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) (45 minutes)

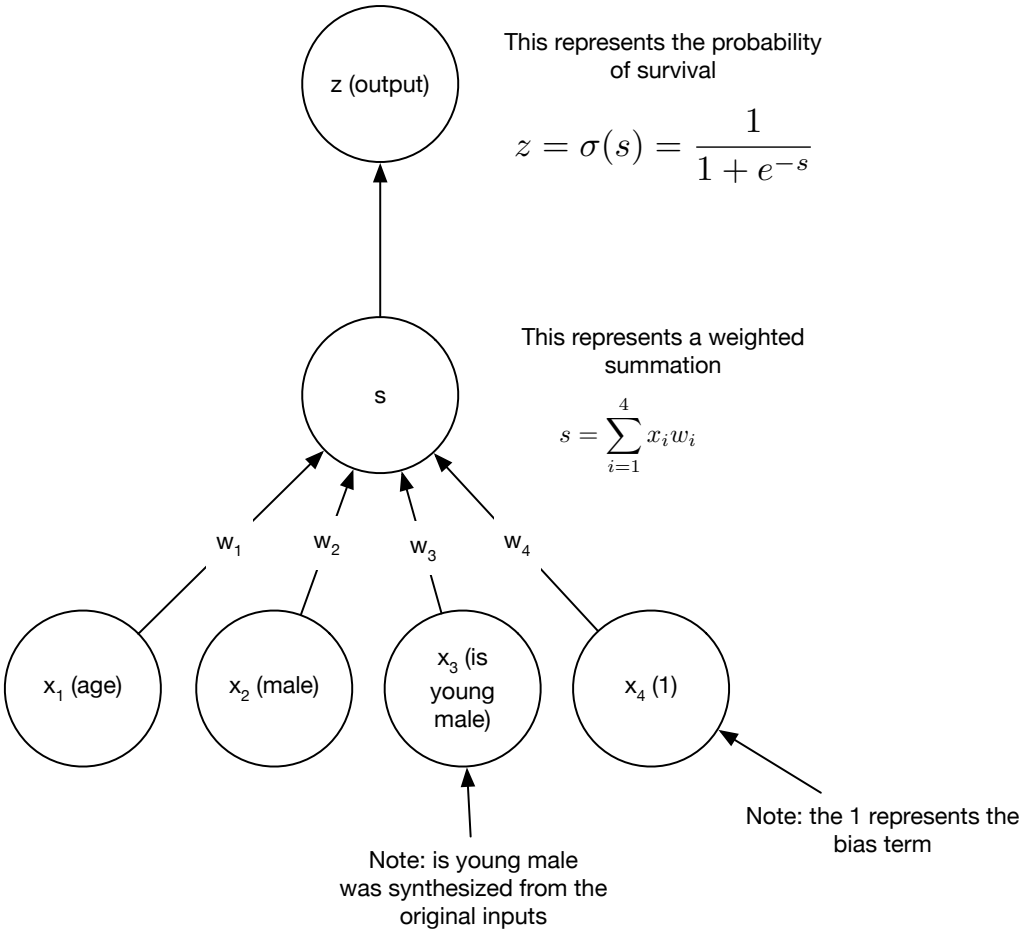
Go through the Assignment 6 Companion notebook.

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

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 that we used in the companion notebook known as a multilayer perceptron (MLP), 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. First we'll show the logistic regression model that we applied in the notebook.



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

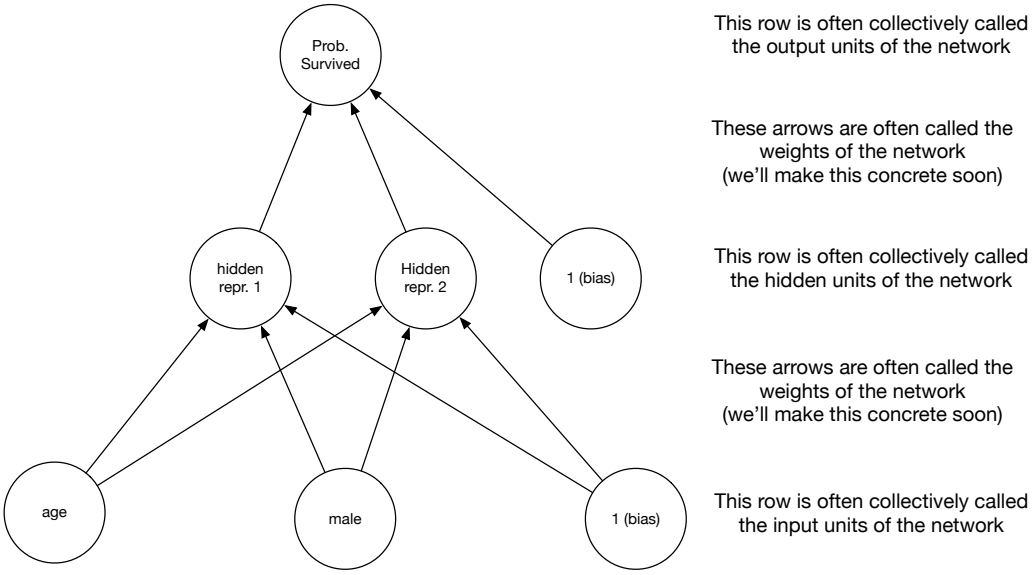
Notice how we had to manually introduce the feature `is young male` in order for the logistic regression model to utilize it to make its prediction. Before giving you the equivalent figure for the multi-layer perceptron, let's look at a little bit more cartoonish version of the multi-layer perceptron. 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)

(optional if what we give you below is not clicking) 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.**

- 3blue1brown What is a neural network?
- 3blue1brown video How neural networks learn?
- 3blue1brown Backpropagation

- Please add your own links via NB.



