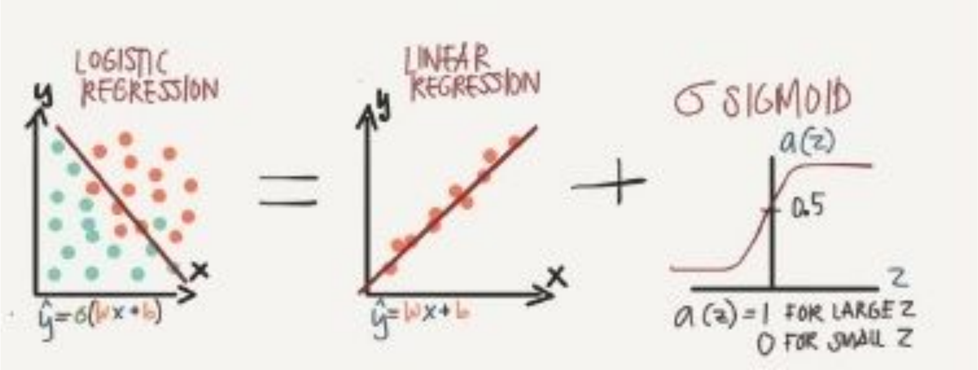
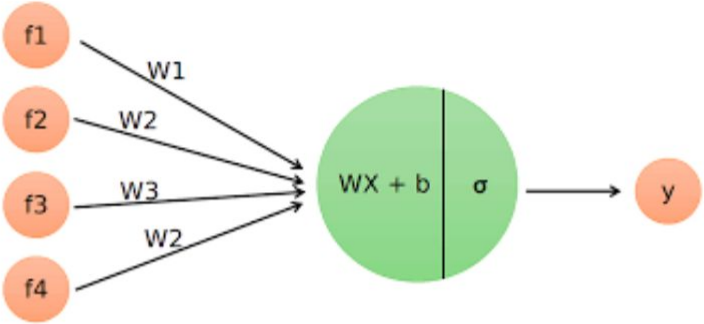
Logistic Regression

Logistic Regression





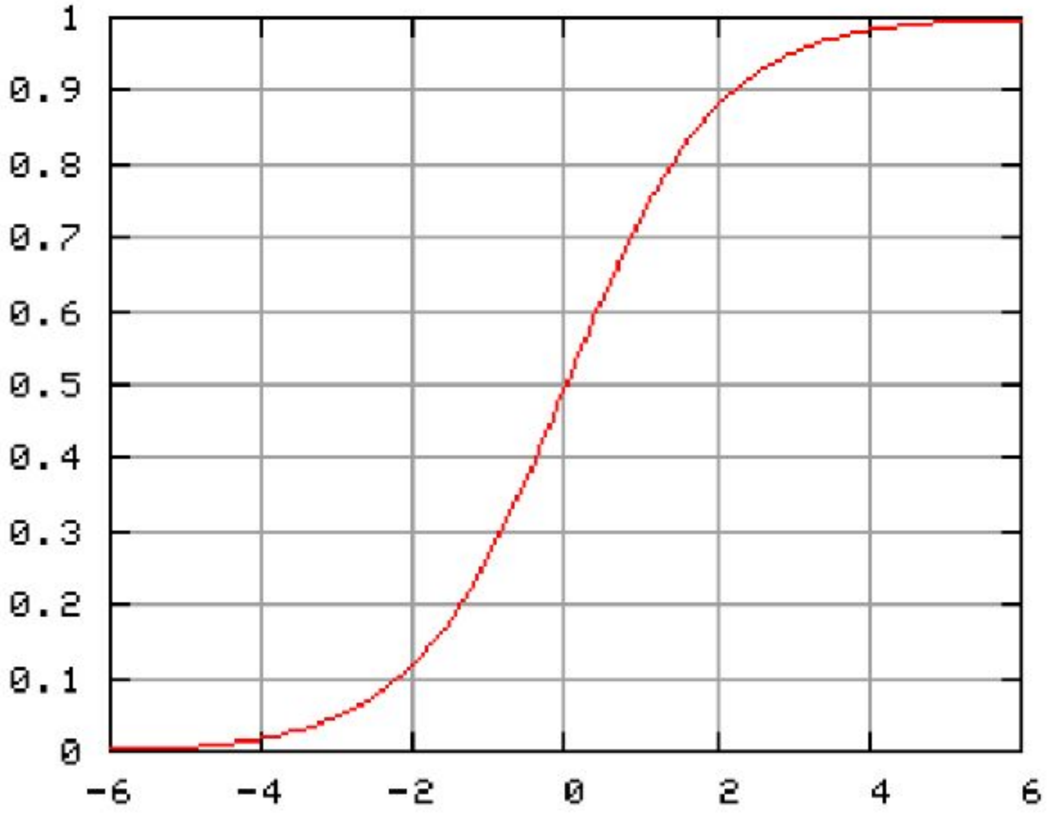
Logistic Regression

Logistic regression is a form of regression analysis in which the outcome variable is binary

