Regression

问题：预测一只宝可梦进化后的 CP 值

确定 Scenario　Task 和 Model

Scenario: input 是进化前宝可梦的各种属性，output 是进化后的宝可梦 CP 值，由于已知目标变量，故 Scenario 是监督学习

Task:我们希望寻找一个 function 预测宝可梦进化后的 CP 值,即目标变量是数值型,因此使用的 Task 是 Regression 类型

Model：Linear model

设定具体参数

X：表示一只宝可梦，用下标表示该宝可梦的属性

X_{cp}：该宝可梦进化前的 CP 值

X_s：该宝可梦是哪一种物种，如皮卡丘

X_{hp}：该宝可梦的生命值

X_w：该宝可梦的重量

X_h：该宝可梦的高度

y：function 的 output，即宝可梦进化后的 CP 值

Regression 的具体过程

Step1：Function set

Linear model: $y = b + w * x_{cp}$

y代表进化后的 CP 值，x_{cp}代表进化前的 CP 值，w与b代表未知参数

扩展 Linear model：$y = b + \sum_{i=1}^{n} w_i x_i$　n代表宝可梦特征个数

Step2：Goodness of Function

参数说明：x_j^i代表第i只宝可梦的第j个属性值

$\hat{y}^i$代表第i只宝可梦输出的真实 CP 值

Loss function 损失函数：衡量一组w与b参数的好坏

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

$L(f)$越小，说明该 function 性能越好，即w与b参数越合理

Step3：Pick the best function

$w^*, b^* = argminL(w,b)$即选择使 Loss 最小的 function

Gradient descent 梯度下降

只要$L(f)$是可微分的，gradient descent 就可以求$L(f)$的最小值，找到较好的 parameters

单参数问题：Loss function 是$L(w)$，前提$L(w)$是可微分的

$w^* = argminL(w)$，寻找最佳的w，实际上就是寻找 L 的切线斜率为零的 global 最小值点

具体过程：

首先随机选择一个初值w^0，计算$\frac{dL}{dw}|_{w=w^0}, w^1 = w^0 - \eta \frac{dL}{dw}|_{w=w^0}$，计算$\frac{dL}{dw}|_{w=w^1}$，$w^2 = w^1 - \eta \frac{dL}{dw}|_{w=w^1}$

$w^{i+1} = w^i - \eta \frac{dL}{dw}|_{w=w^i}$

两个参数的问题

$w^*, b^* = argmin L(w,b) = argmin \sum_{n=1}^{10} \left(\hat{y}^n - \left(b + w * x_{cp}^n\right)\right)^2$，寻找最佳的$w, b$

首先，随机选取两个初始值，w^0和b^0

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

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

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

局限性：gradient descent 可能找到的是 local minimal,但对于线性 Regression，它的 loss function 是一个凸函数，故使用 gradient descent 求得的极小值就是 global minimal

回到 pokemon 的问题上来

求具体的 L 对 w 和 b 的偏微分

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

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

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

How can we do better?

重新设计 model，考虑$\left(x_{cp}\right)^2$，$\left(x_{cp}\right)^3$，$\left(x_{cp}\right)^4$，$\left(x_{cp}\right)^5$的 model

Model 之间的比较

Model 在 training data 上的表现，随着$\left(x_{cp}\right)^i$的高次项的增加，对应的 average error 会不断下降，但是在 testing data 上，model 复杂到一定程度后，error 非但不会下降，反而会上升，即 overfitting（过拟合）现象

讨论其他参数

物种x_s的影响

重新设计 model：

$$if\ x_s = Pidgey:\ y = b_1 + w_1 \cdot x_{cp}$$
$$if\ x_s = Weedle:\ y = b_2 + w_2 \cdot x_{cp}$$
$$if\ x_s = Caterpie:\ y = b_3 + w_3 \cdot x_{cp}$$
$$if\ x_s = Eevee:\ y = b_4 + w_4 \cdot x_{cp}$$

考虑x_{hp}，x_h，x_w的影响

$$f\ x_s = Pidgey:\ y' = b_1 + w_1 \cdot x_{cp} + w_5 \cdot \left(x_{cp}\right)^2$$
$$if\ x_s = Weedle:\ y' = b_2 + w_2 \cdot x_{cp} + w_6 \cdot \left(x_{cp}\right)^2$$
$$if\ x_s = Caterpie:\ y' = b_3 + w_3 \cdot x_{cp} + w_7 \cdot \left(x_{cp}\right)^2$$
$$if\ x_s = Eevee:\ y' = b_4 + w_4 \cdot x_{cp} + w_8 \cdot \left(x_{cp}\right)^2$$
$$y = y' + w_9 \cdot x_{hp} + w_{10} \cdot \left(x_{hp}\right)^2 + w_{11} \cdot x_h + w_{12} \cdot (x_h)^2 + w_{13} \cdot x_w + w_{14} \cdot (x_w)^2$$

