Machine Learning

In Son Zeng

University of Michigan

insonz@umich.edu

February 6, 2019

Overview

1. Concept overview
   - Pattern Recognition
   - Data Mining

2. Concept overview
   - PCA and FA

Pattern Recognition: Introduction

1. Pattern Recognition is the recognition of patterns and regularities in data. It is closely related to artificial intelligence, machine learning (ML), data mining and knowledge discovery in databases (KDD).
2. Machine Learning is one approach to pattern recognition. Some other methods for pattern recognition involve hand-crafted but not learned rules and heuristics.
3. Pattern Recognition is a path to artificial intelligence. Some other approaches to artificial intelligence include symbolic artificial intelligence.
4. Definition: The field of pattern recognition is concerned with the automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions such as classifying the data into different categories.

Pattern Recognition: Introduction

1. Pattern recognition systems are often trained from labeled ”training data”.

Data Mining

1. Data mining is a process to structure the raw data and formulate or recognize the various patterns in the data through the mathematical and computational algorithms.
2. Data mining helps to generate new information and unearth various insights in a data.
3. Data mining has two types: 1) Descriptive - give information about existing data about the structure, 2) Predictive - make forecasts based on the data.
4. Data mining also helps explore trends from the data (one kind of pattern discovery task). Therefore, it requires advanced components of artificial intelligence, pattern distribution and statistics.

Data Analytics

1. Data Analytics is a superset of data mining. It involves extracting, cleaning, transforming, modeling and visualization of data with an intention to uncover meaningful and useful information that can help in deriving conclusion and take decisions.
2. Data analytics is the process of ordering and organizing raw data to determine useful insights and decisions.
3. Data analytics requires the knowledge of CS, statistics, maths, AI/Machine learning and subject knowledge.
4. Data analytics is of several types: exploratory, descriptive, text analytics, predictive analysis, data mining
5. Data analytics can work on both structured, semi-structured and unstructured data

Data Mining vs Data Analytics Responsibilities

1. A Data mining specialist usually builds algorithms to identify meaningful structure in the data.
2. A data analyst usually cannot be a single person. The job profile involves preparation of raw data, its cleansing, transformation and modeling and finally its presentation in the form of chart/non-chart-based visualizations.
3. Data mining is responsible for etracting and discovering meaningful patterns and structure in the data.
4. Data analytics is responsible for developing models, explanations, testing and proposing hypotheses using analytical tools and methods.
5. The output of a data mining task is a data pattern.
6. The output of data analysis is a veried hypothesis or insight on the data.

Natural Language Processing: Problems

1. Case-sensitivity and abbreviation problems are common in electronic health records (EHRs) and patient history records. We need strong NLP system to make corresponding match of abbreviations to the context.
2. NLP helps transform the unstructured data to structured database, which in turn increases the interoperability of EHR and patient data. Interoperability, the level of clarity of delivering message through healthcare stakeholders, is crucial for providing effective healthcare.
3.

FIxed Effect Model

1. $y_i = \alpha + X \cdot \beta$
2. Each individual has same regression effects.
3. We add a random effect for each FIPS because regression effects cannot represent the housing price disparity by FIPS (geographical difference)
4. For each of the FIPS (county), we have an additional intercept called **random intercept**. We already have regression (fixed effect), this random effect gives additional information based on the geographical differences.
5. Now we get each random intercept $\alpha_i \cdot 1(FIPS = i)$. And we also get the output **standard deviation/variance** across all random effects. If we would like to cover as much as the variability of geographical differences, say 95% or 99.5 %, we need to plus and minus 2 or 3 times the standard deviation.

PCA Functions

1. R: prcomp(), princomp(): return all parts of PCA
2. R: loadings(): display variable loading for first n PCs.
3. R: eigen(), svd(): perform eigen-decomposition, SVD of any matrix
4. R: biplot(), ggbiplot(): make a PCA biplot out of an object created by prcomp
5. R package factoextra: fviz_fig() - nice formatted scree plot; fviz_pca_var(): variable plot; fviz_pca_biplot(): biplot
6. We use PCA to find the largest p eigenvalues and eigenvectors of S to capture 95% variability, reducing the dimension.

