Classification Problem

A classification problem is a problem where the goal is to predict whether the target variable belongs to one of the pre-defined possibility. When there is only two choices, it is referred to as a binary classification problem. Otherwise, it is a multi-class classification problem

Example:

- Wining or Losing a game
- Determine whether a email is a spam or not
- Decide whether it will rain or not tomorrow

Although the final forecast target variable is either 1 or 0 (Win or Lose, Spam or not Spam, Rain or not rain), we often forecast the probability of the interested event first. If the probability is larger than 0.5, we classify the instance as “1”, otherwise as “0”

Odds vs Probability

Classification Problem falls under the group of "Supervised machine learning" because we need training example with the target variable known.

First, we define what is an Odds. Let P be the probability of the event we are interested in (say winning a game), then (1-P) will be the probability of losing

$$\text{Odds} = P / (1-P)$$

Example, if probability of winning is 2/3, then the Odds is 2 to 1. If probability of winning Is 3/5, the Odds is 3 to 2 (i.e 1.5)

Furthermore, instead of Odds, we will now focus on the <u>Log of the Odds</u>, i.e.

$$Y = \text{Log}\ (\ P / (1-P)\ )$$

Logistic Regression

Logistic Regression assumes that the Log of Odds is a linear function of the features, i.e.

Y = Log (P/(1-P)) = theta_0 * X_1 + theta_1 * X_2 + … + theta_n * X_n

From Y = Log (P/(1-P)), we can derive

P = 1 / (1 + exp (-Y)) = Sigmoid (y)

The function $\sigma(t) = \frac{1}{1 + e^{-t}}$ is called a Sigmoid function

Sigmoid or Logistic Function

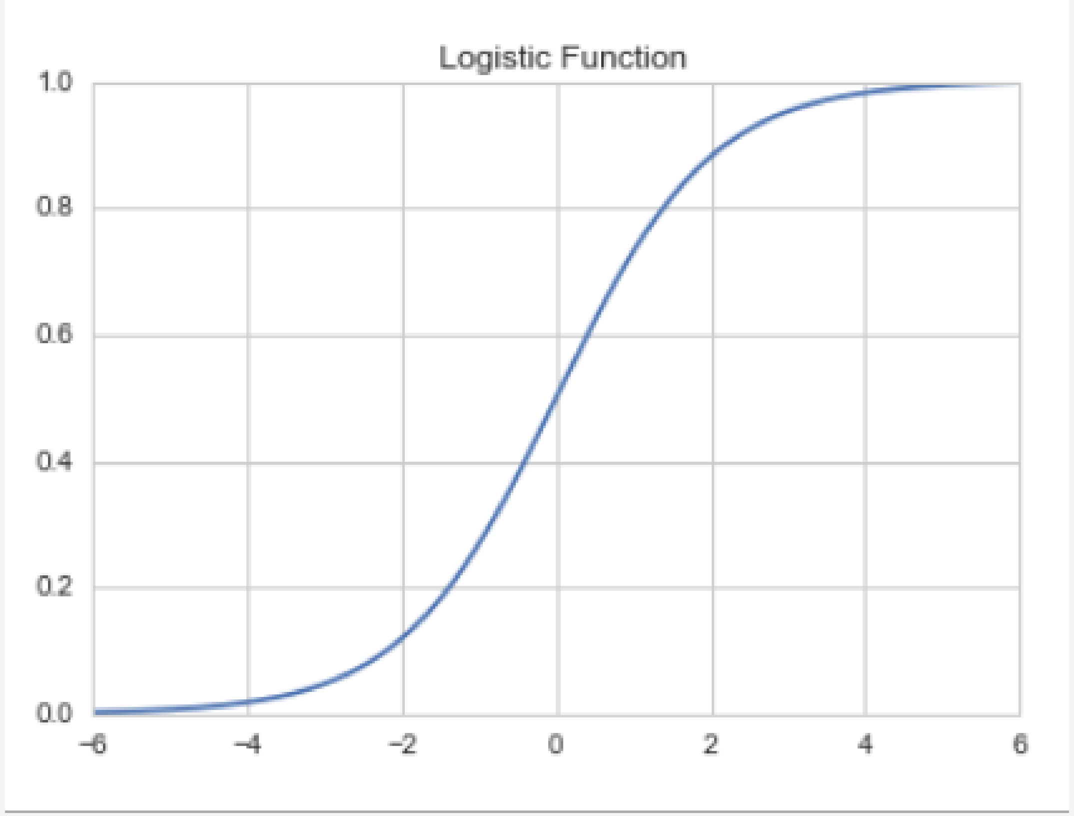
P = 1 / (1 + exp(-y))

Y can be from negative infinity to positive infinity, while P is limited from 0 to 1

Y = m X + b (for one-variable feature)
= theta_1 X + theta_0 (using theta for coeff)

Using the Sigmoid function, we can calculate from the features vector to a probability which can then be used to map to a two-class target variable (i.e. Class Label = 1 when P > 0.5, Class Label = 0 when P < 0.5)

P



Y

Sigmoid Function

Now consider an example where we want to decide whether a student pass or fail based on how many hours he studies before the test

