School of Computer Science
Carnegie Mellon

10-601B Introduction to Machine Learning

Expectation-Maximization (EM)

Readings:

Matt Gormley
Lecture 24
November 21, 2016

Reminders

- Final Exam
  - in-class Wed., Dec. 7

Outline

- **Models:**
  - Gaussian Naïve Bayes (GNB)
  - Mixture Model (MM)
  - Gaussian Mixture Model (GMM)
  - Gaussian Discriminant Analysis
- **Hard Expectation-Maximization (EM)**
  - Hard EM Algorithm
  - Example: Mixture Model
  - Example: Gaussian Mixture Model
  - K-Means as Hard EM
- **(Soft) Expectation-Maximization (EM)**
  - Soft EM Algorithm
  - Example: Gaussian Mixture Model
- **Extra Slides:** Why Does EM Work?
- **Properties of EM**
  - Nonconvexity / Local Optimization
  - Example: Grammar Induction
  - Variants of EM

GAUSSIAN MIXTURE MODEL

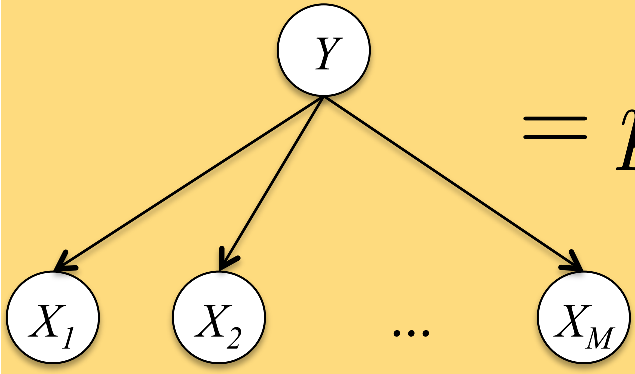
Recall...

Model 3: Gaussian Naïve Bayes

Data: $\mathcal{D} = \{\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\}_{n=1}^{N}$

Model: Product of **prior** and the event model

$$p(\boldsymbol{x}, y) = p(x_1, \ldots, x_K, y)$$



$$= p(y) \prod_{k=1}^{K} p(x_k | y)$$

Gaussian Naive Bayes assumes that $p(x_k|y)$ is given by a Normal distribution.

Mixture-Model

Data: $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^M$

Generative Story:

$$z \sim \text{Multinomial}(\boldsymbol{\phi})$$

$$\mathbf{x} \sim p_{\boldsymbol{\theta}}(\cdot|z)$$

Model:

Joint: $$p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}, z) = p_{\boldsymbol{\theta}}(\mathbf{x}|z)p_{\boldsymbol{\phi}}(z)$$

Marginal: $$p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) = \sum_{z=1}^{K} p_{\boldsymbol{\theta}}(\mathbf{x}|z)p_{\boldsymbol{\phi}}(z)$$

(Marginal) Log-likelihood:

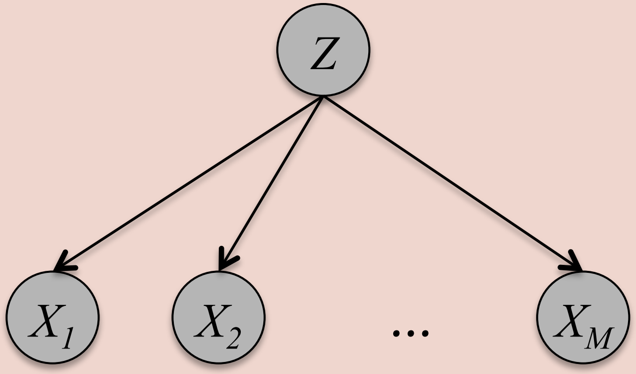
$$\ell(\boldsymbol{\theta}) = \log \prod_{i=1}^{N} p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}^{(i)})$$

$$= \sum_{i=1}^{N} \log \sum_{z=1}^{K} p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|z)p_{\boldsymbol{\phi}}(z)$$

Learning a Mixture Model

Supervised Learning: The parameters decouple!

$$\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{z}^{(i)})\}_{i=1}^{N}$$



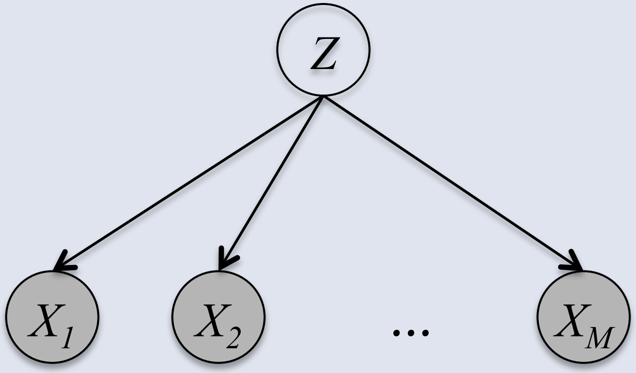
$$\boldsymbol{\theta}^*, \boldsymbol{\phi}^* = \underset{\boldsymbol{\theta},\boldsymbol{\phi}}{\text{argmax}} \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|z^{(i)}) p_{\boldsymbol{\phi}}(z^{(i)})$$

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmax}} \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|z^{(i)})$$

$$\boldsymbol{\phi}^* = \underset{\boldsymbol{\theta}}{\text{argmax}} \sum_{i=1}^{N} \log p_{\boldsymbol{\phi}}(z^{(i)})$$

Unsupervised Learning: Parameters are coupled by marginalization.

$$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$$



$$\boldsymbol{\theta}^*, \boldsymbol{\phi}^* = \underset{\boldsymbol{\theta},\boldsymbol{\phi}}{\text{argmax}} \sum_{i=1}^{N} \log \sum_{z=1}^{K} p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|z) p_{\boldsymbol{\phi}}(z)$$

Learning a Mixture Model

Supervised Learning: The parameters decouple!

$$\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{z}^{(i)})\}_{i=1}^{N}$$

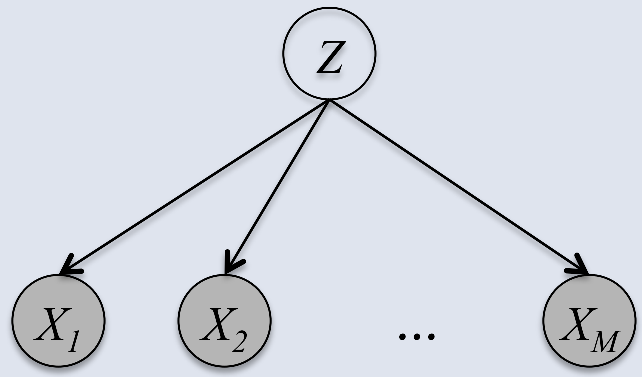
Training certainly isn't as simple as the supervised case.

In many cases, we could still use some black-box optimization method (e.g. Newton-Raphson) to solve this *coupled* optimization problem.

This lecture is about an even simpler method: EM.

