T-61.5040

Learning Models and Methods, (5 cr) L

Spring 2007

Lecturer: **Petteri Pajunen**

Course Assistant: **Ville Viitaniemi**

Lecture 1: 17.1.2007

General Matters
Introduction
Can we Learn From Data?

Times, Dates, Locations

- lectures: wed 12-14 in T4, from 17.1.2007 given by Petteri Pajunen
- exercises: fri 10-12 in T3, from 19.1.2007 given by Ville Viitaniemi
- contact: t615040@mail.cis.hut.fi (lecturer and assistant)
- office hour: lecturer on wed 14-15, B309

Course Material

- no textbook: lecture slides and possibly some other material will appear on course homepage as the course proceeds:
  `http://www.cis.hut.fi/Opinnot/T-61.5040/`
- exercise problems will be available on the website before each exercise session
- solutions will be posted on the website as the course proceeds

- a major topic of the course is *Bayesian Inference*
- for more in-depth information on Bayesian Inference, one of the following books may be useful:
  1. Gelman et al: Bayesian Data Analysis, CRC Press, 1996 (or 2nd ed Chapman & Hall/CRC 2003)
  2. Bernardo and Smith: Bayesian Theory, Wiley, 2000
- these are *not required reading* and do not contain even half of the topics on the course

Prerequisites

- prerequisites are basic math courses, especially probability, matrices and calculus
- some course in modeling data nonlinearly (for example neural networks or pattern recognition) would be helpful
- some programming skills are needed for the computer assignment

Passing the Course

- final exam+compulsory assignment
- first final exam in May 2007, everyone may take this
- later exams can be taken only after your computer assignment has been accepted

Computer Assignment

- instructions will appear on the website by the end of March
- datasets and "technical support" are available in MATLAB and R (`www.r-project.org`)
- but you can use other software/language if you wish
- deadline will be around May/June (announced later)

Exercises

- some (or many) exercise problems are difficult and/or not meant to be solved based on what has been presented at the lectures
- these are intended to illustrate some important ideas in more detail than is possible in the lectures
- these are marked “demo” in the problem sheet
- other problems are more or less possible to be solved by yourself, though they may also be difficult...

Introduction

- emphasis is on presenting fundamental ideas through simple examples and counterexamples
- no overly complicated derivations of theoretical results
- however, somewhat complicated calculations are unavoidable when applying Bayesian Inference
- most skipped details can be found in the two books mentioned before, or from references that will be posted on the website

- don't take the course if you want
  - lots of algorithms to stick data into without thinking
  - rigorous derivations of theoretical results
- do take the course if you want
  - to understand the learning problem (have data, what to do with it?)
  - to understand Bayesian methods at a general level

- *learning*: for our purposes, using a given set of observed data to get information about unobserved quantities
- for example, predicting something or making an optimal decision
- this definition of learning covers most situations where one would like to learn from data in practice
- note that it does not contain algorithms that represent the data in another way (Fourier transform, PCA etc..), although these algorithms can be useful as a part of a learning method

- example: where to sit in a lecture hall?
  - first you try the front row and find that the temperature is too cold
  - next week you try the last row: now the temperature is too high
  - where do you try to sit next week?
  - you have data, and are trying to make an optimal decision using it

- *model*: represents the information we have about the learning problem
- in the lecture hall example, most people have a model which might be described as "temperature changes slowly as a function of position"
- without a model, you might as well choose the next seat randomly
- we will see that learning from a given set of data without a model (i.e. without assuming anything) is not possible in any meaningful way

- modeling data has been done in statistics for a long time
- but traditionally the lack of computing power has resulted in:
  - realistic but unsolvable models, or
  - computable, simple standard models (usually unrealistic)
- one can run into complicated models quite easily:
  - missing data (parts of data are missing in various ways)
  - mixture models (discussed in a later lecture)

- emerging trend: increasingly complex models for which approximate solutions can be computed (neural networks, data mining etc...)
- often these models are “generic” (not constructed for a specific problem and are often used to solve a wide variety of problems)
- this approach represents a compromise between a realistic unsolvable model and a solvable but too simple model
- reasons for not using the problem-specific correct model may include computational constraints, not enough data, model too difficult to solve etc.

- some wrong intuitions:
  - a complex model and lots of data solves all problems (not so)
  - a model and data solves decision problems (not so)
  - results of learning from data are objective (not so)
  - some learning methods are better on average than others, assuming nothing about the problems they are applied to (not so, in a sense to be made clear later)

- in this course, we deal with the problem of learning from a *given set of data*
- in practice, other steps are needed that are often problem specific
- e.g. what data to collect, how to preprocess it etc..
- sometimes the outcome of system is not unique: for example, fault detection requires deciding how the different faults are defined
- most of these extra steps in learning require decision theory

- we end up using *probabilistic modeling*
- suppose we look at a *deterministic* system that gives always the output y_i when given the input x_i
- even in this case there are good reasons to use probabilities
- first, the system may contain variables that we do not observe

- even if the world works in a deterministic way, one can only model a tiny part of it relevant to the problem at hand
- examples:
  - flipping a coin: measuring the initial speed, direction, and rotation of the coin, and predicting where it will land is too difficult
  - the change in direction of a moving car: speed and steering wheel position explain it well, but is also affected by changes in speed, surface friction and unevenness etc.
- therefore we will end up using probabilistic models

Causality

- dependence between inputs and outputs can be one of the following type:
  - output caused by input (steering wheel position $\rightarrow$ direction of car)
  - input caused by output (symptoms $\rightarrow$ illness; the arrow denotes the system, not causality)
  - input and output caused by something else (consumption of soft drinks, number of people drowning; both are probably caused by warm weather)

  - noncausal dependence

- cause of dependence, or anything else that is unobserved, cannot be learnt from given data unless one is willing to use a model containing the causality as a parameter

- without such a model, learning methods can be used to predict unknown outputs. This does not require causality.

Summary

- no long derivations of theoretical results, no black box tools
- demonstrate the impossibility of learning from data without assumptions
- can we learn from data by assuming very little?
- develop probabilistic approach to learning (Bayesian Inference)
- examine basic properties of Bayesian Inference
- does Statistical Learning Theory say that one can

learn without assumptions?

- apply Bayesian Inference in various situations
- how to compute approximate solutions
- missing data
- latent variable models
- Gaussian processes
- making decisions

- goals of the course: to understand
  - the impossibility of learning without assumptions
  - the difficulties of high dimensional problems
  - in what sense probabilistic learning is correct
  - how to think about problems using probabilities
  - how to do BI in practice
  - the need to approximate BI, and some methods to implement this
  - how to make decisions and what this has to do with BI

Cannot Learn From Data

- if one wants to learn from data, there must be something to be learned
- define this as θ, *state of nature*, which is *unknown*
- set of all possible states of nature is Θ
- θ can also be a *parameter* of some function or distribution

- for practical reasons, lets assume that θ has a numerical value (can be a vector or a matrix)
- the learning problem is to get information about the value of θ using data (denote here by D)
- can we say something useful about θ *without using any information except data* D?
- when stated this way, it seems obvious that we can't
- but putting data into an algorithm which is unrelated to the problem, or is not known to be related, is just trying to achieve this

- example: predict output y from a given input x (regression)
- $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, $\theta = \tilde{y}$ corresponding to $\tilde{x} \notin \{x_1, \ldots, x_n\}$?
- no-brain solution: pick some learning method A, train it on data D and use it on $\tilde{x}$ to get a predicted value $\tilde{y}$

- you might as well have chosen a different learning method B
- suppose B gives a different prediction $\tilde{y}$ than method A
- this is not learning from data: we are deciding the prediction result as we choose the learning method A or B (or C,D,E,...)
- any learning method contains implicit information that is combined with data

- how about using data to select the learning method?
- this sounds reasonable and is often done (e.g. model selection)
- construct method C so it is identical to A within the training set and identical to B outside of it
- training data cannot distinguish between A and C, so we must pick one and decide the prediction again

- last straw: A is better than B or C in general, so we just use A
- sure, we used information "A is best" in addition to data but this is OK since we can now use A in every problem and do not need additional information
- this sounds good: just find the "best" learning method and be done with it
- researchers have certainly looked for it (cybernetics, neural networks, fuzzy logic, genetic algorithms, kernel methods etc..)

- are some learning methods better than others on average?
- such a method should have a better average performance than, for example, guessing the predicted values
- it sounds easy to come up with methods that beat guessing
- but it is even easier to prove this is impossible, either rigorously or by simple counterexamples

No Free Lunch Theorems

- it can be shown (D. Wolpert, No Free Lunch Theorems) that no information except training data leads to equal average performance over all learning methods
- for proofs, see original article(s) (reference on webpage)
- proofs are somewhat technical so we settle for examples restricted to predicting bits in the exercise problems

- NFL theorems are based on the idea that observed data without any other information does not restrict the unobserved data in any way
- no information means "all solutions equally possible"
- easiest to demonstrate in discrete, finite problems
- consider $\Theta = \{\text{every 4-bit sequence}\}$

- no information $\Longrightarrow$ every 4-bit sequence equally possible
- observe first three bits, 010, and predict the last bit
- obviously the last bit is equally likely either 0 or 1
- easy to generalize to n-bit outputs, which essentially is all that is needed (unless you have an analog computer or enjoy making decisions with infinite number of choices)

- no information was interpreted as having a uniform distribution over Θ (e.g. over 4-bit sequences)
- NFL theorem $\Longrightarrow$ performance of any learning method averaged over all problems is constant
- applies even for guessing and other apparently useless methods!
- intuitively clear, because regardless of training data, the distribution of non-training data is uniform

- performance $=$ average over all $\theta \in \Theta$
- i.e. if method A predicts 1 for the input 010, we count this as 0 (no error) for $\theta = 0101$ and 1 (error) for $\theta = 0100$
- average performance is computed using a uniform distribution over Θ
- one might object that 'real-world' problems have 'structure', they are not uniformly distributed as assumed above

- another NFL theorem answers the non-uniform critique
- assume that the distribution over Θ can be non-uniform
- no information $=$ we don't know this non-uniform distribution
- averaging uniformly over *all distributions over* Θ $\Longrightarrow$ same performance for all learning methods
- interpretation: to learn from data, you have to know something about the structure of the problem, not just that it has some

- NFL-idea seems almost trivial
- it makes clear that no information leads to no learning, regardless how much data one has
- yet much research seems to rely on this possibility
- heuristics such as cross-validation, Occam's razor, Minimum Description Length etc... don't change the conclusion

- in practice one expects to beat guessing
- if the learning method makes use of strong and correct assumptions, this is possible
- but in some areas such as neurocomputing, implicit assumptions seem very weak, such as simply assuming some vague regularity
- weak assumptions might be expected to hold for a large set of learning problems

- does a method A exists which beats guessing (B) on a large set of problems?
- it can be shown that the set of problems where A and B have even a small difference in performance, is *very small* (demo exercise problem)
- all methods are almost as bad as guessing on almost all problems
- expecting a method to work well, one's problem must come from the very small set of problems (a strong assumption)

- summary:
  - NFL thms $\Longrightarrow$ no information, no learning
  - little information $\Longrightarrow$ little benefit from training data
  - good expected performance $\Longrightarrow$ strong, correct assumption (either implicit or explicit)
  - since one must bring a lot of information to the learning problem, it is preferable to know what is being assumed
  - this motivates the study of probabilistic learning methods where the assumptions are made explicitly

- does this bet sound appealing? It should for those who believe that information can be extracted from lots of data
  - I generate 10^6 bits of training data (given to you), and 100 bits of test data (not given)
  - you win 1000 EUR if you predict more than 60 bits of test data correctly
  - otherwise you lose 1000 EUR
  - this bet should be appealing if you believe your learning method has better performance than 0.6
  - hint: $\sum_{k=0}^{60} Bin(k|100, 0.5) \approx 0.982$

Lecture 2: 24.1.2007

Curse of Dimensionality
Overfitting

- NFL theorems $\Longrightarrow$ no information, no learning
- simple, correct model $\Longrightarrow$ easy to solve with enough data
- what about learning methods in practice?
- perhaps some information and lots of data leads to good results?

Weak Assumptions Lead to Local Learning

- lets examine what happens when a learning method is expected to contain "weak" information
- for example, it can be argued that a learning method such as a neural network prefers solutions that are regular in some sense
- but with a complex enough neural network, one can approximate closely almost any solution
- sounds plausible: there is some information (prefer well-behaving solutions) but not too much (many solutions can be closely approximated)

- lets examine how well this approach can be expected to work
- example: regression by kernel density estimation
- predictor of y as a function of x can be solved using the *joint distribution* $p(x, y)$ (exercise)
- the solution minimizes the MSE $\mathrm{E}(|y - \hat{y}|^2)$

- how to estimate $p(x, y)$ from data (x_i, y_i), $i = 1, \ldots, n$
- one way to estimate it is to use a *kernel density estimator*
- a nonnegative localized function with integral $1/n$ is located at each (x_i, y_i)
- the sum of these is the estimate of $p(x, y)$

- replacing $p(x, y)$ by its estimate will give a solution $\hat{y}$ as a function of x and training data
- exact result computed in exercises
- $\hat{y}$ will be a weighted sum of $y_1, \ldots, y_n$
- the weights are large when x_i is close to x and small when x_i is far from x

- the above solution has the property *x is close to* x_i $\Longrightarrow$ *y is close to* y_i
- methods such as neural networks in general have this property
- for example, a large MLP network is a nonlinear parametric function
- it does not vary very rapidly when input x changes, but is often flexible enough to provide a good fit to training points

- this property seems exactly what we wanted, but it is useful only near the training points
- *local learning* is essentially what was achieved
- i.e. the training data cannot be generalized except in a very small neighbourhood around the training points
- methods such as neural networks do predict outputs arbitrarily far from training points, but in general these predictions are not reliable

- lets examine when local learning works
- demo: *local.R*
- the correct function is x^3, and two high-degree polynomials are fitted to the data
- the polynomials don't vary extremely rapidly and can fit the correct solution exactly
- when 50 points are used, the result is quite good

- with 10 points used for learning, the solutions differ from x^3 a lot
- the only common feature is that the polynomials pass through, or close to the training points
- local learning effectively makes the prediction based on the closest point(s) in the set of observed data
- this is useful only when predicting *near enough* to a training point

- lets consider this geometrically
- training inputs $x_1, \ldots, x_n$ are in a d-dimensional space
- we want to predict the output at some point x in the same space
- local learning can be used if x is close to some x_i

- relevant quantities: n (number of training points), d (dimension of x)
- small d, large n: local learning may be useful
- note that the training data must cover well enough the area where predictions are needed

- when d is even moderately large, unexpected things happen
- this phenomenon is called the Curse of Dimensionality
- CoD is a geometric concept which describes certain properties of high-dimensional spaces
- some examples are given to illustrate this

High-Dimensional Data

- lets examine a d-dimensional vector space (or a bounded subset of it), and n points in it
- for local learning to work, the points must be close enough to each other
- curse of dimensionality $\implies$ n must be *very large* if d is even moderately large

Uniform density over data space

- consider $[0,1]^d$, the d-dimensional hypercube
- cover it with points having distance 0.1 to the closest neighbour
- if $d = 1$, then ten points achieves this
- but if $d = 10$, we need about 10^{10} points for the same distance
- ten is not a very high dimension in many learning problems!

Most points are close to the sides

- consider $[0,1]^d$, and $[0.1,0.9]^d$ inside of it
- smaller cube seems relatively large: each edge has length 0.8 versus 1 in the larger cube
- volume of $[0,1]^d$ is one: the volume of $[0.1,0.9]^d$ is $(0.9-0.1)^d = (0.8)^d$
- for $d = 20$, this volume is approximately 0.01
- a set can look large in all coordinates separately, but be small in the whole data space

Points are far away from each other

- for n points uniformly distributed in $[0,1]^d$, the expected L_∞ distance of point number 1 to its nearest neighbour is

$$D(d,n) \approx \left(\frac{1}{n}\right)^{1/d}$$

- when d grows but n is fixed, this distance converges to 1 (exact result in exercises)
- *almost all points are near the side, and far from each other*

Points tend to be mutually orthogonal

- generate n points from $N(0, \sigma^2 I_d)$
- compute the inner product of $n - 1$ points with the first point x
- compare the inner products with $\|x\|^2$
- result: when d is high, the inner products are small compared to $\|x\|^2$
- demo: `prohist.R`

- CoD $\implies$ local learning is not very useful in high-d spaces
- learning from data requires more information than vague regularity
- note that depending on the problem, seemingly weak information may actually be a strong assumption

- compare regression with classification
- try to predict bits from observed inputs $x \in \mathbb{R}^d$
- this is two-class classification, or regression with binary outputs
- solving this as a regression problem seems impossible without lots of information about the problem

- but as a classification problem a somewhat regular discriminant function if often used
- such a discriminant function makes a strong but usually plausible assumption
- consider two clusters of training data, each containing data from one class only
- solving a classifier defines the classes of points arbitrarily far from training data
- an exercise problem considers $d = 1$ and a "linear" classifier

- so far:
  - NFL: no information $\Longrightarrow$ no learning
  - weak assumptions $\Longrightarrow$ local learning, not useful in high-d problems
  - strong assumptions $\Longrightarrow$ learning is possible, if assumptions are correct

Overfitting

- lets examine what happens if a learning method works by selecting a solution out of a set $\{f(x|\theta)|\theta \in \Theta\}$
- assume the model $f(x|\theta)$ is correct, meaning that data is generated as $y = f(x|\theta_0) + n$ where n is an error term
- many learning methods use training data to compute an estimate $\hat{\theta}$
- the solution is then defined as $f(x|\hat{\theta})$

- estimation requires a *loss function* $L(y, f(x|\theta))$ to measure how good each θ is
- *expected loss (risk)* $R(\theta) = \mathrm{E}(L(y, f(x|\theta)))$
- *observed loss (empirical risk)* $R_{emp}(\theta) = \frac{1}{n} \sum_i L(y_i, f(x_i|\theta))$
- often $\hat{\theta}$ is found by minimizing $R_{emp}(\theta)$

- this loss-framework includes regression, classification, density estimation, ICA etc...
- if $\{f(x|\theta)|\theta \in \Theta\}$ is "small", then minimizing R_{emp} may work
- but many problems require complex models, where the set of solutions is not small
- large set leads to many values of θ minimizing R_{emp}
- called *overfitting*: can't tell solutions apart using only training data

- overfitting can happen even if $f(x|\theta_0)$ is the *correct solution* as defined above
- example: linear model

$$y = \mu x + \beta + n, \;\; n \sim N(0,1)$$

- this is a standard linear regression model

- observe outputs y_0, y_1 at inputs $x = 0, x = 1$
- fit the model (find $\hat{\mu}$, $\hat{\beta}$) by minimizing the squared error

$$R_{emp}(\hat{\beta}, \hat{\mu}) = (y_0 - \hat{\beta})^2 + (y_1 - \hat{\mu} - \hat{\beta})^2$$

- the error is simply the sum of squared errors at $x = 0$ and $x = 1$

- predict output at $x = 2$ as $2\hat{\mu} + \hat{\beta}$
- for comparison, fit a *constant model* $y = \beta + n$
- correct prediction is $2\mu + \beta$: we can compare the two models against this
- depending on μ, sometimes the constant model gets a *smaller mean-square error!*
- demo: `overfitting.R`, and an exercise problem

- lets illustrate overfitting geometrically
- assume $\theta \in \mathbb{R}^d$, a d-dimensional space
- consider a training set of m points
- then

$$\underline{y} = (y_1, \ldots, y_m) \in \mathbb{R}^m$$

- each θ is mapped to

$$\underline{y}(\theta) = (f(x_1|\theta), f(x_2|\theta), \ldots, f(x_m|\theta))$$

- this defines a function $f' : \mathbb{R}^d \to \mathbb{R}^m$
- for simplicity, assume that this function is somewhat regular

- suppose $d \gg m$: there are lots of functions, but little data
- f' is a mapping from a higher-dimensional space to a lower-dimensional space
- example: $d = 3, m = 2$ ("flatten" a cube)
- many θ's map into a single $\underline{y}(\theta)$
- since good solution $= f'(\theta)$ close to $\underline{y}$, we find too many solutions

- if $d \ll m$, then the image of f' is a subspace of $\mathbb{R}^m$
- if $\underline{y}$ is not near this subspace, no solution is very good
- this is called *underfitting*
- if the simple model is correct, then underfitting is unlikely

- if $d \sim m$ (saying $d = m$ is too simple), the dimensions "match" and one can generally find a unique solution
- using the correct model is not enough if the learning method estimates a single value θ from training data
- having too few training points can prevent finding it

- lots of heuristic methods have been proposed to select the right size for $\{f(x|\theta)|\theta \in \Theta\}$
- e.g. Minimum Description Length, Structural Risk Minimization
- in most of these, a penalty proportional to model complexity is added to the loss

- other heuristics for avoiding overfitting are *validation* methods
- part of the data is lost, since it is used only in the validation step
- note that validation does not help if you strictly have no information about the learning problem (NFL again)

- the above discussion was about solving the learning problem by selecting a single value for θ (point estimation) by minimizing R_{emp}
- we will not solve the learning problem this way: intention was to demonstrate how it leads to overfitting
- however, many learning methods solve learning problems in the way described above

- point estimation, and the heuristics for avoiding overfitting, have some additional problems
  1. usually no uncertainty in θ: the exact value cannot generally be found using a finite training set
  2. overfitting avoidance leads to using the wrong model
  3. which heuristic to use? different solutions to exactly the same problem
- next week, Bayes approach is develop which will address these problems

Lecture 3: 31.1.2007

Probabilities in learning

Bayesian Inference

- we have seen that data cannot be generalized without any information (NFL thms)
- vague assumptions (e.g. weak regularity) do not help much
- even knowing the correct model may lead to problems such as overfitting
- solution: represent all information about the learning problem correctly

- we will use *probabilistic modeling* to represent the information
- can be justified, though not in a universally acceptable way
- also suggested by situations, where probabilistic models are obviously correct
- seems that no practical alternative exists (learning methods with demonstrable shortcomings do not count)

