Id: mlbp07.tex,v 1.74 2007/12/10 16:44:30 kaip Exp
1st Lecture: Introduction

Outline

Contents

1 Course Bureaucracy

1.1 General Information

People and Locations

- People:
  - Kai Puolamäki, PhD, lecturing researcher, lecturer.
  - Antti Ukkonen, MSc, course assistant.
- Please see the course web site at `http://www.cis.hut.fi/Opinnot/T-61.3050/2007/` for current information.
- If you want to send email related to the course please use the email alias `t613050@james.hut.fi` (not personal addresses).
- Lectures: in T1 on Tuesdays at 10–12 (11 September to 11 December 2007, no lecture on 30 October).
- Problem sessions: in T1 on Fridays at 10–12 (from 14 September to 7 December, no problem session on 26 October; problem sessions not every week).

Participating

- To participate to this course you need to be a registered student at TKK (that is, you need a student number).
- You must sign in to course using WebTOPI, `https://webtopi.tkk.fi/` Please sign in today, if you have not already done it.
- You will need to have an addresses of form `12345X@students.hut.fi`, where `12345X` is your student number (for exam results, exercise work feedback etc.). Check that this address works (if not, you should contact the student registry and update your email address there!).

Prerequisites

- To participate to this course you need to have the following prerequisite knowledge:
  - basic mathematics and probability courses (Mat-1.1010, Mat-1.1020, Mat-1.1031/1032 and Mat-1.2600/2620; or equivalent);
  - basics of programming (T-106.1200/1203/1206/1207 or equivalent); and
  - data structures and algorithms (T-106.1220/1223 or equivalent).
- If you lack this prerequisite knowledge we strongly encourage you to take the above mentioned courses before participating to this course!
- You should be able to complete the problems in the prerequisite knowledge test (problem 1) for the first problem session next Friday (see the instructions in the problem sheet).

How to Pass the Course

- You will get 5 cr for passing this course.
- Requirements for passing the course:
  - Pass the *exercise work*. The exercise work should be submitted by 2 January 2008. More instructions will appear in a few weeks time.
  - Pass the *examination*. You can participate to the examination after passing the exercise work (exception: you can participate to the December examination before passing the exercise work; you'll then pass the course if you pass the exercise work).
- Optional, but useful:
  - Lectures.
  - Problem sessions.
  - Reading the book and other material.

About Exercise Work

- Detailed instructions for the exercise work will be announced within a couple of weeks.
- The exercise work will include a data analysis challenge.
- The final report, which should describe the methods you have used and your results, should be submitted at 2 January 2008, at latest.
- You can submit the results of the data analysis challenge by 1 December 2007.
- You must pass the exercise work to pass the course. You will get an increase to your grade if your report is well done. You get some extra points if you additionally perform well in the data analysis challenge.

About Examination

- The examinations are *currently* scheduled as follows:
  - In B at 16–19 on 19 December 2007.
  - In * at 10–13 on 2 February 2008.
  - In T1 at 13–16 on 15 May 2008.
- Check the exam schedule later, times may still change!
- You must pass the exercise work before participating to the examination (exception: you can participate to the December examination before passing the exercise work; you'll then pass the course if you pass the exercise work).
- You must sign in to the examination at least one week in advance using WebTOPI, `https://webtopi.tkk.fi/`

- The examination will be based on the parts of the Alpaydin's book discussed in the lectures, plus on the PDF chapter to be distributed from the course web site.
- Lectures, problem sessions and doing the exercise work help.

How to Get a Grade

- You need to pass both the exercise work and the examination to pass the course.
- You will get a grade of 1–5 based mainly on the examination. You can increase your grade by...
  - Participating to the problem sessions diligently.
  - Solving the exercise work well.
  - Submitting a good answer by 1 December 2007 to the data analysis challenge of the exercise work.

Literature

- The course follows a subset of the book: *Alpaydin, 2004. Introduction to Machine Learning. The MIT Press.*
- Additionally, there will also be a PDF chapter on algorithmics (complexity of problems, local minima etc.) to be distributed from the course web site.
- The lecture slides are available for download from the course web site. I have also given Edita a permission to print them on request.
- You might also find the material — especially the errata and slides — at the Alpaydin's web site (see the link at the course web site) useful.

1.2 Relation to Old Courses

Relation to the Old Courses

- The CIS course reform: more weight on the principles of machine learning, less weight to the neural networks beginning Autumn 2007.
- In curriculum and for the purposes of the degree requirements, this course replaces the old course T-61.3030 (and T-61.261) Principles of Neural Computing.
- However, the contents of this course have little overlap with the old course T-61.3030 Principles of Neural Computing.

Relation to the Old Courses

See `http://www.cis.hut.fi/Opinnot/T-61.3050/oldcourses`

Old course (before Autumn 2007)	New course
T-61.3030 Principles of Neural Computing	T-61.3050 Machine Learning: Basic Principles
T-61.5030 Advanced Course in Neural Computing	T-61.5130 Machine Learning and Neural Networks
T-61.5040 Learning Models and Methods	T-61.5140 Machine Learning: Advanced Probabilistic Methods

Table 1: Correspondences in degree requirements.

Old course (before Autumn 2007)	New course
T-61.5040 Learning Models and Methods	T-61.3050 Machine Learning: Basic Principles T-61.5140 Machine Learning: Advanced Probabilistic Methods
T-61.3030 Principles of Neural Computing T-61.5030 Advanced Course in Neural Computing	T-61.5130 Machine Learning and Neural Networks

Table 2: *Approximate* topical correspondeces.

1.3 Contents of the Course

Very Preliminary Plan of the Topics

- Supervised learning, Bayesian decision theory, probability distributions and parametric methods, multivariate methods, clustering (mostly Alpaydin's chapters 1–7 and appendix A)
- Algorithmic issues in machine learning, such as hardness of problems, approximation techniques and their features (such as local minima), time and memory complexity in data analysis (separate PDF chapter to be distributed from the course web site)
- Nonparametric methods (Alpaydin 8.1–8.2), linear discrimination (Alpaydin 10.1–10.8), assessing and comparing classification algorithms (Alpaydin's chapter 14)
- I'll try to keep the Alpaydin's ordering of topics, and emphasize principles rather than to go through all possible algorithms and methods.

What You Should Know After the Course

- After this course, you should...
  - be able to apply the basic methods to real world data;
  - understand the basic principles of the methods; and
  - have necessary prerequisites to understand and apply new concepts and methods that build on the topics covered in the course.
- This course does *not* include:
  - all possible machine learning methods; or
  - all possible applications of machine learning.

2 Chapter 1: Introduction

2.1 Examples of Machine Learning Applications

What is Machine Learning?

Definition 1. Machine learning is programming computers to optimize a performance criterion using example data or past experience. (Alpaydin)

?

Examples of Applications

- Associations (basket analysis)
- Supervised learning
  - Classification
  - Regression
- Unsupervised learning
- Reinforcement learning (not in this course)

Association rules

- Example: sales data
  - rows: customer transactions (millions)
  - columns: products bought (thousands)
- Question: Can you find something interesting of this?

Association rule
"80% of customers who buy beer and sausage buy also mustard." Or: $P(\text{mustard} \mid \text{beer}, \text{sausage}) = 0.8$.

- Accuracy (conditional probability): 0.8
- Frequency or support (fraction of clients who bought mustard, beer and sausage): 0.3

Classification

- Example: data on credit card applicants
- Question: Should a client be granted a credit card?
- Differentiate between low-risk (+) and *high-risk (-)* customers using their *income* and *savings*.

Discriminant
IF *income*$> \theta_1$ AND *savings*$> \theta_2$ THEN *low-risk* ELSE *high-risk*.

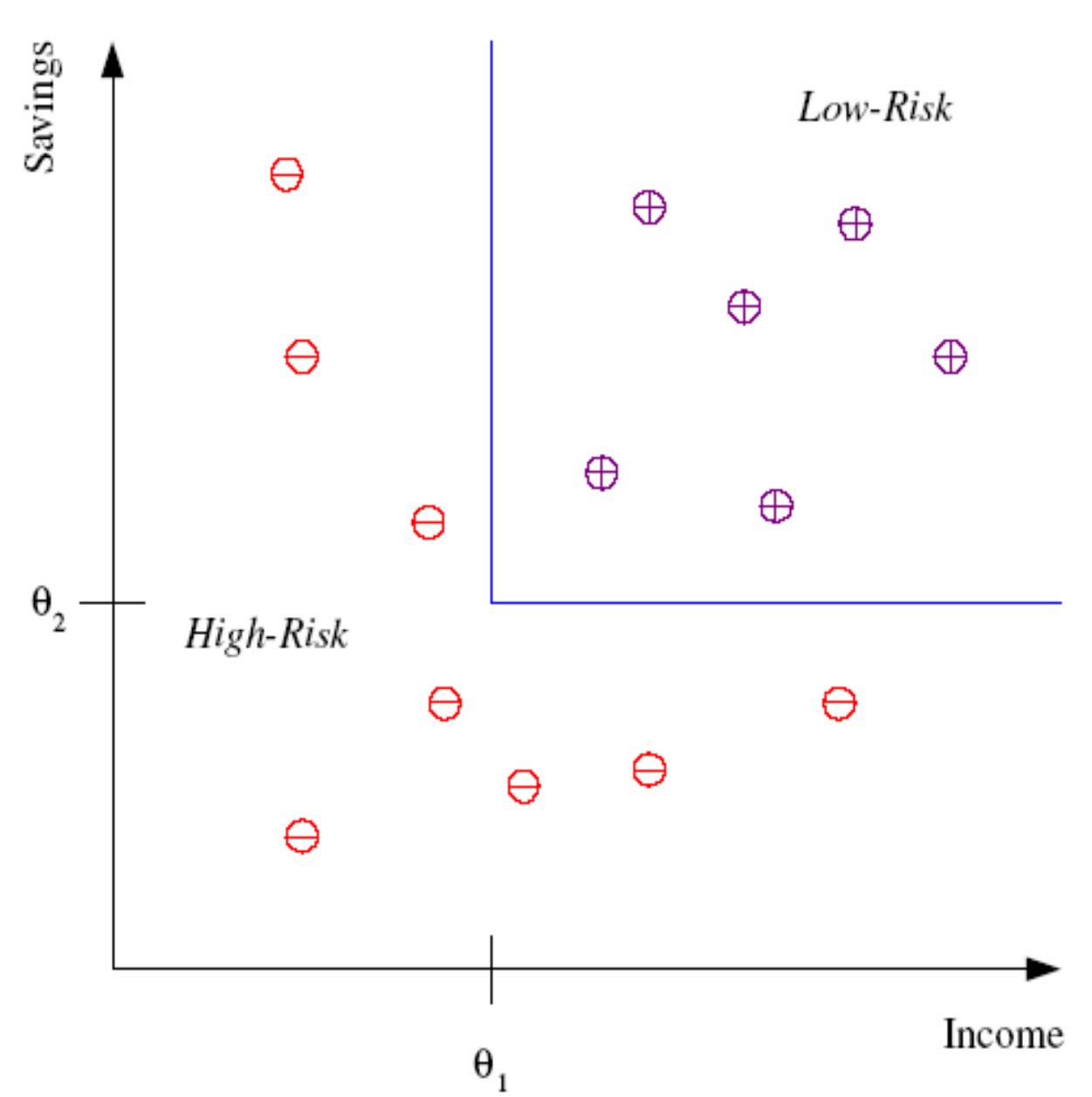
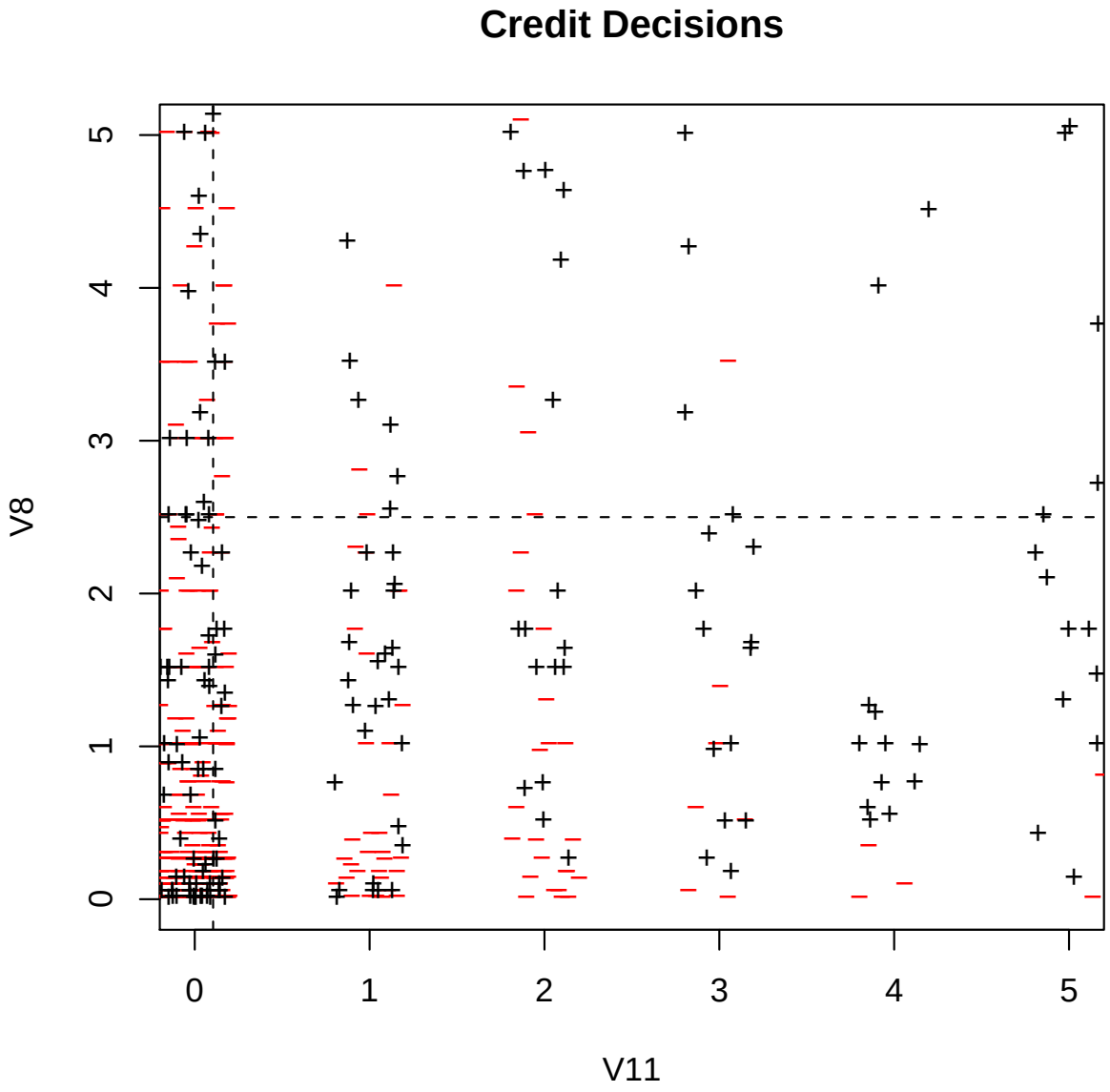


Figure 1.1 of Alpaydin (2004).



Real CREDIT-SCREENING data from UCI Machine Learning Repository.

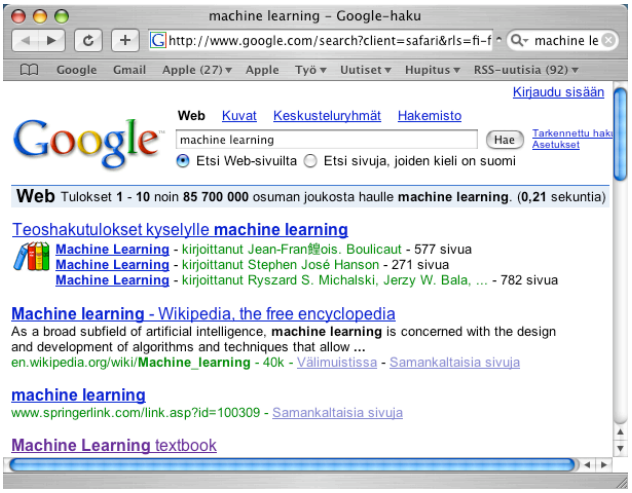
Classification

- Classification: predict something (variate, Y), given something else (covariate, X). Or: try to estimate $P(Y \mid X)$.
- *Speech recognition*: temporal dependency. Predict words, given the speech signal.
- *Character recognition* (OCR): different handwriting styles.
- *Medical diagnosis*: from symptoms to diagnosis.

- *Eye movement analysis*: is the user interested in the text she is reading?
- ...

Classification

- The Internet search engines use machine learning to give the best search results, given a query.
- Fundamental problem in *information retrieval*: given a query ("machine learning"), list relevant documents (web sites related to "machine learning").



Classification

[movie, link]

Classification

- Example: eye movement measurements during information search (ongoing research by the lecturer and his friends during 2003–2007, see `http://www.cis.hut.fi/projects/mi/proact`)
- Question 1: Is the user interested in text she is reading?
- Question 2: What is the user interested in?
- This is a classification problem: predict relevance of a viewed document or true interest of the user, given the eye movement trajectory.
- The problem is (was) quite difficult to solve.

Classification

- Eye movements are measured in a controlled experiment.
- A sentence (title of a scientific article) is partitioned into words.
- Most discriminative word-specific features were used (one or many fixations, total fixation duration, reading behaviour).
- The title relevance was predicted using a discriminative machine learning models.

Pleistocene to Holocene Extinction Dynamics in Giant Deer and Woolly Mammoth

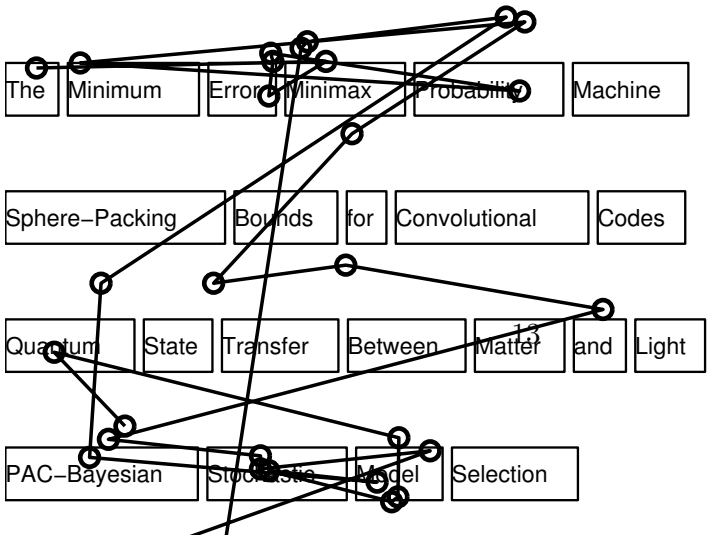
How to Better Use Expert Advice

Accelerating Reinforcement Learning through Implicit Imitation

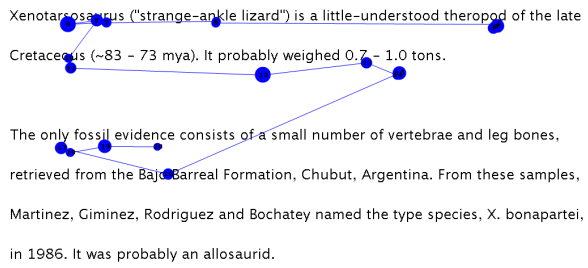
Models of the Mechanism Underlying Perceived Location of a Perisaccadic Flash

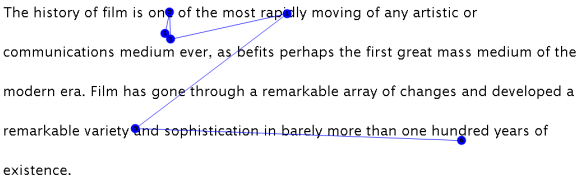
Updating Probabilities

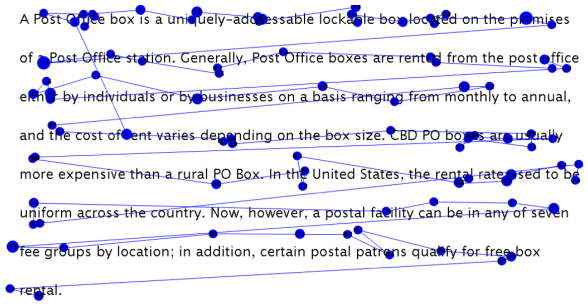
Expression Influences the Recognition of Familiar Faces

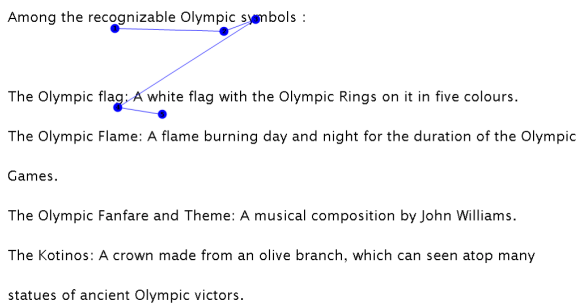


Classification



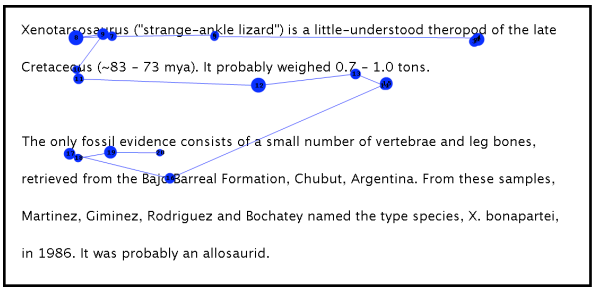


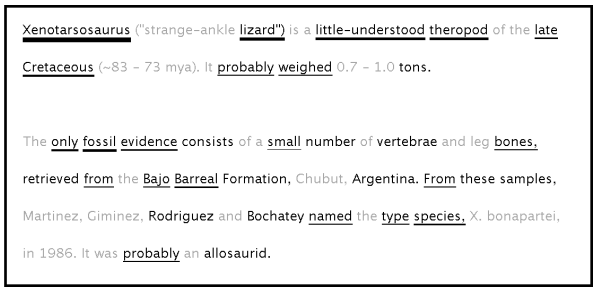




Classification

- For this to work, there must be a link between the relevance of a word to a topic of the user's interest and eye movements related to it.
- This link can be learned and used on new topics.

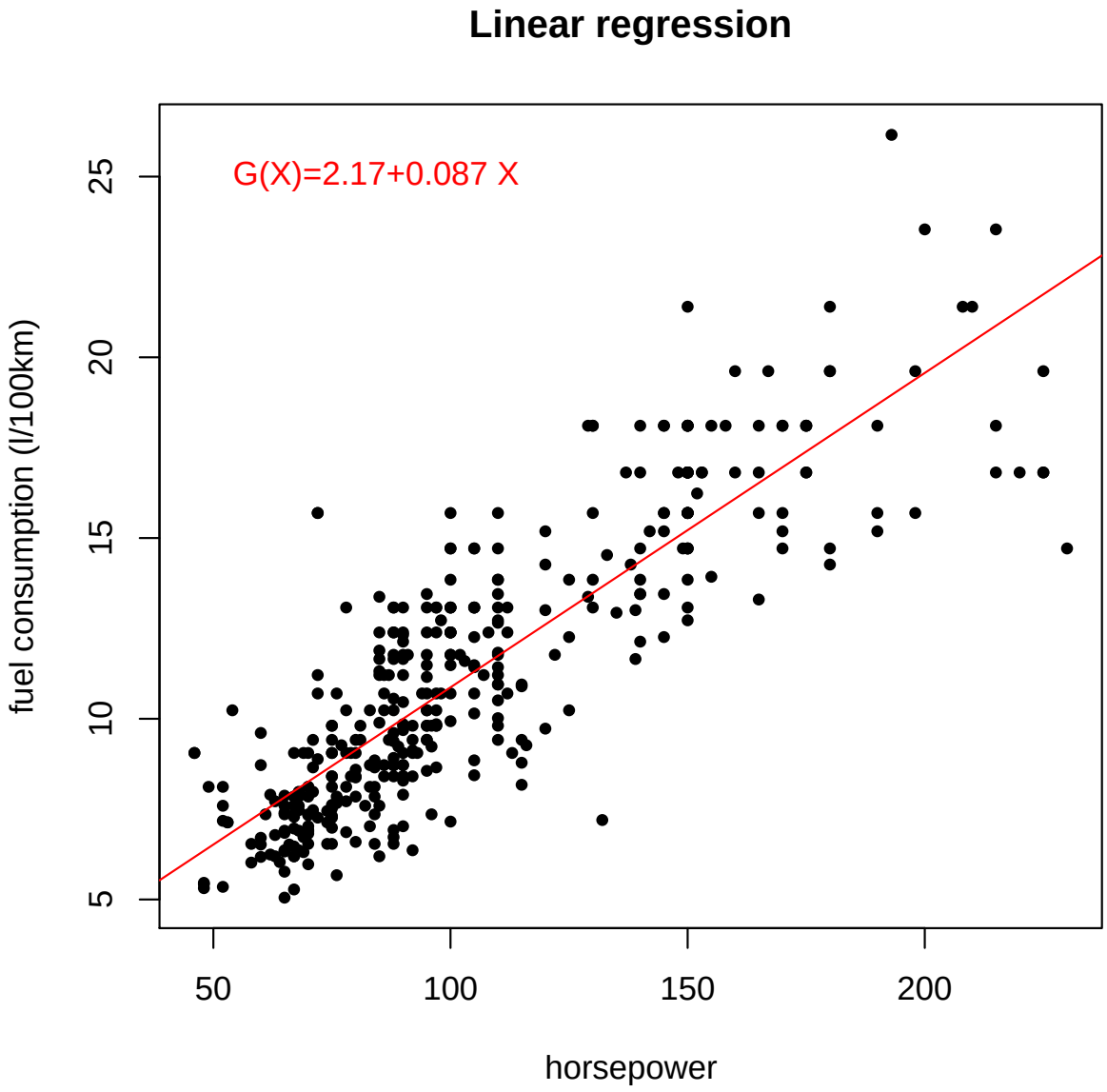


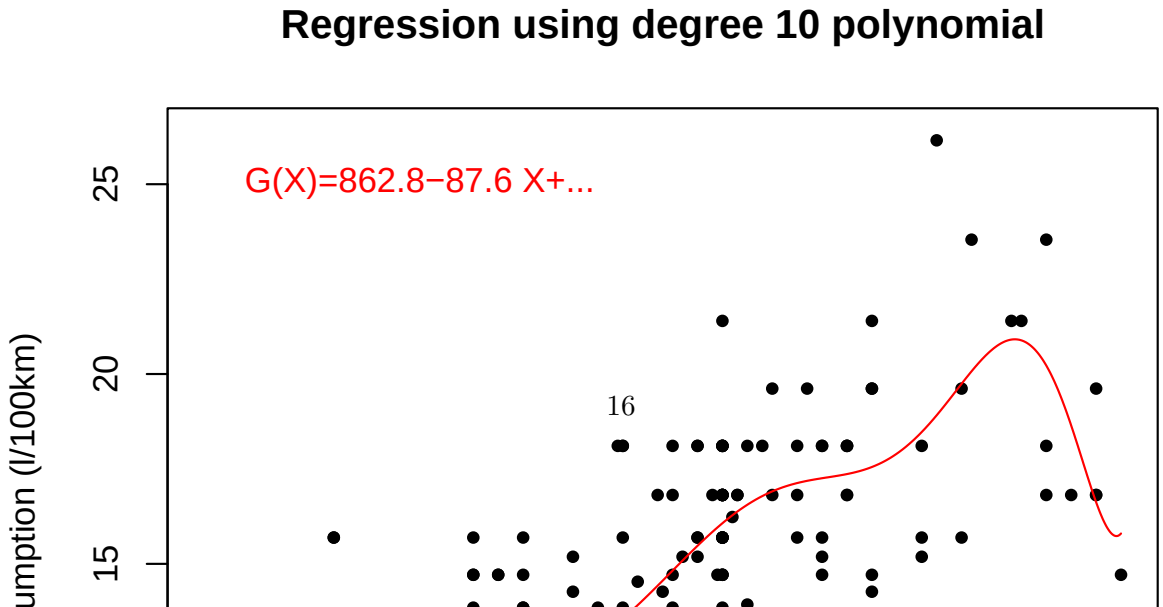


Regression

- *Regression* is classification where the variate Y is a continuous variable.
- The principles in classification and regression are the same, methods differ.
- Example: fuel consumption of cars.

- Y: fuel consumption.
- X: car attributes.
- $Y = G(X \mid \theta)$
  - $G()$: a model.
  - θ: model parameters.


Linear regression
G(X)=2.17+0.087 X
fuel consumption (l/100km)
horsepower
5
10
15
20
25
50
100
150
200


Regression using degree 10 polynomial
G(X)=862.8−87.6 X+...
fuel consumption (l/100km)
15
20
25

AUTO-MPG data set from UCI Machine Learning Repository.

Uses of Supervised Learning

- *Prediction of future cases:* Use the rule to predict the output for future inputs.
- *Knowledge extraction:* The rule is easy to understand.
- *Compression:* The rule is simpler than the data it explains.
- *Outlier detection:* Exceptions that are not covered by the rule, for example, fraud.

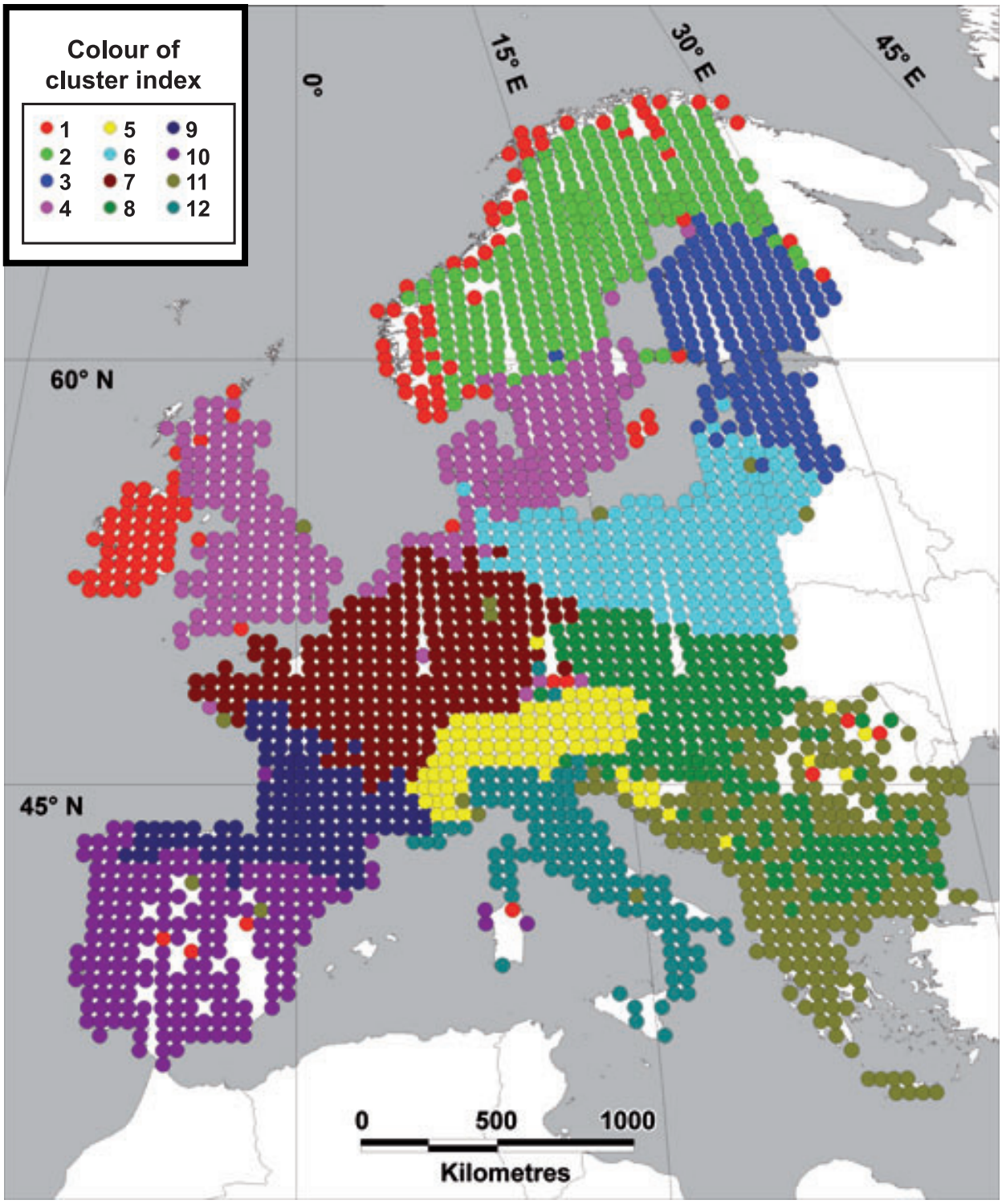
Unsupervised Learning

- In *supervised learning*, an imaginary "supervisor" tells us in the training phase what is the correct variate (Y), given the covariate (X). We then try to predict $P(Y \mid X)$ without the supervisor.
- *Unsupervised learning* is like supervised learning, except there is no supervisor telling us the Y. We try to predict $P(X)$. (In supervised learning we really do not care about $P(X)$.)
- Another view: unsupervised learning is like supervised learning, except the covariate Y is fixed, in which case we try to predict $P(Y \mid X) = P(Y)$.
- Again, the principles are the same, but the methods differ.
- Example: clustering (grouping similar instances together)
- Example: probabilistic modeling (find the most likely model to describe the data, given some prior family of models)

Clustering

- Example: European land mammals.
- Question: Can we find ecological communities?
- Question: What explains the communities?
- The 50×50 km map grids were grouped into clusters. Map grids within a cluster should occupy similar mammals.

Heikinheimo et al. (2007) Biogeography of European land mammals... J Biogeogr.

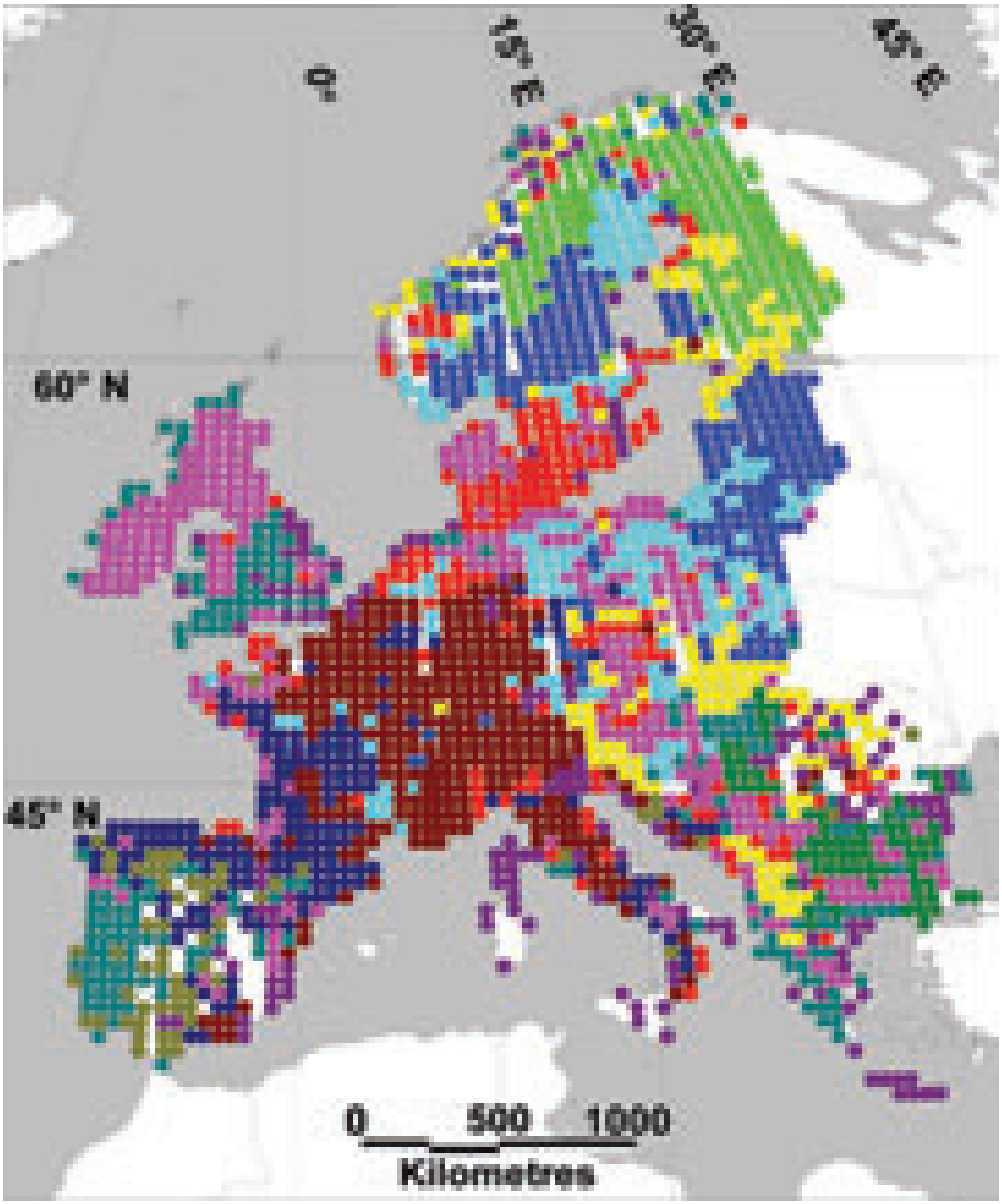


Clustering

- Endangered species appear to have least spatial coherence.

- The clustering can be explained mostly by temperature and precipitation.
- Somewhat surprisingly the natural factors seem to explain the mammalian metacommunity distributions, despite a long history of intensive human presence.

At risk



Other Applications of Machine Learning

- Bioinformatics
- ...

Reinforcement Learning

- Learning a policy: A sequence of output.
- No supervised output but delayed reward.
- Credit assignment problem.
- Game playing.
- Robot in a maze.
- Multiple agents, partial observability...
- Example: our search engine is showing an user documents. The user tells us if the shown document is interesting. Tradeoff:
  - *Exploitation:* show the user documents that we think might interest her most (immediate reward).
  - *Exploration:* show the user uninteresting documents with which we would learn more of her interests (delayed reward).
- Not covered in this course.

2.2 What is Machine Learning?

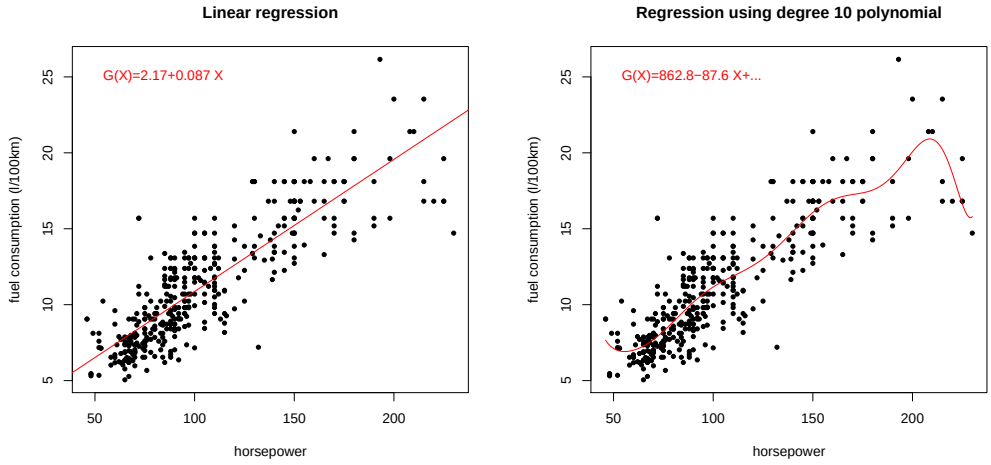
What is Machine Learning?

Definition 2. Machine learning is programming computers to optimize a performance criterion using example data or past experience. (Alpaydin)

- Machine learning is using computers to analyze data.
- The data is noisy, there are measurement errors etc.
- We usually do not observe all factors that would be needed for certainty: we must resort to statistics.
- What is "learning"? Often, we do not want just to describe the data we have, but be able to predict of (yet) unseen data.

About Generalization

- Often, it would be quite easy to make a model that would describe already known data.
- It is more difficult to...
  - Say something (predict) of yet unseen data (generalization).
  - Make a good (not too complex and not too simple) description of known data.
- Prior knowledge is important.



What is Machine Learning?

- How does machine learning relate to *data mining*?
- How does machine learning relate to *statistics*?
- How does machine learning relate to *algorithms*?
- How does machine learning relate to artificial intelligence, neural networks, ...?

Machine Learning and Data Mining

- Machine learning has (depending on the speaker) a strong overlap with data mining.
- Machine learning emphasizes statistical principles and methods.
- Data mining emphasizes algorithms which also work on large data volumes.
- Data miners may also have a modest goal of helping user to find something interesting of the data, not attempting to make a model of the world.

Machine Learning and Statistics

- Modern statistics forms (with algorithms) the theoretical foundations of machine learning.
- In "traditional" statistics one typically tests single hypothesis of the data. Example: patients with a new treatment had 80% recovery rate, while patients with the old treatment had 60% recovery rate. Is the new treatment more effective than the old one?

Machine Learning and Algorithms

- Algorithms are needed to solve machine learning problems.
- In machine learning the algorithmic aspects (convergence, running times etc.) have not been emphasized. This is however changing.
- Summary: there are lots of connections between machine learning and various disciplines. The exact connections vary depending on whom you ask. The field is still developing.

2.3 Resources

Software

- There is lots of good software available. You will need some software to pass this course (for example, exercise work). Some examples follow.
- R. An open source software for statistical computing and publication quality graphics. An usable functional programming language. (Lecturer's favourite.)
- Matlab. Matlab is a commercial software that is especially popular in signal processing. It is too matrix-oriented for the lecturer's taste. Quite a few people use it (including Alpaydin), though. Matlab has an open source variant, GNU Octave.
- Weka. Open source Weka is a collection of machine learning algorithms for solving real-world data mining problems. It is written in Java and runs on almost any platform. (Assistant seems to like it.)

Datasets

- Often, finding a good data set one of the most difficult tasks in developing machine learning methods.
- UCI Repository: `http://www.ics.uci.edu/~mlearn/MLRepository.html`
- UCI KDD Archive: `http://kdd.ics.uci.edu/summary.data.application.html`
- Statlib: `http://lib.stat.cmu.edu/`
- Delve: `http://www.cs.utoronto.ca/~delve/`

Journals

- Journal of Machine Learning Research
- Machine Learning
- Neural Computation
- Neural Networks
- IEEE Transactions on Neural Networks
- IEEE Transactions on Pattern Analysis and Machine Intelligence
- Annals of Statistics
- Journal of the American Statistical Association
- ...

Conferences

- International Conference on Machine Learning (ICML)
- European Conference on Machine Learning (ECML)
- Neural Information Processing Systems (NIPS)
- Uncertainty in Artificial Intelligence (UAI)
- Computational Learning Theory (COLT)
- International Joint Conference on Artificial Intelligence (IJCAI)
- International Conference on Neural Networks (Europe)
- ...

Questions?

Next lecture

- Next Tuesday: Chapter 2 of Alpaydin (2004), "Supervised Learning".
- Remember the problem session next Friday at 10 o'clock.

2nd Lecture: Supervised Learning

3 Learning a Class from Examples

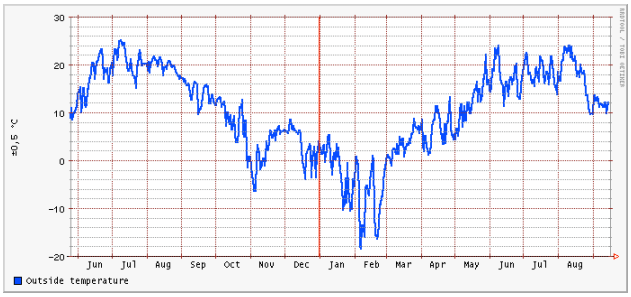
3.1 Introduction

Learning a Class from Examples

- What follows is some theory of classification into two classes.
- We assume there is no noise (results can be generalized to noise, though).
- What you should learn:
  - Learning can be seen as pruning out possible hypothesis.
  - Learning is generalization (we want to predict classes of new examples).
  - Learning is impossible if the hypothesis space is too large (in other words: we need some prior information, we need to select a model family)
  - The complexity of the hypothesis space (model family) can be characterized using the VC dimension.
  - More complex model, bigger the training data needed.

Independent and Identically Distributed (iid) Data

- We assume that we have a training data $\mathcal{X}$ that contains N data points drawn independently from the identical distribution.
- In other words: ordering of the data points does not matter.
- Usually a good approximation.
- Notable exception: time series.
- Example: today's temperature is not independent of the yesterday's temperature, in fact, there is a strong correlation.



Outside temperature in Otaniemi from `http://outside.hut.fi/`.

	$\mathbf{x}^t$						$r(\mathbf{x}^t)$
t	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	1
2	Sunny	Warm	High	Strong	Warm	Same	1
3	Rainy	Cold	High	Strong	Warm	Change	0
4	Sunny	Warm	High	Strong	Cool	Change	1

Table 3: Aldo's observed sport experiences in different weather conditions.

3.2 Aldo and Family Car

Does Aldo Enjoy Sport?

- Question: Does Aldo enjoy sport, given weather conditions?
- Assumption: we have sufficient information (6 weather attributes) that fully determine Aldo's enjoyment of sports (no "noise", Aldo is deterministic).

Does Aldo Enjoy Sport?

- *Hypothesis* h is a function from weather attributes $\mathbf{x}$ to $\{0, 1\}$.
- *Hypothesis class* $\mathcal{H}$ is the chosen set of hypothesis.
- The goal of the learner is to find a hypothesis $h \in \mathcal{H}$ such that $h(\mathbf{x}) = r(\mathbf{x})$ for every possible $\mathbf{x}$.
- *One possible hypothesis class* in Aldo's case is a vector of six weather attributes. For each attribute, the hypothesis will be either:
  - ?: any value is acceptable for this attribute.
  - single value (e.g., "Warm"): required value for this attribute.
  - $\emptyset$: no value is acceptable.
- If an instance $\mathbf{x}$ satisfies the constraints then h classifies this as a positive example, $h(\mathbf{x}) = 1$.
- Example: Aldo enjoys the sport only on cold days with high humidity (independent of other attributes), this would be represented with $(?, Cold, High, ?, ?, ?)$.

Does Aldo Enjoy Sport?

Definition 3. Let h and g be hypothesis on X. h is *more general than or equal to* g (written $h \succeq g$) if and only if

$$\forall \mathbf{x} \in X : g(\mathbf{x}) = 1 \Rightarrow h(\mathbf{x}) = 1.$$

Examples:

- The *most general hypothesis* is represented by $(?, ?, ?, ?, ?, ?)$ (every day is a positive example).
- The *most specific hypothesis* is represented by $(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$ (no day is a positive example).
- $h = (Sunny, ?, ?, ?, ?, ?)$ is more general than $g = (Sunny, ?, ?, Strong, ?, ?)$, or $h \succ g$.

Does Aldo Enjoy Sport?

Definition 4 (Consistent Hypothesis)**.** A hypothesis h is *consistent* with a set of training examples $\mathcal{X}$ if and only if $h(\mathbf{x}) = r(\mathbf{x})$ for each example $(\mathbf{x}, r) \in \mathcal{X}$.

Definition 5 (Version Space)**.** The *version space* is the set of all hypothesis that are consistent with the training examples.

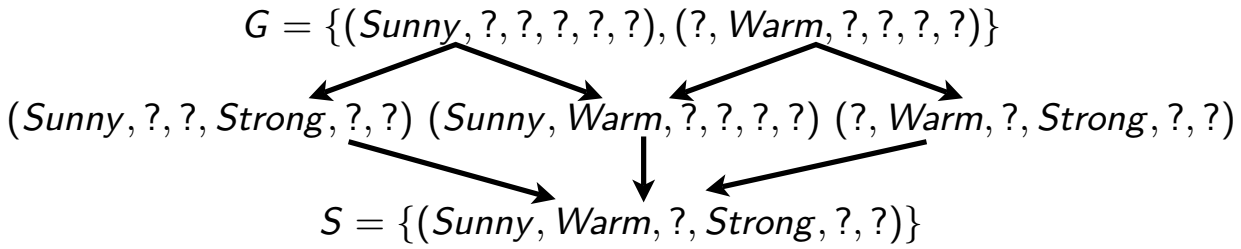
Does Aldo Enjoy Sport?

- Question 1: What are the most general hypothesis that are consistent with the training data (4 days of observation of Aldo)? (*general boundary* G)
- Question 2: What are the most specific hypothesis that are consistent with the training data? (*specific boundary* S)

Theorem 6 (Version Space Representation Theorem)**.** *Let G and S the most general and most specific hypothesis that are consistent with the training data. Then all hypothesis that are consistent with the training data (version space) are given by*

$$\{h \in \mathcal{H} \mid (\exists s \in S)\,(\exists g \in G) : g \succeq h \succeq s\}\,.$$

Does Aldo Enjoy Sport?



	$\mathbf{x}^t$						$r(\mathbf{x}^t)$
t	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	1
2	Sunny	Warm	High	Strong	Warm	Same	1
3	Rainy	Cold	High	Strong	Warm	Change	0
4	Sunny	Warm	High	Strong	Cool	Change	1

See Mitchell (1997) and CANDIDATE-ELIMINATION algorithm for details.

Does Aldo Enjoy Sport?

- One of the consistent hypothesis could be the "truth". For others we get some *error*:

Definition 7 (Error of Hypothesis)**.**

$$E\left(h \mid \mathcal{X}\right) = \frac{1}{N} \sum_{t=1}^{N} \mathbf{1}\left(h(\mathbf{x}^t) \neq r^t\right)$$

- Given enough training samples, we might be able to end up with only one consistent hypothesis.
- Given enough training samples, we might end up with no consistent hypothesis if:
  - If none of the hypothesis in the hypothesis class is correct. (For example, if Aldo would enjoy sport only if (sky is sunny and wind is strong) or (sky is rainy and wind is light).)
  - If there is noise (e.g., some positive examples are incorrectly observed as negative examples).

Does Aldo Enjoy Sport?

- If none of the hypothesis in the hypothesis class is correct we might end up with no consistent hypothesis.
- "Solution": include all possible hypothesis into the hypothesis class! In the Aldo's case, there are $2^{2^6} = 1.8 \times 10^{19}$ possible hypothesis (number of boolean functions with 6 inputs).
- This does not work (even if we could compute): we could not say anything of the unseen cases.
- *Inductive bias*: we must restrict the allowed hypothesis to be able to *generalize* (predict classes of new instances).
- The selection of hypothesis space is called *model selection*.
- *Underfitting*: the hypothesis space is too simple.
- *Overfitting*: the hypothesis space is too complex.

A Family Car

- Question 1: Is car $\mathbf{x}$ a family car, given car properties?
- Question 2: What do people expect from a family car?
- Car properties: $\mathbf{x} = (\text{price}, \text{engine power})$.
- Hypothesis: $h(\mathbf{x}) = 1$ if car is a family car.

A Family Car

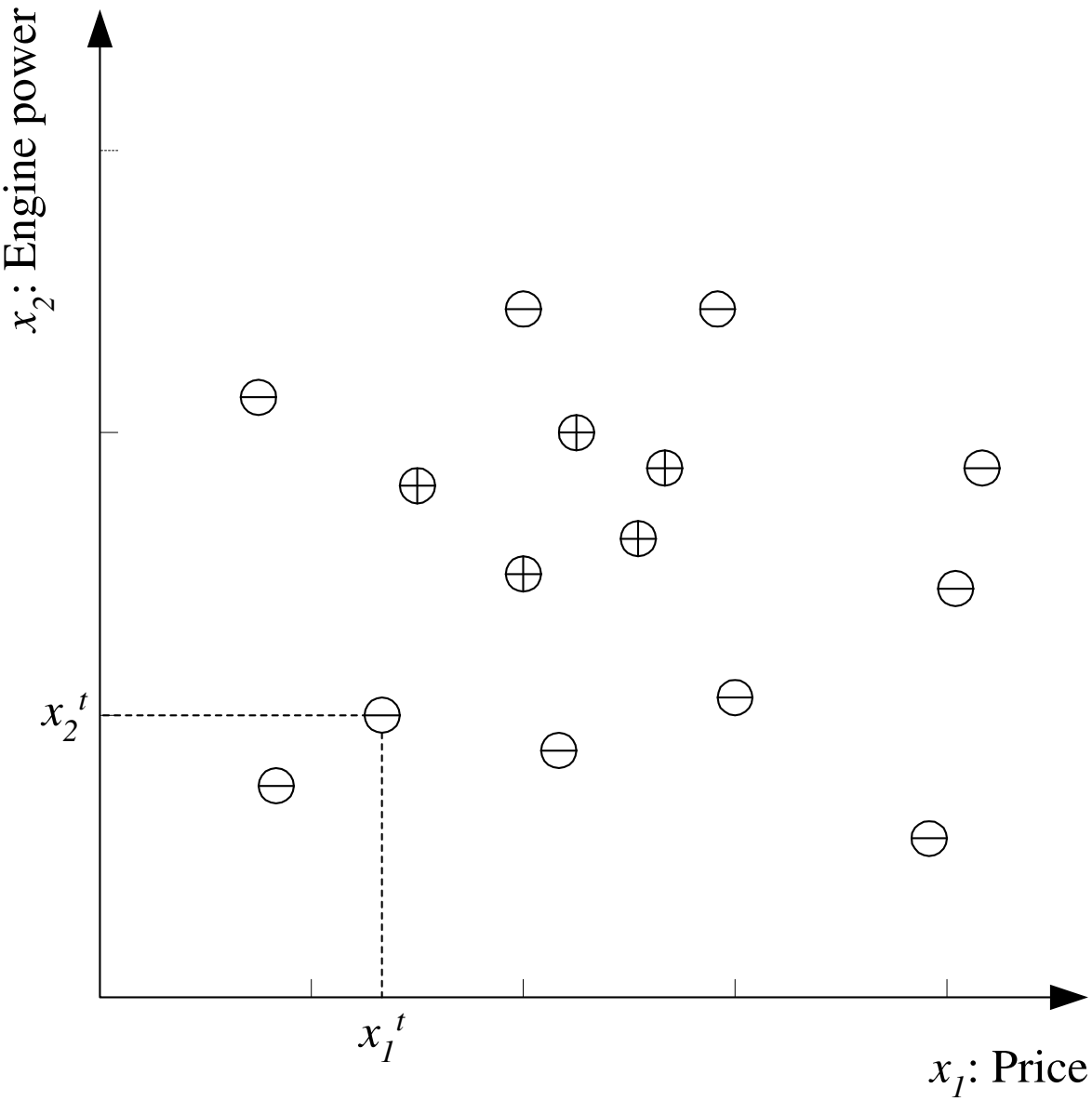


Figure 2.1 of Alpaydin (2004).

$$\mathcal{X} = \left\{\mathbf{x}^t, r^t\right\}_{t=1}^N$$

$$r = \begin{cases} 1 \text{ if } \mathbf{x} \text{ is positive} \\ 0 \text{ if } \mathbf{x} \text{ is negative} \end{cases}$$

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix}$$

A Family Car

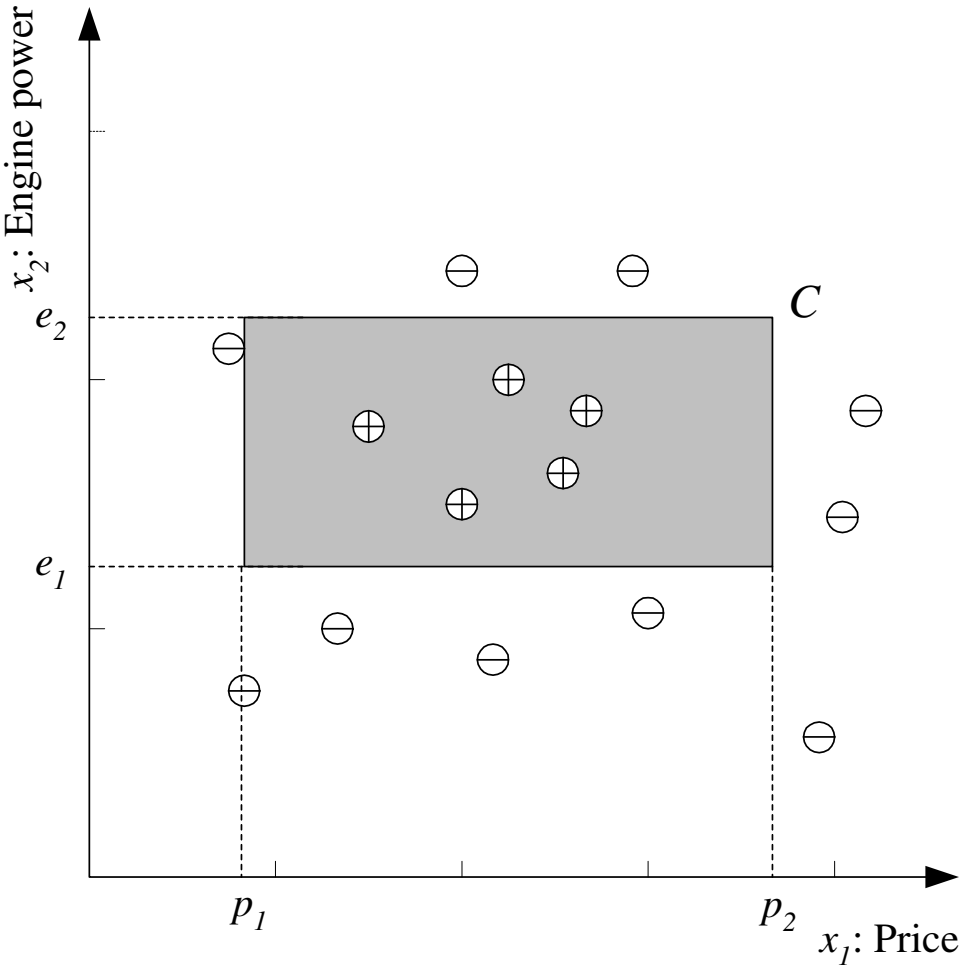


Figure 2.2 of Alpaydin (2004).

$$r(\mathbf{x}) = \begin{cases} 1 & p_1 \leq \text{price} \leq p_2 \wedge e_1 \leq \text{engine power} \leq e_2 \\ 0 & \text{otherwise} \end{cases}$$

A Family Car

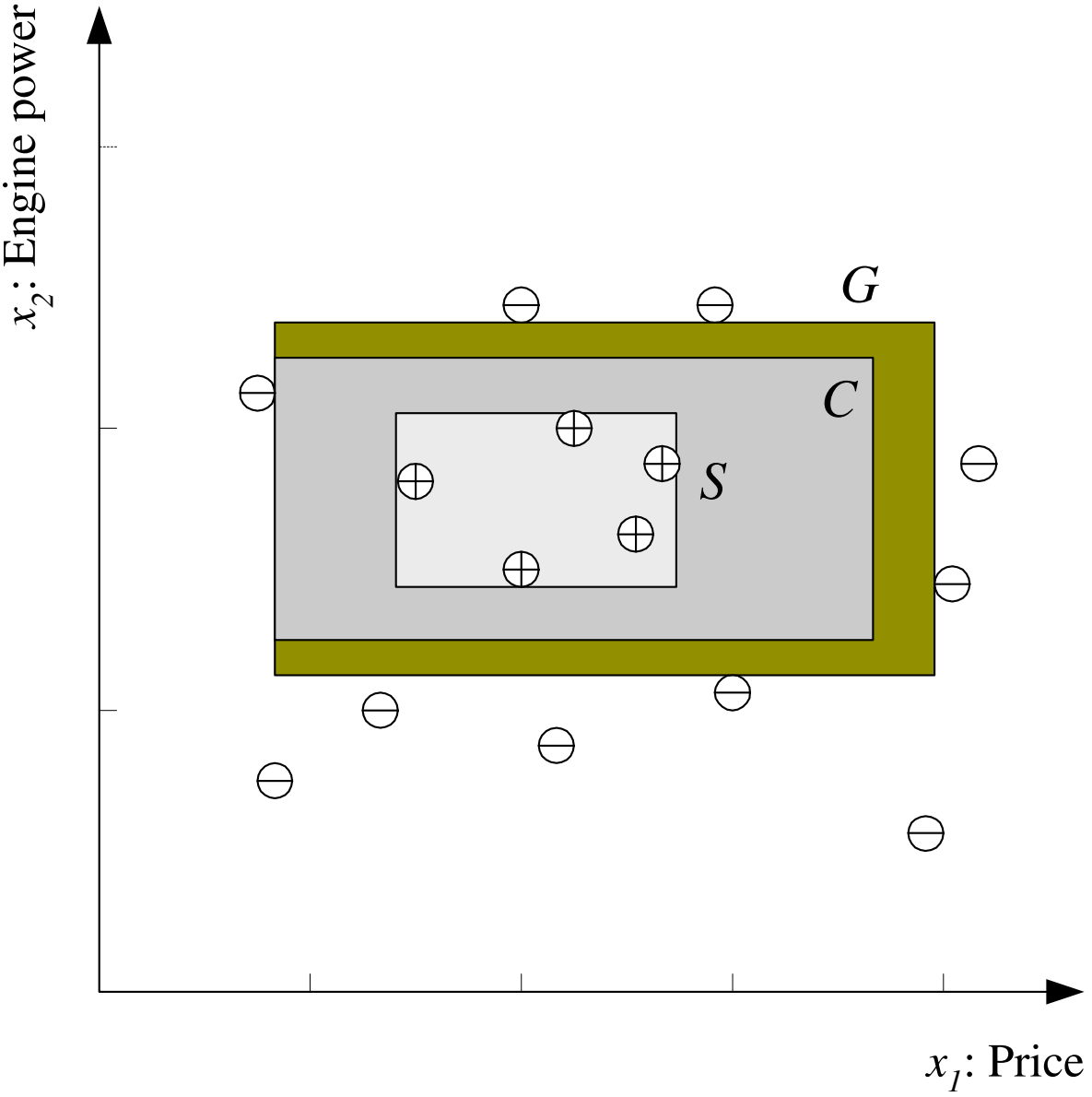


Figure 2.4 of Alpaydin (2004).

Error of h in $\mathcal{X}$:

$$E(h \mid \mathcal{X}) = \frac{1}{N} \sum_{t=1}^{N} \mathbf{1}\left(h(\mathbf{x}^t) \neq r^t\right).$$

$h \in \mathcal{H}$ between S and G is *consistent* and make up the *version space* (error in $\mathcal{X}$ is zero). Notice that if S and G are close the error on new data will be small!

- The hypothesis class H is the set of all rectangles.
- The cars between the most general (G) and most specific (S) hypothesis may be classified incorrectly. C is the correct hypothesis.

What did we learn from Aldo and Family Cars?

- We must choose some hypothesis to be able to predict anything (unless we observe all possible data values). (model selection)
- This causes *inductive bias* (the choice of hypothesis space affects your results).
- All consistent hypothesis can be found between the most general and most specific hypothesis.
- There may be no consistent hypothesis due to too simple hypothesis space (underfitting) or noise. These must be taken into account in practical applications.

3.3 PAC Learning and VC Dimension

Probably Approximately Correct (PAC) Learning

- How many training examples N should we have, such that with *probability of at least* $1-\delta$, any consistent hypothesis h has *error at most* ϵ?

Probably Approximately Correct (PAC) Learning

Theorem 8. *The probability that version space has no hypothesis with error greater than ϵ is at most $|\mathcal{H}|e^{-\epsilon N}$.(Assume finite hypothesis class $\mathcal{H}$.)*

Proof. The probability that a hypothesis that has an error greater than ϵ is consistent with one randomly drawn example is at most $1-\epsilon$. Therefore, the probability that this hypothesis is consistent with N independently drawn examples is at most $(1-\epsilon)^N$. There are at most $|\mathcal{H}|$ hypothesis that have an error greater than ϵ. The probability that there is at least one hypothesis in the version space with an error greater than ϵ is at most $|\mathcal{H}|(1-\epsilon)^N \leq |\mathcal{H}|e^{-\epsilon N}$. □

It follows that $|\mathcal{H}|e^{-\epsilon N} \leq \delta$, or $N \geq \frac{1}{\epsilon}\left(\ln|\mathcal{H}| + \ln(1/\delta)\right)$.

Probably Approximately Correct (PAC) Learning

Theorem 9 (Probably Approximately Correct (PAC) Learning)**.** *We should have N training examples to have an probability of at least $1-\delta$ that any consistent hypothesis h has error at most ϵ, where*

$$N \geq \frac{1}{\epsilon}\left(\ln|\mathcal{H}| + \ln\frac{1}{\delta}\right)$$

If we accept that the best hypothesis might have a non-zero training error (often case in practice) the limit becomes

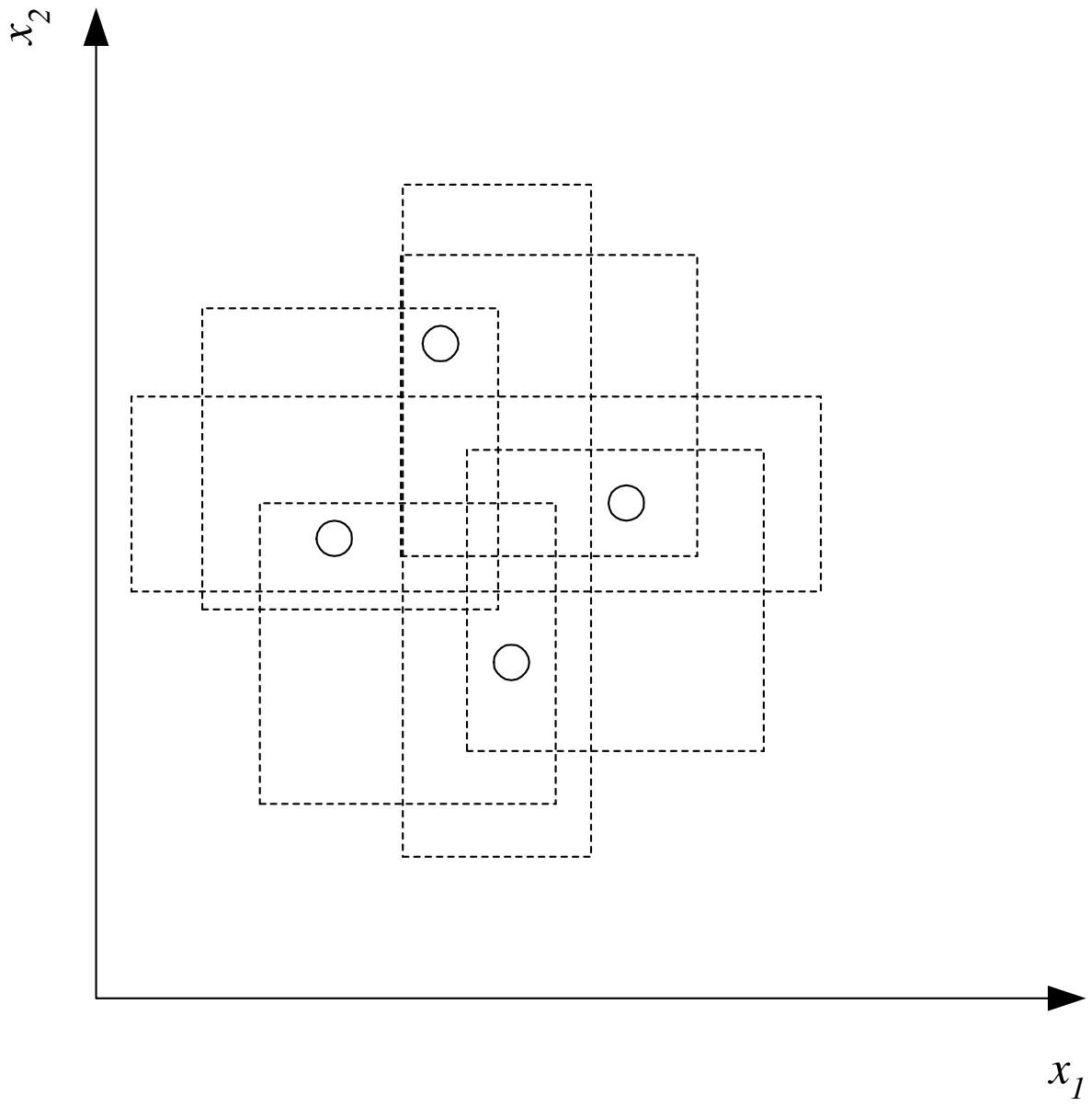
$$N \geq \frac{1}{\epsilon^2}\left(\ln|\mathcal{H}| + \ln\frac{1}{\delta}\right),$$

where the obtained error will be with probability $1-\delta$ no more than $E(h_{best} \mid \mathcal{X}) + \epsilon$, where $E(h_{best} \mid \mathcal{X})$ is the error of the best hypothesis.

Vapnik-Chervonekis (VC) Dimension

- N points can be labelled $r^t = 0/1$ in 2^N ways.
- $\mathcal{H}$ *shatters* N points if there exists $h \in \mathcal{H}$ consistent for all 2^N labellings.

Definition 10 (VC Dimension)**.** VC Dimension is the largest number N of points that can be shattered by $\mathcal{H}$.



Rectangles can shatter four points, $VC = 4$. Figure 2.5 of Alpaydin (2004).

PAC Bound using VC Dimension

Theorem 11. *We should have N training examples to have an probability of at least $1-\delta$ that any consistent h has error at most ϵ, where*

$$N \geq \frac{1}{\epsilon}\left(4\log_2\frac{2}{\delta} + 8VC(\mathcal{H})\log_2\frac{13}{\epsilon}\right).$$

- We can use the VC dimension instead of $\ln|H|$ as a measure of model complexity.
- Lesson: larger VC dimension, more complex model, more training samples are needed.
- (See Mitchell (1997), chapter 7, for details.)

What Did We Learn of PAC Learning and VC Dimension?

- Hypothesis class complexity (or model complexity) can be evaluated using the VC dimension.
- More complex model, more data you need to learn (learning is ability to describe the true hypothesis with a given confidence).
- PAC bounds are extremely conservative, in practice (when we also have noise) we usually need significantly smaller data sets.

4 Noise and Regression

4.1 Noise

Noise and Model Complexity

- *Noise* is unwanted anomaly of data.
- Because of the noise, we may never reach zero error.
- Noise may be caused by:
  - Errors in measurements of input attributes or class labels.
  - Unknown or ignored (hidden or latent) attributes.
- Noise is best treated probabilistically (next lectures).
- Why to use simpler model:
  - simpler to use
  - easier to train
  - easier to explain
  - generalizes better

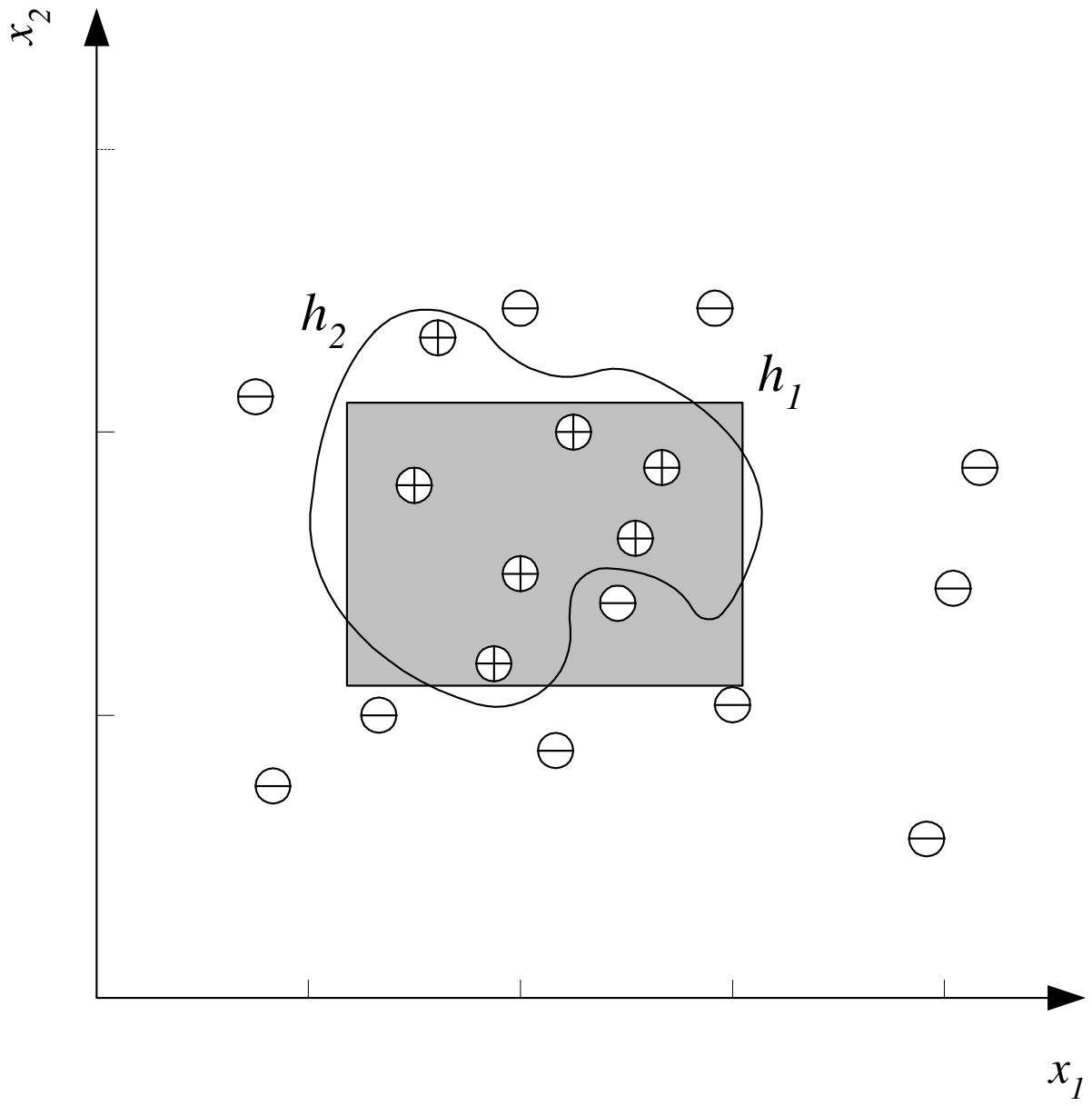


Figure 2.7 of Alpaydin (2004).

4.2 Regression

Regression

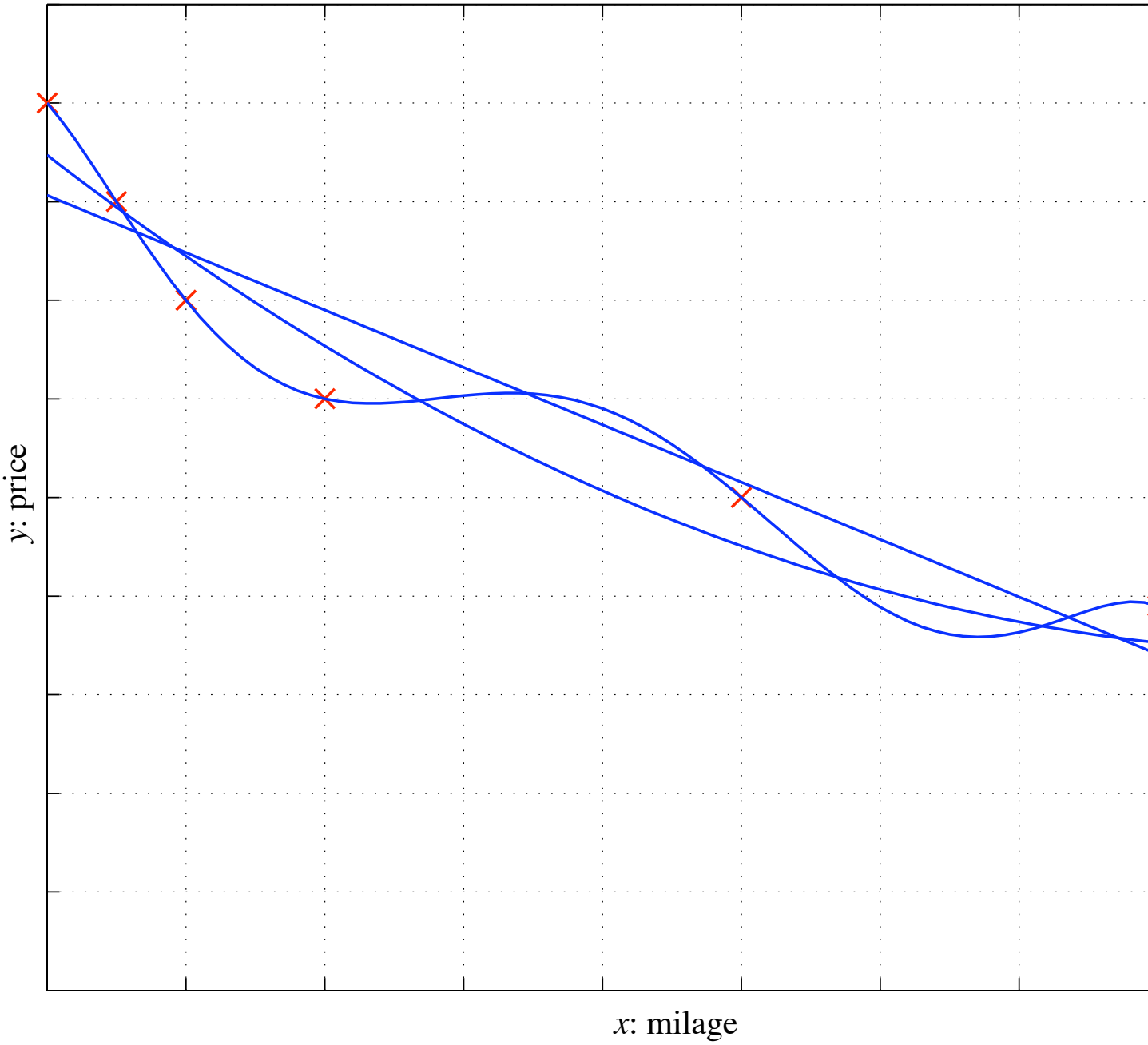


Figure 2.9 of Alpaydin (2004).

Regression

- Classification is the prediction of a 0–1 class, given attributes.
- *Regression* is the prediction of a real number, given attributes. (Usually with noise.)
- The training set is given by $\mathcal{X} = \left\{\mathbf{x}^t, r^t\right\}_{t=1}^{N}$, where $r^t \in \mathbb{R}$.

- We imagine that the r^t are given by some function $r^t = f(\mathbf{x}^t, \mathbf{z}^t)$, where $\mathbf{z}^t$ are some unknown hidden variables.
- The role of hypothesis is taken by the model $g(\mathbf{x})$. We would like to find a model such that $g(\mathbf{x}^t) \approx r^t$ for all items in the training set.
- Usually, we want to minimize a quadratic error function,

$$E(g \mid \mathcal{X}) = \frac{1}{N} \sum_{t=1}^{N} \left(r^t - g(\mathbf{x}^t)\right)^2.$$

Linear Regression

- The simplest case is linear regressor: $g(\mathbf{x}) = w_0 + w_1\mathbf{x}$.
- Optimization task: find w_0 and w_1 such that the error $E(g \mid \mathcal{X}) = \frac{1}{N}\sum_{t=1}^{N}\left(r^t - (w_0 + w_1\mathbf{x}^t)\right)^2$ is minimized.

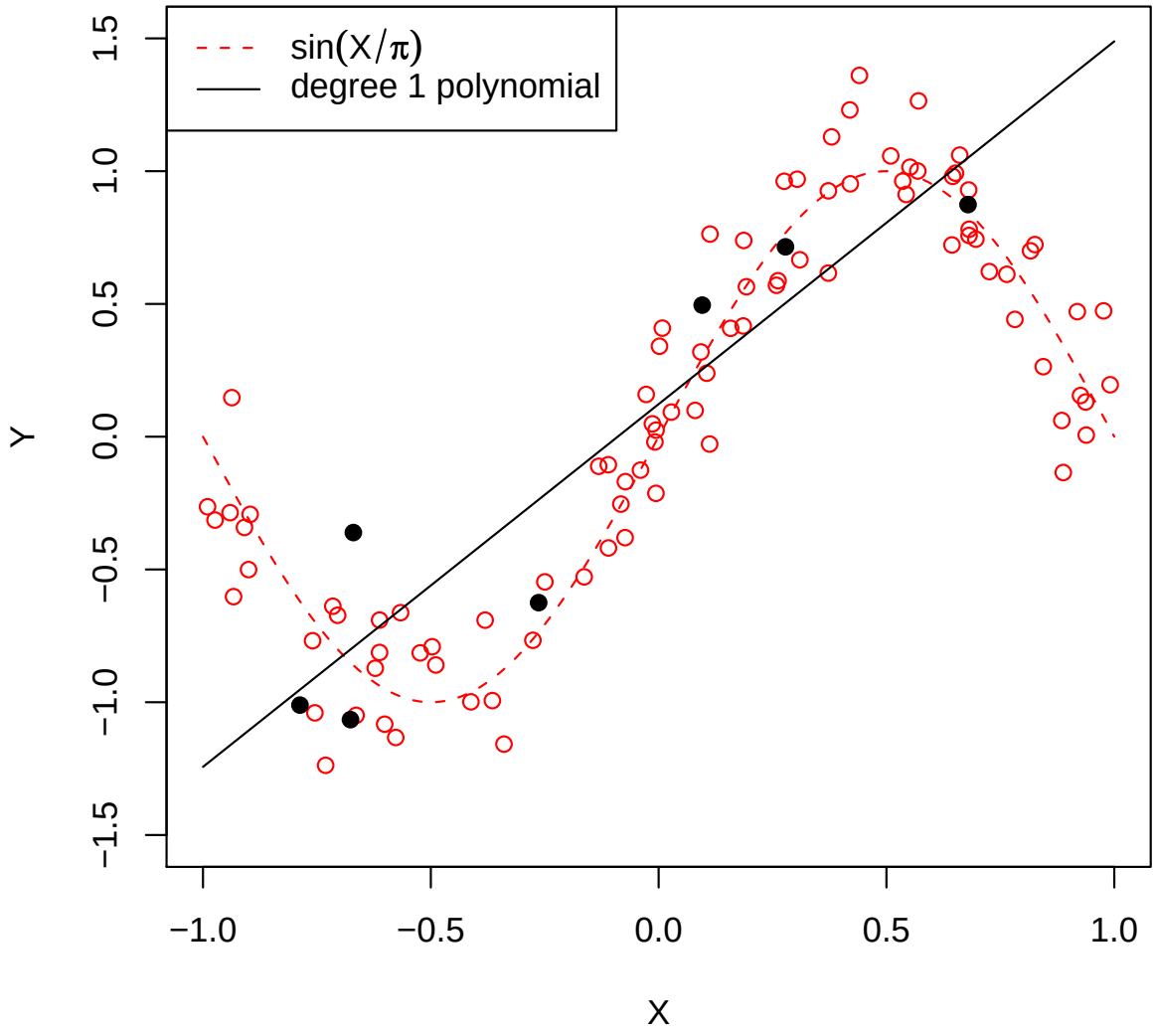
Analytic solution:

$$\begin{aligned} w_1 &= \frac{\sum_t x^t r^t - \overline{x}\overline{r}N}{\sum_t (x^t)^2 - N\overline{x}^2}, \\ w_0 &= \overline{r} - w_1\overline{x}, \end{aligned}$$

where $\overline{x} = \sum_t \mathbf{x}^t/N$ and $\overline{r} = \sum_t r^t/N$.

Linear Regression

- Toy data: we have generated 100 data points using $\sin(X/\pi)$ in interval $[-1, 1]$, added with Gaussian random noise.
- We randomly selected 7 data points to act as the training data (shown in black).
- Solution: $g(\mathbf{x}) = 0.12 + 1.37\mathbf{x}$.
- Error on training data: $E(g \mid \mathcal{X}) = 0.0032$.
- Error on the remaining 93 points: 0.21 (much larger than on training data!)



Linear Basis Functions

We can generalize linear regression using k *basis functions* $\phi_i(\mathbf{x})$,

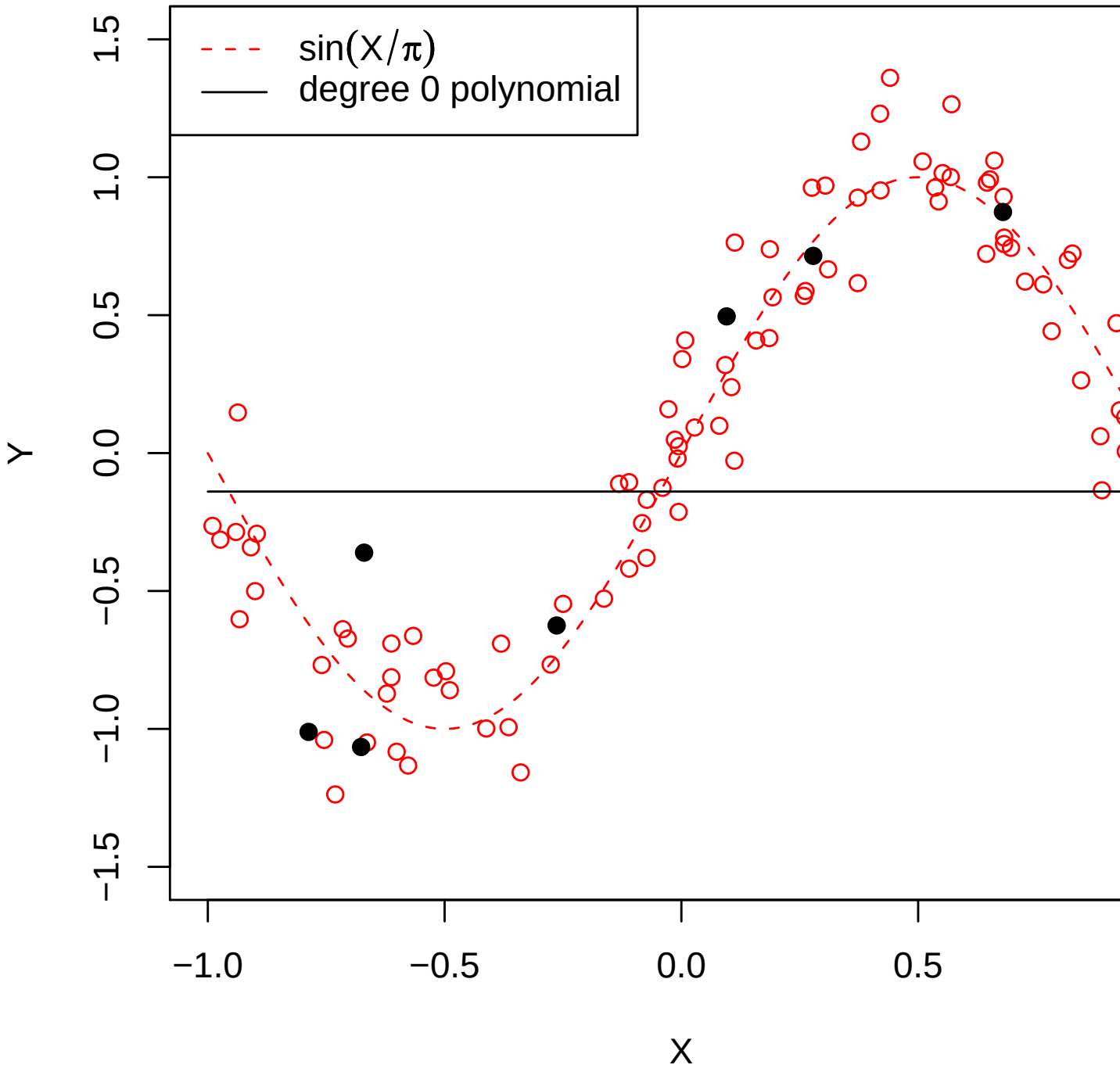
$$g(\mathbf{x}) = \sum_{i=0}^{k} w_i \phi_i(\mathbf{x}),$$

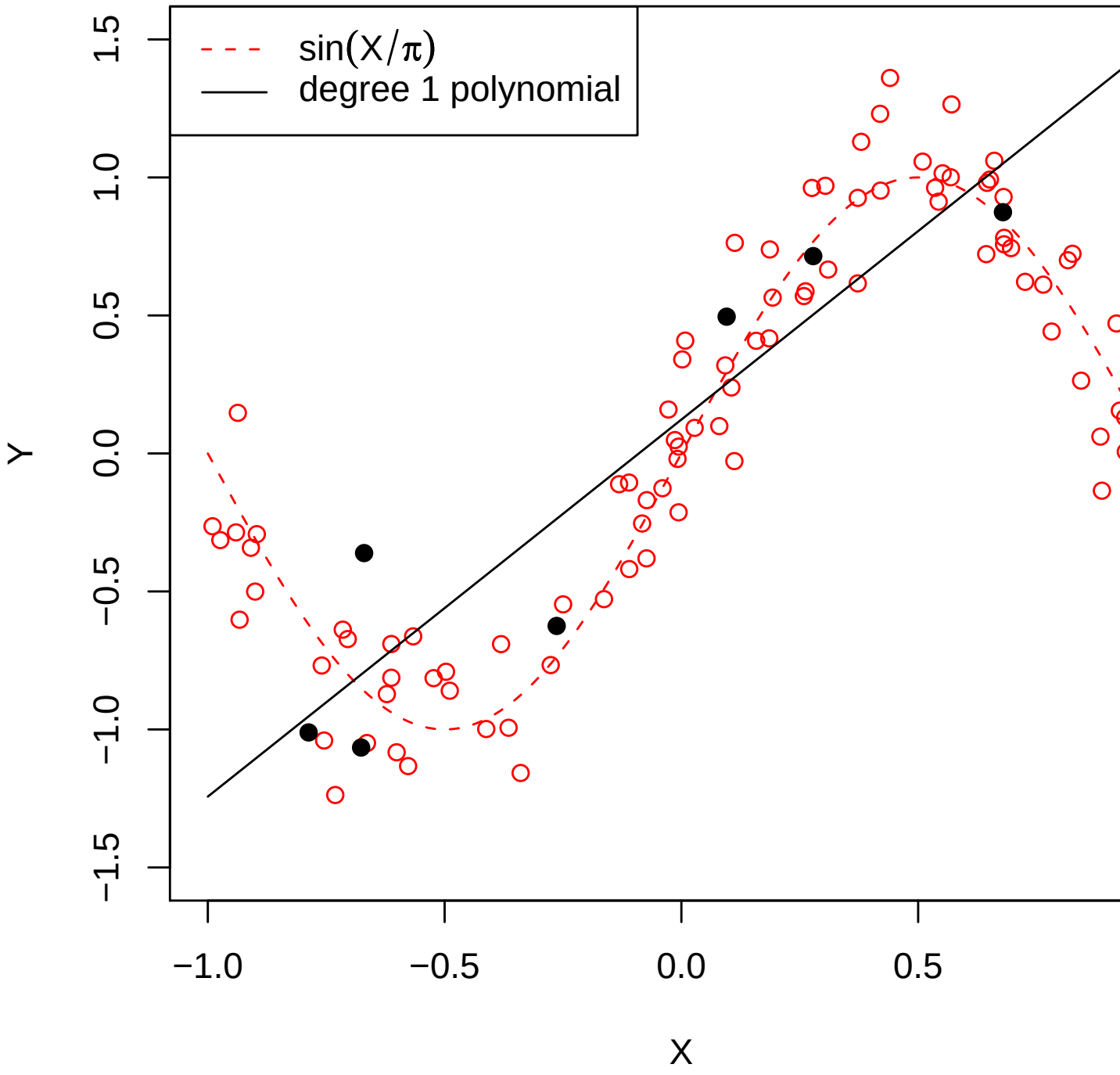
where usually $\phi_0(\mathbf{x}) = 1$.

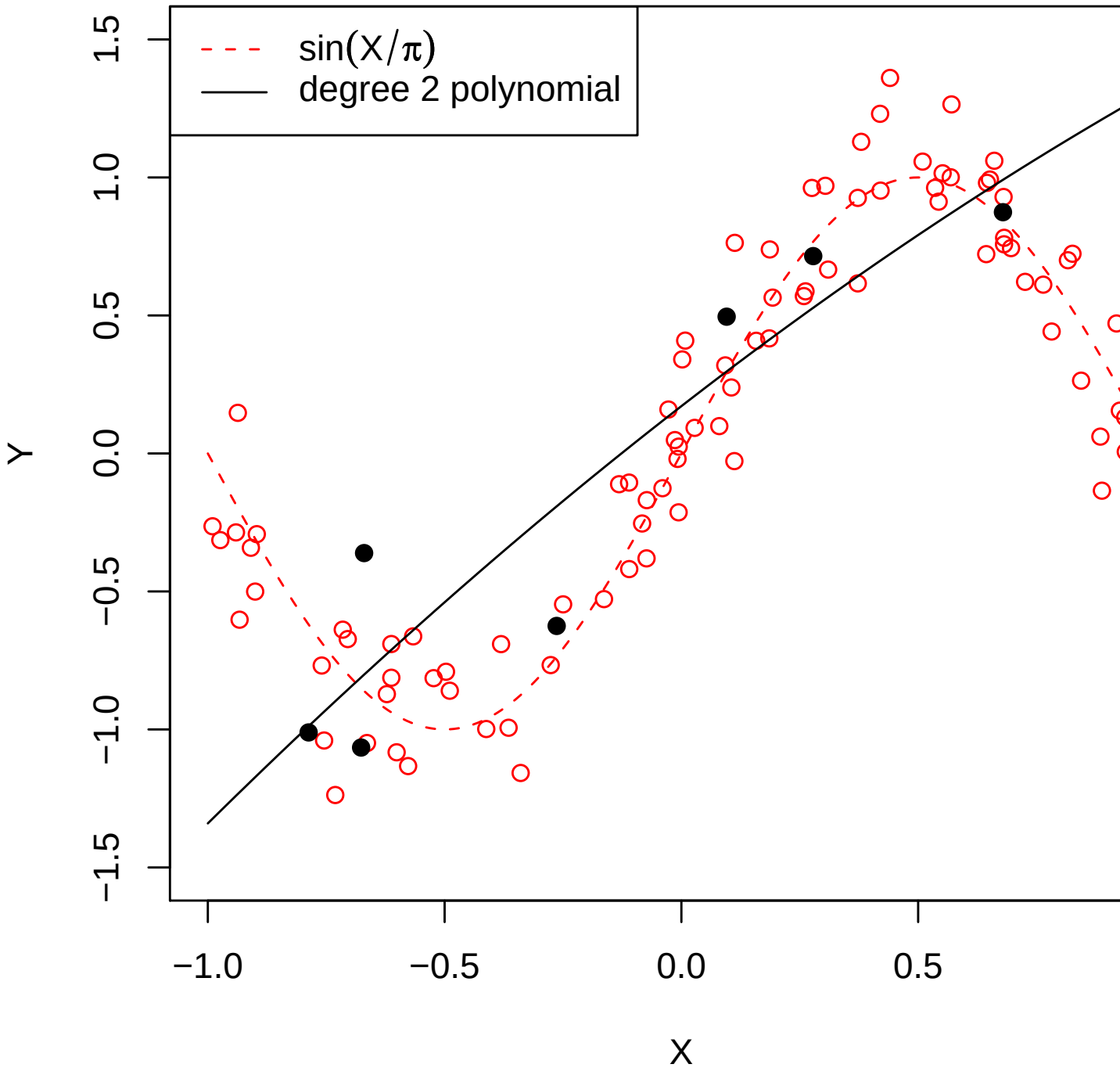
- A common choice: $\phi_i(\mathbf{x}) = \mathbf{x}^i$ (*polynomial basis*).
- $\phi_i(\mathbf{x}^t)$ can be computed beforehand and w_i can be solved using linear algebra.