Class Label = 1 or 0 = Pass or Fail

Predictor = Number of hours studied

One can solve this as a Logistic Regression with one variable

Log (P/(1-P)) = Y = m X + b

m and b are the regression coefficients
X is the number of hours studies
Y is the Log (Odds of Passing)

https://towardsdatascience.com/logistic-regression-python-7c451928efee

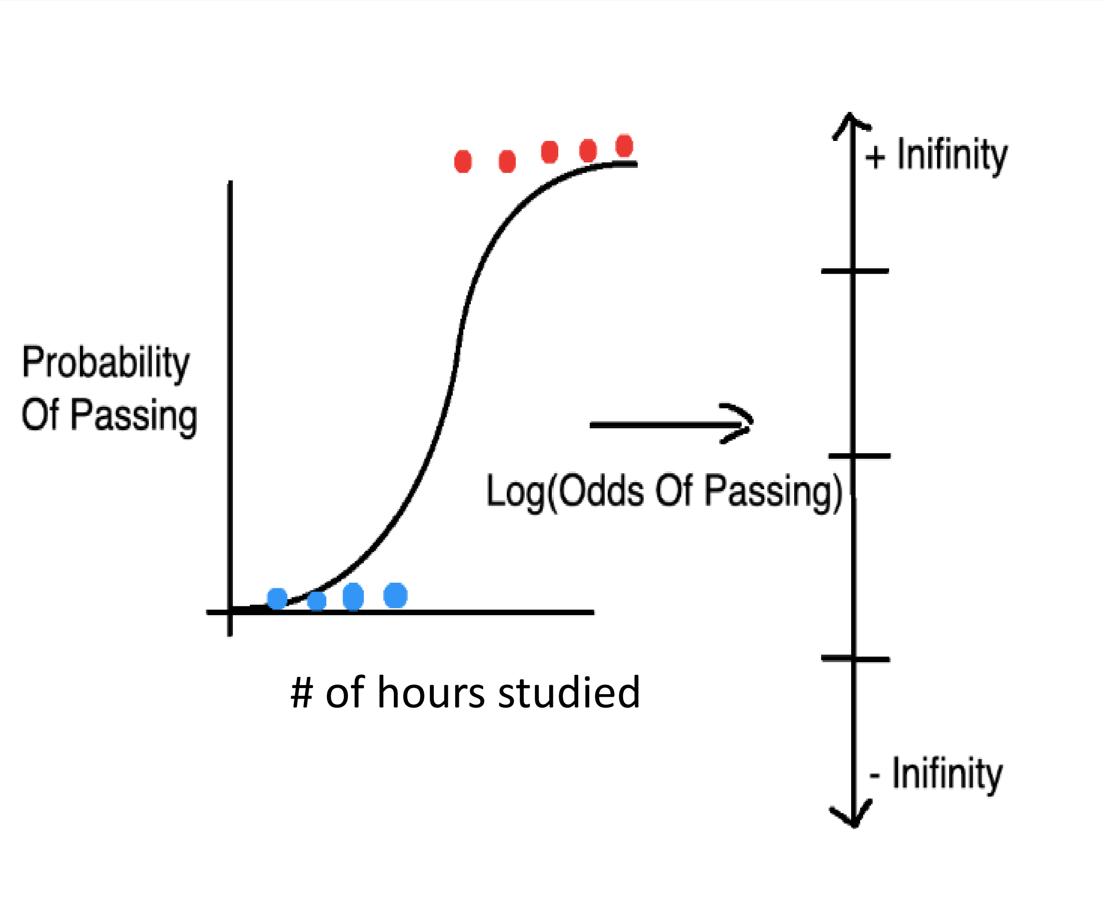
$y \rightarrow -\infty$

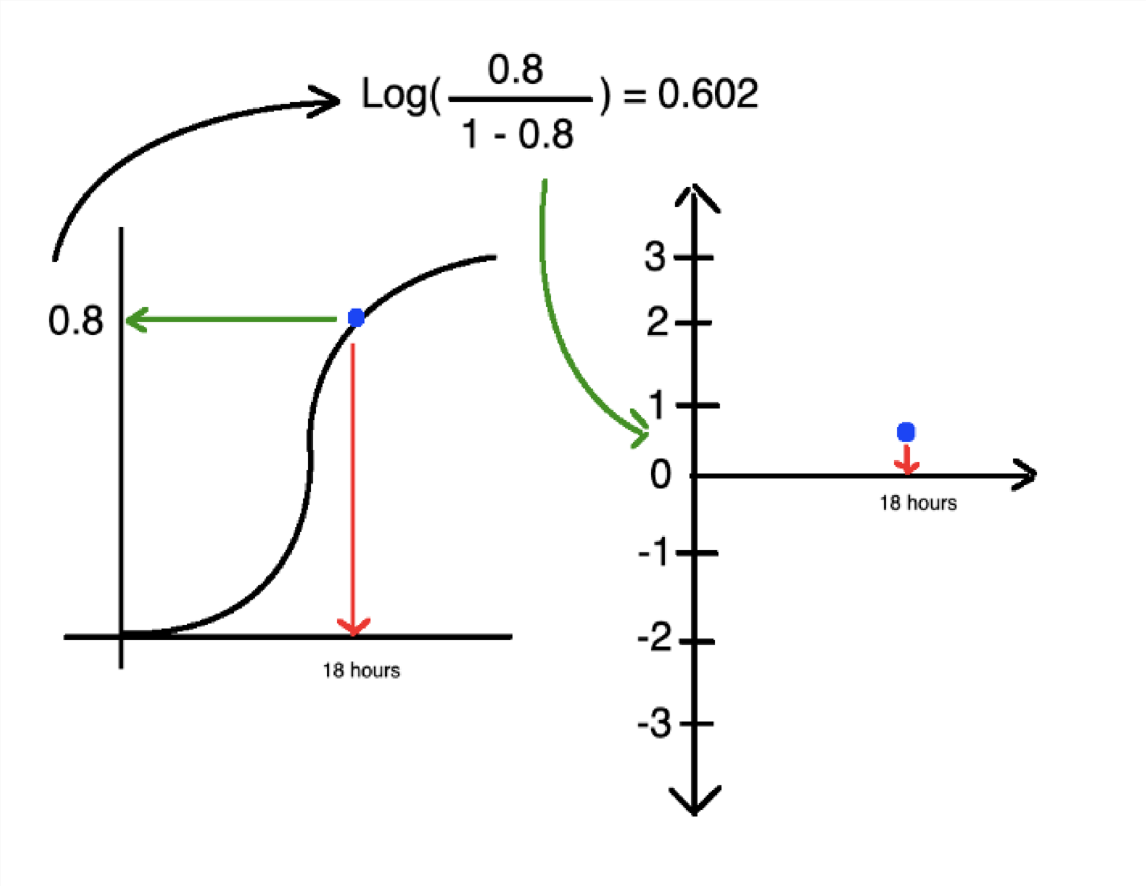
$$P = \frac{1}{1+e^{-(-\infty)}} = \frac{1}{1+e^{+\infty}} = \frac{1}{\infty} = 0$$