Learning Problem Again

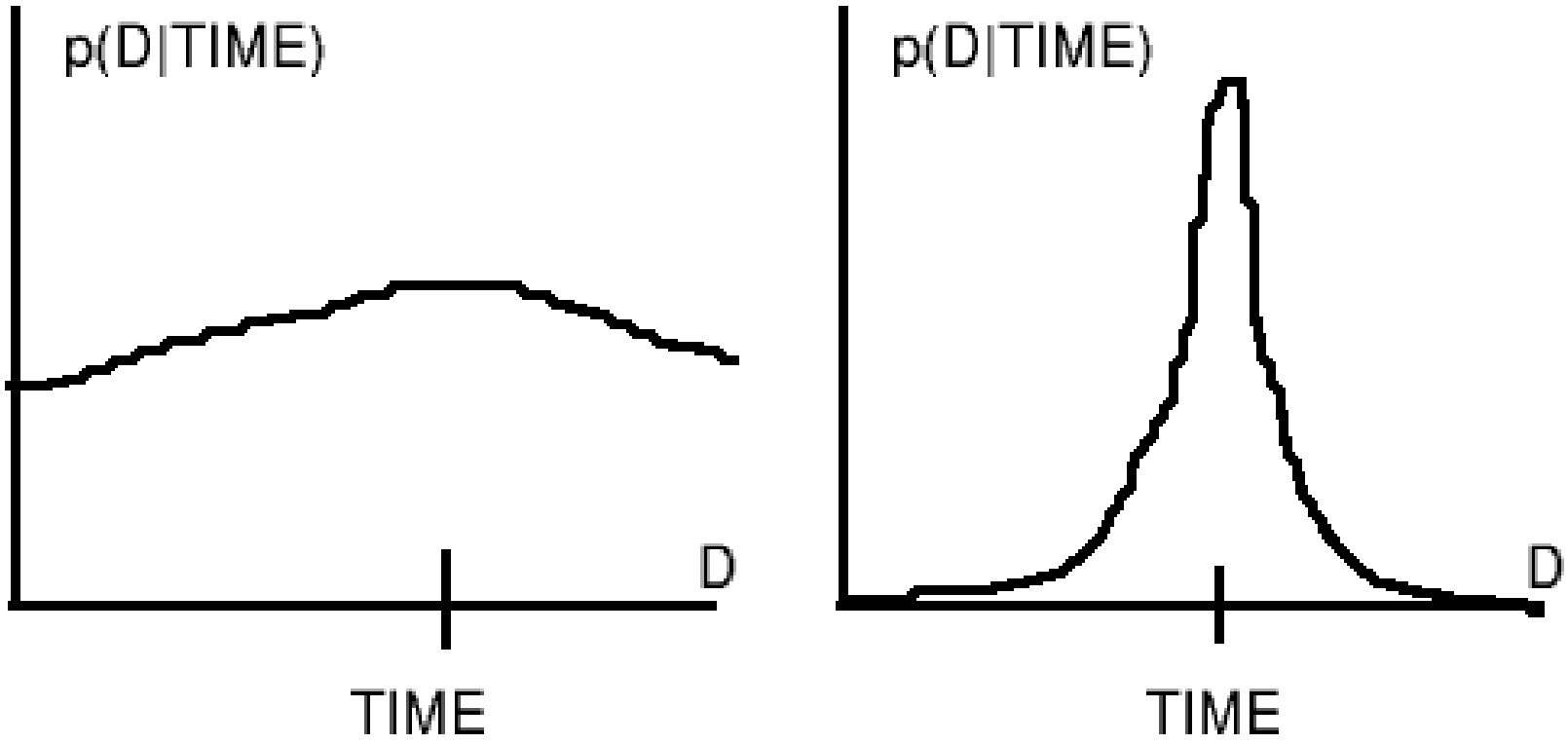
- assume there is a variable whose value is unknown
- denote this by θ, and the set of all possible θ's by Θ
- goal of learning is to get information about θ
- what kind of information we need in addition to data?

- lets go through an example that suggests certain important concepts
- example: what time is it?
- correct but unknown time is θ
- data D is what you read from your watch

- after looking at your watch, do you know what time it is?
- not exactly, because no watch can keep infinitely accurate time
- most of us would expect that the time shown by the watch is on average θ, but can deviate a little from it
- note that different people probably have a different description for the possible deviation

- we must have some information on how D depends on θ
- this *does not come from data* and *is not objective*
- lets pick just one watch: now there is exactly one learning problem
- the owner of the watch should have the best information about the accuracy of the watch

- for example, owner may know that the watch is very accurate
- then others would underestimate the accuracy
- distribution of D is sharp for an accurate watch and wide for a less accurate watch


p(D|TIME)
D
TIME
p(D|TIME)
D
TIME

- what if we just estimate $\hat{\theta} = D$?
- this may be reasonable if the watch is assumed to be accurate enough
- but what if the watch says "3 am"?
- we could all go home and get some sleep if we accept the above solution

- no one really thinks its 3 am, but why?
- because we have information about θ *before looking at the watch*
- this information is also *different for each person*
- to solve the problem properly, one must make a compromise between information about the watch and information about θ

- information about θ is called *prior information*
- it should be obvious that such information exists
- past experience about similar problems, perhaps based on data which is not available anymore
- in the example, several sources of information (having looked at the watch a while ago, knowing the time the lecture is given etc..)

- another problem with the solution $\hat{\theta} = D$
- if the watch is assumed to be inaccurate, then θ may be far from D
- as a single number, $\hat{\theta} = D$ carries no information about this inaccuracy
- in general, result of learning is *not a single value for the unknown* θ

- e.g. you want to catch a bus which departs every hour
- if D is "14:57" you may decide to try to catch the bus
- but if the watch is very likely to be five minutes late, it has a strong influence on the decision
- result of learning should describe the remaining uncertainty in θ

- many learning methods have some of these elements
- for example, a set of solutions $\{f(x|\theta)\}$ contains some information about how D depends on θ
- often no "3 am" safeguards: best-fitting θ is usually selected
- no proper description of remaining uncertainty, at most confidence intervals

- in the watch example, we had to come up with two descriptions of uncertainty
- *prior uncertainty*: what we know about θ
- *how data depends on* θ: what we know about D if θ is given
- in what way should we quantify the uncertainties?

- probability turns out to be a good measure for uncertainty
- probabilities will be used (also) for “one-time events”
- example: what is the probability that it rains tomorrow?
- it is possible to consider such probabilities since the source of uncertainty is *subjective lack of information*
- i.e. use of probabilities is not restricted to events repeatable infinitely many times

Probability as a Measure of Uncertainty

- probability is the *unique* measure of uncertainty, given certain axioms or assumptions
- this can be understood or demonstrated in several ways:
  - by analogy: if something is random when repeated (e.g. coin toss), its uncertainty is represented by a probability distribution
  - Dutch Book Theorem (Ramsey, de Finetti)
  - Cox's Axioms

Betting and Subjective Probability

- what happens if one insists using a non-probability measure for uncertainty?
- non-probability means that your measure does not follow the rules of computing with probabilities
- a betting argument shows that a non-probability measure leads to trouble

- consider a bet T_A which pays the owner 1 EUR if A happens, otherwise it pays nothing
- you would certainly take the bet for free
- also, you would not pay more than 1 EUR for it in any situation
- assume that there is a unique limit price (buy below it and sell above it)

- denote your limit price for T_A as p
- p depends on the actual event A
- lets choose p as an arbitrary non-probability measure $q(A)$
- Ramsey and de Finetti showed that using q leads to irrational behaviour

- exercise problem shows that a non-probability measure $q(A)$ leads to accepting a set of bets *guaranteed to make you lose money*
- this is called a Dutch Book due to bookmakers (persons taking bets e.g. in horseracing) who attempt to set their odds so that whatever happens, they will win money
- Dutch Book Theorem: the possibility of a set of bets guaranteed to make you lose money is equivalent to a non-probability uncertainty system

Cox's Axioms

- Cox derived probability as a measure of uncertainty from a short list of axioms
- further work has reduced and/or changed some of the axioms and as a result, there are a number of different derivations
- some key axioms (but not all):
  1. uncertainties are real numbers $p(A) \in \mathbb{R}$
  2. if A and B are equivalent, then $p(A) = p(B)$
  3. for a certain event A, $p(A) = 1$

- Dutch Book and Cox's Axioms support the use of probability as a measure of uncertainty
- other measures must disagree with at least one of the axioms
- the main axiom one might not accept is the "single real number" axiom
- is there uncertainty about uncertainty?

- there are certain approaches to uncertainty allowing for "sets of probabilities"
- from a practical point of view, one can add new unknowns which are used to give information about θ
- then one can use a probability distribution over all unknowns
- results in a probability over θ anyway

- example: coin toss
- unbiased coin: $p(\text{heads}) = 0.5$ and $p(\text{tails}) = 0.5$
- biased coin: "heads" or "tails" has probability 0.75
- suppose your probabilities are $1/3$ for "balanced coin", $1/3$ for "heads more probable" and $1/3$ for "tails more probable"

- lets compute the total distribution:

$$\begin{aligned} p(\text{tails}) &= \frac{1}{3} p(\text{tails} \mid \text{balance}) \\ &\quad + \frac{1}{3} p(\text{tails} \mid \text{tails probable}) \\ &\quad + \frac{1}{3} p(\text{tails} \mid \text{heads probable}) \\ &= \frac{1}{3} * \frac{1}{2} + \frac{1}{3} * \frac{3}{4} + \frac{1}{3} * \frac{1}{4} = \frac{1}{2} \end{aligned}$$

- similarly we obtain $p(\text{heads}) = \frac{1}{2}$
- averaging over the "extra" uncertainty effecticely removes it

- same probabilities for unbiased and possibly biased coin
- do we lose some information here?
- no, if we are to make a decision based on the probabilities
- any decision, which depends on the outcome of the toss must be the same in both cases

- Dutch Book and Cox's Axioms tell us to use only probabilities for quantifying uncertainty
- the learning problem is to find out what is known about θ after data D is observed
- denote this distribution as

$$p(\theta|D, I)$$

- the "extra" variable I means the information that was available before looking at the data

- we omit the variable I from now on, it has no relevance to any calculations
- it simply denotes the source of subjective information that will not be used explicitly
- since $p(\theta|D)$ is the proper description of uncertainty about θ, nothing more can be done using only data and information I
- Bayesian Inference is mainly about computing this distribution

- the notation $p(A|B)$ means a *conditional probability*
- it should be understood as "probability of A, given that B is known"
- computing with conditional probabilities is technically easy
- but thinking in terms of them is important but not always easy (see exercise problems this week)

- why not use traditional statistics?
- there the emphasis is on the distribution $p(D|\theta)$ (likelihood as a function of θ)
- this is the "uncertainty" of data, given θ
- but data is observed and θ is unknown: $p(D|\theta)$ cannot be used on its own

- ad-hoc methods are used which are based on taking averages over all possible data
- e.g. confidence intervals: some interval $[f(D), g(D)]$ has 95 percent confidence if it contains θ 95 percent of the time, averaged over all possible D
- this does *not* mean that θ is in $[f(D), g(D)]$ with 95 percent probability for a given D!
- also, $p(D|\theta)$ *as a function of* θ is not a proper description of uncertainty in θ

- notice that $p(\theta|D)$ is a probability distribution as a function of θ
- an interval can easily be defined using $p(\theta|D)$ so that it will contain θ with a given probability
- the probability $p(D|\theta)$ is very important part in computing $p(\theta|D)$, but it is not enough on its own
- the remaining part is $p(\theta)$ which describes the uncertainty in θ *before* seeing data

- since $p(\theta)$ can and should be chosen based on subjective information, its use has caused some critisism
- but choosing $p(D|\theta)$ is also subjective
- avoiding subjective choices is impossible, since the model comes from information available before seeing data
- Bayesian inference has some ways of attempting to obtain objective results (more on this later)

- example: line fitting
- model: $y = \mu x + \beta + n$ where $n \sim N(0, \sigma^2)$
- in words, fit a line to data assuming Normally distributed errors
- unknown $\theta = (\mu, \beta)$: it might include σ^2, but for now we assume that it is known

- what is $p(D|\theta)$?
- data is a set of pairs $(x_i, y_i), i = 1, \ldots, n$
- if they are independently generated, then

$$\begin{aligned} p(D|\theta) &= \prod_{i=1}^{n} p(x_i, y_i|\theta) \\ &= \prod_{i} C e^{-\frac{1}{2\sigma^2}(y_i - \mu x_i - \beta)^2} \end{aligned}$$

- the prior $p(\theta)$ depends on what else you assume, and could be any probability distribution
- note that standard regression methods don't use $p(\theta)$
- the posterior $p(\theta|D)$ is computed next week

- summary:
  - learning requires information about uncertain quantities
  - probability is the right way to quantify this (according to Dutch Book, Cox's Axioms)
  - both $p(\theta)$ and $p(D|\theta)$ are subjective
  - result of learning is $p(\theta|D)$, which is not an estimate of θ as a single number

LECTURE 4: 7.2.2007

BAYESIAN INFERENCE

- Dutch Book and Cox's Axioms support using probability as a measure of uncertainty
- Dutch Book Theorem implies that a measure of uncertainty must have certain properties, such as
  - product rule: $p(AB|C) = p(A|BC)p(B|C)$ (AB means both events A and B happen)
  - sum rule: $p(A|B) + p(\overline{A}|B) = 1$ ($\overline{A}$ means A does not happen)
  - equivalent events: $p(A) = p(B)$ if A and B are equivalent events
- Cox's axioms yield the same properties

Bayes' Theorem

- consider uncertain events A and B
- product rule gives

$$p(BA) = p(B|A)p(A)$$
$$p(AB) = p(A|B)p(B)$$

- AB and BA are *equivalent events*
- their probabilities must be the same

- equating $p(BA)$ and $p(AB)$ gives

$$p(B|A)p(A) = p(A|B)p(B)$$

- dividing by $p(B)$ gives the *Bayes' Theorem*

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

- BT allows us to *reverse* conditional probabilities
- in context of the learning problem, BT can be used when
  - prior uncertainty gives the prior $p(\theta)$
  - modeling assumptions give $p(D|\theta)$
- BT results in the *posterior distribution*

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$

- constant denominator $p(D)$ is often omitted:

$$p(\theta|D) \propto p(D|\theta)p(\theta)$$

- when necessary, it can be computed as

$$p(D) = \int p(D|\theta)p(\theta)d\theta$$

- unnormalized posterior can often be used directly

- how to get $p(D|\theta)$ from uncertainty?
- Dutch Book and Cox do not directly give conditional probabilities
- instead, one can start with the *full probability model*

$$p(\theta, D)$$

- obtained by using all relevant quantities in the distribution
- Dutch Book and Cox say that this distribution exists
- the full probability model defines all other probabilites involving θ and D

- integrate over data (sum rule):

$$p(\theta) = \int p(\theta, D) dD$$

- likelihood by the product rule:

$$p(D|\theta) = p(\theta, D) / p(\theta)$$

- both are obtained from the full probability model using rules of computing with probabilities
- in practice, $p(\theta)$ and $p(D|\theta)$ are usually specified directly

- $p(D|\theta)$ and $p(\theta)$ are specified using the information available about the problem
- using the Bayes' Theorem in unnormalized form gives the posterior as

$$p(\theta|D) \propto p(D|\theta)p(\theta)$$

- posterior quantifies the remaining uncertainty about θ taking the information in D into account
- this result is unique, if we believe that our probability model is correct

- note that $p(\theta, D)$ does not have to be "correct" in any general sense
- correctness here means "correctly quantifies our subjective information"
- then the posterior correctly quantifies our information after having seen data D
- practical reasons may justify other learning methods, but in principle learning proceeds as described above

- *point estimation*: many learning methods would give a single value for θ
- posterior does not define a unique point estimate
- a loss function is normally required (not part of the probability model)
- e.g. minimum MSE, maximum likelihood
- exception: the posterior is concentrated on a single value

- example: cross a bridge with a 10-ton truck
- somehow you believe that the bridge holds 11 tons with $p = 0.8$ and 9 tons with $p = 0.2$
- a reasonable point estimate might suggest 11 tons (would you use it and cross the bridge?)
- consequences imply here a nonsymmetric loss

- overfitting: for example, $\hat{\theta} = \text{argmax}\, p(\theta|D)$
- consider a real-valued θ
- then the density $p(\hat{\theta}|D)$ is not a probability
- but an integral over some neighbourhood of $\hat{\theta}$ is:

$$p(\hat{\theta} - \epsilon \leq \theta \leq \hat{\theta} + \epsilon|D) = \int_{\hat{\theta}-\epsilon}^{\hat{\theta}+\epsilon} p(\theta|D)d\theta$$

- it is possible that $p(\hat{\theta}|D)$ is large, but the integral above is small (even for somewhat large ϵ)
- visually, this happens when the posterior has a "narrow peak" around $\hat{\theta}$
- this means that it is not very probable that θ is close to $\hat{\theta}$ (exercise problem)

- recall the linefitting problem:

$$\begin{aligned} y &= \mu x + \beta + n, \;\; n \sim N(0, \sigma^2) \\ &= \begin{bmatrix} x & 1 \end{bmatrix} \theta + n, \;\; \theta = \begin{bmatrix} \mu & \beta \end{bmatrix}' \end{aligned}$$

- $N(y|a,b)$ means a Normal density for y with mean a and variance b

- denote

$$X = \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ 1 & 1 & \dots & 1 \end{bmatrix}'$$

$$Y = [y_1, \dots, y_n]'$$

- likelihood is $p(Y|\theta, X) = N(Y|X\theta, \sigma^2 I)$
- choose first a prior $p(\theta|X) \propto c$ (constant prior)

- use "completing the square"
- consider any expression

$$C \exp(-\frac{1}{2}(\theta' A \theta + \theta' b + b' \theta + c))$$

- this is an unnormalized Normal distribution $N(\theta|m, R)$: write the exponent as

$$-\frac{1}{2}(\theta - m)' R^{-1} (\theta - m)$$

- we see that $R = A^{-1}$ and $m = -Rb$

- compute the posterior:

$$p(\theta|Y,X) \propto p(Y|\theta,X)p(\theta|X)$$
$$= N(Y|X\theta,\sigma^2 I)p(\theta|X)$$

- use "completing the square" on the exponent (ignore additive constants):

$$\log p(\theta|Y,X) \propto -\frac{1}{2}\sigma^{-2}(Y-X\theta)'(Y-X\theta)$$
$$= -\frac{1}{2}\sigma^{-2}(Y'Y+\theta'X'X\theta-\theta'X'Y-Y'X\theta)$$
$$= -\frac{1}{2}(\theta-m_\theta)'R_\theta^{-1}(\theta-m_\theta)$$
$$= -\frac{1}{2}\sigma^{-2}(\theta-(X'X)^{-1}X'Y)'X'X(\theta-(X'X)^{-1}X'Y)$$

- therefore the posterior is

$$p(\theta|Y,X) = N(\theta|m_\theta, R_\theta)$$
$$m_\theta = (X'X)^{-1}X'Y$$
$$R_\theta = \sigma^2(X'X)^{-1}$$

- the posterior mean is familiar from standard linear regression as it is obtained using the *pseudoinverse* $(X'X)^{-1}X'$
- what if the prior is not constant?
- lets choose a Normal prior $p(\theta|X) = N(\theta|a, B)$

- using the same technique as above (completing the square) we obtain

$$
\begin{aligned}
p(\theta|Y,X) &= N(\theta|m_\theta, R_\theta) \\
m_\theta &= (\sigma^{-2}X'X + B^{-1})^{-1}(\sigma^{-2}X'Y + B^{-1}a) \\
R_\theta &= (\sigma^{-2}(X'X) + B^{-1})^{-1}
\end{aligned}
$$

- this is easy to compute for any a, B
- check what happens when $B = \sigma_p^2 I$ and σ_p^2 gets very large

- why did we get the pseudoinverse result at first?
- when the prior is *constant*, the posterior is always proportional to the *likelihood*:

$$p(\theta|D) \propto p(D|\theta)p(\theta) \propto p(D|\theta)$$

- for a Normal posterior, $\max p(\theta|D) = \mathrm{E}(\theta|D) = m_\theta$
- *maximum likelihood* would then also give m_θ
- BI gives this result and the posterior variance directly

Marginalization

- Bayesian Inference seems almost too trivial to be useful and interesting
- but practical applications can be (and usually are) quite complicated
- some interesting properties can be demonstrated without going to problem-specific details
- *marginalization*: removing uninteresting but relevant quantities

- example: which taxi company $C \in \{1, 2\}$ to take to airport
- goal: minimize driving time t
- two routes $r \in \{1, 2\}$
- assume we know distribution of routes for each company $p(r|C)$ and distribution of driving times $p(t|r)$ on each route

- we need to find $p(t|C)$
- why bother with route r if we just care about the driving time?

$$\begin{aligned} p(t|C) &= p(t, r = 1|C) + p(t, r = 2|C) \\ &= p(t|r = 1)p(r = 1|C) + p(t|r = 2)p(r = 2|C) \end{aligned}$$

- route must be included, since it links C and t
- but we don't care about the route, so $p(t|C)$ is what we want to compute

- removing uninteresting but relevant quantities is called *marginalization*
- suppose $\theta = (x, y, z)$ and y is uninteresting
- then the goal is to compute $p(x, z|D)$
- if we get it as $p(x, z|D) \propto p(D|x, z)p(x, z)$, then y is irrelevant
- heuristic, wrong solution is to compute $p(x, y, z|D)$ and estimate y, then use $p(x, \hat{y}, z|D)$ as the distribution over x, z

- correct solution from rules of probability:

$$p(x, z|D) = \int p(x, y, z|D)dy$$

- called marginalization, because we are computing a marginal distribution of the full posterior
- intuition: all possible values of y are considered, and their effect is weighted by the full posterior

- example: noise variance in regression
- often regression noise is modeled as $N(0, \sigma^2)$
- variance σ^2 assumed to be an unknown constant
- leads to the familiar least-squares solution
- Bayes requires considering *all values of* σ^2
- we'll see later that this gives a different result

