CS 181 Spring 2024 Section 4 Notes
Solution

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 how likely different values of the model θ are after updating the prior distribution with the observed data.
- The **posterior predictive** for new data: $p(y^*|x^*,X,Y) = \int p(y^*|x^*,\theta)p(\theta|X,Y)d\theta$
  This tells us how to predict the label of a new data point according to the posterior over models obtained by updating the prior with 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. In contrast to the likelihoods we've seen before which are computed given a specific setting of weights θ, the marginal likelihood accounts for a distribution over weights θ. The marginal likelihood allows us to compare different priors or even different model classes in terms of how well they fit the data (this is model selection).

To summarize, in Bayesian Regression, we encode our assumptions within the prior distribution $p(\theta)$, which is updated in the **posterior** distribution $p(\theta|X,Y)$. The posterior functions as our updated beliefs after seeing our data X,Y, along with the most likely model for that data. The **posterior predictive** enables us to leverage the data we have seen to create predictions across all potential models (hence, the name). Finally, the **marginal likelihood** helps describe the likelihood of the data (producing Y given X and the model θ) and can be interpreted in model selection as selecting the model that yields the highest likelihood of producing the data distribution (most likely to explain the result).

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 $p(Y|X,\theta)$ 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 $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, $\mathbf{x}_i \in \mathbb{R}^m$, $y_i \in \mathbb{R}$. 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 covariance $\mathbf{S}_0$ known):

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mu_0, \mathbf{S}_0) \tag{3}$$

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

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

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

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

where

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

$$\mu_N = \mathbf{S}_N(\mathbf{S}_0^{-1}\mu_0 + \beta\mathbf{X}^\top\mathbf{y}) \tag{7}$$

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 $\mathbf{w}$? We can expand the predictive distribution over y^* given $\mathbf{x}^*$ using the Law of Total Probability:

$$
\begin{aligned}
p(y^*|\mathbf{x}^*, \mathcal{D}) &= \int_{\mathbf{w}} p(y^*|\mathbf{x}^*, \mathbf{w}) p(\mathbf{w}|\mathcal{D}) d\mathbf{w} && (8) \\
&= \int_{\mathbf{w}} \mathcal{N}(y^*|\mathbf{w}^\top \mathbf{x}^*, \beta^{-1}) \mathcal{N}(\mathbf{w}|\mu_N, \mathbf{S}_N) d\mathbf{w} && (9)
\end{aligned}
$$

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:

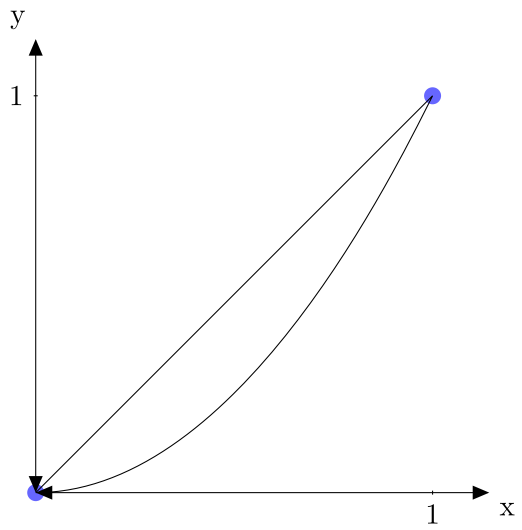
$$
p(y^*|\mathbf{x}^*, \mathcal{D}) = \mathcal{N}(y^*|\mu_N^\top \mathbf{x}^*, \mathbf{x}^{*\top} \mathbf{S}_N \mathbf{x}^* + \beta^{-1}) \tag{10}
$$

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^*. How does this relate to the expressivity of the different model classes?

Solution

1. There are many possible answers, but a natural one is that a_0, a_1, a_2 are uniformly distributed across the real numbers, since we don't know anything about them yet. One could also argue that the weights are more likely to be close to zero than not, so perhaps a Gaussian distribution centered at zero makes sense as well.

