Introduction to Machine Learning

Lecturer: Bert Kappen
Radboud University Nijmegen, Netherlands

September 11, 2018

Book:
Pattern Recognition and Machine Learning
C.M. Bishop
http://research.microsoft.com/~cmbishop/PRML/

Figures from http://research.microsoft.com/~cmbishop/PRML/webfigs.htm
Course setup

- 3 EC course. Weekly tutorials
- Textbook "Pattern Recognition and Machine Learning" (C.M Bishop, 2006)
- All other course materials (slides, exercises) via http://www.snn.ru.nl/~bertk/inl_ml/
- Written exam (open book, no notes/slides/laptop)

Grading

- 2/3 final exam ($\geq 5.0$) + 1/3 avg. assignments
Course contents

Chapter 1 Introduction

• Probability theory
• Model selection
• Curse of dimensionality
• Decision theory
• Information theory

Chapter 2: Probability distributions

Chapter 3: Linear models for regression

Chapter 4: Linear models for classification

Chapter 5: Neural networks
Chapter 1: Introduction

Introduction ML

- General introduction
- Polynomial curve fitting, regression, overfitting, regularization
- Probability theory, decision theory
- Information theory
Recognition of digits

Image is array of pixels $x_i$, each between 0 and 1.
$x = (x_1, \ldots, x_d)$, $d$ is the total number of pixels. ($d = 28 \times 28$).

- Goal = input: pixels $\rightarrow$ output: correct category 0, $\ldots$, 9

- wide variability

- hand-made rules infeasible
Machine Learning

- Training set: large set of (pixel array, category) pairs
  - input data $x$, target data $t(x)$
  - category = class

- Machine learning algorithm
  - training, learning

- Result: function $y(x)$
  - Fitted to target data
  - New input $x \rightarrow$ output $y(x)$
  - Hopefully: $y(x) = t(x)$
  - Generalization on new examples (test set)
Preprocessing

- transformation of inputs \( x \) on training set + on test set

- by hand, rule based
  - scaling, centering
  - aspect ratio, curvature

- machine learning, unsupervised learning

- feature extraction
  - dimension reduction
  - reduces variability within class (noise reduction) \( \rightarrow \) easier to learn
  - reduces variability between classes (information loss) \( \rightarrow \) more difficult to learn
  - trade-off
Types of machine learning tasks

- Supervised learning
  - Classification: targets are classes
  - Regression: target is continuous

- Unsupervised learning (no targets)
  - clustering (similarity between input data)
  - density estimation (distribution of input data)
  - dimension reduction (preprocessing, visualisation)

- Reinforcement learning
  - actions leading to maximal reward
1.1. Polynomial curve fitting

- Regression problem
- Given training set of $N = 10$ data points
  - generated by $t_n = \sin(2\pi x) + \text{noise}$
- Goal: predict value $t$ for new $x$ (without knowing the curve)
We will fit the data using $M$-th order polynomial

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

$y(x, \mathbf{w})$ is nonlinear in $x$, but linear in coefficients $\mathbf{w}$, "Linear model"

Training set: $(x_n, t_n), n = 1, \ldots, N$. Objective: find parameters $\mathbf{w}$, such that

$$y(x_n, \mathbf{w}) \approx t_n, \text{ for all } n$$

This is done by minimizing the Error function

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} (y(x_n, \mathbf{w}) - t_n)^2$$

$E(\mathbf{w}) \geq 0$ and $E(\mathbf{w}) = 0 \iff y(x_n, \mathbf{w}) = t_n$
Minimization of the error function

\[ E(w) = \frac{1}{2} \sum_{n=1}^{N} (y(x_n, w) - t_n)^2 \]

In minimum: gradient \( \nabla_w E = 0 \)

Note: \( y \) linear in \( w \) \( \Rightarrow \) \( E \) quadratic in \( w \)

\( \Rightarrow \nabla_w E \) is linear in \( w \)

\( \Rightarrow \nabla_w E = 0 \): coupled set of linear equations (exercise)
Model comparison, model selection

Back to polynomial curve fitting: how to choose $M$?

Which of these models is the best one?
Define root-mean-square error on training set and on test set \( \{(\tilde{x}_n, \tilde{t}_n)\}_{n=1}^{\tilde{N}} \), respectively:

\[
E_{RMS} = \sqrt{2E(w^*)/N}, \quad E_{RMS} = \sqrt{\frac{1}{\tilde{N}} \sum_{n=1}^{\tilde{N}} (y(\tilde{x}_n, w^*) - \tilde{t}_n)^2}
\]

Too simple (small \( M \)) \( \rightarrow \) poor fit
Too complex (large \( M \)) \( \rightarrow \) overfitting (fits the noise)
Note that series expansion of \( \sin \) contain terms of all orders.
<table>
<thead>
<tr>
<th></th>
<th>M=0</th>
<th>M=1</th>
<th>M=3</th>
<th>M=9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_0^*$</td>
<td>0.19</td>
<td>0.82</td>
<td>0.31</td>
<td>0.35</td>
</tr>
<tr>
<td>$w_1^*$</td>
<td>-1.27</td>
<td>8</td>
<td></td>
<td>232</td>
</tr>
<tr>
<td>$w_2^*$</td>
<td>-25</td>
<td></td>
<td>5321</td>
<td></td>
</tr>
<tr>
<td>$w_3^*$</td>
<td>-17</td>
<td></td>
<td>48568</td>
<td></td>
</tr>
<tr>
<td>$w_4^*$</td>
<td></td>
<td></td>
<td>-231639</td>
<td></td>
</tr>
<tr>
<td>$w_5^*$</td>
<td></td>
<td></td>
<td>640042</td>
<td></td>
</tr>
<tr>
<td>$w_6^*$</td>
<td></td>
<td></td>
<td>-10618000</td>
<td></td>
</tr>
<tr>
<td>$w_7^*$</td>
<td></td>
<td></td>
<td>10424000</td>
<td></td>
</tr>
<tr>
<td>$w_8^*$</td>
<td></td>
<td></td>
<td>-557683</td>
<td></td>
</tr>
<tr>
<td>$w_9^*$</td>
<td></td>
<td></td>
<td>-125201</td>
<td></td>
</tr>
</tbody>
</table>
Model comparison, model selection

Same model complexity $M = 9$ and different data set sizes

Same model complexity: more data $\Rightarrow$ less overfitting
With more data, more complex (i.e. more flexible) models can be used
Regularization

Change the cost function $E$ by adding regularization term $\Omega(w)$ to penalize complexity.

$$\tilde{E}(w) = E(w) + \lambda \Omega(w)$$

For example,

$$\tilde{E}(w) = \frac{1}{2} \sum_{n=1}^{N} (y(x_n, w) - t_n)^2 + \frac{\lambda}{2} ||w||^2$$

(here, $||w||^2 := \sum_{m=0}^{M} w_m^2$)

Weight decay = shrinkage = ridge regression

Penalty term independent of number of training data

- small data sets: penalty term relatively large
- large data sets: penalty term relatively small
<table>
<thead>
<tr>
<th>[ \ln \lambda = -\infty ]</th>
<th>[ \ln \lambda = -18 ]</th>
<th>[ \ln \lambda = 0 ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w_0 )</td>
<td>0.35</td>
<td>0.35</td>
</tr>
<tr>
<td>( w_1 )</td>
<td>232</td>
<td>4.74</td>
</tr>
<tr>
<td>( w_2 )</td>
<td>5321</td>
<td>-0.77</td>
</tr>
<tr>
<td>( w_3 )</td>
<td>48568</td>
<td>-31.97</td>
</tr>
<tr>
<td>( w_4 )</td>
<td>-231639</td>
<td>-3.89</td>
</tr>
<tr>
<td>( w_5 )</td>
<td>640042</td>
<td>55.28</td>
</tr>
<tr>
<td>( w_6 )</td>
<td>-10618000</td>
<td>41.32</td>
</tr>
<tr>
<td>( w_7 )</td>
<td>10424000</td>
<td>-45.95</td>
</tr>
<tr>
<td>( w_8 )</td>
<td>-557683</td>
<td>-91.53</td>
</tr>
<tr>
<td>( w_9 )</td>
<td>-125201</td>
<td>72.68</td>
</tr>
</tbody>
</table>

- **Training set** to optimize (typically many) parameters \( w \)
- **Validation set** to optimize (typically a few) hyperparameters (\( \lambda \) or \( M \))
**Probability theory**

- Consistent framework for quantification and manipulation of uncertainty  
  → Foundation for Bayesian machine learning

- random variable = stochastic variable

Example:
  - boxes ($B = \{r, b\}$) and fruit ($F = \{a, o\}$).

- Consider an experiment of (infinitely) many (mentally) repeated trials  
  (randomly pick a box, then randomly select an item of fruit from that box)  
  under the same macroscopic conditions  
  (number of red/blue boxes and apples/oranges balls in the boxes)  
  but each time with different microscopic details  
  (arrangements of boxes and fruits in boxes).

*Probability* of an event (e.g. selecting a orange) is fraction of times that event occurs in the experiment.

- Notation: $p(F = o) = 9/20$, etc (or $P(\ldots)$, $\mathbb{P}(\ldots)$, Prob$(\ldots)$, etc.)
$X$ can take the values $x_i, \ i = 1, \ldots, M$.

$Y$ can take the values $y_j, \ j = 1, \ldots, L$.

$N$: total number of trials ($N \to \infty$).

$n_{ij}$: number of trials with $X = x_i$ and $Y = y_j$

$c_i$: number of trials with $X = x_i$

$r_j$: number of trials with $Y = y_j$

probability of $X = x_i$ and $Y = y_j$:

$$p(X = x_i, Y = y_j) = \frac{n_{ij}}{N} = p(Y = y_j, X = x_i)$$

Marginal probability of $X = x_i$:

$$p(X = x_i) = \frac{c_i}{N} = \frac{\sum_j n_{ij}}{N} = \sum_j p(X = x_i, Y = y_j)$$

Conditional probability of $Y = y_j$ given $X = x_i$

$$p(Y = y_j | X = x_i) = \frac{n_{ij}}{c_i}$$
• Explicit, unambiguous notation: \( p(X = x_i) \)
• Short-hand notation: \( p(x_i) \)
• \( p(X) \): “distribution“ over the random variable \( X \)
• NB: \( \{x_i\} \) is assumed to be mutually exclusive and complete

The Rules of Probability

<table>
<thead>
<tr>
<th>Rule</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum rule</td>
<td>( p(X) = \sum_Y p(X, Y) )</td>
</tr>
<tr>
<td>Product rule</td>
<td>( p(X, Y) = p(Y</td>
</tr>
<tr>
<td>Positivity</td>
<td>( p(X) \geq 0 )</td>
</tr>
<tr>
<td>Normalization</td>
<td>( \sum_X p(X) = 1 )</td>
</tr>
</tbody>
</table>
\[ p(X, Y) = p(Y|X)p(X) = P(X|Y)p(Y) \Rightarrow \]

\[
p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)} \quad \left( = \frac{p(X|Y)p(Y)}{\sum_Y p(X|Y)p(Y)} \right)
\]

Bayes’ theorem = Bayes’ rule
\[ p(X) \]

\[ p(Y) \]

\[ p(X, Y) \]

\[ p(Y | Y = 1) \]

\[ p(X | Y = 1) \]
Fruits again

Model

\[ p(B = r) = \frac{4}{10} \]
\[ p(B = b) = \frac{6}{10} \]
\[ p(F = a | B = r) = \frac{1}{4} \]
\[ p(F = o | B = r) = \frac{3}{4} \]
\[ p(F = a | B = b) = \frac{3}{4} \]
\[ p(F = o | B = b) = \frac{1}{4} \]

Note that the (conditional) probabilities are normalized:

\[ p(B = r) + p(B = b) = 1 \]
\[ p(F = a | B = r) + p(F = o | B = r) = 1 \]
\[ p(F = a | B = b) + p(F = o | B = b) = 1 \]

\[ p(F = a), p(F = o)? \] Given \( F = o \) what is \( p(B = r, b | F = o) \)?
• Marginal probability

\[ p(F = a) = p(F = a | B = r)p(B = r) + p(F = a | B = b)p(B = b) \]
\[ = \frac{1}{4} \times \frac{4}{10} + \frac{3}{4} \times \frac{6}{10} = \frac{11}{20} \]

and from normalization,

\[ p(F = o) = 1 - p(F = a) = \frac{9}{20} \]

• Conditional probability (reversing probabilities):

\[ p(B = r | F = o) = \frac{p(F = o | B = r)p(B = r)}{p(F = o)} = \frac{3}{4} \times \frac{4}{10} \times \frac{20}{9} = \frac{2}{3} \]

• Terminology:

\( p(B) \): prior probability (before observing the fruit)
\( p(B | F) \): posterior probability (after observing \( F \))
(Conditionally) independent variables

- \( X \) and \( Y \) are called (marginally) independent if

\[
P(X, Y) = P(X)P(Y)
\]

This is equivalent to

\[
P(X|Y) = P(X)
\]

and also to

\[
P(Y|X) = P(Y)
\]

- \( X \) and \( Y \) are called conditionally independent given \( Z \) if

\[
P(X, Y|Z) = P(X|Z)P(Y|Z)
\]

This is equivalent to

\[
P(X|Y, Z) = P(X|Z)
\]

and also to

\[
P(Y|X, Z) = P(Y|Z)
\]
Consider two coins. Coin 1 has probability of head of \( \frac{1}{2} \), coin 2 has probability of head \( \frac{1}{4} \).

Given the sequence of coin tosses:

\[ \text{head, tail, tail} \]

What is the probability that I used coin 1?
\[ p(D|\text{coin1}) = \frac{1 \times 1 \times 1}{2 \times 2 \times 2} = \frac{1}{8}, \quad p(D|\text{coin2}) = \frac{1 \times 3 \times 3}{4 \times 4 \times 4} = \frac{9}{64} \]

\[ p(\text{coin1}|D) = \frac{p(D|\text{coin1})p(\text{coin1})}{p(D)} \]
\[ p(D) = p(D|\text{coin1})p(\text{coin1}) + p(D|\text{coin2})p(\text{coin2}) \]

Assume equal priors \( p(\text{coin1}) = p(\text{coin2}) = \frac{1}{2} \). Then

\[ p(\text{coin1}|D) = \frac{p(D|\text{coin1})}{p(D|\text{coin1}) + p(D|\text{coin2})} = \frac{8}{8 + 9} = \frac{8}{17} \]
Probability densities

- to deal with continuous variables (rather than discrete var’s)

When $x$ takes values from a continuous domain, the probability of any value of $x$ is zero! Instead, we must talk of the probability that $x$ takes a value in a certain interval

$$\text{Prob}(x \in [a, b]) = \int_a^b p(x) \, dx$$

with $p(x)$ the probability density over $x$.

$$p(x) \geq 0$$

$$\int_{-\infty}^{\infty} p(x) \, dx = 1 \quad \text{(normalization)}$$

- NB: that $p(x)$ may be bigger than one.

Probability of $x$ falling in interval $(x, x + \delta x)$ is $p(x)\delta x$ for $\delta x \to 0$
Multivariate densities

- Several continuous variables, denoted by the $d$ dimensional vector $\mathbf{x} = (x_1, \ldots, x_d)$.
- Probability density $p(\mathbf{x})$: probability of $\mathbf{x}$ falling in an infinitesimal volume $\delta \mathbf{x}$ around $\mathbf{x}$ is given by $p(\mathbf{x}) \delta \mathbf{x}$.

$$\text{Prob}(x \in \mathcal{R}) = \int_{\mathcal{R}} p(\mathbf{x}) \, d\mathbf{x} = \int_{\mathcal{R}} p(x_1, \ldots, x_d) \, dx_1 \, dx_2 \ldots \, dx_d$$

and

$$p(\mathbf{x}) \geq 0$$

$$\int p(\mathbf{x}) \, d\mathbf{x} = 1$$

- Rules of probability apply to continuous variables as well,

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

$$p(x, y) = p(y|x)p(x)$$
Expectations and variances

Expectations

\[ \mathbb{E}[f] = \langle f \rangle = \sum_x p(x) f(x) \quad \text{discrete var's} \]

\[ \mathbb{E}[f] = \int_x p(x) f(x) \, dx \quad \text{continuous var's} \]

Variance:

\[ \text{var}[f] = \langle f(x)^2 \rangle - \langle f(x) \rangle^2 \]
\[ \text{var}[x] = \langle x^2 \rangle - \langle x \rangle^2 \]
Covariance

Covariance of two random variables

$$\text{cov}[x, y] = \langle xy \rangle - \langle x \rangle \langle y \rangle$$

Covariance matrix of the components of a vector variable

$$\text{cov}[\mathbf{x}] \equiv \text{cov}[\mathbf{x}, \mathbf{x}] = \langle \mathbf{x} \mathbf{x}^T \rangle - \langle \mathbf{x} \rangle \langle \mathbf{x}^T \rangle$$

with components

$$(\text{cov}[\mathbf{x}])_{ij} = \text{cov}[x_i, x_j] = \langle x_i x_j \rangle - \langle x_i \rangle \langle x_j \rangle$$
Bayesian probabilities

- Classical or frequentists interpretation: probabilities in terms of frequencies of random repeatable events

- Bayesian view: probabilities as subjective beliefs about uncertain event
  - event not necessarily repeatable
  - Bayesian inference to update belief given observations

Why probability theory?

- Cox: common sense axioms for degree of uncertainty $\rightarrow$ probability theory
**Cox axioms**

**Notation.** Let ‘the degree of belief in proposition $x$’ be denoted by $B(x)$. The negation of $x$ (NOT-$x$) is written $\overline{x}$. The degree of belief in a conditional proposition, ‘$x$, assuming proposition $y$ to be true’, is represented by $B(x \mid y)$.

**Axiom 1.** Degrees of belief can be ordered; if $B(x)$ is ‘greater’ than $B(y)$, and $B(y)$ is ‘greater’ than $B(z)$, then $B(x)$ is ‘greater’ than $B(z)$.

[Consequence: beliefs can be mapped onto real numbers.]

**Axiom 2.** The degree of belief in a proposition $x$ and its negation $\overline{x}$ are related. There is a function $f$ such that

$$B(x) = f[B(\overline{x})].$$

**Axiom 3.** The degree of belief in a conjunction of propositions $x, y$ ($x$ AND $y$) is related to the degree of belief in the conditional proposition $x \mid y$ and the degree of belief in the proposition $y$. There is a function $g$ such that

$$B(x, y) = g[B(x \mid y), B(y)].$$

**Box 2.4.** The Cox axioms. If a set of beliefs satisfy these axioms then they can be mapped onto probabilities satisfying $P(\text{FALSE}) = 0$, $P(\text{TRUE}) = 1$, $0 \leq P(x) \leq 1$, and the rules of probability:

$$P(x) = 1 - P(\overline{x}),$$

and

$$P(x, y) = P(x \mid y)P(y).$$
Motivational example: Medical diagnosis

Suppose $w = \{0, 1\}$ is a disease state (absent/present). The disease is rare, say $P(w = 1) = 0.01$. There is a test $x = 0, 1$ that measures whether the patient has the disease.

$$p(x = 1|w = 1) = p(x = 0|w = 0) = 0.9$$

The test is performed and is positive: $x = 1$. What is the probability that the patient has the disease?
\[ p(w = 1|x = 1) = \frac{0.9 \times 0.01}{0.9 \times 0.01 + 0.1 \times 0.99} = \frac{\frac{1}{0.9 \times 0.01}}{1 + \frac{0.1 \times 0.99}{0.9 \times 0.01}} = \frac{1}{12} = 0.0825 \]
Maximum likelihood estimation

Given a data set

\[ \text{Data} = \{x^1, \ldots, x^N\} \]

and a parametrized distribution

\[ p(x|w), \quad w = (w_1, \ldots, w_M), \]

find the value of \( w \) that best describes the data.

The common approach is to assume that the data that we observe are drawn independently from \( p(x|w) \) (independent and identical distributed = i.i.d.) for some unknown value of \( w \):

\[ p(\text{Data}|w) = p(\{x^1, \ldots, x^N\}|w) = \prod_{i=1}^{N} p(x^i|w) \]
Then, the most likely $w$ is obtained by maximizing $p(\text{Data}|w)$ wrt $w$:

$$w_{\text{ML}} = \arg\max_w p(\text{Data}|w) = \arg\max_w \prod_{i=1}^N p(x^i|w)$$

$$= \arg\max_w \sum_{i=1}^N \log p(x^i|w)$$

since $\log$ is a monotonically increasing function.

$w_{\text{ML}}$ is a function of the data. This is called an estimator.

Frequentists methods consider a single true $w$ and data generation mechanism $p(\text{Data}|w)$ provided by 'Nature' and study expected value:

$$\mathbb{E}w_{\text{ML}} = \sum_{\text{Data}} p(\text{Data}|w)w_{\text{ML}}(\text{Data})$$

For instance, $\hat{\mu} = \frac{1}{N} \sum_i x_i$ is estimator for the mean of a distribution. If data are $x_i \sim \mathcal{N}(\mu, \sigma^2)$ then $\mathbb{E}\hat{\mu} = \mu$. 
Bayesian machine learning

Model parameters $w$: uncertain

- Prior assumptions and beliefs about model parameters: the prior distribution $p(w)$
- Observed data $= \{x^1, \ldots, x^N\} = \text{Data}$
- Probability of data given $w$ (the likelihood): $p(\text{Data}|w)$

Apply Bayes’ rule to obtain the posterior distribution

$$p(w|\text{Data}) = \frac{p(\text{Data}|w)p(w)}{p(\text{Data})} \propto p(\text{Data}|w)p(w)$$

$$p(w) : \text{prior}$$

$$p(\text{Data}|w) : \text{likelihood}$$

$$p(\text{Data}) = \int p(\text{Data}|w)p(w) \, dw : \text{evidence}$$

$$\rightarrow p(w|\text{Data}) : \text{posterior}$$
Predictive distribution

Prior to 'learning', the predictive distribution for new observation $x$ is

$$p(x) = \int p(x|w)p(w)\,dw$$

After 'learning', i.e., after observation of Data, the predictive distribution for new observation $x$ becomes

$$p(x|\text{Data}) = \int p(x|w,\text{Data})p(w|\text{Data})\,dw$$

$$= \int p(x|w)p(w|\text{Data})\,dw$$
Bayesian vs frequentists view point

• Bayesians: there is a single fixed dataset (the one observed), and a probability distribution of model parameters $w$ which expresses a subjective belief including uncertainty about the ‘true’ model parameters.
  - They need a prior belief $p(w)$, and apply Bayes rule to compute $p(w|\text{Data})$.
  - Bayesians can talk about frequency that in repeated situations with this data $w$ has a certain value.

