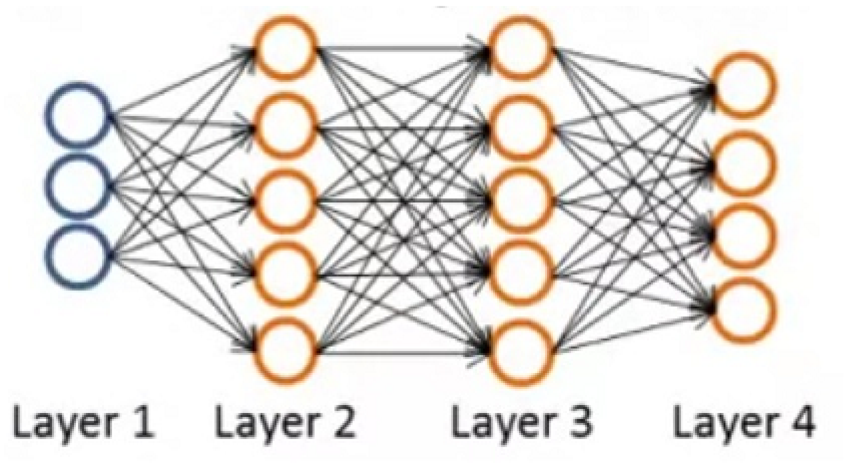
Neural Network (Classification)

We're going to focus on the application of neural networks to classification problems. So suppose we have a network



And we have a training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$ pairs of training examples:

$L =$ total no. of layers in network (for this example $L = 4$).

$S_l =$ no. of units (not counting bias unit) int layer l (in our example $S_1 = 3$, $S_2 = 5$, $S_3 = 5$, $S_4 = S_L = 4$).

We're going to consider two types of classification problems:

Binary Classification

- The first is **Binary classification**, where the labels y are either 0 or 1. In this case, we will have only one output unit (and the output of the neural network would be $h_\Theta(x) \in \mathbb{R}$ is going to be a real number), in this case the number of output units, S_L, (where L is the index of the final layer) the number of units we have in the output layer is going $S_L = 1$ (or $K = 1$ where K also denoting the number of units in the output layer).

Multi-class classification (K classes)

- The second type of classification problem we'll consider will be **multi-class classification** problem where we may have K distinct classes.

$$y \in \mathbb{R}^K \quad \text{E.g.} \underset{\text{pedestrian}}{\begin{bmatrix}1\\0\\0\\0\end{bmatrix}}, \underset{\text{car}}{\begin{bmatrix}0\\1\\0\\0\end{bmatrix}}, \underset{\text{motorcycle}}{\begin{bmatrix}0\\0\\1\\0\end{bmatrix}}, \underset{\text{truck}}{\begin{bmatrix}0\\0\\0\\1\end{bmatrix}}$$

In this case we will have K output units, and our hypothesis or our output vector will be a K dimensional vector ($h_\theta(x) \in \mathbb{R}^k$) and the number of output units will be equal to K ($S_L = k$). Usually we would have $K \geq 3$, because if we had two causes, then we don't need to use the one versus all method. We use the one versus all method only if we have $K \geq 3$ classes, so having only two classes we will need to use only one output unit.

Cost Function - Logistic Regression

The cost function we use for the neural network is going to be a generalization of the one that we use for logistic regression. For logistic regression we used to minimize the cost function $J(\theta)$:

$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \ \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \ \log(1 - h_\theta(x^{(i)}))] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$

And we add the regularization term where this was a sum of theta from $j = 1$ through n, because we did not regularize the bias term θ_0. For a neural network, our cost function is going to be a generalization of this. Where instead of having basically just one, which is the compression output unit, we may instead have K of them.

Cost Function - Neural Network

Our neural network now outputs vectors in $\mathbb{R}^k$ where K might be equal to 1 if we have a binary classification problem ($h_\Theta(x) \in \mathbb{R}^k$). We're going to use the following notation: $(h_\Theta(x))_i$, to denote the i^{th} output. That is, $h_\Theta(x)$ is a k-dimensional vector and so the subscript i in $(h_\Theta(x))_i$ just selects out the i^{th} element of the vector $\mathbb{R}^k$ that is output by our neural network.

