ECS 171 Machine Learning

Lecture4: Logistic Regression, Newton’s method, Perceptron Model

Instructor: Dr. Setareh Rafatirad

MLE for LR Recap

$Sample: \; X_1, X_2, \ldots, X_n$

$y: output\ (response\ variable)$

$$y = \alpha + \beta x$$

when residual error $\epsilon = 0$.

$$y = \overbrace{\alpha + \beta x}^{f(x)} + \epsilon$$

Random noise

Model this error following a distribution

$$\epsilon \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$$

MLE for LR Recap

$$y = f(x) + \epsilon \qquad \epsilon \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$$

$f(x) \to \boldsymbol{\alpha + \beta x}$

$\sigma^2 \to$ ***variance***

$Sample: \ X_1, X_2, \ldots, X_n$

$For\ a\ fixed\ sample: \ X_i, the\ distribution\ of\ Y_i is\ equal\ to$:

$$\underbrace{\mathcal{N}(f(X_i), \sigma^2)}$$

$f(X_i) \to$ ***mean***, $\sigma^2 \to$ ***variance***

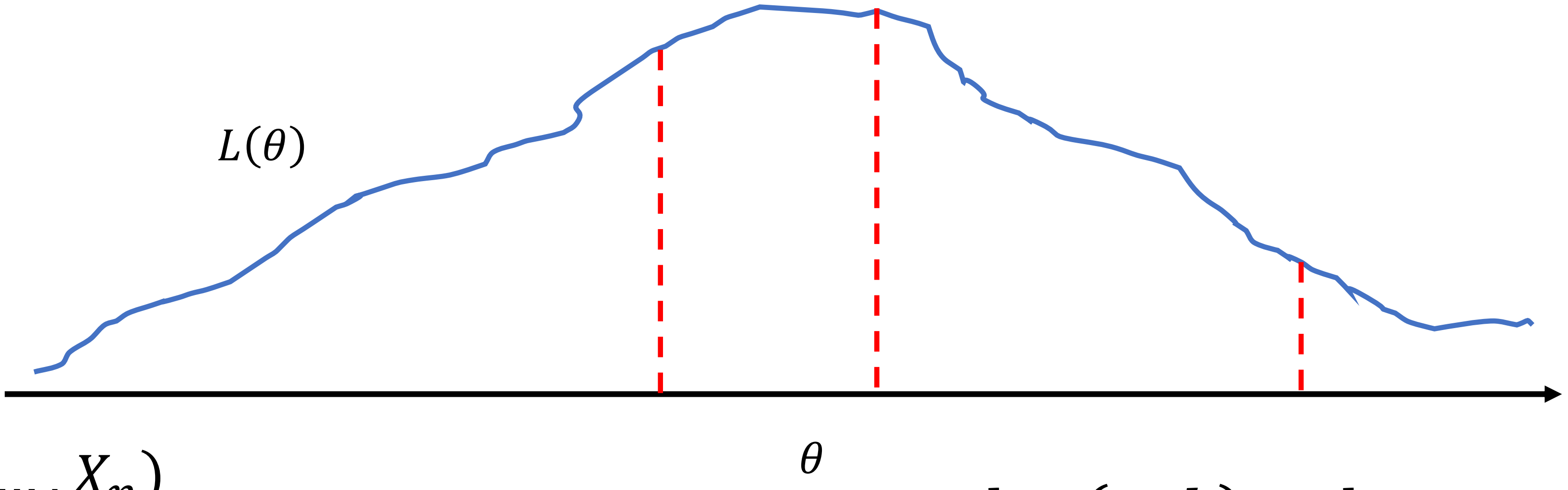
$f(X) = w^T X \to \mathcal{N}(w^T X, \sigma^2)$

p.d.f

$$f(y) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(y-\mu)^2}{2\sigma^2}} \Rightarrow f(y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-f(x_i))^2}{2\sigma^2}}$$

Likelihood Function Recap

$L(\theta), where\ \theta\ is\ the\ unknown\ parameter$



$L(\theta|X_1, \ldots, X_n)$

$= f(X_1, \ldots, X_n |\ \theta)$

$= f(X_i |\ \theta)$

$= f(X_1 |\ \theta) \ldots f(X_n |\ \theta)$

$= \prod_{i=1}^{n} f(X_i |\ \theta)$

$\log(a.b) = \log a + \log b$

$L(\theta|X_1, \ldots, X_n) \cong\ -\ log\ \ L(\theta|X_1, \ldots, X_n)$

Negative Log Likelihood (NLL)