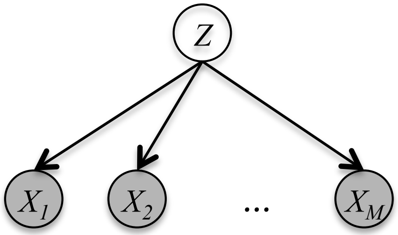
Unsupervised Learning: Parameters are coupled by marginalization.

$$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$$



$$\boldsymbol{\theta}^*, \boldsymbol{\phi}^* = \underset{\boldsymbol{\theta},\boldsymbol{\phi}}{\text{argmax}} \sum_{i=1}^{N} \log \sum_{z=1}^{K} p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|z) p_{\boldsymbol{\phi}}(z)$$

Mixture-Model



Data: $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ where $\mathbf{x}^{(i)} \in \mathbb{R}^M$

Generative Story:

$$z \sim \text{Multinomial}(\boldsymbol{\phi})$$

$$\mathbf{x} \sim p_{\boldsymbol{\theta}}(\cdot | z)$$

Model:

Joint: $$p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}, z) = p_{\boldsymbol{\theta}}(\mathbf{x}|z) p_{\boldsymbol{\phi}}(z)$$

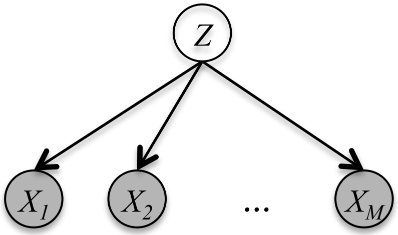
Marginal: $$p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) = \sum_{z=1}^{K} p_{\boldsymbol{\theta}}(\mathbf{x}|z) p_{\boldsymbol{\phi}}(z)$$

(Marginal) Log-likelihood:

$$\ell(\boldsymbol{\theta}) = \log \prod_{i=1}^{N} p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}^{(i)})$$

$$= \sum_{i=1}^{N} \log \sum_{z=1}^{K} p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|z) p_{\boldsymbol{\phi}}(z)$$

Gaussian Mixture-Model



Data: $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^M$

Generative Story: $z \sim \text{Categorical}(\phi)$

$\mathbf{x} \sim \text{Gaussian}(\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$

Model:

Joint: $p(\mathbf{x}, z; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = p(\mathbf{x}|z; \boldsymbol{\mu}, \boldsymbol{\Sigma})p(z; \boldsymbol{\phi})$

Marginal: $p(\mathbf{x}; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{z=1}^{K} p(\mathbf{x}|z; \boldsymbol{\mu}, \boldsymbol{\Sigma})p(z; \boldsymbol{\phi})$

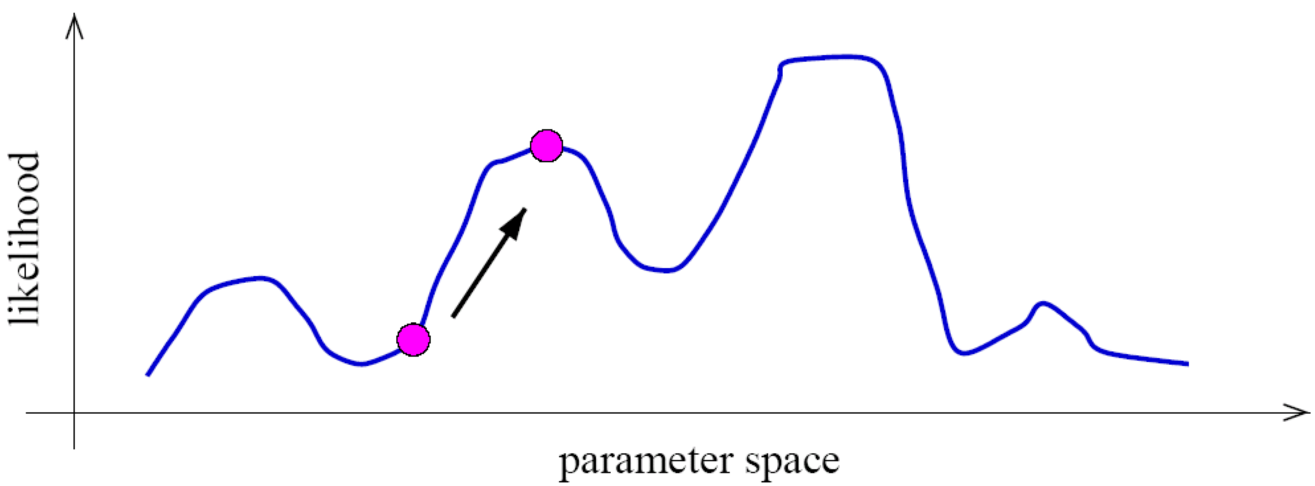
(Marginal) Log-likelihood:

$$\ell(\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \log \prod_{i=1}^{N} p(\mathbf{x}^{(i)}; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$= \sum_{i=1}^{N} \log \sum_{z=1}^{K} p(\mathbf{x}^{(i)}|z; \boldsymbol{\mu}, \boldsymbol{\Sigma})p(z; \boldsymbol{\phi})$$

Identifiability

- A mixture model induces a multi-modal likelihood.
- Hence gradient ascent can only find a local maximum.
- Mixture models are unidentifiable, since we can always switch the hidden labels without affecting the likelihood.
- Hence we should be careful in trying to interpret the "meaning" of latent variables.



aka. Viterbi EM

HARD EM

K-means as Hard EM

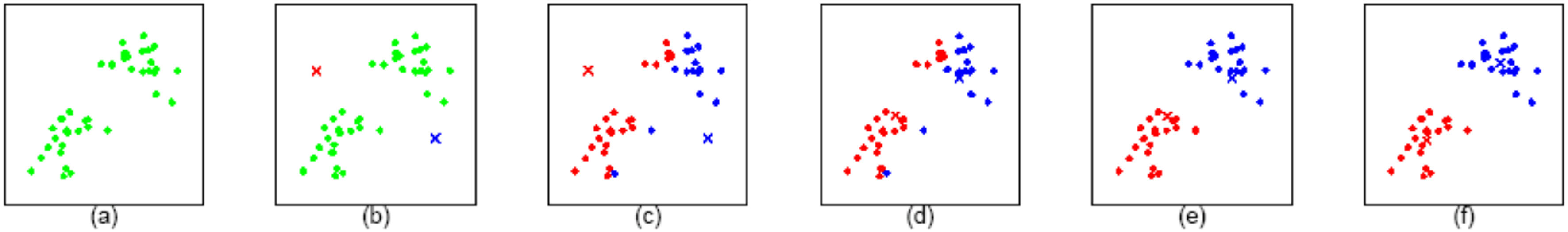
Loop:

- For each point n=1 to N,
  compute its cluster label:

$$z_n^{(t)} = \arg\max_k (x_n - \mu_k^{(t)})^T \Sigma_k^{-1(t)} (x_n - \mu_k^{(t)})$$

- For each cluster k=1:K

$$\mu_k^{(t+1)} = \frac{\sum_n \delta(z_n^{(t)}, k) x_n}{\sum_n \delta(z_n^{(t)}, k)} \qquad \Sigma_k^{(t+1)} = \ldots$$



Whiteboard

- Background: Coordinate Descent algorithm

Hard Expectation-Maximization

- Initialize **parameters** **randomly**
- **while** not converged
  1. **E-Step:**
     Set the **latent variables** to the the values that maximizes likelihood, treating parameters as observed
  2. **M-Step:**
     Set the **parameters** to the values that maximizes likelihood, treating latent variables as observed

