Introduction to Machine Learning

Lecturer: Bert Kappen
Radboud University Nijmegen, Netherlands

October 16, 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`

Inleiding Machine Learning

- 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

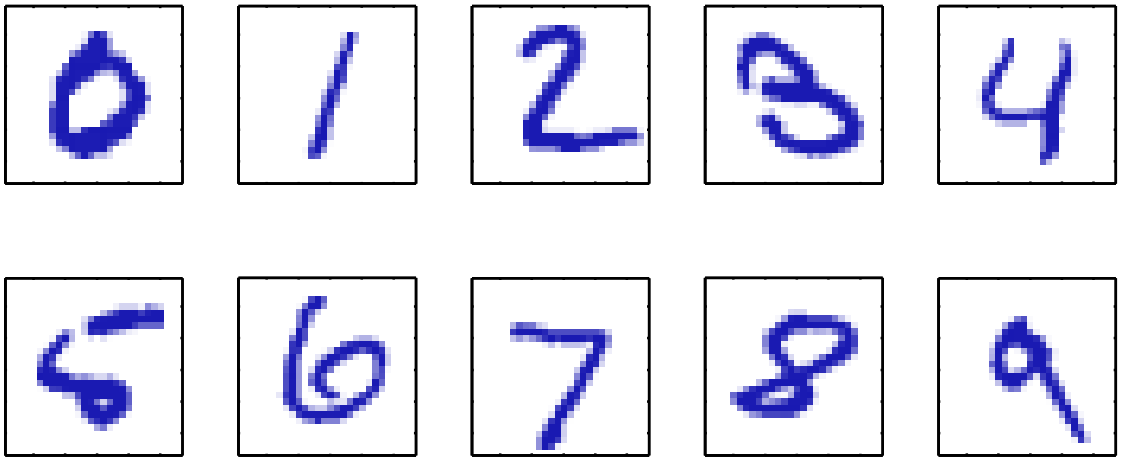


Image is array of pixels x_i, each between 0 and 1.
$\mathbf{x} = (\mathbf{x_1}, \ldots, \mathbf{x_d})$, d is the total number of pixels. $(d = 28 \times 28)$.

- Goal = input: pixels $\rightarrow$ output: correct category $0, \ldots, 9$
- wide variablity
- hand-made rules infeasible

Machine Learning

- Training set: large set of (pixel array, category) pairs
  - input data $\mathbf{x}$, target data $\mathbf{t}(\mathbf{x})$
  - category $=$ class
- Machine learning algorithm
  - training, learning
- Result: function $\mathbf{y}(\mathbf{x})$
  - Fitted to target data
  - New input $\mathbf{x} \rightarrow$ output $\mathbf{y}(\mathbf{x})$
  - Hopefully: $\mathbf{y}(\mathbf{x}) = \mathbf{t}(\mathbf{x})$
  - Generalization on new examples (test set)

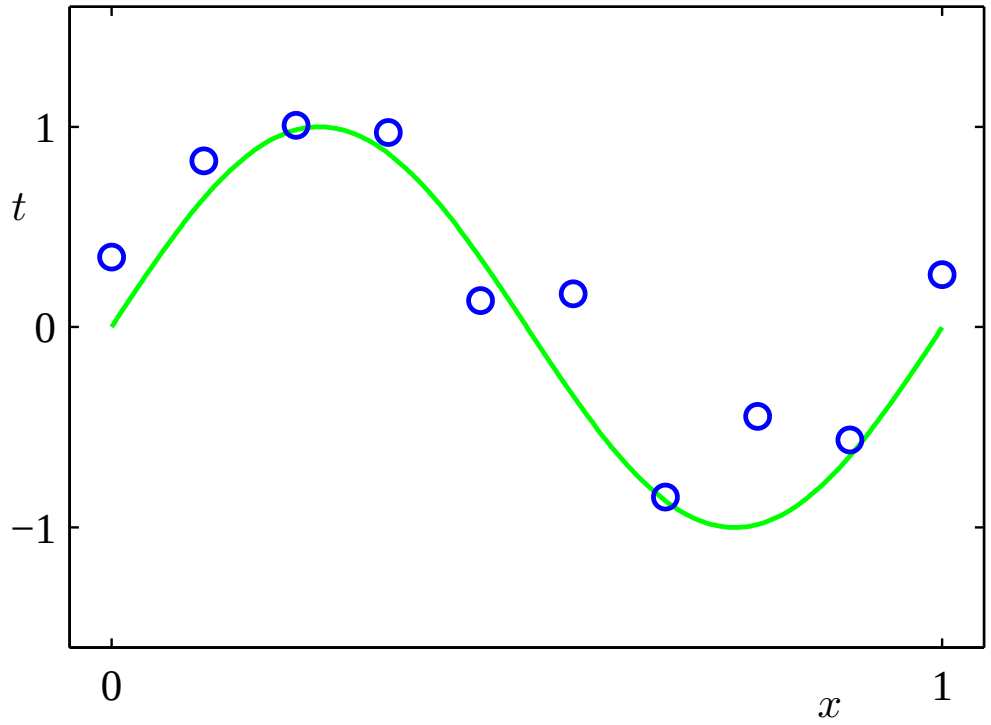
Preprocessing

- transformation of inputs $\mathbf{x}$ 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 $t_n = \sin(2\pi x) + noise$
- Goal: predict value t for new x (without knowing the curve)

We will fit the data using M-th order polynomial

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

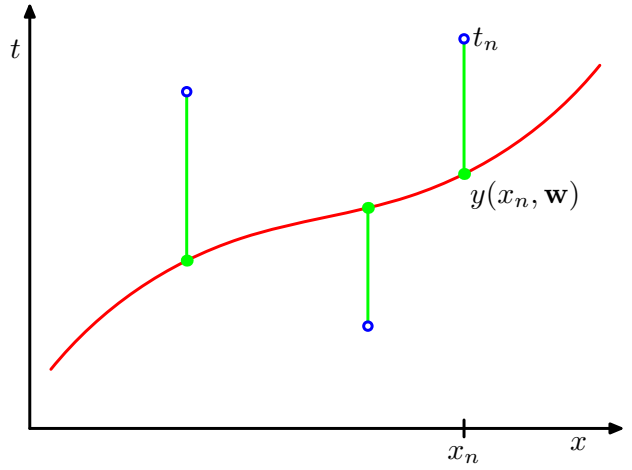
$y(x, \boldsymbol{w})$ is nonlinear in x, but linear in coefficients $\boldsymbol{w}$, "*Linear model*"

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

This is done by minimizing the *Error function*

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

$E(\boldsymbol{w}) \geq 0$ and $E(\boldsymbol{w}) = 0 \Leftrightarrow y(x_n, \boldsymbol{w}) = t_n$



Minimization of the error function

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

In minimum: gradient $\nabla_{\boldsymbol{w}} E = 0$

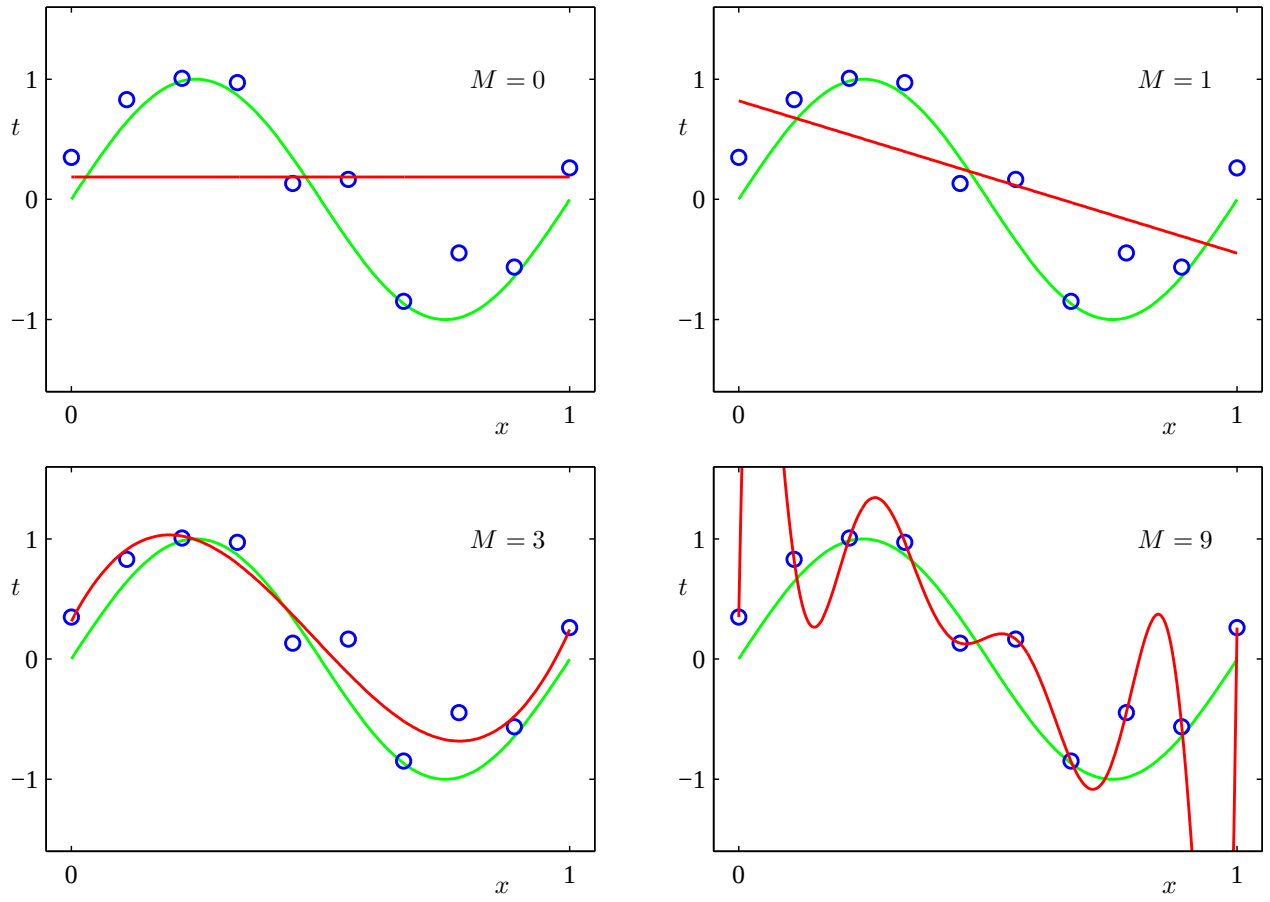
Note: y linear in $\boldsymbol{w} \Rightarrow E$ quadratic in $\boldsymbol{w}$

$\Rightarrow \nabla_{\boldsymbol{w}} E$ is linear in $\boldsymbol{w}$

$\Rightarrow \nabla_{\boldsymbol{w}} E = 0$: coupled set of linear equations (exercise)

Model comparison, model selection

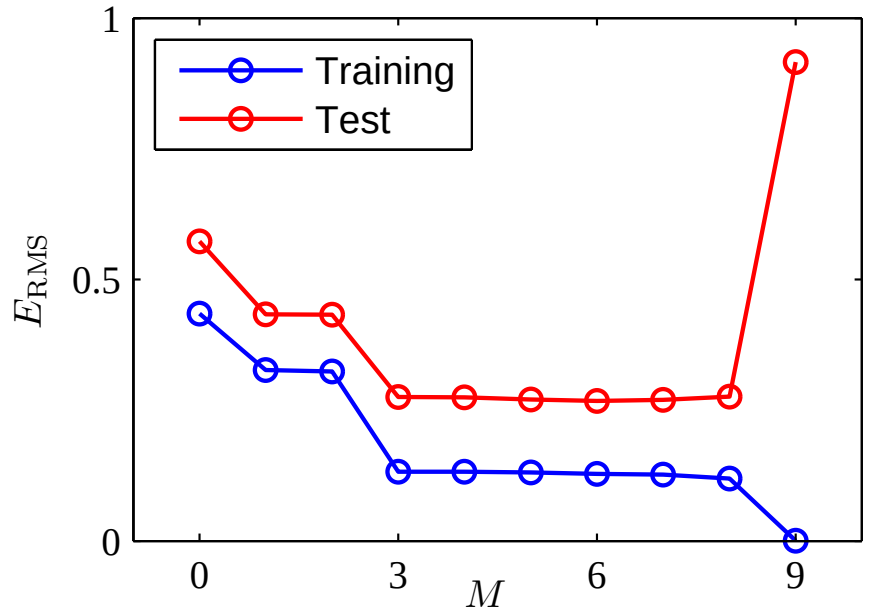
Back to polynomial curve fitting: how to choose M?



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:

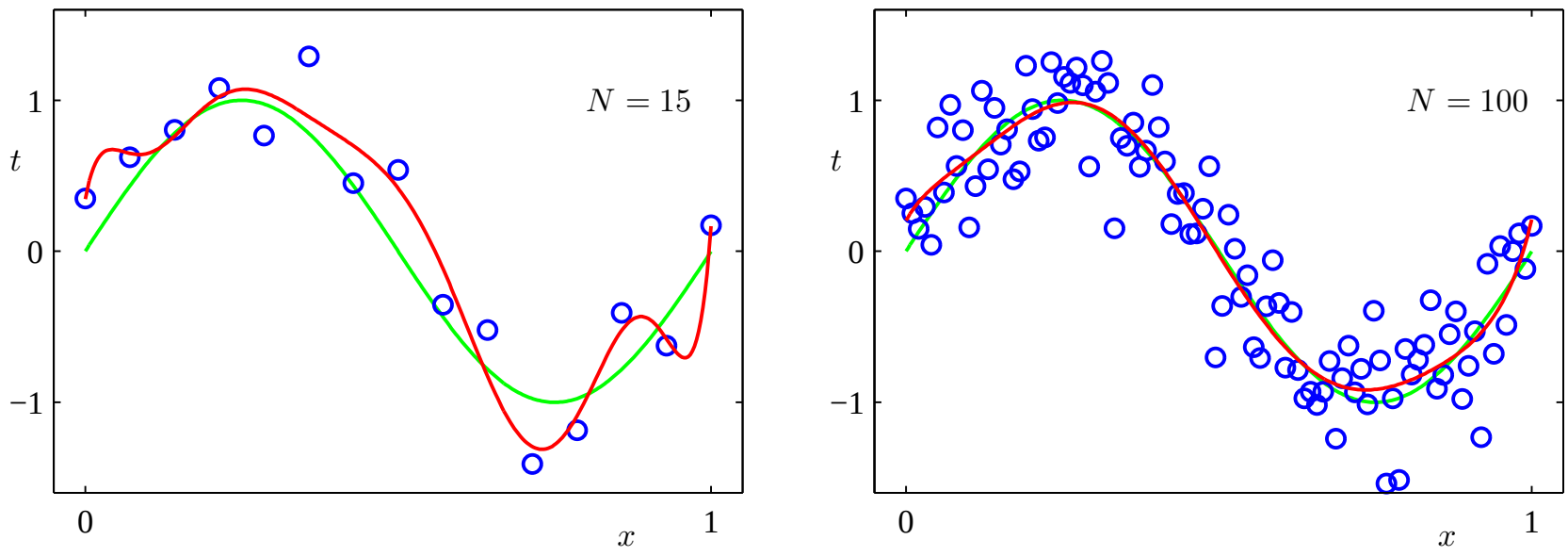
$$E_{RMS} = \sqrt{2E(\boldsymbol{w}^*)/N}, \qquad E_{RMS} = \sqrt{\frac{1}{\tilde{N}} \sum_{n=1}^{\tilde{N}} \left(y(\tilde{x}_n, \boldsymbol{w}^*) - \tilde{t}_n\right)^2}$$



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.

	M=0	M=1	M=3	M=9
w_0^*	0.19	0.82	0.31	0.35
w_1^*		-1.27	8	232
w_2^*			-25	5321
w_3^*			-17	48568
w_4^*				-231639
w_5^*				640042
w_6^*				-10618000
w_7^*				10424000
w_8^*				-557683
w_9^*				-125201

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 $\Omega(\boldsymbol{w})$ to penalize complexity.

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

For example,

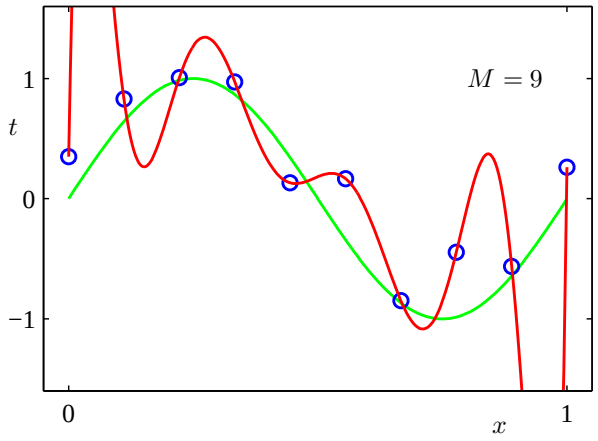
$$\tilde{E}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(y(x_n, \boldsymbol{w}) - t_n\right)^2 + \frac{\lambda}{2}||\boldsymbol{w}||^2$$

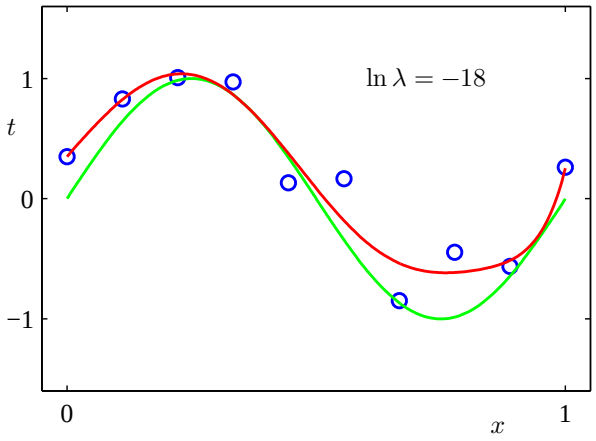
(here, $||\boldsymbol{w}||^2 := \sum_{m=0}^{M} w_m^2$)

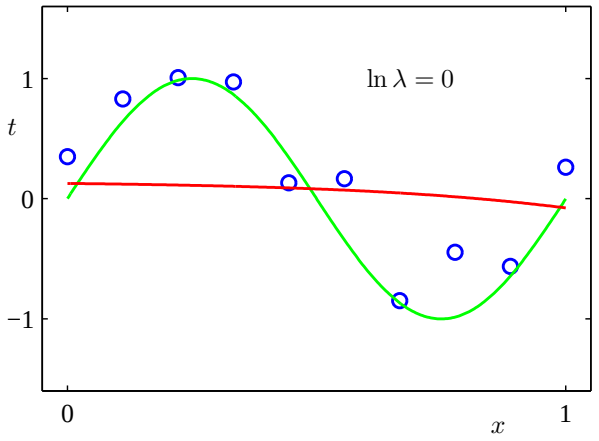
Weight decay = shrinkage = ridge regression

Penalty term independent of number of training data

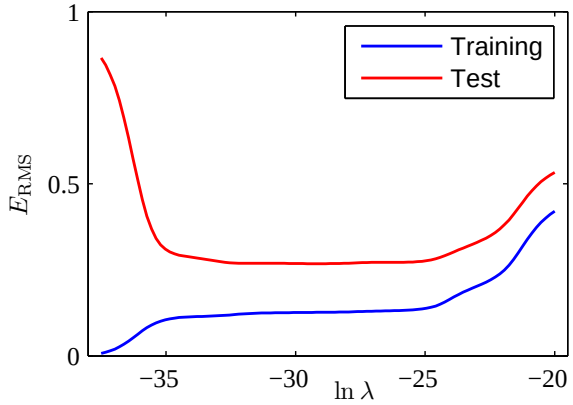
- small data sets: penalty term relatively large
- large data sets: penalty term relatively small







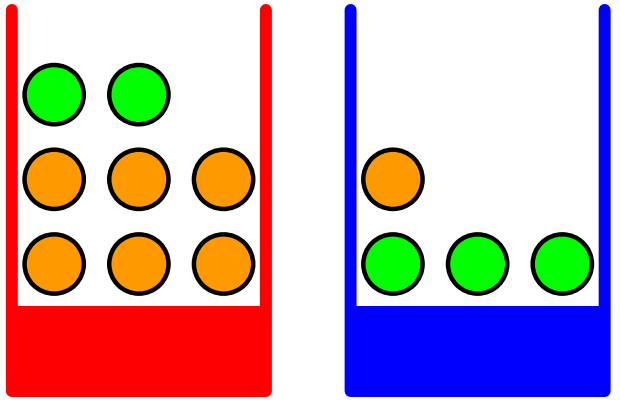
	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
w_0^*	0.35	0.35	0.13
w_1^*	232	4.74	-0.05
w_2^*	5321	- 0.77	-0.06
w_3^*	48568	-31.97	-0.05
w_4^*	-231639	- 3.89	-0.03
w_5^*	640042	55.28	-0.02
w_6^*	-10618000	41.32	-0.01
w_7^*	10424000	-45.95	-0.00
w_8^*	-557683	-91.53	0.00
w_9^*	-125201	72.68	0.01



- *Training set* to optimize (typically many) parameters $\boldsymbol{w}$
- *Validation set* to optimize (typically a few) hyperparameters (λ 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 ($F = \{a, o\}$).

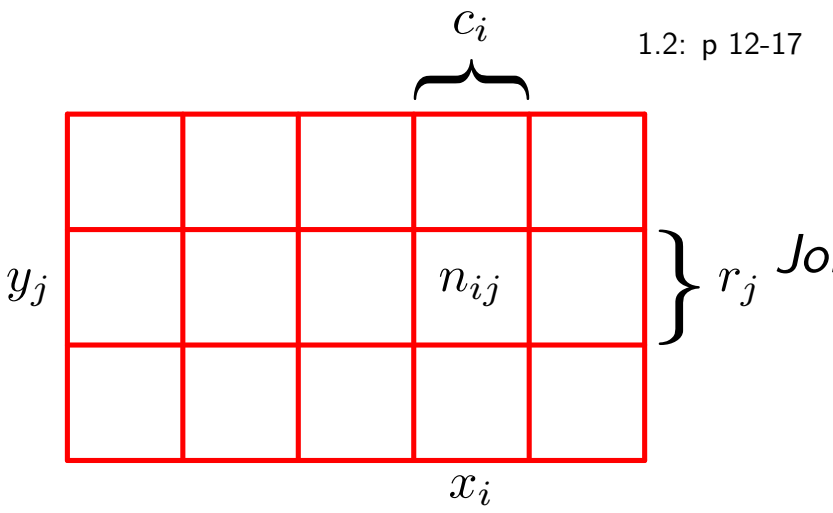
• Consider an experiment of (infinitely) many (mentally) repeated trials
  (*randomly pick a box, then randomly select an item of fruit from that box*)
under the same macroscopic conditions
  (*number of red/blue boxes and apples/oranges balls in the boxes*)
but each time with different microscopic details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

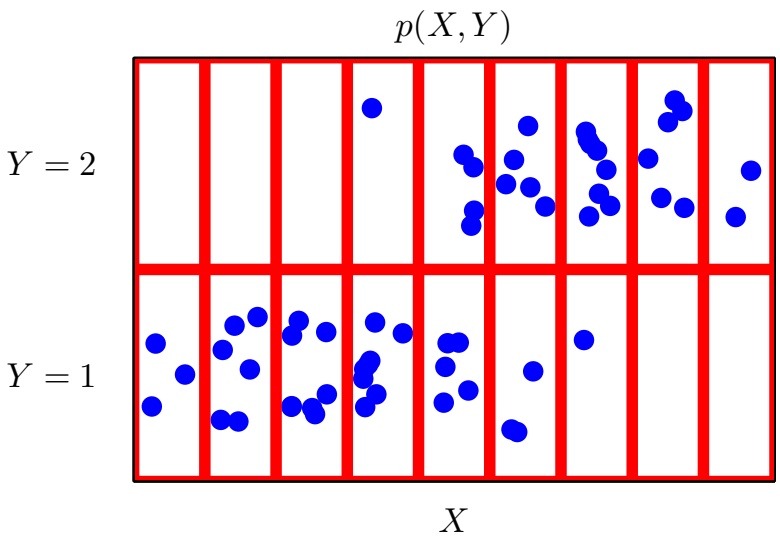
Sum rule	$p(X) = \sum_Y p(X, Y)$
Product rule	$p(X, Y) = p(Y\|X)p(X)$
Positivity	$p(X) \geq 0$
Normalization	$\sum_X p(X) = 1$

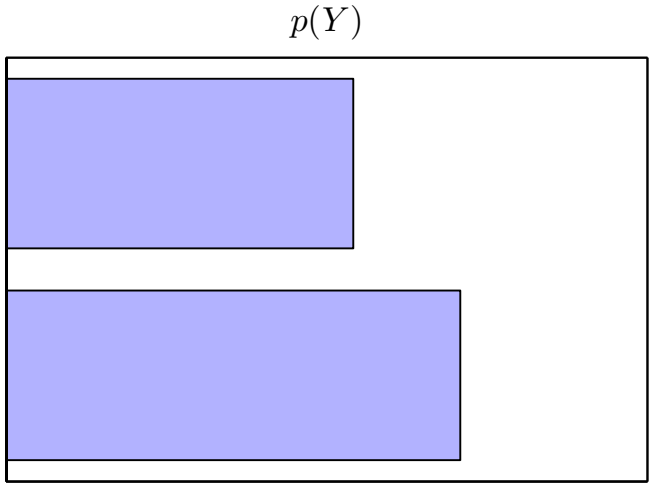
$$p(X,Y) \;=\; p(Y|X)p(X) \;=\; P(X|Y)p(Y) \quad \Rightarrow$$

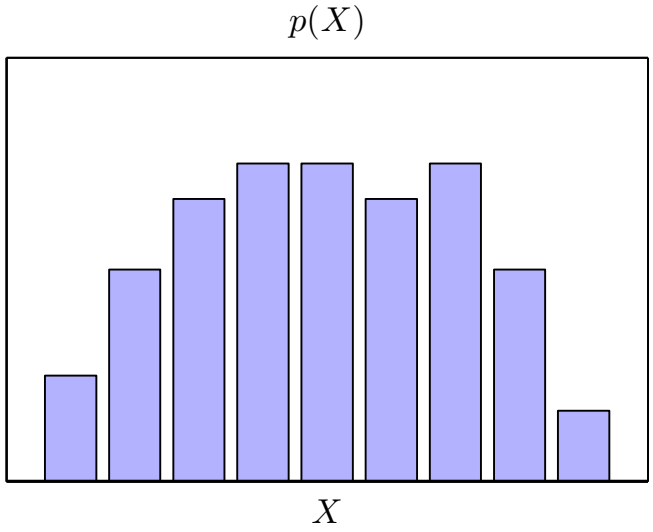
Bayes' theorem

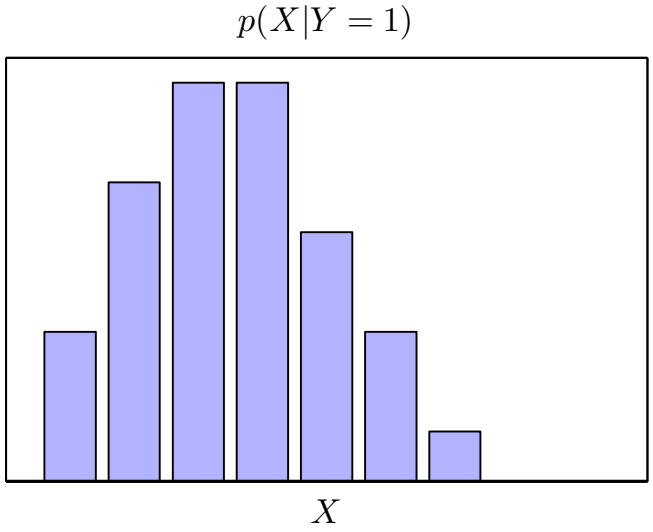
$$p(Y|X) \;=\; \frac{p(X|Y)p(Y)}{p(X)} \qquad \left(\;=\; \frac{p(X|Y)p(Y)}{\sum_Y p(X|Y)p(Y)} \;\right)$$

Bayes' theorem = Bayes' rule


$p(X,Y)$
$Y=2$
$Y=1$
X


$p(Y)$

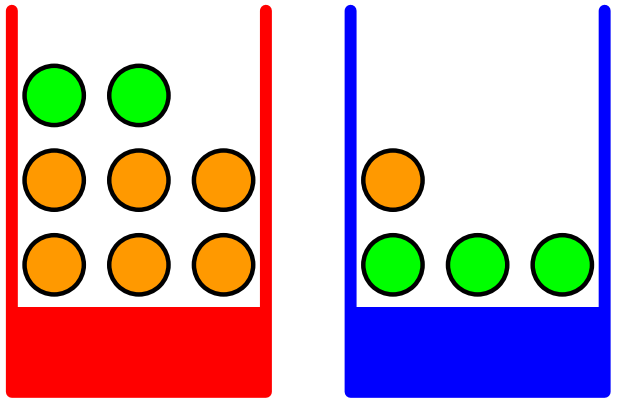

$p(X)$
X


$p(X|Y=1)$
X

Fruits again

Model

$$p(B = r) = 4/10$$
$$p(B = b) = 6/10$$



$$p(F = a|B = r) = 1/4$$
$$p(F = o|B = r) = 3/4$$

$$p(F = a|B = b) = 3/4$$
$$p(F = o|B = b) = 1/4$$

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

$$P(Y|X, Z) = P(Y|Z)$$

two coins

Consider two coins. Coin 1 has probability of head of $1/2$, coin 2 has probability of head $1/4$.

Given the sequence of coin tosses:

$$head, tail, tail$$

What is the probability that I used coin 1?

answer

$$
\begin{aligned}
p(D|coin1) &= \frac{1*1*1}{2*2*2} = \frac{1}{8}, \qquad p(D|coin2) = \frac{1*3*3}{4*4*4} = \frac{9}{64} \\
p(coin1|D) &= \frac{p(D|coin1)p(coin1)}{p(D)} \\
p(D) &= p(D|coin1)p(coin1) + p(D|coin2)p(coin2)
\end{aligned}
$$

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

$$
p(coin1|D) = \frac{p(D|coin1)}{p(D|coin1) + p(D|coin2)} = \frac{8}{8+9} = \frac{8}{17}
$$

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

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

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

$$p(x) \geq 0$$

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

• 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 $\boldsymbol{x} = (x_1, \ldots, x_d)$.
• Probability density $p(\boldsymbol{x})$: probability of $\boldsymbol{x}$ falling in an infinitesimal volume $\delta\boldsymbol{x}$ around $\boldsymbol{x}$ is given by $p(\boldsymbol{x})\delta\boldsymbol{x}$.

$$\text{Prob}(x \in \mathcal{R}) \;=\; \int_{\mathcal{R}} p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \;=\; \int_{\mathcal{R}} p(x_1, \ldots, x_d)\mathrm{d}x_1\mathrm{d}x_2 \ldots \mathrm{d}x_d$$

and

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

$$\int p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = 1$$

• Rules of probability apply to continuous variables as well,

$$p(x) = \int p(x, y)\,\mathrm{d}y$$

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

Expectations and variances

Expectations

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

$$\mathbb{E}[f] = \int_x p(x) f(x)\, \mathrm{d}x \qquad \text{continuous var's}$$

Variance:

$$\mathrm{var}[f] = \big\langle f(x)^2 \big\rangle - \big\langle f(x) \big\rangle^2$$

$$\mathrm{var}[x] = \big\langle x^2 \big\rangle - \big\langle x \big\rangle^2$$

Covariance

Covariance of two random variables

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

Covariance matrix of the components of a vector variable

$$\mathrm{cov}[\boldsymbol{x}] \equiv \mathrm{cov}[\boldsymbol{x}, \boldsymbol{x}] = \left\langle \boldsymbol{x}\boldsymbol{x}^T \right\rangle - \left\langle \boldsymbol{x} \right\rangle \left\langle \boldsymbol{x}^T \right\rangle$$

with components

$$(\mathrm{cov}[\boldsymbol{x}])_{ij} = \mathrm{cov}[x_i, x_j] = \left\langle x_i x_j \right\rangle - \left\langle x_i \right\rangle \left\langle x_j \right\rangle$$

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 neccessarily 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 $B(x\,|\,y)$.

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 $x\,|\,y$ and the degree of belief in the proposition y. There is a function g such that

$$B(x, y) = g\left[B(x\,|\,y), B(y)\right].$$

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

$$P(x, y) = P(x\,|\,y)P(y).$$

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?

$$p(w=1|x=1) = \frac{0.9*0.01}{0.9*0.01+0.1*0.99} = \frac{1}{1+\frac{0.1*0.99}{0.9*0.01}} = \frac{1}{12} = 0.0825$$

Maximum likelihood estimation

Given a data set

$$\text{Data} = \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N\}$$

and a parametrized distribution

$$p(\boldsymbol{x}|\boldsymbol{w}), \quad \boldsymbol{w} = (w_1, \ldots, w_M),$$

find the value of $\boldsymbol{w}$ that best describes the data.

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

$$p(\text{Data}|\boldsymbol{w}) = p(\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N\}|\boldsymbol{w}) = \prod_{i=1}^{N} p(\boldsymbol{x}^i|\boldsymbol{w})$$

Then, the most likely $\boldsymbol{w}$ is obtained by maximizing $p(\text{Data}|\boldsymbol{w})$ wrt $\boldsymbol{w}$:

$$\begin{aligned} \boldsymbol{w}_{\text{ML}} &= \text{argmax}_{\boldsymbol{w}} p(\text{Data}|\boldsymbol{w}) = \text{argmax}_{\boldsymbol{w}} \prod_{i=1}^{N} p(\boldsymbol{x}^i|\boldsymbol{w}) \\ &= \text{argmax}_{\boldsymbol{w}} \sum_{i=1}^{N} \log p(\boldsymbol{x}^i|\boldsymbol{w}) \end{aligned}$$

since $\log$ is a monotonically increasing function.

$\boldsymbol{w}_{\text{ML}}$ is a function of the data. This is called an *estimator*.

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

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

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 $\boldsymbol{w}$: uncertain

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

Apply Bayes' rule to obtain the *posterior* distribution

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

$$\begin{aligned} p(\boldsymbol{w}) &: \text{prior} \\ p(\text{Data}|\boldsymbol{w}) &: \text{likelihood} \\ p(\text{Data}) = \int p(\text{Data}|\boldsymbol{w})p(\boldsymbol{w})\, d\boldsymbol{w} &: \text{evidence} \\ \rightarrow \quad p(\boldsymbol{w}|\text{Data}) &: \text{posterior} \end{aligned}$$

Predictive distribution

Prior to 'learning', the predictive distribution for new observation $\boldsymbol{x}$ is

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}|\boldsymbol{w})p(\boldsymbol{w})\, d\boldsymbol{w}$$

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

$$\begin{aligned} p(\boldsymbol{x}|\text{Data}) &= \int p(\boldsymbol{x}|\boldsymbol{w}, \text{Data})p(\boldsymbol{w}|\text{Data})\, d\boldsymbol{w} \\ &= \int p(\boldsymbol{x}|\boldsymbol{w})p(\boldsymbol{w}|\text{Data})\, d\boldsymbol{w} \end{aligned}$$

Bayesian vs frequentists view point

- Bayesians: there is a single fixed dataset (the one observed), and a probability distribution of model parameters $\boldsymbol{w}$ which expresses a subjective belief including uncertainty about the 'true' model parameters.

  - They need a prior belief $p(\boldsymbol{w})$, and apply Bayes rule to compute $p(\boldsymbol{w}|\text{Data})$.

  - Bayesians can talk about frequency that in repeated situations with this data $\boldsymbol{w}$ 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 $p(\text{Data}|\boldsymbol{w})$, such as bias and variance.

  -They cannot make a claim for this particular data set. This is the price for not having a 'prior'.

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:

$$\hat{w} = \mathrm{argmax}_w p(N_H|w,N) = \mathrm{argmax}_w \left[N_H \log w + (N-N_H)\log(1-w) \right] = \frac{N_H}{N}$$

$\hat{w}$ 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

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

The Bayesian approach considers one data set and assumes a prior $p(w)$ and compute 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 head, 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 mechanstic procedure (Bayesian machinery) and results have a clear interpretation.

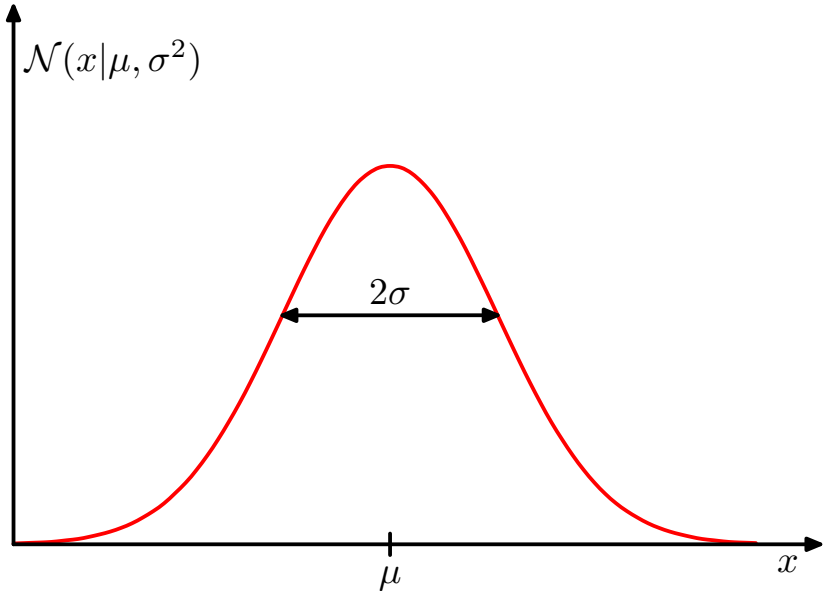
This course place emphasis on Bayesian approach.

