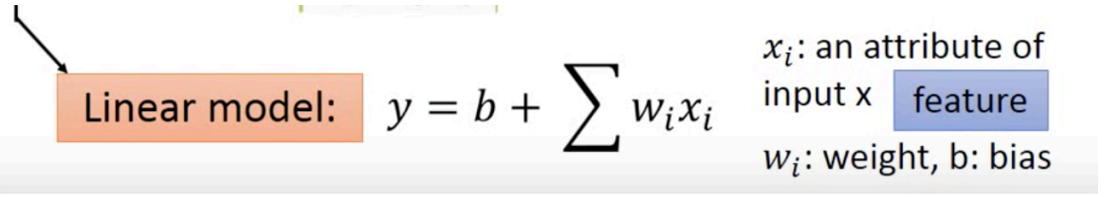
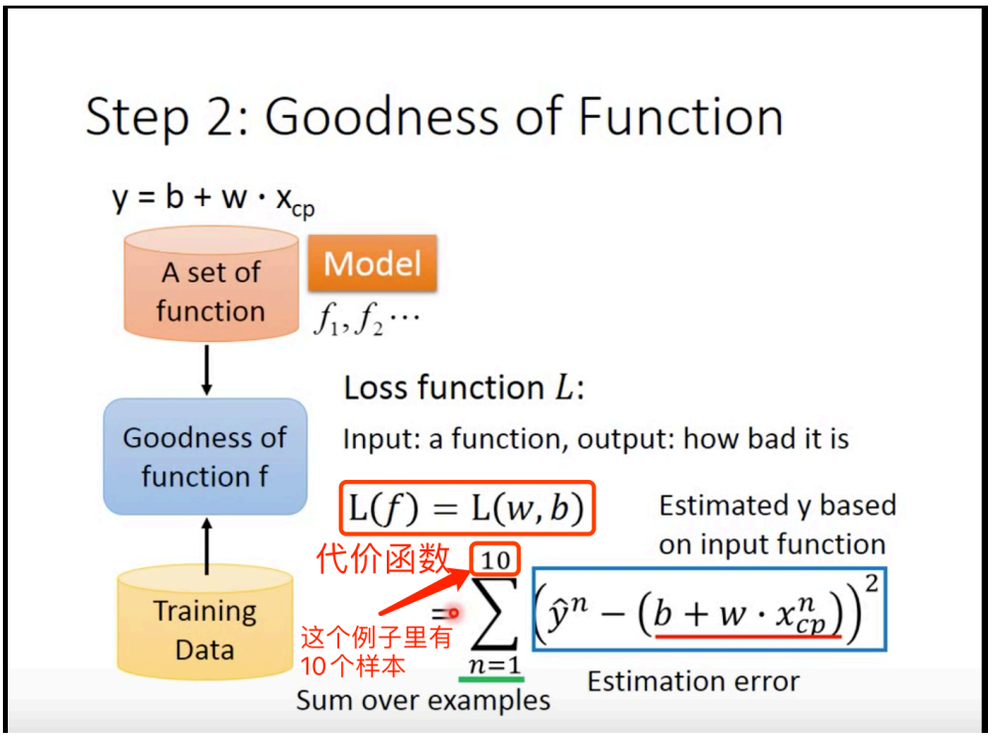
线性模型



xi：样本多维属性， wi：样本该维度的系数，b：偏移， 其实就是求wi

代价函数



把每一组预测系数wi和偏移b带入代价函数，算误差 （这里取平方，是一种简单的误差算法）

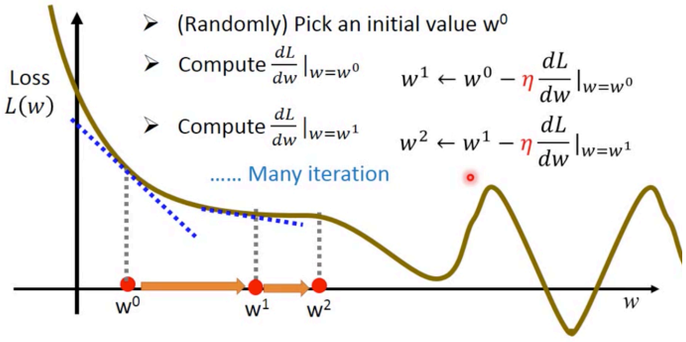
GradientDescent 梯度下降

最有效率的穷举wi的方法

Step 3: Gradient Descent

$$w^* = arg\min_w L(w)$$

- Consider loss function $L(w)$ with one parameter w:
  - (Randomly) Pick an initial value w^0
  - Compute $\frac{dL}{dw}|_{w=w^0}$ $\quad w^1 \leftarrow w^0 - \eta \frac{dL}{dw}|_{w=w^0}$
  - Compute $\frac{dL}{dw}|_{w=w^1}$ $\quad w^2 \leftarrow w^1 - \eta \frac{dL}{dw}|_{w=w^1}$
  - ..... Many iteration



