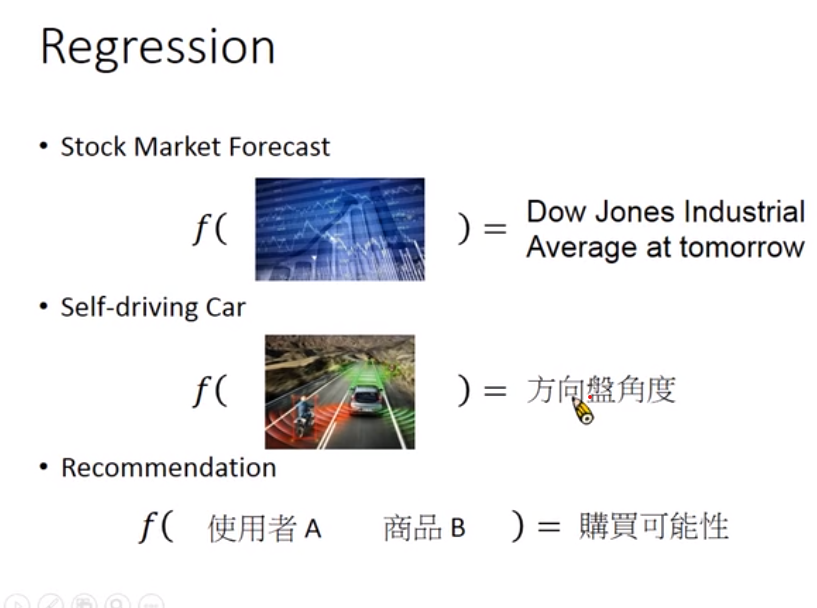
# REGRESSION

回归

预测可能性



假设一个fiction给你一个input求output.

# 问题引入：预测宝可梦的cp

X：表示一只宝可梦，用下标表示该宝可梦的某种属性

Xcp：表示该宝可梦进化前的cp值

Xs： 表示该宝可梦是属于哪一种物种

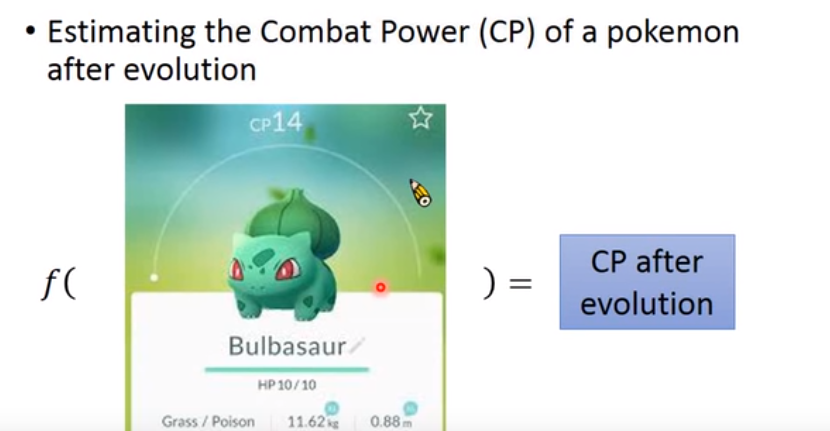
Xhp：表示该宝可梦的hp值即生命值是多少

Xw： 代表该宝可梦的重重量

Xh： 代表该宝可梦的高度

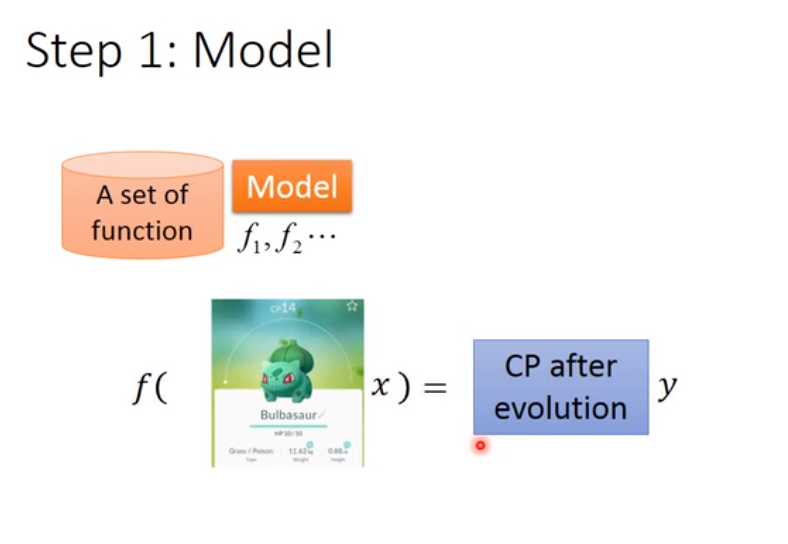
f： 表示我们要找的function

y： 表示function的output，即宝可梦进化后的cp值



（此话为视频中原话）我们做这个主要有三个步骤:1.找一个model

1. 定义function set某一个fiction来evulate的它的好坏
2. 找一个最好的fiction

我们先定义一个model，但是我们 不清楚到底怎样的model合适，先假设最简单的一个。

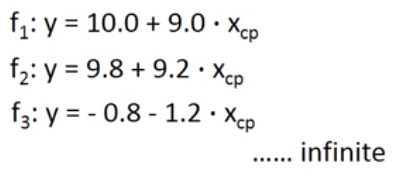
我们认为Y=b+wXcp（一种linear modei)