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 μ and σ^2



Gaussian is normalized,

$$\int_{-\infty}^{\infty} \mathcal{N}(x|\mu, \sigma^2)\,\mathrm{d}x = 1$$

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

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

$$\left\langle x^2 \right\rangle = \int_{-\infty}^{\infty} x^2\mathcal{N}(x|\mu, \sigma^2)\,\mathrm{d}x = \mu^2 + \sigma^2$$

$$\mathrm{var}[x] = \left\langle x^2 \right\rangle - \langle x \rangle^2 = \sigma^2$$

Multivariate Gaussian

In D dimensions

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

$\boldsymbol{x}, \mu$ are D-dimensional vectors.

$\boldsymbol{\Sigma}$ is a $D \times D$ covariance matrix, $|\boldsymbol{\Sigma}|$ is its determinant.

Mean vector and covariance matrix

$$\mathbb{E}[\boldsymbol{x}] = \langle \boldsymbol{x} \rangle = \int \boldsymbol{x} \mathcal{N}(\boldsymbol{x}|\mu, \boldsymbol{\Sigma}) \, \mathrm{d}\boldsymbol{x} \; = \; \mu$$

$$\mathrm{cov}[\boldsymbol{x}] = \left\langle (\boldsymbol{x} - \mu)(\boldsymbol{x} - \mu)^T \right\rangle = \int (\boldsymbol{x} - \mu)(\boldsymbol{x} - \mu)^T \mathcal{N}(\boldsymbol{x}|\mu, \boldsymbol{\Sigma}) \, \mathrm{d}\boldsymbol{x} \; = \; \boldsymbol{\Sigma}$$

We can also write this in component notation:

$$\begin{aligned} \mu_i &= \langle x_i \rangle = \int x_i \mathcal{N}(\boldsymbol{x}|\mu, \boldsymbol{\Sigma}) \, \mathrm{d}\boldsymbol{x} \\ \Sigma_{ij} &= \langle (x_i - \mu_i)(x_j - \mu_j) \rangle = \int (x_i - \mu_i)(x_j - \mu_j) \mathcal{N}(\boldsymbol{x}|\mu, \boldsymbol{\Sigma}) \, \mathrm{d}\boldsymbol{x} \end{aligned}$$

$\mathcal{N}(\boldsymbol{x}|\mu, \boldsymbol{\Sigma})$ 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

$$\int_{y=c}^{d} f(y)\,\mathrm{d}y$$

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

$$\begin{aligned}\int_{y=c}^{d} f(y)\,\mathrm{d}y &= \int_{y(x)=c}^{d} f(y(x))\,\mathrm{d}y(x) \\ &= \int_{x=a}^{b} f(y(x))y'(x)\,\mathrm{d}x\end{aligned}$$

The derivative $y'(x)$ comes in as the ratio between the lengths of the differentials $\mathrm{d}y$ and $\mathrm{d}x$,

$$\mathrm{d}y = y'(x)\,\mathrm{d}x$$

Several variables

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

$$\begin{aligned}\int_{\boldsymbol{y}\in\mathcal{S}} f(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y} &= \int_{\boldsymbol{y}(\boldsymbol{x})\in\mathcal{S}} f(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y}(\boldsymbol{x}) \\ &= \int_{\boldsymbol{x}\in\mathcal{R}} f(\boldsymbol{y}(\boldsymbol{x}))\left|\det\left(\frac{\partial \boldsymbol{y}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)\right|\,\mathrm{d}\boldsymbol{x}\end{aligned}$$

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

$$\det\left(\frac{\partial \boldsymbol{y}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right) = \begin{vmatrix} \frac{\partial y_1(\boldsymbol{x})}{\partial x_1} & \frac{\partial y_1(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial y_1(\boldsymbol{x})}{\partial x_n} \\ \frac{\partial y_2(\boldsymbol{x})}{\partial x_1} & \frac{\partial y_2(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial y_2(\boldsymbol{x})}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial y_n(\boldsymbol{x})}{\partial x_1} & \frac{\partial y_n(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial y_n(\boldsymbol{x})}{\partial x_n} \end{vmatrix}$$

The *absolute value* [2] of the Jacobian comes in as the ratio between that the volume represented by the differential $\mathrm{d}\boldsymbol{y}$ and the volume represented by the differential $\mathrm{d}\boldsymbol{x}$, i.e.,

$$\mathrm{d}\boldsymbol{y} = \left|\det\left(\frac{\partial \boldsymbol{y}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)\right| \mathrm{d}\boldsymbol{x}$$

As a last remark, it is good to know that

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

[2] In the single-variable case, we took the orientation of the integration interval into account ($\int_a^b f(x)\,\mathrm{d}x = -\int_b^a f(x)\,\mathrm{d}x$). 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 (= 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

$$\int_D \mathrm{d}x\mathrm{d}y$$

This integral is easiest evaluated by going to 'polar-coordinates'. The mapping from polar coordinates (r, θ) to Cartesian coordinates (x, y) is

$$x = r\cos\theta \quad (1)$$

$$y = r\sin\theta \quad (2)$$

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,

$$\mathrm{d}x\mathrm{d}y = r\mathrm{d}r\mathrm{d}\theta$$

The area of the disc is now easily evaluated.

$$\int_D \mathrm{d}x\mathrm{d}y = \int_{\theta=0}^{2\pi}\int_{r=0}^{R} r\mathrm{d}r\mathrm{d}\theta = \pi R^2$$

Transformation of densities

When $f(\boldsymbol{y}) = p_y(\boldsymbol{y})$ a normalized probability density, then

$$\begin{aligned} \int f(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y} &= \int f(\boldsymbol{y}(\boldsymbol{x}))\left|\det\left(\frac{\partial \boldsymbol{y}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)\right|\,\mathrm{d}\boldsymbol{x} \\ \int p_y(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y} &= \int p_x(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \end{aligned}$$

Probability densities:

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

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

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

Gaussian integral

How to compute

$$\int_{-\infty}^{\infty} \exp(-x^2)\mathrm{d}x$$

$$\begin{aligned}\Big(\int_{-\infty}^{\infty} \exp(-x^2)\mathrm{d}x\Big)^2 &= \int_{-\infty}^{\infty} \exp(-x^2)\mathrm{d}x \int_{-\infty}^{\infty} \exp(-y^2)\mathrm{d}y \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \exp(-x^2)\exp(-y^2)\mathrm{d}x\mathrm{d}y \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \exp(-(x^2+y^2))\mathrm{d}x\mathrm{d}y\end{aligned}$$

The latter is easily evaluated by going to polar-coordinates,

$$\begin{aligned}
\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\exp(-(x^2+y^2))\mathrm{d}x\mathrm{d}y &= \int_{\theta=0}^{2\pi}\int_{r=0}^{\infty}\exp(-r^2)r\mathrm{d}r\mathrm{d}\theta \\
&= 2\pi\int_{r=0}^{\infty}\exp(-r^2)r\mathrm{d}r \\
&= 2\pi\times\big(-\frac{1}{2}\exp(-r^2)\big)\Big|_0^{\infty} \\
&= \pi
\end{aligned}$$

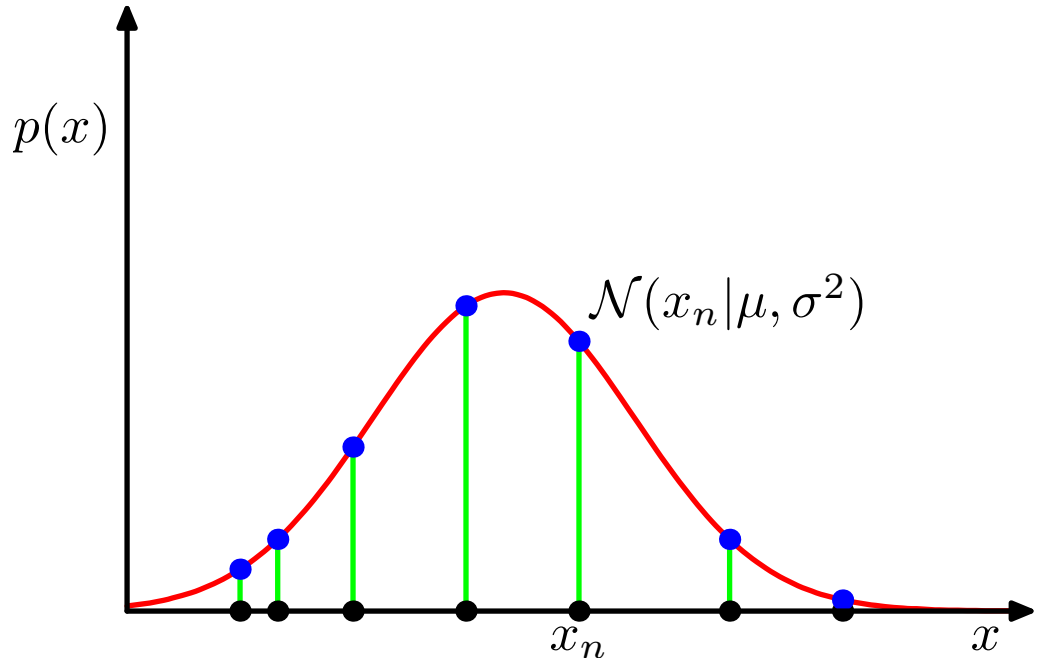
So

$$\int_{-\infty}^{\infty}\exp(-x^2)\mathrm{d}x = \sqrt{\pi}$$

The likelihood for the 1-d Gaussian

Consider 1-d data 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:

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

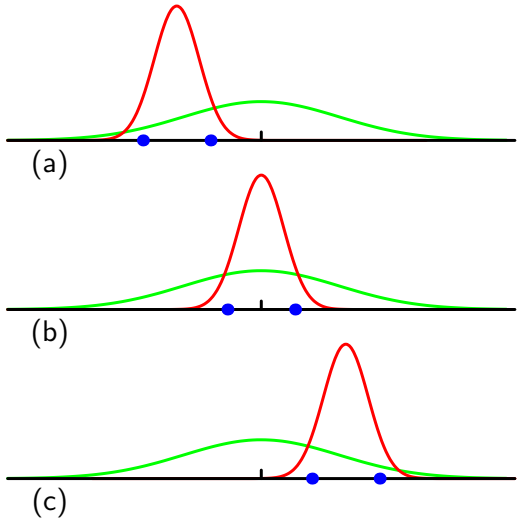
The values of μ and σ that maximize the likelihood are given by (Ex. B1.11)

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

Bias in the ML estimates

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

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



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.

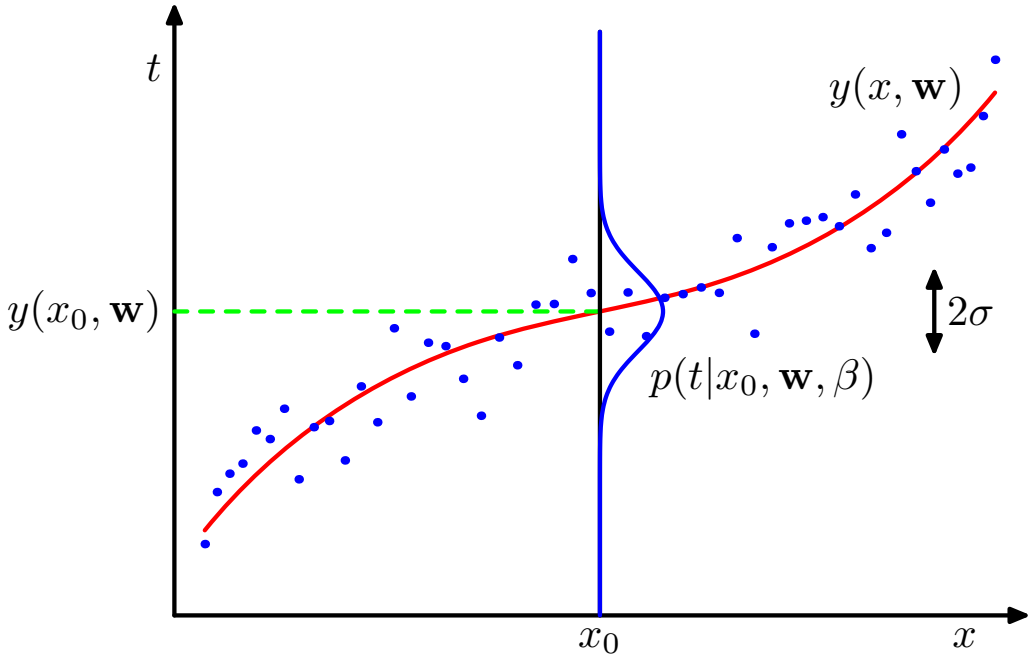
Curve fitting re-visited

Now from a probabilistic perspective.

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

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

$\beta = 1/\sigma^2$ is the precision.



Curve fitting re-visited: ML

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

Likelihood ,

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

Log-likelihood

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

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

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

Curve fitting re-visited MAP

Bayesian approach.
Prior:

$$p(\boldsymbol{w}|\alpha) = \mathcal{N}(\boldsymbol{w}|\alpha^{-1}\boldsymbol{I}) = \left(\frac{\alpha}{2\pi}\right)^{(M+1)/2} \exp\left(-\frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w}\right)$$

M is the dimension of $\boldsymbol{w}$. Variables such as α, controling the distribution of parameters, are called 'hyperparameters'.

Posterior using Bayes rule:

$$\begin{aligned} p(\boldsymbol{w}|\mathbf{t},\mathbf{x},\alpha,\beta) &\propto p(\boldsymbol{w}|\alpha)\prod_{n=1}^{N}\mathcal{N}(t_n|y(x_n,\boldsymbol{w}),\beta^{-1}) \\ -\ln p(\boldsymbol{w}|\mathbf{t},\mathbf{x},\alpha,\beta) &= \frac{\beta}{2}\sum_{n=1}^{N}(y(x_n,\boldsymbol{w})-t_n)^2 + \frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w} + \text{const}(\beta) \end{aligned}$$

Maximizing the posterior wrt $\boldsymbol{w}$ yields $\boldsymbol{w}_{MAP}$. Similar as Eq. 1.4. with $\lambda = \alpha/\beta$

Bayesian curve fitting

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

$$p(t|x, \mathbf{x}, \mathbf{t}) = \int d\boldsymbol{w} p(t|x, \boldsymbol{w}) p(\boldsymbol{w}|\mathbf{x}, \mathbf{t})$$

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

All quantities depend on α and β.

Bayesian curve fitting

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

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

So: prediction given $\boldsymbol{w}$ is

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

The prediction given $\mathbf{x}, \mathbf{t}$ is

$$p(t|x, \mathbf{x}, \mathbf{t}) = \int d\boldsymbol{w} p(t|x, \boldsymbol{w}) p(\boldsymbol{w}|\mathbf{x}, \mathbf{t}, \alpha, \beta)$$

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).

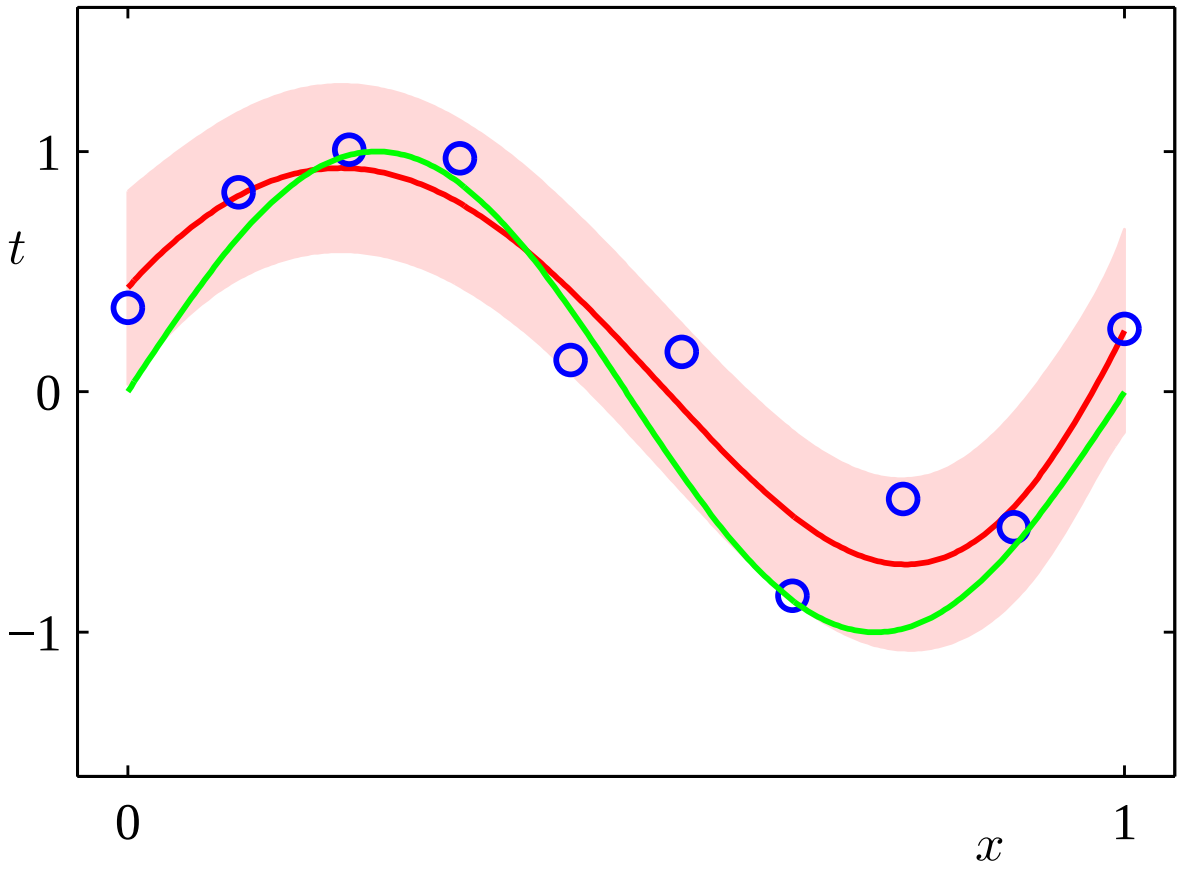
Result Bayesian curve fitting

The result is (section 3.3).

$$
\begin{aligned}
p(t|x, \mathbf{x}, \mathbf{t}) &= \mathcal{N}(t|m(x), s^2(x)) \\
m(x) &= \beta \phi(x)^T \boldsymbol{S} \sum_{n=1}^{N} \phi(x_n) t_n = \phi(x)^T \boldsymbol{w}_{MAP} \\
s^2(x) &= \beta^{-1} + \phi(x)^T \boldsymbol{S} \phi(x) \\
\boldsymbol{S}^{-1} &= \alpha \boldsymbol{I} + \beta \sum_{n=1}^{N} \phi(x_n) \phi(x_n)^T
\end{aligned}
$$

Note, s^2 depend on x. First term as in ML estimate describes noise in target for fixed $\boldsymbol{w}$. Second term describes noise due to uncertainty in $\boldsymbol{w}$.

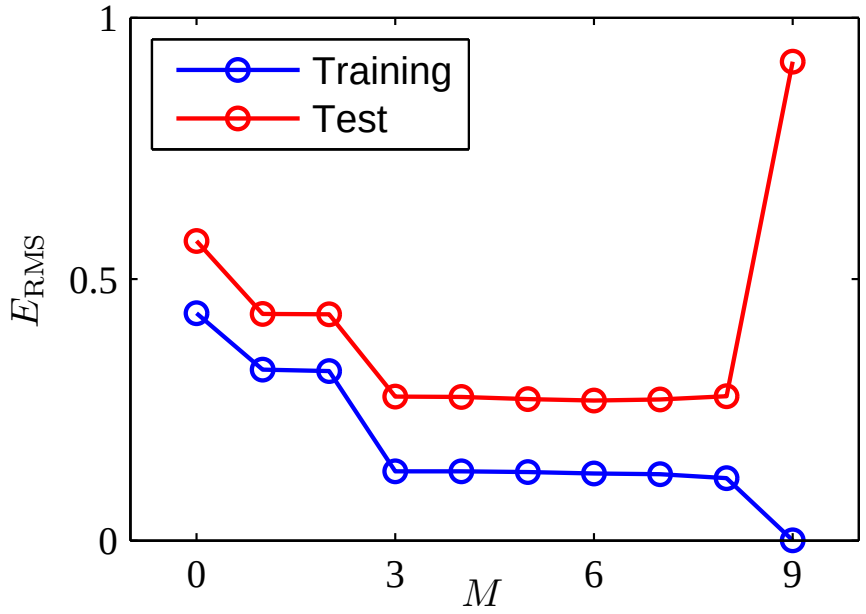
Example



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?

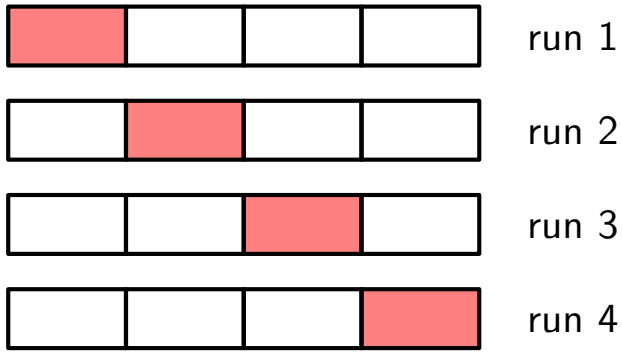


Model selection/Cross validation

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*.

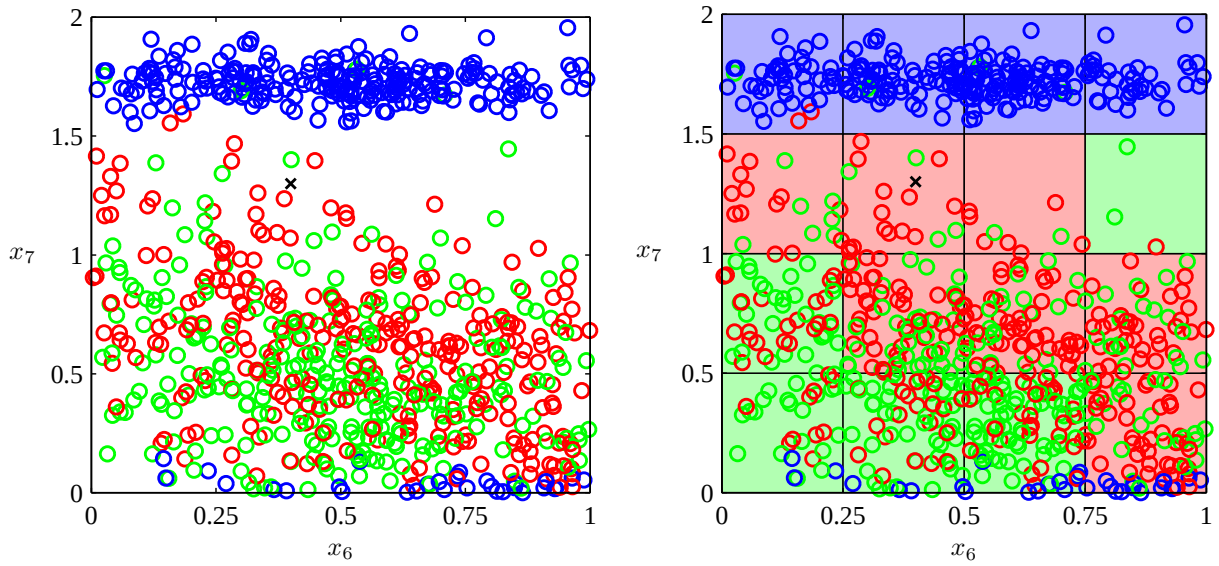


A3: Information criteria: penalty for complex models

- Akaike IC (AIC): $\ln p(D|\boldsymbol{w}_{ML}) - M$
- Bayesian IC (BIC): Bayesian + crude approximations (section 4.4.1)
- Full Bayesian $\rightarrow$ penalties arises automatically (section 3.4)

High-dimensional data/Binning

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).

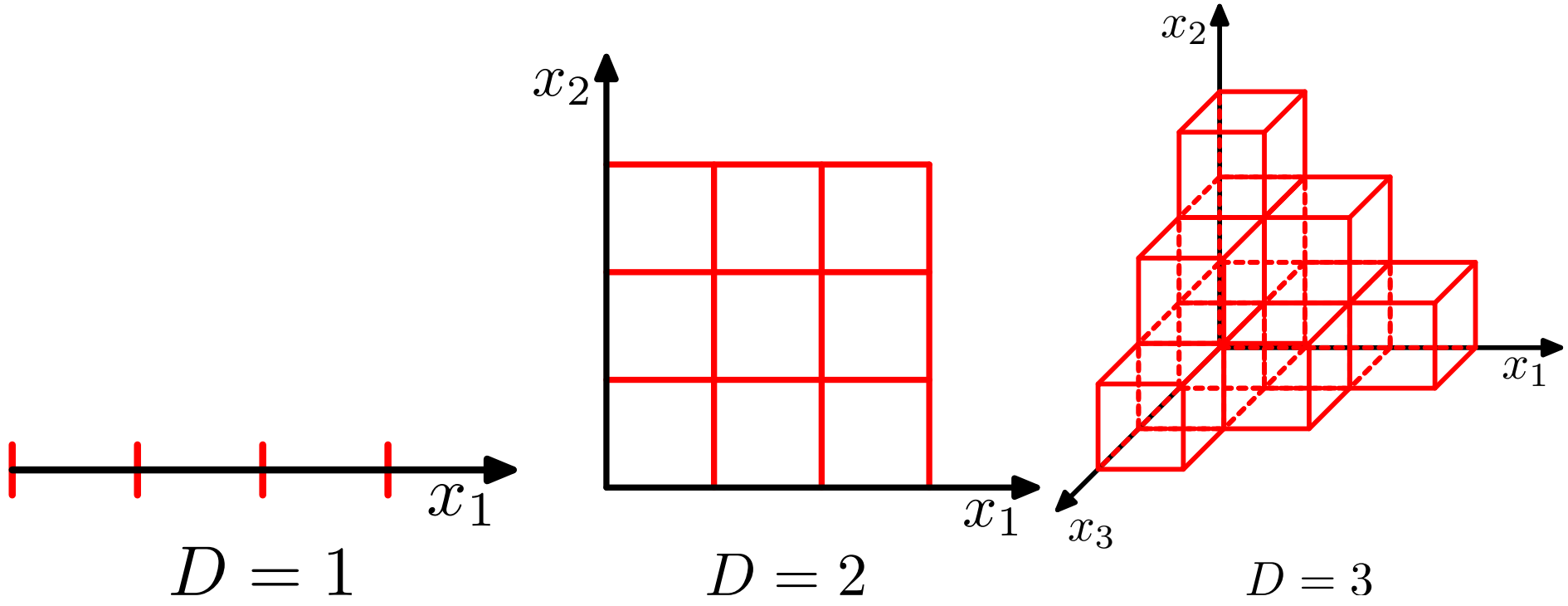
Classification of the 'x' can be done by puting a grid on the space and assigning the class that is most numerous in the particular box.

High-dimensional data/Binning

Q: What is the disadvantage of this approach?

Curse of dimensionality/Binning

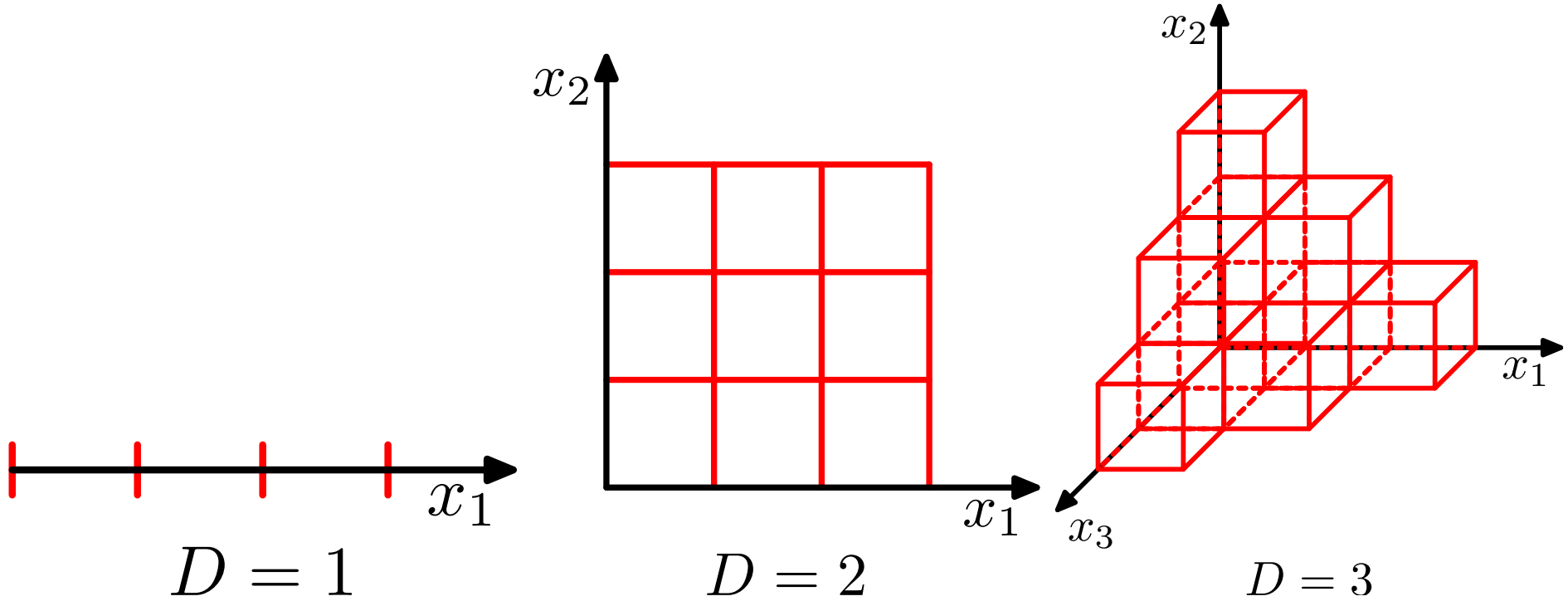
Q: What is the disadvantage of this approach?
A: This approach scales exponentially with dimensions.



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:

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

(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:

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

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

The number of coefficients scales as D^M (unpractically large).

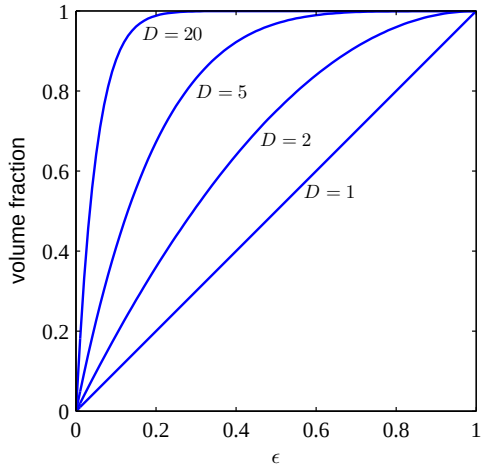
Curse of dimensionality/Spheres

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!

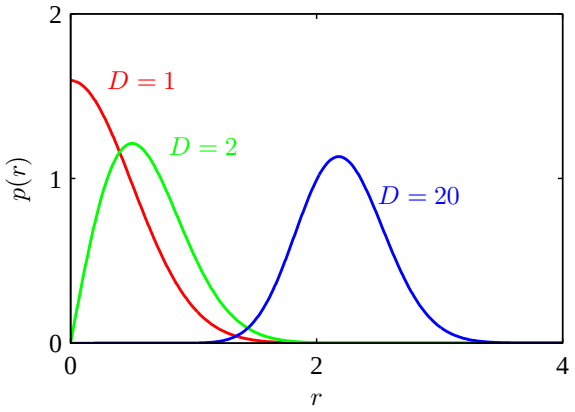


$$V_D(r) = K_D r^D \qquad \frac{V_D(1) - V_D(1-\epsilon)}{V_D(1)} = 1 - (1-\epsilon)^D$$

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 σ is concentrated around a thin shell $r \approx \sigma\sqrt{D-1}$.

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

$$\frac{d}{dr}\left(-\frac{r^2}{2\sigma^2}+(n-1)\log r\right) = 0 \rightarrow r = \sigma\sqrt{n-1}$$

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 corralated way.

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

Decision theory

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

Decision: for new value of $\boldsymbol{x}$ estimate 'best' t.

Example: in a medical application, $\boldsymbol{x}$ 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 $(\boldsymbol{x}, t)$, learn $p(\boldsymbol{x}, t)$ and estimate $p(\boldsymbol{x}, t)$ for new value of $\boldsymbol{x}$ (and possibly all t).

Decision: for new value of $\boldsymbol{x}$ estimate 'best' t.

Example: in a medical application, $\boldsymbol{x}$ 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:

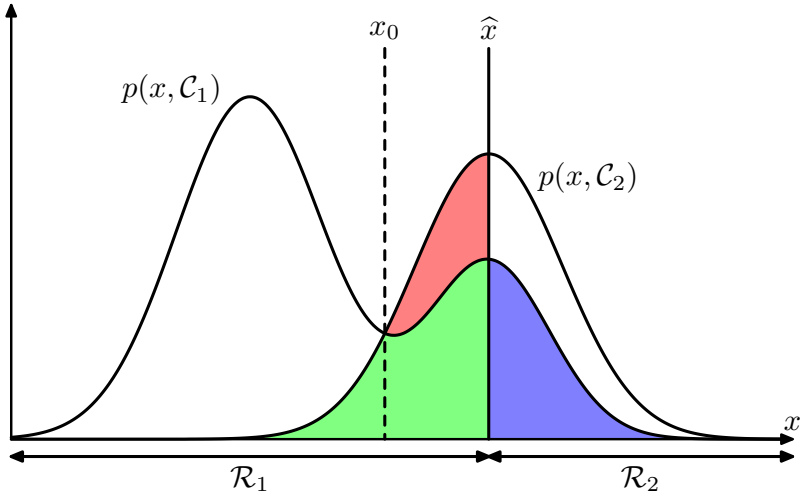
$$p(C_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|C_k)p(C_k)}{p(\boldsymbol{x})}$$

$p(C_k)$ is the prior probability of class C_k. $p(C_k|\boldsymbol{x})$ is the posterior probability of class C_k after seeing the image $\boldsymbol{x}$.

Decision theory

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

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



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

$$k = \text{argmax}_k p(\boldsymbol{x}, C_k) = \text{argmax}_k p(C_k|\boldsymbol{x})$$

Decision theory/Example

Example: in a medical application, $\boldsymbol{x}$ 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 $p(C_1|\boldsymbol{x}) = 0.3$ according to our model. Do we decide that the patient has cancer and therefore start treatment?

Decision theory/Example

Example: in a medical application, $\boldsymbol{x}$ 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 $p(C_1|\boldsymbol{x}) = 0.3$ 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 $p(C_k|\boldsymbol{x})$ is maximal.
Because $p(C_1|\boldsymbol{x}) = 0.3$ and $p(C_2|\boldsymbol{x}) = 0.7$, 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.

	C_1	C_2
C_1	0	1000
C_2	1	0

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

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

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

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

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

Decision theory/Example

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

	C_1	C_2
C_1	0	1000
C_2	1	0

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

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

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

Decision theory/Example

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

	C_1	C_2
C_1	0	1000
C_2	1	0

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

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

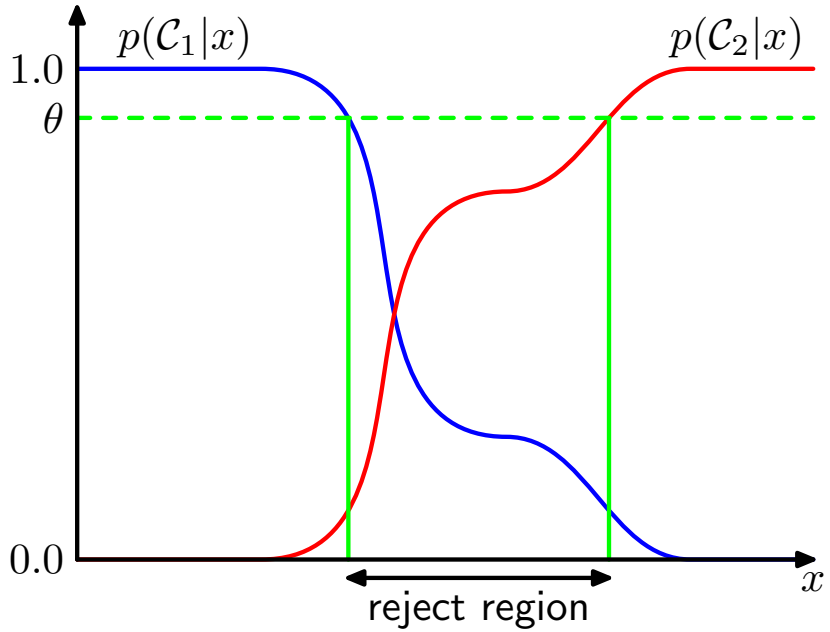
Q: Suppose $p(C_1) = 0.01$ and $p(C_1|\boldsymbol{x}) = 0.3$ according to our model. Do we decide that the patient has cancer and therefore start treatment?

A: If we want to minimize the expected loss, we have to pick j such that $\sum_k L_{kj} p(C_k|\boldsymbol{x})$ is minimal.
For $j = 1$, this yields $\sum_k L_{kj} p(C_k|\boldsymbol{x}) = 0 \times 0.3 + 1 \times 0.7 = 0.7$,
for $j = 2$, this yields $\sum_k L_{kj} p(C_k|\boldsymbol{x}) = 1000 \times 0.3 + 0 \times 0.7 = 300$.
Therefore, we now decide that the patient has cancer (better safe than sorry).

