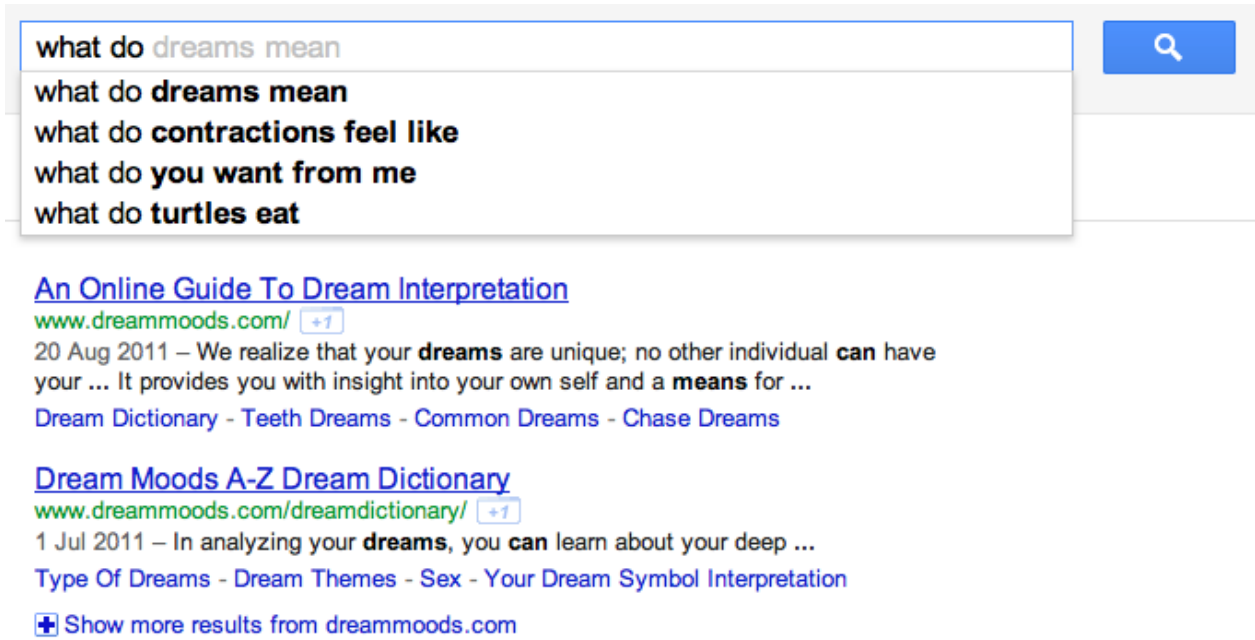
- Data mining
  - Trying to identify interesting and useful associations and patterns in huge datasets
  - Focus on scalable algorithms

  Example: On the order of 3 million people grocery shopping twice a week in just two main chains in Finland $\Rightarrow$ each chain would collect hundreds of thousands of transaction receipts per day!

- Statistics
  - Traditionally: focus on testing hypotheses based on theory
  - Has contributed a lot to data mining and machine learning, and has also evolved by incorporating ideas derived from these fields

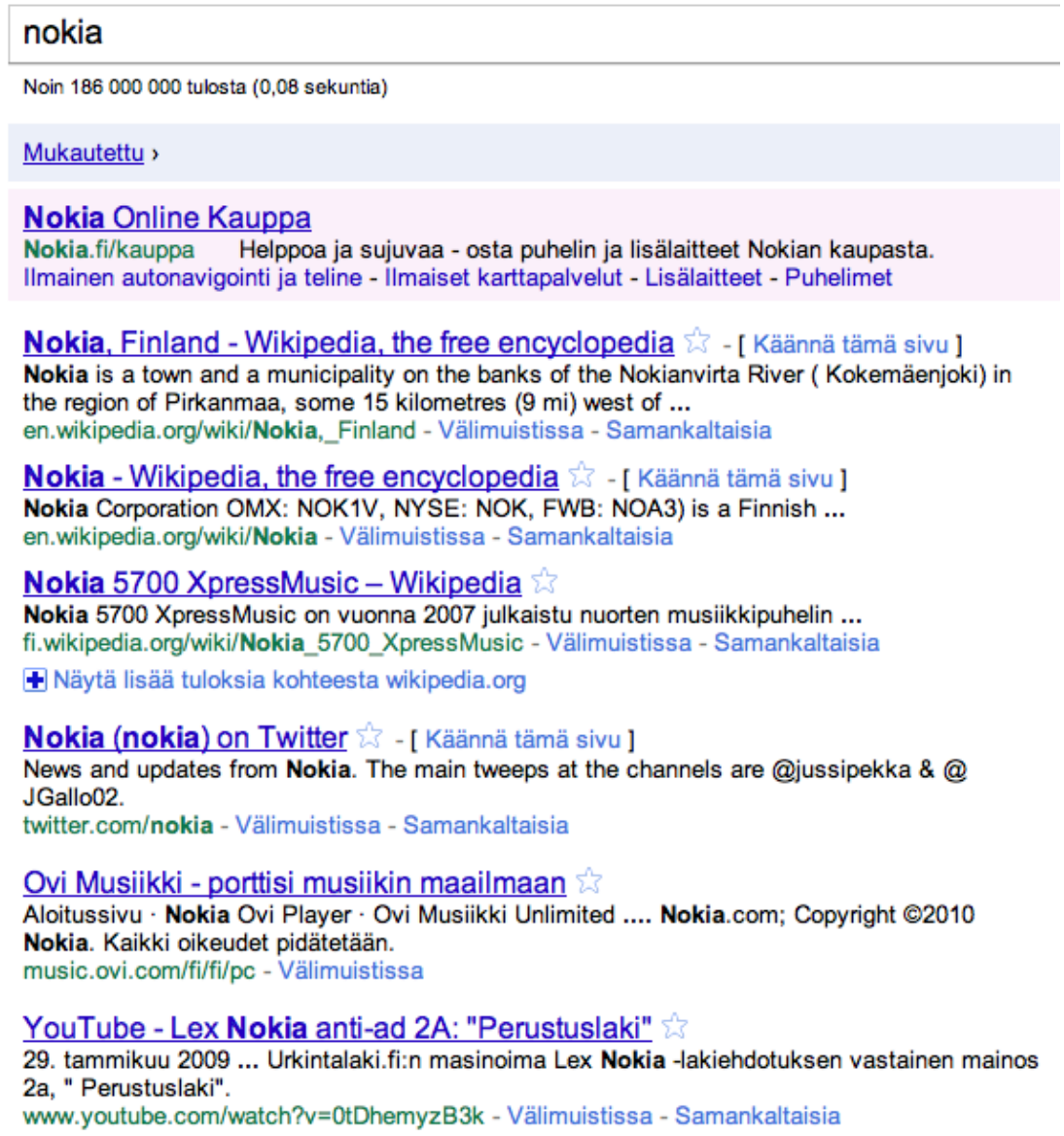
Example 4

- Prediction of search queries
  - ~~The programmer provides a standard dictionary~~ (words and expressions change!)
  - Previous search queries are used as examples!



Example 5

- Ranking search results:
  - Various criteria for ranking results
  - What do users click on after a given search? Search engines can learn what users are looking for by collecting queries and the resulting clicks.



Example 6

- Detecting credit card fraud
  - Credit card companies typically end up paying for fraud (stolen cards, stolen card numbers)
  - Useful to try to detect fraud, for instance large transactions
  - Important to be adaptive to the behaviors of customers, i.e. learn from existing data how users normally behave, and try to detect 'unusual' transactions

Example 7

- Self-driving cars:
  - Sensors (radars, cameras) superior to humans
  - How to make the computer react appropriately to the sensor data?

SMARTER THAN YOU THINK

Google Cars Drive Themselves, in Traffic

Ramin Rahimian for The New York Times

Example 8

- Character recognition:
  - Automatically sorting mail (handwritten characters)
  - Digitizing old books and newspapers into easily searchable format (printed characters)

Example 9

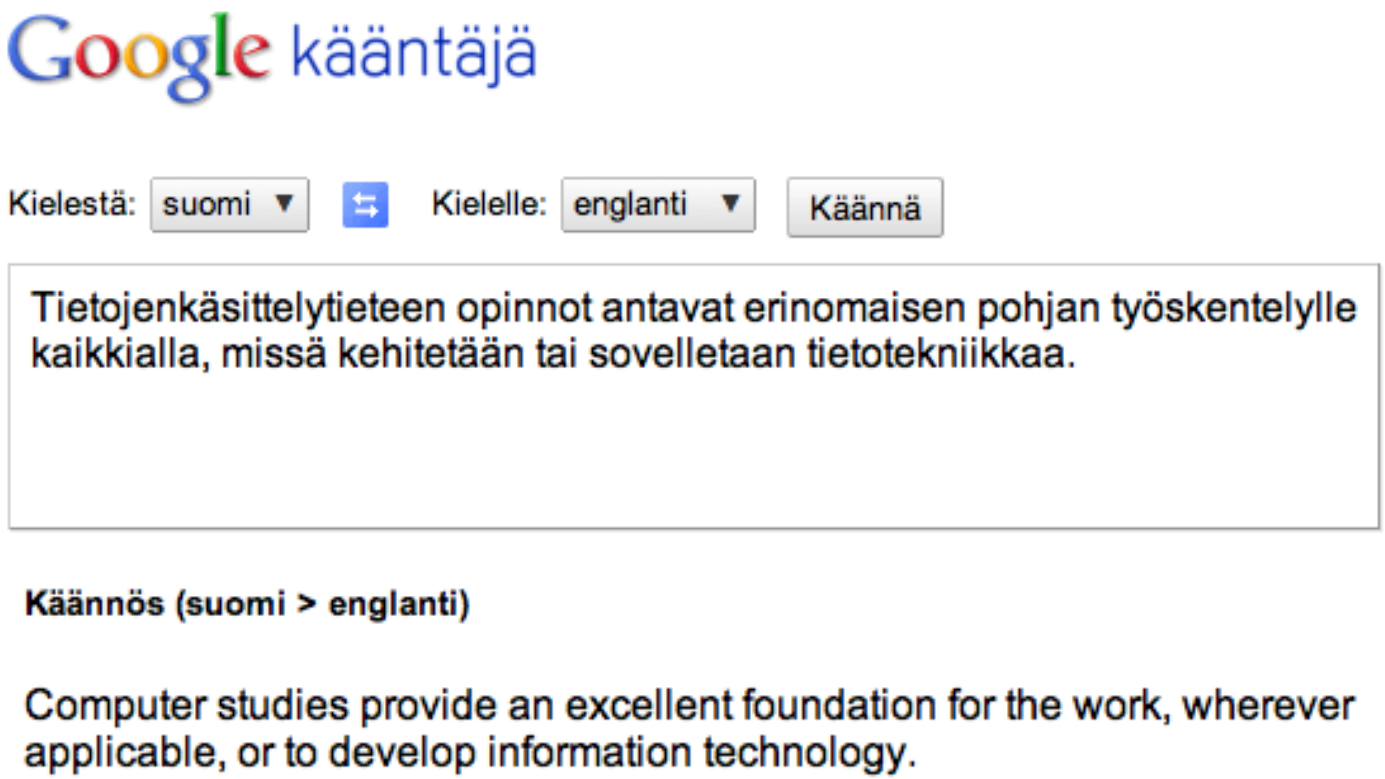
- Recommendation systems ('collaborative filtering'):
  - Amazon: "Customers who bought X also bought Y"...
  - Netflix: "Based on your movie ratings, you might enjoy..."

    Challenge: One million dollars ($1,000,000) prize money recently awarded!

	Seven	Fargo	Aliens	Leon	Avatar
Linda		4		5	5	1		2
			3		4	3		
Jack	1			4		1	5	1
Bill				?	4	1		?
Lucy			2	1	1		5	
John	1				1	4		5
		4				5		5
	2		3				3	

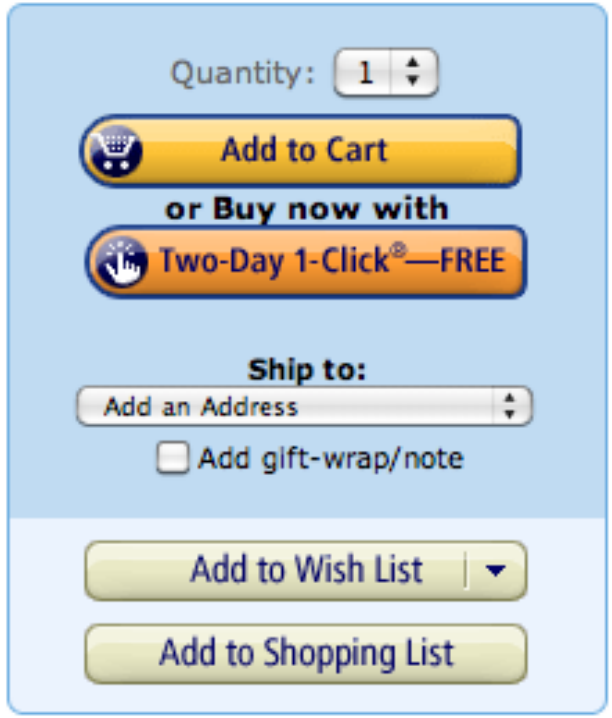
Example 10

- Machine translation:
  - Traditional approach: Dictionary and explicit grammar
  - More recently, *statistical* machine translation based on example data is increasingly being used



Example 11

- Online store website optimization:
  - What items to present, what layout?
  - What colors to use?
  - Can significantly affect sales volume
  - Experiment, and analyze the results! (lots of decisions on how exactly to experiment and how to ensure meaningful results)



Example 12

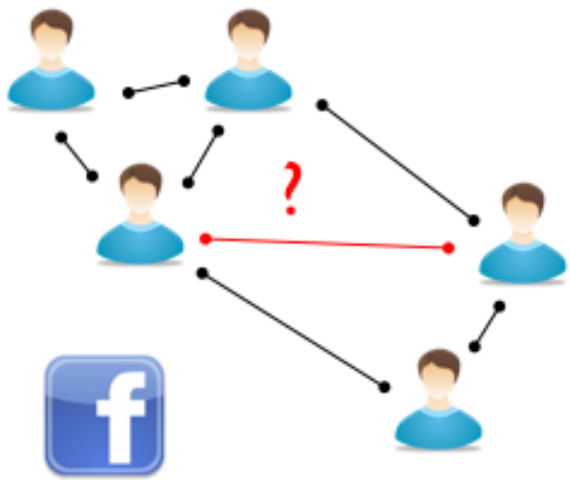
- Mining chat and discussion forums
  - Breaking news
  - Detecting outbreaks of infectious disease
  - Tracking consumer sentiment about companies / products

Example 13

- Real-time sales and inventory management
  - Picking up quickly on new trends (what's *hot* at the moment?)
  - Deciding on what to produce or order (example: Jopo production moved from Taiwan to Finland for a quicker response to incoming sales data – YLE 10.6.2010)

Example 14

- Prediction of friends in Facebook, or prediction of who you'd like to follow on Twitter.



What about privacy?

- Users are surprisingly willing to sacrifice privacy to obtain useful services and benefits
- Regardless of what position you take on this issue, it is important to know what *can* and what *cannot* be done with various types information (i.e. what the dangers are)
- 'Privacy-preserving data mining'
  - What type of statistics/data can be released without exposing sensitive personal information? (e.g. government statistics)
  - Developing data mining algorithms that limit exposure of user data (e.g. 'Collaborative filtering with privacy', Canny 2002)

Course outline

- Introduction
- Data
  - data types and quality, preprocessing
  - similarity/distance measures, visualization
- Supervised learning
  - classification
  - regression
  - evaluation and model selection
- Unsupervised learning
  - clustering
  - anomaly detection

Data

Data: Outline

- Types of data
  - data objects and attributes
  - unordered vs ordered data
- Quality of data
  - measurement errors
  - missing values, inconsistent values, duplicates
- Preprocessing
  - aggregation, sampling, feature extraction
  - discretization and variable transformations
- Similarity and dissimilarity
  - desirable properties
  - common measures
- Summary statistics and visualization

Standard data model

A data set can often be viewed as a *collection of data objects*:

- A *data object* (also called a record, data point, data vector, case, sample point, observation, entity) is described by a set of attributes
- An *attribute* (also called a variable, characteristic, field, feature, or dimension) describes one aspect of a data object

Example:

Student ID	Name	Date of Birth	Credits	Average	...
2328193	Matti	23.09.1985	123	3.6	...
9819234	Tuuli	12.03.1986	98	3.8	...
...	...	...	...	...	...

Here: each row is one data object, each column is an attribute.

Attributes: Number of values

- How many distinct values can an attribute take
  1. Binary attributes: only 2 possible states
     (e.g. on/off, pass/fail)
  2. Discrete attributes with $N < \infty$ states
     (e.g. course grade: 0, 1, 2, 3, 4, or 5)
  3. Discrete attributes with (countably) infinite states
     (e.g. counts: 0,1,2,3, . . .)
  4. Continuous attributes (uncountably infinite)
     (e.g. length or weight: $\in \mathbb{R}$)

- Asymmetric attributes
  1. Binary sparse attributes ('on' infrequent and significant, 'off' common and insignificant)
  2. Other attributes may also have 'special' states, need to take into account

Attributes: Measurement scales

- What operations are valid?
  1. Distinctness: $=$ vs $\neq$
  2. Order: $<$, $\leq$, $>$, $\geq$
  3. Addition and subtraction: $+$, $-$
  4. Multiplication and division: $*$ and $/$

- These define four types of attribute measurement scales:
  1. Nominal (categorical)
  2. Ordinal
  3. Interval
  4. Ratio

  Each measurement scale allows all the operations with number *smaller than or equal to* the number of the measurement scale. Example: 'Interval' attributes allow all operations except multiplication and division.

Attributes: Measurement scales – examples

Attribute Type	Description	Examples	Operations
Nominal	The values of a nominal attribute are just different names, i.e., nominal attributes provide only enough information to distinguish one object from another. (=, ≠)	zip codes, employee ID numbers, eye color, sex: {*male, female*}	mode, entropy, contingency correlation, χ^2 test
Ordinal	The values of an ordinal attribute provide enough information to order objects. (<, >)	hardness of minerals, {*good, better, best*}, grades, street numbers	median, percentiles, rank correlation, run tests, sign tests
Interval	For interval attributes, the differences between values are meaningful, i.e., a unit of measurement exists. (+, -)	calendar dates, temperature in Celsius or Fahrenheit	mean, standard deviation, Pearson's correlation, *t* and *F* tests
Ratio	For ratio variables, both differences and ratios are meaningful. (*, /)	temperature in Kelvin, monetary quantities, counts, age, mass, length, electrical current	geometric mean, harmonic mean, percent variation

General characteristics of data

- Number of data points vs dimensionality (number of attributes)
  1. Most traditional data analysis methods assume (many) more data points than dimensions
  2. Many interesting datasets have extremely high dimensions and few data objects
- Sparsity (relatively few non-zero values)
  1. Efficient storage and computation, for some methods
  2. May be important for modeling
- Resolution (spatial, temporal, . . .)
  1. How small details can the sensors reliably detect
  2. How large datasets can the methods handle

Examples – Record data

- Data ‘matrix’
- Sparse matrix representation?

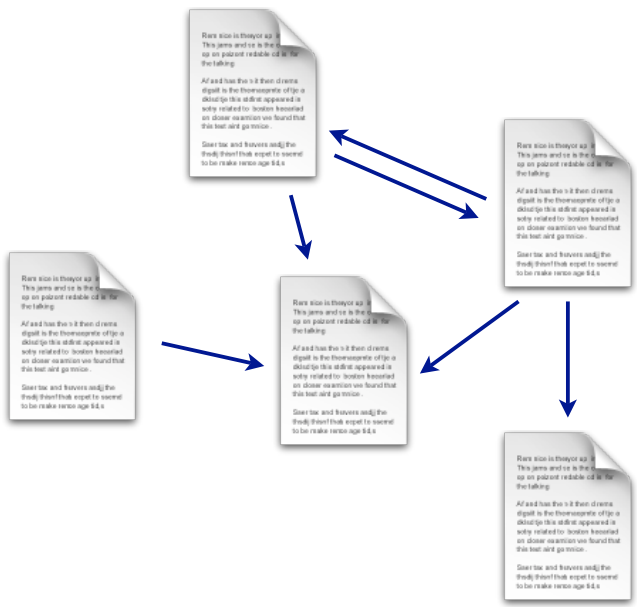
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

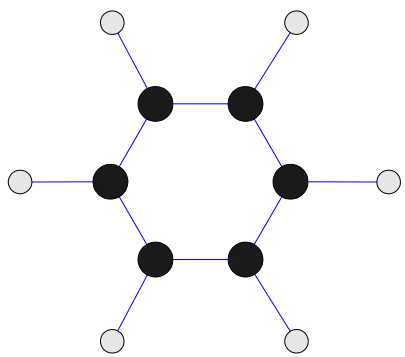
	team	coach	play	ball	score	game	win	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

TID	*Items*
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Examples – Graph data

- Relationships between objects
  - World wide web
  - Facebook users
  - Scientific papers
- Objects are graphs

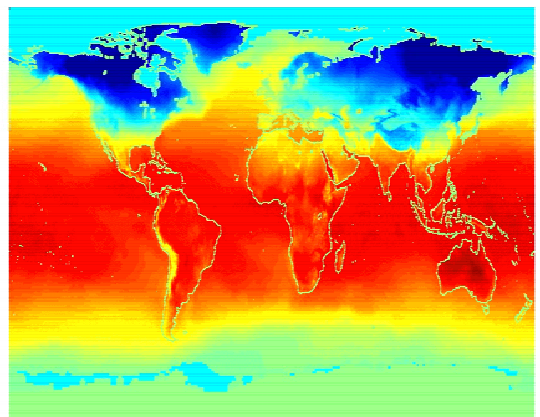




Examples – Ordered data

- Data objects with natural ordering
  - Sequential data (e.g. logins, credit card purchases, . . .)
  - Sequences (e.g. genome)
  - Time-series (e.g. climate science, financial data, . . .)
  - Spatial data (e.g. journey planner data, surface/sea temperatures, multi-spectral images)

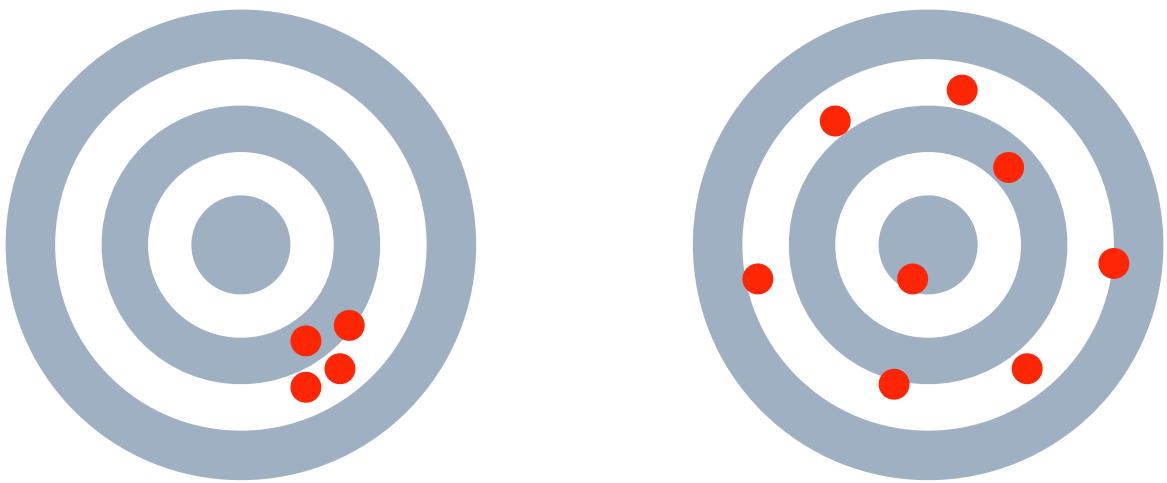
```
GGTTCCGCCTTCAGCCCCGCGCC
CGCAGGGCCCGCCCCGCGCCGTC
GAGAAGGGCCCGCCTGGCGGGCG
GGGGGAGGCGGGGCCGCCCGAGC
CCAACCGAGTCCGACCAGGTGCC
CCCTCTGCTCGGCCTAGACCTGA
GCTCATTAGGCGGCAGCGGACAG
GCCAAGTAGAACACGCGAAGCGC
TGGGCTGCCTGCTGCGACCAGGG
```



Data quality

- Measurement error ('noise')
  - Continuous data: Gaussian, Student's t distribution, ...
  - Binary data: Bernoulli, ...
- Precision and bias
  - Precision: closeness of repeated measurements to each other
  - Bias: systematic deviation from the true underlying value

Left: high precision, high bias. Right: low precision, low bias.



- Outliers i.e. 'anomalous' objects:

  (Objects that are very different from the others)

  - Noise?
  - Data collection error?
  - Legitimate, interesting objects?

- Missing values:

  (Some attribute values are missing for some data objects)

  - Missing at random? Need to model the process?
  - Just eliminating such data objects or attributes?
  - Estimating and imputing missing values?
  - Ignoring or explicitly taking them into account?

