Recognition II

Computer Vision I
CSE252A
Lecture 17

Announcements

• HW3 extended to tonight
• HW4 to be announced today. Due Friday 12/8. Note will take a while to run some things.
• Final Exam: Thursday 12/14 at 7pm-10pm

Example: Face Detection

• Scan window over image.
• Classify window as either:
  – Face
  – Non-face

Classifier

Window

Face
Non-face

See for example, Viola-Jones face detector in OpenCV

Evaluating a binary classifier

• For a detector, there are two types of errors:
  – False Positives, False accept (e.g., non-face is detected as a face)
  – False Negatives, False Reject (e.g., face is missed)
• ROC Curve (Receiver Operator Characteristic)-Plot of tradeoff between False Positives and false negatives

ROC Curve

Evaluating Multi-class classifiers

CIFAR 10
60,000 32x32 color images in 10 classes

Evaluating Multi-class classifiers

Overall accuracy
• Confusion Matrix ~ Example from Coral Reef Classification

Confusion Matrix

2008, 2009 ⇒ 2010 (83.1%)
**Nearest Neighbor Classifier**

\[ \{ R_j \} \text{ are set of training images.} \]

\[ ID = \arg \min_j dist(R_j, I) \]

**K-th Nearest Neighbor Classification**

**Maximum a posteriori classifier (MAP)**

\[ g_j(x) = \frac{P(x|\omega_j)P(\omega_j)}{P(x)} \]

Classification: \[ j = \arg \max_j g_j(x) \]

**Curse of Dimensionality**

- If we want to build a minimum-error rate classifier then we need a very good estimate of \( P(\omega| x) \)

- How do we do this?

- Let’s say our feature space is just 1-dimensional and our feature \( x \in [0, 1] \)

- And let’s say we have 10,000 training samples from which to estimate our \textit{a posteriori} probabilities.

- We could estimate these probabilities using a histogram in which we divided the interval into 100 evenly spaced bins.

- FIGURE 4.15. The k-nearest-neighbor query starts at the test point \( x \) and grows a spherical region until it encloses \( k \) training samples, and it labels the test point by a majority vote of these samples. In this \( k = 5 \) case, the test point would be labeled the category of the black points. From: Richard O. Duda, Peter E. Hart, and David G. Stork: Pattern Classification. Copyright \( \odot \) 2001 by John Wiley & Sons, Inc.
On average each bin would have 100 samples.
We could estimate \( P(x | \omega_i) \) as the number of samples from class \( i \) that fall in the same bin that falls into divided by the total number of samples in that bin.

But this plan does not scale as we increase the dimensionality of the feature space!
- Let’s say our feature space is just 3-D dimensional and our feature \( x \in [0,1]^n \)
- Let’s say we still have 10,000 training samples from which to estimate our \( a \) posteriori probabilities.
- If we estimate these probabilities using a histogram in which we divide the volume into the same width bins as before...

Dimensionality Reduction

Dimensionality reduction: linear projection

- An \( n \)-pixel image \( x \in \mathbb{R}^n \) can be projected to a low-dimensional feature space \( y \in \mathbb{R}^m \) by
  \[
y = W^T x
\]
  where \( W \) is an \( n \) by \( m \) matrix.
- Recognition is performed using nearest neighbor in \( \mathbb{R}^m \).
- How do we choose a good \( W \)?

How do we choose a good \( W \)?

- Drop dimensions (feature selection)
- Random projects
- Principal component analysis
- Linear discriminant analysis
- Independent component analysis
- Or even non-linear dimensionality reduction
Eigenfaces: Principal Component Analysis (PCA)

Assume we have a set of n feature vectors \( x_i \) (\( i = 1, \ldots, n \)) in \( \mathbb{R}^d \). Write

\[
\mu = \frac{1}{n} \sum x_i
\]

\[
Y = \frac{1}{n-1} \sum (x_i - \mu)(x_i - \mu)^T
\]

The unit eigenvectors of \( Y \) — which we write as \( v_1, v_2, \ldots, v_k \) where the order is given by the size of the eigenvalue and \( v_1 \) has the largest eigenvalue — give a set of features with the following properties:

- They are independent.
- Projection onto the basis \( \{v_1, \ldots, v_k\} \) gives the \( k \)-dimensional set of linear features that preserves the most variance.

