Object Detection within Cluttered Images with PCA versus Sparse Matrix

Patrick Talley

under the direction of
Prof. Gesualdo Scutari
Purdue University

Abstract

Finding objects within an image is a challenging task. There is often clutter, and understanding the difference between objects can be challenging. In this paper a comparison between Principal Componenet Analysis, PCA, with eigenvectors, and Low Rank + Sparse Matrix representation were compared. This was done on fishing photos from The Nature Conservancy with the goal of singling out the fish within the image. What was found was that the Sparse Matrix was able to capture the fish's features much more consistently than PCA.

Contents

1 Introduction

The topic is trying to find an object given a dictionary using PCA and sliding window. The image is very noisy, and the object has many different forms. This topic specifically attempts to address a challenge proposed by The Nature Consevancy, where they wish to identify species caught by fishing boats to reallocate human capital for enforcement. In order to tackle to challenge, and find the fish, a basis of fish was found by using 100 images cropped to only contain the fish. Once this basis was found, a sliding window going through a 20x20 block of the tested image was then tested against this basis. If the projection was within a threshold then the block will be treated as containing the fish. This creates a sparse representation of the image useful for further analysis. The reason that this approach was used was specifically to attempt to detect the difference between a fish and the background.
A second option to find this was by the creation of a sparse matrix, and a low rank matrix. While there exists a large amount of clutter, PCA often fails due to corruption, and this approach is more robust. A sparse matrix containing the figures of the fish, and a low rank matrix containing the boat and ocean will be created. A comparison of these approaches for the type of problem as far as containment of the fish, and number of non-zeros of the resulting matrices.

2 Problem Formulation

2.1 PCA with Sliding Window

The main issue that arises when attempting to use principal componenet analysis is that the picture is very noisy and as such it will have difficulties differentiating between the correct object and others. PCA works by creation of a basis using k eigen vectors.
Consider a set of n fish images $X = \{x_1, x_2, ..., x_n\}$, where each x_i represents a vector of the fish image.
Let $\underline{\bar{x}}$ by the average fish among X then normalize the fish such that $\tilde{X} = X - \underline{\bar{x}}$
Then project the faces. Because X is such a large matrix in order to create the eigenvectors singular value decomposition will be used.

$$\tilde{X}^T\tilde{X} = V\Sigma^2V^T$$

Let V_k be the fist k vectors in V.
Then from $V^TV = I$

$$\tilde{X}\underline{v_1} = U\Sigma V^Tv_1 = U\Sigma\begin{bmatrix}1\\0\\...\\0\end{bmatrix} = U\begin{bmatrix}\sigma\\0\\...\\0\end{bmatrix} = \sigma_1u_1$$

$$u_1 = \frac{\tilde{u_1}}{\|\tilde{u_1}\|}$$

Now we have a matrix $A =$ first k elements of U normalized. Create a projection space $P_{x_1} = A^T * \tilde{x_1}$ for all $\tilde{x_i} \in \tilde{X}$
When given an image to test against Y it will be broken up into 20x20 pixel squares, S. This sliding square will cycle through the image and project into the space.

$$P_y = A^T * \underline{s}$$

Then it will be tested agains P_x to determine if this square is a fish or not.

$$\bar{F} = \frac{1}{n} \sum_{i=1}^{n} \|P_y - P_{x_i}\|$$

If $\bar{F} \leq \phi$ where ϕ is the threshold, then it will be included in the image.

2.2 Low Rank Approximation

Given an image Y there arises a need to create a matrix A and a matrix E, where A is the lowest rank that agrees with the data, and E is some sparse matrix. A natural extension of this is to create the following formulation

$$min\ rank(A) + \lambda \|E\|_0\ subj A + E = Y$$

However, the l_0 and the rank is non-convex. So to approximate this minimization problem the nuclear norm, and l_1 norm are used.

$$min\ \|A\|_* + \lambda \|E\|_0\ subj A + E = Y$$

The l_1 norm is the convex envelope of the l_0 norm, that is it is the best approximation while remaining convex. This program is semidefinite and can be solved in polynomial time. In order to determine λ a non-adaptive weight factor is used. For this instance it is $\frac{1}{\sqrt{max\{n,m\}}}$ where n, m are the dimensions of Y.

$$\therefore (A_0, E_0) = argmin\ \|A\|_* + \lambda \|E\|_1\ subj A + E = Y$$

And the matrix we want is the sparse matrix E which should contain the fish within the image.

