LECTURE 5: PARAMETER ESTIMATION

There are a lot of standard texts and courses in optimisation theory. This chapter will cover only a subset of the latter. It covers, however, the following:

- The EM algorithm - application to (e.g.) Gaussian mixture models.
- Discussion regarding the Hessian matrix & its importance in optimisation.
- Gradient (steepest) descent - why is it poor?
- Newton & quasi-Newton methods.
- The Bayesian approach - why is it so different?

The EM algorithm

We consider the case of a mixture of K Gaussians and write the likelihood of the n–th data sample on the k–th Gaussian as

$$P(\mathbf{x}^n \mid k) = \frac{1}{(2\pi)^{p/2}|\mathbf{F}_k|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}^n - \boldsymbol{\mu}_k)^T \mathbf{F}_k^{-1}(\mathbf{x}^n - \boldsymbol{\mu}_k)\right]$$

via Bayes' mixture theorem

$$P(\mathbf{x}^{(n)}) = \sum_{k=1}^{K} P(\mathbf{x}^{(n)} \mid k) P(k)$$

and the total evidence of the data set is

$$P(data) = \prod_{n=1}^{N} P(\mathbf{x}^{(n)})$$

we may take logarithms (monotone function)

$$L = \ln P(data) = \sum_{n=1}^{N} \ln P(\mathbf{x}^{(n)})$$

it is this quantity that is most convenient to maximise. Each Gaussian has three sets of free parameters; $\boldsymbol{\mu}_k, \mathbf{F}_k$ and the prior $P(k)$. Maximising L must be performed under the constraint that $\sum P(k) = 1$.

Taking partial derivatives of L w.r.t. $\boldsymbol{\mu}_k, \mathbf{F}_k$ gives rise to the following:

$$\hat{\boldsymbol{\mu}}_k = \frac{\sum_n P(k \mid \mathbf{x}^{(n)}) \mathbf{x}^{(n)}}{\sum_n P(k \mid \mathbf{x}^{(n)})}$$

$$\hat{\mathbf{F}}_k = \frac{\sum_n P(k \mid \mathbf{x}^{(n)})(\mathbf{x}^{(n)} - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}^{(n)} - \hat{\boldsymbol{\mu}}_k)^T}{\sum_n P(k \mid \mathbf{x}^{(n)})}$$

For the kernel priors, $P(k)$, there is a constraint that $\sum_{k'} P(k') = 1$. We can enforce this constraint by using Lagrange multipliers or auxiliary variables in a softmax function. Either way we obtain

$$\hat{P}(k) = \frac{1}{N} \sum_n P(k \mid \mathbf{x}^{(n)})$$

These represent a set of three coupled non-linear equations. It has been shown that a solution may be obtained via the *expectation- maximisation* (EM) algorithm. This is a successive re-estimation technique.

Maximising L is equivalent to minimising an error or energy function,

$$E = -\ln P(data) = -\sum_{n=1}^{N} \ln P(\mathbf{x}^{(n)})$$

Consider a change in the parameters, such that $E_{old} \rightarrow E_{new}$.

$$E_{new} - E_{old} = -\sum_n \ln \left\{ \frac{\sum_k P_{new}(k) P_{new}(\mathbf{x}^n \mid k)}{\sum_k P_{old}(k) P_{old}(\mathbf{x}^n \mid k)} \right\}$$

by Bayes so

$$E_{new} - E_{old} = -\sum_n \ln \left\{ \frac{\sum_k P_{new}(k) P_{new}(\mathbf{x}^n \mid k)}{P_{old}(\mathbf{x}^n) P_{old}(k \mid \mathbf{x}^n)} P_{old}(k \mid \mathbf{x}^n) \right\}$$

Now we use Jensen's inequality, which states that, if $\lambda_k \geq 0$ and $\sum_k \lambda_k = 1$, then:

$$\ln \left(\sum_k \lambda_k v_k \right) \geq \sum_k \lambda_k \ln v_k$$

for any variable set v_k. Applying that to the above gives

$$E_{new} \leq E_{old} - \sum_n \sum_k P_{old}(k \mid \mathbf{x}^n) \ln \left\{ \frac{P_{new}(k) P_{new}(\mathbf{x}^n \mid k)}{P_{old}(\mathbf{x}^n) P_{old}(k \mid \mathbf{x}^n)} \right\}$$

which is of the form,

$$E_{new} \leq E_{old} + Q$$

If we get rid of all the terms which won't change (old ones) then we can just minimise a functional Q' which is normally referred to as the *auxilliary equation*:

$$Q' = -\sum_n \sum_k P_{old}(k \mid \mathbf{x}^n) \ln \{ P_{new}(k) P_{new}(\mathbf{x}^n \mid k) \}$$

If we take derivatives of Q' w.r.t. $\boldsymbol{\mu}$, $\mathbf{F}$ and P_{new} (remembering that we need them to sum to one, so we must use dummy variables in a softmax function or Lagrange multipliers) then we end up with the same Equations as before, but with with the correct new and old parameter sets. This is summarised as follows.

loop
    use $P_{old}, \boldsymbol{\mu}_{old}, \mathbf{F}_{old}$ to calculate $P_{old}(k \mid \mathbf{x})$
    use $P_{old}(k \mid \mathbf{x})$ to re-estimate a new set of $\boldsymbol{\mu}_{new}$
    use $P_{old}(k \mid \mathbf{x}), \boldsymbol{\mu}_{new}$ to re-estimate a new set of $\mathbf{F}_{new}$
    use $P_{old}(k \mid \mathbf{x}), \boldsymbol{\mu}_{new}, \mathbf{F}_{new}$ to re-estimate P_{new}.
    let $old = new$
until parameters change no more (or less than some small amount).

