神经网路

韩雅妮

神经网络产生的目的：

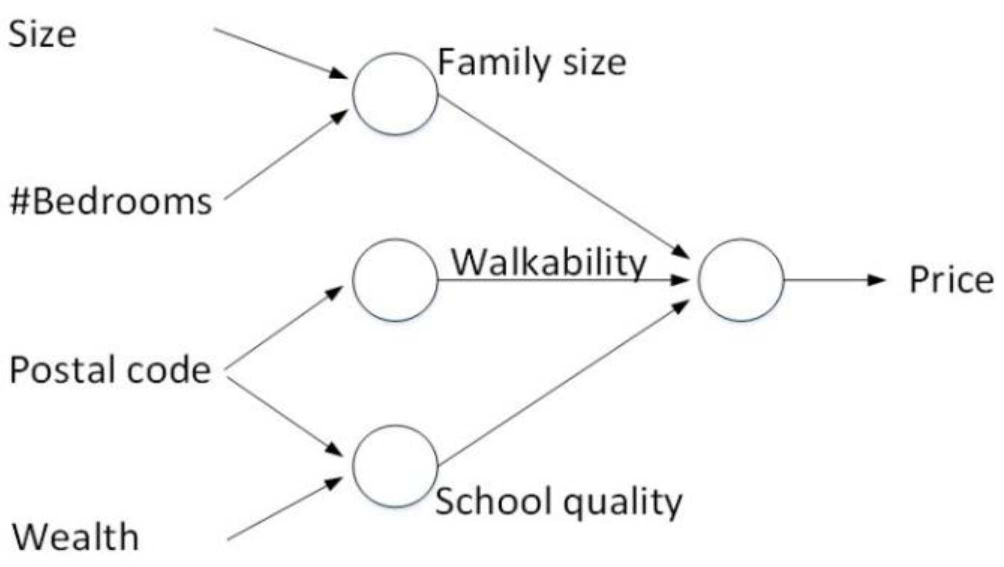
制造能模拟人类大脑的机器，解决不同机器之间的学习问题。

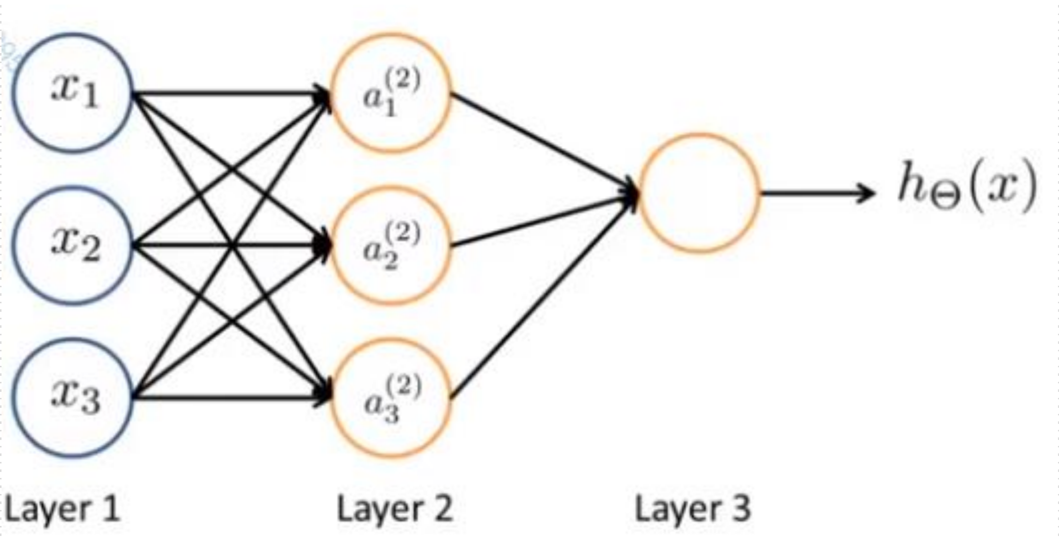
模型描述：

神经元是一个计算单元，从输入通道接收一些信息并计算，然后输出结果。

该神经网络的输入是房屋面积，输出是房屋价格，中间包含了一个神经元，即房价预测函数（蓝色折线）。该神经元的功能就是实现函数f(x)的功能。

Housing Price Prediction





$\boldsymbol{a_i^{(j)}}$：第j层第i个神经元的激活项；

激活项是由一个具体的神经元计算并输出的值。

$\boldsymbol{\theta^{(j)}}$：权重矩阵，控制从一层到另一层的映射。

$$a_1^{(2)} = g(\theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3)$$
$$a_2^{(2)} = g(\theta_{20}^{(1)} x_0 + \theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2 + \theta_{23}^{(1)} x_3)$$
$$a_3^{(2)} = g(\theta_{30}^{(1)} x_0 + \theta_{31}^{(1)} x_1 + \theta_{32}^{(1)} x_2 + \theta_{33}^{(1)} x_3)$$

$$h_\theta(x) = a_1^{(3)} = g(\theta_{10}^{(2)} a_0^{(2)} + \theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} a_2^{(2)} + \theta_{13}^{(2)} a_3^{(2)})$$

