Machine Learning Formulas

MingShun Wu

November 6, 2016

Contents

1 线性回归(Linear Regression)

1.1 各向量形式

在机器学习中，各个变量在单独出现时均为列向量的形式，但若是以多个向量形成的矩阵形式出现时，均为行向量的形式。
以下将会写出各个变量单独出现的情况、各变量以矩阵的形式一起出现的情况。

1. $\vec{x}$

$$\vec{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}_{(n+1)*1} \tag{1}$$

2. $\vec{\theta}$

$$\vec{\theta} = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{pmatrix}_{(n+1)*1} \tag{2}$$

3. $\vec{y}$

$$\vec{y} = \begin{pmatrix} y \end{pmatrix}_{1*1} \tag{3}$$

1.2 批梯度下降

1.2.1 各矩阵形式

下述式子中均为矩阵的形式

1. X

$$\begin{aligned} X &= \begin{pmatrix} \vec{x}^{(1)} \\ \vec{x}^{(2)} \\ \vec{x}^{(3)} \\ \vdots \\ \vec{x}^{(m)} \end{pmatrix} \\ &= \begin{pmatrix} x_0^{(1)} & x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_n^{(2)} \\ x_0^{(3)} & x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & \dots & x_n^{(3)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_0^{(m)} & x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \dots & x_n^{(m)} \end{pmatrix} \\ &= \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_n^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & \dots & x_n^{(3)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \dots & x_n^{(m)} \end{pmatrix}_{m*(n+1)} \end{aligned} \tag{4}$$

2. $\vec{\theta}$

$$\vec{\theta} = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \vdots \\ \theta_n \end{pmatrix}_{(n+1)*1} \tag{5}$$

3. $h_{\vec{\theta}}(\vec{x})$

$$\begin{aligned} h_{\vec{\theta}}(\vec{x}) &= X * \vec{\theta} \\ &= \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_n^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & \dots & x_n^{(3)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \dots & x_n^{(m)} \end{pmatrix} \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \vdots \\ \theta_n \end{pmatrix} \\ &= \begin{pmatrix} 1\theta_0 + x_1^{(1)}\theta_1 + x_2^{(1)}\theta_2 + x_3^{(1)}\theta_3 + \dots + x_n^{(1)}\theta_n \\ 1\theta_0 + x_1^{(2)}\theta_1 + x_2^{(2)}\theta_2 + x_3^{(2)}\theta_3 + \dots + x_n^{(2)}\theta_n \\ 1\theta_0 + x_1^{(3)}\theta_1 + x_2^{(3)}\theta_2 + x_3^{(3)}\theta_3 + \dots + x_n^{(3)}\theta_n \\ \vdots \\ 1\theta_0 + x_1^{(m)}\theta_1 + x_2^{(m)}\theta_2 + x_3^{(m)}\theta_3 + \dots + x_n^{(m)}\theta_n \end{pmatrix}_{m*1} \end{aligned} \tag{6}$$

4. $\vec{y}$

$$\vec{y} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \\ y^{(m)} \end{pmatrix}_{m*1} \tag{7}$$

1.2.2 Cost Function

1. 数值计算形式：

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left[h_\theta(x^{(i)}) - y^{(i)} \right]^2 \tag{8}$$

2. 矩阵计算形式：

$$J(\theta) = \frac{1}{2m} \left[h_{\vec{\theta}}(\vec{x}) - \vec{y} \right]^T \left[h_{\vec{\theta}}(\vec{x}) - \vec{y} \right] \tag{9}$$

1.2.3 偏导数$\frac{\partial J(\theta)}{\partial \theta_j}$计算

1. 数值计算形式

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \left[h_\theta(x^{(i)}) - y^{(i)} \right] x_j^{(i)} \tag{10}$$

2. 矩阵计算形式

$$\nabla J(\theta) = \frac{1}{m} X^T \left[h_{\vec{\theta}}(\vec{x}) - \vec{y} \right] \tag{11}$$

1.2.4 梯度下降迭代方式

1. 数值计算形式

$$
\begin{aligned}
\theta_j &:= \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j} \\
&:= \theta_j - \alpha \frac{1}{m} \left[h_\theta(x^{(i)}) - y^{(i)} \right] x_j^{(i)}
\end{aligned} \tag{12}
$$

2. 矩阵计算形式

$$
\begin{aligned}
\theta &:= \theta - \alpha \nabla J(\theta) \\
&:= \theta - \alpha \frac{1}{m} X^T \left[h_{\vec{\theta}}(\vec{x}) - \vec{y} \right]
\end{aligned} \tag{13}
$$

1.3 Feature Normalization

$$
x_i = \frac{x_i - \mu}{\sigma} \tag{14}
$$

或

$$
x_i = \frac{x_i - \mu}{max - min} \tag{15}
$$

1.4 公式法求解（Normal Equation）

$$
\theta = (X^T X)^{-1} X^T y \tag{16}
$$

2 逻辑回归(Logistic Regression)

