NLP 201: EM Algorithm

Jeffrey Flanigan

November 23, 2021

University of California Santa Cruz
jmflanig@ucsc.edu

Many slides and figures from Eric Xing, Matt Gormely, David Sontag, Shubhendu Trivedi and Noah Smith

Plan for Today

- Unsupervised learning
- K-Means clustering
- Gaussian mixture models
- Latent variable models: Applications
- EM algorithm
- Forward-backward (Baum-Welch) algorithm

Unsupervised Learning

Today, we'll be talking about **unsupervised learning**.

- We'll start with the simplest unsupervised setting: **clustering**
- We'll cover unsupervised learning in graphical models
- Today's main example: **unsupervised part-of-speech (POS) induction**

Learning with Latent Variables

Unsupervised learning as learning in graphical models with partially observed data (**latent variables**).

Applications:

- Document clustering
- POS induction
- Grammar induction
- Alignment (for machine translation)
- and many more

Aside: Types of Machine Learning

- **Supervised learning**: learn from input-output pairs $(x_1, y_1) \dots (x_N, y_N)$
- **Unsupervised learning**: learn from unlabeled examples $x_1 \dots x_N$
- **Semi-supervised learning**: have some labeled data $(x_1, y_1) \dots (x_M, y_M)$ and some unlabeled data $x_1 \dots x_N$
- **Self-supervised learning**: convert an unlabeled dataset into a labeled dataset by predicting the input (ELMO and BERT do this).
- **Reinforcement learning**: agent performs actions, learns from rewards or punishments. Goal: maximize total reward
- **Indirect or weakly supervised learning**: indirect supervision. Example: semantic parser learns to parse from QA pairs

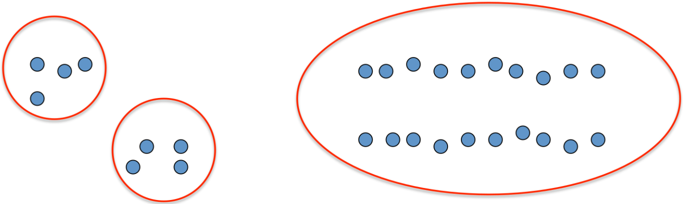
Clustering

Clustering:

- Unsupervised learning
- Requires data, but no labels
- Detect patterns e.g. in
  - Group emails or search results
  - Customer shopping patterns
  - Regions of images
- Useful when don't know what you're looking for
- But: can get gibberish

Clustering

- Basic idea: group together similar instances
- Example: 2D point patterns

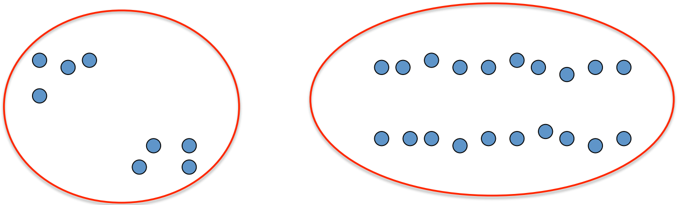


Data are points in a vector space $\mathbb{R}^d$.

Example: embeddings of documents or words, which we want to cluster into groups

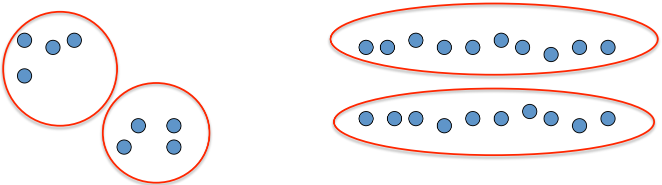
Clustering

- Basic idea: group together similar instances
- Example: 2D point patterns



Clustering

- Basic idea: group together similar instances
- Example: 2D point patterns



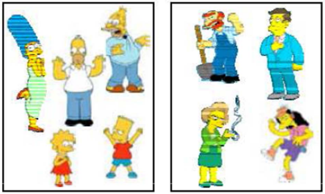
- What could "similar" mean?
  - One option: small Euclidean distance (squared)

$$\mathrm{dist}(\vec{x}, \vec{y}) = ||\vec{x} - \vec{y}||_2^2$$

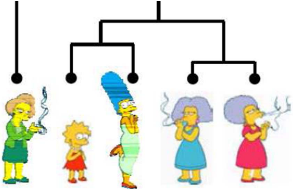
  - Clustering results are crucially dependent on the measure of similarity (or distance) between "points" to be clustered

