Data Mining: Methodology and Algorithms

Ryan A. Rossi
Purdue University,
Department of Computer Science
rrossi@cs.purdue.edu

1 Introduction

1.1 Predictive Modeling (Classification and Regression)

Task Estimate a function $y = f(x, \theta)$ that maps observed x values to y values.

Parametric vs. Non-parametric Models: *Parametric models* take on a specific functional form/model structure (i.e., Binomial). The number of parameters is fixed in advance (naive bayes). *Non-parametric* models are driven by the data instead of taking an explict form (model structure is determined from data; classification tree, nearest neighbor).

Discrimitive vs. Generative Models: *Discrimitive* models the decision boundary directly (support vector machines, nearest neighbors, perceptrons, decision trees). *Generative* models the full joint probability distribution or class-conditional $P(X|Y)$ and class priors $P(Y)$ (naive bayes, logistic regression, probability estimation trees).

Ensemble techniques such as bagging (randomly resample m training examples from the initial m examples) and boosting (assign more weight to misclassified instances; constructs more difficult learning problems, final classifier is a weighted vote of the individual classifiers) reduce bias and variance.

Search *Set of states* defined by knowledge representation. *Set of search operators* define the actions to move in the state space. *Search algorithm* takes an input state, choice of actions (scored by function), stopping criterion.

Internal Scoring Function Associate score with state for use in search.

External Scoring Function (Evaluation): Measure quality of pattern/model. Apply an arbitrary score function such as Zero-one loss (accuracy,sensitivty,precision/recall/F1), absolute loss, squared loss (root mean squared error), Liklihood/conditional liklihood and AUC with cross-validation. Bias (complex models have lower bias; $(E[t]-E[y])^2$) and variance (lower variance if enough data to estimate parameters; $E[(E[y]-y)^2]$).

Evaluation Methodology Ensure disjoint training and test sets. Apply k-fold cross-validation to eliminate dependencies/bias between test sets (applied for model evaluation or to learn parameters). Learning curves (how does accuracy change with additional training data?), Paired t-test to assess significance.

Multiple comparison problems: *overfitting* (algorithms typically select the component with the maximum score (biased); instead we need to adjust for number of components), *oversearching* (search spaces consist of different numbers of models; maximum scored models are biased to differing degrees but most al-

gorithms directly compare scores), *attribute selection errors* (prefer attributes with many values). Adjust for multiple comparision problems by testing on withheld data to remove bias (new data, k-fold cross-validation), estimate distribution (sampling distribution) of max values accurately (randomization tests), adjust probability calculation (Bonferroni adjustment), or alter evaluation function to incorporate complexity penalty (MLD, BDL, AIC, bayesian priors).

1.2 Descriptive Modeling (Global summary/features)

Task: Density estimation attempts to compactly represent the full joint distribution $P(\mathbf{X}) = P(X_1, X_2, ...X_n)$.

Bayesian Networks Models for the overall probability distribution (*density estimation*) such as Bayesian networks (directed; $O(n \cdot 2^k)$ parameters where k is #parents) or Markov networks (undirected) where the graphical models can either be parametric or non-parametric. A Bayesian network specifies a joint probability distribution P given a directed acyclic graph G then the joint distribution is defined by the factorization $P(X_1, ..., X_m) = \prod_{i=1}^{m} P(X_i|Pa_i)$ where Pa_i is the parents of X_i and G is a minimal I-map of P (all the Markov assumptions implied by G are satisified by P; X_i is independent of the rest of the network given its parents, children, and chilren's parents [markov blanket]).

Learning parameters The parameters are estimated using MLE or MAP estimation as the liklihood decomposes according to the structure of the network and there exists a closed form solution.

Learning structure Apply a heuristic search (model space is all nodes and possible edges) with the search operators (add/delete/reverse edge) and evaluate using penalized liklihood with the score function (bayes rules). The possible search techniques are greedy-hill climbing, best-first search, and simulated annealing.

