

P2 Regression - Case Study

#deepLearn #Regression #机器学习

Regression(数值预测) 可以做什么

- 股票预测系统 $f(\text{关于这支股票的各种资料}) = \text{明天的股价}$
 - 自动驾驶汽车 $f(\text{汽车周围的环境}) = \text{接下来的驾驶操作}$
 - 推荐算法 $f(\text{你的各种数据}) = \text{你最可能购买的物品}$
 - 预测宝可梦的CP值 (大雾)
CP值 为一只宝可梦的战斗力
就可以决定是否要进化这支宝可梦 (大雾)
-

任务： $f(\text{一只宝可梦的信息}) = \text{他进化的CP值}$

分类： Regression Problem

输入： 一只宝可梦所有的信息

输出： 这只宝可梦进化后的战斗力

步骤列表

- 找一个model = function set 模型 = 函数集
 - 定一个function set 里面的 function
 - 找一个最好的function
-

Step 1: Model

找一个Model 或者也可以称为 Function set

随便写一个 Model

注意: a_b 表示 b 是 a 的下标

$$Y = b + w * x_{cp}$$

Y: 进化后的CP值

b: 常数项

w: 常数项

X_cp : 进化前宝可梦的CP值

其中 w, b 是参数

(可以为任何数)

这个Model (function set) 里面有无穷无尽个函数 (因为 w 和 b 可以随便取值)

形如 $Y = b + w * x$ 这样的模型 (model) 叫做线性模型 (linear model)

#linear

$$Y = b + \sum w_i * x_i$$

x_i : 称为 feature (特征)

w_i : weight (权重)

b: bias(偏差)

Step 2: Goodness of Function

类型: Supervise Learning Test

我们用 a^b 表示 b 是 a 的上标

Function input: $x^1 X^2 \dots$

(注: 用上标来表示一个完整的Component (元件))

Function output (Value) : Y^1H Y^2H ...

(注：在一个完整的Component 后面 跟上一个大写的H表示他是一个输出)

(注：这不是一个推荐的写法，只不过是因为在markdown方便)

注意：在本次试验中，我们都是输入输出值都是单个数值，所以可能不需要上标下标，但是在后面的试验中，这是一个推荐的方法

Training Data :

10 Pokemons

x^1, y^1H

x^2, y^2H

...

This is the real data(老师注：这是真实的数据) (大雾)

Goodness of Function (定义一个函数的好坏)

如何去定义：定一个损失函数(Loss function) 记为 Loss 或者 L (大写)

Input : a function (输入是一个函数)

Output: how bad it is (这个函数怎么样)

$L(f) = L(w, b)$

衡量一个函数的好坏 等价于 衡量一组参数的好坏

如何去求这个Loss (常见求法：写均方差，如下图所示)

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

y^n : 这是理想值 (真正的数值)

里面的小括号是：这个函数预测出来的值 (预测的数值)

两者求差就是 预测的预测

最后再把预测误差的求和

Step 3: Best Function (选出最好的函数)

Pick The Best Function !

$$f^* = \arg \min_f L(f)$$

要找到一个 f 使得 $L(f)$ 最小

$$w^*, b^* = \arg \min_{w,b} L(w,b)$$

穷举所有的 w 和 b 来找到最小的 $L(w,b)$

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

，可以給我們最好的結果。

找到所有 w b 带进去 损失函数 看看那个最小



Gradient Descent 梯度下降算法

#GradientDescent

该方法不只适用于解这么一个函数,这个函数只要是L损失函数是可以微分的, 就可以求。

简化题目:

Consider loss function $L(w)$ with one parameter w ;

(只考慮有一个参数的损失函数)

