CS 322 Homework 6

out: Tuesday 27 March 2007
due: Wed 11 April 2007

Principal components analysis: Eigenfaces

The data file `hw6faces.mat` contains a $320 \times 200 \times 40$ matrix which comprises 40 images: 10 faces under 4 lighting conditions each. (The images are from the Yale Face Database B.) These images have some similarities, since they are all faces, and we can use PCA to try to capture these similarities using a set of basis images. (This approach was pioneered by Turk and Pentland in 1991, who dubbed the resulting principal components "eigenfaces.")

1 Compute the principal components of the set of images (i.e. the eigenfaces) using the SVD. In this context you should treat the pixels in each image as a single vector: the PCA process is just taking linear combinations of pixel, without regard for where the pixels are in the images. Describe the first three eigenfaces in terms of what they capture about the variations among the images.

Answer:

```
load hw6faces.mat
A = reshape(faces, [320*200 40]);
[U,S,V] = svd(A - mean(A')' * ones(1,40), 0);
efaces = reshape(U, size(faces));
```

The eigenfaces are the columns of `U` reshaped back into images. The first face is a generic-looking face that contains strong left-right lighting, while the second and third faces both look much like the face of the single female in the sample set, with the third face emphasizing her bangs and neck slightly more. See Figure (1). Here we are subtracting the mean before performing PCA – if we do not subtract the mean, then our first eigenface is the mean face, and the second and third eigenfaces are the first and second eigenfaces above, respectively.

2 If I want to approximate the j^{th} image by a linear combination of the first k Eigenfaces, what are the weights to use?

Figure 1: First three eigenfaces

Answer:`U(:,1:k) * S(1:k,1:k) * V(j,1:k)'` is the approximation of the j-th image with the first k eigenfaces. From this, we can conclude that the weights to use come from the j-th column of V^T with each component scaled by the first k singular values

3 Try approximating the faces in the dataset with different numbers of eigenfaces. How many do you need before the faces become recognizable?

Answer: Somewhere between 5 and 8 faces are usually enough, depending on how stringent your requirements for "recognizable" are.

⋆4 Take a picture of yourself or a friend with a neutral expression and soft lighting, so that it looks similar to the faces in the dataset. Using `cpselect`, `cp2tform`, and `imtransform`, align it (by a "linear conformal" mapping, as `cp2tform` calls it) to the images in the dataset. Show the approximations for different numbers of eigenfaces How well can you approximate this new image?

Answer: The face needs to be aligned exactly and have a neutral expression; otherwise features like eyes and mouths tend to be extremely difficult to capture. Even when well-aligned, it is quite difficult to get a good approximation to the image. Including the new image in the sample data and re-running the PCA of course returns much better results, but the idea was to see how well the sample data could capture the new data.

5 How many eigenfaces do you need to approximate the original data matrix to within 1%? (That is, the F-norm error is less than 1% of the F-norm of the data matrix.) How can you tell this from looking at the singular values?

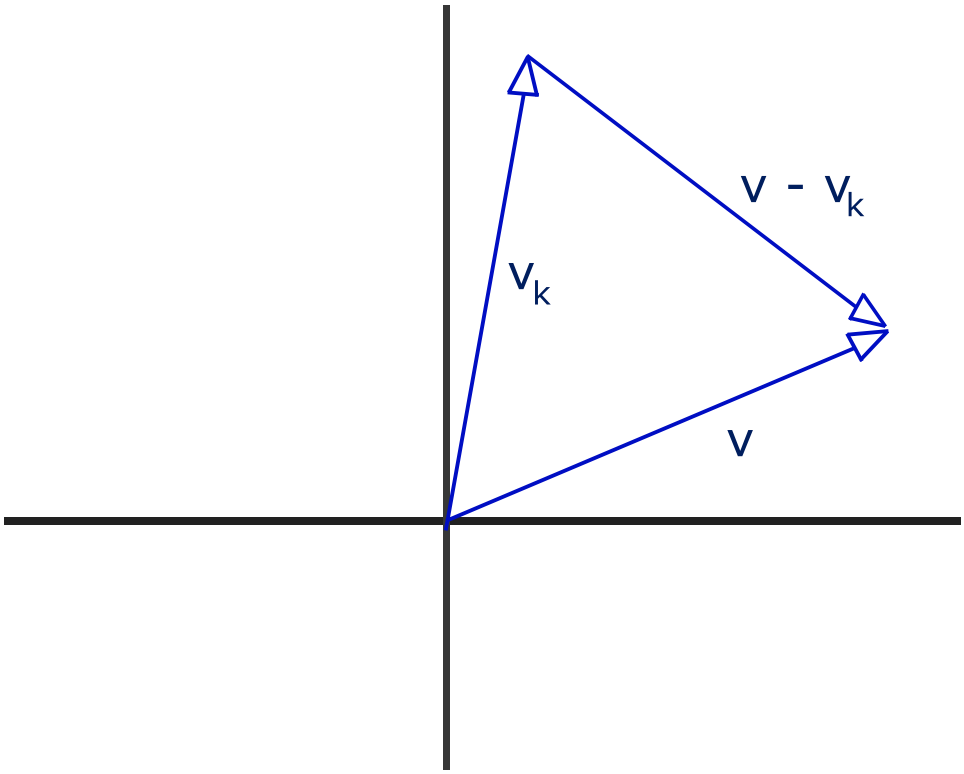

Figure 2: A vector $\mathbf{v}$, an approximation $\mathbf{v}_k$, and the error $\mathbf{v} - \mathbf{v}_k$

Answer: The F-norm of a matrix $\mathbf{A} = \mathbf{USV}^T$ is the F-norm of the matrix S; that is, it is the square root of the sum of squares of the singular values (because $\mathbf{U}$ and $\mathbf{V}$ are orthonormal, they do not change the F-norm). We wish to bound the F-norm of the error matrix $\mathbf{A} - \mathbf{A}_k$, where $\mathbf{A}_k = \mathbf{U}(:, 1:k)\mathbf{S}(1:k, 1:k)\mathbf{V}(:, 1:k)'$. We note that the F-norm of $\mathbf{A} - \mathbf{A}_k$ is simply the F-norm of $\mathbf{S}(k+1:n, k+1:n)$, i.e. the singular values not included in the approximation A_k. Thus, we wish to find k such that $\|\mathbf{S}(k+1:n, k+1:n)\| < 0.01\|\mathbf{S}\|$. When we subtract the mean, $k = 37$. When we do not subtract the mean, $k = 31$.

Note that this is different than computing k such that $\|A_k\| \geq 0.99\|A\|$. To see this, consider the case of 2D vectors in Figure (2). Here we have some vector $\mathbf{v}$, and some approximation $\mathbf{v}_k$. $\|\mathbf{v}_k| \geq 0.99\|\mathbf{v}\|$ (in fact, the two vectors have equal norm), but this does not say anything about the error $\mathbf{v} - \mathbf{v}_k$, which is still quite significant. As a result, in our computation of error, we need to make sure we are bounding the norm of the *error* and not the norm of the *approximation* relative to the solution.