Expectation Maximization

Machine Learning – CSE546
Carlos Guestrin
University of Washington

November 13, 2014

E.M.: The General Case

- E.M. widely used beyond mixtures of Gaussians
  - The recipe is the same…
- Expectation Step: Fill in missing data, given current values of parameters, $\theta^{(t)}$
  - If variable ~~y~~ z is missing (could be many variables)
  - Compute, for each data point $\mathbf{x}^j$, for each value i of z:
    - $P(z=i|\mathbf{x}^j,\theta^{(t)})$
- Maximization step: Find maximum likelihood parameters for (weighted) "completed data":
  - For each data point $\mathbf{x}^j$, create k weighted data points
    -
  - Set $\theta^{(t+1)}$ as the maximum likelihood parameter estimate for this weighted data
- Repeat

The general learning problem with missing data

- Marginal likelihood – **x** is observed, **z** is missing:

max θ

$$\ell(\theta : \mathcal{D}) = \log \prod_{j=1}^{m} P(\mathbf{x}_j \mid \theta)$$

observed part

$$= \sum_{j=1}^{m} \log P(\mathbf{x}_j \mid \theta)$$

$$= \sum_{j=1}^{m} \log \sum_{\mathbf{z}=1}^{k} P(\mathbf{x}_j, \mathbf{z} \mid \theta)$$

estimate z

max θ

latent

E-step

weight

- **x** is observed, **z** is missing
- Compute probability of missing data given current choice of $\theta^{(t)}$
  - Q(**z**|**x**$_j$) for each **x**$_j$
    - e.g., probability computed during classification step
    - corresponds to "classification step" in K-means

$$Q^{(t+1)}(\mathbf{z}=i \mid \mathbf{x}_j) = P(\mathbf{z}=i \mid \mathbf{x}_j, \theta^{(t)})$$

anything

θ of previous iteration

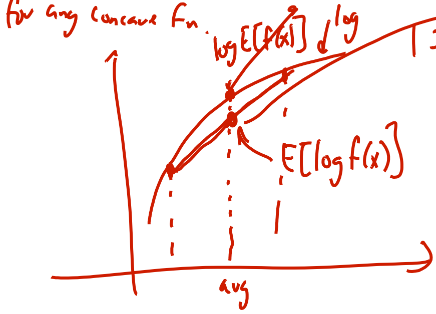
Jensen's inequality

$$\ell(\theta : \mathcal{D}) \;=\; \sum_{j=1}^{m} \log \sum_{\mathbf{z}} P(\mathbf{z} \mid \mathbf{x}_j) P(\mathbf{x}_j \mid \theta)$$

log avg function avg logs

- **Theorem**: log $\sum_{\mathbf{z}}$ P(**z**) f(**z**) ≥ $\sum_{\mathbf{z}}$ P(**z**) log f(**z**)

for any concave fn.



$1 = \log_2(1+1) \geq \log 1 + \log 1 = 0$

$\log E[f(x)] \geq E[\log f(x)]$

Applying Jensen's inequality

$\log \frac{a}{b} = \log a - \log b$

- Use: log $\sum_{\mathbf{z}}$ P(**z**) f(**z**) ≥ $\sum_{\mathbf{z}}$ P(**z**) log f(**z**)

complex, instead max a lower bound

max θ

$$\ell(\theta^{(t)} : \mathcal{D}) \;=\; \sum_{j=1}^{m} \log \sum_{\mathbf{z}} \underbrace{Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j)}_{P} \overbrace{\frac{P(\mathbf{z}, \mathbf{x}_j \mid \theta^{(t)})}{Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j)}}^{f(z)}$$

$$\geq \sum_{j=1}^{m} \sum_{z=1}^{k} Q^{(t+1)}(z \mid x_j) \log \frac{P(z, x_j \mid \theta^{(t)})}{Q^{(t+1)}(z \mid x_j)}$$

$$= \underbrace{\sum_{j=1}^{m} \sum_{z=1}^{k} Q^{(t+1)}(z \mid x_j) \log P(z, x_j \mid \theta^{(t)})}_{\text{weighted max likelihood}} - \underbrace{\sum_{j=1}^{m} \sum_{z=1}^{k} Q^{(t+1)}(z \mid x_j) \log Q^{(t+1)}(z \mid x_j)}_{m H(Q^{(t+1)}) \text{ doesn't depend on } \theta!!}$$

