Workshop #1:

Intro to ML with R

Presented by: UF American Statistical Association

ASA
University of Florida

Table of contents

Welcome!

Please sign in for attendance

01

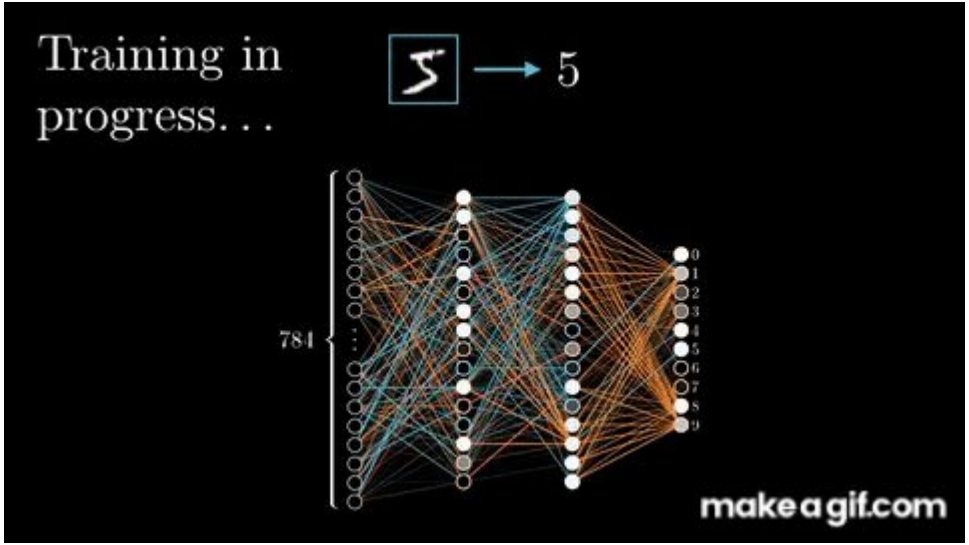
What is Machine Learning?

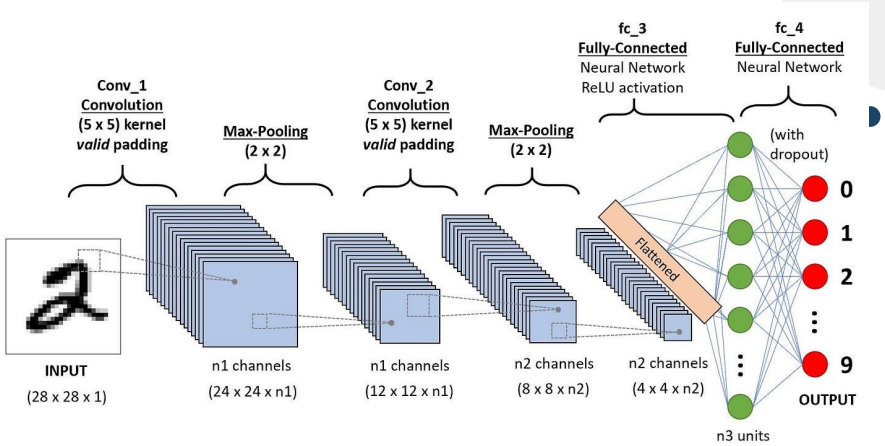
How do we define Machine Learning?

- IBM define Machine learning as " a branch of computer science that focuses on the using data and algorithms to enable AI to imitate the way that humans learn, gradually improving its accuracy."
- It is essentially statistical algorithms that can be used to learn from data and generalize patterns found
  - This allows these algorithms to predict values with varying degrees of accuracy on unseen data
- Is machine learning and deep learning the same thing?
  - Not exactly. They are both subsets of computer science and artificial intelligence
  - Deep Learning uses many layers of neurons to learn unstructured data such as images
  - Deep Learning does not require human intervention for using unstructured data unlike ML algorithms