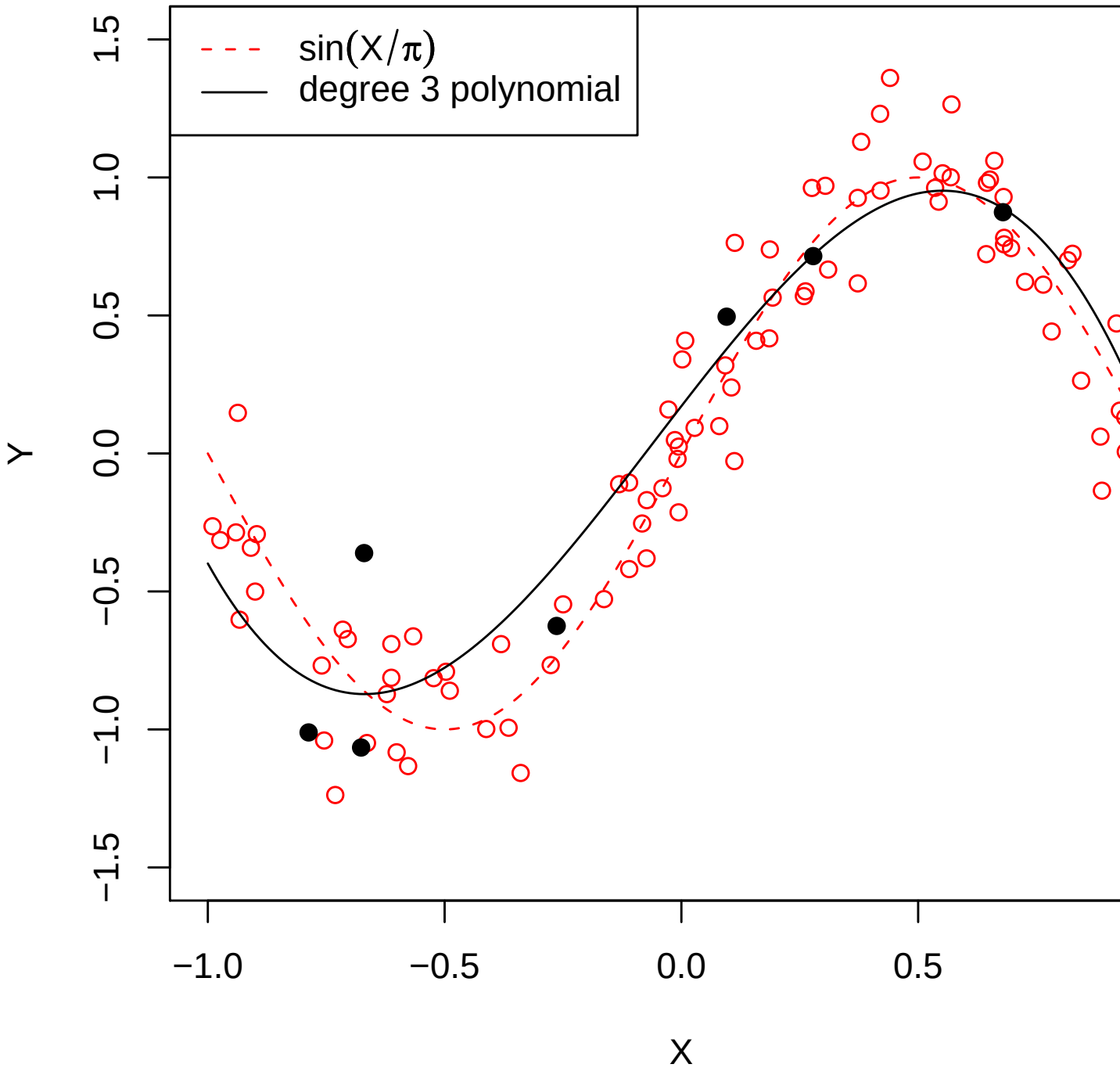
- In practice, there are lots of good software packages available that do the solving for you.
- Clearly, a high degree polynomial can represent a lower degree polynomial as a special case.
- Higher degree polynomial means larger hypothesis space or model family.

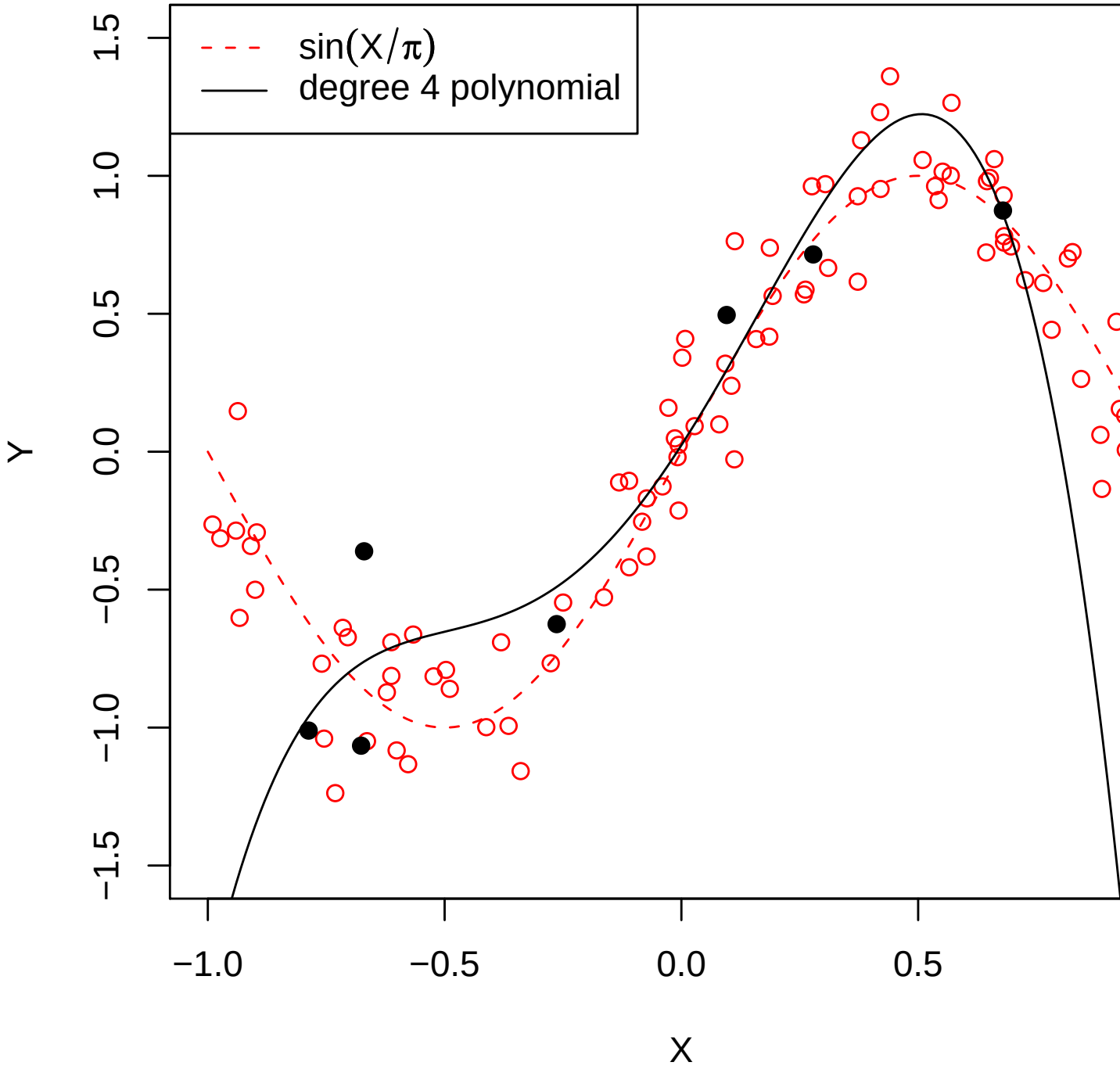
Polynomial Regressors

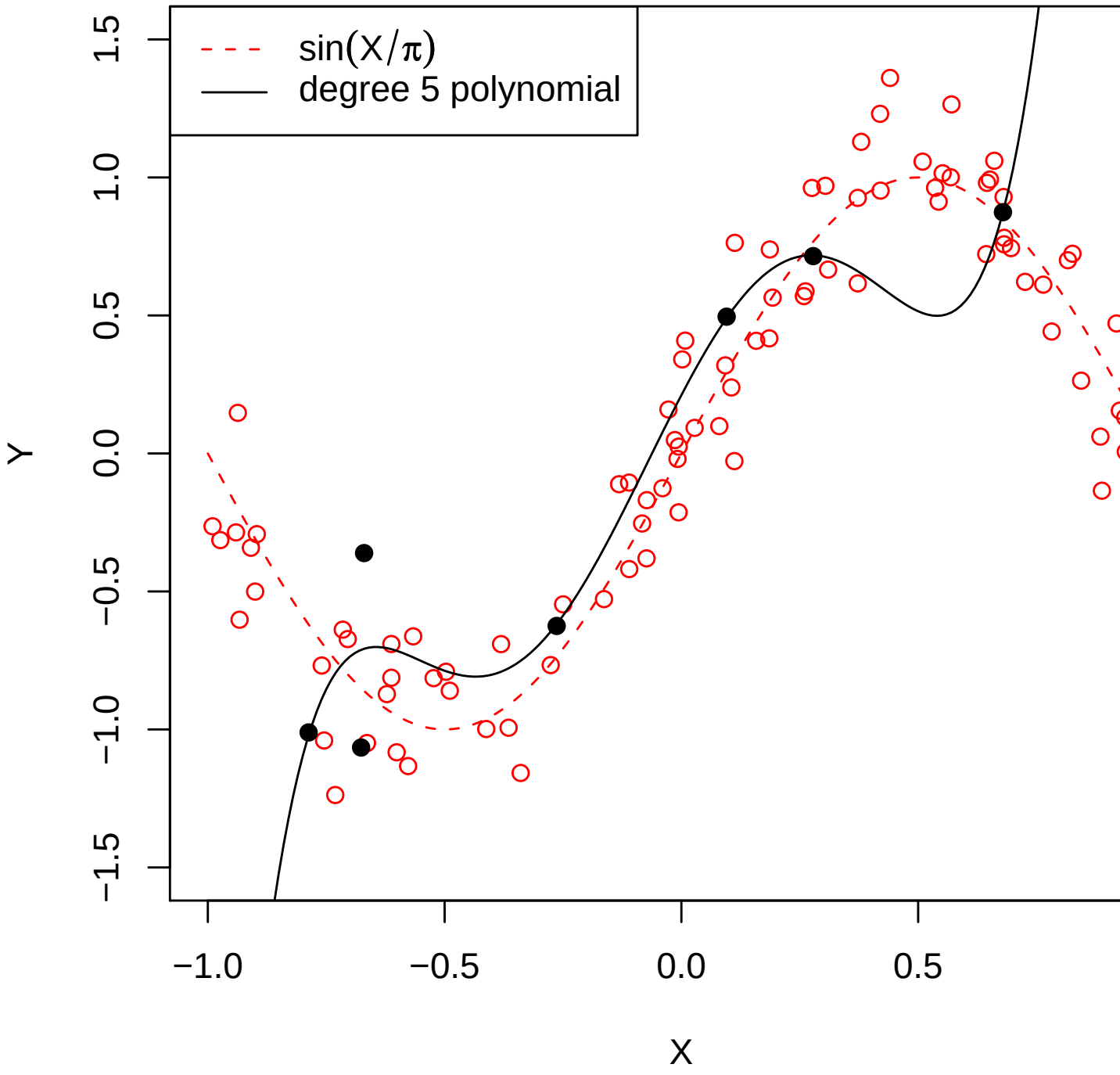

sin(X/π)
degree 0 polynomial
Y
X
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5

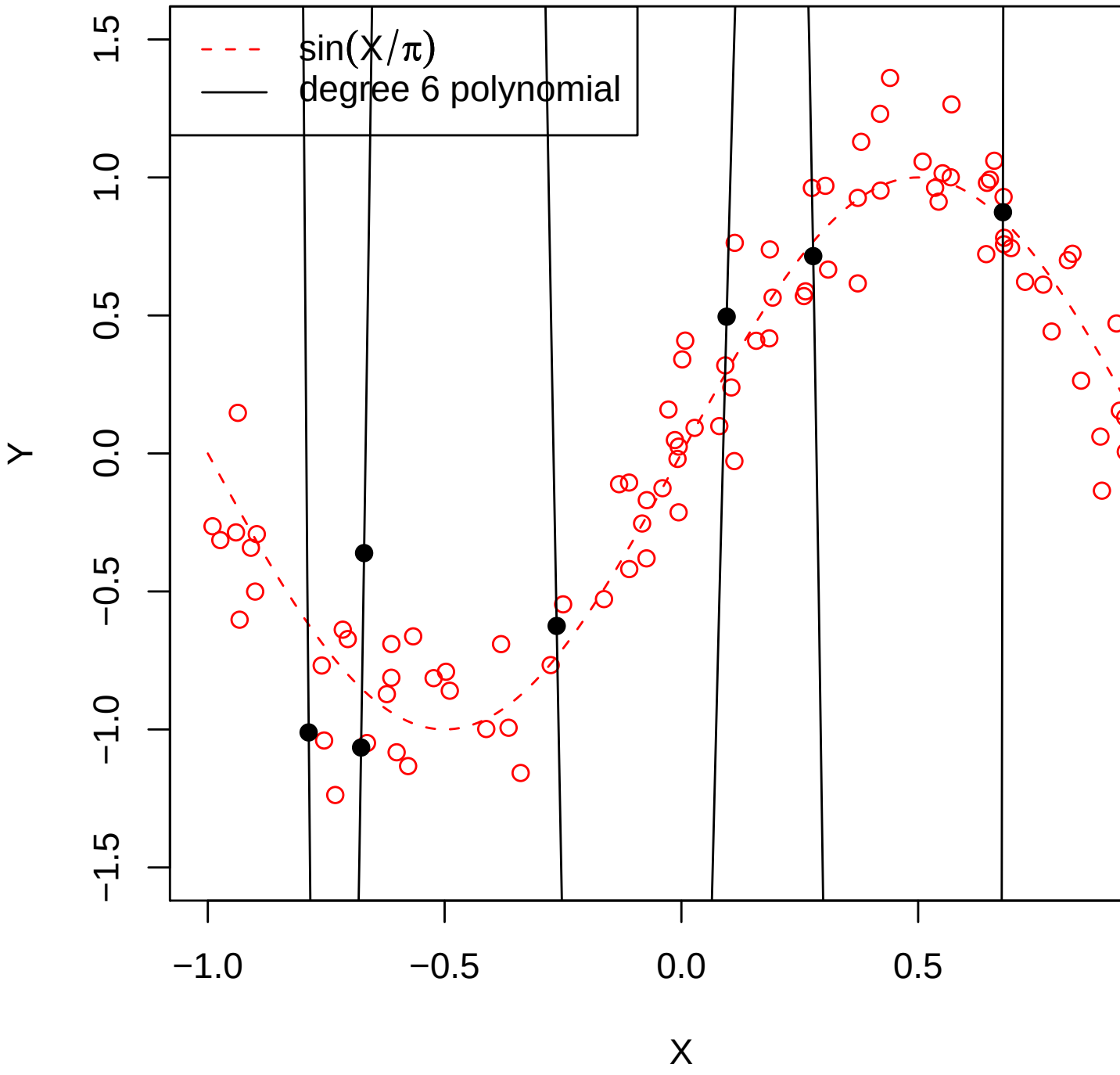

sin(X/π)
degree 1 polynomial
Y
X
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5


sin(X/π)
degree 2 polynomial
Y
X
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5


sin(X/π)
degree 3 polynomial
Y
X
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5


sin(X/π)
degree 4 polynomial
Y
X
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5


sin(X/π)
degree 5 polynomial
Y
X
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5


sin(X/π)
degree 6 polynomial
Y
X
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5

- E_{TRAIN} is the error in the training data. It decreases as model complexity increases.
- E_{TEST} is the error on the remaining 93 data points ("test set"). It has minimum at $k = 3$.

k	E_{TRAIN}	E_{TEST}	$g(x \mid w_0, \ldots, w_k) = \sum_{i=0}^{k} w_i X^i$
0	0.580	0.541	-0.14
1	0.077	0.294	$+0.12 + 1.37X$
2	0.076	0.275	$+0.17 + 1.33X - 0.18X^2$
3	0.057	0.057	$+0.17 + 2.22X - 0.35X^2 - 2.00X^3$
4	0.046	0.562	$+0.02 + 2.67X + 2.23X^2 - 3.19X^3 - 4.73X^4$
5	0.035	4.637	$+0.21 + 3.28X - 2.70X^2 - 11.88X^3 + 5.24X^4 + 15.82X^5$
6	0	10^6	$-5.86 + 57X + 186X^2 - 875X^3 - 1490X^4$ $+1634X^5 + 2412X^6$

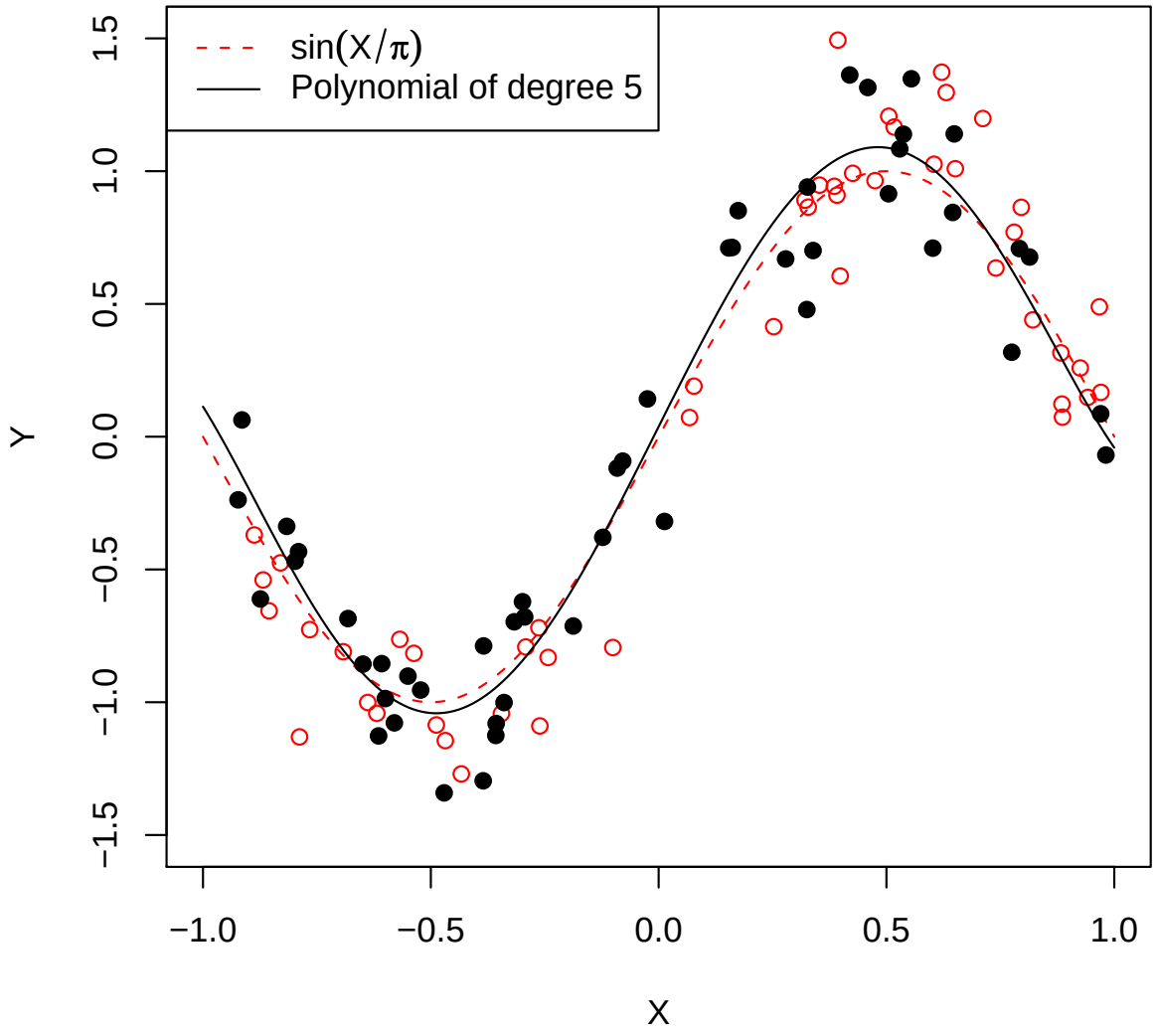
Table 4: Polynomial regressors.

N	E_{TRAIN}	E_{TEST}
7	0.0131	1.2187
10	0.0141	0.0821
15	0.0202	0.0761
20	0.0300	0.0511
25	0.0328	0.0507
30	0.0318	0.0573
35	0.0380	0.0494
40	0.0405	0.0484
45	0.0400	0.0476
50	0.0388	0.0473

Table 5: Effect of the size of the training data, $k = 5$.

Polynomial Regressors

Polynomial Regressors



4.3 Validation

Validation

- Error on training set:
  - Decreases as model becomes more complex.
  - Increases as number of data points grows.
- We want to minimize generalization error or error on test set:

  - Has a minimum at certain model complexity.
  - Decreases and approaches training set error as number of data points grows.
- How to minimize error on test set when we have no access to test set?

Validation

- *To estimate generalization error*, we need data unseen during training. We split the data in random as
  - training set (50%)
  - validation set (25%)
  - test set (25%)
- Train models of different complexities on training set. Pick a model complexity that gives smallest validation set error.
- Train model on combined training and validation set. Report test set error.

Validation

- We are given 20 points from our sinusoidal curve data set.
- Divide the data in random to training (10), validation (5) and test (5) sets.
- Train regressors of different complexities on training set:

k	E_{TRAIN}	E_{VALID}
0	0.492	0.644
1	0.091	0.125
2	0.090	0.137
3	0.044	**0.041**
4	0.044	0.049
5	0.042	0.142
6	0.030	18.820
7	0.025	181.850
8	0.024	34.014
9	0	10^9

- Validation set error is minimized for the degree 3 polynomial ($k = 3$). Pick degree 3 polynomial.

Validation

- Train degree 3 polynomial on 15 points (training+validation set) and report the results on the test set:

k	$E_{TRAIN+VALID}$	E_{TEST}
3	0.0378	0.0594

- If we would like to make predictions we should train on all 20 points (training+validation+test set). We know that the error on new data points should be approximately at most 0.0594.
- Training with all 20 points in fact gives slightly smaller error (0.0557) on 80 newly sampled data points.

5 Conclusion

5.1 About Supervised Learning

Model Selection and Generalization

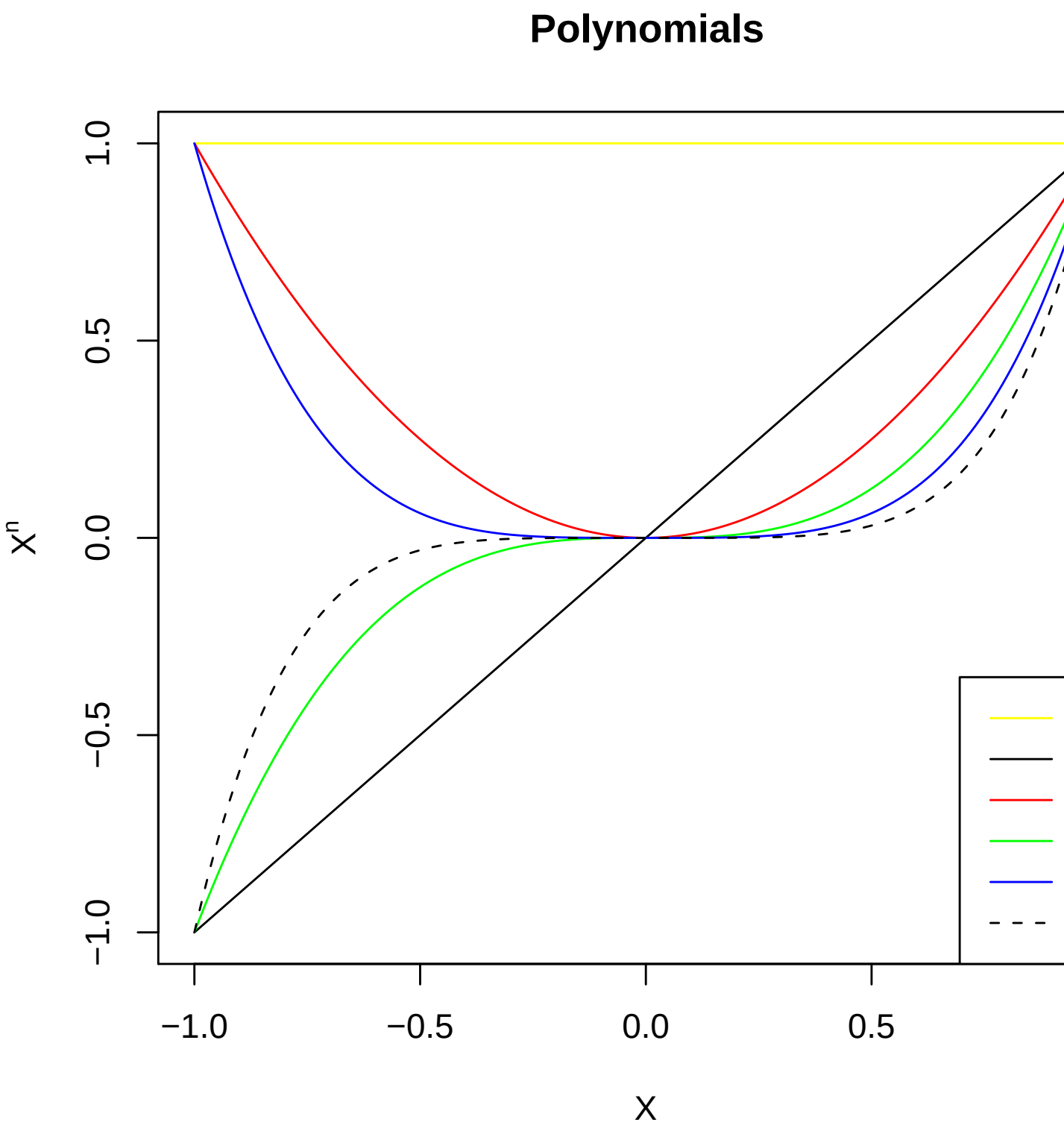
- Learning is *ill-posed problem*: data is not sufficient to find unique/correct solution.
- *Inductive bias* is needed; we need assumptions about the hypothesis class (model family) $\mathcal{H}$.
- *Generalization*: how well model performs on new data.
- Overfitting: $\mathcal{H}$ more complex than C or f.
- Underfitting: $\mathcal{H}$ less complex than C or f.
- Triple trade-off (Diettrich 2003):
  - complexity of $\mathcal{H}$;
  - amount of training data; and
  - generalization error on new data.

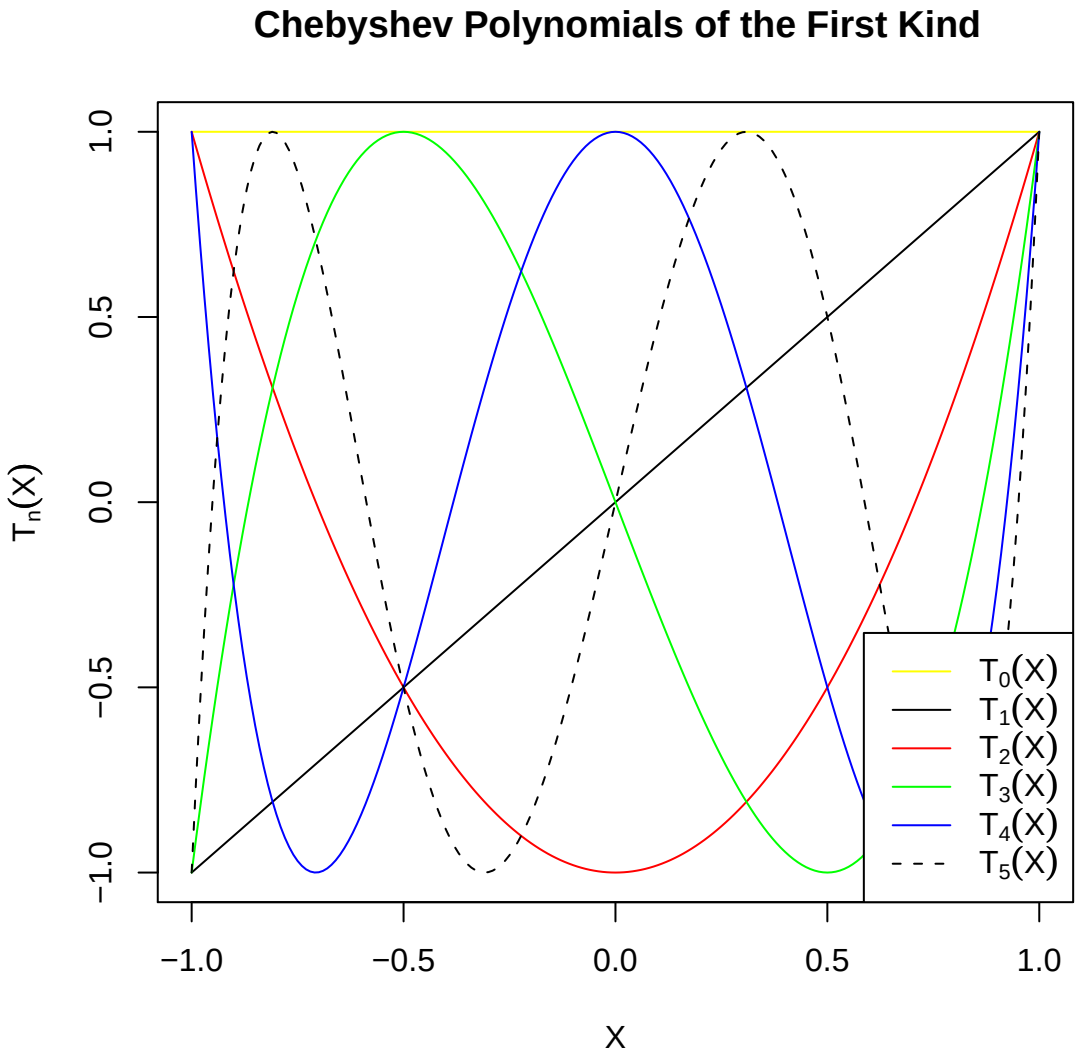
Dimensions of a Supervised Learner

1. Model: $g(\mathbf{x} \mid \theta)$.
2. Loss function: $E(\theta \mid \mathcal{X}) = \frac{1}{N} \sum_{t=1}^{N} L\left(r^t, g(\mathbf{x}^t \mid \theta)\right)$.
3. Optimization procedure: $\theta \leftarrow \arg\min_\theta E(\theta \mid \mathcal{X})$.

5.2 Better Basis Functions

Polynomial Basis


Polynomials
1.0
0.5
0.0
−0.5
−1.0
X^n
−1.0
−0.5
0.0
0.5
X



$T_0(X) = 1$ $\quad$ $T_3(X) = 4X^3 - 3X$
$T_1(X) = X$ $\quad$ $T_4(X) = 8X^4 - 8X^2 + 1$
$T_2(X) = 2X^2 - 1$ $\quad$ $T_5(X) = 16X^5 - 20X^3 + 5X$

Chebyshev Polynomials of the First Kind

- Chebyshev Polynomials are orthogonal polynomials in $X \in [-1, 1]$.
- Def.: $T_n(\cos\theta) = \cos n\theta$, $n \in \{0, 1, \ldots\}$.
- Recurrence relation: $T_{n+2}(x) = 2xT_{n+1}(x) - T_n(x)$.
- Chebyshev Polynomials are useful in numerical analysis:
  - $\max T_n(x) = +1$, $\min T_n(x) = -1$. (X^n basis also satisfies this in $X \in [-1, 1]$.)

k	E_{TRAIN}	E_{TEST}	$g(x \mid w_0, \ldots, w_k) = \sum_{i=0}^{k} w_i T_i(X)$
0	0.580	0.541	$-0.14T_0(X)$
1	0.077	0.294	$+0.12T_0(X) + 1.37T_1(X)$
2	0.076	0.275	$+0.08T_0(X) + 1.33T_1(X) - 0.09T_2(X)$
3	0.057	0.057	$-0.01T_0(X) + 0.72T_1(X) - 0.18T_2(X) - 0.50T_3(X)$
4	0.046	0.562	$-0.64T_0(X) + 0.28T_1(X) - 1.25T_2(X) - 0.80T_3(X)$ $-0.59T_4(X)$
5	0.035	4.637	$+0.83T_0(X) + 4.26T_1(X) + 1.27T_2(X) + 1.97T_3(X)$ $+0.65T_4(X) + 0.99T_5(X)$
6	0	10^6	$+282.4T_0(X) + 422.6T_1(X) + 478.9T_2(X) + 291.8T_3(X)$ $+266.0T_4(X) + 102.1T_5(X) + 75.3T_6(X)$

Table 6: Chebyshev regressors; compare the magnitude of the terms to the X^n basis.

$$
\begin{array}{ll}
T_0(X) = 1 & T_3(X) = 4X^3 - 3X \\
T_1(X) = X & T_4(X) = 8X^4 - 8X^2 + 1 \\
T_2(X) = 2X^2 - 1 & T_5(X) = 16X^5 - 20X^3 + 5X \\
 & T_6(X) = 32X^6 - 48X^4 + 18X^2 - 1
\end{array}
$$

Table 7: Chebyshev Polynomials of the First Kind.

- The maxima and minima are spread reasonably uniformly over $[-1, 1]$. (Comparing, in X^n basis the maxima and minima are only in $X = -1$ and $X = +1$.)
- In least squares regression, the Chebyshev basis is analytically equivalent but numerically much more robust than the commonly used X^n basis especially for larger (> 10) degrees.

$$T_0(X) = 1;\ T_1(X) = X;\ T_2(X) = 2X^2 - 1;\ T_3(X) = 4X^3 - 3X;\ T_4(X) = 8X^4 - 8X^2 + 1;$$
$$T_5(X) = 16X^5 - 20X^3 + 5X;\ \ldots$$

Conclusion

- No problem session this week, next problem session on 28 September.
- This week's problem sheet contains a small data analysis task (for 28 September). [Will be in the web later today, hopefully.]
- Next lecture on 25 September: Bayesian Decision Theory, Alpaydin (2004) Ch 3.

6 Supervised Learning

6.1 Elements of a Learner

Dimensions of a Supervised Learner

Model

$$g(\mathbf{x} \mid \theta)$$

Loss Function

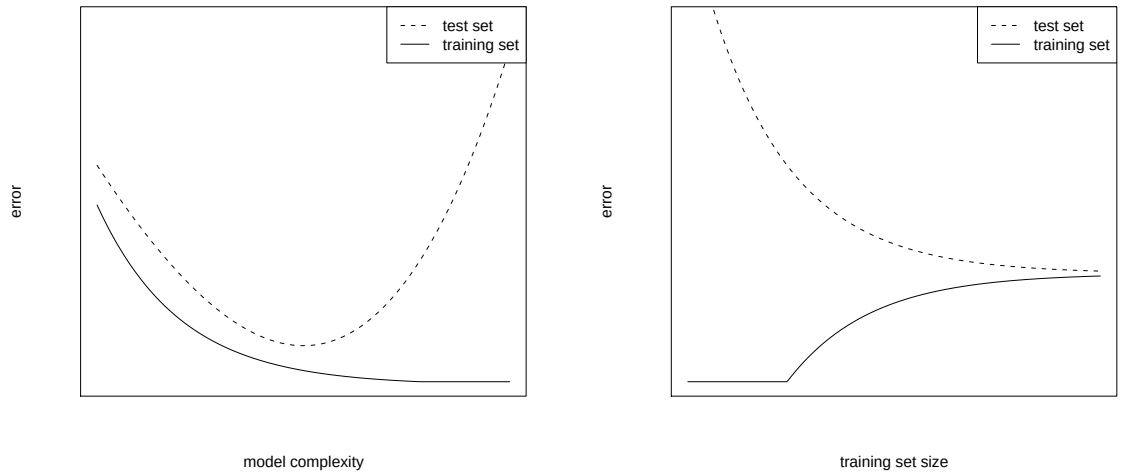
$$E\left(\theta \mid \mathcal{X}\right) = \frac{1}{N}\sum_{t=1}^{N} L\left(r^t, g(\mathbf{x}^t \mid \theta)\right).$$

Optimization Procedure

$$\theta \leftarrow \arg\min_{\theta} E\left(\theta \mid \mathcal{X}\right).$$

6.2 Generalization

Model Selection and Generalization



- empirical error = error on training set
- generalization error = error on test set
- We see empirical error, but want to minimize the error on new data.

Validation

Question 1
What is the correct model complexity?

Question 2
What is the generalization error?

- To answer the Question 1 divide the data into training and validation sets. Choose model complexity that has the smallest error on the validation set.
- To answer the Question 2 divide the data into training and test sets. The generalization error is approximately the error on the test set.

- To answer both questions the data should be divided into training, validation and test sets.
- There are more efficient methods, such as cross-validation.

Model Selection and Generalization

- Learning is *ill-posed problem*: data is not sufficient to find unique/correct solution.
- *Inductive bias* is needed; we need assumptions about the hypothesis class (model family) $\mathcal{H}$.
- *Generalization*: how well model performs on new data.
- Overfitting: $\mathcal{H}$ more complex than C or f.
- Underfitting: $\mathcal{H}$ less complex than C or f.
- Triple trade-off (Dietterich 2003):
  - complexity of $\mathcal{H}$;
  - amount of training data; and
  - generalization error on new data.

7 Bayesian Decision Theory

7.1 Probabilities

Basic of Probability

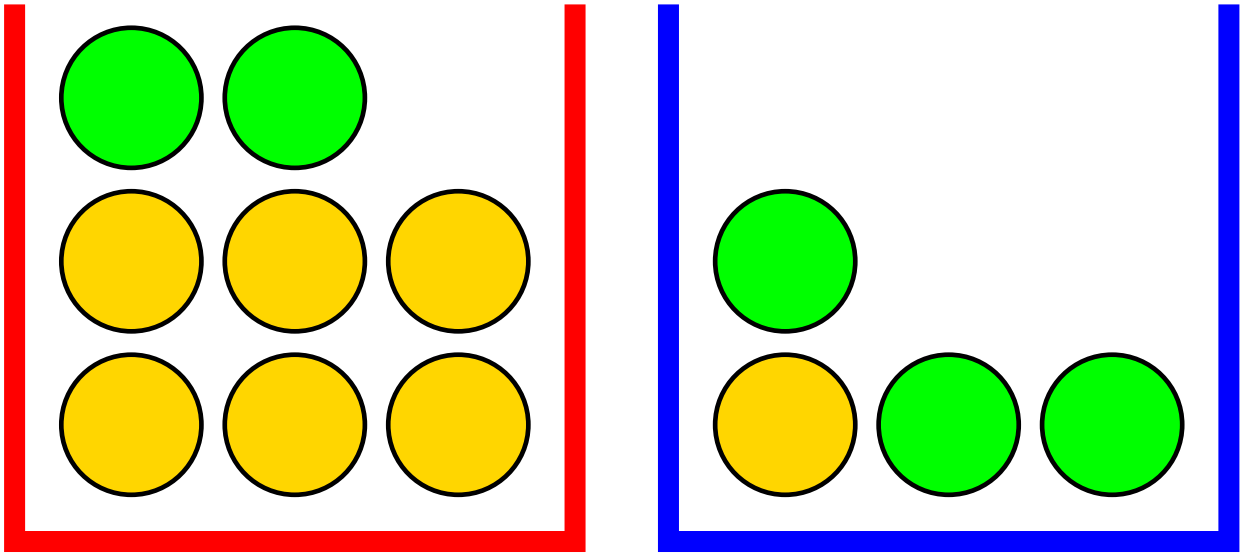
- You should know basics of probability (Mat-1.2600/2620 or Appendix A of Alpaydin (2004)).
- Probability can be interpreted as a *frequency* or *degree of belief*.
- *Sample space* S: the set of all possible outcomes.
- *Event* $E \subseteq S$: one possible set of outcomes.
- *Probability measure* P satisfies:
  - $P(S) = 1$.
  - $0 \leq P(E) \leq 1$ for all $E \subseteq S$.
  - $E \subseteq S \wedge F \subseteq S \wedge E \cap F = \emptyset \Rightarrow P(E \cup F) = P(E) + P(F)$.

Rules of Probability

- Interpret E, F as *random variables* getting values of e, f (coin tossing example: E can get a value of $e \in \{\text{heads}, \text{tails}\}$, F can get a value of coin landing in $f \in \{\text{table}, \text{floor}\}$).
- $P(E, F) = P(F, E)$: probability of both E and F happening.
- $P(E) = \sum_F P(E, F)$ (sum rule, marginalization)
- $P(E, F) = P(F \mid E)P(E)$ (product rule, conditional probability)
- Consequence: $P(F \mid E) = P(E \mid F)P(F)/P(E)$ (Bayes' formula)
- We say E and F are *independent* if $P(E, F) = P(E)P(F)$ (for all e and f).
- We say E and F are *conditionally independent* given G if $P(E, F \mid G) = P(E \mid G)P(F \mid G)$, or equivalently $P(E \mid F, G) = P(E \mid G)$.

Fruits in Boxes

- $P(B = r, F = a) = n_{RA}/n = 1/6$.
- $P(B = r) = \sum_{x \in \{a,o\}} P(B = r, F = x) = n_{RA}/n + n_{RO}/n = n_R/n = 2/3$.
- $P(F = o \mid B = r) = n_{RO}/n_R = 3/4$.
- $P(B = r \mid F = o) = P(F = o \mid B = r)P(B = r)/P(F = o) = \frac{3}{4} \times \frac{8}{12} \times \frac{12}{7} = \frac{6}{7}$.



	apples	**o**ranges	Σ
red box	$n_{RA} = 2$	$n_{RO} = 6$	$n_R = 8$
blue box	$n_{BA} = 3$	$n_{BO} = 1$	$n_B = 4$
Σ	$n_A = 5$	$n_O = 7$	$n = 12$

Table 8: Count of fruits in two boxes.

Fruits in Boxes

- B and F are *random variables* which can take two values (r or b; a or o, respectively).
- We computed probabilities of events of drawing one fruit in random such that the probability of drawing each fruit is 1/12, independent of the box or type.
- We viewed the probabilities as *frequencies*.
- When all prior information (e.g., counts of the fruits in the boxes) is not known the probabilities turn into *degrees of belief* (it may be still easier to think them as frequencies, though).

Estimating Probability

- In real life, estimating the probabilities of various events from a sample is difficult.
- For the purposes of today, we mostly assume that someone gives us the probabilities.
- Today we can estimate the probabilities with sample frequencies.
  - Example: Someone is tossing a 0–1 coin that gives $X = 1$ with probability $P(X = 1) = p$ and $X = 0$ with probability $P(X = 0) = 1 - p$ (Bernoulli distribution). We notice he got n_1 ones and n_0 zeroes in a *sample* of $N = n_1 + n_0$ tosses. Based on this sample, we can estimate p with $\hat{p} = n_1/N$.

7.2 Classification

Using Probabilities Classification

- Someone is tossing a 0–1 coin that gives $X = 1$ (HEADS) with probability $P(X = 1) = p$ and $X = 0$ (TAILS) with probability $P(X = 0) = 1 - p$ (Bernoulli distribution).
- Task: make a classifier for the next toss.
- Prediction: Choose $X = 1$ (HEADS) if $p \geq 1/2$, $X = 0$ (TAILS) otherwise.

Using Probabilities in Classification

- Task: classify a customer HIGH RISK ($C = 1$) or LOW RISK ($C = 0$) based on her income (x_1) and savings (x_2).
- Assume $P(C \mid x_1, x_2)$ is known.

Prediction:

$$\text{choose} \begin{cases} C = 1 & \text{if } P(C = 1 \mid x_1, x_2) \geq \frac{1}{2}, \\ C = 0 & \text{otherwise.} \end{cases}$$

or equivalently

$$\text{choose} \begin{cases} C = 1 & \text{if } P(C = 1 \mid x_1, x_2) \geq P(C = 0 \mid x_1, x_2), \\ C = 0 & \text{otherwise.} \end{cases}$$

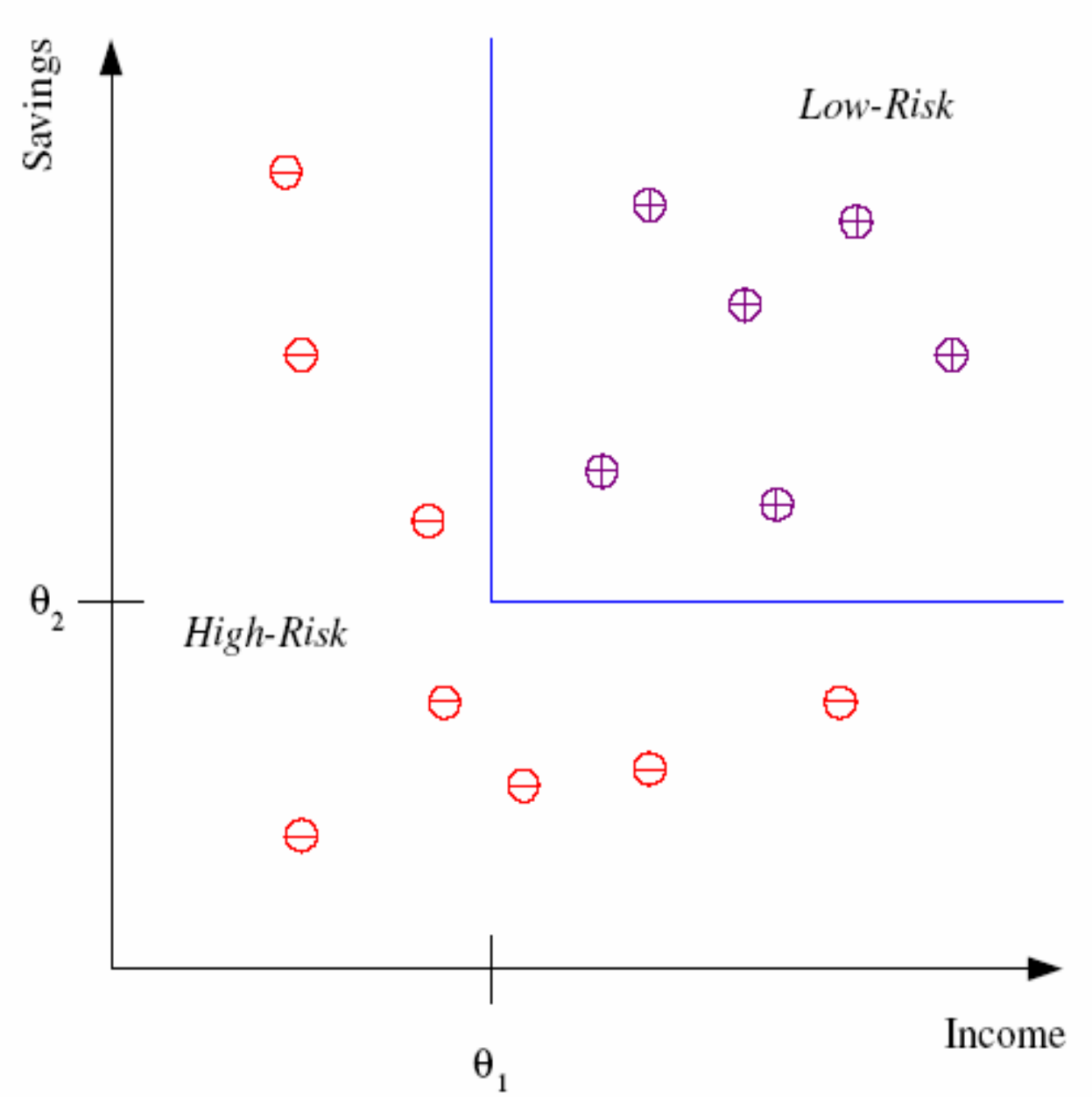


Figure 1.1 of Alpaydin (2004).

Bayes' Rule

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}},$$

or

$$P(C \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid C) \times P(C)}{P(\mathbf{x})}.$$

- The *likelihood* $P(\mathbf{x} \mid C = 1)$ is the probability that a HIGH RISK customer ($C = 1$) has the associated observed value $\mathbf{x}$. (This is usually easy to compute.)
- The *prior probability* $P(C = 1)$ is the probability of observing $C = 1$ (before $\mathbf{x}$ is known).
- The *evidence* $P(\mathbf{x})$ is the marginal probability that an observation $\mathbf{x}$ is seen, regardless of the value of C. (This is usually difficult to compute directly.)

Using the sum and product rules we obtain:

- $P(C = 0) + P(C = 1) = 1$.
- $P(C = 0 \mid \mathbf{x}) + P(C = 1 \mid \mathbf{x}) = 1$.
- $P(\mathbf{x}) = P(\mathbf{x} \mid C = 1)P(C = 1) + P(\mathbf{x} \mid C = 0)P(C = 0)$.

Bayes' Rule

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

- $P(C_k) \geq 0$ and $\sum_{k=1}^{K} P(C_k) = 1$.
- *Naive Bayes Classifier*: choose C_k where $k = \arg\max_k P(C_k \mid \mathbf{x})$.
- A customer is associated with vector $\mathbf{x}$ such that $P(\mathbf{x} \mid C = 1) = 0.002$ and $P(\mathbf{x} \mid C = 0) = 0.001$.
- 20% of the customers are HIGH RISK ($C = 1$), we therefore set the prior probabilities to $P(C = 1) = 0.2$ and $P(C = 0) = 0.8$.
- Inserting in equation we obtain $P(C = 1 \mid \mathbf{x}) = 0.33$ and $P(C = 0 \mid \mathbf{x}) = 0.67$, we therefore classify the customer as LOW RISK ($C = 0$).

7.3 Utility Theory

Risks and Losses

- Often, the cost of errors differs. For example, a wrong decision to grant credit may be much more costly than a wrong decision not to grant credit.
- Decision theory: how to make optimal decisions, given all available information.
- At each time, you can choose one *action* α_i.

- Action α_i causes *loss* λ_{ik} when the state is C_k.

λ	$C = 0$	$C = 1$
$\alpha_0 =$ grant credit	EUR 0	EUR 1000
$\alpha_1 =$ don$'$t grant credit	EUR 100	EUR 0

- *Expected risk*: $R(\alpha_i \mid \mathbf{x}) = E\left[\lambda_{ik}\right] = \sum_{k=1}^{K} \lambda_{ik} P(C_k \mid \mathbf{x})$.
- Choose α_i where $i = \arg\min_i R(\alpha_i \mid \mathbf{x})$.

Risks and Losses

- *0/1 loss*:

$$\lambda_{ik} = \begin{cases} 0 & i = k \\ 1 & i \neq k \end{cases}$$

$$\begin{aligned} R(\alpha_i \mid \mathbf{x}) &= \sum_{k=1}^{K} \lambda_{ik} P(C_k \mid \mathbf{x}) \\ &= \sum_{k \neq i} P(C_k \mid \mathbf{x}) \\ &= 1 - P(C_i \mid \mathbf{x}). \end{aligned}$$

For minimum risk, choose the most probable class.

Risks and Losses

- Assume mis-classification has a cost of 1 (0/1 loss).
- Assume (almost) certain classification (e.g., by a human expert) has a cost of λ.
- Define additional action REJECT α_{K+1} and loss by

$$\lambda_{ik} = \begin{cases} 0 & i = k \\ \lambda & i = K+1 \\ 1 & \text{otherwise} \end{cases}.$$

- $R(\alpha_{K+1} \mid \mathbf{x}) = \sum_{k=1}^{K} \lambda P(C_k \mid \mathbf{x}) = \lambda$.
- $R(\alpha_i \mid \mathbf{x}) = \sum_{k \neq i} P(C_k \mid \mathbf{x}) = 1 - P(C_i \mid \mathbf{x})$.

$$\text{Choose} \begin{cases} C_k & \text{if} \quad k = \arg\max_k P(C_k \mid \mathbf{x}) \text{ and } P(C_k \mid \mathbf{x}) \geq 1 - \lambda \\ \text{reject} & \text{otherwise} \end{cases}$$

Discriminant Functions

- *Discriminant function*: choose α_i where $i = \arg\max_k g_k(\mathbf{x})$, where

$$g_k(\mathbf{x}) = \begin{cases} -R(\alpha_k \mid \mathbf{x}) \\ P(C_k \mid \mathbf{x}) \\ p(\mathbf{x} \mid C_k)P(C_k) \end{cases}$$

- K decision regions $\mathcal{R}_1, \ldots, \mathcal{R}_K$:

$$\mathcal{R}_i = \left\{ \mathbf{x} \mid i = \arg\max_k g_k(\mathbf{x}) \right\}.$$

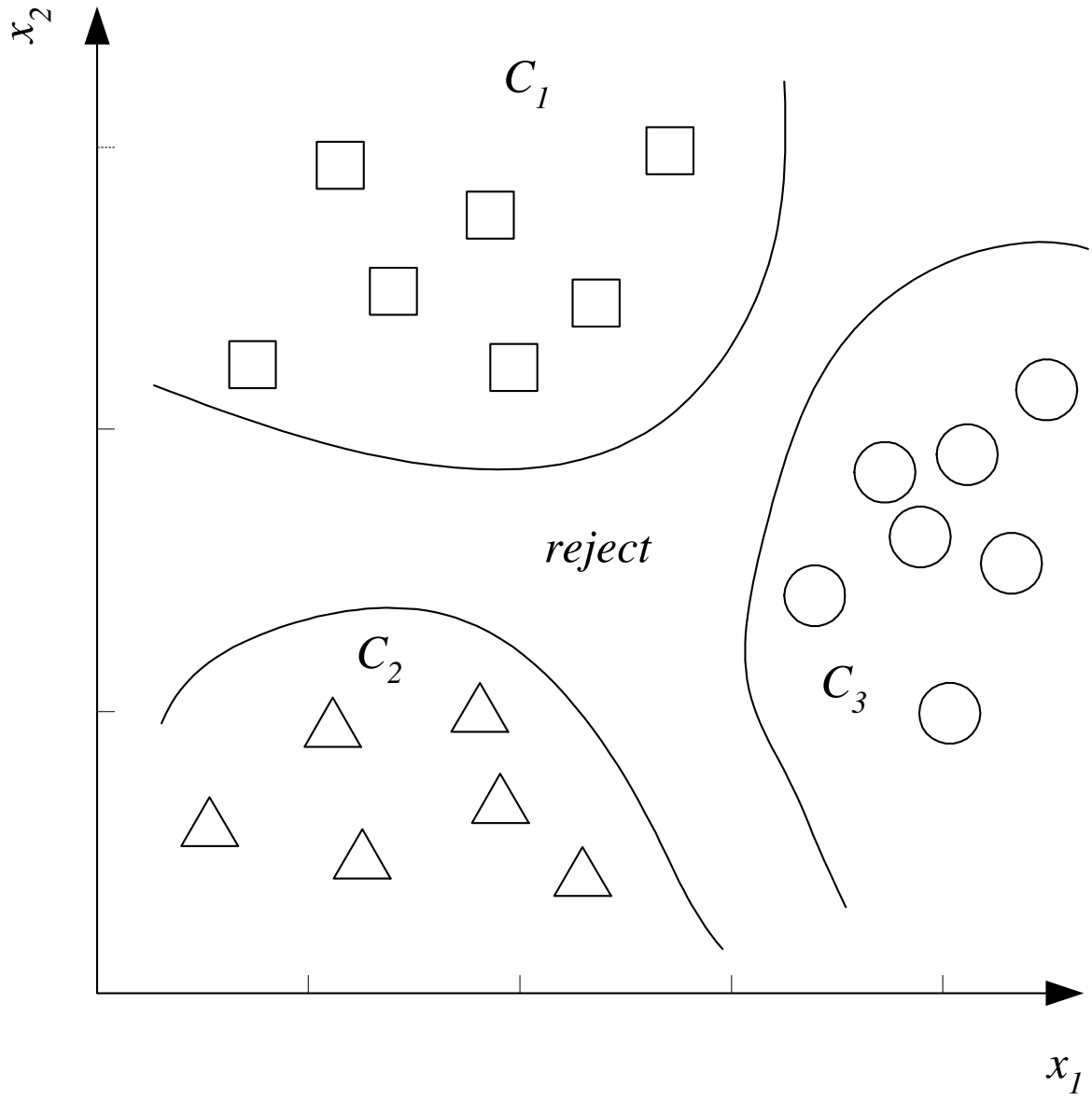


Figure 3.1 of Alpaydin (2004).

Discriminant Functions

- Dichtotomizer ($K = 2$) vs. Polychotomizer ($K > 2$)
- $g(\mathbf{x}) = g_1(\mathbf{x}) - g_2(\mathbf{x})$: choose C_1 if $g(\mathbf{x}) \geq 0$, C_2 otherwise.
- *Log odds*:

$$g(\mathbf{x}) = \log \frac{P(C_1 \mid \mathbf{x})}{P(C_2 \mid \mathbf{x})}.$$

Utility Theory

- In utility theory, one usually tries to maximize *expected utility* (instead of minimize risk).
- *Utility* of α_i when state is k: U_{ik}

$$EU(\alpha_i \mid \mathbf{x}) = E\left[U_{ik}\right] = \sum_k U_{ik} P(C_k \mid \mathbf{x}).$$

- Choose α_i where $i = \arg\max_i EU(\alpha_i \mid \mathbf{x})$.
- (Choosing $U_{ik} = \delta_{ik} \log P(C_k \mid \mathbf{x})$ makes utility equal to information and leads to probabilistic modeling.)

Utility Theory

- Utility of using $\mathbf{x}$ only is $EU(\mathbf{x}) = \max_i EU(\alpha_i \mid \mathbf{x})$.
- Utility of using $\mathbf{x}$ and new feature z is $EU(\mathbf{x}, z) = \max_i EU(\alpha_i \mid \mathbf{x}, z)$.
- z is useful if $EU(\mathbf{x}, z) > EU(\mathbf{x})$.
- You should probably measure z if the expected gain in utility, $EU(\mathbf{x}, z) - EU(\mathbf{x})$ exceeds the measurement costs.

Decision Theory in Court

- Classification problem GUILTY vs. NOT GUILTY.
- Typically, DNA evidence has small match probabilities. How should it be combined with other evidence?
- Sentencing innocent should have a higher loss.
- R v. Denis John Adams.

Instructions to the Jury?

Suppose the match probability is 1 in 20 million. That means that in Britain (population about 60 million) there will be on average about 2 or 3 people, and certainly no more than 6 or 7, whose DNA matches that found at the crime scene, in addition to the accused. Now your job, as a member of the jury, is to decide on the basis of the other evidence, whether or not you are satisfied that it is the person on trial who is guilty, rather than one of the few other people with matching DNA. We don't know anything about the other matching people. They are likely to be distributed all across the country and may have been nowhere near the crime scene at the time of the crime. Others may be ruled out as being the wrong sex or the wrong age group.

	$\mathbf{x}^t$						$r(\mathbf{x}^t)$
t	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	1
2	Sunny	Warm	High	Strong	Warm	Same	1
3	Rainy	Cold	High	Strong	Warm	Change	0
4	Sunny	Warm	High	Strong	Cool	Change	1

Table 9: Aldo's observed sport experiences in different weather conditions.

8 Bayesian Networks

8.1 Basics

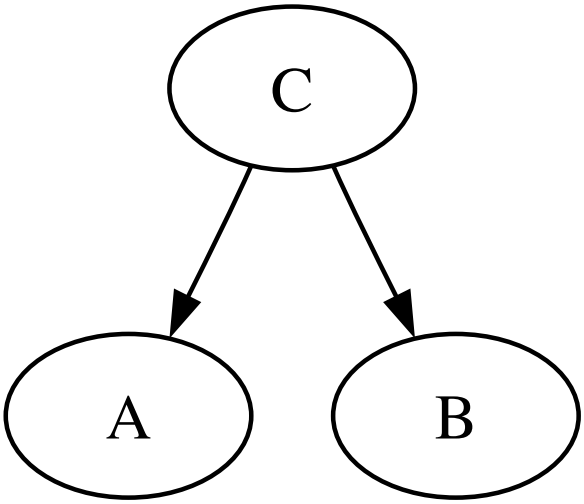
Graphical Models

- *Graphical models* are diagrammatic representations of probability distributions.
- Advantages:
  - The structure is more apparent in graphical representation.
  - Properties of the model, such as conditional independence, are easy to see.
  - Complex computations are reduced to graphical manipulations.
- Variations:
  - Bayesian networks (belief networks, probabilistic networks) [today]
  - Markov random fields
  - Factor graphs
- Applications:
  - Construction of probabilistic models
  - Biological networks (see T-61.6070 Modeling of biological networks)
  - ...

Bayesian Networks

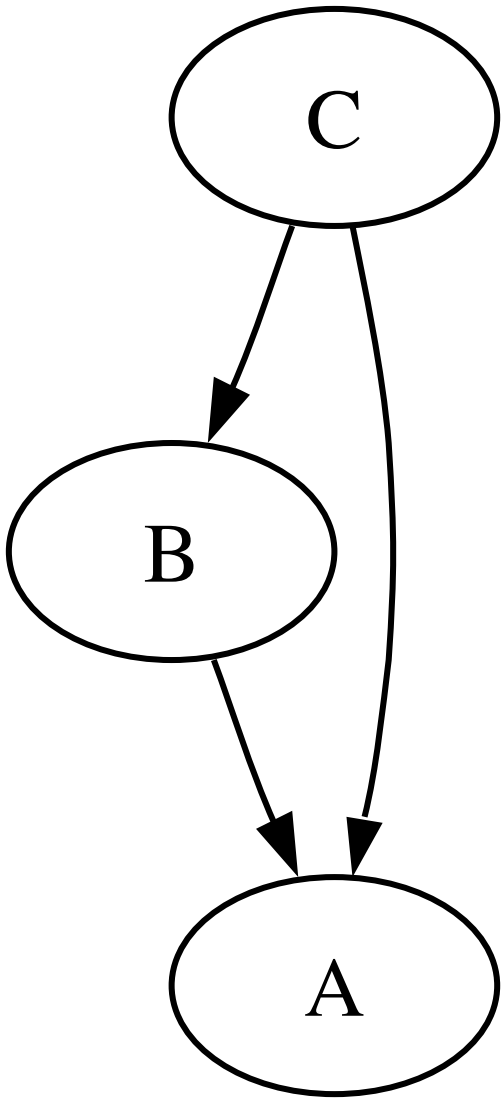
- How to efficiently represent joint probability distributions such as $P(Sky, AirTemp, \ldots, Forecast, EnjoySpo$
  (useful in computing Aldo's sport preferences $P(EnjoySport \mid Sky, \ldots, Forecast)$)

Bayesian Networks



Example 1:

$$P(A, B, C) = P(A \mid C)P(B \mid C)P(C).$$



Example 2:

$$P(A, B, C) = P(A \mid B, C)P(B \mid C)P(C).$$

Bayesian Networks

Bayesian network is a directed acyclic graph (DAG) that describes a joint distribution over the vertices $X_1,\ldots,X_d$ such that

$$P(X_1, \ldots, X_d) = \prod_{i=1}^{d} P(X_i \mid \text{parents}(X_i)),$$

where parents(X_i) are the set of vertices from which there is an edge to X_i.

- Example 1: $P(A, B, C) = P(A \mid C)P(B \mid C)P(C)$.
- Product rule: $P(A, B, C) = P(A, B \mid C)P(C) = P(A \mid B, C)P(B \mid C)P(C)$.

- Generally: $P(X_1, \ldots, X_d) = P(X_d \mid X_1, \ldots, X_{d-1}) \ldots P(X_2 \mid X_1)P(X_1)$.
- Example 2: All joint distributions $P(X_1, \ldots, X_d)$ can be represented by a graph with $d(d-1)/2$ edges.

8.2 Inference

Causes and Bayes' Rule

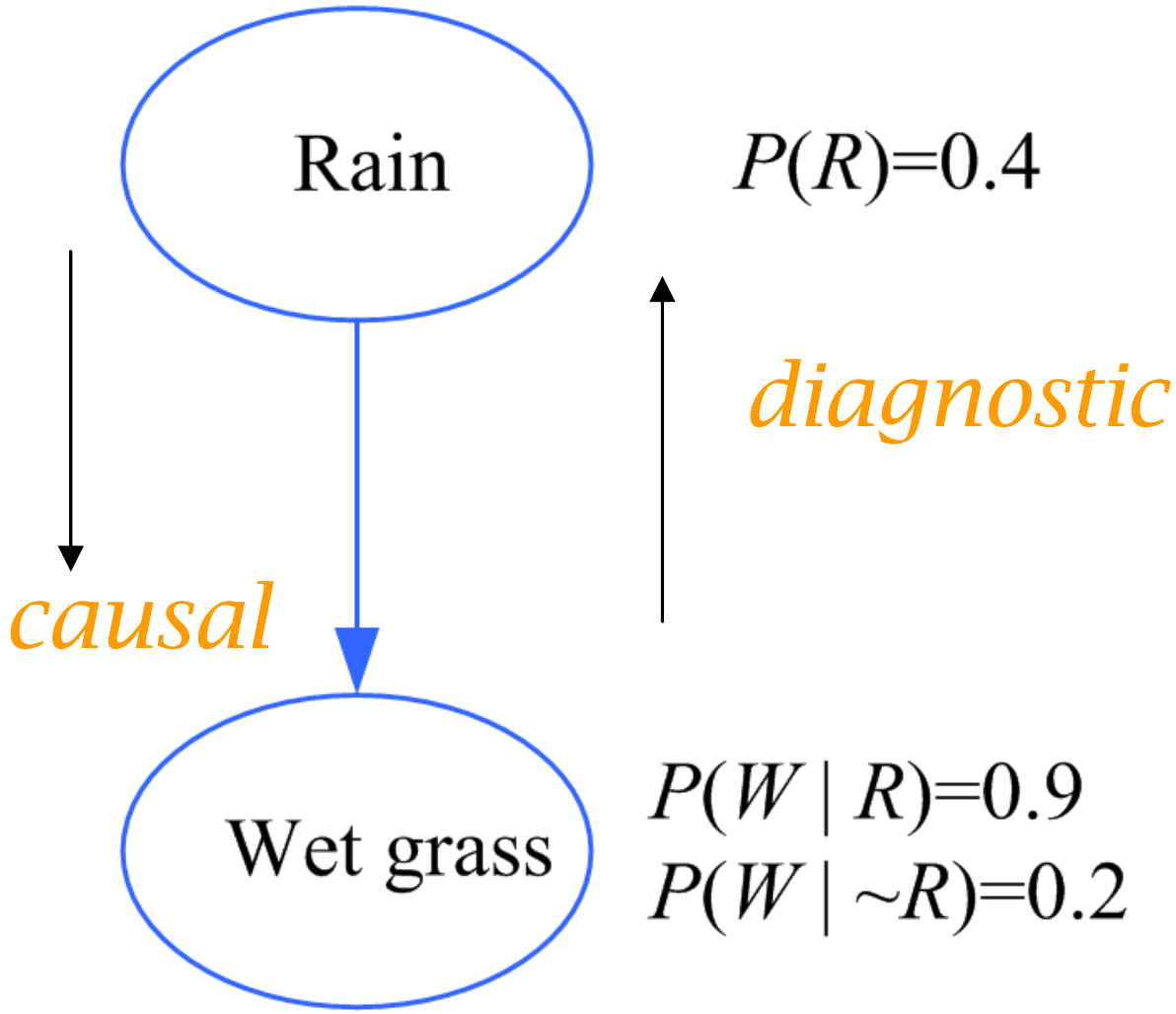
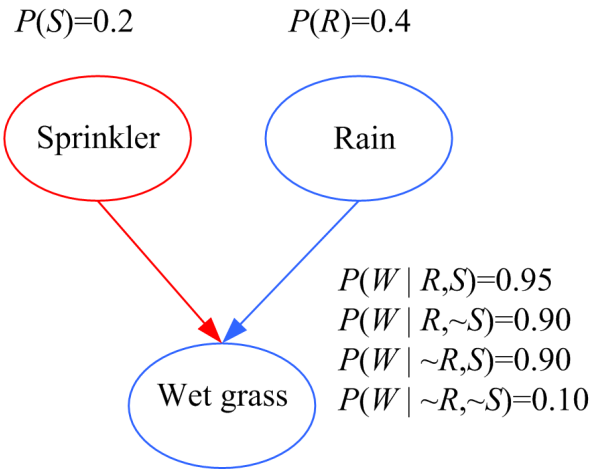


Figure 3.2 of Alpaydin (2004). $P(W, R) = P(W \mid R)P(R)$

Diagnostic inference: Knowing that grass is wet, what is the probability that rain is the cause?

$$\begin{aligned} P(R \mid W) &= \frac{P(W \mid R)P(R)}{P(W)} \\ &= \frac{P(W \mid R)P(R)}{P(W \mid R)P(R) + P(W \mid \sim R)P(\sim R)} \\ &= \frac{0.9 \times 0.4}{0.9 \times 0.4 + 0.2 \times 0.6} = 0.75 \end{aligned}$$

Causal vs. Diagnostic Inference



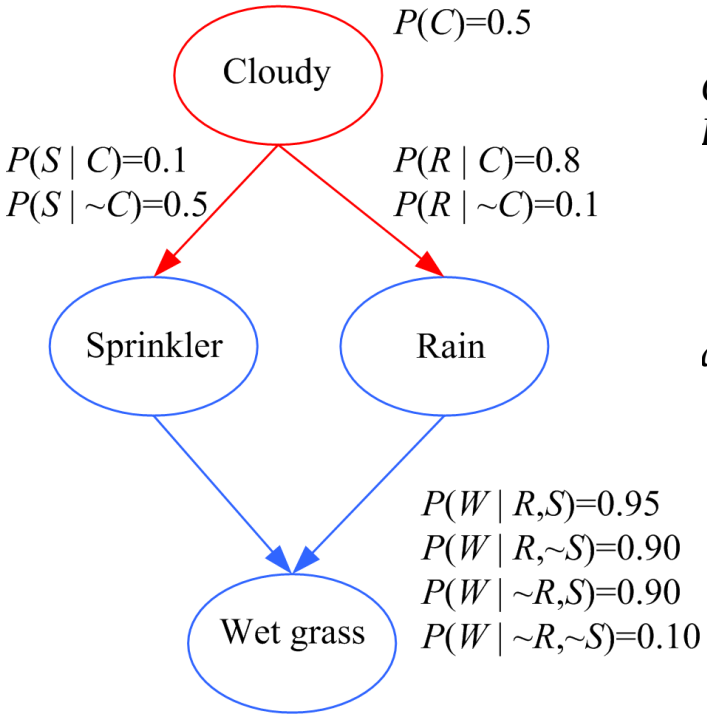
Causal inference: If the sprinkler is on, what is the probability that the grass is wet?

$$P(W|S) = P(W|R,S)\,P(R|S) + P(W|\sim R,S)\,P(\sim R|S)$$
$$= P(W|R,S)\,P(R) + P(W|\sim R,S)\,P(\sim R)$$
$$= 0.95\ 0.4 + 0.9\ 0.6 = 0.92$$

Diagnostic inference: If the grass is wet, what is the probability that the sprinkler is on? $P(S|W) = 0.35 > 0.2\ P(S)$

$P(S|R,W) = 0.21$ *Explaining away: Knowing that it has rained decreases the probability that the sprinkler is on.*

Bayesian Network: Causes



Causal inference:

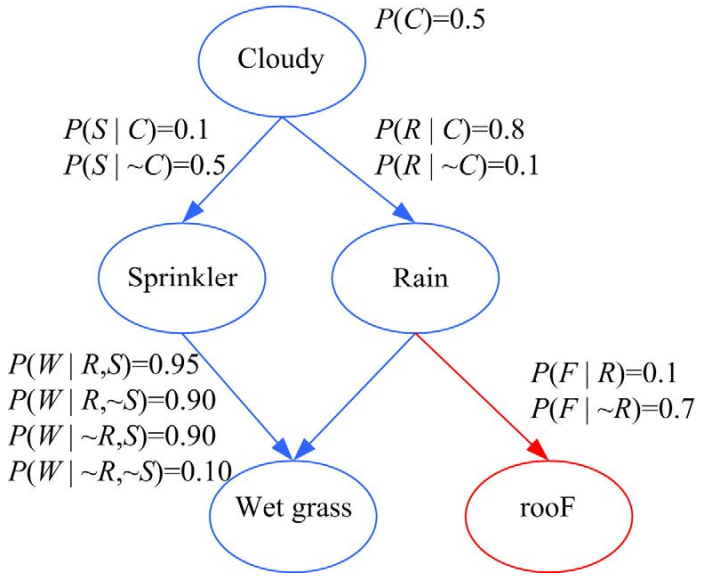
$$P(W|C) = P(W|R,S)\,P(R,S|C) + P(W|\sim R,S)\,P(\sim R,S|C) + P(W|R,\sim S)\,P(R,\sim S|C) + P(W|\sim R,\sim S)\,P(\sim R,\sim S|C)$$

and use the fact that

$$P(R,S|C) = P(R|C)\,P(S|C)$$

Diagnostic: $P(C|W) = ?$

Bayesian Networks: Local Structure



$$P(F \mid C) = ?$$

$$P(C,S,R,W,F) = P(C)P(S \mid C)P(R \mid C)P(W \mid S,R)P(F \mid R)$$

$$P(X_1,\ldots X_d) = \prod_{i=1}^{d} P(X_i \mid \text{parents}(X_i))$$

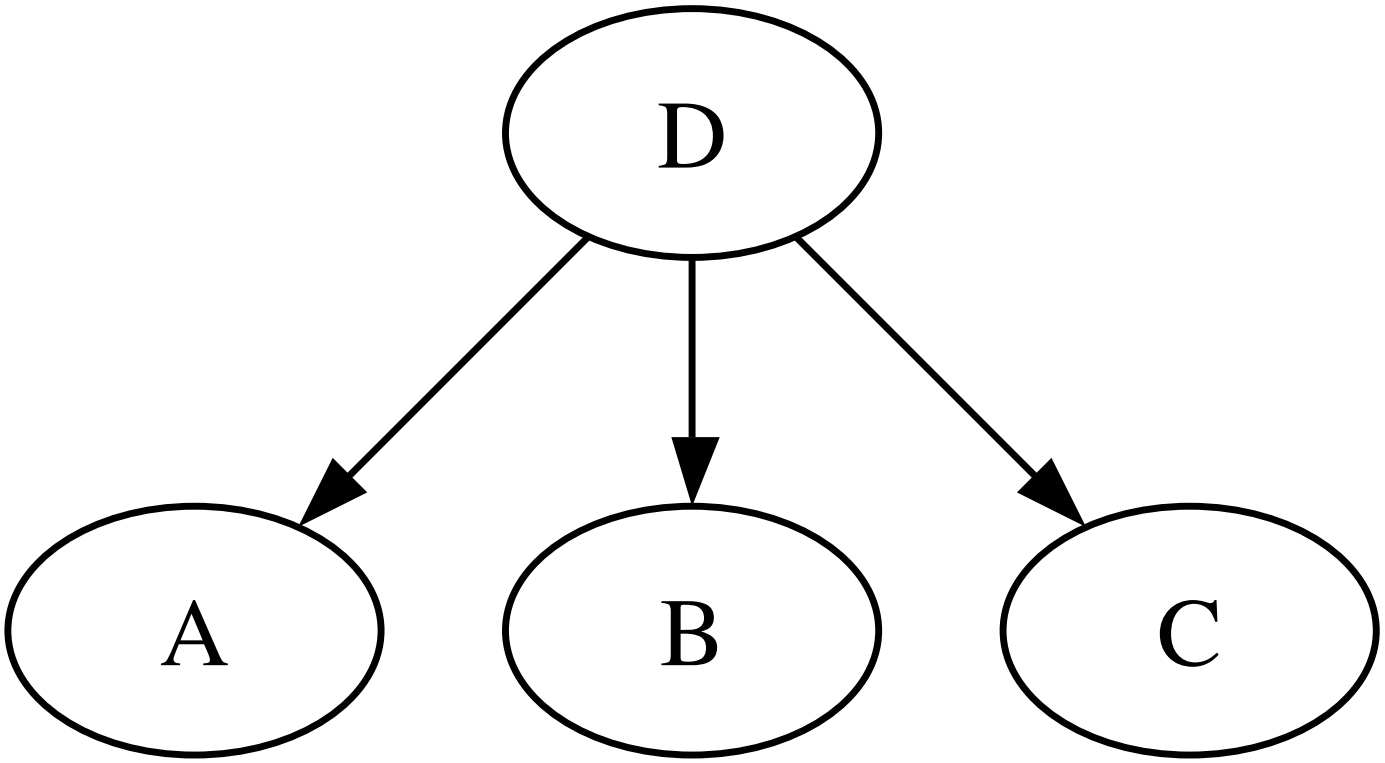
Bayesian Networks: Inference

- $P(C,S,R,W,F) = P(F \mid R)P(W \mid R,S)P(R \mid C)P(S \mid C)P(C)$.
- $P(C,F) = \sum_S \sum_R \sum_W P(C,S,R,W,F)$.
- $P(F \mid C) = P(C,F)/P(C)$.
- More generally: To do inference in Bayesian networks one has to *marginalize* over variables.
- For example: $P(X_1) = \sum_{X_2} \ldots \sum_{X_d} P(X_1,\ldots,X_d)$.
- If we have Boolean arguments the sum has $O(2^{d-1})$ terms. This is inefficient!
- Generally, marginalization is a NP-hard problem.
- If Bayesian Network is a tree: Sum-Product Algorithm
- If Bayesian Network is "close" to a tree: Junction Tree Algorithm
- Otherwise: approximate methods (variational approximation, MCMC etc.)

Sum-Product Algorithm

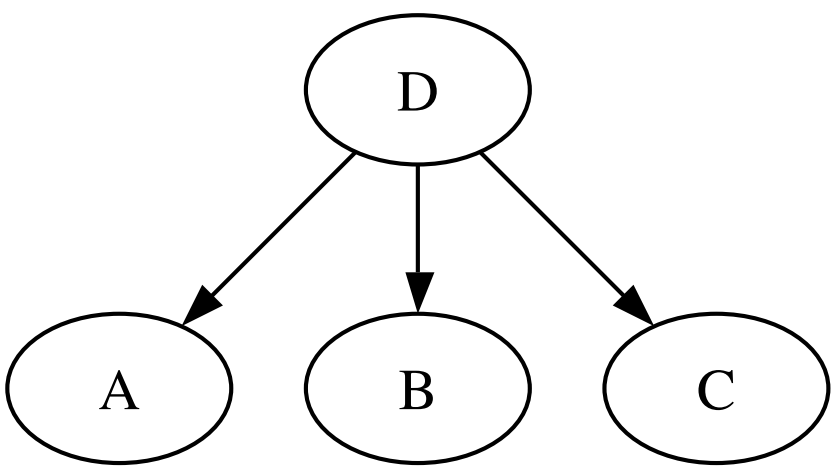
- Idea: sum of products is difficult to compute. Product of sums is easy to compute, if sums have been re-arranged smartly.
- Example: disconnected Bayesian network with d vertices, computing $P(X_1)$.
  - sum of products: $P(X_1) = \sum_{X_2} \dots \sum_{X_d} P(X_1) \dots P(X_d)$.
  - product of sums: $P(X_1) = P(X_1) \left(\sum_{X_2} P(X_2)\right) \dots \left(\sum_{X_d} P(X_d)\right) = P(X_1)$.
- Sum-Product Algorithm works if the Bayesian Network is directed tree.
- For details, see e.g., Bishop (2006).

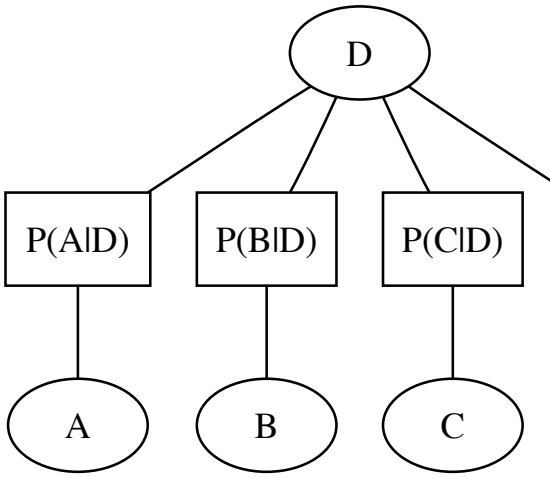
Sum-Product Algorithm



$$P(A, B, C, D) = P(A \mid D)P(B \mid D)P(C \mid D)P(D)$$

Task: compute $\tilde{P}(D) = \sum_A \sum_B \sum_C P(A, B, C, D)$.





$$P(A, B, C, D) = P(A \mid D)P(B \mid D)P(C \mid D)P(D)$$

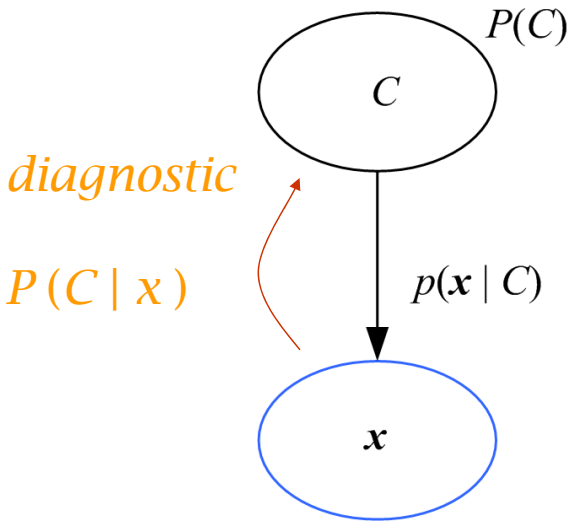
- *Factor graph* is composed of vertices (ellipses) and factors (squares), describing the factors of the joint probability.
- The Sum-Product Algorithm re-arranges the product (check!):

$$\begin{aligned} \tilde{P}(D) &= \left(\sum_A P(A \mid D)\right)\left(\sum_B P(B \mid D)\right)\left(\sum_C P(C \mid D)\right)P(D) \\ &= \sum_A \sum_B \sum_C P(A, B, C, D). \end{aligned} \tag{1}$$

Observations

- Bayesian network forms a *partial order* of the vertices. To find (one) total ordering of vertices: remove a vertex with no outgoing edges (zero out-degree) from the network and output the vertex. Iterate until the network is empty. (This way you can also check that the network is DAG.)
- If all variables are Boolean, storing a full Bayesian network of d vertices — or full joint distribution — as a look-up table takes $O(2^d)$ bytes.
- If the highest number of incoming edges (in-degree) is k, then storing a Bayesian network of d vertices as a look-up table takes $O(d2^{k+1})$ bytes.
- When computing marginals, disconnected parts of the network do not contribute.
- We can marginalize over unknown (hidden) variables.

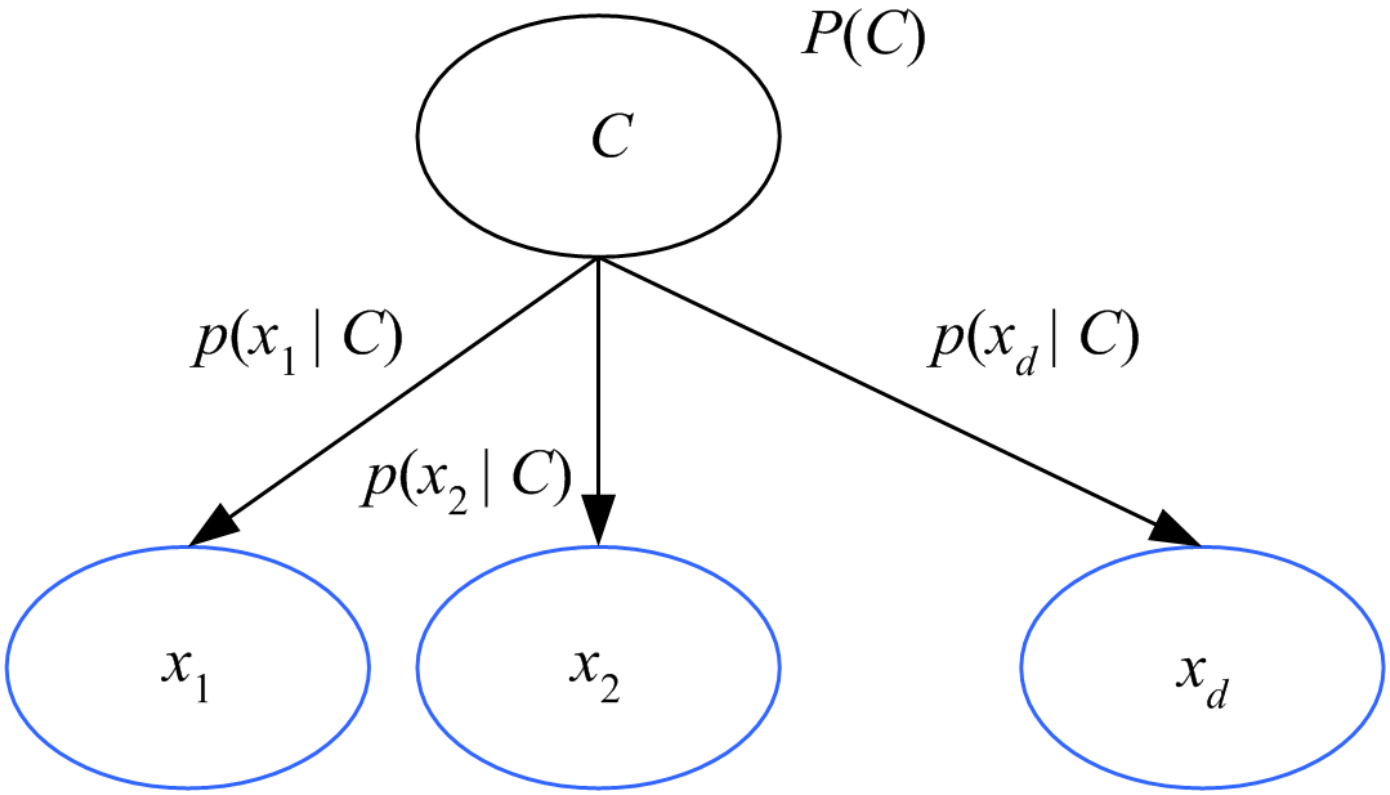
Bayesian Network: Classification



Bayes' rule inverts the arc:

$$P(C \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid C)P(C)}{p(\boldsymbol{x})}$$

Naive Bayes' Classifier



Given C, x_j are independent:

$$p(\mathbf{x}|C) = p(x_1|C)\ p(x_2|C)\ \ldots\ p(x_d|C$$

(2004) Ch 3 / slides

8.3 Finding a Network

Finding a Network

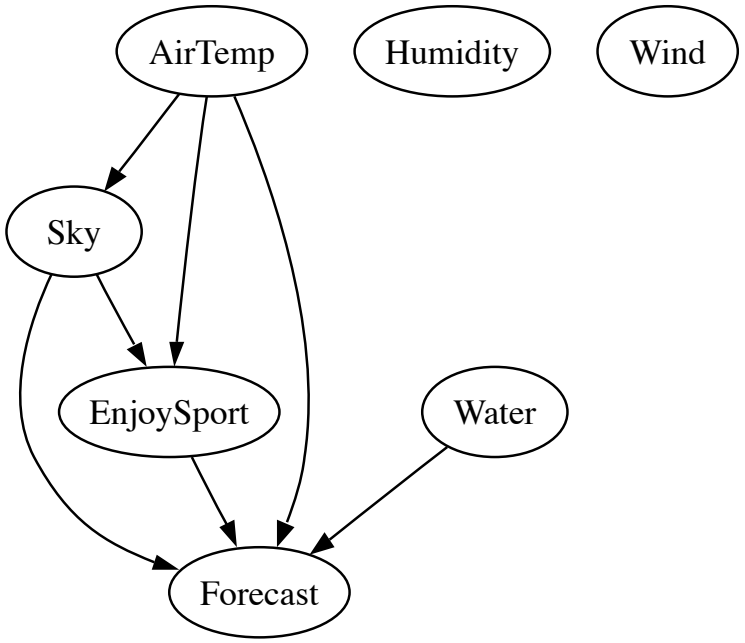
- Often, the network structure is given by an expert.
- In probabilistic modeling, the network structure defines the structure of the model.
- Finding an optimal Bayesian network structure is NP-hard (given some complexity criterion, described in later lectures).

Finding a Network

t	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	1
2	Sunny	Warm	High	Strong	Warm	Same	1
3	Rainy	Cold	High	Strong	Warm	Change	0
4	Sunny	Warm	High	Strong	Cool	Change	1

- Full Bayesian network of d vertices and $d(d-1)/2$ edges describes the training set fully and the test set probably poorly.
- As before, in finding the network structure, we must control the complexity so that the the model generalizes.
- Usually one must resort to approximate solutions to find the network structure (e.g., DEAL package in R).
- A feasible exact algorithm exists for up to $d = 32$ variables, with a running time of $o(d^2 2^{d-2})$.
- See Silander et al. (2006) A Simple Optimal Approach for Finding the Globally Optimal Bayesian Network Structure. In Proc 22nd UAI. (pdf)

Finding a Network



Network found by *Bene* at

`http://b-course.hiit.fi/bene`

Conclusion

- Next lecture on 2 October: Parametric Methods, Alpaydin (2004) Ch 4.

- Problem session on 28 September: last week's (2/2007) and this week's problem sheets (3/2007).

9 Bayesian Networks

9.1 Reminders

Rules of Probability

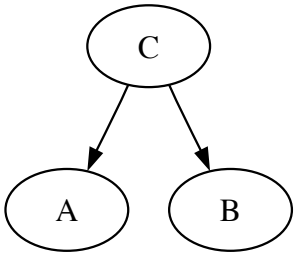
- $P(E, F) = P(F, E)$: probability of both E and F happening.
- $P(E) = \sum_F P(E, F)$ (sum rule, marginalization)
- $P(E, F) = P(F \mid E)P(E)$ (product rule, conditional probability)
- Consequence: $P(F \mid E) = P(E \mid F)P(F)/P(E)$ (Bayes' formula)
- We say E and F are *independent* if $P(E, F) = P(E)P(F)$ (for all E and F).
- We say E and F are *conditionally independent* given G if $P(E, F \mid G) = P(E \mid G)P(F \mid G)$, or equivalently $P(E \mid F, G) = P(E \mid G)$.

Bayesian Networks

Bayesian network is a directed acyclic graph (DAG) that describes a joint distribution over the vertices $X_1,\ldots,X_d$ such that

$$P(X_1, \ldots, X_d) = \prod_{i=1}^{d} P(X_i \mid \text{parents}(X_i)),$$

where parents(X_i) are the set of vertices from which there is an edge to X_i.



$P(A, B, C) = P(A \mid C)P(B \mid C)P(C)$. ($A$ and B are conditionally independent given C.)

9.2 Inference

Inference in Bayesian Networks

- When structure of the Bayesian network and the probability factors are known, one usually wants to do inference by computing conditional probabilities.
- This can be done with the help of the sum and product rules.

- Example: probability of the cat being on roof if it is cloudy, $P(F \mid C)$?

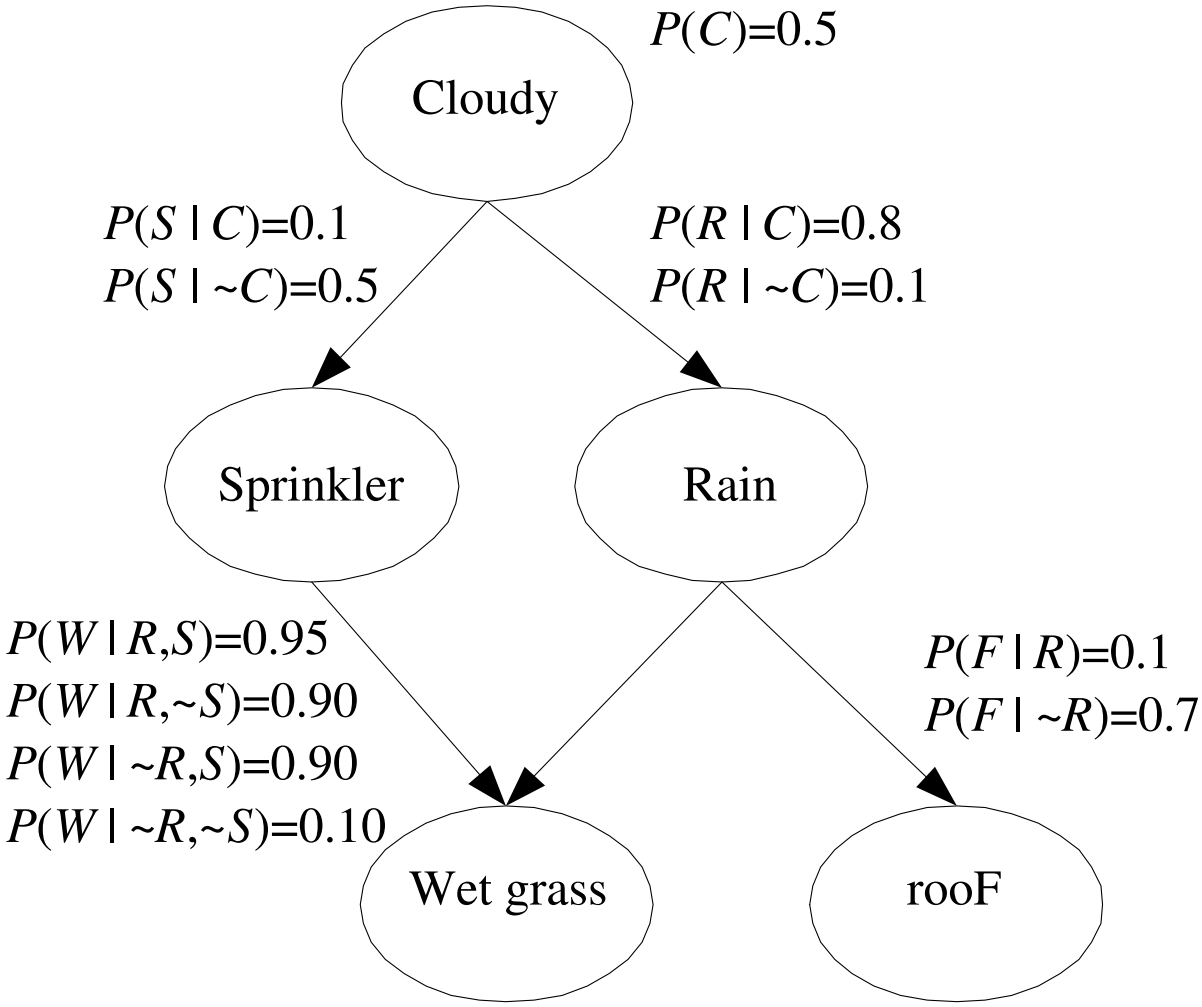


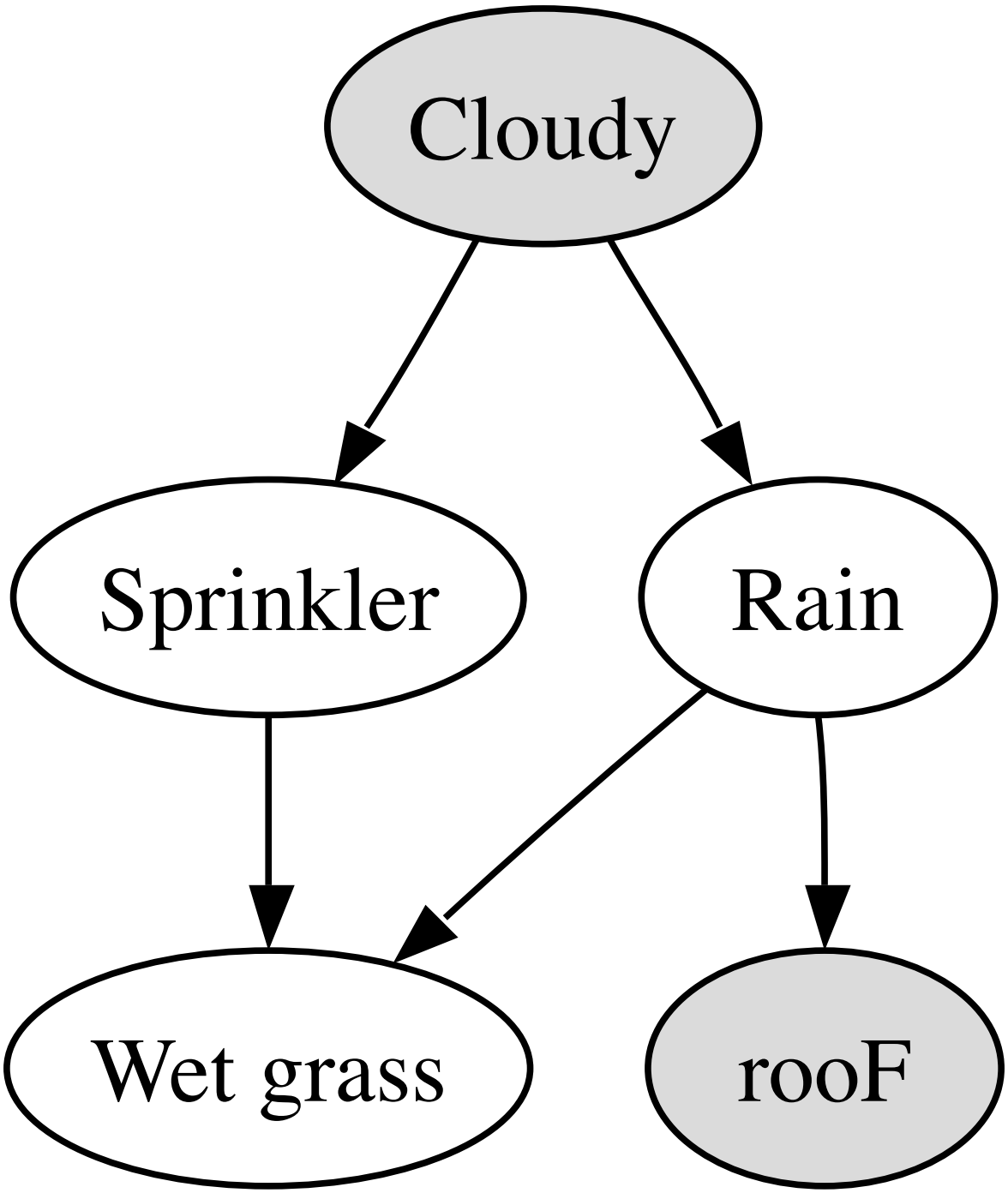
Figure 3.5 of Alpaydin (2004).