先说1个参数w的例子，(不是上面的例子，上面的例子还有b)（代价函数图像L与w的关系）

现在其实就是找图像最低点

1.初始w0位置，求该点导数(多参数时为，偏导)

$\frac{\partial L}{\partial w}$ ←

2.偏导>0说明左侧下降，右侧上升

3.w0该往左偏移（使代价最小）

4.增量为~~导数dL/dw值 * 学习率η（自定义）~~

$w \mathrel{+}= -\left(\frac{\partial L}{\partial w} \times \eta\right)$ → 0

Step 3: Gradient Descent

$$\nabla L = \begin{bmatrix} \frac{\partial L}{\partial w} \\ \frac{\partial L}{\partial b} \end{bmatrix}$$ gradient

- How about two parameters? $w^*, b^* = arg\min_{w,b} L(w,b)$
  - (Randomly) Pick an initial value w^0, b^0
  - Compute $\frac{\partial L}{\partial w}|_{w=w^0,b=b^0}, \frac{\partial L}{\partial b}|_{w=w^0,b=b^0}$

$$w^1 \leftarrow w^0 - \eta \frac{\partial L}{\partial w}|_{w=w^0,b=b^0} \qquad b^1 \leftarrow b^0 - \eta \frac{\partial L}{\partial b}|_{w=w^0,b=b^0}$$

  - Compute $\frac{\partial L}{\partial w}|_{w=w^1,b=b^1}, \frac{\partial L}{\partial b}|_{w=w^1,b=b^1}$

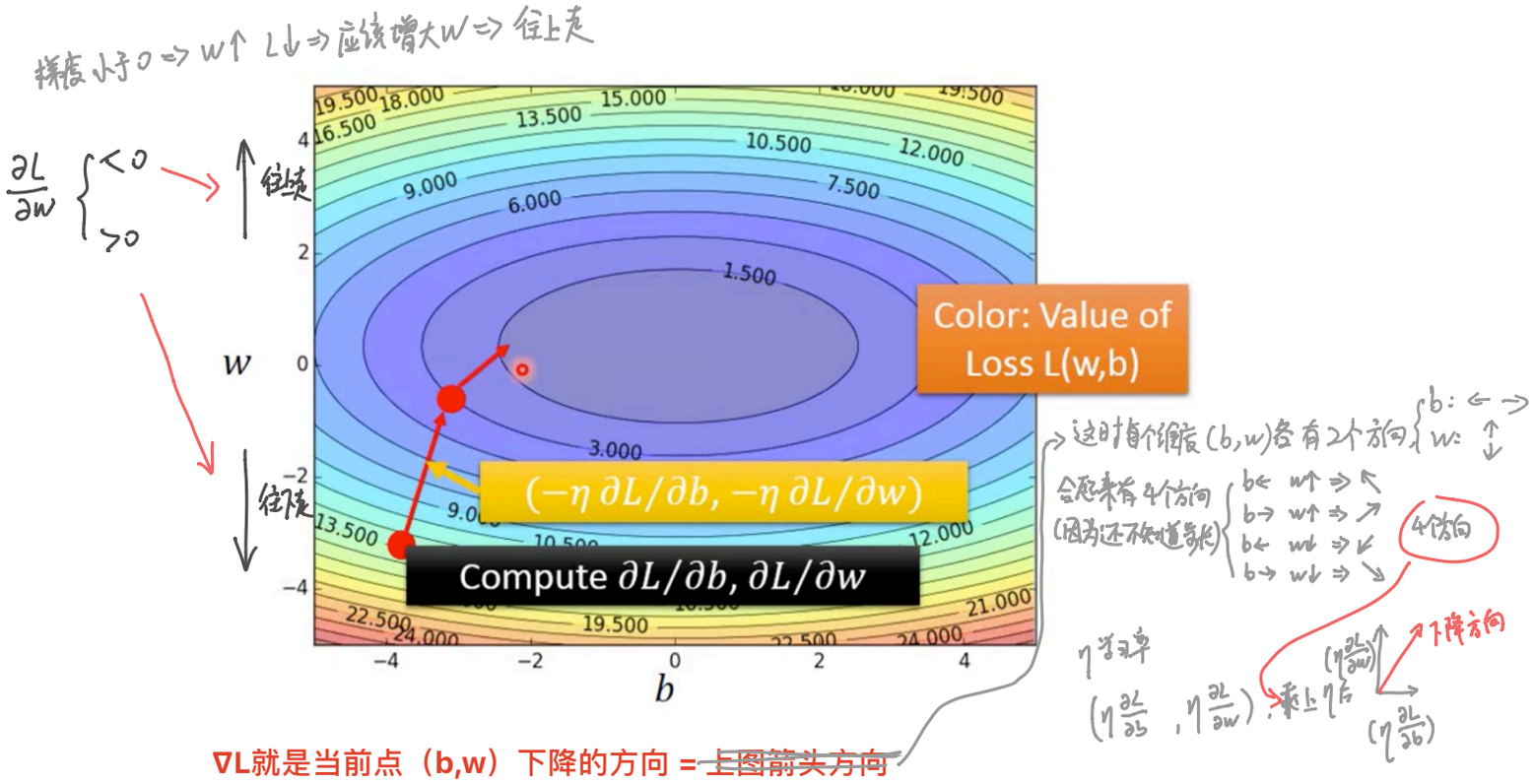
$$w^2 \leftarrow w^1 - \eta \frac{\partial L}{\partial w}|_{w=w^0,b=b^0} \qquad b^2 \leftarrow b^1 - \eta \frac{\partial L}{\partial b}|_{w=w^0,b=b^0}$$

