Statistical Machine Learning

Lecturers: Bert Kappen, Tom Claassen
Radboud University Nijmegen, Netherlands

December 19, 2017

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`

Statistical Machine Learning

- Course setup
  - Lectures (incl. guest lecture), Wed 10:45-12:30, HG00.308 (until 25/10)
    from 14/11: Tue 13:45-15:30, HG00.310
  - Tutorials, Fri 13:45-15:30, HG00.310 (until 27/10),
    from 16/11: Thu 08:45-10:30, HG00.310
  - Instructors: Gabriel Bucur, Jordi Riemens
  - Textbook *"Pattern Recognition and Machine Learning"* (C.M Bishop, 2006)
  - All other course materials (slides, exercises, sample exams) via Blackboard
  - Homework assignments (4 in total, starting week 3)
  - Mini seminar (mandatory)
  - Written exam Thu. 25 Jan. 2018, 13:30-15:30, HG00.068
    (open book, one 'cheat sheet'; no other notes/slides/laptops/etc.)

- 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 6: Kernel methods

Chapter 9: Mixture models and EM

Chapter 1: Introduction

Introduction ML

- General introduction
- Polynomial curve fitting, regression, overfitting, regularization
- Probability theory, decision theory
- Information theory
- Math tools recap

Recognition of digits

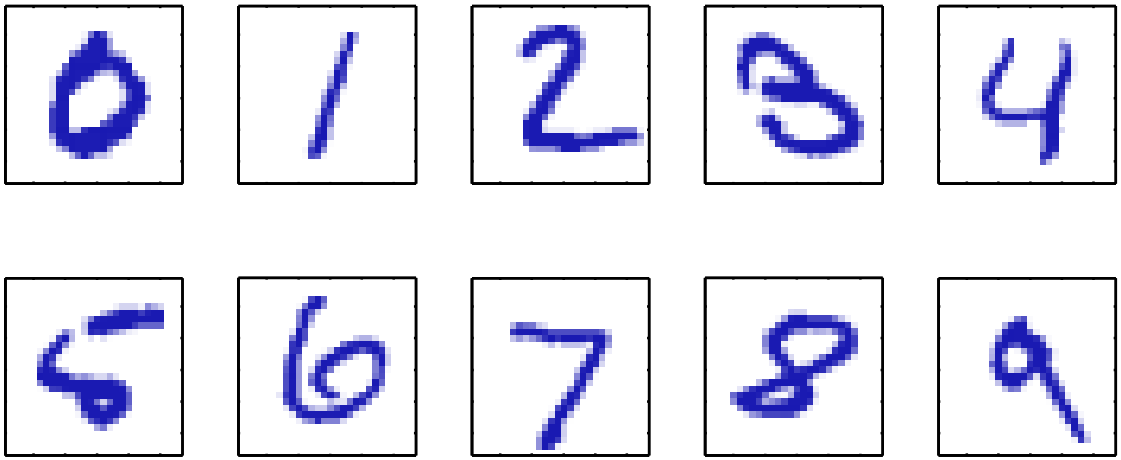


Image is array of pixels x_i, each between 0 and 1.
$\boldsymbol{x} = (x_1, \ldots, x_d)^{\mathrm{T}}$, vector with length d (the total number of pixels, e.g. $d = 28 \times 28$).

- Goal = input: pixels $\rightarrow$ output: correct category $0, \ldots, 9$
- wide variability
- brute force / 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
  - adaptive model
  - learning $=$ tuning model parameters on training set
- Result: function $\mathbf{y}(\mathbf{x})$
  - Fitted to target data
  - New input $\mathbf{x} \rightarrow$ output $\mathbf{y}(\mathbf{x})$
  - Hopefully: $\mathbf{y}(\mathbf{x}) \approx \mathbf{t}(\mathbf{x})$
  - Goal: generalization to new examples (use test set)

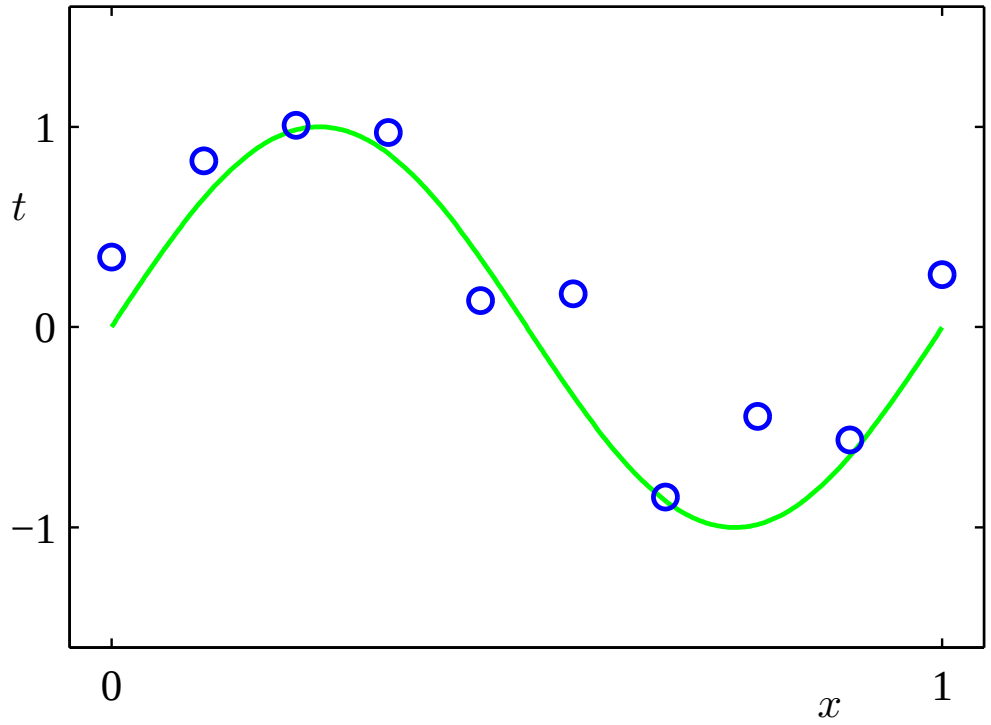
Preprocessing

- transformation of inputs $\mathbf{x} \rightarrow \mathbf{x}'$
  - easier to handle
  - speed up computation
  - training/test set + new instances
- by hand or rule based
  - scaling, centering
  - aspect ratio, greyscale
- 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: known targets
  - Classification: targets are classes
  - Regression: target is continuous
- Unsupervised learning: unknown targets
  - clustering (similarity between input data)
  - density estimation (distribution of input data)
  - dimension reduction (to 2/3D for visualization)
- Reinforcement learning: find optimal target / strategy
  - actions leading to maximal reward
  - exploration vs. exploitation

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)
  - function $\hat{t} = y(x)$

We will fit the data using M-th order polynomial

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

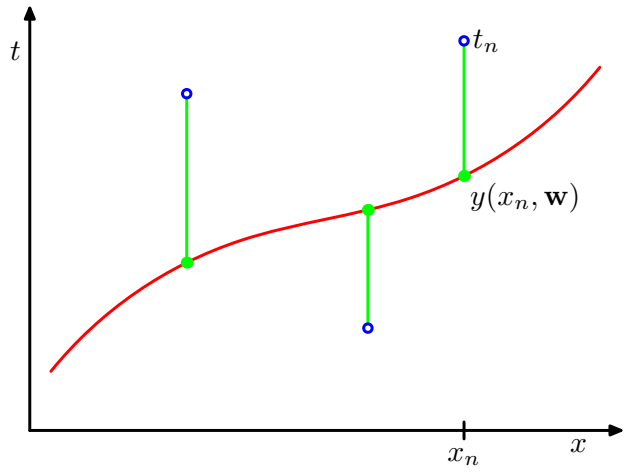
$y(x, \boldsymbol{w})$ is nonlinear in x, but linear in coefficients $\boldsymbol{w}$, "*Linear model*"

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



Finding the minimum: partial derivatives and gradient

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

$$\begin{aligned} \nabla f(x, y, z) &= \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)^T \\ &= \left(y^2, 2xy + 3.1z, 3.1y \right)^T \end{aligned}$$

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 $y_1, y_2, ..., y_k$ 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:

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

with $y(x) = x^4$ and $z = x^2$.

Then $y'(x) = 4x^3$ and $z'(x) = 2x$, and so

$$\begin{aligned} \frac{\mathrm{d}f}{\mathrm{d}x} &= \frac{1}{z(x)} y'(x) - \frac{y(x)}{z(x)^2} z'(x) \\ &= \frac{1}{x^2} 4x^3 - \frac{x^4}{(x^2)^2} 2x \\ &= 2x \end{aligned}$$

Chain rule (2)

Suppose E is a function of $y_1, y_2, ..., y_N$ 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.

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)

Matrix multiplications as summations

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

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

In general, if $\boldsymbol{B}$ is a $M \times K$ matrix with entries B_{ij}, then $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B}$ 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,

$$\sum_{i=1}^{N} \sum_{j=1}^{M} \cdots = \sum_{j=1}^{M} \sum_{i=1}^{N} \cdots$$

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:

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

The Kronecker delta can be viewed as the entries of the identity matrix $\boldsymbol{I}$. In vector notation, (**??**) is equivalent to the statement $\boldsymbol{I}\boldsymbol{v} = \boldsymbol{v}$. In other words, $\delta_{ij} = I_{ij}$[1]

[1] 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:

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

Taylor series, examples

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

$$\begin{aligned}
\sin(x) &= \sin(0) + x\cos(0) + \frac{1}{2}x^2(-\sin(0)) + ... \\
&= 0 + x - 0 + ... \\
&= x \\
\cos(x) &= 1 - \frac{1}{2}x^2 \\
\exp(x) &= 1 + x + \frac{1}{2}x^2 \\
(1+x)^c &= 1 + cx + \frac{c(c-1)}{2}x^2 \\
\ln(1+x) &= x - \frac{1}{2}x^2
\end{aligned}$$

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)

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

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

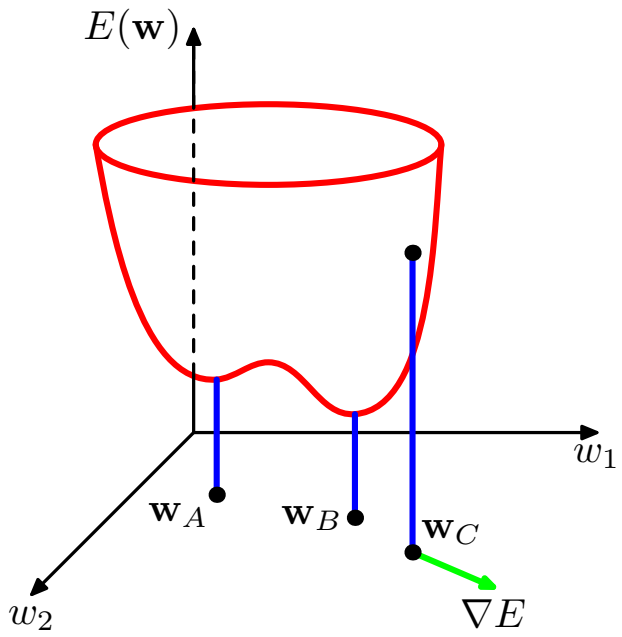
$$f(\boldsymbol{a} + \boldsymbol{\epsilon}) = f(\boldsymbol{a}) + \boldsymbol{\epsilon}^T \nabla f(\boldsymbol{a}) + \frac{1}{2} \boldsymbol{\epsilon}^T \boldsymbol{H} \boldsymbol{\epsilon}$$

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

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

Error landscape

Polynomial curve fitting has a quadratic error function. In general the error function $E(\boldsymbol{w})$ may be a non-quadratic in the parameters $\boldsymbol{w}$.



Gradient descent: walk downwards with small steps in the direction of the negative gradient.

E is minimal when $\nabla E(\boldsymbol{w}) = 0$, but not vice versa!

$\Rightarrow$ gradient based methods find a local minimum, not necessary the global minimum.

Application: Optimization by gradient descent

Gradient descent algorithm for finding $\boldsymbol{w}$ in $y(x, \boldsymbol{w})$:

1. Start with an initial value of $\boldsymbol{w}$ and ϵ small.

2. While "change in $\boldsymbol{w}$ large": Compute $\boldsymbol{w} := \boldsymbol{w} - \epsilon \nabla E$

Stop criterion is $\nabla E \approx 0$, which means that we stop in a local minimum of E.

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

Proof: Denote $\Delta \boldsymbol{w} = -\epsilon \nabla E$.

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

In each gradient descent step the value of E is lowered. Since E bounded from below, the procedure must converge asymptotically.

Note: if $\boldsymbol{w}$ has a closed-form solution then gradient descent not necessary!

Newtons method

One can also use Hessian information for optimization. As an example, consider a quadratic approximation to E around $\boldsymbol{w}_0$ (Taylor expansion up to 2^{nd} order):

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

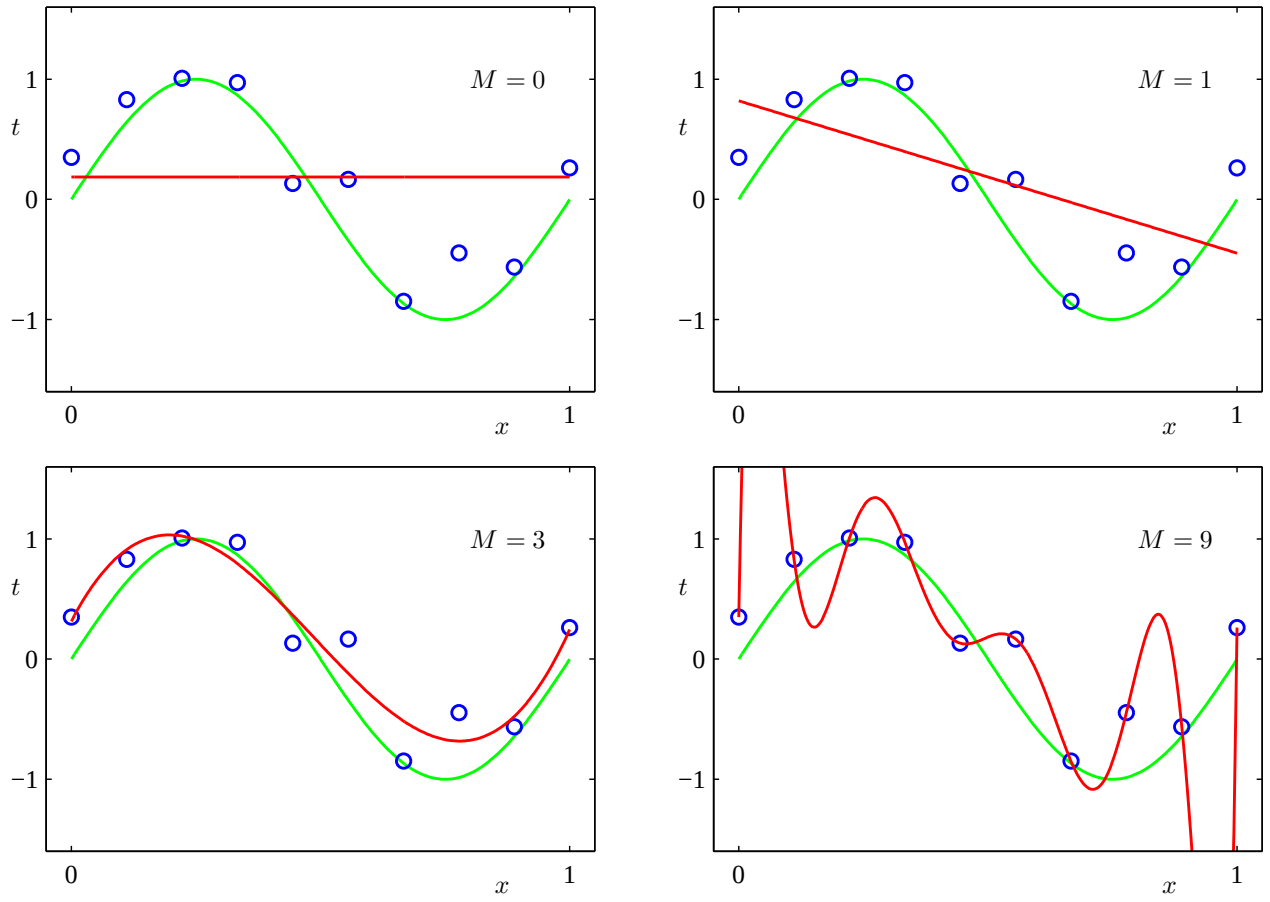
We can solve $\nabla E(\boldsymbol{w}) = 0$ and obtain

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

This is called Newtons method. Inversion of the Hessian may be computational costly. A number of methods, known as quasi-newton methods, are based on approximations of this procedure.

Model comparison, model selection

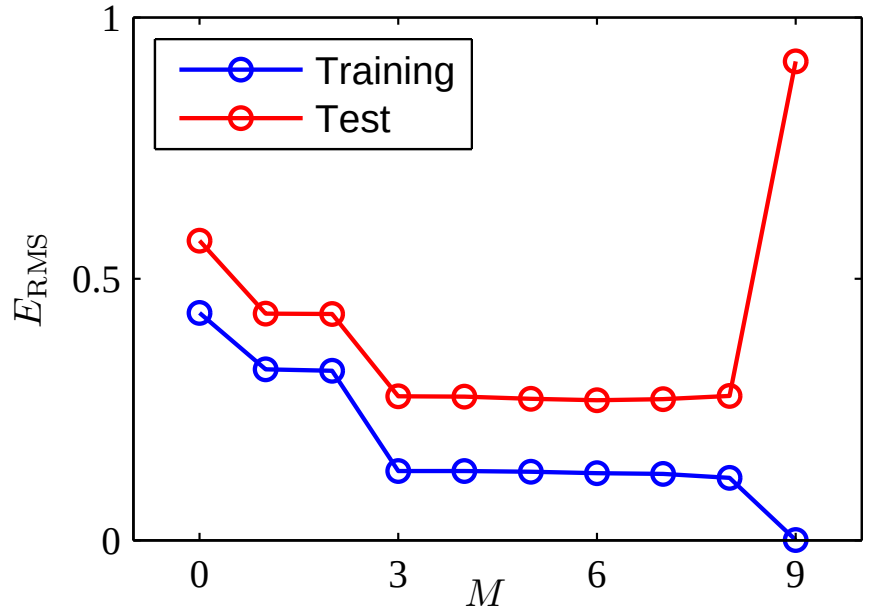
Back to polynomial curve fitting: how to choose M?



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:

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



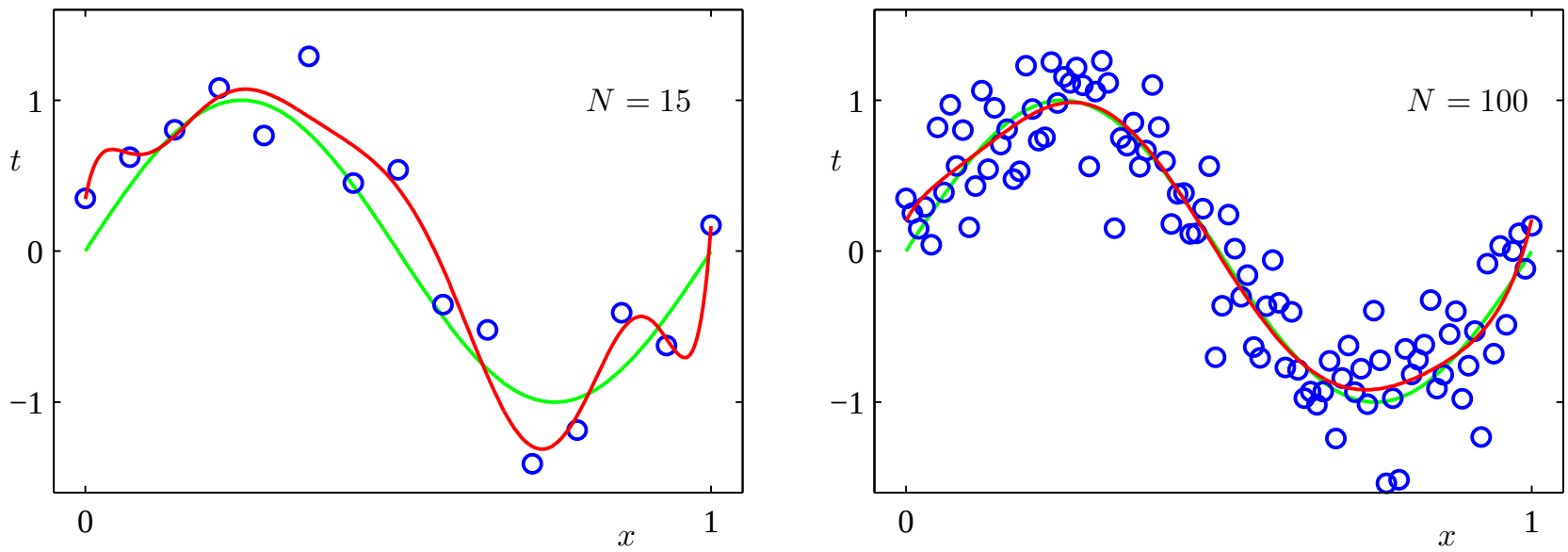
Too simple (small M) $\rightarrow$ poor fit

Too complex (large M) $\rightarrow$ overfitting (fits the noise)

Q: Taylor expansion of $\sin(x)$ contains all odd order terms ... shouldn't $M = 9$ be better?

	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



Overfitting is not due to noise, but more due to sparseness of data.

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,

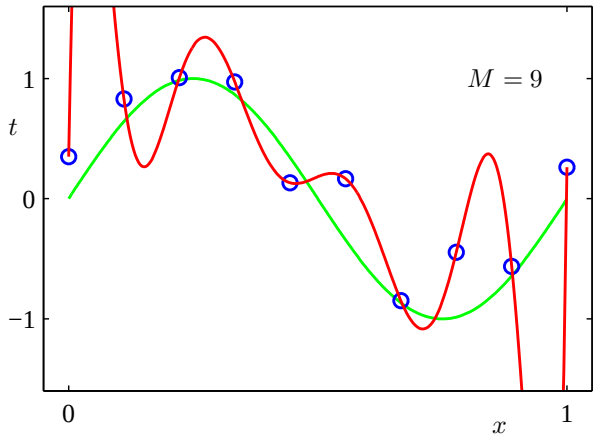
$$\tilde{E}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(y(x_n, \boldsymbol{w}) - t_n\right)^2 + \frac{\lambda}{2}||\boldsymbol{w}||^2$$

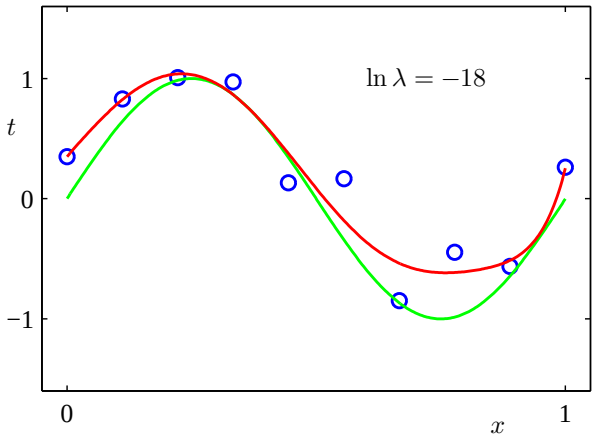
(here, $||\boldsymbol{w}||^2 := \sum_{m=0}^{M} w_m^2$)

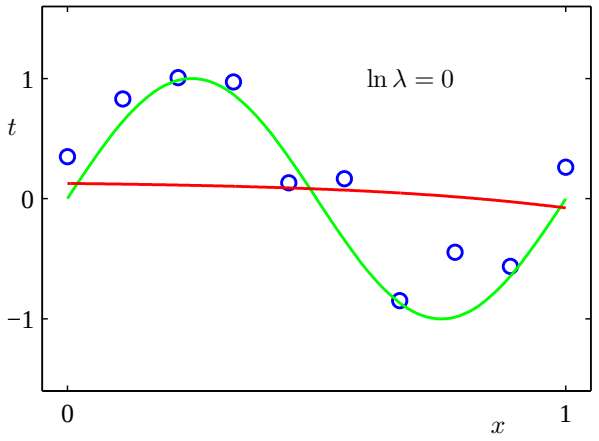
Weight decay = shrinkage = ridge regression

Penalty term independent of number of training data

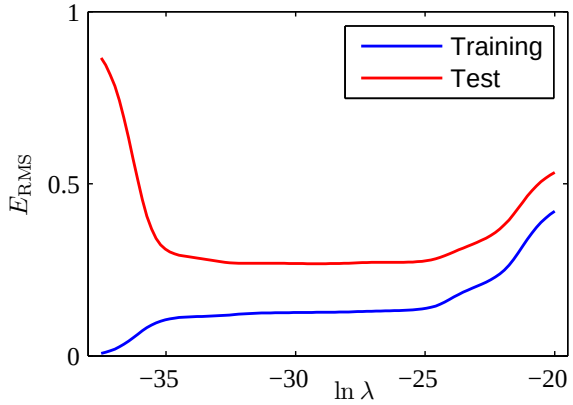
- small data sets: penalty term relatively large
- large data sets: penalty term relatively small
- $\rightarrow$ effective complexity depends on #training data







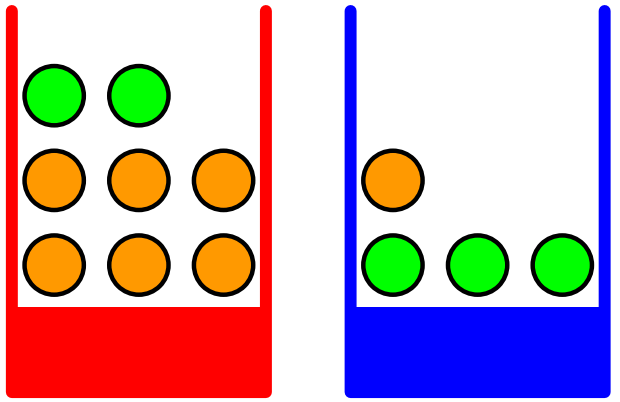
	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
w_0^*	0.35	0.35	0.13
w_1^*	232	4.74	-0.05
w_2^*	5321	- 0.77	-0.06
w_3^*	48568	-31.97	-0.05
w_4^*	-231639	- 3.89	-0.03
w_5^*	640042	55.28	-0.02
w_6^*	-10618000	41.32	-0.01
w_7^*	10424000	-45.95	-0.00
w_8^*	-557683	-91.53	0.00
w_9^*	-125201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



- *Training set* to optimize (typically many) parameters $\boldsymbol{w}$
- *Validation set* to optimize (typically a few) hyperparameters λ, M
- used in e.g. *cross-validation* (§1.3) ... but even better: *Bayesian* approach!

§1.2 Probability theory

- Consistent framework for quantification and manipulation of uncertainty
  $\rightarrow$ Foundation for Bayesian machine learning
- random variable = stochastic variable



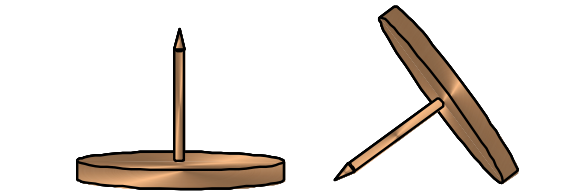
Example:
boxes ($B = \{r, b\}$) and fruit ($F = \{a, o\}$).

- 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.

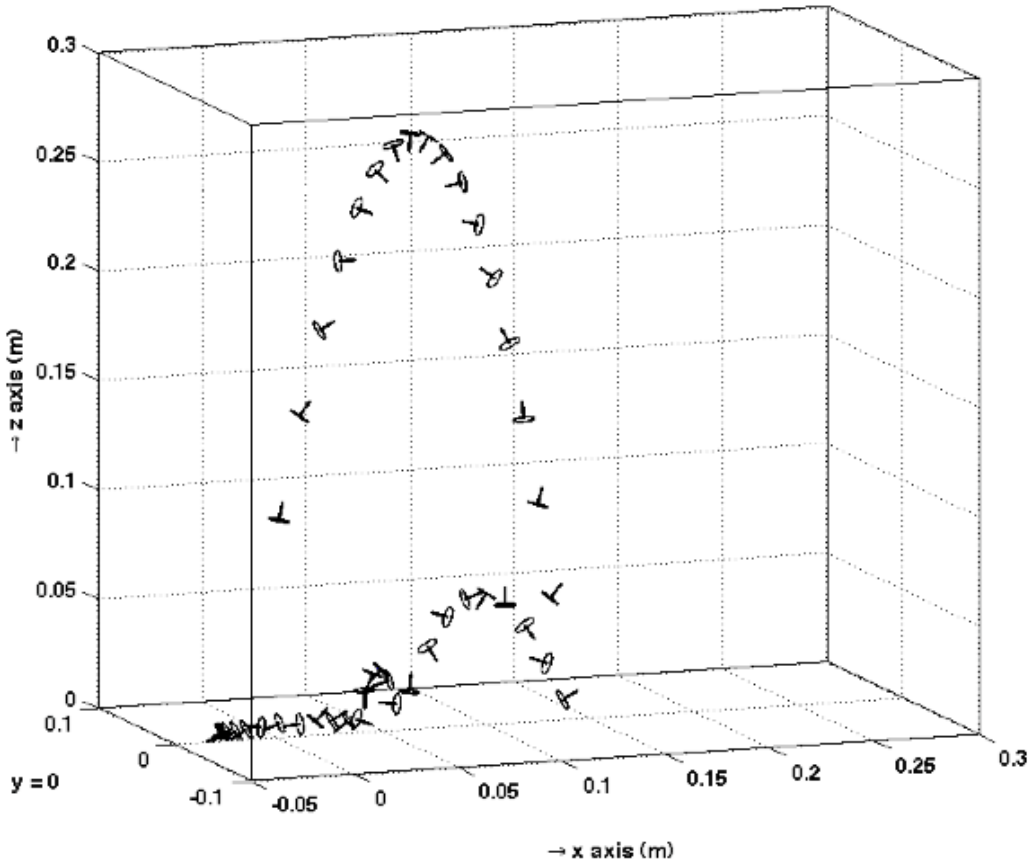
- Notation: $p(F = o) = 9/20$, etc (or $P(\ldots)$, $\mathbb{P}(\ldots)$, Prob$(\ldots)$, etc.)