Decision theory/Reject option

It may be that $\max_j p(C_j|\boldsymbol{x})$ is small, indicating that it is unclear to which class $\boldsymbol{x}$ 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 $\boldsymbol{x}$ for which $\max_j p(C_j|\boldsymbol{x}) > \theta$ (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.

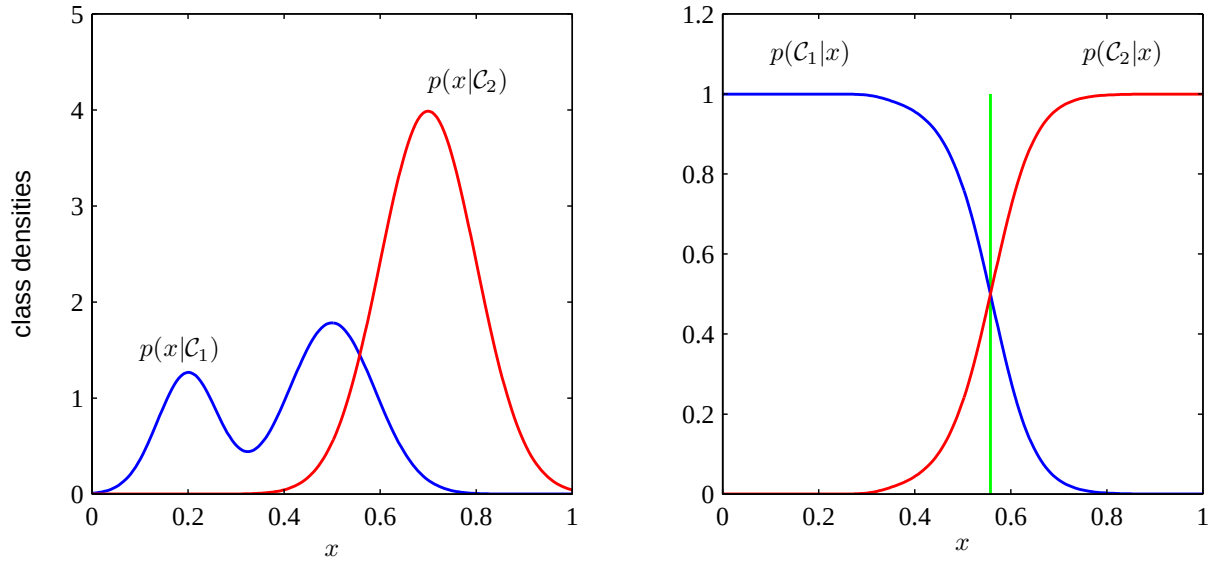
A set of approaches:

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

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

- Learn $f(\boldsymbol{x})$, called a discriminant function, that maps $\boldsymbol{x}$ directly onto a class label $0, 1, 2, \ldots$. Even simpler, only decision boundary is learned. But information on the expected classification error is lost (section 4.1, not treated).

Decision theory/Discriminant functions



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.

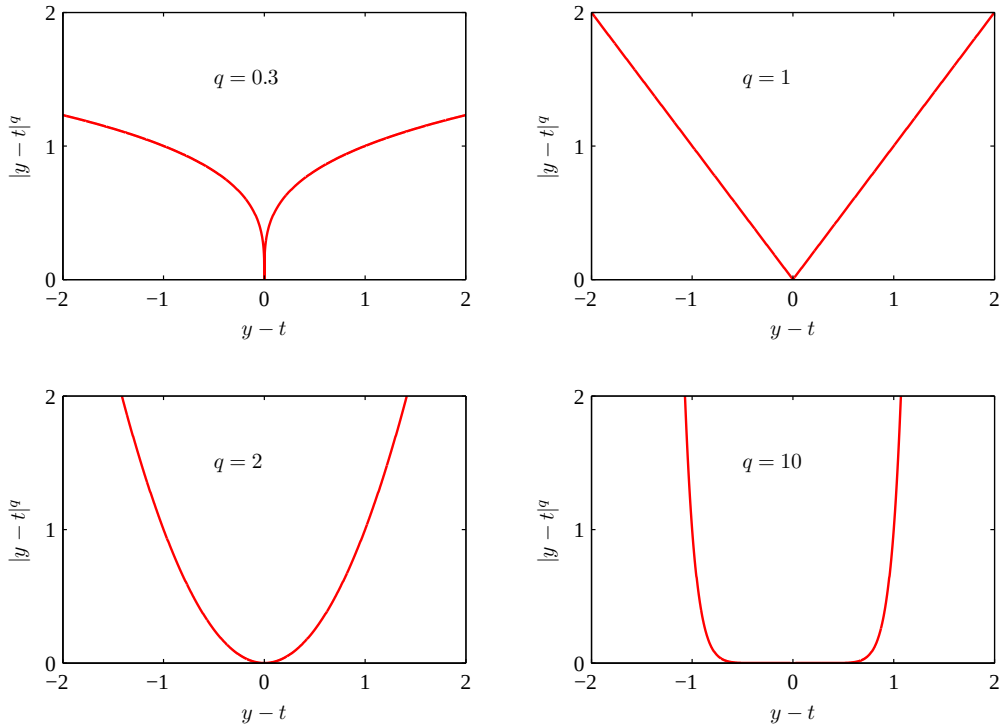
Approaches that model the distribution of both inputs and outputs are called *generative models*, approaches that only model the conditional distribution of the output given the input are called *discriminative models*.

Loss functions for regression

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

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

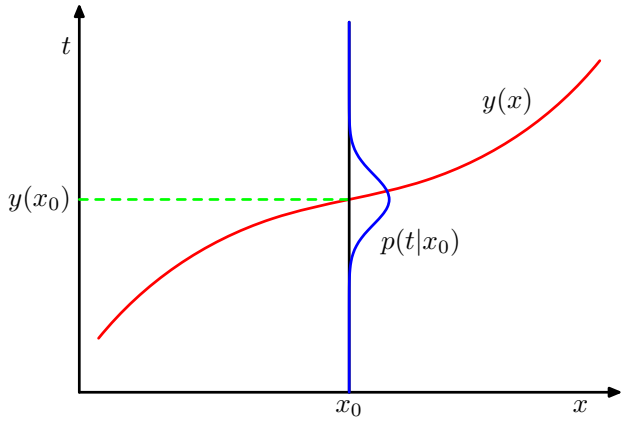
The average/expected loss is:

$$\langle L \rangle \quad = \quad \int L\big(t, y(\boldsymbol{x})\big) p(\boldsymbol{x}, t)\, d\boldsymbol{x}\, dt$$

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

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

i.e., by the mean of the conditional distribution $p(t|\boldsymbol{x})$. (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

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

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).

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 $(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64})$. Then

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

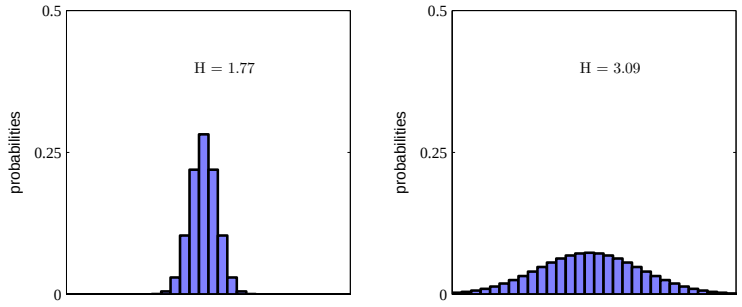
which is smaller than for the uniform distribution.

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

$$\text{Av.codelength} = \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{16} \times 4 + \frac{4}{64} \times 6 = 2\text{bits}$$

Information theory



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 $p(\boldsymbol{x})$ a distribution density over a continuous value $\boldsymbol{x}$ we define the (differential) entropy as

$$H[\boldsymbol{x}] = -\int p(\boldsymbol{x}) \log p(\boldsymbol{x}) d\boldsymbol{x}$$

Suppose that all we know about p is its mean μ and its variance σ^2.

Q: What is the distribution p with mean μ and variance σ^2 that is as *uninformative* as possible, i.e., which maximizes the entropy?

Information theory/Maximum entropy

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

$$H[\boldsymbol{x}] = -\int p(\boldsymbol{x}) \log p(\boldsymbol{x}) d\boldsymbol{x}$$

Suppose that all we know about p is its mean μ and its variance σ^2.

Q: What is the distribution p with mean μ and variance σ^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).

Lagrange multipliers

Minimize $f(\boldsymbol{x})$ under constraint: $g(\boldsymbol{x}) = 0$.

Fancy formulation: define *Lagrangian*,

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

λ is called a Lagrange multiplier.

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

$$\begin{aligned} \max_\lambda L(\boldsymbol{x}, \lambda) &= f(\boldsymbol{x}) && \text{if } \; g(\boldsymbol{x}) = 0 \\ \max_\lambda L(\boldsymbol{x}, \lambda) &= \infty && \text{otherwise} \end{aligned}$$

Lagrange multipliers

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

$$\min_{\boldsymbol{x}} \max_{\lambda} L(\boldsymbol{x}, \lambda) = \max_{\lambda} \min_{\boldsymbol{x}} L(\boldsymbol{x}, \lambda)$$

Procedure:

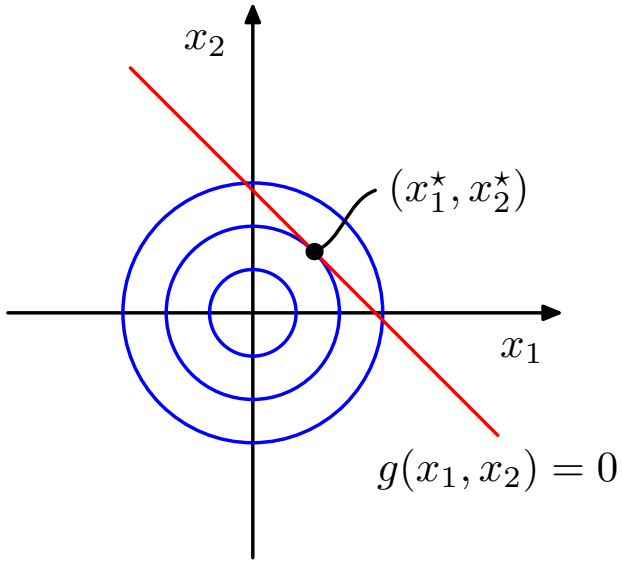
1. Minimize $L(\boldsymbol{x}, \lambda)$ w.r.t $\boldsymbol{x}$, e.g. by taking the gradient and set to zero. This yields a (parametrized) solution $\boldsymbol{x}(\lambda)$.

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

3. The solution of the constraint optimization problem is

$$\boldsymbol{x}^* = \boldsymbol{x}(\lambda^*)$$

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 $x_1^* = x_2^* \frac{1}{2}$

Some remarks

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

- The sign in front of the λ can be chosen as you want:

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

  work equally well.

- More constraints? For each constraint $g_i(\boldsymbol{x}) = 0$ a Lagrange multiplier λ_i, so

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

- Similar methods apply for inequality constraints $g(\boldsymbol{x}) \geq 0$ (restricts λ).

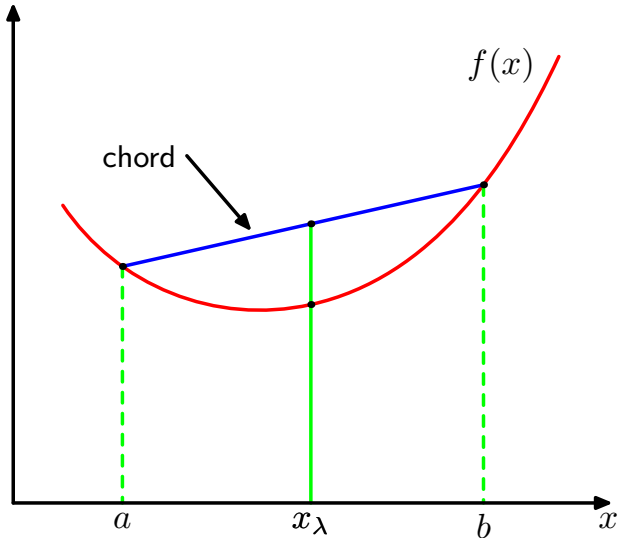
Information theory/KL-divergence

Relative entropy or *Kullback-Leibler divergence* or *KL-divergence*:

$$\begin{aligned} \mathrm{KL}(p||q) &= -\sum_i p_i \ln q_i - \big(-\sum_i p_i \ln p_i\big) \\ &= -\sum_i p_i \ln \{\frac{q_i}{p_i}\} \end{aligned}$$

- 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.
- $\mathrm{KL}(p||q) \neq \mathrm{KL}(q||p)$
- $\mathrm{KL}(p||q) \geq 0, \quad \mathrm{KL}(p||q) = 0 \Leftrightarrow p = q$ (use *convex* functions)
- with continuous variables: $\mathrm{KL}(p||q) = -\int p(\boldsymbol{x}) \ln \{\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\} d\boldsymbol{x}$

Convex functions



Convex function: every chord lies on or above the function.

$$f \text{ is convex} \iff f(\lambda a + (1-\lambda)b) \leq \lambda f(a) + (1-\lambda)f(b) \qquad \forall \lambda \in [0,1], \forall a, b$$

- Examples: $f(x) = ax + b$, $f(x) = x^2$, $f(x) = -\ln(x)$ and $f(x) = x\ln(x)$ (exercise).
- Convex: $\cup$ shaped. Concave: $\cap$ shaped.
- Convex $\Leftarrow$ second derivative non-negative.

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 $\mathrm{KL}(p||q)$, we apply Jensen's inequality with $\lambda_i = p_i$, making use of the fact that $-\ln(x)$ is convex:

$$\mathrm{KL}(p||q) = -\sum_i p_i \ln\left(\frac{q_i}{p_i}\right) \geq -\ln\left(\sum_i p_i \frac{q_i}{p_i}\right) = -\ln\left(\sum_i q_i\right) = 0$$

Information theory and density estimation

Relation with maximum likelihood:

Empirical distribution :

$$p(\boldsymbol{x}) = \frac{1}{N}\sum_{n=1}^{N}\delta(\boldsymbol{x}-\boldsymbol{x}_n)$$

Approximating distribution (model) : $q(\boldsymbol{x}|\boldsymbol{\theta})$

$$\begin{aligned}\mathrm{KL}(p||q) &= -\int p(\boldsymbol{x})\ln q(\boldsymbol{x}|\boldsymbol{\theta})d\boldsymbol{x} - \int p(\boldsymbol{x})\ln p(\boldsymbol{x})d\boldsymbol{x} \\ &= -\frac{1}{N}\sum_{n=1}^{N}\ln q(\boldsymbol{x}_n|\boldsymbol{\theta}) + \text{const.}\end{aligned}$$

Thus, minimizing the KL-divergence between the empirical distribution $p(\boldsymbol{x})$ and the model distribution $q(\boldsymbol{x}|\boldsymbol{\theta})$ is equivalent to maximum likelihood (i.e., maximizing the likelihood of i.i.d. data with respect to the the model parameters $\boldsymbol{\theta}$).

Information theory/mutual information

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

$$\begin{aligned} I[\boldsymbol{x}, \boldsymbol{y}] &\equiv \mathrm{KL}(p(\boldsymbol{x}, \boldsymbol{y}) || p(\boldsymbol{x})p(\boldsymbol{y})) \\ &= -\int\int p(\boldsymbol{x}, \boldsymbol{y}) \ln \Big(\frac{p(\boldsymbol{x})p(\boldsymbol{y})}{p(\boldsymbol{x}, \boldsymbol{y})} \Big) d\boldsymbol{x} d\boldsymbol{y} \end{aligned}$$

- $I(\boldsymbol{x}, \boldsymbol{y}) \geq 0$, equality iff $\boldsymbol{x}$ and $\boldsymbol{y}$ independent

Relation with conditional entropy

$$I[\boldsymbol{x}, \boldsymbol{y}] = H[\boldsymbol{x}] - H[\boldsymbol{x}|\boldsymbol{y}] = H[\boldsymbol{y}] - H[\boldsymbol{y}|\boldsymbol{x}]$$

Chapter 2

Probability distributions

- Ch 2.1 Bayesian inference with binary variables (Bernouiil distribution)
- Ch 2.2 Bayesian inference with multinomial variables (Dirichlet distribution)
- Ch 2.3 Gaussian distribution
  - Bayesian inference
  - Some properties of high dimensional Gaussians

Binary variables / Bernoulli distribution

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

$$\begin{aligned} p(x = 1|\mu) &= \mu, \\ p(x = 0|\mu) &= 1 - \mu \end{aligned}$$

Bernoulli distribution:

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

Mean and variance:

$$\begin{aligned} \mathbb{E}[x] &= \mu \\ \text{var}[x] &= \mu(1 - \mu) \end{aligned}$$

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

$$\begin{aligned} \ln p(D|\mu) &= \sum_n \ln p(x_n|\mu) = \sum_n x_n \ln \mu + (1 - x_n) \ln(1-\mu) \\ &= m \ln \mu + (N - m) \ln(1-\mu) \end{aligned}$$

Maximization w.r.t μ gives maximum likelihood solution:

$$\mu_{ML} = \frac{m}{N}$$

The beta distribution

Distribution for parameters μ. Conjugate prior for Bayesian treatment for problem.

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

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

Normalisation (Ex. 2.5)

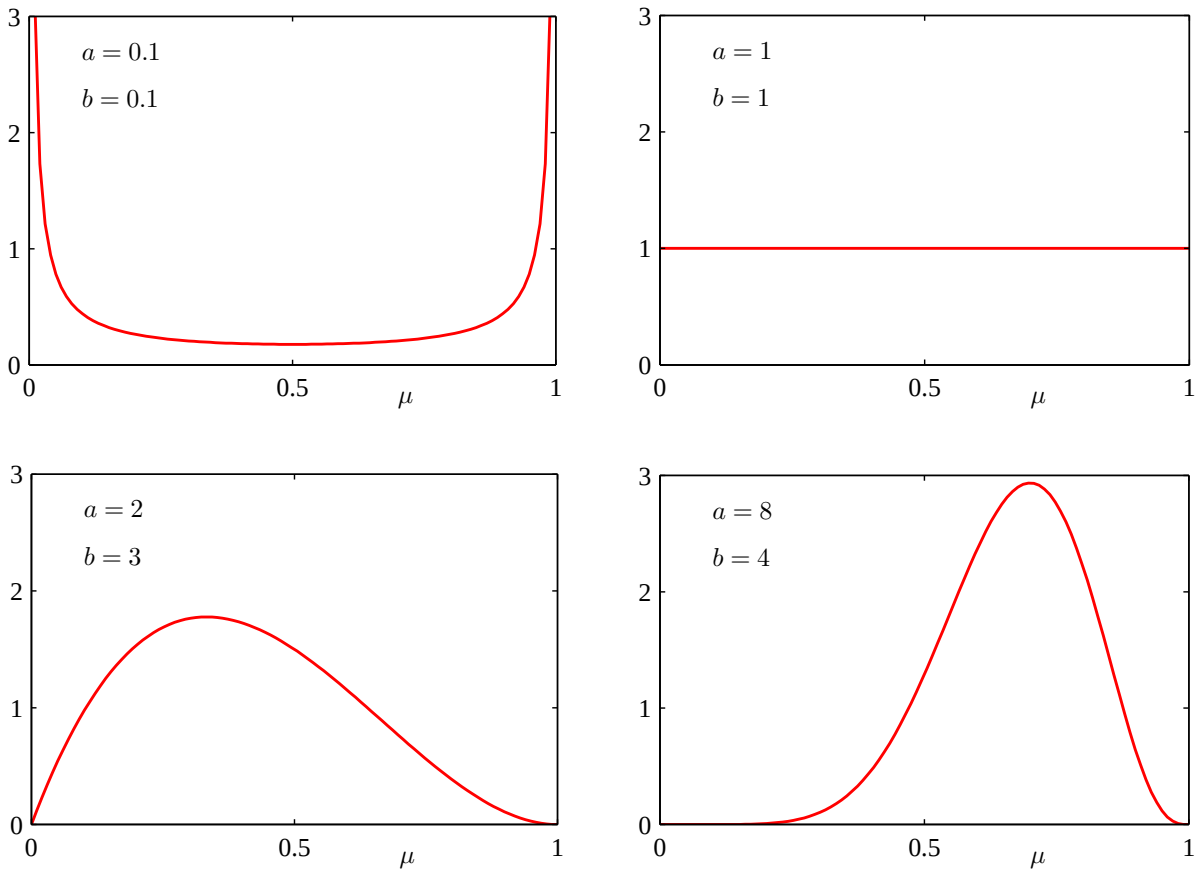
$$\int_0^1 \mu^{a-1}(1-\mu)^{b-1} = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$$

Mean and variance (Ex. 2.6)

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

$$\mathrm{var}[\mu] = \frac{ab}{(a+b)^2(a+b+1)}$$

The beta distribution



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, $(m + l = N)$

$$\begin{aligned} p(D|\mu) &= \prod_n p(x_n|\mu) = \prod_n \mu^{x_n}(1-\mu)^{1-x_n} \\ &= \mu^m(1-\mu)^l \end{aligned}$$

Posterior

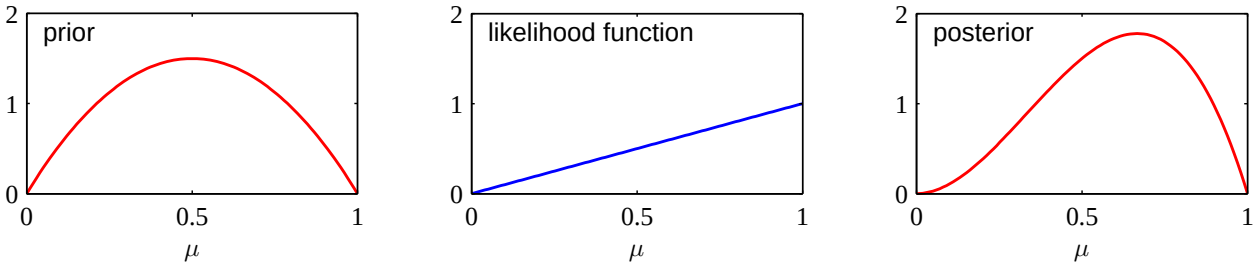
$$\begin{aligned} p(\mu|D) &\propto p(D|\mu)p(\mu) \\ &= \mu^m(1-\mu)^l \times \mu^{a-1}(1-\mu)^{b-1} \\ &= \mu^{m+a-1}(1-\mu)^{l+b-1} \propto \text{Beta}(\mu|a+m, b+l) \end{aligned}$$

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 $p(\mu|D = \mathrm{Beta}(\mu|a+m, b+l)$, 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: $x \in \{v_1, \ldots, v_K\}$,

$$\mu = (\mu_1, \ldots, \mu_K) \qquad \sum_k \mu_k = 1$$

Multinomial variables: Maximum likelihood

Likelihood:

$$\begin{aligned} p(D|\mu) &= \prod_n p(x_n|\mu) = \prod_n \prod_k \mu_k^{\delta_{x_n v_k}} \\ &= \prod_k \mu_k^{\sum_n \delta_{x_n v_k}} \\ &= \prod_k \mu_k^{m_k} \end{aligned}$$

with $m_k = \sum_n \delta_{x_n v_k}$, the total number of datapoints with value v_k. Log likelihood

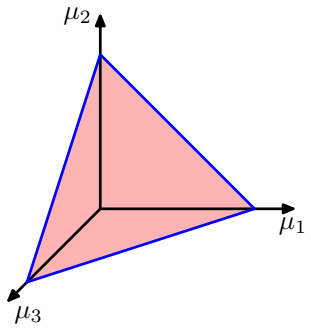
$$\ln p(D|\mu) = \sum_k m_k \ln \mu_k$$

Maximize with contraint $\sum_k \mu_k = 1$ using Lagrange multipliers.

Dirichlet distribution

$$\mathrm{Dir}(\mu|\alpha) \propto \prod_k \mu_k^{\alpha_k}$$

Probability distribution on the *simplex*:



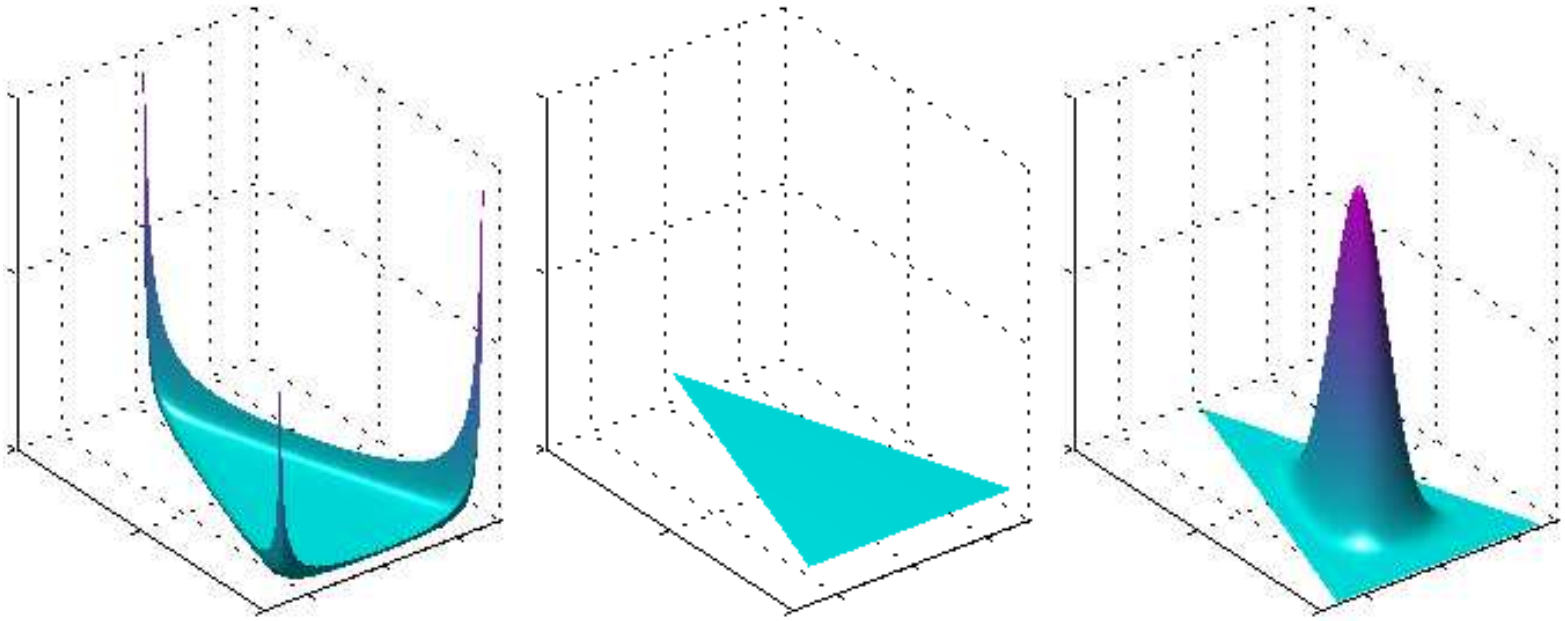
$$S^K = \{(\mu_1, \ldots, \mu_K)|0 \leq \mu_k \leq 1, \sum_{k=1}^{K} \mu_k = 1\}$$

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

$\rightarrow$ Posterior $\mathrm{Dir}(\mu|\alpha + \mathbf{m})$

Parameters α: 'pseudocounts'.

Dirichlet distribution



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

Example: text modeling

Consider a text corpus with $k = 1, \ldots, K$ different words. Build a multivariate word model $p(k|\mu)$ with $\mu = (\mu_1, \ldots, \mu_K)$ the probability of word occurence.

Prior: $\mathrm{Dir}(\mu|\alpha)$ with α pseudo counts and $\alpha_0 = \sum_{k=1}^{K} \alpha_k$
Data: vector of m of word counts and $m_0 = \sum_{k=1}^{K} m_k$ total number of words
Posterior: $\mathrm{Dir}(\mu|\alpha + m)$

The posterior probability of a new word is

$$p(k|m, \alpha) = \int d\mu p(k|\mu) p(\mu|m, \alpha) = \ldots = \frac{m_k + \alpha_k}{m_0 + \alpha_0}$$

(see ex. 2.10). The uncertainty in k can also be estimated.

Given models for different text domains (sports, fashion, politics) labeled by a. The likelihood of the text is

$$p(\mathrm{text}|a) = \prod_{k \in \mathrm{text}} p(k|a)$$

a new text can be classified to the most likely domain.

Gaussian

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 μ and σ^2

In d dimensions

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

Central limit theorem

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

Let X_i be random variables with mean μ and variance σ^2. Define the mean $Y_N = \frac{1}{N}(X_1 + \ldots + X_N)$

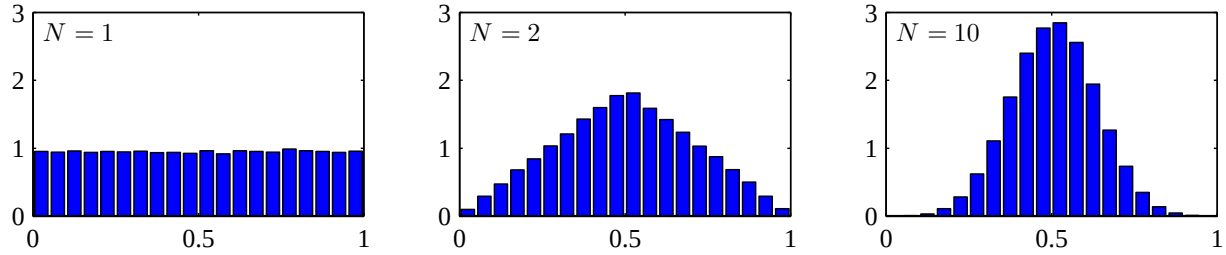
We have

$$\mathbb{E}Y_N = \mu \qquad \mathbb{V}Y_N = \frac{\sigma^2}{N}$$

Central limit theorem: Distribution of

$$Z_N = \frac{\sqrt{N}}{\sigma}(Y_N - \mu)$$

converges to Gaussian $Z_N \sim \mathcal{N}(0, 1)$


$N = 1$
$N = 2$
$N = 10$
0
1
2
3
0.5

Symmetric matrices

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 λ_k and d eigenvectors $\boldsymbol{u}_k$:

$$A\boldsymbol{u}_k = \lambda_k \boldsymbol{u}_k, \quad k = 1, \ldots, d$$

or

$$(A - \lambda_k I)\boldsymbol{u}_k = 0$$

Solution of this equation for non-zero u_k requires λ_k to satisfy the characteristic equation:

$$\det(A - \lambda I) = 0$$

This is a polynomial equation in λ of degree d and has thus d solutions [3] $\lambda_1, \ldots, \lambda_d$.

[3] 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 $\boldsymbol{u}_j$ from the left:

$$\boldsymbol{u}_j^T A \boldsymbol{u}_k = \lambda_k \boldsymbol{u}_j^T \boldsymbol{u}_k$$

Multiply the j-th eigenvalue equation by $\boldsymbol{u}_k$ from the left:

$$\boldsymbol{u}_k^T A \boldsymbol{u}_j = \lambda_j \boldsymbol{u}_k^T \boldsymbol{u}_j = \lambda_j \boldsymbol{u}_j^T \boldsymbol{u}_k$$

Subtract

$$(\lambda_k - \lambda_j) \boldsymbol{u}_j^T \boldsymbol{u}_k = 0$$

Thus, eigenvectors with different eigenvalues are orthogonal

If $\lambda_k = \lambda_j$ then any linear combination is also an eigenvector:

$$A(\alpha \boldsymbol{u}_k + \beta \boldsymbol{u}_j) = \lambda_k(\alpha \boldsymbol{u}_k + \beta \boldsymbol{u}_j)$$

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

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

In summary,

$$\boldsymbol{u}_j^T \boldsymbol{u}_k = \delta_{jk}$$

with δ_{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 $U = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_d)$.

U is an orthogonal matrix [4] , 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,

$$\begin{aligned} \det(UU^T) = 1 &\Rightarrow \det(U)\det(U^T) = 1 \\ &\Rightarrow \det(U) = \pm 1 \end{aligned}$$

[4]

$$U_{ij} = (u_j)_i, \quad (U^T U)_{ij} = \sum_k (U^T)_{ik} U_{kj} = \sum_k U_{ki} U_{kj} = \sum_k (u_i)_k (u_j)_k = \boldsymbol{u}_i^T \cdot \boldsymbol{u}_j = \delta_{ij}$$

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

$$\bar{\mathbf{x}}_1 = U\boldsymbol{x}_1 \qquad \bar{\mathbf{x}}_2 = U\boldsymbol{x}_2$$

then

$$\bar{\mathbf{x}}_1^T\bar{\mathbf{x}}_2 = \boldsymbol{x}_1^T U^T U\boldsymbol{x}_2 = \boldsymbol{x}_1^T\boldsymbol{x}_2$$

Diagonalization

The eigenvector equation can be written as

$$\begin{aligned} AU &= A(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_d) = (A\boldsymbol{u}_1, \ldots, A\boldsymbol{u}_d) = (\lambda_1 \boldsymbol{u}_1, \ldots, \lambda_d \boldsymbol{u}_d) \\ &= (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_d) \begin{pmatrix} \lambda_1 & \ldots & 0 \\ \vdots & & \vdots \\ 0 & \ldots & \lambda_d \end{pmatrix} \\ &= U\Lambda \end{aligned}$$

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'

$$A = \sum_{k=1}^{d} \lambda_k \boldsymbol{u}_k \boldsymbol{u}_k^T$$

Applications

$$A = U\Lambda U^T \Rightarrow A^2 = U\Lambda U^T U\Lambda U^T = U\Lambda^2 U^T$$

$$A^n = U\Lambda^n U^T \qquad A^{-n} = U\Lambda^{-n} U^T$$

Determinant is product of eigenvalues:

$$\det(A) = \det(U\Lambda U^T) = \det(U)\det(\Lambda)\det(U^T) = \prod_k \lambda_k$$

Basis transformation

We can represent an arbitrary vector $\boldsymbol{x}$ in d dimensions on a new basis U as

$$\boldsymbol{x} = UU^T\boldsymbol{x} = \sum_{k=1}^{d} \boldsymbol{u}_k(\boldsymbol{u}_k^T\boldsymbol{x})$$

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

$$A' = U^T AU$$

is the representation on the new basis.

For instance if $\boldsymbol{y} = A\boldsymbol{x}$, $\bar{x} = U^T\boldsymbol{x}, \bar{y} = U^T\boldsymbol{y}$, then

$$A'\bar{x} = U^T AUU^T\boldsymbol{x} = U^T A\boldsymbol{x} = U^T\boldsymbol{y} = \bar{y}$$

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

$$\begin{aligned}
\mathcal{N}(\boldsymbol{x}|\mu, \boldsymbol{\Sigma}) &= \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\mu)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\mu)\right) \\
\mu &= \langle \boldsymbol{x} \rangle = \int \boldsymbol{x}\mathcal{N}(\boldsymbol{x}|\mu, \boldsymbol{\Sigma})\,\mathrm{d}\boldsymbol{x} \\
\boldsymbol{\Sigma} &= \left\langle (\boldsymbol{x}-\mu)(\boldsymbol{x}-\mu)^T \right\rangle = \int (\boldsymbol{x}-\mu)^T(\boldsymbol{x}-\mu)\mathcal{N}(\boldsymbol{x}|\mu, \boldsymbol{\Sigma})\,\mathrm{d}\boldsymbol{x}
\end{aligned}$$

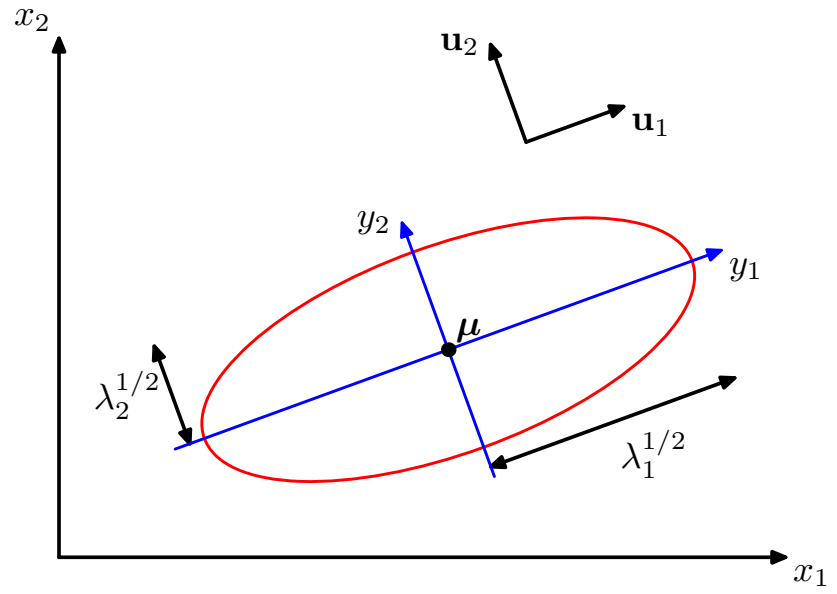
We can also write this in component notation:

$$\begin{aligned}
\mu_i &= \langle x_i \rangle = \int x_i\mathcal{N}(\boldsymbol{x}|\mu, \boldsymbol{\Sigma})\,\mathrm{d}\boldsymbol{x} \\
\Sigma_{ij} &= \langle (x_i-\mu_i)(x_j-\mu_j) \rangle = \int (x_i-\mu_i)(x_j-\mu_j)\mathcal{N}(\boldsymbol{x}|\mu, \boldsymbol{\Sigma})\,\mathrm{d}\boldsymbol{x}
\end{aligned}$$

Multivariate Gaussian

Spectral (eigenvalue) representation of $\mathbf{\Sigma}$:

$$\begin{aligned}
\mathbf{\Sigma} = U\Lambda U^T &= \sum_k \lambda_k \boldsymbol{u}_k \boldsymbol{u}_k^T \\
\mathbf{\Sigma}^{-1} = U\Lambda^{-1}U^T &= \sum_k \lambda_k^{-1} \boldsymbol{u}_k \boldsymbol{u}_k^T \\
(\boldsymbol{x}-\mu)^T U^T U \mathbf{\Sigma}^{-1} U^T U (\boldsymbol{x}-\mu) &= \boldsymbol{y}^T \Lambda^{-1} \boldsymbol{y}
\end{aligned}$$



Multivariate Gaussian

Explain the normalization: use transformation $\boldsymbol{y} = U(\boldsymbol{x} - \mu) = U\mathbf{z}$

$$\begin{aligned}
&\int \exp\left(-\frac{1}{2}(\boldsymbol{x}-\mu)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\mu)\right) d\boldsymbol{x} \\
&= \int \exp\left(-\frac{1}{2}\mathbf{z}^T\boldsymbol{\Sigma}^{-1}\mathbf{z}\right) d\mathbf{z} = \int \left|\det\left(\frac{\partial \mathbf{z}}{\partial \boldsymbol{y}}\right)\right| \exp\left(-\frac{1}{2}\boldsymbol{y}^T\Lambda^{-1}\boldsymbol{y}\right) d\boldsymbol{y} \\
&= \int \exp\left(-\frac{1}{2\lambda_1}y_1^2\right) dy_1 \ldots \int \exp\left(-\frac{1}{2\lambda_d}y_d^2\right) dy_d \\
&= \sqrt{2\pi\lambda_1}\ldots\sqrt{2\pi\lambda_d} = (2\pi)^{d/2}\Big(\prod_i \lambda_i\Big)^{1/2} = (2\pi)^{d/2}\det(\boldsymbol{\Sigma})^{1/2}
\end{aligned}$$

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

Multivariate Gaussian

Compute the expectation value : use shift-transformation $\mathbf{z} = \boldsymbol{x} - \mu$
Take Z as normalisation constant.