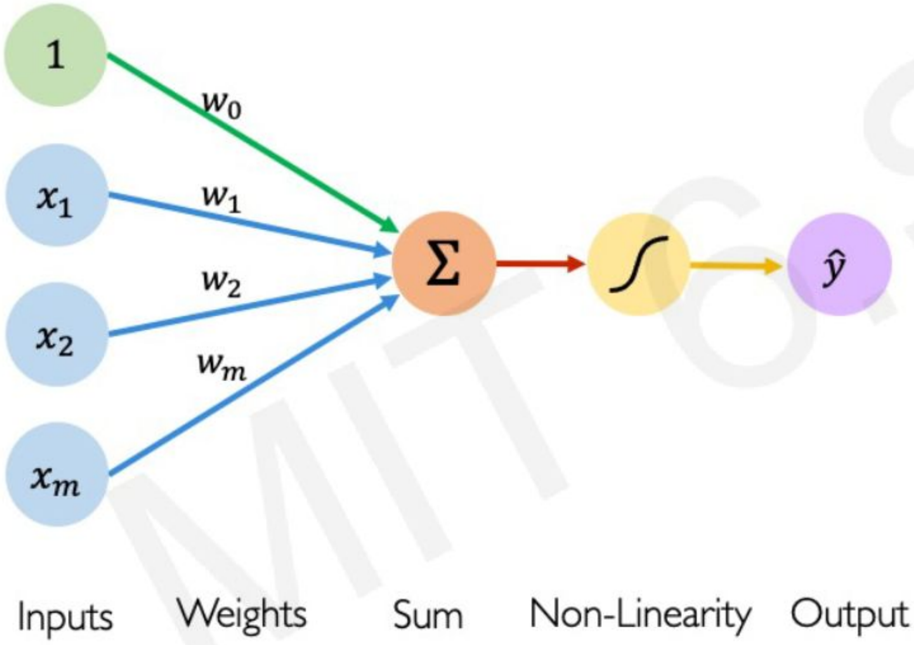
What is the “Logistic” component?

Instead of modeling the outcome, Y, directly, the method models the log odds(Y) using the logistic function.

$$LOGIT(p) = \ln\left(\frac{p}{(1-p)}\right) = z \quad \Leftrightarrow \quad p = \frac{\exp(z)}{1+\exp(z)}$$



The Perceptron: Forward Propagation



$$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i \, w_i \right)$$

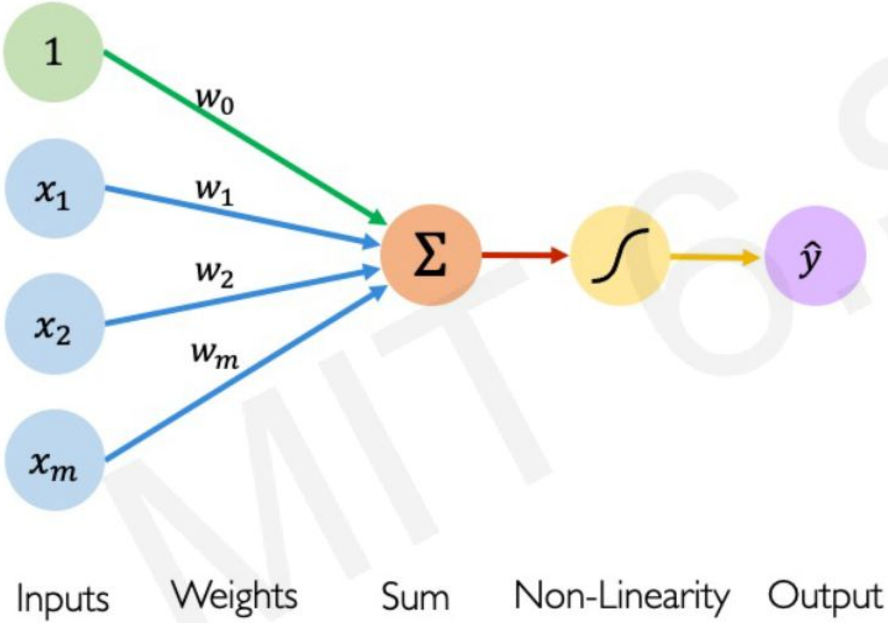
Output

Linear combination of inputs

Non-linear activation function

Bias

The Perceptron: Forward Propagation

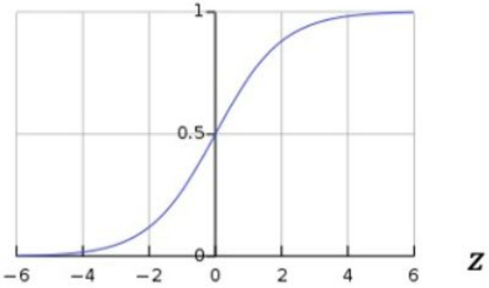


Activation Functions

$$\hat{y} = g\left(w_0 + \boldsymbol{X}^T \boldsymbol{W} \right)$$

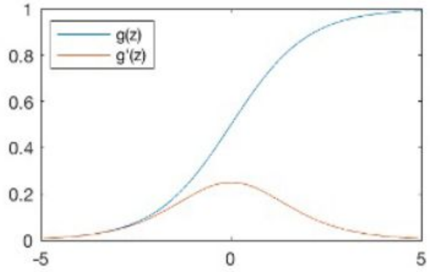
- Example: sigmoid function

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



Common Activation Functions

Sigmoid Function

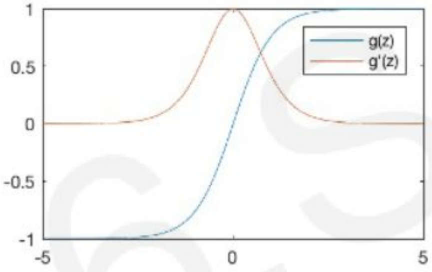


$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

`tf.math.sigmoid(z)`

Hyperbolic Tangent

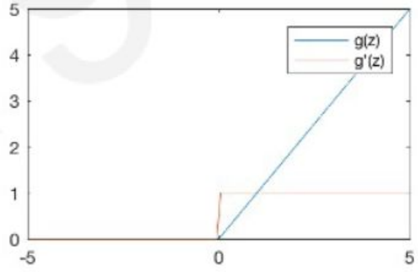


$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

`tf.math.tanh(z)`

Rectified Linear Unit (ReLU)