Experiment with a drawing pin



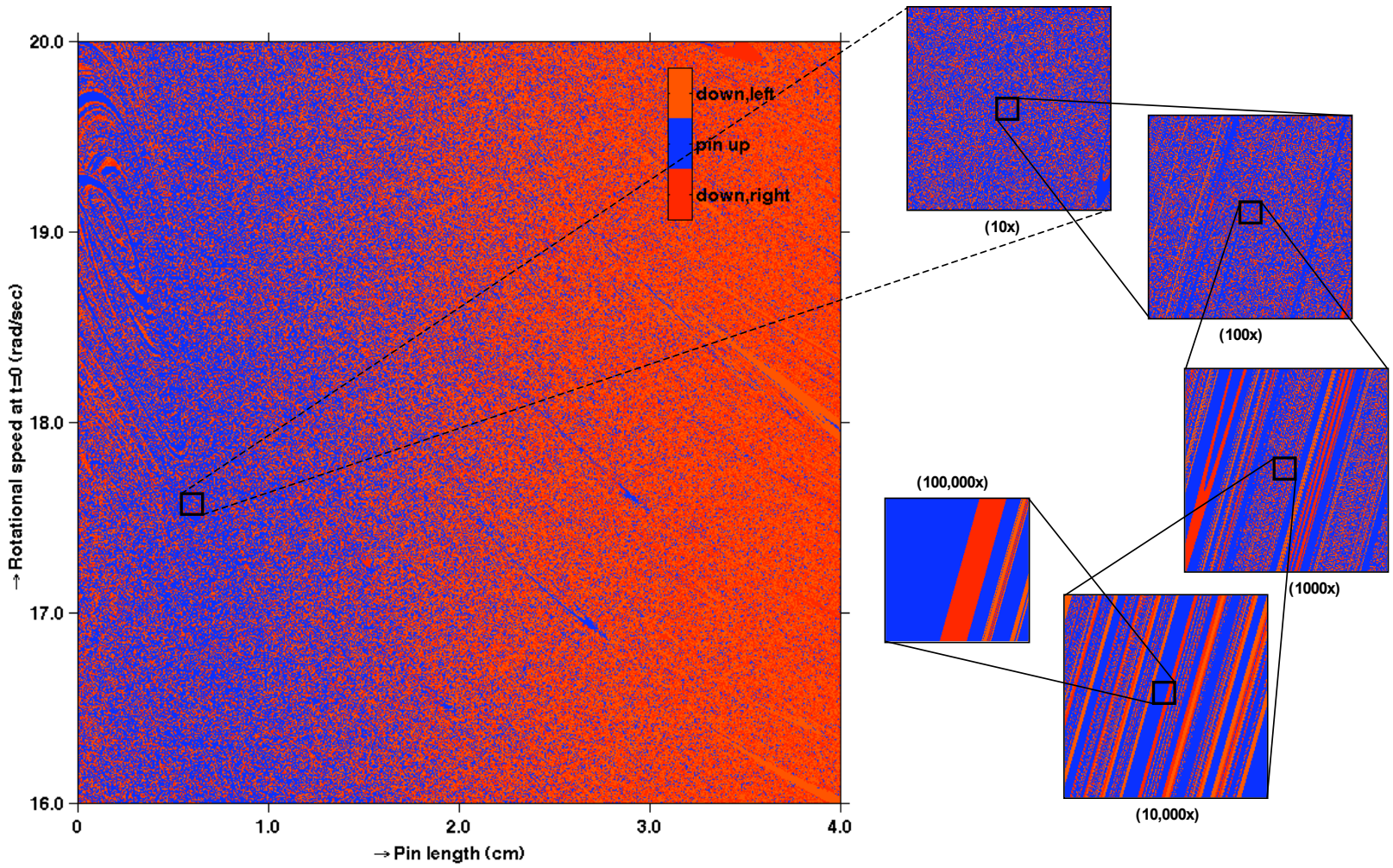
(a)

(b)



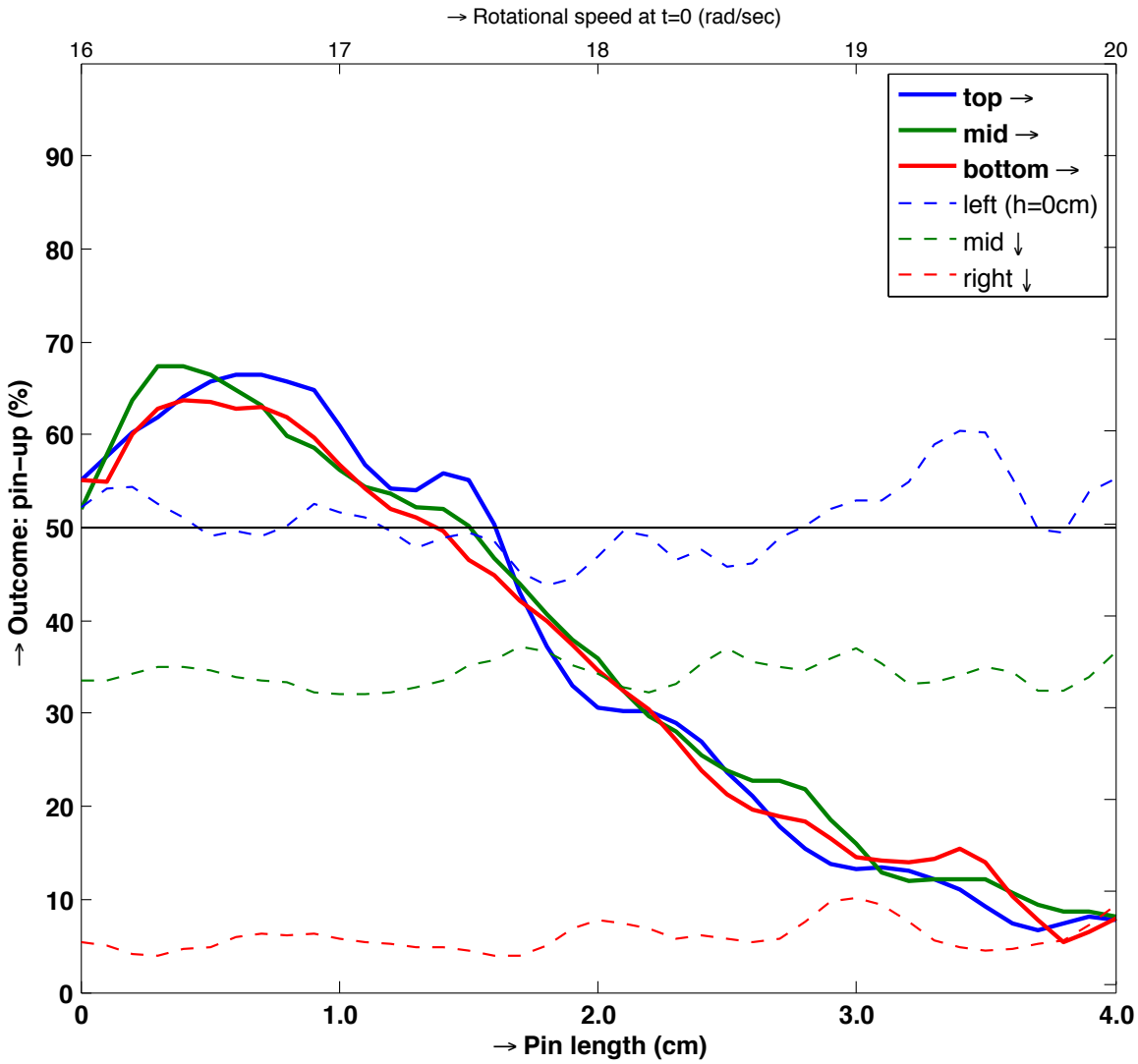
Q: What is the probability of getting 'pin up'?

Outcome as a function of speed vs. pin length



- 'Randomness' in deterministic systems is due to uncertainty about initial states.
- More information does not imply gradual convergence to true value.

Drawing pin outcomes: sliding averages



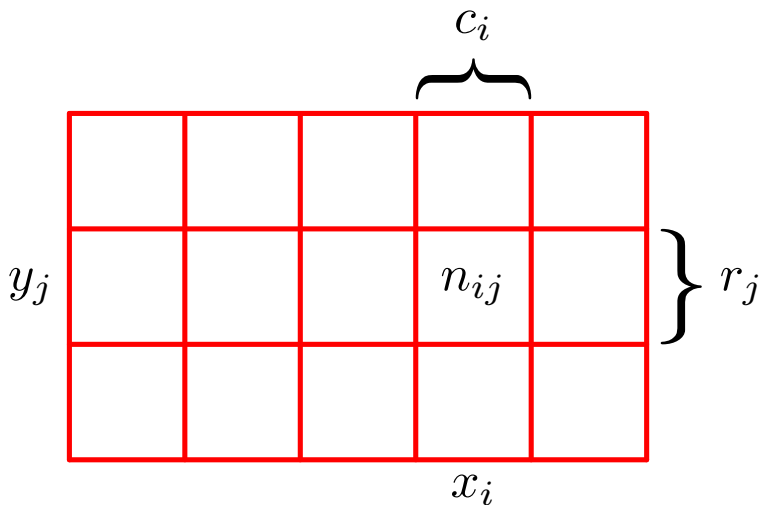
A: For typical drawing pin: $p(Outcome = \text{'pin up'}) \approx 66\%$

... but depends on what you $know$ about the experiment (no unique, objective value!)

Joint, marginal, and conditional probabilities

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 $Y = y_j$



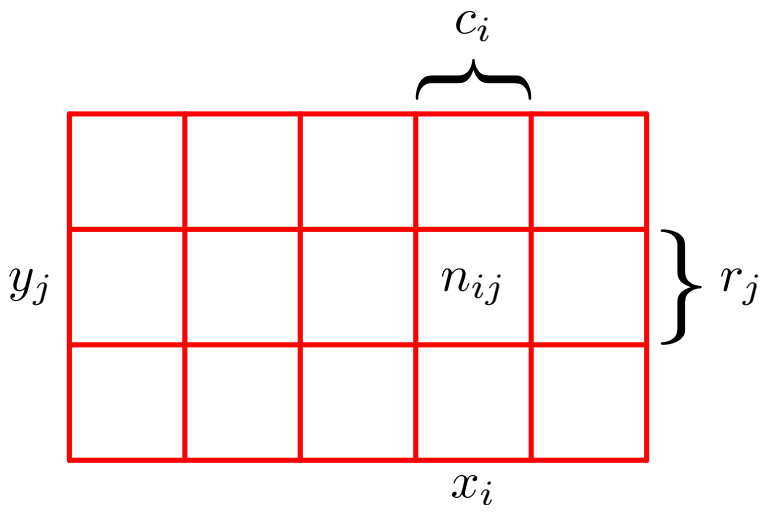
Joint probability of $X = x_i$ *and* $Y = y_j$:

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

Joint, marginal, and conditional probabilities

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 $Y = y_j$

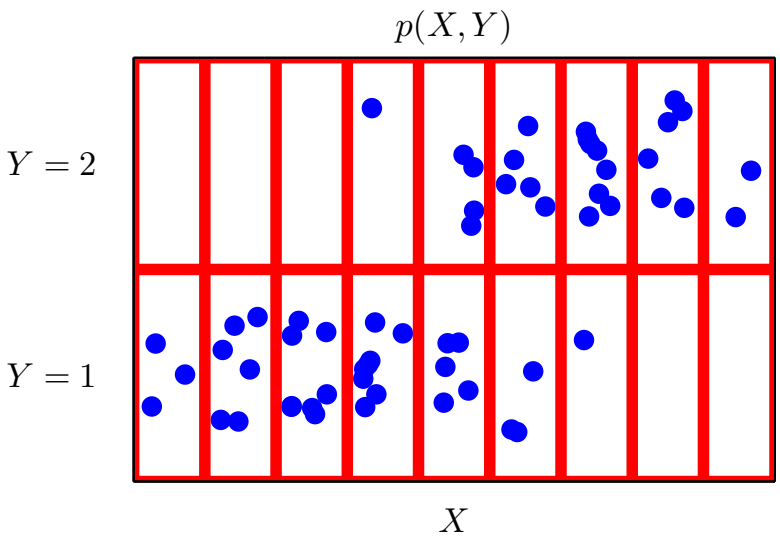


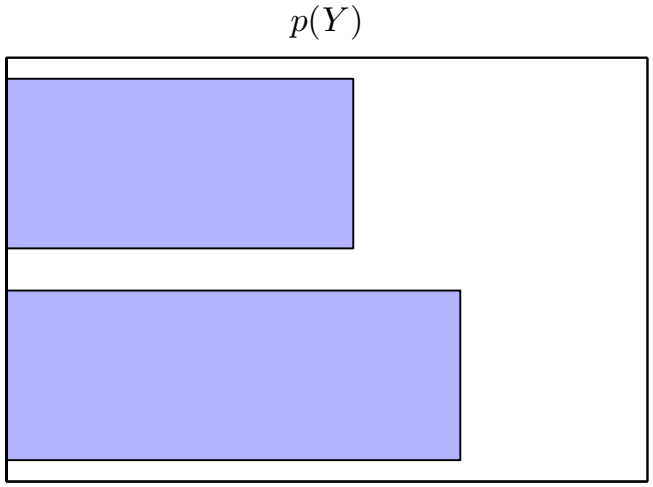
Marginal probability of $X = x_i$:

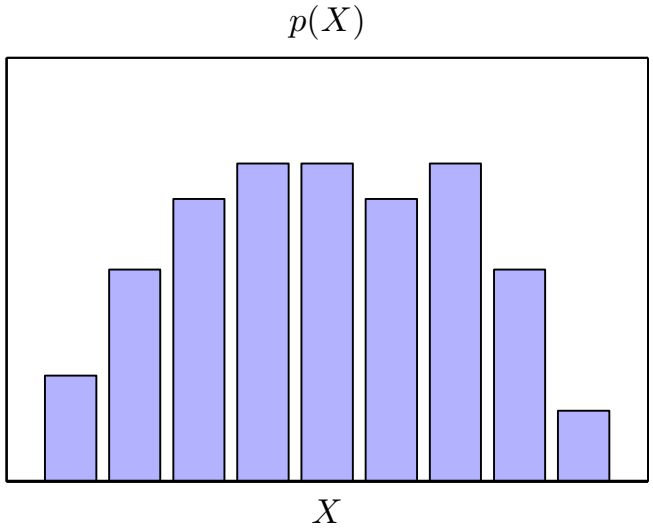
$$p(X = x_i) = \frac{c_i}{N} = \frac{\sum_j n_{ij}}{N} = \sum_j p(X = x_i, Y = y_j)$$

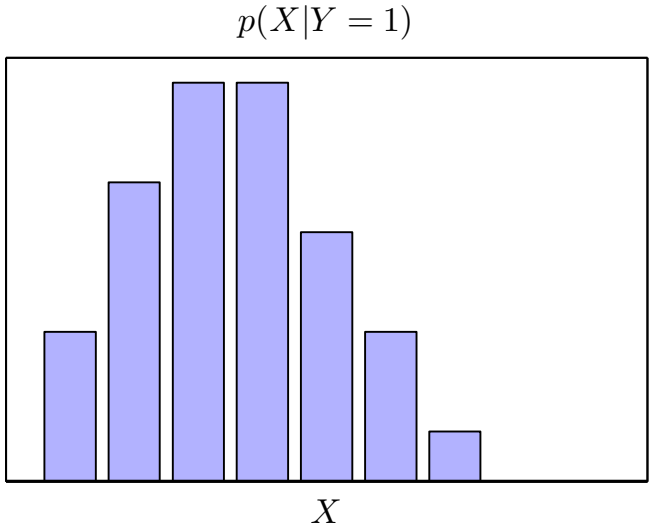
Conditional probability of $Y = y_j$ given $X = x_i$

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


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


$p(Y)$


$p(X)$
X


$p(X|Y=1)$
X

- 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

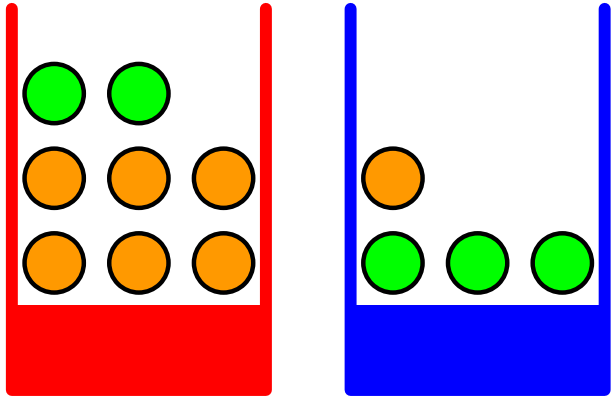
Fruits again

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:

$$p(B = r) + p(B = b) = 1$$
$$p(F = a|B = r) + p(F = o|B = r) = 1$$
$$p(F = a|B = b) + p(F = o|B = b) = 1$$

- 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

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

Probability densities

- to deal with continuous variables (rather than discrete ones)

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

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

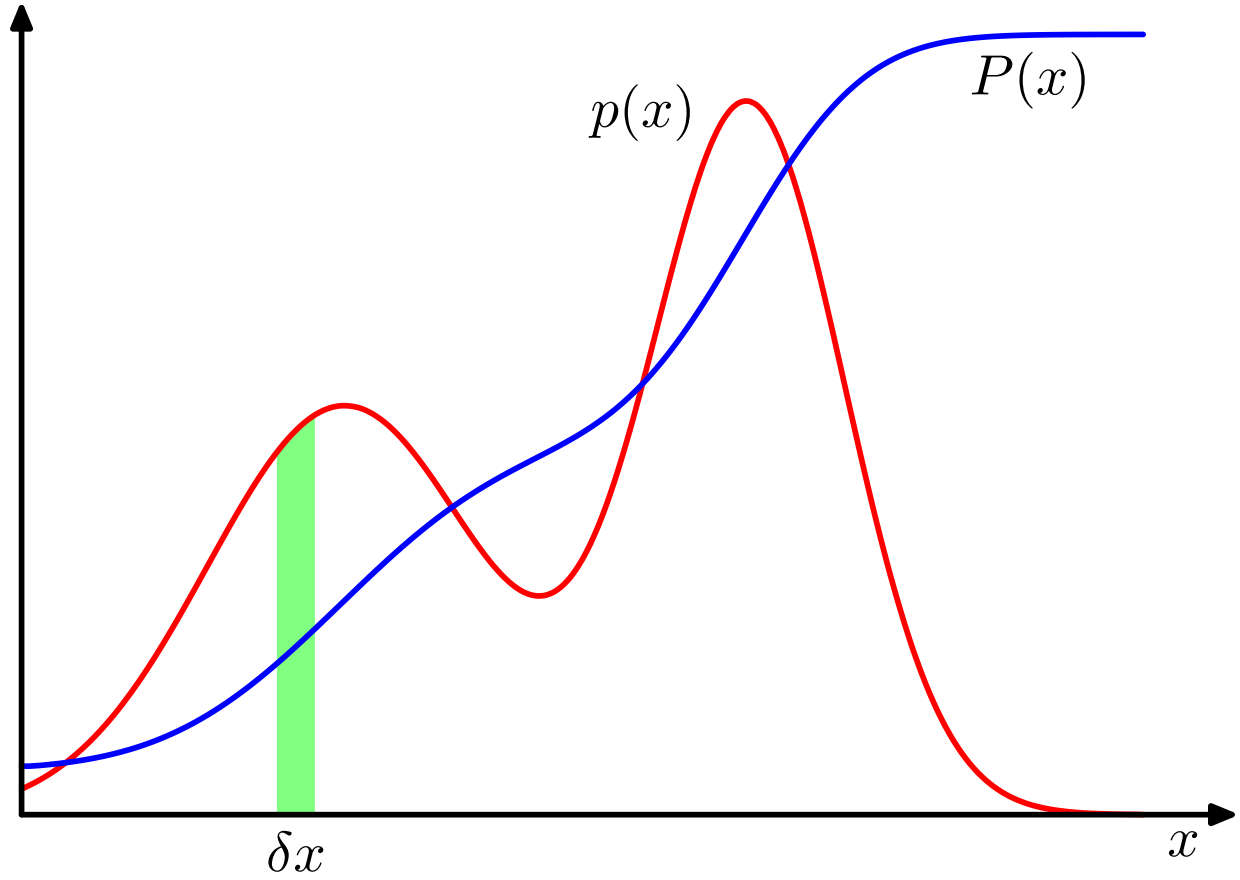
with $p(x)$ the *probability density* over x.

$$p(x) \geq 0$$

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

- 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 $\delta x \to 0$



Cumulative distribution function $F(z) = \int_{-\infty}^{z} p(x)\mathrm{d}x$ (not often used in ML).

Note that:

- Prob$(x \in [a, b]) = F(b) - F(a)$.
- $F'(z) = p(z)$

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 multivariate continuous variables as well,

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

$$p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})$$

Integration

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

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

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 $f(\boldsymbol{x})$.

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.

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 $f(g(x))$ 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 $y = g(x)$ and $x = g^{-1}(y)$ and the end points of the interval $(y = c, y = d)$ are mapped to $(x = a, x = b)$, (so $a = g^{-1}(c)$, etc) then we have the equality

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

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

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

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

$$x = r\cos\theta \tag{2}$$

$$y = r\sin\theta \tag{3}$$

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.

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

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

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

Transformation of densities

Under nonlinear change of variables, a probability transforms $p(\boldsymbol{x})$ transforms differently from an ordinary function, due to "conservation of probability" $p(\boldsymbol{x})\delta\boldsymbol{x}$.

Consider $\boldsymbol{x} = \boldsymbol{g}(\boldsymbol{y})$ then an ordinary function $f(\boldsymbol{x})$ becomes $\tilde{f}(\boldsymbol{y}) = f(\boldsymbol{g}(\boldsymbol{y}))$ by straightforward substitution.

However for probability densities:

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

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

- Values $p(\boldsymbol{x})$ of a probability density depends on choice of variable ($p(\boldsymbol{x})\delta\boldsymbol{x}$ is invariant)
- Maximum of a probability density depends on choice of variable (see exercise 1.4).

Dirac's delta-function

Dirac's delta function $\delta(x)$ is defined such that

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

It can be viewed as the limit $\Delta \to 0$ of the function

$$f(x, \Delta) = \frac{1}{\Delta} \quad \text{if} \quad |x| \leq \frac{\Delta}{2} \quad \text{and} \quad f(x, \Delta) = 0 \quad \text{elsewhere}$$

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

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

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 $f(\boldsymbol{y}) = \sum_i p_i y_i$).

Other (nonlinear) example:

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

Expectations and variances

Expectation is a weighted average of a function $f(x)$ under probability distribution $p(x)$:

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

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

Variance measures variability around expectation value (mean):

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

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

Covariance

Covariance measures how much two random variables vary together ('synchronised variability'):

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

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
  - event may yield only indirect observations
  - Bayesian inference to update belief given observations

Why probability theory?

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

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

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

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 the belief that $\boldsymbol{w}$ has a certain value given the particular data set.

- 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:

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

3

[3] 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

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

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.

Medical example

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?

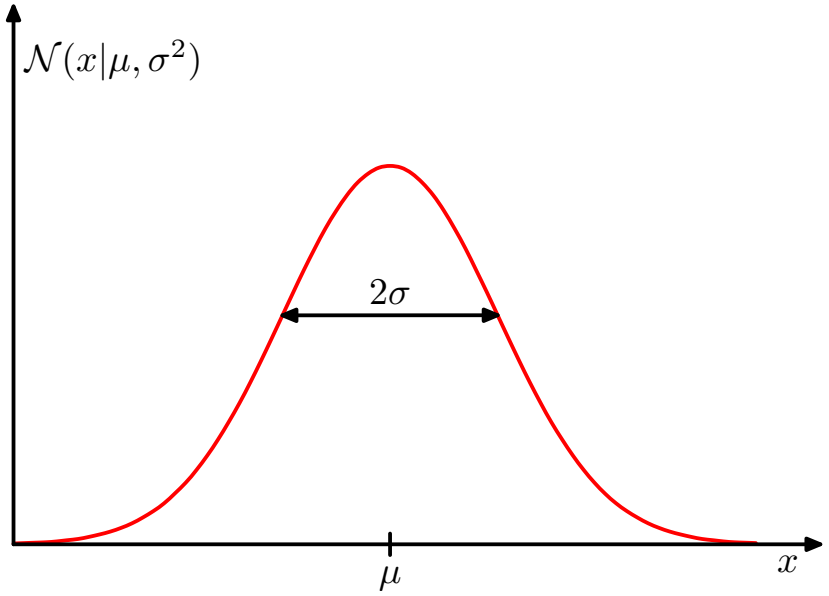
$$p(w=1|x=1) = \frac{0.9*0.01}{0.9*0.01+0.1*0.99} = \frac{1}{1+\frac{0.1*0.99}{0.9*0.01}} = \frac{1}{12} = 0.0825$$

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

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

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

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

Mean vector and covariance matrix

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

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

We can also write this in component notation:

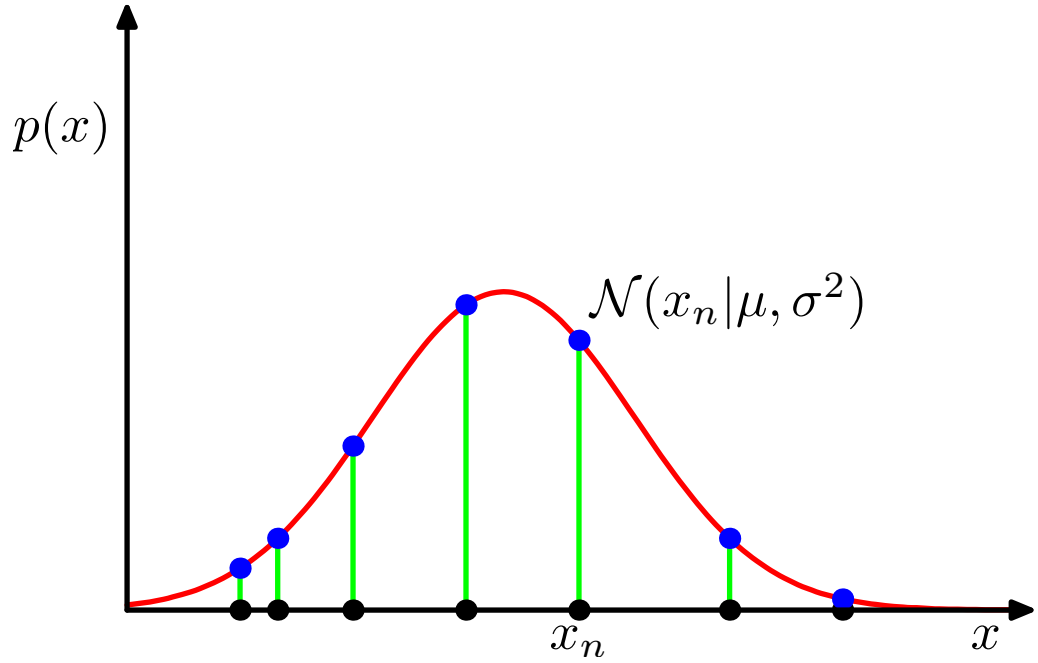
$$\begin{aligned} \mu_i &= \langle x_i \rangle = \int x_i \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \, \mathrm{d}\boldsymbol{x} \\ \Sigma_{ij} &= \langle (x_i - \mu_i)(x_j - \mu_j) \rangle = \int (x_i - \mu_i)(x_j - \mu_j) \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})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

$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is specified by its mean and covariance, in total $D(D+1)/2 + D$ parameters.

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:

$$\begin{aligned}\ln p(\mathbf{x}|\mu,\sigma) &= \ln\left(\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^N \exp\left(-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2\right)\right)\\ &= -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln 2\pi\end{aligned}$$

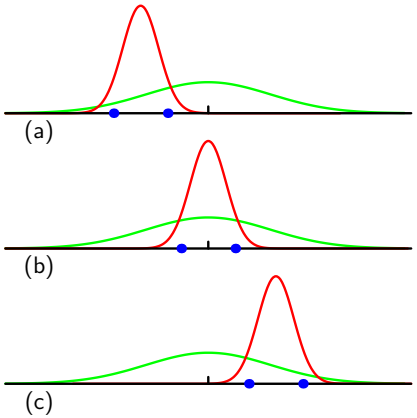
The values of μ and σ that maximize the likelihood are given by

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

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.

(Bayesian approach gives correct expected values)

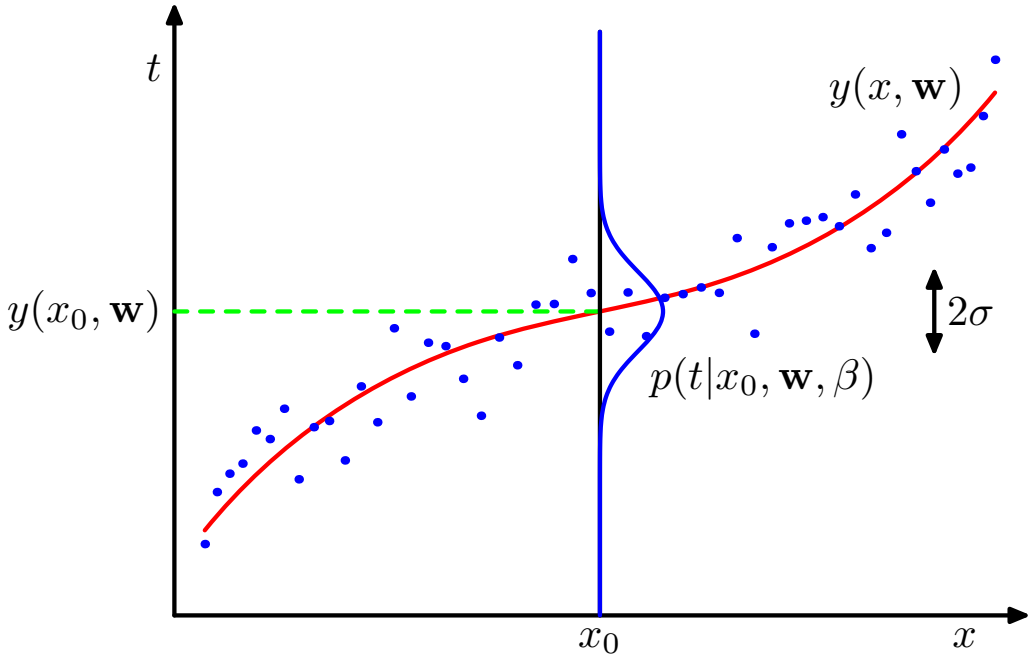
Curve fitting re-visited

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 function,

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

Log-likelihood

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

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

More 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:

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

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

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

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. (How?)

Bayesian curve fitting

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

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

So: assuming Gaussian noise, prediction given $\boldsymbol{w}$ is

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

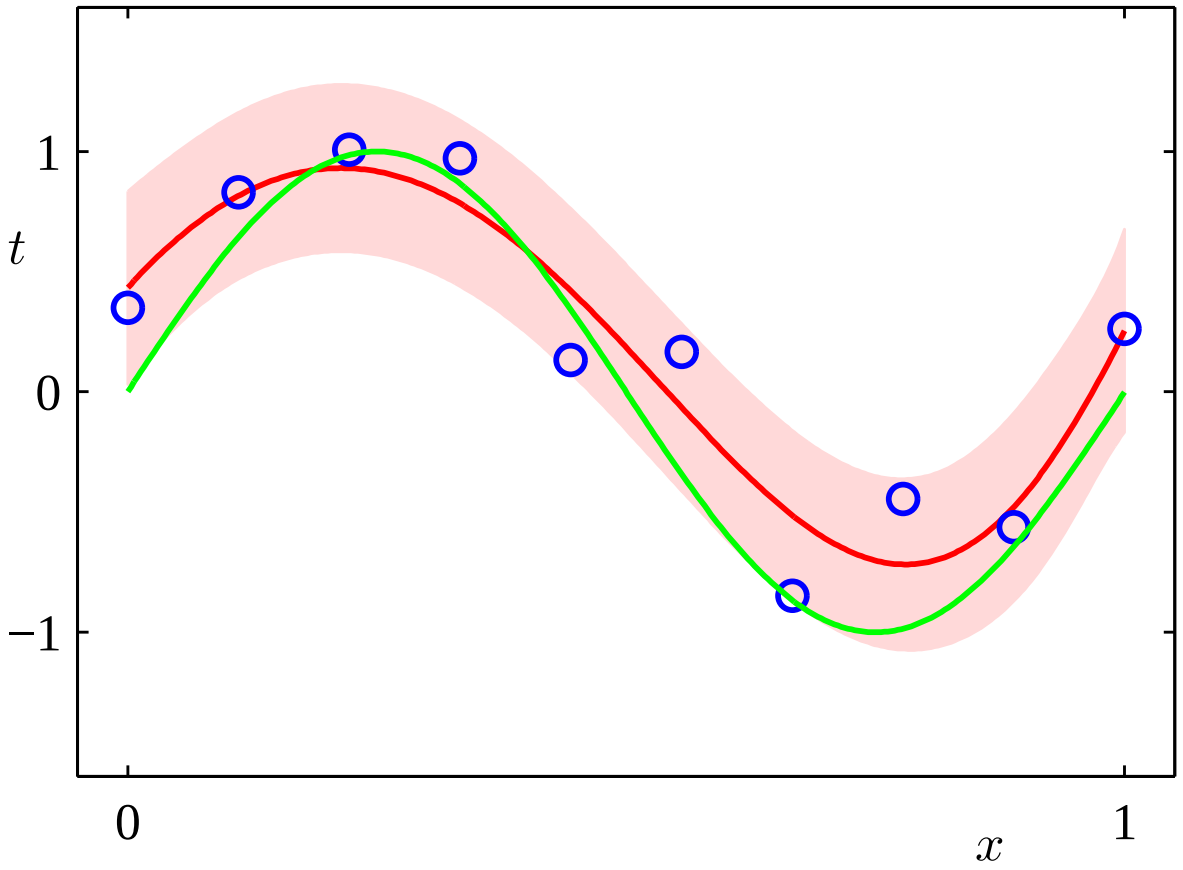
Result Bayesian curve fitting

Predictive distribution

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

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

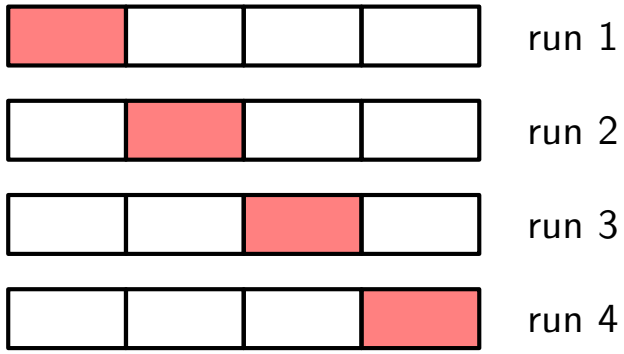
Q: If we have different models to describe the data, which one should we choose?

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*.

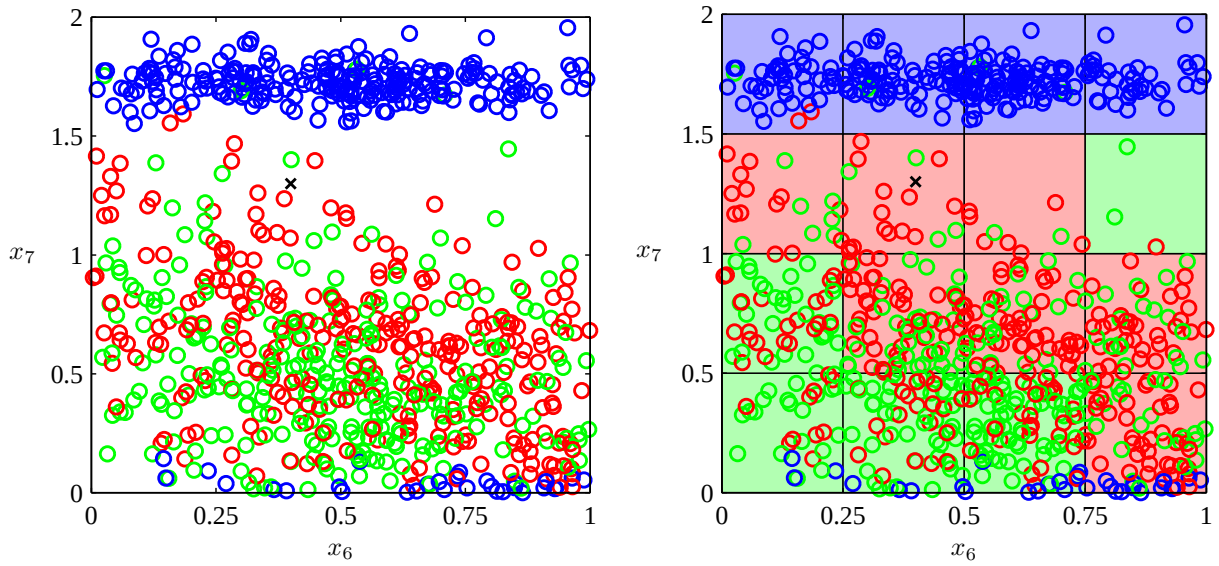


A3: Information criteria: penalty for complex models

- Akaike IC (AIC): $\ln p(D|\boldsymbol{w}_{ML}) - M$
- Bayesian IC (BIC): $\ln p(D|\boldsymbol{w}_{MAP}) - \frac{1}{2}M \ln N$ (Bayesian + crude approximation)
- Full Bayesian $\rightarrow$ penalties arises automatically

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).