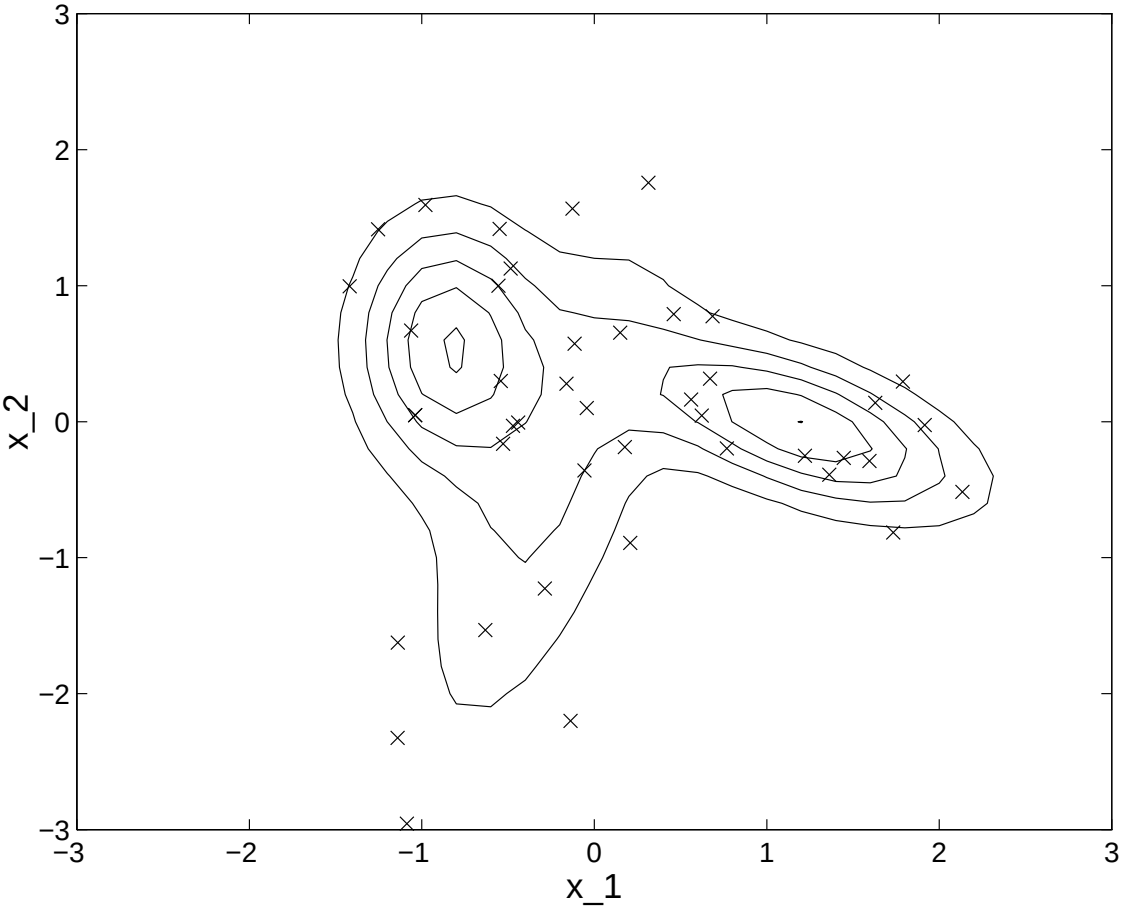

Figure 1: Three Gaussian mixture fit to data using EM.

The Hessian matrix

Consider an error surface, shown in 1-D in Figure 2, as a function of the parameters, or weights, of the model.

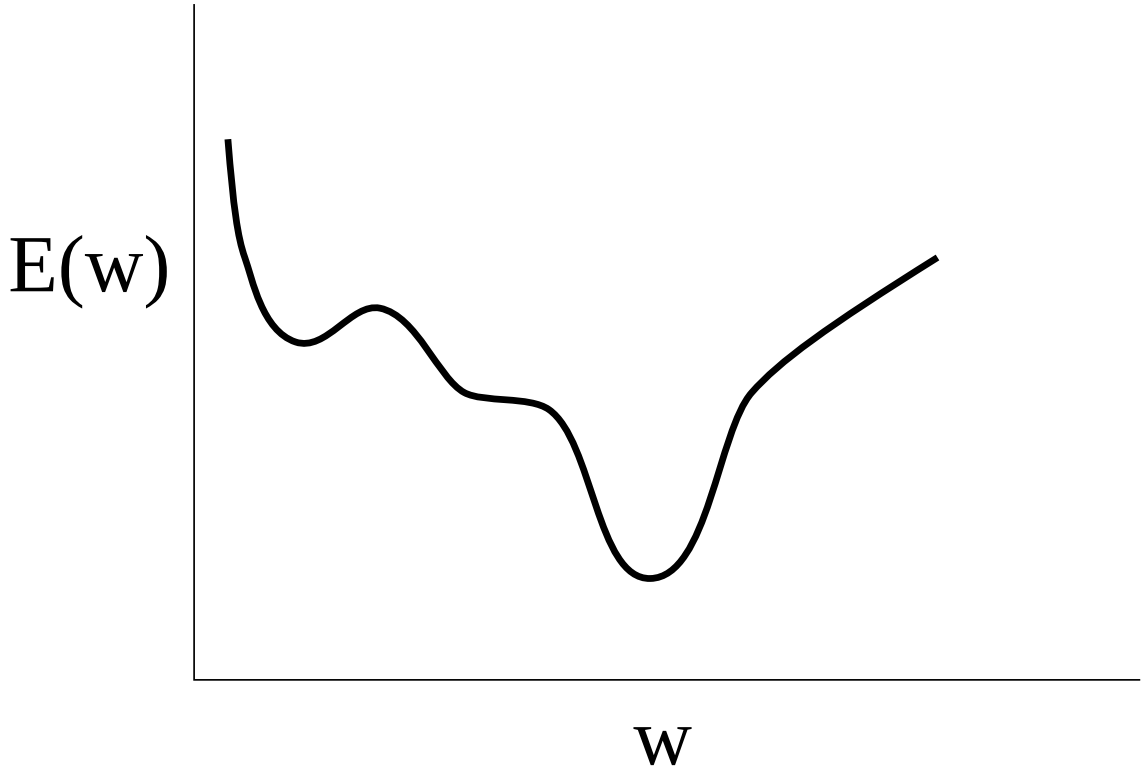

Figure 2: Error surface.

