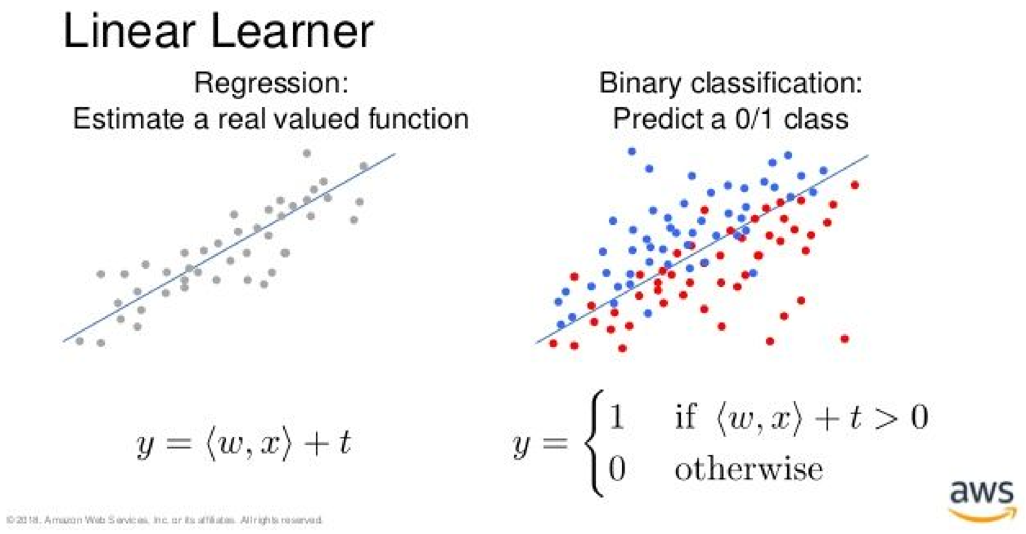
FebaTech

Domain 3 Part 2

Linear Learner



You must be familiar with linear regression in machine learning, it somewhat does a similar thing.

- Can handle both regression (numeric) predictions and classification predictions, by using something called linear threshold function.
- Can do binary or multi-class in case of classification.
- It can do anything as long as the line will fit what you are trying to do.

What training input can it take?

- RecordIO-wrapped protobuf (i.e float32 data)
- CSV format but the first column is assumed to be a label.
- It can take data either in File (takes complete dataset at once) or Pipe mode (takes data when required from S3 bucket)

There are three steps involved in the implementation of the linear learner algorithm: preprocess, train, and validate.

Step 1: Preprocess

- Training data must be normalised before giving to Linear Learner
- You can either do it manually or Linear Learner does this for us automatically
- Input data must be shuffled.

Step 2: Train

- Linear Learner works on stochastic gradient descent under the hood.
- But we can choose an optimization algorithm like Adam, AdaGRad, SGD etc.
- You also specify their hyperparameters, such as momentum, learning rate, and the learning rate schedule.

