Machine Learning and Bayesian Inference

Dr Sean Holden

Computer Laboratory, Room FC06

Telephone extension 63725

Email: sbh11@cl.cam.ac.uk

www.cl.cam.ac.uk/∼sbh11/

Part III: back to Bayes

Bayesian neural networks

Gaussian processes

Where now?

There are some simple *take-home messages* from the study of SVMs:

You can get *state-of-the-art* performance.

You can do this using the *kernel trick* to obtain a *non-linear model*.

You can do this without invoking the *full machinery of the Bayes-optimal classifier*.

BUT:

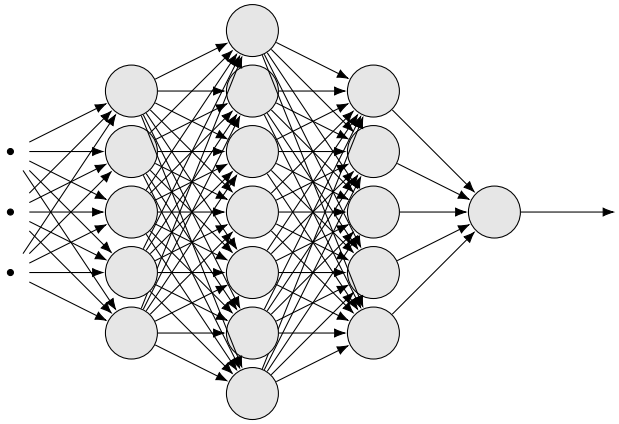
You don't have anything *keeping you honest* regarding *which assumptions you're making*.

As we shall see, by using the *full-strength probabilistic framework* we gain some useful extras.

In particular, the ability to *assign confidences* to our predictions.

The Bayesian approach to neural networks

We're now going to see how the idea of the *Bayes-optimal classifier* can be applied to *neural networks*.



We have:

- A *neural network* computing a function $h_{\mathbf{w}}(\mathbf{x})$. (In fact this can be *pretty much any* parameterized function we like.)
- A training sequence $\mathbf{s}^T = \left[(\mathbf{x}_1, y_1) \ \cdots \ (\mathbf{x}_m, y_m)\right]$, split into

$$\mathbf{y} = \left(y_1 \ \ y_2 \ \ \cdots \ \ y_m \right)$$

and

$$\mathbf{X} = \left(\mathbf{x}_1 \ \ \mathbf{x}_2 \ \ \cdots \ \ \mathbf{x}_m \right).$$

The Bayesian approach to neural networks

We're *only going to consider regression*. Classification can also be done this way, but it's a bit more complicated.

For *classification* we derived the Bayes-optimal classifier as the *maximizer* of:

$$\Pr(C|\mathbf{x}, \mathbf{s}) = \int \Pr(C|\mathbf{w}, \mathbf{x})\, p(\mathbf{w}|\mathbf{s})\, d\mathbf{w}$$

For regression the *Bayes-optimal classifier* ends up having the same expression as we've already seen. We want to compute:

$$p(Y|\mathbf{x}, \mathbf{s}) = \int \underbrace{p(Y|\mathbf{w}, \mathbf{x})}_{\text{Likelihood}} \underbrace{p(\mathbf{w}|\mathbf{s})}_{\text{Posterior}}\, d\mathbf{w}$$

$\mathbf{s}$ is the *training set*.

$\mathbf{x}$ is a *new example* to be classified.

Y is the RV representing the *prediction* for $\mathbf{x}$.

The Bayesian approach to neural networks

It turns out that *if you try to incorporate* the density $p(\mathbf{x})$ modelling how *feature vectors* are generated, things can get complicated. So:

1. We regard all input vectors as *fixed*: they are *not* treated as random variables.
2. This means that, *strictly speaking*, they should no longer appear in expressions like $p(Y|\mathbf{w}, \mathbf{x})$.
3. However, this seems to be uniformly disliked—writing $p(Y|\mathbf{w})$ for an expression that still depends on $\mathbf{x}$ seems confusing.
4. Solution: write $p(Y|\mathbf{w}; \mathbf{x})$ instead. Note the *semi-colon*!

So we're actually going to look at

$$p(Y|\mathbf{y}; \mathbf{x}, \mathbf{X}) = \int \underbrace{p(Y|\mathbf{w}; \mathbf{x})}_{\text{Likelihood}} \underbrace{p(\mathbf{w}|\mathbf{y}; \mathbf{X})}_{\text{Posterior}} \, d\mathbf{w}$$

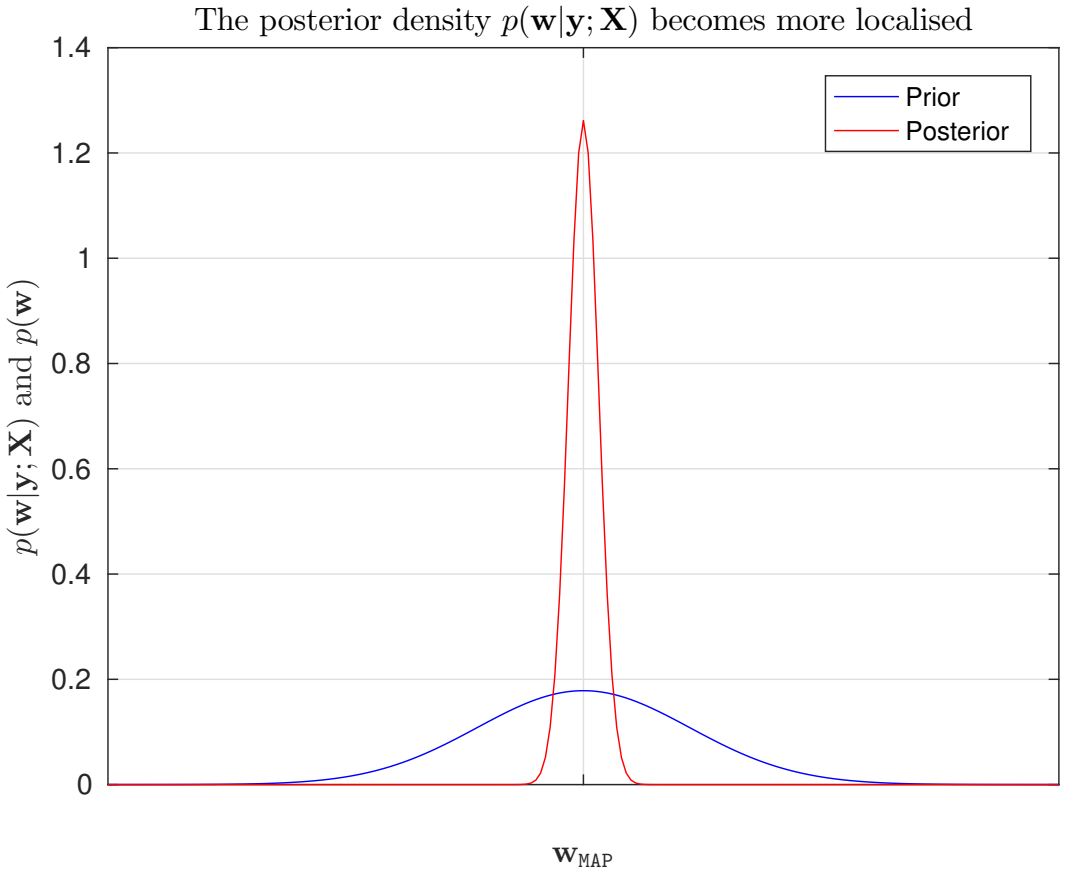
NOTE: this is a *notational hack*. There's nothing new, just an attempt at clarity.