• Frequentists assume a single (unknown) fixed model parameter vector $w$.
  - construct an estimator $\hat{w}$ that is a function of the data. For instance, the maximum likelihood estimator
  - study statistical properties of estimators in similar experiments, each time with different datasets drawn from $p(\text{Data}|w)$, such as bias and variance.
  - They cannot make a claim for this particular data set. This is the price for not having a ‘prior’.

Bert Kappen
Toy example

$w$ is the probability that a coin comes up 'head'. Toss $N$ times with $N_H$ outcomes 'head'.

The likelihood of the data $p(N_H|w, N) = \left( \begin{array}{c} N \\ N_H \end{array} \right) w^{N_H}(1 - w)^{N-N_H}$.

The (frequentist) maximum likelihood estimator:

$$\hat{w} = \arg\max_w p(N_H|w, N) = \arg\max_w [N_H \log w + (N - N_H) \log(1 - w)] = \frac{N_H}{N}$$

$\hat{w}$ is stochastic variable, because it depends on the data set.

But it has 'nice' statistical property that on average (over many data sets) $\hat{w}$ gives the correct value:

$$\mathbb{E}\hat{w} = \sum_{N_H} p(N_H|w, N) \frac{N_H}{N} = w$$

1 Use $\sum_{N_H=0}^N p(N_H|w, N) = 1$. 

Bert Kappen
The Bayesian approach considers one data set and assumes a prior \( p(w) \) and computes the posterior

\[
p(w|N_H, N) = \frac{p(w)p(N_H|w, N)}{p(N_H, N)}
\]

Given \( N \) coin tosses and \( N_H \) heads, what is the probability that the next toss is head?
Bayesian vs frequentists

• Prior: inclusion of prior knowledge may be useful. True reflection of knowledge, or convenient construct? Bad prior choice can overconfidently lead to poor result.

• Bayesian integrals cannot be calculated in general. Only approximate results possible, requiring intensive numerical computations.

• Frequentists methods of ‘resampling the data’, (crossvalidation, bootstrapping) are appealing

• Bayesian framework transparent and consistent. Assumptions are explicit, inference is a mechanistic procedure (Bayesian machinery) and results have a clear interpretation.

This course place emphasis on Bayesian approach.
Gaussian distribution

Normal distribution = Gaussian distribution

\[ \mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left\{ -\frac{(x - \mu)^2}{2\sigma^2} \right\} \]

Specified by \( \mu \) and \( \sigma^2 \)
Gaussian is normalized,

\[ \int_{-\infty}^{\infty} N(x | \mu, \sigma^2) \, dx = 1 \]

The mean (= first moment), second moment, and variance are:

\[ \mathbb{E}[x] = \langle x \rangle = \int_{-\infty}^{\infty} x N(x | \mu, \sigma^2) \, dx = \mu \]

\[ \langle x^2 \rangle = \int_{-\infty}^{\infty} x^2 N(x | \mu, \sigma^2) \, dx = \mu^2 + \sigma^2 \]

\[ \text{var}[x] = \langle x^2 \rangle - \langle x \rangle^2 = \sigma^2 \]
Multivariate Gaussian

In $D$ dimensions

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

$x, \mu$ are $D$-dimensional vectors.

$\Sigma$ is a $D \times D$ covariance matrix, $|\Sigma|$ is its determinant.
Mean vector and covariance matrix

\[ \mathbf{E}[\mathbf{x}] = \langle \mathbf{x} \rangle = \int x \mathcal{N}(x|\mu, \Sigma) \, dx = \mu \]

\[ \text{cov}[\mathbf{x}] = \langle (\mathbf{x} - \mu)(\mathbf{x} - \mu)^T \rangle = \int (\mathbf{x} - \mu)(\mathbf{x} - \mu)^T \mathcal{N}(\mathbf{x}|\mu, \Sigma) \, dx = \Sigma \]

We can also write this in component notation:

\[ \mu_i = \langle x_i \rangle = \int x_i \mathcal{N}(x|\mu, \Sigma) \, dx \]

\[ \Sigma_{ij} = \langle (x_i - \mu_i)(x_j - \mu_j) \rangle = \int (x_i - \mu_i)(x_j - \mu_j) \mathcal{N}(\mathbf{x}|\mu, \Sigma) \, dx \]

\( \mathcal{N}(\mathbf{x}|\mu, \Sigma) \) is specified by its mean and covariance, in total \( D(D + 1)/2 + D \) parameters.
Transformation of variables (1-d)

Often it is easier to do the multidimensional integral in another coordinate frame. Suppose we want to do the integration

\[ \int_{y=c}^{d} f(y) \, dy \]

but the function \( f(y) \) is easier expressed as a \( f(y(x)) \) which is a function of \( x \). So we want to use \( x \) as integration variable. If \( y \) and \( x \) are related via invertible differentiable mappings \( y = y(x) \) and \( x = x(y) \) and the end points of the interval \( y = c, y = d \) are mapped to \( x = a, x = b \), (so \( c = y(a), \) etc) then we have the equality

\[
\int_{y=c}^{d} f(y) \, dy = \int_{y(x)=c}^{d} f(y(x)) \, dy(x)
\]

\[ = \int_{x=a}^{b} f(y(x)) y'(x) \, dx \]

The derivative \( y'(x) \) comes in as the ratio between the lengths of the differentials \( dy \) and \( dx \),

\[ dy = y'(x) \, dx \]
Several variables

With several variables, the substitution rule is generalized as follows. We have the invertible mapping \( y = y(x) \). Let us also assume that the region of integration of \( R \) is mapped by to \( S \), (so \( S = y(R) \)), then we have the equality

\[
\int_{y \in S} f(y) \, dy = \int_{y(x) \in S} f(y) \, dy(x)
\]

\[
= \int_{x \in R} f(y(x)) \left| \det \left( \frac{\partial y(x)}{\partial x} \right) \right| \, dx
\]

The factor \( \det \left( \frac{\partial y(x)}{\partial x} \right) \) is called the Jacobian of the coordinate transformation. Written out in more detail

\[
\det \left( \frac{\partial y(x)}{\partial x} \right) = \begin{vmatrix}
\frac{\partial y_1(x)}{\partial x_1} & \frac{\partial y_1(x)}{\partial x_2} & \cdots & \frac{\partial y_1(x)}{\partial x_n} \\
\frac{\partial y_2(x)}{\partial x_1} & \frac{\partial y_2(x)}{\partial x_2} & \cdots & \frac{\partial y_2(x)}{\partial x_n} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial y_n(x)}{\partial x_1} & \frac{\partial y_n(x)}{\partial x_2} & \cdots & \frac{\partial y_n(x)}{\partial x_n}
\end{vmatrix}
\]
The absolute value\(^2\) of the Jacobian comes in as the ratio between that the volume represented by the differential \(dy\) and the volume represented by the differential \(dx\), i.e.,

\[
dy = \left| \det \left( \frac{\partial y(x)}{\partial x} \right) \right| \, dx
\]

As a last remark, it is good to know that

\[
det \left( \frac{\partial x(y)}{\partial y} \right) = det \left( \left( \frac{\partial y(x)}{\partial x} \right)^{-1} \right) = \frac{1}{\det \left( \frac{\partial y(x)}{\partial x} \right)}
\]

\(^2\)In the single-variable case, we took the orientation of the integration interval into account \((\int_a^b f(x) \, dx = -\int_b^a f(x) \, dx)\). With several variables, this is awkward. Fortunately, it turns out that the orientation of the mapping of the domain always cancels to the 'orientation' of the Jacobian \((=\text{sign of the determinant})\). Therefore we take a positive orientation and the absolute value of the Jacobian.
Polar coordinates

Example: compute the area of a disc.

Consider a two-dimensional disc with radius $R$

$$D = \{ (x, y) | x^2 + y^2 < R^2 \}$$

Its area is

$$\int_D dx\,dy$$

This integral is easiest evaluated by going to ‘polar-coordinates’. The mapping from polar coordinates $(r, \theta)$ to Cartesian coordinates $(x, y)$ is

$$x = r \cos \theta$$

$$y = r \sin \theta$$

Since in polar coordinates, the disc is described by $0 \leq r < R$ (since $x^2 + y^2 = r^2$) and $0 \leq \theta < 2\pi$. 
The Jacobian is

\[ J = \begin{vmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} \end{vmatrix} = \begin{vmatrix} \cos \theta & -r \sin \theta \\ \sin \theta & r \cos \theta \end{vmatrix} = r(\cos^2 \theta + \sin^2 \theta) = r \]

In other words,

\[ dx \, dy = r \, dr \, d\theta \]

The area of the disc is now easily evaluated.

\[ \int_{\Delta} dx \, dy = \int_{\theta=0}^{2\pi} \int_{r=0}^{R} r \, dr \, d\theta = \pi R^2 \]
Transformation of densities

When $f(y) = p_y(y)$ a normalized probability density, then

$$\int f(y) \, dy = \int f(y(x)) \left| \det \left( \frac{\partial y(x)}{\partial x} \right) \right| \, dx$$

$$\int p_y(y) \, dy = \int p_x(x) \, dx$$

Probability densities:

- $p_y(y) \, dy$: probability that point falls in volume element $\delta y$ around $y$
- $p_x(x) \, dx$: same probability, now in terms of $x$

$$p_x(x) = p_y(y(x)) \left| \det \left( \frac{\partial y}{\partial x} \right) \right|$$

- Maximum of a probability density depends on choice of variable (see exercise 1.4).
Gaussian integral

How to compute

\[
\int_{-\infty}^{\infty} \exp(-x^2) dx
\]

\[
\left( \int_{-\infty}^{\infty} \exp(-x^2) dx \right)^2 = \int_{-\infty}^{\infty} \exp(-x^2) dx \int_{-\infty}^{\infty} \exp(-y^2) dy
\]

\[
= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp(-x^2) \exp(-y^2) dx dy
\]

\[
= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp(-(x^2 + y^2)) dx dy
\]
The latter is easily evaluated by going to polar-coordinates,

\[ \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp(-(x^2 + y^2)) \, dx \, dy = \int_{\theta=0}^{2\pi} \int_{r=0}^{\infty} \exp(-r^2) r \, dr \, d\theta \]

\[ = 2\pi \int_{r=0}^{\infty} \exp(-r^2) r \, dr \]

\[ = 2\pi \times \left( -\frac{1}{2} \exp(-r^2) \right) \bigg|_{0}^{\infty} \]

\[ = \pi \]

So

\[ \int_{-\infty}^{\infty} \exp(-x^2) \, dx = \sqrt{\pi} \]
The likelihood for the 1-d Gaussian

Consider 1-d data \( \text{Data} = \mathbf{x} = \{x_1, \ldots, x_N\} \). The likelihood of the data under a Gaussian model is the probability of the data, assuming each data point is independently drawn from the Gaussian distribution:

\[
p(\mathbf{x}|\mu, \sigma) = \prod_{n=1}^{N} \mathcal{N}(x_n|\mu, \sigma^2) = \left(\frac{1}{\sqrt{2\pi\sigma}}\right)^N \exp \left(-\frac{1}{2\sigma^2} \sum_{n=1}^{N} (x_n - \mu)^2\right)
\]
Maximum likelihood

Consider the log of the likelihood:

\[
\ln p(x|\mu, \sigma) = -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (x_n - \mu)^2 - \frac{N}{2} \ln \sigma^2 - \frac{N}{2} \ln 2\pi
\]

The values of \( \mu \) and \( \sigma \) that maximize the likelihood are given by (Ex. B1.11)

\[
\mu_{ML} = \frac{1}{N} \sum_{n=1}^{N} x_n \quad \sigma_{ML}^2 = \frac{1}{N} \sum_{n=1}^{N} (x_n - \mu_{ML})^2
\]
Bias in the ML estimates

Note that $\mu_{ML}, \sigma^2_{ML}$ are functions of the data. We can take their expectation value, assuming that $x_n$ is from a $\mathcal{N}(x|\mu, \sigma)$.

$$
\langle \mu_{ML} \rangle = \frac{1}{N} \sum_{n=1}^{N} \langle x_n \rangle = \mu \quad \langle \sigma^2_{ML} \rangle = \frac{1}{N} \sum_{n=1}^{N} \langle (x_n - \mu_{ML})^2 \rangle = \ldots = \frac{N-1}{N} \sigma^2
$$

The variance is estimated too low. This is called a biased estimator. Bias disappears when $N \to \infty$. In complex models with many parameters, the bias is more severe.
Now from a probabilistic perspective.

Target $t$ is Gaussian distributed around mean $y(x, \mathbf{w}) = \sum_{j=0}^{M} w_j x^j$,

$$p(t|x, \mathbf{w}, \beta) = \mathcal{N}(t|y(x, \mathbf{w}), \beta^{-1})$$

$\beta = 1/\sigma^2$ is the precision.
Curve fitting re-visited: ML

Training data: inputs $x = (x_1, \ldots, x_n)$, targets $t = (t_1, \ldots, t_n)$. (Assume $\beta$ is known.)

Likelihood,

$$ p(t|x, w) = \prod_{n=1}^{N} \mathcal{N}(t_n | y(x_n, w), \beta^{-1}) $$

Log-likelihood

$$ \ln p(t|x, w) = -\frac{\beta}{2} \sum_{n=1}^{N} (y(x_n, w) - t_n)^2 + \text{const}(\beta) $$

With $w_{ML}$ one can make predictions for a new input values $x$. The predictive distribution over the output $t$ is:

$$ p(t|x, w_{ML}) = \mathcal{N}(t | y(x, w_{ML}), \beta^{-1}) $$
Curve fitting re-visited MAP

Bayesian approach.
Prior:
\[ p(w|\alpha) = \mathcal{N}(w|\alpha^{-1}I) = \left(\frac{\alpha}{2\pi}\right)^{(M+1)/2} \exp \left(-\frac{\alpha}{2} w^T w\right) \]

\(M\) is the dimension of \(w\). Variables such as \(\alpha\), controlling the distribution of parameters, are called ‘hyperparameters’.

Posterior using Bayes rule:

\[ p(w|t, x, \alpha, \beta) \propto p(w|\alpha) \prod_{n=1}^{N} \mathcal{N}(t_n|y(x_n, w), \beta^{-1}) \]

\[-\ln p(w|t, x, \alpha, \beta) = \frac{\beta}{2} \sum_{n=1}^{N} (y(x_n, w) - t_n)^2 + \frac{\alpha}{2} w^T w + \text{const}(\beta)\]

Maximizing the posterior wrt \(w\) yields \(w_{MAP}\). Similar as Eq. 1.4. with \(\lambda = \alpha/\beta\)
Bayesian curve fitting

Given the training data \( x, t \) we are not so much interested in \( w \), but rather in the prediction of \( t \) for a new \( x \): \( p(t|x, x, t) \). This is given by

\[
p(t|x, x, t) = \int dwp(t|x, w)p(w|x, t)
\]

It is the average prediction of an ensemble of models \( p(t|x, w) \) parametrized by \( w \) and averaged wrt to the posterior distribution \( p(w|x, t) \).

All quantities depend on \( \alpha \) and \( \beta \).
Bayesian curve fitting

Generalized linear model with ‘basis functions’ e.g., $\phi_i(x) = x^i$,

$$y(x, w) = \sum_i \phi_i(x)w_i = \phi(x)^T w$$

So: prediction given $w$ is

$$p(t|x, w) = \mathcal{N}(t|y(x, w), \beta^{-1}) = \mathcal{N}(t|\phi(x)^T w, \beta^{-1})$$

The prediction given $x, t$ is

$$p(t|x, x, t) = \int dw p(t|x, w)p(w|x, t, \alpha, \beta)$$

The prior is Gaussian, the likelihood is a product of Gaussians. Thus the posterior is Gaussian. Thus, the integral can be performed analytically. The predictive distribution is Gaussian in $t$).
Result Bayesian curve fitting

The result is (section 3.3).

\[ p(t|x, x, t) = \mathcal{N}(t|m(x), s^2(x)) \]

\[ m(x) = \beta \phi(x)^T S \sum_{n=1}^{N} \phi(x_n) t_n = \phi(x)^T w_{MAP} \]

\[ s^2(x) = \beta^{-1} + \phi(x)^T S \phi(x) \]

\[ S^{-1} = \alpha I + \beta \sum_{n=1}^{N} \phi(x_n) \phi(x_n)^T \]

Note, \( s^2 \) depend on \( x \). First term as in ML estimate describes noise in target for fixed \( w \). Second term describes noise due to uncertainty in \( w \).
Polynomial curve fitting with $M = 9$, $\alpha = 5 \times 10^{-3}$, $\beta = 11.1$. Red: $m(x) \pm s(x)$. Note $\sqrt{\beta^{-1}} = 0.3$
Model selection

Q: If we have different models to describe the data, which one should we choose?
Model selection/Cross validation

Q: If we have different models to describe the data, which one should we choose?

A1: If data is plenty, use separate validation set to select model with best generalization performance, and a third independent test set for final evaluation.

A2: Small validation set: use S-fold cross validation.

A3: Information criteria: penalty for complex models
• Akaike IC (AIC): $\ln p(D|w_{ML}) - M$
• Bayesian IC (BIC): Bayesian + crude approximations (section 4.4.1)
• Full Bayesian $\rightarrow$ penalties arises automatically (section 3.4)
Sofar, we considered $x$ one-dimensional. How does pattern recognition work higher dimensions?

Two components of 12-dimensional data that describe gamma ray measurements of a mixture of oil, water and gas. The mixture can be in three states: homogenous (red), annular (green) and laminar (blue).

Classification of the 'x' can be done by putting a grid on the space and assigning the class that is most numerous in the particular box.
High-dimensional data/Binning

Q: What is the disadvantage of this approach?
Curse of dimensionality/Binning

Q: What is the disadvantage of this approach?
A: This approach scales exponentially with dimensions.

In $D$ dimensions: grid with length $n$ consists of . . . hypercubes.
Curse of dimensionality/Binning

Q: What is the disadvantage of this approach?
A: This approach scales exponentially with dimensions.

In $D$ dimensions: grid with length $n$ consists of $n^D$ hypercubes.
Curse of dimensionality/Polynomials

The polynomial function considered previously becomes in $D$ dimensions:

$$y(x, w) = w_0 + \sum_{i=1}^{D} w_i x_i + \sum_{i=1}^{D} \sum_{j=1}^{D} w_{ij} x_i x_j + \sum_{i=1}^{D} \sum_{j=1}^{D} \sum_{k=1}^{D} w_{ijk} x_i x_j x_k$$

(here up to order $M = 3$).

The number of coefficients scales as . . . (unpractically large).
Curse of dimensionality/Polynomials

The polynomial function considered previously becomes in $D$ dimensions:

$$y(x, \mathbf{w}) = w_0 + \sum_{i=1}^{D} w_i x_i + \sum_{i=1}^{D} \sum_{j=1}^{D} w_{ij} x_i x_j + \sum_{i=1}^{D} \sum_{j=1}^{D} \sum_{k=1}^{D} w_{ijk} x_i x_j x_k$$

(here up to order $M = 3$).

The number of coefficients scales as $D^M$ (unpractically large).
Curse of dimensionality/Spheres

Q: In a 10-dimensional hypersphere with radius $R = 1$: what is the volume fraction lying in the outer “shell” between $r = 0.5$ and $r = 1$?
Curse of dimensionality/Spheres

Q: In a 10-dimensional hypersphere with radius $R = 1$: what is the volume fraction lying in the outer “shell” between $r = 0.5$ and $r = 1$?

A: More than 0.999!

\[
V_D(r) = K_D r^D \quad \frac{V_D(1) - V_D(1 - \epsilon)}{V_D(1)} = 1 - (1 - \epsilon)^D
\]

Spheres in high dimension have most of their volume on the boundary.

So in high dimensions, almost all other data points are at more or less the same distance!
Curse of dimensionality/Gaussians

In high dimensions, the distribution of the radius of a Gaussian with variance $\sigma$ is concentrated around a thin shell $r \approx \sigma \sqrt{D - 1}$.

\[
\frac{1}{Z} e^{-\left(\frac{x_1^2 + \ldots + x_n^2}{2\sigma^2}\right)} = \frac{1}{Z} \left| \det \frac{dx_i}{d(r, \phi_j)} \right| e^{-r^2/2\sigma^2} = \frac{1}{Z} r^{n-1} e^{-r^2/2\sigma^2}
\]

\[
\frac{d}{dr} \left( -\frac{r^2}{2\sigma^2} + (n - 1) \log r \right) = 0 \rightarrow r = \sigma \sqrt{n - 1}
\]
Curse of dimensionality

Is machine learning even possible in high dimensions?

