FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional an Multimedia Datasets

A TERM PAPER PRESENTED
BY
CHRISTOPHER L. CAMPELL & PATRICK BEEKMAN
TO
THE DEPARTMENT OF COMPUTER SCIENCE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE
IN THE SUBJECT OF
COMPUTER SCIENCE

APPALACHIAN STATE UNIVERSITY
BOONE, NORTH CAROLINA
APRIL 2017

Research advisor: Dr. Mohan Christopher L. Campell & Patrick Beekman

FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional an Multimedia Datasets

Abstract

Fast searching in traditional multimedia databases is useful for Querying by Example, finding all pairs and finding best-matches. One idea to do this is to map these multimedia objects into points in a k-d space and use k-feature extraction functions given by a domain expert. Unfortunately unless you are a domain expert it is relatively difficult to design these feature extraction functions. Furthermore even with a domain expert it is still difficult to map these functions and distances into points for visualization.

This is where the FastMap algorithm comes in. It is a fast algorithm to map objects into a point in some k-dimensional space while preserving the dis-similarities. Two benefits to this mapping is that it allows for quick spatial querying and gives an easy way to visualize the objects which can be useful for data-mining and clustering.

This project describes the motivation and background for using the FastMap algorithm to cluster objects, some notations and preliminary concepts, implementation details, and our results on their data set and a new data set.

Contents

Listing of figures

1
Introduction and Motivation

Given the wide range of data computer scientists deal with from traditional data sets to multimedia and 'exotic' data, it would be nice to visualize and query all this data using a simple tool that works on all types of data sets. In the past people have relied upon domain experts to create feature extraction functions. This has to be done for each specific data set no matter the type, which takes time and money. Some data sets can be especially hard to figure out what the features should be or there may be some complex warping of the data in order to make it comparable. These things make it even harder to create these feature extraction functions. This is the exact motivation behind the FastMap paper [1].

Using only a distance function that a domain expert may derive and the data (objects) we can create k-dimensional points and map them into a k-d space. Using this technique we can also find similar objects given a query, find the pairs of objects most similar to each other, and visualize the data to determine clusters.

1.1 Motivation

Some specific applications which motivated the creation of the fast map algorithm:

1.1.1 Medical databases

These databases store tons of complex multimedia data that is hard to create a lot of complicated distance functions. This data is often times misaligned from other similar data (eg. bone alignment) so there must be some complex warping of the data to align it and this makes the distance functions magnitudes more complex. This in turn makes mapping these into points in a specific k-dimensional space especially hard.

1.1.2 Time series

Fields that use time series such as businesses and scientists that look at financial data, stock prices, weather patterns or environmental data would benefit from being able to easily compare two series. This could be used to *'find companies whose stock prices move similarly'* or *'Look at previous days whose weather was similar'*.

1.1.3 Data Mining

Given a data set of patient records containing info such as gender, age, blood-pressure, weight, etc. it would be helpful to detect clusters or correlations among symptoms. This could be useful in screening for cancer or checking the likelihood you have a certain disease.

1.2 Previous Work

Some previous work that inspired and was helpful in the formulation of the fastmap algorithm.

1.2.1 Multi-Dimensional Scaling (MDS)

MDS is used to discover the underlying structure of the data using the dis-similarity information among them. This iterative algorithm works by trying to minimize a stress function which is defined as average relative error that the distances in k-d space suffer from. It starts by randomly placing the data into k-d space, then computing the dis-similarity between every point, and moving a single point to minimize its dis-similarity with the rest. It will then iterate through all points doing this procedure.

MDS has been used in lots of practical applications but it's running time is really slow on the magnitude of $O(N^2)$. It is also very bad at querying by example since we would have to search/add a new item which would take another $O(N)$ at best which is just as bad as a sequential scan.

1.2.2 Dimensionality Reduction Techniques

The optimal way to map n-dimensional points into k-dimensional points is the *Karhunen-Loève* ('K-L') transform. It works by computing the eigenvectors of the covariance matrix and projects each onto the k eigenvectors that have the largest eigenvalues. This is often used in pattern matching to choose relevant features. Two problems with this are that it can not be applied on the distance case and it may be slow on large data sets with lots of features.

2
The FastMap Algorithm

The FastMap algorithm is impressive due to the simplicity of its design and its linear efficiency. The purpose of the algorithm is to find N points in a user-defined k-d space, whose Euclidean distances match the distances of an initially provided N x N distance matrix. The FastMap algorithm performs projections onto a line that runs orthogonal through the original dataset. To do this projection we will first choose two objects O_a and O_b which will be known as the pivot objects, the line between these two is how we will base the projections. To find the ideal pivot objects we will use Algorithm 1 stated below. For best results steps 2 and 3 should be repeated a constant number of times, this can be done without affecting the running time. The paper recommends repeating steps 2 and 3 for five iterations.

