



PATTERN RECOGNITION AND MACHINE LEARNING

Jen-Tzung Chien

National Chiao Tung University

CONTENTS

- 1. Introduction**
- 2. Probability Distributions**
- 3. Linear Models for Regression**
- 4. Linear Models for Classification**
- 5. Kernel Methods**
- 6. Sparse Kernel Methods**
- 7. Mixture Models and EM**
- 8. Approximate Inference**

Introduction

Book web site <http://research.microsoft.com/~cmbishop/PRML>

- Pattern Recognition is concerned with **automatic** discovery of **regularities** in data through the use of computer algorithms and with the use of regularities to take actions such as classifying the data into different categories, e.g. hand-written recognition
- Training set $X = \{x_1, \dots, x_N\}$, target vector t , **supervised learning** using $\{x, t\}$
- In machine learning, we are finding $y(x)$ and decoding x as **target vectors** ⇒ Training phase/Learning phase

Testing phase → generalization (insufficient training data)

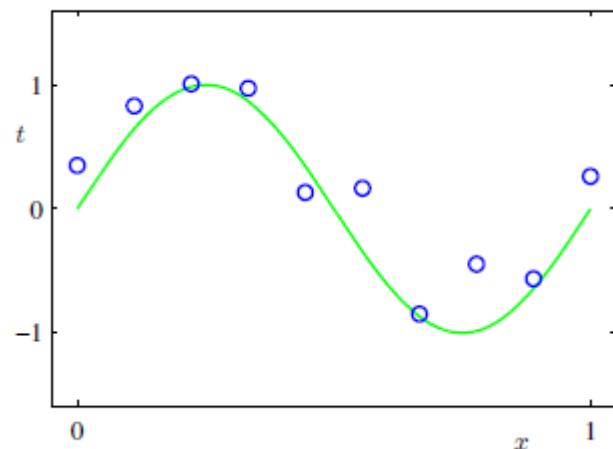
Preprocessing → feature extraction, e.g. face detection in video stream, fast to compute, preserve discriminatory information
→ dimension reduction is important

Introduction

- **Classification:** Input vector → one of discrete categories
Regression: Input vector → one or more continuous variables
- **Unsupervised** learning (without target values)
 - clustering: discovering groups of similar samples
 - density estimation: determine distribution of data within input space.
- Reinforcement learning is concerned with the problem of finding suitable actions to take in a given situation in order to maximize a reward.
- Data projection for high dimension space to 2/3 dims for visualization.

An Example of Regression Problem

Our goal is to **predict** the value of t for some new value of x , without knowledge of the curve.



- $x = (x_1, \dots, x_N)^T$
- $t = (t_1, \dots, t_N)^T$
- $\sin(2\pi x) +$ Gaussian random noise

Polynomial Curve Fitting

\hat{x} (new value) → \hat{t}

make predictions that are optimal according to appropriate criteria

polynomial function of the form

$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_Mx^M = \sum_{j=0}^M w_jx^j$$

nonlinear function of x , $\mathbf{w} = (\omega_0, \omega_1 \dots, \omega_M)^T$

linear function of \mathbf{w} (parameters) → Linear Model

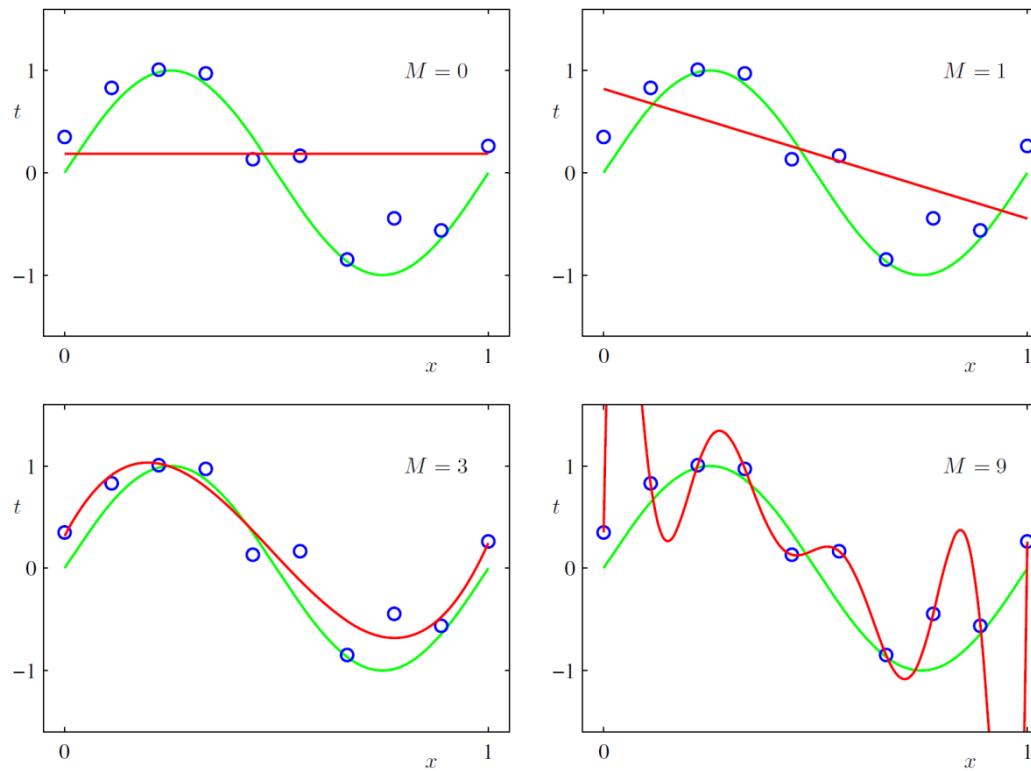
Least squares solution → sum-of-squares error function

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

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} E(\mathbf{w})$$

Selection of Order M Is Important

- Issue: Model comparison or **Model selection**

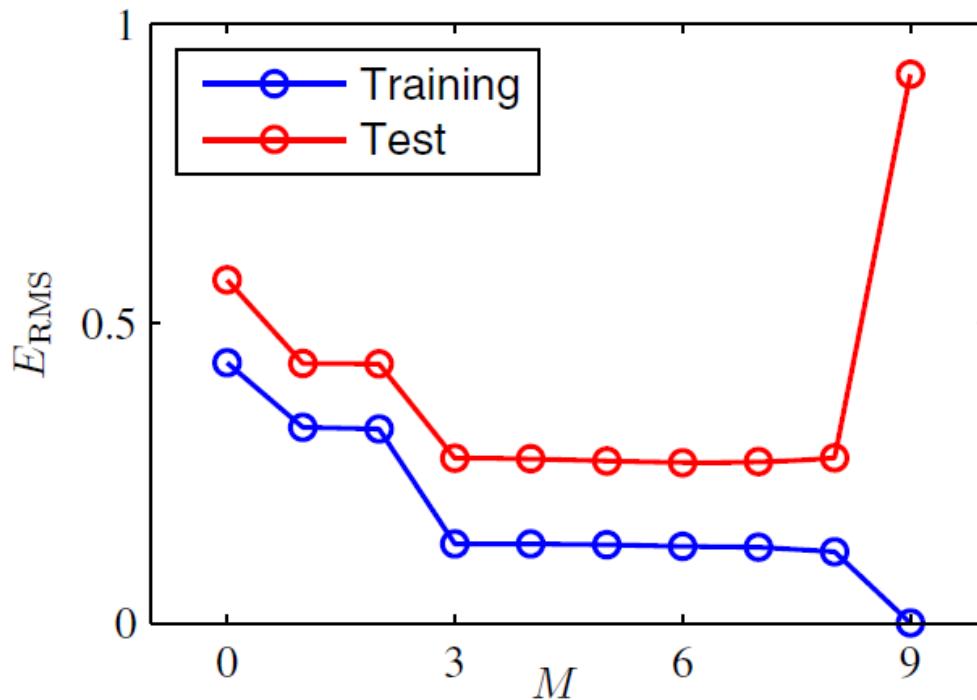


under estimation \rightarrow over-estimation
over-fitting

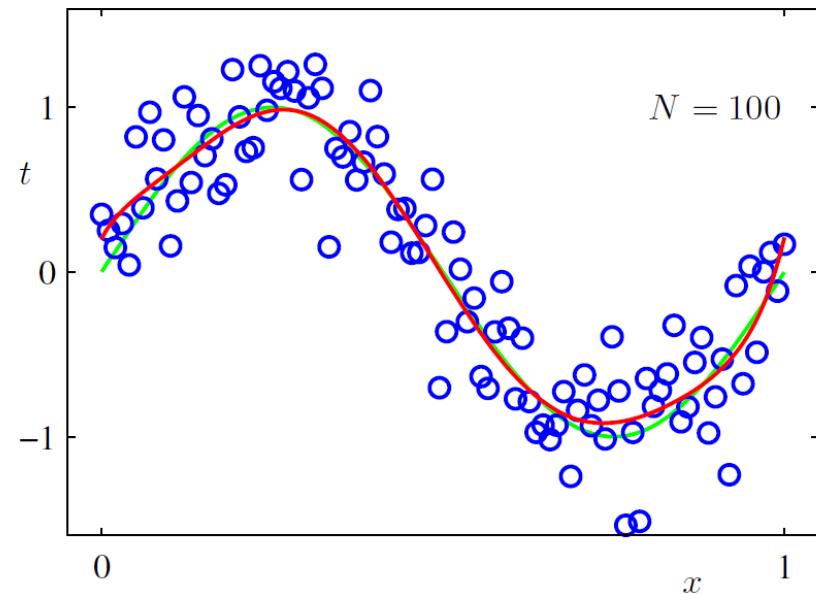
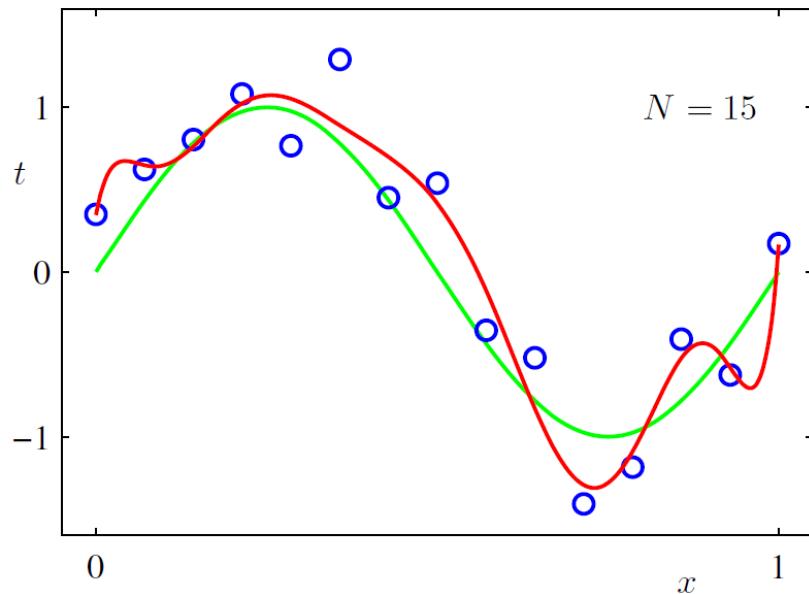
Root-Mean-Square Error vs. Model Order M

- Root-mean-square(RMS) error

$$E_{\text{RMS}} = \sqrt{2E(\mathbf{w}^*)/N}$$



Polynomial Curve Fitting under $M=9$



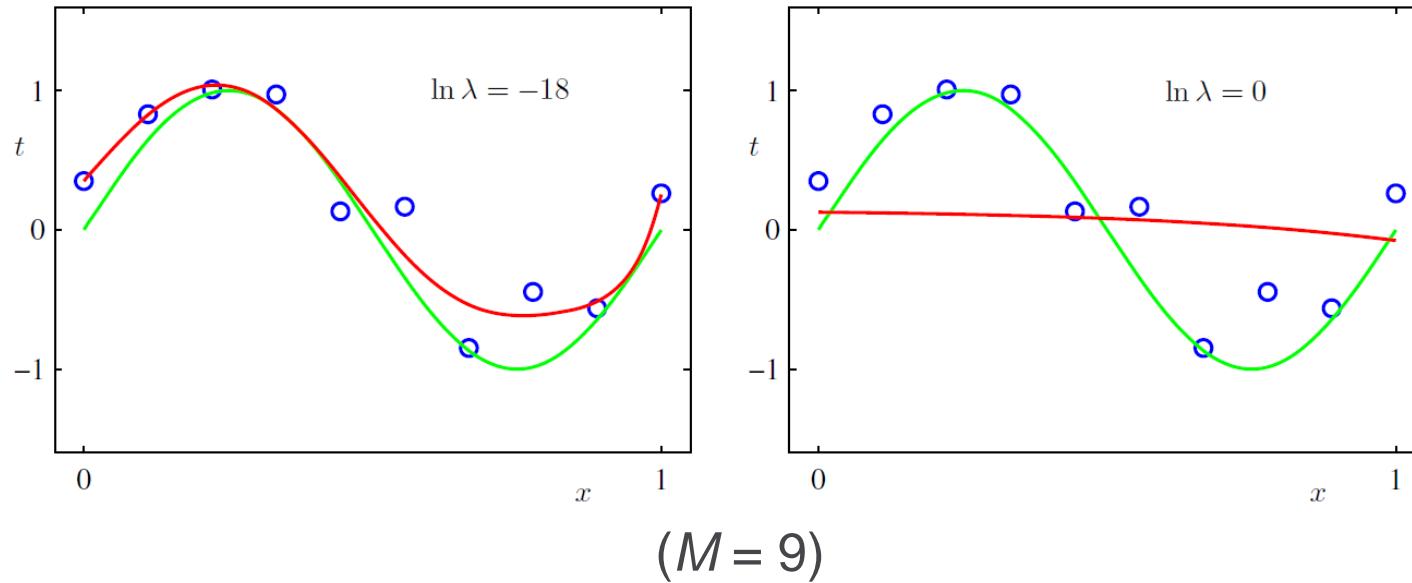
- Maximum likelihood is easy to suffer **over-fitting** problem. **Bayesian approach** is helpful for this issue.
- Using Bayesian model, the effective number of parameters adapts automatically to the size of data set.

Regularization

- One technique can be used to control the over-fitting problem that of '*regularization*'.
- A penalty term is involved in objective function → modified error parameter

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

- Weight decay approach (similar to the one used in neural network)



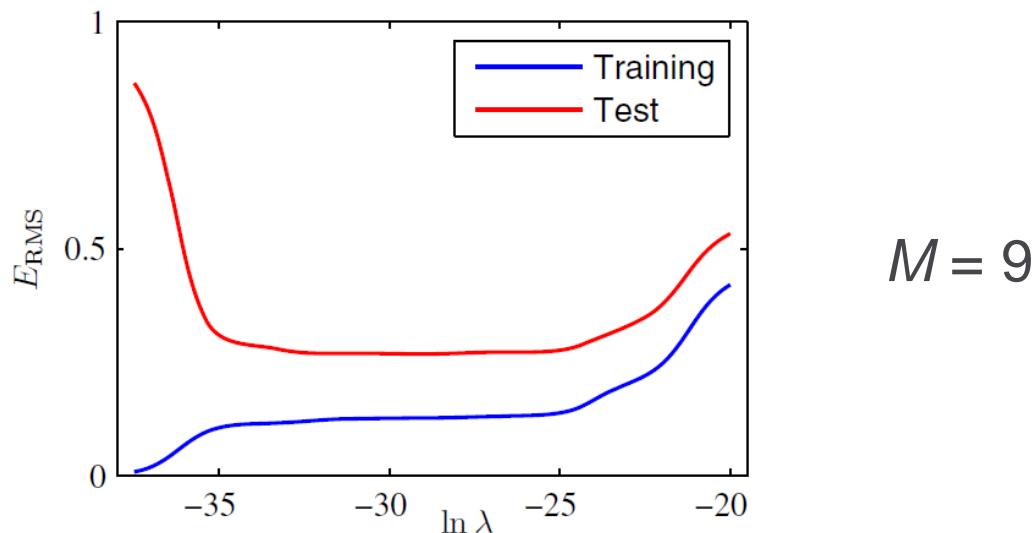
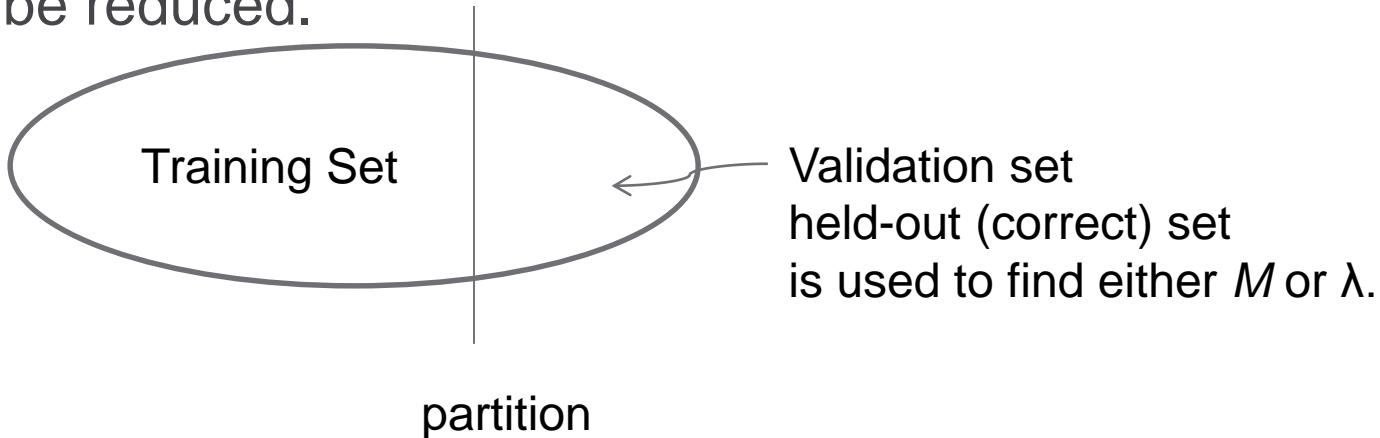
Optimal Weight Parameters

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
w_0^*	0.35	0.35	0.13
w_1^*	232.37	4.74	-0.05
w_2^*	-5321.83	-0.77	-0.06
w_3^*	48568.31	-31.97	-0.05
w_4^*	-231639.30	-3.89	-0.03
w_5^*	640042.26	55.28	-0.02
w_6^*	-1061800.52	41.32	-0.01
w_7^*	1042400.18	-45.95	-0.00
w_8^*	-557682.99	-91.53	0.00
w_9^*	125201.43	72.68	0.01

- λ controls the **complexity** of model & determine degree of **over-fitting**.

Cross Validation

- M can be reduced.



$$M = 9$$

Probability Theory

$$p(X) = \sum_Y p(X, Y)$$

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}$$

$$p(X) = \sum_Y p(X|Y)p(Y)$$

Bayes theorem

discrete random variable vs. continuous random variable

$$p(x \in (a, b)) = \int_a^b p(x) dx \quad \int_{-\infty}^{\infty} p(x) dx = 1$$

$$x = g(y)$$

$$p_y(y) = p_x(x) \left| \frac{dx}{dy} \right| = p_x(g(y)) |g'(y)|$$

c.d.f.

$$P(z) = \int_{-\infty}^z p(x) dx$$

$$\implies P'(x) = p(x) \quad x \text{ can be univariate / multivariate}$$

$$\mathbb{E}[f] = \sum_x p(x)f(x) \quad \text{discrete}$$

$$\mathbb{E}[f] = \int p(x)f(x) dx \quad \text{continuous}$$

$$\mathbb{E}_x[f(x, y)] \quad \mathbb{E}_x[f|y] = \sum_x p(x|y)f(x)$$

Probability Theory

$$\begin{aligned}\text{var}[f] &= \mathbb{E} [(f(x) - \mathbb{E}[f(x)])^2] \\ &= \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2\end{aligned}$$

$$\begin{aligned}\text{cov}[x, y] &= \mathbb{E}_{x,y} [\{x - \mathbb{E}[x]\} \{y - \mathbb{E}[y]\}] \\ &= \mathbb{E}_{x,y}[xy] - \mathbb{E}[x]\mathbb{E}[y]\end{aligned}$$

Bayesian Probabilities

- See biography of Thomas Bayes
classical interpretation of probability
- Bayesian view → quantification of uncertainty
- Probability theory is regarded as an extension of Boolean logic for situations involving **uncertainty**
- We are addressing the uncertainty of parameter \mathbf{W}

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})} \quad \mathcal{D} = \{t_1, \dots, t_N\}$$

posterior \propto likelihood \times prior

$$p(\mathcal{D}) = \int p(\mathcal{D}|\mathbf{w})p(\mathbf{w}) d\mathbf{w}$$

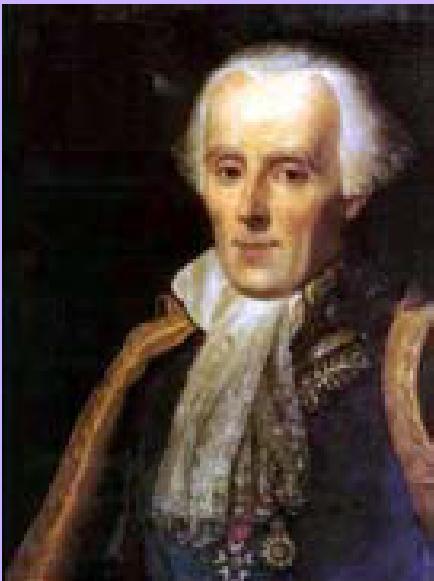
- Maximum likelihood $w_{ML} = \operatorname{argmax}_w \log P(D|w)$
 $-\log p(D|w) \rightarrow$ error function



Thomas Bayes

1701–1761

Thomas Bayes was born in Tunbridge Wells and was a clergyman as well as an amateur scientist and a mathematician. He studied logic and theology at Edinburgh University and was elected Fellow of the Royal Society in 1742. During the 18th century, issues regarding probability arose in connection with gambling and with the new concept of insurance. One particularly important problem concerned so-called inverse probability. A solution was proposed by Thomas Bayes in his paper 'Essay towards solving a problem in the doctrine of chances', which was published in 1764, some three years after his death, in the *Philosophical Transactions of the Royal Society*. In fact, Bayes only formulated his theory for the case of a uniform prior, and it was Pierre-Simon Laplace who independently rediscovered the theory in general form and who demonstrated its broad applicability.



Pierre-Simon Laplace

1749–1827

It is said that Laplace was seriously lacking in modesty and at one point declared himself to be the best mathematician in France at the time, a claim that was arguably true. As well as being prolific in mathematics,

he also made numerous contributions to astronomy, including the nebular hypothesis by which the earth is thought to have formed from the condensation and cooling of a large rotating disk of gas and dust. In 1812 he published the first edition of *Théorie Analytique des Probabilités*, in which Laplace states that “probability theory is nothing but common sense reduced to calculation”. This work included a discussion of the inverse probability calculation (later termed Bayes’ theorem by Poincaré), which he used to solve problems in life expectancy, jurisprudence, planetary masses, triangulation, and error estimation.

Bayesian Probabilities

- Bayesian viewpoint → prior knowledge (selection of prior criticized by math convenience rather than a reflection of any prior beliefs)
- Reducing the dependence on the choice of prior is seen as “**noninformative priors**”.
- Cross-validation can be applied for model comparison.
- Bayesian framework was limited by the difficulties in carrying through **full Bayesian** procedure, i.e. the need to marginalize (sum or integrate) over the whole parameter space, used for making predictions or comparing different models.
⇒ Sampling method → Markov chain Monte Carlo
(computationally intensive)
→ **Variational Bayes**

Gaussian Distribution

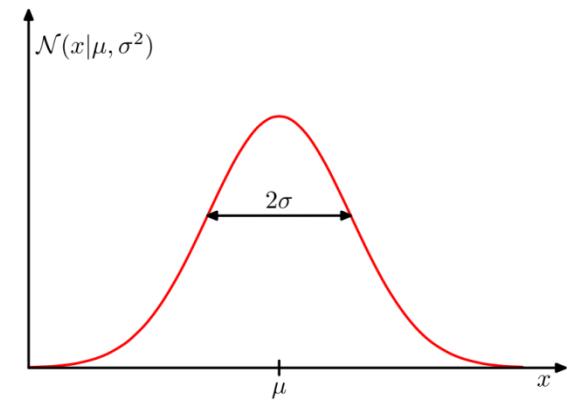
$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} > 0$$

$$\int_{-\infty}^{\infty} \mathcal{N}(x|\mu, \sigma^2) dx = 1 \quad \mathbb{E}[x] = \mu \quad \mathbb{E}[x^2] = \mu^2 + \sigma^2$$

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

$$x \in R^D$$

$$= (x_1, \dots, x_D)^T$$



i.i.d. independent and identically distributed

Let $\mathbf{x} = (x_1, \dots, x_N)^T$ is i.i.d.

$$p(\mathbf{x}|\mu, \sigma^2) = \prod_{n=1}^N \mathcal{N}(x_n|\mu, \sigma^2)$$

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

Maximum Likelihood Estimation

$$\frac{\partial \ln p(\mathbf{X}|\mu, \sigma^2)}{\partial \mu} = 0 \implies \mu_{\text{ML}} = \frac{1}{N} \sum_{n=1}^N x_n$$
$$\frac{\partial \ln p(\mathbf{X}|\mu, \sigma^2)}{\partial \sigma^2} = 0 \implies \sigma_{\text{ML}}^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \mu_{\text{ML}})^2$$

- ML underestimates the variance $\sigma^2 \rightarrow$ bias

$$\mathbb{E}[\mu_{\text{ML}}] = \frac{1}{N} \sum_{n=1}^N \mathbb{E}\{x_n\} = \mu$$

$$\mathbb{E}[\sigma_{\text{ML}}^2] = \left(\frac{N-1}{N}\right) \sigma^2$$

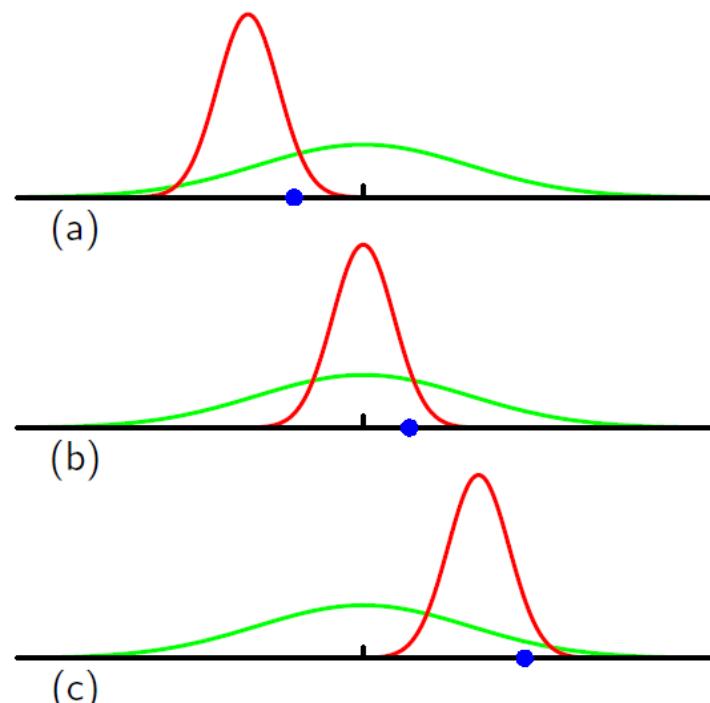
- Averaged across the three data sets, the mean is correct .

Variance is under-estimated

$$\lim_{N \rightarrow \infty} \mathbb{E}[\sigma_{\text{ML}}^2] = \sigma^2$$

- For unbiased variance parameter,

$$\tilde{\sigma}^2 = \frac{N}{N-1} \sigma_{\text{ML}}^2 = \frac{1}{N-1} \sum_{n=1}^N (x_n - \mu_{\text{ML}})^2$$



Curve Fitting Re-visited

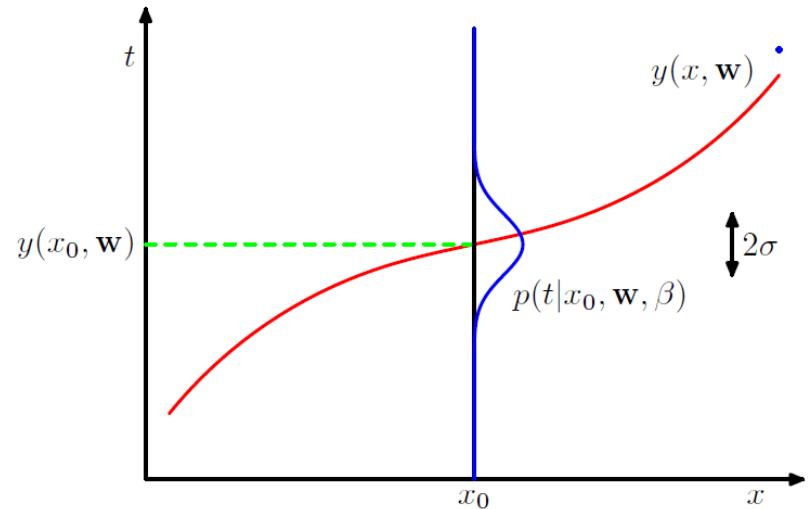
Given $\mathbf{x} = (x_1, \dots, x_N)^T$ $\mathbf{t} = (t_1, \dots, t_N)^T$

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

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

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

$$+ \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi)$$



- Maximum likelihood \leftrightarrow sum-of-squares error function

$$\frac{\partial \ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta)}{\partial \beta} = 0 \implies \frac{1}{\beta_{ML}} = \frac{1}{N} \sum_{n=1}^N \{y(x_n, \mathbf{w}_{ML}) - t_n\}^2$$

$\rightarrow \mathbf{w}_{ML} \rightarrow p(t|x, \mathbf{w}_{ML}, \beta_{ML})$ point estimate

Maximum a Posteriori Estimation

- Bayesian approach considering a **prior** distribution over W

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) = \left(\frac{\alpha}{2\pi}\right)^{(M+1)/2} \exp\left\{-\frac{\alpha}{2}\mathbf{w}^T\mathbf{w}\right\}$$

$$p(\mathbf{w}|\mathbf{x}, \mathbf{t}, \alpha, \beta) \propto p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta)p(\mathbf{w}|\alpha)$$

- Maximum a *posteriori* (**MAP**)

$$\begin{aligned}\mathbf{w}_{\text{MAP}} &= \underset{\mathbf{w}}{\operatorname{argmax}} p(\mathbf{w}|\mathbf{x}, \mathbf{t}, \alpha, \beta) \\ &= \underset{\mathbf{w}}{\operatorname{argmin}} \left\{ \frac{\beta}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\alpha}{2} \mathbf{w}^T \mathbf{w} \right\}\end{aligned}$$

regularized sum-of-squares error function if
regularization parameter $\lambda = \alpha/\beta$

Predictive Distribution

Full Bayesian approach treatment → Integrate over all values of \mathbf{w} (marginalization)

- Predictive distribution

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

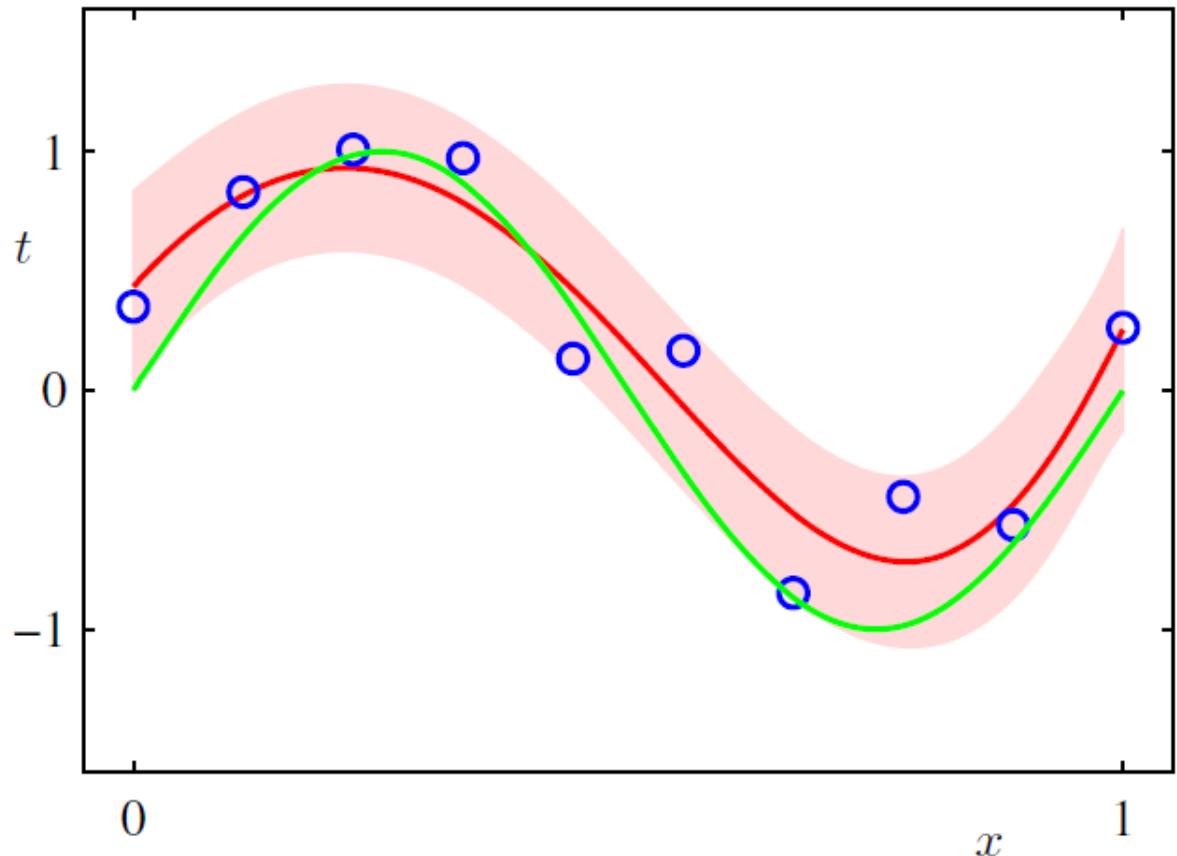
$$p(t|x, \mathbf{x}, \mathbf{t}) = \mathcal{N}(t|m(x), s^2(x))$$

$$\begin{aligned} m(x) &= \beta \phi(x)^T \mathbf{S} \sum_{n=1}^N \phi(x_n) t_n \\ s^2(x) &= \beta^{-1} + \phi(x)^T \mathbf{S} \phi(x). \end{aligned} \quad \phi(x) = \{\phi_i(x) = x^i\}_{i=0}^M$$

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

Bayesian Curve Fitting

$$\begin{aligned}M &= 9 \\ \alpha &= 5 \times 10^{-3} \\ \beta &= 11.1\end{aligned}$$

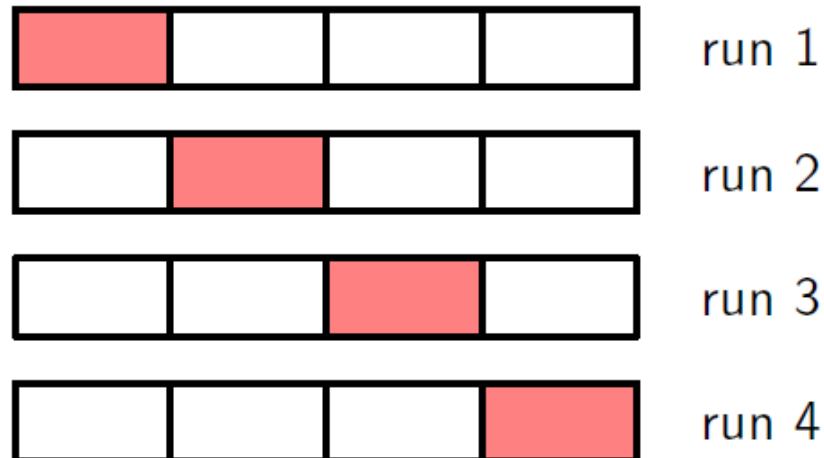


Cross Validation

Optimal order → the best generalization

→ Model complexity → predictive performance on new data
validation set → test set

- Illustration of S-fold cross-validation



Model Selection

Leave-one-out $S=N$

⇒ We have many parameters & regularization parameters
(combination of training runs & setting)

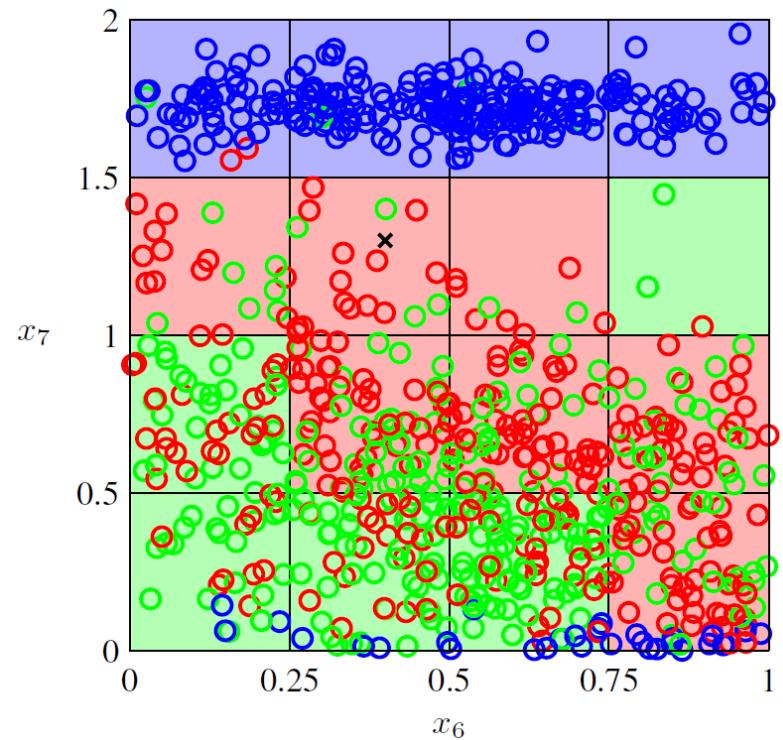
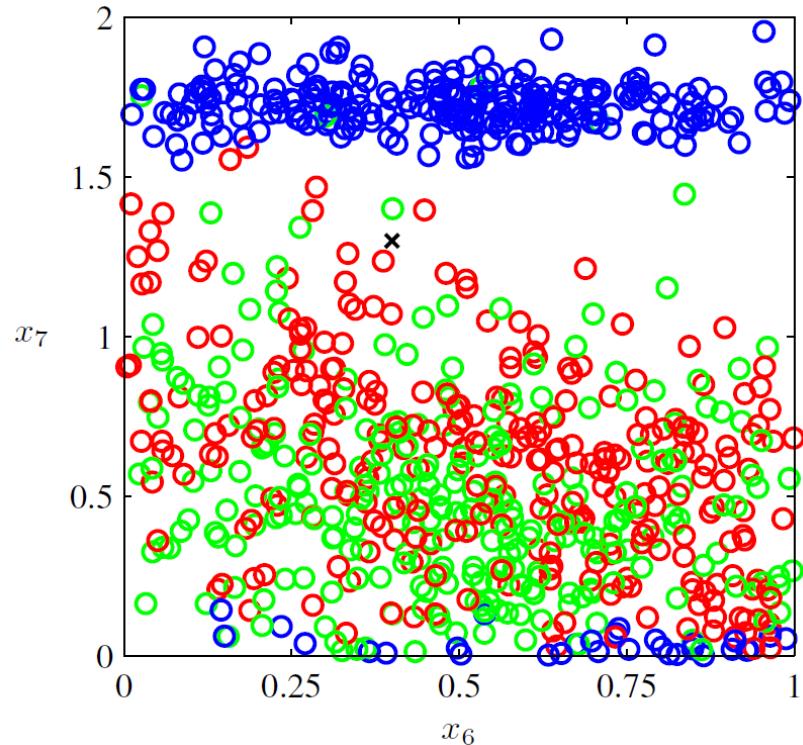
⇒ computationally expensive

⇒ Akaike information criterion (**AIC**)

$$\ln p(\mathcal{D}|\mathbf{w}_{\text{ML}}) - M$$

⇒ Bayesian information criterion (**BIC**)

Curse of Dimensionality



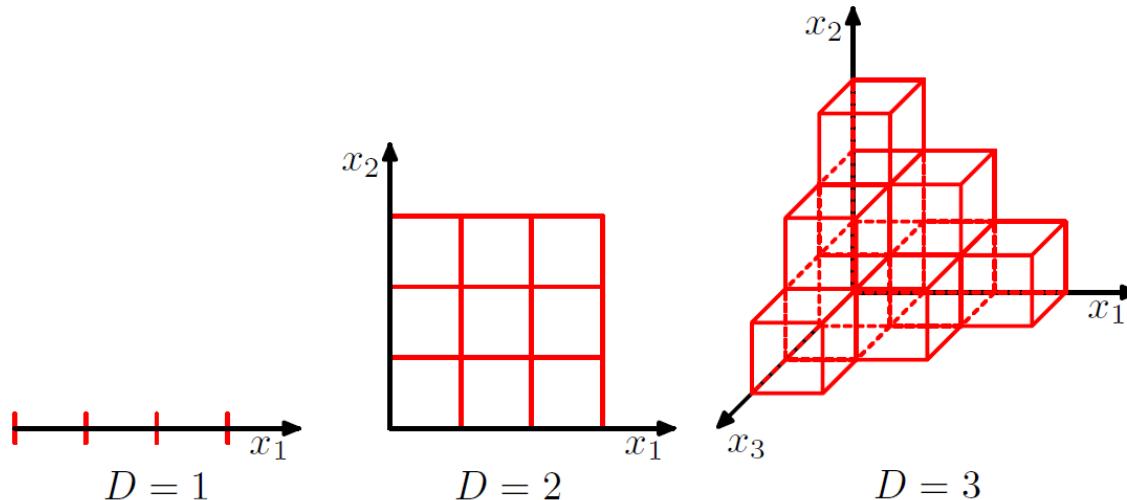
⇒ See Appendix for these data sets

Divide input space into
regular cells



Comparable to k -nearest-
neighbor classifier

Curse of Dimensionality



- Illustration of how the number of regions of a regular grid grows ‘exponentially’ with dimensionality D

Considering a general polynomial with up to order 3 for D input variables

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

number of independent terms $\propto D^3$

Decision Theory

- Make **optimal decisions** in situation involving uncertainty in pattern recognition

Input $\mathbf{x} \rightarrow$ target t X-ray image

C_1 (presence of cancer) C_2 (absence of cancer)

$t = 0$ $t = 1$

→ give the treatment to the patient or not

$$\hat{C}_{MAP} = \operatorname{argmax}_{C_k} P(C_k | \mathbf{x}) = \operatorname{argmax}_{C_k} P(\mathbf{x}, C_k)$$

$$p(\mathcal{C}_k | \mathbf{x}) = \frac{p(\mathbf{x} | \mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}$$

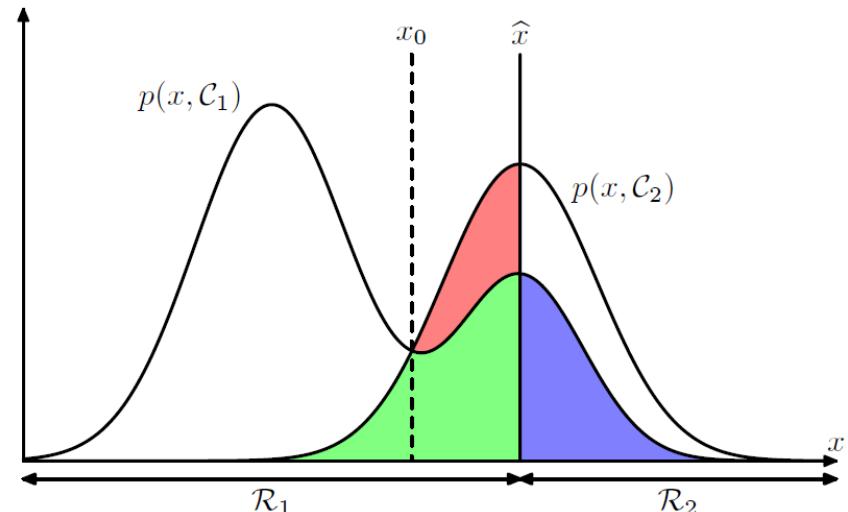
Minimizing the Misclassification Rate

- Decision regions $R_k \rightarrow$ decision boundaries / surfaces

$$\begin{aligned} p(\text{mistake}) &= p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1) \\ &= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x} \end{aligned}$$

- If $\hat{x} = x_0$, optimal choice red region disappears.
- Minimum misclassification rate or **minimum error rate** decision rule
- For k -class problem,

$$\begin{aligned} p(\text{correct}) &= \sum_{k=1}^K p(\mathbf{x} \in \mathcal{R}_k, \mathcal{C}_k) \\ &= \sum_{k=1}^K \int_{\mathcal{R}_k} p(\mathbf{x}, \mathcal{C}_k) d\mathbf{x} \end{aligned}$$



Minimizing the Expected Loss

- **Loss matrix**

$$\{L_{kj}\} \quad \begin{array}{cc} & \text{cancer} \quad \text{normal} \\ \text{cancer} & \left(\begin{array}{cc} 0 & 1000 \\ 1 & 0 \end{array} \right) \\ \text{normal} & \end{array}$$

- Minimize total loss incurred

L_{kj} : we assign \mathbf{x} to class C_j if the true class is C_k

- We are minimizing the **Bayes risk** or the **expected loss**

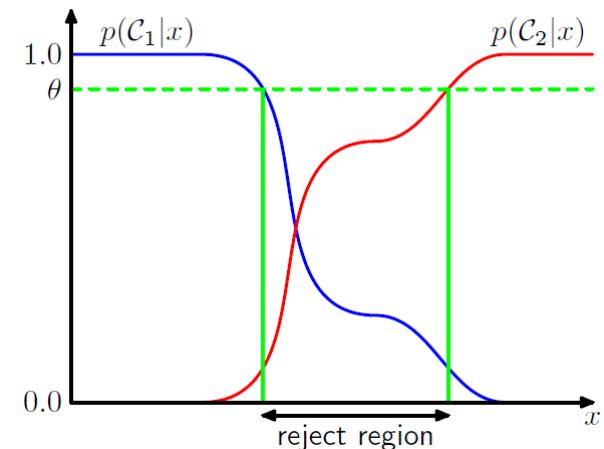
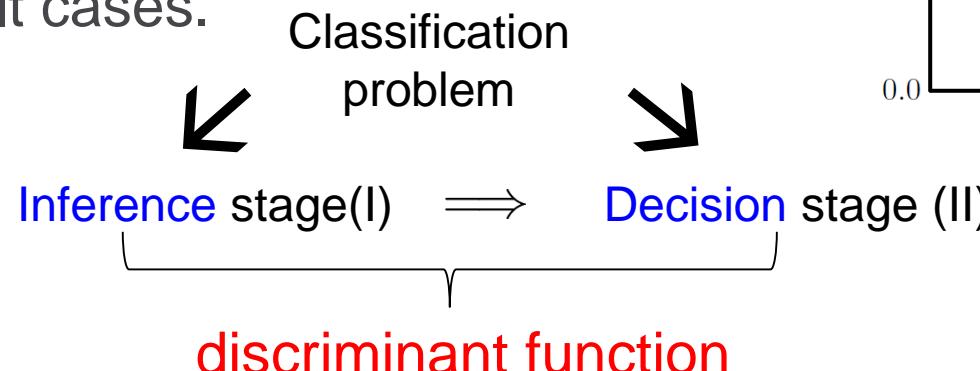
$$\mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(\mathbf{x}, C_k) d\mathbf{x}$$

- Equivalently, we assign new \mathbf{x} to class C_j if

$$\hat{C}_j = \operatorname{argmin}_{C_j} \sum_k L_{kj} P(C_k | \mathbf{x}) \text{ is minimum}$$

The Reject Option

- Avoid making decisions on the difficult cases.



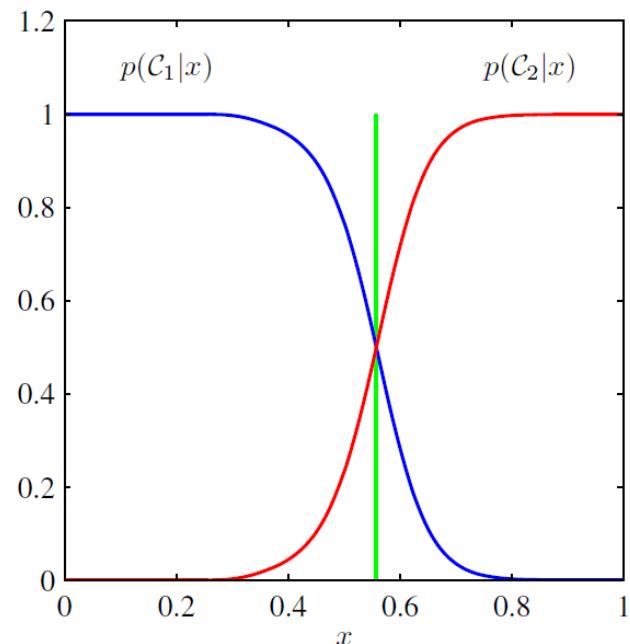
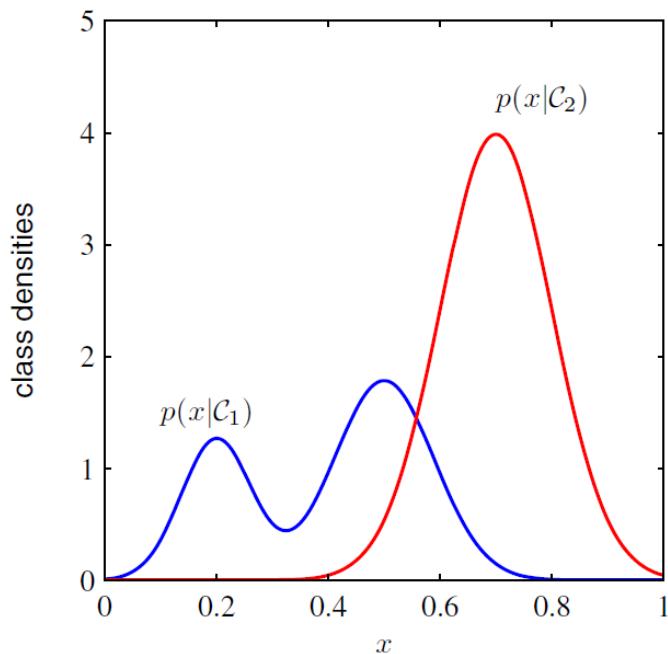
- Three distinct approaches to solving decision problems
- Inference problems \rightarrow decreasing model complexity

(a) $p(\mathbf{x}|\mathcal{C}_k)$
generative
models

(b) $p(\mathcal{C}_k|\mathbf{x})$
discriminative
models

(c) $f(\mathbf{x})$
discriminant
function

Inference and Decision



Outlier detection / novelty detection

- Reasons for computing posterior probabilities $p(\mathcal{C}_k|\mathbf{x})$
 1. Minimizing **risk**
 2. **Reject** option
 3. Compensating for **class prior** (See Text)
 4. **Combining** models (See Text)

Model Combination

$$p(\mathbf{x}_I, \mathbf{x}_B | \mathcal{C}_k) = p(\mathbf{x}_I | \mathcal{C}_k)p(\mathbf{x}_B | \mathcal{C}_k)$$

$$\begin{aligned} p(\mathcal{C}_k | \mathbf{x}_I, \mathbf{x}_B) &\propto p(\mathbf{x}_I, \mathbf{x}_B | \mathcal{C}_k)p(\mathcal{C}_k) \\ &\propto p(\mathbf{x}_I | \mathcal{C}_k)p(\mathbf{x}_B | \mathcal{C}_k)p(\mathcal{C}_k) \\ &\propto \frac{p(\mathcal{C}_k | \mathbf{x}_I)p(\mathcal{C}_k | \mathbf{x}_B)}{p(\mathcal{C}_k)} \end{aligned}$$

Loss Functions for Regression

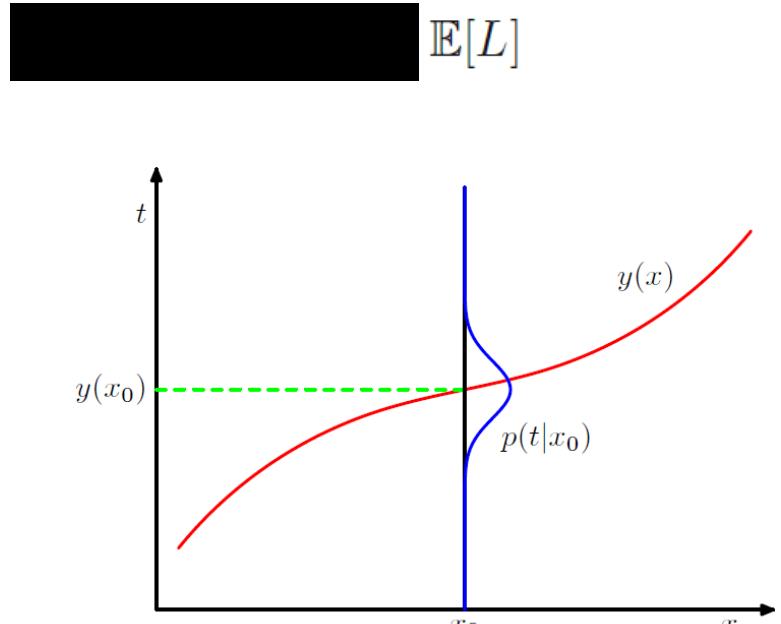
$$L(t, y(\mathbf{x}))$$

$$\mathbb{E}[L] = \iint L(t, y(\mathbf{x})) p(\mathbf{x}, t) d\mathbf{x} dt$$

$$\boxed{\mathbb{E}[L] = \iint \{y(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt}$$

$$\frac{\delta \mathbb{E}[L]}{\delta y(\mathbf{x})} = 2 \int \{y(\mathbf{x}) - t\} p(\mathbf{x}, t) dt = 0$$

$$\implies \boxed{y(\mathbf{x}) = \frac{\int tp(\mathbf{x}, t) dt}{p(\mathbf{x})} = \int tp(t|\mathbf{x}) dt = \mathbb{E}_t[t|\mathbf{x}]}$$



Here, $\{y(\mathbf{x}) - t\}^2 = \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}] + \mathbb{E}[t|\mathbf{x}] - t\}^2$

$$0 = \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 + 2\{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}\{\mathbb{E}[t|\mathbf{x}] - t\} + \{\mathbb{E}[t|\mathbf{x}] - t\}^2$$

$$\boxed{\mathbb{E}[L] = \int \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 p(\mathbf{x}) d\mathbf{x} + \int \{\mathbb{E}[t|\mathbf{x}] - t\}^2 p(\mathbf{x}) d\mathbf{x}}$$

Intrinsic variability
of target data

Inference and Loss Function

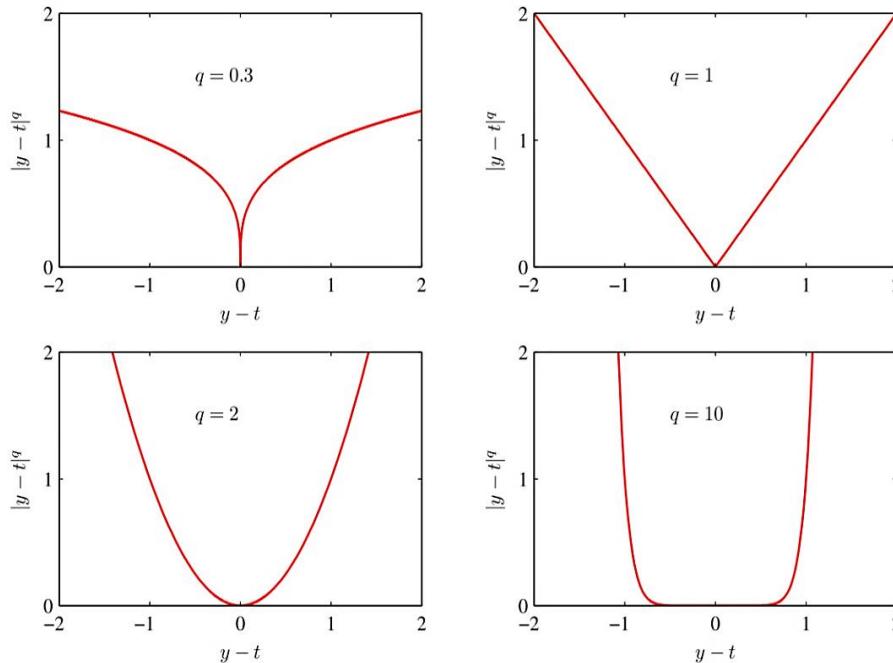
(a) Inference for $p(\mathbf{x}, t)$

$$\rightarrow p(t|\mathbf{x}) \rightarrow \mathbb{E}[t|\mathbf{x}] \rightarrow y(\mathbf{x})$$

(b) Inference for $p(t|\mathbf{x}) \rightarrow \mathbb{E}[t|\mathbf{x}] \rightarrow y(\mathbf{x})$

(c) Directly find $y(\mathbf{x})$ from training data

- *Minkowski* loss $\mathbb{E}[L_q] = \iint |y(\mathbf{x}) - t|^q p(\mathbf{x}, t) d\mathbf{x} dt$





Ludwig Boltzmann

1844–1906

Ludwig Eduard Boltzmann was an Austrian physicist who created the field of statistical mechanics. Prior to Boltzmann, the concept of entropy was already known from classical thermodynamics where it

quantifies the fact that when we take energy from a system, not all of that energy is typically available to do useful work. Boltzmann showed that the thermodynamic entropy S , a macroscopic quantity, could be related to the statistical properties at the microscopic level. This is expressed through the famous equation $S = k \ln W$ in which W represents the number of possible microstates in a macrostate, and $k \simeq 1.38 \times 10^{-23}$ (in units of Joules per Kelvin) is known as Boltzmann's constant. Boltzmann's ideas were disputed by many scientists of their day. One difficulty they saw arose from the second law of thermo-

dynamics, which states that the entropy of a closed system tends to increase with time. By contrast, at the microscopic level the classical Newtonian equations of physics are reversible, and so they found it difficult to see how the latter could explain the former. They didn't fully appreciate Boltzmann's arguments, which were statistical in nature and which concluded not that entropy could never decrease over time but simply that with overwhelming probability it would generally increase. Boltzmann even had a long-running dispute with the editor of the leading German physics journal who refused to let him refer to atoms and molecules as anything other than convenient theoretical constructs. The continued attacks on his work lead to bouts of depression, and eventually he committed suicide. Shortly after Boltzmann's death, new experiments by Perrin on colloidal suspensions verified his theories and confirmed the value of the Boltzmann constant. The equation $S = k \ln W$ is carved on Boltzmann's tombstone.



Claude Shannon

1916–2001

After graduating from Michigan and MIT, Shannon joined the AT&T Bell Telephone laboratories in 1941. His paper ‘A Mathematical Theory of Communication’ published in the *Bell System Technical Journal* in

1948 laid the foundations for modern information theory. This paper introduced the word ‘bit’, and his concept that information could be sent as a stream of 1s and 0s paved the way for the communications revolution. It is said that von Neumann recommended to Shannon that he use the term entropy, not only because of its similarity to the quantity used in physics, but also because “nobody knows what entropy really is, so in any discussion you will always have an advantage”.

Information Theory

$$x, y \quad h(x, y) = h(x) + h(y) \quad h(x) = -\log_2 p(x)$$

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

- **Entropy:** average amount of information needed to specify the state of a random variable
- Example:

8 possible states (equally likely)

$$H[x] = -8 \times \frac{1}{8} \log_2 \frac{1}{8} = 3 \text{ bits}$$

$$\{a, b, c, d, e, f, g, h\} \rightarrow \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}\right)$$

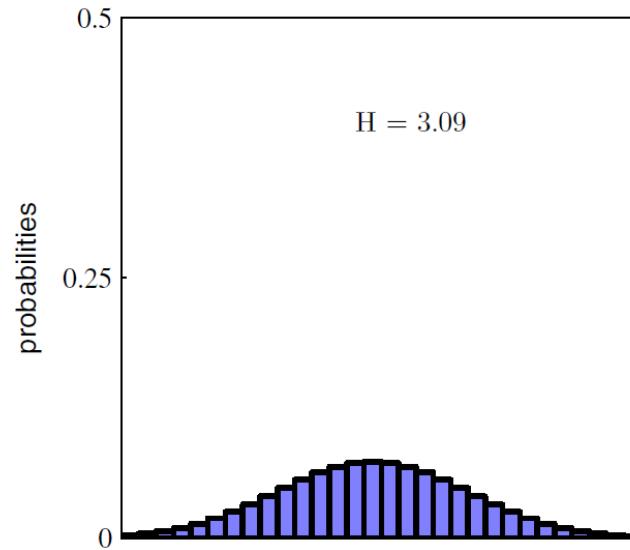
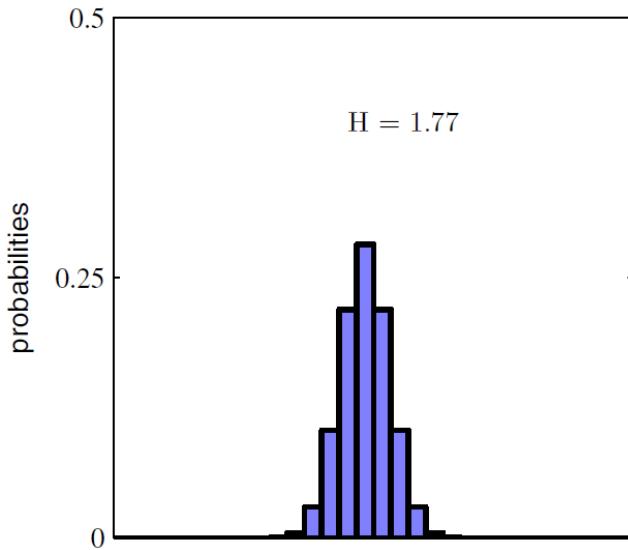
$$H[x] = -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{4} \log_2 \frac{1}{4} - \frac{1}{8} \log_2 \frac{1}{8} - \frac{1}{16} \log_2 \frac{1}{16} - \frac{4}{64} \log_2 \frac{1}{64} = 2 \text{ bits}$$

nonuniform distribution has a smaller entropy than the uniform one $\{0, 10, 110, 1110, 111100, 111101, 111110, 111111\}$

Entropy

$$\text{average code length} = \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{16} \times 4 + 4 \times \frac{1}{64} \times 6 = 2 \text{ bits}$$

- Entropy is a **lower bound** on the no. of bits needed to transmit the state of a random variable . → noiseless coding theorem (Shannon, 1948) $H[p] = - \sum_i p(x_i) \ln p(x_i)$



$$\tilde{H} = - \sum_i p(x_i) \ln p(x_i) + \lambda \left(\sum_i p(x_i) - 1 \right)$$

- If x is continuous random variable,

$H[x] = - \int p(x) \ln p(x) dx$ differential entropy ME with three constraints

$$-\int_{-\infty}^{\infty} p(x) \ln p(x) dx + \lambda_1 \left(\int_{-\infty}^{\infty} p(x) dx - 1 \right) + \lambda_2 \left(\int_{-\infty}^{\infty} xp(x) dx - \mu \right) + \lambda_3 \left(\int_{-\infty}^{\infty} (x - \mu)^2 p(x) dx - \sigma^2 \right)$$

$$p(x) = \exp \left\{ -1 + \lambda_1 + \lambda_2 x + \lambda_3 (x - \mu)^2 \right\}$$

- Back substitution, we obtain

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

- Distribution that maximizes the differential entropy is the Gaussian

$$\implies H[x] = \frac{1}{2} \{ 1 + \ln(2\pi\sigma^2) \}$$

$\sigma^2 \nearrow$ (broader) \rightarrow entropy \nearrow

$$H[y|x] = - \iint p(y, x) \ln p(y|x) dy dx$$

$$H[x, y] = H[y|x] + H[x]$$

conditional
entropy

Kullback-Leibler Divergence & Convex Function

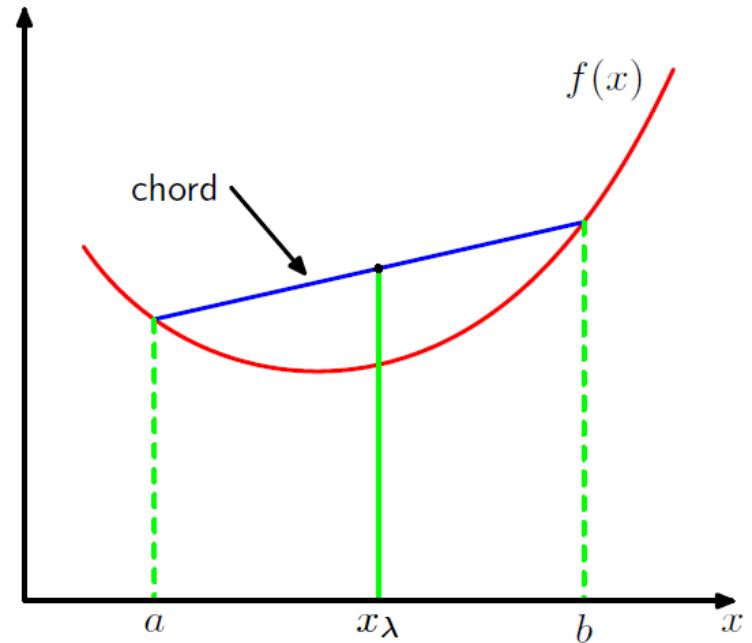
$$\begin{aligned}\text{KL}(p\|q) &= - \int p(\mathbf{x}) \ln q(\mathbf{x}) d\mathbf{x} - \left(- \int p(\mathbf{x}) \ln p(\mathbf{x}) d\mathbf{x} \right) \\ &= - \int p(\mathbf{x}) \ln \left\{ \frac{q(\mathbf{x})}{p(\mathbf{x})} \right\} d\mathbf{x}.\end{aligned}$$

- Relative entropy / Kullback-Leibler (**KL**) divergence (1951) between $p(\mathbf{x})$ and $q(\mathbf{x})$

$$\text{KL}(p\|q) \geq 0 \quad \text{KL}(p\|q) \neq \text{KL}(q\|p)$$

Any value between $x = a$ & $x = b$ can be written by $0 \leq \lambda \leq 1$
 $\lambda a + (1 - \lambda)b$

$$\lambda f(a) + (1 - \lambda)f(b)$$



Convexity: $f(\lambda a + (1 - \lambda)b) \leq \lambda f(a) + (1 - \lambda)f(b)$

$\iff f''(x) \geq 0$ i.e. single optimum

e.g. $x \ln x$ (for $x > 0$) , x^2

If $f(x)$ is convex , $-f(x)$ is concave.

