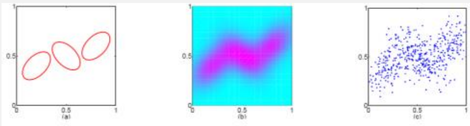
1

BTRY 4840/6840, CS 4775
Computational Genetics and Genomics



October 18, 2018

2

Final projects

- 1-page proposal due Oct 30 (a week from Tuesday)
- Find something you're excited about that is related to class topics
  - Finding overlap with other classes or your research is good
- Projects need to involve modeling, implementation, and data analysis
  - Only rare exceptions to the above; must get approval
- Graduate students expected to produce novel research, undergrads can re-implement a method
- Be ambitious but keep focused – have ~5 weeks
- In class presentations in weeks following proposal

3

Final projects, continued

- Final projects due December 11 by 4:30pm
  - Turn in: report, code, data (if lightweight)
- Report should be ~6-10 pages, but length is less important than the quality of the report
  - Use standard scientific paper format: Introduction, Methods, Results, Discussion
  - Figures are important: take time to make these clear and communicative, reference and explain these in the text
  - Put important excerpts from source code and/or data in Appendix
    - Keep this brief: raw data is less important than interpretation
- Remember: this is equivalent of 2½ problem sets

4

Today's lecture

- Learning: parameter estimation
  - Learning when some variables are latent (hidden) via Expectation-Maximization (EM)
    - Related approach: K-means clustering
    - Example: EM for Gaussian mixture model
  - Baum-Welch algorithm: EM for HMMs
  - Viterbi training

5

Expectation-Maximization

6

Recall: 2d Gaussian mixture model

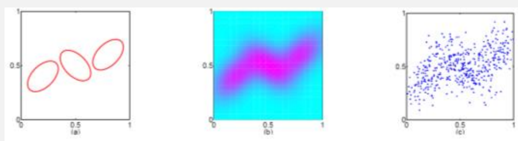


Figure 10.2: Illustration of a mixture of 3 Gaussians in a two-dimensional space showing (a) contours representing one standard deviation for each of the mixture components, (b) the marginal probability density of the mixture distribution, and (c) a sample of 500 points drawn from the marginal distribution.

- Parameters: means, covariances of 3 Gaussians
- Latent variables: which Gaussian each point from

Jordan (2003)

7

Recall: K-means clustering algorithm

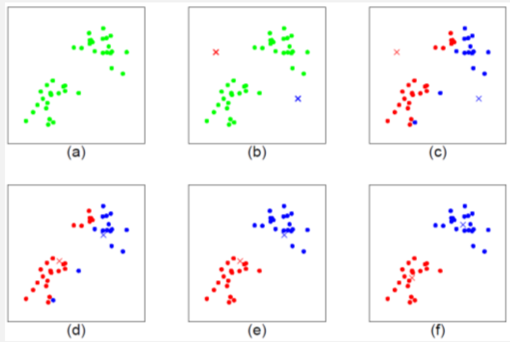
- Wish to estimate
  - Cluster means: $(\mu_1, \dots, \mu_K)$
  - Assignments of points to clusters: $(z_1, \dots, z_N)$
- Algorithm:
  1. Make initial guess of $(\mu_1, \dots, \mu_K)$
     - Can use K random data points or some other way to initialize
  2. Update assignments: [closest cluster via Euclidean distance]

     $$z_n = \arg\min_j \|x_n - \mu_j\|^2$$
  3. Update the cluster means:

     $$\mu_i = \frac{\sum_n x_n \cdot I[z_n = i]}{\sum_n I[z_n = i]}$$
  4. Iterate until convergence

8

Illustration of K-means



Jordan (2003)

9

Recall: More principled approach using statistics

- K-means assigns each point to exactly one cluster
  - So called "hard assignment"
  - Can also perform "soft assignment": probabilistic
    - Use probability that each data point is a member of each cluster
- Consider Z_n as latent variables, and x_n observed
- Seek to estimate parameters $(\pi_1, \dots, \pi_K, \mu_1, \dots, \mu_K)$
  - Can also estimate Σ_is
- Will use posterior probability of each x_n being in i:

$$\tau_n^i = p(Z_n = i | x_n, \boldsymbol{\theta}) = \frac{p(x_n | Z_n = i, \boldsymbol{\theta}_i) p(Z_n = i)}{p(x_n | \boldsymbol{\theta})}$$

$$= \frac{\pi_i \cdot \mathcal{N}(x_n | \mu_i, \Sigma_i)}{\sum_j \pi_j \cdot \mathcal{N}(x_n | \mu_j, \Sigma_j)}$$