Use symmetry $f(\mathbf{z}) = -f(-\mathbf{z}) \Rightarrow \int f(\mathbf{z}) d\mathbf{z} = 0$

$$\begin{aligned}
\mathbb{E}[\boldsymbol{x}] &= \frac{1}{Z} \int \exp\left(-\frac{1}{2}(\boldsymbol{x}-\mu)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\mu)\right) \boldsymbol{x} d\boldsymbol{x} \\
&= \frac{1}{Z} \int \exp\left(-\frac{1}{2}\mathbf{z}^T \boldsymbol{\Sigma}^{-1}\mathbf{z}\right) (\mathbf{z}+\mu) d\mathbf{z} \\
&= \mu
\end{aligned}$$

Multivariate Gaussian

Second order moment: First, shift by $\mathbf{z} = \boldsymbol{x} - \mu$

$$\begin{aligned}
\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^T] &= \int \mathcal{N}(\boldsymbol{x}|\mu, \boldsymbol{\Sigma})\boldsymbol{x}\boldsymbol{x}^T d\boldsymbol{x} \\
&= \int \mathcal{N}(\mathbf{z}|0, \boldsymbol{\Sigma})(\mathbf{z}+\mu)(\mathbf{z}+\mu)^T d\mathbf{z} \\
&= \int \mathcal{N}(\mathbf{z}|0, \boldsymbol{\Sigma})(\mathbf{z}\mathbf{z}^T + \mathbf{z}\mu^T + \mu\mathbf{z}^T + \mu\mu^T) d\mathbf{z} \\
&= \int \mathcal{N}(\mathbf{z}|0, \boldsymbol{\Sigma})\mathbf{z}\mathbf{z}^T d\mathbf{z} + \mu\mu^T
\end{aligned}$$

Multivariate Gaussian

Now use transformation $\boldsymbol{y} = U\mathbf{z}$, and use $\Sigma = U^T \Lambda U$

$$\begin{aligned} \int \mathcal{N}(\mathbf{z}|0, \boldsymbol{\Sigma}) \mathbf{z}\mathbf{z}^T d\mathbf{z} &= \int \mathcal{N}(\boldsymbol{y}|0, \Lambda) U^T \boldsymbol{y}\boldsymbol{y}^T U d\boldsymbol{y} \\ &= U^T \int \mathcal{N}(\boldsymbol{y}|0, \Lambda) \boldsymbol{y}\boldsymbol{y}^T d\boldsymbol{y} U \end{aligned}$$

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

$$\begin{aligned} i \neq j \quad &\rightarrow \quad \int \mathcal{N}(\boldsymbol{y}|0, \Lambda) y_i y_j d\boldsymbol{y} = \int \mathcal{N}(y_i|0, \lambda_i) y_i dy_i \int \mathcal{N}(y_j|0, \lambda_j) y_j dy_j = 0 \\ i = j \quad &\rightarrow \quad \int \mathcal{N}(\boldsymbol{y}|0, \Lambda) y_i^2 d\boldsymbol{y} = \int \mathcal{N}(y_i|0, \lambda_i) y_i^2 dy_i = \lambda_i \end{aligned}$$

So

$$\int \mathcal{N}(\mathbf{z}|0, \boldsymbol{\Sigma}) \mathbf{z}\mathbf{z}^T d\mathbf{z} = U^T \Lambda U = \boldsymbol{\Sigma}$$

Multivariate Gaussian

So, second moment is

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

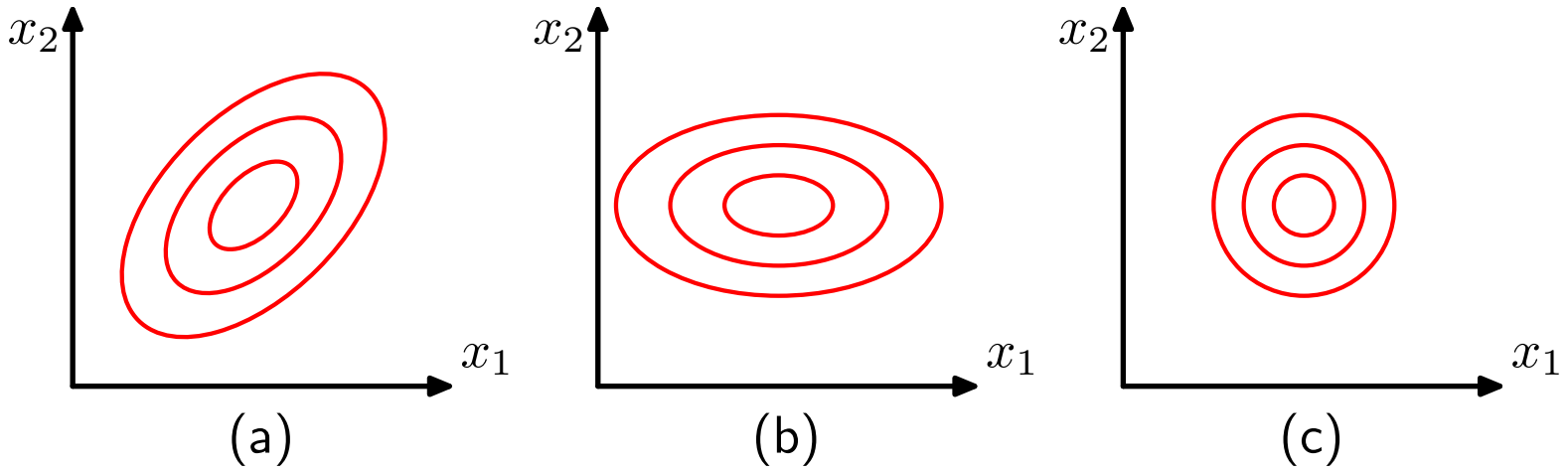
Covariance

$$\mathrm{cov}[\boldsymbol{x}] = \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^T] - \mathbb{E}[\boldsymbol{x}]\mathbb{E}[\boldsymbol{x}]^T = \boldsymbol{\Sigma}$$

Multivariate Gaussian

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

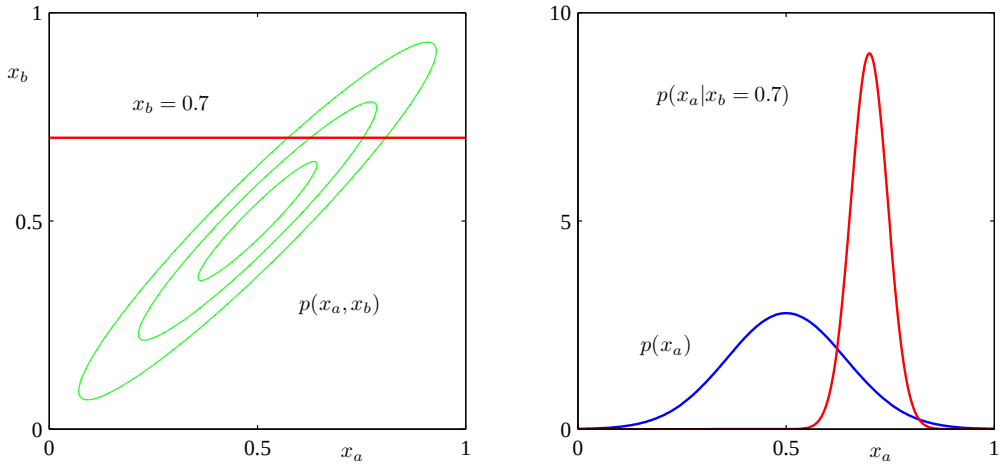
Number of parameters quadratic in d which may be too large for high dimensional applications.



Common simplifications: $\mathbf{\Sigma}_{ij} = \mathbf{\Sigma}_{ii}\delta_{ij}$ ($2d$ parameters) or $\mathbf{\Sigma}_{ij} = \sigma^2\delta_{ij}$ ($d+1$ parameters).

Marginal and conditional Gaussians

Marginal and conditional of Gaussians are also Gaussian



1. (Proof next slides)

$$
\begin{aligned}
p(\boldsymbol{x}_a|\boldsymbol{x}_b) &= \mathcal{N}(\boldsymbol{x}_a|\mu_{a|b}, \Sigma_{a|b}) \\
\Sigma_{a|b} &= \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba} \qquad \mu_{a|b} = \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(\boldsymbol{x}_b - \mu_b)
\end{aligned}
$$

2. (Proof: look at the moments)

$$
p(\boldsymbol{x}_a) = \mathcal{N}(\boldsymbol{x}_a|\mu_a, \boldsymbol{\Sigma}_{aa})
$$

Conditional of Gaussian is Gaussian

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

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

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

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

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

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

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

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

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

Some matrix identities

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

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} M & -MBD^{-1} \\ -D^{-1}CM & D^{-1} + D^{-1}CMBD^{-1} \end{pmatrix}$$

with $M = \left(A - BD^{-1}C\right)^{-1}$.

$$\begin{pmatrix} K_{aa} & K_{ab} \\ K_{ba} & K_{bb} \end{pmatrix} = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}^{-1} = \begin{pmatrix} M & -M\Sigma_{ab}\Sigma_{bb}^{-1} \\ -\Sigma_{bb}^{-1}\Sigma_{ba}M & \Sigma_{bb}^{-1} + \Sigma_{bb}^{-1}\Sigma_{ba}M\Sigma_{ab}\Sigma_{bb}^{-1} \end{pmatrix}$$

with $M = \left(\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}\right)^{-1}$. Thus, $K_{aa} = M$ and

$$\begin{aligned} \Sigma_{a|b} &= K_{aa}^{-1} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba} \\ \mu_{a|b} &= \mu_a - K_{aa}^{-1}K_{ab}(\boldsymbol{x}_b - \mu_b) = \mu_a + M^{-1}M\Sigma_{ab}\Sigma_{bb}^{-1}(\boldsymbol{x}_b - \mu_b) \\ &= \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(\boldsymbol{x}_b - \mu_b) \end{aligned}$$

Bayes' theorem for linear Gaussian model

Given marginal Gaussian on $\mathbf{x}$ and linear relation $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b} + \xi$:

$$\begin{aligned} p(\boldsymbol{x}) &= \mathcal{N}(\boldsymbol{x}|\mu, \boldsymbol{\Lambda}^{-1}) \\ p(\boldsymbol{y}|\boldsymbol{x}) &= \mathcal{N}(\boldsymbol{y}|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}, \boldsymbol{L}^{-1}) \end{aligned}$$

Then (see next slide):

$$\begin{aligned} p(\boldsymbol{y}) &= \mathcal{N}(\boldsymbol{y}|\boldsymbol{A}\mu + \boldsymbol{b}, \boldsymbol{L}^{-1} + \boldsymbol{A}\boldsymbol{\Lambda}^{-1}\boldsymbol{A}^T) \\ p(\boldsymbol{x}|\boldsymbol{y}) &= \mathcal{N}(\boldsymbol{x}|\boldsymbol{\Sigma}\{\boldsymbol{A}^T\boldsymbol{L}(\boldsymbol{y} - \boldsymbol{b}) + \boldsymbol{\Lambda}\mu\}, \boldsymbol{\Sigma}) \\ \boldsymbol{\Sigma} &= (\boldsymbol{\Lambda} + \boldsymbol{A}^T\boldsymbol{L}\boldsymbol{A})^{-1} \end{aligned}$$

We will use these relations for Bayesian linear regression in section 3.3.

Details computation $p(\boldsymbol{y})$

$$\begin{aligned} p(\boldsymbol{x}) &= \mathcal{N}(\boldsymbol{x}|\mu, \boldsymbol{\Lambda}^{-1}) \qquad \mathbb{E}\boldsymbol{x} = \mu \qquad \mathbb{V}\boldsymbol{x} = \boldsymbol{\Lambda}^{-1} \\ p(\boldsymbol{y}|\boldsymbol{x}) &= \mathcal{N}(\boldsymbol{y}|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}, \boldsymbol{L}^{-1}) \end{aligned}$$

We write $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} + \boldsymbol{\epsilon}$ with $\mathbb{E}\boldsymbol{\epsilon} = 0, \mathbb{V}\boldsymbol{\epsilon} = \boldsymbol{L}^{-1}$.

$\boldsymbol{x}, \boldsymbol{y}$ is jointly Gaussian (product of Gaussians). $\boldsymbol{y}$ is Gaussian (marginal of Gaussian).

$$\begin{aligned} \mathbb{E}\boldsymbol{y} &= \mathbb{E}\left(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} + \boldsymbol{\epsilon}\right) = \boldsymbol{A}\mu + \boldsymbol{b} \\ \mathbb{V}\boldsymbol{y} &= \mathbb{V}\left(\mathbf{a}\boldsymbol{x} + \boldsymbol{b} + \boldsymbol{\epsilon}\right) = \mathbb{V}\boldsymbol{A}\boldsymbol{x} + \mathbb{V}\boldsymbol{\epsilon} = \mathbb{V}\boldsymbol{A}\boldsymbol{x} + \boldsymbol{L}^{-1} \\ \mathbb{V}\boldsymbol{A}\boldsymbol{x} &= \mathbb{E}\left(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}\mu\right)\left(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}\mu\right)^T = \boldsymbol{A}\mathbb{E}\left(\boldsymbol{x} - \mu\right)\left(\boldsymbol{x} - \mu\right)^T \boldsymbol{A}^T = \boldsymbol{A}\boldsymbol{\Lambda}^{-1}\boldsymbol{A}^T \end{aligned}$$

Thus,

$$p(\boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{y}|\boldsymbol{A}\mu + \boldsymbol{b}, \boldsymbol{A}\boldsymbol{\Lambda}^{-1}\boldsymbol{A}^T + \boldsymbol{L}^{-1}\right)$$

Details computation $p(\boldsymbol{x}|\boldsymbol{y})$

Write all relevant terms that occur in exponential of the joint Gaussian $p(\boldsymbol{x}, \boldsymbol{y})$:

$$-\frac{1}{2}(\boldsymbol{x}-\mu)^T\boldsymbol{\Lambda}^{-1}(\boldsymbol{x}-\mu)-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b})^T\boldsymbol{L}^{-1}(\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b})$$

Collect all quadratic and linear terms in $\boldsymbol{x}$:

$$-\frac{1}{2}\boldsymbol{x}^T\left(\boldsymbol{\Lambda}^{-1}+\boldsymbol{A}^T\boldsymbol{L}^{-1}\boldsymbol{A}\right)\boldsymbol{x}+\boldsymbol{x}^T\left(\boldsymbol{\Lambda}^{-1}\mu+\boldsymbol{A}^T\boldsymbol{L}^{-1}(\boldsymbol{y}-\boldsymbol{b})\right)$$

Define $\boldsymbol{\Sigma}^{-1}=\boldsymbol{\Lambda}^{-1}+\boldsymbol{A}^T\boldsymbol{L}^{-1}\boldsymbol{A}, \boldsymbol{\Sigma}^{-1}\boldsymbol{m}=\boldsymbol{\Lambda}^{-1}\mu+\boldsymbol{A}^T\boldsymbol{L}^{-1}(\boldsymbol{y}-\boldsymbol{b})$, then

$$-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x}+\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{m}\propto-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{m})$$

Thus

$$p(\boldsymbol{x}|\boldsymbol{y})=\mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\Sigma}\left(\boldsymbol{\Lambda}^{-1}\mu+\boldsymbol{A}^T\boldsymbol{L}^{-1}(\boldsymbol{y}-\boldsymbol{b})\right),\boldsymbol{\Sigma}\right)$$

Bayesian inference for the Gaussian

Aim: inference of unknown parameter μ. Assume σ given.

Likelihood of μ 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 μ with the data set:

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

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

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

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:

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

μ_0, σ_0 hyperparameters. Large σ_0 = large prior uncertainty in μ.

$$p(\mu|\text{Data}) \propto p(\text{Data}|\mu)p(\mu)$$

Posterior is proportional to the product of two Gaussian potentials.

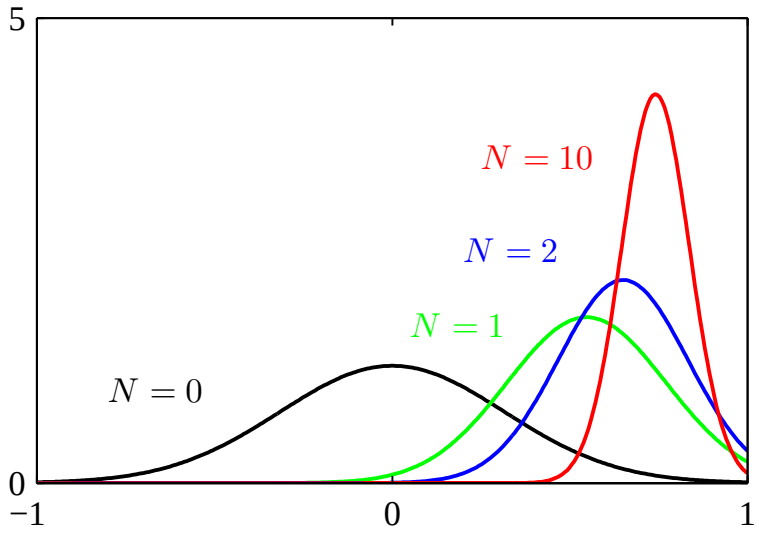
$$\begin{aligned} p(\text{Data}|\mu) &\propto \mathcal{N}(\mu|\bar{x}, \frac{1}{N}\sigma^2) \\ p(\mu) &= \mathcal{N}(\mu|\mu_0, \sigma_0^2) \end{aligned}$$

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

with (Ex. 2.38)

$$\begin{aligned} \mu_N &= \frac{N\sigma_0^2\bar{x} + \sigma^2\mu_0}{N\sigma_0^2 + \sigma^2} = \frac{N\bar{x} + N_0\mu_0}{N + N_0} \\ \frac{1}{\sigma_N^2} &= \frac{N + N_0}{\sigma^2} \end{aligned}$$

Interpretation: μ_0 mean of pseudodata; $N_0 = \frac{\sigma^2}{\sigma_0^2}$ is effective number of pseudocounts



For $N \to \infty$: $\mu_N \to \bar{x}, \sigma_N^2 \to 0$
i.e., posterior distribution is a peak around ML solution
so Bayesian inference and ML coincides.

D dimensional case

Likelihood is Gaussian

$$\begin{aligned} p(D|\mu,\Sigma) &= \prod_n p(x^n|\mu,\Sigma) \propto \exp\left(-\frac{1}{2}\sum_n (x^n-\mu)^T\Sigma^{-1}(x^n-\mu)\right) \\ &\propto \exp\left(-\frac{N}{2}\mu^T\Sigma^{-1}\mu + \sum_n \mu^T\Sigma^{-1}x^n\right) \\ &= \exp\left(-\frac{N}{2}\mu^T\Sigma^{-1}\mu + N\mu^T\Sigma^{-1}m\right) \\ &\propto \exp\left(-\frac{N}{2}(\mu-m)^T\Sigma^{-1}(\mu-m)\right) \propto \mathcal{N}(\mu|m,\frac{1}{N}\Sigma) \end{aligned}$$

with $m = \frac{1}{N}\sum_n x^n$ the mean of the data. Likelihood concentrates on the mean.

Prior is Gaussian $\mathcal{N}(\mu|\mu_0,\Lambda^{-1})$. Posterior is Gaussian

$$p(\mu|D) \propto \exp\left(-\frac{1}{2}\mu^T\left(N\Sigma^{-1}+\Lambda^{-1}\right)\mu + \left(m^T N\Sigma^{-1} + \mu_0^T\Lambda^{-1}\right)\mu\right)$$

multivariate generalization of previous case.

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:

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = w_0 + \sum_{j=1}^{D} w_j x_j$$

Linear in parameters $\boldsymbol{w}$ and in input $\boldsymbol{x}$.

Generalize this to models that need not be linear in input $\boldsymbol{x}$

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\boldsymbol{x})$$

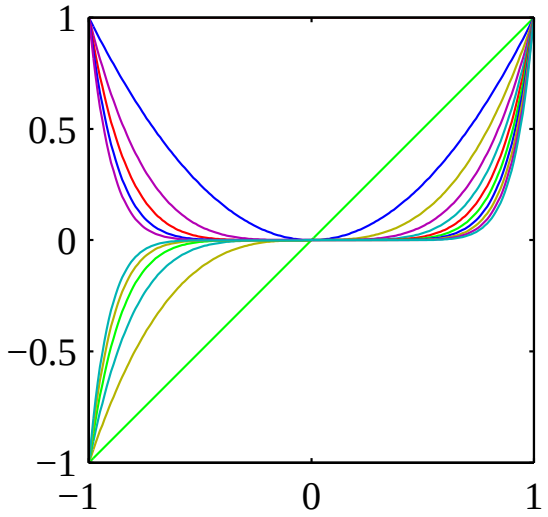
where $\phi_1(\boldsymbol{x}), \ldots, \phi_{M-1}(\boldsymbol{x})$ are *basis functions* or *feature functions*. Defining $\phi_0(\boldsymbol{x}) = 1$, we can write it more compactly:

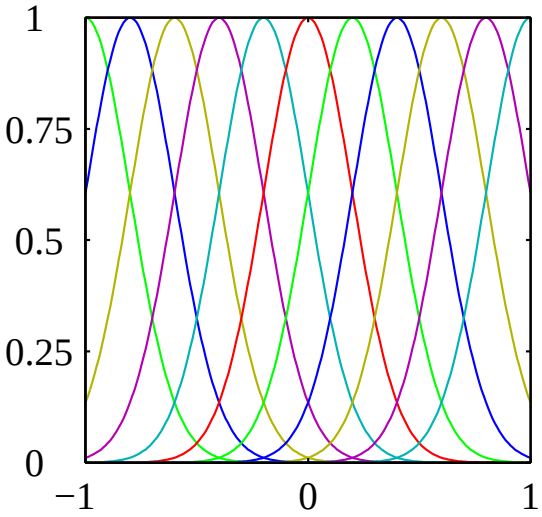
$$y(\boldsymbol{x}, \boldsymbol{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})$$

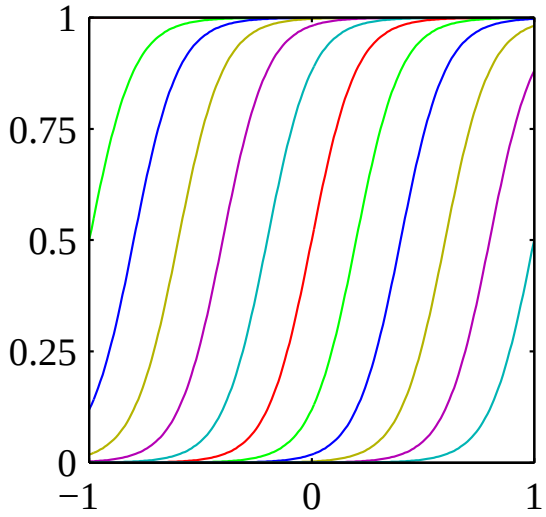
Note: still linear in parameters $\boldsymbol{w}$! Examples of basis functions?

Linear Basis Function Models

Polynomials: $\phi_j(x) = x^j$
Gaussians: $\phi_j(x) = \exp(-\beta(x - \mu_j)^2)$
Sigmoids: $\phi_j(x) = \sigma(\beta(x - \mu_j))$
Fourier: $\phi_j(x) = \exp(ijx)$
Wavelets: ...







Adding some noise

Let's make some noise. . . and assume the following model:

$$t = y(\boldsymbol{x}, \boldsymbol{w}) + \epsilon, \qquad y(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})$$

where we added Gaussian noise $\epsilon \sim \mathcal{N}(0, \beta^{-1})$.

Q: what is the distribution of t, given $\boldsymbol{x}, \boldsymbol{w}, \beta$?

Adding some noise

Let's make some noise. . . and assume the following model:

$$t = y(\boldsymbol{x}, \boldsymbol{w}) + \epsilon, \qquad y(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})$$

where we added Gaussian noise $\epsilon \sim \mathcal{N}(0, \beta^{-1})$.

Q: what is the distribution of t, given $\boldsymbol{x}, \boldsymbol{w}, \beta$?

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

Q: what is the conditional mean $\mathbb{E}(t|\boldsymbol{x}, \boldsymbol{w}, \beta)$?

Adding some noise

Let's make some noise. . . and assume the following model:

$$t = y(\boldsymbol{x}, \boldsymbol{w}) + \epsilon, \qquad y(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})$$

where we added Gaussian noise $\epsilon \sim \mathcal{N}(0, \beta^{-1})$.

Q: what is the distribution of t, given $\boldsymbol{x}, \boldsymbol{w}, \beta$?

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

Q: what is the conditional mean $\mathbb{E}(t|\boldsymbol{x}, \boldsymbol{w}, \beta)$?

$$\mathbb{E}(t|\boldsymbol{x}, \boldsymbol{w}, \beta) = \int t p(t|\boldsymbol{x})\, dt = y(\boldsymbol{x}, \boldsymbol{w})$$

Maximum likelihood: least squares

Q: What is the likelihood of a data set $\{(\boldsymbol{x}_n, t_n)\}_{n=1}^N$?

Maximum likelihood: least squares

Q: What is the likelihood of a data set $\{(\boldsymbol{x}_n, t_n)\}_{n=1}^N$?

$$p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}\left(t_n|\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n), \beta^{-1}\right)$$

Q: How does the log-likelihood depend on the parameters $\boldsymbol{w}$?

Maximum likelihood: least squares

Q: What is the likelihood of a data set $\{(\boldsymbol{x}_i, t_i)\}_{n=1}^N$?

$$p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}\left(t_n|\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n), \beta^{-1}\right)$$

Q: How does the log-likelihood depend on the parameters $\boldsymbol{w}$?

$$\ln p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}, \beta) = C - \frac{1}{2}\beta \underbrace{\sum_{n=1}^{N} \left(t_n - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n)\right)^2}_{\text{sum-of-squares error function}}$$

Q: How to optimize the log-likelihood with respect to parameters $\boldsymbol{w}$?

Maximum likelihood: least squares

Q: What is the likelihood of a data set $\{(\boldsymbol{x}_i, t_i)\}_{i=1}^N$?

$$p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}\left(t_n|\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n), \beta^{-1}\right)$$

Q: How does the log-likelihood depend on the parameters $\boldsymbol{w}$?

$$\ln p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}, \beta) = C - \frac{1}{2}\beta \underbrace{\sum_{n=1}^{N} \left(t_n - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n)\right)^2}_{\text{sum-of-squares error function}}$$

Q: How to optimize the log-likelihood with respect to parameters $\boldsymbol{w}$?
A: differentiate with respect to $\boldsymbol{w}$ and set to zero:

$$\nabla_{\boldsymbol{w}} \ln p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}, \beta) = -\beta \sum_{n=1}^{N} \left(t_n - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n)\right)\boldsymbol{\phi}(\boldsymbol{x}_n)^T = \boldsymbol{0}$$

Maximum likelihood: least squares

Rewriting:

$$\mathbf{0} = \sum_{n=1}^{N} t_n \boldsymbol{\phi}(\boldsymbol{x}_n)^T - \boldsymbol{w}^T \left(\sum_{n=1}^{N} \boldsymbol{\phi}(\boldsymbol{x}_n) \boldsymbol{\phi}(\boldsymbol{x}_n)^T \right)$$

Solving for $\boldsymbol{w}$:

$$\boldsymbol{w}_{ML} = (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{t}$$

where $\boldsymbol{\Phi}$ is an $N \times M$ matrix, the *design matrix*, with elements $\Phi_{nj} = \phi_j(\mathbf{x_n})$:

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\boldsymbol{x}_1) & \phi_1(\boldsymbol{x}_1) & \dots & \phi_{M-1}(\boldsymbol{x}_1) \\ \phi_0(\boldsymbol{x}_2) & \phi_1(\boldsymbol{x}_2) & \dots & \phi_{M-1}(\boldsymbol{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\boldsymbol{x}_N) & \phi_1(\boldsymbol{x}_N) & \dots & \phi_{M-1}(\boldsymbol{x}_N) \end{pmatrix}$$

The matrix $\boldsymbol{\Phi}^T \boldsymbol{\Phi}$ is $M \times M$. If $N < M$ the inverse does not exist.

Also, one can derive:[5]

$$\beta_{ML}^{-1} = \frac{1}{N}\sum_{n=1}^{N}\left(t_n - \boldsymbol{w}_{ML}^T\boldsymbol{\phi}(\boldsymbol{x}_n)\right)^2.$$

Since

$$t = y(\boldsymbol{x}, \boldsymbol{w}) + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \beta^{-1})$$

β_{ML}^{-1} estimates the unexplained variance in t.

5

$$\begin{aligned}
\ln p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}, \beta) &= \frac{N}{2}\log\frac{\beta}{2\pi} - \frac{1}{2}\beta\sum_{n=1}^{N}\left(t_n - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n)\right)^2 \\
\frac{\partial \ln p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}, \beta)}{\partial\beta} &= \frac{N}{2\beta} - \frac{1}{2}\sum_{n=1}^{N}\left(t_n - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n)\right)^2
\end{aligned}$$

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, β.

Step 1: find $\mathbf{\Phi}, \boldsymbol{t}$.

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, β.

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) \\ \phi_0(x_2) & \phi_1(x_2) \\ \vdots & \vdots \\ \phi_0(x_N) & \phi_1(x_N) \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{pmatrix} \qquad \boldsymbol{t} = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{pmatrix}$$

Step 2: find $\boldsymbol{\Phi}^T\boldsymbol{\Phi}, \boldsymbol{\Phi}^T\boldsymbol{t}$:

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, β.

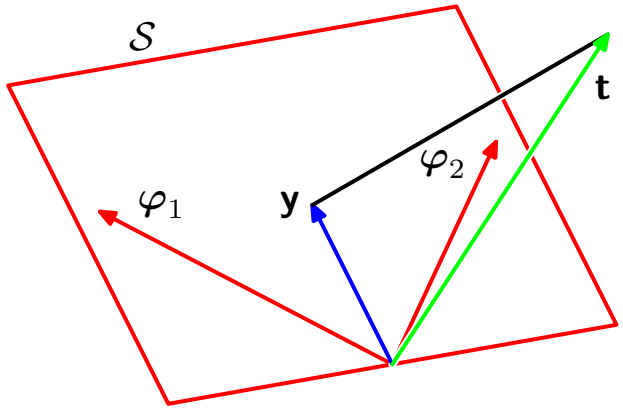
$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) \\ \phi_0(x_2) & \phi_1(x_2) \\ \vdots & \vdots \\ \phi_0(x_N) & \phi_1(x_N) \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{pmatrix} \qquad \boldsymbol{t} = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{pmatrix}$$

$$\boldsymbol{\Phi}^T\boldsymbol{\Phi} = \begin{pmatrix} N & \sum_n x_n \\ \sum_n x_n & \sum_n x_n^2 \end{pmatrix} \qquad \boldsymbol{\Phi}^T\boldsymbol{t} = \begin{pmatrix} \sum_n t_n \\ \sum_n x_n t_n \end{pmatrix}$$

$$\boldsymbol{w}_{ML} = \left(\boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^T\boldsymbol{t} \qquad \beta_{ML} = \ldots$$

Geometry of least squares

$M \leq N$:



$\boldsymbol{t} = (t_1, \ldots, t_N)$ is vector in R^N.

$\boldsymbol{\phi}_j = (\phi_j(x_1), \ldots, \phi_j(x_N)), j = 1, \ldots, M$ are M vectors in R^N that span M dimensional linear subspace S

$\boldsymbol{y} = \sum_{j=1}^{M} w_j \boldsymbol{\phi}_j$ is in S for any $\boldsymbol{w}$.

The optimal $\boldsymbol{w}$ minimizes the quadratic error $\|\boldsymbol{t} - \boldsymbol{y}\|$ and is the ortho-normal projection.

$M > N$: vectors ϕ_j are over complete bases of R^N. Therefore, there are multiple solutions $\boldsymbol{w}$ to solve $\boldsymbol{t} = \sum_{j=1}^{M} w_j \boldsymbol{\phi}_j$. 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*:

$$\ln p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}, \beta) + \lambda E_W(\boldsymbol{w}) = C - \underbrace{\frac{1}{2}\beta \sum_{i=1}^{N} \left(t_n - \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_n)\right)^2}_{\text{sum-of-squares error function}} - \underbrace{\frac{1}{2}\lambda \boldsymbol{w}^T \boldsymbol{w}}_{\text{regularizer}}$$

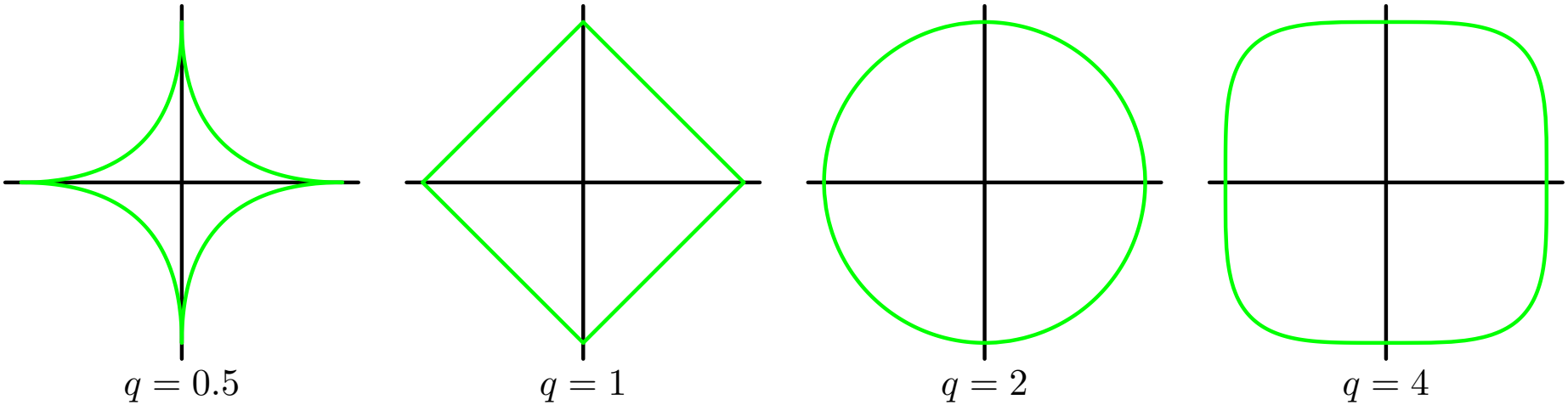
Maximizing with respect to $\boldsymbol{w}$ now gives the following optimum:

$$\boldsymbol{w} = (\lambda \boldsymbol{I} + \boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{t}$$

NB: adding the diagonal makes the matrix of maximal rank.