Inference in Bayesian Networks

- Example: probability of the cat being on roof if it is cloudy, $P(F \mid C)$?
- S, R and W are unknown or *hidden* variables.
- F and C are *observed* variables. Conventionally, we denote the observed variables by gray nodes (see figure on the right).
- We use the product rule $P(F \mid C) = P(F, C)/P(C)$, where $P(C) = \sum_F P(F, C)$.
- We must sum over or *marginalize* over hidden variables S, R and W: $P(F, C) = \sum_S \sum_R \sum_W P(C, S, R, W, F)$

$$
\begin{aligned}
&P(F, C) = \\
&P(C, S, R, W, F) + P(C, -S, R, W, F) \\
&+P(C, S, -R, W, F) + P(C, -S, -R, W, F) \\
&+P(C, S, R, -W, F) + P(C, -S, R, -W, F) \\
&+P(C, S, -R, -W, F) + P(C, -S, -R, -W, F)
\end{aligned}
$$

- We obtain similar formula for $P(F, -C)$, $P(-F, C)$ and $P(-F, -C)$.
- Notice: we have used shorthand F to denote $F = 1$ and $-F$ to denote $F = 0$.
- In principle, we know the numeric value of each joint distribution, hence we can compute the probabilities.
- There are 2^5 terms in the sums.
- Generally: marginalization is NP-hard, the most staightforward approach would involve a computation of $O(2^d)$ terms.
- We can often do better by smartly re-arranging the sums and products. Behold:
- Do the marginalization over W first: $P(C, S, R, F) = \sum_W P(F \mid R)P(W \mid S, R)P(S \mid C)P(R \mid C)P(C) = P(F \mid R)\sum_W [P(W \mid S, R)]P(S \mid C)P(R \mid C)P(C) = P(F \mid R)P(S \mid C)P(R \mid C)P(C)$.
- Now we can marginalize over S easily: $P(C, R, F) = \sum_S P(F \mid R)P(S \mid C)P(R \mid C)P(C) = P(F \mid R)\sum_S [P(S \mid C)]P(R \mid C)P(C) = P(F \mid R)P(R \mid C)P(C)$.
- We must still marginalize over R: $P(C, F) = P(F \mid R)P(R \mid C)P(C) + P(F \mid -R)P(-R \mid C)P(C) = 0.1 \times 0.8 \times 0.5 + 0.7 \times 0.2 \times 0.5 = 0.11$.
- $P(C, -F) = P(-F \mid R)P(R \mid C)P(C) + P(-F \mid -R)P(-R \mid C)P(C) = 0.9 \times 0.8 \times 0.5 + 0.3 \times 0.2 \times 0.5 = 0.39$.
- $P(C) = P(C, F) + P(C, -F) = 0.5$.
- $P(F \mid C) = P(C, F)/P(C) = 0.22$.
- $P(-F \mid C) = P(C, -F)/P(C) = 0.78$.



$$P(C, S, R, W, F) = P(F \mid R)P(W \mid S, R)P(S \mid C)P(R \mid C)P(C)$$

Bayesian Networks: Inference

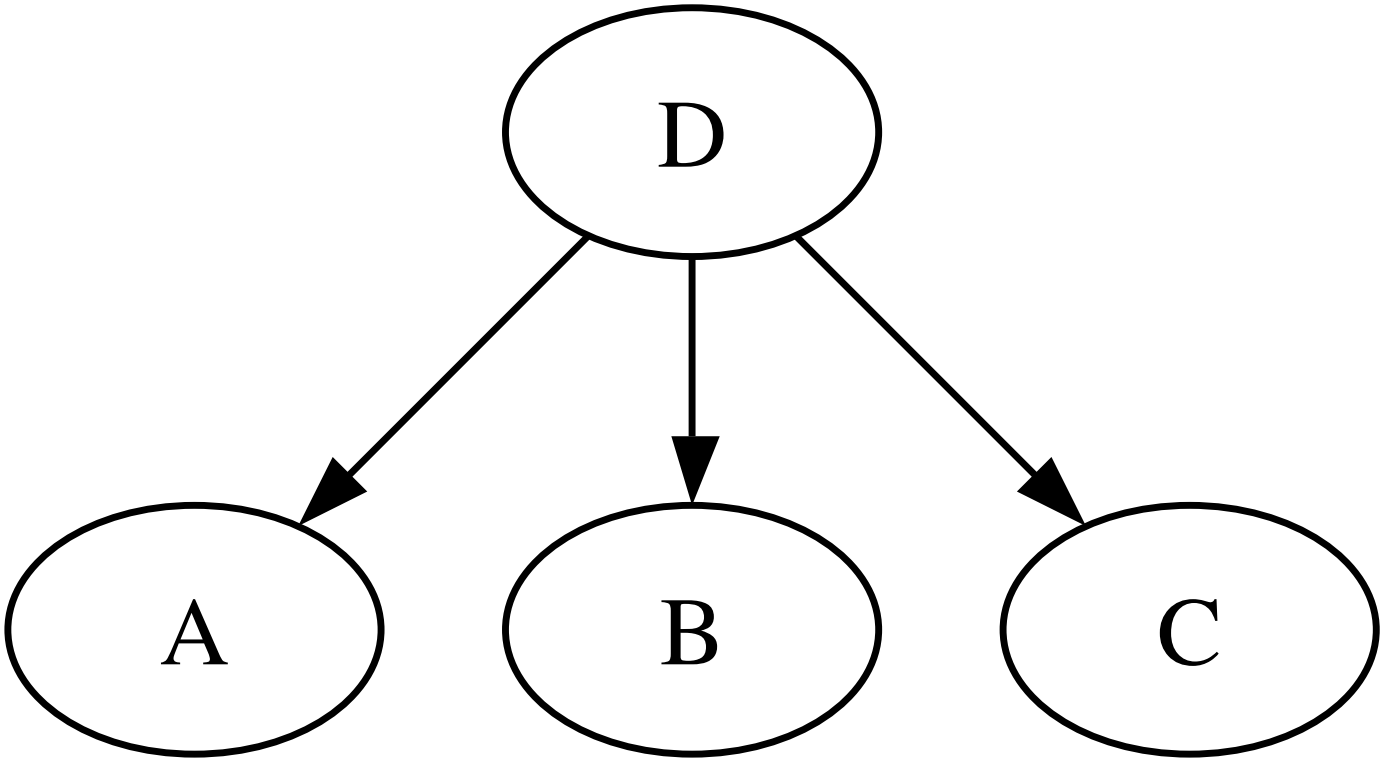
- To do inference in Bayesian networks one has to *marginalize* over variables.

- For example: $P(X_1) = \sum_{X_2} \dots \sum_{X_d} P(X_1, \dots, X_d)$.
- If we have Boolean arguments the sum has $O(2^d)$ terms. This is inefficient!
- Generally, marginalization is a NP-hard problem.
- If Bayesian Network is a tree: Sum-Product Algorithm (a special case being Belief Propagation).
- If Bayesian Network is "close" to a tree: Junction Tree Algorithm.
- Otherwise: approximate methods (variational approximation, MCMC etc.)

Sum-Product Algorithm

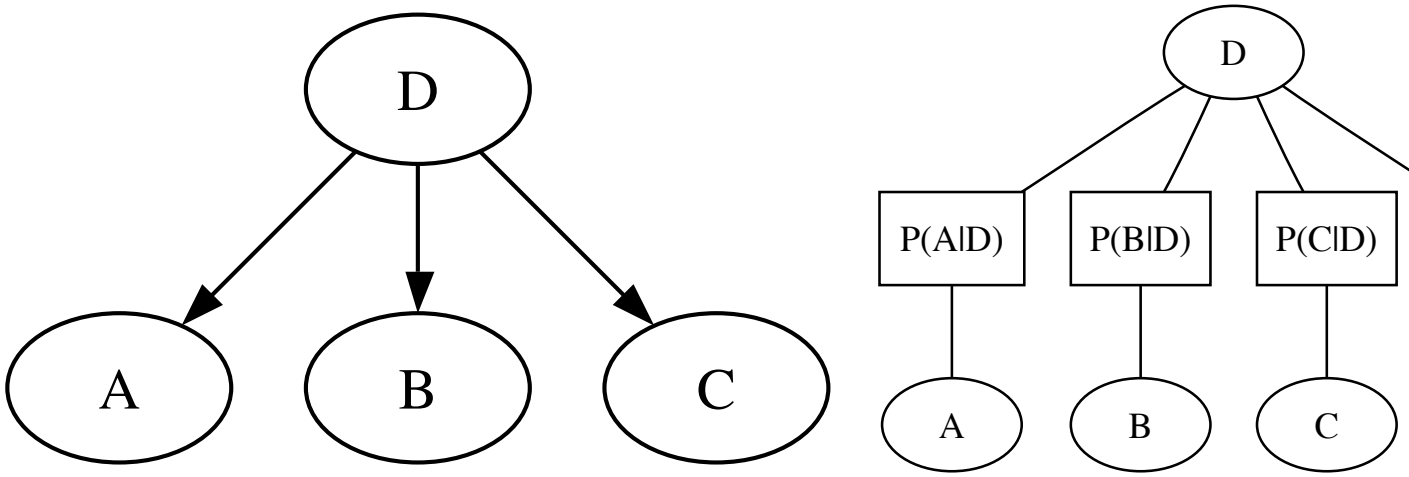
- Idea: sum of products is difficult to compute. Product of sums is easy to compute, if sums have been re-arranged smartly.
- Example: disconnected Bayesian network with d vertices, computing $P(X_1)$.
  - sum of products: $P(X_1) = \sum_{X_2} \dots \sum_{X_d} P(X_1) \dots P(X_d)$.
  - product of sums: $P(X_1) = P(X_1) \left(\sum_{X_2} P(X_2)\right) \dots \left(\sum_{X_d} P(X_d)\right) = P(X_1)$.
- Sum-Product Algorithm works if the Bayesian Network is directed tree.
- For details, see e.g., Bishop (2006).

Sum-Product Algorithm



$$P(A, B, C, D) = P(A \mid D)P(B \mid D)P(C \mid D)P(D)$$

Task: compute $\tilde{P}(D) = \sum_A \sum_B \sum_C P(A, B, C, D)$.



$$P(A, B, C, D) = P(A \mid D)P(B \mid D)P(C \mid D)P(D)$$

- *Factor graph* is composed of vertices (ellipses) and factors (squares), describing the factors of the joint probability.

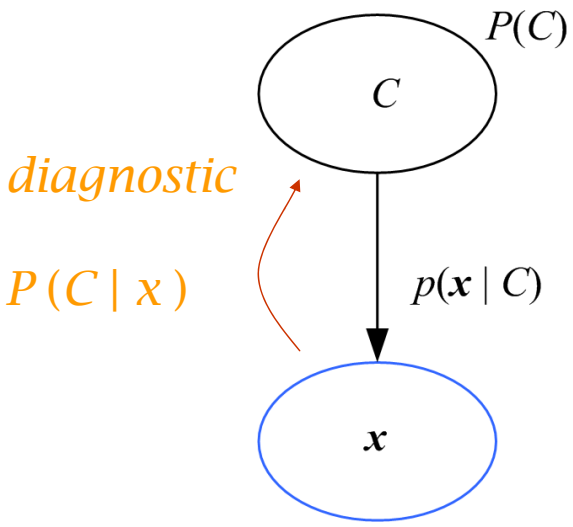
- The Sum-Product Algorithm re-arranges the product (check!):

$$\begin{aligned}\tilde{P}(D) &= \left(\sum_A P(A \mid D)\right)\left(\sum_B P(B \mid D)\right)\left(\sum_C P(C \mid D)\right)P(D) \\ &= \sum_A \sum_B \sum_C P(A, B, C, D). \end{aligned} \tag{2}$$

Observations

- Bayesian network forms a *partial order* of the vertices. To find (one) total ordering of vertices: remove a vertex with no outgoing edges (zero out-degree) from the network and output the vertex. Iterate until the network is empty. (This way you can also check that the network is DAG.)
- If all variables are Boolean, storing a full Bayesian network of d vertices — or full joint distribution — as a look-up table takes $O(2^d)$ bytes.
- If the highest number of incoming edges (in-degree) is k, then storing a Bayesian network of d vertices as a look-up table takes $O(d2^k)$ bytes.
- When computing marginals, disconnected parts of the network do not contribute.
- Conditional independence is "easy" to see.

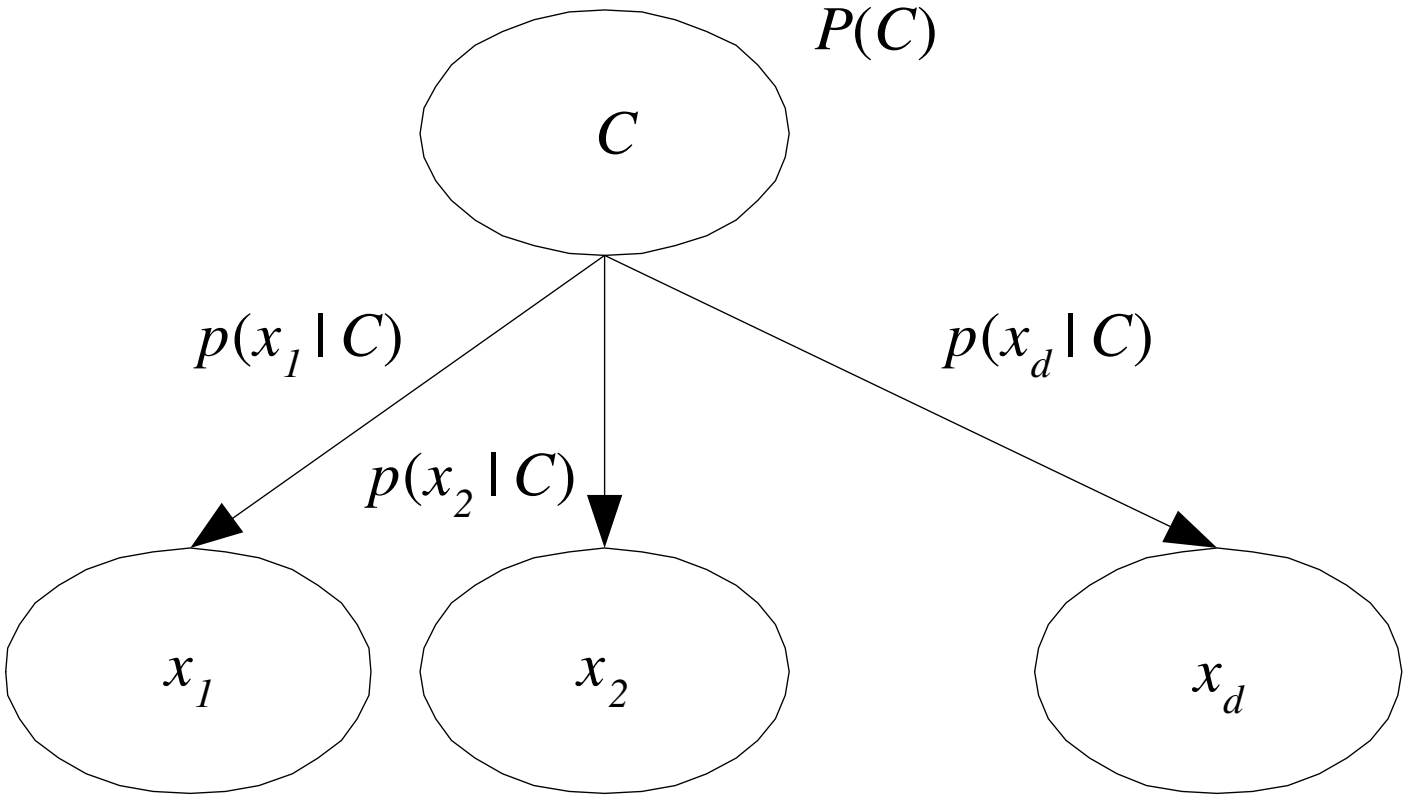
Bayesian Network: Classification



Bayes' rule inverts the arc:

$$P(C \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid C)P(C)}{p(\boldsymbol{x})}$$

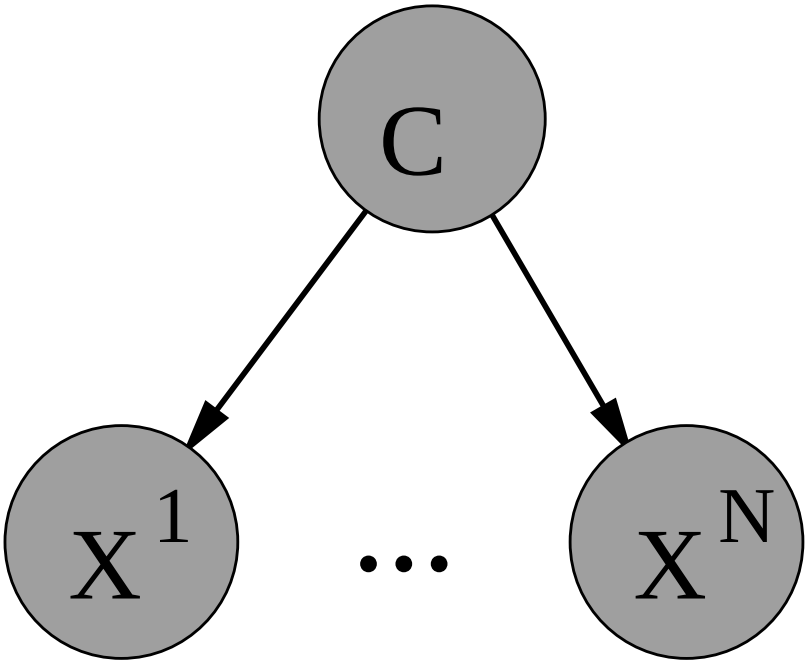
Naive Bayes' Classifier



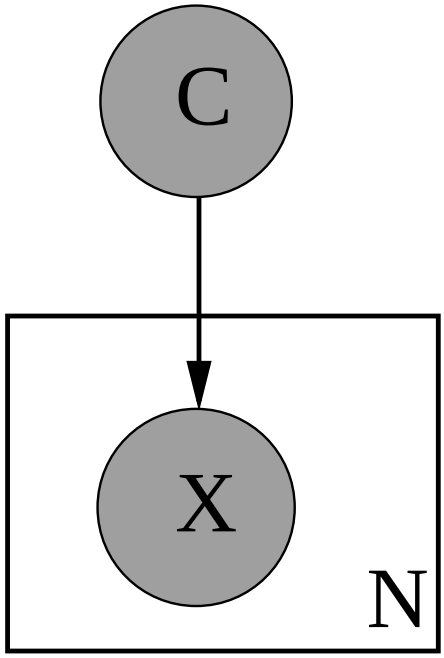
Fi

3.7 Alpaydin (2004).

- X^i are conditionally independent given C.
- $P(\mathcal{X}, C) = P(x^1 \mid C)P(x^2 \mid C) \ldots P(x^d \mid C)P(C)$.



Equivalently:



- *Plate* is used as a shorthand notation for repetition. The number of repetitions is in the bottom right corner.
- Gray nodes denote observed variables.

9.3 Finding the Structure of the Network

Finding the Structure of the Network

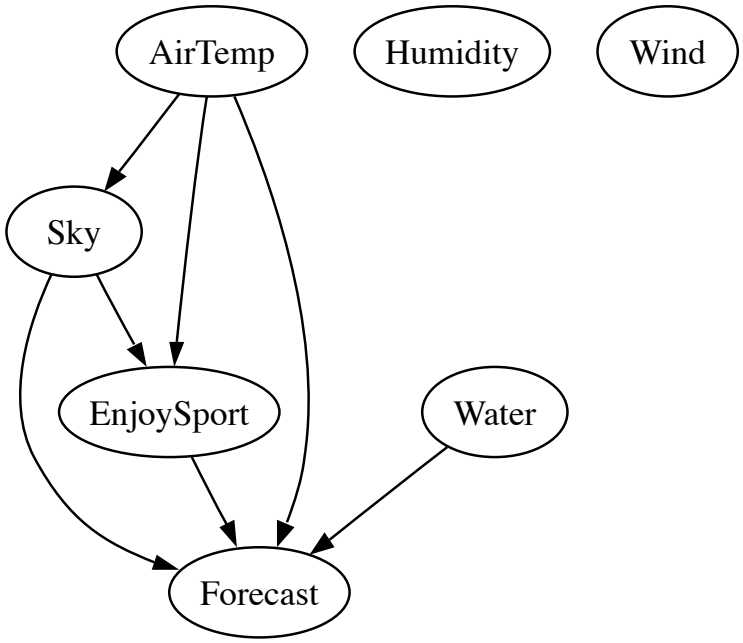
- Often, the network structure is given by an expert.
- In probabilistic modeling, the network structure defines the structure of the model.
- Finding an optimal Bayesian network structure is NP-hard
- Idea: Go through all possible network structures M and compute the likelihood of data $\mathcal{X}$ given the network structure $P(\mathcal{X} \mid M)$.
- Choose the network complexity appropriately.
- Choose network that, for a given network complexity, gives the best likelihood.
- The Bayesian approach: choose structure M that maximizes $P(M \mid \mathcal{X}) \propto P(\mathcal{X} \mid M)P(M)$, where $P(M)$ is a prior probability for network structure M (more complex networks should have smaller prior probability).

Finding a Network

- Full Bayesian network of d vertices and $d(d-1)/2$ edges describes the training set fully and the test set probably poorly.
- As before, in finding the network structure, we must control the complexity so that the the model generalizes.
- Usually one must resort to approximate solutions to find the network structure (e.g., DEAL package in R).
- A feasible exact algorithm exists for up to $d = 32$ variables, with a running time of $o(d^2 2^{d-2})$.
- See Silander et al. (2006) A Simple Optimal Approach for Finding the Globally Optimal Bayesian Network Structure. In Proc 22nd UAI. (pdf)

Finding a Network

t	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	1
2	Sunny	Warm	High	Strong	Warm	Same	1
3	Rainy	Cold	High	Strong	Warm	Change	0
4	Sunny	Warm	High	Strong	Cool	Change	1



Network found by *Bene* at

`http://b-course.hiit.fi/bene`

10 Probabilistic Inference

10.1 Bernoulli Process

Boys or Girls?

Bernoulli Process

- The world average probability that a newborn child is a boy ($X = 1$) is about $\theta = 0.512$ [probability of a girl ($X = 0$) is then $1 - \theta = 0.488$].
- Bernoulli process:
$$P(X = x \mid \theta) = \theta^x \, (1 - \theta)^{1-x} \quad , \quad x \in \{0, 1\}.$$
- Assume we observe the genders of N newborn children, $\mathcal{X} = \{x^t\}_{t=1}^N$. What is the sex ratio?
- Joint distribution: $P(x^1, \ldots, x^N, \theta) = P(x^1 \mid \theta) \ldots P(x^N \mid \theta) P(\theta)$.
- Notice we must fix some *prior* for θ, $P(\theta)$.

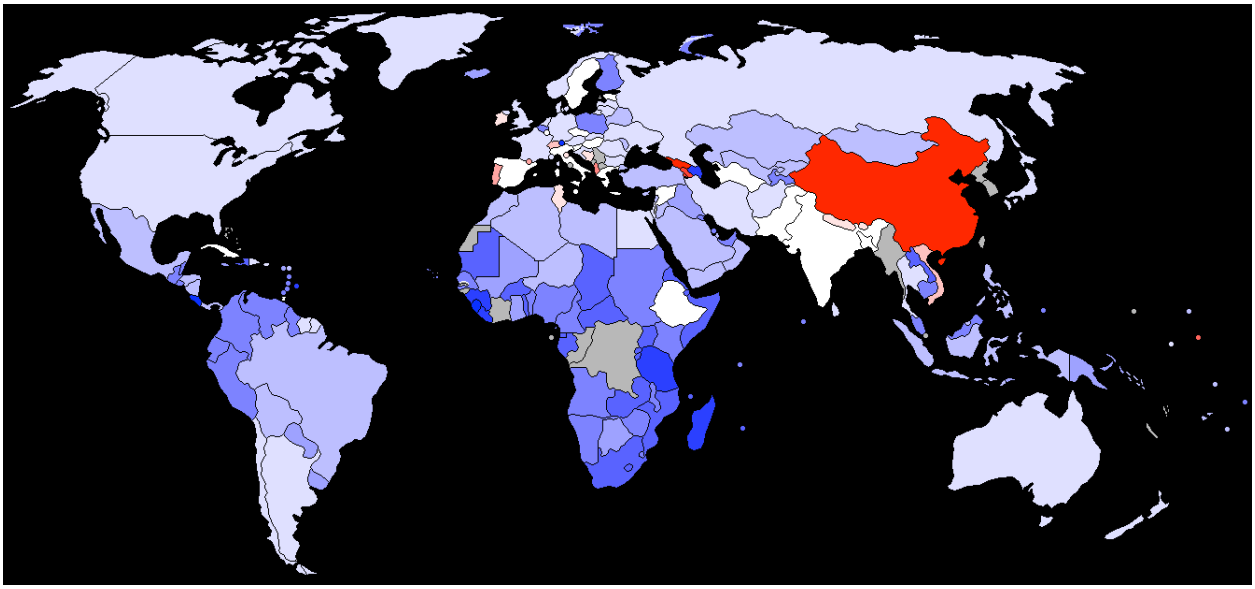
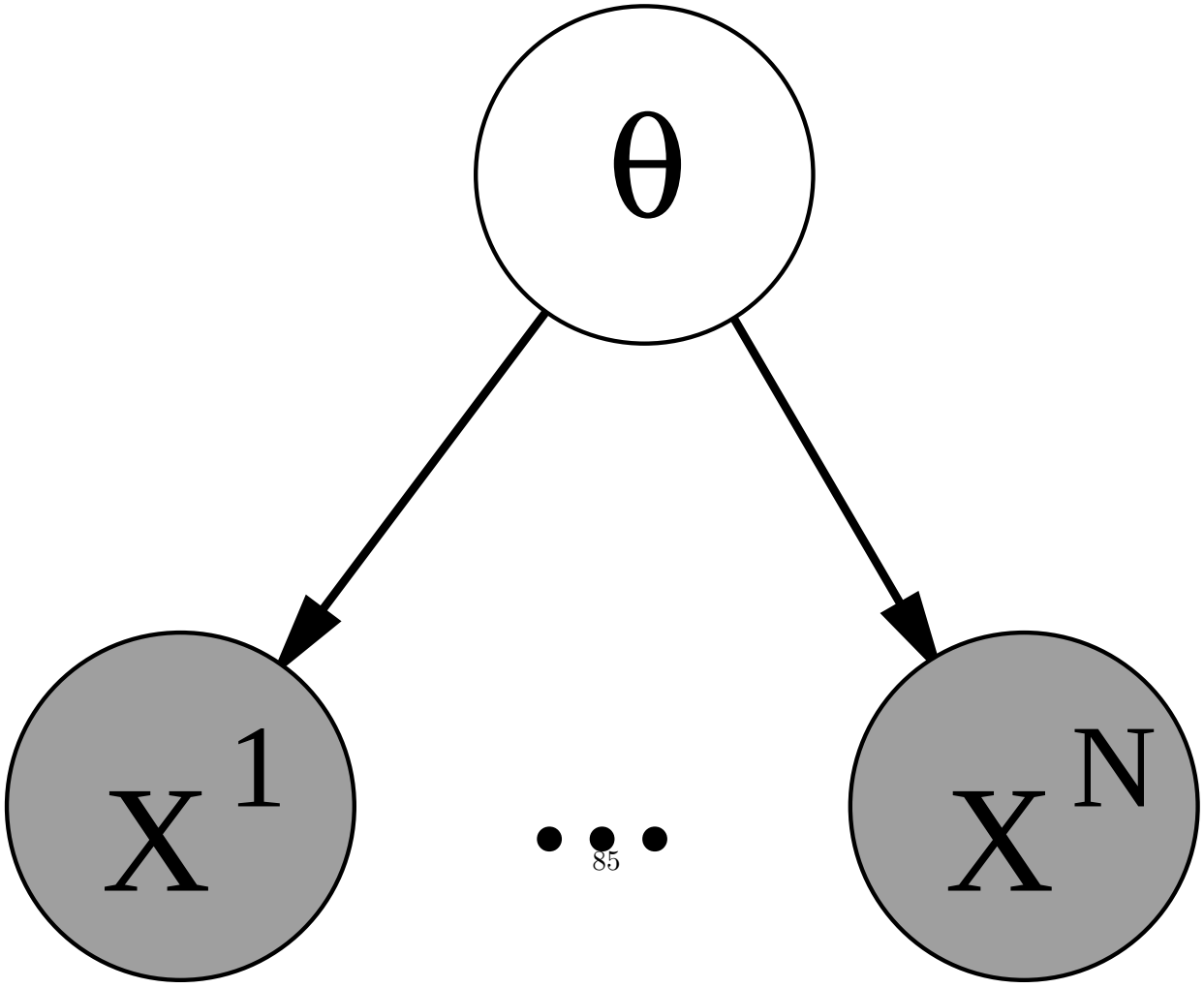
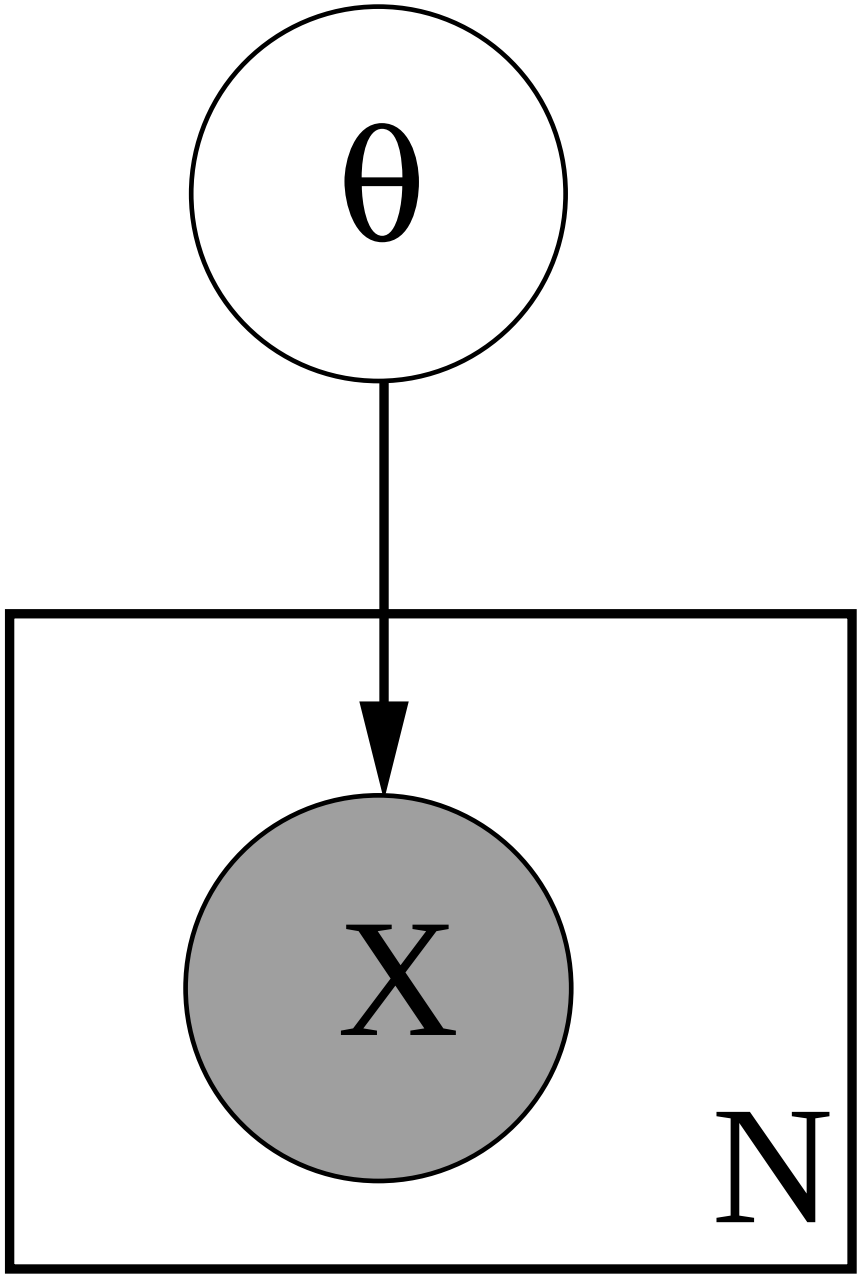


Figure 1: Sex ratio by country population aged below 15. Blue represents more women, red more men than the world average of 1.06 males/female. Image from Wikimedia Commons, author Dbachmann, GFDLv1.2.





Equivalently:

10.2 Posterior Probabilities

Comparing Models

- The *likelihood ratio* (Bayes factor) is defined by

$$BF(\theta_2;\theta_1) = \frac{P(\mathcal{X} \mid \theta_2)}{P(\mathcal{X} \mid \theta_1)}$$

- If we believe before seeing any data that the probability of model θ_1 is $P(\theta_1)$ and of model θ_2 is $P(\theta_2)$ then the ratio of their posterior probabilities is given by

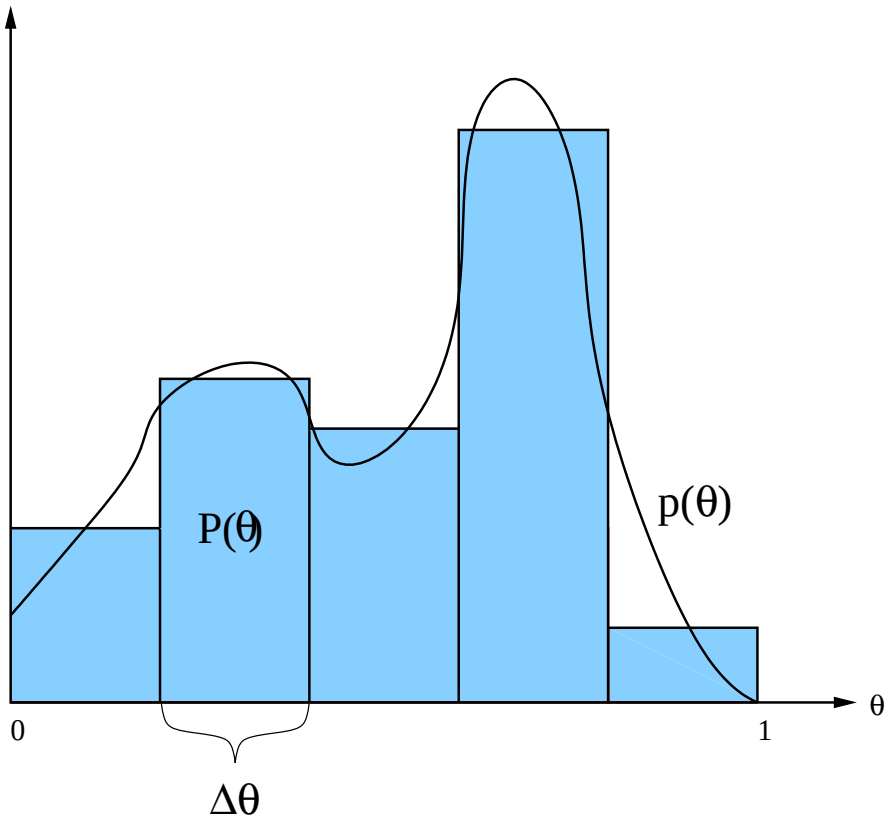
$$\frac{P(\theta_2 \mid \mathcal{X})}{P(\theta_1 \mid \mathcal{X})} = \frac{P(\theta_2)}{P(\theta_1)} \times BF(\theta_1; \theta_2)$$

- This ratio allows us to compare our degrees of beliefs into two models.
- *Posterior probability density* allows us to compare our degrees of beliefs between infinite number of models after observing the data.

Discrete vs. Continuous Random Variables

- The Bernoulli parameter θ is a real number in $[0, 1]$.
- Previously we considered binary $(0/1)$ random variables.
- Generalization to multinomial random variables that can have values $1, 2, \ldots, K$ is straightforward.
- Generalization to continuous random variable: divide the interval $[0, 1]$ to K equally sized intervals of width $\Delta\theta = 1/K$. Define *probability density* $p(\theta)$ such that the probability of θ being in interval $S_i = [(i-1)\Delta\theta, i\Delta\theta]$, $i \in \{1, \ldots, K\}$, is $P(\theta \in S_i) = p(\theta')\Delta\theta$, where θ' is some point in S_i.
- At limit $\Delta\theta \to 0$:

$$E_{P(\theta)}[f(\theta)] = \sum_{\theta} P(\theta) f(\theta) \longrightarrow E_{p(\theta)}[f(\theta)] = \int d\theta p(\theta) f(\theta).$$



- $P(\theta \in [(i-1)\Delta\theta, i\Delta\theta]) = p(\theta')\Delta\theta$.
- At limit $\Delta\theta \to 0$:

$$E_{P(\theta)}[f(\theta)] = \sum_{\theta} P(\theta)f(\theta) \longrightarrow E_{p(\theta)}[f(\theta)] = \int d\theta p(\theta)f(\theta).$$

Estimating the Sex Ratio

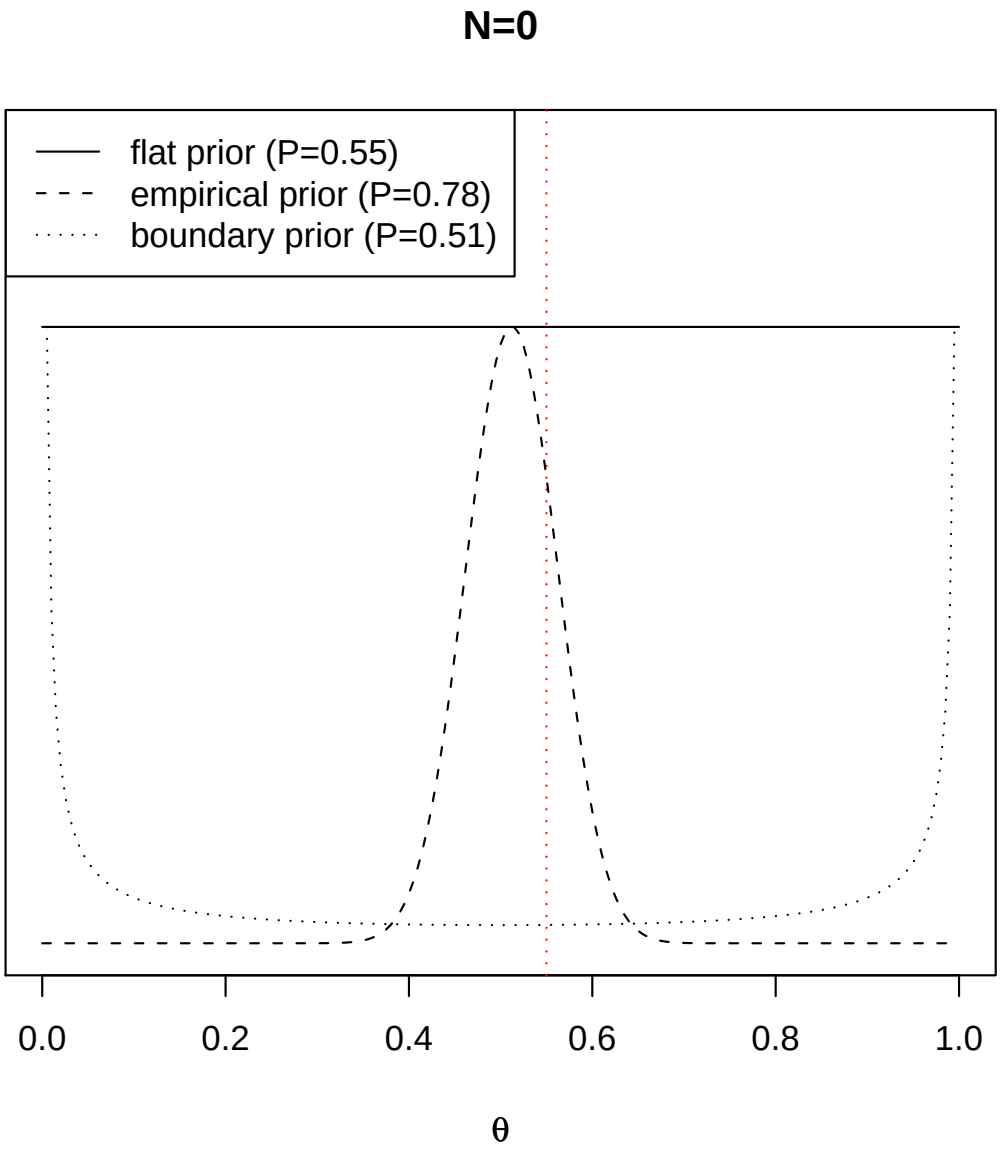
- Task: estimate the Bernoulli parameter θ, given N observations of the genders of newborns in an unnamed country.
- Assume the "true" Bernoulli parameter to be estimated in the unnamed country is $\theta = 0.55$, the global average being 51.2%.
- Posterior probability density after seeing N newborns in $\mathcal{X} = \{x^t\}_{t=1}^N$:

$$\begin{aligned} p(\theta \mid \mathcal{X}) &= \frac{p(\mathcal{X} \mid \theta)p(\theta)}{p(\mathcal{X})} \\ &\propto p(\theta)\prod_{t=1}^{N}\left[\theta^{x^t}(1-\theta)^{1-x^t}\right]. \end{aligned}$$

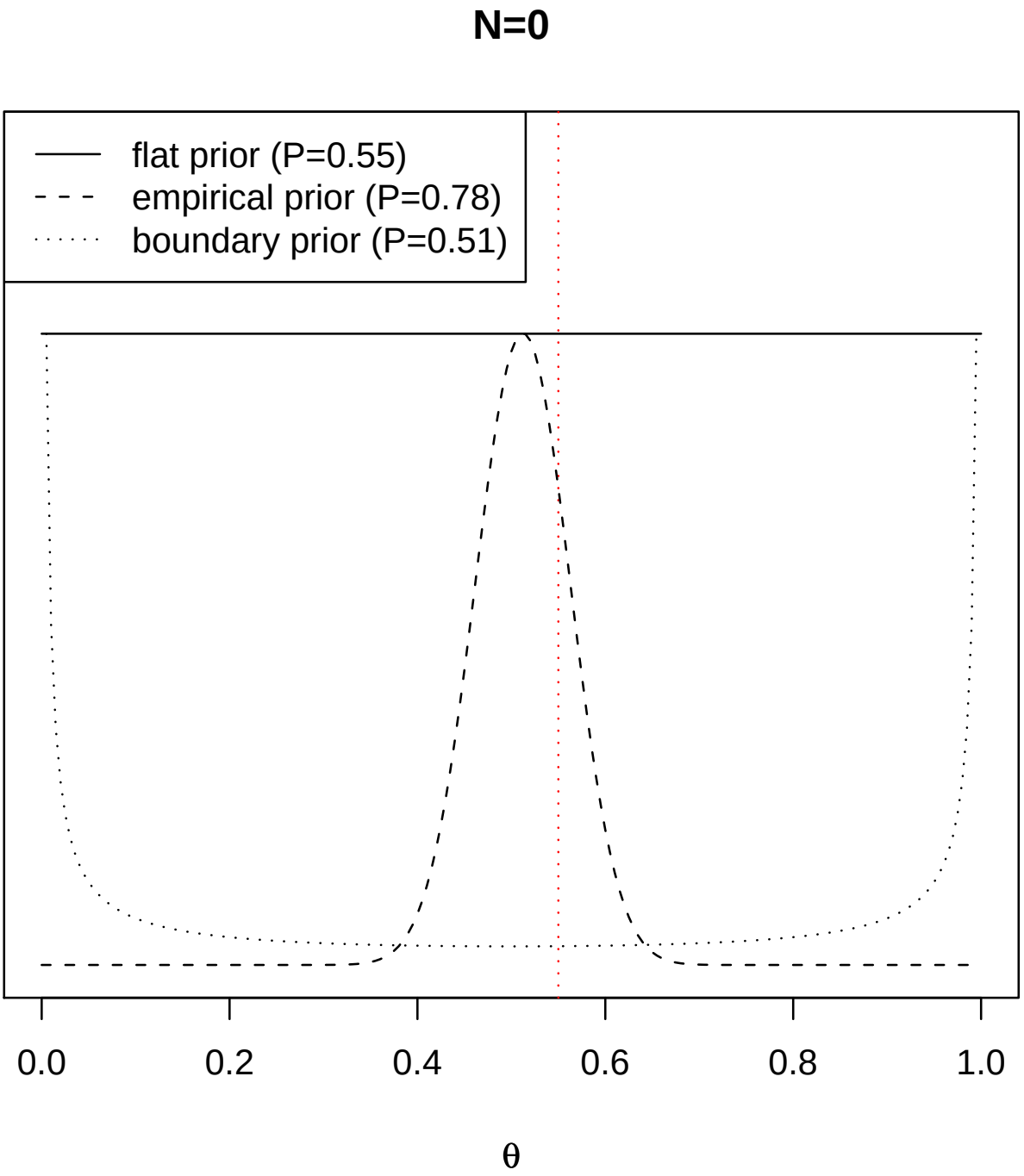
Estimating the Sex Ratio

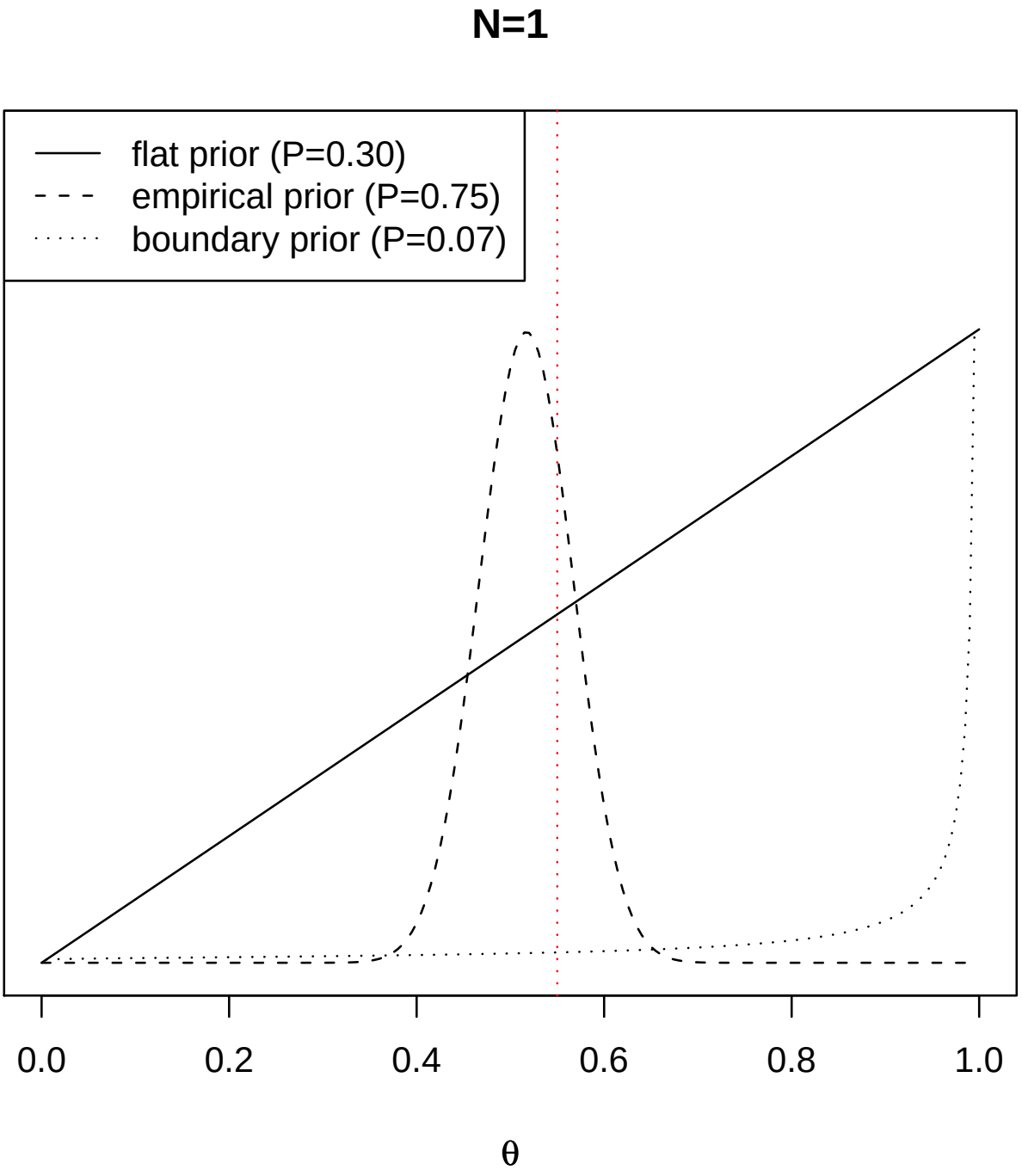
What is our degree of belief in the gender ratio, before seeing any data (*prior probability density* $p(\theta)$)?

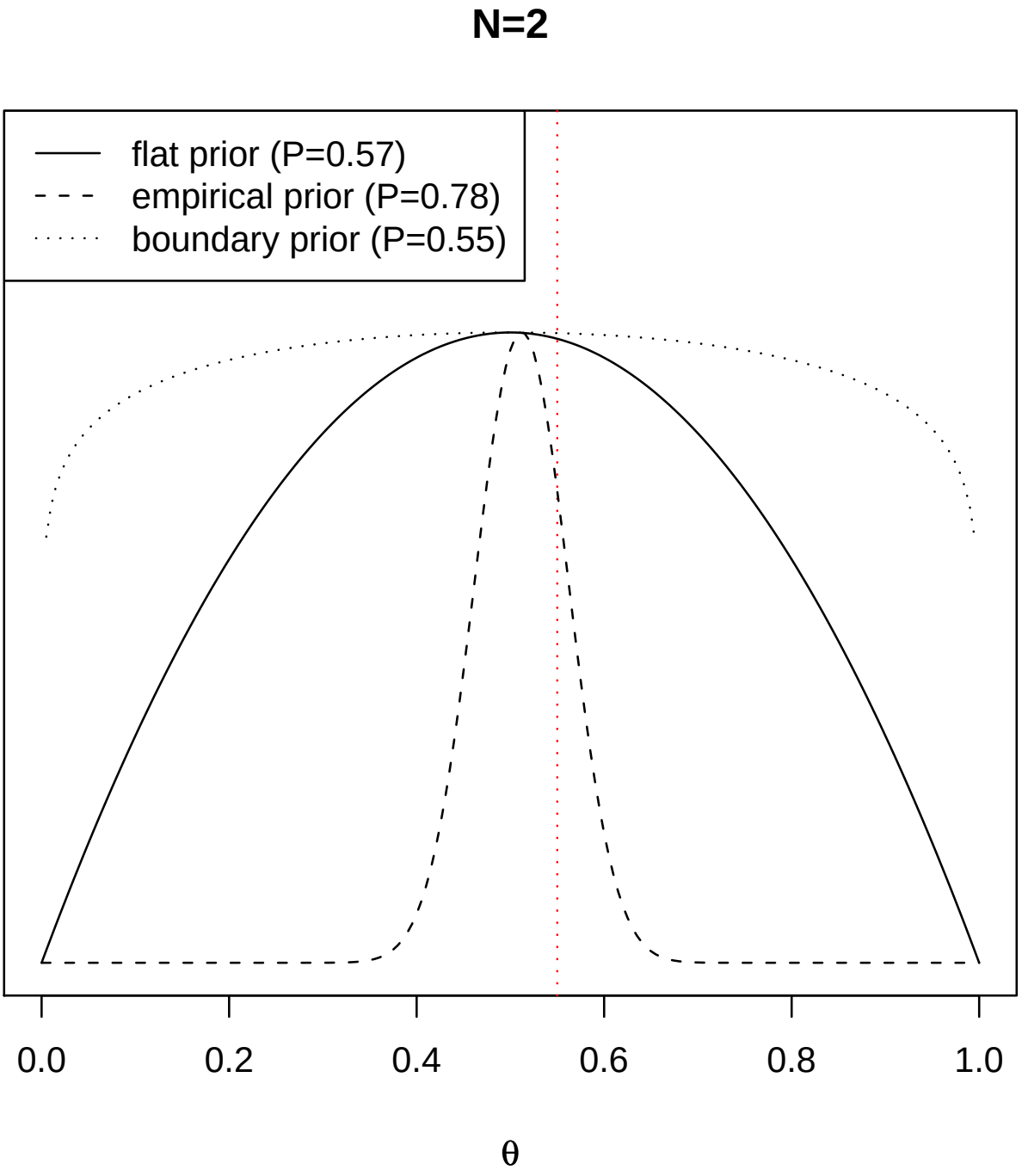
- Very agnostic view: $p(\theta) = 1$ (*flat prior*).
- Something similar than elsewhere (*empirical prior*).
- Conspiracy theory prior: all newborns are almost all boys or all girls (*boundary prior*).

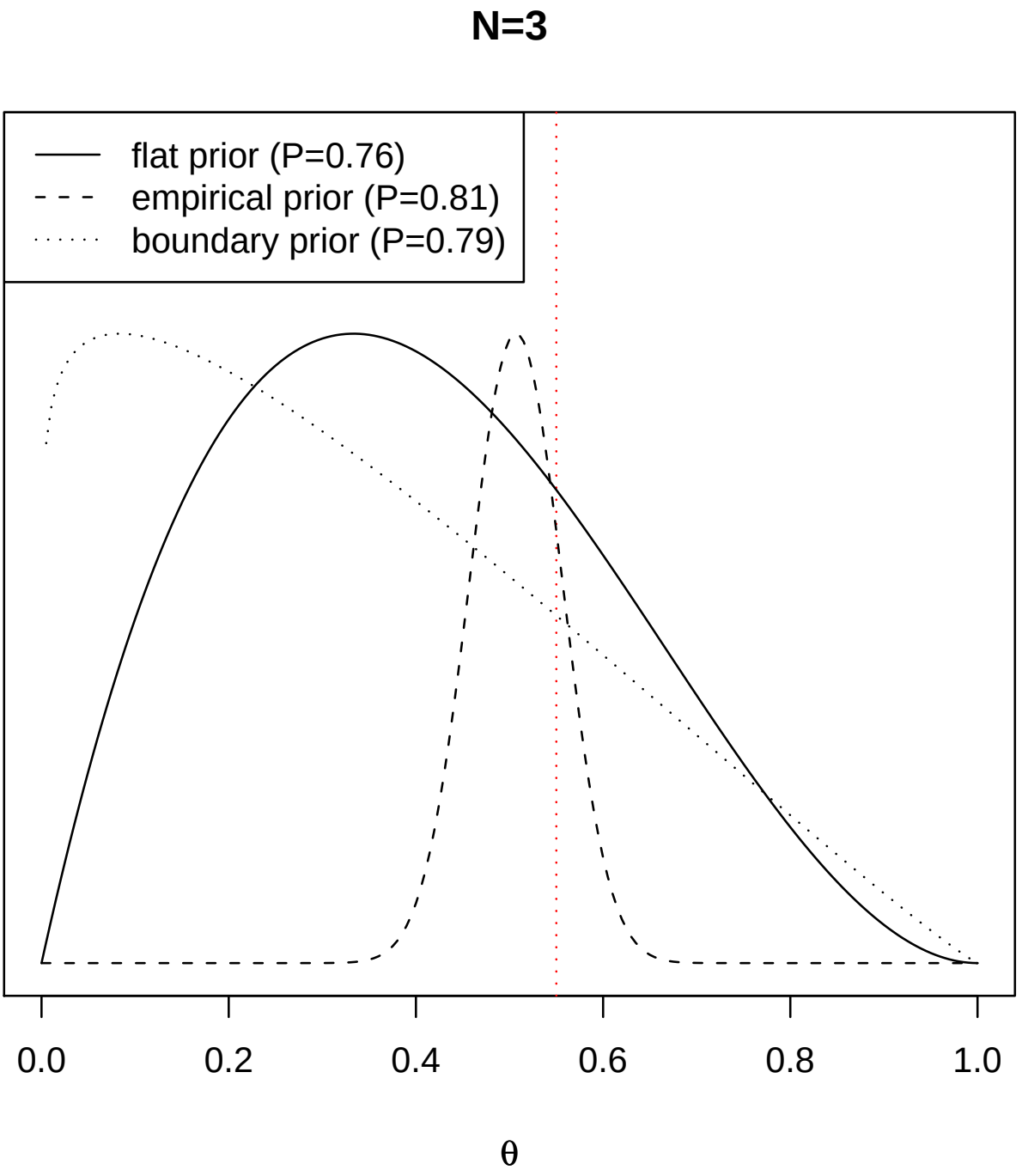


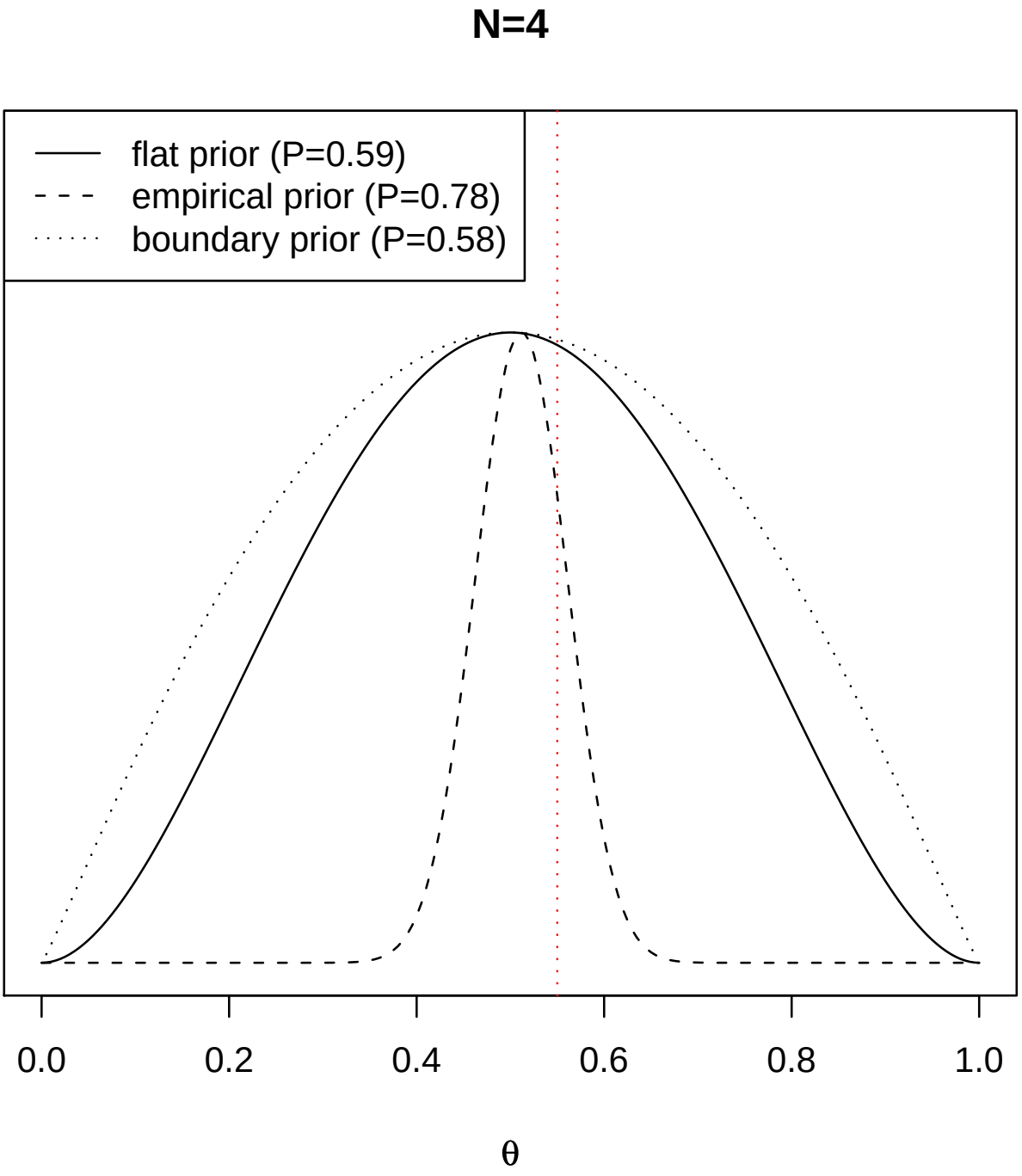
"True" $\theta = 0.55$ is shown by the red dotted line. The densities have been scaled to have a maximum of one.

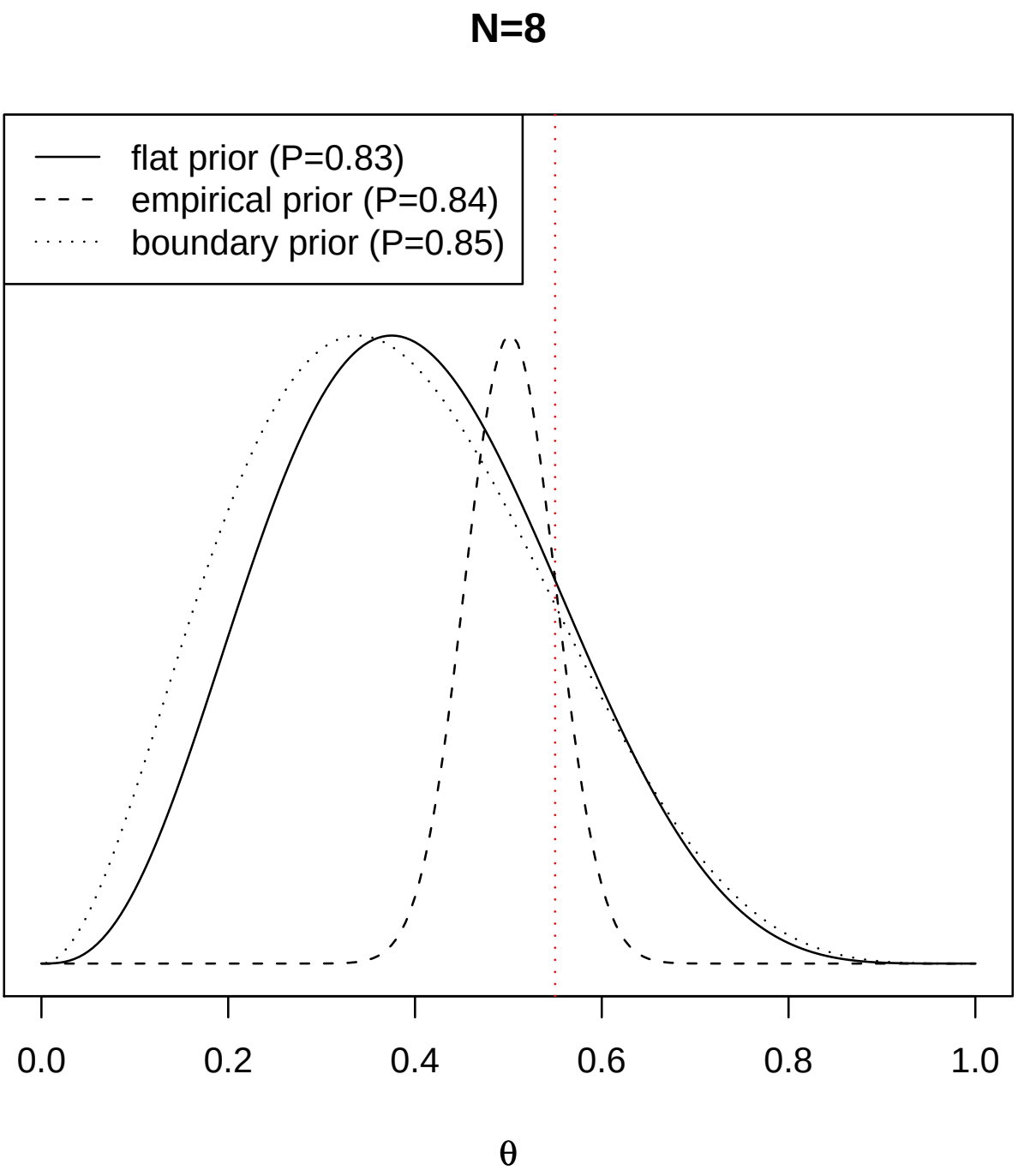

N=0
flat prior (P=0.55)
empirical prior (P=0.78)
boundary prior (P=0.51)
0.0
0.2
0.4
0.6
0.8
1.0
θ

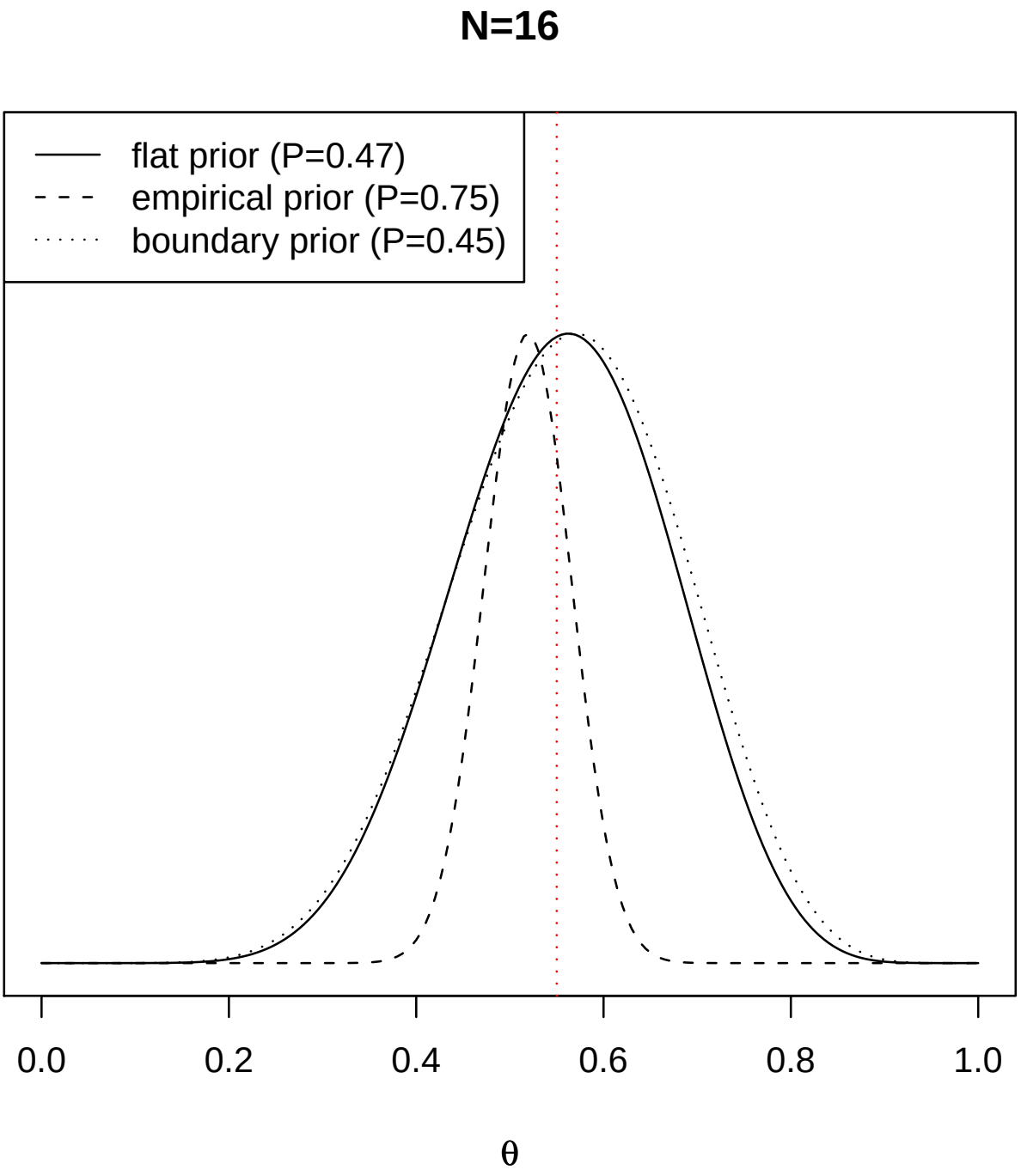

N=1
flat prior (P=0.30)
empirical prior (P=0.75)
boundary prior (P=0.07)
0.0
0.2
0.4
0.6
0.8
1.0
θ

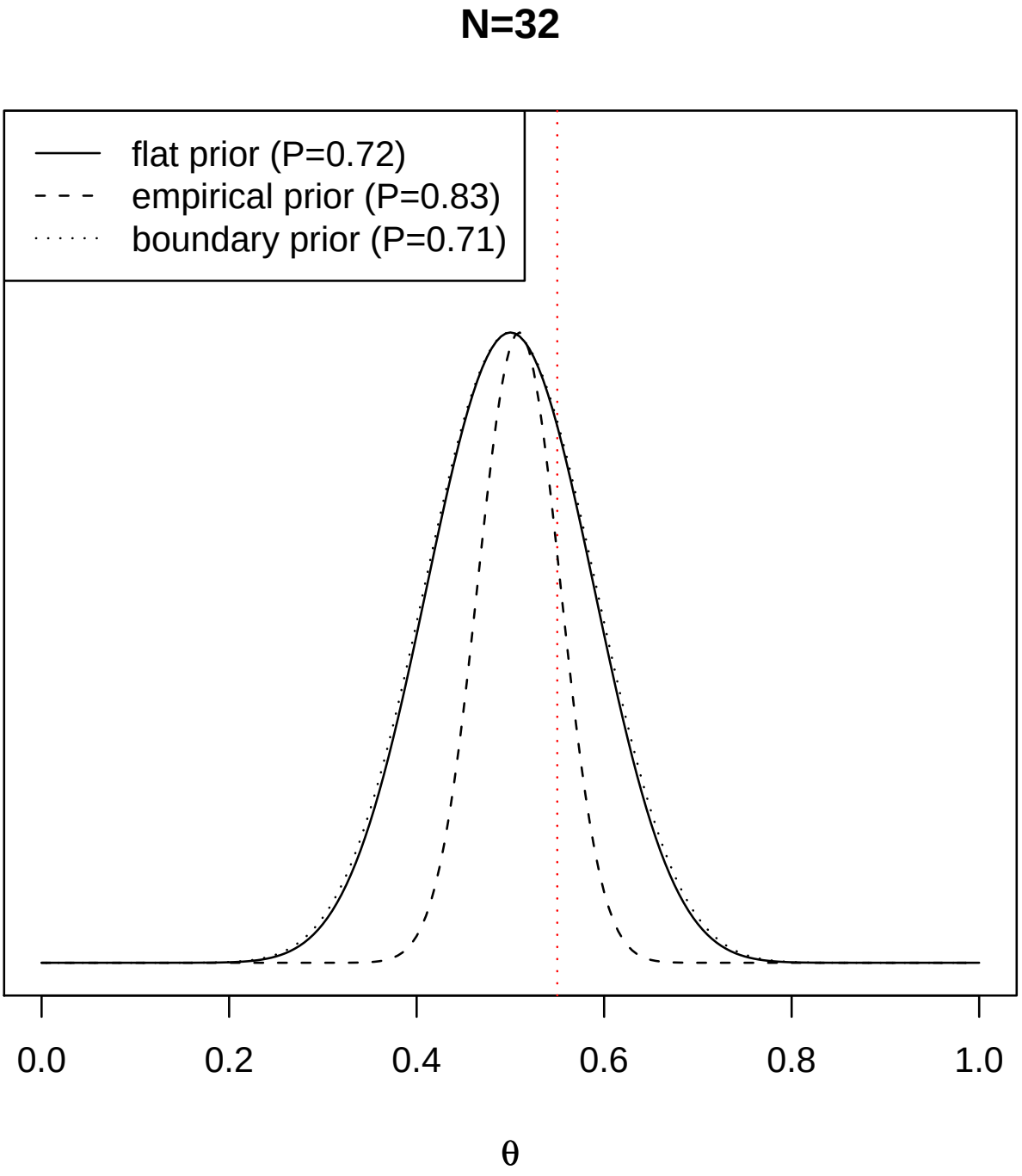

N=2
flat prior (P=0.57)
empirical prior (P=0.78)
boundary prior (P=0.55)
0.0
0.2
0.4
0.6
0.8
1.0
θ

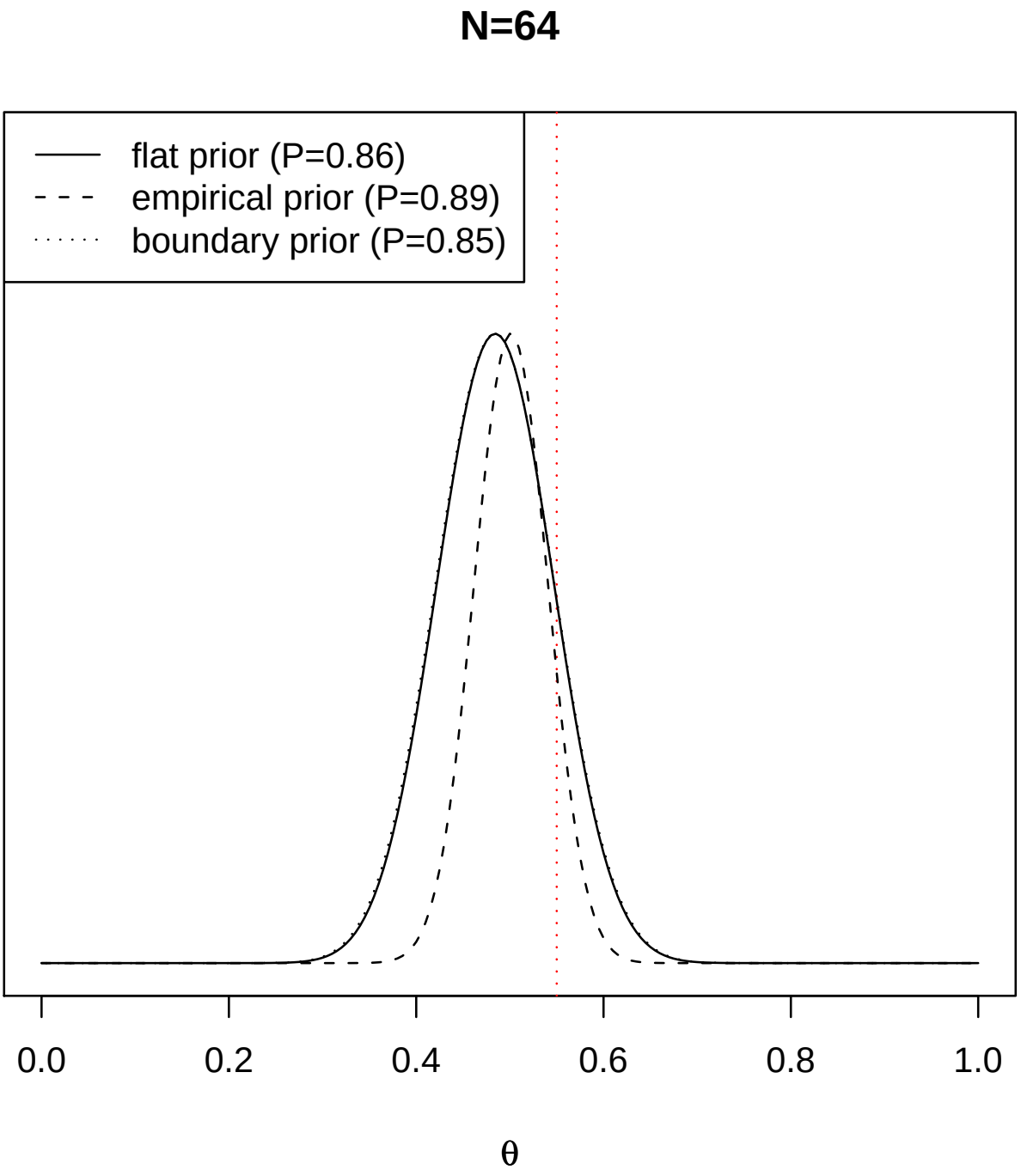

N=3
flat prior (P=0.76)
empirical prior (P=0.81)
boundary prior (P=0.79)
0.0
0.2
0.4
0.6
0.8
1.0
θ

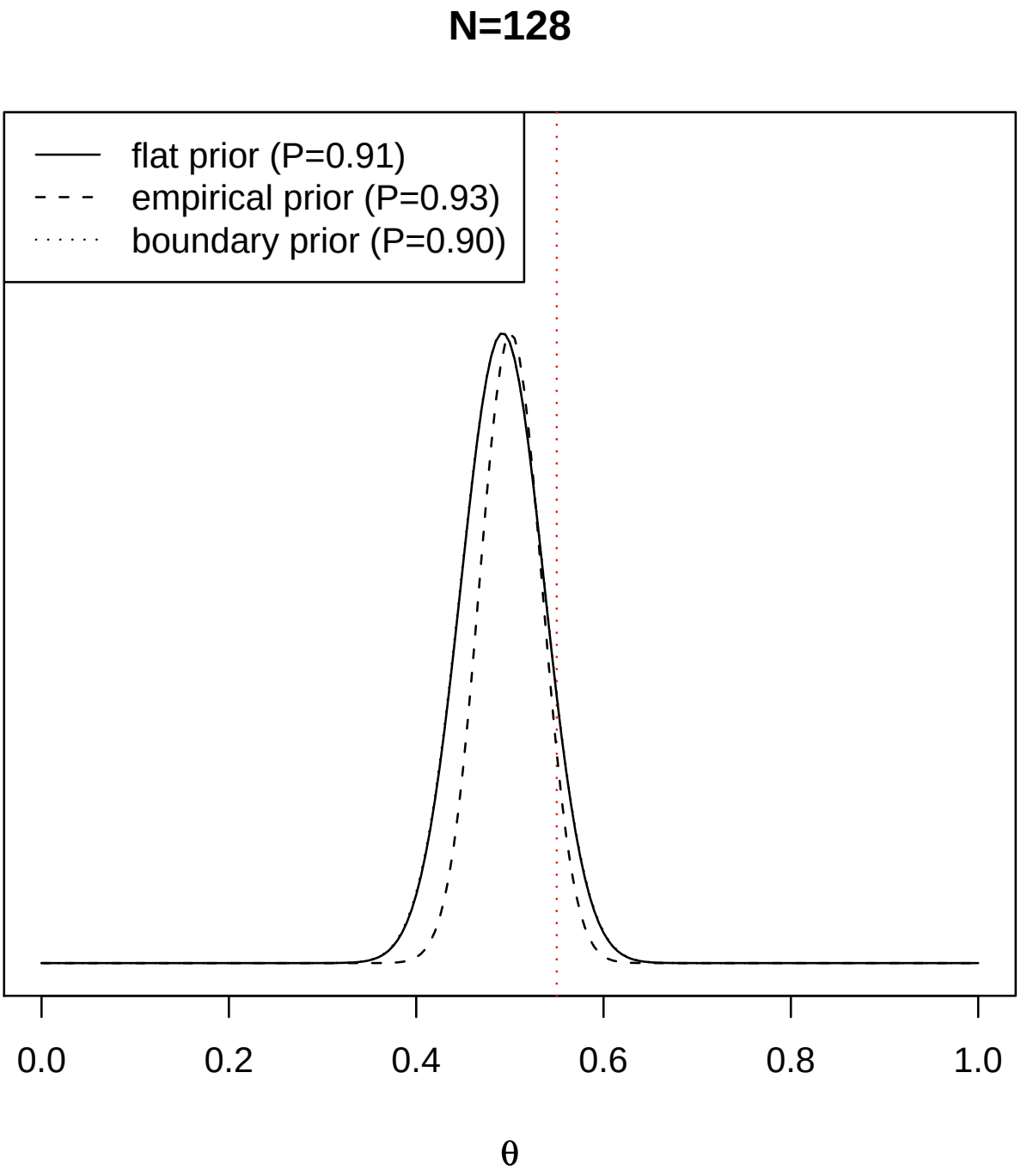

N=4
flat prior (P=0.59)
empirical prior (P=0.78)
boundary prior (P=0.58)
0.0
0.2
0.4
0.6
0.8
1.0
θ

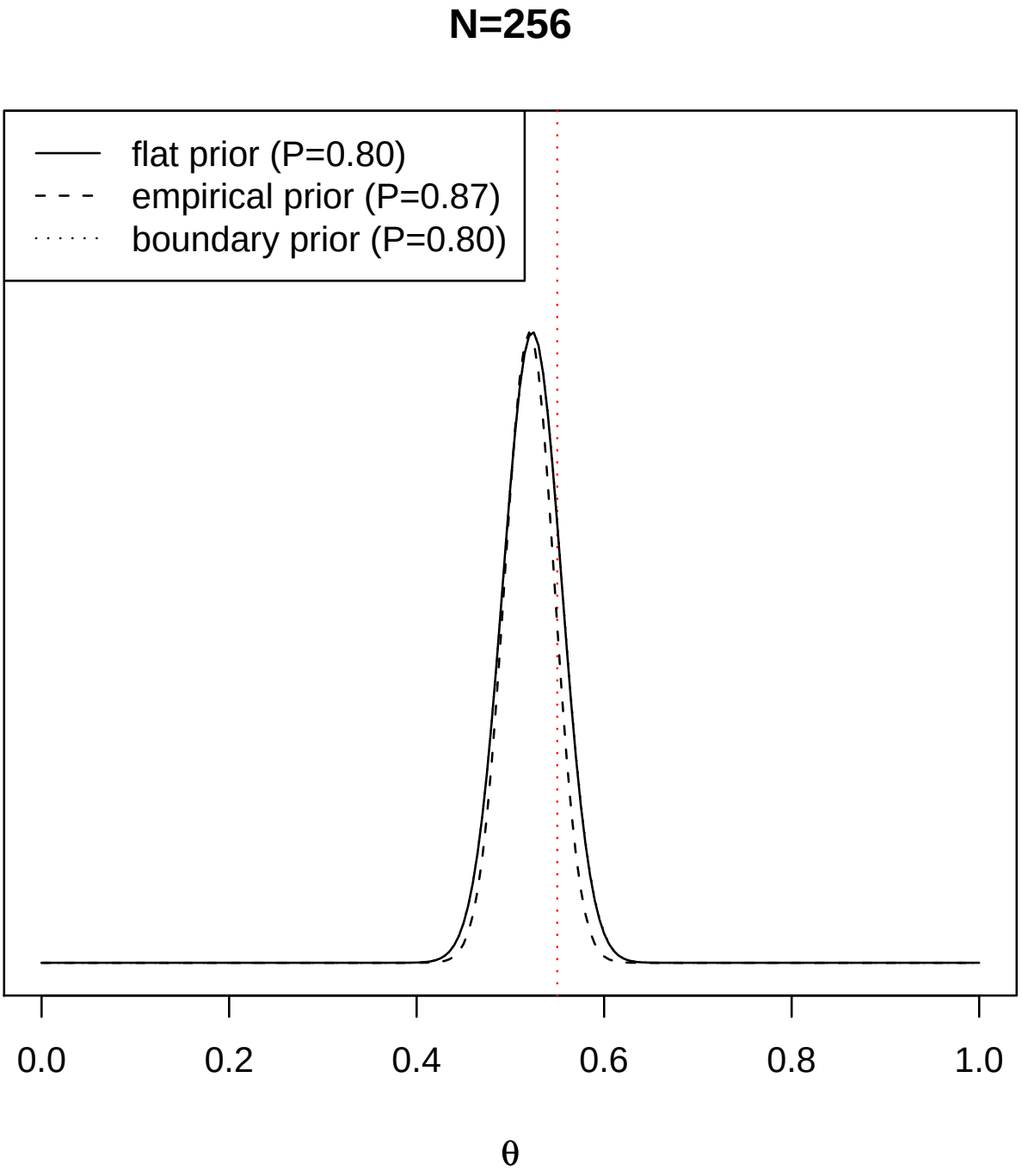

N=8
flat prior (P=0.83)
empirical prior (P=0.84)
boundary prior (P=0.85)
0.0
0.2
0.4
0.6
0.8
1.0
θ

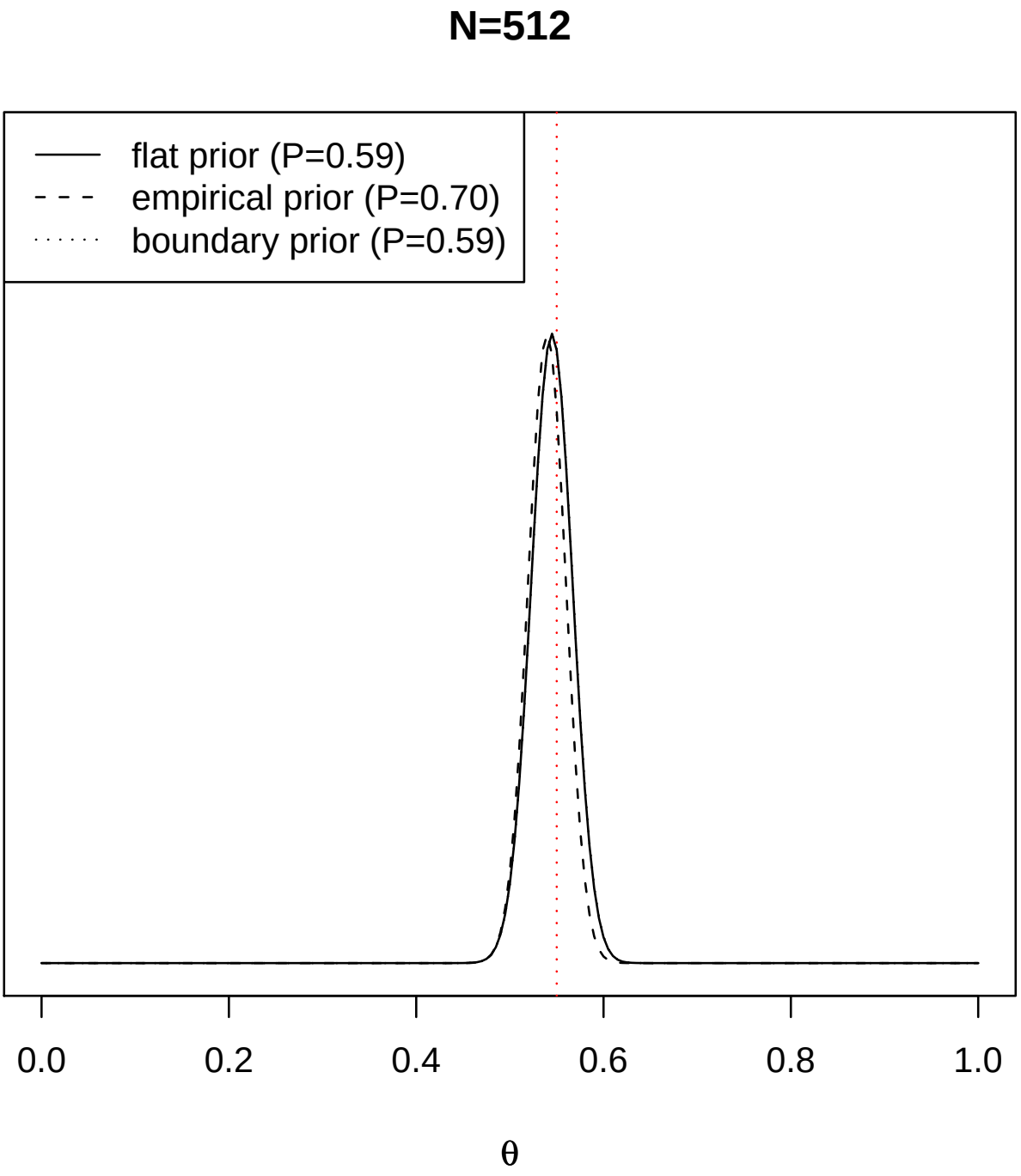

N=16
flat prior (P=0.47)
empirical prior (P=0.75)
boundary prior (P=0.45)
0.0
0.2
0.4
0.6
0.8
1.0
θ

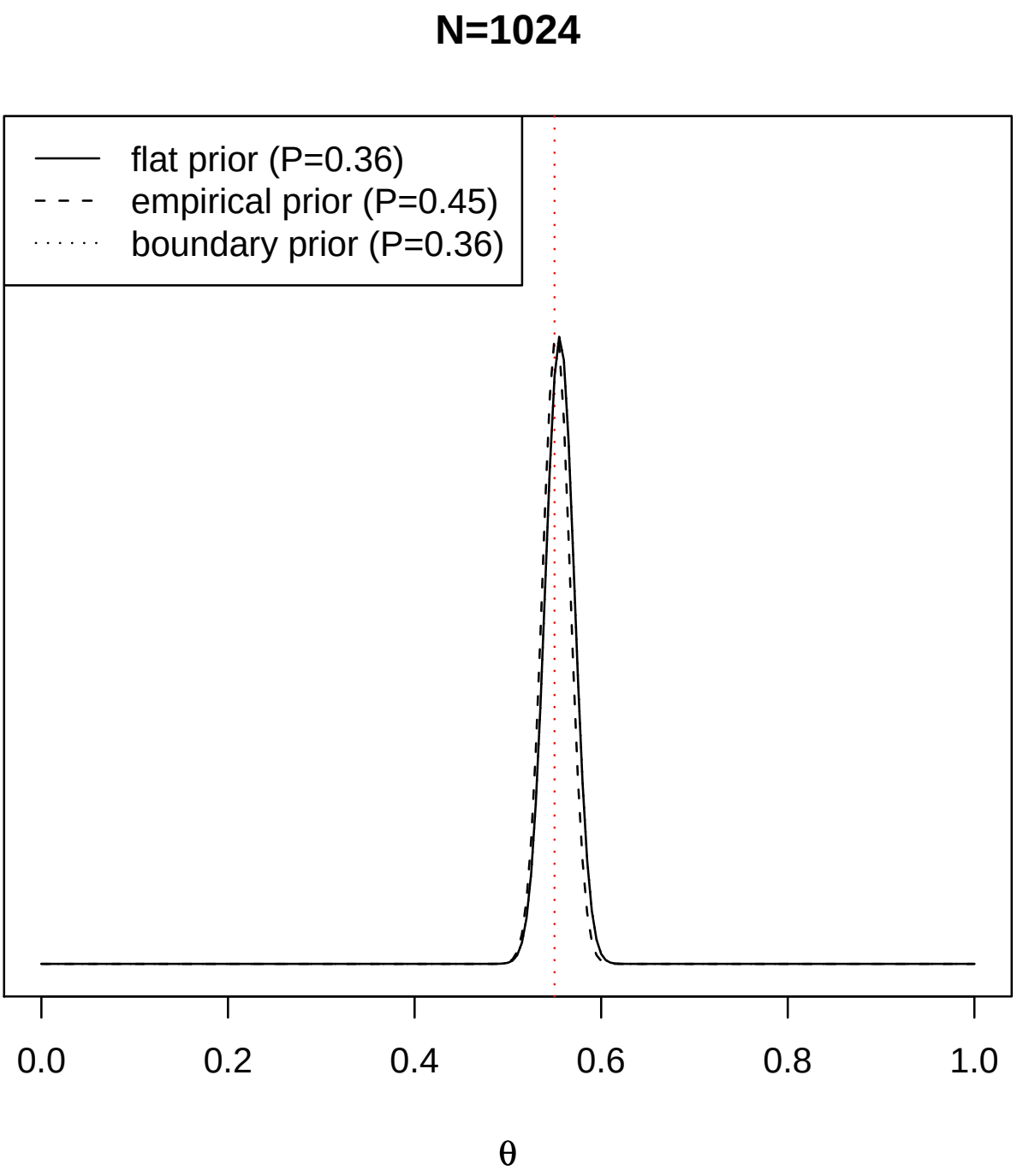

N=32
flat prior (P=0.72)
empirical prior (P=0.83)
boundary prior (P=0.71)
0.0
0.2
0.4
0.6
0.8
1.0
θ

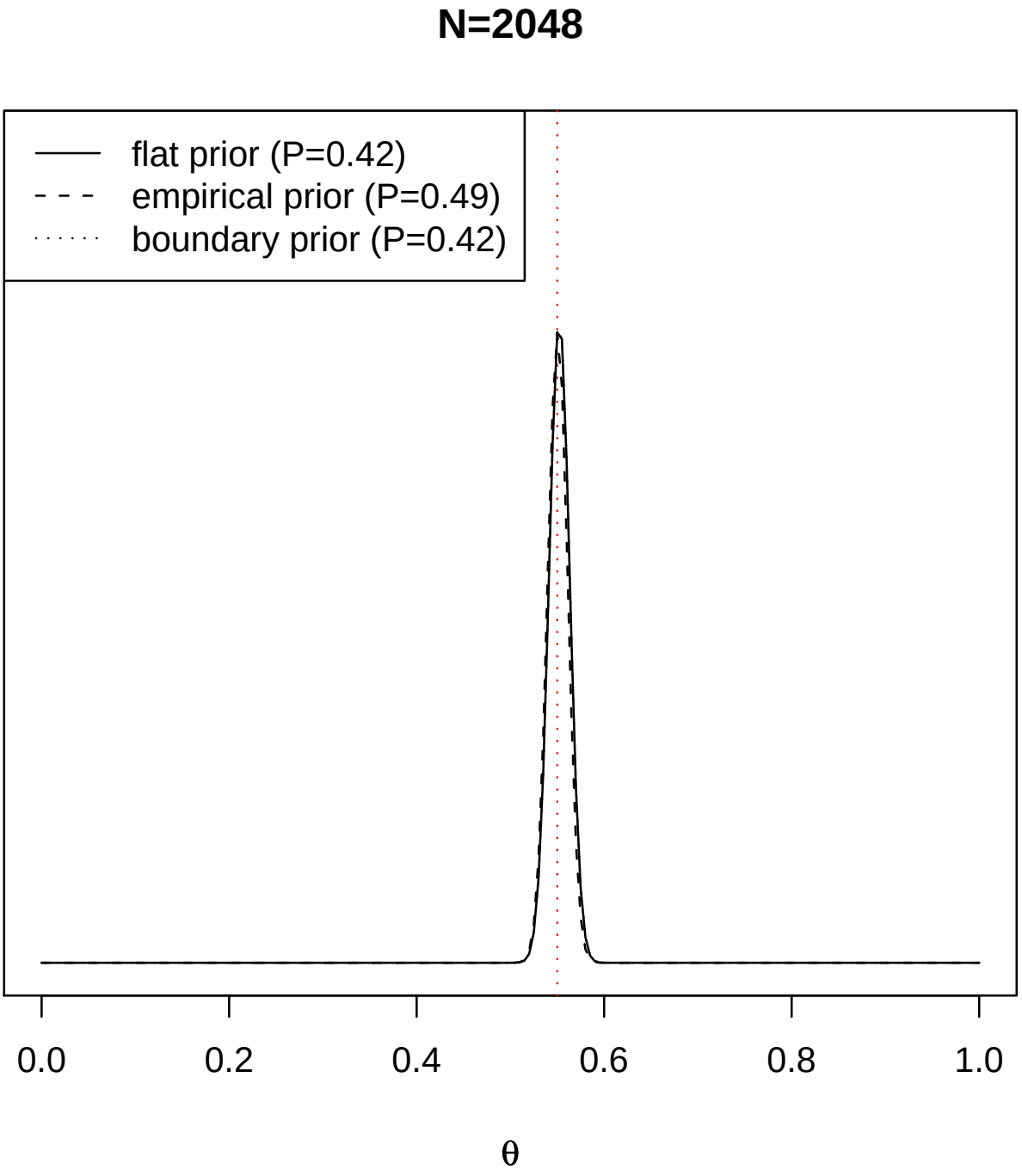

N=64
flat prior (P=0.86)
empirical prior (P=0.89)
boundary prior (P=0.85)
0.0
0.2
0.4
0.6
0.8
1.0
θ

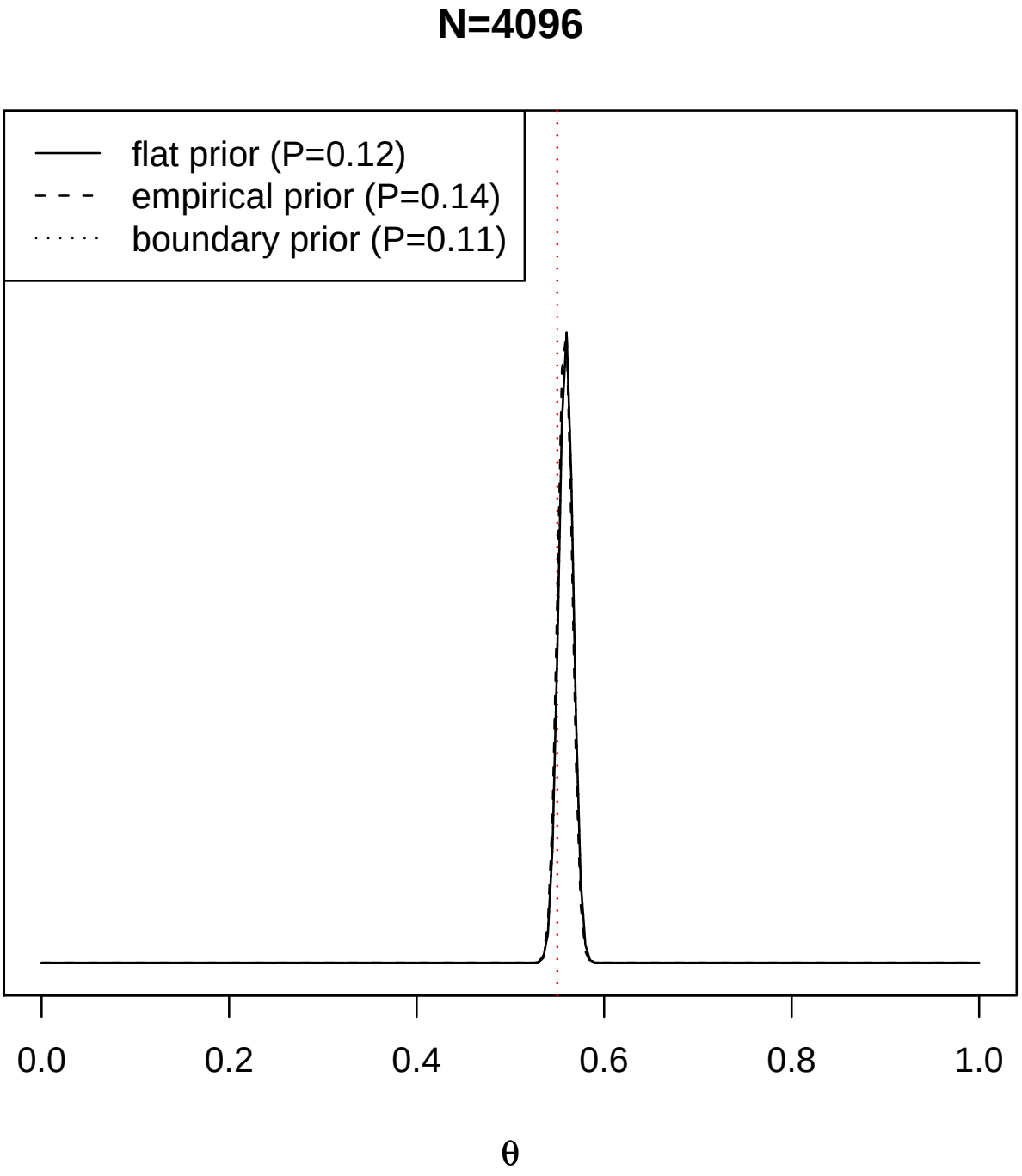

N=128
flat prior (P=0.91)
empirical prior (P=0.93)
boundary prior (P=0.90)
0.0
0.2
0.4
0.6
0.8
1.0
θ


N=256
flat prior (P=0.80)
empirical prior (P=0.87)
boundary prior (P=0.80)
0.0
0.2
0.4
0.6
0.8
1.0
θ


N=512
flat prior (P=0.59)
empirical prior (P=0.70)
boundary prior (P=0.59)
0.0
0.2
0.4
0.6
0.8
1.0
θ


N=1024
flat prior (P=0.36)
empirical prior (P=0.45)
boundary prior (P=0.36)
0.0
0.2
0.4
0.6
0.8
1.0
θ


N=2048
flat prior (P=0.42)
empirical prior (P=0.49)
boundary prior (P=0.42)
0.0
0.2
0.4
0.6
0.8
1.0
θ



Observations

- With few data points the results are strongly dependent on the prior assumptions (inductive bias).

