CS 484 FINAL EXAM REVIEW

NATHANIEL BECHHOFER

1. Types of Attributes

- **Categorical (Qualitative) attributes**
  - **Nominal attributes**: Only naming. Examples include zip codes, gender
  - **Ordinal attributes**: Also includes order. Examples include year of birth, street numbers
- **Numeric (Quantitative) attributes**
  - **Interval attributes**: Meaningful units which allow comparison of differences between values. Examples include dates, temperature
  - **Ratio attributes**: Also includes meaningfulness of ratios. Examples include income, height, age

- **Discrete attributes**: Can be categorical or numeric (e.g. counts), but has a finite (or countably infinite, which means corresponding to any integer) number of possible values. Examples include number of children, marital status, gender
  - Special case is **binary attrubutes**, which can only take on two values such as yes/no or 0/1
- **Continuous attributes**: Real numbers, which imply a potentially infinite number of possible values. Examples include height, weight, temperature

2. Noise vs Outliers

Noise is anything that is not the true data, also defined as the random component of a measurement error. It may have values close to the true data. An outlier is something that is much different than the other values with regards to either its characteristics or attribute values. **Anomalies** are typically defined as data points that don't fit a data model well.

3. Measures of Similarity or Dissimilarity

3.1. Euclidean Distance.

Euclidean Distance. The **Euclidean distance** between two points x and y is given as:

$$\sqrt{\sum_{k=1}^{n}(x_k - y_k)^2}$$

where n is the number of dimensions and x_k and y_k are the k^{th} attributes of x and y.

3.2. **Hamming Distance.** The **Hamming distance** is the number of bits that are different between two objects that have only binary attributes. (Manhattan distance is Euclidean distance without the squared and square root.)

3.3. **Binary Similarity Measures.** If we have two objects x and y that consist of n binary attributes, we can denote counts f_{ij} which measure how many attributes measure 0 or 1 for the x and y objects.

This gives us a way to write the formula for the **simple matching coefficient**:

$$SMC = \frac{\text{number of matching attribute values}}{\text{number of attributes}} = \frac{f_{11} + f_{00}}{f_{01} + f_{10} + f_{11} + f_{00}}$$

If binary attributes are asymmetric, one can use the **Jaccard coefficient**:

$$J = \frac{\text{number of matching presences}}{\text{number of attributes not involved in 00 matches}} = \frac{f_{11}}{f_{01} + f_{10} + f_{11}}$$

3.4. **Cosine Similarity.** For non-binary vectors, we can do something similar to the Jaccard with the **cosine similarity**:

$$\cos(x, y) = \frac{x \cdot y}{\|x\|\|y\|} = \frac{\sum\limits_{i=1}^{n} x_i y_i}{\sqrt{\sum\limits_{i=1}^{n} x_i^2}\sqrt{\sum\limits_{i=1}^{n} y_i^2}}$$

3.5. **Correlation.** We can use **Pearson's correlation coefficient** (often denoted using ρ) to measure the linear relationship between attributes of many types of objects.

$$\rho_{X,Y} = \frac{\text{cov}(X, Y)}{s_X s_Y}$$

4. Impurity Measures

4.1. **Multi-way split.** : Use as many partitions as distinct values, so we may get more than two nodes for a parent node

4.2. **Gini.** The Gini index is defined as $1 - \sum_{i=0}^{c-1} [p(i|t)]^2$ where c is the number of classes and $p(i|t)$ denotes the fraction of records belonging to class i at a given node t. For multiway split, we calculate using the weighted average for each category.

4.3. **Entropy.** Entropy is defined as $-\sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t)$ where c is the number of classes and $p(i|t)$ denotes the fraction of records belonging to class i at a given node t. (We consider $0 \log_2 0 = 0$ for entropy calculations.)

