CSci 4968 and 6270
Computational Vision,
Fall Semester, 2011-2012
Lecture 17: Overview of Recognition; Classifiers and PCA
October 31, 2011

1 Introduction

Overview

- What we know
  - Image processing
  - Edge and feature extraction; SIFT
  - Camera and transformation models
  - Estimation
- Our naive/intuitive sense of the main application of computer vision is "recognizing" images and individual objects in images. This is the subject of the next several weeks.
- Chapter 14 of the Szeliski text gives a good overview of recognition.
  - We will discuss fewer techniques, but more detail about each technique.
- The methods we discuss will make surprisingly little use of object geometry — the spatial arrangement of features — even though we have learned about cameras and transformations.

Types of Recognition

- Specialized algorithms for important but specific categories:
  - Face detection
  - Face recognition
  - Pedestrian detection and recognition.
  - Cars
  - Many more...

  Some of these fall under the category of methods called *object detection*. Note that if a problem is important enough, specialized methods are appropriate.
- *Instance recognition*
  - Train a system on images of particular objects, and then recognize these objects in future images.
  - Examples: particular books or CD covers; particular locations
- *Class-level / category-level recognition*

  - Given images of objects grouped into separate categories — cars, people, books, trees, wagons, horses — label objects in future images by assigning them to their appropriate categories
  - Here is a simple example of the difference between instance-level and category-level recognition:
    * Recognize that a picture is of a particular book; this is instance-level recognition.
    * Recognize that a picture is of a book, without necessarily identifying the particular book; this is category recognition.

- *Scene labeling*
  - Given an image, label all of the objects in it: grass, trees, water, roads, buildings, people, and sky.
  - Makes heavy use of context

Our Discussion

- Introduction to classifiers, including face recognition using principal components analysis and linear discriminant analysis
- Instance recognition:
  - Start from Lowe
  - Main focus, starting Thursday, will be a paper on the "Video Google" technique:

    Sivic, J. and Zisserman, A. Efficient visual search of videos cast as text retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **31(4)**:591606, 2009.
- Object / face detection

  Viola, P. A. and Jones, M. J. Robust real-time face detection. *International Journal of Computer Vision* **57(2)**:137154, 2004.
- Category-level recognition based on the "bag-of-words" approach.

2 Classifers, PCA and Face Recognition

Mathematical Background

We will start with some of the mathematical basis of "projections", dimensionality reduction, means and covariances....

- The length of the projection of one K-dimensional vector, $\mathbf{x}$, onto another K-dimensional vector, $\mathbf{y}$, is
$$\mathbf{x} \cdot \mathbf{y} = \mathbf{x}^\top \mathbf{y} = \mathbf{y}^\top \mathbf{x} \tag{1}$$
and the resulting vector is
$$(\mathbf{x}^\top \mathbf{y})\mathbf{y} \tag{2}$$

- The projection of $\mathbf{x}$ onto m different vectors, $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m$, each of length K, may be written using matrix notation as
$$\mathbf{x}' = \begin{pmatrix} \mathbf{y}_1^\top \\ \mathbf{y}_2^\top \\ \cdots \\ \mathbf{y}_m \end{pmatrix} \mathbf{x} \tag{3}$$
or more simply
$$\mathbf{x}' = \mathbf{Y}^\top \mathbf{x} \tag{4}$$
where $\mathbf{Y}$ is the "projection matrix"
$$\mathbf{Y} = \begin{pmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_m \end{pmatrix} \tag{5}$$
In computer vision, we will usually have the $\mathbf{y}_i$ vectors be unit vectors and orthogonal to each other, with $m \ll K$.

- Let $\{\mathbf{x}_i\}$ be a set of N data vectors, each of length K.

- Then, the mean of the vectors is
$$\boldsymbol{\mu} = \frac{1}{N} \sum_i \mathbf{x}_i. \tag{6}$$

- The covariance matrix of the vectors is
$$\boldsymbol{\Sigma} = \frac{1}{N} \sum_i (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^\top \tag{7}$$

- The spectral decomposition (also called the eigenvalue / eigenvector decomposition) is
$$\boldsymbol{\Sigma} = \sum_{k=1}^{K} \lambda_k \boldsymbol{\gamma}_k \boldsymbol{\gamma}_k^\top \tag{8}$$
where
  - The $\boldsymbol{\gamma}_k$ are orthogonal unit eigenvectors, and
  - $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_K$.