Data often in low dimensional subspace: only a few dimensions are relevant.

- Images of objects are $N$-dimensional, with $N$ the number of pixels.

- not all pixels independent:
  - Smoothness, nearby pixels are similar
  - changing an object (position, orientation in 3d, or lighting) changes pixels in correlated way.

Effectively the data are on a $\ll N$-dimensional manifold (curved subspace)
Decision theory

**Inference**: given pairs \((x, t)\), learn \(p(x, t)\) and estimate \(p(x, t)\) for new value of \(x\) (and possibly all \(t\)).

**Decision**: for new value of \(x\) estimate 'best' \(t\).

**Example**: in a medical application, \(x\) is an X-ray image and \(t\) a class label that indicates whether the patient has cancer \((t = C_1)\) or not \((t = C_2)\).
Decision theory

Inference: given pairs \((x,t)\), learn \(p(x,t)\) and estimate \(p(x,t)\) for new value of \(x\) (and possibly all \(t\)).

Decision: for new value of \(x\) estimate 'best' \(t\).

Example: in a medical application, \(x\) is an X-ray image and \(t\) a class label that indicates whether the patient has cancer \((t = C_1)\) or not \((t = C_2)\).

Bayes' theorem:

\[
p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)}
\]

\(p(C_k)\) is the prior probability of class \(C_k\). \(p(C_k|x)\) is the posterior probability of class \(C_k\) after seeing the image \(x\).
A classifier is specified by defining regions $R_k$, such that all $x \in R_k$ are assigned to class $C_k$. In the case of two classes, the probability that this classifier gives the correct answer is

$$p(\text{correct}) = p(x \in R_1, C_1) + p(x \in R_2, C_2) = \int_{R_1} p(x, C_1) dx + \int_{R_2} p(x, C_2) dx$$

$p(\text{correct})$ is maximized when the regions $R_k$ are chosen such that

$$k = \arg\max_k p(x, C_k) = \arg\max_k p(C_k | x)$$
Decision theory/Example

Example: in a medical application, \( x \) is an X-ray image and \( t \) a class label that indicates whether the patient has cancer (\( t = C_1 \)) or not (\( t = C_2 \)).

Q: Suppose \( p(C_1) = 0.01 \) and \( p(C_1|x) = 0.3 \) according to our model. Do we decide that the patient has cancer and therefore start treatment?
Decision theory/Example

**Example**: in a medical application, $x$ is an X-ray image and $t$ a class label that indicates whether the patient has cancer ($t = C_1$) or not ($t = C_2$).

Q: Suppose $p(C_1) = 0.01$ and $p(C_1|x) = 0.3$ according to our model. Do we decide that the patient has cancer and therefore start treatment?

A: If we want to maximize the chance of making the correct decision, we have to pick $k$ such that $p(C_k|x)$ is maximal. Because $p(C_1|x) = 0.3$ and $p(C_2|x) = 0.7$, the answer is no: we decide that the patient does not have cancer.
### Decision theory/Expected loss

Typically, not all classification errors are equally bad: classifying a healthy patient as sick, is not as bad as classifying a sick patient as healthy.

$$L_{kj} = \begin{pmatrix} 0 & 1000 \\ 1 & 0 \end{pmatrix}$$

Rows ($k$) are true classes (cancer, normal); columns ($j$) are assigned classes (cancer, normal).

The probability to assign an $x$ to class $j$ while $x$ belongs to class $k$ is $p(x \in R_j, C_k)$. Thus the total expected loss is

$$\langle L \rangle = \sum_k \sum_j L_{kj} p(x \in R_j, C_k) = \sum_j \int_{R_j} \sum_k p(x, C_k) L_{kj} dx$$

$\langle L \rangle$ is minimized if each $x$ is assigned to class $j$ such that $\sum_k L_{kj} p(x, C_k)$ is minimal, or equivalently, such that $\sum_k L_{kj} p(C_k | x)$ is minimal.
Decision theory/Example

Example: in a medical application, $x$ is an X-ray image and $t$ a class label that indicates whether the patient has cancer ($t = C_1$) or not ($t = C_2$).

\[
\begin{array}{cc}
C_1 & C_2 \\
C_1 & 0 & 1000 \\
C_2 & 1 & 0 \\
\end{array}
\]

$L_{kj} = \begin{pmatrix} 0 & 1000 \\ 1 & 0 \end{pmatrix}$

Rows ($k$) are true classes (cancer, normal); columns ($j$) are assigned classes (cancer, normal).

Q: Suppose $p(C_1) = 0.01$ and $p(C_1|x) = 0.3$ according to our model. Do we decide that the patient has cancer and therefore start treatment?
Decision theory/Example

Example: in a medical application, \( x \) is an X-ray image and \( t \) a class label that indicates whether the patient has cancer (\( t = C_1 \)) or not (\( t = C_2 \)).

\[
\begin{array}{cc}
C_1 & C_2 \\
C_1 & 0 & 1000 \\
C_2 & 1 & 0 \\
\end{array}
\]

\( L_{kj} = \begin{pmatrix} 0 & 1000 \\ 1 & 0 \end{pmatrix} \)

Rows (\( k \)) are true classes (cancer, normal); columns (\( j \)) are assigned classes (cancer, normal).

Q: Suppose \( p(C_1) = 0.01 \) and \( p(C_1|x) = 0.3 \) according to our model. Do we decide that the patient has cancer and therefore start treatment?

A: If we want to minimize the expected loss, we have to pick \( j \) such that \( \sum_k L_{kj} p(C_k|x) \) is minimal.

For \( j = 1 \), this yields \( \sum_k L_{kj} p(C_k|x) = 0 \times 0.3 + 1 \times 0.7 = 0.7 \),

for \( j = 2 \), this yields \( \sum_k L_{kj} p(C_k|x) = 1000 \times 0.3 + 0 \times 0.7 = 300 \).

Therefore, we now decide that the patient has cancer (better safe than sorry).
Decision theory/Reject option

It may be that $\max_j p(C_j|x)$ is small, indicating that it is unclear to which class $x$ belongs.

In that case, a different decision can be taken: the “reject” or “don’t know” option.

This can be done by introducing a threshold $\theta \in [0, 1]$ and only classify those $x$ for which $\max_j p(C_j|x) > \theta$ (and answer “don’t know” otherwise).
Decision theory/Discriminant functions

Instead of first learning a probability model and then making a decision, one can also directly learn a decision rule (a classifier) without the intermediate step of a probability model.

A set of approaches:

- Learn a model for the class conditional probabilities $p(x|C_k)$. Use Bayes’ rule to compute $p(C_k|x)$ and construct a classifier using decision theory. This approach is the most complex (section 4.2).

- Learn the inference problem $p(C_k|x)$ directly and construct a classifier using decision theory. This approach is simpler, since no input model $p(x)$ is learned (see figure) (section 4.3)

- Learn $f(x)$, called a discriminant function, that maps $x$ directly onto a class label 0, 1, 2, …. Even simpler, only decision boundary is learned. But information on the expected classification error is lost (section 4.1, not treated).
Example that shows that detailed structure in the joint model need not affect class conditional probabilities. Learning only the decision boundary is the simplest approach.

Approaches that model the distribution of both inputs and outputs are called *generative models*, approaches that only model the conditional distribution of the output given the input are called *discriminative models*. 
**Loss functions for regression**

Decision theory generalizes straightforwardly to continuous variables: the loss matrix $L_{jk}$ becomes a loss function $L(t, y(x))$.

$L$ measures the 'cost' when the model outputs $y(x)$ whereas the correct output is $t$.

Examples:

Minkowski loss function $L_q = |y - t|^q$ for various values of $q$. 
Loss functions for regression/Quadratic loss

The average/expected loss is:

$$\langle L \rangle = \int L(t, y(x)) p(x, t) \, dx \, dt$$

For the quadratic loss function $L_2(t, y(x)) = (t - y(x))^2$ one can derive that the expected loss is minimized by taking

$$y(x) = \mathbb{E}_t[t|x]$$

i.e., by the mean of the conditional distribution $p(t|x)$. (The minimum of $\langle L_1 \rangle$ is obtained by the conditional median.)
Information theory

Information is a measure of the 'degree of surprise' that a certain value gives us. Unlikely events are informative, likely events less so. Certain events give us no additional information. Thus, information decreases with the probability of the event.

Let us denote $h(x)$ the information of $x$. Then if $x, y$ are two independent events: $h(x, y) = h(x) + h(y)$. Since $p(x, y) = p(x)p(y)$ we see that

$$h(x) = - \log_2 p(x)$$

is a good candidate to quantify the information in $x$.

If $x$ is observed repeatedly then the expected information is

$$H[x] := \langle - \log_2 p \rangle = - \sum_x p(x) \log_2 p(x)$$

is the entropy of the distribution $p$. 
Information theory

Example 1: $x$ can have 8 values with equal probability, then $H(x) = -8 \times \frac{1}{8} \log \frac{1}{8} = 3$ bits.

Example 2: $x$ can have 8 values with probabilities \((\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64})\). Then

\[
H(x) = -\frac{1}{2} \log \frac{1}{2} - \frac{1}{4} \log \frac{1}{4} - \frac{1}{8} \log \frac{1}{8} - \frac{1}{16} \log \frac{1}{16} - 4 \frac{1}{64} \log \frac{1}{64} = 2 \text{bits}
\]

which is smaller than for the uniform distribution.

*Noiseless coding theorem:* Entropy is a lower bound on the average number of bits needed to transmit a random variable (Shannon 1948).

Q: How can we transmit $x$ in example 2 most efficiently?
**Information theory**

Example 1: \( x \) can have 8 values with equal probability, then \( H(x) = -8 \times \frac{1}{8} \log \frac{1}{8} = 3 \) bits.

Example 2: \( x \) can have 8 values with probabilities \( \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}\right) \). Then

\[
H(x) = -\frac{1}{2} \log \frac{1}{2} - \frac{1}{4} \log \frac{1}{4} - \frac{1}{8} \log \frac{1}{8} - \frac{1}{16} \log \frac{1}{16} - \frac{4}{64} \log \frac{1}{64} = 2 \text{bits}
\]

which is smaller than for the uniform distribution.

*Noiseless coding theorem*: Entropy is a lower bound on the average number of bits needed to transmit a random variable (Shannon 1948).

A: We can encode \( x \) as a 3 bit binary number, in which case the expected code length is 3 bits. We can do better, by coding likely \( x \) smaller and unlikely \( x \) larger, for instance 0, 10, 110, 1110, 111100, 111101, 111110, 111111. Then

\[
\text{Av.code length} = \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{16} \times 4 + \frac{4}{64} \times 6 = 2 \text{bits}
\]
When \( x \) has values \( x_i, i = 1, \ldots, M \), then

\[
H[x] = - \sum_i p(x_i) \log p(x_i)
\]

When \( p \) is sharply peaked (\( p(x_1) = 1, p(x_2) = \ldots = p(x_M) = 0 \)) then the entropy is

\[
H[x] = -1 \log 1 - (M - 1)0 \log 0 = 0
\]

When \( p \) is flat (\( p(x_i) = 1/M \)) the entropy is maximal

\[
H[x] = -M \frac{1}{M} \log \frac{1}{M} = \log M
\]
Information theory/Maximum entropy

For \( p(x) \) a distribution density over a continuous value \( x \) we define the (differential) entropy as

\[
H[x] = - \int p(x) \log p(x) \, dx
\]

Suppose that all we know about \( p \) is its mean \( \mu \) and its variance \( \sigma^2 \).

Q: What is the distribution \( p \) with mean \( \mu \) and variance \( \sigma^2 \) that is as uninformative as possible, i.e., which maximizes the entropy?
Information theory/Maximum entropy

For \( p(x) \) a distribution density over a continuous value \( x \) we define the (differential) entropy as

\[
H[x] = - \int p(x) \log p(x) \, dx
\]

Suppose that all we know about \( p \) is its mean \( \mu \) and its variance \( \sigma^2 \).

Q: What is the distribution \( p \) with mean \( \mu \) and variance \( \sigma^2 \) that is as uninformative as possible, i.e., which maximizes the entropy?

A: The Gaussian distribution \( \mathcal{N}(x|\mu, \sigma^2) \) (exercise 1.34 and 1.35).
Relative entropy or Kullback-Leibler divergence or KL-divergence:

\[
\text{KL}(p||q) = - \sum_i p_i \ln q_i - \left( - \sum_i p_i \ln p_i \right) \\
= - \sum_i p_i \ln \left\{ \frac{q_i}{p_i} \right\}
\]

- Additional amount of information required to specify \(i\) when \(q\) is used for coding rather than the true distribution \(p\).
- Divergence between ‘true’ distribution \(p\) and ‘approximate’ distribution \(q\).
- \(\text{KL}(p||q) \neq \text{KL}(q||p)\)
- \(\text{KL}(p||q) \geq 0, \quad \text{KL}(p||q) = 0 \Leftrightarrow p = q\) (use convex functions)
- with continuous variables: \(\text{KL}(p||q) = - \int p(x) \ln \left\{ \frac{q(x)}{p(x)} \right\} dx\)
Convex functions

A convex function is one where every chord lies on or above the function. Mathematically, a function \( f \) is convex if for all \( \lambda \in [0, 1] \) and all \( a, b \),

\[
    f(\lambda a + (1 - \lambda)b) \leq \lambda f(a) + (1 - \lambda)f(b)
\]

Examples:
- \( f(x) = ax + b \)
- \( f(x) = x^2 \)
- \( f(x) = -\ln(x) \)
- \( f(x) = x \ln(x) \) (exercise).

Convex functions are \( \cup \) shaped, while concave functions are \( \cap \) shaped.

Convex functions have non-negative second derivatives.

\[ f \text{ is convex } \iff \quad f(\lambda a + (1 - \lambda)b) \leq \lambda f(a) + (1 - \lambda)f(b) \quad \forall \lambda \in [0, 1], \forall a, b \]
Convex functions/Jensen’s inequality

Convex functions satisfy *Jensen’s inequality*

\[
f \left( \sum_{i=1}^{M} \lambda_i x_i \right) \leq \sum_{i=1}^{M} \lambda_i f(x_i)
\]

where \( \lambda_i \geq 0, \sum_i \lambda_i = 1 \), for any set points \( x_i \).

In other words:

\[
f(\langle x \rangle) \leq \langle f(x) \rangle
\]

Example: to show that \( KL(p||q) \), we apply Jensen’s inequality with \( \lambda_i = p_i \), making use of the fact that \( -\ln(x) \) is convex:

\[
KL(p||q) = -\sum_i p_i \ln \left( \frac{q_i}{p_i} \right) \geq -\ln \left( \sum_i p_i \frac{q_i}{p_i} \right) = -\ln \left( \sum_i q_i \right) = 0
\]
Information theory and density estimation

Relation with maximum likelihood:

**Empirical distribution:**

\[ p(x) = \frac{1}{N} \sum_{n=1}^{N} \delta(x - x_n) \]

Approximating distribution (model) : \( q(x|\theta) \)

\[
\text{KL}(p||q) = -\int p(x) \ln q(x|\theta) dx - \int p(x) \ln p(x) dx \\
= -\frac{1}{N} \sum_{n=1}^{N} \ln q(x_n|\theta) + \text{const.}
\]

Thus, minimizing the KL-divergence between the empirical distribution \( p(x) \) and the model distribution \( q(x|\theta) \) is equivalent to maximum likelihood (i.e., maximizing the likelihood of i.i.d. data with respect to the the model parameters \( \theta \)).
Information theory/mutual information

Mutual information between $x$ and $y$: KL divergence between joint distribution $p(x, y)$ and product of marginals $p(x)p(y)$,

$$I[x, y] \equiv KL(p(x, y) \| p(x)p(y))$$

$$= - \int \int p(x, y) \ln \left( \frac{p(x)p(y)}{p(x, y)} \right) dx dy$$

• $I(x, y) \geq 0$, equality iff $x$ and $y$ independent

Relation with conditional entropy

Lagrange multipliers

Minimize $f(x)$ under constraint: $g(x) = 0$.

Fancy formulation: define *Lagrangian*,

$$L(x, \lambda) = f(x) + \lambda g(x)$$

$\lambda$ is called a Lagrange multiplier.

The constraint minimization of $f$ w.r.t $x$ equivalent to *unconstrained* minimization of $\max_\lambda L(x, \lambda)$ w.r.t $x$. The *maximization* w.r.t to $\lambda$ yields the following function of $x$

$$\max_\lambda L(x, \lambda) = f(x) \quad \text{if } g(x) = 0$$

$$\max_\lambda L(x, \lambda) = \infty \quad \text{otherwise}$$
Lagrange multipliers

Under certain conditions, in particular $f(x)$ convex (i.e. the matrix of second derivatives positive definite) and $g(x)$ linear,

$$\min_x \max_\lambda L(x, \lambda) = \max_\lambda \min_x L(x, \lambda)$$

Procedure:

1. Minimize $L(x, \lambda)$ w.r.t $x$, e.g. by taking the gradient and set to zero. This yields a (parametrized) solution $x(\lambda)$.

2. Maximize $L(x(\lambda), \lambda)$ w.r.t. $\lambda$. The solution $\lambda^*$ is precisely such that $g(x(\lambda^*)) = 0$.

3. The solution of the constraint optimization problem is

$$x^* = x(\lambda^*)$$
Appendix E

Example

Maximize

\[ f(x_1, x_2) = 1 - x_1^2 - x_2^2 \quad \text{and constraint} \quad g(x_1, x_2) = x_1 + x_2 - 1 = 0 \]

Lagrangian:

\[ L(x_1, x_2, \lambda) = 1 - x_1^2 - x_2^2 + \lambda(x_1 + x_2 - 1) \]

Maximize \( L \) w.r.t. \( x_i \) gives \( x_1(\lambda) = x_2(\lambda) = \frac{1}{2} \lambda \).

Plug into constraint: \( x_1(\lambda) + x_2(\lambda) - 1 = \lambda - 1 = 0 \).

So \( \lambda = 1 \) and \( x_1^* = \frac{x_2^*}{2} \).
Some remarks

• Works as well for maximization (of concave functions) under constraints. The procedure is essentially the same.

• The sign in front of the $\lambda$ can be chosen as you want:

$$L(x, \lambda) = f(x) + \lambda g(x) \text{ or } L(x, \lambda) = f(x) - \lambda g(x)$$

work equally well.

• More constraints? For each constraint $g_i(x) = 0$ a Lagrange multiplier $\lambda_i$, so

$$L(x, \lambda) = f(x) + \sum_i \lambda_i g_i(x)$$

• Similar methods apply for inequality constraints $g(x) \geq 0$ (restricts $\lambda$).
The Perceptron

Relevant in history of pattern recognition and neural networks.

- Perceptron learning rule + convergence, Rosenblatt (1962)

- Perceptron critique (Minsky and Papert, 1969) → "Dark ages of neural networks"

- Revival in the 80’s: Backpropagation
The Perceptron

\[ y(x) = \text{sign}(w^T \phi(x)) \]

where

\[
\text{sign}(a) = \begin{cases} 
+1, & a \geq 0 \\
-1, & a < 0.
\end{cases}
\]

and \( \phi(x) \) is a feature vector (e.g. hard wired neural network).

Figure 3.10. The perceptron network used a fixed set of processing elements, denoted \( \phi_j \), followed by a layer of adaptive weights \( w_j \) and a threshold activation function \( g(\cdot) \). The processing elements \( \phi_j \) typically also had threshold activation functions, and took inputs from a randomly chosen subset of the pixels of the input image.
The Perceptron

Ignore $\phi$, ie. consider inputs $x^{\mu}$ and outputs $t^{\mu} = \pm 1$
Define $w^T x = \sum_{j=1}^{n} w_j x_j + w_0$. Then, the learning condition becomes

$$\text{sign}(w^T x^{\mu}) = t^{\mu}, \quad \mu = 1, \ldots, P$$

We have

$$\text{sign}(w^T x^{\mu} t^{\mu}) = 1 \quad \text{or} \quad w^T z^{\mu} > 0$$

with $z_j^{\mu} = x_j^{\mu} t^{\mu}$. 
Linear separation

Classification depends on sign of $w^T x$. Thus, decision boundary is hyper plane:

$$0 = w^T x = \sum_{j=1}^{n} w_j x_j + w_0$$

Perceptron can solve linearly separable problems.

AND problem is linearly separable.

XOR problem and linearly dependent inputs not linearly separable.
Perceptron learning rule

Learning successful when

\[ w^T z^\mu > 0, \quad \text{all patterns} \]

Learning rule is 'Hebbian':

\[
\begin{align*}
    w_j^{\text{new}} &= w_j^{\text{old}} + \Delta w_j \\
    \Delta w_j &= \eta \Theta(-w^T z^\mu) x_j^\mu t^\mu = \eta \Theta(-w^T z^\mu) z_j^\mu
\end{align*}
\]

\( \eta \) is the learning rate.

Depending on the data, there may be many or few solutions to the learning problem (or non at all)
The quality of the solution is determined by the worst pattern. Since the solution does not depend on the size of $w$:

$$D(w) = \frac{1}{|w|} \min_{\mu} w^T z^\mu$$

Acceptable solutions have $D(w) > 0$.

The best solution is given by $D_{\text{max}} = \max_w D(w)$.
$D_{\text{max}} > 0$ iff the problem is linearly separable.
Convergence of Perceptron rule

Assume that the problem is linearly separable, so that there is a solution $w^*$ with $D(w^*) > 0$.

