Computer Vision
Project Report
Team: mob-psycho

Project Title: Relative Attributes

Team Members
Kanav Gupta (20161151)
Parth Partani (20161034)
Anurag Mehta (20161016)

Introduction/Motivation

Given training data stating how object/scene categories relate according to different attributes, we learn a ranking function per attribute using Rank-SVM. Now this ranking function is used can be used to compare strengths of any particular attribute (smile, age, color etc.) between two images.

This relative classification is much better and is of more use that the binary classification as binary classification only tells us about whether a person in image is young or old and not the relative order. Also sometimes it is very hard to make binary decision on an attribute. Example can be seen in image below:

Unlike binary classification, Relative attributes are great way to classify and compare objects in the world. They indicate the strength of an attribute in an image with respect to other images, thus allowing comparisons between images and classes.

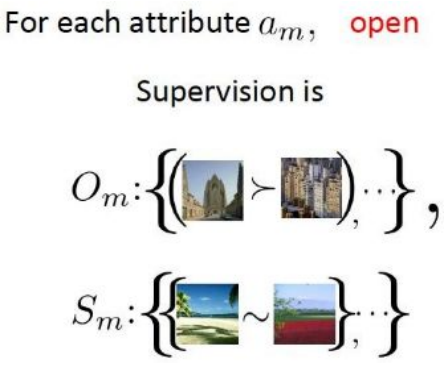
Approach Used

We have divided this project into three components mentioned below:

- Learning Relative Attributes
- Zero-shot Learning
- Automatic generation of image description

Learning Relative Attributes

Ranking function for each attribute is learned using Rank-SVM. Complete workflow is defined below:



As mentioned in image above, we first divide our data into two sets, one with the pair of images having similar attribute strength and other one in which image pair have fairly different attribute strengths. Now we need to learn a scoring (or ranking) function such that it follows specified constraints for these sets i.e score obtained from similar attribute classes should be equal while that of different classes should follow same relative ordering.

Learn a scoring function $r_m(\boldsymbol{x_i}) = \boldsymbol{w_m^T x_i}$ (Image features; Learned parameters)

that best satisfies constraints:

$$\forall (i,j) \in O_m : \boldsymbol{w_m^T x_i} > \boldsymbol{w_m^T x_j}$$
$$\forall (i,j) \in S_m : \boldsymbol{w_m^T x_i} = \boldsymbol{w_m^T x_j}$$

Now as this problem is NP-hard, we introduce slack variable to make approximate solution, similar to that in SVM Classification.

After adding slack variables it becomes optimization described below and can be solved using Rank-SVM.

OPTIMIZATION PROBLEM 1. (RANKING SVM)

minimize: $$V(\vec{w}, \vec{\xi}) = \frac{1}{2}\vec{w} \cdot \vec{w} + C \sum \xi_{i,j,k} \quad (12)$$

subject to:

$$\forall (d_i, d_j) \in r_1^* : \vec{w}\Phi(q_1, d_i) \geq \vec{w}\Phi(q_1, d_j) + 1 - \xi_{i,j,1}$$
$$\ldots \quad (13)$$
$$\forall (d_i, d_j) \in r_n^* : \vec{w}\Phi(q_n, d_i) \geq \vec{w}\Phi(q_n, d_j) + 1 - \xi_{i,j,n}$$
$$\forall i \forall j \forall k : \xi_{i,j,k} \geq 0 \quad (14)$$

C is a parameter that allows trading-off margin size against training error. Geometrically, the margin δ is the distance between the closest two projections within all target rankings. This is illustrated in Figure 2.