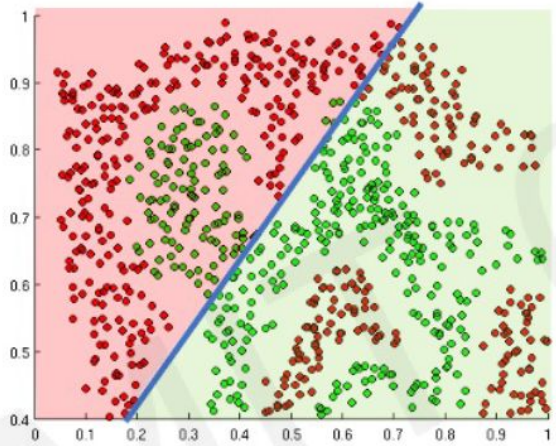
$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

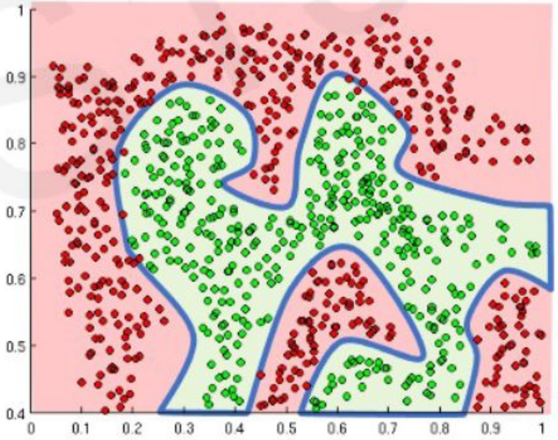
`tf.nn.relu(z)`

Importance of Activation Functions

The purpose of activation functions is to ***introduce non-linearities*** *into the network*



Linear activation functions produce linear decisions no matter the network size



Non-linearities allow us to approximate arbitrarily complex functions

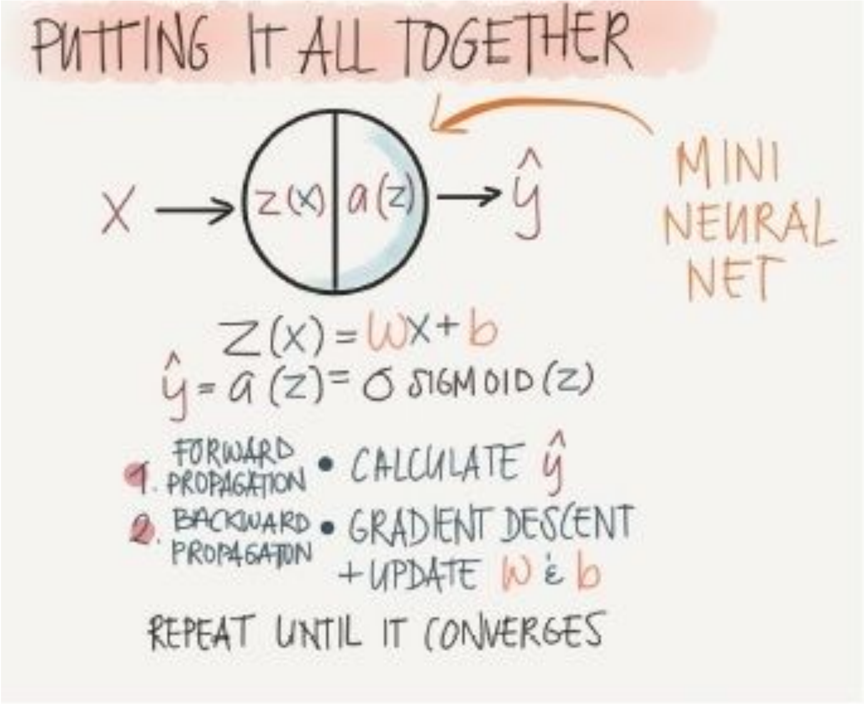
Logistic Regression

$$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b$$

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

$$\begin{aligned} P(y=1) &= \sigma(w \cdot x + b) \\ &= \frac{1}{1+e^{-(w \cdot x+b)}} \end{aligned}$$

$$\begin{aligned} P(y=0) &= 1-\sigma(w \cdot x + b) \\ &= 1-\frac{1}{1+e^{-(w \cdot x+b)}} \\ &= \frac{e^{-(w \cdot x+b)}}{1+e^{-(w \cdot x+b)}} \end{aligned}$$



Logistic Regression

$$z = b + a_1x_1 + a_2x_2 + a_3x_3$$
$$p = 1.0 / (1.0 + e^{-z})$$

Ex:

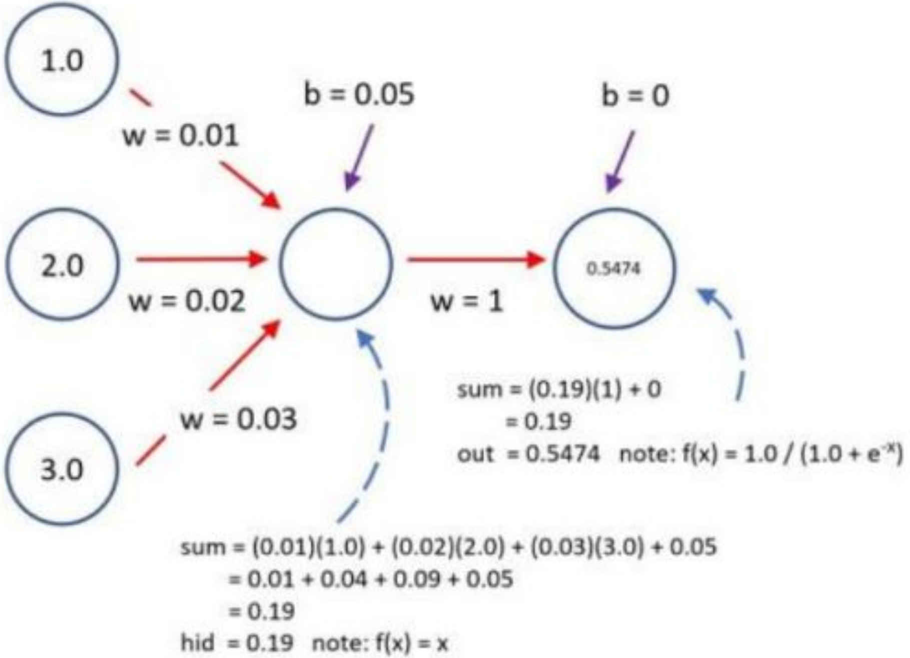
$x_1 = 1.0 \quad a_1 = 0.01$
$x_2 = 2.0 \quad a_2 = 0.02$
$x_3 = 3.0 \quad a_3 = 0.03$
$b = 0.05$

$$\begin{aligned} z &= (0.05) + (0.01)(1.0) + (0.02)(2.0) + (0.03)(3.0) \\ &= 0.05 + 0.01 + 0.04 + 0.09 \\ &= 0.19 \end{aligned}$$

$$\begin{aligned} p &= 1.0 / (1.0 + e^{-0.19}) \\ &= 0.5474 \text{ (predicted class = 1)} \end{aligned}$$

Neural Network

single hidden layer, identity activation $f(x) = x$
single output node, logistic sigmoid activation $f(x) = 1 / (1 + e^{-x})$



Var	Definition	Value in Fig. 5.2
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if “no”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(66) = 4.19$

Let's assume for the moment that we've already learned a real-valued weight for each of these features, and that the 6 weights corresponding to the 6 features are $[2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$, while $b = 0.1$. (We'll discuss in the next section how

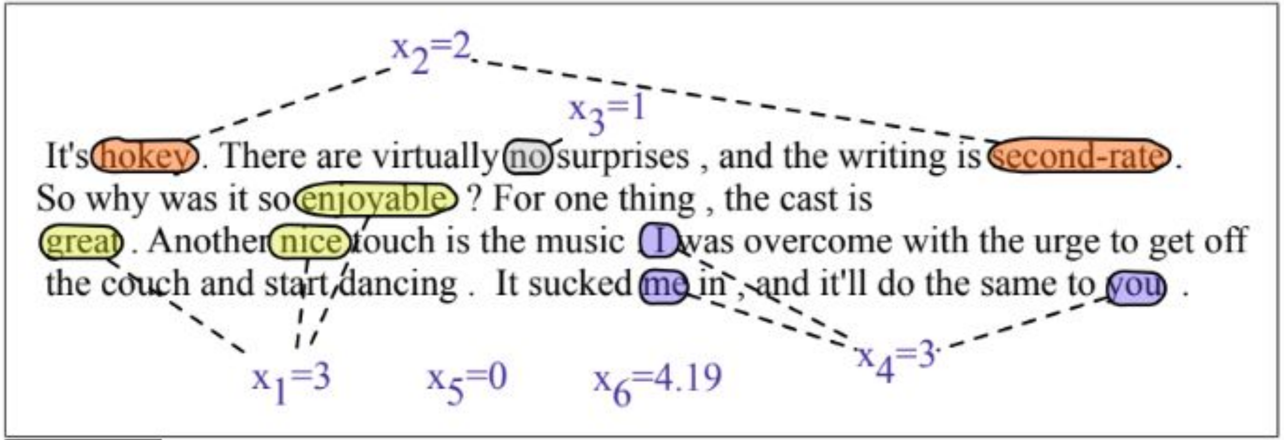


Figure 5.2 A sample mini test document showing the extracted features in the vector x.

Given these 6 features and the input review x, $P(+|x)$ and $P(-|x)$ can be computed using Eq. 5.5:

$$
\begin{aligned}
p(+|x) = P(Y=1|x) &= \sigma(w \cdot x + b) \\
&= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.19] + 0.1) \\
&= \sigma(.833) \\
&= 0.70 \\
p(-|x) = P(Y=0|x) &= 1 - \sigma(w \cdot x + b) \\
&= 0.30
\end{aligned} \tag{5.6}
$$

Algorithm

- Calculate the prediction $\hat{y}$ using the current parameters (W and b)

- Calculate the loss of the current values

- Calculate the gradients of the loss function with respect to the parameters

- Adjust the weights (optimize) using the gradients

- Repeat for the number of epochs, i.e. the number of times to go through the provided examples (dataset)

Learning In Logistic Regression

How are the parameters of the model, the weights w and bias b, learned?

