Data Science Algorithms

Peter Meleney

”All models are wrong, but some are useful.”

-George E.P. Box

This book is still very much a work in progress, and I cannot ensure that it is free of mistakes during this early phase of development. Eventually I intend for this book to be a one-stop resource as an introduction to data science including rigorous mathematical exploration and derivation of the included algorithms and include example applications in R and Python. Thank you for your interest in this project, if you have any comments or questions please contact me at pmeleney@gmail.com.

Created: March 4, 2016
Last Updated: April 12, 2016

Contents

I Unsupervised Algorithms I: Dimensionality Reduction 6

1 Principle Component Analysis 7
1.1 Mathematics . . . 7

II Unsupervised Algorithms II: Clustering and Networks 8

2 K-Means Clustering 9
2.1 K-Means Algorithm . . . 9
2.2 Requirements . . . 10
2.3 Choosing K . . . 10
2.3.1 The ”Elbow” Method . . . 10
2.3.2 Silhouette Method . . . 10
2.4 Assessing Goodness of Fit . . . 11
2.4.1 Error Rate . . . 11
2.4.2 The Bayes Classifier . . . 11
2.5 Example Applications ***ToDo*** . . . 12
2.5.1 Applications in Python ***ToDo*** . . . 12
2.5.2 Applications in R ***ToDo*** . . . 12

3 K-Nearest Neighbors 13
3.1 KNN Algorithm . . . 13
3.1.1 Distance Weighted KNN . . . 13
3.1.2 Instance Weighted KNN ***ToDo*** . . . 14
3.1.3 Attribute Weighted KNN ***ToDo*** . . . 14
3.2 Requirements . . . 14
3.3 Choosing K . . . 14
3.4 Applications of KNN ***ToDo*** . . . 14
3.4.1 Applications in Python ***ToDo*** . . . 14
3.4.2 Applications in R ***ToDo*** . . . 14

4 Outlier Detection **15**
4.1 ***To Do*** 15

III Supervised Algorithms I: Regression **16**

5 Linear Regression **17**
5.1 Introduction 17
5.2 Assumptions 18
5.3 Simple Linear Regression 18
5.3.1 Least Squares Criterion 19
5.3.2 Derivation of β_0 19
5.3.3 Derivation of β_1 20
5.4 Multiple Linear Regression 21
5.5 Pearson Correlation Coefficient 22
5.6 Goodness of Fit 22
5.6.1 Adjusted R^2 for Multiple Linear Regression 23
5.7 Outliers 23
5.8 Extrapolation 23
5.9 Errors of Estimated Coefficients 24
5.9.1 Prediction Bands 24
5.9.2 Confidence Intervals 24
5.10 Checking Assumptions Using Graphs 24
5.11 Model Selection Strategies 24
5.11.1 The R^2_{adj} Approach 24
5.11.2 The p-Value Approach 24
5.12 Quick Methods and Shortcuts 24

IV Supervised Algorithms I: Classification **25**

6 Logistic Regression **26**
6.1 Introduction 26
6.2 Mathematics 27

V Supervised Algorithms III: Algorithms for Classification and Regression **29**

7 Naïve Bayes **30**
7.1 Probability 30
7.1.1 Tree Diagrams 30

7.2 Bayes' Thorem 32
7.3 Naïve Bayes Applied 33
7.3.1 High-Level Class Probabilities 34
7.3.2 Rare Words 34

8 Support Vector Machines 35
8.1 ***To Do*** 35

9 Tree-Based Methods 36
9.1 Decision Trees 36
9.1.1 Tree Pruning 37
9.1.2 Classification Trees 38
9.2 Bagging, Random Forest, and Boosting 39
9.2.1 Bagging 39
9.2.2 Random Forest 39
9.2.3 Boosting 40
9.3 Advantages and Disadvantages of Tree-Based Methods 40

10 Ensemble Methods 42
10.1 ***To Do*** 42

VI Time Series 43

11 Holt-Winters Forecasting 44
11.1 ***To Do*** 44

VII Advanced Algorithms 45

12 Neural Networks 46
12.1 ***To Do*** 46

VIII Supporting Information 47

13 Distance Metrics 48
13.1 Euclidean Distance 48
13.2 Manhattan Distance (Hamming Distance) 48
13.3 Cosine Distance 49
13.4 Jaccard Similarity 49

14 Application Program Interface 50

14.1 Introduction 50
14.2 RESTful APIs 50
14.2.1 RESTful API Operations 51
14.2.2 Example 51
14.2.3 API Resources 51
14.3 API Languages 52
14.3.1 JSON 52
14.3.2 XML 52

15 Set Theory, Relational Algebra, and Structured Query Language 53
15.1 Introduction 53
15.2 Set Theory 53
15.2.1 Basic Concepts and Notation 54
15.3 Structured Query Language 54
15.3.1 Best Practices 55
15.3.2 Entity Relationship Diagrams 55
15.3.3 Relational Schema 55
15.3.4 Database Metadata 55
15.3.5 Basic Queries 55
15.3.6 Aggregation Functions ***ToDo*** 57
15.3.7 Joins ***ToDo*** 57
15.3.8 Subqueries ***ToDo*** 57
15.3.9 Logical Expressions ***ToDo*** 57

Part I

Unsupervised Algorithms I: Dimensionality Reduction

Chapter 1

Principle Component Analysis

Principle component analysis (PCA) is a popular approach for deriving a lower-dimensional set of features from a high-dimenstional set. The PCA algorithm identifies and preserves p directions of highest variance in the data and destroys the $m - p$ directions of lowest variability.

This technique has two primary applications. Firstly, it can reduce the complexity of data input to another algorithm, especially if the full feature set is too large to be processed in a timely manner. Secondly it can be used as an unsupervised technique in it's own right. The similarity of direction in the new PCA space may be treated as a more or less quantitative measure of feature similarity. Additionally, and of particular interested is the case of setting $p = 2$, making optimized binary diagrams possible.

1.1 Mathematics

Given a data set X, with n observations and m features, and user-defined p, the desired number of principle components, PCA computes ϕ

$$Z^{[n\times p]} = \phi^{[p\times m]} X^{[n\times m]} \tag{1.1}$$

Part II

Unsupervised Algorithms II: Clustering and Networks

Chapter 2

K-Means Clustering

K-Means Clustering (KMC) is a greedy learner that partitions a data set into k clusters. The algorithm minimizes the Mean Square Distance between all the data points and the centroid of the group to which they belong. This process is not robust, meaning that the algorithm can fall into a local minimum and never reach a global minimum.

Modifying K-Means Clustering to K-Medians Clustering can be advantageous in some circumstances. What does it mean to have bought 0.723 of an item? Sometimes this is not a meaningful value for a cluster center.

