

Machine Learning - Week 3

赵燕

目录

1	Motivations	2
1.1	Non-linear Hypotheses	2
1.2	Neurons and the Brain	3
2	Neural Networks	3
2.1	Model Representation I	3
2.2	Model Representation II	6
3	Applications	6
3.1	Examples and Intuitions I	6
3.2	Examples and Intuitions II	6
3.3	Multiclass Classification	6

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：

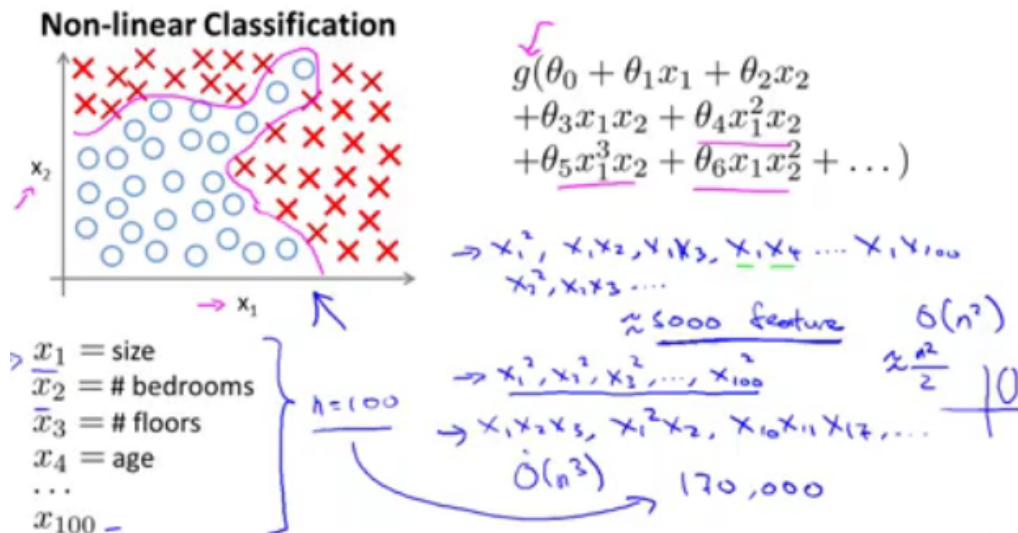


图 1: 举例说明

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($x_1 x_2 + x_1 x_3 + x_1 x_4 + \dots + x_2 x_3 + x_2 x_4 + \dots + x_9 x_{100}$),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），我们怎样才能这么做呢？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：