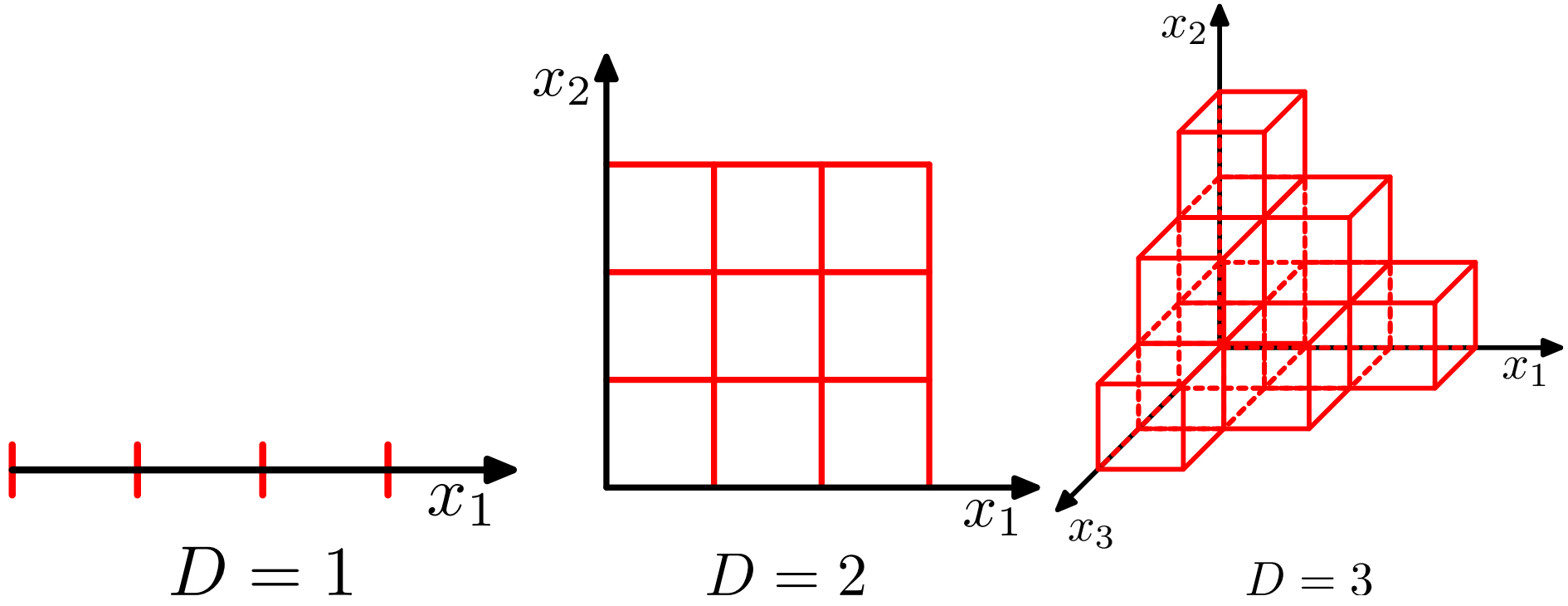
Classification of the 'x' can be done by making a putting a grid on the space and assigning the class that is most numerous in the particular box.

High-dimensional data/Binning

Q: What is the disadvantage of this approach?

Curse of dimensionality/Binning

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



In D dimensions: grid with length n consists of 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 ... ?

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).

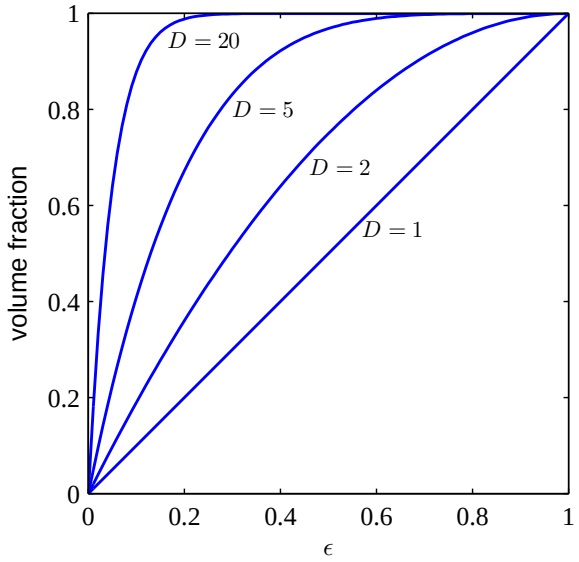
Curse of dimensionality/Spheres

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!

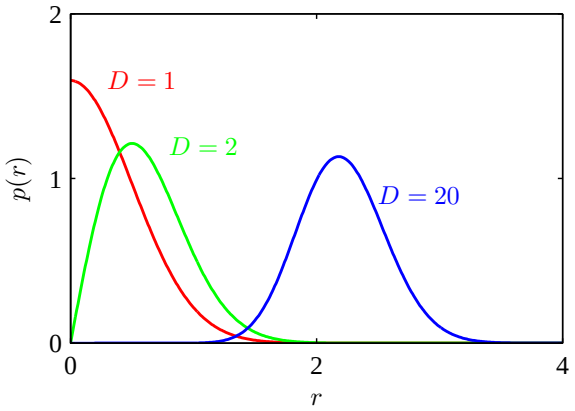
So in high dimensions, almost all data points are at more or less the same distance!



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

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

Curse of dimensionality/Gaussians



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

Intuition developed in low dimensional space may be wrong in high dimensions!

Curse of dimensionality

Is machine learning even possible in high dimensions?

- Data often in low dimensional subspace: only a few dimensions are relevant.
  - Object located in 3-D $\rightarrow$ images of objects are N-D $\rightarrow$ there should be 3-D manifold (curved subspace)
- Smoothness, local interpolation (note: this is also needed in low dimensions).

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:

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

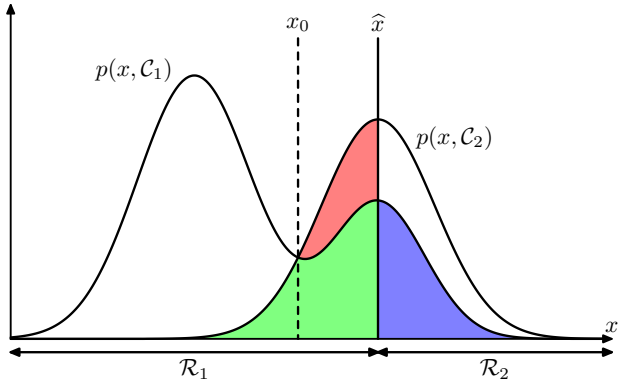
$p(C_k)$ is the prior probability of class C_k. $p(C_k|\boldsymbol{x})$ is the posterior probability of class C_k after seeing the image $\boldsymbol{x}$.

Q: So, how to use this model to decide on the best action?

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.

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

Loss function. Rows are true classes (cancer, normal), columns 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

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

$\langle L \rangle$ is minimized if each $\boldsymbol{x}$ is assigned to class j 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

Loss function. Rows are true classes (cancer, normal), columns 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

Loss function. Rows are true classes (cancer, normal), columns are assigned classes (cancer, normal).

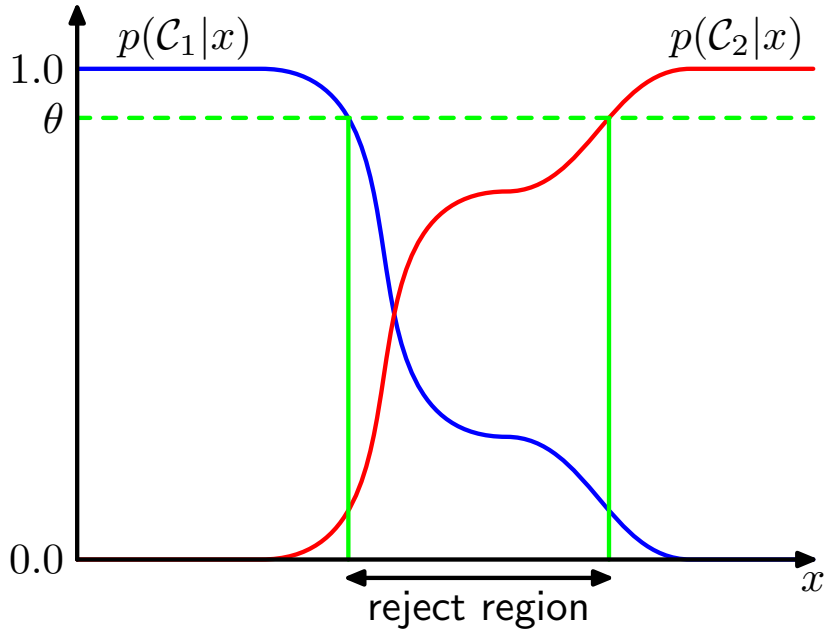
Q: Suppose $p(C_1) = 0.01$ and $p(C_1|\boldsymbol{x}) = 0.3$ according to our model. Do we decide that the patient has cancer and therefore start treatment?

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 $0 \times 0.3 + 1 \times 0.7 = 0.7$,
for $j = 2$, this yields $1000 \times 0.3 + 0 \times 0.7 = 300$.
Therefore, we now decide $j = 1$ 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.

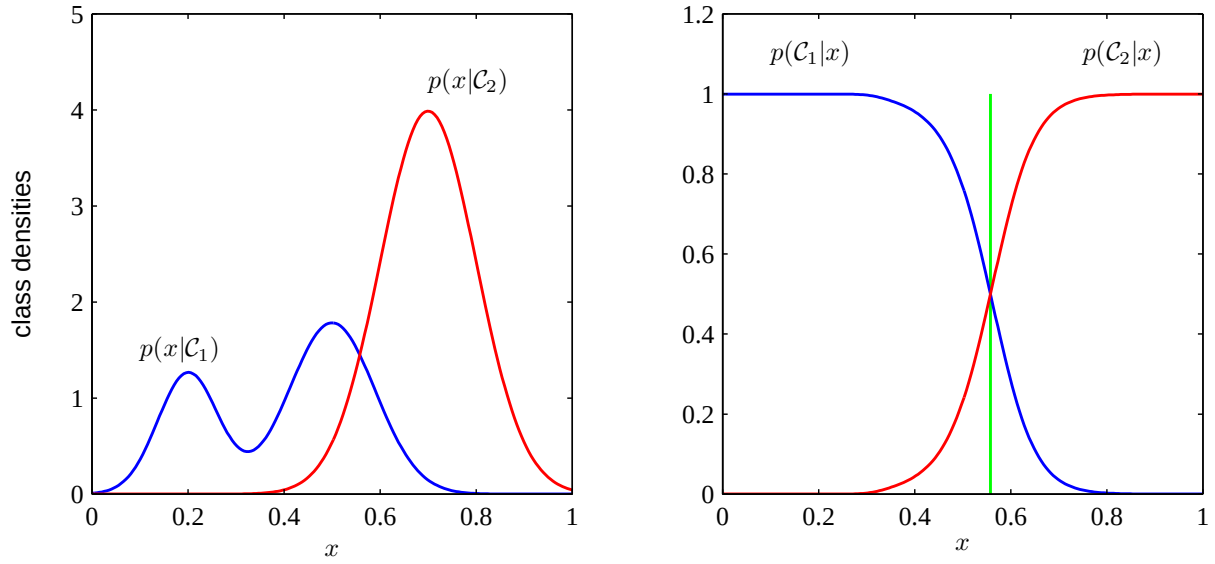
A set of approaches:

- 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, but has the advantage of yielding a model of $p(\boldsymbol{x})$ that can be used to reject unlikely inputs $\boldsymbol{x}$.

- 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).

- 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.

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*.

Decision theory/Discriminant functions

Advantages of learning a class conditional probability instead of discriminant function:

Minimizing risk When minimizing expected loss, the loss matrix may change over time whereas the class probabilities may not (for instance, in a financial application).

Reject option One can reject uncertain class assignments

Unbalanced data One can compensate for unbalanced data sets. For instance, in the cancer example, there may be 1000 times more healthy patients than cancer patients. Very good classification (99.9 % correct) is obtained by classifying everyone as healthy. Using the posterior probability one can compute $p(C_k = \mathrm{cancer}|\boldsymbol{x})$. Although this probability may be low, it may be significantly higher than $p(C_k = \mathrm{cancer})$, indicating a risk of cancer.

Decision theory/Discriminant functions

Combining models Given models for $p(C_k|\boldsymbol{x})$ and $p(C_k|\boldsymbol{y})$ one has a principled approach to classify on both $\boldsymbol{x}$ and $\boldsymbol{y}$. Naive Bayes assumption:

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

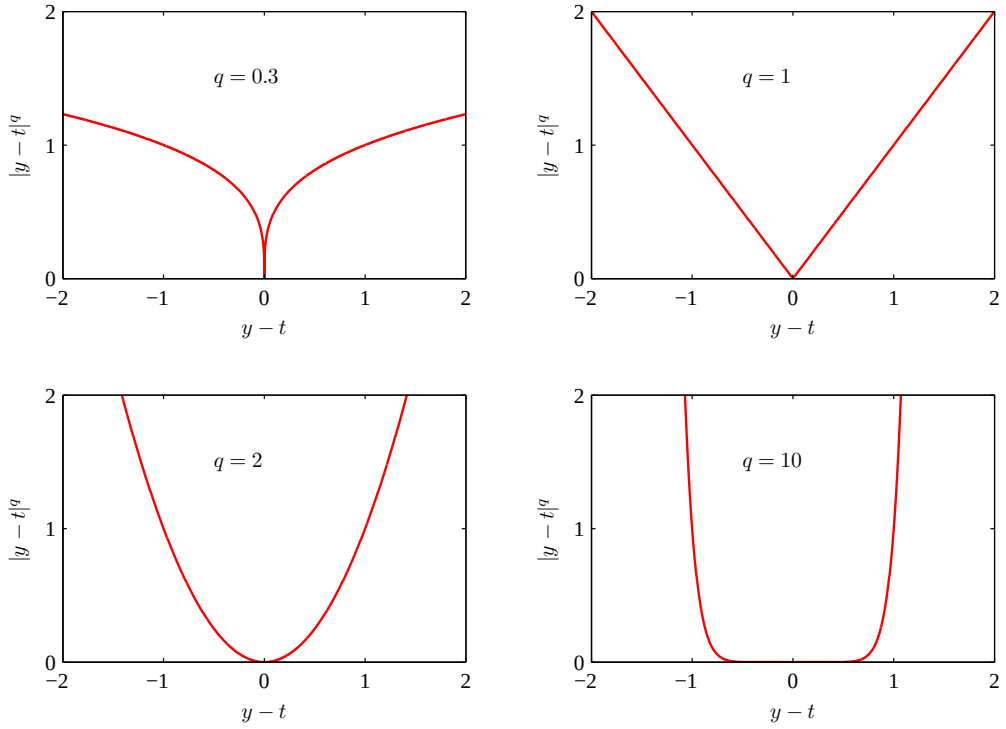
(given the disease state of the patient the blood and X-ray test results are independent). Then

$$p(C_k|\boldsymbol{x}, \boldsymbol{y}) \propto p(\boldsymbol{x}, \boldsymbol{y}|C_k)p(C_k) \propto p(\boldsymbol{x}|C_k)p(\boldsymbol{y}|C_k)p(C_k) \propto \frac{p(C_k|\boldsymbol{x})p(C_k|\boldsymbol{y})}{p(C_k)}$$

Loss functions for regression

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

Examples:



Minkowski loss function $L_q = |y - t|^q$ for various values of q.

Loss functions for regression/Quadratic loss

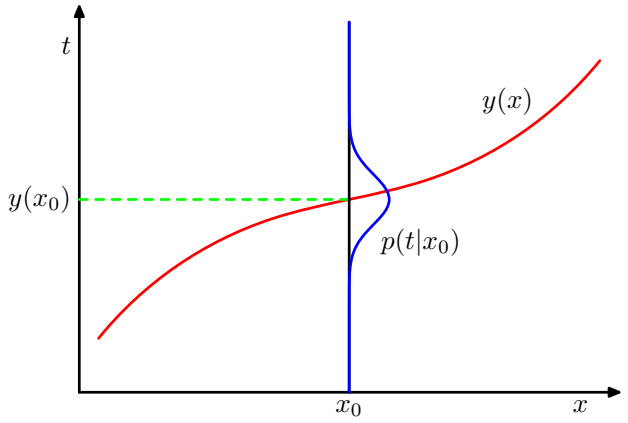
The average/expected loss is:

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

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

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

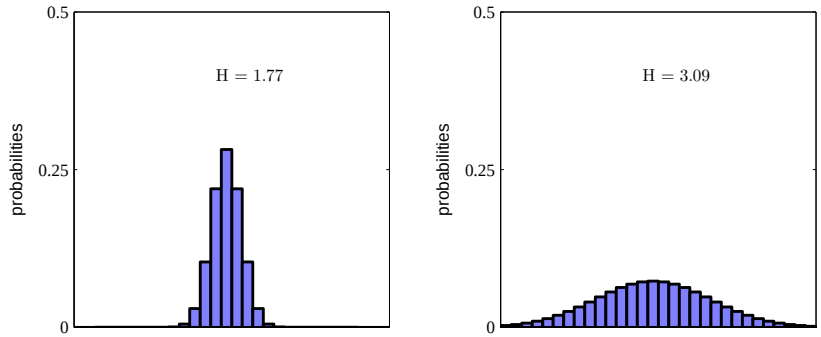
which is smaller than for the uniform distribution.

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).

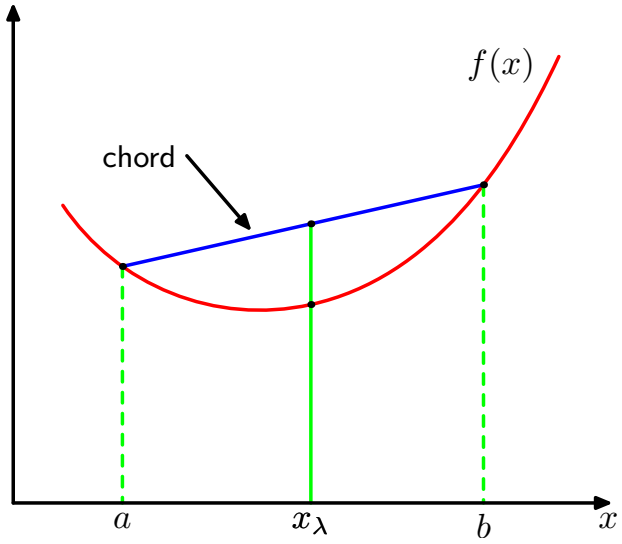
Information theory/KL-divergence

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 $\Leftrightarrow$ 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

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

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:

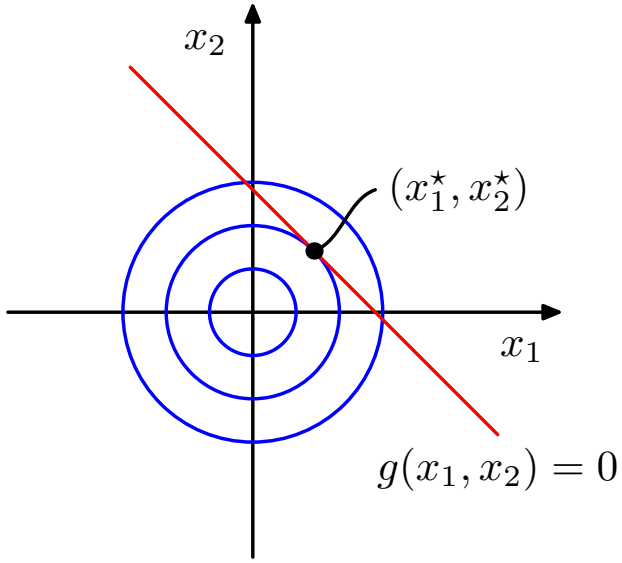
1. Minimize $L(\boldsymbol{x}, \lambda)$ w.r.t $\boldsymbol{x}$, e.g. by taking the gradient and set to zero. This yields a (parametrized) solution $\boldsymbol{x}(\lambda)$.

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



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

Lagrangian:

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

Minimize L w.r.t. x_i gives $x_i(\lambda) = \frac{1}{2}\lambda$.

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

So $\lambda = 1$ and $x_i^* = \frac{1}{2}$

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).

Chapter 2

Probability distributions

- Density estimation
- Parametric distributions
- Maximum likelihood, Bayesian inference, conjugate priors
- Bernoulli (binary), Beta, Gaussian, ..., exponential family
- Nonparametric distribution

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:

$$p(D|\mu) = \prod_n p(x_n|\mu) = \prod_n \mu^{x_n}(1-\mu)^{1-x_n}$$

Log likelihood

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

where $m = \sum_n x_n$, the total number of $x_n = 1$.

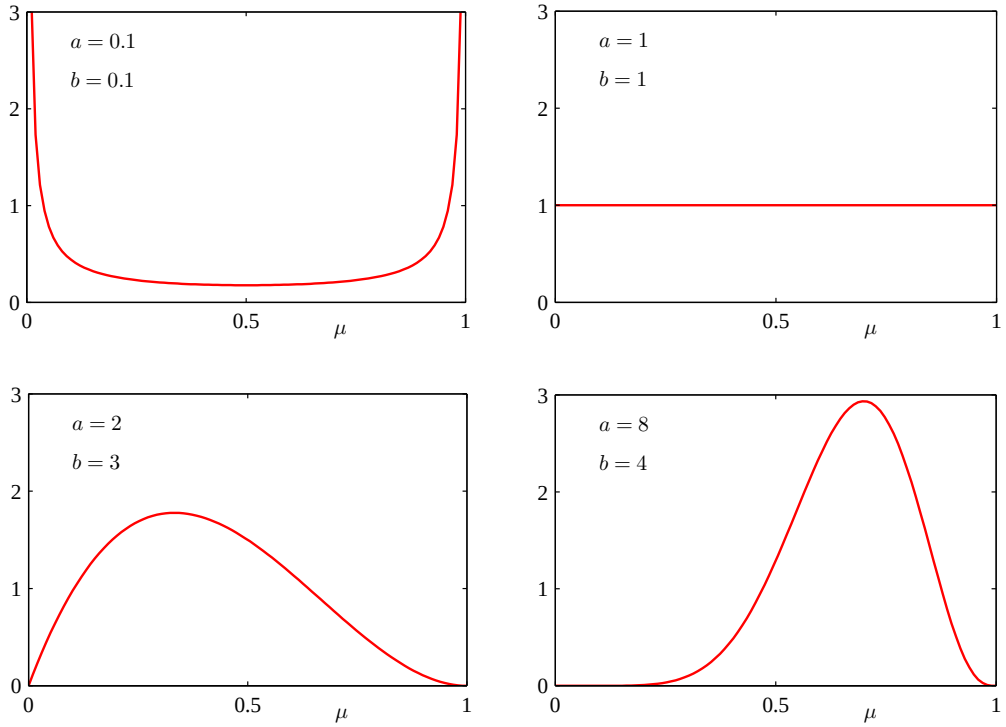
Maximization w.r.t μ gives maximum likelihood solution:

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

The beta distribution

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

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



The beta distribution

$$\begin{aligned} \text{Beta}(\mu|a,b) &= \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1-\mu)^{b-1} \\ &\propto \mu^{a-1}(1-\mu)^{b-1} \quad 0 \leq \mu \leq 1 \end{aligned}$$

Normalisation

$$\int_0^1 \text{Beta}(\mu|a,b) = 1$$

Mean and variance

$$\begin{aligned} \mathbb{E}[\mu] &= \frac{a}{a+b} \\ \text{var}[\mu] &= \frac{ab}{(a+b)^2(a+b+1)} \end{aligned}$$

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

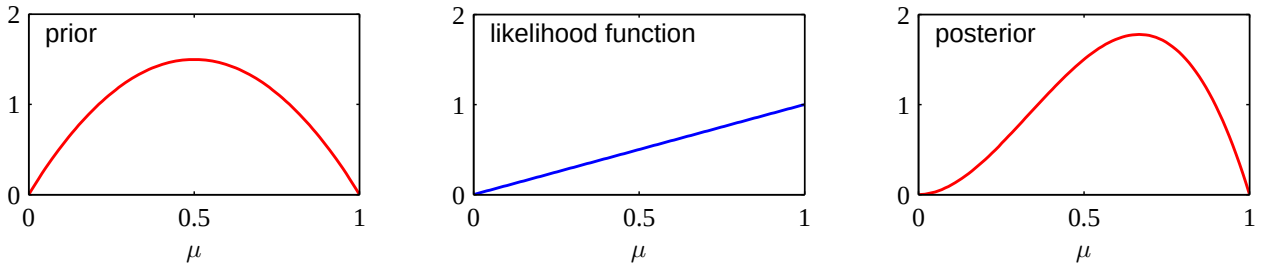
$$\begin{aligned} p(\mu|D) &\propto p(D|\mu)p(\mu) \\ &= \mu^m(1-\mu)^l \times \mu^{a-1}(1-\mu)^{b-1} \\ &= \mu^{m+a-1}(1-\mu)^{l+b-1} \propto \text{Beta}(\mu|a+m, b+l) \end{aligned}$$

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)



Posterior interpreted as updated prior: sequential learning

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 is $\mathrm{Beta}(\mu|a+m, b+l)$, and $\mathbb{E}[\mu|a,b] = a/(a+b)$ we find

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

Multinomial variables

Alternative representation for Bernoulli distribution: $x \in \{v_1, v_2\}$, parameter vector: $\boldsymbol{\mu} = (\mu_1, \mu_2)$, with $\mu_1 + \mu_2 = 1$.

$$p(x = v_k|\boldsymbol{\mu}) = \mu_k$$

In fancy notation:

$$p(x|\boldsymbol{\mu}) = \prod_k \boldsymbol{\mu}_k^{\delta_{xv_k}}$$

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

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

Multinomial variables: Maximum likelihood

Likelihood:

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

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

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

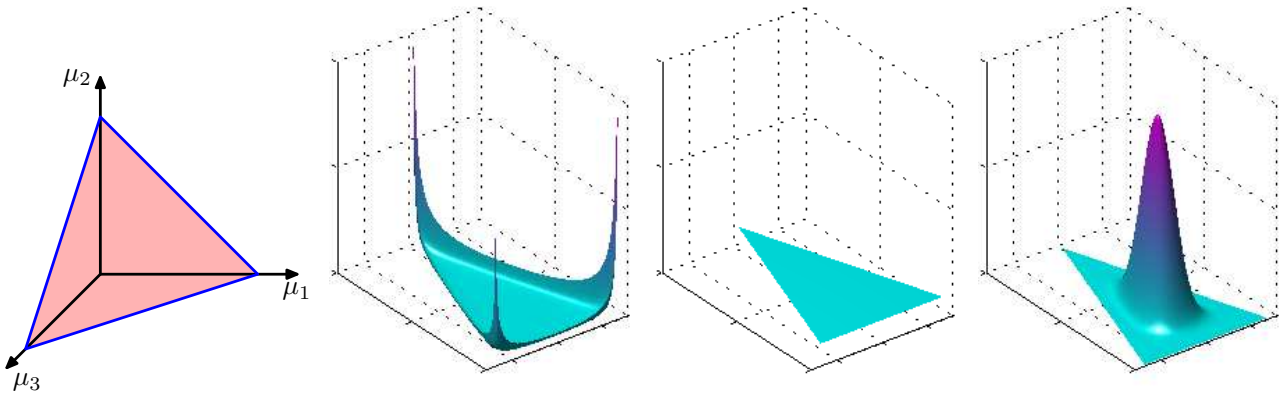
Maximize with constraints using Lagrange multipliers.

Dirichlet distribution

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

Probability distribution on the *simplex*:

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



$$\alpha_k = 0.1, 1, 10$$

Dirichlet distribution

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

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 $\mathrm{Dir}(\boldsymbol{\mu}|\boldsymbol{\alpha})$ + data counts $\boldsymbol{m}$

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

Parameters $\boldsymbol{\alpha}$: 'pseudocounts'.

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

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

Central limit theorem

Sum of large set of random variables is approximately Gaussian distributed.

Let X_i set of random variables (not Gaussian!) with mean μ and variance σ^2.

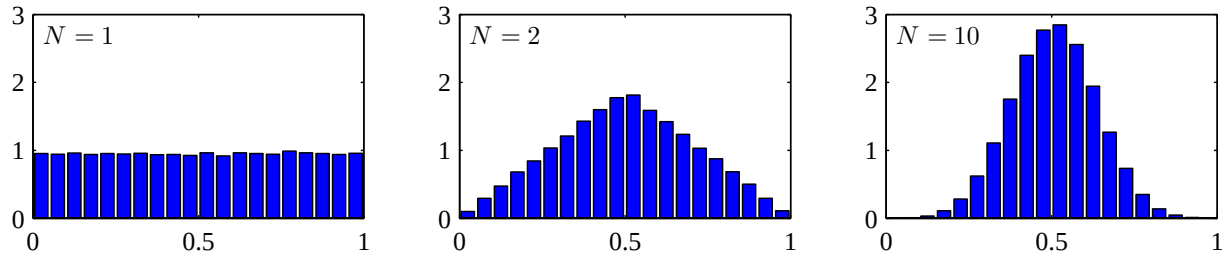
The sum of first n variables is $S_n = X_1 + ... + X_n$. Now if $n \to \infty$

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

Central limit theorem: Distribution of

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

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



Understanding the Gaussian through the covariance matrix Σ

Dependence on $\boldsymbol{x}$ only through the quadratic form

$$\boldsymbol{\Delta}^2 = (\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})$$

Properties of symmetric matrices

$$A_{ij} = A_{ji}, \quad \boldsymbol{A}^T = \boldsymbol{A}$$

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

The inverse of a symmetric matrix $\boldsymbol{A}^{-1}$ is also symmetric:

$$\boldsymbol{I} = \boldsymbol{I}^T = (\boldsymbol{A}^{-1}\boldsymbol{A})^T = \boldsymbol{A}^T(\boldsymbol{A}^{-1})^T = \boldsymbol{A}(\boldsymbol{A}^{-1})^T$$

Thus, $(\boldsymbol{A}^{-1})^T = \boldsymbol{A}^{-1}$. (So precision matrix and covariance matrix are both symmetric.)

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 [4] $\lambda_1, \ldots, \lambda_d$.

[4] 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:

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

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

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

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:

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

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

If $\boldsymbol{u}_k$ is an eigenvector of $\boldsymbol{A}$, then $\alpha \boldsymbol{u}_k$ is also an eigenvector of $\boldsymbol{A}$. 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 [5] , i.e.

$$U^T U = I$$

For orthogonal matrices,

$$U^T = U^T U U^{-1} = U^{-1}$$

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

Furthermore,

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

5

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

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 orthonormal basis U as

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

[6] 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.

6

$$x_i = (UU^T\boldsymbol{x})_i = \sum_{kj} U_{ik}(U^T)_{kj}x_j = \sum_{kj}(\boldsymbol{u}_k)_i(\boldsymbol{u}_k)_j x_j = \sum_{k}(\boldsymbol{u}_k)_i(\boldsymbol{u}_k^T \cdot \boldsymbol{x}) = \left(\sum_k \boldsymbol{u}_k(\boldsymbol{u}_k^T \cdot \boldsymbol{x})\right)_i$$

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

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

As a result, a matrix is diagonal on a basis of its eigenvectors:

$$A' = U^T AU = U^T U\Lambda U^T U = \Lambda$$

Multivariate Gaussian

In d dimensions

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

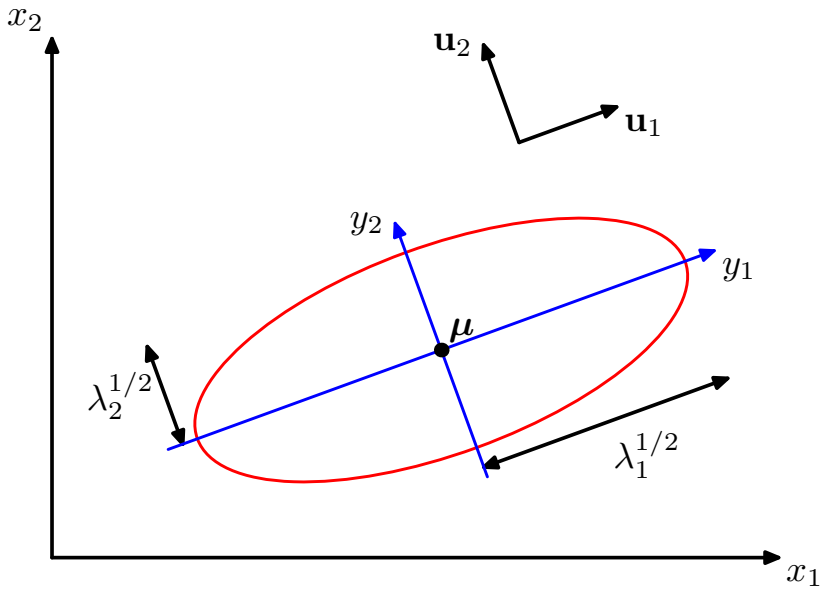
We can also write this in component notation:

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

Multivariate Gaussian

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

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



Multivariate Gaussian

Explain the normalization: use transformation[7] $\boldsymbol{y} = U^T(\boldsymbol{x} - \boldsymbol{\mu}) = U^T\boldsymbol{z}$

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

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

[7] Bishop alternates between defining matrix U as *rows* of eigenvectors $\boldsymbol{u}_i^T$ (2.52), and U as *columns* of eigenvectors $\boldsymbol{u}_i$ (C.37). In the slides we always use the more conventional *column* representation.

