**CS286 Solving Big Data Problems – Exam #2 Study Guide**

**By: Zayd Hammoudeh**

**Lecture #14 – Introduction to Data Science**

**Features**

|  |  |  |
| --- | --- | --- |
| **Feature –** Individual, measurable property of an observation. In supervised learning, it is used for predicting the target.  **Additional Names:**   * **Predictor** * **Attribute** * **Variable**   **Vectorizing Data** – Given an observation, assign features to dimensions. | **Attribute Types**   * **Continuous** (Numeric) – **Data across a continuum**. * **Binary** – Two possible values. * **Categorical** – Finite set of possible values. * **Text** –Vocabulary or n-gram.   + **Bi-gram** – 2 consecutive words. | **Qualities of Good Features/Feature Sets**   * **Correlation** with the target (i.e. class value) * Features are **independent** * **Number of Features in Minimized** * **Number of features is much less than the number of data points**. * **Features do not leak the target** meaning they do not provide information that would not be available under normal circumstances. |

|  |  |  |
| --- | --- | --- |
| **Set Dimensionality**   * – Number of observations * – Number of Features   Ideally: | **Principal Component Analysis Procedure** – **Dimensionality Reduction** | |
| **Procedure**   1. **Calculate covariance matrix of the original input matrix.** 2. **Perform eigenvalue decomposition of the covariance matrix.** 3. **Look for orthogonal dimensions with the greatest variance (principal)** 4. **Project input matrix into those dimensions.**   **Relative covariance preserved across projection**. | – Representation of the total variance of the principal components.  **Number of principal components and eigenvectors equivalent to the number of dimensions in the original input matrix.**  **Since the data is mapped to new dimensions, not easy to map principal components to the original data.** |

|  |  |  |
| --- | --- | --- |
| **Feature Scale** | | **Target Leaks** – **Sample data has “extra” information/features not found outside the sample**. May cause the model to be unrealistically good.  **Bias**   * **Sampling Bias** – Non-representative sample * **Observer Bias** – Corrupted/invalid measurement of features * **Funding Bias** – “Fudging”/manipulating of the data.   **Simpson’s Paradox** – **Trend that appears in different groups disappears (or changes) groups are combined**. **Example**: The average of averages does not equal the average all points. |
| **Potential Issues**   * **Relative Magnitude** – **Some features may have different magnitude than others** causing those features to dominate. * **Absolute Magnitude** – When features are too large or too small, **data may be lost to overflow or underflow** respectively. | **Potential Solutions**   * **Standardization** –Transform data to measure how many standard deviations it is from the mean. * **Normalization** – Transform data such that the range is between 0 and 1. |

**Algorithms and Evaluation**

|  |  |  |  |
| --- | --- | --- | --- |
| **Applications of Clustering**   * **Image Segmentation** * **Dendrogram (including Phylogentic Dendogram)** * **Market Segmentation** | **K-Means Clustering Algorithm**   1. **Initialize the K centroid** 2. **Assign points to the nearest centroid.** 3. **Update the centroids.** 4. **Check Stop Conditions** | **Methods for Selecting the Initial Clusters**   1. **Random** 2. **El Agha et. al. Method** 3. **All at the same point (e.g. origin)**   **Stop Condition Options**   1. **Min change of median from last iteration**. 2. Sufficiently **small intra-cluster distance** 3. Sufficiently **large intercluster distance** | **Intracluster Distance** – Distance between **points within the same cluster**. **Measures the cluster’s similarity**.  **Intercluster Distance** – **Distance between** **the centroids**. **Measures data separation between clusters.** |

|  |  |  |
| --- | --- | --- |
| **Naïve Bayes**   * Assumes all features are **conditionally independent.** * – A class value * – The feature. * The predicted class value is the one with the **highest Naïve Bayes probability**. | **Sensitivity/True Positive Rate**  **Specificity/True Negative Rate**  **Accuracy** | * For some domains (e.g. SPAM and cancer detection), you may use a weight function as different types of misclassification may have different costs. |

|  |  |  |  |
| --- | --- | --- | --- |
| **K-Nearest Neighbors (KNN)**  Example Distance Metrics   * **Euclidean** * **Manhattan** * **Cosine**   **Voting Related Issues:**   * May be **problematic if class distribution is skewed**. * Cloud **use distance as a weight**. * May want to **pick an odd number for to avoid ties**. | **Types Recommendation Filtering**   * + **Item Based**   + **User Based**   **Precision** – Proportion of **top-search results that are relevant**.  **Recall** – Proportion of **relevant results that are top-scoring**. | **Underfitting** – Model is too simple for the data set.  **Overfitting** – Model is overly trained to match the training set.  **Occam’s Razor** – Most plausible explanation is the simplest one.  **Pessimistic Estimation** – Impose a penalty () in cost function based on model.   * **Smaller** – Overfitting * **Larger** – Underfitting | **Sunrise Problem**  Using existing observations the probability is one but there will be a day when it does not rise.  **Smoothing:** Use probability distributions for parameter estimates for sparse data.   * **Smaller** – Overfitting * **Larger** – Underfitting |