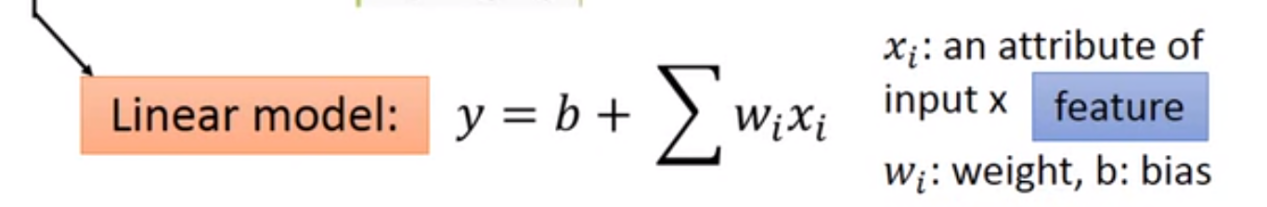
y代表进化后的cp值，Xcp代表进化前的cp值，w和b代表未知参数，可以是任何数值

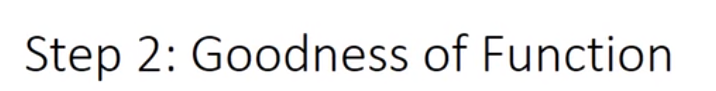
根据不同的w和b，可以确定不同的无穷无尽的function，而这个抽象出来的式子就叫做model，是以上这些具体化的function的集合，即function set.

但我们只是考虑了宝可梦进化前的cp值，需考虑其他因素。

可以随便假设w和b例如：



一种model



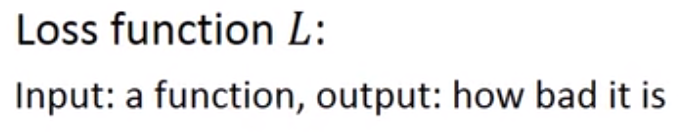
收集traning data才能赵fiction。

###### **参数说明**

：用上标来表示一个完整的object的编号，表示第i只宝可梦(下标表示该object中的component)