Task: Cluster analysis and segmentation (partitioning or probablistically assigning similar points into groups). The traditional algorithms are K-means (partition-based), Mixture models (probabilistic), hierarchical clustering (agglomerative [bottom-up] and divisive [top-down]; nearest neighbor clustering), spectral clustering (hierarchical-divisive).

K-means algorithm starts with k random centroids and 1) assigns points to closest centroid then 2) recomputes cluster centroids and repeats the two steps until no change in assignments (strengths: efficient $O(tkn)$ and spherical clusters, k-mode, k-medoids; weaknesses: local optimum/sensitive to initial seeds, mean must be defined, specify k clusters, not robust to outliers/noise).

Hierarchical methods construct nested clusters rather than preselecting k (dendogram depicts sequence of merges/splits and height indicates distance/similarity) by either merging clusters bottom-up (agglomerative) or dividing clusters top-down (divisive).

Agglomerative (bottom-up) algorithm merges clusters C_i and C_j with min distance/score (worst case complexity $O(n^2)$). The most popular distance/score functions are single-link/nearest neighbor (distance between the closest two points $x \in C_i$ and $y \in C_j$; long thin clusters), complete-link/furthest-neighbor (distance between the furthest points $x \in C_i$ and $y \in C_j$; sensitive to outliers), average-link (average of all distances between pairs of points, one from each cluster), between cluster (squared euclidean distance between the centroids of clusters).

Divisive (top-down) algorithm splits cluster C_k into C_i and C_j by applying a partition clustering method (k-means, spectral clustering) at each iteration. Spectral clustering cuts a weighted graph/similarity matrix into a number of disjoint groups by solving an eigensystem and applying a cut-objective function (strengths: finds non-spherical clusters; weaknesses: $O(n^3)$ for arbitrary eigensystem).

Probablistic clustering (mixture models) specifies a weighted combination of component distributions

(Gaussian, poisson, exponential). Learning involves computing $\{w_1, ...\}$ component distribution parameters (mixing coefficients/prior probabilities) using EM (Expectation (guess MLE θ parameters): $\delta_k(x) = p(k|x) = \frac{w_k N(x|\mu_k, \sum_k)}{\sum_{j=1}^{K} w_j N(x|\mu_j, \sum_j)}$ posterior probabilities; Maximization: (compute log-liklihood using predicted cluster memberships): $\log p(x, z|\theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \delta_k(x_n)[\log w_k + \log N(x_n|\mu, \sigma_k)]$)

Spectral Clustering Compute the eigenvectors of a weighted graph/similarity matrix and cut a specified (or multiple) eigenvectors into a number of disjoint groups. Strengths: non-spherical clusters. Weaknesses: $O(n^3)$ to find all eigenvalues of an arbitrary eigensystem.

Evaluation *Supervised cluster evaluation* measures the extent to which clusters match external class label values (classification/similarity oriented). We might also use clustering tendancy to determine if a given dataset has clusters without clustering. *Unsupervised evaluation* measures goodness of fit without class labels (Cohesion measures similarity of objects in a cluster; separation measures the distinctness of clusters; silhouette coefficient combines both cohesion and separation since $min(cohesion) = max(separation)$). For hierarchical clustering cophenetic distance matrix (agglomerative clustering) can be used for evaluation. Cluster evaluation is often subjective or relative depending on domain.

Density estimation is evaluated using goodness of fit measures ($S(\theta, M) = error(M) + penalty(M)$ where the penalty depends on the number of parameters p and the number of data points n) such as Akaike information critera (AIC; $S = -2\log L + 2p$) or the Bayesian information criterion (BIC; $-2\log L + p\log n$). Structure learning is evaluated by generating synthetic data from specified bayes net and then attempting to recover the structure.

1.3 Pattern Mining (Identify Local Features)

Association rules (and graph mining); find descriptive associations between variables.