算出的 training data 上的 error=1.9，但是 testing data 上的 error=102.3！这么复杂 model 极可能出现 overfitting(即使用高次项变量使得 training data 拟合的好，但在 testing data 表现得很糟糕)，

Regularization 解决 overfitting 的问题

Regularization 可以使曲线变得更加 smooth，从而使得输出对于输入敏感度下降，training data 上的 error 变大，

但在 testing data 上的 error 变小

在无法确定真实数据分布的情况下，我们尽可能去改变 loss function 的评价标准

我们的 model 的表达式尽可能的复杂，包含尽可能多的参数和尽可能多的高次非线性项

但是我们的 loss function 又有能力去控制这条曲线的参数和形状，使之不会出现 overfitting 过拟合现象

在真实数据满足高次非线性曲线分布的时候，loss function 控制训练出来的高次项系数比较大，使得到的曲线比较弯折起伏

在真实数据满足低次线性分布的时候，loss function 控制训练出来的高次项系数比较小甚至为 0，使得到的曲线接近 linear 分布

使用 L1，L2 正则化，可以实现这一评价标准

L1 正则化即加上$\lambda\sum|w_j|$这一项，loss function 为$L=\sum_{i=1}^{n}\left(\hat{y}^i-y^i\right)^2+\lambda\sum|w_j|$

L2 正则化即加上$\lambda\sum\left(w_j\right)^2$这一项，loss function 为$L=\sum_{i=1}^{n}\left(\hat{y}^i-y^i\right)^2+\lambda\sum\left(w_j\right)^2$

相对来说，L2 要更稳定一些，L1 的结果不那么稳定

对于宝可梦 CP 值预测问题，选取 L2 正则化

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

λ值越大代表考虑 smooth 的那个 regularization 那一项的影响力越大，我们找到的 function 就越平滑

我们喜欢比较平滑的 function，因为它对 noise 不那么 sensitive,但是我们又不喜欢太平滑的 function，因为它就失去了对 data 拟合的能力；我们可以调整λ来调节 function 的平滑度

conclusion 总结

- 根据已有的 data 特点(labeled data)，确定使用 supervised learning
- 根据 output 的特点（数值型），确定使用 Regression 回归
- 考虑包括进化前 CP 值、物种、生命值、重量、身高等属性以及高次项的影响，我们的 model 可以采用 input 的一次项和二次项之和的形式，如：

$$f\ x_s=Pidgey:\ y'=b_1+w_1\cdot x_{cp}+w_5\cdot\left(x_{cp}\right)^2$$
$$if\ x_s=Weedle:\ y'=b_2+w_2\cdot x_{cp}+w_6\cdot\left(x_{cp}\right)^2$$
$$if\ x_s=Caterpie:\ y'=b_3+w_3\cdot x_{cp}+w_7\cdot\left(x_{cp}\right)^2$$
$$if\ x_s=Eevee:\ y'=b_4+w_4\cdot x_{cp}+w_8\cdot\left(x_{cp}\right)^2$$
$$y=y'+w_9\cdot x_{hp}+w_{10}\cdot\left(x_{hp}\right)^2+w_{11}\cdot x_h+w_{12}\cdot(x_h)^2+w_{13}\cdot x_w+w_{14}\cdot(x_w)^2$$

而为了保证 function 的平滑性，loss function 应使用 regularization，即

$L(w,b)$=$\sum_{n=1}^{10}\left(\hat{y}^n-\left(b+w*x_{cp}^n\right)\right)^2+\lambda\sum\left(w_j\right)^2$，注意 bias----对 function 平滑性无影响因此不额外计入 loss function

- 利用 gradient descent 对 regularization 版本的 loss function 进行梯度下降迭代处理，每次迭代都减去 L 对改参数的微分与 learning rate 之积

设所有参数组成一个向量：$\left[w_0,\ w_1,\ w_2,\ w_3,\ w_4,\ w_5,\ \cdots\cdots,\ b\right]^T$，那么每次梯度下降的表达式如下：