- Duplicate data, inconsistent data

- Timeliness, relevance, application-specific prior knowledge

Data preprocessing

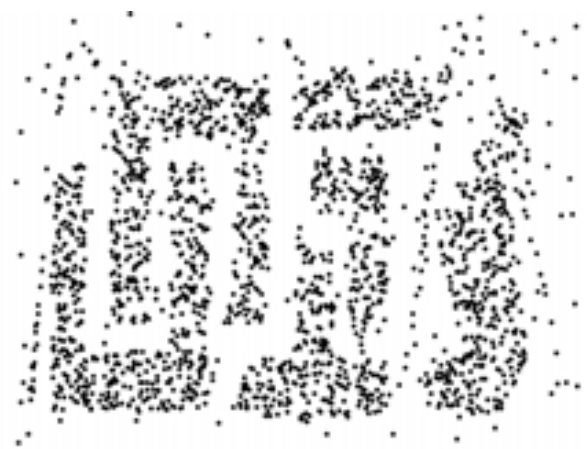
- Aggregation
  - $\Rightarrow$ fewer, less noisy, data points
    - Example: monthly analysis of daily (or hourly) data
    - Loss of resolution

- Sampling (with/without replacement, stratified or not)
  - $\Rightarrow$ fewer data points
    - Visualization, applying computationally demanding data analysis procedures
    - Loss of precision of data statistics

Data preprocessing

- Aggregation
  - $\Rightarrow$ fewer, less noisy, data points
    - Example: monthly analysis of daily (or hourly) data
    - Loss of resolution

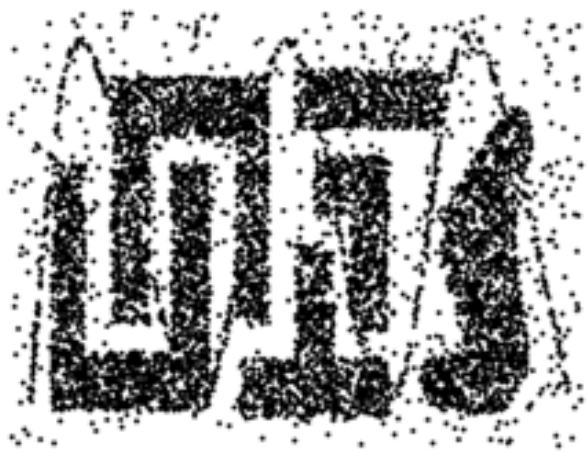
- Sampling (with/without replacement, stratified or not)
  - $\Rightarrow$ fewer data points
    - Visualization, applying computationally demanding data analysis procedures
    - Loss of precision of data statistics



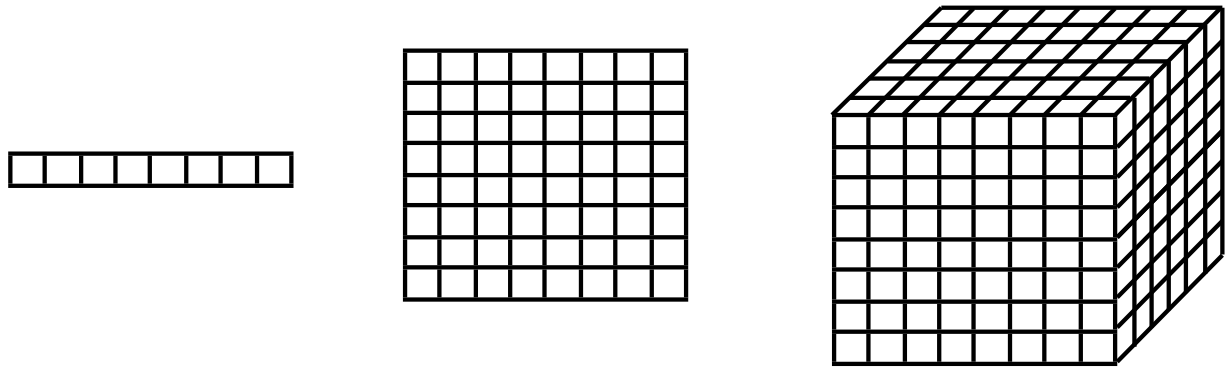
Data preprocessing

- Aggregation
  - $\Rightarrow$ fewer, less noisy, data points
    - Example: monthly analysis of daily (or hourly) data
    - Loss of resolution

- Sampling (with/without replacement, stratified or not)
  - $\Rightarrow$ fewer data points
    - Visualization, applying computationally demanding data analysis procedures
    - Loss of precision of data statistics

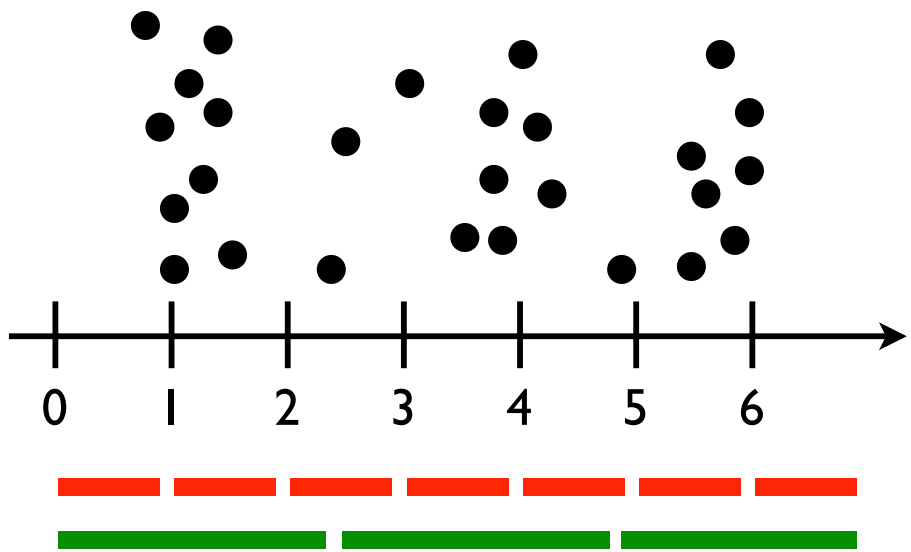


The ‘curse of dimensionality’: As the dimensionality grows, the data becomes increasingly sparse in the space (e.g. n dimensions of binary variables has 2^n joint states)



- Feature subset selection
  (selecting only some of the attributes, manually or automatically)
- Feature extraction
  (computing new ‘attributes’ to replace existing ones, e.g. image color/structure features)
- Dimensionality reduction
  (principal component analysis, other unsupervised methods)

- Discretization of continuous attributes (required for some machine learning algorithms)
  - e.g. $\mathbb{R}$ is divided into $(-\infty, x_1], (x_1, x_2], (x_2, x_3], (x_3, \infty)$
  - Unsupervised or supervised
  - Ideally: use application-specific knowledge



- Binarization
  - Some algorithms *require* binary attributes
  - How NOT to represent nominal variables:

Categorical value	Integer value	x_1	x_2	x_3
Toyota	0	0	0	0
Volkswagen	1	0	0	1
Chevrolet	2	0	1	0
Saab	3	0	1	1
Volvo	4	1	0	0

- Binarization
  - Some algorithms *require* binary attributes
  - How NOT to represent nominal variables:

Categorical value	Integer value	x_1	x_2	x_3
Toyota	0	0	0	0
Volkswagen	1	0	0	1
Chevrolet	2	0	1	0
Saab	3	0	1	1
Volvo	4	1	0	0

  - How better to represent nominal variables:

Categorical value	Integer value	x_1	x_2	x_3	x_4	x_5
Toyota	0	1	0	0	0	0
Volkswagen	1	0	1	0	0	0
Chevrolet	2	0	0	1	0	0
Saab	3	0	0	0	1	0
Volvo	4	0	0	0	0	1

  - Useful even when binary variables are not strictly required!

- Variable transformations
  - Simple functions (e.g. $\log(x)$ often used for strictly positive variables)
  - Ideally: use application-specific knowledge
  - Normalization / standardization:

$$x' = \frac{x - \bar{x}}{s_x}, \tag{1}$$

where $\bar{x}$ is the mean of the attribute values and s_x is the standard deviation. This yields x' with zero mean and a standard deviation of 1.

Useful to ensure that the scale (units) of attribute values do not affect the results.

Similarity and dissimilarity

- Many machine learning algorithms use measures of similarity or dissimilarity between data objects
- Examples:
  - Handwritten letters
  - Segments of DNA
  - Text documents

TCGATTGC ATCCTGTG ACCTGTCG

"Parliament overwhelmingly approved amendments to the Firearms Act on Wednesday. The new law requires physicians to inform authorities of individuals they consider unfit to own guns. It also increases the age for handgun ownership from 18 to 20."

"Parliament's Committee for Constitutional Law says that persons applying for handgun licences should not be required to join a gun club in the future. Government is proposing that handgun users be part of a gun association for at least two years."

"The cabinet on Wednesday will be looking at a controversial package of proposed changes in the curriculum in the nation's comprehensive schools. The most divisive issue is expected to be the question of expanded language teaching."

- Similarity: s
  - Numerical measure of the degree to which two objects are *alike*
  - Higher for objects that are alike
  - Typically between 0 (no similarity) and 1 (completely similar)
- Dissimilarity: d
  - Numerical measure of the degree to which two objects are *different*
  - Higher for objects that are different
  - Typically between 0 (no difference) and ∞ (completely different)
- Transformations
  - Converting from one to the other
- Use similarity or dissimilarity measures?
  - Method-specific

- Proximity between *attribute values*

Attribute Type	Dissimilarity	Similarity
Nominal	$d = \begin{cases} 0 & \text{if } p = q \\ 1 & \text{if } p \neq q \end{cases}$	$s = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}$
Ordinal	$d = \frac{\lvert p-q \rvert}{n-1}$ (values mapped to integers 0 to $n-1$, where n is the number of values)	$s = 1 - \frac{\lvert p-q \rvert}{n-1}$
Interval or Ratio	$d = \lvert p - q \rvert$	$s = -d$, $s = \frac{1}{1+d}$ or $s = 1 - \frac{d - min\_d}{max\_d - min\_d}$

Dissimilarities between *objects*:

- Minkowski distance for vectors in R^n

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{k=1}^{n} |x_k - y_k|^r \right)^{1/r} \tag{2}$$

  - $r = 1$: City-block, hamming
  - $r = 2$: Euclidean
  - $r = \infty$: Supremum

- Distance metric $d(\mathbf{x}, \mathbf{y})$ properties:
  - Positivity: $d(\mathbf{x}, \mathbf{y}) \geq 0$ for all $\mathbf{x}$ and $\mathbf{y}$, with $d(\mathbf{x}, \mathbf{y}) = 0$ if and only if $\mathbf{x} = \mathbf{y}$
  - Symmetry: $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$ for all $\mathbf{x}$ and $\mathbf{y}$
  - Triangle inequality: $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$ for all $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$

- Most dissimilarity measures satisfy positivity. Depending on the application, it may or may not be important to satisfy symmetry and the triangle inequality.

Similarities between objects:

- Typical properties of $s(\mathbf{x}, \mathbf{y})$:
  - $0 \leq s(\mathbf{x}, \mathbf{y}) \leq 1$, with $s(\mathbf{x}, \mathbf{y}) = 1$ if only if $\mathbf{x} = \mathbf{y}$
  - $s(\mathbf{x}, \mathbf{y}) = s(\mathbf{y}, \mathbf{x})$ for all $\mathbf{x}$ and $\mathbf{y}$ (symmetry)

- Examples (binary vectors):
  (f_{ab} is the number of attributes for which $\mathbf{x} = a$ and $\mathbf{y} = b$)
  - Simple matching coefficient

$$\text{SMC} = \frac{f_{11} + f_{00}}{f_{11} + f_{00} + f_{01} + f_{10}} \tag{3}$$

  - Jaccard coefficient

$$J = \frac{f_{11}}{f_{11} + f_{01} + f_{10}} \tag{4}$$

- Examples (arbitrary vectors):
  - Cosine similarity (angle between vectors)

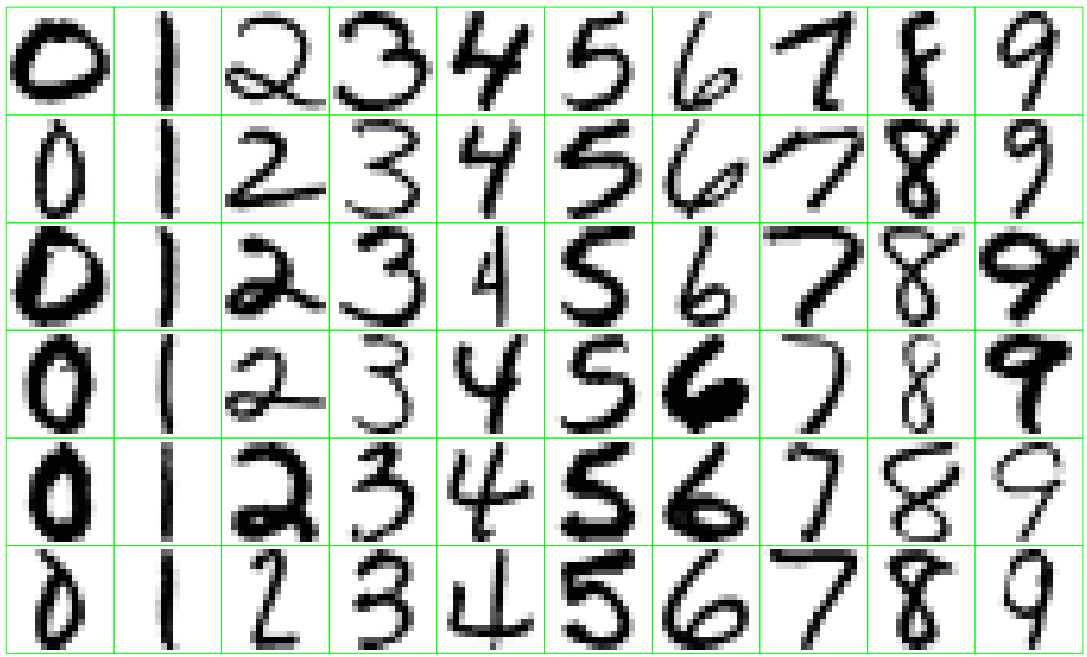
$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^T \mathbf{y}}{||\mathbf{x}|| \, ||\mathbf{y}||} \tag{5}$$

  - (Pearson's) linear correlation coefficient (goes from -1 to 1)
  - (Spearman's) rank correlation coefficient (also -1 to 1)

- Issues to consider:
  - Proximity measures ideally application-specific
  - Use well-known, established measures for the particular type of objects
  - Standardization to avoid arbitrary units affecting results
  - Try a number of different possibilities and evaluate the results

Course dataset 1

- Handwritten digits:
  http://yann.lecun.com/exdb/mnist/index.html



- each image 28×28 pixels, each pixel a gray value between 0 and 255.

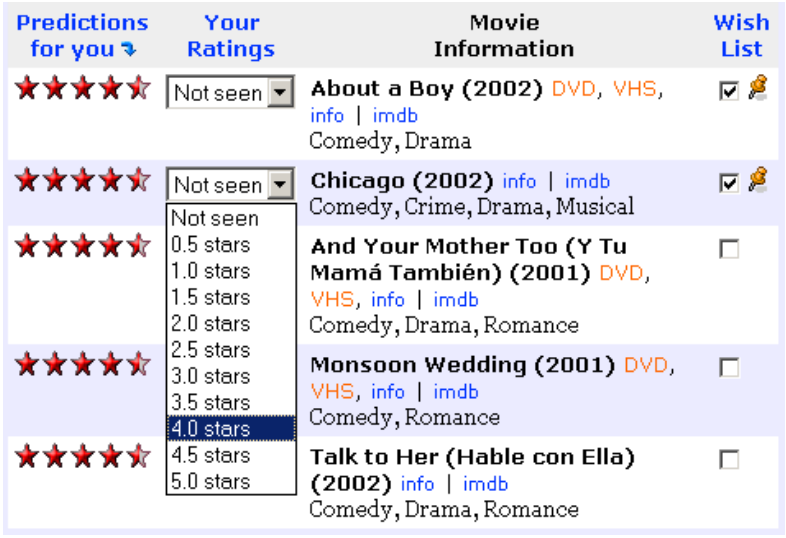
Course dataset 2

- Collection of texts:
  http://people.csail.mit.edu/jrennie/20Newsgroups/
  - Messages from 20 different usenet newsgroups
  - Preprocessed into bag-of-words representation

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

Course dataset 3

- Movielens:
  http://movielens.umn.edu/
  http://www.grouplens.org/node/73



	Seven	Fargo	Aliens	Leon	Avatar
Linda		4		5	5	1		2
			3		4	3		
Jack	1			4		1	5	1
Bill					4	1		
Lucy			2	1	1		5	
John	1				1	4		5
		4				5		5
	2		3				3	

Summary statistics

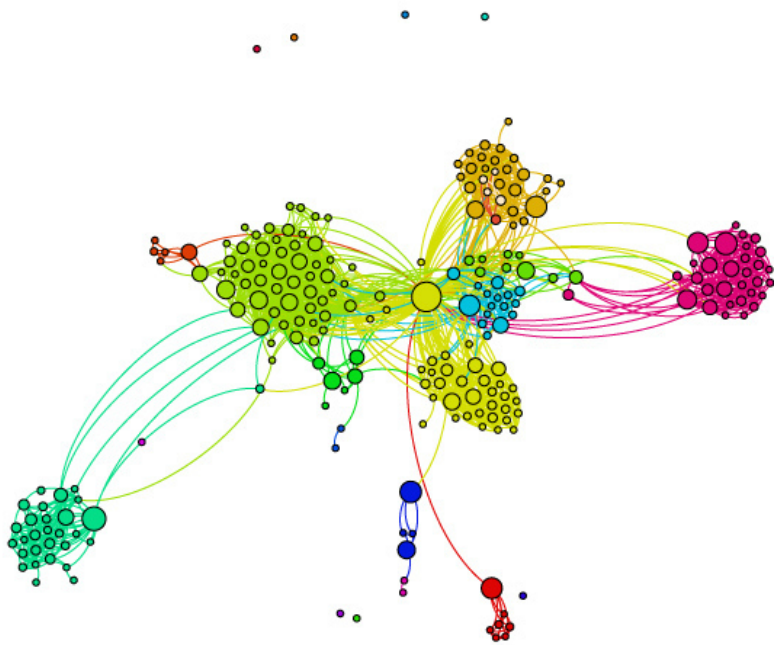
(You should be familiar with most of these already...)

- The *frequency* of an attribute value is the percentage of objects which take that value for that attribute.
- The *mode* of an attribute is the value with highest frequency
- The p^{th} *percentile* x_p of an attribute x is a value such that $p\%$ of the observed values of x are less than x_p
- The *mean* value of an attribute is the sum divided by the number of records
- The *median* value is the 50^{th} percentile
- The *range* is the maximum value minus the minimum value
- The *variance* is the mean squared deviation from the mean, and the *standard deviation* is the square root of the variance
- The *covariance matrix* of a random vector contains the variances of the components on the diagonal and the covariances in off-diagonal elements

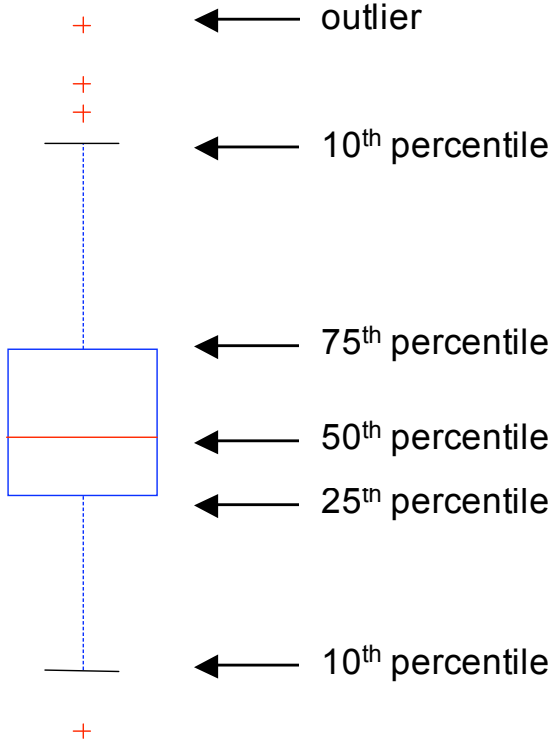
Visualization

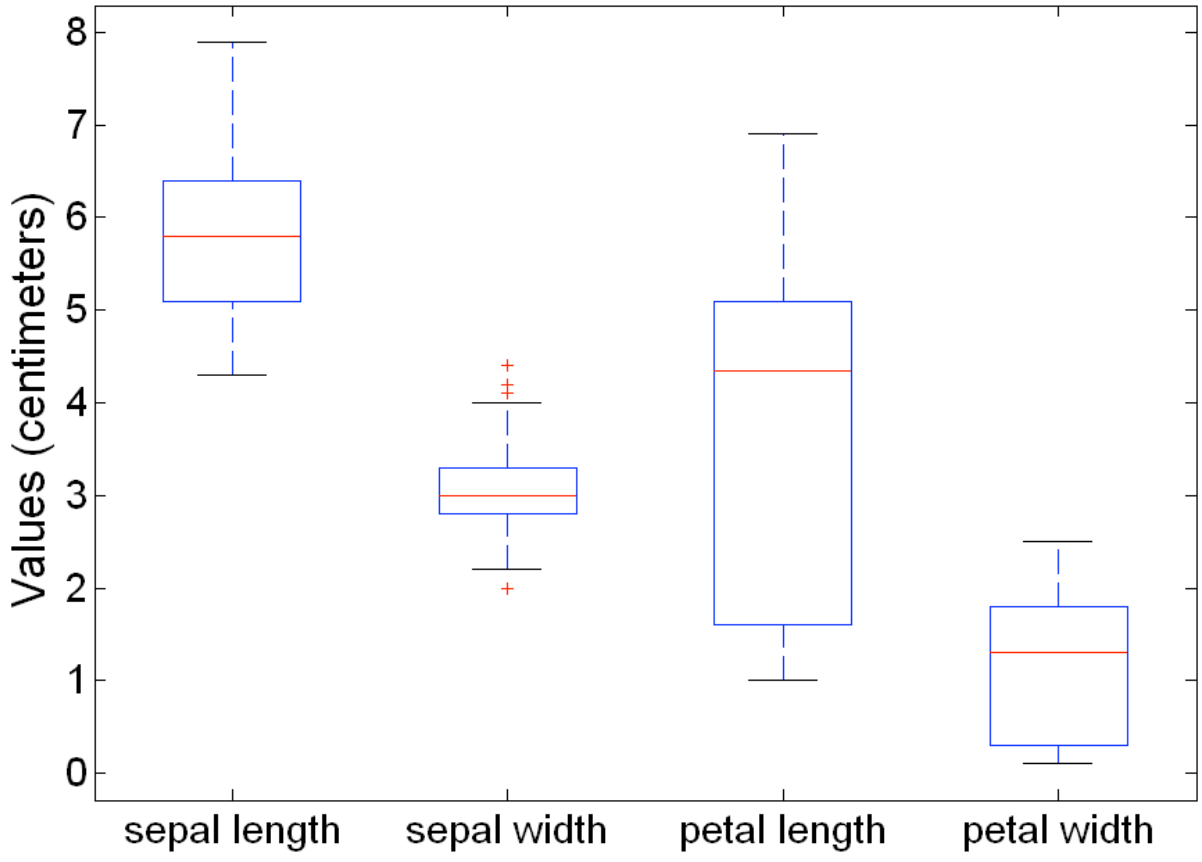
- Goal of visualization:
  Make (some aspect of) data easily understandable to a person

- Why is visualization a useful tool?
  - The human visual system is good at quickly detecting patterns
  - Utilizes domain knowledge of experts that may be hard to formalize
  - Example: Graph of social network (on Facebook)

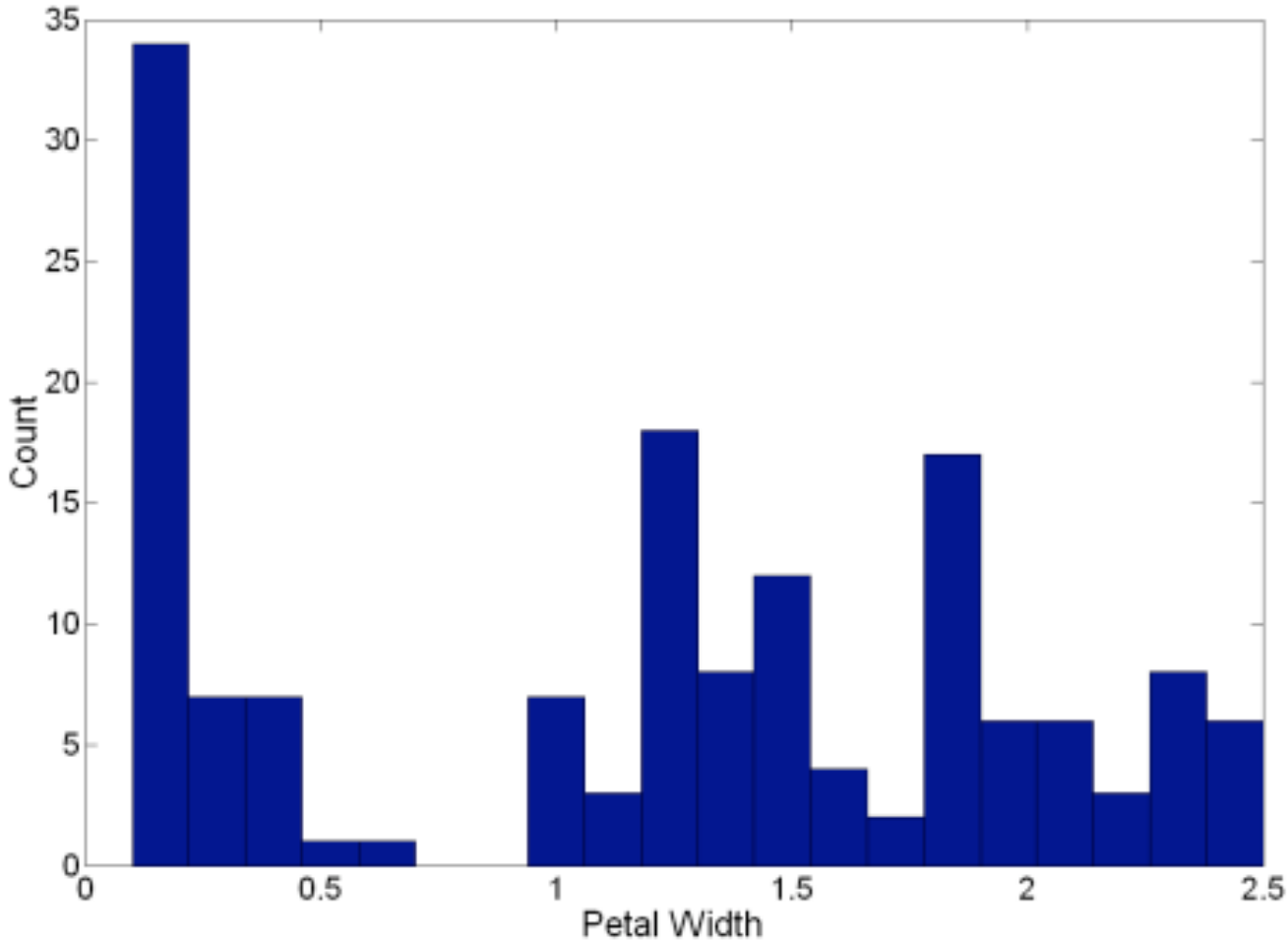


- Examples: Box plot

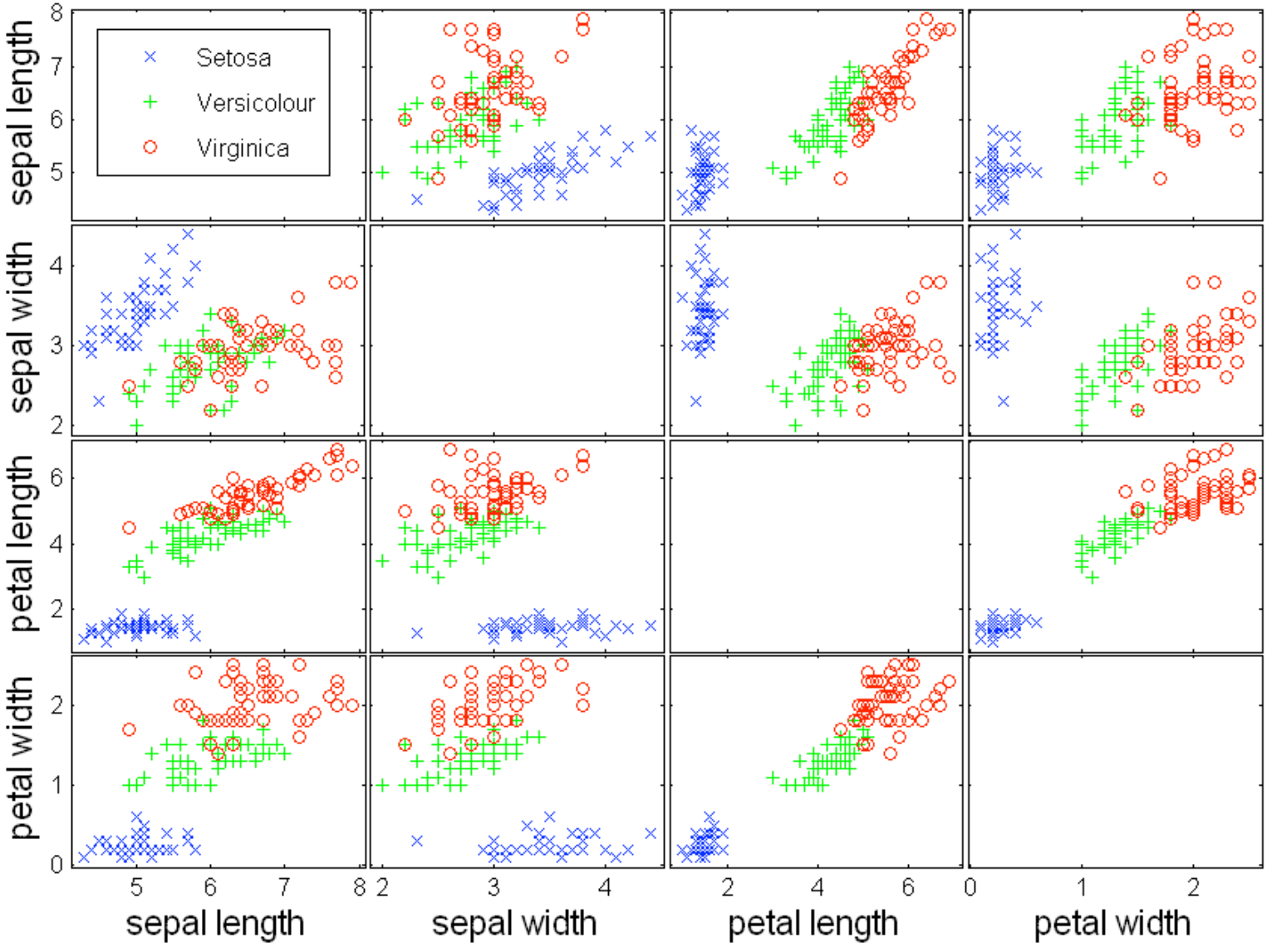




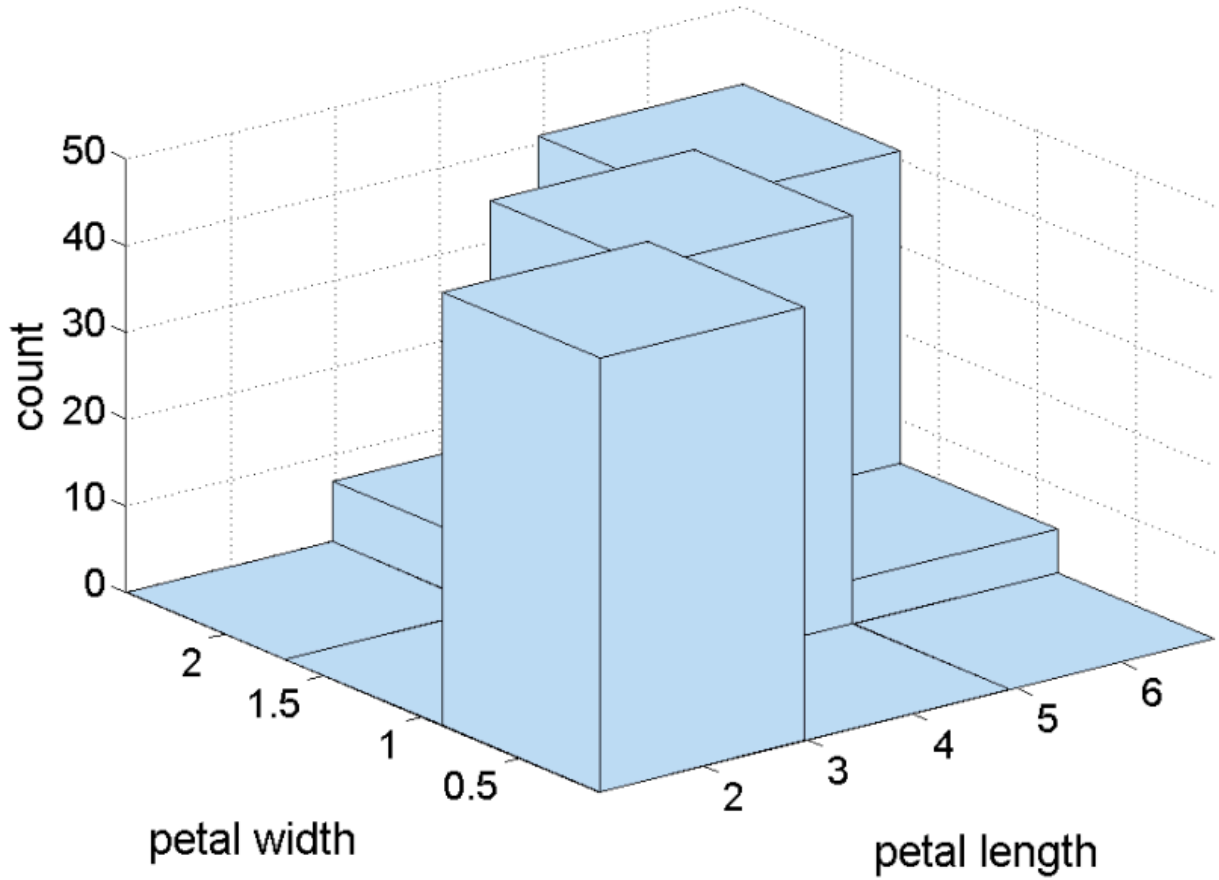
- Examples: Histogram (1d)
  - Bin width and placing important



