Deep Learning Using TensorFlow

Dr. Ash Pahwa

Section 2.2: Problems with Gradient Descent Algorithm

Outline

- Problems with Gradient Descent (GD) Algorithm
- Solutions to GD Algorithm Problem
- Vanishing Gradient Problem of Activation Functions

What is Gradient Descent Algorithm?

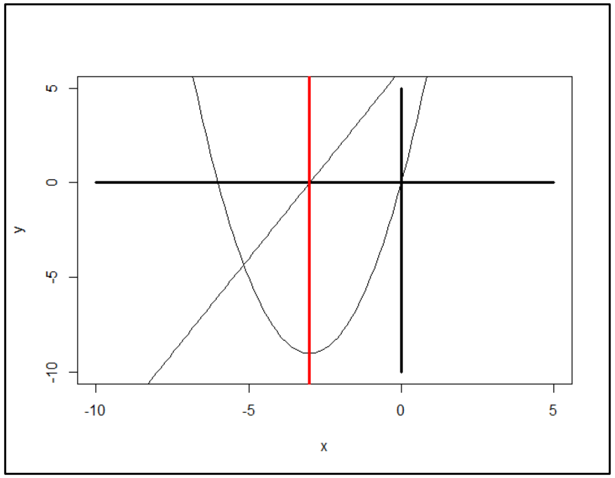
- Suppose a function $y = f(x)$ is given
  - Gradient Descent algorithm allows us to find the values of 'x' where the 'y' value becomes minimum or maximum values
- The Gradient Descent algorithm can be extended to any function with 2 or more variables $'z = f(x, y)'$

- Function y=f(x)
- Gradient Descent Algorithm: Minimum
  - Initialize the value of x
  - Learning rate = η
  - While NOT converged:
    - $x^{t+1} \leftarrow x^t - \eta \frac{\partial y}{\partial x} \|_{x^t}$

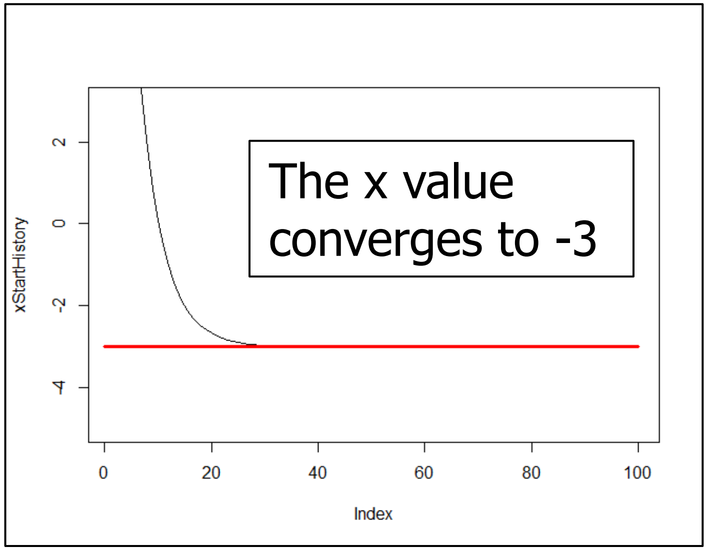
- Function z=f(x,y)
- Gradient Descent Algorithm: Minimum
  - Initialize the value of x and y
  - Learning rate = η
  - While NOT converged:
    - $x^{t+1} \leftarrow x^t - \eta \frac{\partial z}{\partial x} \|_{x^t, y^t}$
    - $y^{t+1} \leftarrow y^t - \eta \frac{\partial z}{\partial y} \|_{x^t, y^t}$

Example 1

- $y = x^2 + 6x$
- To find minimum point, equate the first derivative = 0
  - $\frac{dy}{dx} = 2x + 6 = 0$
  - $x = -3$
- Gradient Descent Algorithm
  - Initialize the value of x
  - Learning rate = η
  - While NOT converged:
    - $x^{t+1} \leftarrow x^t - \eta \frac{\partial y}{\partial x} \|_{x^t}$

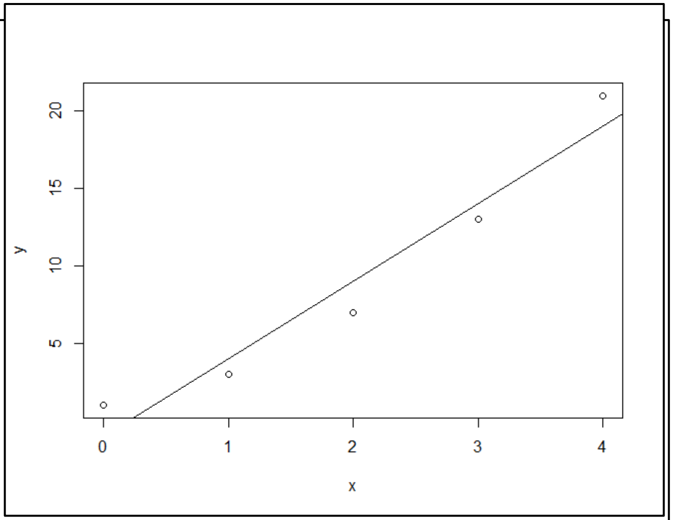



```
> dy_dx = function (w1) { 2*w1 + 6 }
> xStart = 20
> learningRate = 0.1
> maxLimit = 100
> xStartHistory = rep(0,maxLimit)
> for ( i in 1:maxLimit )
+ {
+   xStartHistory[i] = xStart
+   xStart = xStart - learningRate * dy_dx(xStart)
+ }
> plot(xStartHistory,type='l',ylim=c(-5,3))
> lines(c(0,maxLimit),c(-3,-3),lwd=3,col='red')
```



Regression
Using R 'lm' command

```
> x = c(0,1,2,3,4)
> y = c(1,3,7,13,21)
> plot(x,y)
> model = lm(y~x)
> summary(model)

Call:
lm(formula = y ~ x)

Residuals:
 1  2  3  4  5
 2 -1 -2 -1  2

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.0000     1.6733  -0.598  0.59220
x             5.0000     0.6831   7.319  0.00527 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.16 on 3 degrees of freedom
Multiple R-squared:  0.947,   Adjusted R-squared:  0.9293
F-statistic: 53.57 on 1 and 3 DF,  p-value: 0.005268

> abline(model)
>
```