Multivariate Gaussian

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

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

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

Multivariate Gaussian

Second order moment: use transformation $\boldsymbol{y} = U^T(\boldsymbol{x} - \boldsymbol{\mu}) = U^T\boldsymbol{z}$

First, shift by $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{\mu}$

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

Multivariate Gaussian

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

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

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

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

Multivariate Gaussian

So, second moment is

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

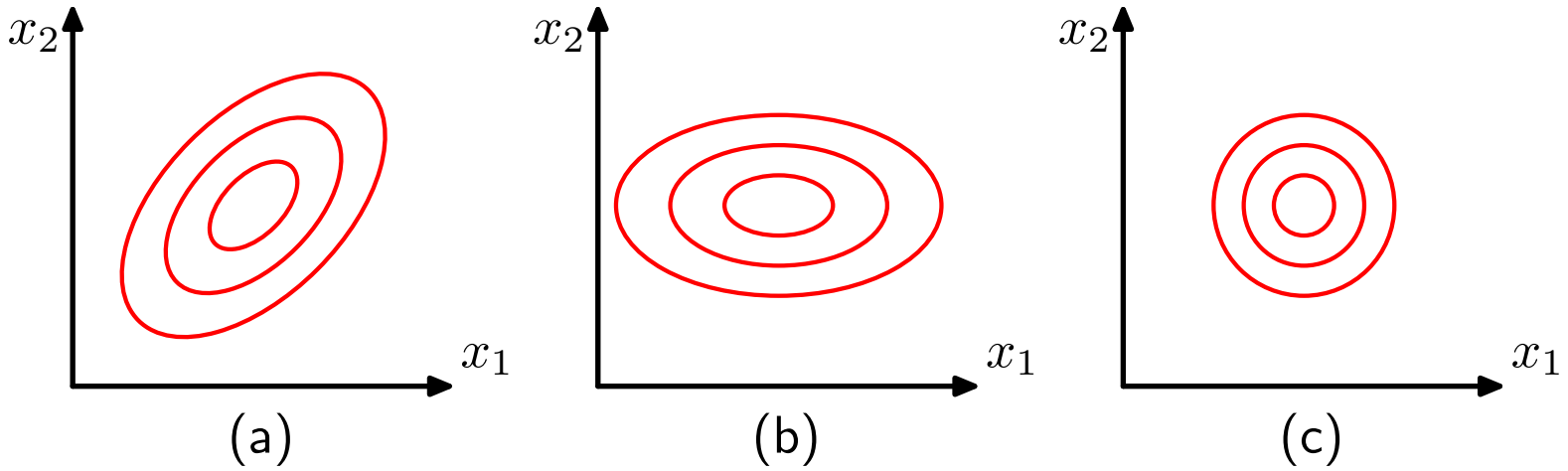
Covariance

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

Multivariate Gaussian

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

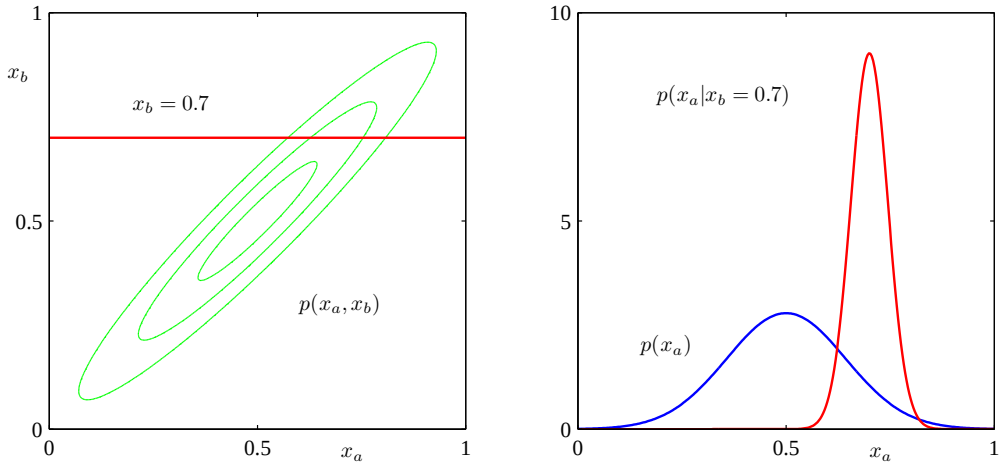
Number of parameters quadratic in d which may be too large for high dimensional applications.



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.

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

2.

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

Conditional of Gaussian is Gaussian

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

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

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

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

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

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

Some matrix identities

We now need to relate K_{aa}^{-1} 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

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

Bayes' theorem for linear Gaussian model

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

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

Then (see next slide):

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

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

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

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

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

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

$$\begin{aligned} \mathbb{E}[\boldsymbol{y}] &= \mathbb{E}[\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} + \boldsymbol{\epsilon}] = \boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{b} \\ \mathrm{cov}[\boldsymbol{y}] &= \mathrm{cov}[\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} + \boldsymbol{\epsilon}] = \mathrm{cov}[\boldsymbol{A}\boldsymbol{x}] + \mathrm{cov}[\boldsymbol{\epsilon}] = \mathrm{cov}[\boldsymbol{A}\boldsymbol{x}] + \boldsymbol{L}^{-1} \\ \mathrm{cov}[\boldsymbol{A}\boldsymbol{x}] &= \mathbb{E}[(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}\boldsymbol{\mu})\,(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}\boldsymbol{\mu})^T] = \boldsymbol{A}\mathbb{E}[(\boldsymbol{x} - \boldsymbol{\mu})\,(\boldsymbol{x} - \boldsymbol{\mu})^T]\boldsymbol{A}^T = \boldsymbol{A}\boldsymbol{\Lambda}^{-1}\boldsymbol{A}^T \end{aligned}$$

Thus,

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

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

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

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

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

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

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

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

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:

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

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

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

Bayesian inference for the Gaussian

Likelihood:

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

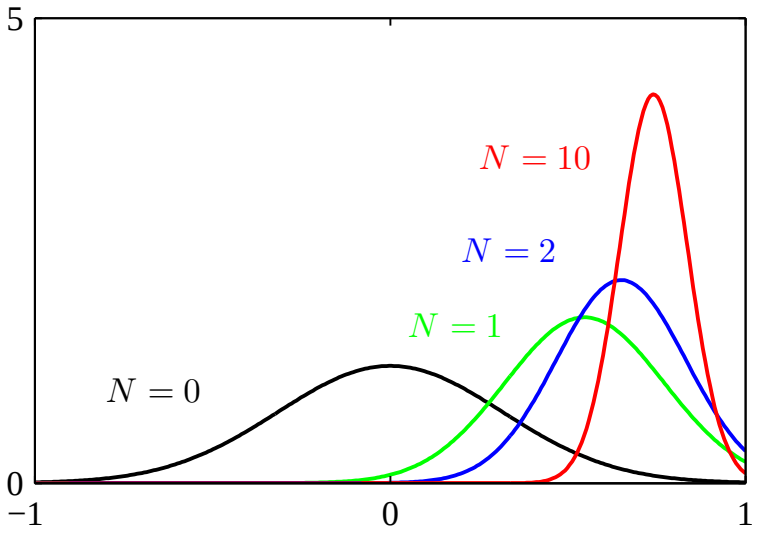
Prior:

$$\begin{aligned} p(\mu) &= \mathcal{N}(\mu|\mu_0, \sigma_0) = \frac{1}{\sqrt{2\pi}\sigma_0}\exp\left(-\frac{1}{2\sigma_0^2}(\mu-\mu_0)^2\right) \\ &= \exp\left(-\frac{1}{2\sigma_0^2}\mu^2 + \frac{\mu_0}{\sigma_0^2}\mu + \text{const.}\right) \end{aligned}$$

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

$$\begin{aligned} p(\mu|\text{Data}) &\propto p(\text{Data}|\mu)p(\mu) \\ &\propto \exp\left(-\frac{1}{2}\left(\frac{N}{\sigma^2}+\frac{1}{\sigma_0^2}\right)\mu^2 + \left(\frac{N\bar{x}}{\sigma^2}+\frac{\mu_0}{\sigma_0^2}\right)\mu + \text{const.}\right) \\ &= \exp\left(-\frac{1}{2\sigma_N^2}\mu^2 + \frac{1}{\sigma_N^2}\mu_N\mu + \text{const.}\right) \propto \exp\left(-\frac{1}{2\sigma_N^2}(\mu-\mu_N)^2 + \text{const.}\right) \end{aligned}$$

$\frac{1}{\sigma_N^2} = \frac{N}{\sigma^2} + \frac{1}{\sigma_0^2}$ and $\mu_N = \sigma_N^2 \left(\frac{N\bar{x}}{\sigma^2} + \frac{\mu_0}{\sigma_0^2} \right)$.

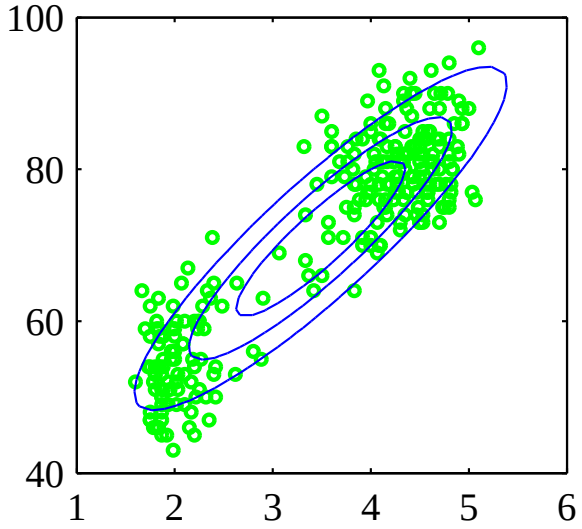


Posterior $p(\mu|D)$ for different number of training points. Prior mean is $\mu = 0$. Likelihood mean is $\mu = 0.8$.

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.

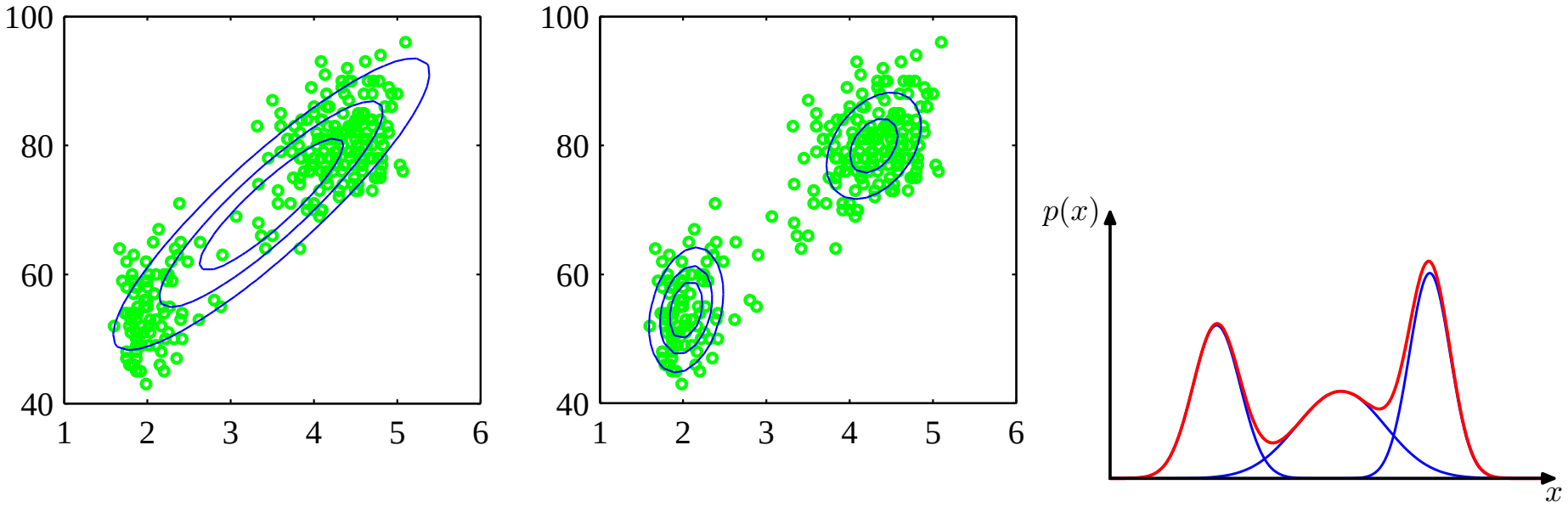
Mixtures of Gaussians

Model for multimodal distribution



Mixtures of Gaussians

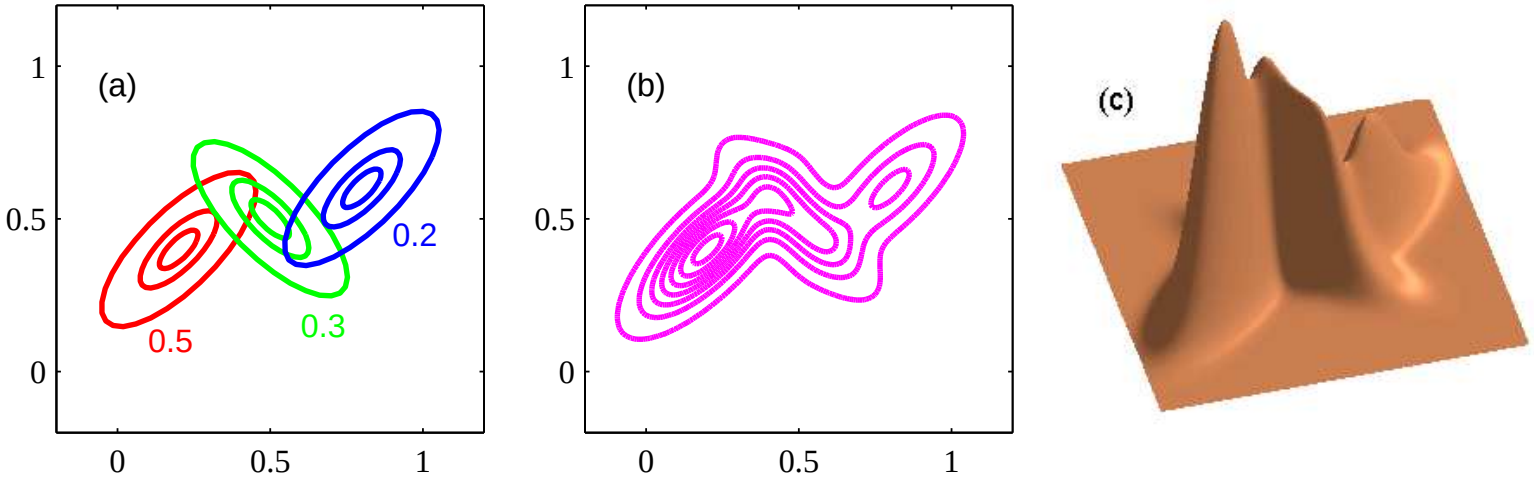
Model for multimodal distribution



$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Each Gaussian is called a *component* of the mixture. The factors π_k are the *mixing coefficients*.

Mixtures of Gaussians



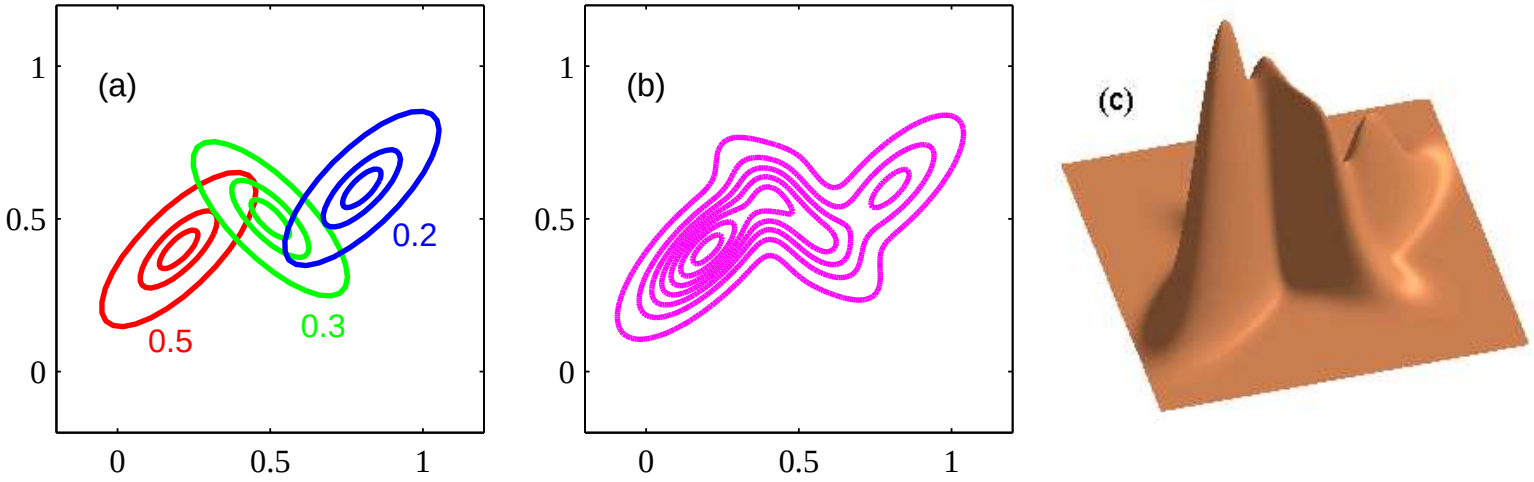
$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Mixing coefficients satisfy

$$\pi_k \geq 0 \quad \text{and} \quad \sum_k \pi_k = 1$$

this implies $p(\boldsymbol{x}) \geq 0$ and $\int p(\boldsymbol{x}) d\boldsymbol{x} = 1$.

Mixtures of Gaussians



$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- k: label of the mixture. Joint distribution $p(\boldsymbol{x}, k) = \pi_k p(\boldsymbol{x}|k)$. Since data is not labeled: marginal distribution $p(\boldsymbol{x}) = \sum_k \pi_k p(\boldsymbol{x}|k)$. $p(\boldsymbol{x}, k)$ in exponential family. However, mixture model $p(\boldsymbol{x})$ not in the exponential family, no simple relation between data and parameters

- One can also consider mixtures of other distributions (mixtures of Bernoulli 9.3.3)

Mixtures of Gaussians

ML: by *gradient ascent* (numerical function maximization).

Learning with hidden variables is generally difficult (slow).

Instead, *EM (Expectation Maximization)* algorithm to deal with hidden variables.

For instance using Bayes' rule compute the "soft assignment' of data x_n to each of the clusters:

$$\gamma_k(x_n) \equiv p(k|x_n) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{k'} \pi_{k'} \mathcal{N}(x_n|\mu_{k'}, \Sigma_{k'})}$$

update according to

$$\mu_k = \frac{\sum_{n=1}^{N} \gamma_k(x_n) x_n}{\sum_{n=1}^{N} \gamma_k(x_n)}$$

See Chapter 9 for more details.

The exponential family

Examples: Gaussian, Bernouilli, Beta, multinomial, Poisson distribution

$$p(\boldsymbol{x}|\boldsymbol{\eta}) = h(\boldsymbol{x})g(\boldsymbol{\eta})\exp(\boldsymbol{\eta}^T\boldsymbol{u}(\boldsymbol{x}))$$

$\boldsymbol{\eta}$: "natural parameters"
$\boldsymbol{u}(\boldsymbol{x})$: (vector) function of $\boldsymbol{x}$, "sufficient statistic"
$g(\boldsymbol{\eta})$ to ensure normalization

The exponential family

Examples: Gaussian, Bernouilli, Beta, multinomial, Poisson distribution

$$p(\boldsymbol{x}|\boldsymbol{\eta}) = h(\boldsymbol{x})g(\boldsymbol{\eta})\exp(\boldsymbol{\eta}^T\boldsymbol{u}(\boldsymbol{x}))$$

$\boldsymbol{\eta}$: "natural parameters"
$\boldsymbol{u}(\boldsymbol{x})$: (vector) function of $\boldsymbol{x}$, "sufficient statistic"
$g(\boldsymbol{\eta})$ to ensure normalization

Example: Bernoulli distribution

$$\begin{aligned} p(x|\mu) &= \text{Bern}(x|\mu) = \mu^x(1-\mu)^{(1-x)} \\ &= \exp\left(x\ln\mu + (1-x)\ln(1-\mu)\right) \\ &= (1-\mu)\exp\left(\ln\left(\frac{\mu}{1-\mu}\right)x\right) \end{aligned}$$

Natural parameters: $\eta = \ln\left(\frac{\mu}{1-\mu}\right)$, sufficient statistic $u(x) = x$.

The exponential family

$$p(\boldsymbol{x}|\boldsymbol{\eta}) = h(\boldsymbol{x})g(\boldsymbol{\eta})\exp(\boldsymbol{\eta}^T\boldsymbol{u}(\boldsymbol{x}))$$

Example: Gaussian distribution

$$\begin{aligned} p(x|\mu,\sigma^2) &= \frac{1}{\sqrt{2\pi\sigma^2}}\exp\big(-\frac{(x-\mu)^2}{2\sigma^2}\big) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}}\exp\big(\frac{\mu}{\sigma^2}x - \frac{1}{2\sigma^2}x^2 - \frac{\mu^2}{2\sigma^2}\big) \\ &= \Big(\frac{1}{\sqrt{2\pi}}\Big)\Big(\frac{1}{\sqrt{\sigma^2}}\exp(-\frac{\mu^2}{2\sigma^2})\Big)\exp\Big([\frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2}]\cdot[x,x^2]^T\Big) \end{aligned}$$

Natural parameters $\boldsymbol{\eta} = (\frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2})^T$, sufficient statistic $\boldsymbol{u}(x) = (x, x^2)^T$, $h(\boldsymbol{x}) = \frac{1}{\sqrt{2\pi}}$.

Rewrite $-\frac{\mu^2}{2\sigma^2} = \frac{\eta_1^2}{4\eta_2}$, and $\frac{1}{\sqrt{\sigma^2}} = \sqrt{-2\eta_2}$, then $g(\boldsymbol{\eta}) = \sqrt{-2\eta_2}\exp\left(\frac{\eta_1^2}{4\eta_2}\right)$.

The exponential family / Maximum likelihood and sufficient statistics

In the ML solution $\boldsymbol{\eta} = \boldsymbol{\eta}_{ML}$, the likelihood is stationary:

$$\nabla_{\boldsymbol{\eta}} \sum_n \log p(\boldsymbol{x}_n|\boldsymbol{\eta}) = 0$$

Since

$$\begin{aligned} \nabla_{\boldsymbol{\eta}} \log p(\boldsymbol{x}_n|\boldsymbol{\eta}) &= \nabla_{\boldsymbol{\eta}} \log \Big(h(\boldsymbol{x}_n) g(\boldsymbol{\eta}) \exp(\boldsymbol{\eta}^T \boldsymbol{u}(\boldsymbol{x}_n)) \Big) \\ &= \nabla_{\boldsymbol{\eta}} \log g(\boldsymbol{\eta}) + \boldsymbol{u}(\boldsymbol{x}_n) \end{aligned}$$

this implies that in ML solution $\boldsymbol{\eta} = \boldsymbol{\eta}_{ML}$

$$-\nabla_{\boldsymbol{\eta}} \log g(\boldsymbol{\eta}) = \frac{1}{N} \sum_n \boldsymbol{u}(\boldsymbol{x}_n)$$

Now, $g(\boldsymbol{\eta})$ is the normalization factor, i.e.,

$$g(\boldsymbol{\eta})^{-1} = \int h(\boldsymbol{x}) \exp(\boldsymbol{\eta}^T \boldsymbol{u}(\boldsymbol{x})) dx$$

So

$$\begin{aligned} -\nabla_{\boldsymbol{\eta}} \log g(\boldsymbol{\eta}) = \nabla_{\boldsymbol{\eta}} \log g(\boldsymbol{\eta})^{-1} &= g(\boldsymbol{\eta}) \nabla_{\boldsymbol{\eta}} \int h(\boldsymbol{x}) \exp(\boldsymbol{\eta}^T \boldsymbol{u}(\boldsymbol{x})) \\ &= g(\boldsymbol{\eta}) \int h(\boldsymbol{x}) \boldsymbol{u}(\boldsymbol{x}) \exp(\boldsymbol{\eta}^T \boldsymbol{u}(\boldsymbol{x})) \\ &= \int \boldsymbol{u}(\boldsymbol{x}) p(\boldsymbol{x}|\boldsymbol{\eta}) dx = \langle \boldsymbol{u}(\boldsymbol{x}) \rangle_{\boldsymbol{\eta}} \end{aligned}$$

and we have that in the ML solution:

$$\frac{1}{N} \sum_n \boldsymbol{u}(\boldsymbol{x}_n) = \langle \boldsymbol{u}(\boldsymbol{x}) \rangle_{\boldsymbol{\eta}_{ML}}$$

ML estimator depends on data only through sufficient statistics $\sum_n \boldsymbol{u}(\boldsymbol{x}_n)$

Conjugate priors

Likelihood:

$$\begin{aligned}\prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\eta}) &= [\prod_n h(\boldsymbol{x}_n)]g(\boldsymbol{\eta})^N \exp(\boldsymbol{\eta}^T \sum_n \boldsymbol{u}(\boldsymbol{x}_n)) \\ &= [\prod_n h(\boldsymbol{x}_n)]g(\boldsymbol{\eta})^N \exp(N\boldsymbol{\eta}^T[\frac{1}{N}\sum_n \boldsymbol{u}(\boldsymbol{x}_n)]) \\ &= [\prod_n h(\boldsymbol{x}_n)]g(\boldsymbol{\eta})^N \exp(N\boldsymbol{\eta}^T\langle\boldsymbol{u}\rangle_{Data})\end{aligned}$$

Conjugate prior

$$p(\boldsymbol{\eta}|\boldsymbol{\chi},\nu) = f(\boldsymbol{\chi},\nu)g(\boldsymbol{\eta})^\nu \exp(\nu\boldsymbol{\eta}^T\boldsymbol{\chi})$$

in which ν: effective number of pseudo data and $\boldsymbol{\chi}$: sufficient statistic.

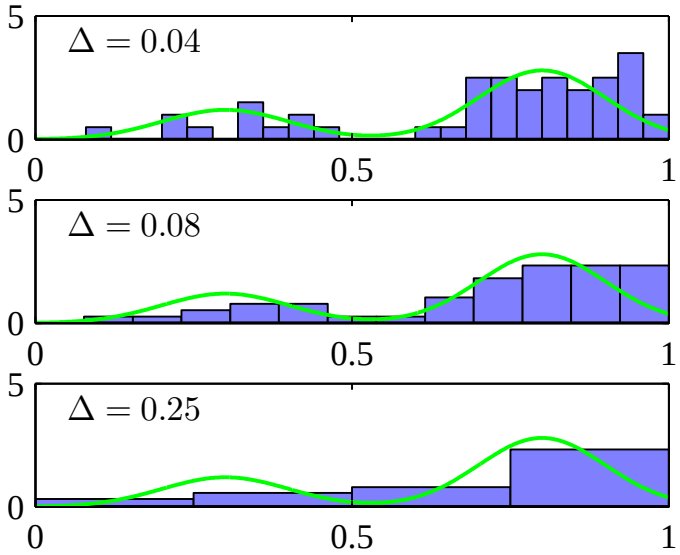
Posterior $\sim$ likelihood $\times$ prior:

$$P(\boldsymbol{\eta}|\boldsymbol{X},\boldsymbol{\chi},\nu) \propto g(\boldsymbol{\eta})^{N+\nu} \exp(\boldsymbol{\eta}^T(N\langle\boldsymbol{u}\rangle_{Data} + \nu\boldsymbol{\chi}))$$

Nonparametric methods/histograms

"Histograms": $p(x) = p_i$ in bin i with width Δ_i, then ML:

$$p_i = \frac{n_i}{N\Delta_i} \qquad \sum_{i=1}^{N} p_i \Delta_i = 1$$



- number of bins exponential in dimension D
- validity requires Δ_i large to estimate p_i, and Δ_i small to have p_i constant.

Nonparametric methods

$$p_i = \frac{n_i}{N\Delta_i} \qquad i = \boldsymbol{x}$$

- Fix Δ_i, determine n_i by data: kernel approach/Parzen window
- Fix $n_i = K$, determine Δ_i by data: K-nearest neighbour

Terminology:

- Parametric distribution: model given by a number of parameters $p(\boldsymbol{x}|\boldsymbol{\theta})$
- Nonparametric distribution: number of parameters grows with the data.

Kernel method

Kernel function:

$$k(\boldsymbol{u}) = \begin{cases} 1, & |u_i| \leq 1/2 \\ 0, & \text{otherwise} \end{cases}$$

$k((\boldsymbol{x} - \boldsymbol{x}_n)/h)$: one if $\boldsymbol{x}_n$ in cube of side h centered around $\boldsymbol{x}$. Total number of data points lying in this cube:

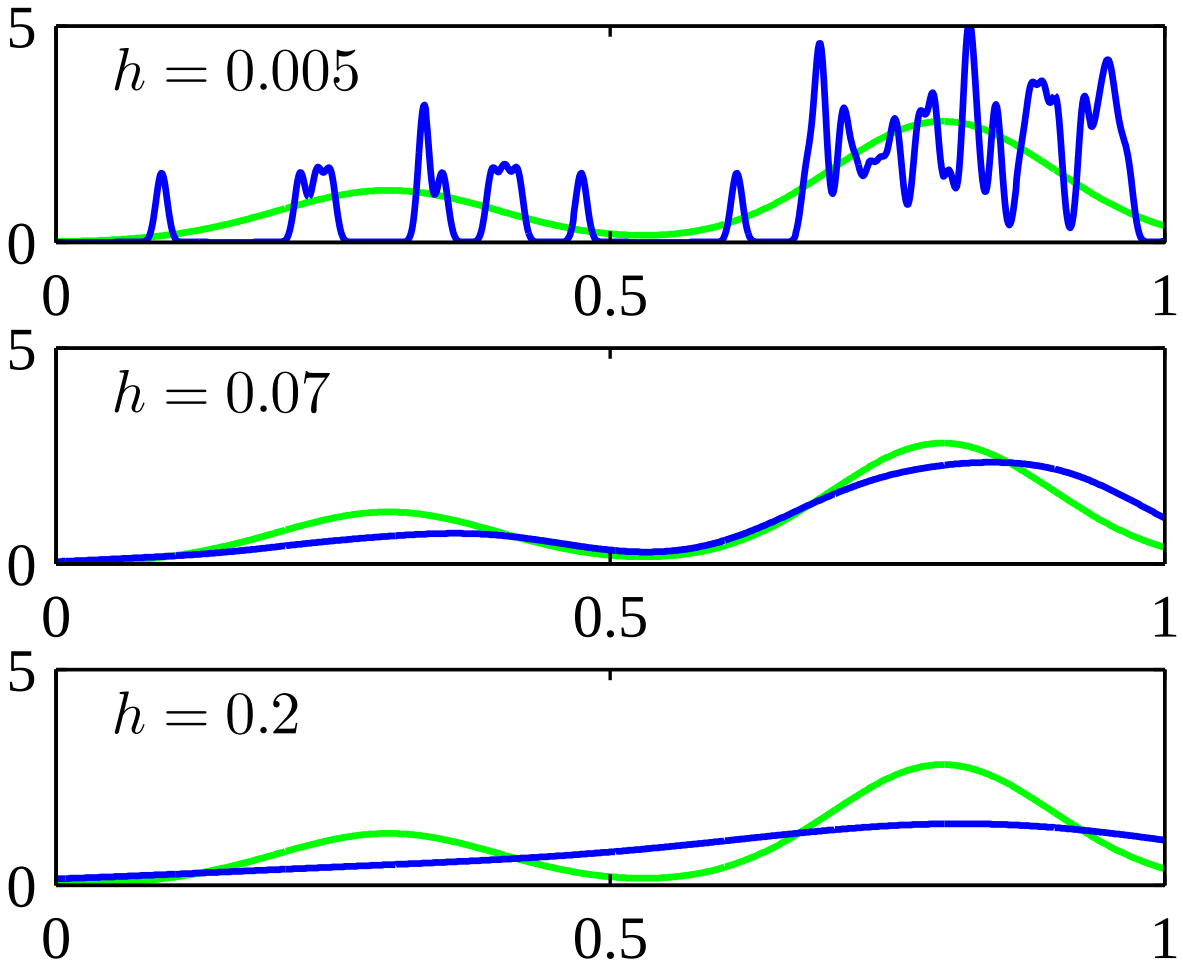
$$n_{\boldsymbol{x}} = \sum_{n=1}^{N} k(\frac{\boldsymbol{x} - \boldsymbol{x}_n}{h})$$

Volume is $V = h^D$, so

$$p(\boldsymbol{x}) = \frac{n_{\boldsymbol{x}}}{NV} = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{h^D} k(\frac{\boldsymbol{x} - \boldsymbol{x}_n}{h})$$

Generalize to any $k(\boldsymbol{u})$ with $\int k(\boldsymbol{u}) d\boldsymbol{u} = 1$.
For example, Gaussian kernel $k(\boldsymbol{u}/h) = \frac{1}{\sqrt{2\pi h^2}} \exp(-||\boldsymbol{u}||^2/2h^2)$.

