Last Update Date：2021/2/28-17:18

Machine Learning-Stanford

2 Linear Regression with One Variable

Linear Regression Model：

$$\begin{aligned}
&\text{Hypothesis:} && h_\theta(x) = \theta_0 + \theta_1 x \\
&\text{Parameters:} && \theta_0, \theta_1 \\
&\text{Cost Function:} && J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \\
&\text{Goal:} && \text{minimize } J(\theta_0, \theta_1)
\end{aligned}$$

- m 代表训练集中实例数量
- x 代表特征/输入变量
- y 代表目标变量/输出变量
- (x, y) 代表训练实例
- $(x^{(i)}, y^{(i)})$ 代表第 i 个训练实例

Gradient Descent Algorithm :

$$\begin{aligned}
&\text{repeat until convergence:} \\
&\qquad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad \text{simultaneously update } j = 0 \text{ and } j = 1
\end{aligned}$$

- $:=$：赋值操作（Assignment）
- α：学习率（learning rate），决定了梯度负方向上的下降步长

使用梯度下降算法求 $J(\theta_0, \theta_1)$ 的最小值，对代价函数 $J(\theta_0, \theta_1)$ 求偏导数：

$$\begin{aligned}
\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) &= \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) && j = 0 \\
\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) &= \frac{1}{m} \sum_{i=1}^{m} \left(\left(h_\theta(x^{(i)}) - y^{(i)} \right) x^{(i)} \right) && j = 1
\end{aligned}$$

梯度下降算法改写为：

$$\begin{aligned}
&\text{repeat until convergence:} \\
&\qquad \begin{matrix} \text{update } \theta_0 \text{ and } \theta_1 \\ \text{simultaneously} \end{matrix} \begin{cases} \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) & j = 0 \\ \theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(\left(h_\theta(x^{(i)}) - y^{(i)} \right) x^{(i)} \right) & j = 1 \end{cases}
\end{aligned}$$

4 Linear Regression with Multiple Variable

4.1 Gradient Descent for Multiple Variables

$$\text{Hypothesis:} \qquad h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

上面的假设中有 $n+1$ 个参数 $(\theta_0, \theta_1, \ldots, \theta_n)$，$n$ 个变量 $(x_1, x_2, \ldots, x_n)$，为了简化公式，引入变量 $x_0 = 1$。

令

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}, \ \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

则

$$h_\theta\left(x\right)=\theta^{T}x$$

Linear Regression Model：

$$\begin{array}{lll}
\text{Hypothesis:} & h_\theta\left(x\right)=\theta_0x_0+\theta_1x_1+\theta_2x_2+\ldots+\theta_nx_n & (x_0=1) \\
\text{Parameters:} & \theta_0,\theta_1,\ldots,\theta_n & \\
\text{Cost Function:} & J\left(\theta_0,\theta_1,..,\theta_n\right)=\dfrac{1}{2m}\sum\limits_{i=1}^{m}\left(h_\theta(x^{(i)})-y^{(i)}\right)^2 & \\
\text{Goal:} & \text{minimize } J\left(\theta_0,\theta_1,..,\theta_n\right) &
\end{array}$$

- n 代表训练集中特征数量
- $x_j^{(i)}$ 代表第 i 个训练实例的第 j 个特征

Gradient Descent Algorithm：

$$\begin{array}{l}
\text{repeat until convergence:} \\
\qquad \theta_j:=\theta_j-\alpha\dfrac{\partial}{\partial\theta_j}J(\theta_0,\theta_1,\ldots,\theta_n) \qquad \text{simultaneously update for every } j=0,1,\ldots,n
\end{array}$$

使用梯度下降算法求 $J\left(\theta_0,\theta_1,..,\theta_n\right)$ 的最小值，对代价函数 $J\left(\theta_0,\theta_1,..,\theta_n\right)$ 求偏导数：

$$\begin{array}{ll}
\dfrac{\partial}{\partial\theta_0}J\left(\theta_0,\theta_1,..,\theta_n\right)=\dfrac{1}{m}\sum\limits_{i=1}^{m}\left(\left(h_\theta(x^{(i)})-y^{(i)}\right)x_0{}^{(i)}\right) & j=0 \\
\dfrac{\partial}{\partial\theta_1}J\left(\theta_0,\theta_1,..,\theta_n\right)=\dfrac{1}{m}\sum\limits_{i=1}^{m}\left(\left(h_\theta(x^{(i)})-y^{(i)}\right)x_1{}^{(i)}\right) & j=1 \\
\ldots & \\
\dfrac{\partial}{\partial\theta_n}J\left(\theta_0,\theta_1,..,\theta_n\right)=\dfrac{1}{m}\sum\limits_{i=1}^{m}\left(\left(h_\theta(x^{(i)})-y^{(i)}\right)x_n{}^{(i)}\right) & j=n
\end{array}$$

Gradient Descent for Multiple Variables：

$$\begin{array}{l}
\text{repeat until convergence:} \\
\qquad \theta_j:=\theta_j-\alpha\dfrac{1}{m}\sum\limits_{i=1}^{m}\left(\left(h_\theta(x^{(i)})-y^{(i)}\right)x_j{}^{(i)}\right) \qquad \text{simultaneously update } \theta_j \text{ for every } j=0,1,\ldots,n
\end{array}$$

4.2 Feature Scaling

特征缩放可以使梯度下降法收敛速度更快，迭代次数更少。

使每个特征的范围缩放大约至 $-1\leqslant x_i\leqslant 1$

$$x_i=\frac{x_i-\mu_i}{s_i}$$

- μ_i 为特征 x_i 的平均值
- s_i 为特征 x_i 的极差或者标准差

4.3 Normal Equation

同梯度下降法，正规方程用来解决以下问题：

$$\text{Goal:} \qquad \text{minimize } J\left(\theta_0,\theta_1,..,\theta_n\right)$$

正规方程通过求解以下方程来确定代价函数 $J\left(\theta_0,\theta_1,..,\theta_n\right)$ 取最小值时 $\theta_0,\theta_1,..,\theta_n$ 的值：

$$\frac{\partial}{\partial\theta_j}J\left(\theta_0,\theta_1,..,\theta_n\right)=0 \qquad \text{for every } j=0,1,\ldots,n$$

即

$$\begin{aligned}\frac{1}{m}\sum_{i=1}^{m}\left(\left(h_\theta(x^{(i)})-y^{(i)}\right)x_j{}^{(i)}\right)&=0\\ \sum_{i=1}^{m}\left(\left(h_\theta(x^{(i)})-y^{(i)}\right)x_j{}^{(i)}\right)&=0\\ \sum_{i=1}^{m}\left(\left(\theta_0x_0^{(i)}+\theta_1x_1^{(i)}+\theta_2x_2^{(i)}+\ldots+\theta_nx_n^{(i)}-y^{(i)}\right)x_j{}^{(i)}\right)&=0\end{aligned}$$