 It is worth making sure you understand the concepts of:

- global minimum,
- local minima,
- saddle & inflexion points.

Consider a local quadratic approximation to the error minimum (Fig. 3). via a 2nd order Taylor series around some point $\mathbf{w}^*$

$$E(\mathbf{w}) = E(\mathbf{w}^*) + (\mathbf{w} - \mathbf{w}^*)^\mathsf{T} \nabla E|_{\mathbf{w}^*} + \frac{1}{2!}(\mathbf{w} - \mathbf{w}^*)^\mathsf{T}\mathbf{H}(\mathbf{w} - \mathbf{w}^*)$$

in which $\mathbf{H}$ is the Hessian matrix,

$$H_{ij} = \left.\frac{\partial^2 E}{\partial w_i \partial w_j}\right|_{\mathbf{w}^*}$$

Note also that

$$\nabla E(\mathbf{w}) \approx \nabla E(\mathbf{w}^*) + \mathbf{H}(\mathbf{w} - \mathbf{w}^*)$$

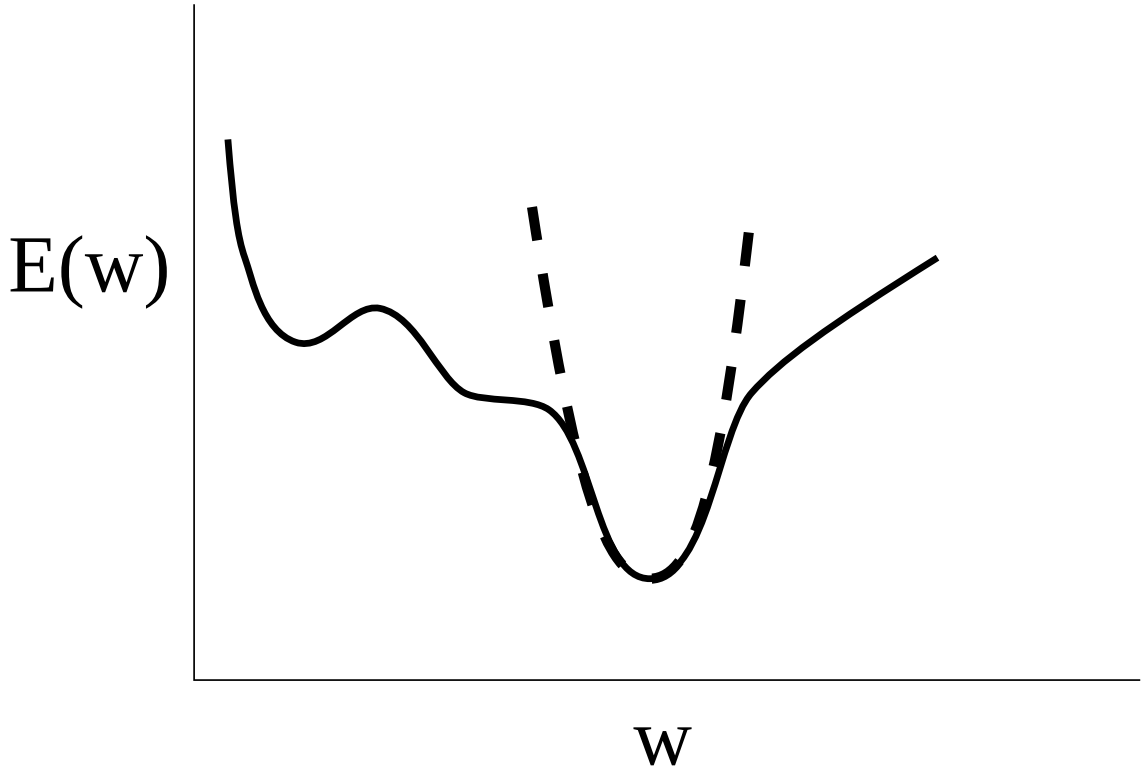

Figure 3: Quadratic approximation.

If $\mathbf{w}^*$ is a minimum point then the gradient is zero and

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

An eigendecomposition of $\mathbf{H}$ is insightful.

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

so the eigenvalues represent the scales of the curvature along the principle axes of the local quadratic error function.

Gradient (steepest) descent

The gradient descent algorithm is the simplest of all optimisation routines. It changes the parameters so that the new values lie some distance down the steepest error gradient from the the old values, i.e.

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \, \nabla E|_{\mathbf{w}(t)}$$

where η is the *adaption rate*, $0 < \eta < 1$.

- ∇E can be estimated on a block of data,
- or on a sample-by-sample basis. The latter is *stochastic* gradient descent.