梯度：$\nabla L = \begin{bmatrix} \frac{\partial L}{\partial w_0} \\ \frac{\partial L}{\partial w_1} \\ \cdots \\ \frac{\partial L}{\partial w_j} \\ \frac{\partial L}{\partial b} \end{bmatrix}$ gradient descent：$\begin{bmatrix} w_0' \\ w_1' \\ \cdots \\ w_j' \\ b' \end{bmatrix}_{L=L'} = \begin{bmatrix} w_0 \\ w_1 \\ \cdots \\ w_j \\ b \end{bmatrix}_{L=L_0} - \eta \begin{bmatrix} \frac{\partial L}{\partial w_0} \\ \frac{\partial L}{\partial w_1} \\ \cdots \\ \frac{\partial L}{\partial w_j} \\ \frac{\partial L}{\partial b} \end{bmatrix}_{L=L_0}$

当梯度稳定不变时，即∇L为 0 时，gradient descent 便停止，此时如果采用的 model 时 linear 的，那么 vector 必然落于 global minimal 处，如果采用的 model 是 Non-linear 的，vector 可能会 local minimal 处

- 这里λ的最佳数值是需要通过我们不断调整来获取的，因此令λ等于 0,10,100，….不断使用 gradient descent 或其他算法得到最佳的 parameters：$\left[w_0，w_1，w_2，w_3，w_4，w_5，……，b\right]^T$，并计算出这组参数确定的 function----f^*对 training data 和 testing data 上的 error 值，直到找到那个使 testing data 的 error 最小的 λ。
- 引入评价标准f^*的 error 机制，令 error=$\frac{1}{n}\sum_{i=1}^{n}|\hat{y}^i - y^i|$，分别计算该$f^*$对 training data 和 testing data（more important）的 error(f^*)大小

参数λ调整过程

先设定λ的值

确定 loss function

找到使 loss 最小的$\left[w_0，w_1，w_2，w_3，w_4，w_5，……，b\right]^T$

确定 function，计算 error

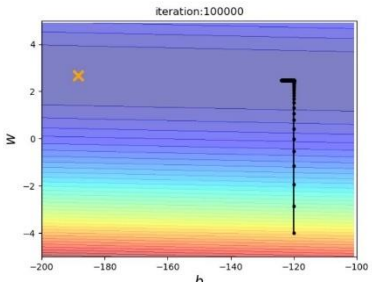
重新设定新的λ值，重复以上步骤

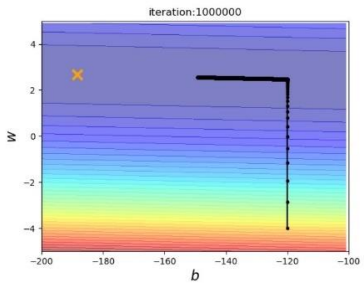
使 testing data 上的 error 最小的λ所对应的$\left[w_0，w_1，w_2，w_3，w_4，w_5，……，b\right]^T$，为所求 function

Gradient descent demo

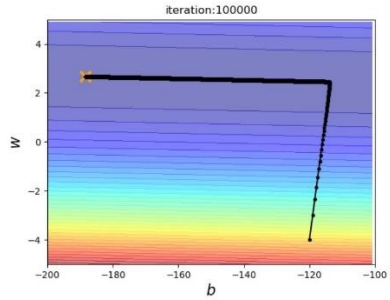
代码见 Adagrad_demo 中,此处只展示结果

learning rate=0.00000001





使用 Adaptive gradient 方法：learning rate=1



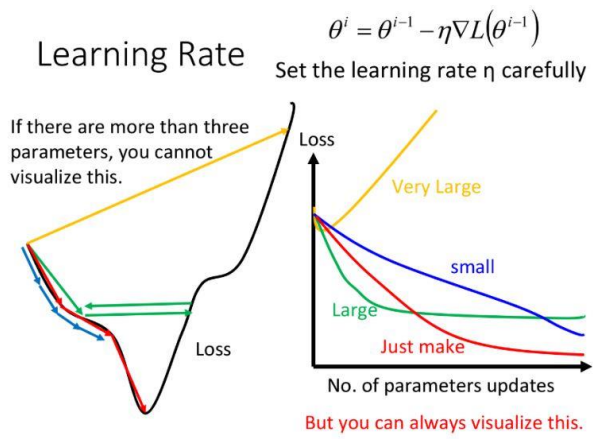
Learning Rate 存在的问题

gradient descent 过程中，影响结果的一个很关键的因素就是 learning rate 的大小：

- 如果 learning rate 合适，就可以如下图红色线段一样顺利地找到 loss 的最小值
- 如果 learning rate 太小，则 loss 减小的太慢，如下图蓝色线段所示
- 如果 learning rate 太大，如下图绿色线段所示，它的步伐太大了，它永远没有办法走到特别低的地方，可

