Review on Practical Graph Mining with R

May 21, 2015

1 Chapter 1 on May 18, 2015

1. A graph is a collection of individual objects interconnected in some way. Graph data analytics refer to the extraction of insightful and actionable knowledge from graph data.

2. Applications of graph mining

   - Web graphs, e.g.., page ranks.
   - Social science graphs, which models the relationships among individuals and organizations. Links represent friendship, political alliance, professional collaboration, etc.
   - Computer networking graphs represent interconnections among various routers across the Internet.
   - Homeland security and cybersecurity graphs
   - Biological graphs
   - Chemical graphs
   - Finance graphs. The structure of stock markets and trading records can be represented using graphs, e.g., nodes are brokers, banks, and customers; links capture the financial trading information. A sequence of such graphs over a period of time can be mined to detect people involved in financial frauds, to predict which stocks will be on the rise, and to distinguish stock purchasing patterns that may lead to profits or losses.
   - Healthcare graphs

3. Dimensionality reduction in general refers to the problem of reducing the data amount while preserving the essential or more salient characteristics of the data. In terms of graph, dimensionality reduction refers to the problem of transforming graphs into low-dimensional vectors so that graph features and similarities are preserved.

2 Chapter 2 on May 18, 2015

1. A graph is a theoretical construct composed of points (i.e., vertices) connected by lines (i.e., edges). Graphs are structural data. The vertices of a graph symbolize discrete pieces of information, while the edges of a graph symbolize the relationships between those pieces.

2. A vertex is also called a node. It is a single point in a graph. Vertices are usually labeled. An edge can be regarded as a line connecting two vertices. Edges may have labels as well. Vertices and edges are the basic building blocks of graphs.

   - A graph G is composed of two sets, i.e., a set of vertices, denoted as $V(G)$; and a set of edges, denoted as $E(G)$.
   - An edge in a graph G is an unordered pair of two vertices (v_1, v_2) such that $v_1 \in V(G)$ and $v_2 \in V(G)$.
   - An edge is said to join its two vertices. Likewise, two vertices are said to be adjacent if and only if there is an edge between them. Two vertices are said to be connected if there is a path between one to the other via any number of edges.
   - A loop is an edge that joins a vertex to itself.
   - An edge is a multiple edge if there is another edge in $E(G)$ which joins the same pair of vertices.
   - Note that multiple edges and loops often make manipulating graphs more difficult. Many proofs and algorithms in graph theory require them to be excluded.
   - A *simple* graph is a graph with no loops or multiple edges.
   - An edge always has exactly two vertices, but a single vertex can be an endpoint for zero, one, or many edges. The *degree* of a vertex v, denoted as $degree(v)$, is the number of times v occurs as an endpoint for the edges $E(G)$. It indicates that the dgree of a vertex is the number of edges leading to it. Note that a loop adds 2 to the degree of a vertex.

3. A subgraph S of a graph G is a set of vertices.

   - A set of vertices $V(S) \subset V(G)$
   - A set of edges $E(S) \subset E(G)$. Every edge in $E(S)$ must be an unordered pair of vertices (v_1, v_2) such that $v_1 \in V(S)$ and $v_2 \in V(S)$. It indicates that an edge can only be part of a subgraph if both its endpoints are part of the subgraph.

4. A subgraph can be induced (i.e., induced graph) in two ways, i.e., by vertices and edges.

5. Two graphs G and H are *isomorphic*, denoted as $G \simeq H$, if there exists a bijection $f : V(G) \to V(H)$ such that an edge $(v_1, v_2) \in E(G)$ if and only if $(f(v_1), f(v_2)) \in E(H)$. Informally, it indicates that two graphs are isomorphic if they can be drawn in the same shape. If G and H are isomorphic, the bijection f is said to be an isomorphism between G and H and between H and G. The isomorphism class of G is all graphs isomorphic to G.

6. Note that when vertices and edges have labels, the notion of *sameness* and *isomorphism* are different. Two graphs having the same structure and thus *isomorphic*, but have different labels, and are thus not exactly the same. Labeled graphs are i*isomorphic* if their underlying unlabeled graphs are *isomorphic*.

