Data Science Algorithms

Peter Meleney

Data Science Algorithms This is a repository for what I learn about algorithms I use for data science. Each section should contain a conceptual overview of the algorithm and the basic math behind its implementation. I also intend to include examples in R and Python [1].

Created: March 4, 2016
Last Updated: March 14, 2016

Contents

Part I

Unsupervised Algorithms I: Dimensionality Reduction

Chapter 1

Principle Component Analysis

1.1 ***To Do***

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: Classification-Only Algorithms

Chapter 5

Logistic Regression

5.1 ***To Do***

Part IV

Supervised Algorithms II: Regression-Only Algorithms

Chapter 6

Linear Regression

6.1 ***To Do***

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.

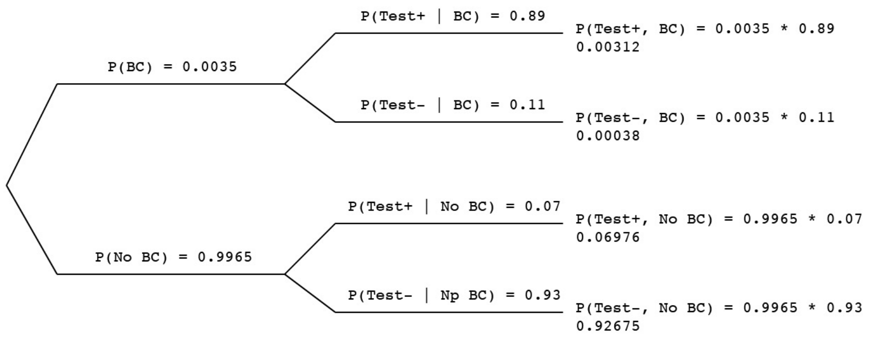

Figure 7.1: A tree diagram showing all possible outcomes of a test for breast cancer.

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:

$$P(BC|Test+) = \frac{P(BC, Test+)}{P(Test+)} \tag{7.3}$$

$$P(BC, Test+) = P(BC) * P(Test+|BC) = 0.89 * 0.0035 = 0.00312 \tag{7.4}$$

$$P(Test+) = P(Test+, BC) + P(Test+, NoBC) \tag{7.5}$$

$$P(Test+, NoBC) = P(NoBC) * P(Test+|NoBC) = 0.9965 * 0.07 = 0.06976 \tag{7.6}$$

$$P(Test+) = 0.00312 + 0.06976 = 0.07288 \tag{7.7}$$

$$P(BC|Test+) = \frac{0.00312}{0.07288} = 0.04281,\ 4.3\% \tag{7.8}$$

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.

$$\sum_{i=0}^{n} P(A_i|B) = 1 \tag{7.9}$$

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

7.2 Bayes' Thorem

Bayes' Theorem works exactly like equation 7.1, it takes P(A_0|B), then re-normalizes that outcome to the sum of all probabilities ($A_0 ... A_n$) given B.

$$P(A_0|B) = \frac{P(B|A_0)P(A_0)}{\sum_{i=0}^{n} P(B|A_i)P(A_i)} \tag{7.10}$$

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:

$$P(App|word_0, word_1, word_2 ...) > P(NotApp|word_0, word_1, word_2 ...) \tag{7.11}$$

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

Bayes' Theorem indicates that:

$$P(App|words) = \frac{P(App) * P(words|App)}{\sum_{i=0}^{n} P(A_i|words)P(A_i)} \tag{7.12}$$

And

$$P(NotApp|words) = \frac{P(NotApp) * P(words|NotApp)}{\sum_{i=0}^{n} P(A_i|words)P(A_i)} \tag{7.13}$$

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 7.12 and 7.13 are the same, we can compare the numerators directly:

$$P(App) * P(words|App) > P(NotApp) * P(words|NotApp) \tag{7.14}$$

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

$$P(App) * P(words|App) = P(App) * \prod_{i=0}^{n} P(word_i|App) \tag{7.15}$$

and we need only determine whether

$$P(App) * \prod_{i=0}^{n} P(word_i|App) > P(NotApp) * \prod_{i=0}^{n} P(word_i|NotApp) \tag{7.16}$$

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 7.16 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

Random Forest

9.1 ***To Do***

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 $\mathbf{p},\ \mathbf{q} \in \mathbb{R}^n$.

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 $\mathbf{p},\ \mathbf{q} \in \mathbb{R}^n$.

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 ***To Do***

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*, 3^{rd} ed. www.openintro.org, Updated: Jan. 13, 2016