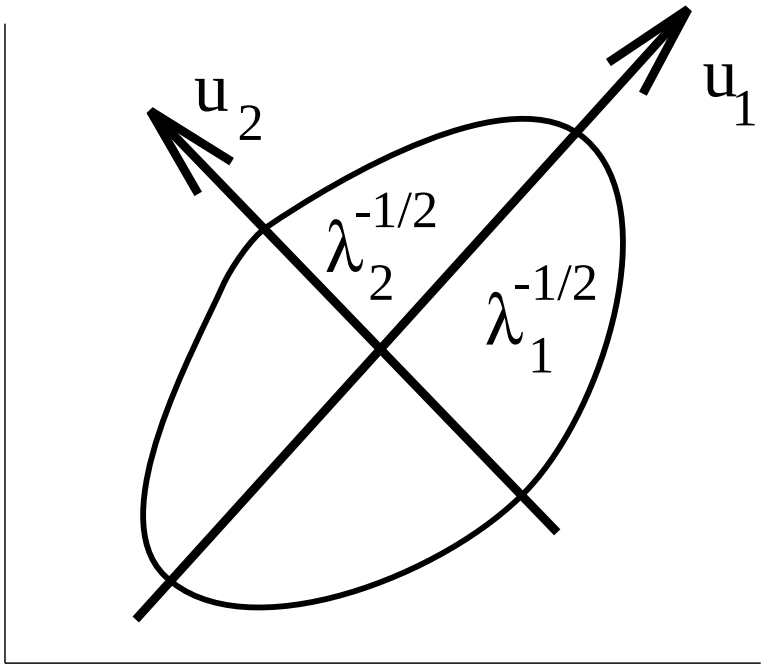

Figure 4: Hessian eigen-vectors.

Convergence

Look at the quadratic approximation. We can write the gradient as a combination of the Hessian eigenvectors,

$$\nabla E = \sum_i \alpha_i \lambda_i \mathbf{u}_i$$

writing

$$\Delta \mathbf{w} = \sum_i \Delta \alpha_i \mathbf{u}_i$$

so

$$\Delta \alpha_i = -\eta \lambda_i \alpha_i$$

hence

$$\alpha_i^{new} = (1 - \eta \lambda_i) \alpha_i^{old}$$

and

$$\alpha_i = \mathbf{u}_i^{\mathsf{T}} (\mathbf{w} - \mathbf{w}^*)$$

For $\Delta \mathbf{w}$ to converge so

$$|1 - \eta \lambda_i| < 1$$

as the eigenvalues are ordered from λ_{max} to λ_{min} hence this can be met by having

$$\eta < \frac{2}{\lambda_{max}}$$

Note that the convergence in each principle direction of the Hessian is governed by the corresponding eigenvalue. This means that for a poorly conditioned Hessian,

giving a long thin valley, most of the effort will be spent going up and down the steep valley walls in one direction and very little in getting to the minimum along the less steep direction.

- Gradient descent is a very poor algorithm!
- Attempts to make is better, such as adding 'momentum' are not very sucessful.
- It does have the advantage of being able to be stochastic, and hence used 'on-line'.

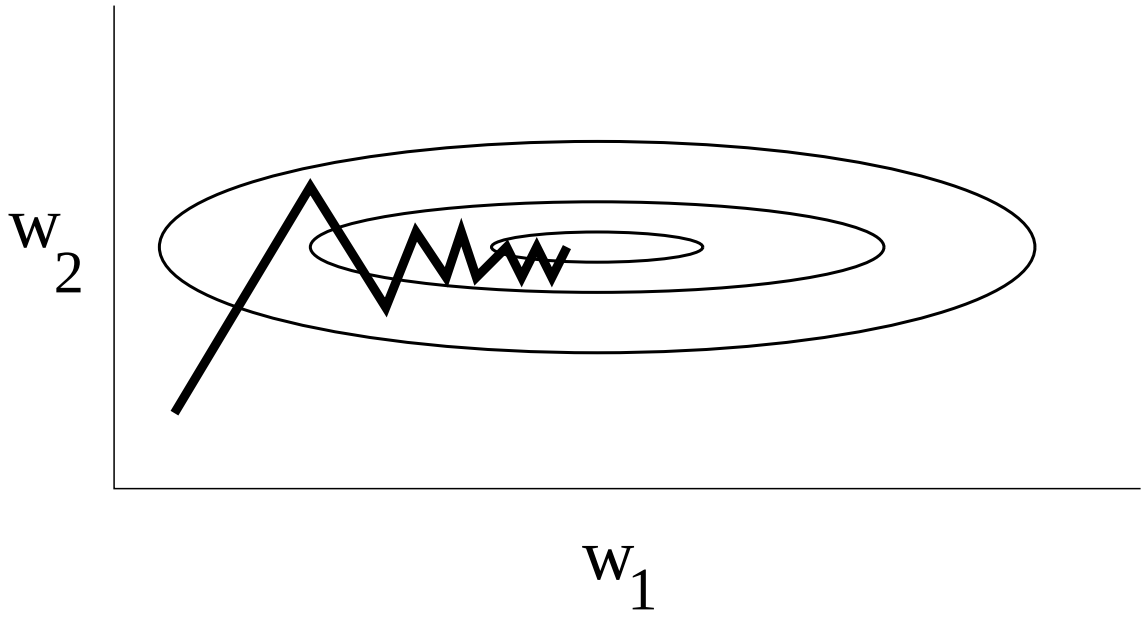

Figure 5: Poor convergence in a thin valley.