7. An automorphism between graphs G and H is an isomorphism f that maps G onto iteself. The automorphism class of G is all graphs automorphic to G. It indicates that the graph structures are the same and the labels are the same too. All automorphisms are isomorphisms, but not all isomorphisms are automorphisms.

8. One common problem that is often encountered in graph mining is the subgraph isomorphism problem. The subgraph isomorphism problem asks if, given two graphs G and H, does G contain a subgraph isomorphic to H. That is, given a larger graph G and a smaller graph H, whether there is a subgraph in G that is the same shape as H. The problem is NP-complete, meaning it is computationally expensive.

9. A directed graph or digraph D is composed of two sets

   - A set of vertices $V(D)$
   - A set of edges $E(D)$, such that each edge is an ordered pair of vertices (t, h). The first vertex t is called the tail, while the latter vertex h is called the head. The edge in a directed graph is usually drawn as an arrow with the arrow-head pointing towards the head vertex.
   - Indegree of a vertex v is the number of edges in $E(D)$ which have v as the head.
   - Outdegree of a vertex v is the number of edges in $E(D)$ which have v as the tail.
   - Digraph isomorphism specifies that two digraphs J and K are isomorphic if and only if their underlying undirected graphs are isomorphic. So similar to labeled graphs, the direction of edges are not considered when considering isomorphism. Two digraphs are isomorphic if you can change all the directed edges to undirected edges and then draw them in the same shape.

10. Families of graphs

- A clique is a set of vertices which are all connected to each other by edges. A set of vertices C is a clique in the graph G if for all pairs of vertices $v_1 \in C$ and $v_2 \in C$, there exists an edge $(v_1, v_2) \in E(G)$. If begin at one vertex in a clique, you can get to any other member of that clique by following only one edge.
- A complete graph with n vertices, denoted as K_n, is a graph such that $V(K_n)$ is a clique.
- A path of length n, denoted as P_n, in a graph G is an ordered set of edges $(v_0, v_1), (v_1, v_2), (v_3, v_4), \cdots, (v_{n-1}, v_n)$ such that each edge $e \in E(G)$. A path is sometimes called a walk. Note that there may be more than one path between the same two vertices in a graph.
- A path is closed if its first and last vertices are the same. A cycle of length n, denoted as C_n, in a graph G is a closed path of length n. Note that a simple cycle is the same as a simple closed path, with the exception that it may visit one vertex exactly twice, i.e., the vertex which is both the start and end of the cycle.
- Trees are graphs that obey certain structural rules and have many appealing mathematical properties. A tree graph has exactly one vertex as the root, i.e., parent. It can have any number of children vertices adjacent to it. Those children can in turn, be parent of their own children vertices. A vertex having no children is called a leaf.
  - A graph G is a tree if and only if there is exactly one simple path from each vertex to every other vertex. Similarly, it can be defined as a graph G if and only if it is a connect graph with no cycles. Trees are often modeled as directed graphs. Hence, the root has an indegree of 0. All the other vertices have indegree of exactly 1. Leaf vertices have an outdegree of 0.

11. A weighted graph W is composed of two sets
    - A set of vertices $V(W)$
    - A set of edges $E(W)$ such that each edge is a pair of vertices v_1 and v_2 and a numeric weight w.
    - Weighted graphs are often used in path-finding problems.
12. Graph representations
    - Adjacency list
      - It is the simplest and most compact way to represent a graph
      - Given a graph G such that $V(G) = \{v_1, v_2, \cdots, v_n\}$, the adjacency list representation of G is a list of length n such that the i^{th} element of the list is a list that contains one element for each vertex adjacent to v_i.