2.1 K-Means Algorithm

1. Scale features to standardize distance.
2. Initialize k centers
3. Assign each point to its nearest center
4. Calculate the centroid of each group of points
5. Move the centers to their respective centroids
6. Repeat steps 3-5 until the centers no longer move with each iteration, this is your final set of groups.

2.2 Requirements

All features should be scaled before K-Means Clustering is applied.

2.3 Choosing K

2.3.1 The "Elbow" Method

This method involves performing KMC for a range of K, usually up to $\sqrt{n}$ or $\ln(n)$, then graphing the mean square distance from the points to their respective centers. Usually somewhere in the graph a distinct change in slope will occur, from strongly negative to weakly negative. This elbow, or very near it will probably represent an optimal choice of K [2].

2.3.2 Silhouette Method

If clusters are properly assigned, then data points within a cluster should be much closer to each other than they are to data points in other clusters. The silhouette can be calculated for each point. In plain English: it is the average distance to points in the same cluster minus the average distance to points in the nearest (not same) cluster, divided by the maximum of the two. This value will always be [1,-1]. If the value is positive, the point is nearer points in its own group, if negative, it is nearer points in the other group [1].

A distance function most appropriate to the situation may be chosen to maximize the validity of the model.

$$\frac{Avg[dist(x,x_j)] - Avg[dist(x,x_i)]}{max\{Avg[dist(x,x_j)], Avg[dist(x,x_i)]\}}$$

$$\frac{\frac{1}{n_{k_N}}\sum_{j=0}^{n_{k_N}} dist(x,x_j) - \frac{1}{n_k}\sum_{i=0}^{n_k} dist(x,x_i)}{max(\frac{1}{n_{k_N}}\sum_{j=0}^{n_{k_N}} dist(x,x_j) - \frac{1}{n_k}\sum_{i=0}^{n_k} dist(x,x_i))} \tag{2.1}$$

i - points in cluster k
j - points in the cluster nearest k
n_k - number of data points in cluster k
n_{k_N}: number of data points in the cluster nearest k

k should then be selected that minimizes the average silhouette of all the data points in the set.

2.4 Assessing Goodness of Fit

2.4.1 Error Rate

The most common approach for quantifying the accuracy of a K-Means model is the error rate:

$$\frac{1}{n}\sum_{i=0}^{n} I(y_i \neq \hat{y}_i) \tag{2.2}$$

Where $I(y_i \neq \hat{y}_i)$ is an *indicator variable* which equals 1 if $y_i \neq \hat{y}_i$ and 0 if $y_i = \hat{y}_i$. Equation 2.2 is referred to as the **training error** when it is computed on the training data set, and the **test error** when computed on a test set.

2.4.2 The Bayes Classifier

It is possible to show that the equation 2.2 is minimized when each observation is assigned to the most likely class.

$$Max(P(y = j|X = x_o)) \tag{2.3}$$

This very simple classification rule is called a **Bayes Classifier**. In a two-class problem it can be restated:

$$P(y = j|X = x_o) > 0.5 \tag{2.4}$$

The boundary between two classes at which the probability of a hypothetical point is equal to 50% is called the **Bayes decision boundary**, and is a theoretical optimal solution to binary classification problems.

A data scientist need not choose the Bayes classifier as the decision boundary. Any value (0,1) may be chosen, the accuracy will be lower, but it may be desirable to exchange accuracy for greater precision or recall.

2.5 Example Applications ***ToDo***

2.5.1 Applications in Python ***ToDo***

`This is in pcr font, and this text is ForestGreen.`

2.5.2 Applications in R ***ToDo***

Chapter 3

K-Nearest Neighbors

3.1 KNN Algorithm

The K-Nearest Neighbors (KNN) algorithm is usually used for classification, but can also be used for regression. Given a positive integer K, and a test observation x_0, the KNN classifier identifies the K nearest labeled observations to x_0, denoted η_0, and the computes the probability that x_0 belongs to class j based on the distribution of those labeled points η_0. KNN can apply weights based on distance, or not. In its simplest, unweighted form, the algorithm identifies tη_0 and assigns x_0 to whichever class has the most points represented in η_0.

$$P(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in \eta_o} I(y_i = j) \tag{3.1}$$

3.1.1 Distance Weighted KNN

KNN can be written so that the distance from x_0 to each point in η_0 is considered by the algorithm. The distance metric may be any of those from chapter 13.

$$P(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in \eta_o} \frac{I(y_i = j)}{Dist(x_0, y_i)^2} \tag{3.2}$$

3.1.2 Instance Weighted KNN ***ToDo***

3.1.3 Attribute Weighted KNN ***ToDo***

3.2 Requirements

All features should be scaled prior to initiating the KNN algorithm.

3.3 Choosing K

The choice of K has a drastic effect on the KNN algorithm. Choosing $K < K_{opt}$ will result in underfitting the decision boundary, while choosing $K > K_{opt}$ will result in overfitting. We expect the training error to monotonically decrease as $K \to \infty$, however the test error will not monotonically decrease. Instead the test error will decrease as $K = 1 \to K_{opt}$ and then increase again as $K_{opt} \to \infty$.

Choosing K_{opt} can be achieved by minimizing the error rate of the cross validation data subset, or through k-fold cross validation ***To Do: Add ref to CV Chapter*** . Be careful not to optimize to the test set, as that set will no longer give you a proper analysis of model accuracy.

3.4 Applications of KNN ***ToDo***

3.4.1 Applications in Python ***ToDo***

3.4.2 Applications in R ***ToDo***

Chapter 4

Outlier Detection

4.1 ***To Do***

Part III

Supervised Algorithms I: Regression

Chapter 5

Linear Regression

5.1 Introduction

Linear regression is the most common supervised learning technique because it is straightforward to apply and relatively easy to interpret. Linear regression has several drawbacks however. Many systems do not follow a linear model, the most common violations of the assumptions of linear regression occur when features interact or when error terms are auto-correlated. Unlike several more advanced algorithmic approaches, linear regression is particularly sensitive to violations of its assumptions.

Linear regression is usually solved to minimize the **sum of squared errors** (SSE), confusingly also called the **residual sum of squares** (RSS). When applying this **cost function**, the solution to simple linear regression is as follows:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i \tag{5.1}$$

$$\hat{\beta}_1 = \frac{\sum_{i=0}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=0}^{n} x_i - \bar{x}} \tag{5.2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \tag{5.3}$$

The solution to multiple linear regression, when y is the solution to is somewhat more involved, and is better treated as a linear algebra problem. The solution is:

$$\hat{\beta} = (X^T X)^{-1} X^T \mathbf{y} \tag{5.4}$$

Derivations of these solutions are below.

5.2 Assumptions

Assumptions 1-3 are critical when applying linear regression, the others are required if linear regression is to be BLUE (best linear unbiased estimator), but their violation may not invalidate the model entirely.