Two components required:

1. Loss function or Cost function: metric for how close the current label ($\hat{y}$) is to the true gold label y.
2. Optimization algorithm: for iteratively updating the weights

Loss Function or Cost Function

For an observation x, how close the classifier output ($\hat{y}=\sigma(w \cdot x+b)$) is to the correct output (y, which is 0 or 1)

$L(\hat{y}, y)$ = How much $\hat{y}$ differs from the true y

Loss Function or Cost Function

Conditional maximum likelihood estimation prefers the correct class labels of the training examples to be more likely.

We choose the parameters w, b that maximize the log probability of the true y labels in the training data given the observations x.

Maximum Likelihood Estimation

$$P(y=1) = \sigma(w \cdot x + b) = \frac{1}{1+e^{-(w \cdot x + b)}}$$

For samples labeled as ‘1’, we try to estimate (w and b) such that the product of all probability p(x) is as close to 1 as possible.

And for samples labeled as ‘0’, we try to estimate (w and b) such that the product of all probability is as close to 0 as possible in other words (1 – p(x)) should be as close to 1 as possible.

$$for\ samples\ labelled\ as\ 1: \prod_{s\ in\ y_i=1} p(x_i)$$

$$for\ samples\ labelled\ as\ 0: \prod_{s\ in\ y_i=0} (1-p(x_i))$$

Maximum Likelihood Estimation

On combining both , (w and b) parameters should be such that the product of both of these products is maximum over all elements of the dataset.

$$L(\beta) = \prod_{s \ in \ y_i = 1} p(x_i) \ * \prod_{s \ in \ y_i = 0} (1 - p(x_i))$$

This function is the one we need to optimize and is called the likelihood function.

Loss Function

If $y = 1$: $p(y|x) = \hat{y}$

If $y = 0$: $p(y|x) = 1 - \hat{y}$

$$P(y|x) = \hat{y}^{y} (1-\hat{y})^{(1-y)}$$

If y = 1 => $\hat{y}$

If y = 0 => 1-$\hat{y}$

Log Likelihood

Since there are only two discrete outcomes (1 or 0), this is a Bernoulli distribution.

p(y|x) for one observation:

$$p(y|x) = \hat{y}^{y}(1-\hat{y})^{1-y}$$

$$\begin{aligned}\log p(y|x) &= \log\left[\hat{y}^{y}(1-\hat{y})^{1-y}\right]\\ &= y\log\hat{y} + (1-y)\log(1-\hat{y})\end{aligned}$$

whatever values maximize a probability will also maximize the

log of the probability.

Log Likelihood

Since there are only two discrete outcomes (1 or 0), this is a Bernoulli distribution.

p(y|x) for n observation:

$$L(\beta) = \prod_{s} \left(p(x_i)^{y_i} \;\; * \;\; (1 - p(x_i))^{1-y_i} \right)$$

$$l(\beta) = \sum_{i=1}^{n} y_i \, log(p(x_i)) + (1 - y_i) \, log(1 - p(x_i))$$

$$where,\ l(\beta)\ is\ log - likelihood$$

Cross Entropy

Log likelihood should be maximized.

In order to turn this into loss function (something that we need to minimize), we'll just flip the sign.

negative log likelihood loss = cross-entropy loss.

$$L_{CE}(\hat{y}, y) = -\log p(y|x) \;=\; -[y\log\hat{y} + (1-y)\log(1-\hat{y})]$$

$$L_{CE}(w, b) \;=\; -[y\log\sigma(w\cdot x+b) + (1-y)\log(1-\sigma(w\cdot x+b))]$$

Binary Cross Entropy

negative log likelihood loss = cross-entropy loss.

Since we are working on 2 classes it becomes

Binary Cross Entropy Loss

$$L_{CE}(\hat{y}, y) = -\log p(y|x) \;=\; -[y \log \hat{y} + (1-y)\log(1-\hat{y})]$$

$$L_{CE}(w, b) \;=\; -[y \log \sigma(w \cdot x + b) + (1-y)\log(1-\sigma(w \cdot x + b))]$$

Binary Cross Entropy Loss

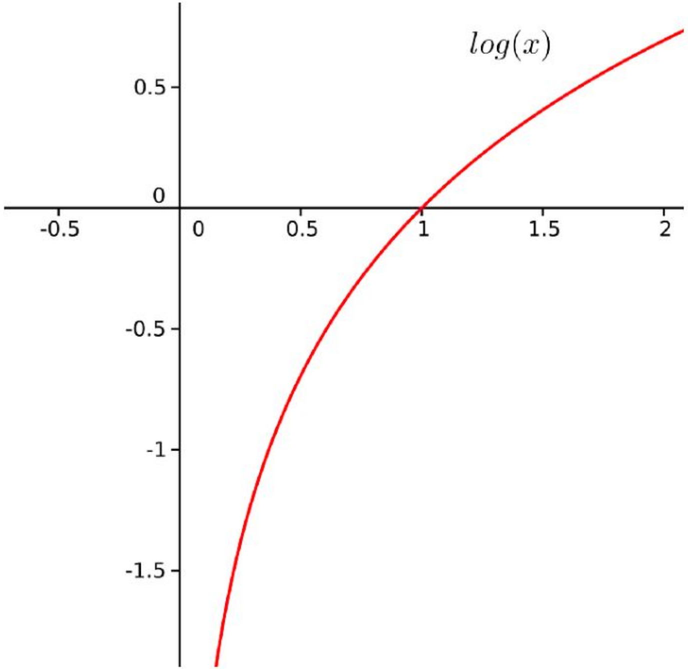
$$L_{CE}(\hat{y}, y) = -\log p(y|x) \;=\; -[y\log\hat{y} + (1-y)\log(1-\hat{y})]$$

Binary cross-entropy is often calculated as the average cross-entropy across all data examples.

$$L = -\frac{1}{N}\left[\sum_{j=1}^{N} [y\log\hat{y} + (1-y)\log(1-\hat{y})] \right]$$

Why Not MSE?

log(p(x))<0 for all p(x) in (0,1).