- Adjacency matrix: It is a $n \times b$ matrix and the cell entry is either 1 or 0, indicating whether two vertices are adjacent to each other or not.
- Incidence matrix: Given a graph G such that $V(G) = \{v-1, v-2, \cdots, v_n\}$ and $E(G) = \{e_1, e_2, \cdots, e_m\}$, the incidence matrix is an $n \times m$ matrix. Let $a_{r,c}$ represent the matrix value at row r and column c.
  - If G is undirected, $a_{r,c} = 1$ if v_r is the head or tail of e_c; otherwise $a_{r,c} = 0$.
  - If G is directed, $a_{r,c} = -1$ if v_r is the tail of e_c; $a_{r,c} = 11$ if v_r is the head of e_c; $a_{r,c} = 0$ if e_c is a loop; otherwise $a_{r,c} = 0$.

3 Chapter 4 on May 18, 2015

1. Kernel methods refers to the transformation of data in a problem's input space into a high-dimensional feature space, allowing the algorithms to be performed on the transformed space, i.e., feature space.
2. The implicit transformation can be used by replacing the operations in the analysis algorithm itself with operations corresponding to a different feature space. Noted that the inner product of vectors in feature space can be obtained by using the inner product of the original space. The algorithm can be represented using the inner products and thereby, there is no need to compute the actual feature space at all. This is the kernel trick, and the function used to computes the inner products is the kernel function. Kernel function must be symmetric, i.e., $K(d_i, d_j) = K(d_j, d_i)$ and postive semi-definite. It can be interpreted as a measurement of similarity between pairs of input data.
   - Polynomial $(x \cdot y + \theta)^d$
   - Gaussian RBF $e^{\frac{|x-y|^2}{c}}$
   - Sigmoidal $tanh(\alpha(x \cdot y) + \theta)$

4 Chapter 5 on May 19 2015

1. Any network of links can be represented as a graph $G = (V, E)$, where V denotes the set of vertices (nodes) and E denotes the set of edges (links). This abstraction enables us to directly reason about links using concepts from graph theory.
2. Link analysis has several distinct task
   - Link-based object classification (LOC) is a technique used to assign class labels to nodes according to their link characteristics.

- Link-based object ranking (LOR) ranks objects in a graph based on several factors affecting their importance in the graph structure, whereas LOC assigns labels specifically belonging to a closed set of finite values to an object. The goal is to associate a relative quantitative assessment with each node.
- Link prediction

3. About social network analysis (SNA)

   - Social network analysis (SNA), *sna* package in R.
   - SNA studies the information flow and interactions between the members of a given network. Graphical representation of a social network is a popular way to understand and analyze the behavior of both the individuals and the overall network.
   - SNA is the representation and analysis of relationships/information flow between individuals, groups, organizations, servers, and other connected entities.
   -

4. Metrics for SNA

   - Degree: the degress of a vertex is the number of edges incident to it.
   - Density: the density of a graph is the number of existing edges over the number of possible ones.
   - Connectedness: the Krackhardt connectedness for a digraph G is equal to the fraction of all dyads (a group of two nodes), u and v, such that there exists an undirected path from u to v in G.
   - Betweenness centrality is a measure of the degree to which a given node lies on the shortest paths between the other nodes in the graph. Note that distance between two nodes in a graph is the minimum number of hops or edge traversals requied to reach the destination node from the starting node. Betweenness is a measure that represents the influence a node has over the connectivity of all the other nodes in a given network. A geodesic is the shortest path between any two nodes in a network. A node has high betweenness if the shortest paths (geodesics) between many pairs of the other nodes in the graph pass through that node.
   - Egocentric network of vertex v in graph G is the subgraph of G consisting of v and its neighbors.
   - Closeness centrality (CLC) is a measure defined for nodes of a given graph. The higher the value of CLC for a node is, the closer the node is to the other nodes in the graph. For a node v, CLC is defined as the ratio of the total number of nodes in the graph minus one to the sum of the shortest distances (geodesics) of the node v to every other

node in the graph. CLC represents how reachable the nodes of the graph are from any given node under consideration.

$$CLC(v) = \frac{|V| - 1}{\sum_{i, v \neq v_i} distance(v, v_i)}$$

5. The PageRank algorithm

   - It is an algorithm that addresses the linl-based object ranking (LOR) problem. The objective is to assign a numerical rank or priority to each web page by exploiting the "link: structure of the web.
   - The fundamentals of this algorithm are based on the count and pquality of backlinks or inlinks to a web page. A backlink of a page P_u is a citation to $P_u$from another page. The links from that page is called outlinks.
   - The computational and mathematical backbone of PageRank is the power method, which computes an eigenvector of a matrix. It is a recursive procedure that can run for days in case of bigger matrices representing the Web.
   - PageRank in R is `page.rank`.

