CS286 Solving Big Data Problems – Exam #2 Study Guide
By: Zayd Hammoudeh

Lecture #14 – Introduction to Data Science

Features

$$Model = SampleData + Algorithm(s)$$

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 $\{m, n\}$
- m – Number of observations
- n – Number of Features

Ideally: $m \gg n$

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 (i.e. principal components)**
4. **Project input matrix into those dimensions.**

Relative covariance preserved across projection.

λ – Representation of the total variance of the principal components.

$$\lambda = [\mathbf{4.2}, \mathbf{0.24}, 0.08, 0.02] = \frac{4.2 + 0.24}{4.2 + 0.24 + 0.08 + 0.02} = 98\%$$

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.
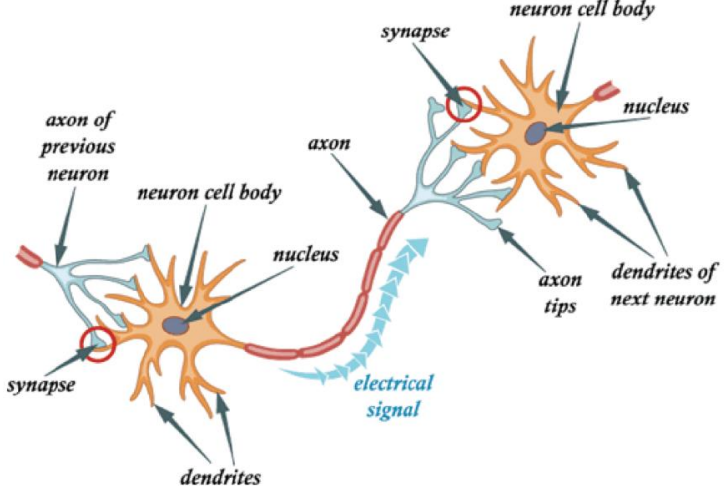

Feature Scale

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.
$$x' = \frac{x - \bar{x}}{\sigma_x}$$

- **Normalization** – Transform data such that the range is between 0 and 1.
$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

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) when groups are combined**. **Example**: The average of averages may not equal the average all points.
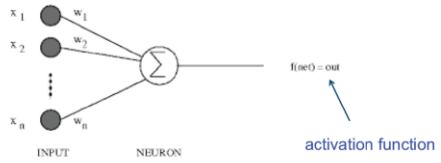

Algorithms and Evaluation

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 mean 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

$$P(C|F_1, \ldots, F_n) = \frac{P(C) \cdot P(F_1|C) \cdot \ldots \cdot P(F_n|C)}{P(F_1, \ldots F_n)}$$

- Assumes all features are **conditionally independent**.
- C – A class value
- F_i – The ith feature.
- The predicted class value is the one with the **highest Naïve Bayes probability**.

Sensitivity/True Positive Rate
$$TPR = \frac{TP}{TP + FN}$$

Specificity/True Negative Rate
$$TNR = \frac{TN}{TN + FP}$$

Accuracy
$$Acc = \frac{TP + TN}{TP + FN + FP + TN}$$

- For some domains (e.g. SPAM and cancer detection), you may use a weight function as different types of misclassification may have different costs.
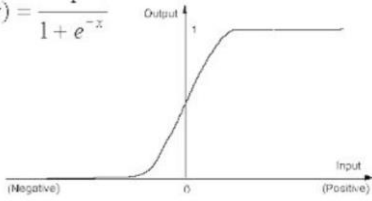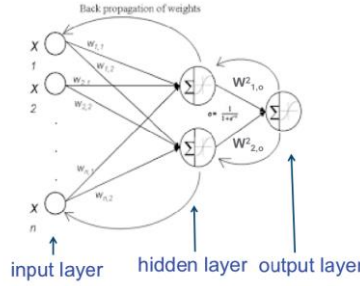
K-Nearest Neighbors (KNN)

Example Distance Metrics
- Euclidean
- Manhattan
- Cosine

Voting Related Issues:
- May be problematic if class distribution is skewed.
- Can use distance as a weight.
- May want to pick an odd number for k to avoid ties.

