Machine Learning - Week 4

赵燕

目录

1 Motivations

1.1 Non-linear Hypotheses

我们之前学过的线性回归和逻辑回归都有一个缺点：当特征量太多时，计算的负荷会非常大。

例如：

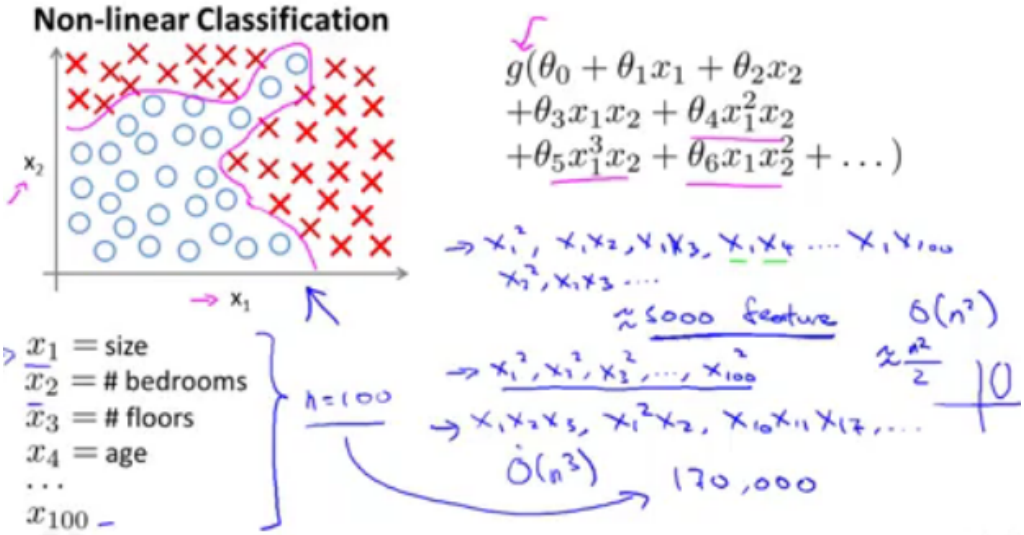


图 1: 举例说明

当我们使用x_1和x_2的多项式进行预测时，我们可以应用的很好。

之前我们已经看到过,使用非线性的多项式项,能够帮助我们建立更好的分类模型。假设我们有非常多的特征,例如大于 100 个变量,我们希望用这 100 个特征来构建一个非线性 的多项式模型,结果将是数量非常惊人的特征组合,即便我们只采用两两特征的组合($x_1x_2 + x_1x_3 + x_1x_4 + ... + x_2x_3 + x_2x_4 + ... + x_99x_100$) ,我们也会有接近 5000个组合而成的特征。这对于一般的逻辑回归来说需要计算的特征太多了。

假设我们希望训练一个模型来识别视觉对象（例如识别一张图片上是否是一辆汽车），我么怎样才能这么做呢？一种方法是我们利用很多汽车的图片和很多非汽车的图片，然后利用这些图片上的一个个像素的值（饱和度或亮度）来作为特征。

假如我们只使用灰度图片，每个像素则只有一个值（而非 RGB值），我们可以选取图片上的两个不同位置的两个像素，然后训练一个逻辑回归算法，利用这两个像素的值来判断图片上是否是汽车：

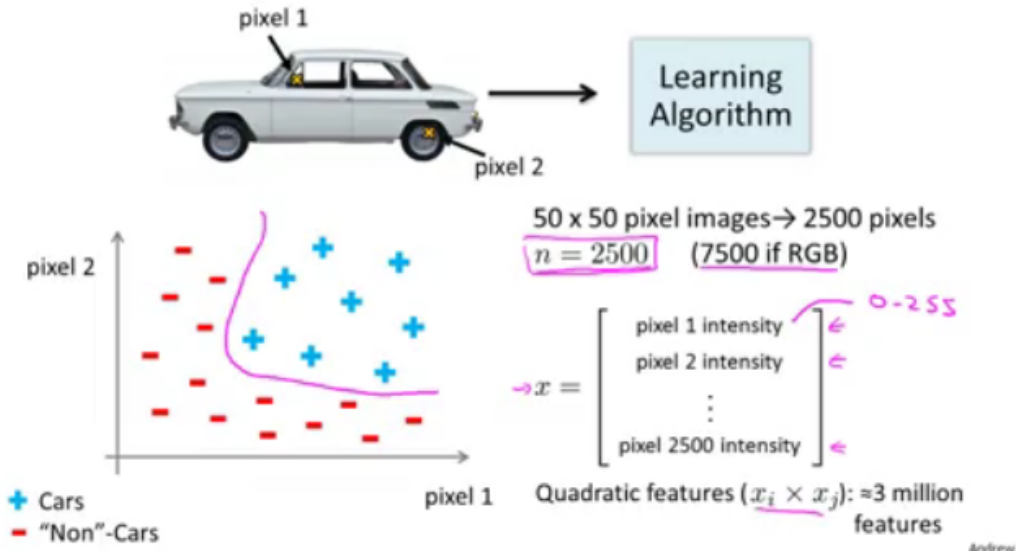


图 2: 计算机视觉-汽车模型

假使我们采用的都是50x50像素的小图片,并且我们将所有的像素视为特征,则会有2500个特征,如果我们要进一步将两两特征组合构成一个多项式模型,则会有约$\frac{2500^2}{2}$ 个（接近3百万个）特征。普通的逻辑回归模型,不能有效地处理这么多的特征,这时候我们需要神经网络。

1.2 Neurons and the Brain

神经网络是一种很古老的算法，它最初产生的目的是制造能模拟大脑的机器。它是计算量有些偏大的算法，然而大概由于近些年计算机的运行速度变快，才足以运行起大规模的神经网络。当想模拟大脑时，是指想制造出与人类大脑作用效果相同的机器。大脑可以学会去以看而不是听的方式处理图像，学会处理我们的触觉。