2.1 当只有2个类别时，使用1个分类器

2.1.1 sigmoid函数

$$sigmoid(z) = \frac{1}{1+e^z} \tag{17}$$

2.1.2 预测函数

1. 数值形式

$$h_\theta(x) = \frac{1}{1+e^{\theta^T x}} \tag{18}$$

2. 矩阵形式

$$h_\theta(X) = \frac{1}{1+e^{X\theta}} \tag{19}$$

2.1.3 Cost Function

1. 数值形式

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)} log h_\theta(x^{(i)}) - (1-y^{(i)}) log(1-h_\theta(x^{(i)}))\right] \tag{20}$$

2. 矩阵形式

$$J(\theta) = \frac{1}{m}\left[-y^T \log h_\theta(x) - (1-y^T)\log\left(1-h_\theta(x)\right]\right. \tag{21}$$

2.1.4 偏导数$\frac{\partial J(\theta)}{\partial \theta_j}$

1. 数值计算形式

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left[h_\theta(x^{(i)}) - y^{(i)}\right] x_j^{(i)} \tag{22}$$

2. 矩阵计算形式

$$\nabla J(\theta) = \frac{1}{m}X^T\left[h_\theta(x) - y\right] \tag{23}$$

2.1.5 梯度下降迭代算法

1. 数值计算形式

$$\begin{aligned} \theta_j &:= \theta_j - \alpha\frac{\partial J(\theta)}{\partial \theta_j} \\ &:= \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left[h_\theta(x^{(i)}) - y^{(i)}\right] x_j^{(i)} \end{aligned} \tag{24}$$

2. 矩阵计算形式

$$\begin{aligned} \theta &:= \theta - \alpha\nabla J(\theta) \\ &:= \theta - \alpha\frac{1}{m}X^T\left[h_\theta(x) - y\right] \end{aligned} \tag{25}$$

2.2 当只有k个类别时，使用k个分类器

3 Regularization

3.1 线性回归

3.1.1 数值计算方式

$$J(\theta)=\frac{1}{2m}\sum_{i=1}^{m}[h_\theta(x^{(i)})-y^{(i)}]^2+\lambda\frac{1}{2m}\sum_{j=1}^{n}\theta_j^2 \tag{26}$$

$$\begin{aligned}\frac{\partial J(\theta)}{\partial\theta_j}&=\frac{1}{2m}\sum_{i=1}^{m}2[h_\theta(x^{(i)}-y^{(i)})]\frac{\partial h_\theta(x^{(i)})}{\partial\theta_j}+\lambda\frac{1}{2m}2\sum_{i=1}^{n}\theta_j\\&=\frac{1}{m}\sum_{i=1}^{m}[h_\theta(x^{(i)})-y^{(i)}]x^{(i)}+\frac{\lambda}{m}\sum_{i=1}^{n}\theta_j\end{aligned} \tag{27}$$

$$\begin{cases}\theta_0:=\theta_0-\alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})x_0^{(i)}, & j=0\\ \theta_j:=\theta_j-\alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})x_j^{(i)}+\alpha\frac{\lambda}{m}\theta_j, & j\neq 0\end{cases}$$

3.1.2 矩阵计算方式

$$J(\theta)=\frac{1}{2m}\left[h_\theta(x)-y\right]^T\left[h_\theta(x)-y\right]+\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 \tag{28}$$

$$\nabla J(\theta)=\frac{1}{2m}X^T\left[h_\theta(x)-y\right]+\frac{\lambda}{m}\sum_{i=1}^{n}\theta_j \tag{29}$$

$$matlab\begin{cases}grad & =1/m*X'*(h-y);,\\ grad(2:end) & =grad(2:end)+lambda/m*theta(2:end);\end{cases}$$

3.2 逻辑回归

3.2.1 数值计算方式

$$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}logh_\theta(x^{(i)})-(1-y^{(i)})log(1-h_\theta(x^{(i)}))\right]+\lambda\frac{1}{2m}\sum_{j=1}^{n}\theta_j^2 \tag{30}$$

$$\begin{cases}\theta_0:=\theta_0-\alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})x_0^{(i)}, & j=0\\ \theta_j:=\theta_j-\alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})x_j^{(i)}+\alpha\frac{\lambda}{m}\theta_j, & j\neq 0\end{cases}$$

3.2.2 矩阵计算方式

$$J(\theta)=\frac{1}{m}\left[-y^T\log h_\theta(x)-(1-y^T)\log\left(1-h_\theta(x)\right]+\lambda\frac{1}{2m}\theta^T\theta\right. \tag{31}$$

