17.4.24（机器学习的一些建议，BP算法MATLAB实现）

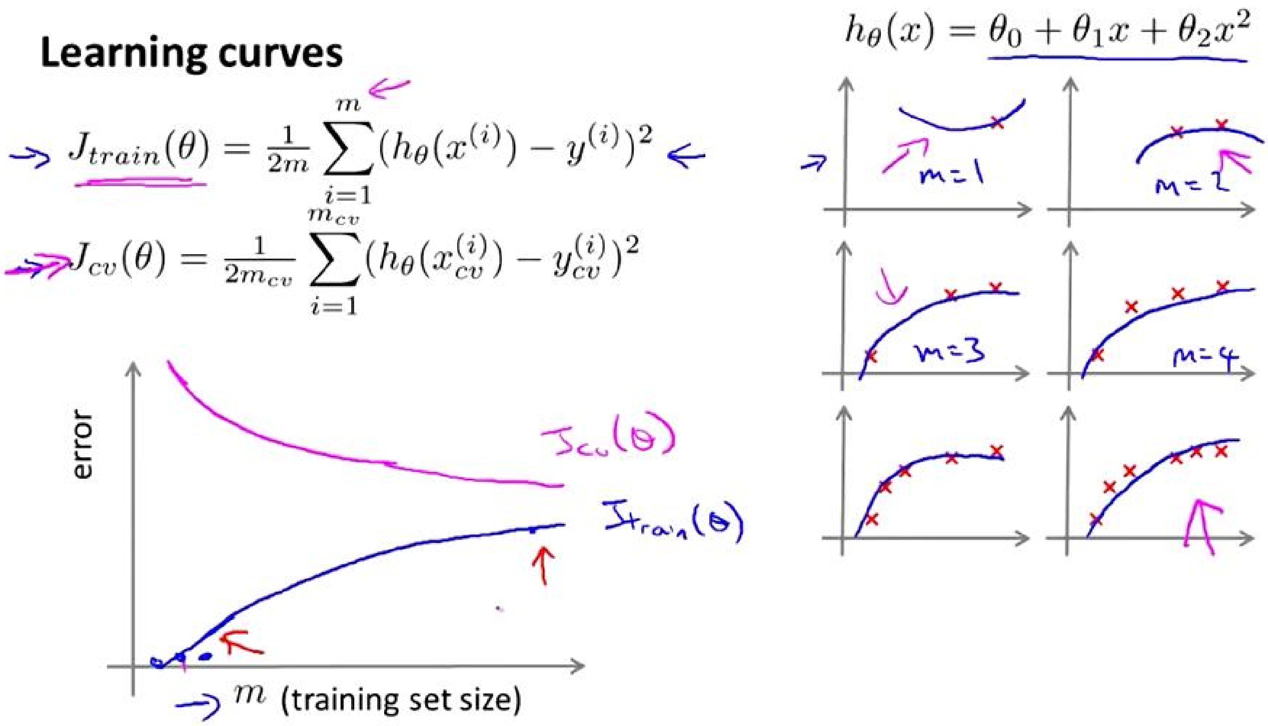
机器学习的一些建议（Advice for Applying Machine Learning）

- 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 $(x_1^2, x_2^2, x_1 x_2, \text{etc})$ → fixes high bias.
- Try decreasing λ → fixes high bias
- Try increasing λ → fixes high variance

当我们的拟合曲线是过拟合时：我们一般的解决办法是，获得更多的训练集，尝试减少特征的数量，尝试增加lambda。当我们的拟合曲线是欠拟合时，解决办法：尝试更多的特征，尝试增加多项式特征，尝试减少lambda。