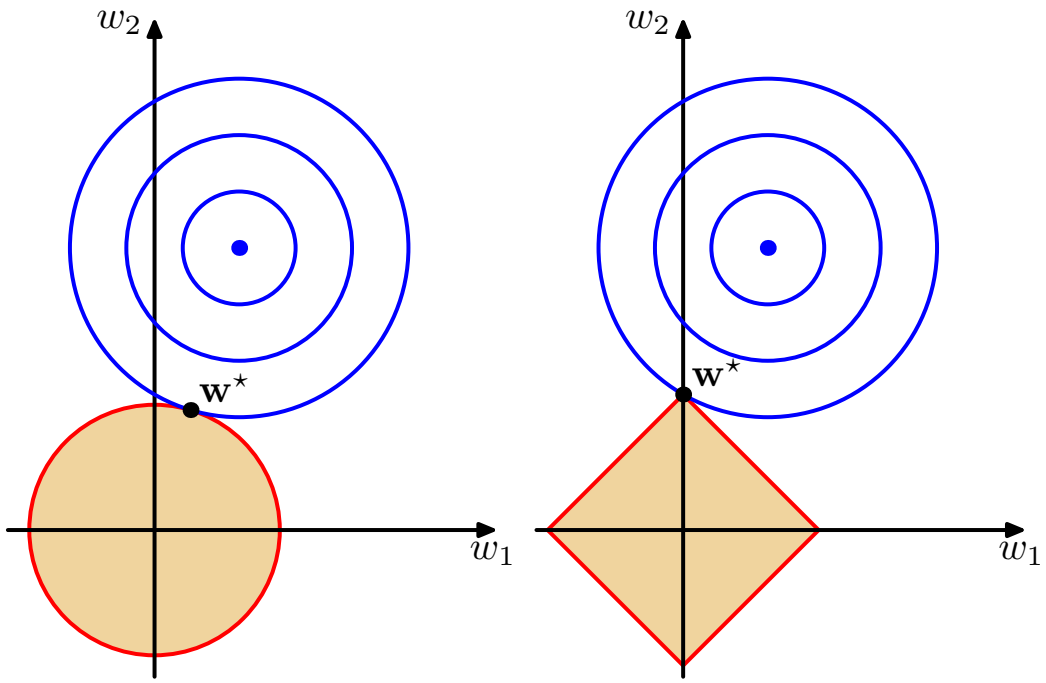
Other regularizers

$$\ln p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}, \beta) + \lambda E_W(\boldsymbol{w}) \propto -\frac{1}{2}\beta \sum_{i=1}^{N} \left(t_n - \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_n)\right)^2 - \frac{1}{2}\lambda \sum_{j=0}^{M-1} |w_j|^q$$



the LASSO with $q = 1$. This gives a *sparse* solution ($w_j = 0$ for many, but a few, j's).

Other regularizers



Regularization is equivalent to constrained optimization

$$\text{argmin}_{\boldsymbol{w}} f(\boldsymbol{w}) + \lambda \sum_{j=0}^{M-1} |w_j|^q \quad \Leftrightarrow \quad \text{argmin}_{\boldsymbol{w}} f(\boldsymbol{w}) + \lambda \left(\sum_{j=0}^{M-1} |w_j|^q - \eta \right)$$

Fix η and find $\lambda(\eta)$ rather than fix λ.

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 λ.

When learning with a finite data set $\mathcal{D}$, the solution $y(\boldsymbol{x}|\mathcal{D})$ that minimizes the quadratic loss depends $\mathcal{D}$.

$$L = \sum_{\mu \in \mathcal{D}} (y(\boldsymbol{x}^\mu) - t^\mu)^2 \qquad \rightarrow \qquad y(\boldsymbol{x}; \mathcal{D})$$

This solution scatters around the true (infinite date) solution $h(\boldsymbol{x}) = \mathbb{E}(t|\boldsymbol{x})$. We write

$$L = \sum_{\mu \in \mathcal{D}} (y(\boldsymbol{x}; \mathcal{D}) - h(\boldsymbol{x}))^2 + 2 \sum_{\mu \in \mathcal{D}} (y(\boldsymbol{x}; \mathcal{D}) - h(\boldsymbol{x}))(h(\boldsymbol{x}) - t^\mu) + \sum_{\mu \in \mathcal{D}} (h(\boldsymbol{x}) - t^\mu)^2$$

The expected value is

$$\mathbb{E}L = \sum_{\mu \in \mathcal{D}} (y(\boldsymbol{x}; \mathcal{D}) - h(\boldsymbol{x}))^2 + \underbrace{\mathbb{E} \sum_{\mu \in \mathcal{D}} (h(\boldsymbol{x}) - t^\mu)^2}_{\text{noise}}$$

We can estimate the first term. Consider the thought experiment that a large number of data sets $\mathcal{D}$ are given. Then we can construct the average solution $\bar{y}(\boldsymbol{x}) = \mathbb{E}_{\mathcal{D}}(y(\boldsymbol{x}|\mathcal{D}))$, and

$$\begin{aligned}
\big(y(\boldsymbol{x};\mathcal{D}) - h(\boldsymbol{x})\big)^2 &= \big(y(\boldsymbol{x};\mathcal{D}) - \bar{y}(\boldsymbol{x})\big)^2 + \big(\bar{y}(\boldsymbol{x}) - h(\boldsymbol{x})\big)^2 \\
&+ 2\big(y(\boldsymbol{x};\mathcal{D}) - \bar{y}(\boldsymbol{x})\big)\big(\bar{y}(\boldsymbol{x}) - h(\boldsymbol{x})\big) \\
\mathbb{E}_{\mathcal{D}}\big(y(\boldsymbol{x};\mathcal{D}) - h(\boldsymbol{x})\big)^2 &= \underbrace{\mathbb{E}_{\mathcal{D}}\big(y(\boldsymbol{x};\mathcal{D}) - \bar{y}(\boldsymbol{x})\big)^2}_{\text{variance}} + \underbrace{\mathbb{E}_{\mathcal{D}}\big(\bar{y}(\boldsymbol{x}) - h(\boldsymbol{x})\big)^2}_{\text{bias}^2}
\end{aligned}$$

Substitution in expected square loss $\mathbb{E}L$:

$$\mathbb{E}_{\mathcal{D}}\mathbb{E}L = \underbrace{\mathbb{E}_{\mathcal{D}}\big(y(\boldsymbol{x};\mathcal{D}) - \bar{y}(\boldsymbol{x})\big)^2}_{\text{variance}} + \underbrace{\mathbb{E}_{\mathcal{D}}\big(\bar{y}(\boldsymbol{x}) - h(\boldsymbol{x})\big)^2}_{\text{bias}^2} + \underbrace{\mathbb{E}\sum_{\mu\in\mathcal{D}}(h(\boldsymbol{x}) - t^{\mu})^2}_{\text{noise}}$$

Variance: scatter of individual solutions $y(\boldsymbol{x};\mathcal{D})$ around their mean $\bar{y}(\boldsymbol{x})$.
Bias: difference between mean solution $\bar{y}(\boldsymbol{x})$ and $h(x)$.
Noise: scatter of the data points t around true solution $h(x)$.

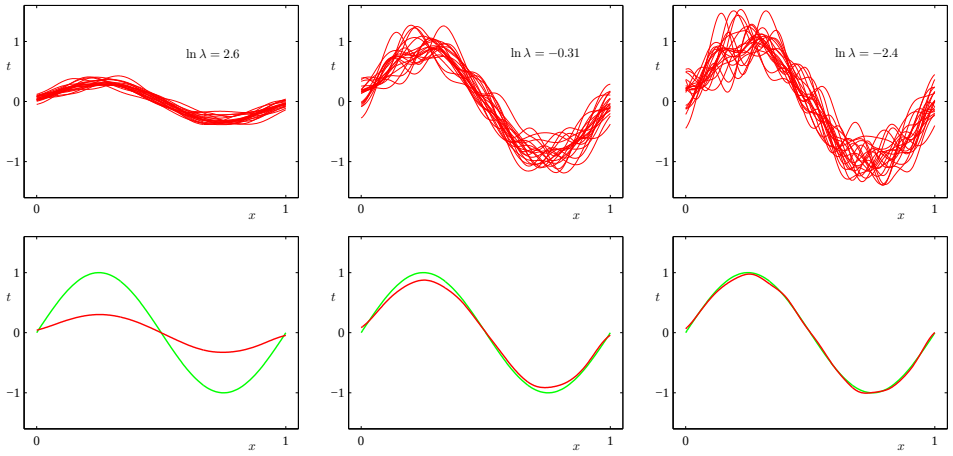
Objective is to minimize bias^2+variance. Noise cannot be optimized.

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

$$L = \frac{1}{2}\sum_{i=1}^{N}\left(t_n - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n)\right)^2 + \frac{1}{2}\lambda\boldsymbol{w}^T\boldsymbol{w}$$

for different λ. Top shows variance, bottom shows bias.

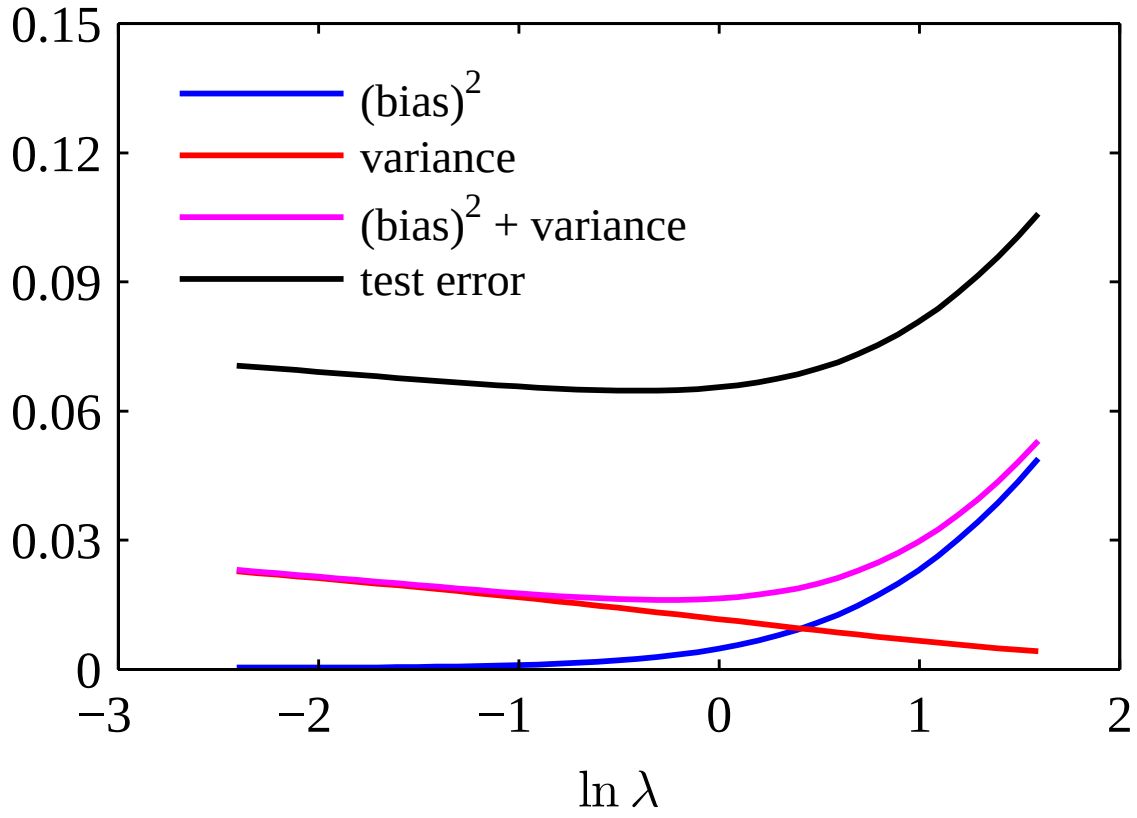


Difference models make different trade-off of bias and variance:

- Flexible/complex model (low λ): low bias, high variance
- Rigid/simple model (high λ): high bias, low variance

Best model strikes the best balance.

Bias-Variance Decomposition



Sum of bias, variance and noise yields expected error on test set. Optimal λ is trade-off between bias and variance.

Decomposition of total error in bias, variance and noise terms usually not known because $h(\boldsymbol{x})$ is not known.

Bayesian linear regression

Let's go back to the linear basis function model:

$$t = y(\boldsymbol{x}, \boldsymbol{w}) + \epsilon, \qquad y(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}), \qquad \epsilon \sim \mathcal{N}(0, \beta^{-1})$$
$$p(t|\boldsymbol{x}, \boldsymbol{w}, \beta) = \mathcal{N}(t|\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}), \beta^{-1})$$

Training data: $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ and $\boldsymbol{t} = (t_1, \ldots, t_N)$.

Likelihood:

$$p(\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\boldsymbol{w}^T \phi(\boldsymbol{x}_n), \beta^{-1}) = \mathcal{N}(\boldsymbol{t}|\boldsymbol{\Phi}\boldsymbol{w}, \beta^{-1}\boldsymbol{I}).$$

Q: what prior $p(\boldsymbol{w})$ can we choose?

Bayesian linear regression

Let's go back to the linear basis function model:

$$t = y(\boldsymbol{x}, \boldsymbol{w}) + \epsilon, \qquad y(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}), \qquad \epsilon \sim \mathcal{N}(0, \beta^{-1})$$
$$p(t|\boldsymbol{x}, \boldsymbol{w}, \beta) = \mathcal{N}(t|\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}), \beta^{-1})$$

Training data: $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ and $\boldsymbol{t} = (t_1, \ldots, t_N)$.

Likelihood:

$$p(\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\boldsymbol{w}^T \phi(\boldsymbol{x}_n), \beta^{-1}) = \mathcal{N}(\boldsymbol{t}|\boldsymbol{\Phi}\boldsymbol{w}, \beta^{-1}\boldsymbol{I}).$$

Q: what prior $p(\boldsymbol{w})$ can we choose?
A: We make life easy by choosing a *conjugate* prior, which is a Gaussian

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_0, S_0)$$

Bayesian linear regression

Then, the posterior will also be Gaussian. Prior and likelihood:

$$\begin{aligned} p(\boldsymbol{w}) &= \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_0, S_0) \\ p(\boldsymbol{t}|\boldsymbol{w}) &= \mathcal{N}(\boldsymbol{t}|\Phi\boldsymbol{w}, \beta^{-1}I) \end{aligned}$$

Then, by applying $2.113 + 2.114 \Rightarrow 2.116$, we get the posterior (Ex. 3.7):

$$p(\boldsymbol{w}|\boldsymbol{t}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_N, S_N)$$

with

$$\begin{aligned} m_N &= S_N(S_0^{-1}\boldsymbol{m}_0 + \beta\boldsymbol{\Phi}^T\boldsymbol{t}) \\ S_N^{-1} &= S_0^{-1} + \beta\boldsymbol{\Phi}^T\boldsymbol{\Phi} \end{aligned}$$

When $S_0^{-1} \to \mathbf{0}$ (broad prior) $m_N \to (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{t}$ which is the ML solution.

$\boldsymbol{\Phi}^T\boldsymbol{\Phi} \propto N$ [6]. Therefore $S_N \to 0$ when N large.

[6] $(\boldsymbol{\Phi}^T\boldsymbol{\Phi})_{ij} = \sum_n (\boldsymbol{\Phi}^T)_{in}\boldsymbol{\Phi}_{nj} = \sum_n \boldsymbol{\Phi}_{ni}\boldsymbol{\Phi}_{nj} = \sum_n \phi_i(\boldsymbol{x}_n)\phi_j(\boldsymbol{x}_n)$

Details

Correspondence with (2.113-114):

$$\begin{aligned} p(\boldsymbol{x}) &= \mathcal{N}(\boldsymbol{x}|\mu, \boldsymbol{\Lambda}^{-1}) & p(\boldsymbol{w}) &= \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_0, S_0) \\ p(\boldsymbol{y}|\boldsymbol{x}) &= \mathcal{N}(\boldsymbol{y}|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}, \boldsymbol{L}^{-1}) & p(\boldsymbol{t}|\boldsymbol{w}) &= \mathcal{N}(\boldsymbol{t}|\Phi\boldsymbol{w}, \beta^{-1}I) \end{aligned}$$

$\mu = \boldsymbol{m}_0, \boldsymbol{\Lambda}^{-1} = S_0, \boldsymbol{A} = \Phi, \boldsymbol{b} = 0, \boldsymbol{L} = \beta I.$

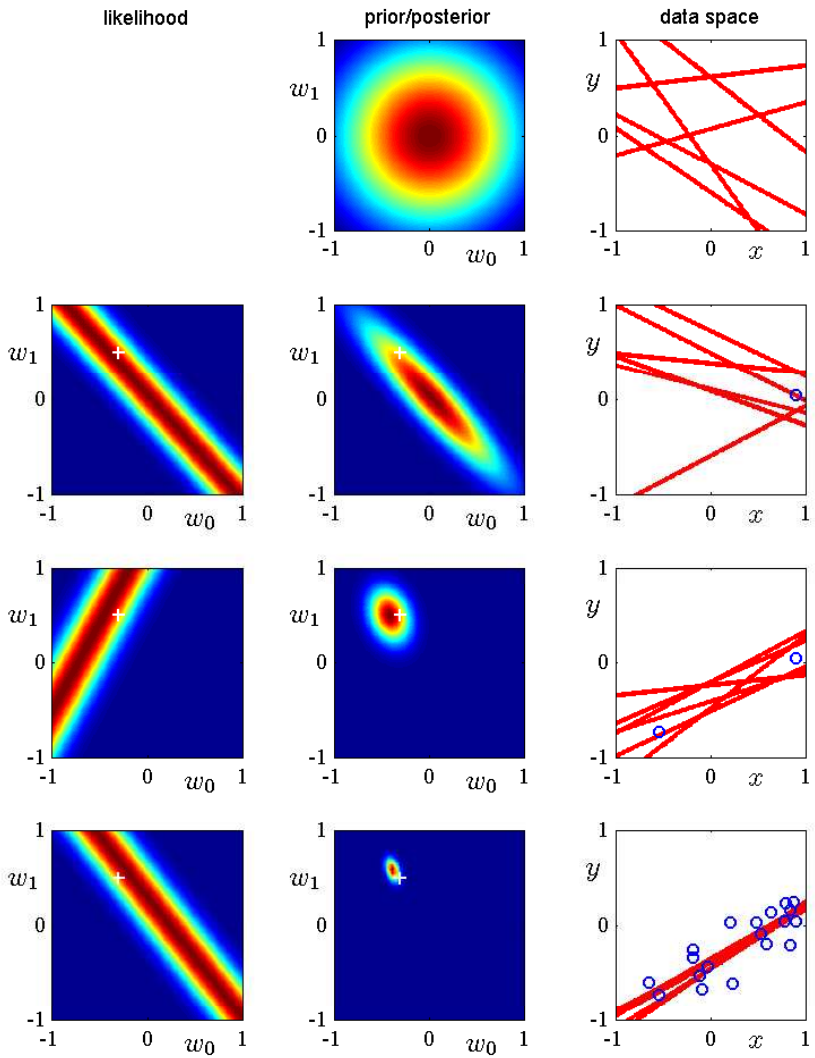
Thus from (2.116), posterior $p(\boldsymbol{w}|\boldsymbol{t}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_N, S_N)$ with

$$\begin{aligned} \boldsymbol{m}_N &= S_N(\beta\Phi^T\boldsymbol{t} + S_0^{-1}\boldsymbol{m}_0) \\ S_N^{-1} &= S_0^{-1} + \beta\Phi^T\Phi \end{aligned}$$

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: $y(x, \boldsymbol{w}) = w_0 + w_1 x$; $p(t|x, \boldsymbol{w}, \beta) = \mathcal{N}(t|y(x, \boldsymbol{w}), \beta^{-1})$, $\beta = 1/\sigma^2 = 25$;
Prior model $p(\boldsymbol{w}|\alpha) = \mathcal{N}(\boldsymbol{w}|0, \alpha^{-1}\boldsymbol{I})$, $\alpha = 2$.
α, β assumed known.



Posterior variance reduces with N.

Predictive distribution

What is the predictive distribution $p(t^*|x^*)$ for new data point x^*? We know:

$$\begin{aligned} p(t^*|\boldsymbol{w}, x^*) &= \mathcal{N}(t^*|\phi^T(x^*)\boldsymbol{w}, \beta^{-1}) \\ p(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{x}) &= \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_N, S_N) \\ p(t^*|x^*, \boldsymbol{t}, \boldsymbol{x}) &= \int d\boldsymbol{w} p(t^*|\boldsymbol{w}, x^*) p(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{x}) \end{aligned}$$

Write $t^* = \phi^T(x^*)\boldsymbol{w} + \epsilon$ with $\mathbb{V}\epsilon = \beta^{-1}$ and $\boldsymbol{w}$ given by posterior:

$$\mathbb{E}t^* = \phi^T(x^*)\boldsymbol{m}_N, \qquad \mathbb{V}t^* = \mathbb{V}\left(\phi^T(x^*)\boldsymbol{w}\right) + \mathbb{V}\epsilon = \phi^T(x^*)\mathbb{V}\left(\boldsymbol{w}\right)\phi(x^*) + \beta^{-1}$$

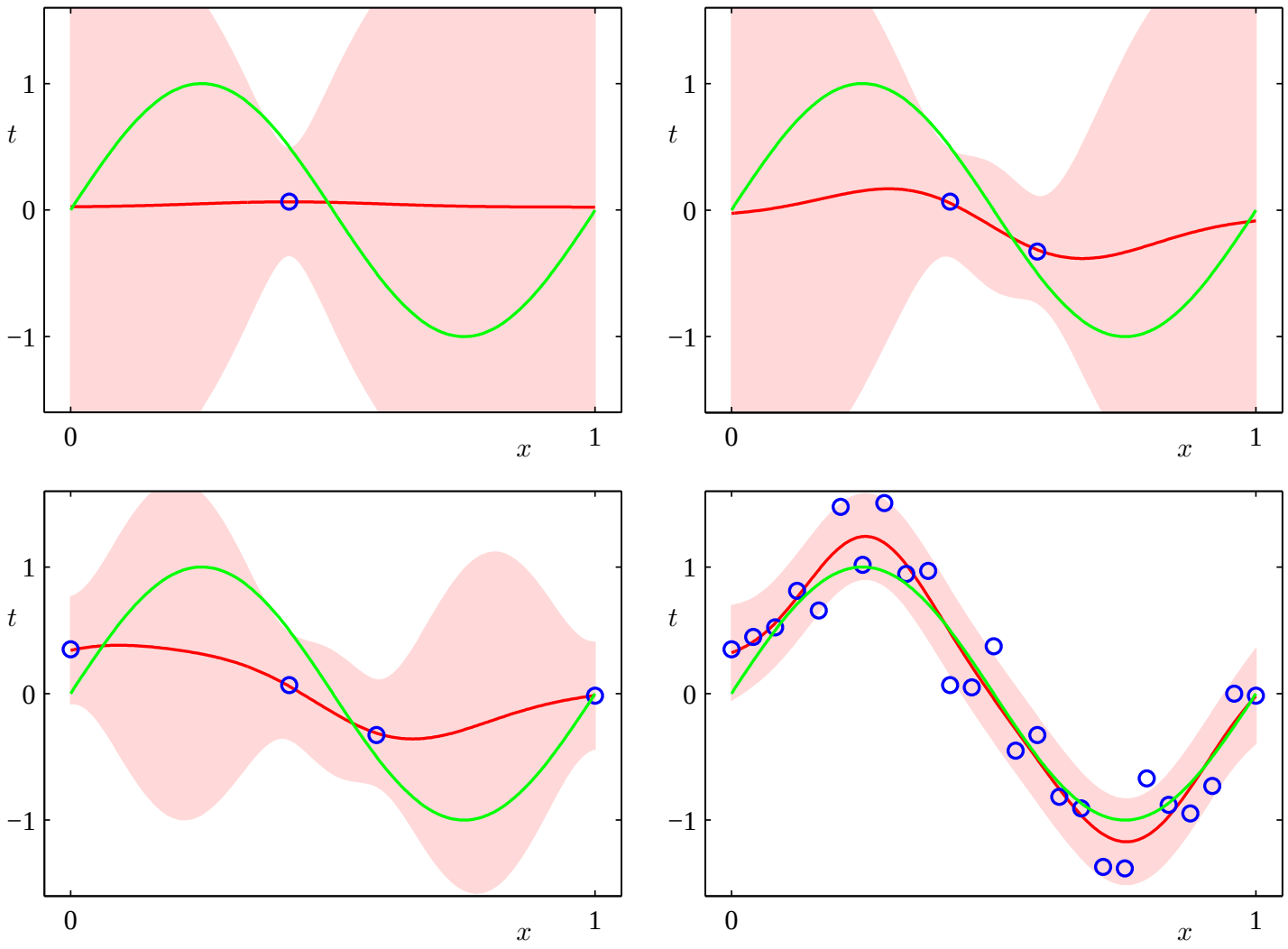
$$p(t^*|x^*, \boldsymbol{t}, \boldsymbol{x}) = \mathcal{N}(t^*|\phi^T(x^*)\boldsymbol{m}_N, \sigma_N^2(x^*))$$

where

$$\sigma_N^2(x^*) = \beta^{-1} + \phi(x^*)^T S_N \phi(x^*)$$

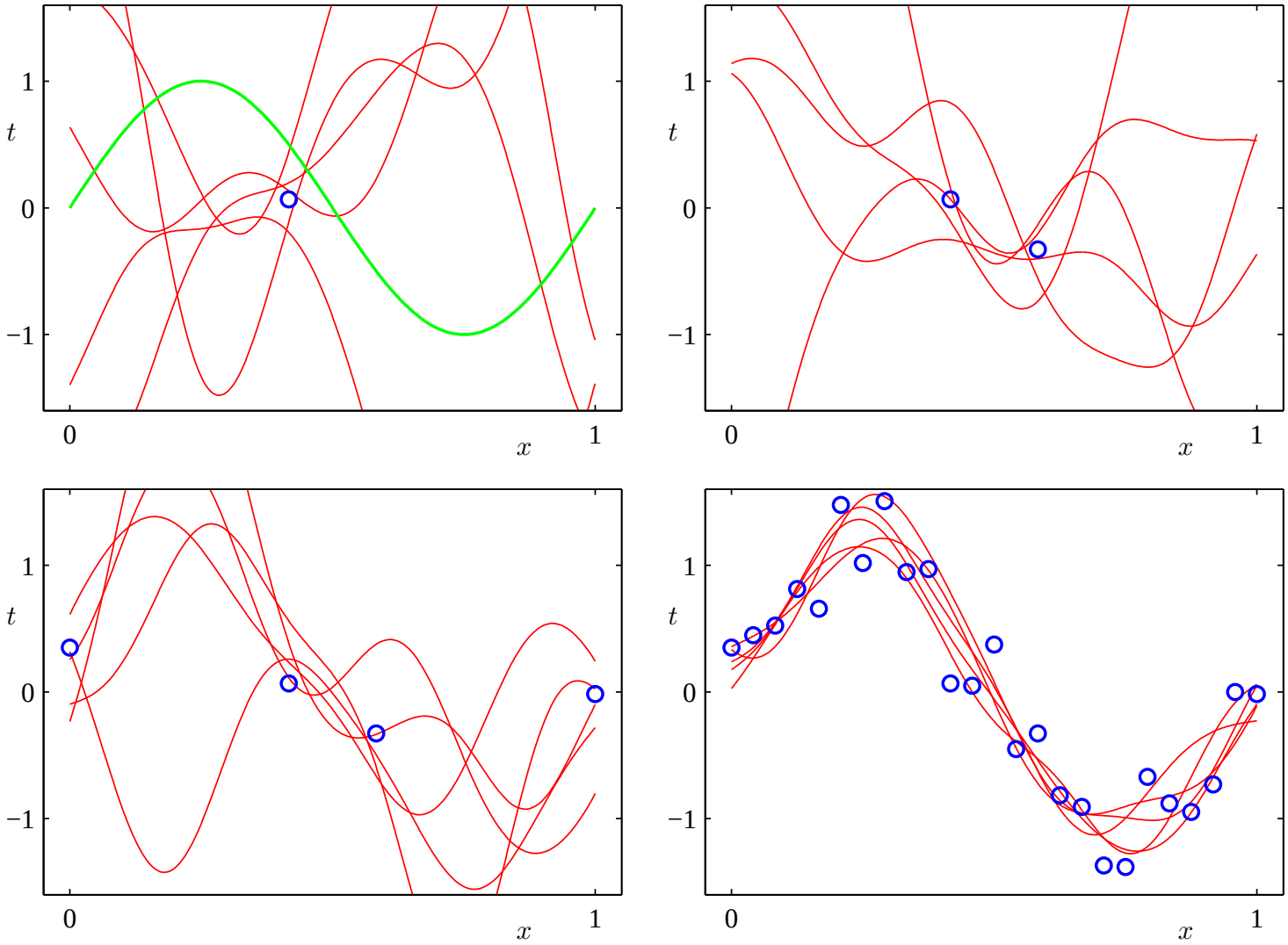
When $N \to \infty$ $S_N \to 0$ and $\sigma_N^2(x^*) \to \beta^{-1}$

Bayesian linear regression: Example



Data points from $t = \sin(2\pi x) + \text{noise}$. $y(x)$ as in Eq. 3.3-4. Red is $\boldsymbol{\phi}(x^*)^T \boldsymbol{m}_N \pm \sigma_N(x^*)$. Data set size $N = 1, 2, 4, 25$.

Bayesian linear regression: Example



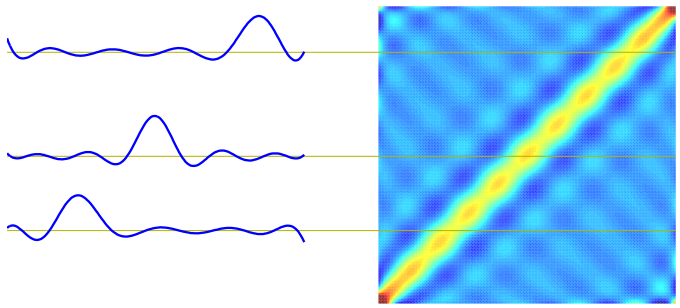
Same data and model. Curves $y(x, \boldsymbol{w})$ with $\boldsymbol{w}$ from posterior $p(\boldsymbol{w}|\boldsymbol{t}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_N, S_N)$.

Equivalent kernel

The output y for the mean posterior prediction $\boldsymbol{w} = \boldsymbol{m}_N = \beta S_N \Phi^T \boldsymbol{t}$ (assume $\boldsymbol{m}_0 = 0$) is

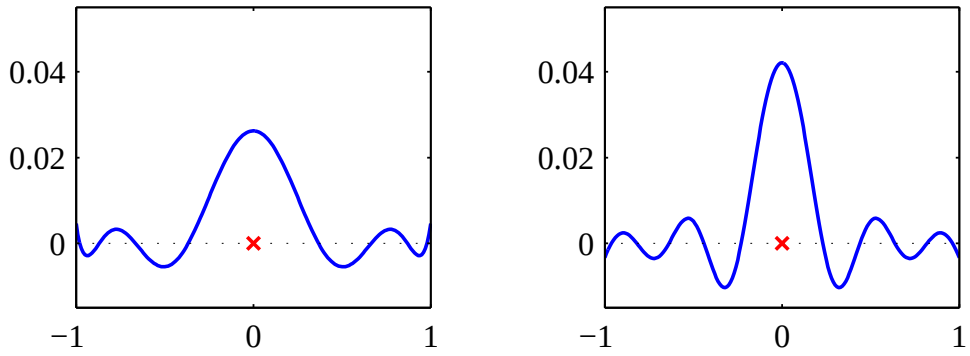
$$\begin{aligned} y(x, \boldsymbol{m}_N) &= \boldsymbol{\phi}(x)^T \boldsymbol{m}_N = \beta \boldsymbol{\phi}(x)^T S_N \Phi(\boldsymbol{x})^T \boldsymbol{t} = \beta \sum_{n=1}^{N} \boldsymbol{\phi}(x)^T S_N \boldsymbol{\phi}(\boldsymbol{x}_n) t_n \\ &= \sum_{n=1}^{N} k(x, x_n) t_n \end{aligned}$$

Mean output is linear combination of outputs at training points. Weighting is localized around x.



Kernel $k(x, x')$ for Gaussian basis function (fig. 3.1). Data are 200 equally spaced $x_n \in [-1, 1]$ and measured targets t_n.

Equivalent kernel



Kernels $k(x, x')$ for polynomial and sigmoid basis functions (fig 3.1) are localized.

Equivalent kernel

The outputs at two different inputs are correlated:

$$\begin{aligned} Cov\left(y(x), y(x')\right) &= Cov(\boldsymbol{\phi}(x)^T\boldsymbol{w}, \boldsymbol{w}^T\boldsymbol{\phi}(x')) = \boldsymbol{\phi}(x)^T\mathbb{V}\boldsymbol{w}\boldsymbol{\phi}(x') = \boldsymbol{\phi}(x)^T S_N\boldsymbol{\phi}(x') \\ &= \beta^{-1}k(x, x') \end{aligned}$$

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 $p(\mathcal{M}_i), i = 1, \ldots, L$ to express our prior uncertainty.

Given the training data $\mathcal{D}$, we wish to compute the posterior probability

$$p(\mathcal{M}_i|\mathcal{D}) \propto p(\mathcal{D}|\mathcal{M}_i)p(\mathcal{M}_i)$$

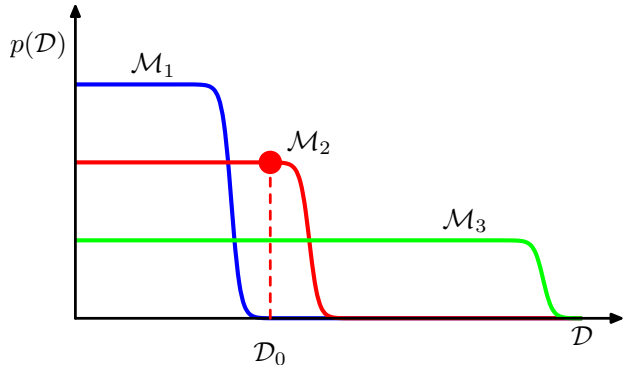
$p(\mathcal{D}|\mathcal{M}_i)$ is called *model evidence*, also *marginal likelihood*, since it integrates over model parameters:

$$p(\mathcal{D}|\mathcal{M}_i) = \int p(\mathcal{D}|\boldsymbol{w}, \mathcal{M}_i)p(\boldsymbol{w}|\mathcal{M}_i)d\boldsymbol{w}$$

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 $\boldsymbol{w}$ from the prior $p(\boldsymbol{w}|\mathcal{M}_i)$ and then generate iid data points according to $p(x|\boldsymbol{w}, \mathcal{M}_i)$. The resulting distribution is $p(\mathcal{D}|\mathcal{M}_i)$.

A simple model has less variability in the resulting data sets than a complex model. Thus, $p(\mathcal{D}|\mathcal{M}_1)$ is more peaked than $p(\mathcal{D}|\mathcal{M}_3)$. Due to normalization, $p(\mathcal{D}|\mathcal{M}_1)$ is necessarily higher than $p(\mathcal{D}|\mathcal{M}_3)$.



For the data set $\mathcal{D}_0$ 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)

$$p(\boldsymbol{w}|\alpha, M) = \left(\frac{\alpha}{2\pi}\right)^{M/2} \exp\left(-\frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w}\right)$$

and a likelihood (3.10)

$$p(\boldsymbol{t}|\boldsymbol{w}, \beta, M) = \left(\frac{\beta}{2\pi}\right)^{N/2} \exp\left(-\frac{\beta}{2}\|\boldsymbol{t} - \Phi\boldsymbol{w}\|^2\right)$$

.