能永远在这个“山谷”的口上振荡而无法走下去

- 如果 learning rate 非常大，如下图黄色线段所示，造成 update 参数以后，loss 反而越来越大



Adaptive learning rate

显然这样手动地调整 learning rate 很耗时，这时需要有一些自动调整 learning rates 的方法

最基本、最简单的原则是：learning rate 通常是随参数的 update 越来越小

Adagrad

Adagrad 就是将不同参数的 learning rate 分开考虑的一种算法(adagrad 算法 update 到后面速度会越来越慢，当然这只是 adaptive 算法中最简单的一种)

$w^{t+1} \leftarrow w^t - \eta^t g^t$，$\eta^t = \frac{\eta}{\sqrt{t+1}}$，$g^t = \frac{\partial L(w^t)}{\partial w}$

Adagrad：$w^{t+1} \leftarrow w^t - \frac{\eta^t}{\sigma^t} g^t$，$\sigma^t = \sqrt{\frac{\sum_{i=0}^{t}(g^i)^2}{t+1}}$，$\eta^t = \frac{\eta}{\sqrt{t+1}}$

化解得：$w^{t+1} \leftarrow w^t - \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}} g^t$

$w^1 \leftarrow w^0 - \frac{\eta^0}{\sigma^0} g^0$，$\sigma^0 = \sqrt{(g^0)^2}$，$\eta^0 = \frac{\eta}{\sqrt{0+1}}$，$g^0 = \frac{\partial L(w^0)}{\partial w}$

$w^2 \leftarrow w^1 - \frac{\eta^1}{\sigma^1} g^1$，$\sigma^1 = \sqrt{\frac{[(g^0)^2 + (g^1)^2]}{2}}$，$\eta^1 = \frac{\eta}{\sqrt{1+1}}$，$g^1 = \frac{\partial L(w^1)}{\partial w}$

$w^3 \leftarrow w^2 - \frac{\eta^2}{\sigma^2} g^2$，$\sigma^2 = \sqrt{\frac{[(g^0)^2 + (g^1)^2 + (g^2)^2]}{3}}$，$\eta^2 = \frac{\eta}{\sqrt{2+1}}$，$g^2 = \frac{\partial L(w^2)}{\partial w}$

Adagrad 的 contradiction 解释

Adagrad 要考虑的是，这个 gradient 有多 surprise，即反差有多大，假设 t=4 的时候g^4与前面的 gradient 反差特别大，那么g^t与$\sqrt{\frac{\sum_{i=0}^{t}(g^i)^2}{t+1}}$之间的大小反差就会比较大，它们的商就会把这一反差效果体现出来

g^0	g^1	g^2	g^3	g^4	……
0.001	0.001	0.003	0.002	0.1	……
g^0	g^1	g^2	g^3	g^4	……
10.8	20.9	31.7	12.1	0.1	……

$w^{t+1} \leftarrow w^t - \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}} g^t$造成反差的效果

Stochastic Gradient Descent 随机梯度下降

随机梯度下降的方法可以让训练更快速，传统的 gradient descent 的思路是看完所有样本点之后再构建 loss function，然后去 update 参数，而 stochastic gradient descent 的做法是，随机选择一个样本点就 update 一次，因此它的 loss

function 不是所有样本点的 error 平方和，而是随机样本点的 error 平方和

Randomly pick an example x^n

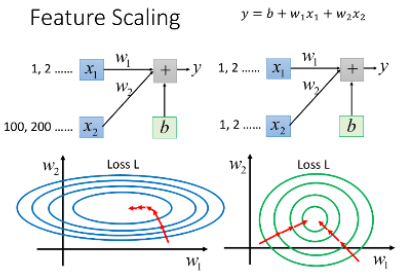
$L^n=\left(\hat{y}^n-\left(b+\sum_{i=1}^{k}w_i x_i^n\right)\right)^2$