$y \rightarrow +\infty$

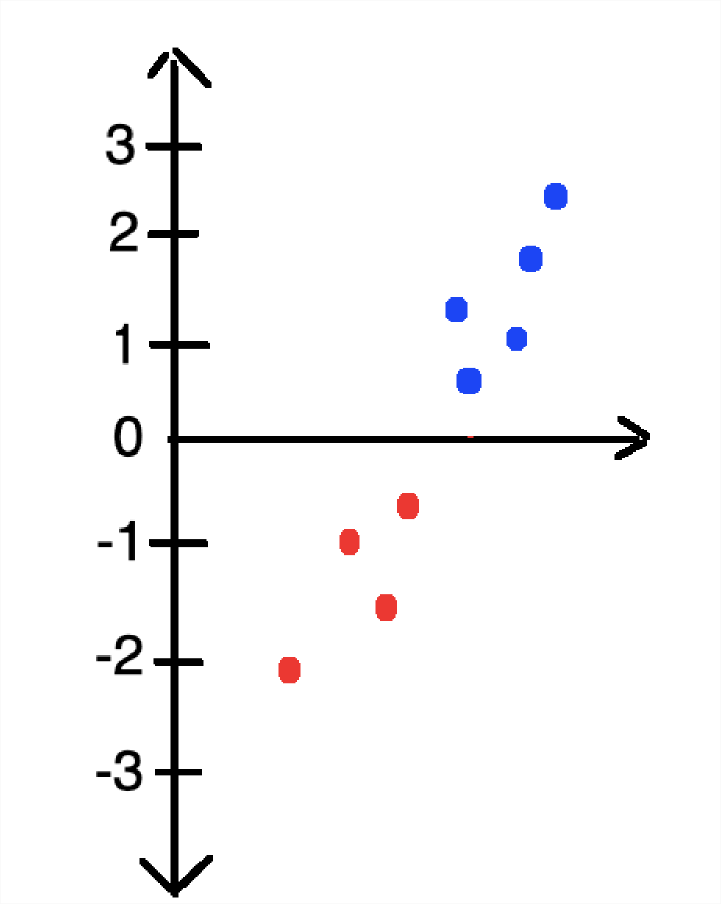
$$P = \frac{1}{1+e^{-\infty}} = \frac{1}{1+\frac{1}{e^{+\infty}}} = \frac{1}{1+\frac{1}{\infty}} = \frac{1}{1+\frac{1}{\infty}} = \frac{1}{1+0} = 1$$

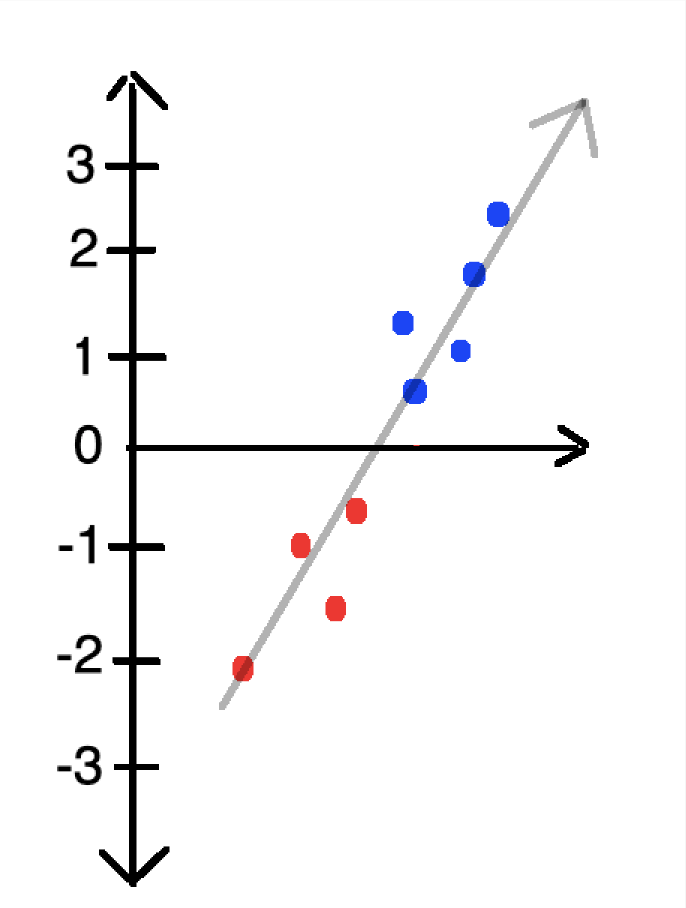
Transform between Probability space to the features space



