Probabilistic Modelling

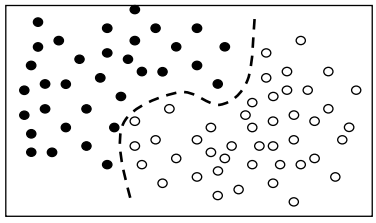
Probabilistic models for data

Just as signals can be given a probabilistic description so can classification and learning.

We will look at *supervised* and *unsupervised* problems:

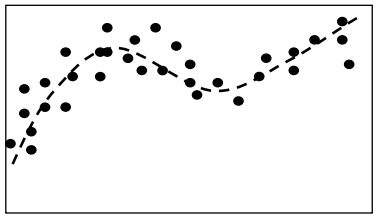
Supervised learning I: classification involves training a discriminator to identify different classes (discrete output) given input/output pairs

classification



Supervised learning II: nonlinear regression involves estimating a nonlinear functional relationship between input and output data (continuous output) given input/output pairs

nonlinear regression

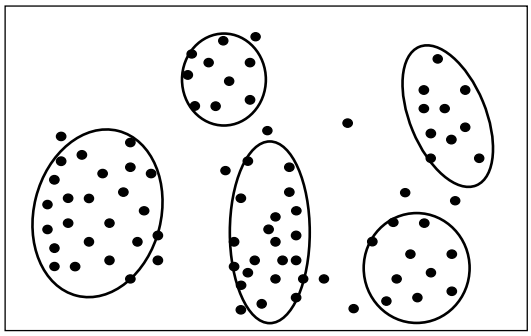


Probabilistic models for data II

clustering

Unsupervised learning: separating data into 'natural' classes or components (generally discrete) given only input data

e.g. *clustering*



Supervised learning characterised by **conditonal probability :**

$$P(\text{output} \mid \text{input})$$

unsupervised learning characterised by **marginal probability :**

$$P(\text{input}) = \sum P(input \mid class) P(class)$$

Gaussian data models

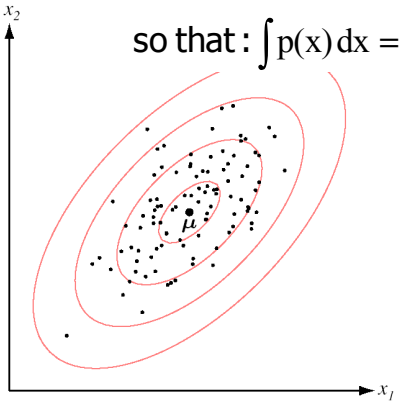
Before we look at model learning and decision theory for data we will review the most basic probabilistic model: the Gaussian distribution (*aka* Normal distribution)

$$p(\mathbf{x}) = N_x(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2} \det(\boldsymbol{\Sigma})^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right)$$

normalising factor

so that : $\int p(x)\,dx = 1$

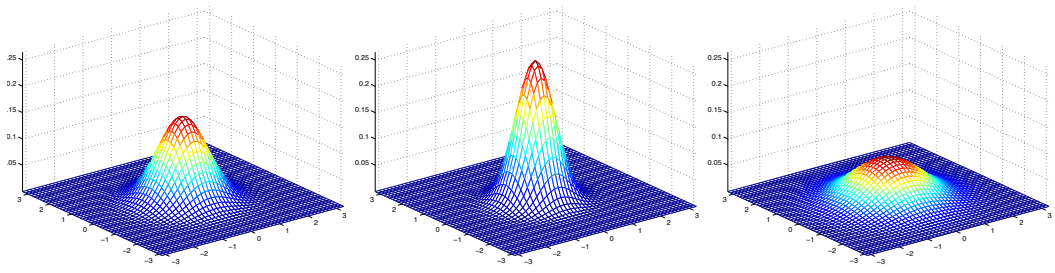
- $\boldsymbol{\mu}$ is the *d*-dimensional **mean** and
- $\boldsymbol{\Sigma}$ is the **covariance matrix**



Good model for '斑点blobs' of data.

- lines of equal probability form ellipses;
- $\boldsymbol{\mu}$ determines the centre of the 'blob';
- $\boldsymbol{\Sigma}$ determines the shape of the ellipse

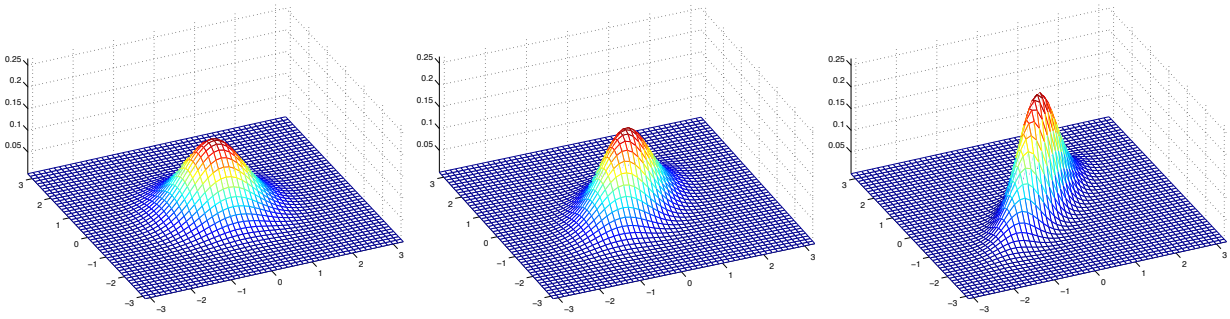
Examples



mean 0

$\Sigma = I$ $\Sigma = 0.6\, I$ $\Sigma = 2\, I$

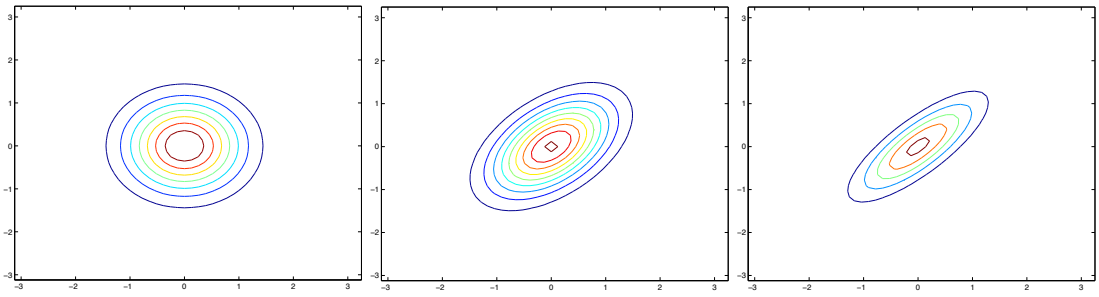
Examples



mean 0

$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}; \quad .\Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

