

## 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<sup>1</sup>. First we'll show the logistic regression model that we applied in the notebook.

<sup>1</sup> 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).



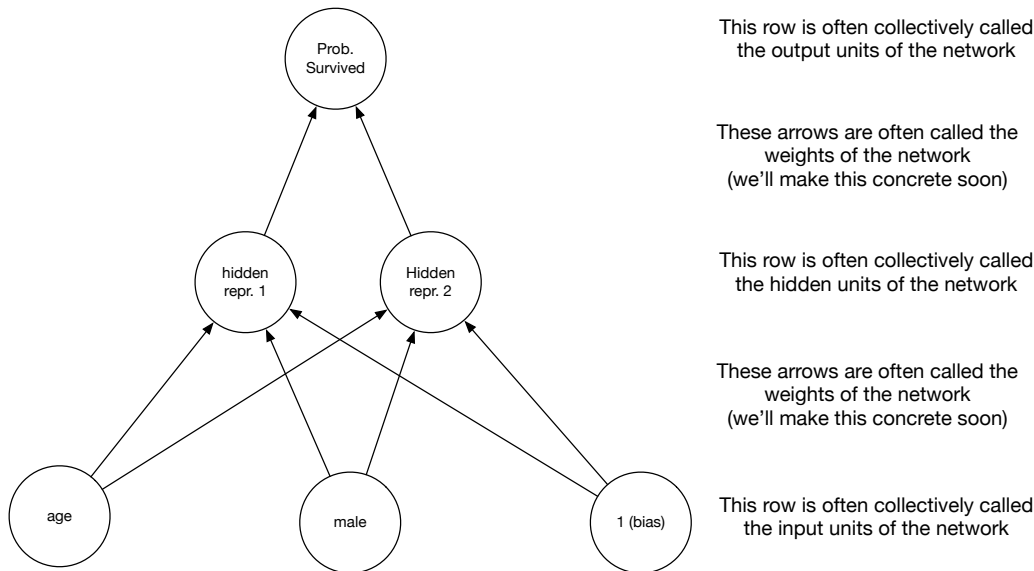
Notice how we had to manually introduce the feature **is young male** in order for the logistic regression model to utilize it to make its prediction. In contrast, here is the neural network that we fit at the end of the notebook.

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



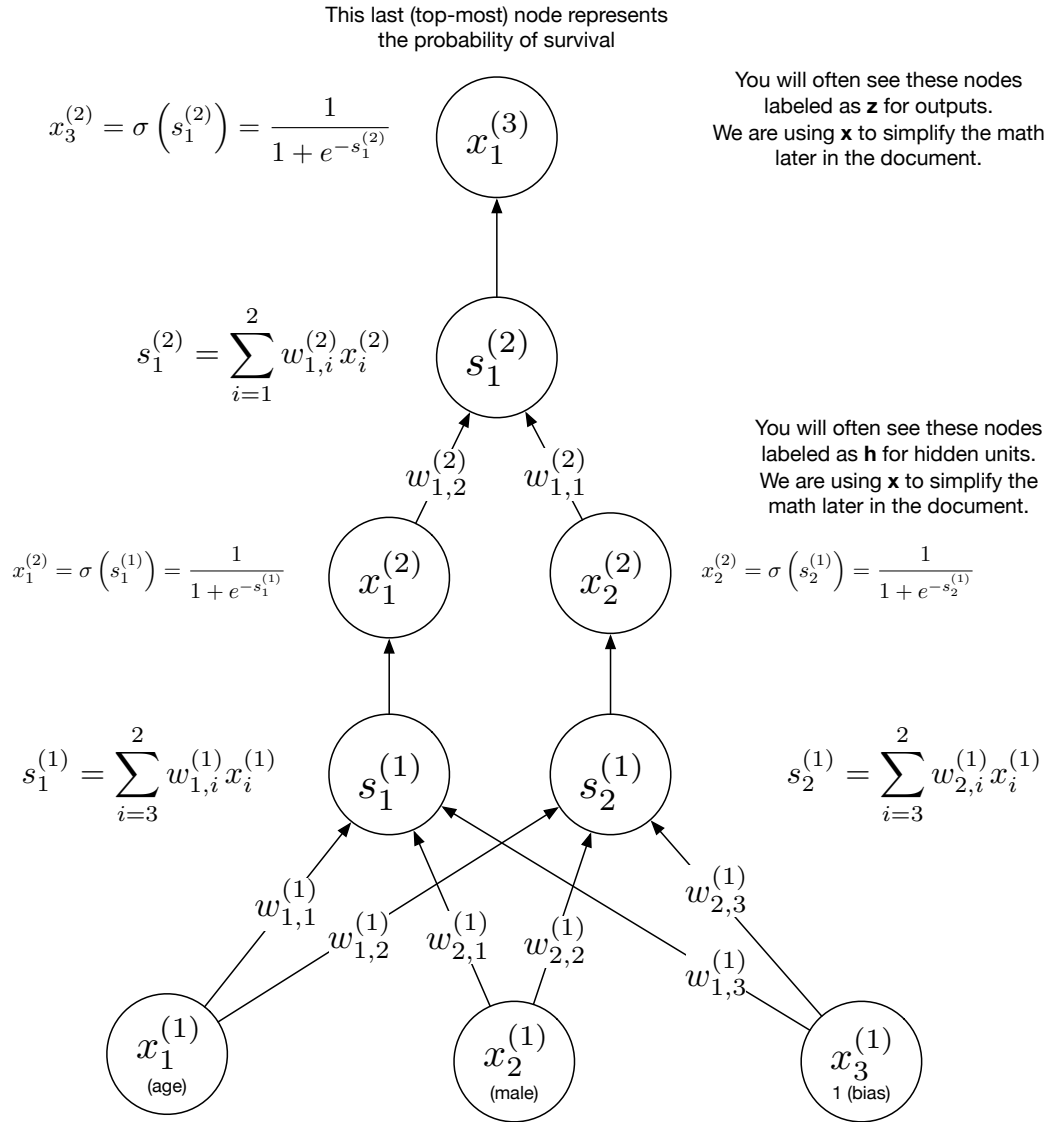
This figure shows the basic idea of a multi-layer perceptron (MLP). Input data (in this case we just use age, male, and a bias term) are propagated via a set of connection weights to a set of hidden representations. These hidden representations are propagated via another set of a connection weights to 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  $i$ th unit in the  $j$ th 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  $i$ th summation unit in the  $j$ th 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  $j$ th unit in layer  $k$  and the  $i$ th 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 backpropagation 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.