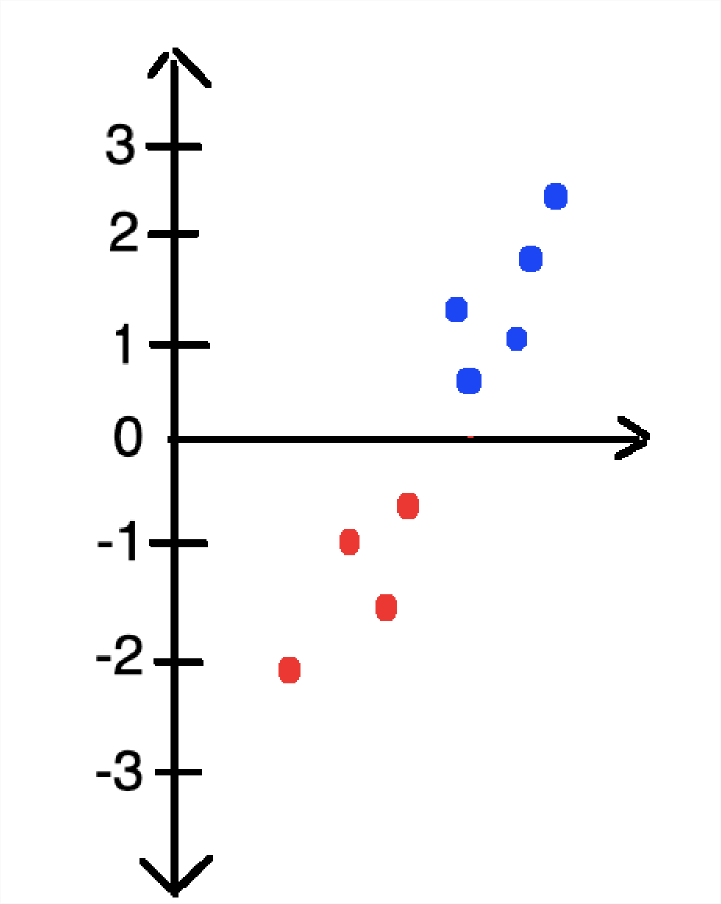


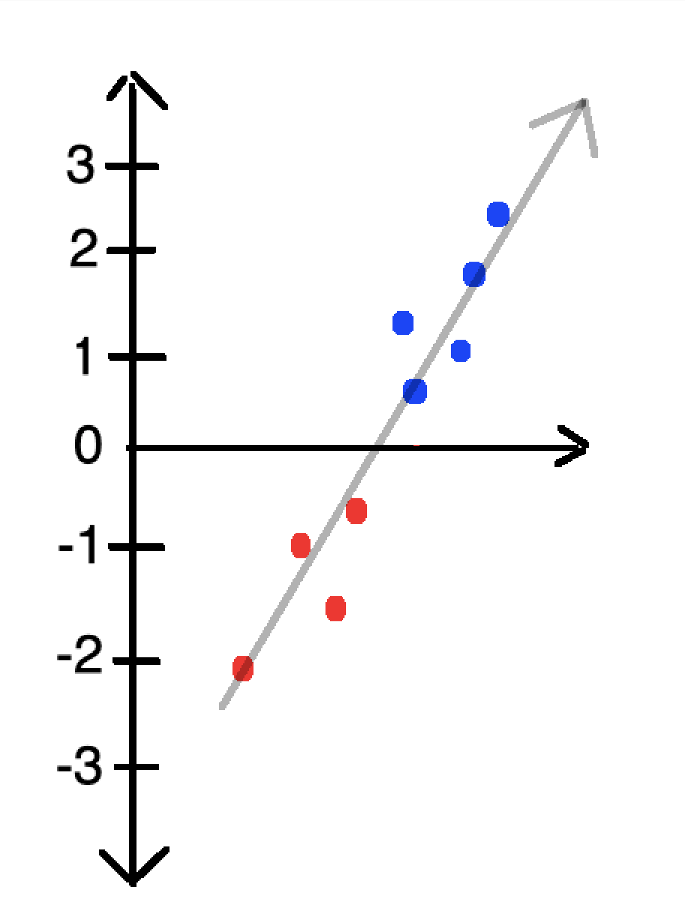
Repeat for each data points



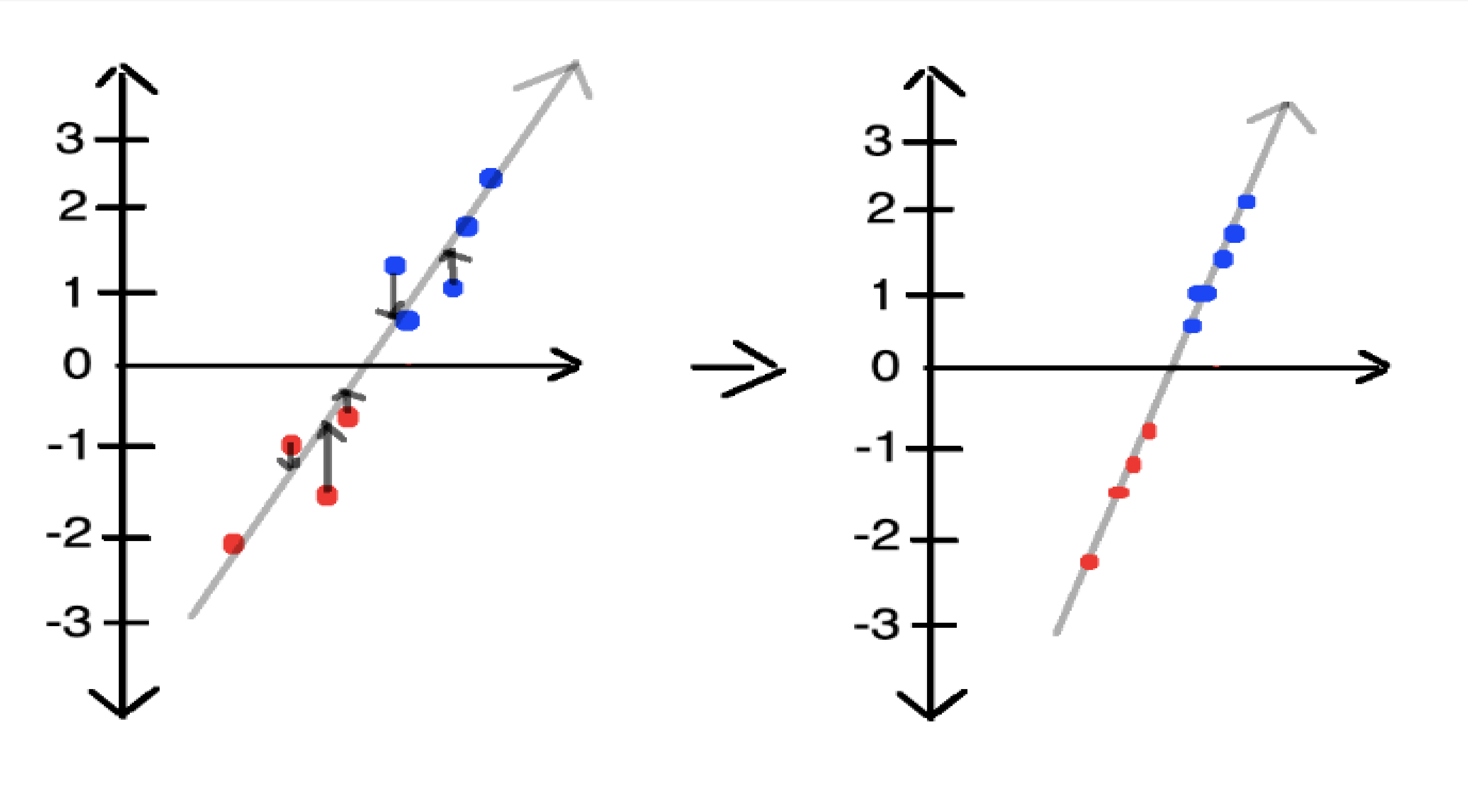


