DATA MINING NOTES

BRENDAN WHITAKER

2010 *Mathematics Subject Classification.* 12-XX

Abstract. A comprehensive set of notes for Data Mining course, taken SP18 at The Ohio State University.

Contents

Part 1

CHAPTER 1

Introduction

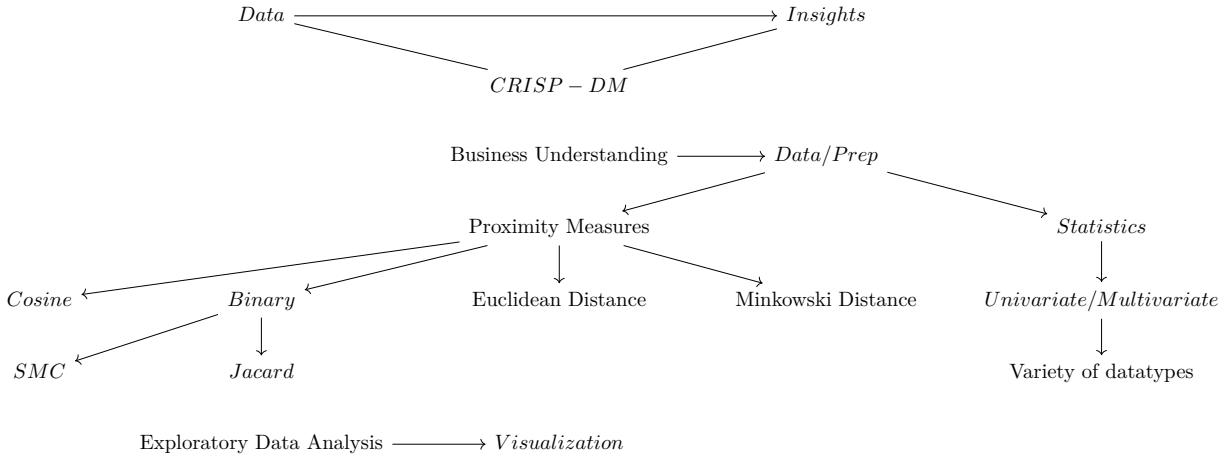
Wednesday, January 10th

Note $class = label = category$. These are the dependent variables, the stuff that our work is determining.

Ratio variables are your typical real-valued numbers, i.e. zero is meaningful. Interval variables have 0 as just another possible value, it is not the additive identity in this case.

Monday, January 22nd

Review of stuff up to now.



Imputation: replacing missing data with the mean is one way we could do it. We could also use regression imputation to estimate the value of the missing variable using the data from other variables. You could also use a random value.

Monday, January 29th

PCA - principal component analysis.

Monday, Febrary 5th We have:

$$P(+|x) = \frac{1}{1 + e^{-\beta_0 - \beta_1 x_1 - \beta_2 x_2}}.$$

Okay so last time, we talked about how you might compute a ROC curve. Recall that a ROC curve plots false positive rate (FPR) on the x-axis and true positive rate (TPR) on the y-axis. So we are going to set our $t = 0$.

1. Exam 1

Need to know proximity measures, types of data (ratio).

- Won't ask to compute covariance, that's mean.
- Know bayes theorem for bayes classifiers.
- PROXIMITY MEASURES, INFER IF WE'RE TALKING ABOUT A SIMILARITY VS DISSIMILARITY.
- know minkowski distance.
- don't know mahalaboni distance.
- simple matching and jaccard coefficients, and cosine.
- you use jacard when data is SPARSE, otherwise SMC.
- Make a reference sheet to use on exam.
- You'd use cosine when it is no longer binary data, but it can be used for anything.
- SMC and Jaccard are for BINARY ONLY.
- Know that the end of boxplots are 90 and 10.
- Interpret this scatterplot or boxplot on exam. Any of the data vis types.
- What is the difference between noise and outliers?
- Principal component analysis is going to be fair game, or part of the calculation, how do you do it, what does it mean, what are you trying to accomplish.
- how many principal components do you keep given some type of data loss requirement.
- Difference between dimensionality reduction, and feature subset selection.
- Dimensionality reduction creates new features, no longer interpretable, the values are meaningless, each of the new dimensions are combinations/transformations of the old ones.
- Feature subset selection just removes a subset of the n dimensions and uses those, which preserves the meaning of the dimensions/features.