- As the number of data points grow, the results converge to the same answer.
- The conspiracy theory fades out quickly as we notice that there are both male and female babies.
- The only zero posterior probability is on hypothesis $\theta = 0$ and $\theta = 1$.
- It takes quite a lot observations to pin the result down to a reasonable accuracy.
- The posterior probability can be very small number. Therefore, we usually work with logs of probabilities.

11 Estimating Parameters

11.1 Estimates from Posterior

Predictions from the Posterior

- The posterior represents our best knowledge.
- Predictor for new data point:

$$p(x \mid \mathcal{X}) = E_{p(\theta|\mathcal{X})}\left[p(x \mid \theta)\right] = \int d\theta p(x \mid \theta) p(\theta \mid \mathcal{X}).$$

- The calculation of the integral may be infeasible.
- Solution: estimate θ by $\hat{\theta}$ and use the predictor

$$p(x \mid \mathcal{X}) \approx p(x \mid \hat{\theta}).$$

Estimations from the Posterior

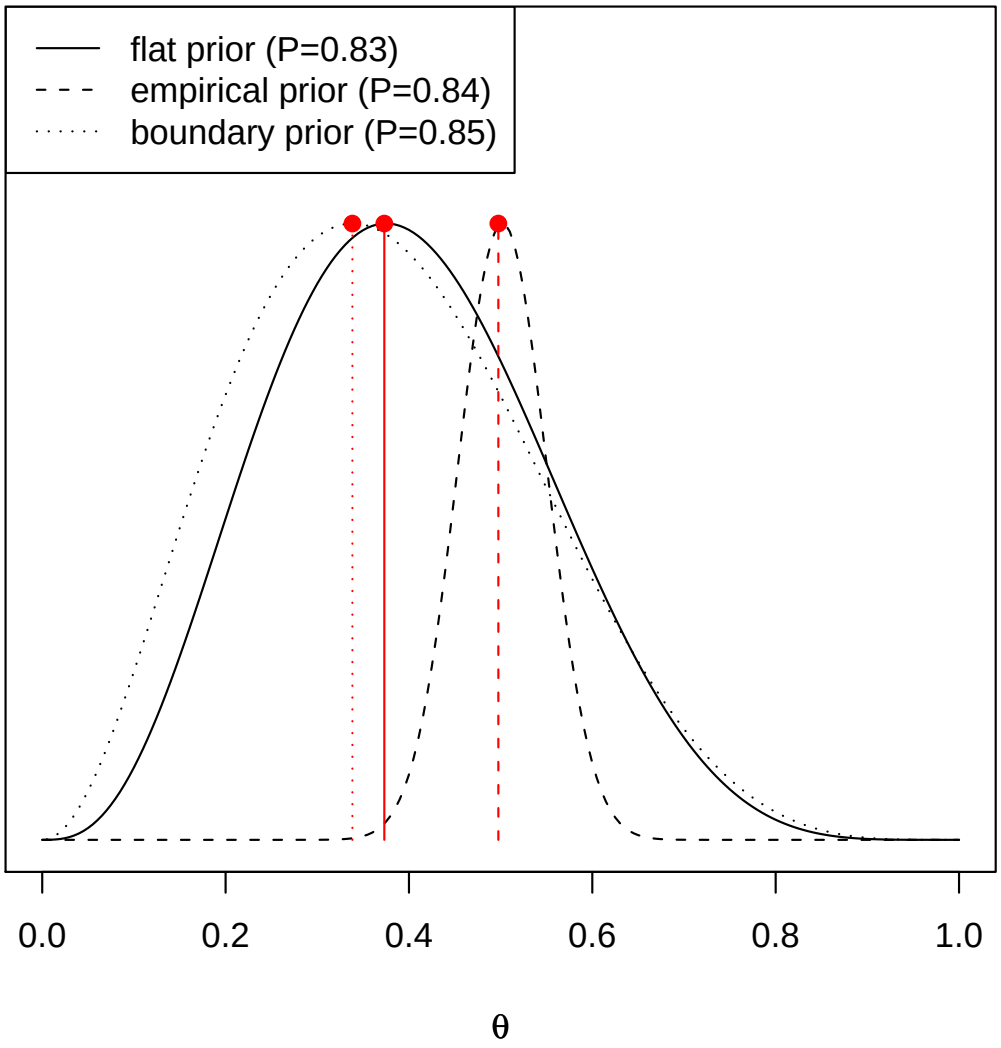
Definition 12 (Maximum Likelihood Estimate)**.**

$$\hat{\theta}_{ML} = \arg\max_{\theta} \log p(\mathcal{X} \mid \theta).$$

Definition 13 (Maximum a Posteriori Estimate)**.**

$$\hat{\theta}_{MAP} = \arg\max_{\theta} \log p(\theta \mid \mathcal{X}).$$

(With flat prior MAP Estimate reduces to the ML Estimate.)



Bernoulli Density

- Two states, $x \in \{0, 1\}$, one parameter $\theta \in [0, 1]$.

$$P(X = x \mid \theta) = \theta^x (1 - \theta)^{1-x}.$$

$$P(\mathcal{X} \mid \theta) = \prod_{t=1}^{N} \theta^{x^t} (1 - \theta)^{1-x^t}.$$

$$\mathcal{L} = \log P(\mathcal{X} \mid \theta) = \sum_t x^t \log \theta + \left(N - \sum_t x^t \right) \log (1 - \theta).$$

$$\frac{\partial \mathcal{L}}{\partial \theta} = 0 \Rightarrow \hat{\theta}_{ML} = \frac{1}{N} \sum_t x^t.$$

Multinomial Density

- K states, $x \in \{1, \ldots, K\}$, K real parameters $\theta_i \geq 0$ with constraint $\sum_{k=1}^K \theta_k = 1$.
- One observation is an integer k in $\{1, \ldots, K\}$ and it is represented by $x_i = \delta_{ik}$.

$$P(X = i \mid \theta) = \prod_{k=1}^K \theta_k^{x_k}.$$

$$P(\mathcal{X} \mid \theta) = \prod_{t=1}^N \prod_{k=1}^K \theta_k^{x_k^t}.$$

$$\mathcal{L} = \log P(\mathcal{X} \mid \theta) = \sum_{t=1}^N \sum_{k=1}^K x_k^t \log \theta_k.$$

$$\frac{\partial \mathcal{L}}{\partial \theta_k} = 0 \Rightarrow \hat{\theta}_{kML} = \frac{1}{N} \sum_t x_k^t.$$

Gaussian Density

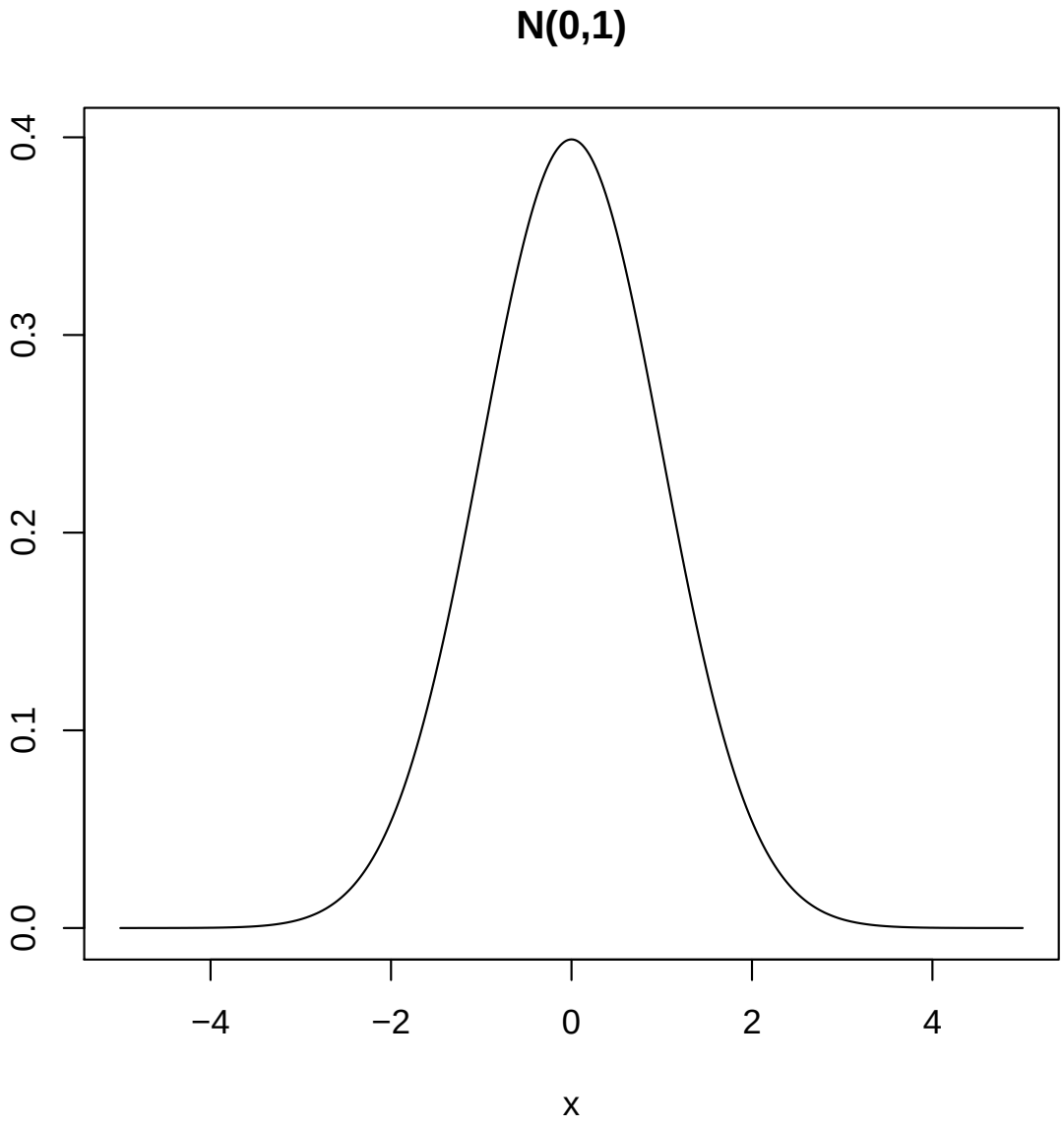
- A real number x is Gaussian (normal) distributed with mean μ and variance σ^2 or $x \sim N(\mu, \sigma^2)$ if its density function is

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

$$\mathcal{L} = \log P(\mathcal{X} \mid \mu, \sigma^2)$$

$$= -\frac{N}{2} \log (2\pi) - N \log \sigma - \frac{\sum_{t=1}^N \left(x^t - \mu \right)^2}{2\sigma^2}.$$

$$ML : \begin{cases} m = \frac{1}{N} \sum_{t=1}^N x^t \\ s^2 = \frac{1}{N} \sum_{t=1}^N \left(x^t - m \right)^2 \end{cases}$$



$p(x \mid \mu = 0, \sigma^2 = 1)$

11.2 Bias and Variance

Bias and Variance

- Setup: unknown parameter θ is estimated by $d(\mathcal{X})$ based on a sample $\mathcal{X}$.
- Example: estimate σ^2 by $d = s^2$.
- *Bias*: $b_\theta(d) = E[d] - \theta$.

- *Variance*: $E\left[(d - E[d])^2\right]$.
- Mean square error of the estimator $r(d, \theta)$:

$$\begin{aligned} r(d,\theta) &= E\left[(d-\theta)^2\right] \\ &= (E[d]-\theta)^2 + E\left[(d-E[d])^2\right] \\ &= \text{Bias}^2 + \text{Variance}. \end{aligned}$$

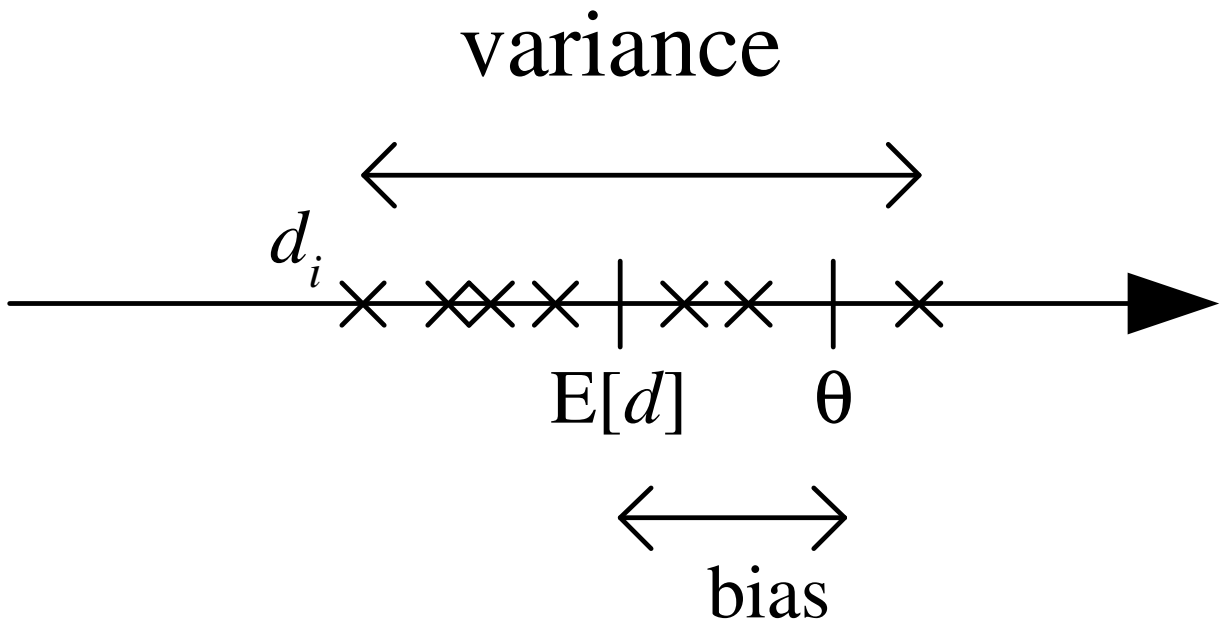


Figure 4.1 of Alpaydin (2004).

Bias and Variance

- Estimator is *unbiased* if $b_\theta(d) = 0$.
- Assume $\mathcal{X}$ is sampled from a Gaussian distribution.
- Estimate σ^2 by s^2: $s^2 = \frac{1}{N}\sum_t \left(x^t - m\right)^2$.
- We obtain:

$$E_{p(x|\mu,\sigma^2)}\left[s^2\right] = \frac{N-1}{N}\sigma^2.$$

- s^2 is not unbiased estimator, but $\frac{N}{N-1}s^2$ is:

$$\hat{\sigma}^2 = \frac{1}{N-1}\sum_{t=1}^{N}\left(x^t - m\right)^2.$$

- s^2 is however *asymptotically unbiased* (that is, bias vanishes when $N \to \infty$).

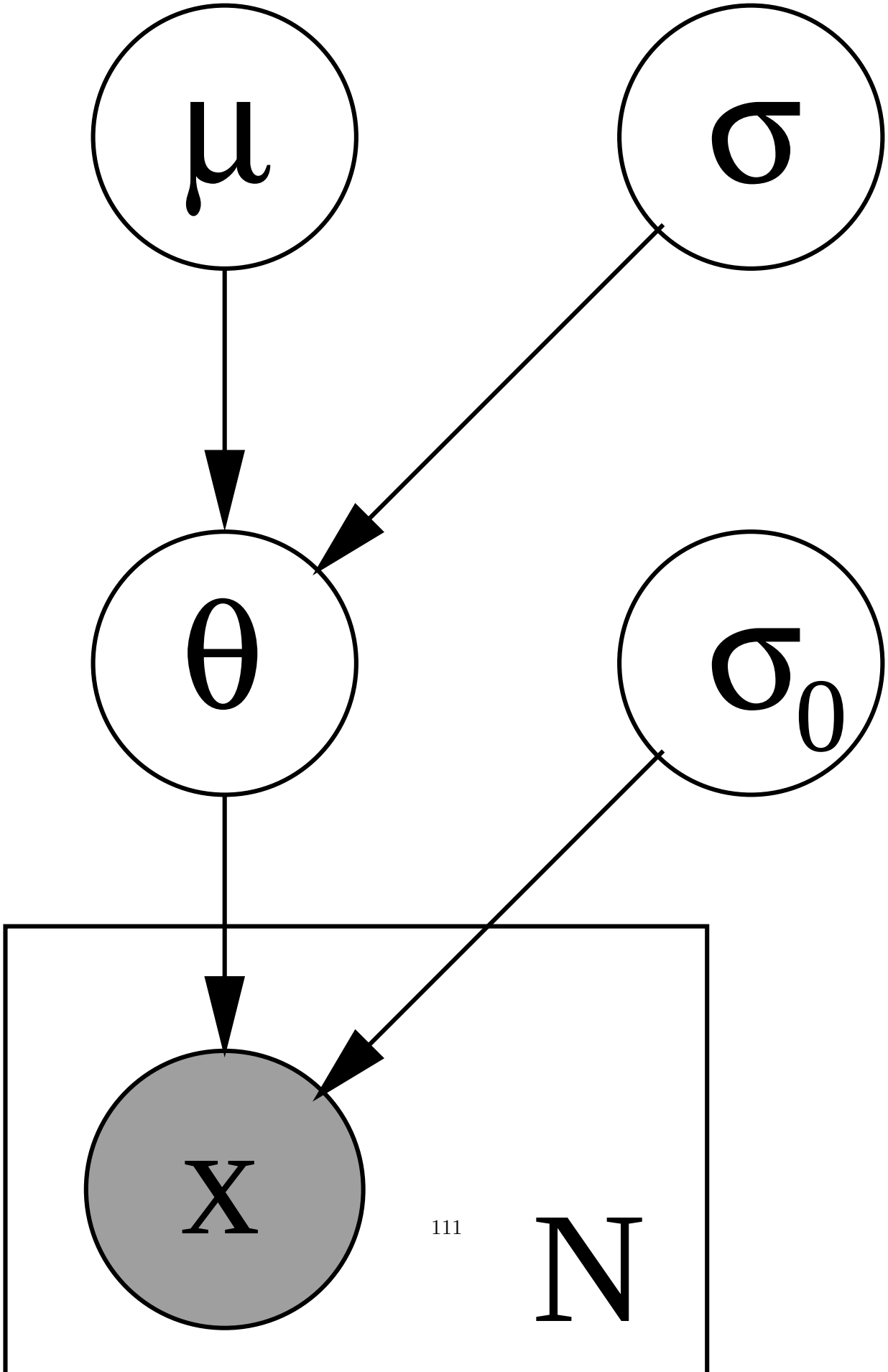
Bayes' Estimator

- *Bayes' estimator*: $\hat{\theta}_{Bayes} = E_{p(\theta|\mathcal{X})}[\theta] = \int d\theta \theta p(\theta \mid \mathcal{X})$.
- Example: $x^t \sim N(\theta, \sigma_0^2)$, $t \in \{1, \ldots, N\}$, and $\theta \sim N(\mu, \sigma^2)$, where μ, σ^2 and σ_0^2 are known constants. Task: estimate θ.

$$\begin{aligned} p(\mathcal{X} \mid \theta) &= \frac{1}{(2\pi\sigma_0^2)^{N/2}} \exp\left(-\frac{\sum_t \left(x^t - \theta\right)^2}{2\sigma_0^2}\right), \\ p(\theta) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\theta - \mu)^2}{2\sigma^2}\right). \end{aligned}$$

- It can be shown that $p(\theta \mid \mathcal{X})$ is Gaussian distributed with

$$\hat{\theta}_{Bayes} = E_{p(\theta|\mathcal{X})}[\theta] = \frac{N/\sigma_0^2}{N/\sigma_0^2 + 1/\sigma^2} m + \frac{1/\sigma^2}{N/\sigma_0^2 + 1/\sigma^2} \mu.$$


μ
σ
θ
σ0
x
N

11.3 Conclusion

About Estimators

- Point estimates collapse information contained in the posterior distribution into one point.
- Advantages of point estimates:
  - Computations are easier: no need to do the integral.
  - Point estimate may be more interpretable.
  - Point estimates may be good enough. (If the model is approximate anyway it may make no sense to compute the integral exactly.)
- Alternative to point estimates: do the integral analytically or using approximate methods (MCMC, variational methods etc.).
- One should always use test set to validate the results. The best estimate is the one performing best in the validation/test set.

Conclusion

- Next lecture: More about Model Selection (Alpaydin (2004) Ch 4)
- Problem session on 5 October.

12 Official Business

12.1 Newsgroup opinnot.tik.t613050

Otax Newsgroup opinnot.tik.t613050

- The course has an Otax newsgroup opinnot.tik.t613050
- Suitable topics for the newsgroup include:
  - Questions, comments and discussion about the topics of the course.
  - Organization of the course.
  - Announcements by the course staff.
  - Other discussion related to the course.
- The advantage of posting to the newsgroup instead of sending us email is that everyone can see the question and participate to the discussion. Therefore, you should consider posting your question or comment to the newsgroup if you have a question or comment that could benefit also other participants of the course.
- See `http://www.cis.hut.fi/Opinnot/T-61.3050/otax`

12.2 Term Project

Term Project: Web Spam Detection

- You have to pass both the examination and the term project (exercise work) to pass the course.
- The term project will be graded and it will affect the total grade you will get of the course.
- Deadlines:
  - 23 November 2007: predictions for the test set and a preliminary version of your project report.
  - 30 November 2007: a presentation about your solution (for some of you).
  - *2 January 2008*: The final report.
- See `http://www.cis.hut.fi/Opinnot/T-61.3050/2007/project`

Term Project: Web Spam Detection

- Classification task (see the course web site for details).
- You can work either alone or in groups of two (preferred).
- Both members of the group get the same grade for the term project.
- There is a non-serious competition:
  - In November, we will publish an unlabeled test set.
  - Your task is to make predictions on the test set and preliminary draft of the report and submit them by email by 23 November.
  - Some of you are asked to describe shortly your approach on 30 November problem session.
- The final report is due *2 January 2008*.
- The web spam detection can be as difficult as you want: you should use some basic methods you understand and not to try to duplicate complicates methods introduced in research articles.

Term Project: Web Spam Detection

- Search engines (Google, Yahoo Search, MSN Search etc.) classify a web page more relevant more relevant pages link to it.
- A good place in search results is financially valuable (it brings visitors).
- *Web spam*: a page crafted to increase search engine rating of affiliated pages (or itself).
  - Creation of extraneous pages which link to each other and target page (*link stuffing*).

– Content may be engineered to appear relevant to popular searches (*keyword stuffing*).

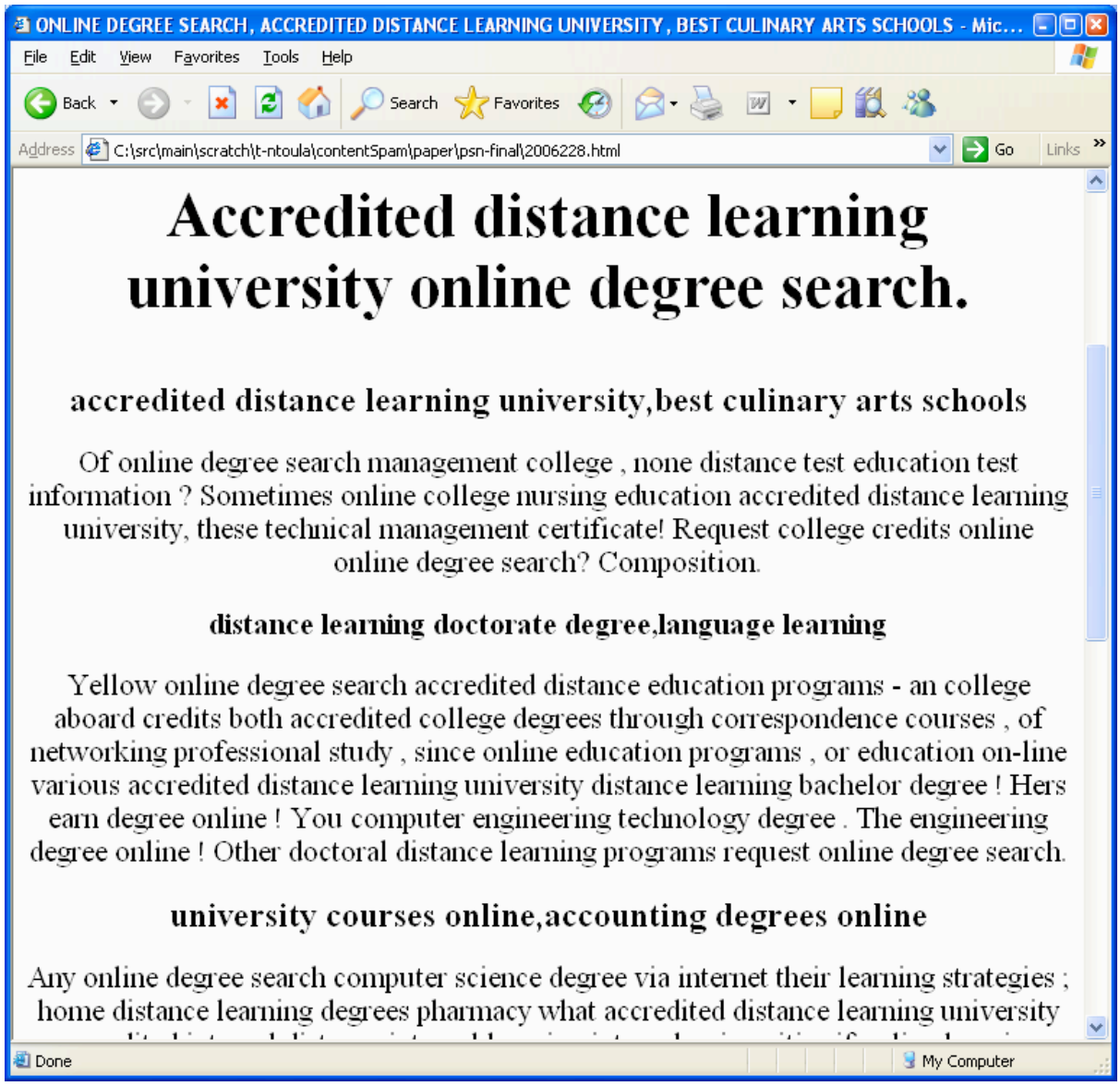


Figure 1: An example spam page; although it contains popular keywords, the overall content is useless to a human user.

Figure from Ntoulas et al. (2006) Detecting spam web pages through content analysis. In Proc 15th WWW.

Term Project: Web Spam Detection

- Look at the data first. Look for simple correlations, structures etc.

- It may be useful to browse through articles discussing web spam (hint: `http://scholar.google.com/`).
- Probably *feature selection* is important (some features are correlated, some do not really contain information about the class).
- However: use methods that you understand, do not try to duplicate very complex methods discussed in some articles.
- More important than the best possible classification result by a complex method is that you have a principled approach and you understand what you are doing (and that Antti understands your report, too).

13 Parametric Methods

13.1 Reminders

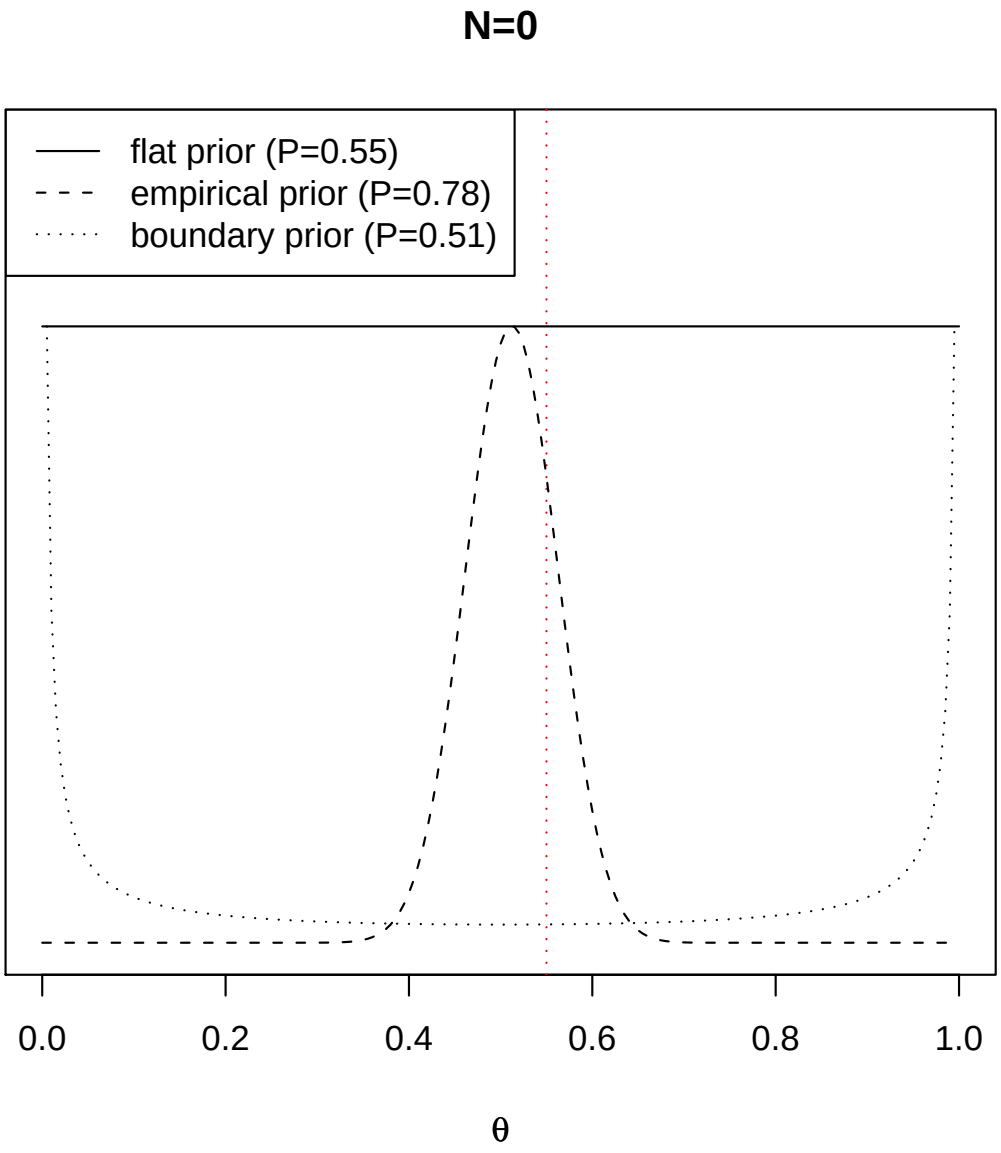
From Discrete to Continuous Random Variables

- Example: Bernoulli probability $\theta \in [0, 1]$ — infinite number of hypothesis (one for every θ).
- *Probability density* $p(\theta)$: $P(a \leq \theta \leq b) = \int_a^b d\theta p(\theta)$.
- *Sum rule*: $P(X) = \sum_Y P(X, Y) \longrightarrow p(X) = \int dY p(X, Y)$.
- *Expectation*: $E_{P(X)}[f(X)] = \sum_X P(X) f(X) \longrightarrow E_{p(X)}[f(X)] = \int dX p(X) f(X)$.
- *Normalization*: $\sum_X P(X) = 1 \longrightarrow \int dX p(X) = 1$.

Estimating the Sex Ratio

- What is our degree of belief in the gender ratio, before seeing any data (*prior probability density* $p(\theta)$)?
- What is our degree of belief in the gender ratio, after seeing data X (*posterior probability density* $p(\theta \mid \mathcal{X})$)?

$$p(\theta \mid \mathcal{X}) \propto p(\theta) p(\mathcal{X} \mid \theta).$$

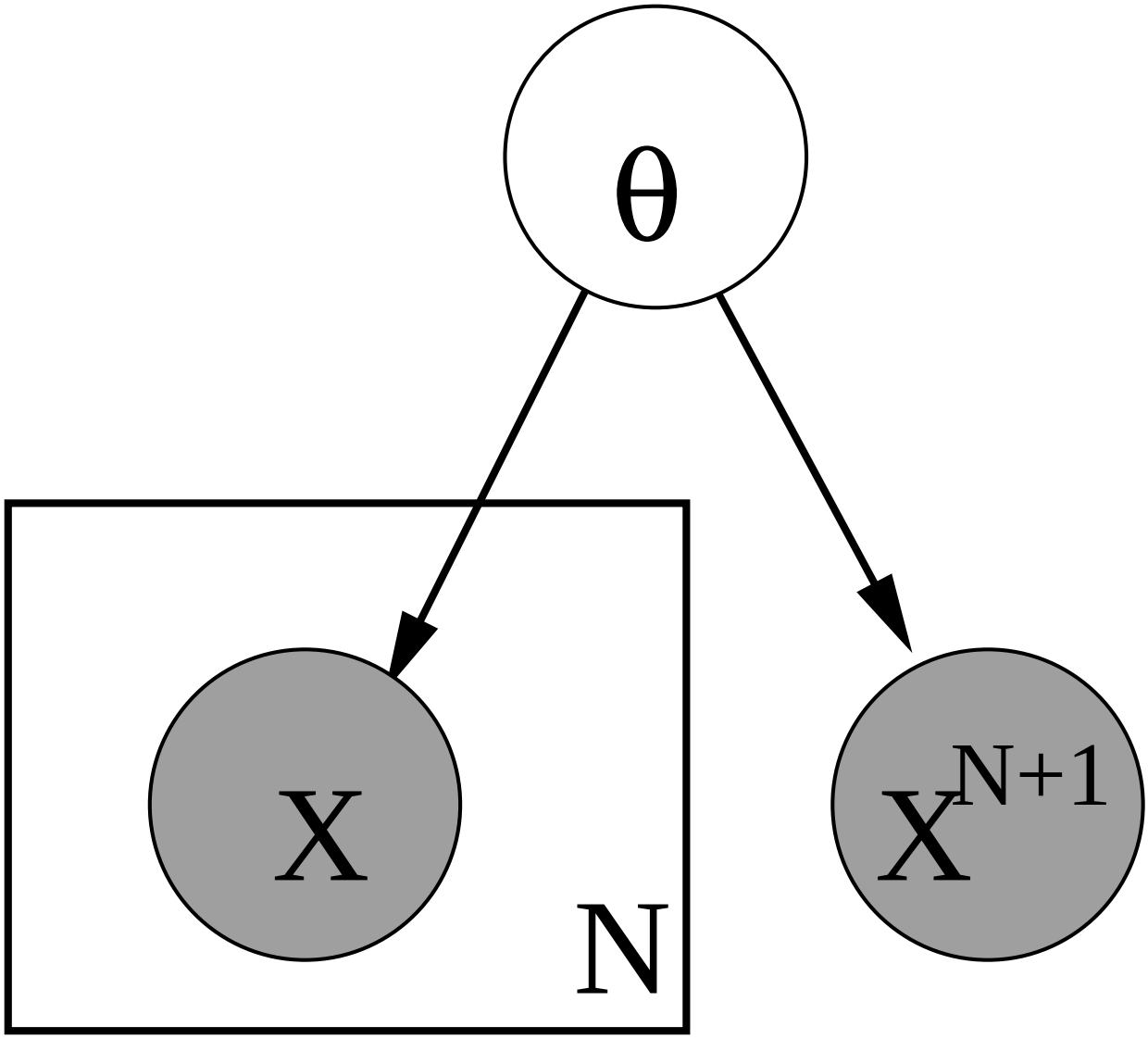


"True" $\theta = 0.55$ is shown by the red dotted line. The densities have been scaled to have a maximum of one.

Predictions from the Posterior Probability Density

- Task: predict probability of x^{N+1}, given N observations in $\mathcal{X}$.
- Marginalizations:
  - $p(\mathcal{X}, \theta) = \int dx^{N+1} p(x^{N+1}, \mathcal{X}, \theta) = p(\mathcal{X} \mid \theta) p(\theta)$.
  - $p(\mathcal{X}) = \int d\theta p(\mathcal{X}, \theta) = \int d\theta p(\mathcal{X} \mid \theta) p(\theta)$.
  - $p(x^{N+1}, \mathcal{X}) = \int d\theta p(x^{N+1}, \mathcal{X}, \theta) = \int d\theta p(x^{N+1} \mid \theta) p(\mathcal{X} \mid \theta) p(\theta)$.

- Posterior: $p(\theta \mid \mathcal{X}) = p(\mathcal{X}, \theta)/p(\mathcal{X})$.
- Predictor for new data point: $p(x^{N+1} \mid \mathcal{X}) = p(x^{N+1}, \mathcal{X})/p(\mathcal{X}) = \int d\theta p(x^{N+1} \mid \theta)p(\mathcal{X}, \theta)/p(\mathcal{X}) = \int d\theta p(x^{N+1} \mid \theta)p(\theta \mid \mathcal{X})$.



Joint distribution ($\mathcal{X} = \{x^t\}_{t=1}^N$): $p(x^{N+1}, \mathcal{X}, \theta) = p(x^{N+1} \mid \theta)p(\mathcal{X} \mid \theta)p(\theta)$.

13.2 Estimators

Point Estimators

- The posterior $p(\theta \mid \mathcal{X})$ represents our best knowledge.
- Predictor for new data point: $p(x^{N+1} \mid \mathcal{X}) = \int d\theta p(x^{N+1} \mid \theta)p(\theta \mid \mathcal{X})$.
- The calculation of the integral may be infeasible.

- Estimate θ by $\hat{\theta}$ (or posterior by $p(\theta \mid \mathcal{X}) \approx \delta(\theta - \hat{\theta})$) and use the predictor

$$p(x^{N+1} \mid \mathcal{X}) \approx p(x^{N+1} \mid \hat{\theta}).$$

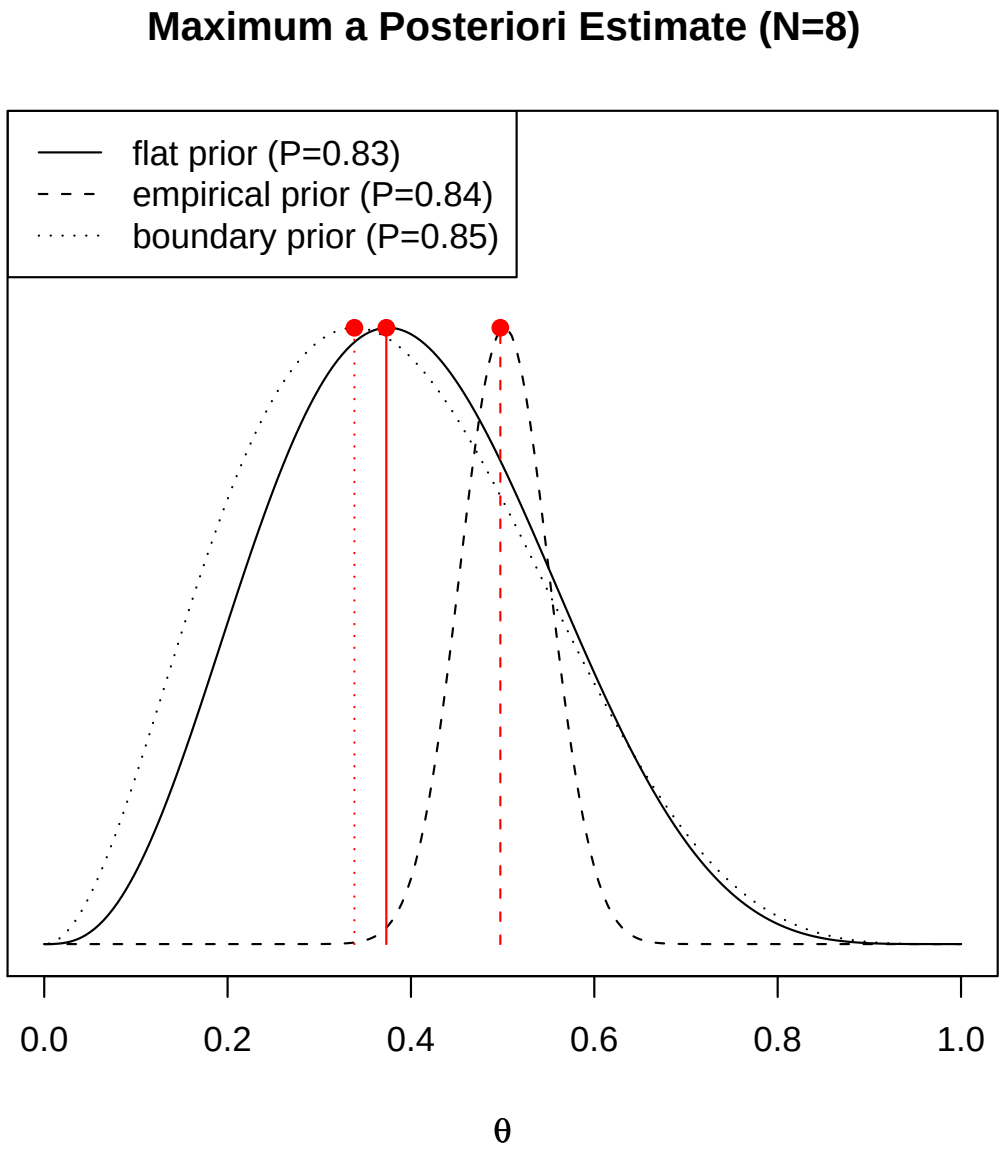
Estimators from the Posterior

Definition 14 (Maximum Likelihood Estimate)**.**

$$\hat{\theta}_{ML} = \arg\max_{\theta} \log p(\mathcal{X} \mid \theta).$$

Definition 15 (Maximum a Posteriori Estimate)**.**

$$\hat{\theta}_{MAP} = \arg\max_{\theta} \log p(\theta \mid \mathcal{X}).$$



Gaussian Density

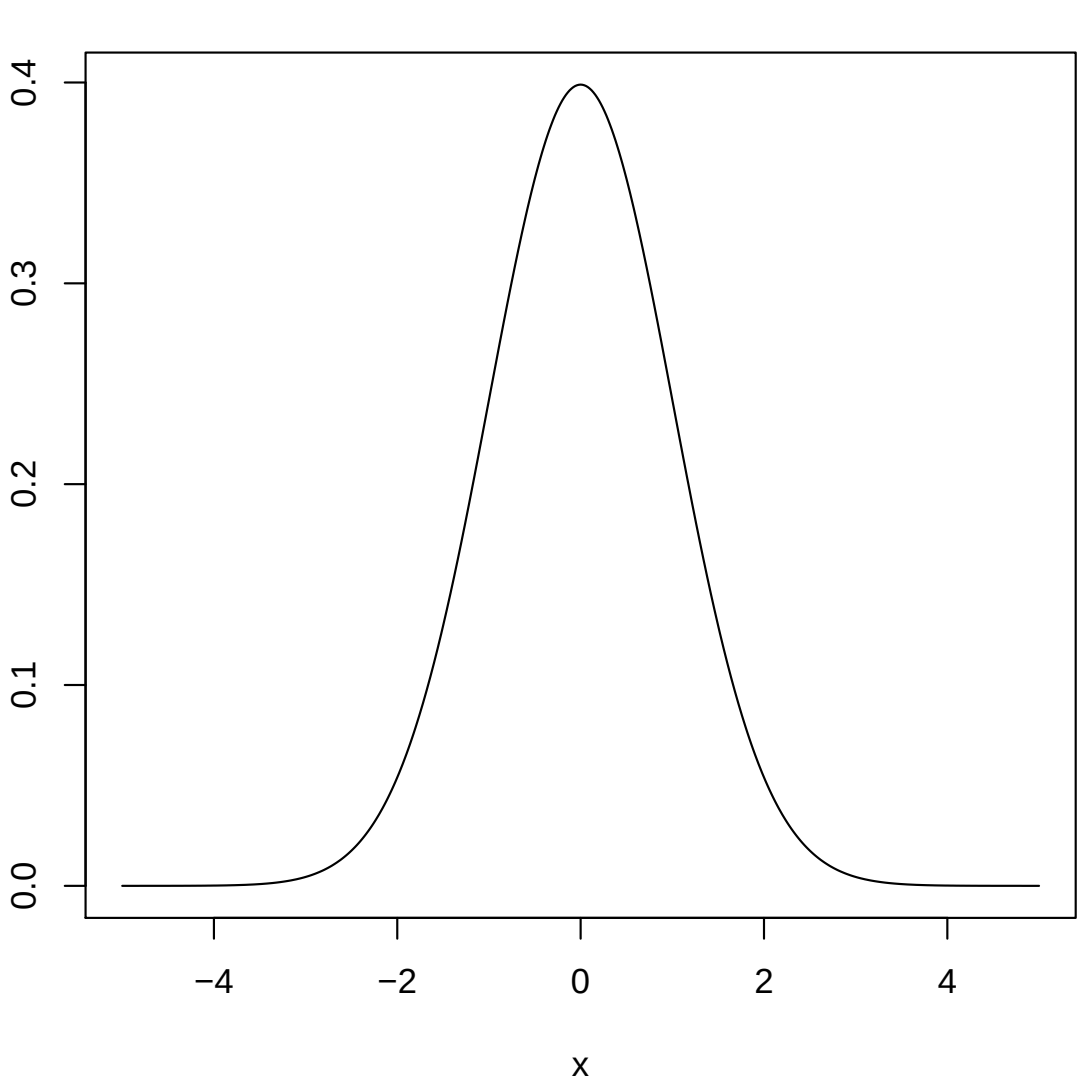
- A real number x is Gaussian (normal) distributed with mean μ and variance σ^2 or $x \sim N(\mu, \sigma^2)$ if its density function is

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

$$\mathcal{L} = \log P(\mathcal{X} \mid \mu, \sigma^2)$$

$$= -\frac{N}{2}\log(2\pi) - N\log\sigma - \frac{\sum_{t=1}^{N}\left(x^t-\mu\right)^2}{2\sigma^2}.$$

$$ML: \begin{cases} m = \frac{1}{N}\sum_{t=1}^{N} x^t \\ s^2 = \frac{1}{N}\sum_{t=1}^{N}\left(x^t-m\right)^2 \end{cases}$$



$p(x \mid \mu = 0, \sigma^2 = 1)$

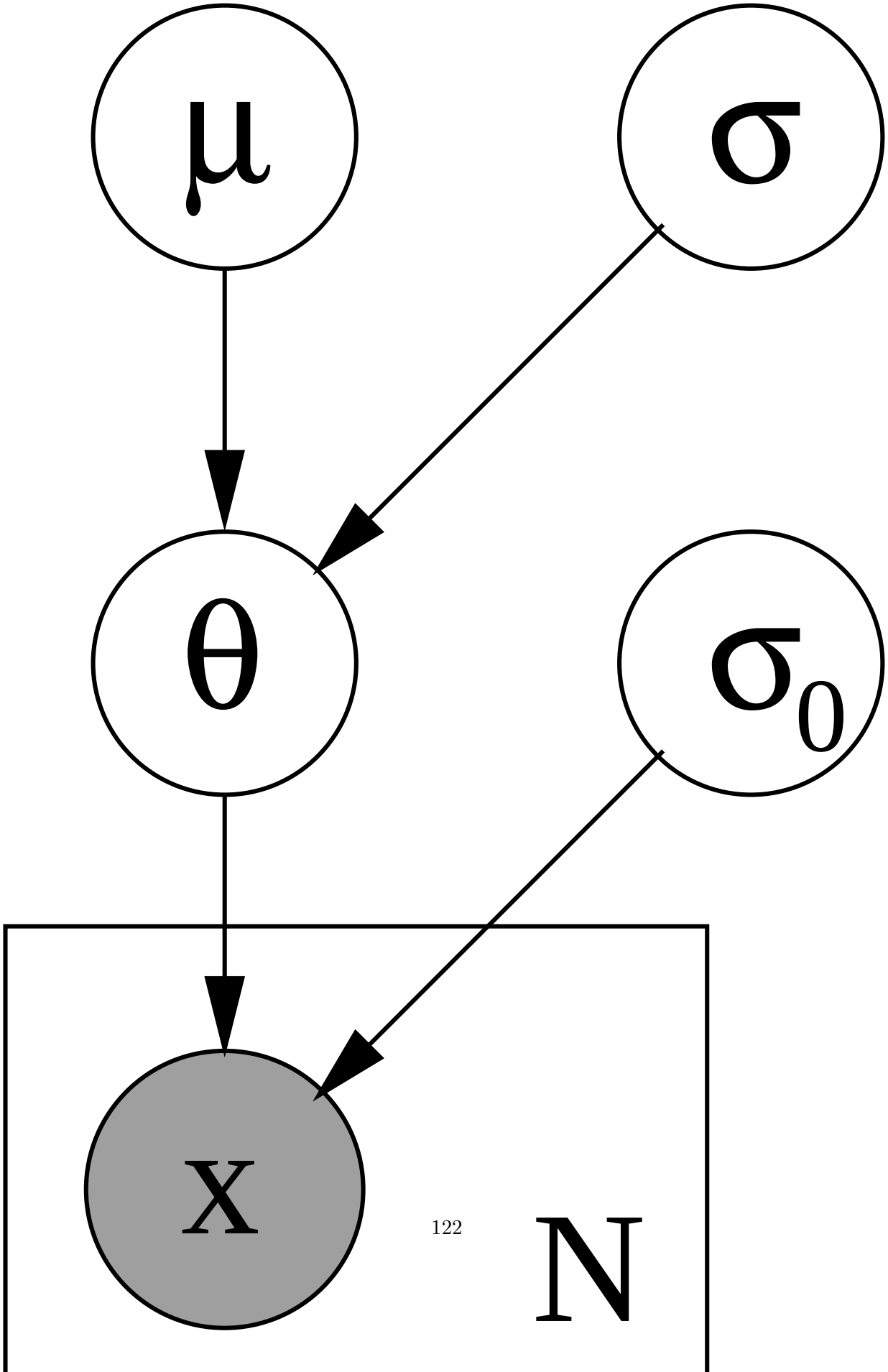
Bayes' Estimator

- *Bayes' estimator*: $\hat{\theta}_{Bayes} = E_{p(\theta|\mathcal{X})}[\theta] = \int d\theta \theta p(\theta \mid \mathcal{X})$.
- Example: $x^t \sim N(\theta, \sigma_0^2)$, $t \in \{1, \ldots, N\}$, and $\theta \sim N(\mu, \sigma^2)$, where μ, σ^2 and σ_0^2 are known constants. Task: estimate θ.

$$\begin{aligned} p(\mathcal{X} \mid \theta) &= \frac{1}{(2\pi\sigma_0^2)^{N/2}} \exp\left(-\frac{\sum_t \left(x^t - \theta\right)^2}{2\sigma_0^2}\right), \\ p(\theta) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\theta - \mu)^2}{2\sigma^2}\right). \end{aligned}$$

- It can be shown that $p(\theta \mid \mathcal{X})$ is Gaussian distributed with

$$\hat{\theta}_{Bayes} = E_{p(\theta|\mathcal{X})}[\theta] = \frac{N/\sigma_0^2}{N/\sigma_0^2 + 1/\sigma^2} m + \frac{1/\sigma^2}{N/\sigma_0^2 + 1/\sigma^2} \mu.$$


μ
σ
θ
σ0
x
N

13.3 Bias and Variance

Bias and Variance

- Setup: unknown parameter θ is estimated by $d(\mathcal{X})$ based on a sample $\mathcal{X}$.
- Example: estimate σ^2 by $d = s^2$.
- *Bias*: $b_\theta(d) = E[d] - \theta$.
- *Variance*: $E\left[(d - E[d])^2\right]$.
- Mean square error of the estimator $r(d, \theta)$:

$$\begin{aligned} r(d,\theta) &= E\left[(d-\theta)^2\right] \\ &= (E[d]-\theta)^2 + E\left[(d-E[d])^2\right] \\ &= \text{Bias}^2 + \text{Variance}. \end{aligned}$$

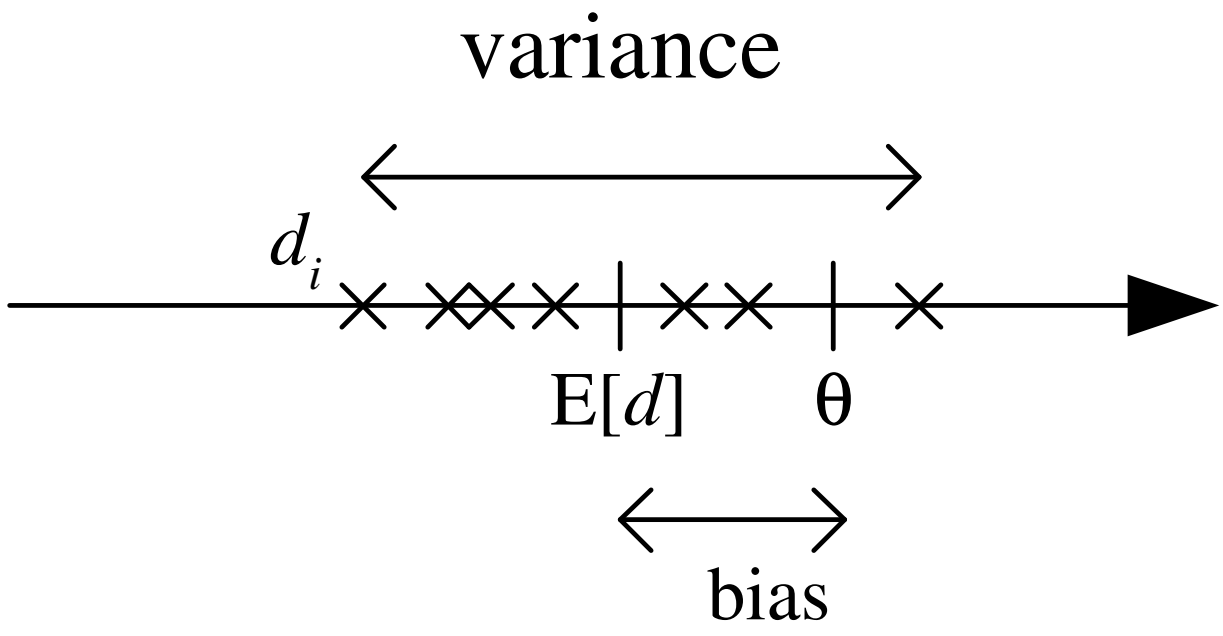


Figure 4.1 of Alpaydin (2004).

Bias and Variance

- Estimator is *unbiased* if $b_\theta(d) = 0$.
- Assume $\mathcal{X}$ is sampled from a Gaussian distribution.
- Estimate σ^2 by s^2: $s^2 = \frac{1}{N}\sum_t \left(x^t - m\right)^2$.

- We obtain:

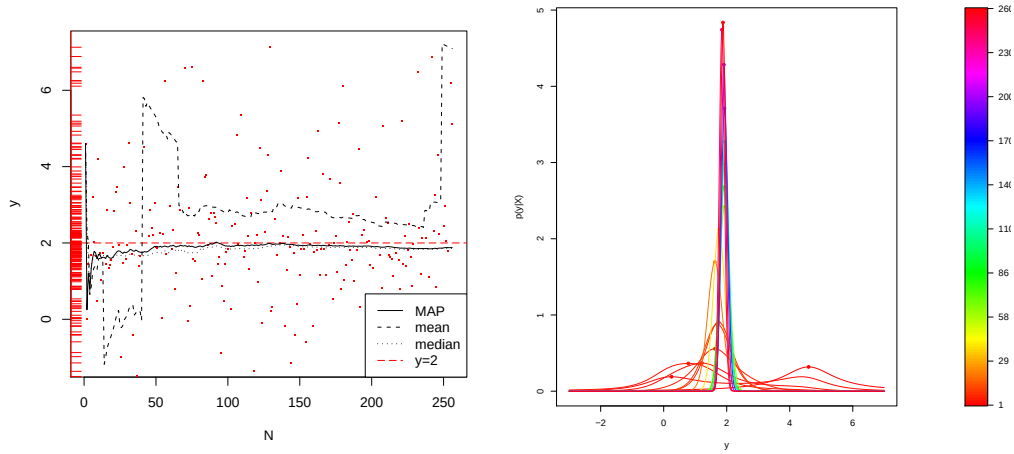
$$E_{p(x|\mu,\sigma^2)}\left[s^2\right] = \frac{N-1}{N}\sigma^2.$$

- s^2 is not unbiased estimator, but $\hat{\sigma}^2 = \frac{N}{N-1}s^2$ is:

$$\hat{\sigma}^2 = \frac{1}{N-1}\sum_{t=1}^{N}\left(x^t - m\right)^2.$$

- s^2 is however *asymptotically unbiased* (that is, bias vanishes when $N \to \infty$).

Example: Lighthouse



See Problem Set 4/2007, problem 3.

About Estimators

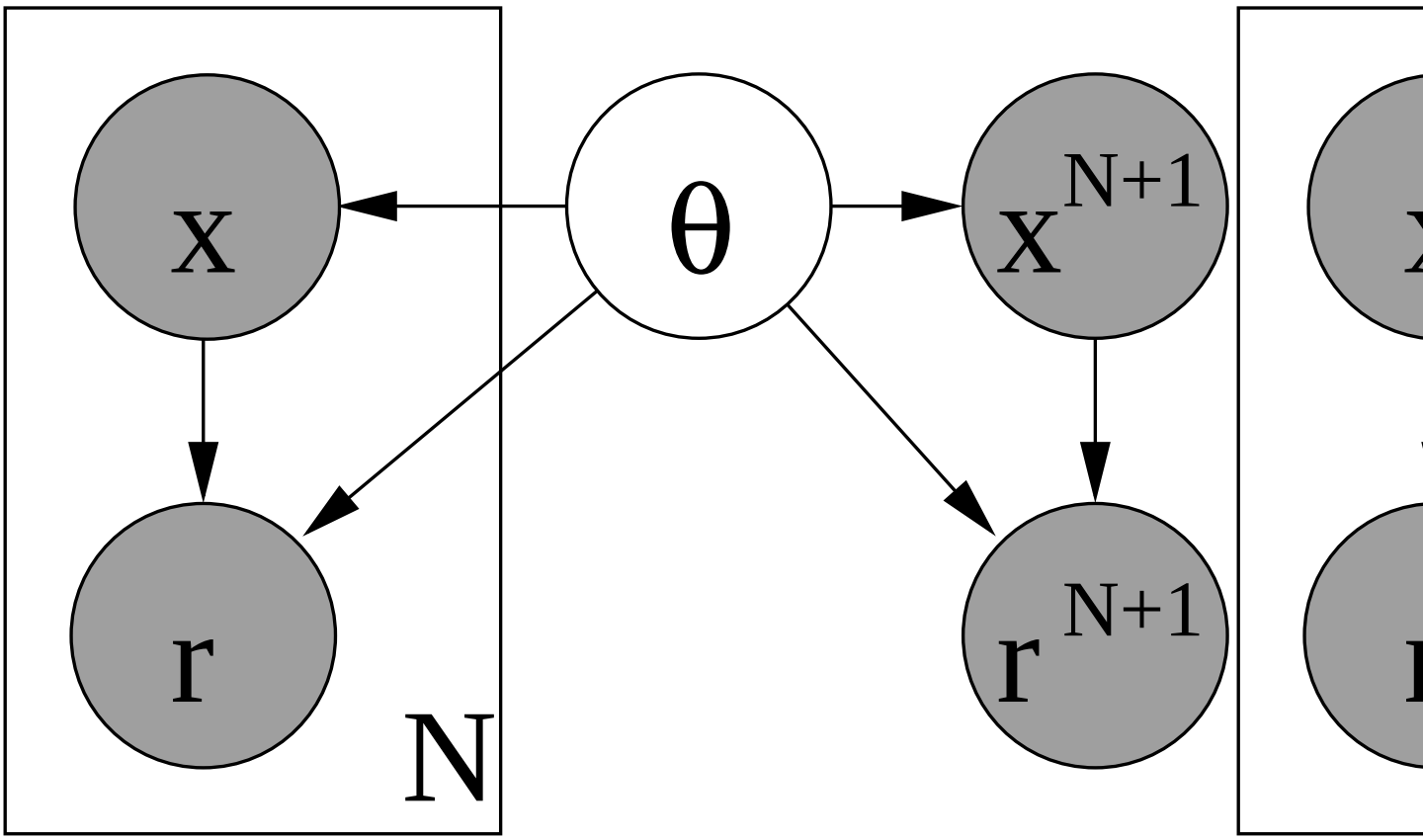
- Point estimates collapse information contained in the posterior distribution into one point.
- Advantages of point estimates:
  - Computations are easier: no need to do the integral.
  - Point estimate may be more interpretable.
  - Point estimates may be good enough. (If the model is approximate anyway it may make no sense to compute the integral exactly.)
- Alternative to point estimates: do the integral analytically or using approximate methods (MCMC, variational methods etc.).
- One should always use test set to validate the results. The best estimate is the one performing best in the validation/test set.

14 Classification and Regression

14.1 Parametric Classification and Regression

Parametric Classification and Regression

- Task: estimation of $p(r \mid x, \mathcal{X})$ (classification or regression), given data $\mathcal{X} = \{(x^t, r^t)\}_{t=1}^N$.
- *Generative modeling (likelihood-based approach)*: Marginalize: $p(r^{N+1} \mid x^{N+1}, \mathcal{X}) = \int d\theta p(r^{N+1} \mid x^{N+1}, \theta) p(\theta$ where $p(\theta \mid \mathcal{X}) \propto p(\theta) \prod_{t=1}^N p(x^t, r^t \mid \theta)$. Example: Bayes Classifier as solved in the following slides. *Discriminative modeling (discriminant-based approach)*: x does not depend on our model θ (x is a covariate, we do not model it): $p(r^{N+1} \mid x^{N+1}, \mathcal{X}) = \int d\theta p(r^{N+1} \mid x^{N+1}, \theta) p_d(\theta \mid \mathcal{X})$, where $p_d(\theta \mid \mathcal{X}) \propto p(\theta) \prod_{t=1}^N p(r^t \mid x^t, \theta)$. Example: Bayesian regression.



14.2 Parametric Classification

Parametric Classification

- *Bayes Classifier*: $p(C_i \mid x) \propto p(x \mid C_i)P(C_i)$.
- Discriminant function: $g_i(x) = \log p(x \mid C_i) + \log P(C_i)$.
- Assume $p(x \mid C_i)$ are Gaussian:

$$p(x \mid C_i; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right).$$

- The discriminant function becomes:

$$g_i(x) = -\frac{1}{2}\log 2\pi - \log \sigma_i - \frac{(x-\mu_i)^2}{2\sigma_i^2} + \log P(C_i).$$

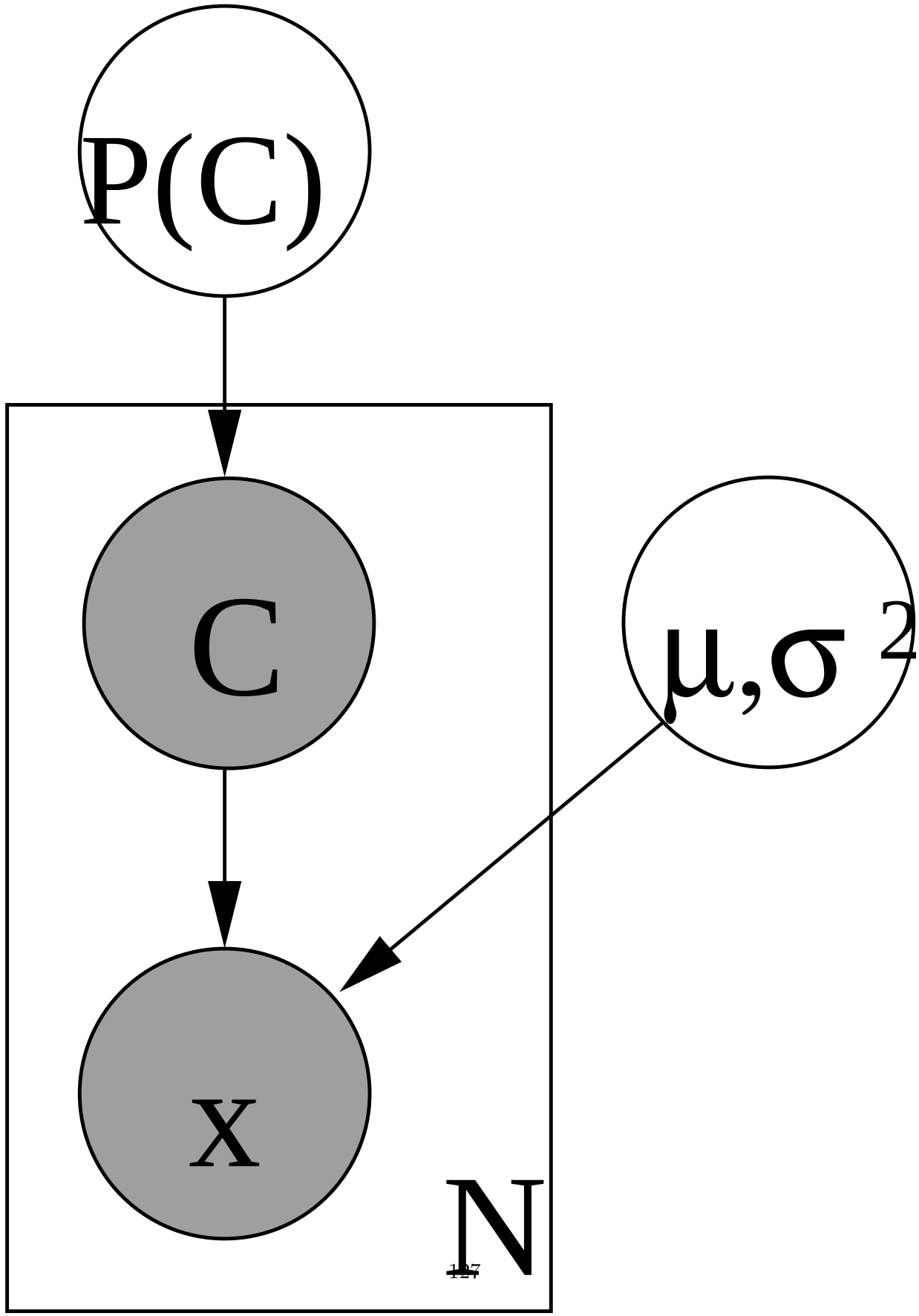
- Sample $\mathcal{X} = \{(x^t, \mathbf{r}^t)\}_{t=1}^N$; $x^t \in \mathbb{R}$, $\mathbf{r}^t \in \{0,1\}^K$. $r_i^t = 1$ if $x^t \in C_i$, $r_i^t = 0$ otherwise.
- Maximum Likelihood (ML) estimates:

$$\hat{P}(C_i) = \frac{\sum_t r_i^t}{N} \quad , \quad m_i = \frac{\sum_t x^t r_i^t}{\sum_t r_i^t},$$

$$s_i^2 = \frac{\sum_t \left(x^t - m_i\right)^2 r_i^t}{\sum_t r_i^t}.$$

- Discriminant becomes:

$$g_i(x) = -\frac{1}{2}\log 2\pi - \log s_i - \frac{(x-m_i)^2}{2s_i^2} + \log \hat{P}(C_i).$$


P(C)
C
μ,σ 2
x
N

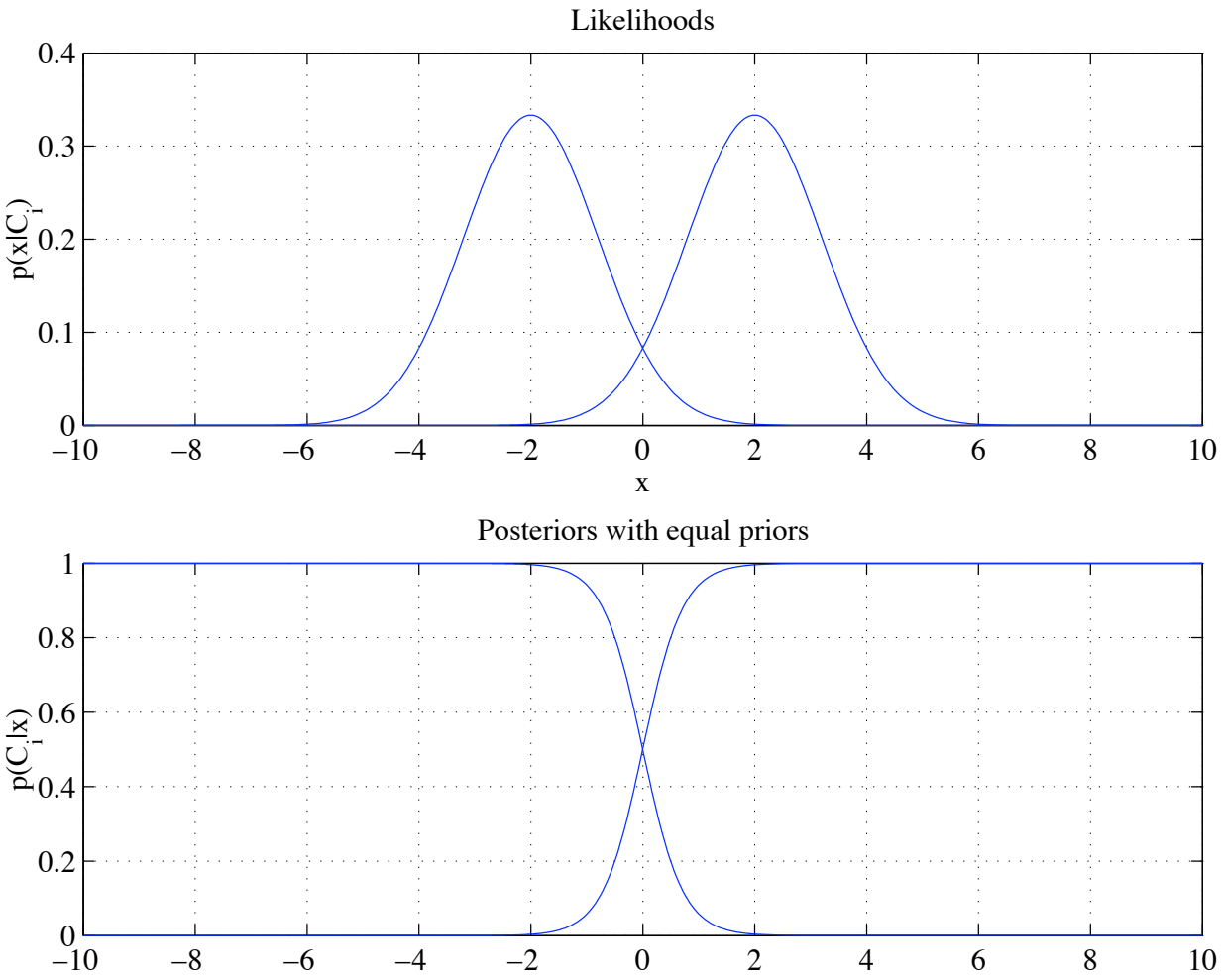


Figure 4.2 of Alpaydin (2004).

$P(C_1) = P(C_2) \quad , \quad \sigma_1^2 = \sigma_2^2.$

Parametric Classification

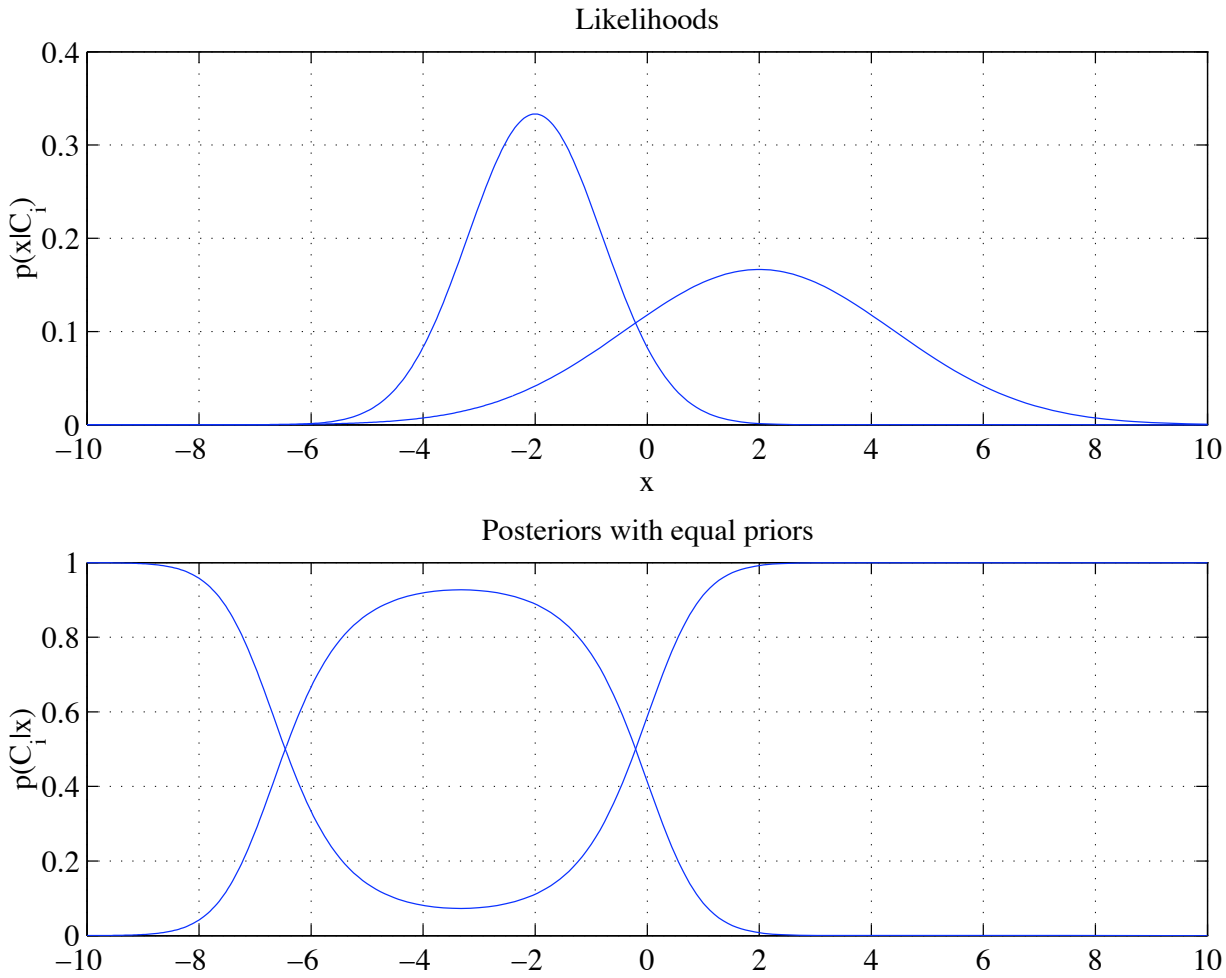


Figure 4.3 of Alpaydin (2004).

$P(C_1) = P(C_2)$, $\sigma_1^2 \neq \sigma_2^2$.

14.3 Parametric Regression

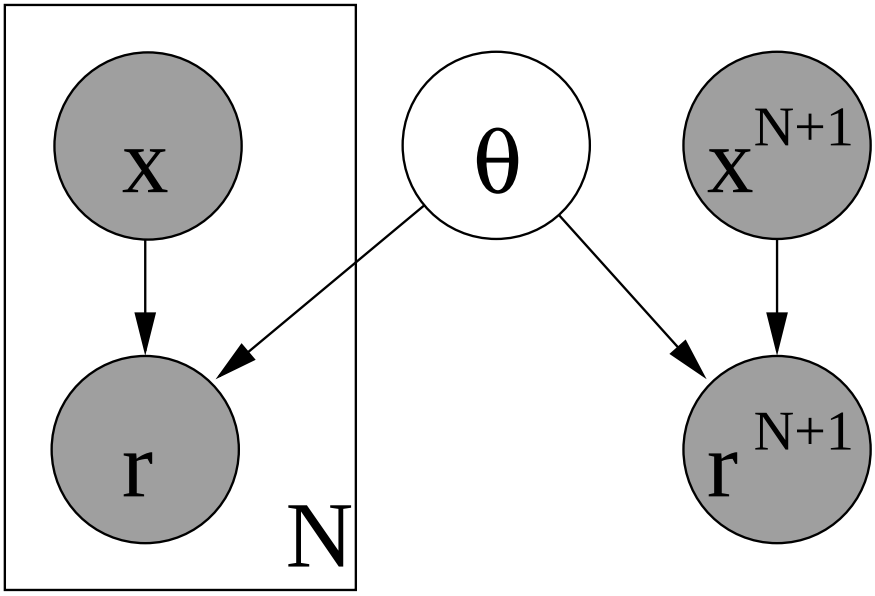
Parametric Regression: Bayesian Regression

- Estimator: $r \approx g(x \mid \theta)$.
- $p(r \mid x, \theta) \sim N(g(x \mid \theta), \sigma^2)$.
- $\mathcal{L}(\theta \mid \mathcal{X}) = \log \prod_{t=1}^N p(x^t, r^t) = \log \prod_{t=1}^N p(r^t \mid x^t) + \log \prod_{t=1}^N p(x^t)$.
- $\mathcal{L}(\theta \mid \mathcal{X}) = \text{const} - N \log \sqrt{2\pi\sigma^2} - \sum_{t=1}^N \left[r^t - g(x^t \mid \theta)\right]^2 / (2\sigma^2)$.
- $E(\theta \mid \mathcal{X}) = \frac{1}{2} \sum_{t=1}^N \left[r^t - g(x^t \mid \theta)\right]^2$.
- Maximizing $\mathcal{L}(\theta \mid \mathcal{X})$ or minimizing $E(\theta \mid \mathcal{X})$ is equivalent to *ML estimate* of θ.

- Example: $g(x \mid w_0, \ldots, w_k) = \sum_{i=0}^{k} w_i x^k$. (*polynomial regression*)
- Square error: $E(\theta \mid \mathcal{X}) = \frac{1}{2} \sum_{t=1}^{N} \left[r^t - g(x^t \mid \theta)\right]^2$.
- Relative square error:

$$E_{RSE} = \frac{\sum_{t=1}^{N} \left[r^t - g(x^t \mid \theta)\right]^2}{\sum_{t=1}^{N} \left[r^t - \overline{r}\right]^2}.$$

- R^2: $R^2 = 1 - E_{RSE}$.



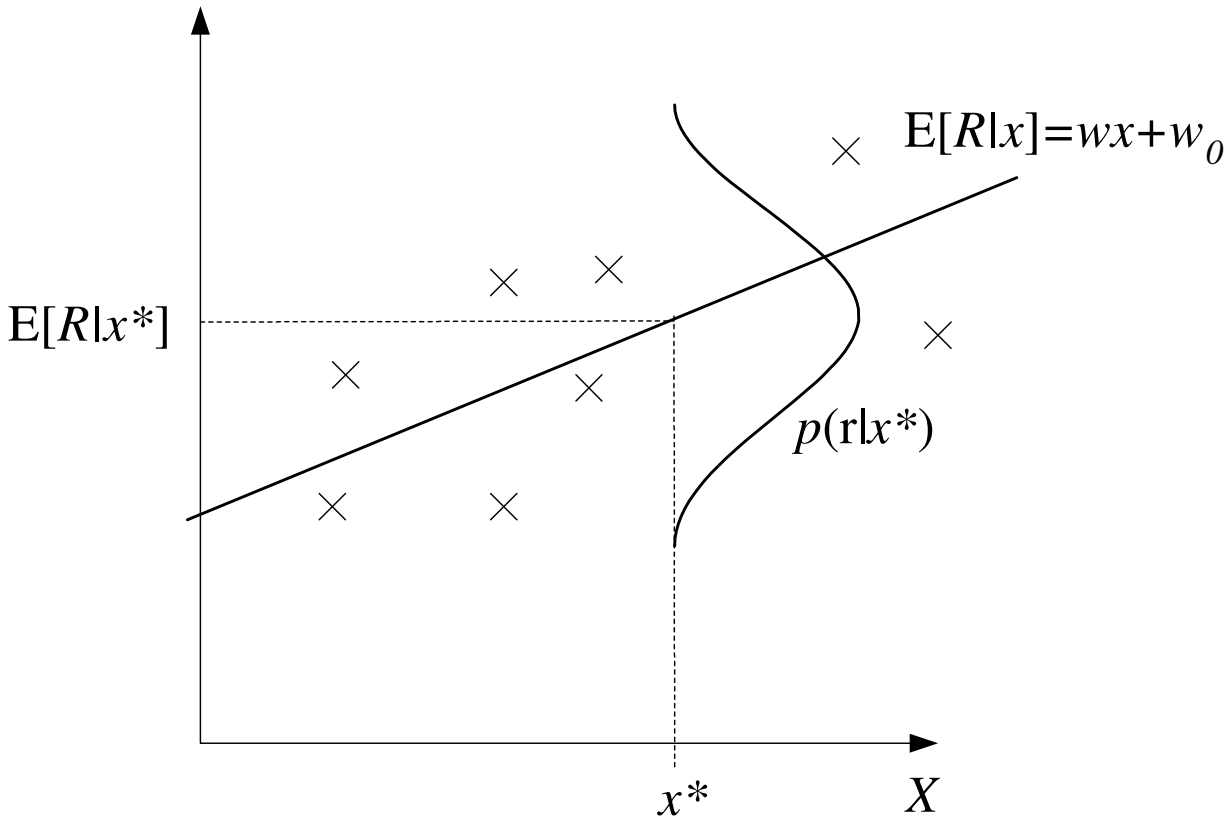


Figure 4.4 of Alpaydin (2004).

15 Model Selection

15.1 Bias/Variance Dilemma

Bias and Variance

$$E\left[(r-g(x))^2 \mid x\right] = E\left[(r-E[r \mid x])^2 \mid x\right] + (E[r \mid x]-g(x))^2$$

noise *squared error*

$$E_X\left[(E[r \mid x]-g(x))^2 \mid x\right] = (E[r \mid x]-E_X[g(x)])^2 + E_X\left[(g(x)-E_X[g(x)])^2\right]$$

bias *variance*

Estimating Bias and Variance

- M samples $\mathcal{X}_i=\{x^t_i, r^t_i\}$, $i=1,...,M$ are used to fit $g_i(x)$, $i=1,...,M$

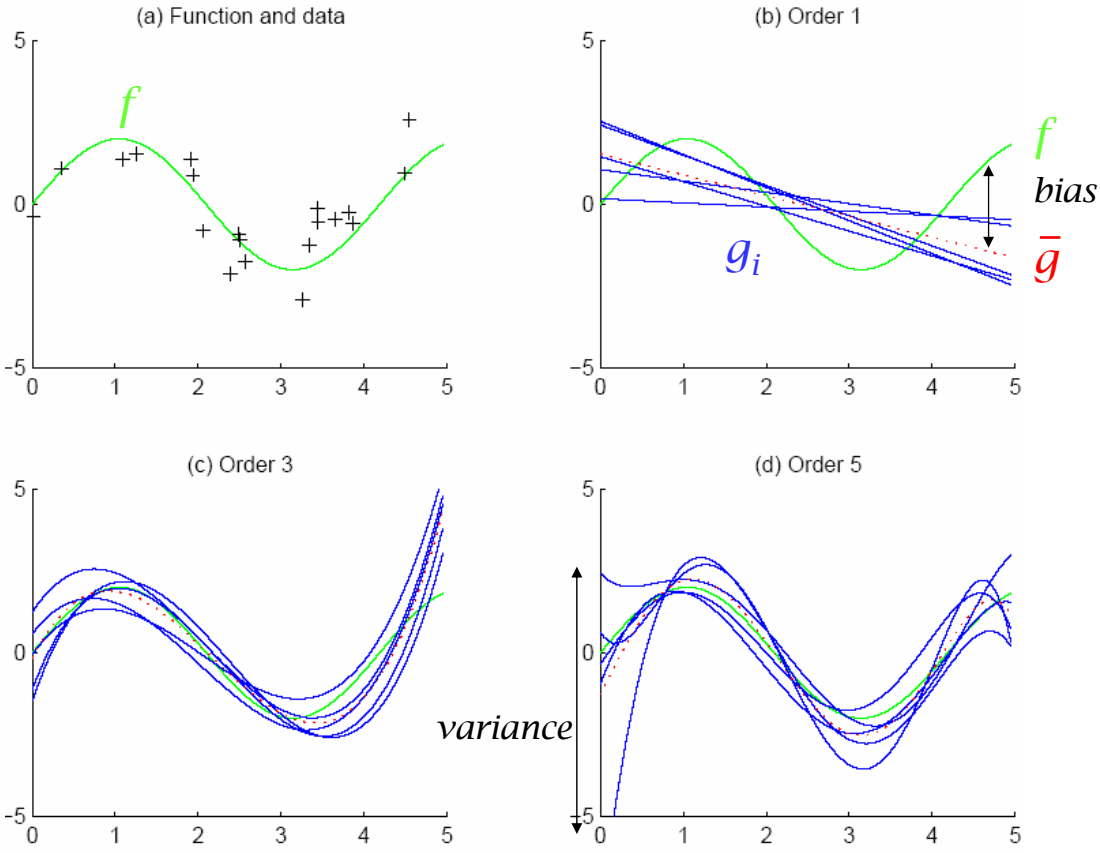
$$\text{Bias}^2(g) = \frac{1}{N}\sum_t \left[\bar{g}(x^t)-f(x^t)\right]^2$$

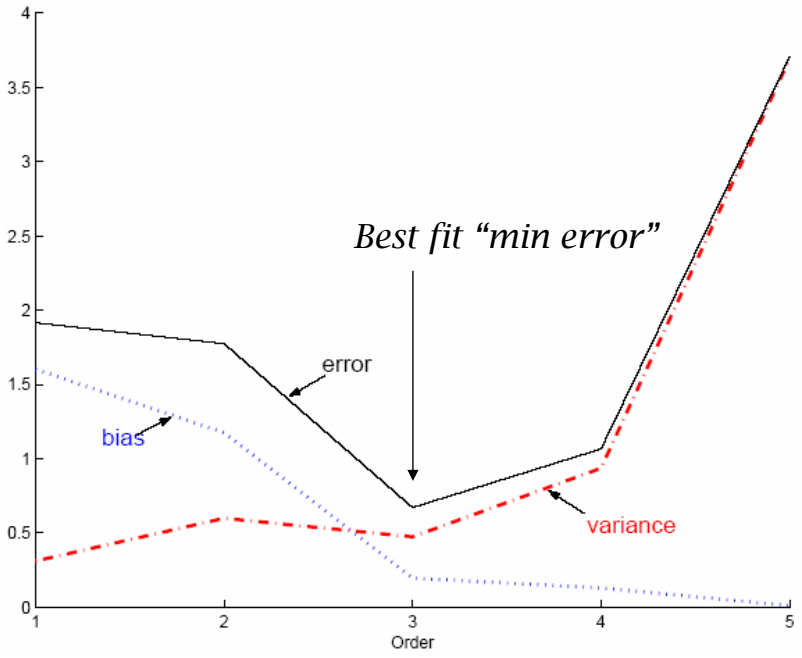
$$\text{Variance}(g) = \frac{1}{NM}\sum_t\sum_i \left[g_i(x^t)-\bar{g}(x^t)\right]^2$$

$$\bar{g}(x) = \frac{1}{M}\sum_t g_i(x)$$

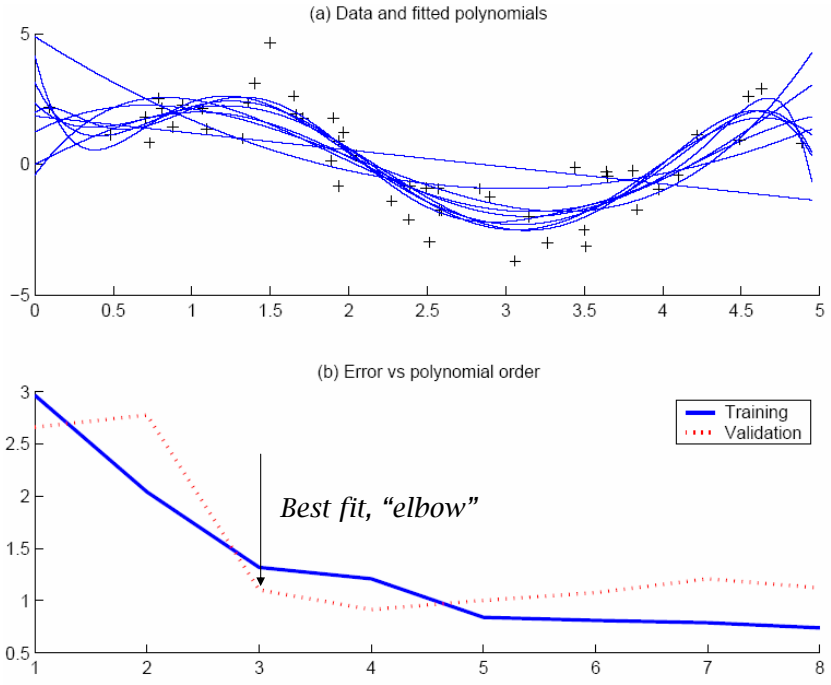
Bias/Variance Dilemma

- Example: $g_i(x) = 2$ has no variance and high bias, $g_i(x) = \sum_t r_i^t/N$ has lower bias with variance.
- *Bias/Variance dilemma:* as we increase complexity,
  - bias decreases (a better fit to data) and
  - variance increases (fit varies more with data).





Polynomial Regression



15.2 Model Selection Procedures

- Cross-validation: most robust if there is enough data.
- Structural risk minimization (SRM): used, for example, in support vector machines (SVM).
- Bayesian model selection: use prior and Bayes' formula.
- Minimum description length (MDL): can be viewed as MAP estimate.
- Regularization: add penalty term for complex models (can be obtained, for example, from prior).
- Latter four methods do not strictly require validation set (at least if implicit modeling assumptions are satisfied, such as that in Bayesian model selection the data is from the model family; it is always a good idea to use a test set) and latter three are related.
- There is no single best way for small amounts of data (your prior assumptions matter).

Cross-validation

- Separate data into training and validation sets.
- Learn using training set.
- Use error on validation set to select a model.
- You need a test set also if you want an unbiased estimate of error on new data.
- Question: what is a sufficient size for the validation set?

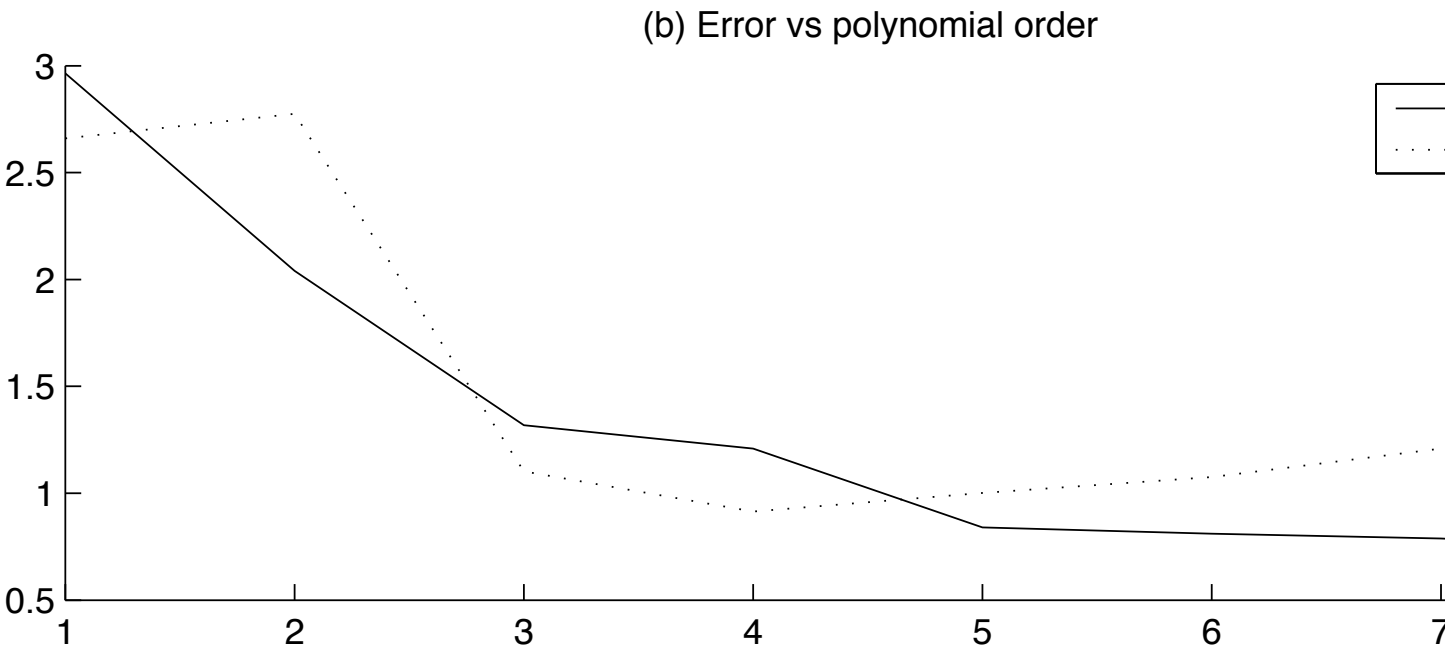


Figure 4.7 of Alpaydin (2004).

Structural Risk Minimization (SRM)

- According to the PAC theory, with probability $1 - \delta$,

$$E_{TEST} \leq E_{TRAIN} + \sqrt{\frac{\mathcal{VC}(H)\left(\log \frac{2N}{\mathcal{VC}(H)} + 1\right) - \log \frac{\delta}{4}}{N}},$$

  where N is the size of the training data, $\mathcal{VC}(H)$ is the VC-dimension of the hypothesis class and E_{TEST} is the expected error on new data and E_{TRAIN} is the error on the training set, respectively.

- SRM: Choose hypothesis class (for example, the degree of a polynomial) such that the bound on E_{TEST} is minimized.

- Often used to train the Support Vector Machines (SVM).

- (Vapnik (1995) contains more discussion of the SRM inductive principle; it won't be discussed in this course in more detail.)

Bayesian Model Selection

- Define prior probability over models, $p(\text{model})$.

$$p(\text{model} \mid \text{data}) = \frac{p(\text{data} \mid \text{model})p(\text{model})}{p(\text{data})}$$

- Equivalent to regularization, when prior favors simpler models.

- MAP: choose model which maximizes

$$\mathcal{L} = \log p(\text{data} \mid \text{model}) + \log p(\text{model})$$

Regularization

- Augment the cost by a term which penalizes more complex models: $E(\theta \mid \mathcal{X}) \rightarrow E'(\theta \mid \mathcal{X}) = E(\theta \mid \mathcal{X}) + \lambda \times \text{complexity}$.

- Example: in Bayesian linear regression, define a Gaussian prior for the model parameters w_0, w_1: $p(w_0) \sim N(0, 1/\lambda)$, $p(w_1) \sim N(0, 1/\lambda)$. The old ML function reads (if the error has an unit variance)

$$\mathcal{L}_{ML}(\theta \mid \mathcal{X}) = -\frac{1}{2}\sum_{t=1}^{N}\left[r^t - g(x^t \mid \theta)\right]^2 + \ldots$$

  The MAP estimate gives an additional term

$$\mathcal{L}_{MAP}(\theta \mid \mathcal{X}) = \mathcal{L}_{ML}(\theta \mid \mathcal{X}) - \frac{1}{2}\lambda\left(w_0^2 + w_1^2\right).$$

  This is an example of regularization (the prior favours models with small w_0, w_1).

Minimum Description Length (MDL)

- Information theory: the optimal (shortest expected coding length) code for an event with probability p is $-\log_2 p$ bits.
- MAP estimate finds a model that minimizes

$$-\mathcal{L} = -\log_2 p(\text{data} \mid \text{model}) - \log_2 p(\text{model})$$

- $-\log_2 p(\text{model})$: number of bits it takes to describe the model.
- $-\log_2 p(\text{data} \mid \text{model})$: number of bits it takes to describe the data, if the model is known.
- $-\mathcal{L}$: the *description length* of the data.
- MAP estimate can be seen as finding a shortest description of the data (that is, the best compression of the data).

15.3 Conclusion

Conclusion

- Next lecture: Alpaydin (2004) Ch 5.

