二、回归学习

1. 回归学习的定义

如前文所述：回归学习是监督学习的一种，输入模型的是一对数据（特征值，标签值），回归学习的标签值，是一些数值。

2、应用场景

- 股票市场的预测

$$f(\text{股票历史信息，公司经营信息}) = \text{明天的股票值}$$

- 自动驾驶汽车

$$f(\text{汽车的各种传感器收集的信息}) = \text{方向盘的角度}$$

- 推荐系统

$$f(\text{使用者}A\text{商品}B) = \text{购买的可能性}$$

3、回归模型的案例分析

- 场景描述：预测宝可梦进化后的CP值

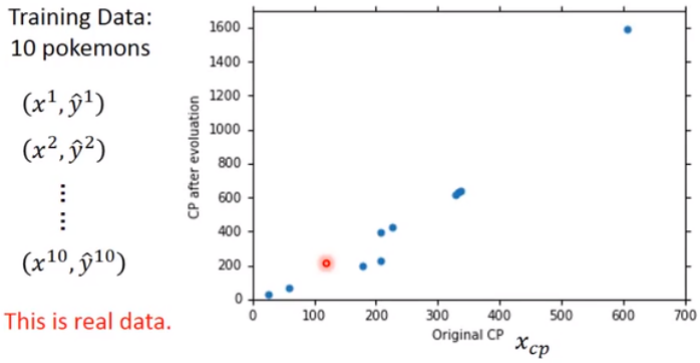
其目标即：
$$
f(x{*初级宝可梦特征值*}) = *y*{进化后宝可梦CP值}

其中的特种值，如x_{CP}当前宝可梦的CP值，x_{HP}当前宝可梦的生命值，x_{Weight}当前宝可梦的重量值，x_{Height}当前宝可梦的高度 *模型构建步骤（如第一课所说） *Step1建立函数池 − Model <imgsrc="C:\Users\18817\Desktop\Model宝可梦.png" alt="建立模型" style="zoom:75这里建立的是一次函数，参数只有b和w,输入的特征值只有x_{cp}。由于b和w有无数中取法，所以这里构成了一个函数池。此外，这样的一个模型，又称为线性模型。所谓线性模型是指可以写成下面形式的模型：

y = b + \sum{w_ix_i}
$$
其中x_i表示输入特征$feature$, w_i表示权重$weight$, b表示$bias$偏置就表示线性模型。