|  |  |  |
| --- | --- | --- |
| **Cross Validation**   * Leave out. * Train on the points. Use the points to estimate the out of sample error. * Repeat this process until all points have appeared in the training set exactly once. * **Example:** **Leave-1-Out**. Generate and train models. Using this approach, the error is: * **Cross validation can be repeated with different model parameters. Select the model with the lowest cross validation error.** | **-Fold Cross Validation** – Divide the training set into equally sized pieces. Example: means each testing set is size .  **Downside of Leave-One-Out** – Computationally expensive. |  |

**Lecture #15 – Introduction to Machine Learning**

|  |  |  |  |
| --- | --- | --- | --- |
| **Machine Learning** – A **subfield of computer science** that deals with the construction and **study of systems that can learn from data rather than follow explicitly programmed instructions** | **Neuron** – A brain cell.  **Dendrite** – Appendages that stick out from a neuron.  **Axon** – Long chain that connects two neurons’ dendrites. Carries the electrical signal between two cells.  **Synapses** – Small cleft or separation between two neurons. | * **When neuron A repeatedly participates in firing neuron B, the strength of the action from A to B increases.** *Cells that fire together wire together.*   + **If they fire separately, the weight decreases.** * **Synaptic Strength** – Can be modeled as a set of weights. * **Hebbian Learning** – Changing of the weights between the neurons in the brain. | **Single Layer Perceptron (SLP) / Artificial Neuron** |

|  |  |  |
| --- | --- | --- |
| **Activation Function** –Defines the relation between the summed input to the perceptron and the perceptron’s output.  **Perceptron’s can be taught to recognize a binary pattern** (e.g. 0 or 1) | **Rosenblatt Perceptron Learning Algorithm**  **Step #1**: Initialize weights and perceptron thresholds (e.g. randomly).  **Step #2**: For each input () and expected/desire output (), do the following:   * + Calculate the net signal via:   + Calculate the perceptron’s output:   + Compute the error (one of the set ):   + Update the weights   **Step #3**: If stop condition met, then terminate, else repeat step #2. | **Sigmoid Activation Function**  **Benefits of the Sigmoid Function:**   * **Good approximation of the step function** * **Permits a continuous output** * **Solves noise saturation for large signals.** * **Solves noise attenuation for small signals.** * **Has a simple first order derivative.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Sigmoid Function** | **Deficiency of the Perceptron** – Cannot solve all functions. **Example**: XOR.   * Discovered by **Minsky and Papert**.   **Perceptron Functions must be linearly separable.** | **Multilayer Perceptron** | **Multilayer Perceptron:** Developed by **Werbos**  **Two Stages: Repeat until convergence**   * **Forward Pass** – Use the inputs to **calculate the outputs and deltas for each neuron**. * **Reverse Pass** – **Adjust the weights by fractions () of the deltas**. Done via **back propagation layer to layer**.   **Multilayer perceptrons are trained to recognize patterns**. **Example**: Handwriting recognition.  **Can compute XOR.** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Algorithm Name** | **Year Published** | **Author** | | Decision Tree | 1986 | Quinlan | | Support Vector Machine | 1995 | Vapnik and Cortes | | Boosting | 1998 | Freund and Shapire | | Random Forest | 2001 | Breiman | | Deep Learning | 2005 | Hinton et. al | |  |

|  |  |  |
| --- | --- | --- |
| **Single Layer Perceptrons can learn linearly separable functions only.**  **Multilayer perceptrons can learn linearly and non-linearly separable functions.** | **Supervised learning has input-output pairs during the training phase** (i.e. labeled data).  **Unsupervised learning does not have outputs** (i.e. unlabeled data). | **Classification Problems** – Have a finite set of outputs.  **Regression Problems** – Do not have a finite set of outputs. |

**Lecture #17 – Classification**

|  |  |  |
| --- | --- | --- |
| **Classification** – A supervised machine learning algorithm. Distinguishes objects of one class from another. Emulates human decision making.  **Example of Classification:**   * Credit card fraud detection * Credit approval   **Example of Regression:**   * Credit card profitability. | **Feature Extractor** – Given an input object, builds/extracts the set of features.  **Machine Learning Algorithm** – Generates the classifier model.  **Classifier Model** – Given an input, predicts a class label/value. | **Feature / Predictor Attribute Types**   * **Continuous** – Decimal or floating point.   + **Example:** 0.1 * **Categorical** – Predefined, finite set of values.   + **Example:** { true, false } * **Work-like** – Large set of defined values.   + **Example:** English dictionary * **Text-Like** – Sequence of word-like objects.   + **Example:** Email message subject. |