- When you run PCA, you aren't doing anything related to the class variable.
- But Feature subset selection on the other hand, is usually done relative to a classifier.

Now let's talk about classification.

- What is objective of classification: create a generalizeable, predictive model. It has to GENERALIZE!
- As you train your model and it becomes more complex, training error keeps dropping, but test error find a local min and then goes up again.
- Producing a classification label is a two step process.

EXAMPLE 1. Suppose we're doing a binary classification.

Step 1:

For a given record x, define:

$$P(+|x) = 0.64 \tag{1}$$
$$P(-|x) = 0.36$$

For logistic regression, we have:

$$P(+|x) = \frac{1}{1 + e^{-*\beta_0 + \beta_1 x)}}.$$

So x enters our model and get the the above posterior out of it.

Step 2:

We set some threshold t and say $x \in +$ if the posterior is $\geq t$ and in $-$ otherwise. And 0.5 is our default t value. You get an ROC curve only by varying t. Note the confusion matrix is dependent on the value of t.

- A ROC curve has FPR on x axis and TPR on y axis.
- When we increase t, precision generally goes up because we expect the model to work better than random.
- Recall precision is TP/(TP + FP), it is the top left box over the sum of the left half of the matrix.
- You should know DECISION TREES, NAIVE BAYES, kNN. Ensembles just a little bit. Rule based classifier.
- **What is reduced-error pruning?** Before we talk about that, normal pruning is just computing error for all nodes and removing nodes that don't give you lift (reduction of error rate). Reduced error pruning using validation data, you compute error rates for each node on the TEST DATA instead, then compute lift and decide to prune any nodes that don't give you a reduction in error.
- for notes sheet just print a shitton of these pages.