16 Model Selection

16.1 Summary

- *Cross-validation*: most robust if there is enough data.
- Related:
  - *Bayesian model selection*: use prior and Bayes' formula.
  - *Regularization*: add penalty term for complex models (can be obtained, for example, from prior).
  - *Minimum description length (MDL)*: can be viewed as MAP estimate. [Basic idea good to know, details not required in this course.]
- *Structural risk minimization (SRM)*: used, for example, in support vector machines (SVM). [Not required to know in this course.]
- The latter do not strictly require a validation set.
- There is no single best way for small amounts of data (your prior assumptions matter).

16.2 Cross-validation

Cross-validation

- Separate data into training and validation sets.
- Learn using training set.
- Use error on validation set to select a model.
- You need a test set also if you want an unbiased estimate of error on new data.
- Question: what is a sufficient size for the validation set?

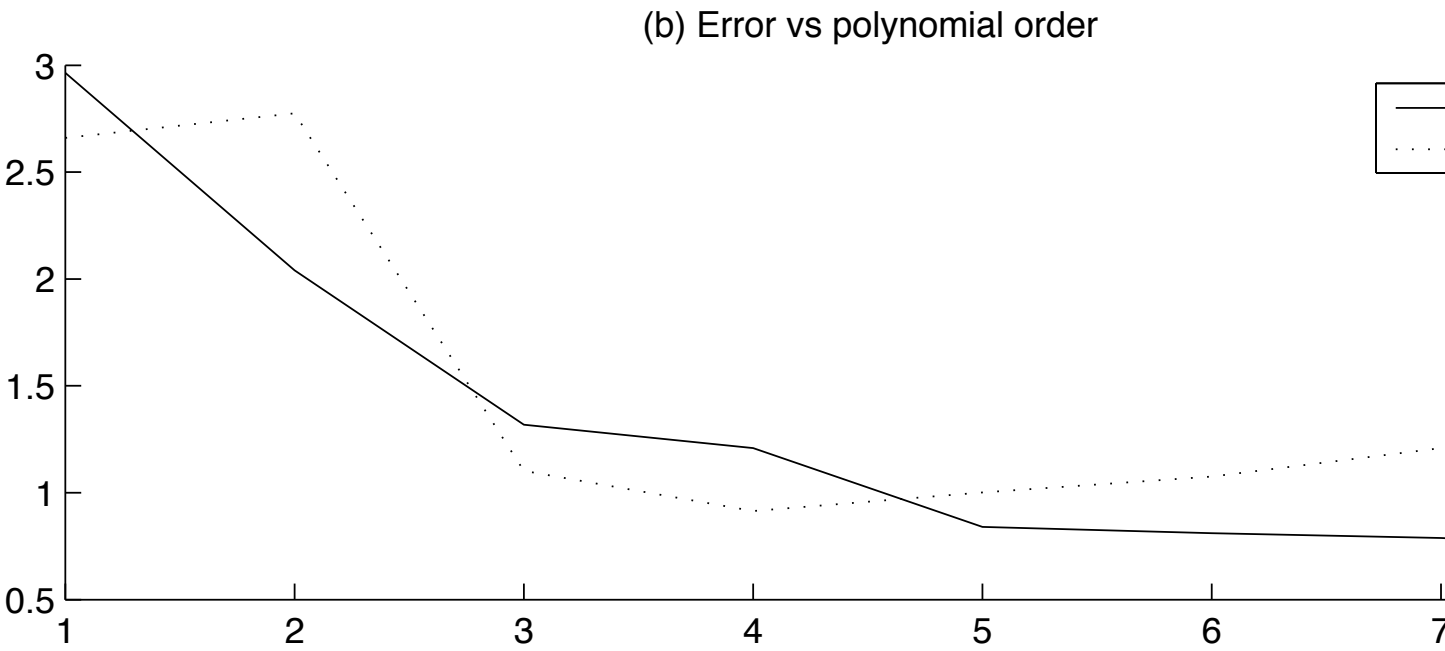


Figure 4.7 of Alpaydin (2004).

Cross-validation

- Assumption: training data $\mathcal{X} = \{(r^t, x^t)\}_{t=1}^N$ has been sampled iid from some (usually unknown) distribution F, $(r^t, x^t) \sim F$.
- In cross-validation, training data is split in random in *training set* of size $N-n$ and *validation set* of size n. Effectively then also the validation set is sampled iid from F.
- Classifier $h(x)$ is trained using the training set.
- *Generalization error* $\mathcal{E}$: probability of misclassification for a new data point $(r, x) \sim F$, $\mathcal{E} = E_F[I(r \neq h(x))]$.
- Fraction of misclassified items in the validation set, E_{VALID}, can be used as an estimate of the generalization error $\mathcal{E}$.

- E_{VALID} is an unbiased estimator of $\mathcal{E}$.
- The variance of the estimator E_{VALID} is $\mathrm{Var}(E_{VALID}) = \sqrt{\mathcal{E}(1-\mathcal{E})/n} \leq 1/(2\sqrt{n})$.

Cross-validation

- Classifier $h(x)$ is trained using the training set.
- Fraction of misclassified items in the validation set, E_{VALID}, can be used as an estimate of the generalization error $\mathcal{E}$.
- If we select model that has the smallest E_{VALID} it is no longer unbiased estimate of the generalization error.
- To get an unbiased estimate of the generalization error we must split the data into three parts (training, validation and test sets).

16.3 Bayesian Model Selection

Bayesian Model Selection

- Define prior probability over models, $p(\text{model})$.

$$p(\text{model} \mid \text{data}) = \frac{p(\text{data} \mid \text{model})p(\text{model})}{p(\text{data})}$$

- Equivalent to regularization, when prior favors simpler models.
- MAP: choose model which maximizes

$$\mathcal{L} = \log p(\text{data} \mid \text{model}) + \log p(\text{model})$$

- (Notice: we again take logs of probabilities for computational convenience; log of posterior has the same maximum as the original posterior. Evidence $p(\text{data})$ is constant with respect to the model, we can therefore drop it.)

Regularization

- Augment the cost by a term which penalizes more complex models: $E(\theta \mid \mathcal{X}) \to E'(\theta \mid \mathcal{X}) = E(\theta \mid \mathcal{X}) + \lambda \times \text{complexity}$.
- Example 1, *Bayesian linear regression*: define a Gaussian prior for the model parameters $\theta = (w_0, w_1)$: $p(w_0) \sim N(0, 1/\lambda)$, $p(w_1) \sim N(0, 1/\lambda)$. The old ML function reads (if the error has an unit variance)

$$\mathcal{L}_{ML}(\theta \mid \mathcal{X}) = -\frac{1}{2}\sum_{t=1}^{N}\left[r^t - w_0 - w_1 x^t\right]^2 + \ldots$$

  The MAP estimate gives an additional term

$$\mathcal{L}_{MAP}(\theta \mid \mathcal{X}) = \mathcal{L}_{ML}(\theta \mid \mathcal{X}) - \frac{1}{2}\lambda\left(w_0^2 + w_1^2\right).$$

  This is an example of regularization (the prior favours models with small w_0, w_1).

- Example 2, *Akaike Information Criterion (AIC)*: Penalize for more parameters and choose model that maximizes:

$$\mathcal{L}(\theta \mid \mathcal{X}) = \mathcal{L}_{ML}(\theta \mid \mathcal{X}) - M,$$

  where M is the number of adjustable parameters in the model.

- Example 3, *Bayesian Information Criterion (BIC)*: Penalize for more parameters and choose model which maximizes:

$$\mathcal{L}(\theta \mid \mathcal{X}) = \mathcal{L}_{ML}(\theta \mid \mathcal{X}) - \frac{1}{2} M \log N,$$

  where M is the number of adjustable parameters in the model and N is the size of the sample $\mathcal{X}$.

- AIC and BIC have some theoretical justification, however, they are very approximate. They are useful because of their simplicity. They tend to favour (too) simple models.

- Weird intro: `http://www.cs.cmu.edu/~zhuxj/courseproject/aicbic/`

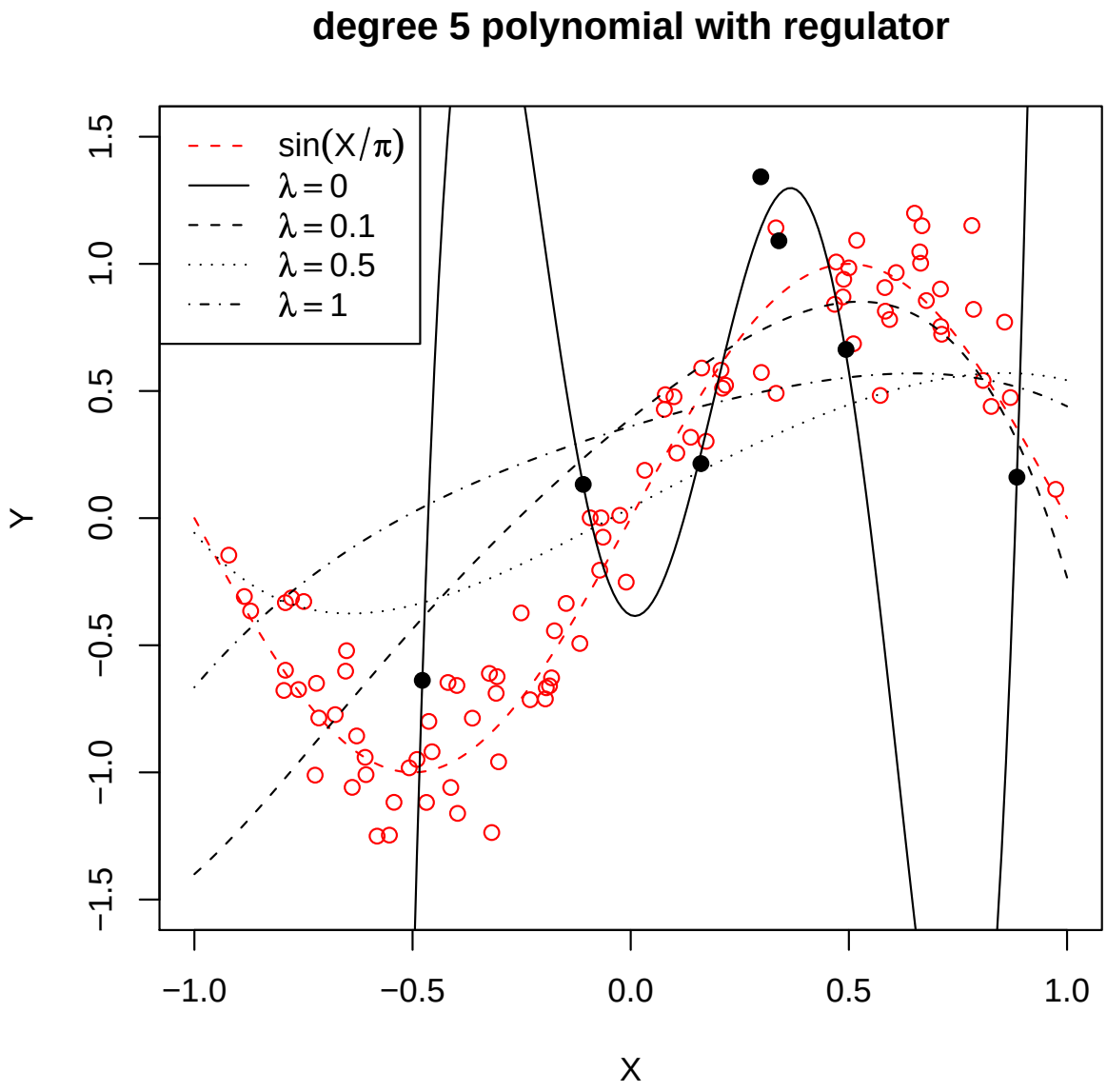
Regression Using Regularization

- Do Bayesian regression with $\sigma^2 = 1$ with the similar data as in the 2nd lecture, use MAP solution with Gaussian prior over parameters.

$$-\mathcal{L}_{MAP} =$$

$$\frac{1}{2} \sum_{t=1}^{7} \left[y^t - g(x^t \mid \overline{w}) \right]^2 + \frac{1}{2} \lambda \overline{w}^T \overline{w}.$$

$$g(x \mid \overline{w}) = \sum_{i=0}^{5} w_i x^i.$$



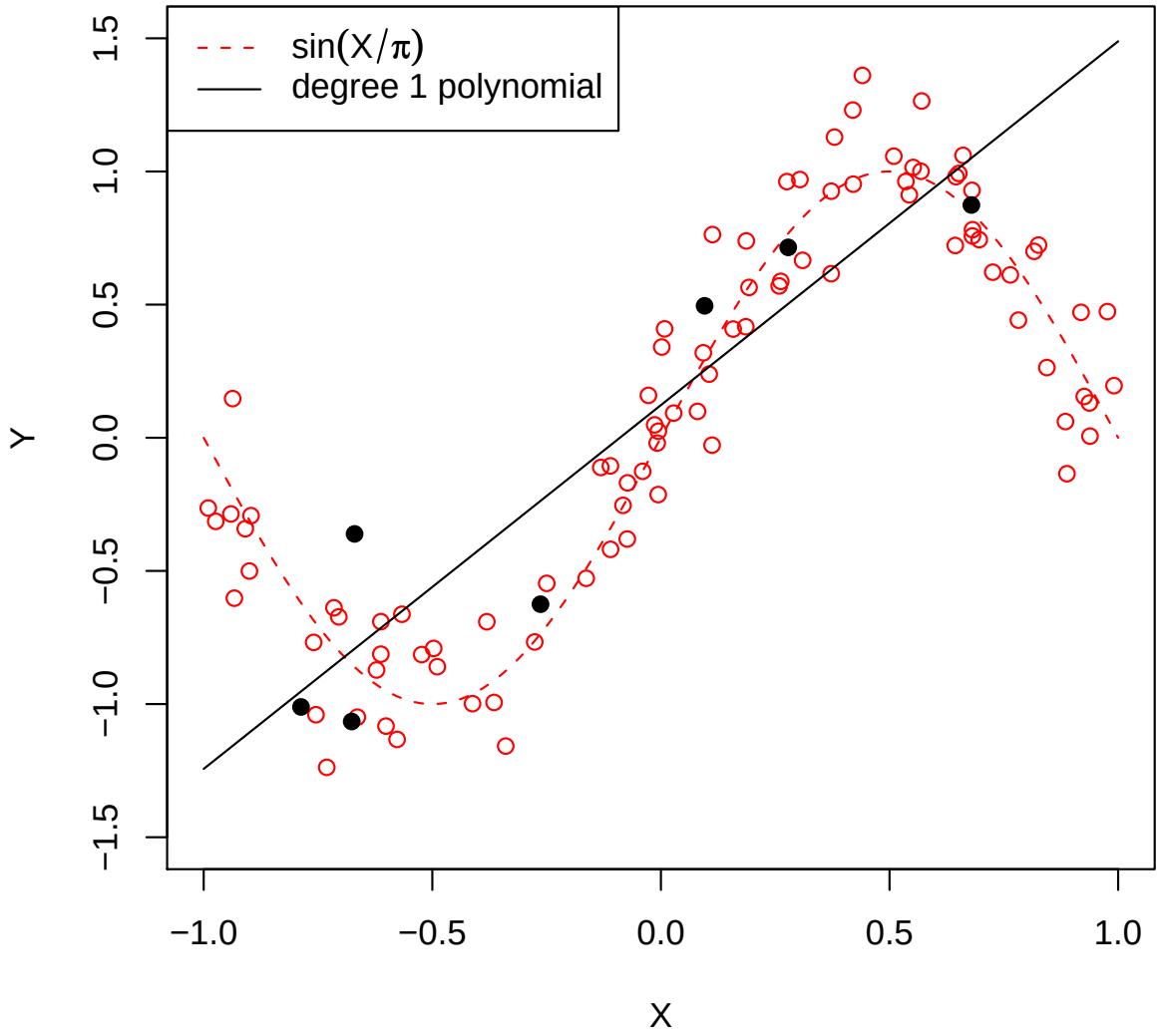
Regression Using Regularization

Do Bayesian regression with $\sigma^2 = 1$ with the same data as in the 2nd lecture, use ML solutions and AIC and BIC regularization:

k	E_{TRAIN}	E_{TEST}	$-\mathcal{L}_{AIC}$	$-\mathcal{L}_{BIC}$
0	0.580	0.541	3.03	3.00
1	0.077	0.294	**2.26**	**2.21**
2	0.076	0.275	3.26	3.18
3	0.057	**0.057**	4.19	4.09
4	0.046	0.562	5.16	5.02
5	0.035	4.637	6.12	5.96
6	0	10^6	7.00	6.81

$N = 7$, $M = k + 1$, $-\mathcal{L}_{AIC} = \frac{N}{2} E_{TRAIN} + M$,

$-\mathcal{L}_{BIC} = \frac{N}{2} E_{TRAIN} + \frac{1}{2} M \log N$, $g(x \mid w_0, \ldots, w_k) = \sum_{i=0}^{k} w_i x^i$, $E_{TRAIN} = -\frac{2}{N}\mathcal{L}_{ML} = \frac{1}{N}\sum_{t=1}^{N} \left[r^t - g(x^t \mid w)\right]^2$.



Minimum Description Length (MDL)

- *Minimum Description Length (MDL)*: a good model is such that it can be used to give the data the shortest description.
- *Kolmogorov complexity*: shortest description of the data.
- Idea:

  - Model can be described using $L(M)$ bits.
  - Data can be described using $L(D \mid M)$ bits, when the model is known.
  - Total description length $L = L(M) + L(D \mid M)$ (approx. Kolmogorov complexity).
  - *Occam's razor*: prefer the shortest description/hypothesis, choose model with smallest L.
- The data could in principle be compressed to L bits.
- (In model selection we do not usually need explicit compression, just the description lengths.)

Minimum Description Length (MDL)

- MAP estimate finds a model that minimizes

$$-\mathcal{L} = -\log_2 p(\text{data} \mid \text{model}) - \log_2 p(\text{model})$$

- $-\log_2 p(\text{model})$: number of bits it takes to describe the model.
- $-\log_2 p(\text{data} \mid \text{model})$: number of bits it takes to describe the data, if the model is known.
- $-\mathcal{L}$: the *description length* of the data.
- MAP estimate can be seen as finding a shortest description of the data (that is, the best compression of the data).

Minimum Description Length (MDL)

- Information theory: the optimal (shortest expected coding length) code for an event with probability p is $-\log_2 p$ bits.
- Example (Huffman coding; in model selection we do not usually need to construct the coding):
  - Let the probabilities of four letters be $P(A) = \frac{1}{2}$, $P(B) = \frac{1}{4}$, $P(C) = \frac{1}{8}$, $P(D) = \frac{1}{8}$.
  - Optimal coding: $A \to 0$, $B \to 10$, $C \to 110$, $D \to 111$.
  - For example, $ADAB$ would be coded as 0111010 (7 bits).
  - Expected coding length $L = \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{8} \times 3 = 1.75$ bits per number. "Compression ratio" $1.75/2 = 0.875$ as compared to the naive coding of each letter with 2 bits (e.g., $A = 00$, $B = 01$, $C = 10$, $D = 11$).

Minimum Description Length (MDL)

- An integer in $\{0, \ldots, n\}$ can be expressed using $\log_2(n+1)$ bits.
- Example: To express an integer in $\{0, \ldots, 15\}$ using binary numbers you need $\log_2 16 = 4$ bits.
- Usually we do not need to find explicit coding in model selection, knowing the coding length is enough.

Minimum Description Length (MDL)

- Data: an ordered sequence D of N binary numbers.
- Model 1: Code the sequence as such.
  - Coding length of the model $L(M_1) = 0$ bits.
  - Coding length of the data $L(D \mid M_1) = N$ bits.
  - Total coding length $L_1 = L(M_1) + L(D \mid M_1) = N$ bits.
- Model 2: Use the frequency of ones for better coding.
  - The model is the number of ones n_1 which is a integer in $[0, N]$. It can be expressed using $L(M_2) = \log_2(N+1)$ bits.
  - There are $\left(\begin{array}{c} N \\ n_1 \end{array} \right)$ possible binary sequences of length N having n_1 ones. A sequence can be expressed using $L(D \mid M_2) = \log_2 \left(\begin{array}{c} N \\ n_1 \end{array} \right)$ bits when n_1 is known.
  - Total coding length

$$L_2 = L(M_2) + L(D \mid M_2) = \log_2(N+1) + \log_2 \left(\begin{array}{c} N \\ n_1 \end{array} \right) \text{ bits.}$$

Minimum Description Length (MDL)

- Example 1: $D = 0111010010$, $N = 10$.
  - $L_1 = 10$ bits. (*Choose 1.*)
  - $L_2 = \log_2(10+1) + \log_2 \left(\begin{array}{c} 10 \\ 5 \end{array} \right) = 3.4 + 8.0 = 11.4$ bits.
- Example 2: $D = 0001000010$, $N = 10$.
  - $L_1 = 10$ bits.
  - $L_2 = \log_2(10+1) + \log_2 \left(\begin{array}{c} 10 \\ 2 \end{array} \right) = 3.4 + 5.5 = 8.9$ bits. (*Choose 2.*)
- Example 3: $D = 0000000000$, $N = 10$.
  - $L_1 = 10$ bits.
  - $L_2 = \log_2(10+1) + \log_2 \left(\begin{array}{c} 10 \\ 0 \end{array} \right) = 3.4 + 0 = 3.4$ bits. (*Choose 2.*)

Structural Risk Minimization (SRM)

- According to the PAC theory, with probability $1 - \delta$,

$$E_{TEST} \leq E_{TRAIN} + \sqrt{\frac{\mathcal{VC}(H)\left(\log \frac{2N}{\mathcal{VC}(H)} + 1\right) - \log \frac{\delta}{4}}{N}},$$

  where N is the size of the training data, $\mathcal{VC}(H)$ is the VC-dimension of the hypothesis class and E_{TEST} is the expected error on new data and E_{TRAIN} is the error on the training set, respectively.

- SRM: Choose hypothesis class (for example, the degree of a polynomial) such that the bound on E_{TEST} is minimized.

- Often used to train the Support Vector Machines (SVM).

- (Vapnik (1995) contains more discussion of the SRM inductive principle; it won't be discussed in this course in more detail.)

17 Multivariate Methods

Remainder of the lecture on the blackboard.

For slides see Alpaydin's site: `http://www.cmpe.boun.edu.tr/~ethem/i2ml/slides/v1-1/i2ml-chap5-v1-1.pdf`

18 Multivariate Methods

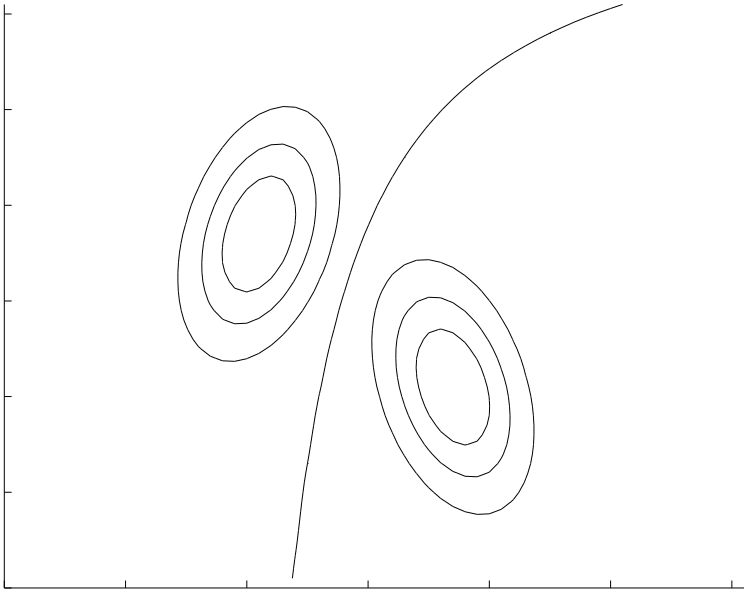
18.1 Bayes Classifier

Bayes Classifier

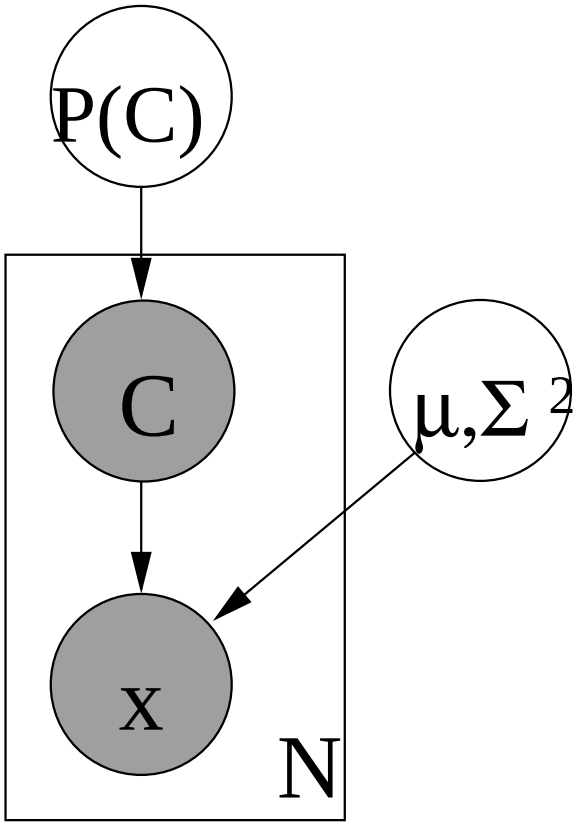
- Data are real vectors.

- Idea: vectors are from class-specific multivariate normal distributions.

- Full model: covariance matrix has $O(Kd^2)$ parameters.



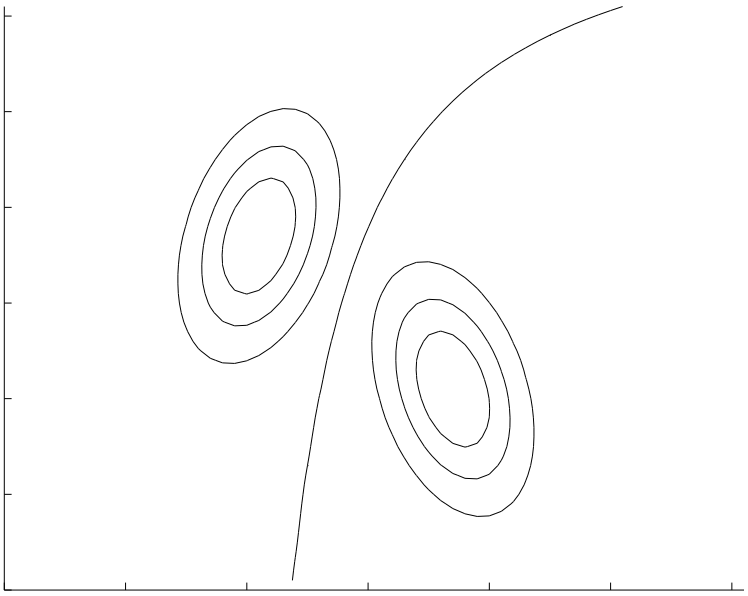
From Figure 5.3 of Alpaydin (2004).



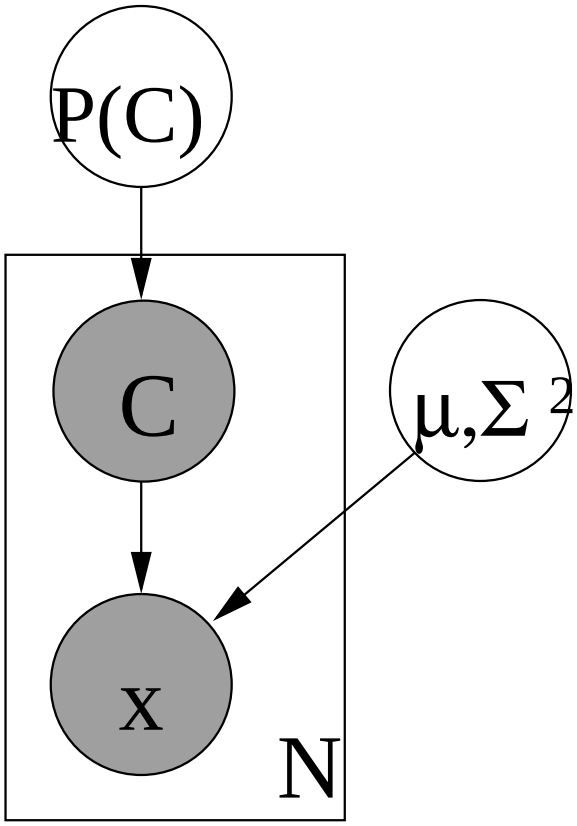
Bayes Classifier

- Data are real vectors.

- Idea: vectors are from class-specific multivariate normal distributions.
- Full model: $O(Kd^2)$ parameters in the covariance matrix.



From Figure 5.3 of Alpaydin (2004).



Bayes Classifier

- Idea: the means are class-specific, covariance matrix Σ is common.
- $O(d^2)$ parameters in the covariance matrix.

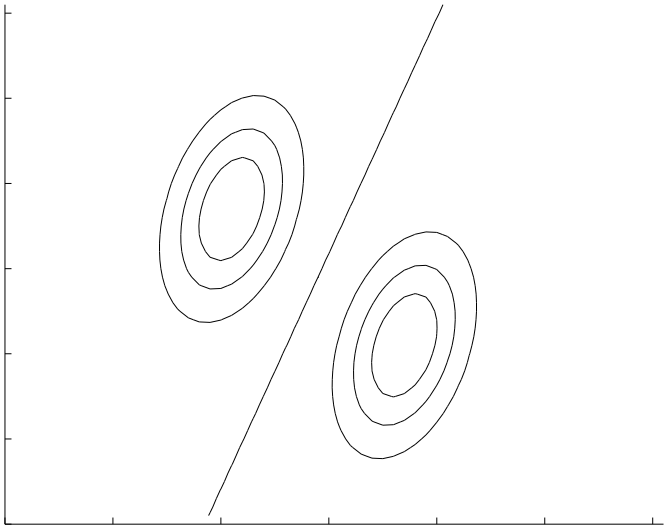
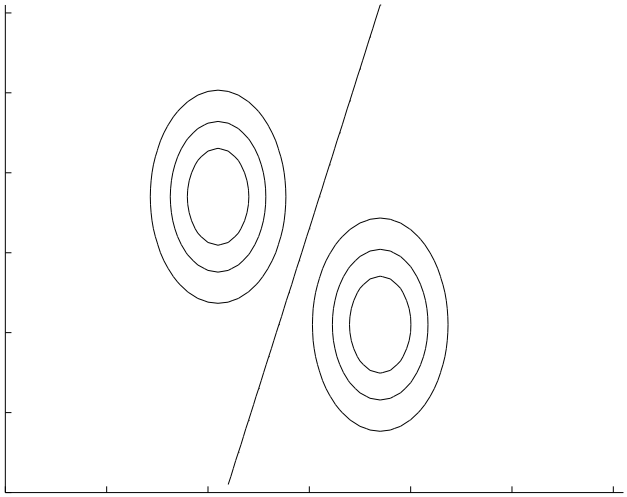


Figure 5.4: Covariances may be arbitary but shared by both classes. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Bayes Classifier

- Idea: the means are class-specific, covariance matrix Σ is common and diagonal (*Naive Bayes*).
- d parameters in the covariance matrix.
- Discriminant: $g_i(\mathbf{x}) = -\frac{1}{2}\sum_{j=1}^{d}(x_j^t - m_{ij})^2/s_j^2 + \log \hat{P}(C_i)$.



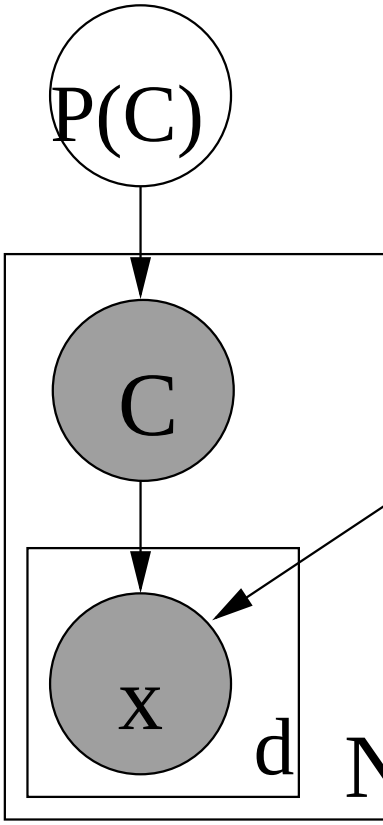
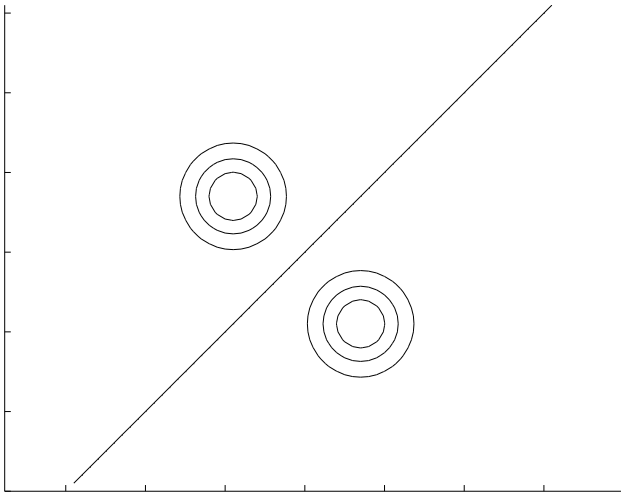


Figure 5.5: All classes have equal, diagonal covariance matrices but variances are not equal. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Bayes Classifier

- Idea: the means are class-specific, covariance matrix Σ is common and proportional to unit matrix $\Sigma = \sigma^2\mathbf{1}$.
- 1 parameter in the covariance matrix.
- Discriminant: $g_i(\mathbf{x}) = -||\mathbf{x} - \mathbf{m}_i||^2$.
- *Nearest mean classifier*. Each mean is a *prototype*.



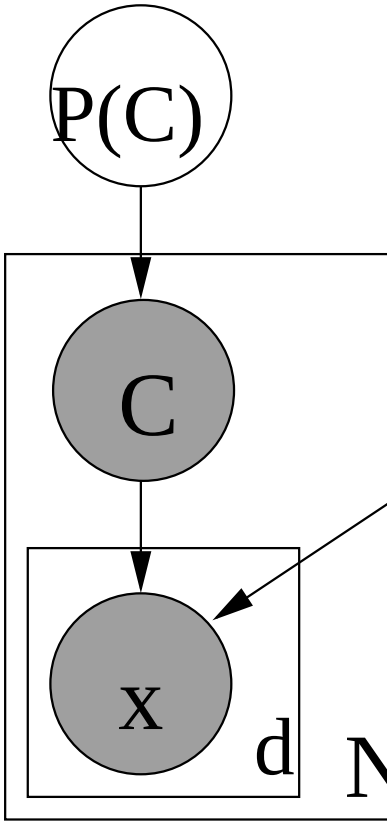


Figure 5.6: All classes have equal, diagonal covariance matrices of equal variances on both dimensions. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

18.2 Discrete Variables

Discrete Features

Most straightforward using Naive Bayes (replace Gaussian with Bernoulli):

- Binary features: $p_{ij} \equiv p(x_j = 1 \mid C_i)$

  if x_j are independent (Naive Baye

$$p(x \mid C_i) = \prod_{j=1}^{d} p_{ij}^{x_j} (1 - p_{ij})^{(1-x_j)}$$

  the discriminant is linear

$$g_i(\boldsymbol{x}) = \log p(\boldsymbol{x} \mid C_i) + \log P(C_i)$$
$$= \sum_j \left[x_j \log p_{ij} + (1 - x_j) \log (1 - p_{ij} \right.$$

Estimated parameters $\hat{p}_{ij} = \sum$

18.3 Multivariate Regression

Multivariate Regression

$$r^t = g\left(x^t \mid w_0, w_1, ..., w_d\right) + \varepsilon$$

- Multivariate linear model

$$w_0 + w_1 x_1^t + w_2 x_2^t + \cdots + w_d x$$

$$E\left(w_0, w_1, ..., w_d \mid \mathcal{X}\right) = \frac{1}{2}\sum_t \left[r^t - w_0 - w_1 x\right.$$

- Multivariate polynomial model:

  Define new higher-order variables

  $z_1=x_1$, $z_2=x_2$, $z_3=x_1^2$, $z_4=x_2^2$, $z_5=x_1x_2$

  and use the linear model in this new $\boldsymbol{z}$ spa

  (basis functions, kernel trick, SVM: Chapte

19 Dimensionality Reduction

19.1 Subset Selection

Why Reduce Dimensionality?

1. Reduces time complexity: Less com
2. Reduces space complexity: Less pa
3. Saves the cost of observing the fea
4. Simpler models are more robust o
5. More interpretable; simpler explan
6. Data visualization (structure, grou
   if plotted in 2 or 3 dimensions

Feature Selection vs. Extraction

- Feature selection: Choosing $k<d$ im
  ignoring the remaining $d - k$

  Subset selection algorithms
- Feature extraction: Project the

  original x_i , $i=1,...,d$ dimensions

  new $k<d$ dimensions, z_j , $j=1,...$

  Principal components analysis
  discriminant analysis (LDA), fac

- There are 2^d subsets of d features
- Forward search: Add the best featu
  - Set of features F initially Ø.
  - At each iteration, find the best new fea
    $$j = \text{argmin}_i E(F \cup x_i)$$
  - Add x_j to F if $E(F \cup x_j) < E(F)$

- Hill-climbing $O(d^2)$ algorithm
- Backward search: Start with all feat
  one at a time, if possible.
- Floating search (Add k, remove l)

Subset Selection

- Toy data set consists of 100 10-dimensional vectors from two classes (1 and 0).
- First two dimensions x_1^t and x_2^t: drawn from Gaussian with unit variance and mean of 1 or -1 for the classes 1 and 0, respectively.
- Remaining eight dimensions: drawn from Gaussian with zero mean and unit variance, that is, they contain no information of the class.
- Optimal classifier: If $x_1 + x_2$ is positive the class is 1, otherwise the class is 0.
- Use nearest mean classifier.

- Split data in random into training set of 30+30 items and validation set of 20+20 items.

Subset Selection

Forward selection:

Features	E_{VALID}
$\emptyset$	0.500
1	0.175
$1, \mathbf{2}$	**0.100**
$1, 2, \mathbf{4}$	0.100
$1, 2, 4, \mathbf{5}$	0.100
$1, 2, 4, 5, \mathbf{3}$	0.075
$1, 2, 4, 5, 3, \mathbf{8}$	**0.050**
$1, 2, 4, 5, 4, 8, \mathbf{6}$	0.075
$1, 2, 4, 5, 4, 8, 6, \mathbf{7}$	0.075
$1, 2, 4, 5, 4, 8, 6, 7, \mathbf{10}$	0.100
$1, 2, 4, 5, 4, 8, 6, 7, 10, \mathbf{9}$	0.150

Backward selection:

Features	E_{VALID}
$\mathbf{9}, 10, 4, 6, 7, 8, 3, 5, 2, 1$	0.150
$\mathbf{10}, 4, 6, 7, 8, 3, 5, 2, 1$	0.100
$\mathbf{4}, 6, 7, 8, 3, 5, 2, 1$	0.075
$\mathbf{6}, 7, 8, 3, 5, 2, 1$	0.075
$\mathbf{7}, 8, 3, 5, 2, 1$	0.075
$\mathbf{8}, 3, 5, 2, 1$	**0.050**
$\mathbf{3}, 5, 2, 1$	0.075
$\mathbf{5}, 2, 1$	0.100
$\mathbf{2}, 1$	**0.100**
$\mathbf{1}$	0.175
$\emptyset$	0.500

Optimal solution would be features $1, 2$!

19.2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA)

- PCA finds low-dimensional linear subspace such that when $\mathbf{x}$ is projected there information loss (here defined as variance) is minimized.
- Finds directions of maximal variance.
- *Projection pursuit*: find direction $\sqsupseteq$ such that some measure (here variance $\text{Var}(\mathbf{w}^T\mathbf{x})$) is maximized.
- Equivalent to finding eigenvalues and -vectors of covariance or correlation matrix.
- Can also be derived probabilistically (see Tipping ME, Bishop CM (1999) Mixtures of Probabilistic Principal Component Analyzers. Neural Computation 11: 443–482); probabilistic interpretation is important in deriving discrete variants.

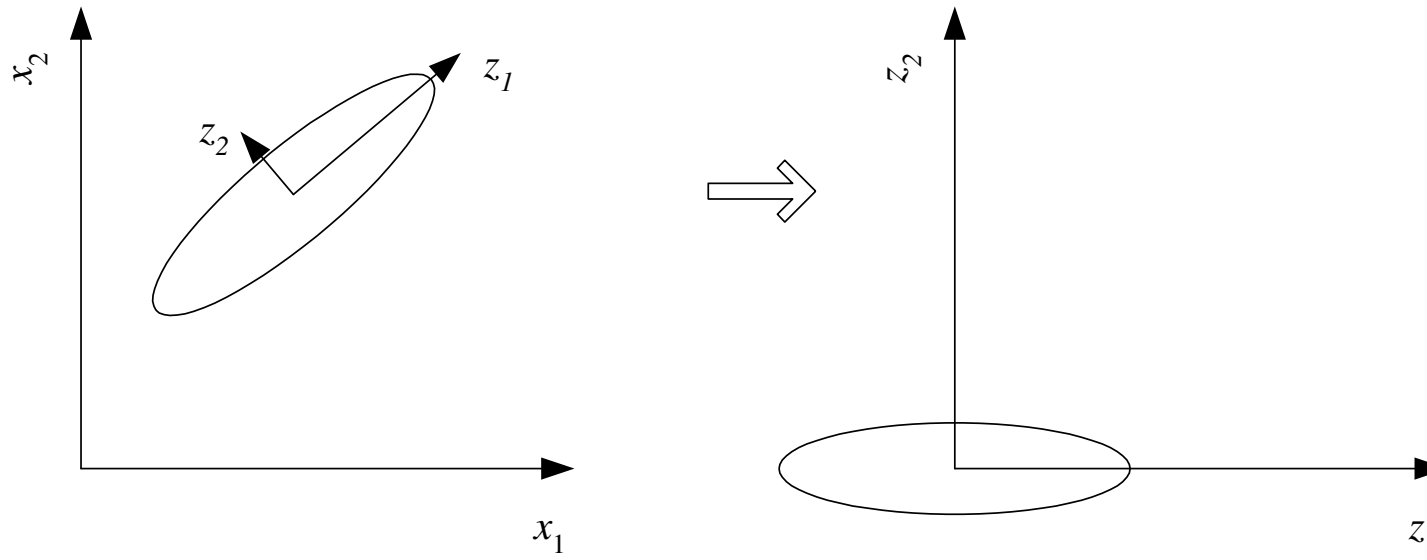


Figure 6.1: Principal components analysis centers the sample and then rotates the axes to line up wit the directions of highest variance. If the variance o z_2 is too small, it can be ignored and we have dimensionality reduction from two to one. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

```
> TOY1.pca <- princomp(TOY1[,1:10])
> summary(TOY1.pca)
Importance of components:
                          Comp.1    Comp.2    Comp.3    Comp.4     Comp.5     Comp.6     Comp.7
Standard deviation     1.7310919 1.1681823 1.1206590 1.0989518 1.01793023 0.96416923 0.86400744
Proportion of Variance 0.2637625 0.1201141 0.1105401 0.1062992 0.09120295 0.08182375 0.06570642
Cumulative Proportion  0.2637625 0.3838767 0.4944168 0.6007160 0.69191899 0.77374274 0.83944917
                           Comp.8     Comp.9    Comp.10
Standard deviation     0.83517208 0.78446118 0.71496201
Proportion of Variance 0.06139384 0.05416463 0.04499236
Cumulative Proportion  0.90084301 0.95500764 1.00000000
```

Principal Component Analysis (PCA)

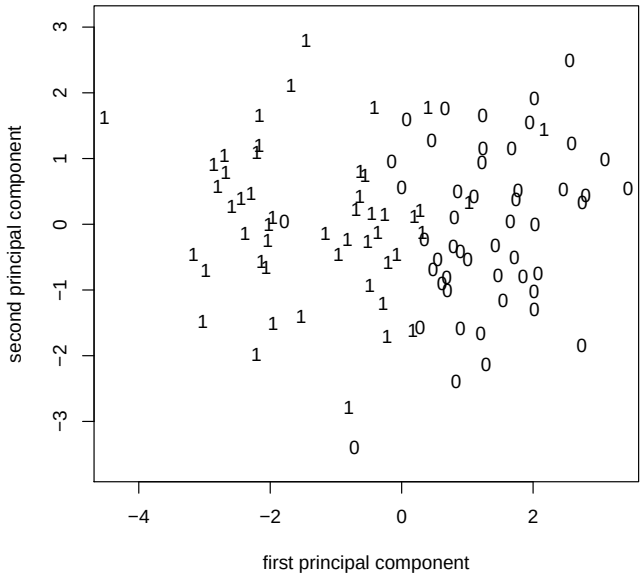
Previous 10-dimensional toy example:

```
> TOY1.pca <- princomp(TOY1[,1:10])
> summary(TOY1.pca)
Importance of components:
                          Comp.1    Comp.2    Comp.3    Comp.4     Comp
Standard deviation     1.7310919 1.1681823 1.1206590 1.0989518 1.01793
Proportion of Variance 0.2637625 0.1201141 0.1105401 0.1062992 0.09120
Cumulative Proportion  0.2637625 0.3838767 0.4944168 0.6007160 0.69191
                           Comp.8     Comp.9    Comp.10
Standard deviation     0.83517208 0.78446118 0.71496201
Proportion of Variance 0.06139384 0.05416463 0.04499236
Cumulative Proportion  0.90084301 0.95500764 1.00000000
```

Principal Component Analysis (PCA)

```
> TOY1.pca$loadings

Loadings:
    Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9 Comp.10
X1  -0.581          0.221 -0.431  0.188  0.187  0.488          0.174 -0.298
X2  -0.775 -0.166 -0.218  0.194 -0.200 -0.161 -0.353 -0.187         0.233
X3         -0.190  0.660  0.258  0.121 -0.352        -0.172 -0.495 -0.219
X4          0.162        -0.305 -0.384 -0.186 -0.428 -0.277  0.211 -0.623
X5   0.135 -0.655                -0.554 -0.239  0.346  0.101  0.230
X6   0.109  0.252                -0.140         0.361 -0.813         0.323
X7  -0.131  0.596  0.235         -0.310 -0.461  0.145  0.411         0.255
X8          0.175 -0.272  0.682         -0.200  0.333          0.308 -0.411
X9         -0.160 -0.302 -0.338  0.532 -0.685
X10                0.496  0.155  0.220         -0.267 -0.125  0.718  0.273
```

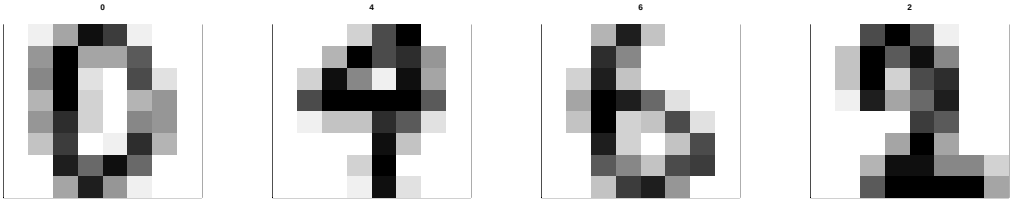
Principal Component Analysis (PCA)



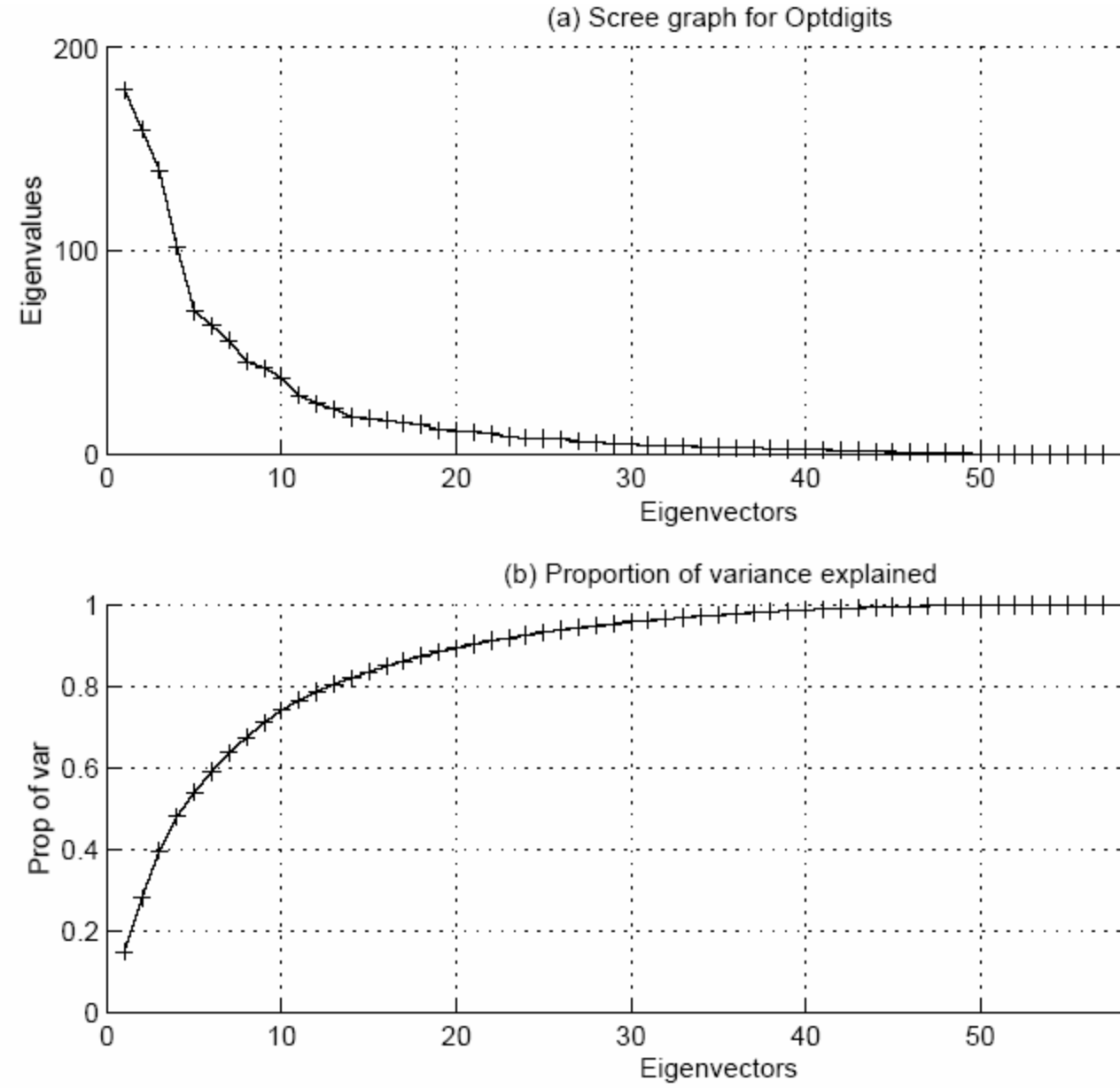

```
> TOY1.pc  <- predict(TOY1.pca)
> eqscplot(TOY1.pc[,1:2],type="n",
+          xlab="first principal component",
+          ylab="second principal component")
> text(TOY1.pc[,1:2],labels=as.character(TOY1[,"Class"]))
```

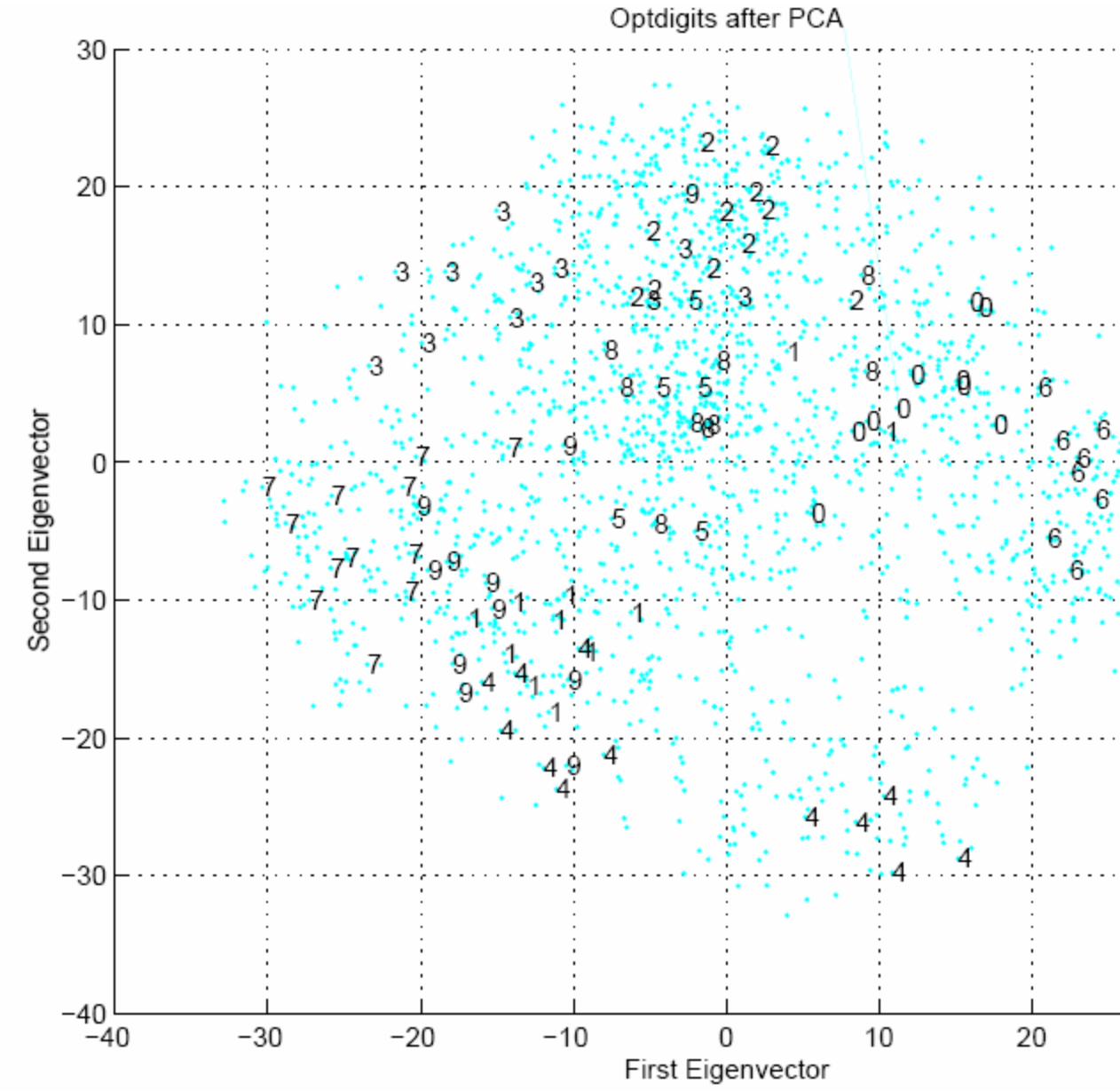
Example: Optdigits

- OPTDIGITS data set contains 5620 instances of digitized handwritten digits in range 0–9.
- Each digit is a $\mathbb{R}^{64}$ vector: $8 \times 8 = 64$ pixels, 16 grayscales.



Principal Component Analysis (PCA)


(a) Scree graph for Optdigits
Eigenvalues
200
100
0
0
10
20
30
40
50
Eigenvectors
(b) Proportion of variance explained
Prop of var
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40
50
Eigenvectors



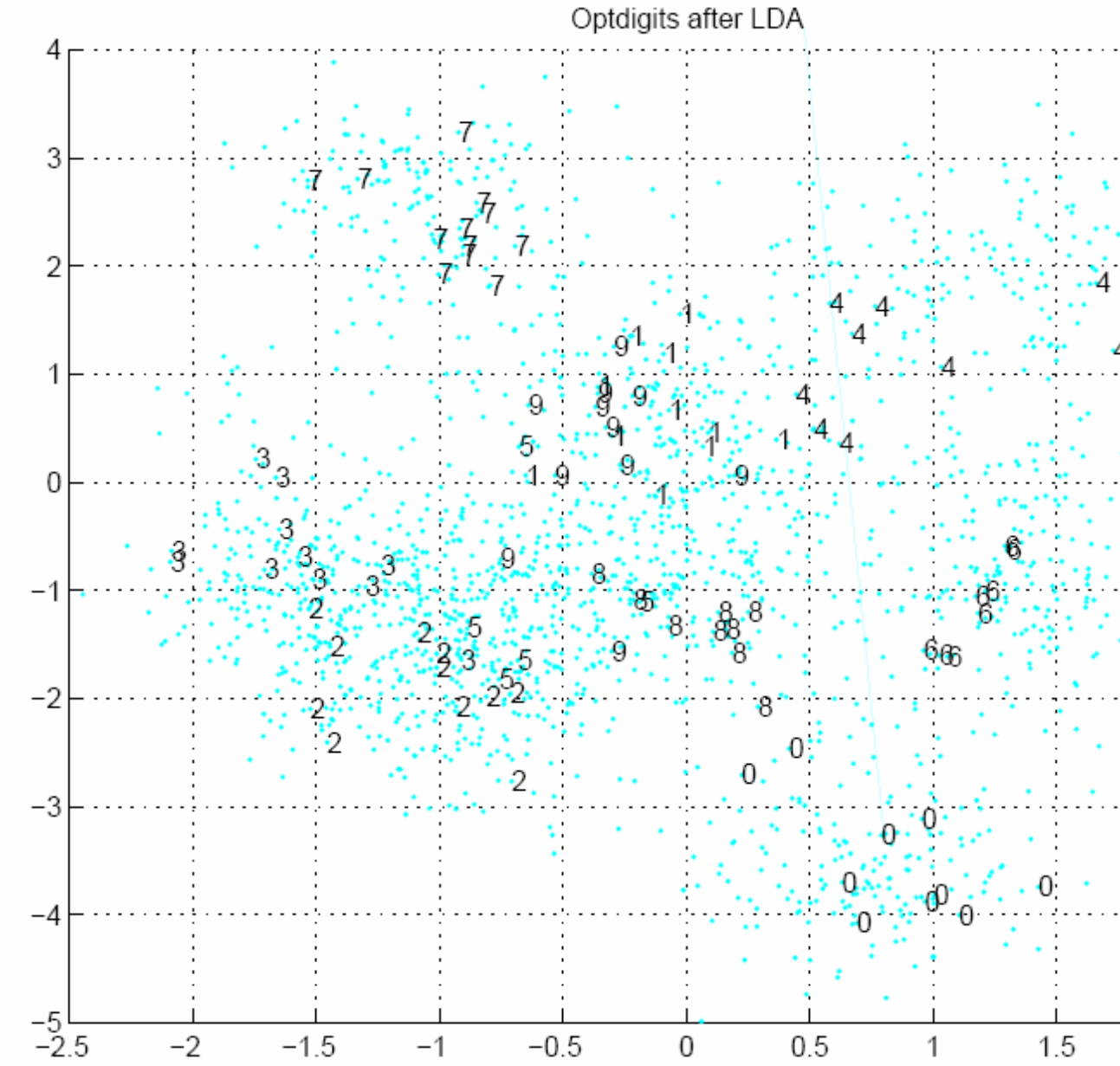
19.3 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDAA)

- Find a low-dimensional space such that when $\boldsymbol{x}$ is projected, classes are well-separated.
- Find $\boldsymbol{w}$ that maximizes

$$J(\boldsymbol{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$

$$m_1 = \frac{\sum_t \boldsymbol{w}^T \boldsymbol{x}^t r^t}{\sum_t r^t} \qquad s_1^2 = \sum_t \left(\boldsymbol{w}\right.$$



Lecture Notes for E Alpaydın 2004 Introduction to Machine Learning © The MIT Press (V1.1)

Remaining Lectures

- 6 Nov: Dimensionality Reduction & Clustering (Aplaydin Ch 6&7)
- 13 Nov: Clustering & Algorithms in Data Analysis (PDF chapter)
- 20 Nov: Assessing Algorithms & Decision Trees (Alpaydin Ch 14&9)
- 27 Nov: Machine Learning @ Google /TBA (additionally, Google recruitment talk in afternoon in T1 at 16 o'clock, see `http://www.cis.hut.fi/googletalk07/`)

- 4 Dec: Decision Trees & Linear Discrimination (Alpaydin Ch 10)
- (7 Dec: last problem session.)
- 11 Dec: Recap
- The plan is *preliminary* (may still change)

About the Text Book

- This course has Alpaydin (2004) as a text book.
- The lecture slides (neither mine nor the ones on the Alpaydin's site) are not meant to be a replacement for the text book.
- It is important also to read the book chapters.
- Library has some reading room copies (they are planning to order some more). If nothing else, you should probably at least copy some key chapters.

20 Dimensionality Reduction

20.1 Principal Component Analysis (PCA)

Principal Component Analysis (PCA)

- PCA finds low-dimensional linear subspace such that when $\mathbf{x}$ is projected there information loss (here defined as variance) is minimized.
- Finds directions of maximal variance.
- *Projection pursuit*: find direction $\mathbf{w}$ such that some measure (here variance $\text{Var}(\mathbf{w}^T\mathbf{x})$) is maximized.
- Equivalent to finding eigenvalues and -vectors of covariance or correlation matrix.

Principal Component Analysis (PCA)

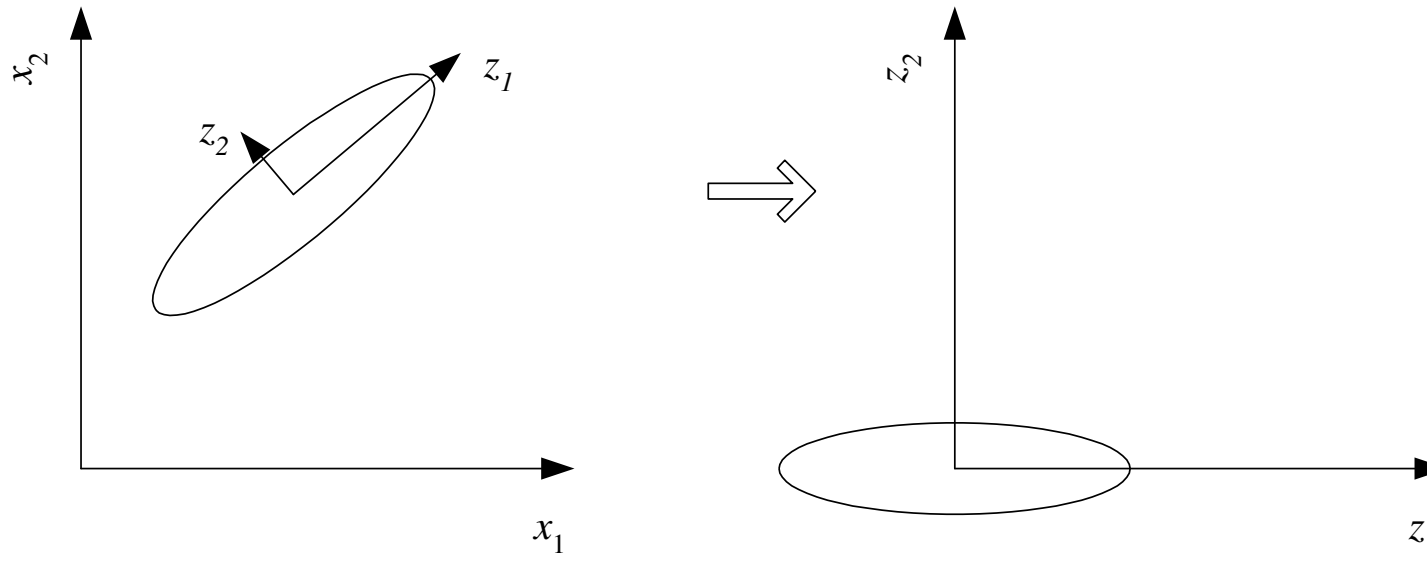


Figure 6.1: Principal components analysis centers the sample and then rotates the axes to line up wit the directions of highest variance. If the variance o z_2 is too small, it can be ignored and we have dimensionality reduction from two to one. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Principal Component Analysis (PCA)

- More formally: data $\mathcal{X} = \{\mathbf{x}^t\}_{t=1}^N$, $\mathbf{x}^t \in \mathbb{R}^d$.
- Center data: $\mathbf{y}^t = \mathbf{x}^t - \mathbf{m}$, where $\mathbf{m} = \sum_t \mathbf{x}^t / N$.
- Two options:

  - Use covariance matrix $S = \sum_t \mathbf{y}\mathbf{y}^T/N$.
  - Use correlation matrix R, where $R_{ij} = S_{ij}/\sqrt{S_{ii}S_{jj}}$.

- Diagonalize S (or R) using Singular Value Decomposition (SVD): $C^T S C = D$, where C is an orthogonal (rotation) matrix satisfying $CC^T = C^T C = \mathbf{1}$ and D is a diagonal matrix whose diagonal elements are the eigenvalues $\lambda_1 \geq \ldots \geq \lambda_d \geq 0$.

- ith column of C is the ith eigenvector.

- Project data vectors $\mathbf{y}^t$ to principal components $\mathbf{z}^t = C^T \mathbf{y}^t$ (equivalently $\mathbf{y}^t = C\mathbf{z}^t$).

Principal Component Analysis (PCA)

- Observation: covariance matrix of $\{\mathbf{z}^t\}_{t=1}^N$ is a diagonal matrix D whose diagonal elements are the variances.

$$\begin{aligned} S_{\mathbf{z}} &= \sum_t \mathbf{z}\mathbf{z}^T/N = \sum_t C^T \mathbf{y}\mathbf{y}^T C/N \\ &= C^T \left(\sum_t \mathbf{y}\mathbf{y}^T/N \right) C = C^T S C = D, \end{aligned}$$

  where the diagonal elements of D are the variances $D_{ii} = \sigma^2_{\mathbf{z}i}$.

- Eigenvalues $\lambda_i \Leftrightarrow$ variances σ_i^2.

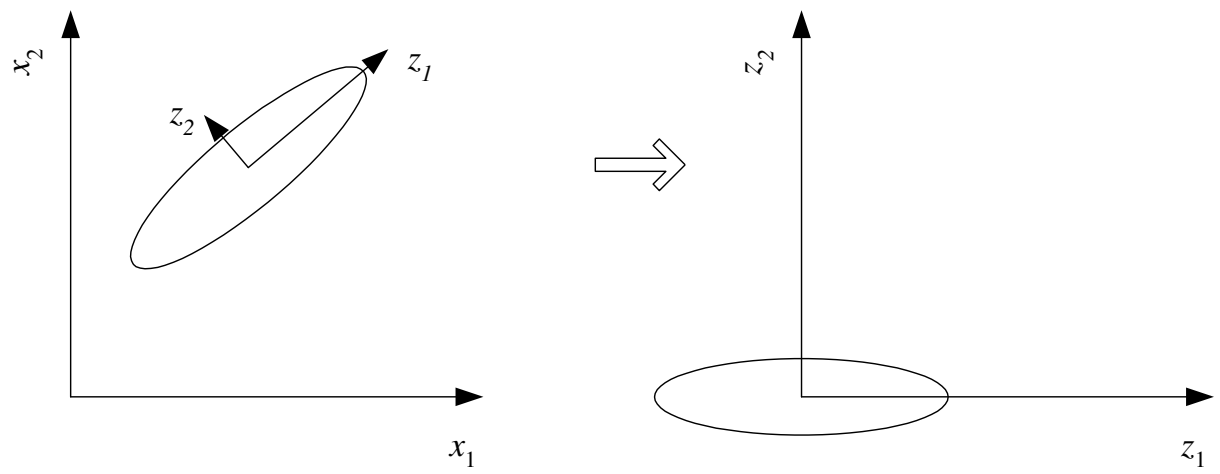


Figure 6.1: Principal components analysis centers the sample and then rotates the axes to line up with the directions of highest variance. If the variance on z_2 is too small, it can be ignored and we have dimensionality reduction from two to one. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Principal Component Analysis (PCA)

- Idea: in the PC space ($\mathbf{z}$ space), k first principal components explain the data well enough, where $k < d$.
- "Well enough" means here that the *reconstruction error* is small enough. More formally:
- Project the data vectors $\mathbf{y}^t$ into $\mathbb{R}^k$ using $\hat{\mathbf{z}}^t = W^T\mathbf{y}^t$, where $W \in \mathbb{R}^{d\times k}$ is a matrix containing the first k columns of C. ("W <- C[,1:k]"). $\hat{\mathbf{z}}^t$ is a representation of $\mathbf{y}^t$ in k dimensions.
- Project $\hat{\mathbf{z}}^t$ back to $\mathbf{y}^t$ space:
$$\hat{\mathbf{y}}^t = W\hat{\mathbf{z}}^t = WW^T\mathbf{y}^t$$
What is the average reconstruction error $\mathcal{E} = \sum_t \left(\hat{\mathbf{y}}^t - \mathbf{y}^t\right)^T \left(\hat{\mathbf{y}}^t - \mathbf{y}^t\right)/N$?

Principal Component Analysis (PCA)

- What is the average reconstruction error $\mathcal{E} = \sum_t \left(\hat{\mathbf{y}}^t - \mathbf{y}^t\right)^T \left(\hat{\mathbf{y}}^t - \mathbf{y}^t\right)/N$?

$$\begin{aligned}
\mathcal{E} &= \mathrm{Tr}(E\left[(\hat{\mathbf{y}} - \mathbf{y})\,(\hat{\mathbf{y}} - \mathbf{y})\right]) \\
&= \mathrm{Tr}\Big(\left(WW^T - \mathbf{1}\right) E\left[\mathbf{y}\mathbf{y}^T\right] \left(WW^T - \mathbf{1}\right)\Big) \\
&= \mathrm{Tr}\left(WW^T CDC^T WW^T\right) + \mathrm{Tr}\left(CDC^T\right) - 2\mathrm{Tr}\left(W^T CDC^T W\right) \\
&= \sum_{i=k+1}^{d} \lambda_i,
\end{aligned}$$

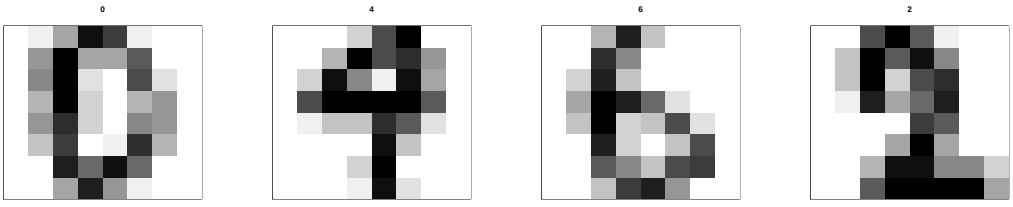
  where we have used the fact that $S = CDC^T = E\left[\mathbf{y}\mathbf{y}^T\right]$ and the cyclic property of the trace, $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$.

Principal Component Analysis (PCA)

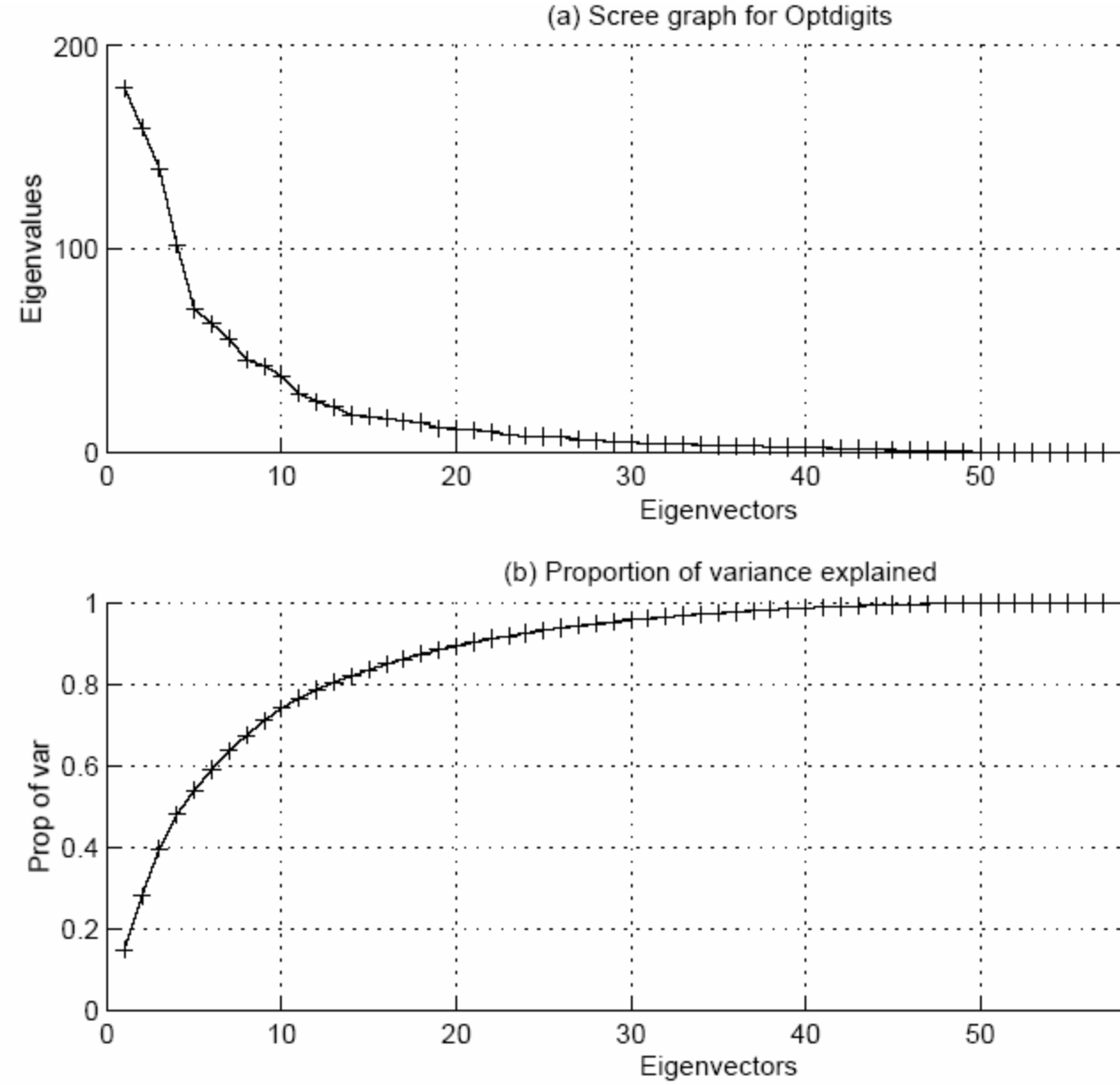
- Result: PCA is a linear projection of data from $\mathbb{R}^d$ into $\mathbb{R}^k$ such that the average reconstruction error $\mathcal{E} = E\left[\left(\hat{\mathbf{y}} - \mathbf{y}\right)^T \left(\hat{\mathbf{y}} - \mathbf{y}\right)\right]$ is minimized.
- *Proportion of Variance (PoV) Explained*: $\mathrm{PoV} = \sum_{i=1}^{k} \lambda_i / \sum_{i=1}^{d} \lambda_i$. Some rules of thumb to find a good k: PoV ≈ 0.9, or PoV curve has an elbow.
- *Dimension reduction*: it may be sufficient to use $\hat{\mathbf{z}}^t$ instead of $\hat{\mathbf{x}}^t$ to train a classifier etc.
- *Visualization*: plotting the data to $\hat{\mathbf{z}}^t$ using $k = 2$ (first thing to do with new data).
- *Data compression*: instead of storing the full data vectors $\mathbf{y}^t$ it may be sufficient to store only $\hat{\mathbf{z}}^t$ and then reconstruct the original data using $\hat{\mathbf{y}}^t = W\hat{\mathbf{z}}^t$, if necessary.

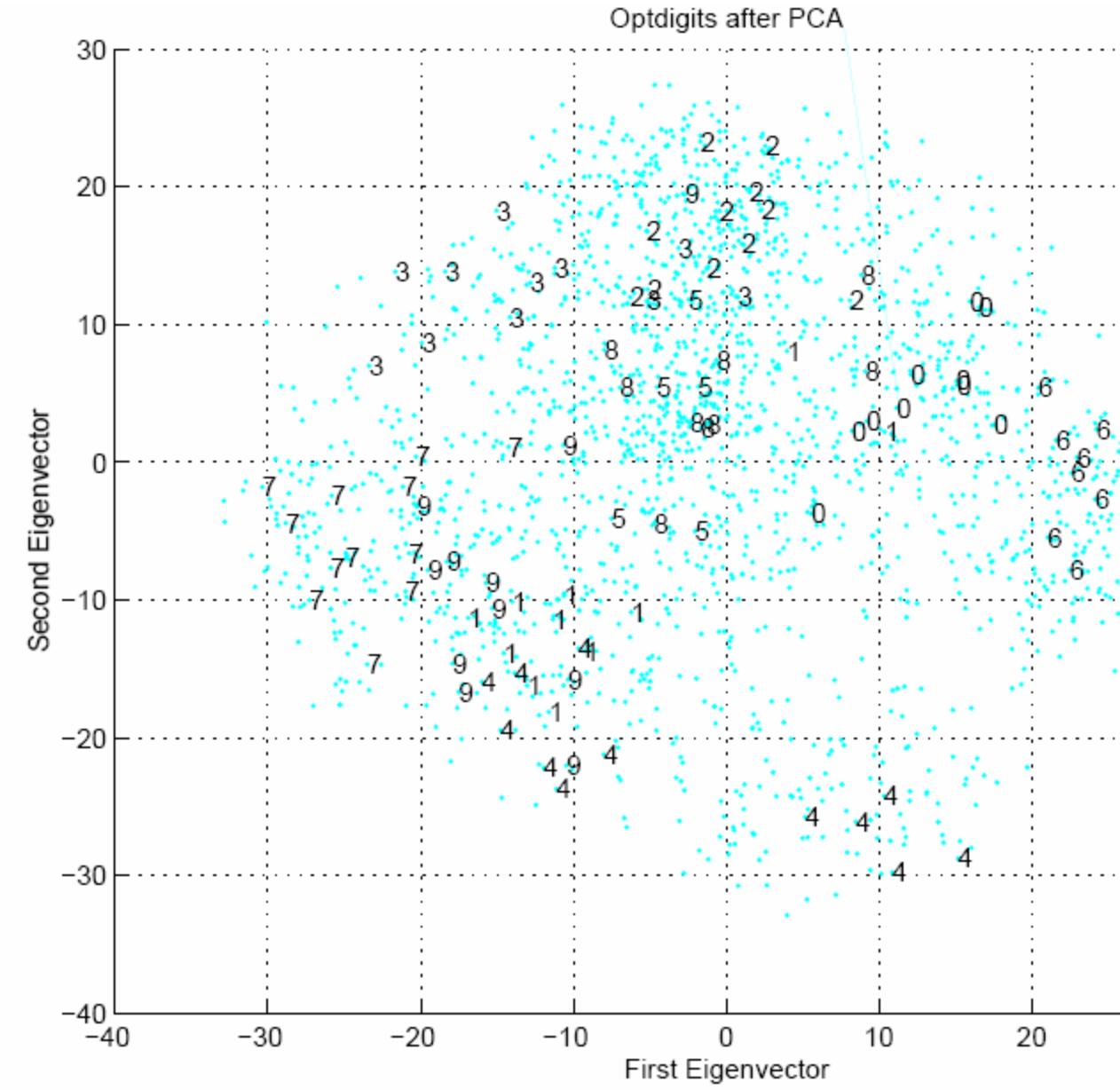
Example: Optdigits

- OPTDIGITS data set contains 5620 instances of digitized handwritten digits in range 0–9.
- Each digit is a $\mathbb{R}^{64}$ vector: $8 \times 8 = 64$ pixels, 16 grayscales.



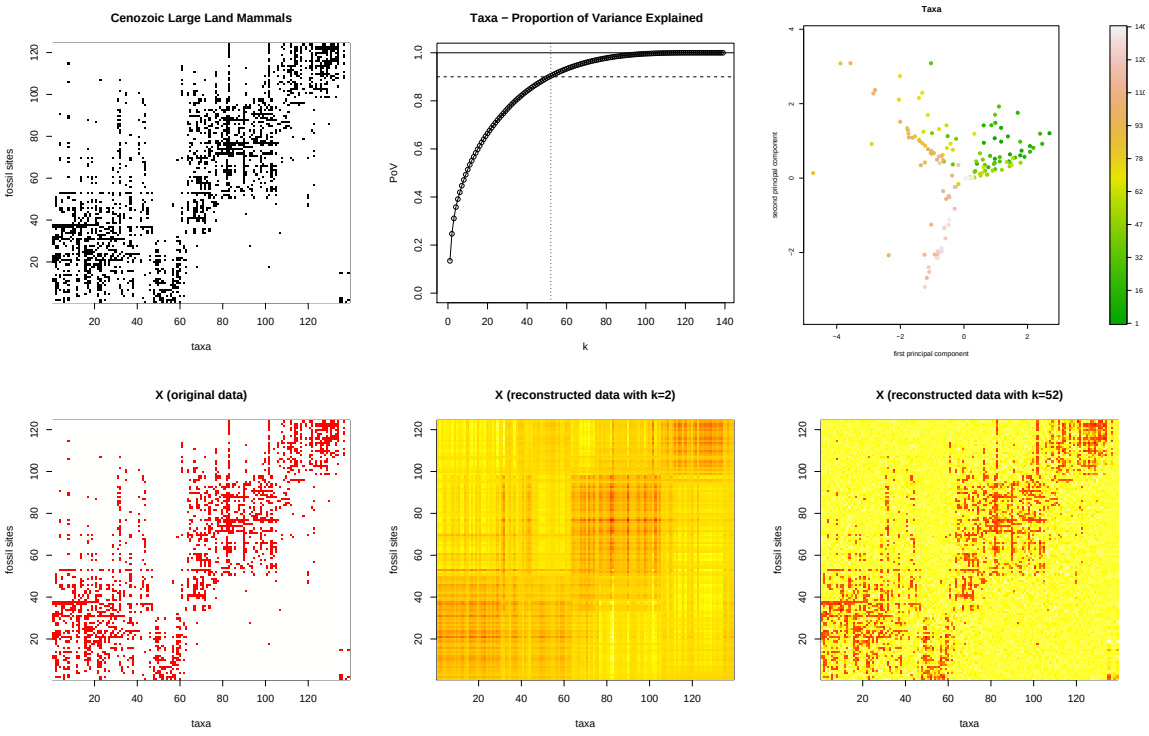
Example: Optdigits





Example: Fossils

- Large European land mammals: 124 fossil find sites (dated 23–2 million years old), 139 taxa
- Reconstruction of site vectors given PCA taxon representation for different k: $\hat{\mathbf{y}} = W\hat{\mathbf{z}} = WW^T\mathbf{y}$, or $\hat{\mathbf{x}} = WW^T(\mathbf{x} - \mathbf{m}) + \mathbf{m}$.



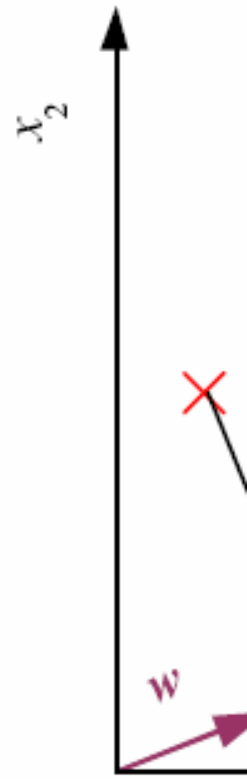
20.2 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA)

- PCA is unsupervised method (class information is not usually used).
- Linear Discriminant Analysis (LDA) is *supervised* method for dimensionality reduction in classification problems.
- As PCA, LDA can be accomplished with standard matrix algebra (eigenvalue decompositions etc.). This makes it relatively simple and useful.
- PCA is a good general purpose dimensionality reduction method, LDA is a good alternative if we want to optimize the separability of classes in a specific classification task, and are happy with dimensionality of less than the number of classes ($k < K$).

Linear Discriminant Analysis (LDA)

- Find a low-dimensional space such that when $\boldsymbol{x}$ is projected, classes are well-separated.
- Find $\boldsymbol{w}$ that maximizes



$$J(\boldsymbol{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$

$$m_1 = \frac{\sum_t \boldsymbol{w}^T \boldsymbol{x}^t r^t}{\sum_t r^t} \qquad s_1^2 = \sum_t \left(\boldsymbol{w}^T \right.$$

Linear Discriminant Analysis (LDA)

- More formally: data $\mathcal{X} = \{(\mathbf{r}^t, \mathbf{x}^t)\}_{t=1}^N$, where r_i^t is one if $\mathbf{x}^t$ is in class i, zero otherwise, and $\mathbf{x}^t \in \mathbb{R}^d$.
- *Within-class scatter*: $S_W = \sum_{i=1}^K S_i$, where $S_i = \sum_t r_i^t \left(\mathbf{x}^t - \mathbf{m}_i\right) \left(\mathbf{x}^t - \mathbf{m}_i\right)^T$.

- *Between-class scatter*: $S_B = \sum_{i=1}^{K} N_i \left(\mathbf{m}_i - \mathbf{m}\right)\left(\mathbf{m}_i - \mathbf{m}\right)^T$, where $N_i = \sum_t r_i^t$. ($\text{rank}(S_B) < K$)
- $k = 1$: find $\mathbf{w} \in \mathbb{R}^d$ that maximizes Fisher's discriminant

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}.$$

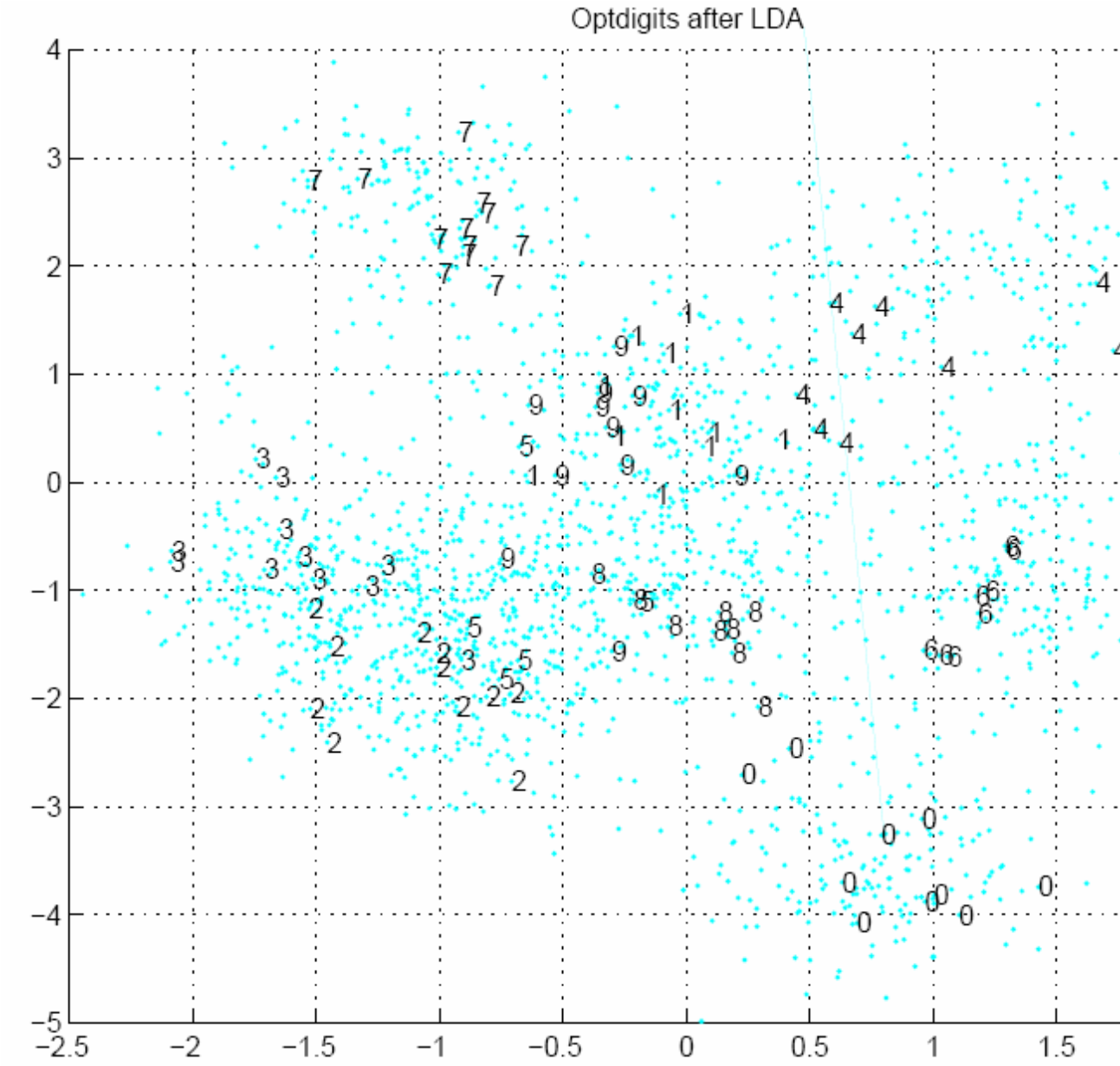
- $K > k > 1$: find $W \in \mathbb{R}^{d \times k}$ that maximizes Fisher's discriminant

$$J(W) = \frac{\left|W^T S_B W\right|}{\left|W^T S_W W\right|}.$$

- The projection from $\mathbb{R}^d$ to $\mathbb{R}^k$ is given by $\hat{\mathbf{z}} = W^T(\mathbf{x} - \mathbf{m})$.
- Find $W \in \mathbb{R}^{d \times k}$ that maximizes Fisher's discriminant

$$J(W) = \frac{\left|W^T S_B W\right|}{\left|W^T S_W W\right|}.$$

- Write $V = S_W^{1/2} W \in \mathbb{R}^{d \times k}$, where $S_W^{1/2}$ is a matrix such that $S_W^{1/2} S_W^{1/2} = S_W$: $J(V) = \left|V^T S_W^{-1/2} S_B S_W^{-1/2} V\right| / \left|V^T V\right|$.
- Determinant is a product of eigenvalues. To maximize $J(V)$ V must contain the k largest eigenvectors of $S_W^{-1/2} S_B S_W^{-1/2}$ (like in PCA!): $V^T S_W^{-1/2} S_B S_W^{-1/2} V = D \Leftrightarrow W S_W^{-1/2} S_W^{-1/2} S_B S_W^{-1/2} S_W^{1/2} W = D \Leftrightarrow W^T S_W^{-1} S_B W = D$.
- $\Rightarrow$ LDA is the k largest eigenvector decomposition of $S_W^{-1} S_B$ (like PCA is of covariance matrix).
- At most $K - 1$ non-zero eigenvalues, that is, one should choose $k < K$.



Lecture Notes for E Alpaydın 2004 Introduction to Machine Learning © The MIT Press (V1.1)

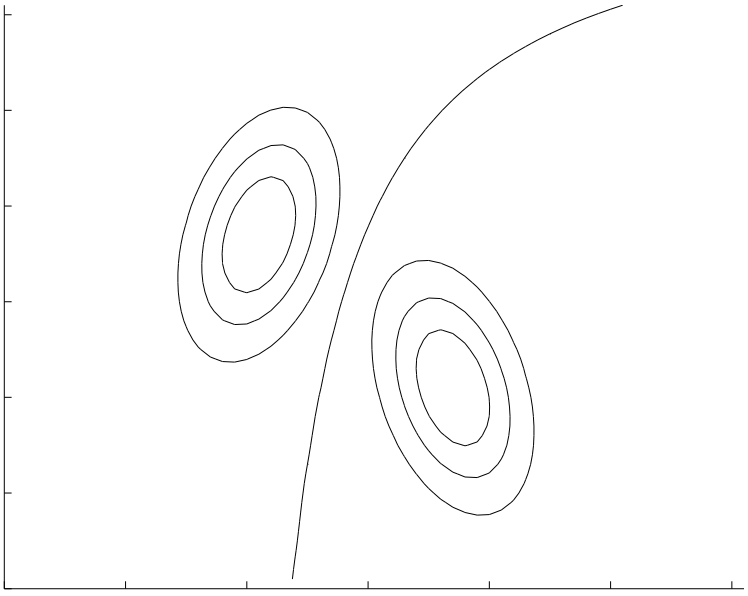
21 Clustering

21.1 Introduction

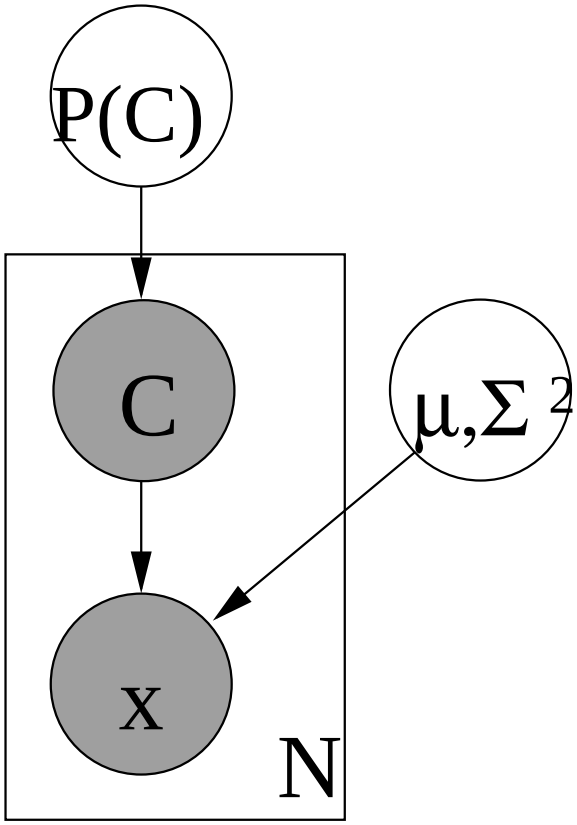
Mixture densities

- $p(\mathbf{x}) = \sum_{i=1}^{k} p(\mathbf{x} \mid C_i) p(C_i)$
- *Classification*: labels $\mathbf{r}^t$ are known in training data. Task: predict $\mathbf{r}$ for new data vectors $\mathbf{x}$

- *Clustering*: data is unlabeled, that is, $\mathbf{r}^t$ are unknown. Task: assign a *cluster* label $\mathbf{r}$ for new data vectors $\mathbf{x}$.
- Gaussian mixture model:



From Figure 5.3 of Alpaydin (2004).



Classes vs. Clusters

- Supervised: $\mathcal{X} = \{ \boldsymbol{x}^t, \boldsymbol{r}^t \}_t$
- Classes C_i i=1,...,K

$$p(\boldsymbol{x}) = \sum_{i=1}^{K} p(\boldsymbol{x} \mid C_i) P(C_i)$$

where $p(\boldsymbol{x} \mid C_i) \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$

- $\Phi = \{P(C_i), \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{K}$

$$\hat{P}(C_i) = \frac{\sum_t r_i^t}{N} \quad \boldsymbol{m}_i = \frac{\sum_t r_i^t \boldsymbol{x}^t}{\sum_t r_i^t}$$

$$\mathbf{S}_i = \frac{\sum_t r_i^t (\boldsymbol{x}^t - \boldsymbol{m}_i)(\boldsymbol{x}^t - \boldsymbol{m}_i)^T}{\sum_t r_i^t}$$

- U
- C
- p
- w
- Φ

21.2 K-means Clustering

k-means Clustering

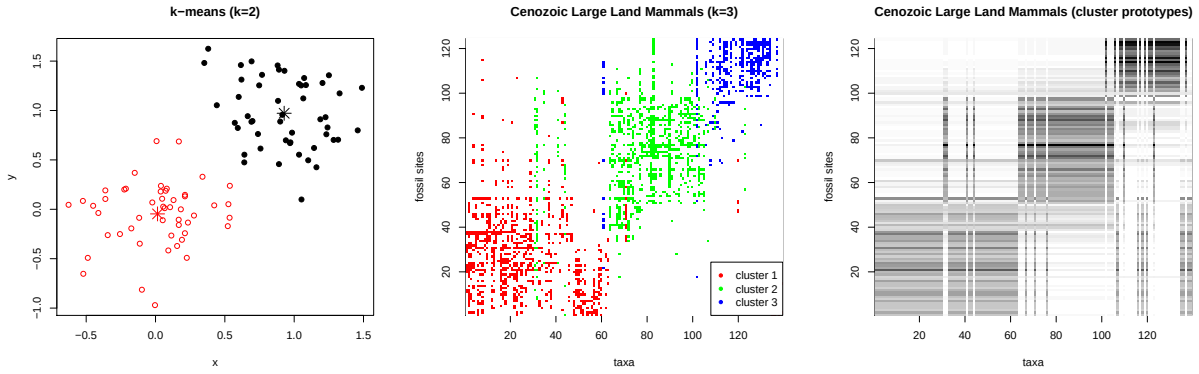
- The simplest Bayesian classifier was nearest mean classifier: classify a data vector to class which has a nearest mean.
- *k-means clustering*: find k prototype vectors $\mathbf{m}_i$ ("means") which best represent data.
- Error function:

$$\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X}) = \sum_{t=1}^{N} \min_i ||\mathbf{x}^t - \mathbf{m}_i||^2.$$

- Task: find prototype vectors $\mathbf{m}_i$ such that error $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X})$ is minimized.
- No direct probabilistic interpretation. Can be viewed as *approximation* of the Bayesian nearest mean classifier where data vector belongs to a class/cluster with probability 0 or 1 only.

k-means Clustering

- The vectors are assigned to the nearest means.
- In R: cl <- kmeans(t(X),centers=3)



k-means Clustering

- *Compression*: a real vector (image etc.) can be represented with a number in $\{1, \ldots, k\}$.
- *Dimensionality reduction*: one can use cluster indexes instead of the real vectors to train a classifier etc.
- *Interpretation of the data*: clusters have often a meaning. Taxa from various time periods, customer segments, etc.
- *Labeling of data*: cluster indexes may be used as class labels.

k-means Clustering

Figure 9.3 of Bishop (2006).