Optimization Problem 1 is convex and has no local optima. By rearranging the constraints (13) as

$$\vec{w}\left(\Phi(\mathrm{q}_k, \mathrm{d}_i) - \Phi(\mathrm{q}_k, \mathrm{d}_j)\right) \geq 1 - \xi_{i,j,k}, \quad (15)$$

Rank-SVM objective function formulation in terms of our problem is described below where epsilon and gamma are slack variables (or error margin in generic terms) and C is hyperparameter known as trade-off constant between maximizing margin and strongly satisfying pairwise constraints. Implementation of Rank SVM is done using Newton's Method (described in implementation details section ater)

Max-margin learning to rank formulation

$$\min \quad \left(\frac{1}{2}||\boldsymbol{w_m^T}||_2^2 + C\left(\sum \xi_{ij}^2 + \sum \gamma_{ij}^2\right)\right)$$

$$\text{s.t} \quad \boldsymbol{w_m^T}(\boldsymbol{x_i} - \boldsymbol{x_j}) \geq 1 - \xi_{ij}, \forall (i,j) \in O_m$$

$$|\boldsymbol{w_m^T}(\boldsymbol{x_i} - \boldsymbol{x_j})| \leq \gamma_{ij}, \forall (i,j) \in S_m$$

$$\xi_{ij} \geq 0; \gamma_{ij} \geq 0$$

Based on [Joachims 2002]

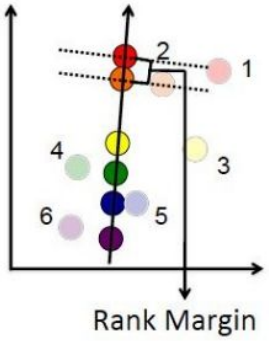
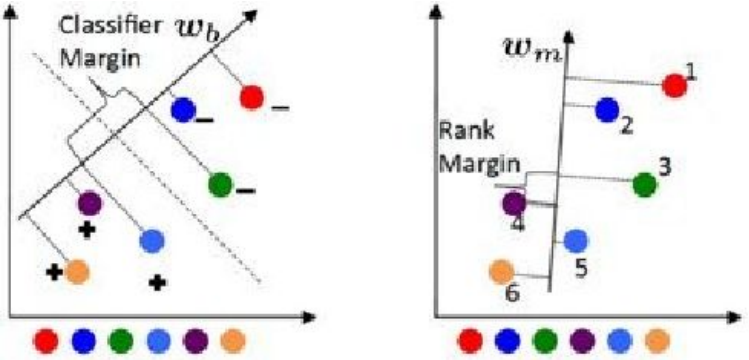


Image → Relative Attribute Score

Difference between Rank-SVM and SVM

Rank-SVM learns a function that explicitly enforces a desired ordering on the training images and the margin is the distance between the closest two projections within all desired (training) rankings whereas, in SVM margin between nearest binary labeled examples (Support Vectors) is enforced. The below diagram explains the same:



After training Rank-SVM model, we obtain a weights vector (w) for each attribute, now this weight vector can be used to calculate relative strength of any attribute for any given test images.

Now obtained weight matrix for attributes have two further applications described in the paper

- Zero Shot Learning
- Generating image descriptions

Zero Shot Learning

“Zero-shot learning consists in learning how to recognise new concepts by just having a description of them.”

As the above definition suggests, zero-shot learning in this case is used to formulate new classes by using its relationship (of attributes) with current classes and no training images. Now if new test image is provided, class of this image can be predicted.

For zero-shot learning we come across two types of classes:

1. Seen Classes: For classes in this category, training images is available and also complete pairwise relationship between classes for all attributes is also available.
2. Unseen Classes: For classes in this category, No training images are available. Only some of its attributes are described in terms of seen classes. Eg smile(a) < smile(b) < smile(c) or smile(a) < smile(b) while a,c are seen classes while b is unseen one.

Now for training our model, we first calculate all attribute scores for all the images and represent each image as a m-vector, where m is number of classes. Now we use the m-vectors of each seen class to form a gaussian mixture model (or any other generative model) by calculating mean vector and covariance matrix for the class.