2 Neural Networks

2.1 Model Representation I

为了构建神经网络模型，我们首先需要思考大脑中的神经网络是怎样的？每一个神经元都可以被认为是一个处理单元/神经核（processing unit/Nucleus），它含有许多输入神经/树突（input/Dendrite），并且有一个输出神经/轴突（output/Axon）。神经网络是大量神经元相互连接并通过电脉冲来交流的一个网络。

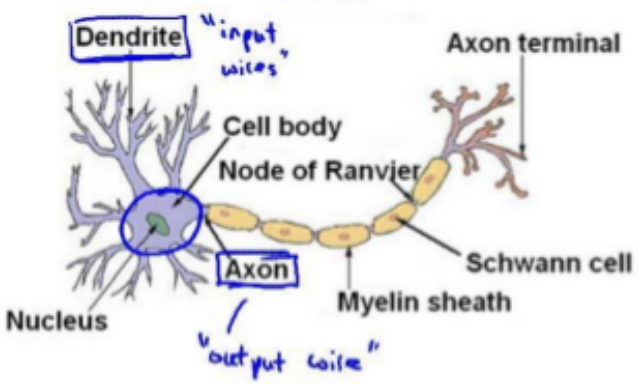


图 3: 大脑中的神经网络

神经网络模型建立在很多神经元之上，每一个神经元又是一个个学习模型。这些神经元（也叫激活单元,antivation unit）采纳一些特征作为输出，并且根据本身的模型提供一个输出。下图是一个以逻辑回归模型作为自身学习模型的神经元示例，在神经网络中，参数又可以被称为权重（weight）。

In this situation, our "theta" parameters are sometimes called "weights".

Visually, a simplistic representation looks like:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \rightarrow [\ \ ] \rightarrow h_\theta(x)$$

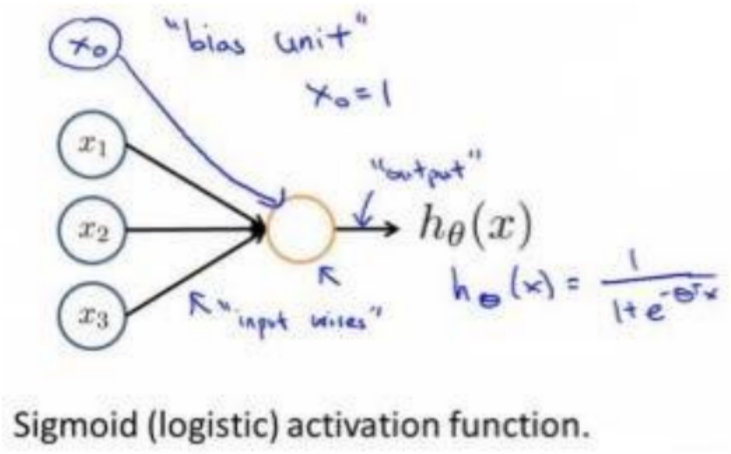


图 4: 神经网络简单模型

Our input nodes (layer 1), also known as the "input layer", go into another node (layer 2), which finally outputs the hypothesis function, known as the "output layer".

We can have intermediate layers of nodes between the input and output layers called the "hidden layers."

In this example, we label these intermediate or "hidden" layer nodes a02an2 and call them "activation units."

$a_i^{(j)}$="activation" of unit i in layer j

$\theta^{(j)}$=matrix of weight controlling function mapping from layer j to layer j+1.

If we have one hidden layer.it would look like:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \rightarrow \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \end{bmatrix} \rightarrow h_\theta(x)$$

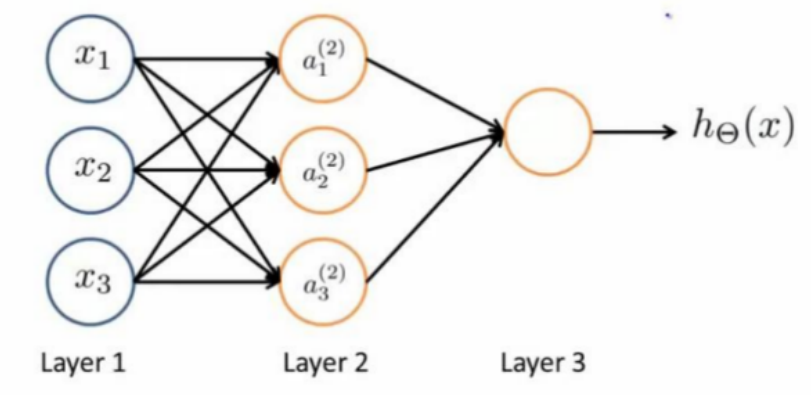


图 5: 神经网络模型

（1）输入单元x_1，x_2，x_3：将原始数据输入给它们

（2）中间单元a_1，a_2，a_3：负责将数据处理，然后呈递到下一层。

（3）输出单元：负责计算$h_\theta(x)$

神经网络模型是许多逻辑单元按照不同层级组织起来的网络,每一层的输出变量都是下一层的输入变量。下图为一个 3 层的神经网络,第一层成为输入层(Input Layer),最后一 层称为输出层(Output Layer),中间一层成为隐藏层(Hidden Layers)。我们为每一层都增加一个偏差单位(bias unit):