令

$$x^{(i)}=\begin{bmatrix}x_0^{(i)}\\x_1^{(i)}\\\vdots\\x_n^{(i)}\end{bmatrix}\qquad X=\begin{bmatrix}x^{(1)T}\\x^{(2)T}\\\vdots\\x^{(m)T}\end{bmatrix}=\begin{bmatrix}x_0^{(1)}&x_1^{(1)}&\ldots&x_n^{(1)}\\x_0^{(2)}&x_1^{(2)}&\ldots&x_n^{(2)}\\\vdots&\vdots&\ddots&\vdots\\x_0^{(m)}&x_1^{(m)}&\ldots&x_n^{(m)}\end{bmatrix}\qquad\theta=\begin{bmatrix}\theta_0\\\theta_1\\\vdots\\\theta_n\end{bmatrix}\qquad y=\begin{bmatrix}y^{(1)}\\y^{(2)}\\\vdots\\y^{(m)}\end{bmatrix}$$

$$\begin{aligned}X^T(X\theta-y)&=\begin{bmatrix}x_0^{(1)}&x_0^{(2)}&\ldots&x_0^{(m)}\\x_1^{(1)}&x_1^{(2)}&\ldots&x_1^{(m)}\\\vdots&\vdots&\ddots&\vdots\\x_n^{(1)}&x_n^{(2)}&\ldots&x_n^{(m)}\end{bmatrix}\begin{bmatrix}x_0^{(1)}\theta_0+x_1^{(1)}\theta_1+\ldots+x_n^{(1)}\theta_n-y^{(1)}\\x_0^{(2)}\theta_0+x_1^{(2)}\theta_1+\ldots+x_n^{(2)}\theta_n-y^{(2)}\\\vdots\\x_0^{(m)}\theta_0+x_1^{(m)}\theta_1+\ldots+x_n^{(m)}\theta_n-y^{(m)}\end{bmatrix}\\&=\begin{bmatrix}\sum_{i=1}^{m}\left(\left(\theta_0x_0^{(i)}+\theta_1x_1^{(i)}+\theta_2x_2^{(i)}+\ldots+\theta_nx_n^{(i)}-y^{(i)}\right)x_0{}^{(i)}\right)\\\sum_{i=1}^{m}\left(\left(\theta_0x_0^{(i)}+\theta_1x_1^{(i)}+\theta_2x_2^{(i)}+\ldots+\theta_nx_n^{(i)}-y^{(i)}\right)x_1{}^{(i)}\right)\\\vdots\\\sum_{i=1}^{m}\left(\left(\theta_0x_0^{(i)}+\theta_1x_1^{(i)}+\theta_2x_2^{(i)}+\ldots+\theta_nx_n^{(i)}-y^{(i)}\right)x_n{}^{(i)}\right)\end{bmatrix}\\&=\underbrace{\begin{bmatrix}0&0&\ldots&0\end{bmatrix}^T}_{n-dimension}\\&=\mathrm{O}\end{aligned}$$

进一步

$$\begin{aligned}X^T(X\theta-y)&=\mathrm{O}\\X^TX\theta&=X^Ty\\\theta&=(X^TX)^{-1}X^Ty\end{aligned}$$

Normal Equation：

$$\theta=(X^TX)^{-1}X^Ty$$

Compare Normal Equation with Gradient Descent：

Gradient Descent	**Normal Equation**
需要选择学习率 α	不需要选择学习率 α
需要多次迭代	不需要迭代
当特征数量 n 很大时也能较好适用	需要计算 $\left(X^TX\right)^{-1}$ ，如果特征数量 n 较大则运算代价大，适用于 $n < 10000$
适用于其他类型的模型	只适用于线性模型，不适合逻辑回归模型等其他模型

6 Logistic Regression

$$\text{Hypothesis:} \qquad h_\theta\left(x\right) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$

- $g(z) = \frac{1}{1+e^{-z}}$ （called Sigmoid function or Logistic function）
- Interpretation of Hypothesis Output：

  $h_\theta(x) = P(y=1|x;\theta)$ （probability that $y=1$, given x , parameterized by θ）
- $0 \leqslant h_\theta(x) \leqslant 1$
- The function image of $h_\theta(x)$ is called the **Decision Boundary**

$$\begin{aligned} &\text{Cost Function:} && J\left(\theta\right) = \frac{1}{m}\sum_{i=1}^{m} Cost\left(h_\theta(x^{(i)}), y^{(i)}\right) \\ & && Cost\left(h_\theta(x^{(i)}), y^{(i)}\right) = \begin{cases} -\log\left(h_\theta(x^{(i)})\right) & \text{if } y=1 \\ -\log\left(1-h_\theta(x^{(i)})\right) & \text{if } y=0 \end{cases} \\ &\text{Simplified Cost Function:} && J\left(\theta\right) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(h_\theta(x^{(i)})\right) + \left(1-y^{(i)}\right)\log\left(1-h_\theta(x^{(i)})\right)\right] \end{aligned}$$

- $J(\theta) = J(\theta_0, \theta_1, \ldots \theta_n)$

Logistic Regression Model：

$$\begin{aligned} &\text{Hypothesis:} && h_\theta\left(x\right) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}} \\ &\text{Parameters:} && \theta_0, \theta_1, \ldots, \theta_n \\ &\text{Cost Function:} && J\left(\theta\right) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(h_\theta(x^{(i)})\right) + (1-y^{(i)})\log\left(1-h_\theta(x^{(i)})\right)\right] \\ &\text{Goal:} && \text{minimize } J(\theta) \end{aligned}$$

使用梯度下降算法求 $J\left(\theta\right)$ 的最小值，对代价函数 $J\left(\theta\right)$ 求偏导数：

$$\begin{aligned}
\frac{\partial}{\partial\theta_j}J(\theta) &= \frac{\partial}{\partial\theta_j}\left[-\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(h_\theta(x^{(i)})\right)+(1-y^{(i)})\log\left(1-h_\theta(x^{(i)})\right)\right]\right] \\
&= -\frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial\theta_j}\left[y^{(i)}\log\left(\frac{1}{1+e^{-\theta^T x^{(i)}}}\right)+(1-y^{(i)})\log\left(1-\frac{1}{1+e^{-\theta^T x^{(i)}}}\right)\right] \\
&= -\frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial\theta_j}\left[-y^{(i)}\log(1+e^{-\theta^T x^{(i)}})+(1-y^{(i)})\left(-\theta^T x^{(i)}-\log(1+e^{-\theta^T x^{(i)}})\right)\right] \\
&= -\frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial\theta_j}\left[-\log(1+e^{-\theta^T x^{(i)}})+(y^{(i)}-1)\theta^T x^{(i)}\right] \\
&= -\frac{1}{m}\sum_{i=1}^{m}\left[\frac{e^{-\theta^T x^{(i)}}}{1+e^{-\theta^T x^{(i)}}}x_j^{(i)}+(y^{(i)}-1)x_j^{(i)}\right] \qquad \frac{\partial}{\partial\theta_j}\theta^T x^{(i)}=x_j^{(i)} \\
&= \frac{1}{m}\sum_{i=1}^{m}\left(\left(h_\theta(x^{(i)})-y^{(i)}\right)x_j{}^{(i)}\right)
\end{aligned}$$

Gradient Descent for Logistic Regression:

$$\begin{aligned}
&\text{repeat until convergence:} \\
&\qquad \theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(\left(h_\theta(x^{(i)})-y^{(i)}\right)x_j{}^{(i)}\right) \qquad \text{simultaneously update } \theta_j \text{ for every } \; j=0,1,\ldots,n
\end{aligned}$$

6.1 Multi-class Classification：One vs All

现有 k 类别分类问题，我们可以将其转变为多个独立的二元分类问题。

在训练集中，将类别 i 设置为正类别($y=i$)，将类别 $1,2...,i-1,i+1,...,k$ 都设置为负类别，训练逻辑回归分类器 $h_\theta{}^{(i)}(x)$，预测 $P(y=i|x;\theta)$ 。训练完成之后，得到 k 个分类器 $h_\theta{}^{(i)}(x) \quad (i=1,2,\ldots,k)$。

对新的样本输入 x 做出分类预测，将其分类为类别 i，i 使得 $h_\theta{}^{(i)}(x)$ 最大：

$$\max_i \; h_\theta^{(i)}\left(x\right)$$

7 Regularization

7.1 The Problem of Overfitting

过拟合：若训练样本有过多的特征，训练出的假设 $h_\theta(x)$ 可能会很好的拟合训练集，$J(\theta)\approx 0$，但是这将导致它无法泛化到新样本中。