Examples

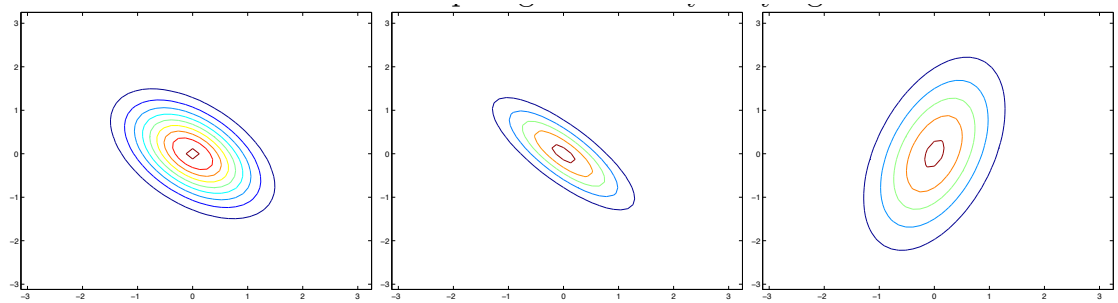


mean 0

$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}; \quad .\Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

When elements in the off diagonal are zero the variables are then statistically independent

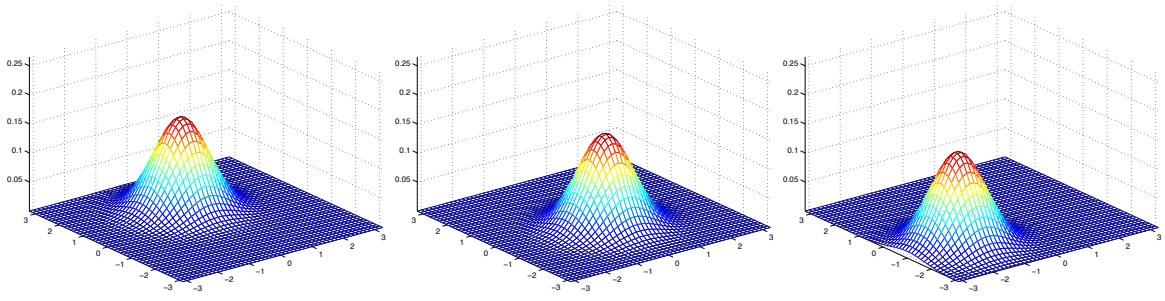
Examples



mean 0

$$\Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}; \quad .\Sigma = \begin{bmatrix} 3 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

Examples

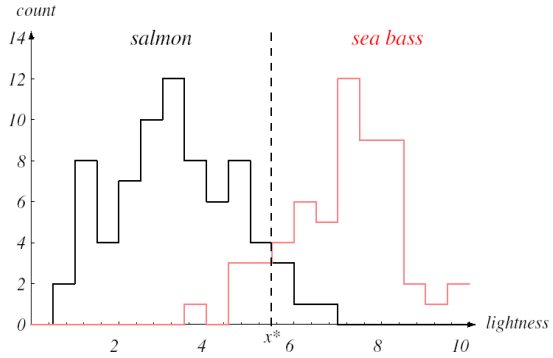


$$\mu = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \quad \mu = \begin{bmatrix} -0.5 \\ 0 \end{bmatrix}; \quad \mu = \begin{bmatrix} -1 \\ -1.5 \end{bmatrix}$$

Bayes Decision theory

Statistical models for classification

Two category classification – fish example (Duda et al)



Implies 2 probability models:

$$P(\text{lightness} < a \mid \text{fish is sea bass})$$
$$P(\text{lightness} < a \mid \text{fish is salmon})$$

Or alternatively a joint probability model:

$$P(\text{lightness}, \text{fish type})$$

(note that this is a mixed continuous/discrete probability distribution)

Bayes rule

Let c_k denote the classes for the classifier (e.g. salmon and sea bass) and let the feature be denoted by x (e.g. lightness). Bayes rule gives:

$$P(c_k \mid x) = \frac{P(x \mid c_k)P(c_k)}{P(x)} = \frac{likelihood \times prior}{evidence}$$

where:

$P(x \mid c_k)$ - **Class conditional probability (as called <u>likelihood</u>)**

$P(c_k)$ - **<u>Prior</u> probability of the class c_k occurring (without any observation)**

$P(x)$ - **'<u>evidence</u>' (the probability of observing x given our overall model)**

Given $P(c_k \mid x)$ we need a decision criterion

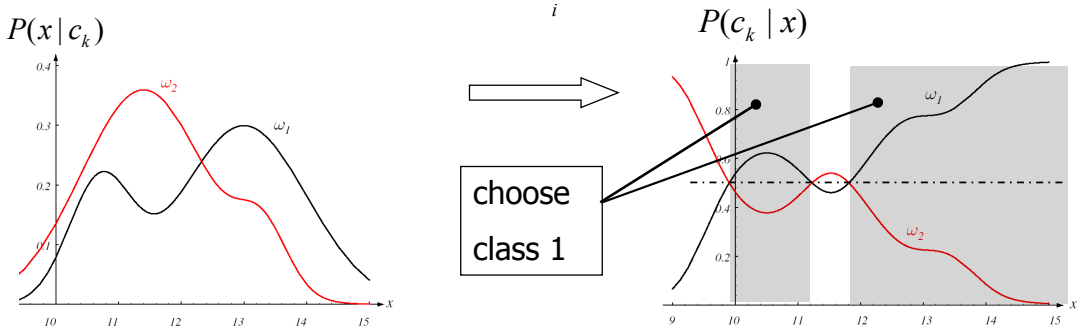
Ch 2.1 Duda

Bayes decision rule

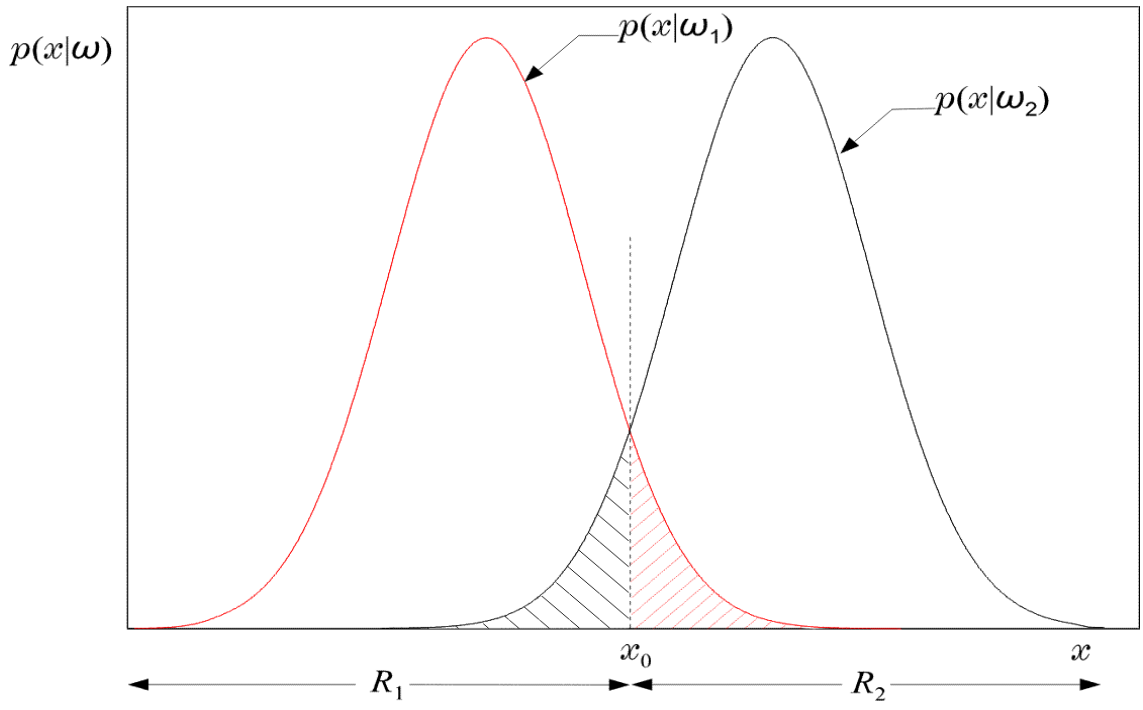
"Minimise the probability of misclassification"