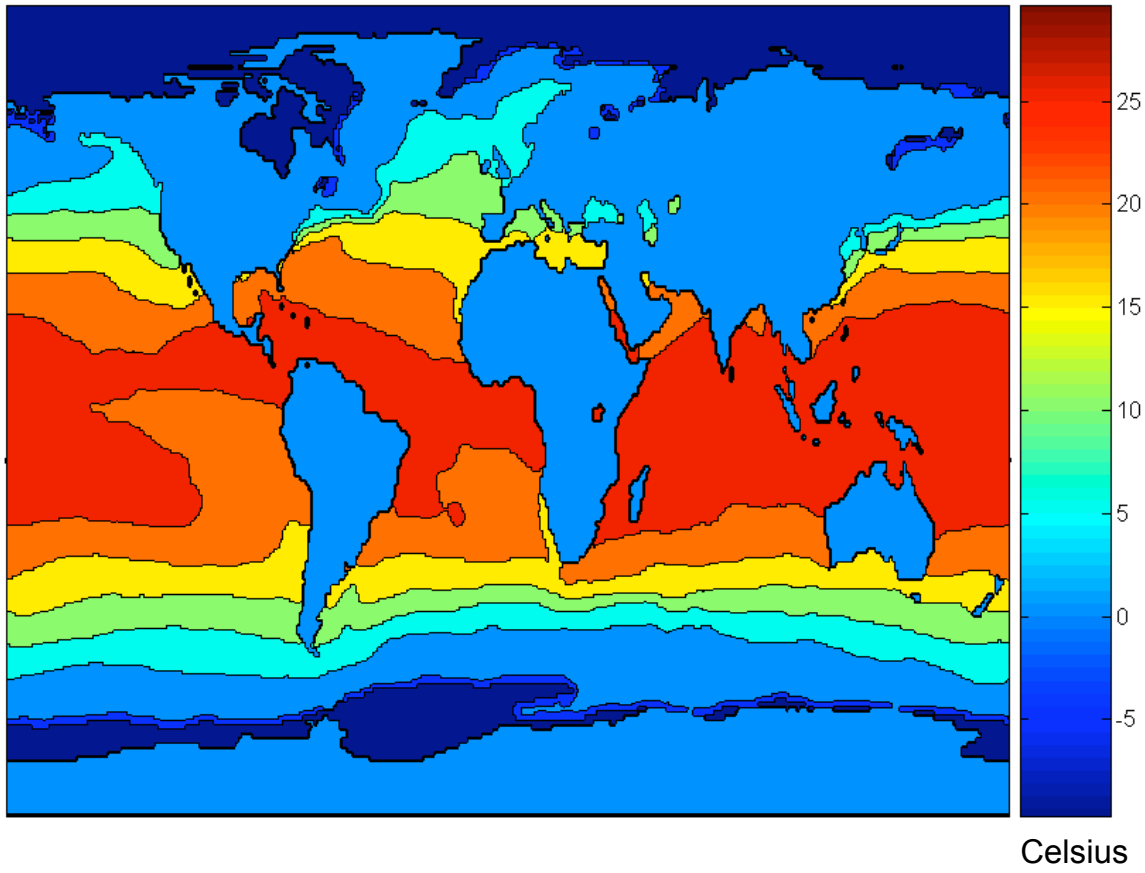
- Examples: Scatterplot



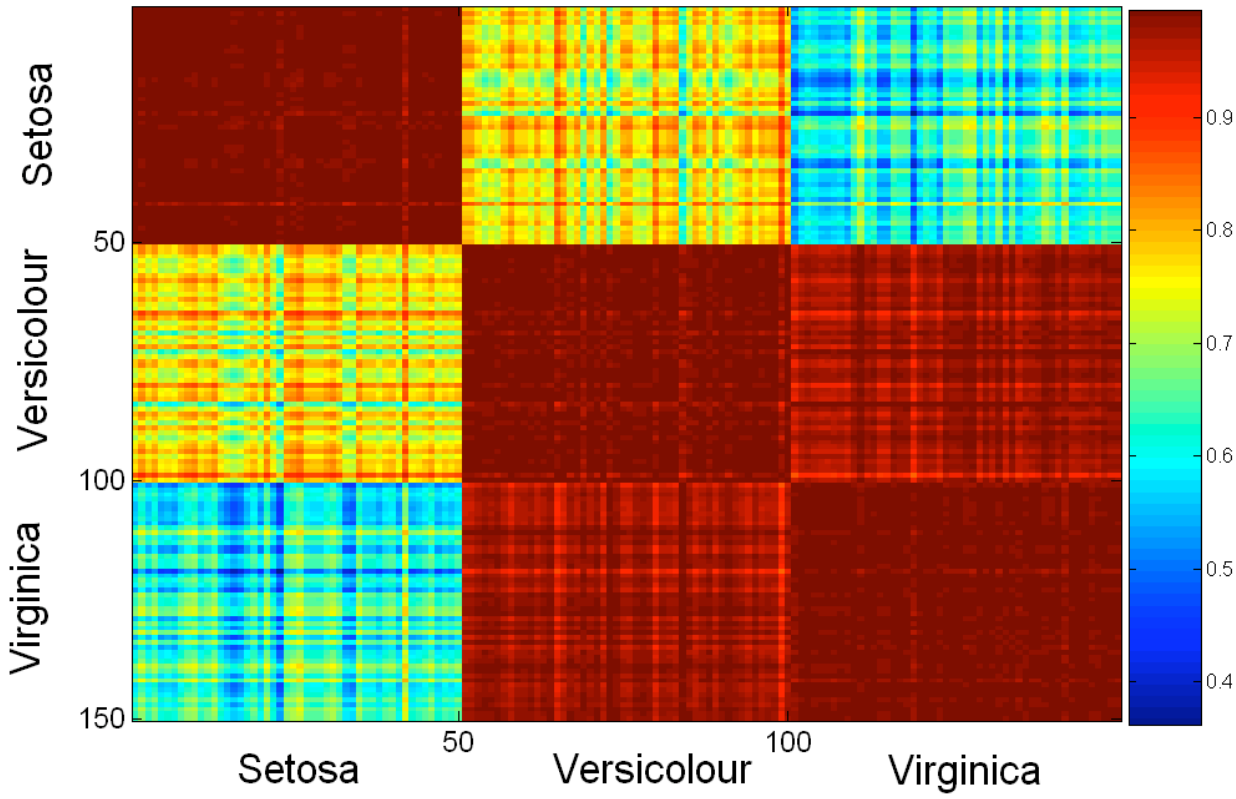
- Examples: Histogram (2d)



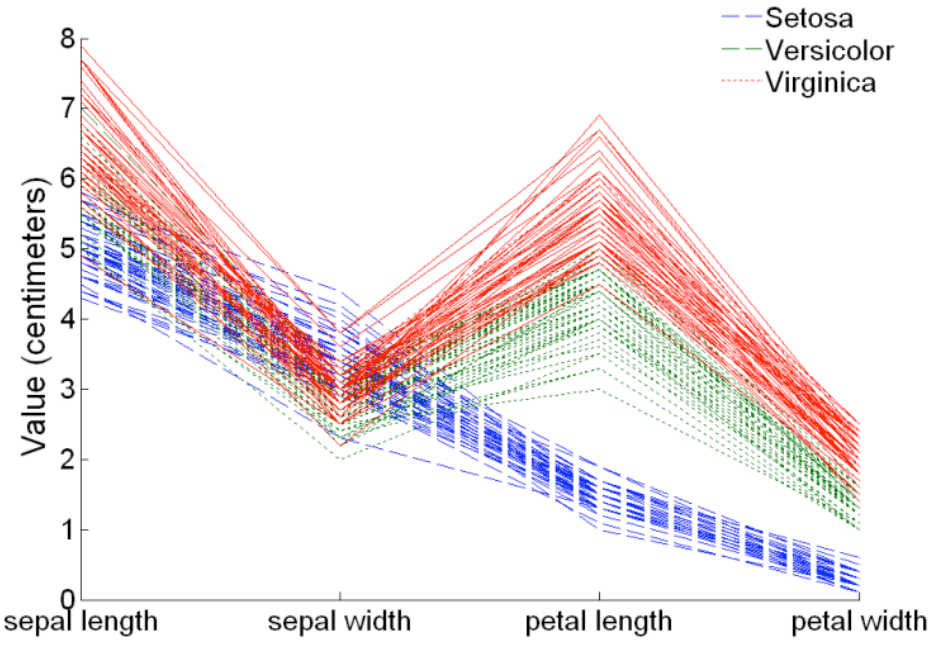
- Examples: Contour plot
  - Note: Usage of color here is questionable because color space is cyclic and not perceptually linear.

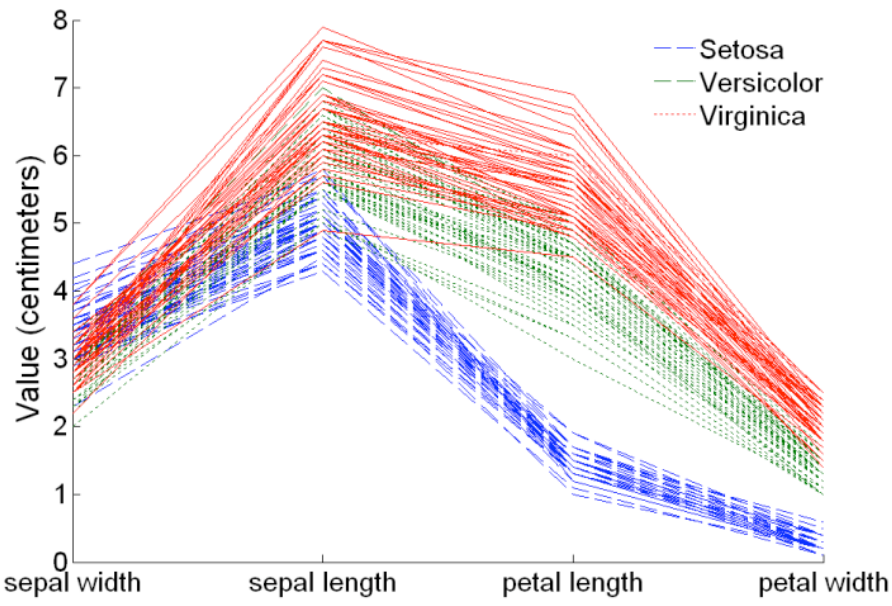


- Examples: Correlation matrix
  - Correlations (or distances) between *pairs of objects* or *pairs of attributes*
  - A good ordering is crucial
    (so this is most useful with labels or after clustering)
  - Again, usage of color scale here is questionable



- Examples: Parallel coordinates
  - Ordering of attributes affects the visualization
  - Color can be used to indicate class or cluster (as in this example)





Classification: Basic concepts

Classification: Basic concepts

- The classification problem
- Cost functions and performance measures
- k Nearest Neighbors (kNN)
- Bayesian classifier
- Naive Bayes
- Generalization performance

The classification problem

- Observe pairs $(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_N, y_N)$ where $\mathbf{x}_i \in \mathcal{X}$, with $\mathcal{X}$ an arbitrary set, and $y_i \in \mathcal{Y}$, with $\mathcal{Y}$ a finite (typically small) set.

- Given a set $\{\mathbf{x}_{N+1} \ldots \mathbf{x}_{N+M}\}$ of *new* objects (each one also $\in \mathcal{X}$), predict the corresponding classes $y_{N+1} \ldots y_{N+M}$ (each one also $\in \mathcal{Y}$). (Typically the new objects are predicted one at a time, independently of each other.)

Example 1: spam filter

- $\mathcal{X}$ is the set of all possible emails (strings)
- $\mathcal{Y}$ is the set { `spam`, `non-spam` }

`From: medshop@spam.com`
`Subject: viagra`
`cheap meds...`
	`spam`

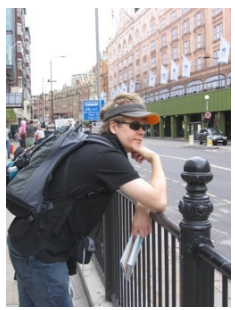
`From: my.professor@helsinki.fi`
`Subject: important information`
`here's how to ace the test...`
	`non-spam`

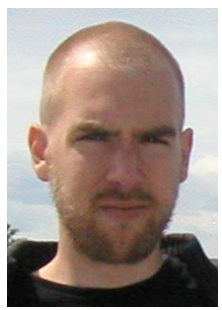
⋮	⋮

`From: mike@example.org`
`Subject: you need to see this`
`how to win $1,000,000...`
	?

Example 2: face recognition

- $\mathcal{X}$ is the set of all possible images
- $\mathcal{Y}$ is the set $\{$ `patrik, doris, antti` $\}$

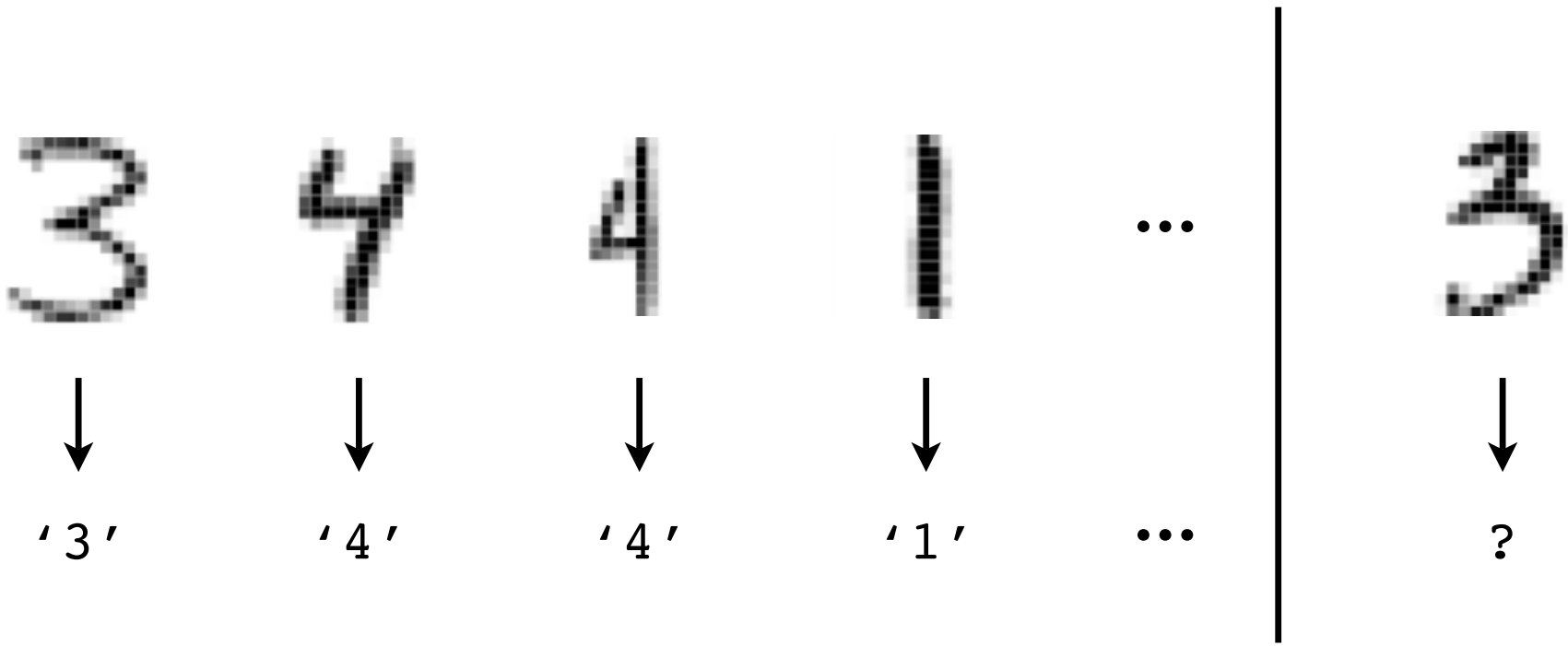






`patrik` `antti` `doris` `patrik` ... | `?`

Example 3: handwritten digit recognition

- $\mathcal{X}$ is the set of all possible images of fixed (small) size
- $\mathcal{Y}$ is the set $\{$ `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9` $\}$



Example 4: credit card fraud detection

- $\mathcal{X}$ is the set of all possible 'last three transactions'
- $\mathcal{Y}$ is the set $\{$ `fraud, normal` $\}$

`Cash advance $1,000`
`Cash advance $10,000`
`Flight out of the country $1,200`
	`fraud`

`Candy bar $1.20`
`Groceries $67.10`
`Restaurant $35.82`
	`normal`

⋮	⋮

`Flight out of the country $380`
`Hotel booking $210`
`Groceries $69.20`
	?

Where do the correct labels come from?

i.e. How do we get labeled training data?

- Predicting the future: time eventually hands us the answer
  - Example: Predicting $\mathcal{Y} = \{$ `rain, no rain` $\}$ tomorrow, based on weather today and yesterday
- Automating something that humans can do: We can use human-labeled data to train the computer to perform the same task
  - Example: Spam filtering, face recognition, ...
- Creating a simple predictor to replace an expensive or slow test: Use the expensive or slow test to build the training data
  - Example: Predicting disease based on symptoms or cheap tests

Prediction type

Given a new input $\mathbf{x}'$, our prediction for the corresponding y' can come in at least one of three different forms:

1. Single class ('forced choice'): $\hat{y}' \in \mathcal{Y}$
2. Single class or "don't know": $\hat{y}' \in (\mathcal{Y} \cup \{\text{"don't know"}\})$
3. Probability distribution over classes: $\widehat{P}(y)$ with $\forall y \in \mathcal{Y} : \widehat{P}(y) \geq 0$ and $\sum_y \widehat{P}(y) = 1$

Note: Unless otherwise stated, the default in the remainder of this course will be prediction type (1), i.e. 'forced choice'.

Prediction type: Example

Predicting whether tomorrow it will

A. Rain a lot (more than 5mm)
B. Rain a little (less than 5mm but more than 0mm)
C. Not rain (0mm)

Prediction type:

1. Forced choice:

date	prediction	outcome
1.1	A	A
2.1	B	A
3.1	A	B
4.1	B	C
5.1	C	C
...	...	...

2. Allowing for "don't know" answer:

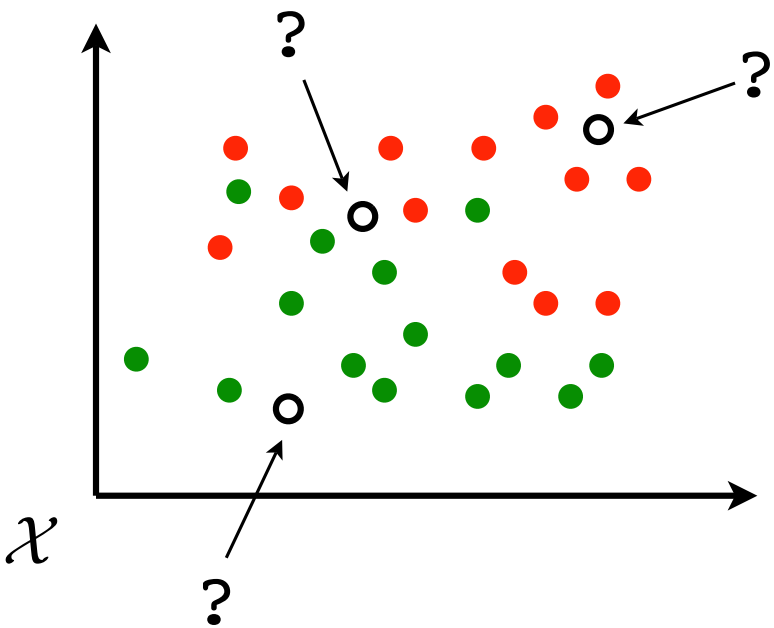
date	prediction	outcome
1.1	A	A
2.1	don't know	A
3.1	A	B
4.1	B	C
5.1	don't know	C
...	...	...

2. Probability distribution over the classes:

date	prediction	outcome
1.1	(A: 0.6, B: 0.3, C: 0.1)	A
2.1	(A: 0.5, B: 0.45, C: 0.05)	A
3.1	(A: 0.8, B: 0.1, C: 0.1)	B
4.1	(A: 0.3, B: 0.34, C: 0.36)	C
5.1	(A: 0.2, B: 0.4, C: 0.4)	C
...	...	...

Illustrative 2d classification example

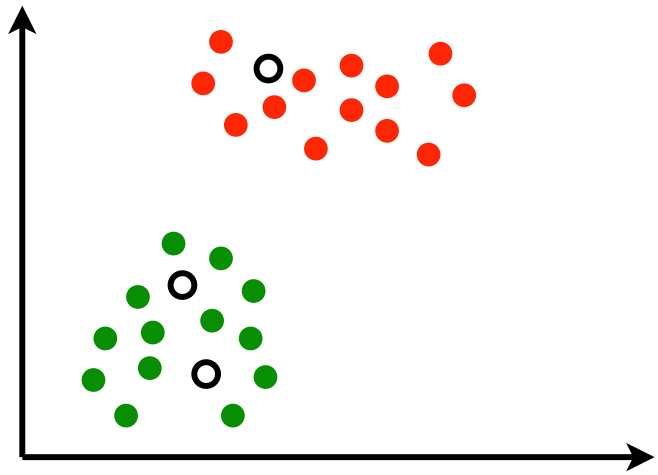
- Point clouds in 2d useful for illustrating basic ideas and approaches...
- ...nevertheless, always keep in mind that real problems can have hundreds (or thousands) of dimensions!

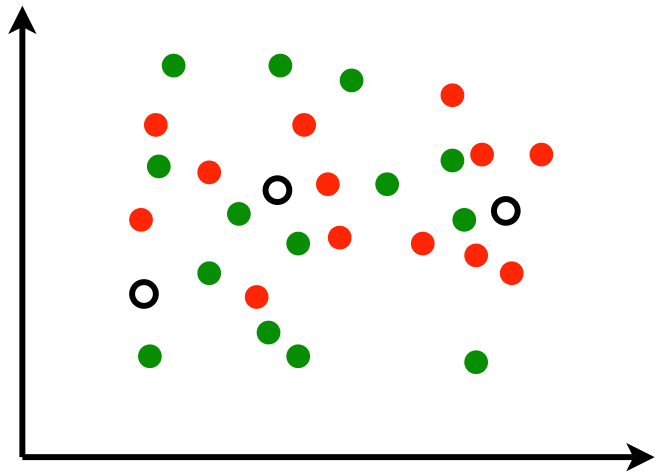


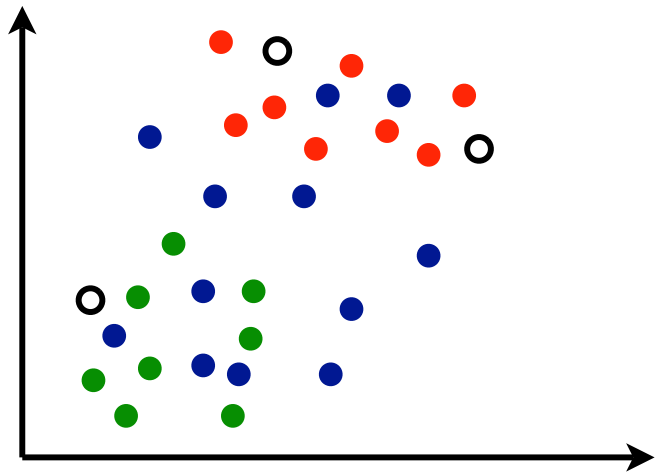
- training example of class 1
- training example of class 2
- test example

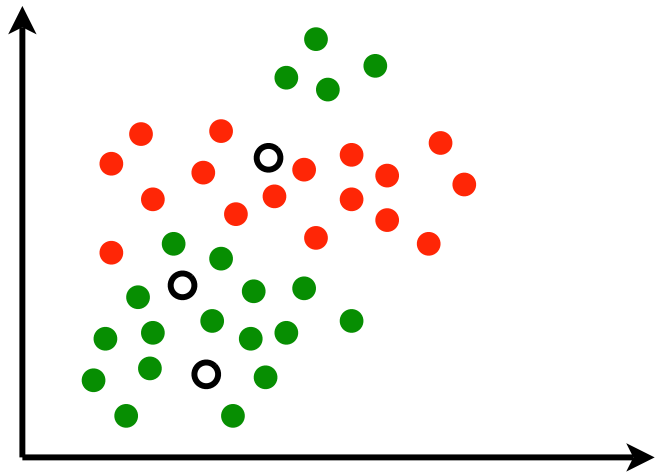
More 2d classification examples

- Separable, linearly separable, non-separable, multiclass, etc



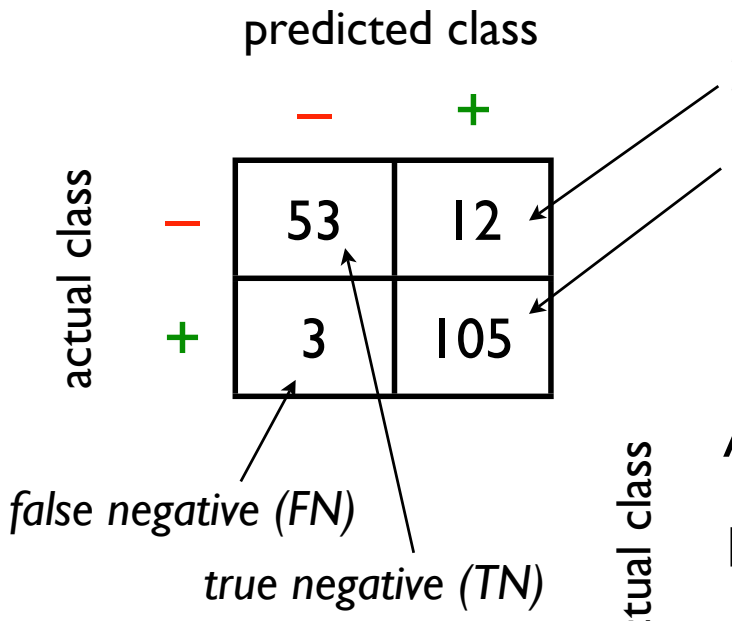






'Confusion' matrix

- For prediction type (1) and (2), we can analyze classification performance as follows:
  - For each data object, store both the true class (y) and the predicted class ($\hat{y}$)
  - Count how many times each pair happened



predicted class

actual class	A	B	C	don't know
A	82	13	3	16
B	2	12	17	11
C	5	24	92	19

Performance metrics for binary classification

- Accuracy $= (TP + TN)/(TP + TN + FP + FN)$
- Error rate $= 1 -$ Accuracy
- True positive rate ('sensitivity') $= TP/(TP + FN)$
- True negative rate ('specificity') $= TN/(TN + FP)$
- False positive rate $= FP/(TN + FP)$
- False negative rate $= FN/(TP + FN)$
- Recall $= TP/(TP + FN)$
- Precision $= TP/(TP + FP)$

		predicted class: −	predicted class: +
actual class	−	TN	FP
actual class	+	FN	TP

Application-specific cost/utility function

- In some applications all correct answers are equally good and all wrong answers equally bad $\Rightarrow$ '0/1 cost'
- In many applications different types of mistakes have different costs
  - Spam filtering: A spam message in the inbox is only a minor annoyance; an important message in the trash is very bad!
  - HIV test: A false positive only results in some additional tests, while a false negative can delay treatment and be fatal
- Formulating problem in terms of 'utility' is equivalent to using 'cost' (just change the sign!)
- Maximize expected utility = minimize expected cost
  (gives best performance in the long run)

Cost function for forced choice

- $C = \text{func}(\text{true class}, \text{predicted class})$
  i.e. a matrix of size $K \times K$, for K classes
- Cost matrix not symmetric in general
- Costs can be dollars, but this assumes 'utility' linear in money
- Goal: minimize expected cost
  (= average cost over many examples)

actual class \ predicted class	–	+
–	\$0	\$1
+	\$30	\$0

Expected cost

Prerequisites reminder: (see Appendix C in the textbook)

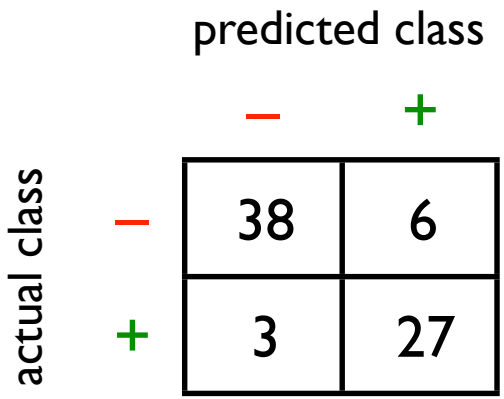
- The *expected value* of a random variable C is given by

$$E\{C\} = \sum_c c\, P(C = c) \tag{6}$$

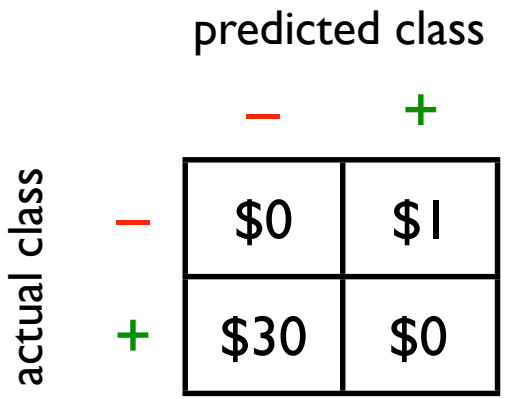
  For instance, if the distribution of C is $P(C = \$0) = 0.2$, $P(C = \$1) = 0.5$, and $P(C = \$8) = 0.3$, then we have $E\{C\} = \$0 \times 0.2 + \$1 \times 0.5 + \$8 \times 0.3 = \$(0 + 0.5 + 2.4) = \$2.9$

- This is the *long run average cost*: When sampling C from its distribution a large number of times the average cost converges to the expected cost.