Concave: $f(\lambda a + (1 - \lambda)b) \geq \lambda f(a) + (1 - \lambda)f(b)$

A convex function $f(x)$ satisfies

$$f\left(\sum_{i=1}^M \lambda_i x_i\right) \leq \sum_{i=1}^M \lambda_i f(x_i) \quad \text{where } \lambda_i \geq 0 \text{ and } \sum_i \lambda_i = 1$$

Jensen's Inequality

Prove by induction.

When λ_i is a probability distribution of a discrete variable x ,

$$\implies f(\mathbb{E}[x]) \leq \mathbb{E}[f(x)]$$

Continuous x $\longrightarrow f\left(\int x p(x) dx\right) \leq \int f(x) p(x) dx$

$$\text{KL}(p\|q) = - \int p(\mathbf{x}) \ln \left\{ \frac{q(\mathbf{x})}{p(\mathbf{x})} \right\} d\mathbf{x} \geq - \ln \int q(\mathbf{x}) d\mathbf{x} = 0$$

- It is because ‘ $-\ln x$ ’ is a convex function
- **KL divergence** serves as a measure of the dissimilarity between $p(\mathbf{x})$ and $q(\mathbf{x})$

Data compression \longleftrightarrow Density estimation

- The most efficient compression is achieved when we know the true distribution. Less coding when a distribution is different from true one.
- KL divergence between $p(\mathbf{x})$ and $q(\mathbf{x})$ is the additional information that must be transmitted.

parametric model $q(\mathbf{x}|\boldsymbol{\theta}) \rightarrow$ true model $p(\mathbf{x})$

$$\hat{\boldsymbol{\theta}} = \operatorname{argmin}_{\boldsymbol{\theta}} KL(p(\mathbf{x})|q(\mathbf{x}|\boldsymbol{\theta})) = \operatorname{argmax}_{\boldsymbol{\theta}} (\ln) q(\mathbf{x}|\boldsymbol{\theta})$$

Given $\{\mathbf{x}_n, n = 1, 2, \dots, N\}$ drawn from $p(\mathbf{x})$

$$KL(p\|q) \simeq \sum_{n=1}^N \{-\ln q(\mathbf{x}_n|\boldsymbol{\theta}) + \ln p(\mathbf{x}_n)\}$$

Relative Entropy and Mutual Information

$$\begin{aligned} I[\mathbf{x}, \mathbf{y}] &\equiv \text{KL}(p(\mathbf{x}, \mathbf{y}) \| p(\mathbf{x})p(\mathbf{y})) \\ &= - \iint p(\mathbf{x}, \mathbf{y}) \ln \left(\frac{p(\mathbf{x})p(\mathbf{y})}{p(\mathbf{x}, \mathbf{y})} \right) d\mathbf{x} d\mathbf{y} \end{aligned}$$

$I(\mathbf{x}, \mathbf{y}) \geq 0$ Equality holds when \mathbf{x} & \mathbf{y} are independent.

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

Reduction of **uncertainty** about \mathbf{x} of being given the value of \mathbf{y} .

Reduction of **uncertainty** about \mathbf{x} as a consequence of the new observation \mathbf{y} .

CONTENTS

1. Introduction
2. Probability Distributions
3. Linear Models for Regression
4. Linear Models for Classification
5. Kernel Methods
6. Sparse Kernel Methods
7. Mixture Models and EM
8. Approximate Inference

Parametric distribution → Nonparametric distribution

↳ depends on size of data set

Conjugate prior

$$D = \mathbf{x}_1, \dots, \mathbf{x}_N \implies p(\mathbf{x})$$

Binary Variables

Bernoulli (Swiss) , 1654-1705

$$x \in \{0, 1\} \quad \text{Bern}(x|\mu) = \mu^x(1-\mu)^{1-x}$$

$$p(\mathcal{D}|\mu) = \prod_{n=1}^N p(x_n|\mu) = \prod_{n=1}^N \mu^{x_n}(1-\mu)^{1-x_n}$$

$$\mu_{\text{ML}} = \frac{1}{N} \sum_{n=1}^N x_n$$

$$\text{Bin}(m|N, \mu) = \binom{N}{m} \mu^m (1-\mu)^{N-m}$$

Number m of
observations of $x=1$

$$\mathbb{E}[m] \equiv \sum_{m=0}^N m \text{Bin}(m|N, \mu) = N\mu$$

$$\text{var}[m] \equiv \sum_{m=0}^N (m - \mathbb{E}[m])^2 \text{Bin}(m|N, \mu) = N\mu(1-\mu)$$



Jacob Bernoulli

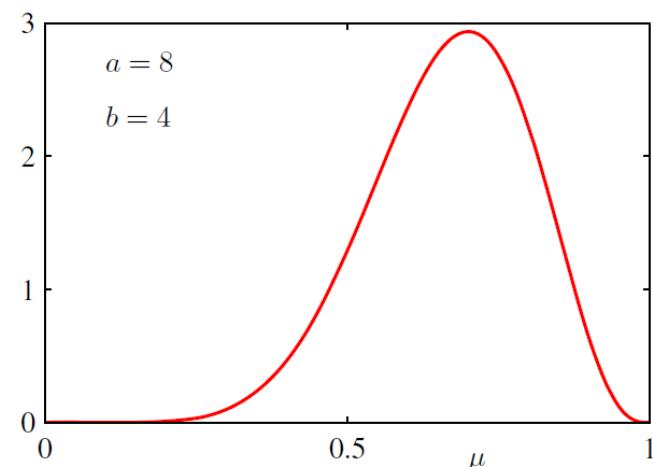
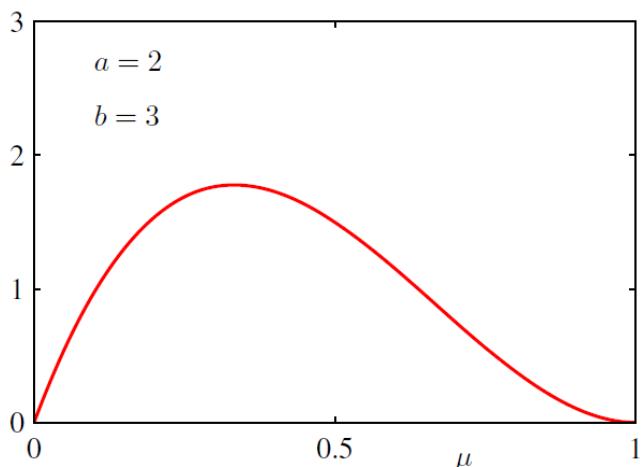
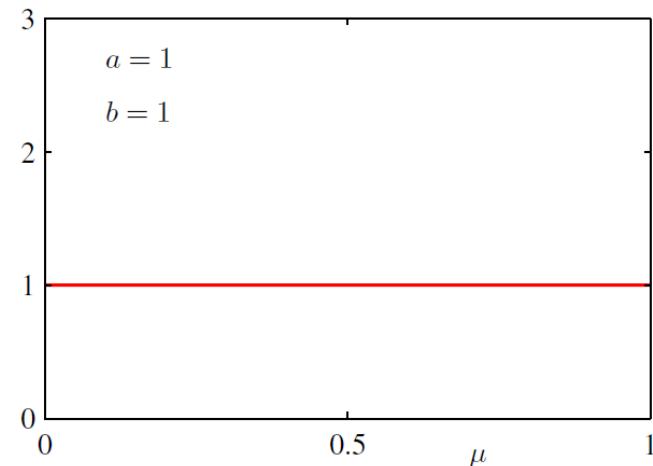
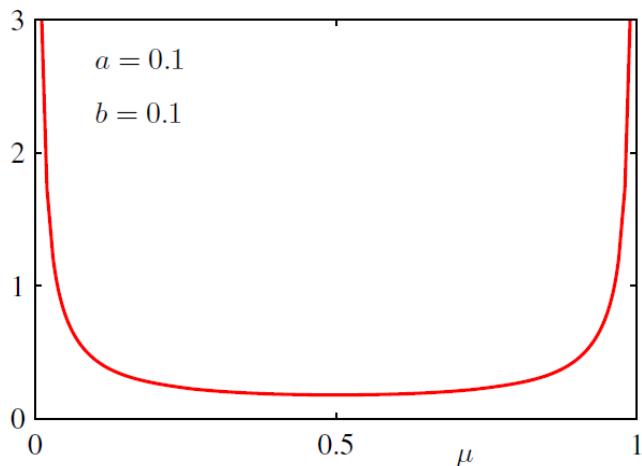
1654–1705

Jacob Bernoulli, also known as Jacques or James Bernoulli, was a Swiss mathematician and was the first of many in the Bernoulli family to pursue a career in science and mathematics. Although compelled

to study philosophy and theology against his will by his parents, he travelled extensively after graduating in order to meet with many of the leading scientists of his time, including Boyle and Hooke in England. When he returned to Switzerland, he taught mechanics and became Professor of Mathematics at Basel in 1687. Unfortunately, rivalry between Jacob and his younger brother Johann turned an initially productive collaboration into a bitter and public dispute. Jacob's most significant contributions to mathematics appeared in *The Art of Conjecture* published in 1713, eight years after his death, which deals with topics in probability theory including what has become known as the Bernoulli distribution.

Conjugate Prior

When performing **Bayesian treatment** for Bernoulli distribution, we should introduce $p(\mu)$ which has the property of “*conjugacy*”



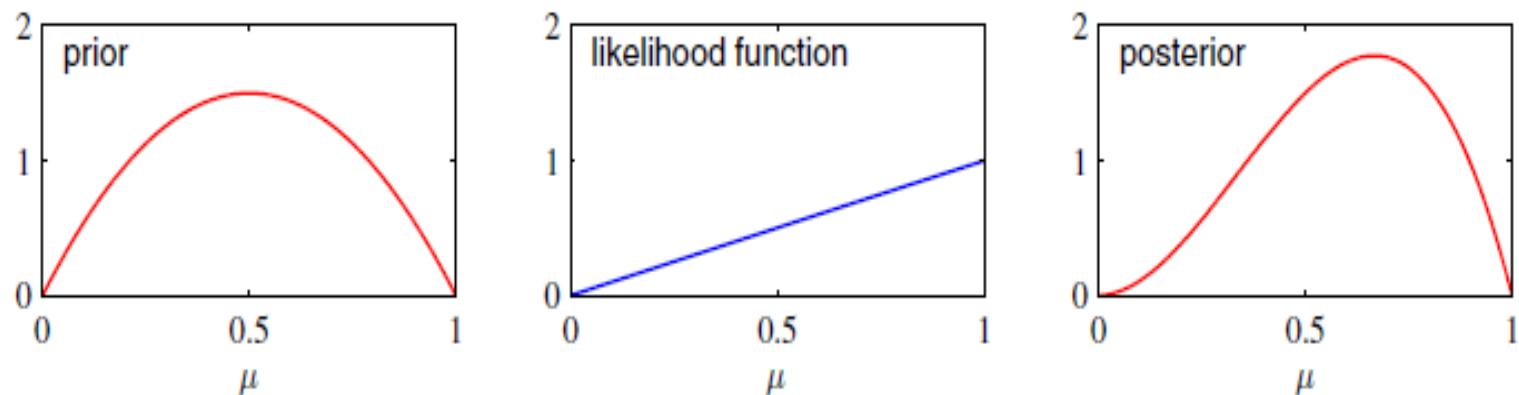
Beta Distribution

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

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

$$p(\mu|m, l, a, b) \propto \mu^{m+a-1} (1-\mu)^{l+b-1}$$

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



Some Properties

a, b can be interpreted as “**effective number of observations**” of x=1 and x=0.

Sequential approach → make use of observations one at a time, or in small batches, and then discard them before the next observations are used. → real-time learning scenarios

- In Bayesian prediction , we evaluate **predictive distribution** of x

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

$$= \frac{m + a}{m + a + l + b} \xrightarrow{\lim_{m,l \rightarrow \infty}} \frac{M}{N} \text{ maximum likelihood estimate}$$

$N \nearrow \text{Var} [\mu] \downarrow$ uncertainty \downarrow beta distribution becomes sharply.

As we observe more and more data, the uncertainty of posterior distribution will steadily **decrease**.

Multinomial Variables

$$\mathbf{x} = (0, 0, 1, 0, 0, 0)^T$$

Constraint $\sum_{k=1}^K x_k = 1 \quad \sum_k \mu_k = 1$

$$p(\mathbf{x}|\boldsymbol{\mu}) = \prod_{k=1}^K \mu_k^{x_k} \quad \text{generalization of the Bernoulli distribution.}$$
$$= p(\mathbf{x}_1, \dots, \mathbf{x}_k | \mu_1, \dots, \mu_k)^{k=1}$$

$$\sum_{\mathbf{x}} p(\mathbf{x}|\boldsymbol{\mu}) = \sum_{k=1}^K \mu_k = 1 \quad \mathbb{E}[\mathbf{x}|\boldsymbol{\mu}] = \sum_{\mathbf{x}} p(\mathbf{x}|\boldsymbol{\mu}) \mathbf{x} = (\mu_1, \dots, \mu_M)^T = \boldsymbol{\mu}$$

$$p(\mathcal{D}|\boldsymbol{\mu}) = \prod_{n=1}^N \prod_{k=1}^K \mu_k^{x_{nk}} = \prod_{k=1}^K \mu_k^{(\sum_n x_{nk})} = \prod_{k=1}^K \mu_k^{m_k}$$

$\mu_{ML} = \operatorname{argmax}_{\boldsymbol{\mu}} \ln p(\mathcal{D}|\boldsymbol{\mu})$ Lagrange optimization is applied

$$\Rightarrow \sum_{k=1}^K m_k \ln \mu_k + \lambda \left(\sum_{k=1}^K \mu_k - 1 \right)$$

$$\mu_k^{\text{ML}} = \frac{m_k}{N}$$

$$\text{Mult}(m_1, m_2, \dots, m_K | \boldsymbol{\mu}, N) = \binom{N}{m_1 m_2 \dots m_K} \prod_{k=1}^K \mu_k^{m_k}$$

$$\sum_{k=1}^K m_k = N$$



Lejeune Dirichlet

1805–1859

Johann Peter Gustav Lejeune Dirichlet was a modest and reserved mathematician who made contributions in number theory, mechanics, and astronomy, and who gave the first rigorous analysis of Fourier series.

His family originated from Richelet in Belgium, and the name Lejeune Dirichlet comes from 'le jeune de Richelet' (the young person from Richelet). Dirichlet's first paper, which was published in 1825, brought him instant fame. It concerned Fermat's last theorem, which claims that there are no positive integer solutions to $x^n + y^n = z^n$ for $n > 2$. Dirichlet gave a partial proof for the case $n = 5$, which was sent to Legendre for review and who in turn completed the proof. Later, Dirichlet gave a complete proof for $n = 14$, although a full proof of Fermat's last theorem for arbitrary n had to wait until the work of Andrew Wiles in the closing years of the 20th century.

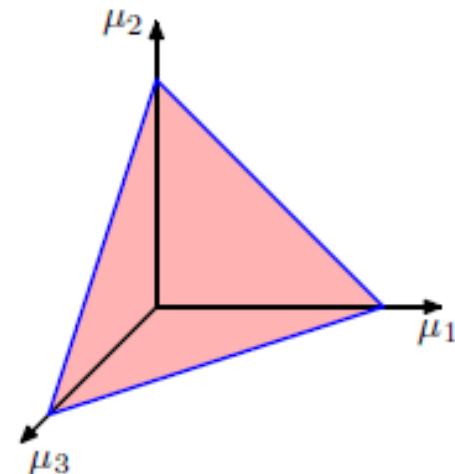
Dirichlet Distribution

Prior distribution for the parameters $\{\mu_k\}$ of a multinomial distribution is a *Dirichlet* distribution.

$$p(\boldsymbol{\mu}|\boldsymbol{\alpha}) \propto \prod_{k=1}^K \mu_k^{\alpha_k-1} \quad 0 \leq \mu_k \leq 1 \quad \sum_k \mu_k = 1$$

Dirichlet distribution is confined to a **simplex** (a bounded linear manifold)

$$\text{Dir}(\boldsymbol{\mu}|\boldsymbol{\alpha}) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1) \cdots \Gamma(\alpha_K)} \prod_{k=1}^K \mu_k^{\alpha_k-1}$$



Dirichlet Distribution for Different Alpha

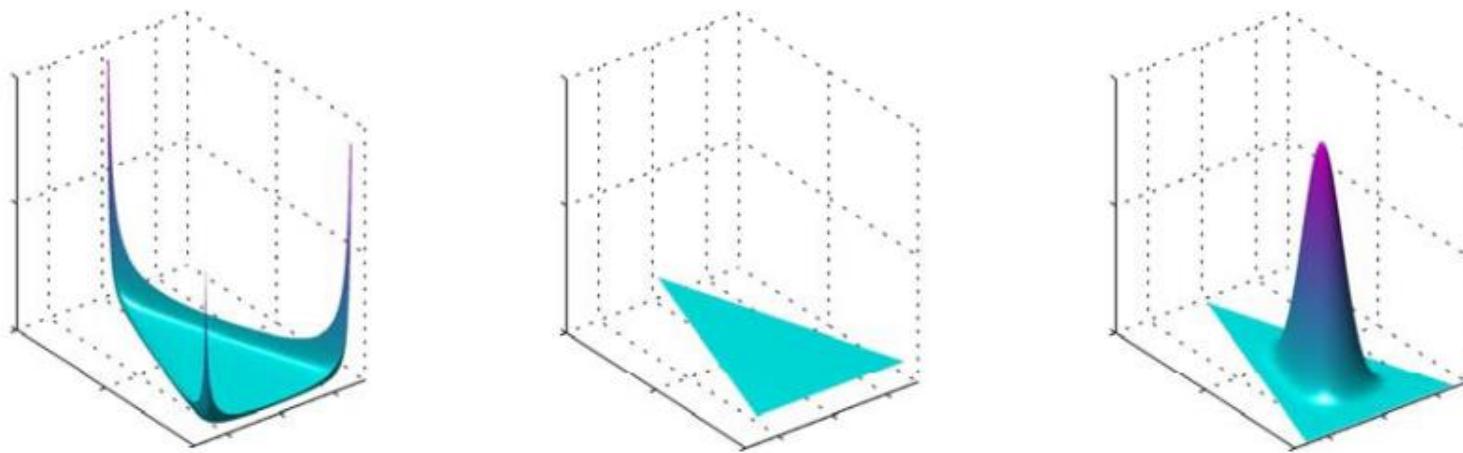


Figure 2.5

for an illustration of simplex
Posterior distribution

α_k : effective no. of observations of $x_k = 1$.

$$\text{Mult}(m_1, m_2, \dots, m_K | \boldsymbol{\mu}, N) \text{ Dir}(\boldsymbol{\mu} | \boldsymbol{\alpha})$$

Dirichlet Posterior Distribution

$$p(\boldsymbol{\mu}|\mathcal{D}, \boldsymbol{\alpha}) \propto p(\mathcal{D}|\boldsymbol{\mu})p(\boldsymbol{\mu}|\boldsymbol{\alpha}) \propto \prod_{k=1}^K \mu_k^{\alpha_k + m_k - 1}$$

↪ in a form of Dirichlet

$$\text{Dir}(\boldsymbol{\mu}|\boldsymbol{\alpha} + \mathbf{m}) = \frac{\Gamma(\alpha_0 + N)}{\Gamma(\alpha_1 + m_1) \cdots \Gamma(\alpha_K + m_K)} \prod_{k=1}^K \mu_k^{\alpha_k + m_k - 1}$$



Carl Friedrich Gauss

1777–1855

It is said that when Gauss went to elementary school at age 7, his teacher Büttner, trying to keep the class occupied, asked the pupils to sum the integers from 1 to 100. To the teacher's amazement, Gauss

arrived at the answer in a matter of moments by noting that the sum can be represented as 50 pairs ($1 + 100$, $2 + 99$, etc.) each of which added to 101, giving the answer 5,050. It is now believed that the problem which was actually set was of the same form but somewhat harder in that the sequence had a larger starting value and a larger increment. Gauss was a German math-

ematician and scientist with a reputation for being a hard-working perfectionist. One of his many contributions was to show that least squares can be derived under the assumption of normally distributed errors. He also created an early formulation of non-Euclidean geometry (a self-consistent geometrical theory that violates the axioms of Euclid) but was reluctant to discuss it openly for fear that his reputation might suffer if it were seen that he believed in such a geometry. At one point, Gauss was asked to conduct a geodetic survey of the state of Hanover, which led to his formulation of the normal distribution, now also known as the Gaussian. After his death, a study of his diaries revealed that he had discovered several important mathematical results years or even decades before they were published by others.

Gaussian Distribution

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

Central limit theorem

Eigenvector equation

$$\boldsymbol{\Sigma} \mathbf{u}_i = \lambda_i \mathbf{u}_i$$

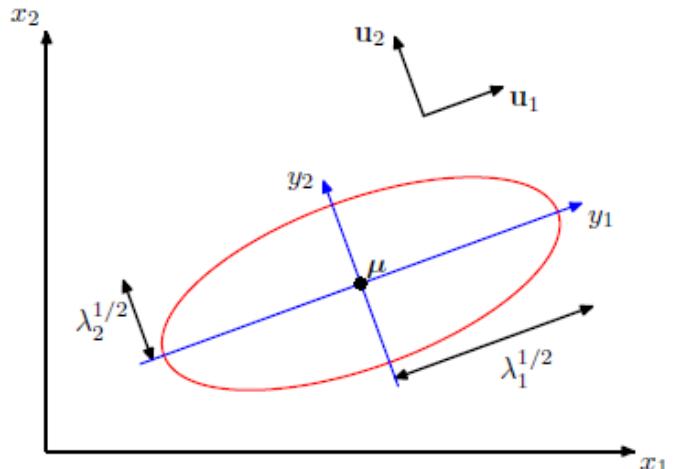
$$\mathbf{u}_i^T \mathbf{u}_j = I_{ij}$$

$$\boldsymbol{\Sigma} = \sum_{i=1}^D \lambda_i \mathbf{u}_i \mathbf{u}_i^T$$

$$\Delta^2 = \sum_{i=1}^D \frac{y_i^2}{\lambda_i}$$

$$\boldsymbol{\Sigma}^{-1} = \sum_{i=1}^D \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^T$$

$$y_i = \mathbf{u}_i^T (\mathbf{x} - \boldsymbol{\mu})$$
$$\mathbf{y} = \mathbf{U}(\mathbf{x} - \boldsymbol{\mu})$$



positive definite \rightarrow positive semidefinite

$$\lambda_i > 0$$

$$\lambda_i \geq 0$$

When transforming from x to y coordinate system. We have **Jacobian** matrix J with

$$J_{ij} = \frac{\partial x_i}{\partial y_j} = U_{ji} \quad J = U^T \quad |\Sigma|^{1/2} = \prod_{j=1}^D \lambda_j^{1/2}$$

$$|J|^2 = |U^T|^2 = |U^T| |U| = |U^T U| = |I| = 1$$

$$p(\mathbf{y}) = p(\mathbf{x})|J| = \prod_{j=1}^D \frac{1}{(2\pi\lambda_j)^{1/2}} \exp\left\{-\frac{y_j^2}{2\lambda_j}\right\}$$

Product of D independent univariate Gaussians

$\int p(\mathbf{y}) d\mathbf{y} = \prod_{j=1}^D \int_{-\infty}^{\infty} \frac{1}{(2\pi\lambda_j)^{1/2}} \exp\left\{-\frac{y_j^2}{2\lambda_j}\right\} dy_j = 1$ multivariate Gaussian is normalized.

$$\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu}$$

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

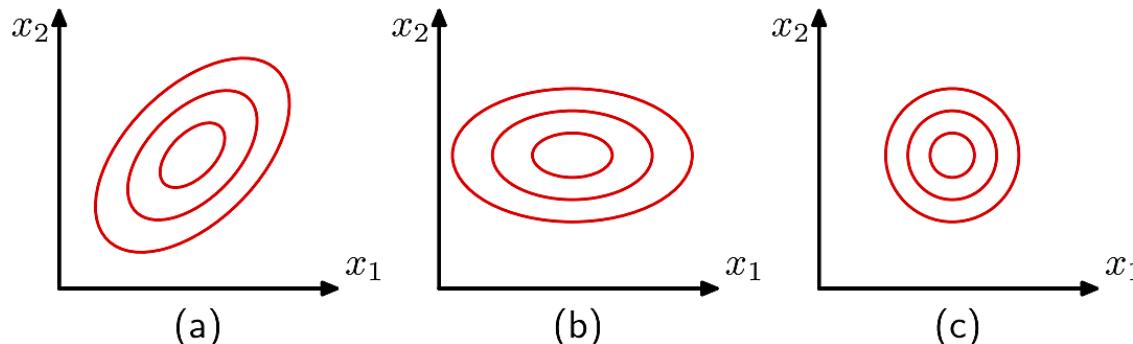
number of parameters

$$\Sigma : \frac{D(D+1)}{2} > \frac{D(D+3)}{2}$$

D grows quadratically \rightarrow High computational load

Two simplifications $\Sigma = \text{diag}(\sigma_i^2)$ diagonal no. of par = $2D$

$\Sigma = \sigma^2 \mathbf{I}$ isotropic no. of par = $D+1$



Gaussian \rightarrow Unimodal

\rightarrow multimodal \rightarrow introduce hidden variable mixture of Gaussians

Markov random field, linear dynamical system



images



tracking

Chap 8,12

Conditional Gaussian Distributions

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad \mathbf{x} = \begin{pmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{pmatrix} \quad \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{pmatrix} \quad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{pmatrix}$$

$\boldsymbol{\Lambda} \equiv \boldsymbol{\Sigma}^{-1}$: precision matrix

$$\boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{pmatrix}$$

If two sets of variables are jointly **Gaussian**, the **conditional distribution** of one set conditioned on the other is **Gaussian**

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

$$P(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad P(\mathbf{x}) = P(\mathbf{x}_a|\mathbf{x}_b)P(\mathbf{x}_b)$$

$$\begin{aligned} -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) = \\ -\frac{1}{2}(\mathbf{x}_a - \boldsymbol{\mu}_a)^T \boldsymbol{\Lambda}_{aa} (\mathbf{x}_a - \boldsymbol{\mu}_a) - \frac{1}{2}(\mathbf{x}_a - \boldsymbol{\mu}_a)^T \boldsymbol{\Lambda}_{ab} (\mathbf{x}_b - \boldsymbol{\mu}_b) \\ -\frac{1}{2}(\mathbf{x}_b - \boldsymbol{\mu}_b)^T \boldsymbol{\Lambda}_{ba} (\mathbf{x}_a - \boldsymbol{\mu}_a) - \frac{1}{2}(\mathbf{x}_b - \boldsymbol{\mu}_b)^T \boldsymbol{\Lambda}_{bb} (\mathbf{x}_b - \boldsymbol{\mu}_b). \end{aligned}$$

$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ The exponent of a Gaussian is written by

$$-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}) = -\frac{1}{2}\mathbf{x}^T \boldsymbol{\Sigma}^{-1} \mathbf{x} + \mathbf{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} + \text{const}$$

The **second-order** term is

$$-\frac{1}{2}\mathbf{x}_a^T \boldsymbol{\Lambda}_{aa} \mathbf{x}_a \implies \boxed{\boldsymbol{\Sigma}_{a|b} = \boldsymbol{\Lambda}_{aa}^{-1}}$$

The **linear** terms are $\mathbf{x}_a^T \{\boldsymbol{\Lambda}_{aa} \boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{ab}(\mathbf{x}_b - \boldsymbol{\mu}_b)\}$

$$= \mathbf{x}_a^T \boldsymbol{\Sigma}_{a|b}^{-1} \boldsymbol{\mu}_{a|b}$$

$$\implies \boxed{\begin{aligned} \boldsymbol{\mu}_{a|b} &= \boldsymbol{\Sigma}_{a|b} \{\boldsymbol{\Lambda}_{aa} \boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{ab}(\mathbf{x}_b - \boldsymbol{\mu}_b)\} \\ &= \boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{aa}^{-1} \boldsymbol{\Lambda}_{ab} (\mathbf{x}_b - \boldsymbol{\mu}_b) \end{aligned}}$$

Linear Gaussian model

$$\begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{pmatrix}$$

Marginal Gaussian distributions

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

We are integrating out \mathbf{x}_b , we find terms involve \mathbf{x}_b

$$-\frac{1}{2}\mathbf{x}_b^T \boldsymbol{\Lambda}_{bb} \mathbf{x}_b + \mathbf{x}_b^T \mathbf{m} = -\frac{1}{2}(\mathbf{x}_b - \boldsymbol{\Lambda}_{bb}^{-1} \mathbf{m})^T \boldsymbol{\Lambda}_{bb} (\mathbf{x}_b - \boldsymbol{\Lambda}_{bb}^{-1} \mathbf{m}) + \frac{1}{2} \mathbf{m}^T \boldsymbol{\Lambda}_{bb}^{-1} \mathbf{m}$$

$$\int \exp \left\{ -\frac{1}{2}(\mathbf{x}_b - \boldsymbol{\Lambda}_{bb}^{-1} \mathbf{m})^T \boldsymbol{\Lambda}_{bb} (\mathbf{x}_b - \boldsymbol{\Lambda}_{bb}^{-1} \mathbf{m}) \right\} d\mathbf{x}_b$$

integral over a
normalized Gaussian

Picking up terms depending on \mathbf{x}_a

$$\frac{1}{2} [\boldsymbol{\Lambda}_{bb} \boldsymbol{\mu}_b - \boldsymbol{\Lambda}_{ba} (\mathbf{x}_a - \boldsymbol{\mu}_a)]^T \boldsymbol{\Lambda}_{bb}^{-1} [\boldsymbol{\Lambda}_{bb} \boldsymbol{\mu}_b - \boldsymbol{\Lambda}_{ba} (\mathbf{x}_a - \boldsymbol{\mu}_a)]$$

$$\begin{aligned} & -\frac{1}{2} \mathbf{x}_a^T \boldsymbol{\Lambda}_{aa} \mathbf{x}_a + \mathbf{x}_a^T (\boldsymbol{\Lambda}_{aa} \boldsymbol{\mu}_a + \boldsymbol{\Lambda}_{ab} \boldsymbol{\mu}_b) + \text{const} \\ = & -\frac{1}{2} \mathbf{x}_a^T (\boldsymbol{\Lambda}_{aa} - \boldsymbol{\Lambda}_{ab} \boldsymbol{\Lambda}_{bb}^{-1} \boldsymbol{\Lambda}_{ba}) \mathbf{x}_a \\ & + \mathbf{x}_a^T (\boldsymbol{\Lambda}_{aa} - \boldsymbol{\Lambda}_{ab} \boldsymbol{\Lambda}_{bb}^{-1} \boldsymbol{\Lambda}_{ba})^{-1} \boldsymbol{\mu}_a + \text{const} \end{aligned}$$

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{pmatrix}$$

$$\boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{pmatrix}$$

\implies

$$\boldsymbol{\Sigma}_a = (\boldsymbol{\Lambda}_{aa} - \boldsymbol{\Lambda}_{ab} \boldsymbol{\Lambda}_{bb}^{-1} \boldsymbol{\Lambda}_{ba})^{-1} \quad \boldsymbol{\Sigma}_a (\boldsymbol{\Lambda}_{aa} - \boldsymbol{\Lambda}_{ab} \boldsymbol{\Lambda}_{bb}^{-1} \boldsymbol{\Lambda}_{ba}) \boldsymbol{\mu}_a = \boldsymbol{\mu}_a$$

Joint Distribution

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \quad \text{marginal distribution}$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{Ax} + \mathbf{b}, \mathbf{L}^{-1}) \quad \text{conditional distribution}$$

We are finding joint distribution of \mathbf{x} & \mathbf{y} , i.e. $\mathbf{z} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}$

$$\begin{aligned}\ln p(\mathbf{z}) &= \ln p(\mathbf{x}) + \ln p(\mathbf{y}|\mathbf{x}) \\ &= -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Lambda} (\mathbf{x} - \boldsymbol{\mu}) \\ &\quad -\frac{1}{2}(\mathbf{y} - \mathbf{Ax} - \mathbf{b})^T \mathbf{L} (\mathbf{y} - \mathbf{Ax} - \mathbf{b}) + \text{const}\end{aligned}$$

Quadratic function of (\mathbf{x} , \mathbf{y}) = \mathbf{z}

Picking up the **second-order** terms. We have

$$\begin{aligned}&-\frac{1}{2}\mathbf{x}^T(\boldsymbol{\Lambda} + \mathbf{A}^T \mathbf{L} \mathbf{A})\mathbf{x} - \frac{1}{2}\mathbf{y}^T \mathbf{L} \mathbf{y} + \frac{1}{2}\mathbf{y}^T \mathbf{L} \mathbf{A} \mathbf{x} + \frac{1}{2}\mathbf{x}^T \mathbf{A}^T \mathbf{L} \mathbf{y} \\ &= -\frac{1}{2} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}^T \begin{pmatrix} \boldsymbol{\Lambda} + \mathbf{A}^T \mathbf{L} \mathbf{A} & -\mathbf{A}^T \mathbf{L} \\ -\mathbf{L} \mathbf{A} & \mathbf{L} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = -\frac{1}{2} \mathbf{z}^T \mathbf{R} \mathbf{z}\end{aligned}$$

To find mean, we identify the **linear** terms of \mathbf{x} or \mathbf{y}

$$\mathbf{x}^T \boldsymbol{\Lambda} \boldsymbol{\mu} - \mathbf{x}^T \mathbf{A}^T \mathbf{L} \mathbf{b} + \mathbf{y}^T \mathbf{L} \mathbf{b} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}^T \begin{pmatrix} \boldsymbol{\Lambda} \boldsymbol{\mu} - \mathbf{A}^T \mathbf{L} \mathbf{b} \\ \mathbf{L} \mathbf{b} \end{pmatrix}$$

Joint Distribution

$$\implies \begin{aligned} \mathbb{E}[\mathbf{z}] &= \mathbf{R}^{-1} \begin{pmatrix} \Lambda\mu - \mathbf{A}^T \mathbf{L} \mathbf{b} \\ \mathbf{L} \mathbf{b} \end{pmatrix} \\ \mathbb{E}[\mathbf{z}] &= \begin{pmatrix} \mu \\ \mathbf{A}\mu + \mathbf{b} \end{pmatrix} \end{aligned}$$

We obtain

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

where

$$\Sigma = (\Lambda + \mathbf{A}^T \mathbf{L} \mathbf{A})^{-1}$$

Maximum Likelihood for the Gaussian

$$\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)^T$$

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{ND}{2} \ln(2\pi) - \frac{N}{2} \ln |\boldsymbol{\Sigma}| - \frac{1}{2} \sum_{n=1}^N (\mathbf{x}_n - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu})$$

$$\frac{\partial}{\partial \boldsymbol{\mu}} \ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu})$$

$$\boldsymbol{\mu}_{\text{ML}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$$

$$\boldsymbol{\Sigma}_{\text{ML}} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \boldsymbol{\mu}_{\text{ML}})(\mathbf{x}_n - \boldsymbol{\mu}_{\text{ML}})^T$$

joint maximization over $\boldsymbol{\mu}$ & $\boldsymbol{\Sigma}$

$$\begin{aligned}\mathbb{E}[\boldsymbol{\mu}_{\text{ML}}] &= \boldsymbol{\mu} \\ \mathbb{E}[\boldsymbol{\Sigma}_{\text{ML}}] &= \frac{N-1}{N} \boldsymbol{\Sigma}\end{aligned}$$

$$\tilde{\boldsymbol{\Sigma}} = \frac{1}{N-1} \sum_{n=1}^N (\mathbf{x}_n - \boldsymbol{\mu}_{\text{ML}})(\mathbf{x}_n - \boldsymbol{\mu}_{\text{ML}})^T$$

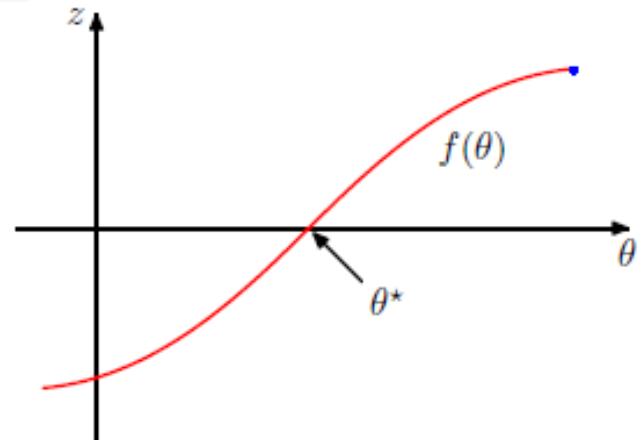
Sequential Estimation

Sequential methods allow data points to be processed one at a time and then discarded → on-line application

$$\begin{aligned}\boldsymbol{\mu}_{\text{ML}}^{(N)} &= \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \\ &= \frac{1}{N} \mathbf{x}_N + \frac{1}{N} \sum_{n=1}^{N-1} \mathbf{x}_n \\ &= \frac{1}{N} \mathbf{x}_N + \frac{N-1}{N} \boldsymbol{\mu}_{\text{ML}}^{(N-1)} \\ &= \boldsymbol{\mu}_{\text{ML}}^{(N-1)} + \frac{1}{N} (\mathbf{x}_N - \boldsymbol{\mu}_{\text{ML}}^{(N-1)})\end{aligned}$$

Robbins-Monro algorithm : consider a pair of variables θ and z governed by $p(z|\theta)$

$$f(\theta) \equiv \mathbb{E}[z|\theta] = \int z p(z|\theta) dz$$



Robbins-Monro Algorithm

Assumption : $\mathbb{E} [(z - f)^2 | \theta] < \infty$

$f(\theta) > 0$ for $\theta > \theta^*$ $f(\theta) < 0$ for $\theta < \theta^*$

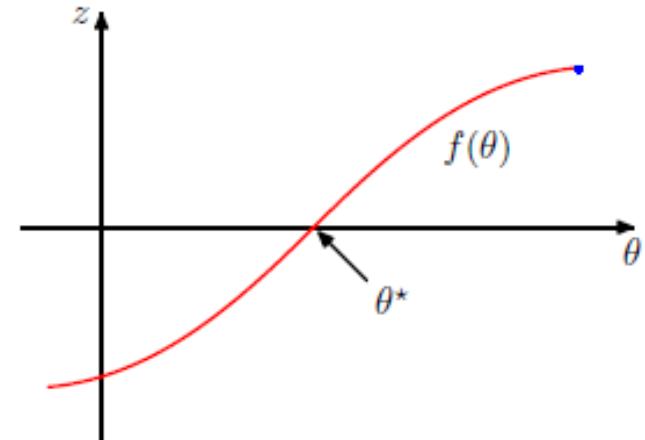
Successive estimate of θ^* :

$$\theta^{(N)} = \theta^{(N-1)} + a_{N-1} z(\theta^{(N-1)})$$

Properties of a_N : $\lim_{N \rightarrow \infty} a_N = 0$

$$\sum_{N=1}^{\infty} a_N = \infty$$

$$\sum_{N=1}^{\infty} a_N^2 < \infty$$



Convergence to the root with probability one.

How general **ML problem** can be solved sequentially using this algorithm?

$$\boxed{\left. \frac{\partial}{\partial \theta} \left\{ \frac{1}{N} \sum_{n=1}^N \ln p(\mathbf{x}_n | \theta) \right\} \right|_{\theta_{ML}} = 0}$$

Sequential ML Estimation

$$\lim_{N \rightarrow \infty} \frac{1}{N} \sum_{n=1}^N \frac{\partial}{\partial \theta} \ln p(x_n | \theta) = \mathbb{E}_x \left[\frac{\partial}{\partial \theta} \ln p(x | \theta) \right]$$

Finding ML solution \rightarrow Finding root of regression function

$$\theta^{(N)} = \theta^{(N-1)} + a_{N-1} \frac{\partial}{\partial \theta^{(N-1)}} \ln p(x_N | \theta^{(N-1)})$$

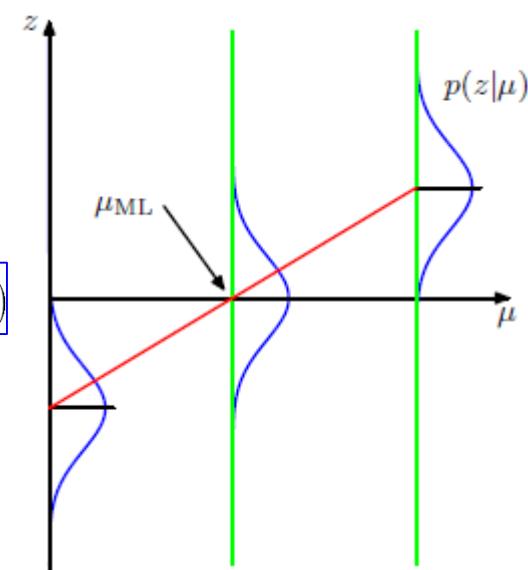
In case of estimating Gaussian mean, i.e. $\theta^{(N)} = \mu_{ML}^{(N)}$ &

$$z = \frac{\partial}{\partial \mu_{ML}} \ln p(x | \mu_{ML}, \sigma^2) = \frac{1}{\sigma^2} (x - \mu_{ML})$$



Gaussian distributed with $\mu_z = (\mu - \mu_{ML}) / \sigma^2$

If $a_N = \sigma^2/N$ we obtain $\mu_{ML}^{(N)} = \mu_{ML}^{(N-1)} + \frac{1}{N} (\mathbf{x}_N - \mu_{ML}^{(N-1)})$



Bayesian Inference for Gaussian Mean

Inferring μ given $\mathbf{X} = \{x_1, \dots, x_N\}$ & σ^2 is known

$$p(\mathbf{X}|\mu) = \prod_{n=1}^N p(x_n|\mu) = \frac{1}{(2\pi\sigma^2)^{N/2}} \exp \left\{ -\frac{1}{2\sigma^2} \sum_{n=1}^N (x_n - \mu)^2 \right\}$$

Conjugate prior: $p(\mu) = \mathcal{N}(\mu|\mu_0, \sigma_0^2)$

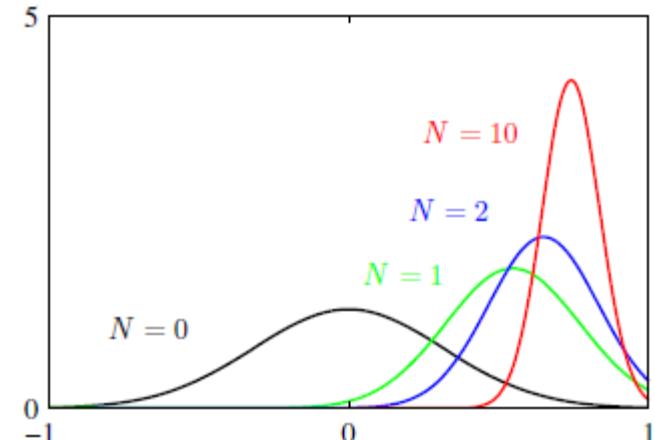
Posterior distribution : $p(\mu|\mathbf{X}) \propto p(\mathbf{X}|\mu)p(\mu)$

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

$$\begin{aligned}\mu_N &= \frac{\sigma^2}{N\sigma_0^2 + \sigma^2}\mu_0 + \frac{N\sigma_0^2}{N\sigma_0^2 + \sigma^2}\mu_{\text{ML}} \\ \frac{1}{\sigma_N^2} &= \frac{1}{\sigma_0^2} + \frac{N}{\sigma^2}\end{aligned}$$

Bayesian paradigm leads naturally to
a sequential view of the inference problem

$$p(\boldsymbol{\mu}|D) \propto \left[p(\boldsymbol{\mu}) \prod_{n=1}^{N-1} p(\mathbf{x}_n|\boldsymbol{\mu}) \right] p(\mathbf{x}_N|\boldsymbol{\mu})$$



Bayesian Inference for Gaussian Precision

Suppose mean is known, we wish to infer the variance or **precision** $\lambda \equiv 1/\sigma^2$

Likelihood : $p(\mathbf{X}|\lambda) = \prod_{n=1}^N \mathcal{N}(x_n|\mu, \lambda^{-1}) \propto \lambda^{N/2} \exp\left\{-\frac{\lambda}{2} \sum_{n=1}^N (x_n - \mu)^2\right\}$

Conjugate prior : $\text{Gam}(\lambda|a, b) = \frac{1}{\Gamma(a)} b^a \lambda^{a-1} \exp(-b\lambda)$

$$\mathbb{E}[\lambda] = \frac{a}{b} \quad \text{var}[\lambda] = \frac{a}{b^2}$$

In case of using $\text{Gam}(\lambda|a_0, b_0)$, we have

$$p(\lambda|\mathbf{X}) \propto \lambda^{a_0-1} \lambda^{N/2} \exp\left\{-b_0\lambda - \frac{\lambda}{2} \sum_{n=1}^N (x_n - \mu)^2\right\}$$

$$= \text{Gam}(\lambda|a_N, b_N)$$

$$a_N = a_0 + \frac{N}{2}$$

$$b_N = b_0 + \frac{1}{2} \sum_{n=1}^N (x_n - \mu)^2 = b_0 + \frac{N}{2} \sigma_{\text{ML}}^2$$

Inference for Gaussian Mean and Precision

Suppose both the **mean** and the **precision** are unknown.

Likelihood :

$$p(\mathbf{X}|\mu, \lambda) = \prod_{n=1}^N \left(\frac{\lambda}{2\pi} \right)^{1/2} \exp \left\{ -\frac{\lambda}{2}(x_n - \mu)^2 \right\}$$
$$\propto \left[\lambda^{1/2} \exp \left(-\frac{\lambda\mu^2}{2} \right) \right]^N \exp \left\{ \lambda\mu \sum_{n=1}^N x_n - \frac{\lambda}{2} \sum_{n=1}^N x_n^2 \right\}$$

Conjugate prior :

$$p(\mu, \lambda) \propto \left[\lambda^{1/2} \exp \left(-\frac{\lambda\mu^2}{2} \right) \right]^\beta \exp \{c\lambda\mu - d\lambda\}$$
$$= \exp \left\{ -\frac{\beta\lambda}{2}(\mu - c/\beta)^2 \right\} \lambda^{\beta/2} \exp \left\{ -\left(d - \frac{c^2}{2\beta} \right) \lambda \right\}$$

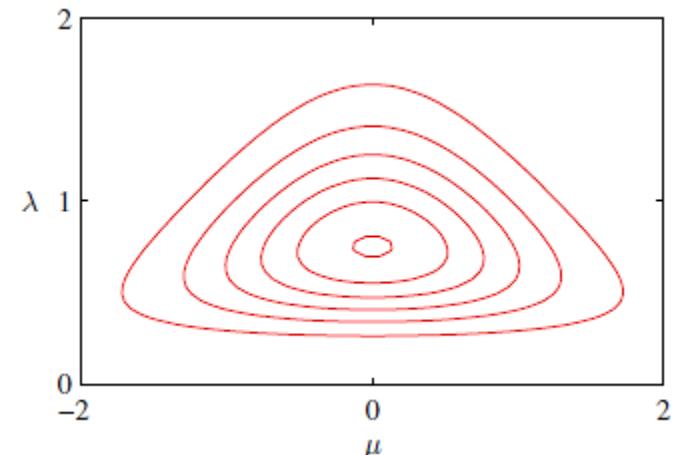
Normal-gamma distribution

$$p(\mu, \lambda) = \mathcal{N}(\mu|\mu_0, (\beta\lambda)^{-1}) \text{Gam}(\lambda|a, b)$$

$$\mu_0 = c/\beta, a = 1 + \beta/2, b = d - c^2/2\beta$$

In case of multivariate Gaussian distribution

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \quad \mathbf{x} \in R^D$$



For known mean & unknown **precision** matrix Λ ,
the conjugate prior is **Wishart** distribution

$$\mathcal{W}(\Lambda | \mathbf{W}, \nu) = B |\Lambda|^{(\nu-D-1)/2} \exp\left(-\frac{1}{2} \text{Tr}(\mathbf{W}^{-1} \Lambda)\right)$$

If both **mean** and **precision** are unknown, the conjugate prior is given by

$$p(\mu, \Lambda | \mu_0, \beta, \mathbf{W}, \nu) = \mathcal{N}(\mu | \mu_0, (\beta \Lambda)^{-1}) \mathcal{W}(\Lambda | \mathbf{W}, \nu)$$

Student's *t*-Distribution

$$\begin{aligned} p(x|\mu, a, b) &= \int_0^\infty \mathcal{N}(x|\mu, \tau^{-1}) \text{Gam}(\tau|a, b) d\tau \\ &= \int_0^\infty \frac{b^a e^{(-b\tau)} \tau^{a-1}}{\Gamma(a)} \left(\frac{\tau}{2\pi}\right)^{1/2} \exp\left\{-\frac{\tau}{2}(x-\mu)^2\right\} d\tau \\ &= \frac{b^a}{\Gamma(a)} \left(\frac{1}{2\pi}\right)^{1/2} \left[b + \frac{(x-\mu)^2}{2}\right]^{-a-1/2} \Gamma(a+1/2) \end{aligned}$$

$$\text{St}(x|\mu, \lambda, \nu) = \frac{\Gamma(\nu/2 + 1/2)}{\Gamma(\nu/2)} \left(\frac{\lambda}{\pi\nu}\right)^{1/2} \left[1 + \frac{\lambda(x-\mu)^2}{\nu}\right]^{-\nu/2-1/2}$$

$$\nu = 2a \text{ and } \lambda = a/b$$

λ :precision ν :degree of freedom

Student's t -Distribution

$$\nu \rightarrow \infty \quad \text{St}(x|\mu, \lambda, \nu) \longrightarrow \mathcal{N}(x|\mu, \lambda^{-1})$$

t -distribution is much **less sensitive** than a Gaussian to the presence of outliers
→ **robustness**

Compared to a Gaussian,
 t distribution contains the **Gaussian** as a special case.

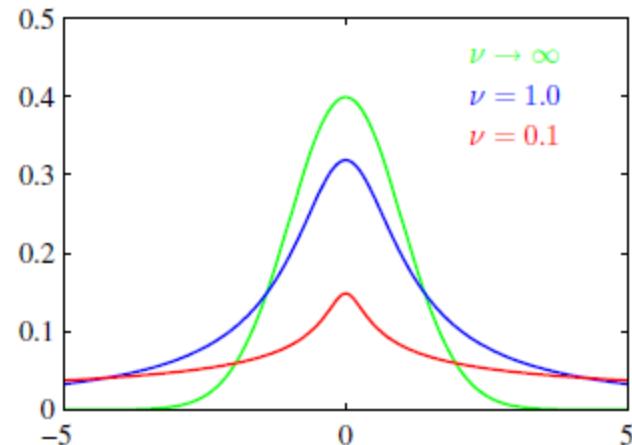
$$\text{St}(x|\mu, \lambda, \nu) = \int_0^\infty \mathcal{N}(x|\mu, (\eta\lambda)^{-1}) \text{Gam}(\eta|\nu/2, \nu/2) d\eta$$

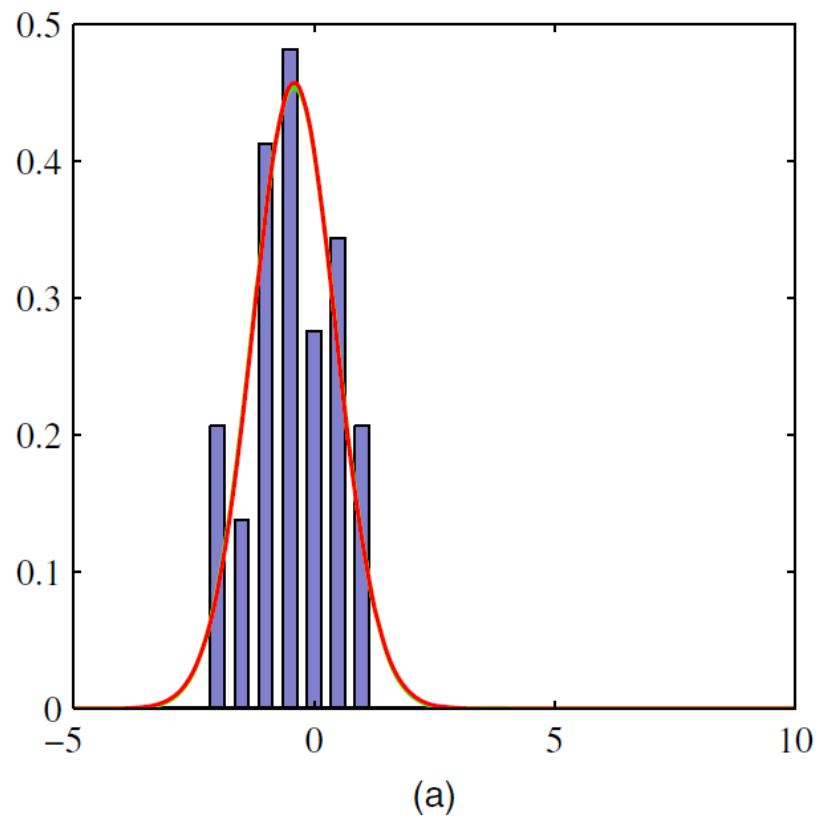
Multivariate case:

$$\text{St}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}, \nu) = \int_0^\infty \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, (\eta\boldsymbol{\Lambda})^{-1}) \text{Gam}(\eta|\nu/2, \nu/2) d\eta$$

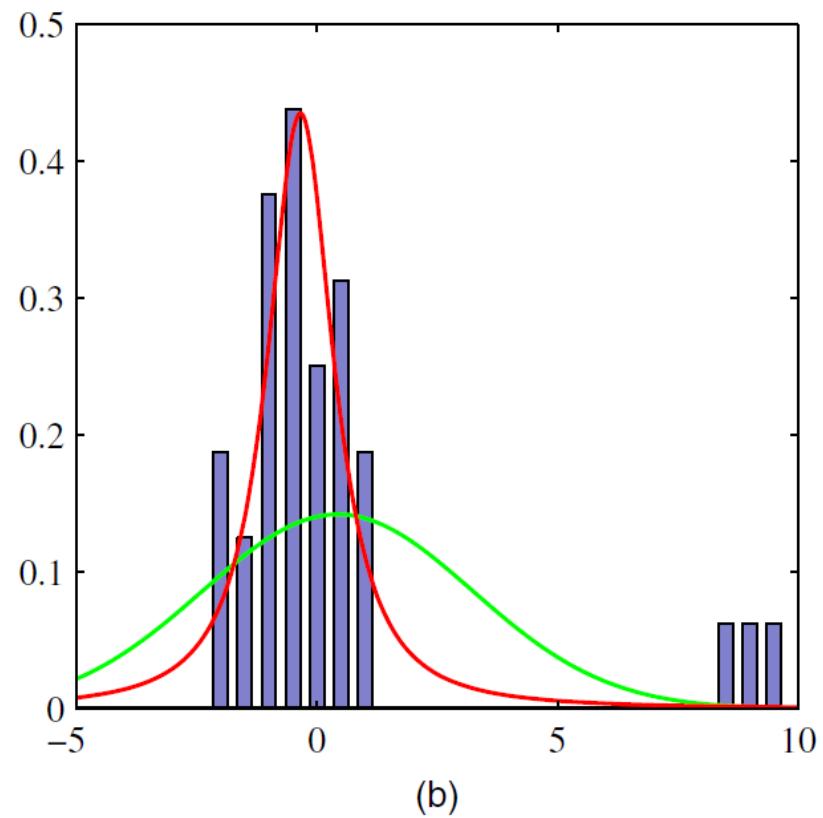
$$\text{St}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}, \nu) = \frac{\Gamma(D/2 + \nu/2)}{\Gamma(\nu/2)} \frac{|\boldsymbol{\Lambda}|^{1/2}}{(\pi\nu)^{D/2}} \left[1 + \frac{\Delta^2}{\nu} \right]^{-D/2 - \nu/2}$$

$$\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu} \quad \text{cov}[\mathbf{x}] = \frac{\nu}{(\nu - 2)} \boldsymbol{\Lambda}^{-1} \quad \text{mode}[\mathbf{x}] = \boldsymbol{\mu}$$

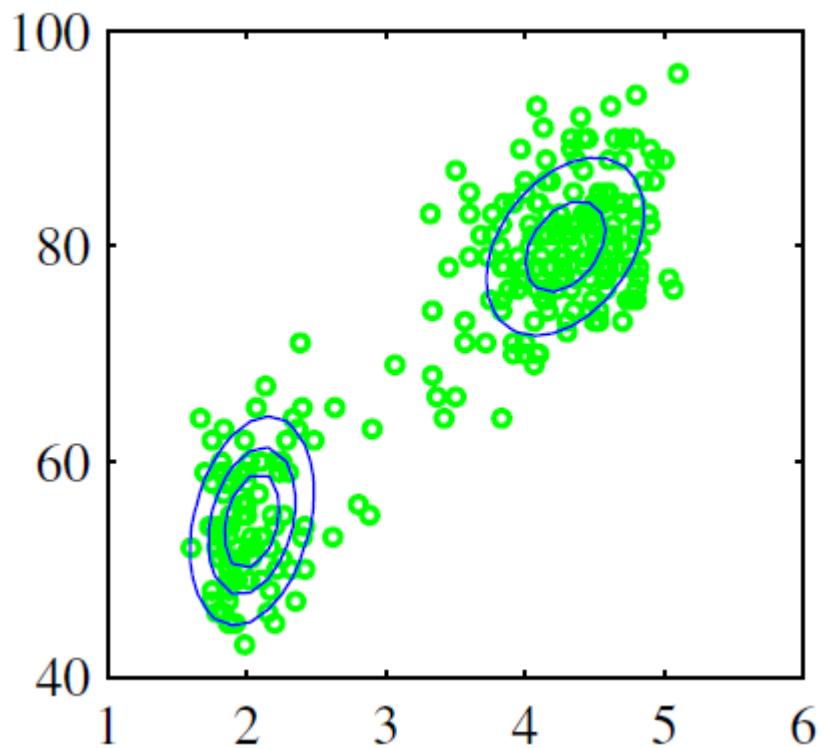
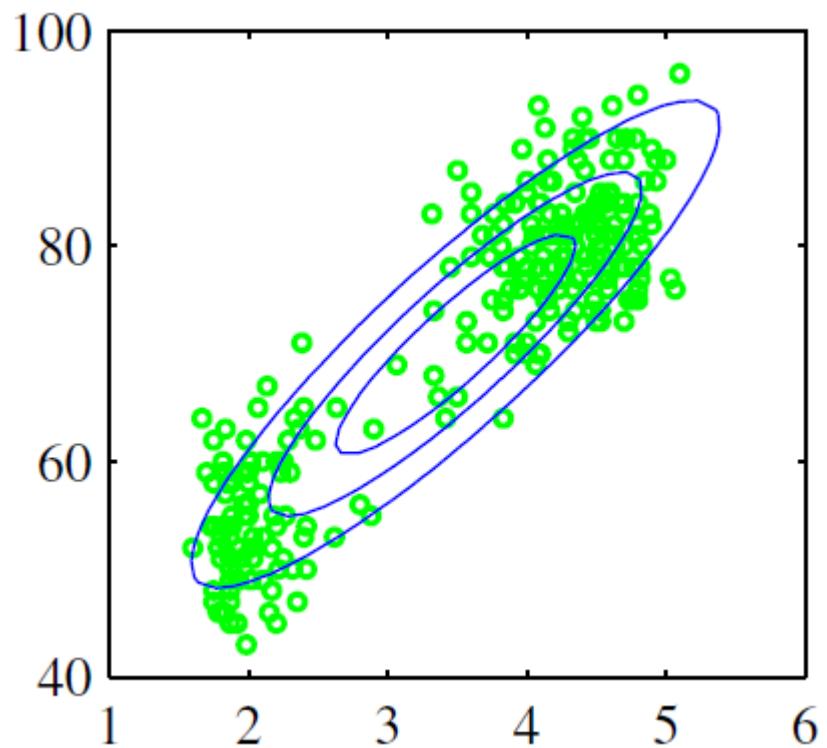




(a)



(b)



Mixtures of Gaussians

A linear combination of Gaussians can give rise very complex densities.