Model Averaging

- another way to think about marginalization, especially when predicting observable values
- consider regression, where output y is predicted as a function of an input x
- assume we are interested in predicting $\tilde{y}$ at an input $\tilde{x}$
- non-Bayesian solution: find an estimate $\hat{\theta}$ for the regression parameters, then predict $\tilde{y} = f(\tilde{x}|\hat{\theta})$
- this can lead to overfitting as seen before

- let's predict $\tilde{y}$ using probabilities
- $\tilde{y}$ is unknown so we must have a distribution over it
- we know $\tilde{x}$ and D, so the distribution must be conditional to these values
- *we don't know* θ, so the distribution is *not* conditional to θ

- we want to compute the *predictive distribution*

$$p(\tilde{y}|\tilde{x}, D)$$

- in general, a predictive distribution is any distribution over an observable quantity (e.g. $p(D)$)
- assume we obtained $p(\theta|D)$
- θ must be included in the analysis, because it is clearly relevant

- use marginalization backwards:

$$p(\tilde{y}|\tilde{x}, D) = \int p(\tilde{y}, \theta|\tilde{x}, D)d\theta$$

- the integrand is the *full posterior of* $\tilde{y}, \theta$
- use the product rule to split the integrand:

$$p(\tilde{y}|\tilde{x}, D) = \int p(\tilde{y}|\tilde{x}, \theta, D)p(\theta|\tilde{x}, D)d\theta$$

- remove unnecessary quantities (requires model assumptions):

$$p(\tilde{y}|\tilde{x}, D) = \int p(\tilde{y}|\tilde{x}, \theta)p(\theta|D)d\theta$$

- $p(\tilde{y}|\tilde{x}, \theta, D) = p(\tilde{y}|\tilde{x}, \theta)$ since θ determines predictions
- $p(\theta|\tilde{x}, D) = p(\theta|D)$ since pairs (x, y) determine θ
- $p(\tilde{y}|\tilde{x}, \theta)$ is also a predictive distribution
- it cannot be used directly since θ is unknown

- integral *averages the predictive distributions using the posterior over* θ
- predictions made by more probable θ carry more weight and vice versa
- follows from marginalization and the product rule (no heuristics used)
- result is the only possible distribution of $\tilde{y}$ quantifying the uncertainty in the prediction

- possible misunderstanding of model averaging
- *averaging is over probabilities, not predicted values!*
- assume you write "0" badly, so it looks a bit like "9"
- handwritten digit recognition model might predict "0" for θ_1 and "9" for θ_2 ($p(\theta_i|D) = 0.5,\ i = 1, 2$)
- model averaging does not yield 4.5
- instead, model averaging puts some probability on both predictions

Overfitting

- caused by too many values of θ fitting to data
- for any such θ, $p(\theta|D)$ may be high
- if the predictions for such θ's are different, this is overfitting
- this does not always happen: the predictions may also be similar

- model averaging automatically reveals overfitting
- assume $p(\theta_1|D) = p(\theta_2|D) = p(\theta_3|D) = 1/3$
- if θ_i predicts $\tilde{y} = i$ with probability one, then

$$p(\tilde{y} = i|\tilde{x}, D) = 1/3, \;\; i \in \{1, 2, 3\}$$

- but if each θ_i predicts $\tilde{y} = 1$, then the predictive distribution is $p(\tilde{y} = 1|\tilde{x}, D) = 1$

- different predictions by probable θ's mean a *high predictive variance* and thus overfitting
- similar predictions lead to *low predictive variance*
- thus, the predictive distribution reveals overfitting and model-selection heuristics are not needed
- predictive variance can be different at different $\tilde{x}$: the whole model does not necessarily overfit

- example: linear regression using fixed, nonlinear basis functions
- can be solved almost as easily as the linear example
- predictive distributions can also be computed easily
- demo: `demo_breg.R`

Lecture 5: 14.2.2007

Statistical Learning Theory
Support Vector Machine

Statistical Learning Theory

- motivation:
  - there are some "theories of learning" that seem to support learning from data without information
  - contradiction with NFL thms: important to examine what these theories actually claim
  - we concentrate on Statistical Learning Theory, perhaps the most well-known theory
  - Bayes-approach is used to explain the apparent conflict with No Free Lunch theorems

- Statistical Learning Theory contains results on generalizing a finite set of training data
- SLT $\stackrel{?}{\Longrightarrow}$ model-free learning is possible?
- not so: SLT does not make such a claim about learning from a *given set of data*
- SLT is not incorrect, but irrelevant to the learning problem defined earlier

- most details are skipped, as SLT turns out not to contradict NFL theorems
- the goal is to understand what the main result of SLT actually claims
- after this, you should be better equipped to decide whether studying such theories any further is worth the effort
- we can't skip SLT completely, since it is still being used to justify various model-free learning methods

- lets concentrate on binary outputs (binary regression or two-class classification)
- SLT requires that you predefine a set of solutions H which contains some (but not all) functions

$$h : X \rightarrow \{0, 1\}$$

- then SLT measures the "smallness" of the set H using Vapnik-Chervonenkis dimension

- SLT seems to say that if h fits training data well and is from a "small" set H, then h works well *in general*
- one wouldn't expect to find a good solution from a "small" H by chance
- note that this theory does not require you to know anything about the relationship between inputs $x \in X$ and the corresponding outputs in $\{0, 1\}$

Vapnik-Chervonenkis Dimension

- denote any training set of size n by

$$Z_n = \{(x_i, y_i) | x_i \in X,\ y_i \in \{0,1\},\ i = 1, \ldots, n\}$$

- for each classifier $h \in H$ we obtain a *dichotomy*

$$(|y_1 - h(x_1)|, \ldots, |y_n - h(x_n)|)$$

- $|y_k - h(x_k)| = 1$ if h predicts wrong, otherwise zero
- fix Z_n, go through all $h \in H$ and denote the number of *different* dichotomies by $N(Z_n)$

- then maximize $N(Z_n)$ over *all sets* Z_n with n fixed
- we have gotten rid of the specific training set Z_n and the specific classifier h
- the *growth function*

$$G(n) = \log \max N(Z_n)$$

depends only on H, the domain X and the number of points n

- for any $n \geq 1$, one of the following holds:

$$G(n) = n \log 2$$
$$G(n) \leq v(\log(n/v) + 1)$$

- the integer v is the *VC-dimension*
- defined as the integer v satisfying

$$G(v) = v \log 2$$
$$G(v+1) < (v+1) \log 2$$

- example: H contains all linear classifiers on a plane
- any three points x_1, x_2, x_3 not on the same line can be classfied in all possible ways by classifiers from H
- then $G(3) = \log 2^3$
- any four points cannot be classified arbitrarily by lines
- therefore $G(4) < \log 2^4$ and the VC-dimension is three

- assume the "true" solution is defined by the unknown distribution $p(x, y)$
- the data $d = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ has been generated from the distribution $p(x, y)$
- define

$$c = \mathrm{E}(|y - h(x)|) \text{ (average error)}$$

$$s = \frac{1}{m} \sum_{i=1}^{m} |y_i - h(x_i)| \text{ (training error)}$$

- we want a small c, but can only compute s

- the quantity c is important, since it measures how well h performs *in general*, not just on the training set
- in our case of binary outputs, we have

$$\begin{aligned} c &= \mathrm{E}(|y - h(x)|) \\ &= \int\int |y - h(x)| p(x, y) dx dy \\ &= \mathbb{P}(|y - h(x)| = 1) \end{aligned}$$

- i.e. c is the *error rate* of the classifier h

- we oviously cannot compute c without knowing $p(x, y)$
- but we can compute s, the training error
- can we get information about c, given only training data d, set H, and training error s?
- NFL theorems say this is impossible

- main SLT result gives an upper bound for c as a function of s, m (number of training points), and the VC-dimension
- following inequality holds simultaneously for all $h \in H$ *with probability* $1 - \nu$:

$$c \leq s + \frac{\varepsilon}{2}\left[1 + \sqrt{1 + \frac{4s}{\varepsilon}}\right]$$

$$\varepsilon = 4\frac{G(2m) - \log(\nu/4)}{m}$$

- ε is a function of m, ν, and the VC-dimension

- example:

$$
\begin{aligned}
s &= 0, \text{ (no training error)} \\
v &= 1, \text{ (small VC-dimension)} \\
m &= 0.5 * \exp(10) \approx 11000 \text{ training points} \\
\nu/4 &= \exp(-4) \implies \nu \approx 0.07
\end{aligned}
$$

- then the bound is

$$c \leq \varepsilon \leq 8e^{-10}(10 + 1 + 4) = 15 * 8 * e^{-10} \approx 0.005$$

- so perfect training performance, small VC-dimension and about eleven thousand samples seems to guarantee small c with probability $1 - \nu = 0.93$?

- is there a free lunch?
- it seems so: just select a set H with a small VC-dimension, then choose h which minimizes s
- the bound is often small, so we have found a good solution with no information?
- the set H can be freely chosen so this can't be explained by implicit information in selecting H

- unfortunately, there is no free lunch
- Bayes-approach: check what is known and what is uncertain
  - *known*: data d, set H, training error s, number of points m
  - *unknown*: average error c
- we should compute $p(c|d, H, s, m)$

- SLT uses only the *likelihood* $p(s|c, m, \dots)$
- i.e. distribution of s as a function of c
- not surprising that $p(s|c, \dots)$ is concentrated around c
- for example, $c = 0.1$ means the classifier makes an error once out of ten classifications on average
- not rocket science to expect that approximately one out of ten *training points* are misclassified
- so $p(s|c, \dots)$ has a peak around c

- add the extra term to obtain $s + \frac{\epsilon}{2}[\dots] > s$
- now $p(s + \frac{\epsilon}{2}[\dots]|c, \dots)$ is simply $p(s|c, \dots)$ shifted to the right
- most of the probability mass is now on values larger than c
- so we can say $c \leq s + \frac{\epsilon}{2}[\dots]$ with high probability

- *this is only true when s is random and depends on a constant c*!
- so we can say "if c is small, then s is probably small too"
- we *cannot* say "if s is small, then c is probably small"
- why? Conditional probabilities have to be reversed using Bayes' Theorem
- so we need to compute $p(c|s, \dots)$ which requires $p(c)$

- example: four points, one classifier $h \in H = \{h\}$
- c is the number of errors divided by 4
- compute $p(s|c)$ on two training points:

c	0	0.25	0.5	0.75	1
$p(s = 0\|c)$	1	0.5	1/6	0	0
$p(s = 0.5\|c)$	0	0.5	2/3	0.5	0
$p(s = 1\|c)$	0	0	1/6	0.5	1

- then $c \leq s + 1/4$ with probability at least $5/6$

- suppose $s = 0$ so the bound is $c \leq 1/4$
- do we get $c \leq 1/4$ with probability at least $5/6$?

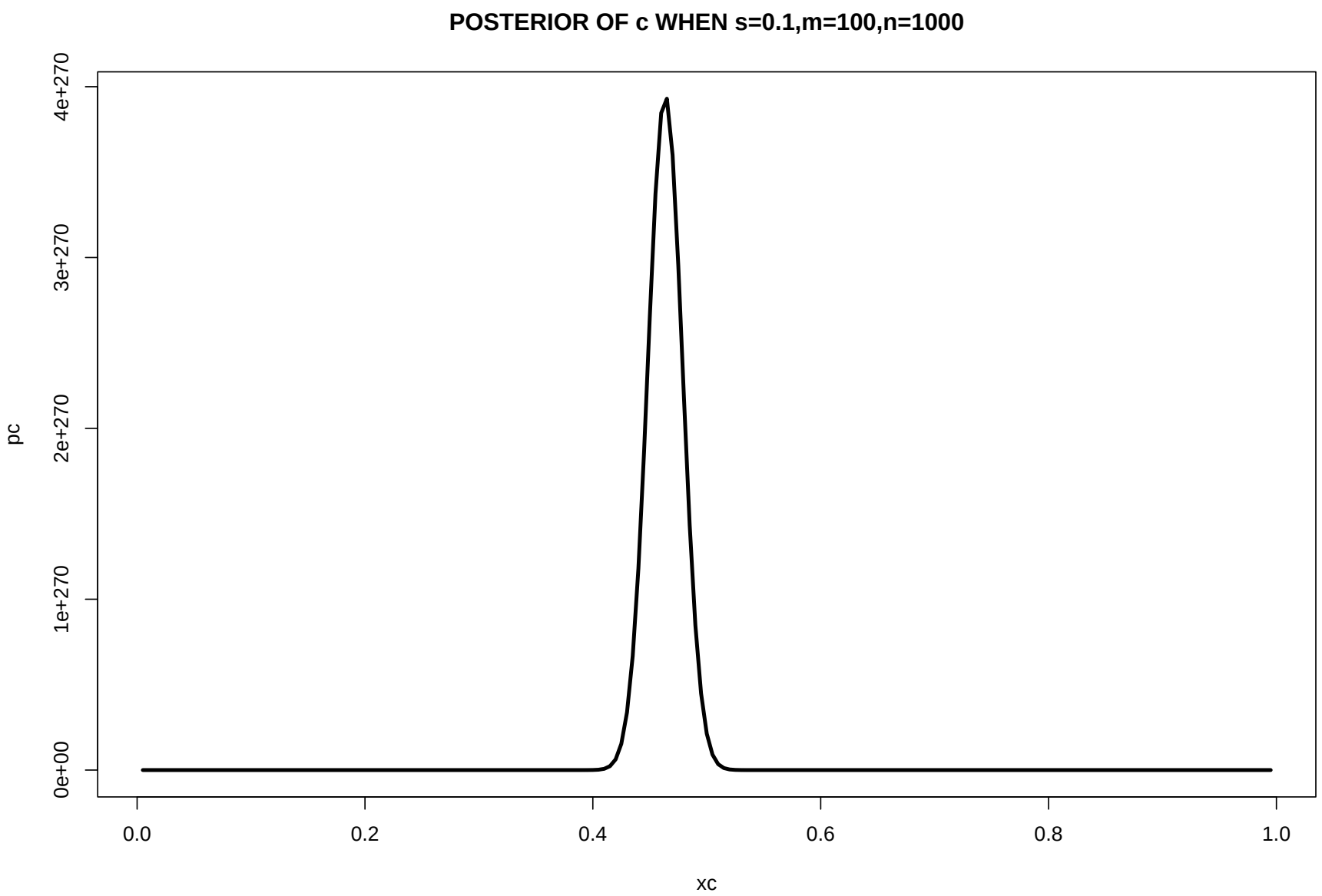
$$p(c = 0|s = 0) = 1/4$$
$$p(c = 1/4|s = 0) = 1/2$$

- since $3/4 < 5/6$, this demonstrates that we can't reverse probabilities without consequences

- the problem with SLT is similar to the fingerprint exercise problem
- $p(match|innocent)$ was small, but $p(innocent|match)$ was large
- reason was a very nonuniform prior of innocence
- in SLT, $p(c)$ is implicitly uniform, but it should not be

- uniform $p(c)$ implies that you have useful information about the error rate of h (exercise)
- if you don't, then a reasonable prior is a Binomial distribution for nc (exercise)
- this distribution is very nonuniform when n is even moderately large

- the flaw in SLT is demonstrated in detail in exercise problems
- example: binary classifier $H = \{h\}$
- $s = 0.1$, $m = 100$, and there are 1000 points to be classified
- small s and small VC-dimension v should imply that c is small?
- the posterior $p(c|d, s, m, v, h)$ looks like this (demo `fixed.R`):


POSTERIOR OF c WHEN s=0.1,m=100,n=1000
pc
0e+00
1e+270
2e+270
3e+270
4e+270
0.0
0.2
0.4
0.6
0.8
1.0
xc

- SLT bound gives $c \leq 0.33$ with probability $1 - \nu \approx 0.93$
- this clearly disagrees with the posterior
- the posterior would be concentrated *exactly* around $c = 0.5$, but c is measured *partly on the training set*
- *we already know* that for 100 training points the error is small ($s = 0.1$)

Support Vector Machine

- a two-class classifier often justified by SLT
- but we saw SLT does not show that learning without information is possible
- SVM has no properties that would give it a good off-training set error rate in general
- however, it has a number of practical advantages

- input vectors are mapped into a high-dimensional feature space using a *fixed nonlinear mapping*
- the fixed mapping is selected so that inner products in the feature space are easy to compute
- data is classified linearly in the feature space
- this corresponds to a nonlinear classifier in the data space

- SVM has some practical advantages:
  - flexible model: easy to implement complicated nonlinear classifiers
  - the classifier is typically determined by a small part of training data
  - computational benefits: quadratic programming solves the classifier, inner products are computed easily

Optimal Linear Classifier

- data: vectors $x_i \in \mathbb{R}^d$, class labels $y_i \in \{-1, 1\}$
- linearly separable training data $(x_1, y_1), \ldots, (x_m, y_m)$
- first component of x is always one: then a linear classifier is a hyperplane defined by $w'x = 0$
- the *normal vector* w defines the classifier

- x is classified by the sign of the inner product $w'x$
- OLC is defined as the linear classifier that
  1. makes no errors on the training set:

$$y_i w' x_i \geq 1, \;\; i = 1, \ldots, m$$

  2. is as far as possible from the closest training point:

$$w^* = \text{argmax}_w \min_{i=1,\ldots,m} \frac{y_i w' x_i}{\|w\|}$$

- the expression $\frac{y_i w' x_i}{\|w\|}$ is the projection of x_i onto w, scaled by $\|w\|$
- this gives the distance of x_i to the hyperplane
- the above conditions result in the optimization problem

$$\frac{1}{2}\|w\|^2$$
$$y_i w' x_i \geq 1,\ i = 1, \ldots, m$$

- the solution is defined by the closest training point to the hyperplane
- there can be more than one: these points are called *support vectors*
- later we will see that the solution is in a sense defined by the support vectors

- OLC's "optimality": start with
  $S = \{w \mid y_i(w'x_i) \geq 1, i = 1, \ldots, m\}$
- VC-dimension of a subset
  $S(A) = \{w \in S \mid \|w\| \leq A\}$ gets smaller when A gets smaller
- SLT bound $c \leq s + \ldots$ is minimized when VC-dimension is minimized (due to $s = 0$)
- smallest $S(A) \neq \emptyset$ is $S(A = \min_{w \in S} \|w\|)$, so SLT suggests to minimize $\|w\|$ in S

- this derivation is useless: pick any $w_1 \in S$
- $S(A = \min_{w \in S} \|w\|)$ has at least the same VC-dimension as $\{w_1\}$
- so w_1 minimizes the bound as well as OLC
- all classifiers in S can't be "optimal" at the same time

- optimality of OLC fails in two ways:
  - having a small bound for c says nothing about the performance of the classifier (SLT's flaw)
  - even if it does, any classifier with $s = 0$ achieves the same bound
- OLC is optimal only in the sense it was defined before: it maximizes the distance to the closest training point

- regardless of optimality issues, we minimize

$$J = \frac{1}{2}\|w\|^2$$

  with constraints

$$y_i(w'x_i) \geq 1,\ i = 1, \ldots, n$$

- constraints ensure $s = 0$ and minimizing J maximizes the distance to support vectors

- can be solved by Quadratic Programming, giving the solution in terms of Lagrange coefficients u_i^*:

$$w = \sum_i u_i^* y_i x_i$$

- most $u_i^* = 0$: the nonzero ones correspond to *support vectors* x_i
- new vectors can be classified by computing the sign of

$$w'x = \sum_i u_i^* y_i x_i' x$$

- to obtain SVM from this linear classifier, the vectors x_i are mapped nonlinearly to $z_i = g(x_i)$
- demo: `Kernels/polykern.R`
- inner products of z_i and z_j need to be computed:

$$H(x_i, x_j) = z_i' z_j = \sum_{k=1}^{D} g_k(x_i) g_k(x_j)$$

- using this directly is difficult if D is large (it often is)

- for some mappings $z = g(x)$, the *kernel* $H(x_i, x_j)$ is a simple function of x_i, x_j:
  - polynomials of degree q: $H = (1 + x_i' x_j)^q$
  - localized basis functions: $H(x, x_i) = \exp(-\|x - x_i\|^2 \sigma^{-2})$
  - Fourier-series: $H = \frac{\sin(q+0.5)(x_i - x_j)}{\sin(x_i - x_j)/2}$
- recall that we assumed that one component is always constant: this holds e.g. for the polynomial kernel since one basis function of a polynomial is a constant

- now SVM can be defined as follows:

1 choose a feature space for which there is an inner product kernel $H(\cdot, \cdot)$

2 compute an inner product matrix with components $H_{ij} = H(x_i, x_j)$

3 solve the quadratic optimization problem and obtain u_i^*'s

4 classify new vectors x by taking the sign of $\sum_i u_i^* y_i H(x_i, x)$

- benefits of SVM:
  - the dimension of z is irrelevant (computationally) when solving the optimization problem
  - the resulting classifier is nonlinear in x-space
  - new samples are easy to classify linearly using the kernel

- disadvantages:
  - nonlinear mapping g is chosen arbitrarily, but the solution depends on it
  - OLC is not optimal in a general sense: it is simply a linear classifier that maximizes the distance to closest training point
  - the maximum distance holds in the z-space, but in the x-space distances are different

- SVM demos: `Kernels/linear.R` (demos by Ralf Herbrich, somewhat modified)
- `exmp.svm1`: linear kernel
- `exmp.svm2`: polynomial kernel, $q = 7$
- `exmp.svm3`: RBF kernel

Lecture 6: 21.2.2007

Bayesian Modeling: One-Variable Models, Priors

One-Variable Normal Models

- we will go through a few examples using Normal distribution illustrating certain aspects of Bayesian Inference
- benefits: closed-form solutions, easy to interpret what happens, can be used as building blocks in more complicated models
- drawbacks: Bayesian inference in practice almost always requires more complicated techniques