前向传播向量化实现

令$z_1^{(2)}=\boldsymbol{\theta}_{10}^{(1)}\boldsymbol{x}_0+\boldsymbol{\theta}_{11}^{(1)}\boldsymbol{x}_1+\boldsymbol{\theta}_{12}^{(1)}\boldsymbol{x}_2+\boldsymbol{\theta}_{13}^{(1)}\boldsymbol{x}_3$

则$\boldsymbol{a}_1^{(2)}=g(z_1^{(2)})$，类似，$\boldsymbol{a}_2^{(2)}=g(z_2^{(2)})$，$\boldsymbol{a}_3^{(2)}=g(z_3^{(2)})$

此处，z值是某个特定神经元的输入值$\boldsymbol{x}_0,\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_3$的加权线性组合。

将特征向量定义为$\mathrm{x}=\begin{bmatrix}\boldsymbol{x}_0\\ \boldsymbol{x}_1\\ \boldsymbol{x}_2\\ \boldsymbol{x}_3\end{bmatrix}$, $\boldsymbol{z}^{(2)}=\begin{bmatrix}z_1^{(2)}\\ z_2^{(2)}\\ z_3^{(2)}\end{bmatrix}$

$$\boldsymbol{a}^{(2)}=g(\boldsymbol{z}^{(2)})$$

其中，a, z都是三维向量，g为sigmoid函数，逐个元素的作用于$\boldsymbol{z}^{(2)}$中的各个元素。

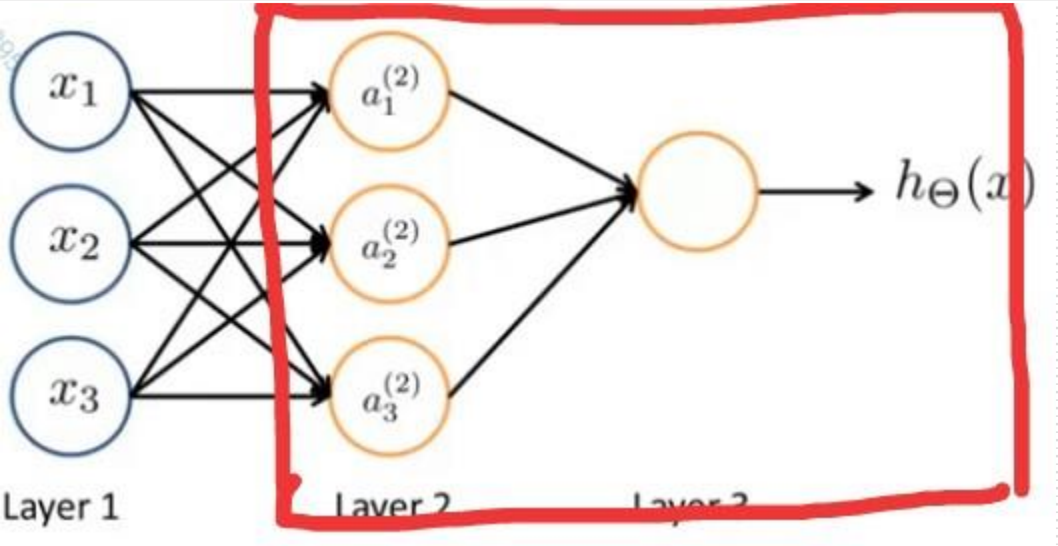
添加偏置单元，$\boldsymbol{a}_0^{(2)}=1$，$\boldsymbol{a}^{(2)}$变为4维向量。