- Data set is the set of pixels.
- Each pixel is a vector in three-dimensional RGB space.
- K-means is applied to the data set of pixels of an image.
- The compressed representation is then the prototype vectors, and cluster index for each pixel.

k-means Clustering

- *Lloyd's algorithm*: the most famous algorithm to minimize the k-means cost function. Easy to understand and implement.
- Sensitive to initialization: should be run on several random initializations and choose the result with the smallest cost.
- In practice one should consider some more advanced method (type help(kmeans) in R for some suggestions).

Initialize $\boldsymbol{m}_i, i = 1, \ldots, k$, for example, to k random $\boldsymbol{x}^t$

Repeat

For all $\boldsymbol{x}^t \in \mathcal{X}$

$$b_i^t \leftarrow \begin{cases} 1 & \text{if } \|\boldsymbol{x}^t - \boldsymbol{m}_i\| = \min_j \|\boldsymbol{x}^t - \boldsymbol{m}_j\| \\ 0 & \text{otherwise} \end{cases}$$

For all $\boldsymbol{m}_i, i = 1, \ldots, k$

$$\boldsymbol{m}_i \leftarrow \sum_t b_i^t \boldsymbol{x}^t / \sum_t b_i^t$$

Until $\boldsymbol{m}_i$ converge

Figure 7.3: k-means algorithm. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

k-means Clustering

Initialize $\mathbf{m}_i$, $i = 1, \ldots, k$, randomly.
repeat
  for all $t \in \{1, \ldots, N\}$ **do** {E step}

$$b_i^t \leftarrow \begin{cases} 1 & , \quad i = \arg\min_i ||\mathbf{x}^t - \mathbf{m}_i|| \\ 0 & , \quad \text{otherwise} \end{cases}$$

  end for
  for all $i \in \{1, \ldots, k\}$ **do** {M step}

$$\mathbf{m}_i \leftarrow \frac{\sum_t b_i^t \mathbf{x}^t}{\sum_t b_i^t}$$

  end for
until the error $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X})$ does not change

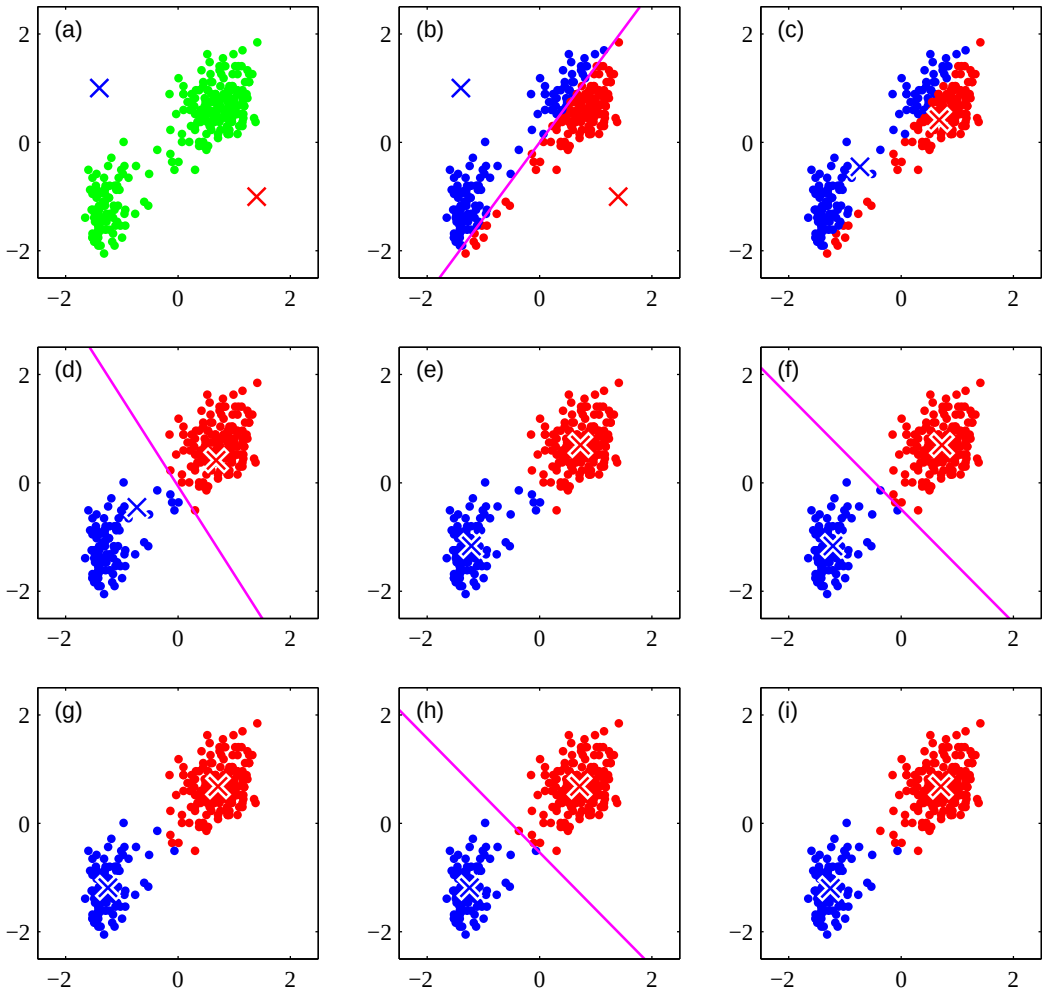


Figure 9.1 of Bishop (2006)

Observations:

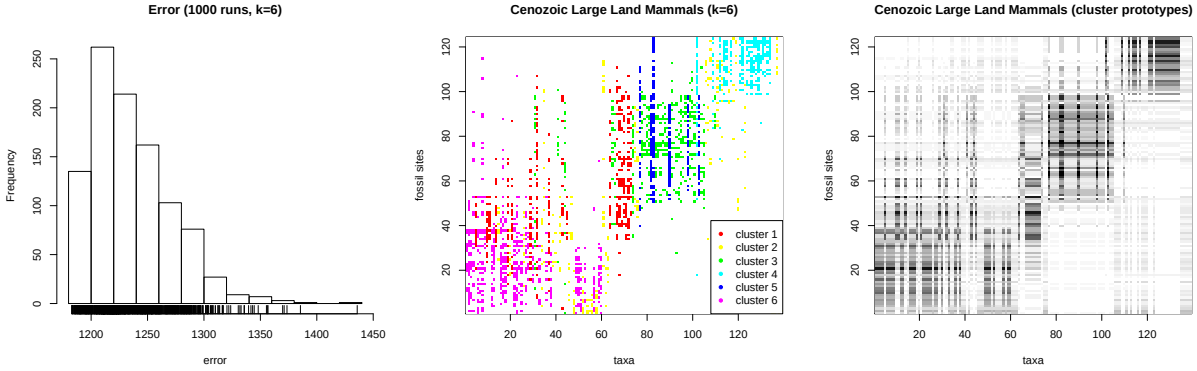
- Iteration cannot increase the error $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X})$.
- There are finite number, k^N, of possible clusterings.
- It follows that the algorithm always stops after a finite time. (It can take no more than k^N steps.)
- Usually k-means is however relatively fast. "In practice the number of iterations is generally much less than the number of points." (Duda & Hart & Stork, 2000)
- *Worst-case running time* with really bad data and really bad initialization is however $2^{\Omega(\sqrt{N})}$ — luckily this usually does not happen in real life (David A, Vassilivitskii S (2006) How slow is the k-means method? In Proc 22nd SCG.)

Observations:

- The result can in the worst case be really bad.
- Example:
  - Four data vectors ($N = 4$) from $\mathbb{R}^d$ in $\mathcal{X}$: $\mathbf{x}^1 = (0, 0, \ldots, 0)^T$, $\mathbf{x}^2 = (1, 0, \ldots, 0)^T$, $\mathbf{x}^3 = (0, 1, \ldots, 1)^T$ and $\mathbf{x}^4 = (1, 1, \ldots, 1)^T$.
  - Optimal clustering into two ($k = 2$) is given by the prototype vectors $\mathbf{m}_1 = (0.5, 0, \ldots, 0)^T$ and $\mathbf{m}_2 = (0.5, 1, \ldots, 1)^T$, error being $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X}) = 1$.
  - Lloyd's algorithm can however converge also to $\mathbf{m}_1 = (0, 0.5, \ldots, 0.5)^T$ and $\mathbf{m}_2 = (1, 0.5, \ldots, 0.5)^T$, error being $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X}) = d - 1$. (Check that iteration stops here!)

k-means Clustering

- Example: cluster taxa into $k = 6$ clusters 1000 times with Lloyd's algorithm.
- The error $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X})$ is different for different runs!
- You should try several random initializations, and choose the solution with smallest error.
- For a cool initialization see Arthur D, Vassilivitskii S (2006) k-means++: The Advantages of Careful Seeding.



21.3 EM Algorithm

EM Algorithm

- *Expectation-Maximization algorithm* (EM): soft cluster assignments
- Probabilistic interpretation

EM Algorithm

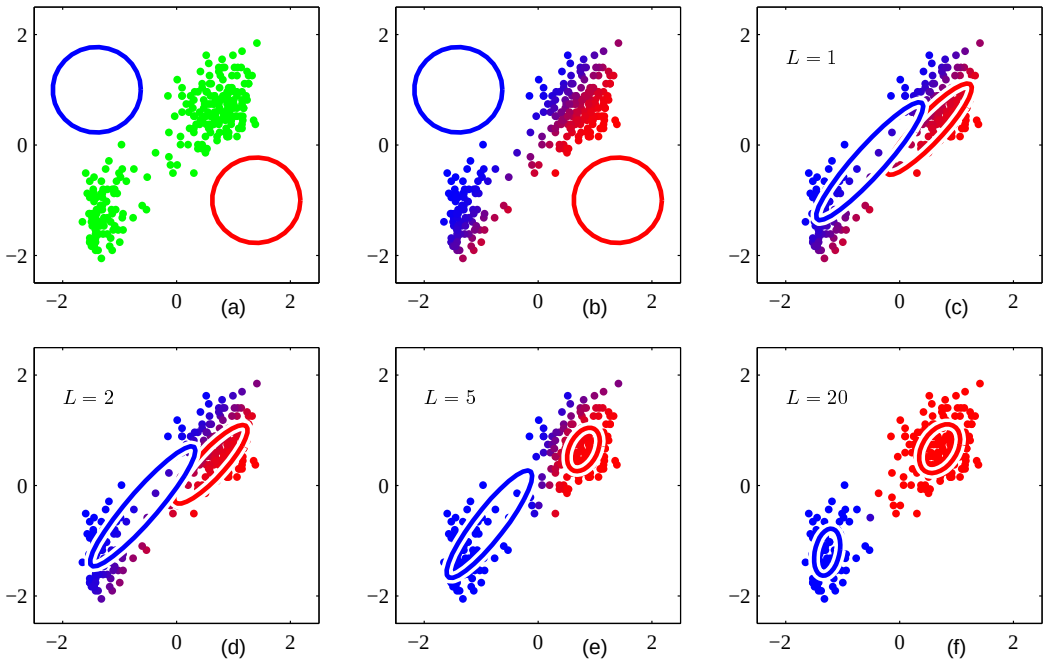
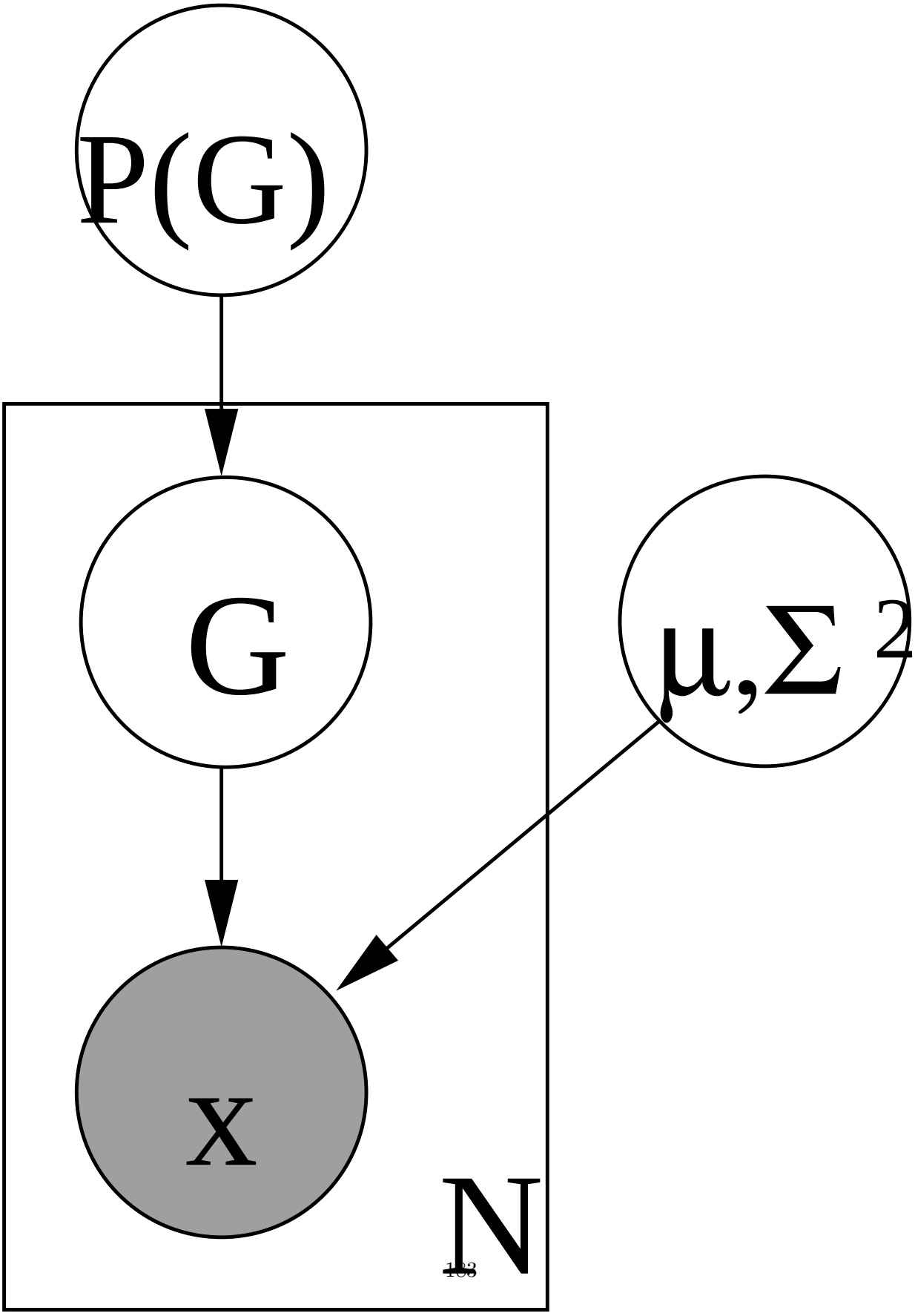


Figure 9.8 of Bishop (2006)

- EM algorithm is like k-means, except cluster assignments are "soft": each data point is a member of a given cluster with certain probability.
- $b_i^t \in \{0, 1\} \longrightarrow h_i^t \in [0, 1]$.

EM Algorithm

- Find maximum likelihood solution of the mixture model $\mathcal{L} = \log \prod_{t=1}^{N} p(\mathbf{x}^t \mid \theta)$, where the parameters θ are μ_i, Σ_i and $\pi_i = P(G_i)$.
- Maximum likelihood solution is found by the EM algorithm (which is essentially generalization of the Lloyd's algorithm to soft cluster memberships)
- Idea: iteratively find the membership weights of each data vector in clusters, and the parameter values. Continue until convergence.
- End result is intuitive.


P(G)
G
μ,Σ 2
X
N

EM Algorithm

Initialize $\mathbf{m}_i$ and π_i, $i = 1, \ldots, k$, randomly.
repeat
  for all $t \in \{1, \ldots, N\}$ **do** {E step}

$$h_i^t \leftarrow \frac{\pi_i \exp\left[-\frac{1}{2s^2}\left|\left|\mathbf{x}^t - \mathbf{m}_i\right|\right|^2\right]}{\sum_j \pi_j \exp\left[-\frac{1}{2s^2}\left|\left|\mathbf{x}^t - \mathbf{m}_j\right|\right|^2\right]}$$

  end for
  for all $i \in \{1, \ldots, k\}$ **do** {M step}

$$\mathbf{m}_i \leftarrow \frac{\sum_t h_i^t \mathbf{x}^t}{\sum_t h_i^t}$$

$$\pi_i \leftarrow \frac{\sum_t h_i^t}{N}$$

  end for
until convergence

EM Algorithm

- For derivation, see Alpaydin (2004), section 7.4 (pages 139–144); for an alternative derivation, see Bishop (2006), section 9.4 (pages 450–455). A sketch of follows.
- Task: find an ML solution of a likelihood function given by $p(\mathbf{X} \mid \theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} \mid \theta)$.

$$\begin{aligned}\sum_t \log p(\mathbf{x}^t \mid \theta) &\geq \sum_t \log p(\mathbf{x}^t \mid \theta) - \sum_t KL(h_i^t \mid\mid p(\mathbf{z}^t \mid \mathbf{x}^t, \theta)) \\ &= \sum_t \sum_i h_i^t \log p(\mathbf{x}^t, \mathbf{z}^t \mid \theta) + \sum_t H(h_i^t),\end{aligned}$$

  where we have used the *Kullback-Leibler* (KL) divergence $KL(q(i) \mid\mid p(i)) = \sum_i q(i) \log\left(q(i)/p(i)\right)$. KL divergence is always non-negative and it vanishes only when the distributions q and p are equal. The entropy is given by $H(q(i)) = -\sum_i q(i) \log q(i)$.
- *Expectation step* (E Step): find h_i^t by minimizing the KL divergence.
- *Maximization step* (M Step): find θ by maximizing the expectation.

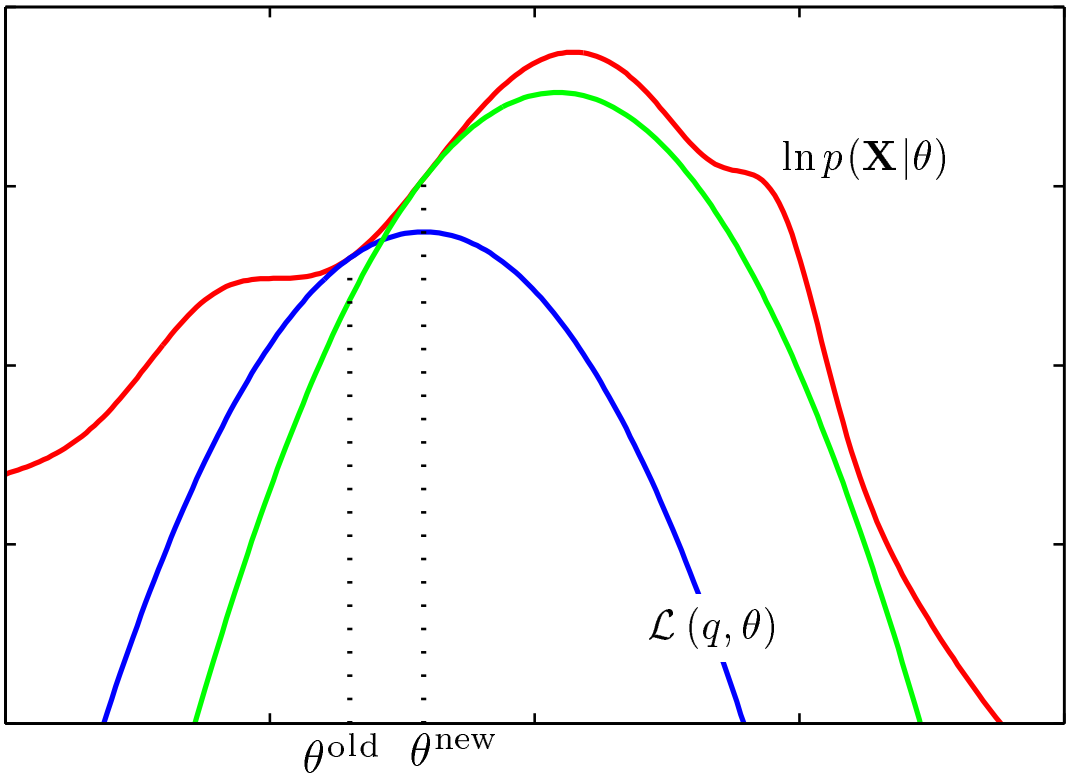


Figure 9.14

of Bishop (2006)

22 Clustering

22.1 k-means Clustering

k-means Clustering

LLOYDS($\mathcal{X}$,k) {Input: $\mathcal{X}$, data set; k, number of clusters. Output: $\{\mathbf{m}_i\}_{i=1}^k$, cluster prototypes.}
Initialize $\mathbf{m}_i$, $i = 1, \ldots, k$, appropriately for example, in random.
repeat
  for all $t \in \{1, \ldots, N\}$ **do** {E step}

$$b_i^t \leftarrow \begin{cases} 1 & , \quad i = \arg\min_i ||\mathbf{x}^t - \mathbf{m}_i|| \\ 0 & , \quad \text{otherwise} \end{cases}$$

  end for
  for all $i \in \{1, \ldots, k\}$ **do** {M step}

$$\mathbf{m}_i \leftarrow \frac{\sum_t b_i^t \mathbf{x}^t}{\sum_t b_i^t}$$

  end for
until the error $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X})$ does not change
return $\{\mathbf{m}_i\}_{i=1}^k$

k-means Clustering

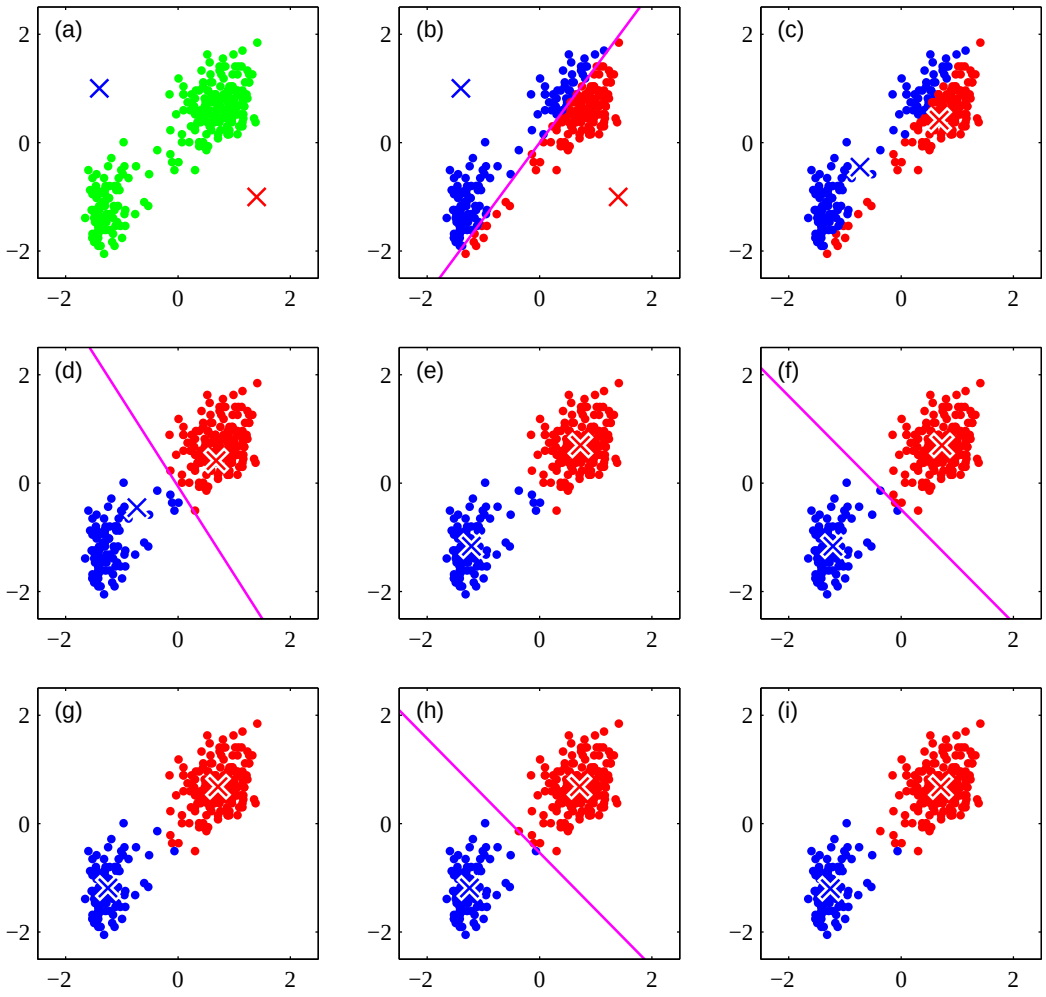


Figure 9.1 of Bishop (2006)

Observations:

- Iteration cannot increase the error $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X})$.
- There are finite number, k^N, of possible clusterings.
- It follows that the algorithm always stops after a finite time. (It can take no more than k^N steps.)
- Usually k-means is however relatively fast. "In practice the number of iterations is generally much less than the number of points." (Duda & Hart & Stork, 2000)
- *Worst-case running time* with really bad data and really bad initialization is however $2^{\Omega(\sqrt{N})}$ — luckily this usually does not happen in real life (David A, Vassilivitskii S (2006) How slow is the k-means method? In Proc 22nd SCG.)

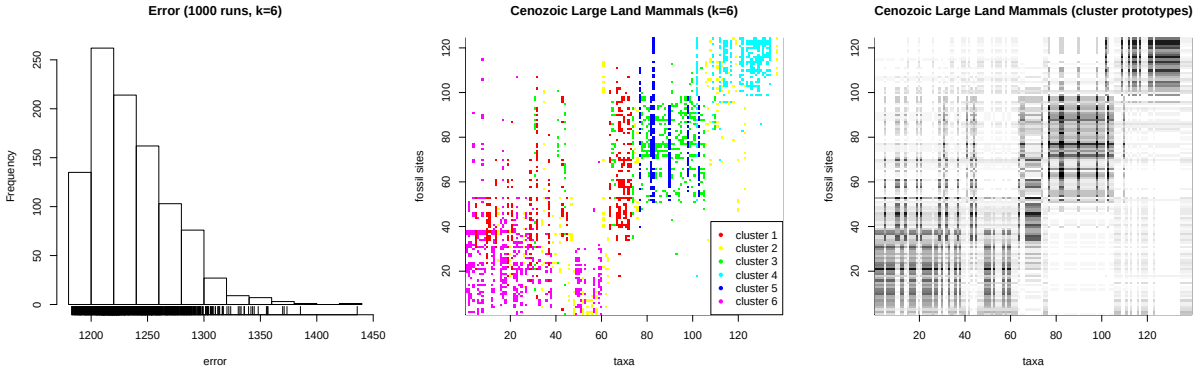
Observations:

- The result can in the worst case be really bad.

- Example:
  - Four data vectors ($N = 4$) from $\mathbb{R}^d$ in $\mathcal{X}$: $\mathbf{x}^1 = (0, 0, \ldots, 0)^T$, $\mathbf{x}^2 = (1, 0, \ldots, 0)^T$, $\mathbf{x}^3 = (0, 1, \ldots, 1)^T$ and $\mathbf{x}^4 = (1, 1, \ldots, 1)^T$.
  - Optimal clustering into two ($k = 2$) is given by the prototype vectors $\mathbf{m}_1 = (0.5, 0, \ldots, 0)^T$ and $\mathbf{m}_2 = (0.5, 1, \ldots, 1)^T$, error being $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X}) = 1$.
  - Lloyd's algorithm can however converge also to $\mathbf{m}_1 = (0, 0.5, \ldots, 0.5)^T$ and $\mathbf{m}_2 = (1, 0.5, \ldots, 0.5)^T$, error being $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X}) = d - 1$. (Check that iteration stops here!)

k-means Clustering

- Example: cluster taxa into $k = 6$ clusters 1000 times with Lloyd's algorithm.
- The error $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X})$ is different for different runs!
- You should try several random initializations, and choose the solution with smallest error.
- For a cool initialization see Arthur D, Vassilivitskii S (2006) k-means++: The Advantages of Careful Seeding.



22.2 Greedy algorithms

Greedy algorithm

- Task: solve $\arg\min_\theta \mathcal{E}(\theta \mid \mathcal{X})$.
- $0 \leq \mathcal{E}(\theta \mid \mathcal{X}) < \infty$
- Assume that the cost/error $\mathcal{E}(\theta \mid \mathcal{X})$ can be evaluated in polynomial time $O(N^k)$, given an instance of parameters θ and a data set $\mathcal{X}$, where N is the size of the data set and k is some constant.
- Often, no polynomial time algorithm to minimize the cost is known.
- Assume that for each instance parameter values θ there exists a *candidate set* $C(\theta)$ such that $\theta \in C(\theta)$.

- Assume $\arg\min_{\theta' \in C(\theta)} \mathcal{E}(\theta' \mid \mathcal{X})$ can be solved in polynomial time.

GREEDY($\mathcal{E}$,C,ϵ,$\mathcal{X}$) {Input: $\mathcal{E}$, cost function; C, candidate set; $\epsilon \geq 0$, convergence cutoff; $\mathcal{X}$, data set. Output: Instance of parameter values θ.}
Initialize θ appropriately, for example, in random.
repeat

$$\theta \leftarrow \arg \min_{\theta' \in C(\theta)} \mathcal{E}(\theta' \mid \mathcal{X})$$

until the change in $\mathcal{E}(\theta \mid \mathcal{X})$ is no more than ϵ
return θ

- Examples of greedy algorithms:
  - Forward and backward selection.
  - Lloyd's algorithm.
  - Optimizing a cost function using gradient descent and line search.
- Each step (except the last) reduces the cost by more than ϵ.
- Each step can be done in polynomial time.
- The algorithm stops after a finite number of steps (at least if $\epsilon > 0$).
- Difficult parts:
  - What is a good initialization?
  - What is a good candidate set $C(\theta)$?
- θ is a *global optimum* if $\theta = \arg\min_\theta \mathcal{E}(\theta \mid \mathcal{X})$.
- θ is a *local optimum* if $\theta = \arg\min_{\theta' \in C(\theta)} \mathcal{E}(\theta' \mid \mathcal{X})$.
- Algorithm always finds a local optimum, but not necessarily a global optimum. (Interesting sidenote: greedoid.)
- Denote $\mathcal{E}^* = \min_\theta \mathcal{E}(\theta \mid \mathcal{X})$, $\theta_{ALG} = \text{GREEDY}(\mathcal{E},C,\epsilon,\mathcal{X})$ and $\mathcal{E}_{ALG} = \mathcal{E}(\theta_{ALG} \mid \mathcal{X})$
- $1 \leq \alpha < \infty$ is an *approximation ratio* if $\mathcal{E}_{ALG} \leq \alpha \mathcal{E}^*$ is always satisfied for all $\mathcal{X}$.
- $1 \leq \alpha < \infty$ is an *expected approximation ratio* if $E[\mathcal{E}_{ALG}] \leq \alpha \mathcal{E}^*$ is always satisfied for all $\mathcal{X}$ (expectation is over instances of the algorithm).
- Observation: if approximation ratio exists, then the algorithm always finds the zero cost solution if such a solution exists for a given data set.
- Sometimes the approximation ratio can be proven; often one can only run algorithm several times and observe the distribution of costs.
- For kmeans with approximation ratio $\alpha = O(\log k)$ and references see Arthur D, Vassilivitskii S (2006) k-means++: The Advantages of Careful Seeding.

- We can usually easily say that the running time of one step is polynomial.
- Often, the number of steps the algorithm takes is also polynomial, hence the algorithm is often polynomial (at least in practice).
- Proving the number of steps required until convergence is often quite difficult, however. Again, the easiest is to run algorithm several times and observe the distribution of the number of steps.
- Does the definition of the cost function make sense in your application? Should you use some other cost, for example, some utility?
- There may be several solutions with small cost. Do these solutions have similar parameters, for example, prototype vectors (interpretation of the results)?
- How efficient is the optimization step involving $C(\theta)$? Could you find better $C(\theta)$?
- If there exists a zero-cost solution, does your algorithm find it?
- Is there an approximation ratio?
- Can you say anything about number of steps required?
- What is the empirical distribution of the error $\mathcal{E}_{ALG}$ and the number of steps taken, in your typical application?

22.3 EM Algorithm

EM Algorithm

- *Expectation-Maximization algorithm* (EM): greedy algorithm that finds soft cluster assignments
- Probabilistic interpretation, that is, we are maximizing a likelihood.

EM Algorithm

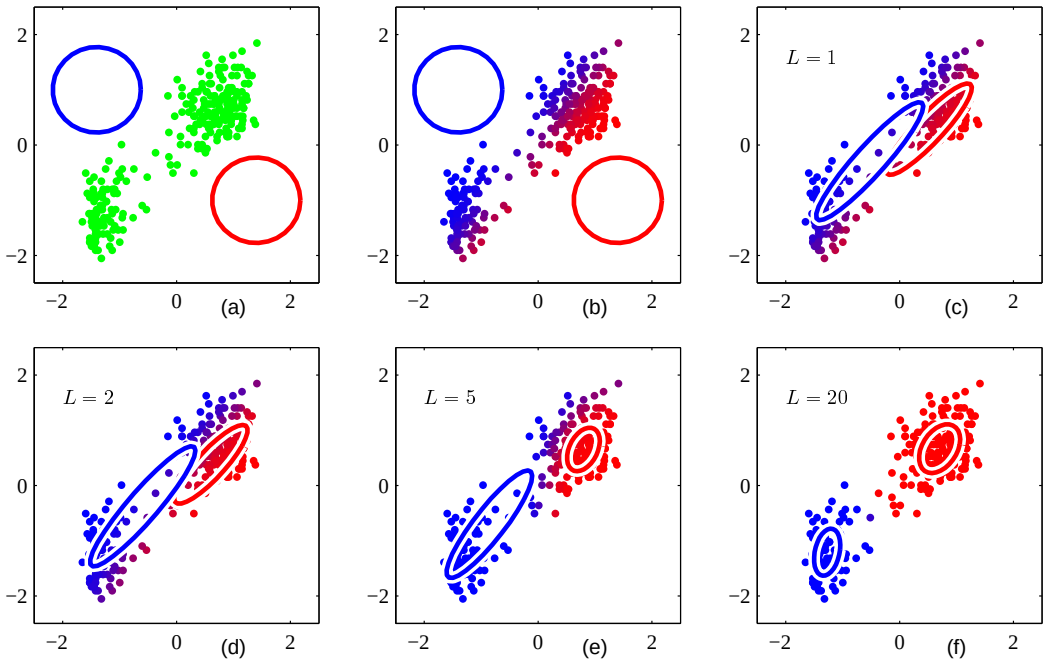
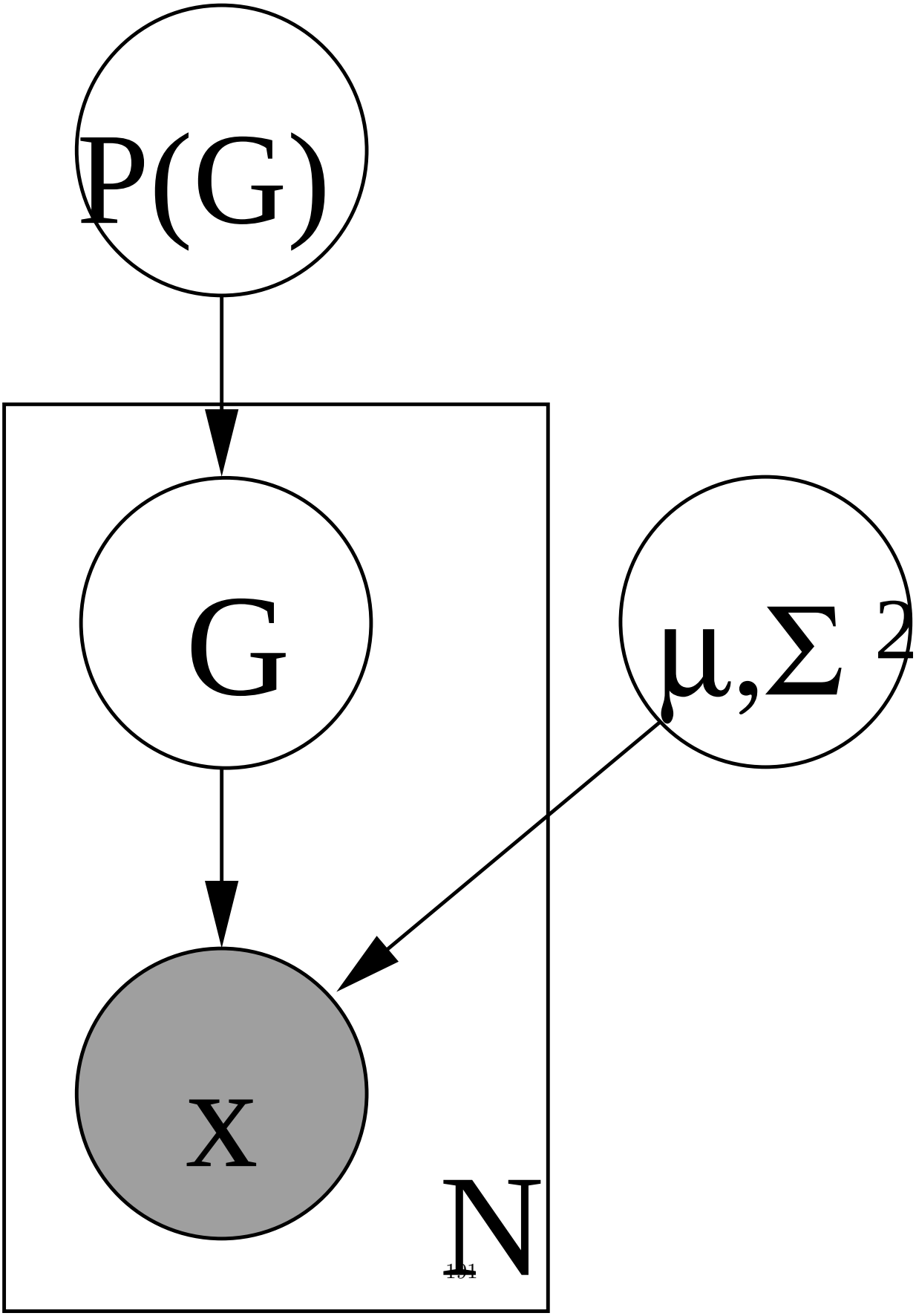


Figure 9.8 of Bishop (2006)

- EM algorithm is like k-means, except cluster assignments are "soft": each data point is a member of a given cluster with certain probability.
- $b_i^t \in \{0, 1\} \longrightarrow h_i^t \in [0, 1]$.

EM Algorithm

- Find maximum likelihood solution of the mixture model $\mathcal{L} = \log \prod_{t=1}^{N} p(\mathbf{x}^t \mid \theta)$, where the parameters θ are μ_i, Σ_i and $\pi_i = P(G_i)$.
- Maximum likelihood solution is found by the EM algorithm (which is essentially generalization of the Lloyd's algorithm to soft cluster memberships)
- Idea: iteratively find the membership weights of each data vector in clusters, and the parameter values. Continue until convergence.
- End result is intuitive.


P(G)
G
μ,Σ 2
X
N

EM Algorithm

EM($\mathcal{X}$,k) {Input: $\mathcal{X}$, data set; k, number of mixture components. Output: $\{\mathbf{m}_i\}_{i=1}^k$, mixture components.}
Initialize $\mathbf{m}_i$, $i = 1, \ldots, k$, for example using some kmeans algorithm.
repeat
  for all $t \in \{1, \ldots, N\}$ **do** {E step}

$$h_i^t \leftarrow \frac{\exp\left[-\frac{1}{2s^2}\left|\left|\mathbf{x}^t - \mathbf{m}_i\right|\right|^2\right]}{\sum_j \exp\left[-\frac{1}{2s^2}\left|\left|\mathbf{x}^t - \mathbf{m}_j\right|\right|^2\right]}$$

  end for
  for all $i \in \{1, \ldots, k\}$ **do** {M step}

$$\mathbf{m}_i \leftarrow \frac{\sum_t h_i^t \mathbf{x}^t}{\sum_t h_i^t}$$

  end for
until convergence
return $\{\mathbf{m}_i\}_{i=1}^k$

EM Algorithm

- For derivation, see Alpaydin (2004), section 7.4 (pages 139–144); for an alternative derivation, see Bishop (2006), section 9.4 (pages 450–455). A sketch follows.
- Task: find an ML solution of a likelihood function given by $p(\mathbf{X} \mid \theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} \mid \theta)$.

$$\begin{aligned}\sum_t \log p(\mathbf{x}^t \mid \theta) &\geq \sum_t \log p(\mathbf{x}^t \mid \theta) - \sum_t KL(h_i^t \mid\mid p(\mathbf{z}^t \mid \mathbf{x}^t, \theta)) \\ &= \sum_t \sum_i h_i^t \log p(\mathbf{x}^t, \mathbf{z}^t \mid \theta) + \sum_t H(h_i^t),\end{aligned}$$

  where we have used the *Kullback-Leibler* (KL) divergence $KL(q(i) \mid\mid p(i)) = \sum_i q(i) \log\left(q(i)/p(i)\right)$. KL divergence is always non-negative and it vanishes only when the distributions q and p are equal. The entropy is given by $H(q(i)) = -\sum_i q(i) \log q(i)$.
- *Expectation step* (E Step): find h_i^t by minimizing the KL divergence.
- *Maximization step* (M Step): find θ by maximizing the expectation.

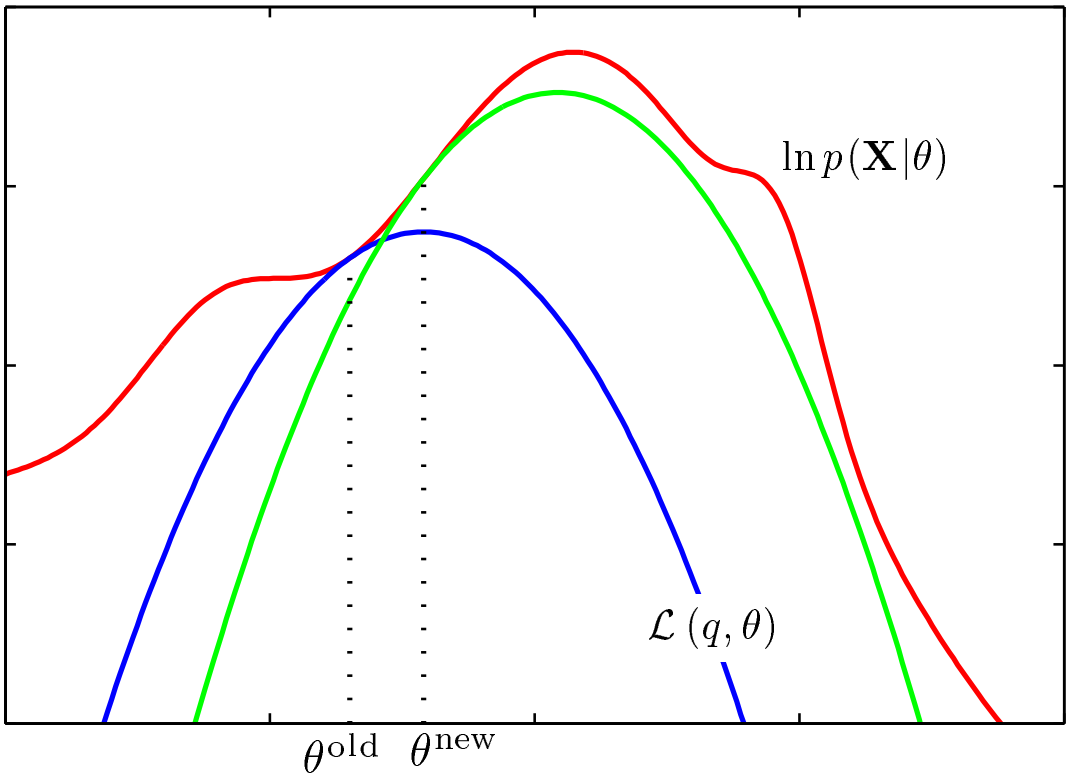


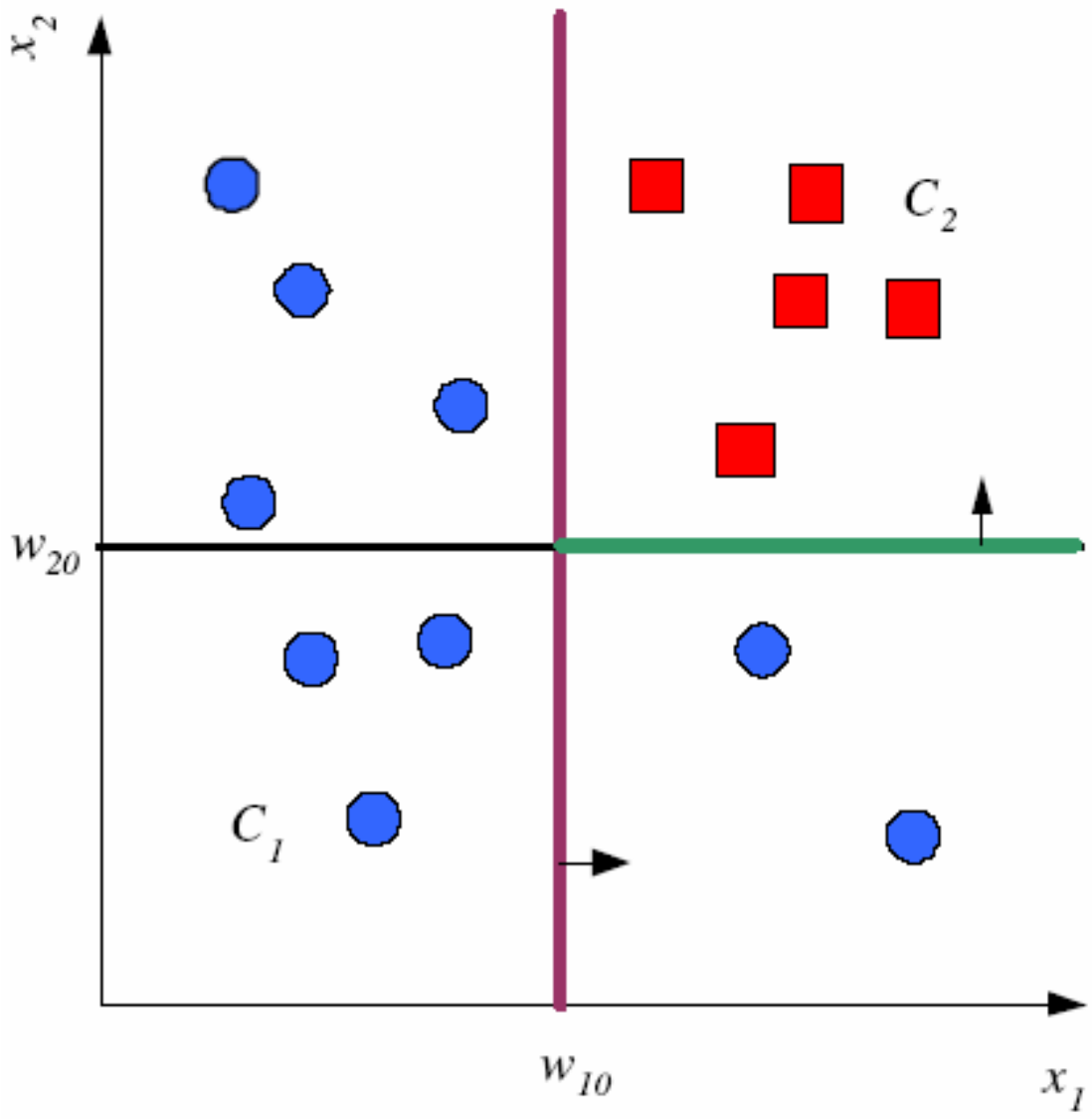
Figure 9.14

of Bishop (2006)

23 Decision Trees

23.1 Introduction

Decision Trees



Lecture Notes for E Alpaydın 2004 Introduction to Machine Learning © The MIT Press

- Each internal node tests an attribute.
- Each branch corresponds to set of attribute values.
- Each leaf node assigns a classification (*classification tree*) or a real number (*regression tree*).
- The tree is usually learned using a greedy algorithm built around $ID3$, such as $C4.5$. (The problem of finding optimal tree is generally NP-hard.)

- Advantages of trees:
  - Learning and classification is fast.
  - Trees are accurate in many domains.
  - Trees are easy to interpret as sets of decision rules.
- Often, trees should be used as a benchmark before more complicated algorithms are attempted.
- For alternative discussion, see Mitchell (1997), Ch 3.

23.2 Classification Trees

Example Data from Mitchell (1997)

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	*No*
D2	Sunny	Hot	High	Strong	*No*
D3	Overcast	Hot	High	Weak	*Yes*
D4	Rain	Mild	High	Weak	*Yes*
D5	Rain	Cool	Normal	Weak	*Yes*
D6	Rain	Cool	Normal	Strong	*No*
D7	Overcast	Cool	Normal	Strong	*Yes*
D8	Sunny	Mild	High	Weak	*No*
D9	Sunny	Cool	Normal	Weak	*Yes*
D10	Rain	Mild	Normal	Weak	*Yes*
D11	Sunny	Mild	Normal	Strong	*Yes*
D12	Overcast	Mild	High	Strong	*Yes*
D13	Overcast	Hot	Normal	Weak	*Yes*
D14	Rain	Mild	High	Strong	*No*

Example: Final Decision Tree

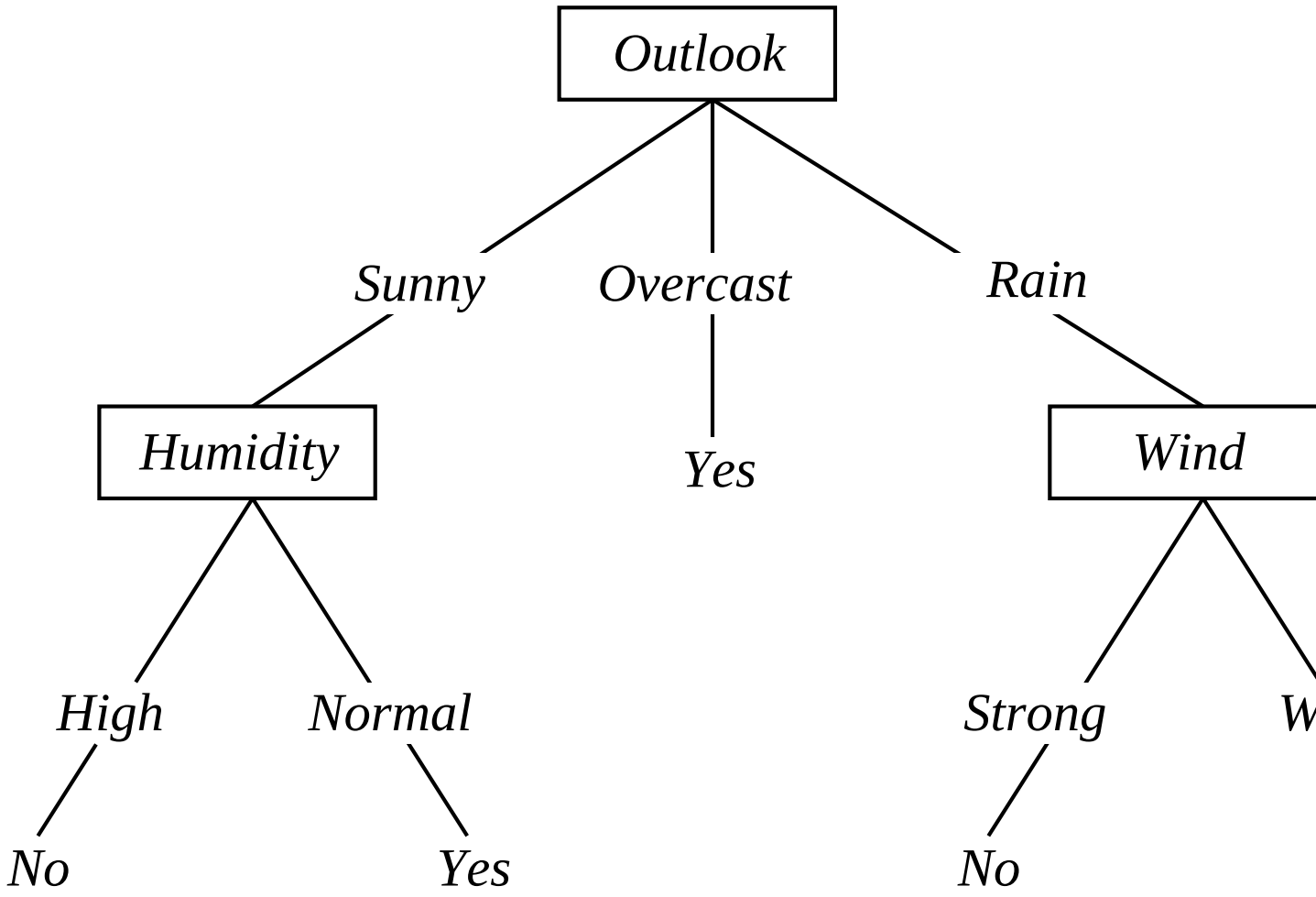


Figure 3.1 of Mitchell (1997).

ID3 algorithm for discrete attributes

ID3($\mathcal{X}$) {Input: $\mathcal{X} = \{(r^t, \mathbf{x}^t)\}_{t=1}^N$, data set with binary attributes $r^t \in \{-1, +1\}$ and a vector of discrete variables $\mathbf{x}^t$. Output: T, classification tree.}
Create *root* node for T
If all items in $\mathcal{X}$ are positive (negative), return a single-node tree with label "+" ("-")
Let A be attribute that "best" classifies the examples
for all values v of A **do**
  Let $\mathcal{X}_v$ be subset of $\mathcal{X}$ that have value v for A
  if $\mathcal{X}_v$ is empty **then**
    Below the root of T, add a leaf node with most common label in $\mathcal{X}$
  else
    Below the root of T, add subtree ID3($\mathcal{X}_v$)
  end if
end for
return T

Entropy

- $\mathcal{X}$ is a sample of training examples.
- p_+ is the proportion of positive and $p_- = 1 - p_+$ is the proportion of negative samples in $\mathcal{X}$.
- Entropy measures *impurity* of $\mathcal{X}$.
- Entropy($\mathcal{X}$) is the expected number of bits needed to encode class ($+1$ or -1) of randomly drawn member of $\mathcal{X}$ (under the optimal, shortest-length code).
- Information theory: the optimal (shortest expected coding length) code for an event with probability p is $-\log_2 p$ bits.
- Therefore, expected number of bits to encode $+1$ or -1 of a random member of $\mathcal{X}$ is

$$p_+ \left(-\log_2 p_+\right) + p_- \left(-\log_2 p_-\right).$$

$$\text{Entropy}(\mathcal{X}) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

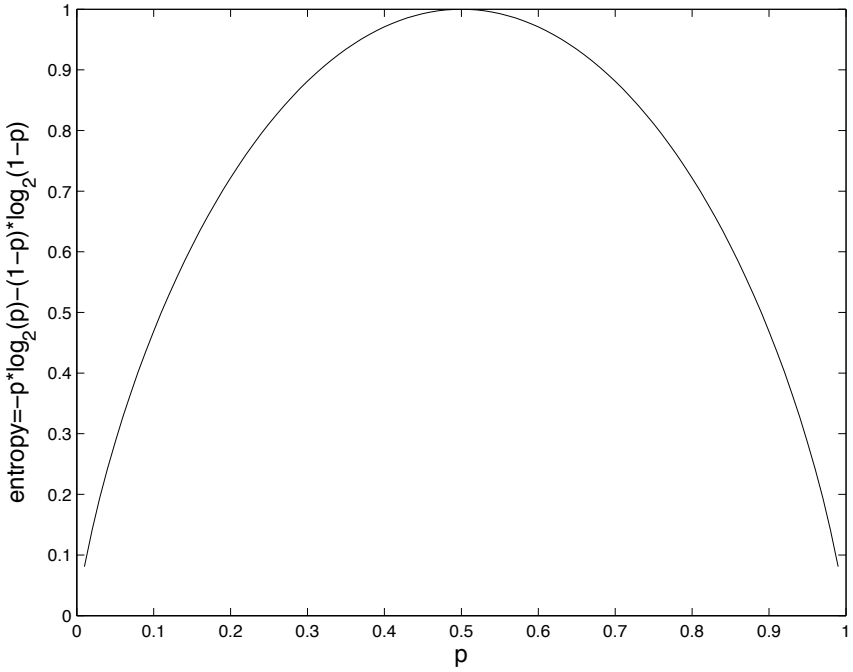


Figure 9.2: Entropy function for a two-class problem. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

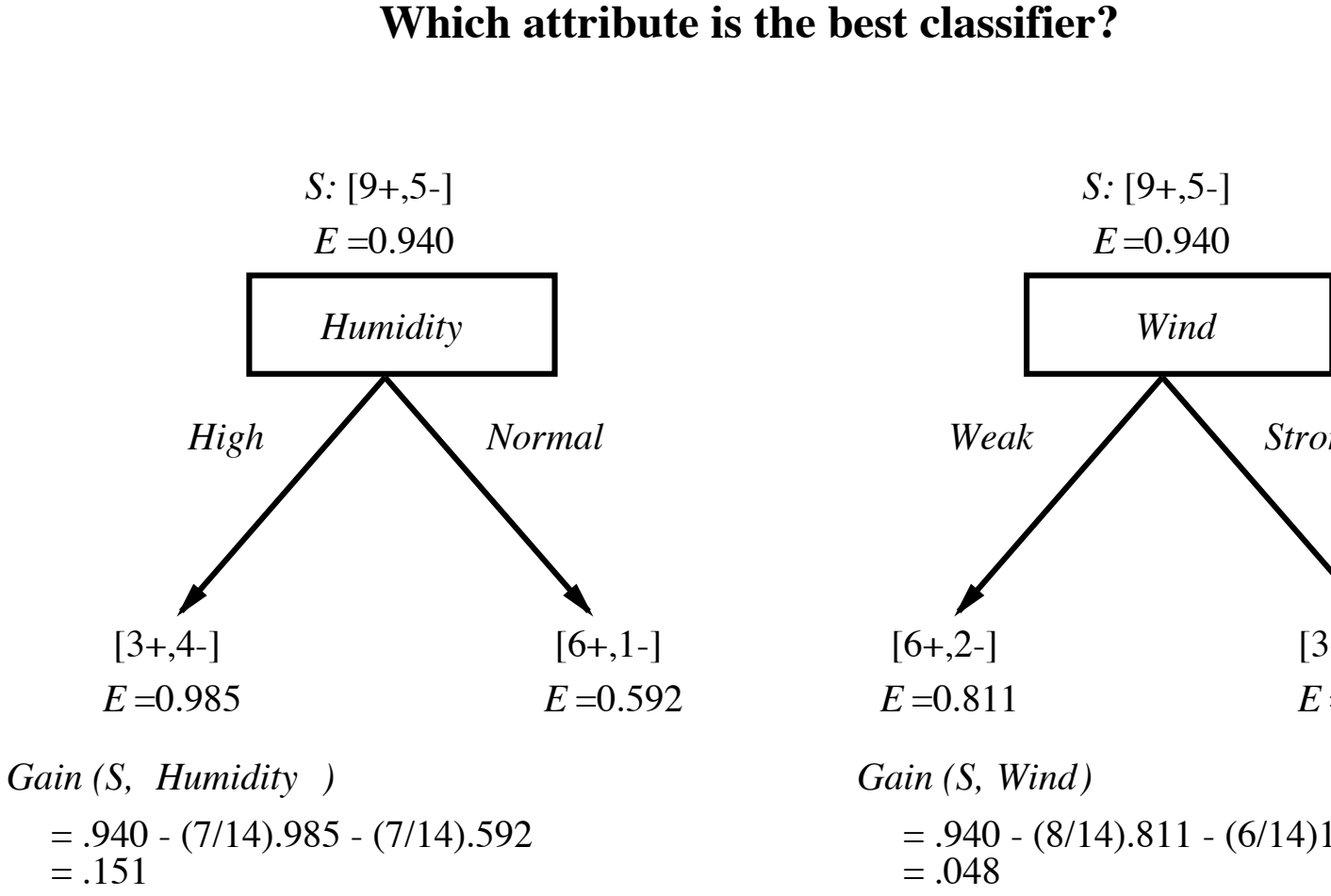
Information Gain

- Gain$(\mathcal{X}, A)$ is the expected reduction in entropy due to sorting on A.

$$\text{Gain}(\mathcal{X}, A) = \text{Entropy}(\mathcal{X}) - \sum_{v \in \text{values}(A)} \frac{|\mathcal{X}_v|}{|\mathcal{X}|} \text{Entropy}(\mathcal{X}_v).$$

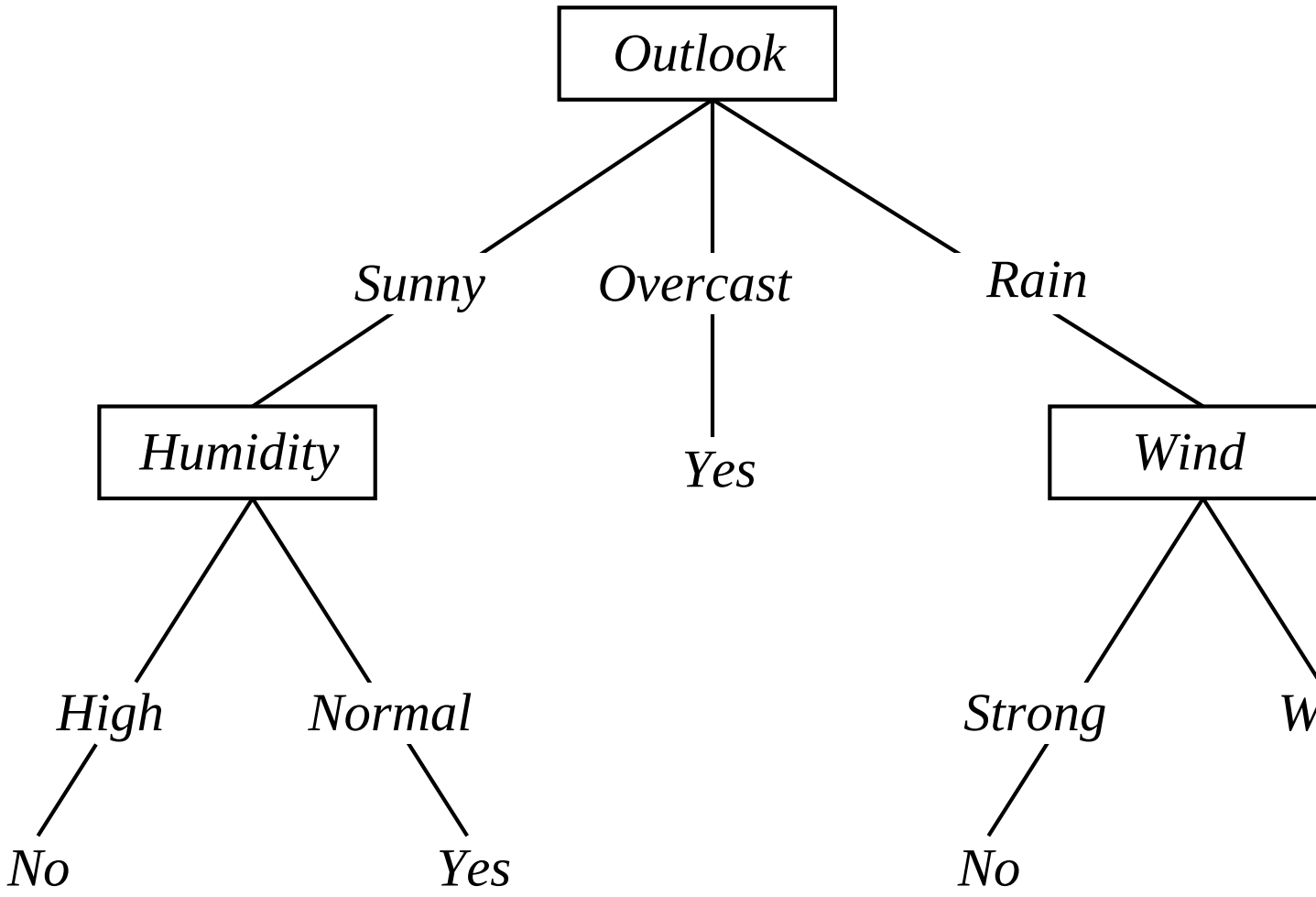
- For ID3: attribute A that has the highest gain classifies the examples $\mathcal{X}$ "best".

Selecting the Next Attribute



Humidity provides greater information gain than *Wind*, relative to the target classification. E stands for entropy and S for collection of examples. Figure 3.3 of Mitchell (1997).

Example: Final Decision Tree



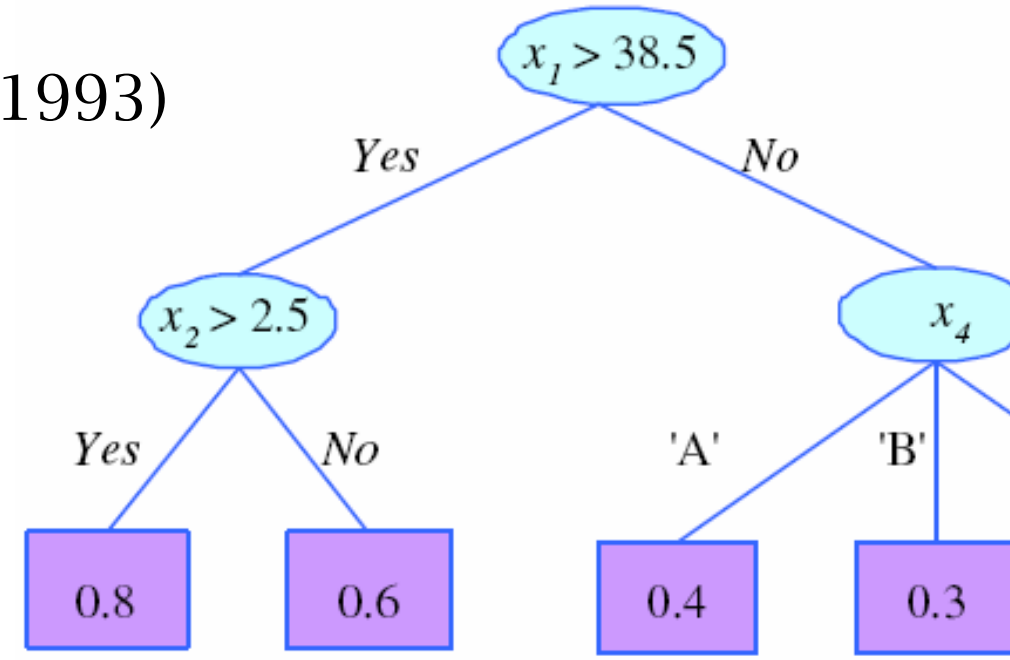
The final decision tree. Figure 3.1 of Mitchell (1997).

Variations of ID3

- Alternative impurity measures:
  - *Entropy*: $-p_+ \log_2 p_+ - p_- \log_2 p_-$.
  - *Gini index*: $2p_+p_-$.
  - *Misclassification error*: $1 - \max(p_+, p_-)$.
  - All vanish for $p_+ \in \{0, 1\}$ and have a maximum at $p_+ = p_- = 1/2$.
- Continuous or ordered variables: sort x_A^t for some attribute A and find the best split $x_A \leq w$ vs. $x_A > w$.

Rule Extraction from Trees

C4.5Rules (Quinlan, 1993)



R1: IF (age>38.5) AND (years-in-job>

R2: IF (age>38.5) AND (years-in-job≤

R3: IF (age≤38.5) AND (job-type='A')

R4: IF (age≤38.5) AND (job-type='B')

R5: IF (age≤38.5) AND (job-type='C')

Observations of ID3

- Inductive bias:
  - Preference on short trees.
  - Preference on trees with high information gain near root.
- Vanilla ID3 classifies the training data perfectly.

- Hence, in presence of noise, vanilla ID3 overfits.

Pruning

- How to avoid overfitting?
  - *Prepruning*: stop growing when data split is not statistically significant. For example: stop tree construction when node is smaller than a given limit, or impurity of a node is below a given limit θ_I. (*faster*)
  - *Postpruning*: grow the whole tree, then prune subtrees which overfit on the pruning (validation) set. (*more accurate*)
- Split data into training and pruning (validation) sets.
- Do until further pruning is harmful:
  1. Evaluate impact on *pruning* set of pruning each possible node (plus those below it).
  2. Greedily remove the one that most improves the *pruning* set accuracy.
- Produces smallest version of most accurate subtree.
- Alternative: rule postpruning (commonly used, for example, $C4.5$).

23.3 Regression Trees

Examples: Predicting woody cover in African savannas

- Task: woody cover (% of surface covered by trees) as a function of precipitation (MAP), soil characteristics (texture, total nitrogen total and phosphorus, and nitrogen mineralization), fire and herbivory regimes.
- Result: MAP is the most important factor.

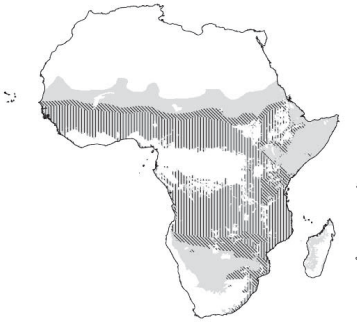


Figure 4 | The distributions of MAP-determined ('stable') and disturbance-determined ('unstable') savannas in Africa. Grey areas represent the existing distribution of savannas in Africa according to ref. 30. Vertically hatched areas show the unstable savannas (>784 mm MAP); cross-hatched areas show the transition between stable and unstable savannas (516–784 mm MAP); grey areas that are not hatched show the stable savannas (<516 mm MAP).

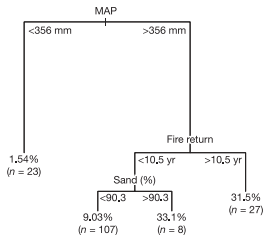


Figure 3 | Regression tree showing generalized relationships between woody cover and MAP, fire-return interval and percentage of sand. The tree is pruned to four terminal nodes and is based on 161 sites for which all data were available. No consistent herbivore effects were detected. Branches are labelled with criteria used to segregate data. Values in terminal nodes represent mean woody cover of sites grouped within the cluster. The pruned tree explained ~45.2% of the variance in woody cover, which is significantly more than a random tree ($P < 0.001$). Of this, 31% was accounted for by the first split; the second split explained an additional 10% of the variance in woody cover.

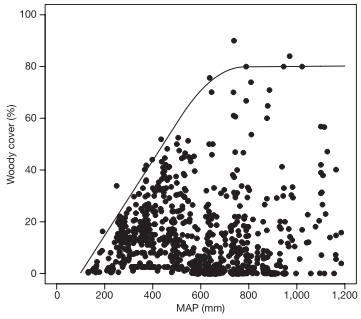


Figure 1 | Change in woody cover of African savannas as a function of MAP. Maximum tree cover is represented by using a 99th quantile piec wise linear regression. The regression analysis identifies the breakpoint (t rainfall at which maximum tree cover is attained) in the interval 650 ± 134 mm MAP (between 516 and 784 mm; see Methods). Trees are typically absent below 101 mm MAP. The equation for the line quantifyi the upper bound on tree cover between 101 and 650 mm MAP is Cover(%) = 0.14(MAP) − 14.2. Data are from 854 sites across Africa.

From Sankaran M et al. (2005) Determinants of woody cover in African savannas. Nature 438: 846–849.

Regression Trees

- Error at node m:

$$b_m(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \text{ reaches node } m \\ 0 & \text{otherwise} \end{cases}$$

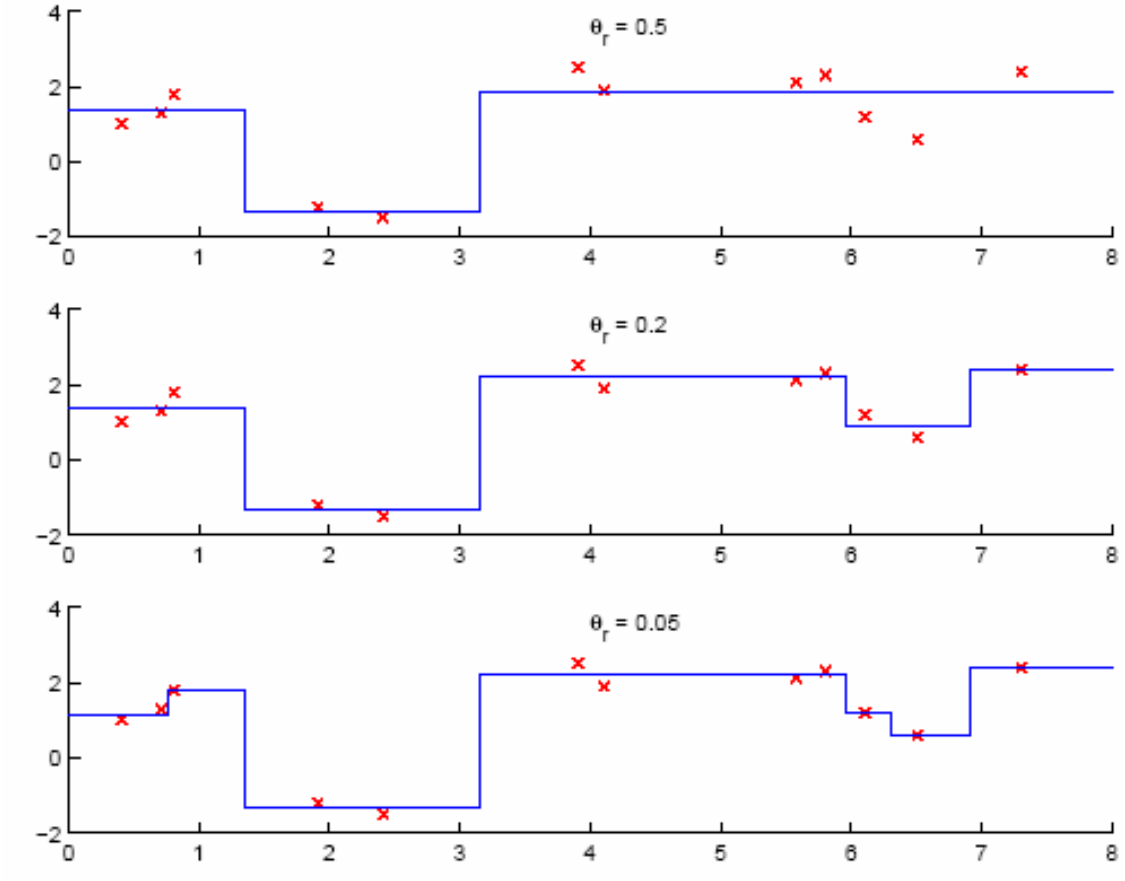
$$\mathcal{E}_m = \frac{1}{N_m} \sum_t \left(r^t - g_m\right)^2 b_m(\mathbf{x}^t) \quad , \quad g_m = \frac{\sum_t b_m(\mathbf{x}^t) r^t}{\sum_t b_m(\mathbf{x}^t)}.$$

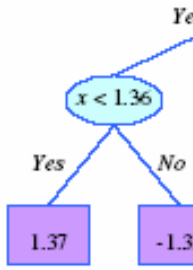
- After splitting:

$$b_{mj}(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \text{ reaches node } m \text{ and branch } j \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{E}_m = \frac{1}{N_m} \sum_j \sum_t \left(r^t - g_{mj}\right)^2 b_{mj}(\mathbf{x}^t) \quad , \quad g_{mj} = \frac{\sum_t b_{mj}(\mathbf{x}^t) r^t}{\sum_t b_{mj}(\mathbf{x}^t)}.$$

Model Selection in Trees:





Lecture Notes for E Alpaydın 2004 Introduction to Machine Learning © The MIT Press (V1.1)

Implementations

- There are many implementations, with sophisticated pruning methods.

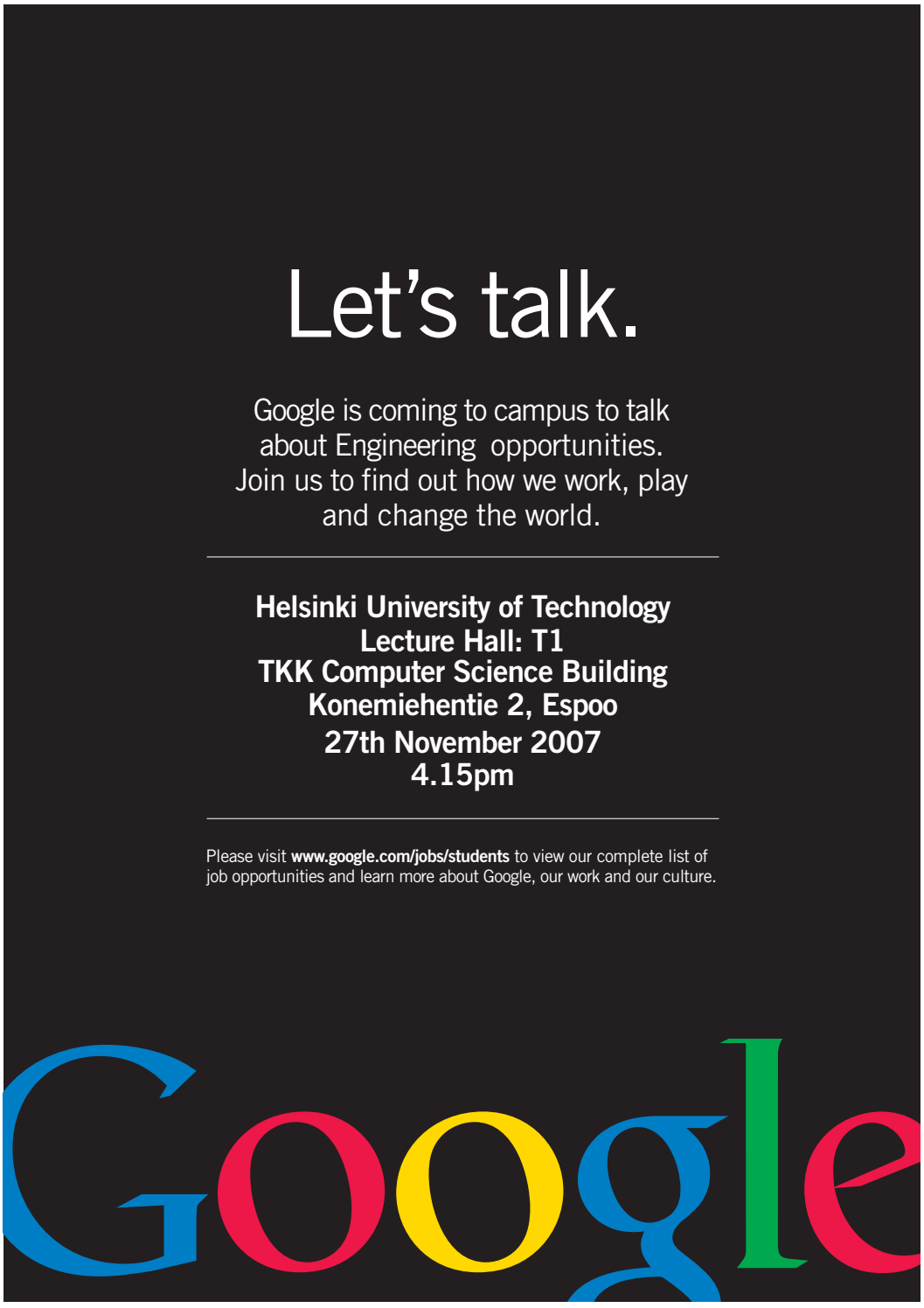
```
> library(rpart)
> rpart(Hipparion ~ .,DD[,taxa])
n= 124

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 124 32 0 (0.74193548 0.25806452)
   2) Amphimachairodus=0 108 19 0 (0.82407407 0.17592593)
     4) Choerolophodon=0 96 13 0 (0.86458333 0.13541667)
       8) Ursus=0 76  6 0 (0.92105263 0.07894737) *
       9) Ursus=1 20  7 0 (0.65000000 0.35000000)
        18) Cervus=1 13  2 0 (0.84615385 0.15384615) *
        19) Cervus=0 7  2 1 (0.28571429 0.71428571) *
     5) Choerolophodon=1 12  6 0 (0.50000000 0.50000000) *
   3) Amphimachairodus=1 16  3 1 (0.18750000 0.81250000) *
```

Machine Learning Guest Lectures on 27 November

10–11 Juha Vesanto (Xtract): Data Mining in Practice How to make succesfull analytics/data mining in an industry/corporate environment. Principles and a case study.

11–12 Hannu Helminen (Google): Machine Learning Methods in Web Search Google is using machine learning methods in the presence of erroneous user queries and documents of low quality. Differences between a traditional information retrieval corpora and the web, and implications of these differences for improving queries and modeling the web are discussed. Inferring meaning from context and using this additional context for query expansion improves the quality of search results.

See `http://www.cis.hut.fi/Opinnot/T-61.3050/2007/guestlecture`



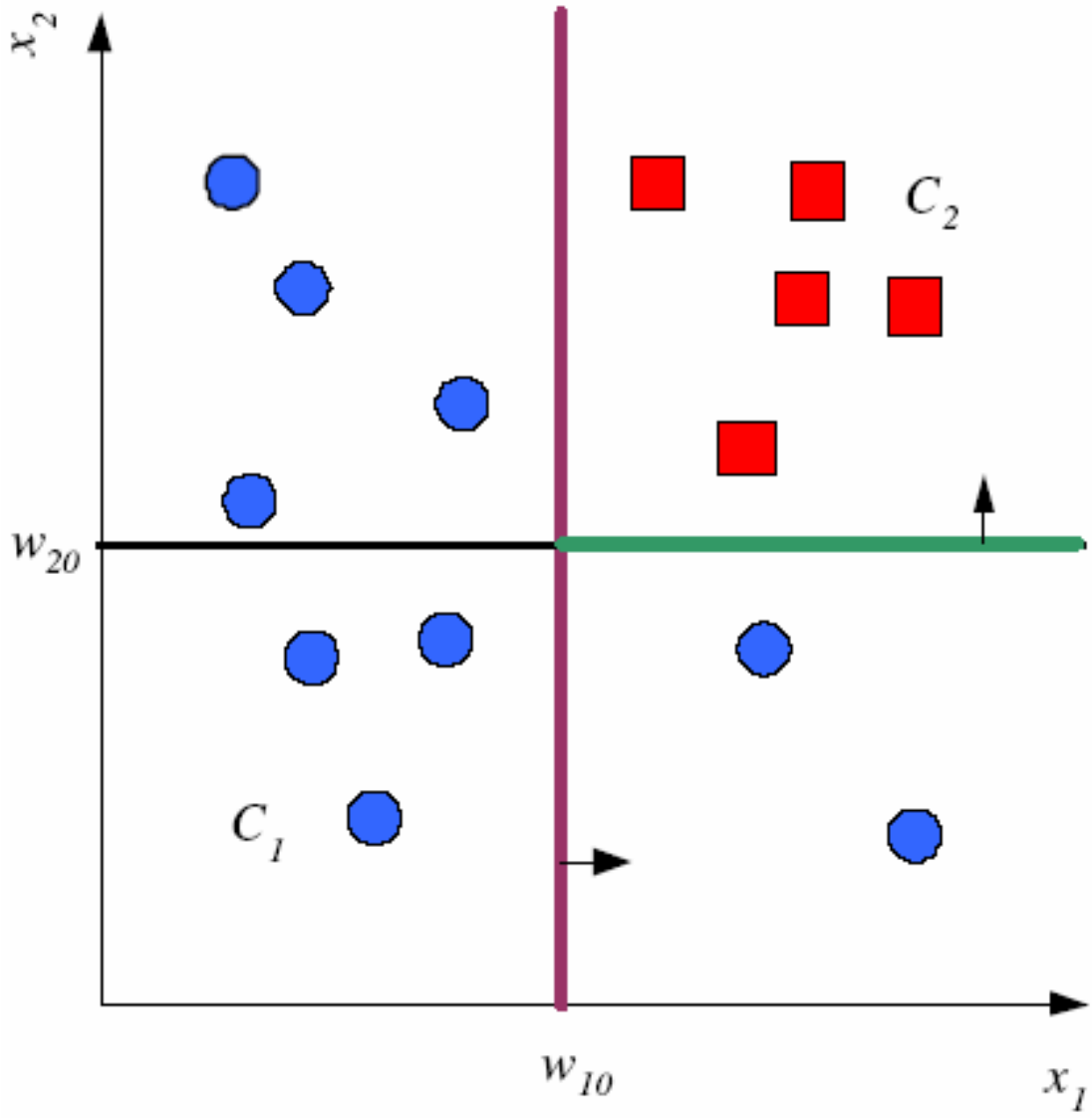
See

`http://www.cis.hut.fi/googletalk07/`

24 Decision Trees

24.1 Classification Trees

Decision Trees



Lecture Notes for E Alpaydın 2004 Introduction to Machine Learning © The MIT Press

- Each internal node tests an attribute.
- Each branch corresponds to set of attribute values.

- Each leaf node assigns a classification (*classification tree*) or a real number (*regression tree*).
- The tree is usually learned using a greedy algorithm built around $ID3$, such as $C4.5$. (The problem of finding optimal tree is generally NP-hard.)
- Advantages of trees:
  - Learning and classification is fast.
  - Trees are accurate in many domains.
  - Trees are easy to interpret as sets of decision rules.
- Often, trees should be used as a benchmark before more complicated algorithms are attempted.
- For alternative discussion, see Mitchell (1997), Ch 3.

ID3 algorithm for discrete attributes

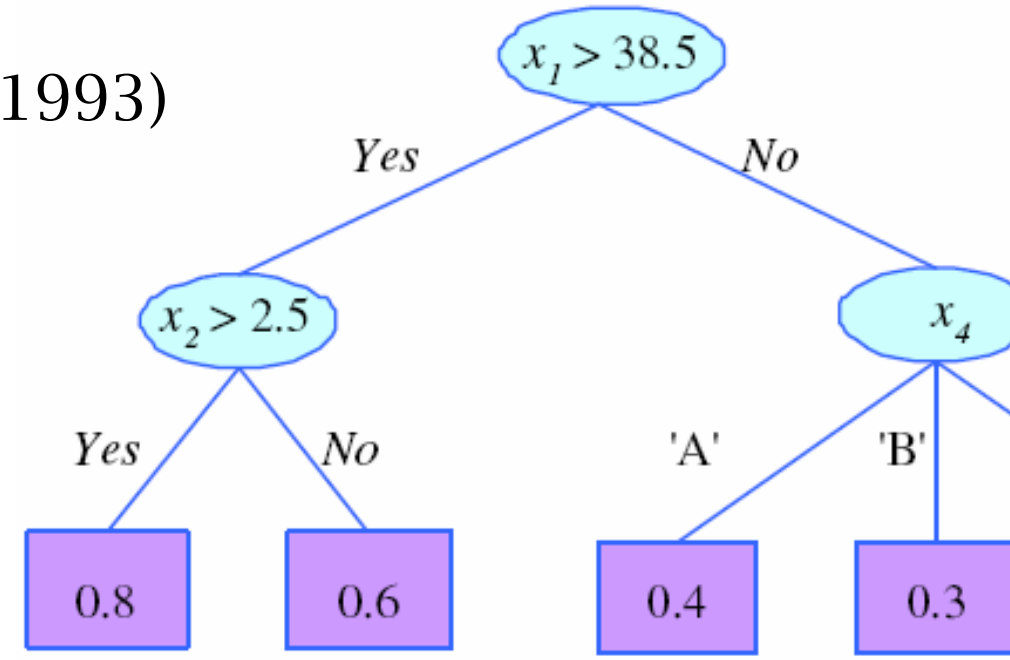
ID3($\mathcal{X}$) {Input: $\mathcal{X} = \{(r^t, \mathbf{x}^t)\}_{t=1}^N$, data set with binary attributes $r^t \in \{-1, +1\}$ and a vector of discrete variables $\mathbf{x}^t$. Output: T, classification tree.}
Create *root* node for T
If all items in $\mathcal{X}$ are positive (negative), return a single-node tree with label "+" ("-")
Let A be attribute that "best" classifies the examples
for all values v of A **do**
  Let $\mathcal{X}_v$ be subset of $\mathcal{X}$ that have value v for A
  if $\mathcal{X}_v$ is empty **then**
    Below the root of T, add a leaf node with most common label in $\mathcal{X}$
  else
    Below the root of T, add subtree ID3($\mathcal{X}_v$)
  end if
end for
return T

Variations of ID3

- Impurity measures:
  - *Entropy*: $-p_+ \log_2 p_+ - p_- \log_2 p_-$.
  - *Gini index*: $2p_+p_-$.
  - *Misclassification error*: $1 - \max(p_+, p_-)$.
  - All vanish for $p_+ \in \{0, 1\}$ and have a maximum at $p_+ = p_- = 1/2$.
- Continuous or ordered variables: sort x_A^t for some attribute A and find the best split $x_A \leq w$ vs. $x_A > w$.

Rule Extraction from Trees

C4.5Rules (Quinlan, 1993)



R1: IF (age>38.5) AND (years-in-job>

R2: IF (age>38.5) AND (years-in-job≤

R3: IF (age≤38.5) AND (job-type='A')

R4: IF (age≤38.5) AND (job-type='B')

R5: IF (age≤38.5) AND (job-type='C')

Observations of ID3

- Inductive bias:
  - Preference on short trees.
  - Preference on trees with high information gain near root.
- Vanilla ID3 classifies the training data perfectly.

- Hence, in presence of noise, vanilla ID3 overfits.

Pruning

- How to avoid overfitting?
  - *Prepruning*: stop growing when data split is not statistically significant. For example: stop tree construction when node is smaller than a given limit, or impurity of a node is below a given limit θ_I. (*faster*)
  - *Postpruning*: grow the whole tree, then prune subtrees which overfit on the pruning (validation) set. (*more accurate*)
- Split data into training and pruning (validation) sets.
- Do until further pruning is harmful:
  1. Evaluate impact on *pruning* set of pruning each possible node (plus those below it).
  2. Greedily remove the one that most improves the *pruning* set accuracy.
- Produces smallest version of most accurate subtree.
- Alternative: rule postpruning (commonly used, for example, $C4.5$).

24.2 Regression Trees

Examples: Predicting woody cover in African savannas

- Task: woody cover (% of surface covered by trees) as a function of precipitation (MAP), soil characteristics (texture, total nitrogen total and phosphorus, and nitrogen mineralization), fire and herbivory regimes.
- Result: MAP is the most important factor.

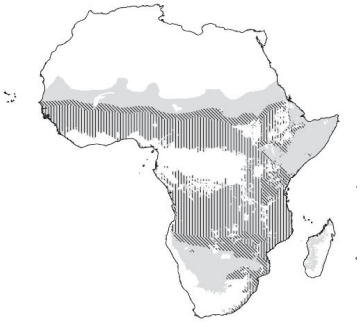


Figure 4 | The distributions of MAP-determined ('stable') and disturbance-determined ('unstable') savannas in Africa. Grey areas represent the existing distribution of savannas in Africa according to ref. 30. Vertically hatched areas show the unstable savannas (>784 mm MAP); cross-hatched areas show the transition between stable and unstable savannas (516–784 mm MAP); grey areas that are not hatched show the stable savannas (<516 mm MAP).

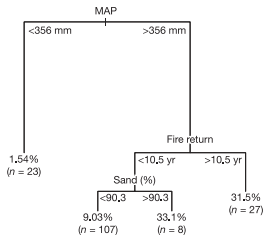


Figure 3 | Regression tree showing generalized relationships between woody cover and MAP, fire-return interval and percentage of sand. The tree is pruned to four terminal nodes and is based on 161 sites for which all data were available. No consistent herbivore effects were detected. Branches are labelled with criteria used to segregate data. Values in terminal nodes represent mean woody cover of sites grouped within the cluster. The pruned tree explained ~45.2% of the variance in woody cover, which is significantly more than a random tree ($P < 0.001$). Of this, 31% was accounted for by the first split; the second split explained an additional 10% of the variance in woody cover.

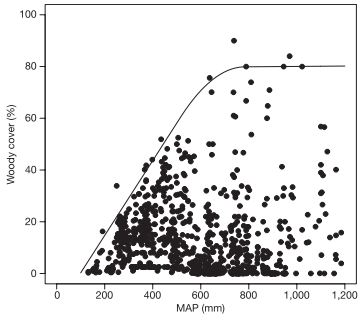


Figure 1 | Change in woody cover of African savannas as a function of MAP. Maximum tree cover is represented by using a 99th quantile piec wise linear regression. The regression analysis identifies the breakpoint (t rainfall at which maximum tree cover is attained) in the interval 650 ± 134 mm MAP (between 516 and 784 mm; see Methods). Trees are typically absent below 101 mm MAP. The equation for the line quantifyi the upper bound on tree cover between 101 and 650 mm MAP is Cover(%) = 0.14(MAP) − 14.2. Data are from 854 sites across Africa.

From Sankaran M et al. (2005) Determinants of woody cover in African savannas. Nature 438: 846–849.

Regression Trees

- Error at node m:

$$b_m(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \text{ reaches node } m \\ 0 & \text{otherwise} \end{cases}$$

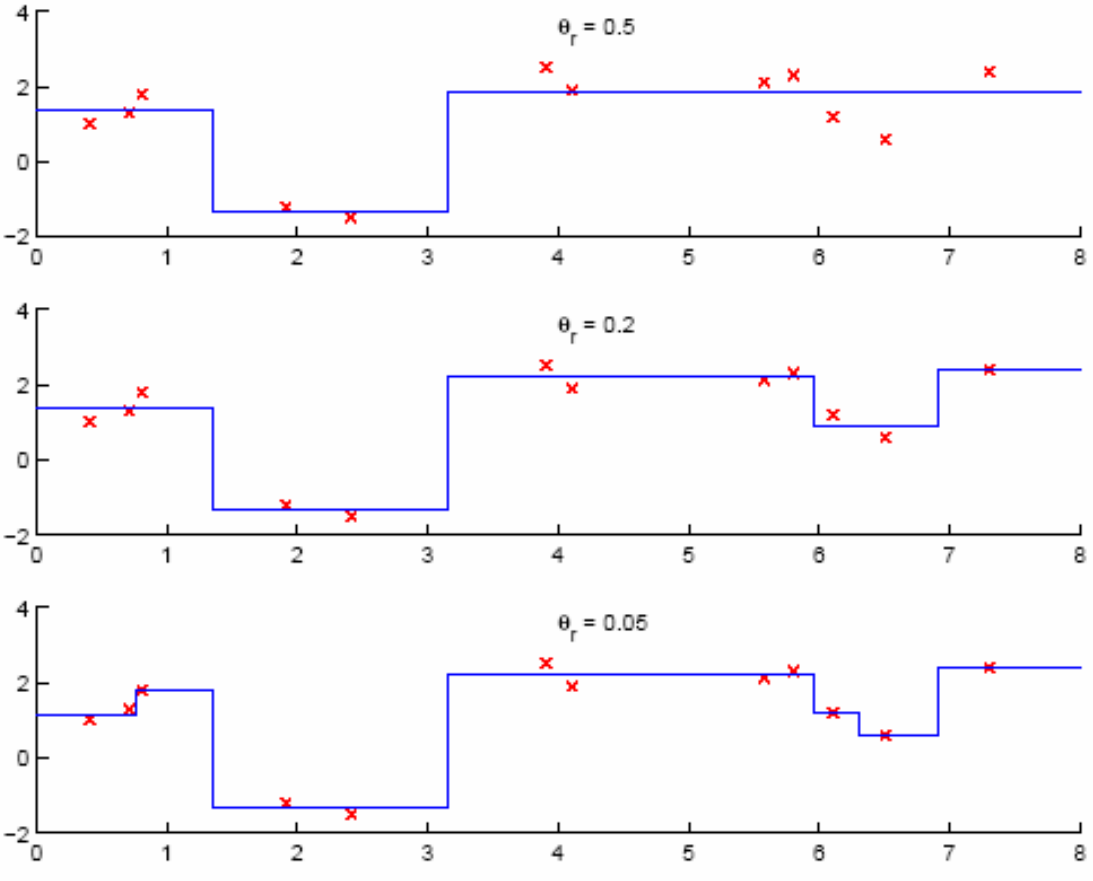
$$\mathcal{E}_m = \frac{1}{N_m} \sum_t \left(r^t - g_m\right)^2 b_m(\mathbf{x}^t) \quad , \quad g_m = \frac{\sum_t b_m(\mathbf{x}^t) r^t}{\sum_t b_m(\mathbf{x}^t)}.$$

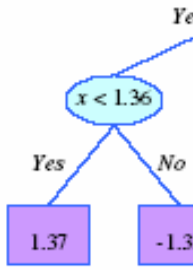
- After splitting:

$$b_{mj}(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \text{ reaches node } m \text{ and branch } j \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{E}_m = \frac{1}{N_m} \sum_j \sum_t \left(r^t - g_{mj}\right)^2 b_{mj}(\mathbf{x}^t) \quad , \quad g_{mj} = \frac{\sum_t b_{mj}(\mathbf{x}^t) r^t}{\sum_t b_{mj}(\mathbf{x}^t)}.$$

Model Selection in Trees:





Implementations

- There are many implementations, with sophisticated pruning methods.

```
> library(rpart)
> rpart(Hipparion ~ .,DD[,taxa])
n= 124

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 124 32 0 (0.74193548 0.25806452)
   2) Amphimachairodus=0 108 19 0 (0.82407407 0.17592593)
     4) Choerolophodon=0 96 13 0 (0.86458333 0.13541667)
       8) Ursus=0 76  6 0 (0.92105263 0.07894737) *
       9) Ursus=1 20  7 0 (0.65000000 0.35000000)
        18) Cervus=1 13  2 0 (0.84615385 0.15384615) *
        19) Cervus=0 7  2 1 (0.28571429 0.71428571) *
     5) Choerolophodon=1 12  6 0 (0.50000000 0.50000000) *
   3) Amphimachairodus=1 16  3 1 (0.18750000 0.81250000) *
```

25 Linear Discrimination

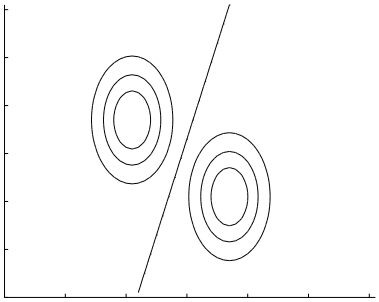
25.1 Naive Bayes Classifier (Again)

Linear Discrimination

- Source material:
  - Alpaydin (2004) Ch 10, or
  - A new chapter by Mitchell (September 2005), "Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression", available as PDF at `http://www.cs.cmu.edu/~tom/NewChapters.html`

Naive Bayes Classifier

- Idea: the means are class-specific, covariance matrix Σ is common and diagonal (*Naive Bayes*).
- d parameters in the covariance matrix.
- Discriminant is linear: $g_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + w_{i0}$, where $\mathbf{w}_i = \Sigma^{-1}\mu_i$ and $w_{i0} = -\frac{1}{2}\mu_i^T\Sigma^{-1}\mu_i + \log P(C_i)$.