3 Main Contribution

To perform PCA for this image, it was done through the creation of an eigenspace which to project a vector into. In order to try and match and object, a sliding window technique was used. The reason that this attempting to work around the noise and similarity to other objects within with image to the fish.
This was tested against a sparse representation in order to determine which of the two could correctly identify this image among the noise. Given the need to use these detection algorithms within a robust range of cases having alternatives to the requirements of eigenfaces. The sparse matrix does not require well constructed test cases to be used. This makes it attractive to be put in place more applications.

4 Numerical Results

5 PCA

A basis using cropped fish were used from a variety of images. The dataset was provided by The Nature Conservancy. To simplify the test cases they were resized from the original resolution of 1280x720 to 240x240. The image was also altered to greyscale for simplicity.

Figure 1: Example of image change. Left original, right greyed and squared version.

The fish images were resized to uniformy 20x20 resolution. These images were not as clean as would be required by PCA, but given the limitations of the dataset they were the best that could be used. Again these images were all set to greyscale for the purposes of this.

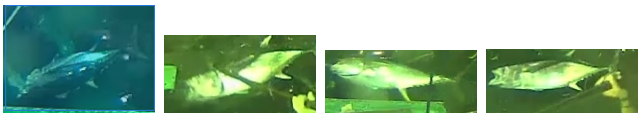


Figure 2: Example of original version of fish used.

In order to create an eigenspace the following code was used.

```
k = 45
trial = 2

# load Data
path = "./HeadsAndTails/"
files = readdir(path)
n = 100
finalDim = (20,20)
fileInd = rand(1:size(files,1),n)
curfile = joinpath(path,files[1])
temp = convert(Array{Float64,2},convert(Image{Gray},load(curfile)))
temp = Images.imresize(temp,finalDim)
```

```
dim = size(temp)
D = zeros(prod(dim), n)
D[:,1] = vec(temp)
for i=2:n
    curfile = joinpath(path,files[fileInd[i]])
    temp = convert(Array{Float64,2},convert(Image{Gray},load(curfile)))
    temp = Images.imresize(temp,finalDim)
    D[:,i] = vec(temp)
end

# Normalize
x_bar = zeros(size(D,1),1)
for i =1:size(D,2)
    x_bar = x_bar + D[:,i]
end
x_bar = x_bar/size(D,2)

# Get residuals
X_tilde = D - repmat(x_bar ,1,size(D,2))

covm = X_tilde'*X_tilde
U, S, V = svd(covm)
Vk = zeros(size(V,1),k)
Vk = V[:,1:k]
Uk = X_tilde*Vk
norms = sqrt(diag(Uk'*Uk))

A = Uk./ repmat(norms',size(Uk,1),1)

# Project
Px = zeros(k,size(D,2))
for i=1:n
    Px[:,i] = A'*X_tilde[:,i]
end
;
```

The resulting eigenfish from this can be seen in figure 3.

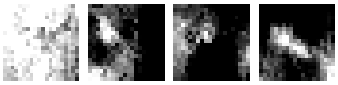


Figure 3: Top four eigenfish.

