AHMEDABAD UNIVERSITY

School of Engineering and Applied Science
Ahmedabad University
End term examination – Spring 2016

Course Title: Advanced Data Structures and Algorithms
Date of Examination: 9.6.2014
Instructor's name: Dr. Ratnik Gandhi

Roll No: 1401117
Name: Kishan R. Raval
Semester: 5th
Total Marks: 40

**Q.a) List down a set of learning algorithms in practice.**

Machine learning is a field of computer science, probability theory, and optimization theory which allows complex tasks to be solved for which a logical/procedural approach would not be possible or feasible.

There are several different categories of machine learning, including (but not limited to):

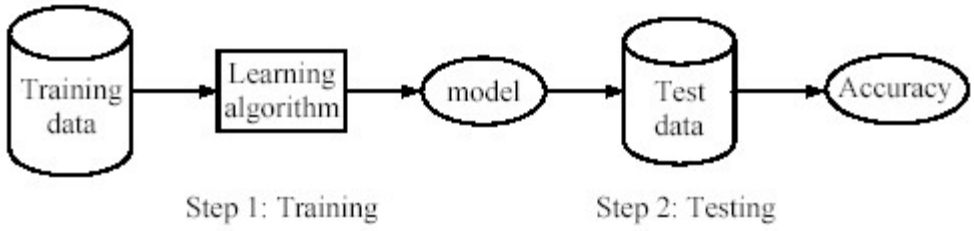
- **Supervised learning**

  In supervised learning, we are given a data set and already know what our correct output should look like, having the idea that there is a relationship between the input and the output.

  It is a machine learning task that makes it possible for our phone to recognize our voice, email to filter spam and many other applications.

  Algorithms under this set are:

  - Decision Trees
  - Regression and Classification
  - Neural Networks
  - Instance-Based Learning
  - Ensemble learning
  - Kernel Methods and Support Vector Machines (SVM)s
  - VC Dimensions
  - Bayesian Learning and Bayesian Inference



- **Unsupervised Learning**

  Unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables.

  We can derive this structure by clustering the data based on relationships among the variables in the data. We do not have any feedback system in algorithms of unsupervised learning. Prediction done by Netflix and algorithms that how Amazon knows what we want to buy before we do are part of unsupervised learning.

  Algorithms under this set are:

  - Randomized optimization
  - Clustering
  - Feature Selection and Transformation
  - Information Theory

- **Reinforcement learning**

  Reinforcement learning is part of Machine Learning as well as Artificial Intelligence. It allows machines and software agents to automatically determine the ideal behavior within a specific context, in order to maximize its performance. Feedback mechanism is used as a part of learning for machine.

  Algorithms under this set are:

  - Markov Decision Processes
  - Reinforcement Learning
  - Game Theory

**Q.b) Identify set of learning algorithms based on optimization techniques and describe them.**

Goal of optimization technique is to minimize the test-error. Kernel methods are a class of classification techniques where major operations are conducted by kernel evaluations. e.g.: Support Vector Machine

A Support Vector Machine (SVM) is an algorithm that outputs an optimal hyper-plane which categorizes new examples for given labeled training data (supervised learning). It finds the optimal solution that maximizes the distance between the hyper-plane and the points that are close to decision boundary. There are several techniques to solve the problem. This can be solved by quadratic and linear programs.

There are many types of optimization in mathematical programming, such as linear, quadratic, semi-definite, semi-infinite, integer, nonlinear, goal, geometric, fractional, etc. For example, linear programs have a linear objective and linear constraints. Convex problems are part of semi-definite programming.

Mathematical optimization is a sub-field of OR (Operation Research), OR optimization problems can be solved more efficiently using Machine Learning Techniques.

**Q.c) There is a family of algorithms based on Distance Metric Learning describe some of these algorithms.**

There are several techniques that are based on the measures of similarity between objects. Instead of finding similarity, we find dissimilarity between the objects, which would be easier and would give same results. For measuring dis-similarity, one of the parameter that can be used is **distance**. Which is also known as separability or divergence.

A distance metric is a real-valued function $d$, such that for any points $x$, $y$, and $z$:

$$d(x,y) and d(x,y) = 0, iff\ x = y$$

$$d(x,y) = d(y,x)$$

$$d(x,z) = d(x,y) + d(y,z)$$

The first property shows that distance cannot be negative. Second shows symmetry nature of distance and third point shows triangular inequality. Higher the distance between two points, more dissimilar the two points are.

There are several measures of distance which satisfies these properties.

- **Euclidean distance**

  The Euclidean distance is the most common distance metric used in low dimensional data sets. It is also known as the $L_2$ norm. The Euclidean distance is the usual manner in which distance is measured in real world.

  $$d_{euclidean}(x,y) = \sqrt{\Sigma_i (x_i - y_i)^2}$$

  Where $x$ and $y$ are $m$-dimensional vectors and denoted by $x = (x1, x2, x3...xm)$ and $y = (y1, y2, y3...ym)$ represent the $m$ attribute values of two records. While Euclidean metric is useful in low dimensions, it does not work well in high dimensions and for categorical variables. Drawback of the methods is that each attribute is treated as totally different from all of the attributes.

