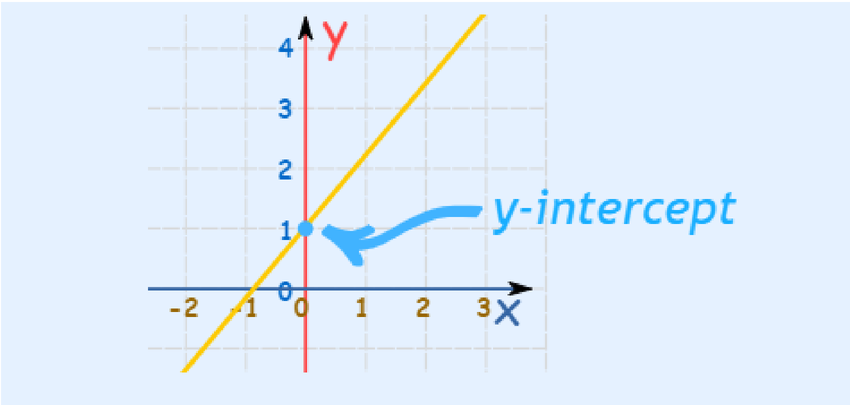
<u>**基础知识**</u>

What is a Slope?

Slope is very important in the equation because it tells you how much you can expect Y to change as X increases. It is denoted by m in the formula $y = mx + c$.

$$m = \frac{\sum_{i-1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

What is Intercept?



In the above diagram the line crosses the y axis at $\mathbf{y} = \mathbf{1}$

Y-intercept is the place where the regression line $y = mx + b$ crosses the y-axis (where x = 0), and is denoted by b.

<u>**Linear Regression**</u>

What is Regression 回归?

Purpose of Regression – Prediction (Predict weather, Predict stock flow). 回归之所以能预测是因为他通过历史数据，摸透了“套路”，然后通过这个套路来预测未来的结果。

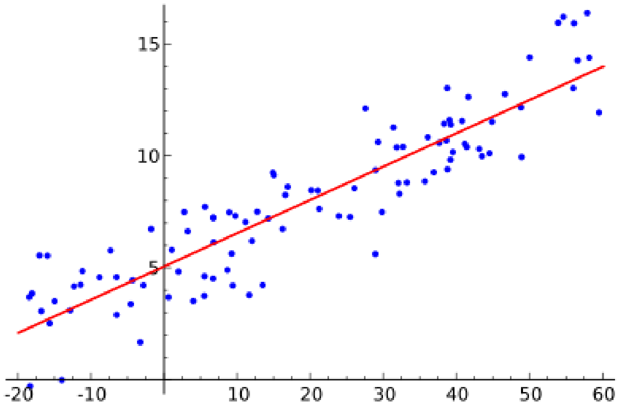
What is Linear 线性?

Example of Linear –「房子」越大，「租金」就越高 。

可是不是每个东西都是线性 – 充电越久，瓦数不会越高。

What is Linear Regression 线性回归?

如果两个或者更过的 Variable 之间存在 Linear Relationship，那么可以通过历史数据来发现 Variable 之间的关系，建立一个模型来预测未来的 Variable。



A Method to predict a Target Variable by fitting the Best Linear Relationship between the dependent and independent variable.

Types of Linear Regression

I. **Simple Linear Regression** – Only one input feature ($y = wx + b$)
II. **Multiple Linear Regression** – Multiple input features ($y = b + \sum w_i x_i$

Simple Linear Regression 单一线性回归

Simple Linear Regression helps to find the linear relationship between two continuous variables ($weight\ w^*$ and $bias\ b^*$), one independent ($feature\ x^*$) and one dependent features $output\ y$.

$$y = b + wx$$

$$x - Feature\ (Attribute\ of\ input\ x)$$

$$y - 给出的预测值\ \ (Dependent\ Variable)$$

$$w - Weight\ (Coefficient)$$

$$b - Bias\ (Constant)$$

Multiple Linear Regression 多元线性回归

Multiple Linear Regression is used to explain the relationship between one continuous dependent variable and two or more independent variables.

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

$$x_i - Features\ (同一个Data的不同Feature)$$

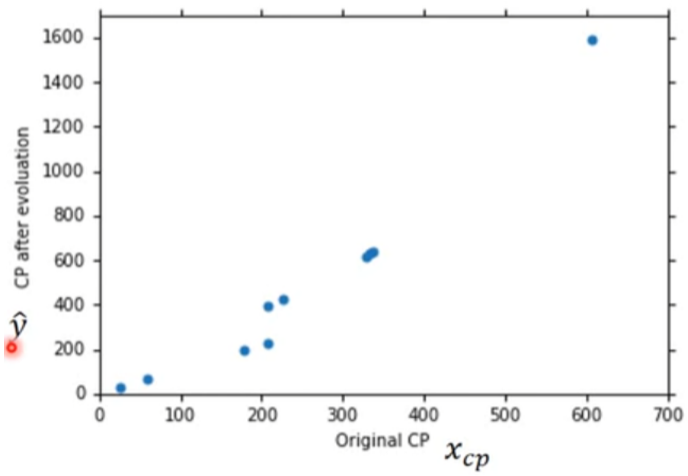
$$y - y - 给出的预测值\ \ (Dependent\ Variable)$$

$$w_i - Weight\ for\ different\ Features$$

$$b - Bias\ (Constant)$$

Model Training Performance

要训练 Linear Regression 之前，需要准备一组 Dataset。这组 Dataset 的数据有$Input\ Feature\ x^n$和 $Ground\ Truth\ \hat{y}^n$。



Plot 出来的数据大概长这样，x-axis 是$Input\ Feature\ x^n$ 而 y-axis 是$Ground\ Truth\ \hat{y}^n$。有了一组 Dataset 之后，就可以开始训练 Linear Regression。

首先先随机设置一个$weight$ 和 $bias$ 的值。这时候就使用这组 Dataset 来跑这个 Function。跑了之后，使用 Loss Function 来知道参数 ($weight$ & $bias$) 有没有调好。

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

算出来的$L(f)$的值越大，代表这个模型的预测很糟糕，没办法很好的给出正确的预测。

争对算出的$L(f)$，需要找出最好的$weight和bias$ 也就是 Loss Function 的值越小越好。

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

$$\theta^w = w^*, b^* = \arg\min_{w,b} L(w,b) = \arg\min_{w,b} \sum_{i=1}^{n} (\hat{y}^i - (b + w * x^n))^2$$