- Think of what these eigenvalues might look like when the data are clustered along a line in 2d or 3d or a plane in 3d (or higher):
  - We will have large eigenvalues for the eigenvectors along the line or the eigenvectors in the plane.
  - We will have small eigenvalues for the eigenvectors orthogonal to the line or plane.
- The eigenvalue/spectral decomposition of a matrix is the mathematical starting point for many techniques in classifiers.

Classifiers: Problem Statement

Given N feature point vectors $\mathbf{x}_i \in \mathcal{R}^K$ and class labels y_i, "learn" a function $f(\mathbf{x})$ that maps future feature point vector $\mathbf{x}$ to class labels.

- The vectors $\mathbf{x}_i$ could be
  - Pixel values
  - SIFT descriptor vectors
  - Entire images!!
- For the common two-class labeling problem, the labels y_i are usually either $\{0, 1\}$ or $\{-1, 1\}$. The choice between 0 and -1 is usually made for mathematical convenience.
  - As an example, in the face detection problem, the value 1 could represent "face" and the value -1 (or 0) could represent "not a face".
- When there are more than two labels, we have a "multi-class" labeling problem.
  - As an example, in the face recognition problem, we could have a unique label for each face stored in our database.

The Simplest Classifier: Nearest Neighbor(s)

- For each input $\mathbf{x}$ find the $\mathbf{x}_i$ closest to it according some measure of distance, and use the associated y_i as the label for $\mathbf{x}$.
- Mathematically we write this as

$$f(\mathbf{x}) = \{y_i \mid i = \underset{j=1,\ldots,N}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{x}_j\|\}$$

  where argmin returns the label that minimimizes the distance function $\|\mathbf{x} - \mathbf{x}_j\|$.
- This generalizes to making the label decision function f depend on the $k > 1$ nearest neighbors. This could be the majority or some distance-weighted average and threshold.
- This asymptototically close to optimal!
  - As the number of samples goes to infinity the classification accuracy is closer and closer to perfect.

- Generalizes nicely to the multi-class labeling problem.
- Major problem: finding the closest point in high-dimensions can be prohibitively slow!
- Solutions:
  - Selectively eliminate points from the data set
  - Approximation algorithms
  - Dimensionality reduction — map the points onto a lower dimensional space and work in that space.

Dimensionality Reduction: PCA

Principal Components Analysis, or PCA, is a straightforward method of dimensionality reduction, and as such it is widely used. One of the first applications in computer vision was face recognition. It seems like quite an intellectual leap to get from dimensionality reduction to face recogntion, so we'll start with the basics...

- As before, let $\{\mathbf{x}_i\}$ be a set of N data vectors, each of length K.
- Let
$$\boldsymbol{\mu} = \frac{1}{N}\sum_i \mathbf{x}_i \tag{9}$$
be the mean of the vectors.
- The covariance matrix of the vectors is
$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_i (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^\top \tag{10}$$
- We could compute the full spectral/eigenvalue decomposition,
$$\boldsymbol{\Sigma} = \sum_{k=1}^{K} \lambda_k \boldsymbol{\gamma}_k \boldsymbol{\gamma}_k^\top \tag{11}$$
but when K is large (e.g. 10,000 for a 100x100 image), this can be prohibitively expensive.
- In principal component analysis (PCA), we compute only the first $K_0 \ll K$ eigenvectors, $\boldsymbol{\Gamma} = \left(\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_{K_0}\right)$, using one of a variety of different techniques.
- We then use $\boldsymbol{\Gamma}$ as a low-dimensional basis for representing the data set $\{\mathbf{x}_i\}$.
- A data vector, $\mathbf{x}_j$, is projected onto these K_0 eigenvectors to form a dramatically-shortened vector:
$$\mathbf{x}'_j = \boldsymbol{\Gamma}^\top(\mathbf{x}_j - \boldsymbol{\mu}) = \begin{pmatrix} \boldsymbol{\gamma}_1^\top(\mathbf{x}_j - \boldsymbol{\mu}) \\ \boldsymbol{\gamma}_2^\top(\mathbf{x}_j - \boldsymbol{\mu}) \\ \cdots \\ \boldsymbol{\gamma}_{K_0}^\top(\mathbf{x}_j - \boldsymbol{\mu}) \end{pmatrix} \tag{12}$$
- Since $K_0 \ll K$, this is a much more compact representation than storing the entire vector $\mathbf{x}$.