Types Recommendation Filtering
- o Item Based
- o User Based

Precision – Proportion of top-search results that are relevant.

$$Precision = \frac{TP}{TP + FP}$$

Recall – Proportion of relevant results that are top-scoring.

$$Recall = \frac{TP}{TP + FN}$$

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 (λ) in cost function based on model.
- Smaller λ – Overfitting
- Larger λ – Underfitting

Sunrise Problem
$$P(sun\ will\ rise\ tomorrow) = ?$$
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.

$$\hat{\theta}_i = \frac{x_i + \alpha}{N + \alpha \cdot d}$$

- Smaller α – Overfitting
- Larger α – Underfitting

Cross Validation
- Leave k out.
- Train on the $N - k$ points. Use the k 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 N models. Using this approach, the error is:

$$Error_{LeaveOneOut} = \frac{1}{N} \cdot \sum_{i=1}^{N} e_i$$

- Cross validation can be repeated with different model parameters. Select the model with the lowest cross validation error.

K-Fold Cross Validation – Divide the training set into k equally sized pieces. Example: $k = 10$ means each testing set is size $\frac{N}{10}$.

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.*
  - o 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 ($\vec{x}$) and expected/desire output (d), do the following:
- o Calculate the net signal via:
$$\vec{x}^T \vec{w}$$
- o Calculate the perceptron's output:
$$y_j(t) = sign(\vec{x}^T \vec{w})$$
- o Compute the error (one of the set $\{-1.0.1\}$):
$$e_j(t) = d_j(t) - y_j(t)$$
- o Update the weights
$$w_i(t + 1) = w(t) + \alpha \cdot e_j(t) \cdot x_{i,j}$$
Step #3: If stop condition met, then terminate, else repeat step #2.

Sigmoid Activation Function

$$f(x) = \frac{1}{1 + e^{-x}}$$

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.	Multilayer Perceptron	Multilayer Perceptron: Developed by Werbos

Sigmoid Function

$$f(x) = \frac{1}{1 + e^{-x}}$$

Output

1

(Negative)   0   Input (Positive)

Deficiency of the Perceptron – Cannot solve all functions. **Example**: XOR.
- Discovered by **Minsky and Papert**.

Perceptron can only solve functions that are linearly separable.

Multilayer Perceptron

Back propagation of weights

x_1 $w_{t,1}$
x_2 $w_{t,2}$ $\sum$ $W^2_{1,o}$ $\sum$ f
  $e = \frac{1}{1+e^{-x}}$
  $\sum$ $W^2_{2,o}$
x_n $w_{t,3}$

input layer   hidden layer   output layer

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 (α) 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

machine learning

supervised   ↔   unsupervised

Lots of other algorithms that sit in the middle here somewhere

- **Classification**
  - KNN
  - Decision tree
- **Regression**
  - linear
  - logistic

- **Clustering**
  - K-means
  - DBSCAN
- **Dimensionality reduction**
  - PCA
  - SVD

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 uses only inputs, not 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 – Using Mahout for Naïve Bayes

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.
		  o **Example:** 0.1
		• **Categorical** – Predefined, finite set of values.
		  o **Example:** { true, false }
		• **Word-like** – Large set of defined values.
		  o **Example:** English dictionary
		• **Text-Like** – Sequence of word-like objects.
		  o **Example:** Email message subject.

Bayes Theorem	Assumptions of Naïve Bayes		Calculating Conditional Probability
Bayes Theorem $$P(A\|B) = \frac{P(B) \cdot P(B\|A)}{P(A)}$$ • **Can be used to prove the Monty Hall problem.**	**Assumptions of Naïve Bayes** 1. **Input data is labeled.** 2. **Predictors are in N-dimensional space.** 3. **Target is a set of categorical values.** 4. **Class C is dependent on the set of input attributes.** 5. **All features are conditionally independent.**	**Naive Bayes may not be Bayesian** since it simplifies the problem by the independence assumption. **Bag of Words** – Transform a text input into an unordered set of words.	**Calculating Conditional Probability** $$P(C\|A) = \frac{N_{A,C} + 1}{N_C + k}$$ • N_C – Number of elements of class C • $N_{A,C}$ – Number of elements of class C with attribute A • k – Number of classes.

