CS 181 Spring 2020 Section 4 Notes: Bayesian Approaches, Neural Networks

1 Bayesian Regression

1.1 Motivations

The Bayesian frame of reference helps us answer three types of questions regarding data X and labels Y:

- The **posterior** over models: $p(\theta|X,Y) \propto p(Y|X,\theta)P(\theta|X)$
  This tells us what values of the model θ maximize the likelihood of the observed data.
- The **posterior predictive** for new data: $p(y^*|x^*) = \int p(y^*|x^*,\theta)p(\theta|X,Y)$
  This tells us how to predict the label of a new data point, given the observed data.
- The **marginal likelihood** of data: $p(Y|X) = \int p(Y|X,\theta)p(\theta)d\theta$
  This tells us how likely the data is, marginalizing over possible models θ. This is different from the posterior in that the posterior optimizes the model we've selected, while the marginal likelihood allows us to compare models in terms of how well they fit the data (this is model selection).

1.2 Conjugate Pairs

Focusing in on the concept of posteriors, note that $p(\theta)$ represents our prior beliefs regarding the optimal model θ, and $p(\theta|X,Y)$ represents our updated beliefs after observing some data X,Y. There is no guarantee that $p(\theta)$ and $p(\theta|X,Y)$ share a nice, clean relationship, but sometimes they do, thanks to some special structure in the distribution of labels $p(Y|X,\theta)$. When a certain model can represent $p(\theta)$ and $p(\theta|X,Y)$ with the same distribution, we call this a **conjugacy pair**.

1.2.1 Beta-Binomial Conjugacy

One example, seen in lecture, occurs when we model the prior distribution $p(\theta)$ using a Beta distribution, and we model the data $p(Y|X,\theta 0$ according to a Bernoulli (or Binomial) distribution. In this case, the posterior distribution $p(\theta|X,Y)$ follows a Beta distribution, just like the prior. Note that the parameter values may have changed, but the class of the distribution is the same!

1.3 Bayesian Linear Regression

Let $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^m$, $y_i \in \mathbb{R}$. Consider the generative model:

$$y_i \sim \mathcal{N}(\mathbf{w}^\top \mathbf{x}_i, \beta^{-1}) \tag{1}$$

The likelihood of the data has the form:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{w}, \beta^{-1}\mathbf{I}) \tag{2}$$

Put a conjugate prior on the weights (assume precision β^{-1} known):

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0) \tag{3}$$

We want a posterior distribution on $\mathbf{w}$. Using Bayes' Theorem:

$$p(\mathbf{w}|D) \propto p(D|\mathbf{w})p(\mathbf{w}) \tag{4}$$

It turns out that our posterior after n examples is also Gaussian:

$$p(\mathbf{w}|D) = \mathcal{N}(\mathbf{w}|\mathbf{m}_n, \mathbf{S}_n) \tag{5}$$

where

$$\mathbf{S}_n = \left(\mathbf{S}_0^{-1} + \beta\mathbf{X}^\top\mathbf{X}\right)^{-1} \tag{6}$$

$$\mathbf{m}_n = \mathbf{S}_n(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\mathbf{X}^\top\mathbf{y}) \tag{7}$$

This tells us that Gaussian-Gaussian is yet another example of a conjugacy pair.

1.4 Concept Question

Why do we care about conjugacy pairs? What makes them convenient to study?

When the prior distribution and posterior distribution are different, it becomes very difficult to model updates in our beliefs as we see new data. Conjugacy pairs are remarkably helpful because they allow us to safely assume that the class of distributions doesn't change as we see new data. We can instead focus on adjusting the parameters of such a distribution and thus achieve much deeper insights regarding a model.

1.5 Posterior Predictive Distributions

We have seen how to obtain a posterior distribution over $\mathbf{w}$. But, given this posterior and a new data point $\mathbf{x}^*$, how do we actually make a prediction y^*? How do we deal with *uncertainty* about $\mathbf{w}$? Consider this:

$$p(y^*|\mathbf{x}^*, \mathcal{D}) = \int_{\mathbf{w}} p(y^*|\mathbf{x}^*, \mathbf{w})p(\mathbf{w}|\mathcal{D})d\mathbf{w} \tag{8}$$

$$= \int_{\mathbf{w}} \mathcal{N}(y^*|\mathbf{w}^\top\mathbf{x}^*, \beta^{-1})\mathcal{N}(\mathbf{w}|\mathbf{m}_n, \mathbf{S}_n)d\mathbf{w} \tag{9}$$