MLE Recap

$$L(\theta) \triangleq -\frac{N}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(Y_i - w^T X_i)^2$$

$$RSS(w) \triangleq \sum_{i=1}^{N}(Y_i - w^T X_i)^2 = ||\epsilon||^2$$

sum of squared errors (SSE)

squared norm of residual errors.

$$NLL(\theta) \triangleq -\log f(X_i \mid \theta) = -\sum_{i=1}^{N}\log p(Y_i \mid X_i, \theta)$$

Logistic Regression

Logistic Regression or Linear Regression?

- Classification problem
- Example
  - Spam detection (1 or 0)
  - Tumor detection (1 or 0)

$if\ the\ hypothesis\ is\ a\ linear\ regression\ model:$

$$Z = w^T x$$

$$y^{(i)} = \begin{cases} 0 & ; predicted\ value < thresold \\ 1 & ;\ predicted\ value\ \geq threshold \end{cases}$$

Drawbacks of using Linear Regression for classification (predicting labels):

- **Sensitivity to outliers**
- **Sensitivity to the selected threshold**

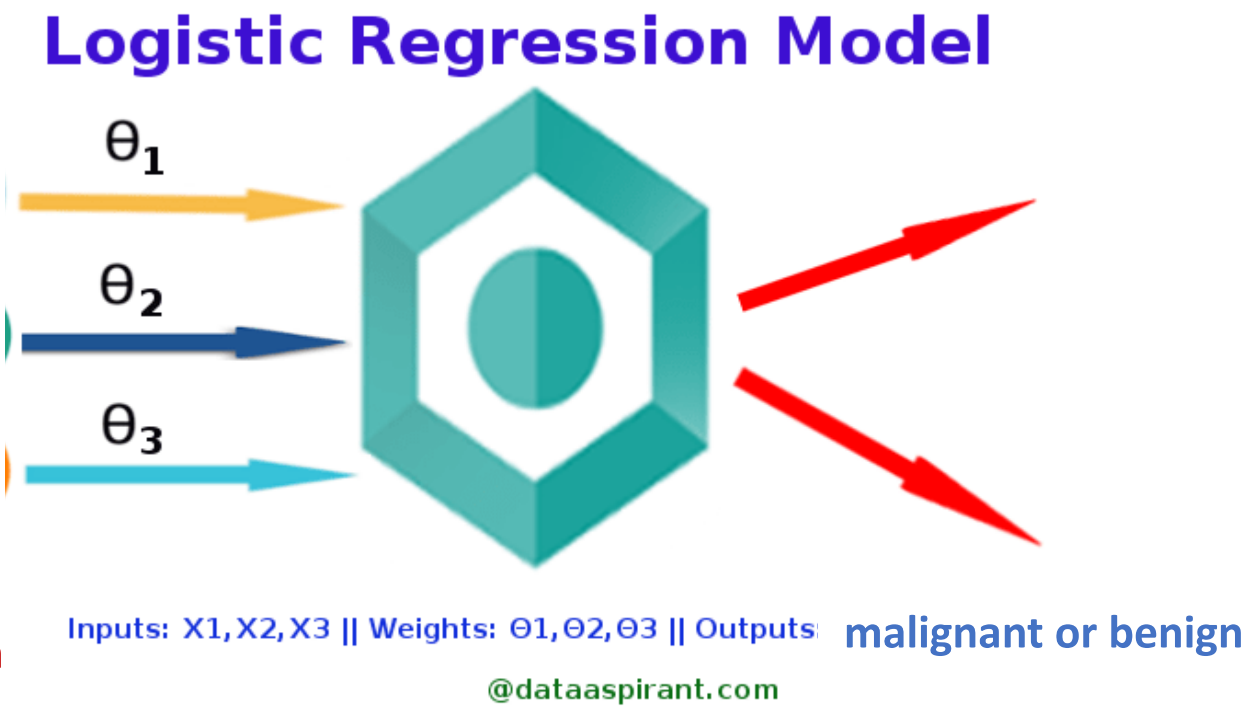


Figure 1: Logistic Regression Model (Source:http://dataaspirant.com/2017/03/02/how-logistic-regression-model-works/)

Logistic Regression is a better way to perform classification within the regression framework.