：用表示一个实际观察到的object输出，上标为i表示是第i个object。

有了traning data 之后我们可以定义一个fiction的好坏，我们要定义另一个fiction——loss fiction一个特别的fiction。

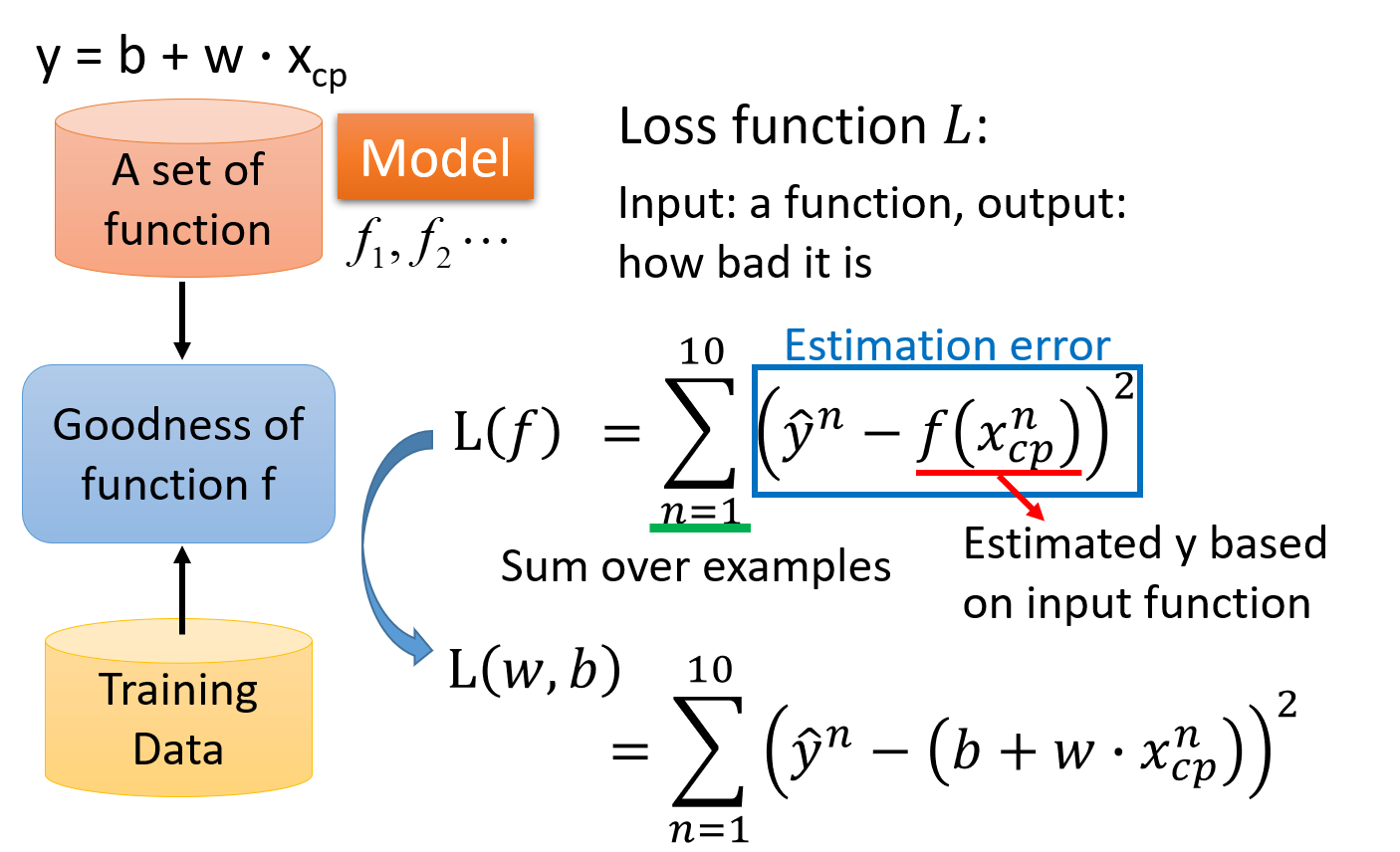


由于f是由w和b 决定的，L衡量w 和b的好坏

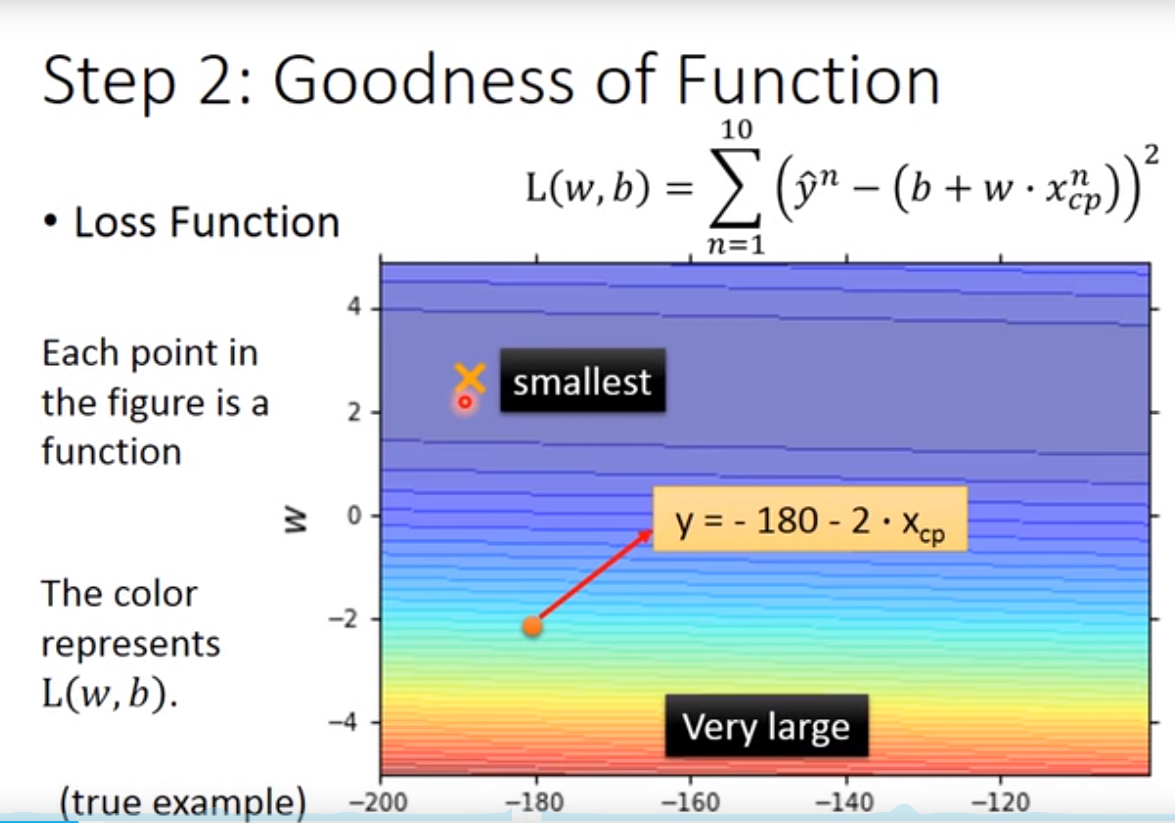
之前提到的model是由我们自主选择的，这里的loss function也是，最常用的方法就是采用类似于方差和的形式来衡量参数的好坏，即预测值与真值差的平方和；这里真正的数值减估测数值的平方，叫做估测误差，Estimation error，将10个估测误差合起来就是loss function