4.4. **Classification Error.** Clearly just $1 - \max_i [p(i|t)]$

4.5. **Gain.** The gain can be found using a formula:

$$I(parent) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

where $I(.)$ is the impurity measure of a given node, N is the number of records at the parent node, k is the number of attribute values, and $N(v_j)$ is the number of records associated with the child node v_j. Gain is called **information gain** when using entropy as the impurity measure.

5. Conditional probability

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} = \frac{P(B \mid A)\, P(A)}{P(B \mid A)P(A) + P(B \mid \neg A)P(\neg A)}$$

6. Naive Bayes

6.1. **Conditional independence.** Naive Bayes only works with conditional independence. X is conditionally independent of Y given Z if $P(X|Y,Z) = P(X|Z)$.

6.2. **Classification.** To choose how to classify a record, one merely finds the class that maximizes $P(Y) \prod_{i=1}^{d} P(X_i|Y)$.

6.3. **M-estimate.** If the classifier gets zeros, there may be an issue with categorization. An alternative uses the following formula:

$$P(x_i|y_j) = \frac{n_c + mp}{n + m}$$

where n is the total number of instances from class y_j, n_c is the number of training examples from y_j that take on the value x_i, m is a parameter known as the equivalent sample size, and p is a user-specified parameter.

7. Evaluation

K-fold cross-validation, where k is a user-specified number, usually 5 or 10. When performing five-fold cross-validation, the data is first partitioned into five parts of (approximately) equal size, called folds. Next, a sequence of models is trained. The first model is trained using the first fold as the test set, and the remaining folds (2?5) are used as the training set. The model is built using the data in folds 2?5, and then the accuracy is evaluated on fold 1. Then another model is built, this time using fold 2 as the test set and the data in folds 1, 3, 4, and 5 as the training set. This process is repeated using folds 3, 4, and 5 as test sets. For each of these five splits of the data into training and test sets, we compute the accuracy. In the end, we have collected five accuracy values.

Accuracy is the number of correct predictions (TP and TN) divided by the number of all samples

Precision measures how many of the samples predicted as positive are actually positive

Recall measures how many of the positive samples are captured by the positive predictions

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

8. Association rules

8.1. Rule Evaluation Metrics.

Support (s) is the proportion of transactions that contain both X and Y

Confidence (c) Measures how often items in Y appear in transactions that contain X

The difference between these two is just whether the denominator is all transactions or all transactions that contain X

8.2. Apriori principle.

If an itemset is frequent, then all of its subsets must also be frequent

8.3. Candidate generation.

One prominent way to generate candidates is the $F_{k-1} \times F_1$ merging strategy. This consists of extending each frequent itemset of size $k-1$ with other frequent items (where frequent refers to items that satisfy the specified minimum support threshold).

8.4. Apriori algorithm.

Apriori generates candidates and tests them on the fly. It starts by looking for the frequent itemsets of size 1, then keeps generating new candidate k-itemsets using the frequent $(k-1)$-itemsets found in the previous iteration and checking their frequency until no new frequent itemsets are generated. More specifically, the apriori principle is used; instead of checking the support of all the larger itemsets, we can

9. Rule-Based Classifiers

9.1. **Rule Coverage and Accuracy.** **Coverage** of a rule: Fraction of records that satisfy the antecedent of a rule

Accuracy of a rule: Fraction of records that satisfy both the antecedent and consequent of a rule

9.2. **Mutually exclusive rules.** Classifier contains mutually exclusive rules if the rules are independent of each other; every record is covered by at most one rule

9.3. **Exhaustive rules.** Classifier has exhaustive coverage if it accounts for every possible combination of attribute values; each record is covered by at least one rule

9.4. **Nearest-Neighbor Classifiers.** Requires:

- The set of stored records
- Distance Metric to compute distance between records
- The value of k, the number of nearest neighbors to retrieve

To classify an unknown record:

- Compute distance to other training records
- Identify k nearest neighbors
- Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote, sometimes weighting according to $1/d^2$)

9.5. **Bagging.** Bootstrap by sampling with replacement and build classifier on each bootstrap sample

9.6. **Boosting.** Initially, all N records are assigned equal weights. Records that are wrongly classified will have their weights increased, while records that are classified correctly will have their weights decreased

10. Cluster Analysis

10.1. **Vector quantization.** Uses prototype vectors for clusters and each data point is given an index denoting its position in the table of prototype vectors.