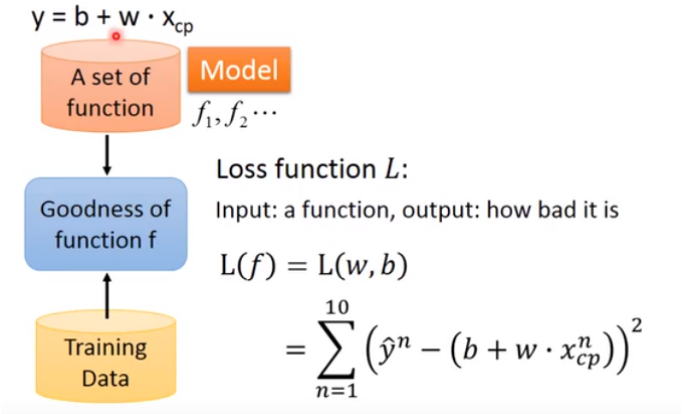
- Step2 定义$Function$ 的好坏

  首先我们要收集一些带有标签的数据

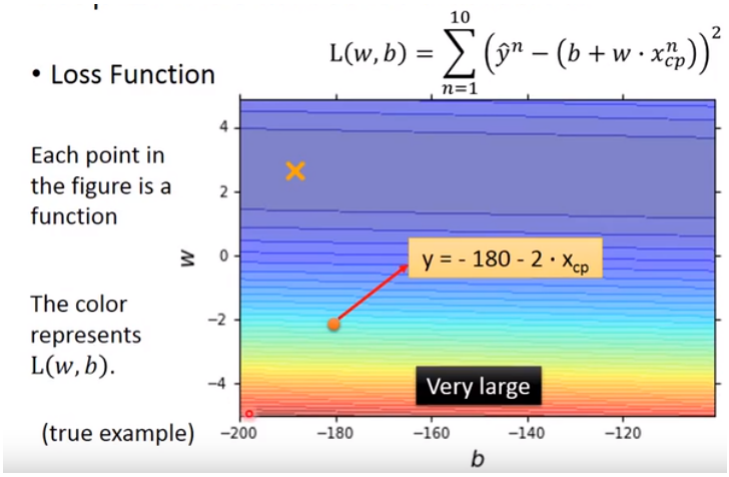


这里就有10对真实数据，其中$(x^i, \hat{y}^i)$表示第i个数据的特征值和标签对。

收集完数据就可以建立对$Function$的评价。

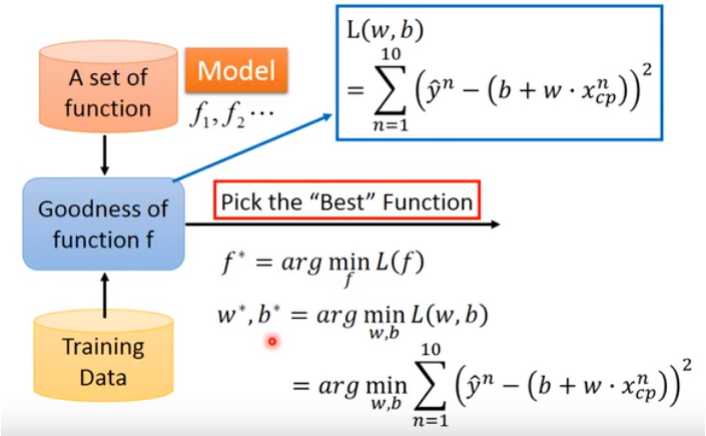


对$Function$的评价使用$LossFuction$,这里使用的是最小二乘法，其含义，就是利用w和b计算特征值，推测出一个进化后的CP值，然后计算这个预测值跟真实cp值之间的差距，差距越小表示函数越好，差距越大表示函数选择的越差。



这张图每一个点就对应一个相应的$Function$的$Loss$值，颜色越红表示$Loss$值越大，相反则越小，那么我们在选择模型的时候就会选择颜色偏蓝的函数。

- Step3 选择最好$Function$

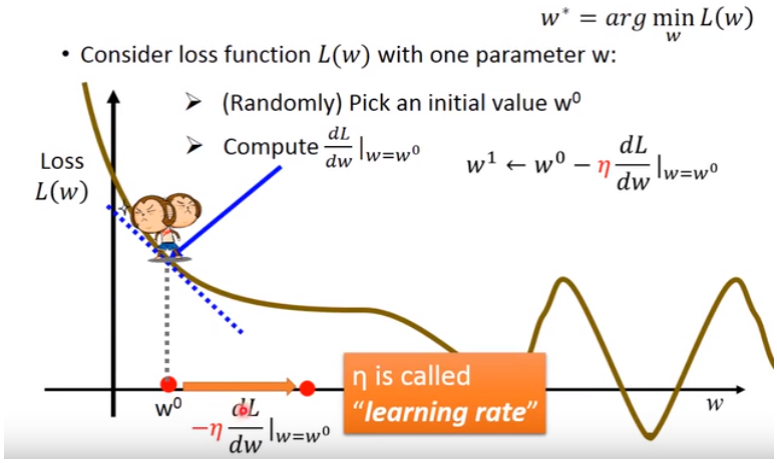


列出所有的w和b的值，选择一个使$Loss$最小的，就是最好的$Function$。

以上就是整个模型构建的过程

4、 梯度下降法

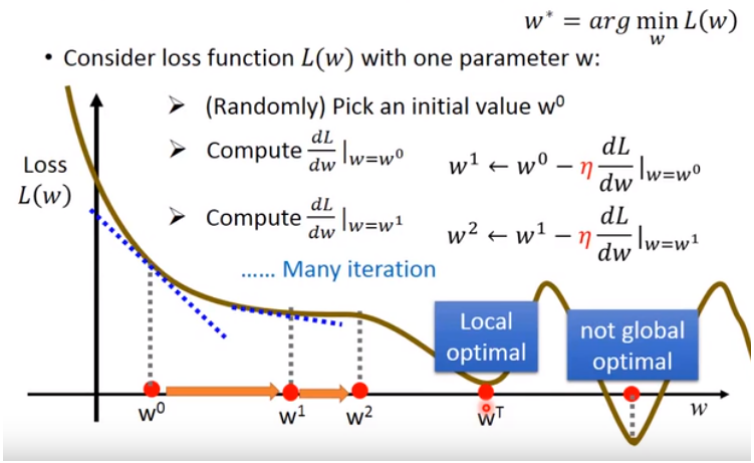
如上所述，如果使用穷举所有$Funtion$并进行比较的方法，由于有无数个所以根本无法完全计算并进行比较，因此提出了梯度下降法使得$Function$每经过一次计算都想最优$Function$靠近。