1.4 Anomaly detection

Find data points that are considerably different from the remainder of the data. *Supervised* (classification with imbalanced classes), *Semi-supervised* (labels available only for normal data), *Unsupervised* (assume anomalies are very rare compared to normal data). Build a profile of normal behavior based on patterns or summary statistics for the overall population. Use deviations to detect anomalies. Types of methods: visual and statistical-based, **cluster-based** (points in small cluster candidate anomalies; compute distance between candidate points and non-candidate clusters; if far from other clusters then anomalies), **distance-based** (nearest neighbor; points with less than p neighboring points within distance d are anomalies), **density-based** (local outlier factor), **model-based** (mixture models: estimate probability distribution from data, if liklihood is less than threshold move point to anomaly distribution), **projection-based/spectral decomposition** (anomalies are clearly defined in a lower dimensional space).

Evaluation: ability to identify known anomalous instances or injected anomalies (random cases or profiled cases) or use internal measure to evaluate (density of cluster; circular).

1.5 Components of Data Mining Algorithms

Model or Pattern Structure The underlying structure or functional forms that we seek from the data.

Model is a global description of the data

Pattern is a local distinguishing property of the data

Scoring Function Judging the quality of a fitted model.

Optimization and Search Method Optimizing the score function and searching over different model and pattern structures.

Data Mining Algorithm Tuple {Task, Data representation, Knowledge representation, Learning/search/optimization (scoring function), Evaluation (scoring function)}

⟨ Name, Task, Structure, Score Function, Search Method ⟩

⟨ CART, Classification/Regression, Decision Tree, CV Loss Function, Greedy Search over Structures ⟩

⟨ Neural Network, Regression, Nonlinear functions, Squared Error, Gradient Descent on Parameters ⟩

⟨ A Priori, Rule Pattern Discovery, Association Rules, Support/Accuracy, Breadth-first w/ pruning ⟩

⟨ Support Vector Machines, Classification/Regression, Support Vectors, Accuracy, BF w/ pruning ⟩

2 Mathematical Background

Mean $E[X] = \sum_x x\ p(x)$

Variance $Var(X) = E[X - E[X]]^2 = E[X^2] - E[X]^2$

Linearity of expectation $E[X + Y] = E[X] + E[Y]$

Properties of Covariance $Cov(aX, bY) = abCov(X, Y)$, $Cov(X+a, Y+b) = Cov(X, Y)$, $Cov(aX+b, X) = aVar(X)$

Standard Deviation $\sigma^2 = \sqrt{Var(X)}$

Independence $P(A|B) = P(A)$ and $P(A, B) = P(A)P(B)$

Conditional Probability $P(A|B) = \frac{P(A,B)}{P(B)}$

Bayes Theorem $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

Common Distributions Bernoulli, Binomial, Multinomial, Poisson, Normal, ...

Hypothesis Testing $p(type\ I) = p(reject\ H_0 \mid H_0\ is\ valid)$, $p(type\ II) = p(accept\ H_0 \mid H_0\ is\ invalid)$, and $power\ = 1 - \beta$ where $beta$ is $p(type\ II)$

2.1 Estimation

Maximum Liklihood Estimation $P(\theta|X) = log \sum_i P(x_i|\theta)$

Maximum a posteriori $P(\theta|X) = log \sum_i P(x_i|\theta) + P(\theta)$

2.2 Scoring Functions

Scoring functions can be used to optimize the search inside the algorithm or evaluating the results outside of the algorithm and using this information as a means to guide the search.

Entropy $H[X] = -P(X)\log P(X)$

Conditional Entropy $H[Y|X] = \sum P(X = v)H[Y|X = v]$

Information Gain $IG[Y|X] = H[Y] - H[X|Y]$

Chi-square Test $\chi = \sum_i^k \frac{(o_i - e_i)^2}{e_i}$ calculates the normalized deviation of observed (predicted) values from expected (actual) values

Mean-Squared-Error $MSE = (E[\hat{\theta}] - \theta)^2 + E[\hat{\theta} - E[\hat{\theta}]]^2 = (Bias)^2 + Variance$

Outside Score Functions: Zero-one loss, accuracy, sensitivity/specificity, precision/recall/F1, Absolute loss, Squared loss, Root mean-squared error, liklihood/conditional liklihood, area under the ROC curve.