暴力方法: 穷举所有 w 可能的取值, 全部带进去看那个最小。

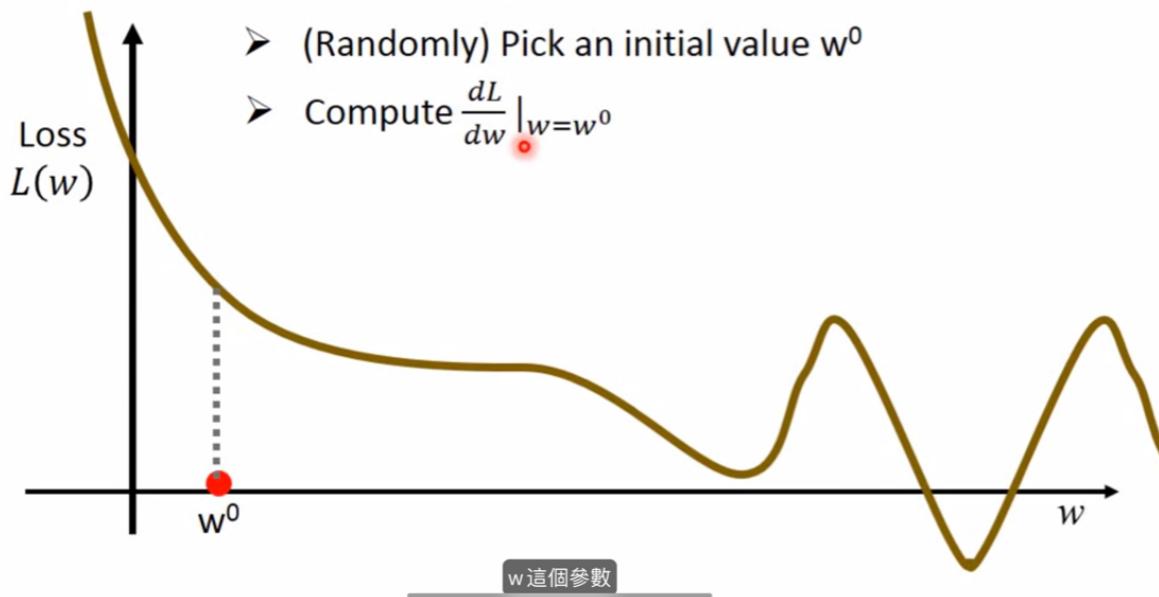
Gradient Descent 如何解:

1. (Randomly) Pick an initial value w^0 (随机取起始点 记为 w^0)
2. compute L 对 w 的微分 也就是 切线斜率

Step 3: Gradient Descent

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

- Consider loss function $L(w)$ with one parameter w :

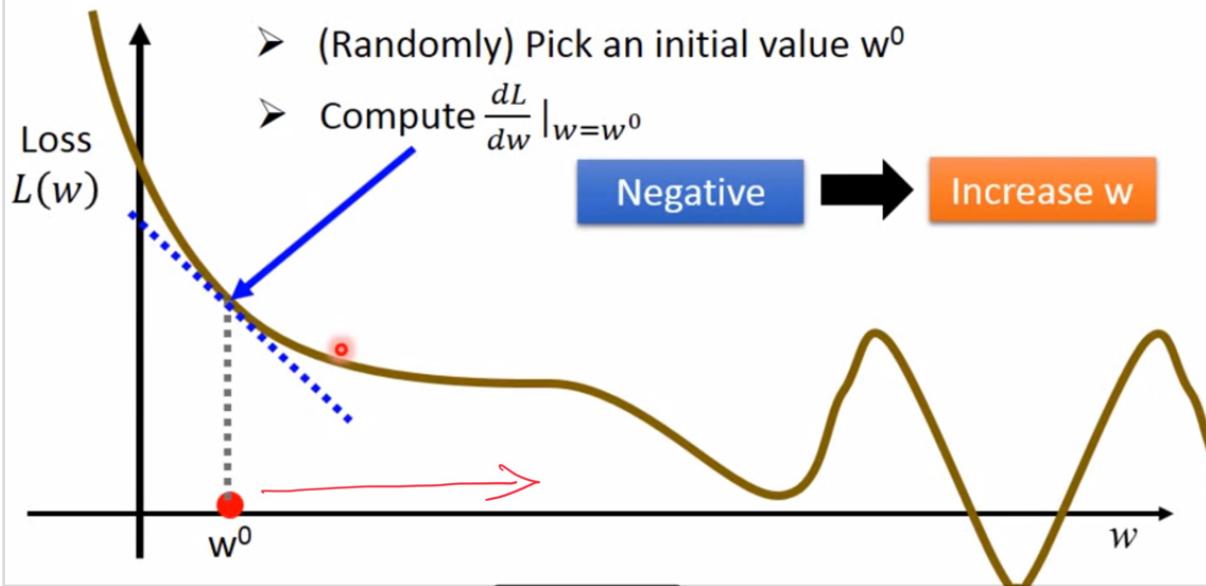


- 如果 微分是负数 也就是 斜率 是负数 也就表明 左边比较高，就应该增加 w 值

Step 3: Gradient Descent

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

- Consider loss function $L(w)$ with one parameter w :



- 如上图所示应该增加 w 值 使得 $L(w)$ 减少

反之 就应该减少 w

3. 那么应该增加多少呢?