- The total cost obtained by the classifier is obtained by elementwise multiplication of the confusion matrix with the cost matrix. In the example below, the total cost is $\$0 \times 38 + \$1 \times 6 + \$30 \times 3 + \$0 \times 27 = \$96$.



Confusion matrix



Cost matrix

- The optimal 'forced choice' answer can be computed from probabilistic predictions, and is not always the most likely class (example on next slide)

Example:

Using the cost matrix below, if the probabilistic prediction is $P(y = \text{`}-\text{'}) = a$, and $P(y = \text{`}+\text{'}) = 1 - a$, then

Expected cost of predicting '−' $= a \cdot \$0 + (1 - a) \cdot \$30 = \$30(1 - a)$

Expected cost of predicting '+' $= a \cdot \$1 + (1 - a) \cdot \$0 = \$a$

Setting these two equal yields $30(1 - a) = a$ which is solved by $a = 30/31 \approx 0.968$. Whenever the probability of the negative class is higher than this, one should predict negative; in all other cases, predicting positive is the appropriate decision.

		predicted class: −	predicted class: +
actual class	−	\$0	\$1
	+	\$30	\$0

Cost function allowing for "don't know" answer

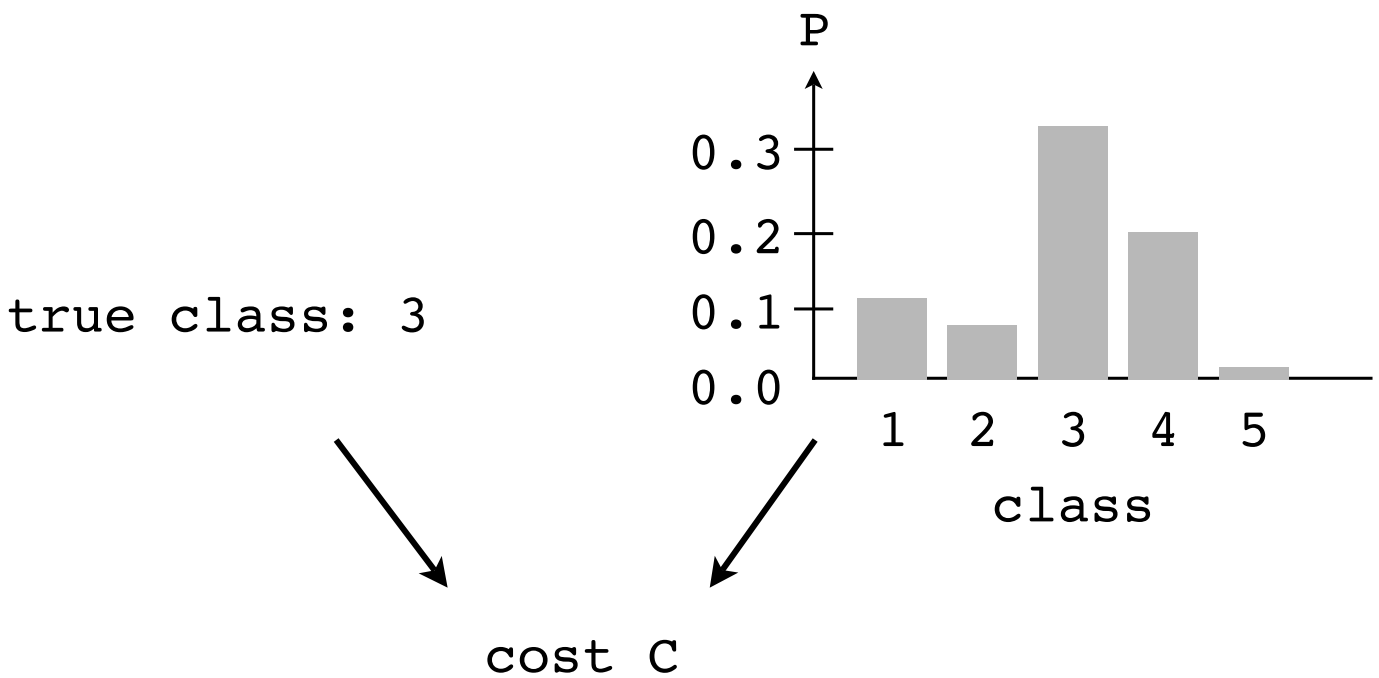
- $C = \text{func}(\text{true class}, \text{predicted class})$
  i.e. a matrix of size $K \times (K+1)$, for K classes

- Otherwise exactly as for the 'forced choice' case. Using probabilistic predictions we could compute the best answer, which would typically be "don't know" when we are not sure enough.

actual class	predicted class: −	predicted class: +	predicted class: don't know
−	\$0	\$1	\$0.50
+	\$30	\$0	\$0.50

Cost function for probabilistic predictions

- $C = \text{func}(y_t, \widehat{P})$, where y_t takes any value in $\mathcal{Y}$, and $\widehat{P}$ is a length K unit sum non-negative vector that specifies the predicted probabilities for each of the K classes
  - Example 1: linear cost $C = 1 - \widehat{P}(y_t)$
  - Example 2: logarithmic cost $C = -\log \widehat{P}(y_t)$



- A cost function for probabilistic predictions is *proper* if, whenever the true probability distribution over the classes is P, the expected cost is minimized for $\widehat{P} = P$.

  - The linear cost is *not* proper (and hence should be used only with caution):

    Say the true distribution is $P(y = 0) = 1/4$, $P(y = 1) = 3/4$, and set the predicted distribution to $\widehat{P}(y = 0) = b$, $\widehat{P}(y = 1) = 1 - b$. The expected cost is

$$\begin{aligned} E\{C\} &= E_{y_t}\{1 - \widehat{P}(y = y_t)\} \\ &= \frac{1}{4}(1 - b) + \frac{3}{4}(1 - (1 - b)) \\ &= \frac{1}{4} + \frac{1}{2}b \end{aligned}$$

    which is minimized for $b = 0$, not the true probability $b = 1/4$.

The logarithmic cost *is proper* (only binary case shown below):

Say the true probability of class $y = 0$ is a (so the true probability of class $y = 1$ is $1 - a$). Assume $0 < a < 1$.

Our prediction is that the probability of $y = 0$ is b (so we are also predicting that the probability of $y = 1$ is $1 - b$). Assume $0 < b < 1$.

The expected logarithmic loss is:

$$E\{C\} = -a \log b - (1 - a) \log(1 - b)$$

Let's find the value of b that minimizes the expected logarithmic loss. Taking the derivative with respect to b gives:

$$\frac{d}{db} E\{C\} = -a \frac{1}{b} - (1 - a) \frac{1}{1 - b}(-1) = -\frac{a}{b} + \frac{1 - a}{1 - b}$$

Setting this to zero gives:

$$\frac{1 - a}{1 - b} = \frac{a}{b} \quad \Longleftrightarrow \quad b - ab = a - ab \quad \Longleftrightarrow \quad b = a$$

Take the second derivative with respect to b:

$$\frac{d^2}{db^2} E\{C\} = \frac{a}{b^2} - \frac{1-a}{(1-b)^2}(-1) = \frac{a}{b^2} + \frac{1-a}{(1-b)^2} > 0$$

Hence $b = a$ is the unique minimum, and so the logarithmic cost is proper according to the definition.

Probabilistic predictions: Calibration

- Informally, a probabilistic predictor is calibrated if the observed relative frequencies agree with the predicted probabilities.
- Definition: A probabilistic predictor is *calibrated* if, whenever the predicted probability distribution is $\widehat{P}(y \mid x) = \alpha$, the observed relative frequencies of events is also α, for all values of α.
- Calibration is a desirable property of probabilistic predictions, independent of the cost function.
- For example: In those cases where $\widehat{P}(\texttt{ rain }) = 0.2$, how often did it actually rain? If it rained in exactly 20% of such cases, then *this prediction* was calibrated. If this holds for all predictions (all probabilities), then the probabilistic predictor is calibrated.

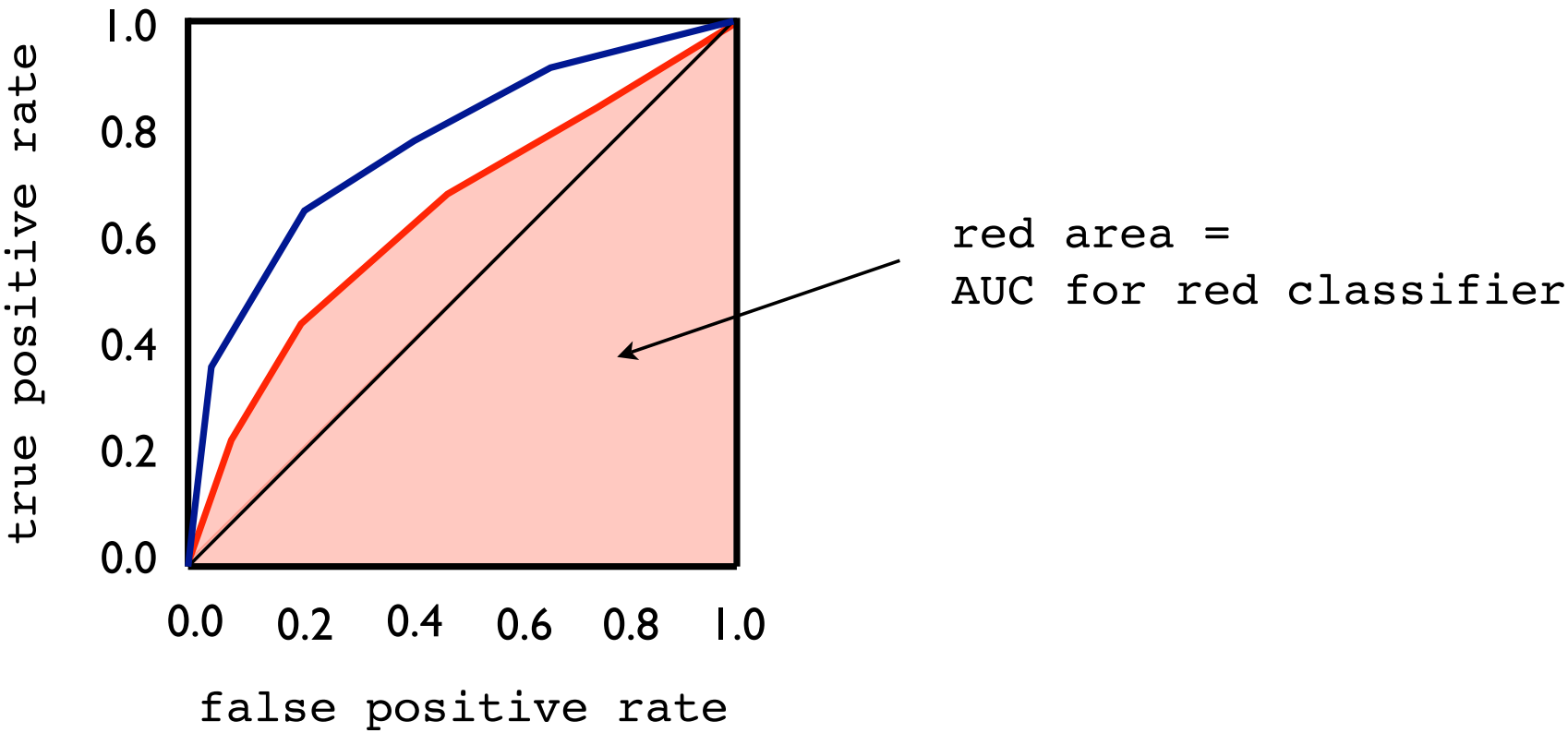
ROC curve for binary classification

- For binary variables, using probabilistic predictions (or any real-numbered predictions that indicate some level of confidence), we can plot a *Receiver Operating Characteristic* (ROC) curve (example on next slide):

  1. Sort the records into increasing estimated probability of positive class
  2. For i in $1 \ldots N$
     - 2.1 Classify all records $j < i$ as negatives, and all records $j \geq i$ as positives
     - 2.2 Plot the corresponding *false positive rate* vs *true positive rate* as a point in the graph (and connect the points into a curve)

- One performance measure is AUC: Area under ROC curve ($0 \leq AUC \leq 1$)

ROC curve for binary classification: Example



ROC curve for binary classification: Example

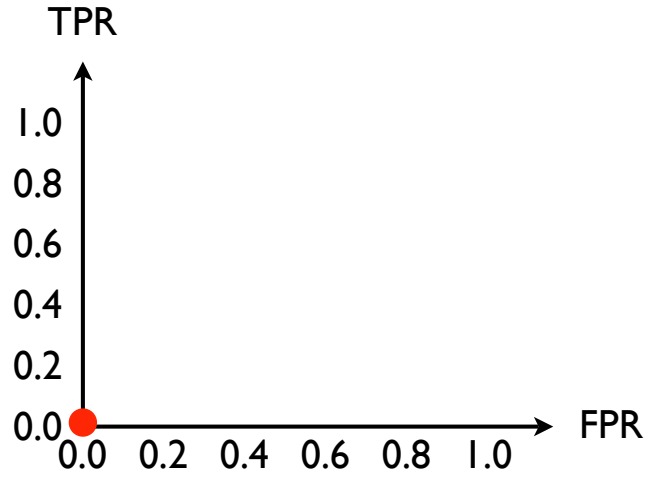
instance	$P(y = \text{`+'} \mid \mathbf{x})$	true class
1	0.92	+
2	0.34	–
3	0.56	–
4	0.49	+
5	0.87	+
6	0.54	+
7	0.13	–
8	0.70	+
9	0.60	+
10	0.53	–

$\Rightarrow$

instance	$P(y = \text{`+'} \mid \mathbf{x})$	true class
7	0.13	–
2	0.34	–
4	0.49	+
10	0.53	–
6	0.54	+
3	0.56	–
9	0.60	+
8	0.70	+
5	0.87	+
1	0.92	+

TPR = TP / (TP+FN) = 'what proportion of all actual positives where classified as positive?'

FPR = FP / (TN+FP) = 'what proportion of all actual negative where classified as positive?'



ROC curve for binary classification: Example

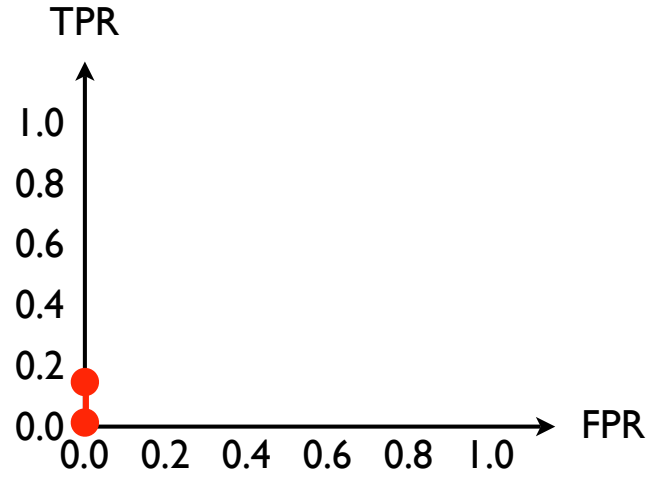
instance	$P(y = \text{`+'} \mid \mathbf{x})$	true class
1	0.92	+
2	0.34	–
3	0.56	–
4	0.49	+
5	0.87	+
6	0.54	+
7	0.13	–
8	0.70	+
9	0.60	+
10	0.53	–

$\Rightarrow$

instance	$P(y = \text{`+'} \mid \mathbf{x})$	true class
7	0.13	–
2	0.34	–
4	0.49	+
10	0.53	–
6	0.54	+
3	0.56	–
9	0.60	+
8	0.70	+
5	0.87	+
1	0.92	+

TPR = TP / (TP+FN) = 'what proportion of all actual positives where classified as positive?'

FPR = FP / (TN+FP) = 'what proportion of all actual negative where classified as positive?'



ROC curve for binary classification: Example

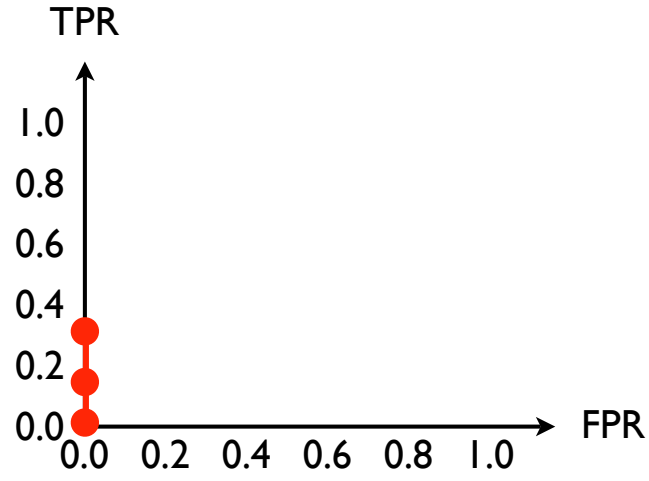
instance	$P(y = \text{'+'} \mid \mathbf{x})$	true class
1	0.92	+
2	0.34	–
3	0.56	–
4	0.49	+
5	0.87	+
6	0.54	+
7	0.13	–
8	0.70	+
9	0.60	+
10	0.53	–

$\Rightarrow$

instance	$P(y = \text{'+'} \mid \mathbf{x})$	true class
7	0.13	–
2	0.34	–
4	0.49	+
10	0.53	–
6	0.54	+
3	0.56	–
9	0.60	+
8	0.70	+
5	0.87	+
1	0.92	+

TPR = TP / (TP+FN) = 'what proportion of all actual positives where classified as positive?'

FPR = FP / (TN+FP) = 'what proportion of all actual negative where classified as positive?'



ROC curve for binary classification: Example

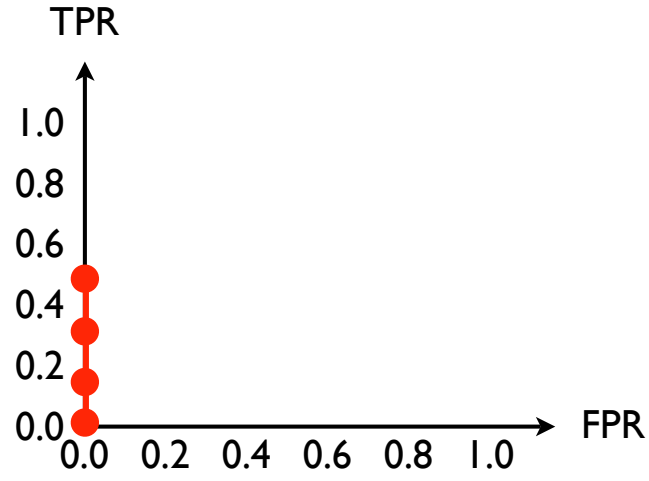
instance	$P(y = \text{`+'} \mid \mathbf{x})$	true class
1	0.92	+
2	0.34	–
3	0.56	–
4	0.49	+
5	0.87	+
6	0.54	+
7	0.13	–
8	0.70	+
9	0.60	+
10	0.53	–

$\Rightarrow$

instance	$P(y = \text{`+'} \mid \mathbf{x})$	true class
7	0.13	–
2	0.34	–
4	0.49	+
10	0.53	–
6	0.54	+
3	0.56	–
9	0.60	+
8	0.70	+
5	0.87	+
1	0.92	+

TPR = TP / (TP+FN) = 'what proportion of all actual positives where classified as positive?'

FPR = FP / (TN+FP) = 'what proportion of all actual negative where classified as positive?'



ROC curve for binary classification: Example

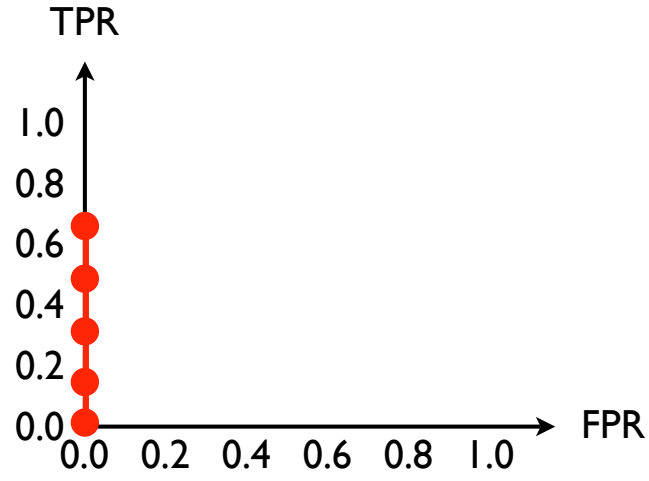
instance	$P(y = \text{`+'} \mid \mathbf{x})$	true class
1	0.92	+
2	0.34	–
3	0.56	–
4	0.49	+
5	0.87	+
6	0.54	+
7	0.13	–
8	0.70	+
9	0.60	+
10	0.53	–

$\Rightarrow$

instance	$P(y = \text{`+'} \mid \mathbf{x})$	true class
7	0.13	–
2	0.34	–
4	0.49	+
10	0.53	–
6	0.54	+
3	0.56	–
9	0.60	+
8	0.70	+
5	0.87	+
1	0.92	+

TPR = TP / (TP+FN) = 'what proportion of all actual positives where classified as positive?'

FPR = FP / (TN+FP) = 'what proportion of all actual negative where classified as positive?'



ROC curve for binary classification: Example

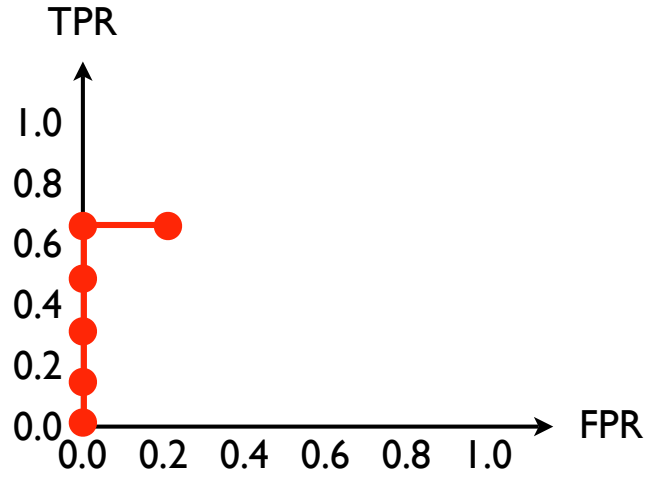
instance	$P(y = \text{'+'} \mid \mathbf{x})$	true class
1	0.92	+
2	0.34	–
3	0.56	–
4	0.49	+
5	0.87	+
6	0.54	+
7	0.13	–
8	0.70	+
9	0.60	+
10	0.53	–

$\Rightarrow$

instance	$P(y = \text{'+'} \mid \mathbf{x})$	true class
7	0.13	–
2	0.34	–
4	0.49	+
10	0.53	–
6	0.54	+
3	0.56	–
9	0.60	+
8	0.70	+
5	0.87	+
1	0.92	+

TPR = TP / (TP+FN) = 'what proportion of all actual positives where classified as positive?'

FPR = FP / (TN+FP) = 'what proportion of all actual negative where classified as positive?'



ROC curve for binary classification: Example

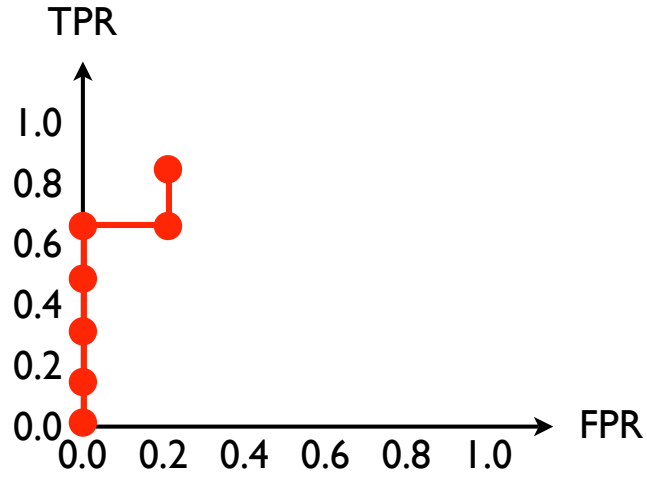
instance	$P(y = \text{`+'} \mid \mathbf{x})$	true class
1	0.92	+
2	0.34	–
3	0.56	–
4	0.49	+
5	0.87	+
6	0.54	+
7	0.13	–
8	0.70	+
9	0.60	+
10	0.53	–

$\Rightarrow$

instance	$P(y = \text{`+'} \mid \mathbf{x})$	true class
7	0.13	–
2	0.34	–
4	0.49	+
10	0.53	–
6	0.54	+
3	0.56	–
9	0.60	+
8	0.70	+
5	0.87	+
1	0.92	+

TPR = TP / (TP+FN) = 'what proportion of all actual positives where classified as positive?'

FPR = FP / (TN+FP) = 'what proportion of all actual negative where classified as positive?'