After obtaining GMM for each seen class, next step is to infer the generative model for unseen classes, for each attribute different procedure is obtained for different relationship types which is well explained in paper (screenshot of same attached below)

1. If $c_j^{(u)}$ is described as $c_i^{(s)} \succ c_j^{(u)} \succ c_k^{(s)}$, where $c_i^{(s)}$ and $c_k^{(s)}$ are seen categories, then we set the m-th component of the mean $\boldsymbol{\mu}_{jm}^{(u)}$ to $\frac{1}{2}(\boldsymbol{\mu}_{im}^{(s)} + \boldsymbol{\mu}_{km}^{(s)})$.
2. If $c_j^{(u)}$ is described as $c_i^{(s)} \succ c_j^{(u)}$, we set $\boldsymbol{\mu}_{jm}^{(u)}$ to $\boldsymbol{\mu}_{im}^{(s)} - d_m$, where d_m is the average distance between the sorted mean ranking-scores $\boldsymbol{\mu}_{im}^{(s)}$'s of seen classes for attribute a_m. It is reasonable to expect the unseen class to be as far from the specified seen class as other seen classes tend to be from each other.
3. Similarly, if $c_j^{(u)}$ is described as $c_j^{(u)} \succ c_k^{(s)}$, we set $\boldsymbol{\mu}_{jm}^{(u)}$ to $\boldsymbol{\mu}_{im}^{(s)} + d_m$.
4. If a_m is not used to describe $c_j^{(u)}$, we set $\boldsymbol{\mu}_{jm}^{(u)}$ to be the mean across all training image ranks for a_m and the m-th diagonal entry of $\Sigma_j^{(u)}$ to be the variance of the same.

In the first three cases, we simply set $\Sigma_j^{(u)} = \frac{1}{S}\sum_{i=1}^{S}\Sigma_i^{(s)}$.

Here c's are the relative relationship of attribute between unseen class and seen classes. Am is attribute m and mean and covariance have general notations.

Following above process, we obtain GMM for seen and unseen classes, now if new test image is given, first we calculate its m-vector by calculating score for each attribute and obtain probabilities of belonging to each of classes from its model and classify it into class which has maximum probability (likelihood).

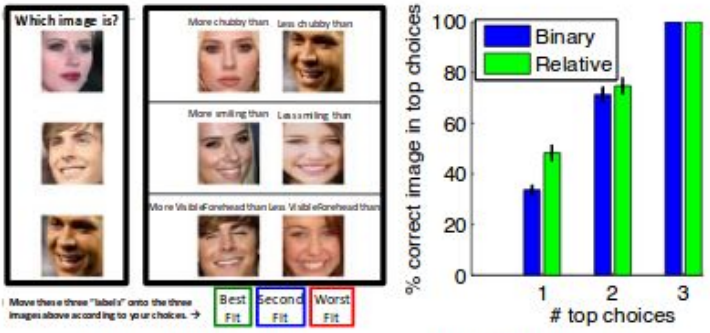
$$c^* = \underset{j \in \{1,\ldots,N\}}{\operatorname{argmax}} \; P\left(\tilde{\boldsymbol{x}}_i \mid \boldsymbol{\mu}_j, \Sigma_j\right).$$

Describing Images in Relative Terms

The goal of this application is to describe new images in terms of existing dataset images for different attributes. In the first part of project we have obtained weights for each attribute that gives us score when multiplied by feature vector and can be used to obtain relative ordering between images for any particular attribute.

Now for any new image (I) to be described we calculate its score for each attribute am. Now we have to find two images from out dataset (for each attribute) s.t attr(j) < attr(i) < attr(k). Although any image with higher score(for k) and lower score (for j) will suffice, we obtain j,k such that they are not very similar to i in terms of attribute strength, and not too far away from i to avoid trivial descriptions. So we pick j,k such that there are ⅛ th of training images between j and i and similarly between i and k, for corner cases where ⅛ th images are not available we choose images with highest strength(for k) and lowest strength (for j)

This description is much more informative and can be used to distinguish each image in better way than binary description. Experiment for same was performed by paper authors and results are attached below:



(a) Human Study Interface (b) Results (both datasets)

Implementation Details

Dataset Used

We have used PubFig dataset with 772 images and taken 11 attributes like smiling, young, male etc for relative comparisons.