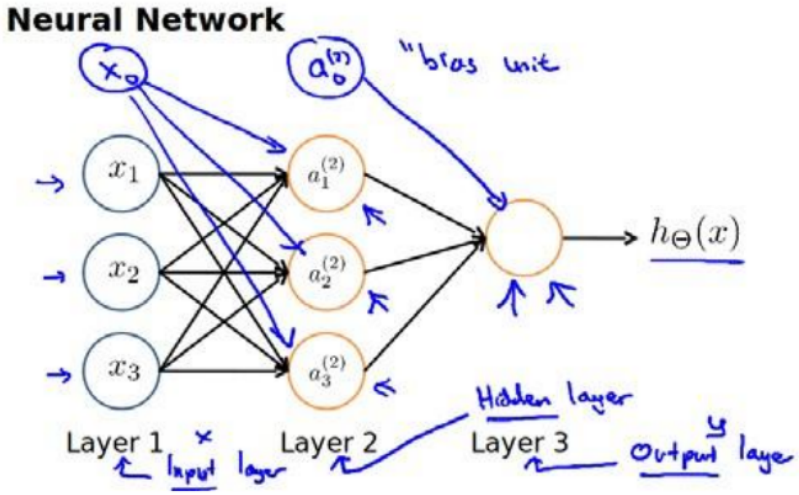


图 6: 神经网络模型

描述模型：

$a_i^{(j)}$代表第j层的第i个激活单元。$\theta^{(j)}$代表从第j层映射到第j+1层时的权重的矩阵，例如$\theta^{(1)}$代表从第一层映射到第二层的权重的矩阵。其尺寸是：以第j+1层的激活单元数量为行数，以第j层的激活单元数加一为列数的矩阵。例如：上图所示的神经网络的$\theta^{(1)}$的尺寸为：3*4。

对于上图所示的模型，激活单元和输出分别表达为：

$$a_1^{(2)} = g(\theta_{10}^{(1)}x_0 + \theta_{11}^{(1)}x_1 + \theta_{12}^{(1)}x_2 + \theta_{13}^{(1)}x_3) \tag{1}$$

$$a_2^{(2)} = g(\theta_{20}^{(1)}x_0 + \theta_{21}^{(1)}x_1 + \theta_{22}^{(1)}x_2 + \theta_{23}^{(1)}x_3) \tag{2}$$

$$a_3^{(2)} = g(\theta_{30}^{(1)}x_0 + \theta_{31}^{(1)}x_1 + \theta_{32}^{(1)}x_2 + \theta_{33}^{(1)}x_3) \tag{3}$$

$$h_\theta(x) = g(\theta_{10}^{(2)}a_0^{(2)} + \theta_{11}^{(2)}a_1^{(2)} + \theta_{12}^{(2)}a_2^{(2)} + \theta_{13}^{(2)}a_3^{(2)}) \tag{4}$$

This is saying that we compute our activation nodes by using a 3×4 matrix of parameters. We apply each row of the parameters to our inputs to obtain the value for one activation node. Our hypothesis output is the logistic function applied to the sum of the values of our activation nodes, which have been multiplied by yet another parameter matrix $\theta^{(2)}$ containing the weights for our second layer of nodes.

Each layer gets its own matrix of weight,$\theta^{(j)}$.

The dimensions of these matrixes of weights is determined as follows:

If network has s_j units in layer and s_{j+1} units in layer j+1,then $\theta^{(j)}$ will be of dimension $s_{j+1} * (s_j + 1)$.

The +1 comes from the addition in $\theta^{(j)}$ of the "bias nodes," x_0 and $\theta^{(j)}$.In other words the output nodes will not include the bias nodes while the inputs will.The following image summarizes our model representation:

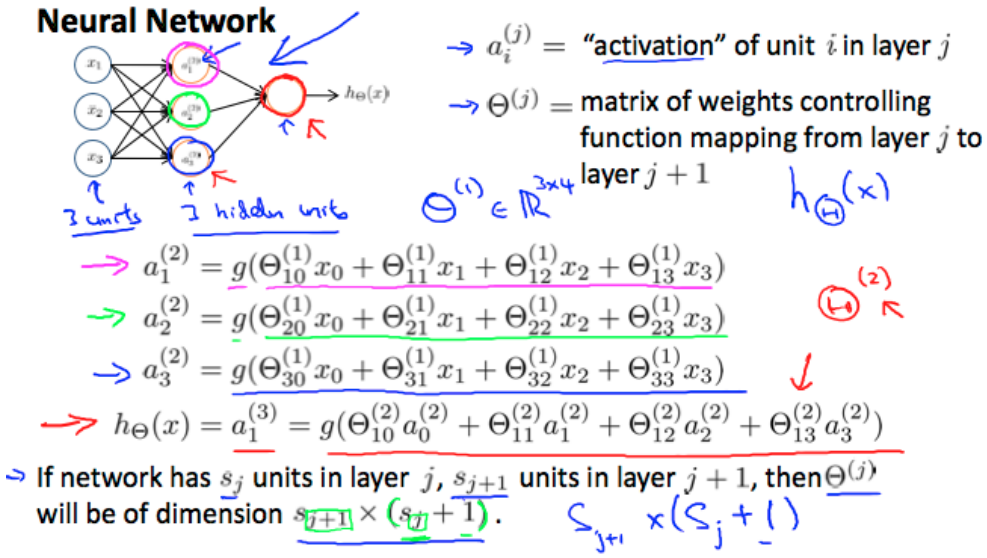


图 7: 神经网络模型

Example:if layer 1 has 2 input nodes and layer 2 has 4 activation nodes.Dimension of $\theta^{(1)}$ is going to be 4*3 where $s_j = 2$ and $s_{j+1} = 4$,so $s_{j+1} * (s_j + 1) = 4 * 3$.