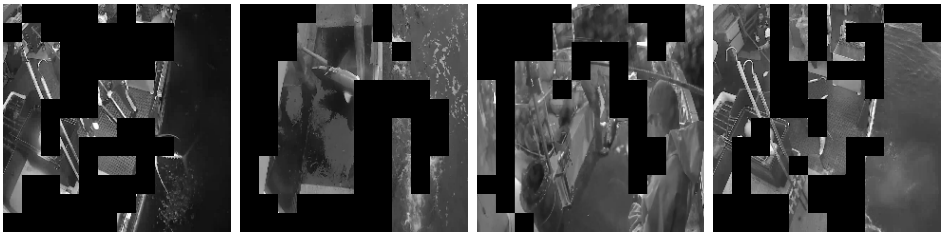
In order to do the sliding window the following code was used. This code takes the resolution, then will travel by a 20x20 block to perform the projection. The threshold was set to 10 for this from experiments.

```
function projtest(A,X,Px, threshold)
    p = A'*vec(X)
```

```
    n = size(Px,2)
    fish = false
    snorm = 0
    for i=1:n
        snorm+= norm(p-Px[:,i])
    end
    anorm = snorm/n
    if anorm <threshold
        return true
    else
        return false
    end
end

L = 240
w = 20
testOut = zeros(L,L)
for i in 1:w:L-w+1
    for j in 1:w:L-w+1
        sample = testim[i:i+w-1,j:j+w-1]
        fish = projtest(A,sample,Px,10)
        if fish
            testOut[i:i+w-1,j:j+w-1] = 1
        end
    end
end

grayim(testOut)
r = round(Int,prod(size(testOut)))
for i = 1:r
    if testOut[i] == 1
        testOut[i]=testim[i]
    end
end
grayim(testOut)
save("pca1.png",testOut)
```

The results of the PCA were disappointing. It seemed to prefer everything but the fish when deciding if the part belonged to the final output or not. While the resulting image was sparse there was no way to identify the fish from this anymore.

(a)



(b)

Figure 4: Original on top, PCA output on bottom

5.1 Sparse Approximation

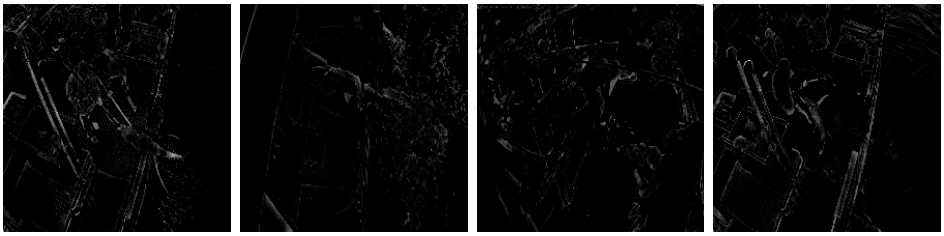
```
testim =
    convert(Array{Float64,2},Images.imresize(convert(Image{Gray},load("./Test/test7.jpg")),
    (220,220)))
Y = testim
X = Variable(size(testim))
E = Variable(size(testim))
lam = 1/sqrt(maximum(size(Y)))

problem = minimize(lam*norm(vec(X),1)+nuclearnorm(E),X+E==Y)
solve!(problem)
```

As stated previously this code solves for

$$(A_0, E_0) = argmin\ \|A\|_* + \lambda\|E\|_1\ subj A + E = Y$$

The output is shown below in figure 5.

(a)



(b)

Figure 5: Original on top, Sparse Approximation output on bottom

A few obvious things are that the image is much more sparse than the resulting one from PCA. Also, the fish has also been outlined which is much improved from PCA. A close up is shown below.

Figure 6: Larger version of sparse output

The features of the fish are fully present in this image, and it was able to reduce the clutter of the image much more than PCA.

6 Conclusions

A comparison on the effectiveness of PCA using sliding window technique, versus Sparse Matrix representation was used. This was done in order to find objects within a cluttered image. What was found was the PCA performed poorly in its ability to find the object that was given to it relative to the Sparse Matrix. However, PCA was much quicker, but being able to quickly get to the incorrect answer does not make up for the limitations it comes to. In general, PCA requires much more time spent before the analysis to clean up the image, while a sparse matrix can be formed without any priot knowledge of the structure.