[1] https://www.ibm.com/topics/machine-learning

Deep Learning

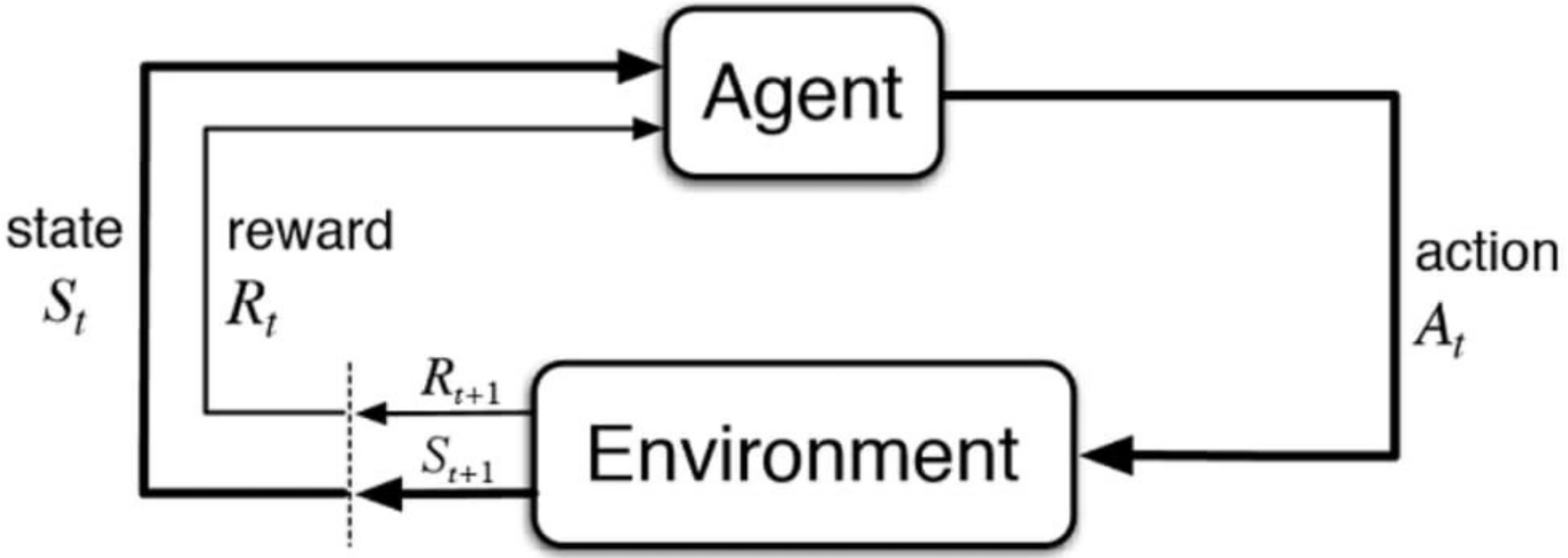




What Are the Different Types of Learning?

- Supervised Learning
  - Model requires a labelled dataset
  - Example dataset: A dataset of images labelled as cats or dogs
  - Regression and Classification
- Unsupervised Learning
  - Model learns using dataset where the data is not labelled
  - Example dataset: A set of customer reviews
  - Clustering Algorithms: K-Means, KNN, Gaussian Mixture Models
- Reinforcement Learning:
  - Learn which actions to take by maximizing or minimizing a reward basis through reinforcement

Reinforcement Learning



02

Why R?

Isn't Python Used For Machine Learning

- Nope, R is a incredibly popular in data science
  - According to the 2022 Kaggle Machine Learning & Data Science Survey, 55.2% of data scientists and ML practitioners use R regularly, compared to 78.4% for Python
  - The TIOBE Index, which measures programming language popularity, consistently ranks R in the top 20 languages, often in the top 15
- The technologies used for machine learning are highly dependent on the company
- R has powerful libraries that allows data scientists to clean and process company data

R Libraries for Machine Learning

- e1071
  - A library originally developed for the e1071 course
  - The first implementation of the Support Vector Machine
  - e1071 package is used mostly for clustering algorithms
- TidyModels
  - Collection of models and algorithms used for machine learning
  - Based of the principles of tidyverse
- Caret
  - Classification And Regression Training
  - Contains a set of functions to speed up developing models
    - data splitting, pre-processing, feature selection, model tuning
- XGBoost
  - A library for the extreme gradient boosting model

03

System Setup

Setup Google Colab

Go to: https://github.com/KKonuru/Intro_ML_R and click the first link in the readme to the colab notebook