假设函数的实际输出值：$\boldsymbol{h}_{\boldsymbol{\theta}}(\mathrm{x})=\boldsymbol{a}^{(3)}=g(\boldsymbol{z}^{(3)})$

前向传播：

从输入单元的激活项开始，然后进行前向传播给隐藏层，计算隐藏层中的激活项，然后继续向前传播，并计算出输出层的激活项。

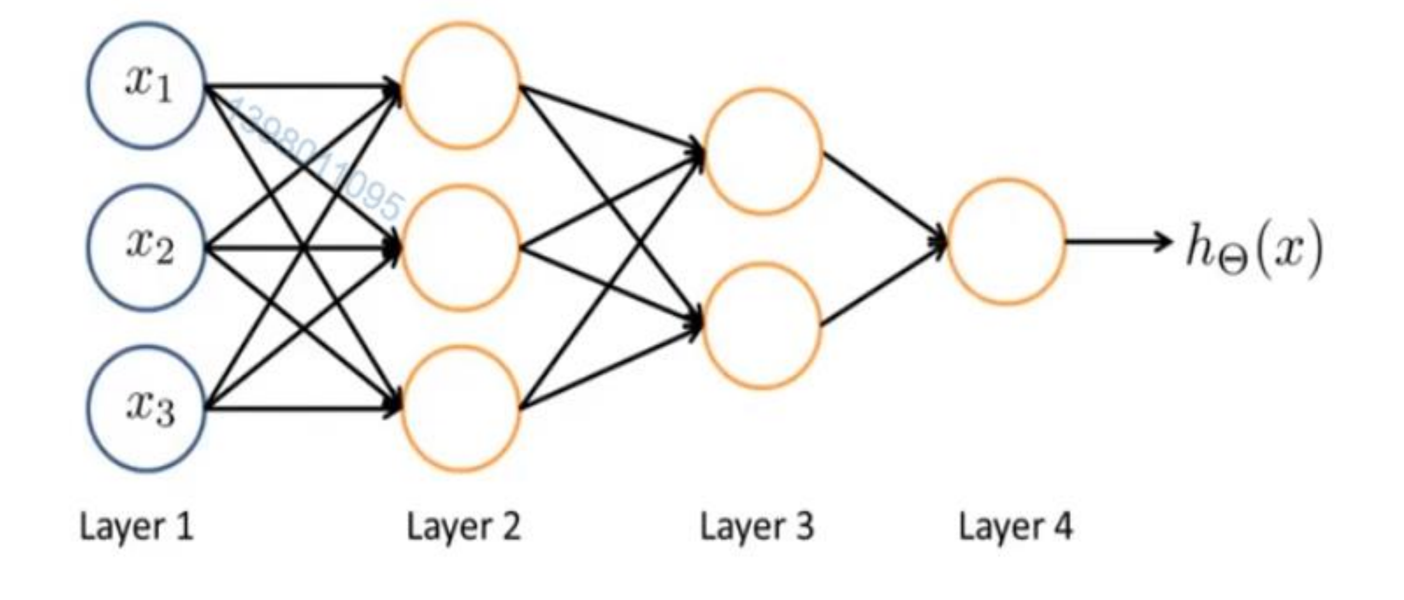
从输入层到隐藏层，再到输出层，依次计算激活项的过程成为前向传播。



类似logistic回归，第三层的神经元作为逻辑回归单元，来预测h_θ(x)。不过，在神经网络中， a_1,a_2,a_3是学习得到的函数输入值。

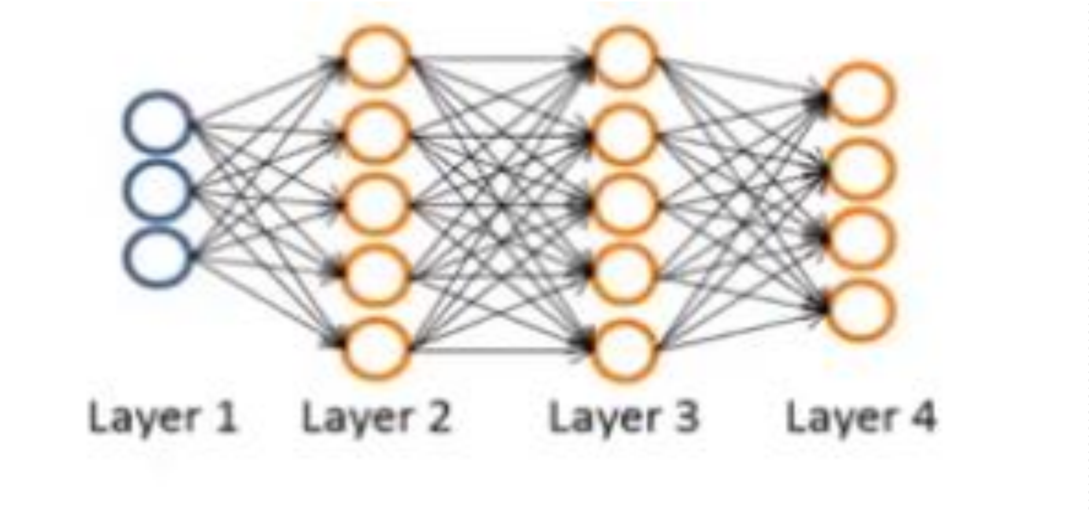
在神经网络中，没有使用$\boldsymbol{x_1, x_2, x_3}$作为输入特征来训练逻辑回归，而是自己训练逻辑回归的输入$\boldsymbol{a_1, a_2, a_3}$。根据为$\boldsymbol{\theta}$选择的不同参数，可以学习到一些复杂的特征，就可得到一个很好的假设函数。

神经网络的架构



架构也就是不同神经元之间的连接方式，其中第1层为输入层，2，3层为隐藏层，第4层为输出层。第4层以第3层训练出的更复杂的特征作为输入，因此能够得到更复杂的非线性假设函数。

假设神经网络



含有m个样本$\{(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), \cdots, (\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)})\}$

L:神经网络的总层数;

s_l:第l层的单元数，也就是神经元的数量，其中不包含偏置单元。

二元分类：输出层的单元数目：k=1；多类别分类：有k个不同的类别，k个输出单元

例：在计算机视觉中，区分行人，汽车，摩托车，卡车。

有四个分类器，每一个识别一类物体。

四个输出$\boldsymbol{h_\theta}(\mathrm{x}) \approx \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$， $\boldsymbol{h_\theta}(\mathrm{x}) \approx \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$，etc.

训练集：$\{(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), \cdots, (\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)})\}$

$\boldsymbol{y}^{(\mathrm{i})}$是$\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$， $\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$， $\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$， $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$中的一个

神经网络的代价函数

在逻辑回归中，代价函数：

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

与逻辑回归不同的是，神经网络中的输出单元不是一个：

$$\mathrm{J}(\boldsymbol{\theta}) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{\mathrm{k}=1}^{K} y_k^{(i)}\log(h_\theta(x^{(i)}))_k+(1- y_k^{(i)})\log(1-(h_\theta(x^{(i)}))_k)\right]+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{\mathrm{i}=1}^{s_l}\sum_{j=1}^{s_l+1}\theta_{ji}^{(l)^2}$$