2. Review

A **clustering** is a set of clusters. There is an important distinction between hierarchical and partitional sets of clusters. **Partitional clustering** is a division of data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset. **Hierarchical clustering** is a set of nested clusters organized as a hierarchical tree. We have **traditional** vs **non-traditional hierarchical clustering**. Traditional is when for each cluster C_i, all other clusters are either a subset or a superset of cluster C_i. Non-traditional hierarchical clustering is any clustering where this condition does not hold. **Non-exclusive clustering** is when points may belong to multiple clusters. **Fuzzy clustering** is when a point belongs to every cluster with some weight between 0 and 1, analogous to a fuzzy set. Note the weights of each point must sum to 1. A **partial clustering** is a clustering that does not include all the data in the dataset, a **complete clustering** is exactly what it sounds like. We can also qualitatively say a clustering is **homogeneous** or **heterogeneous** is they have different sizes, shapes, densities, or if these measures are all the same ish. A **well-separated cluster** is a set of points such that any point in a cluster is closer to every other point in the cluster than to any point not in the cluster. A **center-based** cluster is when each point in a cluster is closer to a pre-defined "center" of the cluster than to the "center" of any other cluster. Types of centers are **centroids**, the average of all points in the cluster, or the **medoid**, the most "representative" point of a cluster. **Contiguous clustering** is when any point in a cluster is closer to one or more other points in the cluster than to any other point in the cluster (like 1-nearest neighbor, the cluster is defined by setting the class of a point to be equal to the class of the point closest to it). **Density-based clustering** is when a cluster is defined as a dense region of points which is separated by a low density region. This is useful when we may have a lot of low-density noise. Contiguous clustering might put everything in one huge cluster if there's a lot of background noise points (think like a low-density lattice), but density based clustering will filter it out. Many clustering techniques are based on trying to minimize or maximize a global **objective function**. The clustering problem then becomes an optimization problem, which, in theory, can be solved by enumerating all possible ways of dividing the points into clusters and evaluating the "goodness" of each potential set of clusters by using the given objective function. Of course, this "exhaustive" approach is computationally infeasible (NP complete) and as a result, a number of more practical techniques for optimizing a global objective function have been developed. **Hierarchical clustering** algorithms typically have local objective functions, while **partitional algorithms** tend to have global objectives. Another way to use a global objective function for clustering is to **fit the data to a parameterized model**: parameters determined from the data, and use mixture models which assume the data is just a mixture of several canonical statistical distributions. Another approach is to map the problem to a different domain and solve an equivalent problem. In this case, we can consider the points as nodes in a **weighted (undirected) graph** where nodes are related by a proximity matrix, and the weighted edge values are the proximities between points. Clustering is then equivalent to breaking the graph into connected components, where we minimize edge weights between clusters and maximize edge weights within clusters. (so in this case the proximity is a similarity value, not a dissimilarity). **K-means clustering** is a partitional clustering approach where each cluster uses a centroid which is typically the mean of the points in the cluster. Each point is assigned to the cluster with the nearest centroid, using a proximity metric, usually Euclidean distance. We give the basic algorithm: [Select k points as the initial centroids. While the centroids have changed, Form k clsuters by assigning all points to the closest centroid, then Recompute the centroid of each cluster.] To evaluate k-means, the most common measure is known as **SSE**, the sum of squared error. It is defined as: $SSE = \sum_{i=1}^{k} \sum_{x \in C_i} (dist(m_i, x))^2$, where C_i is the i-th cluster, and the m_i's are the representative points of the i-th cluster, usually centroids. Increasing k decreases the SSE. **Limitations of k-means:** the randomly selected initial centroids may be poor, we have to handle empty clusters, outliers, and noise, and the algorithm has trouble with clusters of differing sizes, densities, and non-globular shapes. The algorithm also has difficulty with outliers. **Solutions to the initial centroids problem:** multiple runs (helps, but probability is not on your side), sample and use hierarchical clustering to determine initial centroids, select more than k centroids and then pick the most widely separated k from these, use postprocessing, or use bisecting k-means. **Bisecting k-means** is a variant which can produce a partitional or hierarchical clustering and is not as susceptible to initialization issues. The algorithm is as follows: [Initialize a list of clusters, and populate it with just a single cluster containing all the data points. While {the list of clusters has less than k clusters} Select a cluster from the