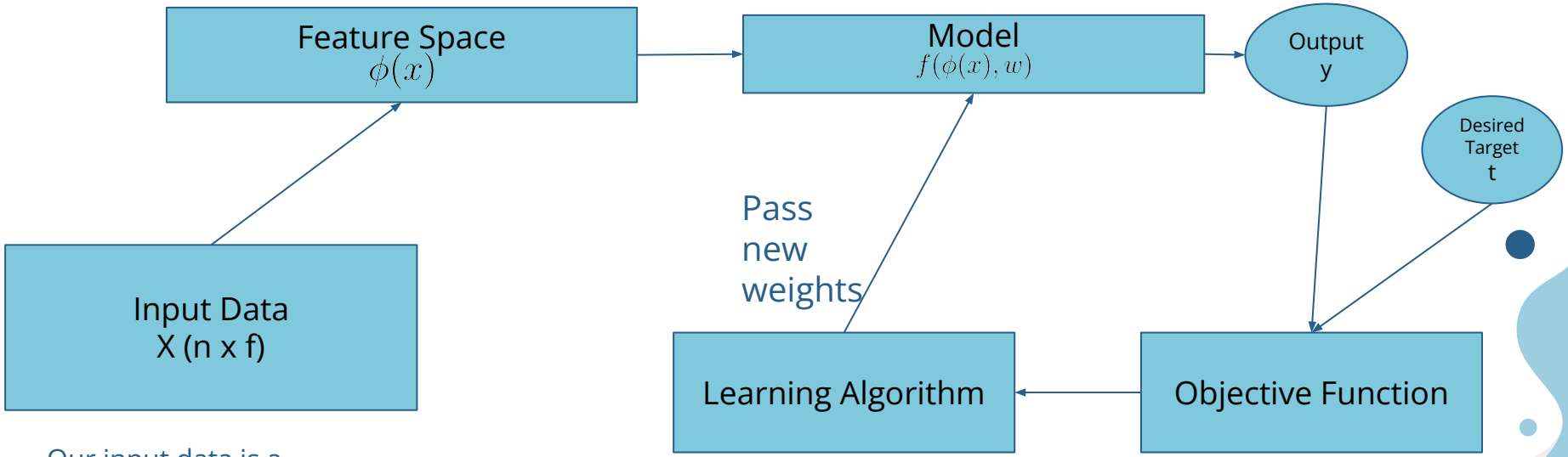
Create a copy of the notebook in your google drive

04

Machine Learning Algorithms

Supervised Learning Flowchart

- We will be covering supervised learning algorithms for classification and regression. But how does this process work?



Our input data is a matrix of n instances with f features

Linear Regression

- Regression model uses a mapper function that uses the input to predict a value
  - These values are continuous
- Input data is stored in matrix $$X$ with n rows and f columns. n is the number of instances and f is the number of features
- A simple linear regression of the form:

  $$y = x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + x_4 * w_4.. + x_f * w_f + w_0$$

  Has feature space:

  $$\phi(x) = [x_1, x_2, x_3, x_4, ..., x_f]^T$$

  We use the objective function Mean Squared Error:

  $$J(w) = \frac{1}{N}\sum_{i=1}^{N}(t_i - y_i)^2$$

  We then derive a learning algorithm to update the weights W
  w = (X^TX) X^T *t

Regularization

- From our linear regression problem, we saw that the model to predict the values uses weights that are updated after every iteration
- But how do we prevent very large weights from being used or penalize outliers
- L1 Regularization
  - Lasso Regularization
  - Creates weight vector with sparsity