$h_\theta(\mathrm{x})$是一个k维向量；

$(h_\theta(x^{(i)}))_{\mathrm{i}}$ 表示选择神经网络中的第i个输出。

$\sum_{i=1}^{m}\sum_{\mathrm{k}=1}^{K} y_k^{(i)}\log(h_\theta(x^{(i)}))_k+(1- y_k^{(i)})\ \log(1-(h_\theta(x^{(i)}))_k)$：表示k个输出单元之和。

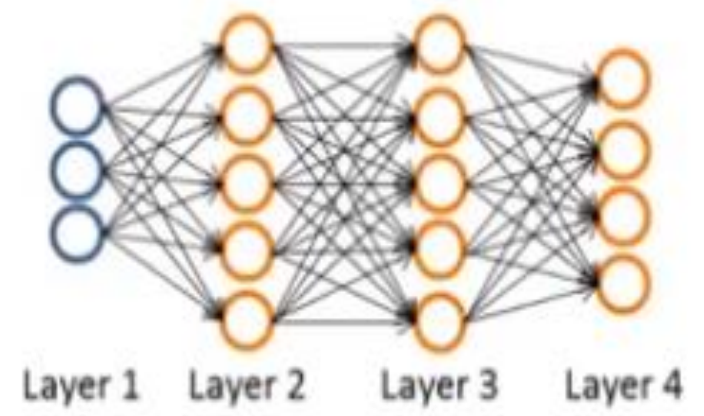
最小化代价函数

$a^{(1)}$=x;

$z^{(2)} = \theta^{(1)}\ a^{(1)};\ a^{(2)}$=g$(z^{(2)})$;

$z^{(3)} = \theta^{(2)}\ a^{(2)};\ a^{(3)}$=g$(z^{(3)})$;

$z^{(4)} = \theta^{(3)}\ a^{(3)};\ a^{(4)}= h_\theta(x)$=g$(z^{(4)})$



$\boldsymbol{\delta}_j^{(l)}$:第1层第j个节点的误差

$\boldsymbol{\delta}_j^{(4)} = a_j^{(4)} - y_j = (\boldsymbol{h_\theta(x)})_{\boldsymbol{j}} - y_j$

该单元的激活值减去训练样本中的真实值

向量化：$\boldsymbol{\delta}^{(4)} = \boldsymbol{a}^{(4)} - y$　　其中，$\boldsymbol{\delta}$, a, y都为向量。

计算网络中前面几层的误差项

$\boldsymbol{\delta}^{(3)} = (\boldsymbol{\theta}^{(3)})^{\boldsymbol{T}}\ \boldsymbol{\delta}^{(4)}\ .*\ \boldsymbol{g'(z^{(3)})} = (\boldsymbol{\theta}^{(3)})^{\boldsymbol{T}}\ \boldsymbol{\delta}^{(4)}\ .*\ \boldsymbol{a}^{(3)}\ .*\ (\boldsymbol{1} - \boldsymbol{a}^{(3)})$

$\boldsymbol{\delta}^{(2)} = (\boldsymbol{\theta}^{(2)})^{\boldsymbol{T}}\ \boldsymbol{\delta}^{(3)}\ .*\ \boldsymbol{g'(z^{(2)})} = (\boldsymbol{\theta}^{(2)})^{\boldsymbol{T}}\ \boldsymbol{\delta}^{(3)}\ .*\ \boldsymbol{a}^{(2)}\ .*\ (\boldsymbol{1} - \boldsymbol{a}^{(2)})$

没有$\boldsymbol{\delta}^{(1)}$，因为第一层是在训练集中实际观察到的，不存在误差。

从输出层开始计算$\boldsymbol{\delta}$项，然后返回上一层，类似于把输出层的误差，反向传播。

采用反向传播算法计算导数项:

$$\frac{\partial}{\partial \theta_{\mathrm{ij}}} \mathrm{J}(\boldsymbol{\theta}) = \boldsymbol{a}_j^{(l)} \boldsymbol{\delta}_i^{(l+1)}$$ （忽略正则化项，$\boldsymbol{\lambda}=0$）

Thanks