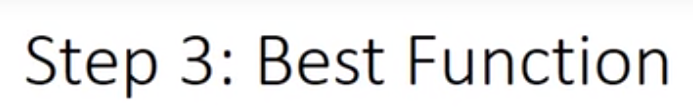
如果越大，说明该function表现得越不好；越小，说明该function表现得越好



L可视化

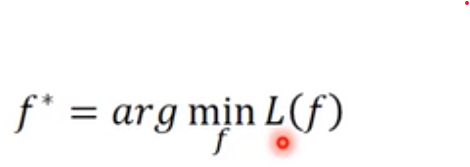


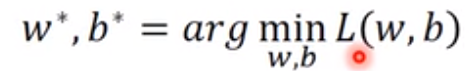
该图中的每一个点都代表一组（w,b)，也就是对应着一个fiction；而该点的颜色对应着的loss function的结果L(w,b),它表示该点对应function的表现有多糟糕，颜色越偏红色代表Loss的数值越大，这个function的表现越不好，越偏蓝色代表Loss的数值越小，这个function的表现越好。



我们从fiction set 选出最好的fiction.

找一个f使L最小，让L最小的fiction写作。





要做的是找出所有的w和b使L最小，这个值是最好的w和b写作

利用线性代数的知识，可以解得这个closed-form solution，但这里采用的是一种更为普遍的方法——（普遍在于只要L可以微分都可以处理）

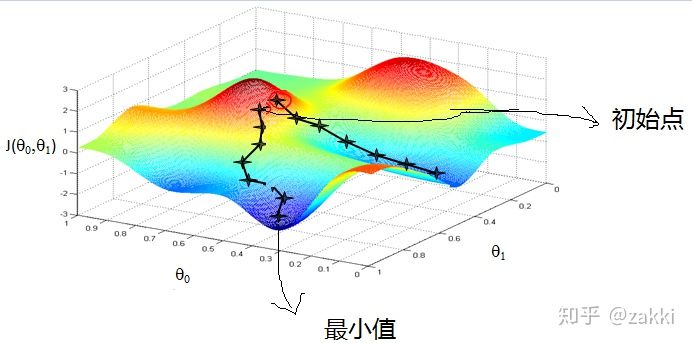
# Geadient Descent（梯度下降）

首先简单介绍一下梯度下降

在微积分中，对多元函数的参数求偏导，把求导后的结果（各个参数的偏导数）以向量的形式表达出来就是梯度。比如 IMG_256 分别对 IMG_257 求偏导，得出的 IMG_258 就是梯度。如果是涉及三个参数的函数，那么 IMG_259 就是梯度。

那么梯度的意义又是什么？从几何的角度，梯度就是在这一点函数增加最快的方向，反之梯度的相反方向就是函数减小最快的方向。例如对于 IMG_256 点，- IMG_257 的方向就是 IMG_258 减小最快的方向。

这里举一个爬山的例子：



比如我们在爬一座山，我们现在在初始点所标注的位置，现在想要尽快下山，应该用什么样的方式呢？我们需要朝着初始点的梯度负方向前进一步，然后在在新的初始点位置继续向梯度负方向前进一步，就这样一步又一步，我们可能会达到一个局部的山峰最低处。如果这个山峰是凸函数形状的，我们这样一步又一步地走下去，可能会走到山脚。

以只带单个参数w的Loss Function L(w)为例，首先保证是**可微**的 我们的目标就是找到这个使Loss最小的，实际上就是寻找切线L斜率为0的global minima最小值点(注意，存在一些local minima极小值点，其斜率也是0)

有一个暴力的方法是，穷举所有的w值，去找到使loss最小的，但是这样做是没有效率的；而gradient descent就是用来解决这个效率问题的

首先随机选取一个初始的点 (当然也不一定要随机选取，如果有办法可以得到比较接近的表现得比较好的当初始点，可以有效地提高查找的效率)

计算L在w=w0的位置的微分，即，几何意义就是切线的斜率

如果切线斜率是negative负的，那么就应该使w变大，即往右踏一步；如果切线斜率是positive正的，那么就应该使w变小，即往左踏一步，每一步的步长step size就是w的改变量