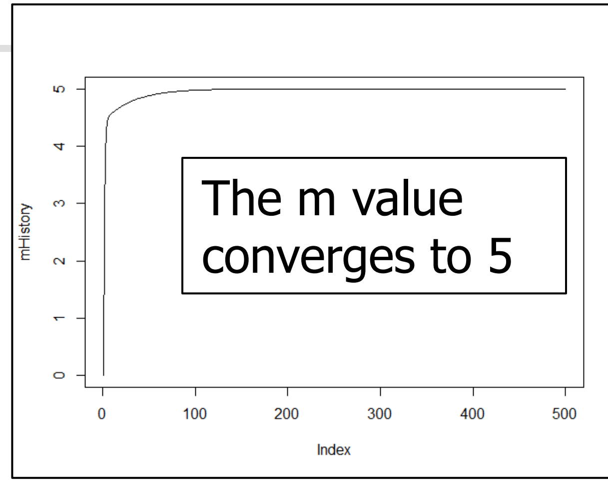
Regression Equation

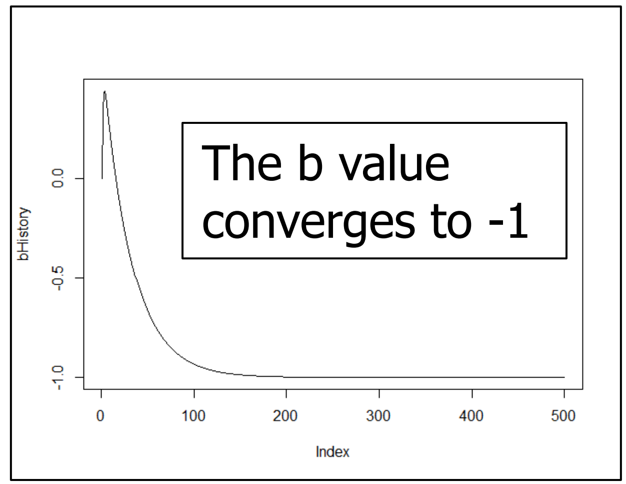
$y = 5x - 1$

Regression
Gradient Descent Algorithm Approach

Regression Equation
$y = 5x - 1$

```
> dRSS_dm = function (m,b) {-2*sum((y-m*x-b)*x)   }
> dRSS_db = function (m,b) { -2*sum(y-m*x-b) }
> mStart = bStart = 0
> learningRate = 0.01;  maxLimit = 500
> mHistory = bHistory = rep(0,maxLimit)
> for ( i in 1:maxLimit )
+ {
+   mHistory[i] = mStart
+   bHistory[i] = bStart
+
+   dW = dRSS_dm(mStart,bStart)
+   db = dRSS_db(mStart,bStart)
+
+   mStart = mStart - learningRate * dW
+   bStart = bStart - learningRate * db
+ }
> plot(mHistory,type='l')
> plot(bHistory,type='l')
```



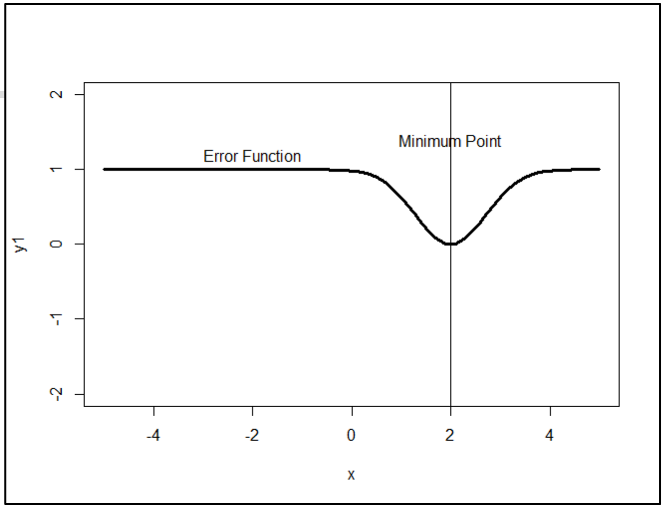


Problems with Gradient Descent Algorithm

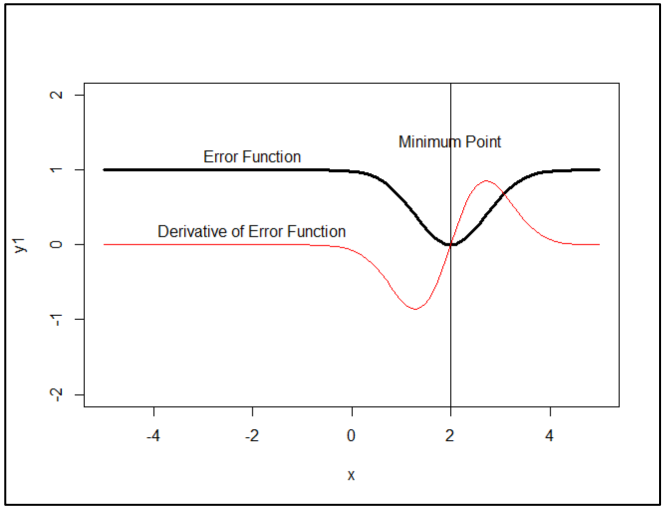
- When the gradient of the error function is low (flat)
  - It takes a long time for the algorithm to converge (reach the minimum point)
- If there are multiple local minima, then
  - There is no guarantee that the procedure will find the global minimum

Error Function

- $Error\ Function:\ y = f(x) = 1 - e^{-(x-2)^2}$
- $\dfrac{dy}{dx} = 2(x-2)e^{-(x-2)^2}$
- The error function 'y' is almost flat
  - $f(x) = 0: -\infty < x < 0\ and$
  - $f(x) = 0: \quad 4 < x < \infty$

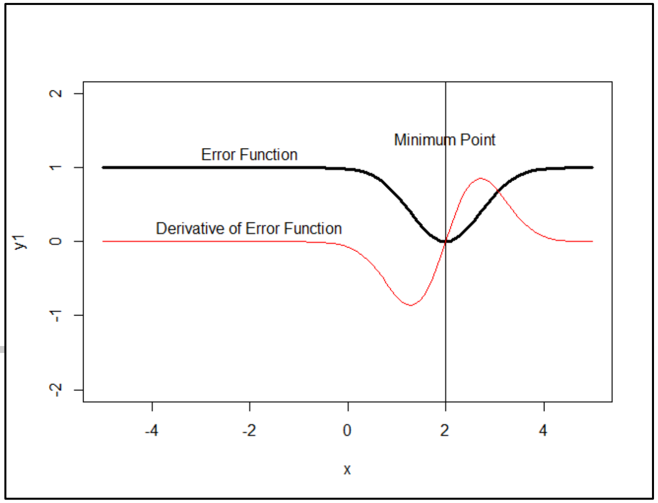



```
> x = seq(-5,5,0.1)
> y1 = 1 - exp(-(x-2)^2)
> plot(x,y1,type='l',ylim=c(-2,2),lwd=3)
> text(-2,1.2,"Error Function")
> text(2,1.4,"Minimum Point")
> abline(v=2)
> dy_dx = function (w1) { 2*(w1-2)*exp(-(w1-2)^2) }
> slope = dy_dx(x)
> points(x,slope,type='l',col='red')
> text(-2,0.2,"Derivative of Error Function")
```