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

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

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

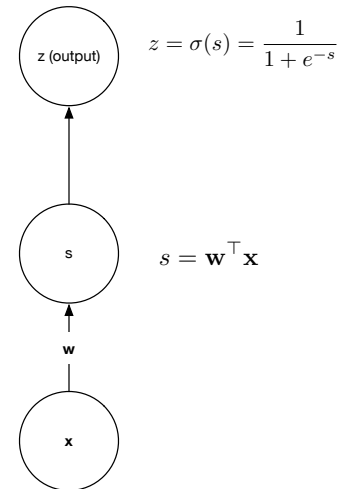
- (a) Some quick probably to make sure they get it.
- (b) Generalize to gradients.

## 5 Backpropagation

Before getting into the derivation of the backpropagation algorithm, let's revisit our logistic regression model from the last few assignments. For the logistic regression model, we were given training inputs  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$  with each  $\mathbf{x}_i$  being a  $d$ -dimensional vector and training outputs  $y_1, y_2, \dots, y_n$  with each  $y_i$  being a binary number that indicates the class that the  $i$ th instance belongs to. Given this training data, our goal was to compute the best possible set of weights by solving the following optimization problem.

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \sum_{i=1}^n \left( -y_i \ln \sigma(\mathbf{w}^\top \mathbf{x}_i) - (1 - y_i) \ln (1 - \sigma(\mathbf{w}^\top \mathbf{x}_i)) \right) \quad (3)$$

Even though you've seen this before, it might seem a bit intimidating. Remember, that this optimization problem arose from using the sigmoid function ( $\frac{1}{1+e^{-x}}$ ) to map  $\mathbf{w}^\top \mathbf{x}_i$  into a probability and then applying the log loss. We can represent the logistic regression model applied to an input point  $\mathbf{x}$  using the data flow representation shown in Figure 5.



### Exercise 4

Starting from the data flow representation of the logistic regression model shown in Figure 5, complete the following steps.

- (a) Add a new circle (node) to the graph that represents the log loss that the output  $z$  incurs when compared to the training output  $y$ .
- (b) Using the technique in the HMC calculus tutorials writeup, compute the gradient of the log loss with respect to weights,  $\mathbf{w}$ . *Hint: when examining the arrow from  $\mathbf{x}$  to  $s$ , instead of writing the partial derivative of  $s$  with respect to its input, you can instead write the gradient. The rest of the process in the HMC*

*calculus tutorial can be applied without modification.*

- (c) Was computing the gradient using this approach easier, harder, or the same in comparison with what you did in assignment 3? Why? (this is subjective, so there is no right answer, we just want you to reflect on the differences between the approaches and which you liked better).