$$\lambda \sum_{j=1}^{f} |w_j|$$

- L2 Regularization
  - Ridge Regularization
  - More harshly penalizes outliers

$$\lambda \sum_{j=1}^{f} w_j^2$$

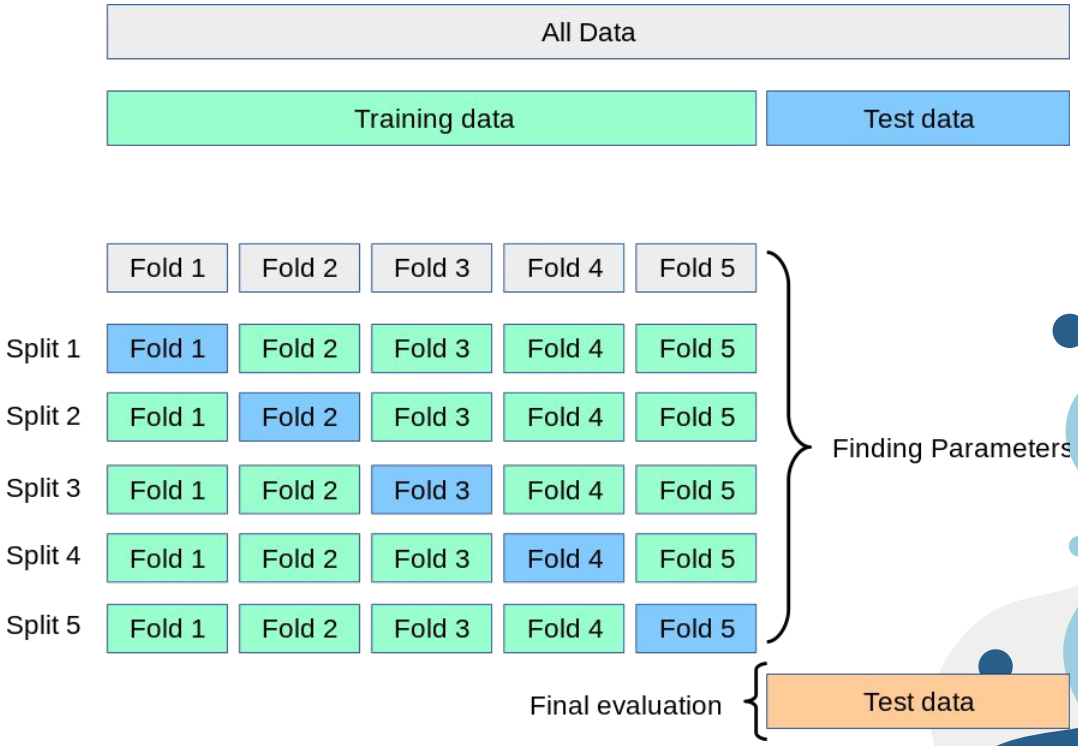
- Add regularization term to the end of the objective function
- Lambda is a **hyperparameter** we adjust when defining the model

$$J(w) = \frac{1}{N} \sum_{i=1}^{N} (t_i - y_i)^2 + \lambda \sum_{j=1} f|w_j|$$

Cross Validation

- A technique to check the effectiveness of a model
- Different types of cross validation
    - K-fold cross-Validation
    - Repeated K-fold
    - Leave one out
    - Validation Set

- We use cross validation in the training to test the performance on unseen set



Naive Bayes Classifier

- Classification algorithm that uses Bayes theorem
- Assumes the features are independent and normally distributed
- Example. We have two classes A and B

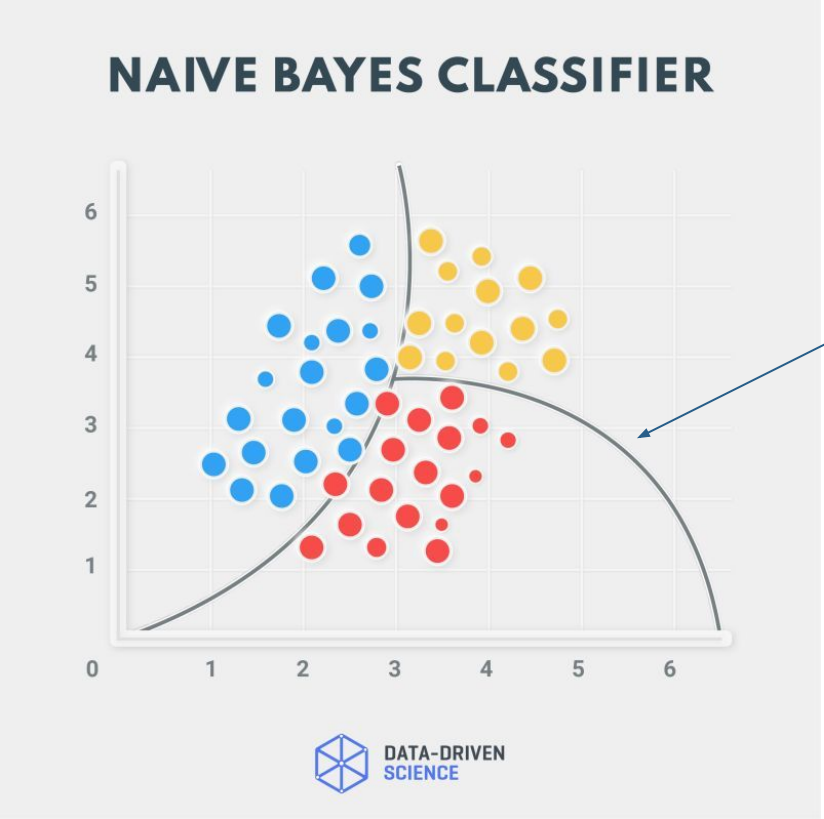
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