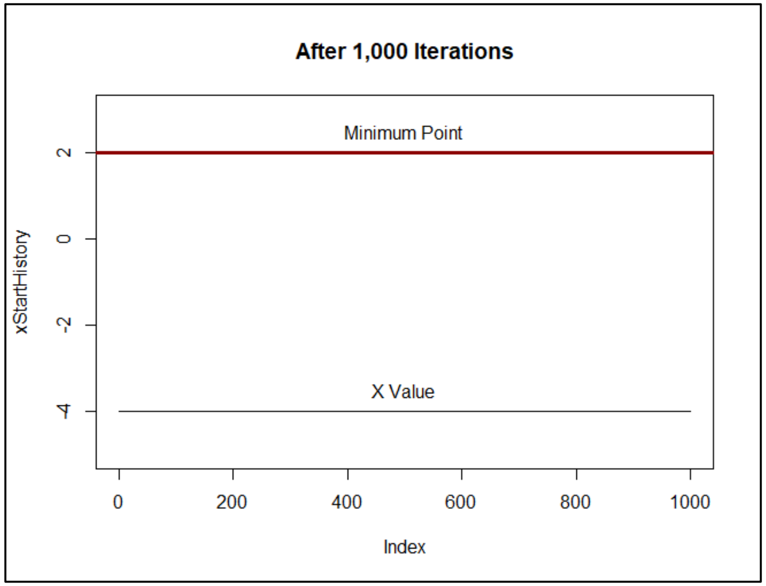


Gradient Descent Algorithm

- Starting Point = -4
- Number of iteration = 1,000

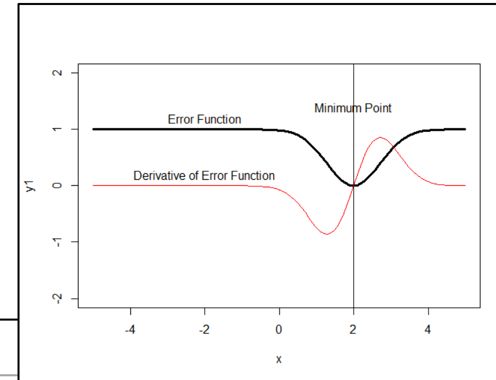


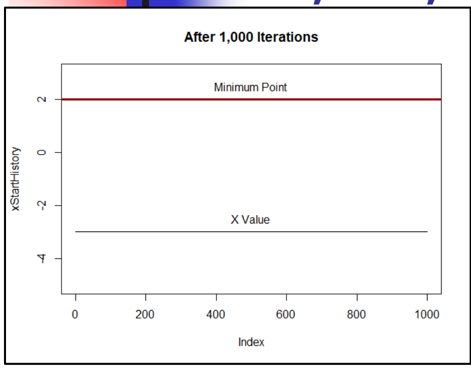

```
> xStart = -4
> learningRate = 0.1
> maxLimit = 1000
> xStartHistory = rep(0,maxLimit)
> for ( i in 1:maxLimit )
+ {
+   xStartHistory[i] = xStart
+   xStart = xStart - learningRate * dy_dx(xStart)
+ }
> plot(xStartHistory,type='l',ylim=c(-5,3),
main="After 1,000 Iterations")
> text(500,-3.5,"X Value")
> abline(h=2, col='dark red', lwd=3)
> text(500,2.5,"Minimum Point")
>
```