防止过拟合：

- 减少特征数量
  - 手动选择保留那些特征
  - 模型选择算法
- 正则化：保留所有特征，但是降低参数 θ_j 的值

7.2 Cost Function

降低参数 θ_j 的值，以简化假设 $h_\theta(x)$

$$\text{Regularized Cost Function:} \qquad J\left(\theta\right)=\frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)})-y^{(i)}\right)^2+\lambda\sum_{j=1}^{n}\theta_j^2\right]$$

- $\lambda \sum_{j=1}^{n} \theta_j^2$ 为正则化项。
- λ 为正则化参数（Regularization Parameter），用来控制两个目标之间平衡关系：
  - 第一个目标，使假设 $h_\theta(x)$ 更好的拟合训练集
  - 第二个目标，降低参数 θ_j 的值，从而简化假设 $h_\theta(x)$，避免出现过拟合现象

7.3 Regularized Linear Regression

代价函数加上正则项：

$$\text{Cost Function:} \qquad J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}\theta_j{}^2\right]$$

Gradient Descent：

$$\begin{aligned}&\text{repeat until convergence:}\\ &\qquad \theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_0{}^{(i)}\right)\\ &\qquad \theta_j := \left(1 - \alpha\frac{\lambda}{m}\right)\theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j{}^{(i)}\right) \qquad j = 1, 2, \ldots, n\end{aligned}$$

Normal Equation： (why?)

$$\theta = \left(X^TX + \lambda\begin{bmatrix}0 & 0 & 0 & \ldots & 0\\ 0 & 1 & 0 & \ldots & 0\\ 0 & 0 & 1 & \ldots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \ldots & 1\end{bmatrix}\right)^{-1} X^Ty$$

7.4 Regularized Logistic Regression

代价函数加上正则项：

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

Gradient Descent：

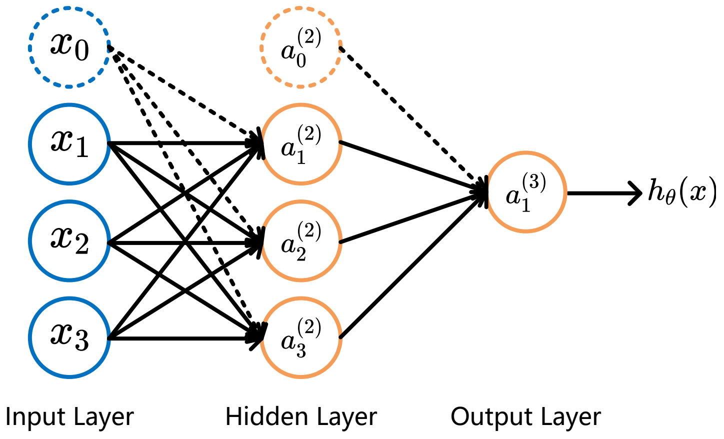
$$\begin{aligned}&\text{repeat until convergence:}\\ &\qquad \theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_0{}^{(i)}\right)\\ &\qquad \theta_j := \left(1 - \alpha\frac{\lambda}{m}\right)\theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j{}^{(i)}\right) \qquad j = 1, 2, \ldots, n\end{aligned}$$

8 Neural Networks: Representation

8.1 Model Representation

Neural Network:

- x 为输入单元（input unit）
- a 为神经元（neuron）或者激活单元（activation unit）
- x_0, a_0^2 为偏差单元（bias unit）



引入一些符号来更好的描述神经网络：

- $a_i^{(j)}$ 表示第 j 层的第 i 个激活单元
- $\Theta^{(j)}$ 表示从第 j 层到第 $j+1$ 层的权重矩阵，

  若神经网络的第 j 层有 s_j 个激活单元，第 $j+1$ 层有 s_{j+1} 个激活单元，那么 $\Theta^{(j)}$ 的大小为 $s_{j+1} \times (s_j + 1)$

对于上图所示的神经网络，可以表示为：

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

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

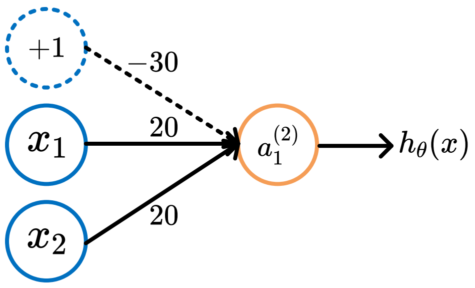
Vectorized Implementation:

$$a^{(1)} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \begin{aligned} a^{(2)} &= g(\Theta^{(1)} a^{(1)}) \\ \text{Add} \quad a_0^{(2)} &= 1 \\ h_\theta(x) &= g(\Theta^{(2)} a^{(2)}) \end{aligned}$$

8.2 Examples and Intuitions

拟合简单逻辑运算的神经网络

AND $(x_1 \wedge x_2)$

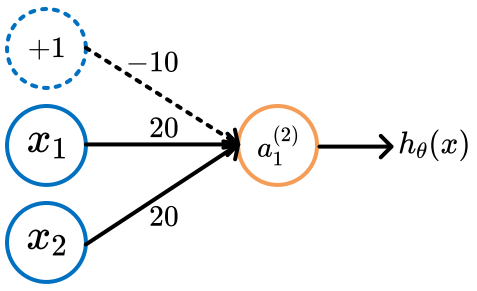


x_1	x_2	$h_\theta(x)$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

- $g(z)=\frac{1}{1+e^{-z}}$
- $x_1, x_2 \in \{0,1\}$

其中权重矩阵 $\Theta^{(1)}=[-30 \quad 20 \quad 20]$，则假设 $h_\Theta(x)=g(-30+20x_1+20x_2)$

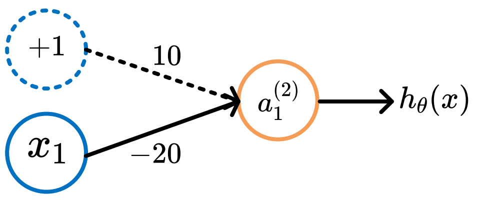
OR $(x_1 \vee x_2)$



x_1	x_2	$h_\theta(x)$
0	0	$g(-10)\approx 0$
0	1	$g(10)\approx 1$
1	0	$g(10)\approx 1$
1	1	$g(30)\approx 1$

其中权重矩阵 $\Theta^{(1)}=[-10 \quad 20 \quad 20]$，则假设 $h_\Theta(x)=g(-10+20x_1+20x_2)$

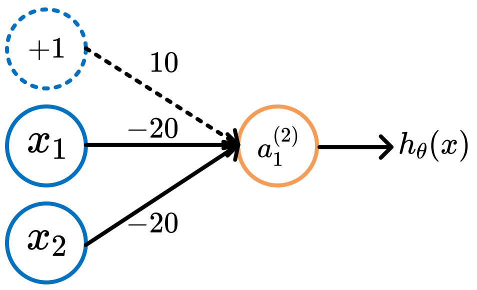
NOT $(\neg x_1)$



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

其中权重矩阵 $\Theta^{(1)}=[10 \quad -20]$，则假设 $h_\Theta(x)=g(10-20x_1)$

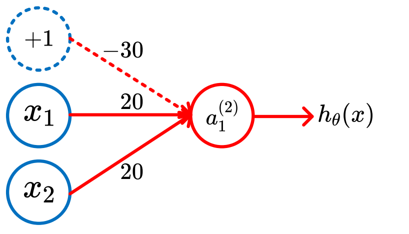
(NOT x_1) AND (NOT x_2) $(\neg x_1 \wedge \neg x_2)$



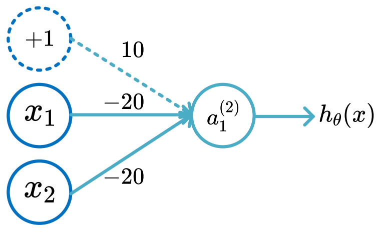
x_1	x_2	$h_\theta(x)$
0	0	$g(10)\approx 1$
0	1	$g(-10)\approx 0$
1	0	$g(-10)\approx 0$
1	1	$g(-30)\approx 0$

