Assignment 6: Introduction to Neural Networks and Backpropagation

Machine Learning

Fall 2019

Learning Objectives

- Understand the difficulties that neural networks address in comparison to algorithms like logistic regression.
- Interpret the results of applying a simple neural network to a dataset.
- Understand the architecture of a particular type of neural network called a multi-layer perceptron.
- Learn how to represent the multivariable chain rule graphically.
- Understand how the backpropagation algorithm can be used to compute the gradient of a loss function with respect to the parameters of a neural network.

1 Assignment Structure

In this assignment we'll be doing the following things in order to meet the learning goals articulated above.

1. Motivate the idea of neural networks through a Jupyter notebook that examines the Titanic Dataset from Kaggle.
2. Introduce the architecture of a particular type of neural network called a multi-layer perceptron.
3. See another way to think about the chain rule for multivariable functions that uses a graphical representation.
4. Learn about, and ultimately derive, the backpropagation algorithm.

Prior Knowledge Utilized

Here are some things we'll be utilizing in this assignment. When appropriate, we'll call these out in a particular section with some helpful text to jog your memory.

- Logistic regression algorithm
- Multivariable chain rule
- Binary classification problem setting

2 Motivation for Neural Networks

In order to motivate the idea of neural networks, we'll be examining the Titanic Kaggle dataset. If you didn't work with this data, it might help you to briefly skim the walk-through in assignment 5.

External Resource(s) (TODO minutes)

Go through the Assignment 6 Companion notebook.

3 Our First Neural Network: the Multilayer Perceptron (MLP)

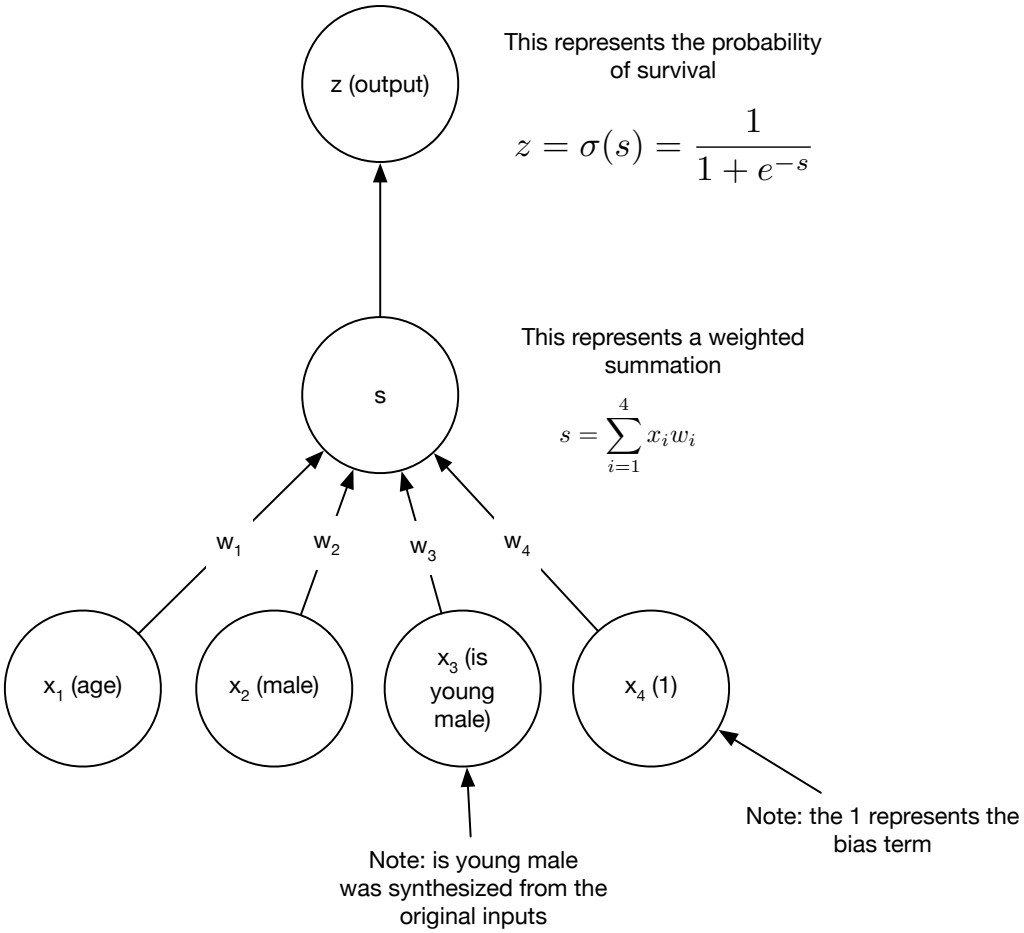
TODO: include refresher on logistic regression.