The M-step maximizes lower bound on weighted data

- Lower bound from Jensen's:

want max θ

$$\ell(\theta^{(t)} : \mathcal{D}) \geq \underbrace{\sum_{j=1}^{m} \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j) \log P(\mathbf{z}, \mathbf{x}_j \mid \theta^{(t)})}_{\text{focus on this first term in M-step}} + m.H(Q^{(t+1)})$$

doesn't depend on θ

- Corresponds to weighted dataset:
  - $<\mathbf{x}_1, \mathbf{z}=1>$ with weight $Q^{(t+1)}(\mathbf{z}=1|\mathbf{x}_1)$ 0.3
  - $<\mathbf{x}_1, \mathbf{z}=2>$ with weight $Q^{(t+1)}(\mathbf{z}=2|\mathbf{x}_1)$ 0.6
  - $<\mathbf{x}_1, \mathbf{z}=3>$ with weight $Q^{(t+1)}(\mathbf{z}=3|\mathbf{x}_1)$ 0.1
  - $<\mathbf{x}_2, \mathbf{z}=1>$ with weight $Q^{(t+1)}(\mathbf{z}=1|\mathbf{x}_2)$:
  - $<\mathbf{x}_2, \mathbf{z}=2>$ with weight $Q^{(t+1)}(\mathbf{z}=2|\mathbf{x}_2)$:
  - $<\mathbf{x}_2, \mathbf{z}=3>$ with weight $Q^{(t+1)}(\mathbf{z}=3|\mathbf{x}_2)$:
  - …

The M-step

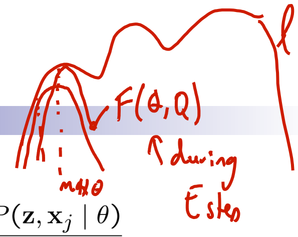
$$\ell(\theta^{(t)} : \mathcal{D}) \geq \sum_{j=1}^{m} \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j) \log P(\mathbf{z}, \mathbf{x}_j \mid \theta^{(t)}) + m.H(Q^{(t+1)})$$

- Maximization step: fixed choice of Q

$$\theta^{(t+1)} \leftarrow \arg\max_{\theta} \sum_{j=1}^{m} \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j) \log P(\mathbf{z}, \mathbf{x}_j \mid \theta)$$

- Use expected counts instead of counts:
  - If learning requires Count(**x**,**z**)
  - Use $E_{Q(t+1)}$[Count(**x**,**z**)]

Convergence of EM



- Define potential function F(θ,Q):

$$\ell(\theta : \mathcal{D}) \;\geq\; F(\theta, Q) = \sum_{j=1}^{m} \sum_{\mathbf{z}} Q(\mathbf{z} \mid \mathbf{x}_j) \log \frac{P(\mathbf{z}, \mathbf{x}_j \mid \theta)}{Q(\mathbf{z} \mid \mathbf{x}_j)}$$

- EM corresponds to coordinate ascent on F
  - Thus, maximizes lower bound on marginal log likelihood
  - We saw that M-step corresponds to fixing Q, max θ
  - E-step fix θ and max Q

coord ascent

↑ same kind of analysis

M-step is easy

$$\theta^{(t+1)} \leftarrow \arg\max_{\theta} \sum_{j=1}^{m} \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j) \log P(\mathbf{z}, \mathbf{x}_j \mid \theta)$$

- Using potential function

$$F(\theta, Q^{(t+1)}) \;=\; \sum_{j=1}^{m} \sum_{\mathbf{z}} Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j) \log P(\mathbf{z}, \mathbf{x}_j \mid \theta) + m.H(Q^{(t+1)})$$

E-step also doesn't decrease potential function 1

- Fixing θ to $\theta^{(t)}$:

$$\ell(\theta^{(t)} : \mathcal{D}) \;\geq\; F(\theta^{(t)}, Q) = \sum_{j=1}^{m} \sum_{\mathbf{z}} Q(\mathbf{z} \mid \mathbf{x}_j) \log \frac{P(\mathbf{z}, \mathbf{x}_j \mid \theta^{(t)})}{Q(\mathbf{z} \mid \mathbf{x}_j)}$$