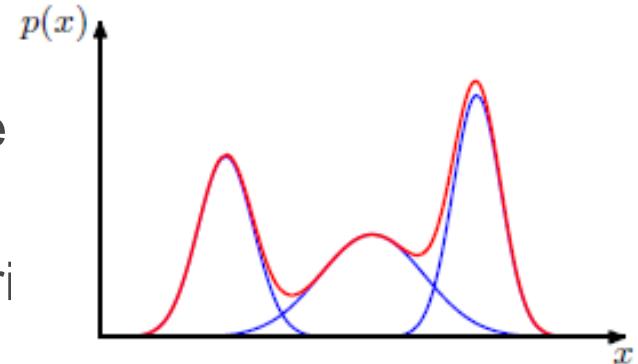
mixtures of Bernoulli distribution → Discrete vari

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

$$\sum_{k=1}^K \pi_k = 1$$

$$0 \leq \pi_k \leq 1$$

$$p(\mathbf{x}) = \sum_{k=1}^K p(k)p(\mathbf{x}|k)$$



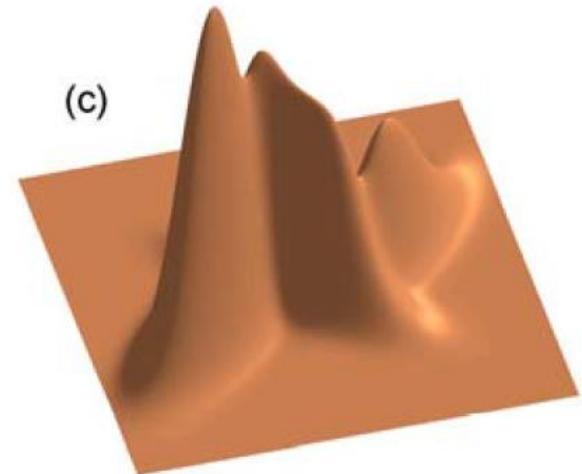
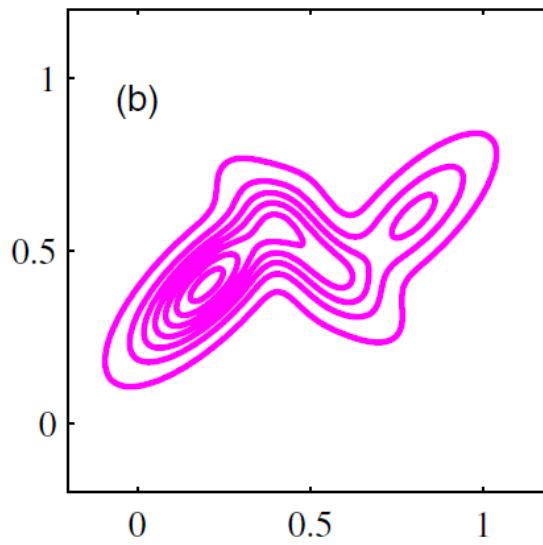
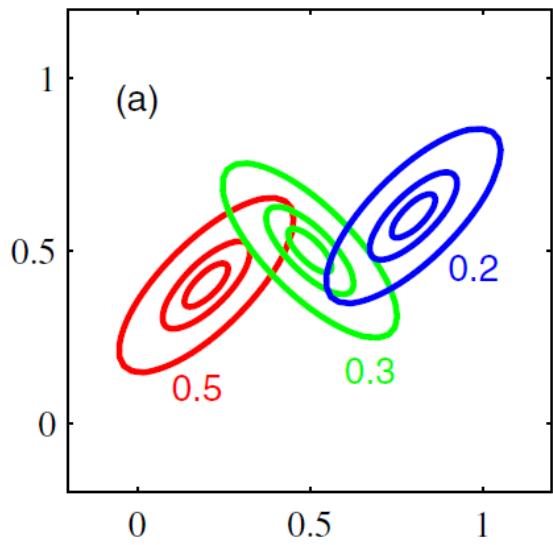
$$\Lambda = \{ \pi \equiv \{\pi_1, \dots, \pi_K\}, \boldsymbol{\mu} \equiv \{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K\}, \boldsymbol{\Sigma} \equiv \{\boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_K\} \}$$

ML: $\ln p(\mathbf{X}|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$

$$\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$$

No closed-form solution

1. iterative numerical optimization
2. expectation-maximization algorithm



The Exponential Family

$$p(\mathbf{x}|\boldsymbol{\eta}) = h(\mathbf{x})g(\boldsymbol{\eta}) \exp\{\boldsymbol{\eta}^T \mathbf{u}(\mathbf{x})\} \quad g(\boldsymbol{\eta}) \int h(\mathbf{x}) \exp\{\boldsymbol{\eta}^T \mathbf{u}(\mathbf{x})\} d\mathbf{x} = 1$$

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

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

$$\eta = \ln\left(\frac{\mu}{1-\mu}\right) \quad \sigma(\eta) = \frac{1}{1+\exp(-\eta)}$$

$$p(\mathbf{x}|\boldsymbol{\mu}) = \prod_{k=1}^M \mu_k^{x_k} = \exp\left\{\sum_{k=1}^M x_k \ln \mu_k\right\}$$

$$p(\mathbf{x}|\boldsymbol{\eta}) = \exp(\boldsymbol{\eta}^T \mathbf{x}) \quad \boldsymbol{\eta} = (\eta_1, \dots, \eta_M)^T$$

$$\mathbf{u}(\mathbf{x}) = \mathbf{x} \quad h(\mathbf{x}) = 1 \quad g(\boldsymbol{\eta}) = 1 \quad \sum_{k=1}^M \mu_k = 1$$

Or, we can arrange the multinomial distribution by

$$\begin{aligned}
& \exp \left\{ \sum_{k=1}^M x_k \ln \mu_k \right\} \\
&= \exp \left\{ \sum_{k=1}^{M-1} x_k \ln \mu_k + \left(1 - \sum_{k=1}^{M-1} x_k \right) \ln \left(1 - \sum_{k=1}^{M-1} \mu_k \right) \right\} \\
&= \exp \left\{ \sum_{k=1}^{M-1} x_k \ln \left(\frac{\mu_k}{1 - \sum_{j=1}^{M-1} \mu_j} \right) + \ln \left(1 - \sum_{k=1}^{M-1} \mu_k \right) \right\} \\
p(\mathbf{x}|\boldsymbol{\eta}) &= \left(1 + \sum_{k=1}^{M-1} \exp(\eta_k) \right)^{-1} \exp(\boldsymbol{\eta}^T \mathbf{x}) \quad \ln \left(\frac{\mu_k}{1 - \sum_j \mu_j} \right) = \eta_k \\
h(\mathbf{x}) &= 1 \quad \mu_k = \frac{\exp(\eta_k)}{1 + \sum_j \exp(\eta_j)}
\end{aligned}$$

$$\begin{aligned}
p(x|\mu, \sigma^2) &= \frac{1}{(2\pi\sigma^2)^{1/2}} \exp \left\{ -\frac{1}{2\sigma^2}(x - \mu)^2 \right\} \\
&= \frac{1}{(2\pi\sigma^2)^{1/2}} \exp \left\{ -\frac{1}{2\sigma^2}x^2 + \frac{\mu}{\sigma^2}x - \frac{1}{2\sigma^2}\mu^2 \right\} \\
&= h(\mathbf{x})g(\eta) \exp\{\eta^T \mu(\mathbf{x})\} \\
h(\mathbf{x}) &= (2\pi)^{-1/2} \quad \boldsymbol{\eta} = \begin{pmatrix} \mu/\sigma^2 \\ -1/2\sigma^2 \end{pmatrix} \quad \mathbf{u}(x) = \begin{pmatrix} x \\ x^2 \end{pmatrix} \\
g(\boldsymbol{\eta}) &= (-2\eta_2)^{1/2} \exp \left(\frac{\eta_1^2}{4\eta_2} \right)
\end{aligned}$$

Maximum Likelihood and Sufficient Statistics

$$\begin{aligned} & \nabla g(\boldsymbol{\eta}) \int h(\mathbf{x}) \exp \left\{ \boldsymbol{\eta}^T \mathbf{u}(\mathbf{x}) \right\} d\mathbf{x} \\ & + g(\boldsymbol{\eta}) \int h(\mathbf{x}) \exp \left\{ \boldsymbol{\eta}^T \mathbf{u}(\mathbf{x}) \right\} \mathbf{u}(\mathbf{x}) d\mathbf{x} = 0 \\ \implies & -\frac{1}{g(\boldsymbol{\eta})} \nabla g(\boldsymbol{\eta}) = g(\boldsymbol{\eta}) \int h(\mathbf{x}) \exp \left\{ \boldsymbol{\eta}^T \mathbf{u}(\mathbf{x}) \right\} \mathbf{u}(\mathbf{x}) d\mathbf{x} = \mathbb{E}[\mathbf{u}(\mathbf{x})] \\ & \boxed{-\nabla \ln g(\boldsymbol{\eta}) = \mathbb{E}[\mathbf{u}(\mathbf{x})]} \end{aligned}$$

Exponential family:

$$p(\mathbf{X}|\boldsymbol{\eta}) = \left(\prod_{n=1}^N h(\mathbf{x}_n) \right) g(\boldsymbol{\eta})^N \exp \left\{ \boldsymbol{\eta}^T \sum_{n=1}^N \mathbf{u}(\mathbf{x}_n) \right\}$$
$$\frac{\nabla \ln p(\mathbf{X}|\boldsymbol{\eta})}{\nabla \boldsymbol{\eta}} = 0 \implies \boxed{-\nabla \ln g(\boldsymbol{\eta}_{\text{ML}}) = \frac{1}{N} \sum_{n=1}^N \mathbf{u}(\mathbf{x}_n)}$$

Bernoulli distribution: sum of $\{x_n\}$

Gaussian distribution: sum of $\{x_n^2\}$, $\{x_n\}$ $\mathbf{u}(x) = (x, x^2)^T$

$$\lim_{N \rightarrow \infty} \frac{1}{N} \sum_{n=1}^N \mu(\mathbf{x}_n) = \mathbb{E}[\mathbf{u}(\mathbf{x})]$$

Conjugate Priors

For any member of exponential family,

$$p(\boldsymbol{\eta}|\boldsymbol{\chi}, \nu) = f(\boldsymbol{\chi}, \nu)g(\boldsymbol{\eta})^\nu \exp\left\{\nu \boldsymbol{\eta}^T \boldsymbol{\chi}\right\}$$
$$p(\boldsymbol{\eta}|\mathbf{X}, \boldsymbol{\chi}, \nu) \propto g(\boldsymbol{\eta})^{\nu+N} \exp\left\{\boldsymbol{\eta}^T \left(\sum_{n=1}^N \mathbf{u}(\mathbf{x}_n) + \nu \boldsymbol{\chi}\right)\right\}$$

Noninformative Priors

A **noninformative** prior is intended to have as **little influence** on the posterior distribution as possible. ‘letting the data speak for themselves’

$$p(x|\lambda) \quad p(\lambda) = \text{constant as uniform prior}$$

If the domain of λ is **unbounded**, such priors are called “*improper*”

$\int g(\lambda)d\lambda$ diverges prior can not be correctly normalized.

Nonlinear change of variables $\lambda = \eta^2$

$$p_\eta(\eta) = p_\lambda(\lambda) \left| \frac{d\lambda}{d\eta} \right| = p_\lambda(\eta^2) 2\eta \propto \eta$$

Examples of noninformative priors

$$p(x|\mu) = f(x - \mu) \quad \text{translation invariance}$$

if we take a shift $\hat{x} = x + c$ $p(\hat{x}|\hat{\mu}) = f(\hat{x} - \hat{\mu})$

$$\hat{\mu} = \mu + c \quad \int_A^B p(\mu) d\mu = \int_{A-c}^{B-c} p(\mu) d\mu = \int_A^B p(\mu - c) d\mu$$

if we take $\sigma_0^2 \rightarrow \infty$ Gaussian \rightarrow uniform \rightarrow noninformative prior

$$p(x|\sigma) = \frac{1}{\sigma} f\left(\frac{x}{\sigma}\right) \quad \text{scale invariance}$$

If we take a scaling $\hat{x} = cx$

$$p(\hat{x}|\hat{\sigma}) = \frac{1}{\hat{\sigma}} f\left(\frac{\hat{x}}{\hat{\sigma}}\right) \quad \text{where} \quad \hat{\sigma} = c\sigma$$

$$\int_A^B p(\sigma) d\sigma = \int_{A/c}^{B/c} p(\sigma) d\sigma = \int_A^B p\left(\frac{1}{c}\sigma\right) \frac{1}{c} d\sigma$$

$$\implies p(\sigma) = p\left(\frac{1}{c}\sigma\right) \frac{1}{c} \quad p(\sigma) \propto 1/\sigma$$

$\int_0^\infty p(\sigma) d\sigma$ diverges \rightarrow improper prior

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

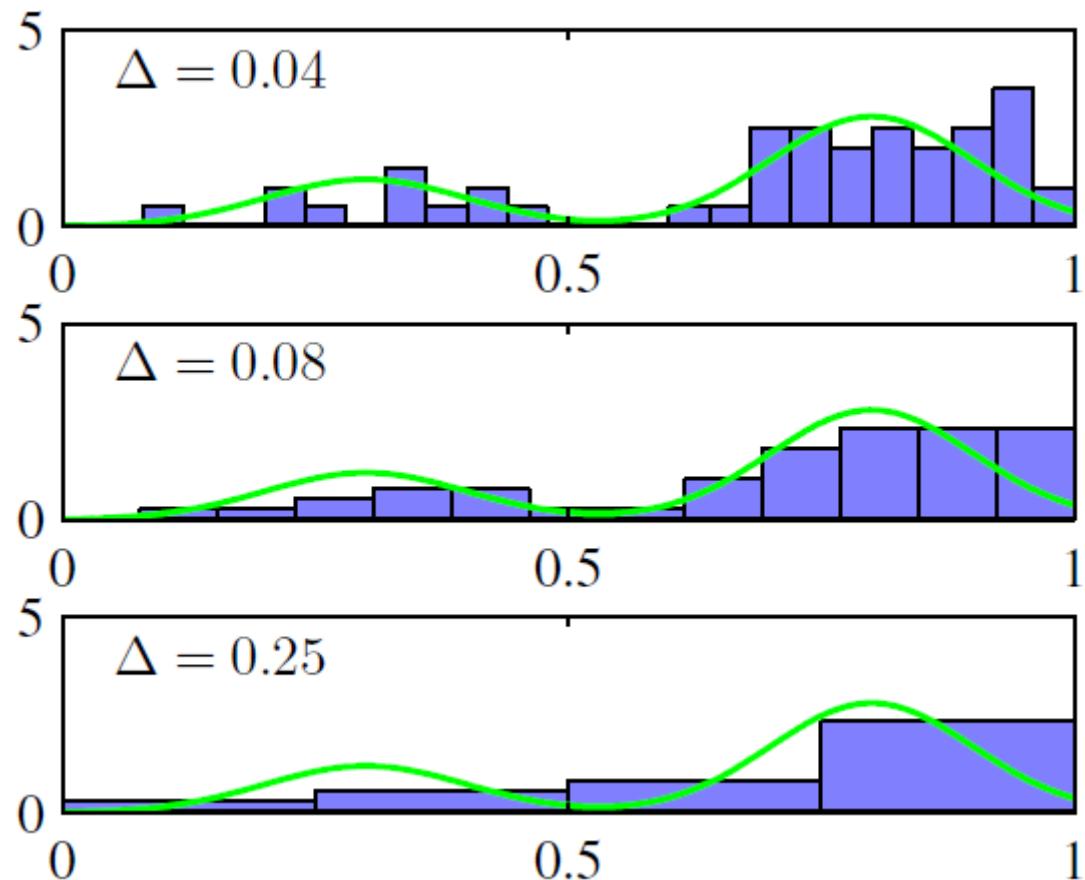
$p(\lambda) \propto 1/\lambda$ or $p(\sigma) \propto 1/\sigma$ $\lambda = 1/\sigma^2$ noninformative prior using Gamma

Nonparametric Methods

Nonparametric approaches make few assumptions

Histogram methods for density estimation

$$p_i = \frac{n_i}{N\Delta_i}$$



Kernel Density Estimators

Probability mass $P = \int_{\mathcal{R}} p(\mathbf{x}) d\mathbf{x}$

N observations K samples lie in a region R

$$\text{Bin}(K|N, P) = \frac{N!}{K!(N-K)!} P^K (1-P)^{1-K}$$

$$\mathbb{E}[K/N] = P \quad \text{var}[K/N] = P(1-P)/N$$

If $N \nearrow K \simeq NP$, R is sufficiently small. $p(\mathbf{x})$ is roughly constant over R

$$P \simeq p(\mathbf{x})V$$

$\implies p(\mathbf{x}) = \frac{K}{NV}$ If R is sufficiently small, $p(\mathbf{x})$ means the density

(i) Fixed K , we determine the value of $V \Rightarrow K$ -nearest-neighbor technique

(ii) Fixed V , we find $K \Rightarrow$ Kernel approach

If $N \rightarrow \infty$, either (i) or (ii) converges to the true $p(x)$

Kernel (density) method : R is a small hypercube centered on x

We want to count the number K of points falling within R using the kernel function

$$k(\mathbf{u}) = \begin{cases} 1, & |u_i| \leq 1/2, \\ 0, & \text{otherwise} \end{cases} \quad i = 1, \dots, D$$

Parzen window

$$\Rightarrow K = \sum_{n=1}^N k\left(\frac{\mathbf{x} - \mathbf{x}_n}{h}\right)$$

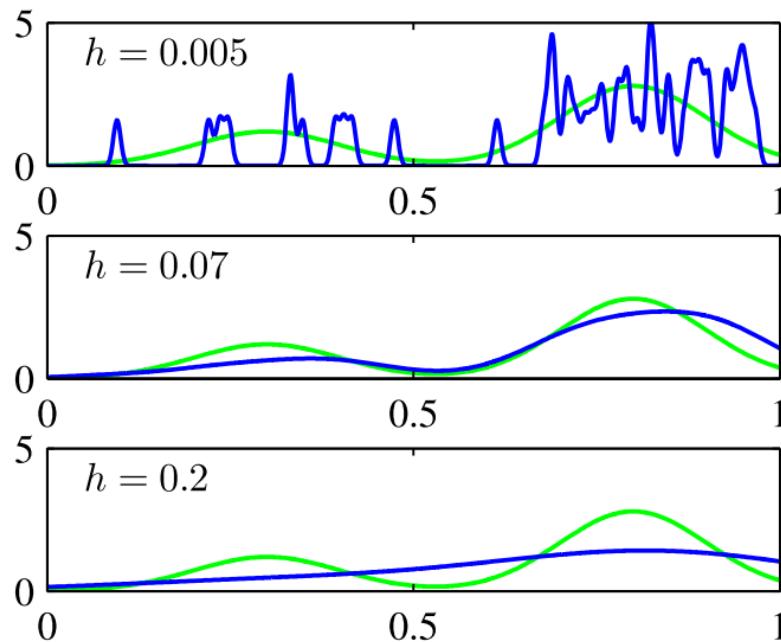
$$p(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^N \frac{1}{h^D} k\left(\frac{\mathbf{x} - \mathbf{x}_n}{h}\right)$$

(Parzen estimator)

Kernel density estimator suffers from the artificial discontinuities in cube boundaries for smoothing the density model, we can use Gaussian Kernel function

$$p(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^N \frac{1}{(2\pi h^2)^{1/2}} \exp\left\{-\frac{\|\mathbf{x} - \mathbf{x}_n\|^2}{2h^2}\right\}$$

Kernel Density Estimators



We can choose any other Kernel function $k(\mathbf{u})$ subject to

$$k(\mathbf{u}) \geq 0 \quad \int k(\mathbf{u}) d\mathbf{u} = 1$$

No training phase is involved because only storage of training sample is required. Computation grows linearly with size N .

Nearest-Neighbour Methods

One of the difficulties with the kernel density estimation is the fixing of h . Large h will lead to over-smoothing and a washing out of structure.

⇒ Fix K & Find V

But, optimum choice of K is also a problem.

K nearest-neighbor density estimation ⇒

K nearest-neighbor classification

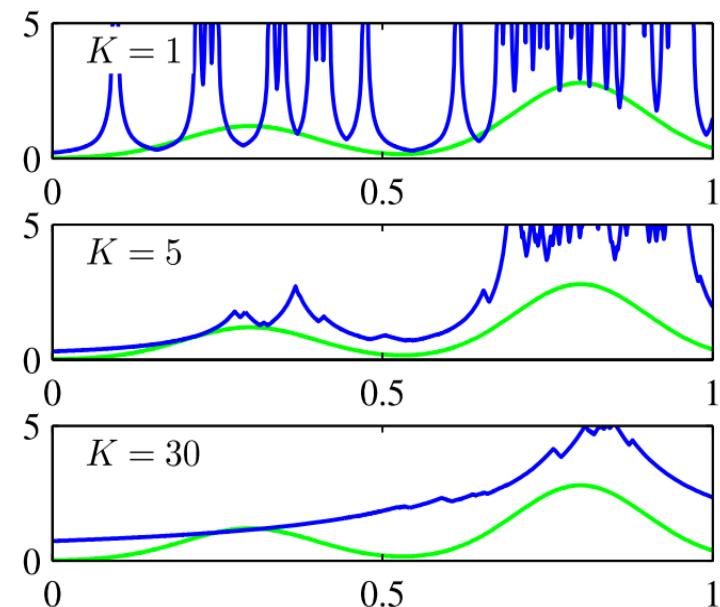
$$\sum N_k = N \quad \sum K_k = K \quad N_k, K_k \in C_k$$

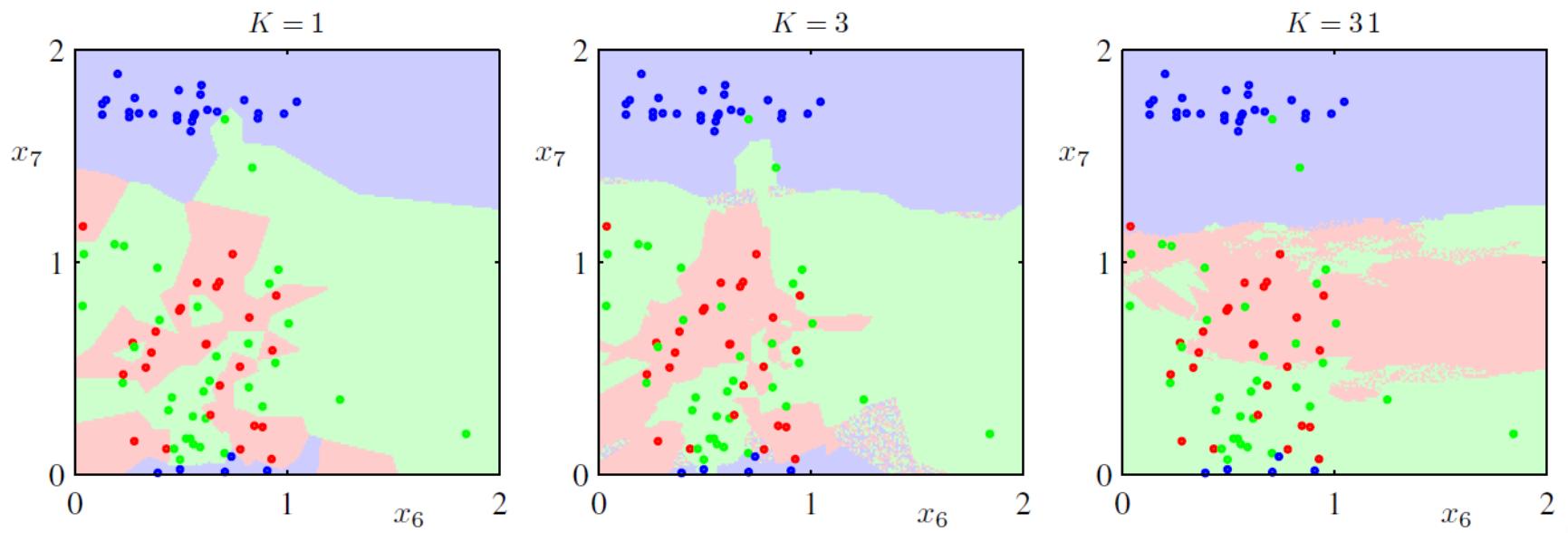
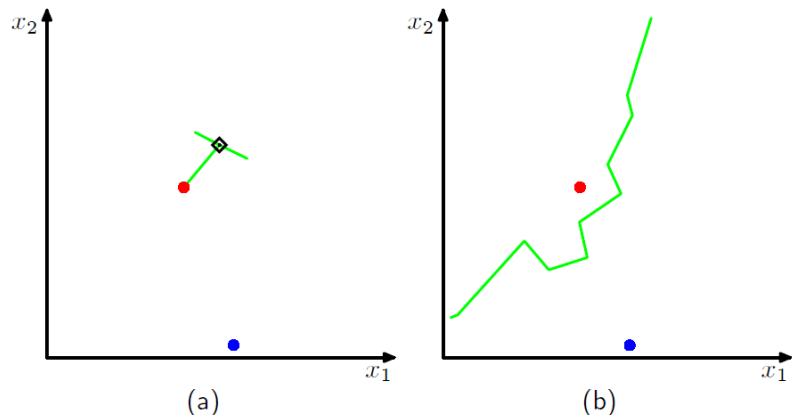
$$p(\mathbf{x}|\mathcal{C}_k) = \frac{K_k}{N_k V} \quad p(\mathbf{x}) = \frac{K}{NV}$$

$$p(\mathcal{C}_k) = \frac{N_k}{N} \quad p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})} = \frac{K_k}{K}$$

Entire training data should be stored.

$$[\text{ErrorRate}]_{K=NN(K=1)} \leq 2[\text{ErrorRate}]_{\text{optimalBayes}}$$





Entire training data should be stored.

$$[\text{ErrorRate}]_{K-NN(K=1)} \leq 2[\text{ErrorRate}]_{\text{optimalBayes}}$$

CONTENTS

1. Introduction
2. Probability Distributions
3. Linear Models for Regression
4. Linear Models for Classification
5. Kernel Methods
6. Sparse Kernel Methods
7. Mixture Models and EM
8. Approximate Inference

Linear Models for Regression

- Unsupervised learning \Rightarrow **Supervised** learning
- Input variables \rightarrow Target variables

$$\{x_n\} \quad \{t_n\} \quad n = 1, \dots, N$$

- We are finding $y(\mathbf{x}) \rightarrow t$ & even the **predictive distribution**

Linear Basis Function Models

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \dots + w_D x_D \quad \mathbf{x} = (x_1, \dots, x_D)^T$$

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

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

- Linear combinations of fixed nonlinear functions where $\phi_0(\mathbf{x}) = 1$
 $\{\phi_j(\mathbf{x})\}$ can be interpreted as the process of **feature extraction**.
- Linear models using **nonlinear basis** model.

- Polynomial regression is built using basis function I . $\phi_j(x) = x^j$
- Also, we can use II. $\phi_j(x) = \exp\left\{-\frac{(x - \mu_j)^2}{2s^2}\right\}$ changes in one region will affect all other regions.

μ_j :location of basis function in input space

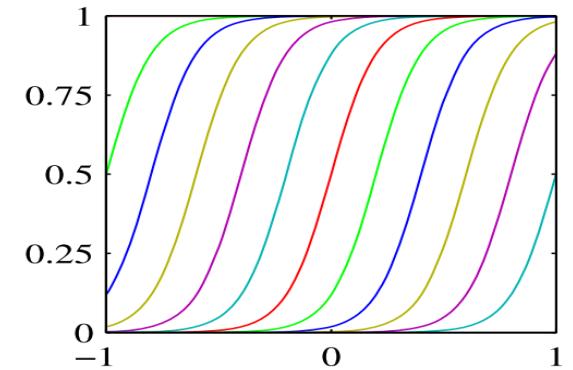
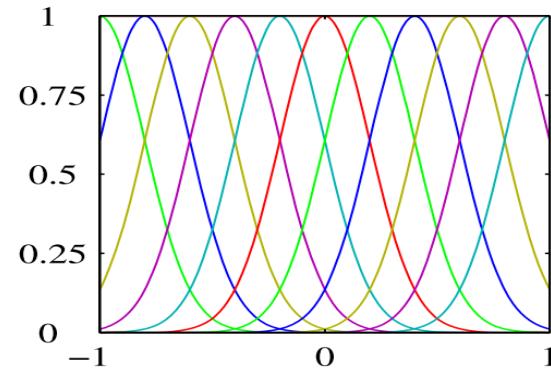
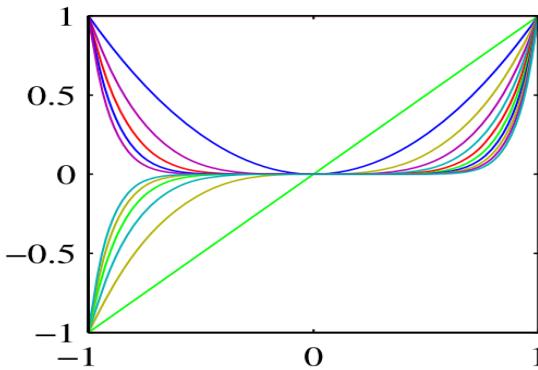
II. **Gaussian** basis function with spatial scale S

III. **Sigmoidal** basis function

$$\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right)$$

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

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



Maximum Likelihood and Least Squares

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon \quad \epsilon \sim N(O, \beta^{-1})$$

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

→ $\mathbb{E}[t|\mathbf{x}] = \int tp(t|\mathbf{x}) dt = y(\mathbf{x}, \mathbf{w})$

- Input data $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ Target variables $t = \{t_1, \dots, t_N\}$

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

$$\begin{aligned} \ln p(\mathbf{t}|\mathbf{w}, \beta) &= \sum_{n=1}^N \ln \mathcal{N}(t_n | \mathbf{w}^T \phi(\mathbf{x}_n), \beta^{-1}) \\ &= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E_D(\mathbf{w}) \quad E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 \end{aligned}$$

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

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

normal equations for the least squares problem

where

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

- If we make the bias parameter explicit, the error function becomes

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \left\{ t_n - w_0 - \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}_n) \right\}^2$$

$$\frac{\partial E_D(w_0)}{\partial w_0} = 0 \implies w_0 = \bar{t} - \sum_{j=1}^{M-1} w_j \bar{\phi}_j$$

$$\bar{t} = \frac{1}{N} \sum_{n=1}^N t_n, \quad \bar{\phi}_j = \frac{1}{N} \sum_{n=1}^N \phi_j(\mathbf{x}_n)$$

w_0 compensates for the difference between \bar{t} and the weighted sum of $\bar{\phi}_j$

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

Geometry of Least Squares

$$t \rightarrow y$$

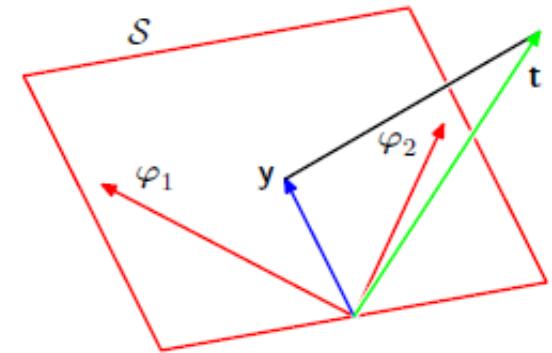
y is an “orthogonal projection” of data

vector t onto the subspace spanned by $\phi_j(\mathbf{x})$

$\Phi^T \Phi$ is close to singular

→ degeneracy → SVD

→ regularization term ensures nonsingular matrix.



Sequential Learning

- We can use “*stochastic gradient descent*” or “*sequential gradient descent*”
- Let error function $E = \sum_n E_n$
- After presentation of pattern n ,

$$\begin{aligned} \mathbf{w}^{(\tau+1)} &= \mathbf{w}^{(\tau)} - \eta \nabla E_n \\ &= \mathbf{w}^{(\tau)} + \eta(t_n - \mathbf{w}^{(\tau)T} \phi_n) \phi_n \end{aligned}$$

$$\phi_n = \phi(\mathbf{x}_n)$$

least-mean-squares or the *LMS algorithm*

Regularized Least Squares

- To control **over-fitting**, we minimize

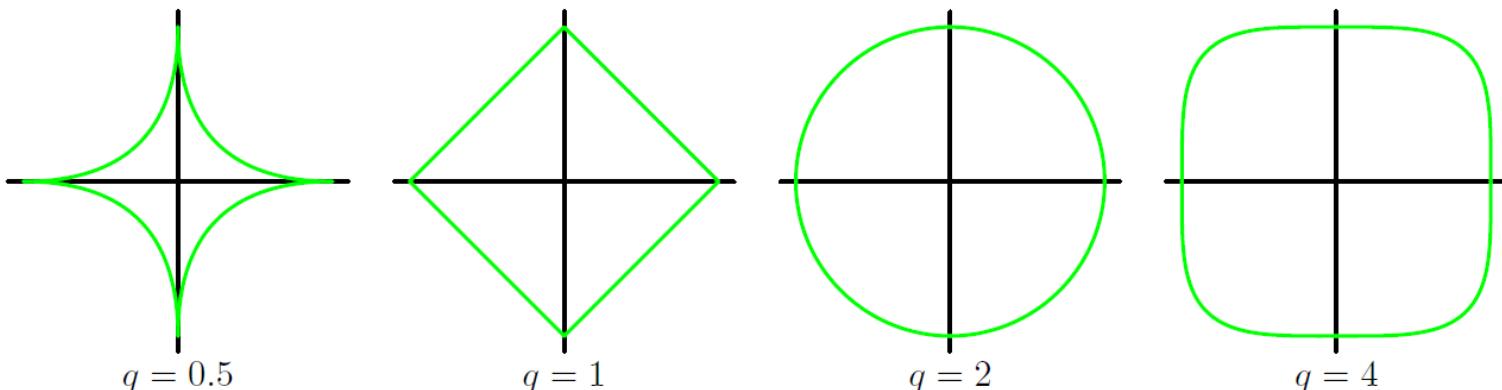
$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

$E_W(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$: weight decay
parameter shrinkage

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2$$

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

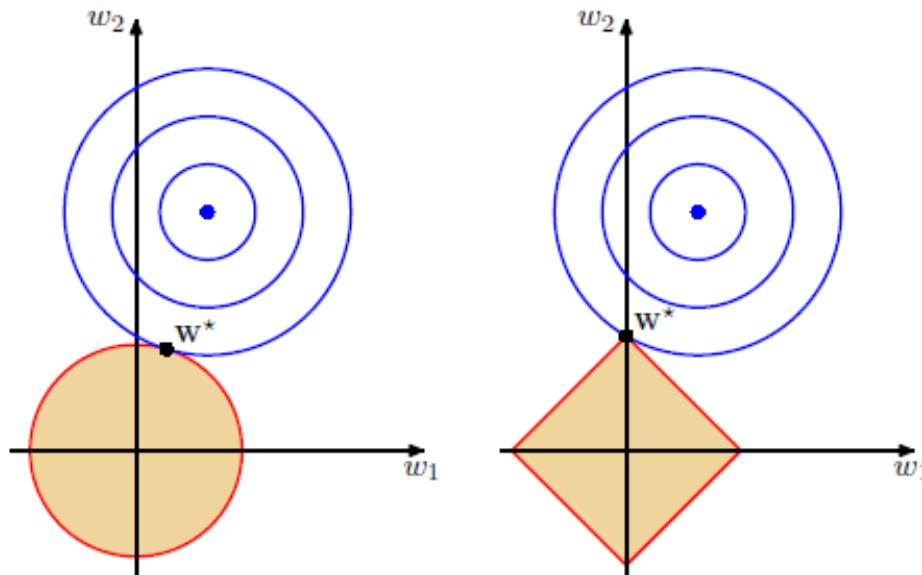
- Sometimes, we use $E_W(\mathbf{w}) = \frac{1}{2} \sum_{j=1}^M |w_j|^q$
- $q = 1$: *lasso regularizer*
 $q = 2$: *quadratic regularizer*



Lasso Regularizer

- In case $q = 1$ & λ is sufficiently large, some w_j vanishes & **sparse** model happens
- We should minimize the unregularized sum-of-squares error subject to

$$\sum_{j=1}^M |w_j|^q \leq \eta$$



Linear Regression with Multiple Outputs

- target variables $t \in R^K$

$$\mathbf{y}(\mathbf{x}, \mathbf{w}) = \mathbf{W}^T \phi(\mathbf{x})$$

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

$$\phi(\mathbf{x}) \in R^M$$

- Given $T = \{t_1, \dots, t_N\} \& \{x_1, \dots, x_N\}$, the **log likelihood** function is given by

$$\ln p(\mathbf{T}|\mathbf{X}, \mathbf{W}, \beta) = \sum_{n=1}^N \ln \mathcal{N}(\mathbf{t}_n | \mathbf{W}^T \phi(\mathbf{x}_n), \beta^{-1}) = \frac{NK}{2} \ln \left(\frac{\beta}{2\pi} \right) - \frac{\beta}{2} \sum_{n=1}^N \| \mathbf{t}_n - \mathbf{W}^T \phi(\mathbf{x}_n) \|^2$$

$$\mathbf{W}_{ML} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{T}$$

This can be extended to general Gaussian noise with **full covariance matrix**.

The Bias-Variance Decomposition

- Model **regularization** is a critical issue.
- We consider a frequentist viewpoint of **model complexity**.
 \Rightarrow “**bias-variance trade-off**”
- Considering loss function for regression,
optimal prediction is obtained by See Page 46

$$h(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}] = \int tp(t|\mathbf{x}) dt$$

The **expected squared loss** can be written by

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x} + \int \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt$$

Given the data set D ,

$$\{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2 =$$

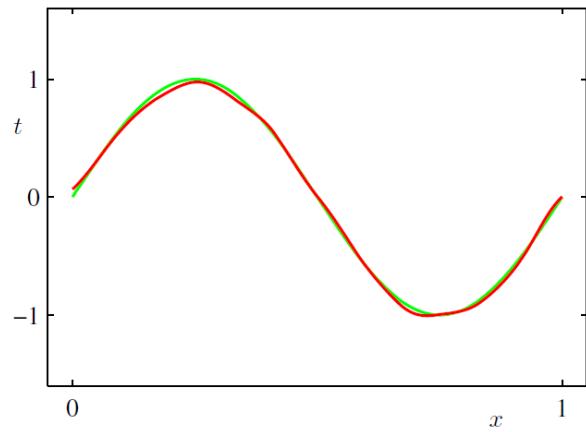
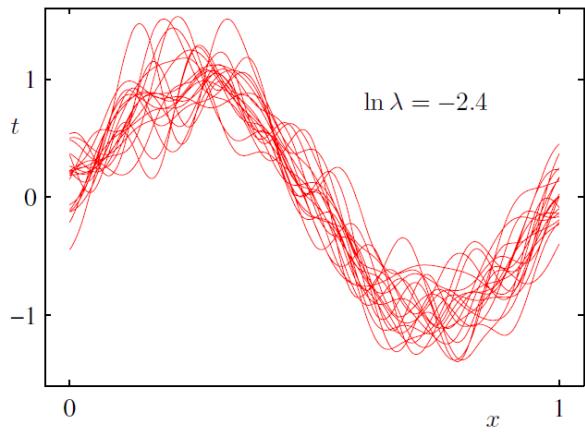
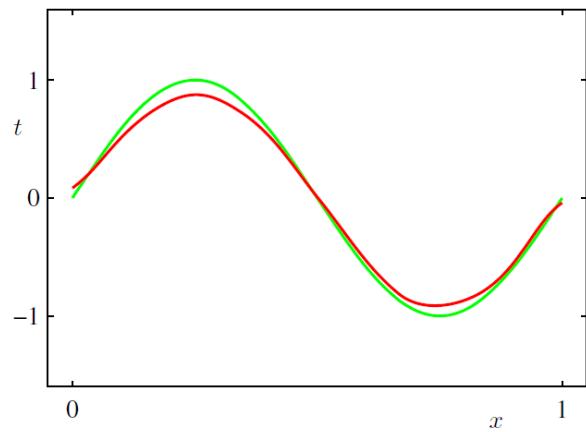
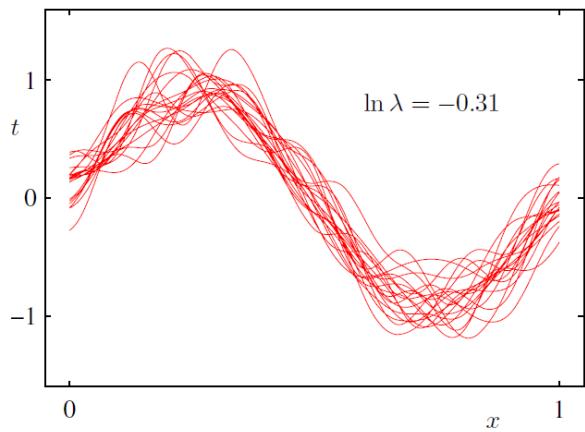
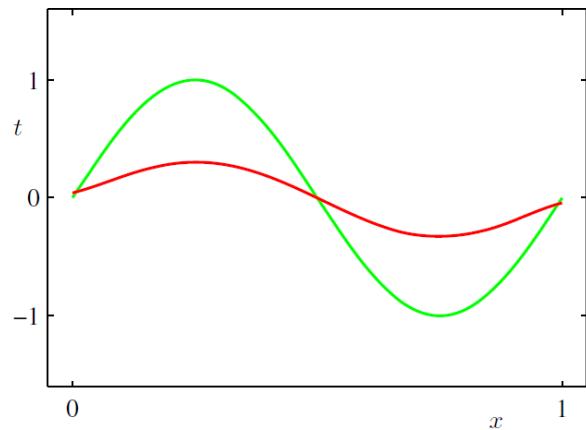
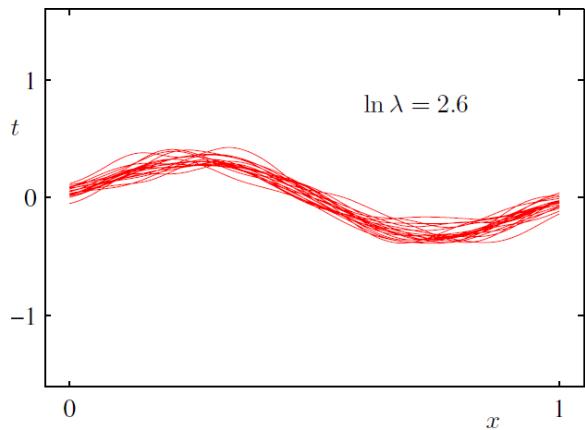
$$\begin{aligned} & \{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] + \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 \\ &= \{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2 + \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 \\ &\quad + 2\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}. \end{aligned}$$

$$\begin{aligned} \mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2] \\ = \underbrace{\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2}_{\text{(bias)}^2} + \underbrace{\mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2]}_{\text{variance}} \end{aligned}$$

$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise}$$

- Our goal is to minimize the expected loss. There is a trade-off between bias and variance.
 - $\lambda \downarrow$ Flexible model has **low bias & high variance**.
 - $\lambda \uparrow$ Rigid model has **high bias & low variance**.
- Optimal predictive model should find the best balance between bias and variance .

$$\frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}$$



- The result of **averaging many solution** for the ‘complex model’ is a very good fit to the regression function.

Average prediction

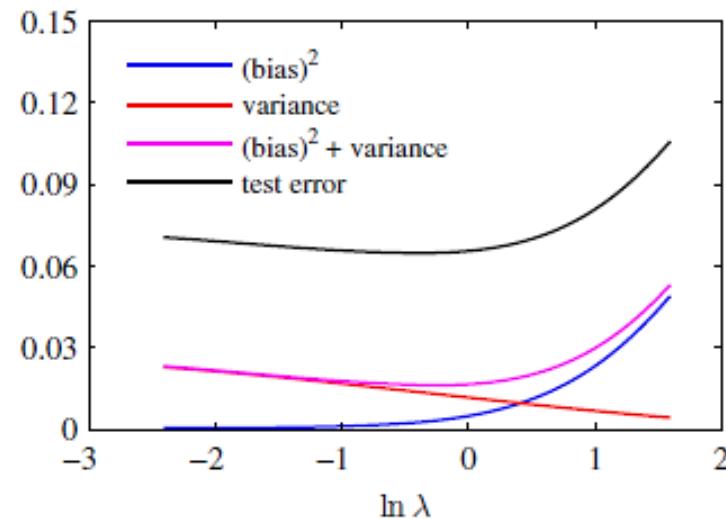
$$\bar{y}(x) = \frac{1}{L} \sum_{l=1}^L y^{(l)}(x)$$

$$(\text{bias})^2 = \int \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x}$$

$$\text{variance} = \int \mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2] p(\mathbf{x}) d\mathbf{x}$$

Bias-variance decomposition

↔ model complexity



- However, we only have the single observed data set.

Bayesian Linear Regression

- Number of basis functions should be dependent on number of data size.
- Using held-out data can be computationally expensive and wasteful of valuable data. Bayesian treatment using only training data.

Parameter Distribution

- $p(\mathbf{t}|\mathbf{w})$ has the exponential of a quadratic function of \mathbf{w} .
- Conjugate prior for \mathbf{w} is given by

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

$$p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{t}|\mathbf{w})p(\mathbf{w}) = N(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$$

$$\begin{aligned}\mathbf{m}_N &= \mathbf{S}_N (\mathbf{S}_0^{-1} \mathbf{m}_0 + \beta \Phi^T \mathbf{t}) \\ \mathbf{S}_N^{-1} &= \mathbf{S}_0^{-1} + \beta \Phi^T \Phi.\end{aligned}$$

To simplify the treatment, we consider a zero-mean isotropic Gaussian as a prior

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1} \mathbf{I})$$

Parameter Distribution

- Posterior distribution has the form $p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$

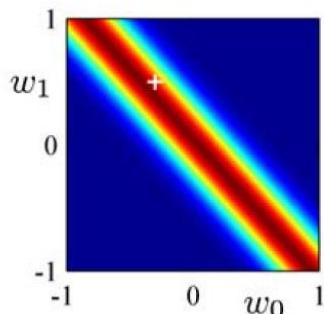
$$\begin{aligned}\mathbf{m}_N &= \beta \mathbf{S}_N \Phi^T \mathbf{t} \\ \mathbf{S}_N^{-1} &= \alpha \mathbf{I} + \beta \Phi^T \Phi\end{aligned}$$

- We have $\ln p(\mathbf{w}|\mathbf{t}) = -\frac{\beta}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 - \frac{\alpha}{2} \mathbf{w}^T \mathbf{w} + \text{const}$
- Maximization of this posterior distribution is equivalent to the minimization of the sum-of-squares error with a **quadratic regularization** term with $\lambda = \alpha/\beta$.
- Figure 3.7 Illustrates for **sequential Bayesian learning**
- Gaussian prior can be extended to

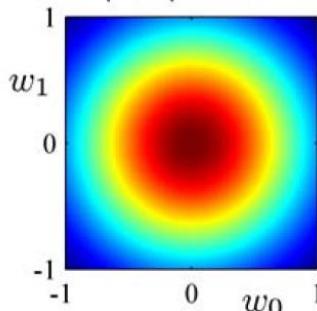
$$p(\mathbf{w}|\alpha) = \left[\frac{q}{2} \left(\frac{\alpha}{2} \right)^{1/q} \frac{1}{\Gamma(1/q)} \right]^M \exp \left(-\frac{\alpha}{2} \sum_{j=1}^M |w_j|^q \right)$$

likelihood

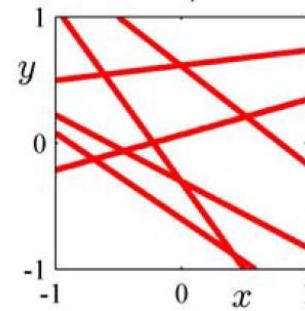
$$p(t|x, \mathbf{w})$$



prior/posterior



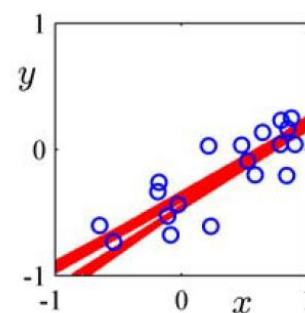
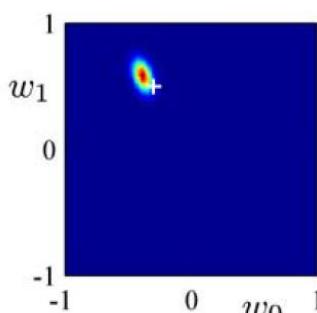
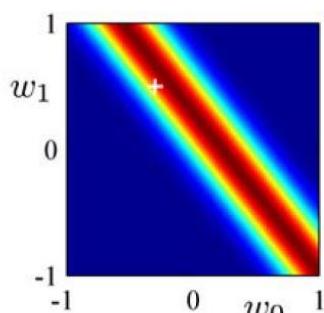
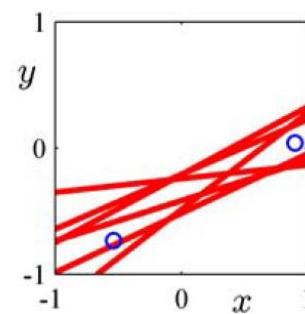
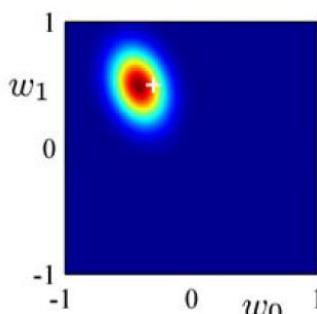
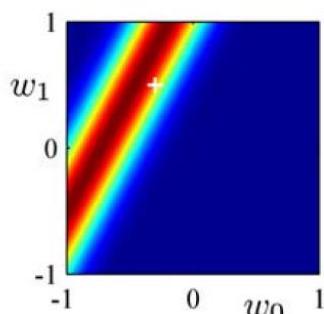
data space



$$f(x, \mathbf{a}) = a_0 + a_1 x$$

$$a_0 = -0.3 \text{ and } a_1 = 0.5$$

$$\text{U}(x|-1, 1)$$



Predictive Distribution

- We are interested in predicting t for new x , predictive distribution is given by

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

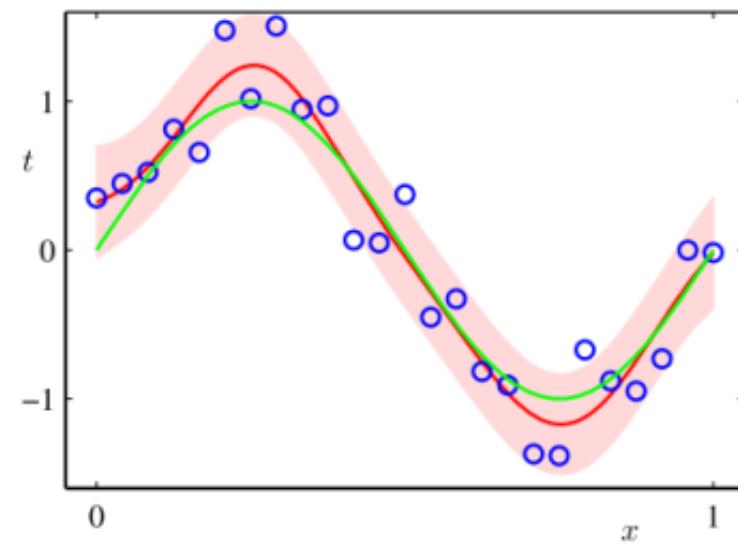
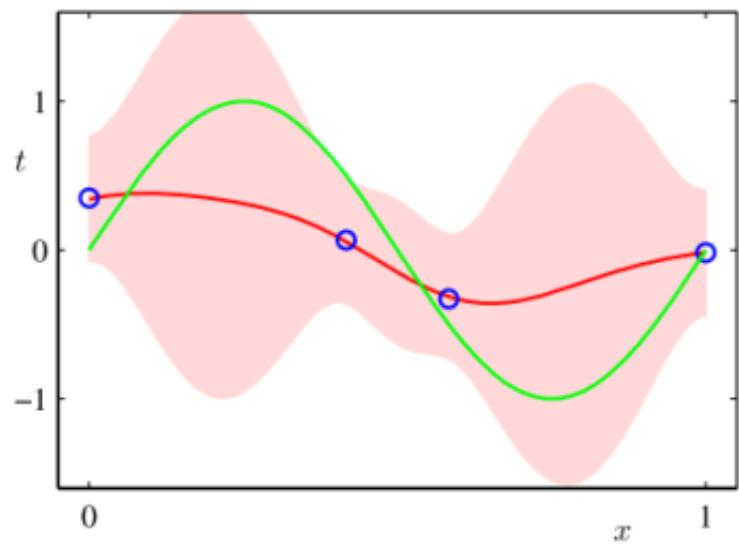
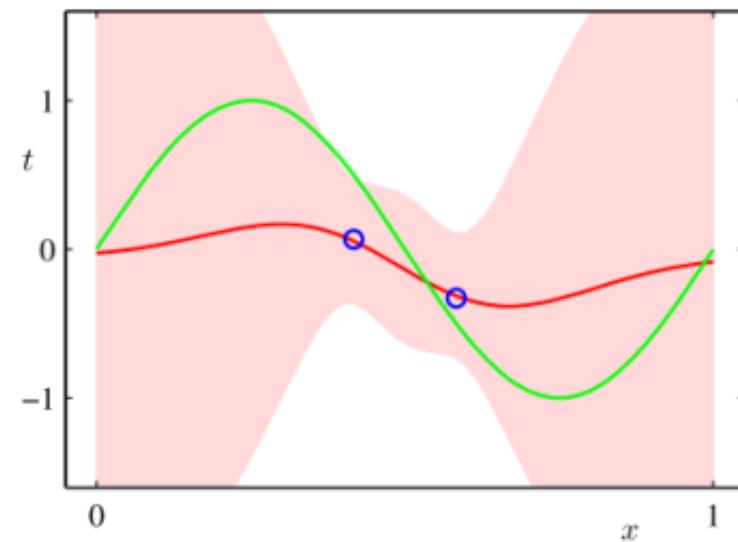
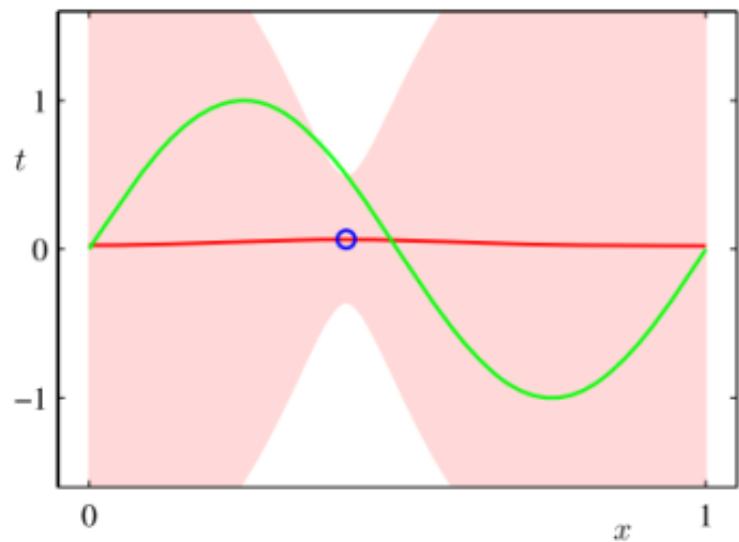
- We obtain $p(t|\mathbf{x}, \mathbf{t}, \alpha, \beta) = \mathcal{N}(t|\mathbf{m}_N^T \boldsymbol{\phi}(\mathbf{x}), \sigma_N^2(\mathbf{x}))$

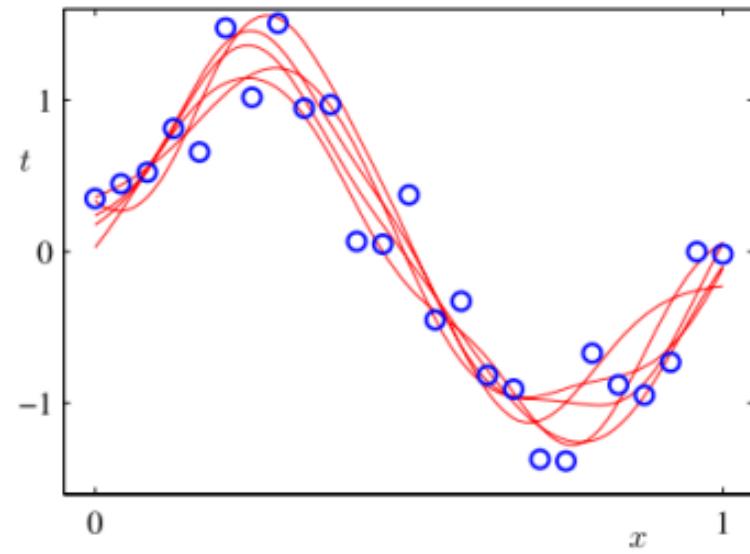
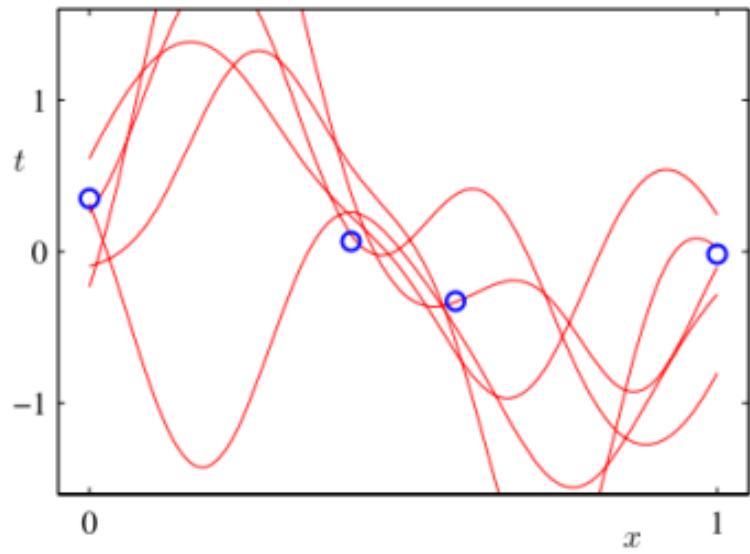
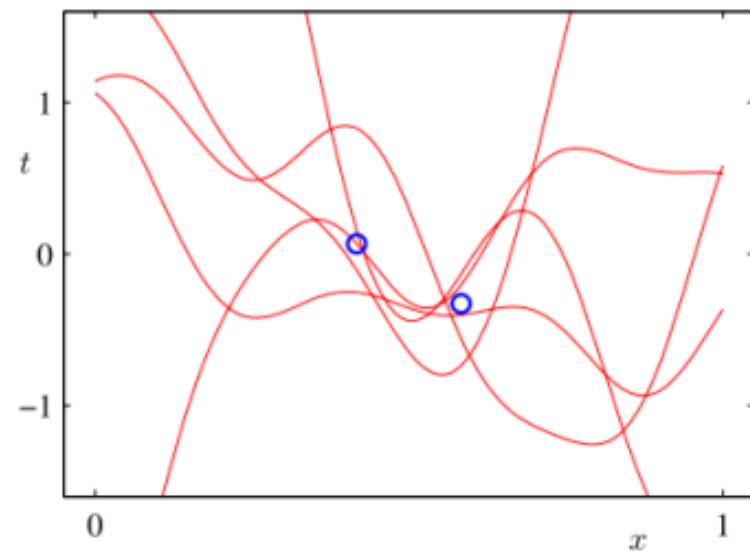
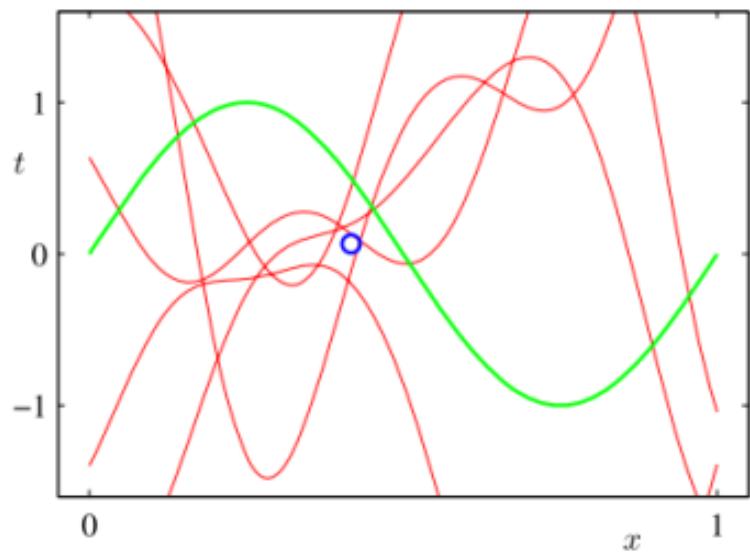
where

$$\sigma_N^2(\mathbf{x}) = \frac{1}{\beta} + \boldsymbol{\phi}(\mathbf{x})^T \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x})$$

$$\sigma_{N+1}^2(\mathbf{x}) \leq \sigma_N^2(\mathbf{x}) \text{ when } N \rightarrow \infty \quad \boldsymbol{\phi}^T(\mathbf{x}) \mathbf{S}_n \boldsymbol{\phi}(\mathbf{x}) \rightarrow 0$$

- See Figures **red curve** shows the **mean** of predictive distribution. **red shaded region** implies the **variance** or uncertainty.



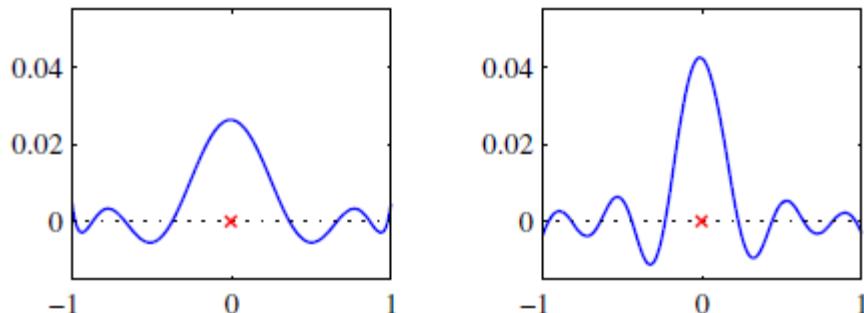


Equivalent Kernel

- By substituting $M_N(W_{map})$, the predictive mean is written by

$$y(\mathbf{x}, \mathbf{m}_N) = \mathbf{m}_N^T \phi(\mathbf{x}) = \beta \phi(\mathbf{x})^T \mathbf{S}_N \Phi^T \mathbf{t} = \sum_{n=1}^N \beta \phi(\mathbf{x})^T \mathbf{S}_N \phi(\mathbf{x}_n) t_n = \sum_{n=1}^N k(\mathbf{x}, \mathbf{x}_n) t_n$$

- See Fig 3.10 for different \mathbf{x} & \mathbf{x}'
- We should weight local evidence more strongly than distant evidence.