梯度下降法过程：

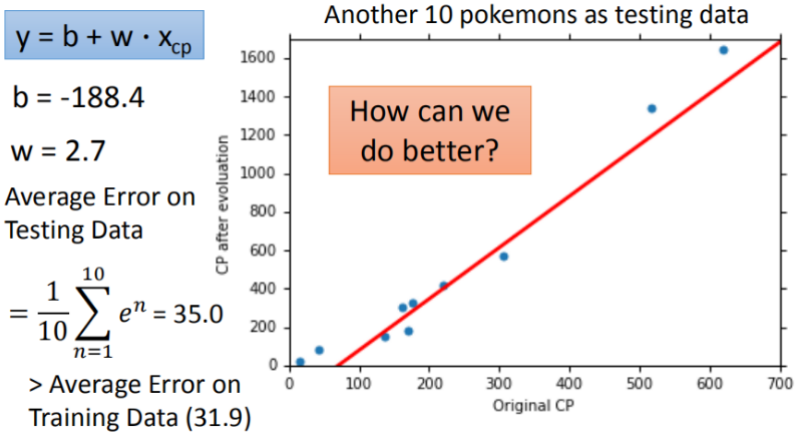
- 随机初始化所有参数
- 得到一个$Loss$之后计算相对于参数的梯度
- 参数向着$Loss$减少的方向移动

如图所示，减去一个梯度值，$Loss$就会减少一部分，在进行梯度下降的过程中设置η值，作为学习率，事实上，可以形象的理解成，移动的步长，也就是更新的幅度的大小，经过多次的迭代计算就会到达一个低点。当然可能遇到局部最优，而不是全局最优，不过这个在线性回归的问题上不会遇到。当遇到多参数时，就使用参数的偏导替代导数。



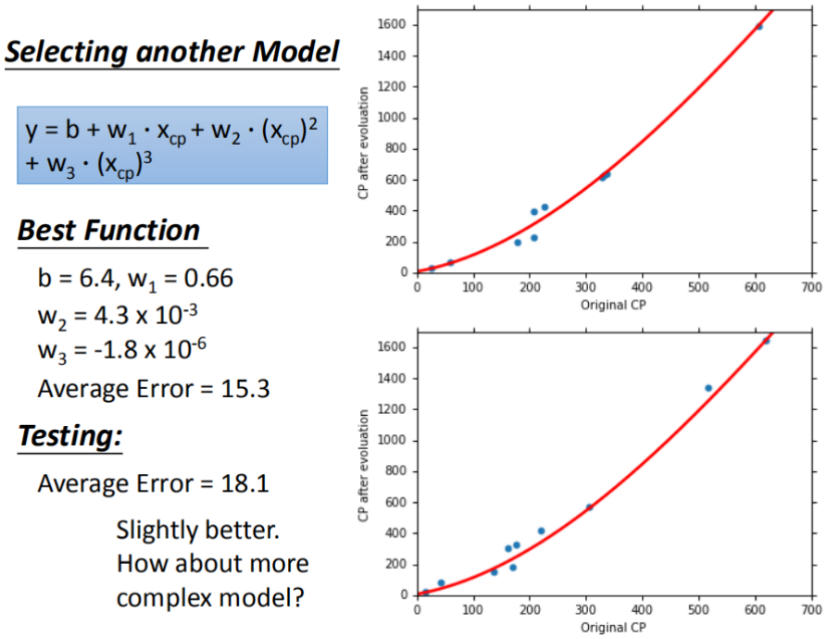
5、过拟合问题

刚刚采用的模型是简单的一次线性模型



但我们评估模型时，不仅评估在训练集上的$loss$,还要评估模型在未见过的数据上的表现及Test data，我们看到在现有的模型在训练数据上的loss是31.9，在测试数据上是35.0

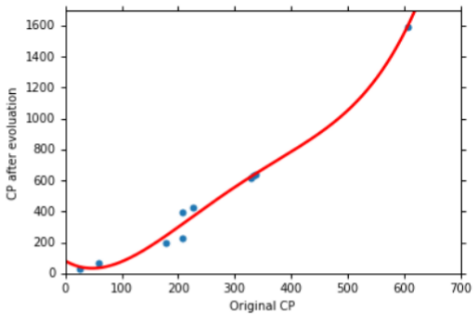
除了一次模型我们可以使用更加复杂的模型



比如这里的二次模型，更加复杂，训练loss降为15.3，测试loss降为了18.1，是更好的模型选择。

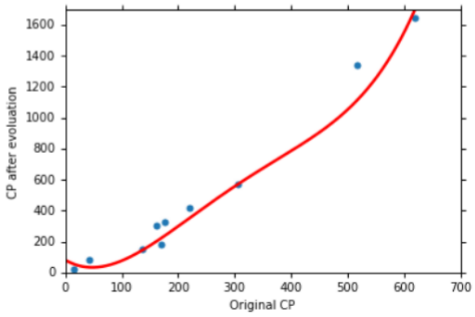
Selecting another Model

$$y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3 + w_4 \cdot (x_{cp})^4$$