使用 Gradient Descent 就能通过不断的 Update 来找到最好的值。Gradient Descent 的厉害之处就是只要$L(f)$ 是可微分 (Partial Derivative)，Gradient Descent 都能使用来找到比较好的参数。

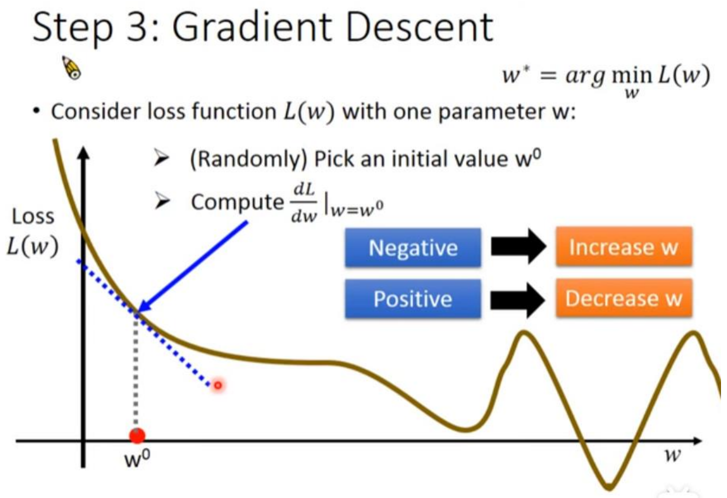
要使用 Gradient Descent，首先先要去$L(f)$的微分值$\frac{\partial L}{\partial w_i}$。

$$Mean\ Square\ Error \to L(w,b) = \left(\hat{y} - (b + wx)\right)^2$$

$$\frac{\partial L}{\partial w} = 2 * \left(\hat{y} - (b + wx)\right) * (-x)$$

$$\frac{\partial L}{\partial b} = 2 * \left(\hat{y} - (b + wx)\right) * (-1)$$

如果算出来的$\frac{\partial L}{\partial w_i}$**是 Positive**，这时候就需要**减少 weight** 的值。如果算出来的$\frac{\partial L}{\partial w_i}$**是 Negative**，这时候就需要**增加 weight** 的值。

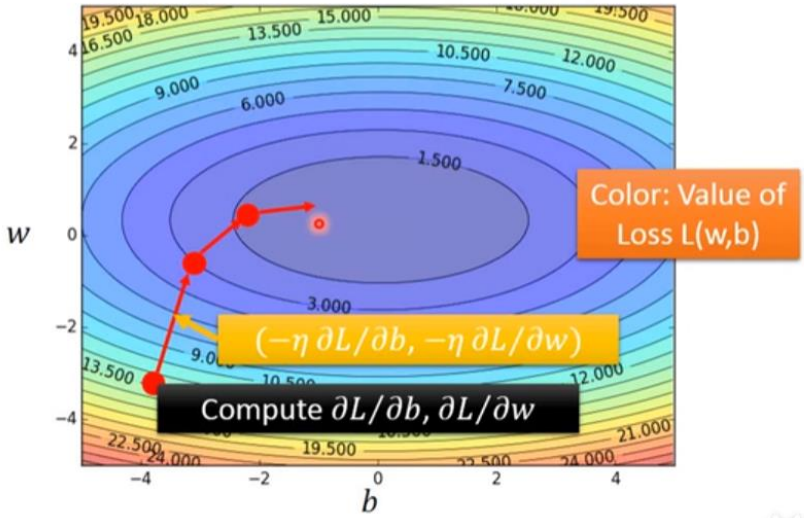


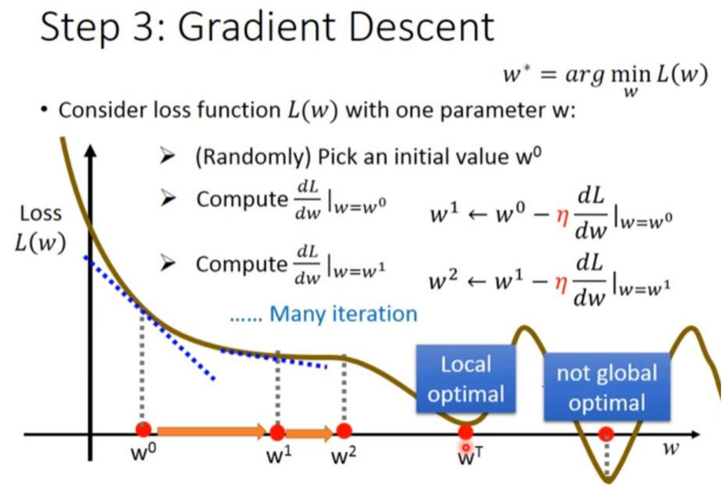
$$Gradient\ Descent \to \theta^N = \theta^{N-1} - \eta\nabla L(\theta^{N-1})$$

$$\eta \to Learning\ Rate$$

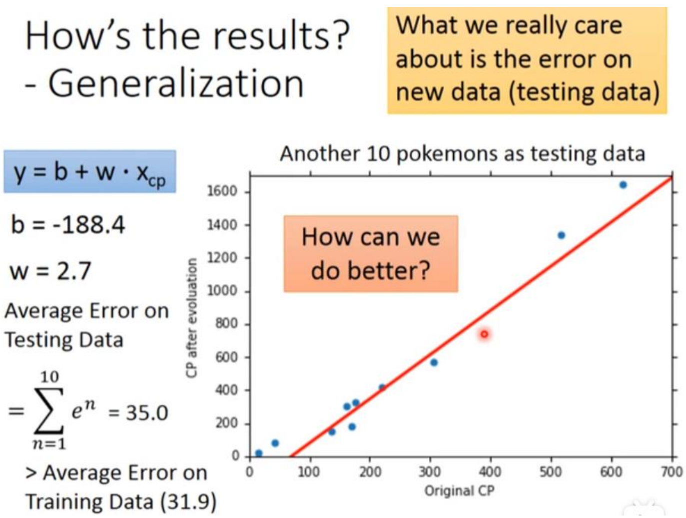
$$\nabla L(\theta^{N-1}) \to Partial\ Derivative\ of\ Loss\ Function = \begin{bmatrix} \frac{\partial L}{\partial w} \\ \frac{\partial L}{\partial b} \end{bmatrix}$$