Now that you've seen a neural network in action, we'll be digging into how a neural network works. The presentation will be specific to a particular type of neural network known as a multilayer perceptron, but the main ideas generalize to many other types of networks.

Thinking back to the companion notebook, we observed that the features in the original dataset (`age` and `sex`) were not conducive to predicting whether someone would survive. We showed that by augmenting the input features with a column called `is young male` that captured whether or not a person was young *and* male, that the algorithm could effectively learn the task. The fundamental idea of a neural network is that the network automatically constructs useful representations of the input data *as a part of the learning process.*

Graphically we can contrast these approaches in the following way[1]. First we'll show the logistic regression model that we applied in the notebook.

[1] to interpret these graphs, think of the circles (also called nodes) as representing values that are put into or computed by the network. Directed lines (also called edges) as representing data flowing in the network. Each edge multiplies the value flowing into it by a weight (represented by a text label on the edge).



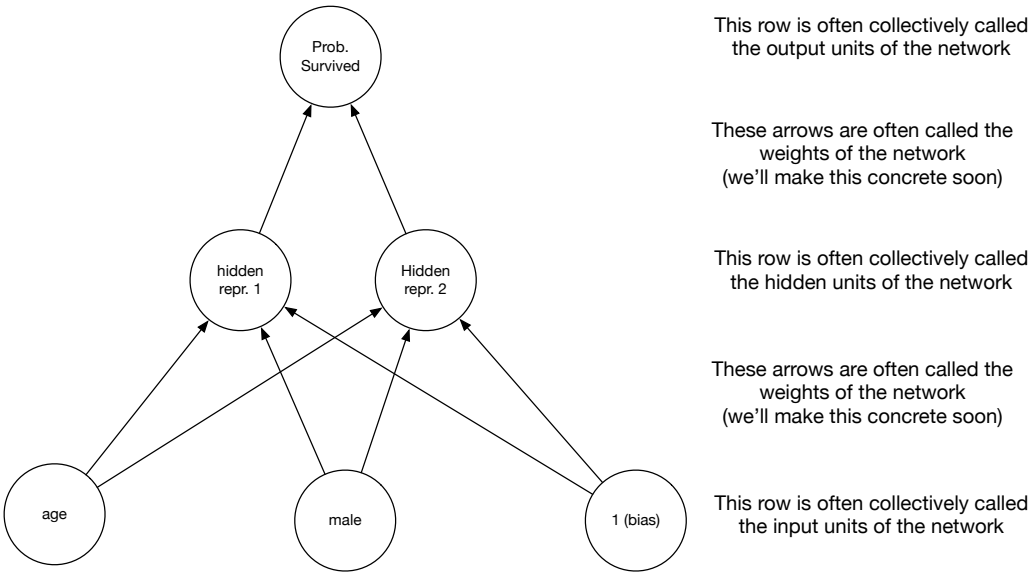
Notice how we had to manually introduce the feature `is young male` in order for the logistic regression model to utilize it to make its prediction. In contrast, here is the neural network that we fit at the end of the notebook.

Before giving you the equivalent figure for the multi-layer perceptron, let's look at a little bit more cartoonish version of the multi-layer perceptron. This version will leave off the math and the particular notation we are using. Once you have a good sense of what this is, you can look at the more precise version which is to follow.

External Resource(s)

Here are some additional resources that explain the concept of a multi-layer perceptron. If the explanations we give below are not working for you, consider checking out some of these. **You do not need to consult these resources if you feel like our explanations are working well for you.**