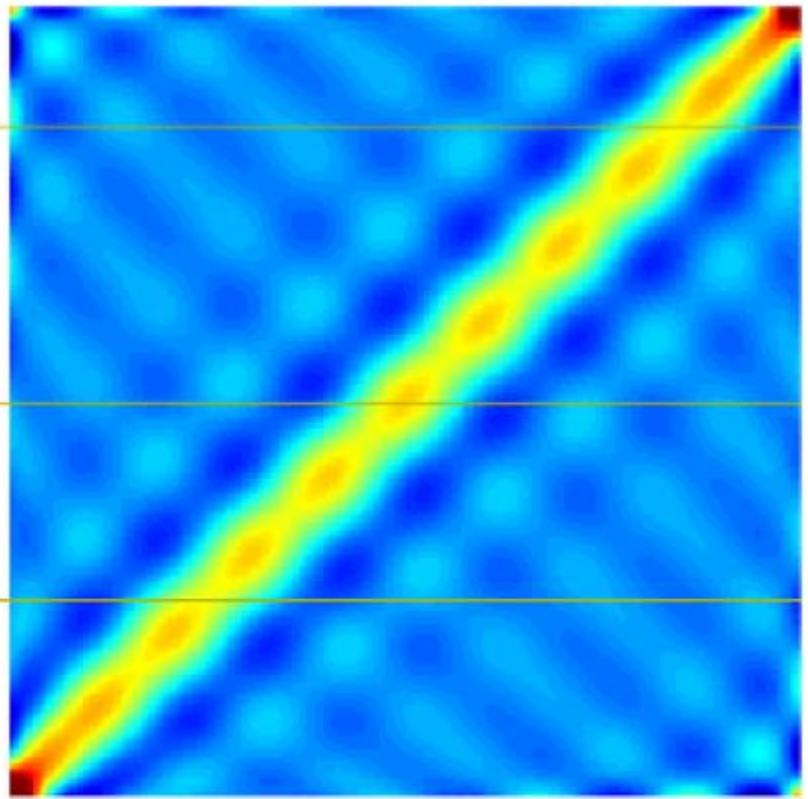
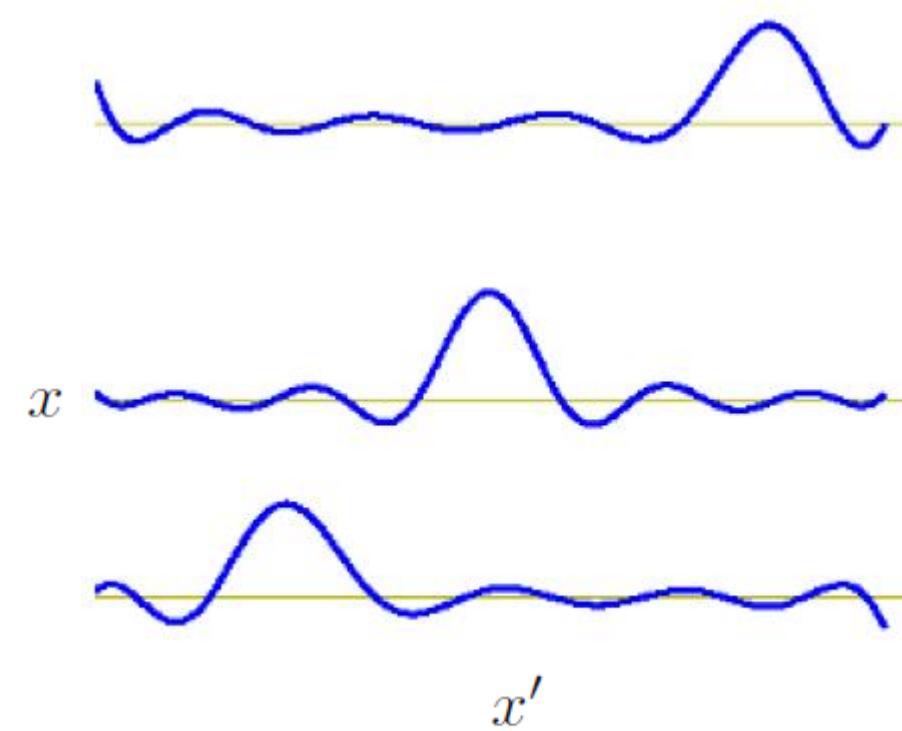


$$\begin{aligned} \text{cov}[y(\mathbf{x}), y(\mathbf{x}')] &= \text{cov}[\phi(\mathbf{x})^T \mathbf{w}, \mathbf{w}^T \phi(\mathbf{x}')] \\ &= \phi(\mathbf{x})^T \mathbf{S}_N \phi(\mathbf{x}') = \beta^{-1} k(\mathbf{x}, \mathbf{x}') \end{aligned}$$

Nearby points are strongly correlated with their predictive means.

- We can verify

$$\sum_{n=1}^N k(\mathbf{x}, \mathbf{x}_n) = 1$$



$$k(x, x')$$

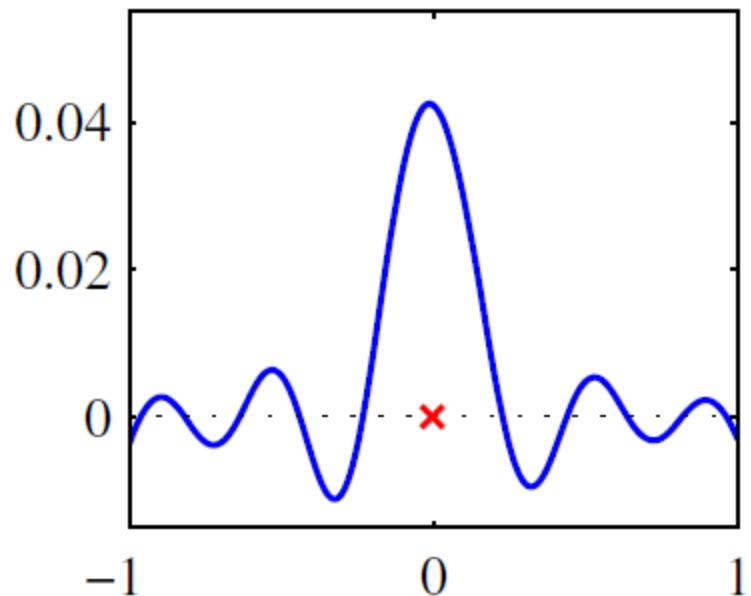
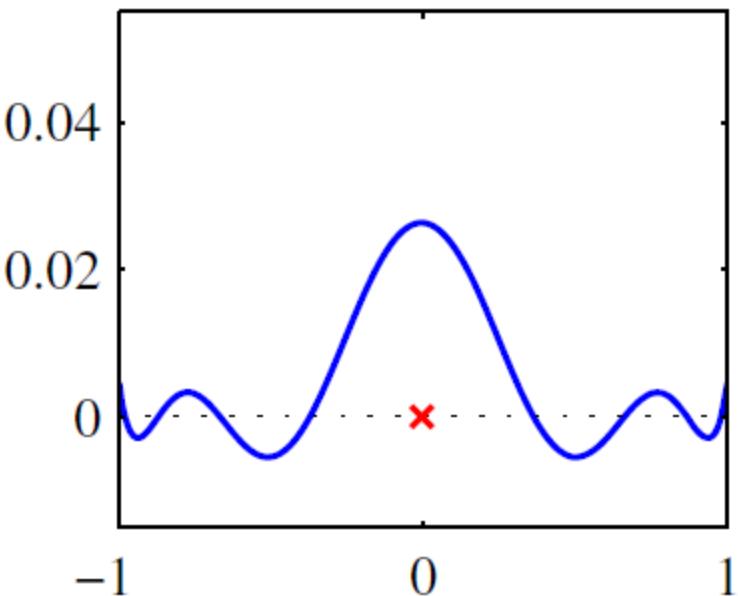


Figure 3.11 Examples of equivalent kernels $k(x, x')$ for $x = 0$ plotted as a function of x' , corresponding (left) to the polynomial basis functions and (right) to the sigmoidal basis functions shown in Figure 3.1. Note that these are localized functions of x' even though the corresponding basis functions are nonlocal.

Bayesian Model Comparison

- We consider the problem of **model selection** from a **Bayesian** perspective.
- We can determine **regularization** parameter. Models can be compared **without validation** data. Cross-validation can be avoided. In comparison of L models $\{\mathcal{M}_i\}$, we select optimal model according to the posterior distribution. $p(\mathcal{M}_i|\mathcal{D}) \propto p(\mathcal{M}_i)p(\mathcal{D}|\mathcal{M}_i)$
- **Predictive** distribution is given by

$$p(t|\mathbf{x}, \mathcal{D}) = \sum_{i=1}^L p(t|\mathbf{x}, \mathcal{M}_i, \mathcal{D})p(\mathcal{M}_i|\mathcal{D})$$

- Use the single most probable model alone to model prediction
→ **model selection**
- For a model governed by w , the **model evidence** is given by

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

- Probability of generating D from a model whose parameters are sampled at random from a prior

$$p(\mathbf{w}|\mathcal{D}, \mathcal{M}_i) = \frac{p(\mathcal{D}|\mathbf{w}, \mathcal{M}_i)p(\mathbf{w}|\mathcal{M}_i)}{p(\mathcal{D}|\mathcal{M}_i)}$$

$$p(\mathcal{D}) = \int p(\mathcal{D}|w)p(w) dw \simeq p(\mathcal{D}|w_{\text{MAP}}) \frac{\Delta w_{\text{posterior}}}{\Delta w_{\text{prior}}}$$

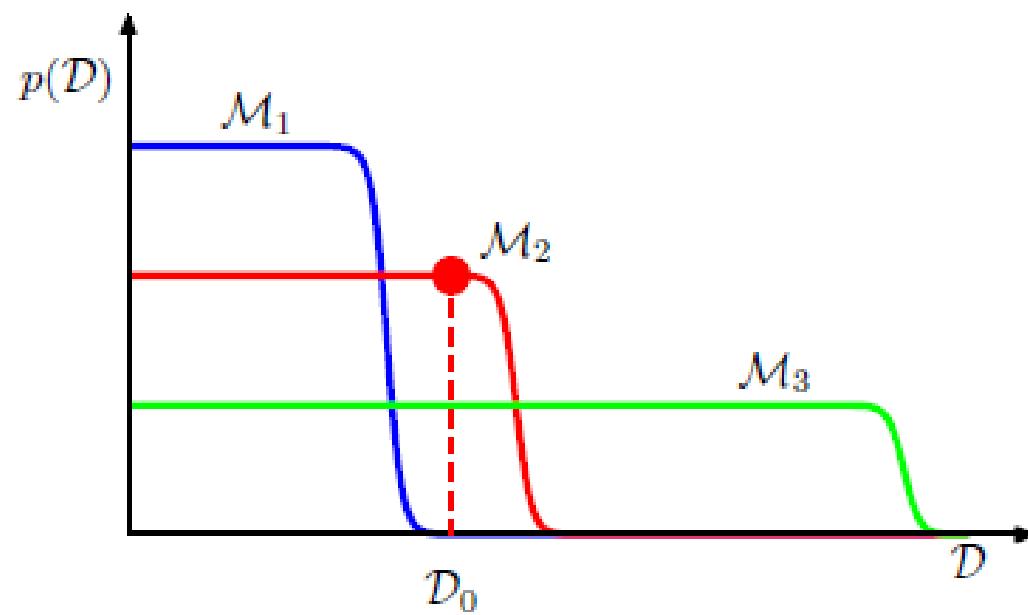
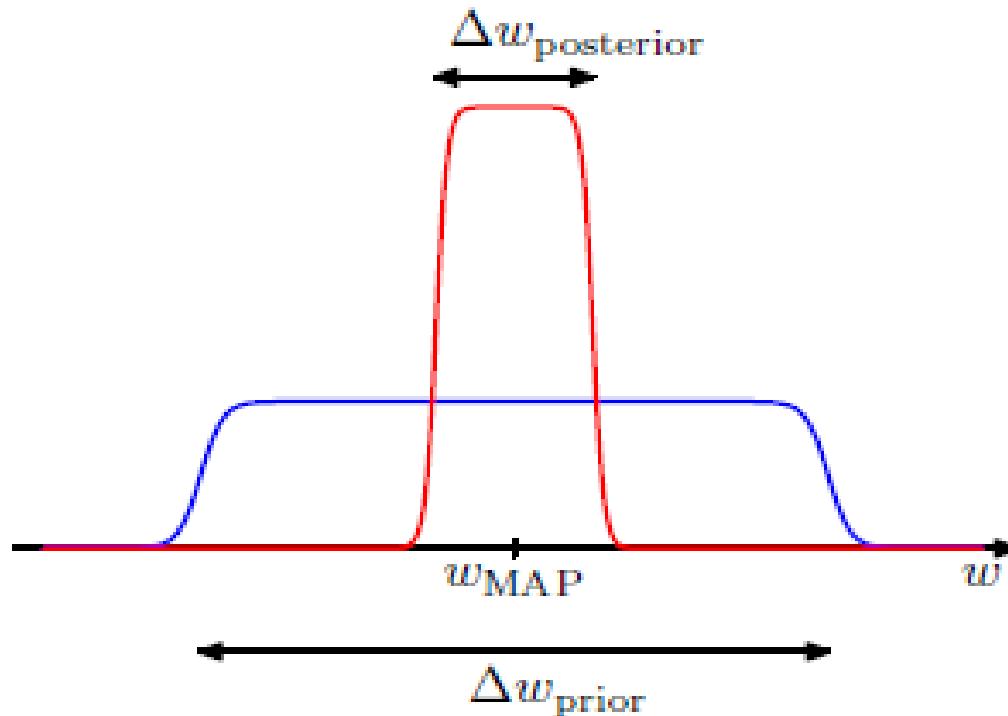
$$\ln p(\mathcal{D}) \simeq \ln p(\mathcal{D}|w_{\text{MAP}}) + \ln \left(\frac{\Delta w_{\text{posterior}}}{\Delta w_{\text{prior}}} \right)$$

Here, $\Delta w_{\text{posterior}} < \Delta w_{\text{prior}}$, model penalty is negative

- Optimal model complexity is given by a trade-off between two competing terms.

$$p(w) \rightarrow p(D|w) \rightarrow p(D)$$

- Simple** model has little variability $p(D)$ is confined to a small region.
- Complex** model can generate a variety of different datasets. $p(D)$ is spread over a large region. Its predictive probability spreads too broad.



- Bayesian model comparison on average favor the correct model.

“Expected” Bayes factor:

$$\int p(\mathcal{D}|\mathcal{M}_1) \ln \frac{p(\mathcal{D}|\mathcal{M}_1)}{p(\mathcal{D}|\mathcal{M}_2)} d\mathcal{D}$$

\mathcal{M}_1 correct

\mathcal{M}_2 incorrect

↑
↓
Bayes factor

$$\text{KL Divergence KL} > \int p(\mathcal{D}|\mathcal{M}_1) \ln \frac{p(\mathcal{D}|\mathcal{M}_1)}{p(\mathcal{D}|\mathcal{M}_2)} d\mathcal{D}$$

These approaches need to make assumptions about the form of model.

The Evidence Approximation

- Usually, the complete marginalization over α, β & w is analytically intractable. we can approximate this by setting the optimal hyperparameters via maximization of **marginal likelihood** integrating over w .
 \implies **empirical Bayes** or **type 2 maximum likelihood** or **generalized maximum likelihood** or “evidence approximation”

- Predictive distribution:

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

Laplace Approximation

- If $p(\alpha, \beta|t) = \delta(\alpha - \hat{\alpha})\delta(\beta - \hat{\beta})$

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

$$(\hat{\alpha}, \hat{\beta}) = \operatorname{argmax}_{(\alpha, \beta)} p(\alpha, \beta|t) = \operatorname{argmax}_{(\alpha, \beta)} p(\alpha, \beta)$$

- Training data only. No cross-validation

Marginal over \mathbf{w} is intractable using t -distribution.

⇒ Laplace Approximation (local Gaussian approximation)

- This approximation is centered on the mode of **posterior** distribution.
Poor approximation ⇐= “skewed” mode

Evaluation of the Evidence Function

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

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

where

$$\begin{aligned} E(\mathbf{w}) &= \beta E_D(\mathbf{w}) + \alpha E_W(\mathbf{w}) \\ &= \frac{\beta}{2} \|\mathbf{t} - \Phi \mathbf{w}\|^2 + \frac{\alpha}{2} \mathbf{w}^T \mathbf{w} \\ &= E(\mathbf{m}_N) + \frac{1}{2} (\mathbf{w} - \mathbf{m}_N)^T \mathbf{A} (\mathbf{w} - \mathbf{m}_N) \end{aligned}$$

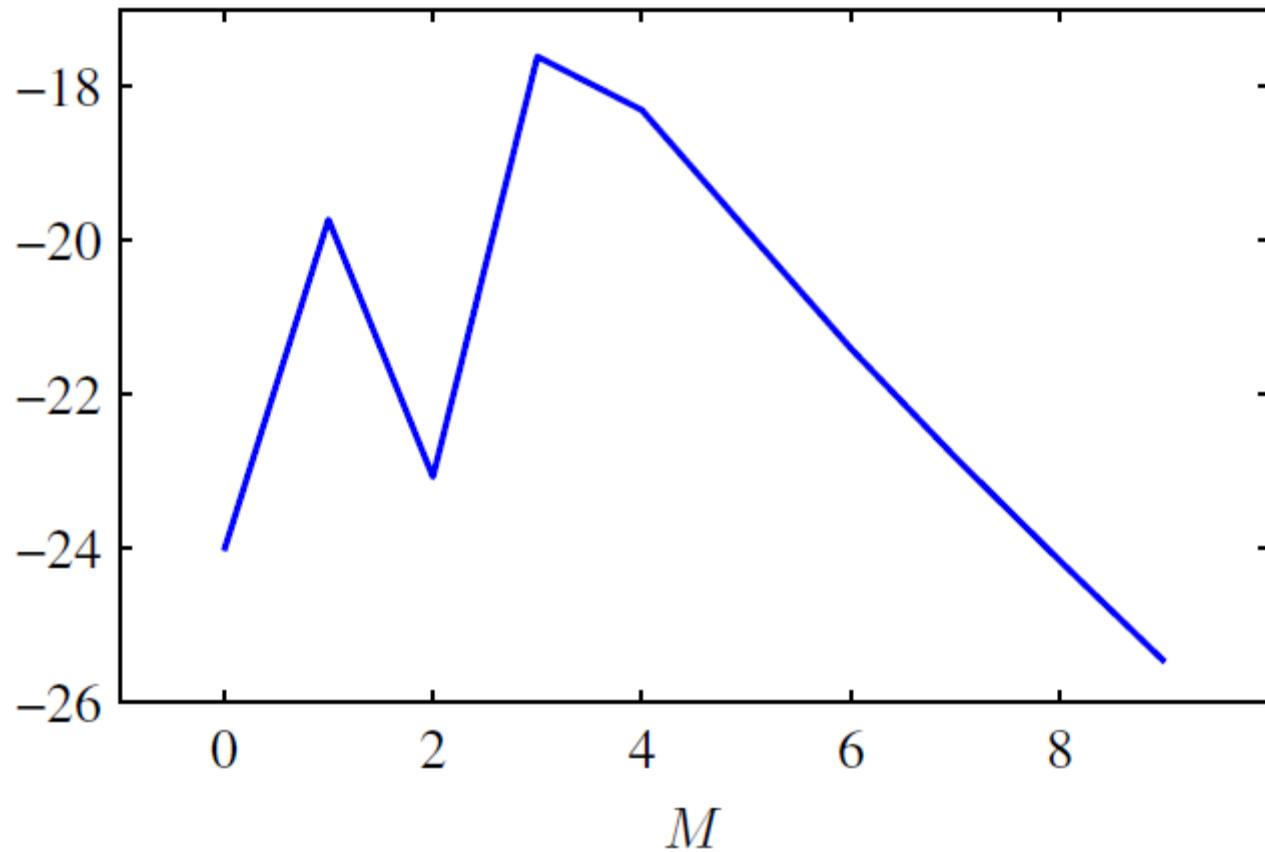
by verification

$$\begin{aligned} &\int \exp\{-E(\mathbf{w})\} d\mathbf{w} \\ &= \exp\{-E(\mathbf{m}_N)\} \int \exp\left\{-\frac{1}{2} (\mathbf{w} - \mathbf{m}_N)^T \mathbf{A} (\mathbf{w} - \mathbf{m}_N)\right\} d\mathbf{w} \\ &= \exp\{-E(\mathbf{m}_N)\} (2\pi)^{M/2} |\mathbf{A}|^{-1/2}. \end{aligned}$$

$$\begin{aligned} \mathbf{A} &= S_N^{-1} = S_0^{-1} + \beta \Phi^T \Phi \\ &= \alpha I + \beta \Phi^T \Phi \\ &= \nabla \nabla E(\mathbf{w}) \end{aligned}$$

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

In case of using polynomial regression, M is the order of the polynomial.



Maximizing the Evidence Function

$$\hat{\alpha} = \text{argmax}_{\alpha} p(t|\alpha, \beta)$$

- Given the **eigenvector** equation $(\beta \Phi^T \Phi) \mathbf{u}_i = \lambda_i \mathbf{u}_i$,
A has eigenvalues $\{ \alpha + \lambda_i \}$

$$\begin{aligned}\frac{d}{d\alpha} \ln |\mathbf{A}| &= \frac{d}{d\alpha} \ln \prod_i (\lambda_i + \alpha) = \frac{d}{d\alpha} \sum_i \ln(\lambda_i + \alpha) = \sum_i \frac{1}{\lambda_i + \alpha} \\ &= \sum_i \frac{1}{\lambda_i + \alpha}\end{aligned}$$

$$\frac{d}{d\alpha} \ln p(t|\alpha, \beta) = \frac{M}{2\alpha} - \frac{1}{2} \mathbf{m}_N^T \mathbf{m}_N - \frac{1}{2} \sum_i \frac{1}{\lambda_i + \alpha} = 0$$

$$\alpha \mathbf{m}_N^T \mathbf{m}_N = M - \alpha \sum_i \frac{1}{\lambda_i + \alpha} = \gamma = \sum_i \frac{\lambda_i}{\alpha + \lambda_i}$$

- $\alpha = \frac{\gamma}{\mathbf{m}_N^T \mathbf{m}_N}$ **implicit solution** $\alpha^{(0)} \rightarrow M_N^{(0)} \rightarrow \gamma^{(0)} \rightarrow \alpha^{(1)} \rightarrow \dots$

Bayesian solution

Implicit Solution

- Only **training data** are used.

$$\hat{\beta} = \operatorname{argmax}_{\beta} P(t|\alpha, \beta)$$

$$\frac{d}{d\beta} \ln |\mathbf{A}| = \frac{d}{d\beta} \sum_i \ln(\lambda_i + \alpha) = \frac{1}{\beta} \sum_i \frac{\lambda_i}{\lambda_i + \alpha} = \frac{\gamma}{\beta}$$

$$\frac{d}{d\alpha} \ln p(t|\alpha, \beta) = \frac{N}{2\beta} - \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{m}_N^T \phi(\mathbf{x}_n)\}^2 - \frac{\gamma}{2\beta} = 0$$

$$\boxed{\frac{1}{\beta} = \frac{1}{N-\gamma} \sum_{n=1}^N \{t_n - \mathbf{m}_N^T \phi(\mathbf{x}_n)\}^2} \quad \beta^{(0)} \rightarrow M_N^{(0)} \rightarrow \gamma^{(0)} \rightarrow \beta^{(1)} \rightarrow \dots$$

Effective Number of Parameters

$$0 \leq \frac{\lambda_i}{\lambda_i + \alpha} \leq 1 \quad 0 \leq \gamma \leq M$$

- γ measures the **effective total number** of well determined parameter.

- If $\lambda_i \gg \alpha$ $\frac{\lambda_i}{\lambda_i + \alpha} \rightarrow 1$
 $w_i \rightarrow w_{ML} \rightarrow$ well determined

$$\frac{1}{\beta} = \frac{1}{N - \gamma} \sum_{n=1}^N \{t_n - \mathbf{m}_N^T \phi(\mathbf{x}_n)\}^2$$

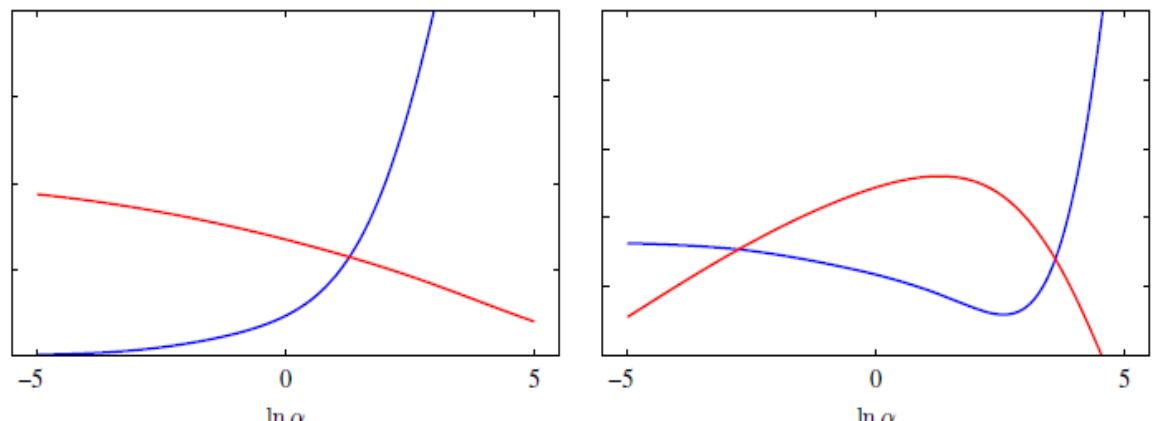
$$\sigma_{ML}^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \mu_{ML})^2$$

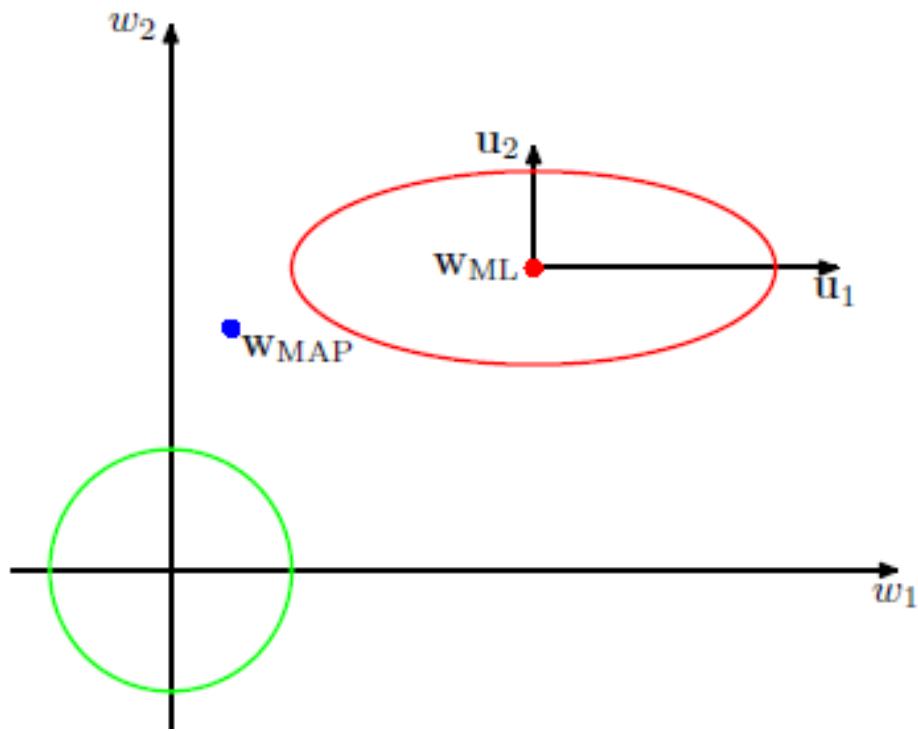
- In case of $N \gg M$

$\lambda_i \uparrow r \rightarrow M$

$$\begin{aligned} \alpha &= \frac{M}{2E_W(\mathbf{m}_N)} \\ \beta &= \frac{N}{2E_D(\mathbf{m}_N)} \end{aligned}$$

easy-to-compute approximation





CONTENTS

1. Introduction
2. Probability Distributions
3. Linear Models for Regression
4. Linear Models for Classification
5. Kernel Methods
6. Sparse Kernel Methods
7. Mixture Models and EM
8. Approximate Inference

LINEAR MODELS FOR CLASSIFICATION

- Goal : take an input vector x and to assign it to one of k discrete classes c_k where $k = 1, \dots, k$

decision boundaries or decision surfaces, linearly separable ?

Assignment of t : $k = 5 \quad x \in c_2 \quad t = (0, 1, 0, 0, 0)^T$

Three approaches : 1. construct *discriminant function* $y(x) \in c_k$

2. find $p(c_k|x)$ & make decision

3. use generative model $p(x|c_k)$ &

prior probabilities $p(c_k)$ & $p(c_k|x) = \frac{p(x|c_k)p(c_k)}{p(x)}$

- Linear regression model $y(x) = (\mathbf{w}^T \mathbf{x} + w_0)$

$$\xrightarrow{\hspace{1cm}} y(x) = f(\mathbf{w}^T \mathbf{x} + w_0) \longleftrightarrow \text{link function}$$

for classification

$$0 \leq y(x) \leq 1$$



activation function
nonlinear

Generalized Linear model

Discriminant Functions

- Two classes $y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 \geq 0, y \in c_1$, Decision boundary $y(\mathbf{x}) = 0 \leq 0, y \in c_2$
- If \mathbf{x}_A & \mathbf{x}_B lie in hyperplane $y(\mathbf{x}_A) = y(\mathbf{x}_B) = 0$

$$\boxed{\mathbf{w}^T(\mathbf{x}_A - \mathbf{x}_B) = 0}$$

- Normal distance of $\hat{\mathbf{x}}$ to $y(\mathbf{x}) = 0$

$$\|\hat{\mathbf{x}}\| \cos \theta = \|\hat{\mathbf{x}}\| \frac{\hat{\mathbf{x}}^T W}{\|\hat{\mathbf{x}}\| \|W\|}$$

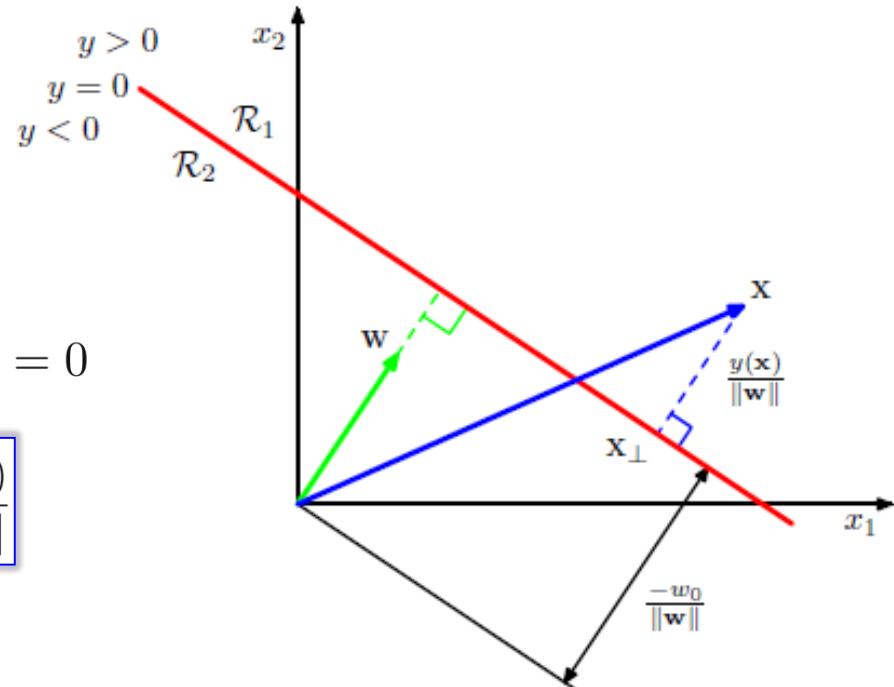
$$= \frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|}$$

$$\mathbf{x} = \mathbf{x}_\perp + r \frac{\mathbf{w}}{\|\mathbf{w}\|} \quad y(\mathbf{x}_\perp) = \mathbf{w}^T \mathbf{x}_\perp + w_0 = 0$$

$$W^T \mathbf{x} = W^T \mathbf{x}_\perp + \gamma \frac{W^T W}{\|W\|} \implies \boxed{r = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}}$$

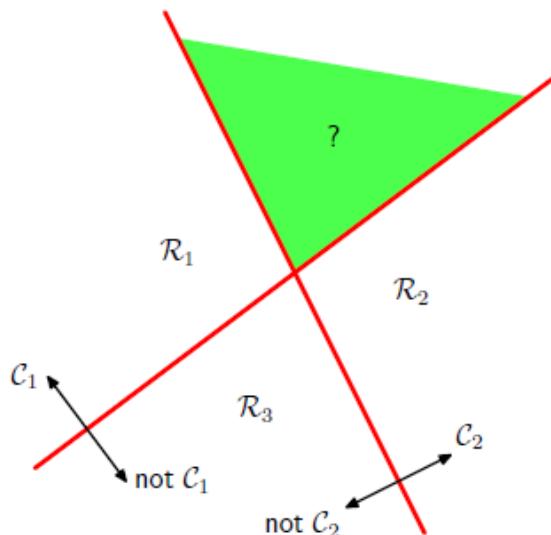
$$= -w_0 + \gamma \frac{W^T W}{\|W\|}$$

$$y(\mathbf{x}) = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}} \quad \tilde{\mathbf{w}} = (w_0, \mathbf{w}) \quad \tilde{\mathbf{x}} = (x_0, \mathbf{x})$$

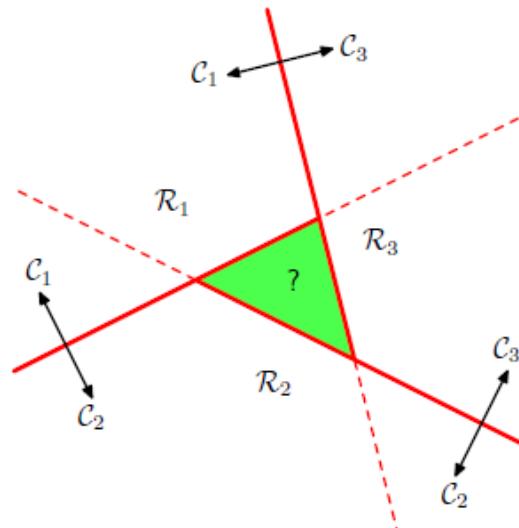


Multiple Classes

$k > 2$ Some difficulties happen when combining a number of two-class discriminant functions.



One-versus-the-rest
classifier



One-versus-one
classifier

- We can avoid these difficulties by considering a single K -class discriminant comprising K linear functions

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

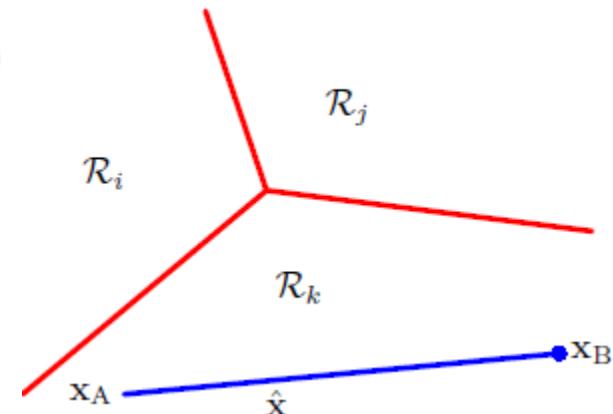
$$\hat{\mathbf{x}} = \lambda \mathbf{x}_A + (1 - \lambda) \mathbf{x}_B$$

$$y_k(\hat{\mathbf{x}}) = \lambda y_k(\mathbf{x}_A) + (1 - \lambda) y_k(\mathbf{x}_B)$$

- \mathbf{X} is assigned to C_k if $y_k(\hat{\mathbf{x}}) > y_j(\hat{\mathbf{x}})$ for all $j \neq k$
- decision boundary C_k and C_j is given by

$$y_k(\mathbf{x}) = y_j(\mathbf{x}) \implies (\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} + (w_{k0} - w_{j0}) = 0$$

(D - 1)-dimensional hyperplane



Three approaches to learning the parameters of linear discriminant function

Least Squares for Classification

- k classes Least squares approximate the conditional expectation $E[t|x] \rightarrow$ class posterior prob.
Closed-form solution to regression is available.
- We have $y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0} \quad k = 1, \dots, K$

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^T \widetilde{\mathbf{x}} \quad \mathbf{x} \in C_k \text{ if } y_k = \widetilde{\mathbf{w}}_k^T \widetilde{\mathbf{x}} \text{ is the largest .}$$

- Given a training set $\{X_n, t_n\}_{n=1}^N$, the **sum-of-squares error** function can be written by

$$\tilde{X} = \begin{bmatrix} -\tilde{X}_1^T \\ \vdots \\ -\tilde{X}_N^T \end{bmatrix} \quad \tilde{W} = [\tilde{W}_1 \quad \cdots \quad \tilde{W}_k] \quad \tilde{T} = \begin{bmatrix} -\tilde{t}_1^T \\ \vdots \\ -\tilde{t}_N^T \end{bmatrix}$$

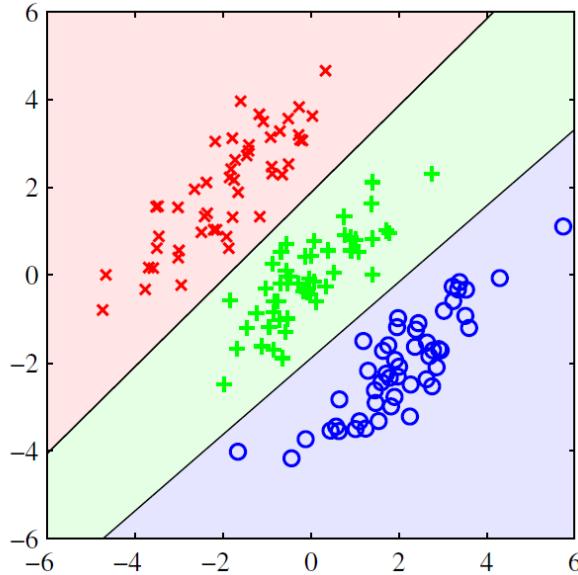
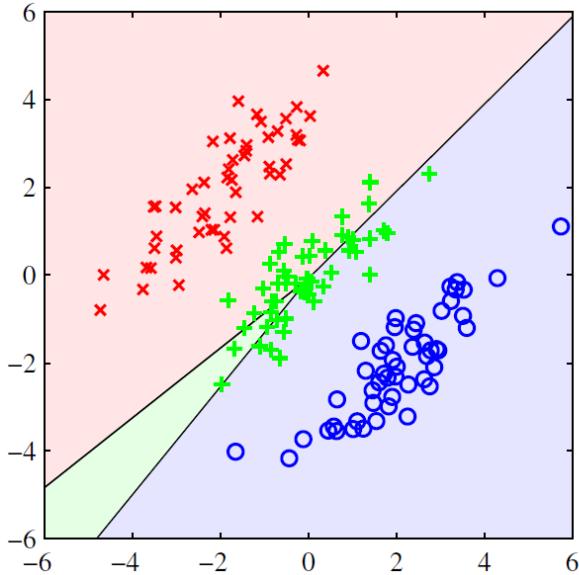
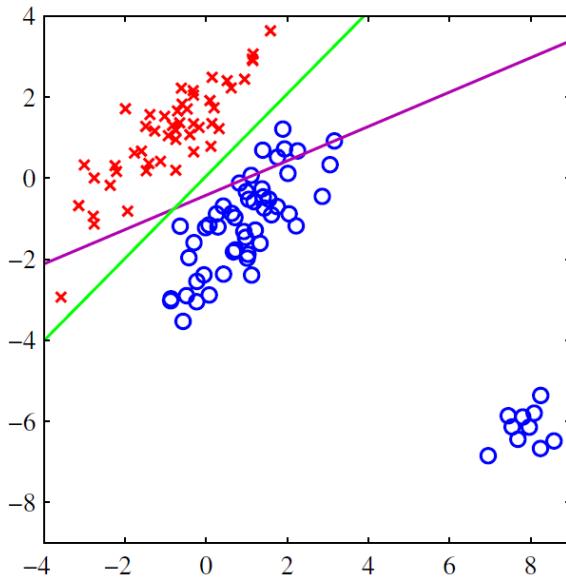
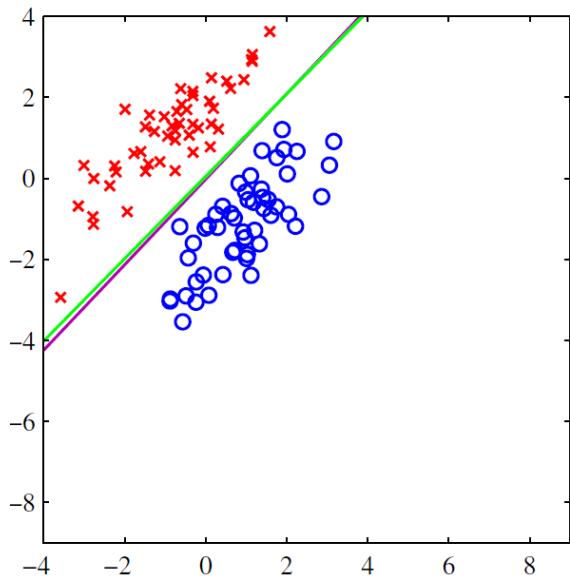
$$E_D(\tilde{\mathbf{W}}) = \frac{1}{2} \text{Tr} \left\{ (\tilde{\mathbf{X}} \tilde{\mathbf{W}} - \mathbf{T})^T (\tilde{\mathbf{X}} \tilde{\mathbf{W}} - \mathbf{T}) \right\}$$

$$\nabla_{\tilde{W}} E_D = 0 \implies \tilde{\mathbf{W}} = (\tilde{\mathbf{X}}^T \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^T \mathbf{T} = \tilde{\mathbf{X}}^\dagger \mathbf{T}$$

- Discriminant function is obtained by

$$\mathbf{y}(\mathbf{x}) = \tilde{\mathbf{W}}^T \tilde{\mathbf{x}} = \mathbf{T}^T \left(\tilde{\mathbf{X}}^\dagger \right)^T \tilde{\mathbf{x}}$$

- Least squares is highly sensitive to outliers.
- ML under the assumption of a Gaussian conditional distribution whereas binary target vectors clearly have a distribution far from Gaussian. More appropriate probabilistic models are needed.



Comparison of least-squares discriminant & logistic regression model

Fisher's Linear Discriminant

- Dimensionality reduction $y = \mathbf{w}^T \mathbf{x} \geq_{C_2}^{C_1} w_0$
- We can select a projection that maximizes the class separation.

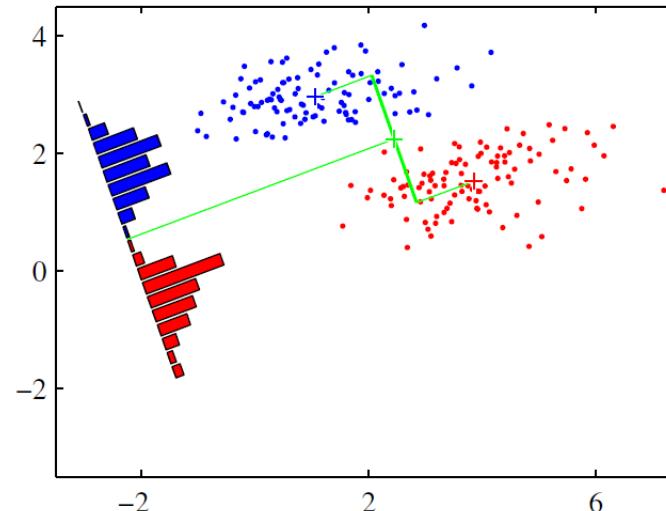
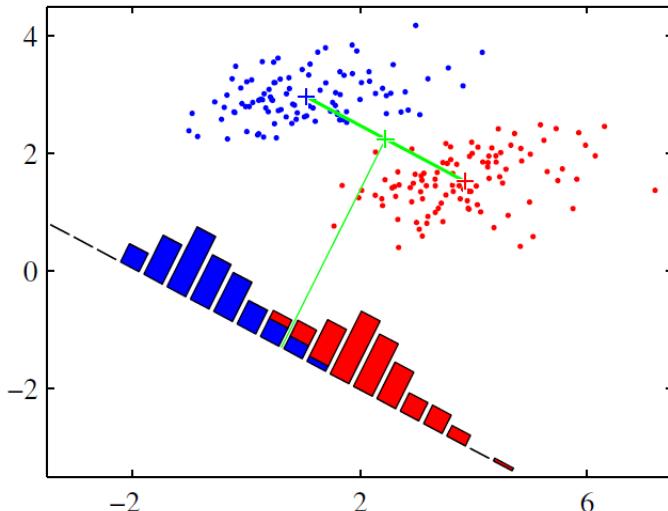
$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n$$

$$\mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n$$

$m_2 - m_1 = \mathbf{w}^T (\mathbf{m}_2 - \mathbf{m}_1)$ separation of the projected class mean.

Optimize separation under constraint $\sum_i w_i^2 = 1$,

we obtain $\boxed{\mathbf{w} \propto (\mathbf{m}_2 - \mathbf{m}_1)}$



- Fisher's criterion is to maximize a function that gives a large separation between the projected class means while also gives small variance within each class.
- within-class variance $s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2$

Fisher criterion

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

where

$$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$$

$$\mathbf{S}_W = \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T$$

$$\nabla_N J(\mathbf{w}) = 0 \quad (\mathbf{w}^T \mathbf{S}_B \mathbf{w}) \mathbf{S}_W \mathbf{w} = (\mathbf{w}^T \mathbf{S}_W \mathbf{w}) \mathbf{S}_B \mathbf{w}$$

$\mathbf{S}_B \mathbf{w}$ is in the direction of $\mathbf{m}_2 - \mathbf{m}_1$

Solution to Fisher's Linear Discriminant

We don't care about magnitude of \mathbf{w} , only its direction. Then, we obtain

$$\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$$

If within-class covariance is isotropic, $S_{\mathbf{w}} \propto I$

$$W \propto (m_2 - m_1)$$

Classification Rule $y(\mathbf{x}) >_{C_2}^{C_1} y_0$ (threshold)

we can assume $p(y|\mathcal{C}_k)$ & use Gaussian
use ML to determine y_0

Relation to Least Squares

(1-of- K coding) By using different coding, **least-squares** is equivalent to the **Fisher solution**.

Target for \mathcal{C}_1 is N/N_1

Target for \mathcal{C}_2 is $-N/N_2$

The sum-of-squares error function $E = \frac{1}{2} \sum_{n=1}^N (\mathbf{w}^T \mathbf{x}_n + w_0 - t_n)^2$

$$\sum_{n=1}^N (\mathbf{w}^T \mathbf{x}_n + w_0 - t_n) = 0 \quad \text{--- } ①$$

$$\sum_{n=1}^N (\mathbf{w}^T \mathbf{x}_n + w_0 - t_n) \mathbf{x}_n = 0 \quad \text{--- } ②$$

Using the specified target values, we have

$$\sum_{n=1}^N t_n = N_1 \frac{N}{N_1} - N_2 \frac{N}{N_2} = 0$$

① will give $w_0 = -\mathbf{w}^T \mathbf{m}$ where $\mathbf{m} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n = \frac{1}{N} (N_1 \mathbf{m}_1 + N_2 \mathbf{m}_2)$

Also, ② will give $\left(\mathbf{S}_W + \frac{N_1 N_2}{N} \mathbf{S}_B \right) \mathbf{w} = N(\mathbf{m}_2 - \mathbf{m}_1)$

Because $(\mathbf{S}_B \mathbf{w})$ is in the direction $\mathbf{m}_2 - \mathbf{m}_1$, we obtain

$$\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$$

Fisher's Discriminant for Multiple Classes

The Perceptron Algorithm

$$y(\mathbf{x}) = f(\mathbf{w}^T \phi(\mathbf{x}))$$

$f(\cdot)$ nonlinear activation function $f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$

$t = +1$ for class \mathcal{C}_1 . $t = -1$ for class \mathcal{C}_2

Objective : error function minimization

total number of misclassified patterns.

Discontinuities happen everywhere

Perceptron criterion is derived.

$\mathbf{w}^T \phi(\mathbf{x}_n) > 0$ Let

The Perceptron Algorithm

All patterns satisfy

$$\mathbf{w}^T \phi(\mathbf{x}_n) > 0 \quad t \in \{-1, +1\}$$

Perceptron criterion

$$E_P(\mathbf{w}) = - \sum_{n \in \mathcal{M}} \mathbf{w}^T \phi_n t_n$$



Frank Rosenblatt
1928–1969

Rosenblatt's perceptron played an important role in the history of machine learning. Initially, Rosenblatt simulated the perceptron on an IBM 704 computer at Cornell in 1957, but by the early 1960s he had built special-purpose hardware that provided a direct, parallel implementation of perceptron learning. Many of his ideas were encapsulated in "Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms" published in 1962. Rosenblatt's work was criticized by Marvin Minsky, whose objections were published in the book "Perceptrons", co-authored with

Seymour Papert. This book was widely misinterpreted at the time as showing that neural networks were fatally flawed and could only learn solutions for linearly separable problems. In fact, it only proved such limitations in the case of single-layer networks such as the perceptron and merely conjectured (incorrectly) that they applied to more general network models. Unfortunately, however, this book contributed to the substantial decline in research funding for neural computing, a situation that was not reversed until the mid-1980s. Today, there are many hundreds, if not thousands, of applications of neural networks in widespread use, with examples in areas such as handwriting recognition and information retrieval being used routinely by millions of people.

Perceptron Convergence Theorem

M : set of all misclassified patterns

Stochastic Gradient Descent Algorithm

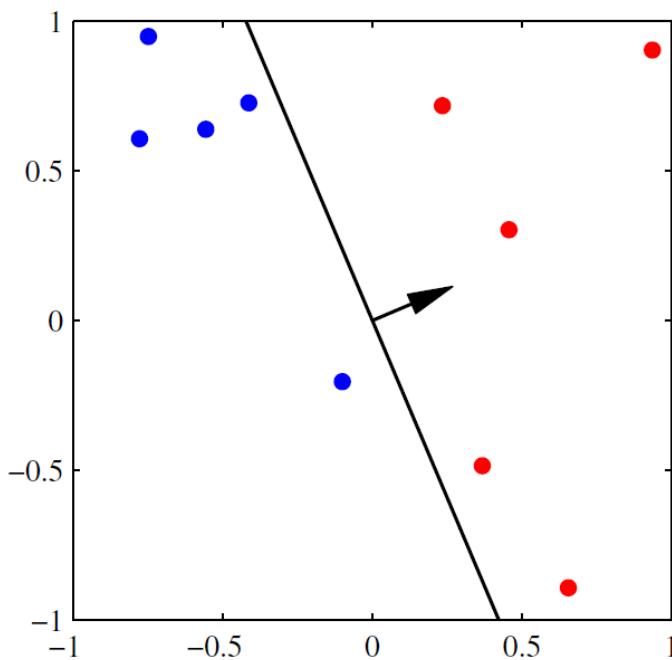
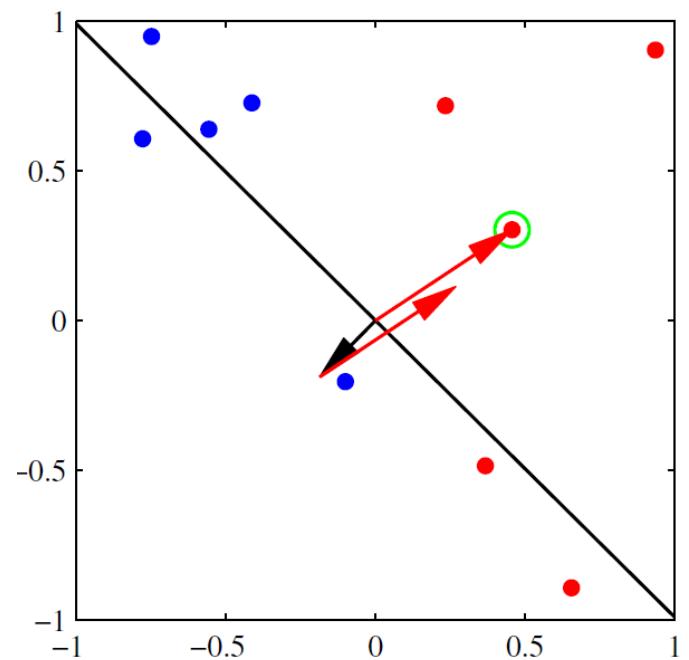
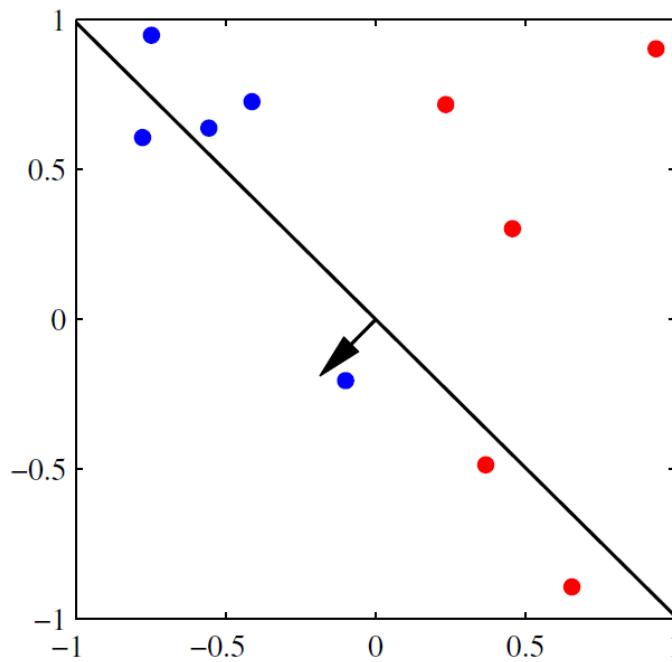
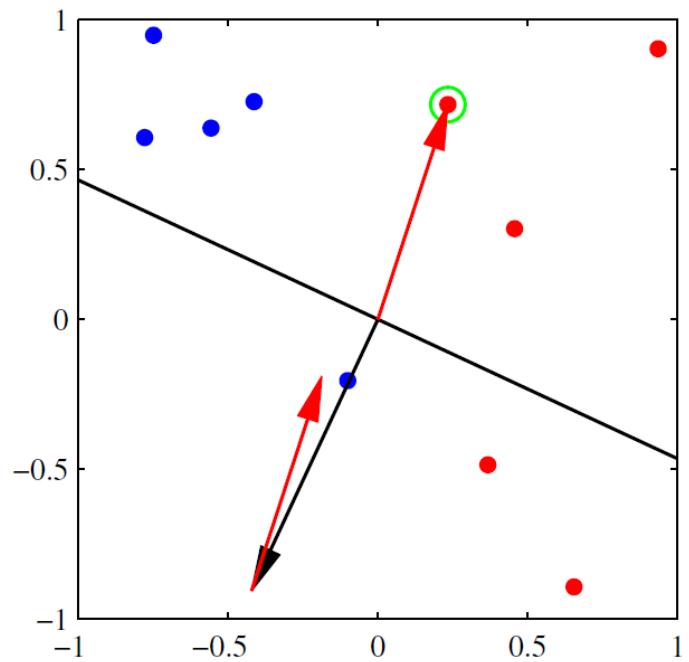
$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta \phi_n t_n$$

- Watch carefully the movement of hyperplane and w due to the first and the second misclassified patterns.

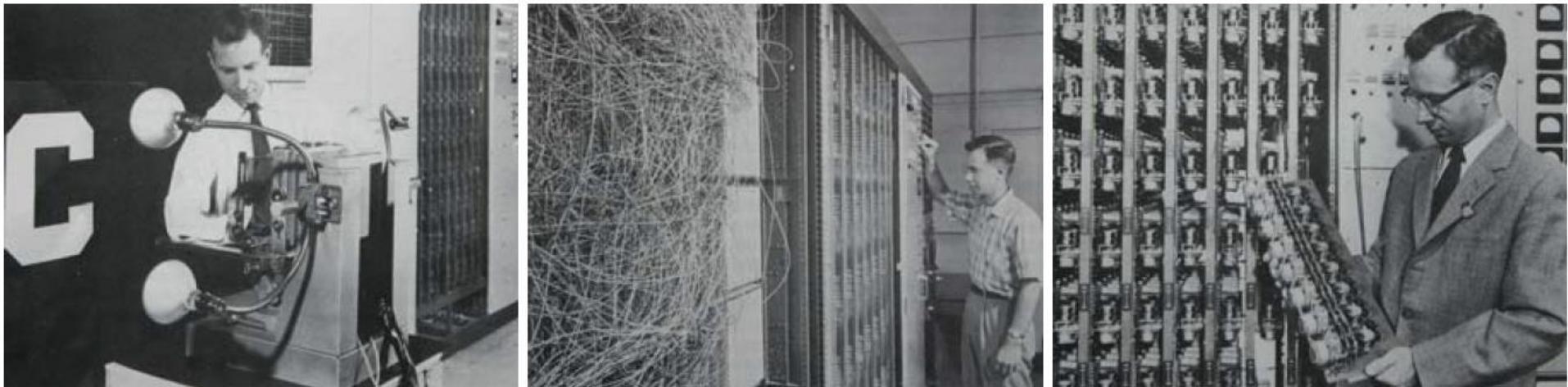
Contribution to the error from a misclassified pattern.

$$-\mathbf{w}^{(\tau+1)T} \phi_n t_n = -\mathbf{w}^{(\tau)T} \phi_n t_n - (\phi_n t_n)^T \phi_n t_n < -\mathbf{w}^{(\tau)T} \phi_n t_n$$

- Perceptron learning rule is not guaranteed to reduce total error at each stage when the samples are **not linearly separable**.
- Perceptron convergence theorem**: if training samples are linearly separable, an exact solution is guaranteed to be found.



Mark 1 Perceptron Hardware



No **probabilistic** output is available in perceptron algorithm.

The **adaline** (adaptive linear element) use the same functional form of model but different training approaches when compared with perceptron algorithm.