取决于现在的微分值有多大 微分值大表示陡峭 也表示离极值点远, 就应该移动的快, 反之就越小

另外还取决于一个常数项 η (实际上应该是 η 一他) 我们称之为 learning rate (学习率)

- 最后的更新应该为:

$$w^1 \leftarrow w^0 - \eta \frac{dL}{dw} |_{w=w^0}$$



4. 重复之前的步骤 重新计算 新的点微分值 (也就是执行 步骤 2)

5. 反复更新后也就会到一个 Local optimal (局部最优) 的位置 也就是 微分值为 0 处

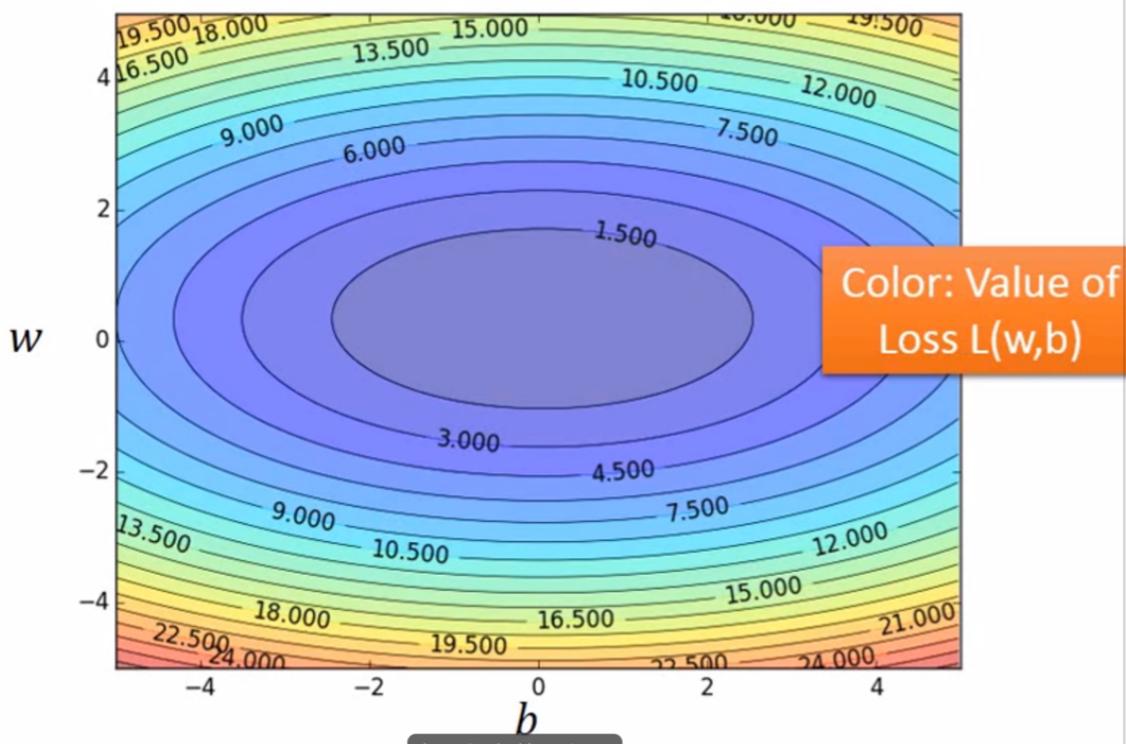
如果说有两个参数?

其实是一样的

1. 随机选取两个初始值
2. 计算 w_0, b_0 的偏微分
3. 更新 w_0, b_0 参数 获得一组新的参数 w_0, b_1 , 在回到 2 循环做
4. 直到达到一个 局部最优解 (微分为 0) 结束

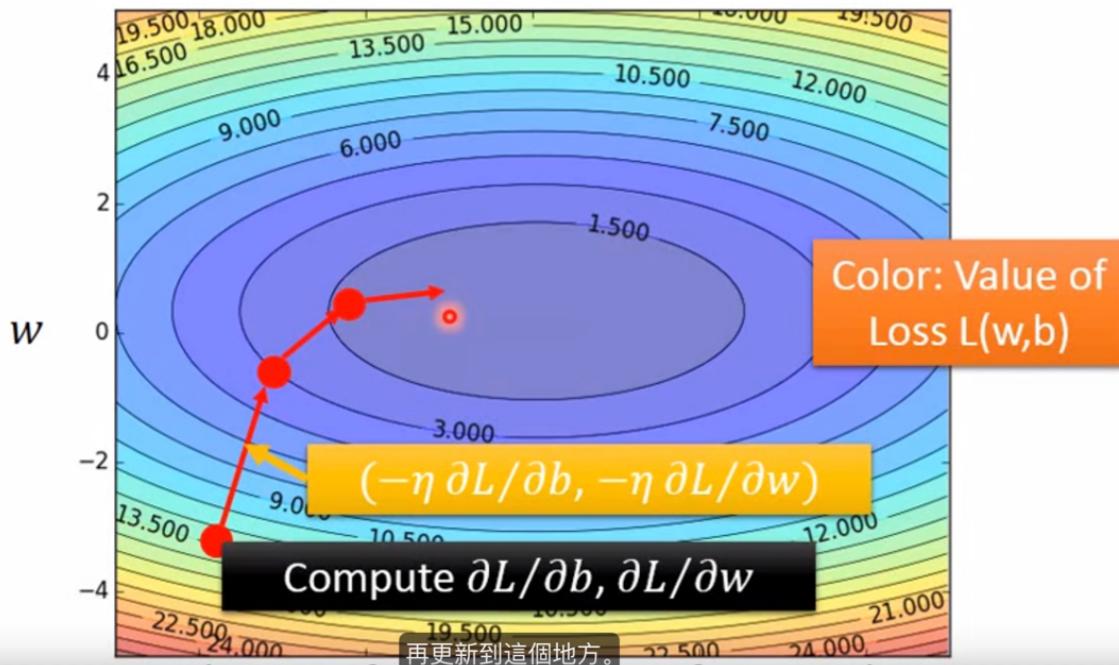
结果图:

