Online Student Training for "Artificial Intelligence & Machine Learning"

(4th Feb, 2021 – 17th Mar, 2021)

LOGISTIC REGRESSION

Another probabilistic approach for classification

Faculty Trainer

Naw Varsha Pipada

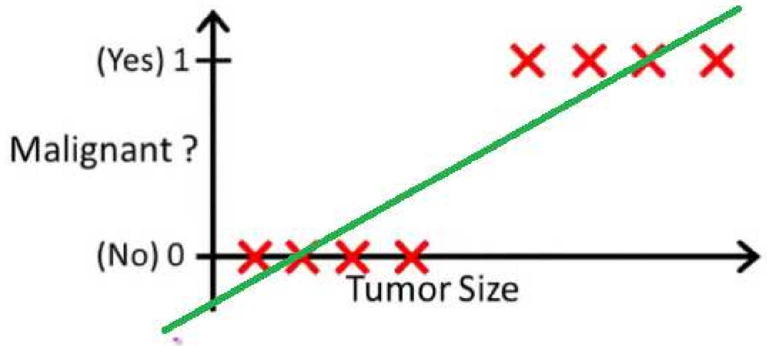
Department of Computer Science & Engineering

Engineering College Bikaner

CONTENTS

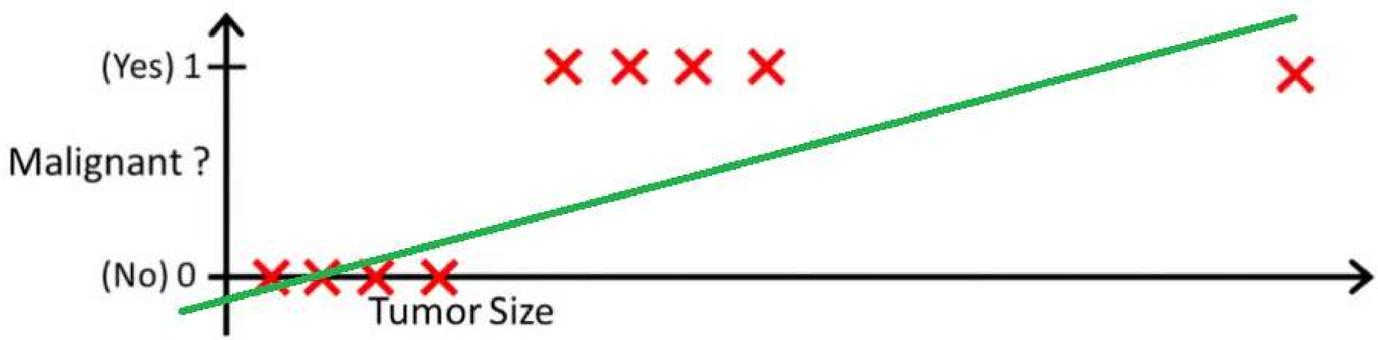
LINEAR REGRESSION FOR CLASSIFICATION?

- The example below we're fitting a straight line through *{tumor size, tumor type}* sample set
- Above, malignant tumors get 1 and non-malignant ones get 0, and the green line is our hypothesis h(x) or regression line.
- To make predictions we may say that for any given tumor size x, if h(x) gets bigger than 0.5 we predict malignant tumor, otherwise we predict benign.



LINEAR REGRESSION FOR CLASSIFICATION?

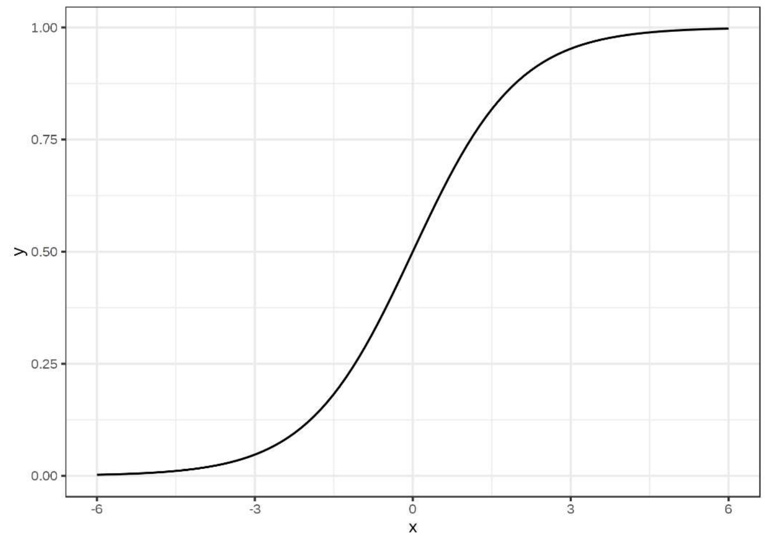
- After adding another sample with a huge tumor size and running linear regression again, h(x) > 0.5 → malignant doesn't work anymore. To keep making correct predictions we need to change it to h(x)>0.2 or something - but that not how the algorithm should work.
- We cannot change the hypothesis each time a new sample arrives