P(A) is the priori or prior probability
P(A|B) is the posteriori probability or the probability once we have collected data
P(B|A) is the likelihood probability

The class a instance belongs to is the class with the highest likelihood probability.

We try to learn a gaussian distribution for each class in the feature space

Naive Bayes Classifier



We get a discriminant function separating the classes after forming gaussian distribution for each class

Support Vector Machine

- Supervised algorithm that can be used for regression and classification tasks
- It find a optimal line/hyperplane that separates classes with the largest possible distance
- This algorithm can be used in N-dimensional space

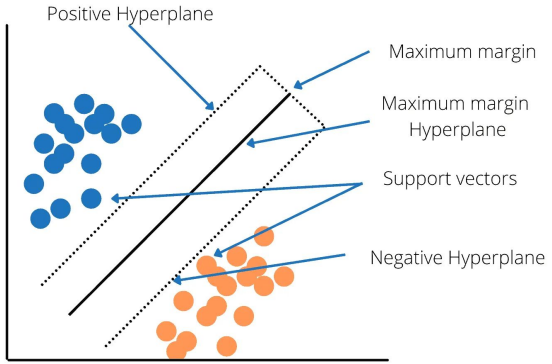
$y = w^T x + w_0$ w is vector of f weights and x is a vector of f features

Problem formulation: We want a discriminant such that the projection of the classes onto the orthogonal to the discriminant is the most separable and the variance is small

$$J(w) = \frac{(m_1 - m_2)^2}{\sigma_1^2 + \sigma_2^2}$$

where m1 and m2 is the projection mean and σ is the variance of the class projection.

Then we can derive the equation for w for the learning algorithm



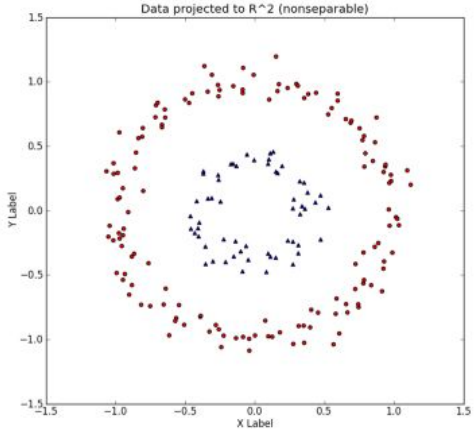
Soft Margin SVM

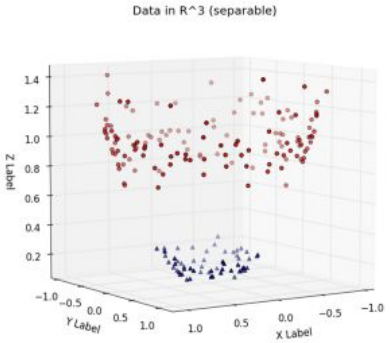
Is it always possible to separate two classes?
No. In practice we use soft margin SVM which allows some misclassified.

Wouldn't this strict separation allow high influence of outliers and overfitting?
Yes. That's why we introduce the hyperparameter C. A lower C value creates a larger margin that allows for misclassification. As C approaches infinity, it is a hard margin svm that does not allow for misclassification. It places higher penalty in the objective function for misclassification or a point being in the margin.