学习曲线是学习算法的一个很好的 合理检验（sanity check）。学习曲线是将

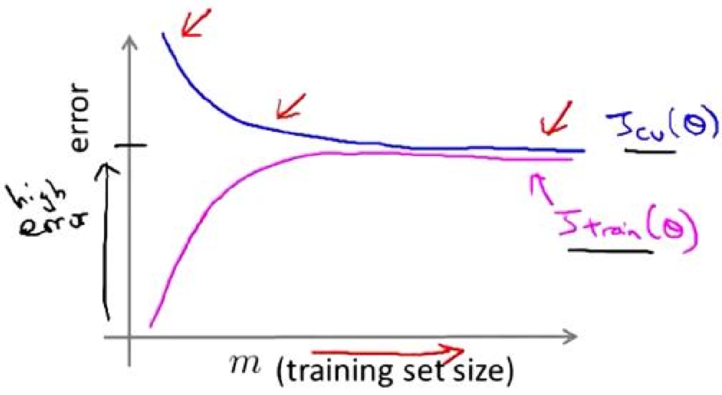
训练集误差和交叉验证集误差作为训练集实例数量（m）的函数绘制的图表。

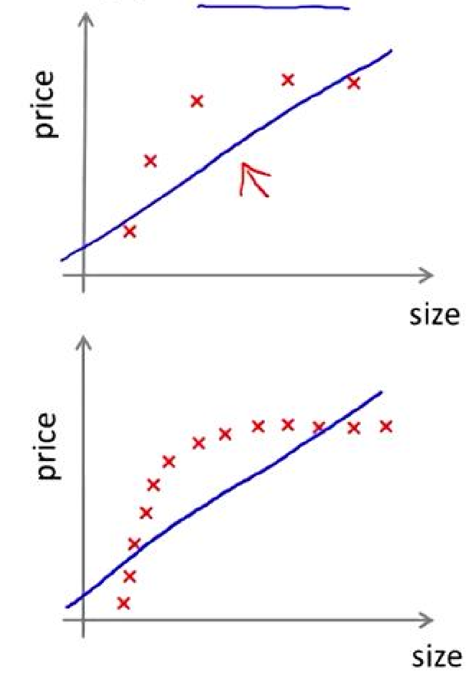


当训练集较少时，得出来的假设函数，在cv集或更多的训练集时，就不能很好地拟合。

High bias

$h_\theta(x) = \theta_0 + \theta_1 x$



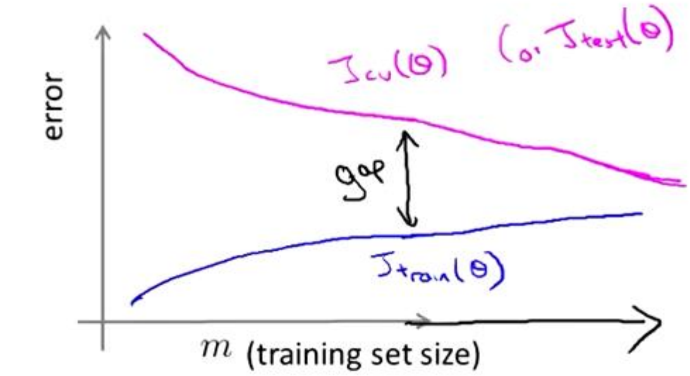


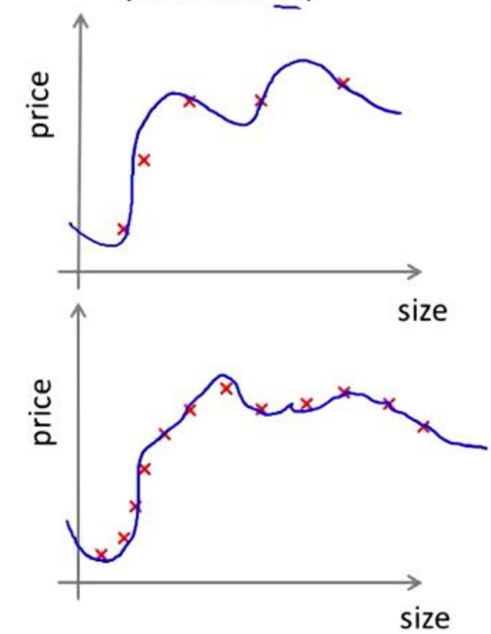
If a learning algorithm is suffering from high bias, getting more training data will not (by itself) help much.

上图所示的学习函数，欠拟合时，增加训练集的数据不会有什么帮助。

High variance

$h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{100} x^{100}$
(and small λ)





If a learning algorithm is suffering from high variance, getting more training data is likely to help.

如果学习曲线如上图所示，过拟合时，增加更多的训练集可能会有帮助。
cost function 正则化的时候，去掉theta0，就是要去掉和X0对应的。

Neural network:

$$h_\Theta(x) \in \mathbb{R}^K \quad (h_\Theta(x))_i = i^{th} \text{ output}$$

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

正则化加上的就是所有theta的和*（lambda/2m）。

```
y_temp = zeros(m,size(y,2));
for i=1:m
    l = y(i);
    y_temp(i,l) = 1;
end
J = -sum(sum(y_temp.*log(s1)+(1-y_temp).*log(1-s1))/m)+lambda*sum(sum(Theta2(:,2:size(Theta2,2)).^2))/(2*m)...
    +lambda*sum(sum(Theta1(:,2:size(Theta1,2)).^2))/(2*m);
```

神经网络的代价函数，多分类问题，所以要把实际分类值做处理（用for循环），处理成下图类型，才能和假设函数求出来的预测值进行计算。

$$y = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \ldots \quad \text{or} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}.$$

反向传播求解每层的delte（实际误差）想法：

2. For each output unit k in layer 3 (the output layer), set

$$\delta_k^{(3)} = (a_k^{(3)} - y_k),$$

where $y_k \in \{0,1\}$ indicates whether the current training example belongs to class k ($y_k = 1$), or if it belongs to a different class ($y_k = 0$). You may find logical arrays helpful for this task (explained in the previous programming exercise).

3. For the hidden layer $l = 2$, set

$$\delta^{(2)} = \left(\Theta^{(2)}\right)^T \delta^{(3)} .* g'(z^{(2)})$$

4. Accumulate the gradient from this example using the following formula. Note that you should skip or remove $\delta_0^{(2)}$. In Octave/MATLAB, removing $\delta_0^{(2)}$ corresponds to `delta_2 = delta_2(2:end)`.

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

5. Obtain the (unregularized) gradient for the neural network cost function by dividing the accumulated gradients by $\frac{1}{m}$:

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)} = \frac{1}{m}\Delta_{ij}^{(l)}$$

实际代码：

```
delte2 = s1 - y_temp;
Theta2_grad = delte2'*s/m+lambda*[zeros(size(Theta2,1),1) Theta2(:,2:size(Theta2,2))]/m;
temp = X*Theta1';
delte1 = delte2*Theta2(:,2:size(Theta2,2)).*sigmoidGradient(temp);
Theta1_grad = delte1'*X/m+lambda*[zeros(size(Theta1,1),1) Theta1(:,2:size(Theta1,2))]/m;
```

所有的数据在一起用矩阵计算，就是先用输出层的预测值减去处理过的实际值，求出delte(L)，然后用delte(l)与delte(l+1)之间的关系公式求出delte(l)，以此类推，求出所有隐藏层的delte。求出所有的deltel以后，用梯度求解公式求对应的梯度。求解梯度的时候要正则化。

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)} = \frac{1}{m}\Delta_{ij}^{(l)} \qquad \text{for } j = 0$$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)} = \frac{1}{m}\Delta_{ij}^{(l)} + \frac{\lambda}{m}\Theta_{ij}^{(l)} \qquad \text{for } j \geq 1$$

正则化的时候还是不算theta(0).