Probabilistic Generative Models

$$\begin{aligned} p(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \frac{1}{1 + \exp(-a)} = \sigma(a) \end{aligned}$$

where $a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$
 $= \ln [p(\mathcal{C}_1|\mathbf{x})/p(\mathcal{C}_2|\mathbf{x})]$

$$\sigma(-a) = 1 - \sigma(a) \quad \text{symmetric}$$

$$a = \ln \left(\frac{\sigma}{1 - \sigma} \right) \quad \text{logistic sigmoid}$$

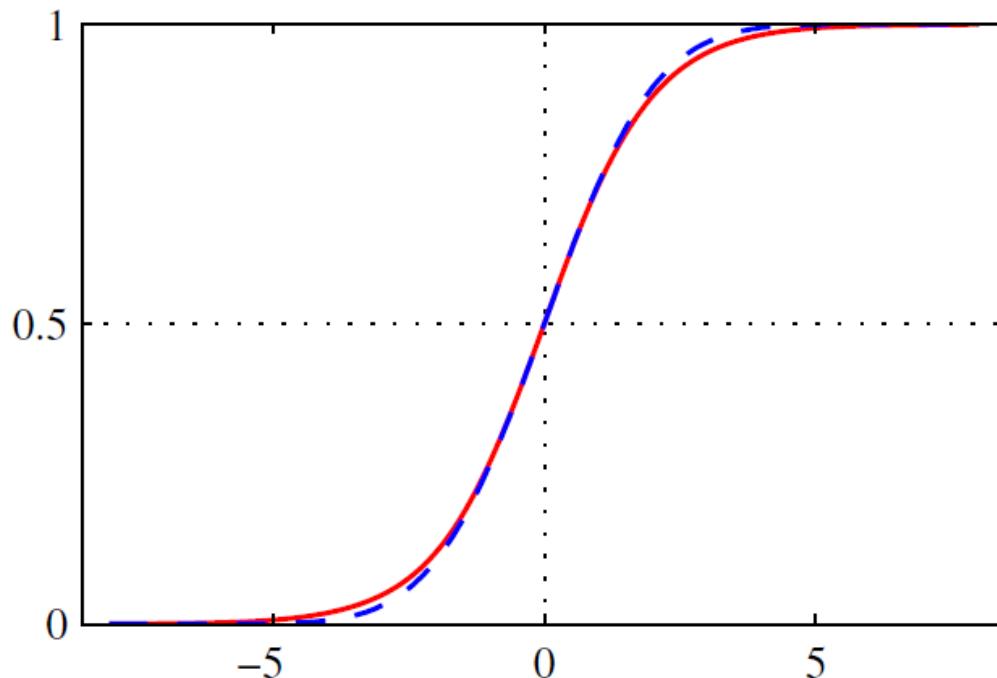
For $K > 2$

$$\begin{aligned} p(\mathcal{C}_k|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)} \\ &= \frac{\exp(a_k)}{\sum_j \exp(a_j)} \end{aligned}$$

normalized exponential

$$a_k = \ln p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)$$

or *softmax* function



Plot of the logistic sigmoid function $\sigma(a)$ defined by (4.59), shown in red, together with the scaled probit function $\Phi(\lambda a)$, for $\lambda^2 = \pi/8$, shown in dashed blue, where $\Phi(a)$ is defined by (4.114). The scaling factor $\pi/8$ is chosen so that the derivatives of the two curves are equal for $a = 0$.

$$\Phi(a) = \int_{-\infty}^a \mathcal{N}(\theta|0, 1) d\theta$$

Continuous Inputs

- Class-conditional densities

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

- Considering two classes

$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x} + w_0)$$

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_2 + \ln \frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}$$

- Class-conditional density \leftrightarrow posterior probability
- For the general case of K classes, we have

$$a_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

$$\mathbf{w}_k = \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k$$

$$w_{k0} = -\frac{1}{2}\boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + \ln p(\mathcal{C}_k)$$

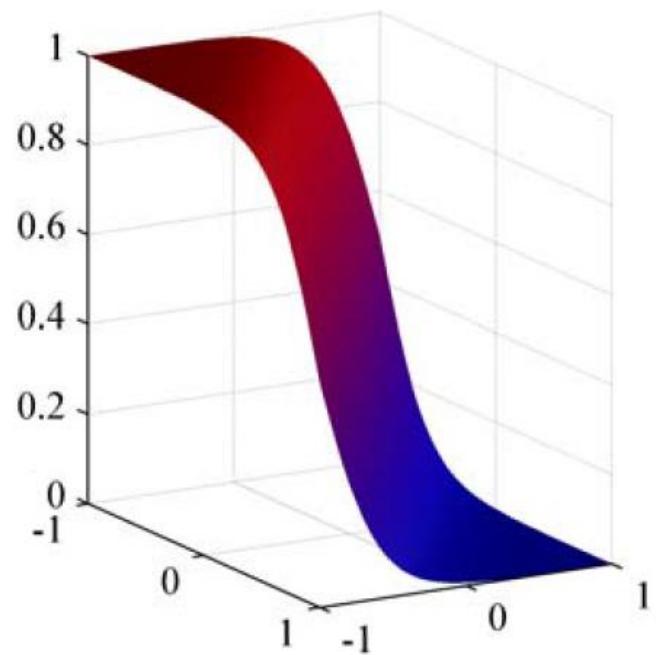
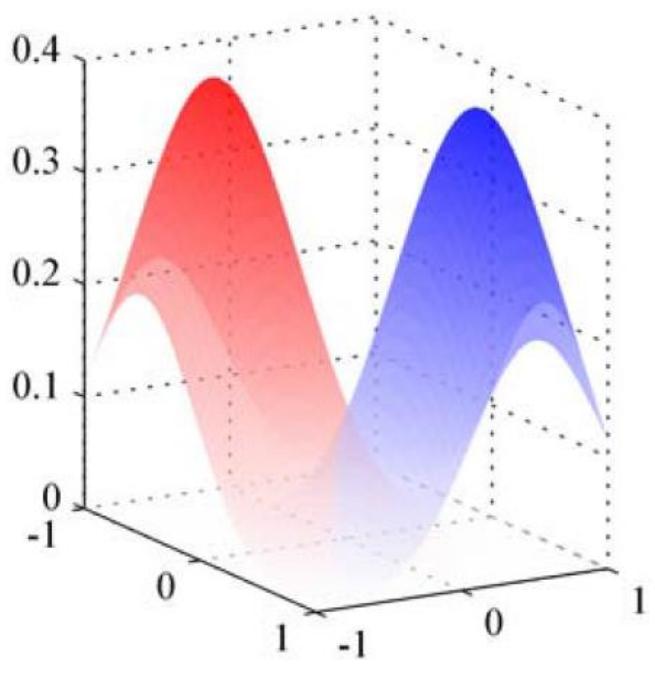
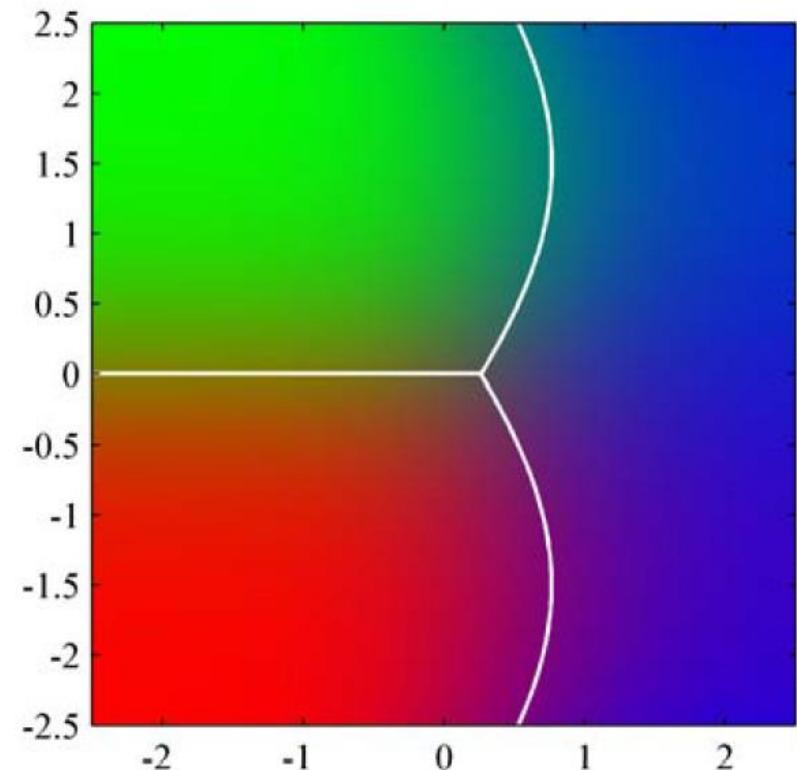
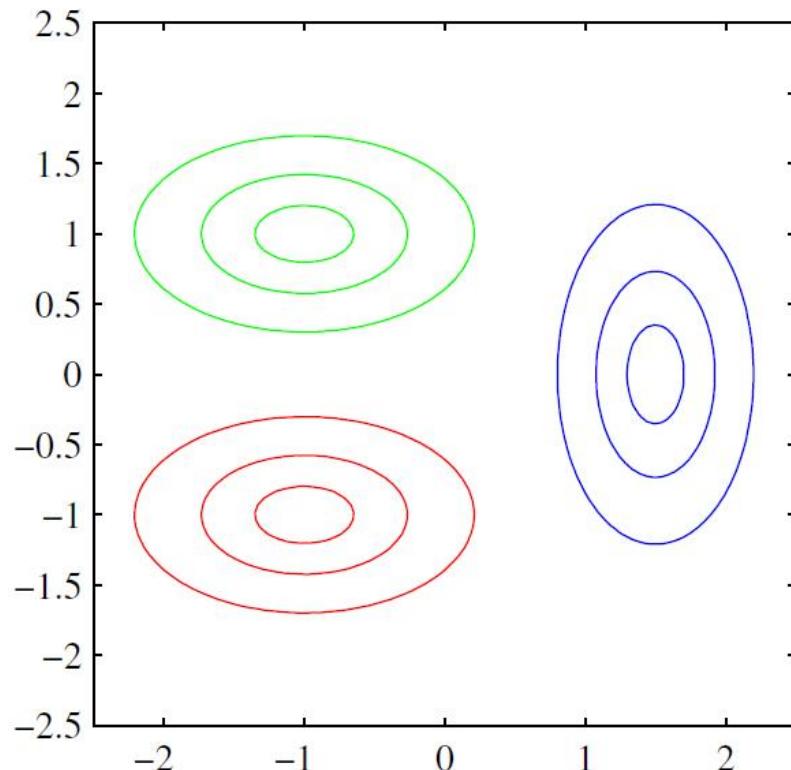


Figure 4.10 The left-hand plot shows the class-conditional densities for two classes, denoted red and blue. On the right is the corresponding posterior probability $p(\mathcal{C}_1|x)$, which is given by a logistic sigmoid of a linear function of x . The surface in the right-hand plot is coloured using a proportion of red ink given by $p(\mathcal{C}_1|x)$ and a proportion of blue ink given by $p(\mathcal{C}_2|x) = 1 - p(\mathcal{C}_1|x)$.

Decision boundaries when two posterior are equal.

→minimum misclassification rate



Maximum Likelihood Solution

$$\text{Gaussian class-conditional density} \quad \{\mathbf{x}_n, t_n\}_{n=1}^N \quad \begin{array}{ll} t_n = 1 & \mathcal{C}_1 \\ t_n = 0 & \mathcal{C}_2 \end{array}$$

$$p(\mathcal{C}_1) = \pi \quad p(\mathcal{C}_2) = 1 - \pi$$

$$p(\mathbf{x}_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(\mathbf{x}_n|\mathcal{C}_1) = \pi \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) \quad t_n = 1$$

$$p(\mathbf{x}_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(\mathbf{x}_n|\mathcal{C}_2) = (1 - \pi) \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}) \quad t_n = 0$$

Likelihood function $\mathbf{t} = (t_1, \dots, t_N)^T$

$$p(\mathbf{t}, \mathbf{X} | \pi, \mu_1, \mu_2, \boldsymbol{\Sigma}) = \prod_{n=1}^N [\pi \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_1, \boldsymbol{\Sigma})]^{t_n} [(1 - \pi) \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_2, \boldsymbol{\Sigma})]^{1-t_n}$$

$$\frac{\partial p(\mathbf{t}, \mathbf{X} | \pi, \mu_1, \mu_2, \boldsymbol{\Sigma})}{\partial \pi} = 0$$

The functions depending on π are $\sum_{n=1}^N \{t_n \ln \pi + (1 - t_n) \ln(1 - \pi)\}$

We obtain $\pi = \frac{1}{N} \sum_{n=1}^N t_n = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2}$

$$\begin{aligned} \mu_1: \quad & \sum_{n=1}^N t_n \ln \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = -\frac{1}{2} \sum_{n=1}^N t_n (\mathbf{x}_n - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1) + \text{const} \\ \mu_2: \quad & \end{aligned}$$

$\nabla_{\mu_1} p(\mathbf{t} | \pi, \mu_1, \mu_2, \boldsymbol{\Sigma}) = 0$ We obtain $\boxed{\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_{n=1}^N t_n \mathbf{x}_n}$ sample mean of \mathcal{C}_1

$\nabla_{\mu_2} p(\mathbf{t} | \pi, \mu_1, \mu_2, \boldsymbol{\Sigma}) = 0$ We obtain $\boxed{\boldsymbol{\mu}_2 = \frac{1}{N_2} \sum_{n=1}^N (1-t_n) \mathbf{x}_n}$ sample mean of \mathcal{C}_2

$$\begin{aligned} \boldsymbol{\Sigma} : \quad & -\frac{1}{2} \sum_{n=1}^N t_n \ln |\boldsymbol{\Sigma}| - \frac{1}{2} \sum_{n=1}^N t_n (\mathbf{x}_n - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1) \\ & -\frac{1}{2} \sum_{n=1}^N (1-t_n) \ln |\boldsymbol{\Sigma}| - \frac{1}{2} \sum_{n=1}^N (1-t_n) (\mathbf{x}_n - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_2) \\ & = -\frac{N}{2} \ln |\boldsymbol{\Sigma}| - \frac{N}{2} \text{Tr} \{ \boldsymbol{\Sigma}^{-1} \mathbf{S} \} \end{aligned}$$

$$\mathbf{S} = \frac{N_1}{N} \mathbf{S}_1 + \frac{N_2}{N} \mathbf{S}_2 \quad \mathbf{S}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \boldsymbol{\mu}_1) (\mathbf{x}_n - \boldsymbol{\mu}_1)^T$$

weighted average of covariance matrices

$$\mathbf{S}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \boldsymbol{\mu}_2) (\mathbf{x}_n - \boldsymbol{\mu}_2)^T$$

$$\nabla_{\boldsymbol{\Sigma}^{-1}} p(\mathbf{t} | \pi, \mu_1, \mu_2, \boldsymbol{\Sigma}) = 0 \quad \boxed{\boldsymbol{\Sigma} = \mathbf{S}}$$

Maximum likelihood solution is not robust to outliers.

Discrete Features

$x_i \in \{0, 1\}$ D inputs

$$p(\mathbf{x}|\mathcal{C}_k) = \prod_{i=1}^D \mu_{ki}^{x_i} (1 - \mu_{ki})^{1-x_i}$$

$$a_k(\mathbf{x}) = \sum_{i=1}^D \{x_i \ln \mu_{ki} + (1 - x_i) \ln(1 - \mu_{ki})\} + \ln p(\mathcal{C}_k)$$

Exponential Family

$$p(\mathbf{x}|\boldsymbol{\lambda}_k) = h(\mathbf{x})g(\boldsymbol{\lambda}_k) \exp \left\{ \boldsymbol{\lambda}_k^T \mathbf{u}(\mathbf{x}) \right\}$$

$$p(\mathbf{x}|\boldsymbol{\lambda}_k, s) = \frac{1}{s} h \left(\frac{1}{s} \mathbf{x} \right) g(\boldsymbol{\lambda}_k) \exp \left\{ \frac{1}{s} \boldsymbol{\lambda}_k^T \mathbf{x} \right\}$$

- Posterior class probability is given by a logistic sigmoid acting on a “linear” function

$$a(\mathbf{x}) = \frac{1}{s} (\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2)^T \mathbf{x} + \ln g(\boldsymbol{\lambda}_1) - \ln g(\boldsymbol{\lambda}_2) + \ln p(\mathcal{C}_1) - \ln p(\mathcal{C}_2)$$

Two classes $\rightarrow K$ classes

$$a_k(\mathbf{x}) = \frac{1}{s} \boldsymbol{\lambda}_k^T \mathbf{x} + \ln g(\boldsymbol{\lambda}_k) + \ln p(\mathcal{C}_k)$$

Probabilistic Discriminative Models

- **Indirect** : $p(\mathbf{x}|\mathcal{C}_k) \rightarrow$ Maximum likelihood $\rightarrow p(\mathcal{C}_k|\mathbf{x}) \rightarrow$ Decision $p(\mathcal{C}_k)$
- **Direct**: $p(\mathcal{C}_k|\mathbf{x}) \rightarrow$ Discriminative training \rightarrow Fewer adaptive parameters \rightarrow Predictive performance is improved

Logistic regression

- Generalized linear model, using fixed basis functions

$$p(\mathcal{C}_1|\boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma(\mathbf{w}^T \boldsymbol{\phi}) \quad p(\mathcal{C}_2|\boldsymbol{\phi}) = 1 - p(\mathcal{C}_1|\boldsymbol{\phi})$$

- In case of Gaussian class conditional density, mean $2M$ shared covariance Matrix $M(M+1)/2$
- Total number of parameters $M(M+5)/2+1$ quadratic function of M
- Using **maximum likelihood**, we should calculate

$$\frac{d\sigma}{da} = \sigma(1 - \sigma)$$

- Given $\{\phi_n, t_n\}_{n=1}^N, t_n \in \{0, 1\}, \phi_n = \phi(x_n)$, likelihood function has the form

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^N y_n^{t_n} \{1 - y_n\}^{1-t_n} \quad y_n = p(\mathcal{C}_1|\phi_n)$$

$$y_n = \sigma(a_n)$$

$$a_n = \mathbf{w}^T \phi_n$$

$$\mathbf{t} = (t_1, \dots, t_N)^T$$

- We generate $E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}$

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (y_n - t_n) \phi_n$$

the same form as the gradient of the sum-of-squares error function for the linear regression in

⇒ **sequential** learning algorithm can be presented.

Iterative Reweighted Least Squares

- No closed-form solution is available due to the nonlinearity of the logistic sigmoid function.
- Error function can be minimized by *Newton-Raphson iterative optimization*

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

local quadratic approximation to
log likelihood function

$$\begin{aligned} E_D(\mathbf{w}) &= \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 \\ \nabla E(\mathbf{w}) &= \sum_{n=1}^N (\mathbf{w}^T \phi_n - t_n) \phi_n = \Phi^T \Phi \mathbf{w} - \Phi^T \mathbf{t} \\ \mathbf{H} = \nabla \nabla E(\mathbf{w}) &= \sum_{n=1}^N \phi_n \phi_n^T = \Phi^T \Phi \end{aligned}$$

- Newton-Raphson updating:

$$\begin{aligned}\mathbf{w}^{(\text{new})} &= \mathbf{w}^{(\text{old})} - (\Phi^T \Phi)^{-1} \{ \Phi^T \Phi \mathbf{w}^{(\text{old})} - \Phi^T \mathbf{t} \} \\ &= (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t}\end{aligned}$$

- In case of **cross-entropy error function**,

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (y_n - t_n) \phi_n = \Phi^T (\mathbf{y} - \mathbf{t})$$

$$\mathbf{H} = \nabla \nabla E(\mathbf{w}) = \sum_{n=1}^N y_n(1-y_n) \phi_n \phi_n^T = \Phi^T \mathbf{R} \Phi$$

- Weighting matrix

$$\mathbf{R}_{N \times N} = [R_{nn}] \quad \text{where} \quad R_{nn} = y_n(1-y_n)$$

$$\begin{aligned}\mathbf{w}^{(\text{new})} &= \mathbf{w}^{(\text{old})} - (\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T (\mathbf{y} - \mathbf{t}) \\ &= (\Phi^T \mathbf{R} \Phi)^{-1} \{ \Phi^T \mathbf{R} \Phi \mathbf{w}^{(\text{old})} - \Phi^T (\mathbf{y} - \mathbf{t}) \} \\ &= (\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T \mathbf{R} \mathbf{z}\end{aligned}$$

where $\mathbf{z} = \Phi \mathbf{w}^{(\text{old})} - \mathbf{R}^{-1} (\mathbf{y} - \mathbf{t})$

$\mathbf{w} \rightarrow \mathbf{R} \rightarrow \mathbf{w} \rightarrow \mathbf{R} \rightarrow \dots \Rightarrow$ iterative reweighted least squares

Multiclass Logistic Regression

$$p(\mathcal{C}_k | \phi) = y_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

where

$$a_k = \mathbf{w}_k^T \phi \quad \frac{\partial y_k}{\partial a_j} = y_k(I_{kj} - y_j)$$

1-of- K coding scheme : Using likelihood function

$$p(\mathbf{T} | \mathbf{w}_1, \dots, \mathbf{w}_K) = \prod_{n=1}^N \prod_{k=1}^K p(\mathcal{C}_k | \phi_n)^{t_{nk}} = \prod_{n=1}^N \prod_{k=1}^K y_{nk}^{t_{nk}}$$

$$\mathbf{T}_{N \times k} = [t_{nk}]$$

$$E(\mathbf{w}_1, \dots, \mathbf{w}_K) = -\ln p(\mathbf{T} | \mathbf{w}_1, \dots, \mathbf{w}_K) = -\sum_{n=1}^N \sum_{k=1}^K t_{nk} \ln y_{nk}$$

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \dots, \mathbf{w}_K) = \sum_{n=1}^N (y_{nj} - t_{nj}) \phi_n$$

$$\nabla_{\mathbf{w}_k} \nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \dots, \mathbf{w}_K) = -\sum_{n=1}^N y_{nk} (I_{kj} - y_{nj}) \phi_n \phi_n^T$$

The Laplace Approximation

- Bayesian treatment of logistic regression is important
- No closed-form solution to the integral over \mathbf{w}
- Laplace approximation is used to find a Gaussian approximation to a probability density defined over a set of continuous variables.

$$p(z) = \frac{f(z)}{[Z=\int f(z)dz]}$$

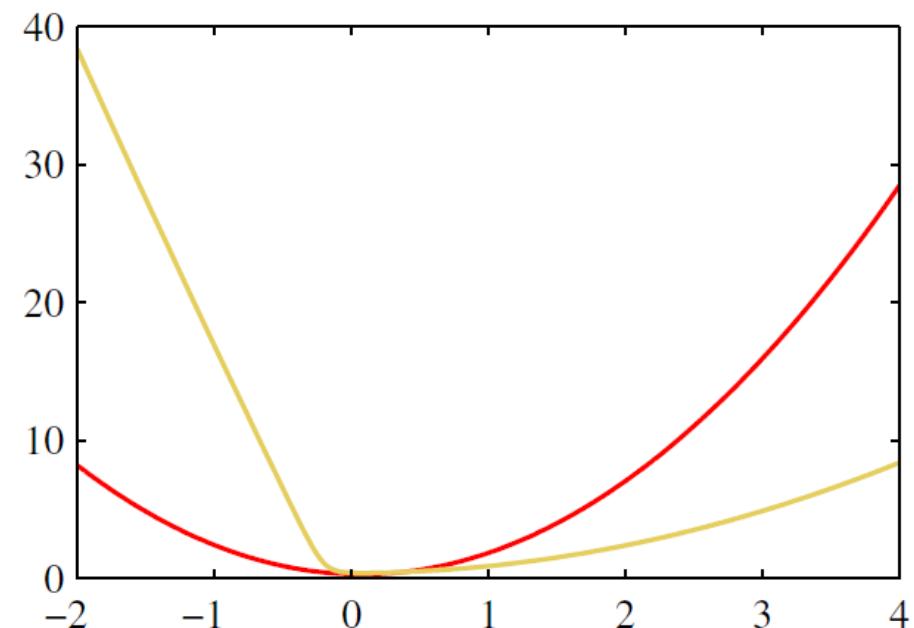
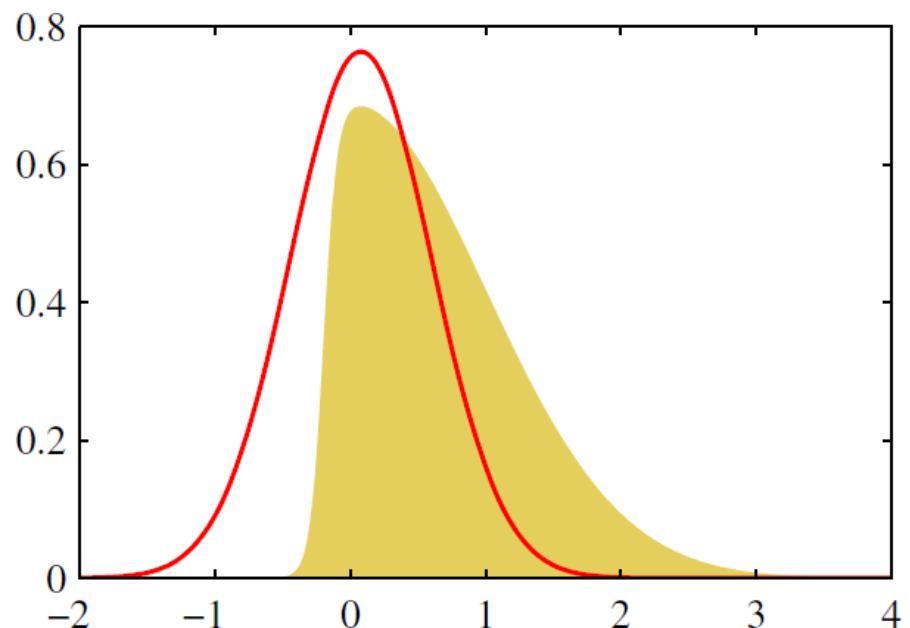
- We want to find a Gaussian approximation $q(z)$ which is centered on a “mode” of the distribution $p(z)$.

$$p'(z_0) = 0, \text{ i.e. } \left. \frac{df(z)}{dz} \right|_{z=z_0} = 0$$

- Consider a quadratic function via Taylor expansion

$$\ln f(z) \simeq \ln f(z_0) - \frac{1}{2} A(z - z_0)^2 \quad \therefore \left. \frac{d \ln f(z)}{dz} \right|_{z=z_0} = 0$$

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



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

- In the case of M-dimensional space $\mathbf{z} \in R^M$

$$\ln f(\mathbf{z}) \simeq \ln f(\mathbf{z}_0) - \frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^T \mathbf{A}(\mathbf{z} - \mathbf{z}_0) \quad \mathbf{A} = -\nabla \nabla \ln f(\mathbf{z})|_{\mathbf{z}=\mathbf{z}_0}$$

$$f(\mathbf{z}) \simeq f(\mathbf{z}_0) \exp \left\{ -\frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^T \mathbf{A}(\mathbf{z} - \mathbf{z}_0) \right\} \quad q(\mathbf{z}) = \frac{|\mathbf{A}|^{1/2}}{(2\pi)^{M/2}} \exp \left\{ -\frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^T \mathbf{A}(\mathbf{z} - \mathbf{z}_0) \right\}$$

- Laplace approximation is useful where number of data points ↗
- But, only continuous variable is applicable

$$\begin{aligned} Z &= \int f(\mathbf{z}) d\mathbf{z} \\ &\simeq f(\mathbf{z}_0) \int \exp \left\{ -\frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^T \mathbf{A}(\mathbf{z} - \mathbf{z}_0) \right\} d\mathbf{z} \\ &= f(\mathbf{z}_0) \frac{(2\pi)^{M/2}}{|\mathbf{A}|^{1/2}} \end{aligned}$$

Model Comparison and BIC

- Bayesian model comparison → evidence $Z = \int f(\mathbf{z}) d\mathbf{z}$

- Data set D a set of models $\{\mathcal{M}_i\}$ parameter $\{\boldsymbol{\theta}_i\}$

- Model evidence $p(D|\mathcal{M}_i)$

or $p(D) = \int p(D|\boldsymbol{\theta})p(\boldsymbol{\theta}) d\boldsymbol{\theta}$ $\frac{p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(D)} = p(\boldsymbol{\theta}|D) = \frac{f(\boldsymbol{\theta})}{[Z=\int f(\boldsymbol{\theta})d(\boldsymbol{\theta})]}$

$$\left. \begin{array}{l} f(\boldsymbol{\theta}) = p(D|\boldsymbol{\theta})p(\boldsymbol{\theta}) \\ Z = p(D) \end{array} \right\} Z \simeq f(\mathbf{z}_0) \frac{(2\pi)^{M/2}}{|\mathbf{A}|^{1/2}}$$

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

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

- Consider the **Gaussian prior** distribution is **broad**, and the Hessian has full rank, we have

$$\ln p(D) \simeq \ln p(D|\boldsymbol{\theta}_{\text{MAP}}) - \frac{1}{2} M \ln N$$

- **Bayesian Information Criterion (BIC)** (Schwarz, 1978)

Bayesian Logistic Regression

- Exact Bayesian inference for logistic regression is intractable including evaluation of the posterior distribution.

Laplace Approximation

- Gaussian prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{m}_0, \mathbf{S}_0)$
- Posterior $p(\mathbf{w} | \mathbf{t}) \propto p(\mathbf{w})p(\mathbf{t} | \mathbf{w})$

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

where $y_n = \sigma(\mathbf{w}^T \phi_n)$

- Gaussian approximation $q(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{w}_{\text{MAP}}, \mathbf{S}_N) \rightarrow p(\mathbf{w} | \mathbf{t})$

$$\frac{d}{d\mathbf{w}} \ln p(\mathbf{w} | \mathbf{t}) \Big|_{\mathbf{w}_{\text{MAP}}} = 0$$

$$\mathbf{S}_N = -\nabla \nabla \ln p(\mathbf{w} | \mathbf{t}) = \mathbf{S}_0^{-1} + \sum_{n=1}^N y_n(1 - y_n) \phi_n \phi_n^T$$

Predictive Distribution

$$p(\mathcal{C}_1|\phi, \mathbf{t}) = \int p(\mathcal{C}_1|\phi, \mathbf{w})p(\mathbf{w}|\mathbf{t}) d\mathbf{w} \simeq \int \sigma(\mathbf{w}^T \phi) q(\mathbf{w}) d\mathbf{w}$$

$$\sigma(\mathbf{w}^T \phi) = \int \delta(a - \mathbf{w}^T \phi) \sigma(a) da \quad \int \sigma(\mathbf{w}^T \phi) q(\mathbf{w}) d\mathbf{w} = \int \sigma(a) p(a) da$$

where $p(a) = \int \delta(a - \mathbf{w}^T \phi) q(\mathbf{w}) d\mathbf{w}$

$$\mu_a = \mathbb{E}[a] = \int p(a)a da = \int q(\mathbf{w}) \mathbf{w}^T \phi d\mathbf{w} = \mathbf{w}_{\text{MAP}}^T \phi$$

$$\begin{aligned}\sigma_a^2 &= \text{var}[a] = \int p(a) \{a^2 - \mathbb{E}[a]^2\} da \\ &= \int q(\mathbf{w}) \{(\mathbf{w}^T \phi)^2 - (\mathbf{m}_N^T \phi)^2\} d\mathbf{w} = \phi^T \mathbf{S}_N \phi\end{aligned}$$

- Variational approximation to the predictive distribution

$$p(\mathcal{C}_1|\mathbf{t}) = \int \sigma(a)p(a) da = \int \sigma(a)\mathcal{N}(a|\mu_a, \sigma_a^2) da$$

by using

$$\int \Phi(\lambda a) \mathcal{N}(a|\mu, \sigma^2) da = \Phi\left(\frac{\mu}{(\lambda^{-2} + \sigma^2)^{1/2}}\right)$$

$$\Phi(a) = \int_{-\infty}^a \mathcal{N}(\theta|0, 1) d\theta \quad \text{CDF of Gaussian}$$

$$\sigma(a) \simeq \Phi(\lambda a) \quad \lambda^2 = \pi/8$$

$$p(\mathcal{C}_1|\phi, \mathbf{t}) = \int \sigma(a) \mathcal{N}(a|\mu, \sigma^2) da \simeq \sigma(\kappa(\sigma^2)\mu)$$

where $\kappa(\sigma^2) = (1 + \pi\sigma^2/8)^{-1/2}$

- Decision boundary : $p(\mathcal{C}_1|\phi, \mathbf{t}) = 0.5 \rightarrow \mu_a = 0$

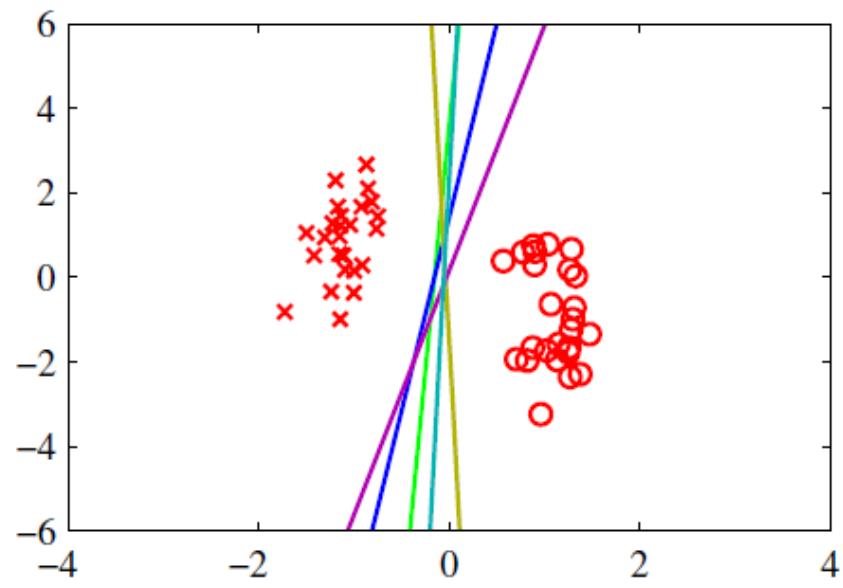
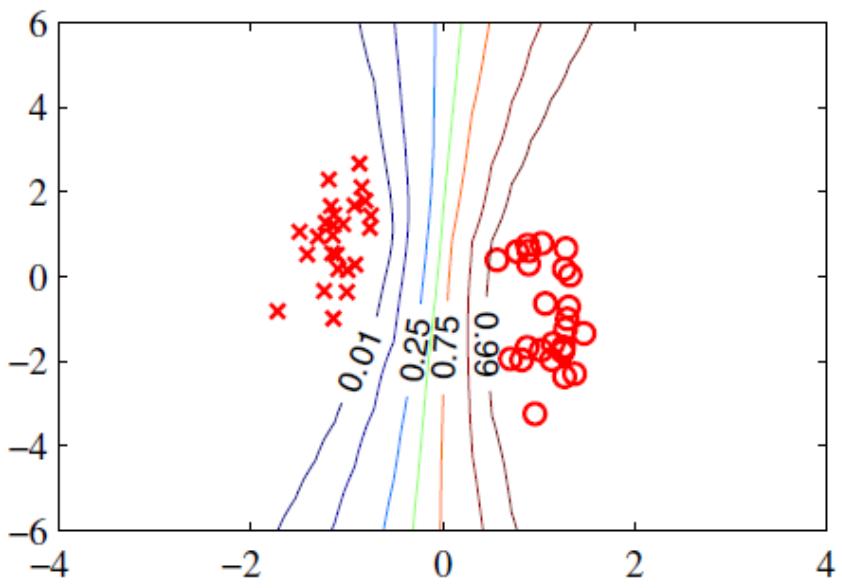


Figure 10.13 Illustration of the Bayesian approach to logistic regression for a simple linearly separable data set. The plot on the left shows the predictive distribution obtained using variational inference. We see that the decision boundary lies roughly mid way between the clusters of data points, and that the contours of the predictive distribution splay out away from the data reflecting the greater uncertainty in the classification of such regions. The plot on the right shows the decision boundaries corresponding to five samples of the parameter vector w drawn from the posterior distribution $p(w|t)$.

CONTENTS

1. Introduction
2. Probability Distributions
3. Linear Models for Regression
4. Linear Models for Classification
5. Kernel Methods
6. Sparse Kernel Methods
7. Mixture Models and EM
8. Approximate Inference

Feed-forward Network Functions

- Multilayer Perceptron (MLP)
- Mathematical representations of information processing in biological systems
- Linear models for regression and classification

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

- Neural network model

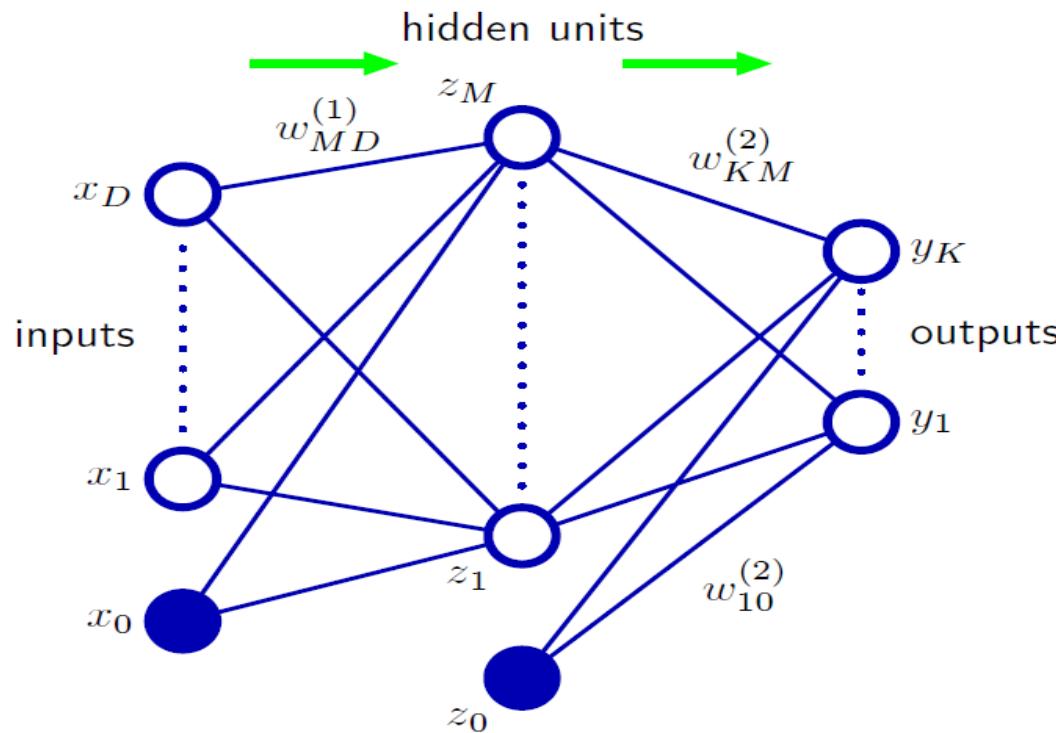
$$\mathbf{x} = (x_1, x_2, \dots, x_D)^T$$

$$a_j = \sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

$$z_j = h(a_j)$$

$$a_k = \sum_{j=1}^M w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$

where $j = 1, \dots, M$, $k = 1, \dots, K$



$$y_k = \sigma(a_k)$$

where $\sigma(a) = \frac{1}{1 + \exp(-a)}$, softmax activation function

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma \left(\sum_{j=1}^M w_{kj}^{(2)} h \left(\sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right)$$

- Neural network model is simply a nonlinear function from a set of inputs $\{x_i\}$ to a set of outputs $\{y_k\}$ controlled by $\mathbf{w} = \{\{w_{ji}^{(1)}\}, \{w_{kj}^{(2)}\}\}$.
 \Rightarrow deterministic variables for neural network
- We can give the additional input variables $x_0 = 1, z_0 = 1$

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma \left(\sum_{j=0}^M w_{kj}^{(2)} h \left(\sum_{i=0}^D w_{ji}^{(1)} x_i \right) \right)$$

- Extension from single-hidden-layer network (two-layer network) to **multilayer network**
- Extension to skip-layer connections

General Feed-forward Topology

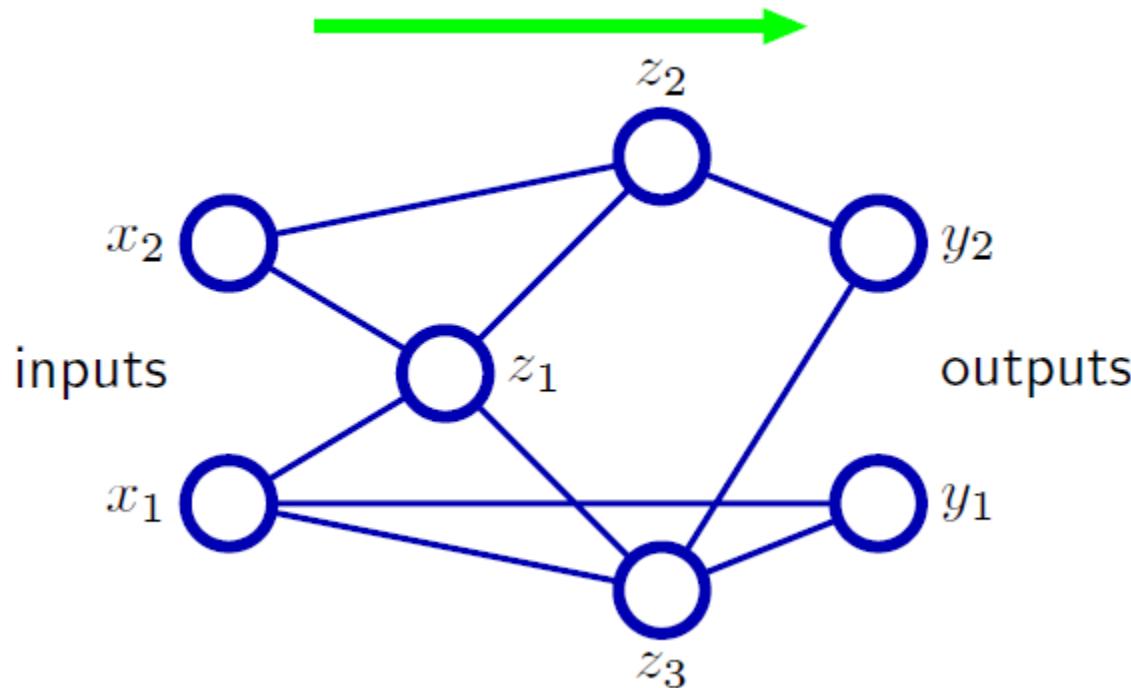
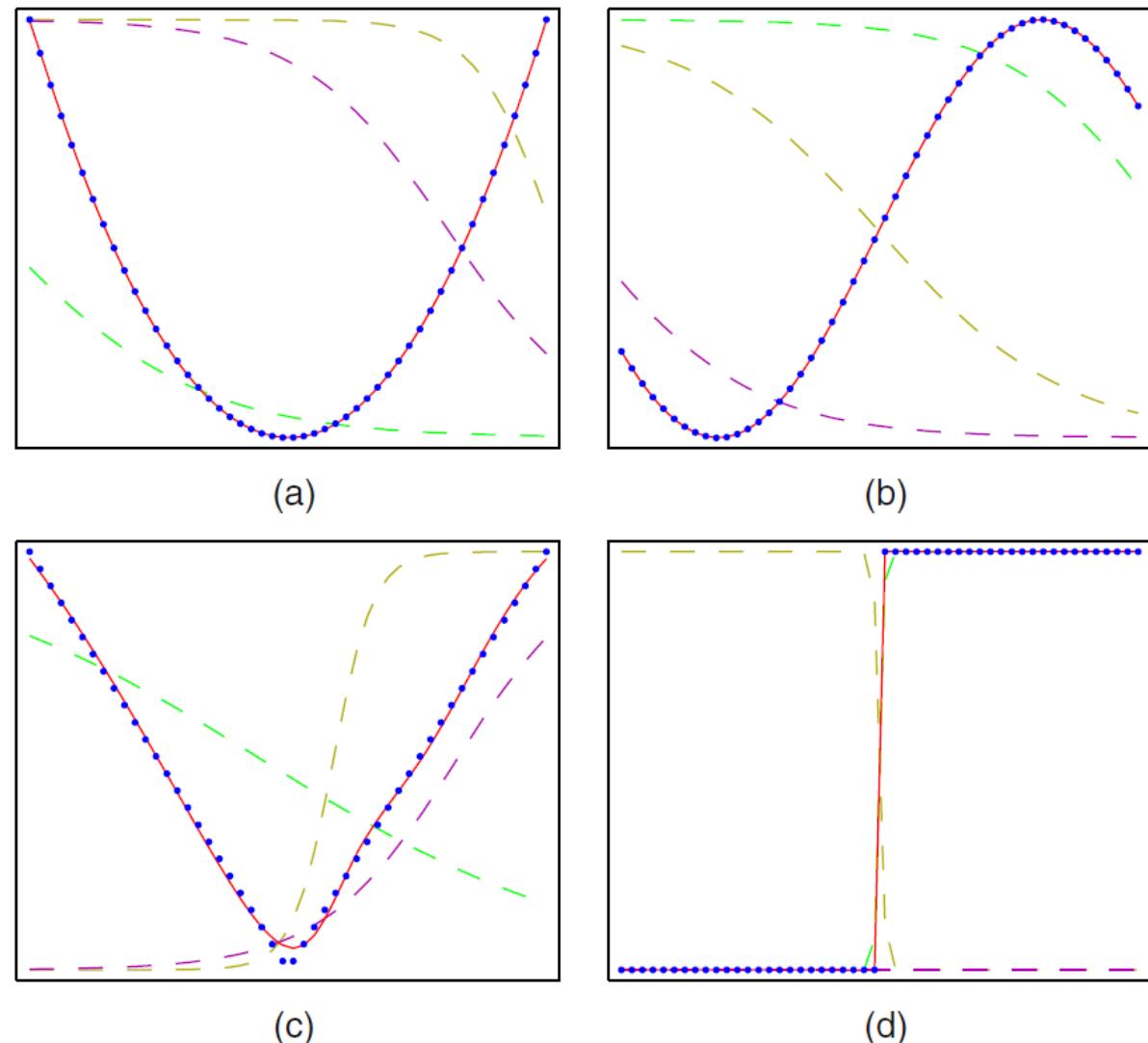
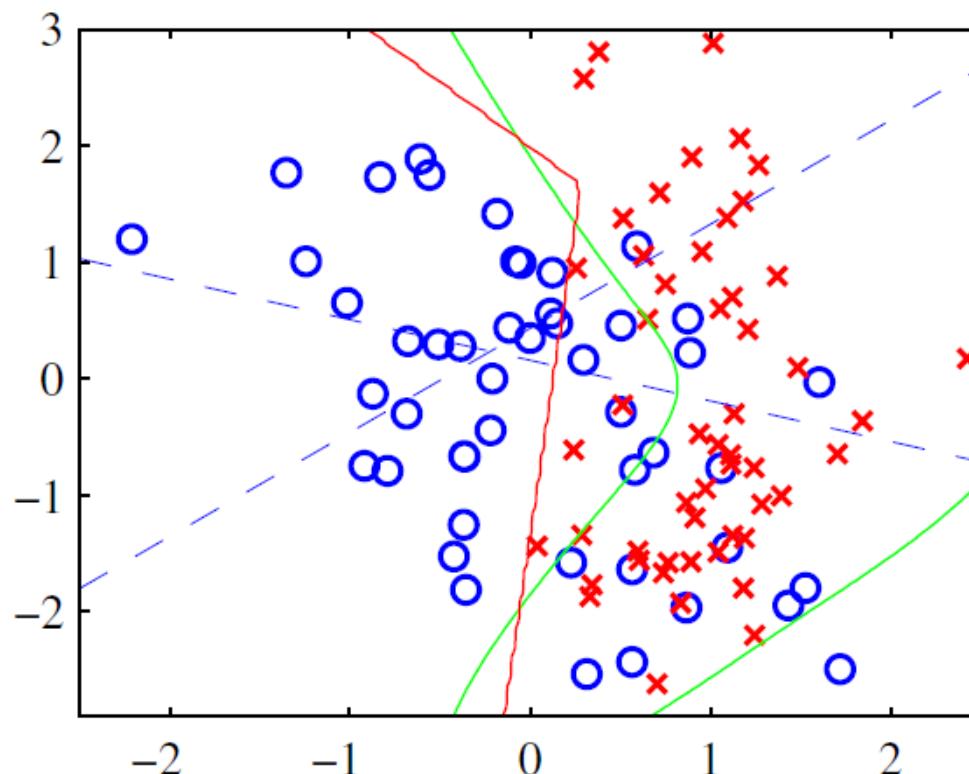


Figure 5.3 Illustration of the capability of a multilayer perceptron to approximate four different functions comprising (a) $f(x) = x^2$, (b) $f(x) = \sin(x)$, (c), $f(x) = |x|$, and (d) $f(x) = H(x)$ where $H(x)$ is the Heaviside step function. In each case, $N = 50$ data points, shown as blue dots, have been sampled uniformly in x over the interval $(-1, 1)$ and the corresponding values of $f(x)$ evaluated. These data points are then used to train a two-layer network having 3 hidden units with ‘tanh’ activation functions and linear output units. The resulting network functions are shown by the red curves, and the outputs of the three hidden units are shown by the three dashed curves.



Neural networks are said to be “universal approximators”

Figure 5.4 Example of the solution of a simple two-class classification problem involving synthetic data using a neural network having two inputs, two hidden units with ‘tanh’ activation functions, and a single output having a logistic sigmoid activation function. The dashed blue lines show the $z = 0.5$ contours for each of the hidden units, and the red line shows the $y = 0.5$ decision surface for the network. For comparison, the green line denotes the optimal decision boundary computed from the distributions used to generate the data.



Objective for Network Training

- Given input vectors $\{\mathbf{x}_n\}_{n=1}^N$ and target vector $\{\mathbf{t}_n\}_{n=1}^N$,
the minimization of a **sum-of-squares error** function

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \|\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \mathbf{t}_n\|^2$$

Relation to **maximum likelihood criterion**

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

$$= \mathcal{N}(t - y(\mathbf{x}, \mathbf{w})|0, \beta^{-1})$$

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^N p(t_n|\mathbf{x}_n, \mathbf{w}, \beta)$$

$$\mathbf{w}_{\text{ML}} = \operatorname{argmax}_{\mathbf{w}} p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta)$$

Taking the negative logarithm, we have

$$\frac{\beta}{2} \sum_{n=1}^N \{y(\mathbf{x}_n, \mathbf{w}) - t_n\}^2 - \frac{N}{2} \ln \beta + \frac{N}{2} \ln(2\pi)$$

ML \Leftrightarrow Minimization of sum-of-squares error function

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

$$\boxed{\frac{1}{\beta_{\text{ML}}} = \frac{1}{N} \sum_{n=1}^N \{y(\mathbf{x}_n, \mathbf{w}_{\text{ML}}) - t_n\}^2}$$

- For the case of multiple target variables

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

$$\frac{1}{\beta_{\text{ML}}} = \frac{1}{NK} \sum_{n=1}^N \|\mathbf{y}(\mathbf{x}_n, \mathbf{w}_{\text{ML}}) - \mathbf{t}_n\|^2$$

Cross Entropy Error Function

- Consider a **binary** classification, let

$$y(\mathbf{x}, \mathbf{w}) = p(C_1 | \mathbf{x})$$

$$1 - y(\mathbf{x}, \mathbf{w}) = p(C_2 | \mathbf{x})$$

$$p(t | \mathbf{x}, \mathbf{w}) = y(\mathbf{x}, \mathbf{w})^t \{1 - y(\mathbf{x}, \mathbf{w})\}^{1-t} \quad \begin{array}{l} t = 1 \rightarrow C_1 \\ t = 0 \rightarrow C_2 \end{array}$$

- Cross entropy error function**

$$E(\mathbf{w}) = - \sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}$$

- In case of **K binary classifications**,

$$p(\mathbf{t} | \mathbf{x}, \mathbf{w}) = \prod_{k=1}^K y_k(\mathbf{x}, \mathbf{w})^{t_k} [1 - y_k(\mathbf{x}, \mathbf{w})]^{1-t_k}$$

- Taking the negative logarithm of the likelihood function then gives the cross-entropy error function

$$E(\mathbf{w}) = - \sum_{n=1}^N \sum_{k=1}^K \{t_{nk} \ln y_{nk} + (1 - t_{nk}) \ln(1 - y_{nk})\}$$

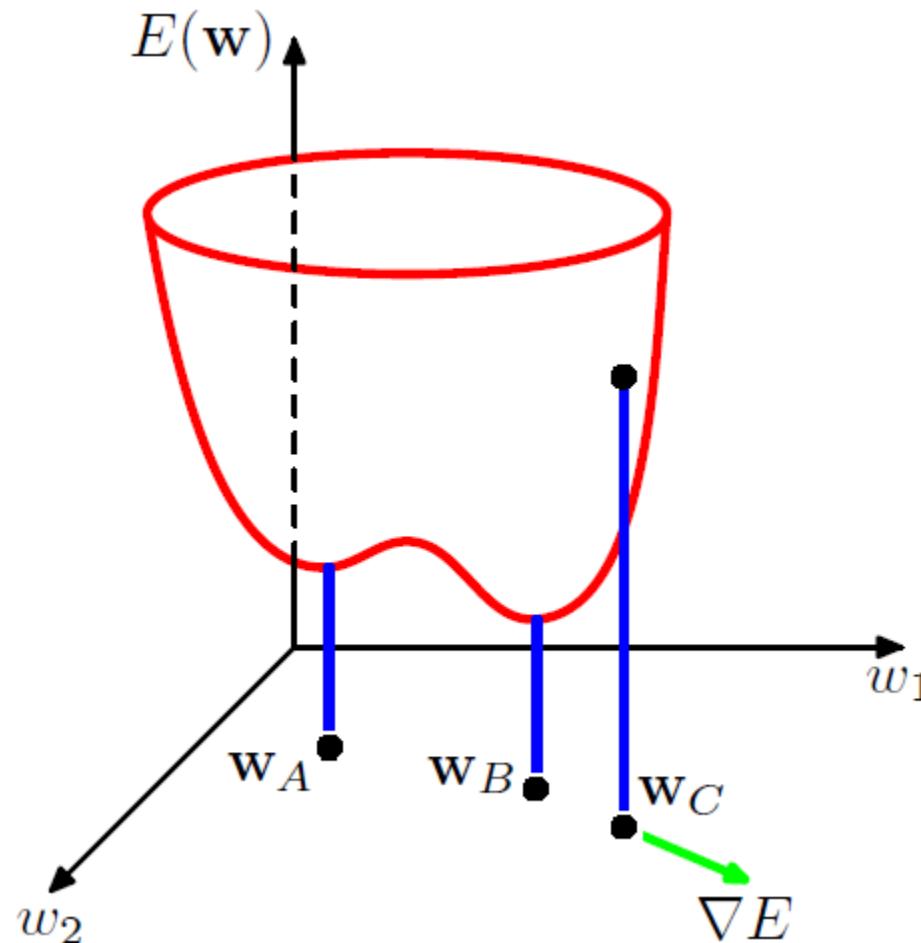
where $y_{nk} = y_k(\mathbf{x}_n, \mathbf{w})$

- In case of standard **multiclass** classification, **1-of- K** coding scheme $y_k(\mathbf{x}, \mathbf{w}) = p(t_k = 1 | \mathbf{x})$, $\sum_k y_k = 1$

$$E(\mathbf{w}) = - \sum_{n=1}^N \sum_{k=1}^K t_{kn} \ln y_k(\mathbf{x}_n, \mathbf{w})$$

$$y_k(\mathbf{x}, \mathbf{w}) = \frac{\exp(a_k(\mathbf{x}, \mathbf{w}))}{\sum_j \exp(a_j(\mathbf{x}, \mathbf{w}))}$$

Geometrical view of the error function $E(\mathbf{w})$ as a surface sitting over weight space. Point \mathbf{w}_A is a local minimum and \mathbf{w}_B is the global minimum. At any point \mathbf{w}_C , the local gradient of the error surface is given by the vector ∇E .



Parameter optimization

$$\mathbf{w} \rightarrow \mathbf{w} + \delta \mathbf{w}$$

$$\delta E \simeq \delta \mathbf{w}^T \nabla E(\mathbf{w})$$

- Smallest value will occur at a point which $\nabla E(\mathbf{w}) = 0$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \Delta \mathbf{w}^{(\tau)}$$

Local quadratic approximation

$\hat{\mathbf{w}}$ is some point around \mathbf{w} , the Taylor expansion

$$E(\mathbf{w}) \simeq E(\hat{\mathbf{w}}) + (\mathbf{w} - \hat{\mathbf{w}})^T \mathbf{b} + \frac{1}{2} (\mathbf{w} - \hat{\mathbf{w}})^T \mathbf{H} (\mathbf{w} - \hat{\mathbf{w}})$$

$$\mathbf{b} \equiv \nabla E|_{\mathbf{w}=\hat{\mathbf{w}}}$$

$$\mathbf{H} = \nabla \nabla E \quad (\mathbf{H})_{ij} \equiv \left. \frac{\partial^2 E}{\partial w_i \partial w_j} \right|_{\mathbf{w}=\hat{\mathbf{w}}}$$

- We obtain $\nabla E \simeq \mathbf{b} + \mathbf{H}(\mathbf{w} - \widehat{\mathbf{w}})$
- Let \mathbf{w}^* be a local minimum and $(\nabla E)_{\mathbf{w}^*} = 0$

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

- For geometric interpretation, we can perform eigen-analysis

$$\mathbf{H}\mathbf{u}_i = \lambda_i \mathbf{u}_i$$

where $\{\mathbf{u}_i\}$ are orthonormal vectors, i.e., $\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}$

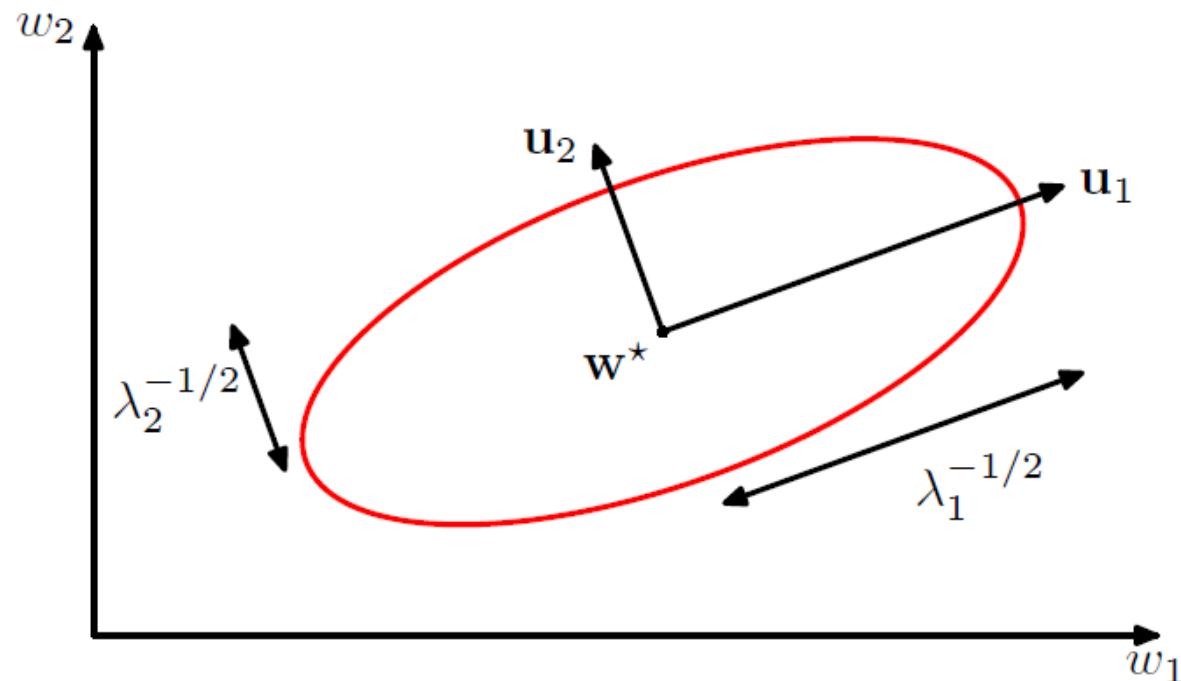
- We expand $\mathbf{w} - \mathbf{w}^* = \sum_i \alpha_i \mathbf{u}_i$, then

$$\begin{aligned} E(\mathbf{w}) &= E(\mathbf{w}^*) + \frac{1}{2} (\sum_i \alpha_i \mathbf{u}_i)^T \mathbf{H} (\sum_i \alpha_i \mathbf{u}_i) \\ &= E(\mathbf{w}^*) + \frac{1}{2} \sum_i \lambda_i \alpha_i^2. \end{aligned}$$

- H is **positive definite** if, and only if,

$$\mathbf{v}^T \mathbf{H} \mathbf{v} > 0 \text{ for all } \mathbf{v} \neq 0$$

or if $\{\lambda_i\}$ are all positive.



Gradient Descent Optimization

- Batch gradient descent

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

where the parameter η is the learning rate.

- Sequential gradient descent method

If

$$E(\mathbf{w}) = \sum_{n=1}^N E_n(\mathbf{w})$$

then

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

Error Backpropagation Algorithm

- We try to find an efficient way of evaluating $\nabla E(\mathbf{w})$ for a feed-forward neural network.

$$E(\mathbf{w}) = \sum_{n=1}^N E_n(\mathbf{w})$$

$$E_n = \frac{1}{2} \sum_k (y_{nk} - t_{nk})^2$$

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}$$

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i$$

- In a general feed-forward network, each unit computes a

$$a_j = \sum_i w_{ji} z_i$$

Evaluation of Error-Function Derivatives

send a connection to j , $z_j = h(a_j)$

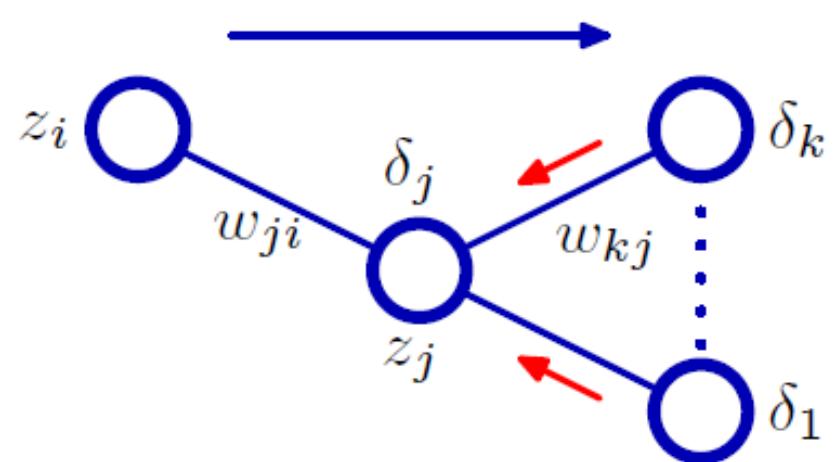
$$\delta_j \equiv \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j}$$

$$= h'(a_j) \sum_k w_{kj} \delta_k$$

$$\delta_k = y_k - t_k$$

- Finally,

$$\frac{\partial E}{\partial w_{ji}} = \sum_n \frac{\partial E_n}{\partial w_{ji}}$$



Error Backpropagation Procedure

1. Apply x_n to network by general computation

$$a_j = \sum_i w_{ji} z_i \quad \& \quad z_j = h(a_j)$$

Find all activations of all **hidden** and **output** units

2. Evaluate $\delta_k = y_k - t_k$ for all **output** nodes

3. **Backpropagate** the δ 's using

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k$$

4. Find the required derivatives $\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i$

for all weights in **first** layer $\{w_{ji}^{(1)}\}$ and **second** layer $\{w_{kj}^{(2)}\}$

Logistic Sigmoid Activation Function

$$h(a) \equiv \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$$

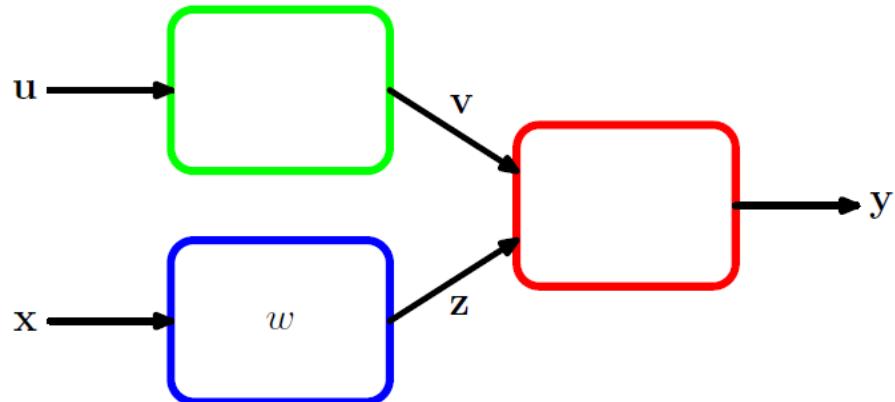
$$\begin{aligned} h'(a) &= \frac{(e^a + e^{-a})(e^a + e^{-a}) - (e^a - e^{-a})(e^a - e^{-a})}{(e^a + e^{-a})^2} \\ &= 1 - h(a)^2 \end{aligned}$$

The Jacobian Matrix

- The Jacobian matrix, defined below, provides a measure

$$J = [J_{ki}] \equiv \left[\frac{\partial y_k}{\partial x_i} \right]$$

of the **local sensitivity** of the outputs to changes in each of input variables.



$$\frac{\partial E}{\partial w} = \sum_{k,j} \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial z_j} \frac{\partial z_j}{\partial w}$$

$$\Delta y_k \simeq \sum_i \frac{\partial y_k}{\partial x_i} \Delta x_i$$

$$\begin{aligned} J_{ki} = \frac{\partial y_k}{\partial x_i} &= \sum_j \frac{\partial y_k}{\partial a_j} \frac{\partial a_j}{\partial x_i} \\ &= \sum_j w_{ji} \frac{\partial y_k}{\partial a_j} \end{aligned}$$

- Recursive backpropagation formula

$$\begin{aligned}\frac{\partial y_k}{\partial a_j} &= \sum_l \frac{\partial y_k}{\partial a_l} \frac{\partial a_l}{\partial a_j} \\ &= h'(a_j) \sum_l w_{lj} \frac{\partial y_k}{\partial a_l}\end{aligned}$$

- A procedure of evaluating the Jacobian matrix can be summarized.

The Hessian Matrix

- Backpropagation algorithm for the first derivatives can be extended to that for the second derivatives of the error, given by

$$\frac{\partial^2 E}{\partial w_{ji} \partial w_{lk}}$$

$$\mathbf{H} = [\mathbf{H}_{ij}] = [\frac{\partial^2 E}{\partial w_i \partial w_j}]_{\mathbf{w} \times \mathbf{w}}$$

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

- Hessian matrix plays an important role in many aspects of neural computing. Various approximation schemes have been used to evaluate the Hessian matrix for a neural network.
- Hessian exact computation is also available with cost $O(W^2)$ where W is the number of weights.