Support Vector Machine

- But our linear discriminant does not work for these cases
- SVM allow different kernels to be used which would map the input data into a higher dimensional space to more easily create a hyperplane that separates the data
- A kernel is another hyperparameter that can be chosen when creating the model
- Kernel Functions:
  - Linear: Data is linearly separable
  - Polynomial: Data is nonlinear or has interaction between features
  - RBF: Data has clusters or complex nonlinear pattern
  - Sigmoid: If the data appears like logistic function





Hyperparameter Tuning

- We have introduced three different parameters. Lambda for regularization, c for SVM, and kernel for SVM
- But how do we know the best value for the model?
- Manual search:
  - Manually adjust the hyperparameter, train, evaluate, adjust
  - Time consuming and inefficient approach
- Grid Search:
  - Train the model using all possible combination of the hyperparameters and then select the combination with the best performance
  - Can be time consuming
- One Parameter at a time:
  - Iterate through one parameter and select the best value before trying the next parameter

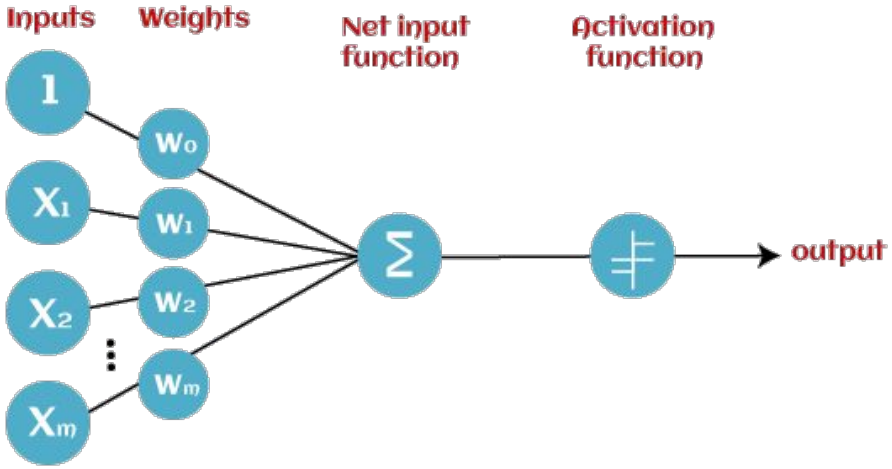
Perceptron

- Supervised Learning algorithm for classification and regression
- It is similar to SVM where it creates a discriminant that best separates classes
- This is the basic neuron unit in a neural networking
- It is different to SVM as it uses a activation function where the input is the output of the mapper function
- Perceptions are used to implement linearly separable functions

$$y = w^T x + w_0$$

$$t = \phi(y(x))$$

$$J(w) = \sum_{n \in M} t_n * y(x_n)$$



Perception Learning Algorithm

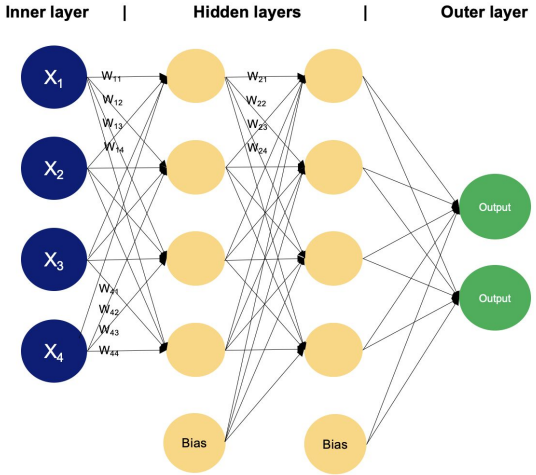
- Previously, our learning algorithms would use the derived formula once to computer w
- Perceptron uses iterations to update the weights
- Initially, assign the weights a random value
- We can use a online update where we add a new instance every iteration and then update the weights
- Gradient descent of the objective function is used to derive the value of the weights

Algorithm: Perceptron Learning Algorithm

$P \leftarrow$ *inputs with label* 1;
$N \leftarrow$ *inputs with label* 0;
Initialize **w** randomly;
while !*convergence* **do**
  Pick random $\mathbf{x} \in P \cup N$;
  if $\mathbf{x} \in P$ *and* $\mathbf{w}.\mathbf{x} < 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$;
  end
  if $\mathbf{x} \in N$ *and* $\mathbf{w}.\mathbf{x} \geq 0$ **then**
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$;
  end
end
//the algorithm converges when all the inputs are classified correctly