Newton & quasi-Newton methods

Consider the gradient $\mathbf{g} = \nabla E(\mathbf{w})$ as being estimated via the quadratic function,

$$\mathbf{g}(\mathbf{w}) = \mathbf{H}(\mathbf{w} - \mathbf{w}^*)$$

where $\mathbf{w}^*$ is at the minimum. Hence,

$$\mathbf{w}^* = \mathbf{w} \underbrace{-\mathbf{H}^{-1}\mathbf{g}}_{\text{Newton step}} \qquad \text{compare this to } \mathbf{w}(t+1) = \mathbf{w}(t) - \eta\mathbf{g}$$

This, unlike, gradient descent, will get to the minimum of any locally quadratic function in a single step! Note that the Newton step vector always points from $\mathbf{w}$ to the minimum, not just down the line of steepest gradient. We could use this, but

- If we are not in an area of the error surface which is not really locally quadratic then the step we make will not get to a minimum (but would get us closer), and

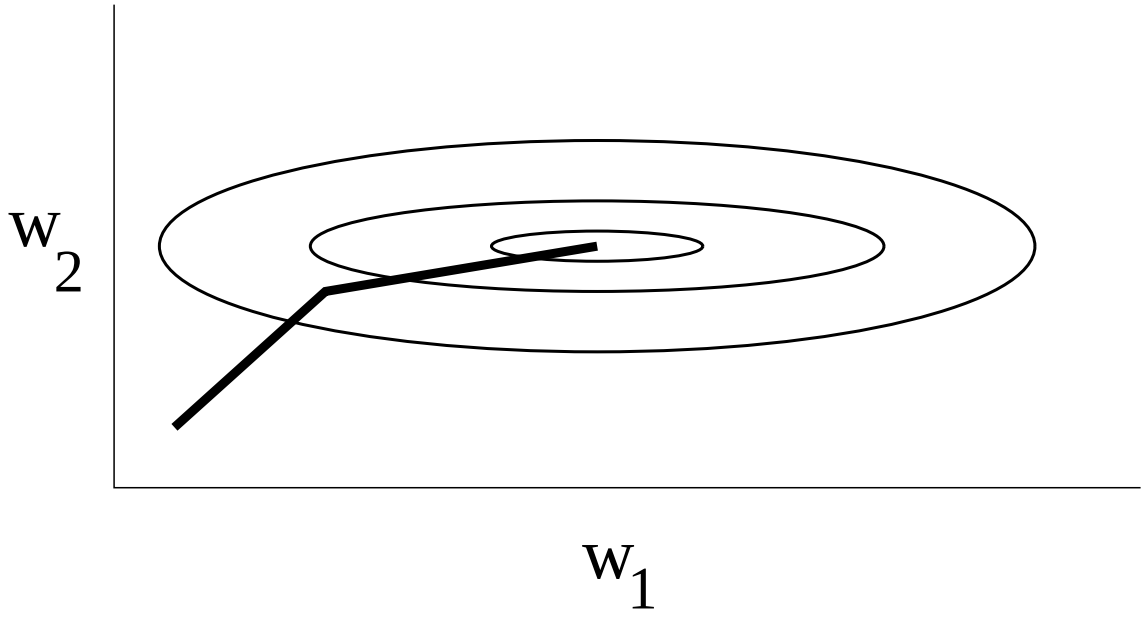

Figure 6: Poor convergence in a thin valley - improved using Newton!

- we would have to re-estimate the Hessian and invert it every step we made. This is $\mathcal{O}(NW^2) + \mathcal{O}(W^3)$ - so horrible.

Quasi-Newton methods

From the Newton equation we have

$$\mathbf{w}(t+1) - \mathbf{w}(t) = -\mathbf{G}(\mathbf{g}(t+1) - \mathbf{g}(t))$$

where $\mathbf{G} = \mathbf{H}^{-1}$. The most widely used method which re-estimates the inverse Hessian is the *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) method. The derivation of this is out of place here but can be found in texts on optimisation (the nomenclature I use is from Bishop's book - but he does not derive it either!)

$$\mathbf{G}(t+1) = \mathbf{G}(t) + \frac{\mathbf{p}\mathbf{p}^\mathsf{T}}{\mathbf{p}^\mathsf{T}\mathbf{v}} - \frac{(\mathbf{G}(t)\mathbf{v})\mathbf{v}^\mathsf{T}\mathbf{G}(t)}{\mathbf{v}^\mathsf{T}\mathbf{G}(t)\mathbf{v}} + (\mathbf{v}^\mathsf{T}\mathbf{G}(t)\mathbf{v})\mathbf{u}\mathbf{u}^\mathsf{T}$$

where

$$\mathbf{p} = \mathbf{w}(t+1) - \mathbf{w}(t)$$

$$\mathbf{v} = \mathbf{g}(t+1) - \mathbf{g}(t)$$

$$\mathbf{u} = \frac{\mathbf{p}}{\mathbf{p}^\mathsf{T}\mathbf{v}} - \frac{\mathbf{G}(t)\mathbf{v}}{\mathbf{v}^\mathsf{T}\mathbf{G}(t)\mathbf{v}}$$

To implement the quasi-Newton approach we need to use the following update equation:

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \alpha(t)\mathbf{G}(t)\mathbf{g}(t)$$

As already said, the Newton step, $-\mathbf{Gg}$, points us in the right direction. How do we choose how far to go each time, i.e. the parameter $\alpha(t)$? The answer is to use a line-search method.

The line search

We know which line (direction) to go. We have two choices,

- Get exactly to the minimum error along the line (exact line search), or
- use approximate line-search which just gets us close (enough).

The former is computationally prohibitive, but can be achieved by interval methods (see Fig. 7). The latter is often achieved by fitting a local parabolic (or other simple