Repeat for each data points

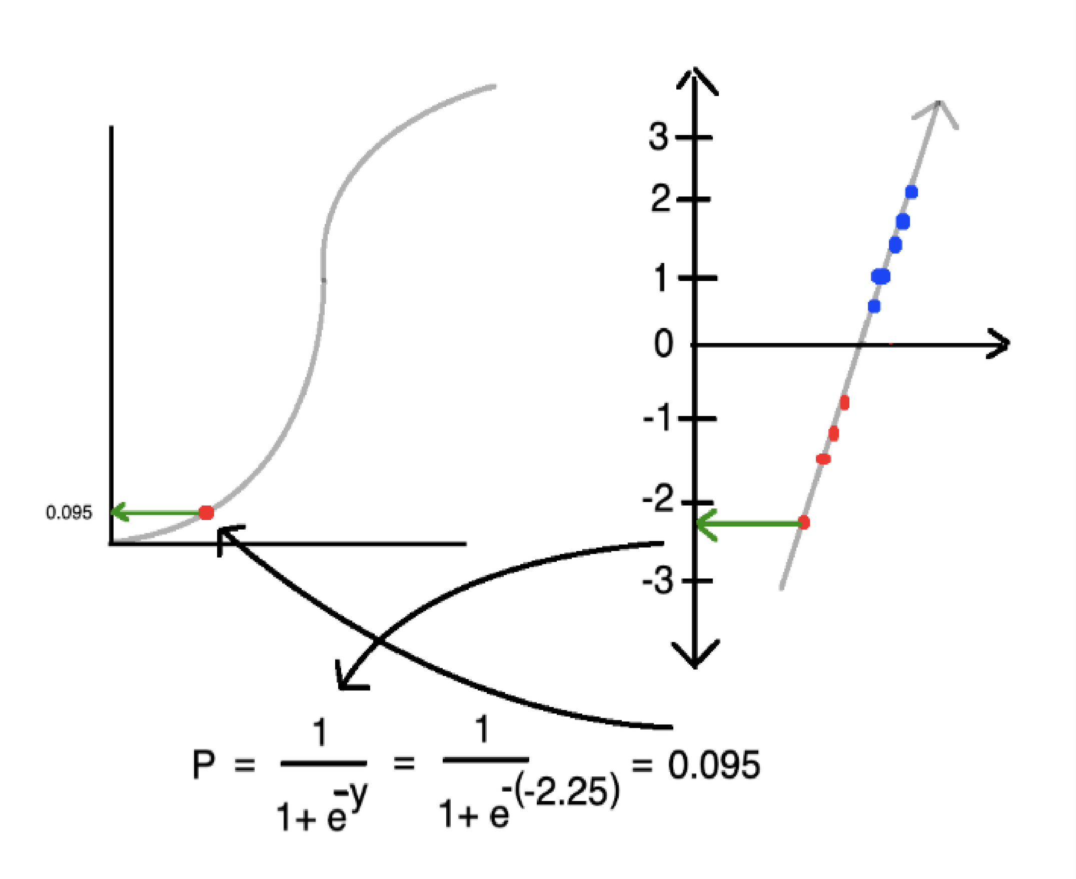




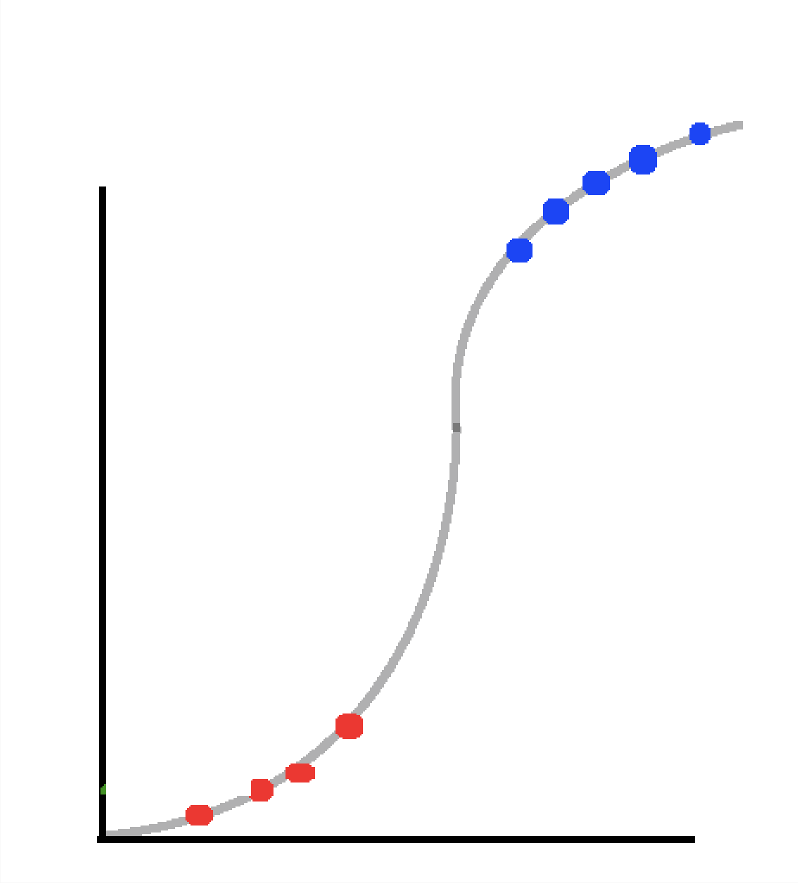
Now try to use Least Square to fit a line