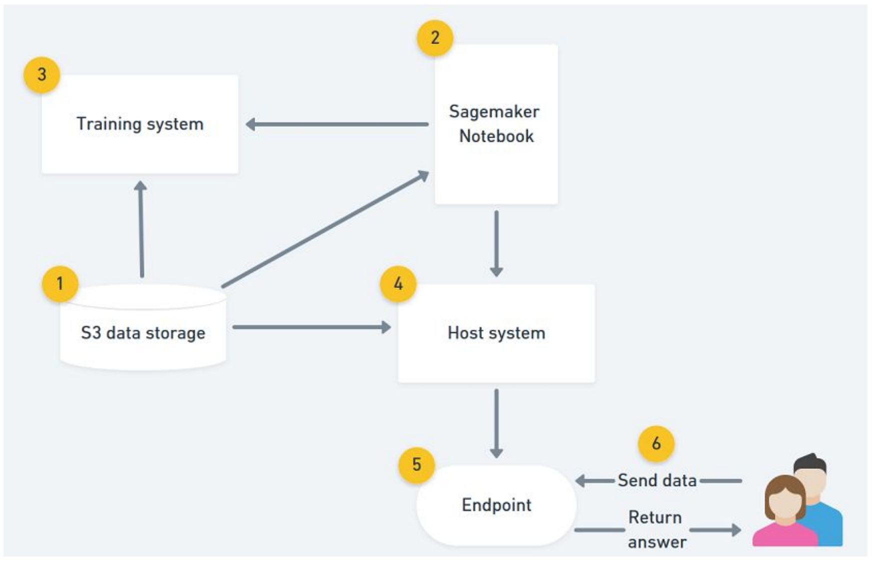
Step 3: Validate and set the threshold

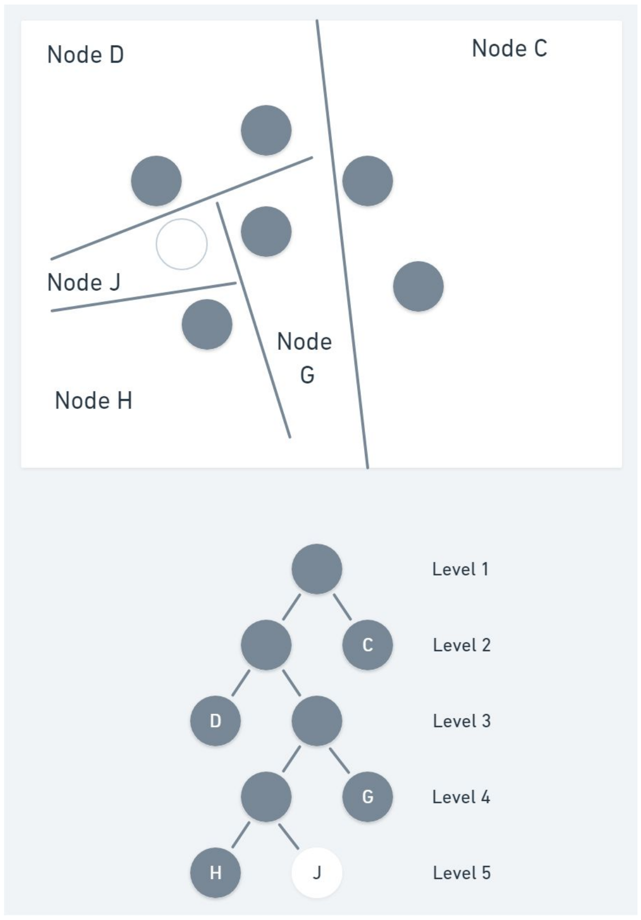
- When training multiple models in parallel, the models are evaluated against a validation set to select the most optimal model once training is complete.

Important Hyperparameters :

- Balance multi-class weights (give equal importance to different classes in loss function)
- L1 and L2 regularization (also called weight decay)
- Learning rate and mini batch size

Random Cut Forest





- It is an algorithm for anomaly detection
- Unsupervised
- The input data is not a well structured or patterned data

ex : anomalies in time series data, breaks in periodicity, or unclassifiable data points.
- With each data point, RCF associates an anomaly score. Low score values indicate that the data point is considered "normal." High values indicate the presence of an anomaly in the data.

The definitions of "low" and "high" depend on the application but common practice suggests that scores beyond three standard deviations from the mean score are considered anomalous.

What training input does it expect?

- RecordIO-protobuf or CSV
- Can use File or Pipe mode
- Optional test channel for computing accuracy, precision, recall and F1 on labeled data

Important Hyperparameters

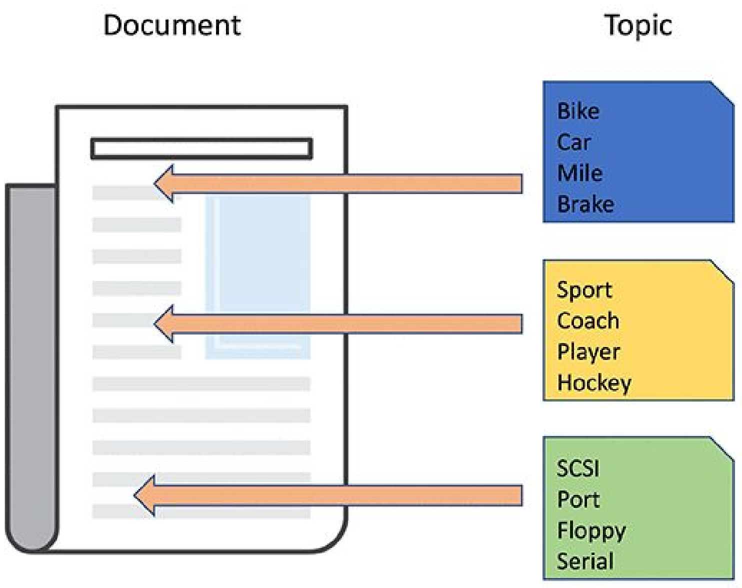
- Num_trees (increasing reduces noise)
- Num_samples_per_tree (should be chosen such that 1/num_samples_per_tree approximates the ratio of anomalous to normal data)

Instance Recommendations for the RCF Algorithm

For training, we recommend the ml.m4, ml.c4, and ml.c5 instance families. For inference we recommend using a ml.c5.xl instance type in particular, for maximum performance as well as minimized cost per hour of usage.

- GPU usage not required.

Neural Topic Model



- This algorithm is used to classify documents into various topics or provide a summary for a given document.
- The classified topics may not make sense to humans.
- It is more than just calculating term frequency–inverse document frequency
  For example 'stars', 'sun', 'clouds', 'planets' might be classified as space but the algorithm dont know what to call it. So it groups the documents just into one.
- It uses "Neural Variational Inference" algorithm behind.

Input/Output Interface for the NTM Algorithm

- Supports four data channels: train and validation, test, auxiliary (these three are optional)
- RecordIO-protobuf or CSV
- Words must be tokenized into integers
- Every document must contain a count for every word in the vocabulary in CSV
- The "auxiliary" channel is for the vocabulary
- File or Pipe mode

Latent Dirichlet Allocation (LDA) in sagemaker

- This is another topic modeling algorithm without deep learning involvement
- Unsupervised

- This algorithm is more general than neural topic model as it could be used not only for grouping documents but also to cluster customers based on purchases or harmonic analysis in music.