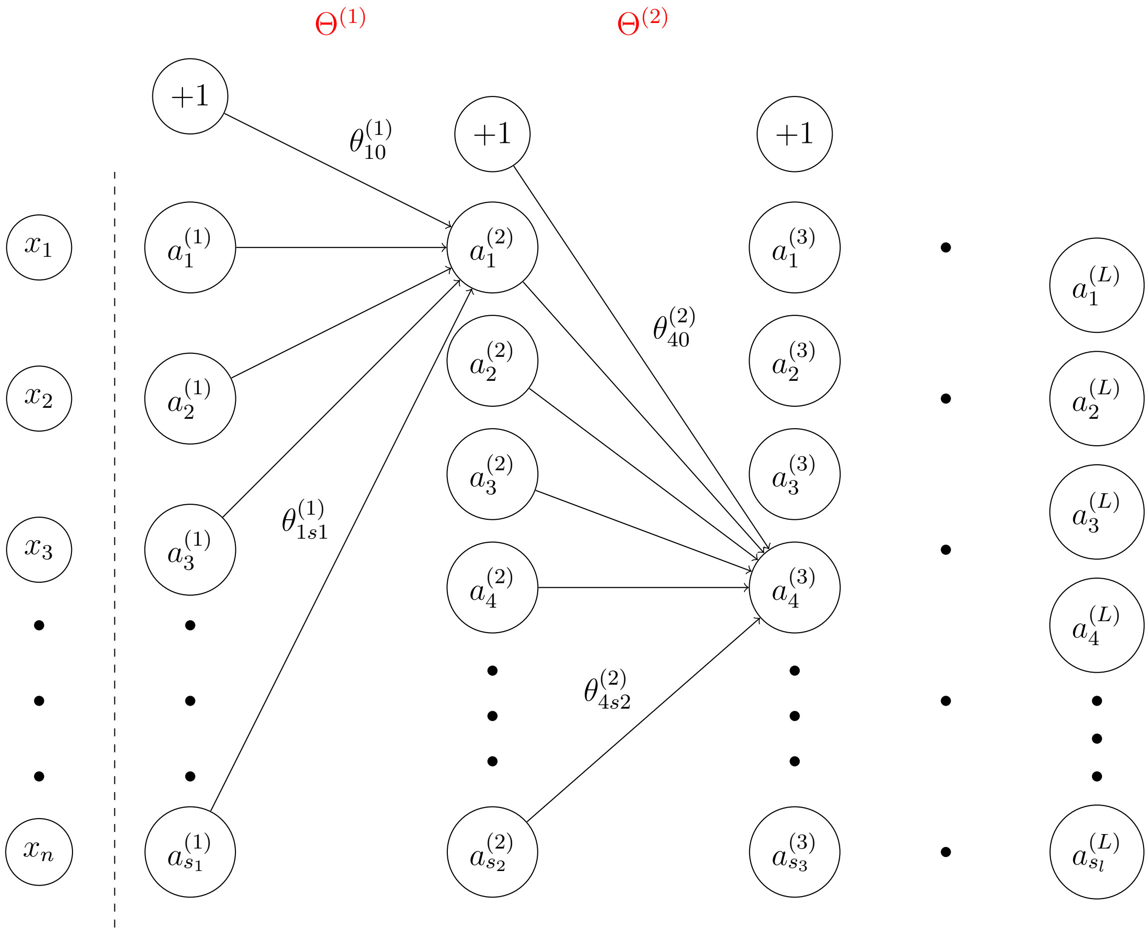
$$matlab\begin{cases}grad & =1/m*X'*(h-y);,\\ grad(2:end) & =grad(2:end)+lambda/m*theta(2:end);\end{cases}$$

3.3 注意

在实际计算θ中，都是先计算没有Regularization的结果，再对(2:end)计算有Regularization的结果，再将其加到没有Regularization的结果中

4 神经网络 - 前向算法

4.1 神经网络示意图 - 前向算法



$$a^{(j)} = g(z^{(j-1)}) \Rightarrow (m, s_j) \tag{32}$$

$$\Theta^{(j)} = \begin{pmatrix} \theta_{10}^{(j)} & \theta_{11}^{(j)} & \theta_{12}^{(j)} & \dots & \theta_{1,s_j}^{(j)} \\ \theta_{20}^{(j)} & \theta_{21}^{(j)} & \theta_{22}^{(j)} & \dots & \theta_{2,s_j}^{(j)} \\ \theta_{30}^{(j)} & \theta_{31}^{(j)} & \theta_{32}^{(j)} & \dots & \theta_{3,s_j}^{(j)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \theta_{s_{j+1}0}^{(j)} & \theta_{s_{j+1}1}^{(j)} & \theta_{s_{j+1}2}^{(j)} & \dots & \theta_{s_{j+1},s_j}^{(j)} \end{pmatrix} \Rightarrow (s_{j+1}, s_j + 1) \tag{33}$$

$$\begin{aligned} z^{(j+1)} &= (1, a^{(j)})(\Theta^{(j)})^T \\ &= \begin{pmatrix} z_1^{(j+1)(1)} & z_2^{(j+1)(1)} & z_3^{(j+1)(1)} & \dots & z_{s_{j+1}}^{(j+1)(1)} \\ z_1^{(j+1)(2)} & z_2^{(j+1)(2)} & z_3^{(j+1)(2)} & \dots & z_{s_{j+1}}^{(j+1)(2)} \\ z_1^{(j+1)(3)} & z_2^{(j+1)(3)} & z_3^{(j+1)(3)} & \dots & z_{s_{j+1}}^{(j+1)(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ z_1^{(j+1)(m)} & z_2^{(j+1)(m)} & z_3^{(j+1)(m)} & \dots & z_{s_{j+1}}^{(j+1)(m)} \end{pmatrix} \end{aligned} \tag{34}$$

$$a^{(j+1)} = g(z^{(j+1)}) \Rightarrow (m, s_{j+1}) \tag{35}$$

$$y = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \\ y^{(m)} \end{pmatrix}_{m*1} \tag{36}$$