Best Function

Average Error = 14.9



Testing:

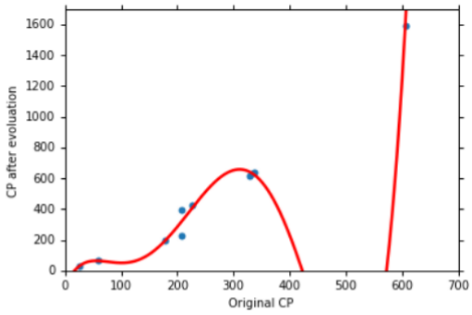
Average Error = 28.8

The results become worse ...

我们还可以使用更加复杂的模型，发现训练loss降的更低，但测试的loss却涨起来了。

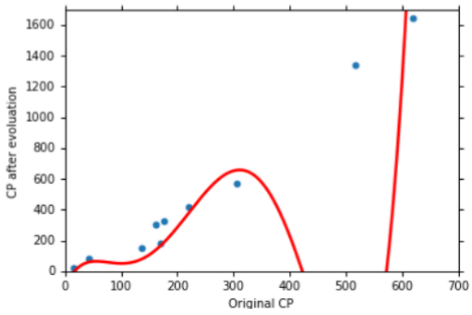
Selecting another Model

$$y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3 + w_4 \cdot (x_{cp})^4 + w_5 \cdot (x_{cp})^5$$



Best Function

Average Error = 12.8



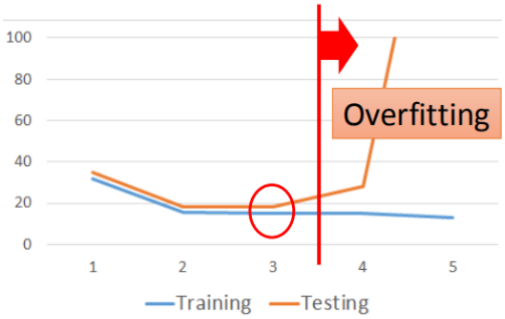
Testing:

Average Error = 232.1

The results are so bad.

当选择的模型更复杂一些时，测试的loss降的更加低了。

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

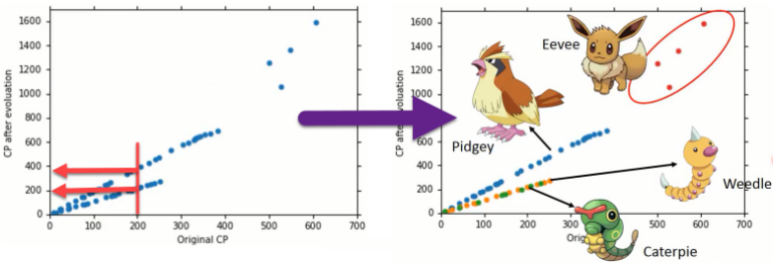
A more complex model does not always lead to better performance on ***testing data***.

This is ***Overfitting***. ➡ Select suitable model

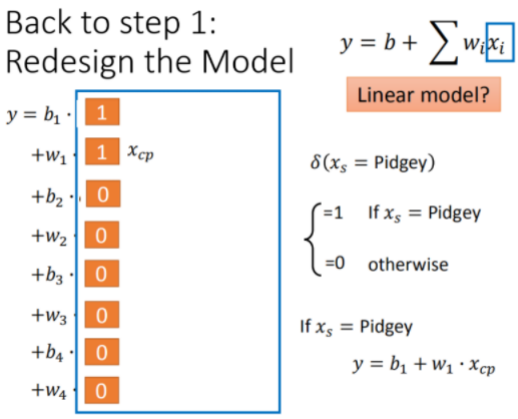
这个就是过拟合问题，当选择的模型越复杂，学习能力就越强，这样他就能充分拟合训练的数据集，但模型拿出去使用时，test数据集其实和训练数据集并不完全一样，导致在新的数据集上表现的很差，简单点说就是针对某种问题学过头了，遇到新的问题，转不过弯来了。

