# VGP337 - Neural Network & Machine Learning

Instructor: Peter Chan

# The journey so far...

- Recall that in linear regression, we are given a dataset where each example is consists of a feature vector X, and a label y, both in real numbers

$$\{y_i, x_{i1}, \ldots, x_{ip}\}_{i=1}^{n}$$

- In particular, we looked at the most basic form of it, called Simple Linear Regression, where we use the following hypothesis function:

$$y = \beta_0 + \beta_1 x$$

- Furthermore, we defined a cost function using Mean Square Error and applied Gradient Descent to solve for the coefficients $\beta_0$ and $\beta_1$, which gives us our prediction model

# Logistic Regression

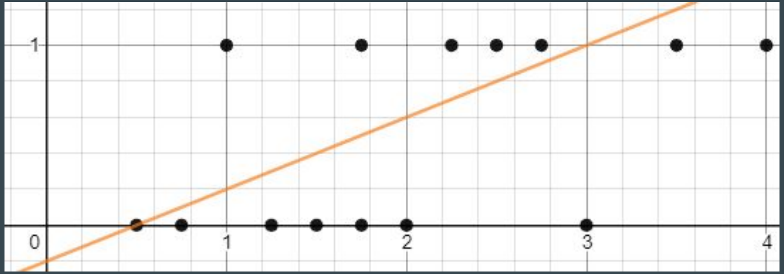
- Linear regression allows us to train a model which can be used to predict something continuous (scalar value)
- Logistic regression is similar to linear regression, except that it is used to predict the likelihood of whether something is True or False
- Strictly speaking, it is not a classification algorithm, but it can be used as one when a threshold is applied to form a decision boundary
- Here is an example of what a dataset for logistic regression may look like:

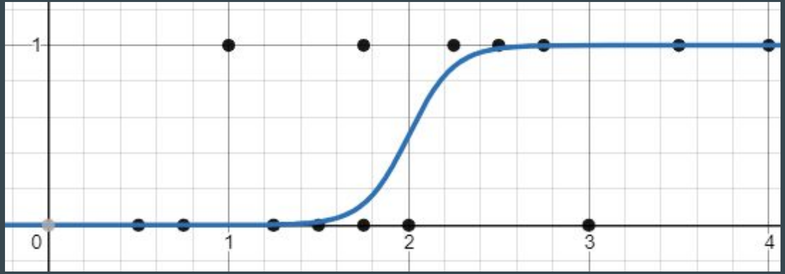
Number of hours spent studying and whether the student passed or failed

| **Hours (x)** | 0.50 | 0.75 | 1.00 | 1.25 | 1.50 | 1.75 | 1.75 | 2.00 | 2.25 | 2.50 | 2.75 | 3.00 | 3.50 | 4.00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Pass (y)** | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |

# Logistic Regression

- Notice that the dependent variable y is binary and can take values 0 or 1
- As shown below, it is clear that linear regression is not a good model for this data
- Instead, in logistic regression, we use a different hypothesis function called the sigmoid function (sometimes referred as the logistic function)



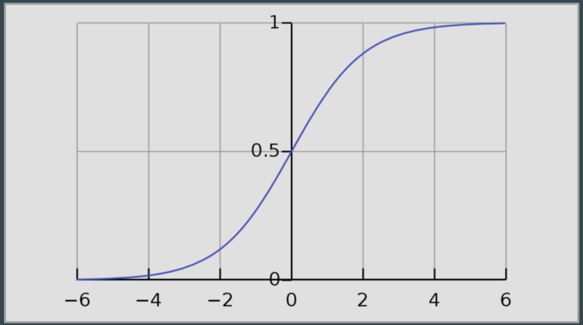
# Sigmoid Function

- The sigmoid function maps arbitrary real values to the range (0, 1)
- Here is how we can apply the sigmoid function to our hypothesis:

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

$$z = \theta^T x$$

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




- We first map our input features as a weighted sum $z \in R$
- Then we apply sigmoid to transform $z$ to (0, 1)

# Decision Boundary

- Since the output of the hypothesis is not discrete (compared to the label y), we think of the result of the prediction as the probability that each input belongs to a particular category.
- We can translate the output of the hypothesis as follows:

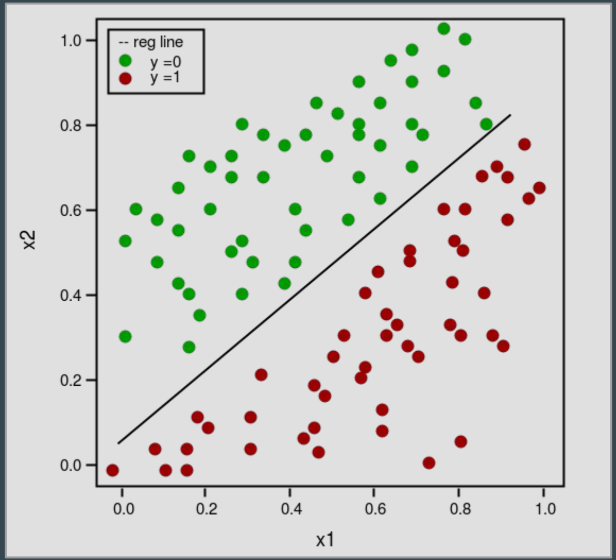
$$h_\theta(x) \geq 0.5 \rightarrow y = 1$$
$$h_\theta(x) < 0.5 \rightarrow y = 0$$

- From this, we can say that:

$$z = \theta^T x \geq 0.0 \rightarrow y = 1$$
$$z = \theta^T x < 0.0 \rightarrow y = 0$$

# Decision Boundary

- The decision boundary is the line that separates the area where y = 0 and y = 1
- You can plot this once you have estimated θ



# How do you estimate θ?

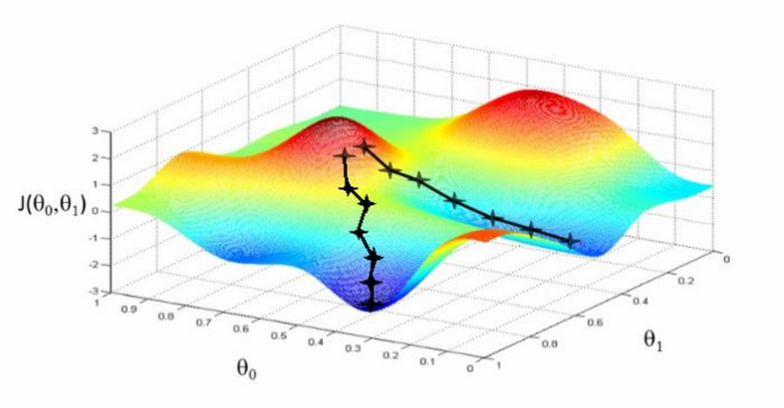
- Once again, we need to define a cost function and then employ gradient descent to minimize the error
- We can try to apply the same method we used for linear regression, namely:

$$MSE = \frac{1}{N} \sum_{(x,y) \in D} (y - prediction(x))^2$$

- Unfortunately, since our hypothesis function is no longer linear,  this would lead to a non-convex function, meaning there will be multiple local minima

# How do you estimate θ?

- Having multiple local minima means that gradient descent can no longer guarantee the best answer
- i.e. Depending on your initial guess, you may get different answers



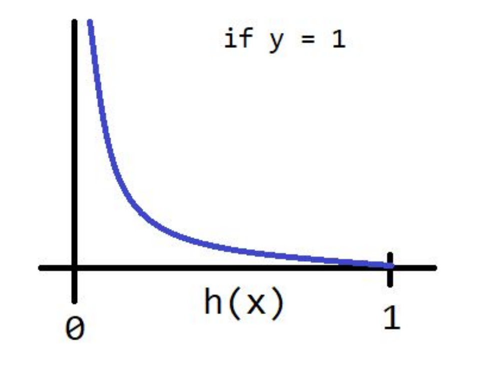
# Cost Function

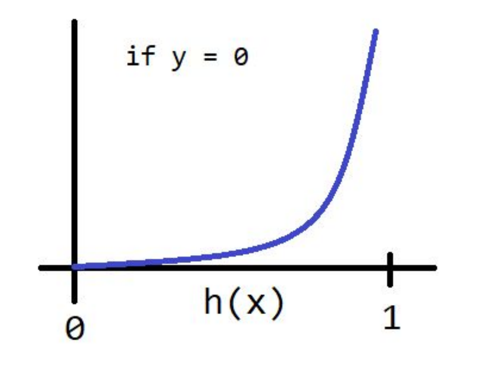
- Instead, the cost function for logistic regression looks like this:

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

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

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



# Cost Function

- Note that if the correct answer for y is 0, then the cost function will be 0 when our prediction is also 0, and infinite when our prediction is 1
- On the other hand, if the correct answer for y is 1, then the cost function will be infinite when our prediction is 0, and vice versa
- The key is that this cost function guarantees that $J(\theta)$ is convex, and hence we can use gradient descent to solve it

# Gradient Descent

- We first compress the cost function into this form:

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

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

- Recall that the general form of gradient descent is as follows:

$$\begin{aligned}&Repeat\ \{\\ &\quad \theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta)\\ &\}\end{aligned}$$

# Gradient Descent

- Taking the derivative of J(θ), we will get:

$$
\begin{aligned}
&Repeat\ \{\\
&\theta_j := \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\\
&\}
\end{aligned}
$$

- And that's it!