Step 3: Gradient Descent



梯度下降原理示意图：

Step 3: Gradient Descent

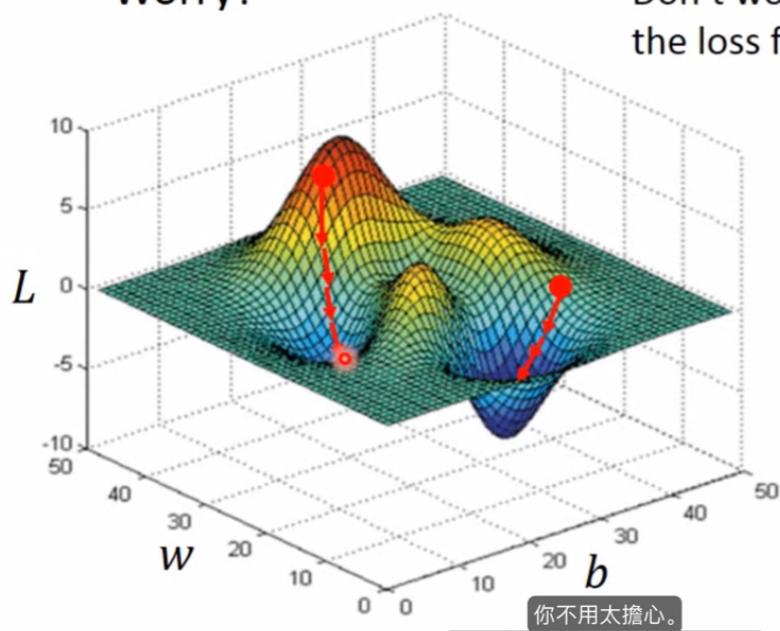


Gradient Descent 让人担心是否会卡在局部最优解?

Step 3: Gradient Descent

- Worry?

Don't worry. In linear regression,
the loss function L is convex.



在linear regression 不用过于担心

Loss function 是 convex (凸函数) 就是说没有 局部最优解的地方!

所以不存在落入局部最优解

所以无论起始点在哪里都会回到同一个值

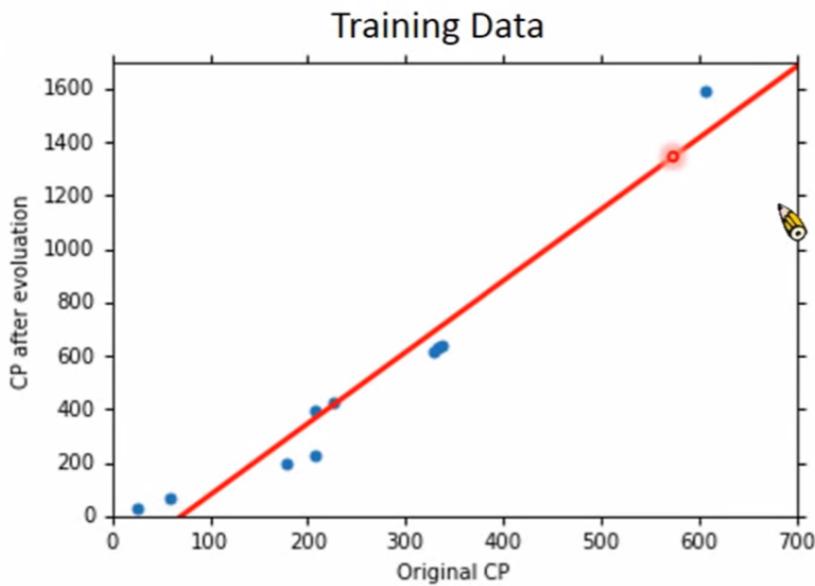
结果:

How's the results?