Actual label: “1” Predicted: “0”

MSE: $(1 - 0)^2$ = 1

Log loss: -(1 * log(0) + 0 * log(1)) = infinity!!

Actual label: “1” Predicted: “1”

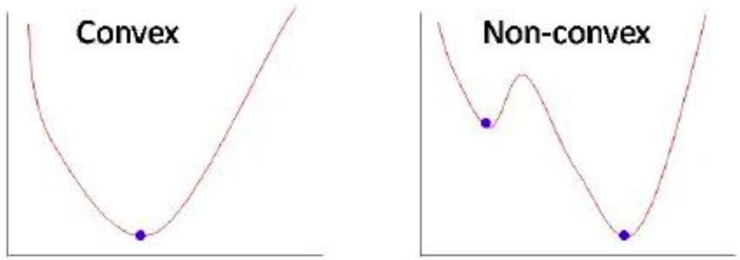
MSE: $(1 - 1)^2$ = 0

Log loss: -(1 * log(1) + 0 * log(0)) = 0

Why Not MSE?

Mean squared error is a non-convex function.

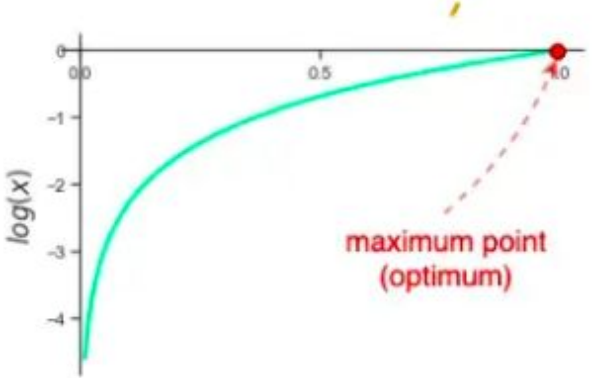
Cross entropy is a convex function.

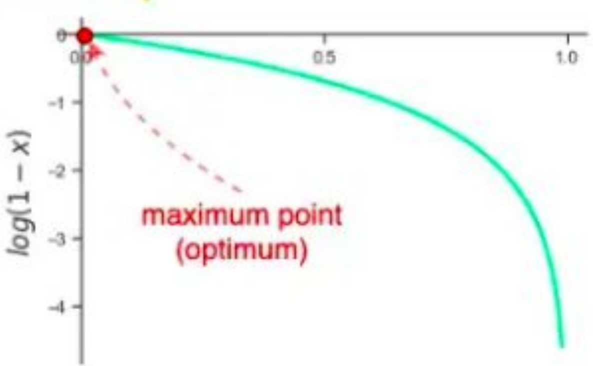


Gradient descent will converge into global minimum only if the function is convex.

Binary Cross Entropy Loss

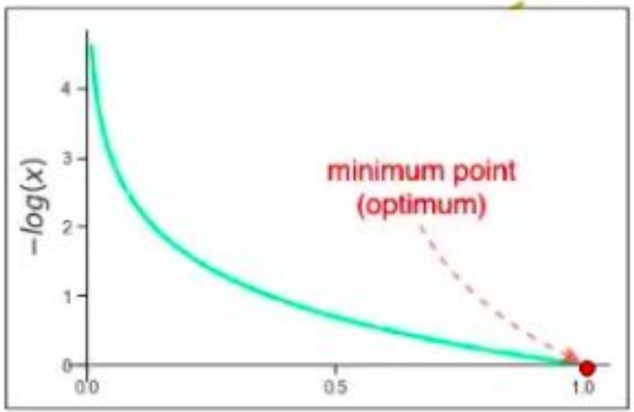
$$L_{CE}(\hat{y}, y) = -\log p(y|x) \; = \; -[y \log \hat{y} + (1-y)\log(1-\hat{y})]$$

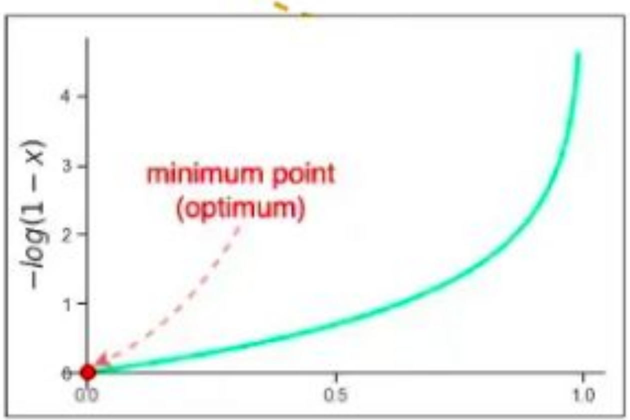




Binary Cross Entropy Loss

$$L_{CE}(\hat{y}, y) = -\log p(y|x) \;=\; -[y \log \hat{y} + (1-y)\log(1-\hat{y})]$$