- example: Normally distributed data with known variance
- a Normal distribution $p(y|\theta, \sigma^2) = N(y|\theta, \sigma^2)$ is defined by its *mean* θ and *variance* σ^2
- we could interpret θ as the most probable value and variance as a measure of how certain we are about θ

- for convenience, we will use *precision* $\lambda = \sigma^{-2}$ instead of variance
- large precision $\implies$ small variance, θ known quite precisely
- small precision $\implies$ large variance, θ not known precisely
- lets examine what happens when we observe Normally distributed data

- likelihood is $p(y|\theta, \lambda) = N(y|\theta, \lambda^{-1})$ where

$$N(y|\theta, \lambda^{-1}) = \lambda^{1/2}(2\pi)^{-1/2} \exp(-\frac{\lambda}{2}(y-\theta)^2)$$

- assume that λ is known but θ is unknown and we observe one value y

- lets choose the prior as $p(\theta) = N(\theta|\theta_0, \lambda_0^{-1})$
- Bayes' Theorem gives the posterior as (exercise):

$$p(\theta|y) = N(\theta \mid \frac{\lambda_0\theta_0 + \lambda y}{\lambda_0 + \lambda}, (\lambda_0 + \lambda)^{-1})$$

- note that both the prior and the posterior are Normal

- lets examine how the mean and precision change as we go from Normal prior to Normal posterior
- precision changes as

$$\lambda_0 \rightarrow \lambda_0 + \lambda$$

- λ is the "data precision": observing one y *increases* the precision by λ
- λ_0, the prior precision, defines how accurately we knew θ before seeing any data

- mean is changed towards y as

$$\theta_0 \to \theta_0 + (y - \theta_0)\frac{\lambda}{\lambda + \lambda_0}$$

- the posterior mean is a weighted average of θ_0 and y
- the weighting depends on the precisions

- the "step size"

$$\frac{\lambda}{\lambda + \lambda_0}$$

  defines the weights

- if prior is more precise ($\lambda \ll \lambda_0$), the step size is close to zero and the posterior mean is close to prior mean

- if prior is not precise ($\lambda \gg \lambda_0$), the step size is close to one and the posterior mean is closer to y

- above discussion illustrates what happens in general
- compromise between data and the prior, automatically weighted by prior and likelihood precisions
- very precise prior makes data useless: we already know θ precisely
- very precise likelihood makes prior useless: data tells the value of θ very precisely

- what is the predictive distribution of a *new* value $\tilde{y}$?
- use "reverse marginalization" to compute

$$\begin{aligned} p(\tilde{y}|y) &= \int p(\tilde{y}, \theta|y) d\theta \\ &= \int p(\tilde{y}|y, \theta) p(\theta|y) d\theta \end{aligned}$$

- integrand contains $p(\tilde{y}|y, \theta) = N(\tilde{y}|\theta, \lambda^{-1})$

- the second term is the Normal posterior $p(\theta|y)$ which was just solved
- integrand is a Normal distribution of θ and $\tilde{y}$
- this can be seen by multiplying the two Normal distributions and writing the product in the form of $N((\theta, \tilde{y})'|A, B)$

- integration over θ gives a marginal distribution of the joint Normal distribution
- therefore $p(\tilde{y}|y)$ is also a Normal distribution
- its mean and variance are (see model averaging exercise problem)

$$\mathrm{E}(\tilde{y}|y) = \mathrm{E}(\theta|y)$$
$$\mathrm{var}(\tilde{y}|y) = \sigma^2 + \mathrm{var}(\theta|y)$$

- the predictive mean is identical to the posterior mean
- this corresponds to intuition: posterior mean is the most probable θ, and most probable $\tilde{y}$ generated is the mean of the Normal distribution
- but predictive variance is *not the likelihood variance*
- $\text{var}(\tilde{y}|y)$ is larger than σ^2, since there is also uncertainty about θ

- the same calculations may be repeated for n independent observations $y_1, \ldots, y_n$ which can also be vectors (exercise)
- posterior precision will be a sum of prior precision and one data precision for each observation, i.e. $\lambda_0 + n\lambda$
- the more data, the higher the precision of the posterior
- demo `bgauss.R` illustrates the above inference

- lets use previous results to construct a simple Bayesian classifier
- warning: this example is simplified too much to be realistic
- the purpose is to build the classifier using the Normal model as a building block

- data x_i belongs to one of two classes (0 and 1)
- data in each class is Normally distributed:

$$x_i \sim N(\mu_j, \lambda_j^{-1}) \text{ if } x_i \text{ is in class } j = 0, 1$$

- knowns: precisions λ_0, λ_1, training data $(x_i, y_i),\ i = 1, \ldots, n$
- unknowns: μ_0, μ_1 and class label $\tilde{y} \in \{0, 1\}$ corresponding to a new $\tilde{x}$
- prior: $p(\mu_0, \mu_1) = N(\mu_0|0, 1)N(\mu_1|1, 1)$

- unrealistic assumption: assume that each class generates equal amount of data
- this means that any $p(y = 0| \dots)$ *not conditional to* x is 0.5
- in a realistic classifier, this probability would be an unknown parameter

- the likelihood is

$$p(D|\theta) = \prod_i p((x_i, y_i)|\mu_0, \mu_1) =$$
$$= \prod_i p(x_i|y_i, \mu_0, \mu_1) p(y_i|\mu_0, \mu_1)$$

- according to our assumptions

$$p(x_i|y_i = 0, \mu_0, \mu_1) = N(x_i|\mu_0, \lambda_0^{-1})$$
$$p(x_i|y_i = 1, \mu_0, \mu_1) = N(x_i|\mu_1, \lambda_1^{-1})$$
$$p(y_i|\mu_0, \mu_1) = 0.5$$

- then the posterior can be written as

$$p(\theta|D) \propto \left[\prod_i p((x_i, y_i)|\mu_0, \mu_1)\right] p(\mu_0, \mu_1) =$$

$$\propto \left[\prod_i N(x_i|\mu_{y_i}, \lambda_{y_i}^{-1})\right] N(\mu_0|0,1)N(\mu_1|1,1)$$

- the posterior splits into factors $p(\mu_0|D)p(\mu_1|D)$
- both can be computed using parameters and data corresponding to one class only
- define n_j as number of vectors in class j, and s_j as sum of vectors in class j

- now each posterior factor is simply the result of observing Normal data and having a Normal prior
- using the formulas derived earlier, we obtain

$$p(\mu_0|D) = N(\mu_0|\frac{\lambda_0 s_0}{n_0\lambda_0 + 1}, (1 + n_0\lambda_0)^{-1}) = N(\mu_0|m_0, v_0)$$

$$p(\mu_1|D) = N(\mu_1|\frac{\lambda_1 s_1 + 1}{n_1\lambda_1 + 1}, (1 + n_1\lambda_1)^{-1}) = N(\mu_1|m_1, v_1)$$

- to use the classifier, we need to compute the predictive distribution of $\tilde{y}$ conditional to the vector $\tilde{x}$ to be classified:

$$p(\tilde{y}|\tilde{x}, D)$$

- since either $\tilde{y} = 0$ or $\tilde{y} = 1$, it is enough to compute

$$\begin{aligned} p(\tilde{y} = 1|\tilde{x}, D) &= \int p(\tilde{y} = 1|\tilde{x}, \mu_1) p(\mu_1|D) d\mu_1 \\ &\propto \int p(\tilde{x}|\tilde{y} = 1, \mu_1) p(\tilde{y} = 1|\mu_1) N(\mu_1|m_1, v_1) d\mu_1 \\ &\propto \int N(\tilde{x}|\mu_1, \lambda_1^{-1}) N(\mu_1|m_1, v_1) d\mu_1 \end{aligned}$$

- as a function of $\tilde{x}$, this is a Normal distribution $N(\tilde{x}|m_1, v_1 + \lambda_1^{-1})$
- therefore we get

$$p(\tilde{y} = 1|\tilde{x}, D) \propto N(\tilde{x}|m_1, v_1 + \lambda_1^{-1})$$
$$p(\tilde{y} = 0|\tilde{x}, D) \propto N(\tilde{x}|m_0, v_0 + \lambda_0^{-1})$$

- the classifier can be implemented by choosing the class j which maximizes $p(\tilde{y} = j|\tilde{x}, D)$

- a curiosity: the classifier works even if your training data comes from one class only!
- if $n_0 = 0$, then $m_0 = 0$, $v_0 = 1$, and

$$p(\tilde{y} = 0|\tilde{x}, D) \propto N(\tilde{x}|0, \lambda_0^{-1} + 1)$$

- all the above can be easily extended to multivariate data, using the corresponding multivariate formulas given in the exercises
- demo: `bclass2.R`

Priors

- Bayes requires specifying the likelihood $p(D|\theta)$ and the prior $p(\theta)$
- the likelihood seems easier, since it tells how data is generated as a function of θ
- prior can be more difficult:
  - often models contain parameters that have no easily understood meaning
  - for arbitrary priors and likelihoods, it may be impossible to compute the posterior in closed form

Conjugate Priors

- closed-form posteriors can be obtained by using *conjugate priors*
- when just the mean is unknown, Normal prior and likelihood lead to a Normal posterior
- assume that likelihood belongs to class $\mathcal{L}$ of distributions and the prior to a class $\mathcal{P}$
- if it follows that the posterior belongs to class $\mathcal{P}$, then we say that the classes $\mathcal{L}$ and $\mathcal{P}$ are conjugate

- if in the above definition $\mathcal{L} = \mathcal{P}$, then the prior is *naturally conjugate* to the likelihood
- e.g. Normal and binomial distributions are naturally conjugate
- conjugacy is practical since one can simply compute the posterior parameters (e.g. mean and variance)
- a realistic model usually results in priors that are not conjugate

- conjugacy often reveals the effect of the prior
- the prior can be interpreted as representing previously observed data
- for example in the Normal example, the posterior mean is symmetric with respect to the prior and the likelihood:

$$\mathrm{E}(\theta|y) = \frac{\lambda_0\theta_0 + \lambda y}{\lambda_0 + \lambda}$$

- prior contains the same information as having observed θ_0 with precision λ_0

Noninformative Priors

- Bayes is often criticized for its use of prior distribution
- by a suitable prior, the end result can be influenced
- but this is true of the whole model as well: the result does (and should) depend on the choice of model
- a better argument is that it is difficult to specify prior information for more or less abstract parameters when one has no idea whatsoever about the value

- there has been considerable interest to solve the question of determining *noninformative priors*
- it is difficult to even define what a noninformative prior is
- heuristic ideas don't work: for example, shouldn't all values be equally probable if we have no prior information?

- suppose the variance σ^2 is the unknown quantity of interest
- we might as well use the standard deviation σ
- use a constant prior, so any unit interval has the same probability
- since $\sigma^2 \in [0,1]$ and $\sigma \in [0,1]$ are the same events, their probability must be the same q

- constant prior says that $\sigma^2 \in [1,2]$ and $\sigma \in [1,2]$ have prior probability q as well
- but these events are not the same: $\sigma \in [1,2]$ is equivalent to $\sigma^2 \in [1,4]$
- the latter event has prior probability $3q \neq q$
- results should not depend on whether you want to use σ^2 or σ

- the *Fisher information* for a scalar θ is

$$I(\theta) = \mathrm{E}\left[\frac{\partial}{\partial\theta}\log p(y|\theta)\right]^2$$

  where the expectation is computed over $p(y|\theta)$

- the *Jeffreys' prior* $p(\theta) \propto \sqrt{I(\theta)}$ is invariant to transformations of parameters

- for example, the problem of σ vs σ^2 is solved by it

- there are many ways of defining noninformative priors, most of which disagree with Jeffreys' prior
- for scalar *location* and *scale* parameters, it seems that most ways lead to the same result
- the Jeffreys' prior for a location parameter is $p(\mu) \propto$ constant and for a scale parameter it is $p(\theta) \propto 1/\theta$ (exercise problem)

- example: table entry problem
- it has been found experimentally that the distribution of the first significant digit in tables of scale data is well described by

$$\log(1 + i^{-1}) / \log 10,\ i = 1, \ldots, 9$$

- one might expect a uniform distribution instead
- scale data: populations of cities, number of cars passing a bridge in a day etc..

- Jeffreys' prior for scale parameter is $p(\sigma) \propto \sigma^{-1}$
- consider the interval $(1, 10)$ where Jeffreys' prior is

$$p(\sigma) = \sigma^{-1} / \log 10$$

- the first digit of σ is i if $\sigma \in [i, i+1)$

- the probability of this is

$$\begin{aligned} p(\sigma \in [i, i+1)) &= \int_i^{i+1} \sigma^{-1} / \log 10 d\sigma \\ &= /_i^{i+1} \log \sigma / \log 10 \\ &= (\log i + 1 - \log i) / \log 10 \\ &= \log(1 + i^{-1}) / \log 10 \end{aligned}$$

- so Jeffreys' prior restricted to the interval $[1, 10)$ predicts the experimental result

Improper Priors

- sometimes one may encounter situations where the constant prior $p(\theta) = c$ seems the correct choice
- this cannot be normalized to a probability distribution if θ varies over an infinite interval (e.g. $\theta \in \mathbb{R}$)
- however, such priors are sometimes used: they are called *improper priors*

- in most cases they lead to proper posteriors: when they do not, the results may be wrong
- it may be better to choose a flat proper prior, and allow the likelihood to dominate the inference as it would for a constant prior
- example: a Normal prior with a very large variance may be close enough to a constant prior

LECTURE 7: 28.2.2007

BAYESIAN MODELING: MULTIVARIATE AND HIERARCHICAL MODELS

- *no teaching next week due to midterm exam week*
- next lecture is on March 14th and next exercises on March 16th

Multivariate Models

- many uncertain quantities in a vector-valued θ
- in principle, Bayesian inference proceeds as before
- specify the prior $p(\theta)$ and the likelihood $p(D|\theta)$, then compute the posterior $p(\theta|D)$

- however, some practical difficulties arise
- assume $\theta = (\theta_1, \ldots, \theta_n)'$ contains some components that are not interesting
- these are called *nuisance parameters*

- as before, we can obtain the *full posterior* $p(\theta|y)$
- at least we can write it as a function of θ using Bayes' theorem $p(\theta|y) \propto p(y|\theta)p(\theta)$
- the posterior is a multivariate function of θ
- unless conjugate priors are used, it is not generally a simple standard distribution

- in practice, various computations on the posterior are required
- for example, we might be interested in θ_1 only
- marginalize out variables $\theta_2, \ldots, \theta_n$
- done by integrating the full posterior over these variables

- the full posterior may be, and often is, impossible to integrate in closed-form
- this integration must be done in one way or another
- for example, the probability $p(0 \leq \theta_1 \leq 1|D)$ cannot be computed from the full posterior without integration

- example: Normally distributed data, mean θ and variance σ^2 unknown
- standard estimates for θ consider σ^2 as a constant
- corresponds to a known σ^2 from Bayes point of view
- unrealistic to assume that σ^2 is known

- suppose only the mean θ is interesting
- this makes σ^2 a *nuisance parameter*
- must be in the model, since it can affect results
- we don't want it in the posterior

- data: $y_1, \ldots, y_n : y_i \sim N(\theta, \sigma^2)$
- Jeffreys' prior: $p(\theta, \sigma^2) \propto \sigma^{-2}$
- the marginalized posterior $p(\theta|y_1, \ldots, y_n)$ can be solved analytically
- the result is not a Normal distribution, unlike for constant σ^2

- the full posterior is easy to write as

$$p(\theta, \sigma^2|y) \propto \sigma^{-2} \prod_i N(y_i|\theta, \sigma^2)$$

- marginalizing over σ^2 yields a *Student-t distribution*

$$p(\theta|y) = t_{n-1}(\mu_y, s^2/n)$$

where

$$\mu_y = \frac{1}{n} \sum_i y_i \text{ (sample mean)}$$

$$s^2 = \frac{1}{n-1} \sum_i (y_i - \mu_y)^2 \text{ (sample variance)}$$

- Student-t distribution has "heavy tails", meaning that $p(\theta|y) \to 0$ slowly when $|\theta - \mu_y|$ increases
- very large (or small) values of θ are more probable than for a Normal distribution
- special case of one observation is calculated in the exercises
- general result is in Gelman's book, pp. 66-69
- demo: `student.R`

- example: assume that $y_1, \ldots, y_n$ are Normally distributed: $y_i \sim N(\theta, \sigma_i^2)$
- note that *each observation y_i has its own variance σ_i^2*
- number of parameters is $n + 1$ and the number of observations is n
- it seems that there is too much freedom in the model

- posterior is obtained in closed form using a Gamma distribution for $p(\sigma_i^2)$ (demo exercise)
- the demo below compares this to estimating θ using a Normal model with constant σ^2
- Bayes result is less sensitive to an outlier
- demo: `robust.R`

Hierarchical Models

- a multivariate model with a special structure
- often prior information is easiest to represent using a hierarchical model
- the structure of the problem may also suggest a hierarchical model

- some benefits of HM's:
  - posterior can be written in a form suitable for simulation (discussed in a later lecture)
  - models can be constructed using simple distributions as building blocks

- example: coin toss
- toss $n = 20$ times and observe $y = 12$ heads
- probability of heads is θ
- you suspect the coin may be biased: denote a biased coin by $b = 1$ and unbiased by $b = 0$
- use a prior $p(b = 0) = p(b = 1) = 0.5$

- assume that biasedness defines θ as

$$p(\theta = 0.5|b = 0) = 1$$
$$p(\theta = 0.7|b = 1) = 0.5$$
$$p(\theta = 0.6|b = 1) = 0.5$$

- now the model is ready: the unknowns are b, θ and we know $y = 12$

- find the probability $p(b = 1|y)$
- the marginal posterior is $p(b|y) = \int p(b, \theta|y) d\theta$
- compute the full posterior:

$$\begin{aligned} p(b, \theta|y) &\propto p(y|b, \theta) p(b, \theta) \\ &= p(y|\theta) p(\theta|b) p(b) \\ &= \text{Bin}(y|n, \theta) p(\theta|b) p(b) \end{aligned}$$

- the equation

$$p(\theta, b|y) \propto p(y|\theta)p(\theta|b)p(b)$$

  is generally true for a hierarchical model

- i.e. data y depends on θ, θ depends on an *unknown* b, and finally there is a prior for b

- the parameter b is called a *hyperparameter*, since it only affects another parameter directly but not data

- this is what makes the model hierarchical

- since $p(\theta|b)$ is nonzero only for $\theta \in \{0.5, 0.6, 0.7\}$, compute the posterior at these values:

$$p(b = 1, \theta = 0.6|y) \propto \text{Bin}(y|n, \theta = 0.6)p(\theta = 0.6|b = 1) \approx 0.045$$
$$p(b = 1, \theta = 0.7|y) \propto \text{Bin}(y|n, \theta = 0.7)p(\theta = 0.7|b = 1) \approx 0.029$$
$$p(b = 0, \theta = 0.5|y) \propto \text{Bin}(y|n, \theta = 0.5)p(\theta = 0.5|b = 0) \approx 0.060$$

- *full posterior* says that values $b = 0, \theta = 0.5$ are most probable, implying that the coin is unbiased
- the marginal posterior is obtained from the full posterior values:

$$p(b = 1|y) \approx 0.074$$
$$p(b = 0|y) \approx 0.060$$

- this is maximized by $b = 1$, which says that the coin is probably biased

- few important things to notice:
  - the data y *does not depend on* b, when θ is known
  - this means that $p(y|\theta, b) = p(y|\theta)$
  - the MAP value of the full posterior conflicts with the MAP of the marginal posterior
  - it is possible to compute $p(\theta) = \int p(\theta|b)p(b)db$, but in general this is more complicated than using a HM

- the full posterior of a HM is often impossible to marginalize by integration
- HM's can be constructed so that they can be approximated by simulation (discussed later)
- one such method requires that we can draw random values from the conditional posteriors $p(\theta|b, y)$ and $p(b|\theta, y)$

- these can be computed as

$$p(b|\theta, y) = \frac{p(y|\theta, b)p(b|\theta)}{p(y|\theta)} = \frac{p(y|\theta)p(b|\theta)}{p(y|\theta)} \propto p(\theta|b)p(b)$$

$$p(\theta|b, y) = \frac{p(y|\theta, b)p(\theta|b)}{p(y|b)} \propto p(y|\theta, b)p(\theta|b)$$

- $p(b|\theta, y)$ can be solved if $p(b)$ is conjugate to $p(\theta|b)$
- $p(\theta|b, y)$ is the non-HM posterior (just consider b as known): conjugate prior gives also this in closed form

Data with a Hierarchical Structure

- example: average length of produced parts
- two sets of measurements (days 1 and 2)
- assume that length varies more between days than within a day
- interesting quantity is the mean length (assume constant variance)

- two easy solutions:
  - consider *all data* as identically distributed
  - consider the data from each day separately
- first case: mean length is the same on both days
- second case: mean length on day 1 has nothing to do with mean length on day 2
- both assumptions unrealistic

- the average length is assumed to change between days
- but we also assume it does not change very much
- this suggests a hierarchical model
- data distribution: day one as $N(\theta_1, \sigma^2)$ and day two as $N(\theta_2, \sigma^2)$

- lets use a prior $p(\theta_i) = N(\theta_i|\mu, \tau^2)$
- the hyperparameters μ, τ are *unknown*
- note that fixed values μ, τ^2 prevent any dependency between the data on different days
- the hierarchical model gives different results than either of the easy solutions

- lets specify the HM
- day 1 data $x_1, \ldots, x_m$ are from $N(\theta_1, \sigma^2)$ and day 2 data $y_1, \ldots, y_n$ are from $N(\theta_2, \sigma^2)$
- prior: $p(\theta_i) = N(\theta_i | \mu, \tau^2)$
- prior for hyperparameters: $p(\mu, \tau^2) \propto \tau^{-1}$

- with known σ^2, the model can be solved (Gelman's book, section 5.4)
- with unknown σ^2 and a prior $p(\sigma^2) \propto \sigma^{-2}$, a closed-form solution is impossible
- but we can compute the conditional posteriors for simulation (exercise problem)

- necessary conditional posteriors are

$$p(\theta_i|\mu,\sigma,\tau,D)$$
$$p(\mu|\theta_1,\theta_2,\sigma,\tau,D)$$
$$p(\sigma^2|\theta_1,\theta_2,\mu,\tau,D)$$
$$p(\tau^2|\theta_1,\theta_2,\mu,\sigma,D)$$

- with some thought some simplifications can be

made:

$$p(\mu|\theta_1, \theta_2, \sigma, \tau, D) = p(\mu|\theta_1, \theta_2, \tau)$$
$$p(\sigma^2|\theta_1, \theta_2, \mu, \tau, D) = p(\sigma^2|\theta_1, \theta_2, D)$$
$$p(\tau^2|\theta_1, \theta_2, \mu, \sigma, D) = p(\tau^2|\theta_1, \theta_2, \mu)$$

- it is relatively straightforward to find these distributions (exercise problem)
- one can use repeatedly the Bayes' Theorem and the product rule and eliminate unneeded parameters

Model Uncertainty

- *model selection* means selecting a model out of several candidates using data
- this is heuristic if done using only the information contained in the model(s) and data
- proper way is to use *one model* $p(\theta, D)$ as discussed before
- so there is nothing to select in theory

- what if one has more than one possible model?
- example: linear vs. nonlinear model
- if data is known to be linear, then the correct model is obviously linear
- but one might be uncertain of the linearity
- linear model underfits nonlinear data, so perhaps it is better to use a nonlinear model?