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

$$b = -188.4$$

$$w = 2.7$$



是那条红色的线，没有办法完全的正确的所有宝可梦的CP值

如果想要知道有多不好

可以计算一下Error值：每一个红色的点和对应红色的点的距离的和就是Error

这个并不是我们真正关心的，我们关心的是

Generalization 预测值

我们真正关心的是预测值 (Testing data)

+ 关注

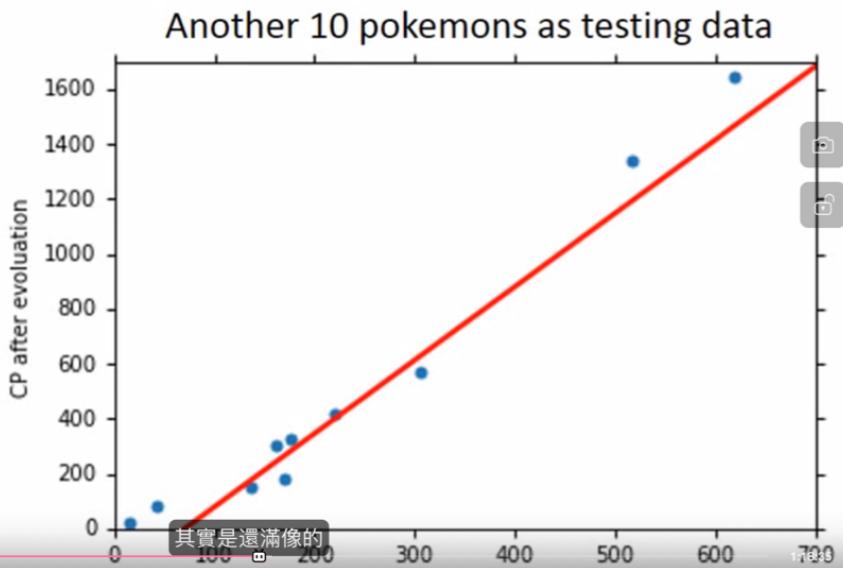
How's the results? - Generalization

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

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

$$b = -188.4$$

$$w = 2.7$$



其实也可以大致预测进化后的CP值

How can we do better

重新设计我们的model

需要一个更加复杂的model

需要引入二次式子

新 model:

$$Y = b + w_1 * x_{cp} + w_2 * (x^2)_{cp}$$

多背景知识，越听越有信心

Selecting another Model

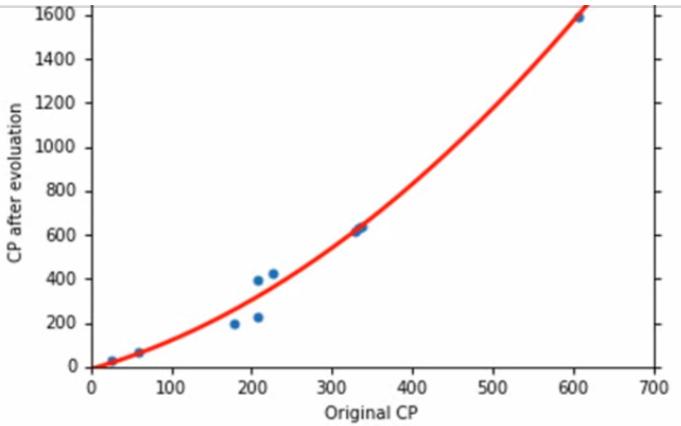
$$y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2$$

Best Function

$$b = -10.3$$

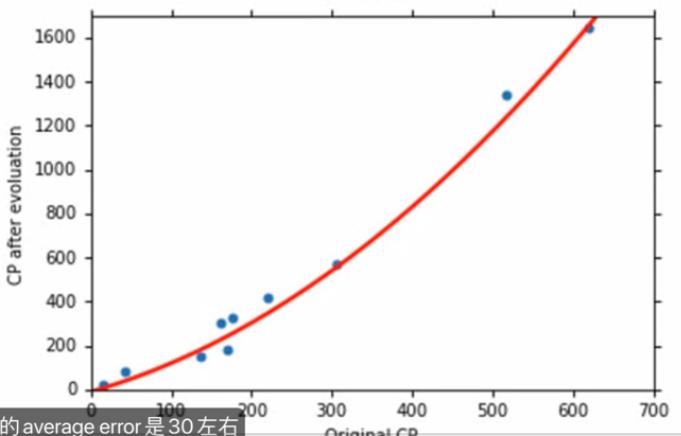
$$w_1 = 1.0, w_2 = 2.7 \times 10^{-3}$$

$$\text{Average Error} = 15.4$$



Testing:

$$\text{Average Error} = 18.4$$



Error值降低了很多

More Better ?