KL-divergence

- Measures distance between distributions

$$KL(Q||P) = \sum_z Q(z) \log \frac{Q(z)}{P(z)}$$

- KL=zero if and only if Q=P

E-step also doesn't decrease potential function 2

- Fixing θ to $\theta^{(t)}$:

$$\ell(\theta^{(t)} : \mathcal{D}) \geq F(\theta^{(t)}, Q) = \ell(\theta^{(t)} : \mathcal{D}) + \sum_{j=1}^{m} \sum_{\mathbf{z}} Q(\mathbf{z} \mid \mathbf{x}_j) \log \frac{P(\mathbf{z} \mid \mathbf{x}_j, \theta^{(t)})}{Q(\mathbf{z} \mid \mathbf{x}_j)}$$

$$= \ell(\theta^{(t)} : \mathcal{D}) - \sum_{j=1}^{m} KL\left(Q(\mathbf{z} \mid \mathbf{x}_j) || P(\mathbf{z} \mid \mathbf{x}_j, \theta^{(t)})\right)$$

E-step also doesn't decrease potential function 3

$$\ell(\theta^{(t)} : \mathcal{D}) \geq F(\theta^{(t)}, Q) = \ell(\theta^{(t)} : \mathcal{D}) - m \sum_{j=1}^{m} KL\left(Q(\mathbf{z} \mid \mathbf{x}_j) || P(\mathbf{z} \mid \mathbf{x}_j, \theta^{(t)})\right)$$

- Fixing θ to $\theta^{(t)}$
- Maximizing $F(\theta^{(t)},Q)$ over Q → set Q to posterior probability:

$$Q^{(t+1)}(\mathbf{z} \mid \mathbf{x}_j) \leftarrow P(\mathbf{z} \mid \mathbf{x}_j, \theta^{(t)})$$

- Note that

$$F(\theta^{(t)}, Q^{(t+1)}) = \ell(\theta^{(t)} : \mathcal{D})$$

EM is coordinate ascent

$$\ell(\theta : \mathcal{D}) \geq F(\theta, Q) = \sum_{j=1}^{m} \sum_{\mathbf{z}} Q(\mathbf{z} \mid \mathbf{x}_j) \log \frac{P(\mathbf{z}, \mathbf{x}_j \mid \theta)}{Q(\mathbf{z} \mid \mathbf{x}_j)}$$

- **M-step**: Fix Q, maximize F over θ (a lower bound on $\ell(\theta : \mathcal{D})$):

$$\ell(\theta : \mathcal{D}) \geq F(\theta, Q^{(t)}) = \sum_{j=1}^{m} \sum_{\mathbf{z}} Q^{(t)}(\mathbf{z} \mid \mathbf{x}_j) \log P(\mathbf{z}, \mathbf{x}_j \mid \theta) + m.H(Q^{(t)})$$

- **E-step**: Fix θ, maximize F over Q:

$$\ell(\theta^{(t)} : \mathcal{D}) \geq F(\theta^{(t)}, Q) = \ell(\theta^{(t)} : \mathcal{D}) - m \sum_{j=1}^{m} KL\left(Q(\mathbf{z} \mid \mathbf{x}_j) || P(\mathbf{z} \mid \mathbf{x}_j, \theta^{(t)})\right)$$

  - "Realigns" F with likelihood:

$$F(\theta^{(t)}, Q^{(t+1)}) = \ell(\theta^{(t)} : \mathcal{D})$$

What you should know

- K-means for clustering:
  - algorithm
  - converges because it's coordinate ascent
- EM for mixture of Gaussians:
  - How to "learn" maximum likelihood parameters (locally max. like.) in the case of unlabeled data
- Be happy with this kind of probabilistic analysis
- Remember, E.M. can get stuck in local minima, and empirically it <u>DOES</u>
- EM is coordinate ascent
- General case for EM

Dimensionality Reduction PCA

Machine Learning – CSE4546
Carlos Guestrin
University of Washington

November 13, 2014

Dimensionality reduction

- Input data may have thousands or millions of dimensions! x
  - e.g., text data has 10,000 words – 10 000 000 000
- **Dimensionality reduction**: represent data with fewer dimensions
  - easier learning – fewer parameters
  - visualization – hard to visualize more than 3D or 4D
  - discover "intrinsic dimensionality" of data
    - high dimensional data that is truly lower dimensional

Lower dimensional projections

- Rather than picking a subset of the features, we can new features that are combinations of existing features