list. For $\{i = 1$ to $numberOfIterations\}$ Bisect the selected cluster using basic k-means ($K = 2$). Add the two clusters from the bisection ($k = 2$ clustering) with the lowest SSE to the list of clusters.] Note in the above algorithm that $numberOfIterations$ is determined a priori, and that a higher value with yield lower SSE bisections, and thus a "better" overall clustering. Some pre-processing techniques for k-means are normalization and elimination of outliers. Post processing techniques include elimination of small clusters that may represent outliers, splitting of loose clusters (high SSE), and merging clusters that are close and have relatively low SSE. Note the SSE for a single cluster is just one term in the outer sum. **Mixture models** are a particular type of statistical model that views that data as a set of observations from a mixture (combination) of different probability distributions. Each distribution corresponds to a cluster, the parameters of the model describe the cluster characteristics, and the most common is the **Gaussian** mixture model, in which each cluster is described by a mean vector and covariance matrix. **Maximum likelihood** can be used, but it is often too difficult to use with mixture models. Instead, we use the **Expectation Maximization** algorithm (EM), whcih calculates the probability that each point belongs to each distribution and then uses these probabilities to compute a new estimate for the parameters. The algorithm is as follows: [Select an initial set of model parameters. Then while the parameters haven't converged, {**expectation step:** For each object, compute the probability that the object belongs to each distribution. }{**maximization step:** Given the probabilities from the E step, find the new estimates of the parameters that maximize expected likelihood.}] Note k-means is just a special case of EM when Gaussians are spherical (off diagonal terms of covariance matrices are zero) and have equal covariance matrices, and different means. Expectation step is analogous to the cluster assignment, and the maximization step is analogous to the recomputing centroids. **Hierarchical clustering** produces a set of nested clusters organized as a hierarchical tree, which can be visualized using a dendrogram. Two types: **Agglomerative** is when we start with the points as individual clusters and merge the closest pair until only k remain. **Divisive** is when we start with one all inclusive cluster and split until we have k. Traditional hierarchical clustering algos use a similarity or distance matrix. We give the more popular algorithm for agglomerative hierarchical clustering: [Compute proximity matrix for all data points. Let each data point be a cluster. Then while more than k clusters remain, {merge the two closest clusters and update the proximity matrix. }] Several ways to compute cluster proximity: MIN, MAX, dist between centroids, group average, Ward's method. Review **Slides 51-53** on actually running algo and constructing dendrogram. MIN handles non-elliptical shapes well, sensitive to noise and outliers. MIN means $dist(C_i, C_j) = \min_{p \in C_i, q \in C_j} dist(p, q)$. MAX is analogous, less susceptible to noise, outliers, but it tends to break large clusters and biased towards globular. [DO MAX EXAMPLE SLIDE 56] Group average defines: $dist(C_i, C_j) = (\sum_{p \in C_i, q \in C_j} dist(p, q))/(|C_i||C_j|)$ where the numerator is the product of the cardinalities. Group avg less susceptible to noise, outliers, biased towards globular. **Ward's method:** Similarity of two clusters is based on the increase in squared error when two clusters are merged, so you pick the two that increase it least when merged. Similar to group average if distance between points is squared. Less susceptible to noise, outliers, biased towards globular, hierarchical analogue of k means, can be used to initialize k means. Hierarchical clustering takes $O(N^2)$ space complexity since prox matrix. Takes $O(N^3)$ time since N steps, at each we recompute matrix. **Limitations of Hierarchical:** decisions to combine can't be undone, no objective fnct is minimized, instead, local criteria are used at each step (like greedy method). Different prox criteria have trouble with noise, outliers, different sized clusters and convex shapes, breaking large clusters. **DBSCAN: Density** = number of points within a radius ϵ. We say a point is a **core point** if it has more than `minPts` within ϵ. These are points in cluster's interior. We say it's a **border point** if it has fewer than `minPts` within ϵ, but is within ϵ of a core point. We say it's a **noise point** if it's neither core nor border. DBSCAN algo: 1. Label all points as core, border, noise. 2. Eliminate noise points. 3. Consider as graph, put an edge between each pair core points less than ϵ apart. 4. Each group of connected core points is cluster. 5. Assign each border point to one of the clusters of it's associated core points. DBSCAN works well with clusters of different shapes and sizes but with uniform density. Bad with varying densities, high dimensionality. DBSCAN assumes for points in a cluster that the k-th nearest neighbors are at roughly the same distance, and noise points have k-th nearest neighbors at farther. Other density algos: **Grid-based**: split possible values of each attribute into a number of intervals. [1. Define a set of grid cells. 