2个参数w和b的例子（与上个例子类似）

1.初始w0与b0，各自求偏导

2.根据各自偏导，各自偏移

∇L就是gradient，他是一个向量，表示各个维度上的斜率（梯度）



∇L就是当前点（b,w）下降的方向 = ~~上图箭头方向~~

η*偏微分就是分量上的步长 = 上图箭头分别在w，b上的分量步长

偏微分计算

$$L(w,b) = \sum_{n=1}^{10} \left(\hat{y}^n - \left(b + w \cdot x_{cp}^n\right)\right)^2$$

$$\frac{\partial L}{\partial w} =? \sum_{n=1}^{10} 2\left(\hat{y}^n - \left(b + w \cdot x_{cp}^n\right)\right)\left(-x_{cp}^n\right)$$

$$\frac{\partial L}{\partial b} =? \sum_{n=1}^{10} 2\left(\hat{y}^n - \left(b + w \cdot x_{cp}^n\right)\right)(-1)$$

发现一组w和b（直线）不能非常好的拟合样本以及预测

均方误差代价函数

预测函数

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

方便后续计算梯度时被2约去

np.dot(x,w)

$$\nabla J(\Theta) = \left\langle \frac{\delta J}{\delta \Theta_0}, \frac{\delta J}{\delta \Theta_1} \right\rangle$$

$$\frac{\delta J}{\delta \Theta_0} = \frac{1}{m} \sum_{i=1}^{m} \left(h_\Theta\left(x^{(i)}\right) - y^{(i)}\right)$$

$$\frac{\delta J}{\delta \Theta_1} = \frac{1}{m} \sum_{i=1}^{m} \left(h_\Theta\left(x^{(i)}\right) - y^{(i)}\right) x_1^{(i)}$$

How's the results? - Generalization

What we really care about is the error on new data (testing data)

$y = b + w \cdot x_{cp}$

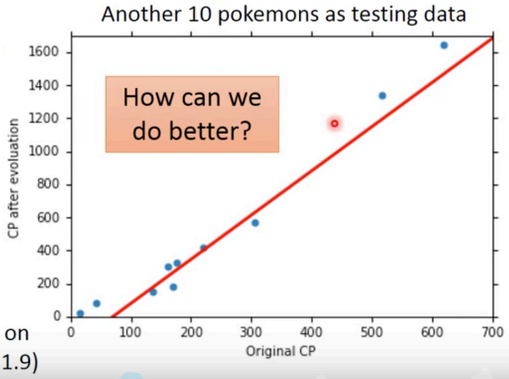
b = -188.4

w = 2.7

Average Error on Testing Data

$= \sum_{n=1}^{10} e^n = 35.0$

> Average Error on Training Data (31.9)