#### w的改变量step size的大小取决于两件事

一是现在的微分值有多大，微分值越大代表现在在一个越陡峭的地方，那它要移动的距离就越大，反之就越小；

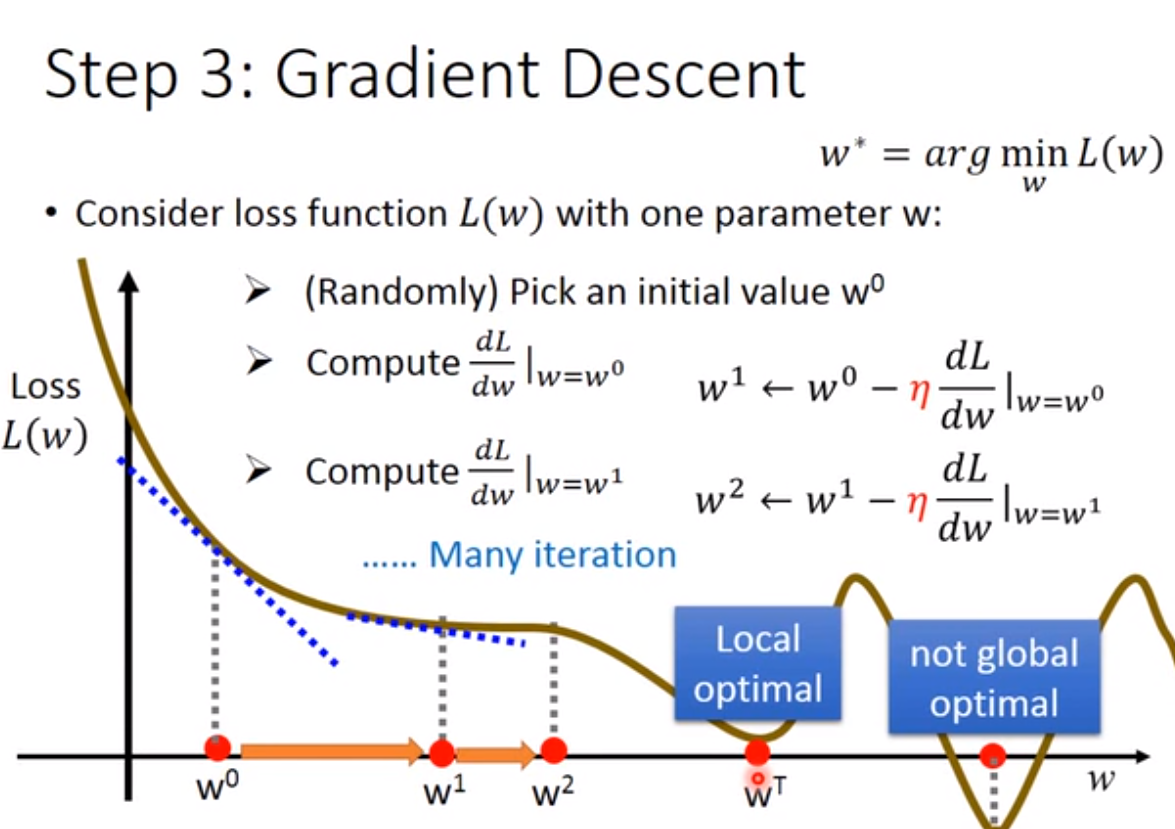
二是一个常数项η，被称为learning rate，即学习率，它决定了每次踏出的step size不只取决于现在的斜率，还取决于一个事先就定好的数值，如果learning rate比较大，那每踏出一步的时候，参数w更新的幅度就比较大，反之参数更新的幅度就比较小

如果learning rate设置的大一些，那机器学习的速度就会比较快；但是learning rate如果太大，可能就会跳过最合适的global minima的点

因此每次参数更新的大小是 ，为了满足斜率为负时w变大，斜率为正时w变小，应当使原来的w减去更新的数值，即

此时对应的斜率为0，我们找到了一个极小值local minima，这就出现了一个问题，当微分为0的时候，参数就会一直卡在这个点上没有办法再更新了，因此通过gradient descent找出来的solution其实并不是最佳解global minima

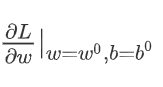
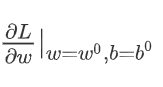
但是，在linear regression上，是没有local minima的，因此可以使用这个方法