Logistic Regression

- When the goal is to classify the data points (samples) into categories (or labels), you can use Logistic Regression.
- Output: 0 or 1 , or a probability estimate

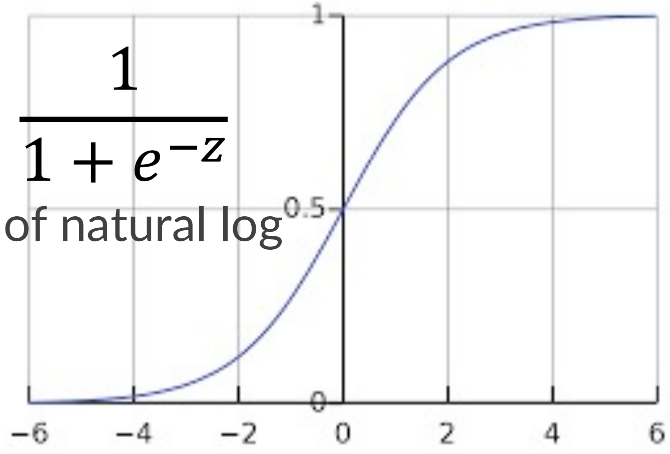
$$z = f(x; w) = w^T x$$

```
def sigmoid(z):
    return 1.0 / (1 + np.exp(-z))
```

Returns a probabilistic estimate *p*

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

e = base of natural log



$$g(x; w) = sigm\ (w^T x) = \frac{1}{1 + e^{-w^T x}}$$

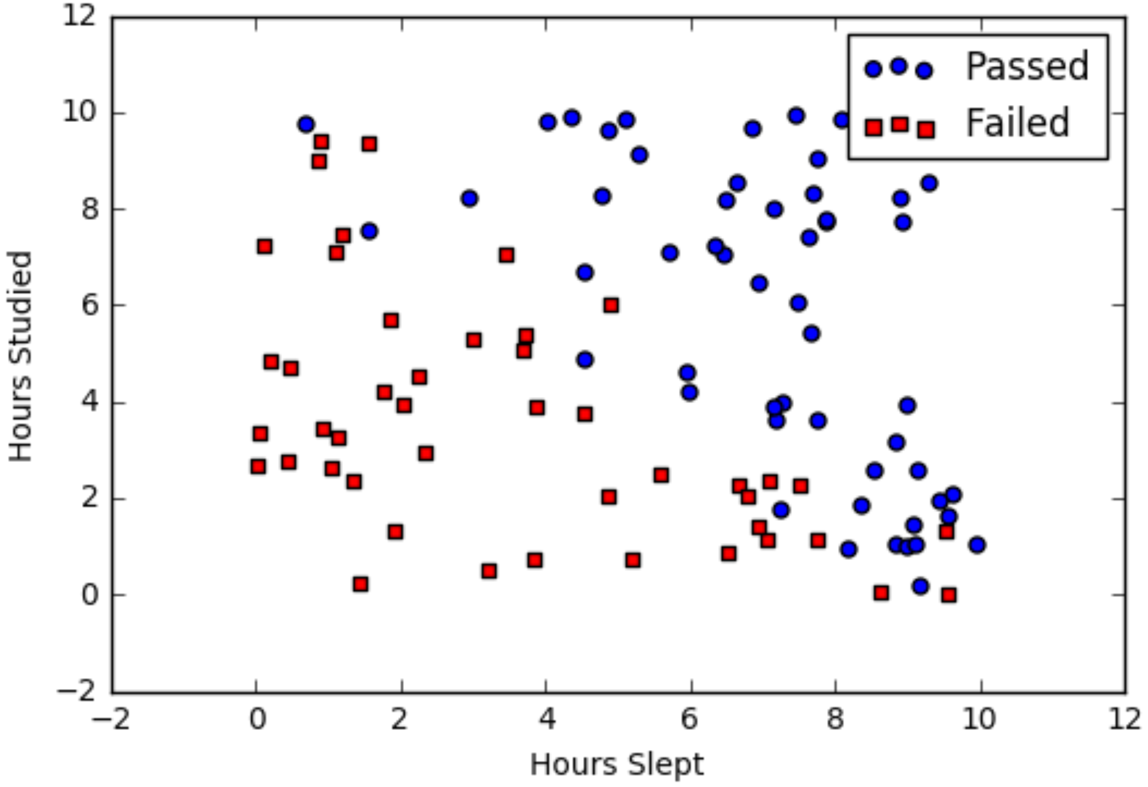
$$y^{(i)} = \begin{cases} 0\ if\ g\left(w^T x^{(i)}\right) < threshold \\ 1\ if\, g\left(w^T x^{(i)}\right) \geq threshold \end{cases}$$