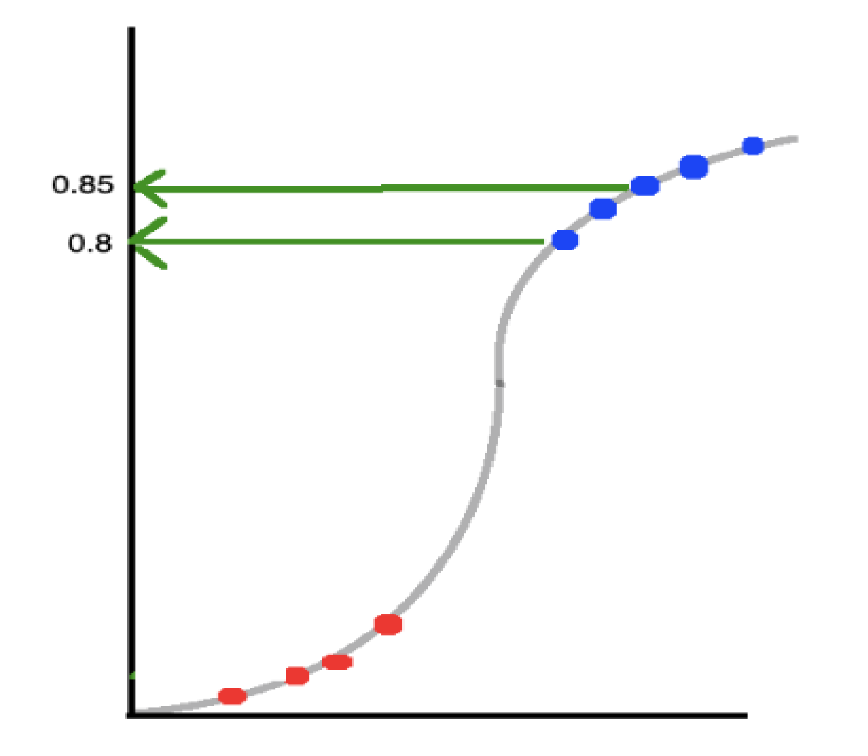
Once we have the fitted line, we can transform from the feature space back to the Probability space