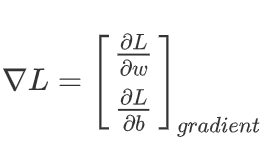
##### **两个参数的问题**

今天要解决的关于宝可梦的问题，是含有two parameters的问题，即

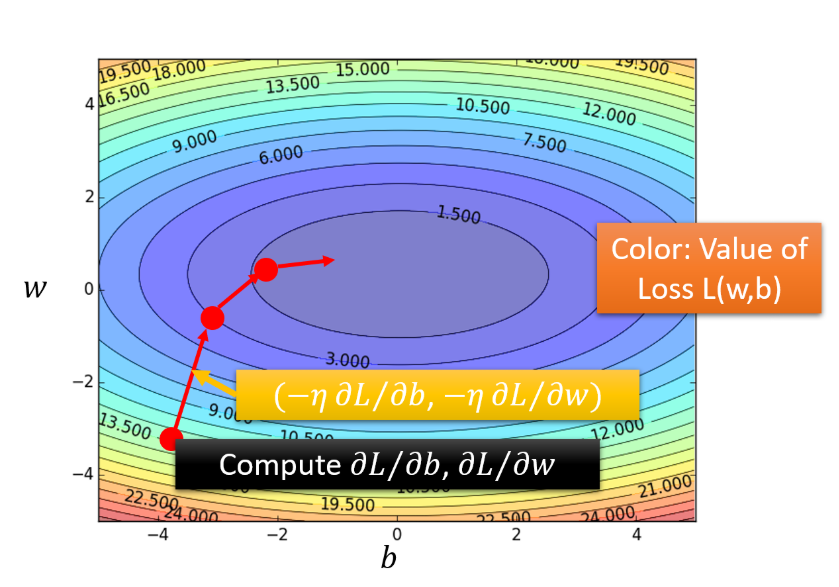
当然，它本质上处理单个参数的问题是一样的

首先，也是随机选取两个初始值，和w0和b0然后分别计算这个点上，L对w和b的偏微分，即 和 。

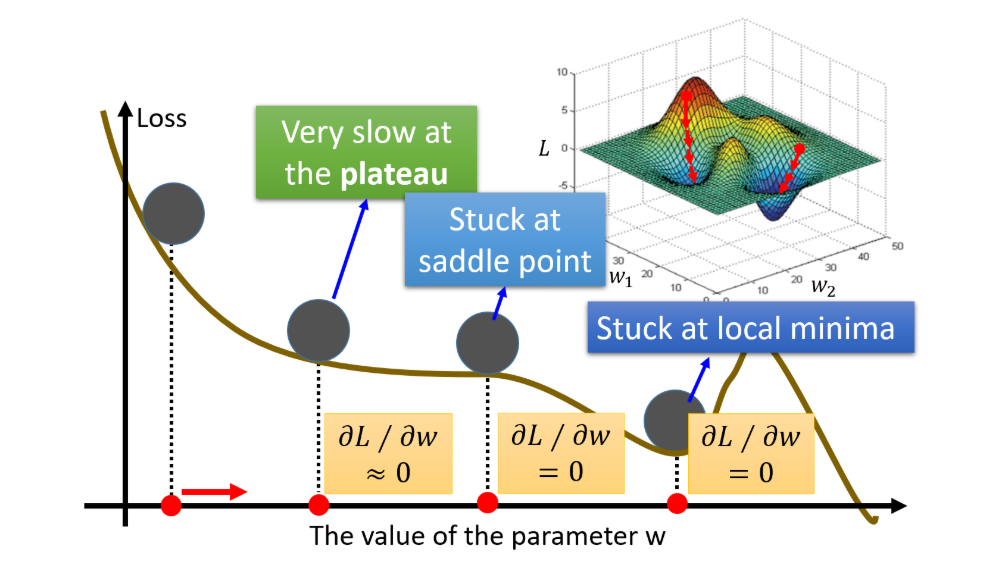
更新参数，当迭代跳出时，对应着极小值点。

实际上，L 的gradient就是微积分中的那个梯度的概念，即

**每次计算得到的梯度gradient，即由**和**组成的vector向量，就是该等高线的法线方向(对应图中红色箭头的方向)；而****的作用就是让原先的朝着gradient的方向即等高线法线方向前进，其中η(learning rate)的作用是每次更新的跨度(对应图中红色箭头的长度)；经过多次迭代，最终gradient达到极小值点**



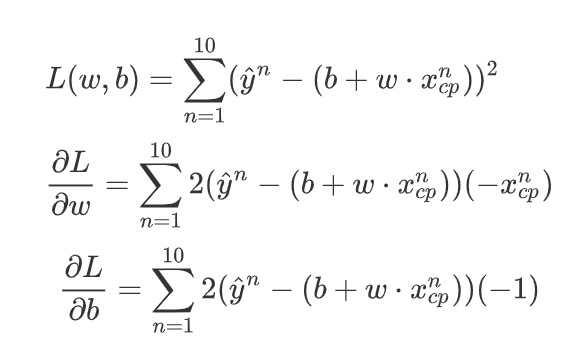
# 缺点：梯度为零的点必是极小值点，但不一定是最小值点。这样会导致一个问题：如果L的图像极小值点比较多，而初始的w0的取值是随机的，在gradient decent 过程中会经常停下来；当遇到坡度平缓，过程效率变低有可能会stuck。



(原话）

但是！在linear regression里，loss function实际上是**convex**的，是一个**凸函数**，是没有local optimal局部最优解的，他只有一个global minima，visualize出来的图像就是从里到外一圈一圈包围起来的椭圆形的等高线(就像前面的等高线图)，因此随便选一个起始点，根据gradient descent最终找出来的，都会是同一组参数。