ROC curve for binary classification: Example

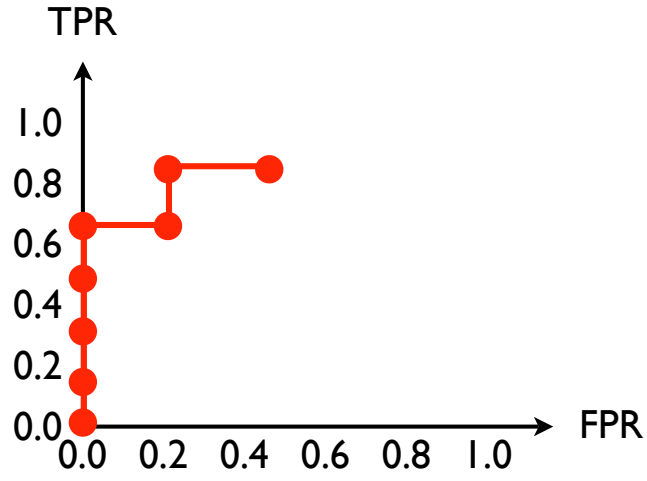
instance	$P(y = \text{`+'} \mid \mathbf{x})$	true class
1	0.92	+
2	0.34	–
3	0.56	–
4	0.49	+
5	0.87	+
6	0.54	+
7	0.13	–
8	0.70	+
9	0.60	+
10	0.53	–

$\Rightarrow$

instance	$P(y = \text{`+'} \mid \mathbf{x})$	true class
7	0.13	–
2	0.34	–
4	0.49	+
10	0.53	–
6	0.54	+
3	0.56	–
9	0.60	+
8	0.70	+
5	0.87	+
1	0.92	+

TPR = TP / (TP+FN) = 'what proportion of all actual positives where classified as positive?'

FPR = FP / (TN+FP) = 'what proportion of all actual negative where classified as positive?'



ROC curve for binary classification: Example

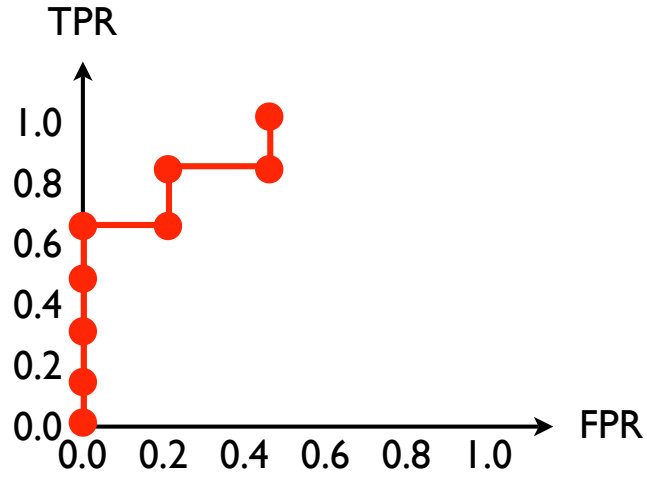
instance	$P(y = \text{`+'} \mid \mathbf{x})$	true class
1	0.92	+
2	0.34	–
3	0.56	–
4	0.49	+
5	0.87	+
6	0.54	+
7	0.13	–
8	0.70	+
9	0.60	+
10	0.53	–

$\Rightarrow$

instance	$P(y = \text{`+'} \mid \mathbf{x})$	true class
7	0.13	–
2	0.34	–
4	0.49	+
10	0.53	–
6	0.54	+
3	0.56	–
9	0.60	+
8	0.70	+
5	0.87	+
1	0.92	+

TPR = TP / (TP+FN) = 'what proportion of all actual positives where classified as positive?'

FPR = FP / (TN+FP) = 'what proportion of all actual negative where classified as positive?'



ROC curve for binary classification: Example

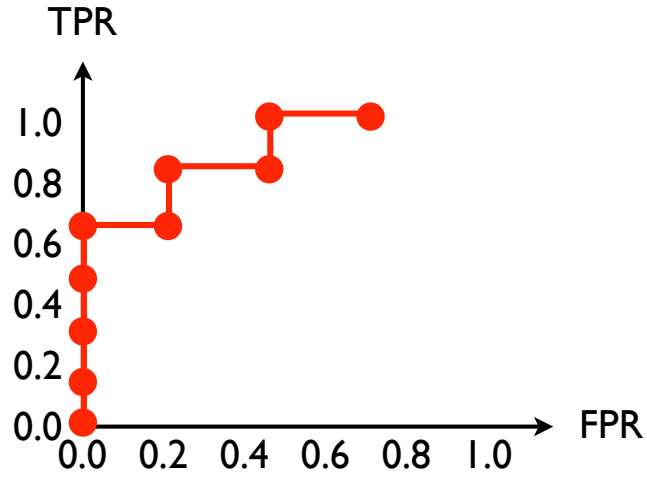
instance	$P(y = \text{'+'} \mid \mathbf{x})$	true class
1	0.92	+
2	0.34	–
3	0.56	–
4	0.49	+
5	0.87	+
6	0.54	+
7	0.13	–
8	0.70	+
9	0.60	+
10	0.53	–

$\Rightarrow$

instance	$P(y = \text{'+'} \mid \mathbf{x})$	true class
7	0.13	–
2	0.34	–
4	0.49	+
10	0.53	–
6	0.54	+
3	0.56	–
9	0.60	+
8	0.70	+
5	0.87	+
1	0.92	+

TPR = TP / (TP+FN) = 'what proportion of all actual positives where classified as positive?'

FPR = FP / (TN+FP) = 'what proportion of all actual negative where classified as positive?'



ROC curve for binary classification: Example

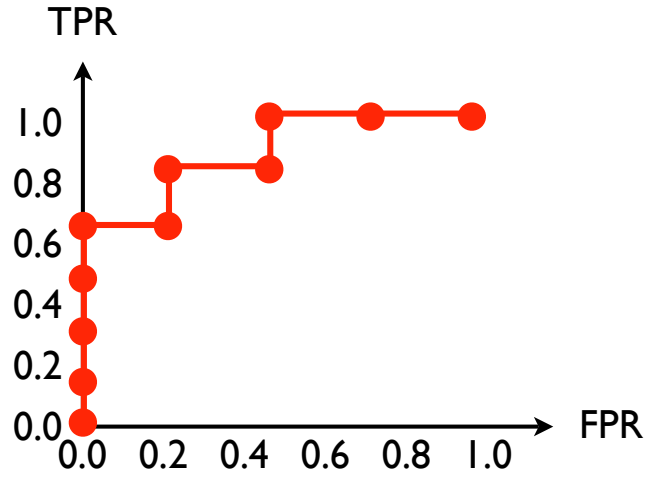
instance	$P(y = \text{`+'} \mid \mathbf{x})$	true class
1	0.92	+
2	0.34	–
3	0.56	–
4	0.49	+
5	0.87	+
6	0.54	+
7	0.13	–
8	0.70	+
9	0.60	+
10	0.53	–

$\Rightarrow$

instance	$P(y = \text{`+'} \mid \mathbf{x})$	true class
7	0.13	–
2	0.34	–
4	0.49	+
10	0.53	–
6	0.54	+
3	0.56	–
9	0.60	+
8	0.70	+
5	0.87	+
1	0.92	+

TPR = TP / (TP+FN) = 'what proportion of all actual positives where classified as positive?'

FPR = FP / (TN+FP) = 'what proportion of all actual negative where classified as positive?'



The classification problem: full specification

- Observe pairs $(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_N, y_N)$ where $\mathbf{x}_i \in \mathcal{X}$, with $\mathcal{X}$ an arbitrary set, and $y_i \in \mathcal{Y}$, with $\mathcal{Y}$ a finite (typically small) set.

- Given a set $\{\mathbf{x}_{N+1} \ldots \mathbf{x}_{N+M}\}$ of *new* objects (each one also $\in \mathcal{X}$), predict the corresponding classes $y_{N+1} \ldots y_{N+M}$ (each one also $\in \mathcal{Y}$). (Typically the new objects are predicted one at a time, independently of each other.)

Full specification:

- $\mathcal{X}$, $\mathcal{Y}$, and N training dataset pairs $(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_N, y_N)$
- Prediction type
- Objective function (minimizing expected cost, or optimizing some other given performance metric)

Goal: optimize the objective function for *new* examples $(\mathbf{x}', y')$.

Simply 'memorizing the training data'

- Simple classifier, with "don't know" output option:

  When presented with a new object $\mathbf{x}'$ to classify, check whether $\mathbf{x}' = \mathbf{x}_i$ for some $\mathbf{x}_i$ in the training data.

  - If so, return the corresponding y_i.
  - If not, return "don't know"

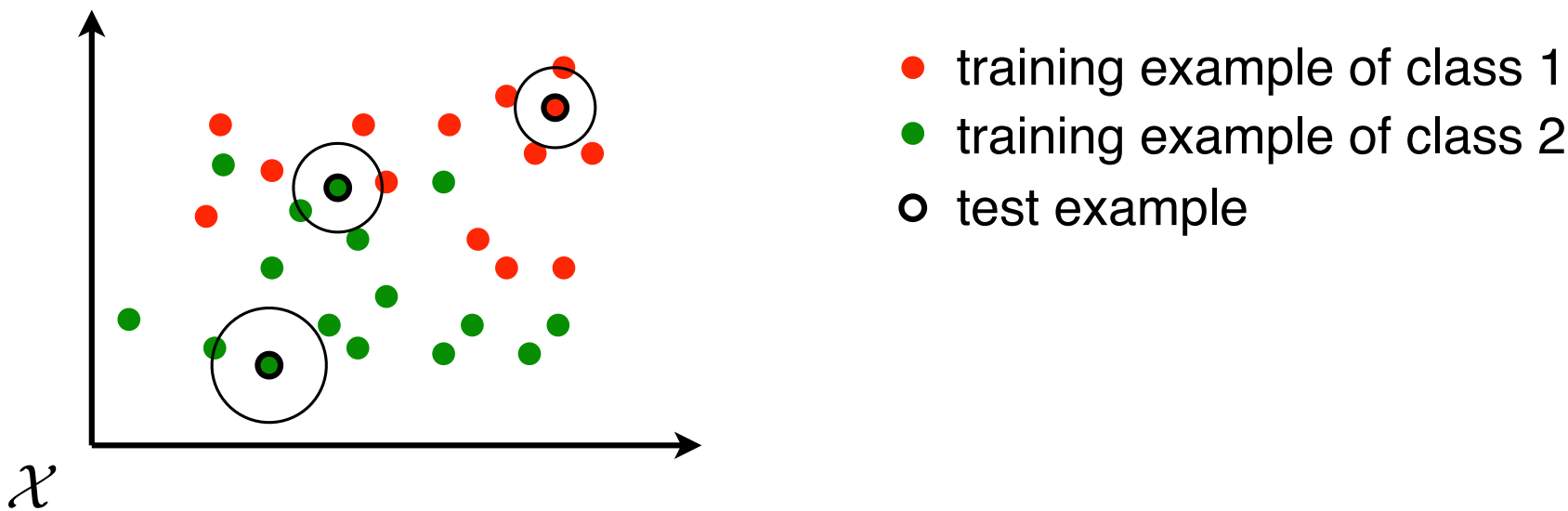
  If there are several matches, use a 'majority voting' scheme.

- Problem: *No generalization!* In many (most) applications, the set $\mathcal{X}$ is very large and there are essentially *never* any perfect matches. So the method always returns "don't know".

Nearest Neighbor classifier

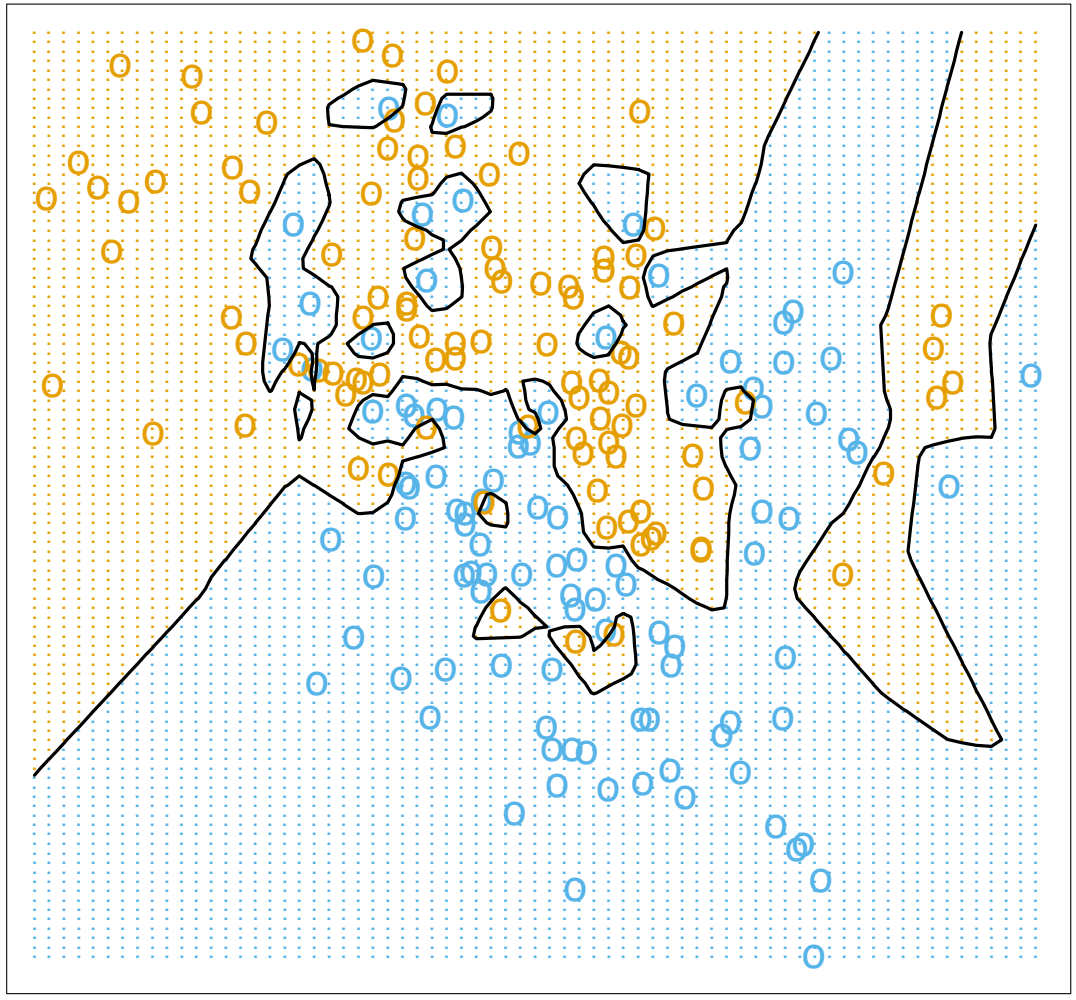
Adapt the above 'simple memory' as follows:

- Instead of saying "don't know" for new unseen objects $\mathbf{x}'$, find the closest match $\mathbf{x}_i$ to $\mathbf{x}'$, and return the corresponding y_i
- Note: 'closest' means here the highest similarity, or equivalently the lowest dissimilarity. The specific choice of proximity measure will of course influence the results!



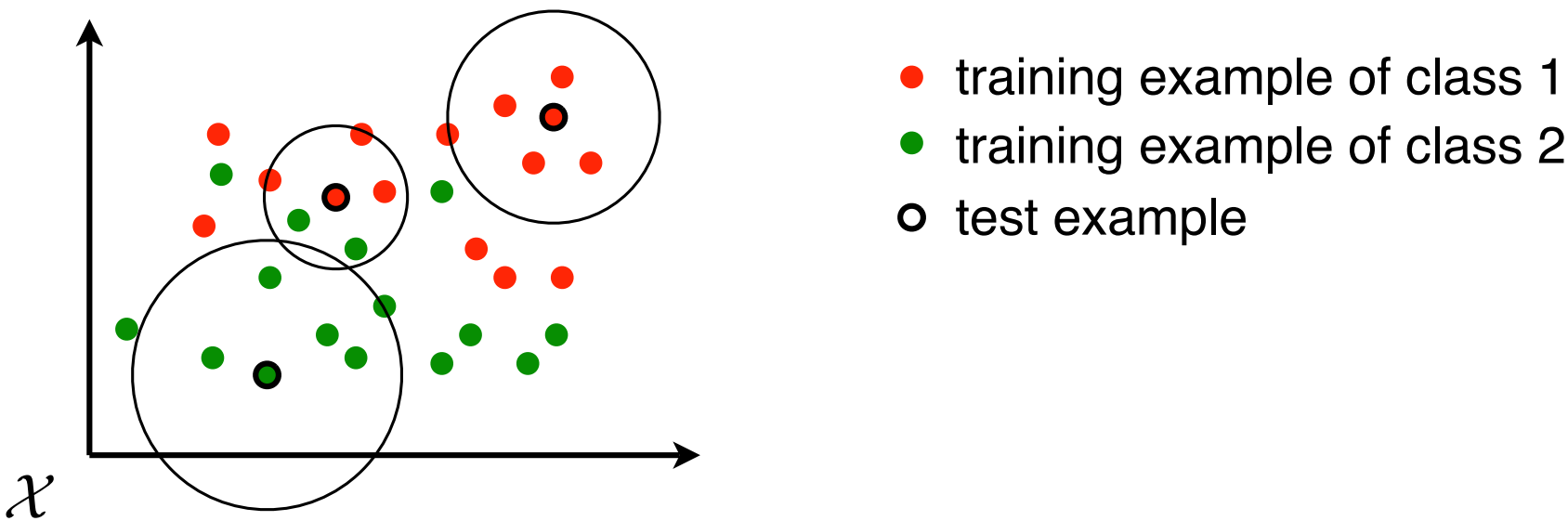
Nearest Neighbor classifier: Decision boundary

- Decision boundary for nearest neighbor classifier (figure from Hastie et al, 2009: 'The elements of statistical learning')



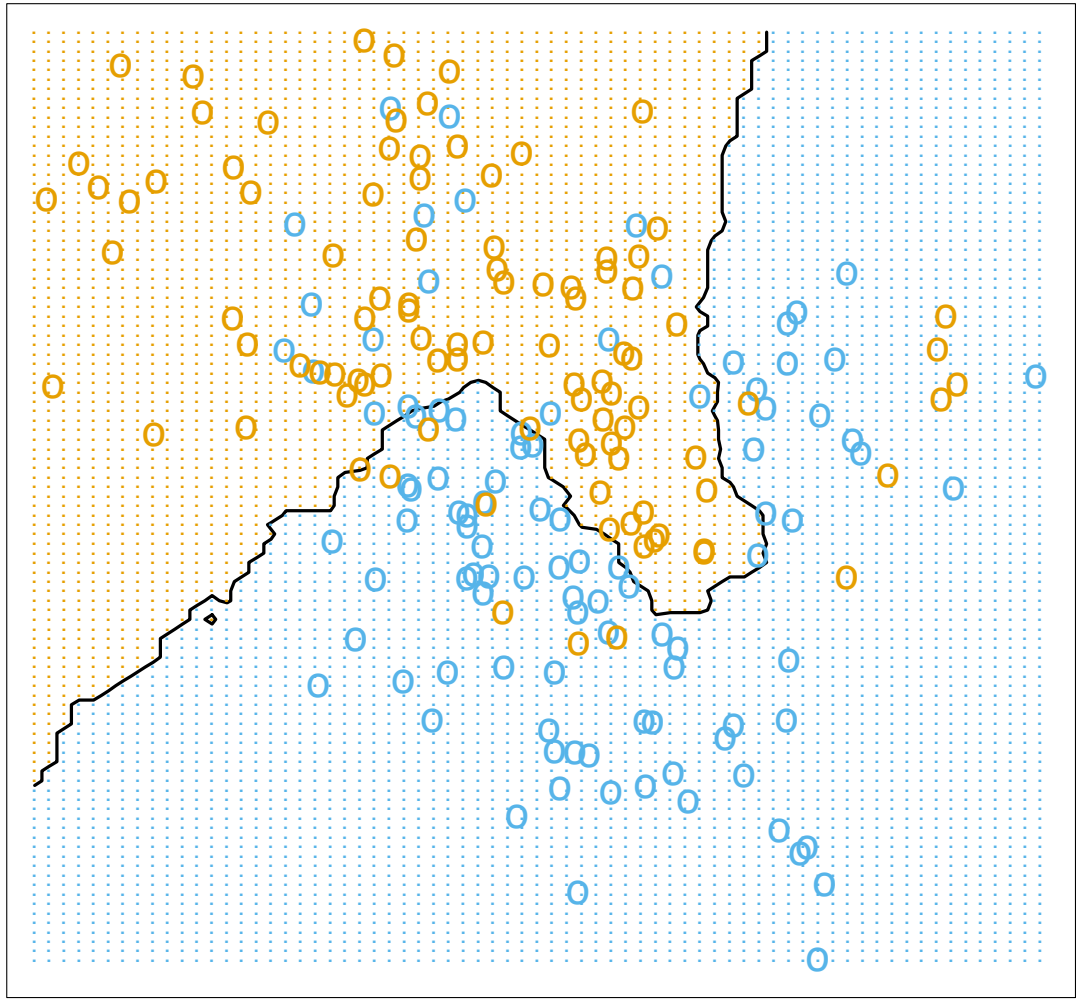
k-Nearest Neighbor classifier (kNN)

- Select the k nearest neighbors (instead of just the single nearest data point).
  - 'Forced choice': Set $y' =$ mode of values of neighbors y_i.
  - Output "don't know" when no clear winner?
  - Or output the empirical class distribution of neighbors...



k-Nearest Neighbor classifier: Decision boundary

- Decision boundary for k Nearest Neighbor classifier, $k = 15$ (figure from Hastie et al, 2009)



What is a good value for k?

- $k = 1$ produces a very jagged decision boundary (capturing small-scale structure, but depends strongly on the specific training examples used)
- $k = 15$ produces a much smoother boundary (capturing only larger-scale structure, and may miss local small-scale structure, but is more robust with respect to the set of training examples)
- We need additional tools and assumptions to say much about the theoretical behavior of the classifier as a function of k. (So far this has all been pretty ad-hoc/intuitive)

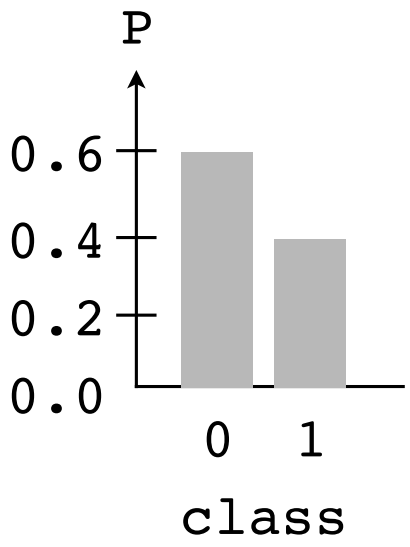
Computational complexity of kNN

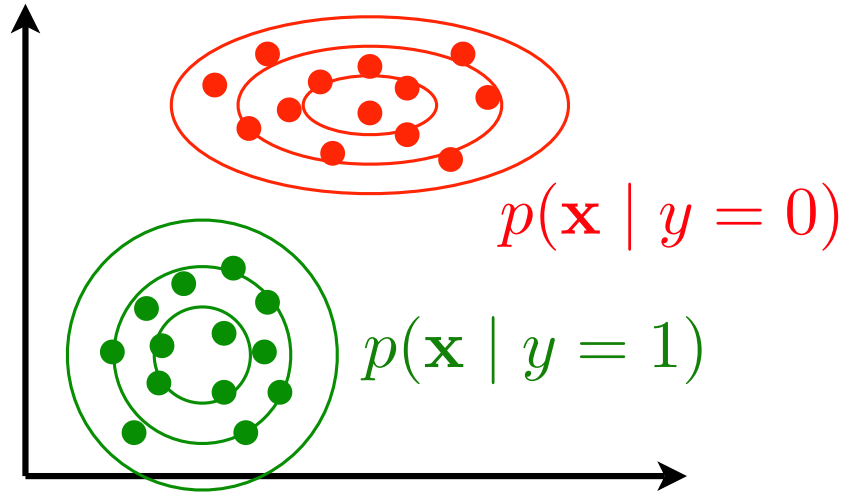
- 'Training phase': Just reading the data into memory, an $O(N)$ operation
- 'Online' (in operation): For each new point to be classified, a straightforward implementation requires computing the proximity to *all* the training examples, i.e. $O(N)$. This can be too expensive in many practical situations (as real-time operation is often required)
- Efficient indexing techniques may be available that reduce the number of proximity computations that need to be performed

Standard probabilistic assumptions for classification

Two assumptions:

- All training records $(\mathbf{x}_i, y_i)$ are drawn independently, identically distributed (i.i.d.) from some joint distribution $P(\mathbf{x}, y)$ (represented by a density as needed for continuous-valued $\mathbf{x}$)
- The test records $(\mathbf{x}', y')$ are also drawn i.i.d. from the *same* distribution





Is this a reasonable assumption?

- Handwritten digit recognition?
- Designing a spam filter?
- News stories (classify into foreign, domestic, economic, sports)
- Classifying web pages into porn vs non-porn?
- ...

Note: By randomly permuting the data records, we can remove temporal/sequential structure. However, in many cases it can be crucial to model this structure.

Bayesian classifier

- If $P(y)$ and $P(\mathbf{x} \mid y)$ were known, we also know

$$P(\mathbf{x}, y) = P(y)P(\mathbf{x} \mid y) \tag{7}$$

from which we get

$$P(y \mid \mathbf{x}) = \frac{P(\mathbf{x}, y)}{P(\mathbf{x})} = \frac{P(y)P(\mathbf{x} \mid y)}{P(\mathbf{x})} \tag{8}$$

which is known as Bayes' rule. Note that the denominator is a constant (so when maximizing the left-hand-side with respect to y it can safely be ignored).

- The latter equation gives us the *true* probabilistic predictions (but this of course requires knowing $P(y)$ and $P(\mathbf{x} \mid y)$)...

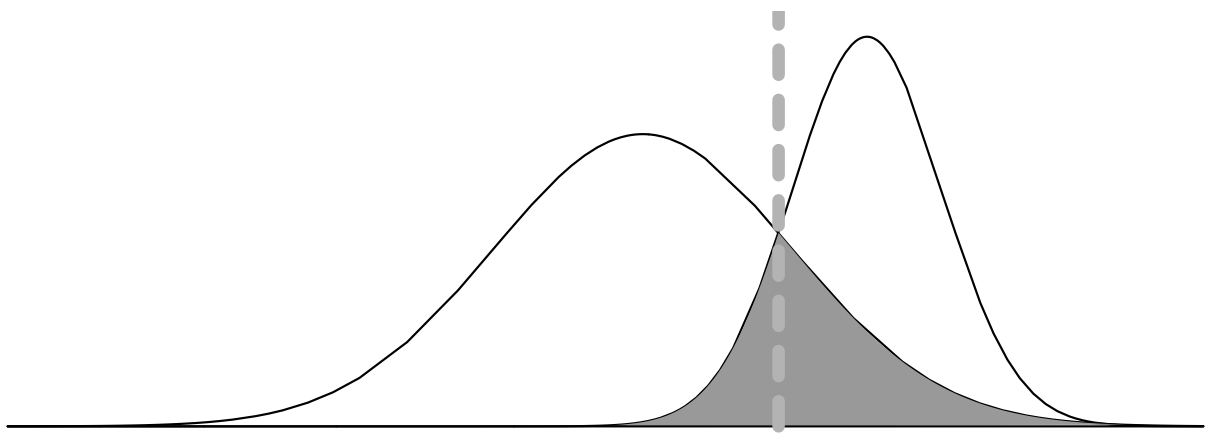
- ...from which we can derive optimal 'forced-choice' classifications (or optimal classifications with "don't know" option)

Bayesian classifier: Example

- For simplicity, say $\mathbf{x}$ is one-dimensional (so write it as x), and $p(x \mid y)$ is Gaussian, for both values of y. Say we know $P(y)$, and we are maximizing standard accuracy. We can explicitly compute the optimal decision boundary: It is given by the point(s!), i.e. value(s) of x, for which

  $$p(x \mid y = 0)P(y = 0) = p(x \mid y = 1)P(y = 1).$$

- The *Bayes error* (='unavoidable error') is the error rate of this optimal classifier.



Say $P(y = 0) = 1/2$, and $P(y = 1) = 1/2$.