At each iteration, $w$ is updated only if $w \cdot z^\mu < 0$. Let $M^\mu$ denote the number of times pattern $\mu$ has been used to update $w$. Thus,

$$w = \eta \sum_\mu M^\mu z^\mu$$

Consider the quantity

$$-1 < \frac{w \cdot w^*}{|w||w^*|} < 1$$

We will show that

$$\frac{w \cdot w^*}{|w||w^*|} \geq \mathcal{O}(\sqrt{M}),$$

with $M = \sum_\mu M^\mu$ the total number of iterations.

Therefore, $M$ cannot grow indefinitely. Thus, the perceptron learning rule converges in a finite number of steps when the problem is linearly separable.
Proof:

\[ w \cdot w^* = \eta \sum_{\mu} M^\mu z^\mu \cdot w^* \geq \eta M \min_{\mu} z^\mu \cdot w^* \]

\[ = \eta M D(w^*) |w^*| \]

\[ \Delta |w|^2 = |w + \eta z^\mu|^2 - |w|^2 = 2 \eta w \cdot z^\mu + \eta^2 |z^\mu|^2 \]

\[ \leq \eta^2 |z^\mu|^2 = \eta^2 N \]

\[ |w| \leq \eta \sqrt{NM} \]

Thus,

\[ 1 \geq \frac{w \cdot w^*}{|w||w^*|} \geq \sqrt{M} \frac{D(w^*)}{\sqrt{N}} \]

Number of weight updates:

\[ M \leq \frac{N}{D^2(w^*)} \]
Capacity of the Perceptron

Consider $P$ patterns in $N$ dimensions in general position:
- no subset of size less than $N$ is linearly dependent.
- general position is necessary for linear separability

Question: What is the probability that a problem of $P$ samples in $N$ dimensions is linearly separable?
Define $C(P, N)$ the number of linearly separable colorings on $P$ points in $N$ dimensions, with separability plane through the origin. Then (Cover 1966):

$$C(P, N) = 2 \sum_{i=0}^{N-1} \binom{P-1}{i}$$

When $P \leq N$ small, then $C(P, N) = 2 \sum_{i=0}^{P-1} \binom{P-1}{i} = 2(1+1)^{P-1} = 2^P$

When $P = 2N$, then 50% is linearly separable: $C(P, N) = 2 \sum_{i=0}^{N-1} \binom{2N-1}{i} = \sum_{i=0}^{2N-1} \binom{2N-1}{i} = 2^{2N-1} = 2^{P-1}$
Proof by induction.

Add one point $X$. The set $C(P, N)$ consists of
- colorings with separator through $X$ (A)
- rest (B)

Thus,

$$C(P + 1, N) = 2A + B = C(P, N) + A$$
$$= C(P, N) + C(P, N - 1)$$

Yields

$$C(P, N) = 2 \sum_{i=0}^{N-1} \binom{P-1}{i}$$
Input-output behaviour:

**Linear units**

\[ y^\mu = w_0 + \sum_{j=1}^{n} w_j x_j^\mu = \sum_{j=0}^{n} w_j x_j \quad x_0 = 1 \]

Desired behaviour: \( y^\mu = t^\mu, \mu = 1, \ldots, P \).

Define a learning rule as gradient descent on a cost function:

\[
E(w) = \frac{1}{2} \sum_{\mu=1}^{P} \left( t^\mu - \sum_{j=0}^{n} w_j x_j^\mu \right)^2
\]

\[
\Delta w_i = -\eta \frac{\partial E}{\partial w_i} = \eta \sum_{\mu} \left( t^\mu - \sum_{j} w_j x_j^\mu \right) x_i^\mu
\]

\[
w_i = w_i + \Delta w_i
\]
Gradient descent optimization

The simplest procedure to optimize $E$ is to start with a random $\mathbf{w}$ and iterate

$$\mathbf{w}^{\tau+1} = \mathbf{w}^\tau - \eta \nabla E(\mathbf{w}^\tau)$$

This is called batch learning, where all training data are included in the computation of $\nabla E$.

Does this algorithm converge? Yes, if $\epsilon$ is "sufficiently small" and $E$ bounded from below.

Proof: Denote $\Delta \mathbf{w} = -\eta \nabla E$.

$$E(\mathbf{w} + \Delta \mathbf{w}) \approx E(\mathbf{w}) + (\Delta \mathbf{w})^T \nabla E = E(\mathbf{w}) - \eta \sum_i \left( \frac{\partial E}{\partial w_i} \right)^2 \leq E(\mathbf{w})$$

In each gradient descent step the value of $E$ is lowered. Since $E$ bounded from below, the procedure must converge asymptotically.
Convergence of gradient descent in a quadratic well

\[ E(w) = \frac{1}{2} \sum_i \lambda_i w_i^2 \]

\[ \Delta w_i = -\eta \frac{\partial E}{\partial w_i} = -\eta \lambda_i w_i \]

\[ w_i^{\text{new}} = w_i^{\text{old}} + \Delta w_i = (1 - \eta \lambda_i) w_i \]

Convergence when \(|1 - \eta \lambda_i| < 1\). Oscillations when \(1 - \eta \lambda_i < 0\).

Optimal learning parameter depends on curvature of each dimension.
Non-linear units/logistic regression

When the transfer function is non-linear,

\[ y^\mu = g(h^\mu), \quad h^\mu = \sum_j w_j x_j^\mu \]

the cost criterion for learning becomes

\[
E(w) = \frac{1}{2} \sum_\mu (t^\mu - y^\mu)^2 = \frac{1}{2} \sum_\mu \left( t^\mu - g \left( \sum_j w_j x_j^\mu \right) \right)^2
\]

\[
\Delta w_i = -\eta \frac{\partial E}{\partial w_i} = \sum_\mu (t^\mu - y^\mu) g'(h^\mu) x_i^\mu
\]
Non-linear units/logistic regression

For $y = \pm 1$ we define

$$p(y|x) = \sigma(hy) \quad h = \sum_j w_j x_j$$

$$\sigma(x) = (1 + e^{-2x})^{-1} = \frac{1}{2}(1 + \tanh(x)).$$

Note that $p(y = -1|x) = \sigma(-h) = 1 - \sigma(h) = 1 - p(y = 1|x)$.

When the target distribution is given by $q(y|x^\mu), \mu = 1, \ldots, N$, we use the KL divergence as cost function for learning:

$$E = \sum_\mu \sum_{y=\pm 1} q(y|x^\mu) \log \left( \frac{q(y|x^\mu)}{p(y|x^\mu)} \right) = - \sum_\mu \sum_{y=\pm 1} q(y|x^\mu) \log p(y|x^\mu) + \text{const.}$$

For a specific training set, we have $q(y|x^\mu) = \delta_{y,t}$. Thus, KL reduces to max likelihood:

$$E = - \sum_\mu \log p(t^\mu|x^\mu)$$
Gradient descent on this learning rule yields

\[
\frac{\partial p(t^\mu | x^\mu)}{\partial w_j} = 2 p(t^\mu | x^\mu) p(-t^\mu | x^\mu) t^\mu x_j^\mu
\]

\[
\frac{\partial E}{\partial w_j} = -2 \sum_\mu p(-t^\mu | x^\mu) t^\mu x_j^\mu
\]

\[
= - \sum_\mu (t^\mu - \langle y \rangle^\mu) x_j^\mu
\]

This is equivalent with HKP (5.57-58).

\[
\frac{d}{dx} \sigma(x) = 2(1 + e^{-2x})^{-2} e^{-2x} = 2\sigma(x) \frac{e^{-2x}}{1 + e^{-2x}} = 2\sigma(x) \sigma(-x)
\]

\[
\langle y \rangle^\mu = \sum_{y=\pm 1} y p(y | x^\mu) = \sum_{y=\pm t^\mu} y p(y | x^\mu) = t^\mu p(t^\mu | x^\mu) - t^\mu p(-t^\mu | x^\mu) = t^\mu (1 - 2 p(-t^\mu | x^\mu))
\]

\[
\langle y \rangle^\mu - t^\mu = -2t^\mu p(-t^\mu | x^\mu)
\]
Logistic regression

Two class classification

\[ p(C_1|\phi) = \sigma(w^T \phi) \quad \sigma(x) = 1/(1 + \exp(-x)) \]

\( M \) dimensional feature space.

Left: Data from two classes in two dimensions \((x_1, x_2)\) and two two Gaussian basis functions \(\phi_1(x), \phi_2(x)\) (green). Right: Data after transformation \((x_1, x_2) \rightarrow \phi_1(x_1, x_2), \phi_2(x_1, x_2)\) are linearly separable (black line). Corresponding to black circle in left figure.

Problems that are not linearly separable in \(x\) might be linearly separable in \(\phi(x)\).
Data \((x_n, t_n), n = 1, \ldots, N, t_n \in \{0, 1\}\). Maximum likelihood to determine parameters \(w\):

\[
p(t_1, \ldots, t_N | w, x_1, \ldots, x_N) = \prod_n \sigma(w^T \phi_n)^{t_n}[1 - \sigma(w^T \phi_n)]^{1-t_n} = \prod_n y_n^{t_n}(1 - y_n)^{1-t_n}
\]

with \(\phi_n = \phi(x_n)\) and \(y_n = \sigma(w^T \phi_n)\) i.e.,

\[
E(w) = - \ln p = - \sum_n [t_n \ln y_n + (1 - t_n) \ln(1 - y_n)]
\]

NB: entropic error function for classification, rather than squared error.
Logistic regression

\[
\frac{\partial E(w)}{\partial w_i} = \sum_n (y_n - t_n) \phi_i(x_n)
\]

(Ex. 4.13) No closed form solution. \(^5\)

\(y_n - t_n\) with is 'error' in sample \(n\).

Overfitting risk when data is linearly separable: \(w \rightarrow \infty\) (i.e. \(\sigma \rightarrow\) step function).

\(^5\)Connection between this (Bishop) result and previous (HKP) results: \(y_n = \sigma(w^T \phi(x_n)) = p(y = 1|x_n) = 1.p(y = 1|x_n) + 0.p(y = 0|x_n) = \langle y \rangle_n = \langle y \rangle^\mu\).
Newton's method

One can also use Hessian information for optimization. As an example, consider a quadratic approximation to $E$ around $w_0$:

\[
E(w) = E(w_0) + b^T(w - w_0) + \frac{1}{2}(w - w_0)H(w - w_0)
\]

\[
b_i = \frac{\partial E(w_0)}{\partial w_i} \quad H_{ij} = \frac{\partial^2 E(w_0)}{\partial w_i \partial w_j}
\]

\[
\nabla E(w) = b + H(w - w_0)
\]

We can solve $\nabla E(w) = 0$ and obtain

\[
w = w_0 - H^{-1}\nabla E(w_0)
\]

This is called Newton's method.

Quadratic approximation is exact when $E$ is quadratic, so convergence in one step.

Quasi-Newton: Consider only diagonal of $H$. 
Iterative least squares

Minimize learning error by Newton-Raphson method

\[ \mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} - \mathbf{H}^{-1} \nabla E(\mathbf{w}) \]

with

\[ \mathbf{H}_{ij} = \frac{\partial^2 E}{\partial w_i \partial w_j} = \frac{\partial}{\partial w_j} \frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_j} \sum_n (y_n - t_n) \phi_i(\mathbf{x}_n) \]

\[ = \sum_n \phi_j(\mathbf{x}_n) y_n (1 - y_n) \phi_i(\mathbf{x}_n) = (\Phi^T \mathbf{R} \Phi)_{ij} \]

\[ \nabla_i E(\mathbf{w}) = \sum_n (y_n - t_n) \phi_i(\mathbf{x}_n) = (\Phi^T (\mathbf{y} - \mathbf{t}))_{ij} \]

with \( \Phi_{nj} = \phi_j(\mathbf{x}_n) \) and \( \mathbf{R}_{n,n'} = y_n (1 - y_n) \delta_{n,n'} \).

\( \mathbf{H}(\mathbf{w}) \) is positive definite for all \( \mathbf{w} \) thus \( E(\mathbf{w}) \) is convex, thus unique optimum (Ex. 4.15).

\[ \mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} - (\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T (\mathbf{y} - \mathbf{t}) \]
Network training

Regression: $t_n$ continue valued, $h_2(x) = x$ and one usually minimizes the squared error (one output)

\[
E(w) = \frac{1}{2} \sum_{n=1}^{N} (y(x_n, w) - t_n)^2
\]

\[
= - \log \prod_{n=1}^{N} \mathcal{N}(t_n | y(x_n, w), \beta^{-1}) + \ldots
\]

Classification: $t_n = 0, 1$, $h_2(x) = \sigma(x)$, $y(x_n, w)$ is probability to belong to class 1.

\[
E(w) = - \sum_{n=1}^{N} \{ t_n \log y(x_n, w) + (1 - t_n) \log (1 - y(x_n, w)) \}
\]

\[
= - \log \prod_{n=1}^{N} y(x_n, w)^{t_n} (1 - y(x_n, w))^{1 - t_n}
\]
Network training

More than two classes: consider network with $K$ outputs. $t_{nk} = 1$ if $x_n$ belongs to class $k$ and zero otherwise. $y_k(x_n, w)$ is the network output

$$E(w) = - \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log p_k(x_n, w)$$

$$p_k(x, w) = \frac{\exp(y_k(x, w))}{\sum_{k'=1}^{K} \exp(y_{k'}(x, w))}$$
$E$ is minimal when $\nabla E(w) = 0$, but not vice versa!

As a consequence, gradient based methods find a local minimum, not necessary the global minimum.
Gradient descent optimization

The simplest procedure to optimize $E$ is to start with a random $w$ and iterate

$$w^{τ+1} = w^τ - η∇E(w^τ)$$

This is called batch learning, where all training data are included in the computation of $∇E$.

Does this algorithm converge? Yes, if $ε$ is ”sufficiently small” and $E$ bounded from below.

Proof: Denote $Δw = -η∇E$.

$$E(w + Δw) ≈ E(w) + (Δw)^T∇E = E(w) - η \sum_i \left( \frac{∂E}{∂w_i} \right)^2 \leq E(w)$$

In each gradient descent step the value of $E$ is lowered. Since $E$ bounded from below, the procedure must converge asymptotically.
Convergence of gradient descent in a quadratic well

\[ E(w) = \frac{1}{2} \sum_i \lambda_i w_i^2 \]

\[ \Delta w_i = -\eta \frac{\partial E}{\partial w_i} = -\eta \lambda_i w_i \]

\[ w_i^{\text{new}} = w_i^{\text{old}} + \Delta w_i = (1 - \eta \lambda_i) w_i \]

Convergence when \( |1 - \eta \lambda_i| < 1 \). Oscillations when \( 1 - \eta \lambda_i < 0 \).

Optimal learning parameter depends on curvature of each dimension.
Learning with momentum

One solution is adding momentum term:

\[
\Delta w_{t+1} = -\eta \nabla E(w_t) + \alpha \Delta w_t \\
= -\eta \nabla E(w_t) + \alpha (-\eta \nabla E(w_{t-1}) + \alpha (-\eta \nabla E(w_{t-2}) + \ldots)) \\
= -\eta \sum_{k=0}^{t} \alpha^k \nabla E(w_{t-k})
\]

Consider two extremes:

**No oscillations** all derivative are equal:

\[
\Delta w_{t+1} \approx -\eta \nabla E \sum_{k=0}^{t} \alpha^k = -\frac{\eta}{1 - \alpha \partial w} \partial E
\]

results in acceleration
**Oscillations** all derivatives are equal but have opposite sign:

\[
\Delta w(t + 1) \approx -\eta \nabla E \sum_{k=0}^{t} (-\alpha)^k = -\eta \frac{\partial E}{1 + \alpha \partial w}
\]

results in deceleration
Newtons method

One can also use Hessian information for optimization. As an example, consider a quadratic approximation to $E$ around $w_0$:

$$
E(w) = E(w_0) + b^T(w - w_0) + \frac{1}{2}(w - w_0)H(w - w_0)
$$

$$
b_i = \frac{\partial E(w_0)}{\partial w_i} \quad H_{ij} = \frac{\partial^2 E(w_0)}{\partial w_i \partial w_j}
$$

$$
\nabla E(w) = b + H(w - w_0)
$$

We can solve $\nabla E(w) = 0$ and obtain

$$
w = w_0 - H^{-1}\nabla E(w_0)
$$

This is called Newtons method.

Quadratic approximation is exact when $E$ is quadratic, so convergence in one step.

Quasi-Newton: Consider only diagonal of $H$. 

Line search

Another solution is line optimisation:

\[ w_1 = w_0 + \lambda_0 d_0, \quad d_0 = -\nabla E(w_0) \]

\(\lambda_0 > 0\) is found by a one dimensional optimisation

\[ 0 = \frac{\partial}{\partial \lambda_0} E(w_0 + \lambda_0 d_0) = d_0 \cdot \nabla E(w_1) = d_0 \cdot d_1 \]

Therefore, subsequent search directions are orthogonal.
Conjugate gradient descent

We choose as new direction a combination of the gradient and the old direction

\[ d'_1 = -\nabla E(w_1) + \beta d_0 \]

Line optimisation \( w_2 = w_1 + \lambda_1 d'_1 \) yields \( \lambda_1 > 0 \) such that \( d'_1 \cdot \nabla E(w_2) = 0 \).

The direction \( d'_1 \) is found by demanding that \( \nabla E(w_2) \approx 0 \) also in the 'old' direction \( d_0 \):

\[ 0 = d_0 \cdot \nabla E(w_2) \approx d_0 \cdot (\nabla E(w_1) + \lambda_1 H(w_1)d'_1) \]

or

\[ d_0 H(w_1)d'_1 = 0 \]

The subsequent search directions \( d_0, d'_1 \) are said to be conjugate.
The conjugate directions can be computed without computing the Hessian matrix, for instance using the Polak-Ribiere rule:

\[ \beta = \frac{\left( \nabla E(w_1) - \nabla E(w_0) \right) \cdot \nabla E(w_1)}{\| \nabla E(w_0) \|^2} \]

For quadratic problems, it can be proven that this rule keeps the last \( n \) directions all mutually conjugate [Press et al., 1996]

\[ d_i^T H d_j = 0 \quad i, j = 1, \ldots, n \]

---

\( ^6 \)We need \( 0 = d_0^T H(w_1) d_1' \). We use \( \nabla E(w_0) \approx \nabla E(w_1) + (w_0 - w_1)^T H(w_1) = \nabla E(w_1) - \lambda_0 d_0^T H(w_1) \) and \( d_1' = -\nabla E(w_1) + \beta d_0 \). Then

\[ 0 = \lambda_0 d_0^T H(w_1) d_1' = \left( \nabla E(w_1) - \nabla E(w_0) \right) \cdot \left( -\nabla E(w_1) + \beta d_0 \right) = \left( \nabla E(w_1) - \nabla E(w_0) \right) \cdot \nabla E(w_1) + \beta \| \nabla E(\cdot) \|_2 \]

where in the last step we used that \( d_0 \cdot \nabla E(w_1) = 0 \).
**Stochastic gradient descent**

One can also consider on-line learning, where only one or a subset of training patterns is considered for computing $\nabla E$.

$$E(w) = \sum_n E_n(w) \quad w_{t+1} = w_t - \alpha_t \nabla E_n(w^\tau)$$

May be efficient for large data sets. This results in a stochastic dynamics in $w$ that can help to escape local minima.
Robbins Monro

Consider the problem to find $x$ such that

$$M(x) = a, \quad M(x) = \langle N(x, \xi) \rangle = \int d\xi p(\xi) N_i(x, \xi)$$

$x, a, M, N$ are vectors. $N_i(x, \xi)$ some non-linear function, $p(\xi)$ is a probability distribution and $a_i$ a constant.

Method of *stochastic approximation* originally due to Robbins and Monro 1951:
- Initialize $x_0$ random
- For $t = 0, \ldots$, Choose $\xi_t \sim p(\xi)$; Update $x_{t+1} = x_t + \alpha_t (a - N(x_t, \xi_t))$

If $M_i(x)$ convex and $x^*$ the unique solution, then one can prove that $\|x_t - x^*\|^2 \to 0$, provided that

$$\sum_{t=1}^{\infty} \alpha_t = \infty \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

For instance $\alpha_t = 1/t$. 
Stochastic gradient descent

Denote training error

\[ E(w) = \frac{1}{P} \sum_{\mu} E_{\mu}(w) \]

we wish to find solution of

\[ \nabla E(w) = \frac{1}{P} \sum_{\mu} \nabla E_{\mu}(w) = 0 \]

This is an instance of the Robbins-Monro problem with \( \xi = \mu = 1, \ldots, P \) and

\[ p(\mu) = \frac{1}{P} \quad a_i = 0 \quad N(w, \mu) = E_{\mu}(w) \]

The SGD method is
- Choose random a pattern \( \mu \in [1, \ldots, P] \)
- Update \( w_{t+1} = w_t - \eta_t \nabla E_{\mu}(w) \)

Extensions of SGD and comparisons see [Sohl-Dickstein et al., 2013].
Feed-forward Network functions

We extend the previous regression model with fixed basis functions

\[ y(x, w) = f \left( \sum_{j=1}^{M} w_j \phi_j(x) \right) \]

to a model where \( \phi_j \) is adaptive:

\[ \phi_j(x) = h(\sum_{i=0}^{D} w_{ji}^{(1)} x_i) \]
Feed-forward Network functions

In the case of $K$ outputs

$$y_k(x, w) = h_2 \left( \sum_{j=1}^{M} w_{k,j}^{(2)} h_1 \left( \sum_{i=0}^{D} w_{j,i}^{(1)} x_i \right) \right)$$

$h_2(x)$ is $\sigma(x)$ or $x$ depending on the problem. $h_1(x)$ is $\sigma(x)$ or $\tanh(x)$.

Left) Two layer architecture. Right) general feed-forward network with skip-layer connections.

