CSE252C Project Report
Facial Attractiveness Scoring

Hani Altwaijry

Abstract

The evaluation of attractiveness followed a scoring methodology rather than the usual classification one has not been attempted. This is desired for the sole reason that any person would be able to further decide between the attractiveness of two faces even if both are deemed attractive. In modelling this scenario, a continuous scoring function is more appropriate. In this project, we implemented this scoring using Support Vector Machines to perform Regression on SIFT and Eigenfaces features extracted from the given data. The best results show a mean squared error of 0.082, which is considered promising in our case. With further refinement, such a tool will turn to be very useful in the hands of Internet dating services.

1 Introduction

What makes a face a beautiful one, is a hard question. Facial attractiveness is judged differently by each individual, although, many can agree on several beautiful faces, however, individual differences in taste exist. Today, millions go on web-based dating services and carry on browsing a huge number of other user profiles. In that process, users mentally classify the profile owners as attractive or unattractive, and further differentiate between attractive ones in terms of whom is the most attractive. These dating services provide no means for any automated filtering based on personalized attractiveness prediction as such tools either do not exist, or perform very poorly.

According to [12], recent research in this area [1, 8] was tackling the problem from a different angle, in which the prediction of whether a face is attractive or not was subject to universal taste. In general, recent research in this area have been little, and focused on using different feature sets for faces with different classification algorithms.

In this project, another attempt at tackling this problem has been conducted mainly through the employment of SIFT, PD-SIFT [6], and Eigenfaces [11] features with Support Vector Regression to perform a continuous scoring of attractiveness. Unlike previous work, this project focused on trying to differentiate between any two faces in terms of which is more attractive with respect to the user’s taste. Thus, the primary goal was set to have a user sort a list of faces from most attractive to least attractive through comparing two faces at a time. This method prevented the user from assigning any score to a face, however, extracted his actual taste since he explicitly provided an ordering of the faces. When presented with a novel face, an attempt was made to place that face within its proper place in the ordered list. This should serve serves the project goal. Finally, the classification version of this problem was also investigated through the division of testing labels into categories for classification purposes, although the fairness of such an act is unknown.

This report is organized in the following manner. Section 2 discusses some of the related work to this project. Section 3 describes the approach of this project in tackling this
problem. Section 4 gives an overview of the dataset and data collection. Section 5 discusses the results of the experiment. Section 6 present the current future direction of this project. Section 7 concludes the project report.

2 Related Work

Relatively little work has been done in the area of personalized facial attractiveness measures. This sections provides several examples.

Whitehill, et al [12], have tackled the problem from the individuals’ perspective, and have employed various computer vision techniques that used feature sets such as Eigenface projections, Gabor filters, Edge orientation histograms (EOH), and Geometric relations with $\epsilon$-SVM. In their experiment, four classes of attractiveness were employed, and they were able to achieve 0.45 correlation in their prediction process. Finally, they also report on the best feature types used in their experiment, which were the Gabor features vs. PCA, EOH, and Geometry.

Sutic, et al [10], followed up using features as Eigenfaces and Geometric ratios with the algorithms: k-Nearest Neighbor (k-NN), Neural Networks, and AdaBoost. Their experiment performed two-class and four-class classifications. Their best results in the two-class case using Eigenfaces with k-NN. The correlation accuracy went upto 0.67 for Eigenfaces with k-NN, and 0.61 for Geometric ratios with k-NN. As for their four-class case, the best results were obtained through k-NN with accuracy of about 0.33.

Eisenthal, et al[5], used Eigenfaces and Geometric features with SVM, k-NN, and linear regression. They obtained their best results using Geometric features with SVM and with linear regression. The correlation accuracy reached 0.6.

3 Approach

In this section, the general approach towards realizing the goal of this project is presented. The approach is summarized in Figure 1.

3.1 Ordered Data vs. Categories

In the categorical approach, the faces are classified into a certain number of classes, such as "Attractive" and "Unattractive", or possibly, "Attractive", "Neutral", and "Unattractive". Such a categorical approach is suitable for reducing the set of faces presented to the user on a dating website, for example, however it fails at stating whether a user would prefer a face over another. The ordered data approach allows for extracting the preferences of the user in terms of a face being more attractive than another, while also allowing us to define certain categories through defining boundaries on the ordered data.

The discouraging aspect of having an ordered list is the cost of obtaining such a list from each user, because a perfectly ordered list would require a user to perform a large number of comparisons, in the average case the number of comparisons for a data of size $n$ is $O(n \log n)$, e.g. if we had 100 faces, we would require 664.39 comparisons on average. This fact steered the ordering approach into using a partial order of the data that preserves the following properties in related sub-lists:

$$\left([F_i, F_{j+1}, \ldots, F_k], [F_{k+1}, \ldots, F_n]\right)$$

where $F_x$ are faces.

