Optimization

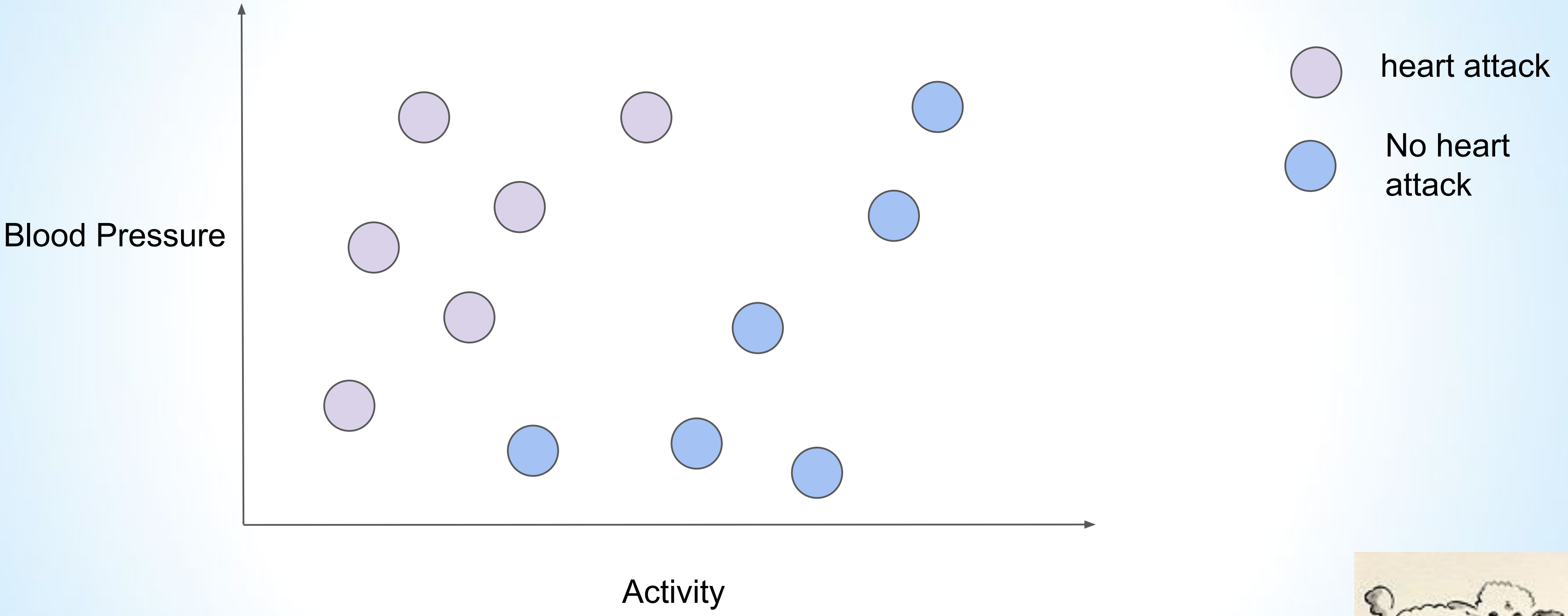
Noa Lubin & Lior Sidi

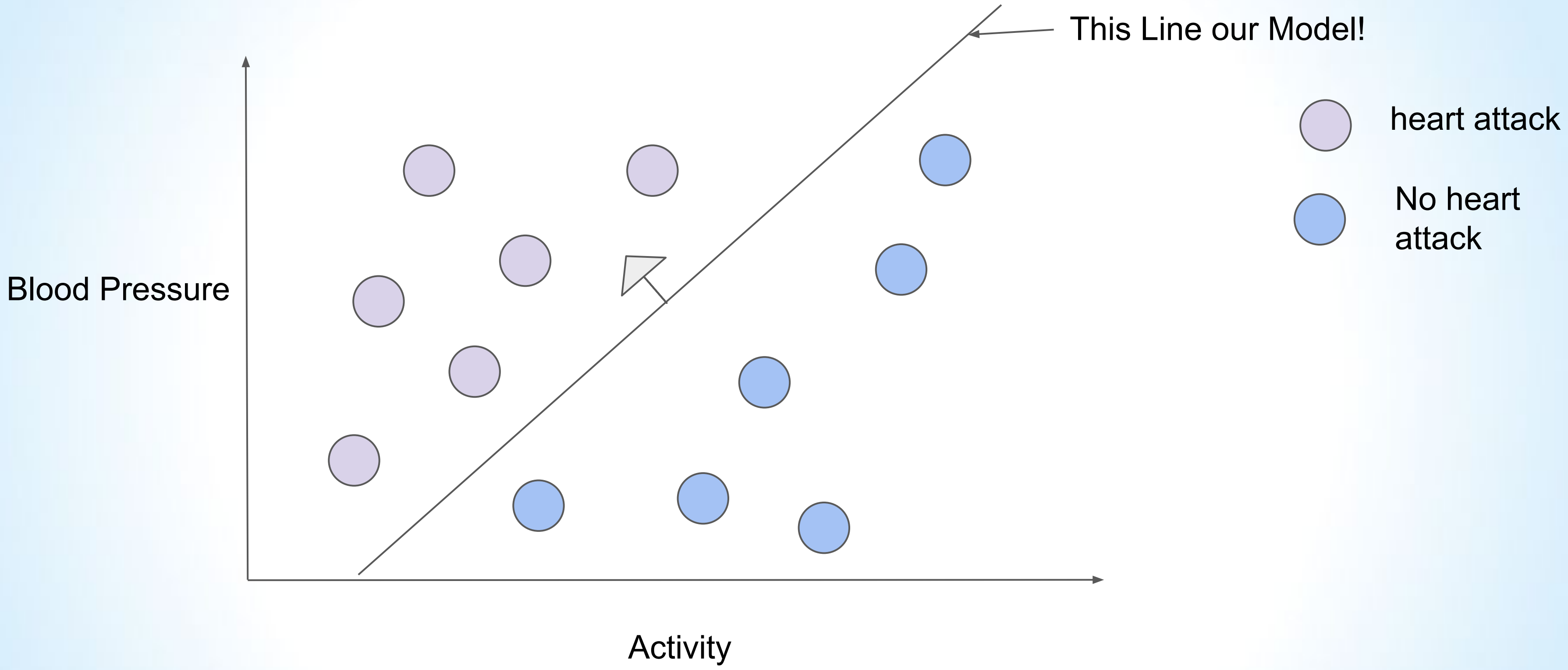
Agenda

- Motivation
- Optimization Theory
- Gradient Descent Toy Example

Motivation

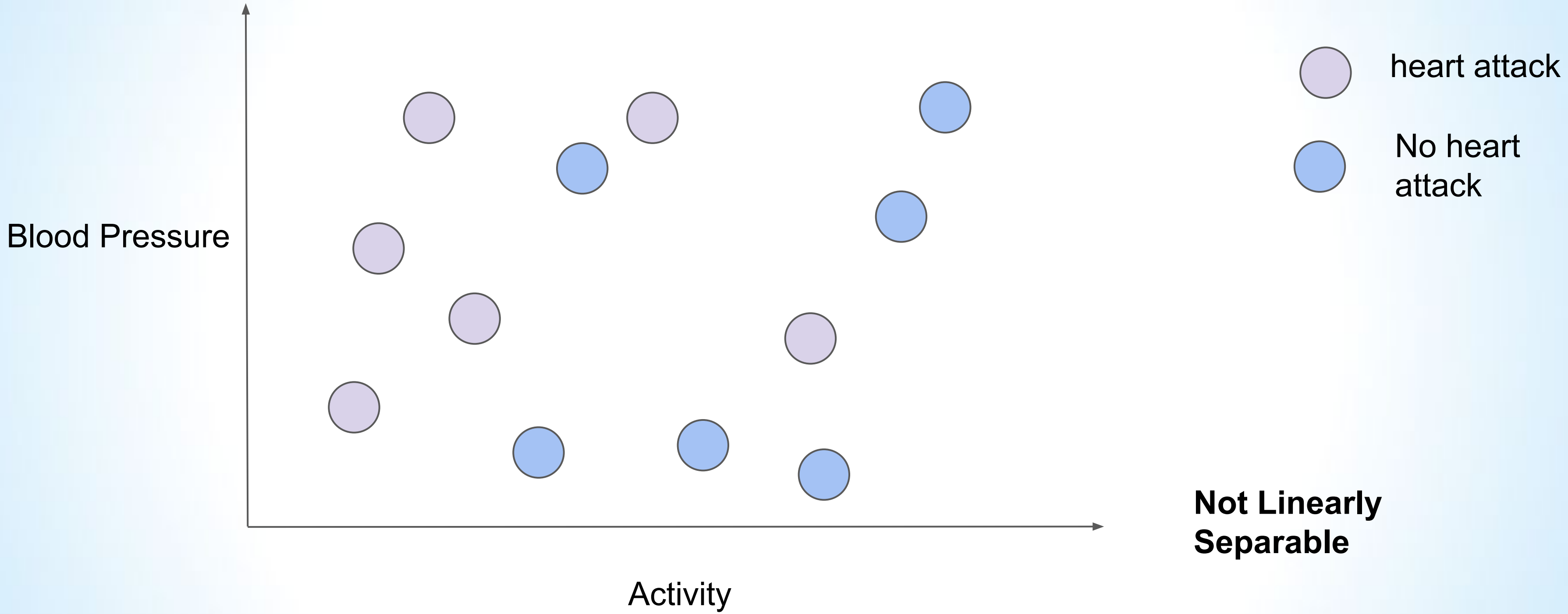
Draw Me a Model





This Line our Model!