$\theta^i=\theta^{i-1}-\eta\frac{\partial L^n(\theta^{i-1})}{\partial\theta}$

Feature Scaling 特征缩放

原理解释

$y=b+w_1x_1+w_2x_2$，假设x_1的值都是很小的，比如 1，2,3….；x_2的值都很大，比如 100,200,300…..此时去画出 loss 的 error ，如果对w_1和w_2都做相同大小的Δw，那么w_1的变化对 y 的影响较小，而w_2的变化对 y 的影响较大；因而w_1对 loss 的偏微分较小，故图像在w_1方向上平滑一些，w_2对 loss 的偏微分较大，故图像在w_2方向上陡峭一些。



归一化

$x_j^i=\frac{x_j^i-min(j)}{\max(j)-min(j)}$，$x_j^i$表示第$i$个样本的第$j$属性值，$\max(j)$表示第$j$属性的所有样本中最大值，$min(j)$表示第$j$属性的所有样本中最小值

标准化

$x_j^i=\frac{x_j^i-mean(j)}{\sigma_j}$，$x_j^i$表示第$i$个样本的第$j$属性值，$mean(j)$表示第$j$属性的所有样本的平均值，$\sigma_j$表示第$j$属性的所有样本的标准差

Where does the error come from? **误差的来源**

bias 偏差与 variance 方差

抽样分布

$\hat{y}$和y^*真值与估测值

$\hat{f}$表示真正的 function，f^*表示对$\hat{f}$的估测值

就好像在打靶，$\hat{f}$是靶的中心点，收集到一些 data 做 training 以后，你会得到一个最佳的 function 即f^*，这个f^*会落在靶上的某一个位置，它跟靶中心有一段距离，这段距离由 bias 和 variance 决定

抽样分布的理论基础

假设独立变量为x(这里的x代表每次独立地从不同的 training data 训练找到的f^*)，那么总体期望$E(x)=u$；总体的方差$Var(x)=\sigma^2$

用样本均值$\bar{x}$估计总体期望u

由于我们只要有限组样本$Sample\ N\ points:\ \{x^1,\ x^2,\ x^3,\ \cdots,\ x^N\}$，故样本均值$\bar{x}=\frac{1}{N}\sum_{i=1}^{N}x^i$；样本均值的期望

$E(\bar{x})=E\left(\frac{1}{N}\sum_{i=1}^{N}x^i\right)=u$；样本均值的方差$Var(\bar{x})=\frac{\sigma^2}{N}$

样本均值$\bar{x}$以总体期望u中心对称分布，可以用来估测总体期望u

用样本方差s^2估测总体方差σ^2

样本均值$\bar{x}=\frac{1}{N}\sum_{i=1}^{N}x^i$，样本方差$s^2=\frac{1}{N-1}\sum_{i=1}^{N}\left(x^i-\bar{x}\right)^2$，样本方差的期望$E(s^2)=\sigma^2$，

样本方差的方差$Var(s^2)=\frac{2\sigma^4}{N-1}$

样本方差s^2以总体方差σ^2中心对称分布，可以用来估测总体方差σ^2

Bias vs Variance

由于简单的 model，bias 较大，variance 较小；复杂的 model，bias 较小，variance 较大；

bias 和 variance 对 error 的影响

$error_{实际} = error_{bias} + error_{variance}$

随着 model 的逐渐复杂

- Bias 逐渐减小，bias 所造成的 error 逐渐下降
- Variance 逐渐增大，variance 所造成的 error 逐渐上升
- 当 bias 和 variance 同时考虑时，实际观测的 error 先减小后增大，因此实际 error 最小值的那个点，即为 bias 和 variance 的 error 之和最小点
- 如果实际 error 主要来自于 variance 很大，说明 overfitting 过拟合；实际 error 来自于 bias 很大，说明 underfitting 欠拟合

实际应用中，必须要明白自己 model 的 error 主要来源

怎么知道 current model 是 bias 大还是 variance 大?

- 如果 model 没有办法 fit training data 的 examples，代表 bias 比较大，这时是 underfitting
- 如果 model 可以 fit training data，在 training data 上得到小的 error，但是在 testing data 上，却得到一个大的 **error，代表 variance 较大，这时是 overfitting**

如果 bias 较大

bias 大代表，现在这个 model 里面可能根本没有包含你的 target，$\hat{f}$可能根本就不在你的 function set 里；

重新设计 model：

- 增加更多的 features 作为 model 的 input 输入变量
- 让 model 变得更加复杂，增加高次项

如果 variance 较大

增加 data

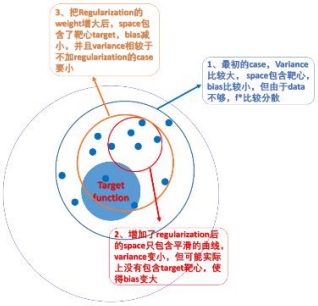
- 增加 data 是一个很有效控制 variance 的方法，假设你 variance 太大的话，collect data 几乎是一个万能的方法，并且它不会影响 bias
- Regularization 正则化