The following is preserved:

$$F_j < F_{j+1} \leq F_k < F_{k+1}$$

Having a partial order allows us to define the fine scoring that we need while lowering the number of comparisons required. This is a good property at the sacrifice of some of the
quality of the data. The partial ordering lets us control the quality of data as opposed to
the number of comparisons.

In pursuit of obtaining a good partial order, an algorithm called Chunksort [3] was con-
sidered that solves the generalized partial sorting problem. It was then realized that the
algorithm reduces to Quicksort in cases of when there is no partial ordering. This further led
us into using Quicksort modifying it to have an increased size of the smallest to-be-sorted
sub-list (which is typically a single element in the normal version). As this provides the
guarantees we need in terms of having a partially sorted list of faces.

3.2 Features

As mentioned in the introduction, two types of features were used. Examples of these
features are shown in Figure 2.

1. Eigenfaces [11] features were extracted from training and testing images based on
eigenface projections on the training data eigenvectors’ space. The number of di-
mensions to project upon was chosen empirically based on the observation of the
decay of the eigenvalues, in our case that was 20. These type of features were ap-
pealing as they were used in many of the classification approaches for attractiveness
and generated some good results. The faces that were treated using this method
had been initially affine transformed into a common canonical face that was defined
on an image size of 170×190 pixels.

2. SIFT [9] and Partial-Descriptor-SIFT (PD-SIFT) [6] features were used as it was
shown by Geng et. al. for their effectiveness in non-holistic face recognition and
their feasibility in conditions where faces are not aligned. For these features, the
original images obtained from the dataset were kept in their original scale, and no
transformations were made on them. These were appealing, as they were never
employed in such a scenario of facial attractiveness and since the features are non-
holistic, specific attractiveness features might be captured by such descriptors. A
final note on the difference between SIFT and PD-SIFT, is that PD-SIFT matches
two descriptors in their common subspace with a correlation measure, as opposed
to using SIFT. Therefore, PD-SIFT was only suitable for use with k-NN.
3.3 Scoring and Classification

To score a newly presented novel face, two methods were implemented. The first method processed the face through a Support Vector Machine for Regression. The second method was a simple k-Nearest Neighbor implementation that gave the face a score identical to the most similar face found for it.

LibSVM [4] was used with Matlab to perform the regression analysis. More specifically, ϵ-SVR and ν-SVR were used. In the experiments discussed by the next section, the regression was performed with its default parameters set, only the different kernels were experimented with, namely Linear, Polynomial, and Gaussian RBF kernels. LibSVM was also used for SVM classification when the ordered list was converted into a categorical one.

K-Nearest Neighbor was implemented in Matlab, and was applied in both the scoring and classification problem. To be specific, K-NN does not perform regression, and handled the scoring problem as a very fine grained classification problem. In this project, only 1-Nearest Neighbor was experimented with. It is probable that having 2-Nearest Neighbor or k-Nearest Neighbor could produce better results especially in terms of scoring where the resulting score is a weighted function of the scores of the K classes found for it.

The Eigenface projections were ready to use directly for both the scoring and the classification problems. The projections were calculated using the following formula obtained from [11]:

\[
\text{Projection}_I = (\omega_1 \omega_2 ... \omega_k)
\]

\[
\omega_k = u_k^T (I - \Psi), \text{ where } u_k \text{ is the k-th eigenvector, and } \Psi \text{ is the mean face.}
\]

On the other hand, the SIFT features could not be directly used with the regressor, and the PD-SIFT approach was only employed with k-NN. The regressor requires a single vector to represent each face, while SIFT produces many features per face. To solve this problem, we employed a technique from [14] that is based on the bag-of-words unsupervised model, where all the SIFT features from the training set are initially clustered using K-means and then used to create a code-word for each face. The choice of K was made emperically based on the average number of sift features per face, and it was very important as it hugely impacted the error rate, as will be seen in the following section. After clustering all the SIFT features, a face’s code-word is charactarized by a vector of K dimensions, where each dimensions corresponds to how much a cluster contributes to that face. This is computed through having all of the SIFT features of a given face vote on all the cluster centers, and choosing the closest one to it for its vote. This method turned out to be suitable for usage with both the regression and classification as well.
4 Experiment

This section discusses the dataset obtained and the experiment performed on them, and then finally concludes with a discussion of the results.