4.2 神经网络 - 前向算法

4.2.1 X、θ、Θ、z、a

1. X

$$\begin{aligned} X &= \begin{pmatrix} (x^{(1)})^T \\ (x^{(2)})^T \\ (x^{(3)})^T \\ \vdots \\ (x^{(m)})^T \end{pmatrix} \\ &= \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_n^{(2)} \\ x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & \dots & x_n^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \dots & x_n^{(m)} \end{pmatrix} \\ &= \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_n^{(2)} \\ x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & \dots & x_n^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \dots & x_n^{(m)} \end{pmatrix} \Rightarrow (m, n) \end{aligned} \tag{37}$$

2. $a^{(1)}$

$$a^{(1)} = X \Rightarrow (m, n) \tag{38}$$

3. $\Theta^{(1)}$

$$\Theta^{(1)} = \begin{pmatrix} \theta_{10}^{(1)} & \theta_{11}^{(1)} & \theta_{12}^{(1)} & \dots & \theta_{1,s_1}^{(1)} \\ \theta_{20}^{(1)} & \theta_{21}^{(1)} & \theta_{22}^{(1)} & \dots & \theta_{2,s_1}^{(1)} \\ \theta_{30}^{(1)} & \theta_{31}^{(1)} & \theta_{32}^{(1)} & \dots & \theta_{3,s_1}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \theta_{s_20}^{(1)} & \theta_{s_21}^{(1)} & \theta_{s_22}^{(1)} & \dots & \theta_{s_2,s_1}^{(1)} \end{pmatrix} \Rightarrow (s_2, s_1 + 1) = (s_2, n + 1) \tag{39}$$

4. $z^{(2)}$

给$a^{(1)}$的每个数据均添加上$a_0=1$后与$\Theta^{(1)}$计算,得到$z^{(2)\text{注[1]}}=(1,a^{(1)})(\Theta^{(1)})^T$

$$
\begin{aligned}
z^{(2)} &= (1,a^{(1)})(\Theta^{(1)})^T \Rightarrow (m,n+1)*(n+1,s_2) \\
&= \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_n^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & \dots & x_n^{(3)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \dots & x_n^{(m)} \end{pmatrix} \begin{pmatrix} \theta_{10}^{(1)} & \theta_{11}^{(1)} & \theta_{12}^{(1)} & \dots & \theta_{1,n}^{(1)} \\ \theta_{20}^{(1)} & \theta_{21}^{(1)} & \theta_{22}^{(1)} & \dots & \theta_{2,n}^{(1)} \\ \theta_{30}^{(1)} & \theta_{31}^{(1)} & \theta_{32}^{(1)} & \dots & \theta_{3,n}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \theta_{s_2,0}^{(j)} & \theta_{s_2,1}^{(j)} & \theta_{s_2,2}^{(j)} & \dots & \theta_{s_2,n}^{(1)} \end{pmatrix}^T \\
&= \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_n^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & \dots & x_n^{(3)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \dots & x_n^{(m)} \end{pmatrix} \begin{pmatrix} \theta_{10}^{(1)} & \theta_{20}^{(1)} & \theta_{30}^{(1)} & \dots & \theta_{s_2,0}^{(1)} \\ \theta_{11}^{(1)} & \theta_{21}^{(1)} & \theta_{31}^{(1)} & \dots & \theta_{s_2,1}^{(1)} \\ \theta_{12}^{(1)} & \theta_{22}^{(1)} & \theta_{32}^{(1)} & \dots & \theta_{s_2,2}^{(1)} \\ \theta_{13}^{(1)} & \theta_{23}^{(1)} & \theta_{33}^{(1)} & \dots & \theta_{s_2,3}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \theta_{1,n}^{(1)} & \theta_{2,n}^{(1)} & \theta_{3,n}^{(1)} & \dots & \theta_{s_2,n}^{(1)} \end{pmatrix} \\
&= \begin{pmatrix} z^{(2)(1)} \\ z^{(2)(2)} \\ z^{(2)(3)} \\ \vdots \\ z^{(2)(m)} \end{pmatrix} \\
&= \begin{pmatrix} z_1^{(2)(1)} & z_2^{(2)(1)} & z_3^{(2)(1)} & \dots & z_{s_2}^{(2)(1)} \\ z_1^{(2)(2)} & z_2^{(2)(2)} & z_3^{(2)(2)} & \dots & z_{s_2}^{(2)(2)} \\ z_1^{(2)(3)} & z_2^{(2)(3)} & z_3^{(2)(3)} & \dots & z_{s_2}^{(2)(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ z_1^{(2)(m)} & z_2^{(2)(m)} & z_3^{(2)(m)} & \dots & z_{s_2}^{(2)(m)} \end{pmatrix} \\
&\Rightarrow (m,n+1)*(n+1,s_2) = (m,s_2)
\end{aligned} \tag{40}
$$