- major problem with "flat" nonlinear models: they overfit linear data
- *if the model corresponds to information about the problem,* then this overfitting is not a problem
- a nonlinear model does not generally put a high probability on the possibility that data is linear

- we can construct a hierarchical model which includes both a linear and a nonlinear model
- a hierarchical nonlinear regression model:

$$
\begin{aligned}
p(Y|\theta,\sigma^2) &= N(Y|X\theta,\sigma^2 K) \\
p(\theta|\sigma^2,\tau^2,W,\theta_0) &= N(\theta|\theta_0,\sigma^2\tau^2 W) \\
p(\theta_0) &= N(\theta_0|\mu,B)
\end{aligned}
$$

- hyperparameters τ^2, σ^2 have inverse Gamma prior distributions, and the covariance matrix W has an inverse Wishart prior distribution

- the model is constructed using conjugate priors, so that conditional posteriors are obtained in closed form (see Gelman et al. for more details on the conjugate priors)
- the nonlinearity of the model comes from the covariance matrix K
- the elements of K are inner products of feature vectors (as in SVM)
- these methods will be discussed in more detail in a later lecture

- the hyperparameters increase the uncertainty in the model
- for example, in a flat model the prior mean of $p(\theta)$ would be fixed e.g. to zero
- the hierarchical prior does not explicitly fix $p(\theta)$ as a zero-mean Normal distribution
- generally a hierarchical prior amounts to a more noninformative prior

- the above nonlinear model includes a linear model as a special case
- if the covariance matrix K is selected to be the identity matrix, then a hierarchical linear regression model is obtained
- these models may be *combined* simply by adding a binary indicator, which selects one of the models
- the indicator is a hyperparameter, and thus requires a hyperprior

- a predictive distribution can be computed as before: all parameters are integrated out, also the indicator variable
- this means that predictions will be made using both models, but the weighting is affected by data
- demo: gpmixture.R

- summary of HM's:
  - learning problem may have a hierarchical structure
  - allow models with no closed-form posterior to be constructed with closed-form conditional posteriors
  - suitable for approximation by certain simulation methods
  - combining models, adding uncertainty lead to HM's

Lecture 8: 14.3.2007

Posterior Approximation by Simulation

Posterior Approximation: Simulation Methods

- the goal of Bayesian inference is to compute the posterior $p(\theta|y)$ in a usable form
- except in simple examples and conjugate models, posterior is not a standard distribution
- in general one can obtain the unnormalized posterior as

$$p(\theta|y) \propto p(y|\theta)p(\theta)$$

- the normalizing factor is

$$p(y) = \int p(\theta, y) d\theta = \int p(y|\theta) p(\theta) d\theta$$

- in almost any realistic, non-conjugate model, this integral is not obtained in closed-form

- most uses of the posterior require integrating over it
- computing the posterior mode does not, but then the mode does not contain any probability information
- integration is unavoidable if probabilities are to be computed
- in general, expectations of the form $\mathrm{E}(h(\theta)|y)$ are needed

- expectation is an *integral* over the posterior:

$$\mathrm{E}(h(\theta)|y) = \int h(\theta)p(\theta|y)d\theta$$

- examples of functions $h(\cdot)$:
  - posterior mean: $h(\theta) = \theta$
  - posterior variance: $h(\theta) = (\theta - \mu)^2$, $\mu = \mathrm{E}(\theta|y)$
  - probability of $\theta \in A$: $h(\theta) = 1$ when $\theta \in A$, zero otherwise

- various approximation methods can be used to compute the integrals
- next week, parameteric approximations are discussed
- the true posterior is approximated as some tractable parametric distribution, for example Normal distribution
- today, more general approximation is discussed

- *simulation*: for our purposes, drawing samples from the posterior distribution
- sort of "reverse estimation": in estimation, a distribution is unknown but data generated from it are known
- in simulation, we know the posterior as a function of θ and then generate data from it
- the simulated data carry information about the posterior

- as in estimation, the data can be used to approximate expectations over the posterior
- for example, drawing m samples from a Normal distribution $N(\mu, \sigma^2)$ can be used to estimate the normal mean and variance using the familiar estimators
- but we can use the samples to estimate any expectation $\mathrm{E}(h(\theta)|y)$

- assume that we have drawn $\theta^1, \ldots, \theta^m$ from the posterior $p(\theta|y)$
- the expectation $\mathrm{E}(h(\theta)|y)$ can be approximated by a a *Monte Carlo integral*

$$\int h(\theta)p(\theta|y)d\theta \approx \frac{1}{m}\sum_{i=1}^{m} h(\theta^i)$$

- if the samples are independent, it is easy to see that
  1. the mean of the approximation is $\mathrm{E}(h(\theta)|y)$
  2. the variance is proportional to $1/m$

- there are several ways of obtaining the samples
- *direct simulation* is possible for many standard distributions
- most mathematical software (R, MATLAB etc..) have functions for drawing samples
- direct simulation is mainly based on uniformly distributed pseudorandom numbers and their transformations

- since direct simulation of a discrete distribution is easy, why not use a piecewise constant approximation to the posterior?
- this means dividing the θ space into disjoint subsets and setting $p(\theta|y)$ to a constant value in each subset
- normalization would be obtained trivially, since the difficult integral would simply be a finite sum
- not generally a good solution: posterior mass can be very concentrated in high-dimensional problems, so this approach is very unreliable
- demo: `sampling.R`

- fortunately, there are ways of obtaining simulated samples from an unnormalized distribution
- the methods discussed below have very appealing properties in theory
- but some important weaknesses limit their use in practice
- some basic methods are introduced, and their properties explored
- notation warning: $p(\theta|y)$ is generally assumed to be *unnormalized* in the rest of this lecture.

Importance Sampling

- importance sampling approximates integrals directly
- assume we want to compute (using an unnormalized posterior)

$$\mathrm{E}(h(\theta)|y) = \int h(\theta)p(\theta|y)d\theta / \int p(\theta|y)d\theta$$

- in importance sampling, a distribution $g(\theta)$ is chosen so that it is easy to draw values from

- insert $1 = g(\theta)/g(\theta)$ to obtain

$$\mathrm{E}(h(\theta)|y) = \frac{\int h(\theta)\frac{p(\theta|y)}{g(\theta)}g(\theta)d\theta}{\int \frac{p(\theta|y)}{g(\theta)}g(\theta)d\theta}$$

- this is equal to

$$\frac{\mathrm{E}_g\left(h(\theta)\frac{p(\theta|y)}{g(\theta)}\right)}{\mathrm{E}_g\left(\frac{p(\theta|y)}{g(\theta)}\right)}$$

  where E_g is an expectation over $g(\theta)$

- simulate $g(\theta)$ to obtain samples $\theta^i,\ i = 1, \ldots, m$ and compute the *importance ratios*

$$w^i = \frac{p(\theta^i|y)}{g(\theta^i)},\ i = 1, \ldots, m$$

- both p and g can be unnormalized
- then approximate the expectations to obtain

$$\mathrm{E}(h(\theta)|y) \approx \frac{\sum_i w^i h(\theta^i)}{\sum_i w^i}$$

- properties:
  - works if $g(\theta)$ is somewhat proportional to $h(\theta)p(\theta|y)$: if $g \ll hp$, then very few simulated values are obtained where they are needed
  - both $p(\theta|y)$ and $g(\theta)$ can be unnormalized
  - can use the same simulated set for new $p(\theta|y)$ and/or new $h(\theta)$
  - does *not* give simulated values from the posterior!

Rejection Sampling

- generates independent samples drawn from the posterior
- again choose a distribution $g(\theta)$ which is easy to sample from (can be unnormalized)
- the importance ratio must be bounded:

$$\frac{p(\theta|y)}{g(\theta)} \leq M \text{ for all } \theta$$

- rejection sampling proceeds as follows:
  1. draw a value θ^* from $g(\theta)$, and u from a uniform distribution on $[0, 1]$
  2. accept the sample $\theta^i = \theta^*$ if $u \leq p(\theta^*|y)/Mg(\theta^*)$, otherwise go back to step 1
- repeating the two steps, one obtains a set of samples θ^i which are distributed as $p(\theta|y)$
- straightforward computation shows this

$$
\begin{aligned}
p(\theta^i \leq x) &= p(\theta^* \leq x | u \leq \frac{p(\theta^*|y)}{Mg(\theta^*)}) \\
&= \frac{p(\theta^* \leq x, u \leq \frac{p(\theta^*|y)}{Mg(\theta^*)})}{p(u \leq \frac{p(\theta^*|y)}{Mg(\theta^*)})} \\
&= \frac{\int_{-\infty}^{x} \int_{0}^{p(\theta|y)/Mg(\theta)} du g(\theta) d\theta}{\int_{-\infty}^{\infty} \int_{0}^{p(\theta|y)/Mg(\theta)} du g(\theta) d\theta} \\
&= \frac{1/M \int_{-\infty}^{x} p(\theta|y) d\theta}{1/M \int_{-\infty}^{\infty} p(\theta|y) d\theta} \\
&= \int_{-\infty}^{x} p(\theta|y) d\theta = p(\theta \leq x|y)
\end{aligned}
$$

- as in importance sampling, the distribution $g(\theta)$ has practical implications
- if the proposal distribution $g(\theta)$ is small where the real distribution is large, the constant M is large
- then a very large proportion of simulated points will be rejected (exercise)

Markov Chain Monte Carlo

- direct and rejection sampling define a random process

$$\theta^0, \theta^1, \theta^2, \ldots \tag{1}$$

  which is an i.i.d. sequence of $p(\theta|y)$-distributed variables

- often it is not practical to use the above methods to obtain i.i.d. samples from the posterior
- more practical simulation methods are based on Markov Chains, which result in nonindependent samples

- a random process $\theta^0, \theta^1, \ldots$ is a *Markov Process*, if it has the Markov property:

$$p(\theta^{i+1}|\theta^i, \theta^{i-1}, \ldots) = p(\theta^{i+1}|\theta^i), \ \ \forall i \geq 0$$

- this means that the current *state* θ^i completely defines the conditional distribution of future states
- i.e. the process has a very short memory

- to define a Markov Chain, we need a transition distribution

$$T(\theta^{i+1}|\theta^i) = p(\theta^{i+1}|\theta^i)$$

- given an initial distribution $p(\theta^0)$, every θ^i has an *unconditional* distribution $p(\theta^i)$
- the Markov Chain has a *stationary distribution* if $p(\theta^{i+1}) = p(\theta^i)$ for all $i \geq 0$

- Markov Chain Monte Carlo (MCMC) means simulating a Markov Chain with a desired stationary distribution
- in our case, we usually want the stationary distribution to be the posterior
- direct and rejection sampling are trivially MCMC (with no memory)
- in general, MCMC results in a sequence of simulated values *which are not independent*

- some needed properties (see refs on webpage):
  - stationary distribution $p(\theta|y)$: correct distribution achieved by construction, stationarity follows from properties below
  - $\pi-$irreducibility: for any set A with $p(\theta \in A|y) > 0$ and any starting value θ^0, there is an integer $n = n(\theta^0, A)$ so that $\mathbb{P}(\theta^n \in A) > 0$
  - Harris-recurrence: given a $\pi-$irreducible chain, for any set B with $p(\theta \in B|y) > 0$, it must hold that $\mathbb{P}(\theta_i \in B \text{ happens infinitely often}|\theta^0) = 1 \ \ \forall \theta^0$
  - aperiodicity: means that the chain cannot "cycle" through a sequence of disjoint sets

Metropolis-Hastings Algorithm

1. pick an initial value θ^0, set $i = 0$
2. draw θ^* from a *jumping distribution* $J(\theta^*|\theta^i)$
3. compute the *jumping ratio* for θ^*:

$$r = \frac{p(\theta^*|y)J(\theta^i|\theta^*)}{p(\theta^i|y)J(\theta^*|\theta^i)}$$

4. set the next simulated θ^{i+1} as

$$\theta^{i+1} = \begin{cases} \theta^*, & \text{with probability } \min(r,1) \\ \theta^i, & \text{with probability } 1 - \min(r,1) \end{cases}$$

5. set $i = i + 1$ and go to step 2

- note the difference to rejection sampling: if θ^* is "rejected", the previous value θ^i is *repeated* instead of discarded
- if the jumping distribution is symmetric ($J(\theta^*|\theta^i) = J(\theta^i|\theta^*)$), then $r = p(\theta^*|y)\,/\,p(\theta^i|y)$ and the algorithm is called the Metropolis algorithm
- M-H and its special cases are widely used due to their properties which follow from theory of Markov Chains

- M-H has the posterior as the stationary distribution (demo exercise)
- $\pi-$irreducibility is case-specific (usually a positive jumping distribution guarantees it)
- Harris-recurrence follows from π-irreductibility for M-H
- aperiodicity (case-specific, holds for most jumping distributions)

- some theoretical results for $M-H$:
  - Harris-recurrent, $\pi-$irreducible, aperiodic chain with a stationary distribution converges to the stationary distribution from any starting value
  - above assumptions and $\mathrm{E}(|h|) < \infty$ are enough to show that

$$\lim \frac{1}{n} \sum_{j=1}^{n} h(\theta^j) = \mathrm{E}(h(\theta)) \text{ almost surely}$$

- M-H solves the computational problem of Bayes *in theory*
- in practice, the initial value biases early samples and the jumping distribution affects convergence speed
- long jumps are rejected, and short jumps converge slowly
- demo: `metropolis.R` and `metropolis2.R`

- lots of heuristic simulation methods have been proposed, perhaps because MCMC methods are computationally intensive and somewhat brute-force solutions
- either they are special cases of M-H, or not. In the latter case, it may be difficult to know that they will converge to $\mathrm{E}(h(\theta)|y)$
- lets go through a couple of special cases of M-H

Gibbs Sampler

- if θ is multivariate and full conditional posteriors are known and easy to sample from, then it is possible to use the Gibbs sampler
- one must be able to simulate all full conditional posteriors

$$p(\theta_k|y, \theta_1, \ldots, \theta_{k-1}, \theta_{k+1}, \ldots, \theta_m)$$

- one iteration of the Gibbs sampler can be defined as follows:
  1. the full sample $\theta^i = (\theta_1^i, \ldots, \theta_m^i)'$ has been drawn
  2. the next component θ_1^{i+1} is drawn from $p(\theta_1 | y, \theta_2^i, \ldots, \theta_m^i)$
  3. then θ_2^{i+1} is drawn from $p(\theta_2 | y, \theta_1^{i+1}, \theta_3^i, \ldots, \theta_m^i)$
  4. obtain all components of θ^{i+1} as above
  5. increase i to $i+1$ and start again

- Gibbs is a special case of M-H with jumps changing one component at a time
- stationary distribution is the posterior since Gibbs is M-H
- the jumping distribution is obtained from the full conditional posteriors
- the acceptance ratio turns out to be one, so all jumps are accepted (see Gelman, 1st edition, p. 328)
- demo: `gibbs.R`

- independence sampler: set $J(\theta^*|\theta^i) = g(\theta^*)$
- jumping distribution is *independent* of the previous state θ^i
- jumping ratio is

$$r = \frac{p(\theta^*|y)g(\theta^i)}{p(\theta^i|y)g(\theta^*)} = \frac{w(\theta^*)}{w(\theta^i)}$$

- w is the *importance ratio* seen in importance sampling
- if $g(\theta) = p(\theta|y)$, then all jumps are accepted and we obtain direct sampling

- Langevin algorithms include a term that directs jumps toward posterior modes
- example:

$$J(\theta^*|\theta^i) \propto \exp(-\frac{1}{2\sigma^2}\|\theta^* - \theta^i - \frac{\sigma^2}{2}(\log p(\theta^i|y))'\|^2)$$

- the jumping distribution is a Normal distribution with mean $\theta^i + \frac{\sigma^2}{2}(\log p)'$
- the derivative of the log-posterior moves the mean towards a (local) posterior mode, thus hopefully achieving faster convergence

- in theory, the fact that MCMC does not result in independent samples is not a problem
- this depends on the convergence of the MC integral despite dependent samples
- but asymptotic results are just that: "asymptotic" does not mean "holds for a very large but finite number of samples"
- in practice, it is important to known how good is the MC approximation computed from the finite set of samples

- dependent samples make it difficult to say anything general about the finite-sample properties of MC approximation
- intuition from iid samples can go wrong: MCMC can spend a lot of time in some small area of the posterior, then jump to another are and spend a lot of time in there
- Monte Carlo integral can have a large error *unless the samples represent the posterior well as a set*
- for example, having samples from one mode of a bimodal distribution does not yield good MC approximations

MCMC Convergence

- here, convergence is nonrigorously defined as "samples begin to represent the posterior well enough"
- in practice, inital value θ^0 biases the early samples
- the chain spends some time simulating samples that do not represent the correct posterior well, as seen in the demo

- what to do about the early samples?
- run the chain until convergence and discard the early samples
- after convergence, keep all samples (or take every k:th sample)
- but knowing when the chain has converged is not generally possible

- in practice, one should be able to
  1. find out the convergence speed, and use it to estimate how many samples need to be discarded
  2. examine the samples as they are simulated, and try to decide whether the chain has converged
- no completely general method has not been found to date

- some heuristics:
  - run parallel chains from different starting points (avoid local modes)
  - compare the variance within chains and between chains (if between chains much larger, then no convergence)
  - simulate starting point(s) from a crude posterior approximation (avoid slow convergence towards posterior mode(s))

Lecture 9: 21.3.2007

Posterior Approximation: Laplace and Variational Methods

Parametric Approximation

- some ways to compute the posterior:
  - normalizable closed-form (rarely possible)
  - MCMC (works in theory, convergence is generally very slow)
  - parametric approximation

- closed-form is obviously a good choice when possible
- MCMC can become infeasible in 'large' problems
- MCMC works if computational costs and memory costs are ignored
- since costs always matter, parametric approximation can be a practical compromise between accuracy and cost of implementing and computing the solution

- this compromise is generally not optimal
- approximate posterior is a wrong posterior, so the choice is between different wrong solutions
- since wrong solutions have consequences, their cost should also be considered
- a probability model does not contain information about such costs

- therefore any single approximation even for the same model cannot be generally optimal
- for example, the same model might describe the absorption of aspirin and the absorption of an antibiotic. Consequences of errors are certainly different.
- in practice, approximations have to be made even if non-probability costs are not explicity considered
- it is better to think about different approximation methods as heuristics with varying properties

- pros/cons of parametric approximation
  - simple representation of the posterior
  - possibly low computational cost
  - overfitting (or underfitting)
  - accurate approximation often means high computational cost

Point Estimation

- simplest posterior approximation method is *point estimation*
- meaning: choose a single value θ_0, and use it to represent the posterior
- formally the point estimate is the Dirac function $\delta(\theta - \theta_0)$

- examples:
  - posterior mean: $\theta_0 = \mathrm{E}(\theta|y)$
  - posterior median (cont θ): $\mathbb{P}(\theta \leq \theta_0) = 0.5$
  - posterior mode: $\theta_0 = \mathrm{argmax}_{\theta_0}\, p(\theta_0|y)$
  - minimum mean-square: $\theta_0 = \mathrm{argmin}_{\theta_0}\, \mathrm{E}((\theta - \theta_0)^2|y)$

- for some approximations, many of these are equivalent
- e.g. a Normal distribution has the same mean, mode, median and MSE (computed wrt to the Normal)
- the posterior mode has some properties that warrant further discussion
- if $p(\theta) \propto c$, then the posterior mode is the maximum likelihood estimate

- in general, the posterior mode is called the MAP-estimate (maximum a posteriori)
- there are certain classes of models, for which the posterior mode can be found iteratively using the EM-algorithm
- more on this in the next lecture

- "Bayes Central Limit Theorem": if

$$y_1, y_2, \ldots, y_n \text{ is i.i.d. from } p(y|\theta_0) \implies$$
$$p(\theta|y_1, \ldots, y_n) \to N(\theta|\theta_0, (nI(\theta_0))^{-1})$$

  as n goes to infinity (see Gelman for assumptions)

- $I(\theta_0)$ is the Fisher information (used earlier to define the Jeffreys' prior)

- the covariance $(nI)^{-1}$ goes to zero when $n \to \infty$

- motivates the use of posterior mode, although this holds only asymptotically

Laplace Approximation

- point estimates ignore *posterior uncertainty*
- variance can be added to a point estimate to obtain *local* uncertainty
- Bayes CLT suggests approximating the posterior by a Normal distribution
- the mean is the posterior mode, and the variance is fitted to the posterior at the mode

- suppose we have found the posterior mode θ_0
- Taylor-expand $\log p(\theta|y)$ at the mode:

$$\log p(\theta|y) = \log p(\theta_0|y) + (\log p(\theta_0|y))'(\theta - \theta_0) + \frac{1}{2}(\log p(\theta_0|y))''(\theta - \theta_0)^2 + \ldots$$