Sum of Squared Errors $S_{SSE} = \sum_i^n (y_i - \hat{y_i})^2$

2.3 Bias-Variance Analysis

Noise $E[(t - E[t])^2]$ Loss incurred independent of algorithm

Bias $(E[t] - E[y])^2$ Loss of mean prediction relative to optimal prediction

Variance $E[(E[y] - y)^2]$ Average loss of prediction compared to mean prediction

$$E[L_s q(t, y)] = E[(t - E[t])^2] + (E[t] - E[y])^2 + E[(E[y] - y)^2]$$

Bias is often related to the size of the model space; more complex models tend to have lower bias. If the size of the parameter space is large then lower bias and higher variance whereas if the size of the parameter space is small then lower variance and higher bias. Variance is related to the size of the dataset. If there is enough data to estimate accurately the parameters, then the model will have low variance. Simple models such as Naive bayes can perform surprisingly well due to lower variance.

2.4 Distances/metrics

Euclidean Distance $d(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$

Manhatten Distance $d(x, y) = \sum_i |x_i - y_i|$

Mahalanobis Distance $d(x, y) = \sqrt{(x_i - y_i)^T \sum^{-1} (x_i - y_i)}$

Covariance $d(x, y) = \sum_i (x_i - \bar{x})(y_i - \bar{y})$

Correlation $d(x, y) = \sum_i \frac{(x_i - \bar{x})(yi - \bar{y})}{\sigma_x \sigma_y} = \frac{cov(x,y)}{\sigma_x \sigma_y}$

3 Predictive Modeling

3.1 Parametric Models

Parametric models take on a specific functional form/model structure (i.e., Binomial). Number of parameters is fixed in advance. In essence, the data is fitted to the model versus fitting the model to the data. High bias if data does not follow a specific functional form/model structure (i.e., normal, binomial). Given a model

with a specified structure and a set of parameters θ; the aim is to estimate the parameters θ that maximize an appropriate score function measuring the fit of the model to the data.

Naive Bayes $P(C|X) = P(C)\prod_i P(X_i|C)$ Naive bayes can be formulated using distributions as a functional form (i.e., normal distribution) or simply using frequencies. Adjust for zero counts by smoothing (laplace).

Perceptron (neural network learning rule)

$$f(x) = \sum_{i=1}^{m} w_i x_i + b, \;\; where\ y = sign[f(x)]$$

3.2 Non-parametric Models

Non-parametric models are driven by the data instead of taking an explict form. Model structure is determined from data. These are data driven and depend on local model structures such as kernel functions that only depend on the local neighborhood of points. These local nieghborhood models scale poorly to high dimensionality. **Curse of dimensionality**: as the number of variables increases the number of data points required to obtain accurate estimates increases exponentially.

k-Nearest Neighbor Define distance measure and then choose k.

Decision Trees

Support Vector Machines Linear classifiers cannot deal with non-linear concepts and noisy data. Solutions: soft margin (allow mistakes in training data), network of simple linear classifiers (neural networks), also map data into richer feature space (e.g., non-linear features) and then use linear classifier. Kernel SVM is non-parametric classifier (keeps all data around in kernel).

3.3 Discrimitive Models

Models the decision boundary direction (possibly the conditional probability distribution $P(Y|X)$). However, samples cannot be generated from the joint distribution. These models often have hard decision boundaries. A few examples include Support Vector Machines, Neural networks.

3.4 Generative Models

Latent Dirchelet Allocation, Gaussian Mixture Model, ...

3.5 Evaluation

Overfitting 1) Regularization, 2) penalty term in classifier scoring function, 3) hold out evaluation set (used to adjust structure of learned models)

Cross-validation is used to eliminate dependencies between test sets. Cross validation is used for *parameter setting* and *model evaluation*.