2.1 Defining a Pivot Axis:

Algorithm 1 (based off of Equation ??) describes a heuristic approach to a k-nearest-neighbors maximization problem. The objective of algorithm 1 is to find a line in which the distance between projections is maximized [1]. In order to accomplish this, the pivot objects (rows in the original sample space) O_a and O_b must be selected. The heuristic algorithm 1 is able to select these pivot objects in linear time.

2.1.1 The *choose-distant-objects* Algorithm:

Algorithm 1: *choose-distant-objects* $(O, dist())$

```
begin
    1. Choose arbitrarily an object, and let it be the second pivot object $O_b$.
    2. Let $O_a$ = (the object that is furthest apart from $O_b$) according to the
       distance function dist().
    3. Let $O_b$ = (the object that is furthest apart from $O_a$).
    4. Report the objects $O_a$ and $O_b$ as the desired pair of objects.
end
```

2.1.2 Complexity Analysis:

The complexity of the aforementioned Algorithm 1 is $O(N)$ (where N is the dimensionality of the distance matrix D). This is due to the heuristic nature of the algorithm; it only wishes to estimate an ideal solution without computing it directly. By repeating steps 2 and 3 of Algorithm 1 a user-specified number (k) times; the linearity of the heuristic is maintained. A higher value of k indicates a desire to ensure that O_a and O_b are a maximum distance apart. Since Algorithm 1 is a heuristic algorithm; it is not guaranteed to maximize distance for all values of

k (particularly low ones). The research paper indicates that a value as low as $k = 5$ satisfies convergence and sufficiently maximizes $D(O_a, O_b)$ [1].

2.2 The Algorithm:

The goal of the FastMap Algorithm is to map objects into points in a k-d space using only an $N\,x\,N$ distance matrix. The FastMap Algorithm utilizes Algorithm 1 to identify two pivot objects O_a and O_b. The line (AB) is formed between points O_a and O_b perpendicular to the desired hyper-plane of projection H. The Cosine Law (Equation 3.1) is utilized to project the objects onto line (AB). Once the objects have been projected onto the pivot line (AB), the distance $D'()$ allows for a projection onto a line orthogonal to the pivot-line which lies along the hyper-plane H. The FastMap algorithm is recursive, and it repeatedly performs the above transformations until the dimensionality of the projection space is reduced to the desired value of k.

2.2.1 The *FastMap* Algorithm:

Algorithm 2: Algorithm *FastMap(k,D(),O)*

```
Data: $N$ x $k$ array $X[]$ /* At the end of the algorithm, the $i$-th row is the image of
      the $i$-th object. */
      2 x $k$ pivot array $PA[]$ /* Stores the ids of the pivot objects - one pair per
      recursive call. */
      int col# = 0; /* points to the column of the $X[]$ array currently being
      updated /*
begin
    1. if $(k \leq 0)$
       { return; }
       else
       {col#++;}

    2. /* Choose pivot objects */
       let $O_a$ and $O_b$ be the result of choose-dist-objects$(O, D())$;

    3. /* record the ids of the pivot objects */
       PA[1,col#] = a; PA[2, col#]=b;

    4. if $(D(O_a, O_b) = 0)$
       set X[i,col#]=0 for every $i$ and return
       /* since all inter-object distances are 0 */

    5. /* project objects on line $(O_a, O_b)$ */
       for each object $O_i$,
       compute $x_i$ using Equation 3.2 and update the global array:
       X[i,col#]=$x_i$

    6. /* consider the projections of the objects on a hyper-plane perpendicular
       to the line $(O_a, O_b)$; the distance function $D'()$ between two projections is
       given by Equation 3.3 /*
       call FastMap$(k - 1, D'(), O)$
end
```

2.2.2 Complexity Analysis:

There will be a total of k iterations of the algorithm to compute each dimension. During each iteration we will have to find the pivot objects (Algorithm 1) which as shown in section 2.1.2 takes $O(N)$ time. Then during the projections (step 5 of Algorithm 2) we will have to look at every object which is $O(N)$ again. During each iteration it has a running time of $O(2N)$ which is simply just $O(N)$, then because we have k iterations of this we get a total running time of $O(Nk)$.

3 Notations and Preliminaries

The FastMap research paper defines several necessary preliminary concepts and formulas necessary to understand the algorithm itself. Some initially confusing terminology involves the labeling of the data as an object, such that O refers to the entire $N \ x \ k$ sample space.