If $h_1, h_2$ linear, the model is linear. If $M < D, K$ it computes principle components (Bishop section 12.4.2).
Feed-forward Network functions

Two layer NN with 3 'tanh' hidden units and linear output can approximate many functions. $x \in [-1, 1]$, 50 equally spaced points. From left to right: $f(x) = x^2, \sin(x), |x|, \Theta(x)$. Dashed lines are outputs of the 3 hidden units.

Two layer NN with two inputs and 2 'tanh' hidden units and sigmoid output for classification. Dashed lines are hidden unit activities.

Feed-forward neural networks have good approximation properties.
Weight space symmetries

For any solutions of the weights, there are many equivalent solutions due to symmetry:
- for any hidden unit $j$ with $\tanh$ activation function, change $w_{ji} \rightarrow -w_{ji}$ and $w_{kj} \rightarrow -w_{kj}$: $2^M$ solutions
- rename the hidden unit labels: $M!$ solutions

Thus a total of $M!2^M$ equivalent solutions, not only for $\tanh$ activation functions.
Error backpropagation

Error is sum of error per pattern

\[ E(w) = \sum_n E^n(w) \]

\[ E^n(w) = \frac{1}{2} \| y(x_n, w) - t_n \|^2 \]

\[ y_k(x, w) = h_2 \left( w_{k0} + \sum_{j=1}^{M} w_{kj} h_1 \left( w_{j0} + \sum_{i=1}^{D} w_{ji} x_i \right) \right) \]

\[ = h_2(a_k) \]

\[ a_k = w_{k0} + \sum_{j=1}^{M} w_{kj} h_1(a_j) = \sum_{j=0}^{M} w_{kj} h_1(a_j) \quad h_1(a_0) = 1 \]

\[ a_j = w_{j0} + \sum_{i=1}^{D} w_{ji} x_i = \sum_{i=0}^{D} w_{ji} x_i \quad x_0 = 1 \]

\( i \) labels inputs, \( j \) labels hiddens, \( k \) labels outputs.
5.3.1 Error backpropagation

We do each pattern separately, so we consider $E^n$

\[
y_k(x^n, w) = h_2(a^n_k) = h_2 \left( \sum_{j=0}^{M} w_{kj} h_1(a^n_j) \right) = h_2 \left( \sum_{j=0}^{M} w_{kj} \left( \sum_{i=0}^{D} w_{ji} x^n_i \right) \right)
\]

\[
\frac{\partial E^n}{\partial w_{kj}} = (y^n_k - t^n_k) \frac{\partial y^n_k}{\partial w_{kj}} = (y^n_k - t^n_k) h'_2(a^n_k) \frac{\partial a^n_k}{\partial w_{kj}} = (y^n_k - t^n_k) h'_2(a^n_k) h_1(a^n_j)
\]

\[
\delta^n_k = h'_1(a^n_j) \delta^n_k w_{kj}
\]

\[
\frac{\partial E^n}{\partial w_{ji}} = \sum_{k=1}^{K} (y^n_k - t^n_k) \frac{\partial y^n_k}{\partial w_{ji}} = \sum_{k=1}^{K} (y^n_k - t^n_k) h'_2(a^n_k) \frac{\partial a^n_k}{\partial w_{ji}}
\]

\[
= \sum_{k=1}^{K} \delta^n_k w_{kj} h'_1(a^n_j) \frac{\partial a^n_j}{\partial w_{ji}} = \sum_{k=1}^{K} \delta^n_k w_{kj} h'_1(a^n_j) x^n_i = \delta^n_j x^n_i
\]

\[
\delta^n_j = h'_1(a^n_j) \sum_{k=1}^{K} \delta^n_k w_{kj}
\]
The back propagation extends to arbitrary layers:

1. $z_i^n = x_i^n$ \textit{forward propagation} all activations $z_j^n = h_1(a_j^n)$ and $z_k^n = h_2(a_k^n)$, etc.

2. Compute the $\delta_k^n$ for the output units, and \textit{back-propagate} the $\delta$ to obtain $\delta_j^n$ each hidden unit $j$

3. $\partial E^n/\partial w_{kj} = \delta_k^n z_j^n$ and $\partial E^n/\partial w_{ji} = \delta_j^n z_i^n$

4. for batch mode, $\partial E/\partial w_{ji} = \sum_n \partial E^n/\partial w_{ji}$

$E$ is a function of $\mathcal{O}(|w|)$ variables. In general, the computation of $E$ requires $\mathcal{O}(|w|)$ operations. The computation of $\nabla E$ would thus require $\mathcal{O}(|w|^2)$ operations.

The backpropagation method allows to compute $\nabla E$ efficiently, in $\mathcal{O}(|w|)$ operations.
Complexity of neural network solution is controlled by number of hidden units

sum squared test error for different number of hidden units and different weight initializations. Error is also affected by local minima.
Part of the cause of local minima is the saturation of the sigmoid functions \( \tanh(\sum w_{ij}x_j) \). When \( w_{ij} \) becomes large, any change in its value hardly affects the output, implying \( \nabla_{ij}E = 0 \).

One can partly prevent this from happening by

- choosing \( \tanh \) instead of \( \sigma \) transfer functions and scaling of inputs and outputs with mean zero and standard deviation one

- proper initialisation of \( w_{ij} \) with mean zero and standard deviation of order \( 1/\sqrt{n_1} \), with \( n_1 \) the number of inputs to neuron \( i \).

- add regularizer such as \( \sum_i w_i^2 \) to cost keeps weights small

- dropouts, other transfer functions, adding noise, ....
**MLPs are universal approximators**

Consider $2^n$ binary patterns in $n$ dimensions and two classes:

$$x^\mu \to c^\mu = \pm 1, \quad x^\mu_i = \pm 1$$

Use $2^n$ hidden units, labeled $j = 0, \ldots, 2^n - 1$, $i$ labels input. Set

$$w_{ji} = b \quad \text{if } i\text{th digit in binary repr. of } j \text{ is 1}$$

$$w_{ji} = -b \quad \text{else}$$

<table>
<thead>
<tr>
<th>$j$</th>
<th>binary</th>
<th>$w_{j1}$</th>
<th>$w_{j2}$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$\sum_i w_{0i}x_i$</th>
<th>$w_{1i}x_i$</th>
<th>$w_{2i}x_i$</th>
<th>$w_{3i}x_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>-b</td>
<td>-b</td>
<td>-1</td>
<td>-1</td>
<td>2b</td>
<td>0</td>
<td>0</td>
<td>-2b</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>-b</td>
<td>b</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>2b</td>
<td>-2b</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>b</td>
<td>-b</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>-2b</td>
<td>2b</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>b</td>
<td>b</td>
<td>1</td>
<td>1</td>
<td>-2b</td>
<td>0</td>
<td>0</td>
<td>2b</td>
</tr>
</tbody>
</table>
MLPs are universal approximators

Use threshold of $(n - 1)b$ at each hidden unit. \( z_j = \Theta[\sum_i w_{ji}x_i - (n - 1)b] \). The remaining problem has \( p = 2^n \) patterns in \( 2^n \) dimensions and is linearly separable.

Define \( c = \text{sign}[\sum_{j=0}^{3} w_j z_j] \).

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( z_0 )</th>
<th>( z_1 )</th>
<th>( z_2 )</th>
<th>( z_3 )</th>
<th>( c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>\text{sign}[w_0]</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>\text{sign}[w_1]</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>\text{sign}[w_2]</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>\text{sign}[w_3]</td>
</tr>
</tbody>
</table>

The combination of linear summation and non-linear functions can create many different functions.

- The MLP with a single hidden layer can map any continuous function [Cybenko, 1989, Hornik et al., 1989]
Convolution networks

A special architecture for images using 4 ideas:
local connections: Connect not fully the bipartite graph between two layers.
weight sharing: use the same parameters to different neurons that detect same feature (feature layer)
(max) pooling: down sample each feature map (not adaptive) many layers: repeat many times.

Fukushima 1982, LeCunn 1990
Imagenet 2012 competition

Imagenet is an annual computer vision competition. Classify images in 1000 classes. 1.2 million training images, 50,000 validation images, and 150,000 testing images.

Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network’s input is 150,528-dimensional, and the number of neurons in the network’s remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

<table>
<thead>
<tr>
<th>Model</th>
<th>Top-1</th>
<th>Top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparse coding [2]</td>
<td>47.1%</td>
<td>28.2%</td>
</tr>
<tr>
<td>SIFT + FVs [24]</td>
<td>45.7%</td>
<td>25.7%</td>
</tr>
<tr>
<td>CNN</td>
<td>37.5%</td>
<td>17.0%</td>
</tr>
</tbody>
</table>

Table 1: Comparison of results on ILSVRC-2010 test set. In italics are best results achieved by others.
Interpreting images

A woman is throwing a frisbee in a park. A dog is standing on a hardwood floor. A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear. A group of people sitting on a boat in the water. A giraffe standing in a forest with trees in the background.

Vinyals et al. 2014
Deep learning papers

Fukushima Neocognitron 1982:

LeCunn Convolutional networks 1990:

Krizhevsky Imagenet 2012:


LeCunn, Bengio, Hinton. Deep learning 2015:
https://www.nature.com/articles/nature14539.pdf
Deep belief networks

[Hinton and Salakhutdinov, 2006]
**Experiments: MNIST data**

784 − 1000 − 500 − 250 − 30 autoencoder
60,000 training images and 10,000 test images

Top to bottom: test samples; DBN; logistic PCA with 30 components; PCA with 30 components. The average squared errors for the last three rows are 3.00, 8.01, and 13.87.
Experiments: MNIST data

Fig. 3. (A) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (B) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (b).

DBN yields better visualisation than PCA
Experiments: documents

Each newswire story was represented as a 2000 dimensional vector of common word counts.

Training:
- 2000-500-250-125-2 autoencoder
- 402,207 training documents, 402,207 test documents

Mining for Structure

Massive increase in both computational power and the amount of data available from web, video cameras, laboratory measurements.

- Develop statistical models that can discover underlying structure, cause, or statistical correlation from data in unsupervised or semi-supervised way.
- Multiple application domains.
Caption Generation with Visual Attention

A man riding a horse in a field.

Xu et.al., ICML 2015
Challenges - I

Very different input representations

- Images – real-valued, dense
- Text – discrete, sparse

Difficult to learn cross-modal features from low-level representations.
Multimodal DBM

\[ h^3 \]

\[ h^2 \]

\[ h^1 \]

Gaussian model

Dense, real-valued image features

\[ V_{\text{image}} \]

Replicated Softmax

Word counts

\[ V_{\text{text}} \]

(Srivastava & Salakhutdinov, NIPS 2012, U MLP 2014)
Text Generated from Images

Given

1. Dog, cat, pet, kitten, puppy, ginger, tongue, kitty, dogs, furry
2. Sea, france, boat, mer, beach, river, bretagne, plage, brittany
3. Portrait, child, kid, ritratto, kids, children, boy, cute, boys, italy

Generated

1. Insect, butterfly, insects, bug, butterflies, lepidoptera
2. Graffiti, streetart, stencil, sticker, urbanart, graff, sanfrancisco
3. Canada, nature, sunrise, ontario, fog, mist, bc, morning
Text Generated from Images

Given

Generated

portrait, women, army, soldier, mother, postcard, soldiers

obama, barackobama, election, politics, president, hope, change, sanfrancisco, convention, rally

water, glass, beer, bottle, drink, wine, bubbles, splash, drops, drop
Human-level control through deep reinforcement learning

http://www.nature.com/nature/journal/v518/n7540/full/nature14236.html#videos
Chapter 2

Probability distributions

- Ch 2.1 Bayesian inference with discrete variables (Bernouilli distribution)
- Ch 2.3 Gaussian distribution
  - Bayesian inference
  - Some properties of high dimensional Gaussians
Binary variables / Bernoulli distribution

$x \in \{0, 1\}$

\[
p(x = 1|\mu) = \mu, \\
p(x = 0|\mu) = 1 - \mu
\]

Bernoulli distribution:

\[
\text{Bern}(x|\mu) = \mu^x (1 - \mu)^{1-x}
\]

Mean and variance:

\[
\mathbb{E}[x] = \mu \\
\text{var}[x] = \mu(1 - \mu)
\]
Binary variables / Bernoulli distribution

Data set (i.i.d) \( D = \{ x_1, \ldots, x_n \} \), with \( x_i \in \{ 0, 1 \} \).

Likelihood:
\[
p(D|\mu) = \prod_{n} p(x_n|\mu) = \mu^m (1 - \mu)^{N-m}
\]
where \( m = \sum_{n} x_n \), the total number of \( x_n = 1 \).

Log likelihood
\[
\ln p(D|\mu) = \sum_{n} \ln p(x_n|\mu) = \sum_{n} x_n \ln \mu + (1 - x_n) \ln(1 - \mu)
\]
\[
= m \ln \mu + (N - m) \ln(1 - \mu)
\]

Maximization w.r.t \( \mu \) gives maximum likelihood solution:

\[
\mu_{ML} = \frac{m}{N}
\]
The beta distribution

Distribution for parameters $\mu$. Conjugate prior for Bayesian treatment for problem.

$$p(\mu|a, b) = \text{Beta}(\mu|a, b) = \frac{\Gamma(a + b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1 - \mu)^{b-1} \quad 0 \leq \mu \leq 1$$

Is called conjugate because prior $p(\mu|a, b)$ has same form as likelihood.

Normalisation (Ex. 2.5)

$$\int_{0}^{1} \mu^{a-1}(1 - \mu)^{b-1} = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a + b)}$$

Mean and variance (Ex. 2.6)

$$\mathbb{E}[\mu] = \frac{a}{a + b}$$

$$\text{var}[\mu] = \frac{ab}{(a + b)^2(a + b + 1)}$$
The beta distribution

- $a = 0.1$
- $b = 0.1$

- $a = 1$
- $b = 1$

- $a = 2$
- $b = 3$

- $a = 8$
- $b = 4$
Bayesian inference with binary variables

Prior:

\[ p(\mu) = \text{Beta}(\mu|a, b) \propto \mu^{a-1}(1 - \mu)^{b-1} \]

Likelihood – Data set (i.i.d) \( D = \{x_1, \ldots, x_N\} \), with \( x_i \in \{0, 1\} \).
Assume \( m \) ones and \( l \) zeros, \((m + l = N)\)

\[
p(D|\mu) = \prod_n p(x_n|\mu) = \prod_n \mu^{x_n}(1 - \mu)^{1-x_n} = \mu^m(1-\mu)^l
\]

Posterior

\[
p(\mu|D) \propto p(D|\mu)p(\mu) = \mu^m(1-\mu)^l \times \mu^{a-1}(1 - \mu)^{b-1} = \mu^{m+a-1}(1 - \mu)^{l+b-1} \propto \text{Beta}(\mu|a + m, b + l)
\]
Bayesian inference with binary variables

Interpretation: Hyperparameters $a$ and $b$ effective number of ones and zeros.

Data: increments of these parameters.

Conjugacy:
(1) prior has the same form as likelihood function.
(2) this form is preserved in the product (the posterior)
(3) sequential learning

Prior $p(\mu|a = 2, b = 2)$, likelihood $N = m = 1(l = 0)$ $p(D|\mu) = \mu$, posterior $p(\mu|a = 3, b = 2)$. 
Bayesian inference with binary variables

Prediction of next data point given data $D$:

$$p(x = 1|D) = \int_0^1 p(x = 1|\mu)p(\mu|D)d\mu = \int_0^1 \mu p(\mu|D)d\mu = \mathbb{E}[\mu|D]$$

with posterior $p(\mu|D = \text{Beta}(\mu|a + m, b + l)$, and $\mathbb{E}[\mu|a, b] = a/(a + b)$ we find

$$p(x = 1|D) = \frac{m + a}{m + a + l + b}$$

Toss a coin once and shows 'head'. Thus, $m = 1, l = 0$.

- the frequentist answer is $p(x = 1|D) = 1$.
- the Bayesian answer assuming 'flat' prior $(a = b = 1)$ is $p(x = 1|D) = \frac{2}{3}$. 
Multinomial variables

Alternative representation: \(x \in \{v_1, v_2\}\), parameter vector: \(\mu = (\mu_1, \mu_2)\), where \(\mu_1 + \mu_2 = 1\).

\[
p(x = v_k | \mu) = \mu_k
\]

In fancy notation:

\[
p(x | \mu) = \prod_k \mu_k^\delta_{xv_k}
\]

Generalizes to multinomial variables: \(x \in \{v_1, \ldots, v_K\}\),

\[
\mu = (\mu_1, \ldots, \mu_K) \quad \sum_k \mu_k = 1
\]
Multinomial variables: Maximum likelihood

Likelihood:

\[
p(D|\mu) = \prod_{n} p(x_n|\mu) = \prod_{n} \prod_{k} \mu_{v_k}^{\delta_{x_n v_k}} = \prod_{k} \mu_{v_k}^{\sum_{n} \delta_{x_n v_k}} = \prod_{k} \mu_{v_k}^{m_k}
\]

with \( m_k = \sum_{n} \delta_{x_n v_k} \), the total number of datapoints with value \( v_k \). Log likelihood

\[
\ln p(D|\mu) = \sum_{k} m_k \ln \mu_k
\]

Maximize with constraint \( \sum_{k} \mu_k = 1 \) using Lagrange multipliers.
Dirichlet distribution

\[ \text{Dir}(\mu|\alpha) \propto \prod_k \mu_k^{\alpha_k} \]

Probability distribution on the simplex:

\[ S^K = \{ (\mu_1, \ldots, \mu_K) | 0 \leq \mu_k \leq 1, \sum_{k=1}^K \mu_k = 1 \} \]

Bayesian inference: Prior \( \text{Dir}(\mu|\alpha) + \text{data counts } m \)

\[ \rightarrow \text{Posterior } \text{Dir}(\mu|\alpha + m) \]

Parameters \( \alpha \): ‘pseudocounts’.
Dirichlet distribution

Left: $\alpha_k = 0.1$, Middle: $\alpha_k = 1$, Right: $\alpha_k = 10$
Gaussian distribution

\[ \mathcal{N}(x | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left\{ -\frac{(x - \mu)^2}{2\sigma^2} \right\} \]

Specified by \( \mu \) and \( \sigma^2 \)

In \( d \) dimensions

\[ \mathcal{N}(x | \mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp \left( -\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) \right) \]
Central limit theorem

Sum of large number of independent random variables is approximately Gaussian distributed.

Let $X_i$ be random variables with mean $\mu$ and variance $\sigma^2$.

The sum of $n$ variables is $S_n = X_1 + \ldots + X_n$. Now if $n \to \infty$

Law of large numbers: The mean of the sum $Y_n = \frac{S_n}{n}$ converges to $\mu$

Central limit theorem: Distribution of

\[ Z_n = \frac{S_n - n\mu}{\sigma \sqrt{n}} \]

converges to Gaussian $\mathcal{N}(Z_n|0, 1)$
Bayesian inference for the Gaussian

Aim: inference of unknown parameter $\mu$. Assume $\sigma$ given.

Likelihood of $\mu$ with one data point:

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

Likelihood of $\mu$ with the data set:

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

$$= \exp\left(-\frac{N\mu^2}{2\sigma^2} + \frac{\mu}{\sigma^2} \sum_n x_n + \text{const.}\right) = \exp\left(-\frac{N}{2\sigma^2}\mu^2 + \frac{N\bar{x}}{\sigma^2}\mu + \text{const.}\right)$$

$$= \exp\left(-\frac{N}{2\sigma^2}(\mu - \bar{x})^2 + \text{const.}\right) \quad \text{with} \quad \bar{x} = \frac{1}{N} \sum_n x_n$$

ML: $\mu = \bar{x}$
Bayesian inference for the Gaussian

Likelihood:

\[ p(\text{Data}|\mu) = \exp\left(-\frac{N}{2\sigma^2}(\mu - \bar{x})^2 + \text{const.}\right) \]

Prior:

\[ p(\mu) = \mathcal{N}(\mu|\mu_0, \sigma_0) = \frac{1}{\sqrt{2\pi}\sigma_0} \exp\left(-\frac{1}{2\sigma_0^2}(\mu - \mu_0)^2\right) \]

\(\mu_0, \sigma_0\) hyperparameters. Large \(\sigma_0\) = large prior uncertainty in \(\mu\).

\[ p(\mu|\text{Data}) \propto p(\text{Data}|\mu)p(\mu) \]
Posterior is proportional to the product of two Gaussian potentials.

\[ p(\text{Data}|\mu) \propto \mathcal{N}(\mu|\bar{x}, \frac{1}{N}\sigma^2) \]

\[ p(\mu) = \mathcal{N}(\mu|\mu_0, \sigma_0^2) \]

\[ p(\mu|\text{Data}) = \mathcal{N}(\mu|\mu_N, \sigma_N^2) \]

with (Ex. 2.38)

\[ \mu_N = \frac{N\sigma_0^2\bar{x} + \sigma^2\mu_0}{N\sigma_0^2 + \sigma^2} = \frac{N\bar{x} + N_0\mu_0}{N + N_0} \]

\[ \frac{1}{\sigma_N^2} = \frac{N + N_0}{\sigma^2} \]

Interpretation: \( \mu_0 \) mean of pseudodata; \( N_0 = \frac{\sigma^2}{\sigma_0^2} \) is effective number of pseudocounts
For $N \to \infty$: $\mu_N \to \bar{x}, \sigma^2_N \to 0$

i.e., posterior distribution is a peak around ML solution
so Bayesian inference and ML coincides.
Symmetric matrices:

\[ A_{ij} = A_{ji}, \quad A^T = A \]

Inverse of a matrix is a matrix \( A^{-1} \) such that \( A^{-1} A = AA^{-1} = I \), where \( I \) is the identity matrix.