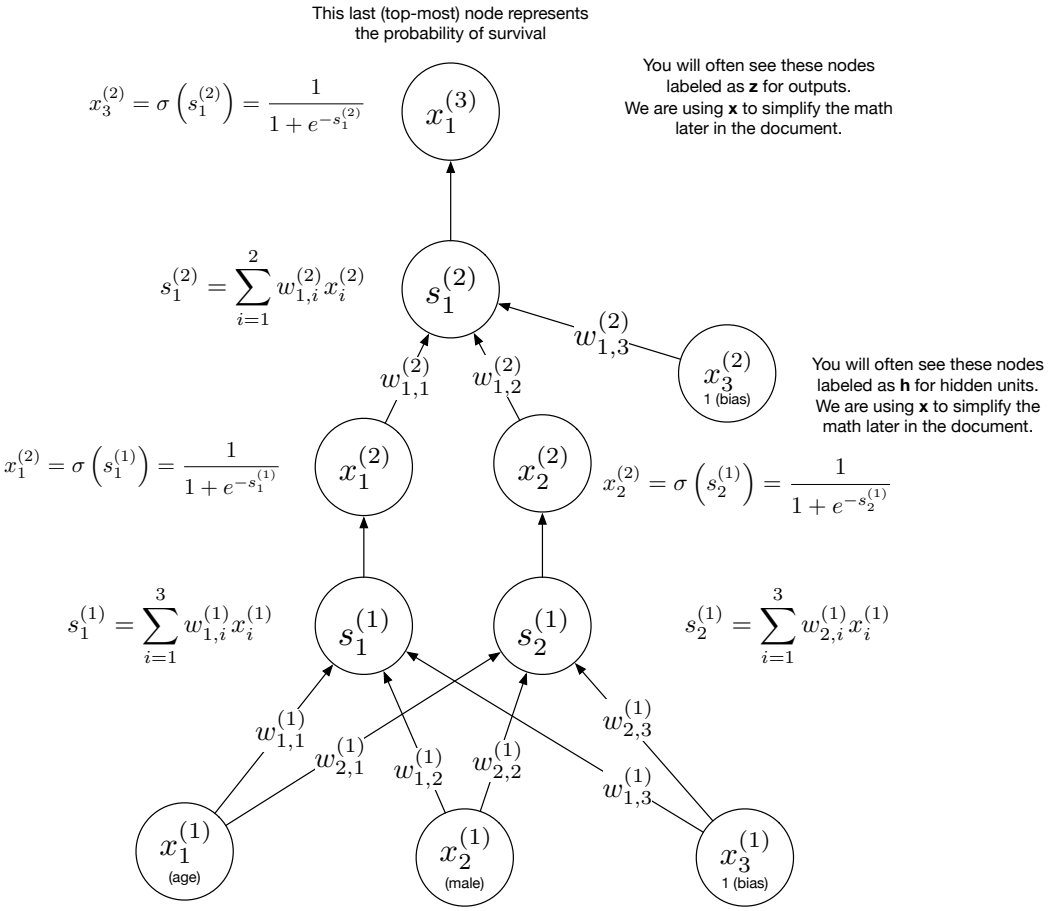
Input data (in this case we just use age, male, and a bias term) are propagated via a set of connection weights to a set of hidden representations. These hidden representations are propagated via another set of a connection weights to the output of the network. In the companion notebook we showed that for the Titanic dataset, the network learned two hidden representations: one that seemed to encode `is young male` and the another 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.
- $w_{i,k}^{(j)}$ will refer to the weight of the connection between the the kth unit in layer j and the ith unit in layer $j + 1$.

⚠ 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 up-front cost to introducing these symbols, it will ultimately make the derivation of the general form of the MLP easier.



Exercise 1 (10 minutes)

Before going on, let's make sure you have a firm handle on what's being represented in the figure above.

(a) Just as in logistic regression, we will try to tune the weights to fit the data. How many weights are there to tune in this network?

(b) While the figure looks pretty crazy, it has a lot of similarities with the logistic regression model. Where does the logistic regression model show up in the figure?

Exercise 2 (30 minutes)

Without using any training data (this is testing your understanding of the model itself), compute the weights in this network $(w_{1,1}^{(1)}, w_{1,2}^{(1)}, w_{1,3}^{(1)}, w_{2,1}^{(1)}, w_{2,2}^{(1)}, w_{2,3}^{(1)}, w_{1,1}^{(2)}, w_{1,2}^{(2)}, w_{1,3}^{(2)})$ such that the MLP has the following 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, $x_3^{(1)}$ and $x_3^{(2)}$ are always 1.

(a) $x_1^{(2)}$ 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).

(b) $x_2^{(2)}$ 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).

(c) $x_1^{(3)}$ 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 before we had algorithms for automatically tuning 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 will get more complicated, the learning algorithm and basic ideas will remain largely the same.

In order to prepare ourselves for the derivation of the backpropagation algorithm, we need to build up a new technique for 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 if we wanted to take all partial derivatives simultaneously) 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 you will eventually 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 back-propagation algorithm).

External Resource(s) (20 minutes)

This HMC calculus tutorials explain the concept of the chain rule using dataflow diagrams beautifully. Go and read the HMC Multivariable Chain Rule Page and come back for some exercises to test your understanding.

Exercise 3 (20 minutes)

(a) Draw a dataflow diagram to represent the function $f(x, y, z) = \cos(x^2 y) + x^2\sqrt{z}$. Compute $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$ using the dataflow diagram method.