2.2 Model Representation II

前向传播算法（Forward Propagation）：每一个a都是由上一层所有的x和每一个x所对应的决定的。我们把这样从左到右的算法称为前向传播法。

前向传播算法相对于使用循环来编码，利用向量化的方法会使得计算更为简便。以上面的神经网络为例，试着计算第二层的值。

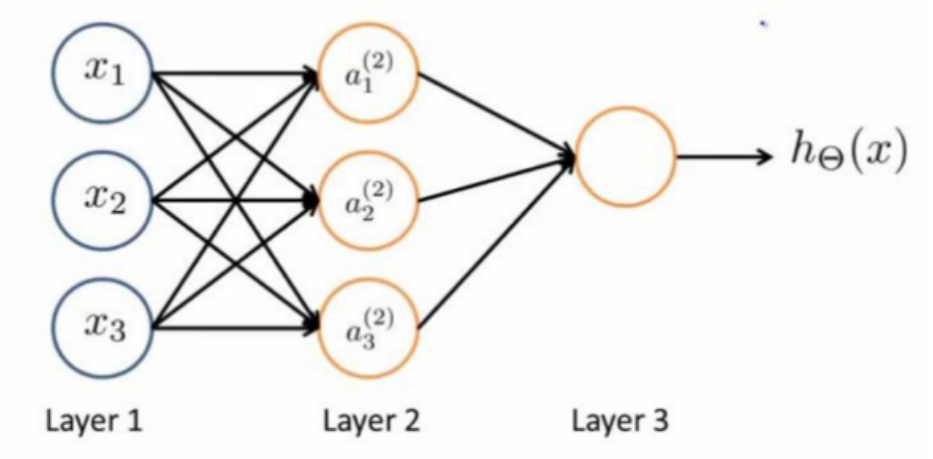


图 8: 神经网络模型

对于上图所示的模型，激活单元和输出分别表达为:

$$a_1^{(2)} = g(\theta_{10}^{(1)}x_0 + \theta_{11}^{(1)}x_1 + \theta_{12}^{(1)}x_2 + \theta_{13}^{(1)}x_3) \tag{5}$$

$$a_2^{(2)} = g(\theta_{20}^{(1)}x_0 + \theta_{21}^{(1)}x_1 + \theta_{22}^{(1)}x_2 + \theta_{23}^{(1)}x_3) \tag{6}$$

$$a_3^{(2)} = g(\theta_{30}^{(1)}x_0 + \theta_{31}^{(1)}x_1 + \theta_{32}^{(1)}x_2 + \theta_{33}^{(1)}x_3) \tag{7}$$

$$h_\theta(x) = g(\theta_{10}^{(2)}a_0^{(2)} + \theta_{11}^{(2)}a_1^{(2)} + \theta_{12}^{(2)}a_2^{(2)} + \theta_{13}^{(2)}a_3^{(2)}) \tag{8}$$

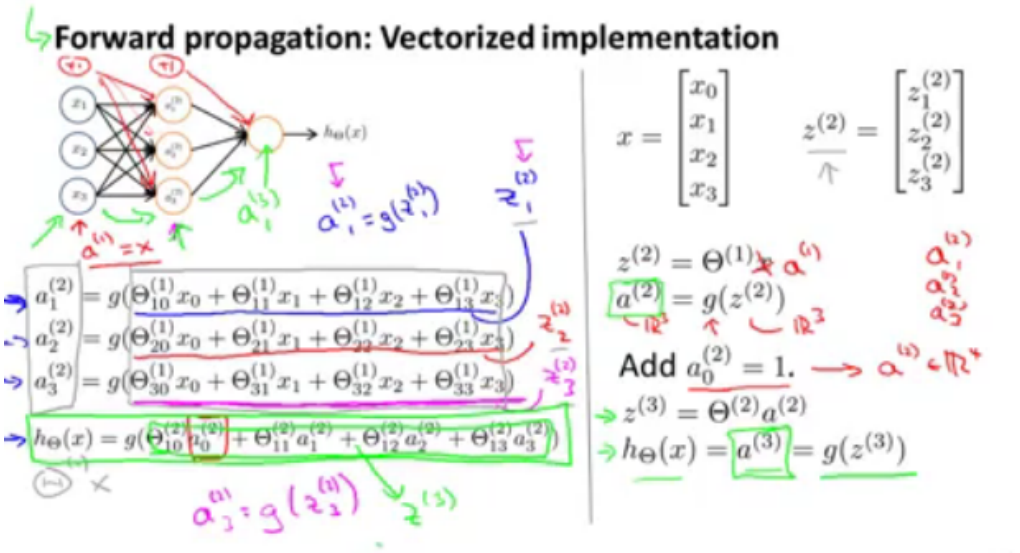


图 9: 神经网络模型

In this section we'll do a vectorized implementation of the above functions.We're going to define a new variable $z_k^{(k)}$ that encompasses the parameters inside our g function.In our previous example if we replaced by

the variable z for all the parameters we should get:

$$a_1^{(2)} = g(z_1^{(2)}) \tag{9}$$

$$a_2^{(2)} = g(z_2^{(2)}) \tag{10}$$

$$a_3^{(2)} = g(z_3^{(2)}) \tag{11}$$

In other words,for layers j=2 and node k,the variable z will be:

$$z_k^{(2)} = \Theta_{k,0}^{(1)} x_0 + \Theta_{k,1}^{(1)} x_1 + ... + \Theta_{k,n}^{(1)} x_n \tag{12}$$

The vector representation of x and z^j is:

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix} \tag{13}$$

$$z^{(j)} = \begin{bmatrix} z_1^{(j)} \\ z_2^{(j)} \\ . \\ . \\ z_n^{(j)} \end{bmatrix} \tag{14}$$

Setting $x = a^{(1)}$,we can rewrite the equation as:

$$z^{(j)} = \Theta^{(j-1)} a^{(j-1)} \tag{15}$$

We are multiplying our matrix $\Theta^{(j-1)}$ with dimensions $s_j \times (n+1)$ (where sj is the number of our activation nodes) by our vector $a^{(j-1)}$ with height (n+1). This gives us our vector $z^{(j)}$ with height s_j. Now we can get a vector of our activation nodes for layer j as follows:

$$a^{(j)} = g(z^{(j)}) \tag{16}$$

Where our function g can be applied element-wise to our vector $z^{(j)}$.