\( A^{-1} \) is also symmetric:

\[ I = I^T = (A^{-1} A)^T = A^T (A^{-1})^T = A (A^{-1})^T \]

Thus, \((A^{-1})^T = A^{-1}\).
Eigenvalues

A symmetric real-valued \( d \times d \) matrix has \( d \) real eigenvalues \( \lambda_k \) and \( d \) eigenvectors \( u_k \):

\[
A u_k = \lambda_k u_k, \quad k = 1, \ldots, d
\]

or

\[
(A - \lambda_k I)u_k = 0
\]

Solution of this equation for non-zero \( u_k \) requires \( \lambda_k \) to satisfy the characteristic equation:

\[
\det(A - \lambda I) = 0
\]

This is a polynomial equation in \( \lambda \) of degree \( d \) and has thus \( d \) solutions \( \lambda_1, \ldots, \lambda_d \).

\(^7\)In general, the solutions are complex. It can be shown that with symmetric matrices, the solutions are in fact real.
Eigenvectors

Consider two different eigenvectors $k$ and $j$. Multiply the $k$-th eigenvalue equation by $u_j$ from the left:

$$u_j^T A u_k = \lambda_k u_j^T u_k$$

Multiply the $j$-th eigenvalue equation by $u_k$ from the left:

$$u_k^T A u_j = \lambda_j u_k^T u_j = \lambda_j u_j^T u_k$$

Subtract

$$(\lambda_k - \lambda_j) u_j^T u_k = 0$$

Thus, eigenvectors with different eigenvalues are orthogonal.
If $\lambda_k = \lambda_j$ then any linear combination is also an eigenvector:

$$A(\alpha u_k + \beta u_j) = \lambda_k (\alpha u_k + \beta u_j)$$

This can be used to choose eigenvectors with identical eigenvalues orthogonal.

If $u_k$ is an eigenvector of $A$, then $\alpha u_k$ is also an eigenvector of $A$. Thus, we can make all eigenvectors the same length one: $u_k^T u_k = 1$.

In summary,

$$u_j^T u_k = \delta_{jk}$$

with $\delta_{jk}$ the Kronecker delta, is equal to 1 if $j=k$ and zero otherwise.

The eigenvectors span the $d$-dimensional space as an orthonormal basis.
Orthogonal matrices

Write \( U = (u_1, \ldots, u_d) \).

\( U \) is an orthogonal matrix \(^8\), i.e.

\[ U^T U = I \]

For orthogonal matrices,

\[ U^T U U^{-1} = U^{-1} = U^T \]

So \( UU^T = 1 \), i.e. the transposed is orthogonal as well (note that \( U \) is in general not symmetric).

Furthermore,

\[ \det(UU^T) = 1 \Rightarrow \det(U) \det(U^T) = 1 \]

\[ \Rightarrow \det(U) = \pm 1 \]

Orthogonal matrices implement rigid rotations, i.e. length and angle preserving.

\[ \bar{x}_1 = Ux_1 \quad \bar{x}_2 = Ux_2 \]

\(^8\) 

\[ U_{ij} = (u_j)_i, \quad (U^T U)_{ij} = \sum_k (U^T)_{ik} U_{kj} = \sum_k U_{ki} U_{kj} = \sum_k (u_i)_k (u_j)_k = u^T_i \cdot u_j = \delta_{ij} \]
then

$$\bar{x}_1^T \bar{x}_2 = x_1^T U^T U x_2 = x_1^T x_2$$
Diagonalization

The eigenvector equation can be written as

\[ AU = A(u_1, \ldots, u_d) = (Au_1, \ldots, Au_d) = (\lambda_1 u_1, \ldots, \lambda_d u_d) \]

\[ = (u_1, \ldots, u_d) \begin{pmatrix} \lambda_1 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & \lambda_d \end{pmatrix} \]

\[ = U\Lambda \]

By right-multiplying by \( U^T \) we obtain the important result

\[ A = U\Lambda U^T \]

which can also be written as 'expansion in eigenvectors'

\[ A = \sum_{k=1}^{d} \lambda_k u_k u_k^T \]
Applications

\[ A = U \Lambda U^T \Rightarrow A^2 = U \Lambda U^T U \Lambda U^T = U \Lambda^2 U^T \]

\[ A^n = U \Lambda^n U^T \quad A^{-n} = U \Lambda^{-n} U^T \]

Determinant is product of eigenvalues:

\[ \det(A) = \det(U \Lambda U^T) = \det(U) \det(\Lambda) \det(U^T) = \prod_k \lambda_k \]
Appendix C

Basis transformation

We can represent an arbitrary vector \( \mathbf{x} \) in \( d \) dimensions on a new basis \( U \) as

\[
\mathbf{x} = U U^T \mathbf{x} = \sum_{k=1}^{d} \mathbf{u}_k (\mathbf{u}_k^T \mathbf{x})
\]

The numbers \( \bar{x}_k = (\mathbf{u}_k^T \mathbf{x}) \) are the components of \( \mathbf{x} \) on the basis \( \mathbf{u}_k, k = 1, \ldots, d \), i.e. on the new basis, the vector has components \((U^T \mathbf{x})\). If the matrix \( A \) is the representation of a linear transformation on the old basis, the matrix with components

\[
A' = U^T A U
\]

is the representation on the new basis.

For instance if \( \mathbf{y} = A \mathbf{x} \), \( \bar{x} = U^T \mathbf{x} \), \( \bar{y} = U^T \mathbf{y} \), then

\[
A' \bar{x} = U^T A U U^T \mathbf{x} = U^T A \mathbf{x} = U^T \mathbf{y} = \bar{y}
\]

So a matrix is diagonal on a basis of its eigenvectors:

\[
A' = U^T A U = U^T U \Lambda U^T U = \Lambda
\]
Multivariate Gaussian

In $d$ dimensions

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

$$
\mu = \langle \mathbf{x} \rangle = \int \mathbf{x} \mathcal{N}(\mathbf{x}|\mu, \Sigma) \, d\mathbf{x}
$$

$$
\Sigma = \langle (\mathbf{x} - \mu)(\mathbf{x} - \mu)^T \rangle = \int (\mathbf{x} - \mu)^T (\mathbf{x} - \mu) \mathcal{N}(\mathbf{x}|\mu, \Sigma) \, d\mathbf{x}
$$

We can also write this in component notation:

$$
\mu_i = \langle x_i \rangle = \int x_i \mathcal{N}(\mathbf{x}|\mu, \Sigma) \, d\mathbf{x}
$$

$$
\Sigma_{ij} = \langle (x_i - \mu_i)(x_j - \mu_j) \rangle = \int (x_i - \mu_i)(x_j - \mu_j) \mathcal{N}(\mathbf{x}|\mu, \Sigma) \, d\mathbf{x}
$$
Multivariate Gaussian

Spectral (eigenvalue) representation of $\Sigma$:

$$\Sigma = U \Lambda U^T = \sum_k \lambda_k u_k u_k^T$$

$$\Sigma^{-1} = U \Lambda^{-1} U^T = \sum_k \lambda_k^{-1} u_k u_k^T$$

$$(x - \mu)^T U^T U \Sigma^{-1} U^T U (x - \mu) = y^T \Lambda^{-1} y$$
Multivariate Gaussian

Explain the normalization: use transformation $y = U(x - \mu) = Uz$

\[
\int \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right) \, dx
= \int \exp \left( -\frac{1}{2} z^T \Sigma^{-1} z \right) \, dz = \int |\det \left( \frac{\partial z}{\partial y} \right)| \exp \left( -\frac{1}{2} y^T \Lambda^{-1} y \right) \, dy
= \int \exp \left( -\frac{1}{2} \lambda_1 y_1^2 \right) \, dy_1 \ldots \int \exp \left( -\frac{1}{2} \lambda_d y_d^2 \right) \, dy_d
= \sqrt{2\pi \lambda_1} \ldots \sqrt{2\pi \lambda_d} = (2\pi)^{d/2} \left( \prod_{i} \lambda_i \right)^{1/2} = (2\pi)^{d/2} \det(\Sigma)^{1/2}
\]

The multivariate Gaussian becomes a product of independent factors on the basis of eigenvectors.
Multivariate Gaussian

Compute the expectation value: use shift-transformation $z = x - \mu$

Take $Z$ as normalisation constant.

Use symmetry $f(z) = -f(-z) \Rightarrow \int f(z)dz = 0$

$$\mathbb{E} [x] = \frac{1}{Z} \int \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right) x dx$$

$$= \frac{1}{Z} \int \exp \left( -\frac{1}{2} z^T \Sigma^{-1} z \right) (z + \mu) dz$$

$$= \mu$$
Multivariate Gaussian

Second order moment: First, shift by $z = x - \mu$

\[
\mathbb{E}[xx^T] = \int \mathcal{N}(x | \mu, \Sigma) xx^T dx
\]

\[
= \int \mathcal{N}(z | 0, \Sigma)(z + \mu)(z + \mu)^T dz
\]

\[
= \int \mathcal{N}(z | 0, \Sigma)(zz^T + z\mu^T + \mu z^T + \mu\mu^T) dz
\]

\[
= \int \mathcal{N}(z | 0, \Sigma)zz^T dz + \mu\mu^T
\]
Multivariate Gaussian

Now use transformation $y = Uz$, and use $\Sigma = U^T \Lambda U$

$$\int \mathcal{N}(z|0, \Sigma)zz^T dz = \int \mathcal{N}(y|0, \Lambda)yy^T U dU$$

$$= U^T \int \mathcal{N}(y|0, \Lambda)yy^T dU$$

Component-wise computation shows $\int \mathcal{N}(y|0, \Lambda)yy^T dU = \Lambda$:

$$i \neq j \quad \rightarrow \quad \int \mathcal{N}(y|0, \Lambda)y_iy_j dU = \int \mathcal{N}(y_i|0, \lambda_i)y_i dy_i \int \mathcal{N}(y_j|0, \lambda_j)y_j dy_j = 0$$

$$i = j \quad \rightarrow \quad \int \mathcal{N}(y|0, \Lambda)y_i^2 dU = \int \mathcal{N}(y_i|0, \lambda_i)y_i^2 dy_i = \lambda_i$$

So

$$\int \mathcal{N}(z|0, \Sigma)zz^T dz = U^T \Lambda U = \Sigma$$
Multivariate Gaussian

So, second moment is

$$\mathbb{E}[x x^T] = \Sigma + \mu \mu^T$$

Covariance

$$\text{cov}[x] = \mathbb{E}[x x^T] - \mathbb{E}[x] \mathbb{E}[x]^T = \Sigma$$
Multivariate Gaussian

Gaussian covariance has $d(d + 1)$ parameters, mean has $d$ parameters.

Number of parameters quadratic in $d$ which may be too large for high dimensional applications.

Common simplifications: $\Sigma_{ij} = \Sigma_{ii}\delta_{ij}$ ($2d$ parameters) or $\Sigma_{ij} = \sigma^2\delta_{ij}$ ($d + 1$ parameters).
Marginal and conditional Gaussians

Marginal and conditional of Gaussians are also Gaussian

1. (Proof next slides)

\[ p(\mathbf{x}_a | \mathbf{x}_b) = \mathcal{N}(\mathbf{x}_a | \mu_{a|b}, \Sigma_{a|b}) \]

\[ \Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab} \Sigma_{bb}^{-1} \Sigma_{ba} \quad \mu_{a|b} = \mu_a + \Sigma_{ab} \Sigma_{bb}^{-1} (\mathbf{x}_b - \mu_b) \]

2. (Proof a bit lengthy, skipped)

\[ p(\mathbf{x}_a) = \mathcal{N}(\mathbf{x}_a | \mu_a, \Sigma_{aa}) \]
Conditional of Gaussian is Gaussian

Exponent in Gaussian $\mathcal{N}(\mathbf{x} | \mu, \Sigma)$: quadratic form

$$-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu) = -\frac{1}{2} \mathbf{x}^T \Sigma^{-1} \mathbf{x} + \mathbf{x}^T \Sigma^{-1} \mu + c = -\frac{1}{2} \mathbf{x}^T K \mathbf{x} + \mathbf{x}^T K \mu + c$$

Precision matrix $K = \Sigma^{-1}$.

Write $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$. We wish to compute the conditional

$$p(\mathbf{x}_a | \mathbf{x}_b) = \frac{p(\mathbf{x}_a, \mathbf{x}_b)}{p(\mathbf{x}_b)} \propto p(\mathbf{x}_a, \mathbf{x}_b)$$

Exponent of conditional: collect all terms with $\mathbf{x}_a$, ignore constants, regard $\mathbf{x}_b$ as constant, and write in quadratic form as above

$$-\frac{1}{2} \mathbf{x}_a^T K \mathbf{x}_a + \mathbf{x}_a^T K \mu = -\frac{1}{2} \mathbf{x}_a^T K_{aa} \mathbf{x}_a + \mathbf{x}_a^T K_{aa} \mu_a - \mathbf{x}_a^T K_{ab}(\mathbf{x}_b - \mu_b)$$

$$= -\frac{1}{2} \mathbf{x}_a^T K_{aa} \mathbf{x}_a + \mathbf{x}_a^T K_{aa} (\mu_a - K_{aa}^{-1} K_{ab} (\mathbf{x}_b - \mu_b)) = -\frac{1}{2} \mathbf{x}_a^T \Sigma_{a|b}^{-1} \mathbf{x}_a + \mathbf{x}_a^T \Sigma_{a|b}^{-1} \mu_{a|b}$$

$$= -\frac{1}{2} (\mathbf{x}_a - \mu_{a|b})^T \Sigma_{a|b}^{-1} (\mathbf{x}_a - \mu_{a|b}) \quad \Sigma_{a|b} = K_{aa}^{-1} \quad \mu_{a|b} = \mu_a - K_{aa}^{-1} K_{ab} (\mathbf{x}_b - \mu_b)$$
Some matrix identities

We now need to relate $K_{aa}^{-1}, K_{ab}$ to the components $\Sigma$.

$$
\begin{pmatrix}
A & B \\
C & D
\end{pmatrix}^{-1} = \begin{pmatrix}
M & -MBD^{-1} \\
-D^{-1}CM & D^{-1} + D^{-1}CMBD^{-1}
\end{pmatrix}
$$

with $M = (A - BD^{-1}C)^{-1}$.

$$
\begin{pmatrix}
K_{aa} & K_{ab} \\
K_{ba} & K_{bb}
\end{pmatrix} = \begin{pmatrix}
\Sigma_{aa} & \Sigma_{ab} \\
\Sigma_{ba} & \Sigma_{bb}
\end{pmatrix}^{-1} = \begin{pmatrix}
M & -M\Sigma_{ab}\Sigma_{bb}^{-1} \\
-\Sigma_{bb}^{-1}\Sigma_{ba}M & \Sigma_{bb}^{-1} + \Sigma_{bb}^{-1}\Sigma_{ba}M\Sigma_{ab}\Sigma_{bb}^{-1}
\end{pmatrix}
$$

with $M = (\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba})^{-1}$. Thus, $K_{aa} = M$ and

$$
\Sigma_{a|b} = K_{aa}^{-1} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}
$$

$$
\mu_{a|b} = \mu_a - K_{aa}^{-1}K_{ab}(\mathbf{x}_b - \mu_b) = \mu_a + M^{-1}M\Sigma_{ab}\Sigma_{bb}^{-1}(\mathbf{x}_b - \mu_b)
$$

$$
= \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(\mathbf{x}_b - \mu_b)
$$
Bayes’ theorem for linear Gaussian model

Given marginal Gaussian on \( x \) and linear relation \( y = Ax + b + \xi \):

\[
p(x) = \mathcal{N}(x|\mu, \Lambda^{-1})
\]
\[
p(y|x) = \mathcal{N}(y|Ax + b, L^{-1})
\]

Then (see next slide):

\[
p(y) = \mathcal{N}(y|A\mu + b, L^{-1} + A\Lambda^{-1}A^T)
\]
\[
p(x|y) = \mathcal{N}(x|\Sigma\{A^T L (y - b) + \Lambda\mu\}, \Sigma)
\]
\[
\Sigma = (\Lambda + A^T L A)^{-1}
\]

We will use these relations for Bayesian linear regression in section 3.3.
Details computation $p(y)$

\[
p(x) = \mathcal{N}(x|\mu, \Lambda^{-1}) \quad \mathbb{E}x = \mu \quad \forall x = \Lambda^{-1}
\]

\[
p(y|x) = \mathcal{N}(y|Ax + b, L^{-1})
\]

We write $y = Ax + b + \epsilon$ with $\mathbb{E}\epsilon = 0, \forall \epsilon = L^{-1}$.

$x, y$ is jointly Gaussian (product of Gaussians). $y$ is Gaussian (marginal of Gaussian).

\[
\begin{align*}
\mathbb{E}y &= \mathbb{E}(Ax + b + \epsilon) = A\mu + b \\
\forall y &= \forall (ax + b + \epsilon) = \forall Ax + \forall \epsilon = \forall Ax + L^{-1} \\
\forall Ax &= \mathbb{E}(Ax - A\mu)(Ax - A\mu)^T = A\mathbb{E}(x - \mu)(x - \mu)^T A^T = A\Lambda^{-1}A^T
\end{align*}
\]

Thus,

\[
p(y) = \mathcal{N} \left( y|A\mu + b, A\Lambda^{-1}A^T + L^{-1} \right)
\]
Details computation \( p(x|y) \)

Write all relevant terms that occur in exponential of the joint Gaussian \( p(x, y) \):

\[
-\frac{1}{2}(x - \mu)^T \Lambda^{-1}(x - \mu) - \frac{1}{2}(y - Ax - b)^T L^{-1}(y - Ax - b)
\]

Collect all quadratic and linear terms in \( x \):

\[
-\frac{1}{2}x^T \left( \Lambda^{-1} + A^T L^{-1} A \right) x + x^T \left( \Lambda^{-1} \mu + A^T L^{-1}(y - b) \right)
\]

Define \( \Sigma^{-1} = \Lambda^{-1} + A^T L^{-1} A, \Sigma^{-1} m = \Lambda^{-1} \mu + A^T L^{-1}(y - b) \), then

\[
-\frac{1}{2}x^T \Sigma^{-1} x + x^T \Sigma^{-1} m \propto -\frac{1}{2}(x - m)^T \Sigma^{-1}(x - m)
\]

Thus

\[
p(x|y) = \mathcal{N} \left( x | \Sigma \left( \Lambda^{-1} \mu + A^T L^{-1}(y - b) \right), \Sigma \right)
\]
Chapter 3

Linear Models for Regression

Regression: . . . ?
Chapter 3

Linear Models for Regression

Regression: predicting the value of *continuous* target/output variables given values of the input variables.

In other words: given a training set of input/output pairs, constructing a function that maps input values to continuous output values.
**Linear Basis Function Models**

Linear regression:

\[ y(x, w) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = w_0 + \sum_{j=1}^{D} w_j x_j \]

Linear in parameters \( w \), and “almost” linear (affine) in input \( x \).

Q: how can we generalize this to models that need not be linear in input \( x \)?
Linear Basis Function Models

Linear regression:

\[ y(x, w) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = w_0 + \sum_{j=1}^{D} w_j x_j \]

Linear in parameters \( w \), and “almost” linear (affine) in input \( x \).

Q: how can we generalize this to models that need not be linear in input \( x \)? A:

\[ y(x, w) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(x) \]

where \( \phi_1(x), \ldots, \phi_{M-1}(x) \) are basis functions or feature functions. Defining \( \phi_0(x) = 1 \), we can write it more compactly:

\[ y(x, w) = \sum_{j=0}^{M-1} w_j \phi_j(x) = w^T \phi(x) \]

Note: still linear in parameters \( w \)! Examples of basis functions?
Linear Basis Function Models

Gaussians: $\phi_j(x) = \exp(-\beta(x - \mu_j)^2)$
Polynomials: $\phi_j(x) = x^j$
Sigmoids: $\phi_j(x) = \sigma(\beta(x - \mu_j))$
Fourier: $\phi_j(x) = \exp(i j x)$
Wavelets: ...
Adding some noise

Let’s make some noise. . . and assume the following model:

\[ t = y(x, w) + \epsilon, \quad y(x, w) = w^T \phi(x) \]

where we added Gaussian noise \( \epsilon \sim \mathcal{N}(0, \beta^{-1}) \).

Q: what is the distribution of \( t \), given \( x, w, \beta \)?
Adding some noise

Let’s make some noise. . . and assume the following model:

\[ t = y(x, w) + \epsilon, \quad y(x, w) = w^T \phi(x) \]

where we added Gaussian noise \( \epsilon \sim \mathcal{N}(0, \beta^{-1}) \).

Q: what is the distribution of \( t \), given \( x, w, \beta \)?

\[ p(t|x, w, \beta) = \mathcal{N}(t|y(x, w), \beta^{-1}) \]

Q: what is the conditional mean \( \mathbb{E}(t|x, w, \beta) \)?
Adding some noise

Let’s make some noise. . . and assume the following model:

\[ t = y(x, w) + \epsilon, \quad y(x, w) = w^T \phi(x) \]

where we added Gaussian noise \( \epsilon \sim \mathcal{N}(0, \beta^{-1}) \).