- It is very important that the values λ_k drop off very quickly with k. Thus, $\mathbf{\Gamma}$ records the axes along which the data varies most.
- In this case, a large amount of the variation in the data has been captured
- Typically, K_0 might be in the range 20-50, whereas K, the length of the original vector, might have on the order of a thousand, ten thousand or one hundred thousand!
- As you might guess, a number of statistical methods have been developed to estimate the appropriate K_0.

Eigenfaces

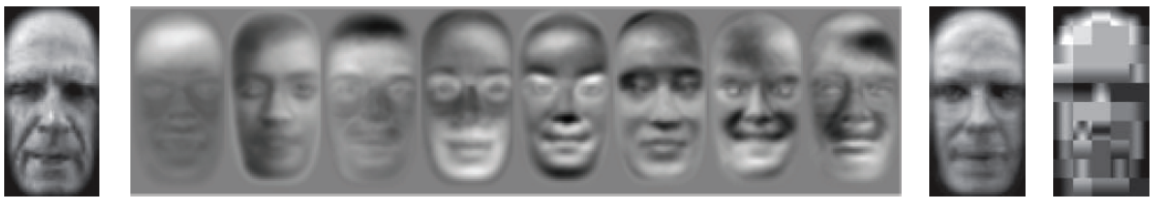
- Data are face-front images, cropped and scaled to the same size
  - Other methods can be applied to pre-process the images to obtain this data.
- Normalize to unit magnitude (of the intensities recorded in the image).
- Convert each image matrix to a vector $\mathbf{x}$ using, e.g., a raster-scan in row-major order.
- The label, y, associated with each vector, $\mathbf{x}$, identifies the person.
- Apply PCA to the training images to obtain our mean image $\boldsymbol{\mu}$ and projection matrix $\mathbf{\Gamma}$.
- Project each training image $\mathbf{x}_i$ onto the low-dimensional space:

$$\mathbf{x}'_i = \mathbf{\Gamma}^\top(\mathbf{x}_i - \boldsymbol{\mu}) \tag{13}$$

- For each new face image that we are trying to "recognize",
  1. Normalize and convert to vector, $\mathbf{x}$
  2. Project onto the low-dimensional "face-space"

$$\mathbf{x}' = \mathbf{\Gamma}^\top(\mathbf{x} - \boldsymbol{\mu}) \tag{14}$$

  3. Find the closest projected training vector $\mathbf{x}'_i$ and "recognize" by taking the label y_i.
- Note that this is still a nearest-neighbor classifier, but think of the dimensionality reduction!
- Here is an example from the Szeliski text (original is in Moghaddam, B. and Pentland, A. Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7)*, 696710, 1997).



(a) (b) (c) (d)

- Note that (c) is the eigenface reconstruction of (a) using just 85 bytes, whereas (d) is the the jpeg compression of (a) into 530 bytes.

Variations

- Ignore the first few vectors — most capture lighting differences among the images.
- Minimize the Mahalanobis distance in order to find the closest i.
- Important: apply PCA to recognize parts of faces, such as eyes. Think carefully about how this might be done.

Linear Discriminants and Fisher Discriminants

These are true classifiers — PCA is not.

- The decision function is
$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x} - c)$$
- Our problem is to learn the projection vector $\mathbf{x}$ and the offset c.
- The linear discriminant assumes that the $\mathbf{x}_i$ values in the two classes have the same full-rank covariance matrix, $\mathbf{\Sigma}$, and different means, $\boldsymbol{\mu}_0$ and $\boldsymbol{\mu}_1$. These are all easily estimated from the training data.
  - For high dimensional $\mathbf{x}_i$ vectors, we can apply PCA first.
- Then we have
$$\mathbf{w} = \mathbf{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$$
- We set c based on the projections of the $\mathbf{x}_i$ onto $\mathbf{w}$ according to whatever criteria we wish.
- When the two distributions are different, Fisher's linear discriminant, derived based on trying to maximize the ratio of the between-class variance and the within-class variance, sets
$$\mathbf{w} = (\mathbf{\Sigma}_1 + \mathbf{\Sigma}_0)^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$$
- We can understand these better by examining them in terms of the projection onto the basis vectors formed by the spectral decomposition of the covariance matrices.
- There are generalizations to multiclass labelings, which we will not discuss here.

Summary

- Overview of recognition:
  - Object detection, object recognition, category recognition, and scene labeling.
- Mathematical basics
  - Projections, projection matrices,
  - Mean vectors and covariance matrices,

  - Spectral/eigenvalue decompositions
- Nearest neighbor classifiers
- PCA and eigenfaces
- Linear discriminants

Next we'll cover the Sivic and Zisserman paper referenced at the beginning of the notes