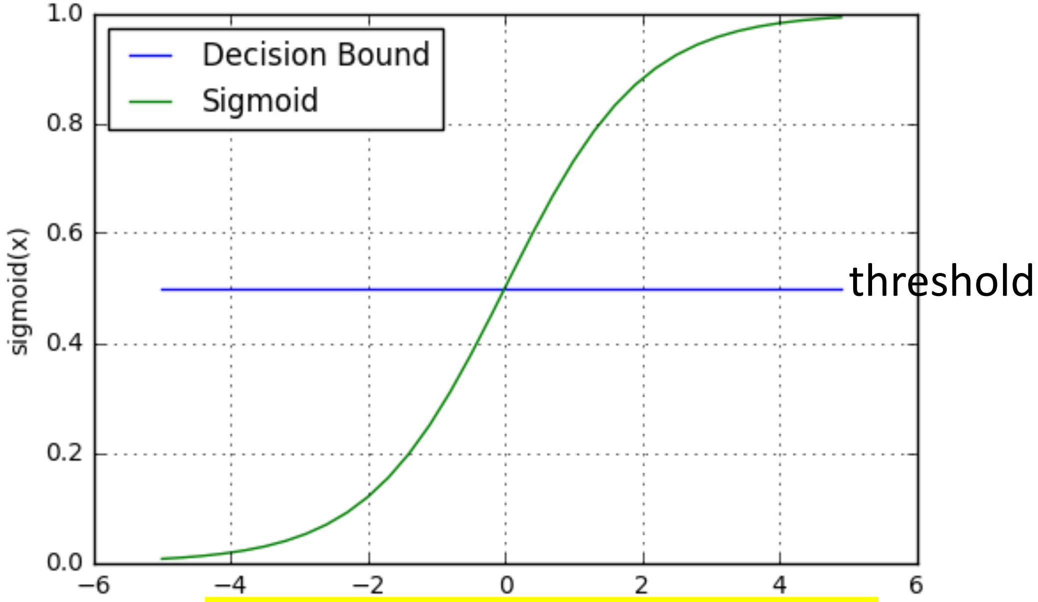
$$\boldsymbol{y^{(i)} = \begin{cases} 0\ if\ p < threshold \\ 1\ if\ p \geq threshold \end{cases}}$$

given the S curve which can capture more complex relationships compared to linear regression and its less sensitivity to outliers

Logistic Regression cont.

Studied	Slept	Passed
4.85	9.63	1
8.62	3.23	0
5.43	8.23	1
9.21	6.34	0





$$y^{(i)} = \begin{cases} 0 \text{ } if \text{ } p < threshold \\ 1 \text{ } if \text{ } p \geq threshold \end{cases}$$

MLE for Logistic Regression

1. Formulate Logistic regression using the sigmoid function
2. Formulate logistic regression as a MLE problem
3. Use Gradient Descent to find the optimal parameters of the model

$$p\left(y^{(i)} = 1 \middle| x^{(i)}; w\right) = g(x^{(i)}; w)$$
$$p\left(y^{(i)} = 0 \middle| x^{(i)}; w\right) = 1 - g(x^{(i)}; w)$$

OLS does not do a good job in non-linear complex problems. Where you don’t have a close form solution. In that case, Gradient Descent is a commonly used to tune the model parameters.

$$p\left(y^{(i)} \middle| x^{(i)}; w\right) = g\left(x^{(i)}; w\right)^{y^{(i)}} \left(1 - g\left(x^{(i)}; w\right)\right)^{1-y^{(i)}}$$

MLE for Logistic Regression cont.

