Computer Vision
Project Report
Team: mob-psycho

Project Title: Relative Attributes

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

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

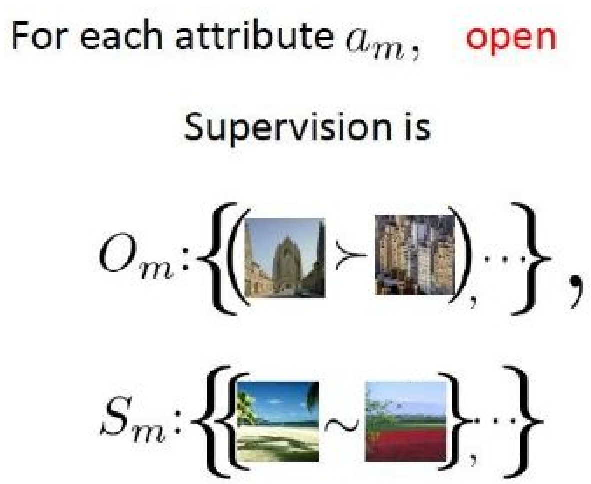
We have divided this project into three components mentioned below:

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

For the mid evaluation we have implemented 1st part of the project.

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

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

that best satisfies constraints:

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

This optimization problem is solved using Rank-SVM as described in image below:

Optimization Problem 1. (Ranking SVM)

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

subject to:

$$\begin{aligned} &\forall (d_i, d_j) \in r_1^* : \vec{w}\Phi(q_1, d_i) \geq \vec{w}\Phi(q_1, d_j) + 1 - \xi_{i,j,1} \\ &\qquad \ldots \\ &\forall (d_i, d_j) \in r_n^* : \vec{w}\Phi(q_n, d_i) \geq \vec{w}\Phi(q_n, d_j) + 1 - \xi_{i,j,n} \end{aligned} \tag{13}$$

$$\forall i \forall j \forall k : \xi_{i,j,k} \geq 0 \tag{14}$$

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

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

Rank-SVM objective function formulation in terms of our problem will be

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]

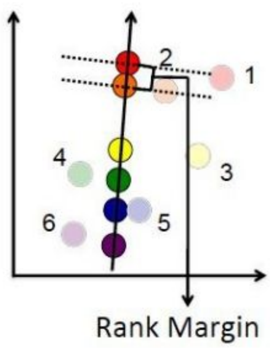


Image → Relative Attribute Score

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.

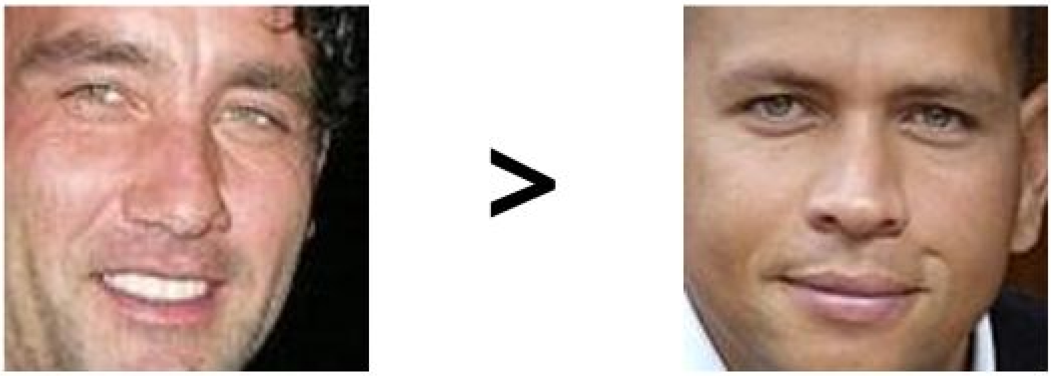
Now we will find a line in this direction using our current 'w' matrix and then we will update 'w' using that solution.

```
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
```

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. For an example we took smiling attribute and found out scores for a pair images which is shown below:



Value = 0.2403 Value = 0.0882

So here we can clearly see that first person is smiling more than the second one. Similar comparisons can also be done for any set of images and for any of eleven attributes.