In Real Life



Not Linearly Separable

Mathematical Formulation

Features:

$x1$ - blood pressure

$x2$ - activity

$f(x1,x2) =$ Heart Attack if $w0 + w1*x1 + w2*x2 >= 0$

No Heart Attack otherwise

Mathematical Formulation - Matrix Form

Features:

x1 - blood pressure → x = <1, x1,x2>

x2 - activity

w = <w0, w1,w2>

wx = 1*w0+x1*w1+x2*w2

f(x1,x2) = Heart Attack if wx >=0
No Heart Attack otherwise

This is a linear classifier!

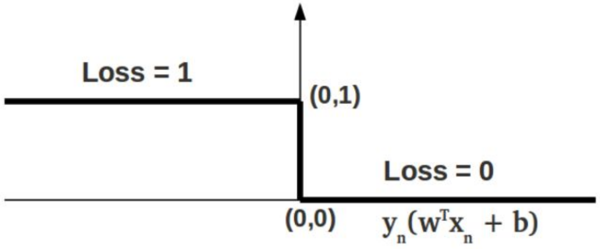
How can we find w?

Loss Function

A loss function or cost function indicates “how wrong” we are with our current hypothesis

For example:

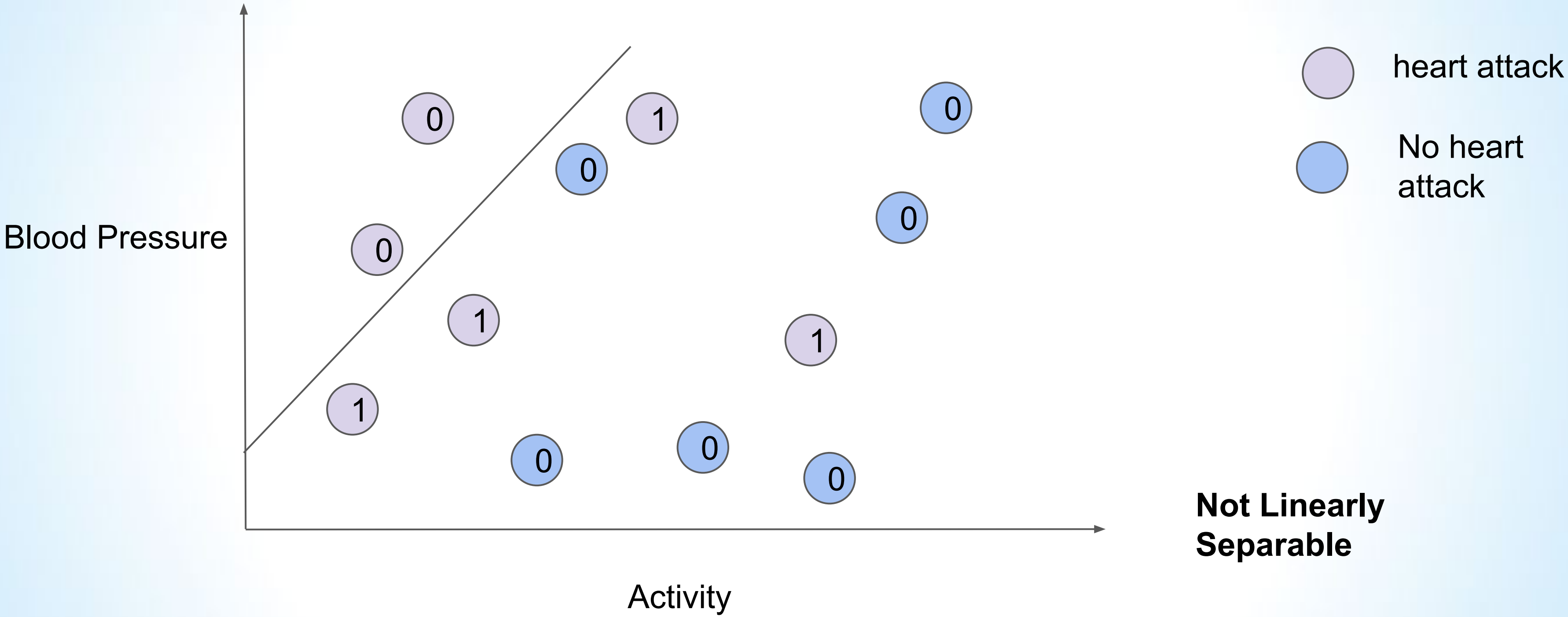
0 - 1 loss: 1 If the classification is incorrect
0 If the classification is correct



$$l(\acute{y}, y) = \begin{cases} 0 & y = \acute{y} \\ 1 & y \neq \acute{y} \end{cases} = II\{y \neq \acute{y}\}$$