We can then add a bias unit (equal to 1) to layer j after we have computed $a^{(j)}$. This will be element $a_0^{(j)}$ and will be equal to 1. To compute our final hypothesis, let's first compute another z vector:

$$z^{(j+1)} = \Theta^{(j)} a^{(j)} \tag{17}$$

We get this final z vector by multiplying the next theta matrix after $\Theta^{(j-1)}$ with the values of all the activation nodes we just got. This last theta matrix $\Theta^{(j)}$ will have only one row which is multiplied by one column $a^{(j)}$ so that our result is a single number. We then get our final result with:

$$h_\Theta(x) = a^{(j+1)} = g(z^{(j+1)}) \tag{18}$$

Notice that in this last step, between layer j and layer j+1, we are doing exactly the same thing as we did in logistic regression. Adding all these intermediate layers in neural networks allows us to more elegantly produce interesting and more complex non-linear hypotheses.

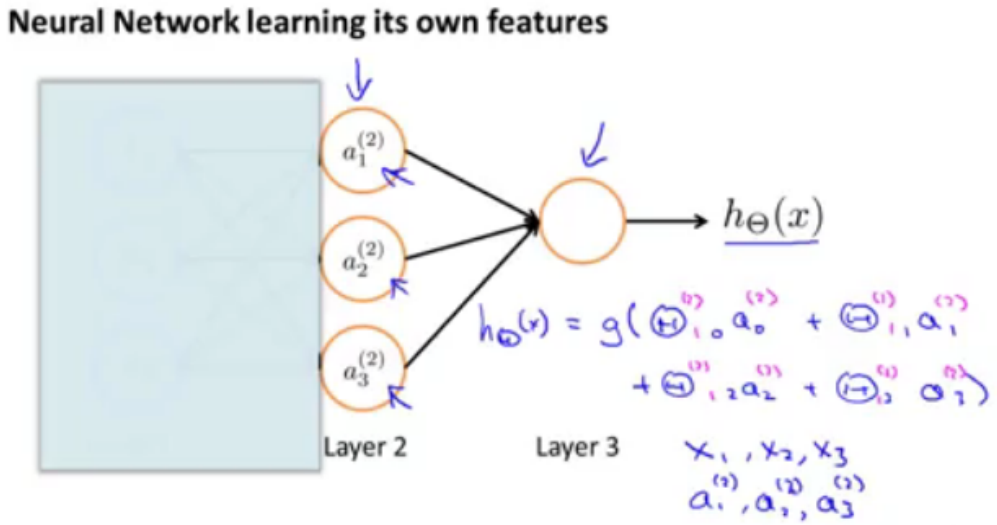


图 10: 神经网络模型

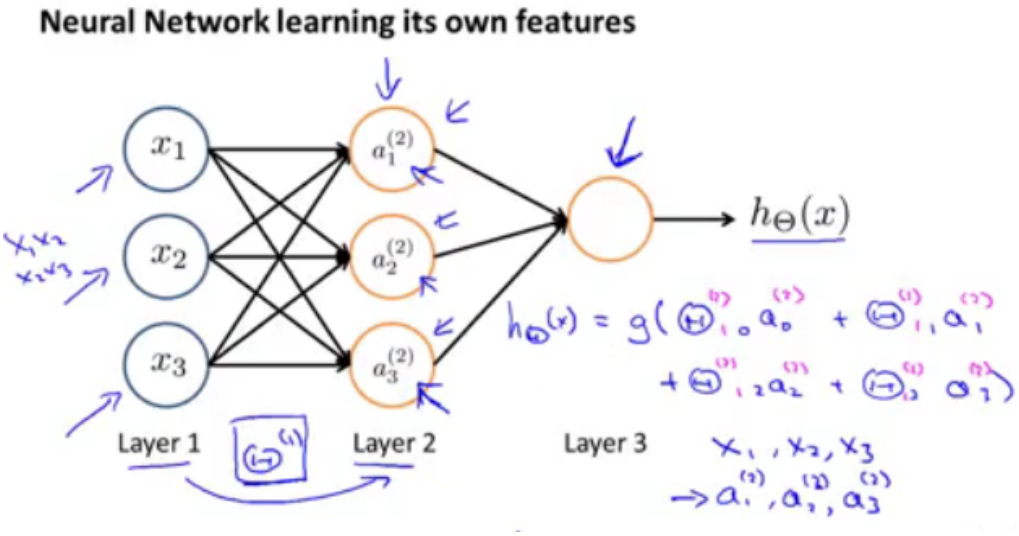


图 11: 神经网络模型

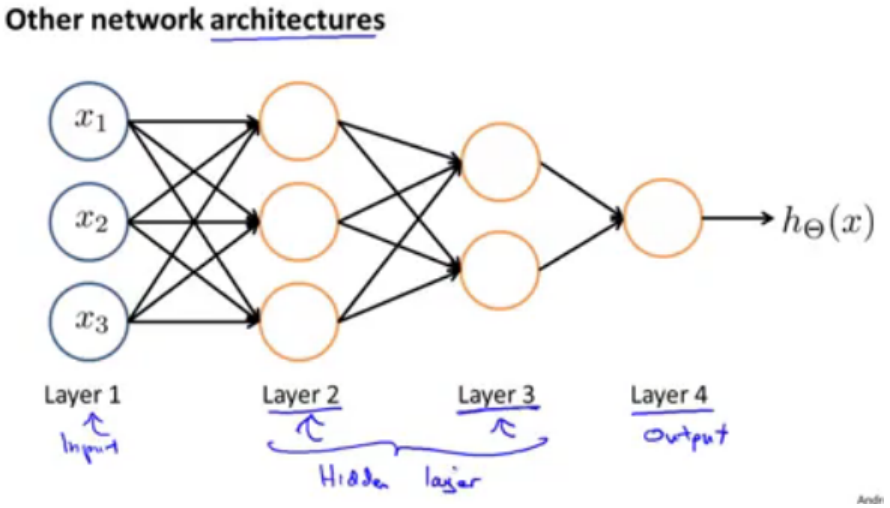


图 12: 神经网络的架构

其实神经网络就像是逻辑回归，只不过输入特征值不再是x_1，x_2，x_3，我们把逻辑回归中的输入向量x_1，x_2，x_3变成了中间层的a_1，a_2，a_3，即：

$$h_\theta(x) = g(\theta_{10}^{(2)}a_0^{(2)} + \theta_{11}^{(2)}a_1^{(2)} + \theta_{12}^{(2)}a_2^{(2)} + \theta_{13}^{(2)}a_3^{(2)}) \tag{19}$$