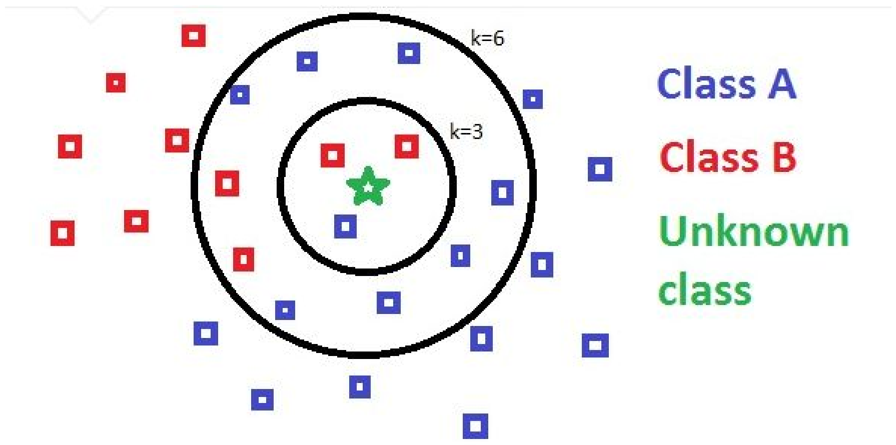
What training input does it expect?

- Has a training channel and optional test channel to check for accuracy
- RecordIO-protobuf or CSV
- Each document has counts for every word in vocabulary (in CSV format)
- Pipe mode only supported with recordIO
- This algorithm is similar to a neural topic model, but CPU based. Therefore maybe cheaper/ more efficient.

Important Hyperparameters :

- Num_topics : we can decide how many topics we want in output/
- Alpha0
    a) Initial guess for concentration parameter
    b) Smaller values generate sparse topic mixtures
    c) Larger values (> 1.0) produce uniform mixtures.

KNN



- Works by finding the K nearest data points to the new example, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).
- For classification KNN returns the most frequent label, for regression it returns the average value.
- Training with the k-NN algorithm has three steps: sampling, dimension reduction, and index building.

Input/Output Interface for the k-NN Algorithm:

- You can give the train channel which contains your data, the test channel emits accuracy or MSE.
- For training we can take recordIO-protobuf or CSV (for CSV the first column will be label)
- Can use File or Pipe mode

How is it Used?

- First it samples the data assuming that you have large data to work with.
- Sagemaker includes a dimensionality reduction step as well. So if it has a lot of features, it will reduce it to avoid the curse of dimensionality. So we can compute the nearest neighbours.
- It has “sign” or “fjlt” options to do this.
- Build an index for looking up for neighbours and then serialize the model, so then you query the model for K.

Important Hyperparameters

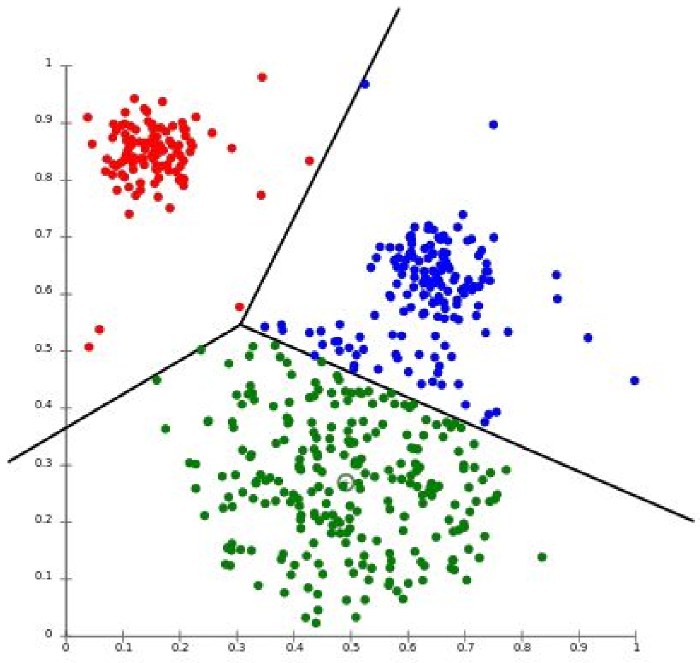
- How many neighbours I should look at. We need to experiment for which value of K we can get better results.

Instance types:

- can train on CPU or GPU

    a) MI.m5.2xlarge
    b) MI.p2.xlarge
- Inference
    a) CPU for lower latency
    b) GPU for higher throughput on large batches

K-Means Clustering In Sagemaker



- It's an Unsupervised clustering technique.
- It's important to remember that KNN is supervised as we are learning from labels of other data points.
- Divide the data into K groups (that's why K-Means) where each member of the group is as similar as possible to each other.

- Othen we use euclidean distance to measure the distance between two points.
- Sagemaker brings Web-scale K-Means clustering.

Training Inputs:

- Training channel to analyze your data, but as it is unsupervised the test channel is optional.
- RecordIO-protobuf or CSV
- File or Pipe on either.