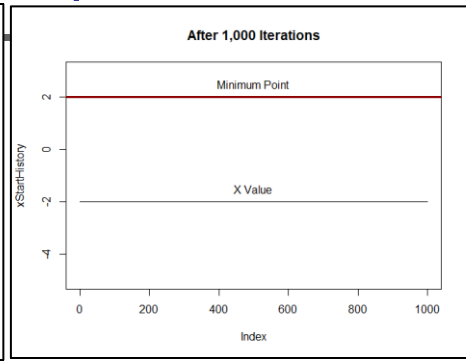


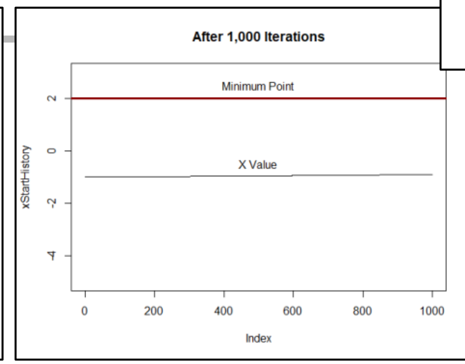
Starting Point = -4, DOES NOT CONVERGE

After 1,000 iteration Starting Point -3, -2, -1, -0.5

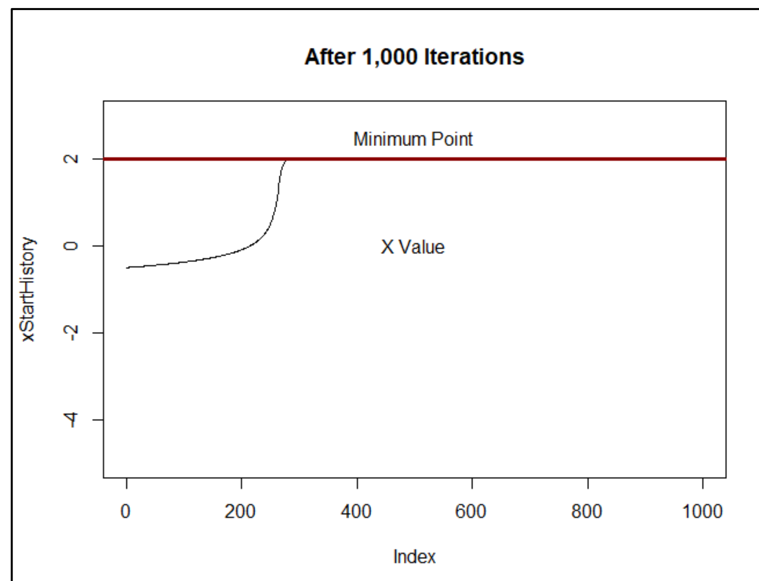








Starting Point = -3, -2, -1: 'x' value DOES NOT CONVERGE

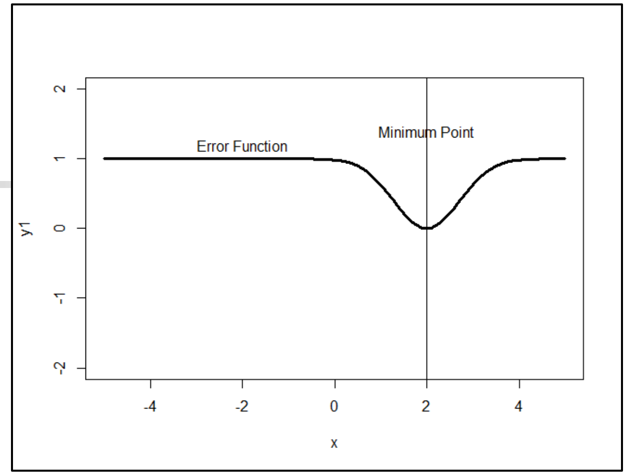


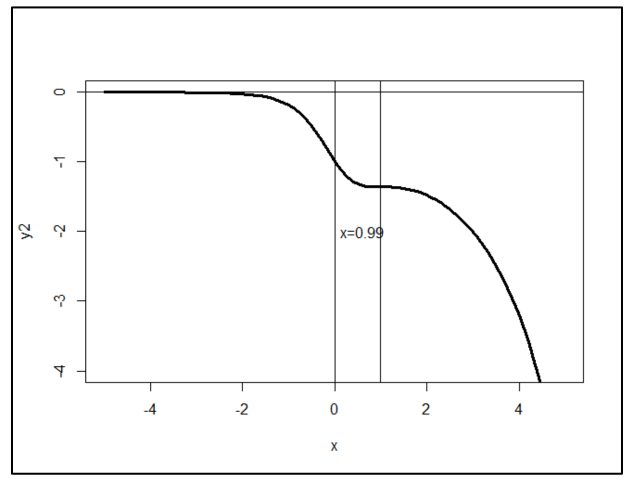
Starting Point = -0.5,

'x' value CONVERGES AFTER 250 ITERATIONS

Problems with Gradient Descent Algorithm

- If the error function has very small gradient
  - Starting point is on the flat surface
  - Gradient Descent Algorithm will take a long time to converge





Solutions to Gradient Descent Algorithm Problem

- Solution#1: Increase the step size
  - Momentum based GD
  - Nesterov GD
- Solution#2: Reduce the data points for approximate gradient
  - Stochastic Gradient Descent
  - Mini Batch Gradient Descent
- Solution#3: Adjust the Learning rate (η)
  - AdaGrad
  - RMS Prop
  - Adam

- Function y=f(x)
- Gradient Descent Algorithm: Minimum
  - Initialize the value of x
  - Learning rate $= \eta$
  - While NOT converged:
    - $x^{t+1} \leftarrow x^{t} - \eta \frac{\partial y}{\partial x} \|_{x^{t}}$

Vanishing Gradient Problem

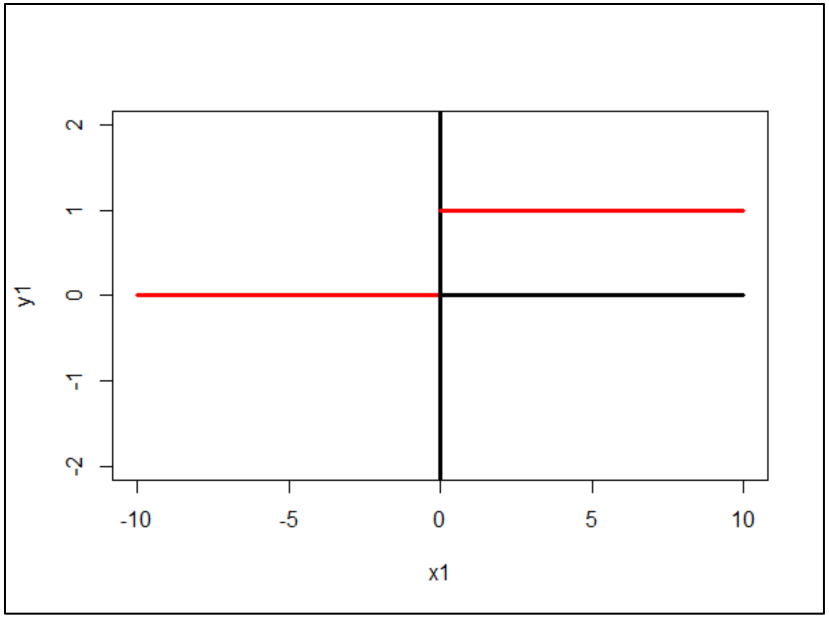
Activation Functions

Types of Activation Functions

1. Binary Step Function
2. Linear Function
3. Sigmoid Function
4. Hyperbolic Tangent Function
5. ReLU (Rectified Linear Unit) Function
6. Leaky ReLU Function
7. Softmax Function