PCA Functions

1. Goal: Project a vector in q dimensions to p dimensions, where $p << q$.
2. Notation: $y_{q\times 1} \approx u_1 x_1 + ... + u_p x_p$, where $u^T u = I_{p\times p}$, orthonormal.
3. We find u_1 such that the empirical variance of $x_1, ..., x_N$ given by $\frac{1}{N-1}\sum_{i=1}^{N}(u_1^T y_i - u_1^T \bar{y})^2$ is maximized.
4. Algorithm:
   1) Maximize $u_1^T S u_1, u_1 \in R^q$, such that $u_1^T u_1 = 1$.
   2) Introduce Lagrange Multiplier λ so
   $max u_1^T S u_1 - \lambda(u - 1^T u_1 - 1) \rightarrow S u_1 = \lambda u_1$
   3) Therefore, λ is an eigenvalue of S, and u_1 is the corresponding eigenvector of S.
   4) We in short find $max \lambda$, the largest eigenvalue of S, with u_1 being the corresponding eigenvector.

FA Functions

1. R:
2. Factors are **latent**.
3. Goal: Find $\hat{\Sigma} \approx \hat{\Lambda}\hat{\Lambda}^T + \hat{\Phi}$, one approach is $min||\hat{\Sigma} - \hat{\Lambda}\hat{\Lambda}^T + \hat{\Phi}||_F$
4. Algorithm: Start with initial estimate of $\hat{\Phi}$, we iterate until convergence
   1) Apply PCA to $\hat{\Sigma} - \hat{\Phi}$, K largeest eigenvalues to obtain $\hat{Lambda}$
   2) Update $\hat{\Phi} = diag(\hat{\Sigma} - \hat{\Lambda}\hat{\Lambda}^T)$
5. Algorithm 2: MLE method through EM algorithm.

Important FA Detail

1. Assume all p variables are quantitative. Assume X satisfies $X_j = \sum_{k=1}^{K} \lambda_{jk} F_k + u_j$. Matrix: $X_{p\times 1} = \Lambda_{p\times K} F_{K\times 1} + U_{p\times 1}$
2. λ_{jk} is factor loading; Λ, F, U are unobserved.
3. Assumptions: 1) F and U are independent; 2) $E(F) = E(U) = 0$; 3) $Cov(F) = I$; 4) $Cov(U) = \Phi_{p\times p}$
4. Covariance decomposition: Let $\Sigma = Var(X) \rightarrow \Sigma = \Lambda\Lambda^T + \Phi$
5. Diagonal terms of Σ are $Var(X_j) = \sum_{k=1}^{K} \lambda_{jk}^2 = \Phi_{jj}$ = communality + uniqueness
6. Degree of freedom: $d = \frac{p(p+1)}{2} - (pK + p - \frac{K(K-1)}{2})$. If $d < 0$, the model cannot be fitted

Important FA Detail

1. Factor rotations: For any orthogonal matrix T, $\Sigma = \Lambda\Lambda^T + \Phi = (\Lambda T)(\Lambda T)^T + \Phi$
2. EM algorithm: 1) The E-step , 2) The M-step updates the variable based on maximized likelihood (local)

Multi-dimensional Scaling

1. R: cmdscale();
2. Double-centering trick: $G = -\frac{1}{2}(I_n - \frac{1}{n}11^T)D(I_n - \frac{1}{n}11^T)$
3. Distance functions:
   1) R: gower.dist() -
   2) R:

Examples for classification

1. Identifying tumor cells as benign or malignant
2. Classifying credit card transactions as legitimate or fraudulent
3. Classifying secondary structures of protein as alpha-helix, beta-sheet or random coil
4. Categorizing news stories as finance, weather, or sports, etc.
5. Filtering spams emails out of user's mailbox
6. Recognizing handwritten letters and digits as one of A,B,C...,0,1,...,9
7. Recognizing animal pictures: is this a dog? is this a koala?
8. Assign people to either guilty or crime-free.
9. We need to find **decision rule** to make prediction and draw **boundary** to separate the classes as accurate as possible.

Examples of visualization for classification

1. R: require(ggplot2); ggpairs() - Scatter plot for data
2. R: require(GGally); ggparcoord() - Parallel coordinates plot for data. This works well if the number of class is small, but poorly if the number of class is large.
3.

Methods to Split Test and Train Dataset

1. For example, we want 75% of the sample size to be training set.
2. R Method 1:
   1) smpsize = floor(0.75 * nrow(data)); set.seed(123)
   2) trainindex = sample(seq_len(nrow(data)), size = smpsize)
   3) train = data[trainindex,]; test = data[-trainindex,]
3. R Method 2:

Cross Validation (CV)

1. Divide the data into V parts (V-fold cross-validation). V can be anything; popular values are 5, 10, 15 and n.
2. Hold out one of the parts to be the **test data**.
3. Use all remaining data as **training data** to develop a classification (clustering) rule.
4. Apply the classification rule to the held-out part and compute the error rate (**test error**).
5. Repeat (using for loop) steps 2-4 for each of the V parts to ensure that each partition of the data become the test data once
6. Average all test errors to obtain the **cross-validation error**.
7. The cross-validation error tends to be closer to the **true error rate** than to the apparent error rate???
8. The computational cost of CV can become a concern, particularly if there are many **tuning parameters**.