2. Assign objects to the appropriate cells and compute the density of each cell. 3. Eliminate cells with a density below some threshold τ. 4. Form clusters from adjacent cells.] **Subspace clustering:** cluster in a subspace, so cut out one or more dimensions/attributes/features. Data may form nice clusters in a subspace but not in a superspace. The **CLIQUE** algo finds subspace clusters using the principle that if a set forms a density cluster in k dimensions, then it is a cluster in all subspaces of those dimensions. [1. Find all dense areas in 1-dim spaces for each attribute. 2. Set $k \longleftarrow 2$. 3. While {there are candidate dense k-dimensional cells} { generate all candidate dense k-dimensional cells, from the $k-1$ dense cells. Eliminate cells with less than ξ points. Set $k \longleftarrow k+1$. } 4. Find clusters by taking union of adjacent, high-density cells. 5. Summarize each cell using inequalities corresponding to the ranges of the attributes.] **Graph-based clustering.** The **proximity graph**, which is constructed as follows:[1. have prox matrix. 2. Each point is node. 3. Fully connected, edge weights are proxims.] I don't know what this means: (MIN is single link, MAX is complete link, both start with prox graph). Clusters are connected components. Problem: for most data, objects are highly similar with a small number of objects, weakly similar to most other objects. **Sparsification** is used to break all links by setting low-sim values to zero before clustering. One approach is to Break all links that have a similarity (dissimilarity) below (above) a specified threshold. Another is to Keep only links to k-nearest neighbors of point (kNN graph). Benefits of sparsif.: drastically reduces amount of information that needs to be processed. Running time down, problem size capacity increased. Also clustering may work better: reduces noise, outliers, sharpens distinction. **Min span tree** of graph is subgraph that has no cycles, contains all nodes, has min total edge weight. **Min span tree clustering algo:** [1. Compute min span tree for (DISSIMILARITY GRAPH). While {there are non-singleton clusters remaining:} {create new cluster by breaking link for biggest dissimilarity. }] MST is **divisive hierarchical algorithm**. Computing MST and breaking largest dissim link can be thought of as sparsification. **Limitations of merging schemes**: most clustering techs have a *global model of clusters*, can't handle when clusters vary in size, shape, density. **Computing shared nearest neighbor similarity graph algo**: [1. Find k-nearest neighbors of all points. 2. if two points are not kNN of each other, set similarity to 0. Else similarity is number of shared neighbors (max of $k-1$).] **Jarvis-Patrick clustering algo:** [1. Compute SNN sim graph (last algo). 2. Sparsify the SNN graph by applying similarity threshold. 3. Find connected components, those are clusters.] JP clustering is brittle, may split true clusters or join clusters that should be kept separate since one link may determine such things. **SNN density-based cluster algo:** [1. Compute sim matrix. 2. Construct SNN sim graph. 3. Apply DBSCAN with ϵ and $minPts$ using SNN graph.] This algo can handle different densities. We discuss **Cluster Validity.** Ways to validate: 1. Determine **clustering tendency:** which is to determine if a non-random structure exists in data. 2. Compare to externally given class labels. 3. Evaluate without external info (use only data). 4. Compare two different cluster analyses. 5. Determine "correct" # of clusters. Types of measures: external index, internal index, relative index. **External index** is used to see if matches external class labels (supervised). Entropy is example. **Internal index** used to measure goodness without external info (unsupervised). **Relative index** compares two clusters(ings). **Cohesion** measures how closely related objects in a cluster are. **Cluster SSE**= $\sum_{x \in c_i} dist(c_i, x)^2$, where c_i is centroid. **Cluster separation** measures how distinct clusters are from each other. Between cluster Sum of squres. **Total SSB**= $\sum_{i=1}^{k} m_i dist(c_i, c)^2$ where m_i is size of cluster i, c_i is centroid of cluster i, c is average of all data points. Total SSB + Total SSE = constant called total sum of squares. Total SSE is sum of SSE for all clusters. **Statistical framework for SSE** consider values of SSE for random data, if the SSE for true data is atypical for this dist, it is likely a good clustering. **Silhouette coefficient** like cohesion and sep, but for ind points. For a point i, compute a, avg dist of i to points in its cluster. Compute b, min (avg dist of i to points in another cluster). Then coeff is $s = 1 - 1/b$ if $a < b$ or $s = b/a - 1$ if $a \geq b$. Closer to 1 the better. **Prox graph** can also be used for cohesion, sep. **cluster cohesion** is sum of weight of all links (edges) within cluster. **Cluster sep** is sum weights betwee nodes in cluster and nodes outside cluster. **external measures of validity**: entropy, purity(max class prob), recall, precision, F-measure. **Precision** = $\frac{TP}{TP+FP}$. **Recall** = $\frac{TP}{TP+FN}$.(same as TPR). **F-measure** = $2(precision)(recall)/(precision + recall) = \frac{2TP}{2TP+FN+FP}$. **Confusion Matrix**. Predicted class on top, actual class on left. Top left is TP, top right is FN, bottom left is FP, bottom right is TN. **Accuracy** $\frac{TP+TN}{TP+TN+FP+FN}$. **Error Rate:** $1 - Accuracy$. TPR (sensitivity) $\frac{TP}{TP+FN}$. TNR (specificity) = $\frac{TN}{TN+FP}$. FPR = $\frac{FP}{FP+TN} = 1 - TNR$. FNR = $\frac{FN}{FN+TP} = 1 - TPR$. [FIGURE OUT HOW TO COMPUTE ENTROPY, PURITY slide 111]