2. Yes, with zero error, we must have $f(0) = 0$ and $f(1) = 1$. Thus, we can calculate the posterior relationships $\hat{y}(0) = a_0 = 0$ and $\hat{y}(1) = a_0 + a_1 + a_2 = 1 \Rightarrow a_1 + a_2 = 1$. Our posterior distributions are thus $a_0 = 0$, a_1 is still uniform over real numbers, and $a_2 = 1 - a_1$.

3. We see that $y^*(1/2) = a_0 + a_1(1/2) + a_2(1/2)^2 = \frac{1}{2}a_1 + \frac{1}{4}(1 - a_1) = \frac{1}{4} + \frac{1}{4}a_1 \sim \text{Unif}(0, 1/2)$. Thus, the posterior predictive for y^* is a uniform distribution with support $[0, 1/2]$.

4. For a linear regression, we must have $\hat{y}(0) = a_0 = 0$ and $\hat{y}(1) = a_0 + a_1 = 1 \Rightarrow a_1 = 1$. Thus, $y^*(1/2) = a_0 + a_1(1/2) = 1/2$, and the posterior predictive for y^* is a dirac delta distribution centered at $1/2$, which means $y^* = 1/2$ with probability 1. If we compare the two models' posterior distributions, we see that the linear regression has a much spikier distribution, all concentrated at $y^* = 1/2$, which is stronger than the parabolic regression's distribution spread evenly across $[0, 1/2]$. Thus, we can reason about the models in the following way: a less expressive (linear) model has a smaller range of possible predictives but conveys a stronger conviction for the predictions it does issue, while a more expressive (parabolic) model has a larger range of possible predictives but conversely a weaker convictions for all the predictions it can issue.

End 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 $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mu_0, \mathbf{S}_0)$ where $\mathbf{S}_0$ is the covariance matrix, the posterior distribution $p(\mathbf{w}|\mathcal{D})$ is given by

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

where

$$\begin{aligned} \mu_N &= \mathbf{S}_N(\mathbf{S}_0^{-1}\mu_0 + \beta\mathbf{X}^\top\mathbf{y}) \\ \mathbf{S}_N &= \left(\mathbf{S}_0^{-1} + \beta\mathbf{X}^\top\mathbf{X}\right)^{-1} \end{aligned}$$

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

$$\begin{aligned} \ln p(\mathbf{w}|\mathcal{D}) &\propto \ln p(\mathbf{y}|\mathbf{X},\mathbf{w}) + \ln p(\mathbf{w}) \\ &= const - \frac{\beta}{2}(\mathbf{y}-\mathbf{X}\mathbf{w})^\top(\mathbf{y}-\mathbf{X}\mathbf{w}) - \frac{1}{2}(\mathbf{w}-\mu_0)^\top\mathbf{S}_0^{-1}(\mathbf{w}-\mu_0) \end{aligned}$$

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

Solution

As the problem statement suggestions, the first step is to take ln[(likelihood)(prior)] and collect normalization terms that don't depend on **w**:

$$\begin{aligned}\ln p(\mathbf{w}|D) &\propto \ln p(\mathbf{y}|\mathbf{X},\mathbf{w}) + \ln p(\mathbf{w}) && (11)\\ &= const - \frac{\beta}{2}(\mathbf{y}-\mathbf{X}\mathbf{w})^\top(\mathbf{y}-\mathbf{X}\mathbf{w}) - \frac{1}{2}(\mathbf{w}-\mu_0)^\top\mathbf{S}_0^{-1}(\mathbf{w}-\mu_0) && (12)\end{aligned}$$

Expanding, we have:

$$\begin{aligned}const - \frac{1}{2}\Big(&\beta\mathbf{y}^\top\mathbf{y} - \beta\mathbf{y}^\top\mathbf{X}\mathbf{w} - \beta\mathbf{w}^\top\mathbf{X}^\top\mathbf{y} + \beta\mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w} && (13)\\ &+\mathbf{w}^\top\mathbf{S}_0^{-1}\mathbf{w} - \mathbf{w}^\top\mathbf{S}_0^{-1}\mu_0 - \mu_0^\top\mathbf{S}_0^{-1}\mathbf{w} + \mu_0^\top\mathbf{S}_0^{-1}\mu_0\Big) && (14)\end{aligned}$$