注[2]

5. $a^{(2)}$

$$a^{(2)} = g(z^{(2)}) \Rightarrow (m,s_2) \tag{41}$$

6. 后续同理

$$
\begin{aligned}
\Theta^{(2)} &= \begin{pmatrix} \theta_{10}^{(2)} & \theta_{11}^{(2)} & \theta_{12}^{(2)} & \dots & \theta_{1,s_2}^{(2)} \\ \theta_{20}^{(2)} & \theta_{21}^{(2)} & \theta_{22}^{(2)} & \dots & \theta_{2,s_2}^{(2)} \\ \theta_{30}^{(2)} & \theta_{31}^{(2)} & \theta_{32}^{(2)} & \dots & \theta_{3,s_2}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \theta_{s_30}^{(2)} & \theta_{s_31}^{(2)} & \theta_{s_32}^{(2)} & \dots & \theta_{s_3,s_2}^{(2)} \end{pmatrix} \Rightarrow (s_3, s_2+1) \\
z^{(3)} &= (1,a^{(2)})(\Theta^{(2)})^T \Rightarrow (m,s_2+1)*(s_2+1,s_3) = (m,s_3) \\
a^{(3)} &= g(z^{(3)}) \Rightarrow (m,s_3) \\
&\vdots
\end{aligned} \tag{42}
$$

注[1]从$a^{(1)}$得到$a^{(2)}$需要经过sigmoid()函数，后续的从$a^{(j)}$得到$a^{(j+1)}$均需要经过sigmoid()函数

注[2]上式$z_{s_2}^{(2)(m)}$中，(2)表示第2层神经网络，(m)表示第m个训练集，s_2表示第2层神经网络的最后一个单元

7. 一般式

$$
\begin{aligned}
a^{(j)} &= g(z^{(j-1)}) \Rightarrow (m, s_j) \\
\Theta^{(j)} &= \begin{pmatrix} \theta_{10}^{(j)} & \theta_{11}^{(j)} & \theta_{12}^{(j)} & \cdots & \theta_{1,s_j}^{(j)} \\ \theta_{20}^{(j)} & \theta_{21}^{(j)} & \theta_{22}^{(j)} & \cdots & \theta_{2,s_j}^{(j)} \\ \theta_{30}^{(j)} & \theta_{31}^{(j)} & \theta_{32}^{(j)} & \cdots & \theta_{3,s_j}^{(j)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \theta_{s_{j+1}0}^{(j)} & \theta_{s_{j+1}1}^{(j)} & \theta_{s_{j+1}2}^{(j)} & \cdots & \theta_{s_{j+1},s_j}^{(j)} \end{pmatrix} \Rightarrow (s_{j+1}, s_j + 1) \\
z^{(j+1)} &= (1, a^{(j)})(\Theta^{(j)})^T \\
&= \begin{pmatrix} 1 & a_1^{(j)(1)} & a_2^{(j)(1)} & a_3^{(j)(1)} & \cdots & a_{s_j}^{(j)(1)} \\ 1 & a_1^{(j)(2)} & a_2^{(j)(2)} & a_3^{(j)(2)} & \cdots & a_{s_j}^{(j)(2)} \\ 1 & a_1^{(j)(3)} & a_2^{(j)(3)} & a_3^{(j)(3)} & \cdots & a_{s_j}^{(j)(3)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & a_1^{(j)(m)} & a_2^{(j)(m)} & a_3^{(j)(m)} & \cdots & a_{s_j}^{(j)(m)} \end{pmatrix} \begin{pmatrix} \theta_{10}^{(j)} & \theta_{20}^{(j)} & \theta_{30}^{(j)} & \cdots & \theta_{s_{j+1},0}^{(j)} \\ \theta_{11}^{(j)} & \theta_{21}^{(j)} & \theta_{31}^{(j)} & \cdots & \theta_{s_{j+1},1}^{(j)} \\ \theta_{12}^{(j)} & \theta_{22}^{(j)} & \theta_{32}^{(j)} & \cdots & \theta_{s_{j+1},2}^{(j)} \\ \theta_{13}^{(j)} & \theta_{23}^{(j)} & \theta_{33}^{(j)} & \cdots & \theta_{s_{j+1},3}^{(j)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \theta_{1,s_j}^{(j)} & \theta_{2,s_j}^{(j)} & \theta_{3,s_j}^{(j)} & \cdots & \theta_{s_{j+1},s_j}^{(j)} \end{pmatrix} \\
&= \begin{pmatrix} z^{(j+1)(1)} \\ z^{(j+1)(2)} \\ z^{(j+1)(3)} \\ \vdots \\ z^{(j+1)(m)} \end{pmatrix} \\
&= \begin{pmatrix} z_1^{(j+1)(1)} & z_2^{(j+1)(1)} & z_3^{(j+1)(1)} & \cdots & z_{s_{j+1}}^{(j+1)(1)} \\ z_1^{(j+1)(2)} & z_2^{(j+1)(2)} & z_3^{(j+1)(2)} & \cdots & z_{s_{j+1}}^{(j+1)(2)} \\ z_1^{(j+1)(3)} & z_2^{(j+1)(3)} & z_3^{(j+1)(3)} & \cdots & z_{s_{j+1}}^{(j+1)(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ z_1^{(j+1)(m)} & z_2^{(j+1)(m)} & z_3^{(j+1)(m)} & \cdots & z_{s_{j+1}}^{(j+1)(m)} \end{pmatrix} \\
&\Rightarrow (m, s_j + 1) * (s_j + 1, s_{j+1}) = (m, s_{j+1}) \\
a^{(j+1)} &= g(z^{(j+1)}) \Rightarrow (m, s_{j+1})
\end{aligned}
\tag{43}
$$