Learning Rate 是用来控制 Update 的速度，Learning Rate 越大 Update 的值就越大。刚好的 Learning Rate 值能够让模型更好的 Converge。





反复计算$L(f)$ 然后 Update 参数直到模型找到 Local/Global Minimum Point。但是在 Linear Regression 里，不需要担心模型找到的是 Local Minimum Point 而不是 Global Minimum Point。**因为使用 MSE 的 Loss Function 的话，它是 Convex 的所以它不会有 Local Minimum**，它只会有一个 Minimum Point。



训练之后，使用 Testing Data 来测试并计算 Average Error，Error 越低代表模型训练的越好。

但是不是每个问题都能用 Linear 的 Function 来解决，当 Linear 的 Function 训练后不能得到很好的效果，可以使用别种 Function 来试看解决。在这里，w_1和w_2是不同的，但是 x 是同样的。

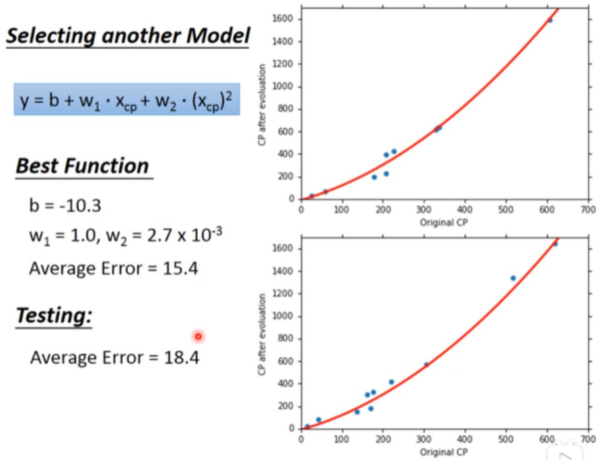
$$y = b + w_1 x + w_2 (x)^2$$

$$x - Feature\ (Attribute\ of\ input\ x)$$

$$y - \text{给出的预测值}\ (Dependent\ Variable)$$

$$w - Weight\ (Coefficient)$$

$$b - Bias\ (Constant)$$



计算后，可以看出这个 Function 更 Fit 这组 Data。算出来的 Average Error 也比 Linear Function 来的小。可以继续测试使用另一个维度更高的 Function。

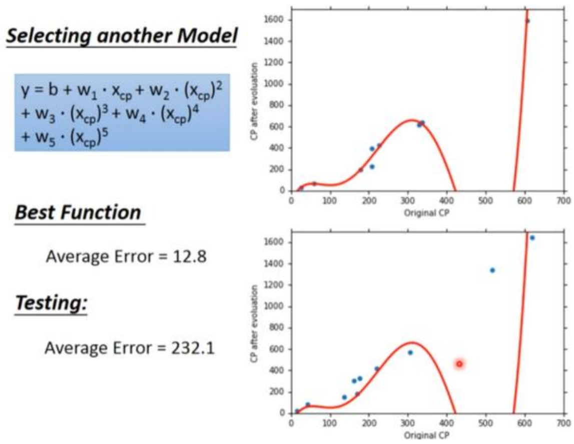
$$y = b + w_1 x + w_2 (x)^2 + w_3 (x)^3 + w_4 (x)^4 + \cdots + w_n (x)^n$$

$$x - Feature\ (Attribute\ of\ input\ x)$$

$$y - \text{给出的预测值}\ (Dependent\ Variable)$$

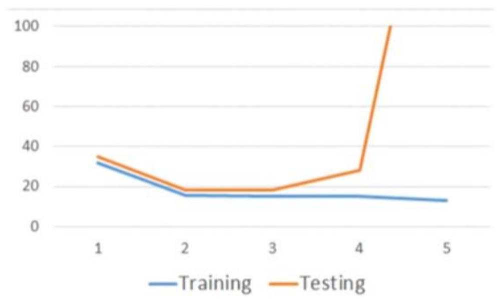
$$w - Weight\ (Coefficient)$$

$$b - Bias\ (Constant)$$



维度越高不代表 Model 在 Testing Data 的表现越好，需要找到适合的 Function 来训练才能得到最好的效果。

Model Selection



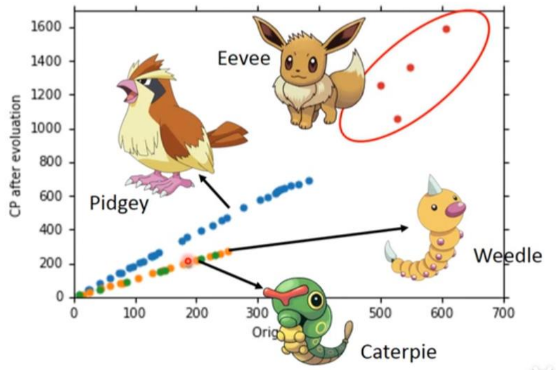
	Training	Testing
1	31.9	35.0
2	15.4	18.4
3	15.3	18.1
4	14.9	28.2
5	12.8	232.1

越复杂，维度越高的 Model 可以在 Training Data 取得很小的 Error，但是在 Testing Data 就不是这么一回事，原因是因为这个模型可能已经 Overfit 了。

Example

在训练模型的时候，需要考虑 Data 之间的关系，比如说可能不同总类会影响计算的轨迹。

What are the hidden factors?



上图是一个列子，不同物种的预测值是不一样的，这时候就需要对每一个物种进行训练。

Back to step 1: Redesign the Model

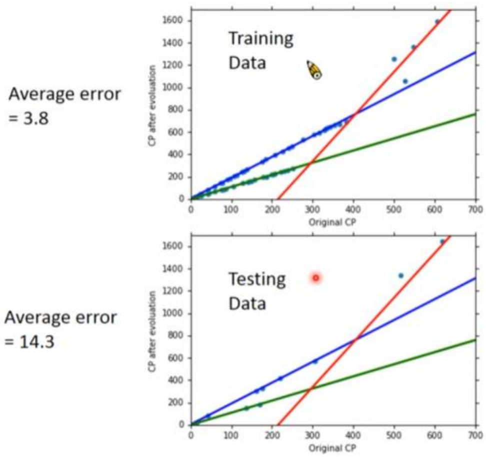
$$y = b + \sum w_i x_i$$

Linear model?