$z_1 = 2.5\, x_1 - 3.2\, x_2 - 3.1\, x_3 + 70\, x_4$

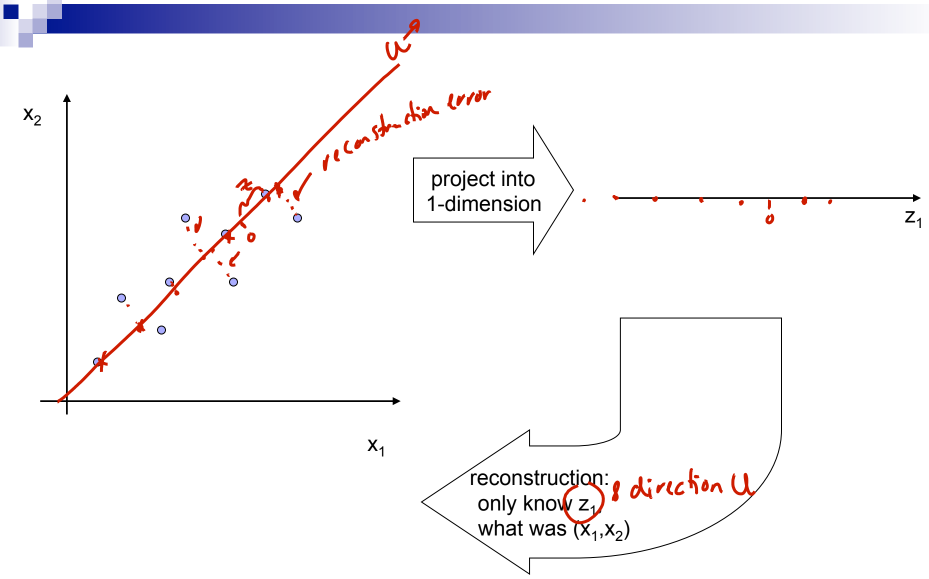
$x \xrightarrow{\text{learn}} z \xrightarrow{\text{reconstruct}} \hat{x}$

$\hat{x}$ is close to x

$z = {}^{k}A^{n}\, x$

- Let's see this in the unsupervised setting
  - just **X**, but no Y

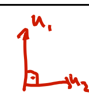
Linear projection and reconstruction



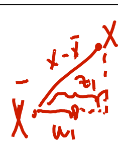
Principal component analysis – basic idea

- Project n-dimensional data into k-dimensional space while preserving information:
  - e.g., project space of 10000 words into 3-dimensions
  - e.g., project 3-d into 2-d
- Choose projection with minimum reconstruction error

Linear projections, a review



- Project a point into a (lower dimensional) space:
  - **point**: $\mathbf{x} = (x_1,\ldots,x_d)$ 10 000 0000
  - **select a basis** – set of basis vectors – $(\mathbf{u}_1,\ldots,\mathbf{u}_k)$ 10^6
    - we consider orthonormal basis:
      - $\mathbf{u}_i \cdot \mathbf{u}_i = 1$, and $\mathbf{u}_i \cdot \mathbf{u}_j = 0$ for $i \neq j$
  - **select a center** – $\bar{\mathbf{x}}$, defines offset of space
  - **best coordinates** in lower dimensional space defined by dot-products: $(z_1,\ldots,z_k)$, $z_i = (\mathbf{x}-\bar{\mathbf{x}}) \cdot \mathbf{u}_i$
    - minimum squared error



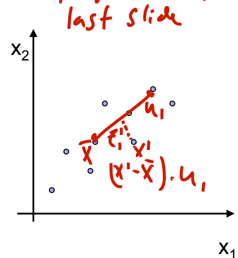
$$z_1 = (x-\bar{x})\cdot u_1 \quad z_1 = \arg\min_{z} \left((x-\bar{x}) - z u_1\right)^2$$

PCA finds projection that minimizes reconstruction error

- Given N data points: $\mathbf{x}^i = (x_1^i,\ldots,x_d^i)$, i=1…N
- Will represent each point as a projection:
  - $\hat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$ where: $\underbrace{\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}^i}_{\text{avg of data}}$ and $\underbrace{z_j^i = (\mathbf{x}^i - \bar{\mathbf{x}}) \cdot \mathbf{u}_j}_{\text{projections from last slide}}$
- PCA:
  - Given k<<d, find ($\mathbf{u}_1,\ldots,\mathbf{u}_k$) minimizing reconstruction error:

    for a choice of basis $u_1 \ldots u_k$

    $$\underbrace{error_k = \sum_{i=1}^{N} (\mathbf{x}^i - \hat{\mathbf{x}}^i)^2}$$



Understanding the reconstruction error

approx point → $\hat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$

coeffs → $z_j^i = (\mathbf{x}^i - \bar{\mathbf{x}}) \cdot \mathbf{u}_j$

Given k<<d, find ($\mathbf{u}_1,\ldots,\mathbf{u}_k$) minimizing reconstruction error:

$$error_k = \sum_{i=1}^{N} (\mathbf{x}^i - \hat{\mathbf{x}}^i)^2$$

- Note that $\mathbf{x}^i$ can be represented exactly by d-dimensional projection:

$$\mathbf{x}^i = \bar{\mathbf{x}} + \sum_{j=1}^{d} z_j^i \mathbf{u}_j$$

- Rewriting error:

min error$_k$ $= \sum_{i=1}^{N} (x^i - \hat{x}^i)^2 = \sum_{i=1}^{N} \left[\bar{x} + \sum_{j=1}^{d} z_j^i u_j - \left[\bar{x} + \sum_{j=1}^{k} z_j^i u_j \right] \right]^2 = \sum_{i=1}^{N} \left[\sum_{j=k+1}^{d} z_j^i u_j \right]^2$

$= \sum_{i=1}^{N} \left[\sum_{j=k+1}^{d} z_j^i \overbrace{u_j \cdot u_j}^{1} z_j^i + \sum_{j=k+1}^{d} \sum_{j' \neq j} z_j^i \overbrace{u_j \cdot u_{j'}}^{0} z_{j'}^i \right]$

$= \sum_{i=1}^{N} \sum_{j=k+1}^{d} (z_j^i)^2$ ← minimizing projection error ≡ min square of thrown out coeffs.

Reconstruction error and covariance matrix

memory lane

$\Sigma u = \lambda u$ ← eigen vector

$$\min_u \; error_k = \sum_{i=1}^{N} \sum_{j=k+1}^{d} [\underbrace{\mathbf{u}_j \cdot (\mathbf{x}^i - \bar{\mathbf{x}})}_{z_j^i}]^2$$

$$= \sum_{i=1}^{N} \sum_{j=k+1}^{d} u_j^T (x^i - \bar{x})(x^i - \bar{x})^T u_j$$

flip sums, u_j don't depend on i

$$= \sum_{j=k+1}^{d} u_j^T \underbrace{\left[\sum_{i=1}^{N} (x^i - \bar{x})(x^i - \bar{x})^T \right]}_{N\Sigma} u_j$$

$$= N \sum_{j=k+1}^{d} u_j^T \Sigma u_j$$ ← choose to ignore u_j that minimize this error

$$\Sigma = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} (\mathbf{x}^i - \bar{\mathbf{x}})(\mathbf{x}^i - \bar{\mathbf{x}})^T$$

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \\ & \sigma_2^2 & \\ & & \ddots \end{pmatrix}$$

$$\sigma_{rs}^{MLE} = \frac{1}{N} \sum_{i=1}^{N} (x_r^i - \bar{x}_r)(x_s^i - \bar{x}_s)$$

in vector form

Minimizing reconstruction error and eigen vectors

- Minimizing reconstruction error equivalent to picking orthonormal basis ($\mathbf{u}_1,\ldots,\mathbf{u}_d$) minimizing:

$$error_k = \mathrm{N} \sum_{j=k+1}^{d} \mathbf{u}_j^T \Sigma \mathbf{u}_j$$

- Eigen vector:

$\Sigma u = \lambda u$ (eigenvector, eigen value)

$u^T \Sigma u = \lambda u^T u = \lambda$

- Minimizing reconstruction error equivalent to picking ($\mathbf{u}_{k+1},\ldots,\mathbf{u}_d$) to be eigen vectors with smallest eigen values

ignored dims

$\min_{u_1 \ldots u_n} error_k$ $\equiv$ throwing out $\underbrace{u_{k+1} \ldots u_d}_{\text{directions with smallest eigen values}}$

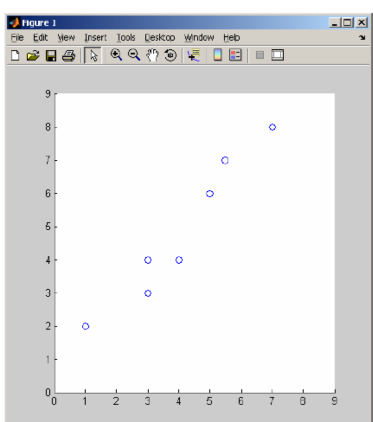
$\equiv$ keep $u_1 \ldots u_k$ to be eigen vectors of Σ with largest eigen values

