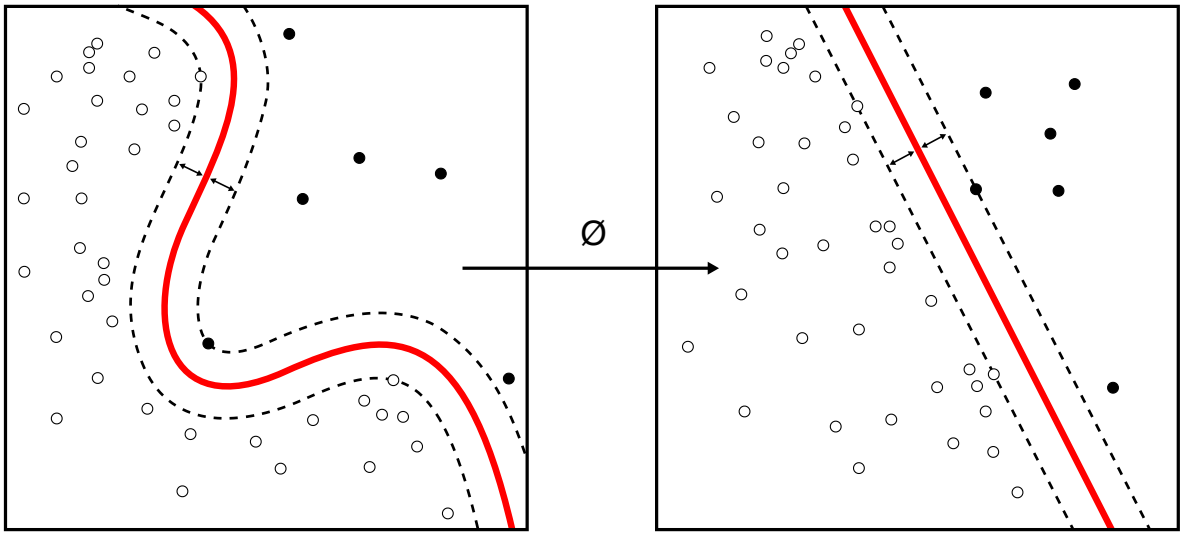
Machine Learning

使用平台：作业Kaggle，免费GPU资源Google Colab

第一节：基本概念

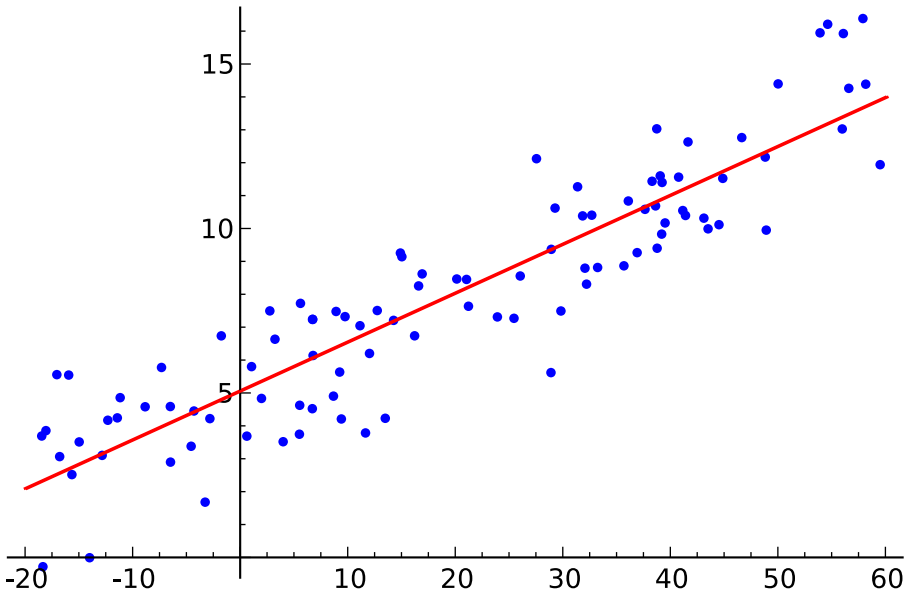
监督学习主要做两类工作，一是分类Classification，二是回归Regression

1. 分类Classification



> 我们试图将输入变量映射到离散类别中。例如给予患有肿瘤的患者，我们必须预测肿瘤是恶性的还是良性的。

2. 回归Regression

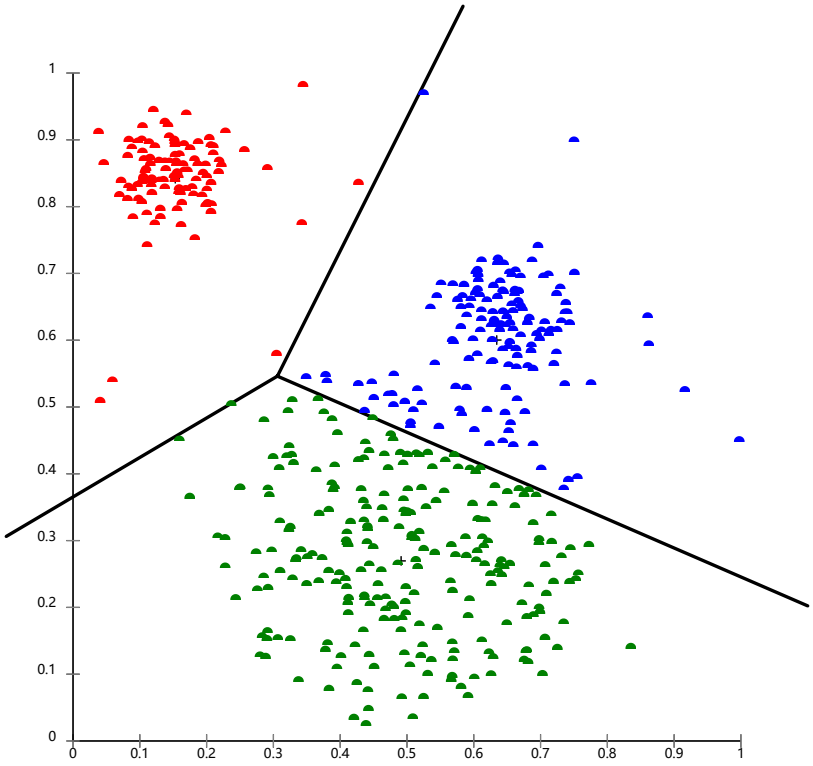


> 我们试图预测连续输出中的结果，这意味着我们试图将输入变量映射到某个连续函数。例如给定一个人的照片，根据照片预测年龄，这就是一个回归的问题。

非监督学习主要工作是“聚类”与“非聚类”

1. 聚类

> 获取1,000,000个不同基因的集合，并找到一种方法将这些基因自动分组成不同变量的相似或相关的组，例如寿命，位置，角色等。



2. 非聚类

> “鸡尾酒会算法”，允许您在混乱的环境中查找结果。 （即在鸡尾酒会上识别来自声音网格的个人声音和音乐）

Machine Learning ≈ Looking for function

（比如给定一个图片/音频/etc.，输出是图中物体/语音/etc.，我们要做的就是找到这个函数）

在“分类”与“回归”以外，机器学习还有**“Structured Learning”**，不只是做选择题

> Create sth with structure(image,document)

猜测函数的流程（以预测某个channel今天的点击量为例）：

1. 设一含有未知参数的函数

   假设$y=b+wx_1$

   y：今日点击量，x1：昨日点击量(feature)

   w(weight)和b(bias)是未知参数（从数据中得出）

   没有乘以x的是bias，乘以x的是weight

2. 定义Loss函数

   L(b,w)输入为b和w ↑

   假设b=0.5k,w=1

   $y=0.5k+x_1$

   用预测的值与真实值差的绝对值作为评估效果

求出每一天的误差e_i

Loss:L=$\frac{1}{N}\sum_n e_n$

e=$|y-\overline{y}|$（MAE　mean absolute error）

3. Optimization(最优化)

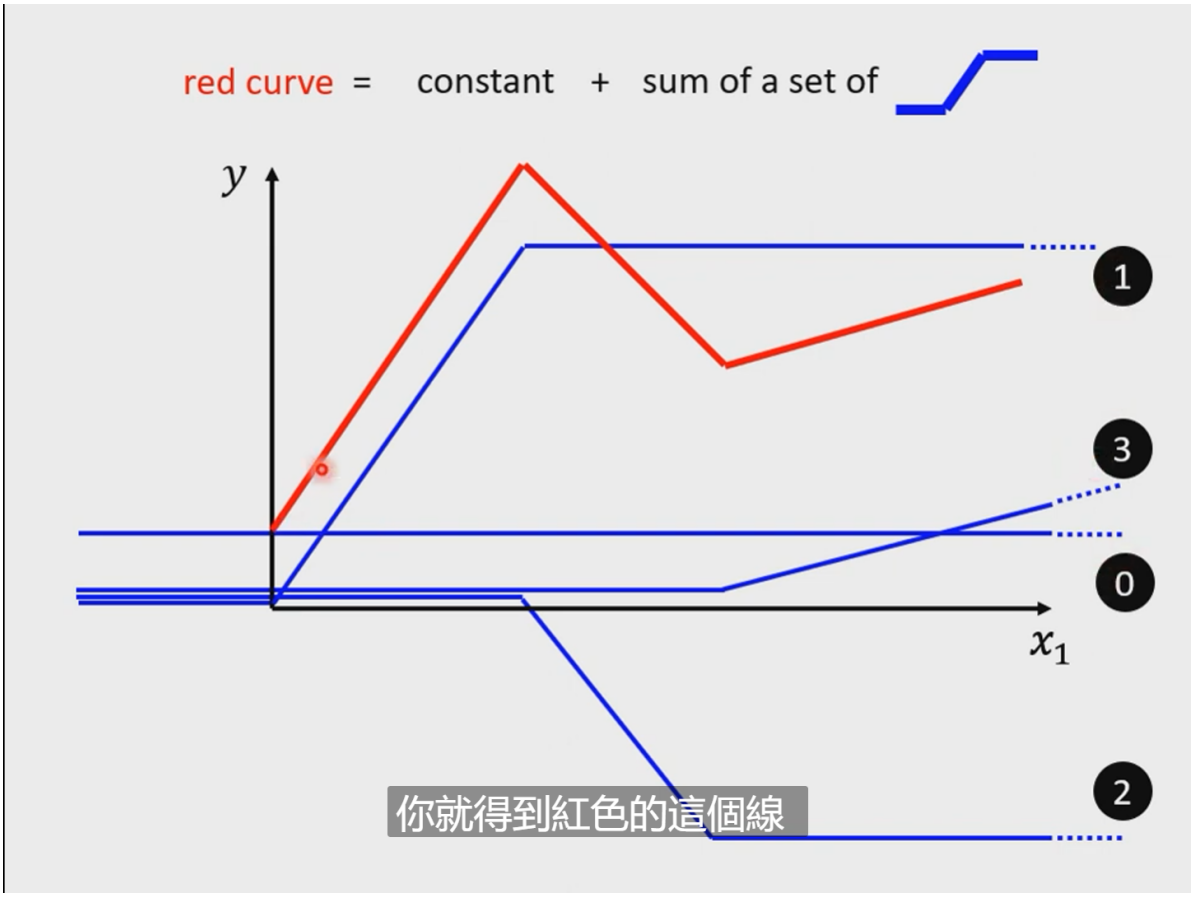
w^*,b^*=arg min(L)

Gradient Descent(梯度下降法)

- 随机选取初始值w^0、b^0
- 计算$\frac{\delta L}{\delta w}|_{w=w^0}$ $\frac{\delta L}{\delta b}|_{w=w^0}$
- 设步长step为$\eta\frac{\delta L}{\delta w}|_{w=w^0,b=b^0}$、$\eta\frac{\delta L}{\delta b}|_{w=w^0,b=b^0}$（斜率越大处步长越大）　η:**learning rate**，是自己设置的
- 类似η这些自己自己设置的就是hyperparameters
- $w^1=w^0-\eta\frac{\delta L}{\delta w}|_{w=w^0,b=b^0}$,$b^1=b^0-\eta\frac{\delta L}{\delta b}|_{w=w^0,b=b^0}$,更新w和b，不断迭代

会有local minima和global minima，gradient descent会有local minima的问题（在实际操作中是个假问题？）**？？？？？？？？**

Linear model有model本身的限制，真实函数图像可能非常复杂，我们可以根据以下这种使用linear叠加的方法计算真实函数图像

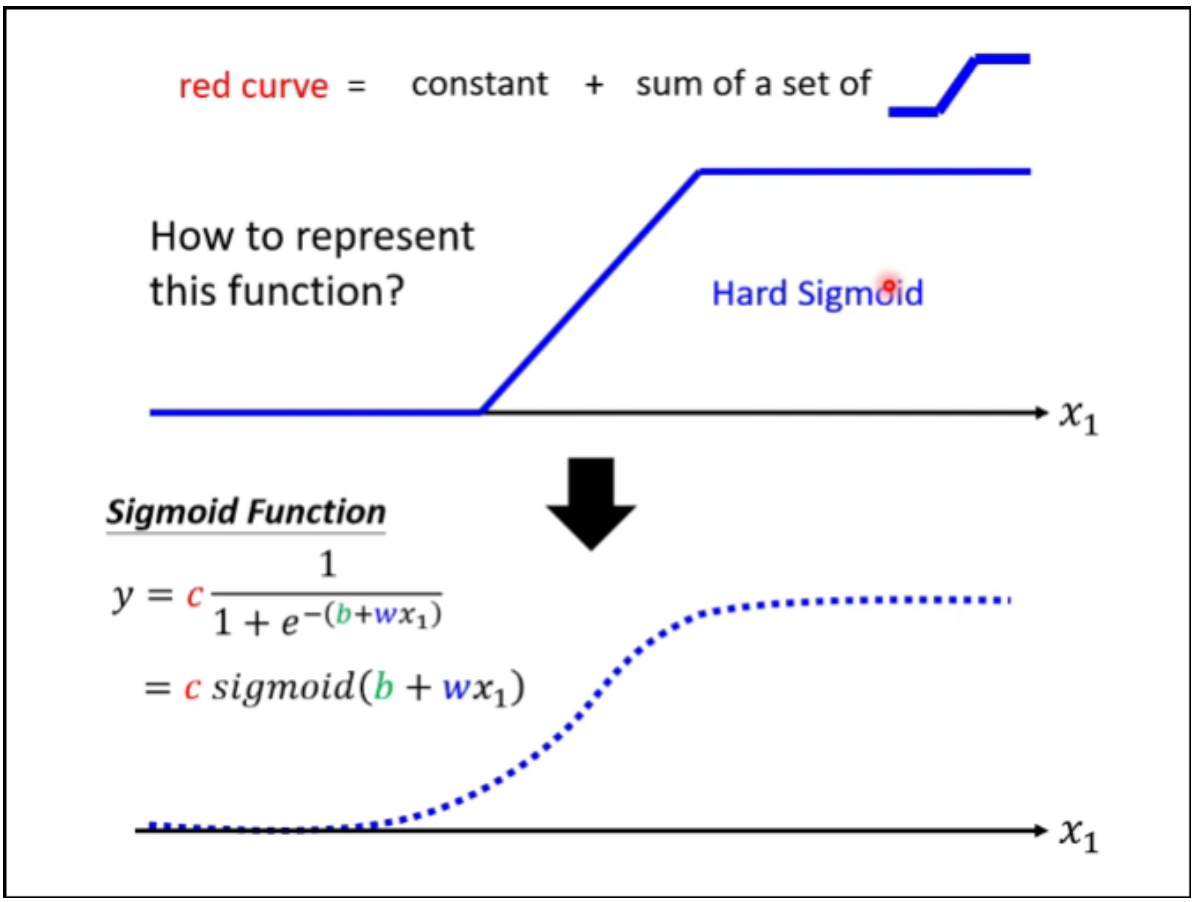


用这种方法可以表示任意函数

可以用Sigmoid函数逼近蓝色的函数

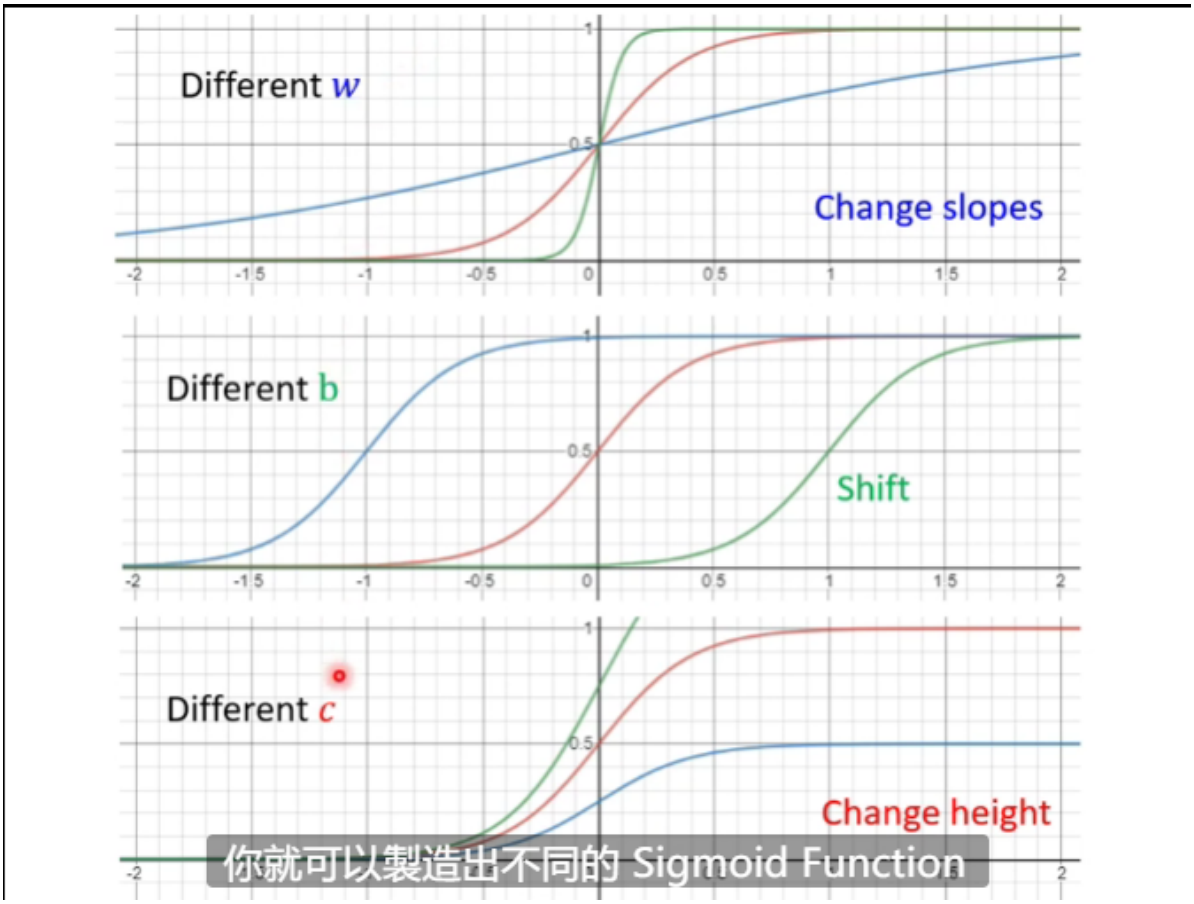
Sigmoid函数为：$y=c\frac{1}{1+e^{-(b+wx_1)}}=c*sigmoid(b+wx_1)$

sigmoid可以翻译为S型的



蓝色函数叫做Hard Sigmoid

为了获得多种多样的函数，我们去调整b和w和c



More Features

New Model: More Features

$$y = b + wx_1$$

$$y = b + \sum_i c_i \, sigmoid(b_i + w_i x_1)$$

$$y = b + \sum_j w_j x_j$$

$$y = b + \sum_i c_i \, sigmoid\left(b_i + \sum_j w_{ij} x_j\right)$$

我們就把 Sigmoid 裡面的東西換掉

计算θ向量

Optimization of New Model

$$\boldsymbol{\theta}^* = arg \min_{\boldsymbol{\theta}} L$$

- (Randomly) Pick initial values $\boldsymbol{\theta}^0$
- Compute gradient $\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^0)$

  $\boldsymbol{\theta}^1 \leftarrow \boldsymbol{\theta}^0 - \eta \boldsymbol{g}$
- Compute gradient $\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^1)$

  $\boldsymbol{\theta}^2 \leftarrow \boldsymbol{\theta}^1 - \eta \boldsymbol{g}$
- Compute gradient $\boldsymbol{g} = \nabla L(\boldsymbol{\theta}^2)$

  $\boldsymbol{\theta}^3 \leftarrow \boldsymbol{\theta}^2 - \eta \boldsymbol{g}$

是 0 向量 是 Zero Vector

Optimization of New Model

$\theta^* = arg \min_\theta L$

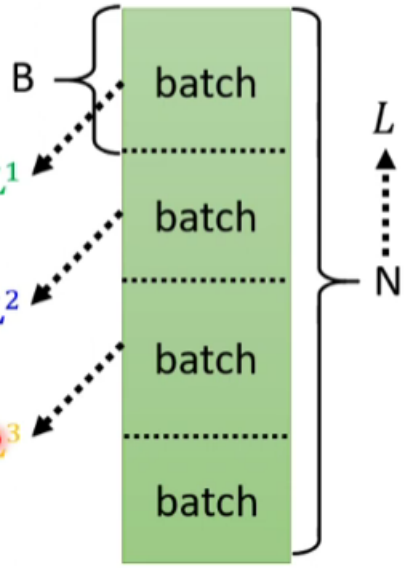
- (Randomly) Pick initial values θ^0
- Compute gradient $g = \nabla L^1(\theta^0)$ L^1

  update $\theta^1 \leftarrow \theta^0 - \eta g$
- Compute gradient $g = \nabla L^2(\theta^1)$ L^2

  update $\theta^2 \leftarrow \theta^1 - \eta g$
- Compute gradient $g = \nabla L^3(\theta^2)$ L^3

  update $\theta^3 \leftarrow \theta^2 - \eta g$

1 **epoch** = see all the batches once



叫做一個 Epoch

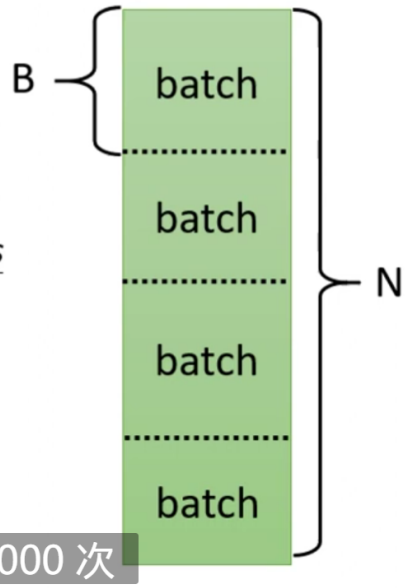
Optimization of New Model

Example 1

- 10,000 examples (N = 10,000)
- Batch size is 10 (B = 10)

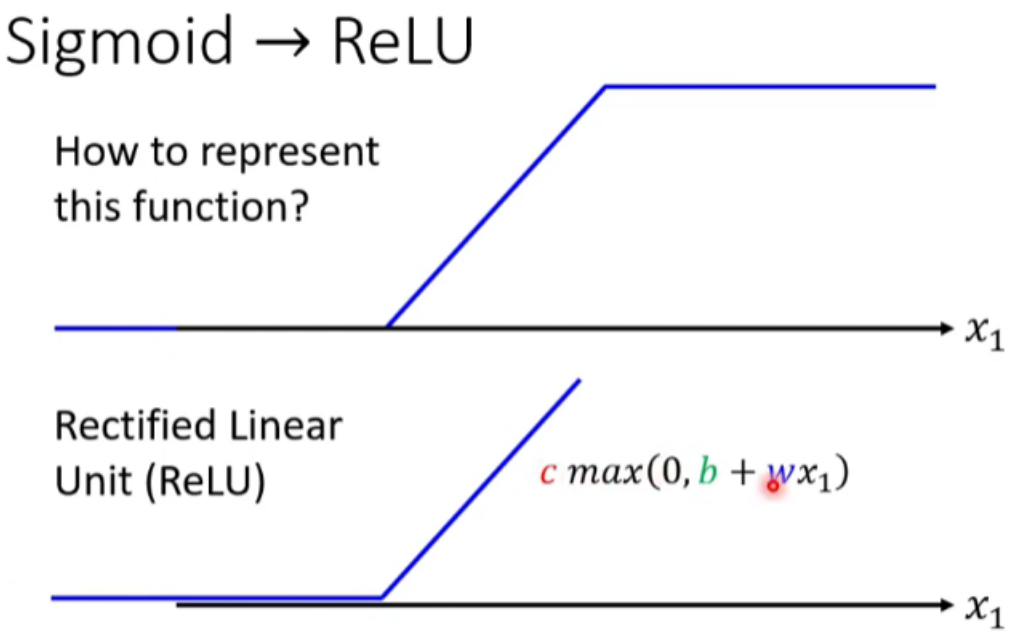
How many update in **1 epoch**?

1,000 updates



你其實已經更新了參數 1000 次

ReLU函数表达式



可以寫成 c max(0, b + wx1)

为什么ReLU比sigmoid好呢？？？