← Hallucinate some data

← Standard Bayes Net training

Hard EM for Mixture Models

Algorithm 1 Hard EM for MMs

1: **procedure** HARDEM($\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$)
2: Randomly initialize parameters, $\boldsymbol{\theta}, \phi$
3: **while** not converged **do**
4: E-Step:

$$z^{(i)} \leftarrow \underset{z}{\operatorname{argmax}} \log p(\mathbf{x}^{(i)}|z; \boldsymbol{\theta}) + \log p(z; \boldsymbol{\phi})$$

5: M-Step:

$$\phi \leftarrow \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \sum_{i=1}^N \log p(z^{(i)}; \boldsymbol{\phi})$$

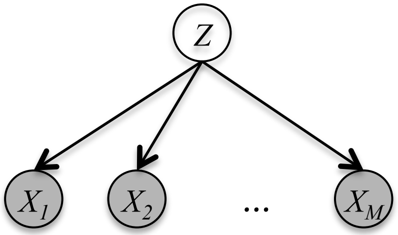
$$\boldsymbol{\theta} \leftarrow \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{i=1}^N \log p(\mathbf{x}^{(i)}|z; \boldsymbol{\theta})$$

6: **return** $(\phi, \boldsymbol{\theta})$

Implementation: For loop over possible values of latent variable

Implementation: supervised Bayesian Network learning

Gaussian Mixture-Model



Data: $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^M$

Generative Story:
$$z \sim \text{Categorical}(\phi)$$
$$\mathbf{x} \sim \text{Gaussian}(\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$$

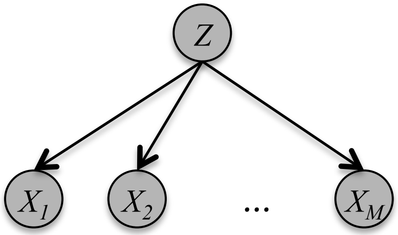
Model:

Joint: $$p(\mathbf{x}, z; \phi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = p(\mathbf{x}|z; \boldsymbol{\mu}, \boldsymbol{\Sigma})p(z; \phi)$$

Marginal: $$p(\mathbf{x}; \phi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{z=1}^{K} p(\mathbf{x}|z; \boldsymbol{\mu}, \boldsymbol{\Sigma})p(z; \phi)$$

(Marginal) Log-likelihood:
$$\ell(\phi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \log \prod_{i=1}^{N} p(\mathbf{x}^{(i)}; \phi, \boldsymbol{\mu}, \boldsymbol{\Sigma})$$
$$= \sum_{i=1}^{N} \log \sum_{z=1}^{K} p(\mathbf{x}^{(i)}|z; \boldsymbol{\mu}, \boldsymbol{\Sigma})p(z; \phi)$$

Gaussian Discriminant Analysis



Data: $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{z}^{(i)})\}_{i=1}^{N}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^M$ and $z^{(i)} \in \{1, \ldots, K\}$

Generative Story:

$$z \sim \text{Categorical}(\phi)$$

$$\mathbf{x} \sim \text{Gaussian}(\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$$

Model:

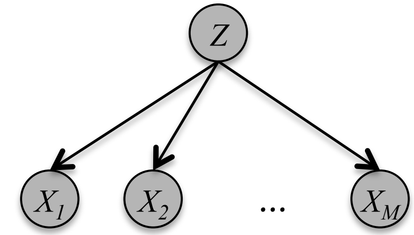
Joint: $p(\mathbf{x}, z; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = p(\mathbf{x}|z; \boldsymbol{\mu}, \boldsymbol{\Sigma})p(z; \boldsymbol{\phi})$

Log-likelihood:

$$\ell(\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \log \prod_{i=1}^{N} p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$= \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)}|z^{(i)}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \log p(z^{(i)}; \boldsymbol{\phi})$$

Gaussian Discriminant Analysis



Data: $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{z}^{(i)})\}_{i=1}^{N}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^M$ and $z^{(i)} \in \{1, \ldots, K\}$

Log-likelihood:

$$\ell(\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)} | z^{(i)}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \log p(z^{(i)}; \boldsymbol{\phi})$$

Maximum Likelihood Estimates:

Take the derivative of the Lagrangian, set it equal to zero and solve.

$$\phi_k = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(z^{(i)} = k), \forall k$$

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{N} \mathbb{I}(z^{(i)} = k)\mathbf{x}^{(i)}}{\sum_{i=1}^{N} \mathbb{I}(z^{(i)} = k)}, \forall k$$

$$\boldsymbol{\Sigma}_k = \frac{\sum_{i=1}^{N} \mathbb{I}(z^{(i)} = k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^T}{\sum_{i=1}^{N} \mathbb{I}(z^{(i)} = k)}, \forall k$$

Implementation:
Just counting

Hard EM for GMMs

Algorithm 1 Hard EM for GMMs

procedure HardEM($\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$)
    Randomly initialize parameters, $\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$
    while not converged **do**
        E-Step:

$$z^{(i)} \leftarrow \underset{z}{\operatorname{argmax}} \log p(\mathbf{x}^{(i)} | z; \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \log p(z; \boldsymbol{\phi})$$

        M-Step:

$$\phi_k \leftarrow \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(z^{(i)} = k), \forall k$$

$$\boldsymbol{\mu}_k \leftarrow \frac{\sum_{i=1}^{N} \mathbb{I}(z^{(i)} = k)\mathbf{x}^{(i)}}{\sum_{i=1}^{N} \mathbb{I}(z^{(i)} = k)}, \forall k$$

$$\boldsymbol{\Sigma}_k \leftarrow \frac{\sum_{i=1}^{N} \mathbb{I}(z^{(i)} = k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^T}{\sum_{i=1}^{N} \mathbb{I}(z^{(i)} = k)}, \forall k$$

    return $(\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$

Implementation: For loop over possible values of latent variable

Implementation: Just counting as in supervised setting

K-means as Hard EM

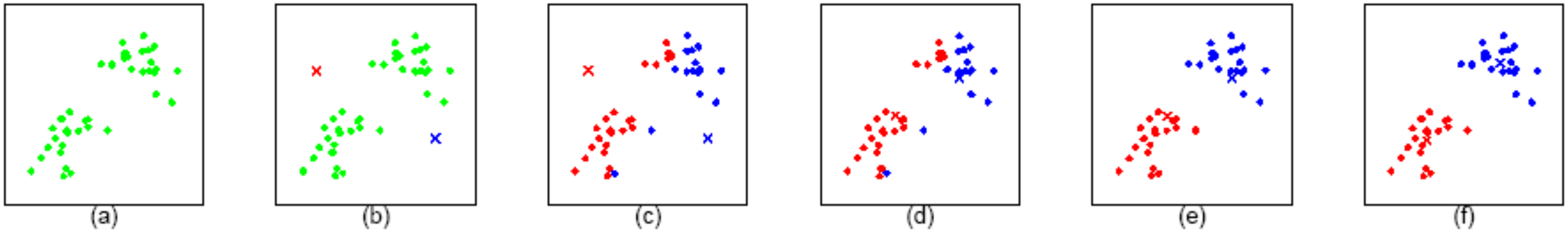
Loop:

- For each point n=1 to N, compute its cluster label:

$$z_n^{(t)} = \arg\max_k (x_n - \mu_k^{(t)})^T \Sigma_k^{-1(t)} (x_n - \mu_k^{(t)})$$

- For each cluster k=1:K

$$\mu_k^{(t+1)} = \frac{\sum_n \delta(z_n^{(t)}, k) x_n}{\sum_n \delta(z_n^{(t)}, k)} \qquad \Sigma_k^{(t+1)} = \ldots$$