### 5.1 Forward Pass

#### **⚠ Notice**

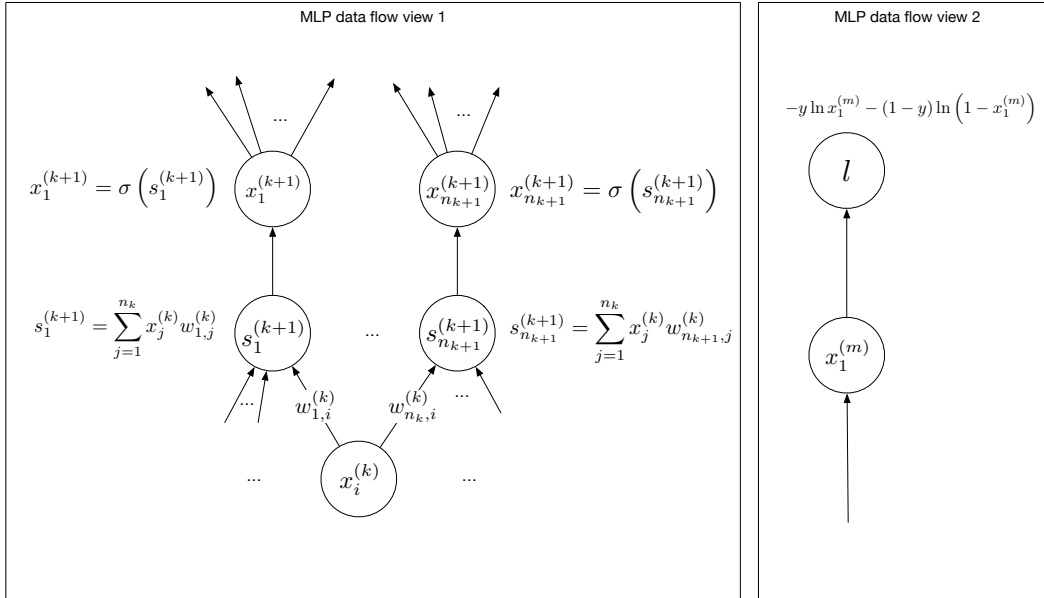
The notation for the general MLP gets pretty cluttered. It's important to realize that much of the complexity in this section is simply a result of keeping track of various indices (e.g., layer, unit number) rather than anything conceptually difficult. Our advice is to be diligent in going through these figures and make sure you have a firm grasp on the notation we are using. Please post on NB if something is confusing (or even just to verify your understanding).

Let's generalize the diagram of the multilayer perceptron in the Titanic example in the following ways.

1. We'll allow for an arbitrary number of layers in the network. We'll call the number of layers  $m$  (the number of layers is also called the depth of the network. networks with large  $m$  are known as deep learning).
2. We'll allow for an arbitrary number of units at each layer in the network. We'll use the notation  $n_k$  to refer to the number of units in the  $k$ th layer of the network.

These generalizations are summarized in the following figure that shows the data flow between two layers of the network (MLP data flow view 1) and between the output layer and the loss function.





Before we get into the math of how we can compute the gradient of the weights in this network using backpropagation, we must first perform the forward pass through the network. **In the forward pass, we use the formulas in the figure above to calculate the value for every circle in the network.** The forward pass is thus a very straightforward application of the equations shown here. For the backward pass, we can assume that we have the values for each of these nodes.

## 5.2 Backward Pass: Applying the Chain Rule

The stage is now set for you to finish the derivation of one of the most important algorithms in computer science!!! Before you finish this off, let's take a look at what you've learned and the tools you have at your disposal.

1. You know how to compute gradients using the data flow representation of a multivariable function.
2. We've applied this technique to deriving the gradient of the logistic regression model.
3. We've written the MLP in this same data flow representation.

Before we do this we'll augment the data flow graph just a little bit with to provide a view that focuses more directly on the weights coming into a particular neuron. In doing this, we'll refer to the weights coming into the  $i$ th node in layer  $k+1$  as  $w_i^{(k)}$ .

Let's get to it.