What's going on? Turning prior into posterior

Let's make a brief sidetrack into what's going on with the posterior density

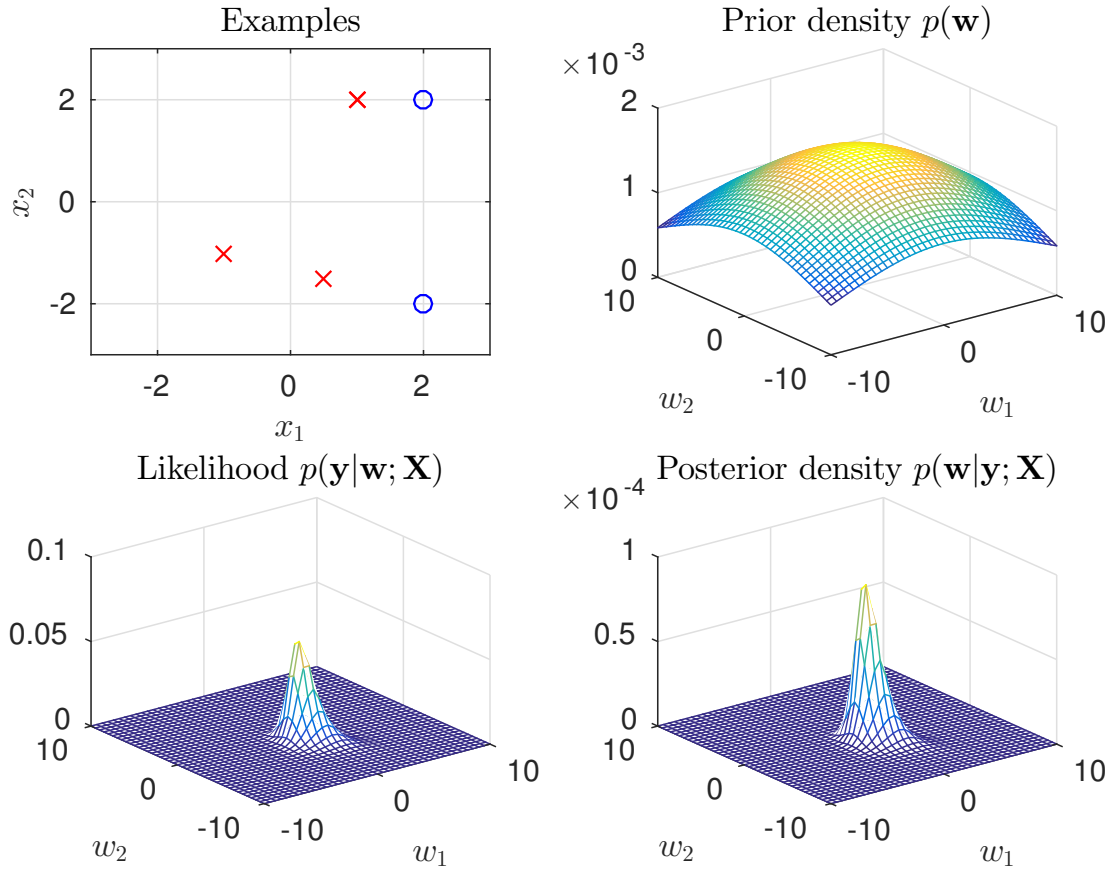
$$p(\mathbf{w}|\mathbf{y};\mathbf{X}) \propto p(\mathbf{y}|\mathbf{w};\mathbf{X})p(\mathbf{w}).$$

Typically, the *prior starts wide* and as we see more data the *posterior narrows*



What's going on? Turning prior into posterior

This can be seen very clearly if we use real numbers:



The Bayesian approach to neural networks

So now we have three things to do:

1. *STEP 1:* remind ourselves what $p(Y|\mathbf{w};\mathbf{x})$ is.
2. *STEP 2:* remind ourselves what $p(\mathbf{w}|\mathbf{y};\mathbf{X})$ is.
3. *STEP 3:* do the integral. (*This is the fun bit...*)

The first two steps are straightforward as *we've already derived them* when looking at *maximum-likelihood* and *MAP* learning.

STEP 1: assuming Gaussian noise is added to the labels so

$$y = h_{\mathbf{w}}(\mathbf{x}) + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ we have the usual likelihood

$$p(Y|\mathbf{w}; \mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{1}{2\sigma_n^2}\left(Y - h_{\mathbf{w}}(\mathbf{x})\right)^2\right).$$

Here, the subscript in σ_n^2 reminds us that it's the variance of the *noise*.

Traditionally this is re-written using the *hyperparameter*

$$\beta = \frac{1}{\sigma_n^2}$$

so the *likelihood* is

$$p(Y|\mathbf{w}; \mathbf{x}) \propto \exp\left(-\frac{\beta}{2}\left(Y - h_{\mathbf{w}}(\mathbf{x})\right)^2\right).$$

STEP 2: the *posterior* is also exactly as it was when we derived the *MAP* learning algorithms.

$$p(\mathbf{w}|\mathbf{y};\mathbf{X}) \propto p(\mathbf{y}|\mathbf{w};\mathbf{X})p(\mathbf{w})$$

and as before, the likelihood is

$$p(\mathbf{y}|\mathbf{w};\mathbf{X}) \propto \exp\left(-\frac{\beta}{2}\sum_{i=1}^{m}\left(y_i - h_{\mathbf{w}}(\mathbf{x}_i)\right)^2\right)$$
$$= \exp\left(-\beta E(\mathbf{w})\right)$$

and using a Gaussian prior with mean $\mathbf{0}$ and covariance $\mathbf{\Sigma} = \sigma^2\mathbf{I}$ gives

$$p(\mathbf{w}) \propto \exp\left(-\frac{\alpha}{2}||\mathbf{w}||^2\right)$$

where traditionally the second *hyperparameter* is $\alpha = 1/\sigma^2$. Combining these

$$p(\mathbf{w}|\mathbf{y};\mathbf{X}) = \frac{1}{Z(\alpha,\beta)}\exp\left(-\left(\frac{\alpha||\mathbf{w}||^2}{2} + \beta E(\mathbf{w})\right)\right).$$

What's going on? Turning prior into posterior

Considering the central part of $p(\mathbf{w}|\mathbf{y};\mathbf{X})$:

$$\frac{\alpha||\mathbf{w}||^2}{2} + \beta E(\mathbf{w}).$$

What happens as the number m of examples increases?

- The first term *corresponding to the prior* remains fixed.
- The second term *corresponding to the likelihood* increases.

So for small training sequences the prior dominates, but for large ones $\mathbf{w}_{\text{ML}}$ is a good approximation to $\mathbf{w}_{\text{MAP}}$.

Step 3: putting together steps 1 and 2, the integral we need to evaluate is:

$$I \propto \int \underbrace{\exp\left(-\frac{\beta}{2}\left(Y - h_{\mathbf{w}}(\mathbf{x})\right)^2\right)}_{\text{Likelihood}} \underbrace{\exp\left(-\left(\frac{\alpha||\mathbf{w}||^2}{2} + \beta E(\mathbf{w})\right)\right)}_{\text{Posterior}} d\mathbf{w}.$$

Obviously this *gives us all a sad face* because there is *no solution*.

So what can we do now. . . ?

The Bayesian approach to neural networks

In order to make further progress it's necessary to perform integrals of the general form

$$\int F(\mathbf{w}) p(\mathbf{w}|\mathbf{y};\mathbf{X})\; d\mathbf{w}$$

for various functions F and this is generally not possible.

There are two ways to get around this:

1. We can use an *approximate form* for $p(\mathbf{w}|\mathbf{y};\mathbf{X})$.
2. We can use *Monte Carlo* methods.

We'll be taking a look at both possibilities.

Method 1: approximation to $p(\mathbf{w}|\mathbf{y};\mathbf{X})$

$$I \propto \int \underbrace{\exp\left(-\frac{\beta}{2}\left(Y - h_{\mathbf{w}}(\mathbf{x})\right)^2\right)}_{\text{Likelihood } p(Y|\mathbf{w};\mathbf{x})} \underbrace{\exp\left(-\left(\frac{\alpha||\mathbf{w}||^2}{2} + \beta E(\mathbf{w})\right)\right)}_{\text{Posterior } p(\mathbf{w}|\mathbf{y};\mathbf{X})} d\mathbf{w}.$$

The first approach introduces a *Gaussian approximation* to $p(\mathbf{w}|\mathbf{y};\mathbf{X})$ by using a *Taylor expansion* of

$$S(\mathbf{w}) = \frac{\alpha||\mathbf{w}||^2}{2} + \beta E(\mathbf{w})$$

at the *maximum a posteriori* weights $\mathbf{w}_{\text{MAP}}$.