4.1 Dataset

For the collection of images to run this experiment on, a web-crawler was implemented to scour the website www.HotOrNot.com for facial images. A set of 790 female faces was obtained, and was then further passed into a face recognition system, namely www.Face.com, to obtain coordinates of facial features such as the eyes, the nose, and the mouth, which were then used to conduct an affine transformation procedure to a canonical face model. Afterwards, the faces were manually checked for false detections and for faces that were highly occluded or under extreme lighting conditions. After removing those extreme cases, the dataset reduced to 456 faces. Although it seems the number of faces was cut down dramatically, a lot of faces still experienced a large amount of rotations that made the problem a lot harder, unlike previous work that relied on frontal faces only, or faces with a very small degree of rotations and controlled lighting, e.g. [-4, 4] degrees of frontal as in [12]. Finally, we would like to note that the implemented crawler fully respected the robots exclusion protocol at http://www.HotOrNot.com/robots.txt.

Two versions were kept of each face, one which was affine transformed, and one which was the original. The affine transformed faces were used with the Eigenfaces’ features, while the original ones were used with the SIFT based features.

4.2 Labeling

Since this experiment focuses on personalized attractiveness measures, preference data had to be collected to label the faces obtained from HotOrNot. The preference data was collected from 3 males on a set of 200 randomly selected faces out of the 460 faces. The preference data was used to create a training set and a testing set through splitting the preference data in half by separating the odd and even indices into two lists, as in the following example:

Ordered list of six faces: \([F_1 > F_2 > F_3 > F_4 > F_5 > F_6]\)

→ List 1: \([F_1 > F_3 > F_5]\), and List 2: \([F_2 > F_4 > F_6]\)

Although the collected preference data is fairly small, it was considered acceptable for testing purposes in this experiment. However, more data has to be collected for more concrete results.

The subjects provided their preference data through sorting the faces using a Web-Based GUI, shown in Figure 3, through selecting the more attractive face between two faces. The shown faces where resized according to the viewers browser window size, however, no transformations have been made on them. The web application implemented a QuickSort algorithm with a minimum sized sub-list of 10 faces, which gives for each possible sub-list a lower bound of 5 and an upper bound of 10 non-relatively ordered faces. The subjects took about 20-30 minutes to complete this task, and have reported that the process was tedious. They specifically mentioned that having a "pivoted" face that they compared different faces to was annoying. This is an indicator for changing the method of partial sorting if the project is to be applied in the real world.

4.3 Training

The ordered data was projected into real-numbered labels on the line from 0.0 to 1.0, as depicted in Figure 1, and was split in this experiment into two halves, however other divisions are possible, in case more training data is needed. The training sets were used to validate the scoring results by inputting them into the regression and classification tests.
Figure 3: Web GUI to sort faces.

### Regression Results

<table>
<thead>
<tr>
<th>Feature Type - Regression Method</th>
<th>Preference Set #1</th>
<th>Preference Set #2</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigenfaces-$\epsilon$-SVR</td>
<td>11.7</td>
<td>9.5</td>
<td>10.6</td>
</tr>
<tr>
<td>SIFT $K = 130$-$\epsilon$-SVR</td>
<td>0.083</td>
<td>0.084</td>
<td>0.083</td>
</tr>
<tr>
<td>SIFT $K = 130$-$\nu$-SVR</td>
<td>0.084</td>
<td>0.084</td>
<td>0.084</td>
</tr>
<tr>
<td>SIFT $K = 100$-$\epsilon$-SVR</td>
<td>0.083</td>
<td>0.084</td>
<td>0.083</td>
</tr>
<tr>
<td>SIFT $K = 100$-$\nu$-SVR</td>
<td>0.084</td>
<td>0.084</td>
<td>0.083</td>
</tr>
<tr>
<td>SIFT $K = 65$-$\epsilon$-SVR</td>
<td>0.082</td>
<td>0.084</td>
<td>0.083</td>
</tr>
<tr>
<td>SIFT $K = 65$-$\nu$-SVR</td>
<td>0.084</td>
<td>0.084</td>
<td>0.084</td>
</tr>
</tbody>
</table>

Table 1: This table presents the results of the different regression methods on the different feature types.

### 5 Results

In this section the results of running the experiment with only two of the three labeled data obtained in the previous step. The error is discussed in terms of the mean squared error calculated as the difference between the predicted labels and ground truth ones. Firstly results of the regression method will be discussed followed by the results of the classification case.

#### 5.1 Scoring

##### 5.1.1 Regression

Two types of regression were performed on the data, namely $\epsilon$-SVR and $\nu$-SVR. The regression was run with default parameters. The number of dimensions in the Eigenfaces projections were set to 20, and three different values for $K$ in the bag-of-words model for SIFT are presented. The mean squared error results are shown in Table 1. For brevity only the best kernel results are provided.

