Ensemble Methods

Types of Methods:

1. Bagging & Boosting
   - → manipulate data distribution (resampling)
2. Random Forest: randomly select feature subsets & built decision trees
   - → manipulate input features
3. Randomly partition classes into two subsets, treat as +ve & −ve, & learn a binary classifier (do many times)
   - → manipulate class labels
4. Using different models

Bagging (Bootstrap Aggregating):

- → given a set of n training samples, create k samples by drawing at random w/ replacement (called boostrapping)
- → build a classifier on each bootstrap sample
- → use majority voting
- → reduces variance and increases prediction accuracy
- → helps when learner is unstable (small change in training set → large change in output classifier)
- → can degrade results for stable learners

Random Forests:

- → bagging for decision trees that decorrelates the trees
- → considers only $k \approx \sqrt{m}$ attributes at each split
- → on average, $(m-k)/m$ of the splits will not consider the strong predictor

Bagging & Decision Trees:

- → Create 100s - 100s deep trees
- → final class is majority vote

Boosting:

- → take collection of weak classifiers & turn them into a strong one
- → family of methods

Training:

- → produce a sequence of classifiers (the same base learner)
- → each classifier depends on the previous one and focuses on the errors
- → examples previously incorrectly predicted recieve higher weights

Testing:

- → for test case, results of the series of classifiers are combined to determine final class of test case.

- → records wrongly classified → weights increased
- → records correctly classified → weights decreased

weight increased means it is more likely to be chosen again in the following rounds

Ada Boost:

weighted training set — non negative weights sum to 1

(x_1, y_1, w_1)
(x_2, y_2, w_2)
$\vdots$
(x_n, y_n, w_n)

→ Build a classifier h_t whose accuracy on training set $> 1/2$ (better than random)

change weights

- → start w/ constant prediction w/ new learner added each time

$$y_t^{(i)} = \sum_{k=1}^{t} \phi_k(x^{(i)}) = y_{t-1}^{(i)} + \phi_t(x^{(i)})$$

prediction at round t — prediction at previous round — new model

- → boosting requires base learner to be unstable
- → susceptible to noise, outliers can hurt performance
- → large number of models can lead to overfitting

XGBoost:

- → extension of Ada Boost
- → greedily builds a tree of weak learners
- → gain based on 1st & 2nd order derivatives of loss functions and child nodes based on split
- → new model added to old w/ a shrinkage factor

Classification & Regression Tree (CART)

- → decision rules similar to decision tree
- → leaf nodes contain weighted prediction
- → predict sample as the sum of scores predicted by each tree

Clustering:

KMeans Clustering:

- → given training set $\{x^{(1)}, \ldots, x^{(n)}\}$, group into a few cohesive "clusters"
- → initialize cluster centroids $\mu_1, \mu_2, \ldots, \mu_n \in \mathbb{R}^m$ randomly
- → Repeat until convergence:

  For every i, set $c^{(i)} := \arg\min_j \| x^i - \mu_j \|^2$

  For every j, set $\mu_j := \dfrac{\sum_{i=1}^{n} 1\{c^i = j\} x^i}{\sum_{i=1}^{n} 1\{c^i = j\}}$

probability of getting initial centroid in each cluster is $K!/K^K$

solutions:

- → multiple runs
- → sample & use heirchal clustering to determine initial centroids
- → bisecting KMeans

Cost function: $J(c, \upsilon) = \sum_{i=1}^{n} \| x^i - \mu_{c(i)} \|^2$

Elbow method: plot the cost function J for different K values & stop when decrease in error is small

problems:

- → clusters of different sizes, densities, shapes
- → outliers

Solutions: using many clusters and putting some back together

Mixture of Gaussians Model

- → given training set $\{x^{(1)}, \ldots, x^{(n)}\}$
- → assume the points were generated by randomly choosing $z^{(i)}$ from $\{1, \ldots, k\}$ then randomly generating $x^{(i)}$ from corresponding gaussian distribution, one of k guassians associated w/ the $z^{(i)}$'s

Density-Based Clustering (DBSCAN):

- → the density is the number of points within specified radius (Eps)
- → point p is a core point if it has at least specified number of points (MinPts) within Eps (points interior of a cluster)
- → a border point has fewer than MinPts within Eps, but is in neighborhood of core point
- → noise point is any point that is not core or border

does not work well under:

- → varying densities
- → high dimensional data

Cluster validity:

why?

- → to avoid finding patterns in noise, compare clustering algs, compare 2 sets of clusters, compare two clusters

Cluster validity via correlation:

two matrices:

- → proximity or distance matrix of data
- → ideal proximity matrix implied by clustering solution
  - → 1 row, 1 column for each data point
  - → 1 if belong to same cluster, 0 if not

compute correlation between the two

high correlation means points that belong to same cluster are close to each other

External measures:

entropy: For cluster j, compute $p_{ij} = \dfrac{m_{ij}}{m_j}$ probability member of class i belongs to cluster j

m_j = # elements in cluster j

m_{ij} = # elements of class i in cluster j

$e_j = \sum_{i=1}^{L} p_{ij} \log_2 p_{ij}$ where L = # of classes

total entropy: $e = \sum_{i=1}^{K} \frac{m_j}{m} e_j$, m_j = size of cluster

m = total data points

K = # clusters

purity: for 1 cluster $\text{purity}_j = \max p_{ij}$

total purity: $\sum_{i=1}^{K} \frac{m_i}{m} \text{purity}_j$

Internal measures:

cluster cohesion: how closely related objects in cluster are (ex: SSE)

cluster separation: how distinct or well-separated a cluster is from other clusters

silhouette coefficient:

a = avg distance of i to the points in its cluster

b = minimum(avg dist of i to points in another cluster)

$s = (b - a) / \max(a, b)$ → closer to 1 the better