Q: what is the distribution of \( t \), given \( x, w, \beta \)?

\[ p(t|x, w, \beta) = \mathcal{N}(t|y(x, w), \beta^{-1}) \]

Q: what is the conditional mean \( \mathbb{E}(t|x, w, \beta) \)?

\[ \mathbb{E}(t|x, w, \beta) = \int tp(t|x) \, dt = y(x, w) \]
Maximum likelihood: least squares

Q: What is the likelihood of a data set \( \{(\mathbf{x}_n, t_n)\}_{n=1}^{N} \)?
Maximum likelihood: least squares

Q: What is the likelihood of a data set \( \{(x_n, t_n)\}_{n=1}^{N} \)?

\[
p(t|X, w, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n | w^T \phi(x_n), \beta^{-1})
\]

Q: How does the log-likelihood depend on the parameters \( w \)?
Maximum likelihood: least squares

Q: What is the likelihood of a data set \( \{(x_i, t_i)\}_{n=1}^N \)?

\[
p(t | X, w, \beta) = \prod_{n=1}^N \mathcal{N}(t_n | w^T \phi(x_n), \beta^{-1})
\]

Q: How does the log-likelihood depend on the parameters \( w \)?

\[
\ln p(t | X, w, \beta) = C - \frac{1}{2} \beta \sum_{n=1}^N (t_n - w^T \phi(x_n))^2
\]

Q: How to optimize the log-likelihood with respect to parameters \( w \)?
Maximum likelihood: least squares

Q: What is the likelihood of a data set \( \{(x_i, t_i)\}_{i=1}^{N} \)?

\[
p(t|X, w, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n | w^T \phi(x_n), \beta^{-1})
\]

Q: How does the log-likelihood depend on the parameters \( w \)?

\[
\ln p(t|X, w, \beta) = C - \frac{1}{2} \beta \sum_{n=1}^{N} \left( t_n - w^T \phi(x_n) \right)^2
\]

The sum-of-squares error function

Q: How to optimize the log-likelihood with respect to parameters \( w \)?

A: differentiate with respect to \( w \) and set to zero:

\[
\nabla_w \ln p(t|X, w, \beta) = -\beta \sum_{n=1}^{N} \left( t_n - w^T \phi(x_n) \right) \phi(x_n)^T = 0
\]
Maximum likelihood: least squares

Rewriting:

\[ 0 = \sum_{n=1}^{N} t_n \phi(x_n)^T - w^T \left( \sum_{n=1}^{N} \phi(x_n) \phi(x_n)^T \right) \]

Solving for \( w \):

\[ w_{ML} = (\Phi^T \Phi)^{-1} \Phi^T t \]

where \( \Phi \) is an \( N \times M \) matrix, the design matrix, with elements \( \Phi_{nj} = \phi_j(x_n) \):

\[ \Phi = \begin{pmatrix}
\phi_0(x_1) & \phi_1(x_1) & \ldots & \phi_{M-1}(x_1) \\
\phi_0(x_2) & \phi_1(x_2) & \ldots & \phi_{M-1}(x_2) \\
\vdots & \vdots & \ddots & \vdots \\
\phi_0(x_N) & \phi_1(x_N) & \ldots & \phi_{M-1}(x_N)
\end{pmatrix} \]

The matrix \( \Phi^T \Phi \) is \( M \times M \). If \( N < M \) the inverse does not exist.
Also, one can derive:

$$\beta_{ML}^{-1} = \frac{1}{N} \sum_{n=1}^{N} (t_n - w_{ML}^T \phi(x_n))^2.$$
Example: One dimensional linear regression

\[ M = 2, \phi_0 = 1, \phi_1 = x. \] Then

\[ y = w_0 + w_1 x + \epsilon \]

Given data \( \{(x_n, t_n), n = 1, \ldots, M\} \), find \( w_0, w_1, \beta \).

Step 1: find \( \Phi, t \).
Example: One dimensional linear regression

\( M = 2, \phi_0 = 1, \phi_1 = x \). Then

\[ y = w_0 + w_1 x + \epsilon \]

Given data \( \{(x_n, t_n), n = 1, \ldots, M\} \), find \( w_0, w_1, \beta \).

\[ \Phi = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) \\ \phi_0(x_2) & \phi_1(x_2) \\ \vdots & \vdots \\ \phi_0(x_N) & \phi_1(x_N) \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{pmatrix} \quad t = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{pmatrix} \]

Step 2: find \( \Phi^T \Phi, \Phi^T t \):
Example: One dimensional linear regression

\( M = 2, \phi_0 = 1, \phi_1 = x \). Then

\[
y = w_0 + w_1 x + \epsilon
\]

Given data \( \{(x_n, t_n), n = 1, \ldots, M\} \), find \( w_0, w_1, \beta \).

\[
\Phi = \begin{pmatrix}
\phi_0(x_1) & \phi_1(x_1) \\
\phi_0(x_2) & \phi_1(x_2) \\
\vdots & \vdots \\
\phi_0(x_N) & \phi_1(x_N)
\end{pmatrix}
= \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{pmatrix}
\quad t = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{pmatrix}
\]

\[
\Phi^T \Phi = \begin{pmatrix} N & \sum_n x_n \\ \sum_n x_n & \sum_n x_n^2 \end{pmatrix}
\quad \Phi^T t = \begin{pmatrix} \sum_n t_n \\ \sum_n x_n t_n \end{pmatrix}
\]

\[
w_{ML} = (\Phi^T \Phi)^{-1} \Phi^T t
\quad \beta_{ML} = \ldots
\]
Geometry of least squares

$M \leq N$:

$t = (t_1, \ldots, t_N)$ is vector in $R^N$.

$\phi_j = (\phi_j(x_1), \ldots, \phi_j(x_N)), j = 1, \ldots, M$ are $M$ vectors in $R^N$ that span $M$ dimensional linear subspace $S$.

$y = \sum_{j=1}^{M} w_j \phi_j$ is in $S$ for any $w$.

The optimal $w$ minimizes the quadratic error $\|t - y\|$ and is the ortho-normal projection.

$M > N$: vectors $\phi_j$ are over complete bases of $R^N$. Therefore, there are multiple solutions $w$ to solve $t = \sum_{j=1}^{M} w_j \phi_j$. The regression problem is ill-posed.
Regularized least squares

To avoid overfitting, we can add a regularization term to the log-likelihood, e.g. weight decay:

\[
\ln p(t|X, w, \beta) + \lambda E_W(w) = C - \frac{1}{2} \beta \sum_{i=1}^{N} (t_n - w^T \phi(x_n))^2 - \frac{1}{2} \lambda w^T w
\]

Maximizing with respect to \(w\) now gives the following optimum:

\[
w = (\lambda I + \Phi^T \Phi)^{-1} \Phi^T t
\]

NB: adding the diagonal makes the matrix of maximal rank.
Other regularizers

\[ \ln p(t|X, w, \beta) + \lambda E_W(w) \propto -\frac{1}{2} \beta \sum_{i=1}^{N} (t_n - w^T \phi(x_n))^2 - \frac{1}{2} \lambda \sum_{j=0}^{M-1} |w_j|^q \]

the LASSO with \( q = 1 \). This gives a \textit{sparse} solution \((w_j = 0 \text{ for many, but a few, } j\text{'s})\).
Other regularizers

Regularization is equivalent to constrained optimization

\[
\arg\min_w f(w) + \lambda \sum_{j=0}^{M-1} |w_j|^q \quad \Leftrightarrow \quad \arg\min_w f(w) + \lambda \left( \sum_{j=0}^{M-1} |w_j|^q - \eta \right)
\]

Fix \( \eta \) and find \( \lambda(\eta) \) rather than fix \( \lambda \).

Shows that \( q \leq 1 \) finds sparse solutions.
Bias-Variance Decomposition

Complex models tend to overfit. Simple models tend to be too rigid. Number of terms in polynomial, weight decay constant $\lambda$.

(Treatment of 1.5.5). Consider a regression problem with squared loss function

$$E(L) = \int (y(\mathbf{x}) - t)^2 p(\mathbf{x}, t) d\mathbf{x} dt$$

$p(\mathbf{x}, t)$ is the true underlying model, $y(\mathbf{x})$ is our estimate of $t$ at $\mathbf{x}$.

The optimal $y$ minimizes $E(L)$:

$$\frac{\delta E(L)}{\delta y(\mathbf{x})} = 2 \int (y(\mathbf{x}) - t) p(\mathbf{x}, t) dt = 0$$

$$y(\mathbf{x}) = \frac{\int tp(\mathbf{x}, t) dt}{p(\mathbf{x})} = \int tp(t|\mathbf{x}) dt = \mathbb{E}(t|\mathbf{x}) \equiv h(\mathbf{x})$$
The square loss can also be written as

$$\mathcal{E}(L) = \int (y(x) - t)^2 p(x, t) dx dt$$

$$= \int (y(x) - h(x) + h(x) - t)^2 p(x, t) dx dt$$

$$= \int \left( (y(x) - h(x))^2 + (h(x) - t)^2 \right) dx dt + 2 (y(x) - h(x)) (h(x) - t) p(x, t) dx dt$$

$$= \int (y(x) - h(x))^2 p(x) dx + \int (t - h(x))^2 p(x, t) dx dt$$

The first term depends on $y$ and is minimized when $y(x) = h(x)$. The second term is the noise variance in the conditional distribution of $t$ given $x$, averaged over $x$: $\int \text{var}(t|x) p(x) dx$, with $\text{var}(t|x) = \int dx p(t|x) (t - h(x))^2$

We further analyze the first term.
Bias-Variance Decomposition

When learning with a finite data set $\mathcal{D}$, the solution $y(x|\mathcal{D})$ that minimizes the quadratic loss depends $\mathcal{D}$.

$$\mathcal{E} = \sum_{\mu \in \mathcal{D}} (y(x^\mu) - t^\mu)^2 \rightarrow y(x; \mathcal{D})$$

Consider the thought experiment that a large number of data sets $\mathcal{D}$ are given. Then we can construct the average solution $\bar{h}(x) = \mathbb{E}_\mathcal{D}(y(x|\mathcal{D}))$, and

$$\left(y(x; \mathcal{D}) - h(x)\right)^2 = \left(y(x; \mathcal{D}) - \bar{h}(x)\right)^2 + \left(\bar{h}(x) - h(x)\right)^2$$

$$\mathbb{E}_\mathcal{D}\left(y(x; \mathcal{D}) - h(x)\right)^2 = \underbrace{\mathbb{E}_\mathcal{D}\left(y(x; \mathcal{D}) - \bar{h}(x)\right)^2}_{\text{variance}} + \underbrace{\mathbb{E}_\mathcal{D}\left(\bar{h}(x) - h(x)\right)^2}_{\text{bias}^2}$$

Substitution in expected square loss $\mathcal{E}(L)$:

$$\mathbb{E}_\mathcal{D}\mathcal{E}(L) = \int \mathbb{E}_\mathcal{D}(y(x) - h(x))^2 p(x) dx + \int (t - h(x))^2 p(x, t) dx dt$$

$$= \text{bias}^2 + \text{variance} + \text{noise}$$
Bias$^2$: difference between average solution $\mathbb{E}_D(y(x; D))$ and true solution $h(x) = \mathbb{E}(t|x)$.

Variance: scatter of individual solutions $y(x; D)$ around their mean $\mathbb{E}_D(y(x; D))$.

Noise: (true) scatter of the data points $t$ around their mean $\mathbb{E}(t|x)$.

Objective is to minimize bias$^2$ + variance. Noise cannot be optimized.

Difference models make different trade-off of bias and variance:

- Flexible/complex model: low bias, high variance
- Rigid/simple model: high bias, low variance

Best model strikes the best balance.
3.2 Bias-Variance Decomposition

100 data sets, each with 25 data points from \( t = \sin(2\pi x) + \text{noise. } y(x) \) as in Eq. 3.3-4. Parameters optimized using Eq. 3.27 for different \( \lambda \). Left shows variance, right shows bias.

Simple (high \( \lambda \)) solution has low variance and high bias. Complex (low \( \lambda \)) solution has high variance and low bias (NB relation to Bayesian averaging, but different)
Bias, variance and noise terms usually not known individually, because $\mathbb{E}(t|\mathbf{x})$ is not known, and require average over many data sets.

Sum of bias, variance and noise yields expected error on test set. Optimal $\lambda$ is trade-off between bias and variance.
Bayesian linear regression

Let's go back to the linear basis function model:

\[ t = y(x, w) + \epsilon, \quad y(x, w) = w^T \phi(x), \quad \epsilon \sim \mathcal{N}(0, \beta^{-1}) \]

\[ p(t|x, w, \beta) = \mathcal{N}(t|w^T \phi(x), \beta^{-1}) \]

Training data: \( X = (x_1, \ldots, x_N) \) and \( t = (t_1, \ldots, t_N) \).

Likelihood:

\[ p(t|x, w, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|w^T \phi(x_n), \beta^{-1}) = \mathcal{N}(t|\Phi w, \beta^{-1} I). \]

Q: what prior \( p(w) \) can we choose?
Bayesian linear regression

Let’s go back to the linear basis function model:

\[
t = y(x, w) + \epsilon, \quad y(x, w) = w^T \phi(x), \quad \epsilon \sim N(0, \beta^{-1})
\]

\[
p(t|x, w, \beta) = N(t|w^T \phi(x), \beta^{-1})
\]

Training data: \(X = (x_1, \ldots, x_N)\) and \(t = (t_1, \ldots, t_N)\).

Likelihood:

\[
p(t|x, w, \beta) = \prod_{n=1}^{N} N(t_n|w^T \phi(x_n), \beta^{-1}) = N(t|\Phi w, \beta^{-1} I).
\]

Q: what prior \(p(w)\) can we choose?

A: We make life easy by choosing a conjugate prior, which is a Gaussian

\[
p(w) = N(w|m_0, S_0)
\]
Bayesian linear regression

Then, the posterior will also be Gaussian. Prior and likelihood:

\[
p(w) = \mathcal{N}(w|m_0, S_0)
\]
\[
p(t|w) = \mathcal{N}(t|\Phi w, \beta^{-1}I)
\]

Then, by applying 2.113 + 2.114 \(\Rightarrow\) 2.116, we get the posterior (Ex. 3.7):

\[
p(w|t) = \mathcal{N}(w|m_N, S_N)
\]

with

\[
m_N = S_N(S_0^{-1}m_0 + \beta\Phi^T t)
\]
\[
S_N^{-1} = S_0^{-1} + \beta\Phi^T \Phi
\]

Since posterior is Gaussian, its mode is \(m_N\).

When \(S_0^{-1} \rightarrow 0\) (broad prior) \(m_N \rightarrow (\Phi^T \Phi)^{-1}\Phi^T t\) which is the ML solution.

\[\Phi^T \Phi \propto N^{10}\]. Therefore \(S_N \rightarrow 0\) when \(N\) large.

\[
10(\Phi^T \Phi)_{ij} = \sum_n (\Phi^T)_{in} \Phi_{nj} = \sum_n \Phi_{ni} \Phi_{nj} = \sum_n \phi_i(x_n) \phi_j(x_n)
\]
Bayes’ theorem for linear Gaussian model

Given marginal Gaussian on $x$ and linear relation $y = Ax + b + \xi$:

\[
p(x) = \mathcal{N}(x|\mu, \Lambda^{-1})
\]
\[
p(y|x) = \mathcal{N}(y|Ax + b, L^{-1})
\]

Then (see next slide):

\[
p(y) = \mathcal{N}(y|A\mu + b, L^{-1} + A\Lambda^{-1}A^T)
\]
\[
p(x|y) = \mathcal{N}(x|\Sigma\{A^TL(y - b) + \Lambda\mu\}, \Sigma)
\]
\[
\Sigma = (\Lambda + A^T L A)^{-1}
\]
Details

Correspondence with (2.113-114):

\[ p(x) = \mathcal{N}(x|\mu, \Lambda^{-1}) \quad p(w) = \mathcal{N}(w|m_0, S_0) \]
\[ p(y|x) = \mathcal{N}(y|Ax + b, L^{-1}) \quad p(t|w) = \mathcal{N}(t|\Phi w, \beta^{-1}I) \]

\[ \mu = m_0, \Lambda^{-1} = S_0, A = \Phi, b = 0, L = \beta I. \]

Thus from (2.116), posterior \( p(w|t) = \mathcal{N}(w|m_N, S_N) \) with

\[ m_N = S_N(\beta \Phi^T t + S_0^{-1}m_0) \]
\[ S_N^{-1} = S_0^{-1} + \beta \Phi^T \Phi \]
Bayesian linear regression in 1 dimension

Generate $N$ data points $t_n = a_0 + a_1 x_n + \epsilon_n$ with $a_0 = -0.3$ and $a_1 = 0.5$, $x_n \in U(-1, 1)$ and $\epsilon_n \sim \mathcal{N}(\epsilon|0, \beta^{-1})$ with $\sigma = 0.2$. Aim: estimate $a_0, a_1$ from data.

Likelihood model: $y(x, w) = w_0 + w_1 x$; $p(t|x, w, \beta) = \mathcal{N}(t|y(x, w), \beta^{-1})$, $\beta = 1/\sigma^2 = 25$;
Prior model $p(w|\alpha) = \mathcal{N}(w|0, \alpha^{-1} I)$, $\alpha = 2$. $\alpha, \beta$ assumed known.

Posterior variance reduces with $N$. 
Predictive distribution

What is the predictive distribution \( p(t^*|x^*) \) for new data point \( x^* \)? We know:

\[
\begin{align*}
    p(t^*|w, x^*) &= \mathcal{N}(t^*|\phi^T(x^*)w, \beta^{-1}) \\
p(w|t, x) &= \mathcal{N}(w|m_N, S_N) \\
p(t^*|x^*, t, x) &= \int dw p(t^*|w, x^*)p(w|t, x)
\end{align*}
\]

Write \( t^* = \phi^T(x^*)w + \epsilon \) with \( \mathbb{V}\epsilon = \beta^{-1} \):

\[
\begin{align*}
    \mathbb{E}t^* &= \phi^T(x^*)m_N, \\
    \mathbb{V}t^* &= \mathbb{V}\left(\phi^T(x^*)w\right) + \mathbb{V}\epsilon = \phi^T(x^*)\mathbb{V}(w)\phi(x^*) + \beta^{-1}
\end{align*}
\]

\[
p(t^*|x^*, t, x) = \mathcal{N}(t^*|\phi^T(x^*)m_N, \sigma^2_N(x^*))
\]

where

\[
\sigma^2_N(x^*) = \beta^{-1} + \phi(x^*)^T S_N \phi(x^*)
\]

When \( N \to \infty \), \( S_N \to 0 \) and \( \sigma^2_N(x^*) \to \beta^{-1} \).
3.3

Bayesian linear regression: Example

Data points from $t = \sin(2\pi x) + \text{noise}$. $y(x)$ as in Eq. 3.3-4. Red is $\phi(x^*)^T m_N \pm \sigma_N(x^*)$. Data set size $N = 1, 2, 4, 25$. 
Same data and model. Curves $y(x, \mathbf{w})$ with $\mathbf{w}$ from posterior $p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m_N}, S_N)$. 
Equivalent kernel

The output $y$ for the mean posterior prediction $w = m_N = \beta S_N \Phi^T t$ (assume $m_0 = 0$) is

$$y(x, m_N) = \phi(x)^T m_N = \beta \phi(x)^T S_N \Phi(x)^T t = \beta \sum_{n=1}^{N} \phi(x)^T S_N \phi(x_n) t_n$$

$$= \sum_{n=1}^{N} k(x, x_n) t_n$$

Mean output is linear combination of outputs at training points. Weighting is localized around $x$.

Kernels for basis functions in fig. 3.1: Gaussian (left), polynomials (middle), sigmoids (right)
Equivalent kernel

The outputs at two different inputs are correlated:

\[
Cov \left( y(x), y(x') \right) = Cov(\phi(x)^T w, w^T \phi(x')) = \phi(x)^T V w \phi(x') = \phi(x)^T S_N \phi(x')
\]

\[= \beta^{-1} k(x, x') \]

Nearby points are highly correlated, and distant points are uncorrelated.

One can specify a regression problem in terms of a kernel directly instead of a linear combination of basis functions (chapter 6).
Bayesian model comparison

Maximum likelihood suffers from overfitting, which requires testing models of different complexity on separate data.

Bayesian approach allows to compare different models directly on the training data, but requires integration over model parameters.

Consider $L$ probability models and a set of data generated from one of these models. We define a prior over models $p(M_i), i = 1, \ldots, L$ to express our prior uncertainty.

Given the training data $D$, we wish to compute the posterior probability

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

$p(D|M_i)$ is called model evidence, also marginal likelihood, since it integrates over model parameters:

$$p(D|M_i) = \int p(D|w, M_i)p(w|M_i)dw$$
Bayesian model comparison

Consider three models of increasing complexity $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$. Consider drawing data sets from these models: we first sample a parameter vector $\mathbf{w}$ from the prior $p(\mathbf{w}|\mathcal{M}_i)$ and then generate iid data points according to $p(x|\mathbf{w}, \mathcal{M}_i)$. The resulting distribution is $p(D|\mathcal{M}_i)$.

A simple model has less variability in the resulting data sets than a complex model. Thus, $p(D|\mathcal{M}_1)$ is more peaked than $p(D|\mathcal{M}_3)$. Due to normalization, $p(D|\mathcal{M}_1)$ is necessarily higher than $p(D|\mathcal{M}_3)$.