引入二次项来更好的拟合，预测上好了很多18.4

Selecting another Model

$y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2$

Best Function

b = -10.3

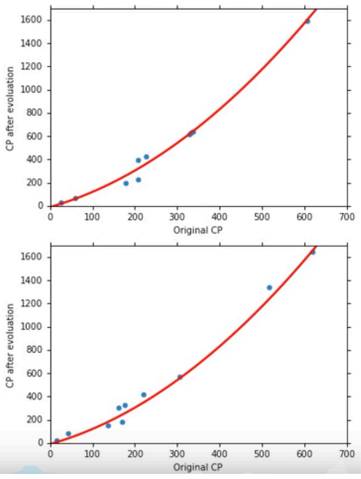
w_1 = 1.0, w_2 = 2.7 x 10^{-3}

Average Error = 15.4

Testing:

Average Error = 18.4

Better! Could it be even better?



引入三次项，预测上稍微又好了一点18.1

Selecting another Model

$y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3$

Best Function

b = 6.4, w_1 = 0.66

w_2 = 4.3 x 10^{-3}

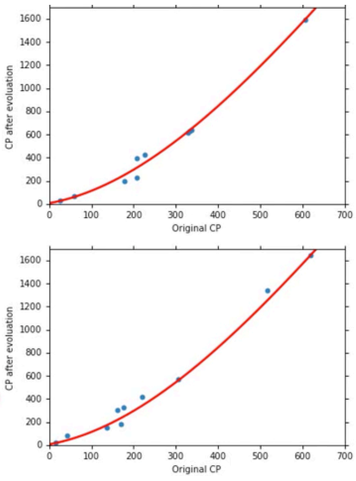
w_3 = -1.8 x 10^{-6}

Average Error = 15.3

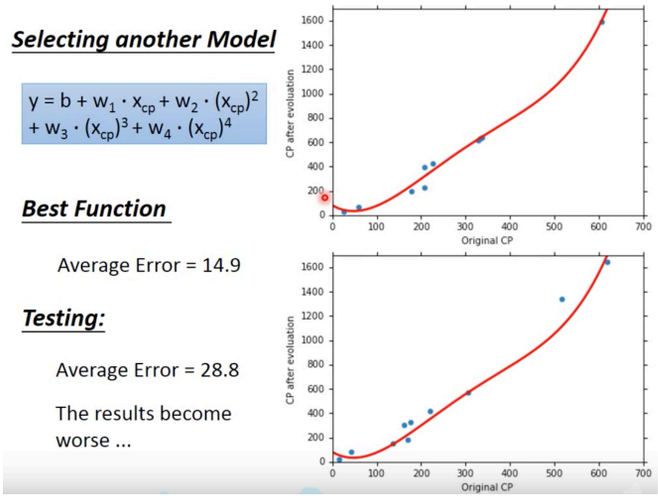
Testing:

Average Error = 18.1

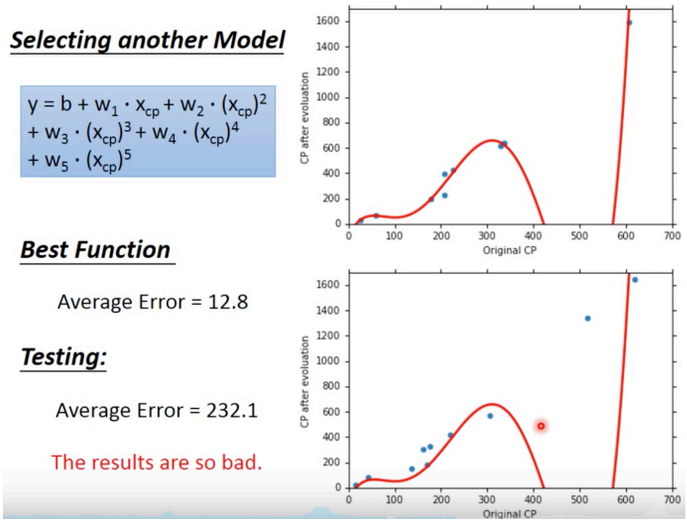
Slightly better. How about more complex model?