Brute force approach

1.
2.

Generative models for classification

1. If we know P(X,Y), we are done. In modeling, we need 1) Prior class prob $Y \sim \pi$ and class conditional distributions $X|Y = k \sim p_k$
2. Parametric model: $p_k(x) := p_k(x|\theta_k) \rightarrow \hat{p}(Y = k|X = x) \propto \hat{\pi}_k p_k(x|\hat{\theta}_k)$
3. Suppose $X|Y \sim p_k = N(\mu_k, \sigma_k)$
4. Therefore, the optimal classifier is the maximum of k classes for f(x) $f(x) = argmax_k P(Y = k) \cdot p_k(x) = argmax_k \pi_k \cdot N(x|\mu_k, \Sigma_k)$
   Take log we get QDA rule:
   $argmax_k \delta_k = argmax_k \left(log(\pi_k) - \frac{1}{2} log|\Sigma_k| - \frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) \right)$
5. For LDA (assume equal Σ), the function is simplified as $\delta_k = \left(log(\pi_k) - \frac{1}{2}\mu_k^T \Sigma^{-1}\mu_k - x^T\Sigma^{-1}\mu_k \right)$, with the quadratic term eliminated.

Logistic Regression

1. Maximize the conditional likelihood using multinomial likelihood with probability $P(Y = k|X)$
2. R: glm(formula, family = binomial) - logistic regression
3. R: require(nnet); multinom() - perform multinomial logistic regression
4. R: update(model, . -var) - delete variables
5. R: predict(model, type="response") - predict accuracy in training data; ifelse() to separate response class
6. R: predict(model, newdata= test, type="response") - predict accuracy in test data
7. Logistic regression does not work properly if the response classes are fully separated from each other

Linear Discriminant Analysis

1. Assumptions: 1) Equal Variance for every class, 2) data from each class follow normal distribution $N(\mu_k, \Sigma^2)$
2. Maximize the full joint likelihood based on the joint density $P(x, Y = k) = \pi_k \cdot \phi(x; \mu_k, \Sigma)$
3. R: require(MASS); qda(): perform LDA
4. R: predict(lda.model, data=train) - predict accuracy in training data
5. R: predict(lda.model, newdata=test) - predict accuracy in test data
6. LDA discriminant direction vector is $\Sigma^{-1}(\mu_k - \mu_l)$, the only term including data x.
7. Mahalanobis distance: $d_M(x, \mu_k)^2 = (x - \mu_k)^T \Sigma^{-1} (x - \mu_k)$
8. LDA is not robust, easily influenced by outliers; Logistic regression is more robust. LDA has more assumptions that Logistic regression.

Quadratic Discriminant Analysis

1. Assumptions: 1) Different Variance for every class, 2) data from each class follow normal distribution $N(\mu_k, \Sigma_k^2)$
2. R: require(MASS); lda(): perform LDA
3. R: predict(lda.model, data=train) - predict accuracy in training data
4. R: predict(lda.model, newdata=test) - predict accuracy in test data
5. R: require(caret); train(); confusionMatrix() - Derive the confusion matrix for accuracy

K nearest neighbor classifier (KNN)

1. R: require(class); knn(): Perform k-nearest neighbour classification for test set from training set. For each row of the test set, the k nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote, with ties broken at random. If there are ties for the kth nearest vector, all candidates are included in the vote.
2. R: knn.cv() - Perform k-nearest neighbour cross-validatory classification from training set. This uses leave-one-out cross validation.
3. k-NN is an adapted kernel method, a kernel method depending on the data. In the nearest neighbor kernel, K is constant over the nearest k data points, and zero otherwise. As such, k-NN is a local method which relies on the rule not changing too dast in a neighborhood.