我们可以把a_0，a_1，a_2，a_3看成更加高级的特征值，也就是x_0，x_1，x_2，x_3的进化体，并且它们是由x决定的，因为是梯度下降的，所以a是变化的，并且变得越来越厉害，所以这些更高级的特征值远比仅仅将x次方厉害，也能更好的预测新数据。

这就是神经网络相比于逻辑回归和线性回归的优势。

3 Applications

3.1 Examples and Intuitions I

从本质上讲，神经网络能够通过学习得出其自身的一系列特征。在普通的逻辑回归中,我们被限制为使用数据中的原始特征 x_1，x_2，x_3，...，x_n，虽然可以使用一些二项式项来组合这些特征,但是仍然受到这些原始特征的限制。在神经网络中,原始特征只是输入层，在上面三层的神经网络例子中,第三层也就是输出层做出的预测利用的是第二层的特征，而非输入层中的原始特征，我们可以认为第二层中的特征是神经网络通过学习后自己得出的一系列用于预测输出变量的新特征。

神经网络中，单层神经元（无中间层）的计算可用来表示逻辑运算，比如逻辑与AND、逻辑或OR。

举例说明：逻辑与AND

A simple example of applying neural networks is by predicting x_1 AND x_2, which is the logical 'and' operator and is only true if both x_1 and x_2 are 1.

The graph of our functions will look like:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \rightarrow \left[g(z^{(2)})\right] \rightarrow h_\Theta(x)$$

Remember that x_0 is our variable and is always 1.

Let's set our first theta matrix as:

$$\Theta^{(1)} = \begin{bmatrix} -30 & 20 & 20 \end{bmatrix} \tag{20}$$

This will cause the output of our hypothesis to only be positive if both x_1 and x_2 are 1. In other words:

$$\begin{aligned} &h_\Theta(x) = g(-30 + 20x_1 + 20x_2) \\ &x_1 = 0 \ \ and \ \ x_2 = 0 \ \ then \ \ g(-30) \approx 0 \\ &x_1 = 0 \ \ and \ \ x_2 = 1 \ \ then \ \ g(-10) \approx 0 \\ &x_1 = 1 \ \ and \ \ x_2 = 0 \ \ then \ \ g(-10) \approx 0 \\ &x_1 = 1 \ \ and \ \ x_2 = 1 \ \ then \ \ g(10) \approx 1 \end{aligned}$$

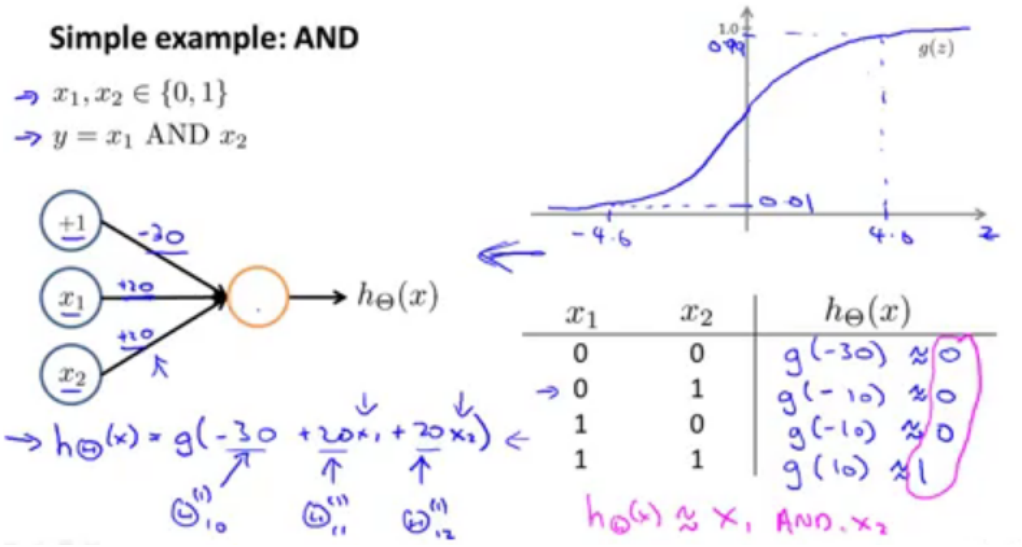


图 13: 逻辑与AND

So we have constructed one of the fundamental operations in computers by using a small neural network rather than using an actual AND gate. Neural networks can also be used to simulate all the other logical gates. The following is an example of the logical operator 'OR', meaning either x_1 is true or x_2 is true, or both:

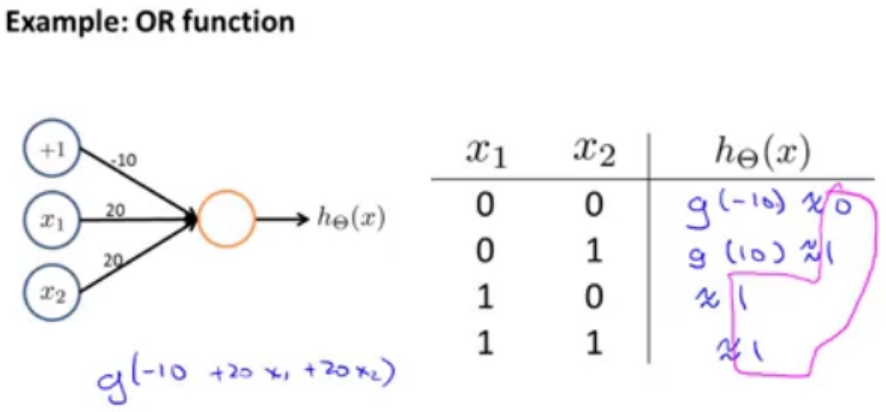
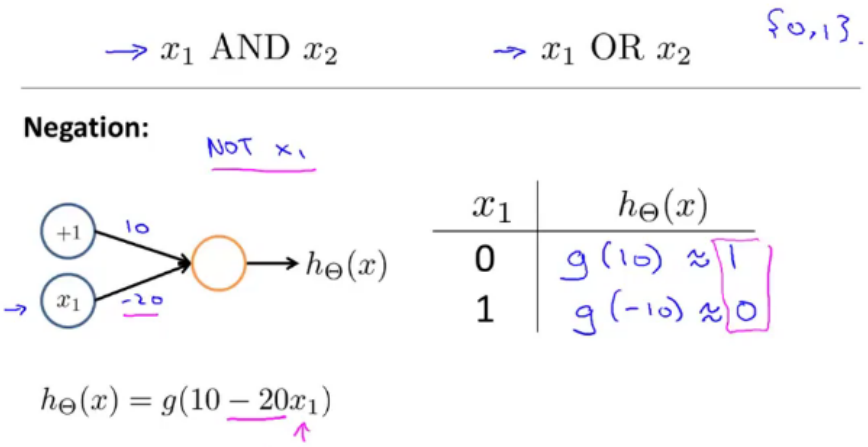


图 14: 逻辑或OR

3.2 Examples and Intuitions II

逻辑非运算（NOT）：



x_1	$h_\Theta(x)$
0	$g(10) \approx 1$
1	$g(-10) \approx 0$

图 15: 逻辑非NOT

二元逻辑运算符（Binary logical operators）当输入特征为布尔值（0或1）时，我们可以用一个单一的激活层作为二元逻辑运算符，为了表示不同的运算符，我们来选择不同的权重即可。

The $\Theta^{(1)}$ matrices for AND,NOR,and OR are:

$$
\begin{aligned}
&AND: \\
&\quad \Theta^{(1)} = \begin{bmatrix} -30 & 20 & 20 \end{bmatrix} \\
&NOR: \\
&\quad \Theta^{(1)} = \begin{bmatrix} 10 & -20 & -20 \end{bmatrix} \\
&OR: \\
&\quad \Theta^{(1)} = \begin{bmatrix} -10 & 20 & 20 \end{bmatrix}
\end{aligned}
$$

We can combine these to get the XNOR logical operator (which gives 1 if x_1 and x_2 are both 0 or both 1).

$$
\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} \rightarrow \begin{bmatrix} a^{(3)} \end{bmatrix} \rightarrow h_\Theta(x)
$$

For the transition between the first and second layer, we'll use a $\Theta^{(1)}$ matrix that combines the values for AND and NOR:

$$
\Theta^{(1)} = \begin{bmatrix} -30 & 20 & 20 \\ 10 & -20 & -20 \end{bmatrix}
$$

For the transition between the second and third layer, we'll use a $\Theta^{(2)}$ matrix that uses the value for OR:

$$
\Theta^{(2)} = \begin{bmatrix} -10 & 20 & 20 \end{bmatrix}
$$

Let's write out the values for all nodes:

$$a^{(2)} = g(\Theta^{(1)} \cdot x) \tag{21}$$

$$a^{(3)} = g(\Theta^{(2)} \cdot a^{(2)}) \tag{22}$$

$$h_\Theta(x) = a^{(3)} \tag{23}$$

And there we have the XNOR operator using a hidden layer with two nodes! The following summarizes the above algorithm:

我们可以利用神经元来组合成更为复杂的神经网络以实现更复杂的运算。例如我们要实现 XNOR(同或门运算（异或非门运算）) 功能(输入的两个值必须一样,均为 1 或均为 0),即 XNOR=(x_1 AND x_2)OR((NOT x_1)AND(NOT x_2))