1. The objective can be estimated by a linear combination of the independent features.
2. Features do not interact or are additive only.
3. Errors are independent (no auto-correlation).
4. Outliers and influential points are removed.
5. Errors are homoskedastic.
6. Errors are normally distributed.

5.3 Simple Linear Regression

Linear regression assumes that the objective is a linear combination of the input features. Simple linear regression, when the objective y is a function of only one independent feature x, has the form:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \tag{5.5}$$

Where ϵ_i represents the error between the linear relationship and the actual value of y_i. The linear regression algorithm approximates equation 5.5 by enforcing the relationship below:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i \tag{5.6}$$

Where $\hat{y}_i$ is the estimated output based on the linear relationship. The actual output y_i must include an error term e_i because the relationship is not perfect.

$$y_i = \hat{\beta}_0 + \hat{\beta}_1 x_i + e_i \tag{5.7}$$

Subtracting equation 5.7 from equation 5.6 gives us the relationship:

$$e_i = \hat{y}_i - y_i \tag{5.8}$$

This remainder, e_i is commonly used in the **cost function**. The most common cost function is the sum of the square of the cost function, called the **least squares criterion**.

5.3.1 Least Squares Criterion

The least squares criterion minimizes the **residual sum of squares** (RSS):

$$RSS = \sum_{i=0}^{n} e_i^2 \tag{5.9}$$

$$RSS = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 \tag{5.10}$$

$$RSS = \sum_{i=0}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 \tag{5.11}$$

We can minimize the RSS cost function by setting $\frac{\partial RSS}{\partial \hat{\beta}_1} = 0$ and $\frac{\partial RSS}{\partial \hat{\beta}_0} = 0$ and solving for β_1 and β_0 respectively.

5.3.2 Derivation of β_0

$$\begin{aligned}
\frac{\partial RSS}{\partial \hat{\beta}_0} = 0 &= \frac{\partial}{\partial \hat{\beta}_0}\left[\sum_{i=0}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2\right] \\
0 &= \sum_{i=0}^{n}\left[\frac{\partial}{\partial \hat{\beta}_0}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2\right] \\
0 &= -2\sum_{i=0}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i) \\
0 &= \sum_{i=0}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i) \\
0 &= \sum_{i=0}^{n} y_i - \sum_{i=0}^{n} \hat{\beta}_0 - \sum_{i=0}^{n} \hat{\beta}_1 x_i
\end{aligned}$$

$$n\beta_0 = \sum_{i=0}^{n} y_i - \hat{\beta}_1 \sum_{i=0}^{n} x_i$$

$$\beta_0 = \frac{\sum_{i=0}^{n} y_i}{n} - \hat{\beta}_1 \frac{\sum_{i=0}^{n} x_i}{n}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \tag{5.12}$$

5.3.3 Derivation of β_1

$$\begin{aligned}
\frac{\partial RSS}{\partial \hat{\beta}_1} = 0 &= \frac{\partial}{\partial \hat{\beta}_1}\left[\sum_{i=0}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2\right] \\
0 &= \sum_{i=0}^{n}\left[\frac{\partial}{\partial \hat{\beta}_1}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2\right] \\
0 &= -2\sum_{i=0}^{n}(\frac{\partial}{\partial \hat{\beta}_1}\hat{\beta}_1 x_i)(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i) \\
0 &= -2\sum_{i=0}^{n} x_i(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i) \\
0 &= \sum_{i=0}^{n} x_i(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i) \\
0 &= \sum_{i=0}^{n} (y_i x_i - \hat{\beta}_0 x_i - \hat{\beta}_1 x_i^2) \\
0 &= \sum_{i=0}^{n} (y_i x_i - (\bar{y} - \hat{\beta}_1 \bar{x}) x_i - \hat{\beta}_1 x_i^2) \\
0 &= \sum_{i=0}^{n} (y_i x_i - \bar{y} x_i - \hat{\beta}_1 \bar{x} x_i - \hat{\beta}_1 x_i^2) \\
0 &= \sum_{i=0}^{n} (y_i x_i - \bar{y} x_i) - \hat{\beta}_1 \sum_{i=0}^{n} (\bar{x} x_i - x_i^2) \\
\hat{\beta}_1 &= \frac{\sum_{i=0}^{n} (y_i x_i - \bar{y} x_i)}{\sum_{i=0}^{n} (\bar{x} x_i - x_i^2)}
\end{aligned}$$

Equation 5.13 can be further simplified into a more convenient form if we note that:

$$\sum_{i=0}^{n} (\bar{x}\bar{y} - y_i \bar{x}) = 0 \tag{5.13}$$

$$\sum_{i=0}^{n} (\bar{x}^2 - x_i\bar{x}) = 0 \tag{5.14}$$

Then:

$$\begin{aligned}
\hat{\beta}_1 &= \frac{\sum_{i=0}^{n} (y_i x_i - \bar{y}x_i) + \sum_{i=0}^{n} (\bar{x}\bar{y} - y_i\bar{x})}{\sum_{i=0}^{n} (\bar{x}x_i - x_i^2) + \sum_{i=0}^{n} (\bar{x}^2 - x_i\bar{x})} \\
\hat{\beta}_1 &= \frac{\sum_{i=0}^{n} (y_i x_i - \bar{y}x_i + \bar{x}\bar{y} - y_i\bar{x})}{\sum_{i=0}^{n} (\bar{x}x_i - x_i^2 + \bar{x}^2 - x_i\bar{x})} \\
\hat{\beta}_1 &= \frac{\sum_{i=0}^{n} (y_i x_i - \bar{y}x_i + \bar{x}\bar{y} - y_i\bar{x})}{\sum_{i=0}^{n} (\bar{x}x_i - x_i^2 + \bar{x}^2 - x_i\bar{x})}
\end{aligned}$$

$$\hat{\beta}_1 = \frac{\sum_{i=0}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=0}^{n} x_i - \bar{x}} \tag{5.15}$$

$$\hat{\beta}_1 = \frac{Cov(x, y)}{Var(x)} \tag{5.16}$$

5.4 Multiple Linear Regression

Deriving the solution to the multiple linear regression trained on the **least squares criterion** is very similar to the derivation for simple linear regression we just went through in section 5.3, and the solution takes significantly fewer steps. All coefficients β_0 through β_m can be solved simultaneously, using linear algebra.