10

Expectation-maximization for Gaussian mixtures

- Algorithm:
  1. Make initial guess of $\boldsymbol{\theta}^{(0)} = \left(\pi_1^{(0)}, \dots, \pi_K^{(0)}, \mu_1^{(0)}, \dots, \mu_K^{(0)}\right)$
     - Will take Σ_is as known, but these can also be included
  2. Update posterior probability of assignments $\tau_n^{i(t)}$ "E step"
     - Here t is the iteration number
     - Note: $\tau_n^{i(t)} = E\left[I[Z_n = i] | x_n, \boldsymbol{\theta}^{(t)}\right]$
       - Bernoulli RV: expectation is its probability (previous slide)
     - Equation for $\tau_n^{i(t)}$ relies on current parameters $\pi_i^{(t)}, \mu_i^{(t)}$
  3. Update the parameters: "M step"

     $$\mu_i^{(t+1)} = \frac{\sum_n x_n \cdot \tau_n^{i(t)}}{\sum_n \tau_n^{i(t)}} \qquad \pi_i^{(t+1)} = \frac{1}{N} \sum_{n=1}^{N} \tau_n^{i(t)}$$
     - Sets parameters to their MLE given hidden variable assignments
  4. Iterate until small difference in likelihood between iterations
     - Can prove that in EM, likelihood increases in each iteration

11

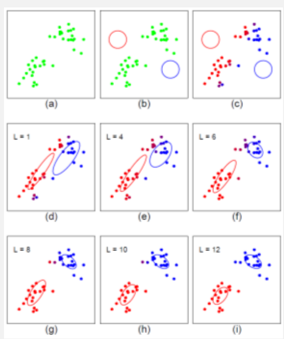
If covariance matrix unknown

- Can update covariance matrices Σ_i as

$$\Sigma_i^{(t+1)} = \frac{\sum_{n=1}^{N} \tau_n^{i(t)} \left(x_n - \mu_i^{(t+1)}\right)\left(x_n - \mu_i^{(t+1)}\right)^T}{\sum_{n=1}^{N} \tau_n^{i(t)}}$$

12

Example: $K = 2$, $L = \#$ iterations



Jordan (2003)

Recall: Desired uses of HMMs

- Evaluation:
  - **Given**: observed $\boldsymbol{x}$ and HMM specification
    Question: what is the joint probability of $\boldsymbol{x}$ and a given $\boldsymbol{z}$?
    Question: what is the likelihood of $\boldsymbol{x}$ based on the HMM?
- Decoding:
  - **Given**: observed $\boldsymbol{x}$ and HMM
    Question: what sequence of hidden states produced $\boldsymbol{x}$?
  - Viterbi decoding: most likely hidden state sequence
  - Posterior probability of hidden states: probability of each state z_i producing each x_i
- Learning:
  - **Given**: observed $\boldsymbol{x}$ and HMM without complete probabilities
    Question: what emission, transition probabilities produced $\boldsymbol{x}$?

Hidden Markov models

How do we learn transition, emission probabilities from **labeled** data?

Supervised learning (Review)

Recall: Supervised learning MLEs

- Determining $\boldsymbol{\theta}$ from labeled data: fairly simple
- Given $x_1, x_2, \dots, x_N$ with correct $z_1, z_2, \dots, z_N$
- Let

  $A_{kl} = \#$ times transition $z_i = k \rightarrow z_{i+1} = l$ occus for all i

  $E_k(b) = \#$ times $z_i = k$ emits b
- Can show that MLE for the parameters are

  $$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}} \text{ and } e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')}$$
- In practice often use pseudocounts (priors)
  - Avoids 0 probabilities and can aid in overfitting small datasets

Hidden Markov models

How do we learn transition, emission probabilities from **unlabeled** data?