(b) Draw a dataflow diagram to represent the function $f(\mathbf{x}) = (\mathbf{c}^\top \mathbf{x})^2$. Compute $\nabla_{\mathbf{x}} f$ using the dataflow diagram method. Hint: we're generalizing what is on the HMC page a bit. You can have vector quantities flowing through your diagram as well and all the ideas will carry over except you will have a gradient instead of a partial derivative.

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 $\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_n}$ with each $\mathbf{x_i}$ being a d-dimensional vector and training outputs $y_1, y_2, \ldots, y_n$ with each y_i being a binary number that indicates the class that the ith instance belongs to. Given this training data, our goal was to compute the best possible set of weights by solving the following optimization problem.

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \sum_{i=1}^{n} \left(-y_i \ln \sigma\left(\mathbf{w}^\top \mathbf{x_i}\right) - (1 - y_i) \ln\left(1 - \sigma\left(\mathbf{w}^\top \mathbf{x_i}\right)\right)\right) \tag{3}$$

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 1.

Notice

We have moved the weights from the arrows into circles. In the dataflow graph we will label the edges with partial derivatives (or gradients).

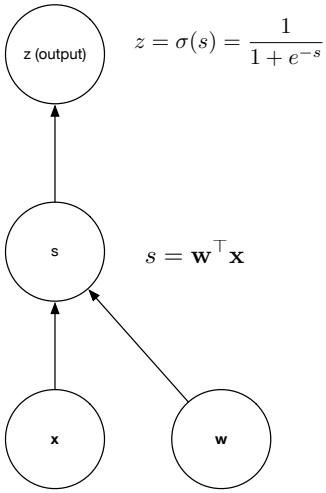


Figure 1: Dataflow in the logistic regression model

Exercise 4 (30 minutes)

Starting from the data flow representation of the logistic regression model shown in Figure 1, 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: remember 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.*. Note that we are only showing the path of one input $\mathbf{x}$ through the network. If you want to work with multiple inputs to take a gradient, all you need to do is sum over the gradient with respect to each training instance.

(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).e.

5.1 Forward Pass

⚠ Notice

The notation for the general case of 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.

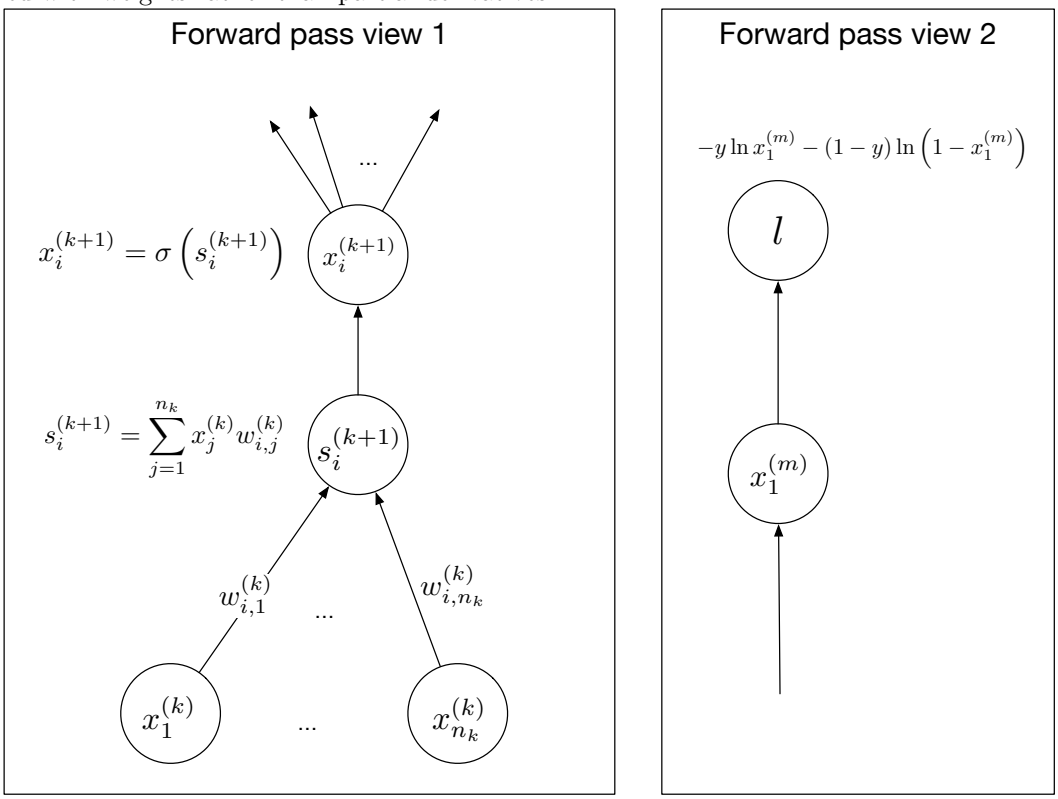
1. We'll allow for an arbitrary number of layers in the network. We'll call the number of layers m (the number of layers is also called the depth of the network. networks with large m are known as deep learning).

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 kth 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. You may see examples of MLP's with activation functions other

than sigmoid. The math to handle these other functions is very similar, so we are just using sigmoid here.

In the figure below we show how data propagates through the network in the forward direction. Note that this is not a dataflow diagram in the sense that the arrows are labeled with weights rather than partial derivatives.



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 mutlivariable 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 ith node in layer $k+1$ as $w_i^{(k)}$.