Hence choose c_i if: $P(c_i \mid x) > P(c_j \mid x) \quad \forall j \neq i$, or in terms of priors and class conditional densities: $\frac{P(x \mid c_i)P(c_i)}{P(x)} > \frac{P(x \mid c_j)P(c_j)}{P(x)}$

($P(x)$ plays no role but ensures that $\sum_i P(c_i \mid x) = 1$).



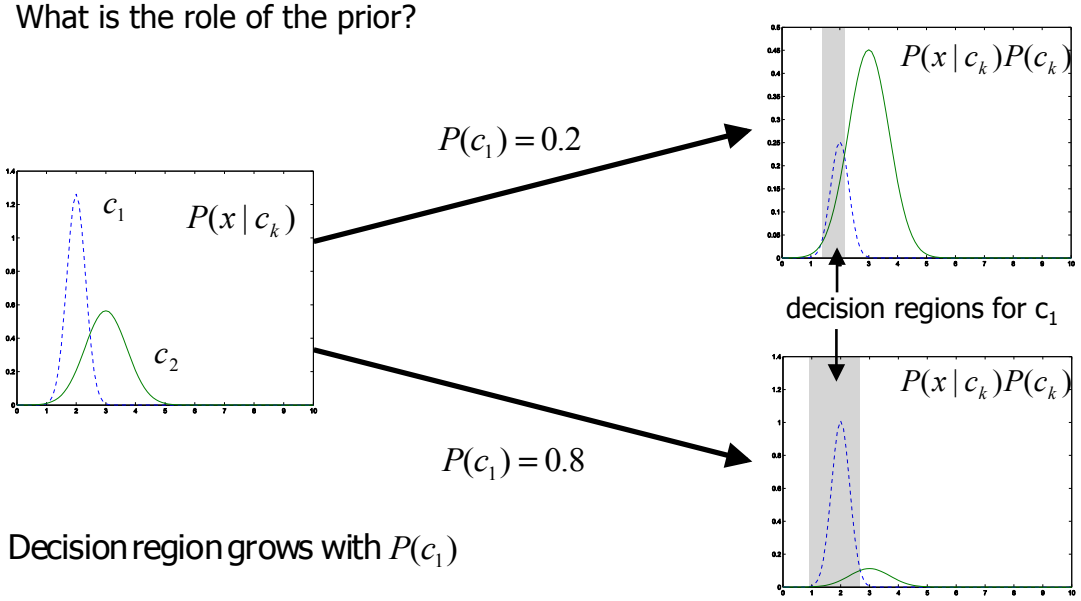
PDF of measuring a particular feature value x given the pattern is in category ω_i
Density functions are normalized, and thus the area under each curve is 1.0.



$$R_1(\rightarrow \omega_1) \text{ and } R_2(\rightarrow \omega_2)$$

The influence of the prior

What is the role of the prior?



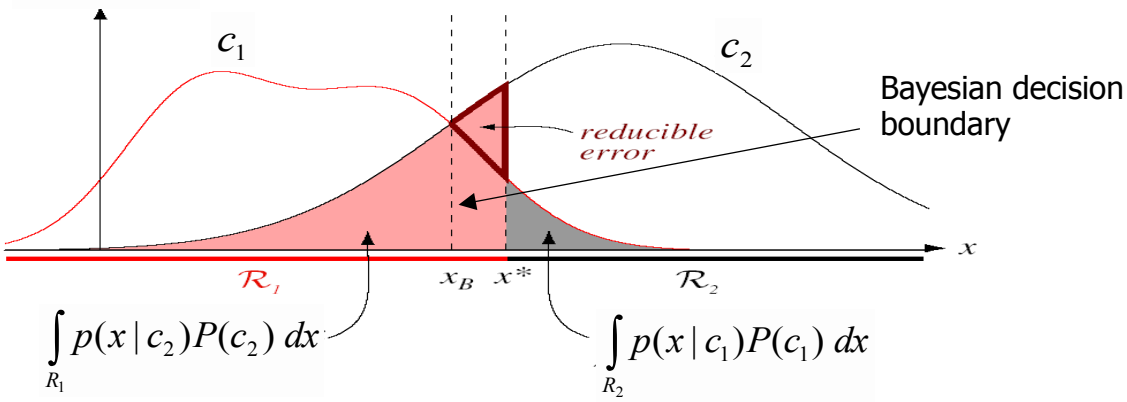
Decision region grows with $P(c_1)$

Probability of Error

Given decision regions R_k defined by (non-optimal) threshold x^* we can calculate the probability of error

$$\text{Probability of error (2 class case)} = P(x \in R_2, c_1) + P(x \in R_1, c_2)$$

$$= \int_{R_1} p(x \mid c_2)P(c_2)\,dx + \int_{R_2} p(x \mid c_1)P(c_1)\,dx$$



Ch 2.7 Duda

Discriminant functions and Neural Networks

In neural networks we introduced *discriminant functions*, $g_k(x)$. Such that $g_k(x) > g_j(x)$ for all $j \neq k$ implies choose k. In terms of Bayesian decision boundaries we have: $g_k(x) = P(c_k \mid x)$