Additionally assume $p(x \mid y = 0) = \mathcal{N}(x; \mu_0, \sigma^2)$ and $p(x \mid y = 1) = \mathcal{N}(x; \mu_1, \sigma^2)$, where

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{9}$$

The maximum accuracy (probability of correct classification) is given by using the decision boundary where

$$\begin{aligned}
p(x_t \mid y = 0)P(y = 0) &= p(x_t \mid y = 1)P(y = 1) \\
\exp\left(-\frac{(x_t - \mu_0)^2}{2\sigma^2}\right) &= \exp\left(-\frac{(x_t - \mu_1)^2}{2\sigma^2}\right) \\
(x_t - \mu_0)^2 &= (x_t - \mu_1)^2 \\
x_t^2 - 2x_t\mu_0 + \mu_0^2 &= x_t^2 - 2x_t\mu_1 + \mu_1^2 \\
2(\mu_1 - \mu_0)x_t &= \mu_1^2 - \mu_0^2 \\
x_t &= \frac{(\mu_1 - \mu_0)(\mu_1 + \mu_0)}{2(\mu_1 - \mu_0)} = \frac{\mu_1 + \mu_0}{2}
\end{aligned}$$

The Bayes error rate in this case is equal to

$$\frac{1}{2}\int_{-\infty}^{x_t}\mathcal{N}(x;\mu_1,\sigma^2)+\frac{1}{2}\int_{x_t}^{\infty}\mathcal{N}(x;\mu_0,\sigma^2)$$

$$=\int_{x_t}^{\infty}\mathcal{N}(x;\mu_0,\sigma^2)$$

When the variances are the same but the class probabilities are unequal, i.e. $P(y=0)\neq P(y=1)$ the optimal threshold (again, just a single threshold) is biased away from $(\mu_0+\mu_1)/2$.

If the variances differ then in general there will be *two* thresholds!

Estimating class prevalences $P(y)$

- How to estimate $P(y)$?
  - Simplest estimate: For each class, just divide the empirical frequency by the total number of examples:

$$P(y = \alpha) = \frac{\#[y = \alpha]}{N} \tag{10}$$

    For example: The proportion of females in a computer science class at our department is estimated by the number of females in this class divided by the size of the class. (This is an unbiased estimate if the students taking the course is a random sample of the students at the department... this may or may not be true.)

  - In the i.i.d. setting it is consistent, and it works reasonably well when there are a sufficient number of examples of all classes

Estimating class-conditional distributions $P(\mathbf{x} \mid y)$

- How to estimate $P(\mathbf{x} \mid y)$ for each value of y?
  - For high-dimensional $\mathbf{x}$ this is an extremely non-trivial problem
  - Discrete $\mathbf{x}$:

    Dividing empirical frequencies by total count for every element of $\mathcal{X}$ does not work in practice in most cases (too many states, too few data points $\Rightarrow$ unreliable, many zeros)
  - Continuous-valued $\mathbf{x}$:

    Multivariate normal (Gaussian) distribution (n dimensions):

$$\mathcal{N}(\mathbf{x}; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right)$$

    'Only' requires estimates of the mean μ and the covariance matrix Σ, so the total number of parameters is $n + n(n+1)/2$. For low-dimensional data this may be ok, but even this is unreliable for high-dimensional data.

Naïve Bayes classifier

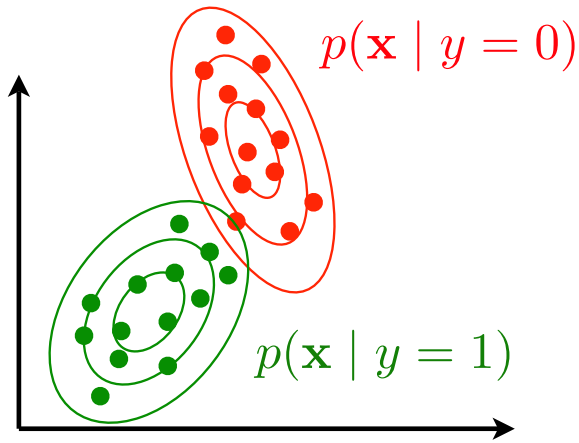
- The central idea of the Naïve Bayes classifier is to assume that the class-conditional distributions factorize, i.e. that for each class, the joint distribution over examples is given by the product of distributions in each dimension:

$$P(\mathbf{x} \mid y = y_0) = \prod_{i=1}^{n} P(x_i \mid y = y_0) \tag{11}$$

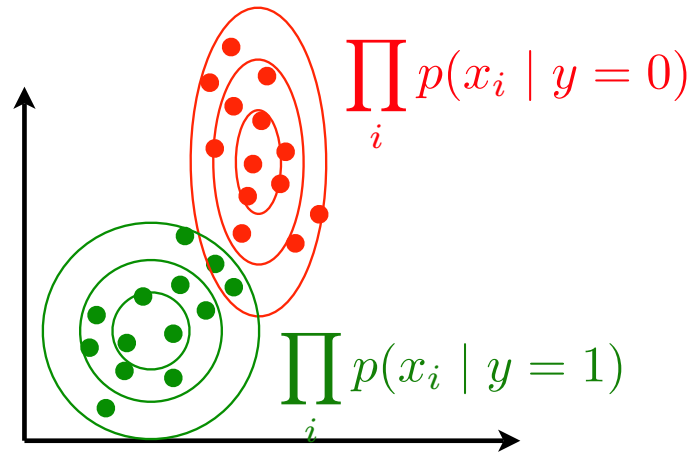
- In other words, one estimates the distributions *of each class-attribute pair separately* and uses these to approximate the class-conditional distributions in the joint space.

Naïve Bayes classifier – example 1

- Gaussian class-conditional distributions:
  - Instead of $n + n(n+1)/2$ parameters for each class-conditional distribution, we only use $n + n$ (mean and variance for each dimension)
  - Fit to data is only approximate, especially when there are strong correlations between attributes within one or more classes



true class-conditional distributions



Naive Bayes approximation (note: axis-aligned!)

Naïve Bayes classifier – example 2

- Discrete attributes (n attributes, each with L possible values):
  - Instead of $L^n - 1$ parameters for each class-conditional distribution (for the joint distribution), we only need $(L-1)n$ parameters (for the n marginal distributions each over L values)

$p(\mathbf{x} \mid y = 1)$

x_1 \ x_2	0	1
0	0.4	0.0
1	0.2	0.4

$\prod_i p(x_i \mid y = 1)$

x_1 \ x_2	0	1	
0	0.24	0.16	0.4
1	0.36	0.24	0.6
	0.6	0.4	

$p(\mathbf{x} \mid y = 0)$

x_1 \ x_2	0	1
0	0.1	0.5
1	0.3	0.1

true class-conditional distributions

$\prod_i p(x_i \mid y = 0)$

x_1 \ x_2	0	1	
0	0.24	0.36	0.6
1	0.16	0.24	0.4
	0.4	0.6	

Naive Bayes approximation

Naïve Bayes – fitting univariate class conditionals

- Continuous variables: densities (see Appendix C)
  - Gaussian
  - Exponential
  - Student's t
  - Histogram (i.e. discretizing)
  - Kernel density estimation
  - ...
- How to estimate the parameters?
  - Moment matching
  - Maximum likelihood
  - ...

Naïve Bayes – fitting univariate class conditionals

- Discrete variables with finite state space:
  - We already gave a simple estimator when discussing estimating $P(y)$: just normalizing the counts. In this case, for a given class $y = y_0$, and a given attribute x_i we would normalize the counts by

$$P(x_i = \alpha \mid y = y_0) = \frac{\#[x_i = \alpha \text{ and } y = y_0]}{\#[y = y_0]} \tag{12}$$

  - For some class-attribute pairs, some of the counts may be zero $\Rightarrow$ some of the class-conditional probabilities are zero!
    $\Rightarrow$ for some (new) data points $\mathbf{x}$, we have $\forall y : P(\mathbf{x} \mid y) = 0$, so we cannot compute $P(y \mid \mathbf{x})$ (it is undefined!)

- A simple solution: Add a constant c (for example $c = 1$) to the numerator, and add Lc (where L is the number of distinct values of x_i) to the denominator $\Rightarrow$ no more zeros, probabilities still sum to one, and can even be justified on theoretical grounds

- Example (spam filtering):

	'viagra'	'millions'	'grade'	'confidential'	
msg #1	0	1	1	0	...
msg #2	0	0	1	1	...
msg #3	0	0	1	0	...
msg #4	0	1	0	1	...
msg #5	0	0	0	0	...
spam #1	1	1	0	0	...
spam #2	1	0	0	1	...
spam #3	0	1	0	1	...
spam #4	0	0	0	1	...
spam #5	0	1	0	0	...

Simply normalizing the frequencies:

$$P(\text{msg}) = 5/10 = 0.5$$
$$P(\text{spam}) = 5/10 = 0.5$$

$$P(\text{viagra} \mid \text{msg}) = 0/5 = 0$$
$$P(\text{millions} \mid \text{msg}) = 2/5 = 0.4$$
$$P(\text{grade} \mid \text{msg}) = 3/5 = 0.6$$
$$P(\text{confidential} \mid \text{msg}) = 2/5 = 0.4$$

$$P(\text{viagra} \mid \text{spam}) = 2/5 = 0.4$$
$$P(\text{millions} \mid \text{spam}) = 3/5 = 0.6$$
$$P(\text{grade} \mid \text{spam}) = 0/5 = 0$$
$$P(\text{confidential} \mid \text{spam}) = 3/5 = 0.6$$

Compute: $P(\text{spam} \mid \text{“confidential: how to win millions”})$ (next slide)

Problem: A new message “premium grade viagra available now!!!” would get 0 probability under both classes!

email = "confidential: how to win millions"

$$
\begin{aligned}
P(\text{spam} \mid \text{email}) &= P(\text{email} \mid \text{spam})P(\text{spam})/P(\text{email}) \\
&= P(\neg\text{viagra} \mid \text{spam})P(\text{millions} \mid \text{spam})P(\neg\text{grade} \mid \text{spam}) \\
&\quad P(\text{confidential} \mid \text{spam})P(\text{spam})/P(\text{email}) \\
&= 0.6 * 0.6 * 1.0 * 0.6 * 0.5/P(\text{email}) = 0.108/P(\text{email}) \\
P(\text{msg} \mid \text{email}) &= P(\text{email} \mid \text{msg})P(\text{msg})/P(\text{email}) \\
&= P(\neg\text{viagra} \mid \text{msg})P(\text{millions} \mid \text{msg})P(\neg\text{grade} \mid \text{msg}) \\
&\quad P(\text{confidential} \mid \text{msg})P(\text{msg})/P(\text{email}) \\
&= 1.0 * 0.4 * 0.4 * 0.4 * 0.5/P(\text{email}) = 0.032/P(\text{email}) \\
P(\text{email}) &= 0.108 + 0.032 = 0.140 \\
P(\text{spam} \mid \text{email}) &= 0.108/0.140 \approx 0.77 \\
P(\text{msg} \mid \text{email}) &= 0.032/0.140 \approx 0.23
\end{aligned}
$$

- Example (spam filtering):

	'viagra'	'millions'	'grade'	'confidential'	
msg #1	0	1	1	0	...
msg #2	0	0	1	1	...
msg #3	0	0	1	0	...
msg #4	0	1	0	1	...
msg #5	0	0	0	0	...
spam #1	1	1	0	0	...
spam #2	1	0	0	1	...
spam #3	0	1	0	1	...
spam #4	0	0	0	1	...
spam #5	0	1	0	0	...

Instead, let's add $c = 1$ to all counts:

$$
\begin{aligned}
P(\text{msg}) &= (5+1)/12 = 0.5 \\
P(\text{spam}) &= (5+1)/12 = 0.5 \\
P(\text{viagra} \mid \text{msg}) &= (0+1)/7 = 1/7 \\
P(\text{millions} \mid \text{msg}) &= (2+1)/7 = 3/7 \\
P(\text{grade} \mid \text{msg}) &= (3+1)/7 = 4/7 \\
P(\text{confidential} \mid \text{msg}) &= (2+1)/7 = 3/7 \\
P(\text{viagra} \mid \text{spam}) &= (2+1)/7 = 3/7 \\
P(\text{millions} \mid \text{spam}) &= (3+1)/7 = 4/7 \\
P(\text{grade} \mid \text{spam}) &= (0+1)/7 = 1/7 \\
P(\text{confidential} \mid \text{spam}) &= (3+1)/7 = 4/7
\end{aligned}
$$

Note: Adding 2 to the denominator because each variable is binary (has 2 states). Now all probabilities are > 0.

Practical issues with Naïve Bayes

- In high dimensions, the probabilities/densities $P(\mathbf{x} \mid y_j)$ tend to become very close to zero for all classes y_j. The same applies to the marginals $P(\mathbf{x})$.

$\Rightarrow$ Numerical problems in comparing the $P(\mathbf{x} \mid y_j)$ with each other, and in computing $P(\mathbf{x} \mid y_j)P(y_j)/P(\mathbf{x})$ (floating point underflow, $0/0$ problems)

- Solution: Use logarithms and add a suitable constant before exponentiating, as follows
  1. $L(y_j \mid \mathbf{x}) = \log(P(\mathbf{x} \mid y_j)P(y_j)) = \sum_i \log(P(x_i|y_j)) + \log(P(y_j))$
  2. $L(y_j \mid \mathbf{x}) = L(y_j \mid \mathbf{x}) - \max_j(L(y_j \mid \mathbf{x}))$
  3. $P_t(y_j \mid \mathbf{x}) = \exp(L(y_j \mid \mathbf{x}))$
  4. $P(y_j \mid \mathbf{x}) = P_t(y_j \mid \mathbf{x}) / \sum_j P_t(y_j \mid \mathbf{x})$

Generative vs discriminative

- Naïve Bayes is a representative of what we call a *generative* approach to classification...
  - From the model, it is actually possible to synthesize (generate) new examples: First draw $y \sim P(y)$ then draw $\mathbf{x} \sim P(\mathbf{x} \mid y)$
- ...as opposed to a *discriminative* approach, which only models $P(y \mid \mathbf{x})$ or even more simply just the function $\hat{y} = f(\mathbf{x})$
- Note: The generative approach is, in a sense, solving a more difficult problem than is asked for.

Model complexity, overfitting, and model selection

So, how good is my classifier?

- Apply the learned classifier to the training data?
  - k-Nearest Neighbor with $k = 1$ performs at 100% accuracy (each point is the nearest neighbor of itself, so label is correct!)
  - k-Nearest Neighbor with $k > 1$ generally performs $< 100\%$
  - Naïve Bayes also generally gives $< 100\%$ accuracy
  - Many other classifiers (such as decision trees or rule-based classifiers, coming soon!) can achieve 100% accuracy as well, as long as all records $\mathbf{x}$ are distinct

  $\Rightarrow$ so never use kNN with $k > 1$ or Naive Bayes??!?

- But... the goal of classification is to perform well on *new* (unseen) data. How can we test that?

- If the i.i.d. model for the classification problem is correct, then...

...we can estimate the *generalization error* (expected error when samples are drawn from the underlying distribution) by *using only part of the available data for 'training' and leaving the rest for 'testing'*.

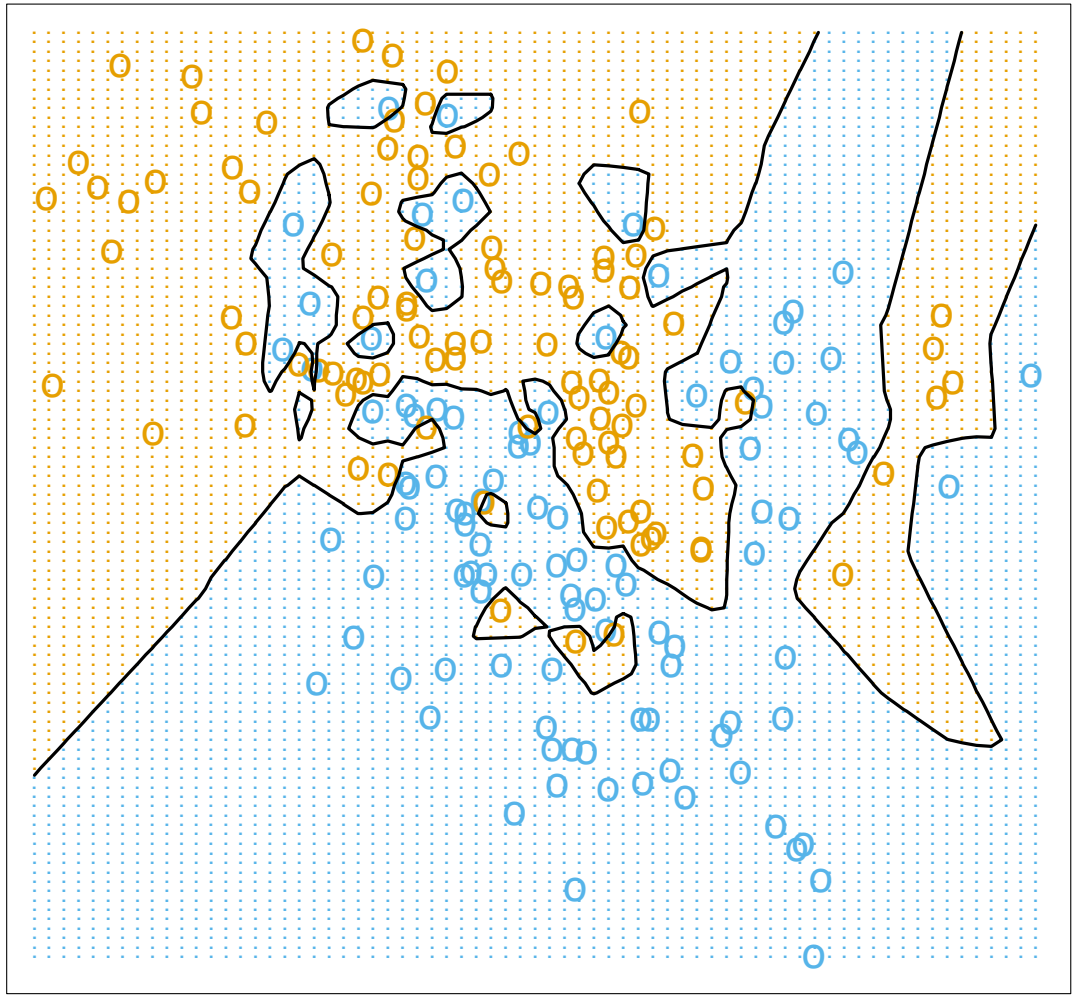
(under the stated assumptions the test data is now 'new data', so we can with this approach get unbiased estimates of the generalization error)

- Typically (almost invariably), *the performance on the test set is worse than on the training set*, because the classifier has learned some features specific to the training set (in addition to features of the underlying distribution)

- Comparing 1NN, kNN, Naive Bayes, and any other classifier on a held-out test set:
  - Now all generally perform $< 100\%$
  - Not a priori clear which method is best, this is an *empirical* issue (and depends on the amount of data, the structure in the data, etc)
- Flexible classifiers may *overfit* the training data.
  - Spam filter: Picking up on words that discriminate the two classes in the training set but not in the distribution
  - Credit card fraud detection: Some thieves may use the card to buy regular items; those particular items (or item combinations) may then be marked suspicious
  - Face recognition: It happened to be darker than average when one of the pictures was taken. Now darkness is associated with that particular identity.

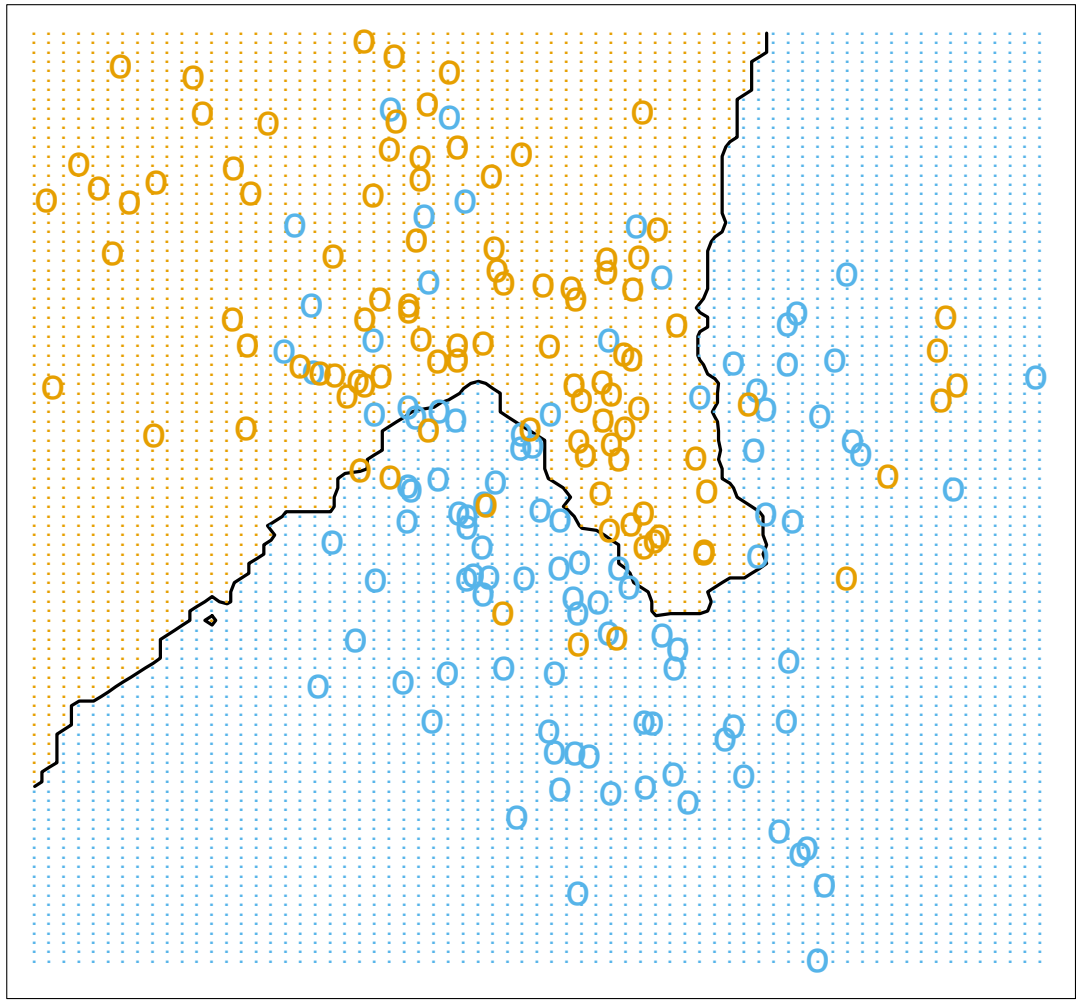
1NN example (recap)

- Decision boundary for k Nearest Neighbor classifier, $k = 1$ (figure from Hastie et al, 2009)



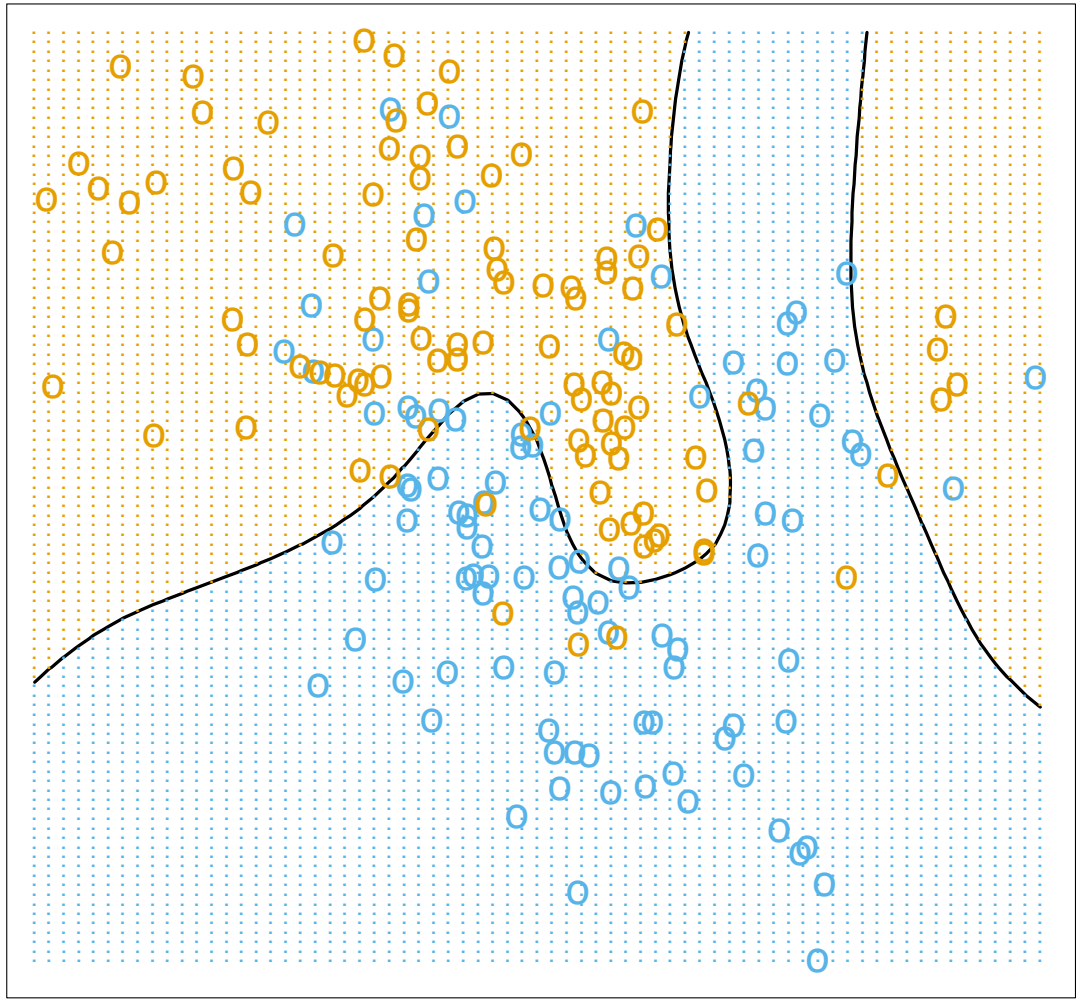
kNN example (recap)

- Decision boundary for k Nearest Neighbor classifier, $k = 15$ (figure from Hastie et al, 2009)



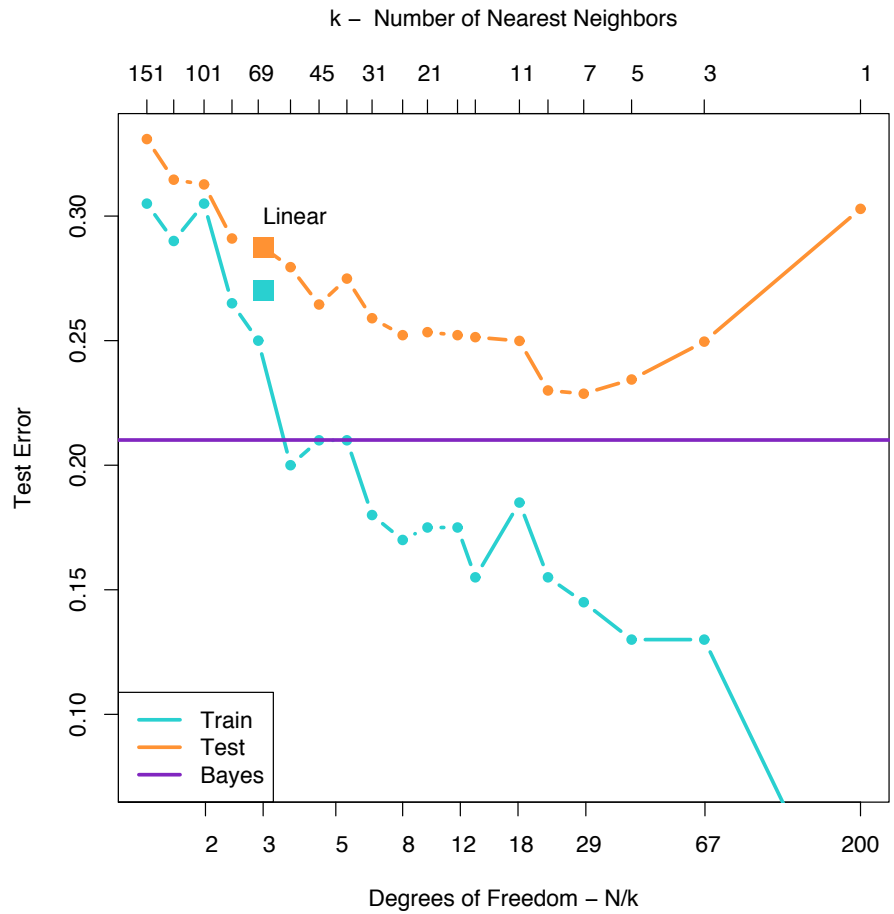
Bayes Optimal Classifier

- Bayes optimal decision boundary (figure from Hastie et al, 2009)