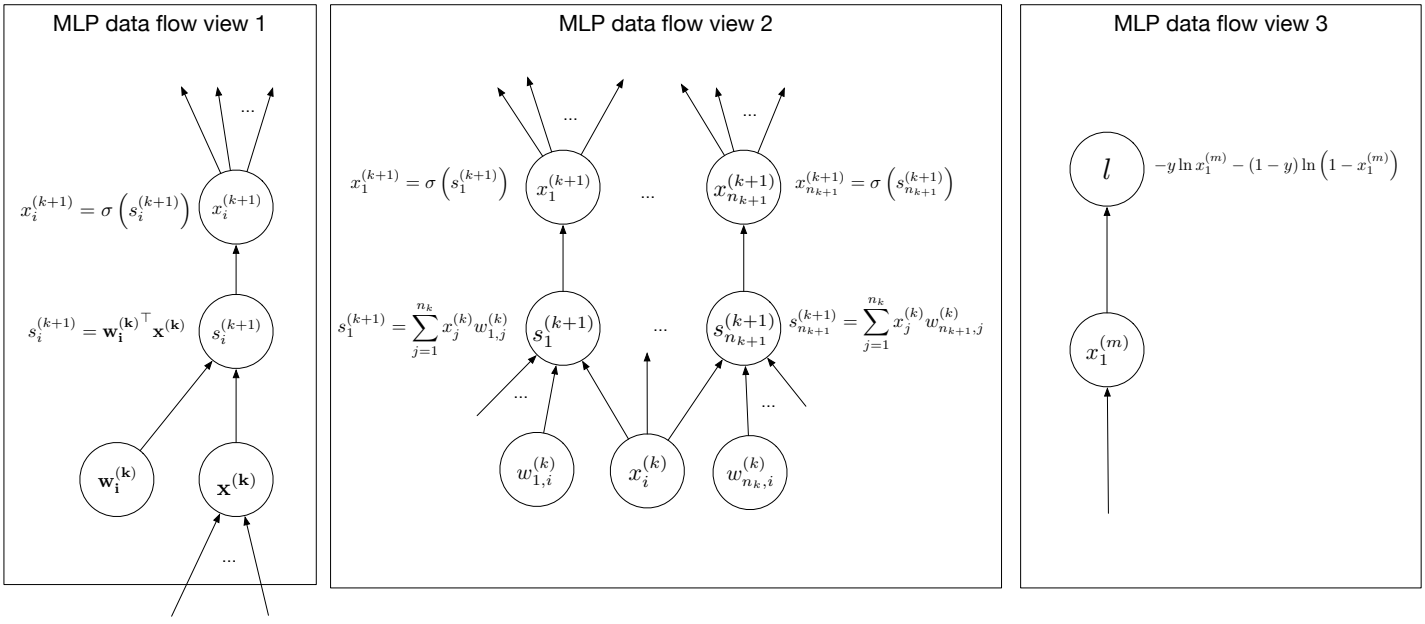


Figure 2: A data flow graph for the multi-layer perceptron.

Let's get to it.

Exercise 5 (75 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 2.

(a) 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 $\frac{\partial l}{\partial x_i^{(k+1)}}$.

(b) 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 $\frac{\partial l}{\partial x_1^{(k+1)}}, \ldots, \frac{\partial l}{\partial x_{n_{k+1}}^{(k+1)}}$.

(c) Referencing view 3, create a data flow graph and compute $\frac{\partial l}{\partial x_1^{(m)}}$.

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). Without defining the derivatives at lower layers in terms of higher layers, we would have been left to sum over all possible paths from some unit in the network to the loss. The number of paths would grow exponentially with the depth of the network. Here, we are able to compute derivatives without a significant computational expense. The learning rules for almost every other type of neural network are just variations on this same theme. This is incredibly powerful, 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.

Open in Colab

Assignment 6 Companion Notebook: Motivation for Neural Networks

Learning Objectives:

- Understand the limitations of logistic regression through an example
- Become familiar with feedforward, multi-layer neural networks
- Fit a neural network to a dataset and examine the results

Titanic: The Sequel

In order to motivate the need for neural networks, we're going to return to the Titanic dataset and examine a particular limitation of logistic regression.

We'll begin by loading the data.

In [0]:

```
import gdown
import pandas as pd

gdown.download('https://drive.google.com/uc?authuser=0&id=1XIFiL3WxxR6M2nWgADi3xWvuRO6A-Ov8&export=download', 'titanic_train.csv', False)
df = pd.read_csv('titanic_train.csv')
df
```

```
Downloading...
From: https://drive.google.com/uc?authuser=0&id=1XIFiL3WxxR6M2nWgADi3xWvuRO6A-Ov8&export=download
To: /content/titanic_train.csv
100%|██████████| 61.2k/61.2k [00:00<00:00, 36.7MB/s]
```

Out[0]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S
12	13	0	3	Saundercock, Mr. William Henry	male	20.0	0	0	A/5. 2151	8.0500	NaN	S
13	14	0	3	Andersson, Mr. Anders Johan	male	39.0	1	5	347082	31.2750	NaN	S
14	15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14.0	0	0	350406	7.8542	NaN	S