引入四次项，样本拟合不断变好，可是预测变差了28.8



引入五次项，炸了



过拟合（越复杂的模型不一定能很好的预测）

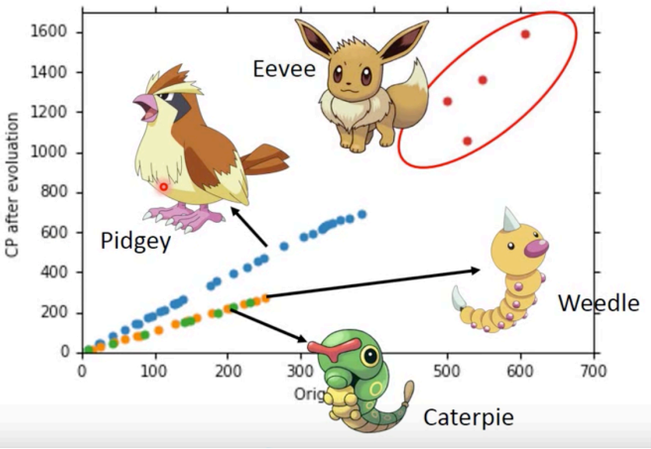
Model Selection

	Training	Testing
1	31.9	35.0
2	15.4	18.4
3	15.3	18.1
4	14.9	28.2
5	12.8	232.1

A more complex model does not always lead to better performance on ***testing data***.

修改模型，寻找隐藏因素

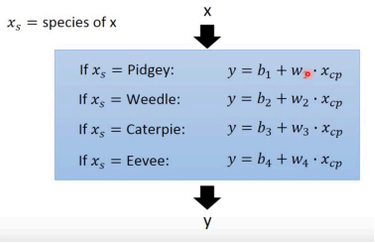
What are the hidden factors?



这个例子中，隐藏因素为物种

用if else 分物种分类处理

Back to step 1:
Redesign the Model



转成数学表达式，就是让那一项系数变1，不需要的项系数变0

Back to step 1:
Redesign the Model

$$y = b + \sum w_i x_i$$

Linear model?

$$y = b_1 \cdot 1$$
$$+ w_1 \cdot 1$$
$$+ b_2 \cdot 0$$
$$+ w_2 \cdot 0$$
$$+ b_3 \cdot 0$$
$$+ w_3 \cdot 0$$
$$+ b_4 \cdot \delta(x_s = \text{Eevee})$$
$$+ w_4 \cdot \delta(x_s = \text{Eevee}) x_{cp}$$

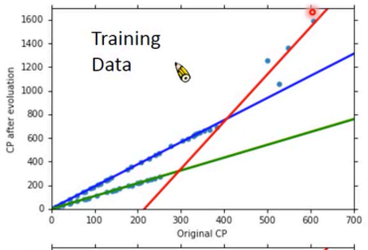
$\delta(x_s = \text{Pidgey})$

$$\begin{cases} =1 & \text{If } x_s = \text{Pidgey} \\ =0 & \text{otherwise} \end{cases}$$