Diagonal Approximation

- For this case, the **diagonal** elements are computed by

$$\frac{\partial^2 E_n}{\partial w_{ji}^2} = \frac{\partial^2 E_n}{\partial a_j^2} z_i^2 \quad (\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}})$$

$$\delta_j = \frac{\partial E_n}{\partial a_j} = h'(a_j) \sum_k w_{kj} \delta_k$$

$$\frac{\partial^2 E_n}{\partial a_j^2} = h'(a_j)^2 \sum_k \sum_{k'} w_{kj} w_{k'j} \frac{\partial^2 E_n}{\partial a_k \partial a_{k'}} + h''(a_j) \sum_k w_{kj} \frac{\partial E_n}{\partial a_k}$$

- If we neglect off-diagonal elements in the second-derivative terms, we obtain

$$\frac{\partial^2 E_n}{\partial a_j^2} = h'(a_j)^2 \sum_k w_{kj}^2 \frac{\partial^2 E_n}{\partial a_k^2} + h''(a_j) \sum_k w_{kj} \frac{\partial E_n}{\partial a_k}$$

Outer Product Approximation

- In case of a single output

$$E = \frac{1}{2} \sum_{n=1}^N (y_n - t_n)^2$$

$$\mathbf{H} = \nabla \nabla E = \sum_{n=1}^N \nabla y_n \nabla^T y_n + \sum_{n=1}^N (y_n - t_n) \nabla \nabla y_n$$

$$(\nabla E = \sum_{n=1}^N (y_n - t_n) \nabla y_n)$$

- If the network is well trained, $y_n \rightarrow t_n$, we have the outer-product approximation

$$\mathbf{H} \cong \sum_{n=1}^N (\nabla y_n \nabla^T y_n) = \sum_{n=1}^N \mathbf{b}_n \mathbf{b}_n^T$$

- In case of **binary classification**

$$E(\mathbf{w}) = - \sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}$$

$$\mathbf{H} \simeq \sum_{n=1}^N y_n(1 - y_n) \mathbf{b}_n \mathbf{b}_n^T$$

Inverse Hessian

- Using outer-product approximation,

$$\mathbf{H}_N = \sum_{n=1}^N \mathbf{b}_n \mathbf{b}_n^T$$

where $\mathbf{b}_n \equiv \nabla_{\mathbf{w}} a_n$

- Sequential procedure for building Hessian, for the first L data points,

$$\mathbf{H}_{L+1} = \mathbf{H}_L + \mathbf{b}_{L+1} \mathbf{b}_{L+1}^T$$

- Using the Woodbury identity

$$(\mathbf{M} + \mathbf{v}\mathbf{v}^T)^{-1} = \mathbf{M}^{-1} - \frac{(\mathbf{M}^{-1}\mathbf{v})(\mathbf{v}^T\mathbf{M}^{-1})}{1 + \mathbf{v}^T\mathbf{M}^{-1}\mathbf{v}}$$

we obtain

$$\boxed{\mathbf{H}_{L+1}^{-1} = \mathbf{H}_L^{-1} - \frac{\mathbf{H}_L^{-1}\mathbf{b}_{L+1}\mathbf{b}_{L+1}^T\mathbf{H}_L^{-1}}{1 + \mathbf{b}_{L+1}^T\mathbf{H}_L^{-1}\mathbf{b}_{L+1}}}$$

Inverse Hessian

for $L = 1, 2, \dots, N$. Finally \mathbf{H}_N^{-1} is calculated.

- Initial value $\mathbf{H}_0 = \alpha \mathbf{I}$ for a small α , we actually find $(\mathbf{H} + \alpha \mathbf{I})^{-1}$.

Finite Differences

$$\frac{\partial^2 E}{\partial w_{ji} \partial w_{lk}} = \frac{1}{4\epsilon^2} \{ E(w_{ji} + \epsilon, w_{lk} + \epsilon) - E(w_{ji} + \epsilon, w_{lk} - \epsilon) \\ - E(w_{ji} - \epsilon, w_{lk} + \epsilon) + E(w_{ji} - \epsilon, w_{lk} - \epsilon) \} + O(\epsilon^2).$$

$$\frac{\partial^2 E}{\partial w_{ji} \partial w_{lk}} = \frac{1}{2\epsilon} \left\{ \frac{\partial E}{\partial w_{ji}}(w_{lk} + \epsilon) - \frac{\partial E}{\partial w_{ji}}(w_{lk} - \epsilon) \right\} + O(\epsilon^2)$$

Exact Evaluation of the Hessian

$$\delta_k = \frac{\partial E_n}{\partial a_k}, \quad M_{kk'} \equiv \frac{\partial^2 E_n}{\partial a_k \partial a_{k'}}$$

1. Both weights in the **second** layer

$$\frac{\partial^2 E_n}{\partial w_{kj}^{(2)} \partial w_{k'j'}^{(2)}} = z_j z_{j'} M_{kk'}$$

2. Both weights in the **first** layer

$$\begin{aligned} \frac{\partial^2 E_n}{\partial w_{ji}^{(1)} \partial w_{j'i'}^{(1)}} &= x_i x_{i'} h''(a_{j'}) I_{jj'} \sum_k w_{kj'}^{(2)} \delta_k \\ &+ x_i x_{i'} h'(a_{j'}) h'(a_j) \sum_k \sum_{k'} w_{k'j'}^{(2)} w_{kj}^{(2)} M_{kk'} \end{aligned}$$

Exact evaluation of the Hessian

3. One weight in each layer

$$\frac{\partial^2 E_n}{\partial w_{ji}^{(1)} \partial w_{kj'}^{(2)}} = x_i h'(a_j) \left\{ \delta_k I_{jj'} + z_{j'} \sum_{k'} w_{k'j}^{(2)} M_{kk'} \right\}$$

CONTENTS

1. Introduction
2. Probability Distributions
3. Linear Models for Regression
4. Linear Models for Classification
5. Kernel Methods
6. Sparse Kernel Methods
7. Mixture Models and EM
8. Approximate Inference

Kernel Methods

- In previous methods, we obtain point estimator of the parameter vector or determine a posterior distribution over the vector. Training data is discarded, and predictions are based on the learned parameter θ .
- See text *memory-based* methods
- They require a metric of measuring the similarity of any two vectors → fast to train but slow at making predictions for test data points.
- Many linear parametric models can be re-cast into an equivalent “dual representation”, predictions are based on linear combinations of a *kernel function* evaluated at training samples.

6. Kernel Methods

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$$

- See text large-margin classifier → *support vector machines*
- simplest Kernel function $\kappa(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$ (liner Kernel)

$k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$ stationary Kernel : invariant to translation in input space.

$k(\mathbf{x}, \mathbf{x}') = k(||\mathbf{x}', \mathbf{x}||)$ homogeneous kernel or radial basis function : depend only on the magnitude of distance

6.1. Dual Representations

$$J(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{ \mathbf{w}^T \phi(\mathbf{x}_n) - t_n \}^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}$$

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = 0 \quad \mathbf{w}_{\text{opt}} = -\frac{1}{\lambda} \sum_{n=1}^N \{ \mathbf{w}^T \phi(\mathbf{x}_n) - t_n \} \phi(\mathbf{x}_n)$$

$$= \sum_{n=1}^N a_n \phi(\mathbf{x}_n) = \Phi^T \mathbf{a}$$

- Where $\Phi^T = [\Phi_1^T(\mathbf{x}_1), \dots, \Phi_1^T(\mathbf{x}_N)]$ $\mathbf{a} = (a_1, \dots, a_N)^T$

$$a_n = -\frac{1}{\lambda} \{ \mathbf{w}^T \phi(\mathbf{x}_n) - t_n \}$$

- If we substitute $\mathbf{w} = \Phi^T$ into $J(\mathbf{w})$, we obtain

6.1. Dual Representations

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^T \Phi \Phi^T \Phi \Phi^T \mathbf{a} - \mathbf{a}^T \Phi \Phi^T \mathbf{t} + \frac{1}{2} \mathbf{t}^T \mathbf{t} + \frac{\lambda}{2} \mathbf{a}^T \Phi \Phi^T \mathbf{a}$$
$$\mathbf{t} = (t_1, \dots, t_N)^T$$

- We define a Gram matrix $\mathbf{K} = \Phi \Phi^T$ with elements

$$K_{nm} = \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$$

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^T \mathbf{K} \mathbf{K} \mathbf{a} - \mathbf{a}^T \mathbf{K} \mathbf{t} + \frac{1}{2} \mathbf{t}^T \mathbf{t} + \frac{\lambda}{2} \mathbf{a}^T \mathbf{K} \mathbf{a}$$

$$\nabla_{\mathbf{a}} J(\mathbf{a}) = 0 \quad \mathbf{a}_{\text{opt}} = (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}$$

- Linear regression model becomes

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) = \mathbf{a}^T \Phi \phi(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}$$

6.1. Dual Representations

- The least-squares solution can be entirely expressed in terms of the kernel function $k(\mathbf{x}, \mathbf{x}')$ → Dual formulation avoid expressing by $\phi(\mathbf{x})$, Entirely using $k(\mathbf{x}, \mathbf{x}')$

6.2. Constructing Kernels

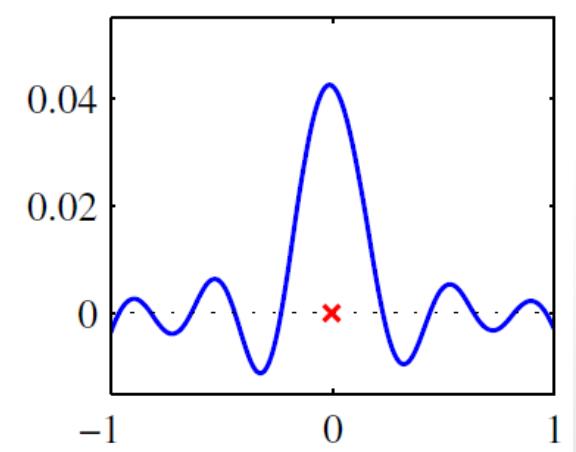
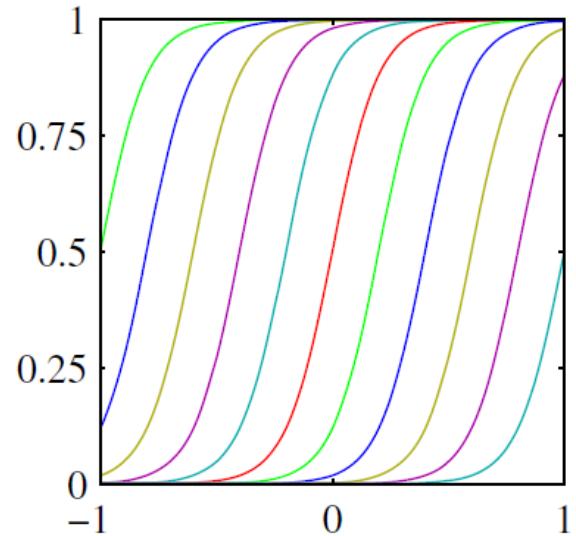
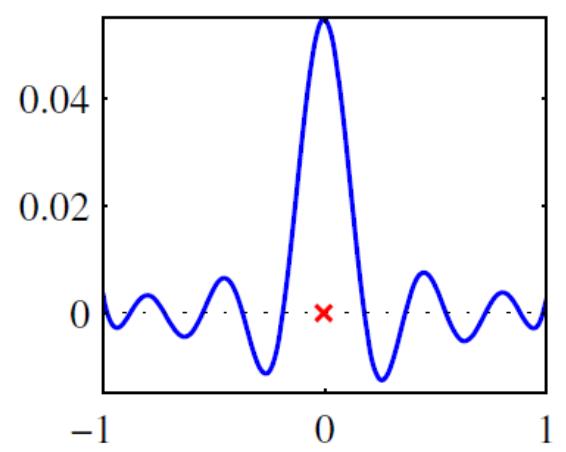
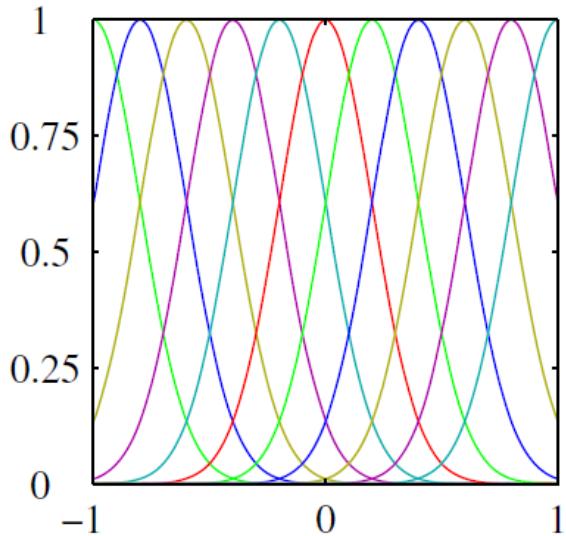
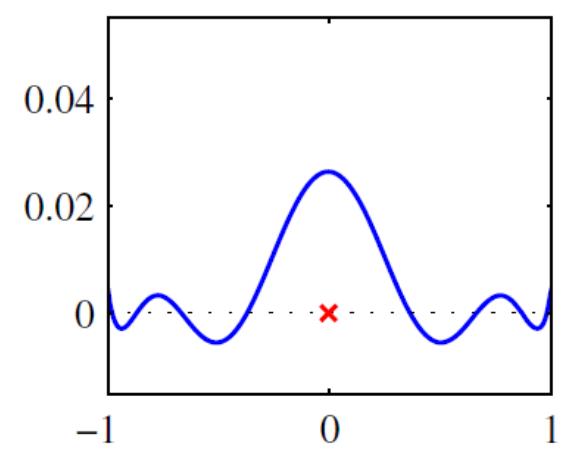
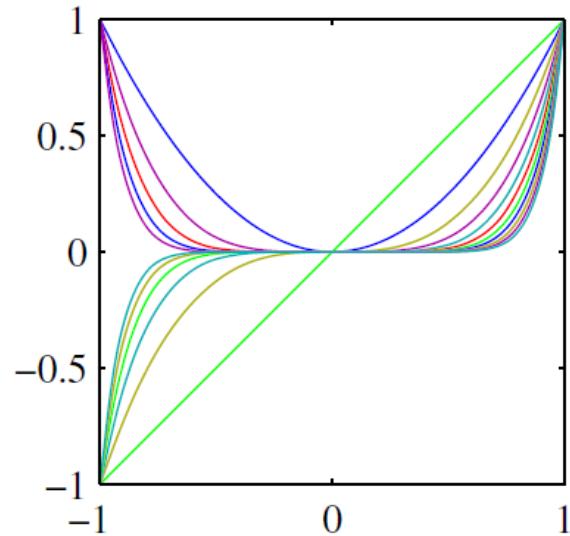
$$k(x, x') = \phi(x)^T \phi(x') = \sum_{i=1}^M \phi_i(x) \phi_i(x')$$

$\mathbf{x}' = 0$ for different \mathbf{x}

kernel functions ↔ basis functions

- In particular, $k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z})^2$ scalar product
 $= (x_1 z_1 + x_2 z_2)^2$ two-dim input space

6.2. Constructing Kernels



6.2. Constructing Kernels

- We have $\phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T$ comprises all possible second order terms
- K is a valid kernel $\rightarrow K$ should be “positive semidefinite”

Techniques for Constructing New Kernels.

$$k(\mathbf{x}, \mathbf{x}') = ck_1(\mathbf{x}, \mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = q(k_1(\mathbf{x}, \mathbf{x}'))$$

$$k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}'))$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = k_3(\phi(\mathbf{x}), \phi(\mathbf{x}'))$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{A} \mathbf{x}'$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b)$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a)k_b(\mathbf{x}_b, \mathbf{x}'_b)$$

$\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$ not necessarily disjoint

$k(\mathbf{x}, \mathbf{x}')$, $k_2(\mathbf{x}, \mathbf{x}')$ are valid kernels

6.2. Constructing Kernels

- If \mathbf{x}, \mathbf{x}' are two images $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}')^M$
→ weighted sum of all possible products of M pixels in \mathbf{x} &
 M pixels in \mathbf{x}'
- Also, Gaussian kernel is popular

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2)$$

- It is not interpreted as a probability density, normalization term is omitted.

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\mathbf{x}^T \mathbf{x} / 2\sigma^2) \exp(\mathbf{x}^T \mathbf{x}' / \sigma^2) \exp(-(\mathbf{x}')^T \mathbf{x}' / 2\sigma^2)$$

→ feature vector $\phi(\mathbf{x})$ or $\phi(\mathbf{x}')$ has infinite dimensionality.

- By replacing $\mathbf{x}^T \mathbf{x}' \rightarrow \kappa(\mathbf{x}, \mathbf{x}')$ nonlinear kernel

$$k(\mathbf{x}, \mathbf{x}') = \exp \left\{ -\frac{1}{2\sigma^2} (\kappa(\mathbf{x}, \mathbf{x}) + \kappa(\mathbf{x}', \mathbf{x}') - 2\kappa(\mathbf{x}, \mathbf{x}')) \right\}$$

6.2. Constructing Kernels

- Probabilistic generative model → discriminative model
- Given a generative model $p(\mathbf{x})$, we can define
 $k(\mathbf{x}, \mathbf{x}') = p(\mathbf{x})p(\mathbf{x}')$ ↗ when \mathbf{x} and \mathbf{x}' have high probabilities
- We can consider sums over products of different probability distributions

$$k(\mathbf{x}, \mathbf{x}') = \sum_i p(\mathbf{x}|i)p(\mathbf{x}'|i)p(i) \quad i : \text{latent variable}$$

- Also, $k(\mathbf{x}, \mathbf{x}') = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{x}'|\mathbf{z})p(\mathbf{z}) d\mathbf{z}$ \mathbf{z} : continuous latent variable

- In case of sequence data $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_L\}$

e.g. HMM, $k(\mathbf{X}, \mathbf{X}') = \sum_{\mathbf{Z}} p(\mathbf{X}|\mathbf{Z})p(\mathbf{X}'|\mathbf{Z})p(\mathbf{Z})$

6.2. Constructing Kernels

- Fisher kernel: given $p(\mathbf{x}|\theta)$, we can compute

$$\mathbf{g}(\theta, \mathbf{x}) = \nabla_{\theta} \ln p(\mathbf{x}|\theta) \quad \text{Fisher score}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{g}(\theta, \mathbf{x})^T \mathbf{F}^{-1} \mathbf{g}(\theta, \mathbf{x}') \quad \text{Fisher kernel}$$

Where $\mathbf{F} = \mathbb{E}_{\mathbf{x}} [\mathbf{g}(\theta, \mathbf{x}) \mathbf{g}(\theta, \mathbf{x})^T]$ Fisher information matrix
Information geometry

Fisher kernel is invariant by $\theta \rightarrow \psi(\theta)$ nonlinear re-parameterization

$$\mathbf{F} \simeq \frac{1}{N} \sum_{n=1}^N \mathbf{g}(\theta, \mathbf{x}_n) \mathbf{g}(\theta, \mathbf{x}_n)^T \quad \text{Sample average covariance matrix}$$

6.2. Constructing Kernels

- For simplicity, we can use $k(\mathbf{x}, \mathbf{x}') = \mathbf{g}(\boldsymbol{\theta}, \mathbf{x})^T \mathbf{g}(\boldsymbol{\theta}, \mathbf{x}')$
→ noninvariant kernel
- Also, $k(\mathbf{x}, \mathbf{x}') = \tanh(a\mathbf{x}^T \mathbf{x}' + b)$
sigmoidal kernel
support vector machine \leftrightarrow neural network

6.3. Radial Basis Function Networks

- Radial Basis Function Networks \leftrightarrow Mixture Density Networks

Each basis function depends only on the radial distance from a center $\mu_j \rightarrow \phi_j(\mathbf{x}) = h(\|\mathbf{x} - \mu_j\|)$

$$f(\mathbf{x}) = \sum_{n=1}^N w_n h(\|\mathbf{x} - \mathbf{x}_n\|) \quad \begin{matrix} \{\mathbf{x}_1, \dots, \mathbf{x}_N\} \\ \{t_1, \dots, t_N\} \end{matrix}$$

$f(\mathbf{x}_n) = t_n$ $\{w_n\}$ are found by least squares

RBF comes from the interpolation problem when the inputs are noisy.

$$E = \frac{1}{2} \sum_{n=1}^N \int \{y(\mathbf{x}_n + \xi) - t_n\}^2 \nu(\xi) d\xi \quad \xi \text{ is noise}$$

6.3. Radial Basis Function Networks

$$y(\mathbf{x}_n) = \sum_{n=1}^N t_n h(\mathbf{x} - \mathbf{x}_n)$$

Use calculus
of variations

- We obtain $y(\mathbf{x}_n) = \sum_{n=1}^N t_n h(\mathbf{x} - \mathbf{x}_n)$

- Where
$$h(\mathbf{x} - \mathbf{x}_n) = \frac{\nu(\mathbf{x} - \mathbf{x}_n)}{\sum_{n=1}^N \nu(\mathbf{x} - \mathbf{x}_n)}$$

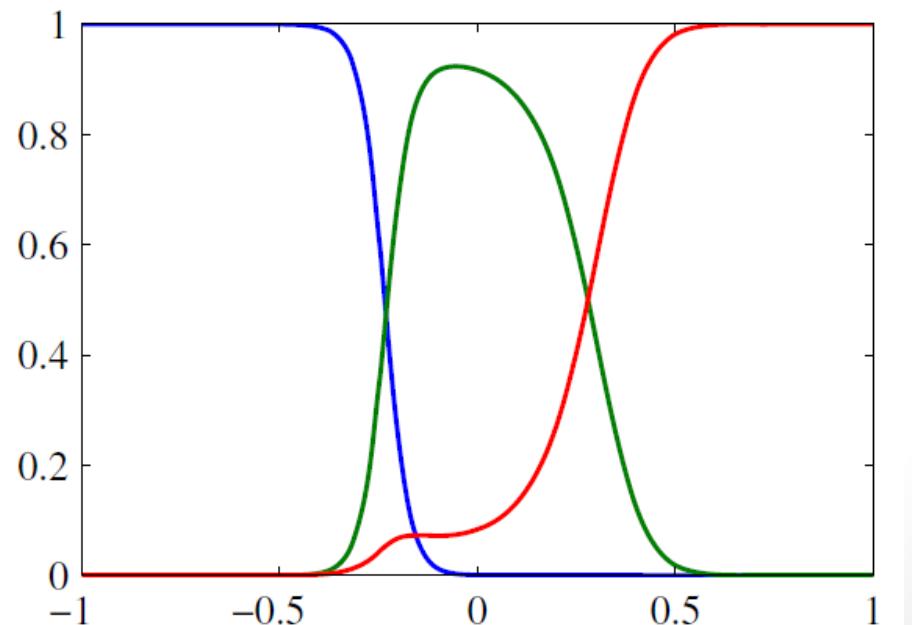
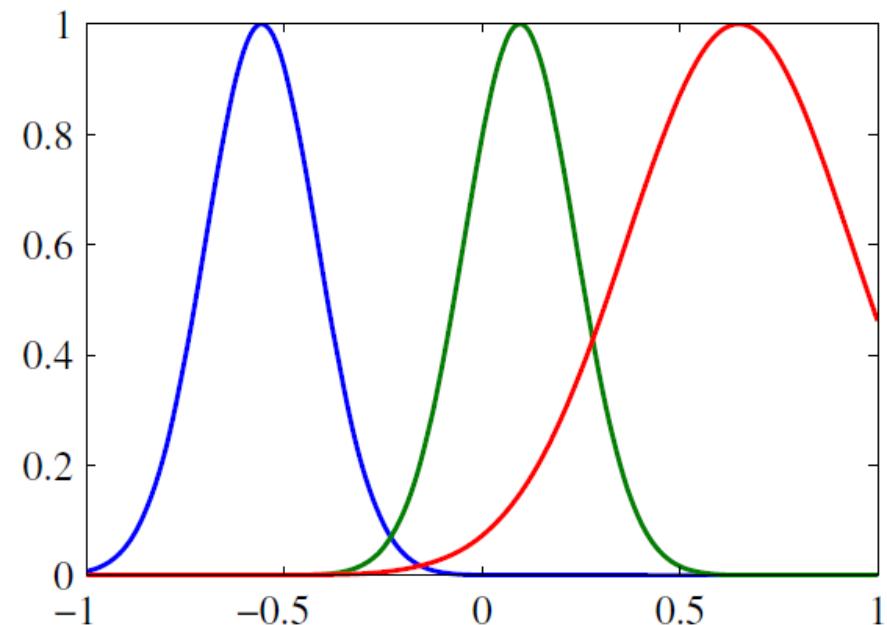
In case of isotropic
noise

- This $y(\mathbf{x}_n)$ is called *Nadaraya-Watson* model

$$\sum_n h(\mathbf{x} - \mathbf{x}_n) = 1 \quad \text{normalized}$$

6.3. Radial Basis Function Networks

- Fig 6.2



6.3. Radial Basis Function Networks

- Find μ_i from $\{\mathbf{x}_n\}$ & then $\{w_i\}$
 - | -means clustering can be used. Randomly chosen subset of data points.

6.3.1 Nadaraya-Watson model

- Given $\{\mathbf{x}_n, t_n\}$, a Parzen density estimator is given by

$$p(\mathbf{x}, t) = \frac{1}{N} \sum_{n=1}^N f(\mathbf{x} - \mathbf{x}_n, t - t_n)$$

$f(\mathbf{x}, t)$: component density function

- In using linear regression model,

6.3.1 Nadaraya-Watson model

$$\begin{aligned}y(\mathbf{x}) &= \mathbb{E}[t|\mathbf{x}] = \int_{-\infty}^{\infty} tp(t|\mathbf{x}) dt \\&= \frac{\int tp(\mathbf{x}, t) dt}{\int p(\mathbf{x}, t) dt} \\&= \frac{\sum_n \int tf(\mathbf{x} - \mathbf{x}_n, t - t_n) dt}{\sum_m \int f(\mathbf{x} - \mathbf{x}_m, t - t_m) dt}\end{aligned}$$

- Assuming $\int_{-\infty}^{\infty} f(\mathbf{x}, t)t dt = 0$ zero mean

6.3.1 Nadaraya-Watson model

$$\begin{aligned}y(\mathbf{x}) &= \frac{\sum_n g(\mathbf{x} - \mathbf{x}_n) t_n}{\sum_m g(\mathbf{x} - \mathbf{x}_m)} && \text{where } g(\mathbf{x}) = \int_{-\infty}^{\infty} f(\mathbf{x}, t) dt \\&= \sum_n k(\mathbf{x}, \mathbf{x}_n) t_n && \text{where } k(\mathbf{x}, \mathbf{x}_n) = \frac{g(\mathbf{x} - \mathbf{x}_n)}{\sum_m g(\mathbf{x} - \mathbf{x}_m)}\end{aligned}$$

- It is also called *kernel regression*.

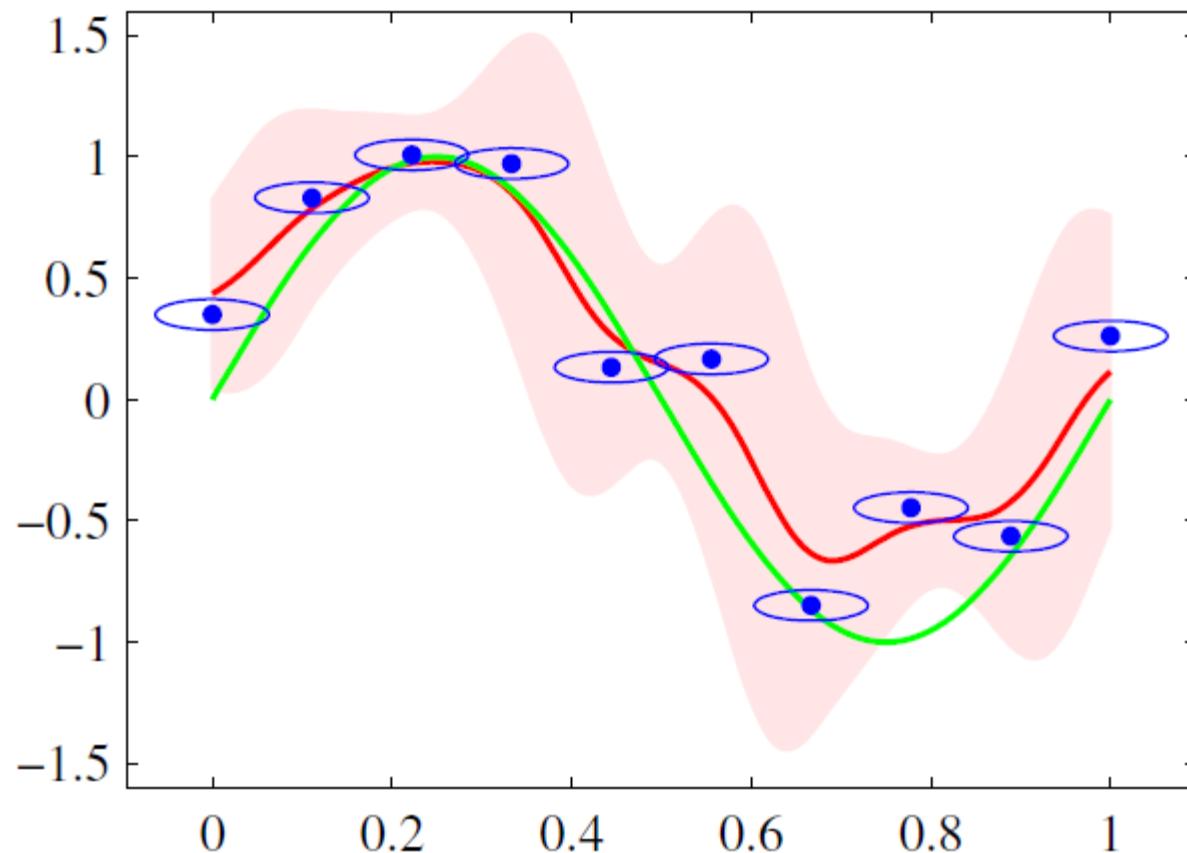
$$\sum_{n=1}^N k(\mathbf{x}, \mathbf{x}_n) = 1$$

- conditional distribution

$$p(t|\mathbf{x}) = \frac{p(t, \mathbf{x})}{\int p(t, \mathbf{x}) dt} = \frac{\sum_n f(\mathbf{x} - \mathbf{x}_n, t - t_n)}{\sum_m \int f(\mathbf{x} - \mathbf{x}_m, t - t_m) dt}$$

6.3.1 Nadaraya-Watson model

- Fig 6.3



- Sine function → data points → regression function

6.4. Gaussian Processes

- linear regression revisited

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) \quad M \text{ fixed basis functions}$$

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{0}, \alpha^{-1} \mathbf{I}) \text{ isotropic Gaussian}$$

Given $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, we have $\mathbf{y} = [y(\mathbf{x}_1), \dots, y(\mathbf{x}_N)]^T$

$$\rightarrow \mathbf{y} = \Phi \mathbf{w} \quad \Phi_{nk} = \phi_k(\mathbf{x}_n)$$

\mathbf{y} is a linear combination of \mathbf{w} $\rightarrow \mathbf{y}$ is Gaussian

$$\mathbb{E}[\mathbf{y}] = \Phi \mathbb{E}[\mathbf{w}] = \mathbf{0}$$

$$\text{cov}[\mathbf{y}] = \mathbb{E} [\mathbf{y} \mathbf{y}^T] = \Phi \mathbb{E} [\mathbf{w} \mathbf{w}^T] \Phi^T = \frac{1}{\alpha} \Phi \Phi^T = \mathbf{K}$$

$$K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m) = \frac{1}{\alpha} \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m)$$

6.4. Gaussian Processes

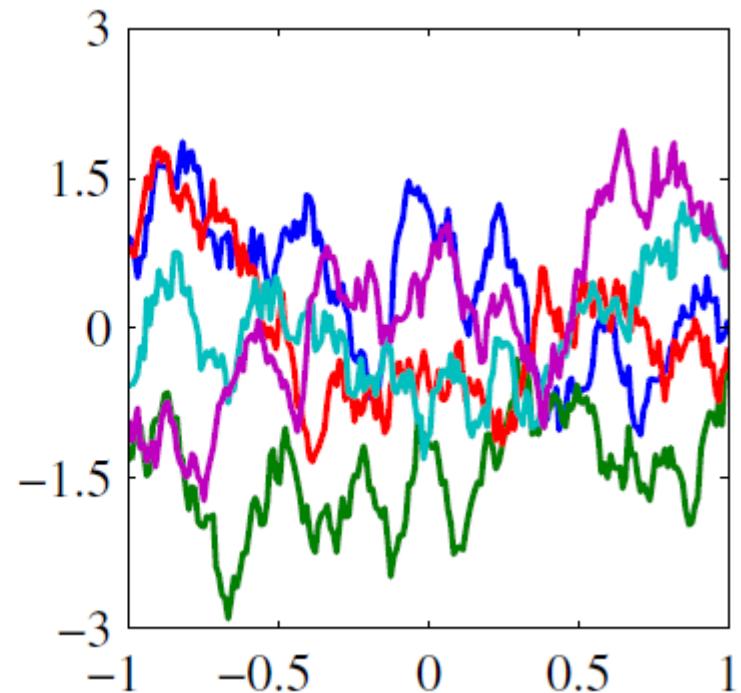
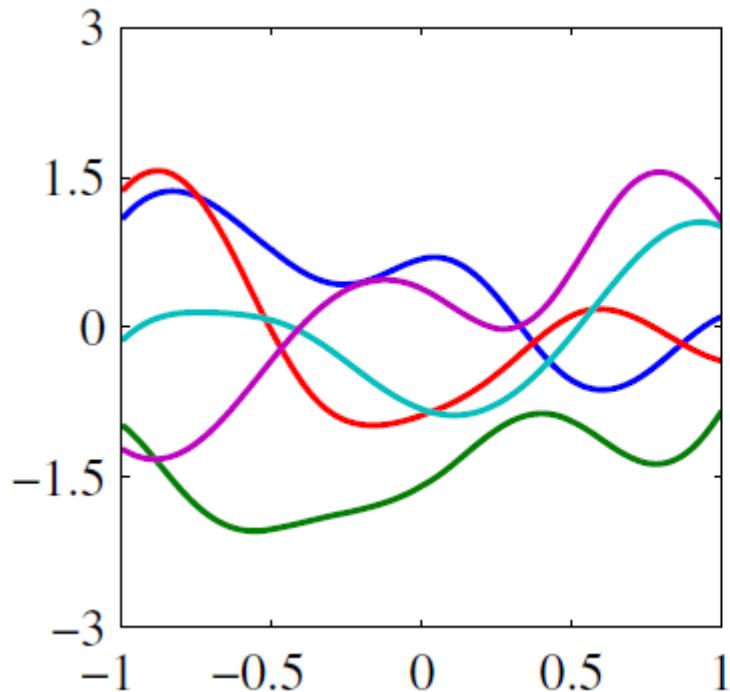
- In case of $\mathbf{x} \in R^2$, $y(\mathbf{x})$ is known as a *Gaussian random field*.

$$\mathbb{E}[y(\mathbf{x}_n)y(\mathbf{x}_m)] = k(\mathbf{x}_n, \mathbf{x}_m)$$

- Gaussian kernel : $k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2/2\sigma^2)$
- exponential kernel: $k(\mathbf{x}, \mathbf{x}') = \exp(-\theta|\mathbf{x} - \mathbf{x}'|)$

6.4. Gaussian Processes

- Fig 6.4



6.4.2 Gaussian processes for regression

$$t_n = y_n + \epsilon_n \quad y_n = y(\mathbf{x}_n)$$

$$p(t_n|y_n) = \mathcal{N}(t_n|y_n, \beta^{-1})$$

- Give $\mathbf{t} = (t_1, \dots, t_N)^T$ and $\mathbf{y} = (y_1, \dots, y_N)^T$, we have

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

- From the definition of a Gaussian process, the marginal distribution \mathbf{y} is given by a Gaussian, i.e.

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K})$$

- Also, the marginal distribution of \mathbf{t} is obtained by

$$p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{y})p(\mathbf{y}) d\mathbf{y} = \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C})$$

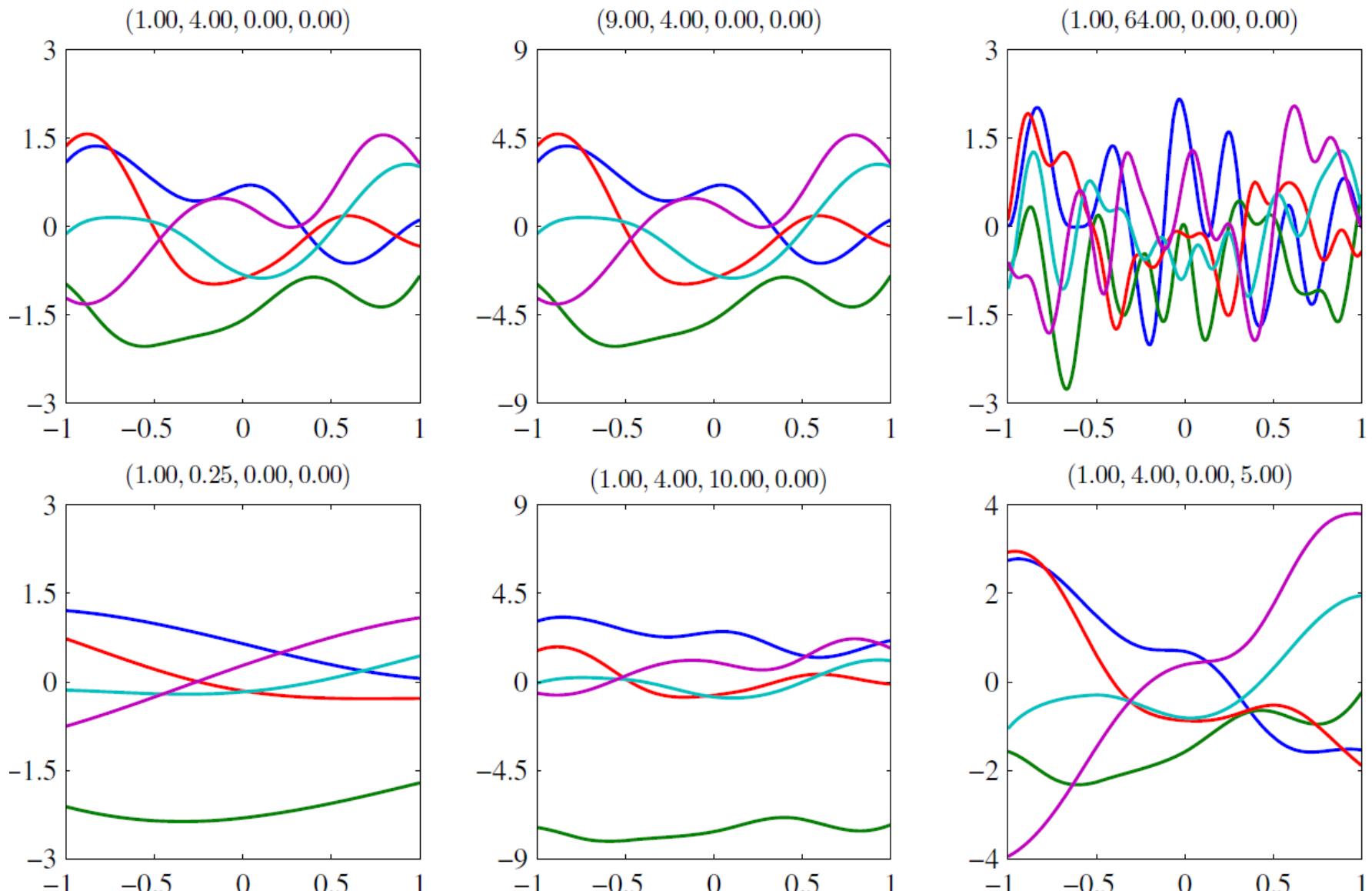
6.4.2 Gaussian processes for regression

- Where $C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \beta^{-1} \delta_{nm}$
- One widely used kernel:

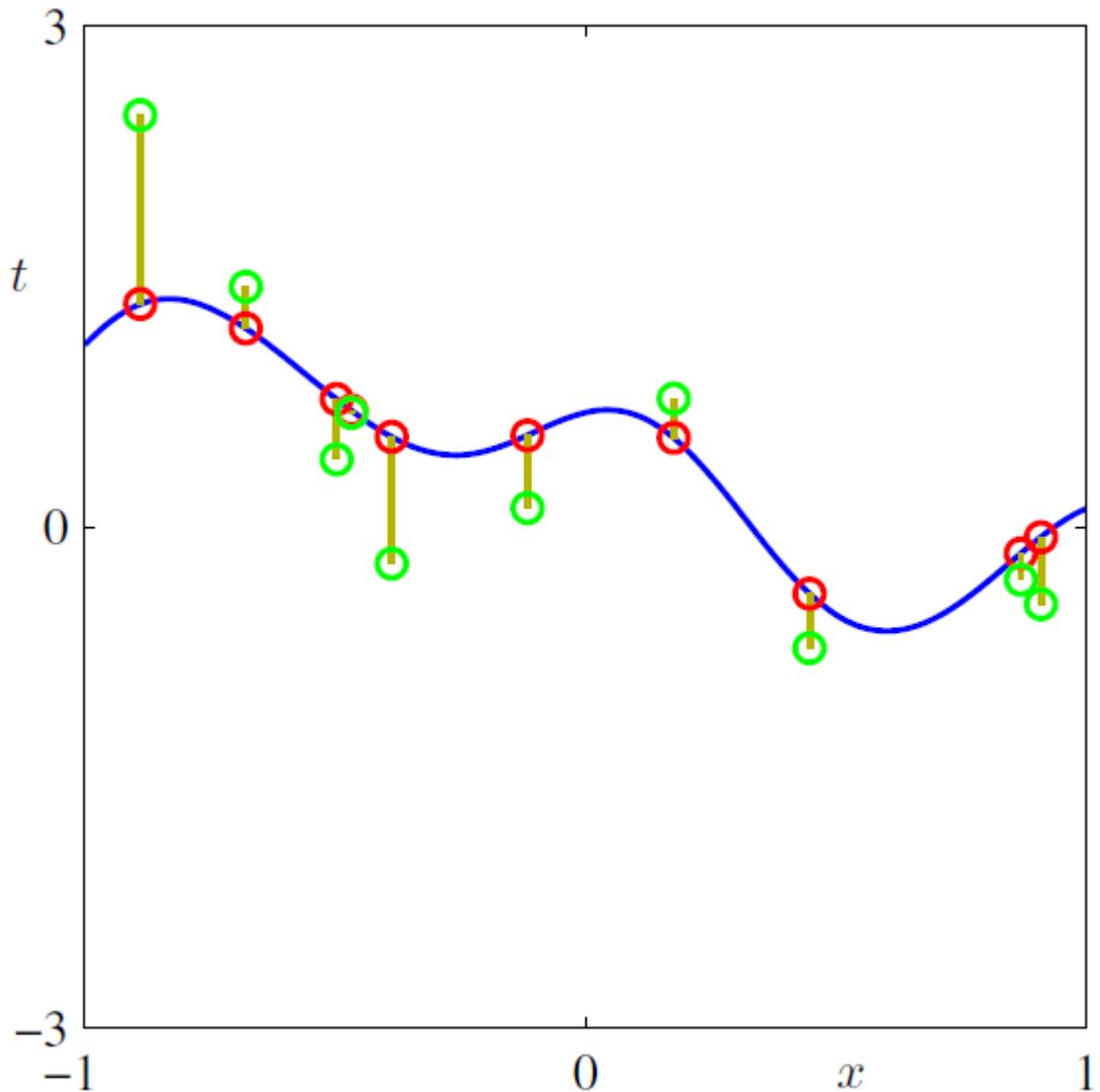
$$k(\mathbf{x}_n, \mathbf{x}_m) = \theta_0 \exp \left\{ -\frac{\theta_1}{2} \|\mathbf{x}_n - \mathbf{x}_m\|^2 \right\} + \theta_2 + \theta_3 \mathbf{x}_n^T \mathbf{x}_m$$

- See Fig 6.5 and 6.6 with different values of $(\theta_0, \dots, \theta_3)$ and $p(\mathbf{t})$
- Our goal is to make predictions, i.e. predicting t_{N+1} for input vector \mathbf{x}_{N+1} . To calculate $p(t_{N+1} | \mathbf{t}_N)$,
 $\mathbf{t} = (t_1, \dots, t_N)^T = \mathbf{t}_N$
- We compute $p(\mathbf{t}_{N+1}) = p(t_1, \dots, t_{N+1}) = \mathcal{N}(\mathbf{t}_{N+1} | \mathbf{0}, \mathbf{C}_{N+1})$

• Fig 6.5



• Fig 6.6



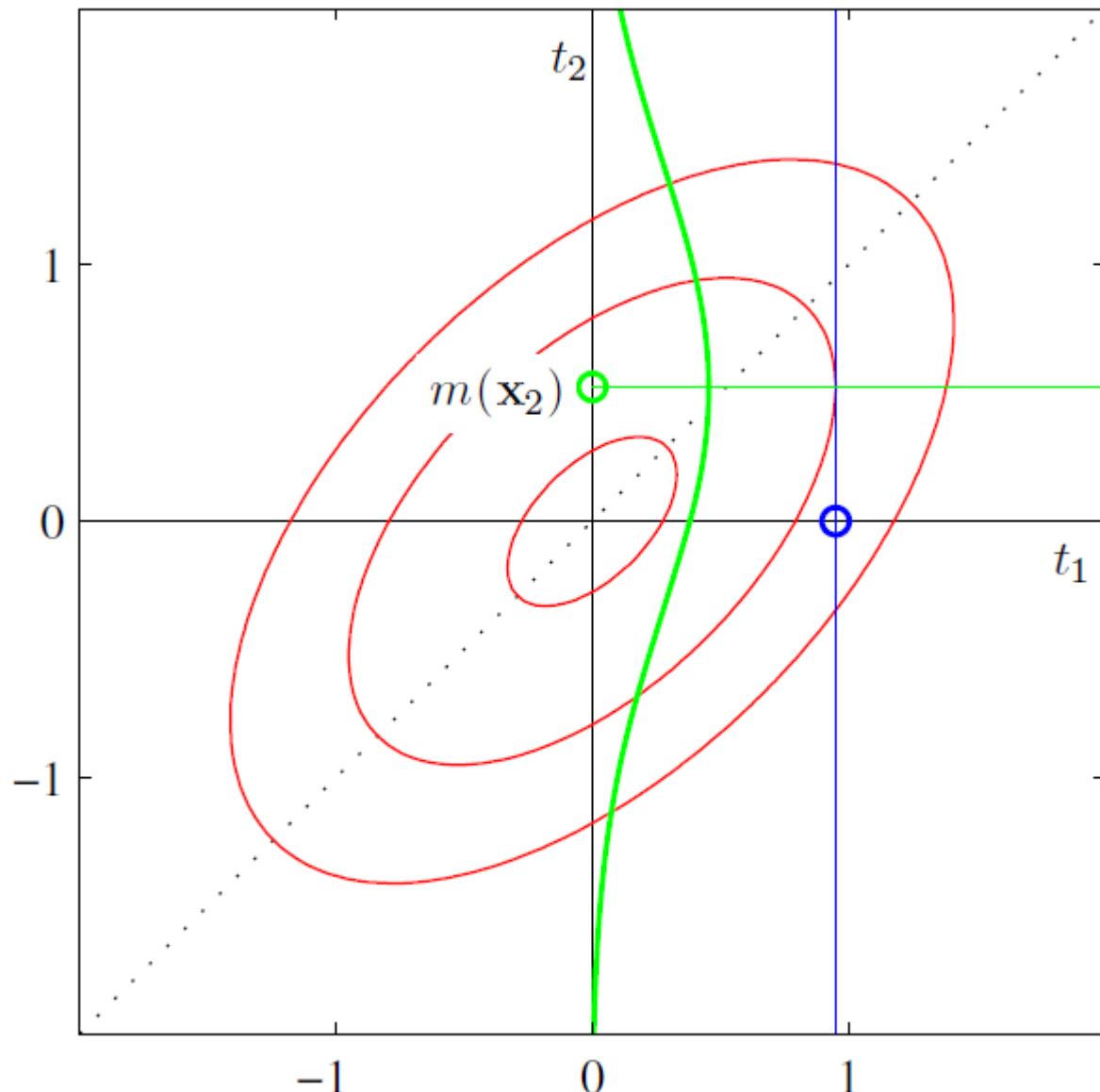
6.4.2 Gaussian processes for regression

$$\mathbf{C}_{N+1} = \begin{pmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^T & c \end{pmatrix} \quad \text{Covariance matrix}$$

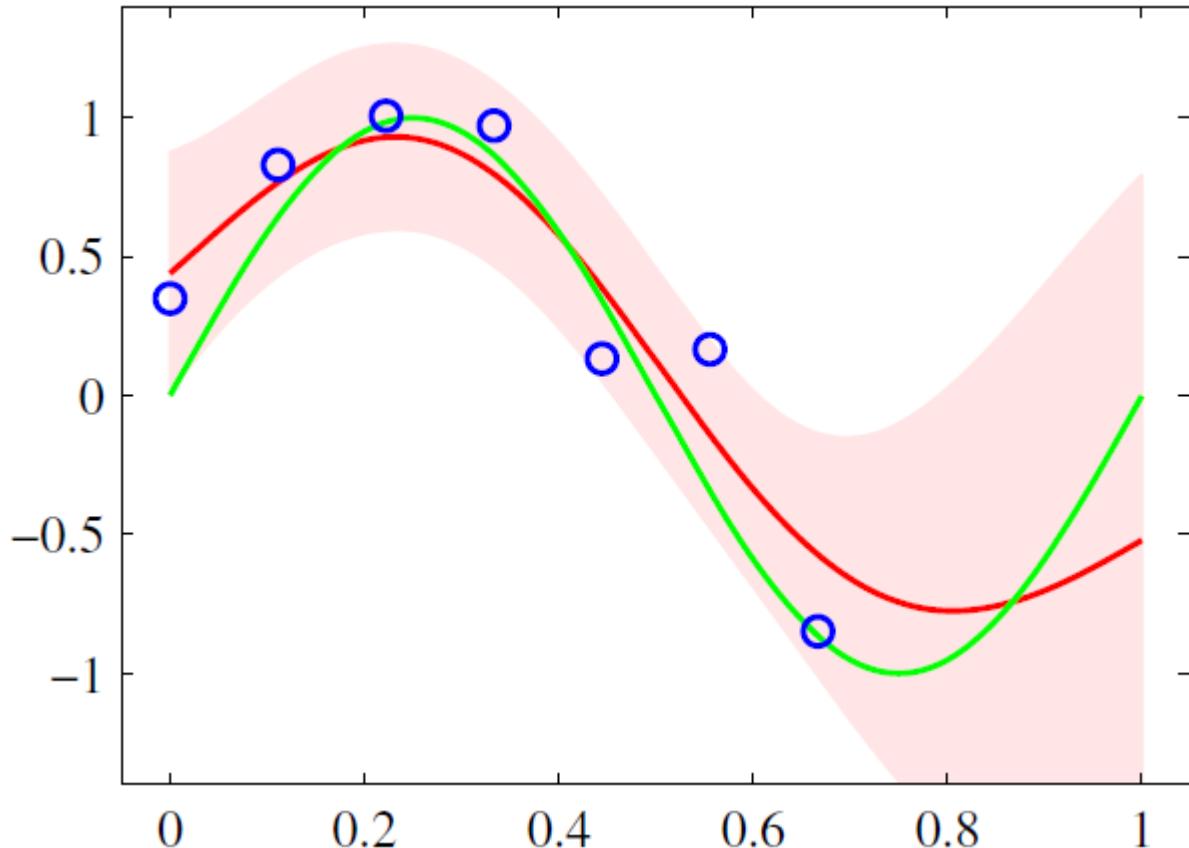
- \mathbf{k} vector has elements $k(\mathbf{x}_n, \mathbf{x}_{N+1})$ for $n = 1, \dots, N$
 $c = k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) + \beta^{-1}$
- We can determine the conditional distribution

$$p(t_{N+1} | \mathbf{t}) = \mathcal{N}(\underbrace{\mathbf{t}_{N+1}}_{\mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{t}}, \underbrace{\sigma^2(\mathbf{x}_{N+1})}_{c - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k}})$$

• Fig 6.7



- Fig 6.8



- This is key result that defines the Gaussian Process Regression

$$m(\mathbf{x}_{N+1}) = \sum_{n=1}^N a_n k(\mathbf{x}_n, \mathbf{x}_{N+1})$$

6.4.3 Learning the hyperparameters

$$\hat{\boldsymbol{\theta}} = \operatorname{argmax} \log p(\mathbf{t}|\boldsymbol{\theta})$$

- the type 2 maximum likelihood $\boldsymbol{\theta} = C_N$ covariance function.
- We don't use a fix C_N , but use a parametric model learnt from data.

$$\ln p(\mathbf{t}|\boldsymbol{\theta}) = -\frac{1}{2} \ln |\mathbf{C}_N| - \frac{1}{2} \mathbf{t}^T \mathbf{C}_N^{-1} \mathbf{t} - \frac{N}{2} \ln(2\pi)$$

$$\frac{\partial}{\partial \theta_i} \ln p(\mathbf{t}|\boldsymbol{\theta}) = -\frac{1}{2} \operatorname{Tr} \left(\mathbf{C}_N^{-1} \frac{\partial \mathbf{C}_N}{\partial \theta_i} \right) + \frac{1}{2} \mathbf{t}^T \mathbf{C}_N^{-1} \frac{\partial \mathbf{C}_N}{\partial \theta_i} \mathbf{C}_N^{-1} \mathbf{t}$$

$$\frac{\partial}{\partial \mathbf{x}} (A^{-1}) = -A^{-1} \frac{\partial A}{\partial \mathbf{x}} A^{-1}$$

$$\frac{\partial}{\partial \mathbf{x}} \ln |A| = \operatorname{Tr} \left(A^{-1} \frac{\partial A}{\partial \mathbf{x}} \right)$$

6.4.3 Learning the hyperparameters

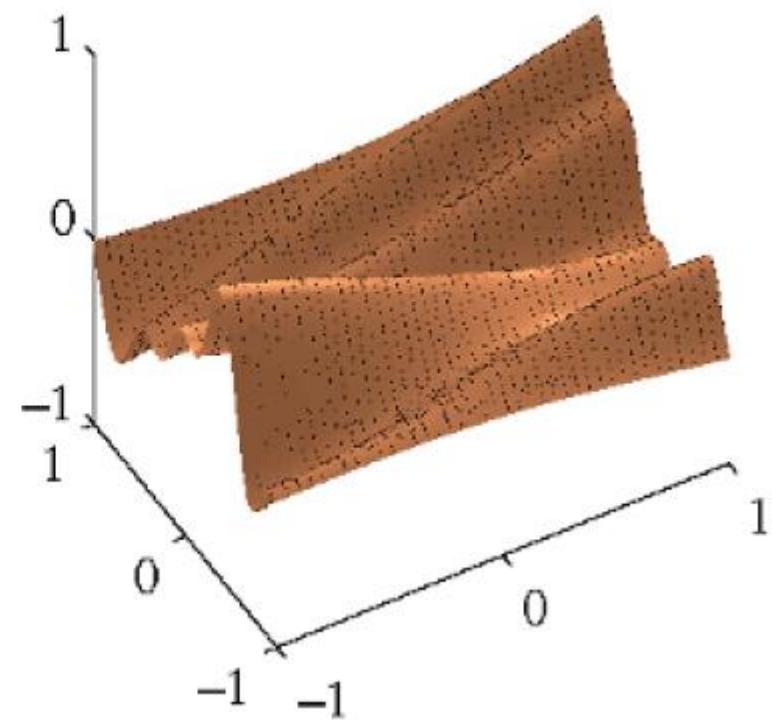
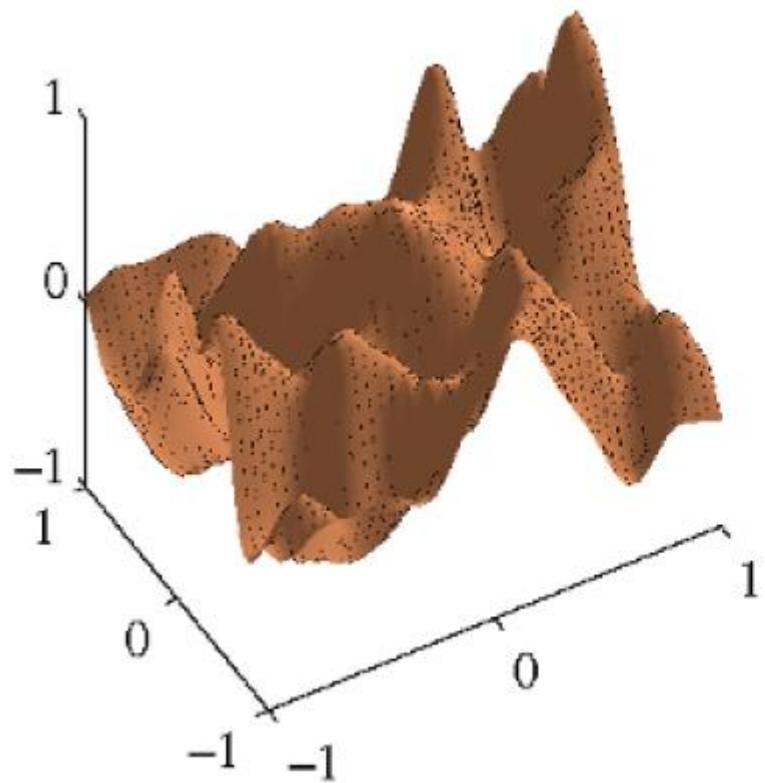
- Heteroscedastic problem: noise variance itself will also depend on it. $\beta \rightarrow \beta(\mathbf{x})$
- In this case, we use a second Gaussian process to model $\ln \beta(\mathbf{x})$

6.4.4 Automatic relevance determination

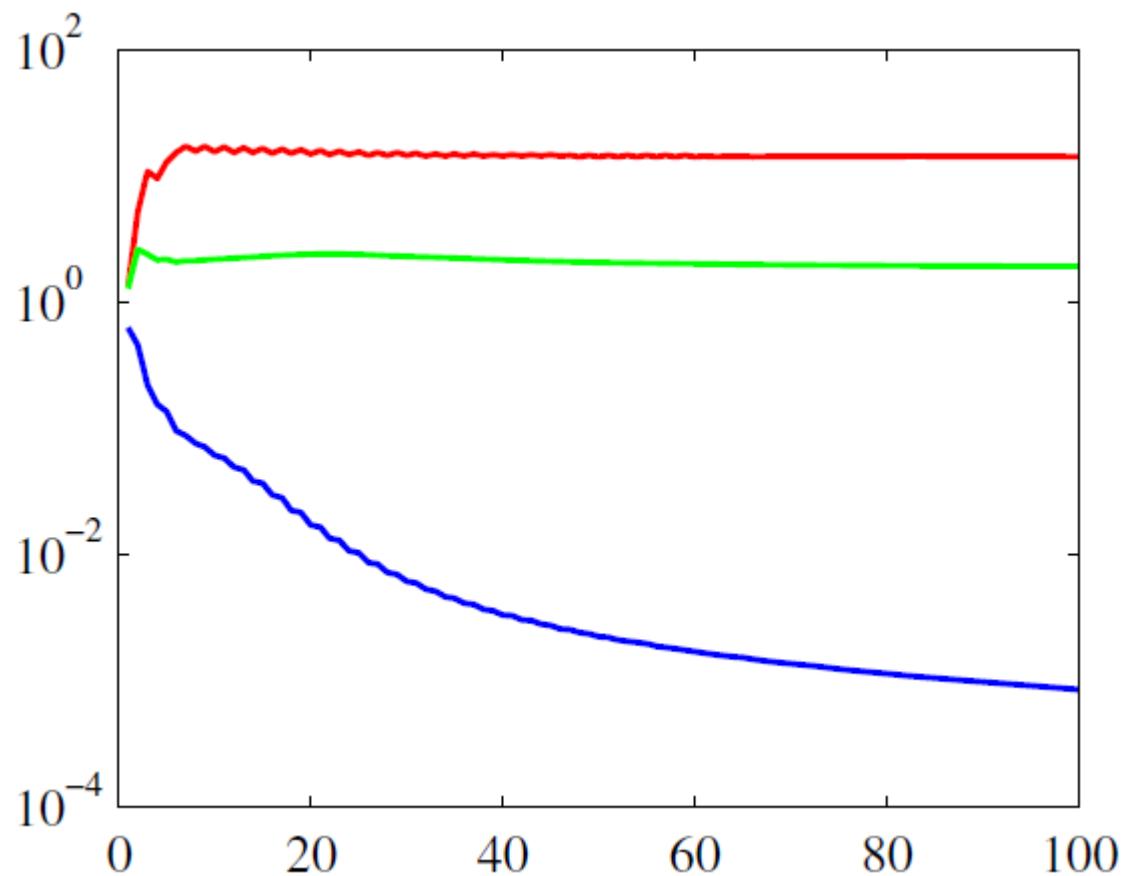
- Maximum likelihood allows the relative importance of different inputs to be inferred from the data.
→ *automatic relevance determination (ARD)*
- Consider a Gaussian process $\mathbf{x} = (x_1, x_2) \in R^2$ with kernel

$$k(\mathbf{x}, \mathbf{x}') = \theta_0 \exp \left\{ -\frac{1}{2} \sum_{i=1}^2 \eta_i (x_i - x'_i)^2 \right\}$$

• Fig 6.9



• Fig 6.10



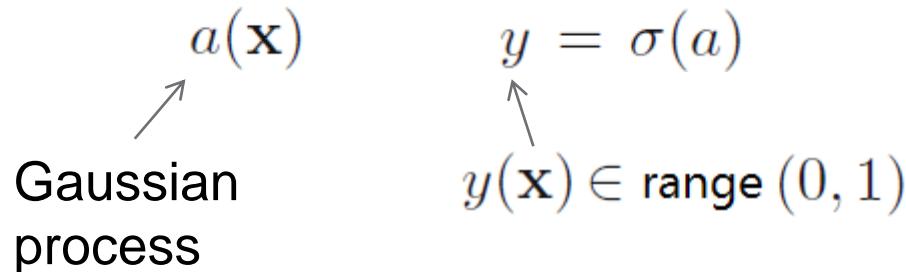
6.4.4 Automatic relevance determination

- In Fig 6.9 $\eta_i \searrow \rightarrow$ the function becomes relatively insensitive η_i is learnt by (adaption) data using maximum likelihood.
- In Fig 6.10 η_3 indicating that x_3 is irrelevant for predicting .
- ARD framework can be extended to using the Kernel shown in (6.72)

6.4.5 Gaussian processes for classification

- two-class problem $t \in \{0, 1\}$ we can adapt Gaussian processes to classification problems by transforming the output of the Gaussian process using an activation function.