- todo



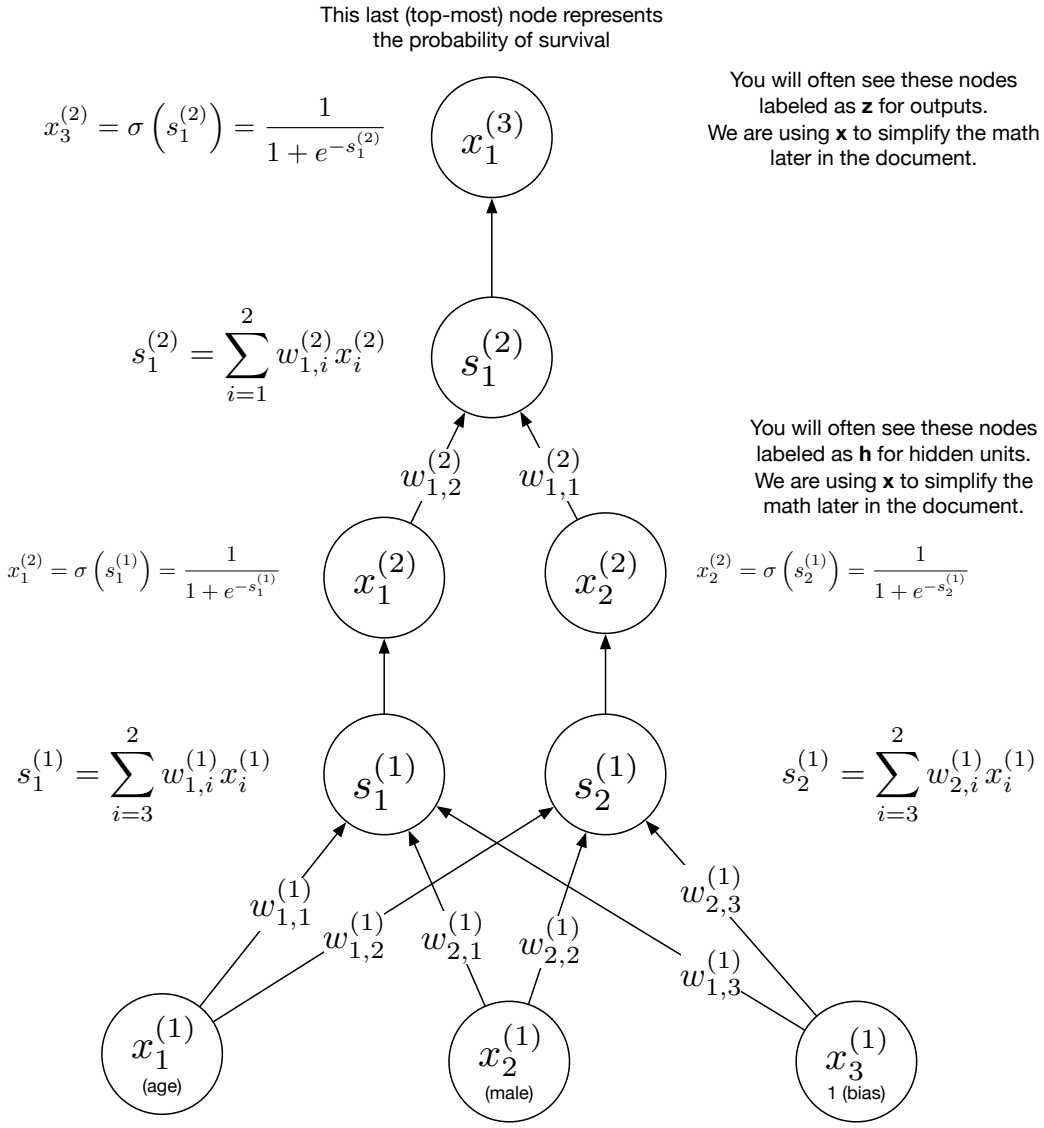
This figure shows the basic idea of a multi-layer perceptron (MLP. Input data (in this case we just use age, male, and a bias term) are propagated via a set of connection weights to a set of hidden representations. These hidden representations are propagated via another set of a connection weights to the output of the network. In the companion notebook we showed that in the Titanic example, the network learned two hidden representations: one that encoded `is young male` and the other that encoded sex. Of particular importance is that we did not have to manually introduce the `is young male` feature.

Next, let's make this cartoon picture concrete. We'll use the following notation.

- $x_i^{(j)}$ will refer to the ith unit in the jth layer of the network ($j = 1$ will correspond to the inputs, $j = 2$ will correspond to the hidden representations, and $j = 3$ will correspond to the output). For instance, in the figure above the circle labeled *male* would be $x_2^{(1)}$, *hidden repr. 1* would be $x_1^{(2)}$ and *prob survived* would be $x_1^{(3)}$.
- $s_i^{(j)}$ will refer to the ith summation unit in the jth layer of the network (as before, $j = 1$ will correspond to the inputs, $j = 2$ will correspond to the hidden representations, and $j = 3$ will correspond to the output). The summation unit will play a similar role to s in the logistic regression figure.
- $w_{i,j}^{(k)}$ will refer to the weight of the connection between the the jth unit in layer k and the ith unit in layer $k + 1$.