This allows us to use a *standard integral*.

The result will be *approximate* but we hope it's good!

Let's recall how Taylor series work...

Reminder: Taylor expansion

In *one dimension* the *Taylor expansion* about a point $x_0 \in \mathbb{R}$ for a function $f : \mathbb{R} \to \mathbb{R}$ is

$$\begin{aligned} f(x) \approx f(x_0) &+ \frac{1}{1!}(x - x_0) f'(x_0) \\ &+ \frac{1}{2!}(x - x_0)^2 f''(x_0) \\ &+ \cdots + \frac{1}{k!}(x - x_0)^k f^k(x_0). \end{aligned}$$

What does this look like for the kinds of function we're interested in? As an *example* We can try to approximate

$$\exp\left(-f(x)\right)$$

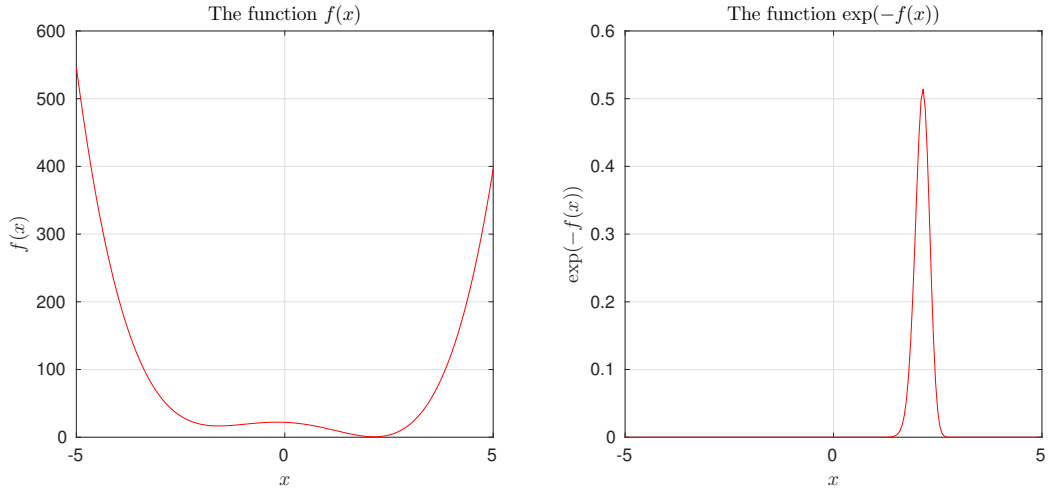
where

$$f(x) = x^4 - \frac{1}{2}x^3 - 7x^2 - \frac{5}{2}x + 22.$$

This has a *form similar to* $S(\mathbf{w})$, but in one dimension.

Reminder: Taylor expansion

The functions of interest look like this:



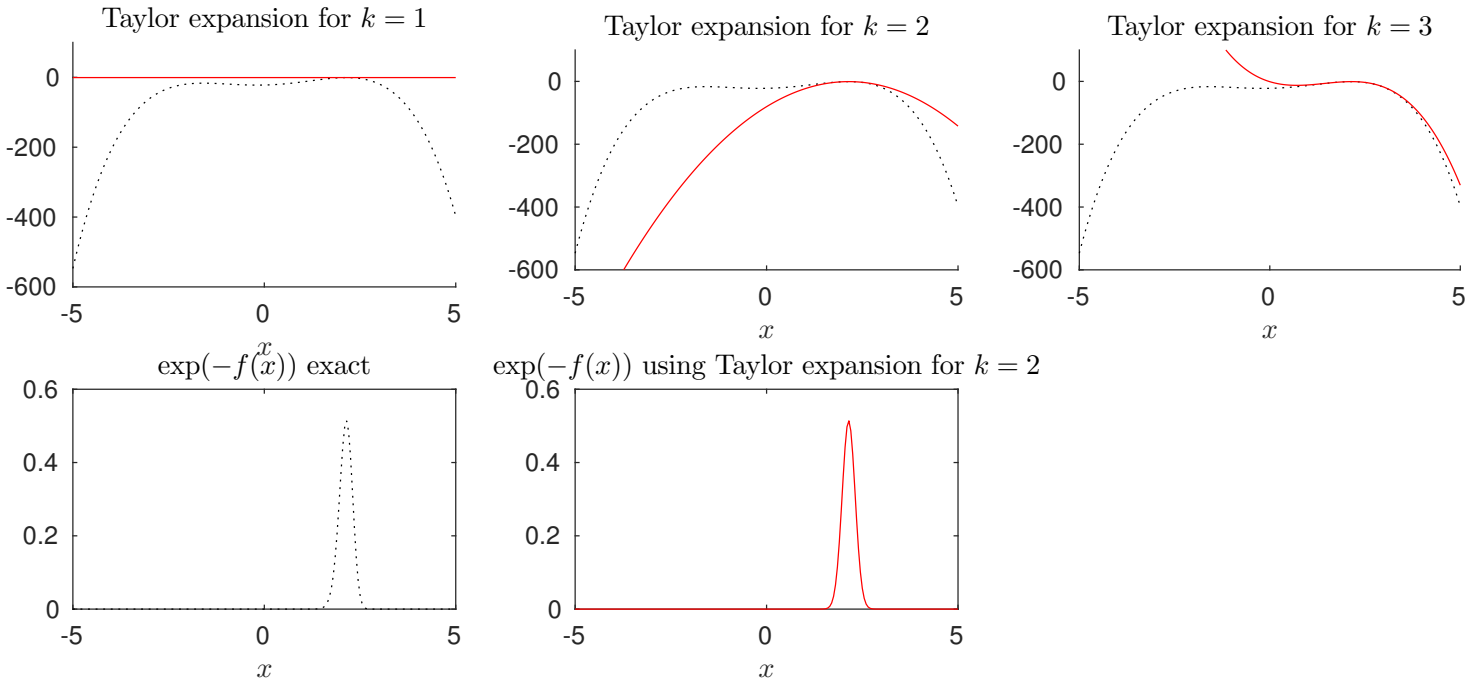
By replacing $-f(x)$ with its *Taylor expansion about its maximum*, which is at

$$x_{\text{max}} = 2.1437$$

we can see what the *approximation to* $\exp(-f(x))$ looks like. Note that the $\exp$ *hugely emphasises peaks*.

Reminder: Taylor expansion

Here are the approximations for $k = 1$, $k = 2$ and $k = 3$.



The use of $k = 2$ looks promising...

Reminder: Taylor expansion

In *multiple dimensions* the Taylor expansion for $k = 2$ is

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + \frac{1}{1!}(\mathbf{x} - \mathbf{x}_0)^T \nabla f(\mathbf{x})|_{\mathbf{x}_0} + \frac{1}{2!}(\mathbf{x} - \mathbf{x}_0)^T \nabla^2 f(\mathbf{x})\big|_{\mathbf{x}_0} (\mathbf{x} - \mathbf{x}_0)$$

where ∇ denotes *gradient*

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f(\mathbf{x})}{\partial x_1} \quad \frac{\partial f(\mathbf{x})}{\partial x_2} \quad \cdots \quad \frac{\partial f(\mathbf{x})}{\partial x_n} \right)$$

and $\nabla^2 f(\mathbf{x})$ is the matrix with elements

$$M_{ij} = \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}$$

(Looks complicated, but it's just the obvious extension of the 1-dimensional case.)