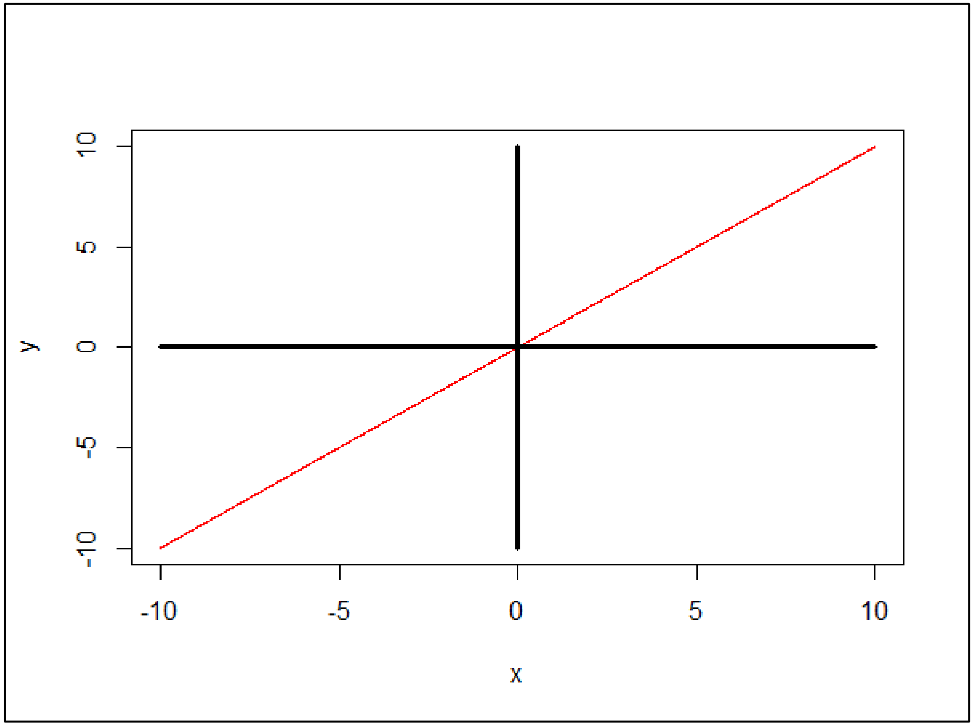
1. Binary Step Function

- $y = f(x) = 0 \; when \; x < 0$
- $y = f(x) = 1 \; when \; x > 0$



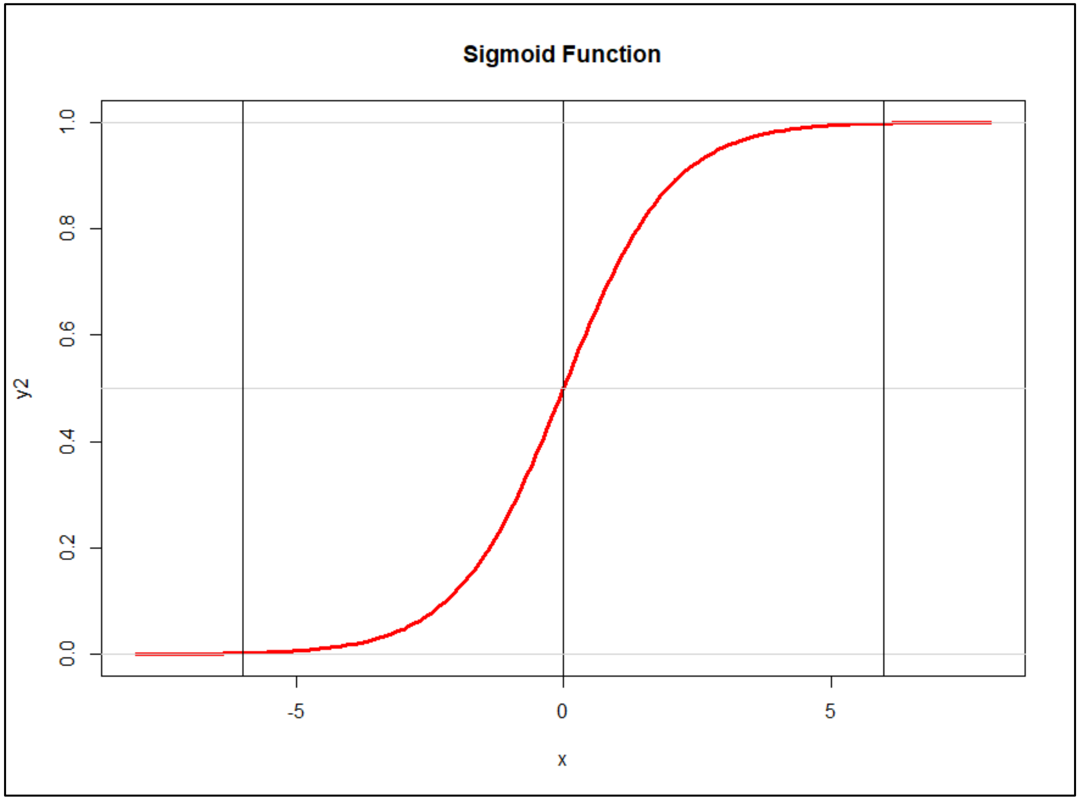
2. Linear Function

- $y = f(x) = x$



3. Sigmoid Function

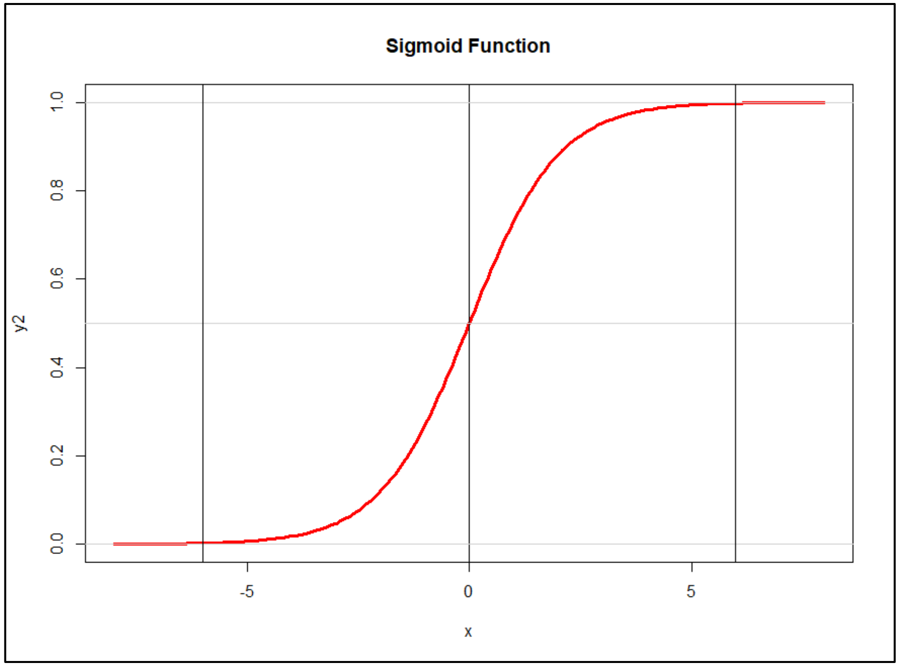
$$f(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$



Sigmoid Function Problems

$$f(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$

- Output of a Sigmoid Function
  - Varies between 0 and 1
  - Gives only positive values
    - Not symmetric around origin
- Gradient very small when
    - (X > 3) & (X < -3)
    - This contributes towards vanishing gradient problem
    - Computations are time consuming and complex
    - It is slow in convergence