Or in fact for any monotonic nonlinear function *f* (such as the log):

$$g_k(x) = f(P(c_k \mid x))$$

$$\begin{aligned}
\lim_{n\to\infty} \frac{1}{n} J(\mathbf{w}) &\equiv \tilde{J}(\mathbf{w}) \qquad (25)\\
&= P(\omega_k) \int [g_k(\mathbf{x};\ \mathbf{w}) - 1]^2 p(\mathbf{x}|\omega_k) d\mathbf{x} + P(\omega_{i\neq k}) \int g_k^2(\mathbf{x};\ \mathbf{w}) p(\mathbf{x}|\omega_{i\neq k}) d\mathbf{x}\\
&= \int g_k^2(\mathbf{x};\ \mathbf{w}) p(\mathbf{x}) d\mathbf{x} - 2 \int g_k(\mathbf{x};\ \mathbf{w}) p(\mathbf{x}, \omega_k) d\mathbf{x} + \int p(\mathbf{x}, \omega_k) d\mathbf{x}\\
&= \int [g_k(\mathbf{x};\ \mathbf{w}) - P(\omega_k|\mathbf{x})]^2 p(\mathbf{x}) d\mathbf{x} + \underbrace{\int P(\omega_k|\mathbf{x}) P(\omega_{i\neq k}|\mathbf{x}) p(\mathbf{x}) d\mathbf{x}}_{\text{independent of } \mathbf{w}}.
\end{aligned}$$

Discriminant functions and Neural Networks

Thus, for neural networks discriminants, $g_k(x;w)$, minimising the sum-of-squares criterion and in the infinite data limit minimises:

$$\int | g_k(x;w) - P(c_k \mid x) |^2 \ p(x) dx$$

Thus in the limit of infinite data the outputs of the trained network will approximate (in a least-squares sense) the true a posteriori probabilities, that is, the output units represent the a posteriori probabilities

$$g_k(\mathbf{x};\ \mathbf{w}) \simeq P(\omega_k|\mathbf{x}).$$

Caution in interpreting these results. Key assumption underlying the argument is that the *network can indeed represent* the functions $P(\omega_k|x)$; with insufficient hidden units, this will not be true.

Decision theory and other discriminant functions

There are situations where minimising the probability of misclassification is not appropriate:

> e.g. Medical diagnosis – it is more important to incorrectly diagnose a tumour than to miss one.

To deal with this we need to introduce a loss function R(i|j) that quantifies the *cost* of choosing c_i when the true state is c_j. We can then minimise the *expected loss:*

$$\text{decision} = \underset{i}{\operatorname{argmin}} \sum_j R(i \mid j) P(c_j \mid x)$$

Minimum probability of misclassification has a 0-1 loss function:

$$R(i \mid j) = \begin{cases} 0, \text{ if } i = j \\ 1, \text{ if } i \neq j \end{cases}$$

Examples of statistical classification I

Suppose $p(x \mid c_k)$ is Gaussian:

$$p(x \mid c_k) \propto \frac{1}{\det(\Sigma_k)^{1/2}} \exp\left(-\frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) \right)$$

where μ_k is the mean and Σ_k is the covariance.

We can choose a discriminant function of the form:

$$g_k(x) = \ln\left(p(x \mid c_k) p(c_k) \right) = \ln p(x \mid c_k) + \ln P(c_k)$$

giving:

$$g_k(x) = -\frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) - \frac{1}{2} \ln \det(\Sigma_k) + \ln P(c_k)$$

NOTE: this is only quadratic in x

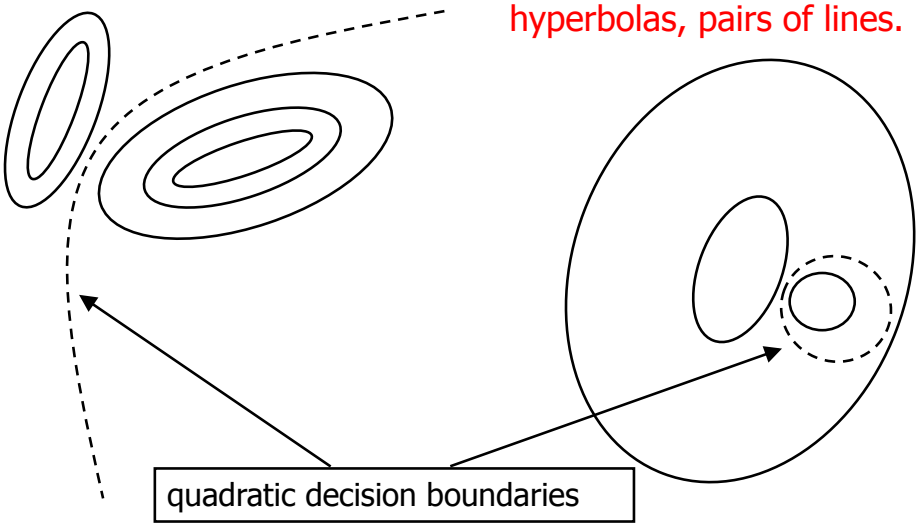
Examples of statistical classification II

General decision boundaries for Gaussian classifiers:

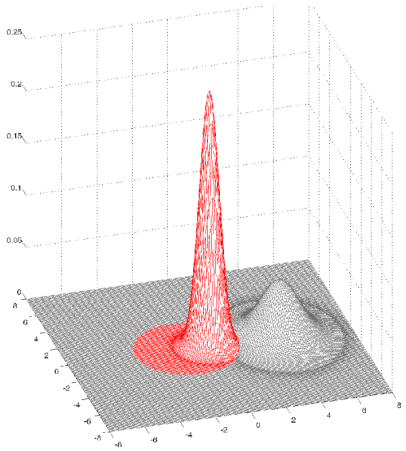
That is, $g_i(x)$ is quadratic and the surfaces

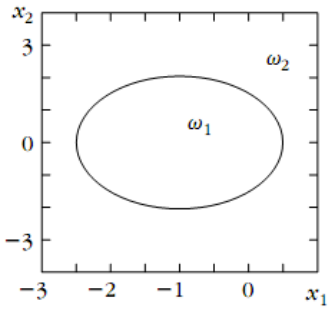
$$g_i(\underline{x}) - g_j(\underline{x}) = 0$$

quadrics, ellipsoids, parabolas, hyperbolas, pairs of lines.

