MLPR Tutorial Sheet 6 (Answers)

Reminder: If you need more guidance to get started on a question, seek clarifications and hints on the class forum. Move on if you’re getting stuck on a part for a long time. Full answers will be released after the last group meets.

1. More practice with Gaussians:

$N$ noisy independent observations are made of an unknown scalar quantity $m$:

$$x^{(n)} \sim N(m, \sigma^2).$$

a) I don’t give you the raw data, $\{x^{(n)}\}$, but tell you the mean of the observations:

$$\bar{x} = \frac{1}{N} \sum_{n=1}^{N} x^{(n)}.$$  

What is the likelihood$^1$ of $m$ given only this mean $\bar{x}$? That is, what is $p(\bar{x} \mid m)$?  

**Answer:**

The mean of some Gaussian outcomes, like any linear combination, is Gaussian distributed. So we just have to identify the mean and variance of this derived quantity. The mean:

$$E[\bar{x}] = \frac{1}{N} \sum_{n=1}^{N} E[x^{(n)}] = \frac{1}{N} \sum_{n=1}^{N} m = m.$$ 

The variance:

$$\text{var}[\bar{x}] = \frac{1}{N^2} \text{var} \left[ \sum_{n=1}^{N} x^{(n)} \right] = \frac{1}{N^2} \left[ N\sigma^2 \right] = \frac{\sigma^2}{N}.$$  

where variances add because the noisy outcomes are independent. So:

$$p(\bar{x} \mid m) = N(\bar{x}; m, \sigma^2 / N).$$

b) A **sufficient statistic** is a summary of some data that contains all of the information about a parameter.

i) Show that $\bar{x}$ is a sufficient statistic of the observations for $m$, assuming we know the noise variance $\sigma^2$. That is, show that $p(m \mid \bar{x}) = p(m \mid \{x^{(n)}\}_{n=1}^{N})$.

ii) If we don’t know the noise variance $\sigma^2$ or the mean, is $\bar{x}$ still a sufficient statistic in the sense that $p(m \mid \bar{x}) = p(m \mid \{x^{(n)}\}_{n=1}^{N})$? Explain your reasoning.

**Answer:**

1. I’m using the traditional statistics usage of the word “likelihood”: it’s a function of parameters given data, equal to the probability of the data given the parameters. Personally I avoid saying “likelihood of the data” (Cf p29 of MacKay’s textbook), although you’ll see that usage too.

2. The sum of Gaussian outcomes is Gaussian distributed; you only need to identify a mean and variance.
i) The posterior given the observations comes from Bayes’ rule:

\[ p(m \mid \{x^{(n)}\}_{n=1}^N) \propto p(m) p(\{x^{(n)}\}_{n=1}^N \mid m) \]

\[ \propto p(m) \prod_n \mathcal{N}(x^{(n)}; m, \sigma^2) \]

\[ \propto p(m) \exp \left( -\frac{1}{2\sigma^2} \sum_n (x^{(n)} - m)^2 \right) \]

\[ \propto p(m) \exp \left( -\frac{Nm^2}{2\sigma^2} + m \sum_n x^{(n)} \right) . \]

The proportionality sign, ‘\(\propto\)’, means that I can drop any multiplicative term that doesn’t depend on \(m\).

Given just the mean \(\bar{x}\) the posterior would be:

\[ p(m \mid \bar{x}) \propto p(m) p(\bar{x} \mid m) \]

\[ \propto p(m) \mathcal{N}(\bar{x}; m, \sigma^2/N) \]

\[ \propto p(m) \exp \left( -\frac{N}{2\sigma^2} (\bar{x} - m)^2 \right) \]

\[ \propto p(m) \exp \left( -\frac{Nm^2}{2\sigma^2} + \frac{mN\bar{x}}{\sigma^2} \right) , \]

which, substituting the definition of \(\bar{x}\), is equal to the posterior given all of the observations. Therefore \(\bar{x}\) is a sufficient statistic. Our beliefs about the parameter \(m\) given just \(\bar{x}\) are identical to those given the whole dataset.

ii) The working above assumed that we knew the variance \(\sigma^2\). If we don’t know \(\sigma^2\) then \(\bar{x}\) is not sufficient for computing the posterior over the mean. Counter example: if all the \(x^{(n)}\) samples were identical, then we would infer that the variance is zero and have a delta-function posterior over the mean. If the \(x^{(n)}\) samples had high variance (which they could have with the same mean \(\bar{x}\)), we’d realize the variance is larger, and not be certain about the mean. We can still use \(\bar{x}\) as a point-estimate of the mean, it just doesn’t tell us the whole posterior distribution.

