**Introduction to Machine Learning**

***Definition:*** Use of statistical methods to give computers the ability to learn and make predictions using given data sets, without being explicitly programmed.

**Supervised Classification/Learning**: Segmentation of data (spectral domain) into regions that can be associated with patterns (or classes of interest) for a particular application. Commonly used for image recognition (i.e. facial recognition) and making recommendations (of songs, videos, movies).

-Features (input characteristics) and Labels (output classification)

**Decision Surface (DS):** Barrier that separates two groups of data points with different characteristics (data 🡪 decision surface)

-In general, fit the training data for features (X) and labels (y), and test the data on test data for X.

**Supervised Classification Algorithms**

**Naïve Bayes (NB):** Algorithm for determining the decision surface by comparing only the joint probabilities of all elements in each case, or the frequency of each element in a sample (naïve because it does not consider the order of which the elements appear)

-Advantages: Simple to implement, efficient

-Disadvantages: Breaks in cases when order of elements matter (i.e. word phrases)

-Examples: Text identification

-Python libraries needed: Scikit-learn (sk-learn), Numpy, Scipy, iPython, Matplotlib

-Scikit-learn: <http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html>

**Bayes Rule:** P(A|B) = P(B|A) \* P(A) / P(B) (used for text learning, or learning from documents)

-Sensitivity (correctly identify in a positive state) and specificity (correctly identify in a negative state)

-Prior probability \* Test evidence = Posterior probability (joint probability)

**Support Vector Machines (SVMs):**

-Find a separating line (hyperplane) between two classes (linear and nonlinear)

-Input: data; Output: separator/hyperplane

-Best separator: maximizes distance between the line and its nearest points, relative to both classes

-Takes care of outliers automatically

-Nonlinear: Add a nonlinear feature (i.e. x2+y2, |x|, |y|)

-Scikit-learn: <http://scikit-learn.org/stable/modules/svm.html>

**Kernel Trick:** Map a data set with low dimensional space (not separable) to high dimensional space (separable), which in return, creates a nonlinear separation in the original data set.

-kernel: ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’ or a callable; Default: ‘rbf’

-C is 1 by default and it’s a reasonable default choice. If you have a lot of noisy observations you should decrease it. It corresponds to regularize more the estimation

-gamma: defines how far the influence of a single training example reaches. Low values: far (decision boundary more ragged); High values: close (decision boundary more linear)

-Advantages: Work well in complicated domains with a clear margin of separation

-Disadvantages: Don’t work well in very large data sets and data with lots of noise, prone to overfitting

**Overfitting:** Overly complex decision boundaries that do not generalize the data well (usually results from overly complex data features and decision boundaries)

**Decision Trees (DTs):**

-Nonlinear decision surfaces with linear decision surfaces

-Yes or no based on each data parameter; If yes, move on to next decision (or match data point if at the end of the tree), if no, disregard data point (or move on to a corresponding decision)

-Scikit-learn: <http://scikit-learn.org/stable/modules/tree.html>

-min\_samples\_split: default = 2; lower number 🡪 more and complex splits (prone to overfitting)

-criterion: default = ‘gini’ (another measure of impurity); can be changed to ‘entropy’

**Entropy:** Measure of impurity in a data set; controls how a DT decides where to split the data

-When making a DT, choose variables that generate decision surfaces which give the lowest impurity in each region/subset of the data.

-pi is the fraction of examples in class i

-Sum all classes available

-All examples in the same class: Entropy = 0.0

-Examples are evenly split between all classes: Entropy = 1.0

**Information Gain:** Entropy (parent) – [weighted average] Entropy (children)

-Decision tree algorithm maximizes information gain

**Bias and Variance:** High bias 🡪 little learning; High variance 🡪 too responsive to the trained data (need a bias-variance tradeoff)

-Advantages: Easy to implement; ability to understand the data a lot better (than SVMs)

-Disadvantages: Prone to overfitting, especially for data with many features; picking the right parameters for the decision tree

**K Nearest Neighbors:**

-Given a data point, take the k nearest data points (neighbors) and group them (by computing Euclidian distances on the coordinate system).

-Count the number of each type of surrounding data points. The selected data point would most likely be the type of data point with the highest count.