# Multi-class Classification

- What if your dataset has examples with more than two category?
- Instead of y = {0, 1}, you may have y = {0, 1, ..., n}
- In this case, we simply divide our problem into n+1 binary classification problems
- Each problem will classify values as *m* and *not m*
- In other words, we choose one class and then combine the rest into another class
- Finally, we use the highest hypothesis value as our prediction

# Multi-class Classification

- This technique is referred as One-vs-all

$$y \in \{0, 1 \ldots n\}$$

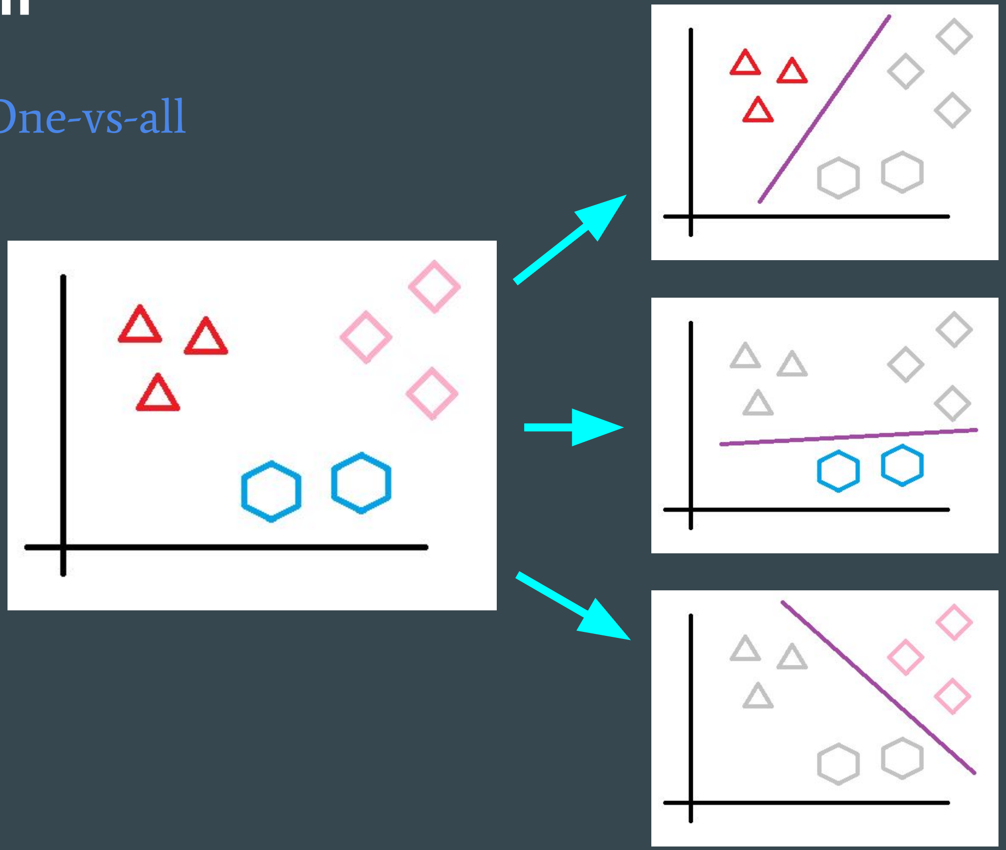
$$h_\theta^{(0)}(x) = P(y = 0|x;\theta)$$

$$h_\theta^{(1)}(x) = P(y = 1|x;\theta)$$

$$\cdots$$

$$h_\theta^{(n)}(x) = P(y = n|x;\theta)$$

$$\text{prediction} = \max_i(h_\theta^{(i)}(x))$$



# Here are some good references

[Logistic Regression: Calculating a Probability]

[Understanding Logistic Regression]

[Logistic Regression from scratch in Python]

[Logistic Regression — Detailed Overview]