The basic equation for multiple linear regression and its remainder are very similar to equation their simple linear regression equivalents: 5.5:

$$\mathbf{y} = X\beta + \epsilon \tag{5.17}$$

$$\begin{aligned}
\mathbf{e} &= \mathbf{y} - X\hat{\beta} \\
SSE &= \mathbf{e}^T\mathbf{e} \\
SSE &= (\mathbf{y} - X\hat{\beta})^T(\mathbf{y} - X\hat{\beta}) \\
SSE &= \mathbf{y}^T\mathbf{y} - \mathbf{y}^T X\hat{\beta} - \mathbf{y}X^T\hat{\beta}^T + \hat{\beta}^T X^T X\hat{\beta}
\end{aligned}$$

$$
\begin{aligned}
SSE &= \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T X\hat{\beta} + \hat{\beta}^T X^T X\hat{\beta} \\
\frac{\partial SSE}{\partial \hat{\beta}} &= \frac{\partial}{\partial \hat{\beta}}(\mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T X\hat{\beta} + \hat{\beta}^T X^T X\hat{\beta}) \\
0 &= -2X^T Y + 2X^T X\hat{\beta} \\
X^T X\hat{\beta} &= X^T Y \\
\hat{\beta} &= (X^T X)^{-1} X^T Y
\end{aligned}
$$

5.5 Pearson Correlation Coefficient

The **Pearson correlation coefficient** is a measure of the linear correlation of two variables. Usually denoted R, the property is equal to the ratio of the covariance to the products of the standard deviations.

$$
\begin{aligned}
R &= \frac{Cov(x,y)}{\sigma_x \sigma_y} && (5.18) \\
&= \frac{\sum_{i=0}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=0}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=0}^{n}(y_i - \bar{y})^2}} && (5.19) \\
&= \frac{1}{n-1}\sum_{i=0}^{n}\frac{x_i - \bar{x}}{\sigma_x}\frac{y_i - \bar{y}}{\sigma_y} && (5.20)
\end{aligned}
$$

There are many formulas for R, all of them equivalent, use whichever is most convenient.

$$R \in [-1, 1] \tag{5.21}$$

R is always within the range [-1,1], where -1 represents perfect negative correlation, and 1 represents perfect positive correlation.

5.6 Goodness of Fit

The **coefficient of determination** is usually used to assess how well a particular linear model fits a particular set of data and is usually reported whenever a linear regression is performed. Denoted R^2, this number quantitatively indicates how well the regression explains the objective variable. It is equal to unity minus the fraction of unexplained variance:

$$R^2 = 1 - \frac{\sum_{i=0}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n}(y_i - \bar{y})^2} = 1 - \frac{\mathbf{e}^T\mathbf{e}}{n[Var(\mathbf{y})]} \tag{5.22}$$

If linear regression is performed correctly, the linear model should never have higher variance than simply using the mean, so:

$$0 \leq R^2 \leq 1 \tag{5.23}$$

Finally, the coefficient of determination is also equal to the square of the correlation coefficient and the square root of the coefficient of determination is equal to the absolute value of the correlation coefficient.

5.6.1 Adjusted R^2 for Multiple Linear Regression

The adjusted R^2 value (R^2_{adj} is based on the calculation for R^2 but penalizes the model for each additional variable. In theory, it would be possible to fit any arbitrary data set of size m with a linear model using $m-1$ terms, R^2_{adj} accounts for this:

$$R^2_{adj} = 1 - (1 - R^2)\frac{n-1}{n-k-1} \tag{5.24}$$

Where:
n = number of samples.
k = number of predictor variables.

It is important to note that R^2 and R^2_{adj} are not equivalent when $k = 1$, and that R^2_{adj} is no longer the fraction of explained variance, rather it is best used as a measure of whether the addition of the latest variable to a multiple linear regression is actually a benefit to the model.

5.7 Outliers

5.8 Extrapolation

Extrapolation from a linear model should only be done with extreme caution, or not at all. The data show a correlation between two or more variables within the bounds of the data set, moving beyond those bounds may invalidate the model.

A good example of this is exhibited by springs and Hooke's Law. A spring may respond elastically within a particular range of applied force, but beyond that range (above the yield strength) the spring will begin to exhibit plastic deformation, and not return to its original shape.

5.9 Errors of Estimated Coefficients

5.9.1 Prediction Bands

5.9.2 Confidence Intervals

5.10 Checking Assumptions Using Graphs

5.11 Model Selection Strategies

5.11.1 The R^2_{adj} Approach

5.11.2 The p-Value Approach

5.12 Quick Methods and Shortcuts

In simple linear regression, the slope $\hat{\beta}_1$ can be estimated to be:

$$\hat{\beta}_{x_1} \approx \frac{s_y}{s_{x_1}} R \tag{5.25}$$

The intercept point $\hat{\beta}_0$ can then be estimated from equation 5.12.

Part IV

Supervised Algorithms I: Classification

Chapter 6

Logistic Regression

6.1 Introduction

Logistic regression is a relatively simple method for binary classification. Multi-class logistic regression is possible, but usually avoided in favor of discriminant analysis, SVM, or some other more advanced method. Logistic regression is common because it is relatively straightforward to interpret: the function computes the natural log of the odds that a particular observation belongs to one class or the other. If we assume that the log odds are a linear function of the components of x, we can use the same approach as in Chapter 5 to train the model.

$$ln\Big[\frac{p(X)}{1-p(X)}\Big] = \beta_0 + \beta_1 x \tag{6.1}$$

The odds at which observation x is assigned to one class or the other can be specified based on circumstance (l), but the best estimator in terms of accuracy is when that limit is set to log odds = 0, equivalent to odds = 1, and $p(X) = 0.5$. Usually a default value is specified for the special case when $p(X) = l$.

Probability vs Odds

Most people are familiar with the numerical representation of probability. If something has a 10% probability of occurring, then it has a 1/10 chance of occurring. Odds treat the same subject matter with a significantly different approach. While probability is bound numerically on the scale [0,1], odds are represented numerically on a scale $[0, \infty]$. If something has a 1 in 10 chance of occurring, then the odds are 1 to 9, denoted 1:9, and calculated numerically as $1/9 = 0.1\overline{11}$. Just to reiterate, an outcome C_0 with probability $p(C_0) = 0.1$, has odds $od(C_0) = 0.1\overline{11}$.

$$Od(X) = \frac{P(X)}{1 - P(x)} \tag{6.2}$$

Using odds has two considerable drawbacks: Firstly, half of the true range of odds is compressed between 0 and 1 while the other half is within all numbers 1 to ∞. Probability has a nicer symmetry that odds lack. Secondly, and more importantly, odds in the range 0-1 may be mistaken as numerical probabilities of significantly unequal value. Data scientists use odds when working with logistic regression because it is the most directly interpretable value from the function, but I will attempt to make it abundantly clear when to interpret numbers as odds or as probabilities.