-Choosing k: Small k 🡪 More ragged decision surface, but prone to overfitting; Large k 🡪 Smooth decision surface, but too generalized (low accuracy)

-k = 10 is optimal

-Scikit-learn: <http://scikit-learn.org/stable/modules/neighbors.html>

-Advantages: Simple to implement, easy to interpret output, fast runtime

-Disadvantages: Prone to overfitting

**Ensemble methods:** Meta-classifiers built from many (usually) decision trees

-Scikit-learn: <http://scikit-learn.org/stable/modules/ensemble.html>

**Boosting**: An ensemble technique that attempts to create a strong classifier from several weak ones.

-Build a model from training data, then creates a second model that tries to correct the errors from the first one. More models are added until error becomes small enough.

**AdaBoost (Boosted Decision Tree):**

-Best used to increase performance of decision trees on binary classification problems; although it can be used to boost any machine learning algorithm

-Most commonly used with decision trees with only one level (decision stump, which only gives 1 or -1)

-Classify data using decision stump 🡪 calculate misclassification rate ((correct – N) / N)

-Find weighted sum of misclassification rate:

-Calculate stage value:

-More accurate models have more weight or contribution to the final prediction

-More weight to incorrectly predicted instances, and less weight to correctly predicted instances

-Predicting: For the weak classifier (outputs only 1 or -1), the predicted values are weighted by the stage value. Take the sum of all weighted predictions: if sum > 0, then first choice (1) is inferred; if sum < 0, then second choice (-1) is inferred.

-Sk-learn: <http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html>

-Advantages: Easy to interpret data

-Disadvantages: Doesn’t fare well for outliers and noisy data; need to prepare good quality data

**Random Forest:**

-Builds multiple decision trees and merges them together to get a more accurate and stable prediction

-Random subset of the features is considered for splitting a node, sometimes using random thresholds

-Drop the data features that are too marginal or are not important enough

-n\_estimators: Number of trees the algorithm builds before taking the maximum voting or taking averages of predictions; higher number increases performance but is also slower.

-max\_features: Max. number of features Random Forest can try in an individual tree

-min\_sample\_leaf: The minimum number of leaves that are required to split an internal node

-Sk-learn: <http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

-Advantages: Easy to measure the relative importance of each feature on the prediction; Hard to overfit; Default parameters give good results

-Disadvantages: May be slow and ineffective for real time predictions (long computation time)

-Applications:

**Selecting the right algorithm:**

1. Understanding the problem (is it supervised classification?)

2. Checking and refining the data

3. Research what the algorithm does and have a general understanding

4. Find and read documentation (sk-learn)

5. Implement and train using training data (for all data, take 90% to train, and remaining 10% to test)

6. Make predictions on test data

**Data Sets**

-Person of Interest (POI): Stakeholder

-Training set size: Diminishing returns when the number of data points in the training set increases; approaches 100% or a theoretically possible maximum.

-More data > Having a good ML algorithm

-Types: numerical, categorical, time series, text

Case Study: Enron Fraud

Dataset: <https://www.cs.cmu.edu/~enron/>

**Discrete Supervised Learning:** Arbitrary/continuous inputs, discrete outputs

**Continuous Supervised Learning:** Arbitrary/continuous inputs, continuous outputs

-Discrete outputs can be turned into continuous outputs by measuring values instead of categorizing, and vise versa.

**Regressions:**

-Use functions (linear, quadratic, logarithmic, etc.) to model data regressions

-Linear: Slope and intercept (y = mx + b)

Linear Regression: <http://scikit-learn.org/stable/modules/linear_model.html>

**Least Squares Method**:

-Minimizes sum of squares of errors

-Find m and b such that sum of squares of errors is minimized

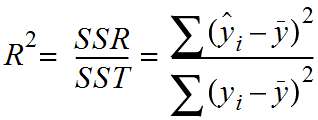
-Ordinary least squares (OLS): sk-learn

-Gradient descent

-However, more points = higher error (which need to be minimized), but more points also give a “better” fit.