- posterior is maximized at θ_0 so

$$(\log p(\theta_0|y))' = 0$$

- we also know that $(\log p(\theta_0|y))'' < 0$ since θ_0 is the mode
- dropping higher terms and taking exponents gives

$$p(\theta|y) \approx p(\theta_0|y) \exp(-\frac{1}{2}(-(\log p(\theta_0|y))'')(\theta - \theta_0)^2)$$

- this is a Normal distribution of θ

- this works even if $p(\theta|y)$ is not normalized, since the exponent tells us the mean and the variance
- we obtain $p(\theta|y) \approx N(\theta|\theta_0, \sigma^2)$
- the mean is obviously θ_0
- a Normal density has the exponent $-\frac{1}{2\sigma^2}(\theta - \theta_0)^2$

- therefore the variance is

$$\sigma^2 = \frac{-1}{(\log p(\theta_0|y))''}$$

- the variance does not depend on the normalization of $p(\theta|y)$ due to the logarithm and the differentiation
- if needed, one can compute the normalization factor

$$\frac{1}{\sqrt{2\pi\sigma^2}} = \sqrt{\frac{-(\log p(\theta_0|y))''}{2\pi}}$$

- one can also Laplace-approximate integrals

$$\mathrm{E}(h(\theta)|y) = \int h(\theta)p(\theta|y)d\theta$$

- use the same idea, but approximate the integrand $z(\theta) = h(\theta)p(\theta|y)$ around its mode θ_0
- Taylor-expand $\log(z(\theta))$ and take exponent:

$$z(\theta) \approx z(\theta_0)\exp(\frac{1}{2}[\log(z(\theta_0))]''(\theta - \theta_0)^2)$$

- this is an unnormalized Normal distribution with mean θ_0 and variance

$$\sigma^2 = \frac{1}{-(\log(z(\theta_0)))''}$$

- the integral is approximated as follows:

$$\begin{aligned}
\mathrm{E}(h(\theta)|y) &\approx \int z(\theta_0) \exp(-\frac{1}{2\sigma^2}(\theta - \theta_0)^2) d\theta \\
&= z(\theta_0)\sqrt{2\pi\sigma^2} \int \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{1}{2\sigma^2}(\theta - \theta_0)^2) d\theta \\
&= z(\theta_0)\sqrt{2\pi\sigma^2}
\end{aligned}$$

- the last integral vanished since its over $N(\theta|\theta_0, \sigma^2)$, which is a normalized probability distribution
- the Laplace-approximation is

$$\mathrm{E}(h(\theta)|y) \approx z(\theta_0)\sqrt{\frac{2\pi}{-(\log(z(\theta_0)))''}}$$

- note that θ_0 is the mode of $z(\theta)$ and not $p(\theta|y)$

- Laplace approximation allows local model averaging around the model: i.e. it describes the posterior uncertainty locally
- however, the approximation is centered on posterior mode so most point estimation problems remain
- when there are multiple modes and the highest one is a bad solution, Laplace approximation fails just as point estimates do
- demo: `laplace.R`

Kullback-Leibler Divergence

- posterior approximation can be thought like this:
  1. the true posterior is $p(\theta|y)$
  2. the approximate posterior is $q(\theta)$, constrained in some way (e.g. a parametric distribution)
  3. the problem is to find $q(\theta)$ that is 'as close as possible' to $p(\theta|y)$
- how to measure the closeness of distributions?

- denote a measure between distributions as $D(p, q)$
- there is no unique way of choosing the measure: the reasons are the same that discredit point estimation
- for example if $D(p, q) \ll D(p, q')$, it is possible that q causes a very costly error while q' doesn't
- probabilites cannot measure such costs so there is no generally optimal measure

- *Kullback-Leibler divergence* is an information-theoretic measure between distributions
- KL divergence is defined as

$$D(q\|p) = \int q(\theta) \log \frac{q(\theta)}{p(\theta)} d\theta$$

- some properties: $D(p\|q)$ is
  - nonnegative: $D(p\|q) \geq 0$
  - equal to zero iff $q = p$
  - not symmetric: $D(p\|q) \neq D(q\|p)$

- there is a coding interpretation of KL divergence
- a discrete random variable x can be coded with average number of bits equal to its *entropy*
- each x_i has codelength $-\log p(x_i)$
- entropy is $H(x) = -\sum_i p(x_i) \log p(x_i)$
- assume we guess the distribution of x as $q(x)$

- the average codelength must be computed over the *real distribution*, so we obtain

$$H_q(x) = \sum_i -p(x_i) \log q(x_i)$$

- the KL divergence $D(p\|q)$ is now

$$D(p\|q) = \sum_i p(x_i) \log \frac{p(x_i)}{q(x_i)} = -H(x) + H_q(x)$$

- then $H(x) + D(p\|q) = H_q(x)$, which means that the KL divergence is the number of extra bits caused by using the wrong distribution $q(x)$

Variational Approximation

- variational approximation means finding $q(\theta)$ which minimizes $D(q\|p)$, and using it as the approximate posterior
- the coding interpretation involved $D(p\|q)$, not $D(q\|p)$
- variation approximation uses $D(q\|p)$ because integration is performed over $q(\theta)$, not over the intractable $p(\theta|D)$
- since KL divergence is not symmetric, in general, $D(q\|p) \neq D(p\|q)$

- example: discrete $\theta \in \{\theta_0, \theta_1, \dots\}$
- point estimate: $q_0, q_1, \dots$ with $q_i \in \{0, 1\}$, $\sum_i q_i = 1$
- posterior: $p_0, p_1, \dots, \sum_i p_i = 1$, $p_i \geq 0$
- KL divergence is

$$\begin{aligned} D(q\|p) &= \sum_i q_i \log \frac{q_i}{p_i} \\ &= \log \frac{1}{p_j} \text{ (only one nonzero } q_j) \end{aligned}$$

- min D is obtained by maximizing p_j, so we get the posterior mode as the result

- the variational approximation in general tries to match the posterior area with the largest probability mass
- naturally the way the probability mass is measured depends on the approximating distribution $q(\theta)$
- in the posterior mode example, the posterior mass is measured at a single point
- if $q(\theta)$ is for example a Normal distribution, then the results are different

- example: multimodal posterior
- posterior mode is at a point where $p(\theta|y)$ is maximized
- but if the mode is on top a very narrow peak, it may be that the posterior mass contained in that peak is small
- variational approximation generally finds the peak with most posterior mass (exercise)
- demo: `varnorm.R`

- in variational approximation $q(\theta)$ must be constrained: otherwise $q(\theta) = p(\theta|y)$ and we get nowhere
- if $q(\theta)$ is a parametric distribution, minimizing KL-divergence is a parametric optimization problem
- in *free form approximation* it is only assumed that $q(\theta)$ is factorizable: $q(\theta) = q(\theta_1)q(\theta_2)\ldots q(\theta_d)$
- this makes minimizing KL divergence a functional optimization problem

Freeform Approximation Example

- independent data $y_1, \ldots, y_n$ from $N(\mu, \lambda^{-1})$
- λ is the *precision* (inverse of variance)
- the Jeffreys' prior for the mean is $p(\mu) \propto 1$
- for the variance σ^2, Jeffreys' prior is $p(\sigma^2) \propto \sigma^{-2}$
- since $\lambda = \sigma^{-2}$, we have to transform the prior:

$$p(\lambda) = p(\sigma^2) \left| \frac{d\sigma^2}{d\lambda} \right| \propto \lambda \left| \frac{d\sigma^2}{-\lambda^2 d\sigma^2} \right| = \lambda^{-1}$$

- posterior is

$$p(\mu, \lambda | y) \propto p(y | \mu, \lambda) \lambda^{-1}$$

- denote $\overline{y} = \frac{1}{n} \sum_i y_i$
- solve the variational approximation
  $q(\mu, \lambda) = q(\mu) q(\lambda)$

- assume that $q(\lambda)$ is known
- find $q(\mu)$ that minimizes $D(q\|p)$, keeping $q(\lambda)$ fixed
- to simplify notation, all constant terms wrt the current minimization are dropped (e.g $\log q(\lambda)$ in the next slide)
- the notation E_λ means an expectation over $q(\lambda)$
- E_μ is defined correspondingly

$$
\begin{aligned}
D(q\|p) &= \mathrm{E}_\mu \, \mathrm{E}_\lambda \log \frac{q(\mu)}{p(y|\mu,\lambda)} \\
&= \mathrm{E}_\mu \, \mathrm{E}_\lambda \left[\log q(\mu) + \frac{\lambda}{2} \sum (y_i - \mu)^2 \right] \\
&= \mathrm{E}_\mu \left(\log q(\mu) + \frac{1}{2} \left[\mathrm{E}_\lambda \, \lambda \right] \sum_i (y_i - \mu)^2 \right) \\
&= \mathrm{E}_\mu \left(\log q(\mu) + \frac{1}{2} A \sum_i (y_i - \mu)^2 \right)
\end{aligned}
$$

- the quantity A is known, since $\mathrm{E}_\lambda \, \lambda = \int \lambda q(\lambda) d\lambda$

- we can write

$$E_\mu \left(\log q(\mu) + \frac{1}{2} A \sum_i (y_i - \mu)^2 \right) = \mathrm{E}_\mu \log \frac{q(\mu)}{p'(\mu)}$$
$$= D(q(\mu) \| p'(\mu))$$

- straightforward calculation gives

$$p'(\mu) = N(\mu | \overline{y}, \lambda_1^{-1}), \;\; \lambda_1 = An$$

- setting $q(\mu) = p'(\mu)$ minimizes $D(q \| p')$ and thus minimizes the original KL-divergence
- first step has been completed and $q(\mu)$ is known

- next, find $q(\lambda)$ by minimizing the KL-divergence

$$\begin{aligned} D(q\|p) &= \mathrm{E}_\lambda \, \mathrm{E}_\mu \log \frac{q(\lambda)}{p(y|\mu,\lambda)p(\lambda)} \\ &= \mathrm{E}_\lambda \left(\log \frac{q(\lambda)}{p(\lambda)} + \mathrm{E}_\mu \left[-\log p(y|\mu,\lambda) \right] \right) \end{aligned}$$

- the expectation over $q(\mu)$ gives a function of λ:

$$\begin{aligned} \mathrm{E}_\mu \left[-\log p(y|\mu,\lambda) \right] &= -\frac{n}{2} \log \lambda + \frac{1}{2} \lambda (n\lambda_1^{-1} + S) \\ &= F(\lambda) \\ S &= \sum_i (y_i - \overline{y})^2 \end{aligned}$$

- then the KL-divergence is

$$D(q\|p) = \mathrm{E}_\lambda \left(F(\lambda) + \log \frac{q(\lambda)}{p(\lambda)} \right)$$
$$= \mathrm{E}_\lambda \log \frac{q(\lambda)}{p'(\lambda)} = D(q\|p')$$

- this results in a distribution

$$\log p'(\lambda) = -F(\lambda) + \log p(\lambda)$$
$$= \frac{n}{2} \log \lambda - \frac{1}{2}\lambda(n\lambda_1^{-1} + S) + \log p(\lambda) \implies$$
$$p'(\lambda) = \lambda^{n/2-1} \exp(-\frac{\lambda}{2}(n\lambda_1^{-1} + S))$$

- the Gamma distribution $\Gamma(a, b)$ has density

$$\Gamma(\theta|a, b) \propto \theta^{a-1} \exp(-b\theta)$$

- therefore $q(\lambda) = \Gamma(\frac{n}{2}, \frac{1}{2}(n\lambda_1^{-1} + S))$

- each step started from an assumption that the other distribution is known
- since in the beginning neither distribution is known, these steps should be iterated several times
- the parametric form of each distribution is obtained in the first iteration but the parameter values can change
- demo: `freeform.R`

Lecture 10: 28.3.2007

Latent Variable Models

Latent Variable Models

- we have used regression as an example of a learning problem
- data includes both inputs x and corresponding outputs y
- if some or all inputs x are *unobserved*, then we end up with a latent variable model

- a toy example: learning a bit-to-bit function
- observe input $x(n) \in \{0,1\}$ and output $y(n) \in \{0,1\}$
- suppose you have $x(0) = 0, y(0) = 0$ and $x(1) = 1, y(1) = 1$
- what's $p(y(2)|x(2))$?

- it seems that the situation is symmetric wrt 0 and 1
- but if there is a latent input $z(n)$ so that

$$p(y(n) = 1 | x(n) = 1 \text{ or } z(n) = 1) = 1$$
$$p(y(n) = 0 | x(n) = 0 \text{ and } z(n) = 0) = 1$$

- then $y(2) = 1$ is more probable than $y(2) = 0$
- point: existence of the input $z(n)$ affects the prediction *even if* $z(n)$ is never observed

- some ways of coming up with a latent variable model:
  - such a model is realistic in the problem (e.g. ICA, image deblurring or superresolution)
  - model does not fit the data: perhaps some unobserved input causes this
  - adding latent variables helps solving the model (mixture models)

- in neurocomputing, unsupervised learning methods are latent variable models
- clustering can be thought of as a latent variable model
- the latent variable would define the cluster index for each observation
- PCA explains the outputs as linear combinations of unobserved principal components

- in the computer assignment the observed data is the output and the high-resolution image the unobserved input
- the pixel intensities of the high-res image are latent variables
- qualitatively speaking, latent variables are "data-like" parameters
- it is natural to think of pixel intensities as data instead of parameters

- unsupervised learning methods in general are taught in other courses in detail
- lets concentrate on *mixture models*, especially Normal Mixtures
- benefits: many latent variable models can be thought of as mixture models, computational benefits

Normal Mixtures

- consider data $y = \{y_1, \ldots, y_n\}$ where each y_i has a *multimodal distribution* with m modes
- lets attempt to model the data using m different Normal distributions $N(\mu_j, \sigma_j^2)$
- we'll construct the mixture model by assuming that each y_i has been generated by a specific *mixture component* $N(\mu_j, \sigma_j^2)$
- y_i is not Normally distributed, because *we don't know which component has generated it*

- what is the distribution of y_i?
- lets add *latent variables*, which indicate the guilty distribution
- *indicators* $L_{ij} \in \{0, 1\}$:

  $L_{ij} = 1$ means that y_i belongs to component j

  $\sum_{j=1}^{m} L_{ij} = 1$, each data point belongs to exactly one component

- lets define λ_j as the probability that $L_{ij} = 1$ *when* y_i *is unknown*
- a benefit of the indicators is that they simplify the model
- the *complete-data likelihood* is (exercise)

$$p(y_i, L_{i\cdot}|\theta, \lambda) = \prod_{j=1}^{m} (\lambda_j N(y_i|\mu_j, \sigma_j^2))^{L_{ij}}$$

- integrate out the latent variables so that $L_{ik} = 1$ when $k = 1$:

$$p(y_i|\theta, \lambda) = \sum_{k=1}^{m} \prod_{j=1}^{m} (\lambda_j N(y_i|\mu_j, \sigma_j^2))^{L_{ij}}$$
$$= \sum_{k=1}^{m} \lambda_k N(y_i|\mu_k, \sigma_k^2)$$

- observed data has distribution

$$p(y|\theta,\lambda) = \prod_{i=1}^{n} \sum_{j=1}^{m} \lambda_j N(y_i|\mu_j,\sigma_j^2)$$

- when the product is expanded, the result is a sum with m^n terms
- this cannot be even maximized in closed form

- full conditional posteriors are simpler (constant prior for θ):

$$p(\theta|y, L) \propto \prod_j \prod_{i:L_{ij}=1} N(y_i|\mu_j, \sigma_j^2) \text{ (exercise)}$$

$$p(L_{ik} = 1|\theta, y, \lambda) = \frac{\lambda_k N(y_i|\mu_k, \sigma_k^2)}{\sum_j \lambda_j N(y_i|\mu_j, \sigma_j^2)}$$

- the probabilities λ_j are also unknown: the can be easily estimated given the other parameters (exercise)

- with enough mixture components, any distribution can be closely approximated
- θ and λ cannot be solved in closed-form
- Gibbs Sampler is possible due to full conditional posteriors
- but it has poor convergence properties

- example: simulate a two-component mixture model

$$p(y_i|\theta,\lambda) = \lambda_1 N(y_i|\mu_1,1) + \lambda_2 N(y_i|\mu_2,1)$$

  using Gibbs Sampler

- demo shows that there are serious convergence problems

- demo: `gibbsmixture.R`

- the posterior mode can be approximated using Expectation Maximization
- first, lets motivate it by deriving something simpler
- consider the model

$$p(y|\theta) = \prod_{i=1}^{n} p(y_i|\theta) = \prod_{i=1}^{n} \left[\lambda_1 N(y_i|\mu_1, \sigma^2) + \lambda_2 N(y_i|\mu_2, \sigma^2)\right]$$

where σ^2 is known

- find μ_1 and μ_2 that maximize the log-likelihood

$$\log p(y|\theta,\lambda) = \sum_{i=1}^{n} \log \Big(\lambda_1 N(y_i|\mu_1,\sigma^2) + \lambda_2 N(y_i|\mu_2,\sigma^2) \Big)$$

- differentiate it (denoted by operator d):

$$\begin{aligned} d \log p(y|\theta,\lambda) &= \sum_{i=1}^{n} d \log p(y_i|\theta,\lambda) \\ &= \sum_{i=1}^{n} [p(y_i|\theta,\lambda)]^{-1} d p(y_i|\theta,\lambda) \end{aligned}$$

- to differentiate wrt μ_j, note that

$$\frac{\partial p(y_i|\theta,\lambda)}{\partial \mu_j} = \lambda_j N(y_i|\mu_j,\sigma^2)\sigma^{-2}(y_i-\mu_j)$$

- then we obtain as the derivative

$$\frac{\partial \log p(y|\theta)}{\partial \mu_j} = \sum_{i=1}^{n} \sigma^{-2}(y_i-\mu_j)\tau_{ij}$$

$$\tau_{ij} = \frac{\lambda_j N(y_i|\mu_j,\sigma^2)}{\sum_k \lambda_k N(y_i|\mu_k,\sigma^2)}$$

- weight τ_{ij} is close to one when y_i is much closer to μ_j than any μ_k with $k \neq j$
- also, $\tau_{ij} = p(L_{ij} = 1|\theta, y, \lambda)$
- can easily be computed *if we know* $\mu_1, \mu_2, \lambda_1, \lambda_2$

- derivative

$$\frac{\partial \log p}{\partial \mu_j} = \sum_{i=1}^{n} \frac{y_i - \mu_j}{\sigma^2} \tau_{ij}$$

  is a weighted average of $(y - \mu_j)/\sigma^2$

- e.g. if y is on average larger than μ_j, the derivative will be positive

- the average is easy to compute *if all* τ_{ij}*:s are known*

- lets maximize $\log p(y|\theta)$ using *Newton-Rhapson iteration*

$$\mu_{new} = \mu_{old} - (\log p)' / (\log p)''$$

- first derivative was just computed
- the second derivative is approximately computed in the exercises

- start by selecting initial values for the means and λ:s
- then compute τ_{ij}:s
- then compute the derivatives and use Newton-Rhapson

$$\mu_{j,new} = \frac{\sum_i p(L_{ij} = 1|\theta, y_i) y_i}{\sum_i p(L_{ij} = 1|\theta, y_i)} = \frac{\sum_i \tau_{ij} y_i}{\sum_i \tau_{ij}}$$

- $\mu_{j,new}$ is an a weighted average of data
- the weights τ_{ij} emphasise some observations more than others
- for data not from component j, $\tau_{ij} \approx 0$
- average is mostly over data from component j
- demo: `twonormal.R`

Expectation-Maximization Algorithm

- the previous derivation suggests that the posterior mode is a computationally feasible approximation
- there is a general algorithm for finding the posterior mode for latent variable models
- it is called the *Expectation-Maximization* (EM) algorithm for reasons seen later

- the finite mixture model motivates the EM algorithm
- we saw that it is easy to deal with distributions

$$p(L|\theta, y) \text{ (discrete distribution)}$$

$$p(\theta|L, y) \propto p(y|L, \theta)p(\theta|L)$$
(Normal)

- but it is difficult to deal with $p(\theta|y)$ directly: this is a product of a possibly multimodal or otherwise difficult likelihood and a prior

- we want to find θ that maximizes $p(\theta|y)$
- Jensen's inequality $\log \mathrm{E}(x) \geq \mathrm{E}(\log x)$ gives

$$\begin{aligned}
\log p(\theta|y) &= \log \int p(\theta, L|y) dL \\
&= \log \int q(L) \frac{p(\theta, L|y)}{q(L)} dL \\
&\geq \int q(L) \log \frac{p(\theta, L|y)}{q(L)} dL = -D(q\|p) \\
&= F(q, \theta)
\end{aligned}$$

- $F(q, \theta)$ bounds $\log p(\theta|y)$ from below
- it can be written as

$$F(q, \theta) = E_q(\log p(\theta, L|y)) - E_q(\log q(L))$$

- the EM-algorithm can be written as follows:

E-step maximize $F(q_n, \theta_{n-1})$ by choosing a maximizing distribution q_n given θ_{n-1}

M-step maximize $F(q_n, \theta_n)$ by choosing a maximizing θ_n

iterate increase n and go to E-step

E-step

- maximize $F(q_n, \theta_{n-1}) = -D(q_n \| p)$
- due to properties of KL divergence,
  $q_n(L) = p(\theta_{n-1}, L|y)$ maximizes $F(q_n, \theta_{n-1})$
- this step corresponds to the computation of τ_{ij}:s
- $q(L)$ allows us to compute

$$\mathrm{E}_q(\log p(\theta, L|y)) = \int \log p(\theta, L|y) q(L) dL,$$

M-step

- find θ_n which maximizes
  $F(q_n, \theta_n) = \mathrm{E}_q(\log p) - \mathrm{E}_q(\log q_n)$
- $\mathrm{E}_q(\log q(L))$ remains constant, so maximize

$$\mathrm{E}_q(\log p(\theta, L|y)) = \int \log p(\theta, L|y) q(L) dL,$$

- this corresponds to setting $\mu_{j,new} = \sum_i \tau_{ij} y_i / \sum_i \tau_{ij}$

- maximizing F makes intuitive sense since it is a lower bound to $\log p(\theta|y)$
- two problems: maximization of F is an iteration, and F is only a lower bound
- in general, global maximum of $p(\theta|y)$ is not found but each iteration of EM is guaranteed not to decrease the posterior (exercise)

Example: EM for Normal Mixture Model

- earlier example illustrates the results below
- the E-step gives the distribution $q(L)$ as

$$\tau_{ij} = p(L_{ij} = 1|\theta', y_i) = \frac{\lambda_j N(y_i|\mu'_j, \Sigma'_j)}{\sum_k \lambda_k N(y_i|\mu'_k, \Sigma'_k)}$$

- it tells the posterior probability that data y_i comes from mixture component j, given *old values* of parameters θ' (prime refers to old values)

- M-step gives the update equations

$$\lambda_j = \frac{1}{N}\sum_i \tau_{ij} \quad \text{(mixture proportions)}$$

$$\mu_j = \frac{\sum_i y_i \tau_{ij}}{\sum_i \tau_{ij}} \quad \text{(mean values)}$$

$$\Sigma_j = \frac{\sum_i \tau_{ij}(y_i - \mu_j)(y_i - \mu_j)^T}{\sum_i \tau_{ij}} \quad \text{(covariance matrices)}$$

- note that the values μ_j used in updating Σ_j are the *new values*, i.e. those that were just updated
- in the following demo, three clusters of 2D data are generated
- a Normal mixture model is solved using the EM-algorithm
- demo: `emnorm.R`

Variational Approximation for Normal Mixtures

- the derivation of EM-algorithm using KL-divergence can be generalized
- minimize the function

$$F(\lambda_L, \lambda_\theta) = \int \int q(L)q(\theta) \log \frac{q(L)q(\theta)}{p(\theta, L|y)} dL d\theta$$

which is $D(q(L)q(\theta)||p(\theta, L|y))$

- substituting $q(\theta) = \delta(\theta - \theta_0)$ would give the function $-F(q, \theta_0)$ as in the EM-algorithm
- the variational approximation is more general, since it allows an arbitrary approximation distribution $q(\theta)$
- in the Normal mixture case, the approximating distributions can be obtained using free-form approximation

- the notation λ_L and λ_θ refer to parameters that define the approximating distributions $q(L)$ and $q(\theta)$
- for example, $q(\mu_j)$ is Normal so λ_θ contains the parameters of that Normal distribution
- the problem is to find the values of these parameters that minimize the KL-divergence

- the EM-like algorithm is now
  1. minimize F by choosing λ_L while keeping λ_θ fixed
  2. minimize F by choosing λ_θ while keeping λ_L fixed
  3. return to step 1 until convergence
- the full details of this example are quite complicated and are omitted here
- the details can be found in the paper
  A Variational Bayesian Framework for Graphical Models,
  H. Attias, NIPS-10, 2001, MIT Press.

