8 Friday
AUGUST

KNN

→ KNN —— K Nearest Neighbor.

→ As the name suggests it considers K Nearest Neighbors (Data points) to predict the class or continous value for the new Datapoint

→ In this, we donot learn the weights from training data to predict output but use entire training instances to predict output.

→ The distance metric and K value are two important considerations while using the KNN algorithm.

→ Euclidean distance is most popular. You can also use Hamming distance, Manhattan distance, Minkowski distance as per your need.

→ For using KNN in classification:
We have to check the K-nearest neighbours and predict value for the test case.
For using KNN in regression:
We have to find the mean / median of the K-nearest neighbours and predict the value for test case

- For choosing k, we could draw error curves and pick the best one
- For classification (binary), k has to be odd
- Data pre-processing is required.
- Code for this is as follows:

1) Find optimal K from error curves.
2) Calculate distance b/w (X, X_i) from $i = 1, 2 \ldots n$
3) Sort them in ascending order
4) Take the first k values
5) Find out which class has most frequency That is predicted class.

10 Sunday
AUGUST

K-Means

→ K-means clustering is one of the simplest unsupervised ML algorithm.

→ In this, we define a number K, which is the number of centroids i.e the center of the cluster.

→ The 'means' in k-means refers to averaging of the data; i.e finding the centroid.

→ The prediction is made based on these clusters, i.e point is allocated to cluster through reducing the in-cluster sum of squares.

→ First a group of random points are taken as centroids from each cluster, then iterative calculations are done to optimize the position of centroids.

ALGORITHM:

Randomly initialize K centroids $\mu_1, \mu_2 \dots \mu_K \in R^n$

Repeat {

Cluster assignment

for i = 1 to m

$c^{(i)}$:= index (from 1 to K) of cluster centroid close to $x^{(i)}$

2008

Moving centroid:

for $k = 1$ to K

$\mu_k :=$ average of points assigned to cluster k

- $\mu_{c^{(i)}} =$ cluster centroid of cluster to which example $x^{(i)}$ has been assigned

OPTIMIZATION:

$$J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots \mu_K) = \frac{1}{m} \sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||^2$$

$$\min_{\substack{c^{(1)}, \ldots, c^{(m)}, \\ \mu_1, \ldots, \mu_K}} J(c^{(1)}, \ldots c^{(m)}, \mu_1, \ldots \mu_K).$$

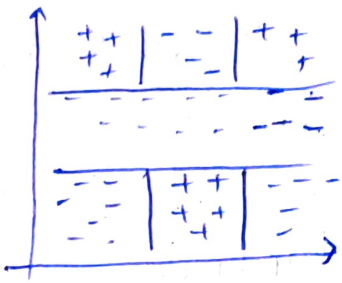
DECISION TREES

→ Decision trees is gonna follow Greedy, Top-Down, Recursive partitioning.

→ While partitioning a region R_p. we are looking for a split S_p. as

$$S_p(j,t) = \Big(\{x \mid x_j < t, x \in R_p\}, \{x \mid x_j \geq t, x \in R_p\} \Big)$$

(R_1 = first set, R_2 = second set)



→ How to choose splits?

Define $L(R)$: Loss on R.

For Given C classes, define

$\hat{P}_c$ to be the proportion of examples in R that are of class C.