Method 1: approximation to $p(\mathbf{w}|\mathbf{y};\mathbf{X})$

Applying this to $S(\mathbf{w})$ and expanding around $\mathbf{w}_{\text{MAP}}$

$$S(\mathbf{w}) = \frac{\alpha||\mathbf{w}||^2}{2} + \beta E(\mathbf{w}) \approx S(\mathbf{w}_{\text{MAP}}) + \frac{1}{2}(\mathbf{w} - \mathbf{w}_{\text{MAP}})^T \mathbf{A}(\mathbf{w} - \mathbf{w}_{\text{MAP}}).$$

- As $\mathbf{w}_{\text{MAP}}$ *minimises* the function the first derivatives are zero and the corresponding term in the Taylor expansion *disappears*.
- The quantity $\mathbf{A} = \nabla\nabla S(\mathbf{w})|_{\mathbf{w}_{\text{MAP}}}$ can be simplified.

This is because

$$\mathbf{A} = \nabla\nabla \left(\frac{\alpha||\mathbf{w}||^2}{2} + \beta E(\mathbf{w}) \right) \Big|_{\mathbf{w}_{\text{MAP}}} = \alpha\mathbf{I} + \beta\nabla\nabla E(\mathbf{w}_{\text{MAP}}).$$

Method 1: approximation to $p(\mathbf{w}|\mathbf{y};\mathbf{X})$

We actually already know something about how to get $\mathbf{w}_{\text{MAP}}$:

1. A method such as *backpropagation* can be used to compute $\nabla S(\mathbf{w})$.
2. The vector $\mathbf{w}_{\text{MAP}}$ can then be obtained using any standard optimisation method (such as *gradient descent*).

It's also likely to be straightforward to compute $\nabla\nabla E(\mathbf{w})$:

The quantity $\nabla\nabla E(\mathbf{w})$ can be evaluated using an *extended form of backpropagation*.

A useful integral

Dropping *for this slide only* the special meaning usually given to the vector $\mathbf{x}$, here is a useful standard integral:

If $\mathbf{A} \in \mathbb{R}^{n \times n}$ is symmetric then for $\mathbf{b} \in \mathbb{R}^n$ and $c \in \mathbb{R}$

$$\int_{\mathbb{R}^n} \exp\left(-\frac{1}{2}\left(\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{b} + c\right)\right) d\mathbf{x}$$

$$= (2\pi)^{n/2}|\mathbf{A}|^{-1/2} \exp\left(-\frac{1}{2}\left(c - \frac{\mathbf{b}^T\mathbf{A}^{-1}\mathbf{b}}{4}\right)\right).$$

You're not expected to know how to evaluate this, but *see the handout on the course web page* if you're curious[1].

To make this easy to refer to, let's call it the *BIG INTEGRAL*.

[1] No, I won't ask you to evaluate it in the exam...

Method 1: approximation to $p(\mathbf{w}|\mathbf{y};\mathbf{X})$

Defining

$$\Delta\mathbf{w} = \mathbf{w} - \mathbf{w}_{\text{MAP}}$$

we now have an approximation

$$p(\mathbf{w}|\mathbf{y};\mathbf{X}) \approx \frac{1}{Z}\exp\left(-S(\mathbf{w}_{\text{MAP}}) - \frac{1}{2}\Delta\mathbf{w}^T\mathbf{A}\Delta\mathbf{w}\right).$$

Using the *BIG INTEGRAL*

$$Z = (2\pi)^{W/2}|\mathbf{A}|^{-1/2}\exp(-S(\mathbf{w}_{\text{MAP}}))$$

where W is the number of weights.

Let's plug this approximation back into the *expression for the Bayes-optimum* and see what we get...

Method 1: approximation to $p(\mathbf{w}|\mathbf{y};\mathbf{X})$

$$I \propto \int \underbrace{\exp\left(-\frac{\beta}{2}\left(Y - h_{\mathbf{w}}(\mathbf{x})\right)^2\right)}_{\text{Likelihood } p(Y|\mathbf{w};\mathbf{x})} \underbrace{\exp\left(-\frac{1}{2}\Delta\mathbf{w}^T\mathbf{A}\Delta\mathbf{w}\right)}_{\text{Approximation to } p(\mathbf{w}|\mathbf{y};\mathbf{X})} d\mathbf{w}.$$

There is still *no solution!* We need *another approximation...*

We can introduce a *linear approximation*[2] of $h_{\mathbf{w}}(\mathbf{x})$ at $\mathbf{w}_{\text{MAP}}$:

$$h_{\mathbf{w}}(\mathbf{x}) \approx h_{\mathbf{w}_{\text{MAP}}}(\mathbf{x}) + \mathbf{g}^T\Delta\mathbf{w}$$

where $\mathbf{g} = \nabla h_{\mathbf{w}}(\mathbf{x})\big|_{\mathbf{w}_{\text{MAP}}}$.

(By linear approximation we just mean the Taylor expansion for $k = 1$.)

[2] We really are making assumptions here—this is OK if we assume that $p(\mathbf{w}|\mathbf{y};\mathbf{X})$ is *narrow*, which depends on $\mathbf{A}$.

Method 1: second approximation

This leads to

$$p(Y|\mathbf{y};\mathbf{x},\mathbf{X}) \propto \exp\left(-\frac{\beta}{2}\left(Y - h_{\mathbf{w}_{\text{MAP}}}(\mathbf{x}) - \mathbf{g}^T\Delta\mathbf{w}\right)^2 - \frac{1}{2}\Delta\mathbf{w}^T\mathbf{A}\Delta\mathbf{w}\right) d\mathbf{w}.$$

SUCCESS!!!

This integral can be evaluated (this is an *exercise*) using the *BIG INTEGRAL* to give *THE ANSWER...*

$$p(Y|\mathbf{y};\mathbf{x},\mathbf{X}) \simeq \frac{1}{\sqrt{2\pi\sigma_Y^2}} \exp\left(-\frac{(y - h_{\mathbf{w}_{\text{MAP}}}(\mathbf{x}))^2}{2\sigma_Y^2}\right)$$

where

$$\sigma_Y^2 = \frac{1}{\beta} + \mathbf{g}^T\mathbf{A}^{-1}\mathbf{g}.$$

Method 1: final expression

Hooray! But what does it mean?

This is a *Gaussian density*, so we can now see that:

$p(Y|\mathbf{y};\mathbf{x},\mathbf{X})$ *peaks* at $h_{\mathbf{w}_{\text{MAP}}}(\mathbf{x})$.

That is, the *MAP solution*.