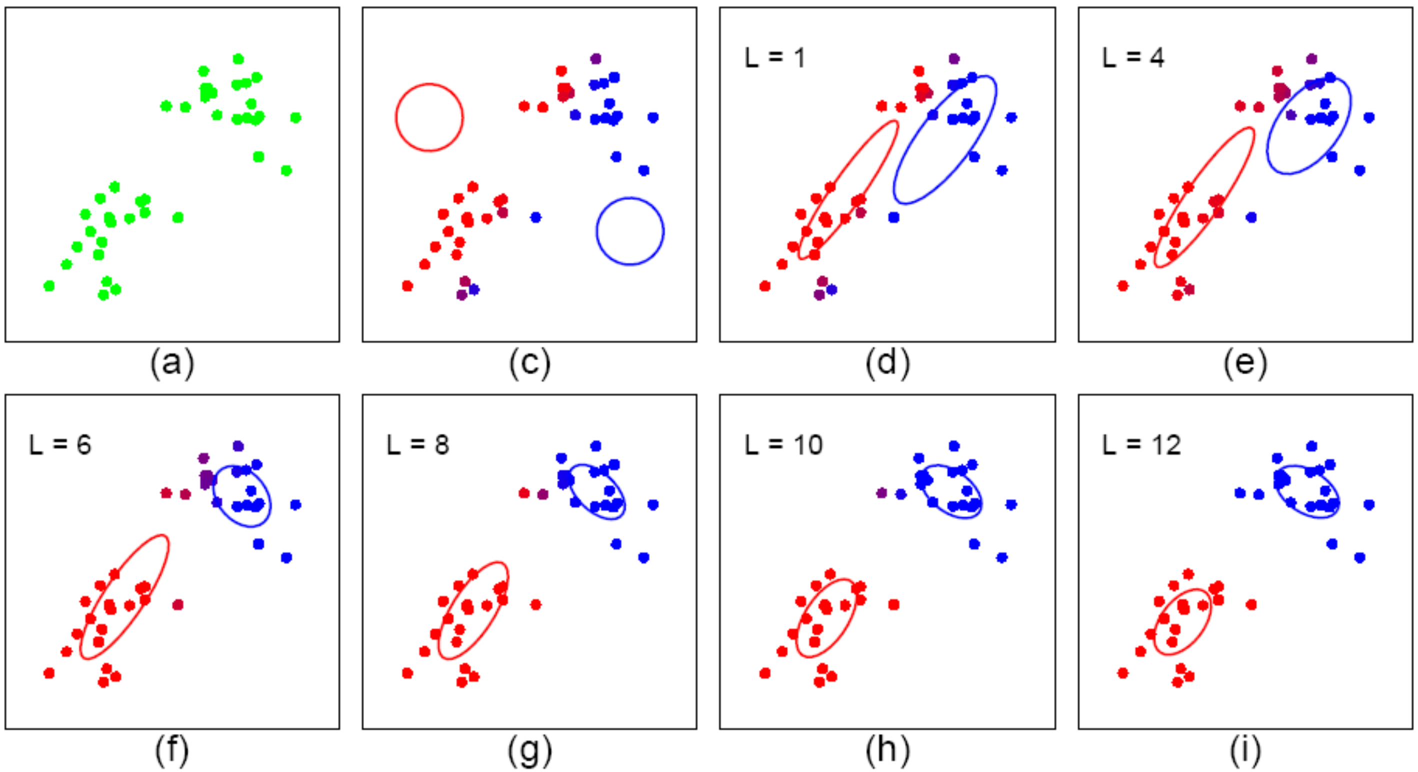


The standard EM algorithm

(SOFT) EM

(Soft) EM for GMM

- Start:
  - "Guess" the centroid μ_k and coveriance Σ_k of each of the K clusters
- Loop:



(Soft) Expectation-Maximization

- Initialize **parameters randomly**
- **while** not converged
  1. **E-Step:**
     Create one training example for each possible value of the **latent variables**
     Weight each example according to model's confidence
     Treat parameters as observed
  2. **M-Step:**
     Set the **parameters** to the values that maximizes likelihood
     Treat pseudo-counts from above as observed

← Hallucinate some data

← Standard Bayes Net training

Posterior Inference for Mixture Model

We obtain the posterior $p(z^{(i)} = k | x^{(i)}; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ as follows:

$$p(z^{(i)} = k | \mathbf{x}^{(i)}; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{p(\mathbf{x}^{(i)} | z^{(i)} = k; \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(z^{(i)} = k; \boldsymbol{\phi})}{\sum_{j=1}^{K} p(\mathbf{x}^{(i)} | z^{(i)} = j; \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(z^{(i)} = j; \boldsymbol{\phi})} \quad (1)$$

(Soft) EM for GMM

Algorithm 1 Soft EM for GMMs

procedure SoftEM($\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$)

Randomly initialize parameters, $\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$

while not converged **do**

E-Step:

$$c_k^{(i)} \leftarrow p(z^{(i)} = k | \mathbf{x}^{(i)}; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

M-Step:

$$\phi_k \leftarrow \frac{1}{N} \sum_{i=1}^{N} c_k^{(i)}, \forall k$$

$$\boldsymbol{\mu}_k \leftarrow \frac{\sum_{i=1}^{N} c_k^{(i)} \mathbf{x}^{(i)}}{\sum_{i=1}^{N} c_k^{(i)}}, \forall k$$

$$\boldsymbol{\Sigma}_k \leftarrow \frac{\sum_{i=1}^{N} c_k^{(i)} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^T}{\sum_{i=1}^{N} c_k^{(i)}}, \forall k$$

return $(\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$

- Initialize **parameters randomly**
- **while** not converged
  1. **E-Step:**
     Create one training example for each possible value of the **latent variables**
     Weight each example according to model's confidence
     Treat parameters as observed
  2. **M-Step:**
     Set the **parameters** to the values that maximizes likelihood
     Treat pseudo-counts from above as observed

Hard EM vs. Soft EM

Algorithm 1 Hard EM for GMMs

procedure HARDEM($\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$)
  Randomly initialize parameters, $\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$
  while not converged **do**
    E-Step:

$$z^{(i)} \leftarrow \underset{z}{\text{argmax}} \log p(\mathbf{x}^{(i)}|z; \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \log p(z; \boldsymbol{\phi})$$

    M-Step:

$$\phi_k \leftarrow \frac{1}{N} \sum_{i=1}^N \mathbb{I}(z^{(i)} = k), \forall k$$

$$\boldsymbol{\mu}_k \leftarrow \frac{\sum_{i=1}^N \mathbb{I}(z^{(i)} = k)\mathbf{x}^{(i)}}{\sum_{i=1}^N \mathbb{I}(z^{(i)} = k)}, \forall k$$

$$\boldsymbol{\Sigma}_k \leftarrow \frac{\sum_{i=1}^N \mathbb{I}(z^{(i)} = k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^T}{\sum_{i=1}^N \mathbb{I}(z^{(i)} = k)}, \forall k$$

  return $(\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$

Algorithm 1 Soft EM for GMMs

procedure SOFTEM($\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$)
  Randomly initialize parameters, $\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$
  while not converged **do**
    E-Step:

$$c_k^{(i)} \leftarrow p(z^{(i)} = k|\mathbf{x}^{(i)}; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