引入三次式子?

4, 5, 6 ...?

Not Better (过拟合)

在 train 上获得一个更好的结果

但是在 test 上却变差了

20:03 8月16日周一

< Regression - Case Study

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

Model 变烂了原来 500 应该是 正的，现在竟然是 负的！

但是在 training data 效果非常好

但是在 test data 结果会非常不准（烂）

< Regression - Case Study

Model Selection

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

A more complex model yields lower error on training data.

讓你的 error rate 越來越低。

在理论上 式子越复杂，就能使得 error rate降低

Model Selection



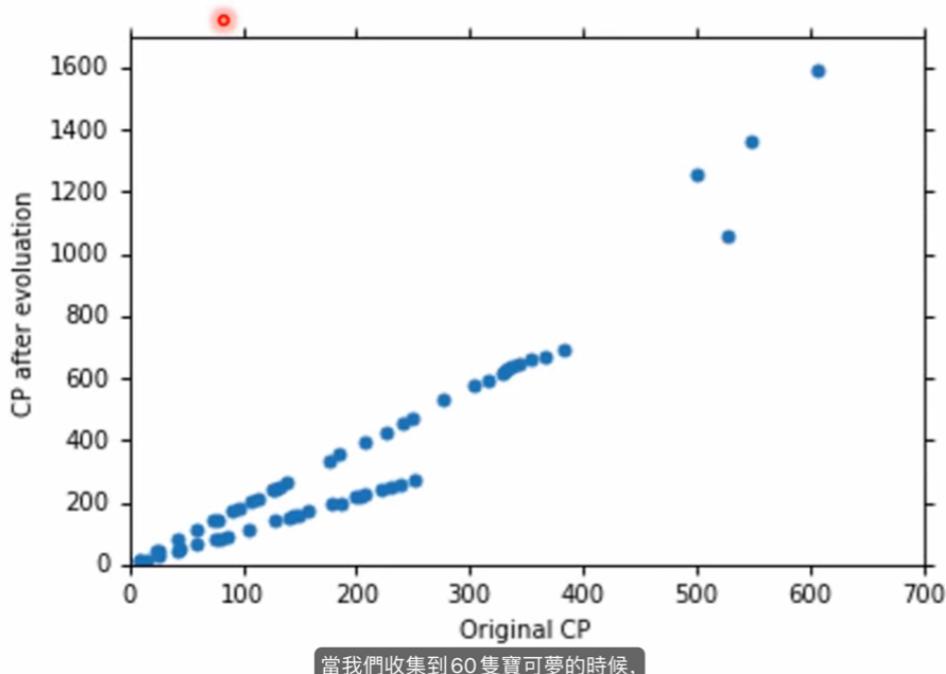
但是在 testing data 上，

但是在 test data 上 第四个第五个error就会暴增。

结论：越复杂的模型往往能在 training data 获得优秀的结果，但是不一定能在 test data 获得优秀的结果，这个就叫 Overfitting (过度拟合)

Let's collect more data

Let's collect more data

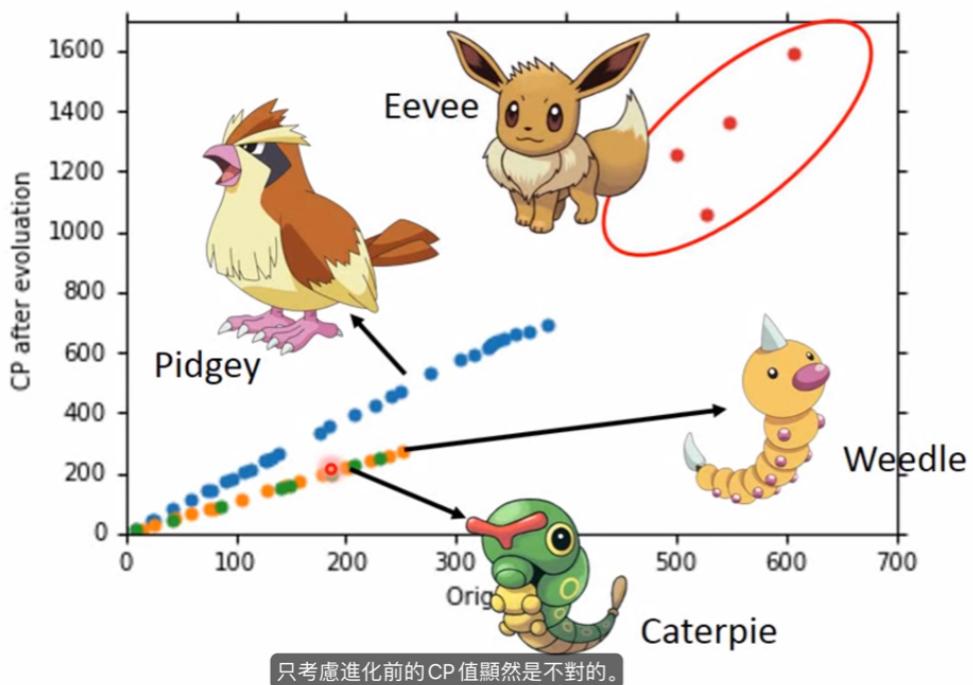


我们会发现他不是一个简单的线性关系

有一些隐藏的因素影藏在里面

就是类型

What are the hidden factors?