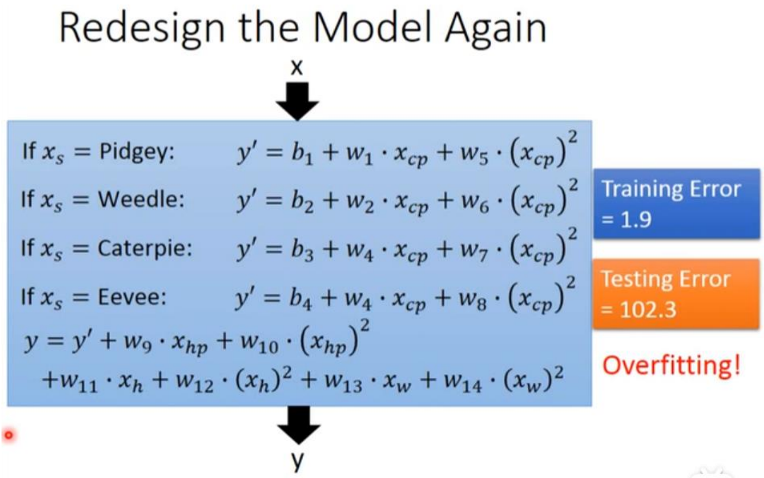
$$
\begin{aligned}
y = & \ b_1 \cdot \delta(x_s = \text{Pidgey}) \\
& + w_1 \cdot \delta(x_s = \text{Pidgey}) x_{cp} \\
& + b_2 \cdot \delta(x_s = \text{Weedle}) \\
& + w_2 \cdot \delta(x_s = \text{Weedle}) x_{cp} \\
& + b_3 \cdot \delta(x_s = \text{Caterpie}) \\
& + w_3 \cdot \delta(x_s = \text{Caterpie}) x_{cp} \\
& + b_4 \cdot \delta(x_s = \text{Eevee}) \\
& + w_4 \cdot \delta(x_s = \text{Eevee}) x_{cp}
\end{aligned}
$$

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

在定义 Function 的时候根据不同物种需要 Update 不同的参数，这时候可以使用 δ 来进行晒寻，是对应的物种 δ 就等于 1 其他都是 0。这时候就这剩下该物种的 Function。



以这样的方法训练，可以更好的对不同物种的数据做出预测，训练和测试的 Error 都明显降低。除了考虑物种，还能看看还有什么 Hidden Factors 能影响训练效果，让训练效果更好。



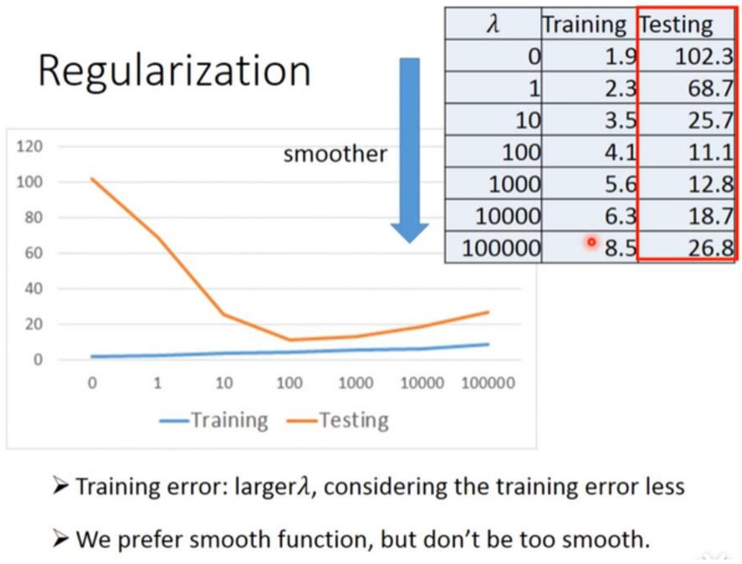
这时候可以考虑更多的 Hidden Factors。但是随着越多的 Features，模型训练后很容易就 Overfit 了，这时候就需要找出有用的 Features，拿掉没用的 Features。除此之外，还能使用 Regularization 来让模型不这么容易 Overfit。

Regularization

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

$$\lambda \sum (w_i)^2 \rightarrow w_i\textit{的值越小，做了Square之后就会更小，算出的Loss也会变小}$$

Regularization 能让那个 Function 更加的 Smooth (越小的 w_i，Function 就会越平滑)，就是说 Input 的改变不会给 Output 带来太大的变化。



加入了 Regularization 之后，Training Error 会随着 λ 的增加，而变得越来越大。λ 的值越大代表考虑 Ground Truth 和 Predicted Value 的权重比较低，而考虑让 Function 更加平滑的权重比较大。但是如果 Function 太 Smooth，Testing Error 也是会变得糟糕。

Regularization 是不需考虑 Bias 的值，这是因为 Bias 的值不会影响 Function 的平滑度，只有 $\boldsymbol{weight\ w_i}$ 才会影响 Function 的平滑度。

<u>Logistic Regression</u>

Logistic Regression (逻辑回归) 是分类模型，常用于 Binary Classification。Idea of Logistic Regression is to find a relationship between features and probability of particular outcome.

Types of Logistic Regression:

I. **<u>Binomial Logistic Regression</u>** – Dependent Variable has **only two 2 possible outcomes/classes**
II. **<u>Multinomial Logistic Regression</u>** – Dependent variable has **two or more possible outcomes/classes without ordering** (Good, Great and Bad)
III. **<u>Ordinal Logistic Regression</u>** – Dependent variable has **two or more possible outcomes/classes with ordering** (Star Rating from 1 to 5)

Logistic Regression in Machin Learning Training

Logistic Regression 其实就是找一个 Posterior Probability。在给出一组 features 之后得到 Class 1 的几率是多少。

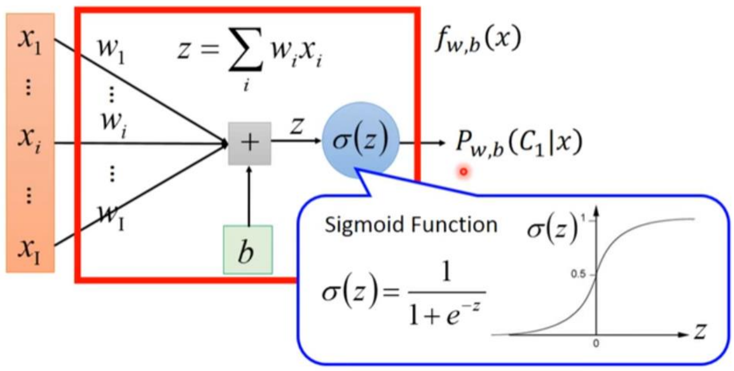
$$P_{w,b}(C_1|X) \rightarrow Posterior\ Probability\ (Given\ X\ 是\ C_1 的几率是多少)$$

$$P_{w,b}(C_1|X) \geq 0.5 \rightarrow \begin{cases} Output\ Class\ 1 \\ Otherwise, Output\ Class\ 2 \end{cases}$$

在 Binomial Logistic Regression 里，算出来的 Posterior Probability 大于 0.5 的话就定义为 Class 1 如果小于 0.5 就定义为 Class 2。这个 0.5 的 Threshold 可以自己设置。

$$P_{w,b}(C_1|X) = f_{w,b}(x) = \sigma(z) = \sigma\left(\sum_i W_i X_i + b\right) \rightarrow Output\ between\ 0\ and\ 1$$

$$sigmoid\ functoin \rightarrow \sigma(z) = \frac{1}{1+e^{-z}}$$



上图简单介绍 Logistic Regression 的 Overview。放入$input\ X_i$ 乘上 $weight\ w_i$ 在加上 $bias\ b$ 就能算出$output\ z$，然后再把$output\ z$ 放入$sigmoid\ function\ \sigma(z)$ 就能得到想要找的 Posterior Probability。因为 Sigmoid Function 的关系所以算出来的的值会是在 0 到 1 之间。

Model Training Performance

Training Data					
	x^1	x^2	x^3	……	x^N
	C_1	C_1	C_2		C_1

假设上图是一组训练 Data，$input \to X^N, ground\ truth\ output \to C_1/C_2$。现在的目的是寻找一组 $weight\ w^*$和$Bias\ b^*$ 是可以最大化计算出正确的 Class 的机率。

$$L(w,b) = f_{w,b}(x^1) * f_{w,b}(x^2)\left(1 - f_{w,b}(x^3)\right) \dots f_{w,b}(X^N)$$

如果$weight\ w^*$和$Bias\ b^*$ 已经找到最好的值，那么算出来的$L(w,b)$将会是最大接近 1 的值。

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

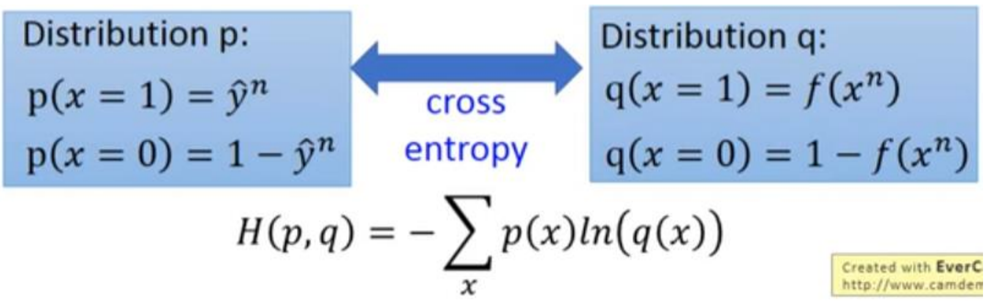
最大化的 Equation 是求 ground truth 和 model output 的相似度，越高代表 model output 与 ground truth 越接近，也表示$weight\ w^*$和$Bias\ b^*$越贴合这组模型的 Data。

最小化的 Equation 是求 ground truth 和 model output 的差别，越低代表 model output 与 ground truth 越接近，也表示$weight\ w^*$和$Bias\ b^*$越贴合这组模型的 Data。而在训练的时候求最小话是比较简单。

$$-\ln L(w,b) \to -\ln f_{w,b}(x^N) = -[\hat{Y}^N \ln f(x^N) + \left(1 - \hat{Y}^N\right)\ln(1 - f(x^N)]$$

$$\sum_n -[\hat{y}^N \ln f_{w,b}(x^N) + (1 - \hat{y}^N)\ln(1 - f_{w,b}(x^N)]$$

上面的计算方式就是两个 Bernoulli Distribution 的 Cross Entropy Loss 的计算方式。



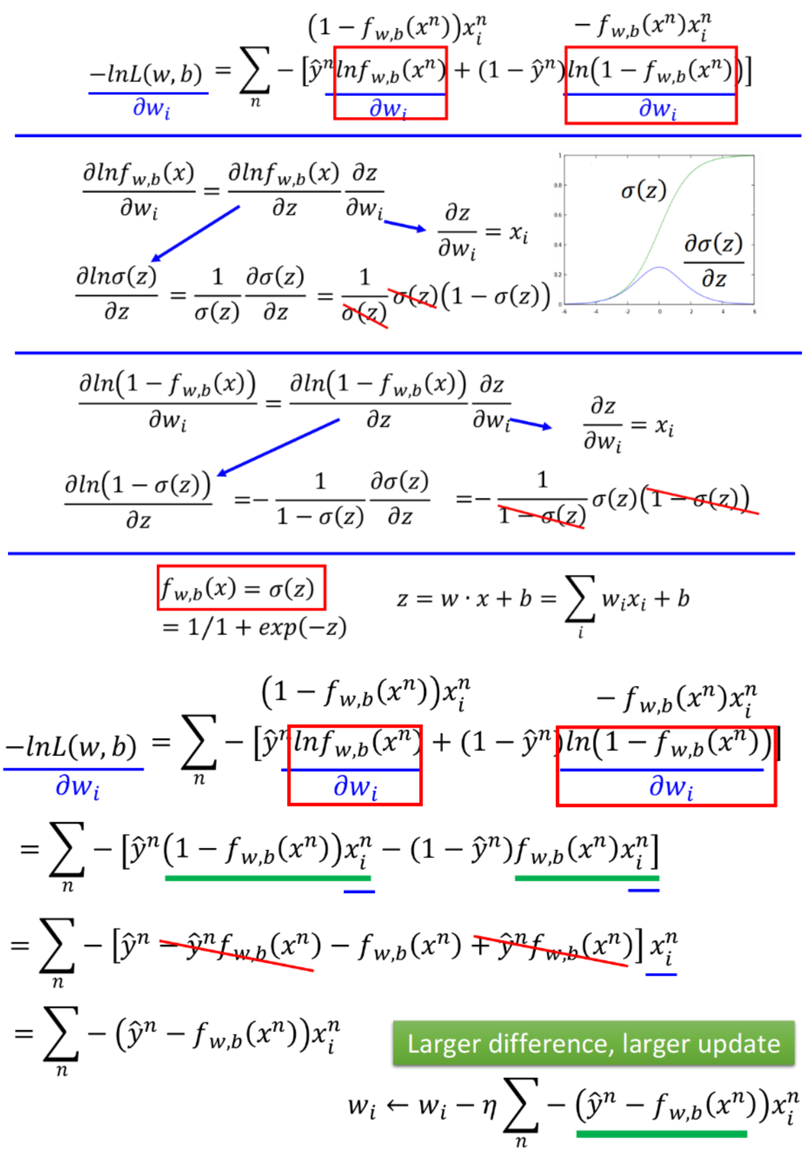
在这一个步骤，我们希望计算出的 Loss 是越接近 0 越好。

Update Parameters

在计算出 Loss 之后，需要想办法 Update $weight\ w^*$和$Bias\ b^*$ 来让这个模型 Predict 的 output 更贴合 Training Data 的 Ground Truth。在这个阶段，我们就能使用 Gradient Descent 来找到最适合的$weight\ w^*$和$Bias\ b^*$的值。

$$Gradient\ Descent \rightarrow \theta^N = \theta^{N-1} - \eta \nabla L(\theta^{N-1})$$

首先需要算出 Loss Function 的 Partial Derivatives。 要 Update $weight\ w^*$ 就对 Loss Function 做 $weight\ w^*$的 Partial Derivative。要 Update $Bias\ b^*$ 就对 Loss Function 做 $Bias\ b^*$的 Partial Derivative。



算出了 Loss Function 的 Partial Derivatives 就能使用 Gradient Descent 的 Formula 来 Update $weight\ w^*$和$Bias\ b^*$。算出来的 Partial Derivative 其实很简单，就是找出 Ground Truth 和模型 Predicted Output 的差别再乘上 Input。如果差别越大，就要 Update 的越多。

为什么 Logistic Regression 不用 Square Error

当训练 Logistic Regression 使用 Square Error 的时候，一样定义初始的 Function。

$$f_{w,b}(x) = \sigma(\sum_i w_i x_i + b)$$

$$L(f) = \frac{1}{2}\sum_n \left(f_{w,b}(x^n) - \hat{y}^n\right)^2$$

计算了 Loss Function 之后一样使用 Gradient Descent 来 Update $weight\ w^*$和$Bias\ b^*$。这时候就要对 Loss Function 做 Partial Derivative。

$$\frac{\partial\left(f_{w,b}(x) - \hat{y}\right)^2}{\partial w_i} = 2\left(f_{w,b}(x) - \hat{y}\right) * \frac{\partial\left(f_{w,b} - \hat{y}\right)}{\partial w_i}$$

$$\frac{\partial\left(f_{w,b} - \hat{y}\right)}{\partial w_i} = \frac{\partial\left(f_{w,b} - \hat{y}\right)}{\partial z} * \frac{\partial z}{\partial w_i}$$

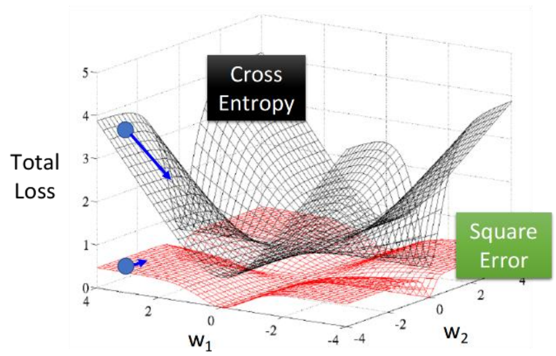
$$\frac{\partial\left(f_{w,b}(x) - \hat{y}\right)^2}{\partial w_i} = 2\left(f_{w,b}(x) - \hat{y}\right) * \frac{\partial\left(f_{w,b} - \hat{y}\right)}{\partial z} * \frac{\partial z}{\partial w_i}$$

$$\frac{\partial\left(f_{w,b}(x) - \hat{y}\right)^2}{\partial w_i} = 2\left(f_{w,b}(x) - \hat{y}\right) * f_{w,b}(x)\left(1 - f_{w,b}(x)\right)x_i$$

当$\hat{y}^n = 1$ 的时候，如果预测出来的$f_{w,b}(x^n) \approx$ 0(Far from Target)，那么$\frac{\partial L}{\partial w_i}$会等于 0。

当$\hat{y}^n = 0$ 的时候，如果预测出来的$f_{w,b}(x^n) \approx$ 1(Far from Target)，那么$\frac{\partial L}{\partial w_i}$也会等于 0。

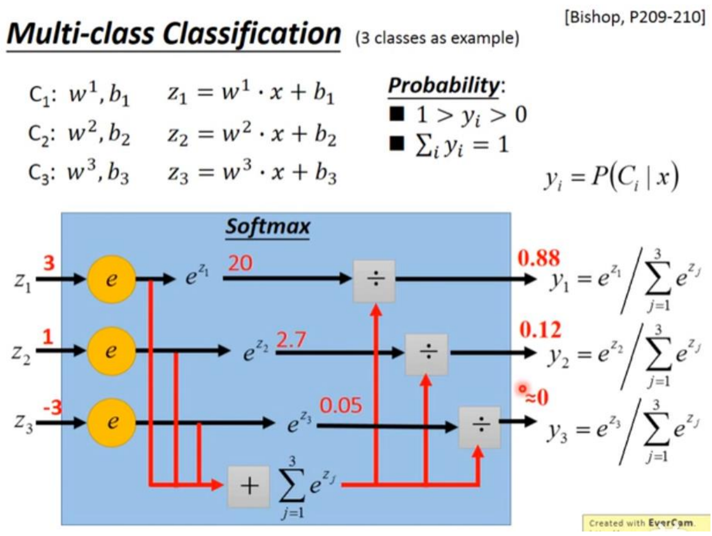
Cross Entropy v.s. Square Error



从上图能看到 Cross Entropy Loss 跟 Square Error 的对比。距离目标越远，Cross Entropy Loss 的微分值就会越大。在 Square Error 的情况，距离目标远的时候，Square Error 算出来的微分值是非常小，导致 Update Parameters 的速度非常慢。如果为了解决 Update 慢的问题而提高 Learning Rate，会导致的问题是当已经很靠近 Minimum Point 的时候不能 Converge。

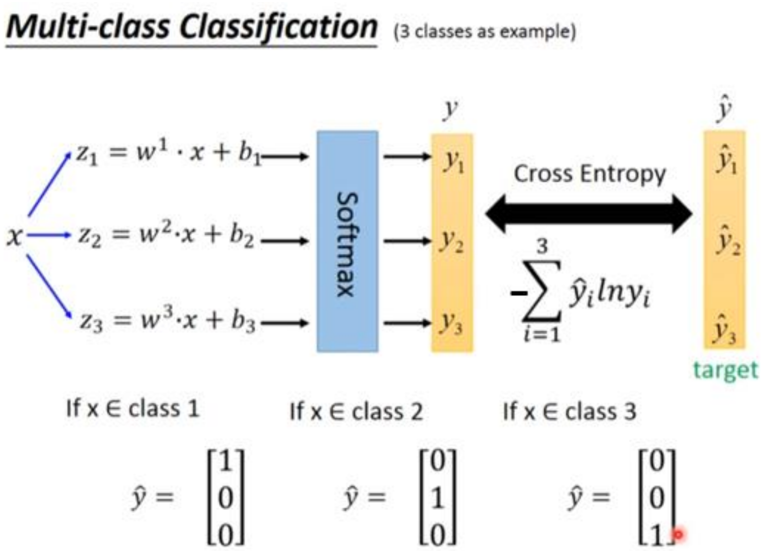
Multinomial Logistic Regression

在训练 Multinomial Logistic Regression 的时候，其实和 Binomial Logistic Regression 没有太大的区别。在 Binomial Logistic Regression 里，使用了 Sigmoid Function，而在 Multinomial Logistic Regression 里，使用了 SoftMax Function。



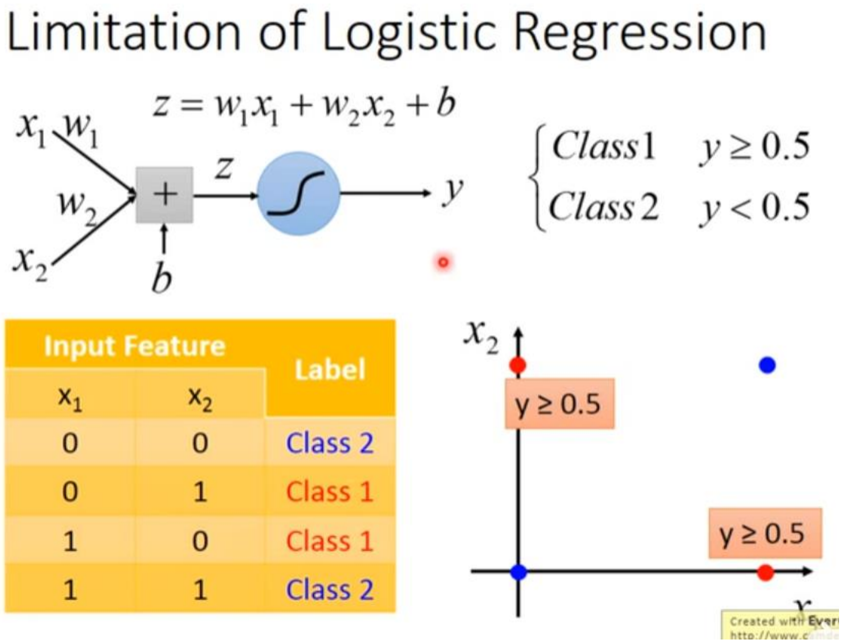
$$Softmax \rightarrow y_n = \frac{e^{z_n}}{\sum_{j=1}^{3} e^{z_j}}$$

SoftMax 算出来的其实就是 Probability，给入$input\ x$ 后，每一个 Class 发生的几率。算出 SoftMax 之后，就使用 Cross Entropy Loss 来计算 Predicted Probability 和 Actual Class 之间的差距。然后再使用 Gradient Descent 来 Update 参数。

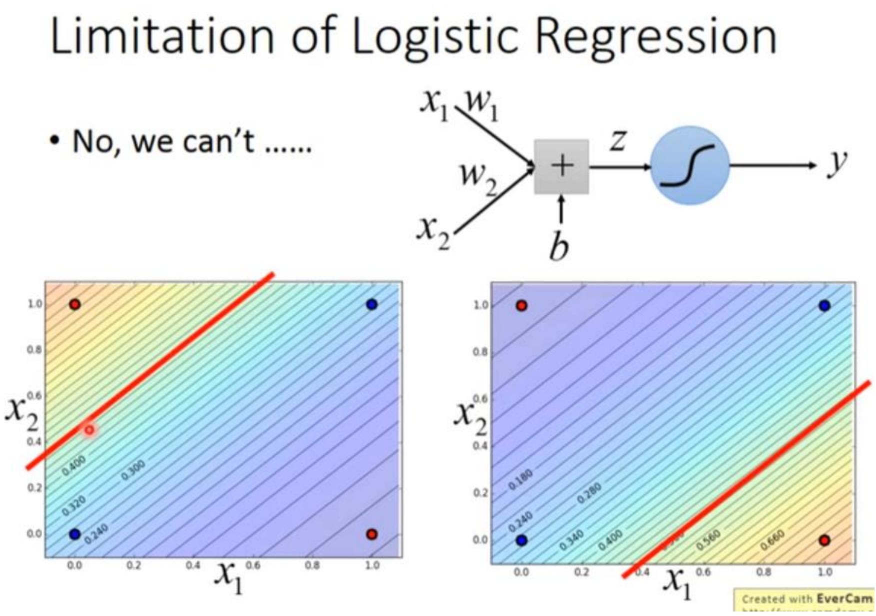


训练 Multi-Class Classification 的时候，Ground Truth 是 1 的时候代表这个 Data 是属于这个 Class，不属于这个 Class 的 Ground Truth 都是 0。不使用 1，2，3 来定义不同的 Class Ground Truth 是为了不让 Data 之间存在关系，Ground Truth 1 靠近 2，而 2 靠近 3，1 和 3 之间比较远。

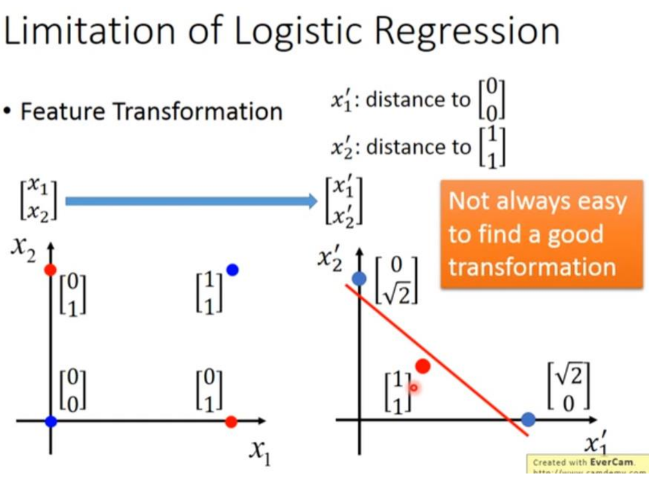
Limitation of Logistic Regression



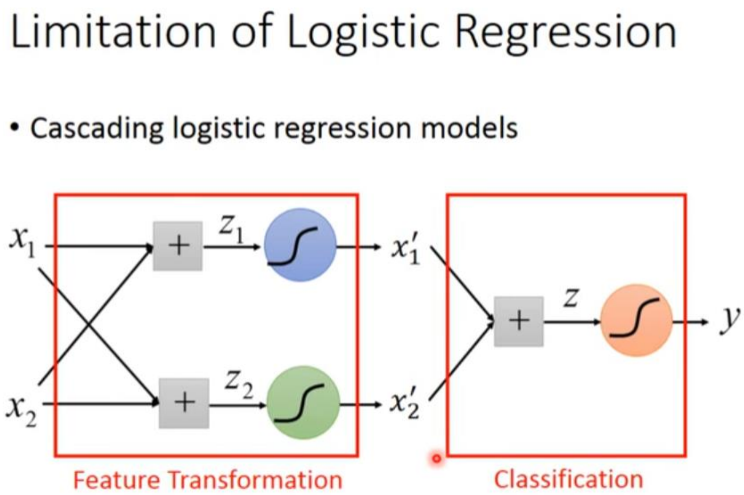
Logistic Regression 其实就是在画出 Boundary 来进行分类 (数据在线上是一个 Class，在线下是另一个 Class)。当遇到以上这种情况，Logistic Regression 的方法没办法画出 Boundary 来进行分类。



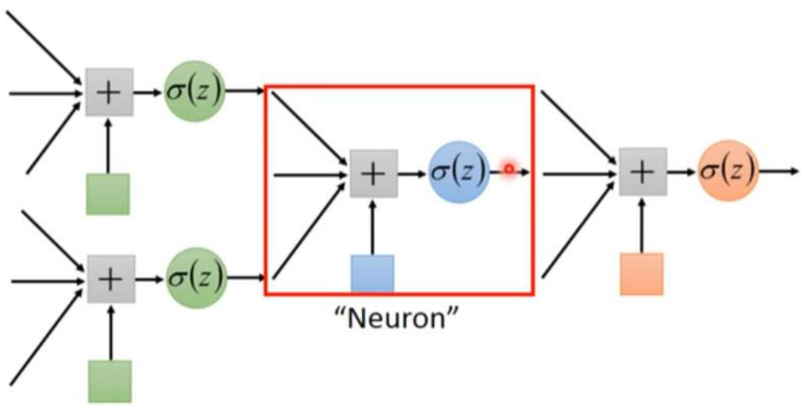
不管怎么调试 Weight 和 Bias，都不能把以上的这个 Data 分类好。这时候有一个方法可以解决这个问题，就是使用 **Feature Transformation** 对数据进行转化，来让 Logistic Regression 有办法对这组数据做分类。



做了 Feature Transformation 之后，这时候，Logistic Regression 就有办法画出 Boundary 来对这组数据做分类。但是麻烦的是，不知道要使用什么 Feature Transformation 的方法，希望这个 **Feature Transformation 可以让机器来执行**。



这时候如果使用两个 Logistic Regression 叠加和计算，就能实现 Feature Transformation 的任务，然后再把x_1'和x_2'带入另一个 Logistic Regression 就能完成分类。



叠加在一起的 Logistic Regression 也被称作为 Neural Network。而在 Neural Network 里，一个 Logistic Regression 也被称为 Neuron。

对 Cross Entropy Loss 最微分

$$\frac{\partial \sigma(z)}{\partial z} = \sigma(z)\big(1-\sigma(z)\big)$$

$$Cross\ Entropy\ Loss = -\sum \hat{y}\ln y + (1-\hat{y})\ln(1-y)$$

$$Cross\ Entropy\ Loss = -\sum \hat{y}\ln \sigma(z) + (1-\hat{y})\ln\big(1-\sigma(z)\big)$$

$$\frac{\partial C}{\partial z} = \frac{\partial}{\partial z}\left(-\sum \hat{y}\ln y + (1-\hat{y})\ln(1-y)\right)$$

$$\frac{\partial \hat{y}\ln \sigma(z)}{\partial z} = \hat{y}\left(\frac{1}{\sigma(z)} * \sigma(z)\big(1-\sigma(z)\big)\right)$$

$$\frac{\partial \hat{y}\ln \sigma(z)}{\partial z} = \hat{y}\big(1-\sigma(z)\big) = \hat{y} - \hat{y}\sigma(z)$$

$$\frac{\partial (1-\hat{y})\ln\big(1-\sigma(z)\big)}{\partial z} = (1-\hat{y})\left(\frac{1}{1-\sigma(z)} * \big(-\sigma(z)\big(1-\sigma(z)\big)\big)\right)$$

$$\frac{\partial (1-\hat{y})\ln\big(1-\sigma(z)\big)}{\partial z} = (1-\hat{y}) * -\sigma(z) = -\sigma(z) + \hat{y}\sigma(z)$$

$$\frac{\partial C}{\partial z} = -\sum \hat{y} - \hat{y}\sigma(z) - \sigma(z) + \hat{y}\sigma(z)$$

$$\boldsymbol{\frac{\partial C}{\partial z} = -\sum \hat{y} - \sigma(z) = \ \sigma(z) - \hat{y}}$$