The marginal likelihood is

$$\begin{aligned} p(\boldsymbol{t}|\alpha, \beta, M) &= \int d\boldsymbol{w} p(\boldsymbol{w}|\alpha, M) p(\boldsymbol{t}|\boldsymbol{w}, \beta, M) \\ &= \left(\frac{\alpha}{2\pi}\right)^{M/2} \left(\frac{\beta}{2\pi}\right)^{N/2} \int d\boldsymbol{w} \exp(-E(\boldsymbol{w})) \end{aligned}$$

Evidence framework

$$E(\boldsymbol{w}) = \frac{\beta}{2}(\boldsymbol{t} - \Phi\boldsymbol{w})^T(\boldsymbol{t} - \Phi\boldsymbol{w}) + \frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w} = \frac{\beta}{2}\left(\boldsymbol{t}^T\boldsymbol{t} - 2\boldsymbol{w}^T\Phi^T\boldsymbol{t} + \boldsymbol{w}^T\Phi^T\Phi\boldsymbol{w}\right) + \frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{w}$$
$$= \frac{1}{2}\boldsymbol{w}^T A\boldsymbol{w} + \frac{\beta}{2}\boldsymbol{t}^T\boldsymbol{t} - \beta\boldsymbol{w}^T\Phi^T\boldsymbol{t} = \frac{1}{2}\boldsymbol{w}^T A\boldsymbol{w} + \frac{\beta}{2}\boldsymbol{t}^T\boldsymbol{t} - \boldsymbol{w}^T A\boldsymbol{m}$$

with $A = \alpha I + \beta\Phi^T\Phi$ and $A\boldsymbol{m} = \beta\Phi^T\boldsymbol{t}$. Thus,

$$\begin{aligned} E(\boldsymbol{w}) &= \frac{1}{2}(\boldsymbol{w} - \boldsymbol{m})^T A(\boldsymbol{w} - \boldsymbol{m}) - \frac{1}{2}\boldsymbol{m}^T A\boldsymbol{m} + \frac{\beta}{2}\boldsymbol{t}^T\boldsymbol{t} \\ &= \frac{1}{2}(\boldsymbol{w} - \boldsymbol{m})^T A(\boldsymbol{w} - \boldsymbol{m}) + E(\boldsymbol{m}) \end{aligned}$$

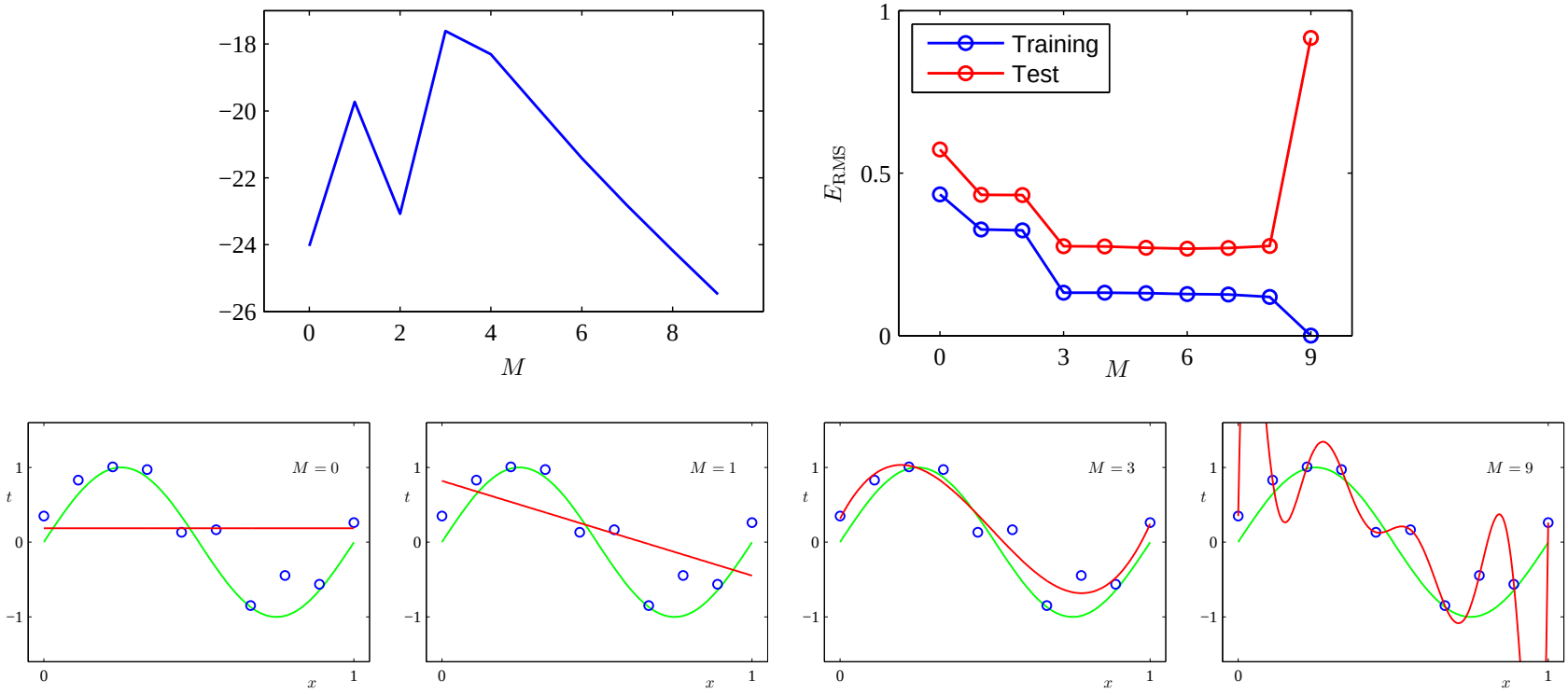
With $\int d\boldsymbol{w} \exp(-E(\boldsymbol{w})) = \exp(-E(\boldsymbol{m}))(2\pi)^{-M/2}|A|^{-1/2}$:

$$\log p(\boldsymbol{t}|\alpha, \beta, M) = \frac{M}{2}\log\alpha + \frac{N}{2}\log\beta - E(\boldsymbol{m}) - \frac{1}{2}\log|A| - \frac{M}{2}\log 2\pi$$

[7]. Note that $|A| = \prod \lambda_i$, thus $\log|A| = \mathcal{O}(M)$.

[7] NB typo Eq. 3.86 $-\frac{N}{2}\log 2\pi$

Evidence framework



Evidence framework comparing different M for fixed α, β. $M = 1$ 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.

The Perceptron

Relevant in history of pattern recognition and neural networks.

- Perceptron learning rule + convergence, Rosenblatt (1962)
- Perceptron critique (Minsky and Papert, 1969) $\rightarrow$ "Dark ages of neural networks"
- Revival in the 80's: Backpropagation and Hopfield model. Statistical physics entered.
- 1995. Bayesian methods take over. Start of modern machine learning. NN out of fashion.
- 2006 Deep learning, big data.

The Perceptron

$$y(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}))$$

where

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

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

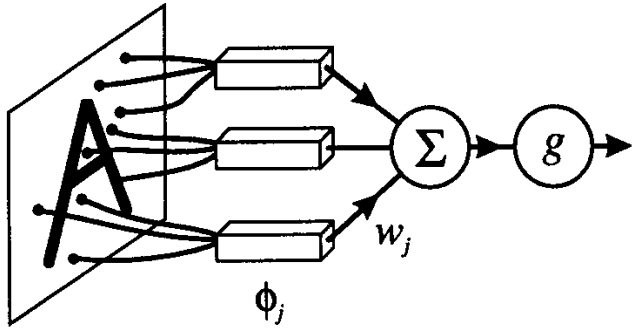
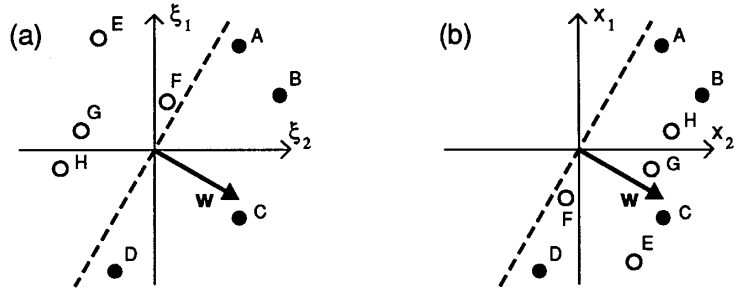


Figure 3.10. The perceptron network used a fixed set of processing elements, denoted ϕ_j, followed by a layer of adaptive weights w_j and a threshold activation function $g(\cdot)$. The processing elements ϕ_j typically also had threshold activation functions, and took inputs from a randomly chosen subset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

The Perceptron

Ignore ϕ, ie. consider inputs x^μ 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 $z_j^\mu = x_j^\mu t^\mu$.

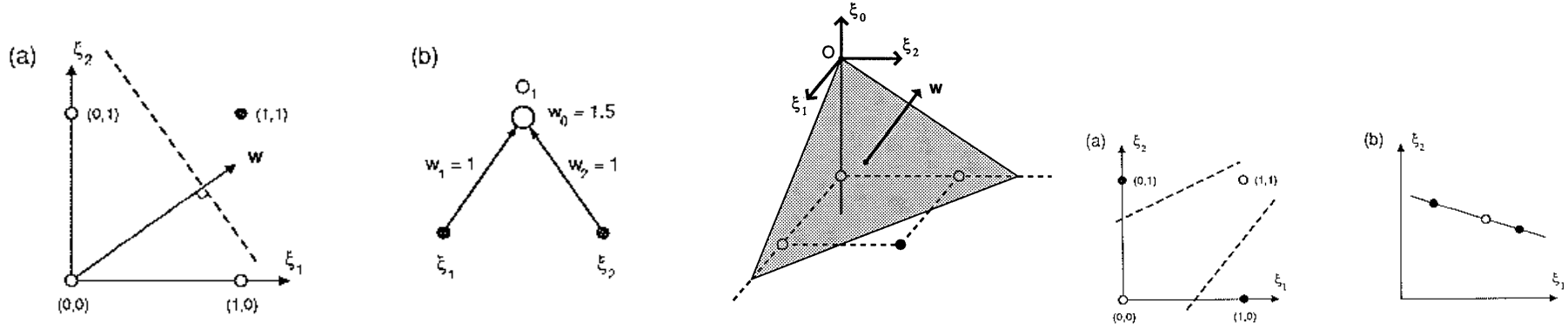
Linear separation

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

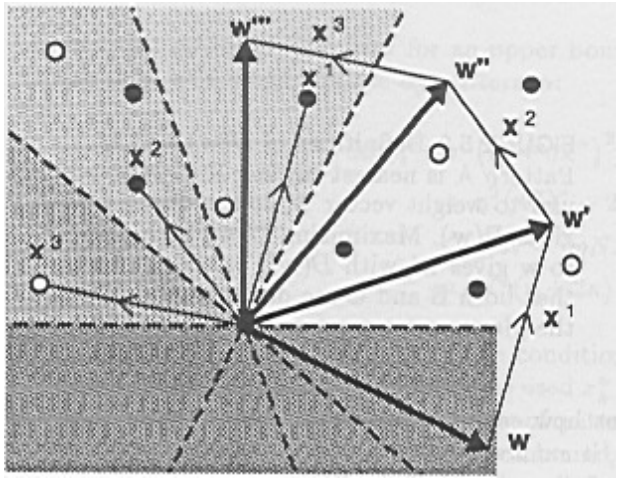
Learning succesful when

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

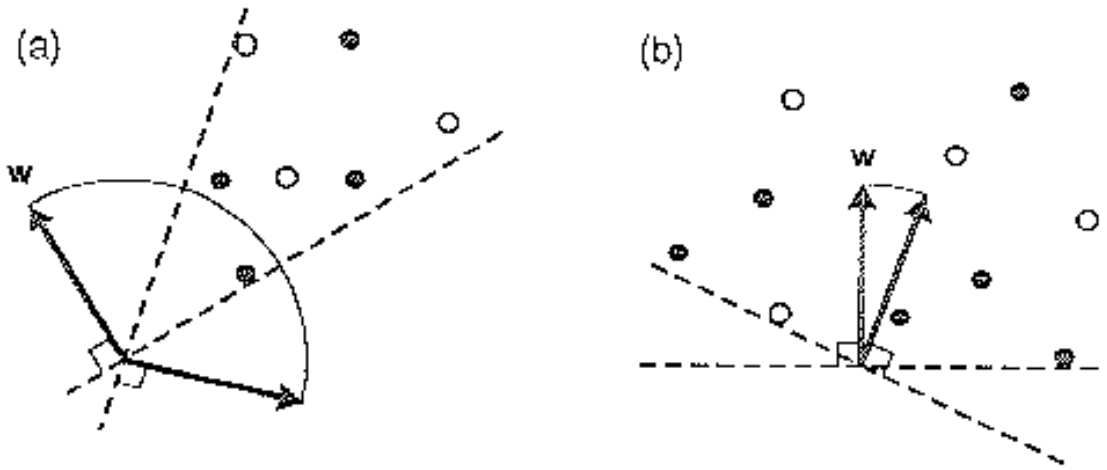
Learning rule is 'Hebbian':

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

η is the learning rate.



Depending on the data, there may be many or few solutions to the learning problem (or non at all)

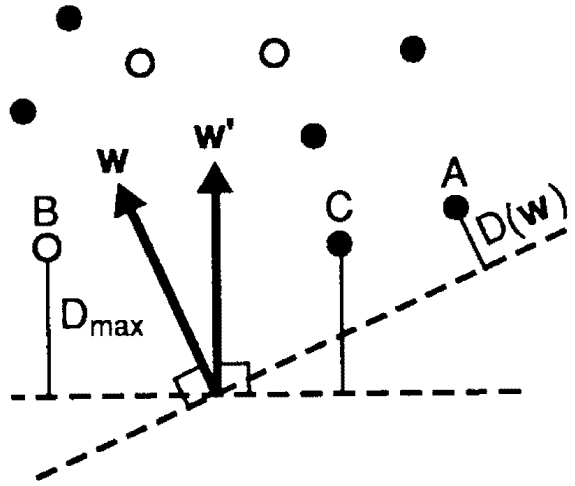


The quality of the solution is determined by the worst pattern. Since the solution does not depend on the size of w:

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

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^μ denote the number of times pattern μ has been used to update w. Thus,

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

Consider the quanty

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

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 can not grow indefinitely. Thus, the perceptron learning rule converges in a finite number of steps when the problem is linearly separable.

Proof:

$$\begin{aligned} w \cdot w^* &= \eta \sum_\mu M^\mu z^\mu \cdot w^* \geq \eta M \min_\mu z^\mu \cdot w^* \\ &= \eta M D(w^*)|w^*| \\ \Delta|w|^2 &= |w + \eta z^\mu|^2 - |w|^2 = 2\eta w \cdot z^\mu + \eta^2 |z^\mu|^2 \\ &\leq \eta^2 |z^\mu|^2 = \eta^2 N \\ |w| &\leq \eta \sqrt{NM} \end{aligned}$$

Thus,

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

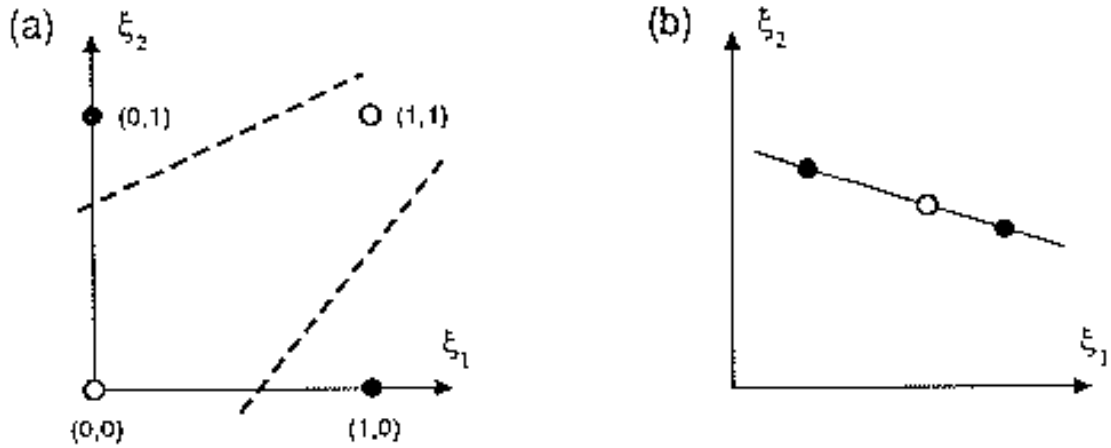
Number of weight updates:

$$M \leq \frac{N}{D^2(w^*)}$$

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



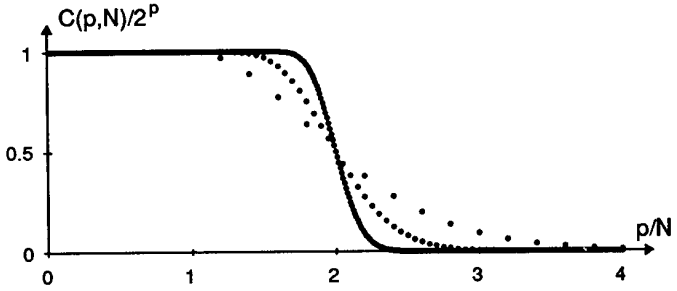
Question: What is the probability that a problem of P samples in N dimensions is linearly separable?

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):

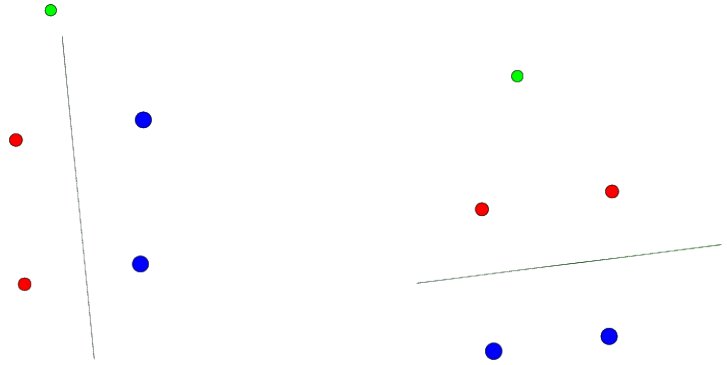
$$C(P, N) = 2 \sum_{i=0}^{N-1} \left(\begin{array}{c} P-1 \\ i \end{array} \right)$$

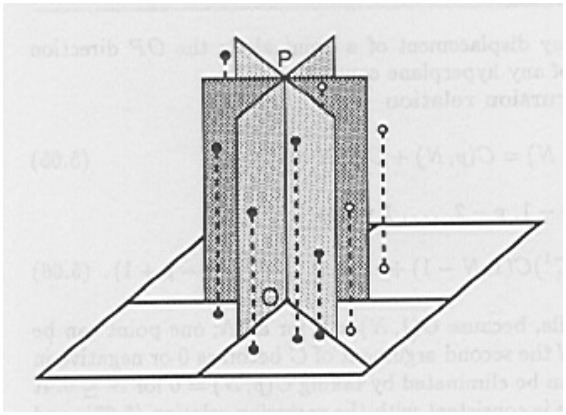
When $P \leq N$ small, then $C(P, N) = 2 \sum_{i=0}^{P-1} \left(\begin{array}{c} P-1 \\ i \end{array} \right) = 2(1+1)^{P-1} = 2^P$

When $P = 2N$, then 50 % is linearly separable: $C(P, N) = 2 \sum_{i=0}^{N-1} \left(\begin{array}{c} 2N-1 \\ i \end{array} \right) = \sum_{i=0}^{2N-1} \left(\begin{array}{c} 2N-1 \\ i \end{array} \right) = 2^{2N-1} = 2^{P-1}$



Proof by induction.





Add one point X. The set $C(P, N)$ consists of

- colorings with separator through X (A)
- rest (B)

Thus,

$$\begin{aligned} C(P+1, N) &= 2A + B = C(P, N) + A \\ &= C(P, N) + C(P, N-1) \end{aligned}$$

Yields

$$C(P, N) = 2 \sum_{i=0}^{N-1} \left(\begin{array}{c} P-1 \\ i \end{array} \right)$$

Linear units

Input-output behaviour:

$$y^\mu = w_0 + \sum_{j=1}^{n} w_j x_j^\mu = \sum_{j=0}^{n} w_j x_j \qquad x_0 = 1$$

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

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

$$\begin{aligned} E(w) &= \frac{1}{2} \sum_{\mu=1}^{P} \left(t^\mu - \sum_{j=0}^{n} w_j x_j^\mu \right)^2 \\ \Delta w_i &= -\eta \frac{\partial E}{\partial w_i} = \eta \sum_{\mu} \left(t^\mu - \sum_{j} w_j x_j^\mu \right) x_i^\mu \\ w_i &= w_i + \Delta w_i \end{aligned}$$

Gradient descent optimization

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

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

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 $\Delta \boldsymbol{w} = -\eta \nabla E$.

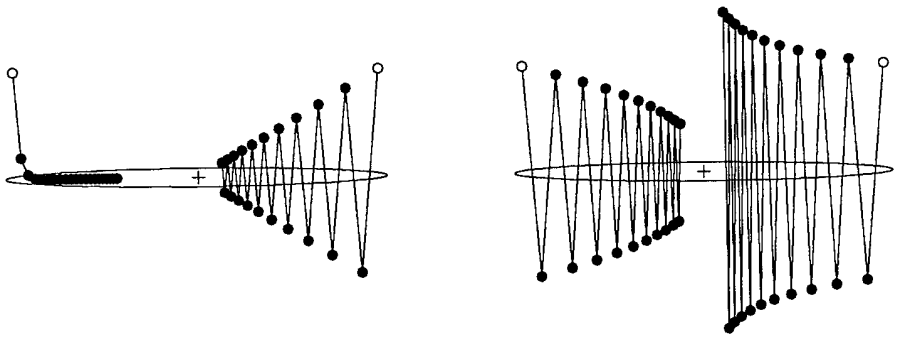
$$E(\boldsymbol{w} + \Delta \boldsymbol{w}) \approx E(\boldsymbol{w}) + (\Delta \boldsymbol{w})^T \nabla E = E(\boldsymbol{w}) - \eta \sum_i \left(\frac{\partial E}{\partial w_i}\right)^2 \leq E(\boldsymbol{w})$$

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

$$
\begin{aligned}
E(w) &= \frac{1}{2}\sum_i \lambda_i w_i^2 \\
\Delta w_i &= -\eta \frac{\partial E}{\partial w_i} = -\eta \lambda_i w_i \\
w_i^{\text{new}} &= w_i^{\text{old}} + \Delta w_i = (1 - \eta\lambda_i) w_i
\end{aligned}
$$

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



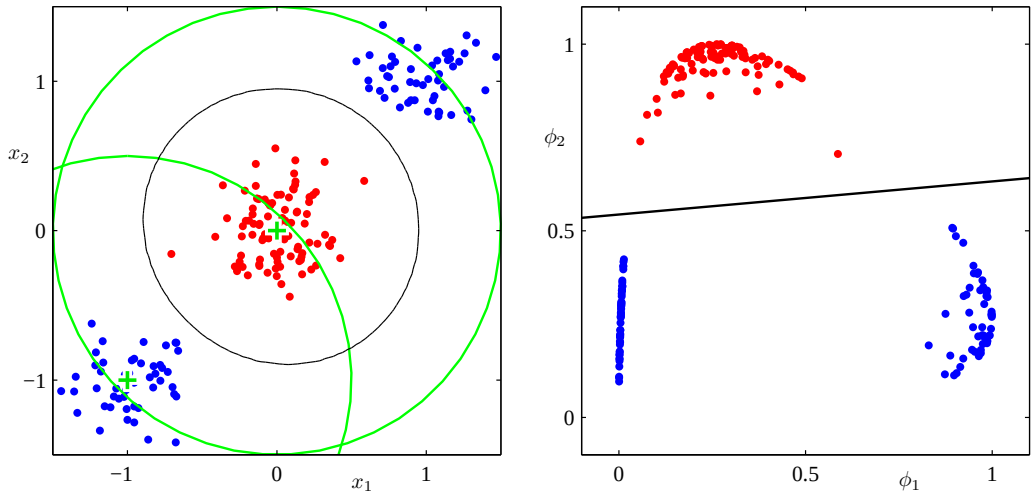
Optimal learning parameter depends on curvature of each dimension.

Logistic regression

Two class classification

$$p(C_1|\phi) = \sigma(\boldsymbol{w}^T\boldsymbol{\phi}) \qquad \sigma(x) = 1/(1+\exp(-x))$$

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 $\boldsymbol{x}$ might be linearly separable in $\boldsymbol{\phi}(\boldsymbol{x})$.

Data $(\boldsymbol{x}_n, t_n), n = 1, \ldots, N$, $t_n \in \{0, 1\}$. Maximum likelihood to determine parameters $\boldsymbol{w}$:

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

with $\boldsymbol{\phi}_n = \boldsymbol{\phi}(x_n)$ and $y_n = \sigma(\boldsymbol{w}^T \boldsymbol{\phi}_n)$ i.e.,

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

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

Logistic regression

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

(Ex. 4.13) No closed form solution.

$y_n - t_n$ with is 'error' in sample n.

Overfitting risk when data is linearly separable: $\boldsymbol{w} \to \infty$ (i.e. $\sigma \to$ step function).

Newtons method

One can also use Hessian information for optimization. As an example, consider a quadratic approximation to E around $\boldsymbol{w}_0$:

$$\begin{aligned} E(\boldsymbol{w}) &= E(\boldsymbol{w}_0) + \boldsymbol{b}^T(\boldsymbol{w} - \boldsymbol{w}_0) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_0)\boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}_0) \\ b_i &= \frac{\partial E(\boldsymbol{w}_0)}{\partial w_i} \qquad H_{ij} = \frac{\partial^2 E(\boldsymbol{w}_0)}{\partial w_i \partial w_j} \\ \nabla E(\boldsymbol{w}) &= \boldsymbol{b} + \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}_0) \end{aligned}$$

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

$$\boldsymbol{w} = \boldsymbol{w}_0 - \boldsymbol{H}^{-1}\nabla E(\boldsymbol{w}_0)$$

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.

Iterative least squares

Minimize learning error by Newton-Raphson method

$$\boldsymbol{w}^{(new)} = \boldsymbol{w}^{(old)} - \boldsymbol{H}^{-1}\nabla E(\boldsymbol{w})$$

with

$$\begin{aligned}
\boldsymbol{H}_{ij} &= \frac{\partial^2 E}{\partial w_i \partial w_j} = \frac{\partial}{\partial w_j}\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_j}\sum_n (y_n - t_n)\phi_i(\boldsymbol{x}_n) \\
&= \sum_n \phi_j(\boldsymbol{x}_n) y_n(1-y_n)\phi_i(\boldsymbol{x}_n) = \left(\boldsymbol{\Phi}^T R \boldsymbol{\Phi}\right)_{ij} \\
\nabla_i E(\boldsymbol{w}) &= \sum_n (y_n - t_n)\phi_i(\boldsymbol{x}_n) = \left(\boldsymbol{\Phi}^T(\boldsymbol{y}-\boldsymbol{t})\right)_i
\end{aligned}$$

with $\Phi_{nj} = \phi_j(\boldsymbol{x}_n)$ and $R_{n,n'} = y_n(1-y_n)\delta_{n,n'}$.

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

$$\boldsymbol{w}^{(new)} = \boldsymbol{w}^{(old)} - \left(\boldsymbol{\Phi}^T R \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T(\boldsymbol{y}-\boldsymbol{t})$$

Gaussian processes

(Regression setting). Data set $\{\boldsymbol{x}_n, t_n\}_{n=1:N}$ Assume a model

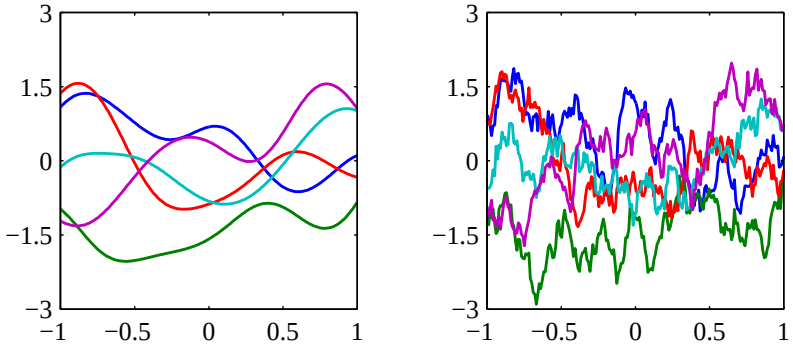
$$t_n = y_n + \epsilon_n \qquad \boldsymbol{t} = \boldsymbol{y} + \boldsymbol{\epsilon}$$

with $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\beta^{-1}\boldsymbol{I}_N)$ and

$$\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{y}|0, \boldsymbol{K})$$

$\boldsymbol{K}$ is a $N \times N$ covariance matrix that depends on the entire input data $\boldsymbol{x}_N = \boldsymbol{x}_{1:N}$ with components

$$K_{n,n'} = \theta_0 \exp\left(-\frac{\theta_1}{2}\|\boldsymbol{x}_n - \boldsymbol{x}_{n'}\|^2\right)$$



Since $\boldsymbol{t}_N = \boldsymbol{y} + \boldsymbol{\epsilon}$ we get $p(\boldsymbol{t}_N) = \mathcal{N}(\boldsymbol{t}_N|0, C_N)$ (with $\boldsymbol{t}_N = t_{1:N}$):

$$\mathbb{E}\boldsymbol{t}_N = 0 \qquad C_N = \mathbb{V}\boldsymbol{t}_N = \mathbb{V}\boldsymbol{y} + \mathbb{V}\boldsymbol{\epsilon} = \boldsymbol{K} + \beta^{-1}\boldsymbol{I}$$

Same holds when we add extra data point x, t: $\boldsymbol{t}_{N+1} = (\boldsymbol{t}_N, t)$ and $\boldsymbol{x}_{N+1} = (\boldsymbol{x}_N, \boldsymbol{x})$:

$$p(\boldsymbol{t}_{N+1}) = \mathcal{N}(\boldsymbol{t}_{N+1}|0, C_{N+1})$$

$p(\boldsymbol{t}_{N+1}) = p(t_N, t)$ is Gaussian. We can compute the marginal $p(t|\boldsymbol{t}_N)$ using results Eqs. 2.81-82

$$\begin{aligned} p(\boldsymbol{x}_a, \boldsymbol{x}_b) &= \mathcal{N}(\boldsymbol{x}_a, \boldsymbol{x}_b|\mu, \boldsymbol{\Sigma}) \qquad \mu = (\mu_a, \mu_b), \quad \Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix} \\ p(\boldsymbol{x}_a|\boldsymbol{x}_b) &= \mathcal{N}(\boldsymbol{x}_a|\mu_{a|b}, \Sigma_{a|b}) \\ \Sigma_{a|b} &= \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba} \qquad \mu_{a|b} = \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(\boldsymbol{x}_b - \mu_b) \end{aligned}$$

$\boldsymbol{x}_a = t, \boldsymbol{x}_b = \boldsymbol{t}_N, \mu_a = \mu_b = 0$ and

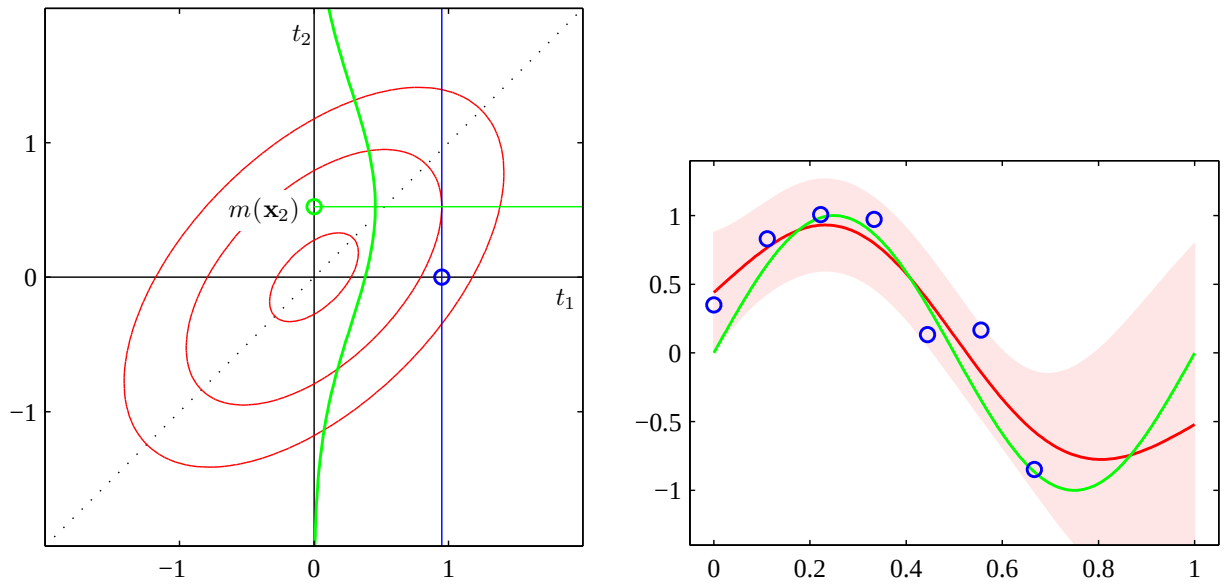
$$\Sigma = C_{N+1} = \begin{pmatrix} c & \boldsymbol{k}^T \\ \boldsymbol{k} & C_N \end{pmatrix}$$

with $c = K(\boldsymbol{x}, \boldsymbol{x}) + \beta^{-1}$ and $k_n = K(\boldsymbol{x}_n, \boldsymbol{x}), n = 1, \ldots, N$

Thus,

$$p(t|\boldsymbol{t}_N) = \mathcal{N}(t|\mu_{N+1}, \sigma^2_{N+1}), \qquad \mu_{N+1} = \boldsymbol{k}^T C_N^{-1} \boldsymbol{t}_N \quad \sigma^2_{N+1} = c - \boldsymbol{k}^T C_N^{-1} \boldsymbol{k}$$

NB: μ_{N+1} and σ^2_{N+1} depend on $\boldsymbol{x}$.



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 $p(t|\boldsymbol{t}_N)$ versus x

Learning the parameters

The GP depends on the choice of the kernel K and in particular on its parameter values θ. Given data $\boldsymbol{x}_N, \boldsymbol{t}_N$ we can learn the parameters by maximum likelihood.

Since $p(\boldsymbol{t}_N|\theta)$ is Gaussian the data likelihood

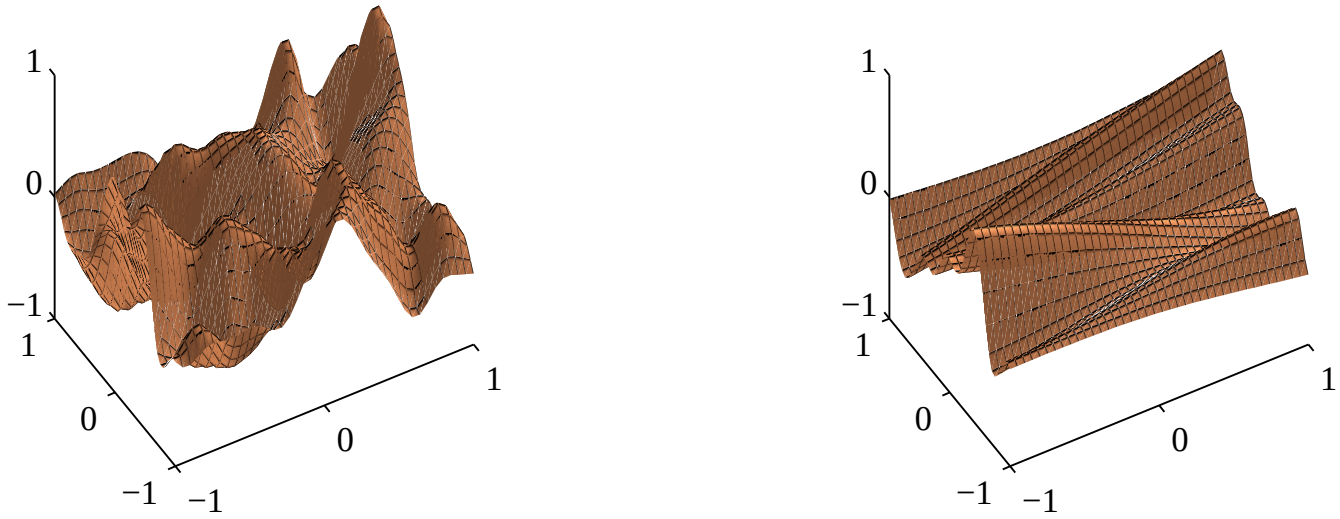
$$\log p(\boldsymbol{t}_N|\theta) = -\frac{1}{2}\log|C_N| - \frac{1}{2}\boldsymbol{t}_N^T C_N^{-1}\boldsymbol{t}_N - \frac{N}{2}\log 2\pi \qquad C_N = \boldsymbol{K}(\theta) + \beta^{-1}I$$

can be maximized by gradient ascent.

Automatic relevance determination

Choose different length scales for different dimensions

$$K(\boldsymbol{x}, \boldsymbol{x}') = \theta_0 \exp\left(-\frac{1}{2}\sum_{i=1}^{d} \eta_i (x_i - x_i')^2\right)$$

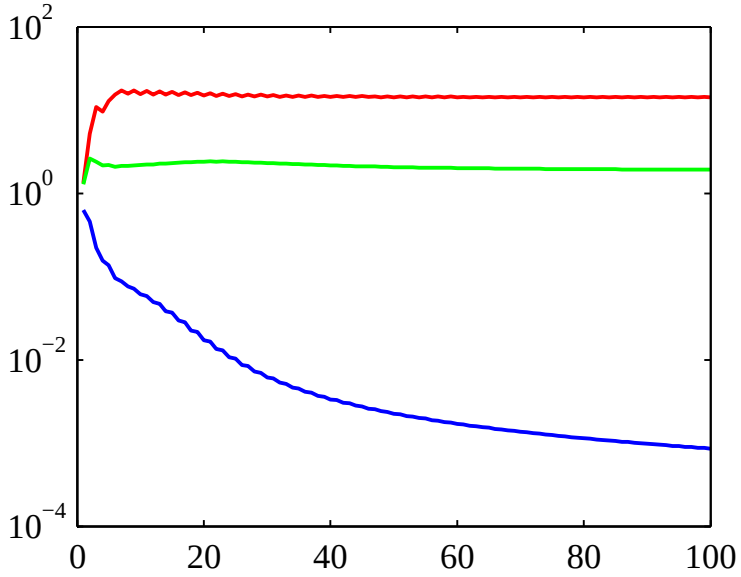


Samples of prior functions. Left $\eta_1 = \eta_2 = 1$. Right $\eta_1 = 1, \eta_2 = 0.01$.

Learn η_i (and θ_0) to discover relevant dimensions.

Example of ARD

Input is three dimensional $\mathbf{x} = (x_1, x_2, x_3)$.
x_1, x_3 sampled from a Gaussian. Output $t = \sin(2\pi x_1) + \epsilon$.
$x_2 = x_1 + \epsilon_1$



η_1 (red), η_2 (green) and η_3 (blue) versus learning iteration

Population variance explained by GWAS

Trait/Disease	H^2 Pedigree Studies	h^2 GWAS hits
Type I diabetes	0.9	0.6
Type II diabetes	0.3-0.6	0.05-0.1
Obesity (BMI)	0.4-0.6	0.01-0.02
Crohn's disease	0.6-0.8	0.1
Multiple Sclerosis	0.3-0.8	0.1
Schizophrenia	0.7-0.8	0.01
Bipolar disorder	0.6-0.7	0.02
ADHD	0.6-1.0	0

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

Data $\boldsymbol{t}_N, \boldsymbol{x}_N$ and ARD kernel $\boldsymbol{K}(\boldsymbol{\eta})$.

$$p(\boldsymbol{t}_N|\boldsymbol{\eta}) = \mathcal{N}(\boldsymbol{t}_N, C_N) \qquad C_n = \boldsymbol{K}(\boldsymbol{\eta}) + \beta^{-1} I$$

is Gaussian.