1-0 loss

Let's guess a model

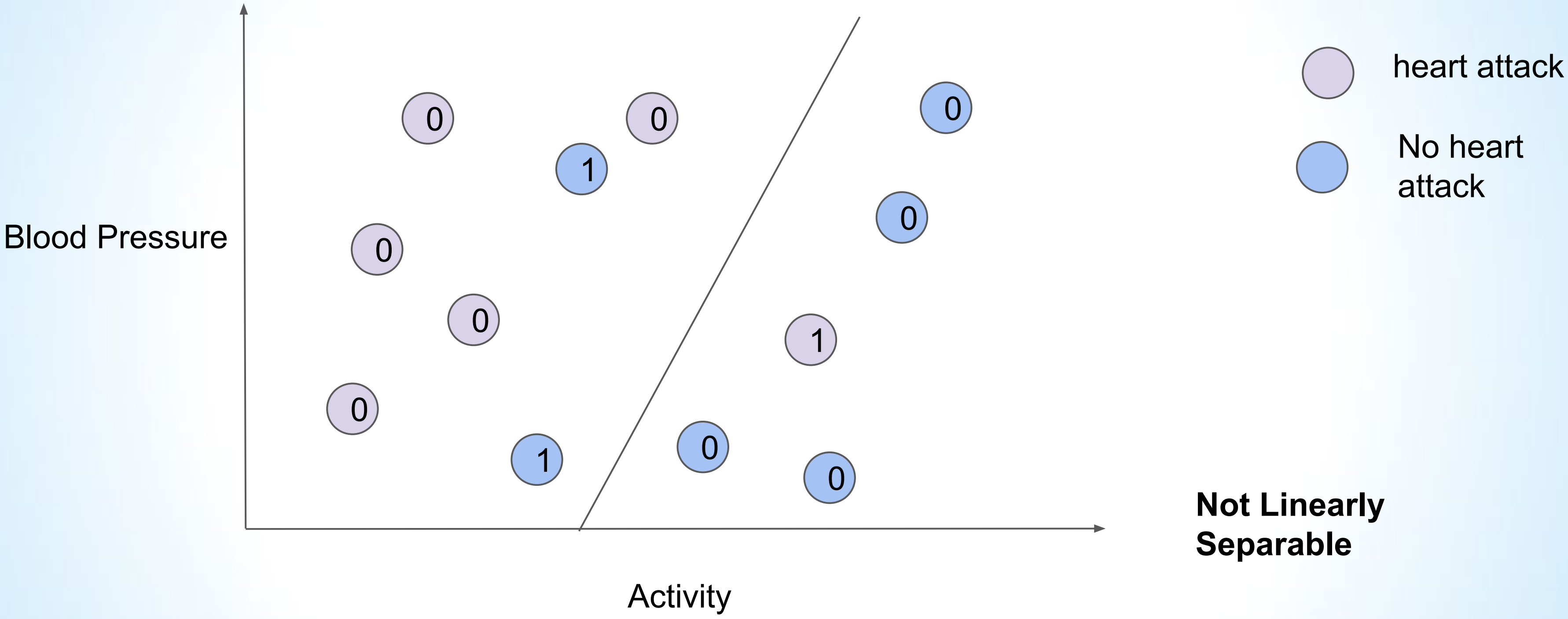


Not Linearly Separable

Loss = 4

1-0 loss

Let’s guess a model

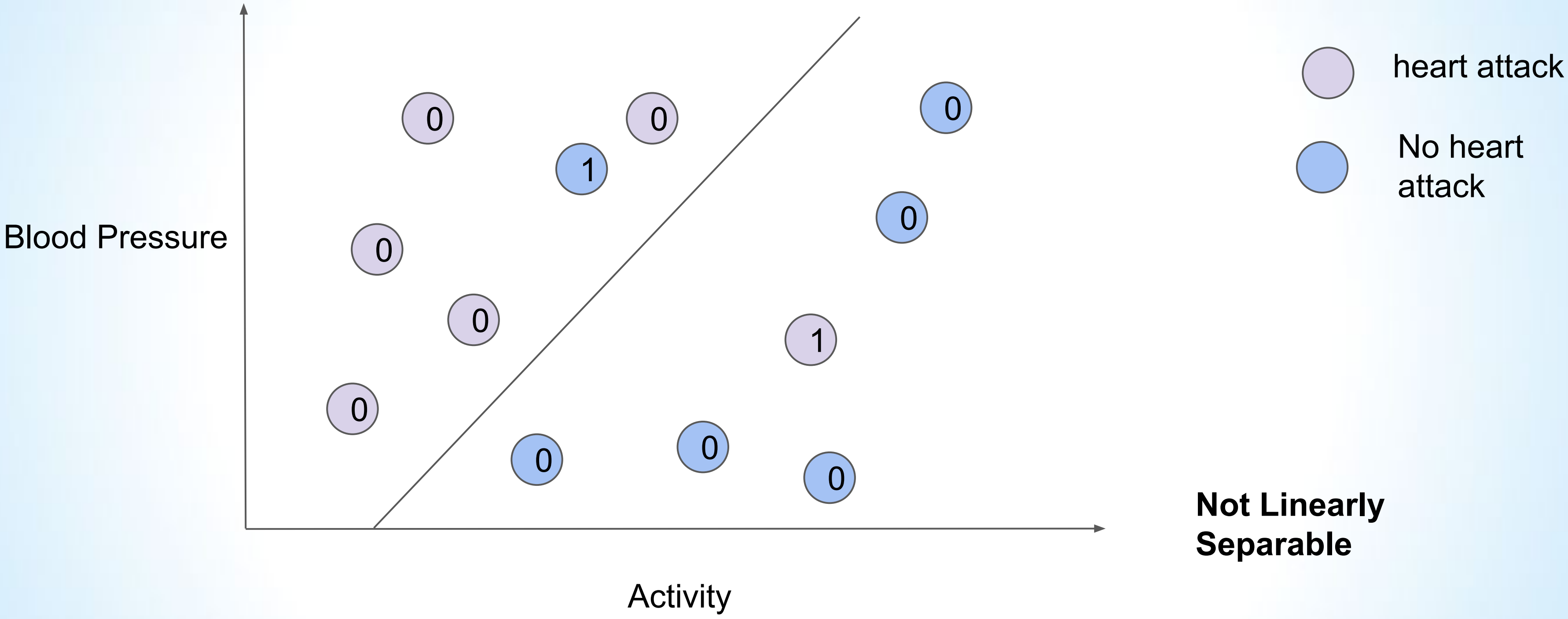


Not Linearly Separable

Loss = 3

1-0 loss

Let’s guess a model



Not Linearly Separable

Loss = 2

1-0 loss

Can you guess a non - linear model with 1-0 loss = 0?

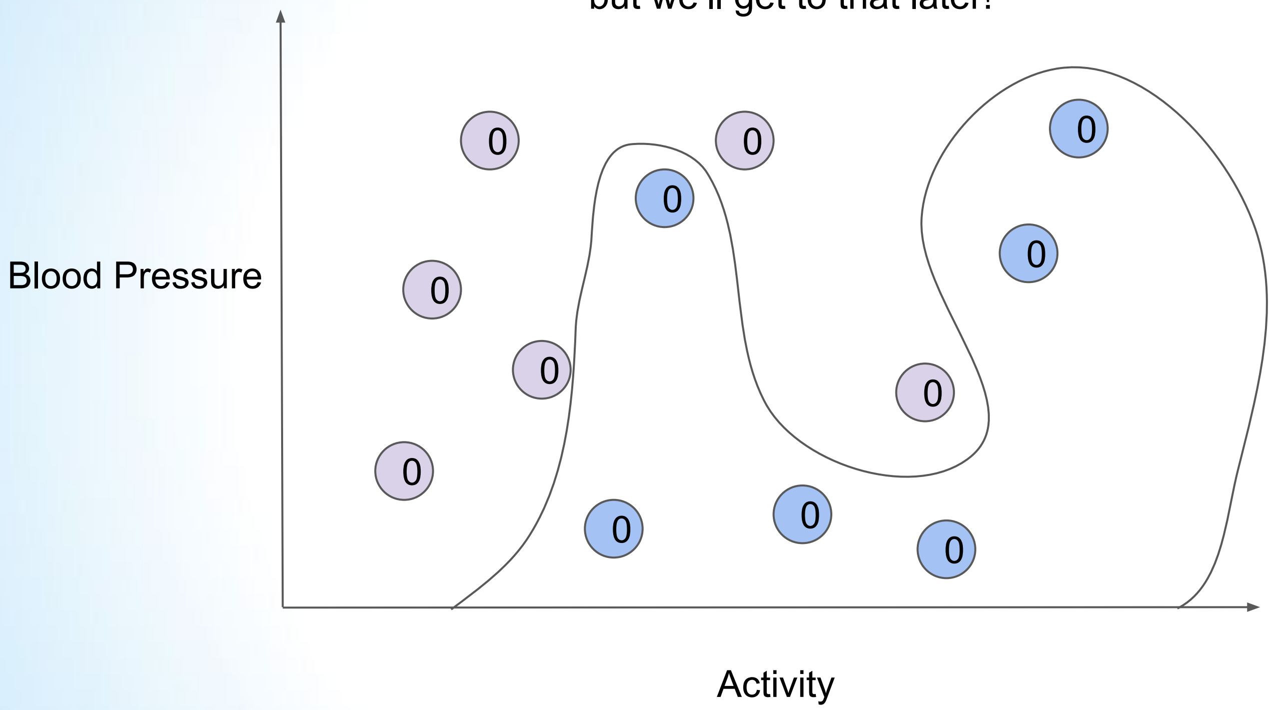
● heart attack

● No heart attack

Not Linearly Separable

1-0 loss

This what we call overfitting
but we'll get to that later!



● heart attack

● No heart attack

Not Linearly Separable

Loss = 0

Optimization Theory

Reminder:

Used to learn model params
Epochs - the number of times our model sees the train set

Used to choose model hyper-params

Used to evaluate model

ERM principal

- Minimize your loss on the observed data (training set)

$$\theta^* = argmin_\theta \frac{1}{m}\sum_{j=1}^{m} l(y_j, f_\theta(x_j))$$

Optimization

The optimization method is independent of the loss function.

Loss - Where do we want to get?
Optimization - How do we want to get there?

We’ll now discuss optimization methods.