1. Formulate Logistic regression using the sigmoid function
2. Formulate logistic regression as a MLE problem
3. Use Gradient Descent to find the optimal parameters of the model

$$p\left(y^{(i)} \middle| x^{(i)}; w\right) = g\left(x^{(i)}; w\right)^{y^{(i)}} \left(1 - g\left(x^{(i)}; w\right)\right)^{1-y^{(i)}}$$

Assumption: Samples are iid

$$l(w) \triangleq \log p(D|w) = \sum_{i=1}^{N} \log p\left(y^{(i)} \middle| x^{(i)}; w\right)$$

$$= \sum_{i=1}^{N} \log\left(g\left(x^{(i)}; w\right)^{y^{(i)}} \left(1 - g\left(x^{(i)}; w\right)\right)^{1-y^{(i)}} \right)$$

$$= \boldsymbol{\sum_{i=1}^{N} \left(y^{(i)} \log g\left(x^{(i)}; w\right) + \left(1 - y^{(i)}\right) \log\left(1 - g\left(x^{(i)}; w\right)\right)\right)}$$

MLE for Logistic Regression cont.

1. Formulate Logistic regression using the sigmoid function
2. Formulate logistic regression as a MLE problem
3. Use Gradient Descent to find the optimal parameters of the model

$$l(w) = \sum_{i=1}^{N} (y^{(i)} \log g\left(x^{(i)}; w\right) + \left(1 - y^{(i)}\right) \log(1 - g\left(x^{(i)}; w\right))$$

$$\frac{\partial l(w)}{\partial w_j} = \frac{\partial (\sum_{i=1}^{N} (y^{(i)} \log g\left(x^{(i)}; w\right) + \left(1 - y^{(i)}\right) \log\left(1 - g\left(x^{(i)}; w\right)\right))}{\partial w_j}$$

$$= \left(y^{(i)} - g\left(x^{(i)}; w\right)\right) x_j^{(i)}$$

Now take Gradient Ascent!

Gradient Descent (Ascent)

$$\frac{\partial l(w)}{\partial w_j} = \left(y^{(i)} - g\left(x^{(i)}; w\right)\right) x_j^{(i)}$$

$$w_j = w_j + a \frac{\partial l(w)}{\partial w_j}$$

Step size or learning rate

Updated weights

$$w_j = w_j + a\left(y^{(i)} - g\left(x^{(i)}; w\right)\right) x_j^{(i)}$$

Is there another way to tune the parameters ?

Newton's Method

- Another method we could use instead of GD to update the weights

First Derivative

$$w_j = w_j + a \frac{\frac{\partial l(w)}{\partial w_j}}{\frac{\partial^2 l(w)}{(\partial w_j)^2}}$$

Second Derivative

First derivative shows the slope of the tangent line.

Second derivative measures the instantaneous rate of change of the first derivative. The sign of the second derivative tells whether the slope of the tangent line is increasing or decreasing.

Newton’s Method cont.

- Newton’s is also called the **Newton-Raphson** method
- It is a root-finding algorithm that uses the first few terms of the **Taylor series** of a function f(x) about a point x

$$w_{n+1} = w_n - \eta_k \frac{\frac{\partial l(w)}{\partial w_j}}{\frac{\partial^2 l(w)}{(\partial w_j)^2}}$$

$$\frac{\partial l(w)}{\partial w_j} \longrightarrow g_k$$

$$\frac{\partial^2 l(w)}{(\partial w_j)^2} \longrightarrow H_k$$

$$- H_k^{-1} g_k = d_k$$

Step size

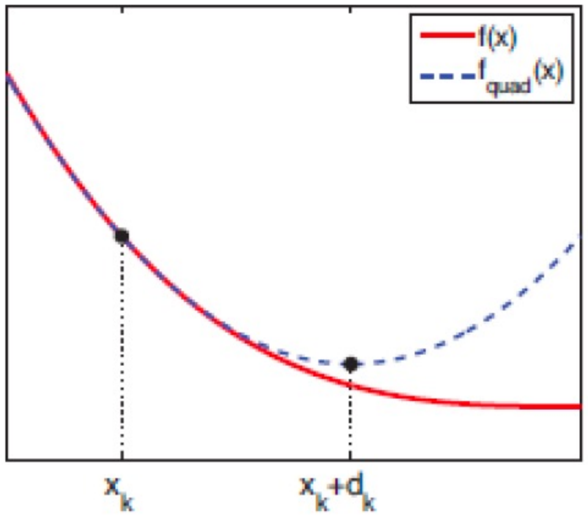
$$w_{n+1} = w_n - \eta_k \frac{g_k}{H_k} = w_n + a\, d_k$$