Unsupervised learning: Baum-Welch

Baum-Welch algorithm overview

1. Guess at the values of $\boldsymbol{\theta}^{(0)} = \left(a_{kl}^{(0)}, e_k^{(0)}(b)\right)$
2. Estimate $A_{kl}^{(t)}, E_k^{(t)}(b)$ counts considering all paths
   - Sum the probabilities of all possible transitions / emissions
     - Not using fixed counts of labeled sequences
   - **Are expected values** and generally real valued (not integers)
   - The "E step"
3. Update $a_{kl}^{(t+1)}, e_k^{(t+1)}(b)$ using $A_{kl}, E_k(b)$
   - The "M step"
   - Uses MLE – identical to supervised learning for this
4. Repeat

Calculating $A_{kl}^{(t)}, E_k^{(t)}(b)$ by expectation

- Can set expected counts of transitioning using every step i as:

  $$A_{kl}^{(t)} = E\left[A_{kl} | \boldsymbol{x}, \boldsymbol{\theta}^{(t)}\right] = \sum_i P\left(z_i = k, z_{i+1} = l | \boldsymbol{x}, \boldsymbol{\theta}^{(t)}\right)$$
  $$= \sum_i \frac{f_k^{(t)}(i) a_{kl}^{(t)} e_l^{(t)}(x_{i+1}) b_l^{(t)}(i+1)}{P(\boldsymbol{x}|\boldsymbol{\theta}^{(t)})}$$
  - Here $f_k^{(t)}(i)$ is forward probability using $\boldsymbol{\theta}^{(t)}$ as parameters
    - All other values with superscript (t) are also those using $\boldsymbol{\theta}^{(t)}$
- Similarly have

  $$E_k^{(t)}(b) = \frac{1}{P(\boldsymbol{x}|\boldsymbol{\theta}^{(t)})} \sum_i I[x_i = b] \cdot f_k^{(t)}(i) \cdot b_k^{(t)}(i)$$

Transition probability conditional on data $\boldsymbol{x}$, current parameters $\boldsymbol{\theta}^{(t)}$

19

- Consider the probability of transitioning from k to l given $\boldsymbol{x}$:

$P(z_i = k, z_{i+1} = l|\boldsymbol{x}, \boldsymbol{\theta}^{(t)}) = \frac{1}{P(\boldsymbol{x}|\boldsymbol{\theta}^{(t)})} \cdot P(z_i = k, z_{i+1} = l, \boldsymbol{x}|\boldsymbol{\theta}^{(t)})$

$= \frac{1}{P(\boldsymbol{x}|\boldsymbol{\theta}^{(t)})} P(x_1, \ldots, x_i, z_i = k, z_{i+1} = l, x_{i+1}, \ldots, x_N|\boldsymbol{\theta}^{(t)})$

$= \frac{1}{P(\boldsymbol{x}|\boldsymbol{\theta}^{(t)})} P(z_{i+1} = l, x_{i+1}, \ldots, x_N|z_i = k, \boldsymbol{\theta}^{(t)}) P(x_1, \ldots, x_i, z_i = k|\boldsymbol{\theta}^{(t)})$

$= \frac{1}{P(\boldsymbol{x}|\boldsymbol{\theta}^{(t)})} P(z_{i+1} = l, x_{i+1}, \ldots, x_N|z_i = k, \boldsymbol{\theta}^{(t)}) f_k^{(t)}(i)$

$= \frac{1}{P(\boldsymbol{x}|\boldsymbol{\theta}^{(t)})} P(x_{i+2}, \ldots, x_N|z_{i+1} = l, \boldsymbol{\theta}^{(t)}) P(x_{i+1}|z_{i+1} = l, \boldsymbol{\theta}^{t}) P(z_{i+1} = l|z_i = k, \boldsymbol{\theta}^{t}) f_k^{(t)}(i)$

$= \frac{1}{P(\boldsymbol{x}|\boldsymbol{\theta}^{(t)})} f_k^{(t)}(i) a_{kl}^{(t)} e_l^{(t)}(x_{i+1}) b_l^{(t)}(i+1)$

So $P(z_i = k, z_{i+1} = l|\boldsymbol{x}, \boldsymbol{\theta}^{(t)}) = \frac{1}{P(\boldsymbol{x}|\boldsymbol{\theta})} f_k^{(t)}(i) a_{kl}^{(t)} e_l^{(t)}(x_{i+1}) b_l^{(t)}(i+1)$

- Why does this lead to different $a_{kl}^{(t+1)}$? Influenced by data $\boldsymbol{x}$

Can sum over multiple training sequences when available

20

- Can set expected counts of transitioning using every step i and all sequences $\boldsymbol{x}$ as:

$$A_{kl}^{(t)} = \sum_{\boldsymbol{x}} \sum_i \frac{f_k^{(t)}(i) a_{kl}^{(t)} e_l^{(t)}(x_{i+1}) b_l^{(t)}(i+1)}{P(\boldsymbol{x}|\boldsymbol{\theta}^{(t)})}$$

