Recognition using linear models

Course Notes by Mathieu Lamarre

Presented to Prof. Kaleem Siddiqi for 308-558B
School of Computer Science, McGill University

Introduction

Using linear combinations of models to solve object recognition problems is not a new idea, still it's widely use in current research. One of the motivations for using linear models is that they enable us to use well-known techniques from Linear Algebra to do the hard work. Indeed, powerful tools (ex: Matlab) exist to solve large system of linear equations, or to give optimally estimated solutions under the least squares criterion for any systems where there is more equations than there is unknown. It turns out that in most real world applications we have much more data, therefore equations, than unknowns. So using these tools is a big help to get solutions with reduced error.

Basic idea of linear model recognition

The underlying assumption behind recognition by linear combinations of models is that we can reconstruct a novel view of an object using linear combinations of the basis vectors. That is you can find a coordinate vector $\mathbf{g} = (g_1, g_2, \ldots, g_n)$ given the model basis $W = \{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}\}$ such that the novel view of the object is:

$$\mathbf{x} = g_1\mathbf{v_1} + g_2\mathbf{v_2} + \ldots + g_n\mathbf{v_n}.$$

(Where we use the standard notation of writing the vectors as bold characters). In this case each $\mathbf{v_i}$ and $\mathbf{x}$ are images vector and $\mathbf{g}$ is a coordinate vector of the novel view relative to the subspace W. This assumption is proved analytically for wire frame 3D object viewed under rigid 3-D transformations in [4] (which is a landmark paper in the field; there is a link to a PDF file in the reference section).

The recognition problem can be divided in two parts: building the model basis and using it to recognized novel views (images) of an object. There exist many different ways to build the model. In class we have seen appearance-based models. I give my notes on the subject in the next paragraph. However, I give you a small bonus paragraph on one

instance of a shape-based model that I have seen in Prof. Langer's course 308-646A. I'll conclude with my notes on the 2nd part of the problem: projection on the model subspace and how to use it for recognition.

Appearance based model

All good appearance based model starts with a huge database of images of modeled objects viewed from many directions in controlled environment. If the environment is not controlled a pre-processing phase is added. This is often the hardest problem: to get a set of images that can be combined on a per pixel basis. The pre-processing step can be a simple background-foreground segmentation to remove all background information from the images, which is very important if you want to obtain a low-dimensional subspace. You may also need a tracker to normalize the object absolute position in images, for example, using a color based centroid tracker.

For now, assume that we have a set of brightness and scale normalized images of all types of objects $\mathbf{U} = [\ \mathbf{X}^{(1)}_{1,1}, \ldots, \mathbf{X}^{(P)}_{R,L}\ ]_{N^2 \times m}$ express in matrix form where the $\mathbf{X}^{(p)}_{r,l}$ are column vectors which are actually the images (N^2-dimensional). We have m = RLP such images. R denotes the number of viewpoints, L the number of different lighting conditions and P the number of objects.

Once we have such a set we can perform principal component analysis PCA (also called Karhunen-Loeve decomposition (KL)). The following steps are easily done by software such as Matlab or with numerical recipes in C that can be found on the Web:

- Compute the average of all column vectors in **U** denoted $\mathbf{X_{avg}}$;
- Subtract the average from each column vector to get the dataset matrix $\mathbf{D}=[(\mathbf{X}^{(1)}_{1,1}-\mathbf{X_{avg}}), \ldots, (\mathbf{X}^{(1)}_{1,1}-\mathbf{X_{avg}})]_{N^2 \times m}$;
- Compute $\mathbf{Q}_{N^2 \times N^2} = \mathbf{D}\mathbf{D}^{\mathbf{T}}$;
- Compute the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_m$ of **Q** and the associated eigenvectors $\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_m}$;

Using the eigenspace theorem we know that with the eigenvalues and eigenvectors we can express any images $\mathbf{X_j}$ using a linear combinations of eigenvectors added to the average image $\mathbf{X_{avg}}$. That is:

$$\vec{X}_j = \tilde{X} + \sum_{i=1}^{m} g_{ji}\vec{e}_i$$

Where gji is the projection of $\mathbf{X_j}$ on $\mathbf{e_i}$.

It turns out that with PCA we can make the following approximation:

$$\vec{X}_j \approx \tilde{X} + \sum_{i=1}^{k} g_{ji}\vec{e}_i$$

Using a subset of the eigenvectors we can represent most of the variance of the data set. This subset is easy to choose because the magnitude of each eigenvalue is a measure of how much variance in the data set is due to the associated eigenvectors. So it's a simple matter of sorting the eigenvalues in decreasing order and taking the k greatest and their associated eigenvectors as the best possible k-dimensional basis for the new subspace.

Nice applications of this technique for face recognition can be found in [2,3], especially look in [3] to find images of the so-called principal "eigenfaces". You will see that these principal components do not look like humans face. Some people argued that "eigenfaces" are not efficient because most of the spawned subspace is not composed of human looking face. Therefore, its dimensionality could be further reduced.

Another way to approach the face recognition problem using linear models is found in [6,7]. Instead of using an appearance-based model, they use a shape and texture model. The shape is a vector field of the optical flow between a reference image and the image being modeled. Without going into the details, it turns out that segmenting the shape from texture gives a model subspace which is actually composed only of human looking faces. Look in [7] to find images taken along the 6 first principal components of shape and texture subspaces and compare them with the ones you saw in the previous paper. The result are stunning because all those images do not represent real people yet most of them look like people you could see on the bus.