其中权重矩阵 $\Theta^{(1)}=[10 \quad -20 \quad -20]$，则假设 $h_\Theta(x)=g(-10+20x_1+20x_2)$

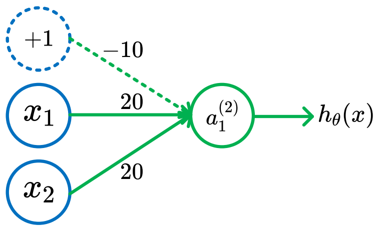
XNOR (NOT XOR) $\neg(x_1 \oplus x_2)=(x_1 \wedge x_2)\vee(\neg x_1 \wedge \neg x_2)$



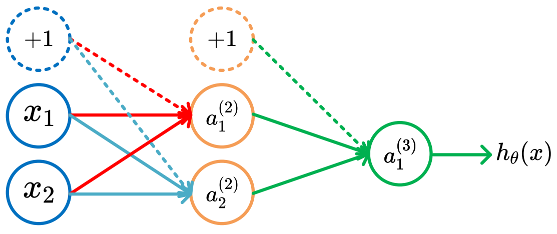
$x_1 \wedge x_2$



$\neg x_1 \wedge \neg x_2$



$x_1 \vee x_2$



$\neg(x_1 \oplus x_2)=(x_1 \wedge x_2)\vee(\neg x_1 \wedge \neg x_2)$

x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_\theta(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

8.3 Muilt-class Classification

对于 K 类型分类问题，训练集：$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$，其中

$$y(i) \in \mathbb{R}^K \text{ is one of } \begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix}, \begin{bmatrix}0\\1\\\vdots\\0\end{bmatrix}, \dots, \begin{bmatrix}0\\0\\\vdots\\1\end{bmatrix}$$

若 $y_k^{(i)} = 1$，则 x^i 分类的类别 k

用神经网络进行 K 类别分类，在输出层有 K 个激活单元，假设 $h_\theta(x) \in \mathbb{R}^K$

$$\text{Cost Function:} \quad J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[y_k^{(i)}\log\left(h_\theta^k(x^{(i)})\right) + (1-y_k^{(i)})\log\left(1-h_\theta^k(x^{(i)})\right)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_{j+1}}\sum_{j=1}^{s_j}{\Theta_{ij}^{(l)}}^2$$

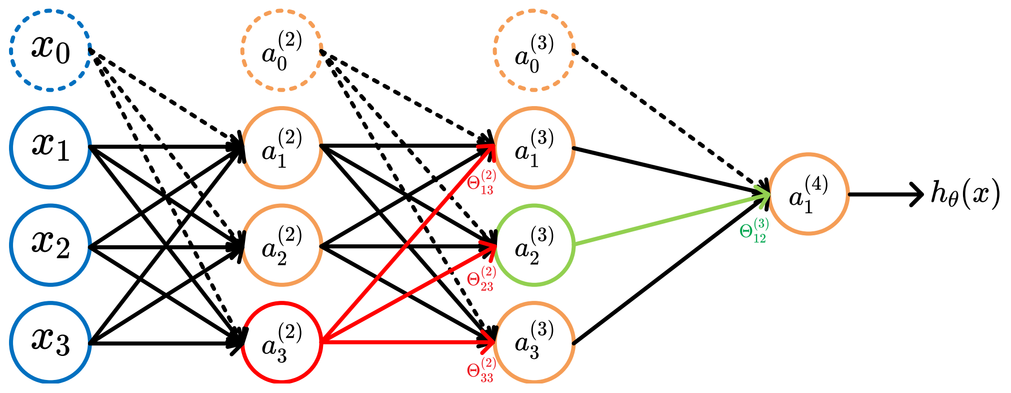
- L 表示神经网络的层数
- s_j 表示第 j 层中激活单元数
- $h_\theta^k(x)$ 表示输出层中第 k 个输出
- $\Theta_{ij}^{(l)}$ 表示从第 l 层到第 $l+1$ 层的权重矩阵中元素 (i, j)

8.4 Backpropagation Algorithm

为了使用梯度下降算法或者其他优化算法实现 minimize $J(\theta)$，都需要计算 $\frac{\partial}{\partial\Theta_{ij}^{(l)}}J(\theta)$，可以采用反向传播算法计算。

$$\begin{aligned}&\text{Output Layer:} \quad && \delta^{(L)} = a^{(L)} - y_j \\ &\text{Hidden Layer:} \quad && \delta^{(l)} = {\Theta^{(l)}}^T\delta^{(l+1)}.*g'(z^{(l)}) = {\Theta^{(l)}}^T\delta^{(l+1)}.*a^{(l)}.*(1-a^{(l)}) \qquad l = L-1, L-2, \dots, 2\end{aligned}$$

- $\delta_j^{(l)}$ 表示第 l 层中第 j 个激活单元的预测误差（不需要对输入层计算误差）
- $z^{(l)} = \Theta^{(l-1)}a^{(l-1)}$
- $a^{(l)} = g(z^{(l)})$
- $.*$ 表示矩阵点乘



$$\begin{aligned}\delta_1^{(4)} &= a_1^{(4)} - y^{(i)} \\ \delta_2^{(3)} &= \Theta_{12}^{(3)}\delta_1^{(4)}a_2^{(3)}(1-a_2^{(3)}) \\ \delta_3^{(2)} &= (\Theta_{13}^{(2)}a_1^{(3)} + \Theta_{23}^{(2)}a_2^{(3)} + \Theta_{33}^{(2)}a_3^{(3)})a_3^{(2)}(1-a_3^{(2)})\end{aligned}$$

Backpropagation Algorithm：

```
Train set: $\{(x^{(1)},y^{(1)}),(x^{(2)},y^{(2)}),\dots(x^{(m)},y^{(m)})\}$
Set $\Delta_{ij}^{(l)}=0$ for all $i=1,2,\dots,s_{l+1};j=0,1,2,\dots,s_l;l=1,2,\dots,L-1$
For $i=1$ to $m$
    Set $a^{(1)}=x^{(i)}$
    Perform forward propagation to compute $a^{(l)}$ for $l=2,3,\dots,L$
    Using $y^{(i)}$, compute $\delta^{(L)}=a^{(L)}-y^{(i)}$
    Compute $\delta^{(L-1)},\delta^{(L-2)},\dots,\delta^{(2)}$, remove $\delta_0^{(l)}$
    $\Delta^{(l)}:=\Delta^{(l)}+\delta^{(l+1)}{a^{(l)}}^T$
$\frac{\partial}{\partial\Theta_{ij}^{(l)}}J(\theta):=\begin{cases}\frac{1}{m}\Delta_{ij}^{(l)}+\frac{\lambda}{m}\Theta_{ij}^{(l)} & \text{if } j\neq 0\\ \frac{1}{m}\Delta_{ij}^{(l)} & \text{if } j=0\end{cases}$
```

8.5 Gradient Cheaking

为了检测使用反向传播算法求得的 $\frac{\partial}{\partial\Theta_{ij}^{(l)}}J(\theta)$ 是否正确，提出梯度检测法。

由双侧差分求导数近似值：

$$\frac{\mathrm{d}J(\theta)}{\mathrm{d}\theta}\approx\frac{J(\theta+\varepsilon)-J(\theta-\varepsilon)}{2\varepsilon}$$