6. Vertices as authority and hubs

   - A vertex is considered an authority if it has many pages that link to it (i.e., has a high indegree).
   - A vertex is considered an hun if it has many outgoing links (has a high outdegree).

7. Link prediction

   - Link prediction is used to predict new links in a graph.
   - Some notations
     - $G(t_i, t_j)$ represents the snapshot of graph G between time t_i and t_j.
     - *core* represents the set of vertices with at least a threshold number of edges
     - $k_{training}$ is the number of edges a vertex in the training set must have in order to be in the core set
     - E_{old} represents the set of edges in the training set
     - E_{new} represents the set of edges in the test set
   - Link prediction can be performed based on proximity measures. Proximity measures are used to find similarity between a pair of objects. A threshold on the number of edges a vertex needs to be adjacent to (in both training and testing data) is defined so that the prediction process is performed on a subset of graph. The core set contains

vertices that are adjacent to 3 or more edges in the graph. Given the training set $G(V, E_{old})$, the aim is to predict new edges among the vertices in core, in the test set.

- The proximity measures assign a numerical value $score(u, v)$ to the edge connecting a pair of nodes u and v. A list L is produced by ranking all such pairs in decreasing order of their scores. The score is defined as a negative value of the shortest path length.
- Some example proximity measures
  - Node neighborhood-based methods
    - Common neighbors method is a simple measure which take into account the intersection set of the neighbors of the vertices u and v. The score will be $score(u, v) = |N(u) \cap N(v)|$.
    - Jaccard coefficient is a proximity measure based on the node neighborhood principle. $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$.To measure dissimilarity, just subtract $J(A, B)$ from 1. The Jaccard coefficent can be defined for vertices u and v similarly and used for formulating scores $score(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$.
    - Adamic-Adar method computes the similarity between any two vertices using a common feature of the two, namely z. It is defined as $\sum_z \frac{1}{log(freq(z))}$, where $freq(z)$ is the frequency of occurrence of the common feature between u and v. Hence $score(u, v) = \sum_{z \in N(u) \cap N(v)} \frac{1}{log(|N(z)|)}$.
  - Ensemble of all path-based methodologies
    - PageRank
    - SimRank
  - Higher level methodologies
    - Unseen bigrams
    - Clustering

5 Chapter 6 on May 19 2015

1. Two algorithms for measuring proximity between vertices in a graph
   - Shared nearest neighbor (SNN) defines the similarity between two vertices in terms of the number of neighbors (i.e., directly connected vertices) they have in common.
   - The Neumann kernel is a generalization of the HITS that models the relatedness of vertices within a graph based on the number of immediate and more distant connections between vertices.
2. The shared nearest neighbor graph: let $V = \{v_1, v_2, \cdots, v_n\}$ be the set of nodes in a KNN graph G. In the SNN graph derived from$G$, the neighbors of v_i are the nodes $v_j, j \neq i$ such that v_i and v_j have at least k neighbors

in common or equivalently, if there exist at least k distinct paths of length two between v_i and v_j n G. SNN is used for undirected but possibly edge-weighted graphs.

3. Evaluating relatedness using Neumann Kernels: It can be used for both directed and undirected graphs.

6 Chapter 7 on May 19 2015

1. Frequent subgraph mining (FSM) is the process of discovering subgraphs that occur often in a database of other graphs. FSM can be used to highlight interesting structural properties in graph-based data. It is a type of unsupervised learning.