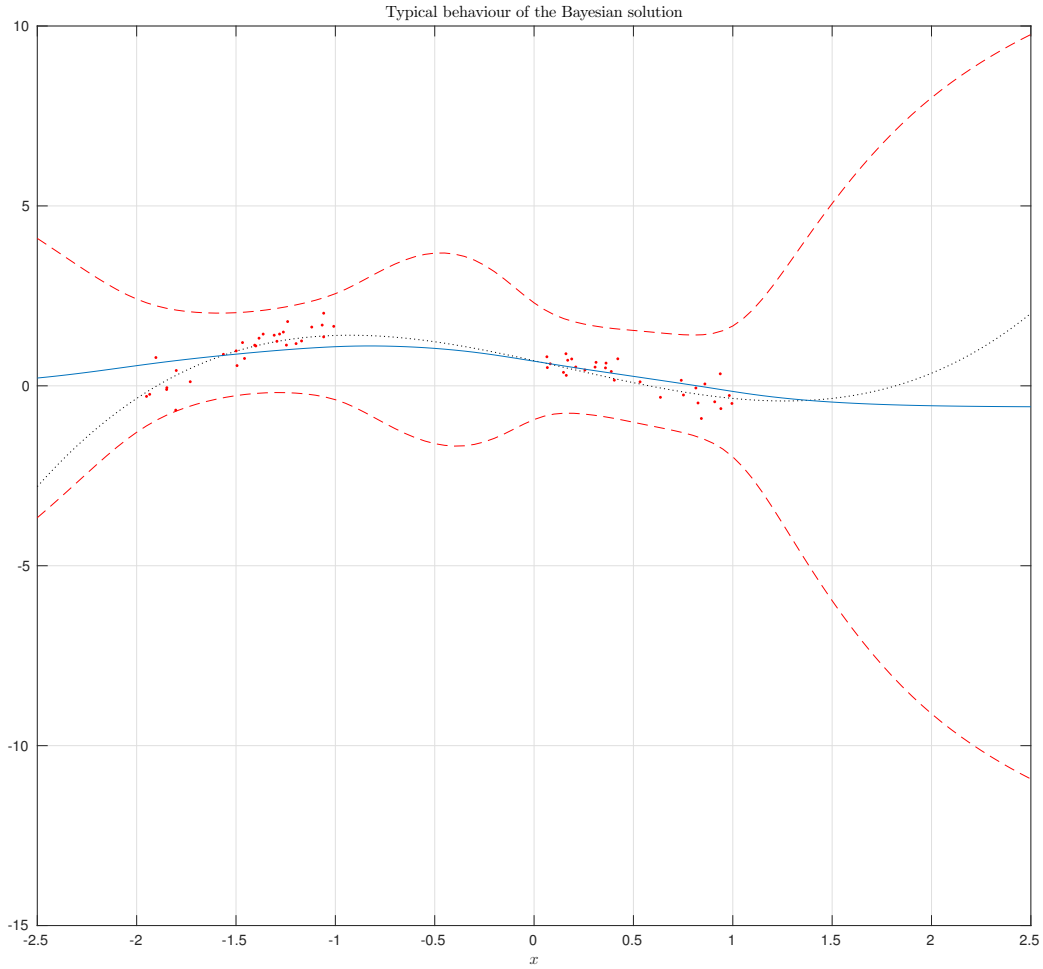
The *variance* σ_Y^2 can be interpreted as a measure of *certainty*:

The first term of σ_Y^2 is $1/\beta$ and corresponds to the noise.

The second term of σ_Y^2 is $\mathbf{g}^T\mathbf{A}^{-1}\mathbf{g}$ and corresponds to the width of $p(\mathbf{w}|\mathbf{y};\mathbf{X})$.

Method 1: final expression

Hooray! But what does it mean? Interpreted graphically:



Plotting $\pm 2\sigma_Y$ around the prediction gives a *measure of certainty*.

Method II: Markov chain Monte Carlo (MCMC) methods

The second solution to the problem of performing integrals

$$I = \int F(\mathbf{w}) p(\mathbf{w}|\mathbf{y}; \mathbf{X}) d\mathbf{w}$$

is to use *Monte Carlo* methods. The basic approach is to make the approximation

$$I \approx \frac{1}{N} \sum_{i=1}^{N} F(\mathbf{w}_i)$$

where the $\mathbf{w}_i$ have distribution $p(\mathbf{w}|\mathbf{y}; \mathbf{X})$. Unfortunately, generating $\mathbf{w}_i$ with a *given distribution* can be non-trivial.

MCMC methods

A simple technique is to introduce a random walk, so

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \epsilon$$

where ϵ is *zero mean spherical Gaussian* and has *small variance*. Obviously the sequence $\mathbf{w}_i$ does not have the required distribution. However, we can use the *Metropolis algorithm*, which does *not* accept all the steps in the random walk:

1. If $p(\mathbf{w}_{i+1}|\mathbf{y};\mathbf{X}) > p(\mathbf{w}_i|\mathbf{y};\mathbf{X})$ then accept the step.
2. Else accept the step with probability $\frac{p(\mathbf{w}_{i+1}|\mathbf{y};\mathbf{X})}{p(\mathbf{w}_i|\mathbf{y};\mathbf{X})}$.

In practice, the Metropolis algorithm has several shortcomings, and a great deal of research exists on improved methods, see:

R. Neal, "Probabilistic inference using Markov chain Monte Carlo methods," University of Toronto, Department of Computer Science Technical Report CRG-TR-93-1, 1993.

A (very) brief introduction to how to learn hyperparameters

So far in our coverage of the Bayesian approach to neural networks, the *hyperparameters* α and β were assumed to be known and fixed.

- But this is not a good assumption because...
- ...α corresponds to the width of the prior and β to the noise variance.
- So we really want to learn these from the data as well.
- How can this be done?

We now take a look at one of several ways of addressing this problem.

Note: from now on I'm going to leave out the dependencies on $\mathbf{x}$ and $\mathbf{X}$ as leaving them in starts to make everything cluttered.

The prior and likelihood depend on α and β respectively so we now make this clear and write

$$p(\mathbf{w}|\mathbf{y}, \alpha, \beta) = \frac{p(\mathbf{y}|\mathbf{w}, \beta)p(\mathbf{w}|\alpha)}{p(\mathbf{y}|\alpha, \beta)}.$$

Don't worry about recalling the *actual expressions* for the prior and likelihood—we're not going to delve deep enough to need them.

Let's write down directly something that might be useful to know:

$$p(\alpha, \beta|\mathbf{y}) = \frac{p(\mathbf{y}|\alpha, \beta)p(\alpha, \beta)}{p(\mathbf{y})}.$$

Hierarchical Bayes and the evidence

If we know $p(\alpha, \beta|\mathbf{y})$ then a straightforward approach is to *use the values for α and β that maximise it*:

$$\operatorname*{argmax}_{\alpha,\beta} p(\alpha, \beta|\mathbf{y}).$$

Here is a standard trick: *assume that the prior $p(\alpha, \beta)$ is flat*, so that we can just maximise

$$p(\mathbf{y}|\alpha, \beta).$$

This is called *type II maximum likelihood* and is one common way of doing the job.

Hierarchical Bayes and the evidence

The quantity

$$p(\mathbf{y}|\alpha, \beta)$$

is called the *evidence* or *marginal likelihood*.

When we re-wrote our earlier equation for the posterior density of the weights, making α and β explicit, we found

$$p(\mathbf{w}|\mathbf{y}, \alpha, \beta) = \frac{p(\mathbf{y}|\mathbf{w}, \beta)p(\mathbf{w}|\alpha)}{p(\mathbf{y}|\alpha, \beta)}.$$

So *the evidence is the denominator in this equation*.

This is the *common pattern* and leads to the idea of *hierarchical Bayes*: the *evidence for the hyperparameters* at one level is the *denominator in the relevant application of Bayes' theorem*.

Gaussian processes: inference with functions instead of parameters

There is an alternative approach to Bayesian regression and classification:

The fundamental idea is to *not* think in terms of *weights* $\mathbf{w}$ *that specify functions*.

Instead the idea is to *deal with functions directly*.

Fundamental to this is the concept of a *Gaussian process*.

Gaussian processes: inference with functions instead of parameters

We will continue to omit the dependencies on $\mathbf{x}$ and $\mathbf{X}$ to keep the notation simple.

We have seen that *inference* can be performed by:

1. Computing the *posterior density* $p(\mathbf{w}|\mathbf{y})$ of the parameters given the observed labels.
2. Computing the *Bayes-optimal* prediction

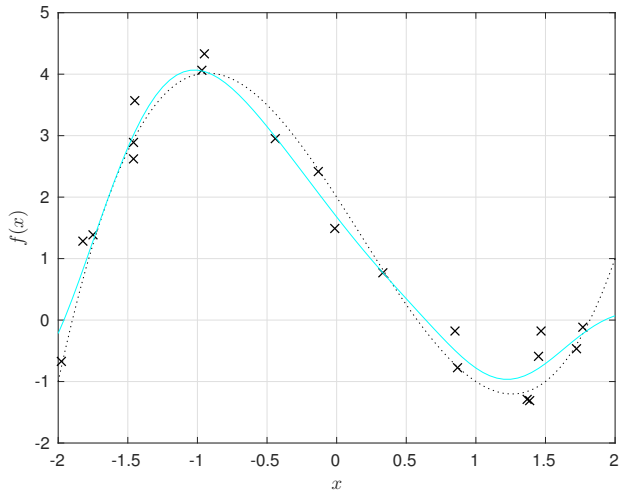
$$p(Y|\mathbf{y}) = \int p(Y|\mathbf{w})p(\mathbf{w}|\mathbf{y})\, d\mathbf{w}$$

   which is the expected value of the *likelihood* for a new point $\mathbf{x}$.
3. Choosing any *hyperparameters* $\mathbf{p}$ using the *evidence* $p(\mathbf{y}|\mathbf{p})$.