    M-Step:

$$\phi_k \leftarrow \frac{1}{N} \sum_{i=1}^N c_k^{(i)}, \forall k$$

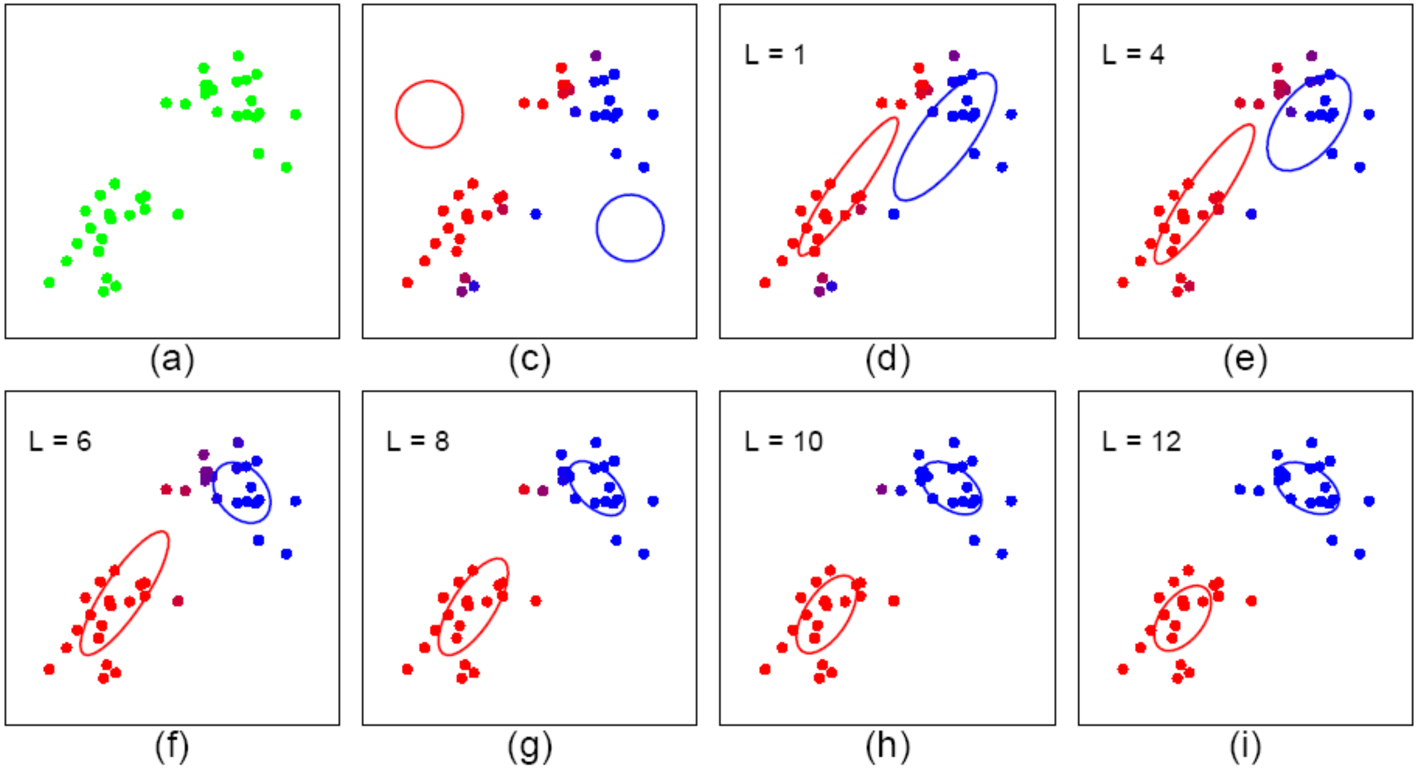
$$\boldsymbol{\mu}_k \leftarrow \frac{\sum_{i=1}^N c_k^{(i)}\mathbf{x}^{(i)}}{\sum_{i=1}^N c_k^{(i)}}, \forall k$$

$$\boldsymbol{\Sigma}_k \leftarrow \frac{\sum_{i=1}^N c_k^{(i)}(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^T}{\sum_{i=1}^N c_k^{(i)}}, \forall k$$

  return $(\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$

(Soft) EM for GMM

- Start:
  - "Guess" the centroid μ_k and coveriance Σ_k of each of the K clusters
- Loop:



Extra Slides

WHY DOES EM WORK?

Extra Slides

Theory underlying EM

- What are we doing?

- Recall that according to MLE, we intend to learn the model parameter that would have maximize the likelihood of the data.

- But we do not observe z, so computing

$$\ell_c(\theta; D) = \log \sum_z p(x, z \mid \theta) = \log \sum_z p(z \mid \theta_z) p(x \mid z, \theta_x)$$

  is difficult!

- What shall we do?

Complete & Incomplete Log Likelihoods

Extra Slides

- Complete log likelihood

  Let X denote the observable variable(s), and Z denote the latent variable(s). If Z could be observed, then

$$\ell_c(\theta; x, z) \stackrel{\text{def}}{=} \log p(x, z \mid \theta)$$

  - Usually, optimizing $l_c()$ given both z and x is straightforward (c.f. MLE for fully observed models).
  - Recalled that in this case the objective for, e.g., MLE, decomposes into a sum of factors, the parameter for each factor can be estimated separately.
  - **But given that Z is not observed, $l_c()$ is a random quantity, cannot be maximized directly**.

- Incomplete log likelihood

  With z unobserved, our objective becomes the log of a marginal probability:

$$\ell_c(\theta; x) = \log p(x \mid \theta) = \log \sum_z p(x, z \mid \theta)$$

  - **This objective won't decouple**

Expected Complete Log Likelihood

Extra Slides

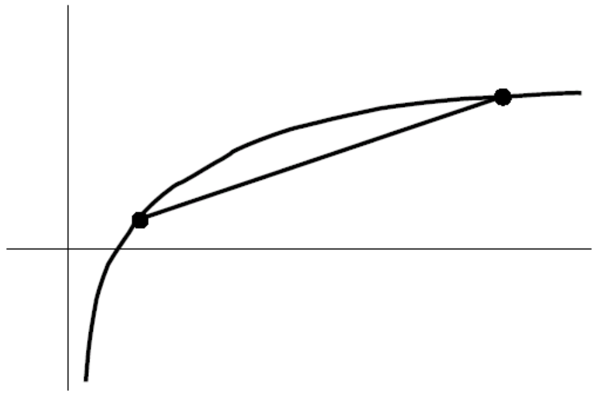
- For ***any*** distribution $q(z)$, define *expected complete log likelihood*:

$$\langle \ell_c(\theta; x, z) \rangle_q \stackrel{\text{def}}{=} \sum_z q(z \mid x, \theta) \log p(x, z \mid \theta)$$

  - A deterministic function of θ
  - Linear in $l_c()$ --- inherit its factorizabiility
  - Does maximizing this surrogate yield a maximizer of the likelihood?