Sequence of Mahout Commands • **mahout seqdirectory** • **mahout seq2sparse** • **mahout split** • **mahout trainnb** • **mahout testnb**			

Lecture #13a – HBase

Relational Database – A datastore that whose **type and structure is defined before storage**. Uses SQL.

Deficiencies of Relational Databases
- **Do not scale horizontally.**
- **Sharding is difficult to manage (join and transactions do not scale across shards)**

HBase – A distributed database where puts and gets are accessed via a key.

Table Splits – Occur automatically as the table grows.

Horizontal Partitioning – Putting rows into different tables. HBase uses **key ranges/regions** to **define the horizontal partitions (shards)**.

Row Key – Used to store and access data.

Column Family – Used to group and store similar data (i.e. **subcolumns – column qualifiers**). Attributes of column families include:
- **Number of versions**
- **Time to Live (TTL)**
- **Compression**
- **Keep in memory or preserve to disk.**
- **Column family are separated into different files**

Pros of HBase:
- **Scalable to handle data volume and velocity.**
- **Fast reads and writes by key.**

Cons of HBase:
- **Does not support joins**
- **Good schema design is needed to achieve the best performance.**

Data is stored in a key-value model.

Information Required to Access a Single Cell Value in HBase
- **Row Key**
- **Column Family**
- **Column Qualifier** (i.e. subcolumn name)
- **Timestamp/Version**

Version in HBase
- **Each put and delete adds a new version/cell.**
- Stores last 3 versions by default.
- **Version** – **Stored as a long which is the current time in milliseconds** (if no specific version is specified)

Table Physical View
- Stored as a **sorted map**.
- Ordered by **row key and column qualifier in ascending order**.
- **Ordered by version/timestamp in descending order**.

Basic Table Operations
- **Create table and defining of column families is done before data is imported.**

Basic CRUD Operations
- **Put** – Insert data into rows (both create and update)
- **Get** – Access data from one row
- **SCAN** – Access data from a range of rows
- **Delete** – Delete a row or a range of rows/columns.

Region – Another name for a **key range**. All key **range is contiguous**.

Region Server – Serve data for reads from or writes to a particular region/key range.

Write Ahead Log (**WAL**) – Disk commit log used for recovery.
- **Primary Role:** Durability (log on disk)
- Updates appended sequentially

Block Cache – Read cache. Uses the Least Recently Used (**LRU**) paradigm for block eviction.

Memstore – Write cache.
- **In memory**.
- **Sorted set of key-value pairs**. Updates quickly stored since in memory.
- **One memstore for each column family**.

HFile – Sorted key-value pairs on disk.
- Ideally one per column family.

HBase Region Flush – All contents of memstore flushed to an HFile on disk.
- Since Memstore is sorted, HFile is also sorted.

Minor Compaction – Merge multiple, small HFiles into fewer larger ones.

Major Compaction – Merge all HFiles into one large HFile with all records marked for deletion removed.

Data Model for Fast Writes and Reads
- Layout on physical disk is predictable minimizing disk seek.
- Fast access since get and put is by row key.
- **Since tables are sorted, scan of a key range is fast**.

Creation of a Key Region – Table starts as a single range. **When a region becomes too large, it splits** into two child regions.

Region Server – Initiates a region split. The child/daughter regions are opened on the same server.

HBase Use Case #1 – **High Velocity and Volume Writes.**
- **Examples:** Stock ticker, sensors, log files, system metrics. **Real time monitoring**.

HBase Use Case #2 – Information exchange with **high volume read and write**.
- **Examples**: Email, chat, Facebook.

HBase Use Case #3 – **High volume read.**
- Examples: Content serving, web application back end, search index, online pre-computed view, online catalog.

Lecture #13b – Recommendation

Input to Recommender Systems – Interactions between users and items.

Output of Recommender Systems – Suggestions of additional interactions.

Common Examples: Movies, music, restaurant choices, sale items at stores

Basis of Recommendation – *Behavior of a crowd helps us understand what individuals will do.*

Popular Items – Co-occur with everything making them not very useful for recommendation.

Anomalous Co-Occurrence – Far more useful for recommendation. **They are the source indicators of preference**.