所以显然只考虑CP值是不对的！

所以在设计 Model 时要考虑全面

要重新设计一下 Model

如果物种是 XXX

$Y = \dots$

如果物种是 XXXX

$Y = \dots$

不同的物种用不同的方程

Back to step 1: Redesign the Model

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

Linear model?

$$y = b_1 \cdot \delta(x_s = \text{Pidgey})$$

$$+ w_1 \cdot \delta(x_s = \text{Pidgey}) x_{cp}$$

$$+ b_2 \cdot \delta(x_s = \text{Weedle})$$

$$+ w_2 \cdot \delta(x_s = \text{Weedle}) x_{cp}$$

$$+ b_3 \cdot \delta(x_s = \text{Caterpie})$$

$$+ w_3 \cdot \delta(x_s = \text{Caterpie}) x_{cp}$$

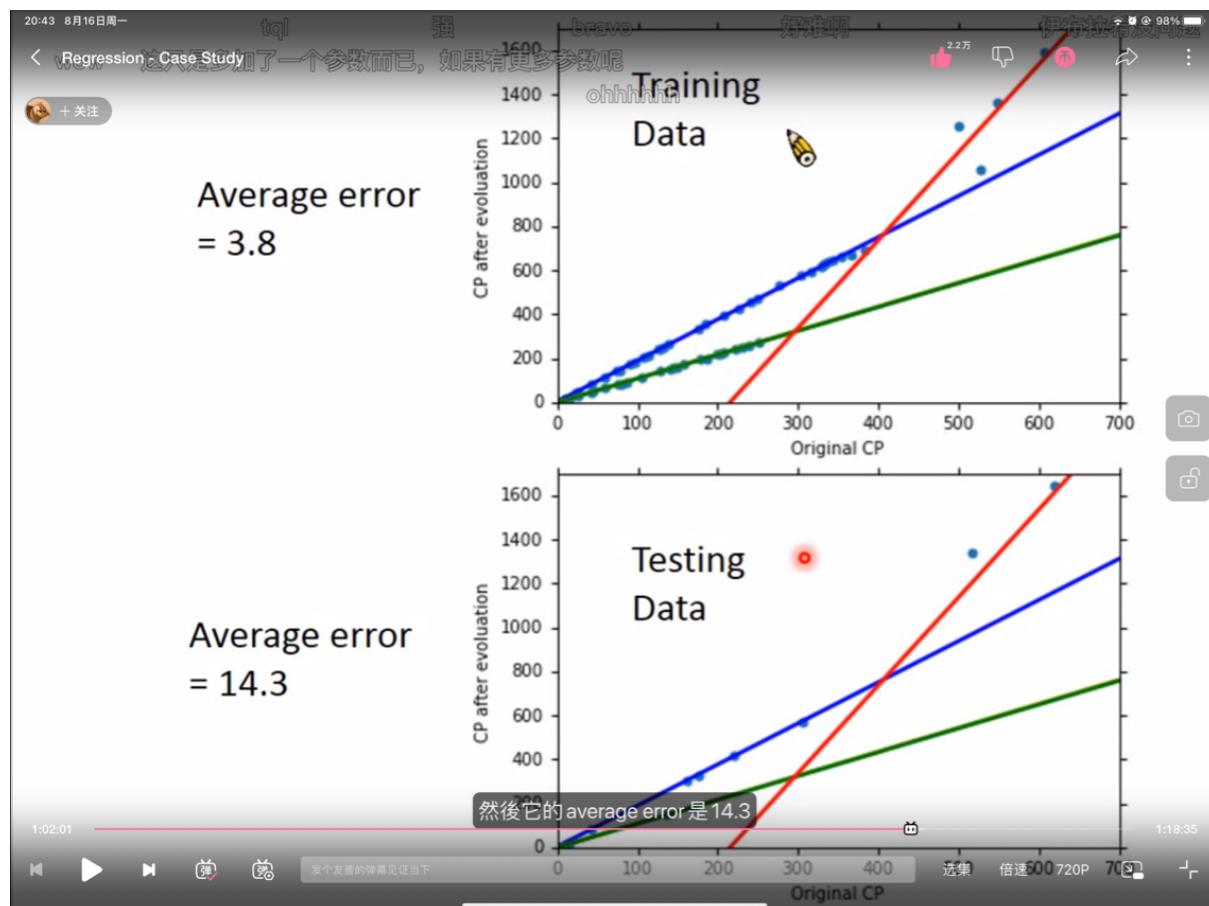
$$+ b_4 \cdot \delta(x_s = \text{Eevee})$$