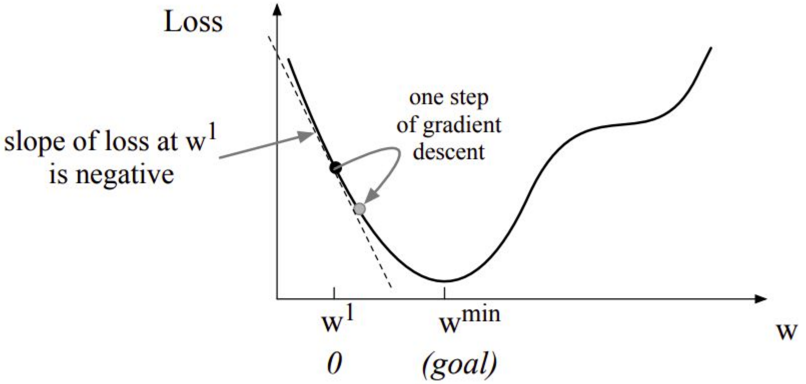


Gradient Descent Algorithm

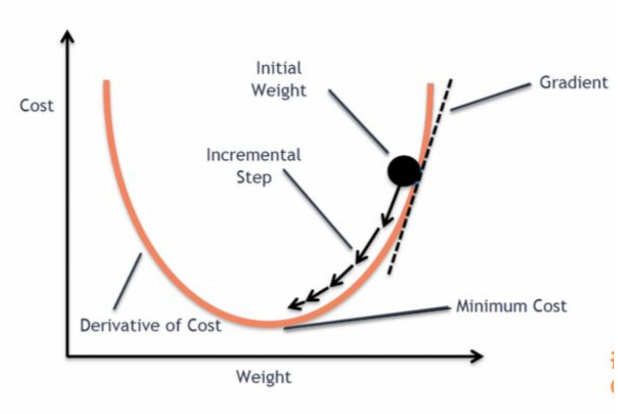
Goal:

- To find the optimal weights (in the case of logistic regression θ = w,b)
- Minimize the loss function

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} L_{CE}(y^{(i)}, x^{(i)}; \theta)$$



Gradient Descent Algorithm



Gradient Descent steps are:

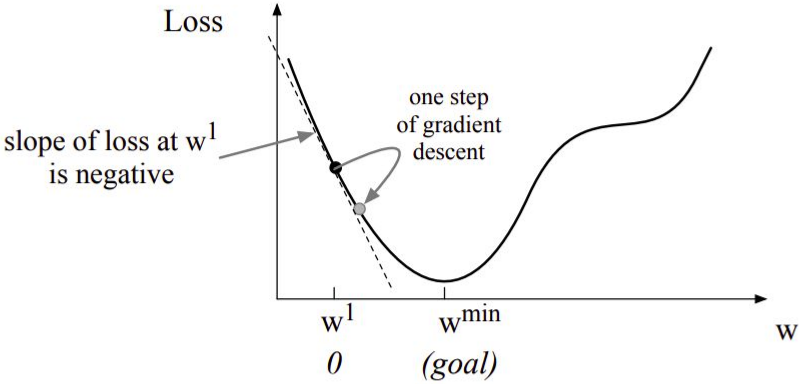
1. choose a starting point (initialisation)
2. calculate gradient at this point
3. make a scaled step in the opposite direction to the gradient (objective: minimise)
4. repeat points 2 and 3 until one of the criteria is met:
   - maximum number of iterations reached
   - step size is smaller than the tolerance (due to scaling or a small gradient).

Gradient Descent Algorithm

Goal:

- To find the optimal weights (in the case of logistic regression θ = w,b)
- Minimize the loss function

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} L_{CE}(y^{(i)}, x^{(i)}; \theta)$$