Algorithm 22.5: Principal components analysis identifies a collection of linear features that are independent, and capture as much variance as possible from a dataset.

Some details: Use Singular value decomposition, “trick” described in text to compute basis when \( n << d \)

Singular Value Decomposition

- Any \( m \times n \) matrix \( A \) may be factored such that \( A = U \Sigma V^T \)

\[
\begin{bmatrix}
    m & n
\end{bmatrix} = \begin{bmatrix}
    m & m & n \end{bmatrix} \begin{bmatrix}
    m & n
\end{bmatrix}
\]

- \( U : m \times m \), orthogonal matrix
  - Columns of \( U \) are the eigenvectors of \( AA^T \)
- \( V : n \times n \), orthogonal matrix,
  - Columns are the eigenvectors of \( A^T A \)
- \( \Sigma : m \times n \), diagonal with non-negative entries (\( \sigma_1, \sigma_2, \ldots, \sigma_s \)) with \( s = \min(m,n) \) are called the singular values
  - Singular values are the square roots of eigenvalues of both \( AA^T \) and \( A^T A \) and \( A^T A, U \) and \( U^T = \) Corresponding Eigenvectors
  - Result of SVD algorithm: \( \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_s \)

Performing PCA with SVD

- Singular values of \( A \) are the square roots of eigenvalues of both \( AA^T \) and \( A^T A \) & Columns of \( U \) are corresponding Eigenvectors
  - And \( \sum x_i u_i = [a_1, a_2, \ldots, a_n] \) \( A u_i \) & Columns of \( U \) are Corresponding Eigenvectors
  - Covariance matrix is:
    \[
    \Sigma = \frac{1}{n} \sum (x_i - \mu)(x_i - \mu)^T
    \]
    - So, ignoring \( 1/n \) subtract mean image \( \mu \) from each input image, create data matrix, and perform thin SVD on the data matrix.

SVD Properties

- In Matlab \( [u \ s \ v] = \text{svd}(A) \), and you can verify that: \( A = u \text{diag}(s) v^T \)
- \( r = \text{Rank}(A) = \# \) of non-zero singular values.
- \( U, V \) give us orthonormal bases for the subspaces of \( A \):
  - 1st \( r \) columns of \( U \): Column space of \( A \)
  - Last \( m - r \) columns of \( U \): Left nullspace of \( A \)
  - 1st \( r \) columns of \( V \): Row space of \( A \)
  - Last \( n - r \) columns of \( V \): Nullspace of \( A \)
- For \( d \leq r \), the first \( d \) columns of \( U \) provide the best \( d \)-dimensional basis for columns of \( A \) in least squares sense.

Comment on images collections

\[
A = U \Sigma V^T
\]

\[
[m \times n] = [m \times m][m \times n][n \times n]
\]

- The matrix \( A \) is sometimes called the data matrix.
- Columns of \( A \) are vectorized images.
- So, we have \( m \) pixels and \( n \) images.
- For large images (e.g., 1k x 1k), we often have more \( n > m \). Using SVD is preferred.
- For CIFAR, we have \( m = 3072 \), and we have \( n = 60k \) and so explicit form with covariance matrix or SVD can work.
**Thin SVD**

- Any $m$ by $n$ matrix $A$ may be factored such that
  
  $A = U \Sigma V^T$
  
  $[m \times n] \times [m \times m] \times [n \times n]$

- If $m > n$, then one can view $\Sigma$ as:

  
  $\Sigma = \begin{bmatrix} 
  \Sigma & \\
  0 & 
  \end{bmatrix}$

  
  - Where $\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_s)$ with $s = \min(m,n)$, and lower matrix is $(n-m \times m)$ of zeros.