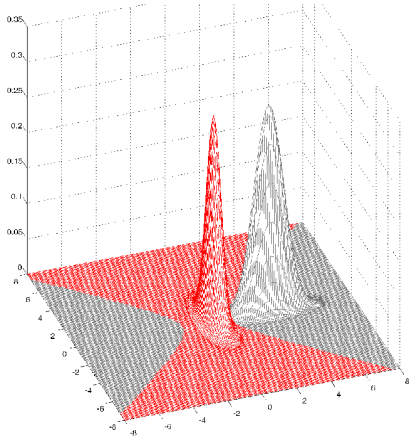


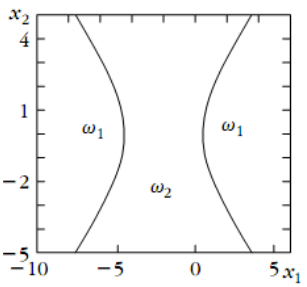
- Example 1:





- Example 2:





Examples of statistical classification III

Assume that Σ_k is common for each class. Since each $g_k(x)$ has the terms:

$$x^T \Sigma_k^{-1} x \text{ and } \frac{1}{2} \ln \det(\Sigma_k)$$

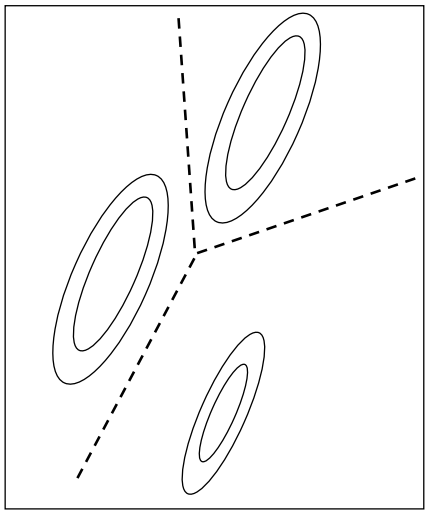
they can be dropped, giving:

$$g_k(x) = \mu_k^T \Sigma_k^{-1} x - \frac{1}{2} \mu_k^T \Sigma_k^{-1} \mu_k + \ln P(c_k)$$

which we can write as:

$$g_k(x) = \alpha^T x + \beta$$

(i.e. a linear discriminant function)

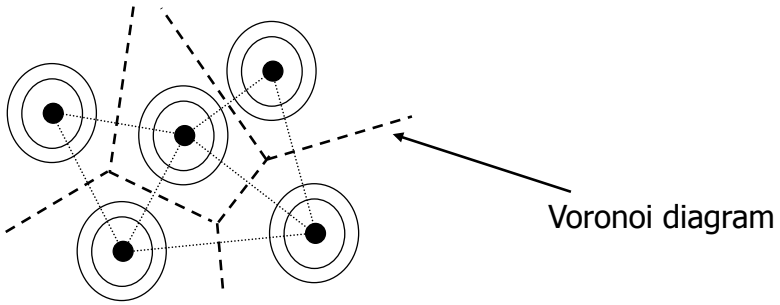


Examples of statistical classification IV

Furthermore if Σ_k is isotropic ($\Sigma_k = \sigma^2 I$) and all the priors $P(c_k)$ as assumed equal:

$$g_k(x) = -\frac{\| x - \mu_k \|^2}{2\sigma^2}$$

That is: the *means* act as *prototypes* or *templates* and the nearest template is chosen



Model Learning

Model Learning

We have looked at decision theory given a statistical model, but how do we obtain the model?

Suppose we have a *family* of probability models for $\boldsymbol{x} \sim p(\boldsymbol{x};\boldsymbol{w})$, where $\boldsymbol{w}$ represents an unknown parameter.

We will consider two new forms of learning:

- Maximum Likelihood
- Bayesian learning

Supervised and unsupervised learning are treated identically – but with different probability models.

Supervised vs Unsupervised Models

Assume the *family* of probability models:

$$p(\mathbf{x}) = g(\mathbf{x};\mathbf{w}) \qquad \text{(unsupervised)}$$
$$p(\mathbf{x} \mid \mathbf{y}) = g(\mathbf{x},\mathbf{y};\mathbf{w}) \qquad \text{(supervised)}$$

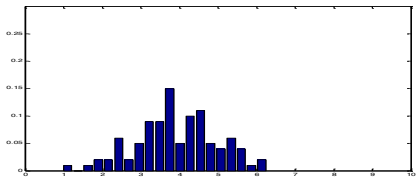
(y output data). Given a set of observation data:

$$\chi = \{\mathbf{x}_1, \mathbf{x}_2 \ldots, \mathbf{x}_N\} \qquad \text{(unsupervised)}$$
$$\chi = \{(\mathbf{x}_1,\mathbf{y}_1),(\mathbf{x}_2,\mathbf{y}_2)\ldots,(\mathbf{x}_N,\mathbf{y}_N)\} \qquad \text{(supervised)}$$

How do we choose $\mathbf{w}$ given χ?

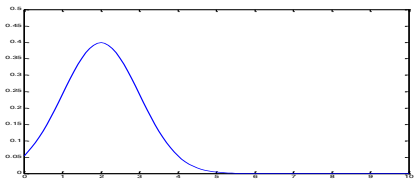
1-d density estimation example

Data histogram



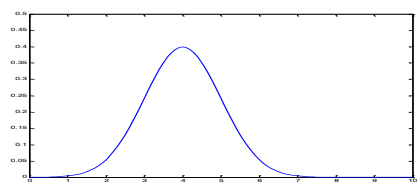
Which model is best?

Model 1



Why?

Model 2



Maximum Likelihood I

Assuming samples are independent...

$$p(\chi \mid w) = \prod_{n=1}^{N} p(\mathbf{x_n} \mid \mathbf{w}) = L(\mathbf{w})$$

$L(\mathbf{w})$ is the likelihood of the observed data given the parameter $\mathbf{w}$

Intuitively choosing $\mathbf{w}$ that makes $L(\mathbf{w})$ large is good (makes the data more likely). So ...

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\text{argmax}}\, L(\mathbf{w}) \qquad \text{(maximum likelihood)}$$

we could argue (and will) that $p(\mathbf{x} \mid \mathbf{w})$ is not the relevant quantity (we know the data exists with probability 1!)

Ch 3.2.1 Duda

Maximum Likelihood II

In practice we minimise:

$$E = -\ln L(w) = -\sum_{n} \ln p(\mathbf{x_n} \mid \mathbf{w})$$

the *negative log likelihood*.

In supervised learning E takes a form similar to an *error function.*

General ML estimates will require *numerical optimization*.

Depending on model structure we could use:

- Gradient descent (back prop.)
- Levenberg Marquardt
- Conjugate gradient

ML example: nonlinear regression I

Given data $\chi = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \ldots, (\mathbf{x}_N, y_N)\}$ we wish to fit a model of the form :

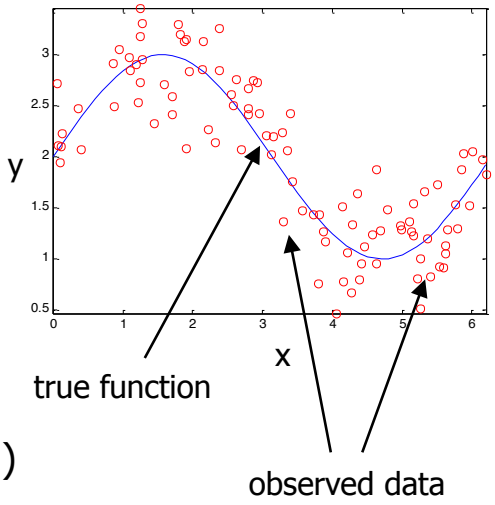
$$y = f(\mathbf{x}, \mathbf{w}) + e$$

where $f(\circ, \mathbf{w})$ is a nonlinear function and *e* an *error* with Gaussian distribution

$$p(e) = N_e(0, \sigma^2)$$

implies conditional probability for *y* is also Gaussian :

$$p(y_n \mid \mathbf{x}_n, \mathbf{w}) = N_{y|x}(f(\mathbf{x}_n, \mathbf{w}), \sigma^2)$$