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
15	16	1	2	Amanda Adolfina Hewlett, Mrs. (Mary D Kingcome)	female	55.0	0	0	248706	16.0000	NaN	S
16	17	0	3	Rice, Master. Eugene	male	2.0	4	1	382652	29.1250	NaN	Q
17	18	1	2	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	13.0000	NaN	S
18	19	0	3	Vander Planke, Mrs. Julius (Emelia Maria Vande...	female	31.0	1	0	345763	18.0000	NaN	S
19	20	1	3	Masselmani, Mrs. Fatima	female	NaN	0	0	2649	7.2250	NaN	C
20	21	0	2	Fynney, Mr. Joseph J	male	35.0	0	0	239865	26.0000	NaN	S
21	22	1	2	Beesley, Mr. Lawrence	male	34.0	0	0	248698	13.0000	D56	S
22	23	1	3	McGowan, Miss. Anna "Annie"	female	15.0	0	0	330923	8.0292	NaN	Q
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.5000	A6	S
24	25	0	3	Palsson, Miss. Torborg Danira	female	8.0	3	1	349909	21.0750	NaN	S
25	26	1	3	Asplund, Mrs. Carl Oscar (Selma Augusta Emilia...	female	38.0	1	5	347077	31.3875	NaN	S
26	27	0	3	Emir, Mr. Farred Chehab	male	NaN	0	0	2631	7.2250	NaN	C
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.0000	C23 C25 C27	S
28	29	1	3	O'Dwyer, Miss. Ellen "Nellie"	female	NaN	0	0	330959	7.8792	NaN	Q
29	30	0	3	Todoroff, Mr. Lalio	male	NaN	0	0	349216	7.8958	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
861	862	0	2	Giles, Mr. Frederick Edward	male	21.0	1	0	28134	11.5000	NaN	S
862	863	1	1	Swift, Mrs. Frederick Joel (Margaret Welles Ba...	female	48.0	0	0	17466	25.9292	D17	S
863	864	0	3	Sage, Miss. Dorothy Edith "Dolly"	female	NaN	8	2	CA. 2343	69.5500	NaN	S
864	865	0	2	Gill, Mr. John William	male	24.0	0	0	233866	13.0000	NaN	S
865	866	1	2	Bystrom, Mrs. (Karolina)	female	42.0	0	0	236852	13.0000	NaN	S
866	867	1	2	Duran y More, Miss. Asuncion	female	27.0	1	0	SC/PARIS 2149	13.8583	NaN	C
867	868	0	1	Roebling, Mr. Washington Augustus II	male	31.0	0	0	PC 17590	50.4958	A24	S
868	869	0	3	van Melkebeke, Mr. Philemon	male	NaN	0	0	345777	9.5000	NaN	S
869	870	1	3	Johnson, Master. Harold Theodor	male	4.0	1	1	347742	11.1333	NaN	S
870	871	0	3	Balkic, Mr. Cerin	male	26.0	0	0	349248	7.8958	NaN	S
871	872	1	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47.0	1	1	11751	52.5542	D35	S
872	873	0	1	Carlsson, Mr. Frans Olof	male	33.0	0	0	695	5.0000	B51 B53 B55	S
873	874	0	3	Vander Cruyssen, Mr. Victor	male	47.0	0	0	345765	9.0000	NaN	S
874	875	1	2	Abelson, Mrs. Samuel (Hannah Wizosky)	female	28.0	1	0	P/PP 3381	24.0000	NaN	C
875	876	1	3	Najib, Miss. Adele Kiamie "Jane"	female	15.0	0	0	2667	7.2250	NaN	C
876	877	0	3	Gustafsson, Mr. Alfred Ossian	male	20.0	0	0	7534	9.8458	NaN	S
877	878	0	3	Petroff, Mr. Nedelio	male	19.0	0	0	349212	7.8958	NaN	S
878	879	0	3	Laleff, Mr. Kristo	male	NaN	0	0	349217	7.8958	NaN	S
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	11767	83.1583	C50	C
880	881	1	2	Shelley, Mrs. William (Imanita Parrish Hall)	female	25.0	0	1	230433	26.0000	NaN	S
881	882	0	3	Markun, Mr. Johann	male	33.0	0	0	349257	7.8958	NaN	S

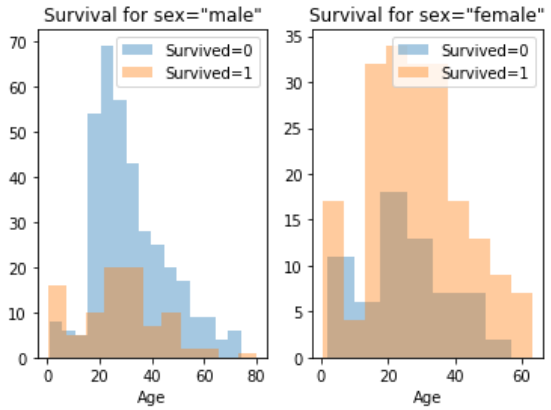
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
882	883	0	3	Dahlberg, Miss. Gerda Ulrika	female	22.0	0	0	7552	10.5167	NaN	S
883	884	0	2	Banfield, Mr. Frederick James	male	28.0	0	0	C.A./SOTON 34068	10.5000	NaN	S
884	885	0	3	Sutehall, Mr. Henry Jr	male	25.0	0	0	SOTON/OQ 392076	7.0500	NaN	S
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.1250	NaN	Q
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

From some of the exploratory analysis in the assignment 5 walkthrough, we know that both age and sex are good predictors of whether or not someone survived the Titanic sinking. That said, we saw that the effect surivival rate for women wasn't dramatically changed by age, but the survival rate for men was dramatically changed. Here are some plots showing this result.