- Jensen's inequality

$$\begin{aligned} \ell(\theta; x) &= \log p(x \mid \theta) \\ &= \log \sum_z p(x, z \mid \theta) \\ &= \log \sum_z q(z \mid x) \frac{p(x, z \mid \theta)}{q(z \mid x)} \\ &\geq \sum_z q(z \mid x) \log \frac{p(x, z \mid \theta)}{q(z \mid x)} \end{aligned}$$



$$\Rightarrow \quad \ell(\theta; x) \geq \langle \ell_c(\theta; x, z) \rangle_q + H_q$$

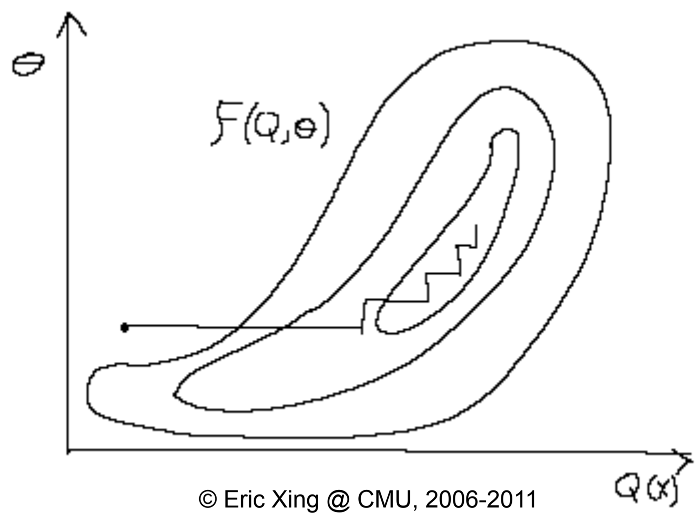
Extra Slides

Lower Bounds and Free Energy

- For fixed data x, define a functional called the free energy:

$$F(q,\theta) \stackrel{\text{def}}{=} \sum_z q(z|x) \log \frac{p(x,z|\theta)}{q(z|x)} \le \ell(\theta;x)$$

- The EM algorithm is coordinate-ascent on F:
  - **E-step:** $$q^{t+1} = \arg\max_q F(q,\theta^t)$$
  - **M-step:** $$\theta^{t+1} = \arg\max_\theta F(q^{t+1},\theta^t)$$



E-step: maximization of expected l_c w.r.t. q

Extra Slides

- Claim:

$$q^{t+1} = \arg\max_q F(q, \theta^t) = p(z \mid x, \theta^t)$$

  - This is the posterior distribution over the latent variables given the data and the parameters. Often we need this at test time anyway (e.g. to perform classification).

- Proof (easy): this setting attains the bound $l(\theta;x) \geq F(q,\theta)$

$$F(p(z|x,\theta^t),\theta^t) = \sum_z p(z|x,\theta^t) \log \frac{p(x,z \mid \theta^t)}{p(z|x,\theta^t)}$$

$$= \sum_z p(z|x,\theta^t) \log p(x \mid \theta^t)$$

$$= \log p(x \mid \theta^t) = \ell(\theta^t; x)$$

- Can also show this result using variational calculus or the fact that $\ell(\theta;x) - F(q,\theta) = \mathrm{KL}\left(q \,\|\, p(z \mid x,\theta)\right)$

E-step ≡ plug in posterior expectation of latent variables

Extra Slides

- Without loss of generality: assume that $p(x,z|\theta)$ is a generalized exponential family distribution:

$$p(x,z|\theta) = \frac{1}{Z(\theta)} h(x,z) \exp\left\{\sum_i \theta_i f_i(x,z)\right\}$$

  - Special cases: if $p(X|Z)$ are GLIMs, then $f_i(x,z) = \eta_i^T(z)\xi_i(x)$

- The expected complete log likelihood under $q^{t+1} = p(z \mid x, \theta^t)$ is

$$\begin{aligned}
\left\langle \ell_c(\theta^t; x, z)\right\rangle_{q^{t+1}} &= \sum_z q(z \mid x, \theta^t) \log p(x, z \mid \theta^t) - A(\theta) \\
&= \sum_i \theta_i^t \left\langle f_i(x,z)\right\rangle_{q(z|x,\theta^t)} - A(\theta) \\
&\overset{p\sim\text{GLIM}}{=} \sum_i \theta_i^t \left\langle \eta_i(z)\right\rangle_{q(z|x,\theta^t)} \xi_i(x) - A(\theta)
\end{aligned}$$

M-step: maximization of expected l_c w.r.t. θ

Extra Slides

- Note that the free energy breaks into two terms:

$$F(q,\theta) = \sum_z q(z|x)\log\frac{p(x,z|\theta)}{q(z|x)}$$
$$= \sum_z q(z|x)\log p(x,z|\theta) - \sum_z q(z|x)\log q(z|x)$$
$$= \langle \ell_c(\theta;x,z)\rangle_q + H_q$$

  - The first term is the expected complete log likelihood (energy) and the second term, which does not depend on θ, is the entropy.

- Thus, in the M-step, maximizing with respect to θ for fixed q we only need to consider the first term:

$$\theta^{t+1} = \arg\max_\theta \langle \ell_c(\theta;x,z)\rangle_{q^{t+1}} = \arg\max_\theta \sum_z q(z|x)\log p(x,z|\theta)$$

  - Under optimal q^{t+1}, this is equivalent to solving a standard MLE of fully observed model $p(x,z|\theta)$, with the sufficient statistics involving z replaced by their expectations w.r.t. $p(z|x,\theta)$.

Extra Slides

Summary: EM Algorithm

- A way of maximizing likelihood function for latent variable models. Finds MLE of parameters when the original (hard) problem can be broken up into two (easy) pieces:
  1. Estimate some "missing" or "unobserved" data from observed data and current parameters.
  2. Using this "complete" data, find the maximum likelihood parameter estimates.

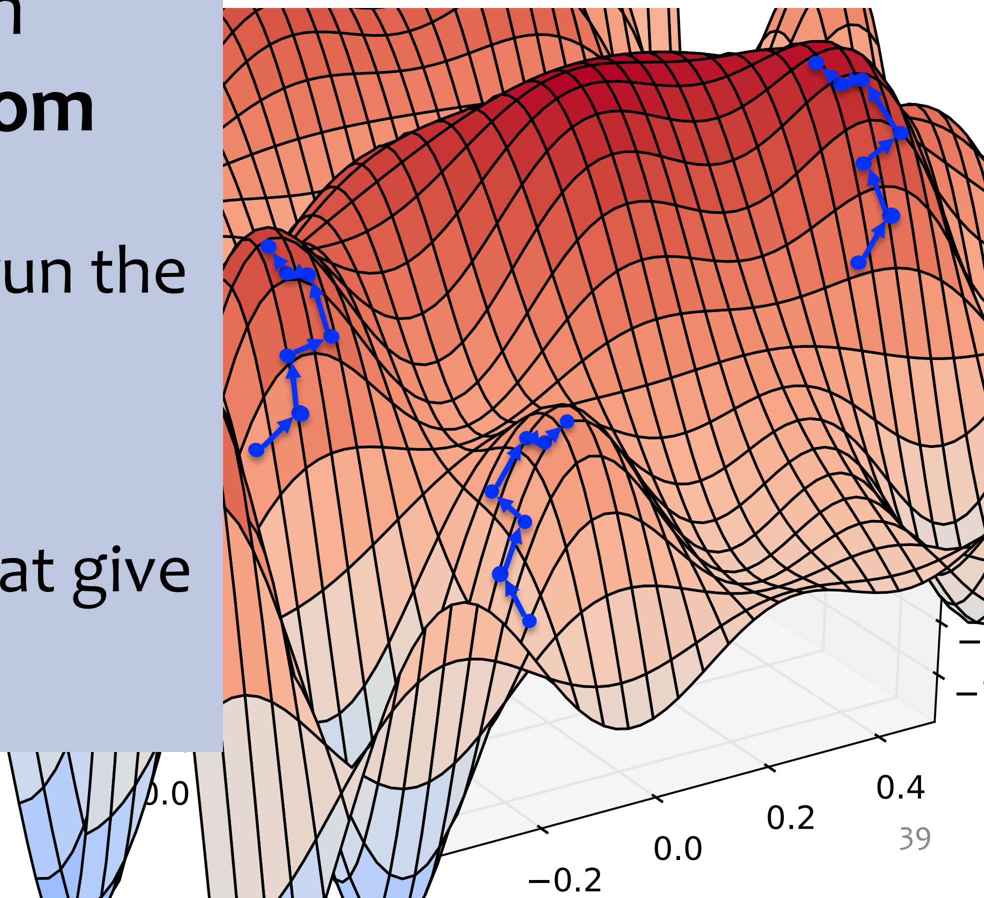
- Alternate between filling in the latent variables using the best guess (posterior) and updating the parameters based on this guess:
  - E-step: $q^{t+1} = \arg\max_{q} F(q, \theta^{t})$
  - M-step: $\theta^{t+1} = \arg\max_{\theta} F(q^{t+1}, \theta^{t})$