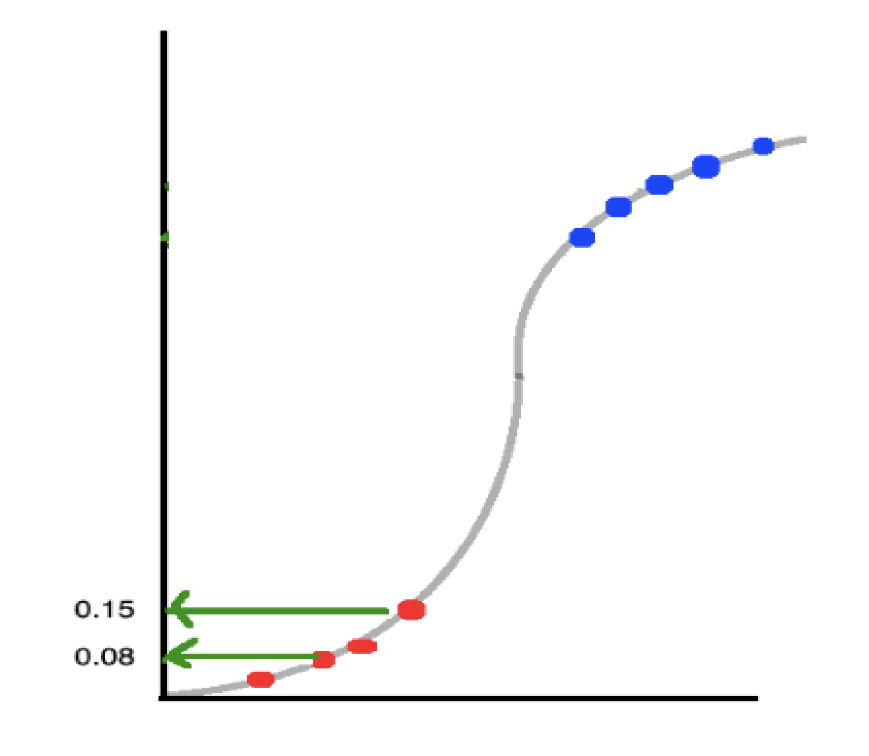


$$P = \frac{1}{1+e^{-y}} = \frac{1}{1+e^{-(-2.25)}} = 0.095$$



Now we have the data point on the Sigmoid curve





Likelihood = 0.8 x 0.82 x 0.85 x 0.89 x 0.91…

Likelihood = 0.8 x 0.82 x 0.85 x 0.89 x 0.91 x (1 - 0.15) x (1 - 0.12) x (1 - 0.08) x (1 - 0.05)

Likelihood Function

- Likelihood Function is the function that calculates the probability of observing the data that we have observed.

$$L(\theta; x) = \prod_{i=1}^{i=N} Prob(x_i; \theta)$$

- Maximum likelihood estimation is a method that determines values for the parameters (θ) of a model. The parameter values are found such that they maximise the likelihood that the process described by the model produced the data that were actually observed.

- Instead of considering L(θ;x)), we will consider the Log of the likelihood as maximizing Log(L) is the same as maximizing L.

How to find the “best-fitted line

- Remember in Linear Regression, to find the best-fitted line by

  Minimize the cost function J $(\theta; x)$ = MSE = $\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$

  where h(x) is the prediction function $h_\theta(x) = \theta_0 + \theta_1 x$

  Hypothesis

- In Logistic regression, Cost Function is the negative of the Log (Likelihood) function

  Maximizing Likelihood = Maximizing Log(likelihood)

  = Minimizing Cost Function defined by – Log(likelihood)

$$J(\theta; x) = \begin{cases} -\log(h(x)) \text{ when } y = 1 \\ -\log(1 - h(x)) \text{ when } y = 0 \end{cases}$$

h(x) = 1/ (1- exp(- theta * x)

Optimization problem

- In most machine learning models, we find the best fit model by first defining a cost function

$$J\ (\theta; x)$$

Then we use a solver to find the value of the theta's so that the cost function is minimized

- In linear regression, one can find closed form solution
- In a more general optimization problem, there is no closed form solution, one will need to use various numerical methods