FREQUENT PATTERN MINING. Items and **baskets**, baskets are subsets of the set of items. k itemset has k items. **Support count** σ is frequency of occurrence of itemset. **Support** is fraction of all transactions that contain an itemset. **Frequent itemset** is itemset whose support is $\geq$ minsup threshold. Association rule is a map $X \rightarrow Y$ where X, Y are itemsets. **Support(s)** of an assoc. rule is fraction of transactions that contain both. $s = \sigma(X \cup Y)/|T|$. where $|T|$ is total num transacs. **Confidence(c)** is how often items in Y appear in transactions with X. $c = \sigma(X \cup Y)/\sigma(X)$. Goal is to find all rules having support $\geq$ minsup, and confid. $\geq$ minconf. Algo: [1. Generate frequent itemsets (all with support $\geq$ minsup. 2. Generate high-confid. rules from each frequent itemset, each rule is binary partitioning of frequent itemset.] **Apriori** principle: if an itemset is frequent, then all of its subsets must also be frequent. Equivalently, if an itemset is infrequent, all of it's supersets must be infrequent. The **apriori** algorithm uses $F_{k-1} \times F_{k-1}$ to generate candidates. It generates $\binom{n}{k}$ candidate k-itemsets, where n is the number of frequent $k-1$ itemsets. Then it prunes infrequent candidates. **Apriori algorithm:** [1. $k = 1$. 2. Generate frequent itemsets of length 1. 3. While {we have identified new frequent itemsets}{Generate using F_{k-1}. Prune itemsets that contain infrequent subsets of length $k-1$. Count support of each new candidate. Eliminate infrequent ones. }] $F_{k-1} \times F_1$: extend each frequent $k-1$-itemset with frequent 1 itemsets, and only do this when the 1s come after all items in the $k-1$s when sorted in order. If minsup is too high, miss itemsets involving rare interesting items. If too low, computationally expensive. **Multiple Minsup**: Define $MS(i)$ for each singleton. Then $MS(i, j) = min(MS(i), MS(j))$. **Challenges:** support no longer anti-monotone. If $Support(milk, coke) = 0.015$ and $Support(milk, coke, broc) = 0.005$, and $MS(broc) = 0.001$, then the superset of an infreq could be freq. An itemset is **maximal frequent** if none of its immediate supersets are frequent. A frequent itemset is a **closed**