  就是在 loss function 里面再加一个与 model 高次项系数相关的 term，它希望你的 model 里高次项的参数越小越好，以就是说希望寻找到的曲线越平滑越好这个新加的 term 前面可以有一个 weight，代表你希望你的曲线有多平滑；但 regularization 会影响 bias，因为它实际上调整了 function set 的 space 范围，变成他只包含那些比较平滑的曲线，这个缩小的 space 可能没有包含原先在更大 space 内的$\hat{f}$，因此伤害了 bias，因此在做 regularization 的时候，需要调整 regularization 的 weight，在 variance 和 bias 之间取得平衡

使用 regularization 对 bias 和 variance 的影响图例解释

1. 蓝色区域代表最初的情况，此时 model 比较复杂，function set 的 space 范围比较大，包含了 target 靶心，但由于 data 不足，f^*比较分散，variance 比较大
2. 红色区域代表进行 regularization 之后的情况，此时 model 的 function set 范围缩小成只包含平滑曲线，space 减小，variance 也下降，当缩小后的 space 实际上并不包含原先已经包含的 target 靶心，因此该 model 的 bias 变大
3. 橙色区域代表增大 regularization 的额 weight 的 case，增大 weight 实际上增大 function set 的 space，慢慢调整至包含 target 靶心，此时该 model 的 bias 变小，而相较于一开始的 case，由于限定了曲线的平滑度（由 weight 控制平滑度的阙值），该 model 的 variance 也表较小

   实际上通过 regularization 优化 model 的过程就是上述的 1、2、3 步骤，不断地调整 regularization 的 weight，是 model 的 bias 和 variance 达到一个最佳平衡的状态（可以通过 error 来评价状态的好坏，weight 需要慢慢调参）



Model selection

先在 training set 上找出每个 model 最好的 function f^*，然后用 validation set 来选择你的 model

N-fold Cross Validation

例：做 3-fold 的 Validation，意思是把 training set 分成三份，你每一次拿其中一份当做 Validation set，另外两份当 training data，分别在每个情景下都计算一下 3 个 model 的 error，然后计算一下它的 average error；然后你会发现在这三个情境下的 average error；

Conclusion 总结

1. 一般来说，error 是 bias 和 variance 共同作用的结果
2. Model 比较简单和比较复杂的 case：
   - 当 model 比较简单的时候，variance 表较小，bias 较大，此时f^*会比较集中，但是 function set 可能没有包含真实值$\hat{f}$；此时 model 受 bias 影响较大
   - 当 model 比较复杂时，bias 较小，variance 较大，此时 function set 会包含真实值$\hat{f}$，但是f^*会比较分散；此时 model 受 variance 影响较大
3. 区分 bias 较大 or variance 较大的情况
   - 如果样本集的样本点都有大部分不在 model 训练出来的f^*上，说明这个 model 太简单，bias 较大，是欠拟合
   - 如果样本点基本都在 model 训练出来的f^*上，但是 testing data 上测试得到的 error 很大，说明这个 model 太复杂，variance 较大，是过拟合
4. bias 大 or variance 大的处理方法
   - 当 bias 比较大时，需要做的是重新设计 model，包括考虑添加新的 input 变量，考虑给 model 添加高次项，然后对每一个 model 对应的f^*计算出 error，选择 error 值最小的 model(随着 model 变复杂，bias 下降，variance 上升，分别计算 error，取两者的平衡点)
   - 当 variance 比较大时，一个很好的办法是增加 data（可以凭借经验子集自己 generate data），当 data 数量足够时，得到的f^*实际上是比较集中的；如果现实中没有办法 collect 更多的数据，那么就采用 regularization 正则化的方法，以曲线的平滑度为条件控制 function set 的范围，用 weight 控制平滑度阈值，使得最终的 model 即包含$\hat{f}$，variance 又不会太大
5. 如何选择 model

选择 model 的时候，我们拥有的 testing data 与真实的 testing data 之间存在 bias，因此我们要将 training data 分成 training set 和 validation set 两部分，经过 validation 挑选出来的 model 再用全部的 training data 训练一遍参数，最后用 testing data 去测试 error，这样得到的 error 是模拟过 testing bias 的 error，与实际情况下的 error 会比较符合