The Mean-Squared-Error (MSE), calculated as the difference between the actual and the predicted squared for all the testing faces, gives us insight on how good is our predictions. We immediately notice the abnormal MSE values for the Eigenfaces. The regression results for this case used linear kernels and the predicted values had very large magnitudes reaching up to 7.0, where they normally should not be greater than 1.0 by such a large amount, the case was similar for values less than 0.0. Until now, we have no insight on what caused such behavior. Evaluating the results for Eigenfaces with different kernels such as polynomial or Gaussian RBFs yielded a constant value of 0.5, which is also a strange behavior. However,
Table 2: This table presents the results of the 1-Nearest Neighbor method on the different feature types. Least is better.

<table>
<thead>
<tr>
<th>Feature Type</th>
<th>Preference Set #1</th>
<th>Preference Set #2</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigenfaces</td>
<td>0.17</td>
<td>0.16</td>
<td></td>
</tr>
<tr>
<td>SIFT $K = 130$</td>
<td>0.18</td>
<td>0.17</td>
<td>0.17</td>
</tr>
<tr>
<td>SIFT $K = 100$</td>
<td>0.12</td>
<td>0.16</td>
<td>0.14</td>
</tr>
<tr>
<td>SIFT $K = 65$</td>
<td>0.14</td>
<td>0.17</td>
<td>0.15</td>
</tr>
<tr>
<td>PD-SIFT</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
</tr>
</tbody>
</table>

the results for SIFT and the bag-of-words model used, yielded good results and reported a MSE of 0.083 on average. Calculating the Root-Mean-Squared-Error (RMSE) yields 0.28, which if interpreted with respect to the actual and predicted labels says that the predicted results deviate from the actual score by 0.28 on average. This result is not very impressive, however, considering that the faces used were not under ideal conditions and exhibited a large amount of rotations and poor lighting, one can hope to improve the results by finding the proper parameters for the regression, and for the parameter $K$ of the bag-of-words model. Finally, We would like to note that the reported results represent the best result between three kernels (linear, polynomial, and Gaussian RBF).

5.1.2 Nearest Neighbor

A 1-Nearest Neighbor implementation was used to produce scoring results for test data. In this method, a test face is given the same score as the closest face to it in the training set. Table 2 shows the results of running this method.

Comparing the results of k-NN to SVR shows that 1-NN is most probably unsuitable for scoring purposes, especially in the form of 1-NN. PD-SIFT yielded very poor results, we attribute that to two factors. First, the formula for choosing the closest face was not properly calibrated to this problem. Second, 1-NN’s poor performance in scoring purposes.

The nearest neighbor approach could be used differently in this domain, perhaps one can use the Nearest Neighbor approach to calculate a score of a given face based on how much every face in the training set contributes to it.

5.2 Classification

As mentioned earlier, we converted the problem into a classification one by defining boundaries that divide the ordered labeled data into separate groups. For purposes of this project, only two groups were made and labeled as "Attractive" and "Unattractive". Both the SVM methods and Nearest Neighbor methods were employed. The Nearest Neighbor method also incorporated the PD-SIFT descriptors and the matching correlation function as suggested by [6]. Table 3 shows the percentage of correct classifications for each feature type and the corresponding method used.

The classification results for the boundry-created classification problem shows that the attractiveness assessments are not random and some attractiveness measurements were learned, as correlation values reached 63% for SVM with $K=100$. This points out that further refinement of the parameters and features will eventually yield better results.

6 Future Work

Work on this project is far from complete. The ability to improve the results is very feasible, as seen with the variance of $K$ in the SIFT features, the results were impacted. The parameters for the SVR cases were not fine-tuned due to time constraints. Variants of k-NN, i.e. with different $k$ values, were not tested which might be more relevant in the
Table 3: This table presents the results for the classification instance, with all the different feature types and methods used. Higher is better.

scoring scenario. A larger collection of preference data should rule out erroneous scoring and classification due to high inconsistencies in a given subject’s taste.

The faces used were real-world data that was not constrained in any manner, and the faces experienced a huge amount of variance in terms of illumination and rotation. More concrete testing shall be executed on another dataset of faces that is does not suffer from such problems, such as the GENKI dataset used by Whitehill [12]. Furthermore, other machine learning techniques exist that have not been employed, and moreover, other types of features are possible such as Fisherfaces [2], Laplacianfaces [7], Randomfaces [13], or maybe a combination that capitalizes on capturing beauty features that interest a given subject.

7 Conclusion

The results obtained so far are promising, currently the best mean squared error is 0.082, however, we are confident that with further refinement of the regression parameters and features, better results will sprout. Having proper features that could capture attractiveness is key to this problem. In quantifying beauty, we utilized features designed for facial recognition as if we were treating the problem as some sort of weak facial recognition problem, and perhaps, if treated differently the results would be better. This question remains unanswered. The accuracy of the results is comparable to those of prior work, especially when considering that we chose to work with real world data instead of ideal conditions. Nonetheless, improvements are required if the final goal of this project is set to see light in the real world.

References