frequent itemset if none of its immediate supersets has the same support as the itemset. $Max \subseteq Closed \subseteq Freq$. Could also use depth first instead of bread(apriori). Maximal freq itemsets provide compact representation of freq itemsets. Closed provide minimal representation, don't lose support information. **Mining association rules:** [1. Frequent itemset generation. 2. Rule generation: generate high-confidence rules from each freq itemset, each rule is binary partitioning of a freq itemset. Want conf $\geq minConf$.] **Anti-monotone:** support of an itemset never exceeds support of its subsets. **Rule generation:** freq itemset L, find all non-empty subsets $f \subseteq L$ such that $f \rightarrow L - f$ satisfies minconf. $2^k - 2$ candidate association rules. Confidence of rules generated from same subset has anti-monotone: $c(ABC \rightarrow D) \leq c(AB \rightarrow CD) \leq c(A \rightarrow BCD)$. Anti-monotone wrt num items on rhs of rule. In apriori, candidate rule generated by merging two rules that share same prefix in rule target space. new target space is union of old, new domain is intersec of old. Prune everything below low confidence rule. Apply statistical test to determine interestingness of rule. Sequence is ordered list of elements(transactions) $s = \langle e_1 e_2 e_3 ... \rangle$ each element contains an unordered collection of events (items) $e_i = \{ i_1, ..., i_k \}$. Each element is a time or location. Length $|s|$ is num of elements. k-sequence means k events(items), not elements. Subsequence if order preserved and elements are subsets. **Support of subsequence** w is fraction of all data sequences that contain w. **Sequential pattern** is a frequent subsequence (support $\geq minsup$). Sequential pattern mining is finding all subseqs with support $\geq minsup$. $\binom{n}{k}$ k-subsequences can be extracted from a given sequence. **Generalized sequential pattern** (GSP) algo [1. Find all 1 element frequent sequences. 2. Repeat until no new freq seqs are found: {1. candidate generation: merge pairs of frequent subseqs found in the $k-1$th pass to generate candidate sequences with k items. 2. Candidate pruning: prune candidate k seqs that contain infreq $k-1$ seqs. 3. Find support of remaining. 4. Eliminate infreqs. }] **How to merge:** If $k = 2$, merging will product $\langle \{ i_1 \}, \{ i_2 \} \rangle$ and $\langle \{ i_1, i_2 \} \rangle$. If $k > 2$, need to have that if we remove the first event from w_1, it is the same as result from removing the last event from w_2. Resulting candidate is w_1 plus last event of w_2. If the last two events in w_2 belong to same element, the last event in w_2 becomes part of last element in w_1. Otherwise, last event in w_2 becomes separate element in new seq. **TEXT MINING.** Steps: 1. parsing, inputting. 2. Deriving patterns. 3. Evaluating, interpretting. Tasks: document categorization, text clustering, entity extraction, sentiment analysis, document summarization, entity relation modeling, keyword association analysis (find correlation of keywords that appear together often), similarity detection of documents, entity link analysis, sequence analysis (predict recurring event), anomaly detection, hypertext anaylsis. **Bag of words**: just collect frequencies. (shallow, limited amount of preserved info from orig text). NLP methods are deeper. Tasks: ambiguous POS or sense "design", "root". Anaphora: direct object association, presupposition (implications). **Info retrival problem:** find docs from keywords. **Text categorization:** manual methods often rule-based, not scalable, automatic use vector space so we can use classification methods, or prob, or generative model based, use Naive Bayes classifier. Vector space model is 1 hot word embeddings, one unique word per dimiension. Two ways to use frequency: term frequency: more occurrences means more relevance of that term, or inverse doc freq: less frequent among documents means more important (more special words). RAW TF: $f(t,d)$ is how many times term t appears in doc d. **Simple normalization:** $f(t,d) / \sum_t (f(t,d))$. **Double norm:** Divide $f(t,d)$ be the largest word frequency count in doc d. Formula: $DN = 1/2 + \frac{\frac{1}{2} \cdot f(t,d)}{\max\{ f(t',d) : t' \in d \}}$. Ranges from 1/2 to 1. **IDF weighting**: $IDF(t) = \log\left(\frac{n}{k}\right)$. Log is base 10, n is total number of docs, k is number of docs in which t appears. **TF-IDF**: $weight(t) = TF(t,d) \cdot IDF(t)$. Given a document we have a sequence of TF-IDF weights. We can find similarity using **dot product:** $Sim(D_i, D_j) = \sum_{t=1}^{N} w_{i,t} \cdot w_{j,t}$. Or Normalized dotproduct/cosine: $Sim(D_i, D_j) = \frac{\sum_{t=1}^{N} w_{i,t} \cdot w_{j,t}}{\sqrt{\sum (w_{i,t}^2)} \sqrt{\sum (w_{j,t}^2)}}$. **ANOMALY DETECTION** Types of problems: 1. find all data points with anomaly scores greater than t. 2. Find all data points having the top n largest anomaly scores. 3. Compute the anomaly score of a test point given a document. **Applications:** Credit card fraud, fraid, intrusion, fault. **Supervised anomaly detection:** need to have training dataset with lables anomaly or not. **unsup anomaly detection**: no class label. **Semi-supervised:** some data but not all is labeled. **assumption:** more "normal" than abnormal data. May be anomalous even if none of attributes are alone. May seem unusual wrt all objects, but not wrt neigborhood. **Masking:** presence of several anomalies masks presence of all. **Swamping:** normal objects labeled as anomalies since anomalies distort model. **Steps:** build profile of normal behavior, then detect. **Types:** Graphical (boxplot 1d, scatterplot 2d), spin plot(3d)) statsitical (choose a dist), or assume we have a majority and anomalous distribution, they partition space, so assume all belong to M majority, move them one by 1 to A and compute log likelihood before and after. If difference is greater than threshold, move to A permanently. , proximity(use NN distance), density (anomaly scores is inverse of density around object). Density approach 1: define density as reciprocal of avg dist to k NNs. Approach 2: define density as number of objects within a distance d of point. **Local Outlier factor**: Compute density of local neighborhood. LOF is ratio of avg density of NNs of point and density of point itself. Outliers have largest LOF. Cluster based: Discard small cluster far away, or discard smaller than some threhold, or assess degree to which a point belongs to any cluster. **RECOMMENDATION SYSTEMS. Content based sysetms:** examine properties of items recommended. **Collaborative filtering:** recommend items based on similarity measures betwen users or items or both. Items recommended are those liked by similar user. Formal model X set of customers, S set of items, $X \times S \rightarrow R$. R is set of ratings, totally ordered. Problems: utility matrix U sparse since most people have not rated most movies. Three approaches: content based, collaborative, latent factor. **Content based:** recommend items to customer x similar to previous items rated highly by x. For each item, create **item profile**, a set of features, like for movies, it would be author, title, actor,.... Create **user profiles** as well. these vectors have same componenets as item profiles, weighted avg of rated item pofiles. Take user ratings matrix, normalize by subtracting avg rating, and then compute user profile. Content based good because no need for other users, new and unpopular items can be recommended, explanations. Bad because hard to find features, bad for new users. **Collab filtering** cosine is dotproduct divided by the product of abs vals. Jaccard is intersection overunion. Prediction for item i of user x: $r_{xi} = \frac{\sum_{y \in N} s_{xy} r_{yi}}{\sum_{y \in N} s_{xy}}$ where N is set of k users most similar to x who rated i. **Item-item collab filter**: $r_{xi} = \frac{\sum_{j \in N(i,x)} s_{ij} r_{xi}}{\sum_{j \in N(i,x)} s_{ij}}$ item i. $N(i,x)$ set of items rated by x similar to i. RMSE is sqrt of sum of squres of diffs between true and preds. item item works better than user user, items are simpler. **Cold start**: not enough users, **sparsity:** hard to find users who rated same items, **firest rater** cannot recommend item that has never been rated **popularity bias**: recommends popular stuf to people with unique taste. **Latent factor**: assume we can approximate the rating matrix R as a product of two thin matrices factor matrix Q and transpose of user matrix P. USE SVD.