In [0]:

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
legend_entries = []
plt.subplot(1,2,1)
for groups in df[df['Sex'] == 'male'].groupby('Survived'):
    sns.distplot(groups[1]['Age'].dropna(), kde=False)
    legend_entries.append('Survived=%d'% groups[0])
plt.legend(legend_entries)
plt.title('Survival for sex="male"')

plt.subplot(1,2,2)
legend_entries = []
for groups in df[df['Sex'] == 'female'].groupby('Survived'):
    sns.distplot(groups[1]['Age'].dropna(), kde=False)
    legend_entries.append('Survived=%d'% groups[0])
plt.legend(legend_entries)
plt.title('Survival for sex="female"')
plt.show()

# while we're at it, silence annoying sklearn warnings
# import warnings filter
from warnings import simplefilter
# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)
```



It would be great if we could use logistic regression in order to leverage this information. A good first pass would be to take both of these features (where we will encode `sex` using an `is_male` feature) and pass them into a logistic regression. The plot below

shows the results of this analysis where we represent the training data along with the lines of equal probability for the resultant model. We'll divide this into two cells, one where we prepare the data and the other where we fit the model and show the plot

In [0]:

```
# get rid of null values for age since this is just an illustrative example.
# this would not be a good thing to do if we were trying to evalutate the
# performance of a model.
df_filtered = df[['Age', 'Sex', 'Survived']].dropna()
experiment_1_data = pd.concat((pd.get_dummies(df_filtered['Sex'], drop_first=True),
df_filtered['Age']), axis=1)
experiment_1_outputs = df_filtered['Survived']
experiment_1_data
```

Out[0]:

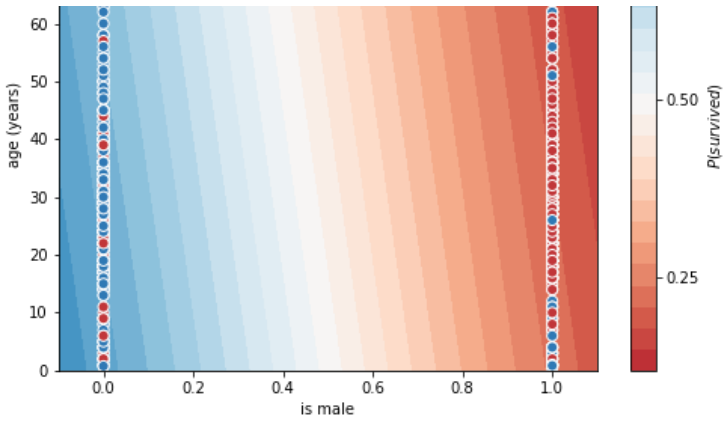
	male	Age
0	1	22.0
1	0	38.0
2	0	26.0
3	0	35.0
4	1	35.0
6	1	54.0
7	1	2.0
8	0	27.0
9	0	14.0
10	0	4.0
11	0	58.0
12	1	20.0
13	1	39.0
14	0	14.0
15	0	55.0
16	1	2.0
18	0	31.0
20	1	35.0
21	1	34.0
22	0	15.0
23	1	28.0
24	0	8.0
25	0	38.0
27	1	19.0
30	1	40.0
33	1	66.0
34	1	28.0
35	1	42.0
37	1	21.0
38	0	18.0
...	...	...
856	0	45.0
857	1	51.0
858	0	24.0
860	1	41.0
861	1	21.0
862	0	48.0

	male	Age
862	0	48.0
864	1	24.0
865	0	42.0
866	0	27.0
867	1	31.0
869	1	4.0
870	1	26.0
871	0	47.0
872	1	33.0
873	1	47.0
874	0	28.0
875	0	15.0
876	1	20.0
877	1	19.0
879	0	56.0
880	0	25.0
881	1	33.0
882	0	22.0
883	1	28.0
884	1	25.0
885	0	39.0
886	1	27.0
887	0	19.0
889	1	26.0
890	1	32.0

714 rows × 2 columns

In [0]:

```
from sklearn.linear_model import LogisticRegression
import numpy as np

model = LogisticRegression()
model.fit(experiment_1_data, experiment_1_outputs)

xx, yy = np.mgrid[-.1:1.1:.01, 0:85:.1]
grid = np.c_[xx.ravel(), yy.ravel()]
probs = model.predict_proba(grid)[:, 1].reshape(xx.shape)

f, ax = plt.subplots(figsize=(8, 6))
contour = ax.contourf(xx, yy, probs, 25, cmap="RdBu",
                      vmin=0, vmax=1)
ax_c = f.colorbar(contour)
ax_c.set_label("$P(survived)$")
ax_c.set_ticks([0, .25, .5, .75, 1])

ax.scatter(experiment_1_data['male'], experiment_1_data['Age'], c=experiment_1_outputs, s=50,
           cmap="RdBu", vmin=-.2, vmax=1.2,
           edgecolor="white", linewidth=1)