Sparse prior over lenght scales $\eta_{1:D}$: $p(\boldsymbol{\eta}|\theta)$ by proper choice of hyper parameters θ.

Compute Bayesian posterior

$$p(\boldsymbol{\eta}|\boldsymbol{t}_N, \theta) \propto p(\boldsymbol{t}_N|\boldsymbol{\eta}) p(\boldsymbol{\eta}|\theta)$$

Estimate $\mathbb{E}\eta_i = \int d\boldsymbol{\eta} \eta_i p(\boldsymbol{\eta}|\boldsymbol{t}_N, \theta)$ and variance by Monte Carlo sampling.

Simulation Data

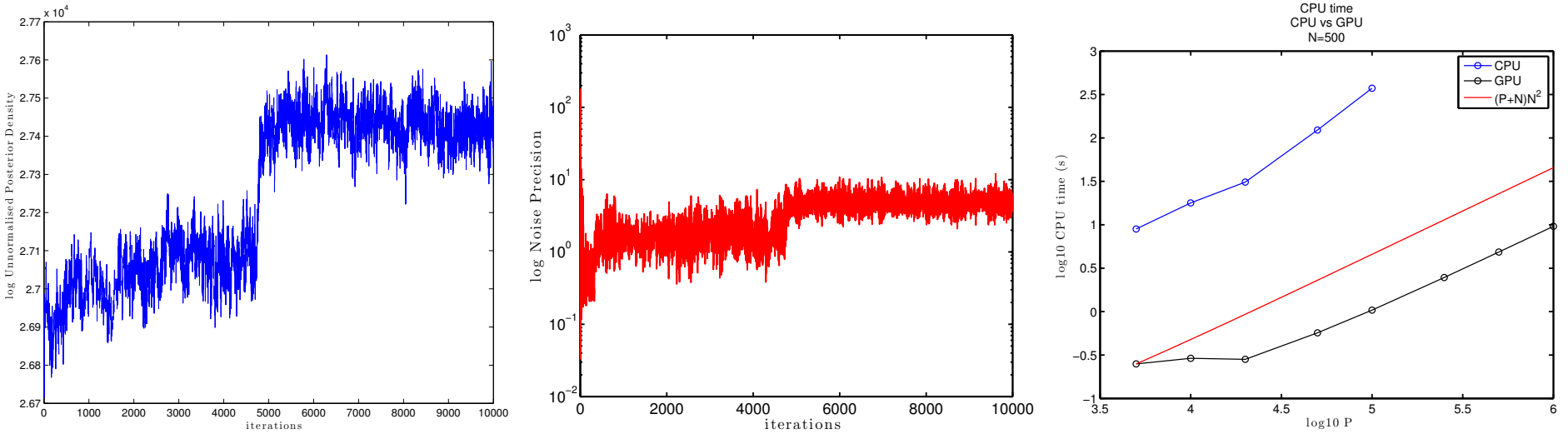
- Construct 500 genotype samples with 5000 snps from IMAGE-1

- 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

Monte Carlo

GP posterior computation uses Hamiltonian Monte Carlo sampling

- Run times hours to days.
- Multi-modality: hard to find high density region

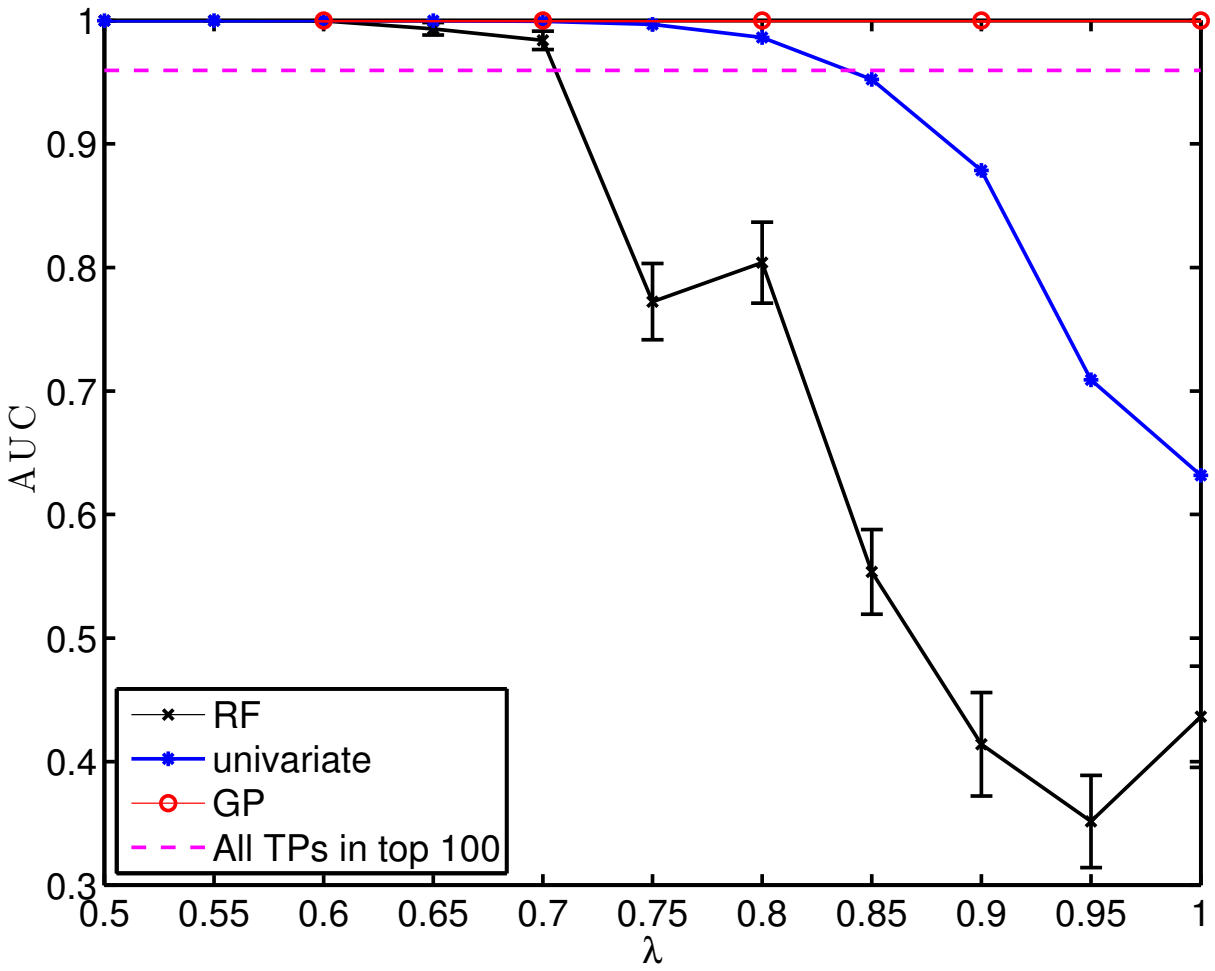


5-snp model; $\lambda = 1$; Signal/Noise$= 4/1$; 5000 snps. Left: Log posterior density. Middle: Log noise precision. Right: GPU vs CPU times

10^5 snps and 2000 samples in 20 seconds on GPU. Approx 300 times speed-up over CPU

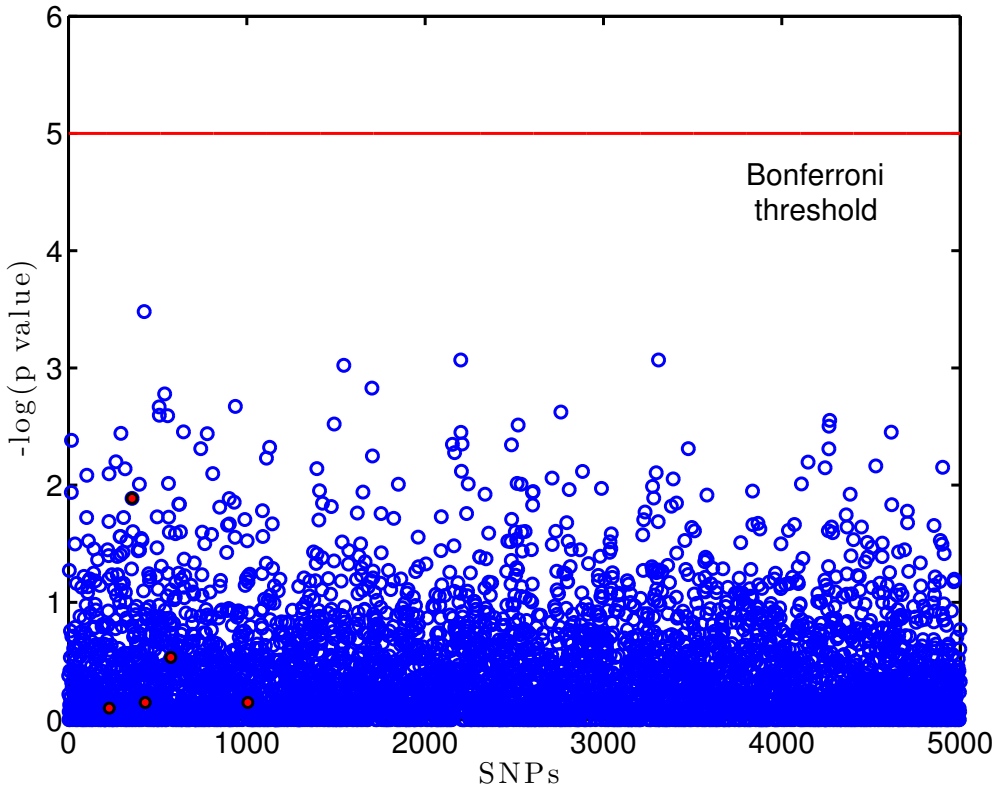
Classification performance

Compute feature strength for each SNP ($(\mathbb{E}\ \tau_k)^{-1}$ in case of GP)
Classify SNPs with highest strength as positive and rest as negative.
Construct ROC curve and compute AUC.

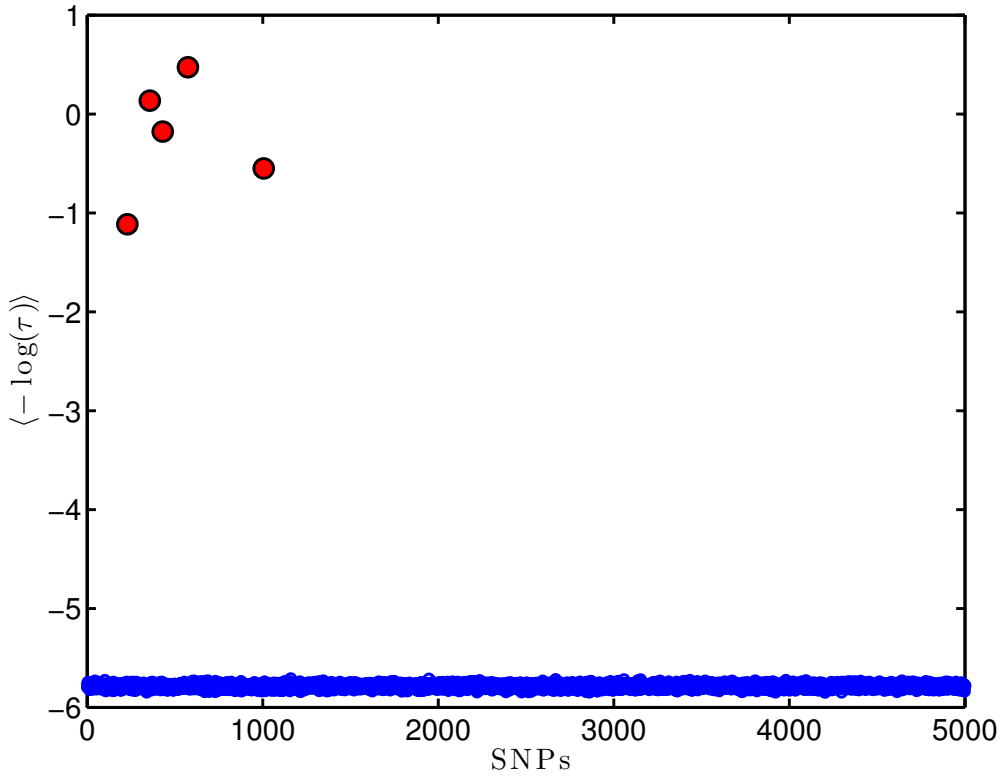


Significance - $\lambda = 1$

Univariate

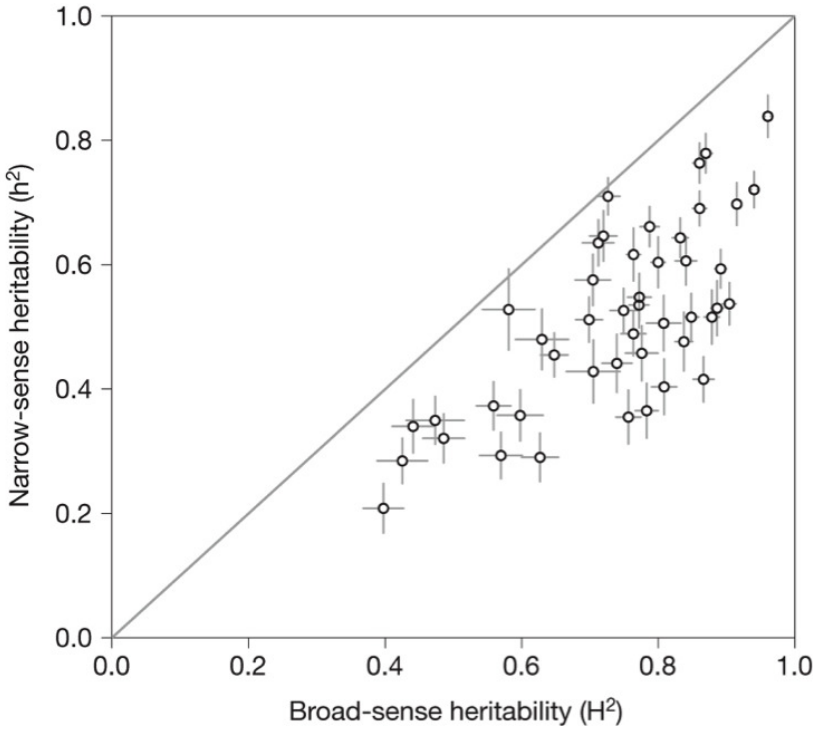


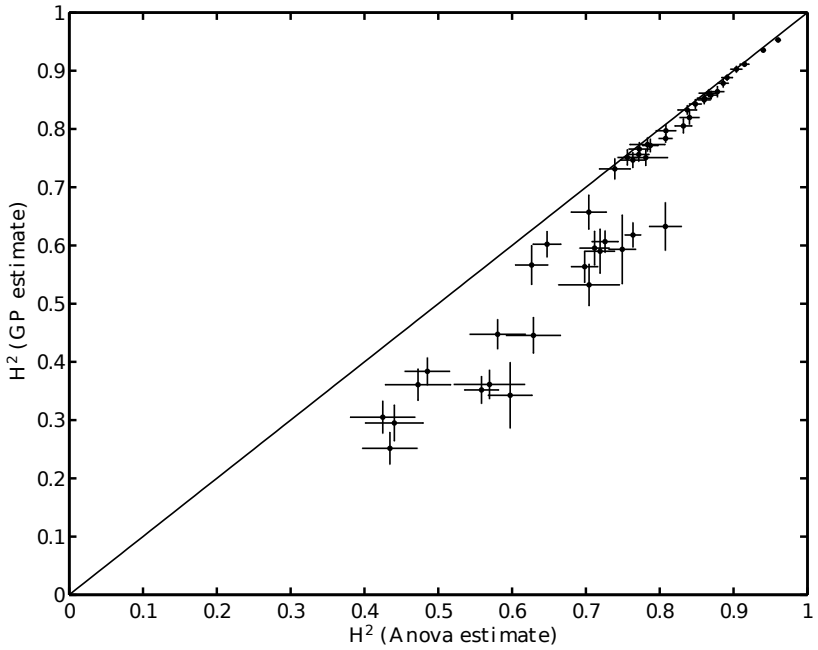
GP



Finding missing heritability in Yeast (Bloom et al., Nat. 2013.)

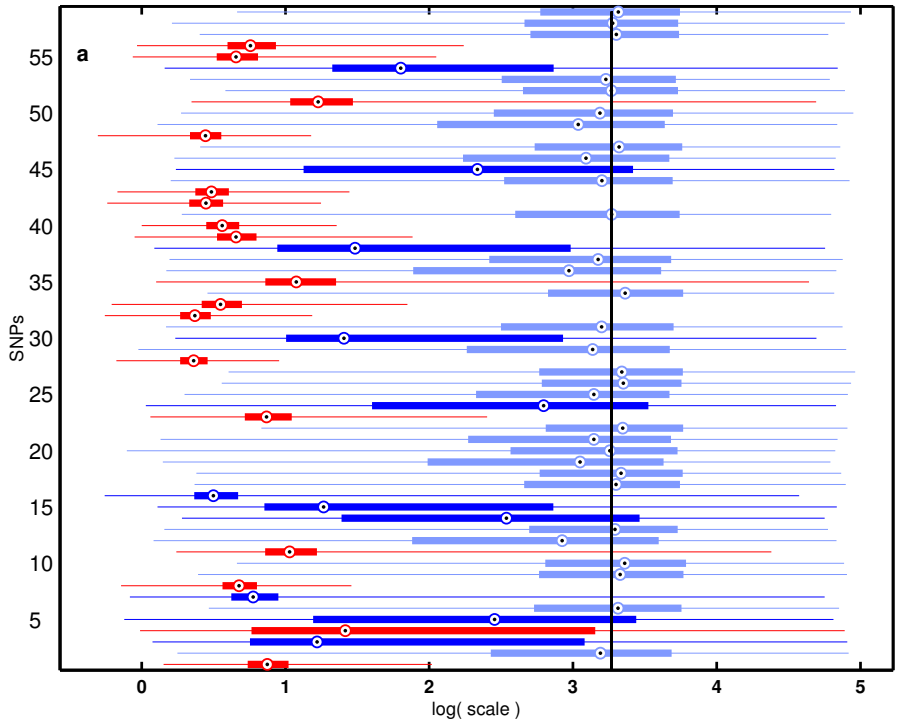
- 1008 genetically distinct inbred cell-lines. 46 different phenotypes
- Pairwise interactions explain additional 3 % median genetic variance per trait.



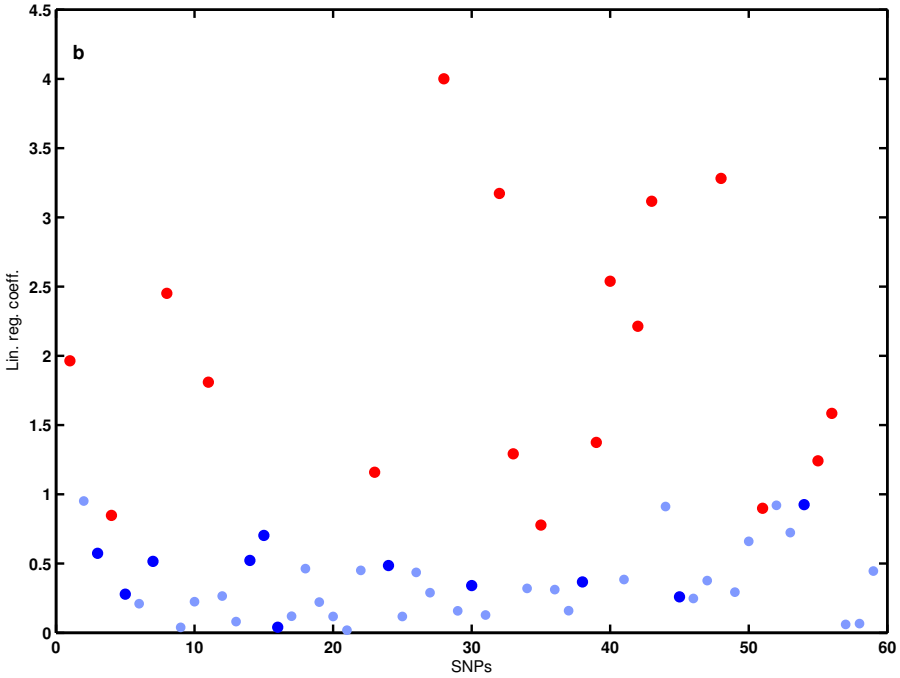


Identifying important variants - Zeocin

GPR - marginal posteriors



Linear regression coefficients



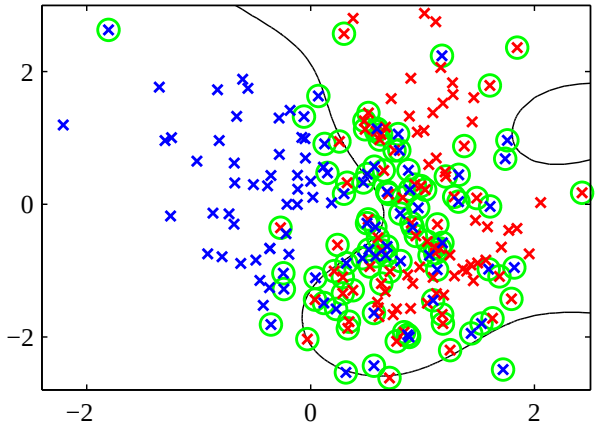
- red: GP and linear agree. dark blue: GP only
- SNP 16 found relevant by GPR, but has no additive effect.
- All missing heritability $\sim 35\%$ apparently explained by GPR

Support vector machines

Kernel methods (for classification): Training data $\{\boldsymbol{x}_n, t_n = \pm 1\}_{n=1:N}$

$$y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n t_n K(\boldsymbol{x}, \boldsymbol{x}_n) \qquad class(\boldsymbol{x}) = \text{sign}(y(\boldsymbol{x}))$$

select a subset of $a_n \geq 0$. This is a convex optimization.



Number of SVs increases with data

Network training

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

$$\begin{aligned} E(\boldsymbol{w}) &= \frac{1}{2}\sum_{n=1}^{N}(y(\boldsymbol{x}_n, \boldsymbol{w}) - t_n)^2 \\ &= -\log\prod_{n=1}^{N}\mathcal{N}(t_n|y(\boldsymbol{x}_n, \boldsymbol{w}), \beta^{-1}) + \ldots \end{aligned}$$

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

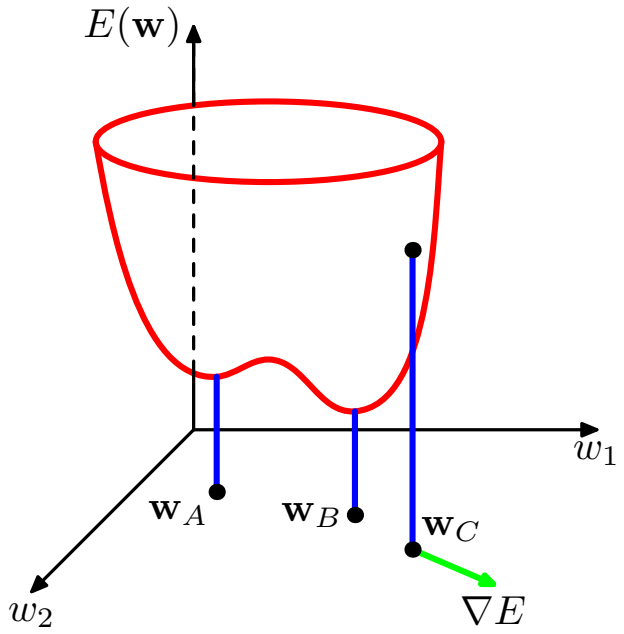
$$\begin{aligned} E(\boldsymbol{w}) &= -\sum_{n=1}^{N}\{t_n \log y(\boldsymbol{x}_n, \boldsymbol{w}) + (1 - t_n)\log(1 - y(\boldsymbol{x}_n, \boldsymbol{w}))\} \\ &= -\log\prod_{n=1}^{N} y(\boldsymbol{x}_n, \boldsymbol{w})^{t_n}(1 - y(\boldsymbol{x}_n, \boldsymbol{w}))^{1-t_n} \end{aligned}$$

Network training

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

$$
\begin{aligned}
E(\boldsymbol{w}) &= -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \log p_k(\boldsymbol{x}_n, \boldsymbol{w}) \\
p_k(\boldsymbol{x}, \boldsymbol{w}) &= \frac{\exp(y_k(\boldsymbol{x}, \boldsymbol{w}))}{\sum_{k'=1}^{K} \exp(y_{k'}(\boldsymbol{x}, \boldsymbol{w}))}
\end{aligned}
$$

Parameter optimization



E is minimal when $\nabla E(\boldsymbol{w}) = 0$, 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 $\boldsymbol{w}$ and iterate

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

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 $\Delta \boldsymbol{w} = -\eta \nabla E$.

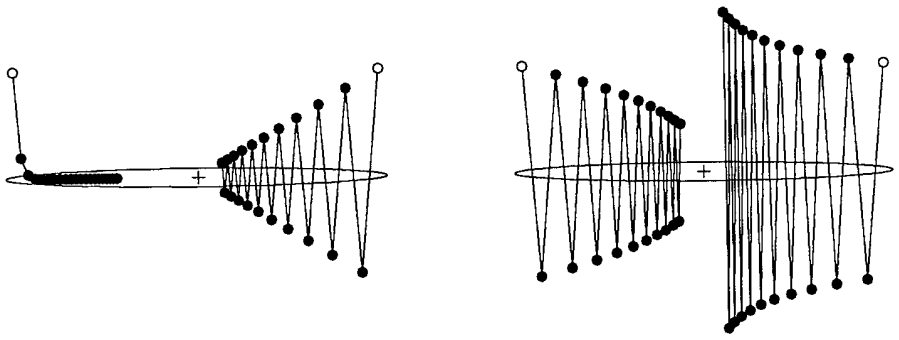
$$E(\boldsymbol{w} + \Delta \boldsymbol{w}) \approx E(\boldsymbol{w}) + (\Delta \boldsymbol{w})^T \nabla E = E(\boldsymbol{w}) - \eta \sum_i \left(\frac{\partial E}{\partial w_i}\right)^2 \leq E(\boldsymbol{w})$$

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

$$
\begin{aligned}
E(w) &= \frac{1}{2}\sum_i \lambda_i w_i^2 \\
\Delta w_i &= -\eta \frac{\partial E}{\partial w_i} = -\eta \lambda_i w_i \\
w_i^{\text{new}} &= w_i^{\text{old}} + \Delta w_i = (1 - \eta\lambda_i) w_i
\end{aligned}
$$

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:

$$\begin{aligned}\Delta w_{t+1} &= -\eta\nabla E(w_t) + \alpha\Delta w_t \\ &= -\eta\nabla E(w_t) + \alpha\left(-\eta\nabla E(w_{t-1}) + \alpha\left(-\eta\nabla E(w_{t-2}) + \ldots\right)\right) \\ &= -\eta\sum_{k=0}^{t}\alpha^k\nabla E(w_{t-k})\end{aligned}$$

Consider two extremes:

No oscillations all derivative are equal:

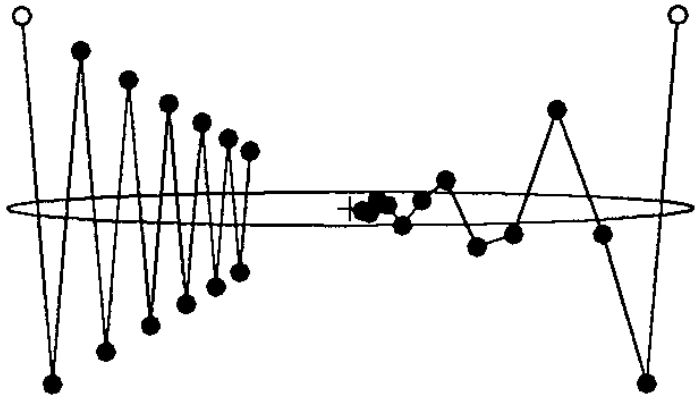
$$\Delta w_{t+1} \approx -\eta\nabla E\sum_{k=0}^{t}\alpha^k = -\frac{\eta}{1-\alpha}\frac{\partial E}{\partial w}$$

results in acceleration

Oscillations all derivatives are equal but have opposite sign:

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

results in decceleration



Newtons method

One can also use Hessian information for optimization. As an example, consider a quadratic approximation to E around $\boldsymbol{w}_0$:

$$\begin{aligned} E(\boldsymbol{w}) &= E(\boldsymbol{w}_0) + \boldsymbol{b}^T(\boldsymbol{w} - \boldsymbol{w}_0) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}_0)\boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}_0) \\ b_i &= \frac{\partial E(\boldsymbol{w}_0)}{\partial w_i} \qquad H_{ij} = \frac{\partial^2 E(\boldsymbol{w}_0)}{\partial w_i \partial w_j} \\ \nabla E(\boldsymbol{w}) &= \boldsymbol{b} + \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}_0) \end{aligned}$$

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

$$\boldsymbol{w} = \boldsymbol{w}_0 - \boldsymbol{H}^{-1}\nabla E(\boldsymbol{w}_0)$$

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:

$$w_1 = w_0 + \lambda_0 d_0, \qquad d_0 = -\nabla E(w_0)$$

$\lambda_0 > 0$ 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 $d_1' \cdot \nabla E(w_2) = 0$.

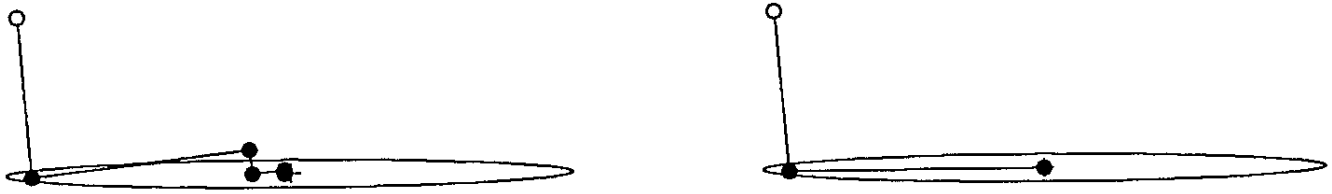
The direction d_1' is found by demanding that $\nabla E(w_2) \approx 0$ also in the 'old' direction d_0:

$$0 = d_0 \cdot \nabla E(w_2) \approx d_0 \cdot (\nabla E(w_1) + \lambda_1 H(w_1) d_1')$$

or

$$d_0 H(w_1) d_1' = 0$$

The subsequent search directions d_0, d_1' are said to be conjugate.



Polak-Ribiere rule

The conjugate directions can be computed without computing the Hessian matrix, for instance using the Polak-Ribiere rule:[8]

$$\beta = \frac{(\nabla E(w_1) - \nabla E(w_0)) \cdot \nabla E(w_1)}{\|\nabla E(w_0)\|^2}$$

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

$$d_i^T H d_j = 0 \qquad i, j = 1, \ldots, n$$

[8] 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

$$0 = \lambda_0 d_0^T H(w_1) d_1' = (\nabla E(w_1) - \nabla E(w_0)) \cdot (-\nabla E(w_1) + \beta d_0) = -(\nabla E(w_1) - \nabla E(w_0)) \cdot \nabla E(w_1) + \beta \|\nabla E($$

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 learing, where only one or a subset of training patterns is considered for computing ∇E.

$$E(\boldsymbol{w}) = \sum_n E_n(\boldsymbol{w}) \qquad \boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha_t \nabla E_n(\boldsymbol{w}^\tau)$$

May be efficient for large data sets. This results in a stochastic dynamics in $\boldsymbol{w}$ that can help to escape local minima.

Robbins Monro

Consider the problem to find x such that

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

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

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

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

$$p(\mu) = \frac{1}{P} \qquad a_i = 0 \qquad N(w, \mu) = E_{\mu}(w)$$

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

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

to a model where ϕ_j is adaptive:

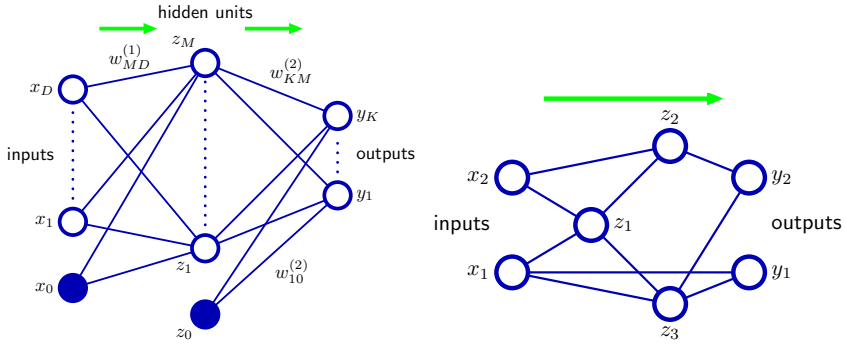
$$\phi_j(\boldsymbol{x}) = h(\sum_{i=0}^{D} w_{ji}^{(1)} x_i)$$

Feed-forward Network functions

In the case of K outputs

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

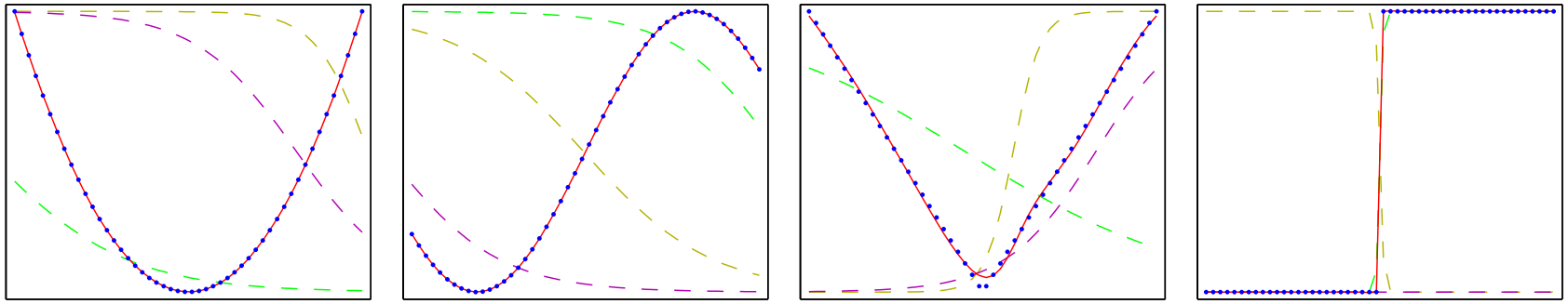
$h_2(x)$ is $\sigma(x)$ or x depending on the problem. $h_1(x)$ is $\sigma(x)$ or $\tanh(x)$.



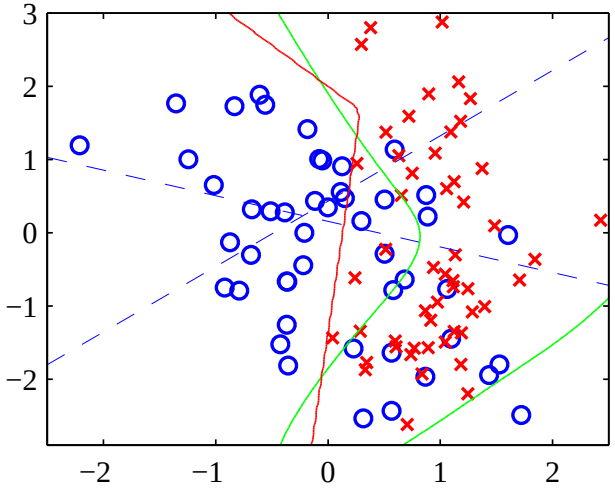
Left) Two layer architecture. Right) general feed-forward network with skip-layer connections.

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 $w_{ji} \to -w_{ji}$ and $w_{kj} \to -w_{kj}$: 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

$$
\begin{aligned}
E(\boldsymbol{w}) &= \sum_n E^n(\boldsymbol{w}) \qquad E^n(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{y}(\boldsymbol{x}_n, \boldsymbol{w}) - \boldsymbol{t}_n\|^2 \\
y_k(\boldsymbol{x}, \boldsymbol{w}) &= h_2\left(w_{k0} + \sum_{j=1}^{M} w_{kj} h_1\left(w_{j0} + \sum_{i=1}^{D} w_{ji} x_i\right)\right) \\
&= h_2(a_k) \\
a_k &= w_{k0} + \sum_{j=1}^{M} w_{kj} h_1(a_j) = \sum_{j=0}^{M} w_{kj} h_1(a_j) \qquad h_1(a_0) = 1 \\
a_j &= w_{j0} + \sum_{i=1}^{D} w_{ji} x_i = \sum_{i=0}^{D} w_{ji} x_i \qquad x_0 = 1
\end{aligned}
$$

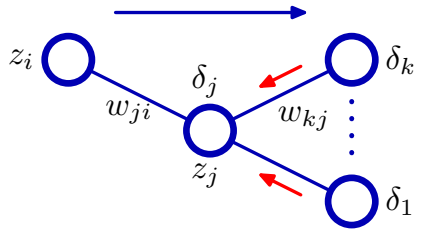
i labels inputs, j labels hiddens, k labels outputs.