But shouldn't we deal with *functions* directly, not via parameters?

Gaussian processes: inference with functions instead of parameters

What happens if we *deal directly with functions* f, rather than choosing them via *parameters*?



Can we change the equation for prediction to

$$p(Y|\mathbf{y}) = \int p(Y|f)p(f|\mathbf{y})\, df$$

in any sensible way?

Gaussian processes: inference with functions instead of parameters

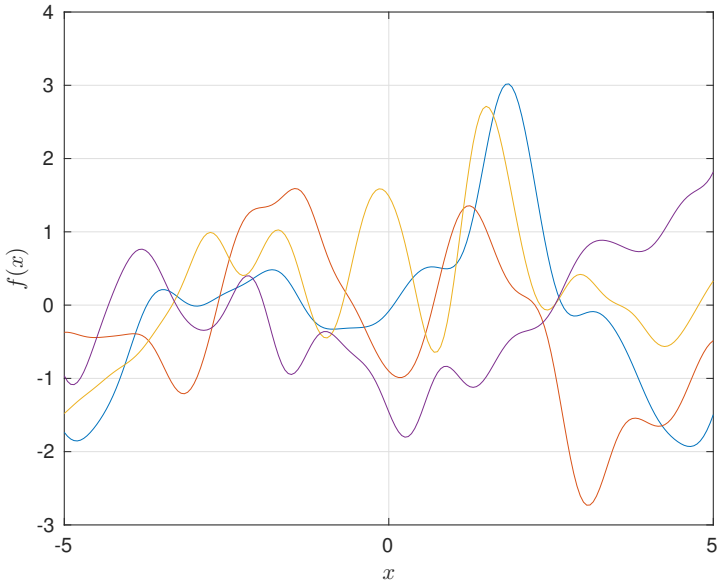
Can we change the equation for prediction to

$$p(Y|\mathbf{y}) = \int p(Y|f)p(f|\mathbf{y})\ df$$

in any sensible way?

This obviously requires us to talk about *probability densities over functions*. That is probably not something you have ever seen before.

In the diagram: four samples $f \sim p(F)$ from a *probability density defined on functions*.



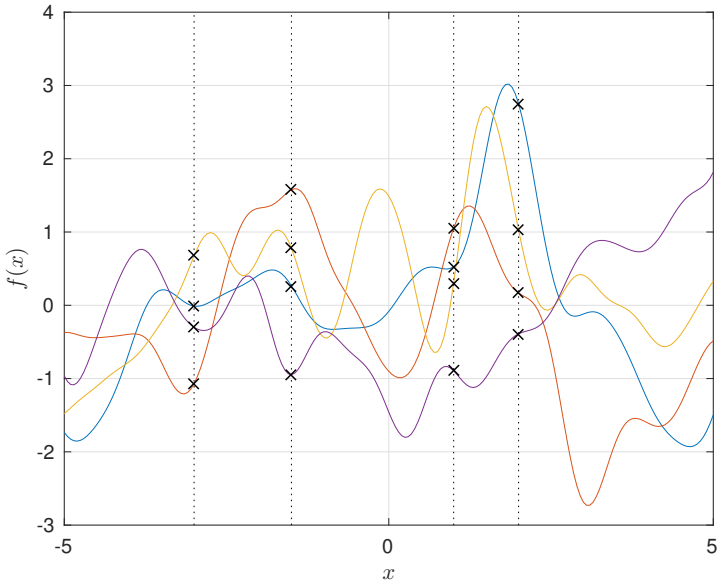
This is quite straightforward, using the concept of a *Gaussian process (GP)*.

Gaussian processes: inference with functions instead of parameters

Definition: say we have a set of RVs. This set forms a *Gaussian process* if any *finite subset* of them is *jointly Gaussian distributed*.

The same four samples $f \sim p(F)$, where F is in fact a GP.

The crosses mark the values of the sampled functions at four different values of x.



Because F is a GP *any* such finite set of values has a jointly Gaussian distribution.

Gaussian processes: inference with functions instead of parameters

What happens when we randomly select a function that is a GP?

- We are only ever interested in a *finite number of its values*.
- This is because we only need to deal with the values in the *training set* and for *any new points* we want to predict.
- Consequently we can use a GP as a *prior* rather than having a prior $p(\mathbf{w})$.

Note again the key point: we are randomly selecting *functions* and we can say something about their behaviour for *any finite collection of arguments*.

And that is enough, as we only ever have *finite quantities of data*.

Gaussian processes: inference with functions instead of parameters

To specify a GP on vectors in $\mathbb{R}^n$, we just need:

1. A *mean function* $m : \mathbb{R}^n \to \mathbb{R}$.
2. A *covariance function* $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$.

$$m(\mathbf{x}) = \mathbb{E}_{f\sim F}\left[f(\mathbf{x})\right]$$
$$k(\mathbf{x}_1, \mathbf{x}_2) = \mathbb{E}_{f\sim F}\left[(f(\mathbf{x}_1) - m(\mathbf{x}_1))(f(\mathbf{x}_2) - m(\mathbf{x}_2))\right]$$

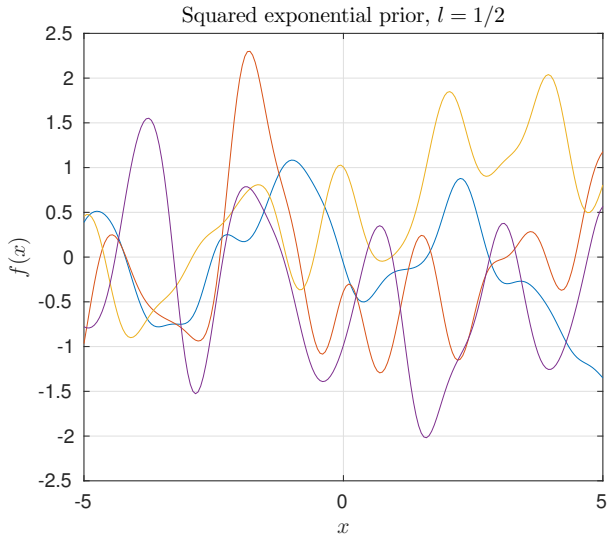
We then write

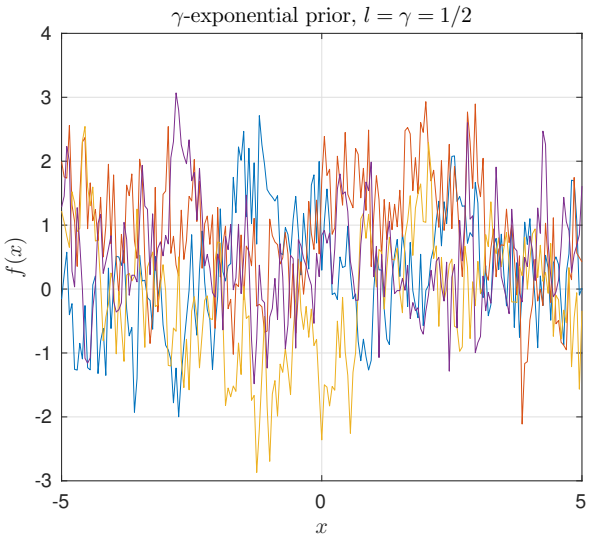
$$F \sim \mathrm{GP}(m, k)$$

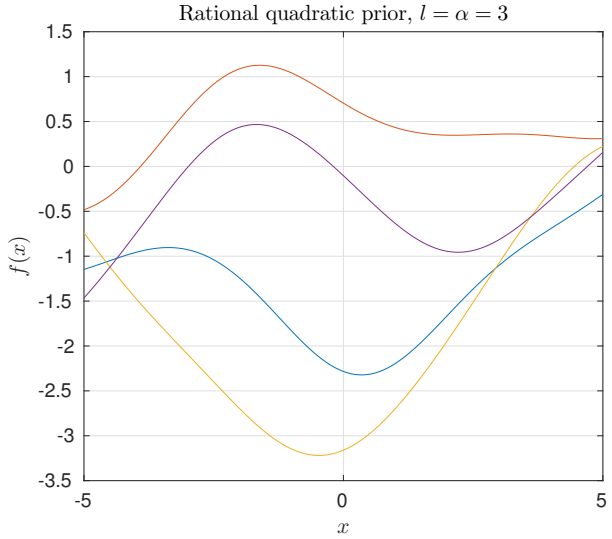
to denote that F is a GP.

