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* 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:

$$\begin{aligned}\ell(\theta : \mathcal{D}) &= \log \prod_{j=1}^{m} P(\mathbf{x}_j \mid \theta) \\ &= \sum_{j=1}^{m} \log P(\mathbf{x}_j \mid \theta) \\ &= \sum_{j=1}^{m} \log \sum_{\mathbf{z}} P(\mathbf{x}_j, \mathbf{z} \mid \theta)\end{aligned}$$

E-step

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

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

Jensen's inequality

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

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

Applying Jensen's inequality

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

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

The M-step maximizes lower bound on weighted data

- Lower bound from Jensen's:

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

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

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

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

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

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 DOES
- EM is coordinate ascent
- General case for EM

Dimensionality Reduction PCA

Machine Learning – CSE4546

Carlos Guestrin

University of Washington

November 13, 2014

Dimensionality reduction

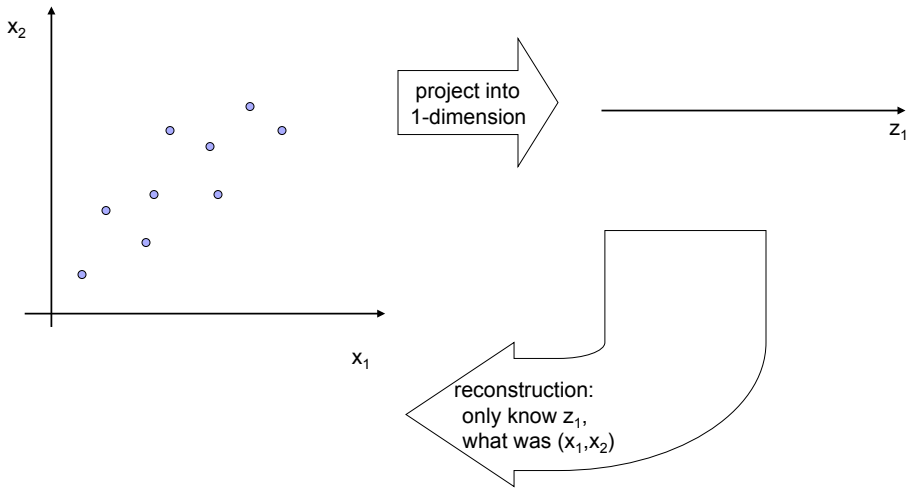
- Input data may have thousands or millions of dimensions!
  - e.g., text data has
- **Dimensionality reduction**: represent data with fewer dimensions
  - easier learning – fewer parameters
  - visualization – hard to visualize more than 3D or 4D
  - discover "intrinsic dimensionality" of data
    -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

- 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

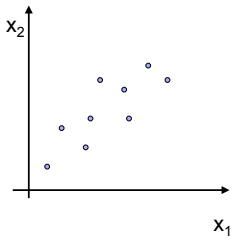
Linear projections, a review

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

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: $\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}^i$ and $z_j^i = (\mathbf{x}^i - \bar{\mathbf{x}}) \cdot \mathbf{u}_j$
- PCA:
  - Given k<<d, find $(\mathbf{u}_1,\ldots,\mathbf{u}_k)$ minimizing reconstruction error:

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



Understanding the reconstruction error

- 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:

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

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

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

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

Reconstruction error and covariance matrix

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

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

Minimizing reconstruction error and eigen vectors

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

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

- Eigen vector:

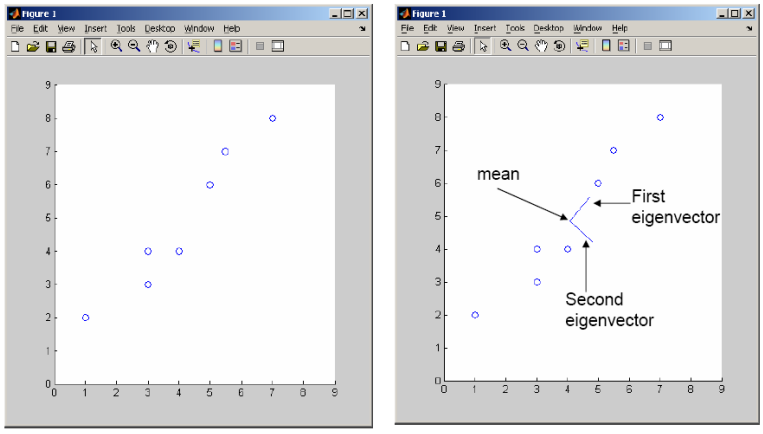
- Minimizing reconstruction error equivalent to picking ($\mathbf{u}_{k+1},\ldots,\mathbf{u}_d$) to be eigen vectors with smallest eigen values

Basic PCA algoritm

- Start from m by n data matrix **X**
- **Recenter**: subtract mean from each row of **X**
  - $\mathbf{X}_c \leftarrow \mathbf{X} - \overline{\mathbf{X}}$
- **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

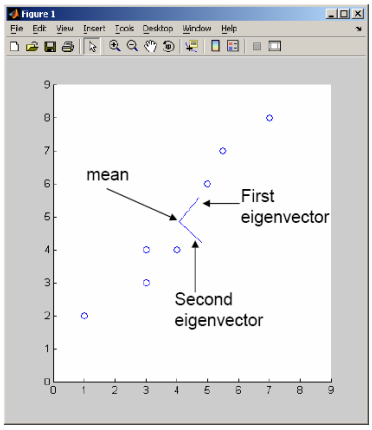
$$\hat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$$

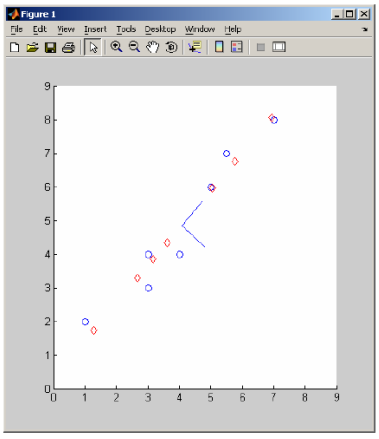


PCA example – reconstruction

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

only used first principal component

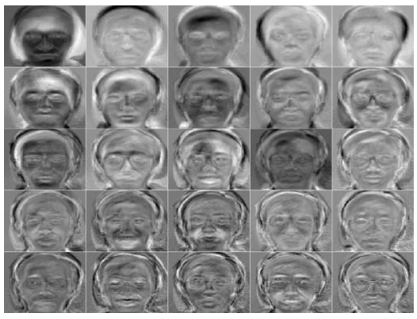




Eigenfaces [Turk, Pentland '91]

- Input images:

- Principal components:



Eigenfaces reconstruction

- Each image corresponds to adding 8 principal components:

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
  - great implementations available, e.g., python, R, Matlab svd

SVD

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

PCA using SVD algoritm

- Start from m by n data matrix $\mathbf{X}$
- **Recenter**: subtract mean from each row of $\mathbf{X}$
  - $\mathbf{X}_c \leftarrow \mathbf{X} - \overline{\mathbf{X}}$
- 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