Important: Remember that terms like $\mathbf{y}^\top\mathbf{X}\mathbf{w}$ are the same scalar as $\mathbf{w}^\top\mathbf{X}^\top\mathbf{y}$. Collecting together quadratic and linear terms, factoring the **w**s out, and moving terms that don't depend on **w** into the constant, we have

$$const - \frac{1}{2}\left(\mathbf{w}^\top(\mathbf{S}_0^{-1} + \beta\mathbf{X}^\top\mathbf{X})\mathbf{w} - 2\mathbf{w}^\top(\mathbf{S}_0^{-1}\mu_0 + \beta\mathbf{X}^\top\mathbf{y})\right) \tag{15}$$

Put aside what we have done so far. Recall that our target looks like:

$$-\frac{1}{2}\left((\mathbf{w}-\mu_N)^\top\mathbf{S}_N^{-1}(\mathbf{w}-\mu_N)\right), \tag{16}$$

When expanded, this looks like

$$-\frac{1}{2}\left(\mathbf{w}^\top\mathbf{S}_N^{-1}\mathbf{w} - \mu_N^\top\mathbf{S}_N^{-1}\mathbf{w} - \mathbf{w}^\top\mathbf{S}_N^{-1}\mu_N + \mu_N^\top\mathbf{S}_N^{-1}\mu_N\right) \tag{17}$$

Drop the term that doesn't have **w** and combine the two middle terms

$$-\frac{1}{2}\left(\mathbf{w}^\top\mathbf{S}_N^{-1}\mathbf{w} - 2\mathbf{w}^\top\mathbf{S}_N^{-1}\mu_N\right) \tag{18}$$

This looks like what we ended up with in (15) where

$$\mathbf{S}_N^{-1} = \mathbf{S}_0^{-1} + \beta\mathbf{X}^\top\mathbf{X}$$

and

$$\mathbf{S}_N^{-1}\mu_N = \mathbf{S}_0^{-1}\mu_0 + \beta\mathbf{X}^\top\mathbf{y},$$

as required. This has the desired form of an (unnormalized) Gaussian.

End 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.

What properties do Neural Networks (NNs) enable?

1. **Rings and XOR example**: Neural Nets are capable of "learning" the representation of the XOR function, which exhibits the ability to match non-trivial functions. Additionally, in the Rings example, we show that NNs can also fit even more complex examples with multiple linear decision boundaries.
2. **Deep Neural Nets**: These specific types of NNs can be understood as a composition of functions. Deep NNs are best suited to capture complex functions and representations that are difficult to capture with simpler NN architectures.

2.2 Activation Functions

There are 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. Additionally, Deep NNs tend to be easier to train, and facilitate feature reusability.

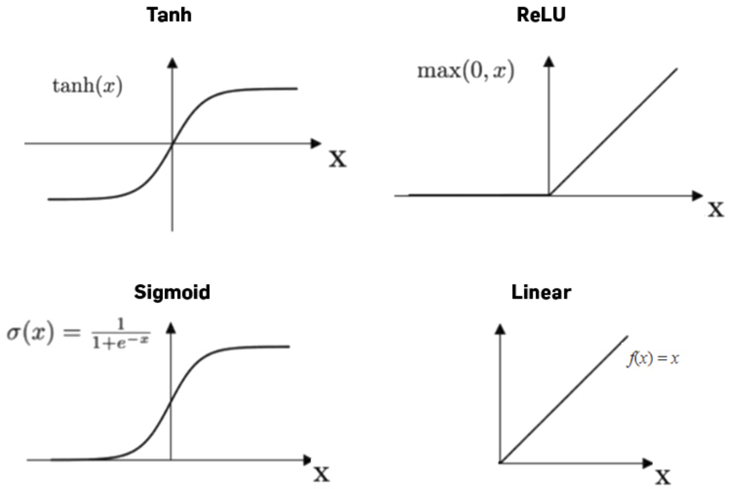


Figure: tanh, ReLu, and Sigmoid activation functions, compared with a simple linear relationship. Source: AI Wiki

Why do we need activation functions? 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?