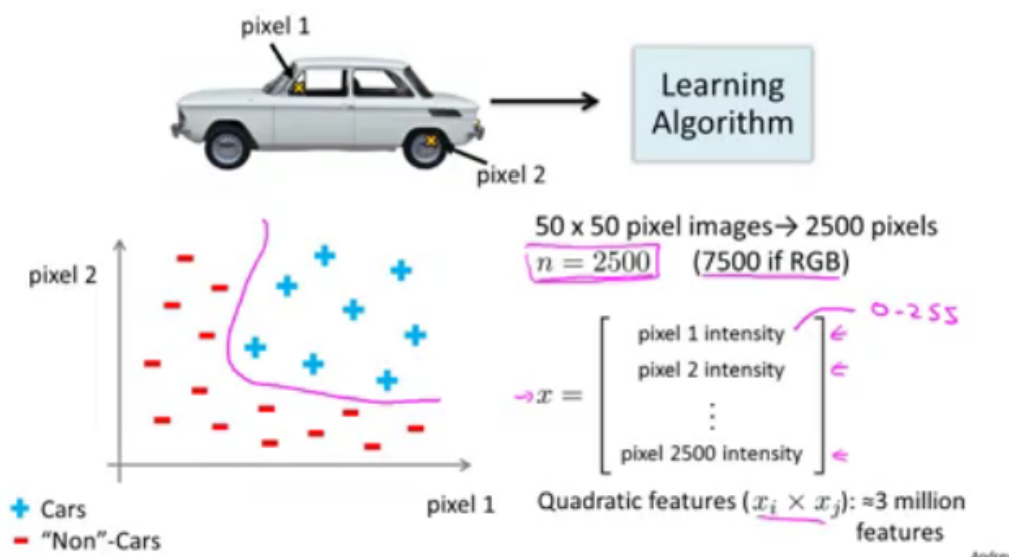


图 2: 计算机视觉-汽车模型

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。

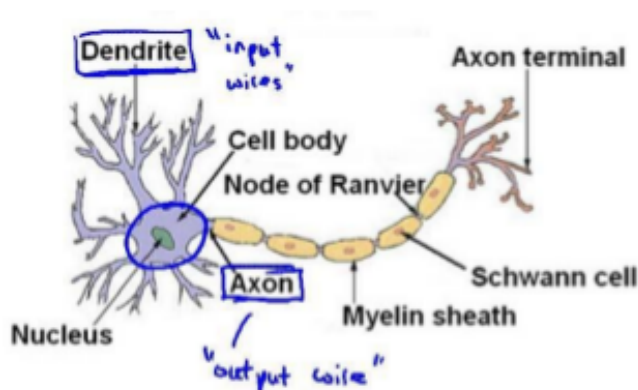


图 3: 大脑中的神经网络

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, activation unit）采纳一些特征作为输入，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:

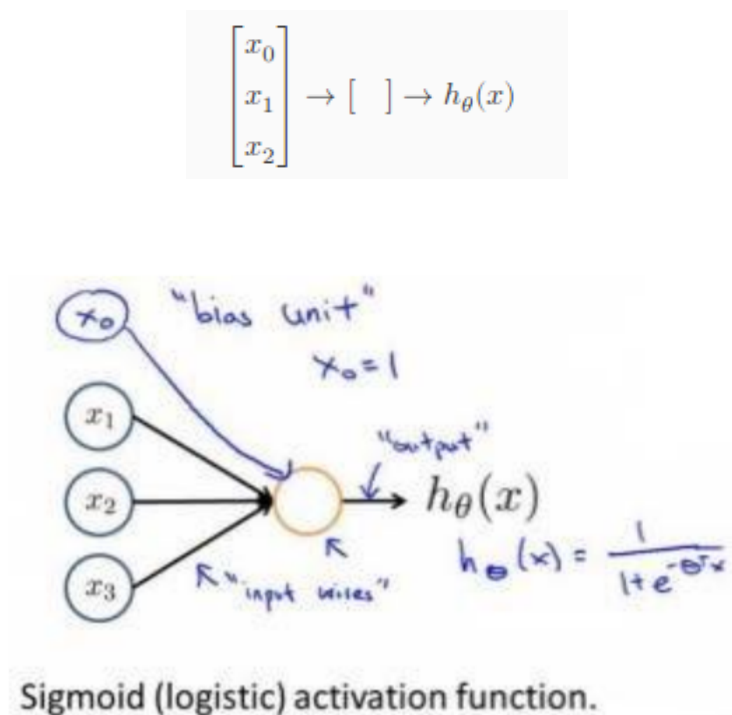


图 4: 神经网络简单模型

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 $a^{(2)}$ 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:

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

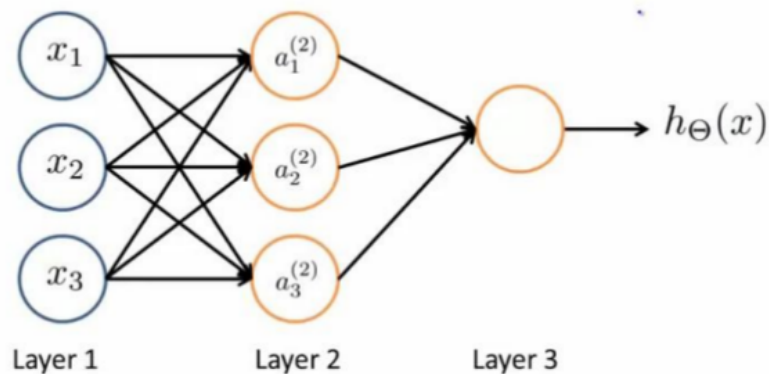


图 5: 神经网络模型

- (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):