Kernel methods



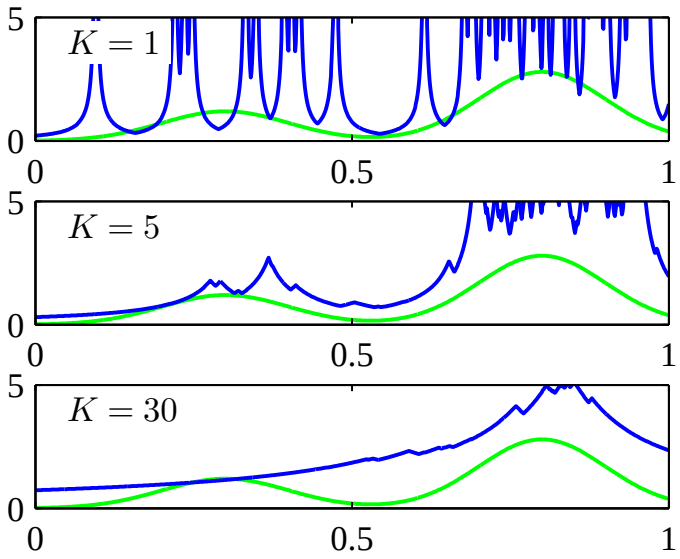
h is smoothing parameter. Trade-off between bias (large h) and variance (small h).

Nearest neighbour methods

$$p(\boldsymbol{x}) = \frac{K}{NV(\boldsymbol{x})}$$

$V(\boldsymbol{x})$=Volume of smallest ball centered on $\boldsymbol{x}$ that contains K data points.

For instance with $K = 1$ and one data point at y in one dimension, we obtain $p(x) = \frac{1}{2N|y-x|}$.



NB: not a true density model (integral over $\boldsymbol{x}$ diverges)

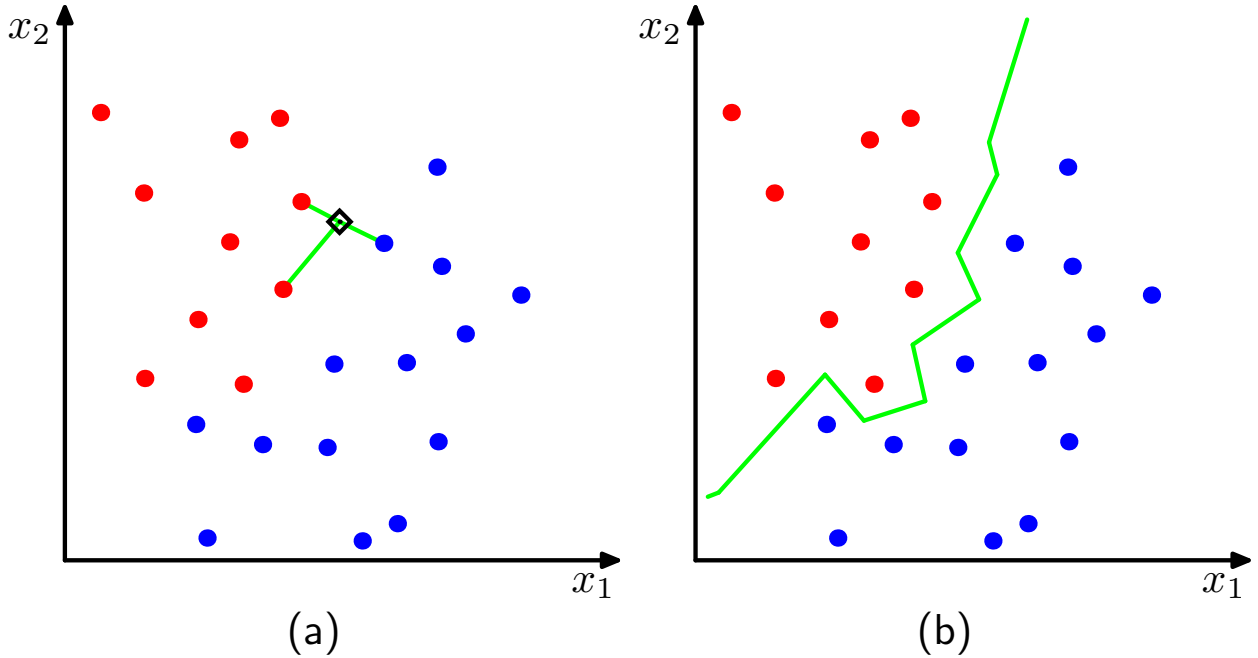
Nearest Neighbor classification

Training set N_k points in class C_k, $\sum_k N_k = N$

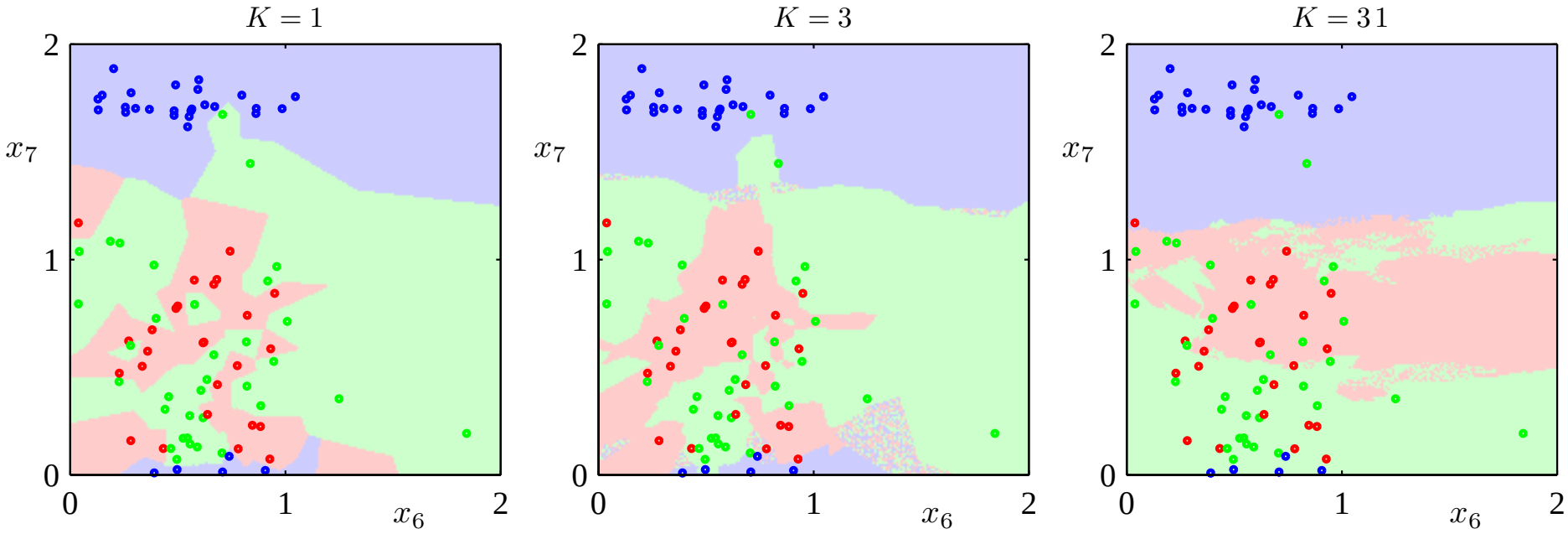
Consider test point $\boldsymbol{x}$ and its K nearest neighbours in volume V. Then

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

K (number of neighbours) is smoothing parameter.



Nearest neighbour methods



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, like the polynomial curve fitting in $\S 1.1$.

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, like the polynomial curve fitting in $\S 1.1$.

This chapter: generalized approach to $linear$ models for regression

- more flexible models
- exploit probabilistic noise models (chapter 2!)
- fully Bayesian predictive distributions
- connection to model selection

Linear Basis Function Models

'Very' 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 "almost" linear (affine) in input $\boldsymbol{x}$.

Most functions are not linear in input $\boldsymbol{x}$ ... can we generalize our model to that?

Linear Basis Function Models

'Very' 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 "almost" linear (affine) in input $\boldsymbol{x}$.

Most functions are not linear in input $\boldsymbol{x}$ ... can we generalize our model to that?

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

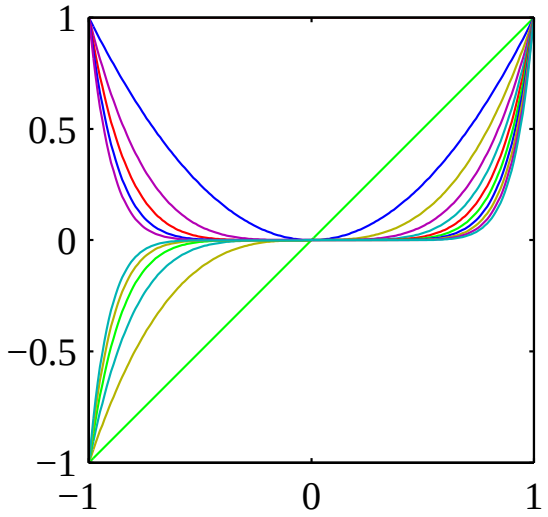
where $\phi_1(\boldsymbol{x}), \ldots, \phi_{M-1}(\boldsymbol{x})$ are *basis functions* or *feature functions*. Defining $\phi_0(\boldsymbol{x}) = 1$, we can write it more compactly:

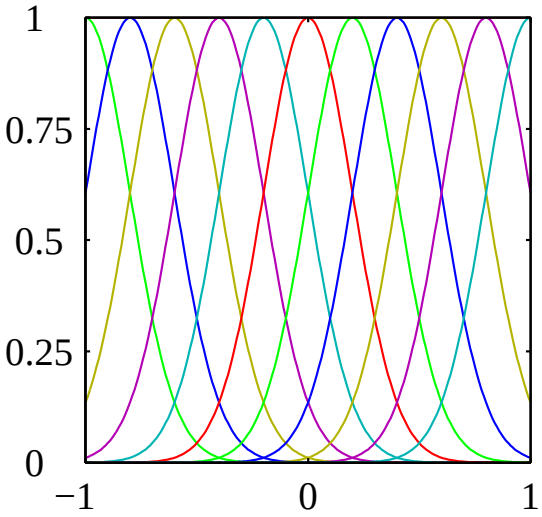
$$y(\boldsymbol{x}, \boldsymbol{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})$$

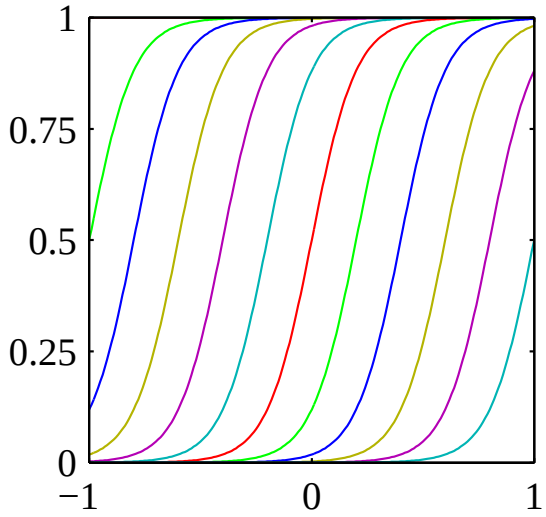
Note: still linear in parameters $\boldsymbol{w}$! Examples of basis functions?

Linear Basis Function Models

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







Adding some noise

Let's introduce 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 $\epsilon \sim \mathcal{N}(0, \beta^{-1})$.

Adding some noise

Let's introduce 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 $\epsilon \sim \mathcal{N}(0, \beta^{-1})$.

Some typical exam questions:

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

Adding some noise

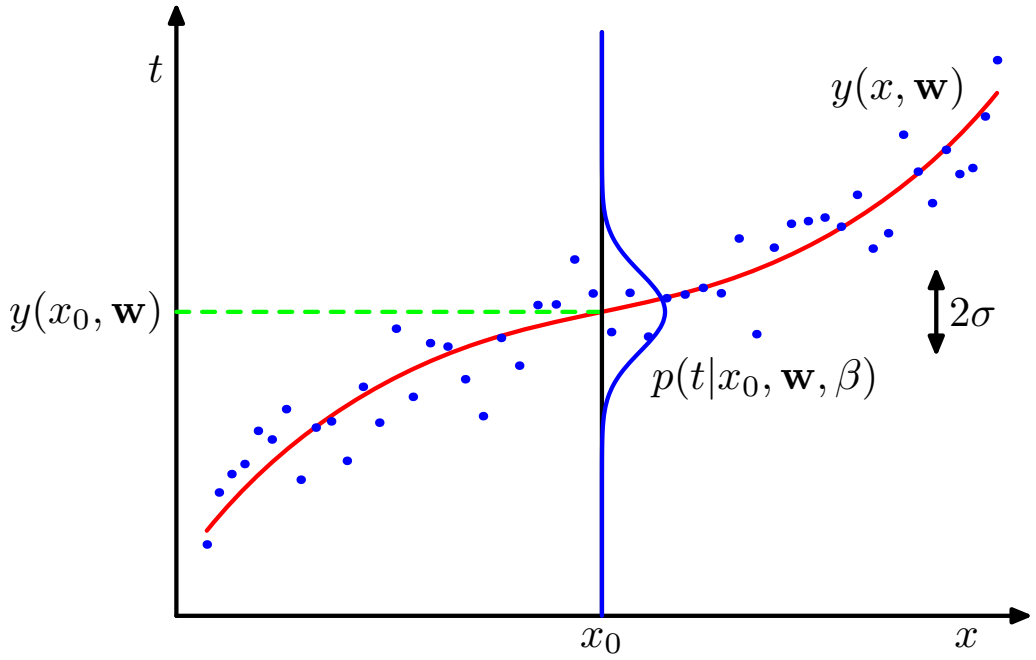
Let's introduce some noise . . . and assume the following model:

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

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

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

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



Adding some noise

Let's introduce 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 conditional mean $\mathbb{E}(t|\boldsymbol{x})$?

Adding some noise

Let's introduce 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 conditional mean $\mathbb{E}(t|\boldsymbol{x})$?

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

Maximum likelihood: least squares

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

Maximum likelihood: least squares

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

$$\begin{aligned}
\ln p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}, \beta) &= \ln \prod_{i=1}^{N} \mathcal{N}\big(t_n|\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n), \beta^{-1}\big) \\
&= \sum_{i=1}^{N} \ln \mathcal{N}\big(t_n|\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n), \beta^{-1}\big) \\
&= \frac{N}{2}\ln\beta - \frac{1}{2}\beta \underbrace{\sum_{i=1}^{N}\big(t_n - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n)\big)^2}_{\text{sum-of-squares error function}} - C
\end{aligned}$$

Q: How to find the maximum likelihood solution for the parameters $\boldsymbol{w}$?

Maximum likelihood: least squares

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

$$\begin{aligned}
\ln p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}, \beta) &= \ln \prod_{i=1}^{N} \mathcal{N}\big(t_n|\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n), \beta^{-1}\big) \\
&= \sum_{i=1}^{N} \ln \mathcal{N}\big(t_n|\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n), \beta^{-1}\big) \\
&= \frac{N}{2}\ln\beta - \frac{1}{2}\beta \underbrace{\sum_{i=1}^{N}\big(t_n - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n)\big)^2}_{\text{sum-of-squares error function}} - C
\end{aligned}$$

Q: How to find the maximum likelihood solution for the parameters $\boldsymbol{w}$?
A: Differentiate with respect to $\boldsymbol{w}$, set to zero, and solve equations to find $\boldsymbol{w}_{ML}$:

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

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 $\boldsymbol{w}$:

$$\boldsymbol{w}_{ML} = \underbrace{(\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T}_{\text{Moore-Penrose pseudo-inverse}} \quad \boldsymbol{t}$$

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

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

Maximizing $p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}, \beta)$ w.r.t. β gives:

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

Regularized least squares

To avoid overfitting, we can add a regularization term to the log-likelihood, e.g. *weight decay*:

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

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

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

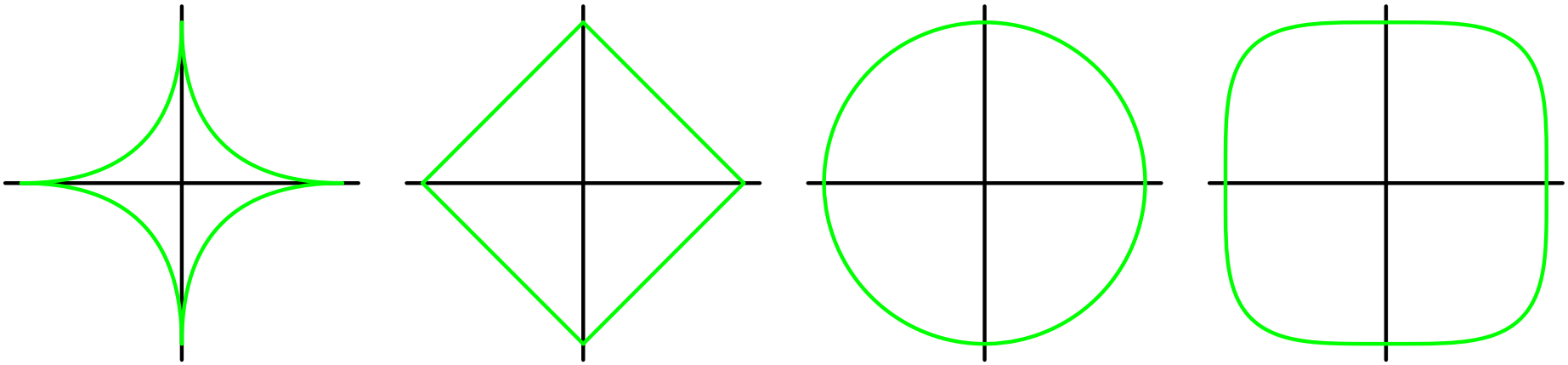
Other regularizers also possible, e.g. a more general error term would be

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

.

Regularized least squares

For example $q = 1$ gives the famous LASSO with regularizer $\lambda \sum_{j=1}^{M} |w_j|$.



$q = 0.5$ $q = 1$ $q = 2$ $q = 4$

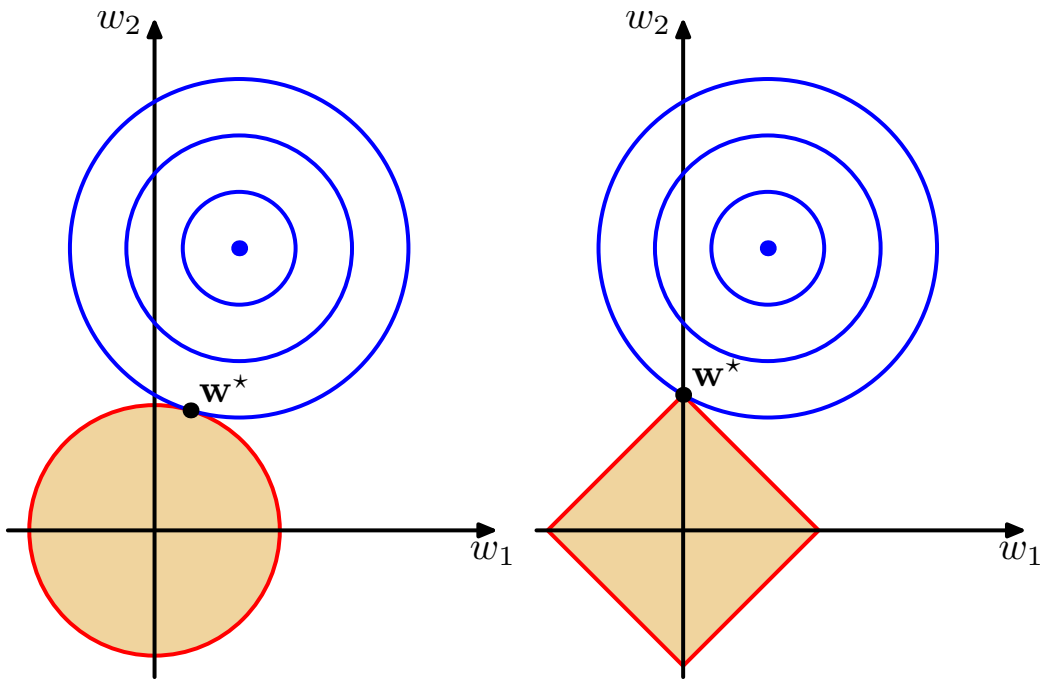
Q: Why does the LASSO result in a *sparse* solution ($w_j \neq 0$ for only a few j's)?

Regularized least squares

A: The solution $\boldsymbol{w}$ to the general regularized error term

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

can be viewed as the *unregularized* solution under constraint $\sum_{j=1}^{M}\left|w_j\right|^q \leq \eta$.



Bias-Variance Decomposition

Complex models tend to overfit. Simple models tend to be too rigid. Number of terms in polynomial, weight decay constant λ.

(Treatment of 1.5.5). Consider a regression problem with squared loss function

$$\mathcal{E}(L) = \iint \big(y(\boldsymbol{x}) - t\big)^2 p(\boldsymbol{x}, t) d\boldsymbol{x} dt$$

$p(\boldsymbol{x}, t)$ is the true underlying model, $y(\boldsymbol{x})$ is our estimate of t at $\boldsymbol{x}$.

The optimal y minimizes $\mathcal{E}(L)$:

$$\begin{aligned} \frac{\delta \mathcal{E}(L)}{\delta y(\boldsymbol{x})} &= 2 \int \big(y(\boldsymbol{x}) - t\big) p(\boldsymbol{x}, t) dt = 0 \\ y(\boldsymbol{x}) &= \frac{\int t p(\boldsymbol{x}, t) dt}{p(\boldsymbol{x})} = \int t p(t|\boldsymbol{x}) dt = \mathbb{E}(t|\boldsymbol{x}) \end{aligned}$$

Bias-Variance Decomposition

The expected squared loss can also be written as

$$\begin{aligned}
\mathcal{E}(L) &= \iint \big(y(\boldsymbol{x}) - t\big)^2 p(\boldsymbol{x}, t) d\boldsymbol{x} dt \\
&= \iint \big(y(\boldsymbol{x}) - \mathbb{E}(t|\boldsymbol{x}) + \mathbb{E}(t|\boldsymbol{x}) - t\big)^2 p(\boldsymbol{x}, t) d\boldsymbol{x} dt \\
&= \iint \Big(\big(y(\boldsymbol{x}) - \mathbb{E}(t|\boldsymbol{x})\big)^2 + \big(\mathbb{E}(t|\boldsymbol{x}) - t\big)^2 \\
&\quad + 2\big(y(\boldsymbol{x}) - \mathbb{E}(t|\boldsymbol{x})\big)\big(\mathbb{E}(t|\boldsymbol{x}) - t\big)\Big) p(\boldsymbol{x}, t) d\boldsymbol{x} dt \\
&= \int \big(y(\boldsymbol{x}) - \mathbb{E}(t|\boldsymbol{x})\big)^2 p(\boldsymbol{x}) d\boldsymbol{x} + \underbrace{\iint \big(\mathbb{E}(t|\boldsymbol{x}) - t\big)^2 p(\boldsymbol{x}, t) d\boldsymbol{x} dt}_{\text{intrinsic noise}}
\end{aligned}$$

First term depends on $y(\boldsymbol{x})$ and is minimized when $y(\boldsymbol{x}) = \mathbb{E}(t|\boldsymbol{x})$. The second term, the variance in the conditional distribution of t given $\boldsymbol{x}$, averaged over $\boldsymbol{x}$: $\int \mathrm{var}(t|\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}$ is *independent* of the solution $y(\boldsymbol{x})$. The cross-term vanishes on integration over t (odd function around $\mathbb{E}(t|\boldsymbol{x})$).

Bias-Variance Decomposition

In the case of a finite data set $\mathcal{D}$, our learning algorithm will give a solution $y(\boldsymbol{x};\mathcal{D})$ that will not minimize the exact expected square loss (but an empirical square loss): different data sets will give different solutions $y(\boldsymbol{x};\mathcal{D})$.

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

$$\begin{aligned}\big(y(\boldsymbol{x};\mathcal{D}) - \mathbb{E}(t|\boldsymbol{x})\big)^2 &= \big(y(\boldsymbol{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}(y) + \mathbb{E}_{\mathcal{D}}(y) - \mathbb{E}(t|\boldsymbol{x})\big)^2 \\ &= \big(y(\boldsymbol{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}(y)\big)^2 + \big(\mathbb{E}_{\mathcal{D}}(y) - \mathbb{E}(t|\boldsymbol{x})\big)^2 \\ &+ 2\big(y - \mathbb{E}_{\mathcal{D}}(y)\big)\big(\mathbb{E}_{\mathcal{D}}(y) - \mathbb{E}(t|\boldsymbol{x})\big)\end{aligned}$$

So for a given $\boldsymbol{x}$ the average of this quantity over many data sets

$$\mathbb{E}_{\mathcal{D}}\Big[\big(y(\boldsymbol{x};\mathcal{D}) - \mathbb{E}(t|\boldsymbol{x})\big)^2\Big] = \underbrace{\big(\mathbb{E}_{\mathcal{D}}(y) - \mathbb{E}(t|\boldsymbol{x})\big)^2}_{(\text{bias})^2} + \underbrace{\mathbb{E}_{\mathcal{D}}\big[\{y(\boldsymbol{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}(y)\}^2\big]}_{\text{variance}}$$

as the cross term vanishes on averaging.

Bias-Variance Decomposition

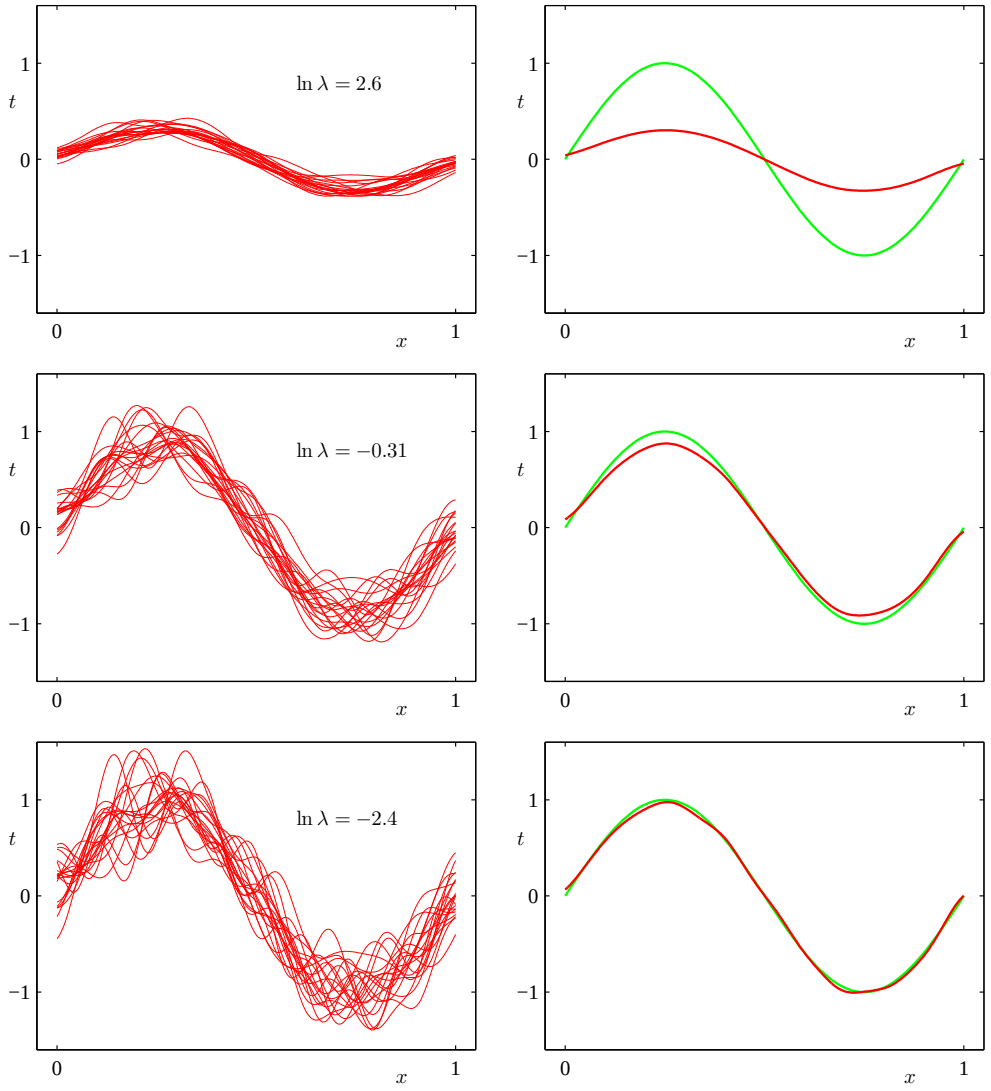
Substitution of the previous expression in the expected square loss:

$$\begin{aligned}
\mathcal{E}(L) &= \underbrace{\int \big(\mathbb{E}_{\mathcal{D}}(y) - \mathbb{E}(t|\boldsymbol{x})\big)^2 p(\boldsymbol{x}) d\boldsymbol{x}}_{(\text{bias})^2} \\
&+ \underbrace{\int \mathbb{E}_{\mathcal{D}}\big[\{y(\boldsymbol{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}(y)\}^2\big] p(\boldsymbol{x}) d\boldsymbol{x}}_{\text{variance}} \\
&+ \underbrace{\iint \big(\mathbb{E}(t|\boldsymbol{x}) - t\big)^2 p(\boldsymbol{x}, t) d\boldsymbol{x} dt}_{\text{noise}}
\end{aligned}$$

Expected square loss = bias^2 + variance + noise

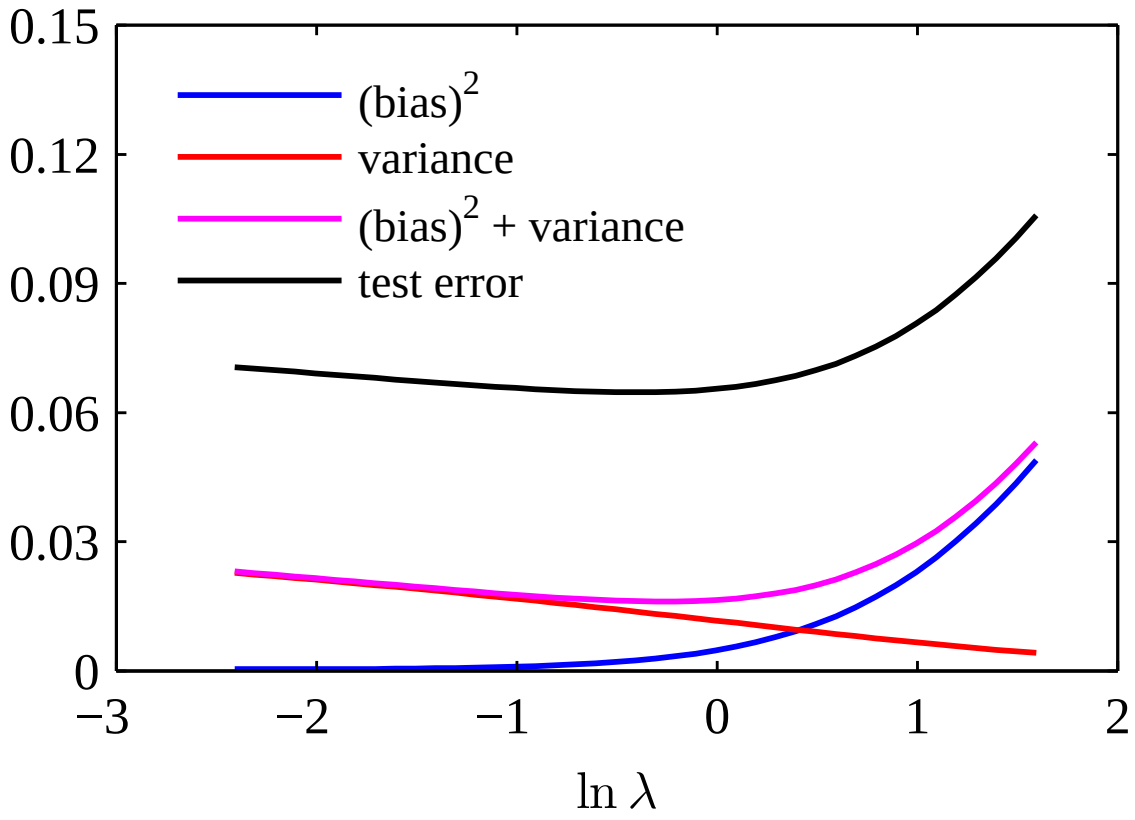
Bias^2: difference between average solution $\mathbb{E}_{\mathcal{D}}(y(\boldsymbol{x};\mathcal{D}))$ and true solution $\mathbb{E}(t|\boldsymbol{x})$.
Variance: scatter of individual solutions $y(\boldsymbol{x},\mathcal{D})$ around their mean $\mathbb{E}_{\mathcal{D}}(y(\boldsymbol{x};\mathcal{D}))$.
Noise: (true) scatter of the data points t around their mean $\mathbb{E}(t|\boldsymbol{x})$.

Bias-Variance Decomposition



100 data sets, each with 25 data points from $t = \sin(2\pi x) + \text{noise}$. $y(x)$ as in Eq. 3.3-4. Parameters optimized using Eq. 3.27 for different λ. Left shows variance, right shows bias.

Bias-Variance Decomposition



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

True bias is usually not known, because $\mathbb{E}(t|\boldsymbol{x})$ 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:

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

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:

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

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:

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

Then, by applying $2.113 + 2.114 \Rightarrow 2.116$, we get the posterior $p(\boldsymbol{w}|\boldsymbol{t})$.