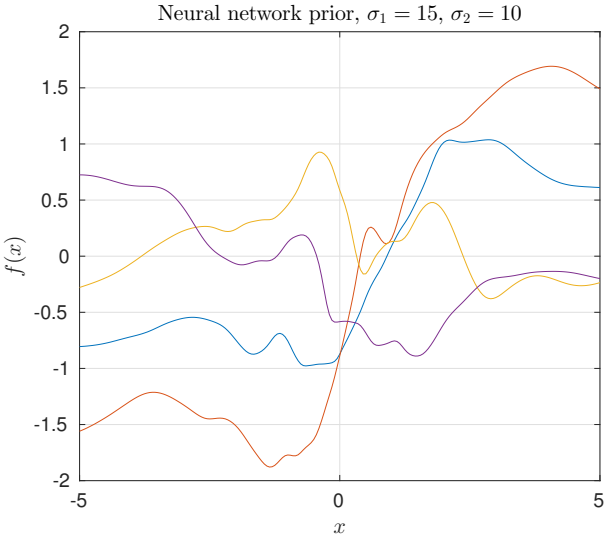
By specifying m and k we get different kinds of function when sampling F.

GP priors


Squared exponential prior, $l = 1/2$
$f(x)$
x


γ-exponential prior, $l = \gamma = 1/2$
$f(x)$
x


Rational quadratic prior, $l = \alpha = 3$
$f(x)$
x


Neural network prior, $\sigma_1 = 15$, $\sigma_2 = 10$
$f(x)$
x

Covariance functions

Polynomial;

$$k(\mathbf{x}_1, \mathbf{x}_2) = (c + \mathbf{x}_1^T \mathbf{x}_2)^k$$

Exponential:

$$k(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\frac{|\mathbf{x}_1 - \mathbf{x}_2|}{l}\right)$$

Squared exponential:

$$k(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\frac{|\mathbf{x}_1 - \mathbf{x}_2|^2}{2l^2}\right)$$

Gamma exponential:

$$k(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\left(\frac{|\mathbf{x}_1 - \mathbf{x}_2|}{l}\right)^\gamma\right)$$

Covariance functions

Rational quadratic;

$$k(\mathbf{x}_1, \mathbf{x}_2) = \left(1 + \frac{|\mathbf{x}_1 - \mathbf{x}_2|^2}{2\alpha l^2}\right)^{-\alpha}$$

Exponential:

$$k(\mathbf{x}_1, \mathbf{x}_2) = \sin^{-1}\left(\frac{2(\mathbf{x}_1')^T \boldsymbol{\Sigma} \mathbf{x}_2'}{((1 + 2(\mathbf{x}_1')^T \boldsymbol{\Sigma} \mathbf{x}_1')(1 + 2(\mathbf{x}_2')^T \boldsymbol{\Sigma} \mathbf{x}_2'))^{1/2}}\right)$$

where $(\mathbf{x}')^T = \begin{bmatrix} 1 & \mathbf{x}^T \end{bmatrix}$.

As usual these have associated *hyperparameters*.

These have to be dealt with correctly as always.

Gaussian processes: generating data

Say we have some data

$$y_i = f(\mathbf{x}_i)$$

for $i = 1, \ldots, m$ and $f \sim \mathrm{GP}(m, k)$. (Remember, the $\mathbf{x}_i$ are *fixed*, not RVs.)

Any *finite set of points* must be *jointly Gaussian*. So

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{m}, \mathbf{K})$$

where

$$\mathbf{m}^T = \begin{bmatrix} m(\mathbf{x}_1) & \cdots & m(\mathbf{x}_m) \end{bmatrix}$$

and $\mathbf{K}$ is the *Gram matrix* $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

Note 1: this is *not* $p(\mathbf{y}|f)$. We can completely remove the need for integration!

Note 2: from now on we will assume $m(\mathbf{x}) = 0$. (It is straightforward to incorporate a non-zero mean.)

Gaussian processes: generating data with noise

Now add noise to the data.

Say we add Gaussian noise so

$$y_i = f(\mathbf{x}_i) + \epsilon_i.$$

Again, $i = 1, \ldots, m$ and $f \sim \mathrm{GP}(m, k)$, but now we also have

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2).$$

As we are *adding Gaussian RVs*, we have

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{0}, \mathbf{K} + \sigma^2 \mathbf{I}).$$

BUT: in order to do *prediction* we actually need to involve a new point $\mathbf{x}'$, for which we want to predict the corresponding value y'.

Gaussian processes: prediction

SO: we incorporate $\mathbf{x}'$, for which we want to predict the corresponding value y'.

By exactly the same argument

$$p(y', \mathbf{y}) = \mathcal{N}(\mathbf{0}, \mathbf{K}')$$

where

$$\mathbf{K}' = \begin{bmatrix} k & \mathbf{k}^T \\ \mathbf{k} & \mathbf{K} + \sigma^2 \mathbf{I} \end{bmatrix}$$

$$\mathbf{k}^T = \begin{bmatrix} k(\mathbf{x}, \mathbf{x}_1) & \cdots & k(\mathbf{x}, \mathbf{x}_m) \end{bmatrix}$$

$$k = k(\mathbf{x}, \mathbf{x}) + \sigma^2.$$

Note 1: all we've done here is to expand the Gram matrix by an extra row and column to get $\mathbf{K}'$.

Note 2: whether or not you include σ^2 in k is a matter of choice. What difference does it make? (This is an *Exercise*.)

Gaussian density: marginals and conditionals

For a normal RV $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right).$$

Split $\mathbf{x}$ so

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}$$

and correspondingly

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}.$$

What are $p(\mathbf{x}_1)$ and $p(\mathbf{x}_1|\mathbf{x}_2)$?

Gaussian density: marginals and conditionals

Define the *precision matrix*

$$\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1} = \begin{bmatrix} \boldsymbol{\Lambda}_{11} & \boldsymbol{\Lambda}_{12} \\ \boldsymbol{\Lambda}_{21} & \boldsymbol{\Lambda}_{22} \end{bmatrix}.$$

It is possible to show that

$$p(\mathbf{x}_1) = \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11})$$
$$p(\mathbf{x}_1|\mathbf{x}_2) = \mathcal{N}(\boldsymbol{\mu}_1 - \boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\Lambda}_{12}(\mathbf{x}_2 - \boldsymbol{\mu}_2), \boldsymbol{\Lambda}_{11}^{-1}).$$

Inverting a block matrix

In the last slide, we see:

$$\boldsymbol{\Sigma}^{-1} = \begin{bmatrix} \boldsymbol{\Lambda}_{11} & \boldsymbol{\Lambda}_{12} \\ \boldsymbol{\Lambda}_{21} & \boldsymbol{\Lambda}_{22} \end{bmatrix}.$$

Re-writing $\boldsymbol{\Sigma}$ as

$$\boldsymbol{\Sigma} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}$$

it is possible to show (it is an *Exercise* to do this) that

$$\begin{aligned} \boldsymbol{\Lambda}_{11} &= \mathbf{A}' \\ \boldsymbol{\Lambda}_{12} &= -\mathbf{A}'\mathbf{B}\mathbf{D}^{-1} \\ \boldsymbol{\Lambda}_{21} &= -\mathbf{D}^{-1}\mathbf{C}\mathbf{A}' \\ \boldsymbol{\Lambda}_{22} &= \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{C}\mathbf{A}'\mathbf{B}\mathbf{D}^{-1} \end{aligned}$$

where

$$\mathbf{A}' = (\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}.$$

GP regression

To do *prediction* all that's left is to compute $p(y'|\mathbf{y})$.