**Purpose of Q1:**

Computing the posterior and predictions in Bayesian linear regression and Gaussian processes are just examples of dealing with noisy observations of linear combinations of Gaussian variables. We can do a lot of useful statistical reasoning if we know how to deal with Gaussians, so it’s worth practicing manipulating them. This question was meant to contain some easier manipulations than some of the ones we’ve seen in lectures.

---

2. **Conjugate priors:** (This question sets up some intuitions about the larger picture for Bayesian methods. But if you’re finding the course difficult, look at Q3 first.)

A **conjugate prior** for a likelihood function is a prior where the posterior is a distribution in the same family as the prior. For example, a Gaussian prior on the mean of a Gaussian distribution is conjugate to Gaussian observations of that mean.

a) The **inverse-gamma distribution** is a distribution over positive numbers. It’s often used to put a prior on the variance of a Gaussian distribution, because it’s a conjugate prior.
The inverse-gamma distribution has pdf (as cribbed from Wikipedia):
\[
p(z \mid \alpha, \beta) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} z^{-\alpha-1} \exp\left(-\frac{\beta}{z}\right), \quad \text{with } \alpha > 0, \ \beta > 0,
\]
where \(\Gamma(\cdot)\) is a gamma function.\(^3\)

Assume we obtain \(N\) observations from a zero-mean Gaussian with unknown variance,
\[
x^{(n)} \sim \mathcal{N}(0, \sigma^2), \quad n = 1 \ldots N,
\]
and that we place an inverse-gamma prior with parameters \(\alpha\) and \(\beta\) on the variance. Show that the posterior over the variance is inverse-gamma, and find its parameters.

Hint: you can assume that the posterior distribution is a distribution; it normalizes to one. You don’t need to keep track of normalization constants, or do any integration. Simply show that the posterior matches the functional form of the inverse-gamma, and then you know the normalization (if you need it) by comparison to the pdf given.

**Answer:**
The posterior is given by Bayes’ rule. I’m going to write the variance as \(\upsilon = \sigma^2\) so the square won’t confuse me when comparing coefficients.

\[
p(\upsilon \mid \{x^{(n)}\}, \alpha, \beta) \propto \upsilon^{-\alpha-1} \exp\left(-\frac{\beta}{\upsilon}\right) \prod_n \mathcal{N}(x^{(n)}; 0, \upsilon)
\]
\[
\propto \upsilon^{-\alpha-1} \left(\frac{\beta}{\upsilon}\right)^{-N/2} \exp\left(-\frac{1}{2\upsilon} \sum_n (x^{(n)})^2\right)
\]
\[
\propto \upsilon^{-(\alpha+N/2)-1} \exp\left(-\frac{\beta + \frac{1}{2} \sum_n (x^{(n)})^2}{\upsilon}\right)
\]
\[
= \text{inverse-gamma}\left[\upsilon; \alpha + N/2, \beta + \frac{1}{2} \sum_n (x^{(n)})^2\right].
\]

The posterior is also inverse-gamma, so the prior is conjugate to the likelihood. Some people aren’t happy with the step to the final line. The distribution has only been written down up to a constant. But that constant is specified: we know the distribution must integrate to one. Therefore the pdf can only be one function, and that function matches the standard form of the inverse-gamma distribution.

Warning: some books and library routines parameterize distributions differently, so you have to be sure that \(\alpha\) and \(\beta\) mean what you think they do before using them with someone else’s result or function.

b) i) If a conjugate prior exists, then the data can be replaced with sufficient statistics. Can you explain why?

ii) Explain whether there could be a conjugate prior for the hard classifier:
\[
P(y=1 \mid x, w) = \Theta(w^\top x + b) = \begin{cases} 1 & w^\top x + b > 0 \\ 0 & \text{otherwise.} \end{cases}
\]

This question is intended as a tutorial discussion point. It might be hard to write down a mathematically rigorous argument. But can you explain

---

3. Numerical libraries often come with a `gammaln` or `lgamma` function to evaluate the log of the gamma function.
whether it is easy to represent beliefs about the weights of a classifier in a fixed-size statistic, regardless of what data you gather? A picture may help.