Clustering algorithms

- Partition algorithms (Flat)
  - K-means
  - Mixture of Gaussian
  - Spectral Clustering



- Hierarchical algorithms
  - Bottom up – agglomerative
  - Top down – divisive



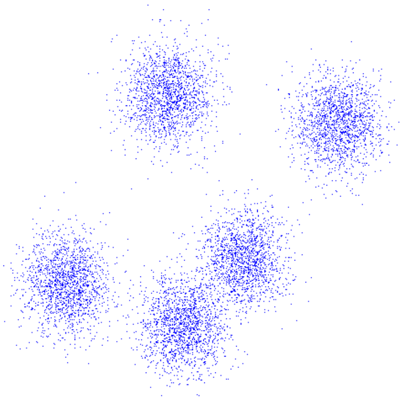
K-Means

- An iterative clustering algorithm

  – Initialize: Pick *K* random points as cluster centers

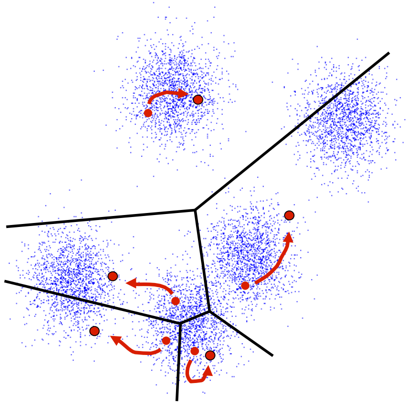
  – Alternate:
    1. Assign data points to closest cluster center
    2. Change the cluster center to the average of its assigned points

  – Stop when no points' assignments change

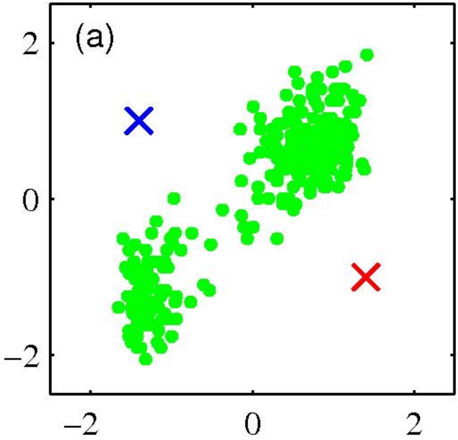


K-Means

- An iterative clustering algorithm
  - Initialize: Pick K random points as cluster centers
  - Alternate:
    1. Assign data points to closest cluster center
    2. Change the cluster center to the average of its assigned points
  - Stop when no points' assignments change



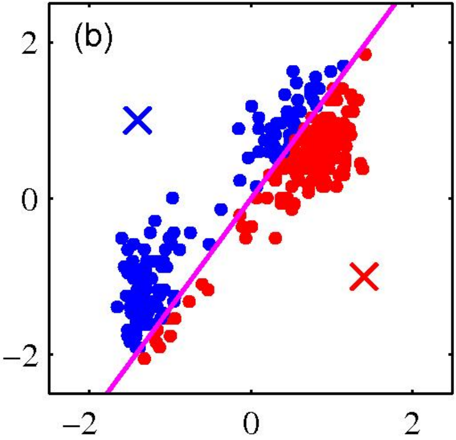
K-means clustering: Example



- Pick *K* random points as cluster centers (means)