4.2.2 y

$$
y = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \\ y^{(m)} \end{pmatrix}_{m*1} \tag{44}
$$

为进行矩阵运算，要将其转化为如下形式:注[3]

$$Y = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 1 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \dots & 0 & 0 \end{pmatrix}_{m,s_L} \tag{45}$$

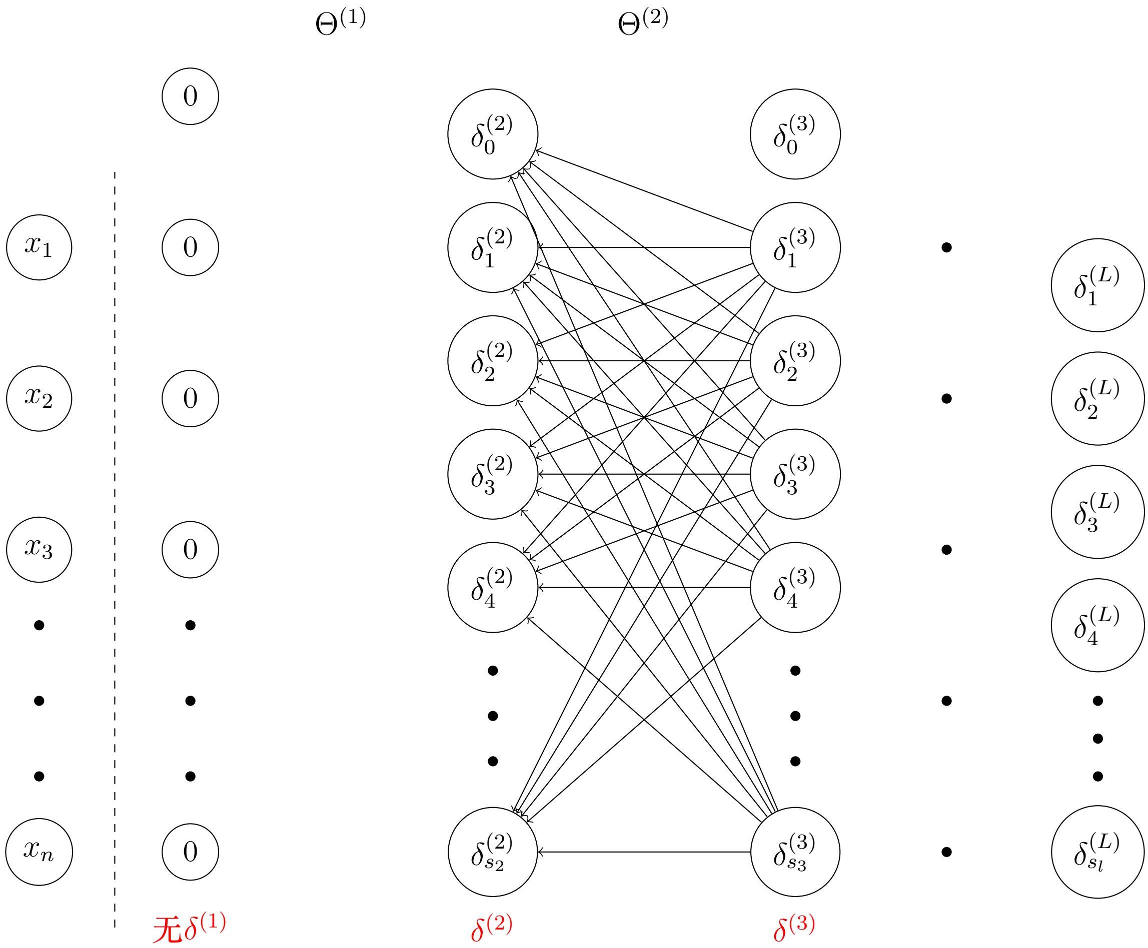
注[4]