- **Manhattan distance (city block distance)**

  This metric is also known as the $L_1$ norm or the rectilinear distance. It is defined as the sum of distances travelled along each axis. The Manhattan distance looks at the absolute differences between coordinates. In some situations, this metric is more preferable to Euclidean distance, because the distance along each axis is not squared so a large difference in one dimension will not dominate the total distance.

  $$d_{manhattan}(x,y) = \Sigma_{i=1}^{m} |x_i - y_i|$$

- $L_\infty$ **norm (sup norm or the maximum norm)**

  The $L_\infty$ norm is the maximum of the absolute differences in any single dimension. This distance metric looks only at the measurement on which attributes deviates the most. Chebyshev distances are piecewise linear.

  $$d_{L\infty} = max_{i=1,2,...,m} |x_i - y_i|$$

- **Mahalanobis Distance**

  Mahalanobis distance is a well-known statistical distance function. Here, a measure of variability can be incorporated into the distance metric directly. Mahalanobis distance is ideal for samples drawn from identical distributions. It takes the correlations within a data set between the variable into the consideration. Mahalanobis distance is same as Euclidean distance when two data sets are uncorrelated. It is useful to use this distance in more advanced clustering technique to find outliers in multi-dimensional data.

  $$d_{mahalanobis}(x, y) = \sqrt{(x-y)^T S^{-1}(x-y)}$$

  Here, $S^{-1}$ is inverse of co-variance matrix.

  In the case of $\Sigma = I$, Mahalanobis distance is the same with Euclidean distance:

  $$d_{mahalanobis}(x, y) = \sqrt{(x-y)^T I^{-1}(x-y)} = \sqrt{(x-y)^2} = d_{euclidean}(x, y)$$

**Q.d) Algorithms that uses distance metric are**

- **K-nearest neighbor algorithm** (KNN) - is one of the basic and common classification algorithms.
- **Neural Networks** - is typically a collection of neuron like processing units with weighted connections between the units. The most common application of neural network is to train it on historical data and then use this model to predict the outcome for new combinations of inputs.
- **Support Vector Machine (SVM)** - is binary learning with some highly elegant property. It calculates a hyperplane between that separates two different classes in the training set.
- **Kernel Methods** - These methods usually consider classification as a linear function of the training data or a hyperplane that can be defined by a set of linear equations on the training data.
- **Clustering Methods** and **K-means algorithm** also uses distance metric. K-means algorithm does the best splitting of the data into a predetermined number of clusters (k). Which is used for clustering large data sets much faster than other clustering methods due to lower running time.

**Q.e) K-nearest neighbors algorithm (KNN)**

The kNN algorithm is belongs to the family of instance-based, competitive learning and lazy learning algorithms. It takes training set as an input which contains multi-dimensional data and class of every point and based on these training set, it finds class of other test data by finding distance between them.

Whenever we have a new point to classify, we find its K nearest neighbors from the training data. The distance is calculated based on different measures which is explained in previous section.

Simple KNN algorithm:

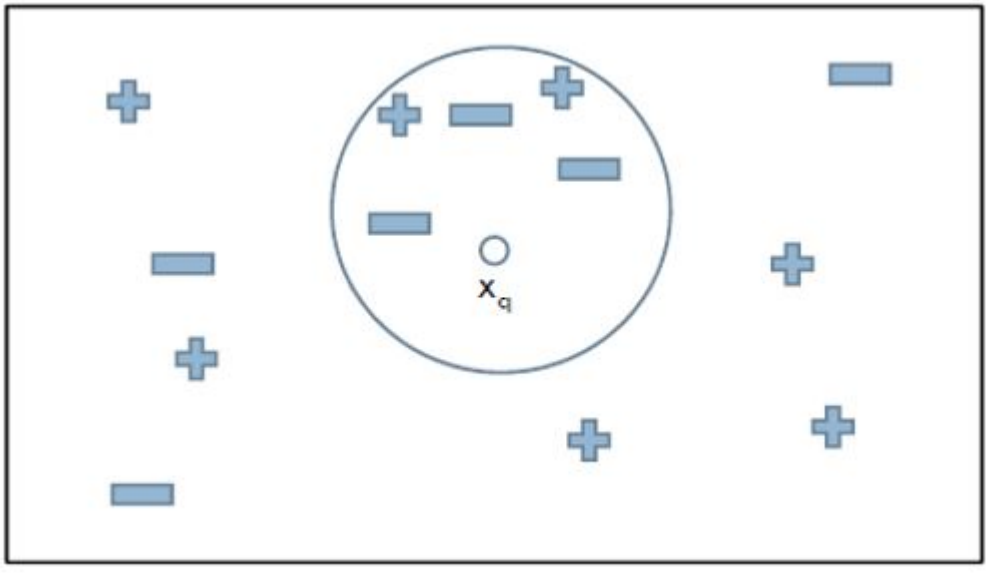



Figure 1: If $K = 5$, then in this case $x_q$ will be classified as negative since three of its nearest neighbors are classified as negative.