# 宝可梦的问题

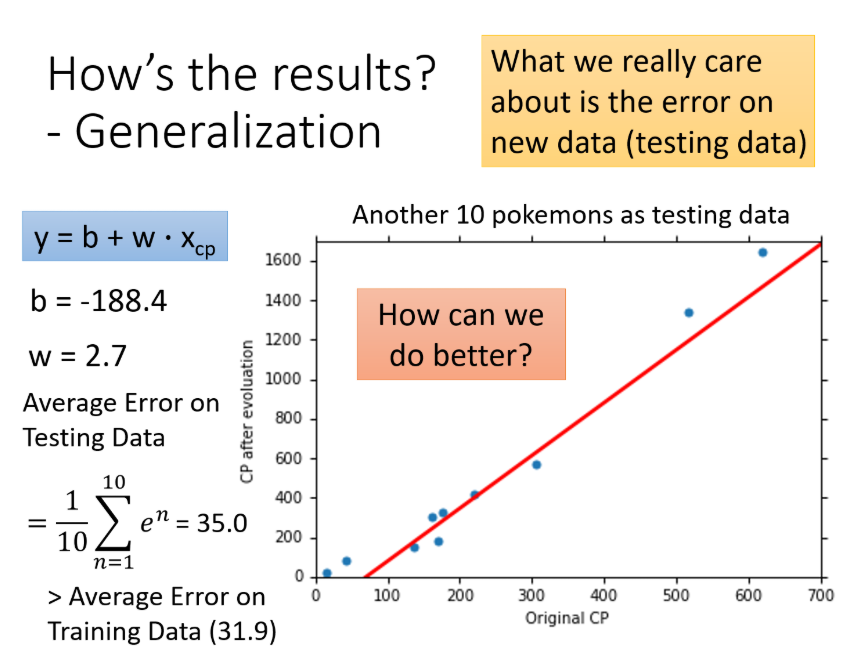


求具体的l 对w和b的偏微分

# 结果

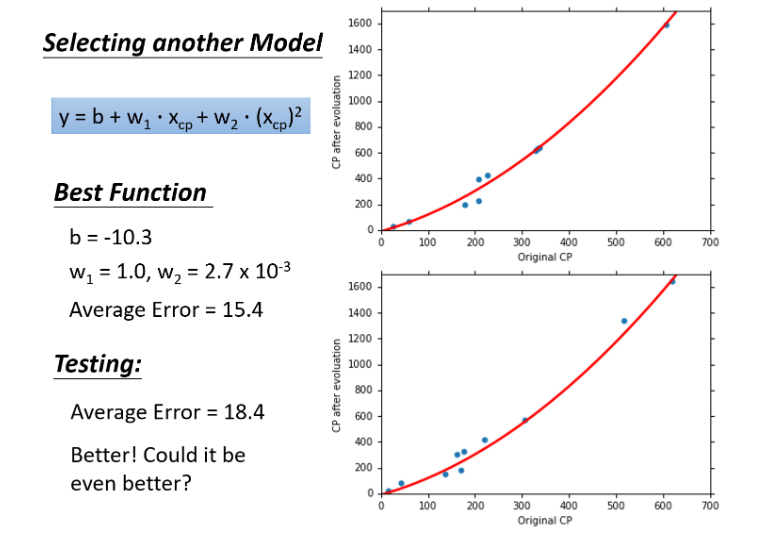
根据gradient descent，我们得到的中最好的参数是b=-188.4, w=2.7

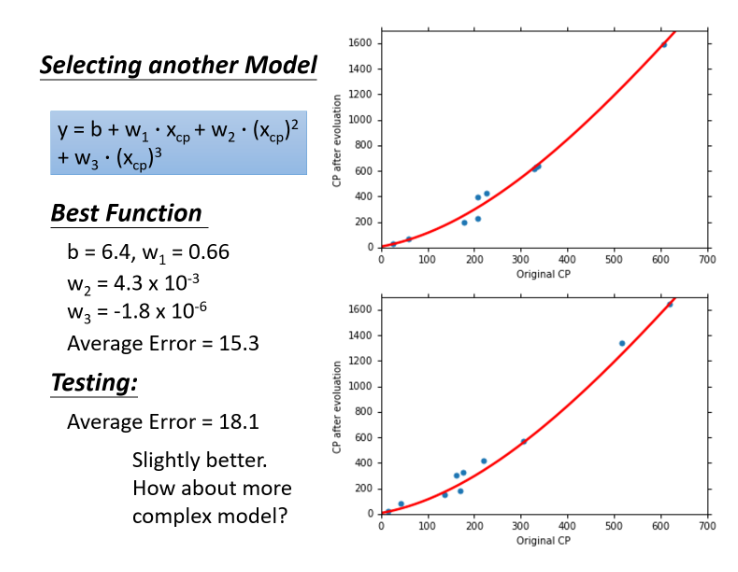
我们需要有一套评估系统来评价我们得到的最后这个function和实际值的误差error的大小；这里我们将training data里每一只宝可梦 i进化后的实际cp值与预测值之差的绝对值叫做ei，而这些误差之和Average Error on Training Data。为31.9