注[3]y所对应的值所在的索引位置值为1，其他位置均为0

注[4]上式$m * s_L$中的s_L表示共有s_L个分类器，s_L表示的是输出层的unit数

5 神经网络 - 后向算法

5.1 神经网络示意图 - 后向算法



$$
\begin{cases}
\delta^L & = a^L - y, \quad l = L \\
\delta^{L-1} & = (\Theta^{(L-1)})^T \delta^L . * g^{'}(z^{L-1}) \\
 & = (\Theta^{(L-1)})^T \delta^L . * g(z^{L-1}) . * (1 - g(z^{L-1})), \quad l = L - 1 \\
\delta^l & = (\Theta^{(l)})^T [\delta^{l+1}(2 : end)] . * g^{'}(z^l) \\
 & = (\Theta^{(l)})^T [\delta^{l+1}(2 : end)] . * g(z^l) . * (1 - g(z^l)), \quad 2 <= l <= L - 2
\end{cases}
$$

无 $\delta^{(1)}$

$$\Delta^{(l)} := \Delta^{(l)} + \delta^{(l+1)} (a^{(l)})^T \tag{46}$$

$$
D_{ij}^{(l)} = \begin{cases} \frac{1}{m} \Delta_{ij}^{(l)}, & j = 0 \\ \frac{1}{m} (\Delta_{ij}^{(l)} + \Theta_{ij}^{(l)}), & j \neq 0 \end{cases}
$$

$$\frac{\partial J(\Theta)}{\partial \Theta_{ij}^{(l)}} = D_{ij}^{(l)}$$

5.2 神经网络 - 后向算法

5.2.1 输出层结果：a^L

$$a^L = \begin{pmatrix} a_1^{(L)(1)} & a_2^{(L)(1)} & a_3^{(L)(1)} & \dots & a_{s_L}^{(L)(1)} \\ a_1^{(L)(2)} & a_2^{(L)(2)} & a_3^{(L)(2)} & \dots & a_{s_L}^{(L)(2)} \\ a_1^{(L)(3)} & a_2^{(L)(3)} & a_3^{(L)(3)} & \dots & a_{s_L}^{(L)(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_1^{(L)(m)} & a_2^{(L)(m)} & a_3^{(L)(m)} & \dots & a_{s_L}^{(L)(m)} \end{pmatrix}_{m,s_L} \tag{47}$$

5.2.2 格式化后的Y

$$Y = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 1 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \dots & 0 & 0 \end{pmatrix}_{m,s_L} \tag{48}$$

5.2.3 δ^L

$$\begin{aligned} \delta^L &= a^L - y \\ &= \begin{pmatrix} a_1^{(L)} \\ a_2^{(L)} \\ a_3^{(L)} \\ \vdots \\ a_{s_L}^{(L)} \end{pmatrix} - \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \end{aligned} \tag{49}$$

- 此时，δ^L, a^L, y表示的是均向量（不是矩阵）

5.2.4 δ^{L-1}

$$\begin{aligned} \delta^{L-1} &= (\Theta^{(L-1)})^T \delta^L .* g'(z^{L-1}) \\ &= (\Theta^{(L-1)})^T \delta^L .* g(z^{L-1}) .* (1 - g(z^{L-1})) \end{aligned} \tag{50}$$

1. 其中，式$g'(z) = g(z)(1 - g(z))$，此为sigmoid()函数的特性
2. 此时，z^{L-1}表示的是一个向量（不是矩阵）
3. 此时不需要舍弃δ_0^L，因为根本就没有

5.2.5 $\delta^l (2 <= l <= L-2)$

$$\begin{aligned} \delta^l &= (\Theta^{(l)})^T [\delta^{l+1}(2:end)] .* g'(z^l) \\ &= (\Theta^{(l)})^T [\delta^{l+1}(2:end)] .* g(z^l) .* (1 - g(z^l)) \end{aligned} \tag{51}$$

1. 因$a^{(1)}$直接从X得到，不会有误差，故无$\delta^{(1)}$
2. (2:end)表示舍弃第一个数据$\delta_0^{s_{L-1}}$（Matlab索引从1开始）
3. 对比于从a^l到a^{l+1}要添加一个$a_0^l = 1$；从δ^{l+1}到δ^l要舍弃一个δ_0^{l+1}
4. 同样地，此时z^l表示的是一个向量（不是矩阵）

5.2.6 Δ^l（用迭代的方式计算）

1. 数值计算方式

$$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)} \tag{52}$$

2. 矩阵计算方式

$$\Delta^{(l)} := \Delta^{(l)} + \delta^{(l+1)} (a^{(l)})^T \tag{53}$$

5.2.7 $D_{ij}^{(l)}$

1. $j = 0$时

$$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} \tag{54}$$

2. $j \neq 0$时

$$D_{ij}^{(l)} := \frac{1}{m} (\Delta_{ij}^{(l)} + \Theta_{ij}^{(l)}) \tag{55}$$

5.2.8 $\frac{\partial J(\Theta)}{\partial \Theta_{ij}^{(l)}}$

$$\frac{\partial J(\Theta)}{\partial \Theta_{ij}^{(l)}} = D_{ij}^{(l)} \tag{56}$$