6.2 Mathematics

In Chapter 5 we showed how to train models of the form:

$$f(x) = \beta_o + \beta_1 x \tag{6.3}$$

Linear functions like these are unbound in the values they are allowed to take any real value: $f(x) \in (-\infty, \infty)$. Unlike linear functions, probability is bound within the range [0,1]. We want to use the same approach to fitting the function we took in 5, so we need a way to translate the bound bound function P(X) into the unbound function $f_{lin}(x)$. We do this by calculating the odds instead of the probability, since the odds are bound on only one side $Od(X) \in [0, \infty]$. Finally, we notice that the natural logarithm (or any logarithm for that matter) transforms a number bound on $[0, \infty]$ into an unbound number:

$$ln(x) \in [0, \infty), \text{ when } x \in [1, \infty) \tag{6.4}$$

$$ln(x)] \in (-\infty, 0], \text{ when } x \in (0, 1] \tag{6.5}$$

This is why we can set the log of the odds equal to a linear function. If solve equation 6.1 for the probability, we find that we've set the probability equal to the logistic function, also known as the logit function, and we've fitted the exponent to a linear function of the feature set X.

$$
\begin{aligned}
ln\Big[\frac{p(X)}{1-p(X)}\Big] &= \beta_0 + \beta_1 x \\
\frac{p(X)}{1-p(X)} &= e^{\beta_0+\beta_1 x} \\
P(X) &= e^{\beta_0+\beta_1 x}(1-P(X)) \\
P(X) &= e^{\beta_0+\beta_1 x} - e^{\beta_0+\beta_1 x}P(X) \\
P(X)(1+e^{\beta_0+\beta_1 x}) &= e^{\beta_0+\beta_1 x}
\end{aligned}
$$

$$P(X) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}} \tag{6.6}$$

Part V

Supervised Algorithms III: Algorithms for Classification and Regression

Chapter 7

Naïve Bayes

Naïve Bayes, also called Idiot Bayes, is an extremely powerful method of probability calculation, and therefore prediction. It is powerful because it is insensitive to even severe violations of the assumption of independence. Bayes' Theorem can be thought of as the logical extension of a tree diagram, so we will digress for a moment to get an intuitive understanding of Bayes[3].

7.1 Probability

Calculating **conditional probability**:

$$P(A|B) = \frac{P(A,B)}{P(B)} \tag{7.1}$$

Or equivalently, the **general multiplication rule of probabilities**[3], also known as the **chain rule of probability**[1]:

$$P(A,B) = P(A|B) * P(B) \tag{7.2}$$

7.1.1 Tree Diagrams

A common example of conditional probability is the outcome of medical tests. The probability that someone tests positive or negative for breast cancer (BC) given their situation of either being afflicted or cancer free is usually known, determined experimentally or thorough research.

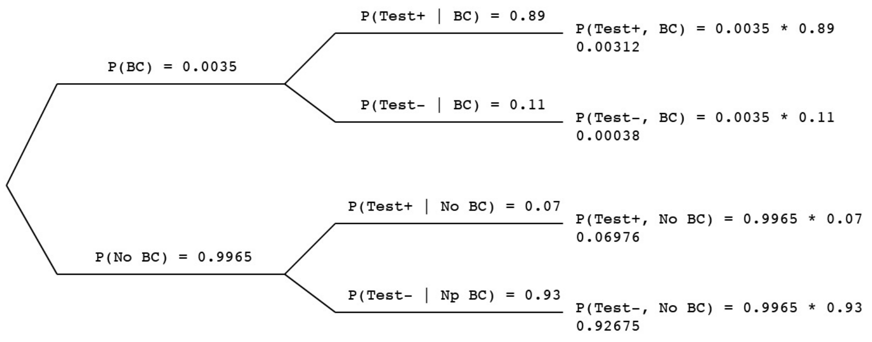

Figure 7.1: A tree diagram showing all possible outcomes of a test for breast cancer.

However, this calculation is not applicable to the real world. We don't know if someone has BC or not, only whether their test result was positive or not. Instead of calculating P(Test+|BC), P(Test+|NoBC) we want to know P(BC|Test+). Using Bayes' Theorem, the tree diagram above contains all the information necessary to calculate P(BC|Test+). The probability that someone has breast cancer given a positive test can then be calculated:

$$P(BC|Test{+}) = \frac{P(BC, Test{+})}{P(Test{+})}$$

$$P(BC, Test{+}) = P(BC) * P(Test + |BC)$$
$$P(BC, Test{+}) = 0.89 * 0.0035 = 0.00312$$

$$P(Test{+}) = P(Test{+}, BC) + P(Test{+}, NoBC)$$

$$P(Test{+}, NoBC) = P(NoBC) * P(Test + |NoBC)$$
$$P(Test{+}, NoBC) = 0.9965 * 0.07 = 0.06976$$

$$P(Test{+}, BC) = P(BC) * P(Test + |BC)$$
$$P(Test{+}, BC) = 0.035 * 0.89 = 0.0312$$

$$P(Test{+}) = 0.00312 + 0.06976 = 0.07288$$

$$P(BC|Test{+}) = \frac{0.00312}{0.07288} = 0.04281,\ 4.3\%$$

So the probability that someone has cancer, even given a positive test result is only 4.3%. This calculation is predicated on the fact that the sum of all the potential outcomes of some conditional probability must equal unity.

$$\sum_{i=0}^{n} P(A_i|B) = 1 \tag{7.3}$$

Where: $A_0, A_1, A_2...A_n$ represent all potential outcomes given B.

7.2 Bayes' Thorem

At its essence Bayes' Theorem is merely the expansion of equation 7.1. It is more understandable if taken piecewise. If we want to know $P(A_0|B)$ then from equation 7.1 we have:

$$P(A_0|B) = \frac{P(A_0, B)}{P(B)} \tag{7.4}$$

The top of the fraction is solved by equation 7.2:

$$P(A_0, B) = P(B|A_0)P(A_0) \tag{7.5}$$

The trick is in computing the denominator of equation 7.4. The logic behind equation 7.6 is that $P(B)$ is equal to the sum of all possibilities in which B occurs. Since from equation 7.2:

$$P(B, A_i) = P(B|A_i) * P(A_i)$$

If we know the probability the each possible instances of A_i: $[A_0, A_1, A_2...A_n]$, and the probability of B occurring given any particular A_i, then we can compute:

$$P(B) = \sum_{i=0}^{n} P(B|A_i)P(A_i) \tag{7.6}$$

By substituting equations 7.5 and 7.6 into equation 7.4 we can restate the probability as:

$$P(A_0|B) = \frac{P(B|A_0)P(A_0)}{\sum_{i=0}^{n} P(B|A_i)P(A_i)} \tag{7.7}$$

The real power of Bayes' Theorem is that we can compute the conditional probability $P(A_0|B)$ without knowing the probability of $P(B)$.

7.3 Naïve Bayes Applied

Naïve Bayes commonly usually used to classify natural language. We will take the example of classifying tweets as either about an app we're making or about something else, given the words in the tweet. We will classify a tweet as being about our app whenever:

$$P(App|word_0, word_1, word_2...) > P(NotApp|word_0, word_1, word_2...) \tag{7.8}$$

This is called the **maximum a posteriori** rule, or the MAP rule.