可以得到 $\frac{\partial}{\partial\Theta_{ij}^{(l)}}J(\theta)$ 的近似值：

$$\begin{aligned}\frac{\partial}{\partial\theta_1}&\approx\frac{J(\theta_1+\varepsilon,\theta_2,\dots,\theta_t)-J(\theta_1-\varepsilon,\theta_2,\dots,\theta_t)}{2\varepsilon}\\ \frac{\partial}{\partial\theta_2}&\approx\frac{J(\theta_1,\theta_2+\varepsilon,\dots,\theta_t)-J(\theta_1,\theta_2-\varepsilon,\dots,\theta_t)}{2\varepsilon}\\ \dots\\ \frac{\partial}{\partial\theta_t}&\approx\frac{J(\theta_1,\theta_2,\dots,\theta_t+\varepsilon)-J(\theta_1,\theta_2,\dots,\theta_t-\varepsilon)}{2\varepsilon}\end{aligned}$$

- $\theta\in\mathbb{R}^t$ 为所有权重矩阵 $\Theta^{(1)},\Theta^{(2)},\dots,\Theta^{(L-2)}$ 的向量展开形式，t 为所有权重矩阵中参数的总数量
- ε 一般取 10^{-4} 为宜

比较反向传播算法求得的 $\frac{\partial}{\partial\Theta_{ij}^{(l)}}J(\theta)$ 与双侧差分法求得的 $\frac{\partial}{\partial\Theta_{ij}^{(l)}}J(\theta)$ 的近似值，确保它们是相近的值。

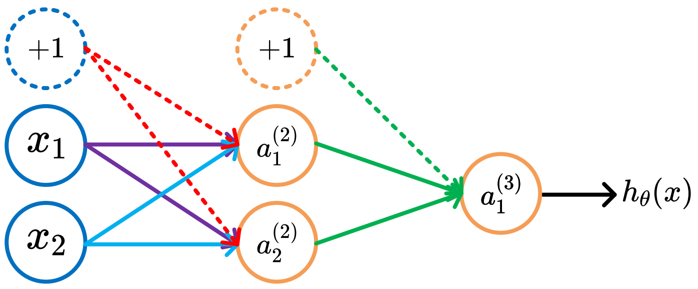
8.6 Random Initialization

在初始化神经网络的权重矩阵中参数时，不要将所有参数都初始化为同一值。

若将所有参数都初始化为同一值，在训练时会出现权重矩阵中同一列参数都相等的情况，且这种情况将延续整个训练过程：

$$\Theta^{(l)}=\begin{bmatrix}\Theta_{10}^{(l)} & \Theta_{11}^{(l)} & \dots & \Theta_{1s_l}^{(l)}\\ \Theta_{20}^{(l)} & \Theta_{21}^{(l)} & \dots & \Theta_{2s_l}^{(l)}\\ \vdots & \vdots & \ddots & \vdots\\ \Theta_{s_{l+1}0}^{(l)} & \Theta_{s_{l+1}1}^{(l)} & \dots & \Theta_{s_{l+1}s_l}^{(l)}\end{bmatrix}\qquad(\Theta_{1j}^{(l)}=\Theta_{2j}^{(l)}=\dots=\Theta_{s_{l+1}j}^{(l)},j=0,1,\dots,s_l)$$

如下图（相同颜色的线条表示相同的权值）：



这将导致：

$$a_1^{(l)} = a_2^{(l)} = \cdots = a_{s_l}^{(l)}$$

是一种高度冗余的现象，同一层的所有激活单元都只能获取到同一特征，神经网络学习效率低下。

所以，通常将权重矩阵中参数随机初始化：

$$\text{initialize each } \Theta_{ij}^{(l)} \text{ to a random value in } [-\epsilon, +\epsilon]$$

- 可以根据 $\Theta^{(l)}$ 的尺寸来选择 ϵ

$$\epsilon = \sqrt{\frac{6}{s_l + s_{l+1}}}$$

- 这里的 ϵ 与梯度检测中的 ε 无关

8.7 Putting It Together

Training a neural network:

- 1.Randomly initialize weights
- 2.Implement forword propagation to compute $h_\theta(x^{(i)})$ for every $x^{(i)}$
- 3.Implement code to compute cost function $J(\theta)$
- 4.Implement back propagation to compute partial derivatives $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\theta)$
- 5.Use gradient checking to compare $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\theta)$ computed using back propagation and using numerical estimate of $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\theta)$

  Then disable gradient checking code
- 6.Use gradient descent or advanced optimization method with back propagation to try to minimize $J(\theta)$ as a function of parameter Θ

9 Advice for Applying Machine Learning

9.1 Deciding What to Try Next

当训练之后的机器学习模型在测试时表现的很糟糕时，应该怎么改进：

- Get more training examples
- Try smaller sets of features
- Try getting additional features
- Try adding polynomial features $(x_1^2, x_2^2, x_1 x_2, \text{etc.})$
- Try decreasing λ
- Try increasing λ

上面的改进方法不是随便选择的，需要通过机器学习诊断来确定上面哪一种改进方法最有效。

Machine Learning Diagnostic

Diagnostic：A test that you can run and gain insight what is/isn't working with a learning algorithm,and gain guidance as to how best to improve its performance.

9.2 Evaluating a Hypothesis

为了对模型进行评价，将数据分为两部分，训练集（training set）、测试集（test set），典型的做法是随机选择 70% 作为训练集，剩余的 30% 作为测试集。

Training/test procedure for linear regression：

- 从训练集中学习得到参数 θ
- 计算测试集误差：

$$J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} \left(h_\theta(x_{test}^{(i)}) - y_{test}^{(i)} \right)^2$$

  - m_{test} 代表测试集中实例数量
  - $(x_{test}^{(i)}, y_{test}^{(i)})$ 代表第 i 个测试实例

Training/test procedure for logistic regression：

- 从训练集中学习得到参数 θ
- 计算测试集误差（两种方式）：
  - $\text{Cost Function:} \qquad J_{test}(\theta) = -\frac{1}{m_{test}} \sum_{i=1}^{m_{test}} \left[y_{test}{}^{(i)} \log\left(h_\theta(x_{test}{}^{(i)})\right) + (1 - y_{test}{}^{(i)}) \log\left(1 - h_\theta(x_{test}{}^{(i)})\right)\right]$
  - Misclassification error（0/1 misclassification）

$$test\ error = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} err\left(h_\theta(x_{test}^{(i)}), y_{test}^{(i)}\right)$$

$$err\left(h_\theta(x), y\right) = \begin{cases} 1 & \text{if } \ h_\theta(x) \geqslant threshold, y = 0 \ \text{ or } \ h_\theta(x) < threshold, y = 1 \\ 0 & \text{otherwise} \end{cases}$$

    - $threshold$ 代表阈值，取值区间 $(0, 1)$

9.3 Model Selection and Training/Validation/Test Sets

在进行模型选择时（如在对多项式模型次数的选择），可以使用以下方式来对不同的模型进行对比评估。

将数据分为三部分，训练集（training set）、交叉验证集（cross validation set）、测试集（test set），经典比例 60%，20%，20%。

Training/Validation/Test Sets Error:

$$\begin{aligned}
&\text{training error:} && J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \\
&\text{cross validation error:} && J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} \left(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)} \right)^2 \\
&\text{test error:} && J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} \left(h_\theta(x_{test}^{(i)}) - y_{test}^{(i)} \right)^2
\end{aligned}$$

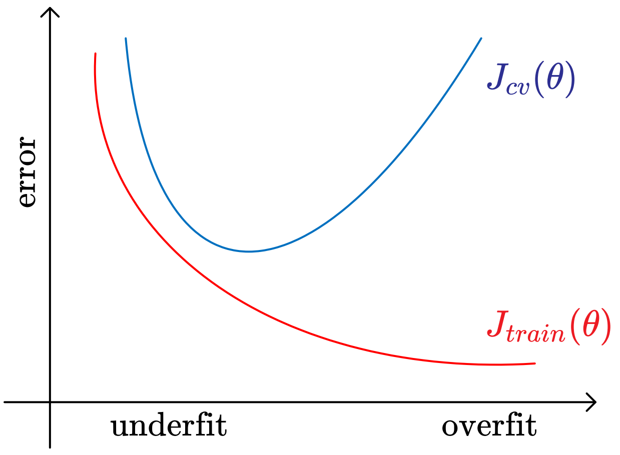
有 k 个模型 $h_{\theta^{(1)}}(x), h_{\theta^{(2)}}(x), \dots, h_{\theta^{(k)}}(x)$，需要从中选出表现最好的一个模型：