Answer:

i) When we have a conjugate prior, it means that the posterior, like the prior, can be expressed as some neat standard distribution. The parameters of that distribution are enough to represent all of our beliefs about the parameters given the data. Therefore these parameters are sufficient statistics: given them, we don’t need to store the data any more.

For completeness, the parameters of the conjugate posterior distribution are sufficient statistics, even if you don’t want to use that conjugate prior. Notation: parameters \( \theta \), data \( D \), assumption that we should use a conjugate prior \( C \), and assumption we should use another prior \( M \). The posterior given a conjugate prior,

\[
p(\theta \mid D, C) = \frac{p(D \mid \theta) p(\theta \mid C)}{p(D \mid C)},
\]

is available in a compact form. The posterior given another prior is:

\[
p(\theta \mid D, M) \propto p(D \mid \theta) p(\theta \mid M) \propto \frac{p(\theta \mid D, C)}{p(\theta \mid C)} p(\theta \mid M),
\]

So the parameters of the conjugate prior and posterior distributions summarize everything we need to know about the data to construct beliefs about the parameters. (As long as we haven’t divided by zero. The conjugate prior \( p(\theta \mid C) \) better have full support over the parameters.)

ii) No, there isn’t a compact conjugate prior for the classifier, because there isn’t a compact way to describe the posterior.

I didn’t expect you to come up with all the details in this answer. However, I do want you to build some intuitions about the posterior distribution over classifiers, as we’ll be talking about how to approximate it later.

Below is a sketch of some extreme decision boundaries given some binary labels (crosses and circles) in a 2D feature space:

Each of these boundaries is defined by two datapoints, and a small move could make them assign zero probability to the data. Given a uniform prior, the posterior would be uniform over a polygon in parameter space (in higher dimensions a “polytope”). The parameters of the boundaries that I’ve drawn give the vertices of this polygon.
Not all of the data-points are relevant. There are four in the picture that could be removed without changing the posterior distribution. However, in general, especially in higher dimensions, the boundary of the posterior could get more and more complicated as we see more datapoints. There is no fixed size representation of a polygon with an arbitrary number of vertices.

Only in 1D would we guarantee a compact posterior. Given features:

\[ \times \times \times \times \times \circ \circ \circ \circ \circ \circ \circ \circ \circ \circ \]

we just need to note where the central \( \times \) and \( \circ \) are. If there aren’t two clearly central points, because the data aren’t linearly separable, the posterior isn’t well defined for the hard classifier model.

For keen students: The rest of this indented part contains more background for those who are interested. Probably not for discussion in tutorials. Some distributions with compact sufficient statistics and conjugate priors are exponential family distributions:

\[
p(x | \theta) = \frac{1}{Z(\theta)} g(x) e^{\theta^T \phi(x)},
\]

where \( \phi(x) \) is a vector of statistics extracted from the variable \( x \). The \( g(\cdot) \) function is arbitrary, often specifying constraints, and \( Z(\theta) \) is the normalization constant\(^4\). The likelihood given multiple observations \{\( x^{(n)} \)\} is then a function of the parameters with a simple dependence on the data:

\[
L(\theta) \propto \prod_n \frac{1}{Z(\theta)} e^{\theta^T \phi(x^{(n)})} \propto \frac{1}{Z(\theta)^N} e^{\theta^T \left[ \sum \phi(x^{(n)}) \right]}. 
\]

The fixed-length vector \( \sum_n \phi(x^{(n)}) \) is a sufficient statistic of the dataset. Unless the observation model can be written as an exponential family distribution, the statistics of the data usually don’t add up as above, and we don’t get compact sufficient statistics or conjugate priors. However, Fink’s (1997) compendium of conjugate priors gives other cases that work out neatly.