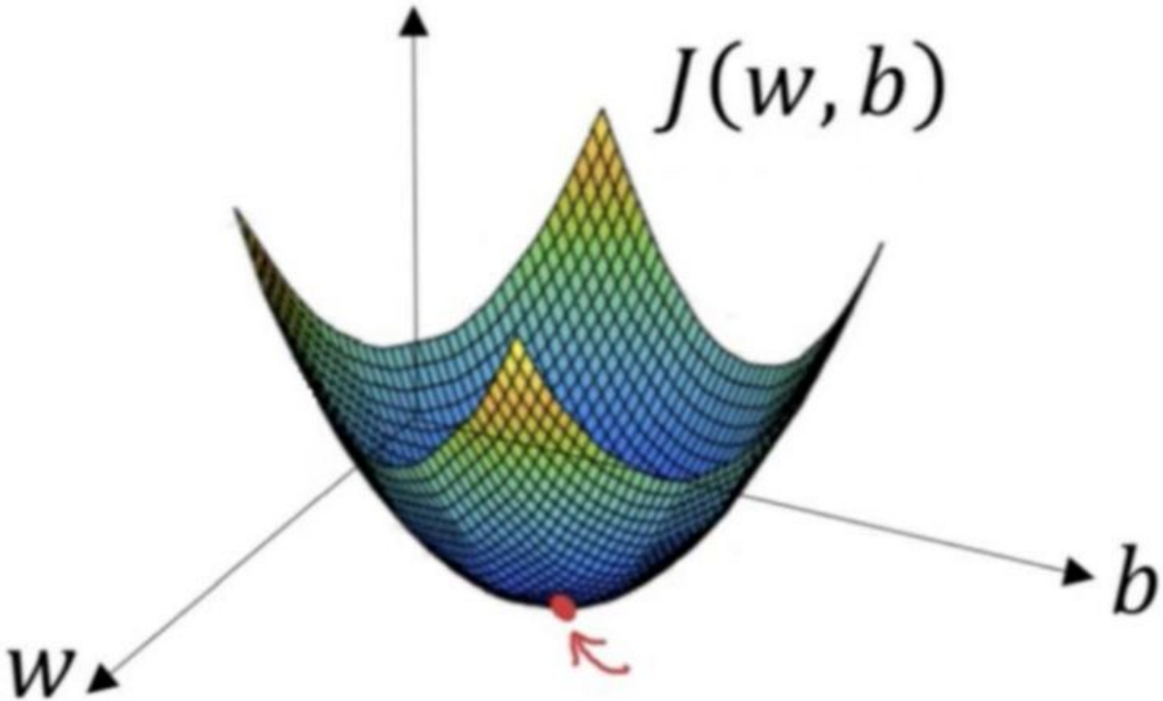
Gradient Descent

$$J(w,b) = \frac{\sum_1^m L(\hat{y}^i, y^i)}{m} = \frac{\sum_1^m y^i log\hat{y}^i + (1-y^i)log(1-\hat{y}^i)}{m}$$

In order to minimize the cost function for minimal error across the training data set to find w and b

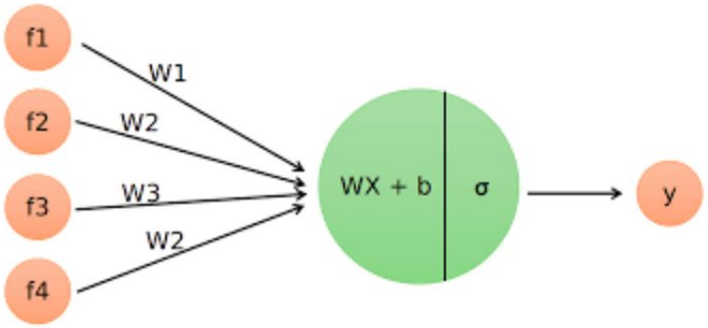
The value of the parameters can be achieved using gradient descent technique.

Gradient Descent



Gradient– Backward Propagation

$$L_{CE}(w,b) = -[y\log\sigma(w\cdot x+b)+(1-y)\log(1-\sigma(w\cdot x+b))]$$



Gradient

$$\boxed{z = w_1x_1 + w_2x_2 + b} \longrightarrow \boxed{\hat{y} = a = \sigma(z)} \longrightarrow \boxed{L(\hat{y}, y)}$$

$$\Leftrightarrow \boxed{a = \hat{y}}$$

$$\boxed{w_1} \Rightarrow \frac{\partial(L)}{\partial w_1} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial(z)}{\partial w_1}$$

$$\frac{\partial L}{\partial a} = \frac{\partial}{\partial a}\left(-y \log a - (1-y)\log(1-a)\right)$$

$$= -y\left(\frac{1}{a}\right) - (-1)\frac{(1-y)}{(1-a)}$$

$$\boxed{\frac{\partial L}{\partial a} = \left(\frac{-y}{a}\right) + \left(\frac{1-y}{1-a}\right)}$$

$$\boxed{\frac{\partial a}{\partial z} = a(1-a)}$$

$$\boxed{\frac{\partial z}{\partial w_1} = x_1}$$

$$\frac{\partial L}{\partial w_1} = \left(\left(\frac{-y}{a} + \frac{(1-y)}{1-a}\right) \cdot (a)(1-a)\right) \cdot x_1$$

$$= (a-y) \cdot x_1$$

Update for w_1,

$$\frac{\partial L}{\partial w_1} = (a-y) \cdot x_1$$

Here, $(a-y) = \frac{\partial L}{\partial z}$

$$\boxed{w_1 = w_1 - \alpha \frac{\partial L}{\partial w_1}}$$

Similarly, for all parameters

$$\boxed{w_i = w_i - \alpha \frac{\partial L}{\partial w_i}}$$

$i = 1, 2, \dots, m$

m = no. of parameters

$$\boxed{b = b - \alpha \frac{\partial L}{\partial b}}$$

Where, $\frac{\partial L}{\partial b} = (a-y)$

2. Consider a classification problem where we are building a logistic regression classifier. The task is to predict if a given city has a risk of a disease epidemic or not. The data is defined using two input

Practice Problem 6

a. Construct a logistic regression model with two predictors for the riding mower example with $\beta_0 = -25.9382, \beta_1 = 0.1109, \beta_2 = 0.9638$, where β_1 and β_2 are for the "Income" and "Lot_Size" variables, respectively.

b. Using the logistic regression model with probability cutoff = 0.75, classify the following 6 customers as "Owner" or "Nonowner" : if $p \geq 0.75$ then the case as a "Owner". Present the results in a classification matrix.

Customer#	Income	Lot_Size	Ownership
1	60.0	18.4	Owner
2	64.8	21.6	Owner
3	84.0	17.6	Nonowner
4	59.4	16.0	Nonowner
5	108.0	17.6	Owner
6	75	19.6	Nonowner

only

References

- https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc
- https://arunaddagatla.medium.com/maximum-likelihood-estimation-in-logistic-regression-f86ff1627b67
- https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e
- https://www.marktechpost.com/2021/04/08/logistic-regression-with-keras/
- https://www.datasciencecentral.com/logistic-regression-as-a-neural-network/
- https://github.com/SSaishruthi/LogisticRegression_Vectorized_Implementation/blob/master/Logistic_Regression.ipynb
- https://towardsdatascience.com/where-did-the-binary-cross-entropy-loss-function-come-from-ac3de349a715
- https://www.youtube.com/watch?v=nzNp05AyBM8
-