⚠ Notice

There are a lot of symbols here! Take some time to unpack each of them. Make sure you know what superscripts and subscripts represent. While there is an upfront cost to introducing these symbols, it will ultimately make our lives considerably simpler.



Exercise 1 (TODO minutes)

There's a lot to interpret an unpack. Make sure they can do it. TODO.

(a) TODO: relate the structure of the MLP to logistic regression. You should see it represented multiple times.

Exercise 2 (TODO minutes)

Without using any training data (this is testing your understanding of the model itself), compute the weights in this network $(w_{1,1}^{(1)}, w_{1,2}^{(1)}, w_{1,3}^{(1)}, w_{2,1}^{(1)}, w_{2,2}^{(1)}, w_{2,3}^{(1)}, w_{1,1}^{(2)}, w_{1,2}^{(2)})$ such that the MLP has the follow-

ing behavior. Recall that $x_1^{(1)}$ is the passenger's age, $x_2^{(1)}$ is a binary variable that is 1 if the passenger is male and 0 if female, $x_3^{(1)}$ is a constant term (always 1).

- $x_1^{(2)}$ encodes whether or not the passenger is female (i.e., it should take a value close to 1 when the passenger is female and 0 when the passenger is male).
- $x_2^{(2)}$ encodes whether or not the passenger is a young male (i.e., it should take a value close to 1 when the passenger is male under the age of say 5 and 0 otherwise).
- $x_1^{(3)}$ should be close to 1 (i.e., predict survival) when the passenger is female *or* a male under the age of 5.

Believe it or not, computing these weights by hand was fairly common back before we had algorithms for automatically learning the weights from data. The reason for this was that early techniques for learning the weights were very inefficient and often unable to converge to good solutions. Later in this document we will be learning about the back-propagation algorithm that can be used to efficiently compute the gradient of the weights in this neural network with respect to some cost function. **Just as we did with logistic regression, we can use this gradient in order to optimize the weights of the network using gradient descent.** What's beautiful is that even though the model itself got more complicated, the learning algorithm and basic ideas remain largely the same.

In order to prepare ourselves for the derivation of the backpropagation algorithm, we need to build up a bit more powerful method of applying the chain rule to multivariate functions.

4 A Graphical View of the Multivariable Chain Rule

In assignment 3 we learned the multivariable chain rule, which allowed us to take partial derivatives (or the gradient) of the composition of a multivariable and a single variable function. In the listing below, h is a function from a vector to a scalar, f is from a vector to a scalar, and g is from a scalar to a scalar.

$$\begin{aligned} h(\mathbf{w}) &= g(f(\mathbf{w})) && \text{h}(\mathbf{w}) \text{ is the composition of } f \text{ with } g \\ \nabla h(\mathbf{w}) &= g'(f(\mathbf{w}))\nabla f(\mathbf{w}) && \text{this is the multivariable chain rule} \end{aligned} \tag{1}$$

$$\frac{\partial h(\mathbf{w})}{\partial w_i} = g'(f(\mathbf{w}))\frac{\partial f}{\partial w_i} \qquad \text{this is for a single partial deriv. (rather than the gradient)} \tag{2}$$

If we were to write out the MLP example in the previous section using this notation, we'd have a huge mess. The function would probably barely fit on one line of this document. Luckily, there's another way to apply the chain rule that uses the concept of a

dataflow diagram. What we will soon see is that not only will the dataflow diagram make our lives easier from a mathematical perspective, it will actually make our lives easier from a computational perspective (that last bit is a foreshadowing of the backpropagation algorithm, which we'll soon meet).

External Resource(s) (20 minutes)

This HMC calculus tutorials explain this concept beautifully. Go and read the HMC Multivariable Chain Rule Page and come back for some exercises to test your understanding.

Exercise 3

(a) Some quick probably to make sure they get it.

(b) Generalize to gradients.

5 Backpropagation

Before getting into the derivation of the backpropagation algorithm, let's revisit our logistic regression model.

TODO replicate the figure from before.

Exercise 4

(a) Add a log loss function node to the top.