Shown here for *K*=2

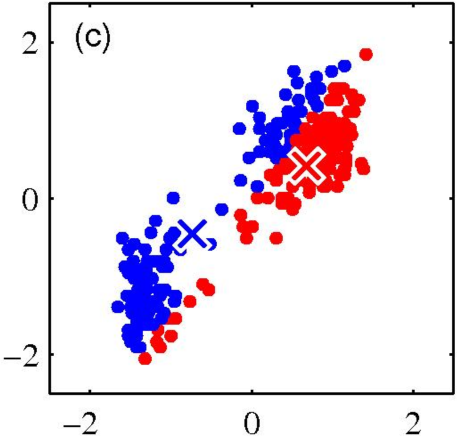
K-means clustering: Example



Iterative Step 1

- Assign data points to closest cluster center

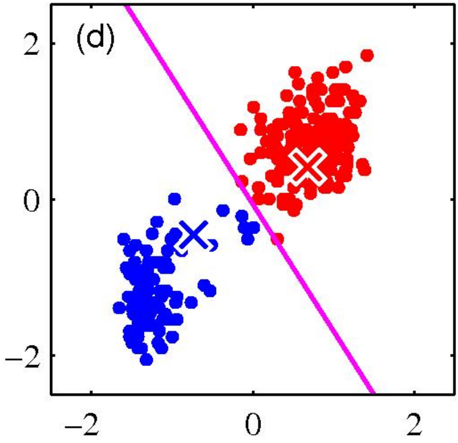
K-means clustering: Example



Iterative Step 2

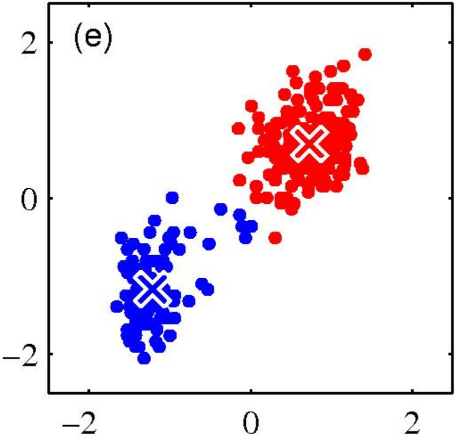
- Change the cluster center to the average of the assigned points

K-means clustering: Example

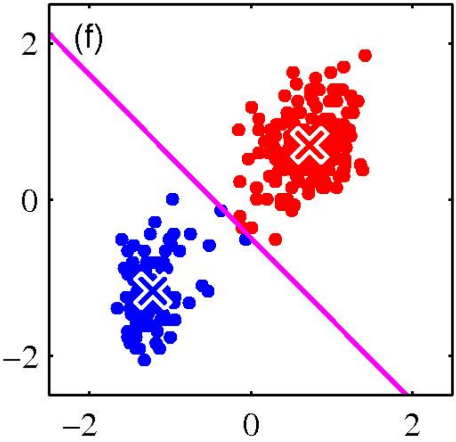


- Repeat until convergence

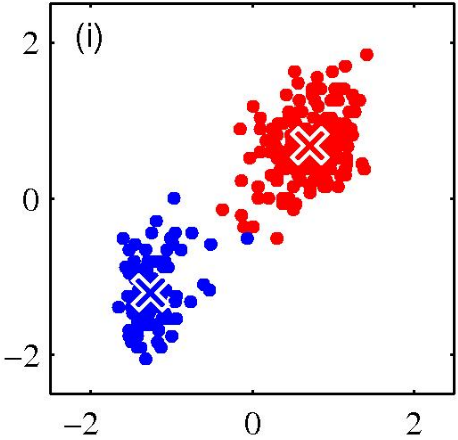
K-means clustering: Example



K-means clustering: Example



K-means clustering: Example



Kmeans Convergence

Objective

$$\min_{\mu}\min_{C}\sum_{i=1}^{k}\sum_{x\in C_i}|x-\mu_i|^2$$

1. Fix μ, optimize *C:*

$$\min_{C}\sum_{i=1}^{k}\sum_{x\in C_i}|x-\mu_i|^2 = \min_{c}\sum_{i}^{n}\left|x_i-\mu_{x_i}\right|^2$$

$\boxed{\textit{Step 1 of kmeans}}$

2. Fix *C,* optimize μ:

$$\min_{\mu}\sum_{i=1}^{k}\sum_{x\in C_i}|x-\mu_i|^2$$

   – Take partial derivative of μ_i and set to zero, we have

$$\mu_i = \frac{1}{|C_i|}\sum_{x\in C_i} x$$

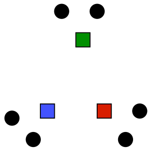
$\boxed{\textit{Step 2 of kmeans}}$

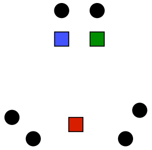
Kmeans takes an alternating optimization approach, each step is guaranteed to decrease the objective – thus guaranteed to converge

[Slide from Alan Fern]

Initialization

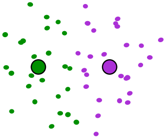
- K-means **algorithm** is a heuristic
  - Requires initial means
  - It does matter what you pick!
  - What can go wrong?
  - Various schemes for preventing this kind of thing: variance-based split / merge, initialization heuristics



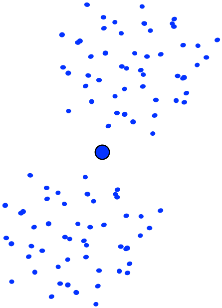


K-Means Getting Stuck

A local optimum:

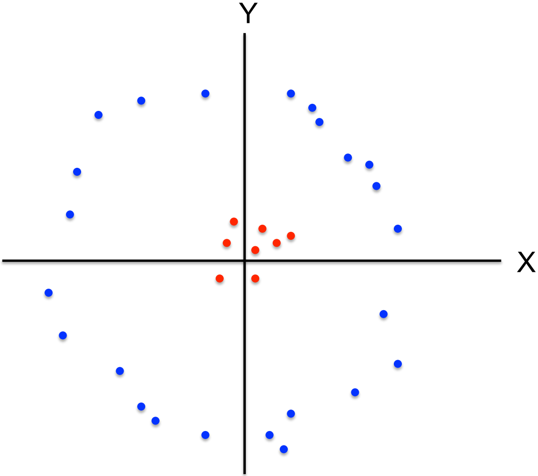


Would be better to have
one cluster here

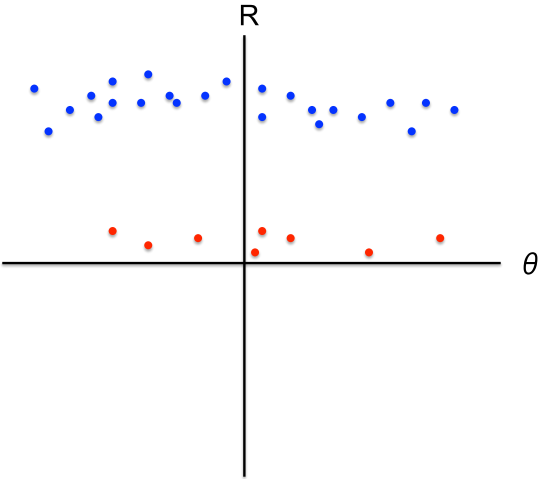


… and two clusters here

K-means not able to properly cluster



Changing the features (distance function) can help



Let's look at another (closely related) clustering method:

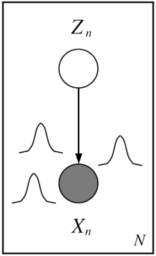
Gaussian mixture models (GMMs)

Mixture Models

Mixture Models, con'd

- A density model $p(x)$ may be multi-modal.
- We may be able to model it as a mixture of uni-modal distributions (e.g., Gaussians).
- Each mode may correspond to a different sub-population (e.g., male and female).

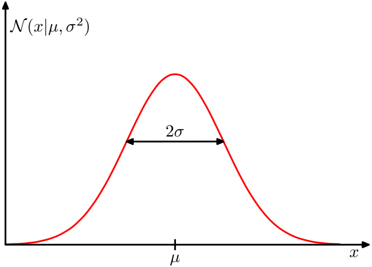
$\Rightarrow$



Reminder: univariate Gaussian distribution

$$\mathcal{N}(x;\, \mu, \sigma^2) \;=\; \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{ -\frac{1}{2\sigma^2}(x-\mu)^2 \right\}$$

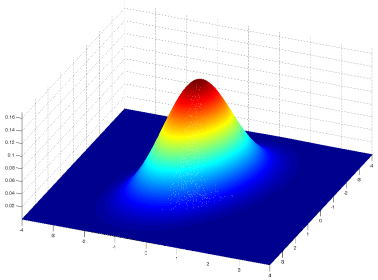
- mean μ determines location
- variance σ^2; standard deviation $\sqrt{\sigma^2}$ determines the spread around μ



Multivariate Gaussian

- Gaussian distribution of a random vector $\mathbf{x}$ in $\mathbb{R}^d$:

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$



- The $\frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}}$ factor ensures it's a pdf (integrates to one).

Aside: From Univariate to Multivariate Gaussian

Univariate Gaussian centered at zero: $\mathcal{N}(x) \propto \exp\Big(-\frac{1}{2\sigma^2}x^2\Big)$
Let's add another dimension:

$$\mathcal{N}(\mathbf{x}) \propto \exp\Big(-\tfrac{1}{2\sigma^2}(x_1^2 + x_2^2)\Big) = \exp\Big(-\tfrac{1}{2\sigma^2}\mathbf{x}^T\mathbf{x}\Big)$$

Center it at μ, and allow for x_1 and x_2 to have different variances:

$$\begin{aligned}\mathcal{N}(\mathbf{x}) &\propto \exp\Big(-\frac{(x_1-\mu_1)^2}{2\sigma_1^2} - \frac{(x_2-\mu_2)^2}{2\sigma_2^2}\Big)\\ &= \exp\Big(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\Big)\end{aligned}$$

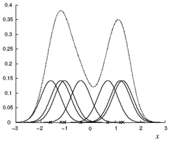
where $\Sigma = \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix}$. This is a 2d Gaussian squashed or expanded along x_1 or x_2. A general Σ allows for rotations.