LOGISTIC REGRESSION

- A solution for classification is logistic regression.
- Instead of fitting a straight line, the logistic regression model uses the logistic function to squeeze the output of a linear equation between 0 and 1.
- The logistic function is defined as:

$$\text{logistic}(\eta) = \frac{1}{1 + exp(-\eta)}$$



LINEAR TO LOGISTIC REGRESSION

- The step from linear regression to logistic regression is kind of straightforward. In the linear regression model, we have modelled the relationship between outcome and features with a linear equation:

$$\hat{y}^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_p x_p^{(i)}$$

- For classification, we prefer probabilities between 0 and 1, so we wrap the right side of the equation into the logistic function. This forces the output to assume only values between 0 and 1.

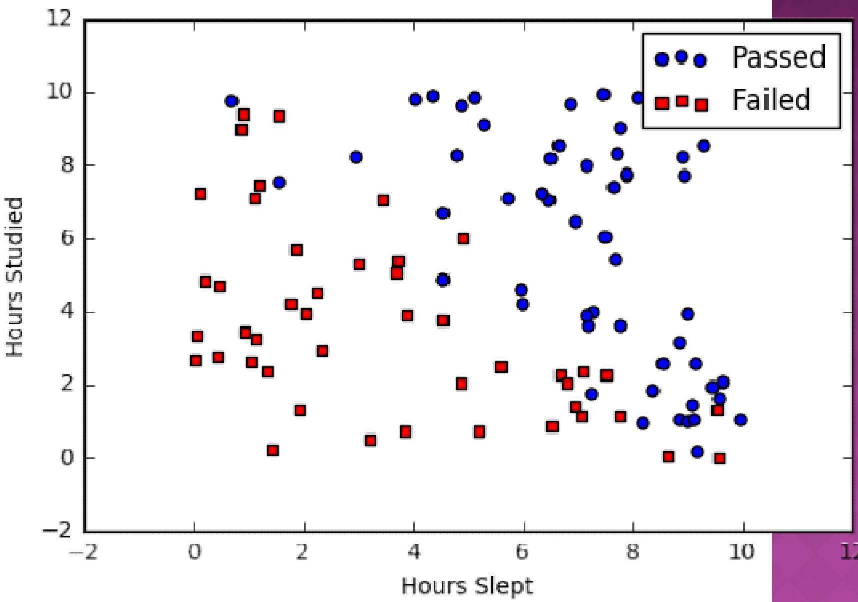
$$P(y^{(i)} = 1) = \frac{1}{1 + exp(-(\beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_p x_p^{(i)}))}$$

TYPES OF LOGISTIC REGRESSION

- Binary
- Multiclass
- Ordinal

EXAMPLE

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



EXAMPLE

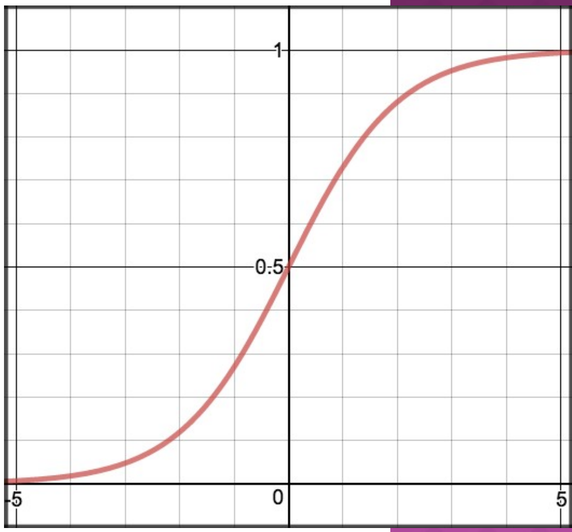
- In order to map predicted values to probabilities, we use the sigmoid function.
- The function maps any real value into another value between 0 and 1.
- In machine learning, we use sigmoid to map predictions to probabilities.