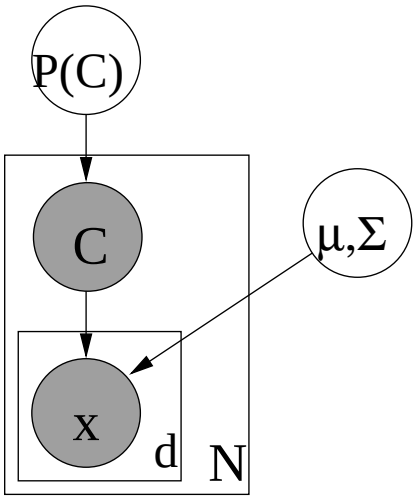
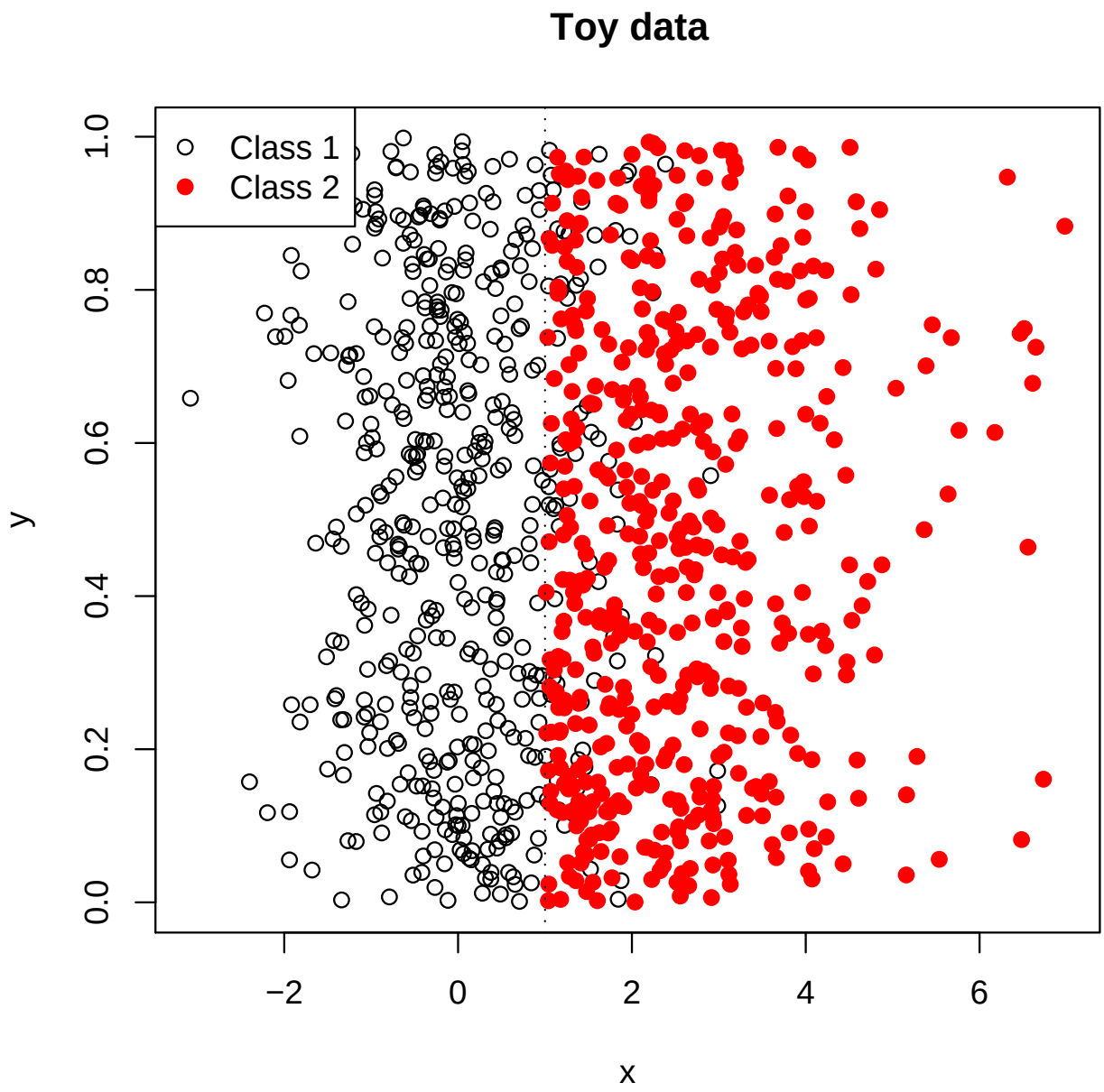
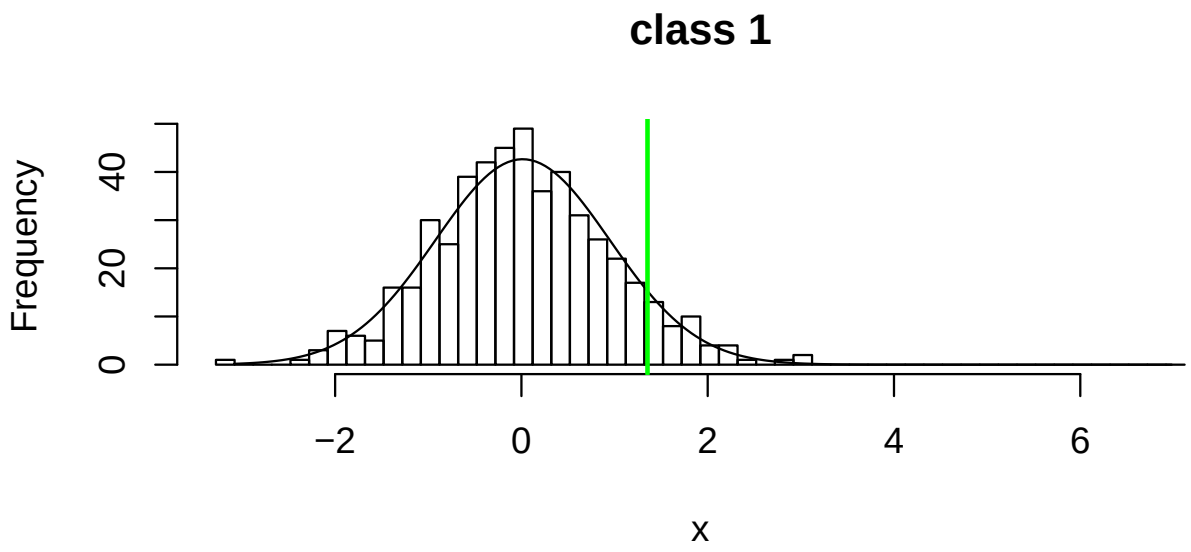
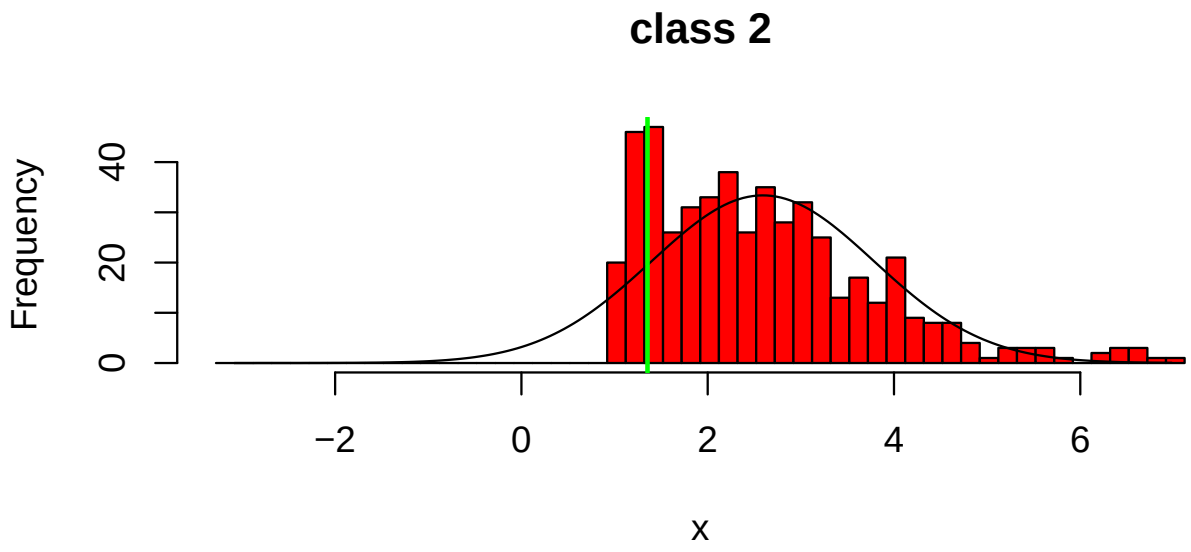


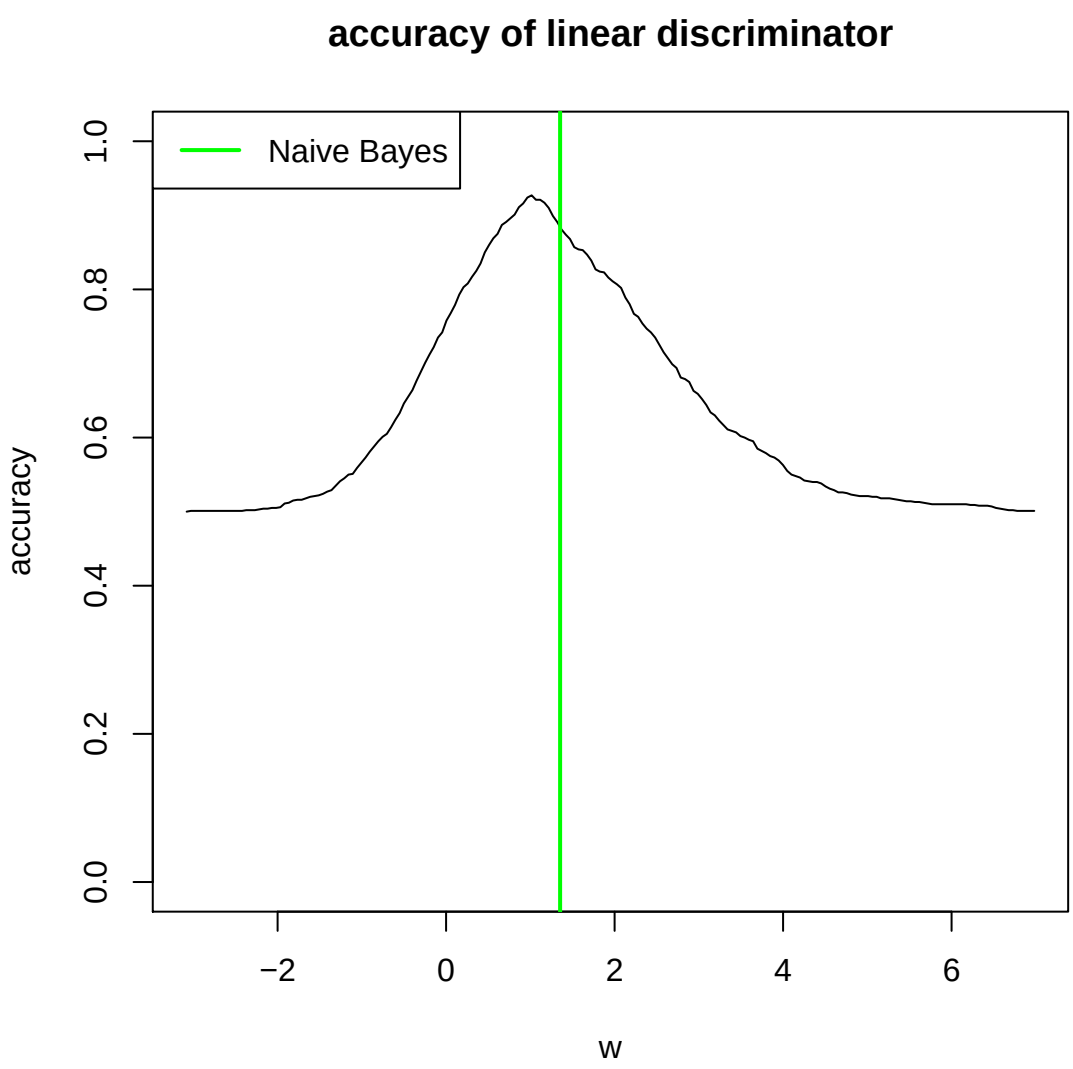
Figure 5.5: All classes have equal, diagonal covariance matrices but variances are not equal. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Naive Bayes Classifier


Toy data
Class 1
Class 2
y
x
0.0
0.2
0.4
0.6
0.8
1.0
−2
0
2
4
6


class 1
Frequency
0
20
40
−2
0
2
4
6
x


class 2
Frequency
0
20
40
−2
0
2
4
6
x



Accuracy of discriminator "class 1 if $x < w$, class 2 if $x \geq w$".

Naive Bayes Classifier

- $\mathcal{X} = \{(r^t, \mathbf{x}^t)\}_{t=1}^N$, $r^t \in \{0,1\}$, $\mathbf{x}^t \in \mathbb{R}^d$.
- Naive Bayes assumption: $P(\mathbf{x}^t \mid r^t) = \prod_{i=1}^d P(x_i^t \mid r^t)$.
- Using Bayes rule,

$$P(r \mid \mathbf{x}) = \frac{P(r)\prod_{i=1}^d P(x_i \mid r)}{\sum_{s\in\{0,1\}} P(s)\prod_{i=1}^d P(x_i \mid s)}.$$

- Discriminant is linear: $g_i(\mathbf{x}) = \log P(r_i = 1 \mid \mathbf{x}) + \text{const.} = \mathbf{w}_i^T\mathbf{x} + w_{i0}$, where $\mathbf{w}_i = \Sigma^{-1}\mu_i$ and $w_{i0} = -\frac{1}{2}\mu_i^T\Sigma^{-1}\mu_i + \log P(C_i)$.

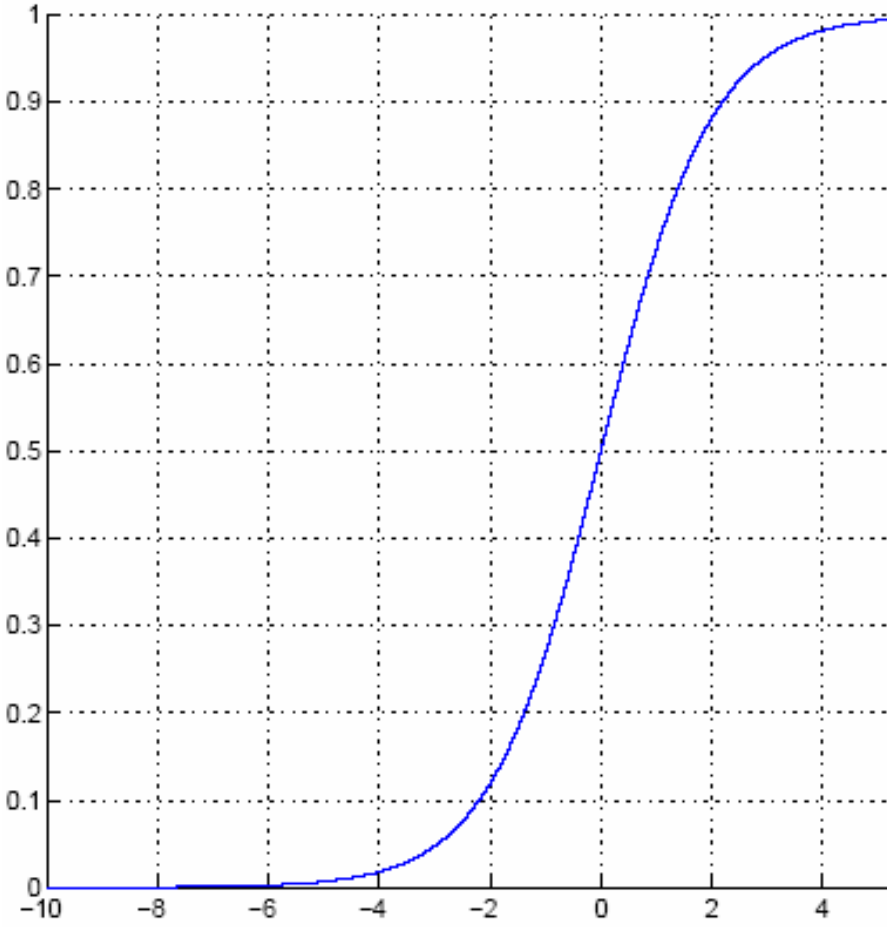
- Observation:
$$\log \frac{P(r=1 \mid \mathbf{x})}{1-P(r=1 \mid \mathbf{x})} = \mathbf{w}^T\mathbf{x} + \mathbf{w}_0.$$

25.2 Logistic Regression

Logistic Regression

- *Logit*: $\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$.
- *Sigmoid*: $\text{sigmoid}(t) = \text{logit}^{-1}(t) = 1/(1+e^{-t})$.
- Derivative of sigmoid: $\text{sigmoid}'(t) = \text{sigmoid}(t)\,(1-\text{sigmoid}(t))$.

Sigmoid (Logistic) Function



1. Calculate $g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$ and choose
2. Calculate $y = \text{sigmoid}(\boldsymbol{w}^T \boldsymbol{x} + w_0)$ and

Cost Function for Logistic Regression

- $$P(R \mid X, W) = \prod_{t=1}^{n} P(r^t \mid \mathbf{x}^t, W)$$

- $$\mathcal{L} = e^{-P(R|X,W)} = -\sum_{t=1}^{N} \left(r^t \log y^t - (1-r^t)\log\left(1-y^t\right)\right),$$

  where $y^t = P(r^t = 1 \mid \mathbf{x}) = \text{sigmoid}(\mathbf{w}^t\mathbf{x} + w_0)$.

- Task: find $W = (\mathbf{w}, w_0)$ such that $\mathcal{L}$ is minimized.

- No EM algorithm. Use gradient ascent.

Gradient Ascent

GRADASC($\mathcal{L}(\theta)$, θ^0) {Input: $\mathcal{L}(\theta)$, cost function; θ^0, initial parameters. Output: θ, a local minimum of $\mathcal{L}$.}
$\theta \leftarrow \theta^0$ $\{\theta, \theta^0 \in \mathbb{R}^d.\}$
$t \leftarrow 1$
repeat
  for all $i \in \{1, \ldots, d\}$ **do**
    $\Delta\theta_i \leftarrow \partial\mathcal{L}(\theta)/\partial\theta_i$
  end for
  for all $i \in \{1, \ldots, d\}$ **do**
    $\theta_i \leftarrow \theta_i - \eta_t \Delta\theta_i$
  end for
  $t \leftarrow t + 1$
until convergence
return θ

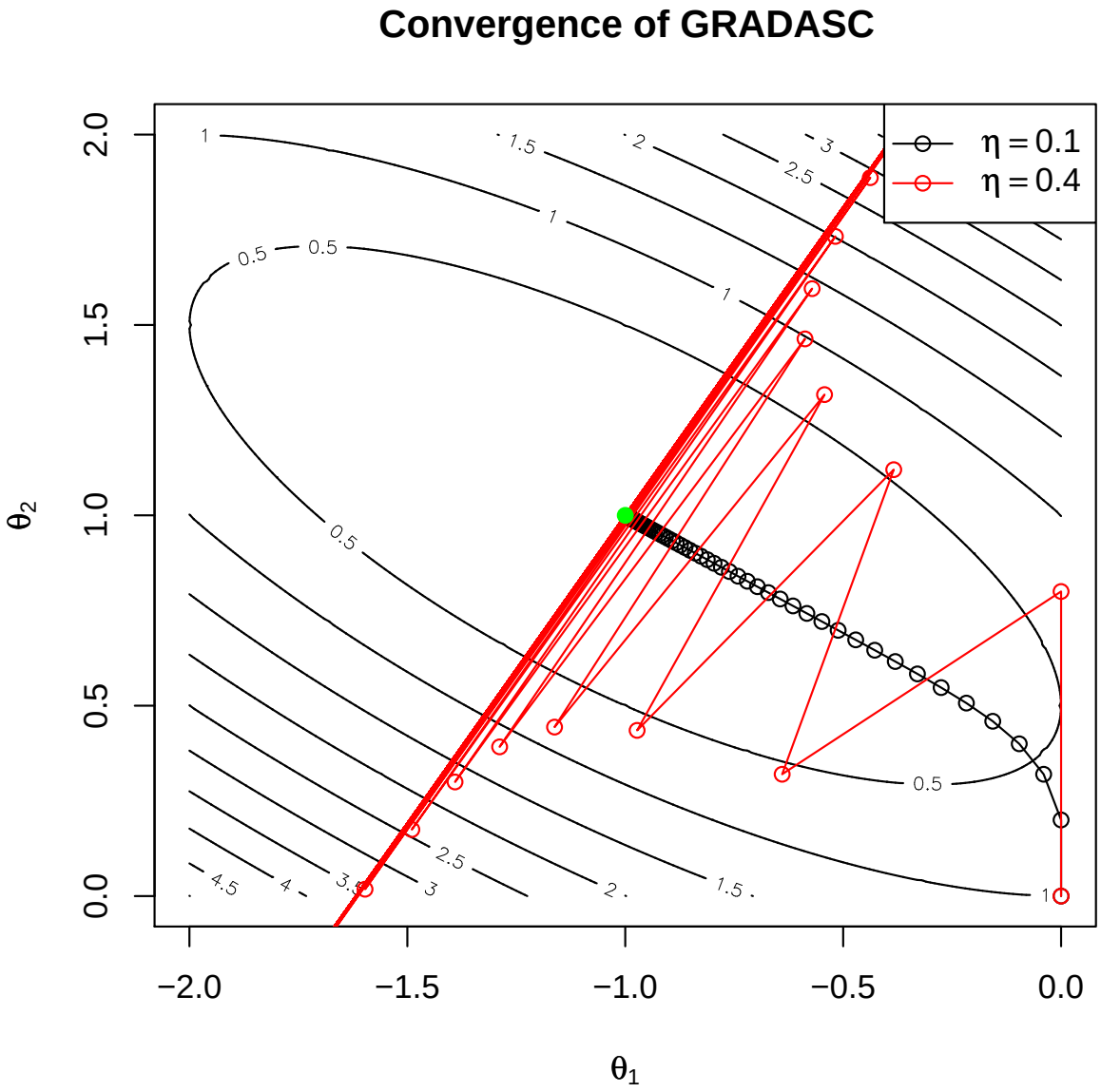
Gradient Ascent

- The function GRADASC always converges if $\sum_{t=1}^{\infty} \eta_t = \infty$ and $\sum_{t=1}^{\infty} \eta_t^2 < \infty$, where $\eta_t \geq 0$ for all t, for example, $\eta_t = 1/t$.

- The function GRADASC often converges also for constant small enough $\eta_t = \eta > 0$.

- GRADASC is inefficient.

- Usually one should use a more sophisticated gradient ascent algorithm, such as conjugate gradient, from some numerical library (e.g., in R type `help(optim)`).



Minimizing $\mathcal{L}(\theta) = (\theta_1 + \theta_2)^2 + (\theta_2 - 1)^2$, using $\theta^0 = (0, 0)^T$.

Gradient Ascent

- Logistic regression may converge to $w \to \pm\infty$ (see right), especially when data is high dimensional and sparse. This causes problems.
- Solution: minimize regularized cost $\mathcal{L} \to \mathcal{L} + \frac{1}{2}\lambda \left(w_0^2 + \mathbf{w}^T\mathbf{w}\right)$.

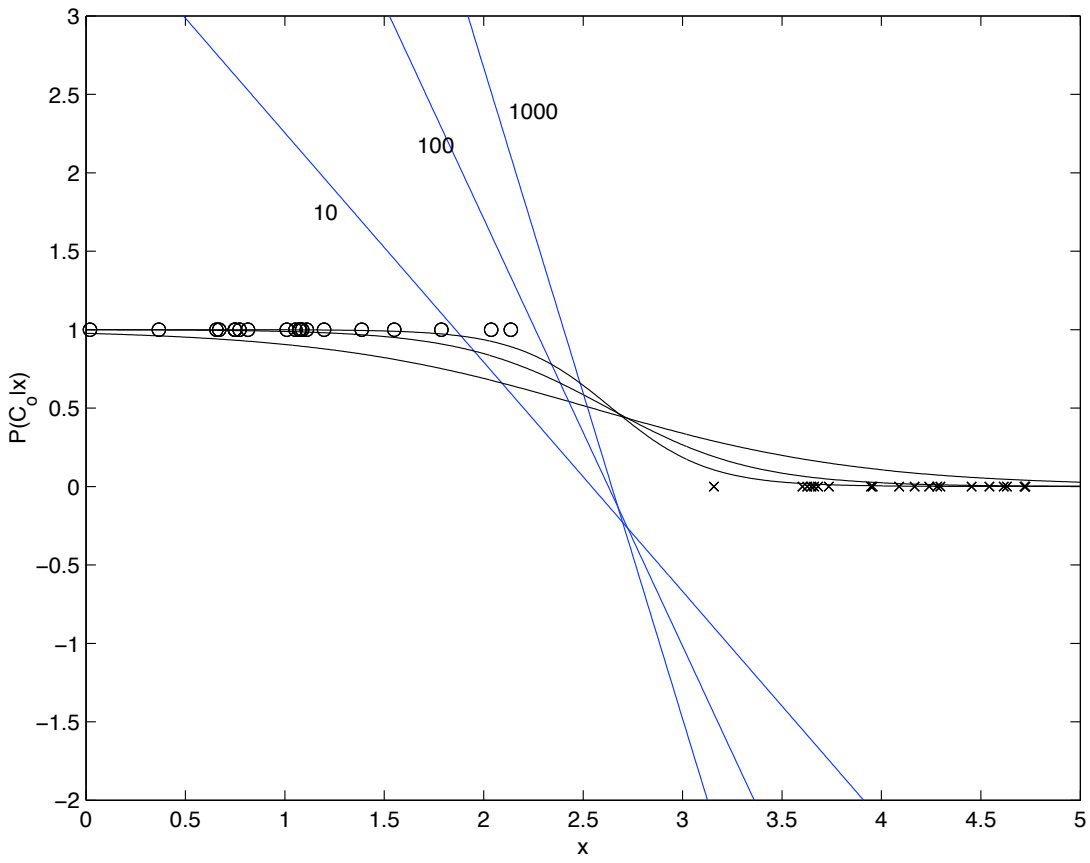


Figure 10.7: For a univariate two-class problem (shown with '$\circ$' and '$\times$'), the evolution of the line $wx + w_0$ and the sigmoid output after 10, 100, and 1,000 iterations over the sample. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

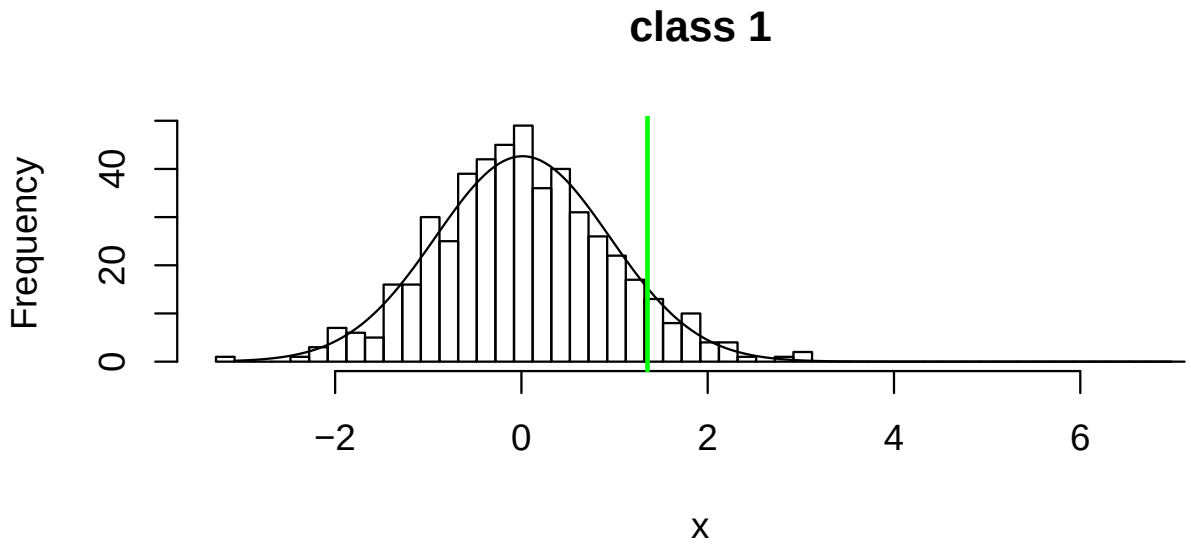
Generalized Linear Models

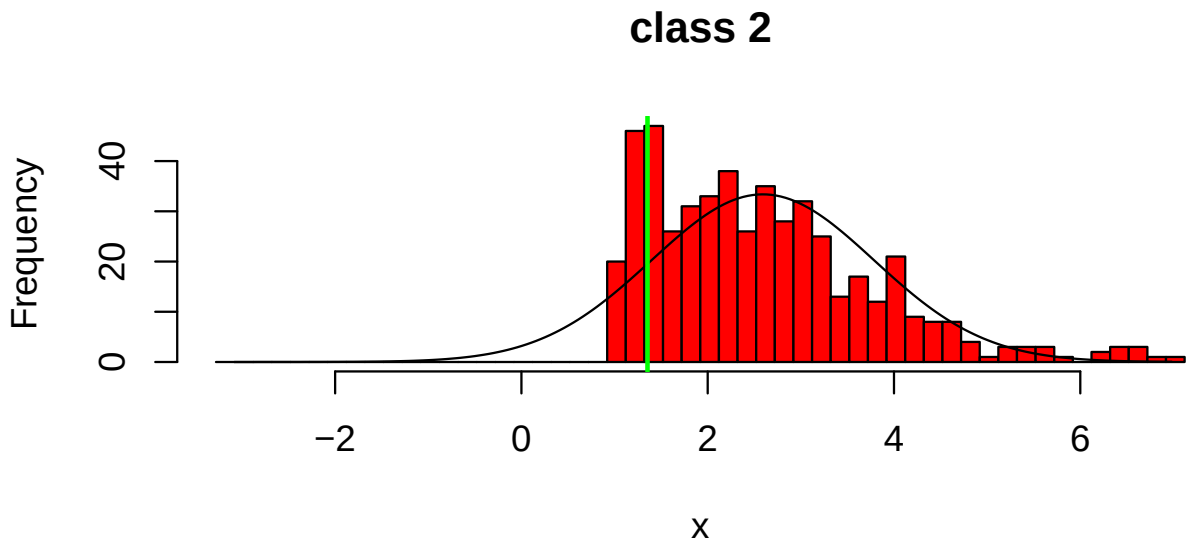
- Logistic regression is a special case of Generalized Linear Models (GLM)
  - logit is a *link function*.

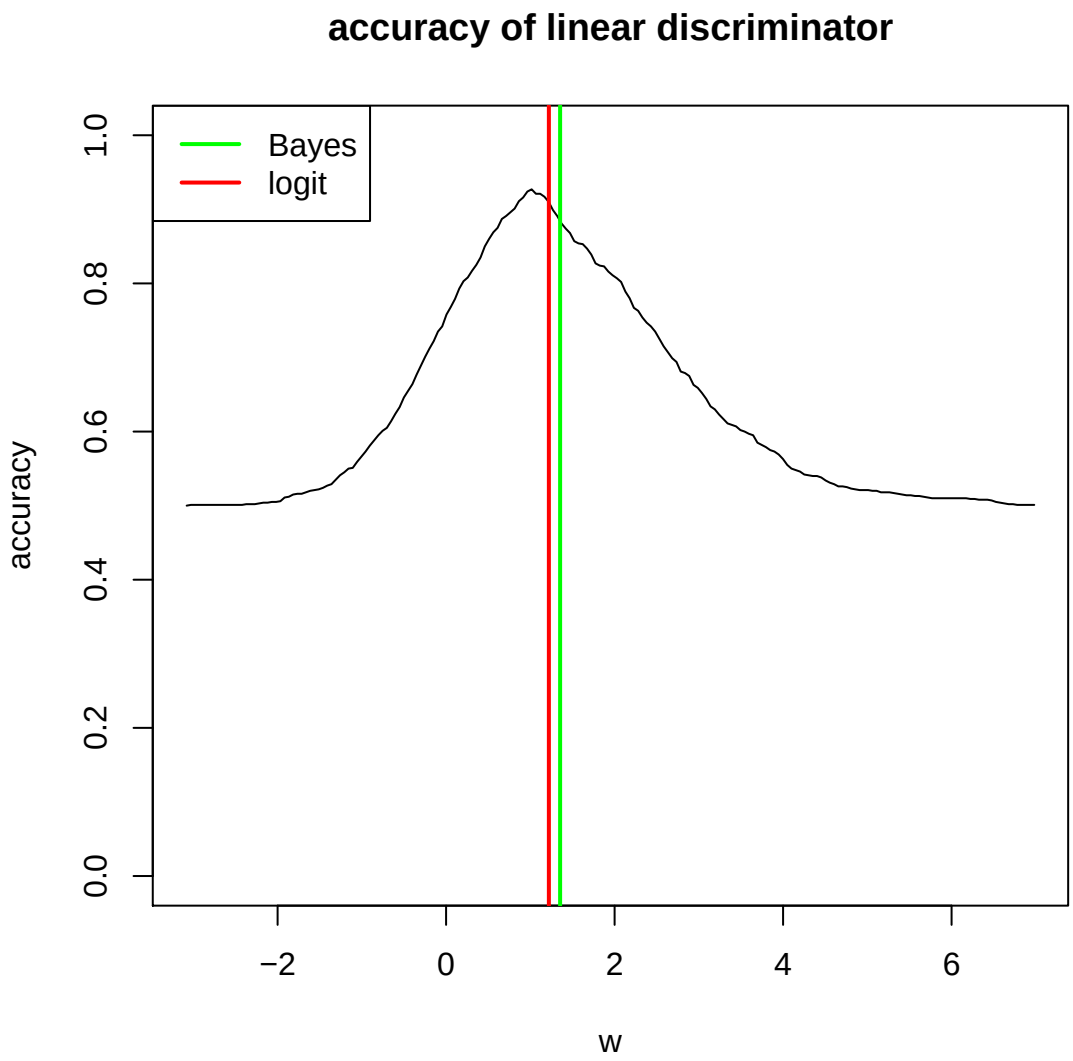
- Many respectable numerical packages contain GLM implementation which includes logistic regression (e.g., in R `help(glm)`). You should probably use these in real life applications instead of programming one on your own.

25.3 Logistic Regression vs. Naive Bayes

Naive Bayes Classifier







Accuracy of discriminator "class 1 if $x < w$, class 2 if $x \geq w$".

Naive Bayes vs. Logistic Regression

- Naive Bayes classifier estimates parameters of $P(\mathbf{r})$ and $P(\mathbf{x} \mid r)$ (means, covariances, etc.). (*generative* classifier, because we can generate the data points, given parameters)
- Logistic regression directly estimates the parameters of $P(r \mid \mathbf{x})$. (*discriminative* classifier, because we can directly discriminate wrt. r, given $\mathbf{x}$; no generative model for $p(\mathbf{x})$ is needed)
- If Naive Bayes assumptions hold (data from multivariate Gaussians with diagonal covariate matrix) and the number of training examples is very large, Naive Bayes and logistic regression give identical classification.
- The differences:

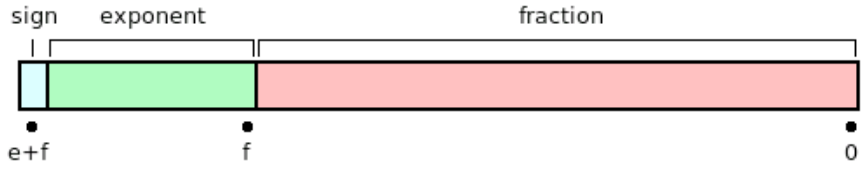
  - If data is not Gaussian etc. (that is, NB assumptions do not hold), logistic regression often gives better result (at least for large amounts of data).
  - Logistic regression needs more data. Naive Bayes needs $N = O(\log d)$ samples, while logistic regression needs $N = O(d)$. Ng & Jordan (2002) On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes. In Proc NIPS 14..

- Generative classifier: more bias, less variance. There is a model for $P(\mathbf{x})$. This is good if there is little data and/or the model for $\mathbf{x}$ is correct enough.
- Discriminative classifier: less bias, more variance. There is no model for $P(\mathbf{x})$, it is estimated directly from data. This is good if the NB model for $\mathbf{x}$ is wrong and/or there is enough data.

25.4 Floating Point Numbers

IEEE Floating Point Arithmetics

- The floating point numbers are stored in three parts in binary:
  - fraction ($f = 52$ bits in double precision)
  - exponent ($e = 11$ bits in double precision)
  - sign (1 bit)
- This includes the following types of numbers:
  - normalized numbers (normal non-zero numbers)
  - zero (± 0)
  - infinities ($\pm\infty$)
  - NaN
  - denormalized numbers ($\pm$ something very small or very large)



The three fields in an IEEE 754 float.

Image by Charles Esson, GFDL.

Numerical Computation: Computing Sums and Products

- Sometimes it is enough to use + and * operators to compute sums and products. According to R: 3.14*42=131.88; 3.14+42+5=50.14.
- Sometimes it is not. According to R: 3.14e-200*42e-201*1e300=0; 1e-400*1e400=NaN; 1e-16+1-1=0.
- In probabilistic modeling it is typical to...
  - Have numbers of different orders of magnitudes, including very small numbers.

  - Do sums and products with them.

- Important numbers (examples from the R floating point implementation in Mac OS X, `help(.Machine)`):
  - Smallest positive floating point number ϵ (*machine epsilon*) for which $1 + \epsilon \neq 1$: 2.2×10^{-16}.
  - The largest finite floating point number: 1.7×10^{308}.
  - The smallest positive floating point number: 2.2×10^{-308}.

Numerical Computation: Representing Numbers

- In many practical applications, 2.2×10^{-308} is too large for representing intermediate probabilities.
- Solution: store numbers as logs.
- Probabilities are usually always positive. (Generally, software should however be written so that to work consistently also with zero probabilities.)
- R is consistent also for zero probabilities: log(0)=-Inf; exp(-Inf)=0.
- Other software may behave differently. Read the documentation and test.

Numerical Computation: Computing Products

- Task: compute the product $y = \prod_{i=1}^{n} x_i$.
- 1e-200*1e-200*1e300=0 (wrong!).
- Solution: use logs.
- $\log y = \sum_{i=1}^{n} \log x_i$.
- log(1e-200)+log(1e-200)+log(1e300)=log(1e-100) (correct).
- Division: $\log(x/y) = \log x - \log y$. Product with negatives.

Numerical Computation: Computing Sums

- Task: compute sum $y = \sum_{i=1}^{n} x_i$.
- exp(-1000)+exp(-999)=0 (wrong!).
- Solution: scale numbers appropriately before doing the sum.
- $\log y = \log x_{MAX} + \log\left(\sum_{i=1}^{n} \exp\left(\log x_i - \log x_{MAX}\right)\right)$, where $\log x_{MAX} = \max_i \log x_i$.
- -999+log(exp(-1)+exp(0))=-998.6 (correct).
- Something like this: safesum <- function(x) { xmax <- max(x) ; xmax+log(sum(exp(x-xmax)))) }

Numerical Computation: Example

$$P(C_i \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid C_i)P(C_i)}{\sum_{k=1}^{K} P(\mathbf{x} \mid C_k)P(C_k)} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

Store numbers as logs and denote: $a[i] = \log P(\mathbf{x} \mid C_i)$, $b[i] = \log P(C_i)$.

```
safesum <- function(x) { xmax <- max(x); xmax+log(sum(exp(x-xmax))) }

evidence <- safesum(a+b)

posterior <- sum(c(a[i],b[i],-evidence))

exp(posterior) #P(C_i | x)
```

26 Announcements

26.1 Examination

- To pass the course you must pass the *examination* and the *term project*.
- Grading:
  - Examination grade $E \in [0, 1]$ (0 smallest passed grade)
  - Term project grade $T \in [0, 1]$ (0 smallest passed grade)
  - Problem session grade $P \in [0, 1]$
  - Course grade $\texttt{min}(5,\texttt{floor}(4E + 2T + P))$

Examination

- Currently scheduled at 19 Dec & 2 Feb & 15 May (check the times and locations from the examination schedule!)
- You must sign in to the examination at least one week in advance using WWWTopi
- Calculator (with memory erased) is allowed
- No other extra material is allowed.
- 4–6 problems (to pass you have to get about half of the points)

Reading List

- The examination is based on the topics covered in the lectures
- See `http://www.cis.hut.fi/Opinnot/T-61.3050/2007/examination`

26.2 Course Feedback

Course Feedback

- Please give course feedback at `http://www.cs.hut.fi/Opinnot/Palaute/kurssipalaute-en.html`
- (Open until 7 January 2008)

27 Summary of the Course

27.1 Summary of the Course

Objectives

- After this course, the student should...
  1. be able to *apply* the basic methods to real world data;
  2. *understand* the basic principles of the methods; and
  3. have necessary *prerequisites* to understand and apply new concepts and methods that build on the topics covered in the course.
- The topic is difficult (and interdisciplinary, involving at least computer science, mathematics, computational modeling and statistics)

Learning Tasks

- Supervised learning
  - classification
  - regression
- Unsupervised learning
  - clustering etc.
- Reinforcement learning [not in this course]

Concept Learning

- Task: classify a previously unseen instance into positive or negative
- Hypothesis class $\mathcal{H}$
- Learning: use positive and negative examples to prune out the hypothesis
- If none of the hypothesis in the hypothesis class is correct we might end up with no consistent hypothesis.

- *Inductive bias*: we must restrict the allowed hypothesis to be able to *generalize* (predict classes of new instances).
- The choice of a hypothesis space is called *model selection*.
- *Underfitting*: the hypothesis space is too simple.
- *Overfitting*: the hypothesis space is too complex.
- VC dimension can be used to measure the complexity of the hypothesis space.

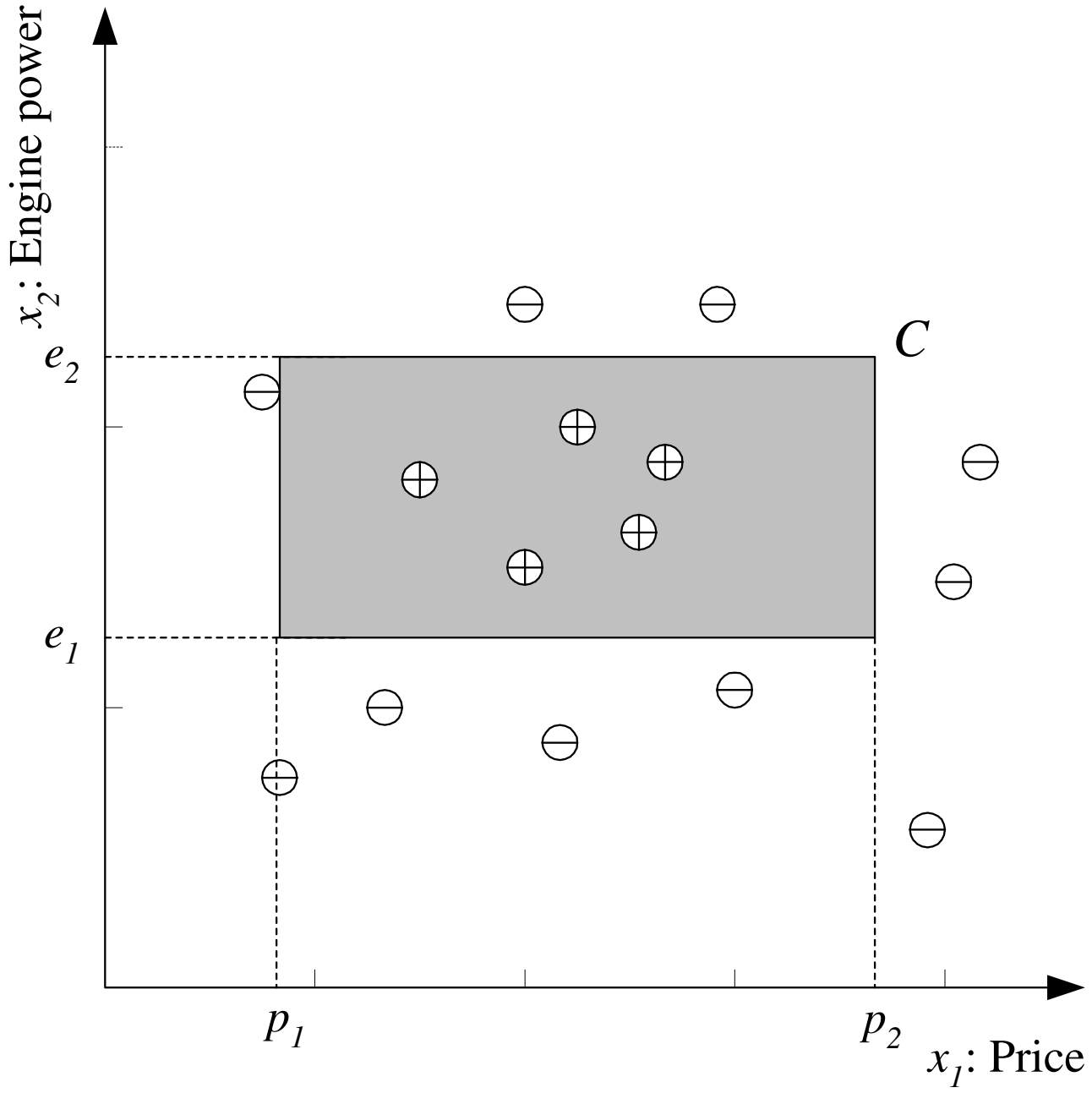


Figure 2.2 of Alpaydin (2004).

Regression with Noise

- Classification is the prediction of a 0–1 class, given attributes.
- *Regression* is the prediction of a real number, given
- Usually, we want to minimize a quadratic error function,

$$E(g \mid \mathcal{X}) = \frac{1}{N} \sum_{t=1}^{N} \left(r^t - g(\mathbf{x}^t)\right)^2.$$

Dimensions of a Supervised Learner

Model

$$g(\mathbf{x} \mid \theta)$$

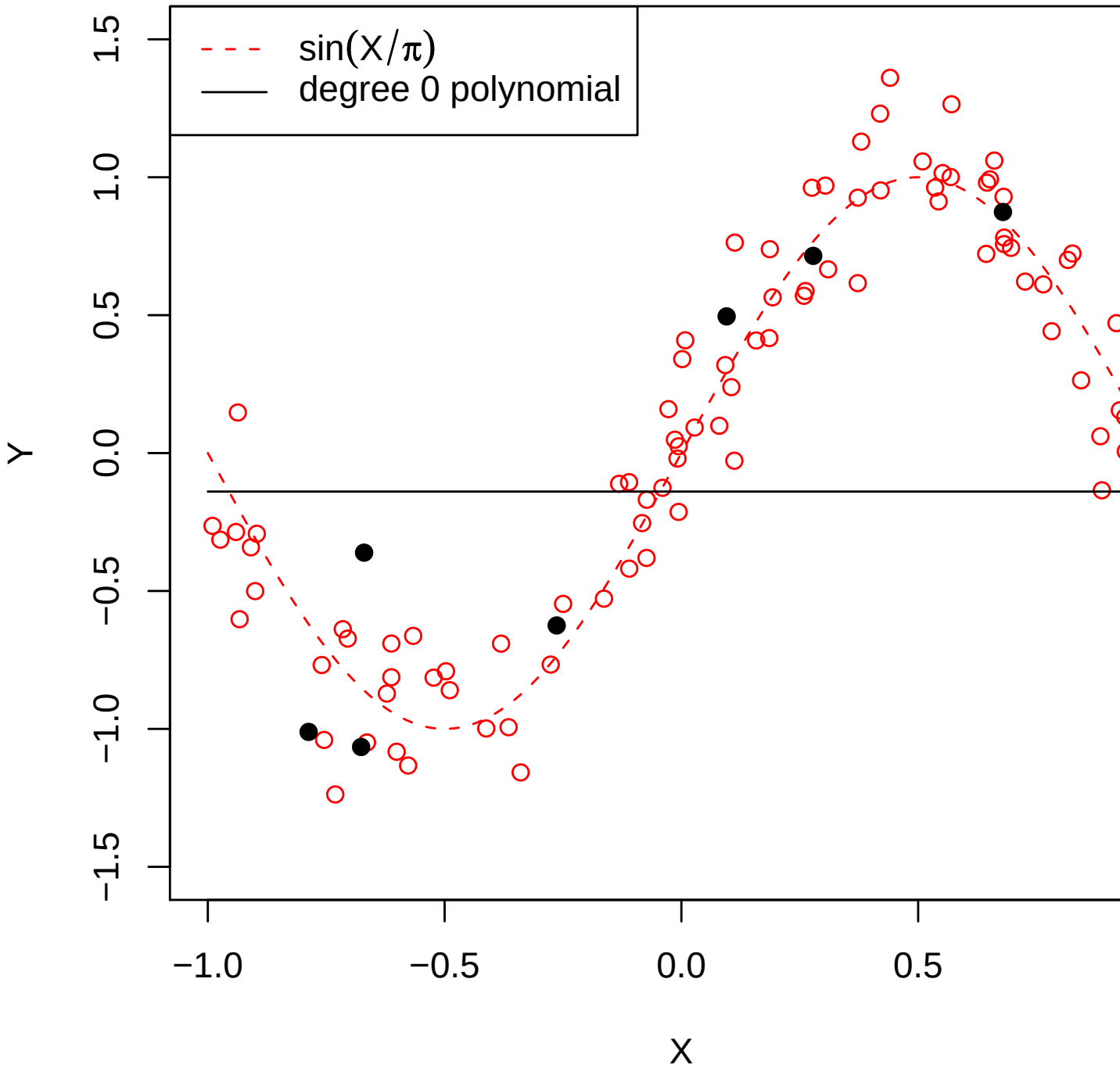
Loss Function

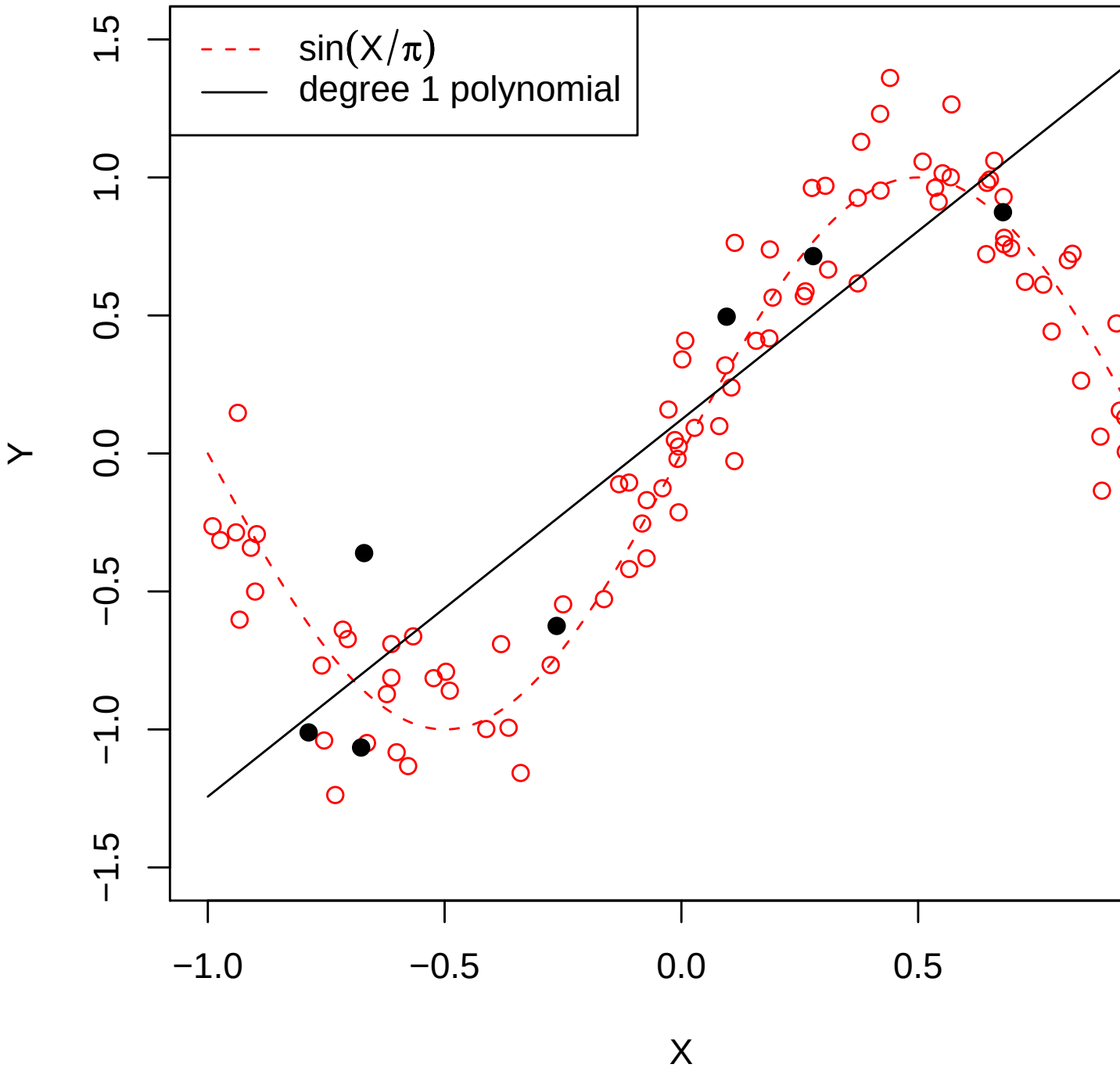
$$E\left(\theta \mid \mathcal{X}\right) = \frac{1}{N} \sum_{t=1}^{N} L\left(r^t, g(\mathbf{x}^t \mid \theta)\right).$$

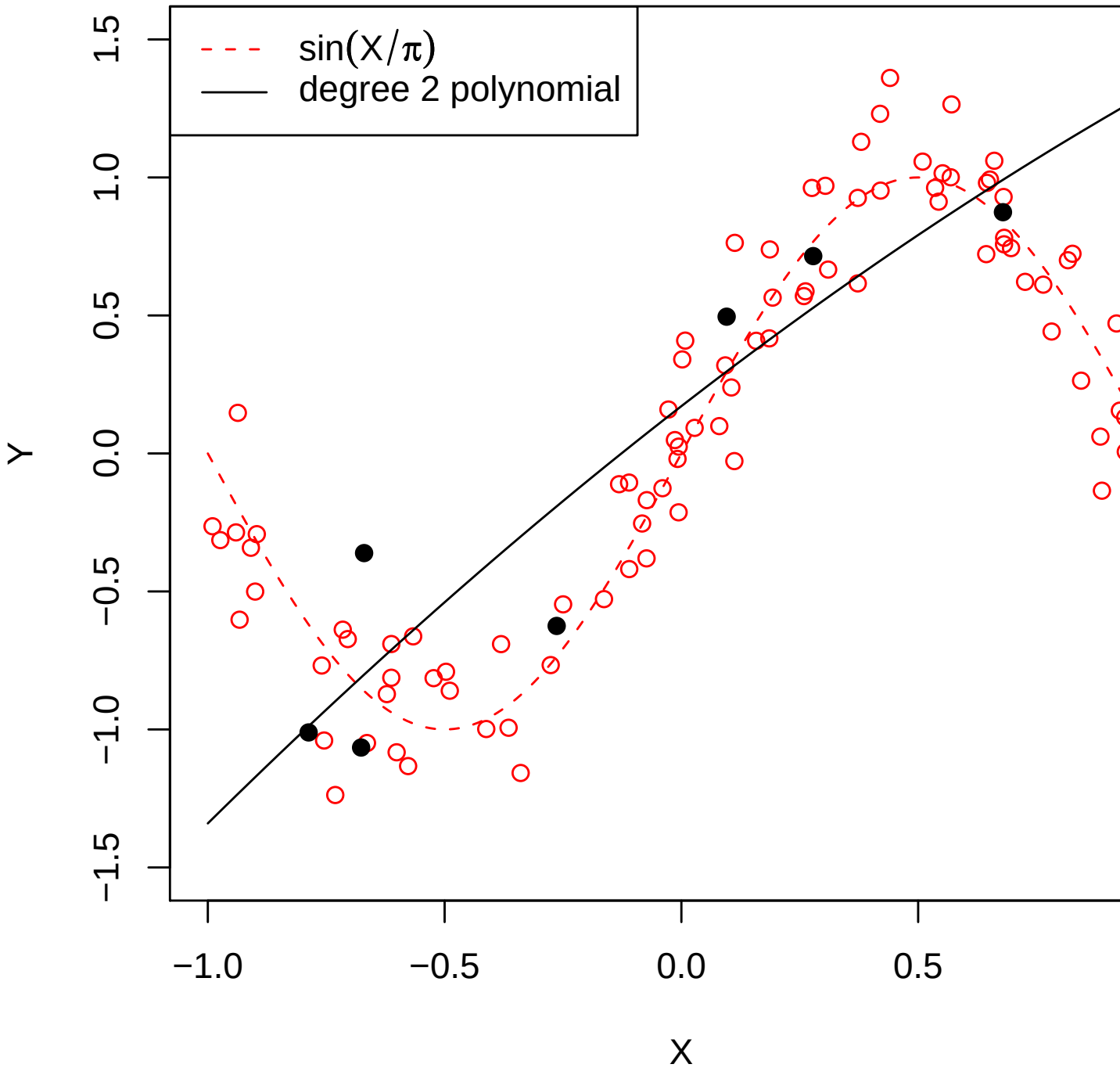
Optimization Procedure

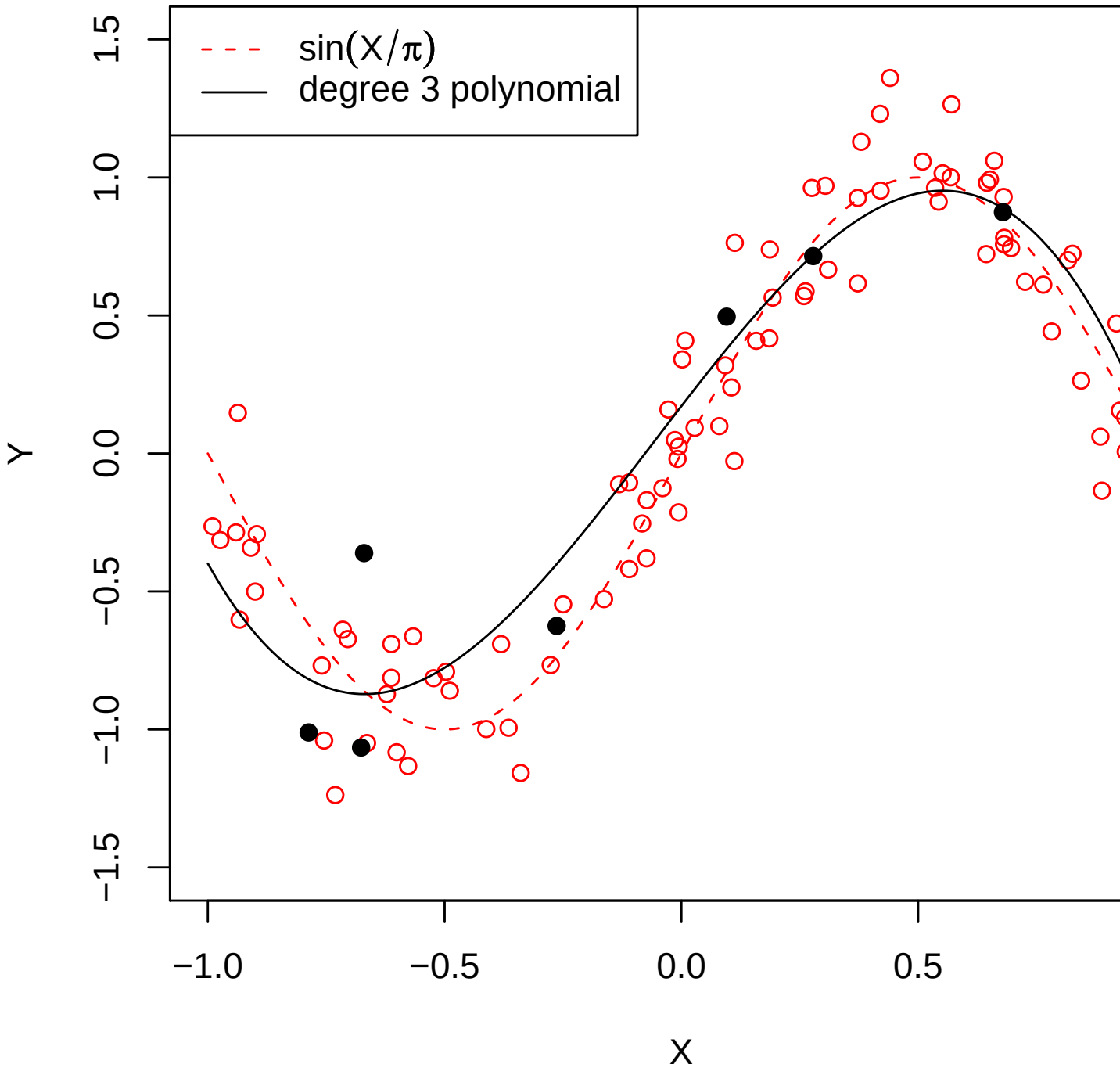
$$\theta \leftarrow \arg\min_{\theta} E\left(\theta \mid \mathcal{X}\right).$$

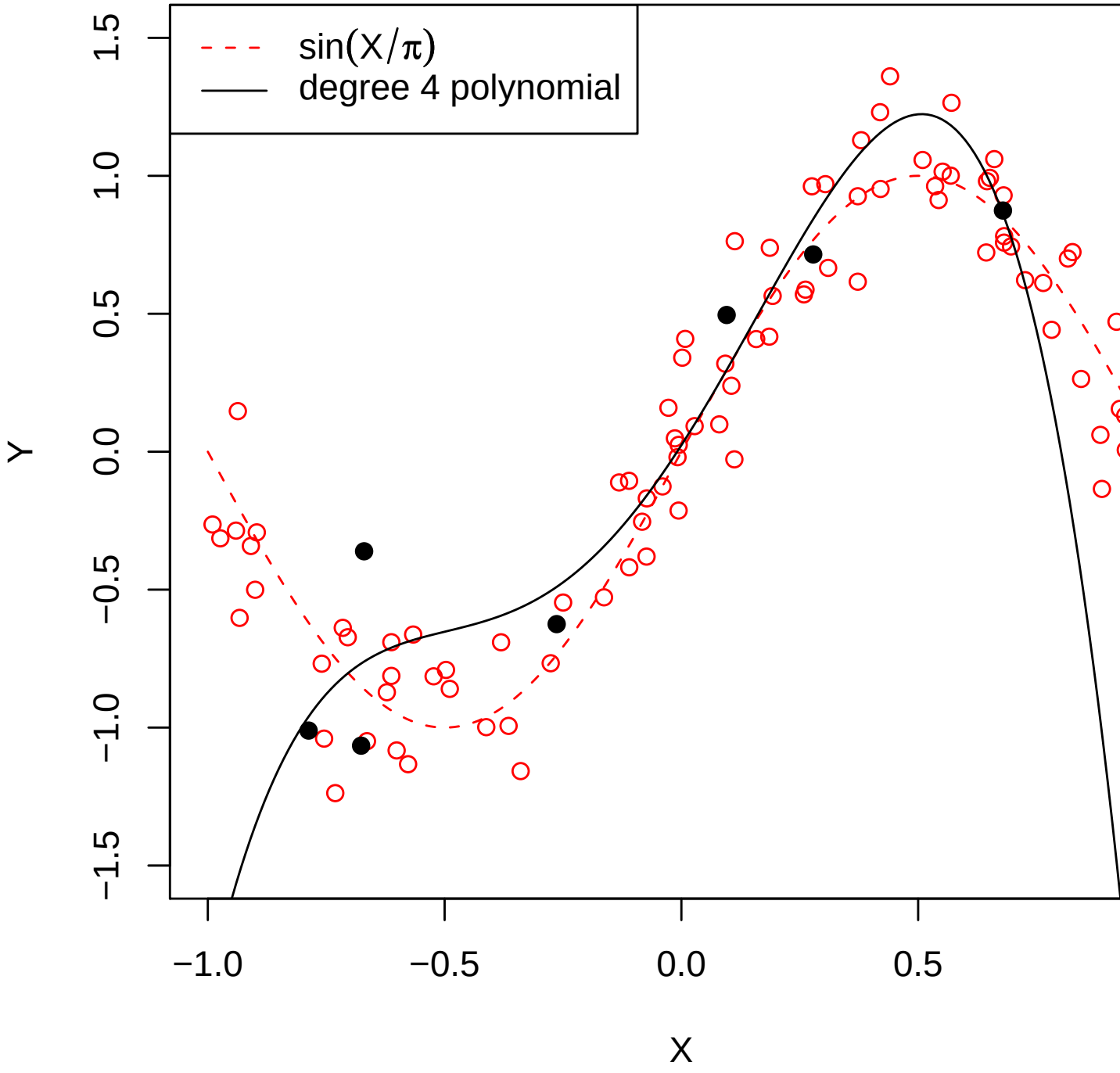
Polynomial Regressors

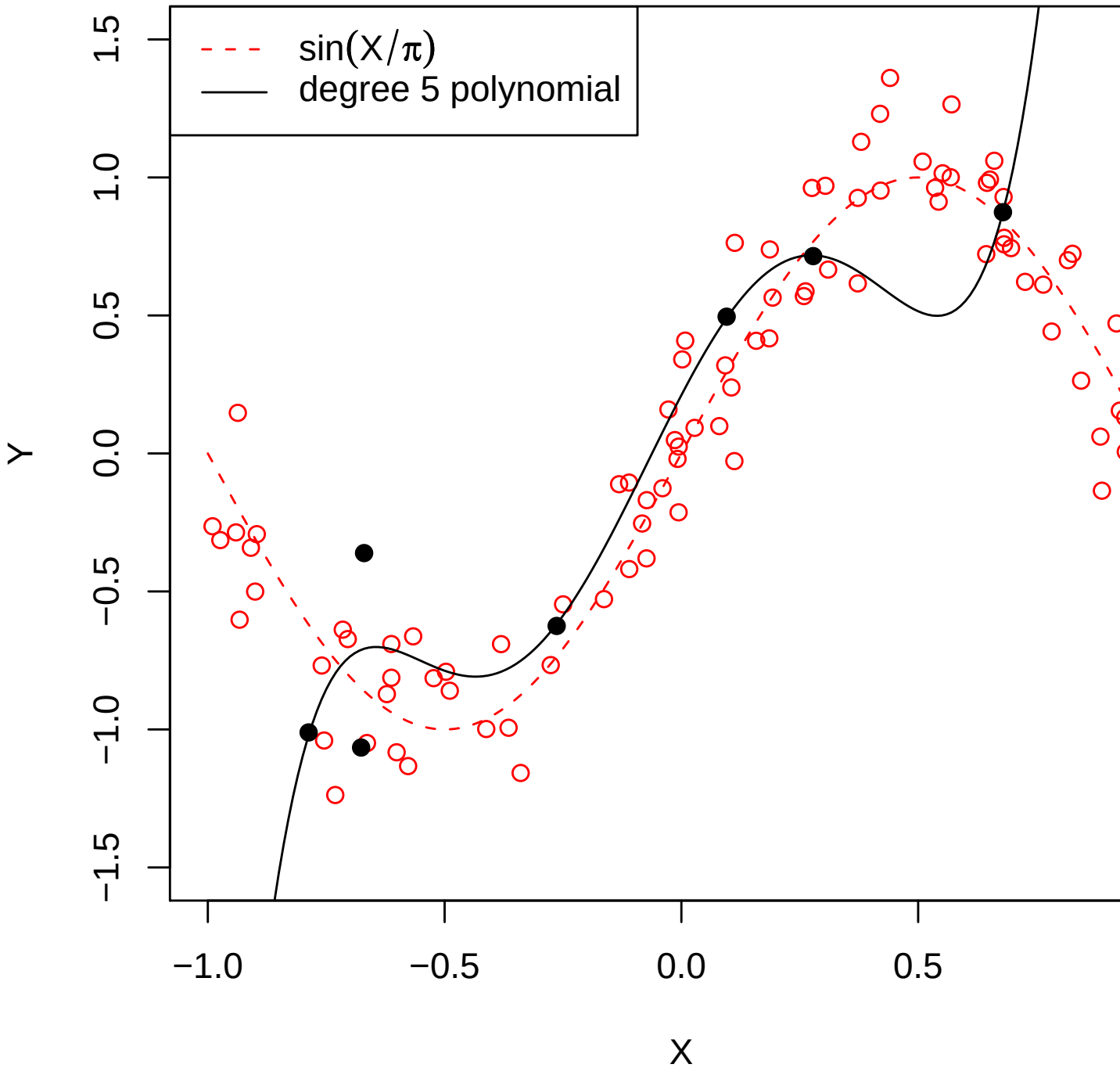

sin(X/π)
degree 0 polynomial
Y
X
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5

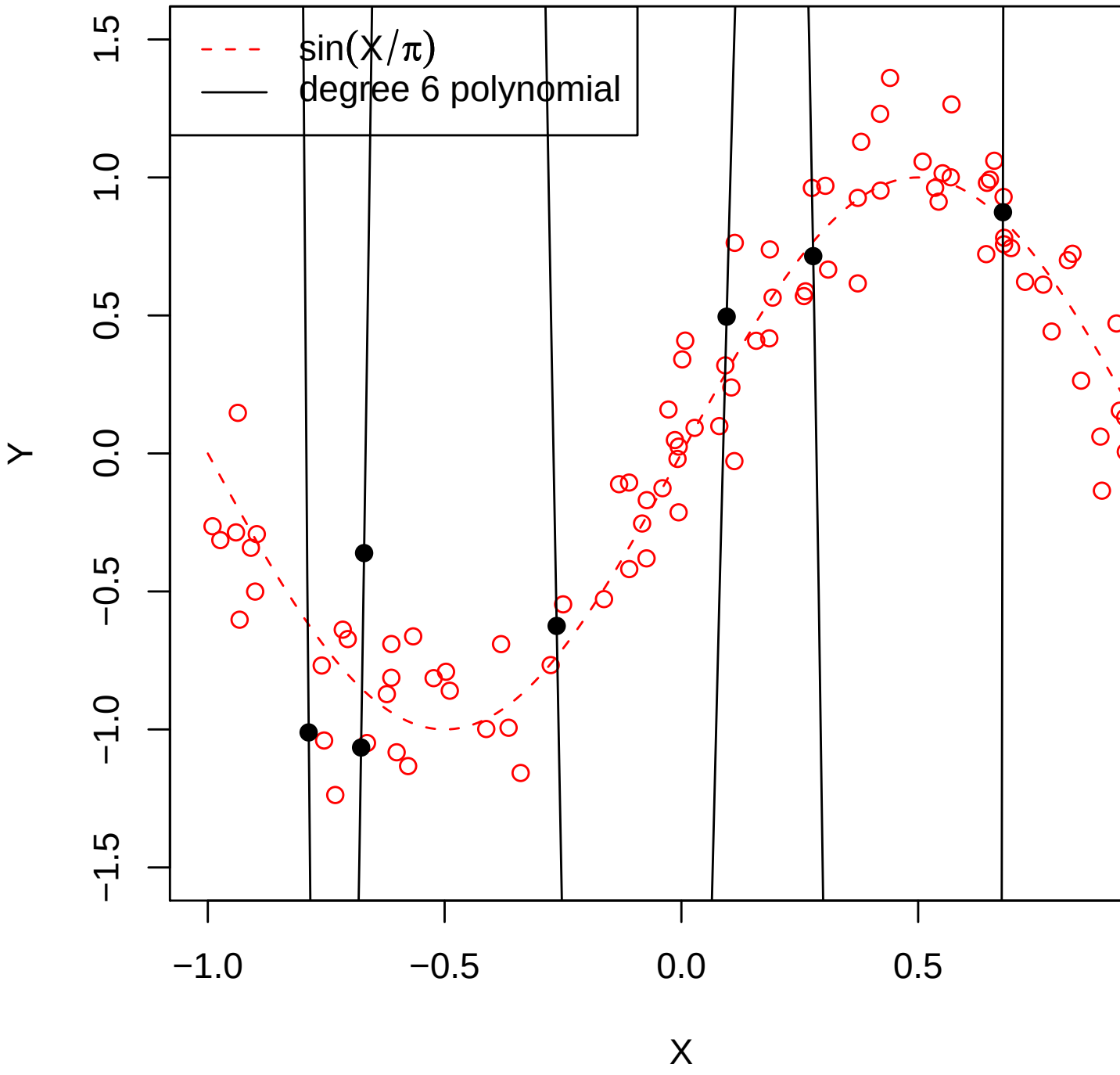

sin(X/π)
degree 1 polynomial
Y
X
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5


sin(X/π)
degree 2 polynomial
Y
X
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5

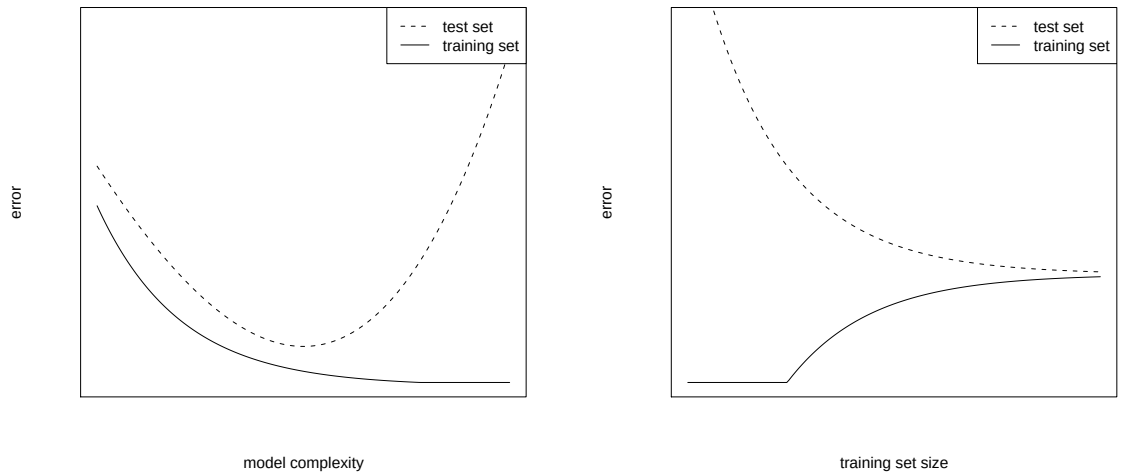

sin(X/π)
degree 3 polynomial
Y
X
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5


sin(X/π)
degree 4 polynomial
Y
X
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5


sin(X/π)
degree 5 polynomial
Y
X
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5


sin(X/π)
degree 6 polynomial
Y
X
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5

Model Selection and Generalization



- empirical error = error on training set
- generalization error = error on test set
- We see empirical error, but want to minimize the error on new data.
- Training vs. validation vs. test sets

K-Fold Cross-Validation

- How to use the training/validation data most efficiently?

CV($\mathcal{X}$,A,K){Input: $\mathcal{X}$, data $\mathcal{X} = \{(r^t, x^t)\}_{t=1}^N$; A, classification algorithm; K, number of folds. Output: $\mathcal{E}$, error measure.}
Partition $\mathcal{X}$ in random into K roughly equally sized parts $\mathcal{X}_i$.
for all $i \in \{1, \ldots, K\}$ **do**
  Train A using $\mathcal{X} \setminus \mathcal{X}_i$ as a training set.
  Let $\mathcal{E}_i$ be the error of A in $\mathcal{X}_i$ (for example, the fraction of incorrectly labeled items).
end for
$\mathcal{E} \leftarrow \sum_{i=1}^K |\mathcal{X}_i| \mathcal{E}_i / |\mathcal{X}|$
return $\mathcal{E}${$\mathcal{E}$ can be used as a validation set error in model selection.}

Rules of Probability

- In presence of noise, we have to use probabilities.
- In principle, you can derive everything in probabilistic inference from the basic axiom, including the sum and product rules.

Rules of Probability

- $P(E, F) = P(F, E)$: probability of both E and F happening.
- $P(E) = \sum_F P(E, F)$ (sum rule, marginalization)
- $P(E, F) = P(F \mid E)P(E)$ (product rule, conditional probability)
- Consequence: $P(F \mid E) = P(E \mid F)P(F)/P(E)$ (Bayes' formula)
- We say E and F are *independent* if $P(E, F) = P(E)P(F)$ (for all E and F).
- We say E and F are *conditionally independent* given G if $P(E, F \mid G) = P(E \mid G)P(F \mid G)$, or equivalently $P(E \mid F, G) = P(E \mid G)$.

Bayes' Rule

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

- $P(C_k) \geq 0$ and $\sum_{k=1}^{K} P(C_k) = 1$.
- *Naive Bayes Classifier*: choose C_k where $k = \arg\max_k P(C_k \mid \mathbf{x})$.

Classifier Using Probabilistic Model

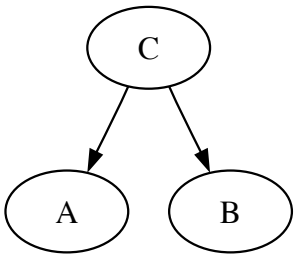
- First compute posterior class probability $P(C \mid \mathbf{x})$.
- Choose class C with the largest posterior probability.
- Another option: Choose class which minimizes risk (or maximizes utility), if the loss of misclassification is not a constant.
- A class for uncertainty: reject-option

Bayesian Networks

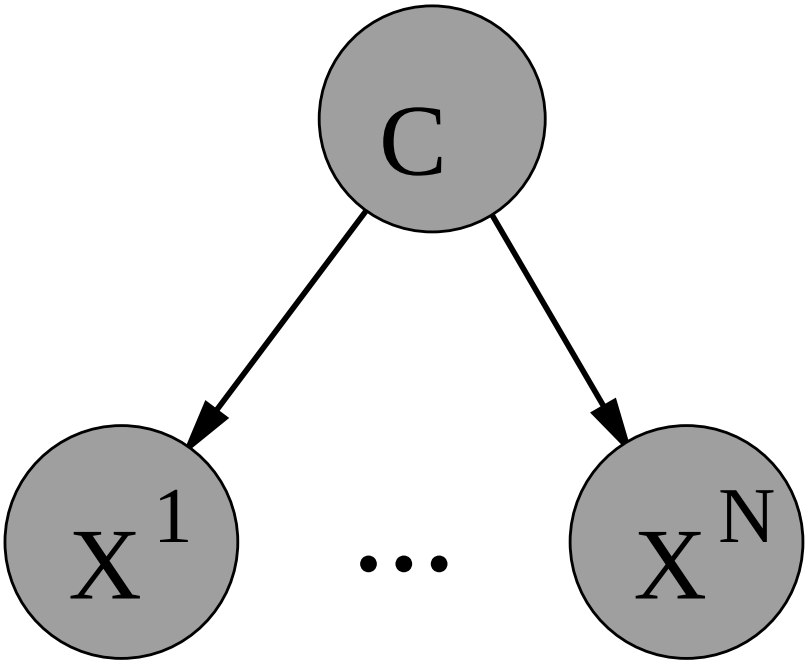
Bayesian network is a directed acyclic graph (DAG) that describes a joint distribution over the vertices $X_1, \ldots, X_d$ such that

$$P(X_1, \ldots, X_d) = \prod_{i=1}^{d} P(X_i \mid \text{parents}(X_i)),$$

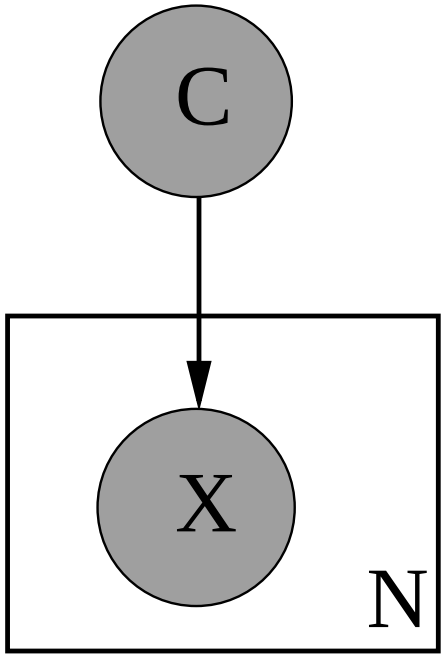
where parents(X_i) are the set of vertices from which there is an edge to X_i.



$P(A, B, C) = P(A \mid C)P(B \mid C)P(C)$. ($A$ and B are conditionally independent given C.)



Equivalently:



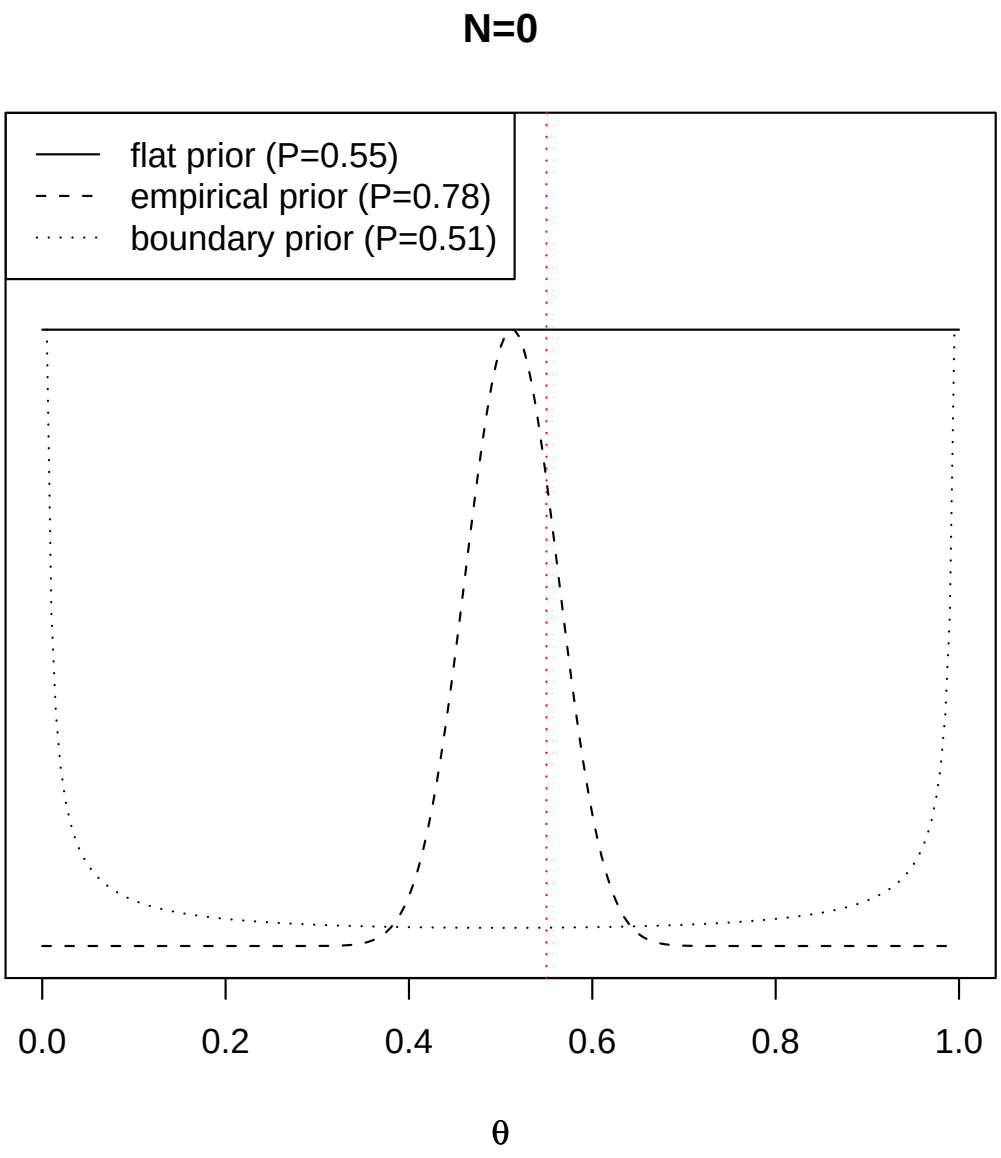
- *Plate* is used as a shorthand notation for repetition. The number of repetitions is in the bottom right corner.
- Gray nodes denote observed variables.

Estimating the Sex Ratio

- What is our degree of belief in the gender ratio, before seeing any data (*prior probability density* $p(\theta)$)?

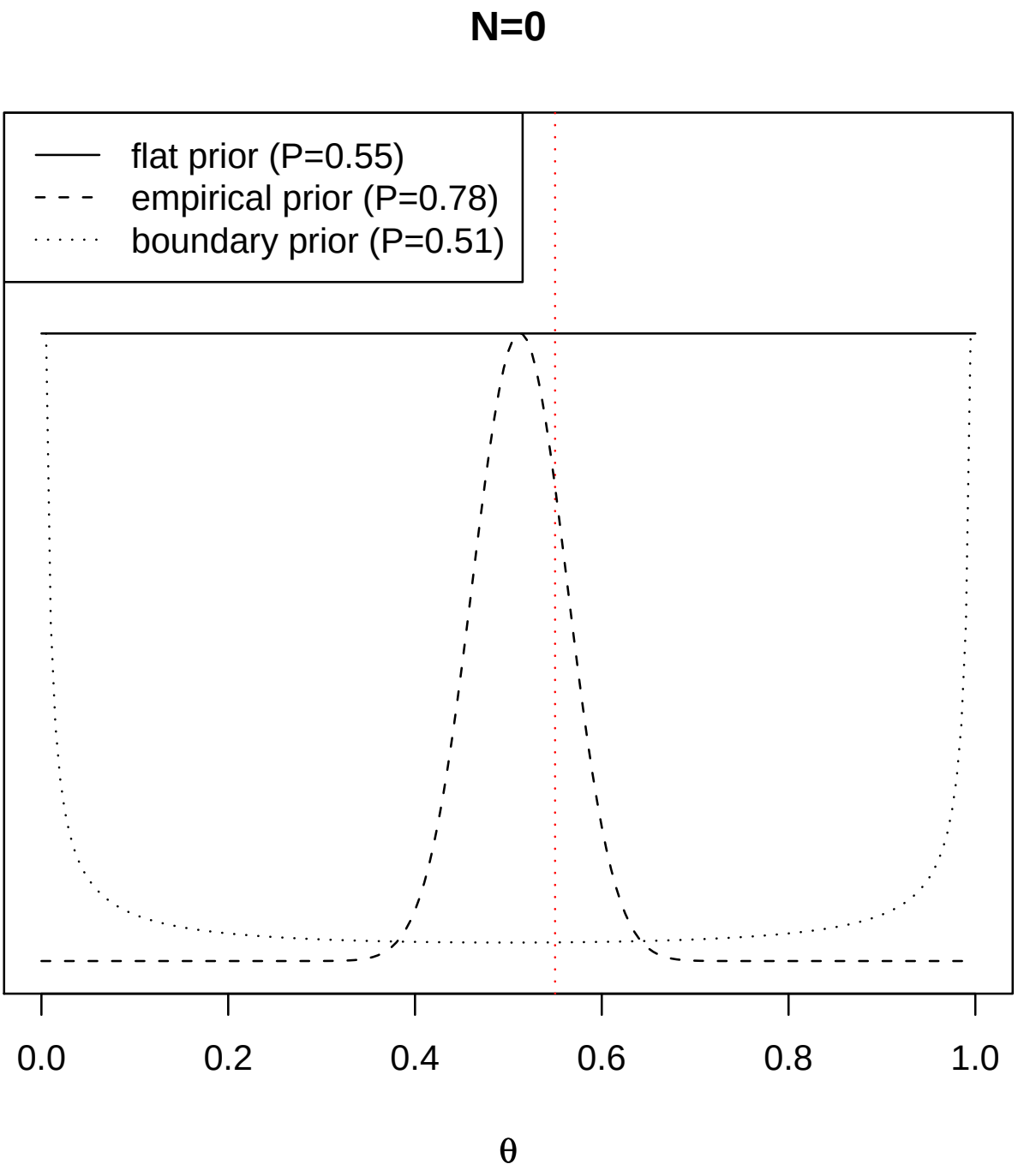
- What is our degree of belief in the gender ratio, after seeing data X (*posterior probability density* $p(\theta \mid \mathcal{X})$)?

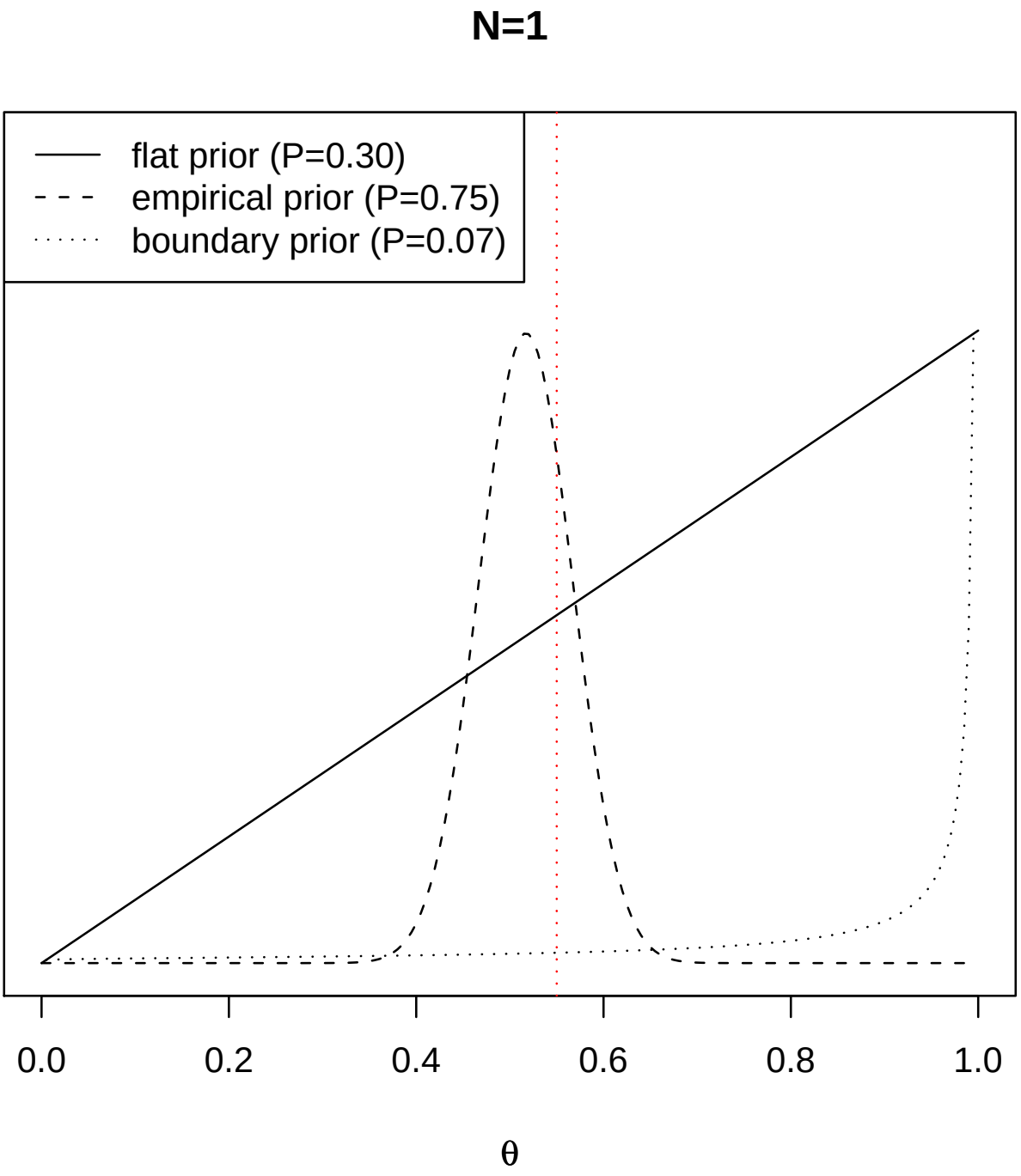
$$p(\theta \mid \mathcal{X}) \propto p(\theta)p(\mathcal{X} \mid \theta).$$

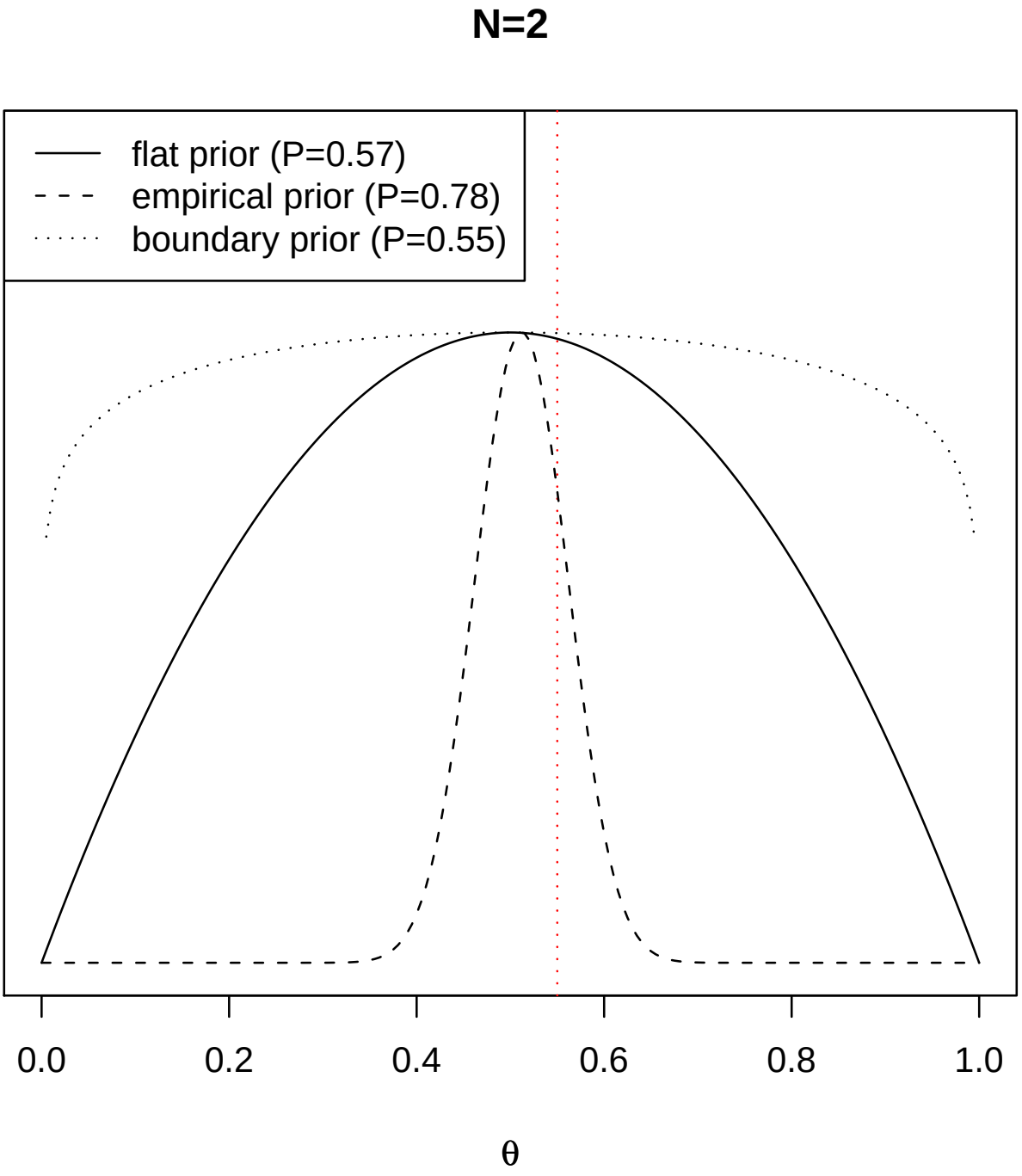


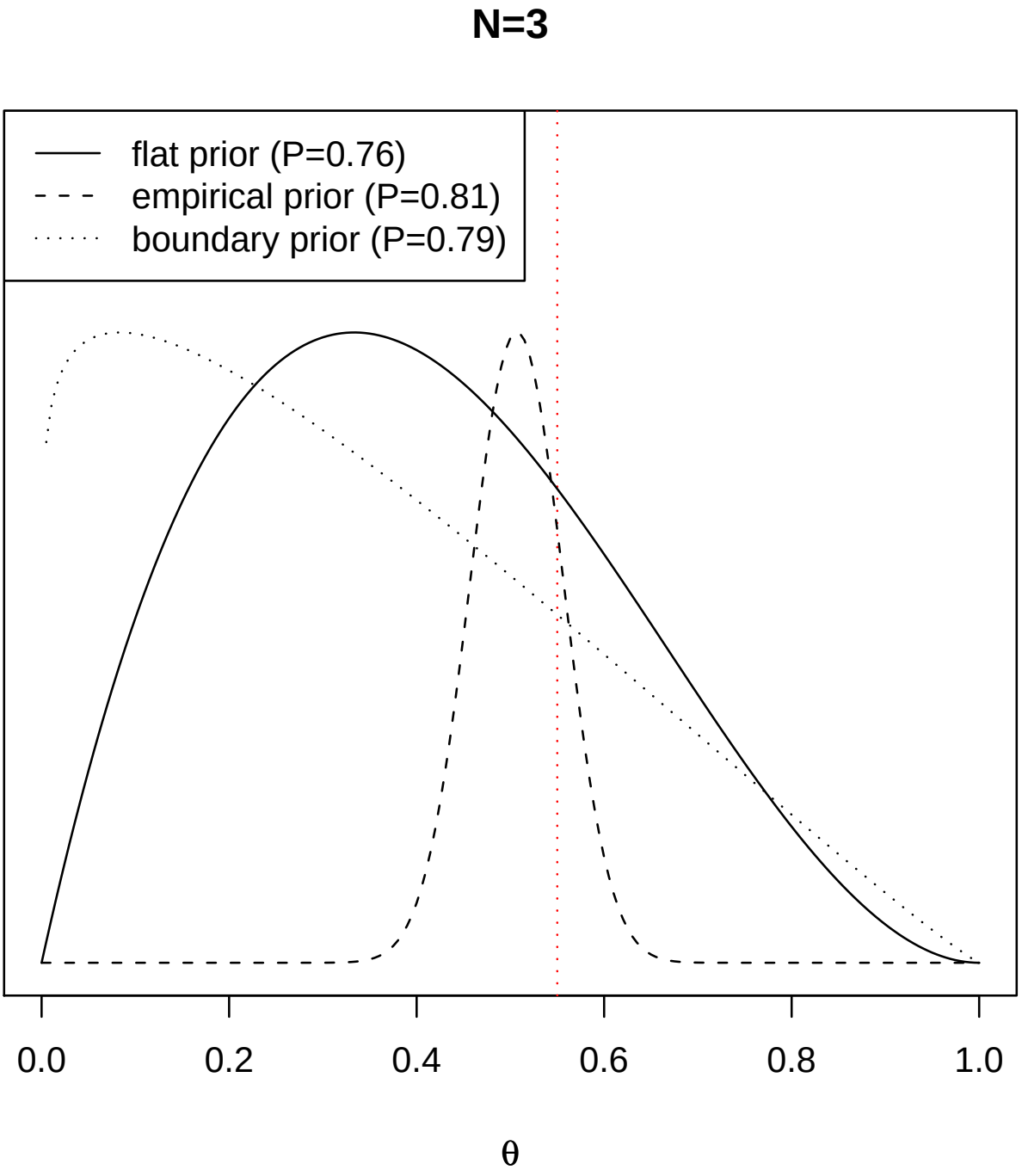
"True" $\theta = 0.55$ is shown by the red dotted line. The densities have been scaled to have a maximum of one.