Because everything is Gaussian this turns out to be easy:

$$p(y', \mathbf{y}) = \mathcal{N}(\mathbf{0}, \mathbf{K}')$$

$$\mathbf{K}' = \begin{bmatrix} k & \mathbf{k}^T \\ \mathbf{k} & \mathbf{L} \end{bmatrix}$$

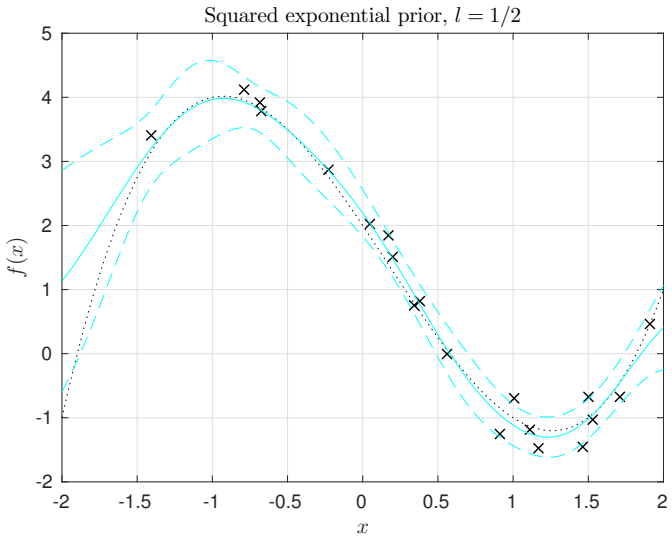
$$\mathbf{L} = \mathbf{K} + \sigma^2 \mathbf{I}.$$

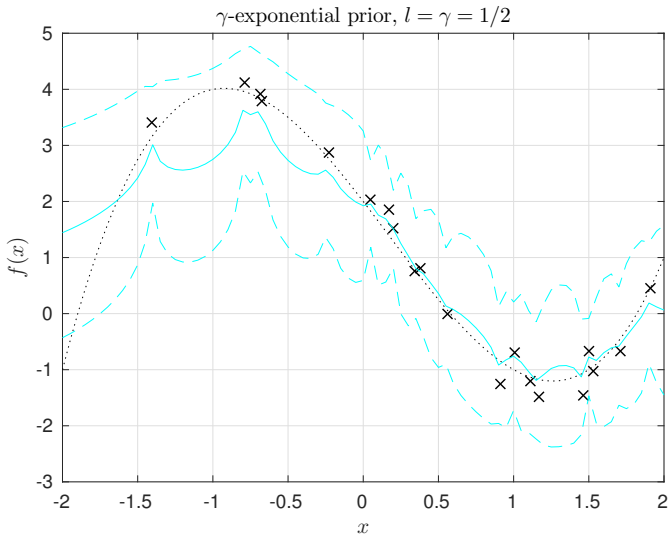
From these we want to know $p(y'|\mathbf{y})$.

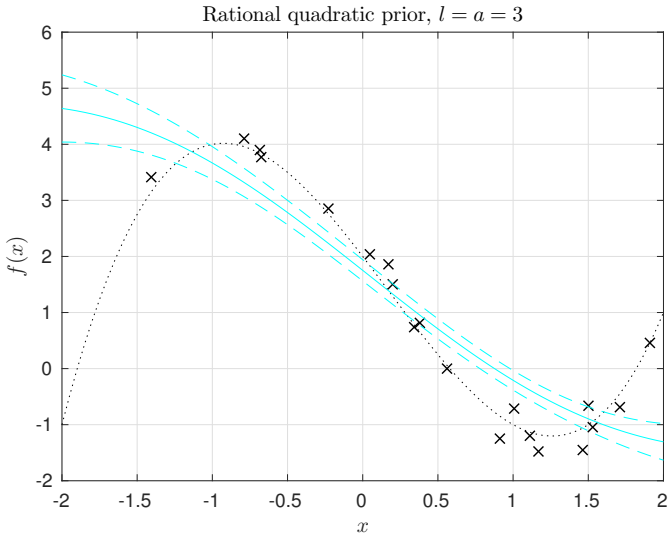
Only two things are needed: the *inverse formula* for a block matrix and the *formula for obtaining a conditional* from a joint Gaussian. Using these we can show (it is an *Exercise* to derive this) that

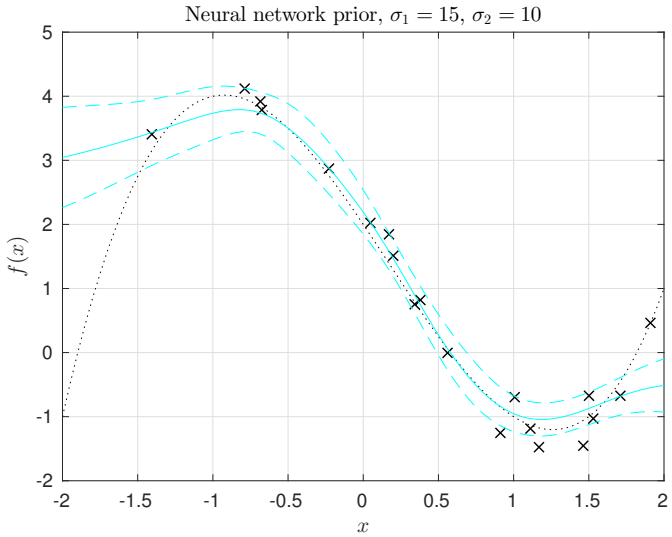
$$p(y'|\mathbf{y}) = \mathcal{N}(\underbrace{\mathbf{k}^T \mathbf{L}^{-1} \mathbf{y}}_{\text{Mean}}, \underbrace{k - \mathbf{k}^T \mathbf{L}^{-1} \mathbf{k}}_{\text{Variance}}).$$

GP regression









Learning the hyperparameters

A nice side-effect of this formulation is that we get *a usable expression for the marginal likelihood.*

If we incorporate the hyperparameters $\mathbf{p}$, which in this case are any parameters associated with $k(\mathbf{x}_1, \mathbf{x}_2)$ along with σ^2, then we've just computed

$$p(y'|\mathbf{y}, \mathbf{p}) = \frac{p(y', \mathbf{y}|\mathbf{p})}{p(\mathbf{y}|\mathbf{p})}.$$

The denominator is the marginal likelihood, and we computed it above on *slide 44*:

$$p(\mathbf{y}|\mathbf{p}) = \mathcal{N}(\mathbf{0}, \mathbf{L}) = \frac{1}{\sqrt{(2\pi)^m |L|}} \exp\left(-\frac{1}{2}\mathbf{y}^T \mathbf{L}^{-1} \mathbf{y}\right).$$

Learning the hyperparameters

As usual this looks nicer if we consider its log

$$\log p(\mathbf{y}|\mathbf{p}) = -\frac{1}{2}\log|\mathbf{L}| - \frac{1}{2}\mathbf{y}^T\mathbf{L}^{-1}\mathbf{y} - \frac{d}{2}\log 2\pi.$$

This is a *rare beast*:

1. It's a sensible formula that tells you how good a set $\mathbf{p}$ of hyperparameters is.
2. That means you can use it as an *alternative to cross-validation* to search for hyperparameters.
3. As a bonus *you can generally differentiate it* so it's possible to use *gradient-based search*.