Logistic Regression:

* Basically: Take the summation of (bias \* x) then add constant at the end to influence the output before equating it to . After Z is found then it is applied to a sigmoid function. Keep in mind that bias is different for each x and c is constant and applied after summation.
  + Sigmoid function basically means that as Z is larger or positive (X-axis) the Y becomes more likely to be 1, and vice versa with the more negative Z is the more likely it is to be 0.
    - Keep in mind that data has to be fitted so that features don’t make Z a span of like 200 then 1.2. Ya know, condense it so that features will result in a small Z value positive or negative
* Limitation of Logistic Regression:
  + Logistic regression is good if there is a distinction between class one and class two, basically binary classification. Yes or no and gives a probability between them. Ex: 70% class A | 30% Class B
  + In addition, Linear or line regression is good if we can establish a solid line through two distinct data sets, if the separation between the two classes is like a curve or zig zag, then it sucks, cuz there isn’t a definite line to separate the two classes.
  + A screenshot of a computer screen

    Description automatically generated
  + Keep in mind that you can use linear regression if there are multiple classes, but like I said you need a definite line if you want accuracy high.

Linear Regression Vs Logistic Regression:

* Linear Regression:
  + Used to predict a continuous value: (Price | Height | Age | Distance)
  + Equation: y = β₀ + β₁x₁ + β₂x₂ + ... + βₖxₖ + ε
  + Where:
    - y is the target variable (continuous)
    - x₁, x₂, ..., xₖ are the k input features
    - β₀ is the intercept or bias term
    - β₁, β₂, ..., βₖ are the coefficients or weights for each feature
    - ε is the error term (residual)
  + Method To Fit Equation: **Ordinary Least Squares**
  + ***The goal of linear regression is to find the optimal values of the coefficients that minimize the sum of squared residuals between the predicted values and the actual target values***
* Logistic Regression:
  + Used to predict a categorical value: (Yes or No | Male or Female | Win or Not Win)
  + Equation:
  + Where:
    - p is the predicted probability of the target variable being 1
    - z = β₀ + β₁x₁ + β₂x₂ + ... + βₖxₖ
    - β₀ is the intercept or bias term
    - β₁, β₂, ..., βₖ are the coefficients or weights for each feature
    - x₁, x₂, ..., xₖ are the k input features
  + Method to Fit Equation: **Maximum Likelihood Estimation**
  + Outputs a percentage or probability, rather than a continuous value
  + ***The goal of logistic regression is to find the optimal values of the coefficients that maximize the likelihood of observing the target variable values in the training data.***
* Examples:
  + Suppose an economist wants to use predictor variables (1) weekly hours worked and (2) years of education to predict the annual income of individuals.
    - In this scenario, he would use **linear regression** because the response variable (annual income) is continuous.
  + Suppose a college admissions officer wants to use the predictor variables (1) GPA and (2) ACT score to predict the probability that a student will get accepted into a certain university.
    - In this scenario, she would use **logi9stic regression** because the response variable is categorical and can only take on two values – accepted or not accepted.
  + Suppose a real estate agent wants to use the predictor variables (1) square footage, (2) number of bedrooms, and (3) number of bathrooms to predict the selling house prices.
    - In this scenario, she would use **linear regression** because the response variable (price) is continuous.
  + Suppose a computer programmer wants to use the predictor variables (1) number of words and (2) country of origin to predict the probability that a given email is spam.
    - In this scenario, he would use **logistic regression** because the response variable is categorical and can only take on two values – spam or not spam.
* A screenshot of a graph

  Description automatically generated
* A diagram of a data analysis

  Description automatically generated with medium confidence

Deep Learning:

* The basis of neural networks. Taking feature data apply a logistic function to them, categorize the output to whichever nodes aligns better, then apply a different logistic function to them, then either repeat (multilayer rendition) or apply a softmax or sigmoid to get values between 0 and 1 (2 layer). Then you have your probability.
* So lets say the input layer has 20 representing 20 different features and there are 100 rows of data elements, then the next layer has 10.
  + Originally we have a 100 rows, with 20 data columns per each row to describe the data point. This single row can be described as a 1x20 matrix or a vector of 20 values
  + So with the dataset we have a 100x20 matrix we then transform that matrix into a 100x10 matrix since there are 10 nodes in the next layer.
  + How do we do this?
    - We initially apply a weight matrix W with dimensions 20x10 via matrix multiplication. 20 is the input features and 10 is the number of output nodes in the next layer, Each column in W represents the weights associated with a particular node out of the 10 nodes.
    - Keep in mind that with each node there is a weight and bias associated with it. So the next step is to apply a bias via matrix addition. The bias is a vector of 10 or a 1x10 matrix.
    - **Linear Combination:** For each row(instance) in the input matrix X, we compute the linear combination of the features and the corresponding weights, plus the bias term.
    - Then we apply the logistic function to get the output between 0-1. Use sigmoid or whatever function you want.
  + So each column in the new matrix is basically the probability of belonging to a certain node, and since there are 10 nodes in the first layer, there are 10 columns per row or vector signifying the probability of belonging to a certain node.
  + Then either repeat with same amount of output nodes or shrink it to where you eventually reach final layer.

Transfer Learning:

* When utilizing a neural network, transfer learning is a ML technique where a pre-trained model is reused as the starting point for a model on a new, but related task. This approach is particularly popular in deep learning because it allows for training deep neural networks with comparatively little data, which is beneficial in the data science field where most real-world problems do not have millions of labeled data points to train such complex models.
* The core idea behind transfer learning is to leverage the knowledge a model has learned from a task with a lot of available labeled training data in a new task that doesn’t have much data. Instead of starting the learning process from scratch, we start with patterns learned from solving a related task. This is particularly useful in computer vision and natural language processing tasks, where the computational power required is immense.
* In practice, transfer learning works by reusing the early and middle layers of a neural network trained on a large dataset, such as ImageNet, and only retraining the latter layers for the new task. This approach allows the model to leverage the labeled data of the task it was initially trained on, significantly reducing the amount of data and computational resources needed for the new task. For example, a model trained to recognize a backpack in an image can be repurposed to identify sunglasses by only retraining the latter layers to learn what separates sunglasses from other objects.
* Transfer learning is not an exclusive part of machine learning but has become quite populat in combination with neural networks that require huge amounts of data and computational power. It’s a design methodology within the field that allows for rapid progress when modeling new tasks by leveraging existing models.

How do we Evaluate Our Networks:

* Complex Relationships Using Deep Learning
  + Can be captured by using deep neural networks
  + Can be represented accurately and predicted well
  + Can give perfect performance in the training set
  + Can perform poorly in the real world
  + Needs to be validated
* Overfitting
  + Overfitting is when the learned model increases complexity to fit the observed training data too well
  + Good at predicting our training set, but when introduced to new training data, it will suck
  + Let’s say we want to increase complexity to better fit the model
  + A diagram of a function

    Description automatically generated
    - As you can see it eventually the line gets better at representing the dataset, but imagine if we used the 3rd model on a unknown dataset, it would suck bad
  + Problems with Overfitting
    - Increasing parameters increases error rate
    - Complex relationship maybe too complex for reality
    - Models and analysis are not generalized
* Test Set
  + Standard practice in machine learning
  + Created prior to any analysis
  + Will never be used to learn or fit any parameters
  + Can evaluate performance of network on test set
  + Analogous to running a new experiment
  + Should ideally only be used once
  + Reusing a test set will lead to bias
  + Bias results will lead to optimistic performance estimates
* Validation Set
  + Can be used to compare which approach is best
  + Not used to learn parameters
  + Used repeatedly to estimate the performance of a model
  + Can be used to pick out the best performance model
* Method
  + Use training set to learn parameters
  + Then use validation set to estimate performance and then use this metric to refine the model (more vs less complexity)
  + Then use test dataset once to measure final performance.