6.4.5 Gaussian processes for classification

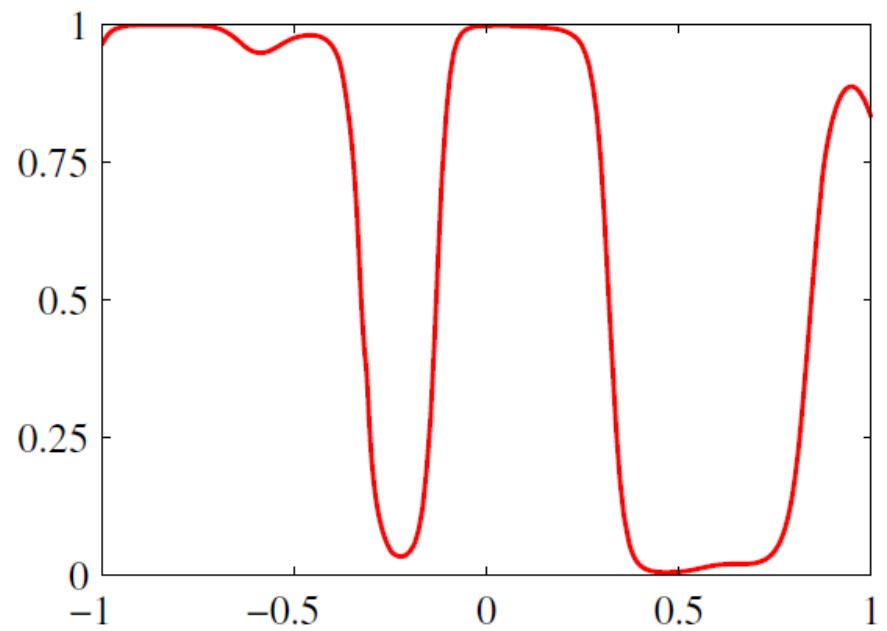
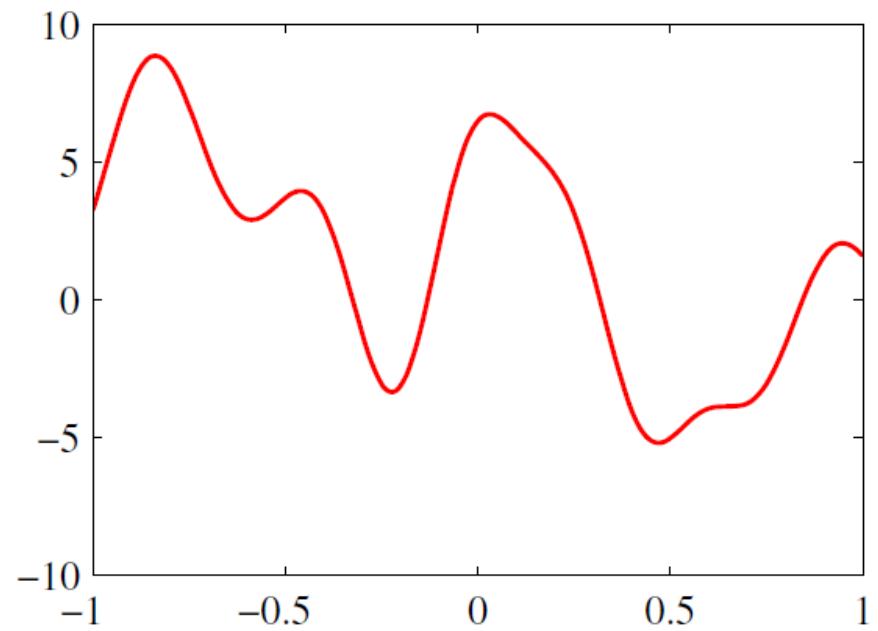


- See Fig 6.11 $p(t|a) = \sigma(a)^t(1 - \sigma(a))^{1-t}$
- Given training samples $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ and $\{t_1, \dots, t_N\} = \mathbf{t}$ and a single test point $\{\mathbf{x}_{N+1}, t_{N+1}\}$. We determine $p(t_{N+1}|\mathbf{t})$. To do so , we need to compute “Gaussian process prior”

$$p(\mathbf{a}_{N+1}) = p(a(\mathbf{x}_1), \dots, a(\mathbf{x}_{N+1})) = \mathcal{N}(\mathbf{a}_{N+1} | \mathbf{0}, \mathbf{C}_{N+1})$$

- Unlike the regression case, the covariance matrix no longer includes a noise term due to the supervised training.
- But, we introduce a noise-like term, \mathbf{C}_{N+1} has elements

• Fig 6.11



6.4.5 Gaussian processes for classification

$$C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \nu \delta_{nm}$$

- For two-class problems,

$$p(t_{N+1} = 0 | \mathbf{t}_N) = 1 - p(t_{N+1} = 1 | \mathbf{t}_N)$$

- predictive distribution

$$p(t_{N+1} = 1 | \mathbf{t}_N) = \int p(t_{N+1} = 1 | a_{N+1}) p(a_{N+1} | \mathbf{t}_N) da_{N+1}$$

- Where $p(t_{N+1} = 1 | a_{N+1}) = \sigma(a_{N+1})$

- Three approaches :
 1. *variational inference*
 2. *expectation propagation*
 3. Laplace approximation

6.4.6 Laplace approximation

$$\begin{aligned} p(a_{N+1} | \mathbf{t}_N) &= \int p(a_{N+1}, \mathbf{a}_N | \mathbf{t}_N) d\mathbf{a}_N \\ &= \frac{1}{p(\mathbf{t}_N)} \int p(a_{N+1}, \mathbf{a}_N) p(\mathbf{t}_N | a_{N+1}, \mathbf{a}_N) d\mathbf{a}_N \\ &= \frac{1}{p(\mathbf{t}_N)} \int p(a_{N+1} | \mathbf{a}_N) p(\mathbf{a}_N) p(\mathbf{t}_N | \mathbf{a}_N) d\mathbf{a}_N \\ &= \int p(a_{N+1} | \mathbf{a}_N) p(\mathbf{a}_N | \mathbf{t}_N) d\mathbf{a}_N \end{aligned}$$

- Where $p(a_{N+1} | \mathbf{a}_N) = \mathcal{N}(a_{N+1} | \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{a}_N, c - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k})$
- We need to find $p(\mathbf{a}_N | \mathbf{t}_N)$ to do that, we should calculate

$$p(\mathbf{t}_N | \mathbf{a}_N) = \prod_{n=1}^N \sigma(a_n)^{t_n} (1 - \sigma(a_n))^{1-t_n} = \prod_{n=1}^N e^{a_n t_n} \sigma(-a_n)$$

6.4.6 Laplace approximation

- Taylor expansion of

$$\begin{aligned}\ln p(\mathbf{a}_N | \mathbf{t}_N) &= \Psi(\mathbf{a}_N) = \ln p(\mathbf{a}_N) + \ln p(\mathbf{t}_N | \mathbf{a}_N) \\ &= -\frac{1}{2} \mathbf{a}_N^T \mathbf{C}_N^{-1} \mathbf{a}_N - \frac{N}{2} \ln(2\pi) - \frac{1}{2} \ln |\mathbf{C}_N| + \mathbf{t}_N^T \mathbf{a}_N \\ &\quad - \sum_{n=1}^N \ln(1 + e^{a_n}) + \text{const.}\end{aligned}$$

$$\nabla \Psi(\mathbf{a}_N) = \mathbf{t}_N - \boldsymbol{\sigma}_N - \mathbf{C}_N^{-1} \mathbf{a}_N$$

No closed-form solution

- Iterative reweighted least squares (IRLS) algorithm.
Newton-Raphson method

$$\nabla \nabla \Psi(\mathbf{a}_N) = -\mathbf{W}_N - \mathbf{C}_N^{-1}$$

6.4.6 Laplace approximation

- Hessian matrix $\mathbf{H} = -\nabla\nabla\Psi(\mathbf{a}_N) = \mathbf{W}_N + \mathbf{C}_N^{-1}$
- $p(\mathbf{a}_N | \mathbf{t}_N)$ is log convex, A single maxima/mode
- Newton-Raphson algorithm

$$\mathbf{W}^{(\text{new})} = \mathbf{W}^{(\text{old})} - \mathbf{H}^{-(1)} \nabla \mathbb{E}(\mathbf{W})$$

- Here $\mathbf{a}_N^{\text{new}} = \mathbf{C}_N(\mathbf{I} + \mathbf{W}_N \mathbf{C}_N)^{-1} \{ \mathbf{t}_N - \boldsymbol{\sigma}_N + \mathbf{W}_N \mathbf{a}_N \}$
- iterated until converge to the mode \mathbf{a}_N^*
- Where $\mathbf{a}_N^* = \mathbf{C}_N(\mathbf{t}_N - \boldsymbol{\sigma}_N)$ ($\because \nabla\Psi(\mathbf{a}_N) = 0$)
- Gaussian approximation to the posterior distribution $p(\mathbf{a}_N | \mathbf{t}_N)$
$$q(\mathbf{a}_N) = \mathcal{N}(\mathbf{a}_N | \mathbf{a}_N^*, \mathbf{H}^{-1})$$

6.4.6 Laplace approximation

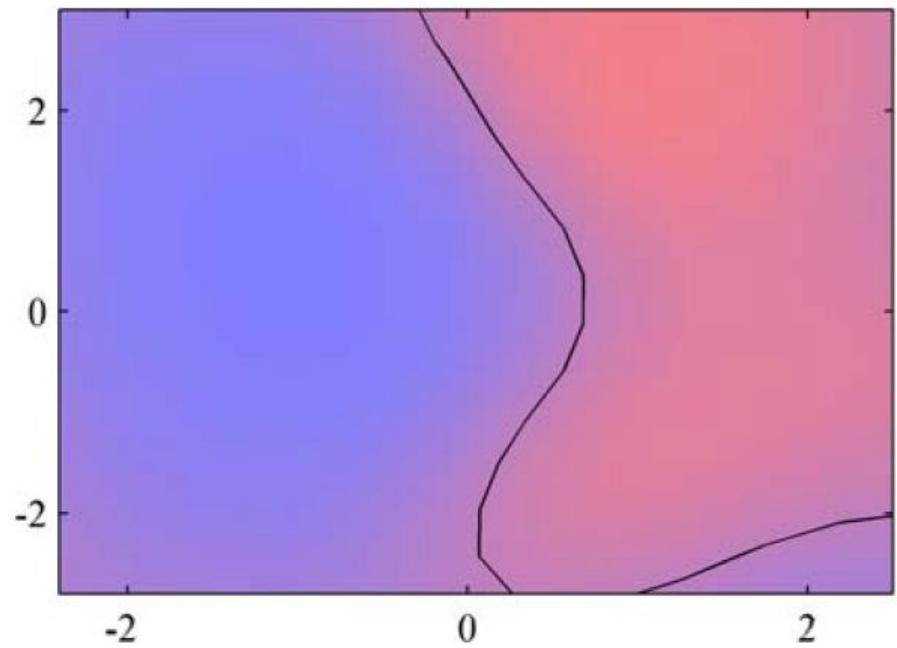
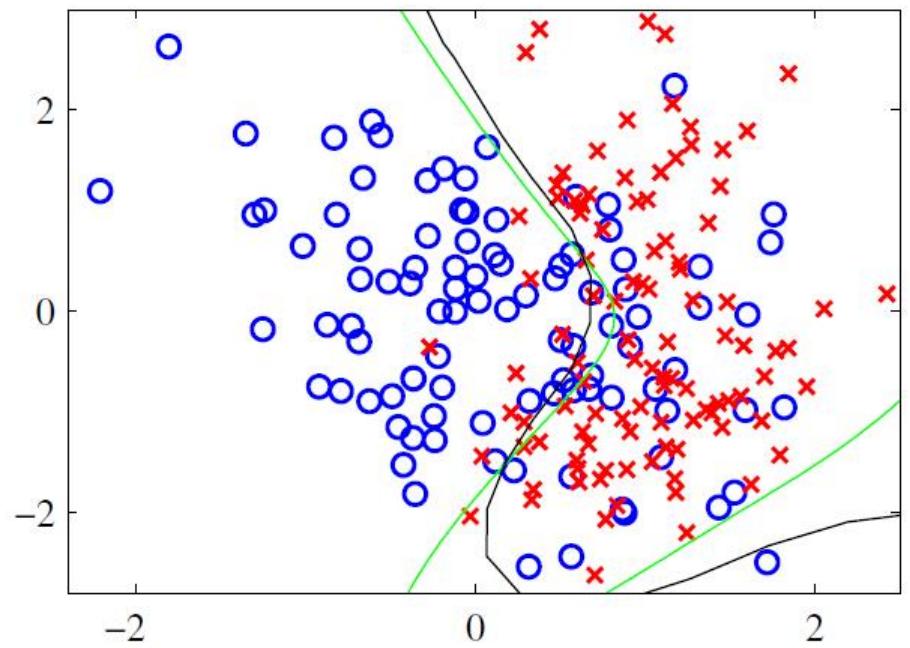
- Hence $q(a_N | \mathbf{t}_N) \cong p(a_{N+1} | a_N) q(a_N) da_N$
 $= \mathcal{N}(a_{N+1} | \mathbf{k}^T(\mathbf{t}_N - \boldsymbol{\sigma}_N), c - \mathbf{k}^T(\mathbf{W}_N^{-1} + \mathbf{C}_N)^{-1}\mathbf{k})$

Then

$$\begin{aligned} p(t_{N+1}=1 | \mathbf{t}_N) &= \int p(\mathbf{t}_{N+1}=1 | \mathbf{a}_{N+1}) p(\mathbf{a}_{N+1} | \mathbf{t}_N) d\mathbf{a}_{N+1} \\ &\quad \int \sigma(a) \mathcal{N}(a | \mu, \sigma^2) da \simeq \sigma(\kappa(\sigma^2)\mu) \\ \kappa(\sigma^2) &= (1 + \pi\sigma^2/8)^{-1/2} \end{aligned}$$

We also need to determine the parameters θ of the covariance Kernel function.

- Fig 6.12



6.4.7 Connection to neural networks

- Sufficiently large $M \rightarrow$ approximate any function with arbitrary accuracy.

$f(\mathbf{x}, \mathbf{w})$ $p(\mathbf{w})$ Bayesian neural network

$y(\mathbf{x})$ \mathbf{y} network output

- the distribution of functions generated by a neural network will tend to a Gaussian process if $M \rightarrow \infty$

CONTENTS

1. Introduction
2. Probability Distributions
3. Linear Models for Regression
4. Linear Models for Classification
5. Kernel Methods
6. Sparse Kernel Methods
7. Mixture Models and EM
8. Approximate Inference

Sparse Kernel Machines

- $k(\mathbf{x}_n, \mathbf{x}_m)$ for all samples is infeasible sparse solutions, so that predictions for new inputs depend only on the kernel function evaluated at a subset of the training data points.
- *Support vector machine (SVM)* → classification, regression
- SVM: determination of the model parameters corresponds to a “convex optimization” problem.
- Lagrange multipliers are extensively used.
Vapnik (1995) RVM *Relevance vector machine* → much sparser solution

Maximum Margin Classifiers

- two-class classification problem

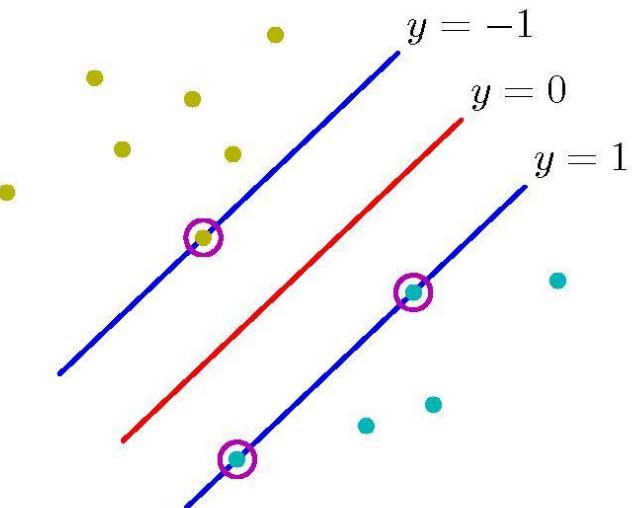
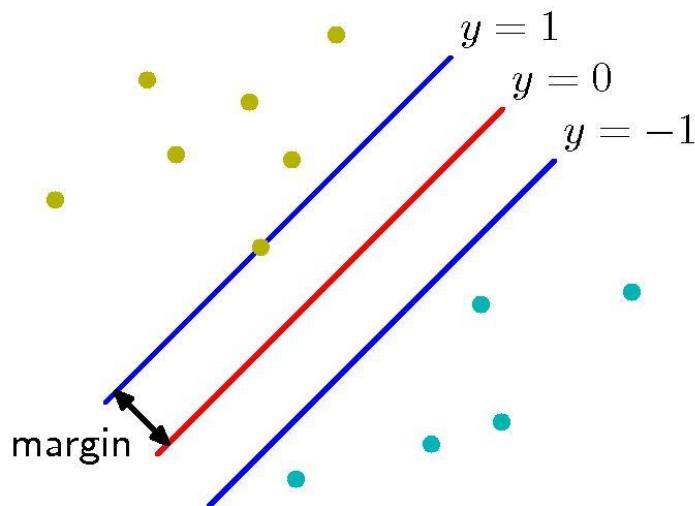
$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$$

$$\{\mathbf{x}_1, \dots, \mathbf{x}_N\} \quad \{t_1, \dots, t_N\} \quad t_n \in \{-1, 1\}$$

$$\begin{aligned} y(\mathbf{x}_n) &> 0 \text{ for points having } t_n = +1 \\ y(\mathbf{x}_n) &< 0 \text{ for points having } t_n = -1 \end{aligned} \quad \left. \begin{array}{l} t_n y(\mathbf{x}_n) > 0 \\ \text{for all samples} \end{array} \right\}$$

Maximum Margin Classifiers

- Fig 7.1



- location of this Boundary is determined by a subset of the data points.
⇒ support vectors

Maximum Margin Classifiers

- Perceptron algorithm depends on the initial values of w and b and the order in which the data points are presented.
- In SVM, decision boundary is chosen to be the one for which the margin is maximized.

→ *computational learning theory, statistical learning theory.*

- See text
$$\mathbf{x} = \mathbf{x}_\perp + \gamma \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

$$\mathbf{w}^T \mathbf{x} + w_0 = y(\mathbf{x}) = \mathbf{w}^T \mathbf{x}_\perp + w_0 + \gamma \|\mathbf{w}\|$$

$$\Rightarrow \gamma = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}$$

We are only interested in data points are correctly classified.

Maximum Margin Classifiers

- Distance of \mathbf{x}_n to the decision surface

$$\frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} = \frac{t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b)}{\|\mathbf{w}\|}$$

- Maximum margin solution is found by

$$(\hat{\mathbf{w}}, \hat{b}) = \arg \max_{\mathbf{w}, b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n [t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b)] \right\}$$

brace indicating the term being minimized

perpendicular distance to the closest point \mathbf{x}_n

- Direct solution would be very complex → convert it to an equivalent problem

Maximum Margin Classifiers

- Rescaling $\mathbf{w} \rightarrow \kappa \mathbf{w}$, $b \rightarrow \kappa b$ $t_n y(\mathbf{x}_n) / \|\mathbf{w}\|$ distance is unchanged.
- We can set $t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) = 1$ for the point \mathbf{x}_n that is closest to the surface. In this case, all data points will satisfy the constraints

$$t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1, \quad n = 1, \dots, N$$

- Those points satisfying “equality” are said “active” “support vectors”. The remaining point are inactive.
- There are at least two active constraints.
- Optimization problem is changed to

Maximum Margin Classifiers

$$(\hat{\mathbf{w}}, \hat{b}) = \arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \leftarrow \arg \max_{\mathbf{w}, b} \|\mathbf{w}\|^{-1}$$

- Quadratic programming : Minimize a quadratic function subject to a set of linear inequality constraints.
- Parameter “ $\|\cdot\|$ ” is implicitly determined by the constraints.
- Constrained optimization problem
“Lagrange” See text about Lagrange
- Lagrange function :

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N a_n \{ t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) - 1 \}$$

Maximum Margin Classifiers

$$\frac{\partial L(\mathbf{w}, b, \mathbf{a})}{\partial \mathbf{w}} = 0 \quad \mathbf{w} = \sum_{n=1}^N a_n t_n \phi(\mathbf{x}_n)$$

$$\frac{\partial L(\mathbf{w}, b, \mathbf{a})}{\partial b} = 0 \quad 0 = \sum_{n=1}^N a_n t_n$$

- Dual representation of the maximum margin problem

$$\max_a \left\{ \tilde{L}(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \right\}$$

- Where kernel function $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$ is applied.

7.1. Maximum Margin Classifiers

- M variables → computational complexity $O(M^3)$
In case of $M < N$ dual problem appears disadvantageous.
- By using kernels, we can make $M > N$ $k(\mathbf{x}, \mathbf{x}')$ positive definite ⇒ bounded $\tilde{L}(\mathbf{a})$
- In classification problem, we evaluate the sign of $y(\mathbf{x})$. rearranged by substituting

$$\mathbf{w} = \sum_{n=1}^N a_n t_n \phi(\mathbf{x}_n)$$

$$y(\mathbf{x}) = \sum_{n=1}^N a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b$$

Maximum Margin Classifiers

- See Appendix E constrained optimization of this form satisfies the *Karush-Kuhn-Tucker* (KKT) conditions (1939,1951)

$$a_n \geq 0$$

$$t_n y(\mathbf{x}_n) - 1 \geq 0$$

$$a_n \{t_n y(\mathbf{x}_n) - 1\} = 0$$

Maximum Margin Classifiers

- For every data point, either $a_n = 0$ or $t_n y(\mathbf{x}_n) = 1$
Only support vectors are retained.

Once $\mathbf{a} = (a_1, \dots, a_N)^T$ is found, parameter b is found by
 $t_n y(\mathbf{x}_n) = 1$ for any support vector \mathbf{x}_n

$$\begin{aligned} \text{i.e. } \quad & t_n \left(\sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) + b \right) = 1 \\ \Rightarrow \quad & b = \frac{1}{N_S} \sum_{n \in \mathcal{S}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right) \end{aligned}$$

- Express the maximum margin classifier by means of minimization of an error function.

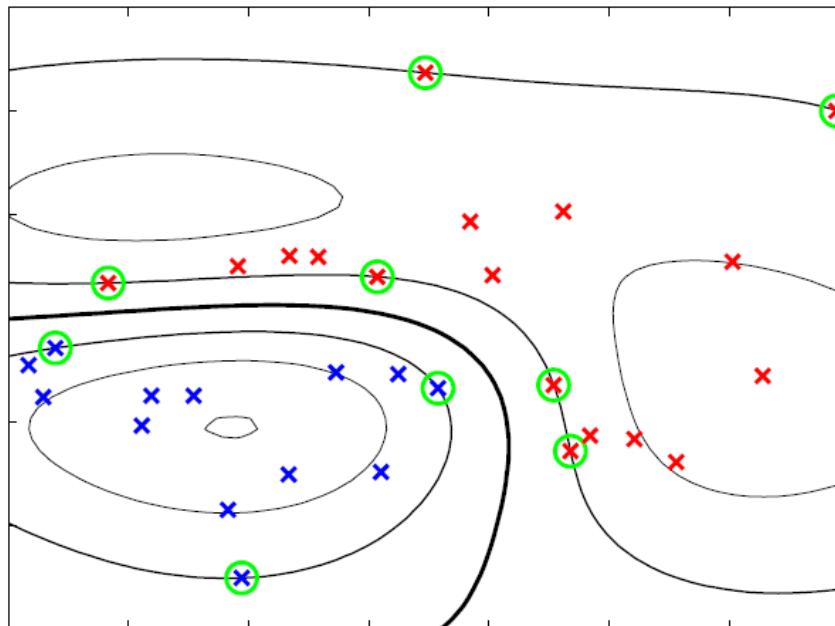
Maximum Margin Classifiers

$$\sum_{n=1}^N E_\infty(y(\mathbf{x}_n)t_n - 1) + \lambda \|\mathbf{w}\|^2$$

$$E_\infty(z) = 0 \quad \text{if } z \geq 0$$

$$E_\infty(z) = \infty \quad \text{if } z < 0$$

- Fig 7.2



Maximum Margin Classifiers

- Support vector machine by using a Gaussian kernel
nonlinearly separable in Z-D space
→ linearly separable in nonlinear feature space

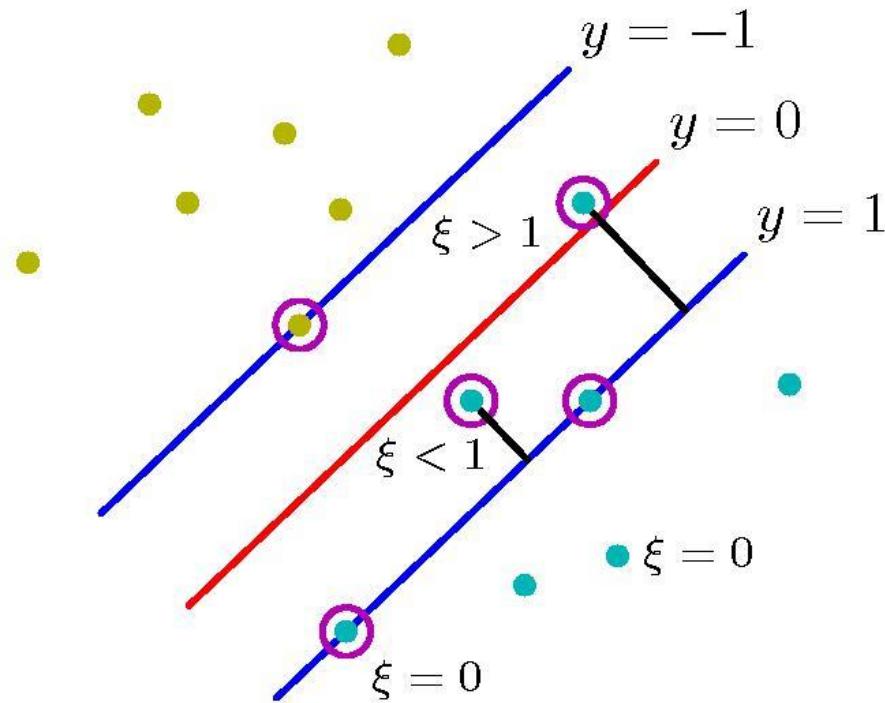
Overlapping class distributions

- We need to modify the support vector machine so as to allow some of the training points to be misclassified, with the penalty which is increases by the distance from that boundary.
- Slack variables are introduced.

$$\xi_n \geq 0 \quad n = 1, \dots, N$$

Overlapping class distributions

- Fig 7.3



Overlapping class distributions

- $\xi_n = 0$ data is on or inside the correct margin boundary
 $\xi_n = |t_n - y(\mathbf{x}_n)|$ for other points.
- $\xi_n = 1$ when data point is on the decision boundary $y(\mathbf{x}_n) = 0$
- $\xi_n > 1$ when data is misclassified.
- Classification constraints are replaced by

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n \quad n = 1, \dots, N$$

- Hard margin constraint is relaxed to be soft margin allowing misclassified samples.
- This framework is still sensitive to outliers
- Our goal is to maximize the margin while softly penalizing points that lie on the wrong side. Therefore

Overlapping class distributions

$$\min_{\mathbf{w}} \left(C \sum_{n=1}^N \xi_n + \frac{1}{2} \|\mathbf{w}\|^2 \right)$$

$C \rightarrow \infty$ support vector machine for separable data.

- $\sum_n \xi_n$ “upper bound” on the number of misclassified points.

$$\min_{\mathbf{w}} \left\{ C \sum_n \xi_n + \frac{1}{2} \|\mathbf{w}\|^2 \right\}$$

Subject to $t_n y(\mathbf{x}_n) \geq 1 - \xi_n$ and $\xi_n \geq 0$

- Lagrangian is written by

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n - \sum_{n=1}^N a_n \{t_n y(\mathbf{x}_n) - 1 + \xi_n\} - \sum_{n=1}^N \mu_n \xi_n$$

Overlapping class distributions

- KKT conditions are given by

$$a_n \geq 0$$

$$t_n y(\mathbf{x}_n) - 1 + \xi_n \geq 0$$

$$a_n (t_n y(\mathbf{x}_n) - 1 + \xi_n) = 0$$

$$\mu_n \geq 0$$

$$\xi_n \geq 0$$

$$\mu_n \xi_n = 0$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^N a_n t_n \phi(\mathbf{x}_n)$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^N a_n t_n = 0$$

Overlapping class distributions

$$\frac{\partial L}{\partial \xi_n} = 0 \Rightarrow a_n = C - \mu_n$$

$$a_n \geq 0 \quad \mu_n \geq 0 \Rightarrow a_n \leq C$$

- We obtain dual Lagrange in the form

$$\min_{\{a_n\}} \left\{ \tilde{L}(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \right\}$$

Subject to $0 < a_n < C \leftarrow$ box constraints

$$\sum_{n=1}^N a_n t_n = 0$$

- Solution interpretation $a_n = 0$ inactive

$$a_n > 0 \quad t_n y(\mathbf{x}_n) = 1 - \xi_n \quad \text{from (7.25)}$$

Overlapping class distributions

- If $a_n < C$ then $\mu_n > 0$ ($\because a_n = C - \mu_n$) then $\xi_n = 0$
this point lies on the margin.
- If $a_n = C$ then $\mu_n = 0$, $\xi_n \leq 1$ or $\xi_n > 1$
- To determine b , support vectors a_n satisfy

$0 < a_n < C$, $\xi_n = 0$, $t_n y(\mathbf{x}_n) = 1$ Then, we have

$$t_n \left(\sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) + b \right) = 1$$

$$\Rightarrow b = \frac{1}{N_{\mathcal{M}}} \sum_{n \in \mathcal{M}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right)$$

\mathcal{M} is a set with data points having $0 < a_n < C$ support vectors

Overlapping class distributions

- Alternative SVM called ν -SVM . This involves maximizing

$$\tilde{L}(\mathbf{a}) = -\frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

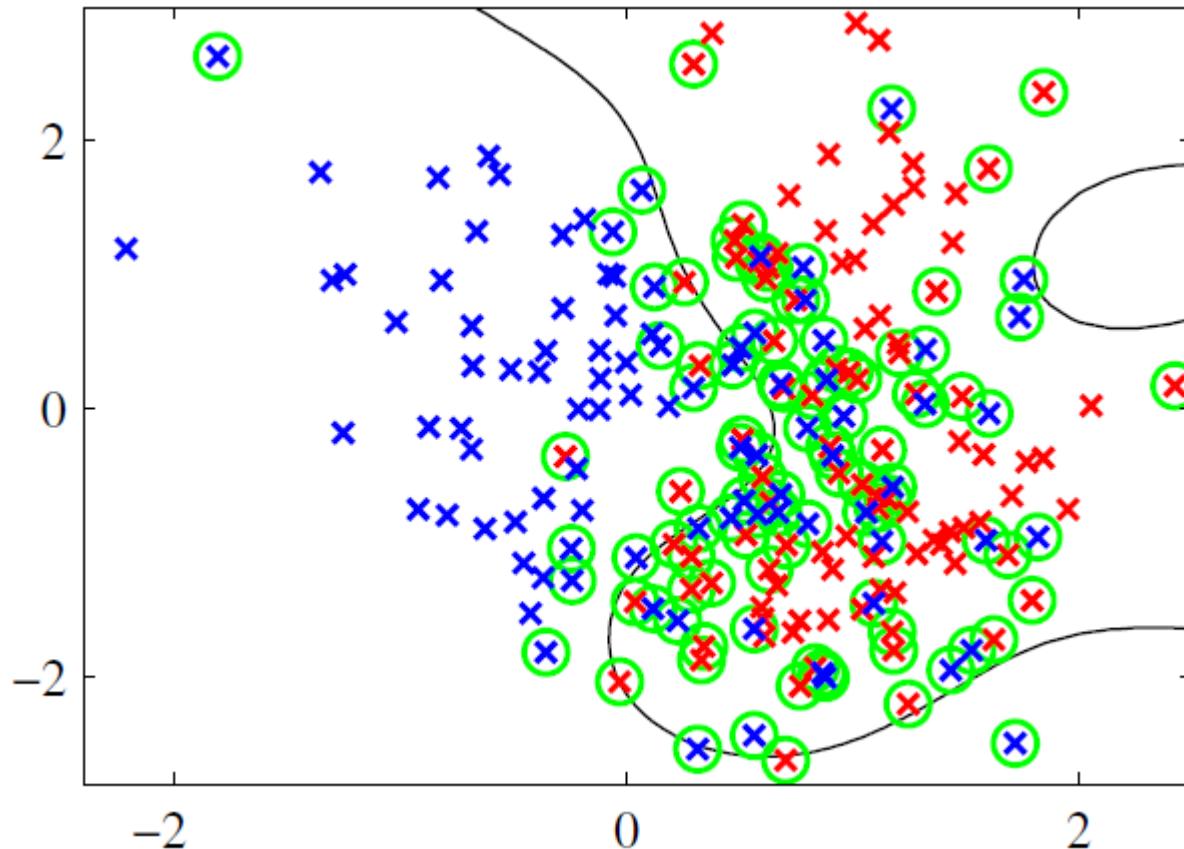
subject to $0 \leq a_n \leq 1/N$

$$\sum_{n=1}^N a_n t_n = 0 \quad \sum_{n=1}^N a_n \geq \nu$$

- ν replaces C , ν interpreted as upper bound on the fraction of *margin errors* and lower bound on the fraction of support vectors.

Overlapping class distributions

- Fig 7.4



Overlapping class distributions

- Although predictions for new inputs are made using only the support vectors, the training phase of estimation makes use of the whole data set. Efficient algorithms for solving quadratic programming problem is important.
- Direct solution is infeasible. Some method were proposed.
 1. Chunking(1982) See Text full quadratic programming → a series of smaller ones.
 2. protected conjugate gradients (1998)
 3. *Decomposition methods* (Osuna *et al.*, 1996)
 4. *sequential minimal optimization (SMO)* (Platt, 1999)
- SVM somehow manages to avoid the curse of dimensionality sue to the use of “Kernel function”.

Overlapping class distributions

- This is not the case, because there are constraints at feature values that restrict the effective dimensionality of feature space.

$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) &= (1 + \mathbf{x}^T \mathbf{z})^2 = (1 + x_1 z_1 + x_2 z_2)^2 \\ &= 1 + 2x_1 z_1 + 2x_2 z_2 + x_1^2 z_1^2 + 2x_1 x_2 z_1 z_2 + x_2^2 z_2^2 \\ &= (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1 x_2, x_2^2)(1, \sqrt{2}z_1, \sqrt{2}z_2, z_1^2, \sqrt{2}z_1 z_2, z_2^2)^T \\ &= \phi(\mathbf{x})^T \phi(\mathbf{z}) \end{aligned}$$

∂ -dim space \mathbf{x} would be constrained to lie on a z -dim nonlinear manifold embedded in the 6-dim feature space.

Overlapping class distributions

- SVM does not provide probabilistic outputs. If we wish to use SVM in a larger probabilistic system. Probabilistic predictions of class label t for new patterns are required.
- To address this issue, we determine the conditional probability by $p(t = 1|\mathbf{x}) = \sigma(Ay(\mathbf{x}) + B)$

A, B are trained by minimizing the cross-entropy error function of training pairs of $\{y(\mathbf{x}_n), t_n\}$ Training of $\{A, B\}$ and SVM are assumed independent \rightarrow poor approximation to posterior probabilities.

Relation to logistic regression

- Again, for data points lying on the correct side of margin boundary. $y_n t_n \geq 1$, $\xi_n = 0$. Otherwise, the remaining points $\xi_n = 1 - y_n t_n$. Then, objective function

$$C \sum_{n=1}^N \xi_n + \frac{1}{2} \|\mathbf{w}\|^2 \rightarrow \sum_{n=1}^N E_{\text{SV}}(y_n t_n) + \lambda \|\mathbf{w}\|^2$$

Where $\lambda = (2C)^{-1}$ and $E_{\text{SV}}(y_n t_n) = [1 - y_n t_n]_+$

$t \in \{0, 1\} \rightarrow \{-1, 1\}$ Considering the logistic regression model with target $t \in \{-1, 1\}$, $p(t = 1|y) = \sigma(y)$ and

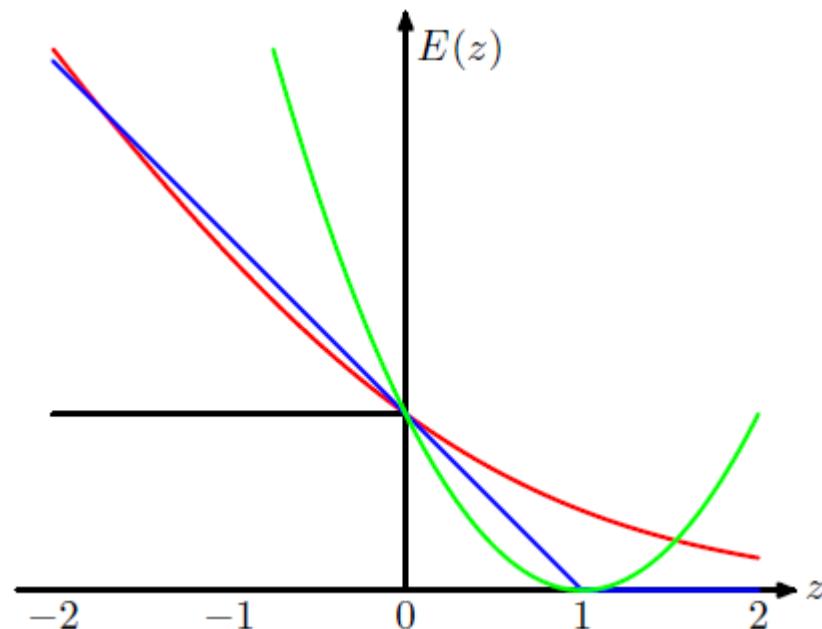
$$p(t = -1|y) = 1 - \sigma(y) = \sigma(-y) \Rightarrow p(t|y) = \sigma(yt)$$

- We construct an error function

$$\sum_{n=1}^N E_{\text{LR}}(y_n t_n) + \lambda \|\mathbf{w}\|^2$$

Relation to logistic regression

- Fig 7.5



Relation to logistic regression

- Where $E_{LR}(yt) = \ln(1 + \exp(-yt))$
- Rescaled logistic regression error function has similar form to SVM error function
- The flat region in $E_{SV}(yt)$ leads to “sparse solution”

7.1.3 Multiclass SVMs

- $K > 2$ classes Various methods have been proposed for combining multiple two-class SVMs to build a multiclass classifier.
 - I. Construct K separate SVMs where the k^{th} model $y_k(\mathbf{x})$ is trained using data from \mathcal{C}_k (positive) and the data from the remaining $K - 1$ classes .

Multiclass SVMs

→ one-versus-the-rest method making predictions by

$$y(\mathbf{x}) = \max_k y_k(\mathbf{x})$$

- Different classifiers were trained on different tasks .
- No guarantee $y_k(\mathbf{x})$ for different classifiers → appropriate scales.
- Also, training sets are imbalanced. The symmetry of the original problem is lost → positive class $t = 1$
negative class $t = -\frac{1}{k-1}$

- Also, the simultaneous training of K SVMs can be performed but training time increased a lot.

Multiclass SVMs

II. Construct $C_2^k = k(k - 1)/2$ SVMs on all pairs of classes classify test points by finding the class with highest number of “vote” → one-versus-one-lead to “ambiguities” in the result.

More training time is required.

More computation time for testing is required.

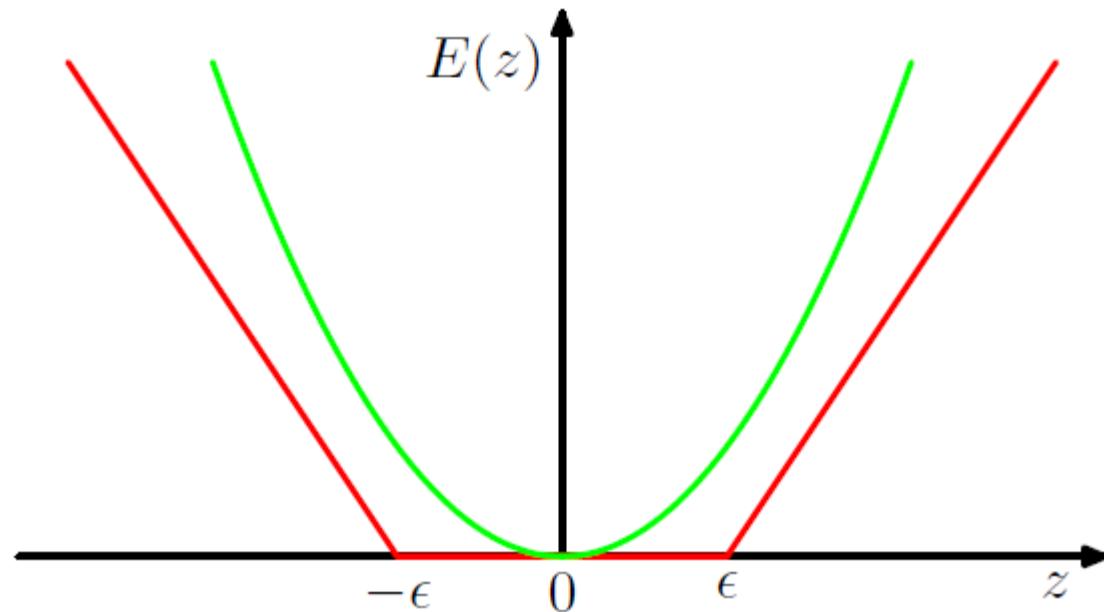
- Testing time can be reduced by organizing the pairwise classifiers into a directed acyclic graph . Only $K - 1$ pairwise classifiers need to be classified.
- It can be also viewed as a decoding problem.
- Single-class SVM → unsupervised learning → probability density estimation
find a smooth boundary enclosing a region of high density

SVMs for regression

- *SVM* for Regression (Support Vector Regression)
- In a simple linear regression, we minimize a regularized error function

$$\frac{1}{2} \sum_{n=1}^N \{y_n - t_n\}^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

- Fig 7.6



SVMs for regression

- To obtain sparse solutions, the ϵ -insensitive error function is applied.

$$E_\epsilon(y(\mathbf{x}) - t) = \begin{cases} 0, & \text{if } |y(\mathbf{x}) - t| < \epsilon; \\ |y(\mathbf{x}) - t| - \epsilon, & \text{otherwise} \end{cases}$$

- A new regularized error function is established by

$$C \sum_{n=1}^N E_\epsilon(y(\mathbf{x}_n) - t_n) + \frac{1}{2} \|\mathbf{w}\|^2$$

- By introducing two slack variables, the optimization problem can be re-expressed. $\xi_n \geq 0$, $\hat{\xi}_n \geq 0$

$\xi_n > 0$ corresponds to a point $t_n > y(\mathbf{x}_n) + \epsilon$

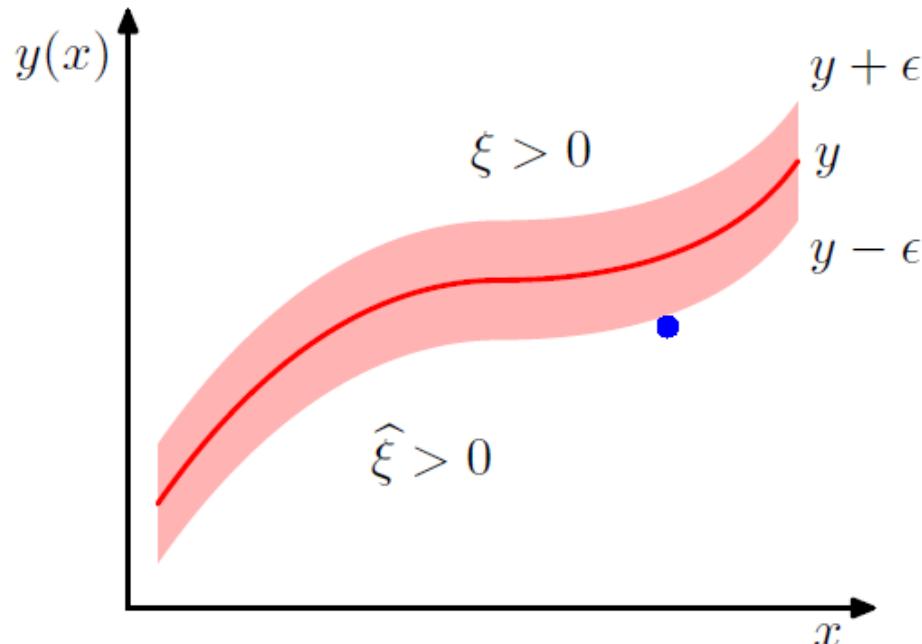
$\hat{\xi}_n > 0$ corresponds to a point $t_n < y(\mathbf{x}_n) - \epsilon$

SVMs for regression

- For the points lying $y_n - \epsilon \leq t_n \leq y_n + \epsilon$ $\xi = \hat{\xi} = 0$
- The error function for support vector regression can then be written by

$$C \sum_{n=1}^N (\xi_n + \hat{\xi}_n) + \frac{1}{2} \|\mathbf{w}\|^2$$

- Fig 7.7



SVMs for regression

- This function should be minimized subject to the constraints $\xi_n \geq 0$ and $\hat{\xi}_n \geq 0$ and

$$t_n \leq y(\mathbf{x}_n) + \epsilon + \xi_n \text{ and } t_n \geq y(\mathbf{x}_n) - \epsilon - \hat{\xi}_n$$

—————

Allowing points to lie outside the tube

- Lagrange optimization:

$$\begin{aligned} L = & C \sum_{n=1}^N (\xi_n + \hat{\xi}_n) + \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N (\mu_n \xi_n + \hat{\mu}_n \hat{\xi}_n) \\ & - \sum_{n=1}^N a_n (\epsilon + \xi_n + y_n - t_n) - \sum_{n=1}^N \hat{a}_n (\epsilon + \hat{\xi}_n - y_n + t_n) \end{aligned}$$

SVMs for regression

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^N (a_n - \hat{a}_n) \phi(\mathbf{x}_n)$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^N (a_n - \hat{a}_n) = 0$$

$$\frac{\partial L}{\partial \xi_n} = 0 \Rightarrow a_n + \mu_n = C$$

$$\frac{\partial L}{\partial \hat{\xi}_n} = 0 \Rightarrow \hat{a}_n + \hat{\mu}_n = C$$

- Dual problem involves maximizing

$$\begin{aligned}\tilde{L}(\mathbf{a}, \hat{\mathbf{a}}) &= -\frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N (a_n - \hat{a}_n)(a_m - \hat{a}_m) k(\mathbf{x}_n, \mathbf{x}_m) \\ &\quad - \epsilon \sum_{n=1}^N (a_n + \hat{a}_n) + \sum_{n=1}^N (a_n - \hat{a}_n) t_n\end{aligned}$$

SVMs for regression

- with respect to $\{a_n\}$ and $\{\hat{a}_n\}$ where $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$
- Constrained optimization with constraints $a_n \geq 0$ and $\hat{a}_n \geq 0$
- Together with $\mu_n \geq 0$ and $\hat{\mu}_n \geq 0$, because $a_n + \mu_n = C$
 $\hat{a}_n + \hat{\mu}_n = C$

$\Rightarrow a_n \leq C$ and $\hat{a}_n \leq C$ we have the box constraints

$$0 \leq a_n \leq C \quad \text{and} \quad 0 \leq \hat{a}_n \leq C$$

- Also, from the result, we have

$$y(\mathbf{x}) = \sum_{n=1}^N (a_n - \hat{a}_n) k(\mathbf{x}, \mathbf{x}_n) + b$$

in terms of kernel function

SVMs for regression

- The Karush-Kuhn-Tucker (KKT) conditions is given by

$$a_n(\epsilon + \xi_n + y_n - t_n) = 0 \quad \hat{a}_n(\epsilon + \hat{\xi}_n - y_n + t_n) = 0$$

$$(C - a_n)\xi_n = 0 \quad (C - \hat{a}_n)\hat{\xi}_n = 0$$

- Product of the dual variables and the constraints must vanish.

$a_n \neq 0 \rightarrow \epsilon + \xi_n + y_n - t_n = 0$ data point either lies on the upper boundary $\xi_n = 0$ or above the upper boundary $\xi_n > 0$

$\hat{a}_n \neq 0 \rightarrow \epsilon + \hat{\xi}_n - y_n + t_n = 0$ data point either lies on the upper boundary $\hat{\xi}_n = 0$ or above the upper boundary $\hat{\xi}_n > 0$

- Two constraints are incompatible plus them $2\epsilon + \xi_n + \hat{\xi}_n = 0$
- For each x_n , either a_n or \hat{a}_n or both must be zero

SVMs for regression

- For those points lying on boundary or outside the tube, i.e. $a_n \neq 0$ or $\hat{a}_n \neq 0$, they are used for prediction. All points $a_n = \hat{a}_n = 0$ within the tube are vanished.
- Parameter “ b ” can be found by considering a point with $0 < a_n < C$, i.e. $\xi_n = 0 \rightarrow \epsilon + y_n - t_n = 0$
- Then,
$$\begin{aligned} b &= t_n - \epsilon - \mathbf{w}^T \phi(\mathbf{x}_n) \\ &= t_n - \epsilon - \sum_{m=1}^N (a_m - \hat{a}_m) k(\mathbf{x}_n, \mathbf{x}_m) \end{aligned}$$
- Also, we can find b from $0 < \hat{a}_n < C$. It is better to average over all such estimates of b .

SVMs for regression

- Alternative, instead of ϵ -insensitive region, we fix a parameter ν that bounds the fraction of points lying outside the tube. (Schölkopf *et al.*, 2000)

$$\begin{aligned}\tilde{L}(\mathbf{a}, \hat{\mathbf{a}}) = & -\frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N (a_n - \hat{a}_n)(a_m - \hat{a}_m) k(\mathbf{x}_n, \mathbf{x}_m) \\ & + \sum_{n=1}^N (a_n - \hat{a}_n) t_n\end{aligned}$$

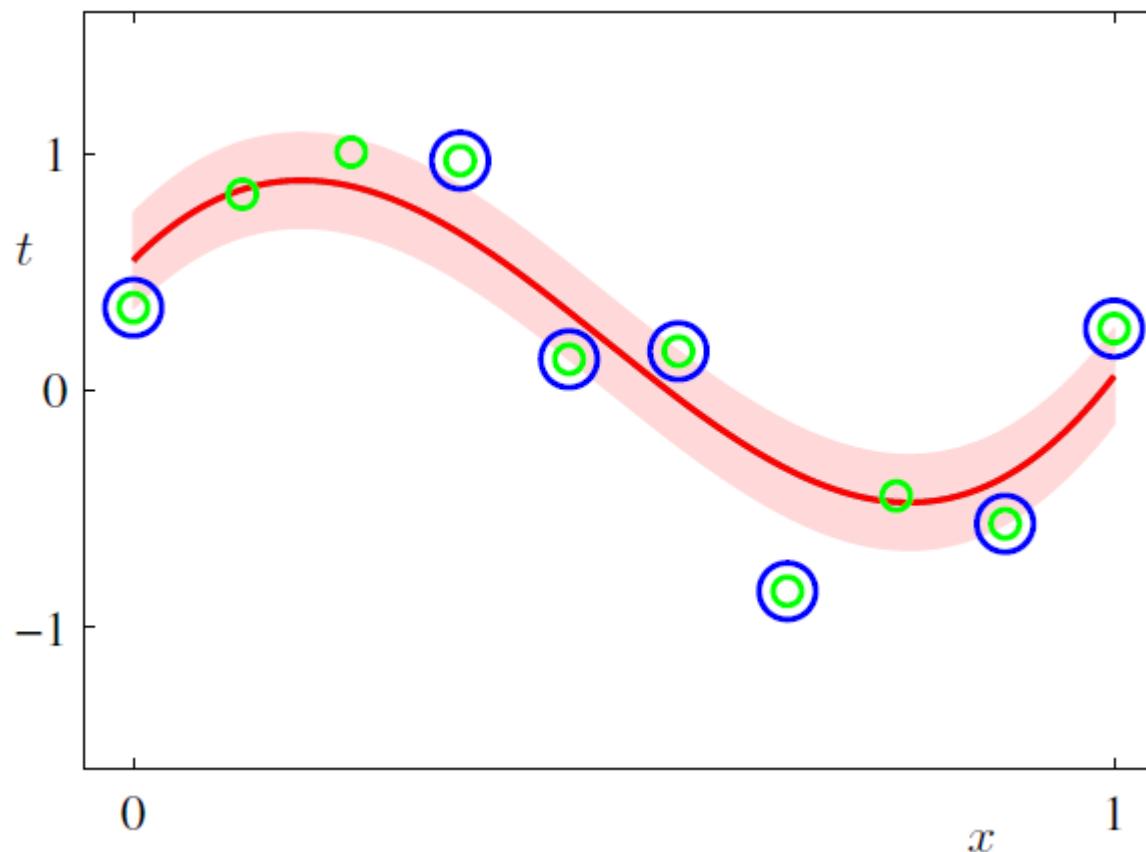
subject to the constraints

$$0 \leq a_n \leq C/N \quad 0 \leq \hat{a}_n \leq C/N$$

$$\sum_{n=1}^N (a_n - \hat{a}_n) = 0 \quad \sum_{n=1}^N (a_n + \hat{a}_n) \leq \nu C$$

SVMs for regression

- At most νN points falling outside the insensitive tube.
- Fig 7.8



Computational learning theory

- support vector machines → computational learning theory
→ statistical learning theory
- **PAC** (Probably Approximately Correct) The goal of the PAC is to understand how large a data set needs to be in order to give good “generalization”.
- $f(x; \mathcal{D})$ has good generalization if

$$\text{Prob}\{\mathbb{E}_{x,t} [I(f(x; \mathcal{D}) \neq t)] < \epsilon\}$$

$t = g(x)$ unknown deterministic

PAC learning is to find $f(x; \mathcal{D}) \rightarrow g(x)$

$p(x, t)$ joint distribution

PAC learning aims to provide bounds on the minimum N needed to meet the requirement.

Computational learning theory

- Vapnik-Chervonenkis (**VC**) dimension provides a measure of the complexity of a space of functions.
- Because lacking assumptions about model distribution. PAC bounds are very conservative, → strongly overestimate, the size of data sets to achieve a given generalization. → *PAC-Bayesian*

Relevance Vector Machines

- Limitations of **SVM**
 1. Outputs of SVM are decision rather than posterior probabilities.
 2. Two classes classification.
 3. Complexity parameter C or ν should be found from held-out data.(cross-validation)
 4. Predictions are based on linear combinations of kernel functions → centered on training data kernel should be positive definite.
- **RVM** is a Bayesian sparse kernel technique for regression and classification, sparse faster performance on test data while maintaining comparable generalization error.

RVM for regression

- A linear model is studied.

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}), \beta^{-1}) \quad \beta = \sigma^{-2} \text{ noise precision}$$

Mean: $y(\mathbf{x}) = \sum_{i=1}^M w_i \phi_i(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$

In RVM, the SVM-like form is used.

$$y(\mathbf{x}) = \sum_{n=1}^N w_n k(\mathbf{x}, \mathbf{x}_n) + b$$

- | | | : $y(\mathbf{x}) = \sum_{n=1}^N (a_n - \hat{a}_n) \times k(\mathbf{x}_1, \mathbf{x}_n) + b$

$$(\mathbf{w}, b) \quad M = N + 1 \text{ number of parameters}$$

RVM for regression

- In contrast to the SVM, no restriction to positive definite kernels.
- Let $\mathbf{X} = [\mathbf{x}_1^T, \dots, \mathbf{x}_N^T]$ $\mathbf{t} = (t_1, \dots, t_N)^T$

Likelihood function

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^N p(t_n|\mathbf{x}_n, \mathbf{w}, \beta^{-1})$$

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{i=1}^M \mathcal{N}(w_i|0, \alpha_i^{-1}) \quad \text{prior distribution } \mathbf{w}$$
$$(\alpha_1, \dots, \alpha_M)^T$$

- Posterior distribution (3.49) $p(\mathbf{w}|\mathbf{t}, \mathbf{X}, \boldsymbol{\alpha}, \beta) = \mathcal{N}(\mathbf{w}|\mathbf{m}, \boldsymbol{\Sigma})$

RVM for regression

- Where $\mathbf{m} = \beta \Sigma \Phi^T \mathbf{t}$

$$\Sigma = (\mathbf{A} + \beta \Phi^T \Phi)^{-1}$$

$\Phi_{N \times M}$ design matrix $= \{\Phi_{ni} = \phi_i(\mathbf{x}_n)\}$

$\mathbf{A} = \text{diag}(\alpha_i)$

$\Phi = \mathbf{K}_{(N+1) \times (N+1)}$ RVM in using

α, β are determined by “evidence approximation”.
or type II maximum likelihood.

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

$$\begin{aligned} \ln p(\mathbf{t}|\mathbf{X}, \boldsymbol{\alpha}, \beta) &= \ln \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C}) \\ &= -\frac{1}{2} \{ N \ln(2\pi) + \ln |\mathbf{C}| + \mathbf{t}^T \mathbf{C}^{-1} \mathbf{t} \} \end{aligned}$$

RVM for regression

- Where $\mathbf{C} = \beta^{-1} \mathbf{I} + \Phi \mathbf{A}^{-1} \Phi^T$

$$\nabla_{\alpha} \ln p(\mathbf{t} | \mathbf{X}, \boldsymbol{\alpha}, \beta) = 0 \quad \frac{\partial}{\partial \beta} \ln p(\mathbf{t} | \mathbf{X}, \boldsymbol{\alpha}, \beta) = 0$$

- Using the methods in Sec 3.5, we obtain the re-estimation equations

$$\alpha_i^{\text{new}} = \frac{\gamma_i}{m_i^2}$$

$$(\beta^{\text{new}})^{-1} = \frac{\|\mathbf{t} - \Phi \mathbf{m}\|^2}{N - \sum_i \gamma_i}$$

γ_i measures how well the corresponding parameter w_i is determined by the data

$$\gamma_i = 1 - \alpha_i \Sigma_{ii}$$

RVM for regression

- Starting from initial α and β evaluating m and $\Sigma \rightarrow \alpha$ and β ... until convergence.
- As a result of optimization, some hyperparameters $\{\alpha_i\}$ are driven to large values, \Rightarrow weight parameters $\{w_i\}$ with posterior distribution having zero mean and zero variance .
- Those parameters and $\phi_i(\mathbf{x})$ play no role in making predictions

$$y(\mathbf{x}) = \sum_{n=1}^N w_n k(\mathbf{x}_1, \mathbf{x}_n) + b$$

- The inputs \mathbf{x}_n corresponding to nonzero weights are called “relevance vectors” analogous to support vectors of an SVM
- Automatic Relevance Determination

7.2.1 RVM for regression

- α^* , β^* maximize the marginal likelihood

$$\begin{aligned} p(t|\mathbf{x}, \mathbf{X}, \mathbf{t}, \alpha^*, \beta^*) &= \int p(t|\mathbf{x}, \mathbf{w}, \beta^*) p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \alpha^*, \beta^*) d\mathbf{w} \\ &= \mathcal{N}(t | \mathbf{m}^T \boldsymbol{\phi}(\mathbf{x}), \sigma^2(\mathbf{x})) \end{aligned}$$

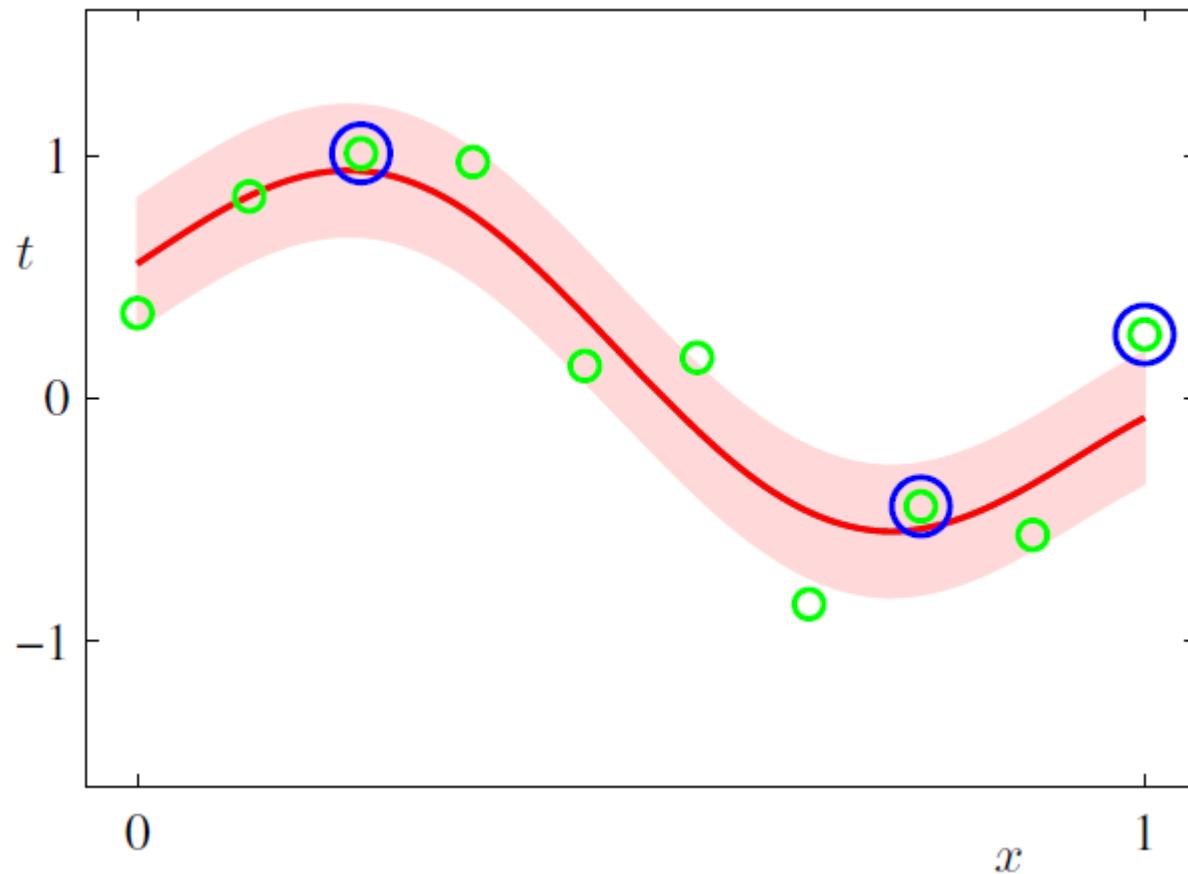
- Where

$$\sigma^2(\mathbf{x}) = (\beta^*)^{-1} + \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\Sigma} \boldsymbol{\phi}(\mathbf{x})$$

- Similar results were obtained in the context of linear regression.

RVM for regression

- Fig 7.9



RVM for regression

- Number of support vectors is significantly larger than number of relevance vectors.
- RVM is more compact than SVM test computational load is reduced a lot.
- Greater sparsity is achieved with little reduction in generalization error .
- Main disadvantage of RVM compared to SVM is that training involves optimizing a “non convex” function. Training time can be longer. Using RVM, parameter complexity and noise variance are determined from a single training run.
- But, SVM needs multiple runs for cross validation of C and ϵ (or ν) RVM requires matrix inversion.