3.1 Formulas

3.1.1 The Cosine Law:

For any triangle $O_a O_i O_b$ the cosine law equation 3.1 gives:

$$d_{b,i}{}^2 = d_{a,i}{}^2 + d_{a,b}{}^2 - 2x_i d_{a,b} \tag{3.1}$$

3.1.2 Pythagorean Theorem:

From the Pythagorean theorem in the two rectangles $O_a E O_i$ and $O_b E O_i$, equation 3.1 can be solved for x_i the first coordinate of object O_i:

$$x_i = \frac{{d_{a,i}}^2 + {d_{a,b}}^2 - {d_{b,i}}^2}{2d_{a,b}} \tag{3.2}$$

3.1.3 Lemma One:

On the hyper-plane H, the Euclidean distance $D'()$ between projections O'_i and O'_j can be computed from the originally provided distance metric $D()$ as:

$$((D'(O'_i, O'_j)^2) = (D(O_i, O_j))^2 - (x_i - x_j)^2 \qquad i, j = 1, \ldots, N \tag{3.3}$$

3.1.4 Choose Distant Objects:

The heuristic algorithm Choose Distant Objects is a heuristic which chooses two distant objects (O_a and O_b) to serve as pivots:

3.2 Symbols

The following symbols are defined within the context of the FastMap algorithm:

- O
  - The original database.
- N
  - The number of objects in the original database.
- n
  - The dimensionality of the original space such that O is an n by m matrix representing the original dataset.

- H
  - The hyperplane perpendicular to the pivot line for which the data is to be projected upon.
- O_a
  - The row in the database O selected as a pivot.
  - In conjunction with O_b these two pivot points serve to define a line perpendicular to the hyperplane H.
- O_b
  - The row in the database O selected as a pivot.
  - In conjunction with O_a these two pivot points serve to define a line perpendicular to the hyperplane H.
- $D()$ or $D(*,*)$
  - A distance or dis-similarity function provided ideally by a domain expert.
  - The FastMap paper utilizes the Euclidean distance due to the fact that it is easily differentiable.
- $||\vec{x}||_2$
  - The length $(L_2 norm)$ of vector $\vec{x}$.
- (AB)
  - The length of line segment AB.
- X
  - The matrix output by FastMap.

  - At the end of the FastMap algorithm X is an N *x* k matrix where the i-th row is the image of the i-th object.

- *PA*

  - A 2 *x* k pivot array which stores the ids of the pivot objects (a and b), once per recursive call.

4 Extension

We sought to apply the FastMap algorithm directly to the famous Machine Learning Iris data set [2]. We wanted to see if we could identify key clusters that differentiate the species of Iris flowers evident in the data. Additionally, we sought to replicate the results in the paper by applying the FastMap algorithm to the Wine dataset [2]. Included below is the experiments performed to validate the correctness of our implementation of the FastMap algorithm.

4.1 FastMap Implementation:

We implemented the FastMap algorithm using Python 3.6.1. We were then able to use sklearn and import the predownloaded datasets (wine and iris) directly into our program. The paper also suggests that the input data be normalized for which we used sklearn's minmaxscaler to preprocess our data. All of the data was stored in numpy arrays which allowed for easy manipulation and fast matrix computations (which is what the data was). After we got our results calculated we were then able to plot the k-dimensional data easily. To do this we used matplotlib's pyplot library to quickly and neatly create a figure and plot all of the data, even allowing us to seamlessly attach the labels to the objects for easier cluster viewing.

4.2 The Wine Data Set:

We replicated the results in the paper by applying the FastMap algorithm to the Wine Datset. The Wine dataset originally contained eleven features/attributes used for classification, and one discrete finite target variable. When run through the FastMap algorithm with a desired output dimensionality of three $(k = 3)$, the figure 4.2.1 is produced. When the dimensionality of the output space is changed to $k = 2$ the figure 4.2.2 is produced.

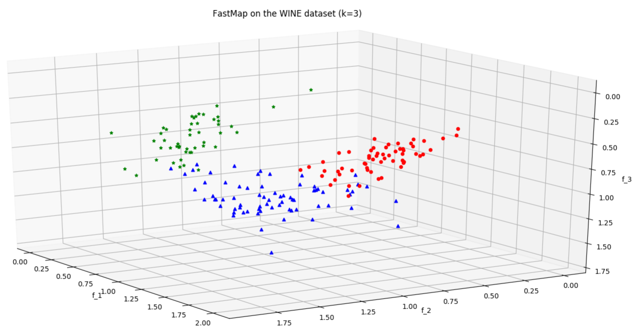

Figure 4.2.1: The FastMap algorithm when applied to the WINE dataset with $(k = 3)$.