- In the M-step we optimize a lower bound on the likelihood. In the E-step we close the gap, making bound=likelihood.

PROPERTIES OF EM

Properties of EM

- EM is *trying* to optimize a **nonconvex** function
- But EM is a **local** optimization algorithm
- Typical solution: **Random Restarts**
  - Just like K-Means, we run the algorithm many times
  - Each time initialize parameters randomly
  - Pick the parameters that give highest likelihood

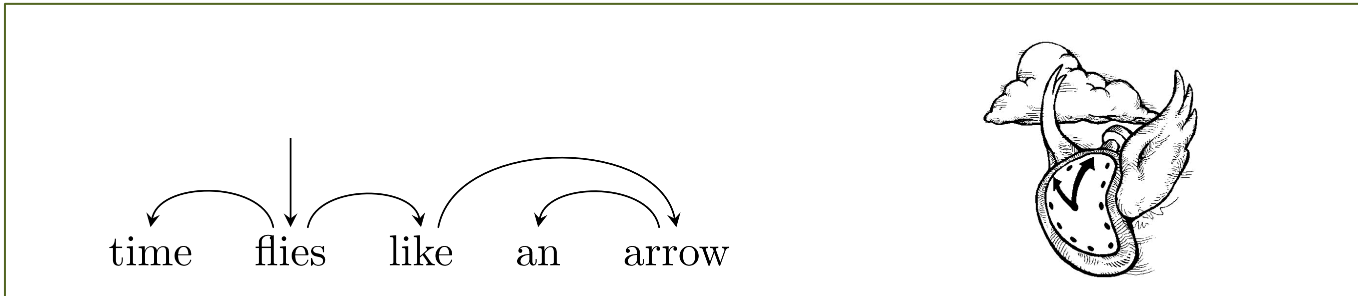


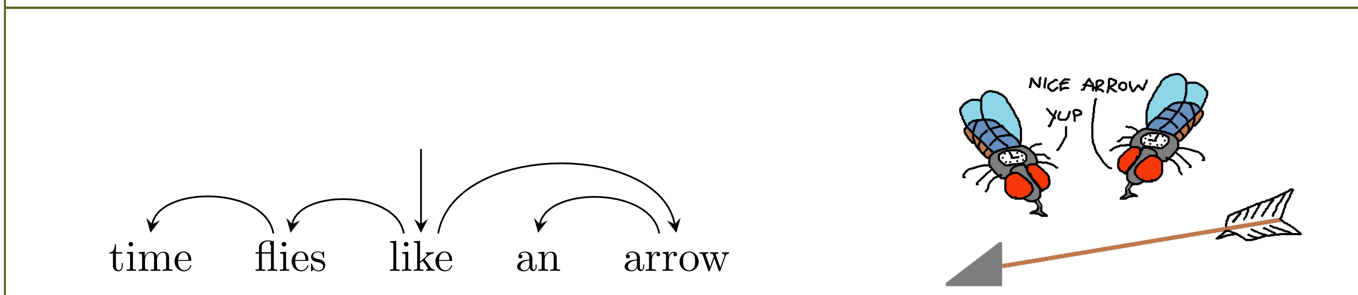
Example: Grammar Induction

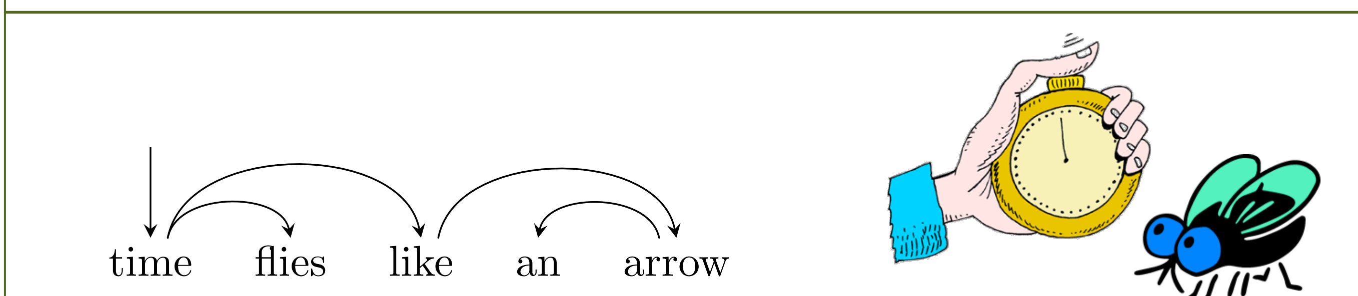
Grammar Induction is an unsupervised learning problem

We try to recover the **syntactic parse** for each sentence without any supervision