Recognizing object with a novel image

To be able to recognize objects from novel images, we need to project images from the database on the universal. The set of all coordinate vectors from one object type in the eigenspace outlines an estimate of the object related subspace inside the eigenspace. We have seen such subspaces in class where they were represented as curves because there was only one independent parameter in the image formation process. However, more practical use of the eigenspace technique will usually require 20+-dimensional universal eigenspaces. It's quite hard to find analytical description that best fit a set of points in such a high dimensional spaces. In [2,5] this problem is skipped; they simply used the Euclidean distance from the projected point to all points in the database to find the closest one. This O(RLP) algorithm might be very inefficient for large database.

I need to make a small remark on the projection operation because in the textbook [5] in step 5 of the EIGENSPACE_LEARN algorithm they state without proof that the coordinate vector **g**, the projection of an image minus the average image on the eigenspace, is simply computed with the following dot product:

$$g_{ji} = (\mathbf{X_j} - \mathbf{X_{avg}}) \bullet \mathbf{e_i}$$

where g_{ji} is one component of the coordinate vector $\mathbf{g_j}$. Why is that so simple? In Linear Algebra, we have seen that projecting a vector on a subspace is somewhat more complicated. Indeed in [1], you can find the definition of the projection of a vector on a subspace:

> Let **b** be a vector in $\Re^n$, and let W be a subspace of $\Re^n$.
> Let $\mathbf{b} = \mathbf{b_W} + \mathbf{b_W}^{\perp}$ where $W^{\perp}$ is the orthogonal complement of W.
> Then $\mathbf{b_W}$ is the projection of **b** on W.

Using only this definition computing the projection of a vector on a subspace is a tedious process requiring the computation of $W^{\perp}$. However, there exist a much better approach. Indeed, if the basis of the subspace is orthogonal, that is the dot product between each basis vector is equal to zero, then the following theorem applies [1]:

> Let $\{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}\}$ be an orthogonal basis for the subspace W of $\Re^n$, and let **b** be any vector in $\Re^n$. The projection of **b** on W is

$$\vec{b}_W = \left(\frac{\vec{b}\bullet\vec{v}_1}{\vec{v}_1\bullet\vec{v}_1}\right)\vec{v}_1 + \left(\frac{\vec{b}\bullet\vec{v}_2}{\vec{v}_2\bullet\vec{v}_2}\right)\vec{v}_2 + \ldots + \left(\frac{\vec{b}\bullet\vec{v}_k}{\vec{v}_k\bullet\vec{v}_k}\right)\vec{v}_k$$

Furthermore, if the basis is orthonormal, all basis vectors have unit length and we can remove the denominators because they are all equal to one.

We could use the Gram-Schmidt Process to get an orthonormal basis for the eigenspace but it turns out that it's already orthogonal. Indeed, in [1] you can find a proof of the following theorem:

> Eigenvectors of a symmetric matrix that correspond to different eigenvalues are orthogonal. That is, the eigenspaces of a symmetric matrix are orthogonal.

It's fairly obvious that $\mathbf{Q} = \mathbf{D}\mathbf{D}^{\mathbf{T}}$ is symmetric (just try out with a 2x2 matrix you will see that multiplying a matrix with its transpose always gives a symmetric matrix). Therefore, the eigenvectors $\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_m}$ are orthogonal. However, the eigenvectors need to be normalized. This explains why the projection operation can be done very efficiently with dot products.

Be sure to check the links in the reference section on the next page…

References

(Note: you must be connected with a McGill IP address to have access to links on this page because the papers are actually on the IEEE server. McGill has a VPN now, so you can get a McGill IP with Sympatico, Videotron or any other ISP.)

[1] J. B. Fraleigh, R. A. *Beauregard: Linear Algebra*, Addison-Wesley, 1995.

[2] T. Jebara. *3D Pose Estimation and Normalization for Face Recognition*. Bachelor's Thesis, McGill Centre for Intelligent Machines. McGill University, 1996. Link: http://jebara.www.media.mit.edu/people/jebara/uthesis/

[3] M. Kirby and L. Sirovich. Appliction of the Karhumen-Loeve Procedure for the Characterization of Human Faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):103--108, 1990. Link:http://ieeexplore.ieee.org/iel1/34/1584/00041390.pdf

[4] S. Ullman and R. Basri. Recognition by Linear Combinations of Models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(10):992--1006, October 1991. Link : http://ieeexplore.ieee.org/iel1/34/3135/00099234.pdf

[5] E. Trucco and A. Verri. *Introductory techniques for 3-D computer vision*. Prentice Hall, 1998.

[6] T. Vetter, M. J. Jones, and T. Poggio. A bootstrapping algorithm for learning linear models of object classes. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 40--47, 1997. Link: http://graphics.informatik.uni-freiburg.de/publication.html (Can also be found on IEEE but the copy on this page is nicer)

[7] T. Vetter and N. Troje. A separated linear shape and texture space for modeling two-dimensional images of human faces. *Technical Report, MPI for biological Cybernetics, TR15*, 1995. Link: ftp://ftp.kyb.tuebingen.mpg.de/pub/mpi-memos/pdf/TR-015.pdf