Eigenface

Outline

• Performance measurement
• Dimensionality reduction
• Face representation
  - Eigenfaces
Face Classification

20 faces (i.e., classes), 9 examples (i.e., training data) of each

Measuring Performance

• Classification accuracy:
  - The percentage of correctly labeled data by a classifier
• Validation: Split experimental data into training and test set
  - **Training set**: Data points used to build a classifier
  - **Test set**: Data points left out of training procedure
Cross-Validation

• \( m \)-fold **cross-validation**
  - Randomly split data into \( m \) equal-sized subsets
  - Train \( m \) times on \( m - 1 \) subsets, test on left-out subset
  - Error is mean test error over left-out subsets

• **Leave-one-out**: Cross-validation with 1-point subsets
  - Very accurate but expensive; variance allows confidence measuring

A Simple Method

• Idea: Search over training set for most similar image (e.g., in SSD sense) and choose its class
• This is the same as a 1-nearest neighbor classifier when feature space = image space
• Issues
  - Large storage requirements (\( nd \), where \( n \) = image space dimensionality and \( d \) = number of faces in training set)
  - Correlation is computationally expensive
Dimension reduction: Motivation

• Using images themselves as feature vectors is easy, but has problem of high dimensionality
  • A 100 x 100 image = 10,000-dimensional feature space!
• We want features that stay in low-dimensional spaces and result in well-separated classes

Dimensionality Reduction: Projection

• Two popular schemes
  – Principal components analysis (PCA): Projection maximizing total variance of data
    \[ S_T = \sum_{k=1}^{N} (x_k - \mu)(x_k - \mu)^T \]
  – Fisher’s Linear Discriminant (FLD): Maximize ratio of between-class variance to within-class variance
Geometric Interpretation

Covariance $\mathbf{C} = \mathbf{X} \mathbf{X}^T$
where $\mathbf{X}$ is zero-mean

Geometric Factorization of Covariance

• SVD of covariance matrix $\mathbf{C} = \mathbf{R} \mathbf{D} \mathbf{R}^T$ describes geometric components of transform by extracting:
  - Diagonal scaling matrix $\mathbf{D}$
  - Rotation matrix $\mathbf{R}$

• E.g., given points $\mathbf{x} = \begin{pmatrix} 2 & -2 & 1 & -1 \\ 5 & -5 & -1 & 1 \end{pmatrix}$, the covariance factors as

  $\mathbf{X} \mathbf{X}^T = \begin{pmatrix} 2.5 & 5 \\ 5 & 13 \end{pmatrix} = \begin{pmatrix} 0.37 & -0.93 \\ 0.93 & 0.37 \end{pmatrix} \begin{pmatrix} 15 & 0 \\ 0 & 5 \end{pmatrix} \begin{pmatrix} 0.37 & 0.93 \\ -0.93 & 0.37 \end{pmatrix}$

  major, minor axis lengths

(adapted from Z. Dodds)
PCA for Dimensionality Reduction

• Any point in $n$-dimensional original space can thus be expressed as a linear combination of the $n$ eigenvectors (the rows of $\mathbf{R}$) via a set of weights $[\omega_1, \omega_2, \ldots, \omega_n]$.

• By projecting points onto only the first $k << n$ principal components (eigenvectors with the largest eigenvalues), we are essentially throwing away the least important feature information.

Projection onto Principal Components

Full $n$-dimensional space (here $n = 2$) $\rightarrow$ $k$-dimensional subspace (here $k = 1$)

adapted from Z. Dodds
Eigenfaces

- Idea: Compress image space to “face space” by projecting onto principal components ("eigenfaces" = eigenvectors of image space)
  - Represent each face as a low-dimensional vector (weights on eigenfaces)
  - Measure similarity in face space for classification
- Advantage: Storage requirements are $nk$ instead of $nd$

Eigenfaces: Initialization

- Calculate eigenfaces
  - Compute $n$-dimensional mean face $\Psi$
  - Compute difference of every face from mean face $\Phi_j = \Gamma_j - \Psi$
  - Form covariance matrix of these $C = AA^T$, where $A = [\Phi_1, \Phi_2, \ldots, \Phi_d]$
  - Extract eigenvectors $u_i$ from $C$ such that $Cu_i = \lambda_i u_i$
  - Eigenfaces are $k$ eigenvectors with largest eigenvalues

Example eigenfaces
Eigenfaces: Initialization

- Project faces into face space
  - Get eigenface weights for every face in the training set
  - The weights \([\omega_{j1}, \omega_{j2}, \ldots, \omega_{jk}]\) for face \(j\)
    are computed via dot products \(\omega_{ji} = u_i^T \Phi_j\)

Calculating Eigenfaces

- Obvious way is to perform SVD of covariance matrix, but this is often prohibitively expensive
  - E.g., for 128 x 128 images, \(C\) is 16,384 x 16,384
- Consider eigenvector decomposition of \(d \times d\) matrix \(A^T A\): \(A^T A v_i = \lambda_i v_i\). Multiplying both sides on the left by \(A\), we have
  \[C A v_i = \lambda_i A v_i\]
- So \(u_i = A v_i\) are the eigenvectors of \(C = AA^T\)
Eigenfaces: Recognition

• Project new face into face space
• Classify
  – Assign class of nearest face from training set

Original face → 8 eigenfaces → \([\omega_1, \omega_2, \ldots, \omega_8]\)

adapted from Z. Dodds