10.2. **Center-based Cluster.** A cluster is a set of objects such that an object in a cluster is closer (more similar) to the center of a cluster, than to the center of any other cluster

These are often centroids, the average of all the points in the cluster, or a medoid, the most "representative" point of a cluster

K-means Clustering is a partitional clustering approach where each cluster is associated with a centroid (center point) and each point is assigned to the cluster with the closest centroid. The Number of clusters, K, must be specified. The basic algorithm is very simple: just pick some centroids at random, assign points to the closest centroid, recompute the centroids, and keep going until the centroids don't change.

The Voronoi diagram for a set of K points in the plane is a partition of all the points of the plane into K regions, such that every point (of the plane) is assigned to the closest point among the K specified points. If we have K K-means clusters, then the plane is divided into K Voronoi regions that represent the points closest to each centroid.

10.3. **Contiguous Cluster.** A cluster is a set of points such that a point in a cluster is closer (or more similar) to one or more other points in the cluster than to any point not in the cluster.

10.4. **Density-based Cluster.** A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density. This is useful when the clusters are irregular or intertwined, and when noise and outliers are present.

DBSCAN is a density-based algorithm. Density = number of points within a specified radius (Eps)

- A point is a core point if it has more than a specified number of points (MinPts) within Eps
- A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point
- A noise point is any point that is not a core point or a border point.

10.5. **Hierarchical Clustering.** Produces a set of nested clusters organized as a hierarchical tree that can be visualized as a dendrogram, a tree like diagram that records the sequences of merges or splits

For **single-link**, use proximity between closest two points that are in different clusters

For **complete-link**, use proximity between furthest two points that are in different clusters

10.6. **Dimensionality Reduction.** : PCA works to find a projection (some set of linearly independent vectors) that captures the largest amount of variation in data.

10.7. **Minimum Description Length (MDL).** We seek to minimize how much information must be transmitted to describe a whole data set. We can send only attribute values x and a classification rule along with info about where the rule fails to make this work. Thus we find the classification rule which minimizes this description length

A random forest is essentially a collection of decision trees, where each tree is slightly different from the others. The idea behind random forests is that each tree might do a relatively good job of predicting, but will likely overfit on part of the data. If we build many trees, all of which work well and overfit in different ways, we can reduce the amount of overfitting by averaging their results. Next, a decision tree is built based on this newly created dataset. However, the algorithm we described for the decision tree is slightly modified. Instead of looking for the best test for each node, in each node the algorithm randomly selects a subset of the features, and it looks for the best possible test involving one of these features. The number of features that are selected is controlled by the parameter. This selection of a subset of features is repeated separately in each node, so that each node in a tree can make a decision using a different subset of the features.

To build a tree, we first take what is called a bootstrap sample of our data. That is, from our n data points, we repeatedly draw an example randomly with replacement (meaning the same sample can be picked multiple times), n times. This will create a dataset that is as big as the original dataset, but some data points will be missing from it (approximately one third), and some will be repeated.

Multilayer perceptrons (MLPs) are also known as (vanilla) feed-forward neural networks, or sometimes just neural networks. The neural network model MLPs can be viewed as generalizations of linear models that perform multiple stages of processing to come to a decision.

In an MLP this process of computing weighted sums is repeated multiple times, first computing hidden units that represent an intermediate processing step, which are again combined using weighted sums to yield the final result

This model has a lot more coefficients (also called weights) to learn: there is one between every input and every hidden unit (which make up the hidden layer), and one between every unit in the hidden layer and the output. After computing a weighted sum for each hidden unit, a nonlinear function is applied to the result. The result of this function is then used in the weighted sum that computes the output