- For each training example $< x, f(x) >$, add the example to the list of training examples.
- Given a query instance $x_q$ to be classified,
  - Let $x_1, x_2.x_k$ denote the $k$ instances from training examples that are nearest to $x_q$.
  - Return the class that represents the maximum of the $k$ instances.

The algorithm can be used in many situations where the data are correlated to other and based on distance, we want to find the class of new point based on previous data sets.

Algorithm:

- **KNN (n points (m dimension), p new points, k)**
- Suppose, all points have m dimension, total number of learning data points are n and k is number of points to compare with.
- For all points p (having dimension m)
  - Find distance from new point p to all others data set (in our case, we use Mahalanobis distance as data taken is correlated)
  - Sort all the distances and select first k.
  - Find most frequent class in these k nearest points.
  - Assign determined class to the point p.

Where, Mahalanobis Distance for point $x$,$y$ and covariance matrix $S$, is given by $\sqrt{(x-y)^T S^{-1}(x-y)}$

**Q.g) Identify computational complexity of this algorithm based on your analysis.**

- Complexity for finding Mahalanobis Distance for between two points:

  Finding covariance would take $O(n * m^2)$ time as it has to take the means of all the data of size $n$ and then has to do it for every dimension (where $m$ is the number of dimensions) in a nested iteration, and thus producing a $m^2$ size matrix.

  **Multiplying Matrix:**

  $(x - y)$ matrix has dimension $m * 1$ and covariance matrix $S$ has dimension $m * m$. Hence, number of computations in multiplying $(1 * m) * (m * m) * (m * 1)$ is $m^2 + m$. Which can be considered as $O(m^2)$

  Hence finding Mahalanobis distance for all n nodes from a new node would take $O(n * m^2)$.

- **Complexity for KNN algorithm:**

  1. Find Mahalanobis distance for all n nodes $\rightarrow O(n * m^2)$
  2. Sort n distances $\rightarrow O(n * log\, n)$
  3. Select first k points $\rightarrow O(1)$
  4. Find most frequent class in those k points $\rightarrow O(k)$
  5. Assign determined class to point p $\rightarrow O(1)$

  Hence finding class of a point using KNN algorithm has complexity is $O(n * m^2)$

**Q.h) Is the algorithm that you studied/implemented - practical for continuous learning of input data?**

The above version is not designed to learn continuously as we have not taken the outcomes of test sets into the consideration for data sets.

**Q.i) Make necessary changes to the algorithm design so that it can learn continuously.**

We can apply continuous learning by taking feedback from user and use the feedback to improve our data set. Which is as following:

- Run all steps of KNN listed in above section and determine class of a point.
- Take feedback from the user, that is: Whether the class that our algorithm has predicted is correct or not.
  - If positive feedback comes, then we are confirmed that the given point is from the predicted class only.
- Add details of the point and class details to existing training data set.

Which would make enlarge our data set gradually and would be helpful class determination of points in future as the dataset is growing.

**Q.j) Is your new algorithm an exact algorithm or an approximation algorithm for learning?**

In many of the situations, Mahalanobis distance does not give proper measure of dissimilarity i.e. Dis-similarity with point P is higher in point A that point B, even though Mahalanobis distance

comes out to be lesser for point A than point B. Which occurs when the samples are drawn from non-identical distributions.

Which shows that algorithm that we have implemented is an approximation algorithm, where Accuracy is defined as:

$$Accuracy = \frac{no.\,of\,correctly\,classified\,samples}{no.\,of\,examples} * 100$$

**Q.k) Present your analysis of the new algorithm and in detail explain its complexity.**

The new algorithm is updating the dataset after finding the classification of a point. The algorithm remains same for finding the classification. Hence, the complexity and its calculation would remain same as described in section g.

**Q.k) If your new algorithm is capable to learn continuously, is it still practical for large size input data?**

The above version of the algorithm is not designed for large set of input as it takes all the dimension of every point into the consideration for finding Mahalanobis distance.

**Q.l) How can you modify data/algorithm so that this dimensionality can be handled?**

Below two methods are proposed for handling large dimensions.

- **Backward Elimination**

  In majority of the datasets, there are some dimensions that does not much affect in determining dis-similarity between points and classification. Here, we are trying to eliminate them and improving the accuracy as well as reducing the no. of dimensions in each point. Which would help while computing faster even in data having considerably large dimensions.

  Algorithm:

  - For all attributes do
    - Delete the attribute
    - For each training example $x_i$ in the training data set
      - Find the K nearest neighbors in the training data set based on the Euclidean distance
      - Predict the class value by finding the maximum n class represented in the K nearest neighbors
      - Calculate the accuracy as

        $$Accuracy = \frac{no.\,of\,correctly\,classified\,samples}{no.\,of\,examples} * 100$$

    - If the accuracy has decreased, restore the deleted attribute

- **Weighted KNN using backward elimination**

  Here, we normalize all the dimension and make range of all dimension from 0 to 1. After normalizing the values, we apply the same algorithm described above. Which turns out to be more accurate than simple backward analysis as each dimension is given same weightage.