It’s tempting to think that logistic regression is in the exponential family. (I’ll admit that I’ve found this idea quite confusing, so you might too!) After all, a binary label \( z \in \pm 1 \) has probability

\[
p(z^{(n)} | w, x^{(n)}) \propto e^{w^T (z^{(n)} x^{(n)})}.
\]

However, each datapoint has a different \( x^{(n)} \), so uses a different vector of statistics: we don’t have a simple statistic \( \phi(z) \). If we want to model the joint distribution \( p(z^{(n)}, x^{(n)} | w) \), we’d need to include the normalization for \( p(z^{(n)} | w, x^{(n)}) \), which depends on \( x^{(n)} \) and \( w \), and again we wouldn’t be in the exponential family.

Lessons from Q2:

When we fit parametric models we use the data to set some parameters, and then throw the data away at test time. In the Bayesian approach to prediction, the posterior distribution is the summary of what we have learned from the data. The posterior is only simple to represent (exactly) for conjugate models. In this course we mainly focus on Gaussian models, but there are some other convenient conjugate distributions.

Later in the course we will cover Bayesian prediction with logistic regression (which is related to the linear classifier in this question). It is helpful to have some intuition about the

\[ A(\theta) = -\log Z(\theta) \]

inside the exponent. Strictly, I have only written the form of the “canonical” exponential family, and there is a more general form.

---

4. The exponential family is also often written with \(-A(\theta) = -\log Z(\theta)\) inside the exponent. Strictly, I have only written the form of the “canonical” exponential family, and there is a more general form.
3. **Regression with input-dependent noise:**

In lectures we turned a representation of a function into a probabilistic model of real-valued outputs by modelling the residuals as Gaussian noise:

\[ p(y \mid x, \theta) = \mathcal{N}(y; f(x; \theta), \sigma^2). \]

The noise variance \( \sigma^2 \) is often assumed to be a constant, but it could be a function of the input location \( x \) (a “heteroscedastic” model).

A flexible model could set the variance using a neural network:

\[ \sigma(x)^2 = \exp(w^{(\sigma)}\top h(x; \theta) + b^{(\sigma)}), \]

where \( h \) is a vector of hidden unit values. These could be hidden units from the neural network used to compute function \( f(x; \theta) \), or there could be a separate network to model the variances.

a) Assume that \( h \) is the final layer of the same neural network used to compute \( f \). How could we modify the training procedure for a neural network that fits \( f \) by least squares, to fit this new model?

**Answer:**

Probabilistic models are often fitted by maximum likelihood (perhaps with regularization), where the cost function to minimize is the negative log-likelihood. (Other cost functions are possible.) I’ll show how we find the derivatives of the cost for one example. These could be used in stochastic gradient descent. We could add up the gradients over a batch of examples and use a batch or mini-batch gradient-based optimizer.

I’ll write the variance as \( v = \sigma^2 \), so the square doesn’t confuse me when differentiating:

\[ c = -\log \mathcal{N}(y; f, v) = \frac{1}{2} \log v + \frac{1}{2v} (y - f)^2 + \frac{1}{2} \log(2\pi). \]

We can work out the effect on the cost of an example that perturbing the mean or variance would have:

\[ \bar{f} = \frac{\partial c}{\partial f} = \frac{1}{v} (f - y), \]

\[ \bar{\sigma} = \frac{\partial c}{\partial \sigma} = \frac{1}{2v} - \frac{1}{2v^2} (y - f)^2 \]

These signals are then backpropagated to earlier stages of the computation.

I’ll write that:

\[ f = w^{(f)}\top h^{(f)} + b^{(f)} \]
\[ a^{(\sigma)} = w^{(\sigma)}\top h^{(\sigma)} + b^{(\sigma)} \]
\[ v = \exp(a^{(\sigma)}) \]

where \( h^{(f)} \) is the final hidden layer before computing the function, which the least squares neural network will also have. \( h^{(\sigma)} \) is a hidden layer used to compute the new variance prediction. Later I assume that the network shares the same hidden layer for both outputs (a common choice, but not necessary): \( h^{(\sigma)} = h^{(f)} = h^{(L)} \).
We backpropagate the variance computation one step:
\[ \bar{a}^{(σ)} = \bar{v} \exp(\bar{a}^{(σ)}) = \bar{v}. \]

Standard results for backpropagation apply to a generic scalar neural network activation as follows:
\[ a = w^\top h + b \quad \Rightarrow \quad \bar{w} = \bar{a}h, \quad \bar{h} = \bar{a}w, \quad \bar{b} = \bar{a}. \]