2. Graphs primarily convey structural information. Finding frequent substructures is often a reliable way to classify data that can be expressed as a graph.

3. All FSM techniques take a set of graphs as input and returns a set of graph that occur frequently in the input. All techniques share a common basic structure as follows

   - Candidate generation
   - Candidate pruning
   - Support counting

4. Pattern growth approach

   - Modern FSM algorithms begin with a single vertex and grow the graph one edge at a time. Each time a new edge is added, a new vertex may need to be added as well. After each extension, the frequency of the new graph is checked.
   - Efficiency is enhanced by avoiding checking graphs that are isomorphic to graphs which have already been checked.

5. Frequent subgraph mining package in R is `subgraphMining`. It contains the SUBDUE, gSpan, and SLEUTH algorithms.

6. gSpan (Graph-based Substucture PAtterN)

   - It is a depth-first pattern growth algorithm
   - It features an important innovation that significantly restricts the number of redundant subgraph candidates that must be examined.
   - It can significantly outperforming its Apriori-based predecessors and the parallel version exhibits good scalability.

- Compared to other general pattern growth algorithms, the only major difference comes during the process of candidate generation, which avoids generating subgraphs that are isomorphic to other subgraphs already considered.
- It represents graphs with a special encoding unique to gSpan that is called gSpan encoding.
- gSpan begins with a single edge (two vertices) subgraphs. Each subgraph is extended one edge at a time, and the support of the newly generated graphs is measured. If the support is equal or greater than *min support*, that subgraph will be expanded further. Once it has been extended such that its support is less then *min support*, the algorithm back-tracks until it has returned to a more promising subgraph, allowing gSpan to expand graphs in a depth-first manner.
- The gSpan encoding: each edge is represented using five tuples, the first and second are the unique indices of starting and ending points, the third and fifth are the labels of vertices, and the fourth is the edge label.
- The rightmost expansion: in a depth-first traversal, the starting node is called the root, and the last node is called the rightmost node. When considering depth-first seach tree, the shortest path from the root to the rightmost node is the rightmost path. When expanding a gSpan encoding via rightmost expansion, it is only possible to add a new edge if one of its endpoints lies on the rightmost path.
- Lexicographic order
- Minimum gSpan encoding (MgSE) is simply the smallest of all the possible gSpan Encoding (the lowest one in the lexicographic ordering). MgSE can be used to decide whether two graphs are isomorphic. Note that the MgSE is always the same for isomorphic graphs but different for non-isomorphic graphs. MgSE is the key to candidate pruning in gSpan.

7. SUBDUE algorithm

- Note that gSpan is an exact algorithm which find the complete set of frequent subgraphs in a database of graphs. It may be time-consuming for large data. In addition, some of the frequent subgraph may not be interesting, e.g., all the frequent 1-vertex subgraph.
- The most iwidely used FSM algorithms, which designed for real-world problems, is the SUBDUE. It uses a constrained form of beam search to discover frequent subgraphs. It reports structures based on the amount of compression they provide for the original gragh. If a subgraph allows for a lot of compression, it is assumed to be interesting and therefore deserves to be further explored. SUBDUE is an efficient but sometimes inexact method.

- SUBDUE is fast due to the adoption of beam search. Beam search is a best-first version of breadth-first search, so only the best k children are expanded at each level of the search. The rest are ignored. k is called the beam width of the algorithm.
- SUBDUE starts with all the frequent single vertex subgraphs and expand them one edge at a time. SUBDUE only explores the best k of these expanded subgraphs.
- The other novel feature is that SUBDUE decides which subgraphs are interesting based on the compression they provided. Compression simply means representing data using a smaller number of bits, analogous to file compression techniques. The subgraphs that SUBDUE considers important are the ones that allow it to compress the original set of graphs into fewest number of bits.
- Description length of a graph G is denoted as $DL(G)$. The value of $DL(G)$ is the integer number of bits required to represent graph G in some binary format.

