Logistic regression

03-22-2023

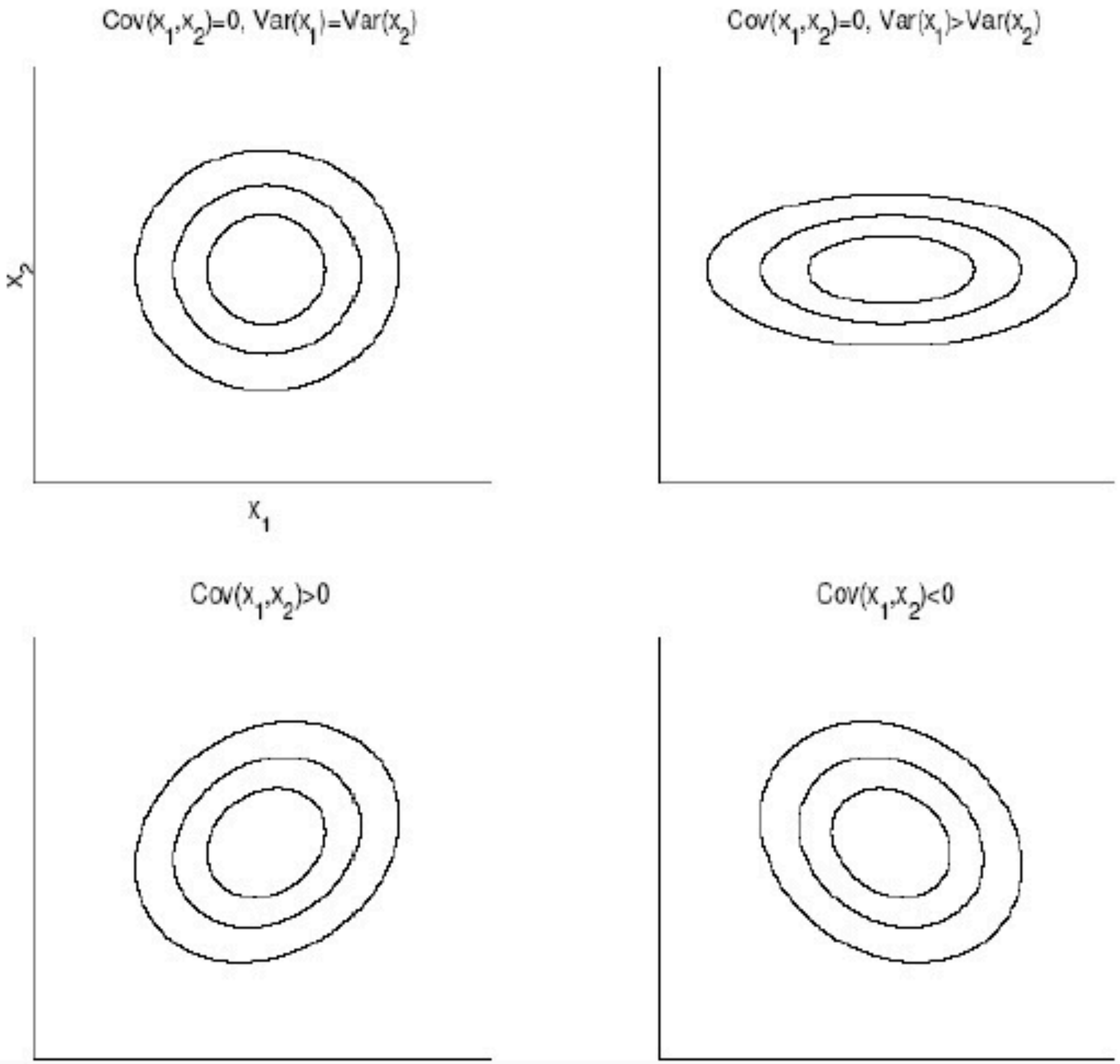
YA TANG YANG

- Bivariate: $d = 2$

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}$$

$$p(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left[-\frac{1}{2(1-\rho^2)}\left(z_1^2 - 2\rho z_1 z_2 + z_2^2\right)\right]$$

$$z_i = (x_i - \mu_i)/\sigma_i$$



Cov(x1,x2)=0, Var(x1)=Var(x2)
Cov(x1,x2)=0, Var(x1)>Var(x2)
x2
x1
Cov(x1,x2)>0
Cov(x1,x2)<0

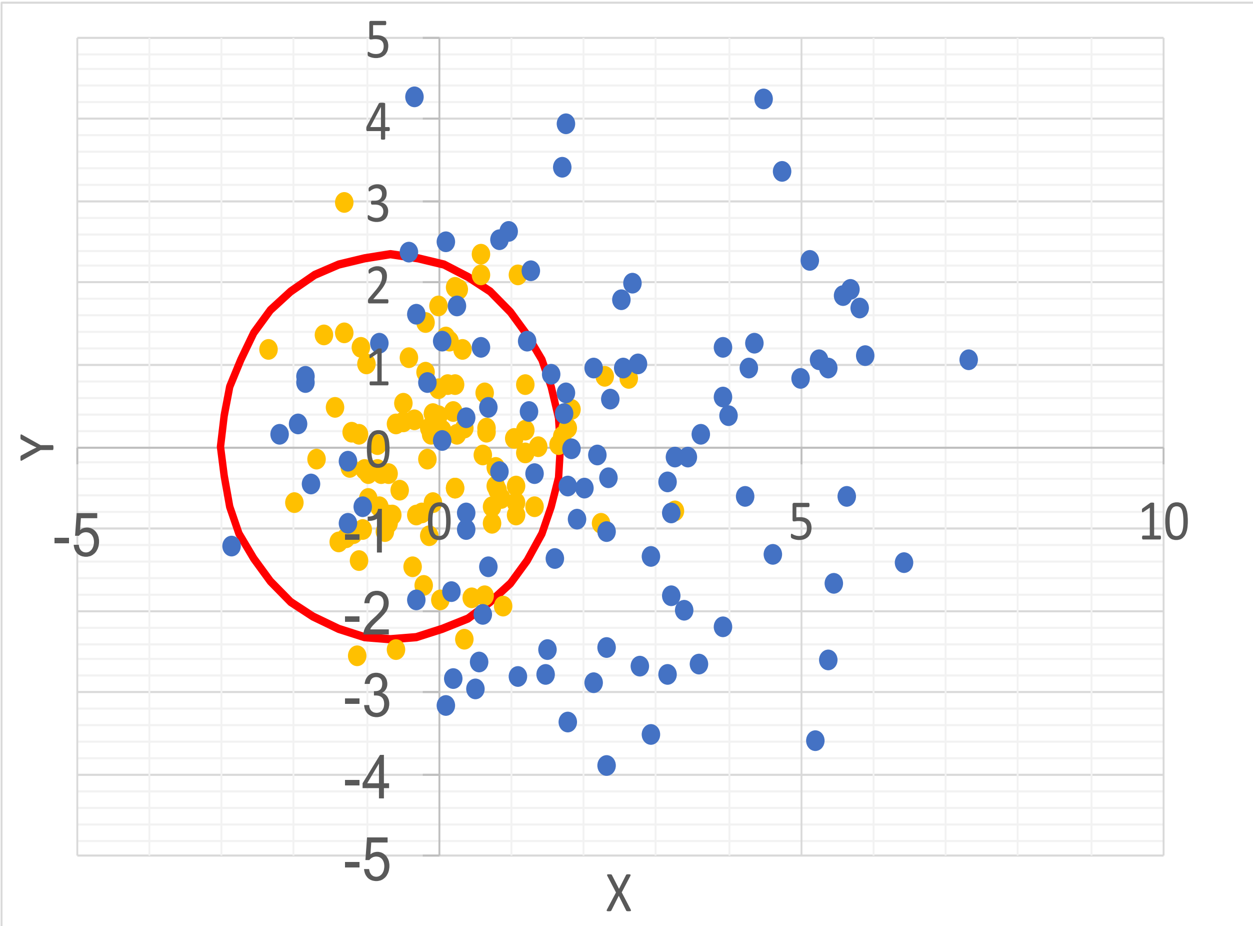
```python
import random

x1_record=[]
x2_record=[]
x1=0.0
x2=0.0
y1=0.0
y2=0.0
count1_error=0
count2_error=0
#decision boundary

for i in range(100):
    x1=random.gauss(0,1) #create random number gauss (mean, sigma)
    x2=random.gauss(0,1)
    x1_record.append(float(x1))
    x2_record.append(float(x2))

    if (x1+0.667)**2+x2**2> 2.34**2 :
        count1_error += 1
    else: pass
print('error rate 1 in %', count1_error)
for i in range(100):
    y1=2.0+random.gauss(0,2)
    y2=0.0+random.gauss(0,2)

    if (y1+0.667)**2+y2**2< 2.34**2:
        count2_error += 1
    else: pass
print('error rate 2 in %', count2_error)
```



5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
0
5
10
Y
X

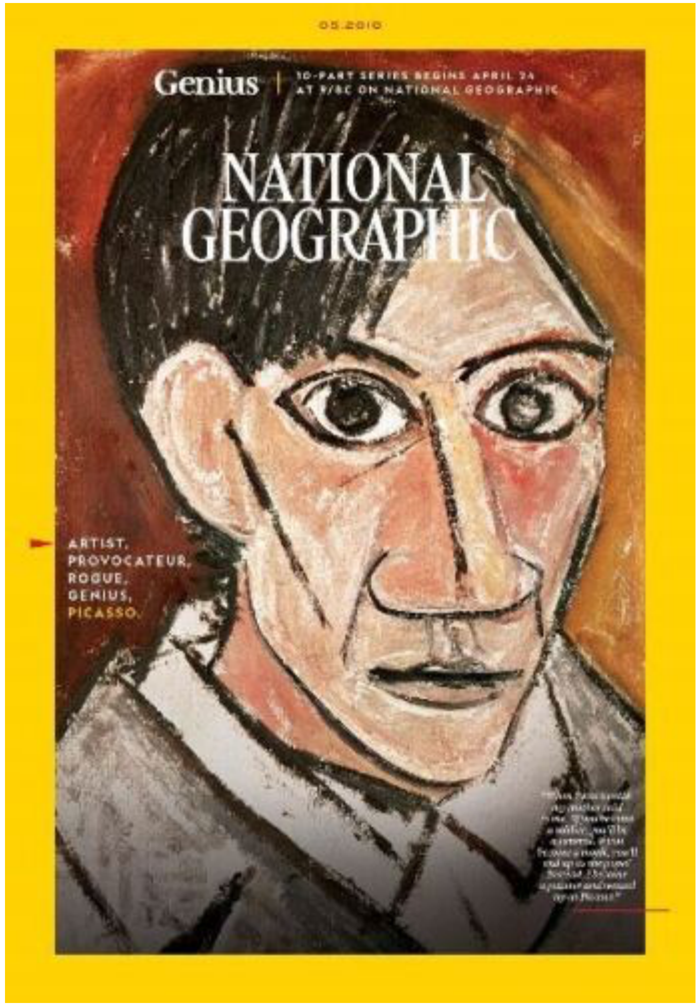
Key math notion

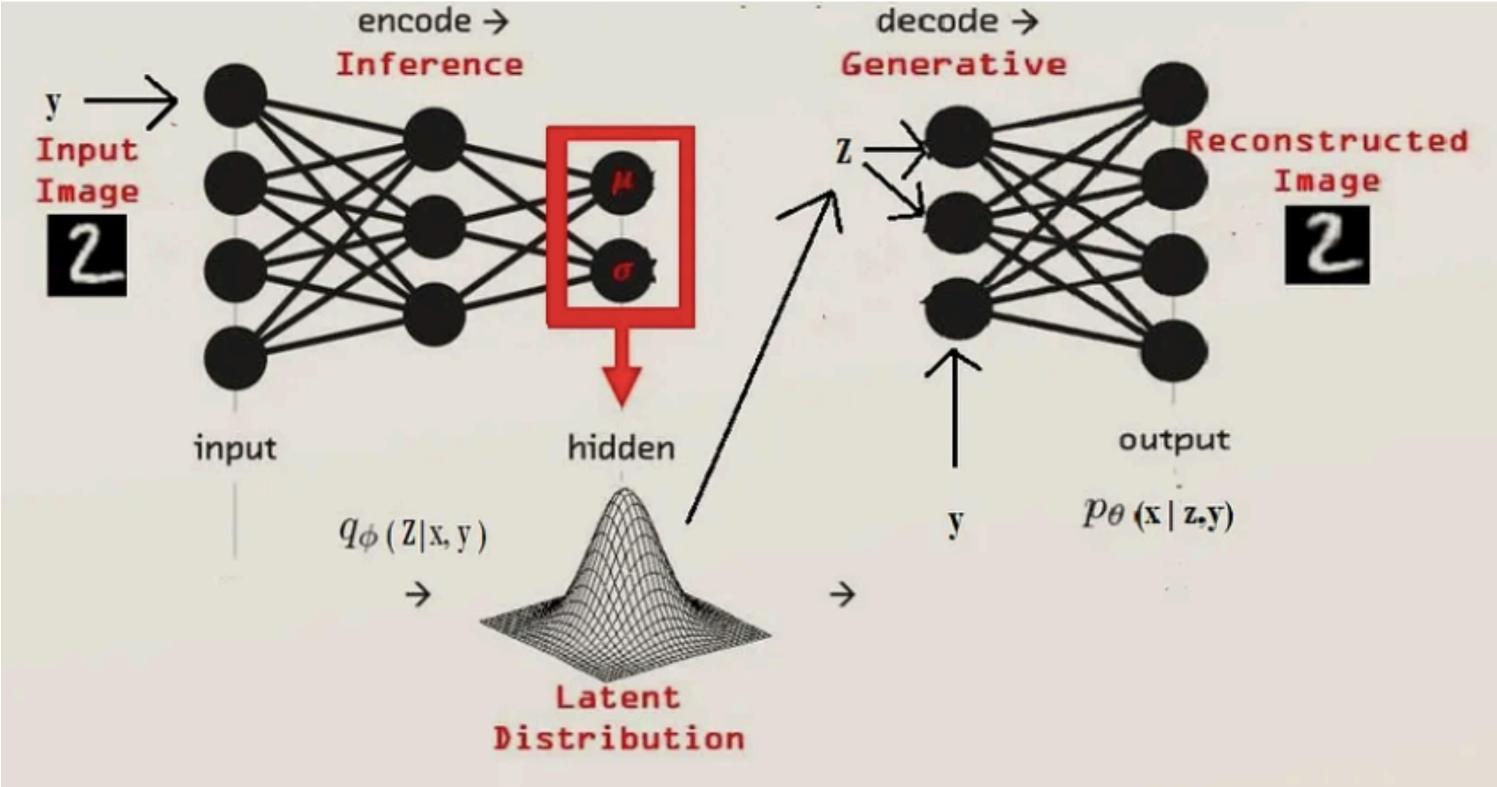
The noise can play an active role to generate new instance !!

Caves Painting
Lascaux, France

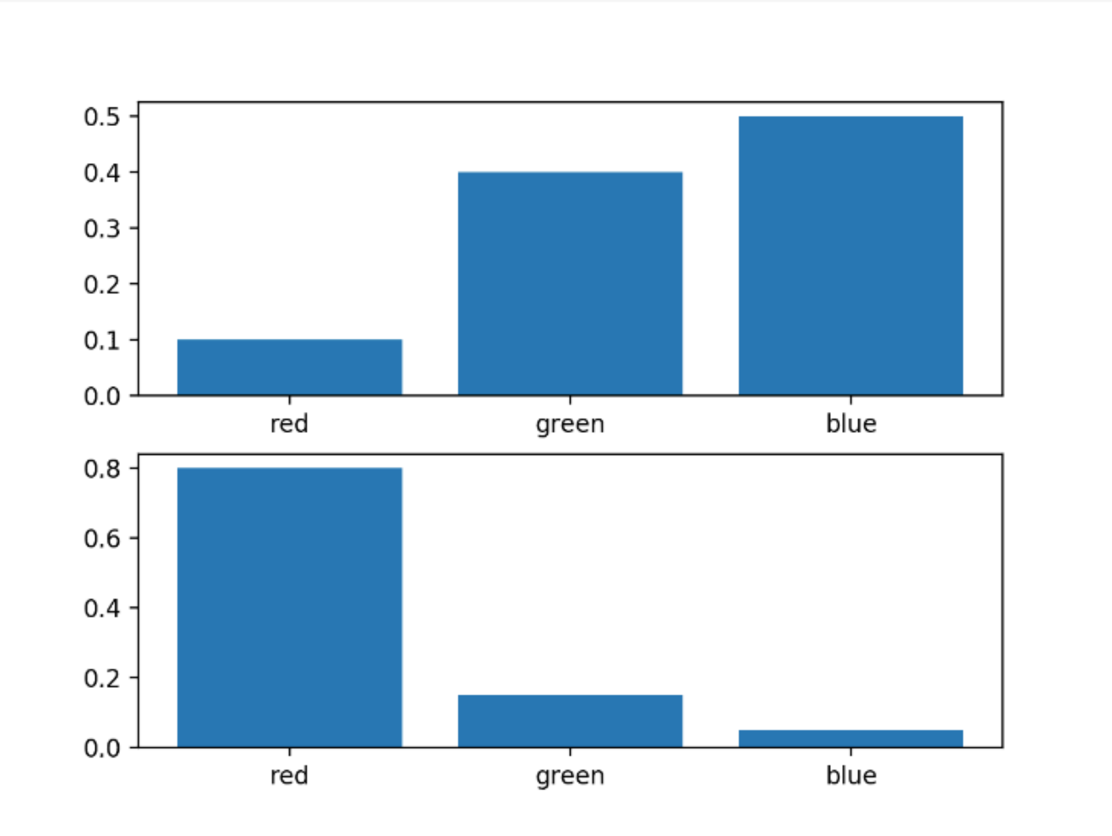

The Guardian


Genius
10-PART SERIES BEGINS APRIL 24
AT 9/8C ON NATIONAL GEOGRAPHIC
NATIONAL GEOGRAPHIC
ARTIST,
PROVOCATEUR,
ROGUE,
GENIUS,
PICASSO.



$$-\underbrace{\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right]}_{\text{reconstruction error}} + \underbrace{\mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right)}_{\text{regularization}}$$

the **Kullback–Leibler divergence** is a type of statistical distance: a measure of how one probability distribution *P* is different from a second, reference probability distribution *Q*



Histogram of Two Different Probability Distributions for the Same Random Variable

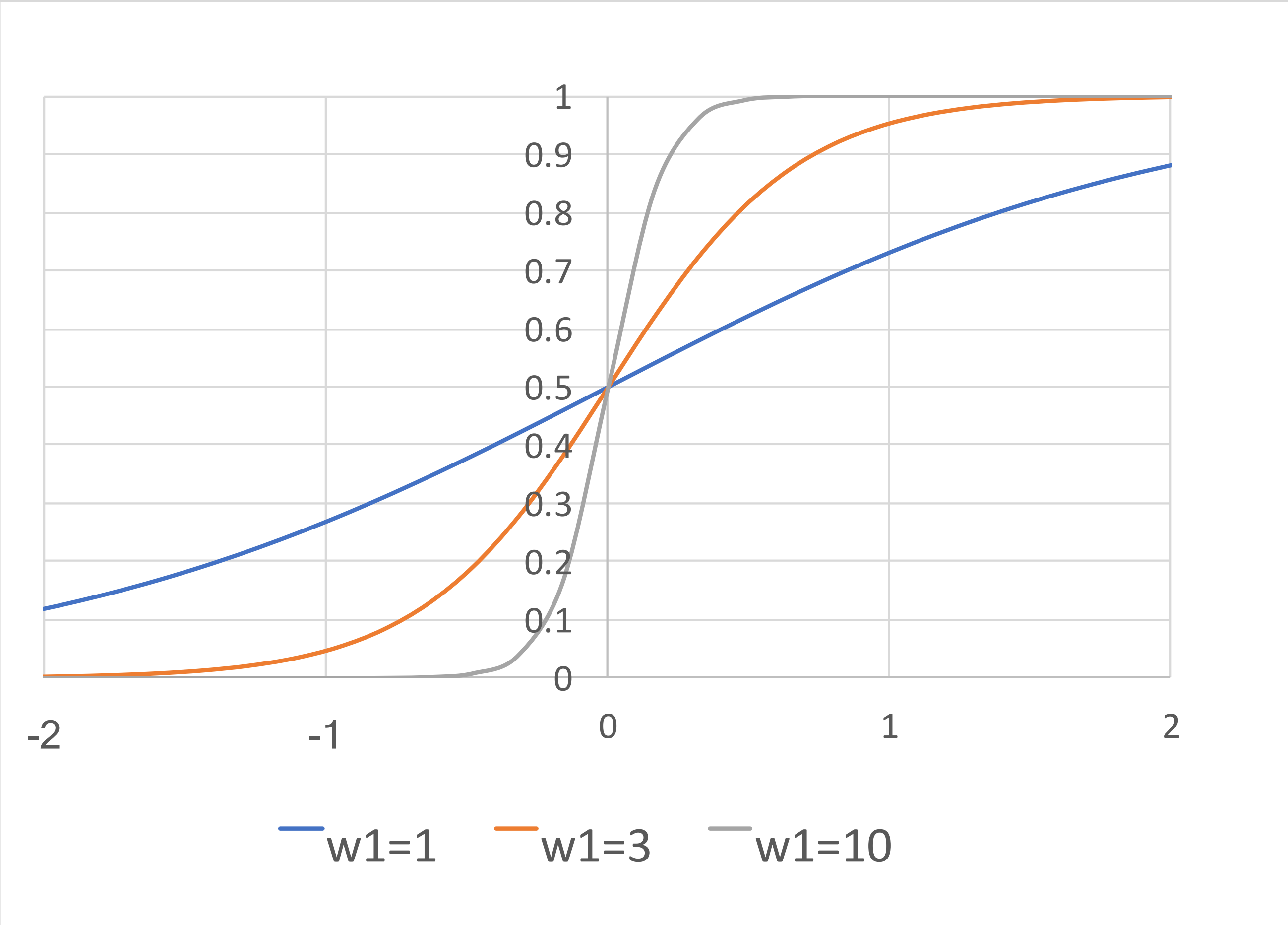
$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right).$$

Cross entropy

Intuitively Understanding the Cross Entropy

$$H(P^*|P) = -\sum_i \underbrace{P^*(i)}_{\text{TRUE CLASS DISTIRBUTION}} \log \underbrace{P(i)}_{\text{PREDICTED CLASS DISTIRBUTION}}$$

$$E\left(\mathbf{w}, w_0 \mid \mathcal{X}\right) = -\sum_t r^t \log y^t + \left(1 - r^t\right) \log\left(1 - y^t\right)$$



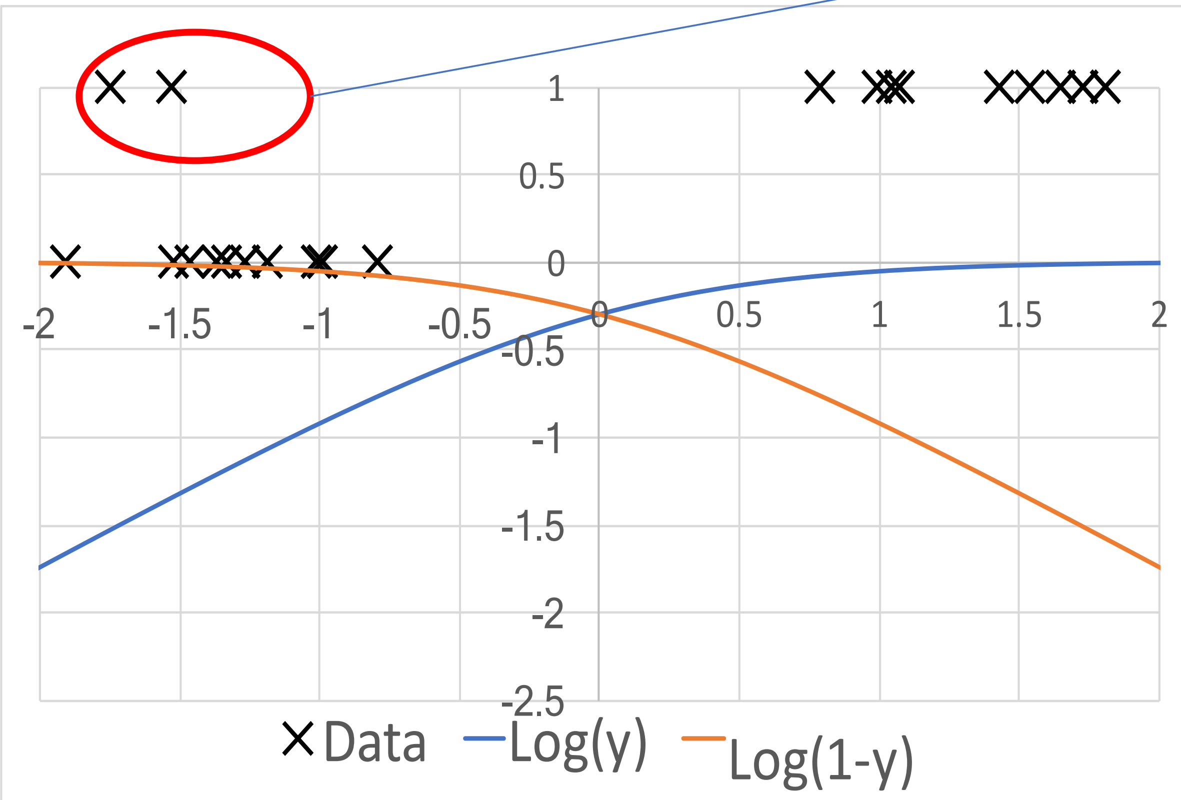
Sigmodal function

Y= 1/(1+exp(-w1*x+w0)

For simplicity
w0= 0

w1 control hardness of the threshold

Cross entropy penalty



Misclassifier contribution
From blue curve

$$E(\mathbf{w}, w_0 | \mathcal{X}) = -\sum_t r^t \log y^t + (1 - r^t) \log (1 - y^t)$$

r^t =1

Log y^t (blue curve)

w1=2 and w0=0

$$\left.\begin{aligned}
&\mathcal{X} = \{\mathbf{x}^t, r^t\}_t \quad r^t \mid \mathbf{x}^t \sim \text{Bernoulli}(y^t) \\
&y = P(C_1 \mid \mathbf{x}) = \frac{1}{1 + \exp\left[-\left(\mathbf{w}^T\mathbf{x} + w_0\right)\right]} \\
&l(\mathbf{w}, w_0 \mid \mathcal{X}) = \prod_t \left(y^t\right)^{\left(r^t\right)} \left(1 - y^t\right)^{\left(1 - r^t\right)} \\
&E = -\log l
\end{aligned}\right\} \text{Safe skip this part}$$

$$\boxed{E(\mathbf{w}, w_0 \mid \mathcal{X}) = -\sum_t r^t \log y^t + \left(1 - r^t\right) \log\left(1 - y^t\right)}$$

Training: Gradient-Descent

$$E(\mathbf{w}, w_0 \mid \mathcal{X}) = -\sum_t r^t \log y^t + (1 - r^t) \log (1 - y^t)$$

$$\text{If } y = \text{sigmoid(a)} \quad \frac{dy}{da} = y(1 - y)$$

$$\Delta w_j = -\eta \frac{\partial E}{\partial w_j} = \eta \sum_t \left(\frac{r^t}{y^t} - \frac{1 - r^t}{1 - y^t} \right) y^t (1 - y^t) x_j^t$$

$$= \eta \sum_t (r^t - y^t) x_j^t, \ j = 1, \ldots, d$$

$$\Delta w_0 = -\eta \frac{\partial E}{\partial w_0} = \eta \sum_t (r^t - y^t)$$

For $j = 0, \ldots, d$

  $w_j \leftarrow$rand(-0.01,0.01)

Repeat

  For $j = 0, \ldots, d$

    $\Delta w_j \leftarrow 0$

  For $t = 1, \ldots, N$

    $o \leftarrow 0$

    For $j = 0, \ldots, d$

      $o \leftarrow o + w_j x_j^t$

    $y \leftarrow \text{sigmoid}(o)$

    $\Delta w_j \leftarrow \Delta w_j + (r^t - y) x_j^t$

  For $j = 0, \ldots, d$

    $w_j \leftarrow w_j + \eta \Delta w_j$

Until convergence

Generating testing data

```
def sigmod(x,w1,w0):
    return 1/(1+ math.exp(-w1*x-w0))
for iteration in range(30):
    x=random.uniform(-2,2)
    zeta=random.uniform(0,1)
    y= sigmod(x,w1_set, w0_set)
    if zeta <= y:
        outcome=1
        x_record.append(float(x))
        r_record.append(int(outcome))
        print( 'iteration', iteration, 'x', round(x,2),' ', outcome)

    else:
        outcome=0
        x_record.append(float(x))
        r_record.append(int(outcome))
        print( 'iteration', iteration,'x', round(x,2),' ', outcome)
```

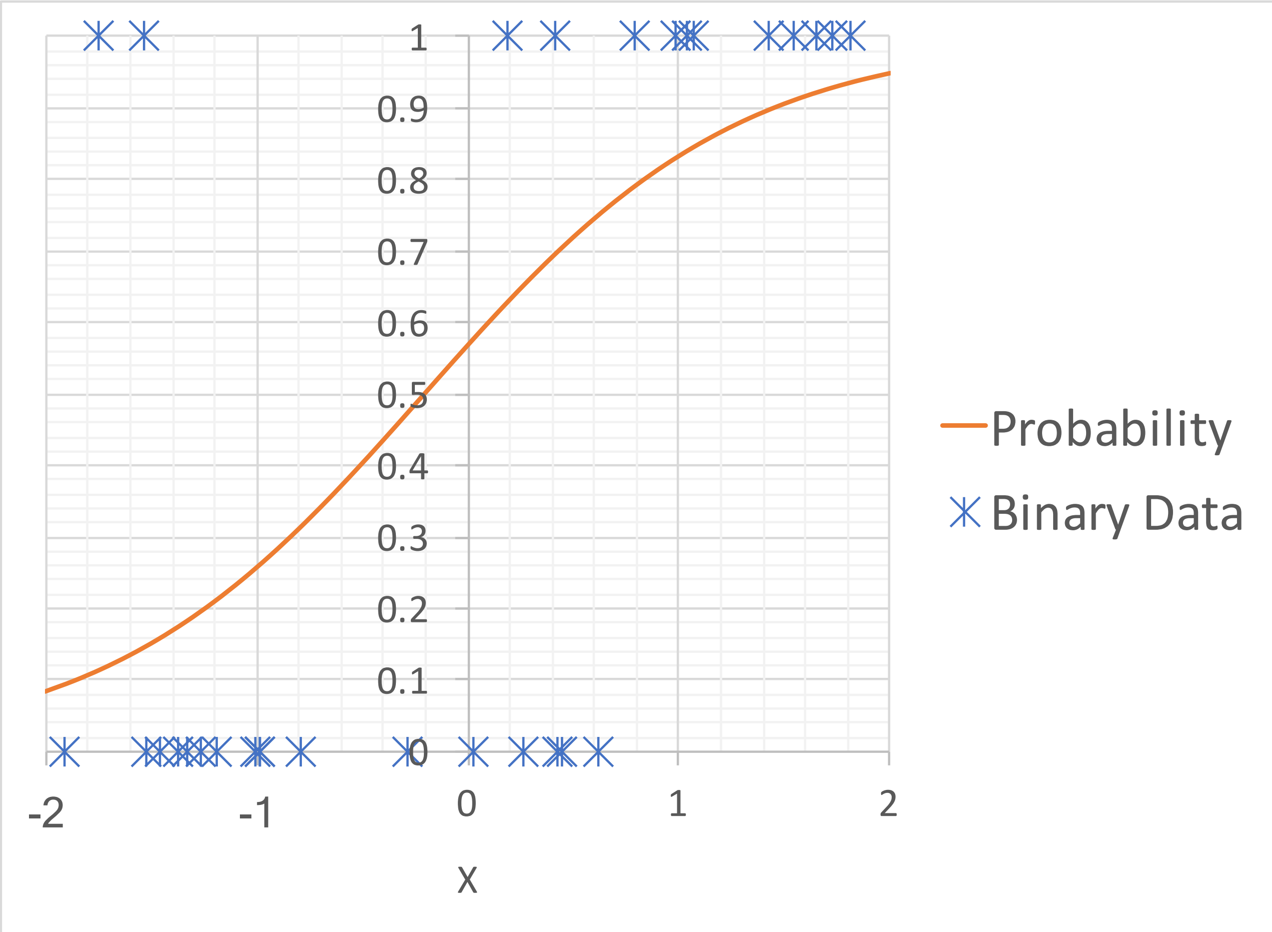
Parameter used:

```
import random
import math
w1_set=1.2
w0_set=0
x=0.0
outcome=0
w1=0.01
w0=0.01
eta=0.01
x_record=[]
r_record=[]
```

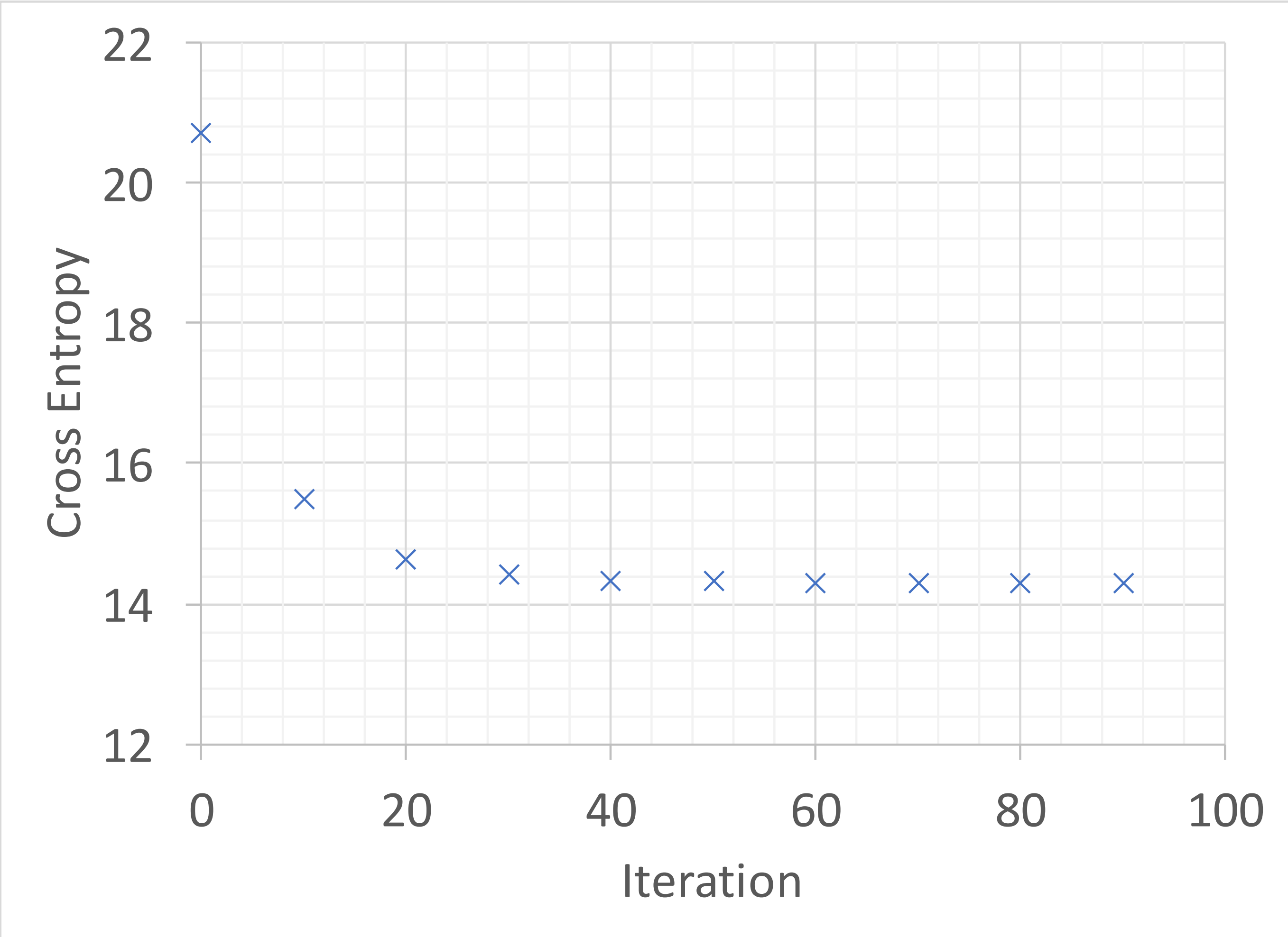
Weight update

```
for iteration2 in range(epoch):
    sum2=0.0
    sum1=0.0
    sum_en=0.0
    for i in range(30):
        y= sigmod(x_record[i],w1,w0)
        sum1= sum1+(r_record[i] -y)*x_record[i]
        sum2=sum2+(r_record[i] -y)
        #computing cross entropy
        sum_en= sum_en-(r_record[i]*math.log(y)+(1-r_record[i])*math.log(1-y))

    w1=w1+eta*sum1
    w0=w0+eta*sum2
    if iteration2 %10==0:
        print( 'iteration', iteration2, 'error', sum_en)
        #print( 'iteration', iteration2, 'w1', round(w1,2), 'w0', round(w0,2))
```



In total , we use 30 sample



Cross Entropy
22
20
18
16
14
12
0
20
40
60
80
100
Iteration

For K>2 class

- Use cross entropy and weight update
- Compute a score
- Use softmax to convert score to probability

$$E(\{\mathbf{w}_i, w_{i0}\}_i \mid \mathcal{X}) = -\sum_t r_i^t \log y_i^t$$

$$\Delta \mathbf{w}_j = \eta \sum_t \left(r_j^t - y_j^t\right)\mathbf{x}^t \quad \Delta w_{j0} = \eta \sum_t \left(r_j^t - y_j^t\right)$$

For $i = 1, \ldots, K$, For $j = 0, \ldots, d$, $w_{ij} \leftarrow$ rand$(-0.01, 0.01)$

Repeat

    For $i = 1, \ldots, K$, For $j = 0, \ldots, d$, $\Delta w_{ij} \leftarrow 0$

    For $t = 1, \ldots, N$

        For $i = 1, \ldots, K$

            $o_i \leftarrow 0$

            For $j = 0, \ldots, d$

                $o_i \leftarrow o_i + w_{ij} x_j^t$

        For $i = 1, \ldots, K$

            $y_i \leftarrow \exp(o_i) / \sum_k \exp(o_k)$

        For $i = 1, \ldots, K$

            For $j = 0, \ldots, d$

                $\Delta w_{ij} \leftarrow \Delta w_{ij} + (r_i^t - y_i) x_j^t$

    For $i = 1, \ldots, K$

        For $j = 0, \ldots, d$

            $w_{ij} \leftarrow w_{ij} + \eta \Delta w_{ij}$

Until convergence

Example