Expectation Maximization

iterative alg. for MLE

Machine Learning – CSE546

Emily Fox

University of Washington

November 6, 2013

Iterative Algorithm

- Motivates a coordinate ascent-like algorithm:
  1. Infer missing values z^i given estimate of parameters $\hat{\theta}$
  2. Optimize parameters to produce new $\hat{\theta}$ given "filled in" data z^i
  3. Repeat

$$z_i \underset{\text{max}}{\overset{\text{est.}}{\rightleftarrows}} \hat{\theta} \qquad \hat{\theta} = \{\hat{\phi}_k, \pi_k\}$$

- Example: MoG (derivation soon… + HW)
  1. Infer "responsibilities" ← prev. iter.

soft weights

$$r_{ik} = p(z^i = k \mid x^i, \hat{\theta}^{(t-1)}) = \frac{\pi_k^{(t-1)} p(x_i \mid \phi_k^{(t-1)})}{\sum_j \pi_j^{(t-1)} p(x_i \mid \phi_j^{(t-1)})}$$

  2. Optimize parameters

$$\max \text{ w.r.t. } \pi_k : \quad \hat{\pi}_k^{(t)} = \frac{1}{N}\sum_i r_{ik} = \frac{r_k}{N}$$

← soft counts!

$$\max \text{ w.r.t. } \mu_k, \Sigma_k :$$

$$\hat{\mu}_k^{(t)} = \frac{\sum r_{ik} x_i}{N}$$

← weighted mean

$$\hat{\Sigma}_k^{(t)} = \frac{1}{r_k} \sum r_{ik} x_i x_i^T - \hat{\mu}_k^{(t)} \hat{\mu}_k^{(t)T}$$

Expectation Maximization (EM) – Setup

- More broadly applicable than just to mixture models considered so far
- Model: x observable – *"incomplete" data* ↙ what we have
  introduce → y not (fully) observable – *"complete" data* ↙ what we wish we had
  θ parameters
- Interested in maximizing (wrt θ):

$$p(x \mid \theta) = \sum_y p(x, y \mid \theta) = \sum_y p(x \mid y, \theta) p(y \mid \theta)$$

- Special case: ← non-invertible, deterministic fcn

$$x = g(y)$$

e.g. $y = \begin{bmatrix} z \\ x \end{bmatrix}$ ← class labels ← obs. in standard mix. models

©Emily Fox 2013 3

Expectation Maximization (EM) – Derivation

- Step 1
  - Rewrite desired likelihood in terms of complete data terms

$p(y, x \mid \theta)$ ← b/c $x = g(y)$

$$p(y \mid \theta) = p(y \mid x, \theta) p(x \mid \theta)$$

quantity of interest

$$\Rightarrow \underbrace{\log p(x \mid \theta)}_{L_x(\theta)} = \log p(y \mid \theta) - \log p(y \mid x, \theta)$$

- Step 2
  - Assume estimate of parameters $\hat{\theta}$
  - Take expectation with respect to $p(y \mid x, \hat{\theta})$ "$E[\cdot \mid x, \hat{\theta}]$"

$$L_x(\theta) = \underbrace{E[\log p(y \mid \theta) \mid x, \hat{\theta}]}_{U(\theta, \hat{\theta})} + \underbrace{E[-\log p(y \mid x, \theta) \mid x, \hat{\theta}]}_{V(\theta, \hat{\theta})}$$

©Emily Fox 2013 4

Expectation Maximization (EM) – Derivation

- Step 3
  - Consider log likelihood of data at any θ relative to log likelihood at $\hat{\theta}$

$$L_x(\theta) - L_x(\hat{\theta}) = [U(\theta,\hat{\theta}) - U(\hat{\theta},\hat{\theta})] + [V(\theta,\hat{\theta}) - V(\hat{\theta},\hat{\theta})]$$

✓

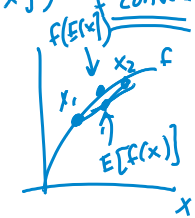
- **Aside: Gibbs Inequality** $E_p[\log p(x)] \geq E_p[\log q(x)]$

Proof: Use Jensen's Ineq. $E[f(x)] \leq f(E[x])$ for any f convex

Here:

$$E_p[\log q] - E_p[\log p] = E_p\left[\log \frac{q}{p}\right]$$

$$\leq \log E_p\left[\frac{q}{p}\right] = \log \int_x \cancel{p(x)} \frac{q(x)}{\cancel{p(x)}} dx = 0$$



Expectation Maximization (EM) – Derivation

$$L_x(\theta) - L_x(\hat{\theta}) = [U(\theta,\hat{\theta}) - U(\hat{\theta},\hat{\theta})] + \underbrace{[V(\theta,\hat{\theta}) - V(\hat{\theta},\hat{\theta})]}_{\geq 0}$$