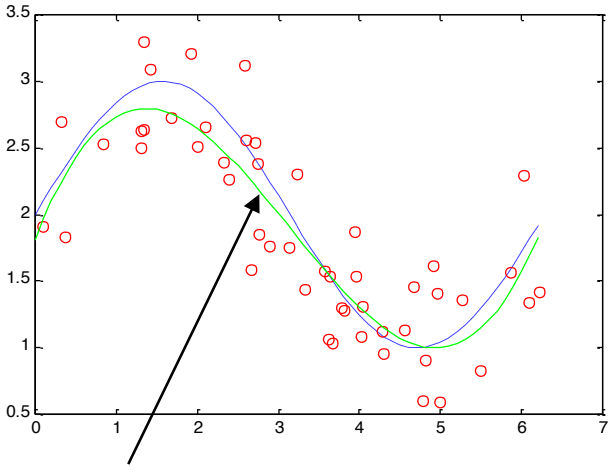


ML example: nonlinear regression II

$$E = -\ln L(\mathbf{w}) = \sum_n \frac{(y_n - f(\mathbf{x}_n, \mathbf{w}))^2}{2\sigma^2} + \text{const.} = \text{Least Squares!}$$

In general LSE ~ ML if errors are :

- independent
- Gaussian
- with common σ^2



example : ML cubic polynomial estimation

ML example II: fitting Gaussians

A simple example of *unsupervised learning* is fitting a Gaussian distribution to some data. In one dimension :

let $\boldsymbol{\chi} = \{x_1, x_2, \ldots, x_N\}$ we wish to fit a density $p(x) = N_x(\mu, \sigma^2)$

$$E = -\ln L(\mathbf{w}) = \sum_n \left(\frac{(x_n - \mu)^2}{2\sigma^2} + \ln \sigma + \frac{1}{2} \ln 2\pi \right)$$

Quadratic in μ, hence ML estimate $\hat{\mu}$:

$$\left. \frac{\partial E}{\partial \mu} \right|_{\hat{\mu}} = \sum_n \frac{(x_n - \hat{\mu})}{\sigma^2} = 0$$

$$\rightarrow \qquad \hat{\mu} = \frac{1}{N} \sum_n x_n \qquad \text{(sample mean!)}$$

ML example II: fitting Gaussians

To solve for ML estimate of σ we use the fact that we already know $\hat{\mu}$:

$$\frac{\partial E}{\partial \sigma} = \sum_n \left(\frac{1}{\sigma} - \frac{(\mathbf{x}_n - \hat{\boldsymbol{\mu}})^2}{\sigma^3} \right)$$

Equating to zero gives :

$$\sigma^2 = \frac{1}{N} \sum_n (\mathbf{x}_n - \hat{\mu})^2 \qquad \text{(sample variance)}$$

Recall that the sample variance is a *biased* estimate for variance. However such ML estimates are consistent and asymptotically optimal.

ML example II: fitting Gaussians

In higher dimensions we get the similar results of :

ML estimate for mean $\hat{\boldsymbol{\mu}}$:

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{n} \mathbf{x_n} \qquad \text{(sample mean)}$$

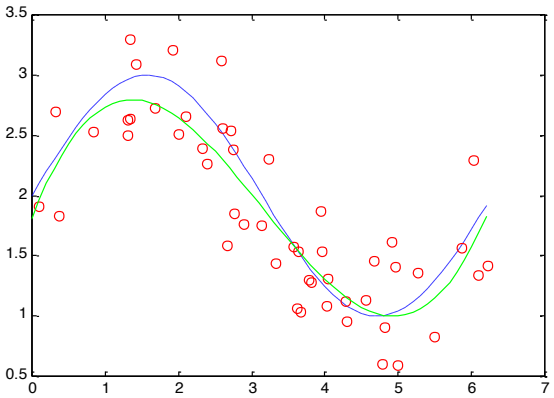
and ML estimate for covariance matrix :

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N}\sum_{n} (\mathbf{x}_n - \hat{\boldsymbol{\mu}})(\mathbf{x}_n - \hat{\boldsymbol{\mu}})^T \qquad \text{(sample covariance)}$$

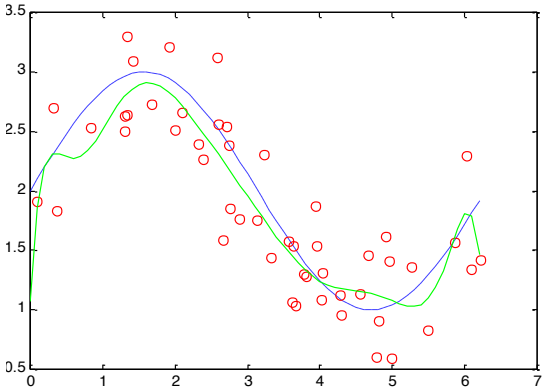
Ch 3.2.3 & 3.2.4 Duda

Overfitting in ML

Given a limited data set maximising **L(*w*)** may lead to *overfitting.* If the model order (dim. ***w***) is large enough we can 'exactly' fit model to data.



3rd order polynomial fit



10th order polynomial fit

Overfitting in ML

Overfitting relates to *Model Order Selection.* As with estimating power spectra the problem comes from a *bias -variance* trade-off. Consider the expected error between an optimal estimator $f(x)$ and the an empirical estimator $\hat{y}(x)$

$$E_x\left\{\left(f(x)-\hat{y}(x)\right)^2\right\}=E_x\left\{\left(f(x)-E_x\{\hat{y}(x)\}+E_x\{\hat{y}(x)\}-\hat{y}(x)\right)^2\right\}$$

$$=\underbrace{E_x\left\{\left(f(x)-E_x\{\hat{y}(x)\}\right)^2\right\}}_{\text{bias}^2}+\underbrace{E_x\left\{\left(E_x\{\hat{y}(x)\}-\hat{y}(x)\right)^2\right\}}_{\text{variance}}$$

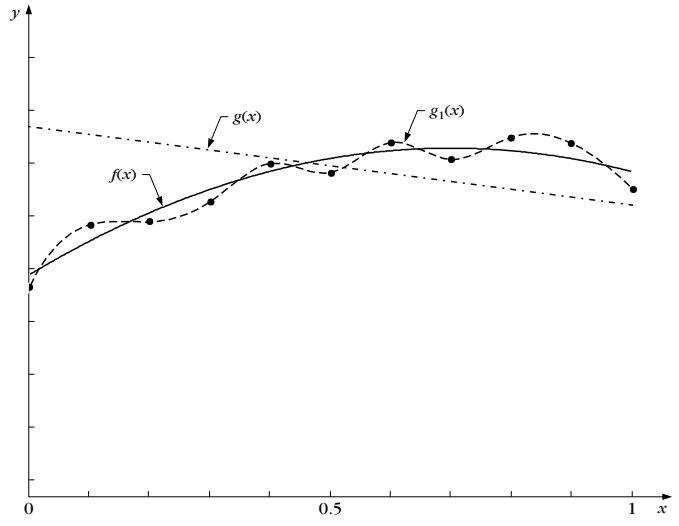
$$+\underbrace{2E_x\left\{\left(f(x)-E_x\{\hat{y}(x)\}\right)\left(E_x\{\hat{y}(x)\}-\hat{y}(x)\right)\right\}}_{=0}$$

Too complex a model → high variance. Too simple a model → high bias.