Types Of Optimization

- **Gradient Descent**
- **Stochastic Gradient Descent**
- **Mini-batch Gradient Descent**
- Momentum
- AdaGrad
- RMSprop
- Adam
- …

Gradient Descent

The idea of gradient descent is:

Take iterative steps to update parameters in the direction of the gradient

$$\underset{\text{New weights}}{\underset{\uparrow}{w^t}} \leftarrow \overset{\text{Previous weights}}{\overset{\downarrow}{w^{t-1}}} - \underset{\text{Step size / learning rate}}{\underset{\uparrow}{\eta}} \cdot \overset{\text{gradient}}{\overset{\downarrow}{\frac{\partial \ell}{\partial w^{t-1}}}}$$

Gradient

A collection of partial derivatives

$$\nabla f(x_1, x_2,x_3) = \left(\frac{\partial f(x_1, x_2, x_3)}{\partial x_1} \frac{\partial f(x_1, x_2, x_3)}{\partial x_2} \frac{\partial f(x_1, x_2, x_3)}{\partial x_3}\right)$$

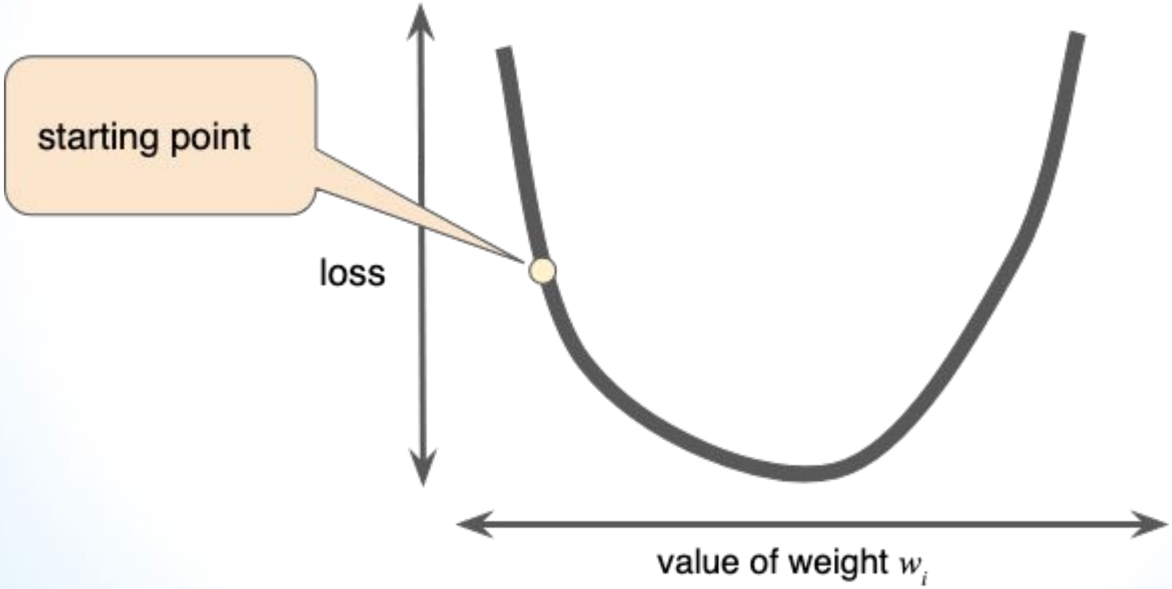
Gradient for multiple inputs

$$\frac{\partial L(\hat{\mathbf{y}}, \mathbf{y})}{\partial \mathbf{W}} = \begin{pmatrix} \frac{\partial L(\hat{\mathbf{y}},\mathbf{y})}{\partial \mathbf{W}_{[1,1]}} & \frac{\partial L(\hat{\mathbf{y}},\mathbf{y})}{\partial \mathbf{W}_{[1,2]}} & \cdots & \frac{\partial L(\hat{\mathbf{y}},\mathbf{y})}{\partial \mathbf{W}_{[1,n]}} \\ \frac{\partial L(\hat{\mathbf{y}},\mathbf{y})}{\partial \mathbf{W}_{[2,1]}} & \frac{\partial L(\hat{\mathbf{y}},\mathbf{y})}{\partial \mathbf{W}_{[2,2]}} & \cdots & \frac{\partial L(\hat{\mathbf{y}},\mathbf{y})}{\partial \mathbf{W}_{[2,n]}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial L(\hat{\mathbf{y}},\mathbf{y})}{\partial \mathbf{W}_{[m,1]}} & \frac{\partial L(\hat{\mathbf{y}},\mathbf{y})}{\partial \mathbf{W}_{[m,2]}} & \cdots & \frac{\partial L(\hat{\mathbf{y}},\mathbf{y})}{\partial \mathbf{W}_{[m,n]}} \end{pmatrix}$$

$$\frac{\partial L(\hat{\mathbf{y}}, \mathbf{y})}{\partial \mathbf{b}} = \begin{pmatrix} \frac{\partial L(\hat{\mathbf{y}},\mathbf{y})}{\partial \mathbf{b}_{[1]}} & \frac{\partial L(\hat{\mathbf{y}},\mathbf{y})}{\partial \mathbf{b}_{[2]}} & \cdots & \frac{\partial L(\hat{\mathbf{y}},\mathbf{y})}{\partial \mathbf{b}_{[n]}} \end{pmatrix}$$