Bayes' Theorem indicates that:

$$P(App|words) = \frac{P(App) * P(words|App)}{\sum_{i=0}^{n} P(A_i|words)P(A_i)} \tag{7.9}$$

And

$$P(NotApp|words) = \frac{P(NotApp) * P(words|NotApp)}{\sum_{i=0}^{n} P(A_i|words)P(A_i)} \tag{7.10}$$

Where $A_0, A_1, A_2...A_n$ represents all the possible classifications of the tweet.

However, we aren't interested in the absolute probability that the tweet in question is about our app, all we're interested in is whether it's more likely that it's about our app than about something else. Therefore, since the denominators of equations 7.9 and 7.10 are the same, we can compare the numerators directly:

$$P(App) * P(words|App) > P(NotApp) * P(words|NotApp) \tag{7.11}$$

Here's the magic. If we assume that the probability of words appearing in a document are independent of one another then:

$$P(App) * P(words|App) = P(App) * \prod_{i=0}^{n} P(word_i|App) \tag{7.12}$$

and we need only determine whether

$$P(App) * \prod_{i=0}^{n} P(word_i|App) > P(NotApp) * \prod_{i=0}^{n} P(word_i|NotApp) \tag{7.13}$$

This is an exceptional assumption to make. Words are not independent of the word that came before it or comes after it. However, the error is on both sides of the MAP inequality, and so balances out on average.

7.3.1 High-Level Class Probabilities

P(App) and P(NotApp) are the a priori probabilities, i.e. if 20% of all tweets are about your app, then P(App) = 0.2, and P(NotApp) = 0.8. These values can be changed if you have an asymmetric attitude regarding **precision** and **recall**.

7.3.2 Rare Words

The approach of Naïve Bayes can be subverted if rare words (RW) appear in the test set but not in the training set. If the word hasn't been seen before then the P(RW|App) and P(RW|NotApp) = 0. This zero is propagated through the entire product in equation 7.13 and the result is a nonsensical 0 = 0.

The solution to this problem is called **additive smoothing** where a value of one is added to every word count, whether or not they've been seen before. Therefore words that have been seen n times will show a count of n+1.

Chapter 8

Support Vector Machines

8.1 ***To Do***

Chapter 9

Tree-Based Methods

Tree-based methods can be used to predict any variable type, numeric or categorical. Simple tree-based models are also relatively interpretable and a general understanding of the importance of the input features can be calculated. On the other hand, tree-based methods often (but not always) under perform in prediction accuracy when compared to SVM or generalized additive methods.

One reason why tree-based methods are relatively easy to explain and interpret is because they approximate human decision making. A tree considers a single variable at a time, adds that information to what it already knows, and then proceeds on until a conclusion is reached. Mathematically and graphically this procedure involves segmenting the predictor space into a number of simple regions, each of which can then assigned a prediction and a confidence level based on the training set.

9.1 Decision Trees

Decision trees are the foundation of all tree-based methods. The process of making a decision tree is relatively straightforward:

1. Divide the predictor space into J distinct and non-overlapping regions $R_1, R_2, ...R_j$.
2. For every test observation in region j predict the average of the training observations in region j (if numerical) or the most common class in region j (if categorical).

The goal of a decision tree is to find the regions $R_1...R_j$ that minimize the residual sum of squares error (or whatever cost function is applied).

$$RSS_{R_j} = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \tag{9.1}$$

Unfortunately it is usually computationally impractical or impossible to consider all possible partitions of the feature space into J subspaces. For this reason the algorithms used to assign regions in tree-based methods is a **top-down** and **greedy** approach known as **recursive binary splitting**.

Using this approach each split point s is chosen such that that split leads to the greatest possible reduction in RSS. In general, at any step s is chosen such that the following equation is minimized:

$$\sum_{i:x_i \in R_1} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2} (y_i - \hat{y}_{R_2})^2 \tag{9.2}$$

Where:

$$R_1 = \{X|X_j < s\} \text{ and } R_2 = \{X|X_j > s\} \tag{9.3}$$

On the first iteration the region R_j being split is the whole sample space, on the next split only one subregion of the sample space is split. This algorithm can run reasonably fast as long as the number of features is not too large.

This process is continued until some stopping criteria is reached, e.g. no region contains more than 5 samples, or until every region is pure.

9.1.1 Tree Pruning

The process described above tends to over-fit the training set, that is, the tree will be too large, contain too many splits. One strategy would be to limit the total number of splits at the outset, or to only allow splits that lead to at least a minimal improvement in RSS.

These two approaches are shortsighted however, and would lead to worsening the greedy nature of decision trees. A better strategy is to grow a very large, presumably over-fit initial tree and then prune it back to obtain a subtree with a more optimal balance of bias and variance. This is achieved by adding a regularization term to the end of the RSS equation in which the regularization parameter α is multiplied by the number of terminal nodes, $|T|$.

$$RSS_{reg} = \sum_{m=1}^{|T|} \sum_{i:x_i \in R_j} (y_i - \hat{y}_{R_m})^2 + \alpha|T| \tag{9.4}$$

This new equation adds a "cost" to adding each additional node, thus increasing $|T|$. The regularization parameter α can be tuned to minimize the average error.

Building a Pruned Regression Tree

1. Use recursive binary splitting to graw a large initial tree, stopping when each terminal node has some minimum number of observations.
2. Apply complexity regularization to the large tree for a range of values of α and obtain obtain a series of optimal subtrees as a function of α
3. Use K-fold cross-validation to choose α that performs best (minimizes mean error).
4. Return the subtree from step 2 that corresponds to the chosen value of α.

9.1.2 Classification Trees

A classification tree is very similar to a regression tree, but instead of returning a numerical response the tree returns a categorical response. In order to minimize error the tree should respond with the most common class at each terminal node.

A natural measurement of the accuracy of classification trees would be the **classification error rate**:

$$E = 1 - max(\hat{p}_{mk}) \tag{9.5}$$

Where $\hat{p}_{mk}$ represents the proportion of training observations in the m^{th} region belonging to the k^{th} class. However, in practice, the error rate is generally not sensitive enough to successfully grow optimal trees, and two other measurements are preferable:

The **Gini Index**:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{9.6}$$

The Gini Index is referred to as a measure of nodal purity. An alternative is the **cross-entropy**:

$$D = -\sum_{k=1}^{K} \left\{ \hat{p}_{mk} \log(\hat{p}_{mk}) \right\} \tag{9.7}$$

It turns out that the Gini Index and the cross-entropy are quite similar numerically. If prediction accuracy is the final goal, the classification error rate should be used.

9.2 Bagging, Random Forest, and Boosting

Decision trees tend to be non-robust, meaning that their shape and predictive capacity can change strongly based on the particular training set they learn from. Bagging, Boosting and Random Forest are three methods intended to both increase the predictive capability of trees and increase the robustness of the method.