Possible solution <u>Cross-validation</u>: Break the data into **3 pieces**: *training*, *validation* and *testing* sets. Use the training data to learn the model parameters, but identify the best model order using the validation or "out-of-sample" data. Then report results on the training and testing sets.

Example

- Which one has lower bias/variance

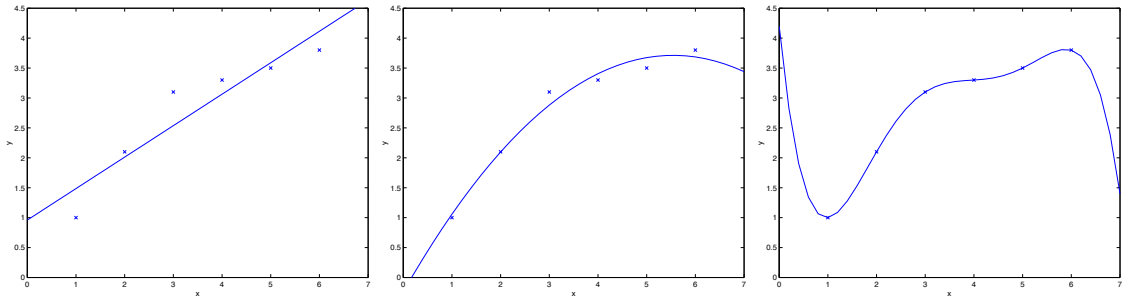


Complex model results in low-bias but a high variance, as one changes from one training set to another.

Simple model results in high bias but low variance.

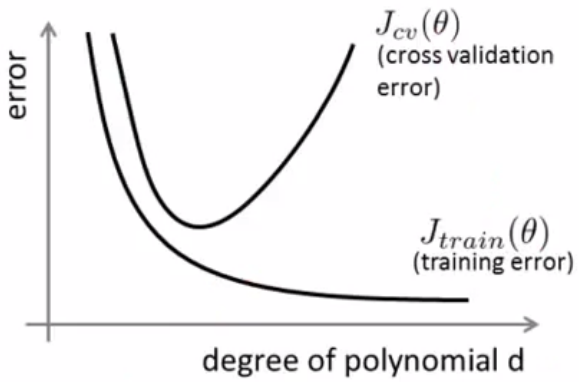
Here…

- Can you identify high bias / variance cases?



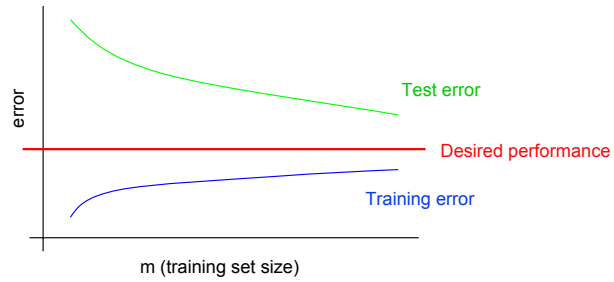
Diagnosing bias/variance: the practical picture

- Finding also a good model size



Bias vs variance

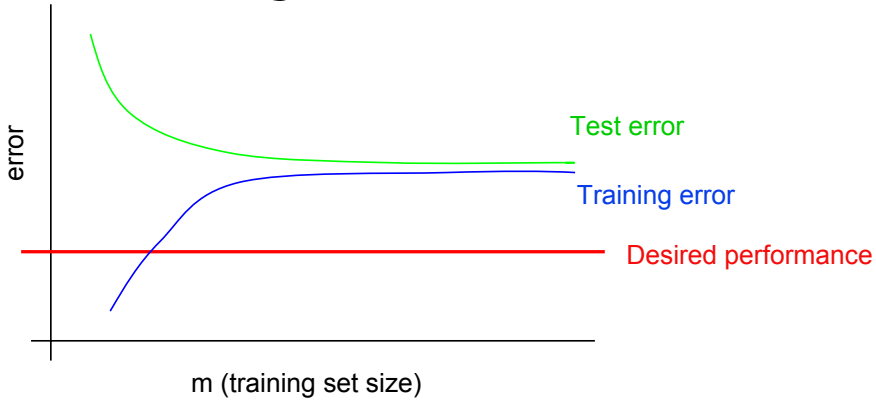
- Typical curve with high variance



- Test error decreases as m increases ➔ larger training set may help
- There is a gap between training & test error

Bias v variance

- Typical curve with high bias



- We are in trouble: training error is high too
- Small gap between the two errors

Bayesian learning I

ML generates an estimator $\mathbf{w}$ for p(x) from a family of densities p($\mathbf{x}|\mathbf{w}$) by optimizing L($\mathbf{w}$) but we consider ($\mathbf{w}$) fixed.
However the data already exists!

We are really interested in the probability of $\mathbf{w}$ given the data.
This extends the notion of random variable to the parameter $\mathbf{w}$.
==> Bayes rule gives:

$$p(\mathbf{w}|\mathbf{x}) = \frac{\mathrm{p}(\mathbf{x}|\mathbf{w})\mathrm{p}(\mathbf{w})}{p(\mathbf{x})}$$

Requires a prior distribution on $\mathbf{w}$, p($\mathbf{w}$) [which we don't know!]. Note also the role played by the likelihood, p($\mathbf{x}|\mathbf{w}$) [which we don't know!].

Bayesian learning II

We can evaluate the probability of new data

$$p(x^{(\mathrm{new})}|\chi) = \int p(x|\mathbf{w})p(\mathbf{w}|\chi)\mathrm{d}\mathbf{w}$$

Not just one value of w is used, instead a weighted average of values (this is related to the popular Kalman filter).

Bayesian Inference → Integration NOT optimization

In practice integrating and combining densities is difficult. Typically we will be happy with point estimates of $\mathbf{w}$

Modern Bayesian methods also consider using computationally intensive Monte Carlo integration.