DECISION BOUNDARY

- Our current prediction function returns a probability score between 0 and 1.
- In order to map this to a discrete class, we select a threshold value or tipping point above which we will classify values into class 1 and below which we classify values into class 2.

$p \geq 0.5, class=1$

$p < 0.5, class=0$



PREDICTION FUNCTION

- A prediction function in logistic regression returns the probability of our observation being positive, True, or "Yes".
- We call this class 1 and its notation is P(class=1)

$$h_\theta(x) = g(\theta^T x)$$

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

- For example, $\theta^T x$ = $\theta x^i := \theta_0 + \theta_1 x_1^i + \cdots + \theta_p x_p^i.$
- This is the equation of multiple linear regression

COST FUNCTION

- We can't (or at least shouldn't) use the same cost function Mean Squared Error(MSE) as we did for linear regression because squaring this prediction (logistic function) as we do in MSE results in a non-convex function with many local minimums.
- If our cost function has many local minimums, gradient descent may not find the optimal global minimum
- So, here the cost function used is

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\mathrm{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) \qquad \text{if } y = 1$$

$$\mathrm{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \qquad \text{if } y = 0$$

COST FUNCTION

- The cost function is compressed to form a single function:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^i \log(h_\theta(x^i)) + (1-y^i)\log(1-h_\theta(x^i))$$

- Multiplying by y and (1−y) in the above equation is a sneaky trick that let’s us use the same equation to solve for both y=1 and y=0 cases.

MINIMIZE COST FUNCTION : GRADIENT DESCENT

- Finding log values before derivative of cost function,

$$\log h_\theta(x^i) = \log \frac{1}{1+e^{-\theta x^i}} = -\log(1+e^{-\theta x^i}),$$

$$\log(1-h_\theta(x^i)) = \log(1-\frac{1}{1+e^{-\theta x^i}}) = \log(e^{-\theta x^i}) - \log(1+e^{-\theta x^i}) = -\theta x^i - \log(1+e^{-\theta x^i}),$$

- Putting in cost function,

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^i \log(h_\theta(x^i)) + (1-y^i)\log(1-h_\theta(x^i))$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} \left[-y^i(\log(1+e^{-\theta x^i})) + (1-y^i)(-\theta x^i - \log(1+e^{-\theta x^i}))\right]$$

MINIMIZE COST FUNCTION : GRADIENT DESCENT

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i\theta x^i - \theta x^i - \log(1+e^{-\theta x^i})\right] = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i\theta x^i - \log(1+e^{\theta x^i})\right], \quad (*)$$

$$-\theta x^i - \log(1+e^{-\theta x^i}) = -\left[\log e^{\theta x^i} + \log(1+e^{-\theta x^i})\right] = -\log(1+e^{\theta x^i})$$

$$\frac{\partial}{\partial\theta_j} y_i\theta x^i = y_i x_j^i,$$

$$\frac{\partial}{\partial\theta_j}\log(1+e^{\theta x^i}) = \frac{x_j^i e^{\theta x^i}}{1+e^{\theta x^i}} = x_j^i h_\theta(x^i),$$

MINIMIZE COST FUNCTION : GRADIENT DESCENT

- So the value of cost function derivative is,

$$\frac{\partial}{\partial \theta_j} J(\theta) = \sum_{i=1}^{m} (h_\theta(x^i) - y^i) x_j^i$$

Remember that the general form of gradient descent is:

$$Repeat\ \{$$
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$
$$\}$$

PROS

- Logistic Regression is one of the simplest machine learning algorithms and is easy to implement yet provides great training efficiency in some cases. Also due to these reasons, training a model with this algorithm doesn't require high computation power.
- It makes no assumptions about distributions of classes in feature space.
- This algorithm allows models to be updated easily to reflect new data, unlike decision trees or support vector machines. The update can be done using stochastic gradient descent.

CONS

- On high dimensional datasets, this may lead to the model being over-fit on the training set
- Non linear problems can't be solved with logistic regression since it has a linear decision surface.
- It is difficult to capture complex relationships using logistic regression

THANK YOU!!!!