These rules apply to the variance activation:
\[ \bar{w}^{(σ)} = \bar{a}^{(σ)} h^{(σ)}, \quad \bar{h}^{(σ)} = \bar{a}^{(σ)} w^{(σ)}, \quad \bar{b}^{(σ)} = \bar{a}^{(σ)} \]
and the function value \( f \) (a least squares network also does this part):
\[ \bar{w}^{(f)} = \bar{f} h^{(f)}, \quad \bar{h}^{(f)} = \bar{f} w^{(f)}, \quad \bar{b}^{(f)} = \bar{f}. \]

We now have the gradients for all of the weights producing the function and variance at the final stage of the network.

If we assume that the hidden layers for the variance and function are shared (the same hidden layer), we can combine the signals for the shared final hidden layer:
\[ \bar{h}^{(L)} = \bar{h}^{(σ)} + \bar{h}^{(f)} \]
and then propagate that back through whatever the neural network is, exactly as the least squares network did. If we don’t share the hidden layers, we’d have to describe where \( h^{(σ)} \) comes from, and how any extra parameters are fitted.

My empirical experience suggests that fitting the neural network in this question is harder than a least squares network. It’s harder to find good learning rates for example.

There are other possible ways to predict the typical size of a neural network’s mistake. For example we could compute the log of the square residuals on the training set, and then fit a second neural network (using the same least squares code as used to fit the function) to fit the log square residuals.

b) In the suggestion above, the activation \( a^{(σ)} = w^{(σ)} \top h + b^{(σ)} \) sets the log of the variance of the observations.

i) Why not set the variance directly to this activation value, \( σ^2 = a^{(σ)} \)?

ii) Harder (I don’t know if you’ll have an answer, but I’m curious to find out): Why not set the variance to the square of this activation value, \( σ^2 = (a^{(σ)})^2 \)?

Answer:

i) A neural network activation is unconstrained, but the model is undefined for negative variances.

ii) Setting \( σ^2 = a^2 \) would mean that the variances were non-negative, and so in principle this idea could work. But it’s not likely to work well in practice. The variance no longer changes monotonically as a function of the activation. Moreover, as the activation passes through zero, the cost function passes
down towards minus infinity and up again, with extreme gradients. There will probably be worse local optima than with the suggested architecture, and gradient fitting procedures are more likely to become unstable.

Taking the log of the variance puts problematic zero variances out at activations of negative infinity.

c) Given a test input \( x^{(*)} \), the model above outputs both a guess of an output, \( f(x^{(*)}) \), and an ‘error bar’ \( \sigma(x^{(*)}) \), which indicates how wrong the guess could be.

The Bayesian linear regression and Gaussian process models covered in lectures also give error bars on their predictions. What are the pros and cons of the neural network approach in this question? Would you use this neural network to help guide which experiments to run?

Answer:

**Model flexibility:** The neural network model in this question is “heteroscedastic”, it has input-dependent noise in the model. Standard Bayesian linear regression and GP models (although they could be generalized) assume fixed noise. Given lots of data, these models will make predictions with the same variance everywhere, even if that’s the wrong thing to do.

**Costs:** It’s easier to fit the neural network model to large datasets than Gaussian processes. One sweep through \( N \) datapoints by stochastic gradient descent with \( H \) hidden units and \( D \)-dimensional inputs costs \( O(NDH) \). Gaussian processes cost \( O(N^3) \). Bayesian linear regression is cubic in the number of basis functions, and it’s hard to cover high-dimensional input spaces. For smaller datasets, the closed-form Gaussian computations of GPs and linear regression are more attractive.

**Uncertainties:** The Bayesian approaches explicitly model uncertainty in the parameters separately from the noise of the observation process. With Gaussian processes, the error bars grow to the prior function standard deviation for inputs far away from any observations. In the neural network, the variance could extrapolate in arbitrary ways in unseen regions. Some thought could go into regularizing it towards sensible solutions — and Bayesian versions of neural networks are possible. However, Gaussian processes are usually the first choice for experiment design, with neural networks adopted carefully for large problems.

You might have thought of other things… if so, please add them to the Forum.