9.2.1 Bagging

Bagging is a method by which many models are trained and then an average or majority vote is taken to decide each test example. Generally we do not have access to many sets of training data, so a method called **bootstrapping**, is used to generate an arbitrary number B of training sets from a single training set, where each bootstrapping set $b \subset B$. When the average outcome of these B trees is taken, the variance is reduced to σ^2/B, i.e. the standard deviation is reduced by a factor of $\sqrt{B}$.

9.2.2 Random Forest

A problem often encountered by bagged tree models is that all of the trees will be highly correlated. In a situation where there exists one very strong indicator and many weak indicators, every (or nearly every) tree in the bagged model will initially split along the strong indicator. Random Forest prediction is very similar to bagging, except that each tree is only allowed to consider a random subset p of the feature set M. This approach is meant to disrupt the correlation of the trees in a bagged model, since most of the trees in the random forest will not be allowed to consider the strong indicator. Usually:

$$p = \text{ceil}[\sqrt{M}] \text{ or } p = \text{ceil}[\ln(M)] \tag{9.8}$$

Random Forest devolves into bagging when $p = m$.

9.2.3 Boosting

Boosting is another method of growing many trees in an attempt to reduce the variance of the decision tree model. Boosting involves growing trees sequentially instead of in parallel like bagging or random forest.

Boosting involves growing an initial tree to the initial data set, and then fitting subsequent trees to the residuals of the first tree. The advantage of a boosted tree is that it is a **slow learner**, meaning that it doesn't learn very much at each step. Generally slow learning algorithms tend to avoid over-fitting training sets better than **fast learners**.

Boosting has three tuning parameters:

1. B - the number of trees.
2. λ - the shrinkage parameter. Usually 0.01 - 0.001, this parameter controls how fast the algorithm learns.
3. d - the number of splits in each tree.

Boosted trees usually perform well when d is small, often $d = 1$ is optimal.

9.3 Advantages and Disadvantages of Tree-Based Methods

Tree-Based models assume a structure in the data being modeled. Specifically:

$$f(X) = \sum_{m=1}^{M} c_m \cdot 1_{(X \in R_m)} \tag{9.9}$$

This is similar to how linear models assume a structure of:

$$f(X) = \beta_0 + \sum_{j=1}^{m} \beta_j x_j \tag{9.10}$$

The tree-based approach assumes that splits occur along paths parallel to the axes of the features. This is apparent if we think of a graphical representation of what is occurring at each interior node of the tree. If a particular feature set has a linear relationship to the parameter being modeled then linear regression will strongly outperform random Forest regression, likewise, if the relationship is complex and/or strongly non-linear, then tree-based methods will outperform linear regression.

Advantages

- Trees are very easy to explain to people.
- Decision trees mirror human decision-making in some ways.
- Simple trees can be displayed graphically and easily interpreted by even non-experts.
- Trees can relatively easily handle complex feature sets, including correlated features, qualitative features, and unimportant features.
- Trees can report feature importance, making the inner workings of the model more transparent.

Disadvantages

- Trees generally do not perform as well as other classification and regression techniques.

Glossary

greedy - an algorithm that always takes the best possible step towards its goal at every stage. It has no ”foresight” or ability to ”see” how less good decisions now might lead to more optimal solutions later.

Chapter 10

Ensemble Methods

10.1 ***To Do***

Part VI

Time Series

Chapter 11

Holt-Winters Forecasting

11.1 ***To Do***

Part VII

Advanced Algorithms

Chapter 12

Neural Networks

12.1 ***To Do***

Part VIII

Supporting Information

Chapter 13

Distance Metrics

13.1 Euclidean Distance

This is the way we usually measure distance. It is the straight-line path in n-dimensional space.

$$Dist_{Euc} = \sqrt{\sum_{i=0}^{n}(p_i - q_i)^2} \tag{13.1}$$

Where $\mathbf{p},\ \mathbf{q} \in \mathrm{I\!R}^n$.

13.2 Manhattan Distance (Hamming Distance)

Manhattan distance is the grid distance between two points. If a straight-line trajectory doesn't make sense we can measure the distance as if we were driving a cab in Manhattan, and only take streets perpendicular to the coordinate axes. This is the same as measuring the legs of a right triangle when the Euclidean distance is measuring the hypotenuse.

$$Dist_{Man} = \sum_{i=0}^{n}(p_i - q_i) \tag{13.2}$$

Where $\mathbf{p},\ \mathbf{q} \in \mathrm{I\!R}^n$.

13.3 Cosine Distance

Cosine distance is an asymmetric distance metric. When one kind of similarity is more important than another similarity, cosine distance may be a good choice. For example if we are grouping customers then similar purchases are more important than similar non-purchases.

$$Dist_{Cos} = \frac{n_m}{\sqrt{n_0}\sqrt{n_1}} \tag{13.3}$$

Where:
n_m is the number of matched purchases.
n_0 is the total number of items purchased by customer 0.
n_1 is the total number of items purchased by customer 1.

13.4 Jaccard Similarity

Jaccard similarity is the ratio of the magnitude of $A \cap B$ to the magnitude of $A \cup B$. It is like the ratio of the size of the INNER JOIN table to the size of the FULL OUTER JOIN table.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{13.4}$$

Chapter 14

Application Program Interface

14.1 Introduction

Most information from http://restful-api-design.readthedocs.org/en/latest/

An Application Program(ming) Interface (API) is the method through which programs can talk to each other. APIs are used all over the computer science world, whenever programs interact. At its most basic, the job of an API is to access the **application state** and make that state available to outside entities that made that request. Operating systems provide APIs so that programs can use the hardware and OS and interact with each other. GUIs use APIs to interact with the backend code. Most useful for data scientists, many websites provide APIs, for example Amazon and eBay allow developers to use the existing retail infrastructure to create specialized web stores.

It may seem counter intuitive that large corporations would allow so much access to the data and services available from their websites for free. However, by making their services widely available these web companies hope to capture as large a marketshare as possible, and thus drive their own growth.

14.2 RESTful APIs

RESTful APIs conform to a set of standards that are more-or-less universally interpretable by machines. REST stands for Representational State Transfer, which essentially refers to the style of web architecture that has many underlying characteristics and governs the behavior of clients and servers.

RESTful APIs include:

- Base url and collection
- An interactive media type (usually JSON)
- 4 operations (GET, PUT, POST, DELETE)
- Driven by hypertest requests (HTTP)

14.2.1 RESTful API Operations

METHOD	SCOPE	Description
GET	C	Reteieve all resources in a collection
GET	R	Retrieve a single resource
HEAD	C	Retrieve headders of all resources in a collection
HEAD	R	Retrieve the headder of a single resource
POST	C	Create a new resource in a collection
PUT	R	Update an entire resource
PATCH	R	Update only particular items in a resource (PROPOSED)
DELETE	R	Delete a resource
OPTIONS	ANY	Return available methods and other options

14.2.2 Example

GET https://api.instagram.com/v1/users/10

Operation	GET
url	https://api.instagram.com/v1
Collection	users
Element	10

14.2.3 API Resources

Resources are the fundamental concept behind APIs, they are like objects in an Object Oriented language. Resources may be gathered together into **collections**. all of the resources in a particular collection must have identical structure, simlar to instances of a particular python class. Resources that exist alone, as part of no class are called a **singleton resource**. Collections can exist at the top-level of an API, or within a particular resource.