Bayesian linear regression

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

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

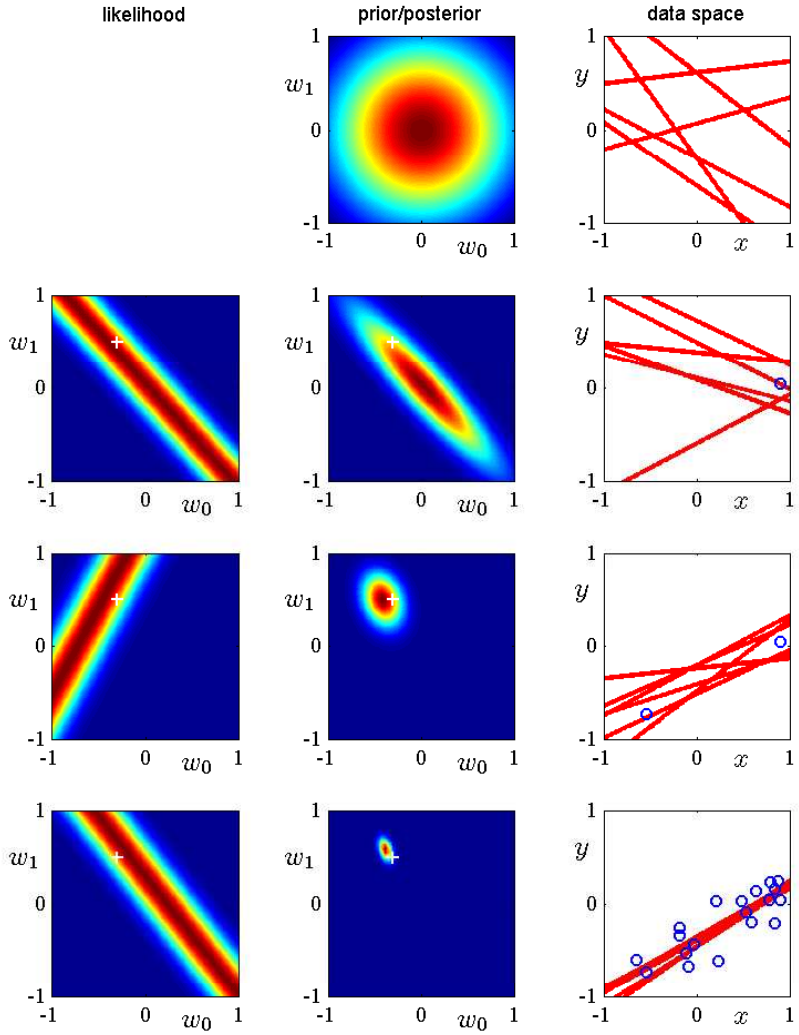
Then, by applying $2.113 + 2.114 \Rightarrow 2.116$, we get the posterior $p(\boldsymbol{w}|\boldsymbol{t})$.

$$\begin{aligned} p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_0, \boldsymbol{S}_0) &\quad\leftrightarrow\quad p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \\ p(\boldsymbol{t}|\boldsymbol{w}) = \mathcal{N}(\boldsymbol{t}|\Phi\boldsymbol{w}, \beta^{-1}\boldsymbol{I}) &\quad\leftrightarrow\quad p(\boldsymbol{y}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}, \boldsymbol{L}^{-1}) \\ p(\boldsymbol{w}|\boldsymbol{t}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_N, \boldsymbol{S}_N) &\quad\leftrightarrow\quad p(\boldsymbol{x}|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\Sigma}\{\boldsymbol{A}^T\boldsymbol{L}(\boldsymbol{y} - b) + \boldsymbol{\Lambda}\boldsymbol{\mu}\}, \boldsymbol{\Sigma}) \\ &\qquad\qquad \text{with } \boldsymbol{\Sigma} = (\boldsymbol{\Lambda} + \boldsymbol{A}^T\boldsymbol{L}\boldsymbol{A})^{-1} \end{aligned}$$

$$\text{So } p(\boldsymbol{w}|\boldsymbol{t}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_N, \boldsymbol{S}_N) \quad \text{, with} \quad \begin{aligned} \boldsymbol{m}_N &= \boldsymbol{S}_N(\boldsymbol{S}_0^{-1}\boldsymbol{m}_0 + \beta\Phi^T\boldsymbol{t}) \\ \boldsymbol{S}_N^{-1} &= \boldsymbol{S}_0^{-1} + \beta\Phi^T\Phi \end{aligned}$$

Note behaviour when $\boldsymbol{S}_0^{-1} \to \boldsymbol{0}$ (broad prior), when $N = 0$, and when $N \to \infty$.

Bayesian linear regression



Data: $t = a_0 + a_1 x + \text{noise}$. Model: $y(x, \boldsymbol{w}) = w_0 + w_1 x$; $p(t|x, \boldsymbol{w}, \beta) = \mathcal{N}(t|y(x, \boldsymbol{w}), \beta^{-1})$, $\beta^{-1} = (0.2)^2$; $p(\boldsymbol{w}|\alpha) = \mathcal{N}(\boldsymbol{w}|0, \alpha^{-1}\boldsymbol{I})$, $\alpha = 2$.

Predictive distribution

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

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

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

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

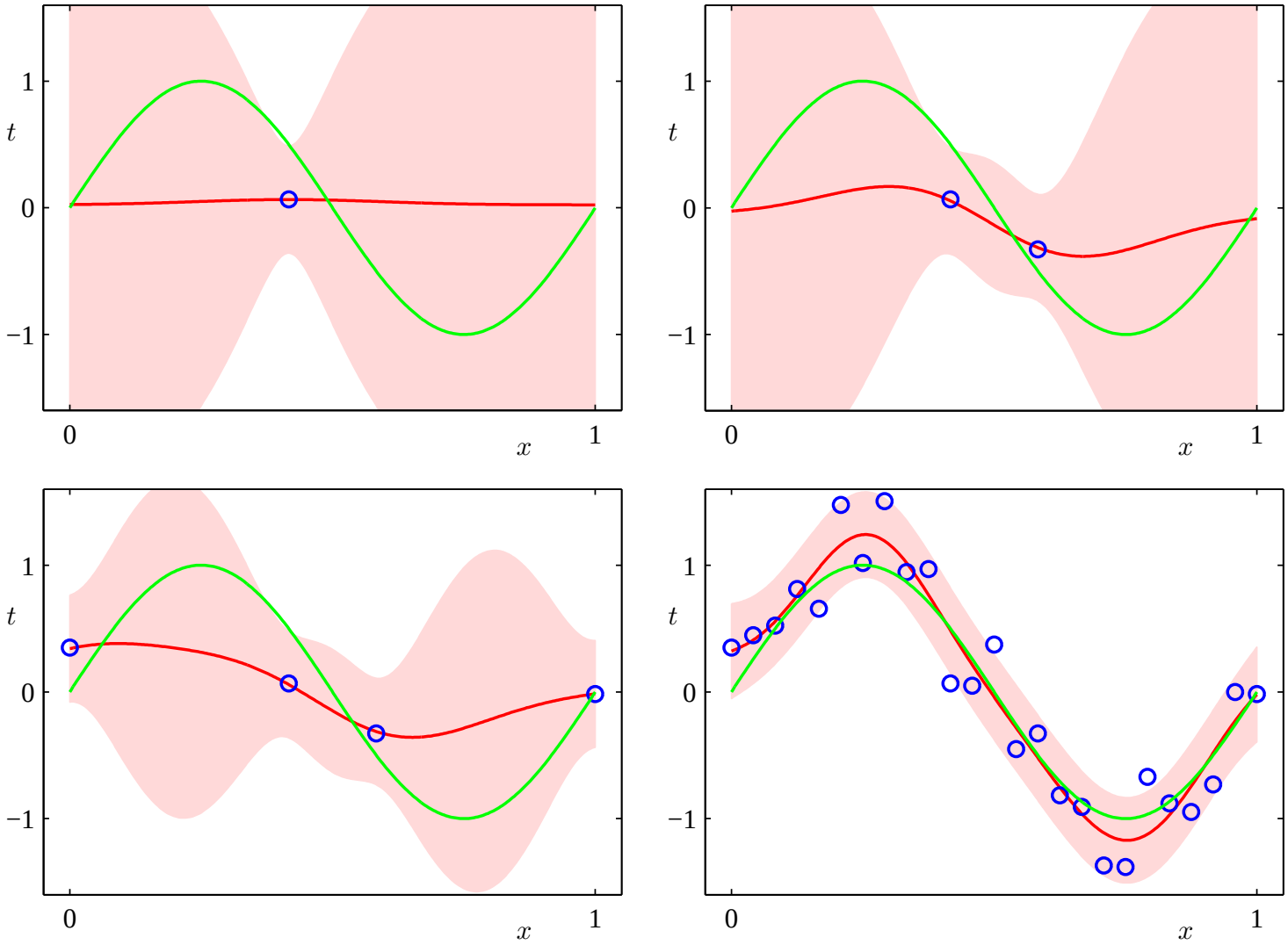
$$p(t^*|x^*, \boldsymbol{t}, \boldsymbol{x}) = \mathcal{N}(t^*|\boldsymbol{\phi}^T(x^*)\boldsymbol{m}_N, \sigma_N^2(x^*))$$

where

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

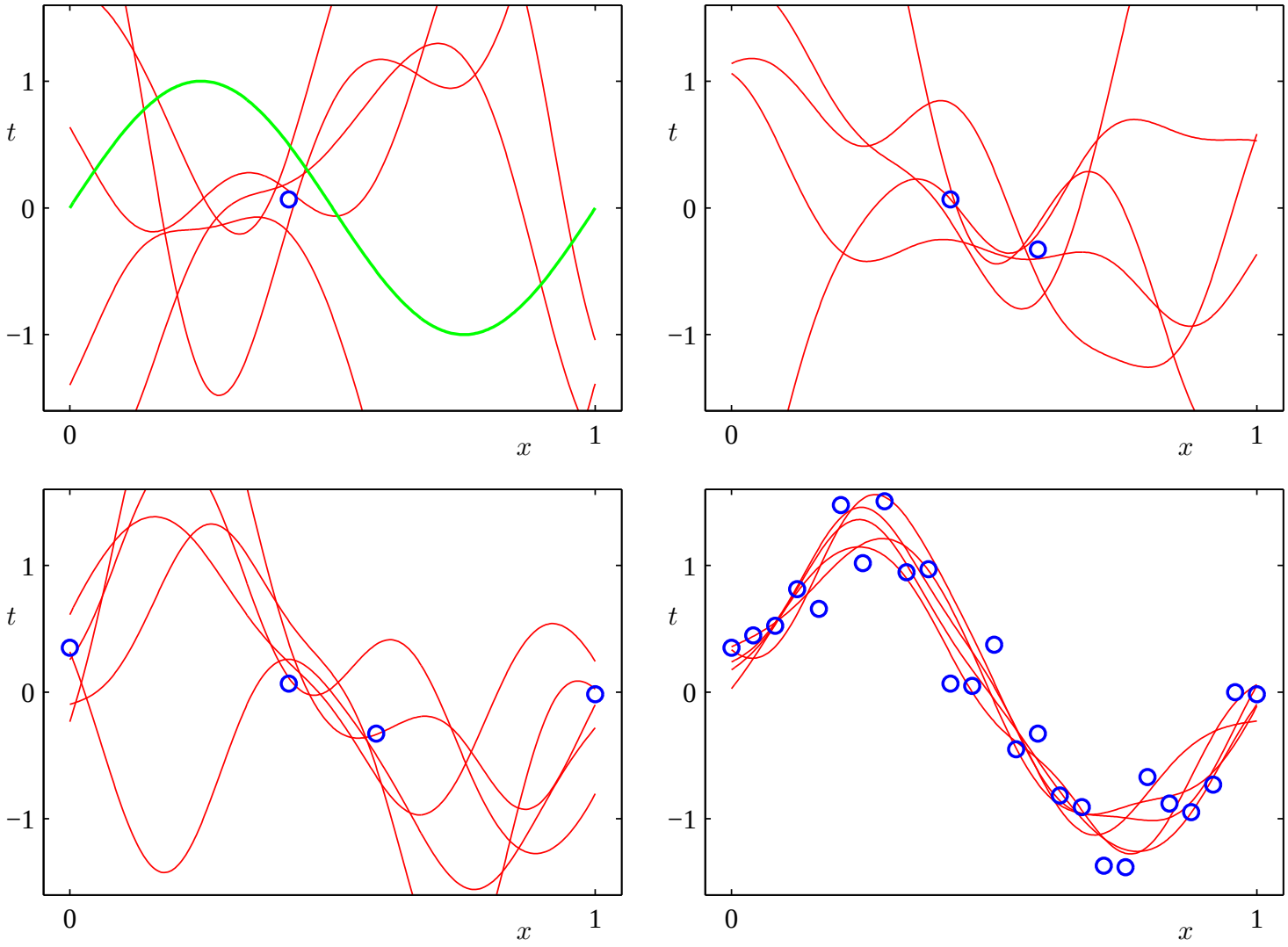
When $N \to \infty$ $S_N \to 0$ and $\sigma_N^2(x^*) \to \beta^{-1}$

Bayesian linear regression: Example



Data points from $t = \sin(2\pi x) + \text{noise}$. $y(x)$ as in Eq. 3.3-4. 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.

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:

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

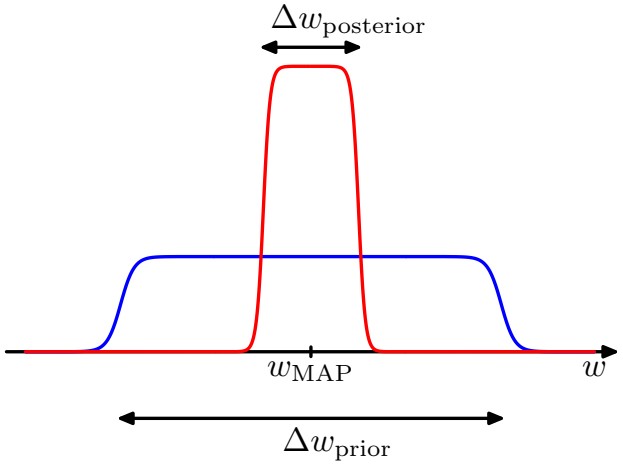
Bayesian model comparison

Flat prior over models $p(\mathcal{M}_i) \implies$ model evidence is proportional to marginal likelihood:

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

We use a very crude estimate of this integral to understand the mechanism of Bayesian model selection.

Consider a one dimensional case (one parameter). Assume roughly block-shaped prior $p(w)$ and posterior $p(w|\mathcal{D})$ as in the figure. Then the prior is $p(w) = 1/\Delta w_{\text{prior}}$ on an interval of width Δw_{prior}, and zero elsewhere. The marginal likelihood is then approximately given by:



$$\begin{aligned} p(\mathcal{D}|\mathcal{M}_i) &\approx \int p(\mathcal{D}|w_{\text{MAP}}, \mathcal{M}_i) \frac{1}{\Delta w_{\text{prior}}} d\boldsymbol{w} \\ &\approx p(\mathcal{D}|w_{\text{MAP}}, \mathcal{M}_i) \frac{\Delta w_{\text{posterior}}}{\Delta w_{\text{prior}}} \end{aligned}$$

Bayesian model comparison

As $\Delta w_{\mathrm{posterior}} < \Delta w_{\mathrm{prior}}$, the Bayesian approach gives a $negative$ correction to the ML estimate.

$$\ln p(\mathcal{D}|\mathcal{M}_i) \quad \approx \quad \ln p(\mathcal{D}|w_{\mathrm{MAP}}, \mathcal{M}_i) + \ln \left(\frac{\Delta w_{\mathrm{posterior}}}{\Delta w_{\mathrm{prior}}} \right)$$

With M_i parameters in model $\mathcal{M}_i$, the same argument gives

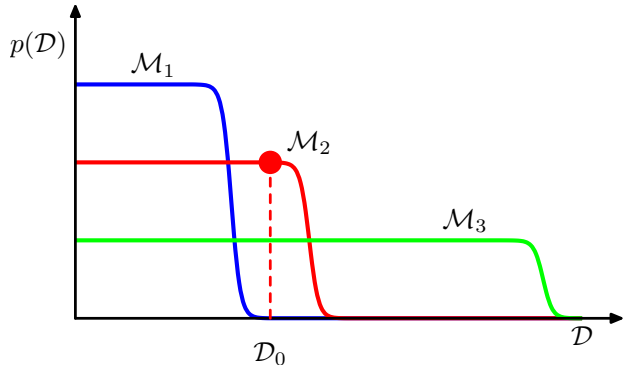
$$\ln p(\mathcal{D}|\mathcal{M}_i) \quad \approx \quad \ln p(\mathcal{D}|\boldsymbol{w}_{\mathrm{MAP}}, \mathcal{M}_i) + M_i \ln \left(\frac{\Delta w_{\mathrm{posterior}}}{\Delta w_{\mathrm{prior}}} \right)$$

With increasing model complexity the first term increases (better fit), but the second term decreases.

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 (too low likelihood) and model $\mathcal{M}_3$ is too complex (too large penalty term). This is known as Bayesian model selection.

The 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,

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

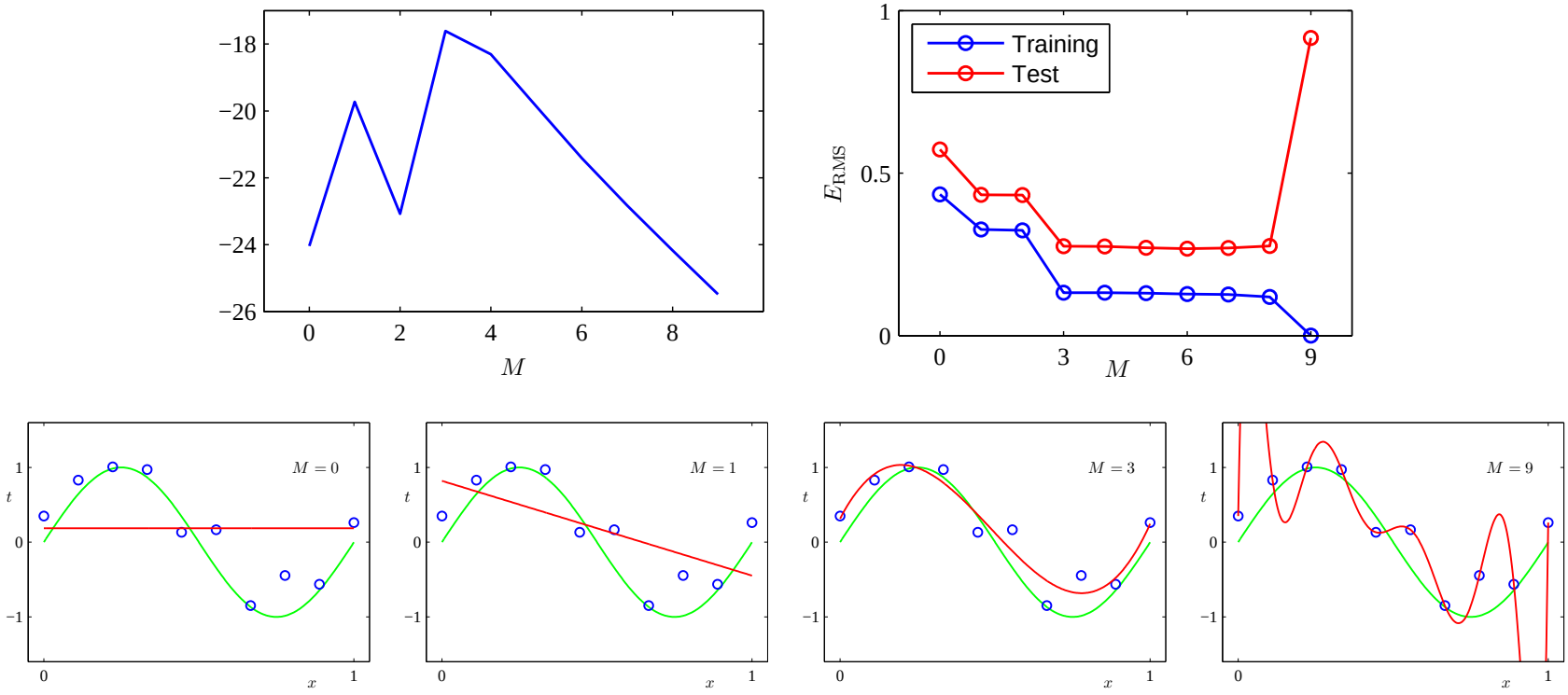
With $\int d\boldsymbol{w} \exp(-E(\boldsymbol{w})) = \exp(-E(\boldsymbol{m}))(2\pi)^{-M/2}|A|^{-1/2}$:

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

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

[8] 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.

Chapter 4

Linear models for classification

- Discriminant function
- Conditional probabilities $P(C_k|\boldsymbol{x})$
- Generative models $P(\boldsymbol{x}|C_k)$

Generalized linear models for classification

$$y(\boldsymbol{x}) = f(\boldsymbol{w}^T\boldsymbol{x} + w_0)$$

with $f(\cdot)$ nonlinear.

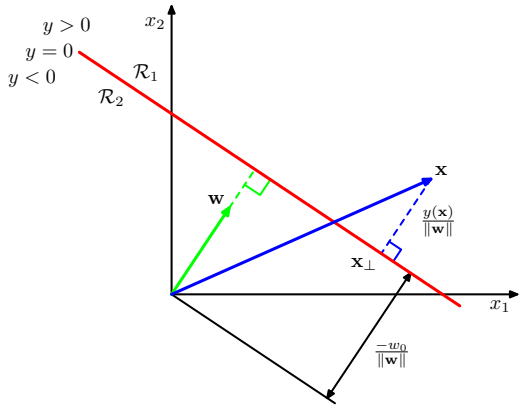
4.1 Discriminant functions

Two category classification problem:

$$y(\boldsymbol{x}) > 0 \Leftrightarrow \boldsymbol{x} \in C_1 \qquad y(\boldsymbol{x}) < 0 \Leftrightarrow \boldsymbol{x} \in C_2$$

with

$$y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$$



If $\boldsymbol{x}_1, \boldsymbol{x}_2$ on the line $y(\boldsymbol{x}) = 0$ then $\boldsymbol{w}^T(\boldsymbol{x}_1 - \boldsymbol{x}_2) = 0$. Line is orthogonal to $\boldsymbol{w}$.

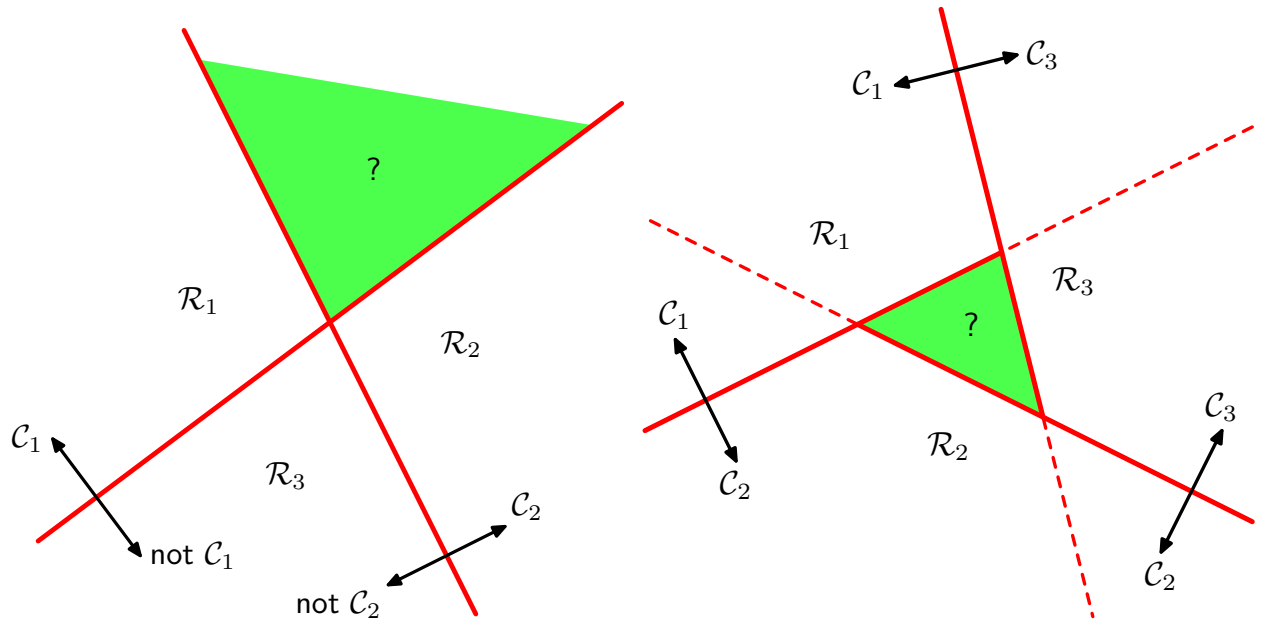
Notation: $\tilde{\boldsymbol{w}} = (w_0, \boldsymbol{w})$ and $\tilde{\boldsymbol{x}} = (1, \boldsymbol{x})$, then

$$y(\boldsymbol{x}) = \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}$$

Multiple classes

$$y_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \boldsymbol{x} + w_{k0} = \sum_{i=1}^{k} w_{ki} x_i + w_{k0} = \sum_{i=0}^{k} w_{ki} x_i$$

Naive approach is to let each boundary separate class k from the rest. This does not work.

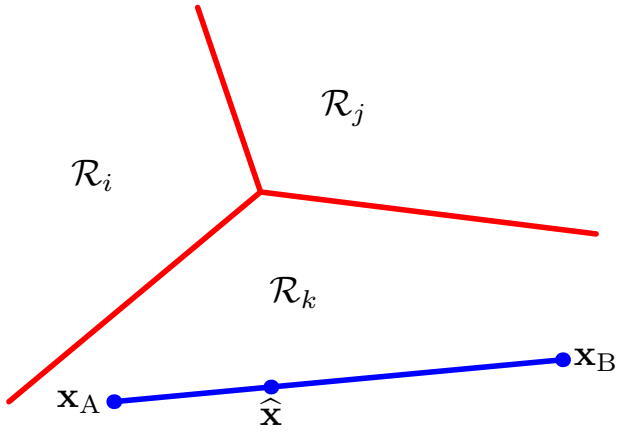


One-versus-the-rest classifier.
Use $K - 1$ classifiers, each classifying one versus the rest.

One-versus-one classifier.
Use $K(K-1)/2$ classifiers, each classifying one pair k, k'.

Multiple classes

Instead, define decision boundaries as: $y_k(\boldsymbol{x}) = y_j(\boldsymbol{x})$.



If $y_k(\boldsymbol{x}) = \boldsymbol{w}_k^T\boldsymbol{x} + w_{0k}, k = 1, 2$ then decision boundary is $(\boldsymbol{w}_1 - \boldsymbol{w}_2)^T\boldsymbol{x} + w_{01} - w_{02} = 0$.

For $K = 3$, we get three lines: $y_1(\boldsymbol{x}) = y_2(\boldsymbol{x})$, $y_1(\boldsymbol{x}) = y_3(\boldsymbol{x})$, $y_2(\boldsymbol{x}) = y_3(\boldsymbol{x})$, that cross in a common point $y_1(\boldsymbol{x}) = y_2(\boldsymbol{x}) = y_2(\boldsymbol{x})$, and which defines the tesselation.

The class regions $\mathcal{R}_k$ are convex. If $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{R}_k$

$$y_k(\boldsymbol{x}) > y_j(\boldsymbol{x}), y_k(\boldsymbol{y}) > y_j(\boldsymbol{y}) \Rightarrow y_k(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{y}) > y_j(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{y})$$

Learning vector quantization

Here is a way to get a nice Voneroi Tesselation, known as learning vector quantization (LVQ) by Kohonen (1988).

Consider labeled data $\{(\vec{x}^\mu, c^\mu), \mu = 1, \ldots, P\}$ with $\vec{x}^\mu \in R^n$ and $c^\mu = 1, \ldots, K$ the class label.

Initialize cluster vectors $\vec{m}_k, k = 1, \ldots, K_1$. The simplest choice is $K_1 = K$, ie. one vector per class, but one can also consider $K_1 > K$.

The algorithm iterates over the data:

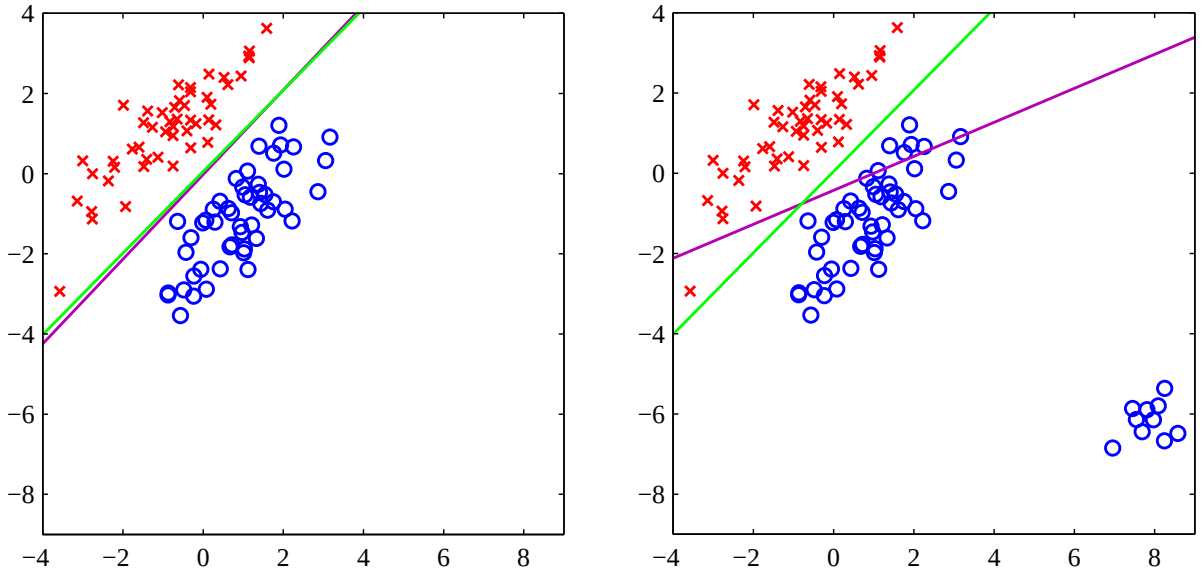
- Choose data sample $\vec{x}^\mu$ and compute the nearest cluster vector: $c = \mathrm{argmin}_k \|\vec{x}^\mu - \vec{m}_k\|$.
- If $c^\mu = c$ (the nearest cluster vector has the correct class), then $\Delta\vec{m}_c = \eta(\vec{x}^\mu - \vec{m}_c)$.
- If $c^\mu \neq c$ (the nearest cluster vector has another class), then $\Delta\vec{m}_c = -\eta(\vec{x}^\mu - \vec{m}_c)$.
- All other cluster vectors are unaltered.

Least-squares technique

Learning the parameters is done by minimizing a cost function, for instance the minus log likelihood or the quadratic cost:

$$
\begin{aligned}
E &= \frac{1}{2}\sum_n \sum_k (\sum_i \tilde{x}_{ni}\tilde{w}_{ik} - t_k^n)^2 \\
\frac{\partial E}{\partial \tilde{w}_{jk}} &= \sum_n (\sum_i \tilde{x}_{ni}\tilde{w}_{ik} - t_k^n)\tilde{x}_{nj} = 0 \\
&\quad \sum_i \left(\sum_n \tilde{x}_{nj}\tilde{x}_{ni} \right) \tilde{w}_{ik} = \sum_n t_k^n x_{nj} \\
\tilde{\boldsymbol{w}}_k &= \boldsymbol{C}^{-1}\boldsymbol{\mu}_k \quad C_{ij} = \sum_n \tilde{x}_{nj}\tilde{x}_{ni} \quad \mu_{kj} = \sum_n t_k^n x_{nj}
\end{aligned}
$$

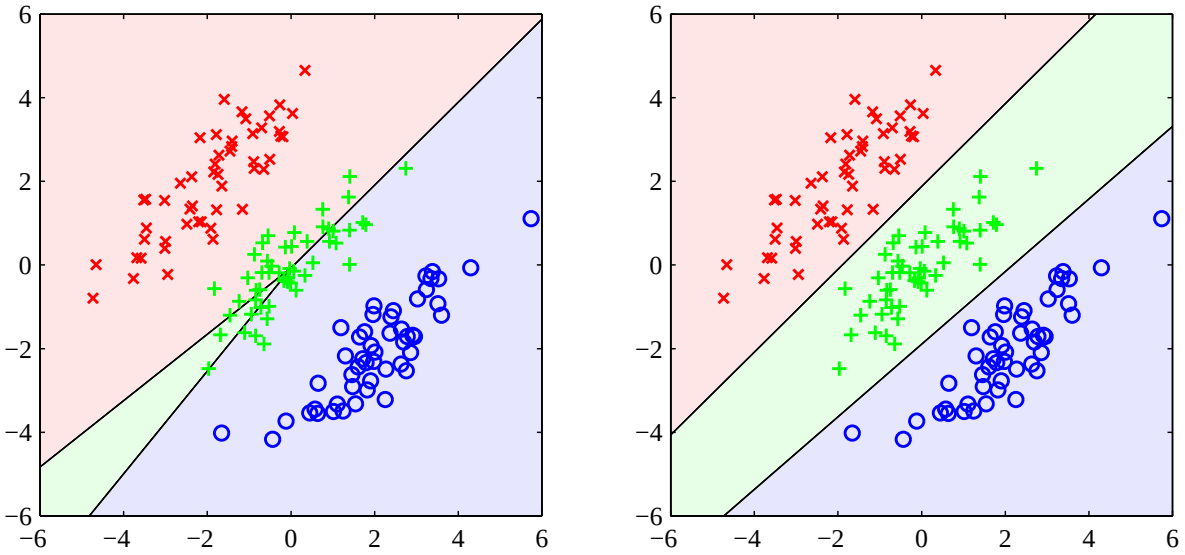
Least-squares technique



Two class classification problem solved with least square (magenta) and logistic regression (green)

Least square solution sensitive to outliers: far away data contribute too much to the error.