Feature Extraction

512 sized GIST features appended to 30 histogram bins are used for training and testing purposes.
Although we have used pre-extracted features for images (provided by author along with dataset), we have function which can extract these features.

1. Rank-SVM

For training Rank-SVM, after classifying pairs in set O or S we initialize a weight and training error penalization matrices. Weight difference for each pair of images with current weight is also initialized which will be updated as weight matrix gets updated in each iteration of training SVM.

Now in each iteration we find out objective (cost) , gradient and support vectors using current weight, penilization etc. using function below

```
def object_fun_linear(w, C, out, n0, A, X):
    out[0:n0] = np.maximum(np.zeros([n0, 1]), out[0:n0])
    obj = np.sum(np.multiply(C, np.multiply(out, out))) / 2.0 + np.dot(np.transpose(w), w) / 2.0
    grad = w - np.transpose(np.transpose(np.multiply(C, out)) * A * X)
    sv = out[0:n0] > 0, abs(out[n0:]) > 0
    return obj[0, 0], grad, sv
```

Now for optimization in SVM we have used Newton Method which make use of hessian matrices to minimize error (or cost). This provides us with direction in which we should proceed.

Now we will find a line in this direction using our current 'w' matrix and then we will update 'w' using that solution.

```
def line_search_linear(w, d, out, C, n0 ,A, X):
    t = 0
    Xd = A * (X * d)

    while 1:
        out2 = out - t * Xd
        sv = np.nonzero( scipy.vstack(( out2[0:n0] > 0, abs(out2[n0:]) > 0 )) )[0]
        g = np.transpose(w) * d + t * np.transpose(d) * d - np.transpose(np.multiply(C[sv], out2[sv])) * Xd[sv]
        h = np.transpose(d) * d + np.transpose(Xd[sv]) * np.multiply(Xd[sv], C[sv])
        g, h = g[0, 0], h[0, 0]
        t -= g / h
        if g * g / h < 1e-8: break
    out = out2
    return t, out
```

After training we obtain weights for each attribute.

2. Zero Shot Learning

For zero shot learning we first obtain weights from Rank-SVM (using seen classes only) and then calculate 11 sized vectors for each image which stores score of each attribute for each image. After calculating mean and variance using numpy, we fit each class in an GMM (scikit-learns module). Now similar procedure is followed as explained in details.

3. Describing Images

All scores for all images in dataset are calculated and stored, now for every new image, we sort the images according to score of that variable in ascending order. Now score close to new images score is searched in the allpreds (refer code) array and i,k indices are found and displayed accordingly.

Note: Refer Jupyter Notebook Comments/Markdown for more details

Results Obtained

After the first part of learning relative attributes we have found out weight matrix corresponding to each of eleven attributes in our PubFig dataset. This weight matrix can be used to compare attribute strength between any two images and also find out compete ordering between multiple provided images.

Image Set 1
Attribute = Smiling

```
Attrbute values for image 1 and 2 :
```

Value =[[0.12842465]]

Value =[[0.08821574]]

Image Set 2

Attribute = Big Lips

Value =[[-0.18477103]]

Value =[[-1.07309342]]