History Matrix – Constructed from the log files to show the history of the users for the items.
- **Users by items**

Co-occurrence Matrix – Quantifies how often two items appear together.
- **Items by items**

Indicator Matrix – Represent anomalous (i.e. interesting) co-occurences.
- **Items by items**

Log Likelihood Ratio (LLR) – Can be helpful to judge with co-occurrences can be used with confidence as indicators of preference.

Cooccurrence Analysis

Which one is the anomalous co-occurrence?

	A	not A
B	13	1000
not B	1000	100,000

	A	not A
B	1	0
not B	0	2

	A	not A
B	1	0
not B	0	10,000

	A	not A
B	10	0
not B	0	100,000

Bottom right is anomalous co-occurrence.
Consistently appear together or separate.

User-Based Filtering – Recommend items by finding similar users. Harder to scale.

Item-Based Filtering – Calculate similarity between items and make recommendations. This can be done offline.

Dithering – **Random re-ordering of recommendation results**. This makes for more off-line computation, but generally makes results better.
- Helps test relevancy of new items that would otherwise go unviewed.

- A – Matrix of User search queries
  - Matrix is $Users$ by $Queries$
- $A^T A$ – Gives a matrix of $Queries$ by $Queries$
  - **Query Co-occurrence Matrix**
  - This enables query recommendation.

- B – Matrix of User video views
  - **Video View Co-occurrence Matrix**
  - Matrix is $Users$ by $Videos$
- $B^T B$ – Gives a matrix of $Videos$ by $Videos$
  - This enables recommendations in the form "*you may like these videos.*"

- $A^T B$ – **Gives query and video view co-occurrence**. ***Just like a search engine.***

Lecture #16a – Introduction to Recommendation

Recommendation – Class of machine learning that seeks to predict a user's preference for or rating of an item.

Two Main Approaches for Recommendation
- **Collaborative Filtering**
- **Content-based Filtering**

User Based Filtering – Recommendation are based on similarity to other users.

Item-based Filtering – Recommendations are made based on similarity to other items.

Similarity Metrics
- **Pearson Correlation** – Ratio of co-variance to product of standard deviations.
  - **-1** – Inversely Proportional
  - **0** – No correlation
  - **1** – Directly proportional

- **Euclidean Distance** – Coordinates indicate item preference. Smaller distance means more similarity.

Tanimoto Coefficient – Ratio of intersection to union.
- Between 0 and 1. Bigger is more similar.

Collaborative-Based Filtering – **Deductive**. Needs user-item history to start to learn item associations.
- **Example:** last.fm

Content Based Filtering – **Inductive**. Relies on domain specific knowledge to construct association between users and items.
- **Example:** Pandora

Challenges of Collaborative Filtering
- **Cold Start** – No user history means no associations on day 1.

- **Scale** – Huge number of products of items and users means lots of computation.

- **Sparsity** – Most users express very little behavior with very few items and no behavior with the vast majority of items.

Neighborhood – A group of similar users.

Types of Neighborhoods:
- **Fixed Size** – Cardinality (size) of the neighborhood is fixed in advance.

- **Threshold-based** – Cardinality is dependent based on a threshold of similarity that is fixed in advance.

Precision – Proportion of top-scoring results that are relevant.

$$precision = \frac{TP}{TP + FP} = \frac{a}{a + c}$$

Recall – Proportion of relevant results that are top-scoring.

$$recall = \frac{a}{a + b} = \frac{TP}{TP + FN}$$

Methods for Improving Recommender Performance
- **Change Model Type**
  - User Based
  - Item Based
  - Content Based
- **Change Distance Metric**
  - Euclidean
  - Tanimoto
  - LogLikelihood Ratio (LLR)
- **Model Parameters**
  - K in KNN

Apriori Algorithm – Used to determine what people buy or use together.
- Can be used for recommendation.

KNN Algorithm – Can be used for both classification and recommendation.

Two Main Algorithms for Recommendation Engine
- **Collaborative Filtering**
- **Matrix Factorization**

Fundamental Problem of Collaborative Filtering – Finding a distance metric.

Base Idea of Collaborative Filtering – Like KNN. Make decisions based off similar things (e.g. items, users, etc.). (**Use collaboration from others to make recommendations**)

Alternative Leas Squares (**ALS**) – Inspired by matrix factorization but has almost nothing to do with it.
- More similar to neural networks.
- Can be random or unstable, but it works fine for some cases.