Analysis of sparsity

- Consider a dataset $\{t_1, t_2\}$ $N = 2$ and $\phi(\mathbf{x})$ single basis, marginal likelihood $p(\mathbf{t}|\alpha, \beta) = \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C})$

where

$$\mathbf{C} = \frac{1}{\beta} \mathbf{I} + \frac{1}{\alpha} \boldsymbol{\varphi} \boldsymbol{\varphi}^T$$

- Then,
$$|\mathbf{C}| = |\mathbf{C}_{-i}| |1 + \alpha_i^{-1} \boldsymbol{\varphi}_i^T \mathbf{C}_{-i}^{-1} \boldsymbol{\varphi}_i|$$

$$\mathbf{C}^{-1} = \mathbf{C}_{-i}^{-1} - \frac{\mathbf{C}_{-i}^{-1} \boldsymbol{\varphi}_i \boldsymbol{\varphi}_i^T \mathbf{C}_{-i}^{-1}}{\alpha_i + \boldsymbol{\varphi}_i^T \mathbf{C}_{-i}^{-1} \boldsymbol{\varphi}_i}$$
- We can write $L(\alpha) = L(\alpha_{-i}) + \lambda(\alpha_i)$
- Where
$$\lambda(\alpha_i) = \frac{1}{2} \left[\ln \alpha_i - \ln (\alpha_i + s_i) + \frac{q_i^2}{\alpha_i + s_i} \right]$$

Analysis of sparsity

$$s_i = \varphi_i^T \mathbf{C}_{-i}^{-1} \varphi_i \quad q_i = \varphi_i^T \mathbf{C}_{-i}^{-1} \mathbf{t}$$

$s_i \nearrow$ means that φ_i is more likely to be pruned.

s_i relative to q_i means that φ_i too over lapping

$$\frac{d\lambda(\alpha_i)}{d\alpha_i} = \frac{\alpha_i^{-1} s_i^2 - (q_i^2 - s_i)}{2(\alpha_i + s_i)^2}$$

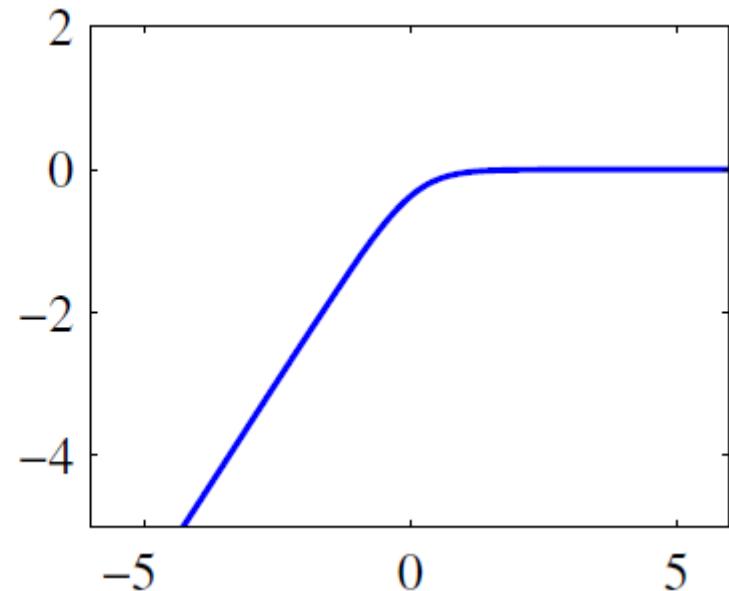
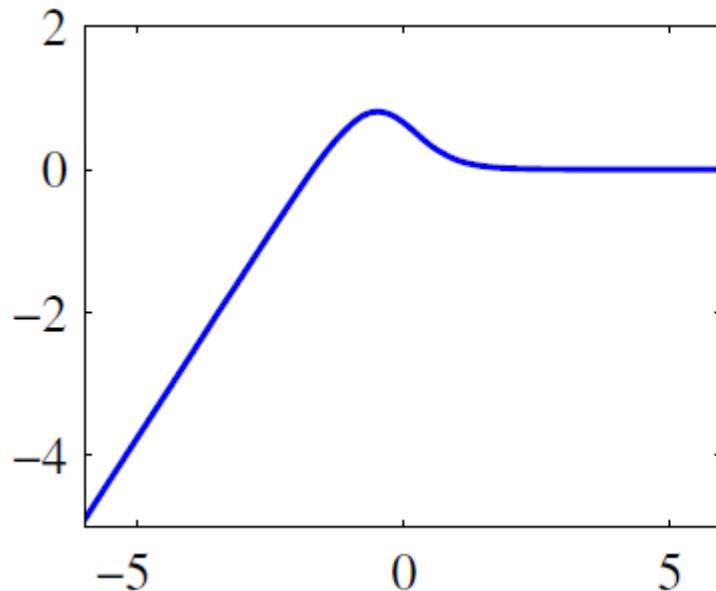
→ Find stationary points of the marginal likelihood with respect to α_i remove

$\alpha_i \geq 0$ if $q_i^2 < s_i$ then $\alpha_i \rightarrow \infty$ provides solution.

$$\text{if } q_i^2 > s_i \text{ then } \alpha_i = \frac{s_i^2}{q_i^2 - s_i}$$

7.2.2 Analysis of sparsity

- Fig 7.10



- Relative size of q_i and s_i determined whether a particular basis will be pruned.

Sequential Sparse Bayesian Learning Algorithm

- Initialize β
- Initialize φ_1 and find α_1 , the remaining $\alpha_j \rightarrow \infty$, $j \neq i$ only φ_1 is included.
- Evaluate Σ and \mathbf{m} , along with q_i and s_i for all basis functions φ_i .
- Select a candidate φ_i .
- If $q_i^2 > s_i$ and $\alpha_i < \infty$, φ_i is already included in the model, then update α_i by $\alpha_i = \frac{s_i^2}{q_i^2 - s_i}$
- If $q_i^2 > s_i$ and $\alpha_i < \infty$, then remove φ_i from the model and set $\alpha_i = \infty$.
- Update β .
- Terminate when converged, otherwise go to 3.

Analysis of sparsity

- In practice, it is convenient to evaluate

$$q_i = \frac{\alpha_i Q_i}{\alpha_i - S_i} \quad s_i = \frac{\alpha_i S_i}{\alpha_i - S_i}$$

- The quality and sparseness variables can then be obtained by

$$\begin{aligned} Q_i &= \beta \varphi_i^T \mathbf{t} - \beta^2 \varphi_i^T \Phi \Sigma \Phi^T \mathbf{t} \\ S_i &= \beta \varphi_i^T \varphi_i - \beta^2 \varphi_i^T \Phi \Sigma \Phi^T \varphi_i \end{aligned}$$

RVM for classification

- By applying the ARD (automatic relevance determination) prior over weights, RVM can be extended to classification problem.

$$t \in \{0, 1\} \quad y(\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^T \phi(\mathbf{x}))$$

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^N p(t_n|\mathbf{x}_n, \mathbf{w}, \beta^{-1})$$

- Considering Laplace approximation, for a fixed α , \mathbf{w}_{MAP} is obtained by maximizing

$$\begin{aligned} \ln p(\mathbf{w}|\mathbf{t}, \alpha) &= \ln \{p(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\alpha)\} - \ln p(\mathbf{t}|\alpha) \\ &= \sum_{n=1}^N \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\} - \frac{1}{2} \mathbf{w}^T \mathbf{A} \mathbf{w} + \text{const} \end{aligned}$$

RVM for classification

- This can be done by iterative reweighted least squares(IRLS)

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

- where $\begin{aligned} \nabla \ln p(\mathbf{w}|\mathbf{t}, \alpha) &= \Phi^T(\mathbf{t} - \mathbf{y}) - \mathbf{A}\mathbf{w} \\ \nabla \nabla \ln p(\mathbf{w}|\mathbf{t}, \alpha) &= -(\Phi^T \mathbf{B} \Phi + \mathbf{A}) \end{aligned}$

- Gaussian approximation to $p(\mathbf{w}|\mathbf{t}, \alpha)$ by using

$$\beta = \text{diag}\{y_n(1 - y_n)\}$$

- Laplace approximation $\Phi = \{\phi_{ni}\}$

- Gaussian mean is obtained by $\nabla \ln p(\mathbf{w}|\mathbf{t}, \alpha) = 0$

i.e. made $\mathbf{w}^* = \mathbf{A}^{-1} \Phi^T(\mathbf{t} - \mathbf{y})$

Gaussian covariance matrix $\Sigma = -\mathbf{H}^{-1} = (\Phi^T \mathbf{B} \Phi + \mathbf{A})^{-1}$

RVM for classification

- Marginal likelihood is obtained.

- Also,
$$\begin{aligned} p(\mathbf{t}|\boldsymbol{\alpha}) &= \int p(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha}) d\mathbf{w} \\ &\simeq p(\mathbf{t}|\mathbf{w}^*)p(\mathbf{w}^*|\boldsymbol{\alpha})(2\pi)^{M/2}|\Sigma|^{1/2} \end{aligned}$$

$$\frac{\partial p(\mathbf{t}|\boldsymbol{\alpha})}{\partial \alpha_i} = 0 \text{ we have } -\frac{1}{2}(w_i^*)^2 + \frac{1}{2\alpha_i} - \frac{1}{2}\Sigma_{ii} = 0$$

$$\text{Let } \gamma_i = 1 - \alpha_i \Sigma_{ii} \rightarrow \alpha_i^{\text{new}} = \frac{\gamma_i}{(w_i^*)^2}$$

Identical to the formula for regression RVM

If we define $\hat{\mathbf{t}} = \Phi \mathbf{w}^* + \mathbf{B}^{-1}(\mathbf{t} - \mathbf{y})$,

We can write $\ln p(\mathbf{t}|\boldsymbol{\alpha}, \beta) = -\frac{1}{2} \left\{ N \ln(2\pi) + \ln |\mathbf{C}| + (\hat{\mathbf{t}})^T \mathbf{C}^{-1} \hat{\mathbf{t}} \right\}$

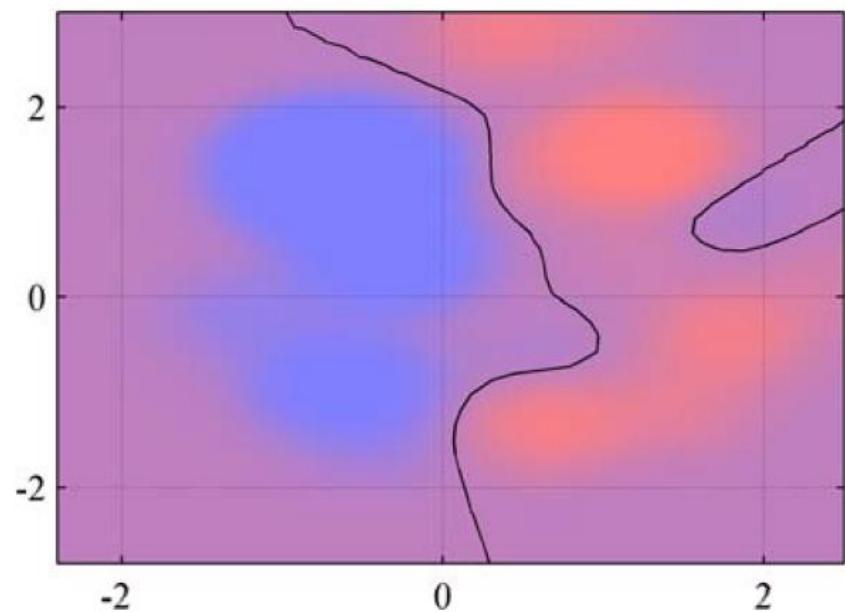
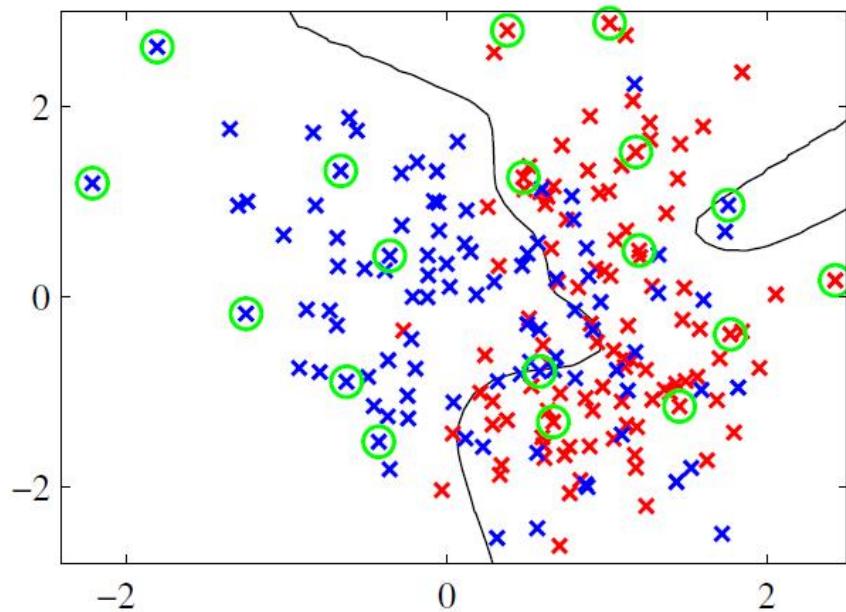
RVM for classification

- Where $\mathbf{C} = \mathbf{B} + \Phi \mathbf{A} \Phi^T$
- Take the same form as RVM regression.

$$\begin{aligned}\ln p(\mathbf{t}|\mathbf{X}, \boldsymbol{\alpha}, \beta) &= \ln \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C}) \\ &= -\frac{1}{2} \{ N \ln(2\pi) + \ln |\mathbf{C}| + \mathbf{t}^T \mathbf{C}^{-1} \mathbf{t} \}\end{aligned}$$

- See Fig 7.12 Relevance vectors indicated by circles. RVM gives a much sparser model. Compared by SVM Fig (7.4)
- Relevance vectors tend not to lie in the region of decision boundary in contrast to SVM.
- $\phi_i(\mathbf{x})$ centered on a data point “near” the boundary will have a vector φ_i that is poorly aligned with \mathbf{t} .

• Fig 7.12



RVM for classification

- In case of $K > 2$ classes, there are K linear models of sec 4.3.4
- A softmax function is used to give outputs

$$y_k(\mathbf{x}) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

- The log likelihood function

$$\ln p(\mathbf{T}|\mathbf{w}_1, \dots, \mathbf{w}_K) = \prod_{n=1}^N \prod_{k=1}^K y_{nk}^{t_{nk}}$$

- One disadvantage of RVM is the relatively long training time compared to SVM.

CONTENTS

1. Introduction
2. Probability Distributions
3. Linear Models for Regression
4. Linear Models for Classification
5. Kernel Methods
6. Sparse Kernel Methods
7. Mixture Models and EM
8. Approximate Inference

Mixture Models and EM

- A general technique for finding maximum likelihood estimators in latent variable models is the expectation – maximum (EM) algorithm.

9.1 K-means clustering

- Suppose we have a data set $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ consisting of N observations of a random D -dimensional Euclidean variable \mathbf{x} .
- Our goal is to partition the data set into some number K of clusters, where we shall suppose for the moment that the value of K is given.

9.1 K-means clustering

- We can formalize this notion by first introducing a set of D -dimensional vectors μ_k , where $k = 1, \dots, K$, in which μ_k is a prototype associated with the k^{th} cluster.
- Our goal is then to find an assignment of data points to clusters, as well as a set of vectors $\{\mu_k\}$, such that the sum of the squares of the distances of each data point to its closest vector μ_k , is a minimum.
- Introduce a corresponding set of binary indicator variables $r_{nk} \in \{0, 1\}$, where $k = 1, \dots, K$

9.1 K-means clustering

- 1-of- K coding :
if data point \mathbf{x}_n is assigned to cluster k , then
 $r_{nk} = 1$, and $r_{nj} = 0$ for $j \neq k$
- Define an objective function, sometimes called a distortion measure :
$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$
- Our goal is to find values for the $\{r_{nk}\}$ and the $\{\boldsymbol{\mu}_k\}$ so as to minimize J .

9.1 K-means clustering

- Two successive steps:
 - minimize J with respect to the r_{nk} , keeping the μ_k fixed
 - minimize J with respect to the μ_k , keeping r_{nk} fixedrepeated until convergence
- Updating r_{nk} and updating μ_k correspond respectively to the E (expectation) and M (maximization) steps of the EM algorithm

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j \| \mathbf{x}_n - \boldsymbol{\mu}_j \|^2 \\ 0 & \text{otherwise.} \end{cases}$$

9.1 K-means clustering

- The objective function J is a quadratic function of μ_k , and it can be minimized by setting its derivative with respect to μ_k to zero giving

$$2 \sum_{n=1}^N r_{nk}(\mathbf{x}_n - \boldsymbol{\mu}_k) = 0$$

- Solve μ_k :
- Set μ_k equal to the mean of all of the data points \mathbf{x}_n assigned to cluster k , it is known as the *K-means* algorithm.

9.1 K-means clustering

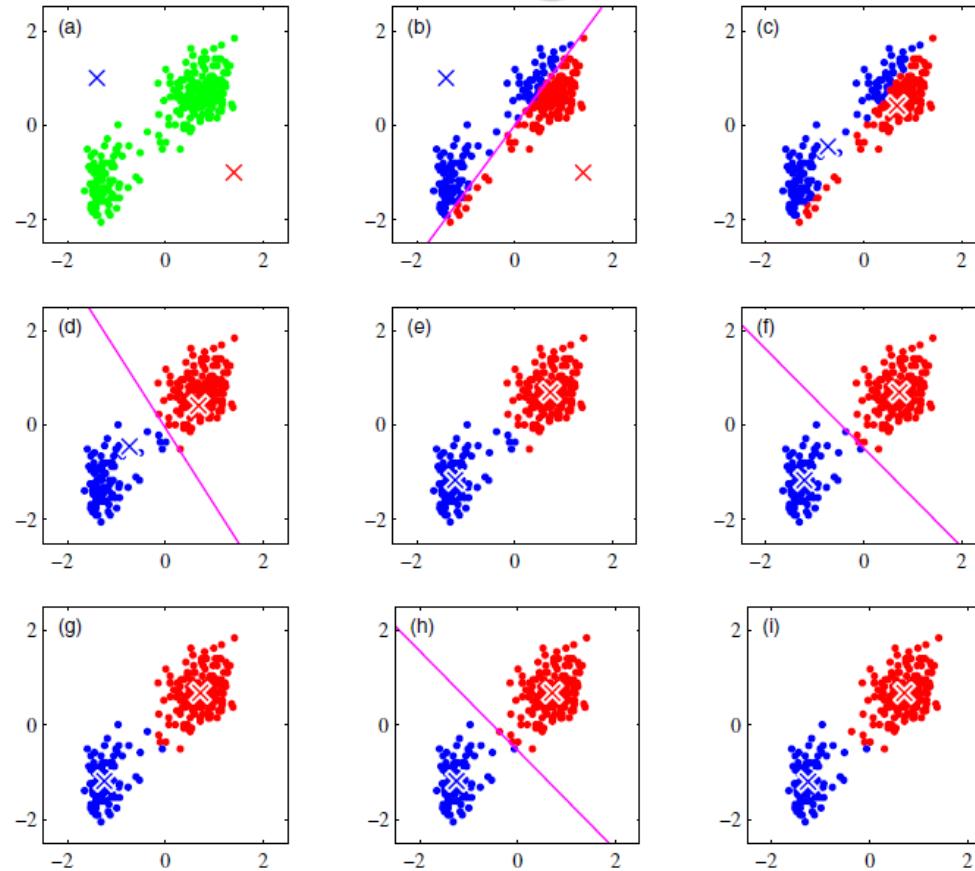
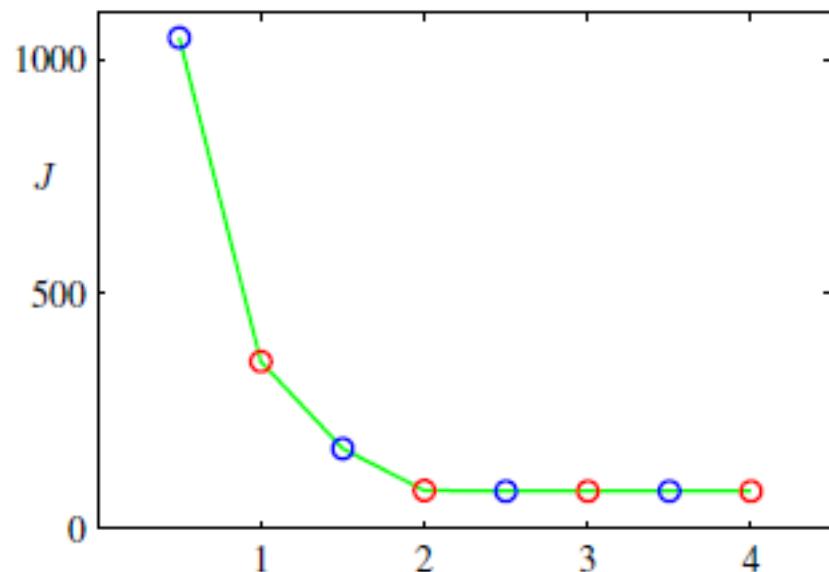


Figure 9.1 Illustration of the K -means algorithm using the re-scaled Old Faithful data set. (a) Green points denote the data set in a two-dimensional Euclidean space. The initial choices for centres μ_1 and μ_2 are shown by the red and blue crosses, respectively. (b) In the initial E step, each data point is assigned either to the red cluster or to the blue cluster, according to which cluster centre is nearer. This is equivalent to classifying the points according to which side of the perpendicular bisector of the two cluster centres, shown by the magenta line, they lie on. (c) In the subsequent M step, each cluster centre is re-computed to be the mean of the points assigned to the corresponding cluster. (d)–(l) show successive E and M steps through to final convergence of the algorithm.

9.1 K-means clustering

Figure 9.2 Plot of the cost function J given by (9.1) after each E step (blue points) and M step (red points) of the K -means algorithm for the example shown in Figure 9.1. The algorithm has converged after the third M step, and the final EM cycle produces no changes in either the assignments or the prototype vectors.



9.2 Mixtures of Gaussians

- Gaussian mixture distribution

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad z_k \in \{0, 1\} \text{ and } \sum_k z_k = 1.$$

- The marginal distribution over \mathbf{z} is specified in terms of the mixing coefficients π_k , such that

$$p(z_k = 1) = \pi_k$$

$$0 \leq \pi_k \leq 1$$

$$\sum_{k=1}^K \pi_k = 1$$

9.2 Mixtures of Gaussians

- \mathbf{z} uses a 1-of- K representation :

$$p(\mathbf{z}) = \prod_{k=1}^K \pi_k^{z_k}$$

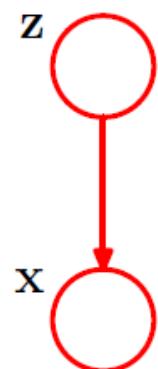
$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^K \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}$$

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

9.2 Mixtures of Gaussians

Figure 9.4 Graphical representation of a mixture model, in which the joint distribution is expressed in the form $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$.



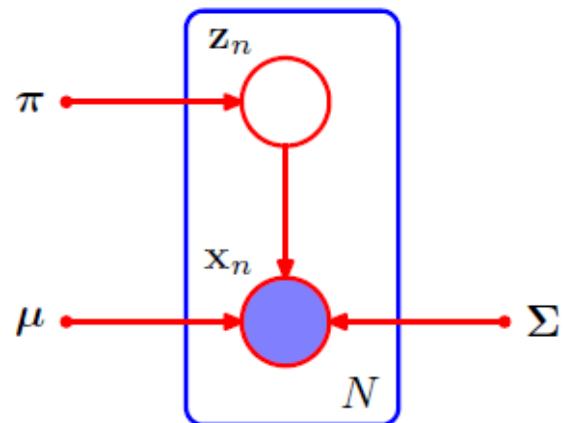
9.2.1 Maximum likelihood

- Suppose we have a data set of observations $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, and we wish to model this data using a mixture of Gaussians.
- Latent variables will be denoted by an $N \times K$ matrix Z with rows \mathbf{z}_n^T .
- Log of the likelihood function is given by

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

9.2.1 Maximum likelihood

Figure 9.6 Graphical representation of a Gaussian mixture model for a set of N i.i.d. data points $\{x_n\}$, with corresponding latent points $\{z_n\}$, where $n = 1, \dots, N$.



9.2.2 EM for Gaussian mixtures

- expectation-maximization algorithm, or EM algorithm (Dempster et al., 1977; McLachlan and Krishnan, 1997)
- Setting the derivatives of $\ln p(\mathbf{X}|\pi, \mu, \Sigma)$ with respect to the means μ_k of the Gaussian components to zero, we obtain

$$0 = - \sum_{n=1}^N \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n | \mu_j, \Sigma_j)}}_{\gamma(z_{nk})} \Sigma_k (\mathbf{x}_n - \mu_k)$$

- Multiplying by Σ_k and rearranging we obtain

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n \quad \text{, where } N_k = \sum_{n=1}^N \gamma(z_{nk})$$

9.2.2 EM for Gaussian mixtures

- If we set the derivative of $\ln p(\mathbf{X}|\pi, \mu, \Sigma)$ with respect to Σ_k to zero,

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T$$

- Using a Lagrange multiplier and maximizing the following quantity

$$\ln p(\mathbf{X}|\pi, \mu, \Sigma) + \lambda \left(\sum_{k=1}^K \pi_k - 1 \right)$$

which gives

$$0 = \sum_{n=1}^N \frac{\mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \Sigma_j)} + \lambda$$

$$\lambda = -N \quad \pi_k = \frac{N_k}{N}$$

9.2.2 EM for Gaussian mixtures

1. Initialize the means μ_k , covariances Σ_k and mixing coefficients π_k , and evaluate the initial value of the log likelihood.
2. **E step.** Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

9.2.2 EM for Gaussian mixtures

3. M step. Re-estimate the parameters using the current responsibilities

$$\begin{aligned}\mu_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n \\ \Sigma_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \mu_k^{\text{new}}) (\mathbf{x}_n - \mu_k^{\text{new}})^T \\ \pi_k^{\text{new}} &= \frac{N_k}{N}\end{aligned}$$

where

$$N_k = \sum_{n=1}^N \gamma(z_{nk}).$$

9.2.2 EM for Gaussian mixtures

4. Evaluate the log likelihood

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

9.3 An Alternative View of EM

- The goal of the EM algorithm is to find maximum likelihood solutions for models having latent variables.
- We denote the set of all observed data by \mathbf{X} , in which the n^{th} row represents \mathbf{x}_n^T , and similarly we denote the set of all latent variables by \mathbf{Z} , with a corresponding row \mathbf{z}_n^T .
- Log likelihood function is given by

$$\ln p(\mathbf{X}|\theta) = \ln \left\{ \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\theta) \right\}$$

9.3 An Alternative View of EM

- The General EM Algorithm
 1. Choose an initial setting for the parameters θ^{old} .
 2. **E step** Evaluate $p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}})$
 3. **M step** Evaluate θ^{new} given by

$$\theta^{\text{new}} = \arg \max_{\theta} \mathcal{Q}(\theta, \theta^{\text{old}})$$

where

$$\mathcal{Q}(\theta, \theta^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\theta).$$

4. Check for convergence of either the log likelihood or the parameter values. If the convergence criterion is not satisfied, then let $\theta^{\text{old}} \leftarrow \theta^{\text{new}}$ and return to step 2.

9.3.1 Gaussian mixtures revisited

$$p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \prod_{n=1}^N \prod_{k=1}^K \pi_k^{z_{nk}} \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_{nk}}$$

$$\ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^N \sum_{k=1}^K z_{nk} \{ \ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \}$$

$$\begin{aligned}\mathbb{E}[z_{nk}] &= \frac{\sum_{j=1}^K z_{nj} [\pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)]^{z_{nj}}}{\sum_{j=1}^K [\pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)]^{z_{nj}}} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} = \gamma(z_{nk})\end{aligned}$$

$$\mathbb{E}_{\mathbf{Z}}[\ln p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})] = \sum_{n=1}^N \sum_{k=1}^K \gamma(z_{nk}) \{ \ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \}$$

9.3.2 Relation to K-means

$$p(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(2\pi\epsilon)^{1/2}} \exp \left\{ -\frac{1}{2\epsilon} \|\mathbf{x} - \boldsymbol{\mu}_k\|^2 \right\}$$

- Consider the EM algorithm for a mixture of K Gaussians

$$\gamma(z_{nk}) = \frac{\pi_k \exp \{-\|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2/2\epsilon\}}{\sum_j \pi_j \exp \{-\|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2/2\epsilon\}}$$

- In the limit $\epsilon \rightarrow 0$ the expected complete-data log likelihood

$$\mathbb{E}_{\mathbf{Z}}[\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})] \rightarrow -\frac{1}{2} \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 + \text{const.}$$

9.3.3 Mixtures of Bernoulli distributions

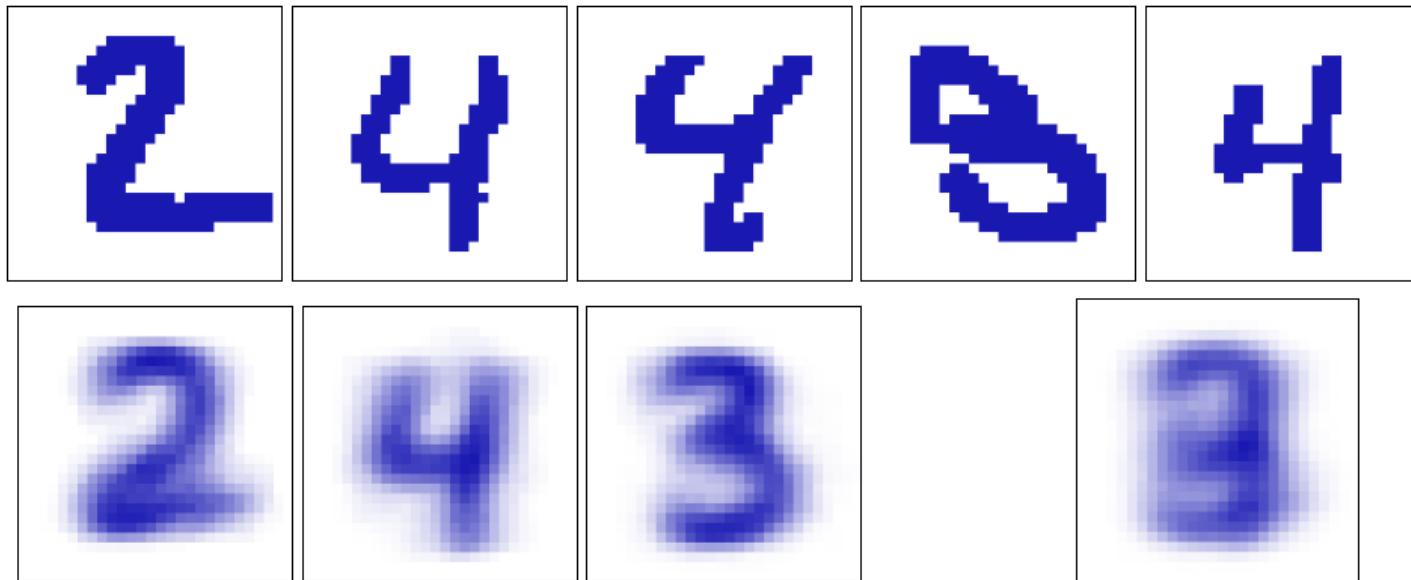


Figure 9.10 Illustration of the Bernoulli mixture model in which the top row shows examples from the digits data set after converting the pixel values from grey scale to binary using a threshold of 0.5. On the bottom row the first three images show the parameters μ_{ki} for each of the three components in the mixture model. As a comparison, we also fit the same data set using a single multivariate Bernoulli distribution, again using maximum likelihood. This amounts to simply averaging the counts in each pixel and is shown by the right-most image on the bottom row.

9.3.4 EM for Bayesian linear regression

- The complete-data log likelihood function is then given by

$$\ln p(\mathbf{t}, \mathbf{w} | \alpha, \beta) = \ln p(\mathbf{t} | \mathbf{w}, \beta) + \ln p(\mathbf{w} | \alpha)$$

$$\begin{aligned}\mathbb{E} [\ln p(\mathbf{t}, \mathbf{w} | \alpha, \beta)] &= \frac{M}{2} \ln \left(\frac{\alpha}{2\pi} \right) - \frac{\alpha}{2} \mathbb{E} [\mathbf{w}^T \mathbf{w}] + \frac{N}{2} \ln \left(\frac{\beta}{2\pi} \right) \\ &\quad - \frac{\beta}{2} \sum_{n=1}^N \mathbb{E} [(t_n - \mathbf{w}^T \boldsymbol{\phi}_n)^2].\end{aligned}$$

- Setting the derivatives with respect to α to zero, we obtain the M step re-estimation equation

$$\alpha = \frac{M}{\mathbb{E} [\mathbf{w}^T \mathbf{w}]} = \frac{M}{\mathbf{m}_N^T \mathbf{m}_N + \text{Tr}(\mathbf{S}_N)}$$

- An analogous result holds for β .

9.4. The EM Algorithm in General

- Our goal is to maximize the likelihood function that is given by

$$p(\mathbf{X}|\theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\theta)$$

$$\ln p(\mathbf{X}|\theta) = \mathcal{L}(q, \theta) + \text{KL}(q\|p)$$

where we have defined

$$\begin{aligned}\mathcal{L}(q, \theta) &= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z}|\theta)}{q(\mathbf{Z})} \right\} \\ \text{KL}(q\|p) &= - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X}, \theta)}{q(\mathbf{Z})} \right\}\end{aligned}$$

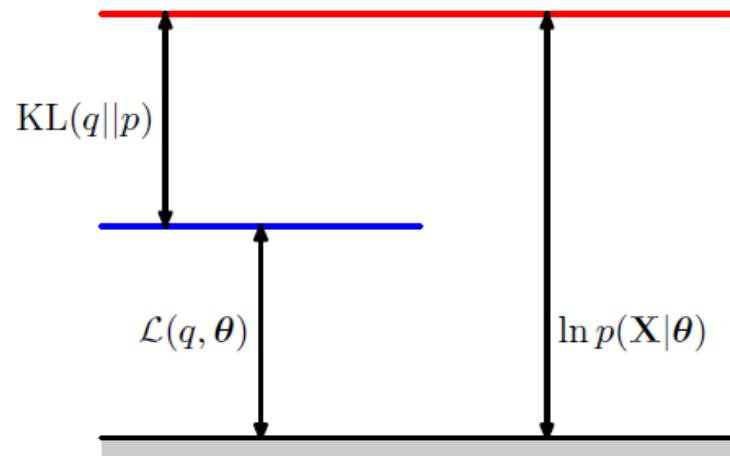
9.4. The EM Algorithm in General

- The equation can be also verified by substituting

$$\ln p(\mathbf{X}, \mathbf{Z}|\theta) = \ln p(\mathbf{Z}|\mathbf{X}, \theta) + \ln p(\mathbf{X}|\theta)$$

into $\mathcal{L}(q, \theta)$

Figure 9.11 Illustration of the decomposition given by (9.70), which holds for any choice of distribution $q(\mathbf{Z})$. Because the Kullback-Leibler divergence satisfies $\text{KL}(q||p) \geq 0$, we see that the quantity $\mathcal{L}(q, \theta)$ is a lower bound on the log likelihood function $\ln p(\mathbf{X}|\theta)$.



9.4. The EM Algorithm in General

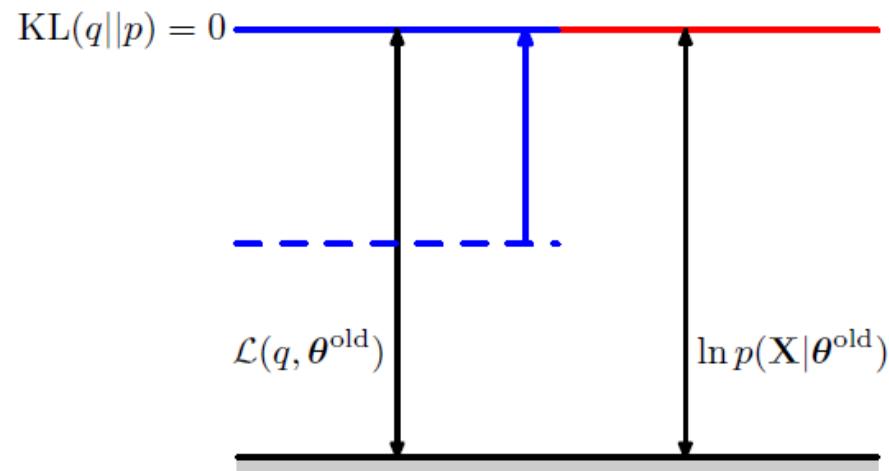
- Illustration for EM algorithm
1. Suppose that the current value of the parameter vector is θ^{old} . In the E step, the lower bound $\mathcal{L}(q, \theta^{\text{old}})$ is maximized with respect to $q(\mathbf{Z})$ while holding θ^{old} fixed. Largest value of $\mathcal{L}(q, \theta^{\text{old}})$ will occur when the Kullback-Leibler divergence vanishes, in other words when $q(\mathbf{Z})$ is equal to the posterior distribution $p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}})$
 2. In the subsequent M step, the distribution $q(\mathbf{Z})$ is held fixed and the lower bound $\mathcal{L}(q, \theta)$ is maximized with respect to θ to give some new value θ^{new} .

9.4. The EM Algorithm in General

- If we substitute $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}})$ after the E step, the lower bound takes the form

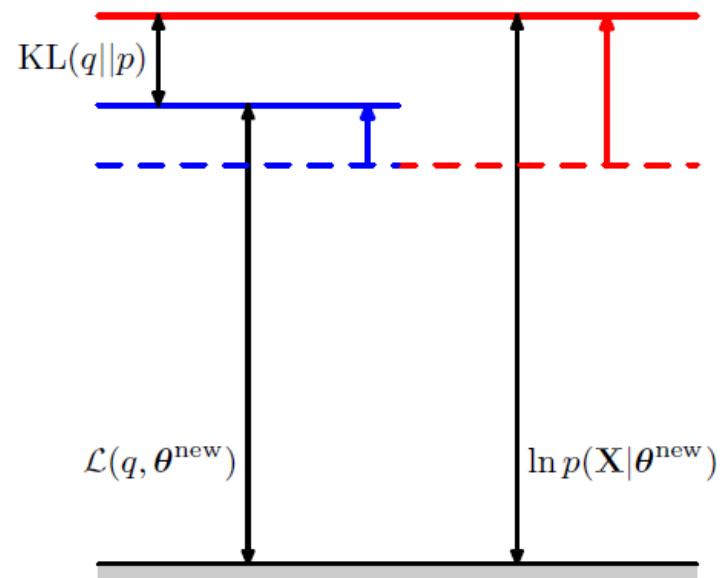
$$\begin{aligned}\mathcal{L}(q, \theta) &= \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\theta) - \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}}) \ln p(\mathbf{Z}|\mathbf{X}, \theta^{\text{old}}) \\ &= \mathcal{Q}(\theta, \theta^{\text{old}}) + \text{const}\end{aligned}\quad (9.74)$$

Figure 9.12 Illustration of the E step of the EM algorithm. The q distribution is set equal to the posterior distribution for the current parameter values θ^{old} , causing the lower bound to move up to the same value as the log likelihood function, with the KL divergence vanishing.



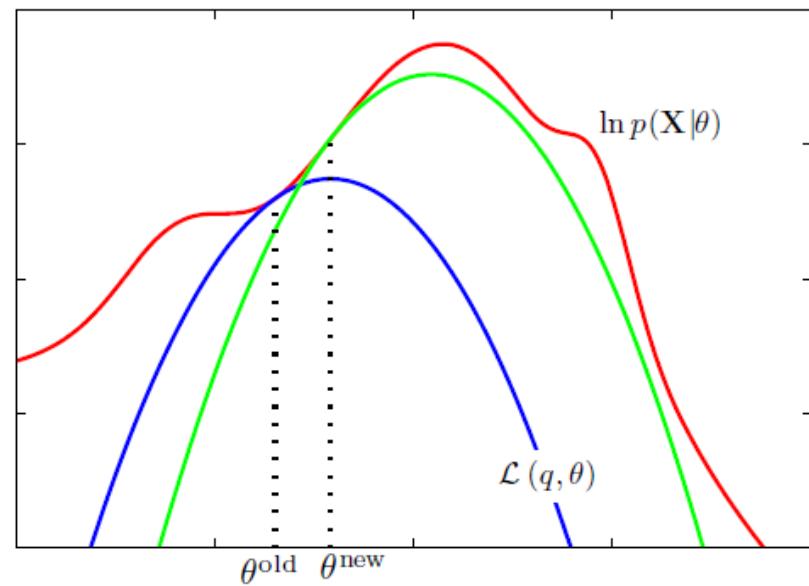
9.4. The EM Algorithm in General

Figure 9.13 Illustration of the M step of the EM algorithm. The distribution $q(\mathbf{Z})$ is held fixed and the lower bound $\mathcal{L}(q, \theta)$ is maximized with respect to the parameter vector θ to give a revised value θ^{new} . Because the KL divergence is nonnegative, this causes the log likelihood $\ln p(\mathbf{X}|\theta)$ to increase by at least as much as the lower bound does.



9.4. The EM Algorithm in General

Figure 9.14 The EM algorithm involves alternately computing a lower bound on the log likelihood for the current parameter values and then maximizing this bound to obtain the new parameter values. See the text for a full discussion.



CONTENTS

1. Introduction
2. Probability Distributions
3. Linear Models for Regression
4. Linear Models for Classification
5. Kernel Methods
6. Sparse Kernel Methods
7. Mixture Models and EM
8. Approximate Inference

Approximate Inference

- For many models of practical interest, it will be infeasible to evaluate the posterior distribution or indeed to compute expectations with respect to this distribution because of the dimensionality of latent space is too high.
- In the case of continuous variables, the required integrations may not have closed-form analytical solutions
- For discrete variables, the marginalizations involve summing over all possible configurations of the hidden variables.
- In such situations, we need to resort to approximation schemes (fall broadly into two classes, according whether they rely on *stochastic* or *deterministic* approximations)

Approximate Inference

- Stochastic techniques such as Markov Chain or Monte Carlo, have enabled the widespread use of Bayesian methods across many domains
- In this chapter, we introduce a range of deterministic approximation such as variational inference, variational Bayes.

10.1 Variational Inference

- Standard calculus is concerned with finding derivative of functions. Similarly, we can define a *functional* as a mapping that takes a function as the input and that returns the value of the functional as the output. An example would be $H[p]$, which takes a probability distribution $p(x)$ as the input and returns the quantity.
- Now let us consider in more detail how the concept of variational optimization can be applied to the inference problem. As in our discussion in EM, we can decompose the log marginal probability using

$$\ln p(\mathbf{X}) = \mathcal{L}(q) + KL(q||p)$$

Where $\mathcal{L}(q) = \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right\} d\mathbf{Z}$, $KL(q||p) = - \int q(Z) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})} \right\} d\mathbf{Z}$

10.1 Variational Inference

- This differs from EM only in that the parameter vector θ no longer appears, because the parameters are now stochastic and are absorbed into \mathbf{Z} .
- We therefore consider instead a restricted family of distribution $q(\mathbf{z})$ and then seek the member of this family for which KL divergence is minimized.

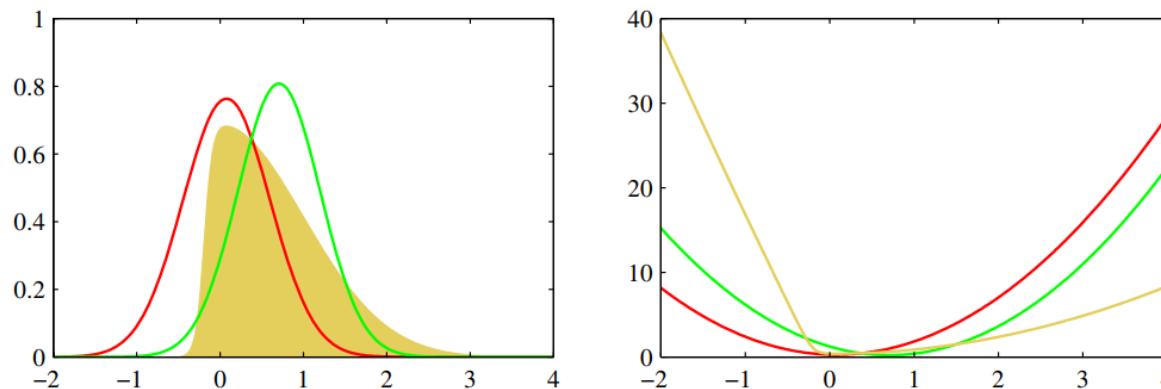


Figure 10.1 Illustration of the variational approximation for the example considered earlier in Figure 4.14. The left-hand plot shows the original distribution (yellow) along with the Laplace (red) and variational (green) approximations, and the right-hand plot shows the negative logarithms of the corresponding curves.

10.1.1 Factorized Distributions

- Here we consider an alternative way in which to restrict the family of distributions $q(\mathbf{z})$.
- Suppose we partition the elements of \mathbf{Z} into disjoint groups. We then assume that the q distribution factorizes with respect to these groups, so that $q(\mathbf{z}) = \prod_{i=1}^M q_i(\mathbf{z}_i)$

$$\begin{aligned}\mathcal{L}(q) &= \int \prod_i q_i \left\{ \ln p(X, \mathbf{z}) - \sum_i \ln q_i \right\} d\mathbf{z} \\ &= \int \left\{ q_j \left\{ \ln p(X, \mathbf{z}) \prod_{i \neq j} q_i d\mathbf{z}_i \right\} d\mathbf{z}_j - \int q_j \ln q_j d\mathbf{z}_j + \text{const} \right\} \\ &= \int q_j \ln \tilde{p}(X, \mathbf{z}_j) d\mathbf{z}_j - \int q_j \ln q_j d\mathbf{z}_j + \text{const} \\ &= - \int q_j \ln q_j d\mathbf{z}_j + \int q_j \ln(\tilde{p}(X, \mathbf{z}_j)/q_j) d\mathbf{z}_j \\ &= -KL(q_j || \tilde{p}(X, Z_j))\end{aligned}$$

10.1.1 Factorized Distributions

- Where we have defined a new distribution
$$\ln \tilde{p}(X, Z_j) = E_{i \neq j}[\ln p(X, Z)] + const.$$
- General optimum solution happens at $q_j(z_j) = \tilde{p}(X, z_j)$. Thus:
$$\ln q_j^*(z_j) = E_{i \neq j}[\ln p(X, Z)] + const.$$
- The additive constant in previous equations is set by normalizing the distribution $q_j^*(z_j)$. Thus if we take the exponential of both sides and normalize, we have:
$$q_j^*(z_j) = \frac{\exp(E_{i \neq j}[\ln p(X, Z)])}{\int \exp(E_{i \neq j}[\ln p(X, Z)]) dZ_j}$$
- Convergence is guaranteed because $\mathcal{L}(q)$ is convex w.r.t $q_i(Z_i)$.

10.1.2 Properties of Factorized Approximations

- Consider a Gaussian distribution $p(\mathbf{z}) = N(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$ over two correlated variables $\mathbf{z} = (z_1, z_2)$, $\boldsymbol{\mu} = [\mu_1 \mu_2]^T$, $\mathbf{z} = [z_1 z_2]^T$, $\boldsymbol{\Lambda}^{-1} = \begin{bmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{bmatrix}$.
- Only latent variables, We wish to use $q(\mathbf{z}) = q_1(z_1)q_2(z_2)$ to approximate $p(\mathbf{z})$.

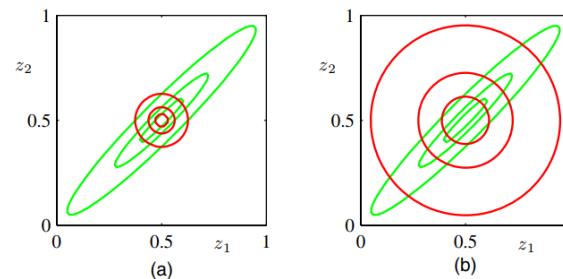
$$\begin{aligned}\ln q_1^*(z_1) &= E_{z_2}[\ln p(\mathbf{z})] + \text{const.} \\ &= E_{z_2} \left[-\frac{1}{2}(z_1 - \mu_1)^2 \Lambda_{11} - (z_1 - \mu_1) \Lambda_{12} (z_2 - \mu_2) \right] + \text{const.} \\ &= -\frac{1}{2} z_1^2 \Lambda_{11} + z_1 \mu_1 \Lambda_{11} - z_1 \Lambda_{12} (E[z_2] - \mu_2) + \text{const.}\end{aligned}$$

- Using this technique of completing square $q_1^*(z_1) = N(z_1 | m_1, \Lambda_{11}^{-1})$ where $m_1 = \mu_1 - \Lambda_{11}^{-1} \Lambda_{12} (E[z_2] - \mu_2)$
- By symmetry $q_2^*(z_2)$ is also Gaussian and can be written as $q_2^*(z_2) = N(z_2 | m_2, \Lambda_{22}^{-1})$, where $m_2 = \mu_2 - \Lambda_{22}^{-1} \Lambda_{21} (E[z_1] - \mu_1)$

10.1.2 Properties of Factorized Approximations

- By way of comparison, suppose instead that we had been minimizing the reverse KL divergence. As we shall see, this form of KL is used in an alternative approximate inference framework called *expectation propagation*.
- Let $q(\mathbf{z}) = \prod_{i=1}^M q_i(z_i)$, The KL divergence can then be written in the form $\text{KL}(p||q) = - \int p(\mathbf{z}) [\sum_{i=1}^M \ln q_i(z_i)] d\mathbf{z} + \text{const}$. We can now optimize with respect to each of the factors using Lagrange multiplier to give $q_j^*(z_j) = \int p(\mathbf{z}) \prod_{i \neq j} d\mathbf{z}_i = p(z_j)$

Figure 10.2 Comparison of the two alternative forms for the Kullback-Leibler divergence. The green contours corresponding to 1, 2, and 3 standard deviations for a correlated Gaussian distribution $p(\mathbf{z})$ over two variables z_1 and z_2 , and the red contours represent the corresponding levels for an approximating distribution $q(\mathbf{z})$ over the same variables given by the product of two independent univariate Gaussian distributions whose parameters are obtained by minimization of (a) the Kullback-Leibler divergence $\text{KL}(q||p)$, and (b) the reverse Kullback-Leibler divergence $\text{KL}(p||q)$.



10.1.2 Properties of Factorized Approximations

- Large KL when $p(Z) \rightarrow 0$ unless $q(Z) \rightarrow 0$. $KL(q||p)_{\alpha \rightarrow -1} = - \int q(Z) \ln \left\{ \frac{p(Z)}{q(Z)} \right\} dZ$, $KL(p||q)_{\alpha \rightarrow 1} = - \int p(Z) \ln \left\{ \frac{q(Z)}{p(Z)} \right\} dz$, where $p(Z)$ is nonzero.
- The two forms of KL divergence are members of the *alpha-family* of divergences defined by $D_\alpha(p||q) = \frac{4}{1-\alpha^2} (1 - \int p(x)^{\frac{1+\alpha}{2}} q(x)^{\frac{1-\alpha}{2}} dx)$. Where $-\infty < \alpha < \infty$ is a continuous parameter.

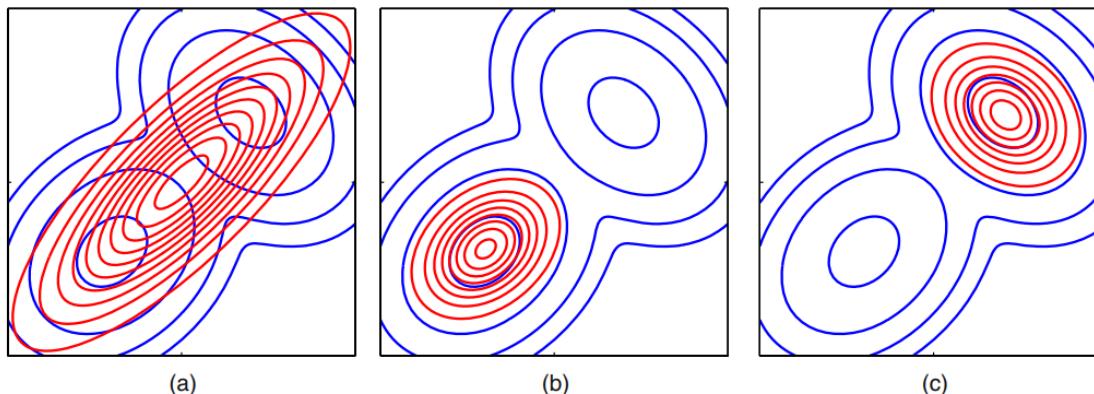


Figure 10.3 Another comparison of the two alternative forms for the Kullback-Leibler divergence. (a) The blue contours show a bimodal distribution $p(Z)$ given by a mixture of two Gaussians, and the red contours correspond to the single Gaussian distribution $q(Z)$ that best approximates $p(Z)$ in the sense of minimizing the Kullback-Leibler divergence $KL(p||q)$. (b) As in (a) but now the red contours correspond to a Gaussian distribution $q(Z)$ found by numerical minimization of the Kullback-Leibler divergence $KL(q||p)$. (c) As in (b) but showing a different local minimum of the Kullback-Leibler divergence.

10.1.3 Example: The Univariate Gaussian

- We now illustrate the factorized variational approximation using a Gaussian distribution over a single variable x .
- Given $D = \{x_1, \dots, x_N\}$ of observed values of x which are assumed to be drawn independently from the Gaussian. The likelihood function is given by $p(D|\mu, \tau) = \left(\frac{\tau}{2\pi}\right)^{\frac{N}{2}} \exp\left\{-\frac{\tau}{2} \sum_{n=1}^N (x_n - \mu)^2\right\}$

10.1.3 Example: The Univariate Gaussian

- Note that posterior distribution does not factorize in this way. The optimum $q_{\mu}(\mu)$ and $q_{\tau}(\tau)$ can be obtained from general result as follows: $\ln q_{\mu}^*(\mu) = E_{\tau}[\ln p(D|\mu, \tau) + \ln p(\mu|\tau)] + const. = -\frac{E[\tau]}{2}\{\lambda_0(\mu - \mu_0)^2 + \sum_{n=1}^N(x_n - \mu)^2\} + const.$ $q_{\mu}^*(\mu) = N(\mu|\mu_N, \lambda_N^{-1})$ where $\mu_N = \frac{\lambda_0\mu_0 + N\bar{x}}{\lambda_0 + N}$, $\lambda_N = (\lambda_0 + N)E[\tau]$.
- Also, $\ln q_{\tau}^*(\tau) = E_{\mu}[\ln p(D|\mu, \tau) + \ln p(\mu|\tau)] + \ln p(\tau) + const. = (a_0 - 1)\ln \tau - b_0\tau + \frac{N+1}{2}\ln \tau - \frac{\tau}{2}E_{\mu}[\sum_{n=1}^N(x_n - \mu)^2 + \lambda_0(\mu - \mu_0)^2] + const.$; $q_{\tau}^*(\tau) = Gam(\tau|a_N, b_N)$ where $a_N = a_0 + \frac{N+1}{2}$; $b_N = b_0 + \frac{1}{2}E_{\mu}[\sum_{n=1}^N(x_n - \mu)^2 + \lambda_0(\mu - \mu_0)^2]$

10.1.3 Example: The Univariate Gaussian

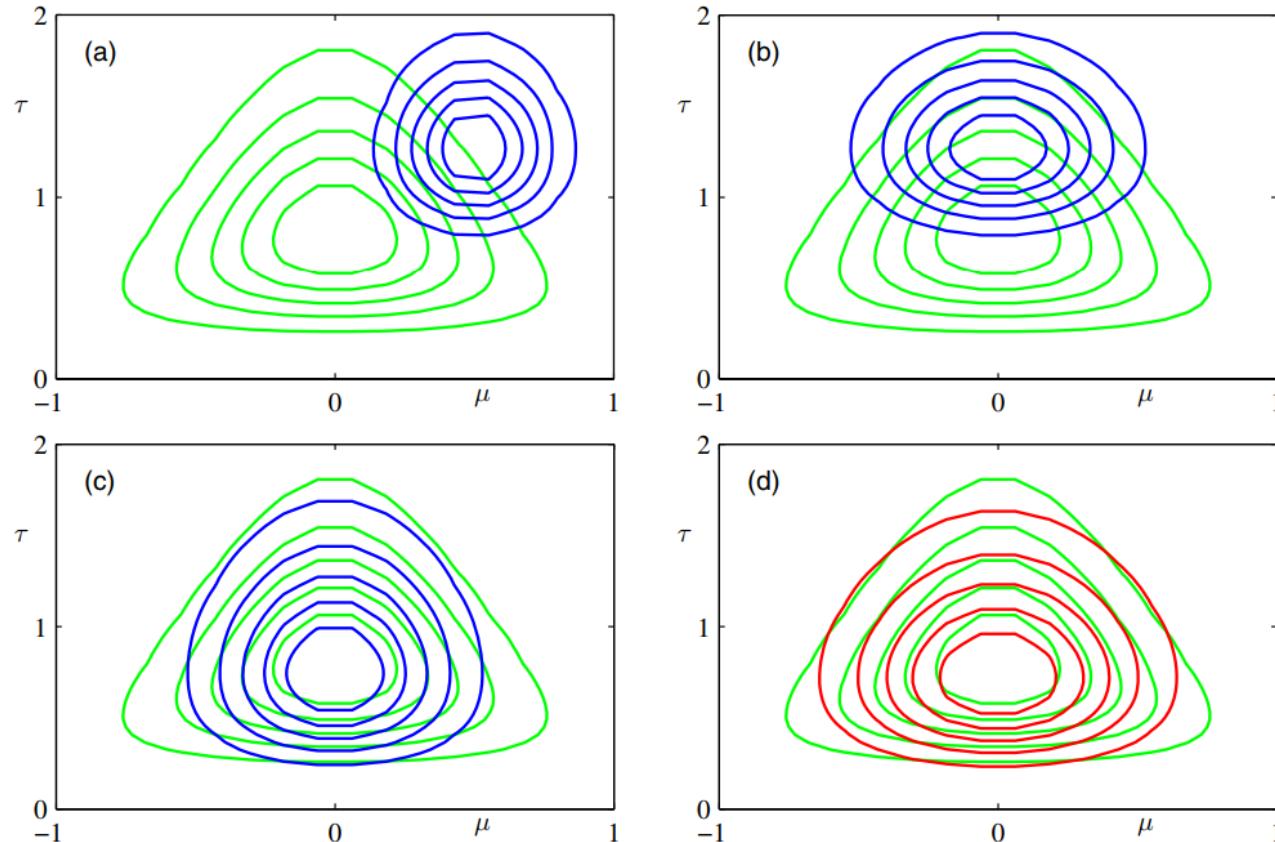


Figure 10.4 Illustration of variational inference for the mean μ and precision τ of a univariate Gaussian distribution. Contours of the true posterior distribution $p(\mu, \tau | D)$ are shown in green. (a) Contours of the initial factorized approximation $q_\mu(\mu)q_\tau(\tau)$ are shown in blue. (b) After re-estimating the factor $q_\mu(\mu)$. (c) After re-estimating the factor $q_\tau(\tau)$. (d) Contours of the optimal factorized approximation, to which the iterative scheme converges, are shown in red.

10.1.4 Model Comparison

- As well as performing inference over the hidden variables \mathbf{Z} , we may also wish to compare a set of candidate models, labelled by the index m , and having prior probabilities $p(m)$.
- Our goal is to approximate the posterior probabilities $p(m|\mathbf{X})$.
- We consider $q(\mathbf{Z}, m) = q(\mathbf{Z}|m)q(m)$ rather than $q(\mathbf{Z})q(m)$. $\ln p(\mathbf{X}) = \mathcal{L}_m - \sum_m \sum_{\mathbf{Z}} q(\mathbf{Z}|m)q(m) \ln \frac{p(\mathbf{Z}, m|\mathbf{X})}{q(\mathbf{Z}|m)q(m)}$, where lower bound is given by $\mathcal{L}_m = \sum_m \sum_{\mathbf{Z}} q(\mathbf{Z}|m)q(m) \ln \frac{p(\mathbf{Z}, m|\mathbf{X})}{q(\mathbf{Z}|m)q(m)}$; $q_m^* = \operatorname{argmax}_{q(m)} \{\mathcal{L}_m + \lambda(\sum_m q(m) - 1)\} \propto p(m) \exp\{\mathcal{L}_m\}$ → used for model selection,

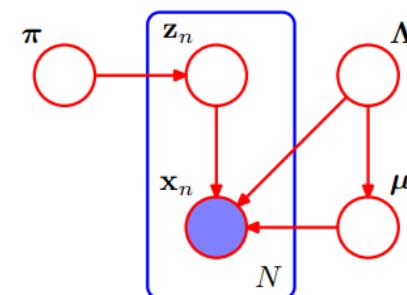
10.2. Illustration: Variational Mixture of Gaussians

- Now, we return to the discussion of Gaussian mixture model and apply the variational inference machinery developed.
- Our starting point is the likelihood function for the Gaussian mixture model.
- We can write down the conditional distribution of \mathbf{Z} , given the mixing coefficients $\boldsymbol{\pi}$, in the form $p(\mathbf{z}|\boldsymbol{\pi}) = \prod_{n=1}^N \prod_{k=1}^K \pi_k^{z_{nk}}$.
- Similarly, we can write the conditional distribution of the observed data vectors, given the latent variables and component parameters $p(\mathbf{x}|\mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = \prod_{n=1}^N \prod_{k=1}^K N(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})^{z_{nk}}$

10.2. Illustration: Variational Mixture of Gaussians

- Next, we introduce prior over the parameter μ, Λ and π . We therefore choose a Dirichlet distribution over the mixing coefficient $\pi : p(\pi) = \text{Dir}(\pi|\alpha_0) = C(\alpha_0) \prod_{k=1}^K \pi_k^{\alpha_0-1}$
- Similarly, we introduce an independent Gaussian-Wishart prior governing the mean and precision of each Gaussian component, given by $p(\mu, \Lambda) = p(\mu|\Lambda)p(\Lambda) = \prod_{k=1}^K N(\mu_k | m_0, (\beta_0 \Lambda_k)^{-1}) W(\Lambda_k | W_0, \nu_0)$

Figure 10.5 Directed acyclic graph representing the Bayesian mixture of Gaussians model, in which the box (plate) denotes a set of N i.i.d. observations. Here μ denotes $\{\mu_k\}$ and Λ denotes $\{\Lambda_k\}$.



10.2.1 Variational Distribution

- In order to formulate a variational treatment of this model, we next write down the joint distribution of all of the random variables, given by $p(\mathbf{X}, \mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda})p(\mathbf{Z}|\boldsymbol{\pi})p(\boldsymbol{\pi})p(\boldsymbol{\mu}|\boldsymbol{\Lambda})p(\boldsymbol{\Lambda})$
- From graphical model, $q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = q(\mathbf{Z})q(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) \rightarrow p(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}|\mathbf{X})$
- Let us consider the derivation of the update equation for the factor $q(\mathbf{Z})$. The log of the optimized factor is given by $\ln q^*(\mathbf{Z}) = \sum_{n=1}^N \sum_{k=1}^K z_{nk} \ln \rho_{nk} + const.$ Where $\ln \rho_{nk} = E_{\pi_k}[\ln \pi_k] + \frac{1}{2}E_{\Lambda_k}[\ln |\Lambda_k|] - \frac{D}{2}\ln 2\pi - \frac{1}{2}E_{\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k}[(\mathbf{x}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Lambda}_k (\mathbf{x}_n - \boldsymbol{\mu}_k)]$. Taking the exponential of both sides, we obtain: $q^*(z) \propto \prod_{n=1}^N \prod_{k=1}^K \rho_{nk}^{z_{nk}} \rightarrow \text{normalize} \rightarrow q^*(\mathbf{Z}) = \prod_{n=1}^N \prod_{k=1}^K r_{nk}^{z_{nk}}$ where $r_{nk} = \frac{\rho_{nk}}{\sum_{j=1}^K \rho_{nj}}$
- Finally : $\ln q^*(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = \ln p(\boldsymbol{\pi}) + \sum_{k=1}^K \ln p(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) + E_Z[\ln p(\mathbf{Z}|\boldsymbol{\pi})] + \sum_{k=1}^K \sum_{n=1}^N E[z_{nk}] \ln N(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1}) + const. = \ln q^*(\boldsymbol{\pi}) + \sum_{k=1}^K \ln q^*(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k)$