14.3 API Languages

14.3.1 JSON

JSON stands for JavaScript Object Notation. It is a lightweight data-interchange format that can be passed between applications. It is easy for machines to parse and read. JSON in particular is also relatively easy for humans to parse and read. JSON strings operate much like python dictionaries where they have key : value pairs. I have heard and read several times the JSON is the best language for RESTful APIs at the moment.

JSON has three data types:

Scalar	Can be a number, string, boolean, or null. Critically it is a single value.
Array	Contains an ordered list of values of arbitrary type.
Object	Are an unordered set of key:value pairs. Keys are strings, values can be any dtype.

14.3.2 XML

Chapter 15

Set Theory, Relational Algebra, and Structured Query Language

15.1 Introduction

Structured Query Language (SQL) is the most common syntax for searching databases. SQL is based on set theory and relational algebra, though in the real world, many if not most databases do not follow the tenants of these branches of academia. Much of the skill that employers are looking for in advanced SQL users is how to interact with databases that don't follow these rules with relative ease and rigor.

15.2 Set Theory

Set theory is the branch of mathematical logic that studies sets, which, informally, are collections of objects. SQL is largely based on set theory, and the best maintained databases conform to the best practices of set theory. Of the items listed below, only 1 and 2 are adhered to more or less strictly, all others can and are often violated, especially in small and/or data-naïve, companies.

1. **columns** should represent a unique category of data
2. **order of columns** shouldn't matter
3. **tables** should be the smallest logical subset of data
4. **rows** should represent a unique category of data

5. **attributes** should be unique to the table in which they reside.
6. **primary keys** should be never be repeated.

15.2.1 Basic Concepts and Notation

In	$B \in A$	If o is a **member** (or **element**) of A, then $o \in A$. Sets can also be objects, so set B can also be a member of A, $B \in A$.
Proper Subset	$B \subset A$	A proper subset means that B is contained within A and A has at least one element not in B. So $B \subset A, B \neq A$.
Subset	$B \subseteq A$	B is contained within A including the situation where $B = A$
Union	$B \cup A$	All space in B and A, including overlapping space.
Symmetric Difference	$B \triangle A$	All space in B and A not including overlapping space. The complement of intersection.
Relative Complement	$A \setminus B$	The part of A not contained within B. If $A \subset B$, $A \setminus B$ will return NULL
Intersection	$B \cap A$	The space included in both A and B. The complement to symmetric difference.
Cartesian Product	$B \times A$	All possible ordered pairs of A and B.
Power Set	2^A	All possible subsets of A including the empty set and A.

15.3 Structured Query Language

SQL is the most common database query language. It was written to be easy to read and write by humans. SQL does is case and white space agnostic, but convention is to capitalize all SQL commands and not column, table or db names.

SQL is popular because:

1. Its semantics are easy to learn
2. Can directly access data without graphical representation
3. It is easy to replicate and verify SQL queries compared to spreadsheet programs.

15.3.1 Best Practices

1. The most common mistake SQL users make is accidentally running excessively long queries. This can be avoided by running queries on subqueries that have a LIMIT imposed on them. Since inner queries are run first, and outer queries are only run on whatever is SELECT(ed) from the inner query, this technique can ensure that you don't accidentally query millions or billions of rows.
2. Use the EXPLAIN keyword prior to execution, running this command will tell you how many rows will be searched by this query, and give you some idea of the amount of time required to run. This will often catch improperly specified queries that will take much longer than expected.
3. Always select only the minimum amount of data you require. The less data you query, the faster your query will run, and the less hardware you will take. This is especially true when joining tables.
4. Making ER Diagrams and Relational Schema prior to SQL querying will greatly improve query accuracy. These diagrams are especially important before you are familiar or have memorized the database structure.
5. Write queries from the inside out. This way you ensure that your outer queries are acting on the table you expect it to.
6. Clean data prior to analysis. Duplicated rows are multiplied by JOINs, so cleaning prior to query will reduce database load.

15.3.2 Entity Relationship Diagrams

15.3.3 Relational Schema

15.3.4 Database Metadata

SHOW tables	Lists the tables contained within the current db
DESCRIBE <table>	Lists the columns contained within <table>

15.3.5 Basic Queries

Basic queries have a simple syntax that was written to be easy to be easily read by humans. The issue is that this syntax does not and cannot be directly translated as-is to a computer database for analysis. The order must be rearranged.

In the basic query, the first command SELECT, identifies the columns the user is interested in, and the third command FROM, identifies the table where the columns exist. It is quickly obvious that this syntax cannot be directly "read" by computers, a database system must know which table to search before it knows what columns it's looking for.

Another item to keep in mind is that LIMIT is the final command considered by SQL. If your query includes an aggregation, grouping, or down-selection, these will be executed prior to LIMIT being applied, and so the query may take a long time to execute even if only a few rows are actually being returned by the query. Look to section 15.3.1 for solutions to this issue.

Syntax Order	**Runtime Order**
SELECT	FROM
DISTINCT	WHERE
FROM	GROUP BY
WHERE	HAVING
GROUP BY	SELECT
HAVING	DISTINCT
ORDER BY	ORDER BY
LIMIT	LIMIT

Command	**Description**
SELECT	Indicates which columns are to be returned
DISTINCT(<column>)	Identifies columns that should be unique in the output. Commonly these columns araer the target of aggregation.
FROM	Indicates the table that the query is acting on, this is where nested queries come from.
WHERE	Conditional that can be selected on.
GROUP BY	This is how aggregation functions know what groups to aggregate over.
HAVING	Conditional that can be applied to aggregated columns.
ORDER BY	Order the output by values in a particular column, ASC by default, DESC is optional.
LIMIT	reduces the output to a small number of rows. Can be used to make fast queries of subsets from databases, but should be careful.

15.3.6 Aggregation Functions ***ToDo***

15.3.7 Joins ***ToDo***

15.3.8 Subqueries ***ToDo***

15.3.9 Logical Expressions ***ToDo***

Bibliography

[1] Foreman, John, *Data Smart: Using Data Science to Transform Information into Insight*, Indianapolis: Wiley, 2014. Print.

[2] Mason Gallow, *General Assembly Data Science*, Fall 2015.

[3] Diez, David; Barr, Christopher and Cetinkaya-Rundel, Mine. *OpenIntro Statistics*, 3rd ed. www.openintro.org, Updated: Jan. 13, 2016