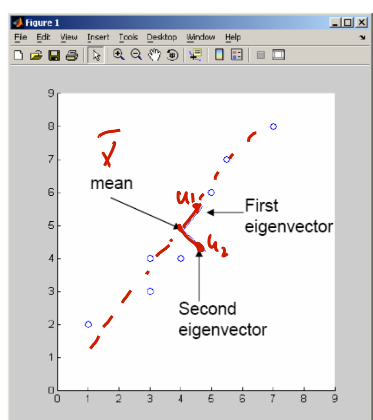
Basic PCA algoritm

- Start from ~~m~~ N by ~~n~~ d data matrix **X**
- **Recenter**: subtract mean from each row of **X**
  - $\mathbf{X}_c \leftarrow \mathbf{X} - \overline{\mathbf{X}}$ $\leftarrow$ $X_c = $ N, d, $x^i - \bar{x}$
- **Compute covariance matrix**:
  - $\Sigma \leftarrow 1/N\ \mathbf{X}_c^T\ \mathbf{X}_c$
- Find **eigen vectors and values** of Σ
- **Principal components:** k eigen vectors with highest eigen values

PCA example

$$\hat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$$

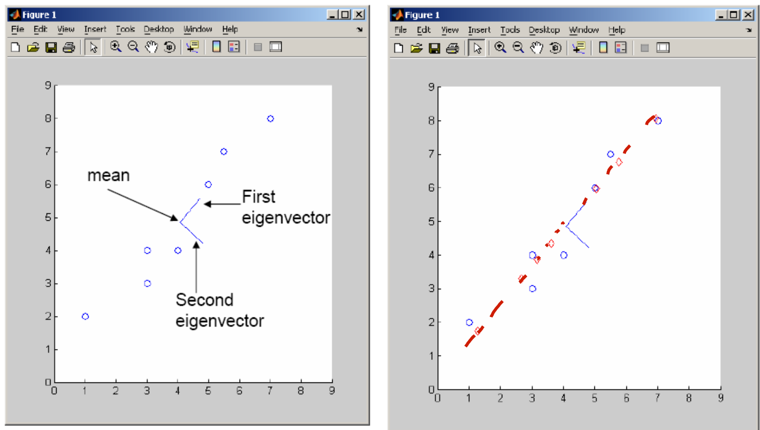




PCA example – reconstruction

$$\widehat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$$

only used first principal component

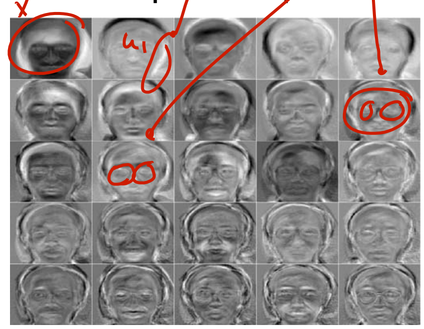


Eigenfaces [Turk, Pentland '91]

- Input images:

- Principal components:

glasses
long hair



Eigenfaces reconstruction

- Each image corresponds to adding 8 principal components:

k=0

Scaling up

- Covariance matrix can be really big!
  - Σ is d by d
  - Say, only 10000 features
  - finding eigenvectors is very slow…
- Use singular value decomposition (SVD)
  - finds to k eigenvectors of Σ by just looking at X_c
  - great implementations available, e.g., python, R, Matlab svd

SVD

- Write $\mathbf{X} = \mathbf{W}\,\mathbf{S}\,\mathbf{V}^T$
  - $\mathbf{X} \leftarrow$ data matrix, one row per datapoint
  - $\mathbf{W} \leftarrow$ weight matrix, one row per datapoint – coordinate of $\mathbf{x}^i$ in eigenspace
  - $\mathbf{S} \leftarrow$ singular value matrix, diagonal matrix
    - in our setting each entry is eigenvalue λ_j
  - $\mathbf{V}^T \leftarrow$ singular vector matrix
    - in our setting each row is eigenvector $\mathbf{v}_j$

PCA using SVD algoritm

- Start from m by n data matrix **X**
- **Recenter**: subtract mean from each row of **X**
  - $\mathbf{X}_c \leftarrow \mathbf{X} - \overline{\mathbf{X}}$
- Call SVD algorithm on $\mathbf{X}_c$ – ask for k singular vectors
- **Principal components:** k singular vectors with highest singular values (rows of $\mathbf{V}^T$)
  - **Coefficients** become:

What you need to know

- Dimensionality reduction
  - why and when it's important
- Simple feature selection
- Principal component analysis
  - minimizing reconstruction error
  - relationship to covariance matrix and eigenvectors
  - using SVD