![Diagram showing the distribution of data sets for models $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$]

For the data set $D_0$ the Bayesian approach will select model $\mathcal{M}_2$ because model $\mathcal{M}_1$ is too simple (does not explain the data) and model $\mathcal{M}_3$ is too complex (can explain too many data sets). This is known as Bayesian model selection.
Evidence framework for Bayesian linear regression

The Bayesian linear regression approach assume a prior (3.48)

\[ p(w|\alpha, M) = \left(\frac{\alpha}{2\pi}\right)^{M/2} \exp \left(-\frac{\alpha}{2}w^T w\right) \]

and a likelihood (3.10)

\[ p(t|w, \beta, M) = \left(\frac{\beta}{2\pi}\right)^{N/2} \exp \left(-\frac{\beta}{2}\|t - \Phi w\|^2\right) \]

The marginal likelihood is

\[
p(t|\alpha, \beta, M) = \int dw p(w|\alpha, M)p(t|w, \beta, M) = \left(\frac{\alpha}{2\pi}\right)^{M/2} \left(\frac{\beta}{2\pi}\right)^{N/2} \int dw \exp(-E(w))
\]
Evidence framework

\[ E(w) = \frac{\beta}{2} (t - \Phi w)^T (t - \Phi w) + \frac{\alpha}{2} w^T w = \frac{\beta}{2} \left( t^T t - 2 w^T \Phi^T t + w^T \Phi^T \Phi w \right) + \frac{\alpha}{2} w^T w \]

\[ = \frac{1}{2} w^T A w + \frac{\beta}{2} t^T t - \beta w^T \Phi^T t = \frac{1}{2} w^T A w + \frac{\beta}{2} t^T t - w^T A m \]

with \( A = \alpha I + \beta \Phi^T \Phi \) and \( A m = \beta \Phi^T t \). Thus,

\[ E(w) = \frac{1}{2} (w - m)^T A (w - m) - \frac{1}{2} m^T A m + \frac{\beta}{2} t^T t \]

\[ = \frac{1}{2} (w - m)^T A (w - m) + E(m) \]

With \( \int d\omega \exp(-E(w)) = \exp(-E(m)) (2\pi)^{-M/2} |A|^{-1/2} \):

\[ \log p(t|\alpha, \beta, M) = \frac{M}{2} \log \alpha + \frac{N}{2} \log \beta - E(m) - \frac{1}{2} \log |A| - \frac{M}{2} \log 2\pi \]

\(^{11}\) Note that \( |A| = \prod \lambda_i \), thus \( \log |A| = O(M) \).

\(^{11}\)NB typo Eq. 3.86 \(-\frac{N}{2} \log 2\pi\)
Evidence framework comparing different $M$ for fixed $\alpha, \beta$. $M = 1$ improves over $M = 0$. $M = 2$ does not improve over $M = 1$. $M = 3$ improves over $M = 2$. Models $M = 3 - 8$ have different likelihood but increasing complexity.
References


Gaussian processes

(Regression setting). Data set \(\{x_n, t_n\}_{n=1:N}\) Assume a model

\[
t_n = y_n + \epsilon_n \quad t = y + \epsilon
\]

with \(\epsilon \sim \mathcal{N}(\epsilon|\beta^{-1}I_N)\) and

\[
y \sim \mathcal{N}(y|0, K(x_{1:N}))
\]

\(K(x_{1:N})\) is a \(N \times N\) covariance matrix with components

\[
K(x_n, x_{n'}) = \theta_0 \exp \left(-\frac{\theta_1}{2} \|x_n - x_{n'}\|^2\right)
\]
Since \( t = y + \epsilon \) we get \( p(t|x_{1:N}) = \mathcal{N}(t|0, C_N(x_{1:N})) \) (\( t = t_{1:N} \)):

\[
\mathbb{E}t = 0 \quad C_N(x_{1:N}) = \nabla t = \nabla y + \nabla \epsilon = K(x_{1:N}) + \beta^{-1}I
\]

Same holds when we add extra data point \( x, t \): \( t_{N+1} = (t_{1:N}, t) \) and \( x_{N+1} = (x_{1:N}, x) \):

\[
p(t_{N+1}|x_{1:N+1}) = \mathcal{N}(t_{N+1}|0, C_{N+1})
\]

\( p(t_{N+1}) = p(t_{1:N}, t_{N+1}) \) is Gaussian. We can compute the marginal \( p(t_{N+1}|t_{1:N}) \) using results Eqs. 2.81-82 (sheets 157-160)

\[
p(x_a, x_b) = \mathcal{N}(x_a, x_b|\mu, \Sigma) \quad \mu = (\mu_a, \mu_b), \quad \Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}
\]

\[
p(x_a|x_b) = \mathcal{N}(x_a|\mu_{a|b}, \Sigma_{a|b})
\]

\[
\Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba} \quad \mu_{a|b} = \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(x_b - \mu_b)
\]

\( x_a = t_{N+1}, x_b = t = t_{1:N}, \mu = 0 \) and

\[
\Sigma = C_{N+1} = \begin{pmatrix} c & k^T \\ k & C_N \end{pmatrix}
\]
Thus,

\[ p(t_{N+1}|t_{1:N}) = \mathcal{N}(t|\mu_{N+1}, \sigma_{N+1}^2), \quad \mu_{N+1} = k^T C_N^{-1} t \quad \sigma_{N+1}^2 = c - k^T C_N^{-1} k \]

Left: Computation of \( p(t_2|t_1) \) from joint distribution \( p(t_1, t_2) \) which has mean zero and covariance depending on \( x_1, x_2 \). Right: Marginal \( p(t_{N+1}|t_{1:N}, x_{1:N}, x) \) versus \( x \)
Automatic relevance determination

Choose different length scales for different dimensions

\[ K(\mathbf{x}, \mathbf{x}') = \theta_0 \exp \left( -\frac{1}{2} \sum_{i=1}^{d} \eta_i (x_i - x_i')^2 \right) \]

Left \( \eta_1 = \eta_2 = 1 \). Right \( \eta_1 = 1, \eta_2 = 0.01 \).
# Population variance explained by GWAS

<table>
<thead>
<tr>
<th>Trait/Disease</th>
<th>$H^2$ Pedigree Studies</th>
<th>$h^2$ GWAS hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type I diabetes</td>
<td>0.9</td>
<td>0.6</td>
</tr>
<tr>
<td>Type II diabetes</td>
<td>0.3-0.6</td>
<td>0.05-0.1</td>
</tr>
<tr>
<td>Obesity (BMI)</td>
<td>0.4-0.6</td>
<td>0.01-0.02</td>
</tr>
<tr>
<td>Crohn’s disease</td>
<td>0.6-0.8</td>
<td>0.1</td>
</tr>
<tr>
<td>Multiple Sclerosis</td>
<td>0.3-0.8</td>
<td>0.1</td>
</tr>
<tr>
<td>Schizophrenia</td>
<td>0.7-0.8</td>
<td>0.01</td>
</tr>
<tr>
<td>Bipolar disorder</td>
<td>0.6-0.7</td>
<td>0.02</td>
</tr>
<tr>
<td>ADHD</td>
<td>0.6-1.0</td>
<td>0</td>
</tr>
</tbody>
</table>

Source: Visscher et al. 2012 (ADHD: Stergiakouli et al. 2011)
Features of the regression problem

• Sparse
  – More dimensions than data samples
  – Low number of significant SNPs

• Unknown mechanism
  – Possible interaction models are of unknown order and complexity.

• Common approaches
  – linear univariate or pairwise
  – path ways
Bayesian GP regression with sparse ARD

GP regression: \( p(t_{1:N} | y) \) and \( p(y | \eta, x_{1:N}) = \mathcal{N}(y, 0, K(x_{1:N}, \eta)) \)

Sparse ARD priors over length scales \( \eta_{1:D} \): \( p(\eta | \theta) \) by proper choice of \( \theta \).

Integrate out \( y \):

\[
p(t_{1:N} | x_{1:N}, \eta) = \int dyp(t_{1:N} | y)p(y | x_{1:N}, \eta)
\]

is Gaussian.

Compute Bayesian posterior

\[
p(\eta | t_{1:N}, x_{1:N}, \theta) \propto p(t_{1:N}, \eta | x_{1:N}, \theta) = p(t_{1:N} | x_{1:N}, \eta)p(\eta | \theta)
\]
Monte Carlo Issues

GP posterior computation uses Hamiltonian Monte Carlo sampling

- Run times hours to days.
- Multi-modality: hard to find high density region

5-snp model; \( \lambda = 1 \); Signal/Noise= 4/1; 5000 snps. Left: Log posterior density. Middle: Log noise precision. Right: GPU vs CPU times

10^5 snps and 2000 samples in 20 seconds on GPU. Approx 300 times speed-up over CPU
Simulation Data

- Construct 500 genotype samples with 5000 snps from IMAGE-1
- Construct phenotype model
  - Randomly choose model snps
  - Generate phenotype from 5-way interaction model
    * $\lambda = 0$ linear/additive model
    * $\lambda = 1$ non-linear (AND5) model
Classification performance

Compute feature strength for each SNP ($\left(\mathbb{E} \tau_k\right)^{-1}$ in case of GP)
Classify SNPs with highest strength as positive and rest as negative.
Construct ROC curve and compute AUC.
**Significance - \( \lambda = 1 \)**

**Univariate**

**GP**

- \( -\log(p \text{ value}) \)
- \( \langle -\log(\tau) \rangle \)

Bonferroni threshold
Finding missing heritability in Yeast (Bloom et al., Nat. 2013.)

- 1008 genetically distinct inbred cell-lines. 46 different phenotypes
- Pairwise interactions explain additional 3 % median genetic variance per trait.
Identifying important variants - Zeocin

- red: GP and linear agree. dark blue: GP only
- SNP 16 found relevant by GPR, but has no additive effect.
- All missing heritability $\sim 35\%$ apparently explained by GPR
Support vector machines

Kernel methods (for classification): Training data \( \{x_n, t_n = \pm 1\}_{n=1:N} \)

\[
y(x) = \sum_{n=1}^{N} a_n t_n K(x, x_n) \quad \text{class}(x) = \text{sign}(y(x))
\]

select a subset of \( a_n \geq 0 \). This is a convex optimization.

Number of SVs increases with data
Partial derivatives, gradient, Nabla symbol

Let $f(x_1, \ldots, x_n) = f(x)$ be a function of several variables. The gradient of $f$, denoted as $\nabla f$ (the symbol “$\nabla$” is called 'nabla'), is the vector of all partial derivatives:

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

NB, the partial derivative $\frac{\partial f(x_1, \ldots, x_i, \ldots, x_n)}{\partial x_i}$ is computed by taking the derivative with respect to $x_i$ while keeping all other variables constant.

Example:

$$f(x, y, z) = xy^2 + 3.1yz$$

Then

$$\nabla f(x, y, z) = \left( y^2, 2xy + 3.1z, 3.1y \right)^T$$

At local minima (and maxima, and so-called saddle points) of a differentiable function $f$, the gradient is zero, i.e., $\nabla f = 0$.

Example:

$$f(x, y) = x^2 + y^2 + (y + 1)x$$

So

$$\nabla f(x, y) = (2x + y + 1, 2y + x)^T$$
Then we can compute the point \((x^*, y^*)\) that minimizes \(f\) by setting \(\nabla f = 0\),

\[
\begin{align*}
2x^* + y^* + 1 &= 0 \\
2y^* + x^* &= 0
\end{align*}
\]

\[\Rightarrow (x^*, y^*) = \left(-\frac{2}{3}, \frac{1}{3}\right)\]
Chain rule

Suppose $f$ is a function of $y_1, y_2, ..., y_k$ and each $y_j$ is a function of $x$, then we can compute the derivative of $f$ with respect to $x$ by the chain rule

$$\frac{df}{dx} = \sum_{j=1}^{k} \frac{\partial f}{\partial y_j} \frac{dy_j}{dx}$$

Example:

$$f(y(x), z(x)) = \frac{y(x)}{z(x)}$$

and $y(x) = x^4$ and $z = x^2$ then

$$\frac{df}{dx} = \frac{1}{z(x)} y'(x) - \frac{y(x)}{z(x)^2} z'(x) = 2x$$
Chain rule (2)

Suppose $E$ is a function of $y_1, y_2, \ldots, y_N$ and each $y_j$ is a function of $w_0, \ldots, w_M$, then we can compute the derivative of $E$ with respect to $w_i$ by the chain rule

$$\frac{\partial E}{\partial w_i} = \sum_{j=1}^{N} \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial w_i}$$

Example:

$$E(w) = \frac{1}{2} \sum_{j=1}^{N} (y_j(w) - t_j)^2 \Rightarrow \frac{\partial E}{\partial y_j} = y_j - t_j$$

and

$$y_j(w) = \sum_{i=0}^{M} x_{j}^{i} w_i \Rightarrow \frac{\partial y_j}{\partial w_i} = x_{j}^{i}$$

So

$$\frac{\partial E}{\partial w_i} = \sum_{j=1}^{N} (y_j(w) - t_j)x_{j}^{i}$$

$t_j$ and $x_{j}^{i}$ are parameters in this example.
Matrix multiplications as summations

If $A$ is a $N \times M$ matrix with entries $A_{ij}$ and $v$ an $M$-dimensional vector with entries $v_i$, then $w = Av$ is a $N$-dimensional vector with components

$$w_i = \sum_{j=1}^{M} A_{ij} v_j$$

Similarly, if $B$ is a $M \times K$ matrix with entries $B_{ij}$, then $C = AB$ is a $N \times K$ matrix with entries

$$C_{ik} = \sum_{j=1}^{M} A_{ij} B_{jk}$$
Dummy indices

The indices that are summed over are ‘dummy’ indices, they are just a label, so e.g.,

\[
\sum_{k=1}^{M} A_{ik} B_{kj} = \sum_{l=1}^{M} A_{il} B_{lj}
\]

furthermore, the entries of the vectors and matrices are just ordinary numbers, so you don’t have to worry about multiplication order. In addition, if the summation of indices is over a range that does not depend on other indices, you may interchange the order of summation,

\[
\sum_{i=1}^{N} \sum_{j=1}^{M} \ldots = \sum_{j=1}^{M} \sum_{i=1}^{N} \ldots
\]

So e.g., by changing summation order and renaming dummy indices,

\[
w_k = \sum_{j=1}^{M} \sum_{i=1}^{N} A_{ij} B_{jk} = \sum_{l=1}^{N} \sum_{i=1}^{M} B_{lk} A_{li}
\]
Kronecker delta

The notation $\delta_{ij}$ denotes usually the Kronecker delta symbol, i.e.,

\[
\begin{cases}
\delta_{ij} = 1 & \text{if} \quad i = j \\
\delta_{ij} = 0 & \text{otherwise}
\end{cases}
\]

It has the nice property that it ‘eats’ dummy indices in summations:

\[
\sum_{j=1}^{M} \delta_{ij} v_j = v_i \quad \text{for all} \quad 1 \leq i \leq M
\]  \hspace{1cm} (3)

The Kronecker delta can be viewed as the entries of the identity matrix $I$. In vector notation, (3) is equivalent to the statement $Iv = v$. In other words, $\delta_{ij} = I_{ij}$\textsuperscript{12}

\textsuperscript{12}Bishop used $\delta$ in his previous book, and $I$ in the current book
Taylor series, 1-d

Assuming that $f(x)$ has derivatives of all orders in $x = a$, then the Taylor expansion of $f$ around $a$ is

$$f(a + \epsilon) = \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!} \epsilon^k = f(a) + \epsilon f'(a) + \frac{\epsilon^2}{2} f''(a) + \ldots$$

The prefactors in the Taylor series can be checked by computing the Taylor expansion of a polynomial.

Linearization of a function around $a$ is taking the Taylor expansion up to first order:

$$f(a + x) = f(a) + x f'(a)$$
**Taylor series, examples**

**Examples:** check that for small $x$ the following expansions are correct up to second order:

\[
\begin{align*}
\sin(x) & = x \\
\cos(x) & = 1 - \frac{1}{2}x^2 \\
\exp(x) & = 1 + x + \frac{1}{2}x^2 \\
(1 + x)^c & = 1 + cx + \frac{c(c - 1)}{2}x^2 \\
\ln(1 + x) & = x - \frac{1}{2}x^2
\end{align*}
\]
Taylor expansion in several dimensions

The Taylor expansion of a function of several variables, $f(x_1, \ldots, x_n) = f(\mathbf{x})$ is (up to second order)

$$f(\mathbf{x}) = f(\mathbf{a}) + \sum_i (x_i - a_i) \frac{\partial}{\partial x_i} f(\mathbf{a}) + \frac{1}{2} \sum_{ij} (x_i - a_i)(x_j - a_j) \frac{\partial}{\partial x_i} \frac{\partial}{\partial x_j} f(\mathbf{a})$$

or in vector notation, with $\epsilon = \mathbf{x} - \mathbf{a}$

$$f(\mathbf{a} + \epsilon) = f(\mathbf{a}) + \epsilon^T \nabla f(\mathbf{a}) + \frac{1}{2} \epsilon^T \mathbf{H} \epsilon$$

with $\mathbf{H}$ the Hessian, which is the symmetric matrix of partial derivatives

$$H_{ij} = \left. \frac{\partial^2}{\partial x_i \partial x_j} f(\mathbf{x}) \right|_{\mathbf{x}=\mathbf{a}}$$
Integration

The integral of a function of several variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$

$$\int_{\mathcal{R}} f(\mathbf{x})\,d\mathbf{x} \equiv \int_{\mathcal{R}} f(x_1, x_2, \ldots, x_n)\,dx_1\,dx_2\ldots\,dx_n$$

is the volume of the $n + 1$ dimensional region lying 'vertically above' the domain of integration $\mathcal{R} \subset \mathbb{R}^n$ and 'below' the function $f(\mathbf{x})$. 
Separable integrals

The most easy (but important) case is when we can separate the integration, e.g. in 2-d,

\[ \int_{x=a}^{b} \int_{y=c}^{d} f(x)g(y) \, dx \, dy = \int_{x=a}^{b} f(x) \, dx \int_{y=c}^{d} g(y) \, dy \]

Example,

\[ \int \exp \left( \sum_{i=1}^{n} f_i(x_i) \right) \, dx = \int \prod_{i=1}^{n} \exp(f_i(x_i)) \, dx = \prod_{i=1}^{n} \int \exp(f_i(x_i)) \, dx_i \]
Iterated integration

A little more complicated are the cases, in which integration can be done by iteration, 'from inside out'. Suppose we can write the 2-d region $\mathcal{R}$ as the set $a < x < b$ and $c(x) < y < d(x)$ then we can write

$$\int_{\mathcal{R}} f(y, x) \, dy \, dx = \int_{x=a}^{b} \left[ \int_{y=c(x)}^{d(x)} f(y, x) \, dy \right] \, dx$$

The first step is evaluate the inner integral, where we interpret $f(y, x)$ as a function of $y$ with fixed parameter $x$. Suppose we can find $F$ such that $\partial F(y, x)/\partial y = f(y, x)$, then the result of the inner integral is

$$\int_{y=c(x)}^{d(x)} f(y, x) \, dy = F(d(x), x) - F(c(x), x)$$

The result, which we call $g(x)$ is obviously a function of $x$ only,

$$g(x) \equiv F(d(x), x) - F(c(x), x)$$

The next step is the outer integral, which is now just a one-dimensional integral of the function $g$,

$$\int_{x=a}^{b} \left[ \int_{y=c(x)}^{d(x)} f(y, x) \, dy \right] \, dx = \int_{x=a}^{b} g(x) \, dx$$

Now suppose that the same 2-d region $\mathcal{R}$ can also be written as the set $s < y < t$ and $u(y) < x < v(y)$,
then we can also choose to evaluate the integral as

$$\int_R f(y, x) \, dx \, dy = \int_{y=s}^{t} \left[ \int_{x=u(y)}^{v(y)} f(y, x) \, dx \right] \, dy$$

following the same procedure as above. In most regular cases the result is the same (for exceptions, see handout (*)).

Integration with more than two variables can be done with exactly the same procedure, ‘from inside out’.

In Machine Learning, integration is mostly over the whole of $\mathbb{R}$ space, or over a subspace. Iterated integration is not often used.
Dirac’s delta-function

Dirac’s delta function $\delta(x)$ is defined such that

$$\delta(x) = 0 \text{ if } x \neq 0 \quad \text{and} \quad \int_{-\infty}^{\infty} \delta(x) \, dx = 1$$

It can be viewed as the limit $\Delta \to 0$ of the function

$$f(x, \Delta) = \frac{1}{\Delta} \text{ if } |x| \leq \frac{\Delta}{2} \quad \text{and} \quad f(x, \Delta) = 0 \text{ elsewhere}$$
The Dirac delta $\delta(x)$ is a spike (a peak, a point mass) at $x = 0$. The function $\delta(x - x_0)$ as a function of $x$ is a spike at $x_0$. As a consequence of the definition, the delta function has the important property

$$\int_{-\infty}^{\infty} f(x) \delta(x - x_0) dx = f(x_0)$$

(cf. Kronecker delta $\sum_j \delta_{ij} v_j = v_i$).

The multivariate deltafunction factorizes over the dimensions

$$\delta(\mathbf{x} - \mathbf{m}) = \prod_{i=1}^{n} \delta(x_i - m_i)$$
Dirac’s delta-function / delta-distribution

The Dirac delta is actually a distribution rather than a function:

\[ \delta(\alpha x) = \frac{1}{\alpha} \delta(x) \]

This is true since

- if \( x \neq 0 \) left and right-hand-side are both zero.
- after transformation of variables \( x' = \alpha x, \, dx' = \alpha dx \) we have

\[
\int \delta(\alpha x) dx = \frac{1}{\alpha} \int \delta(x') dx' = \frac{1}{\alpha}
\]
Functionals vs functions

Function $y$: for any input value $x$, returns output value $f(y)$.

Functional $F$: for any function $y$, returns an output value $F[y]$.

Example (linear functional):

$$F[y] = \int p(x)y(x) \, dx$$

(Compare with $f(y) = \sum_i p_i y_i$).

Other (nonlinear) example:

$$F[y] = \int \frac{1}{2}(y'(x) + V(x))^2 \, dx$$