ax.set(xlim=(-.1, 1.1),
       ylim=(0, 85),
       xlabel="is male", ylabel="age (years)")
plt.show()
```



Notebook Exercise 1 (10 minutes)

(a) In English, explain what this graph is showing (defer interpretation for now, we just want to make sure you have an idea of what is represented in the plot, e.g., x-axis, y-axis, colored areas).

(b) Given the decision boundary, how does the model predict who will survive versus not survive? What does it predict for male babies?

Adding the `is_young_male` feature

Based on this plot it seems that what we need is to have a model that is able to simulataneously predict that all women survived and that all very young boys survived. One way we can achieve this is by engineering a special `is_young_male` feature that is 1 if the person is under the age of 5 and male. The feature will take a value of 0 otherwise.

In [0]:

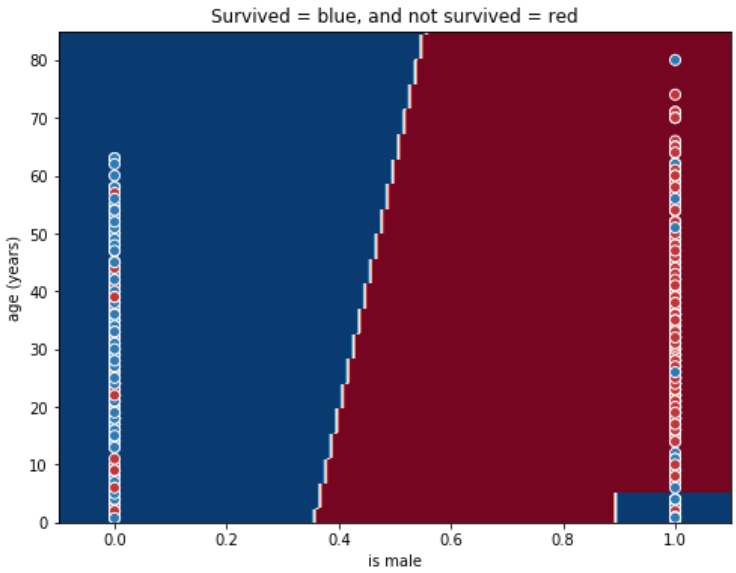
```
is_young_male = (experiment_1_data['male']) & (experiment_1_data['Age'] < 5).astype(int)
is_young_male.name = 'is_young_male'
experiment_2_data = pd.concat((experiment_1_data, is_young_male), axis=1)
experiment_2_data
# these don't change, but just to be consistent with variable naming
experiment_2_outputs = experiment_1_outputs
```

Next, we can take fit the model to this new dataset and create a plot that shows the results of fitting the model. To make the plot a bit easier to interpret, we'll just plot the model's binary output (0 or 1) rather than the probability. This will make it really clear what the model is doing with the points in the lower righthand corner.

In [0]:

```
model = LogisticRegression()
model.fit(experiment_2_data, experiment_1_outputs)

xx, yy = np.mgrid[-.1:1.1:.01, 0:85:.1]
grid = np.c_[xx.ravel(), yy.ravel()]
# Note: we are setting the is_young_male feature for a male value above 0.9 to
# aid in visualization.  Of course, the feature can only take on value 0 or 1
is_young_male_grid = np.logical_and(grid[:,0] >= 0.9, grid[:,1] < 5).astype(np.int)[:,np.newaxis]

grid = np.hstack((grid, is_young_male_grid))
outputs = model.predict(grid).reshape(xx.shape)

f, ax = plt.subplots(figsize=(8, 6))
contour = ax.contourf(xx, yy, outputs, 25, cmap="RdBu",
                      vmin=0, vmax=1)
ax.set_title('Survived = blue, and not survived = red')

ax.scatter(experiment_1_data['male'], experiment_1_data['Age'], c=experiment_2_outputs, s=50,
           cmap="RdBu", vmin=-.2, vmax=1.2,
           edgecolor="white", linewidth=1)

ax.set(xlim=(-.1, 1.1),
       ylim=(0, 85),
       xlabel="is male", ylabel="age (years)")
```

```
plt.show()
```



Notebook Exercise 2 (10 minutes)

(a) Based on this graph, given a passenger's sex and age, what would the model predict?

(b) This seems to have achieved our goal of predicting that young males survived. What are the limitations of this approach of hand coding these sorts of features?

Enter the Neural Network!

Next, we're going to learn about a particular type of neural network called a multilayer perceptron (you'll know exactly what one is by the end of this assignment!). For now, we won't give you a very sophisticated mental model of what this neural network is doing. Just think of it as automating the process of discovering useful representations for learning. We are trying to avoid needing to hand engineer features, such as we did with the `is young male` feature.

In [0]:

```
from sklearn.neural_network import MLPClassifier

model = MLPClassifier(hidden_layer_sizes=(2), activation='logistic', solver='lbfgs', random_state=5
0)
model.fit(experiment_1_data, experiment_1_outputs)
```

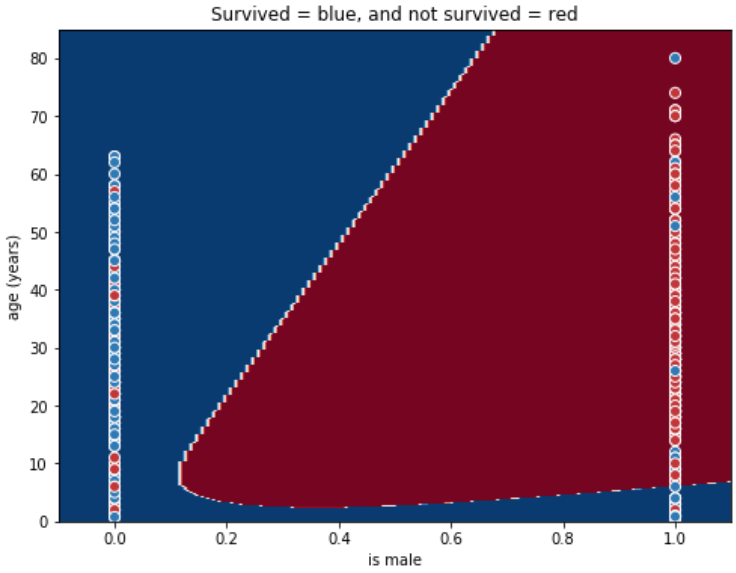
Out[0]:

```
MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
              beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=2, learning_rate='constant',
              learning_rate_init=0.001, max_iter=200, momentum=0.9,
              n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
              random_state=50, shuffle=True, solver='lbfgs', tol=0.0001,
              validation_fraction=0.1, verbose=False, warm_start=False)
```

In [0]:

```
xx, yy = np.mgrid[-.1:1.1:.01, 0:85:.1]
grid = np.c_[xx.ravel(), yy.ravel()]
outputs = model.predict(grid).reshape(xx.shape)

f, ax = plt.subplots(figsize=(8, 6))
contour = ax.contourf(xx, yy, outputs, 25, cmap="RdBu",
                      vmin=0, vmax=1)
ax.set_title('Survived = blue, and not survived = red')

ax.scatter(experiment_1_data['male'], experiment_1_data['Age'], c=experiment_2_outputs, s=50,
           cmap="RdBu", vmin=-.2, vmax=1.2,
```

```
            edgecolor="white", linewidth=1)

ax.set(xlim=(-.1, 1.1),
       ylim=(0, 85),
       xlabel="is male", ylabel="age (years)")
plt.show()
```



Notebook Exercise 3 (5 minutes)

Based on this graph, what does the network predict about whether a passenger will survive or not survive?

Examining the Intermediate Representations in the Network

While we have yet to really unpack *how* the neural network was able to achieve this feat, we can start to interrogate the learned model to undrestand a bit of what it is doing.

For the purposes of this next set of plots and exercise, you should have the following mental model of what the network is doing (all of this will be made 100% precise when you go through the rest of the assignmnet document, but for now things will be explained on a conceptual level).

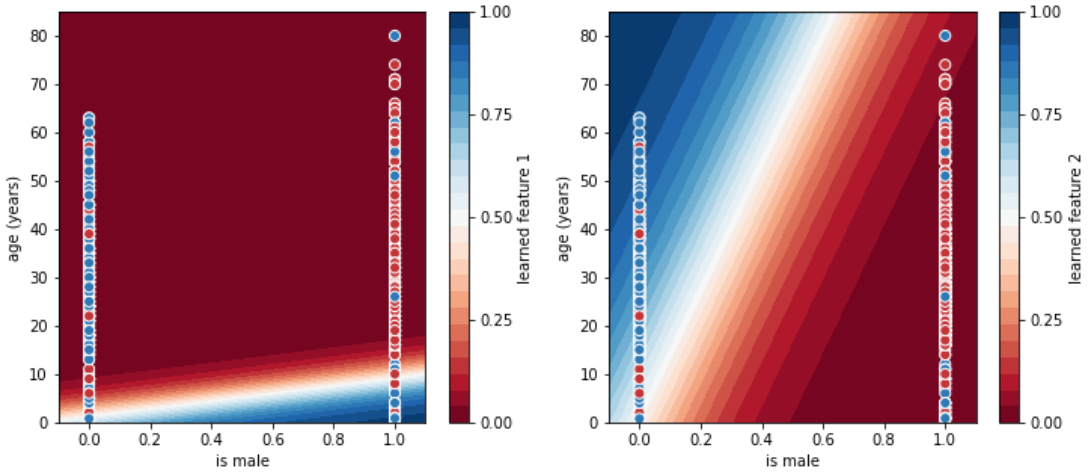
- Neural networks learn internal representations of the input data that help them make predictions (you may recall that this was Big Idea #4 in assignment 1).
- In this case we instructed the neural network to learn exactly 2 internal representations of the input data.
- The network will use these two internal representations, not the original input data, in order to arrive at its final decision.

In the plot below, we show the values for each of the 2 learned internal representations in the network. You can think of these representations as providing a remapping of the data into a new space that is then used to make the prediction as to whether the person was likely to surive or not survive.

In [0]:

```
hidden_units = 1/(1+np.exp(-np.matmul(np.array(experiment_1_data), model.coefs_[0]) + model.interce
pts_[0]))

f = plt.figure(figsize=(12, 5))
hidden_units = 1/(1+np.exp(-np.matmul(grid, model.coefs_[0]) + model.intercepts_[0]))

for i in range(2):
    ax = f.add_subplot(1,2,i+1)
    contour = ax.contourf(xx, yy, hidden_units[:,i].reshape(xx.shape), 25, cmap="RdBu",
                          vmin=0, vmax=1)

    ax.scatter(experiment_1_data['male'], experiment_1_data['Age'], c=experiment_2_outputs, s=50,
               cmap="RdBu", vmin=-.2, vmax=1.2,
               edgecolor="white", linewidth=1)
    ax_c = f.colorbar(contour)
    ax_c.set_label("learned feature %d" %(i+1))
    ax_c.set_ticks([0, .25, .5, .75, 1])
```

```
    ax.set(xlim=(-.1, 1.1),
           ylim=(0, 85),
           xlabel="is male", ylabel="age (years)")
plt.show()
```



Notebook Exercise 4 (10 minutes)

(a) What does first learned feature (left plot) appear to encode?

(b) What does the second learned feature (right plot) appear to encode?

Key takeaways

While this was a relatively simple example of how a neural network could be applied, the capabilities that you just saw in this notebook have immense significance. The ability of a neural network to take input data that may not be suited for prediction (e.g., sex and age) and transform it into a representation that is more useful for prediction is perhaps the most significant aspect of neural networks. Here is a summmary of what happened in this notebook.

- We reexamined the Titanic dataset and showed that adding the `age` feature doesn't really help in making predictions versus just using `sex`.
- We showed that we can manually add a feature called `is_young_male` that can help us in prediction.
- We showed that neural networks can automate this process of feature learning by developing their own internal representations.
- We showed that the neural network, in this case, learned an internal representation that is very similar to the `is_young_male` and `sex` representation.

In the rest of the assignment we'll be unpacking what exactly a neural network is. You'll learn how it functions and how you would fit the parameters of one.