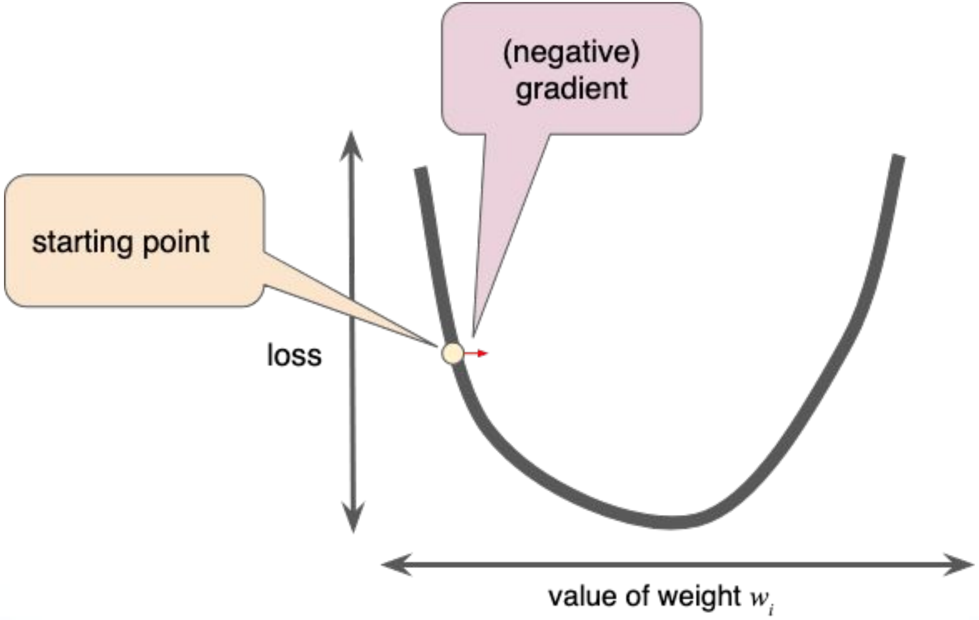
Gradient Descent

- Random starting point



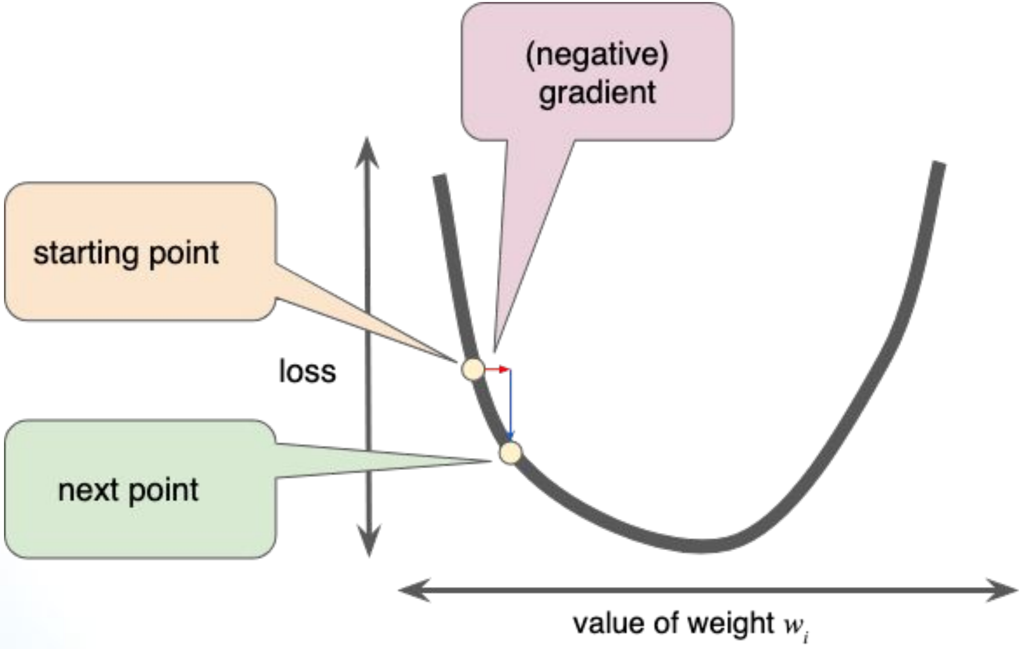
Gradient Descent

- A gradient is a vector so it has both magnitude and direction

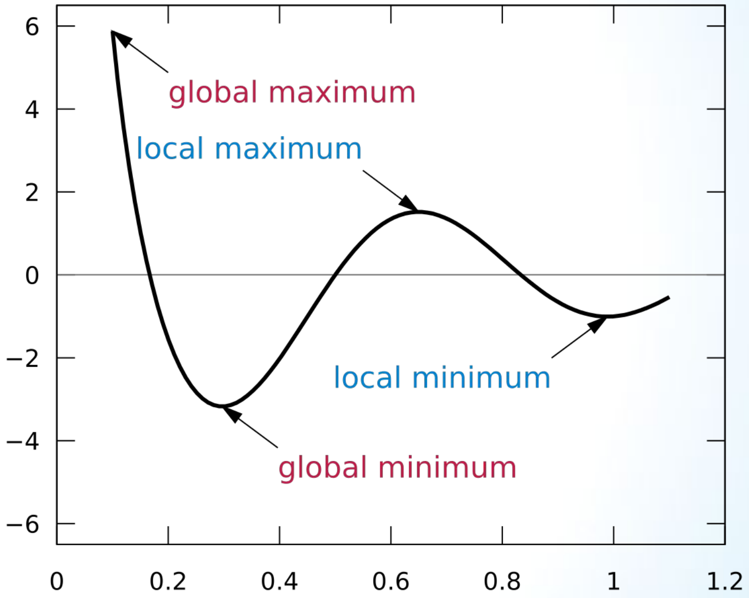


Gradient Descent

Until we meet stop criteria



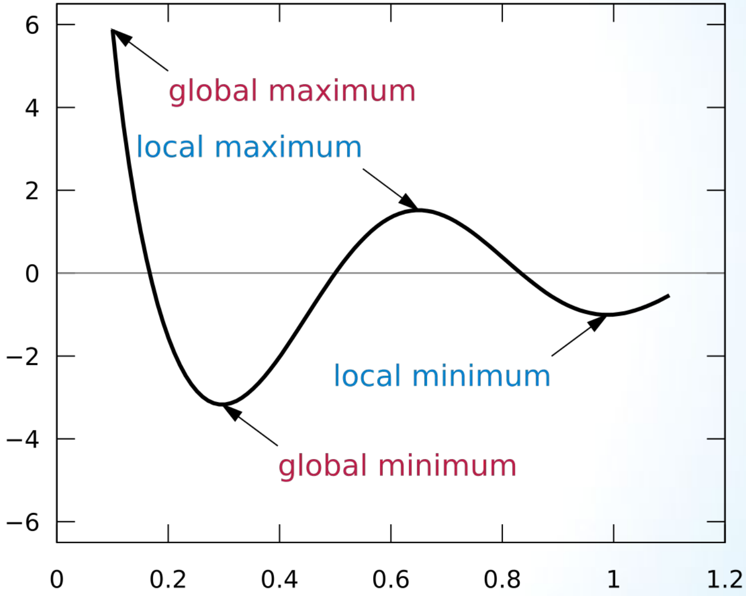
Does Gradient Descent Promise Global Minima?



Does Gradient Descent Promise Global Minima?

No!

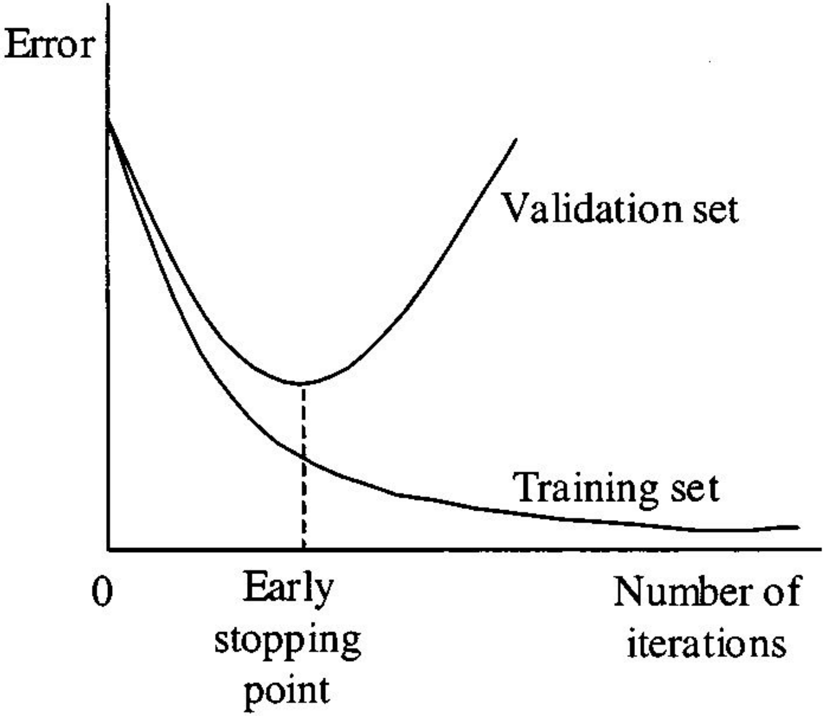
Only when f(x) is convex.



Gradient Descent

1. When do we stop?
    a. Fixed
    b. Early Stopping
        i. When the gradient gets close to zero
        ii. When the loss stops changing much
        iii. When the parameters stop changing much
        iv. When performance on held-out validation set plateaus