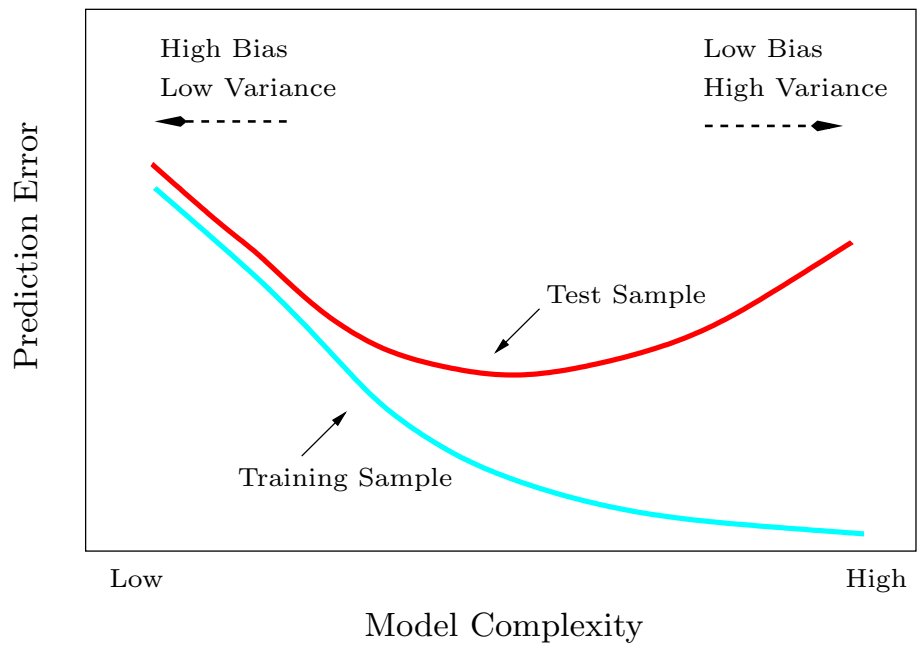
Error vs flexibility (train and test)

- Classification error on training set and test set (training set size: 200 points, test set size: 10,000 points) (figure from Hastie et al, 2009)



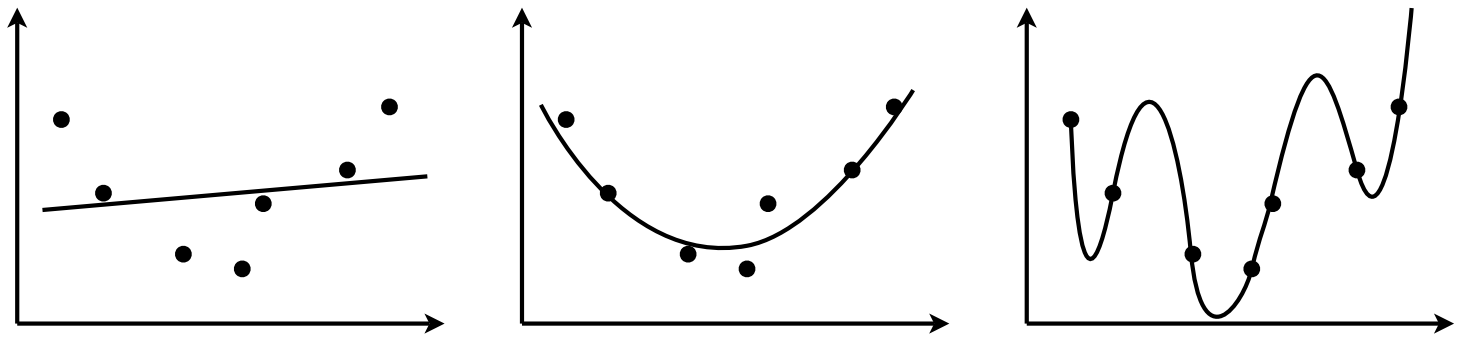
Error vs flexibility (train and test)

- Typical behaviour: The higher the model complexity (more flexible model) the lower the error on the training sample. However, the error curve for a test sample is U-shaped. (figure from Hastie et al, 2009)



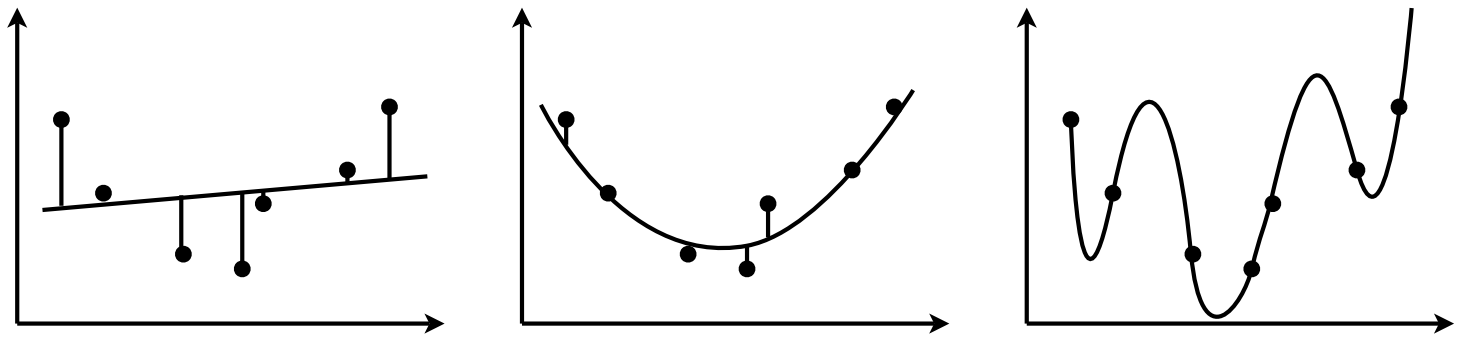
Analogous problem: Curve fitting

- Which of the following curves 'fits best' to the data?



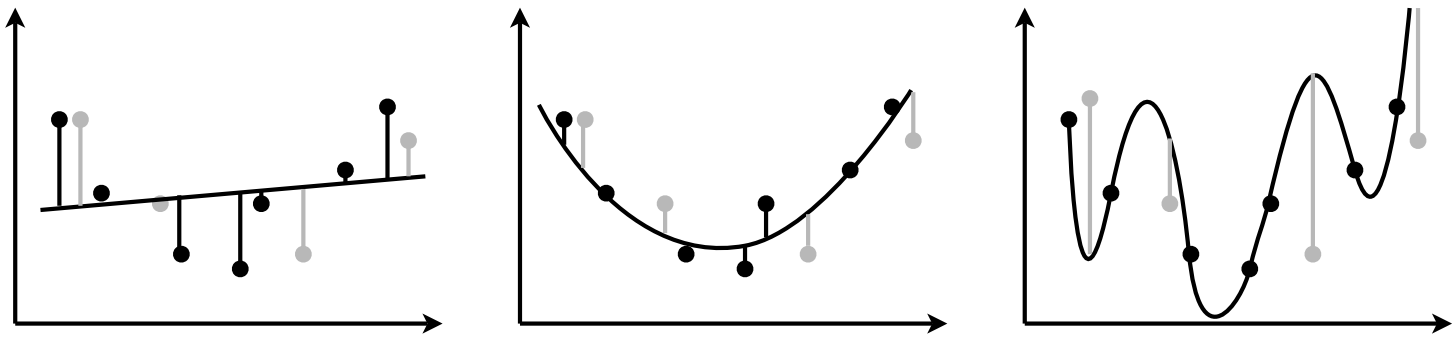
Analogous problem: Curve fitting

- Which of the following curves 'fits best' to the data?



Analogous problem: Curve fitting

- Which of the following curves 'fits best' to the data?



'underfit' 'overfit'

- The more flexible the curve...
  - ...the better you can make it fit your data...
  - ...but the more likely it is to overfit

$\Rightarrow$ ...so you need to be careful to strive for both model simplicity and for good fit to data!

Bias-variance tradeoff

- Based on N training datapoints from the distribution, how close is the learned classifier to the optimal classifier?

  Consider multiple trials: repeatedly and independently drawing N training points from the underlying distribution.

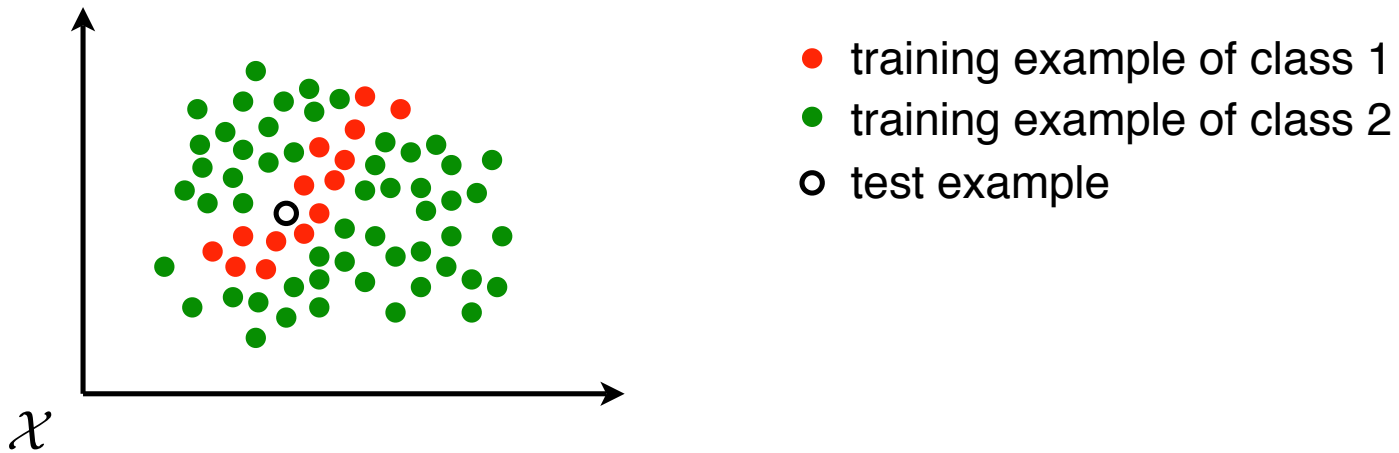
  - *Bias:* Difference between the optimal classifier and the average (over all trials) of the learned classifiers
  - *Variance:* Average squared difference from the (single-trial) learned classifier from the average (over all trials) of the learned classifiers

- Goal: Low bias and low variance.

- High model complexity $\Rightarrow$ low bias and high variance
  Low model complexity $\Rightarrow$ high bias and low variance

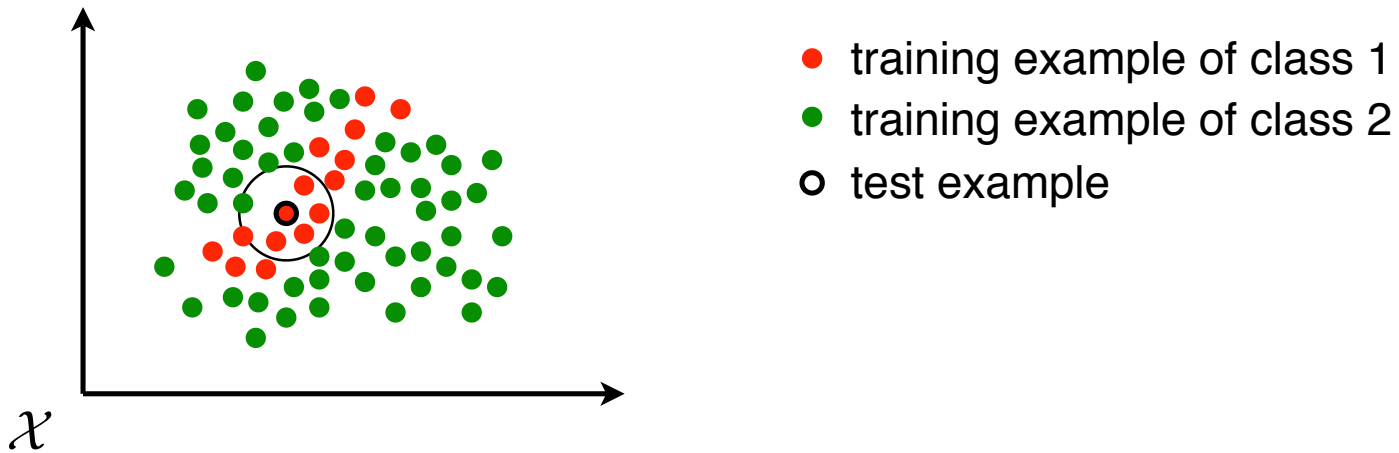
Problems of *kNN* with large *k*

- For a fixed number N datapoints, increasing k smooths out local structure:



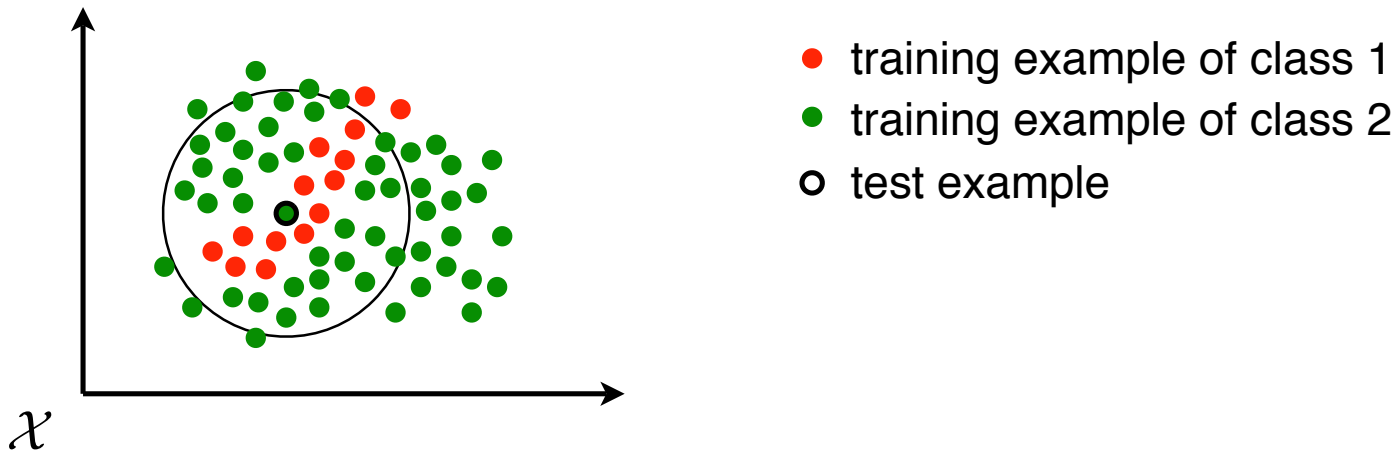
Problems of *kNN* with large *k*

- For a fixed number N datapoints, increasing k smooths out local structure:



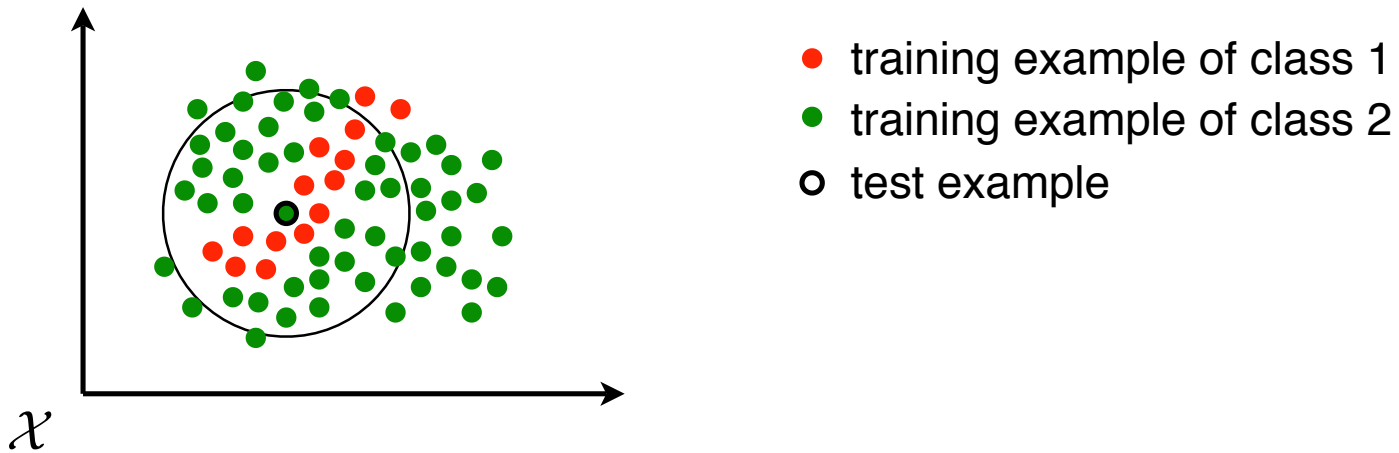
Problems of *kNN* with large *k*

- For a fixed number N datapoints, increasing k smooths out local structure:



Problems of *kNN* with large *k*

- For a fixed number N datapoints, increasing k smooths out local structure:

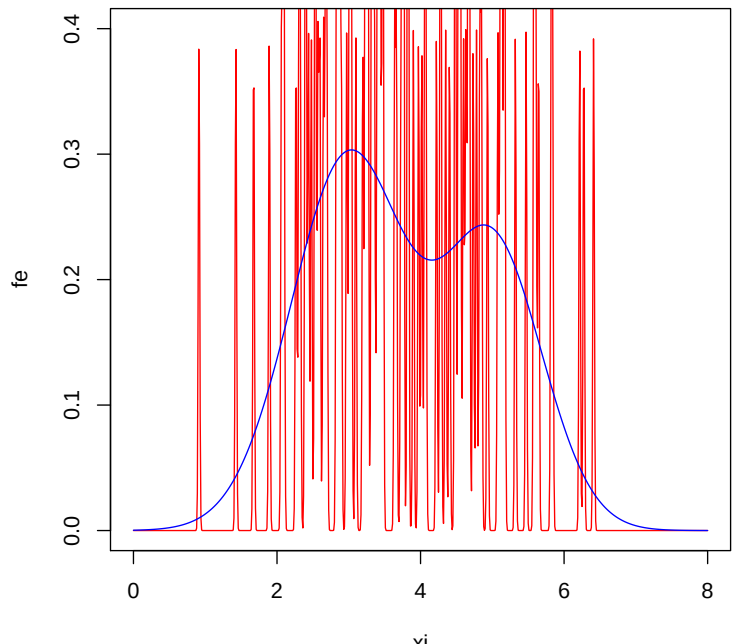


- For finite N, we don't know the best value k...
- (Note: Might even be best to use different number of neighbors in different regions of the space.)

All classifiers have this problem: Naive Bayes

- How to estimate the (univariate) class-conditional marginal distributions? How much smoothing to use?
- E.g. kernel-density estimate, what value to choose for σ?

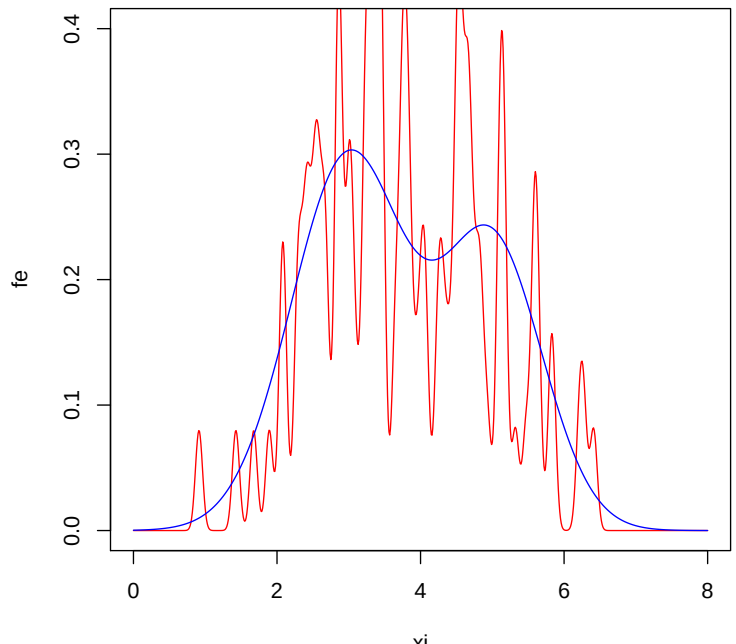
$$p(x_i \mid y) = \frac{1}{N_y} \sum_{j=1}^{N_y} \mathcal{N}\left(x_i;\ x_i^{(j)}, \sigma^2\right) \tag{13}$$



All classifiers have this problem: Naive Bayes

- How to estimate the (univariate) class-conditional marginal distributions? How much smoothing to use?
- E.g. kernel-density estimate, what value to choose for σ?

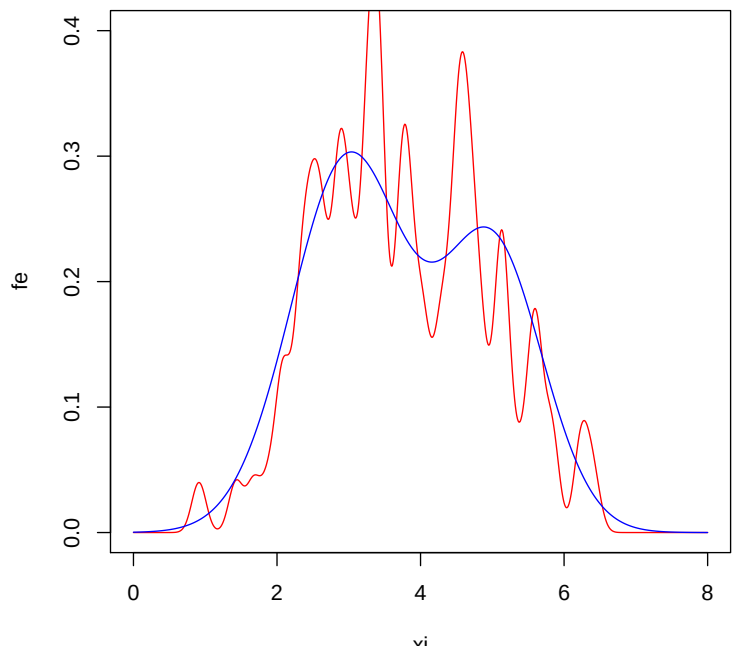
$$p(x_i \mid y) = \frac{1}{N_y} \sum_{j=1}^{N_y} \mathcal{N}\left(x_i;\ x_i^{(j)}, \sigma^2\right) \tag{13}$$



All classifiers have this problem: Naive Bayes

- How to estimate the (univariate) class-conditional marginal distributions? How much smoothing to use?
- E.g. kernel-density estimate, what value to choose for σ?

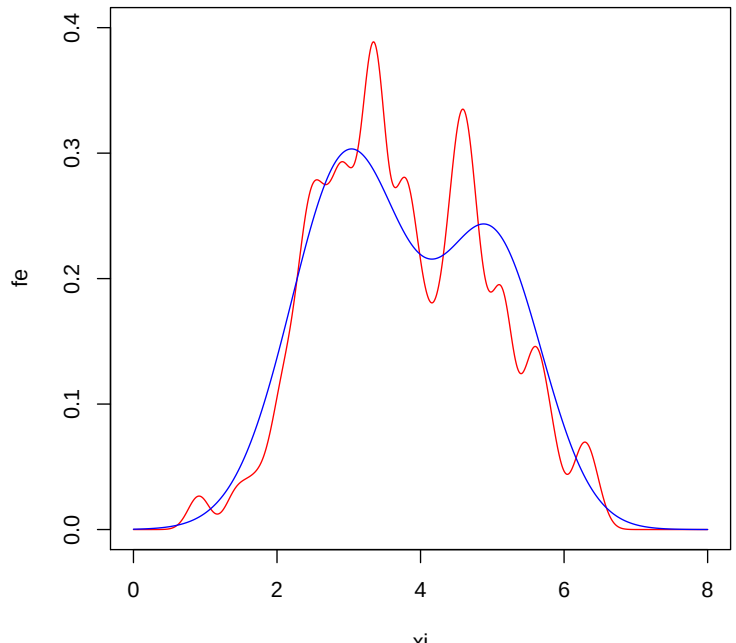
$$p(x_i \mid y) = \frac{1}{N_y} \sum_{j=1}^{N_y} \mathcal{N}\left(x_i;\ x_i^{(j)}, \sigma^2\right) \tag{13}$$



All classifiers have this problem: Naive Bayes

- How to estimate the (univariate) class-conditional marginal distributions? How much smoothing to use?
- E.g. kernel-density estimate, what value to choose for σ?

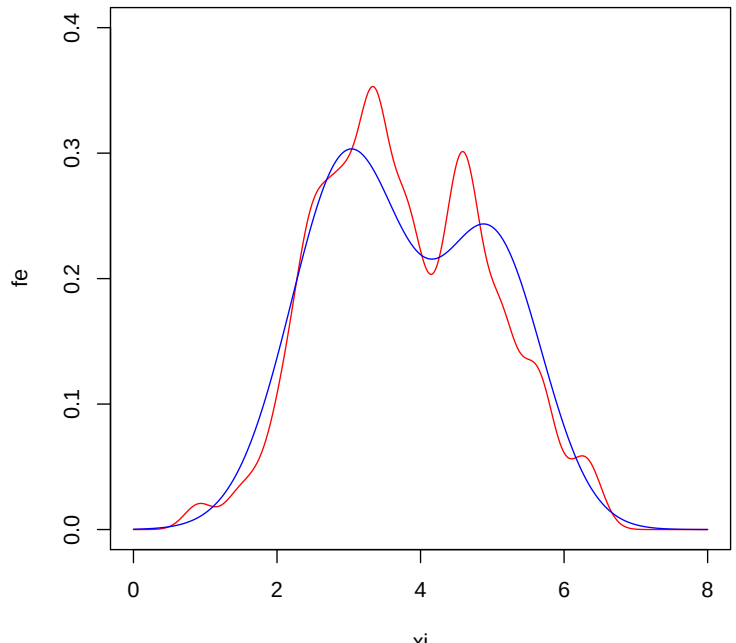
$$p(x_i \mid y) = \frac{1}{N_y} \sum_{j=1}^{N_y} \mathcal{N}\left(x_i;\ x_i^{(j)}, \sigma^2\right) \tag{13}$$



All classifiers have this problem: Naive Bayes

- How to estimate the (univariate) class-conditional marginal distributions? How much smoothing to use?
- E.g. kernel-density estimate, what value to choose for σ?

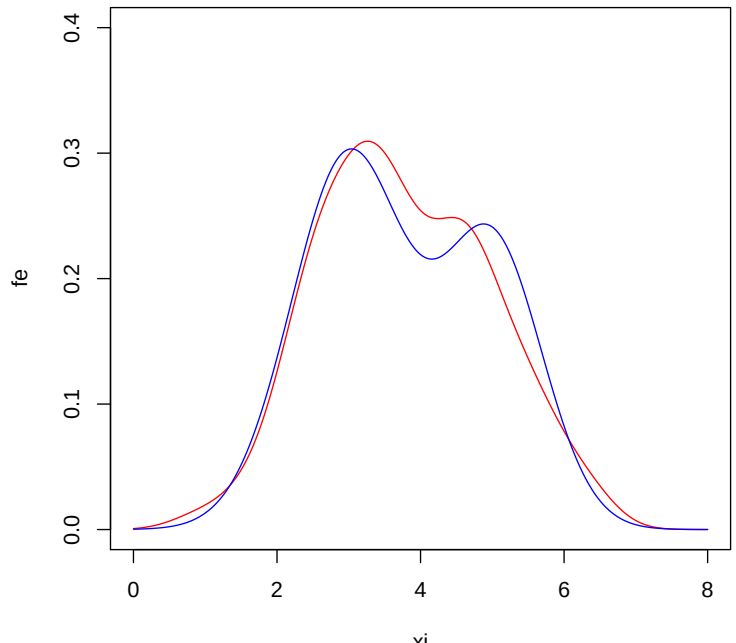
$$p(x_i \mid y) = \frac{1}{N_y} \sum_{j=1}^{N_y} \mathcal{N}\left(x_i;\ x_i^{(j)}, \sigma^2\right) \tag{13}$$



All classifiers have this problem: Naive Bayes

- How to estimate the (univariate) class-conditional marginal distributions? How much smoothing to use?
- E.g. kernel-density estimate, what value to choose for σ?