- Step 4
  - Determine conditions under which log likelihood at θ exceeds that at $\hat{\theta}$

Using Gibbs inequality:

$$V(\theta,\hat{\theta}) = E[-\log p(y \mid x,\theta) \mid x,\hat{\theta}] \geq E[-\log p(y \mid x,\hat{\theta}) \mid x,\hat{\theta}]$$

$$= V(\hat{\theta},\hat{\theta}) \quad \forall \theta$$

If $U(\theta,\hat{\theta}) \geq U(\hat{\theta},\hat{\theta})$

Then

$$L_x(\theta) \geq L_x(\hat{\theta})$$

choosing θ s.t. this is true means we're moving in the right direction (or at least not wrong)

Motivates EM Algorithm

- Initial guess: $\hat{\theta}^{(0)}$
- Estimate at iteration *t*: $\hat{\theta}^{(t)}$

- **E-Step**

  Compute $U(\theta, \hat{\theta}^{(t)}) = E[\log p(y \mid \theta) \mid x, \hat{\theta}^{(t)}]$

- **M-Step**

  Compute $\hat{\theta}^{(t+1)} = \arg\max_{\theta} U(\theta, \hat{\theta}^{(t)})$

From before, $U(\hat{\theta}^{(t+1)}, \hat{\theta}^{(t)}) \geq U(\hat{\theta}^{(t)}, \hat{\theta}^{(t)})$

$\Rightarrow L_x(\hat{\theta}^{(t+1)}) \geq L_x(\hat{\theta}^{(t)})$

Example – Mixture Models

- **E-Step** Compute $U(\theta, \hat{\theta}^{(t)}) = E[\log p(y \mid \theta) \mid x, \hat{\theta}^{(t)}]$
- **M-Step** Compute $\hat{\theta}^{(t+1)} = \arg\max_{\theta} U(\theta, \hat{\theta}^{(t)})$

- Consider $y^i = \{z^i, x^i\}$ i.i.d.

$p(x^i, z^i \mid \theta) = \pi_{z^i} p(x^i \mid \phi_{z^i}) =$

$E_{q_t}[\log p(y \mid \theta)] = \sum_i E_{q_t}[\log p(x^i, z^i \mid \theta)] =$

Coordinate Ascent Behavior

- Bound log likelihood:

$$L_x(\theta) = U(\theta, \hat{\theta}^{(t)}) + V(\theta, \hat{\theta}^{(t)})$$
$$\geq$$
$$L_x(\hat{\theta}^{(t)}) = U(\hat{\theta}^{(t)}, \hat{\theta}^{(t)}) + V(\hat{\theta}^{(t)}, \hat{\theta}^{(t)})$$

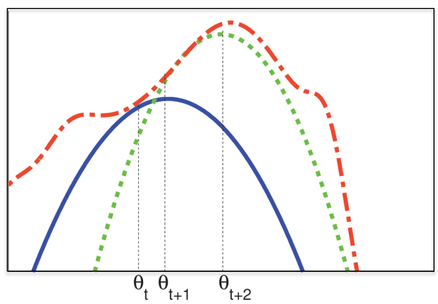


Figure from KM textbook

Comments on EM

- Since Gibbs inequality is satisfied with equality only if *p=q*, any step that changes θ should strictly **increase likelihood**
- In practice, can replace the **M-Step** with increasing *U* instead of maximizing it (**Generalized EM**)
- Under certain conditions (e.g., in exponential family), can show that EM **converges to a stationary point** of $L_x(\theta)$
- Often there is a **natural choice for *y*** … has physical meaning
- If you want to choose any *y*, not necessarily *x=g(y)*, replace $p(y \mid \theta)$ in *U* with $p(y, x \mid \theta)$

Initialization

- In mixture model case where $y^i = \{z^i, x^i\}$ there are many ways to initialize the EM algorithm
- Examples:
  - Choose K observations at random to define each cluster. Assign other observations to the nearest "centriod" to form initial parameter estimates
  - Pick the centers sequentially to provide good coverage of data
  - Grow mixture model by splitting (and sometimes removing) clusters until K clusters are formed
- Can be quite important to convergence rates in practice