Answer: 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 More Types of Neural Nets

Convolutional Neural Nets (CNNs): CNNs are used widely for computer vision tasks such as image recognition, segmentation (partitioning or classifying parts of an image), and object detection. The use of CNNs is best suited for data, such as a set of pixels in an image, that can be represented in a hierarchical format — meaning starting from the smallest details, to curves, shapes and larger, more abstract components of an image. CNNs build increasingly more complex representations of images by pooling and learning from details in previous layers in order to tackle tasks like image classification. These types of neural nets employ methods such as the use of **convolutional layers**, **pooling** (i.e. max pooling), **filters**, **fully-connected layers**, amongst other concepts to self-learn representations of the data.

Other Types of Neural Nets:

1. Residual Net: Residual Nets utilize "skip connections" which prevent issues such as vanishing gradients from occurring in particularly deep CNNs.

2. Recurrent Neural Net (RNN): A recurrent neural net is different than a *feed-forward neural net* in that the inputs from a layer can be fed back into previous input layers (in a recurrent, cyclical way). These representations can be "unrolled" to represent other structures.

3. Auto-encoder: Autoencoders utilize an encoder-decoder structure which enables input data representations to be encoded into a separate dimensional space, and then decoded back to the data's original input space.

4. Attention: Attention forms the backbone of the Transformer architecture, which is the basis for many language model technologies (i.e. GPT). The structure takes in a series of multiple inputs (i.e. words in a sentence) and selectively allocates weights to different parts of input

data or output from other layers (self-attention). This enables the model to selectively focus on the most important pieces of the data.

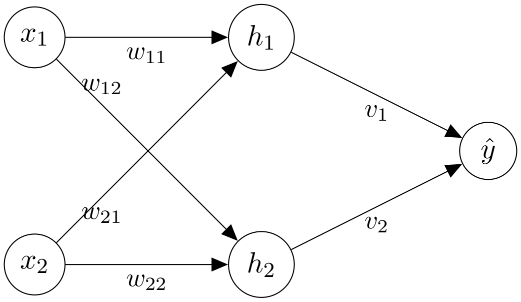


Figure 1: Fully Connected Neural Network

2.5 Exercise: Simple Neural Network Example

Figure 1 (above) describes a simple neural network architecture with two input nodes x_1, x_2, one hidden layer h_1, h_2, and one output node z.

Suppose we have the following weights initialized for $w_{11}, w_{12}, w_{21}, w_{22}$: $\{0.5, 0.1, 0.1, 0.55\}$. Additionally, we have $v_1, v_2 = \{0.9, 0.1\}$

1. Suppose that we use the activation function ReLu in each of our hidden layers, such that $h_1, h_2, \hat{y}$ are given by:

$$h_1 = \text{ReLU}(w_{11}x_1 + w_{21}x_2)$$

$$h_2 = \text{ReLU}(w_{12}x_1 + w_{22}x_2)$$

$$\hat{y} = \sigma(v_1h_1 + v_2h_2)$$

For input $x_1 = -0.5$, $x_2 = 1$, what is the final value of $\hat{y}$? Leave your final expression in terms of the sigmoid function $\sigma(z)$.

2. Derive the expression for the gradient of w_{11}, $\frac{\delta L}{\delta w_{11}}$, using the chain rule. Do not simplfy — leave in terms of the predefined variables. Our loss function L is given by:

$$L = -(y\log(\hat{y}) + (1-y)\log(1-\hat{y}))$$

How would we use this resulting expression to update w_{11}?

Solution

1. By chaining through the network, we calculate values for our hidden layer: $h_1 = \text{ReLU}(0.5 * -0.5 + 0.1 * 1) = 0$, $h_2 = \text{ReLU}(0.1 * -0.5 + 0.55 * 1) = 0.5$.

   Following through the hidden layer, we have $\hat{y} = \sigma(0.9 * 0 + 0.1 * 0.5) = \sigma(0.05)$.

2. Following backwards through the neural net, we can use the chain rule to trace through from $\hat{y}$ to w_{11}, while taking the partial of the loss function L. We arrive upon the following:

$$\frac{\delta L}{\delta w_{11}} = \frac{\delta L}{\delta \hat{y}} \cdot \frac{\delta \hat{y}}{\delta h_1} \cdot \frac{\delta h_1}{\delta w_{11}}$$

End Solution