- Similarly have

$$E_k^{(t)}(b) = \sum_{\boldsymbol{x}} \frac{1}{P(\boldsymbol{x}|\boldsymbol{\theta}^{(t)})} \sum_i I[x_i = b] \cdot f_k^{(t)}(i) \cdot b_k^{(t)}(i)$$

Baum-Welch algorithm: EM for HMMs

21

1. Initialize $\boldsymbol{\theta}^{(0)} = \left(a_{kl}^{(0)}, e_k^{(0)}(b)\right)$ for all k, l, b
2. For $t = 0$ until end:
   1. Run forward algorithm: compute $f_k^{(t)}(i)$ for all k, i using $\boldsymbol{\theta}^{(t)}$
   2. Run backward algorithm: compute $b_k^{(t)}(i)$ for all k, i using $\boldsymbol{\theta}^{(t)}$
   3. Calculate $A_{kl}^{(t)}, E_k^{(t)}(b)$ using earlier equations "E step"
   4. Calculate new $\boldsymbol{\theta}^{(t+1)} = (a_{kl}^{(t+1)}, e_k^{(t+1)}(b))$ for all k, l, b as:

      $a_{kl}^{(t+1)} = \frac{A_{kl}^{(t)}}{\sum_{l'} A_{kl'}^{(t)}}$ and $e_k^{(t+1)}(b) = \frac{E_k^{(t)}(b)}{\sum_{b'} E_k^{(t)}(b')}$ "M step"
   5. Get updated likelihood $P(\boldsymbol{x}|\boldsymbol{\theta}^{(t+1)})$ ← Guaranteed increased
3. Terminate when $\Delta P(\boldsymbol{x}|\boldsymbol{\theta})$ small or fixed # iterations

Notes on Baum-Welch

22

- Runtime complexity:
  - # iterations × $O(K^2N)$
- Will always increase $P(\boldsymbol{x}|\boldsymbol{\theta})$
- But may not find global maximum
  - Gives local optimum
  - Initial assignment can help guide solution to global maximum, but generally no way to guarantee this
- If large # of parameters $\boldsymbol{\theta}$, can overfit / overtrain
  - As we've talked about, overfitting ⇒ too heavily tuned to data

Recall: Randomized training algorithm

23

1. Initialize $\boldsymbol{\theta}^{(0)} = \left(a_{kl}^{(0)}, e_k^{(0)}(b)\right)$ for all k, l, b
2. Iterate:
   1. Run forward algorithm: compute $f_k^{(t)}(i)$for all k, i
   2. Sample n hidden state vectors $\boldsymbol{z}$ from $P(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}^{(t)})$
   3. Calculate $A_{kl}^{(t)}, E_k^{(t)}(b)$ using all paths $\boldsymbol{z}$ [as in supervised]
   4. Calculate new $\boldsymbol{\theta}^{(t+1)} = (a_{kl}^{(t+1)}, e_k^{(t+1)}(b))$ for all k, l, b [MLE]
3. Terminate when $\Delta P(\boldsymbol{x}|\boldsymbol{\theta})$ small or fixed # iterations

- Notes:
  - May not converge as quickly as EM: randomized
  - But stochastic so may yield higher $P(\boldsymbol{x}|\boldsymbol{\theta})$

Viterbi training

24

Algorithm

1. Initialize $\boldsymbol{\theta}^{(0)} = (a_{kl}^{(0)}, e_k^{(0)}(b))$ for all k, l, b [same]
2. Iterate *until convergence*:
   1. Run Viterbi decoding to get $\boldsymbol{z}^* = \arg\max_{\boldsymbol{z}} P(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta}^{(t)})$
   2. Calculate $A_{kl}^{(t)}, E_k^{(t)}(b)$ based on $\boldsymbol{z}^*$ [as in supervised]
   3. Calculate new $\boldsymbol{\theta}^{(t+1)} = (a_{kl}^{(t+1)}, e_k^{(t+1)}(b))$ for all k, l, b [MLE]

- Here convergence is guaranteed
  - Number of paths is discrete, will eventually be unchanged
- Does **not** maximize $P(\boldsymbol{x}|\boldsymbol{\theta})$ in general
  - Does not perform as well as Baum-Welch for most problems

Final notes

- Summary:
  - Expectation-maximization (EM)
    - Gaussian mixture example
  - Baum-Welch algorithm: EM for HMMs
  - Viterbi training