K nearest neighbor classifier (KNN)

1. Algorithm: Given n data pairs $(x_i, y_i)_{i=1}^n$, we first 1) determine the k nearest neighbours of the new point x among the x_i by (default) using Euclidean distance. Then, 2) we assign x to the class corresponding to the majority votes of the k nearest neighbours. We often choose odd k to avoid ties.
2. Strength: Euclidean distance is convenient and popular for quantitative predictors; k-NN has no parametric assumption.
3. Shortcoming: Curse of dimensionality: k-NN fails in high dimensions since data points will be very sparsely population in high dimensions. In other words, it is difficult to find enough observations close to a target point x.
4. Indeed, since the volume of d-dimensional object with same length l is $\rho = l^d \rightarrow l = \rho^{1/d}$

K nearest neighbor classifier (KNN)

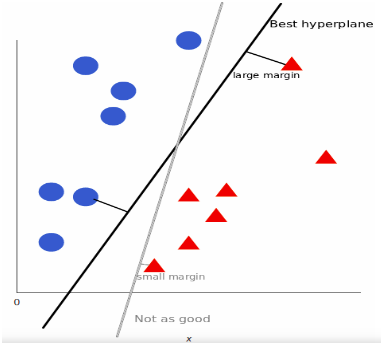
1. Distance function: Euclidean distance is popular, but weighted distances have been widely used to emphasize some variables over others.
2. We may or may not have an explicit classification rule in k-NN.
3. k-NN approximates the conditional probability $P(Y = c|X = x)$ directly. When $n, k \to \infty, \frac{k}{n} \to 0$, then k-NN estimate is consistent.
4. The effective number of parameters is $\frac{n}{k}$. So, the degree of freedom of k-NN highly depends on the choice of k (the smaller k, the more complex the model).
5. When $k = 1$, the training error is 0, but it suffers from overfitting.
6. We should choose k to minimize the mis-classification error on a separate data set, or use CV. Ideally, we use validation data to choose the k minimizing the error, and take an independent test set to estimate test error.

How KNN works (Example)

1. Step 1 : Calculate Similarity based on distance function
2. Euclidean Distance: $d(x,y) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2}$
3. Manhattan Distance: $d(x,y) = \sum_{i=1}^{m}|x_i - y_i|$

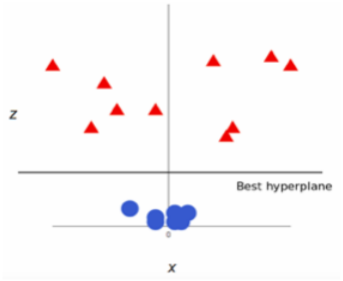
Support Vector Machine (SVM)

1. R: library(e1071); svm() - perform support vector machine
2. R:
3. SVM works for linear data, with hyperplane separating data with different class, with a maximized margin of $\frac{2}{||\beta||}$. Graphical interpretation of SVM objective:

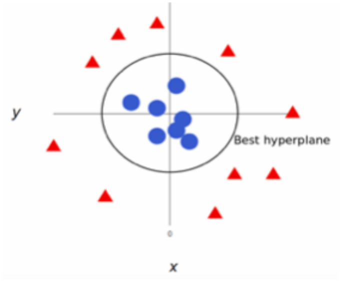


Support Vector Machine (SVM)

1. SVM also works for data with clearly segregated vectors. We can add a third dimension or even more. For example, the next example shows $z = 1 = x^2 + y^2$ is the best separating hyperplane. Again, the margin is still maximized as $\frac{2}{||\beta||}$.
2. By mapping our space to a higher dimension, we can classify nonlinear data using SVM.



3-Dimensional Representation



2-Dimensional Representation

Support Vector Machine (SVM)

1. Calculating transformation function is computationally intensive. SVM does not need the actual vectors to work, so **Kernel function** is useful here.
2. Common types of kernel functions to separate non-linear data:
   1) Polynomial kernels
   2) Radial basis kernels
   3) Linear kernels

Support Vector Machine (SVM)

1. Advantages:

   • High dimensionality: SVM is effective tool in high-dimensional space, useful for **document classification** and **sentiment analysis**, where dimensionality can be extremely large

   • Memory Efficiency: Since only a subset of the training points are used in the actual decision process of assigning new members, just there points need to be stored in memory when making decisions

   • Versatility: The ability to apply new kernels allows substantial flexibility for the decision boundaries, leading to greater classification performance

2. Disadvantages:

   • Kernel Parameters Selection: SVMs are very sensitive to the choice of the kernel parameters. In situations where the number of features for each object exceeds the number of training data samples, SVMs can perform poorly.

   • Non-Probabilistic: Since the classifier works by placing objects

Support Vector Machine (SVM)

1.

Decision Tree

1. R: library(rpart); rpart() - build decision tree
2. R: library(rpart.plot); rpart.plot() - plot decision tree output
3. R: library(party); ctree(); plot() - build and plot decision tree
4. R: rpart.control(minsplit, minbucket, maxdepth) - Tune the maximum depth, minimum number of sample a node must have before it can split, and minimum number of sample a leaf node must have
5. Gini impurity index:

Random Forest (RF)

1. R: library(randomForest); randomForest()
2. importance(); varImpPlot() - check variable importance
3.