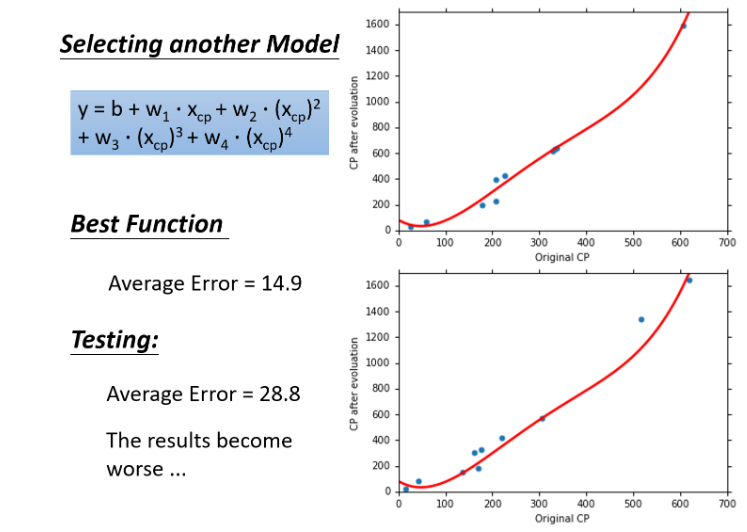


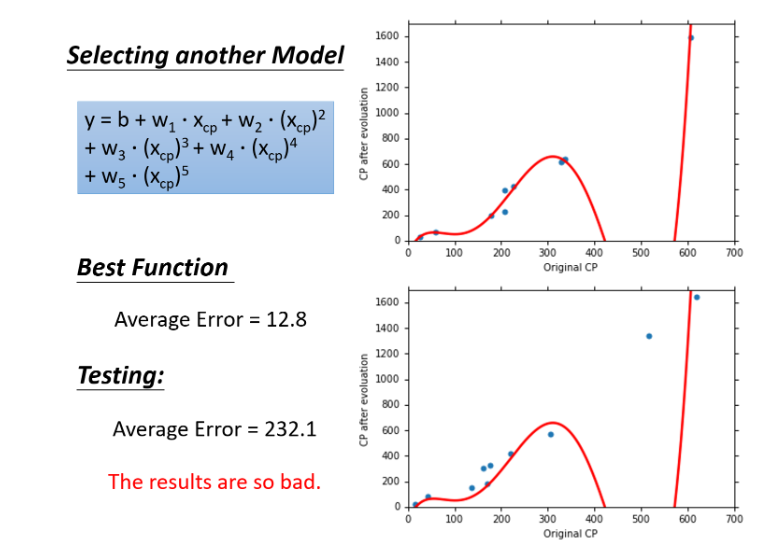
# 如何更好

我们重新去设计model；如果仔细观察一下上图的data，就会发现在原先的cp值比较大和比较小的地方，预测值是相当不准的。

于是我们设计了 平方 三次方 四次方 五次方

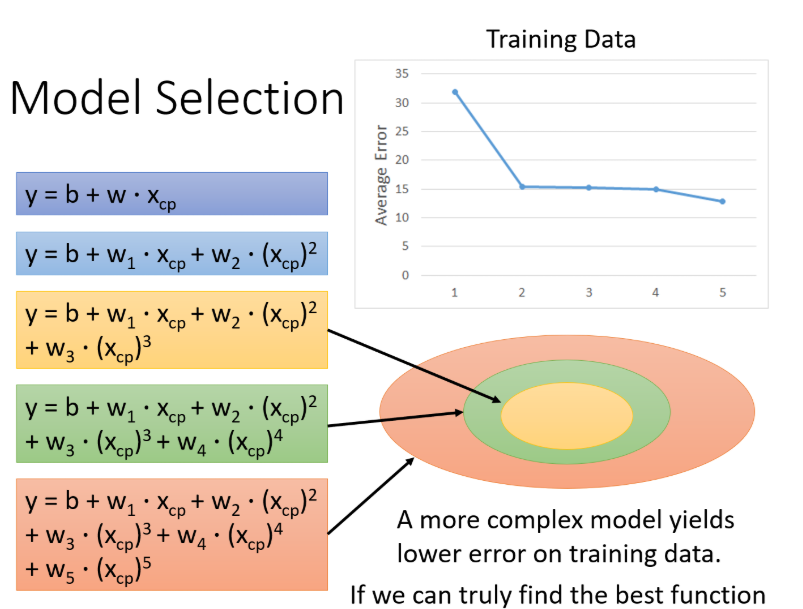






（原话）这5个model的training data的表现：随着高次项的增加，对应的average error会不断地减小；实际上这件事情非常容易解释，实际上低次的式子是高次的式子的特殊情况(令高次项对应的为0，高次式就转化成低次式)

也就是说，在gradient descent可以找到best function的前提下(多次式为Non-linear model，存在local optimal局部最优解，gradient descent不一定能找到global minima)，function所包含的项的次数越高，越复杂，error在training data上的表现就会越来越小；但是，我们关心的不是model在training data上的error表现，而是model在testing data上的error表现



在training data上，model越复杂，error就会越低；但是在testing data上，model复杂到一定程度之后，error非但不会减小，反而会暴增，在该例中，从含有四次方项的model开始往后的model，testing data上的error出现了大幅增长的现象，通常被称为**overfitting过拟合**