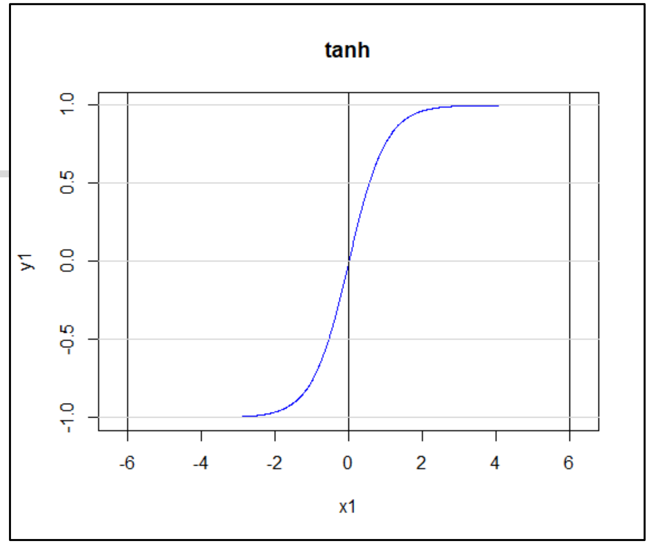


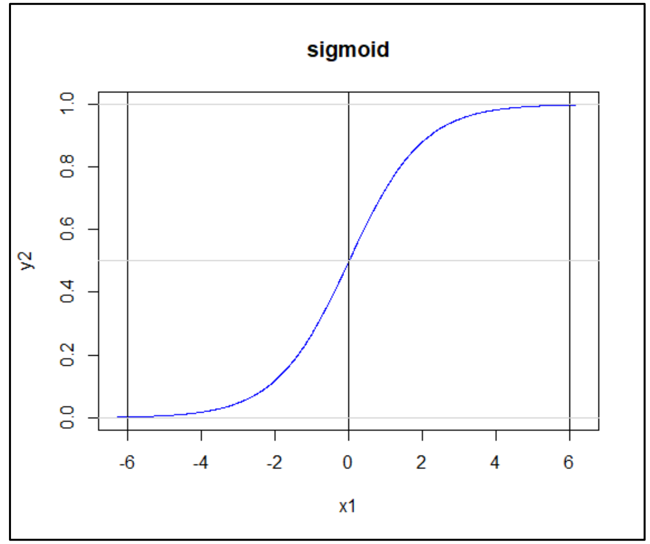
4. Hyperbolic 'tanh' Function

- $\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- Like Sigmoid Function

$$Sigmoid\ Function = f(x) = \frac{1}{1 + e^{-x}}$$

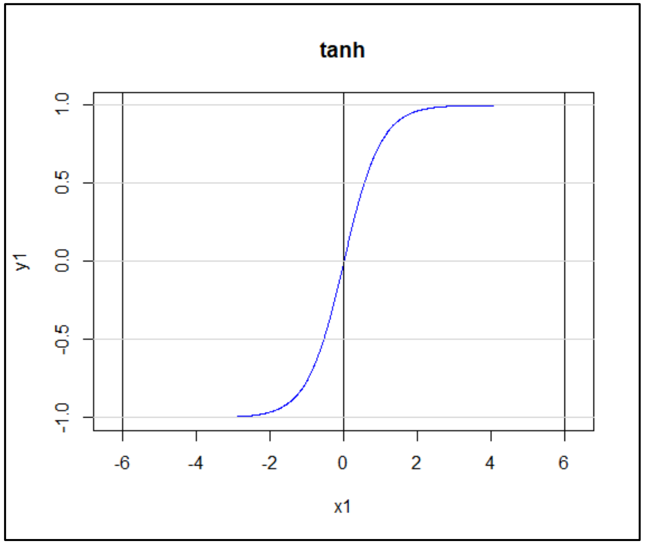
```
> ###########################################
> # Hyperbolic tan function
> # tanh function + Sigmoid
> ############################################
> # tanh Function
> x1 = seq(-2*pi, 2*pi, 0.01)
> y1 = tanh(x1)
> plot(x1,y1,type='l',col="blue",main="tanh")
> abline(v=seq(-6,6,6),col="black",lty=1)
> abline(h=seq(-1,1,0.5),col="lightgrey",lty=1)
> ##########################################
> # Sigmoid Function
> y2 <- 1/(1+exp(-x1))
> plot(x1,y2,type='l',col="blue",main="sigmoid")
> abline(v=seq(-6,6,6),col="black",lty=1)
> abline(h=seq(0,1,0.5),col="lightgrey",lty=1)
> #########################################
```





Hyperbolic 'tanh' Function Problems

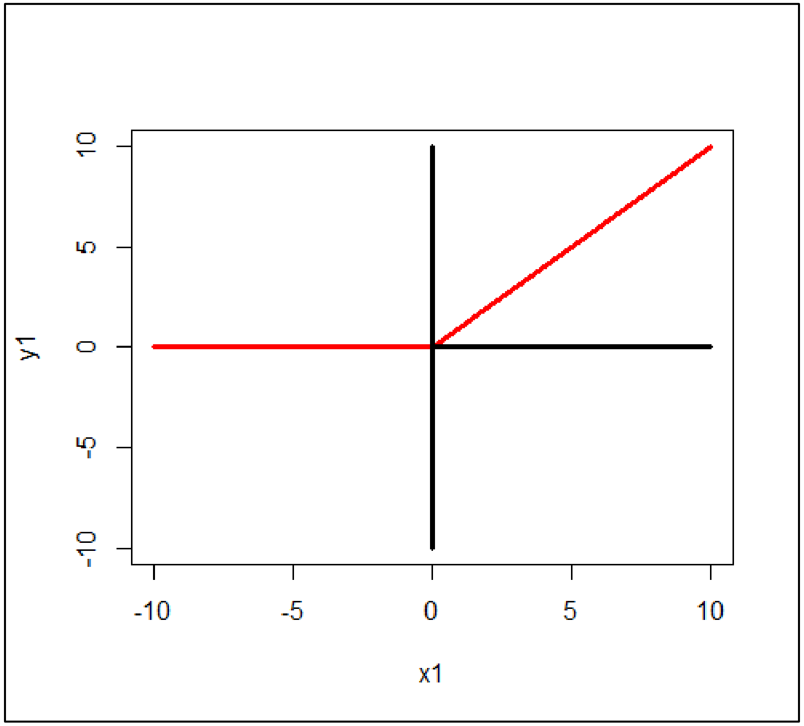
- Output of a 'tanh' Function
  - Varies between -1 and 1
  - Symmetric around origin
- Gradient very small when
  - (X > 3) & (X < -3)
  - This contributes towards vanishing gradient problem



5. Rectified Linear Unit (ReLU) Function

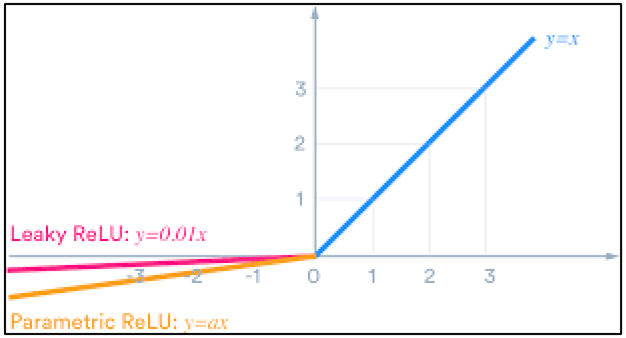
- $ReLU(x) = 0 \; when \; x < 0$
- $ReLU(x) = x \; when \; x \geq 0$
- ------------------------------
- $ReLU(a) = \max(0, a)$

- Results in having sparse Neural Network with fewer neurons
- This makes the Neural Network very efficient



6. Leaky ReLU

- Leaky ReLu allow a small, positive gradient when the unit is not active
- $f(x) = \begin{Bmatrix} x \; if \; x > 0, \\ 0.01x \; otherwise \end{Bmatrix}$