(b) Compute the gradient.

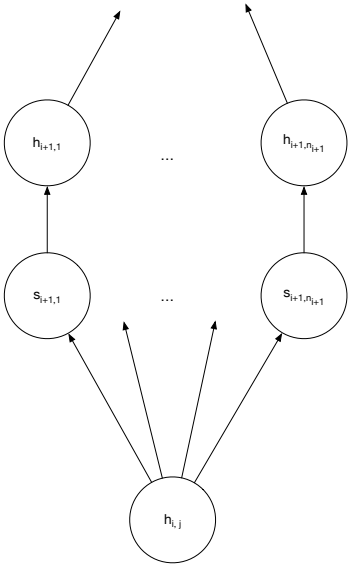
(c) Comment on flow of information in both forwards and backwards direction.

5.1 Forward Pass

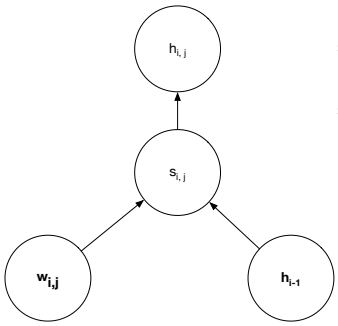
TODO, all symbols are defined in the network.

5.2 Backward Pass: Applying the Chain Rule

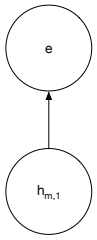
TODO



$$\begin{aligned}\frac{\partial e}{\partial h_{i,j}} &= \sum_{k=1}^{n_{i+1}} \frac{\partial s_{i+1,k}}{\partial h_{i,j}} \frac{\partial h_{i+1,k}}{\partial s_{i+1,k}} \\ &= \sum_{k=1}^{n_{i+1}} w_{j,k}^{i+1} \sigma(s_{i+1,k})(1-\sigma_{i+1,k})\end{aligned}$$



$$\begin{aligned}\nabla_{\mathbf{w_{i,j}}} e &= \frac{\partial e}{s_{i,j}} \nabla_{\mathbf{w_{i,j}}} s_{i,j} \\ &= \frac{\partial e}{h_{i,j}} \frac{\partial h_{i,j}}{s_{i,j}} \nabla_{\mathbf{w_{i,j}}} s_{i,j} \\ &= \frac{\partial e}{h_{i,j}} \sigma(s_{i,j})(1-\sigma(s_{i,j}))\mathbf{h}_{i-1}\end{aligned}$$



The last hidden layer can be considered the output.
You could also call this z.

This is the math for log loss.

$$\frac{\partial e}{\partial h_{m,1}} = -y\frac{1}{h_{m,1}} - (1-y)\frac{1}{1-h_{m,1}}$$

Todo: use a variable for layer instead of just hidden (this would simplify things).
Hidden unit to error:

$$\frac{\partial e}{\partial h_{i,j}} = \sum_{k=1}^{n_{i+1}} \frac{\partial s_{i+1,k}}{\partial h_{i,j}} \frac{\partial h_{i+1,k}}{\partial s_{i+1,k}} \frac{\partial e}{\partial h_{i+1,k}} \tag{3}$$

$$= \sum_{k=1}^{n_{i+1}} w_{j,k}^{i+1} \sigma(s_{i+1,k})(1-\sigma_{i+1,k}) \frac{\partial e}{\partial h_{i+1,k}} \tag{4}$$

Weights to error:

$$\nabla_{\mathbf{w_{i,j}}} e = \frac{\partial e}{s_{i,j}} \nabla_{\mathbf{w_{i,j}}} s_{i,j} \tag{5}$$

$$= \frac{\partial e}{h_{i,j}} \frac{\partial h_{i,j}}{s_{i,j}} \nabla_{\mathbf{w_{i,j}}} s_{i,j} \tag{6}$$

$$= \frac{\partial e}{h_{i,j}} \sigma(s_{i,j})(1 - \sigma(s_{i,j}))\mathbf{h}_{i-1} \tag{7}$$

Output to error (serves as a base case. For simplicity we use $h_{m,1}$ to refer to the single node in the mth layer (which is the output layer).

$$\frac{\partial e}{\partial h_{m,1}} = -y \frac{1}{h_{m,1}} - (1 - y) \frac{1}{1 - h_{m,1}} \tag{8}$$

External Resource(s)

- A Step-by-step Backpropagation Example