Early Stopping

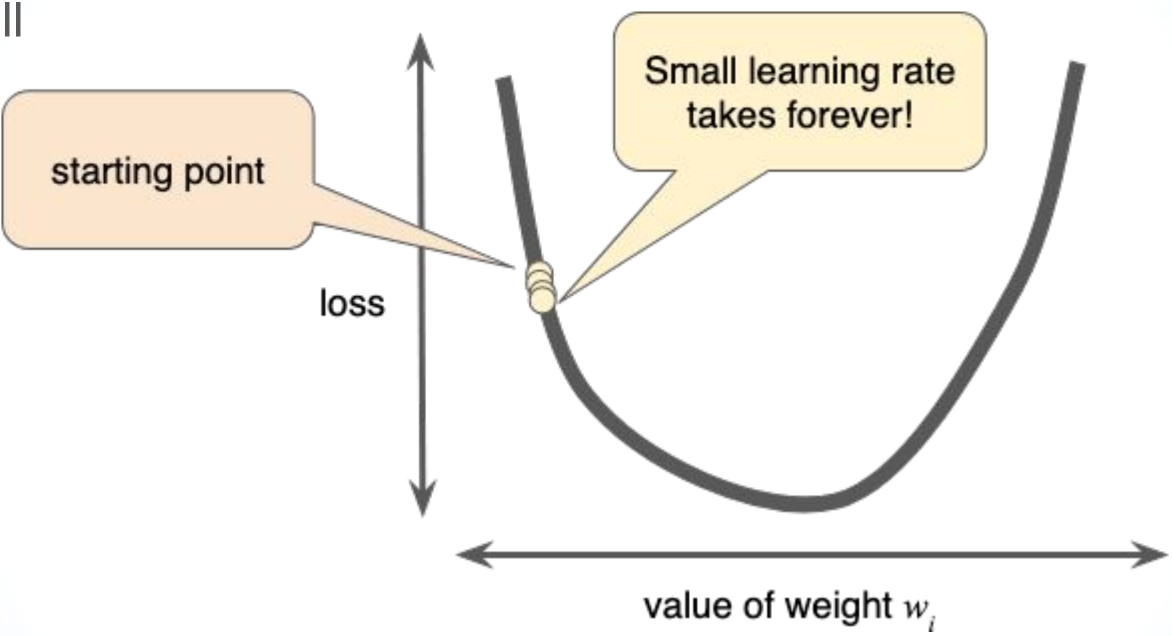


Gradient Descent

2. How do we choose step size aka learning rate? 📝

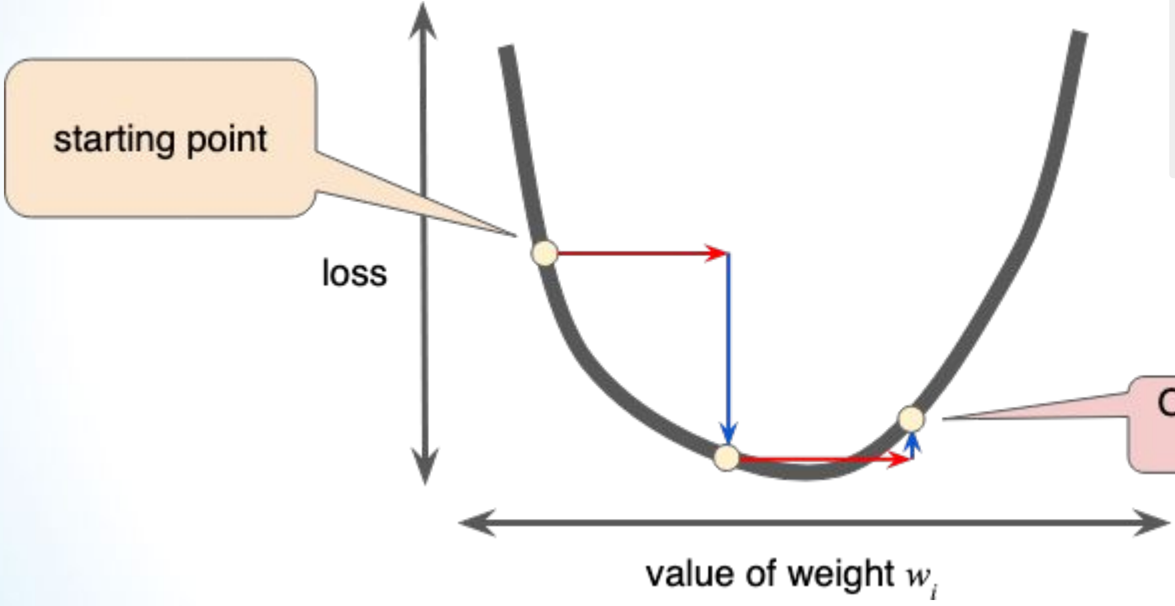
Learning Rate

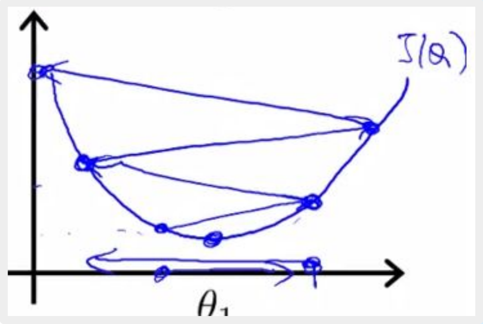
LR is too small



Learning Rate

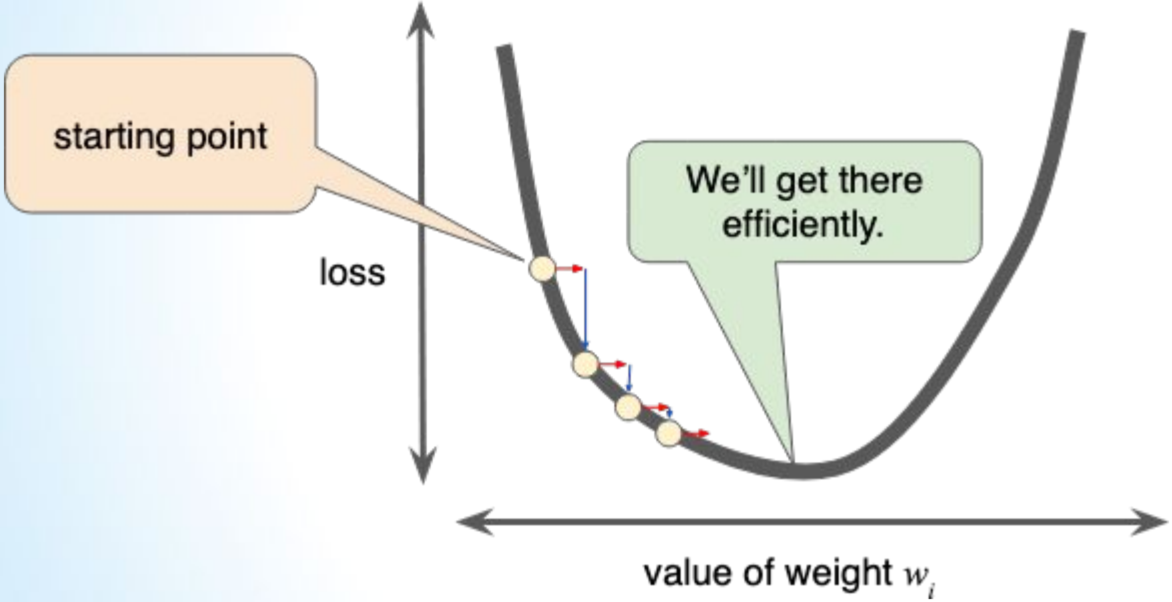
LR is too large





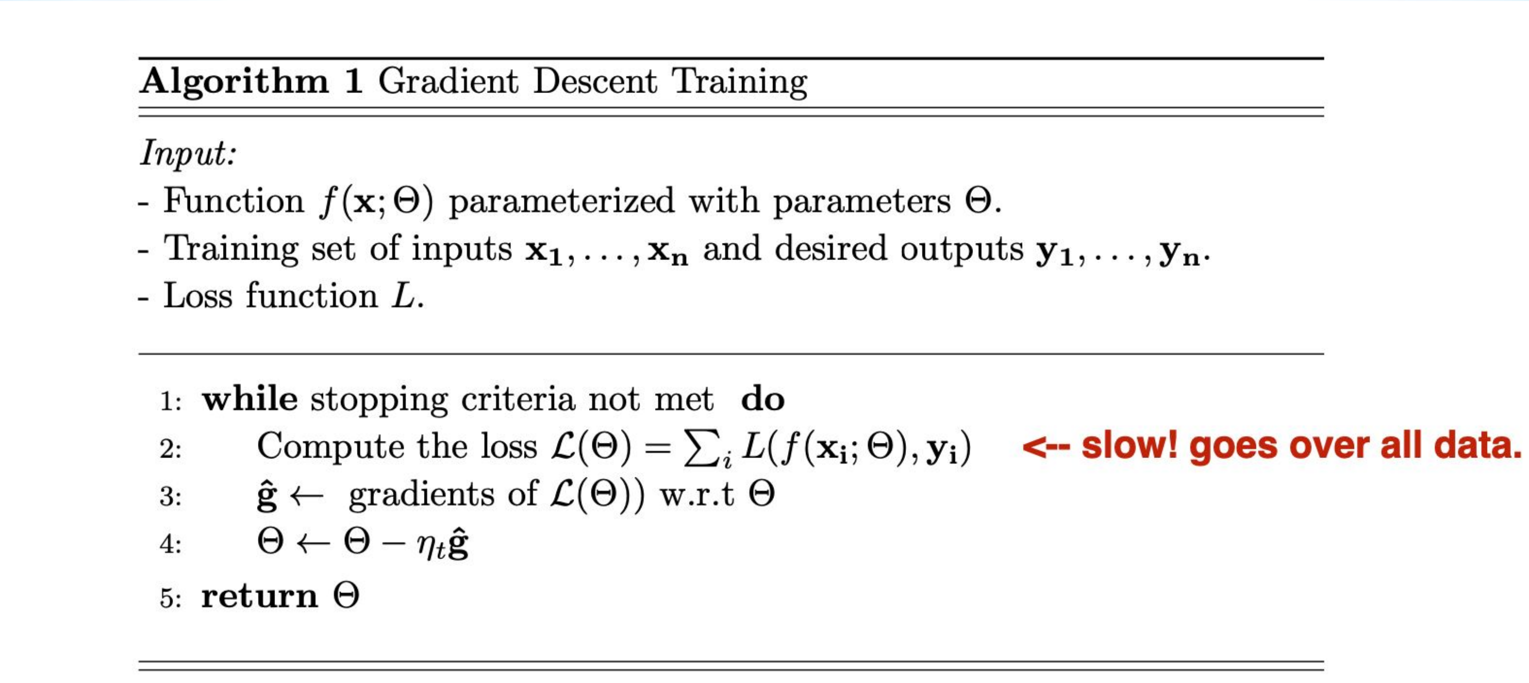
Learning Rate

Learning rate is optimal



How can we choose the best LR?
1. Start with Large LR and reduce in each step
2. Use validation set to empirically choose the LR