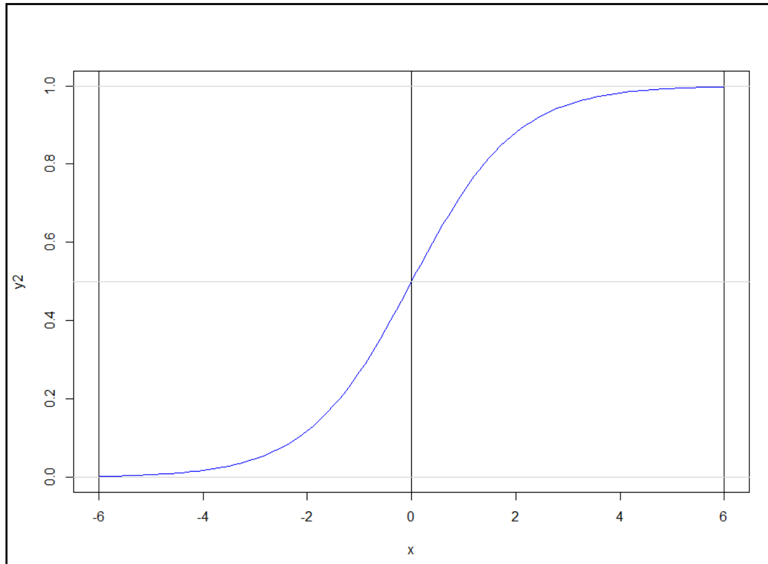


Activation Functions
Sigmoid Function

- Activation Functions
  1. Binary Step Function
  2. Linear Function
  3. Sigmoid Function
  4. Hyperbolic Tangent Function
  5. ReLU (Rectified Linear Unit) Function
  6. Leaky ReLU Function
  7. Softmax Function

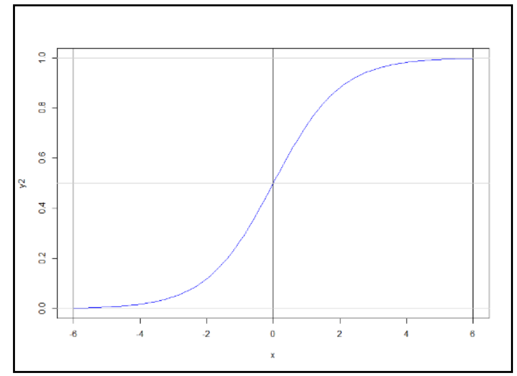
$$f(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$

```
R Code for Sigmoid Function

> x <- seq(-6,6,0.1)
> y1 <- exp(x)/(1+exp(x))
> y2 <- 1/(1+exp(-x))
> plot(x,y2,type='l',col="blue")
> abline(v=seq(-6,6,6),col="black",lty=1)
> abline(h=seq(0,1,0.5),col="lightgrey",lty=1)
```



Derivative of Sigmoid Function

$$f(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$

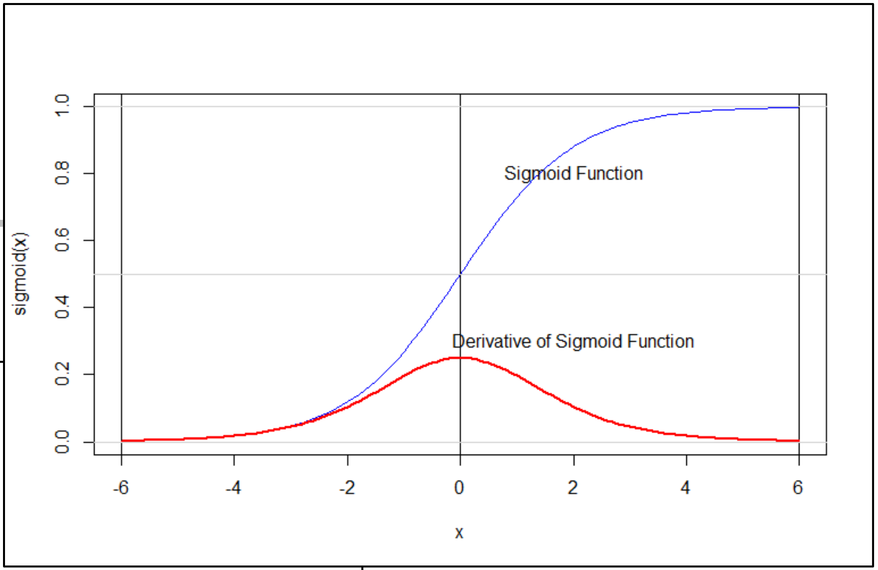


- $\sigma(x) = \frac{1}{1+e^{-x}}$
- $\frac{d\,\sigma(x)}{dx} = \frac{d}{dx}\frac{1}{(1+e^{-x})} = \frac{(1+e^{-x})\frac{d}{dx}(1) - 1\frac{d}{dx}(1+e^{-x})}{(1+e^{-x})^2}$ *Apply Quotient Rule*
- $\frac{d\,\sigma(x)}{dx} = \frac{0-(-1)e^{-x}}{(1+e^{-x})^2} = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{e^{-x}+1-1}{(1+e^{-x})^2} = \frac{1+e^{-x}}{(1+e^{-x})^2} - \frac{1}{(1+e^{-x})^2}$
- $\frac{d\,\sigma(x)}{dx} = \frac{1}{1+e^{-x}} - \left(\frac{1}{1+e^{-x}}\right)^2$
- $\frac{d\,\sigma(x)}{dx} = \sigma(x) - (\sigma(x))^2$
- $\sigma'(x) = \sigma(x)(1-\sigma(x))$

Plot of Sigmoid Function and its Derivative

$$\sigma(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$

$$\sigma'(x) = \sigma(x)(1-\sigma(x))$$



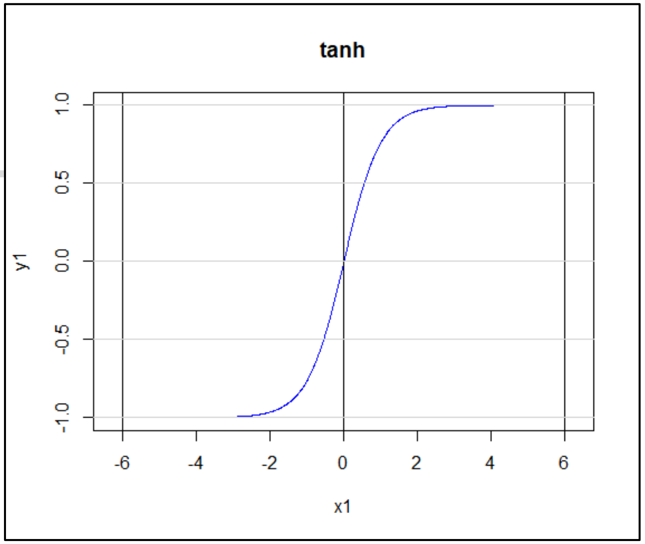

```
> ########################################
> # Plot of the derivative of Sigmoid function
> #########################################
>
> sigmoid = function (x) { 1/(1+exp(-x)) }
>
> der.sigmoid = function (x) { sigmoid(x) * ( 1 - sigmoid(x)) }
>
> x <- seq(-6,6,0.1)
>
> plot(x,sigmoid(x), type='l',col="blue")
>
> abline(v=seq(-6,6,6),col="black",lty=1)
> abline(h=seq(0,1,0.5),col="lightgrey",lty=1)
>
> lines(x,der.sigmoid(x), type='l',col="red",lwd=2)
>
> text(2,0.8,"Sigmoid Function")
> text(2,0.3,"Derivative of Sigmoid Function")
```