**R Squared Method:**

-r2, the closer to 1 the better (maximum possible value is 1)



|  |  |  |
| --- | --- | --- |
| **Property** | **Supervised Classification** | **Regression** |
| Output type:  Trying to find:  Evaluation: | Discrete (category)  Decision boundary  Accuracy | Continuous (number)  Best fit line  Sum of squared error or r2 |

**Multivariate Regression:** Regression with more than 2 variables (pick the right variables)

**Outliers:** Data points that lay significantly outside of the general trend

-Causes: Sensor malfunction, entry error (human error), freak / unexpected event

-Detection: Train and fit the data 🡪 remove the outliers (~10%) 🡪 retrain the data with reduced dataset

-Rejection: Discard outliers, or in some cases (such as fraud detection, finding anomalies), discard normal data points

**Unsupervised Learning**

-Draw inferences from datasets consisting of input data without labeled responses

**Clustering (K-Means):**

-Group data points into clusters by assigning centroids

-Assign: Randomly draw cluster centers (centroids) on the data plane

-Optimize: Minimize total quadratic distance between points and the centroids

-Reposition centroids according to calculation, recalculate, and repeat until optimal results are achieved.

-Clustering depends on the starting/initial points of the centroids, so may want to re-run the algorithm several times and get an ensemble result.

-Visualizing: <http://www.naftaliharris.com/blog/visualizing-k-means-clustering/>

-Sk-learn: <http://scikit-learn.org/stable/modules/clustering.html>

<http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans>

-Observe and set n\_clusters to the appropriate number

-n\_init: Default = 10; number of times the algorithm is initialized/re-run

-Make sure to observe the data visually and set initial cluster centers, or they will end up at local minima and not the optimal locations. Therefore, the algorithm needs to be re-run multiple times to prevent centroids being assigned at stable local minima.

**Dimensionality Reduction:**

-Create a parameter that is a weighted sum of various features of the data

-Max-min scale:

-Sk-learn: <http://scikit-learn.org/stable/modules/preprocessing.html>

<http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html>

**Learning from Text**

-Bag of words: Count the frequency of each word appearance in every text input

-Sk-learn: CountVectorizer

-Not all words are equal (some words contain more information than others)

**Stopwords** = [the, in, for, you, will, have, be, etc.]

stopwords.words('english')

**Stemmer**: Group similar words (i.e. unresponsive, response, responsivity, responsiveness, respond) and represent them as their root word (‘respon’)

Stemmer = SnowballStemmer(‘english’)

Stemmer.stem(‘word’)

**TfIdf Representation:**

-Tf: Term frequency (bag of words)

-Idf: Inverse document frequency (weighting by how often word occurs in corpus)

-Weight rare words higher than common words

-sklearn.feature\_extraction.text.TfidfVectorizer()

<http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html>

**Feature Selection**

-Use the least number of features to gain the most amount of information

-Choose the best data features

-Check for bugs before implementing a new feature (be aware if classifier returns 100% accuracy)

-Useful in text learning (since there are many features)

**Univariate feature selection**: Treats each feature independently and asks how much power it gives you in classifying or regressing

-SelectPercentile: Selects the X% of features that are most powerful (where X is a parameter)

-SelectKBest: Selects the K features that are most powerful (where K is a parameter)

-Sk-learn: <http://scikit-learn.org/stable/modules/feature_selection.html>

-Bias vs. Variance:  
 -High bias: Pays little attention to data; oversimplified (few features used)

-High variance: Pays too much attention to data; overfits (carefully minimized SSE)

-Need tradeoff between the two

-**Lasso Regression:** Minimize (SSE + λ|β|)

-Set parameters of less important features to very small values

y = m1x1 + m2x2 + …. + b

-Sk-learn: <http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html>

**Principal Component Analysis (PCA)**

-Dimensional reduction of data

-Given input data, PCA finds a new and simplified coordinate system by **rotation and translation only**

**-Principal component (PC)**: Combine two variables into one parameter on a coordinate system (not a fit!) which can be used as a new variable/parameter

-**Maximal variance**: Longer axis of the data (Principal component follows the direction of the maximum variance to retain more information)

-Computed by minimizing the linear/Euclidian distance from each data point to the principal component fit (which measures the amount of information lost)

-Do it for both longer and shorter/perpendicular PCs, whichever has the lesser sum gets to be the maximal variance

-Projection of each point onto the direction of the maximal variance *minimizes* the total distance

-Maximum number of PCs allowed is the number of input features of the data

-Note: PCA generated parameters are *not scalable*!

-Sk-learn: <http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>

When to use:

-Latent features driving the patterns in data

-Dimensionality reduction (reduce noise, visualize high dimensional data, reduce inputs)

-Application: Facial recognition (eigenfaces)

-High input dimensionality, general patterns in data surrounding facial features

**Cross Validation**

-Training and testing data sets, evaluating estimator performance

-Sk-learn: <http://scikit-learn.org/stable/modules/cross_validation.html>