- example: clustering data generated from a three-component Normal mixture
- the number of components in the model can be larger
- prior for the component means is such that the mean value $(0, 0)$ is most probable (circled in the demo)
- 'unused' components (those with very small precision) can be identified as those which converge to zero
- demo: `varmixt.R`

- compromise between representing the data well (many components) and avoiding overfitting (small number of components)
- variational approximation avoids overfitting automatically, since having too many components in the model makes the posterior very narrow at the mode
- there are numerous heuristics which attempt the same thing, such as MDL, AIC, and regularization

- the selection of components happens through making some components unused
- when some component is far from data, its effect to the posterior is mainly determined by the prior
- the prior is maximized by mean $(0, 0)$ and zero precision, so the "unused" components converge to these values

Lecture 11: 4.4.2007

Missing Data

Due to Easter Holiday, there are no exercises on 6.4 and no lecture on 11.4

Missing Data

- consider multivariate data $y_i \in \mathbb{R}^d$
- observe a few such vectors

$$y = (y_1, y_2, y_3, y_4, y_5) = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 1 & 0 & 7 \\ 3 & 7 & 2 & 5 & 8 \end{bmatrix}$$

- likelihood: $p(y|\theta) = \prod_i p(y_i|\theta)$

- but what if your data is

$$y = (y_1, y_2, y_3, y_4, y_5)$$

$$= \begin{bmatrix} 1 & ? & 3 & 4 & ? \\ 2 & ? & 1 & 0 & 7 \\ 1 \text{ or } 7 & 7 & \text{in } [2,5] & \text{less than } 9 & 8 \end{bmatrix}$$

- some components of y_i are *missing* (value is not a single number)
- this kind of data is not uncommon in real problems

- with missing data, the posterior $p(\theta|y)$ depends partly on *unobserved data*
- a heuristic solution: discard all vectors y_i that are not completely observed and proceed as usual
- but in the example, no vector is completely observed
- here, and often in practice, simply throwing away data is not optimal

- another heuristic: keep all the observed parts of y (call them y_{obs}), even if this means keeping only parts of vectors
- not generally optimal for two reasons:
  - the discarded components may carry some information directly, such as "less than 9"
  - the fact that a value is missing may give information about the nonobserved value, and therefore information about θ

- *missing values can contain lots of information*
- in predicting the absorbtion of an orally administered drug, bloodsamples are taken and the drug concentration is analyzed
- i.e. data consists of nonnegative values measured at certain times $t_1, t_2, \ldots, t_m$

- for technical reasons, drug concentration below some small value $a > 0$ cannot be measured
- these values become missing data
- but we do know that with high probability, these values are in the very small interval $[0, a]$
- discarding these values obviously loses information

- coin toss, probability of heads is θ
- observe some coin toss results and estimate θ
- for some reason, "heads" will not be observed with probability 0.5, and "tails" with probability 0.25
- just using observed data underestimates θ

- the probabilities define a *missing data mechanism*
- in this example, it is completely defined by data (observed and missing)
- y_{obs} alone does not define it here
- it would if the missingness probabilites were equal
- then using observed data would give correct results

- *general missing data mechanism*
- extend the coin tossing example so that "heads" goes missing with unknown probability ϕ
- now the missing data mechanism depends on data as well as ϕ
- ϕ belongs to the posterior, and data gives information about it: observing lots of "heads" suggests that ϕ is small

- in principle, one could solve the problem by constructing a model for data, θ, ϕ, and "missingness indicators"
- unknown quantities are missing values, θ, and ϕ
- this is complicated in practice: heuristics are often used, or one can make sure that the missing values can be ignored

Missing Data Heuristics

- there exist many heuristics for handling models with missing data
- some of them may be optimal in some cases, but generally they are not
- exercises illustrate some cases where these heuristics fail

- *list deletion*: observed vector y_i is discarded if one or more of the components are missing
- for scalar data, this corresponds to ignoring all missing data
- if the missing data mechanism is *ignorable* (defined later), then list deletion does not bias the results
- but information is lost when the data is multivariate since some observed components are discarded

- *imputation*: guess missing values
- *mean imputation* replaces missing values with the mean of the observed values
- *regression imputation* uses regression on observed data to guess the values
- even with reasonable imputed values, these methods underestimate posterior uncertainty

- above heuristics are generally non-Bayesian (imputation can be Bayesian if several values are simulated e.g. by MCMC)
- however, in some special cases they may correspond to the Bayesian procedure
- starting from the proper Bayesian method, we see some cases where missing data can be ignored
- at the same time we see the general way of handling missing data

Bayesian Treatment of Missing Data

- denote nonrigorously the complete data by $y = (y_{obs}, y_{mis})$ (**obs**erved, **mis**sing)
- the meaning of y_{obs} is that it represents the values of all observed data (y_{mis} correspondingly)
- define the complete data y as a matrix
- any of the elements can be missing (hence the difficulty of proper notation for y_{obs} and y_{mis})

- define a matrix I with the same dimensions as y to indicate missing components
- i.e. $I_{ij} = 0$ if y_{ij} is missing and $I_{ij} = 1$ if y_{ij} is observed
- I is known, since we know which parts of y are missing

- in general, we would like to compute $p(\theta|y)$
- since $p(\theta|y) = p(\theta|y_{obs}, y_{mis})$, this posterior is useless in practice since we don't know y_{mis}
- lets review what we have:
  - knowns: y_{obs} and missingness indicators I
  - unknowns: y_{mis}, θ, and ϕ (parameters for missingness)
  - we want to know $p(\theta|y_{obs}, I)$

- the full posterior is obtained using the product rule:

$$\begin{aligned} p(\theta, \phi, y_{mis} | y_{obs}, I) &= \frac{p(\theta, \phi, y_{mis}, y_{obs}, I)}{p(y_{obs}, I)} \\ &\propto p(\theta, \phi, y, I) \\ &= p(I, y | \theta, \phi) p(\theta, \phi) \end{aligned}$$

- in the following, we assume that the complete-data likelihood can be written as

$$p(I, y | \theta, \phi) = p(I | y, \phi) p(y | \theta)$$

- meaning:
  - $p(I|y,\theta,\phi) = p(I|y,\phi)$: missingness does not depend on θ if y, ϕ are known
  - $p(y|\theta,\phi) = p(y|\theta)$: given θ, data does not depend on ϕ
- marginalizing the full posterior over y_{mis} and ϕ gives

$$p(\theta|y_{obs}, I) \propto \int \int p(I|y,\phi)p(y|\theta)p(\theta,\phi)dy_{mis}d\phi$$

Missing Data Mechanisms

- sometimes the difficult marginalization can be simplified
- this depends on properties of the *missing data mechanism*

$$p(I|y, \phi)$$

- this is in general a function of all data (observed and missing), the indicators and possibly ϕ

Missing Completely at Random

- *missing completely at random (MCAR)* holds when

$$p(I|y,\phi) = p(I|\phi)$$

- the probability does not depend on the values of y_{obs} and y_{mis}
- example: coin toss where each toss is missing with a fixed probability

Missing at Random

- *missing at random (MAR)* holds when

$$p(I|y,\phi) = p(I|y_{obs},\phi)$$

- missingness does not depend on values y_{mis}
- generic example: $y_i = (a_i, b_i)$, always observe a_i but missingness of b_i depends on a_i only
- MAR is very useful, since with another assumption it simplifies handling of missing data

Observed at Random

- *observed at random (OAR)* holds when

$$p(I|y,\phi) = p(I|y_{mis},\phi)$$

- missingness does not depend on observed values y_{obs}
- this case is not very useful on its own

Neither MAR nor OAR

- missing data is *neither MAR nor OAR* when

$$p(I|y,\phi)$$

  cannot be simplified by removing values of y
- this is the general case, and thus requires the full marginalization derived above
- sometimes called non-ignorable but this can be confusing, as seen later
- in the drug absorption example, missingness depends on the value y compared with the threshold:

model is neither MAR nor OAR

- note that in the above definitions, the given condition must hold for *all pairs* y, I
- for example in MAR, $p(I|y, \phi)$ must be a function of only those values y_{ij} for which $I_{ij} = 1$
- this must hold for any possible pair y, I

- if $p(\theta|y_{obs}) = p(\theta|y_{obs}, I)$, then the missing data mechanism can be ignored
- this missing data property is called *ignorability*
- warning! even though *neither MAR nor OAR* is called non-ignorable, it does not mean that all other cases are ignorable!

- what can we do when missing data is ignorable?
- for ignorable missing data mechanism we get

$$p(\theta|y_{obs}) \propto p(y_{obs}|\theta)p(\theta)$$
$$= \int p(y|\theta)p(\theta)dy_{mis}$$

- so it is enough to integrate the missing data out from the complete data model
- without ignorability, one ends up integrating over $p(y|\theta, I)$ which is usually complicated

- ignorability can be guaranteed in certain special cases
- MAR (missing at random) means that

$$p(I|y,\phi) = p(I|y_{obs},\phi)$$

- i.e. the missing values do not affect the indicators
- MAR alone does not guarantee ignorability!

- the indicators can still depend on ϕ
- if missing data mechanism is MAR and condition $p(\phi|\theta) = p(\phi)$ (denote as $\phi \perp\!\!\!\perp \theta$) holds, then we obtain ignorability
- antiexample: data are measurements of an object weighing θ kilos. If the scale has problems in weighting heavy objects, then ϕ (failure probability) and θ (the weight) are not independent
- model can still be MAR, since given ϕ, the missingness values I can be independent of y_{mis}

- MAR, $\phi \perp\!\!\!\perp \theta \implies$ ignorability:

$$
\begin{aligned}
p(\theta|y_{obs}, I) &\propto \int\int p(I|y,\phi)p(y|\theta)p(\theta,\phi)dy_{mis}d\phi \\
&\overset{MAR}{=} \int\int p(I|y_{obs},\phi)p(y|\theta)p(\phi|\theta)p(\theta)dy_{mis}d\phi \\
&\overset{\phi \perp\!\!\!\perp \theta}{=} \int\int p(I|y_{obs},\phi)p(y|\theta)dy_{mis}p(\theta)p(\phi)d\phi \\
&= p(\theta)\int p(y|\theta)dy_{mis}\int p(I|y_{obs},\phi)p(\phi)d\phi \\
&\propto p(\theta)p(y_{obs}|\theta) \\
&\overset{BT}{\propto} p(\theta|y_{obs})
\end{aligned}
$$

- as always, closed-form results are often difficult
- there are two main approaches (assume ignorability):
  - approximate $p(\theta|y_{obs})$ e.g. by the EM-algorithm
  - simulate $p(\theta, y_{mis}|y_{obs})$
- obtaining simulated values of y_{mis} is called *multiple imputation*
- these imputed values can be used later to apply methods that cannot handle missing data

EM for Missing Data

- missing data y_{mis} can be thought of as latent variables
- the EM-algorithm for an ignorable case is easy

E-step: $q(y_{mis}) = p(y_{mis}, \theta | y_{obs})$

M-step: maximize

$$\int \log p(y_{mis}, \theta | y_{obs}) q(y_{mis}) dy_{mis}$$

Data Augmentation

- assume that missingness is ignorable and $p(\theta|y_{obs})$ is difficult to approximate
- $p(y_{mis}|y_{obs},\theta)$ is generally simpler
- also the posterior $p(\theta|y_{mis},y_{obs}) = p(\theta|y)$ is often simpler than $p(\theta|y_{obs})$
- two unknowns θ, y_{mis}, and two "easy" full conditional posteriors

- *data augmentation*:
  1. set $i = 1$, choose initial approximation $p_1(\theta|y_{obs})$
  2. simulate $\theta_1, \ldots, \theta_m$ from $p_i(\theta|y_{obs})$ and then simulate $y_{mis}^j,\ j = 1, \ldots, m$ from
  $$p(y_{mis}|\theta^j, y_{obs})$$
  3. approximate $p_{i+1}(\theta|y_{obs})$ as a mixture
  $$p_{i+1}(\theta|y_{obs}) \propto \sum_{j=1}^{m} p(\theta|y_{mis}^j, y_{obs})$$
  4. set $i := i + 1$ and go to step 2

- DA uses the same full conditional posteriors as the Gibbs Sampler
- if you simulate just one y_{mis} ($m = 1$), then DA is Gibbs Sampler
- the DA algorithm can be shown to converge in a certain sense
- as in Gibbs Sampler, the important part is to be able to simulate the full conditional posteriors

- $p(y_{mis}|y_{obs}, \theta)$ is often easy to simulate
- but $p(\theta|y)$ is as difficult as without missing data
- heuristically, one may simulate the distributions using any simulation methods, for example MCMC
- one should be careful to try to obtain an independent set of simulated values

- assume you have a DA/Gibbs Sampler for nonmissing data
- in the ignorable case,just add y_{mis} to the model
- the steps to simulate θ are unchanged (ϕ and I cause complications in non-ignorable cases)
- reason: the unknown y_{mis} is assumed to be known in these steps

- y_{mis} is simulated from $p(y_{mis}|y_{obs}, \theta)$
- this is often easy due to known θ
- a possible complication is that y_{mis} is not always independent of y_{obs}, given θ
- example: $p(y|\theta) = N(y|\mu, \Sigma)$, $y \in \mathbb{R}^2$
- if $y = [y_{obs}, y_{mis}]'$ (observe only the first component), then $p(y_{mis}|y_{obs}, \theta)$ is not $p(y_{mis}|\theta)$

- in the exercises, the Gibbs sampler is used to simulate a Normal model with missing data
- consider Normal data in $\mathbb{R}^2$ where some of the second components are missing
- compare list deletion, mean imputation, and Gibbs sampling in estimating a Normal model
- demo: `gibbsmissing2.R`

Lecture 12: 18.4.2007

Gaussian Processes

Gaussian Processes

- so far the emphasis has been on solving a learning problem with a given model
- the probabilistic model links the unknown quantities θ and data
- the model should be constructed using the information available about the problem
- without a model (implicit or explicit), there is no learning

- practical difficulties:
  - closed-form results often difficult or impossible
  - approximations computationally heavy (simulation)
  - sometimes difficult to specify the model if θ has no clear interpretation

- above difficulties may be unavoidable if accurate results are needed
- *local learning* was discussed early in the course
- it was demonstrated that certain "all-purpose" learning methods essentially learn locally
- this means that prediction will depend on nearby observed data
- the intuitive idea: if inputs x_i, x_j are close, then outputs y_i, y_j should have similar values

- this amounts to an implicit model which favours "smooth" solutions
- technically this works no better than any other method (NFL theorems)
- but what if the information available is "smooth solutions are more probable"?
- a model is needed which places a high prior probability on smooth regression functions

- consider a parametric regression model, such as $y = f(x|\theta) + n$
- the prior is over parameter θ, not the function $f(x|\theta)$
- given θ, is $f(x|\theta)$ regular or rapidly varying?
- for complicated models this may not be easy to determine

- solving a given learning problem, prior for θ comes from problem-specific information
- in general-purpose methods, parameters often have no direct meaning
- defining the prior over the functions $f(x|\theta)$ may be easier
- no fundamental difference: with "equivalent" priors, all predictions will be identical

- example: likelihoods $g(x|\gamma)$ and $f(x|\theta)$
- assumption: for any γ there is θ so that $g(x|\gamma) = f(x|\theta)$ for all x (and vice versa)
- also assume that $p(\gamma)$ and $p(\theta)$ are such that the same functions get the same prior probability
- then *all predictions are the same* whether you use $f(x|\theta)$ or $g(x|\gamma)$

- *Gaussian Processes* allow solving a regression problem in closed form without requiring a prior on abstract parameters
- can be developed as a parametric model, or using a prior for a stochastic process generating the data
- parametric model corresponds to earlier results, but the equivalent process approach is simpler
- GP classification does not yield closed-form results, but requires approximations

Bayesian Regression

- Gaussian regression: likelihood is

$$p(y|H,\theta) = N(y|H\theta,\sigma^2 I)$$

- H is a known matrix, and is a function of inputs x_i
- example:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}$$

- this means $y_i = \theta_1 + \theta_2 x_i + n_i,\ n_i \sim N(0,\sigma^2)$

- one can also define $[H]_{ij} = h_j(x_i)$, where h_j is some known, nonlinear basis function
- note that this is similar to the Support Vector Machine
- the linear regression on the "features" $h(x_i)$ becomes nonlinear regression on the inputs x_i
- the solution is equally easy in both cases, except for computational complexity

- we are interested in predicting the output $\tilde{y}$ for a new input $\tilde{x}$:

$$p(\theta|y) \propto p(y|\theta)p(\theta)$$
$$p(\tilde{y}|y) = \int p(\tilde{y}|\theta)p(\theta|y)d\theta$$

- the prediction does not depend on θ, since it is marginalized out
- however, we need the prior for θ before it can be marginalized

- lets choose a Normal prior for θ:

$$p(\theta) = N(\theta|0, \sigma_0^2 I)$$

- the outputs of the regression function form a vector

$$Y = H\theta = \sum_{j=1}^{m} \theta_j h_j$$

- Y is a random vector and its distribution is induced by $p(\theta)$
- Y is a zero-mean Normal vector since it is a linear combination of Normally distributed zero-mean variables

- the covariance matrix of Y is

$$\mathrm{E}(YY') = H\,\mathrm{E}(\theta\theta')H' = \sigma_0^2 HH'$$

- this gives the induced prior on Y:

$$p(Y) = N(Y|0, \sigma_0^2 HH') = N(Y|0, Q)$$

- $p(\theta)$ is the parametric prior, and $p(Y)$ is a prior *on the data itself*

- the random vector Y contains the noiseless, linear outputs of the model
- we ultimately need the observable, noisy outputs y
- the predictive distribution of y is

  $p(y) = N(y|0, Q + \sigma^2 I) = N(y|0, C)$ (noise is zero-mean and inc
- the prior $p(y)$ is a function of known quantities H, σ_0^2, σ^2

- the covariance $C = \sigma_0^2 HH' + \sigma^2 I$ defines $p(y)$
- each element of the covariance matrix is

$$[C]_{ij} = \sigma_0^2 \sum_k h_k(x_i) h_k(x_j) + \sigma^2 \delta_{ij}$$

- the "feature vector" of input x_i is

$$h(x_i) = [h_1(x_i), h_2(x_i), \ldots, h_m(x_i)]'$$

- then $[C]_{ij}$ is essentially an inner product between $h(x_i)$ and $h(x_j)$ (compare with SVM again)

- summary:
  - linear regression becomes nonlinear with nonlinear feature space
  - parameters θ can be integrated out, leaving a Normal distribution over outputs y
  - covariance matrix contains inner products of feature vectors

- can we do regression without parameters?
- define a prior on *observable outputs* y as
  $p(y) = N(y|0, C)$
- define covariances as functions of inputs:
  $[C]_{ij} = D(x_i, x_j)$
- it can be computed if D is known, since all inputs x_i are known