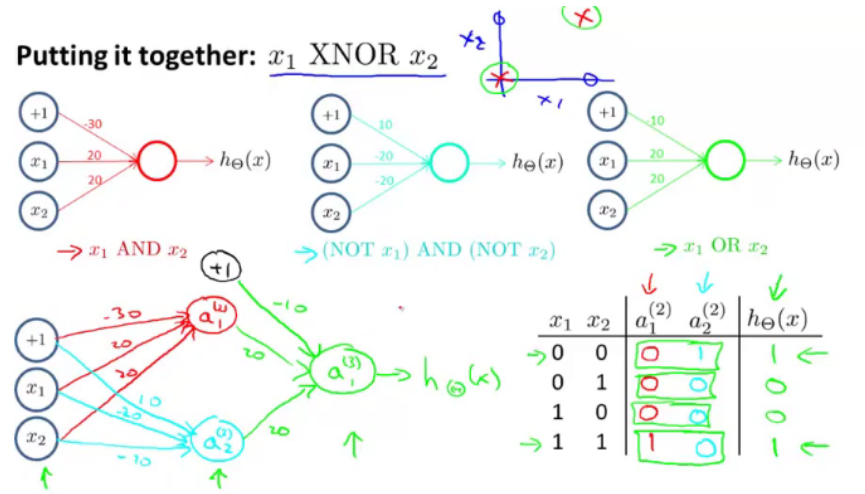


图 16: 同或门（异或非门）运算

3.3 Multiclass Classification

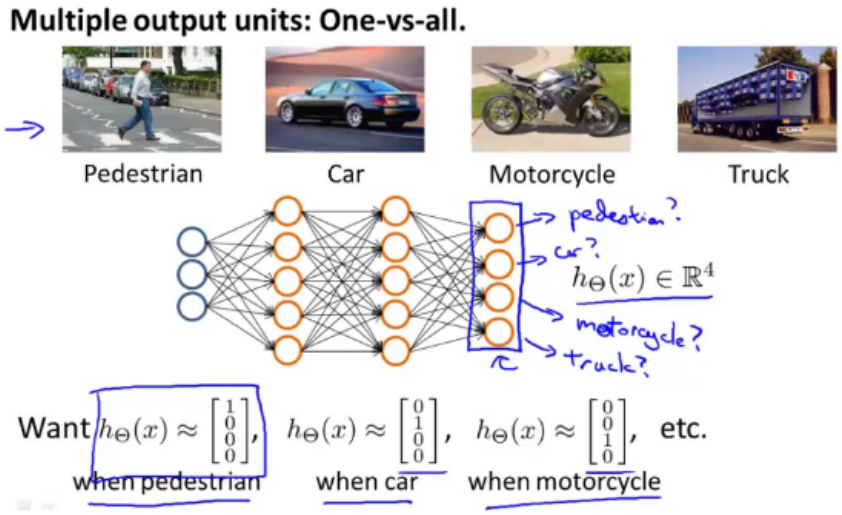


图 17: 多分类问题举例

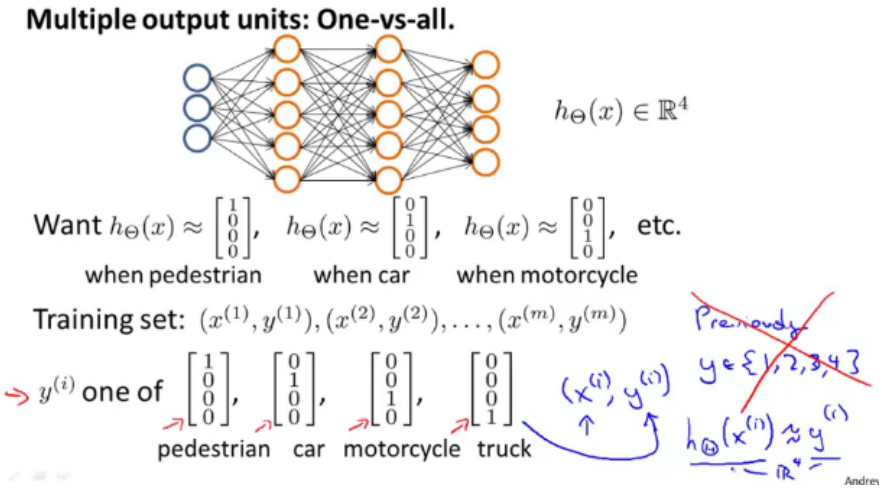


图 18: 多分类：One-vs-all

We can define our set of resulting classes as y:

$$y^{(i)} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\end{bmatrix},$$

Each $y^{(i)}$ represents a different image corresponding to either a car, pedestrian, truck, or motorcycle. The inner layers, each provide us with some new information which leads to our final hypothesis function. The setup looks like:

$$\begin{bmatrix}x_0\\x_1\\x_2\\\cdots\\x_n\end{bmatrix} \rightarrow \begin{bmatrix}a_0^{(2)}\\a_1^{(2)}\\a_2^{(2)}\\\cdots\end{bmatrix} \rightarrow \begin{bmatrix}a_0^{(3)}\\a_1^{(3)}\\a_2^{(3)}\\\cdots\end{bmatrix} \rightarrow \cdots \rightarrow \begin{bmatrix}h_\Theta(x)_1\\h_\Theta(x)_2\\h_\Theta(x)_3\\h_\Theta(x)_4\end{bmatrix}$$

Our resulting hypothesis for one set of inputs may look like:

$$h_\Theta(x) = \begin{bmatrix}0\\0\\1\\0\end{bmatrix} \tag{24}$$

In which case our resulting class is the third one down, or $h_\Theta(x)_3$, which represents the motorcycle.

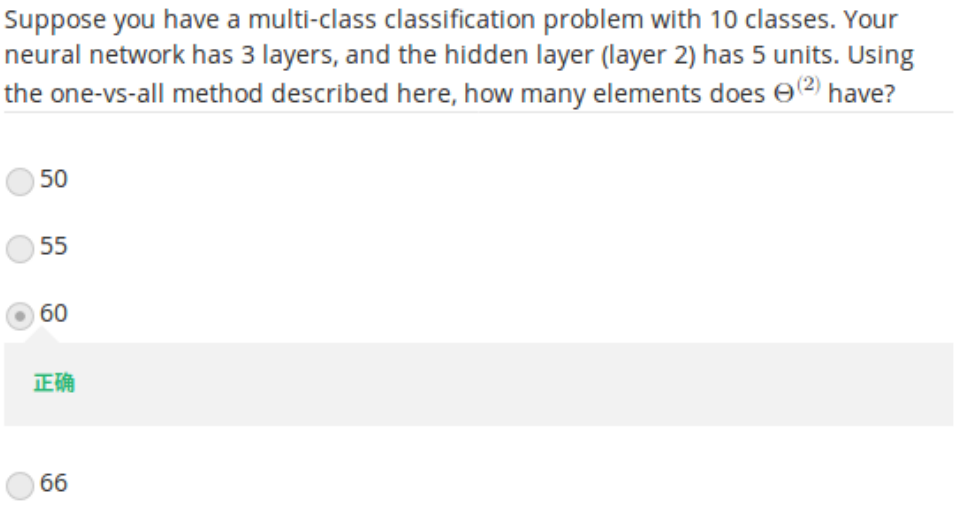


图 19: 多分类习题：One-vs-all