8. SLEUTH algorithm
    - Designed for tree-type graphs
    - Can be used to mine frequent subtrees within a collection of trees

7 Chapter 8 on May 21 2015

1. Two types for graph data, i.e., within-graph clustering and between-graph clustering.
2. Minimum spanning tree clustering
    - Spanning tree is a conncected subgraph with no cycles that includes all vertices in the graph
    - Minimum spanning tree (MST) is the spanning tree with the minimum possible sum of edge weights, if the edge weights represent distance and maximum possible sum of edge weights, if the edge weights present similarity.
    - The canonical algorithm to find MST in an undirected weighted graph is Prim's algorithm. In R, the Prim algorithm is implemented in package *igraph*.
    - Clusters can be created by working on the MST obtained. The removal of $k-1$ edges will create k clusters. The removal begins with those with the larger weights. Can be performed in R using $GraphClusterAnalysis : k_clusterSpanningTree$.

3. Note that MST-based clustering defines similarity between vertices based on the high-weight edges and paths between vertex pairs. Removal of possibly large number of edges results in loss of information in the resulting clusters. A more general notion of similarity between vertex pairs can be captured by the *sharednearestneighborclustering*

   - Unlike the weight-based measures, it provides a natural and robust way of preventing noise in the data set (e.g., the removal or addition of edges into the graph) from affecting the clustering results.
   - The idea is to place a pair of objects into the same cluster, if the number of common neighbors they share is more than some threshold. It can work on both edge-weighted graphs using some proximity measure (e.g. cosine similarity) and unweighted graphs.
   - The Jarvis-Patrick clustering is a popular within-graph SNN clustering algorithm. In R, can be performed using $igraph, RBGL, GraphClusteringAnalysis$ packages.

4. Between centrality clustering

   - Betweenness centrality quantifies the degree to which a vertex occurs on the shortest path between all the other pairs of nodes. The graph can be clustered by removing nodes with high betweenness centrality, under the assumption that such nodes typically separate different clusters in the graph.
   - Girvan and Newman developed a clustering algorithm using edge between centrality. The between centrality of all edges is calculated and the one with the highest edge-betweenness centrality is removed. The process is performed iteratively until the highest edge-betweenness centrality in the graph falls below a user-defined threshold. One shortcoming is that the method divides the graph into non-overlapping clusters.
   - To overcome this limitation, Pinney and Westhead uses the vertex-betweenness centrality in a graph while retaining the vertices that the algorithm partitions based on. It iteratively divides the graph at the vertex with the maximum betweenness centrality. At the end of the algorithm, each removed vertex v is inserteed into all clusters that share at least one edge with v.
   - The algorithm can be performed using R packages $graph, GraphClusterAnalysis, RBGL$.

5. Clustering vertices with kernel k-means

   - k-means clustering can be used to cluster the vertices in a graph, given an appropriate definition of vertex similarity through kernel functions.
   - Implemented in R using packages $igraph, RBGL, GraphClusterAnalysis, graph, kernlab$.

- One drawback is that the graph-based k-mean clustering suffers from the poor initial centroid placement and is prone to local minima. One common strategy is to generate several random sets of centroids and pursue the most promising based on a predetermined measure of cluster quality.

6. How to choose the clustering algorithms

   - Complete (e.g., graph-based k-means) or partial clustering (e.g., maximal clique enumeration).
   - Overlapping (e.g., maximal clique enumeration and vertex betweenness-based centrality algorithms) or exclusive clustering (e.g., k-means).

8 Chapter 9 on May 21 2015

1. Two types, i.e., classifying multiple graphs and classifying individual vertices within a single graph. Kernel-based classification techniques are mainly adopted as they have been successfully applied in practice and are based on an implicit graph-to-vector mapping, allowing us to use well known numerical-vector-based classification methods.

2. Graph classification is to classify a number of individual graphs.

3. Graph classification using direct product kernel

4. Vertex classification using the regularized Laplacian kernel

9 Chapter 11 on May 21 2015

1. Graph-based anomaly detection

   - "White crow anomaly" is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism.
   - "In-disguise anomaly" is an observation that is only a minor deviation from the normal pattern.
   - "Pointwise anomaly" is a point in time at which the observed values are significantly different than the rest of the timestampls in the time series.