Gradient Descent Training 📝



Algorithm 1 Gradient Descent Training

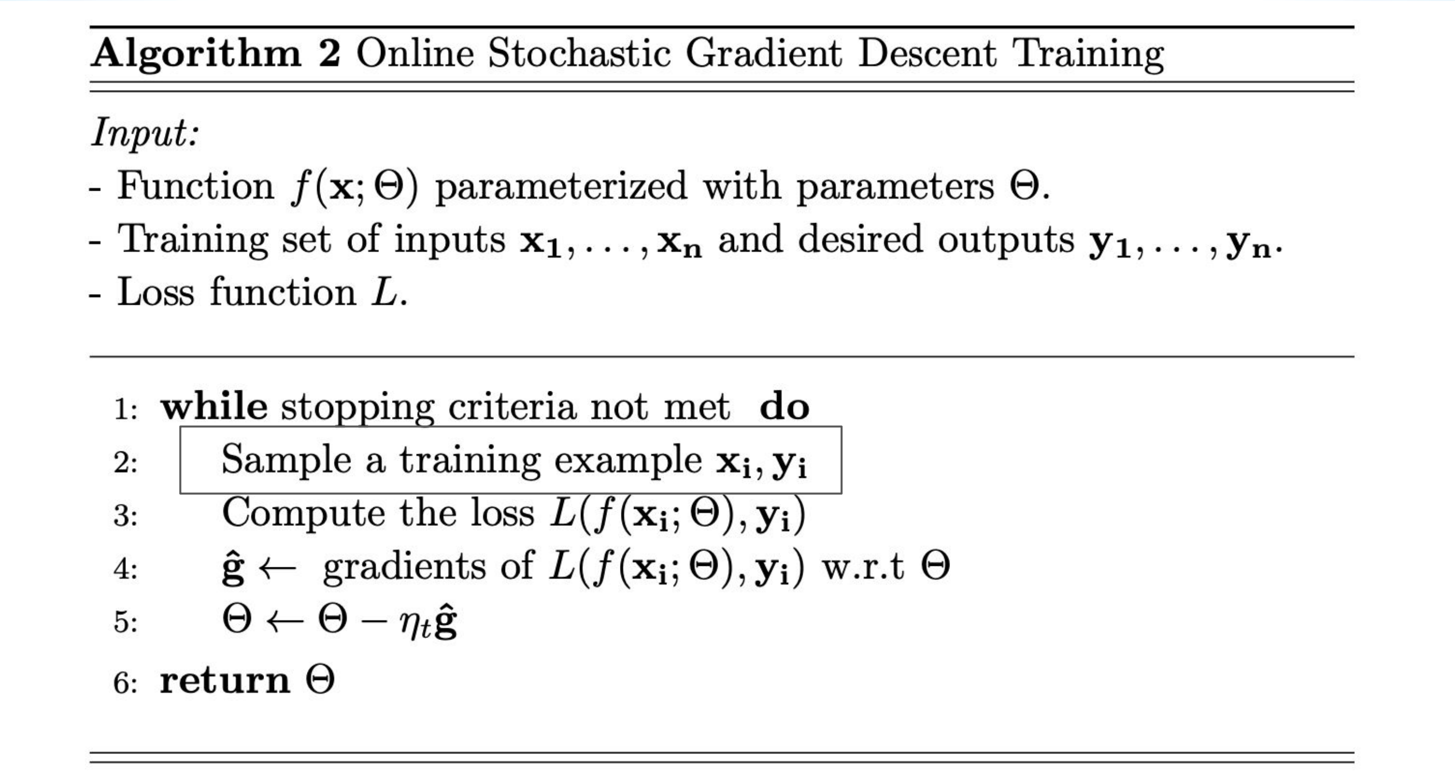
Input:

- Function $f(\mathbf{x};\Theta)$ parameterized with parameters Θ.
- Training set of inputs $\mathbf{x_1},\dots,\mathbf{x_n}$ and desired outputs $\mathbf{y_1},\dots,\mathbf{y_n}$.
- Loss function L.

```
while stopping criteria not met do
    Compute the loss $\mathcal{L}(\Theta) = \sum_i L(f(\mathbf{x_i};\Theta),\mathbf{y_i})$
    $\hat{\mathbf{g}} \leftarrow$ gradients of $\mathcal{L}(\Theta))$ w.r.t $\Theta$
    $\Theta \leftarrow \Theta - \eta_t \hat{\mathbf{g}}$
return $\Theta$
```

<-- slow! goes over all data.

Stochastic Gradient Descent Training 📝



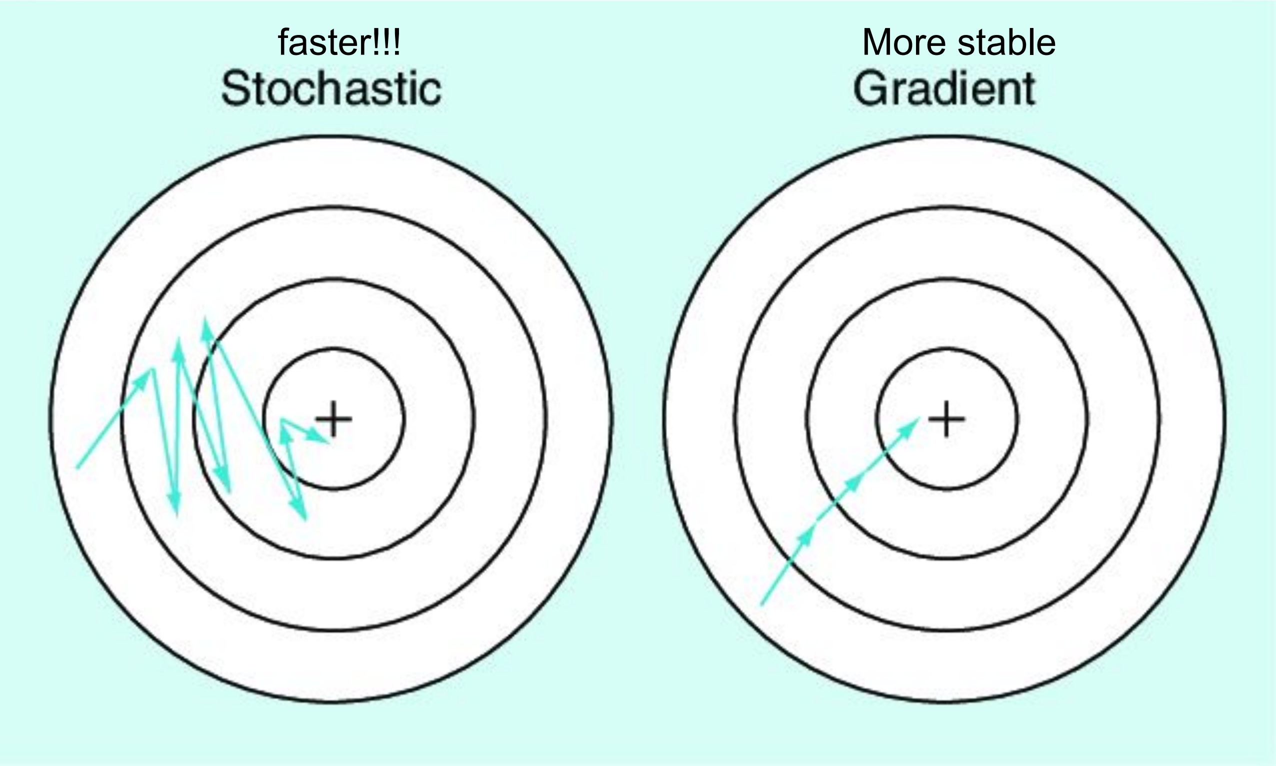
Algorithm 2 Online Stochastic Gradient Descent Training

Input:

- Function $f(\mathbf{x}; \Theta)$ parameterized with parameters Θ.
- Training set of inputs $\mathbf{x_1}, \ldots, \mathbf{x_n}$ and desired outputs $\mathbf{y_1}, \ldots, \mathbf{y_n}$.
- Loss function L.