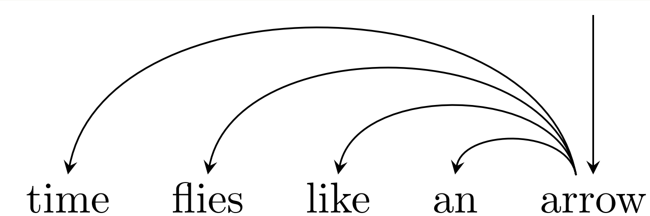
Example: Grammar Induction







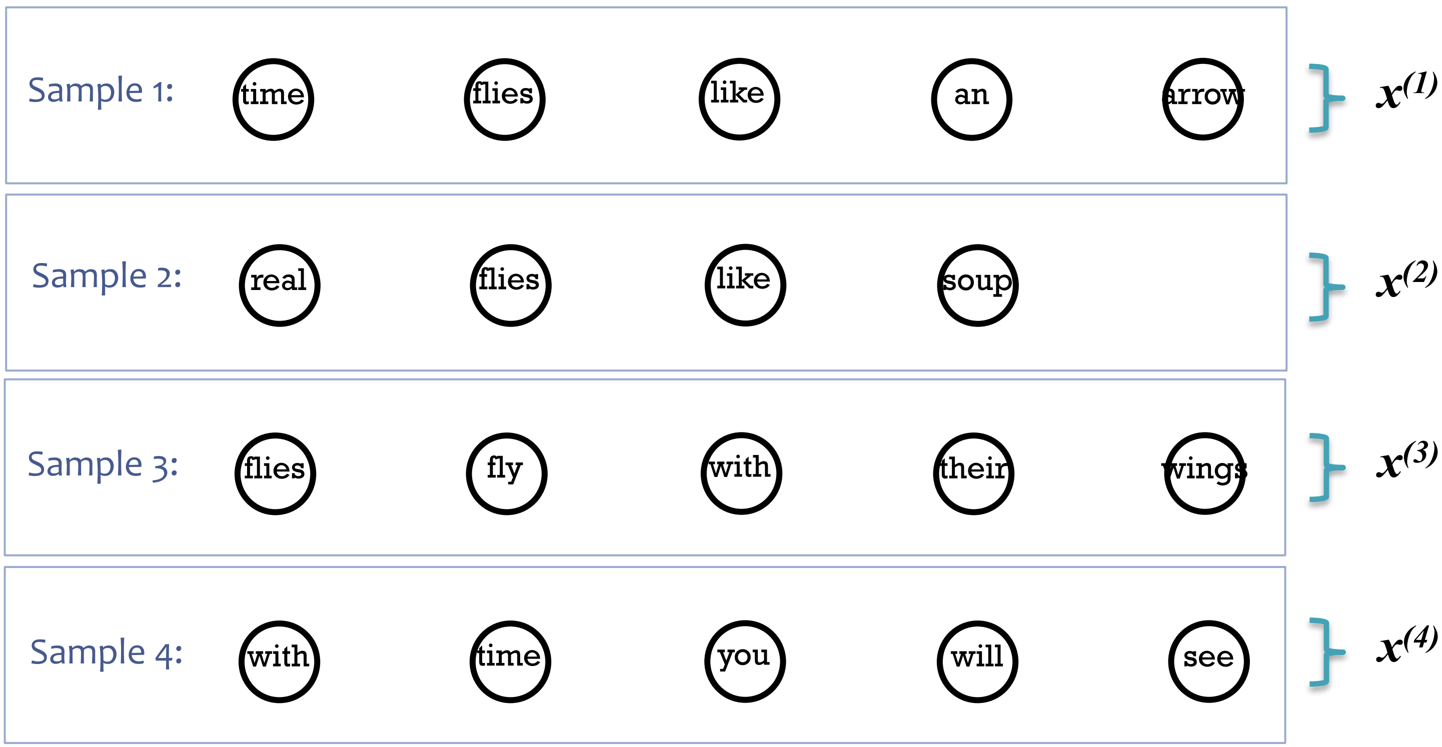
• • •



No semantic interpretation

Example: Grammar Induction

Training Data: Sentences only, without parses



Test Data: Sentences **with** parses, so we can evaluate accuracy

Example: Grammar Induction

Q: Does likelihood correlate with accuracy on a task we care about?

A: Yes, but there is still a wide range of accuracies for a particular likelihood value

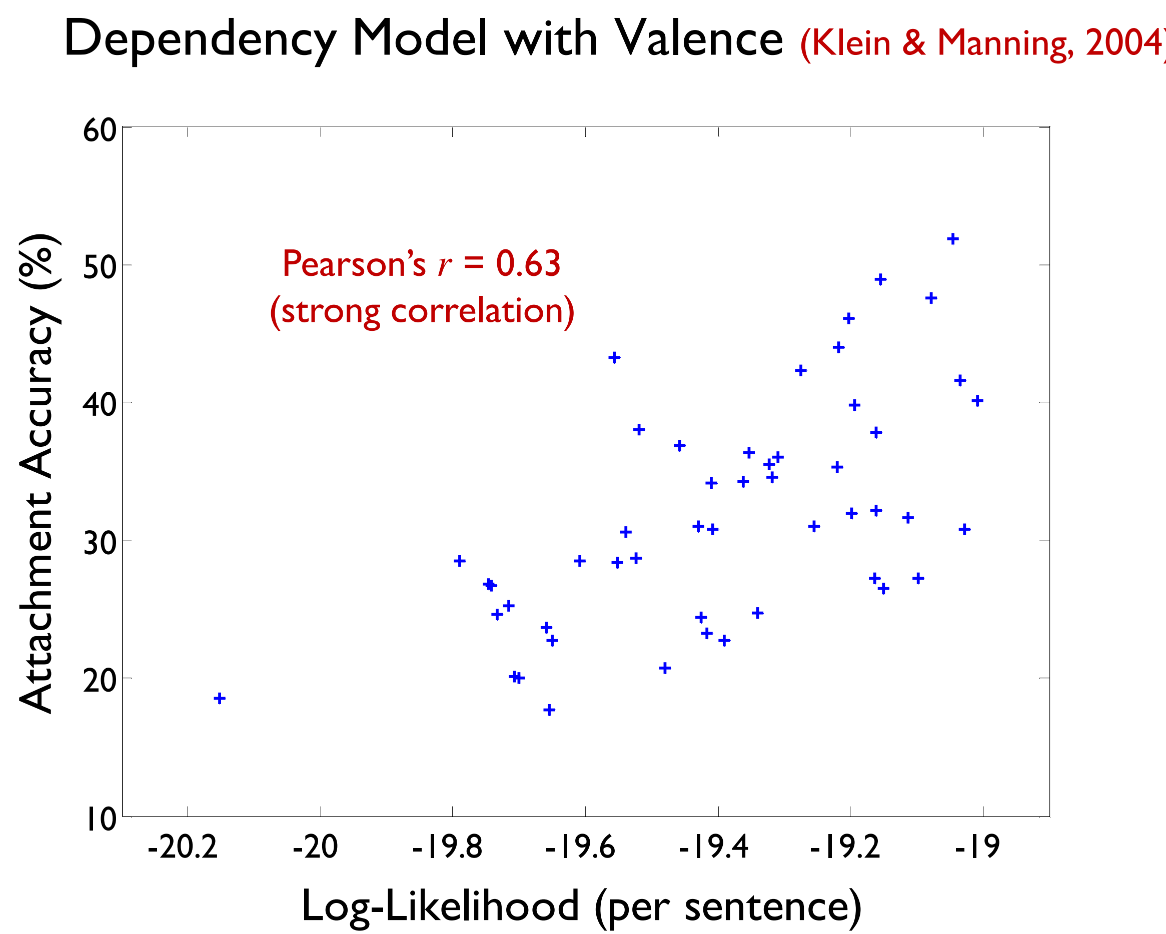


Figure from Gimpel & Smith (NAACL 2012) - slides

Variants of EM

- **Generalized EM**: Replace the M-Step by a single gradient-step that improves the likelihood
- **Monte Carlo EM**: Approximate the E-Step by sampling
- **Sparse EM:** Keep an "active list" of points (updated occasionally) from which we estimate the expected counts in the E-Step
- **Incremental EM / Stepwise EM**: If standard EM is described as a *batch* algorithm, these are the *online* equivalent
- **etc.**

A Report Card for EM

- Some good things about EM:
  - no learning rate (step-size) parameter
  - automatically enforces parameter constraints
  - very fast for low dimensions
  - each iteration guaranteed to improve likelihood

- Some bad things about EM:
  - can get stuck in local minima
  - can be slower than conjugate gradient (especially near convergence)
  - requires expensive inference step
  - is a maximum likelihood/MAP method