For neural networks our cost function is going to be the following ecuation:

$J(\Theta) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} [y_k^{(i)} \ \log((h_\Theta(x^{(i)}))_k) + (1 - y_k^{(i)}) \ \log(1 - (h_\Theta(x^{(i)}))_k)]$

$+\frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{S_l} \sum_{j=1}^{S_l+1} (\Theta_{j,i}^{(l)})^2$

The first term (Logistic regression cost function)
$-\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} [y_k^{(i)} \ \log((h_\Theta(x^{(i)}))_k) + (1 - y_k^{(i)}) \ \log(1 - (h_\Theta(x^{(i)}))_k)]$ is $-\frac{1}{m}$ of a sum of a similar term to what we have for logistic regression, except that we have the sum from $K = 1$ through K. This summation is basically a sum over our K output. So if we had 4 output units then the sum is $k = 1$ to 4 of the logistic regression over each of the four output units in turn.

This is just saying:

- For each training data example (i.e. 1 to m - the first summation)
  - Sum for each position in the output vector

Finally, the second term $\frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{S_l} \sum_{j=1}^{S_l+1} (\Theta_{j,i}^{(l)})^2$ is the regularization term similar to what we had for the logistic regression. This summation term looks really complicated, but all it's doing is it's summing over the terms theta j, i, l for all values of i, j and l.

Except that we don't sum over the terms corresponding to these bias values like we have for logistic progression. Concretely, we don't sum over the terms responding to where i is equal to 0.

Video Question: Suppose we want to try to minimize $J(\Theta)$ as a function of Θ, using one of the advanced optimization methods (fminunc, conjugate gradient, BFGS, L-BFGS, etc.). What do we need to supply code to compute (as a function of Θ)?

- Θ
- $J(\Theta)$
- The (partial) derivative terms $\frac{\partial}{\partial \Theta_{ij}^{(l)}}$ for every i, j, l

> $J(\Theta)$ and the (partial) derivative terms $\frac{\partial}{\partial \Theta_{ij}^{(l)}}$ for every i, j, l

Summary

Let's first define a few variables that we will need to use:

- L = total number of layers in the network
- s_l = number of units (not counting bias unit) in layer l

- K = number of output units/classes

Recall that in neural networks, we may have many output nodes. We denote $h_\Theta(x)_k$ as being a hypothesis that results in the k^{th} output. Our cost function for neural networks is going to be a generalization of the one we used for logistic regression. Recall that the cost function for regularized logistic regression was:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)}\ \log(h_\theta(x^{(i)})) + (1 - y^{(i)})\ \log(1 - h_\theta(x^{(i)}))] + \frac{\lambda}{2m}\sum_{j=1}^n \theta_j^2$$

For neural networks, it is going to be slightly more complicated:

$$J(\Theta) = -\frac{1}{m} \sum_{i=1}^m \sum_{k=1}^K [y^{(i)}_k \ \log((h_\Theta(x^{(i)}))_k) + (1 - y^{(i)}_k)\ \log(1 - (h_\Theta(x^{(i)}))_k)]$$

$$+ \frac{\lambda}{2m}\sum_{l=1}^{L-1} \sum_{i=1}^{S_l} \sum_{j=1}^{S_l+1} (\Theta_{j,i}^{(l)})^2$$

We have added a few nested summations to account for our multiple output nodes. In the first part of the equation, before the square brackets, we have an additional nested summation that loops through the number of output nodes.

In the regularization part, after the square brackets, we must account for multiple theta matrices. The number of columns in our current theta matrix is equal to the number of nodes in our current layer (including the bias unit). The number of rows in our current theta matrix is equal to the number of nodes in the next layer (excluding the bias unit). As before with logistic regression, we square every term.

Note:

- the double sum simply adds up the logistic regression costs calculated for each cell in the output layer.
- the triple sum simply adds up the squares of all the individual Θ's in the entire network.
- the i in the triple sum does **not** refer to training example i.