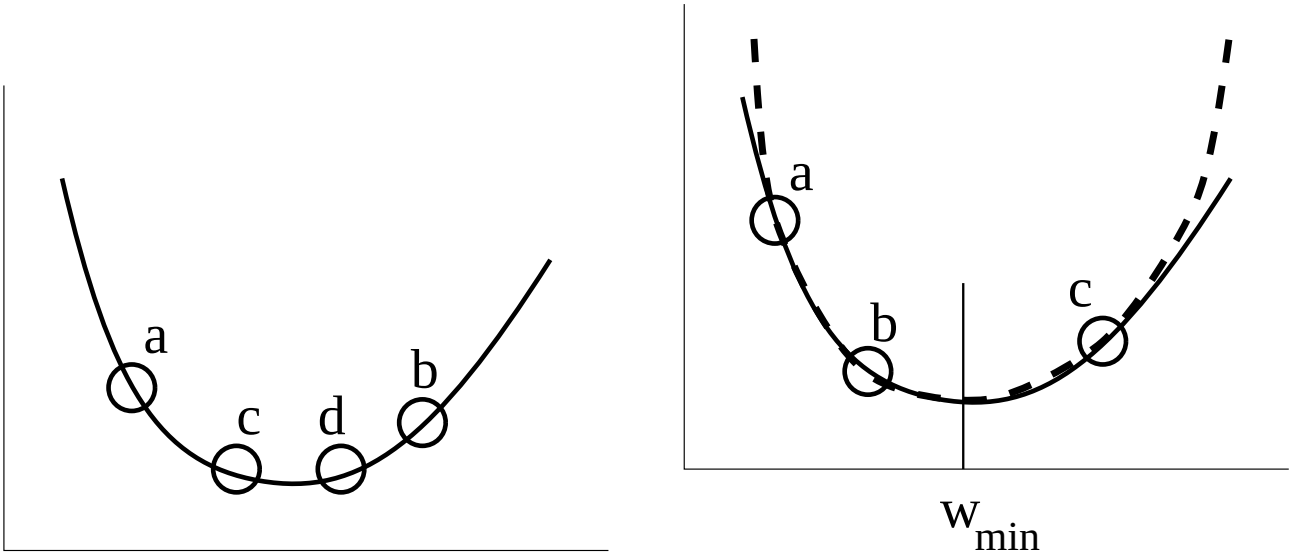

Figure 7: [left] Exact line search - interval method. [right] Approximate line search - parabola method.

function) to the line and just jumping to the function minimum (see Fig. 7).

The Bayesian approach

Why is the Bayesian approach to estimation different? Consider two systems, with an output y, whose density we wish to infer, i.e. $P(y)$. Let the system be conditioned on some data set D and parameterised via a set of parameters $\mathbf{w}$. The maximum-likelihood approach is to optimise some error functional, $E(\mathbf{w}) = -\ln P(\mathbf{w} \mid D)$, and pick the estimator for $\mathbf{w}$ as that at the minimum error, or peak (known as the *mode*) of $P(\mathbf{w} \mid D)$. Let this parameter set be $\mathbf{w}^*$, hence the estimate for the output density is:

$$\widehat{P(y \mid D)} = P(y \mid \mathbf{w}^*, D)$$

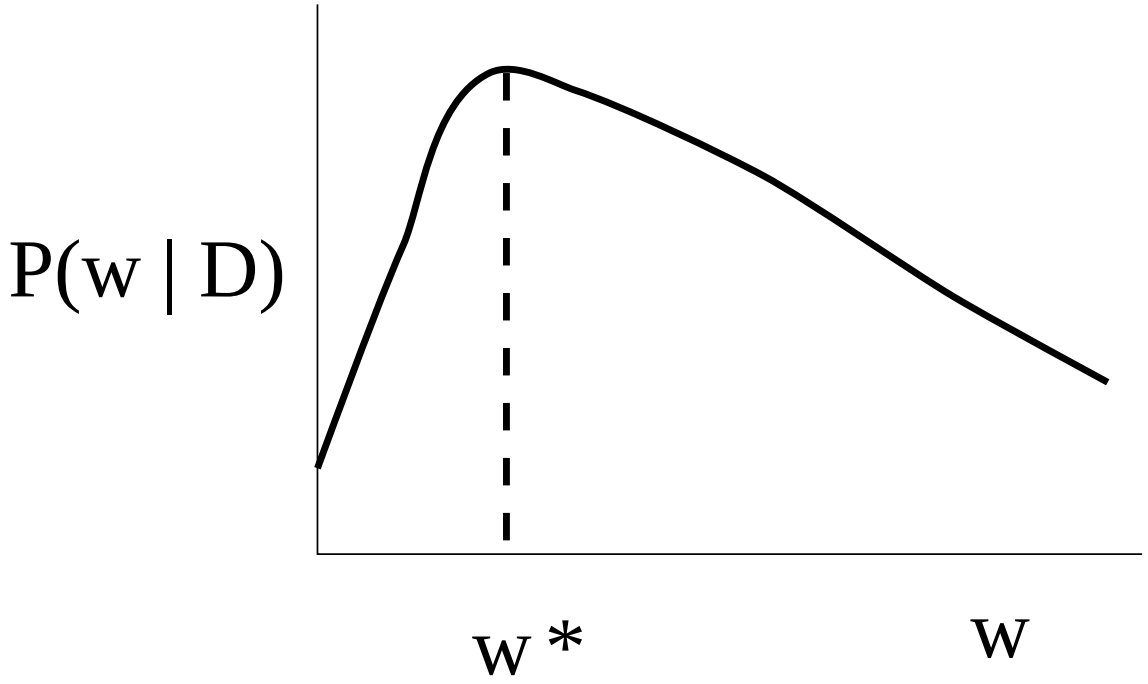

Figure 8: Optimisers find the mode of a distribution.

i.e. we plug in the value for the maximum-likelihood parameter set, $\mathbf{w}^*$. This is known as a *plug-in estimator*. The Bayesian approach attempts to expend efforts not on optimisation, but on integration. This integration is known as *marginalisation* and involves the integrating out of 'nuisance' parameters. In this simple example, the nuisance parameter is $\mathbf{w}$, and the marginal integral is:

$$\widehat{P(y \mid D)} = \int P(y \mid \mathbf{w}, D) P(\mathbf{w} \mid D) d\mathbf{w}$$

We can see that this is an integration over the density of $\mathbf{w}$, and so will give an answer which depends, *not just on the modal value,* $\mathbf{w}^*$*, but rather on the whole distribution.* Applying Bayes' theorem gives:

$$\widehat{P(y \mid D)} = \frac{1}{P(D)} \int P(y \mid \mathbf{w}, D) \quad \underbrace{P(D \mid \mathbf{w})}_{\text{Generative model}} \quad \underbrace{P(\mathbf{w})}_{\text{Prior}} d\mathbf{w}$$

NOTE: The Bayesian approach now requires two things,

- we estimate the likelihood of the *generative model* and
- we choose some form of *prior*.