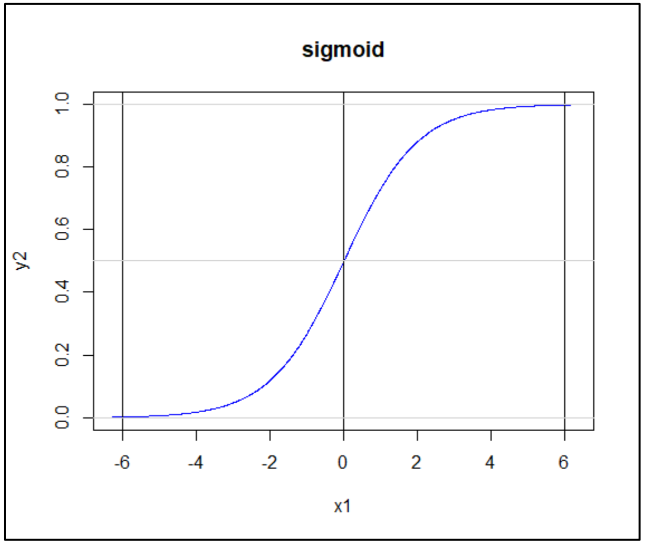
Hyperbolic 'tanh' function

- $\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- Similar to Sigmoid Function

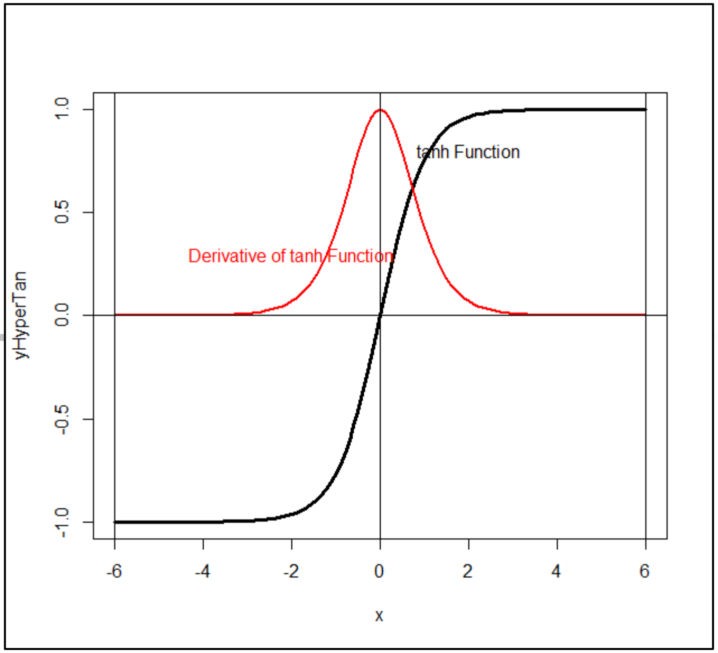
$$Sigmoid\ Function = f(x) = \frac{1}{1 + e^{-x}}$$

```
> ##########################################
> # Hyperbolic tan function
> # tanh function + Sigmoid
> ###########################################
> # tanh Function
> x1 = seq(-2*pi, 2*pi, 0.01)
> y1 = tanh(x1)
> plot(x1,y1,type='l',col="blue",main="tanh")
> abline(v=seq(-6,6,6),col="black",lty=1)
> abline(h=seq(-1,1,0.5),col="lightgrey",lty=1)
> #########################################
> # Sigmoid Function
> y2 <- 1/(1+exp(-x1))
> plot(x1,y2,type='l',col="blue",main="sigmoid")
> abline(v=seq(-6,6,6),col="black",lty=1)
> abline(h=seq(0,1,0.5),col="lightgrey",lty=1)
> ########################################
```





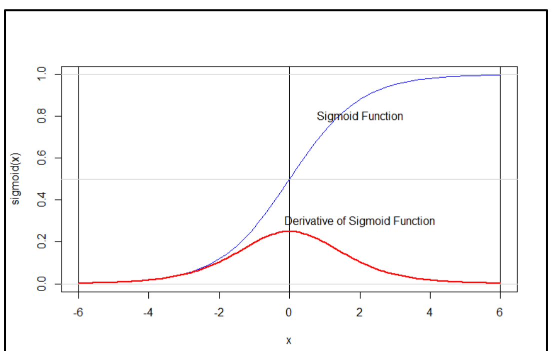
Hyperbolic Tangent

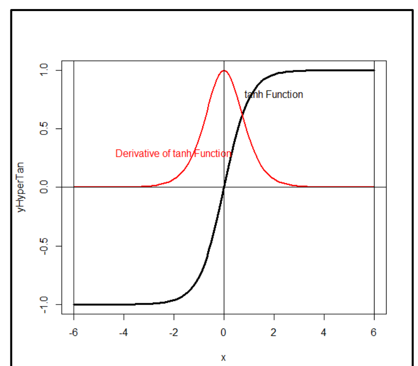


- $g(Z) = \tanh(Z) = \frac{\sinh(Z)}{\cosh(Z)} = \frac{e^{Z} - e^{-Z}}{e^{Z} + e^{-Z}}$
- $\frac{d}{dZ} \tanh(Z) = \frac{d}{dZ} \frac{\sinh(Z)}{\cosh(Z)}$
- $\frac{\cosh(Z) * \frac{d}{dZ} \sinh(Z) - \sinh(Z) * \frac{d}{dZ} \cosh(Z)}{\cosh(Z)^2}$ *Apply Quotient Rule*
- $\frac{\cosh(Z) * \cosh(Z) - \sinh(Z) * \sinh(Z)}{\cosh(Z)^2} = \frac{\cosh(Z)^2 - \sinh(Z)^2}{\cosh(Z)^2} = 1 - tanh(Z)^2$
- $\frac{d}{dZ} \tanh(Z) = 1 - tanh(Z)^2$

Vanishing Gradient Problem

Function	Function Definition	Gradient of Activation Function	Gradient computed at x = -4.0 x = 0.5 x = 4.0
Sigmoid	$\sigma(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$	$\sigma'(x) = \sigma(x)(1-\sigma(x))$	x = -4.0, Gradient=0.0176 x = 0.5, Gradient=0.235 x = 4.0, Gradient=0.0176
Tanh	$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$\frac{d}{dZ}\tanh(Z) = 1 - tanh(Z)^2$	x = -4.0, Gradient=0.0013 x = 0.5, Gradient=0.7864 x = 4.0, Gradient=0.0013
ReLu	$f(x) = x$	1 for x > 0	x = -4.0, Gradient=0 x = 0.5, Gradient=1 x = 4.0, Gradient=1





Summary

- Problems with Gradient Descent (GD) Algorithm
- Solutions to GD Algorithm Problem
- Vanishing Gradient Problem of Activation Functions