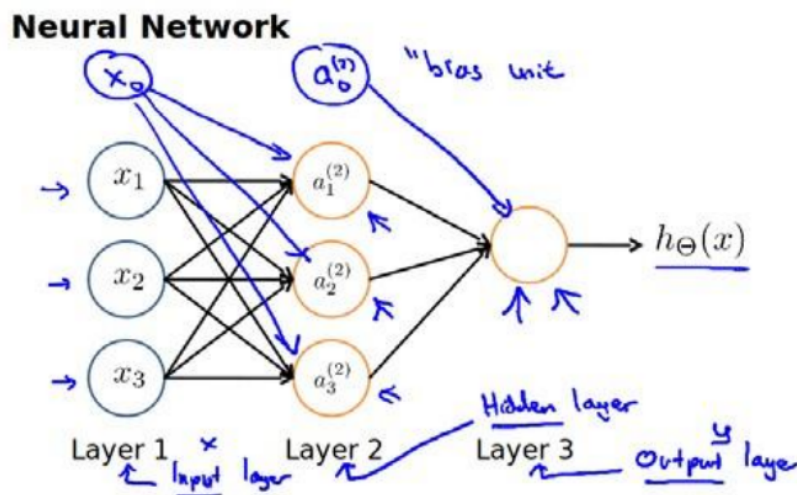


图 6: 神经网络模型

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：

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

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

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

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

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 j 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 $a_0^{(j)}$. In other words the output nodes will not include the bias nodes while the inputs will. The following image summarizes our model representation:

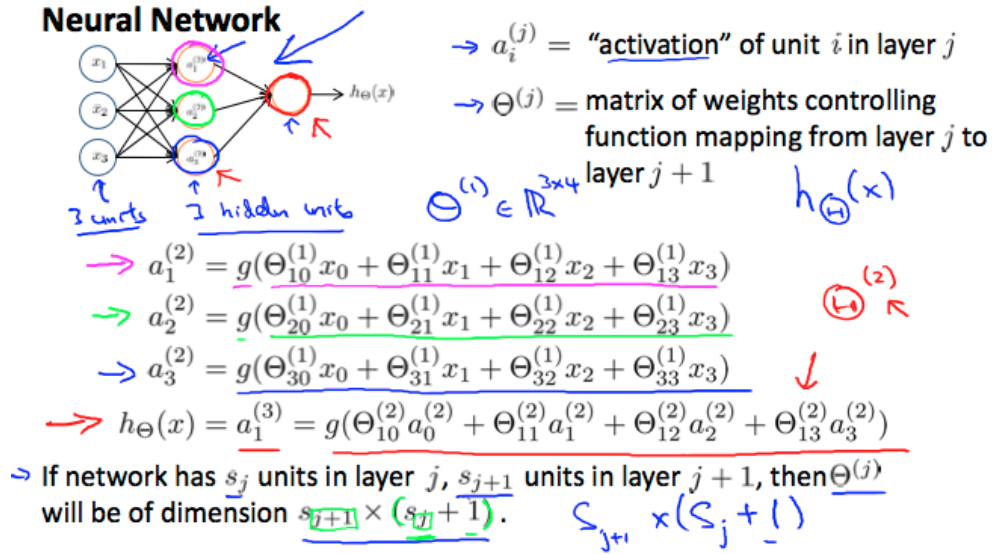


图 7: 神经网络模型

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

3 Applications

3.1 Examples and Intuitions I

3.2 Examples and Intuitions II

3.3 Multiclass Classification