Least-squares technique



Three class classification problem solved with least square (left) and logistic regression (right)

Least square solution poor in multi-class case.

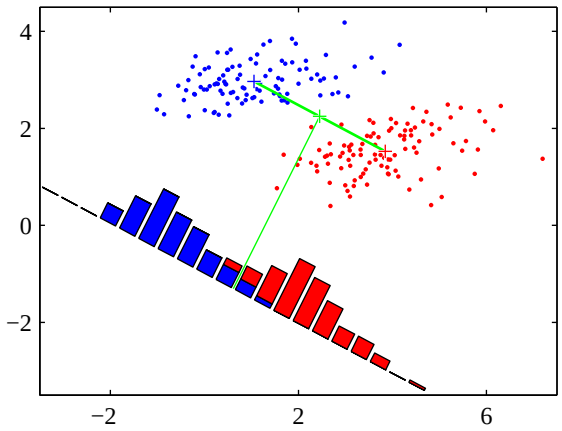
Fisher's linear discriminant

Consider two classes. Take an $\boldsymbol{x}$ and project it down to one dimension using

$$y = \boldsymbol{w}^T \boldsymbol{x}$$

Let the two classes have two means:

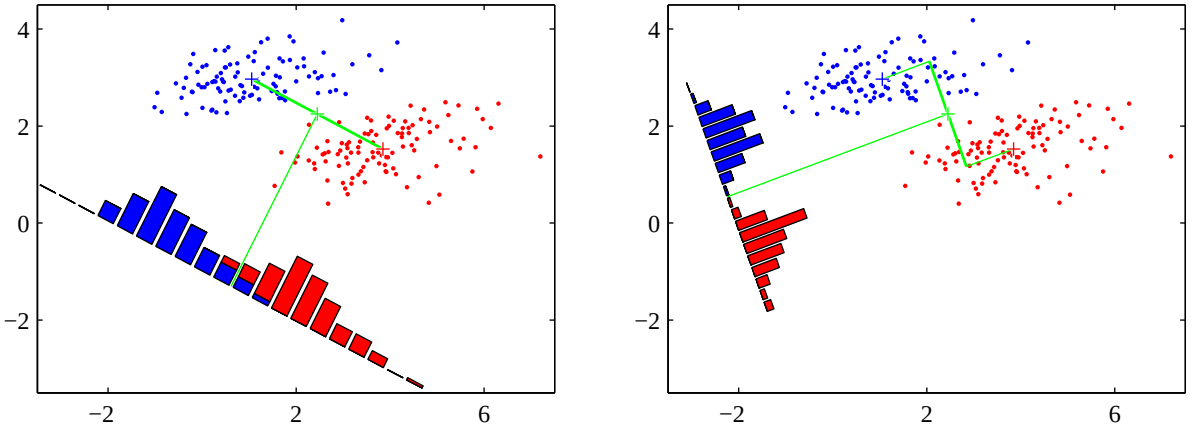
$$\boldsymbol{m}_1 = \frac{1}{N_1} \sum_{n \in C_1} \boldsymbol{x}^n \quad \boldsymbol{m}_2 = \frac{1}{N_2} \sum_{n \in C_2} \boldsymbol{x}^n$$



We can choose $\boldsymbol{w}$ to maximize $\boldsymbol{w}^T(\boldsymbol{m}_1 - \boldsymbol{m}_2)$, subject to $\sum_i w_i^2 = 1$ yields $\boldsymbol{w} \propto (\boldsymbol{m}_1 - \boldsymbol{m}_2)$. (exercise 4.4).

Fisher's linear discriminant

Fisher discriminant analysis: Better separation when simultaneously minimize within class variance.



Within class variance after transformation:

$$s_k^2 = \sum_{n \in C_k} (y^n - m_k)^2, \qquad m_k = \boldsymbol{w}^T \boldsymbol{m}_k, \qquad y^n = \boldsymbol{w}^T \boldsymbol{x}^n \quad k = 1, 2$$

Total within class variance is $s_1^2 + s_2^2$.

The Fisher method minimizes witin class variance and maximizes mean separation by maximizing

$$\begin{aligned} J(\boldsymbol{w}) &= \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\boldsymbol{w}^T S_B \boldsymbol{w}}{\boldsymbol{w}^T S_W \boldsymbol{w}} \\ S_B &= (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T \\ S_W &= \sum_{n \in \mathcal{C}_1} (\boldsymbol{x}^n - \boldsymbol{m}_1)(\boldsymbol{x}^n - \boldsymbol{m}_1)^T + \sum_{n \in \mathcal{C}_2} (\boldsymbol{x}^n - \boldsymbol{m}_2)(\boldsymbol{x}^n - \boldsymbol{m}_2)^T \end{aligned}$$

(Exercise 4.5)

Differentiation wrt $\boldsymbol{w}$ yields (exercise)

$$(\boldsymbol{w}^T S_B \boldsymbol{w}) S_W \boldsymbol{w} = (\boldsymbol{w}^T S_W \boldsymbol{w}) S_B \boldsymbol{w}$$

Drop scalar factors, and $S_B \boldsymbol{w} \propto \boldsymbol{m}_2 - \boldsymbol{m}_1$:

$$\boldsymbol{w} \propto S_W^{-1} (\boldsymbol{m}_2 - \boldsymbol{m}_1)$$

Fisher's linear discriminant

When $S_W \propto$ the unit matrix (isotropic within-class covariance), we recover the naive solution $\boldsymbol{w} \propto (\boldsymbol{m}_2 - \boldsymbol{m}_1)$.

A classifier is built by thresholding $y = \boldsymbol{w}^T \boldsymbol{x}$.

The Fisher discriminant method is best viewed as a dimension reduction method, in this case from d dimensions to 1 dimension, such that optimal classification is obtained in the linear sense.

It can be shown that the least square method with two classes and target values $t_1^n = N/N_1$ and $t_2^n = -N/N_2$ is equivalent to the two class Fisher discriminant solution (section 4.1.5).

Several classes

Class means $\boldsymbol{m}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \boldsymbol{x}^n$ and covariance $\boldsymbol{S}_k = \sum_{n \in \mathcal{C}_k} (\boldsymbol{x}^n - \boldsymbol{m}_k)(\boldsymbol{x}^n - \boldsymbol{m}_k)^T$.
Relate total within class covariance $\boldsymbol{S}_W = \sum_k \boldsymbol{S}_k$ to total covariance:

$$
\begin{aligned}
\boldsymbol{S}_W &= \sum_k \sum_{n \in C_k} (\boldsymbol{x}^n - \boldsymbol{m}_k)(\boldsymbol{x}^n - \boldsymbol{m}_k)^T \\
&= \sum_k \sum_{n \in C_k} (\boldsymbol{x}^n - \boldsymbol{m} + \boldsymbol{m} - \boldsymbol{m}_k)(\boldsymbol{x}^n - \boldsymbol{m} + \boldsymbol{m} - \boldsymbol{m}_k)^T \\
&= \sum_k \sum_{n \in C_k} (\boldsymbol{x}^n - \boldsymbol{m})(\boldsymbol{x} - \boldsymbol{m})^T + (\boldsymbol{x}^n - \boldsymbol{m})(\boldsymbol{m} - \boldsymbol{m}_k)^T + (\boldsymbol{m} - \boldsymbol{m}_k)(\boldsymbol{x}^n - \boldsymbol{m})^T \\
&+ (\boldsymbol{m} - \boldsymbol{m}_k)(\boldsymbol{m} - \boldsymbol{m}_k)^T \\
&= \boldsymbol{S}_T - \sum_k N_k (\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{m}_k - \boldsymbol{m})^T = \boldsymbol{S}_T - \boldsymbol{S}_B
\end{aligned}
$$

with total covariance $\boldsymbol{S}_T = \sum_n (\boldsymbol{x}^n - \boldsymbol{m})(\boldsymbol{x}^n - \boldsymbol{m})^T$ and $\boldsymbol{S}_B = \sum_{k=1}^K N_k (\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{m}_k - \boldsymbol{m})^T$.

Assume input dimension $D > K$, the number of classes.

We introduce $D' > 1$ features $y_{d'} = \boldsymbol{w}_{d'}^T \boldsymbol{x}, d' = 1, \ldots D'$ or

$$\boldsymbol{y} = \boldsymbol{W}^T \boldsymbol{x}, \quad \boldsymbol{W} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{D'})$$

$\boldsymbol{W}^T \boldsymbol{S}_W \boldsymbol{W}$ and $\boldsymbol{W}^T \boldsymbol{S}_B \boldsymbol{W}$ are the D'-dimensional projection of the within class and between class covariance matrices. Maximize

$$J(\boldsymbol{W}) = \text{Tr}\left((\boldsymbol{W}^T \boldsymbol{S}_W \boldsymbol{W})^{-1} (\boldsymbol{W}^T \boldsymbol{S}_B \boldsymbol{W})\right)$$

The solution $\boldsymbol{W} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{D'})$ consists of the largest D' eigenvectors of the matrix $\boldsymbol{S}_W^{-1} \boldsymbol{S}_B$.

Note, $\boldsymbol{S}_B$ is sum of K rank one matrices and is of rank $K - 1$. $\boldsymbol{S}_W^{-1} \boldsymbol{S}_B$ has $(K - 1)$ non-zero eigenvalues.

Thus, $D' \leq K - 1$, because otherwise $\boldsymbol{W}$ contains any of the zero eigenvectors.

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

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).

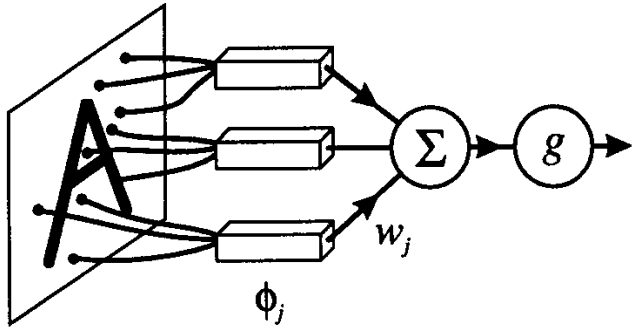
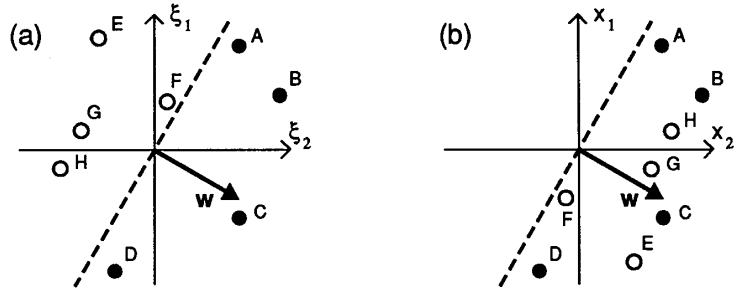


Figure 3.10. The perceptron network used a fixed set of processing elements, denoted ϕ_j, followed by a layer of adaptive weights w_j and a threshold activation function $g(\cdot)$. The processing elements ϕ_j typically also had threshold activation functions, and took inputs from a randomly chosen subset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

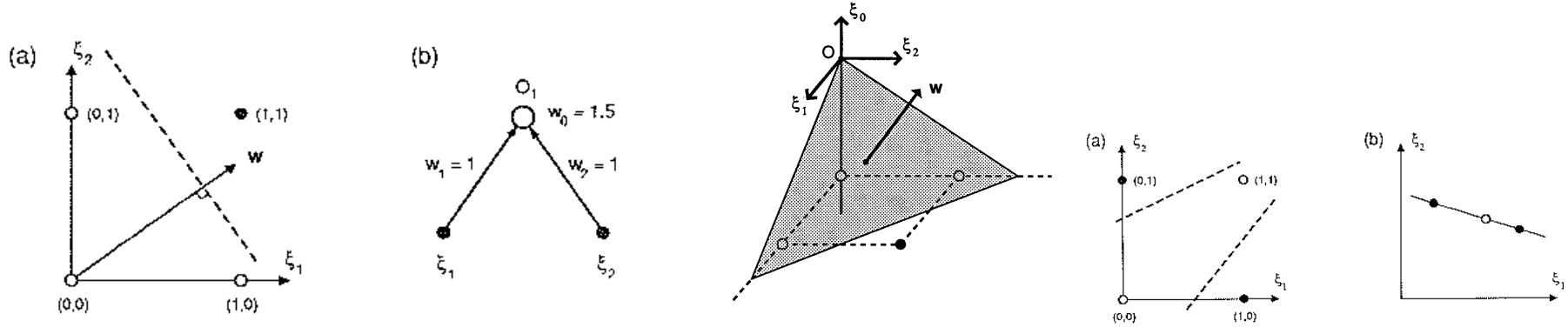
Linear separation

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

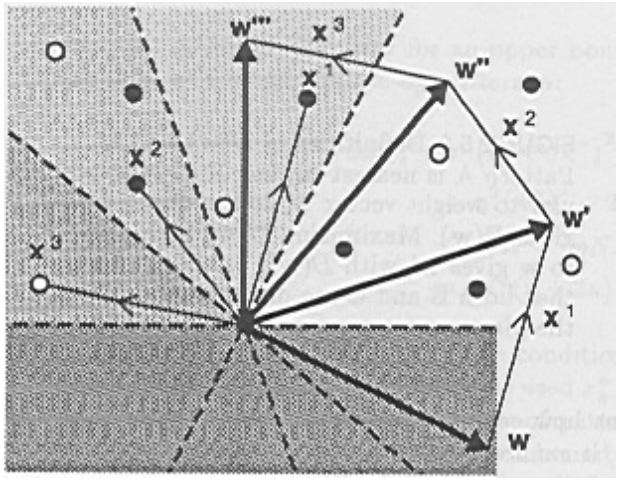
Learning succesful when

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

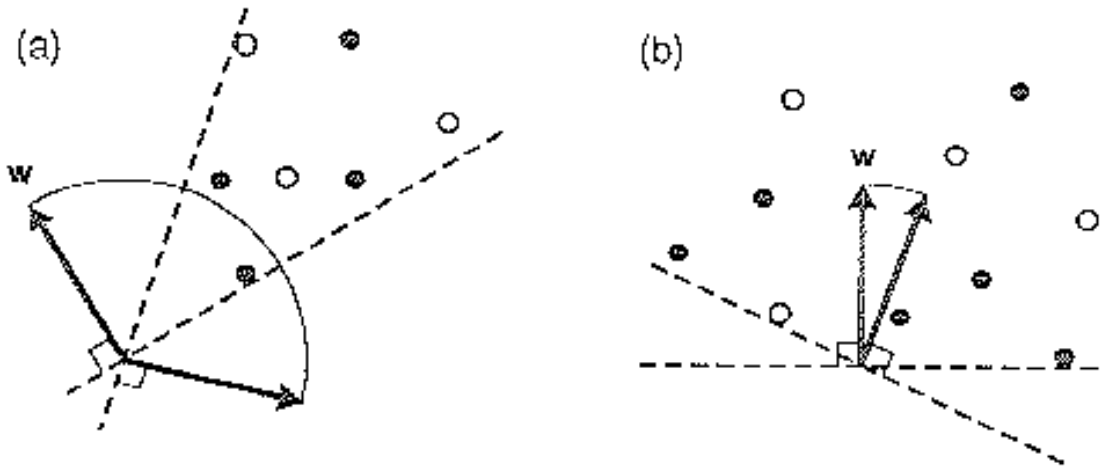
Learning rule is 'Hebbian':

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

η is the learning rate.



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

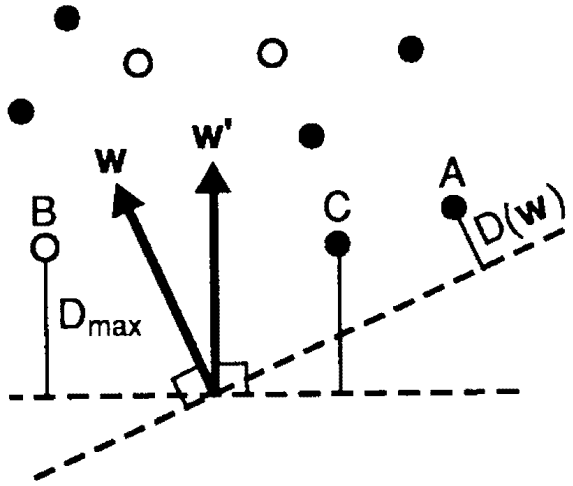


The quality of the solution is determined by the worst pattern. Since the solution does not depend on the size of w:

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

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,

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

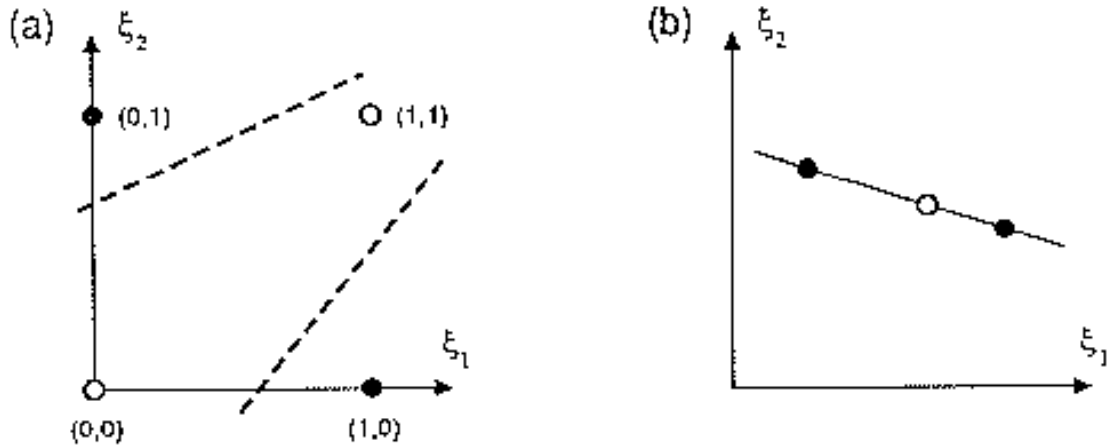
Number of weight updates:

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

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



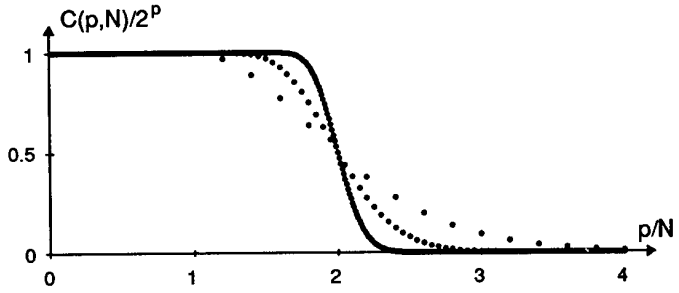
Question: What is the probability that a problem of P samples in N dimensions is linearly separable?

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):

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

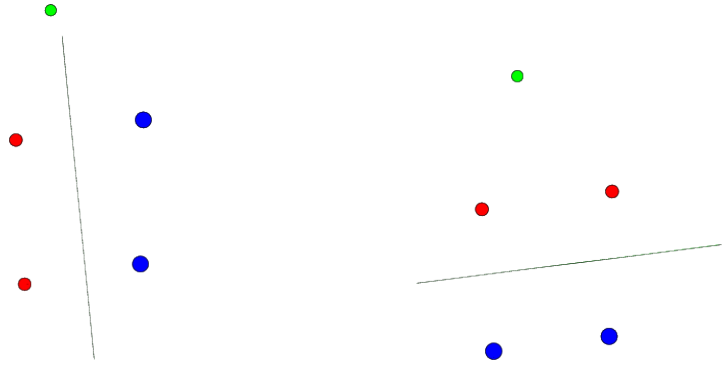
[9] When $P \leq N$, then $C(P, N) = 2 \sum_{i=0}^{P-1} \left(\begin{array}{c} P-1 \\ i \end{array} \right) = 2(1+1)^{P-1} = 2^P$

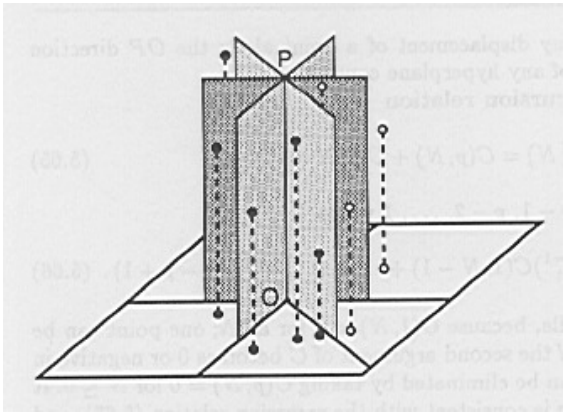
When $P = 2N$, then $C(P, N) = 2 \sum_{i=0}^{N-1} \left(\begin{array}{c} 2N-1 \\ i \end{array} \right) = \sum_{i=0}^{2N-1} \left(\begin{array}{c} 2N-1 \\ i \end{array} \right) = 2^{2N-1} = 2^{P-1}$



[9] Conventions: $\left(\begin{array}{c} n \\ 0 \end{array} \right) = 1$; $\left(\begin{array}{c} n \\ k \end{array} \right) = 0$ when $n < k$.

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

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

The Perceptron algorithm

Perceptron approach: seek $\boldsymbol{w}$ such that $\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n) > 0$ for $\boldsymbol{x}_n \in C_1$ and $\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n) < 0$ for $\boldsymbol{x}_n \in C_2$. Target coding: $t = +1, -1$ for $x \in \{C_1, C_2\}$. Then we want all patterns:

$$t_n \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_n) > 0$$

Perceptron criterion: minimize error from misclassified patterns

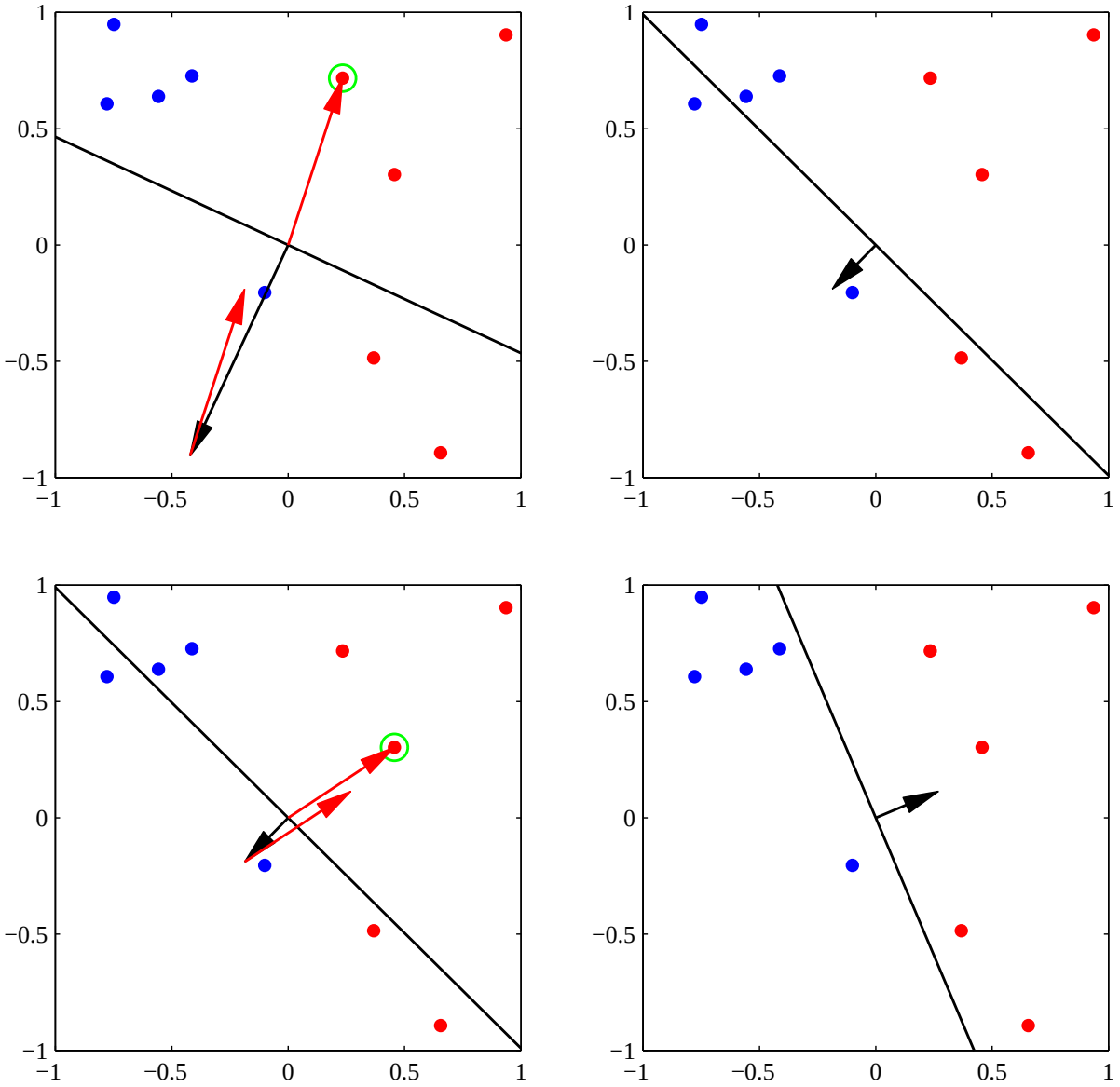
$$C(w) = -\frac{1}{\|w\|} \sum_{n \in M} t_n \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_n)$$

Learning rule: choose pattern n. If misclassified, update according to

$$\boldsymbol{w}^{\tau+1} = \boldsymbol{w}^{\tau} + \eta \boldsymbol{\phi}_n t_n$$

It can be shown that this algorithm converges in a *finite number of steps*, *if the data is linearly separable.*

Convergence of perceptron learning



Perceptron critique

Minsky and Papert: perceptron can only learn linearly separable problems:

Most functions are not linearly separable: e.g. the problem of determining whether objects are singly connected, using local receptive fields

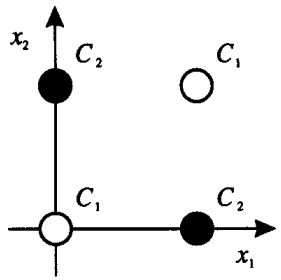


Figure 3.6. The exclusive-OR problem consists of four patterns in a two-dimensional space as shown. It provides a simple example of a problem which is not linearly separable.

Probabilistic generative models

Probabilistic models for classification

$$\begin{aligned} p(C_1|\boldsymbol{x}) &= \frac{p(\boldsymbol{x}|C_1)p(C_1)}{p(\boldsymbol{x}|C_1)p(C_1) + p(\boldsymbol{x}|C_2)p(C_2)} \\ &= \frac{1}{1+\exp(-a)} = \sigma(a) \end{aligned}$$

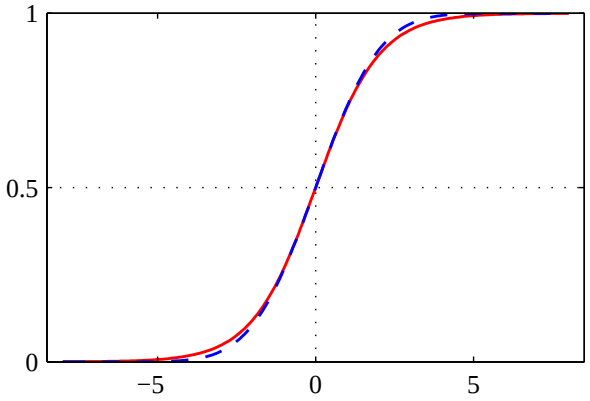
where a is the "log odds"

$$a = \ln \frac{p(\boldsymbol{x}|C_1)p(C_1)}{p(\boldsymbol{x}|C_2)p(C_2)}$$

and σ the logistic sigmoid function (i.e. S-shaped function)

$$\sigma(a) = \frac{1}{1+\exp(-a)}$$

Sigmoid function



Properties:

$$\sigma(-a) = 1 - \sigma(a)$$

inverse:

$$a = \ln \frac{\sigma}{1-\sigma}$$

10

10

$$1 + e^{-a} = 1 + \frac{1-\sigma}{\sigma} = \frac{1}{\sigma} \qquad \rightarrow \qquad \sigma = \frac{1}{1+e^{-a}}$$

Sigmoid/softmax function

Softmax: generalization to K classes

$$p(C_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|C_k)p(C_k)}{\sum_j p(\boldsymbol{x}|C_j)p(C_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

where

$$a_k = \ln\left(p(\boldsymbol{x}|C_k)p(C_k)\right)$$

Note that softmax is invariant under $a_k \to a_k + \text{const}$

Continuous inputs

The discriminant function for a Gaussian distribution is

$$\begin{aligned} a_k(\boldsymbol{x}) &= \log p(\boldsymbol{x}|C_k) + \log p(C_k) \\ &= -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\Sigma_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) - \frac{1}{2}\log|\Sigma_k| + \log p(C_k) \end{aligned}$$

The decision boundaries $a_k(\boldsymbol{x}) = a_l(\boldsymbol{x})$ are quadratic functions in d dimensions.

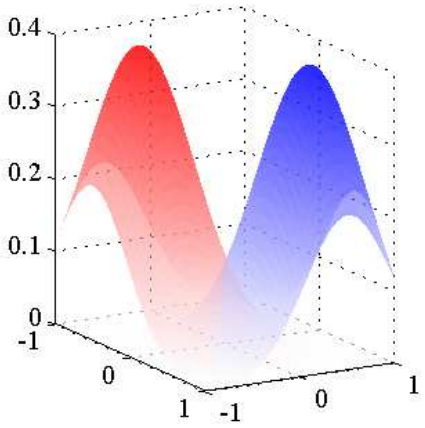
If $\Sigma_k^{-1} = \Sigma^{-1}$, then

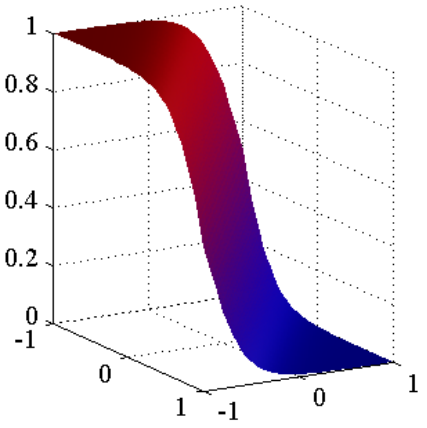
$$\begin{aligned} a_k(\boldsymbol{x}) &= -\frac{1}{2}\boldsymbol{x}\Sigma^{-1}\boldsymbol{x} + \frac{1}{2}\boldsymbol{\mu}_k^T\Sigma^{-1}\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}\Sigma^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^T\Sigma^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\log|\Sigma| + \log p(C_k) \\ &= \boldsymbol{w}_k^T\boldsymbol{x} + w_{k0} + \text{const} \end{aligned}$$

with

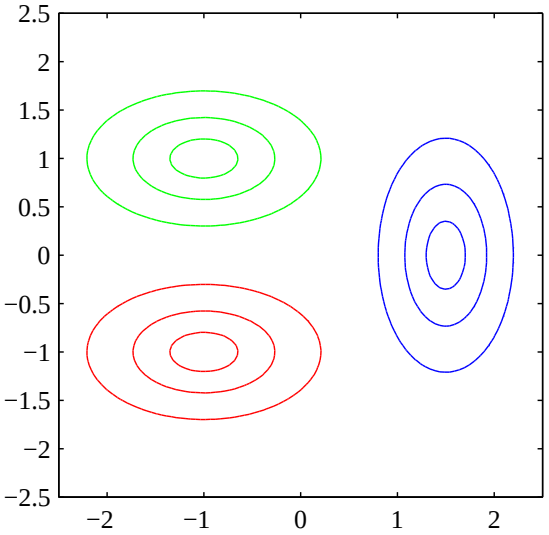
$$\boldsymbol{w}_k^T = \boldsymbol{\mu}_k^T\Sigma^{-1}, \quad w_{k0} = -\frac{1}{2}\boldsymbol{\mu}_k^T\Sigma^{-1}\boldsymbol{\mu}_k + \log p(C_k)$$

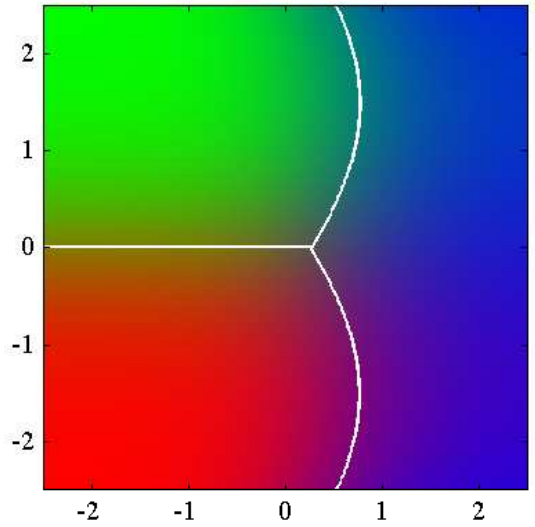
The discriminant function is linear, and the decision boundary is a straight line (or hyper-plane in d dimensions).





When class conditional covariances are equal, decision boundary is straight line.





When class conditional covariances are unequal decision boundary is quadratic.

Maximum likelihood solution

Once we have specified a parametric functional form for $p(\boldsymbol{x}|C_k)$ and $p(C_k)$ we can determine parameters using maximum (joint!) likelihood (as usual !).

Actually, for classification, we should aim for the *conditional* likelihood, but the joint likelihood is in general easier. In a perfect model, it makes no difference, but with imperfect models, it certainly does!