Error backpropagation

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

$$
\begin{aligned}
y_k(\boldsymbol{x}^n, \boldsymbol{w}) &= h_2(a_k^n) = h_2\left(\sum_{j=0}^{M} w_{kj} h_1(a_j^n)\right) = h_2\left(\sum_{j=0}^{M} w_{kj} h_1\left(\sum_{i=0}^{D} w_{ji} x_i^n\right)\right) \\
\frac{\partial E^n}{\partial w_{kj}} &= (y_k^n - t_k^n)\frac{\partial y_k^n}{\partial w_{kj}} = (y_k^n - t_k^n) h_2'(a_k^n)\frac{\partial a_k^n}{\partial w_{kj}} = (y_k^n - t_k^n) h_2'(a_k^n) h_1(a_j^n) \\
&= \delta_k^n h_1(a_j^n) \qquad \delta_k^n = (y_k^n - t_k^n) h_2'(a_k^n) \\
\frac{\partial E^n}{\partial w_{ji}} &= \sum_{k=1}^{K} (y_k^n - t_k^n)\frac{\partial y_k^n}{\partial w_{ji}} = \sum_{k=1}^{K} (y_k^n - t_k^n) h_2'(a_k^n)\frac{\partial a_k^n}{\partial w_{ji}} \\
&= \sum_{k=1}^{K} \delta_k^n w_{kj} h_1'(a_j^n)\frac{\partial a_j^n}{\partial w_{ji}} = \sum_{k=1}^{K} \delta_k^n w_{kj} h_1'(a_j^n) x_i^n = \delta_j^n x_i^n \\
\delta_j^n &= h_1'(a_j^n) \sum_{k=1}^{K} \delta_k^n w_{kj}
\end{aligned}
$$

Error backpropagation



The back propagation extends to arbitrary layers:

1. $z_i^n = x_i^n$ *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 δ_k^n for the output units, and *back-propagate* the δ to obtain δ_j^n each hidden unit j

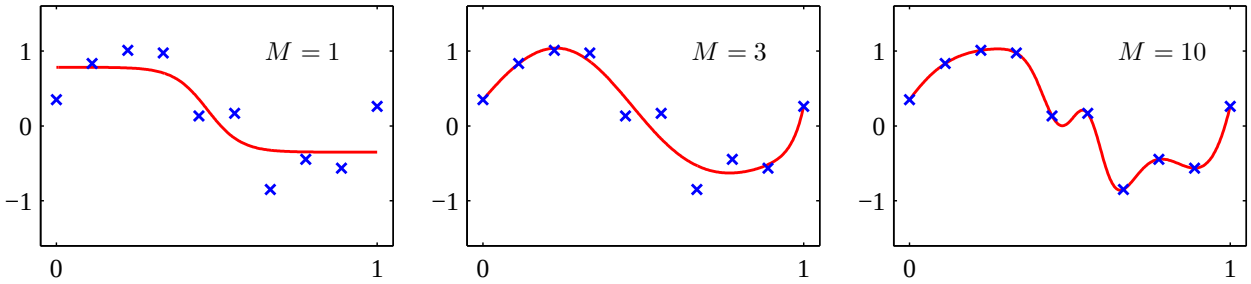
3. $\partial E^n / \partial w_{kj} = \delta_k^n z_j^n$ and $\partial E^n / \partial w_{ji} = \delta_j^n z_i^n$

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

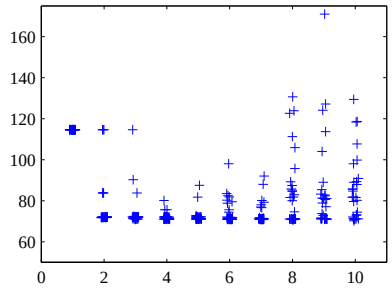
E is a function of $\mathcal{O}(|\boldsymbol{w}|)$ variables. In general, the computation of E requires $\mathcal{O}(|\boldsymbol{w}|)$ operations. The computation of ∇E would thus require $\mathcal{O}(|\boldsymbol{w}|^2)$ operations.

The backpropagation method allows to compute ∇E efficiently, in $\mathcal{O}(|\boldsymbol{w}|)$ operations.

Regularization



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

- chosing $\tanh$ instead of σ 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:

$$x^\mu \to c^\mu = \pm 1, \qquad x_i^\mu = \pm 1$$

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

$$\begin{aligned} w_{ji} &= b \quad \text{if } i\text{th digit in binary repr. of } j \text{ is 1} \\ w_{ji} &= -b \quad \text{else} \end{aligned}$$

j	binary	w_{j1}	w_{j2}
0	00	-b	-b
1	01	-b	b
2	10	b	-b
3	11	b	b

x_1	x_2	$\sum_i w_{0i}x_i$	$w_{1i}x_i$	$w_{2i}x_i$	$w_{3i}x_i$
-1	-1	**2b**	0	0	-2b
-1	1	0	**2b**	-2b	0
1	-1	0	-2b	**2b**	0
1	1	-2b	0	0	**2b**

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

x_1	x_2	z_0	z_1	z_2	z_3	c
-1	-1	1	0	0	0	sign$[w_0]$
-1	1	0	1	0	0	sign$[w_1]$
1	-1	0	0	1	0	sign$[w_2]$
1	1	0	0	0	1	sign$[w_3]$

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

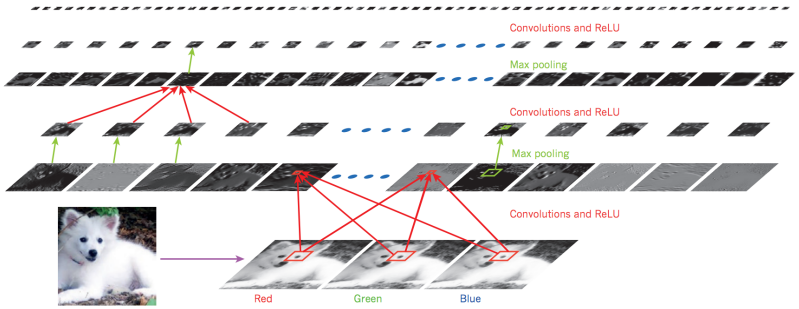
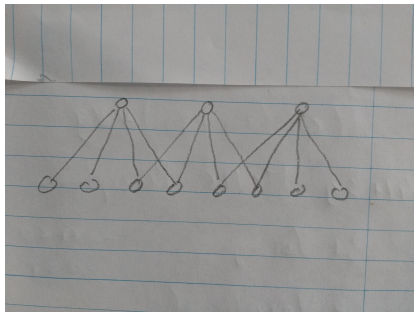
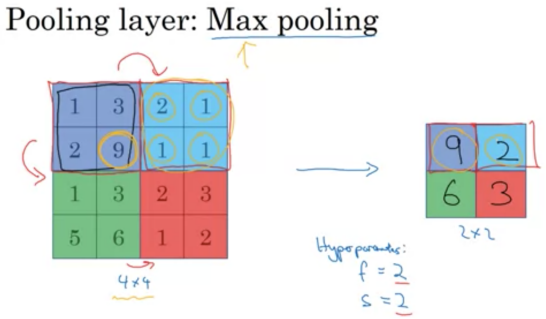


Figure 2 | Inside a convolutional network. The outputs (not the filters) of each layer (horizontally) of a typical convolutional network architecture applied to the image of a Samoyed dog (bottom left; and RGB (red, green, blue) inputs, bottom right). Each rectangular image is a feature map corresponding to the output for one of the learned features, detected at each of the image positions. Information flows bottom up, with lower-level features acting as oriented edge detectors, and a score is computed for each image class in output. ReLU, rectified linear unit.





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.

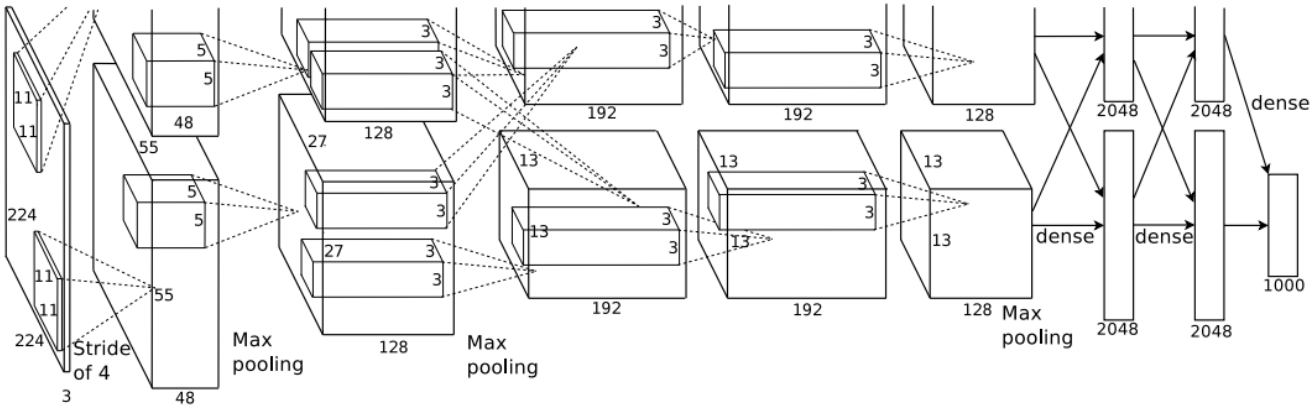
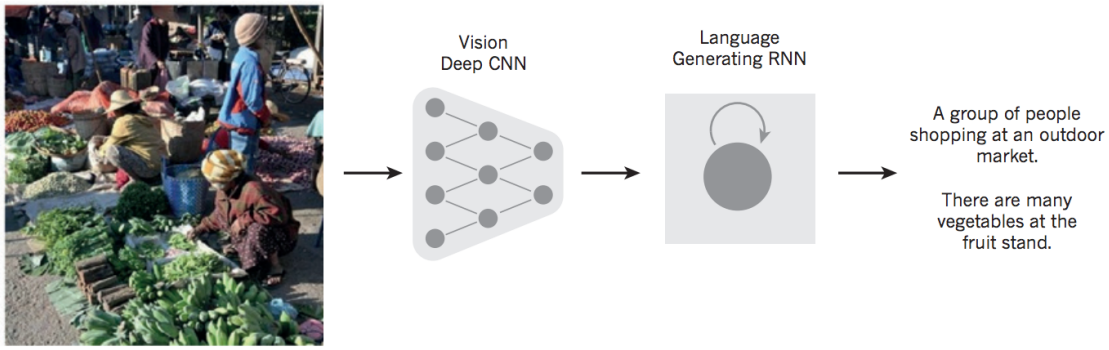


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input 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.

Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

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:
https://www.sciencedirect.com/science/article/pii/0031320382900243

LeCunn Convolutional networks 1990:
http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-
pdf

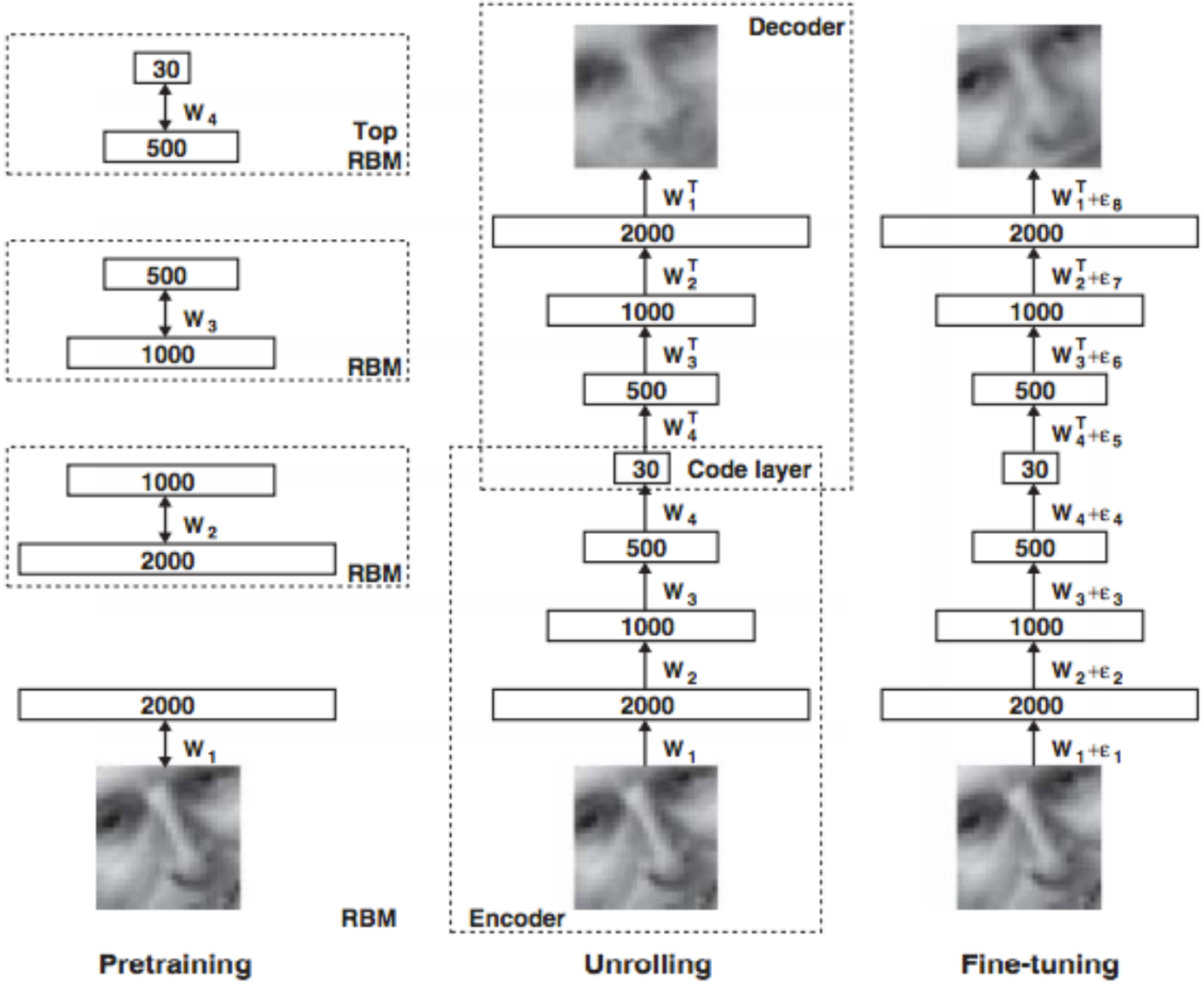
Krizhevsky Imagenet 2012:
https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neur
pdf

Vinyals Interpreting images 2014:http://arxiv.org/abs/1502.03044

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

[Hinton and Salakhutdinov, 2006]

Deep belief networks



[Hinton and Salakhutdinov, 2006]

Experiments: MNIST data

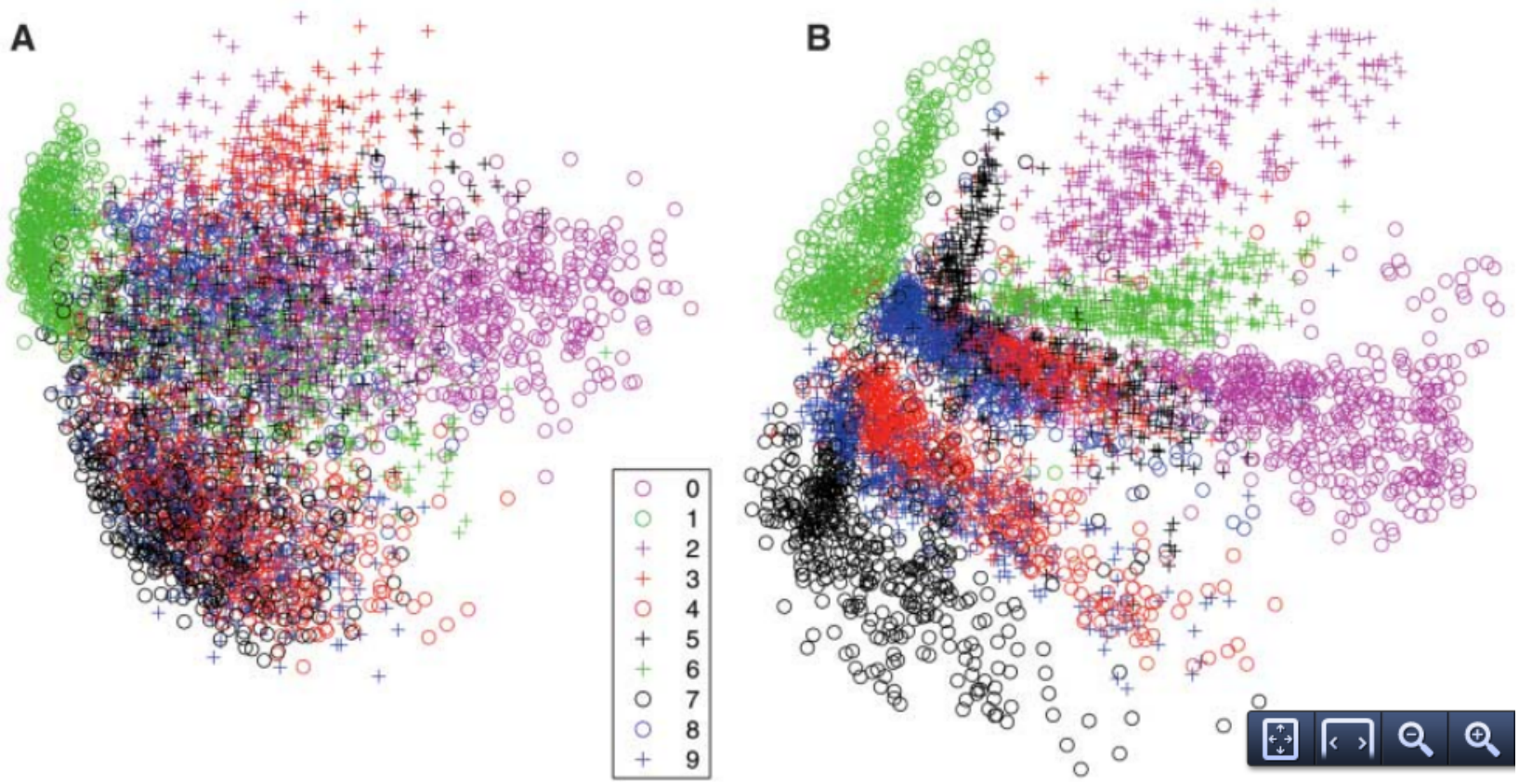
$784 - 1000 - 500 - 250 - 30$ autoencoder
60.000 training images and 10.000 test images

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.

[Hinton and Salakhutdinov, 2006]

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 (*8*).



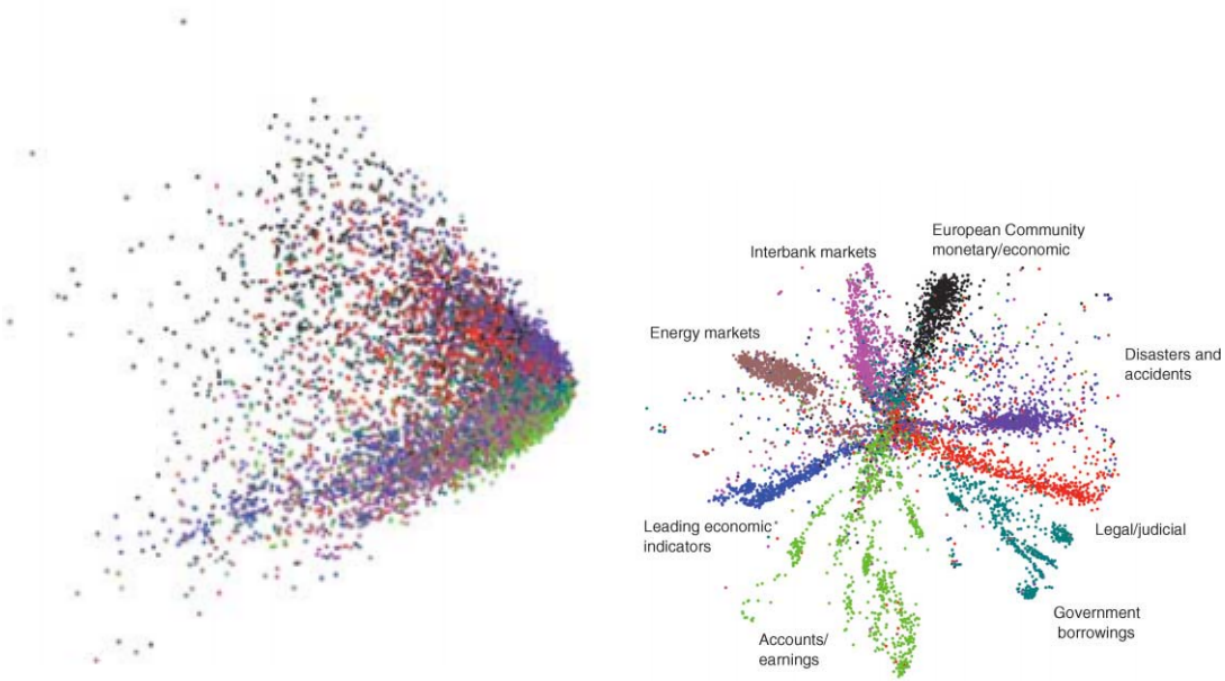
DBN yields better visualisation than PCA

[Hinton and Salakhutdinov, 2006]

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



Left: 2-dimensional LSA. Right: 2000- 500-250-125-2 DBN autoencoder.

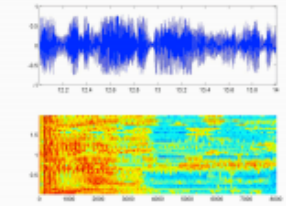
Mining for Structure

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

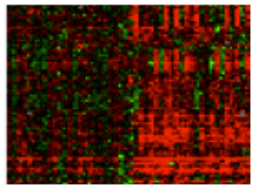
Images & Video

Text & Language

Speech & Audio



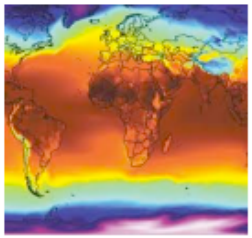
Gene Expression



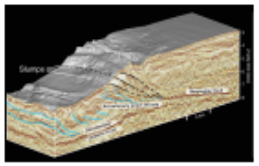
Product Recommendation

Relational Data/ Social Network

Climate Change

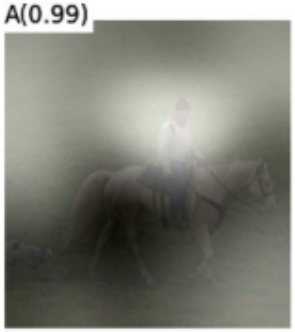


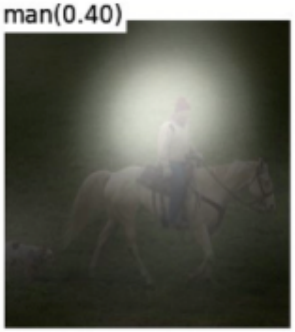
Geological Data

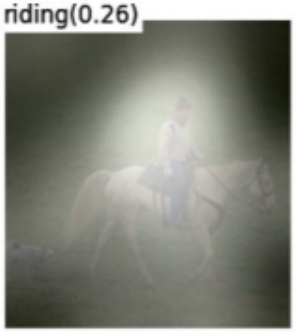


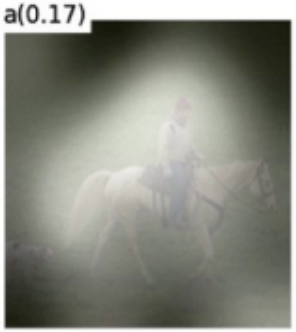
- Develop statistical models that can discover underlying structure, cause, or statistical correlation from data in **unsupervised** or **semi-supervised** way.
- Multiple application domains.

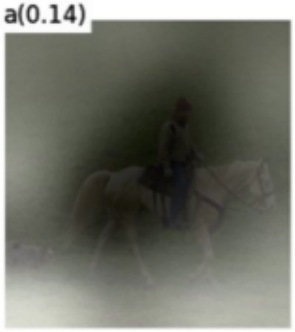
Caption Generation with Visual Attention











A man riding a horse in a field.

Xu et.al., ICML 2015

Challenges - I

Image

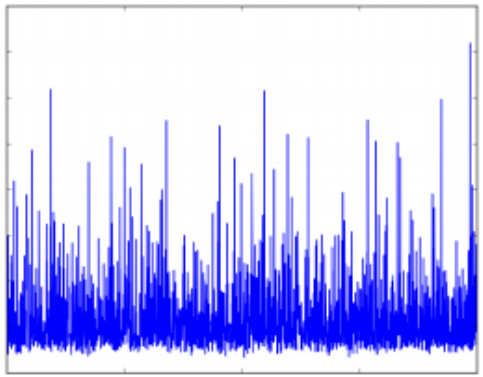
Text

sunset, pacific ocean, baker beach, seashore, ocean

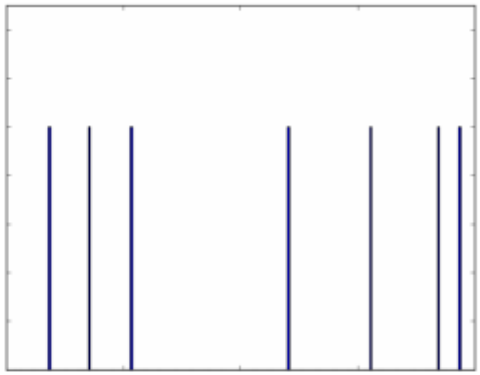
Very different input representations

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

Dense

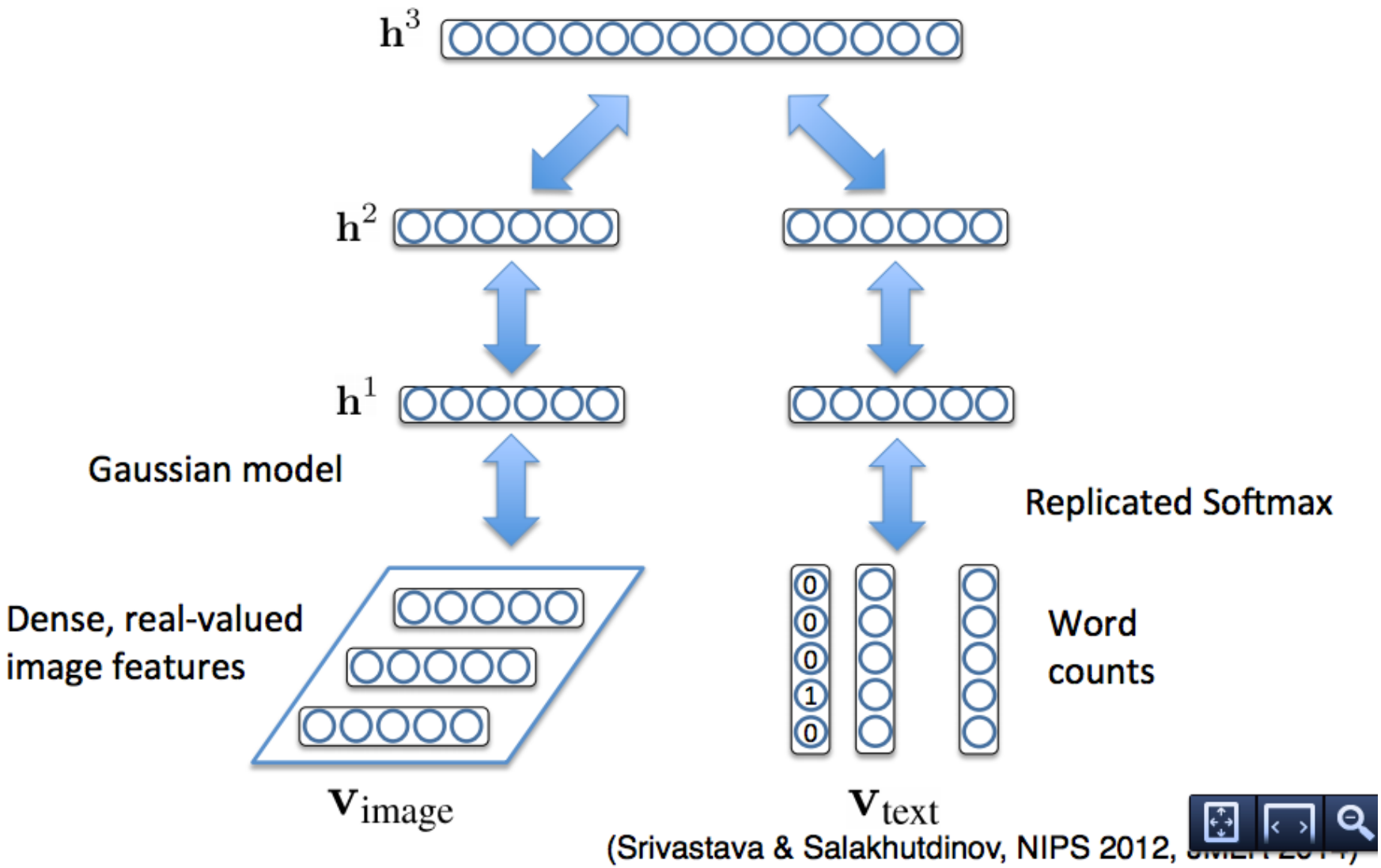


Sparse



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

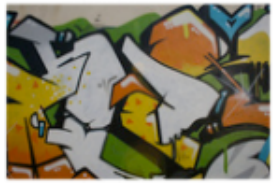
Multimodal DBM



(Srivastava & Salakhutdinov, NIPS 2012,

Text Generated from Images

Given	Generated
	dog, cat, pet, kitten, puppy, ginger, tongue, kitty, dogs, furry
	sea, france, boat, mer, beach, river, bretagne, plage, brittany
	portrait, child, kid, ritratto, kids, children, boy, cute, boys, italy

Given	Generated
	insect, butterfly, insects, bug, butterflies, lepidoptera
	graffiti, streetart, stencil, sticker, urbanart, graff, sanfrancisco
	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

[Mnih et al., 2015]

Human-level control through deep reinforcement learning

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

Partial derivatives, gradient, Nabla symbol

Let $f(x_1, \ldots, x_n) = f(\boldsymbol{x})$ be a function of several variables. The gradient of f, denoted as ∇f (the symbol " ∇" is called 'nabla') , is the vector of all partial derivatives:

$$\nabla f(\boldsymbol{x}) = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right)^T$$

NB, the partial derivative $\partial f(x_1, \ldots, x_i, \ldots, x_n)/\partial x_i$ 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 $\nabla f = 0$,

$$\left.\begin{array}{ll} 2x^* + y^* + 1 & = 0 \\ 2y^* + x^* & = 0 \end{array}\right\} \Rightarrow (x^*, y^*) = (-\frac{2}{3}, \frac{1}{3})$$

Chain rule

Suppose f is a function of $y_1, y_2, \ldots, y_k$ 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

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \sum_{j=1}^{k} \frac{\partial f}{\partial y_j} \frac{\mathrm{d}y_j}{\mathrm{d}x}$$

Example:

$$f(y(x), z(x)) = y(x)/z(x)$$

and $y(x) = x^4$ and $z = x^2$ then

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{1}{z(x)} y'(x) - \frac{y(x)}{z(x)^2} z'(x) = 2x$$

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:

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{j=1}^{N} \left(y_j(\boldsymbol{w}) - t_j \right)^2 \quad \Rightarrow \quad \frac{\partial E}{\partial y_j} = y_j - t_j$$

and

$$y_j(\boldsymbol{w}) = \sum_{i=0}^{M} x_j^i w_i \quad \Rightarrow \quad \frac{\partial y_j}{\partial w_i} = x_j^i$$

So

$$\frac{\partial E}{\partial w_i} = \sum_{j=1}^{N} (y_j(\boldsymbol{w}) - t_j) x_j^i$$

t_j and x_j^j are parameters in this example.

Matrix multiplications as summations

If $\mathbf{A}$ is a $N \times M$ matrix with entries A_{ij} and $\mathbf{v}$ an M-dimensional vector with entries v_i, then $\mathbf{w} = \mathbf{A}\mathbf{v}$ is a N-dimensional vector with components

$$w_i = \sum_{j=1}^{M} A_{ij} v_j$$

Similarly, if $\mathbf{B}$ is a $M \times K$ matrix with entries B_{ij}, then $\mathbf{C} = \mathbf{A}\mathbf{B}$ 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 interchance 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,

$$w_k = \sum_{j=1}^{M} \sum_{i=1}^{N} A_{ij} B_{jk} = \sum_{l=1}^{N} \sum_{i=1}^{M} B_{ik} A_{li}$$

Kronecker delta

The notation δ_{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:

$$\sum_{j=1}^{M} \delta_{ij} v_j = v_i \quad \text{for all} \;\; 1 \leq i \leq M \tag{3}$$

The Kronecker delta can be viewed as the entries of the identity matrix $\mathbf{I}$. In vector notation, (3) is equivalent to the statement $\mathbf{Iv} = \mathbf{v}$. In other words, $\delta_{ij} = I_{ij}$[9]

[9] Bishop used δ 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:

$$f(a+x) = f(a) + xf'(a)$$

Taylor series, examples

Examples: check that for small x the following expansions are correct up to second order:

$$\begin{aligned}
\sin(x) &= x \\
\cos(x) &= 1 - \frac{1}{2}x^2 \\
\exp(x) &= 1 + x + \frac{1}{2}x^2 \\
(1+x)^c &= 1 + cx + \frac{c(c-1)}{2}x^2 \\
\ln(1+x) &= x - \frac{1}{2}x^2
\end{aligned}$$

Taylor expansion in several dimensions

The Taylor expansion of a function of several variables, $f(x_1, \ldots, x_n) = f(\boldsymbol{x})$ is (up to second order)

$$f(\boldsymbol{x}) = f(\mathbf{a}) + \sum_i (x_i - a_i)\frac{\partial}{\partial x_i} f(\mathbf{a}) + \frac{1}{2}\sum_{ij}(x_i - a_i)(x_j - a_j)\frac{\partial}{\partial x_i}\frac{\partial}{\partial x_j} f(\mathbf{a})$$

or in vector notation, with $\epsilon = \boldsymbol{x} - \mathbf{a}$

$$f(\mathbf{a} + \epsilon) = f(\mathbf{a}) + \epsilon^T \nabla f(\mathbf{a}) + \frac{1}{2}\epsilon^T \boldsymbol{H} \epsilon$$

with $\boldsymbol{H}$ the Hessian, which is the symmetric matrix of partial derivatives

$$H_{ij} = \frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x})\bigg|_{\boldsymbol{x}=\mathbf{a}}$$

Integration

The integral of a function of several variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$

$$\int_{\mathcal{R}} f(\mathbf{x}) \mathrm{d}\mathbf{x} \equiv \int_{\mathcal{R}} f(x_1, x_2, \ldots, x_n) \mathrm{d}x_1 \mathrm{d}x_2 \ldots \mathrm{d}x_n$$

is the volume of the $n+1$ dimensional region lying 'vertically above' the domain of integration $\mathcal{R} \subset I\!R^n$ 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,

$$\int_{x=a}^{b}\int_{y=c}^{d} f(x)g(y)\,\mathrm{d}x\mathrm{d}y = \int_{x=a}^{b} f(x)\,\mathrm{d}x \int_{y=c}^{d} g(y)\,\mathrm{d}y$$

Example,

$$\int \exp\left(\sum_{i=1}^{n} f_i(x_i)\right)\mathrm{d}\boldsymbol{x} = \int \prod_{i=1}^{n} \exp(f_i(x_i))\,\mathrm{d}\boldsymbol{x} = \prod_{i=1}^{n} \int \exp(f_i(x_i))\,\mathrm{d}x_i$$

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

$$\int_{\mathcal{R}} f(y,x)\,\mathrm{d}y\mathrm{d}x = \int_{x=a}^{b} \left[\int_{y=c(x)}^{d(x)} f(y,x)\,\mathrm{d}y \right] \mathrm{d}x$$

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

$$\int_{y=c(x)}^{d(x)} f(y,x)\,\mathrm{d}y = F(d(x),x) - F(c(x),x)$$

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,

$$\int_{x=a}^{b} \left[\int_{y=c(x)}^{d(x)} f(y,x)\,\mathrm{d}y \right] \mathrm{d}x = \int_{x=a}^{b} g(x)\,\mathrm{d}x$$

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

$$\int_{\mathcal{R}} f(y,x)\,\mathrm{d}x\mathrm{d}y = \int_{y=s}^{t} \left[\int_{x=u(y)}^{v(y)} f(y,x)\,\mathrm{d}x \right] \mathrm{d}y$$

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 $\boldsymbol{x}$ 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

$$\delta(x) = 0 \text{ if } x \neq 0 \quad \text{and} \quad \int_{-\infty}^{\infty} \delta(x)\mathrm{d}x = 1$$

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

$$\int_{-\infty}^{\infty} f(x)\delta(x - x_0)\mathrm{d}x = f(x_0)$$

(cf. Kronecker delta $\sum_j \delta_{ij} v_j = v_i$).

The multivariate deltafunction factorizes over the dimensions

$$\delta(\boldsymbol{x} - \mathbf{m}) = \prod_{i=1}^{n} \delta(x_i - m_i)$$

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-handside 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):

$$F[y] = \int p(x)y(x)\,\mathrm{d}x$$

(Compare with $f(\mathbf{y}) = \sum_i p_i y_i$).

Other (nonlinear) example:

$$F[y] = \int \frac{1}{2}(y'(x) + V(x))^2\,\mathrm{d}x$$

References

References

[Cybenko, 1989] Cybenko, G. (1989). Approximation by superposition of a sigmoidal function. *Math. Control Signals Systems, Vol. 2*, pages 303–314.

[Hinton and Salakhutdinov, 2006] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

[Hornik et al., 1989] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

[Press et al., 1996] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1996). *Numerical recipes in C*, volume 2. Cambridge university press Cambridge.

[Sohl-Dickstein et al., 2013] Sohl-Dickstein, J., Poole, B., and Ganguli, S. (2013). Fast large-scale optimization by unifying stochastic gradient and quasi-newton methods. *arXiv preprint arXiv:1311.2115*.