Chap 6

(2) (a) support for $\{ e \}$ is 8/10. Support for $\{ b, d \}$ is 2/10. Support for $\{ b, d, e \}$ is 2/10.

(b) confidence of first is 1. Confidence of second is 2/8. Not symmetric.

(c) Support is 4/5. Support is Support is 2/5. Support is 2/5.

(d) Confidence of first is 2/2. Confidence of second is 2/4.

(e) Confidence can increase or decrease, it increased in the above, but if every new basket that contains more than one X contains $X \cap Y$, then the numerator and denominator will decrease by the same amount, which will cause a decrease in confidence. Support can only increase.

(7) (a) We first list frequent 1-itemsets. They are:1,2,3,4,5. So we have:

(2)
$$\{ 1, 2, 3, 4 \}$$
$$\{ 1, 2, 4, 5 \}$$
$$\{ 1, 3, 4, 5 \}$$
$$\{ 2, 3, 4, 5 \}$$

(b) The candidates are the 4-itemsets generated by combining 3s with the same first 2 items when in order:

(3)
$$\{ 1, 2, 3, 4 \}$$
$$\{ 1, 2, 4, 5 \}$$
$$\{ 1, 3, 4, 5 \}$$
$$\{ 2, 3, 4, 5 \}$$

(c) First doesn't get pruned because all subsets of card 3 are frequent. Second gets pruned since $\{ 1, 4, 5 \}$ is not frequent. Third pruned for same reason. Last is pruned since $\{ 2, 4, 5 \}$ not freq. So only $\{ 1, 2, 3, 4 \}$ is left.

(8) Too easy.