  
  - Alternatively, you can write:

    $A = U' \Sigma' V^T$

  
  - In Matlab, thin SVD is: $[U \ S \ V'] = \text{svds}(A)$

**PCA for recognition (Eigenfaces)**

- **Modeling**
  1. Given a collection of $n$ labeled training images,
  2. Compute mean image and covariance matrix.
  3. Compute $k$ Eigenvectors (note that these are images) of covariance matrix corresponding to $k$ largest Eigenvalues. (Or perform using SVD!!)
  4. Project the training images to the $k$-dimensional Eigenspace.

- **Recognition**
  1. Given a test image, project to Eigenspace.
  2. Perform classification to the projected training images.

---

**Eigenfaces:**

**Training Images**

![Eigenfaces: Training Images](image)

[ Turk, Pentland 01]

**Eigenfaces**

**Mean Image**

![Mean Image](image)

**Basis Images**

![Basis Images](image)

---

**Accuracy of PCA + K-NN**

<table>
<thead>
<tr>
<th>Method</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN + 3,072 Features</td>
<td>33.86</td>
</tr>
<tr>
<td>KNN + 200 PCA Comp.</td>
<td>36.54</td>
</tr>
<tr>
<td>KNN + 75 PCA Comp.</td>
<td>39.77</td>
</tr>
<tr>
<td>KNN + 50 PCA Comp.</td>
<td>40.12</td>
</tr>
<tr>
<td>KNN + 40 PCA Comp.</td>
<td>40.93</td>
</tr>
<tr>
<td>KNN + 30 PCA Comp.</td>
<td><strong>41.78</strong></td>
</tr>
<tr>
<td>KNN + 25 PCA Comp.</td>
<td>41.57</td>
</tr>
<tr>
<td>KNN + 15 PCA Comp.</td>
<td>38.75</td>
</tr>
<tr>
<td>KNN + 10 PCA Comp.</td>
<td>34.93</td>
</tr>
</tbody>
</table>

---

**Difficulties with PCA**

- Projection may suppress important detail
  - smallest variance directions may not be unimportant

- Method does not take discriminative task into account
  - typically, we wish to compute features that allow good discrimination
  - not the same as largest variance or minimizing reconstruction error.
**Fisherfaces: Class specific linear projection**

- An $n$-pixel image $x \in \mathbb{R}^n$ can be projected to a low-dimensional feature space $y \in \mathbb{R}^m$ by
  
  $$y = W^T x$$

  where $W$ is an $n \times m$ matrix.

- Recognition is performed using nearest neighbor in $\mathbb{R}^m$.

- How do we choose a good $W$?

**PCA & Fisher’s Linear Discriminant**

- **PCA (Eigenfaces)**
  - $W_{PC} = \arg \max_w \|W^T S_w W\|$
  - Maximizes projected total scatter

- **Fisher’s Linear Discriminant**
  - $W_{FLD} = \arg \max_w \frac{\|W^T S_b W\|}{\|W^T S_w W\|}$
  - Maximizes ratio of projected between-class to projected within-class scatter

- Since $S_b$ is rank $N-c$, project training set to subspace spanned by first $N-c$ principal components of the training set.

- Apply FLD to $N-c$ dimensional subspace yielding $c-1$ dimensional feature space.

- Fisher’s Linear Discriminant projects away the within-class variation (lighting, expressions) found in training set.

- Fisher’s Linear Discriminant preserves the separability of the classes.

**Harvard Face Database**

- 10 individuals
- 66 images per person
- Train on 6 images at 15°
- Test on remaining images
Recognition Results: Lighting Extrapolation

![Graph showing recognition results for different lighting directions. The x-axis represents light direction (0-15 degrees, 30 degrees, 45 degrees), and the y-axis represents error rate. Different methods are compared: Correlation, Eigenfaces, Eigenfaces (w/o 1st 3), Fisherface.](image-url)