Example of joint ML: a binary problem with Gaussian class-conditionals, with a shared covariance matrix. Data is $\{\boldsymbol{x}_n, t_n\}_n$ with labels coded as $t = \{1, 0\}$ for classes C_1, C_2.

Class probabilities are parametrized as

$$p(C_1) = \pi, \quad p(C_2) = 1 - \pi$$

and class-conditional densities as

$$p(\boldsymbol{x}_n|C_1) = \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}), \quad p(\boldsymbol{x}_n|C_2) = \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$$

Then the likelihood is given by

$$L = p(t_1, \boldsymbol{x}_1, \ldots, t_N, \boldsymbol{x}_n | \pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_n p(t_n, \boldsymbol{x}_n | \pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma})$$

$$= \prod_n [\pi \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_1, \boldsymbol{\Sigma})]^{t_n} [(1-\pi)\mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_2, \boldsymbol{\Sigma})]^{1-t_n}$$

$$\log L = \sum_n t_n \log[\pi \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_1, \boldsymbol{\Sigma})] + (1-t_n) \log[(1-\pi)\mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_2, \boldsymbol{\Sigma})]$$

The maximum likelihood solution for π and $\boldsymbol{\mu}_{1,2}$:

$$\frac{\partial \log L}{\partial \pi} = \frac{1}{\pi} \sum_n t_n - \frac{1}{1-\pi} \sum_n (1-t_n) \qquad \pi = \frac{N_1}{N} \qquad N_1 = \sum_n t_n$$

$$\frac{\partial \log L}{\partial \boldsymbol{\mu}_1} = \sum_n t_n \frac{\partial \log \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_1, \boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}_1} = \boldsymbol{\Sigma}^{-1} \sum_n t_n (\boldsymbol{x}_n - \boldsymbol{\mu}_1) \qquad \boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_n t_n x_n$$

$$\frac{\partial \log L}{\partial \boldsymbol{\mu}_2} = \ldots \qquad \boldsymbol{\mu}_2 = \frac{1}{N_2} \sum_n (1-t_n) x_n$$

The maximum likelihood solution for $\boldsymbol{\Sigma}$ given by Eqs. 4.78-80 (exercise).

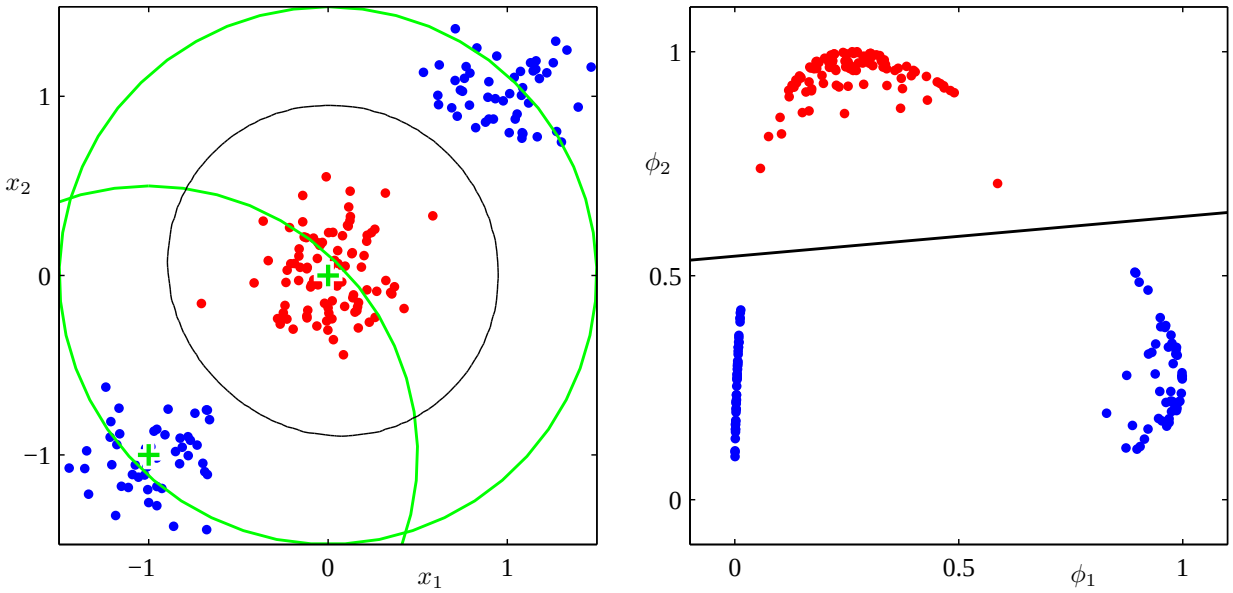
Probabilistic discriminative models

Discriminative versus generative models

Discriminative models:

- no spoiled effort in modeling joint probabilities. Aims directly at the conditional probabilities of interest
- usually fewer parameters
- improved performance when class-conditional $p(\boldsymbol{x}|C)$ assumptions are poor (with joint ML).
- basis functions can be employed

Fixed basis functions



- Basis functions $\boldsymbol{\phi}(\boldsymbol{x})$. Denote $\boldsymbol{\phi}_n = \boldsymbol{\phi}(\boldsymbol{x}_n)$
- Problems that are not linearly separable in $\boldsymbol{x}$ might be linearly separable in $\boldsymbol{\phi}(\boldsymbol{x})$.
- Note: use of basis functions $\boldsymbol{\phi}$ in place of variables $\boldsymbol{x}$ is not obvious in generative models: $\mathcal{N}(x, x^2, x^3|\boldsymbol{\mu}, \Sigma)$??.

Logistic regression

Two class classification

$$p(C_1|\boldsymbol{\phi}) = \sigma(\boldsymbol{w}^T\boldsymbol{\phi})$$

M dimensional feature space: M parameters, while Gaussian class conditional densities would require $2M$ (two means) and $M(M+1)/2$ (common covariance matrix) parameters.

Maximum likelihood to determine parameters $\boldsymbol{w}$, (nb. $t_n \in \{0, 1\}$)

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

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

$$\nabla E(\boldsymbol{w}) = \sum_n (y_n - t_n)\boldsymbol{\phi}_n$$

[11] No closed form solution. Optimization by gradient descent, (or e.g., Newton-Raphson).

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

[11] Note, that when $y = \sigma(x)$ then

$$\frac{dy}{dx} = \sigma'(x) = \frac{d}{dx}\frac{1}{1+\exp(-x)} = \frac{\exp(-x)}{(1+\exp(-x))^2} = \sigma(x)\sigma(-x) = \sigma(x)(1-\sigma(x))$$

Thus with $y = \sigma(w^T\phi)$

$$\frac{\partial y}{\partial w_i} = \phi_i \sigma'(w^T\phi) = \phi_i y(1-y)$$

Iterative least squares

Minimize learning error by Newton-Raphson method

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

with

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

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).

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

Laplace approximation

Assume distribution $p(z)$ is given up to normalization, i.e. in the form

$$p(z) = \frac{1}{Z} f(z) \qquad Z = \int f(z) dz$$

where $f(z)$ is given, but Z is unknown (and the integral is infeasible).

Goal: approximate by a Gaussian $q(z)$, centered around the mode of $p(z)$.

Laplace approximation

Mode z_0 is maximum of $p(z)$, i.e. $dp(z)/dz|_{z_0} = 0$, or

$$\frac{df(z)}{dz}\Big|_{z_0} = 0$$

The logarithm of a Gaussian is a quadratic function of the variables, so it makes sense to make a second order Taylor expansion of $\ln f$ around the mode (this would be exact if p was Gaussian).

$$\ln f(z) \approx \ln f(z_0) - \frac{1}{2}A(z - z_0)^2$$

where

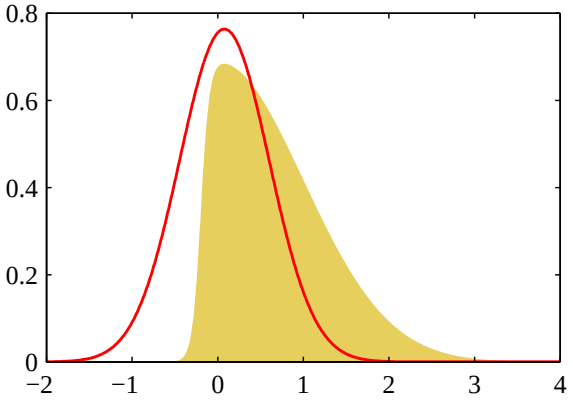
$$A = -\frac{d^2 \ln f(z)}{dz^2}\Big|_{z_0}$$

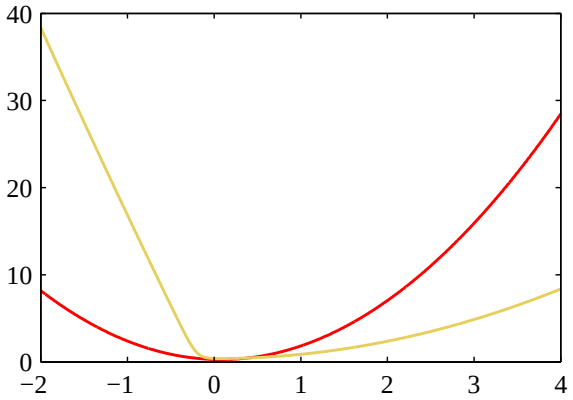
Note that the first order term is absent since we expand around the maximum. Taking the exponent we obtain,

$$f(z) \approx f(z_0) \exp(-\frac{1}{2}A(z - z_0)^2)$$

and the Gaussian approximation is obtained by normalization

$$q(z) = \left(\frac{A}{2\pi}\right)^{1/2} \exp(-\frac{1}{2}A(z - z_0)^2)$$





Laplace approximation in M dimensions

In M dimensions results are similar,

$$q(\boldsymbol{z}) = \left(\frac{\det(\boldsymbol{A})}{(2\pi)^{M/2}} \right) \exp(-\frac{1}{2}(\boldsymbol{z} - \boldsymbol{z}_0)^T \boldsymbol{A}(\boldsymbol{z} - \boldsymbol{z}_0))$$

where

$$A_{ij} = -\frac{\partial^2}{\partial z_i \partial z_j} \ln f(\boldsymbol{z})\big|_{\boldsymbol{z}=\boldsymbol{z}_0}$$

Usually $\boldsymbol{z}_0$ is found by numerical optimization.

A weakness of Laplace approximation is that it relies only on the local properties of the mode. Other methods (e.g. sampling or variational methods) are based on more global properties of p.

Model comparison and BIC

Approximation of Z:

$$
\begin{aligned}
Z &= \int f(\boldsymbol{z}) d\boldsymbol{z} && (4)\\
&\approx f(\boldsymbol{z}_0) \int \exp(-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{z}_0)^T \boldsymbol{A}(\boldsymbol{z}-\boldsymbol{z}_0)) d\boldsymbol{z} && (5)\\
&= f(\boldsymbol{z}_0) \frac{(2\pi)^{M/2}}{\det(\boldsymbol{A})^{1/2}} && (6)
\end{aligned}
$$

Model comparison and BIC

Application: approximation of model evidence.

$$f(\boldsymbol{\theta}) = p(D|\boldsymbol{\theta})p(\boldsymbol{\theta}) \tag{7}$$

$$Z = P(D) = \int p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta} \tag{8}$$

Then applying Laplace approximation, we obtain (Ex. 4. 22)

$$\ln P(D) \approx \ln p(D|\boldsymbol{\theta}_{\mathrm{MAP}}) + \underbrace{\ln p(\boldsymbol{\theta}_{\mathrm{MAP}}) + \frac{M}{2}\ln 2\pi - \frac{1}{2}\ln\det(\boldsymbol{A})}_{\text{Occam factor}}$$

where

$$\boldsymbol{A} = -\nabla\nabla \ln p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})\Big|_{\boldsymbol{\theta}_{\mathrm{MAP}}} = -\nabla\nabla p(\boldsymbol{\theta}|D)\Big|_{\boldsymbol{\theta}_{\mathrm{MAP}}}$$

which can be interpreted as the inverse width of the posterior.

Model comparison and BIC

Under some assumptions, $\boldsymbol{A} = \sum_n \boldsymbol{A}_n \approx N\hat{\boldsymbol{A}}$, and full rank, then

$$\ln\det(\boldsymbol{A}) \approx \ln\det(N\hat{\boldsymbol{A}}) = \ln(N^M \det(\hat{\boldsymbol{A}})) = M\ln N + \mathcal{O}(1)$$

leads to the Bayesian Information Criterion (BIC)

$$\ln P(D) \approx \ln p(D|\theta_{\mathrm{MAP}}) - \frac{1}{2}M\ln N \quad (+const.)$$

Bayesian logistic regression

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

Log posterior = log prior + log likelihood + const

$$\begin{aligned}\ln p(\boldsymbol{w}|\boldsymbol{t}) &= -\frac{1}{2}(\boldsymbol{w}-\boldsymbol{m}_0)^T\boldsymbol{S}_0^{-1}(\boldsymbol{w}-\boldsymbol{m}_0) \\ &\quad + \sum_n \{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} + const\end{aligned}$$

where $y_n = \sigma(\boldsymbol{w}^T\boldsymbol{\phi}_n)$.

Posterior distribution in $p(\boldsymbol{w}|\boldsymbol{t})$?

Laplace approximation: find $\boldsymbol{w}_{\mathrm{MAP}}$ and compute second derivatives:

$$\boldsymbol{S}_N^{-1} = -\nabla\nabla \ln p(\boldsymbol{w}|\boldsymbol{t}) = \boldsymbol{S}_0^{-1} + \sum_n y_n(1-y_n)\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T$$

and

$$q(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{w}_{\mathrm{MAP}}, \boldsymbol{S}_N)$$

Predictive distribution

$$p(C_1|\boldsymbol{\phi}, \boldsymbol{t}) = \int \sigma(\boldsymbol{w}^T\boldsymbol{\phi}) p(\boldsymbol{w}|\boldsymbol{t}) d\boldsymbol{w} \approx \int \sigma(\boldsymbol{w}^T\boldsymbol{\phi}) q(\boldsymbol{w}) d\boldsymbol{w}$$

The function $\sigma(\boldsymbol{w}^T\boldsymbol{\phi})$ depends only on $\boldsymbol{w}$ via its projection on $\boldsymbol{\phi}$. We can marginalize out the other variables, since q is a Gaussian. The marginal is then again a Gaussian for which we can compute its parameters

$$\int \sigma(\boldsymbol{w}^T\boldsymbol{\phi}) q(\boldsymbol{w}) d\boldsymbol{w} = \int \sigma(a) \mathcal{N}(a|\mu_a, \sigma_a^2) da$$

where the parameters turn out to be $\mu_a = \boldsymbol{w}_{\mathrm{MAP}}^T\boldsymbol{\phi}$ and $\sigma_a^2 = \boldsymbol{\phi}^T\boldsymbol{S}_N\boldsymbol{\phi}$.

Unfortunately, this integral cannot be expressed analytically. However $\sigma(x)$ is well approximated by the probit function, i.e., the cumulative Gaussian

$$\Phi(x) = \int_{\infty}^{x} \mathcal{N}(u|0,1) du$$

In particular $\sigma(x) \approx \Phi(\sqrt{\frac{\pi}{8}}x)$. With additional manipulations the predictive distributions

can be shown to be approximated by

$$p(C_1|\boldsymbol{\phi}, \boldsymbol{t}) \approx \sigma((1 + \pi\sigma_a^2/8)^{-1/2}\mu_a) = \sigma(\kappa(\sigma_a^2)\mu_a)$$

Note that the MAP predictive distribution is

$$p(C_1|\boldsymbol{\phi}, \boldsymbol{w}_{MAP}) = \sigma(\mu_a).$$

The decision boundary $p(C_1|\boldsymbol{\phi}, \ldots) = 0.5$ is the same in both approximations, namely at $\mu_a = 0$. Since $\kappa < 1$, the Laplace approximation is less certain about the classifications.

Chapter 5 Neural Networks

Feed-forward neural networks

Non-linear methods using a fixed set of basis functions (polynomials) suffer from curse of dimensionality.

A succesful alternative is to adapt the basis functions to the problem.
- SVMs: convex optimisation, number of SVs increases with data
- MLPs: aka feed-forward neural networks, non-convex optimisation

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:

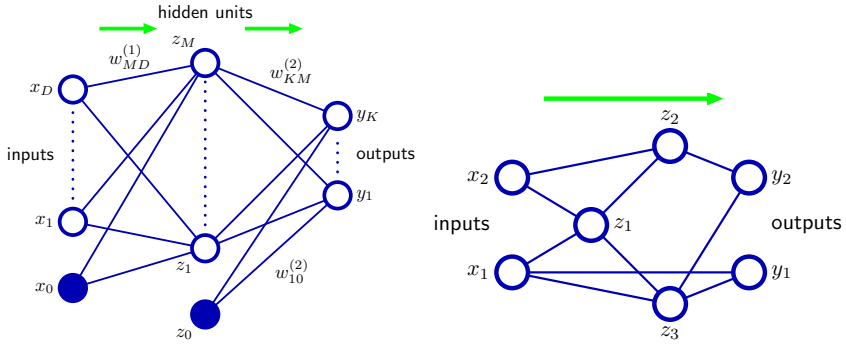
$$\phi_j(\boldsymbol{x}) = h(\sum_{i=0}^{D} w_{ji}^{(1)} x_i)$$

Feed-forward Network functions

In the case of K outputs

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

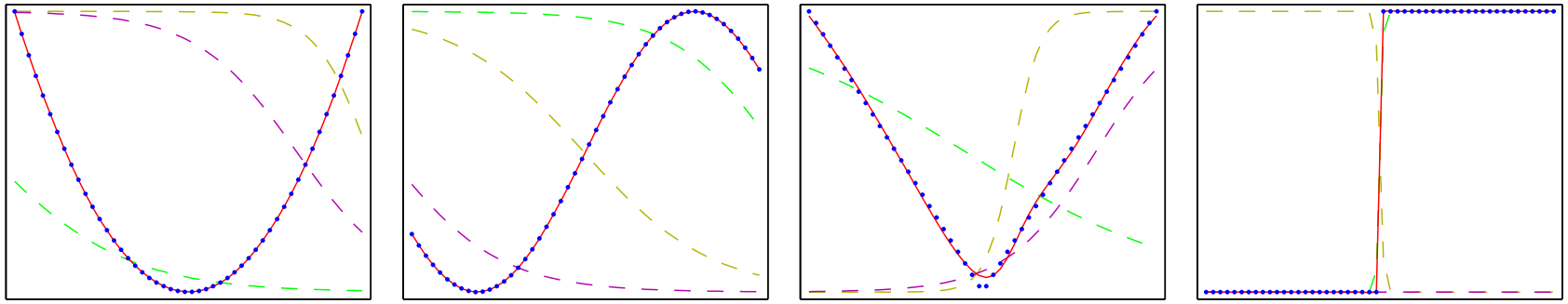
$h_2(x)$ is $\sigma(x)$ or x depending on the problem. $h_1(x)$ is $\sigma(x)$ or $\tanh(x)$.



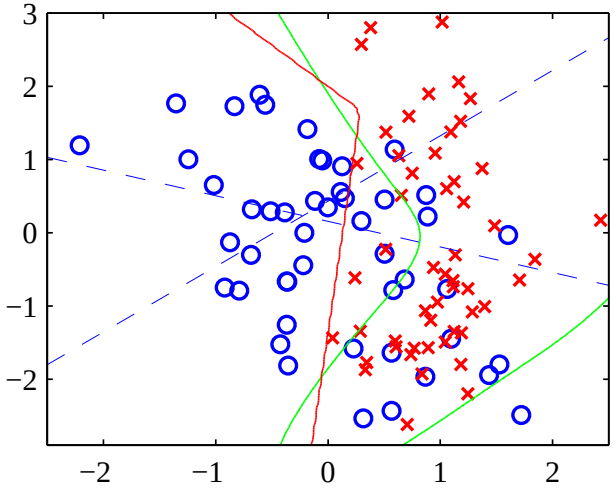
Left) Two layer architecture. Right) general feed-forward network with skip-layer connections.

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.

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.

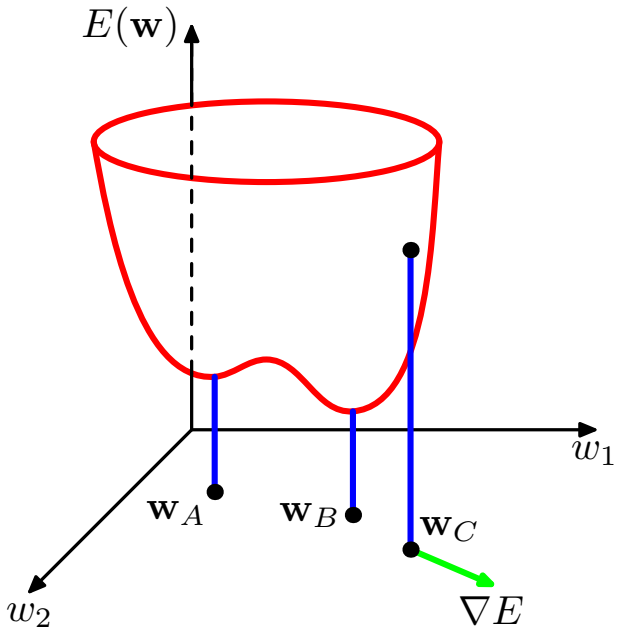
$$\begin{aligned} E(\boldsymbol{w}) &= -\sum_{n=1}^{N}\{t_n \log y(\boldsymbol{x}_n, \boldsymbol{w}) + (1 - t_n)\log(1 - y(\boldsymbol{x}_n, \boldsymbol{w}))\} \\ &= -\log\prod_{n=1}^{N} y(\boldsymbol{x}_n, \boldsymbol{w})^{t_n}(1 - y(\boldsymbol{x}_n, \boldsymbol{w}))^{1-t_n} \end{aligned}$$

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

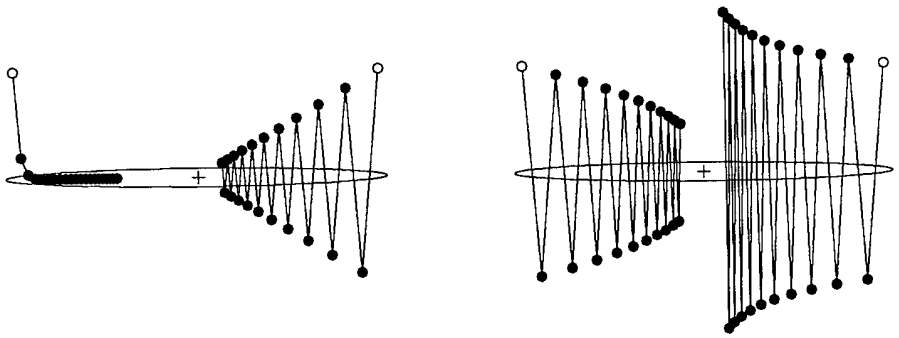
$$E(\boldsymbol{w} + \Delta \boldsymbol{w}) \approx E(\boldsymbol{w}) + (\Delta \boldsymbol{w})^T \nabla E = E(\boldsymbol{w}) - \eta \sum_i \left(\frac{\partial E}{\partial w_i}\right)^2 \leq E(\boldsymbol{w})$$

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:

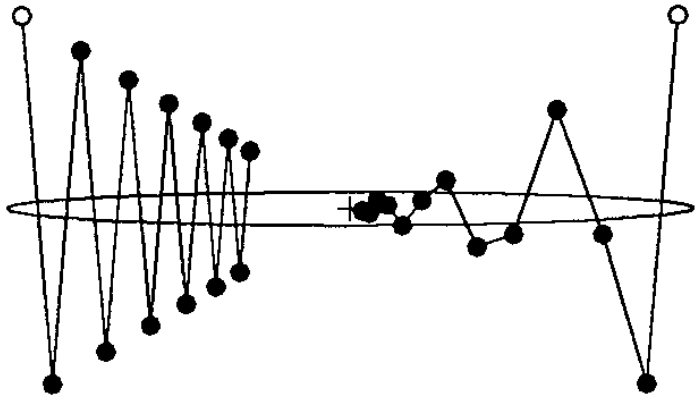
$$\Delta w_{t+1} \approx -\eta \nabla E \sum_{k=0}^{t} \alpha^k = -\frac{\eta}{1-\alpha} \frac{\partial E}{\partial w}$$

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.

Line search

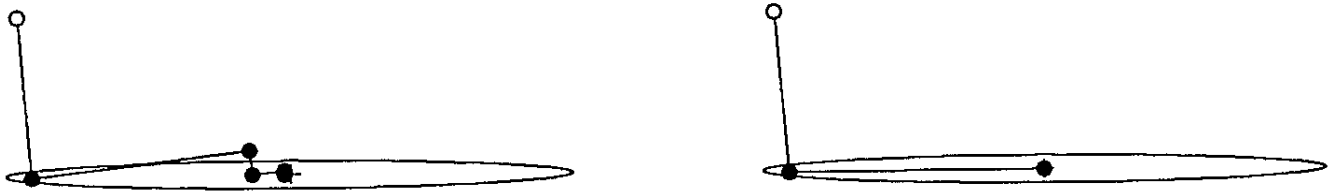
Another solution is line optimisation:

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

λ is found by a one dimensional optimisation

$$0 = \frac{\partial}{\partial \lambda} E(w_0 + \lambda d_0) = d_0 \cdot \nabla E(w_1) = d_0 \cdot d_1$$

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 $w_2 = w_1 + \lambda d_1'$ yields λ such that $d_1' \cdot \nabla E(w_2) = 0$.

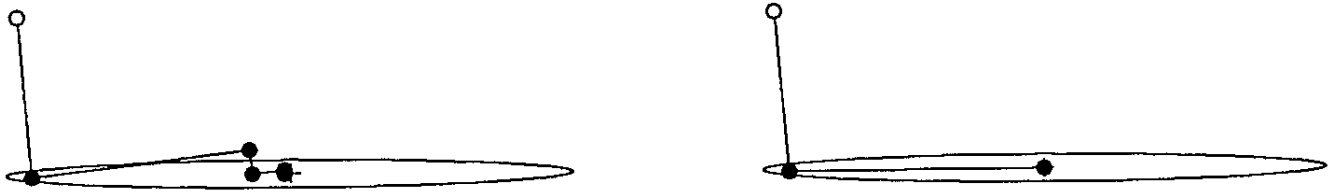
The direction d_1' is found by demanding that $\nabla E(w_2) \approx 0$ also in the 'old' direction d_0:

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

or

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

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:[12]

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

It can be proven that this rule keeps the last n directions all mutually conjugate [**?**].

[12] We need $0 = d_0^T H(w_1) d_1'$. We use $\nabla E(w_0) \approx \nabla E(w_1) + (w_0 - w_1)^T H(w_1) = \nabla E(w_1) - d_0^T H(w_1)$.

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

Method of *stochastic approximation* originally due to Robbins and Monro 1951:
- Solve $M(x) = a$ with $M(x) = \langle N(x, \xi) \rangle$.
- Iterate $x_{t+1} = x_t + \alpha_t(a - N(x, \xi))$
- Convergence requires

$$\sum_t \alpha_t = \infty \qquad \sum_t \alpha_t^2 < \infty$$

For instance $\alpha_t = 1/t$.

Application to stochastic gradient descent:
- $\nabla E(\boldsymbol{w}) = 0$ with $\nabla E(\boldsymbol{w}) = \sum_n \nabla E_n(\boldsymbol{w})$
- Iterate $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \nabla E_n(\boldsymbol{w})$

Extensions of SGD and comparisons see [**?**].

Error backpropagation

Error is sum of error per pattern

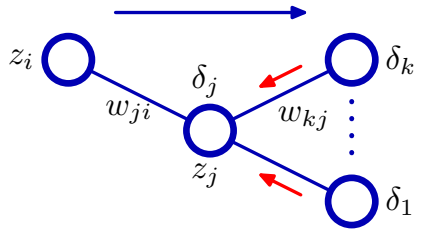
$$
\begin{aligned}
E(\boldsymbol{w}) &= \sum_n E^n(\boldsymbol{w}) \qquad E^n(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{y}(\boldsymbol{x}_n, \boldsymbol{w}) - \boldsymbol{t}_n\|^2 \\
y_k(\boldsymbol{x}, \boldsymbol{w}) &= h_2\left(w_{k0} + \sum_{j=1}^{M} w_{kj} h_1\left(w_{j0} + \sum_{i=1}^{D} w_{ji}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

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

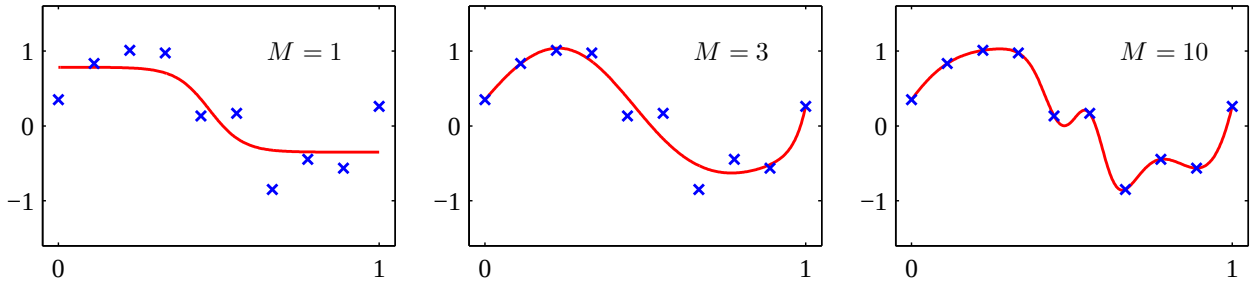
3. $\partial E^n / \partial w_{kj} = \delta_k^n z_j^n$ and $\partial E^n / \partial w_{ji} = \delta_j^n z_i^n$

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

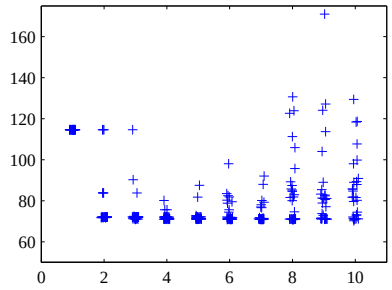
E is a function of $\mathcal{O}(|\boldsymbol{w}|)$ variables. In general, the computation of E requires $\mathcal{O}(|\boldsymbol{w}|)$ operations. The computation of ∇E would thus require $\mathcal{O}(|\boldsymbol{w}|^2)$ operations.

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
- 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

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 $j = 0, \ldots, 2^n - 1$, k labels input. Set

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

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	$w_{0k}x_k$	$w_{1k}x_k$	$w_{2k}x_k$	$w_{3k}x_k$
-1	-1	**2b**	0	0	-2b
-1	1	0	**2b**	-2b	0
1	-1	0	-2b	**2b**	0
1	1	-2b	0	0	**2b**

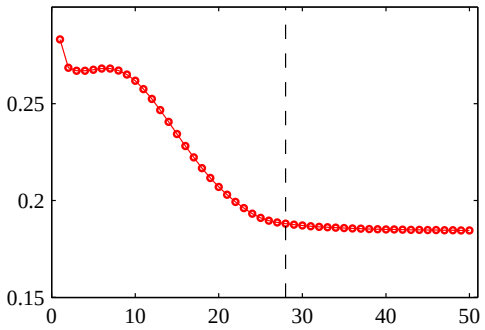
Use threshold of $(n-1)b$ at each hidden unit. The remaining problem has $p = 2^n$ patterns in 2^n dimensions and is linearly separable.

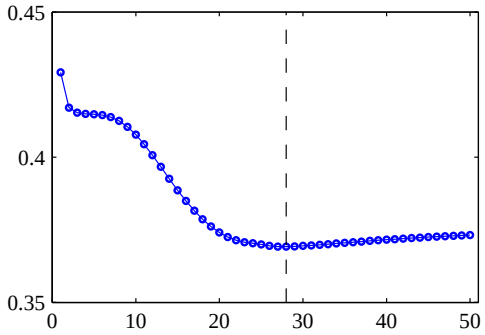
MLPs are universal approximators

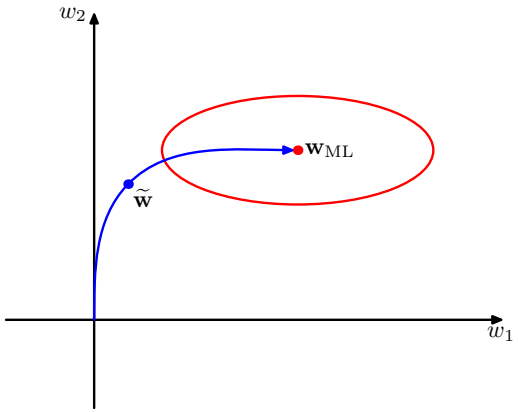
The combination of linear summation and non-linear functions can create many different functions.

- The MLP with a single hidden layer can map any continuous function [**?**, **?**]
- The MLP with multiple hidden layers may (or may not) be more efficient

5.5.2 Early stopping







Early stopping is to stop training when error on test set starts increasing.

Early stopping with small initial weigths has the effect of weight decay:

$$
\begin{aligned}
E(\boldsymbol{w}) &= \frac{1}{2}\left(\lambda_1(w_1 - w_1^*)^2 + \lambda_2(w_2 - w_2^*)^2 + \lambda(w_1^2 + w_2^2)\right) \\
\frac{\partial E}{\partial w_i} &= \lambda_i(w_i - w_i^*) + \lambda w_i = 0, \quad i = 1, 2 \\
w_i &= \frac{\lambda_i}{\lambda_i + \lambda} w_i^*
\end{aligned}
$$

When $\lambda_1 \ll \lambda \ll \lambda_2$, $w_1 \approx \lambda_1/\lambda w_1^*$ and $w_2 \approx w_2^*$.

Weights in 'flat' directions are underspecified by the data and stay small.