$$+ w_4 \cdot \delta(x_s = \text{Eevee}) x_{cp}$$

如果你有修過信號的處理，我想應該知道 δ 這個 function 是指甚麼。

其中 δ 表示判断如果正确表示 1 否则表示 0

结果：



Are there any other hidden factors?

会不会跟 weight 和 Height 或者 HP 有关系？？？

如果不知道哪些有关系，那就把能得到的数值全部塞进去！会怎么样？

overfitting 是的 这老师太有趣了
分类，所以要先算好y撇 是不是能把游戏人的规则给弄出来
Back to step 1: 几百个参数是认真的吗

Redesign the Model Again

X

If $x_s = \text{Pidgey}$: $y' = b_1 + w_1 \cdot x_{cp} + w_5 \cdot (x_{cp})^2$

If $x_s = \text{Weedle}$: $y' = b_2 + w_2 \cdot x_{cp} + w_6 \cdot (x_{cp})^2$

If $x_s = \text{Caterpie}$: $y' = b_3 + w_4 \cdot x_{cp} + w_7 \cdot (x_{cp})^2$

If $x_s = \text{Eevee}$: $y' = b_4 + w_4 \cdot x_{cp} + w_8 \cdot (x_{cp})^2$

$y = y' + w_9 \cdot x_{hp} + w_{10} \cdot (x_{hp})^2$
 $+ w_{11} \cdot x_h + w_{12} \cdot (x_h)^2 + w_{13} \cdot x_w + w_{14} \cdot (x_w)^2$

Training Error = 1.9

Testing Error = 102.3

Overfitting!

Y

我們今天得到的數值很糟是 102.3 這樣子，

并不好，很糟糕！

Back to step 2: Regularization (正则化)

重新定义我们的loss function

Back to step 2: Regularization

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

The functions with
smaller w_i are better

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

- Why smooth functions are preferred?

它是比较平滑的。

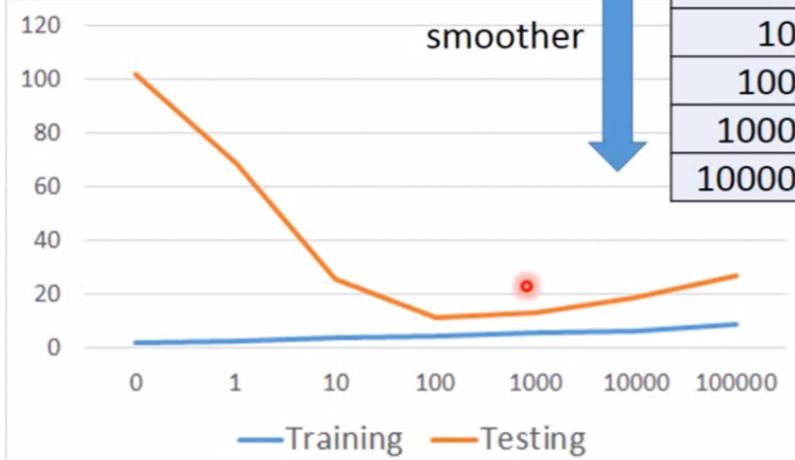
在我们的 Loss 函数 增加一项 参数值的平方，根据 Loss 越小越好的规则，所以我们期待 参数值平方越小越好

为什么呢？：参数值越小，表示这个函数是比较平滑的，当输入有变化，输出对输入的变化是不敏感的

如果我们有一个平滑的函数，当我们输入测试数据时，就会受影响比较小。
应为会有影响项。

实验结果

Regularization



λ 值越大代表考慮 smooth 的那個 regularization 那一項它的影響力越大

training date error 偏大

testing data error 可能会减少！

我们喜欢比较平滑的function 对 input 不那么铭感

但是我们又不喜欢太平滑的function 因为 lambda 太大就会欠缺对 train data 本身的考虑

所以要多平滑？

多次调整 lambda

结论：

1. 感谢参加对宝可梦的研究（大雾）
2. 怎么做梯度下降
3. 过拟合
4. 正则化
5. 获得了一个较低的 error，如果测试更多的数据会怎么样?
 - | 预测应该是会更高效验