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

Dimensionality Reduction
PCA

Machine Learning – CSE4546
Carlos Guestrin
University of Washington

November 13, 2013

Dimensionality reduction

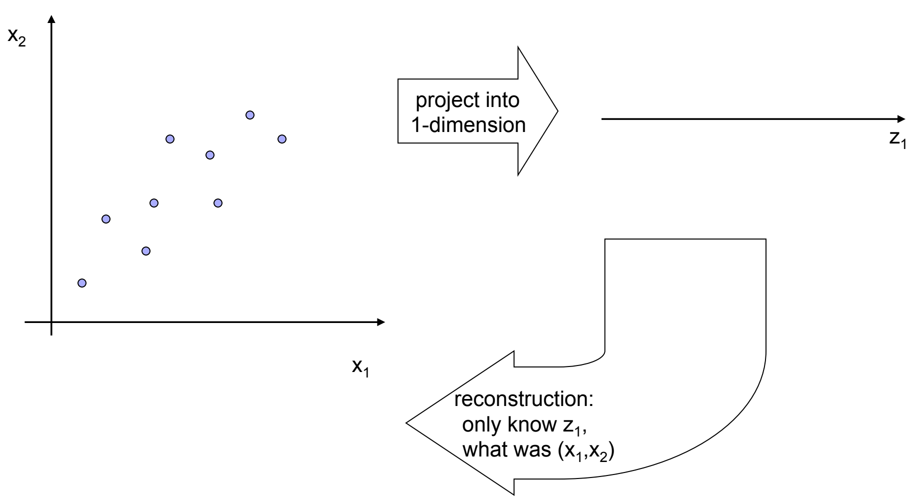
- Input data may have thousands or millions of dimensions!
  - e.g., text data has
- **Dimensionality reduction**: represent data with fewer dimensions
  - easier learning – fewer parameters
  - visualization – hard to visualize more than 3D or 4D
  - discover "intrinsic dimensionality" of data
    -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

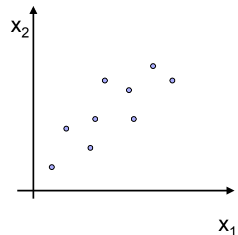
Linear projections, a review

- 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: $\bar{\mathbf{x}} = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \mathbf{x}^i$ 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:

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



Understanding the reconstruction 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:

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

- 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:

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:

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

- Eigen vector:

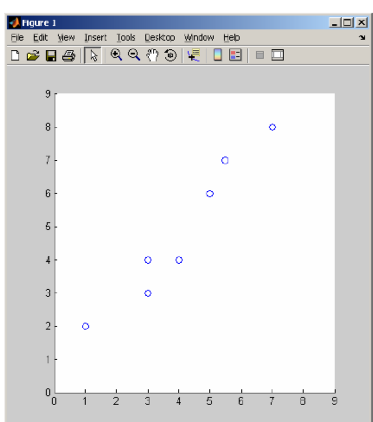
- Minimizing reconstruction error equivalent to picking ($\mathbf{u}_{k+1},\ldots,\mathbf{u}_d$) to be eigen vectors with smallest eigen values

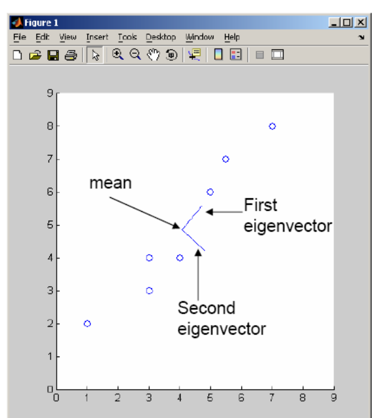
Basic PCA algoritm

- 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

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

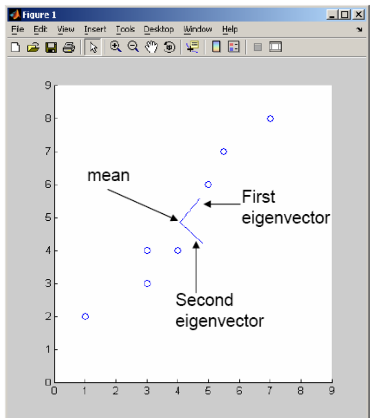


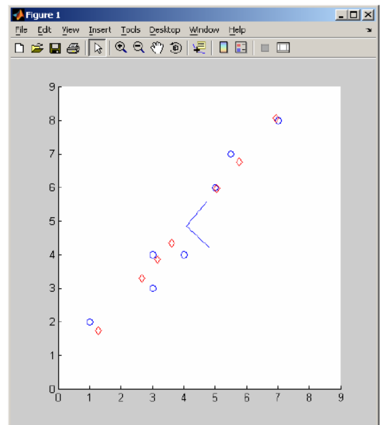


PCA example – reconstruction

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

only used first principal component

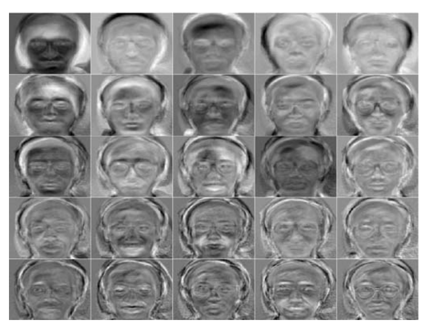




Eigenfaces [Turk, Pentland '91]

- Input images:

- Principal components:



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 $\mathbf{X} = \mathbf{W}\ \mathbf{S}\ \mathbf{V}^T$
  - $\mathbf{X} \leftarrow$ 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