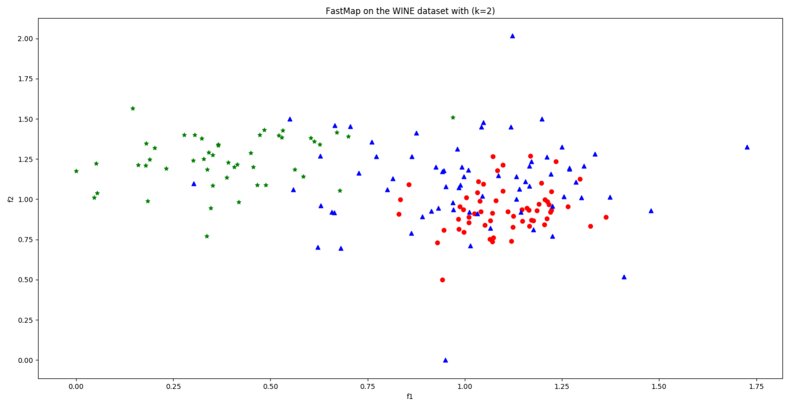

Figure 4.2.2: The FastMap algorithm when applied to the WINE dataset with $(k = 2)$.

4.3 The Iris Data Set:

To extend upon the results in the paper we applied the FastMap algorithm to the Iris Data Set [2]. The results of applying FastMap to the Iris Data Set with $k = 3$ are depicted in figure 4.3.1. When the dimensionality of the output space is changed to $k = 2$, the figure 4.3.2 is produced.

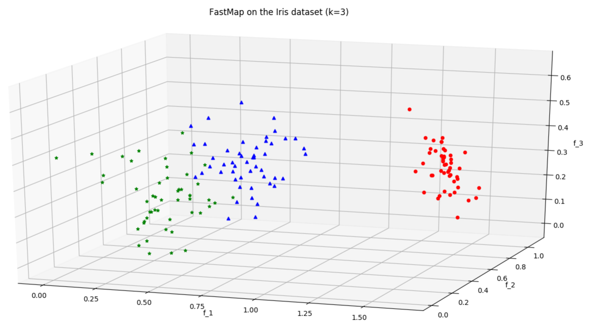

Figure 4.3.1: The FastMap algorithm when applied to the IRIS dataset with $(k = 3)$.

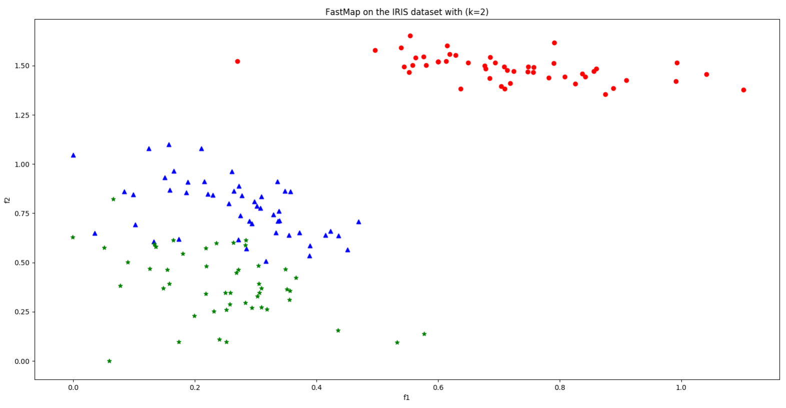

Figure 4.3.2: The FastMap algorithm when applied to the IRIS dataset with $(k = 2)$.

5 Conclusion

With data it can often times be helpful to perform queries such as *'Query by example'*, *'all pairs'*, and nearest neighbors. Given the complex data that computer scientists deal with it can be difficult to map it into some k-d space. Previous works show how to do this with domain expert knowledge, Multi-Dimensional Scaling (MDS), and the *Karhunen-Loève* ('K-L') transform, but they all come with their own problems. That is why the authors of this paper created the FastMap algorithm. It allows for easy mapping of objects into a k-dimensional space which allows for all of those queries to be performed and easy visualization which we can view clustering on.

We were able to recreate the results that they got on the wine dataset without domain knowledge using the simple Euclidean distance function. On top of this we also appplied the algorithm to the well known iris dataset and were able to correctly cluster the types of iris flowers.

References

[1] Faloutsos Christos and King-Ip Lin. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. pages 163–174, San Jose, CA USA, 1995. Association of Computing Machinery (ACM), University of Maryland.

[2] M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Colophon

This term paper was typeset using LaTeX, originally developed by Leslie Lamport and based on Donald Knuth's TeX. The body text is set in 11 point Arno Pro, designed by Robert Slimbach in the style of book types from the Aldine Press in Venice, and issued by Adobe in 2007. A template, which can be used to format a PhD thesis with this look and feel, has been released under the permissive MIT (X11) license, and can be found online at github.com/suchow/ or from the author at suchow@post.harvard.edu.