This is the **posterior predictive** distribution over y^*. This can be interpreted as a weighted average of many predictors, one for each choice of $\mathbf{w}$, weighted by how likely $\mathbf{w}$ is according to the posterior. Since each of the terms on the right hand side follows a normal distribution, we can use some math to find that:

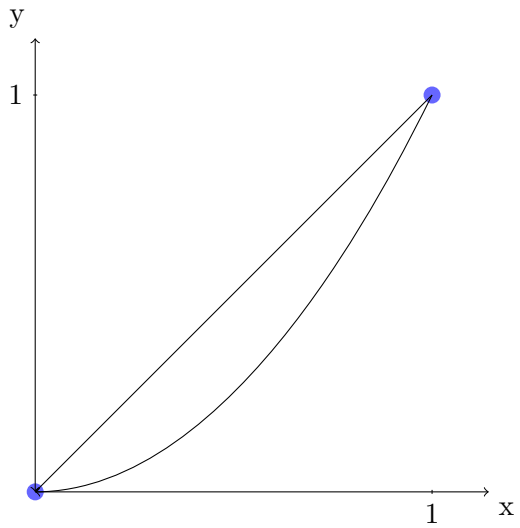
$$p(y^*|\mathbf{x}^*, \mathcal{D}) = \mathcal{N}(y^*|\mathbf{x}^{*\top}\mathbf{m}_n, \mathbf{x}^{*\top}\mathbf{S}_n\mathbf{x}^* + \beta^{-1}) \tag{10}$$

1.6 Exercise: A Simple Bayesian Model

Say you are tasked with fitting a parabolic regression $\hat{y} = a_0 + a_1 x + a_2 x^2$ on data of the form (x, y) for feature x and label y. That is, you want to find the best possible a_0, a_1, a_2 to fit the data you are given.

1. Before seeing any data, what are reasonable prior distributions for the parameters a_0, a_1, a_2?

   You are now presented with two data points, $(0, 0)$ and $(1, 1)$. You are told that the data was generated from some $y = f(x)$ with negligible error, that is $\epsilon \sim \mathcal{N}(0, \sigma^2)$ where $\sigma^2 \approx 0$.



2. Given $\sigma^2 \approx 0$, does the actual function $y = f(x)$ go through the points $(0, 0)$ and $(1, 1)$? Using this information, what relationships must exist between a_0, a_1, a_2? Update your prior distributions on a_0, a_1, a_2 to become posterior distributions.

3. You are now told that $a_1 \sim \text{Unif}(-1, 1)$. Given this and the data you've observed, if a new data point $x^* = 1/2$ is presented, what is the posterior predictive on y^*?

4. What would the posterior predictive on y^* have been for a linear regression $\hat{y} = a_0 + a_1 x$ instead? Compare the strengths of the two models' predictions for y^*. What does this say about model selection?

Solution:

1.7 Exercise: Posterior Distribution By Completing the Square (Bishop 3.7)

We know from (3.10) in Bishop that the likelihood can be written as

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \mathbf{w}) &= \prod_{i=1}^{n} \mathcal{N}(y_i|\mathbf{w}^\top \mathbf{x}_i, \beta^{-1}) \\ &\propto \exp\left(-\frac{\beta}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w})\right) \end{aligned}$$

where precision $\beta = \frac{1}{\sigma^2}$ and in the second line above we have ignored the Gaussian normalization constants. By completing the square, show that with a prior distribution on $\mathbf{w}$ given by $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0)$, the posterior distribution $p(\mathbf{w}|D)$ is given by

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

where

$$\begin{aligned} \mathbf{m}_n &= \mathbf{S}_n(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\mathbf{X}^\top\mathbf{y}) \\ \mathbf{S}_n &= \left(\mathbf{S}_0^{-1} + \beta\mathbf{X}^\top\mathbf{X}\right)^{-1} \end{aligned}$$

Here's the first step. Take ln[(likelihood)(prior)] and collect normalization terms that don't depend on $\mathbf{w}$:

$$\begin{aligned} \ln p(\mathbf{w}|D) &\propto \ln p(\mathbf{y}|\mathbf{X}, \mathbf{w}) + \ln p(\mathbf{w}) \\ &= const - \frac{\beta}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) - \frac{1}{2}(\mathbf{w} - \mathbf{m}_0)^\top\mathbf{S}_0^{-1}(\mathbf{w} - \mathbf{m}_0) \end{aligned}$$

Hint: Remember, you already know what the posterior should look like. Once you simplify your expression enough, try foiling the posterior in terms of $\mathbf{m}_n$ and $\mathbf{S}_n^{-1}$ and see if you can see the relationship between your expression and this posterior.