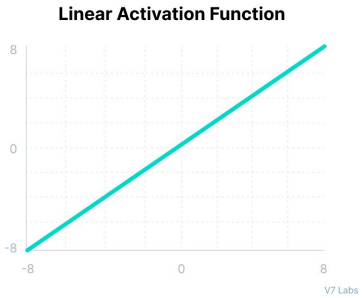
Multi-Layer Perceptron

- In our previous example, we had a issue. A perceptron would only create a linear discriminant. But we saw from our SVM example a class that was in the shape of a ring.
- The solution is to use multiple perceptrons so we have multiple discriminants
- Based on the values from each of the perceptron we can classify the data
- We use a objective function at the outer layer to determine how well our model does
    - The error from the objective function is **back propagated** to each neuron to update the weights after each iteration
- Use a similar learning algorithm for a single perceptron

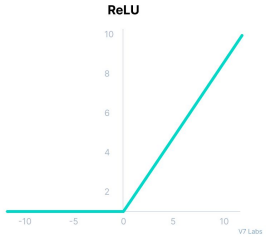


Perceptron Activation Function

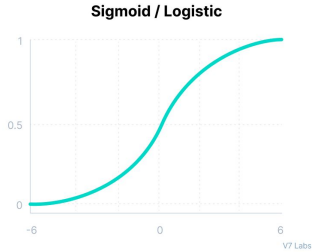
Linear Activation Function



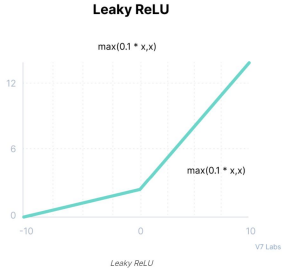
ReLU



Sigmoid Function



Sigmoid Function

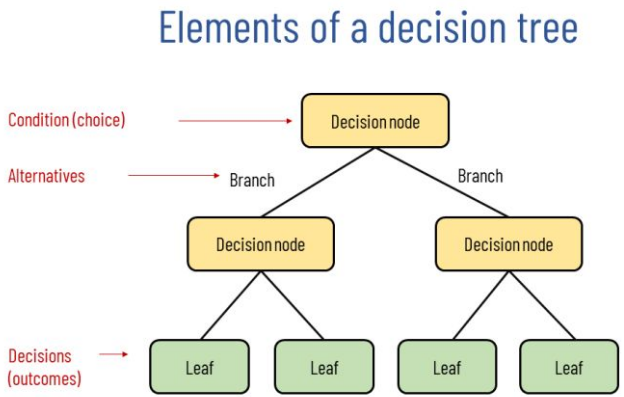


This function is used for binary classification.
Softmax is used for multiple classes

https://www.v7labs.com/blog/neural-networks-activation-functions

Random Forests

- Supervised learning model for classification and regression
- Random Forest model uses multiple decision trees
- A decision tree is a tree that branches to leaf nodes based on certain conditions
- Random forest uses ensemble learning method where it combines the outputs of a set of classifiers (in this case decision trees) to improve results
- Random Forest uses ensemble learning and feature randomness. It only considers a subset of the features



Hyperparameters:

- Node Size
- Number of trees
- Number of features sampled

XGBoost

- Extreme Gradient Boosting
- Supervised Learning Model
- It uses boosted trees which are similar to random forests. It uses tree ensembles where different classifiers (decision trees) are used to improve the results
- Objective function for tree boosting:

$$\text{obj} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \omega(f_i)$$

- Boosting is where multiple weak learners (decision trees) are built sequentially so that each tree improves the previous ones by fixing the misclassified examples
- Uses iterative learning rate
- Allows for parallelization and the use of GPUs

XGBoost

- Similarly to Perceptrons where we must choose the right activation function given our data, we must choose the right objective function

Binary Logistic Function - Used for classification of two classes

Multi Softmax Function - Used for classification of more than two classes. It returns the numerical value for the class

Multi Softprob Function - Use for classification of more than two classes. It returns the probability of belonging to each of the classes. Select the class with the highest probability

Thanks

Do you have any questions?
ufstatsclub@gmail.com
Link Tree