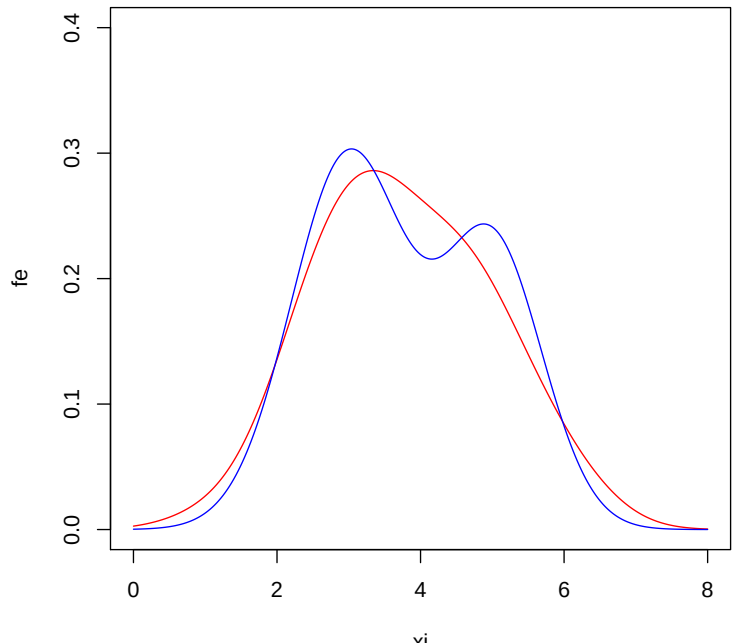
$$p(x_i \mid y) = \frac{1}{N_y} \sum_{j=1}^{N_y} \mathcal{N}\left(x_i;\ x_i^{(j)}, \sigma^2\right) \tag{13}$$



All classifiers have this problem: Naive Bayes

- How to estimate the (univariate) class-conditional marginal distributions? How much smoothing to use?
- E.g. kernel-density estimate, what value to choose for σ?

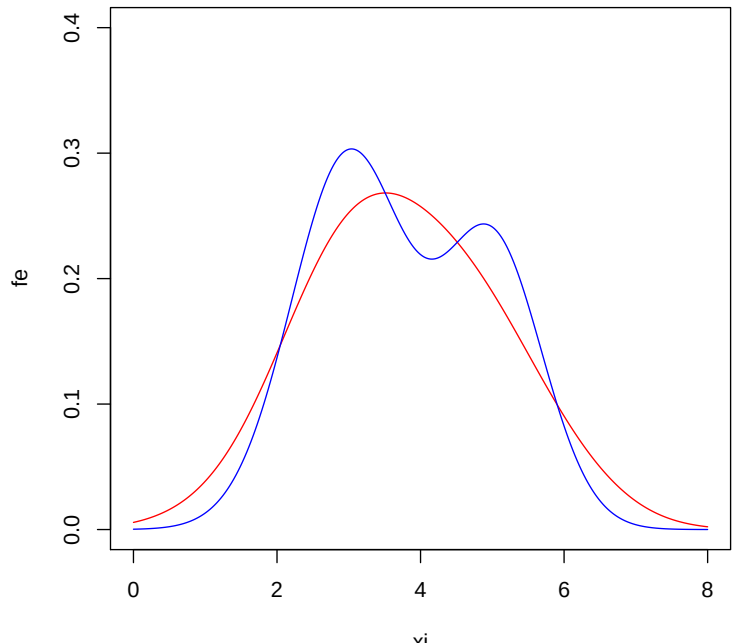
$$p(x_i \mid y) = \frac{1}{N_y} \sum_{j=1}^{N_y} \mathcal{N}\left(x_i;\ x_i^{(j)}, \sigma^2\right) \tag{13}$$



All classifiers have this problem: Naive Bayes

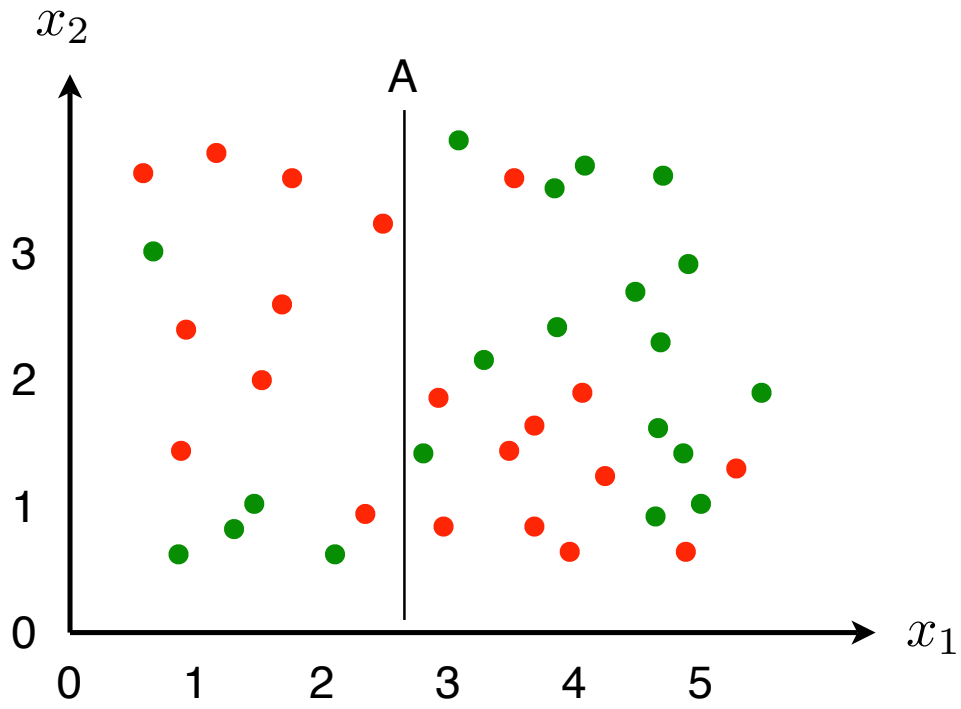
- How to estimate the (univariate) class-conditional marginal distributions? How much smoothing to use?
- E.g. kernel-density estimate, what value to choose for σ?

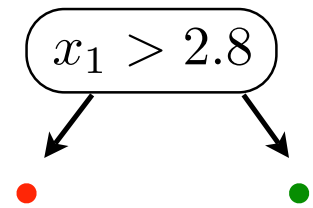
$$p(x_i \mid y) = \frac{1}{N_y} \sum_{j=1}^{N_y} \mathcal{N}\left(x_i;\ x_i^{(j)}, \sigma^2\right) \tag{13}$$



All classifiers have this problem: Decision trees
(Discussed soon in more detail!)

- What is the appropriate depth of the tree?

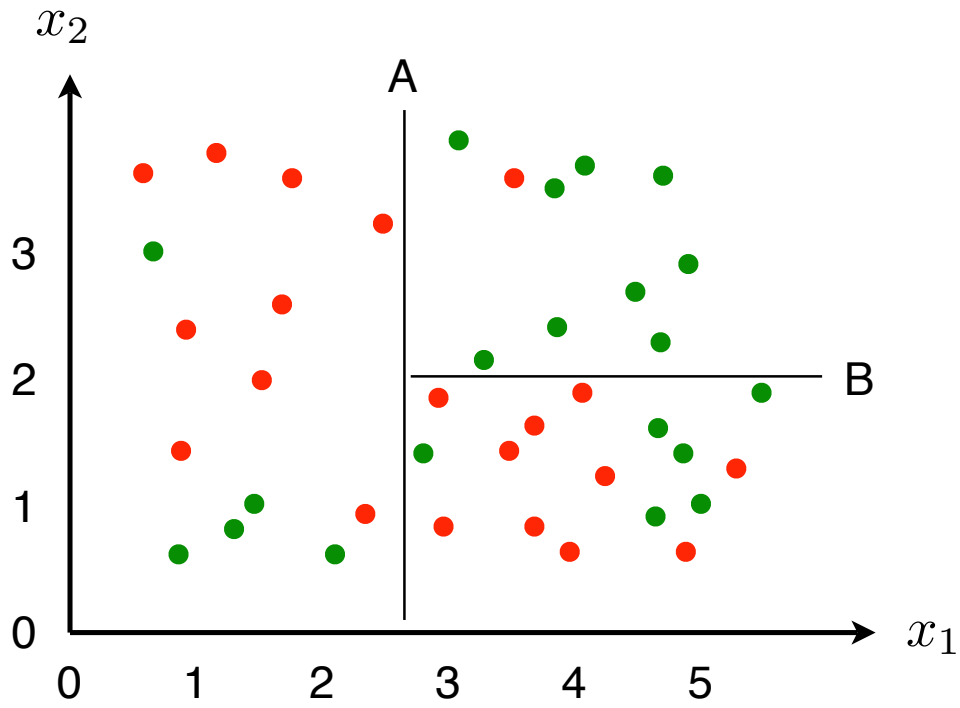


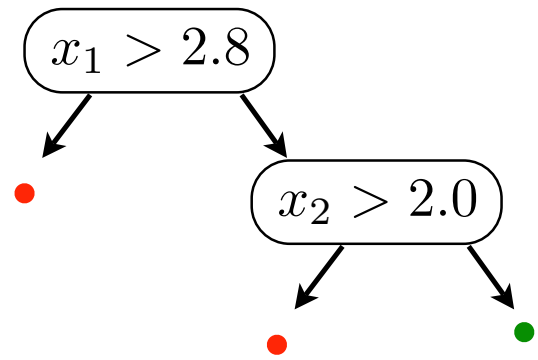


- Use pre-pruning (stop 'early' before adding too many nodes) or post-pruning (after the full tree is constructed, try to improve it by removing nodes)

All classifiers have this problem: Decision trees
(Discussed soon in more detail!)

- What is the appropriate depth of the tree?

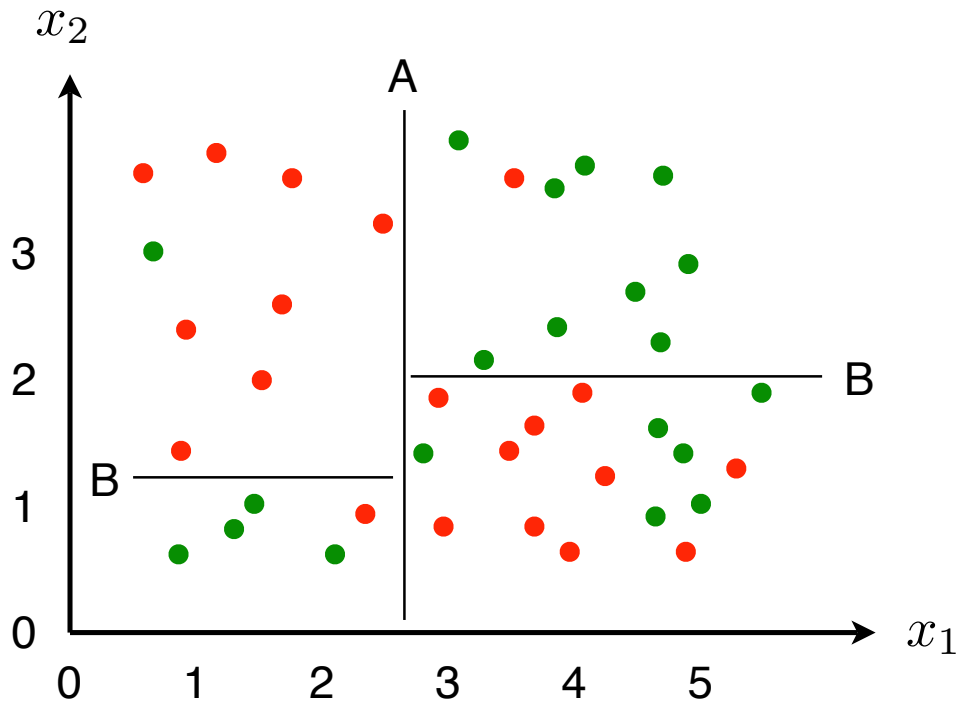


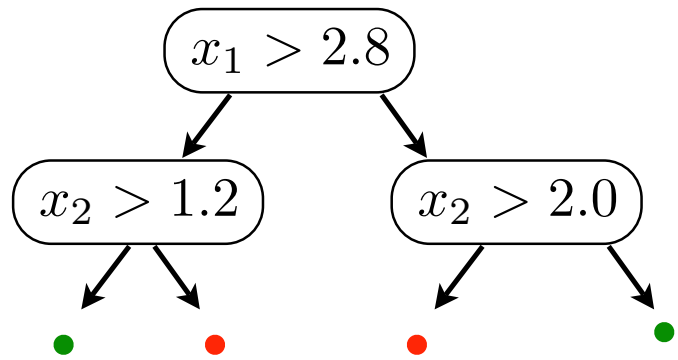


- Use pre-pruning (stop 'early' before adding too many nodes) or post-pruning (after the full tree is constructed, try to improve it by removing nodes)

All classifiers have this problem: Decision trees
(Discussed soon in more detail!)

- What is the appropriate depth of the tree?

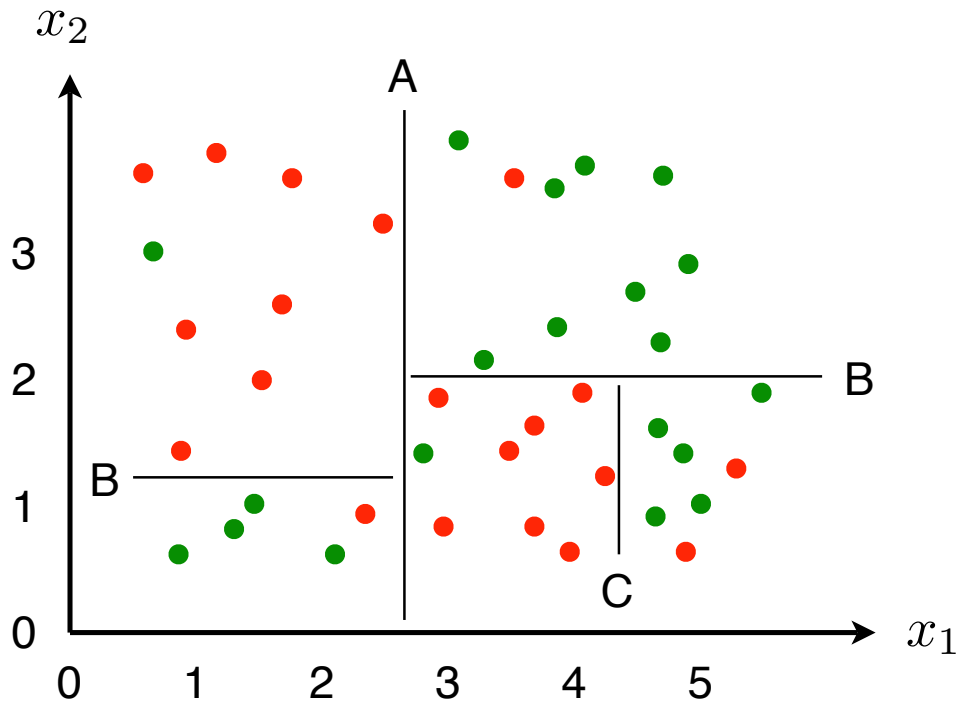


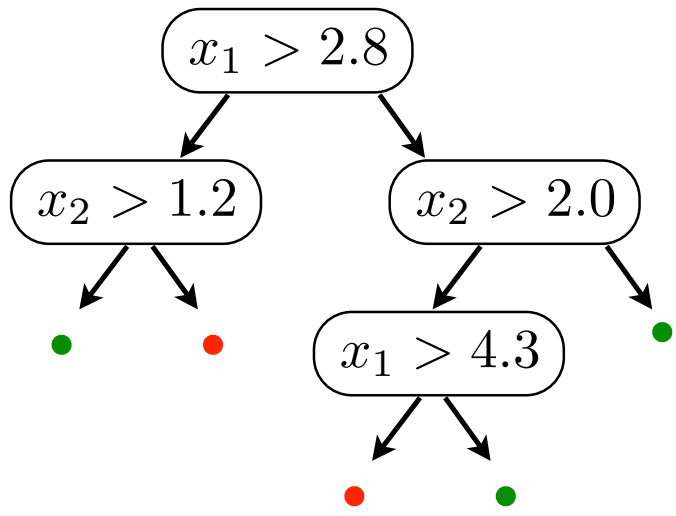


- Use pre-pruning (stop 'early' before adding too many nodes) or post-pruning (after the full tree is constructed, try to improve it by removing nodes)

All classifiers have this problem: Decision trees
(Discussed soon in more detail!)

- What is the appropriate depth of the tree?

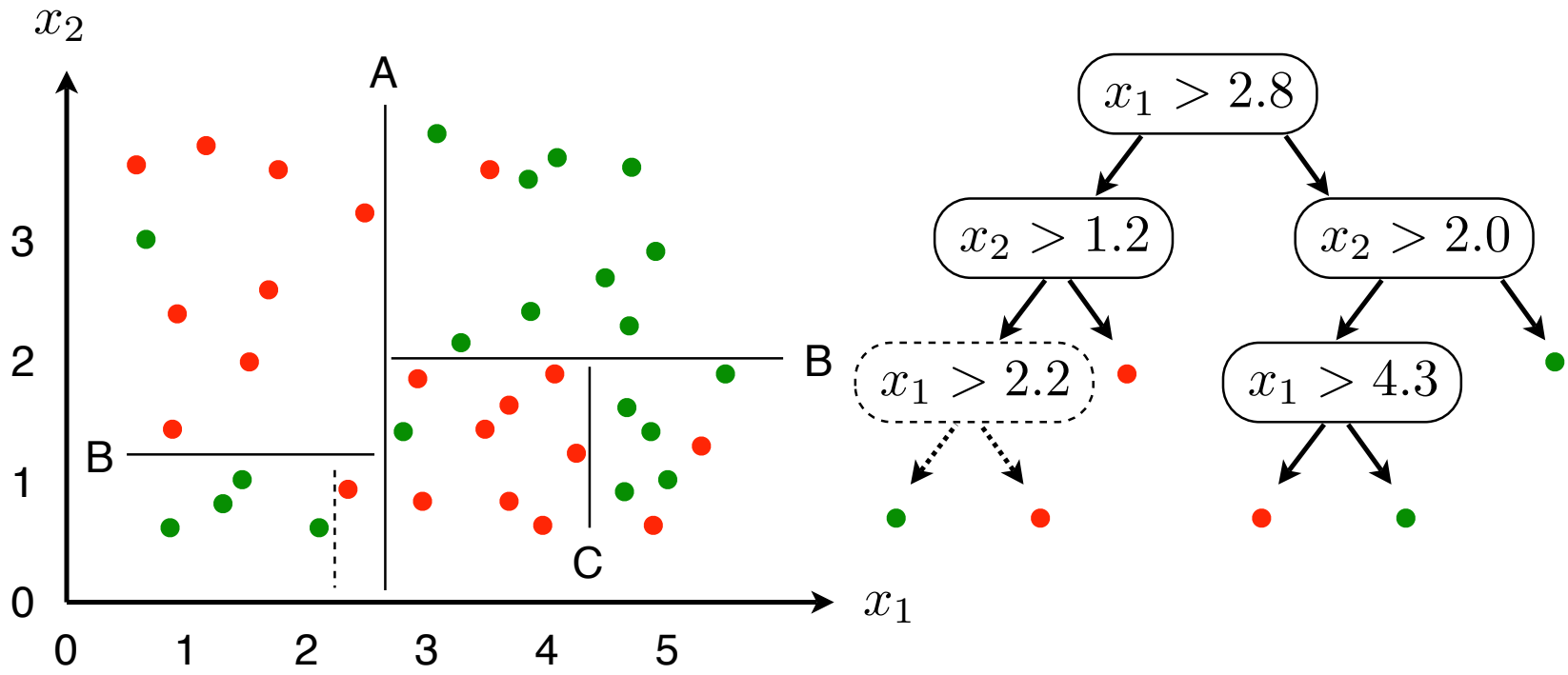




- Use pre-pruning (stop 'early' before adding too many nodes) or post-pruning (after the full tree is constructed, try to improve it by removing nodes)

All classifiers have this problem: Decision trees
(Discussed soon in more detail!)

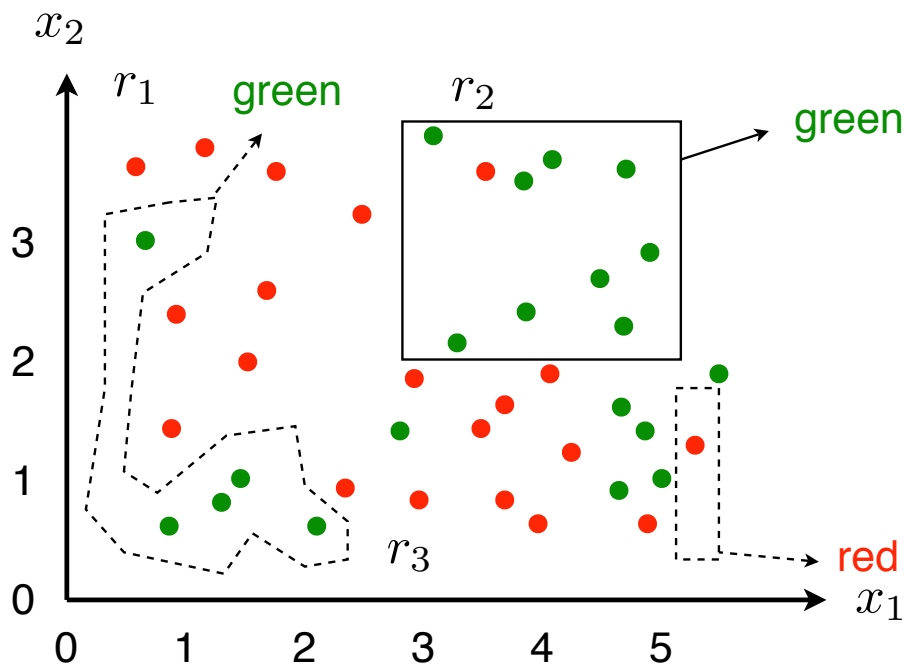
- What is the appropriate depth of the tree?



- Use pre-pruning (stop 'early' before adding too many nodes) or post-pruning (after the full tree is constructed, try to improve it by removing nodes)

All classifiers have this problem: Rule-based classifiers
(Discussed soon in more detail!)

- Rules are likely to be unreliable if:
  - ...they have very low coverage
  - ...they have complicated antecedents



All *regression* methods have this problem as well

- E.g. What degree polynomial to fit to the data?
  - Note that low degree polynomials are *special cases* of higher-degree polynomials, with some coefficients set to zero (i.e. the model classes are 'nested'), e.g. second-degree polynomial as special case of fourth-degree:

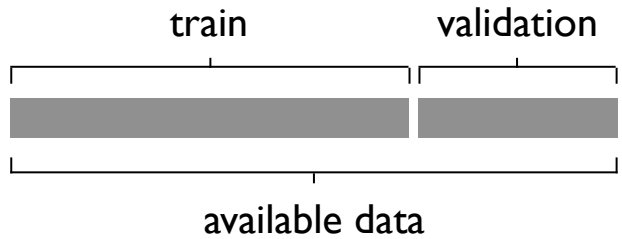
$$y = c_0 + c_1 \cdot x + c_2 \cdot x^2 + 0 \cdot x^3 + 0 \cdot x^4 \tag{14}$$

  $\Rightarrow$ a high-degree polynomial is guaranteed to fit the data *at least* as well as a low-degree one!

  $\Rightarrow$ need some form of regularization to avoid overlearning!

Using 'validation' data to select model complexity

1. Split the data into 'train' and 'validation' subsets:



2. Fit models with varying complexity on 'train' data, e.g.
   - kNN with varying k
   - Naive Bayes with varying amounts of smoothing
   - Decision tree with varying depth
   - Rule-based classifier with various coverage requirements

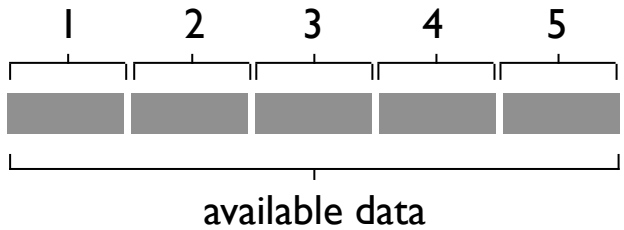
3. Choose the best model based on the performance on the 'validation' set

(Not quite optimal since the amount of training data is not the same as in the original problem. Also: trade-off between the amount of training vs validation data)

Cross-validation

To get more reliable statistics than a single 'split' provides, use K-fold *cross-validation:*

1. Divide the data into K equal-sized subsets:



2. For j goes from 1 to K:
   - 2.1 Train the model(s) using all data except that of subset j
   - 2.2 Compute the resulting validation error on the subset j

3. Average the K results

When $K = N$ (i.e. each datapoint is a separate subset) this is known as *leave-one-out* cross-validation.

Multiple testing (multiple comparison)

Example:

- A number of investment advisors who are predicting whether the market will rise or fall in the next day.
- No advisor is actually any better than a coin-flip
- We have the record of 50 advisors for 10 consecutive days.
- Probability that a specific advisor will be correct at least 8 days out of 10:

$$\frac{\binom{10}{8} + \binom{10}{9} + \binom{10}{10}}{2^{10}} \approx 0.0547 \tag{15}$$

- Probability that at least one of them gets at least 8 correct guesses:

$$1 - (1 - 0.0547)^{50} \approx 0.9399 \tag{16}$$

- The moral of the story: If you are comparing a number of random predictors, it is likely that *some* will have very good empirical performance *even if they are all quite random.*

- It makes sense to select the best predictors, but one should not have much faith in their predictive power unless one *after selection* tests them on a fresh dataset

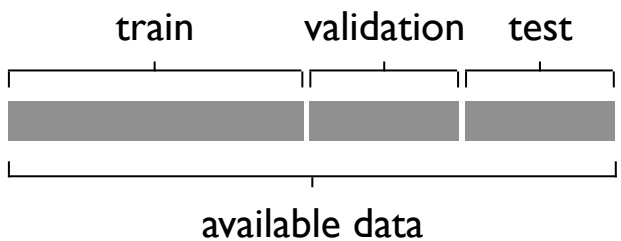
- This problem is strongly related to machine learning:

  1. Many machine learning algorithms employ multiple tests when selecting nodes to split or variables to include in the analysis
  2. Overall, data mining is concerned with finding interesting and useful relationships in data, and if the search is not restricted it is almost always possible to find 'strange patterns', even in completely random data.

- The importance of this issue cannot be overstated!

Estimating generalization performance

- So, at the end, to get an unbiased estimate of the generalization performance, test it on data that *has not been used in any way to guide the search for the best model*. (The test data should be 'locked in a vault'. No peeking!)
- If you are testing several models, beware that the best/worst results may not be good estimates of the generalization error of those models. (Don't fall into that trap, again!)
- Often, data is divided in three: 'train', 'validation', and 'test'



- Can use a cross-validation approach for train/validation, *but no peeking at the test data!*

Explicitly penalizing complexity

- Models can be regularized by explicitly adding a term that penalizes for complexity, as in

  $$(\text{model score}) = (\text{model fit to data}) - \lambda \cdot (\text{model complexity})$$

  where the fit to data can be likelihood, classification accuracy, or similar, and the model complexity can be
  - Number of nodes in a decision tree
  - Norm of the weights (parameters) of the model
  - ...

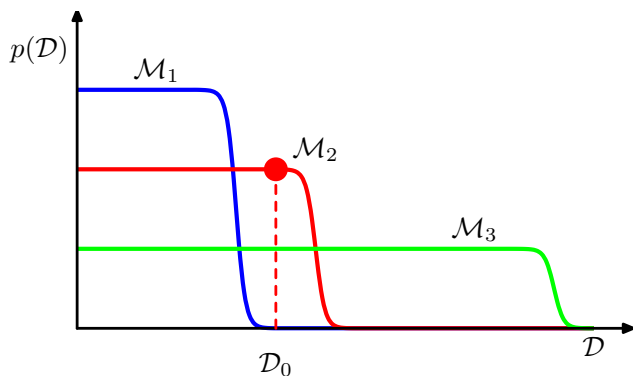
- But how to select λ?
  Cross-validation, Bayesian approach, MDL, BIC, AIC...

Bayesian model selection

- Treat the model parameters as a (hidden) random vector θ, with a prior probaility density $p(\theta \mid \mathcal{M}_i)$ for each model $\mathcal{M}_i$ (with different complexities for different i).
- Select the model $\mathcal{M}_i$ which gives the highest *marginal likelihood* of the observed data, integrating out θ:

$$p(\text{data} \mid \mathcal{M}_i) = \int_\theta p(\text{data} \mid \theta, \mathcal{M}_i)\, p(\theta \mid \mathcal{M}_i) \qquad (17)$$

- This favors the simplest (least flexible) models that nevertheless provide a decent fit to the observed data.



Minimum description length (MDL)

- Basic idea: The best 'model' for the data is the most compact (shortest) description of the data
- Typically, this will consist of describing some general characteristics of the data, and then the details of how the data deviates from those predictions:

$$L(\text{description of model}) + L(\text{description of data given the model})$$

- Thus, this framework provides one way of balancing model complexity with data fit (but the devil is in the details)
- For a thorough treatment, take the course 'Information-Theoretic Modeling'