Image Set 3

Attribute = Narrow Eyes

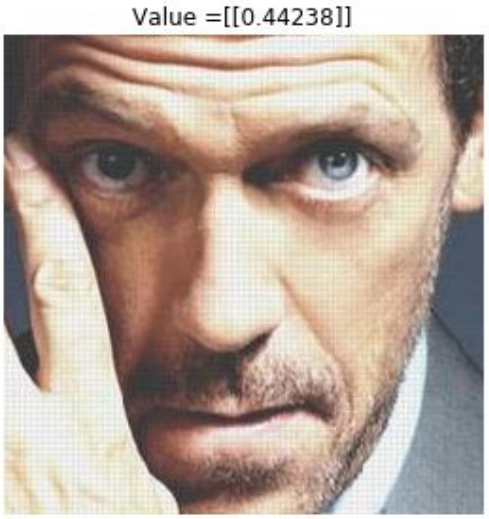
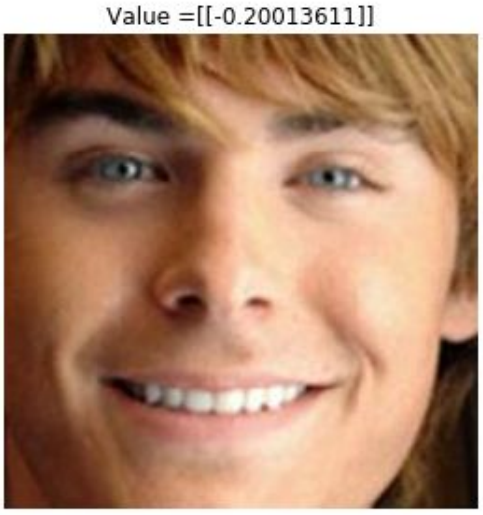


Image Set 4

Attribute = Chubby

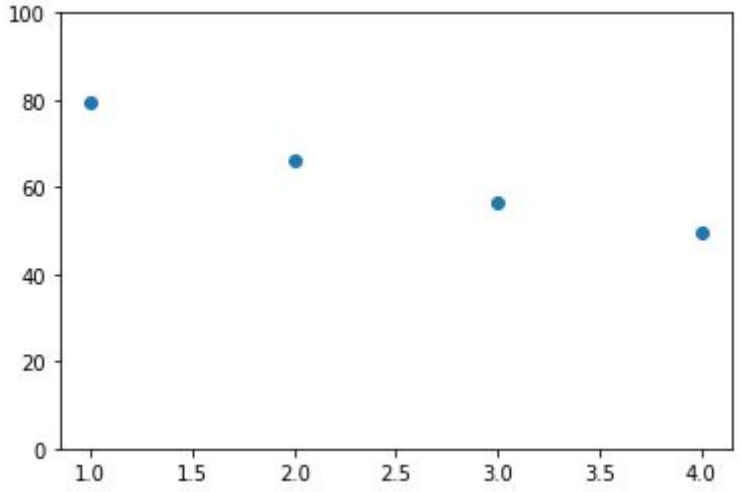


For Zero Shot Learning we obtained following accuracies for different number of unseen categories

Number of Unseen Classes	**Accuracy**
1	79.583 %
2	66.204 %

3	56.592 %
4	49.481 %

Plot for the same is shown below:



Results for describing images are as shown:

Image 1

Attribute = Smiling

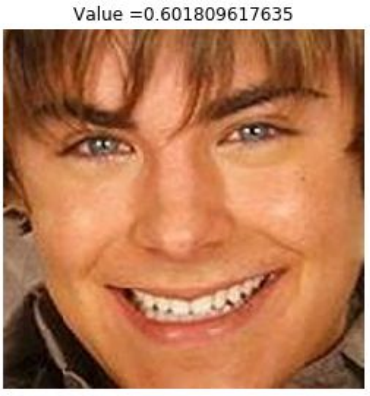


Image 2

Attribute = Male


```
Image is more Male than CliveOwen_119.jpg and less Male than ViggoMortensen_146.jpg
```

Image 3

Attribute = Big Lips


```
Image is more BigLips than ScarlettJohansson_100.jpg and less BigLips than CliveOwen_89.jpg
```

Image 4

Attribute = Smiling


```
Image is more Smiling than MileyCyrus_66.jpg and less Smiling than ScarlettJohansson_176.jpg
```

References

1. https://www.cc.gatech.edu/~parikh/relative.html
2. http://people.csail.mit.edu/torralba/code/spatialenvelope/
3. https://icml.cc/2015/wp-content/uploads/2015/06/icml_ranking.pdf

---