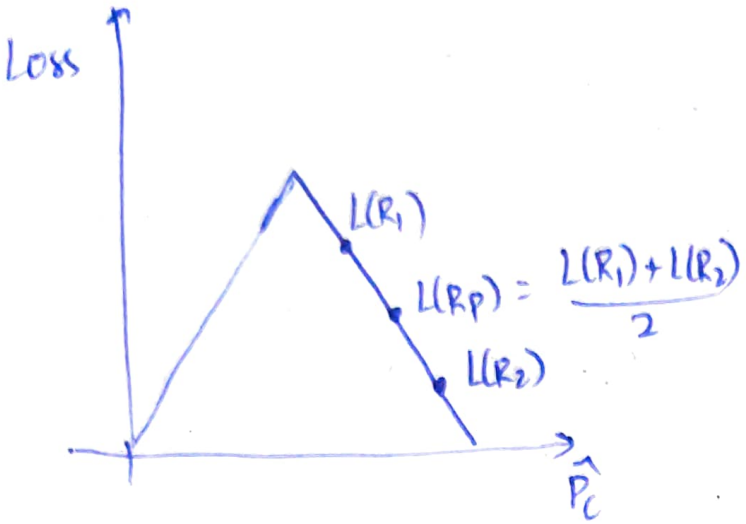
$$L_{misclass} = 1 - \max_c \hat{P}_c$$

We have to

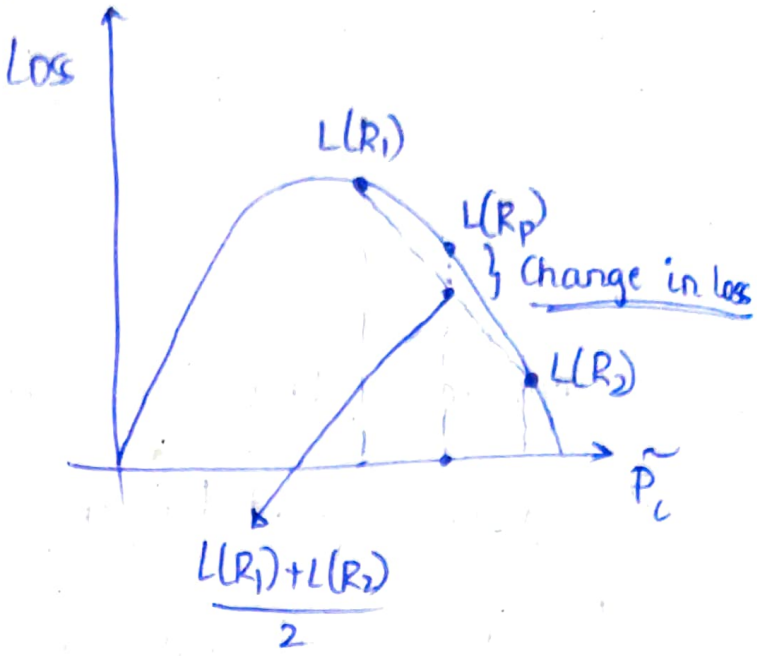
$$\max_{j,t} \underbrace{L(R_p)}_{\text{Parent loss}} - \underbrace{\big(L(R_1) + L(R_2)\big)}_{\text{Children loss}}$$

* Misclassification loss has issues, so we define Cross entropy loss.

$$L_{cross} = -\sum_c \hat{P}_c \log_2 \hat{P}_c$$



Misclassification



Cross-entropy

Run Time:

For n examples, f features and depth d of DT

Test time:

$O(d)$

Train Time

Each point is part of $O(d)$ nodes,

Cost of point at each node is $O(f)$

Total cost = $O(nfd)$.

RANDOM FOREST

→ Random forest consists of a large number of individual decision trees that operate as an ensemble

Ensembling:

Take x_i's which are Random variables that are independent identically distributed (iid)

$Var(x_i) = \sigma^2$ then $Var(\bar{x}) = Var\left(\frac{1}{n}\sum_i x_i\right) = \frac{\sigma^2}{n}$

If they are not independent and have a correlation constant p

then $Var(\bar{x}) = p\sigma^2 + \frac{(1-p)}{n}\sigma^2$.

Ways to Ensemble

- → Different algorithms
- → Different training sets
- → Bagging → (Bagging + DT ⇒ Random Forest)
- → Boosting

Ideal for bagging as have high var & low bias

Bagging – Bootstrap Aggregation

True population P.

and Training set $S \sim P$

Assume $P = S$ Bootstrap samples $Z \sim S$.

So, for

Bootstrap samples $Z_1, \dots Z_m$

Train Model G_m on Z_m

$$G(m) = \frac{\sum_{i=1}^{M} G_i(x)}{M}$$

→ For Random Forests:

At each split, consider only a fraction of your total features.