- 这 k 个模型分别从训练集中学习得到参数 $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(k)}$

- 使用交叉验证集比较这 k 个模型，选择使得 $J_{cv}(\theta^{(i)})$ 最小的模型 i
- 最后使用测试集对模型 i 进行评估，即计算泛化误差

9.4 Diagnosing Bias vs Variance

当模型在新样本中表现很糟糕时，多数是因为两种情况：偏差（bias）过大或者方差（variance）过大，也就是欠拟合（underfit）或者过拟合（overfit），要对模型进行改进，必须知道模型是偏差过大还是方差过大。



High bias（underfit）：

$$\begin{gathered} J_{train} \text{ will be high} \\ J_{cv}(\theta) \approx J_{train}(\theta) \end{gathered}$$

High variance（overfit）：

$$\begin{gathered} J_{train} \text{ will be low} \\ J_{cv}(\theta) \gg J_{train}(\theta) \end{gathered}$$

9.5 Regularization and Bias/Variance

Choosing the regularization parameter λ for regularized linear regression：

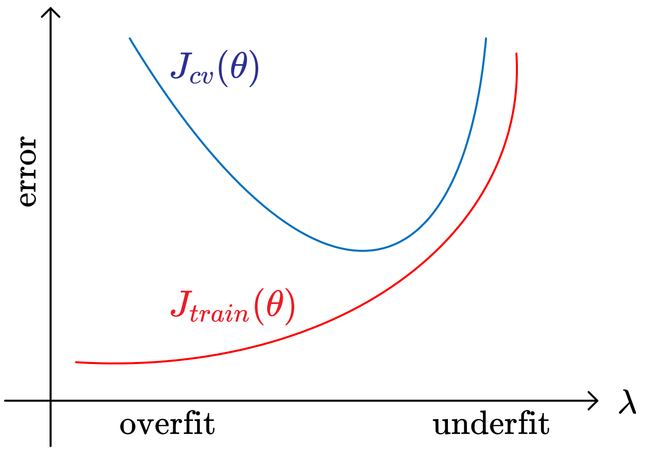
$$\begin{aligned}
&\text{Cost Function:} && J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}{\theta_j}^2\right] \\
&\text{training error:} && J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 \\
&\text{cross validation error:} && J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}\left(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)}\right)^2 \\
&\text{test error:} && J_{test}(\theta) = \frac{1}{2m_{test}}\sum_{i=1}^{m_{test}}\left(h_\theta(x_{test}^{(i)}) - y_{test}^{(i)}\right)^2
\end{aligned}$$

- 注意：在误差计算时没有加上正则化项

有 k 个正则化参数 $\lambda^{(1)}, \lambda^{(2)}, \ldots, \lambda^{(k)}$，需要从中选出最合适的正则化参数（通常在选择正则化参数时，将供选的正则化参数设置为 $0, 0.01, 0.02, 0.04, \ldots, 10.24$）：

- 以 $\lambda^{(1)}, \lambda^{(2)}, \ldots, \lambda^{(k)}$ 为正则化参数的模型分别从训练集中学习得到参数 $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(k)}$
- 使用交叉验证集比较这 k 个正则化参数，选择使得 $J_{cv}(\theta^{(i)})$ 最小的正则化参数 $\lambda^{(i)}$
- 最后使用测试集对以 $\lambda^{(i)}$ 为正则化参数的模型进行评估，即计算泛化误差

Bias/Variance as a function of the regularization parameter λ：

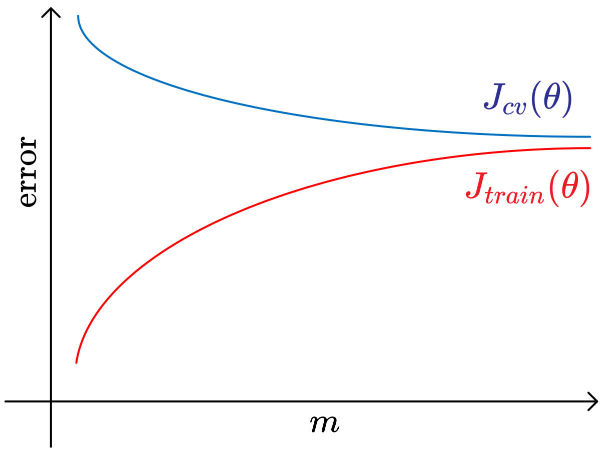


- 当 λ 很小时，正则化项对 $\theta_j\ (j=1,2,\ldots,n)$ 的约束很小，导致过拟合，也即高方差
- 当 λ 很大时，对 $\theta_j\ (j=1,2,\ldots,n)$ 的“惩罚”过大，所以 θ_j 在数值上都很小，导致欠拟合，也即高偏差

9.6 Learning Curves

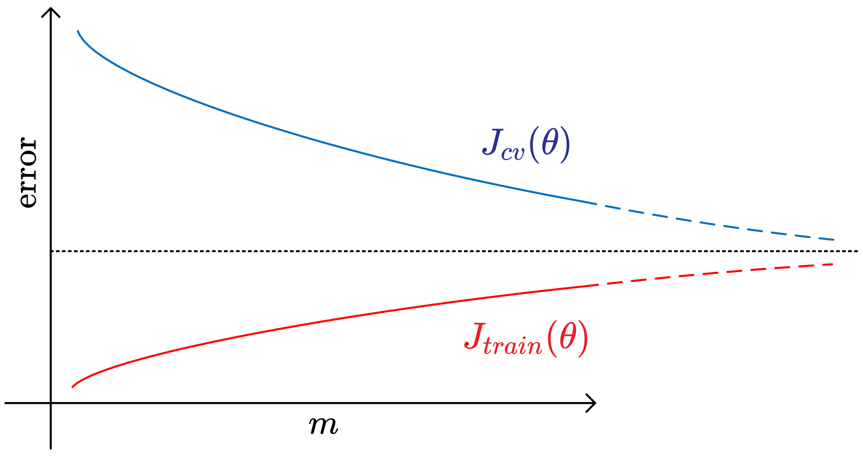
学习曲线是训练集误差 $J_{train}(\theta)$ 与交叉验证集误差 $J_{cv}(\theta)$ 关于训练集实例数量 m 的曲线，可以反映模型是模型是处于高偏差还是高方差。

High bias（underfit）：



- 当出现高偏差时，增加训练集实例数量 m 并不会对模型有太大改进

High variance（overfit）：



- 当出现高方差时，增加训练集实例数量 m 可能会带来帮助

9.7 Deciding What to Try Next（revisited）

Get more training examples	**fixes high variance**
Try smaller sets of features	**fixes high variance**
Try getting additional features	**fixes high bias**
Try adding polynomial features（$x_1^2, x_2^2, x_1x_2, \text{etc.}$）	**fixes high bias**
Try decreasing λ	**fixes high bias**
Try increasing λ	**fixes high variance**

Neural networks and overfitting：

Small Neural Networks	Large Neural Networks
fewer parameters	more parameters
more prone to underfitting	more prone to overfitting（use regularization to address overfitting）
computationally cheaper	computationally more expensive

10 Machine Learning System Design

10.1 Error Metrics for Skewed Classes

在分类问题中，不同标签之间的数量差距十分大，则称这种情况为偏斜类（Skewed Classes），当使用分类误差或者分类准确率在对模型做出评估时并不能反映其真实情况，在这里需要提出一种新的模型评估方法来度量模型在处理偏斜类分类问题上的表现。