3.6 Pathologies of Induction Algorithms

Overfitting Adding components to models that reduce performance or leave it unchanged.

Oversearching Selecting models with lower performance as the size of the search space grows.

Attribute Selection Errors Preferring attributes with many possible values despite lower performance.

Adjusting for multiple comparisons Remove bias by testing on withheld data (new data, cross-validation). Estimate sampling distribution accurately (randomization tests). Adjust probability calculation (Bonferroni adjustment). Alter evaluation function to incorporate complexity of penalty (MDL, BDL, AIC, Bayesian Priors).

4 Descriptive Modeling

Global Summary of data. Model main features of the data. The two main tasks are density estimation and clustering.

4.1 Density Estimation

4.1.1 Compact Representation: Bayesian Networks

Key Idea: use properties of independence to determine more compact representations. If X and Y are independent, then $P(X,Y) = P(X)P(Y)$ requiring O(2) parameters or O(m) where m is the number of variables.

Graphical Models (Bayesian (directed) or Markov (undirected) Networks. These models can be either parametric or non-parametric.

In Bayesian Networks, we have

$$P(X = x) = \prod_{i=1}^{m} P(X_i|Parents(X_i))$$

that is X_i is conditionally independent given the parents. This independence assumption reduces the amount of numbers from $O(2^k)$ to $O(n \times 2^k)$ where k is now much smaller.

This might also be stated each X is independent of the rest of the network given its Markov blanket. A markov blanket includes the parents of X, children, and the childrens parents.

If we know the structure and a choice of parametric family for $P(X_i|Pa_i)$ then we must learn the parameters of the model using maximum liklihood or MAP estimation to construct a network that is closest to the probability that generated the data.

$$L(\theta; D) = \prod_{j=1}^{m(var)} \prod_{i=1}^{n} P(x_j(i)|Pa_j(i)\theta) = \prod_{j=1}^{m} L_j(\theta_j; D)$$

Learning Structure Set of variables (nodes), need to learn the edges and parameters. *Adding an extra arc* increases the number of parameters to be fitted and forces a wrong assumption about causality and domain

structure. If a model is missing an arc it can never be compensated by accurate fitting of parameters. Also misses casuality and domain structure.

1. Constraint based: performs tests of conditional independence. Searches for a network that is consistent with the observed dependencies and independencies. *Strengths*: Intuitive, separate structure learning from the form of the independence tests *Weaknesses*: Sensitive to errors in individual tests.

2. Score based: Define a score to evaluate how well the independencies of a given structure match the observations. Search for a structure that maximizes the score. *Strengths*: Statistically motivated, can make compromises between fitting the data and complexity. *Weaknesses*: Computationally difficult.

   Score function using Bayes Rule: $P(G|D) = \frac{P(D|G)P(G)}{P(D)}$ where $P(G)$ is the prior over structures, $P(D|G)$ is the marginal likelihood and $P(D)$ is the probability of the data and can be ignored since it is the same when comparing structures.

Can be formulated as an optimization problem where the input is the training data, the possible structures (including prior knowledge about structures), and the score function (including priors). The output is a structure that maximizes the score function with respect to the inputs/training data. *Key property:* the score of a network is a sum of terms due to decomposable nature of the model.

Theorem Finding a maximal scoring network structure with at most k parents for each variable is NP-hard for $k > 1$. Solved by using heuristic search where we consider the model space (all nodes, set of possible edges), search operators (add edge, delete edge, reverse edge[modify 2 cpds]), evaluation (penalized liklihood: penalize complexity of structure, AIC, BIC; more edges more likely) and apply a search technique (greedy hill-climbing, best-first search, simulated annealing).

Greedy hill climbing Start with an initial empty network, best tree, or random network. At each step: 1. evaluate all possible changes, 2. apply change that leads to best improvement in score, 3. repeat, stop when no modification improves score. Each step requires evaluating $O(m^2)$ new changes. *Problems:* greedy-hill climbing gets stuck in local maxima (all one edge changes reduce score), plateaus (some one edge changes leave the score unchanged, equivalent networks are neighbors in the search space and receive the same score). Standard heuristics can escape both, random restarts and TABU search.