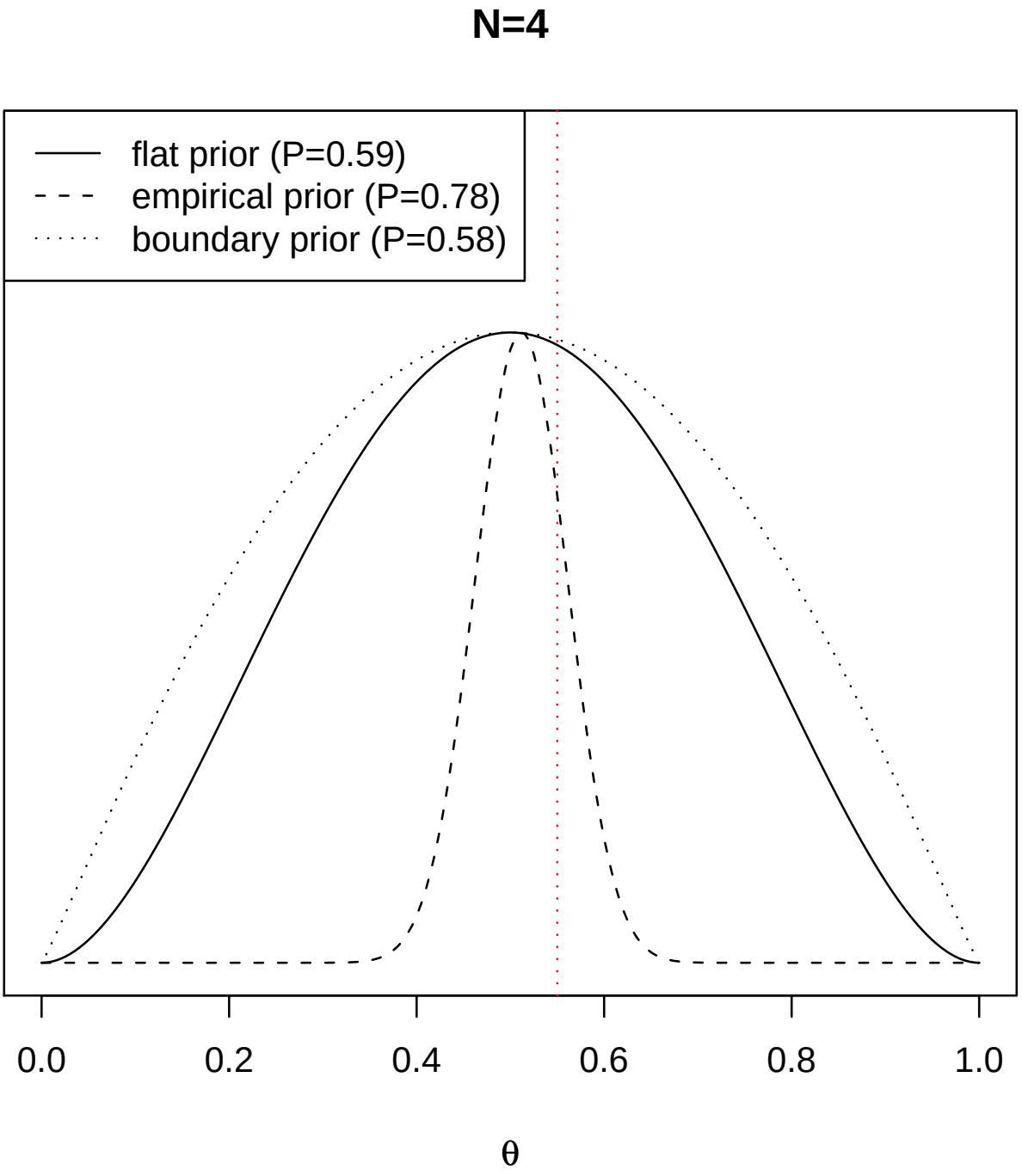
Estimating the Sex Ratio

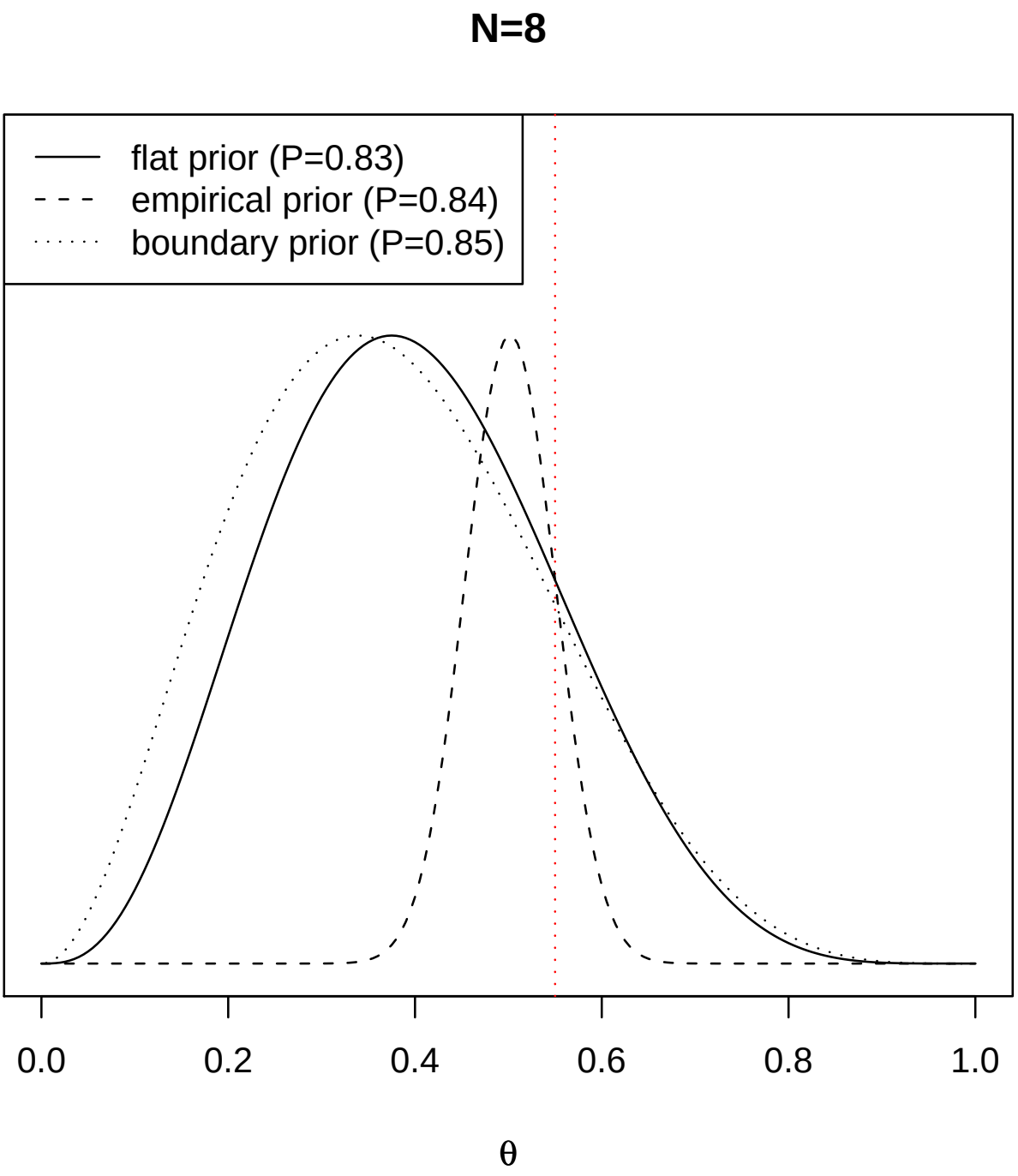

N=0
flat prior (P=0.55)
empirical prior (P=0.78)
boundary prior (P=0.51)
0.0
0.2
0.4
0.6
0.8
1.0
θ

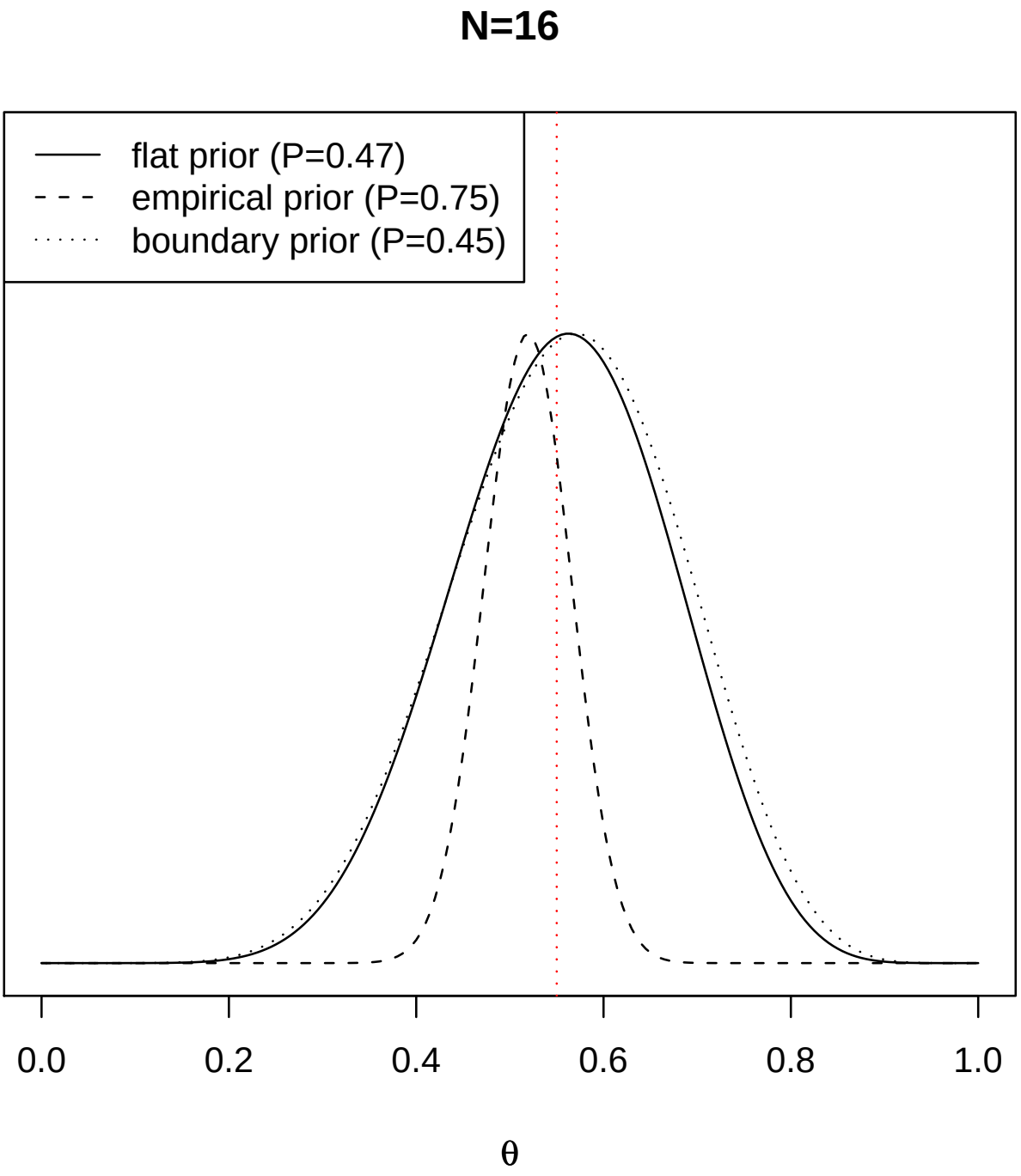

N=1
flat prior (P=0.30)
empirical prior (P=0.75)
boundary prior (P=0.07)
0.0
0.2
0.4
0.6
0.8
1.0
θ

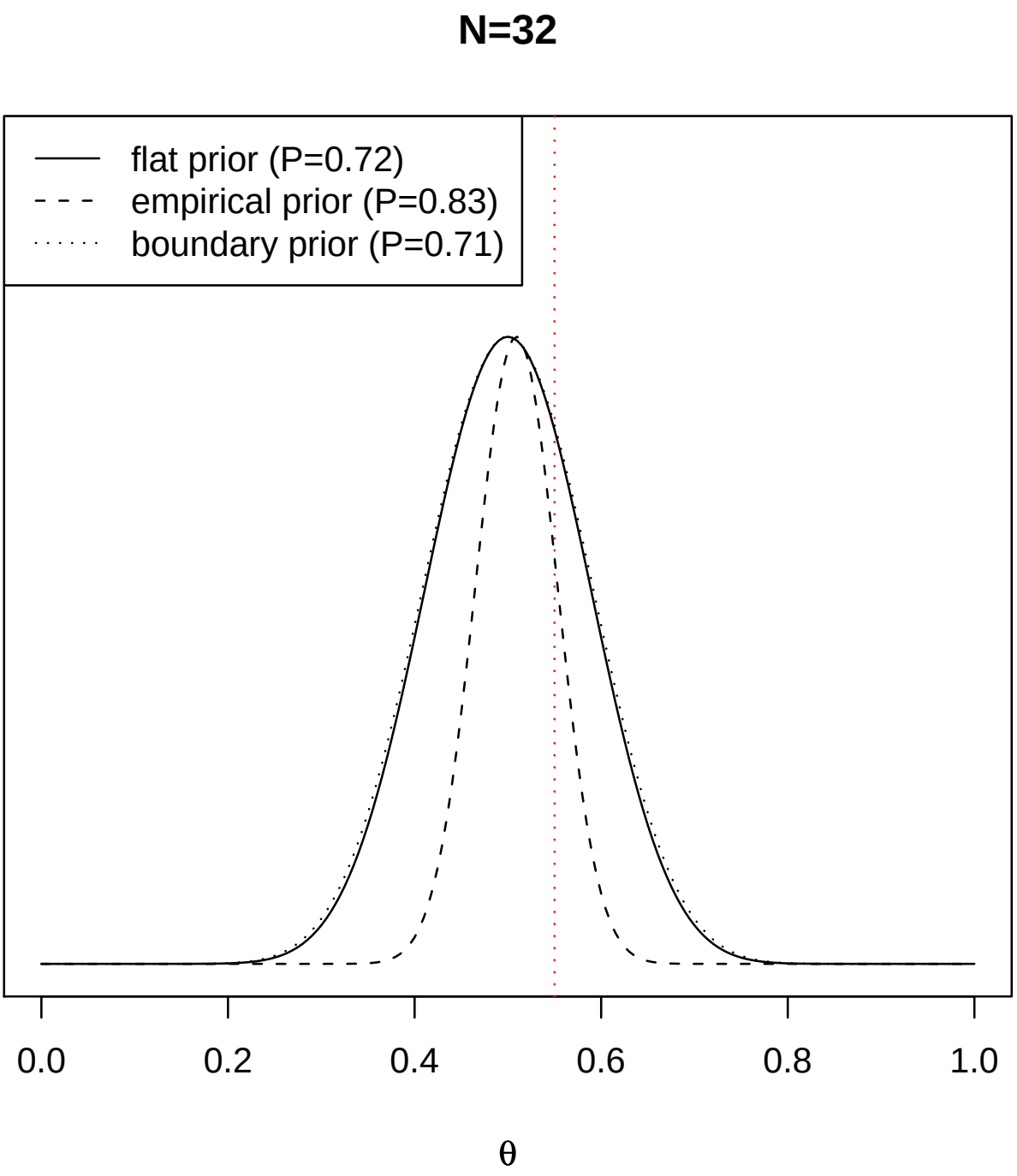

N=2
flat prior (P=0.57)
empirical prior (P=0.78)
boundary prior (P=0.55)
0.0
0.2
0.4
0.6
0.8
1.0
θ

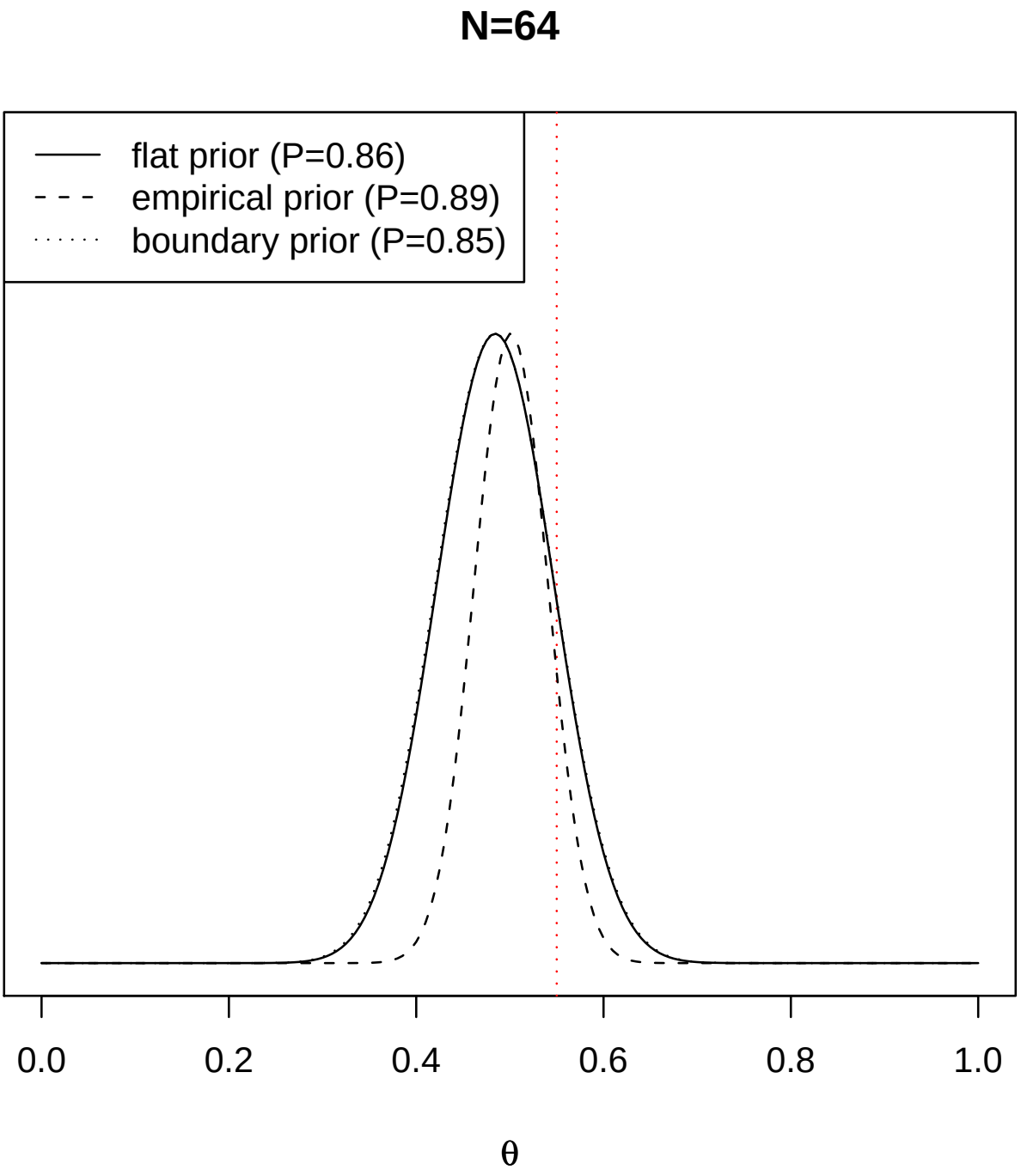

N=3
flat prior (P=0.76)
empirical prior (P=0.81)
boundary prior (P=0.79)
0.0
0.2
0.4
0.6
0.8
1.0
θ

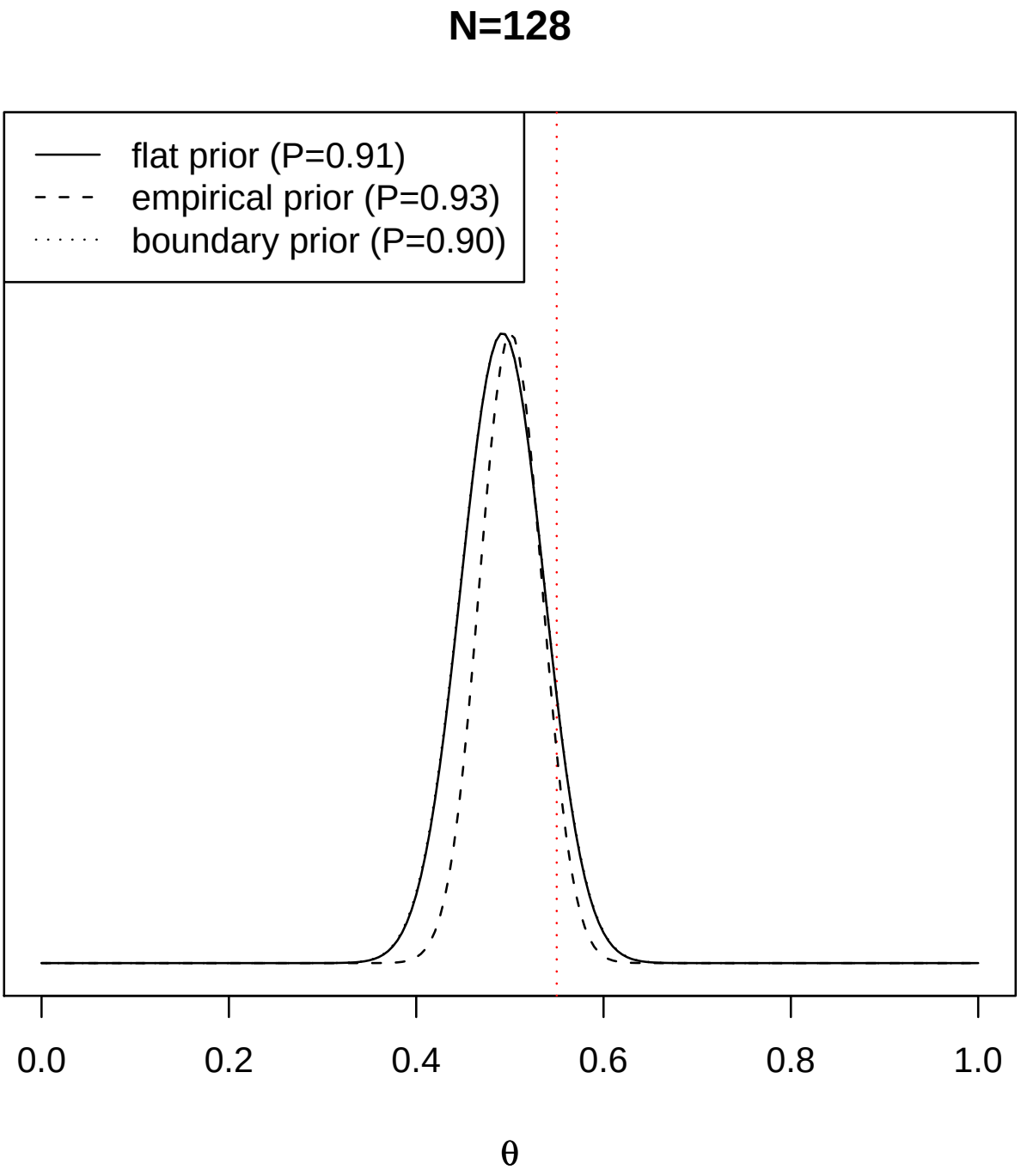

N=4
flat prior (P=0.59)
empirical prior (P=0.78)
boundary prior (P=0.58)
0.0
0.2
0.4
0.6
0.8
1.0
θ

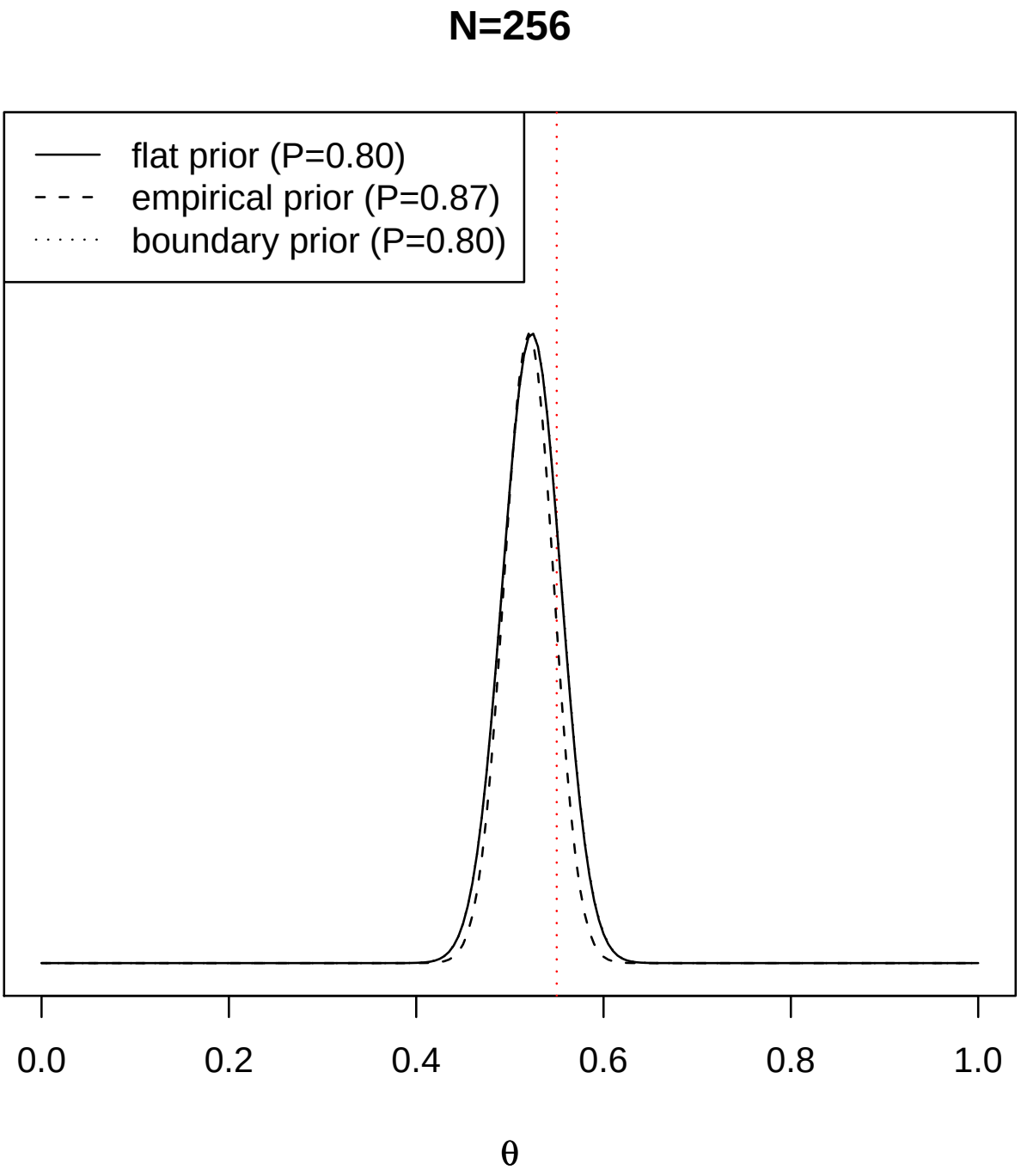

N=8
flat prior (P=0.83)
empirical prior (P=0.84)
boundary prior (P=0.85)
0.0
0.2
0.4
0.6
0.8
1.0
θ

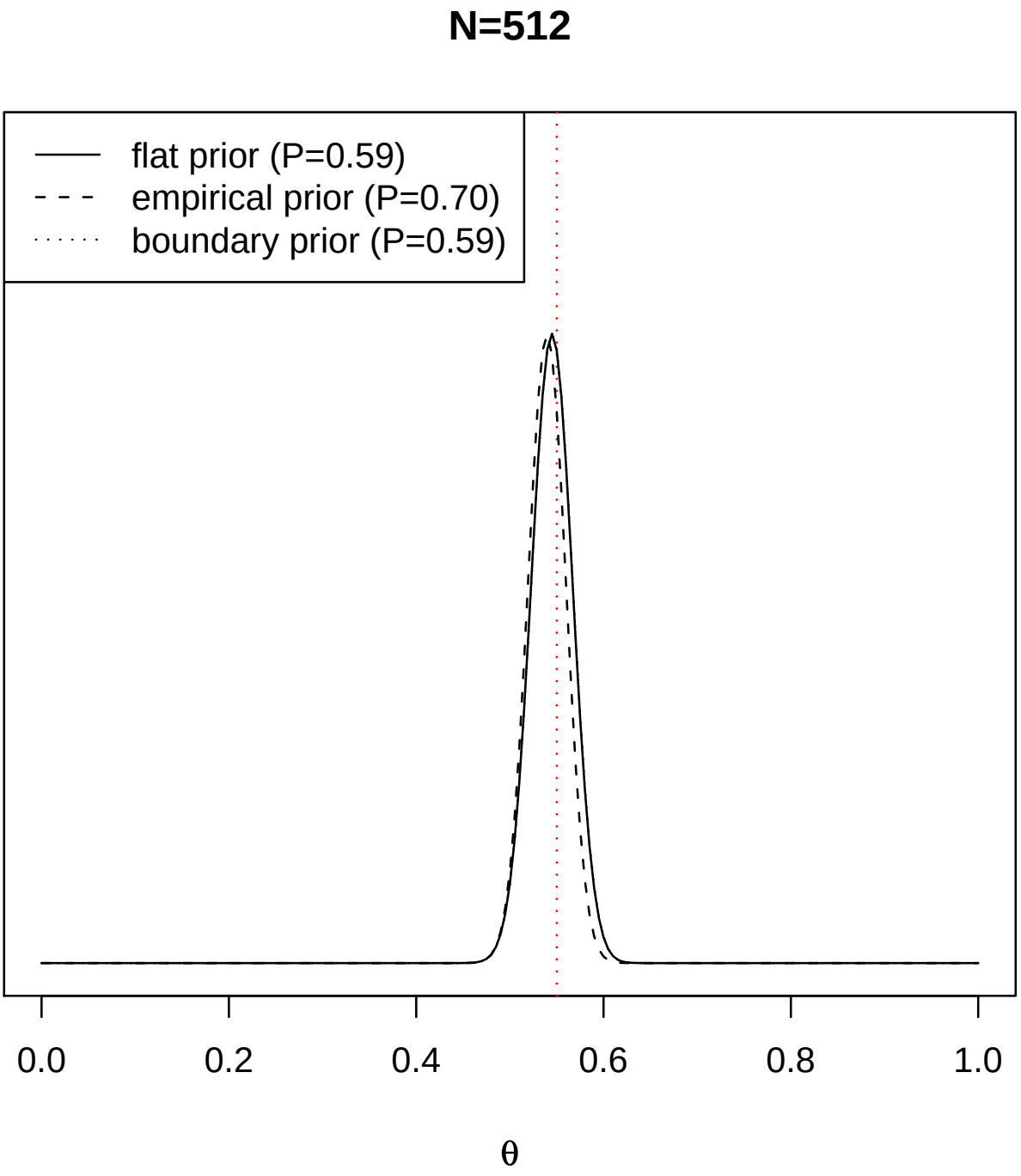

N=16
flat prior (P=0.47)
empirical prior (P=0.75)
boundary prior (P=0.45)
0.0
0.2
0.4
0.6
0.8
1.0
θ

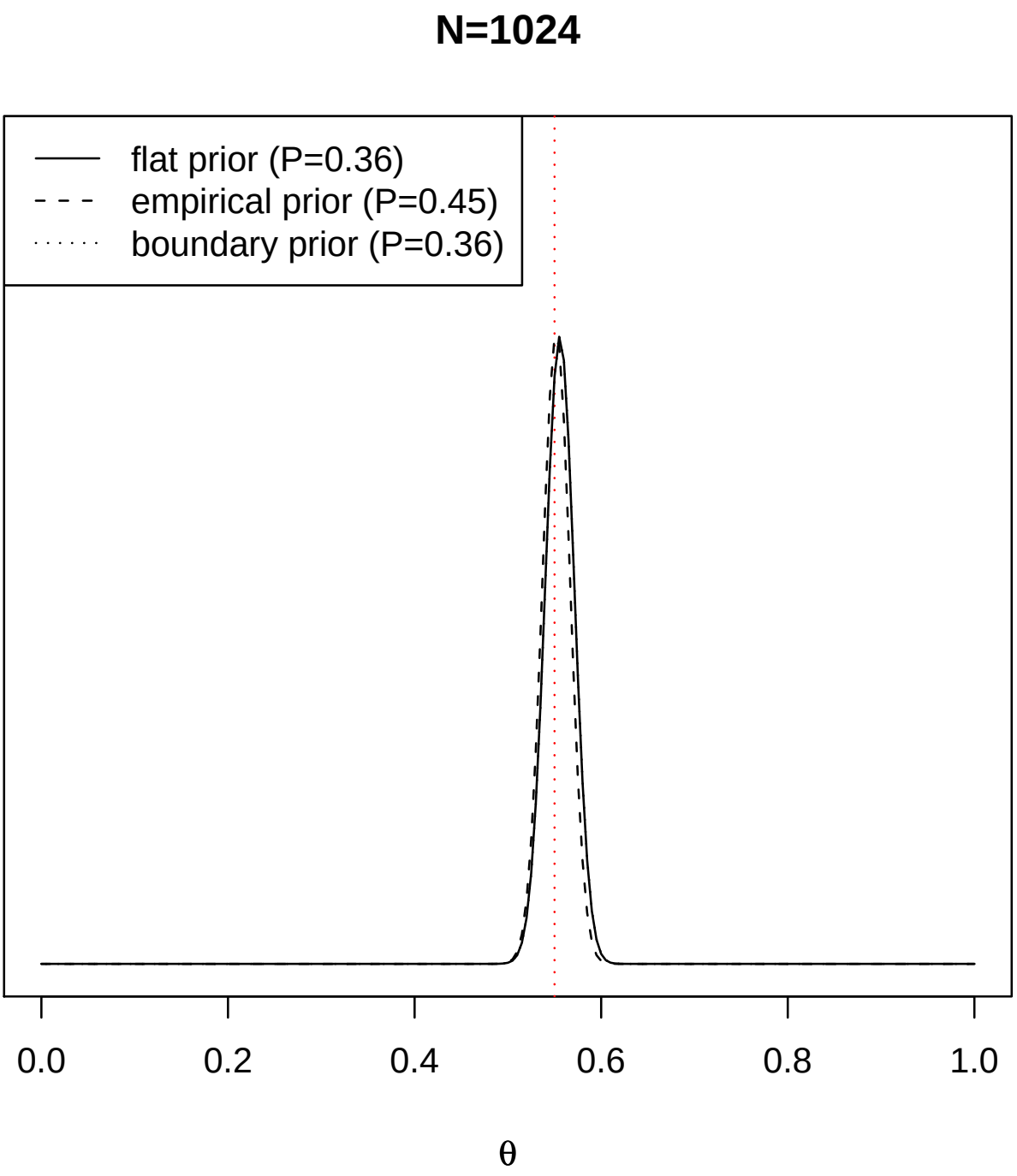

N=32
flat prior (P=0.72)
empirical prior (P=0.83)
boundary prior (P=0.71)
0.0
0.2
0.4
0.6
0.8
1.0
θ

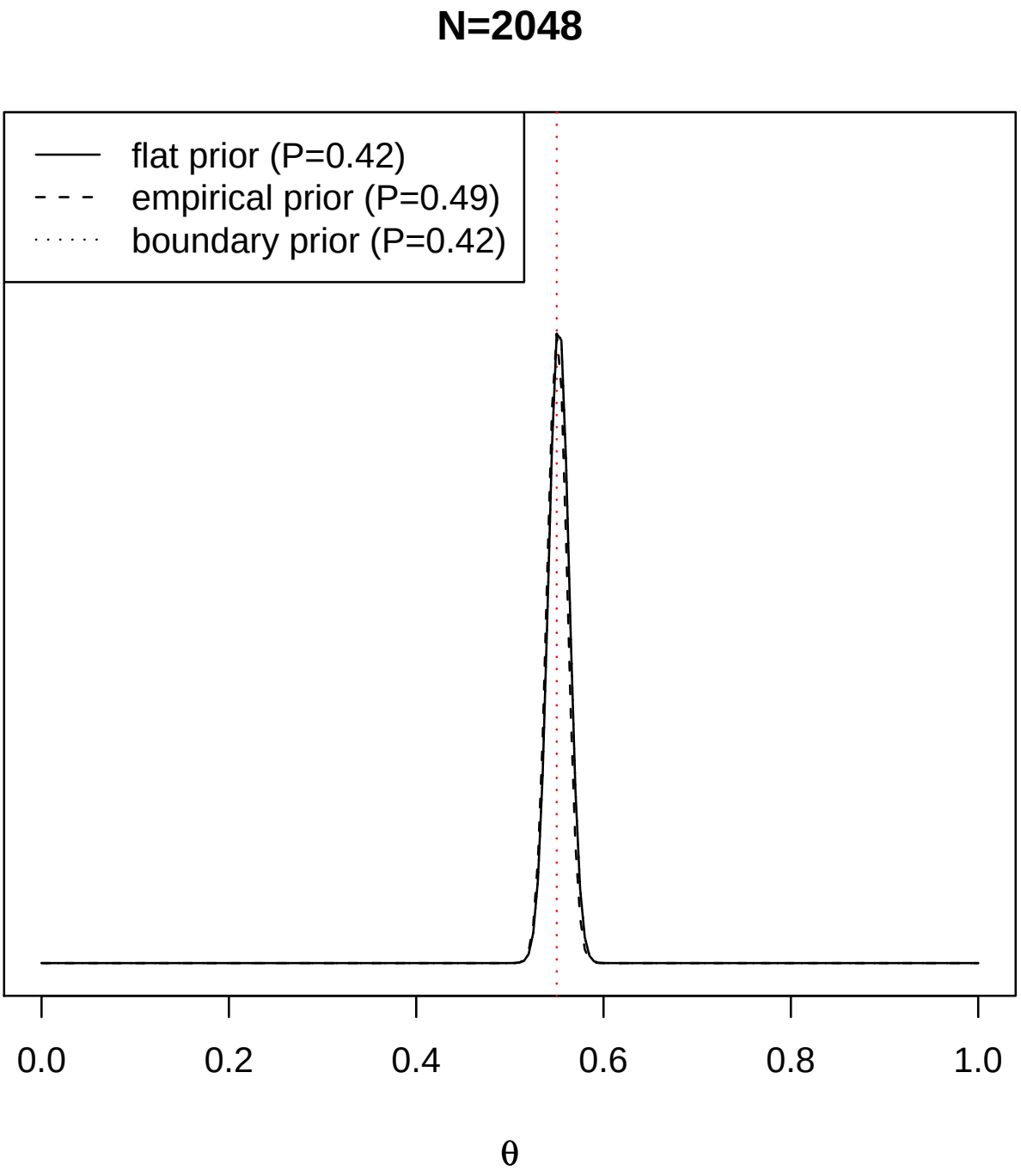

N=64
flat prior (P=0.86)
empirical prior (P=0.89)
boundary prior (P=0.85)
0.0
0.2
0.4
0.6
0.8
1.0
θ

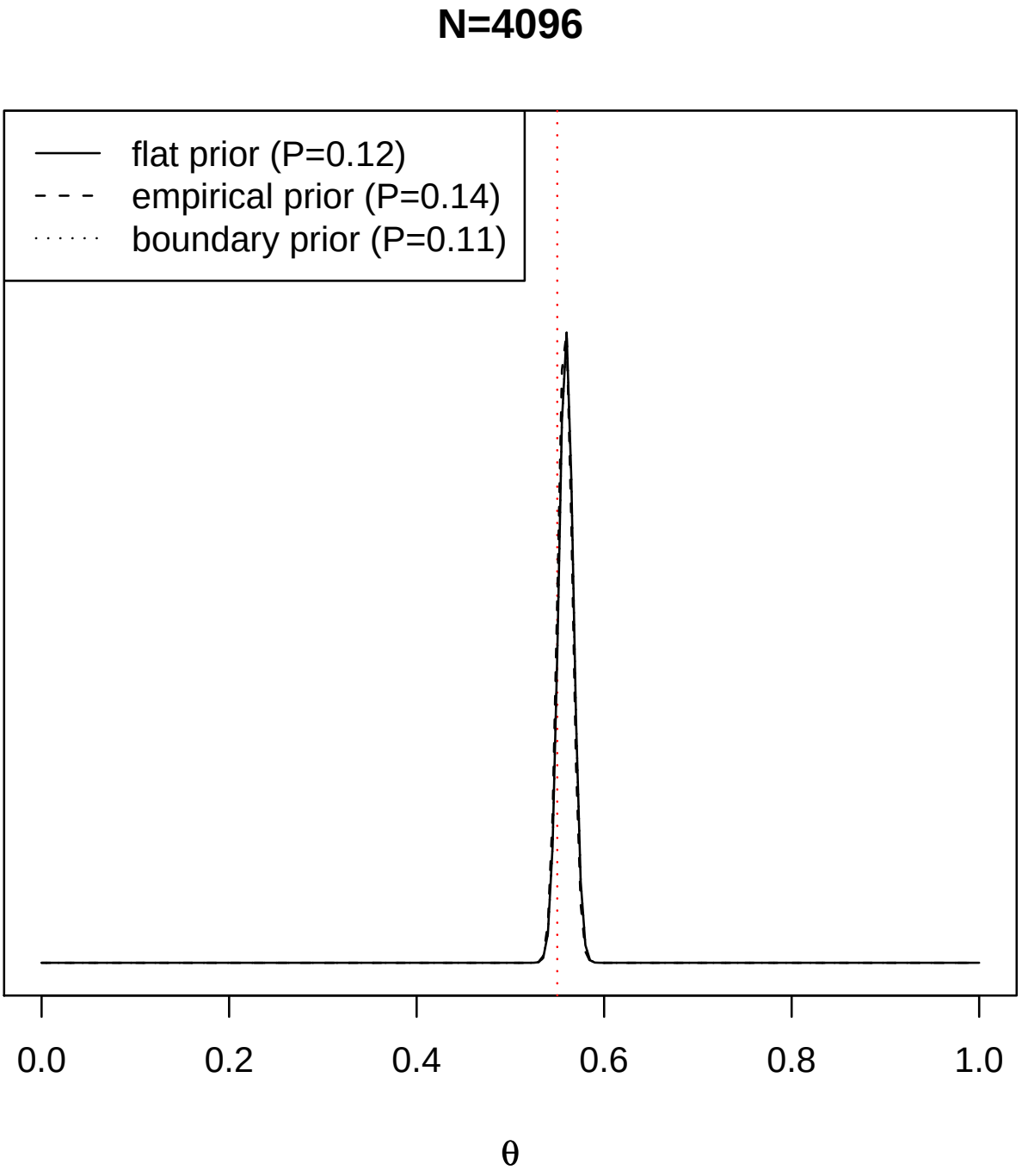

N=128
flat prior (P=0.91)
empirical prior (P=0.93)
boundary prior (P=0.90)
0.0
0.2
0.4
0.6
0.8
1.0
θ


N=256
flat prior (P=0.80)
empirical prior (P=0.87)
boundary prior (P=0.80)
0.0
0.2
0.4
0.6
0.8
1.0
θ


N=512
flat prior (P=0.59)
empirical prior (P=0.70)
boundary prior (P=0.59)
0.0
0.2
0.4
0.6
0.8
1.0
θ


N=1024
flat prior (P=0.36)
empirical prior (P=0.45)
boundary prior (P=0.36)
0.0
0.2
0.4
0.6
0.8
1.0
θ

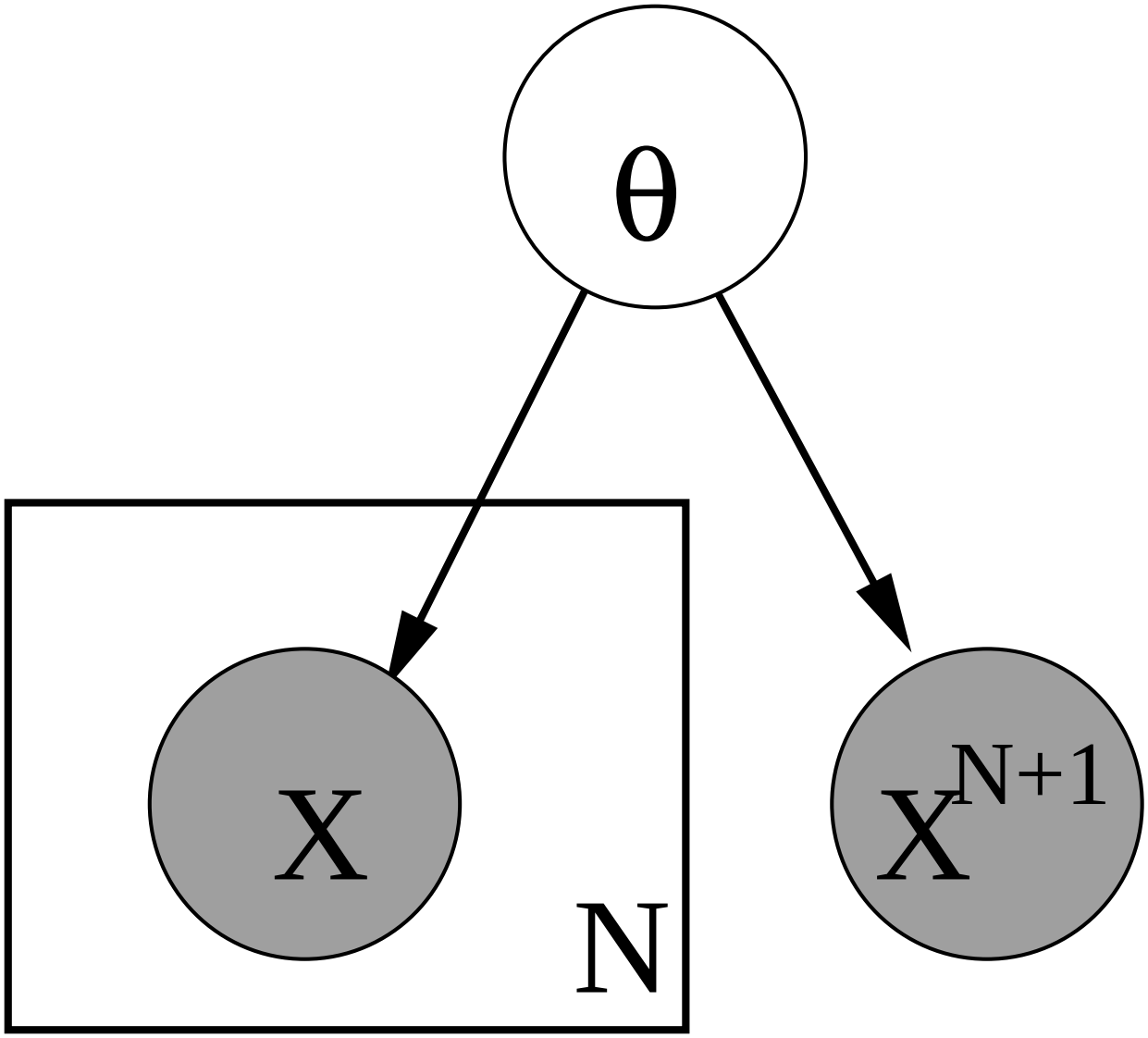

N=2048
flat prior (P=0.42)
empirical prior (P=0.49)
boundary prior (P=0.42)
0.0
0.2
0.4
0.6
0.8
1.0
θ



Predictions from the Posterior Probability Density

- Task: predict probability of x^{N+1}, given N observations in $\mathcal{X}$.

- Marginalizations:
  - $p(\mathcal{X}, \theta) = \int dx^{N+1} p(x^{N+1}, \mathcal{X}, \theta) = p(\mathcal{X} \mid \theta) p(\theta)$.
  - $p(\mathcal{X}) = \int d\theta p(\mathcal{X}, \theta) = \int d\theta p(\mathcal{X} \mid \theta) p(\theta)$.
  - $p(x^{N+1}, \mathcal{X}) = \int d\theta p(x^{N+1}, \mathcal{X}, \theta) = \int d\theta p(x^{N+1} \mid \theta) p(\mathcal{X} \mid \theta) p(\theta)$.
- Posterior: $p(\theta \mid \mathcal{X}) = p(\mathcal{X}, \theta)/p(\mathcal{X})$.
- Predictor for new data point: $p(x^{N+1} \mid \mathcal{X}) = p(x^{N+1}, \mathcal{X})/p(\mathcal{X}) = \int d\theta p(x^{N+1} \mid \theta) p(\mathcal{X}, \theta)/p(\mathcal{X}) = \int d\theta p(x^{N+1} \mid \theta) p(\theta \mid \mathcal{X})$.



Joint distribution ($\mathcal{X} = \{x^t\}_{t=1}^N$): $p(x^{N+1}, \mathcal{X}, \theta) = p(x^{N+1} \mid \theta) p(\mathcal{X} \mid \theta) p(\theta)$.

Point Estimators

- The posterior $p(\theta \mid \mathcal{X})$ represents our best knowledge.
- Predictor for new data point: $p(x^{N+1} \mid \mathcal{X}) = \int d\theta p(x^{N+1} \mid \theta) p(\theta \mid \mathcal{X})$.
- The calculation of the integral may be infeasible.
- Estimate θ by $\hat{\theta}$ (or posterior by $p(\theta \mid \mathcal{X}) \approx \delta(\theta - \hat{\theta})$) and use the predictor

$$p(x^{N+1} \mid \mathcal{X}) \approx p(x^{N+1} \mid \hat{\theta}).$$

Estimators from the Posterior

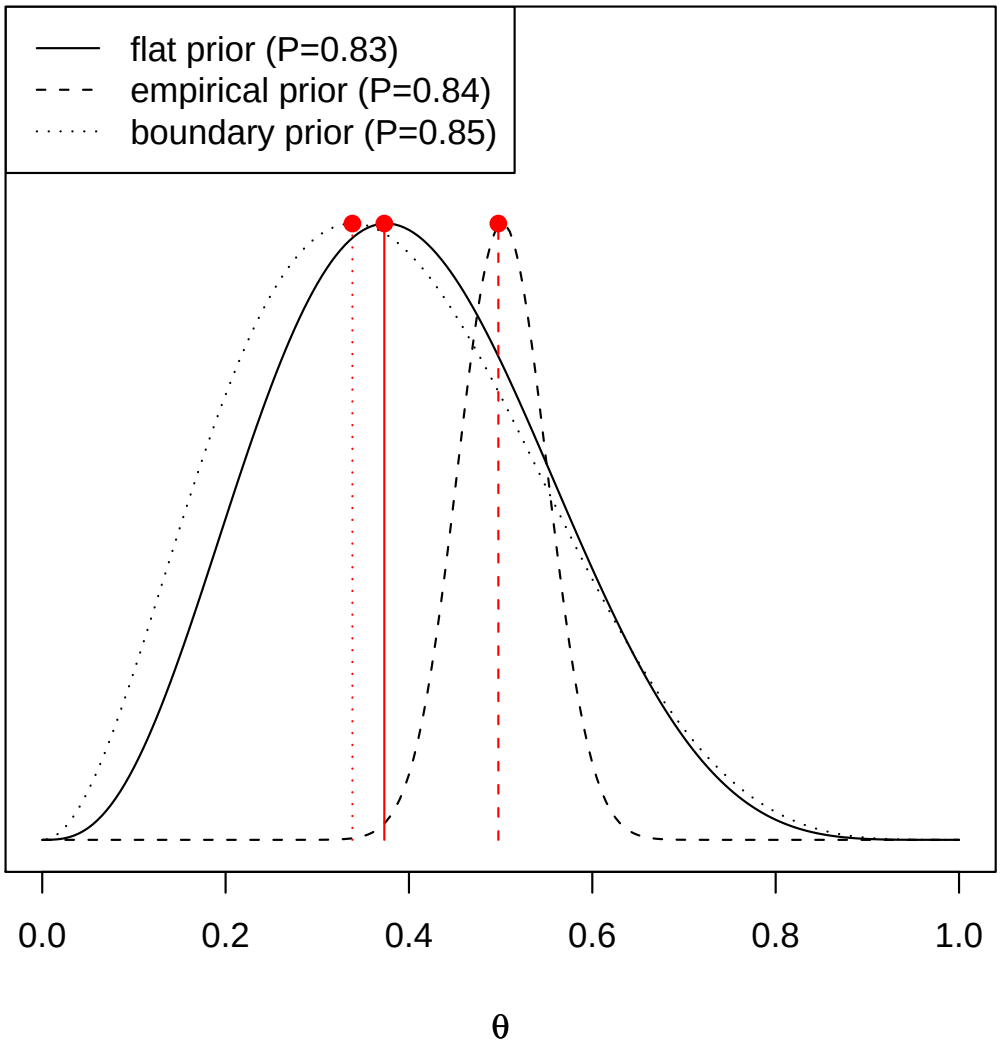
Definition 16 (Maximum Likelihood Estimate)**.**

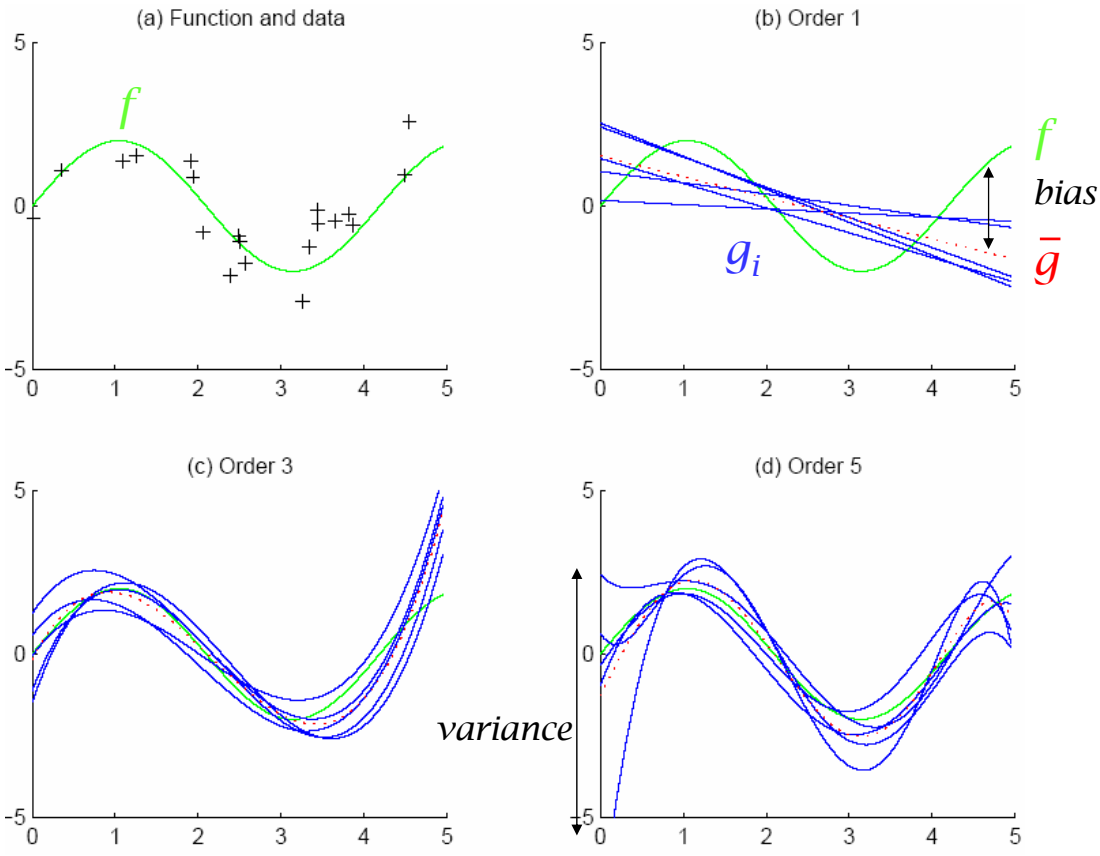
$$\hat{\theta}_{ML} = \arg\max_{\theta} \log p(\mathcal{X} \mid \theta).$$

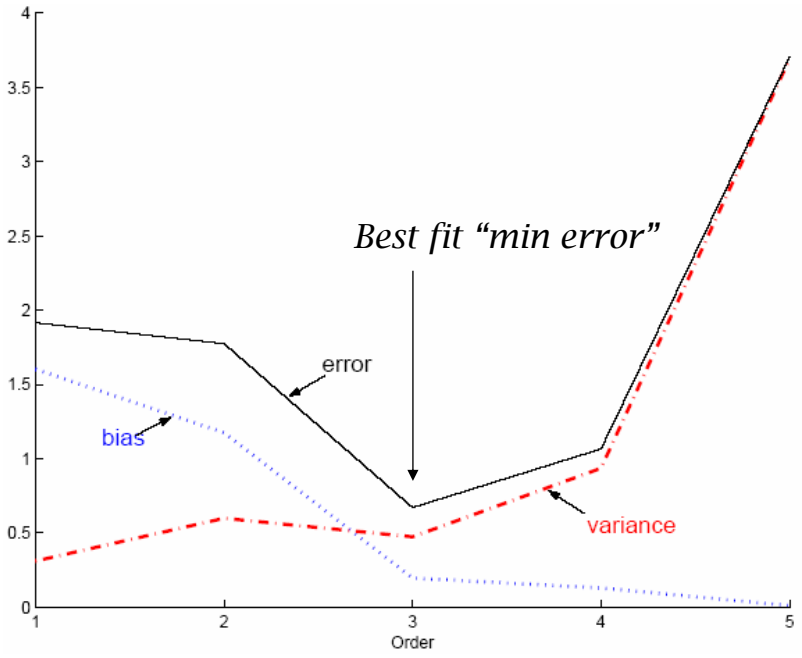
Definition 17 (Maximum a Posteriori Estimate)**.**

$$\hat{\theta}_{MAP} = \arg\max_{\theta} \log p(\theta \mid \mathcal{X}).$$

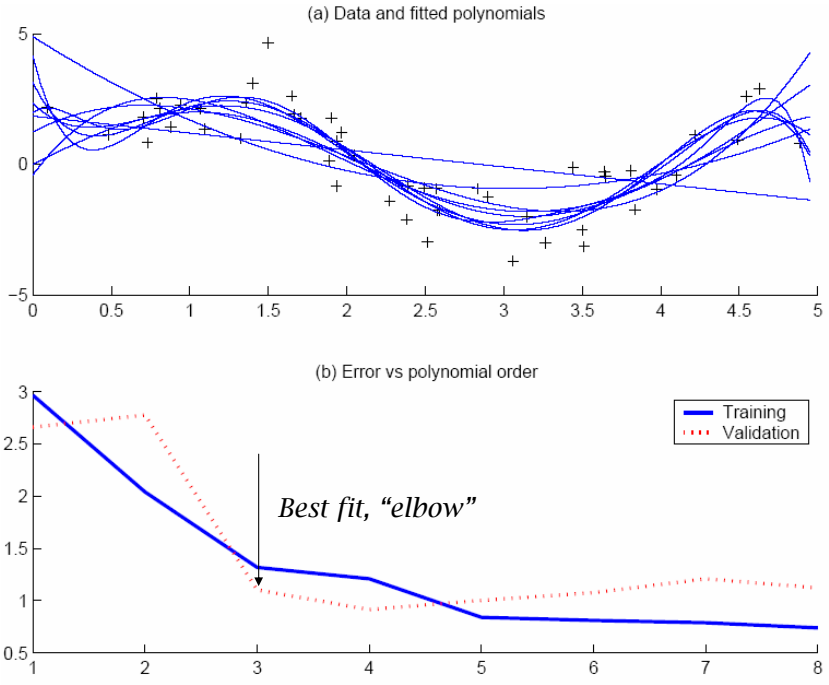
Maximum a Posteriori Estimate (N=8)




(a) Function and data
(b) Order 1
(c) Order 3
(d) Order 5
f
g_i
f
bias
$\bar{g}$
variance



Polynomial Regression



Naive Bayes Classifier

- Idea: the means are class-specific, covariance matrix Σ is common and diagonal (*Naive Bayes*).
- d parameters in the covariance matrix.
- Discriminant is linear: $g_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + w_{i0}$, where $\mathbf{w}_i = \Sigma^{-1}\mu_i$ and $w_{i0} = -\frac{1}{2}\mu_i^T\Sigma^{-1}\mu_i + \log P(C_i)$.

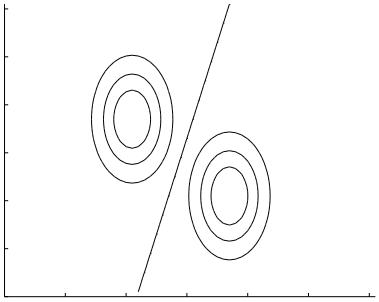
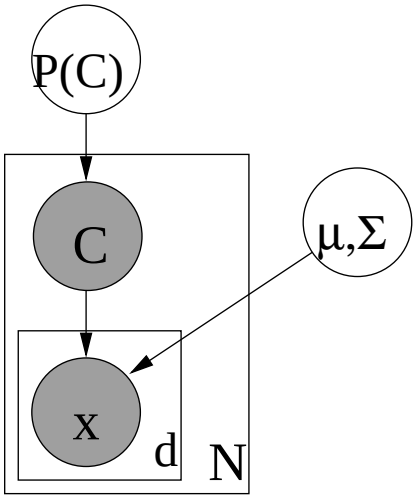


Figure 5.5: All classes have equal, diagonal covariance matrices but variances are not equal. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*



- Cross-validation: most robust if there is enough data.
- Structural risk minimization (SRM): used, for example, in support vector machines (SVM).
- Bayesian model selection: use prior and Bayes' formula.
- Minimum description length (MDL): can be viewed as MAP estimate.
- Regularization: add penalty term for complex models (can be obtained, for example, from prior).
- Latter four methods do not strictly require validation set (at least if implicit modeling assumptions are satisfied, such as that in Bayesian model selection the data is from the model family; it is always a good idea to use a test set) and latter three are related.
- There is no single best way for small amounts of data (your prior assumptions matter).

Subset Selection

- There are 2^d subsets of d features
- Forward search: Add the best featu
  - Set of features F initially Ø.
  - At each iteration, find the best new fea
    $$j = \text{argmin}_i E(F \cup x_i)$$
  - Add x_j to F if $E(F \cup x_j) < E(F)$

- Hill-climbing $O(d^2)$ algorithm
- Backward search: Start with all feat one at a time, if possible.
- Floating search (Add k, remove l)

Principal Component Analysis (PCA)

- Observation: covariance matrix of $\{\mathbf{z}^t\}_{t=1}^N$ is a diagonal matrix D whose diagonal elements are the variances.

$$\begin{aligned} S_{\mathbf{z}} &= \sum_t \mathbf{z}\mathbf{z}^T/N = \sum_t C^T \mathbf{y}\mathbf{y}^T C/N \\ &= C^T \left(\sum_t \mathbf{y}\mathbf{y}^T/N \right) C = C^T S C = D, \end{aligned}$$

  where the diagonal elements of D are the variances $D_{ii} = \sigma^2_{\mathbf{z}i}$.

- Eigenvalues $\lambda_i \Leftrightarrow$ variances σ_i^2.

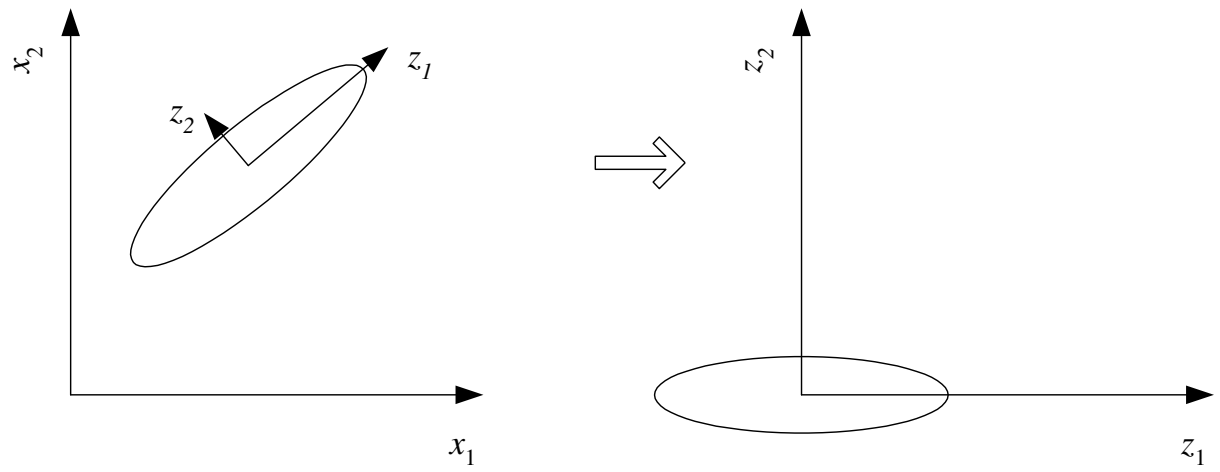
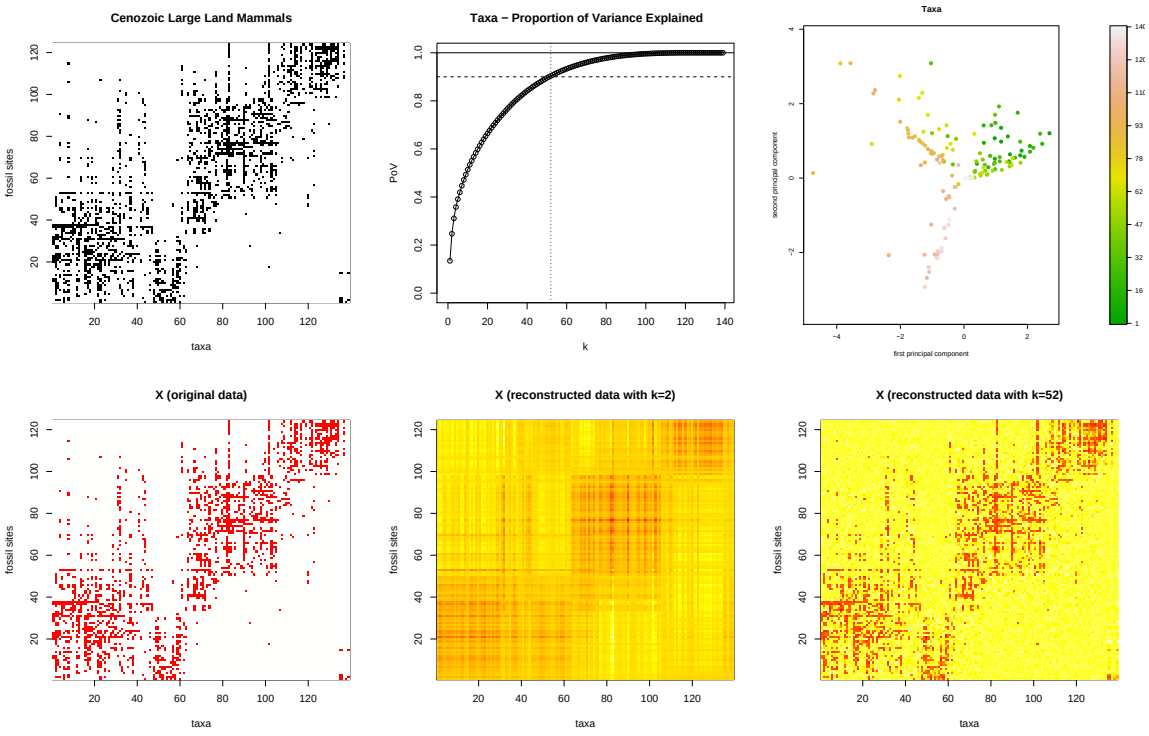


Figure 6.1: Principal components analysis centers the sample and then rotates the axes to line up with the directions of highest variance. If the variance on z_2 is too small, it can be ignored and we have dimensionality reduction from two to one. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

Example: Fossils

- Large European land mammals: 124 fossil find sites (dated 23–2 million years old), 139 taxa
- Reconstruction of site vectors given PCA taxon representation for different k: $\hat{\mathbf{y}} = W\hat{\mathbf{z}} = WW^T\mathbf{y}$, or $\hat{\mathbf{x}} = WW^T(\mathbf{x} - \mathbf{m}) + \mathbf{m}$.



Linear Discriminant Analysis (LDA)

- PCA is unsupervised method (class information is not usually used).
- Linear Discriminant Analysis (LDA) is *supervised* method for dimensionality reduction in classification problems.
- As PCA, LDA can be accomplished with standard matrix algebra (eigenvalue decompositions etc.). This makes it relatively simple and useful.
- PCA is a good general purpose dimensionality reduction method, LDA is a good alternative if we want to optimize the separability of classes in a specific classification task, and are happy with dimensionality of less than the number of classes ($k < K$).

Linear Discriminant Analysis (LDA)

- Find a low-dimensional space such that when $\boldsymbol{x}$ is projected, classes are well-separated.
- Find $\boldsymbol{w}$ that maximizes

$$J(\boldsymbol{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$

$$m_1 = \frac{\sum_t \boldsymbol{w}^T \boldsymbol{x}^t r^t}{\sum_t r^t} \qquad s_1^2 = \sum_t \left(\boldsymbol{w}\right.$$

k-means Clustering

LLOYDS($\mathcal{X}$,k) {Input: $\mathcal{X}$, data set; k, number of clusters. Output: $\{\mathbf{m}_i\}_{i=1}^k$, cluster prototypes.}
Initialize $\mathbf{m}_i$, $i = 1, \ldots, k$, appropriately for example, in random.
repeat
  for all $t \in \{1, \ldots, N\}$ **do** {E step}

$$b_i^t \leftarrow \begin{cases} 1 & , \quad i = \arg\min_i ||\mathbf{x}^t - \mathbf{m}_i|| \\ 0 & , \quad \text{otherwise} \end{cases}$$

end for
for all $i \in \{1, \ldots, k\}$ **do** {M step}

$$\mathbf{m}_i \leftarrow \frac{\sum_t b_i^t \mathbf{x}^t}{\sum_t b_i^t}$$

end for
until the error $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X})$ does not change
return $\{\mathbf{m}_i\}_{i=1}^k$

k-means Clustering

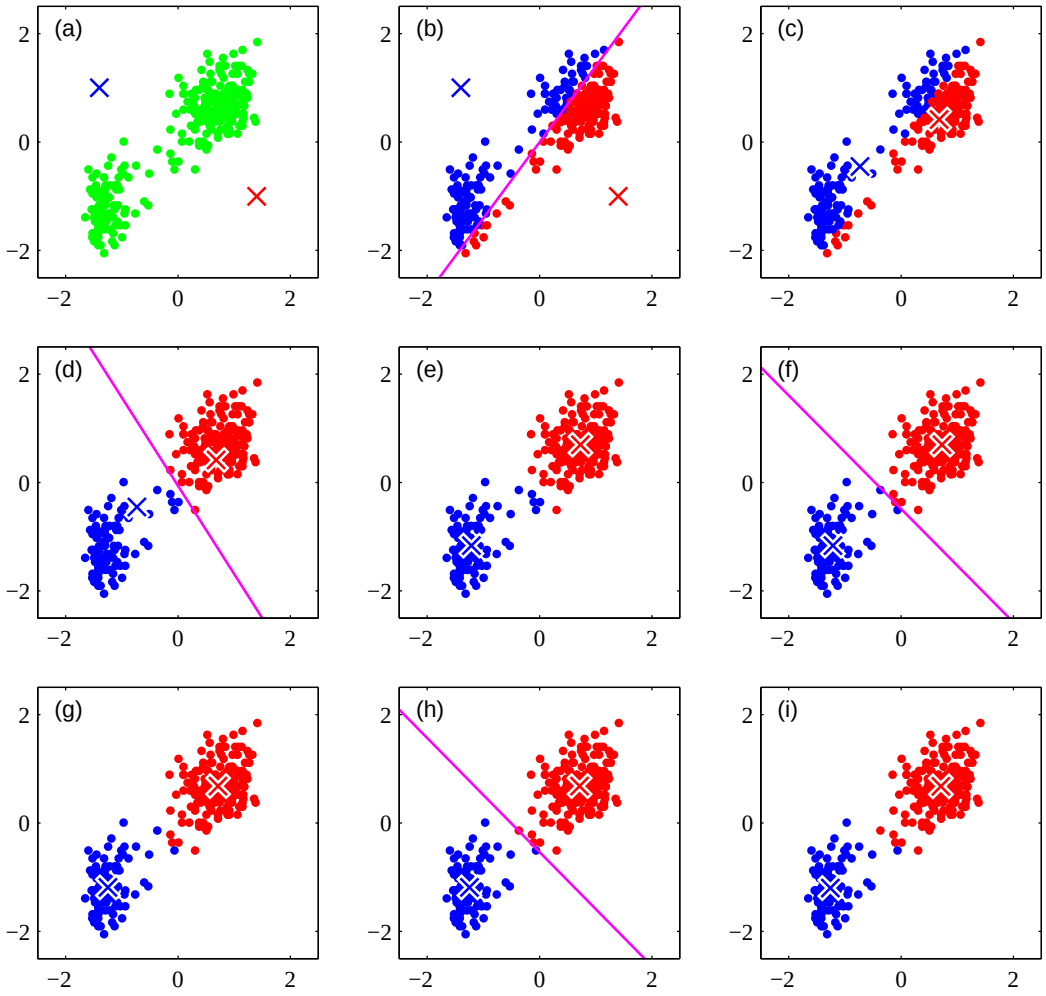
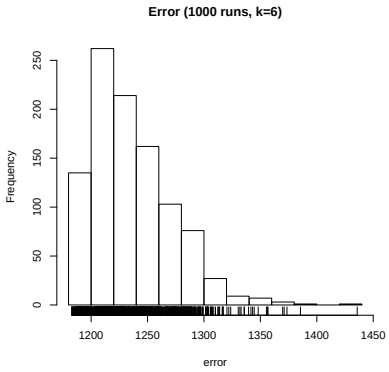


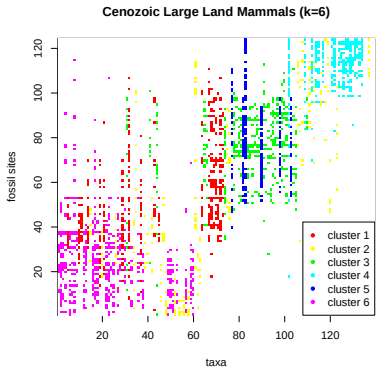
Figure 9.1 of Bishop (2006)

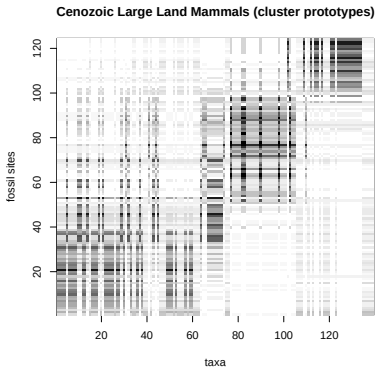
k-means Clustering

- Example: cluster taxa into $k = 6$ clusters 1000 times with Lloyd's algorithm.
- The error $\mathcal{E}(\{\mathbf{m}_i\}_{i=1}^k \mid \mathcal{X})$ is different for different runs!
- You should try several random initializations, and choose the solution with smallest error.

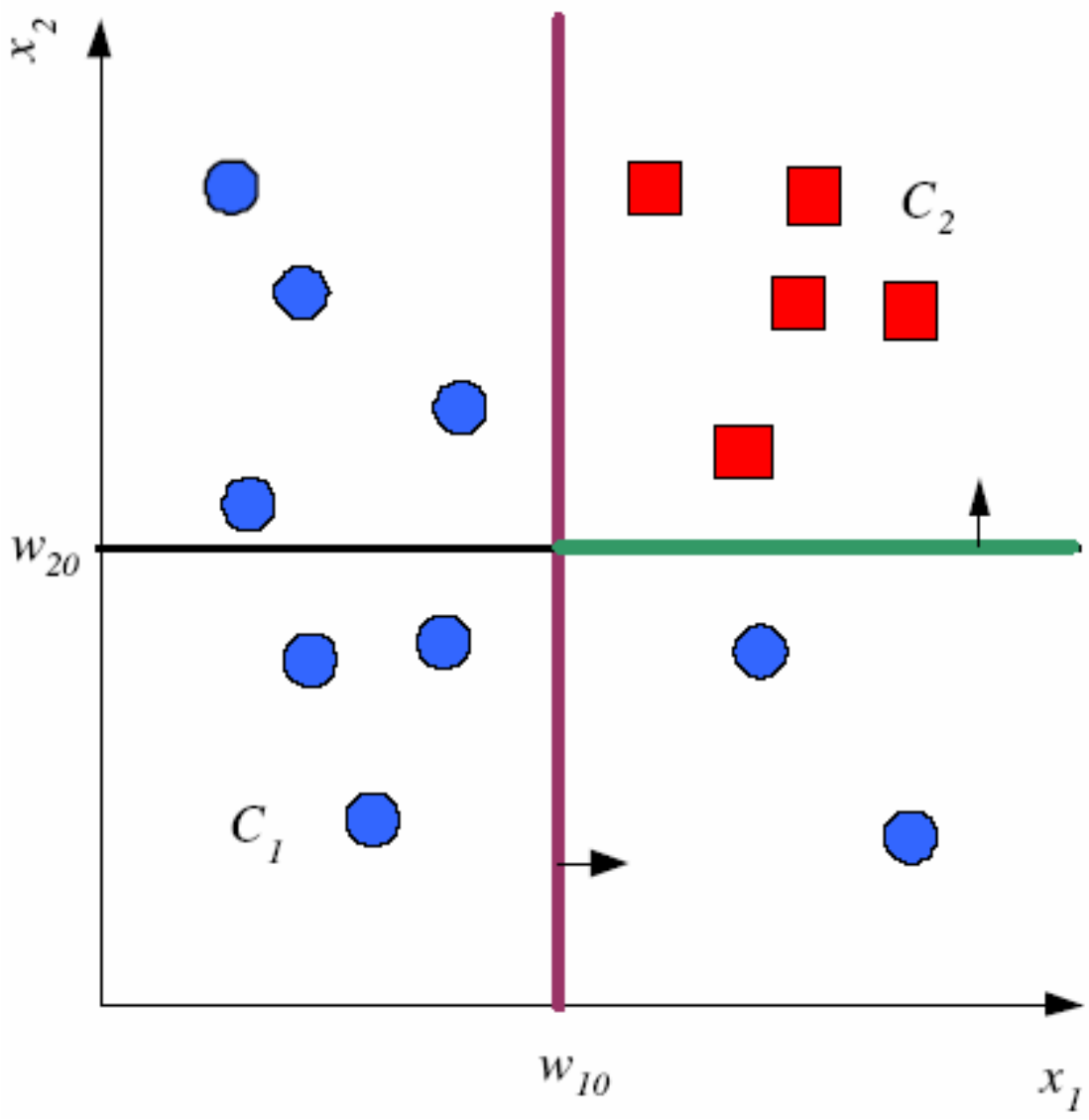
- For a cool initialization see Arthur D, Vassilivitskii S (2006) k-means++: The Advantages of Careful Seeding.







Decision Trees

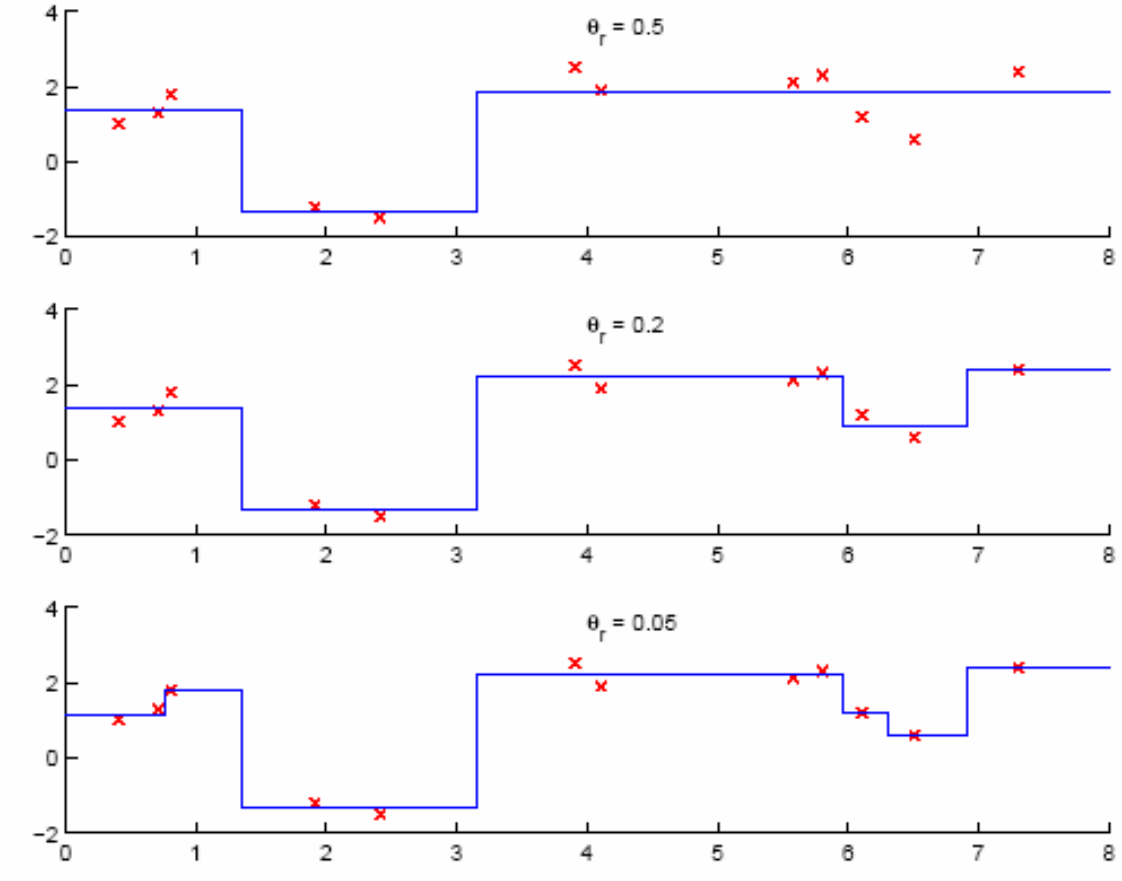


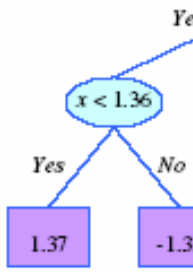
ID3 algorithm for discrete attributes

ID3($\mathcal{X}$) {Input: $\mathcal{X} = \{(r^t, \mathbf{x}^t)\}_{t=1}^N$, data set with binary attributes $r^t \in \{-1, +1\}$ and a vector of discrete variables $\mathbf{x}^t$. Output: T, classification tree.}
Create *root* node for T
If all items in $\mathcal{X}$ are positive (negative), return a single-node tree with label "+" ("-")
Let A be attribute that "best" classifies the examples
for all values v of A **do**
  Let $\mathcal{X}_v$ be subset of $\mathcal{X}$ that have value v for A

if $\mathcal{X}_v$ is empty **then**
Below the root of T, add a leaf node with most common label in $\mathcal{X}$
else
Below the root of T, add subtree ID3($\mathcal{X}_v$)
end if
end for
return T

Model Selection in Trees:





Cost Function for Logistic Regression

- $$P(R \mid X, W) = \prod_{t=1}^{n} P(r^t \mid \mathbf{x}^t, W)$$
- $$\mathcal{L} = -\log P(R \mid X, W) = -\sum_{t=1}^{N} \left(r^t \log y^t - (1 - r^t) \log (1 - y^t) \right),$$

  where $y^t = P(r^t = 1 \mid \mathbf{x}) = \text{sigmoid}(\mathbf{w}^t \mathbf{x} + w_0)$.
- Task: find $W = (\mathbf{w}, w_0)$ such that $\mathcal{L}$ is minimized.
- No EM etc. algorithm. Use gradient ascent.

Gradient Ascent

- Logistic regression may converge to $w \to \pm\infty$ (see right), especially when data is high dimensional and sparse. This causes problems.
- Solution: minimize regularized cost $\mathcal{L} \to \mathcal{L} + \frac{1}{2}\lambda \left(w_0^2 + \mathbf{w}^T \mathbf{w} \right)$.

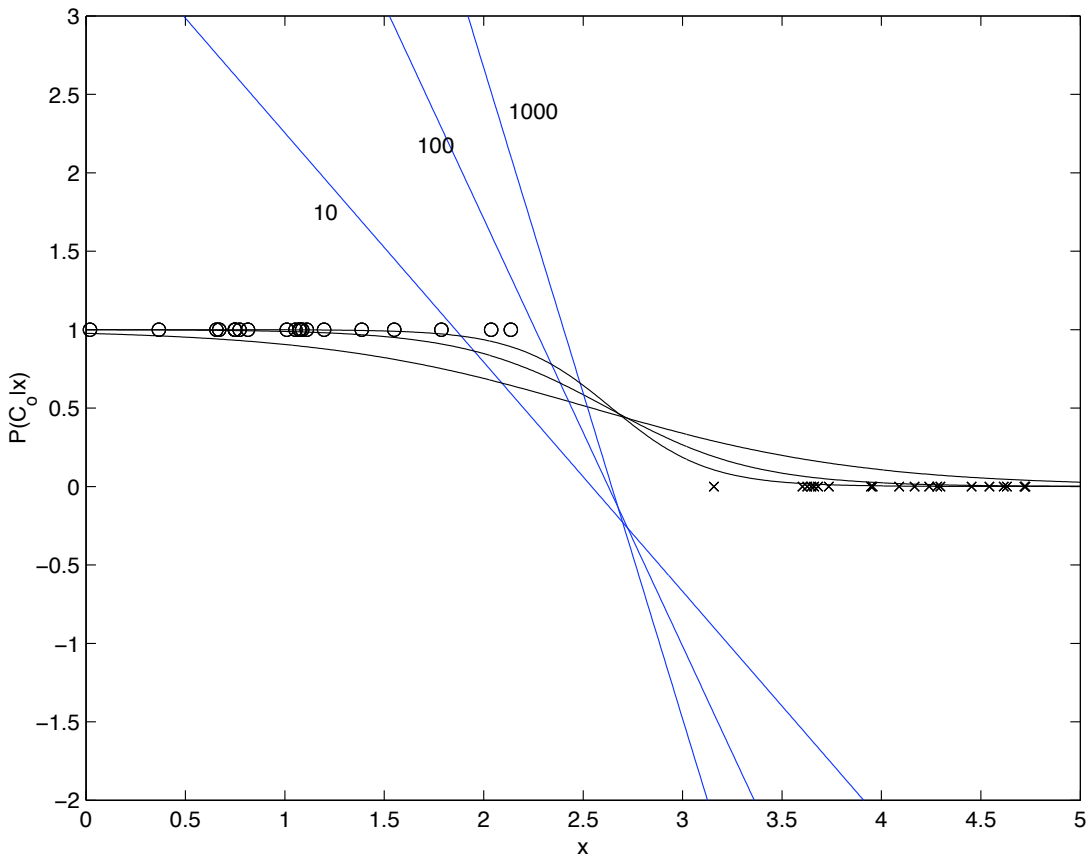


Figure 10.7: For a univariate two-class problem (shown with '$\circ$' and '$\times$'), the evolution of the line $wx + w_0$ and the sigmoid output after 10, 100, and 1,000 iterations over the sample. *From: E. Alpaydın. 2004.* Introduction to Machine Learning. ©*The MIT Press.*

The End
(Some overflow slides follow.)

28 Overflow

28.1 Optimization Algorithms

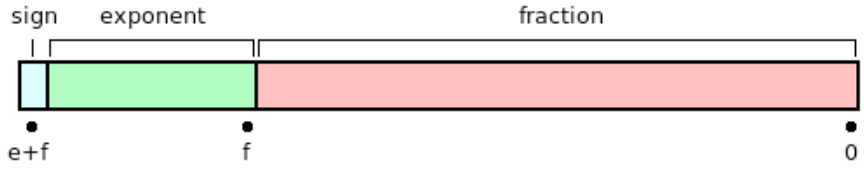
Algorithms in Machine Learning

- Many (most?) machine learning algorithms problems can be stated as optimization problem: "Find parameters θ such that the cost $\mathcal{L}(\theta)$ is minimized."
- Earlier in the course:
  - Some optimization problems can be solved in polynomial time (e.g., PCA)
  - In some optimization problems (typically they are NP-hard) one must use approximation algorithms, such as greedy search. (e.g., Lloyd's algorithm in kmeans clustering).
- Issues to take into account:
  - What is the time and memory complexity?
  - How is data accessed (for large data sets, serial access is fastest)
  - Does the algorithm find a reasonable solution (is there approximation ratio?)
  - Could there be a better greedy optimization step?
  - Is your algorithm numerically robust? (That is, does it work consistently and give accurate results for every possible input.)
- Making numerically robust algorithms is difficult
- The first rule in numerical computation: always use robust numerical libraries when possible
- Of methods with essentially similar performance, choose the simplest/easiest to understand.

28.2 Computing Sums and Products

IEEE Floating Point Arithmetics

- The floating point numbers are stored in three parts in binary:
  - fraction ($f = 52$ bits in double precision)
  - exponent ($e = 11$ bits in double precision)
  - sign (1 bit)
- This includes the following types of numbers:
  - normalized numbers (normal non-zero numbers)
  - zero (± 0)
  - infinities ($\pm\infty$)
  - NaN
  - denormalized numbers ($\pm$ something very small or very large)



The three fields in an IEEE 754 float.

Image by Charles Esson, GFDL.

Numerical Computation: Computing Sums and Products

- Sometimes it is enough to use + and * operators to compute sums and products. According to R: 3.14*42=131.88; 3.14+42+5=50.14.
- Sometimes it is not. According to R: 3.14e-200*42e-201*1e300=0; 1e-400*1e400=NaN; 1e-16+1-1=0.
- In probabilistic modeling it is typical to...
  - Have numbers of different orders of magnitudes, including very small numbers.
  - Do sums and products with them.
- Important numbers (examples from the R floating point implementation in Mac OS X, `help(.Machine)`):
  - Smallest positive floating point number ϵ (*machine epsilon*) for which $1 + \epsilon \neq 1$: 2.2×10^{-16}.
  - The largest finite floating point number: 1.7×10^{308}.
  - The smallest positive floating point number: 2.2×10^{-308}.

Numerical Computation: Representing Numbers

- In many practical applications, 2.2×10^{-308} is too large for representing intermediate probabilities.
- Solution: store numbers as logs.
- Probabilities are usually always positive. (Generally, software should however be written so that to work consistently also with zero probabilities.)
- R is consistent also for zero probabilities: log(0)=-Inf; exp(-Inf)=0.
- Other software may behave differently. Read the documentation and test.

Numerical Computation: Computing Products

- Task: compute the product $y = \prod_{i=1}^{n} x_i$.
- 1e-200*1e-200*1e300=0 (wrong!).
- Solution: use logs.

- $\log y = \sum_{i=1}^{n} \log x_i$.
- log(1e-200)+log(1e-200)+log(1e300)=log(1e-100) (correct).
- Division: $\log(x/y) = \log x - \log y$. Product with negatives.

Numerical Computation: Computing Sums

- Task: compute sum $y = \sum_{i=1}^{n} x_i$.
- exp(-1000)+exp(-999)=0 (wrong!).
- Solution: scale numbers appropriately before doing the sum.
- $\log y = \log x_{MAX} + \log\left(\sum_{i=1}^{n} \exp\left(\log x_i - \log x_{MAX}\right)\right)$, where $\log x_{MAX} = \max_i \log x_i$.
- -999+log(exp(-1)+exp(0))=-998.6 (correct).
- Something like this: `safesum <- function(x) { xmax <- max(x) ; xmax+log(sum(exp(x-xmax)))) }`

Numerical Computation: Example

$$P(C_i \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid C_i)P(C_i)}{\sum_{k=1}^{K} P(\mathbf{x} \mid C_k)P(C_k)} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$$

Store numbers as logs and denote: $a[i] = \log P(\mathbf{x} \mid C_i)$, $b[i] = \log P(C_i)$.

```
safesum <- function(x) { xmax <- max(x); xmax+log(sum(exp(x-xmax)))) }

evidence <- safesum(a+b)

posterior <- sum(c(a[i],b[i],-evidence))

exp(posterior) #P(C_i | x)
```

28.3 Validation and Cross-Validation

Evaluating Classification Algorithms

- Questions:
  - What is the performance of a classification algorithm on unseen data?
  - Which of the two (or more) classification algorithms is better?
- Our results are conditioned on the data set. (In fact, for all algorithms there exists data sets for which it would perform excellently or poorly, No Free Lunch Theorem, Wolpert 1995.)
- Limited amount of training/validation data makes it difficult
  - Choose the model complexity.
  - Evaluate the results.

K-Fold Cross-Validation

- How to use the training/validation data most efficiently?

CV($\mathcal{X}$,A,K){Input: $\mathcal{X}$, data $\mathcal{X} = \{(r^t, x^t)\}_{t=1}^N$; A, classification algorithm; K, number of folds. Output: $\mathcal{E}$, error measure.}
Partition $\mathcal{X}$ in random into K roughly equally sized parts $\mathcal{X}_i$.
for all $i \in \{1, \ldots, K\}$ **do**
  Train A using $\mathcal{X} \setminus \mathcal{X}_i$ as a training set.
  Let $\mathcal{E}_i$ be the error of A in $\mathcal{X}_i$ (for example, the fraction of incorrectly labeled items).
end for
$\mathcal{E} \leftarrow \sum_{i=1}^K |\mathcal{X}_i| \, \mathcal{E}_i / |\mathcal{X}|$
return $\mathcal{E}$ {$\mathcal{E}$ can be used as a validation set error in model selection.}