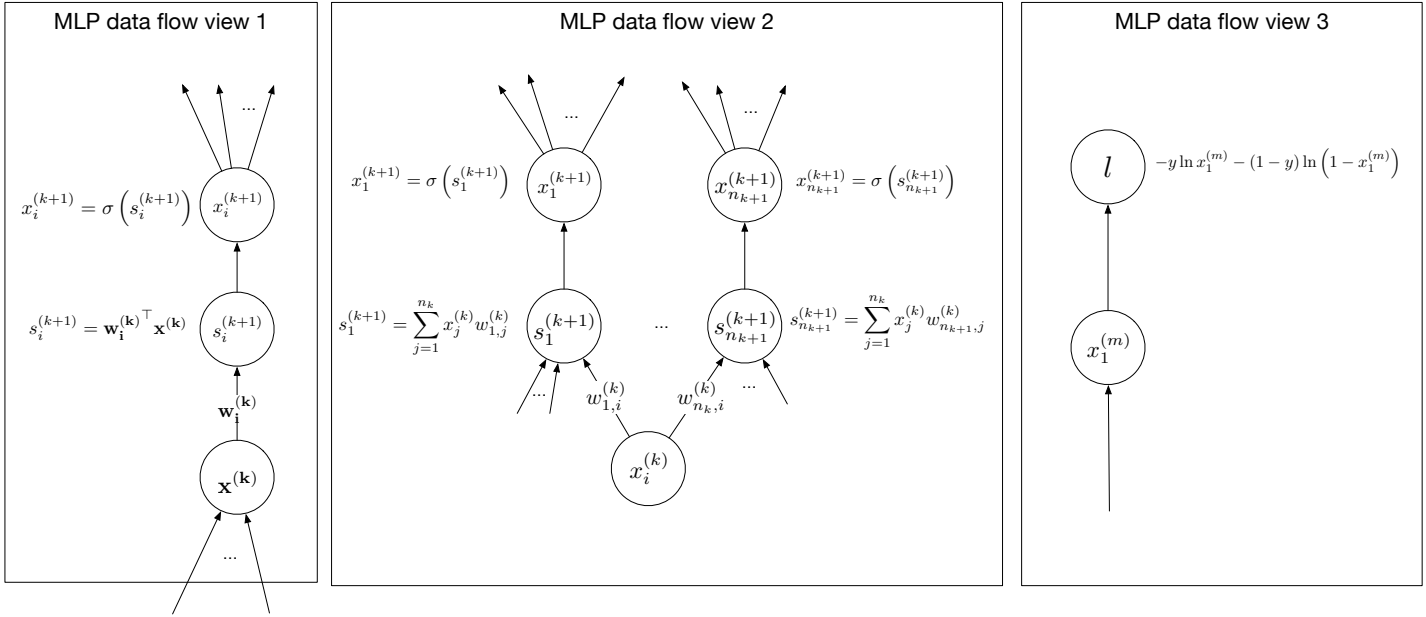


Figure 1: The general data flow graph for a multi-layer perceptron.

### Exercise 5 (60 minutes)

The backpropagation algorithm, not surprisingly, works by computing gradients in the network starting from the output and working backwards. In this exercise you'll work through the major steps of the backpropagation algorithm. This exercise should be done in reference to Figure 1.

- Referencing view 3, compute  $\frac{\partial l}{\partial x_1^{(m)}}$ .
- Referencing view 2, compute  $\frac{\partial l}{\partial x_i^{(k)}}$ . You can assume that you have already calculated the partial derivatives of the loss with respect to the layer  $k + 1$ . That is, you already have  $\frac{\partial l}{\partial x_1^{(k+1)}}, \dots, \frac{\partial l}{\partial x_{n_{k+1}}^{(k+1)}}$ .
- Referencing view 1, compute the gradient of the loss with respect to  $\mathbf{w}_i^{(k)}$ ,  $\nabla_{\mathbf{w}_i^{(k)}} l$ . You can assume you have already computed  $\frac{\partial l}{\partial x_i^{(k+1)}}$ .

## 6 Summary (AKA What the Heck Just Happened?)

That was a lot of math flying at you. We hope that you realize how cool what you just did was. When you combine the gradient you computed with gradient descent, you have derived a learning rule for efficiently tuning the weights of a special type of neural network called a multi-layer perceptron (MLP). The learning rules for almost every other type of neural network are just variations on this same theme. This is incredibly power-

ful, and you now have a very powerful mental model for how learning in neural networks is possible.

## 7 *Suggestions for Going Beyond*

### Notice

If you'd like to take this material farther, here are some suggestions. We do not at all expect you to do any of this, but we felt that these suggestions might be helpful to folks who want to go into additional depth with this material. We will never make any assumption that you did any of these additional tasks in any course materials. If you also have other ideas for additional topics to explore, feel free to go in that direction instead (please let us know what you do though in case it is of interest to others).

- Implement the backpropagation algorithm to compute the gradient for an MLP. Hint: if you do this, e-mail us for hints / guidance. It is pretty challenging, but it does pay off in terms of solidifying the backpropagation algorithm.
- Make sure your implementation is correct by comparing the numerical approximation of the gradient to the gradient you calculated with the backpropagation algorithm.