Precision/Recall：

$y = 1$ in presence of rare class that we want to detect

predicted\actual	Positive	Negative
Positive	TP（True Positive）	FP（False Positive）
Negative	FN（False Negative）	TN（True Negative）

- TP（True Positive）表示样本的真实类别为正，最后预测得到的结果也为正
- FP（False Positive）表示样本的真实类别为负，最后预测得到的结果却为正
- FN（False Negative）表示样本的真实类别为正，最后预测得到的结果却为负
- TN（True Negative）表示样本的真实类别为负，最后预测得到的结果也为负

$$
Accuracy = \frac{TP+TN}{TP+FP+FN+TN}
$$
$$
Precision = \frac{TP}{TP+FP}
$$
$$
Recall = \frac{TP}{TP+FN}
$$

- $Accuracy$（准确率）
- $Precision$（查准率）：在所有预测为正类别的样本中，正确预测为正样本的概率
- $Recall$（召回率）：在所有真实为正类别的样本中，正确预测为正样本的概率

对于同时拥有高查准率与高召回率的模型，可以被评估为表现好。

10.2 Trading off Precision and Recall

$$
\text{Predict} \quad = \begin{cases} 1 & \text{if } h_\theta(x) \geqslant threshold \\ 0 & \text{otherwise} \end{cases}
$$

- 当 $threshold$ 取值越接近于 1，$Precision$ 越大，$Recall$ 越小
- 当 $threshold$ 取值越接近于 0，$Precision$ 越小，$Recall$ 越大

对于查准率与召回率的权衡，对模型做出评估：

$$
F_1 Score = 2\frac{PR}{P+R}
$$

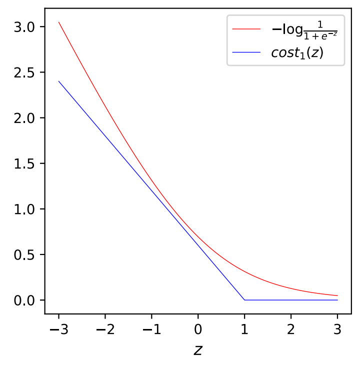
$F_1 Score$ 也可以帮助选择 $threshold$：

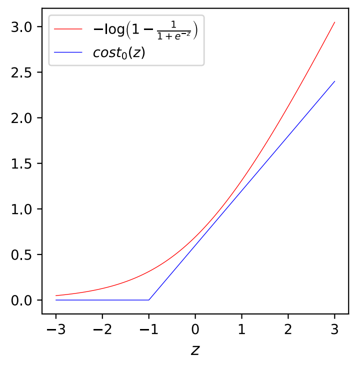
- 设置不同的 $threshold$ 在交叉验证集上测试
- 选择使得 $F_1 Score$ 最高的 $threshold$

11 Support Vector Mechine

11.1 Optimization Objective

Alternative view of logistic regression：





- $z = \theta^T x$

Logistic Regression Cost Function：

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

SVM Cost Function（no kernel）：

$$J\left(\theta\right)=C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^Tx^{(i)})+\left(1-y^{(i)}\right)cost_0(\theta^Tx^{(i)})\right]+\frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

- C 的作用类似于 $\frac{1}{\lambda}$

SVM Hypothesis：

$$h_\theta(x)=\begin{cases}1 & \text{if } \theta^Tx\geqslant 0\\ 0 & \text{otherwise}\end{cases}$$

11.2 Kernels

SVM with Kernels：

将每一个训练实例 x 都作为一个标记 l（landmark）

$$\begin{gathered}\text{Given } (x^{(1)},y^{(1)}),(x^{(2)},y^{(2)}),\ldots(x^{(m)},y^{(m)})\\ \text{Choose } l^{(1)}=x^{(1)},l^{(2)}=x^{(2)},\ldots,l^{(m)}=x^{(m)}\end{gathered}$$

$$\begin{aligned}&\text{For training example } (x^{(i)},y^{(i)}):\\ &\qquad f_1^{(i)}=similarity(x^{(i)},l^{(1)})\\ &\qquad f_2^{(i)}=similarity(x^{(i)},l^{(2)})\\ &\qquad\quad\ldots\\ &\qquad f_i^{(i)}=similarity(x^{(i)},l^{(i)})=exp(-\frac{\left\|x^{(i)}-l^{(i)}\right\|^2}{2\sigma^2})=1\\ &\qquad\quad\ldots\\ &\qquad f_m^{(i)}=similarity(x^{(i)},l^{(m)})\end{aligned}\qquad f^{(i)}=\begin{bmatrix}f_0^{(i)}\\ f_1^{(i)}\\ f_2^{(i)}\\ \vdots\\ f_m^{(i)}\end{bmatrix}\qquad(f_0^{(i)}=1)$$

- $similarity(x,l^{(i)})$（相似度函数）代表 x 与 $l^{(i)}$ 相似度的度量，也被称为核函数（Kernel），在这里使用的是高斯核

$$k(x,l^{(i)})=exp(-\frac{\left\|x-l^{(i)}\right\|^2}{2\sigma^2})$$

Model Representation：

$$\begin{aligned}&\text{Hypothesis:} && h_\theta\left(x\right)=\begin{cases}1 & \text{if } \theta^Tf\geqslant 0\\ 0 & \text{otherwise}\end{cases}\\ &\text{Cost Function:} && J\left(\theta\right)=C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^Tf^{(i)})+\left(1-y^{(i)}\right)cost_0(\theta^Tf^{(i)})\right]+\frac{1}{2}\sum_{j=1}^{m}\theta_j^2\end{aligned}$$

- $\theta\in\mathbb{R}^{m+1}$，故正则化项求和上标 $n=m$

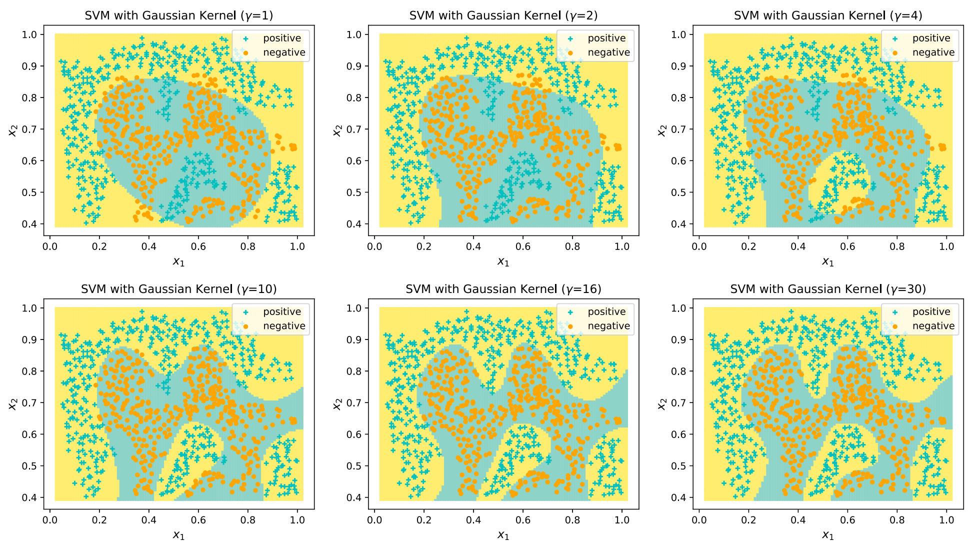
SVM Parameters：

C（类似于$\frac{1}{\lambda}$）

- Large C：high variance，low bias
- Small C：high bias，low variance

σ^2

- Small σ^2：f vary sharply，high variance，low bias
- Large σ^2：f vary smoothly，high bias，low variance