Solution:

2 Neural Networks

2.1 Takeaways

Recall that in the case of binary classification, we can think about a neural network as being equivalent to logistic regression with parameterized, adaptive basis functions. Adaptive means that you don't need to specify a feature basis. We can train a matrix that linearly transforms the data, run the resulting vector through an element-wise non-linearity, potentially repeat this process, and then finally run logistic regression on the resulting vector, where the logistic regression itself has weights that need to be trained.

2.2 Activation Functions

There a wide range of possible non-linear activation functions to choose from when designing neural networks. Among these, the most popular are ReLU, which you will encounter below, and the tanh activation function, which is exactly what it sounds like: $\tanh(z)$. There are many others, including sigmoid and softmax for the final output layers, that researchers decide between when constructing their models.

Note: Activation functions are vital to constructing neural networks because they introduce non-linear relationships between the inputs and outputs of various layers. If there were no special activation functions, every step of a feed-forward neural network would just reduce to matrix multiplication (try this out yourself!), and all the relationships from the initial inputs to the final outputs would just be linear combinations of variables. Only linear relationships would come out of the model!

2.3 Concept Question

What is the difference between the activation functions described here (ReLU, tanh, sigmoid, softmax) and the usage of sigmoid and softmax earlier in logistic regression?

The purpose of these activation functions is to be non-linear, while earlier the sigmoid and softmax functions were used to transform outputs into probabilities. Thus, sigmoid and softmax both necessarily have range $[0, 1]$, while for instance tanh has range $[-1, 1]$. For neural networks, having a range centered around 0, like tanh, is an advantage because it means there is no systemic bias being introduced to the output values, whereas functions like ReLU, sigmoid and softmax result in all non-negative output values which can limit the expressiveness of the neural network.

2.4 Exercise: A Simple NN Classifier

Let's think about a neural network binary classifier with $\mathbf{x} \in \mathbb{R}^2$ ($D = 2$) and with a single two-dimensional hidden layer.
For the non-linear activation function, we use the *ReLU* function defined by the following:

$$\text{ReLU}(z) = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Let's consider a function h defined by the following:

$$h(\mathbf{x}) = \mathbf{w}^\mathsf{T}\phi(\mathbf{x}) + w_0 \tag{12}$$

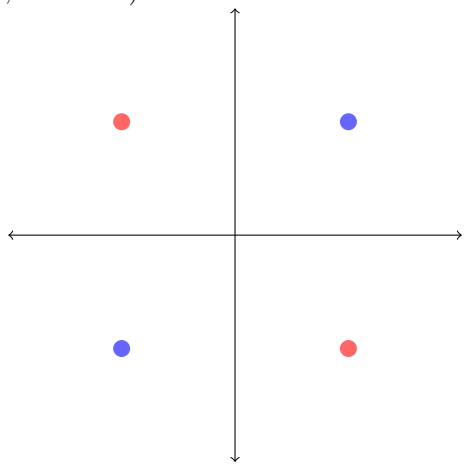
$$= \mathbf{w}^\mathsf{T}\mathbf{ReLU}\left(\mathbf{W}^{hid}\mathbf{x} + \mathbf{w}_0^{hid}\right) + w_0, \tag{13}$$

where example $\mathbf{x} \in \mathbb{R}^{2\times 1}$, weights in the hidden layer $\mathbf{W}^{hid} \in \mathbb{R}^{2\times 2}$, bias in the hidden layer $\mathbf{w}_0^{hid} \in \mathbb{R}^{2\times 1}$, output weight vector $\mathbf{w} \in \mathbb{R}^{2\times 1}$, and output bias $w_0 \in \mathbb{R}$. Everything within the **ReLU()** is the "adaptive basis" and the parameters outside are that of the logistic regression model.
Suppose we want to fit the following data:

$$\mathbf{x}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \; y_1 = 1, \quad \mathbf{x}_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \; y_2 = -1$$

$$\mathbf{x}_3 = \begin{pmatrix} -1 \\ 1 \end{pmatrix} \; y_3 = -1, \quad \mathbf{x}_4 = \begin{pmatrix} -1 \\ -1 \end{pmatrix} \; y_4 = 1$$

This looks as follows (blue = 1, red = -1):



Why can't we solve this problem with a linear classifier? What values of parameters $\mathbf{W}^{hid}, \mathbf{w}_0^{hid}, \mathbf{w}$, and w_0 will allow the neural network to solve the problem? Show that your choice of parameters allows the network to correctly classify the data.
Note: You do not need to learn to find these weights systematically by hand. This is not very possible to do manually in general, but you might see something in this low-dimensional case.
Hint: Think carefully about what the ReLU activation function can do for us. What does it do to various kinds of vectors? Think geometrically.

Solution: