Logistic Regression

One sample, $X \Rightarrow x_1, x_2$ (2 features)

Which class does these features correspond to ??

$\Rightarrow$ 0/1 ??

↳ binary classification

$(1, x_1, x_2) \cdot dot (\theta_3, \theta_1, \theta_2) \} \Rightarrow (X\theta)$

$= \theta_1 x_1 + \theta_2 x_2 + \theta_3$

↳ we update and tune θ_1, θ_2 and θ_3 such that $X\theta$ becomes close to 0 for x_1, x_2 coming from class 0 samples. close to 1 for x_1, x_2 coming from class 1 samples.

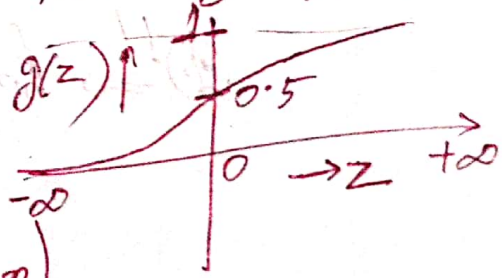
$\theta \Rightarrow (\theta_3, \theta_1, \theta_2) \Rightarrow$ parameters

θ_3 ⇓ bias; θ_1, θ_2 ↘ weights

Its difficult to get $X\theta$ in the range of [0, 1].

Activation function → ? (1) ?

$Z = X\theta = \theta_1 x_1 + \theta_2 x_2 + \theta_3$



$h = g(z) = \frac{1}{1+e^{-z}}$ (sigmoid function)

$h \Rightarrow$ predicted value $\Rightarrow [0, 1]$

$y \Rightarrow$ true label

Now how do we know what values to use for θ??

Ans: Gradient descent optimization

$J(\theta) = -y \log(h) - (1-y) \log(1-h)$

↳ log loss function

$y=0,\ h=0$ (correct) $\Rightarrow -0 \cdot \log(0) - 1 \log 1 \to 0$

$\Rightarrow 0$

$y=1,\ h=1$ (correct) $\Rightarrow -1 \log(1) - 0 \log 0 \to 0$

$\Rightarrow 0$

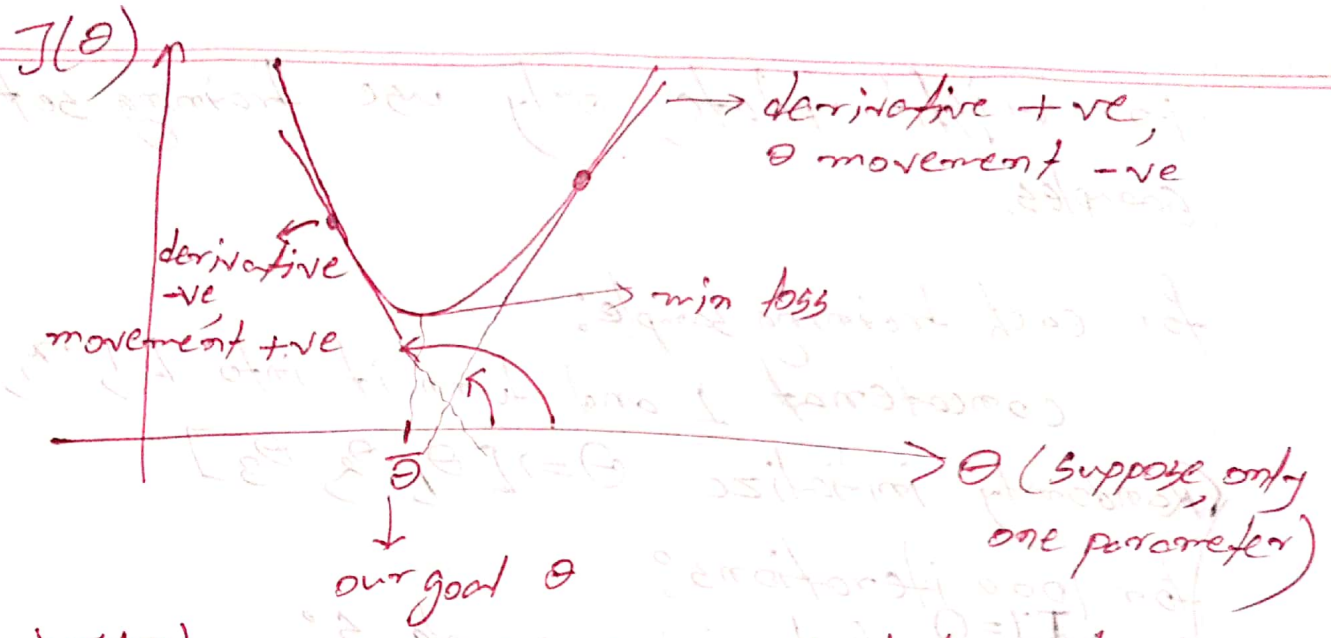
$y=1,\ h=0$ (wrong) $\Rightarrow -1 \log(0) - 0 \cdot \log 1$

$\Rightarrow -(-\infty) - 0$

$\Rightarrow +\infty$

The closer the value of h to y, less will be the loss.

☹!! ⇒ how do we update θ??



$$dv = \frac{\partial J(\theta)}{\partial(\theta)} = x(h - y)$$ ⟹ calculate and see for yourself

$\theta = \theta - dv$

↳ a slight problem.

What if dv is too BIG??

will never reach $\bar{\theta}$.

learning rate (l.r.) is the answer

$\theta = \theta - dv * l.r.$ ⟶ l.r. < 1 and l.r. > 0

typically ⇒ 0.01, 0.001, 0.0001

we set this manually

by looking at validation results for different l.r.

↳ hyperparameter

How many times to update θ??

↳ until loss function value does not decrease any longer.

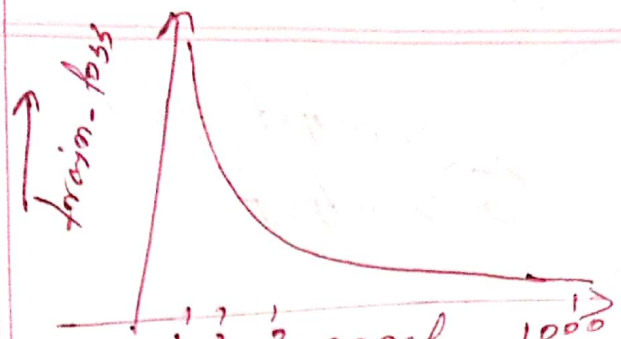
for weight update, only use training set samples.

for each training sample:
    concatenat 1 and turn it into $[1, x_1, x_2]$

Randomly initialize $\Theta \Rightarrow [\theta_1, \theta_2, \theta_3]$

for 1000 iterations:
    $TJ = 0$
    for each training sample s:
        $z = x\theta$ [$x \Rightarrow s$ features, $\theta \Rightarrow$ parameters]
            use np.dot]
        $h = \text{sigmoid}(z)$ → available in python library
        $J = -y \log(h) - (1-y)\log(1-h)$
            → [true label of s]
        $TJ = TJ + J$ → total loss for this iteration
        $dv = x(h-y)$ → it has 3 dimensions
        $\theta = \theta - dv * l.r.$ → also has 3 dimensions
    $TJ = TJ / (\text{training sample no.})$
    Append TJ in a list named "train_loss"

training => updating θ



→ iteration

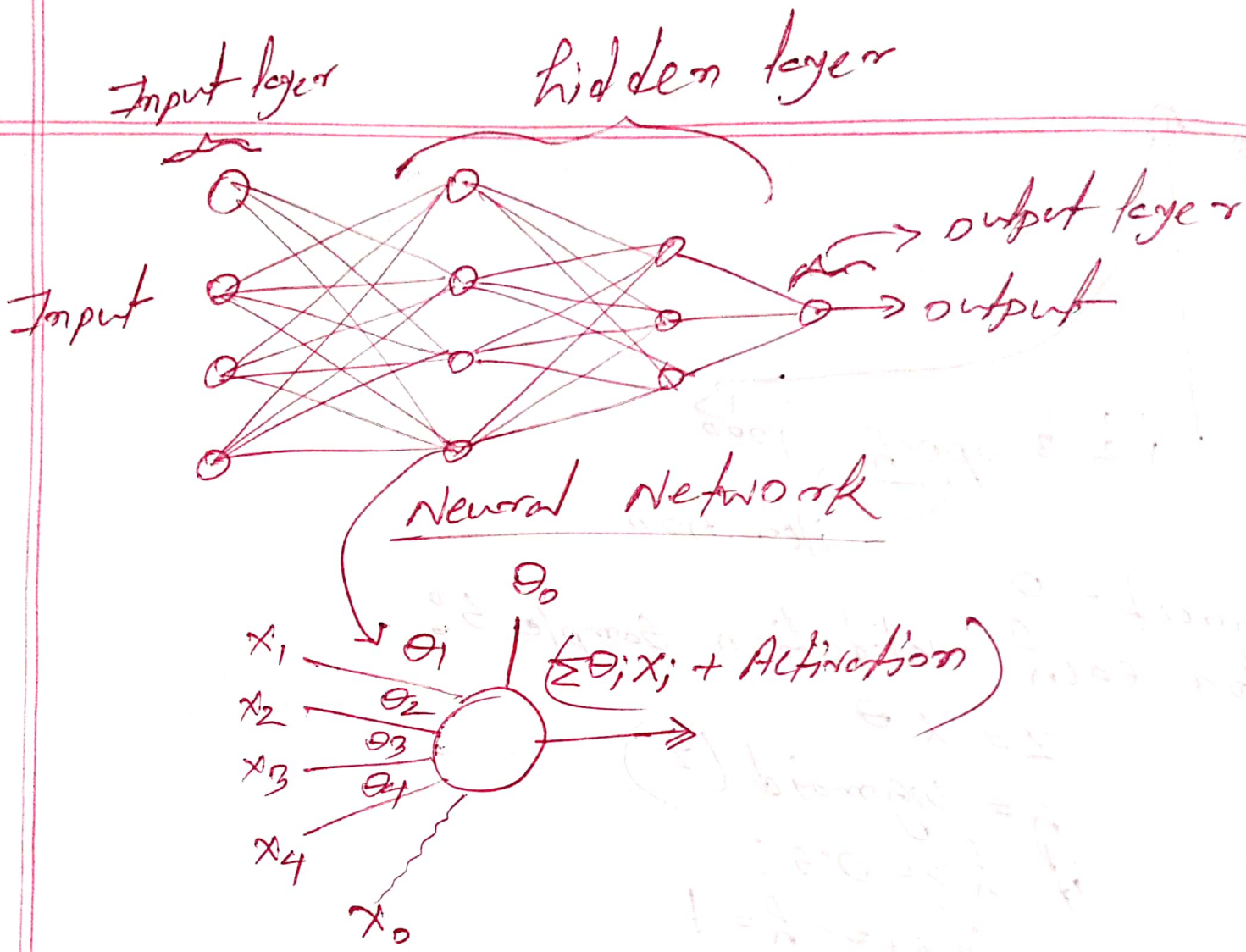
validation

```
correct = 0
for each validation sample s:
        z = X θ
        h = sigmoid(z)
        if h >= 0.5:
            ~~s_total~~ h = 1
        else: h = 0
        if h == y:
            correct += 1
val_acc = correct/(total val sample no.) * 100
```

perform this training-validation for l.r. = 0.1, 0.01, 0.001, 0.0001, 0.00001 and report val_acc in a table. → hyperparameter tuning

Take the l.r. with best val_acc

Use this l.r. for training and determine test_acc with the trained model



Neural Network