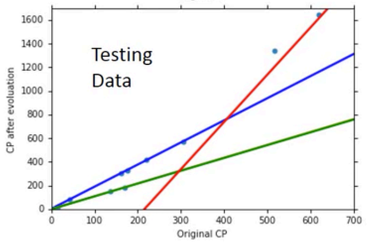
If x_s = Pidgey

每种物种一根预测线

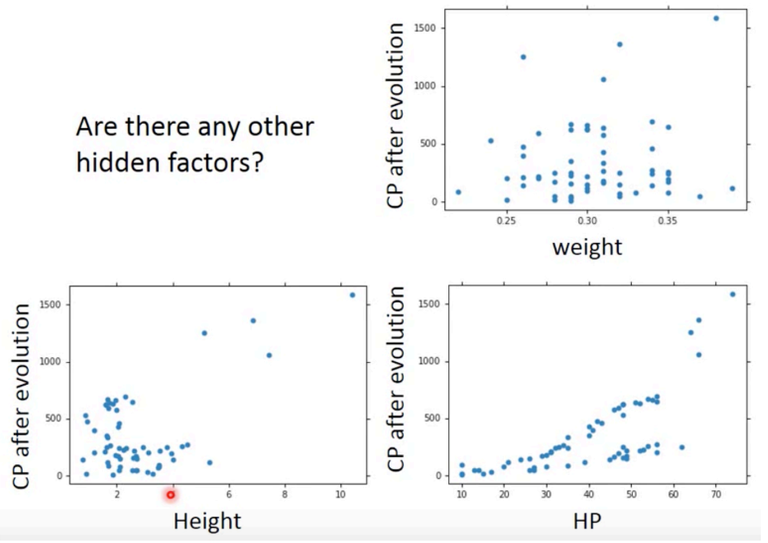
Average error = 3.8



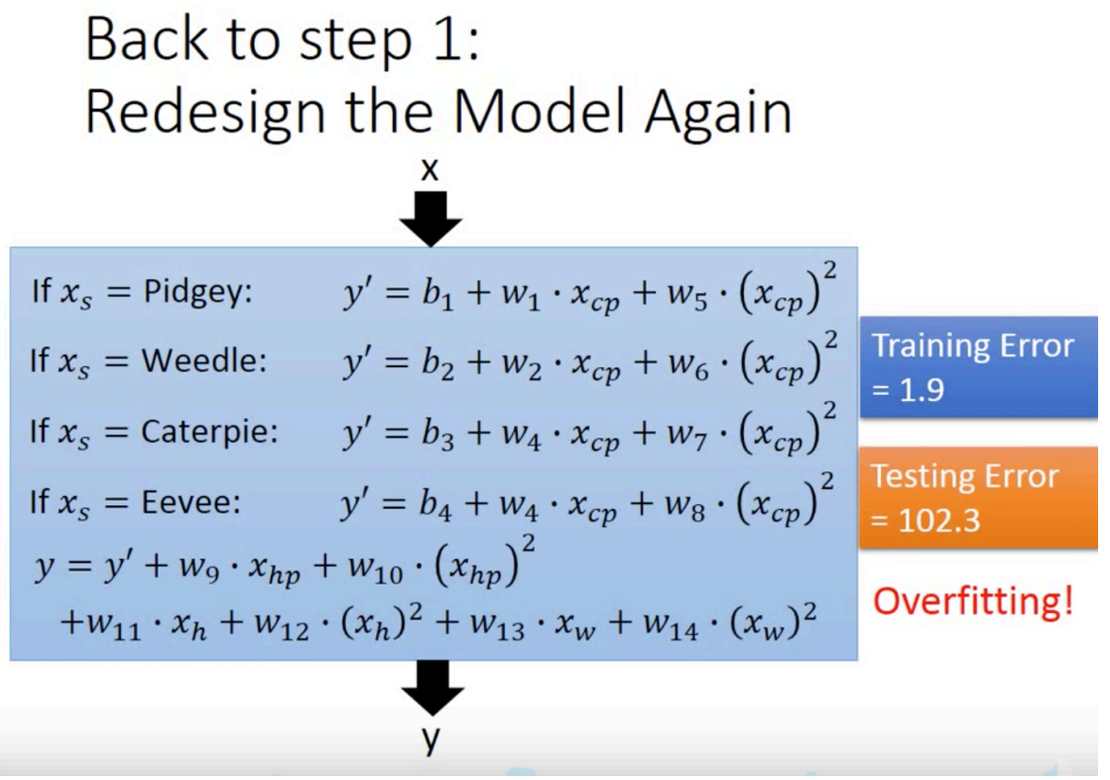
Average error = 14.3



到底预测值与 哪个因素有关?



寻找一个足够复杂的模型，包括所有因素



使代价函数更加平滑

Back to step 2: Regularization

$$y = b + \sum w_i x_i$$

$$L = \sum_n \left(\hat{y}^n - \left(b + \sum w_i x_i \right) \right)^2 + \lambda \sum (w_i)^2$$

The functions with smaller w_i are better

- Why smooth functions are preferred?

$$y = b + \sum w_i x_i$$

$+w_i \Delta x_i$ $+\Delta x_i$

- If some noises corrupt input x_i when testing

A smoother function has less influence.

可以看到，不加入红框项时，此时y随Δx变化=wi*Δx，wi越大=>y变化越大

平滑后，训练样本误差变大，预测误差变小

Regularization

λ	Training	Testing
0	1.9	102.3
1	2.3	68.7
10	3.5	25.7
100	4.1	11.1
1000	5.6	12.8
10000	6.3	18.7
100000	8.5	26.8

- Training error: largerλ, considering the training error less