Bayesian learning: 1-d Gaussians

Suppose $p(x \mid \mu) \sim N_x(\mu, \sigma^2)$ where σ^2 is known. We wish to estimate μ from data. Assume Gaussian prior on μ:

$$p(\mu) \sim N_\mu(\mu_0, \sigma_0{}^2)$$

(intuitively we believe that μ belongs within some range of values)

Inferring μ given some data : $\chi = \{x_1, x_2, ..., x_N\}$:

$$p(\mu \mid \chi) = \frac{p(\chi \mid \mu) p(\mu)}{p(\chi)}$$

Product of Gaussians = Gaussian (why?)

$$-\ln p(\mu \mid \chi) = \underbrace{\left[\sum_n \frac{(x_n - \mu)^2}{2\sigma^2}\right]}_{-\text{loglikelihood}} + \underbrace{\left[\frac{(\mu - \mu_0)^2}{2\sigma_0{}^2}\right]}_{-\text{logprior}} + \textit{const.}$$

Still quadratic in μ, hence Gaussian : $p(\mu \mid \chi) = N_\mu(\mu_N, \sigma_N{}^2)$

Bayesian learning: 1-d Gaussians

Equating the quadratic terms, μ^2 :

$$\frac{\mu^2}{2\sigma_N^2} = \frac{N\mu^2}{2\sigma^2} + \frac{\mu^2}{2\sigma_0^2}$$

giving

$$\sigma_N^2 = \left(\frac{N}{\sigma^2} + \frac{1}{\sigma_0^2}\right)^{-1}$$

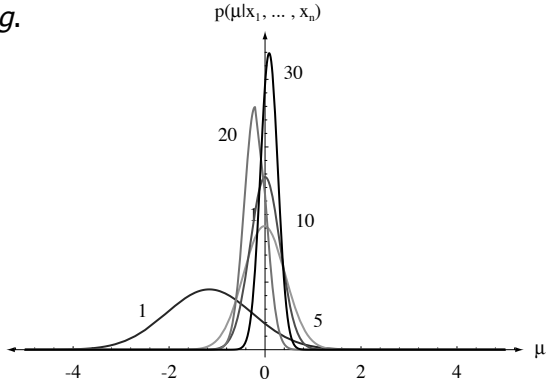
and the linear terms, μ :

$$\frac{2\mu_N \mu}{2\sigma_N^2} = \frac{2\mu_0 \mu}{2\sigma_0^2} + \frac{2}{2\sigma^2}\sum_n x_n \mu$$

giving :

$$\mu_N = \frac{N\sigma_0^2}{N\sigma_0^2 + \sigma^2}\bar{x} + \frac{\sigma^2 \mu_0}{N\sigma_0^2 + \sigma^2}$$

These equations show how the prior information is combined with the empirical information in the samples to obtain the a posteriori density. Roughly speaking, μ_N represents our best guess for μ after observing N samples, and σ^2_N measures our uncertainty about this guess. σ^2_N decreases monotonically with N (approaches σ^2/N as N goes to infinity) each additional observation decreases our uncertainty about the true value of μ. N increases, $p(\mu|x)$ becomes more and more sharply peaked. This behavior is commonly known as *Bayesian learning*.



Bayesian choice of *w*

Often we may wish to get a single value for **w** as an estimate from some data. From a Bayesian perspective this involves defining a loss function (c.f. Bayesian decision theory). 2 common choices are:

1. Posterior Mean:

$$\mathbf{w} = E\{\mathbf{w} \mid \boldsymbol{\chi}\} = \int \mathbf{w}\, \mathrm{p}(\mathbf{w} \mid \boldsymbol{\chi})\, \mathrm{d}\,\mathbf{w}$$

this comes from a Mean Squared Error loss function.

2. Maximum a Posteriori (MAP) estimates:

$$\mathbf{w} = \underset{\mathbf{w}}{\mathrm{argmax}}\left[p(\mathbf{w} \mid \boldsymbol{\chi})\right]$$

closely related to ML estimation – not really *very* Bayesian (comes from a 0-1 loss function)

The Bayes - ML link

We can see that ML and Bayesian learning are linked through the likelihood :

$$\mathrm{p}(\mathbf{w} \mid \boldsymbol{\chi}) \propto L(\mathbf{w})\, \mathrm{p}(\mathbf{w})$$

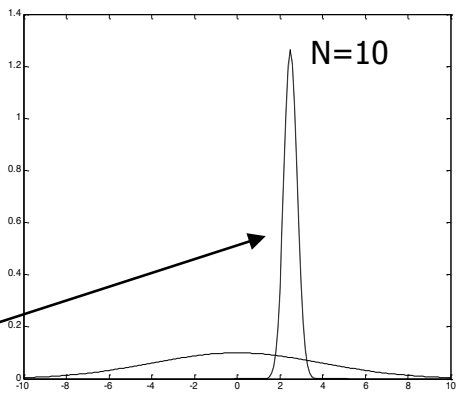
If prior is very 'weak' (e.g. $\sigma_0^2 >> 1$) then likelihood dominates.

Typically as $\mathrm{N} \to \infty$

$$\mathrm{p}(\mathbf{w} \mid \boldsymbol{\chi}) \to L(\mathbf{w})$$

Also, as $\mathrm{N} \to \infty$, $L(\mathbf{w})$ becomes very narrow (variance $\to 0$), so the maximum value characterises the distribution well.

Likelihood ≈ posterior

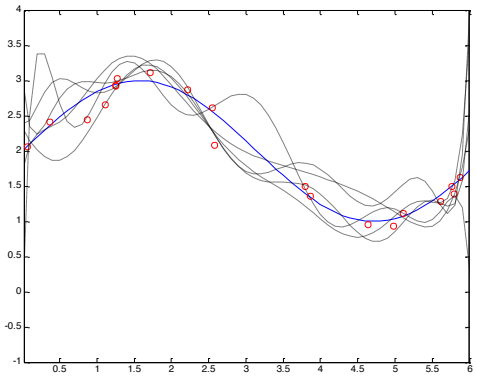


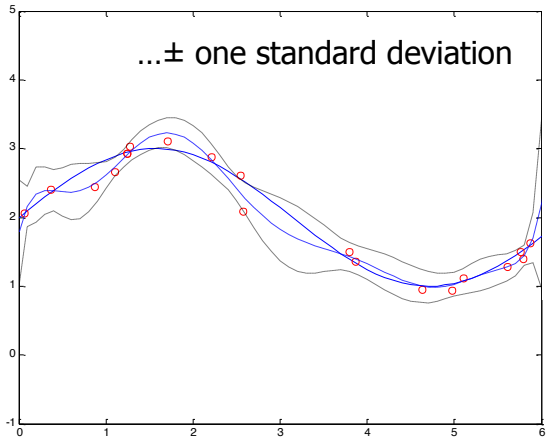
Ch 3.5.1 Duda

Exploring the Posterior Distribution

Recall the Bayesian solution gives us a full posterior distribution this is related to the bias variance trade-off.

Let's looks at some samples from the posterior (dotted) …





Averaging over different samples allows us to estimate error bars (dotted) around our estimator (left).