2. Random walk algorithm can be used to find "white crow anomaly" in dataset without any temporal aspect.

3. GBAD algorithm is applied to find "in-disguise anomalies".

   - It is an unsupervised method based on the SUBDUE system, which designed to find anomalies due to modification, insertion, or deletion.

- GBAD-MDL uses the MDL approach to find the best substructure in the graph and subsequently searches for all the other substructures that look similar to the best. The best substructure is the one that occurs the most frequently and causes maximum compression in the graph using the MDL principal.
- GBAD-P follows the basic principle of MDL by finding the best substructure in the graph. For each occurrence of that substructure, its extension is examined. The extension can be in the form of adding vertex or edge. A probability of occurrence is calculated for each extension and the one with the lower probability is considered anomalous.
- GBAD-MPS also uses the MDL to find the best substructure S in the graph. The parent substructure of S has the exact same structure of S with one or model nodes or edges removed. The algorithm finds the substructure that match the parent substructure of S. Cost is associated with addition of vertex or edge. The substructure that occur infrequently and require a higher cost to transform are considered anomalous.

4. Research on "white crow" anomalies is mainly focused on exploring three different anomalies, i.e., anomalous nodes, edges, and subgraphs.

5. "In-disguise" anomalies are more difficult to detect. The GBAD methods are designed to find such anomalies which may include label modification, vertex or edge insertion, and vertex or edge deletion. Note that they are not designed to find anomalies in multiple graphs.

10 Chapter 12 on May 21 2015

1. Supervised learning performance matrics

   - Confusion matrix for classification problems
     - $accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
     - $accuracy = \frac{FP+FN}{TP+TN+FP+FN}$
     - Recall quantifies the notion of sensitivity. It contains positive recall (true positive rate) and negative recall (true negative rate).

$$TPR = \frac{TP}{TP+FP}$$

$$NR = \frac{TN}{TN+FN}$$

     - Misclassification rate is analogous to recall and has false positive rate and false negative rate.

$$FPR = \frac{FP}{TN+FP}$$

$$TNR = \frac{FN}{TP + FN}$$

- Precision quantifies the notion of specificity. $P^+ = \frac{TP}{TP+FN}$ or $P^- = \frac{TN}{TN+FP}$
- $F - measure$ is the harmonic mean of precision and recall values of a class and provides a way to judge the overall performance of the model for each class.

$$F^+ = \frac{2 \times P^+ \times TPR}{P^+ + TPR}$$

$$F^- = \frac{2 \times P^- \times TNR}{P^- + TNR}$$

- $G - mean$ is the geometric mean of TPR and TNR (positive and negative recalls): $G - mean = \sqrt[2]{TPR \times TNR}$. This one is a better way to quantify the performance of the model since it is less prone to the effects of the underlying class distribution. It works especially well for imbalanced classification problems.
- Critical success index (CSI) is also called the threat score, is the proportion of the correct predictions of class L of the sum of all predicted values of L and all values of L not correctly predicted. It is useful for multi-class classification problems.
- Hit rate (HR) is the success rate of each class.
- Bias is the ratio of the total points with label L to the number of points predicted as L. If > 1 means under-prediction and < 1 means over-prediction for a certain class L.

2. Unsupervised learning performance metrics

- Evaluation using prior knowledge
  - Contingency table
  - Rand statistic is also called the simple matching coefficient. It is a measure where both placing a pair of points with the same class label in the same cluster and placing a pair of points with different class labels in different clusters are given equal importance, i.e., it accounts for both specificity and sensitivity of clustering.
  - Jaccard coefficient is used when placing a pair of points with the same class label in the same cluster is primarily important.
  - Entropy, 0 means homogeneous and 1 means impurity.
  - For clusters
    * Cohesion and separation
    * Dunn index ranges from 0 to infinity and should be maximized.
    * Silhouette coefficient