Inference Each network describes a unique probability distribution P. Inference refers to the process of computing answers to probability queries. There are many types of queries, most of which involve evidence. An evidence e is an assignment of values to a set of E variables in the domain. The simplest query is the probability of observing the evidence (computed by marginalizing the other variables out)

$$P(e) = \sum_{x_1} \cdots \sum_{x_k} P(x_1, ..., x_k, e)$$

We could also be interested in the conditional probability of a variable given the evidence

$$P(X|e) = \frac{P(X, e)}{P(e)}$$

Prediction: probability of an outcome given the starting condition (target is descendant).

Diagnosis: probability of disease given the symptoms? (target is ancestor)

Direction between the variables does not restrict the directions of the queries (bayes rule).

Sampling

4.1.2 Parametric Density Estimation

Models takes on a specific functional form such as the normal distribution, exponential distribution, etc... Parametric models can often be characterized by a relatively small number of parameters.

4.1.3 Nonparametric Density Estimation

Kernel estimators, histograms,...

4.2 Clustering

Algorithms capture clusters of different shapes. Therefore the choice of algorithm depends on the application/objectives.

Every clustering technique assumes $D = \{x_1, ..., x_n\}$ and outputs k clusters $C = \{C_1, ..., C_k\}$ such that x_i is assigned to a unique C_j.

A score function is used for evaluation such that $S(C, D)$ is maximized/minimized (minimize within cluster distance or maximize between cluster distance).

4.2.1 K-means (partition-based)

Representation: canonical item description(s)
Algorithm: Randomly select k centroids.
(1) Compute the distance/score between every point x_i and each centroid C_k and assign x_i to the C_k that minimizes/maximizes the score function (closest).
(2) After assigning the points X to clusters then recompute the centroids (since they possibly changed). Stop when converged (no changes in cluster assignments).

Strengths Efficient $O(tkn)$, finds spherical clusters
Weaknesses Terminates at local optimum (sensitive to initial seeds), applicable only when mean is defined, need to specify k, susceptible to outliers/noise.

$$P(X|C) = \sum_{k=1}^{K} \sum_{x_i \in c_k} ||x_i - c_k||^2$$

4.2.2 Mixture models (probabilistic model-based)

Representation: Mixture components (parameters and weights)

Assumes a probablistic model for each underlying cluster. A mixture model specifies a weighted combination of component distributions. $f(x) = \sum_{k=1}^{K} w_k f_k(x; \theta)$

$$P(X|C) = \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_j P(x_i|C_k)$$

Difficult to represent parametric, if you don't know 3 gaussians then nonparametric might be better.

4.2.3 K-means and Gaussian Mixture Models

There is a strong connection between K-means and GMM. Consider a GMM with common spherical covariance matrix $\sum = \sigma^2 I$. It is easily shown that as $\sigma \to 0$ the weights δ_{ik} for each data point i in the E-step go to 1 for one particular k and 0 for all other ks. In this restriction GMM is equivalent to K-means.

4.2.4 Hierarchical methods

Construct a hierarchy of nested clusters rather than picking k beforehand. Two main approaches, agglomerative merges clusters successively (bottom-up) or divisive (dividing) clusters successively (top-down). The complexity is $O(n^2)$.

Dendogram depicts sequences of merges or splits and height indicates distance.

Agglomerative clustering starts with one point clusters and successively merges the clusters that minimize the distance function (Nearest neighbor clustering)

Divisive clustering starts with a single cluster containing all points and successively splits the clusters by applying a partition-based method (k-means, spectral clustering).

Spectral Clustering Cut a weighted graph into a number of disjoint groups using a similarity matrix M. Strengths: finds non-spherical clusters. Weaknesses: $O(n^3)$ to find all eigenvalues of an arbitrary eigensystem.