5.2.9 δ^l 与 Δ^l的区别与联系

6 调试技巧

6.1 Error Analysis

1. 0-1错分率 or 误分类率 1. 将当前的数据分成2份，70%作为训练集，30%作为测试集
   2. 用新的训练集训练，用新的测试集检验效果
   3. 若$J_{train}(\theta)$很小，$J_{test}(\theta)$很大，则说明出现了过拟合

2. 训练集 & 交叉验证集 & 测试集

3. 过拟合、欠拟合的判断方法 1. 欠拟合对应高偏差，表现为$J_{cv}(\theta) \approx J_{train}(\theta)$，且二者都很高
   2. 过拟合对应高方差，表现为$J_{cv}(\theta) >> J_{train}(\theta)$，且$J_{train}(\theta)$很小
   3. $J_{taain}(\theta)$对应训练集的学习能力；$J_{cv}(\theta)$对应训练结果对新样本的适应能力，适应能力越强，$J_{cv}(\theta)$越小
   4. 在训练集和验证集（测试集）效果均不好，说明欠拟合；在训练集效果很好，但在说明欠拟合，但在验证集（测试集）效果不好，说明过拟合

6.2 Error Metrics for Skewed Classes

1. Skewed Classes 那些两种（或多种）情况发生的概率相关较大的情况称为Skewed Classes。如买彩票中奖的概率与不中奖的概率

2. True vs. False & Positive vs. Negative Predicted:1, Actual:1 — True Positive — TP;
   Predicted:0, Actual:0 — True Negative —TN;
   Predicted:1, Actual:0 — False Positive — FP;
   Predicted:0, Actual:0 — False Negative — FN;
   True & False 对应预测的是否正确;
   Positive & Negative 对应实际是否发生

3. Accuracy & Precision & Recall Accuracy: 发生的概率：$\frac{True}{False} = \frac{TP+TN}{TP+TN+FP+FN}$; Precision: 查准率，在预测为1的情况下，实际为1的概率：$fracTPTP+FP$
   Recall：召回率，在实际为1的情况下，被预测出来的概率: $\frac{TP}{TP+FN}$

4. Precision & Recall均是越高越好，但实际上，两者无法同时都很高。（PS：两者加起来并不一定会等于1，甚至很小情况下才全等于1））

6.3 如何评价Precision与Recall

1. 使用$F_1 Score = 2\frac{PR}{P+R}$

2. 用交叉验证集的F_1值来选取最大的F_1值对应的P和R，不用训练集（或测试集）中的。

6.4 拟合效果不好时的解决方法指导

1. 获取更多数据 —- 解决高方差

2. 减少特征 —- 解决高方差

3. 增加特征 —- 解决高偏差

4. 增加高阶多项式 — 解决高偏差

5. 减小λ —- 解决高偏差
6. 增大λ —- 解决高方差

6.5 不同神经网络的优缺点

1. 小型神经网络 – 更少的参数；容易出现欠拟合
2. 大型神经网络 – 更多的参数；容易出现过拟合
3. parameters越复杂，或隐藏的层越多，对训练集的拟合效果越好，但若对验证集的拟合效果不好，说明已经过拟合，此时再增加神经网络的复杂度并不能提高神经网络的效果

6.6 绘制Learning Curve

绘制Learning Curve时，对训练集计算训练误差时，每次迭代只能使用训练集的部分数据（第i次迭代使用第1到第i个数据）；但对验证集计算验证误差时，每次均应使用所有数据

7 SVM

7.1 Cost Function

$$J(\theta) = C\sum_{i=1}^{m} y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)}) + \frac{1}{2}\sum_{j=1}^{n}\Theta_j^2 \tag{57}$$

其中，$cost_1(\theta^T x^{(i)})$对应$y=1$; $cost_0(\theta^T x^{(i)})$对应$y=0$

7.2 Gaussian Kernel

$$f_i = similarity(x, l^{(i)}) = exp(-\frac{\sum_{j=1}^{n}(x_j - l_j^{(i)})^2}{2\sigma^2}) \tag{58}$$

1. 当$x \approx l^{(i)}$时，$f_i = exp(-\frac{\approx 0^2}{2\sigma^2}) \approx 1$
2. 当x远离$l^{(i)}$时，$f_i = exp(-\frac{inf^2}{2\sigma^2}) \approx 1$
3. 3

7.3 SVM中，C与σ^2对欠拟合或过拟合的影响

1. C：C过大：低偏差，高方差；C过小：高偏差，第方差
2. λ^2: 过大: 高偏差，低方差；过小: 低偏差，高方差

7.4 如何选项使用Logistic Regression还是 SVM

1. n很大时，使用Logistic Regression或无kernel（即linear kernel）的SVM
2. n很小，m中等：使用Gaussian kernel的SVM
3. n很小，m很大：增加特征，并使使用Logistic Regression或无kernel（即linear kernel）的SVM
4. 神经网络可能更加适合，但是使用神经网络训练需要耗费的时间较长