6、优化模型

- 刚刚的模型中，只使用了当前的cp值一个特征，我们发现，宝可梦进化后的cp值事实上和宝可梦的种类关系也很大。

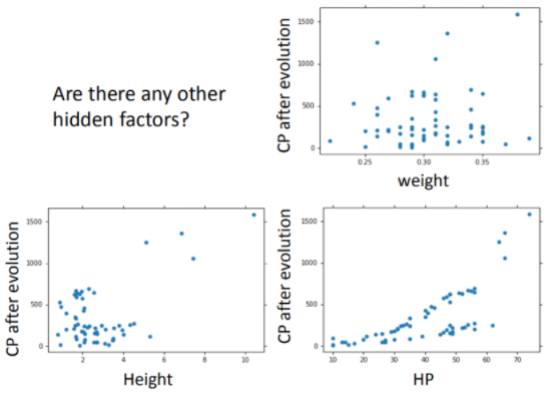


我们就可以把宝可梦的种类当作一个特征放入到模型中：

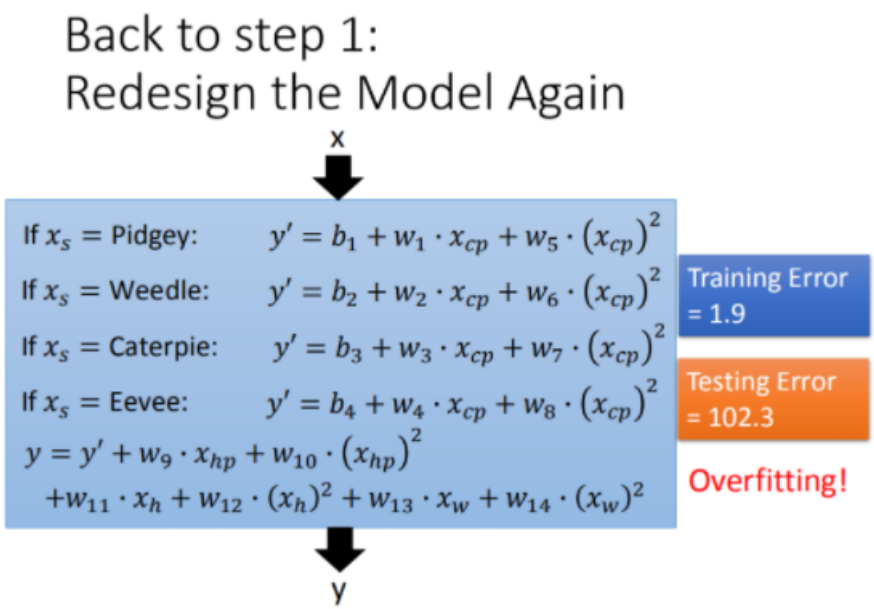


可以得到一个测试loss更低的模型

- 我们再尝试加入更多特征

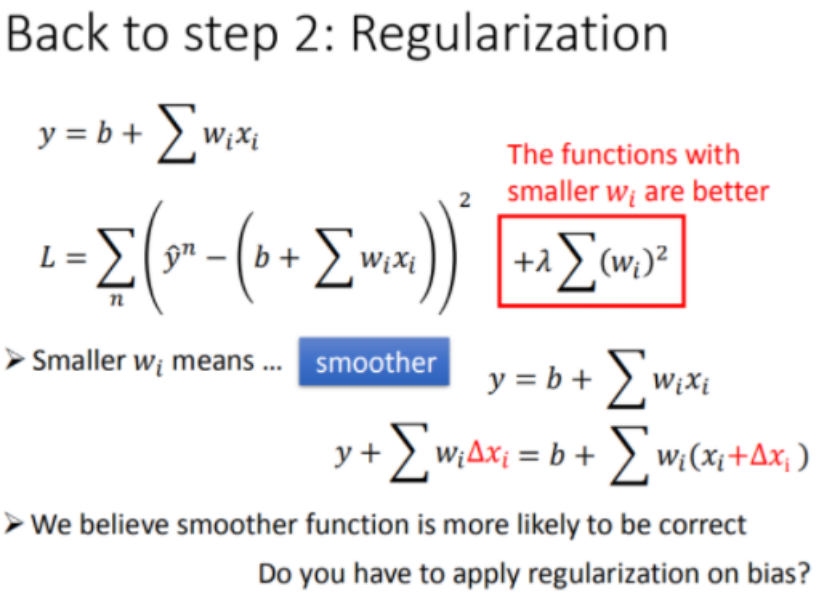


重新设计模型：



结果发生了过拟合问题

- 正则化



在loss函数中加入一个约束参数值大小的项，使得，参数值在保证loss小的情况下，参数值小，参数值小，意味着模型更加简单。这样就可以有效的解决过拟合问题。