- Gradient descent is the most common way to solve this optimization problem

Gradient Descent in Linear Regression

$$h_\theta(x) = \theta_0 + \theta_1 x$$

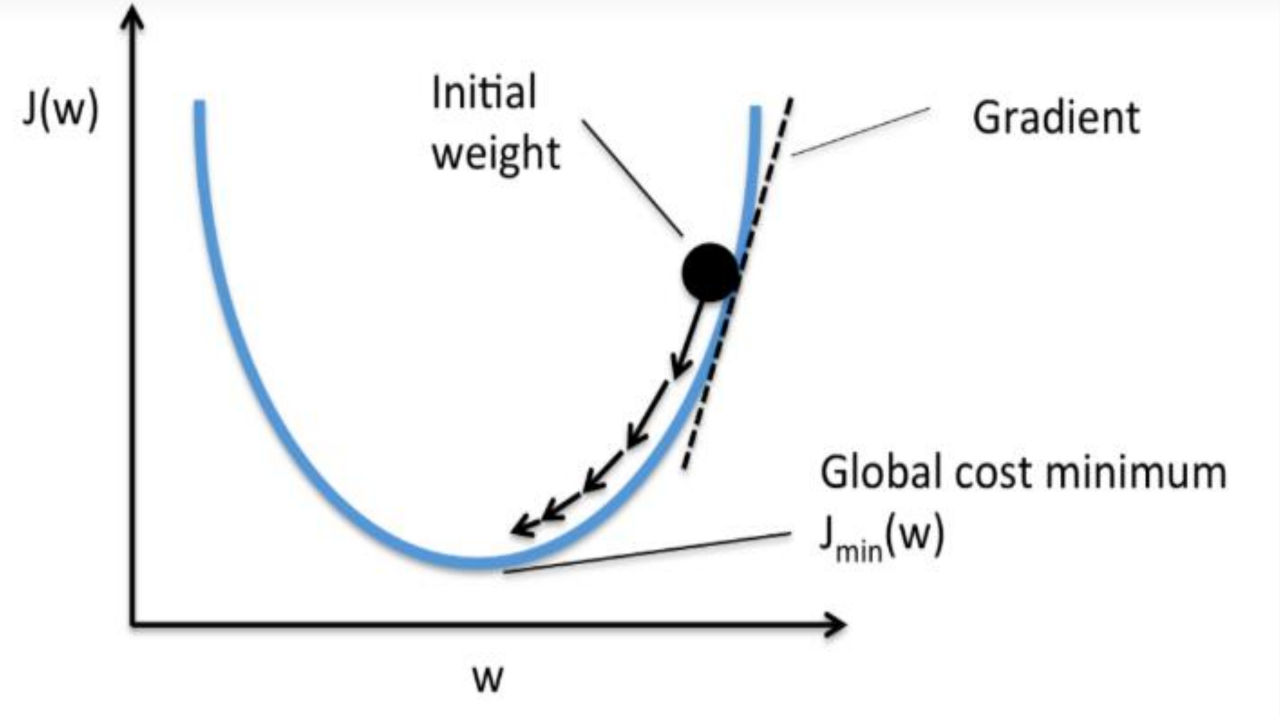
Hypothesis

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1) \quad (\text{for } j = 0 \text{ and } j = 1)$$

Gradient Descent

In practice, we just call the "Fit" method from the library



Gradient Descend Visualization. Credit: rasbt.github.io

https://medium.com/@lachlanmiller_52885/machine-learning-week-1-cost-function-gradient-descent-and-univariate-linear-regression-8f5fe69815fd

Model Performance (All models are wrong, but some are useful)

Confusion Matrix

Predicted Values \ Actual Values	Positive (1)	Negative (0)
Positive (1)	TP	FP
Negative (0)	FN	TN

$$F1Score = 2\left(\frac{Precision \times Recal}{Precision + Recal}\right)$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Precision	$= \frac{TP}{TP + FP}$	Out of the ones you claims positives, how many are correct. To increase Precision, you try to be conservative in claiming positive case, but you risk missing out
Recall	$= \frac{TP}{TP + FN}$	Out of the correct positives, how many you pick up in your prediction. To increase Recall, try to predict positive even though the evidence is not strong, but you risk increase false positive rate

We want both Precision and Recall to be high > 80%, but there is a trade-off
F1-score is a one single metric to combine both Precision and Recall

https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2

Logistic Regression

Learning by doing

Some references

Andrew Ng's popular Machine Learning class video is on
https://www.youtube.com/playlist?list=PLLssT5z_DsK-h9vYZkQkYNWcItqhlRJLN

But let's focus on Cost Function on Linear Regression as well as Logistic Regression which are on
Lecture 2.2, 2.3, 2.4, 2.5, 2.6, 6.2, 6.4 and 6.5

Lecture 2.2 https://www.youtube.com/watch?v=yuH4iRcggMw&list=PLLssT5z_DsK-h9vYZkQkYNWcItqhlRJLN&index=6&t=0s
Lecture 2.3 https://www.youtube.com/watch?v=yR2ipCoFvNo&list=PLLssT5z_DsK-h9vYZkQkYNWcItqhlRJLN&index=6
Lecture 2.4 https://www.youtube.com/watch?v=0kns1gXLYg4&list=PLLssT5z_DsK-h9vYZkQkYNWcItqhlRJLN&index=7
Lecture 2.5 https://www.youtube.com/watch?v=F6GSRDoB-Cg&list=PLLssT5z_DsK-h9vYZkQkYNWcItqhlRJLN&index=8

Lecture 6.2 https://www.youtube.com/watch?v=t1IT5hZfS48&list=PLLssT5z_DsK-h9vYZkQkYNWcItqhlRJLN&index=33
Lecture 6.4 https://www.youtube.com/watch?v=HIQlmHxI6-0&list=PLLssT5z_DsK-h9vYZkQkYNWcItqhlRJLN&index=35

https://medium.com/@rgotesman1/learning-machine-learning-part-3-logistic-regression-94db47a94ea3