It is this dependence on *prior* information that makes the Bayesian approach so different. Consider,

- Can you infer the mean of a set of numbers when $N = 2$?

- How reliable is your estimate?
- Under a Bayesian scheme, the estimates we make are distributions which encode a mixture of *information from the data* and *our prior beliefs about the problem.*
- All methods make subjective assumptions, only Bayesians come up front and make these *explicit*!

A simple illustration of Bayesian learning

Consider a set $D = \{x_t\}_{t=1}^N$ of independently Gaussian distributed random variables

$$P(x_t|\mu,\beta) \quad = \quad \sqrt{\frac{\beta}{2\pi}}\exp\left(-\frac{\beta}{2}\left[x_t-\mu\right]^2\right) \tag{1}$$

We want to infer the mean μ and the precision (inverse variance) β from the data D, whose likelihood is given by

$$\begin{aligned} P(D|\mu,\beta) &= \left(\frac{\beta}{2\pi}\right)^{N/2}\exp\left(-\frac{\beta}{2}\sum_{t=1}^N\left[x_t-\mu\right]^2\right) \\ &= \left(\frac{\beta}{2\pi}\right)^{N/2}\exp\left(-\frac{\beta}{2}\sum_{t=1}^N\left[x_t-\overline{x}\right]^2-\frac{N\beta}{2}[\overline{x}-\mu]^2\right) \end{aligned} \tag{2}$$

where

$$\overline{x} \quad = \quad \frac{1}{N}\sum_{t=1}^N x_t \tag{3}$$

is the empirical sample mean. Let us first find the maximum likelihood estimate for both parameters simultaneously. The log-likelihood for the data is given by

$$\ln P(D|\mu,\beta) \quad = \quad \frac{N}{2}\ln\left(\frac{\beta}{2\pi}\right)-\frac{\beta}{2}\sum_{t=1}^N\left(x_t-\overline{x}\right)^2-\frac{N\beta}{2}[\overline{x}-\mu]^2 \tag{4}$$

Taking the derivatives with respect to μ and β gives:

$$\begin{aligned} \frac{\partial}{\partial\mu}\ln P(D|\mu,\beta) &= N\beta(\overline{x}-\mu) \\ \frac{\partial}{\partial\beta}\ln P(D|\mu,\beta) &= \frac{N}{2\beta}-\frac{1}{2}\sum_{t=1}^N\left(x_t-\overline{x}\right)^2-\frac{N}{2}[\overline{x}-\mu]^2 \end{aligned} \tag{5}$$

Setting these derivatives to zero, we obtain for the joint maximum likelihood estimate:

$$\hat{\mu} \;=\; \overline{x}, \qquad \frac{1}{\hat{\beta}} \;=\; \frac{1}{N}\sum_{t=1}^N\left(x_t-\overline{x}\right)^2 \tag{6}$$

The mean μ is thus approximated by the empirical mean, which is an unbiased estimator for μ. However, the variance $\frac{1}{\beta}$ is approximated by $\frac{1}{N}\sum_{t=1}^{N}(x_t - \overline{x})^2$, which is *not* unbiased. The proper unbiased estimator is known to be

$$\frac{1}{N-1}\sum_{t=1}^{N}(x_t - \overline{x})^2$$

since fitting the mean from the data reduces the number of degrees of freedom by one. Put in another way, the empirical mean $\overline{x}$ inevitably fits part of the noise, so the expression on the right of equation (6) systematically underestimates the variance. This might not be a serious problem for this simple 2-parameter model if N is sufficiently large, but the problem is aggravated in more complex classifiers (such as neural networks), where considerably more parameters have to be fitted from the data.

Let us now turn to the second alternative of first integrating μ out and then finding the maximum likelihood estimate of the marginalised distribution $P(D|\beta)$. Taking a uniform prior for μ, $P(\mu) = C$, we obtain:

$$P(D|\beta) \;=\; \int P(D,\mu|\beta)d\mu \;=\; \int P(D|\mu,\beta)P(\mu)d\mu \;=\; C\int P(D|\mu,\beta)d\mu \tag{7}$$

which after inserting equation (2) yields:

$$\begin{aligned} P(D|\beta) &= C\int\left(\frac{\beta}{2\pi}\right)^{N/2}\exp\left(-\frac{\beta}{2}\sum_{t=1}^{N}[x_t-\overline{x}]^2 - \frac{N\beta}{2}[\overline{x}-\mu]^2\right)d\mu \\ &= C\left(\frac{\beta}{2\pi}\right)^{N/2}\exp\left(-\frac{\beta}{2}\sum_{t=1}^{N}[x_t-\overline{x}]^2\right)\int\exp\left(-\frac{N\beta}{2}[\overline{x}-\mu]^2\right)d\mu \\ &\propto \left(\frac{\beta}{2\pi}\right)^{(N-1)/2}\exp\left(-\frac{\beta}{2}\sum_{t=1}^{N}[x_t-\overline{x}]^2\right) \end{aligned} \tag{8}$$

We obtain the maximum likelihood estimate for β by setting the derivative of the log-likelihood to zero:

$$\frac{\partial}{\partial\beta}\ln P(D|\beta) \;=\; \frac{\partial}{\partial\beta}\left[\frac{N-1}{2}\ln\left(\frac{\beta}{2\pi}\right) - \frac{\beta}{2}\sum_{t=1}^{N}[x_t-\overline{x}]^2\right] \;=\; 0 \tag{9}$$

This leads to the correct *unbiased* estimate for the variance:

$$\frac{1}{\hat{\beta}} \quad = \quad \frac{1}{N-1}\sum_{t=1}^{N}(x_t-\overline{x})^2 \tag{10}$$

The following Figs. give a simple illustration of the Bayesian learning paradigm in action. We assume we know the variance, $\sigma^2 = 1$ and want to infer the mean.