Algorithm 8.1: Newton's method for minimizing a strictly convex function

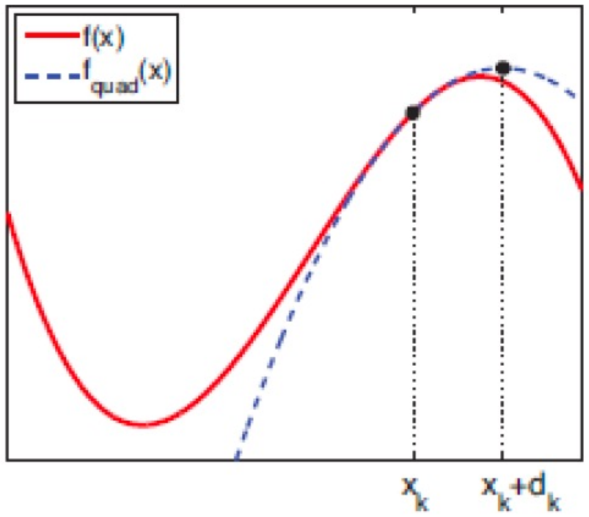
```
Initialize θ_0;
for k = 1, 2, ... until convergence do
    Evaluate g_k = ∇f(θ_k);
    Evaluate H_k = ∇²f(θ_k);
    Solve H_k d_k = −g_k for d_k;
    Use line search to find stepsize η_k along d_k;
    θ_{k+1} = θ_k + η_k d_k;
```

$$w_j := w_j + \boldsymbol{a}\frac{\dfrac{\partial l(w)}{\partial w_j}}{\dfrac{\partial^2 l(w)}{(\partial w_j)^2}}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \mathbf{H}_k^{-1}\mathbf{g}_k$$



(a)



(b)

@IliasTagkopoulos

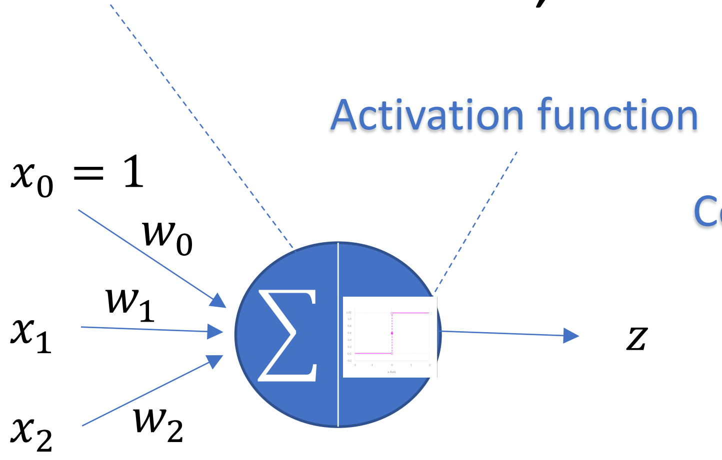
Newton’s Method cont.

- Newton method is good for minimizing a strictly convex function.
- Newton’s method make a quadratic approximation to the original function.
- A good read:
  - To get more information on how Taylor Expansion is used to formulate Newton’s method. Read section “**2 Newton’s Method for Numerical Optimization**” from the following article: http://www.stat.cmu.edu/~cshalizi/350/lectures/26/lecture-26.pdf

Perceptron Learning Algorithm

- The perceptron model takes an input , aggregates it that is calculates the weighted sum, and then returns 1 if the weighted sum is more than a threshold, or else returns 0.

$$w_j = w_j + a\left(y^{(i)} - g\left(x^{(i)}; w\right)\right) x_j^{(i)}$$



Z= output

Combination function: $z = w^T x = w_0 + w_1 x_1 + w_2 x_2$

Threshold can be any scalar value (such as 0)

$$g(z) = \begin{cases} 0 \; if \;\; z < threshold \\ 1 \; if \;\; z \geq threshold \end{cases}$$

- In general, Stochastic Gradient Descent on logistic regression is faster and has a better performance.