- 上图中高斯核是另一种表示形式：

$$k(x, l^{(i)}) = exp(-\gamma \left\| x - l^{(i)} \right\|^2)$$

11.3 Using an SVM

Logistic Regression vs SVM

- if n is large （relative to m）

  Use logistic regression or SVM without kernel

- if n is small， m is intermediate

  Use SVM with Gaussian kernel

- if n is small， m is large

  Crear/Add more features，then use logistic regression or SVM without kernel

12 Clustering

12.1 Unsupervised Learning Introduction

Unsupervised vs Supervised

- 监督学习：在一个有标签的训练集中，目标是找到能够区分正负样本的决策边界，据此需要拟合一个假设函数。
- 无监督学习：训练样本的标记信息是未知的，目标是通过对无标记训练样本的学习来揭示数据的内在性质及规律，为进一步的数据分析提供基础。

12.2 K-means Algorithm

K-means Algorithm：

```
Input:
    $K$ (number of clusters)
    Training set $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$
Randomly initialize $K$ cluster centroids $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$
Repeat:
    $C_i = \varnothing \ (i = 1, 2, \dots, K)$
    for $i = 1$ to $m$ (cluster assignment)
        $\lambda^{(i)} = \min\limits_{k \in \{1,2,\dots,K\}} \|x^{(i)} - \mu_k\|^2$
        $C_{\lambda^{(i)}} = C_{\lambda^{(i)}} \cup \{x^{(i)}\}$
    for $k = 1$ to $K$ (move cluster centroids)
        $\mu_k = \frac{1}{|C_k|} \sum\limits_{x \in C_k} x$
Until $\mu_1, \mu_2, \dots, \mu_K$ no longer to update
Output:
    $C = \{C_1, C_2, \dots, C_K\}$
```

- $x^{(i)} \in \mathbb{R}^n$，删去 $x^{(0)} = 1$ 的惯例
- $\mu^{(k)}$ 称为聚类中心（cluster centroids），若希望将样本聚类成 K 簇，那么应该设置 K 个聚类中心
- 若没有样本被分配到簇 k，常见的做法是直接删除聚类中心 μ_k，最终只会得到 $K-1$ 个簇，或者选择重新随机初始化聚类中心
- $|C_k|$ 表示簇 k 中样本的数量

12.2 Optimization Objective

K-means optimization objective：

$$\begin{aligned} &\text{(Distortion) Cost Function:} && J(\lambda^{(1)}, \lambda^{(2)}, \dots, \lambda^{(m)}, \mu_1, \mu_2, \dots, \mu_K) = \frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - \mu_{\lambda^{(i)}}\|^2 \\ &\text{Goal:} && \text{minimize } J(\lambda^{(1)}, \lambda^{(2)}, \dots, \lambda^{(m)}, \mu_1, \mu_2, \dots, \mu_K) \end{aligned}$$

- $\lambda^{(i)}$: index of cluster（$1, 2, \dots, K$）to which example $x^{(i)}$ is currently assigned
- μ_k : cluster centroid k（$\mu_k \in \mathbb{R}^n$）
- $\mu_{\lambda^{(i)}}$: cluster centroid of cluster to which example $x^{(i)}$ has been assigned
- 在 K-means 算法中
  - 簇分配通过更新 $\lambda^{(1)}, \lambda^{(2)}, \dots, \lambda^{(m)}$ 以 minimize J
  - 移动聚类中心通过更新 $\mu_1, \mu_2, \dots, \mu_K$ 以 minimize J

12.3 Random Initialization

随机初始化聚类中心：随机选择 K 个训练样本，令 K 个聚类中心分别与这 K 个训练样本相等

K-means 算法由于随机初始化聚类中心，可能只达到局部最优，通常可以多次运行 K-means 算法，每一次重新随机初始化，最终选择使得代价函数 J 最小的聚类结果。

13 Dimensionality Reducion

13.1 Principal Component Analysis Algorithm

Object:

试图寻找一个低维“平面”，对高维数据集进行投影，并最小化投影平方误差。

Data Preprocessing:

Training Set: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

- mean normalization

  $\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$

  $\text{Replace each } x_j^{(i)} \text{ with } x_j - \mu_j$

- feature scaling

Principal Component Analysis:

- ①计算协方差矩阵 Σ（Σ 为对称矩阵）：

$$\Sigma = \frac{1}{m}\sum_{i=1}^{n}(x^{(i)})(x^{(i)})^T$$

- ②计算 Σ 的特征向量（P 的列向量即为 Σ 的特征向量，且 P 为正交矩阵，使用到对称矩阵的对角化）：

$$\Sigma = P\Lambda P^T$$

  将 P 使用列向量表示 $P = (p_1, p_2, \ldots, p_n)$

- ③取 P 的前 k 个列向量，组成矩阵 $P_{reduce} = (p_1, p_2, \ldots, p_k)$

$$z^{(i)} = {P_{reduce}}^T x^{(i)} \qquad (z \in \mathbb{R}^{k\times 1})$$

Note:

- 协方差矩阵为正定矩阵（why？）
- 若训练集:

$$X = \begin{bmatrix} {x^{(1)}}^T \\ {x^{(2)}}^T \\ \vdots \\ {x^{(m)}}^T \end{bmatrix}_{m\times n}$$

  则

$$\Sigma = \frac{1}{m}X^T X \qquad z = \begin{bmatrix} {z^{(1)}}^T \\ {z^{(2)}}^T \\ \vdots \\ {z^{(m)}}^T \end{bmatrix}_{m\times k} = XP_{reduce}$$

- $x^{(i)} \in \mathbb{R}^n$

13.2 Choosing the Number of Principal Components

平均投影误差：

$$\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}-x_{approx}^{(i)}\|^2$$

- $x_{approx}^{(i)}$ 为 $x^{(i)}$ 投影到低维“平面”的坐标

数据总方差：

$$\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2$$

Choosing k（number of principal components）：

- 方式1：

  $$\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}-x_{approx}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2}\leqslant 0.01$$

  - “99% 的方差被保留”
  - k 越大，上式结果越小。寻找满足上式时，k 的最小值
- 方式2：

  对 Σ 进行奇异值奇异值分解：

  $$USV^T=\Sigma$$

  寻找满足

  $$\frac{\sum\limits_{i=1}^{k}S_i}{\sum\limits_{i=1}^{m}S_i}\geqslant 0.99$$

  时，k 的最小值

  - S_i 表示 S 主对角线上从左上到右下第 i 个元素
  - “99% 的方差被保留”
  - 上式的意义：主成分数量为 k 时，原始高维数据集信息保留程度的度量

13.3 Reconstruction from Compress Representation

由 PCA：

$$z^{(i)}={P_{reduce}}^Tx^{(i)}$$

由 $z^{(i)}$ 近似重构 $x^{(i)}$：

$$x_{approx}^{(i)}=P_{reduce}z^{(i)}$$

- $x_{approx}^{(i)}$ 为 $x^{(i)}$ 投影到低维“平面”的坐标
- $$x_{approx}=\begin{bmatrix}{x_{approx}^{(m)}}^T\\ {x_{approx}^{(m)}}^T\\ \vdots\\ {x_{approx}^{(m)}}^T\end{bmatrix}_{m\times n}=z{P_{reduce}}^T$$
-

13.4 Advice for applying PCA

Supervised Learning Speedup：

Training Set：$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$
After PCA
New Training Set：$(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \ldots, (z^{(m)}, y^{(m)})$

- 在 Training Set 中使用 PCA **定义**映射 f：$x^{(i)} \to z^{(i)}$，即 P_{reduce} 是对 Training Set 主成分分析得到
- 将映射 f **应用**在CV Set、Test Set，以获得 New CV Set、New Test Set

Bad use of PCA：To fix overfitting

Please use regularization instead

PCA is sometimes used where it shouldn't be

Before implementing PCA, first try running whatever you want to do with the original data $x^{(i)}$, only if that doesn't do what you want, then implement PCA and consider using $z^{(i)}$.