- the *covariance function* $D(x, x')$ must be chosen (compare with the kernel function in SVM)
- all covariances $\mathrm{E}(y_i y_j) = D(x_i, x_j)$ can then be computed
- only restriction for $D(x, x')$ is that the matrix C computed using it must be positive semidefinite
- the *Gaussian Process prior* $N(y|0, C)$ is then completely defined without a parametric model

Gaussian Process Predictions

- lets solve the regression problem using

$$p(y) = N(y|0, C)$$

- predict $\tilde{y}$ at an input $\tilde{x}$, given y and x
- the predictive distribution is obtained by

$$\begin{aligned} p(\tilde{y}, y) &= p(\tilde{y}|y)p(y) \\ \Longrightarrow \quad p(\tilde{y}|y) &= p(y, \tilde{y})/p(y) \end{aligned}$$

- the joint distribution $p(y, \tilde{y})$ is Normal and not conditional to anything
- it is given by the prior $N(0, \tilde{C})$
- the matrix $\tilde{C}$ is computed at all pairs from $\{x_1, \ldots, x_n, \tilde{x}\}$
- the predictive distribution is easily seen to be Normal since it is obtained by considering $p(y, \tilde{y})$ with y known

- the covariance matrix $\tilde{C}$ of $(y, \tilde{y})$ can be partitioned as

$$\tilde{C} = \begin{bmatrix} C & k \\ k' & c \end{bmatrix}$$

$$\begin{aligned} C &= \text{ covariance matrix of } y \\ k &= [D(x_1, \tilde{x}), \ldots, D(x_n, \tilde{x})]', \text{ a vector} \\ c &= D(\tilde{x}, \tilde{x}), \text{ a scalar} \end{aligned}$$

- this gives a predictive distribution

$$p(\tilde{y}|y) \propto \exp[-\frac{1}{2}(y, \tilde{y})'\tilde{C}^{-1}(y, \tilde{y})]$$

- a partitioned matrix $\tilde{C}$ has a partitioned inverse:

$$\tilde{C}^{-1} = \begin{bmatrix} M & m \\ m' & \mu \end{bmatrix}$$

where

$$\mu = (c - k'C^{-1}k)^{-1}$$

$$m = -\mu C^{-1}k$$

$$M = C^{-1} + \frac{mm'}{\mu}$$

- putting these into the equation for the predictive distribution, we find

$$p(\tilde{y}|y) \propto \exp\left[-\frac{(\tilde{y} - k'C^{-1}y)^2}{2(c - k'C^{-1}k)}\right]$$

- can we compute this?
- yes, since c, C, and k depend only on points x and $\tilde{x}$ which we know
- note that the matrix C^{-1} depends only on training data and not $\tilde{x}$

- summary of regression using a Gaussian Process:
  1. choose a covariance function $D(x_i, x_j)$, use it to compute C and then invert C
  2. choose $\tilde{x}$ as the point at which you want to predict $\tilde{y}$: compute k and c using the covariance function
  3. compute the mean of predictive distribution: $k'C^{-1}y$
  4. compute the variance of the predictive distribution: $c - k'C^{-1}k$

- the results depend on the covariance function, which directly speficies properties of the actual data
- one can easily implement various kinds of smoothness assumptions through it
- for example, a high positive covariance when $\|x - x'\|$ is somewhat large favours very smooth solutions

- an exponential covariance can be derived from a neural network with an arbitrary number of neurons (exercise)
- the covariance works even when the number of neurons approaches infinity
- lets examine the effects of covariance functions using

$$D(x, x') = a \exp(-\frac{1}{2}\frac{(x - x')^2}{b^2}) + c\delta(x, x')$$

- demo: `gpreg.R`

- the simple solution presented above relies on a known $D(x, x')$
- this may be unrealistic: some properties of the covariance may be unknown
- the noise variance is the most obvious example
- with fixed $D(x, x')$, smoothing of data is constant wrt number of training points
- Bayesian solution: add parameters θ to the covariance function

- predictions are averaged as

$$p(\tilde{y}|y) = \int p(\tilde{y}|y,\theta)p(\theta|y)d\theta$$

- the first term in the integrand is the Normal predictive distribution for a given θ
- but its dependence on θ is quite complicated: θ appears in the covariance typically in a nonlinear manner

- we can approximate by computing the MAP estimate $\hat{\theta}$, and then using $p(\tilde{y}|y, \hat{\theta})$
- to do this, maximize

$$\log p(\theta|y) = \log p(y|\theta) + \log p(\theta) + D$$

- the log-likelihood part and its derivative are

$$\log p(y|\theta) = -\frac{1}{2} \log |C| - \frac{1}{2} y^T C^{-1} y + D_2$$

$$(\log p(y|\theta))' = -\frac{1}{2} \operatorname{tr}(C^{-1} C') + \frac{1}{2} y^T C^{-1} C' C^{-1} y$$

- the log-prior must also be derivated: then we obtain a gradient for $\log p(\theta|y)$ wrt θ, and we can use gradient ascent to estimate the most probable θ
- example: use the same covariance as before, with parameters a, b and c, and use gradient ascent to optimize covariance parameters
- demo: `gphier.R`

Gaussian Processes for Classification

- technically classification is regression, but a realistic model for classification is usually different
- example: class labels (outputs) in $\{-1, 1\}$
- outputs are not assumed to have noise (labels are either -1 or 1, not e.g. 0.2)
- even if outputs are modeled as noisy class labels, the training outputs are noise-free

- goal is to classify a new input $\tilde{x}$ given D (containing inputs $x_1, \ldots, x_n$ and outputs $y_1, \ldots, y_n$)
- GP regression is not directly applicable because it requires a Normal distribution for the outputs
- use *latent variables* $u_1, \ldots, u_n$ which are obtained by GP regression on inputs x_i
- the class label distribution $p(y = 1|u)$ must be defined

- examples:
  - 'hard' classifier: $p(y = 1|u) = 1$ when $u > 0$
  - 'soft' classifier: $p(y = 1|u) = \Phi(u/\alpha)$, where $\Phi(x) = \int_{-\infty}^{x} N(y|0, 1)dy$ (cdf of a Normal distribution)
- the regression $x \mapsto u$ does most of the work
- the nonlinear $p(y = 1|u)$ simply scales the outputs to probabilities

- denote by $\tilde{u}$ the latent variable corresponding to $\tilde{x}$
- also denote by $u = (u_1, \ldots, u_n)$ the latent variables corresponding to the training inputs
- the predictive distribution (write $U = (u, \tilde{u})$) is

$$\begin{aligned} p(\tilde{y} = 1|\tilde{x}, D) &= \int p(\tilde{y} = 1, U|\tilde{x}, D) dU \\ &= \int p(\tilde{y} = 1|\tilde{u}) p(U|\tilde{x}, D) dU \\ &= \int p(\tilde{y} = 1|\tilde{u}) p(\tilde{u}|u, \tilde{x}, D) p(u|\tilde{x}, D) dU \end{aligned}$$

- lets examine the factors in the integral:
  - $p(\tilde{y} = 1|\tilde{u})$ is a known function of $\tilde{u}$ (chosen earlier)
  - $p(\tilde{u}|u, \tilde{x}, D)$ is GP regression (predict output $\tilde{u}$ using u)
  - $p(u|\tilde{x}, D)$ is difficult, since D has only the outputs $y_i \in \{-1, 1\}$
- for realistic classifier models, no closed-form solution exists

- there are several methods to solving GP classification approximately:
  - simulation (Radford Neal)
  - mean-field algorithms (Opper and Winther)
  - variational approximation (Gibbs and MacKay)
- these methods are not discussed here due to lengthy details: see references on the course webpage

- to demonstrate the feasibility of solving classification problems using GP's, the naive mean-field algorithm of Opper and Winther is applied to certain datasets
- the mean-field approximation assumes that $\tilde{u}$ is Normally distributed, given the training data
- the algorithm is an iterative method for finding coefficients α_i that appear as linear weights in the solution (similar to SVM)
- demo: `opper2.R`

Lecture 13: 25.4.2007

Making Decisions

Decision Theory

- so far we have concentrated on computing a posterior over the unknown quantity θ
- the distribution correctly describes what is known about θ, as long as we believe our model
- but the distribution must be put to use: there is no reason to compute it otherwise
- a general way of quantifying the use of posterior is decision making

- implicit decisions are made even when computing the posterior and/or specifying the model
- otherwise, one should e.g. use infinitely many input variables and keep simulating the posterior indefinitely
- decisions: simplify model, stop simulation
- reasons: too many input variables cost, computation costs

- example: crossing a river
- you estimate that an old bridge fails with probability 0.01 if you try to cross it
- using a new bridge is possible, but takes more time
- what information do you need to make a decision?

- knowing the failure probability is not enough
- if failing bridge means certain death, you probably decide to use the new bridge
- but if failure means falling a meter or two into water, your decision may be different
- probabilities are the same in both cases

- in general, we need to express the "value" of an outcome following a decision
- in the bridge example, we need

	fail	not fail
old	u_1	u_2
new	u_3	u_4

- lets denote this as a function $U(a, \theta)$ where $a \in \{\text{old,new}\}$ and $\theta \in \{\text{fail, not fail}\}$

- the *choice* $a \in \{a_1, \ldots, a_k\}$ denotes the decision
- θ denotes the uncertain outcome
- if U has monetary value, then it is called *payoff*
- later, we use a nonlinear function of money called *utility*

Decision Criteria

- many heuristic criteria have been proposed
- easy to demonstate nonoptimality by counterexamples
- *nonstochastic criteria* ignore the posterior
- *stochastic criteria* use the information in the posterior

- examples of nonstochastic criteria:
- *maximin:*

$$\text{argmax}_a \min_\theta U(a, \theta)$$

- *maximax:*

$$\text{argmax}_a \max_\theta U(a, \theta)$$

- *minimax regret:*

$$\text{argmin}_a \left[\max_{a_i \neq a} \max_\theta \left(U(a_i, \theta) - U(a, \theta) \right) \right]$$

- stochastic criteria use the probability distribution over the outcomes
- some examples:
- *modal outcome:* choose the highest payoff of the most probable outcome
- *expected value/payoff:* choose the highest expected payoff

- including the posterior of the outcome should improve the decisions
- but there are several different stochastic criteria
- only one of them can be correct
- counterexamples suggest some problems with modal outcome and expected payoff

- consider following decisions, outcomes, and payoffs:

	θ_1	θ_2	θ_3
$p(\theta \mid y)$	0.25	0.3	0.45
a_1	500	500	200
a_2	100	100	600

- *modal outcome*: mode is θ_3, and the payoff-maximizing decision is a_2

- modal outcome is essentially point estimation
- therefore it suffers from most point estimation problems
- for example, consider combining θ_1 and θ_2
- now the combined outcome is the mode and the decision a_1 has the highest payoff

- expected payoff is a special case of the optimal decision rule, but is not generally optimal
- consider the decision between receiving 1000 €, or flipping a coin and either receiving 1.1 M€ or having to pay 1 M€
- the first choice has a smaller expected value
- but you would probably choose it anyway

Utility

- *utility* fixes the expected value criterion so that it becomes optimal
- given a few axioms, there exists a numerical measure called utility:
  - the optimal decision is one that maximizes *expected utility*
  - utility is *subjective*: different decision-makers can have different utilities
  - generally a nonlinear function $U(C)$ of monetary payoff C

- $U(C)$ can be found using *certainty equivalents*
- example: cointoss with payoffs 0 and 1 EUR
- certainty equivalent is the maximum amount C you are willing to pay for the bet
- then utility $U(C) = 0.5U(0) + 0.5U(1)$
- by a series of such bets, one can approximate the function $U(C)$

- an optimal decision out of a finite set of choices $a_1, \ldots, a_k$ is the one that maximizes expected utility
- denote the utility of decision a_i given the outcome θ by $U(a_i, \theta)$
- the expected utility for choice a_i is

$$\mathrm{E}(U(a_i, \theta)) = \int U(a_i, \theta) p(\theta|y) d\theta$$

- in practice, one may want to define the monetary payoff $C(a_i, \theta)$ and then use $U(C(a_i, \theta))$

- important points:
  - optimal decisions require *subjective probabilities* and *subjective utility*
  - given certain axioms, it is optimal to maximize expected utility
  - approximate utility obtained by certainty equivalents

- bridge example, failing probability is 0.01:

	fail	not fail	EU
$p(\theta\|y)$	0.01	0.99	
old	-10	10	9.8
new	9	9	9

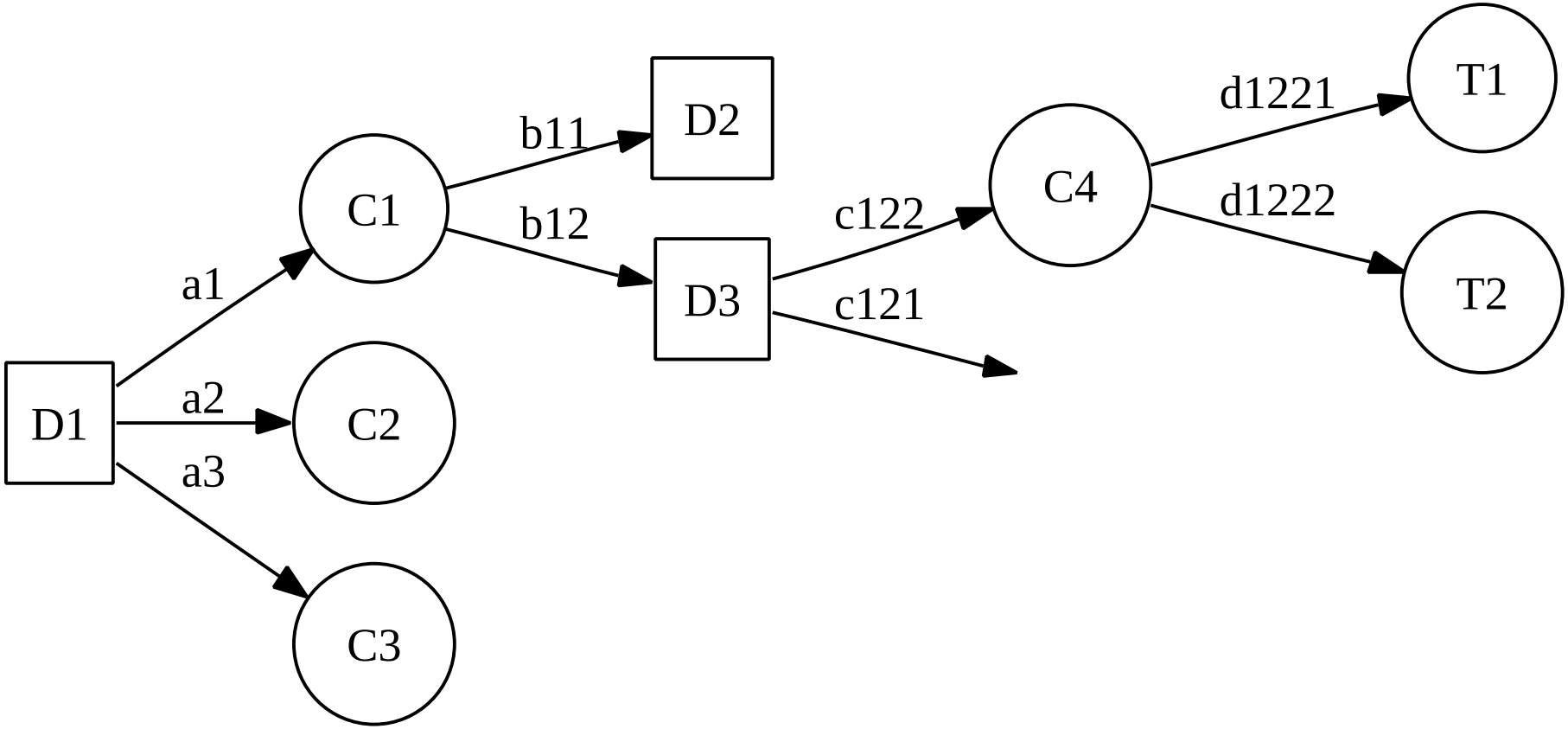
- what if failing is less pleasant?

	fail	not fail	EU
$p(\theta\|y)$	0.01	0.99	
old	-100	10	8.9
new	9	9	9

- maximizing expected utility is optimal in theory
- but decision theory can be difficult to apply
- *sequential decisions* (decisions depend on earlier decisions) are computationally very heavy
- a corresponding *decision tree* explodes in size as a function of number of decisions (e.g. playing chess)

- small enough decision trees can be solved
- a decision tree alternates between *decision and chance nodes*
- decisions are made at decision nodes, leading to chance nodes
- chance node has random outcomes leading to another decision node

- example: first choose from a_1, a_2, a_3
- choice a_i is followed by an outcome b_{i1} or b_{i2}
- outcome b_{ij} follows a choice of c_{ij1} or c_{ij2}
- choice c_{ijk} leads to *terminal outcomes* d_{ijk1}, d_{ijk2}
- assume that the utilities of terminal outcomes are known


D1
a1
a2
a3
C1
C2
C3
b11
b12
D2
D3
c122
c121
C4
d1221
d1222
T1
T2

- how to make the decision $D1$? We don't know the utilities of $C1, C2, C3$
- the solution requires *dynamic programming*
- start by computing the expected utility on lowest chance nodes (e.g. $C4$)
- use terminal outcomes to do this

- using expected utilities such as $\mathrm{E}(U(C4))$, we can make optimal decisions at nodes $(D2, D3)$
- replace $D2, D3$ by the maximum expected utility of the lowest chance nodes
- continue by computing the EU of $C1, C2, C3$
- this allows us to make the decision $D1$ optimally

Model Selection

- not needed in principle, since the only correct model $p(\theta, D)$ is defined by the subjective prior uncertainty
- one cannot have several conflicting prior uncertainties, so a unique model is obtained in theory
- but there are various reasons why selecting a model from a set of candidates may be useful in practice
  - if θ is point estimated, then risk of overfitting may be reduced by model selection
  - a simple model is required for practical reasons (computational and data collection costs)

- not using correct $p(\theta, D)$ is a compromise between
  - cost of obtaining the wrong posterior uncertainty and predictive distribution
  - benefits such as less computation and smaller data collection costs
- probabilities cannot tell you how much computation costs, or how much having the wrong posterior costs
- decision theory solves the problem in principle

- most model selection heuristics use only probabilities
- examples: MDL, MML, AIC, Evidence framework
- they generally disagree on the same data and model!
- most can be thought as giving a "penalty" to complex models

- model selection should in principle be treated as a decision
- therefore one should pick the model which maximizes expected utility
- this is easier said than done, but in some applications costs are important enough to warrant this approach
- the key element is the utility corresponding to each candidate model

- the utility depends on the problem, but may include for example
  - data collection costs: simpler models might use only part of the data, so the unused data does not have to be collected at all
  - computational costs: solving a complicated model is generally costly
  - model accuracy: how well the model performs in making predictions

- a full-flegded decision approach is rarely done explicitly
- but most of the above costs are implicitly considered in practice
- nobody uses a model that:

  . . . takes forever to solve
  . . . requires data that is much too expensive
  . . . could be made much more accurate with little extra cost

- example: assessing quality of healthcare (Fouskakis and Draper, 2005)
- hospitals are modeled as processes which map input variables (results of clinical tests) to an outcome (death within 30 days)
- collecting variables is expensive (36 variables initially considered)
- utility-based model selection was used to select a subset of input variables

- this requires choosing the utility of each subset
- it includes a negative term, which can be obtained from actual costs of performing the necessary clinical tests
- the utility of the model accuracy was obtained by testing the performance, and eliciting a monetary payoff from health experts for the different outcomes

Posterior Approximation

- also a decision (which wrong posterior to compute?)
- often the decision to use the correct posterior has a very small utility (e.g. high computational cost)
- approximations are a compromise between low computational costs and wrong results
- if the cost of wrong results can be high, utilities should be considered in posterior approximation

- example: point estimation, decide which θ_0 approximates the posterior $p(\theta|y)$
- the expected utility is

$$\mathrm{E}(U(\theta_0, \theta)) = \int U(\theta_0, \theta) p(\theta|y) d\theta$$

- certain utilities lead to familiar point estimates

- for example, $U(\theta_0, \theta) = -(\theta_0 - \theta)^2$ leads to least-squares estimate
- $U = \delta(\theta_0 - \theta)$ leads to the MAP estimate, as before
- in this sense choosing a point estimate means making a decision
- note that none of the point estimates are generally better than others: they arise as a function of $U(\theta_0, \theta)$ and the posterior

- in general, choose $q(\theta)$ to approximate $p(\theta|y)$
- a utility $U(q(\theta), \theta)$ is called
  - *proper* if $\mathrm{E}(U(q, \theta))$ is maximized by $q = p(\theta|y)$
  - *local* if $U(q(\theta), \theta_i) = u_i(q(\theta_i))$ (utility at θ_i depends only on $q(\theta_i)$)
- Theorem (see Bernardo's book for details): a continuosly differentiable, local, and proper utility (for distributions) is

$$U(q, \theta) = A \log q + B(\theta)$$

- what is lost in utility by choosing q instead of p?
- if the utility is local, proper, and smooth, then we lose

$$\begin{aligned}
&\mathrm{E}[U(p,\theta) - U(q,\theta)] = \\
&\quad = \mathrm{E}[A\log p + B(\theta) - A\log q - B(\theta)] = \\
&\quad = \int [A(\log p - \log q)]\, p(\theta|y)d\theta = \\
&\quad = A\int p\log\frac{p}{q}d\theta = AD(p\|q)
\end{aligned}$$

- this is Kullback-Leibler divergence, up to multiplication by A

- example: true/false statements in T-61.5040 exam
- a set of statements is given, and your answer to each is a probability $q(\text{TRUE})$
- your *subjective probability* $p(\text{TRUE})$ can be different!
- you will get $1 - 4(1-q)^2$ points if the statement is true, and $1 - 4q^2$ points if it is false

- some properties:
  - to maximize expected number of points, choose $q = p$
  - answering $q = 0.5$ gives zero points (as does not answering at all)
  - answering $q \in \{0, 1\}$ gives 1 or -3 points
  - guessing $q \in \{0, 1\}$ gives on average -1 points