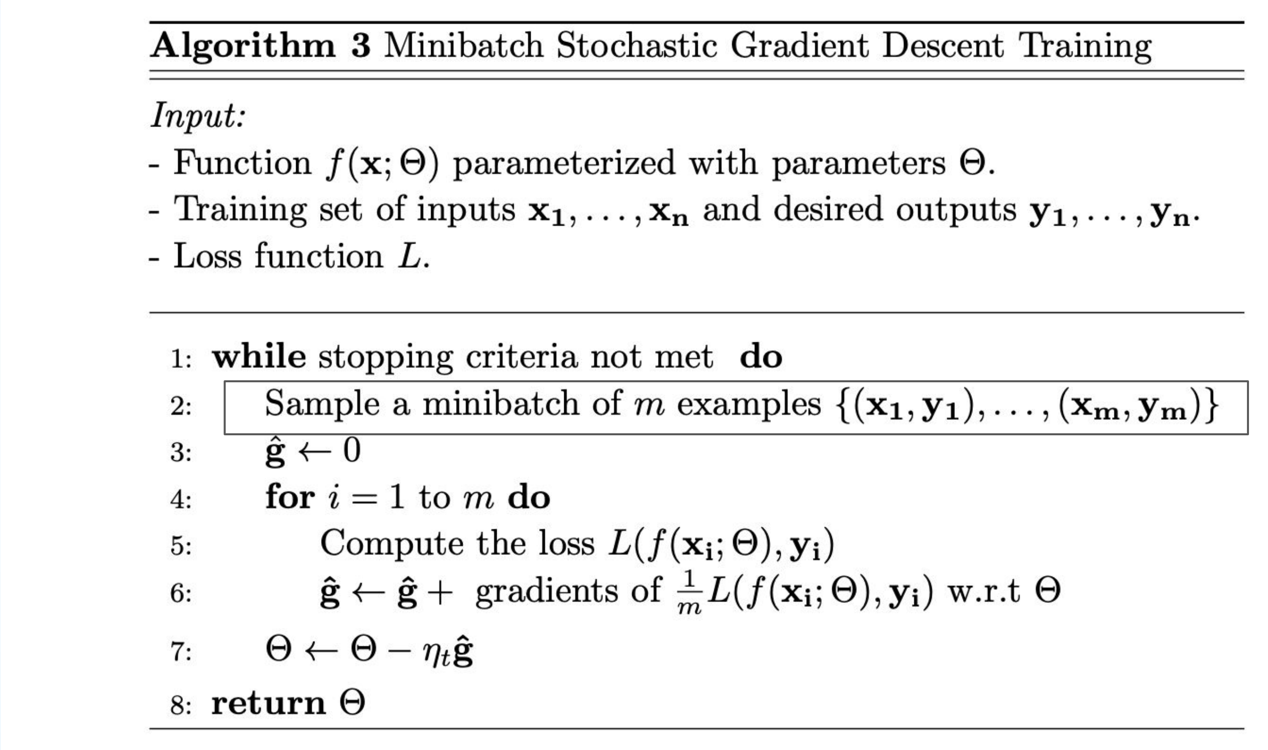
```
while stopping criteria not met do
    $\boxed{\text{Sample a training example } \mathbf{x_i}, \mathbf{y_i}}$
    Compute the loss $L(f(\mathbf{x_i}; \Theta), \mathbf{y_i})$
    $\hat{\mathbf{g}} \leftarrow$ gradients of $L(f(\mathbf{x_i}; \Theta), \mathbf{y_i})$ w.r.t $\Theta$
    $\Theta \leftarrow \Theta - \eta_t \hat{\mathbf{g}}$
return $\Theta$
```

Gradient Descent vs Stochastic Gradient Descent



How can we combine both?

Mini-batch Gradient Descent Training



Algorithm 3 Minibatch Stochastic Gradient Descent Training

Input:

- Function $f(\mathbf{x};\Theta)$ parameterized with parameters Θ.
- Training set of inputs $\mathbf{x_1},\dots,\mathbf{x_n}$ and desired outputs $\mathbf{y_1},\dots,\mathbf{y_n}$.
- Loss function L.

```
while stopping criteria not met do
    Sample a minibatch of $m$ examples $\{(\mathbf{x_1},\mathbf{y_1}),\dots,(\mathbf{x_m},\mathbf{y_m})\}$
    $\hat{\mathbf{g}} \leftarrow 0$
    for $i = 1$ to $m$ do
        Compute the loss $L(f(\mathbf{x_i};\Theta),\mathbf{y_i})$
        $\hat{\mathbf{g}} \leftarrow \hat{\mathbf{g}} +$ gradients of $\frac{1}{m}L(f(\mathbf{x_i};\Theta),\mathbf{y_i})$ w.r.t $\Theta$
    $\Theta \leftarrow \Theta - \eta_t\hat{\mathbf{g}}$
return $\Theta$
```

Gradient Descent Toy Example

Problem - Two Iterations of GD

3 inputs: x1, x2, x3 (and x0=1)

1 output: y

Initial weights: w0 = – 0.7, w1 = 0.2, w2 = 0.7, w3 = 0.9

The true output: 0

The learning rate: 0.2

Let’s say the gradient is based on the perceptron rule: (y pred - y true) x

Show 2 iterations of gradient descent

Solution First Iteration

X0 = 1, x1 = 0, x2 = 0, x3 = 1
Y true = 0

Iteration 0:

W0 = -0.7, w1 = 0.2, w3 = 0.7, w4 = 0.9

1. Original output:

   Wx = w0*x0 + w1*x1 + w2*x2 + w3*x3

   = -0.7*1 + 0.2*0 + 0.7*0 + 0.9*1 = 0.2 > 0 → **Y observed = 1**
2. Let’s calc the gradient:
   a. g0 = (o - t) * x0 = (1 - 0) * 1 = 1
      g1 = (o - t) * x1 = (1 - 0) * 0 = 0
      g2 = (o - t) * x2 = (1 - 0) * 0 = 0
      g3 = (o - t) * x3 = (1 - 0) * 1 = 1
3. Update weights:
   a. w0 = -0.7 - (0.2 * 1) = -0.9
      w1 = 0.2 - (0.2 * 0) = 0.2
      w2 = 0.7 - (0.2 * 0) = 0.7
      w3 = 0.9 - (0.2 * 1) = 0.7

$$w^t \leftarrow w^{t-1} - \underset{0.2}{\eta} \cdot \frac{\partial \ell}{\partial w^{t-1}}$$

Solution Second Iteration

1. Original output:

….

X0 = 1, x1 = 0, x2 = 0, x3 = 1
Y true = 0

Iteration 1:
W0 = **-0.9**, w1 = 0.2, w3 = 0.7, w4 = **0.7**

4. New output:

wx = 1 * (-0.9) + 0 * 0.2 + 0 * 0.7 + 1* 0.7= -0.2 < 0 → **Y observed = 0**

$$w^t \leftarrow w^{t-1} - \underset{0.2}{\eta} \cdot \frac{\partial \ell}{\partial w^{t-1}}$$

Solution Second Iteration → No Change

1. Original output:

   wx = 1 * (-0.9) + 0 * 0.2 + 0 * 0.7 + 1* 0.7= -0.2 < 0 →**Y observed = 0**

2. Let’s calc the gradient:
   a. g0 = (o - t) * x0 = (0 - 0) * 1 = 0
      g1 = (o - t) * x1 = (0 - 0) * 0 = 0
      g2 = (o - t) * x2 = (0 - 0) * 0 = 0
      g3 = (o - t) * x3 = (0 - 0) * 1 = 0

3. Update weights:
   a. w0 = **-0.9** - (0.2 * 0) = **-0.9**
      w1 = **0.2** - (0.2 * 0) = **0.2**
      w2 = **0.7** - (0.2 * 0) = **0.7**
      w3 = **0.7** - (0.2 * 0) = **0.7**

X0 = 1, x1 = 0, x2 = 0, x3 = 1
Y true = 1

Iteration 2:
W0 = -0.9, w1 = 0.2, w3 = 0.7, w4 = 0.7

Convergence ! Gradient and weights didn’t change.

Next Week

- Regularization
- Linear Regression
- Logistic Regression
- Code
- Summary