The prior is given, in these examples by $P(\mu) \sim N(0, 1)$. One example draws 50 samples, the other only 2. Look at the difference between the Bayesian density over μ and the single estimate given by the ML approach! This was produced via the code `blearn.m`, available on the WWW site.

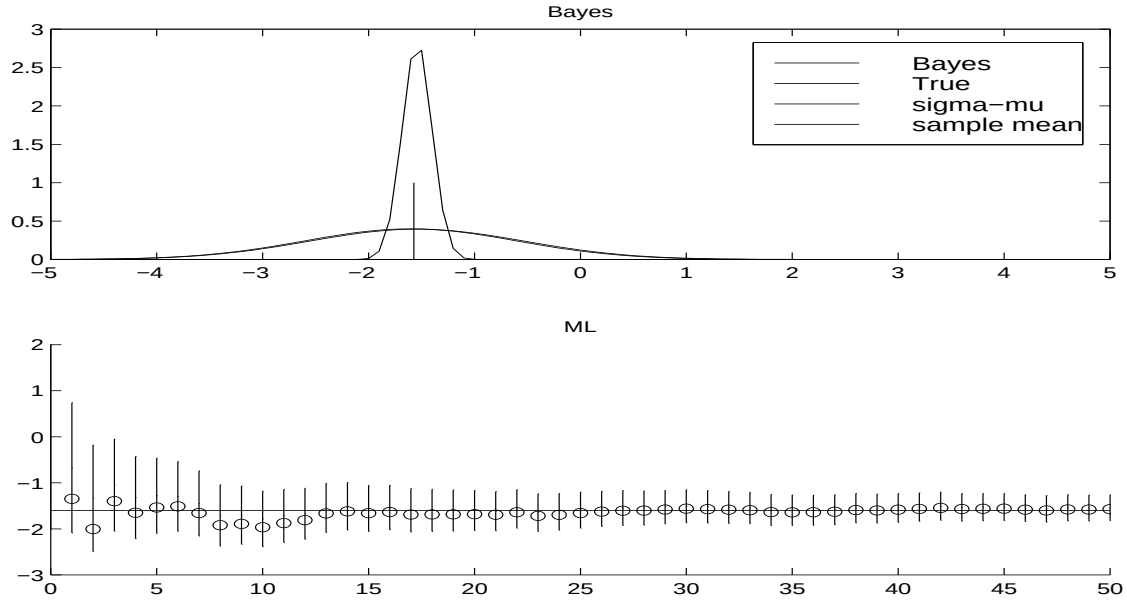

Figure 9: 50 samples.

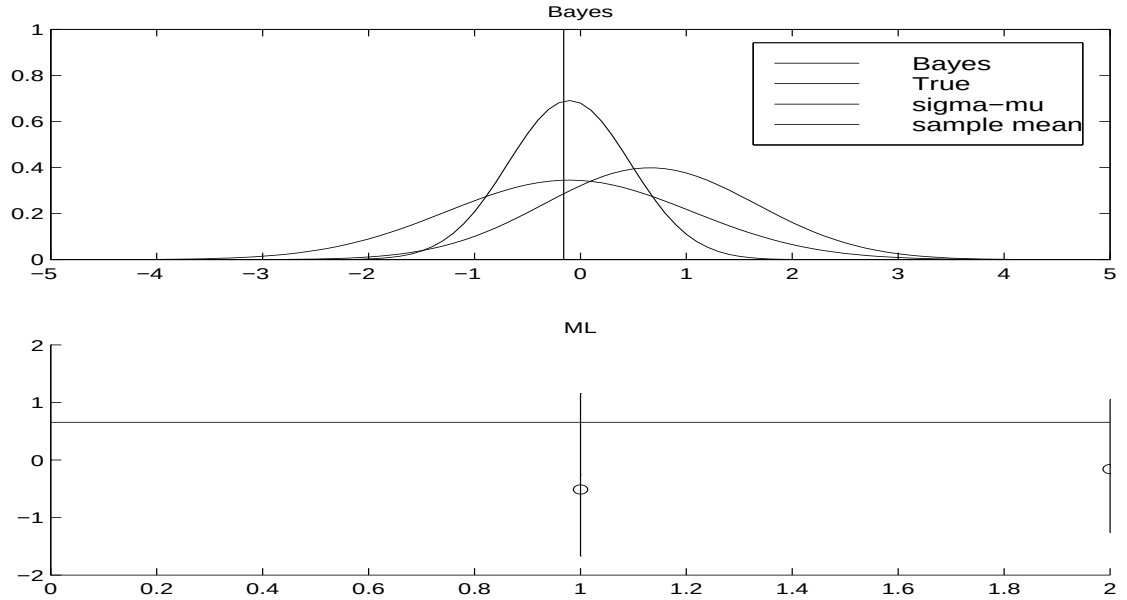

Figure 10: Only 2 samples.

We have, so far, only looked at cases in which the marginal integral is analytic. In general we have three choices:

1. Choose, if possible, the form of the prior so as to make the marginal integral analytic. The obvious drawback is that this form may not truly code our prior beliefs.

2. Make some approximations – we will return to this at the end of the next chapter.
3. Numerically integrate the expression. This can be efficiently achieved in many cases (using, e.g., variants of Monte-Carlo methods) and this is in keeping with the spirit of Bayesian learning.