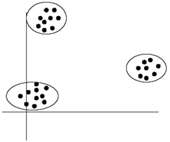
Gaussian Mixture Models (GMMs)

- Consider a mixture of *K* Gaussian components:

$$p(x_n|\mu,\Sigma)=\sum_k \pi_k N(x,|\mu_k,\Sigma_k)$$

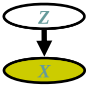
mixture proportion   mixture component





- This model can be used for unsupervised clustering.
  - This model (fit by AutoClass) has been used to discover new kinds of stars in astronomical data, etc.

Gaussian Mixture Models (GMMs)



- Consider a mixture of *K* Gaussian components:
  - *Z* is a latent class indicator vector:

$$p(z_n) = \mathrm{multi}(z_n : \pi) = \prod_k (\pi_k)^{z_n^k}$$

  - *X* is a conditional Gaussian variable with a class-specific mean/covariance

$$p(x_n \mid z_n^k = 1, \mu, \Sigma) = \frac{1}{(2\pi)^{m/2} |\Sigma_k|^{1/2}} \exp\left\{ -\tfrac{1}{2}(x_n - \mu_k)^T \Sigma_k^{-1} (x_n - \mu_k) \right\}$$

  - The likelihood of a sample:

mixture component

mixture proportion

$$\begin{aligned} p(x_n | \mu, \Sigma) &= \sum_k p(z^k = 1 \mid \pi)\, p(x, | z^k = 1, \mu, \Sigma) \\ &= \sum_{z_n} \prod_k \left((\pi_k)^{z_n^k} N(x_n : \mu_k, \Sigma_k)^{z_n^k} \right) = \sum_k \pi_k N(x, | \mu_k, \Sigma_k) \end{aligned}$$

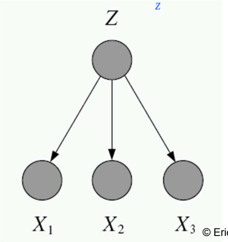
Why is Learning Harder?

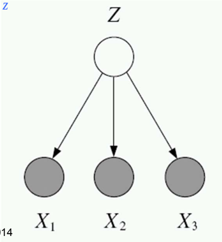
- In fully observed iid settings, the log likelihood decomposes into a sum of local terms (at least for directed models).

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

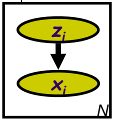
- With latent variables, all the parameters become coupled together via marginalization

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





Toward the EM algorithm



- Recall MLE for completely observed data

- Data log-likelihood

$$\begin{aligned}\ell(\boldsymbol{\theta}; D) &= \log \prod_n p(z_n, x_n) = \log \prod_n p(z_n \mid \pi) p(x_n \mid z_n, \mu, \sigma) \\ &= \sum_n \log \prod_k \pi_k^{z_n^k} + \sum_n \log \prod_k N(x_n; \mu_k, \sigma)^{z_n^k} \\ &= \sum_n \sum_k z_n^k \log \pi_k - \sum_n \sum_k z_n^k \tfrac{1}{2\sigma^2} (x_n - \mu_k)^2 + C\end{aligned}$$

- MLE

$$\hat{\pi}_{k,MLE} = \arg\max_{\pi} \ell(\boldsymbol{\theta}; D),$$

$$\hat{\mu}_{k,MLE} = \arg\max_{\mu} \ell(\boldsymbol{\theta}; D) \qquad \Rightarrow \quad \hat{\mu}_{k,MLE} = \frac{\sum_n z_n^k x_n}{\sum_n z_n^k}$$

$$\hat{\sigma}_{k,MLE} = \arg\max_{\sigma} \ell(\boldsymbol{\theta}; D)$$

- What if we do not know z_n?

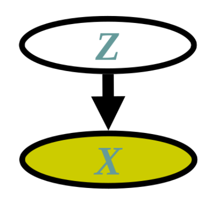
Towards the EM Algorithm

We have a chicken and egg problem:
If we knew the variable assignments, we could solve for the parameters. (Maximization)
If we knew the parameters, we could find the best assignment of latent variables (using our inference algorithms). (Expectation)

Latent Variable Example: Clustering

Our data is a collection of data points $x_1 \dots x_N$.

Each data point has an latent (unobserved) R.V. z_i, which is a cluster id.

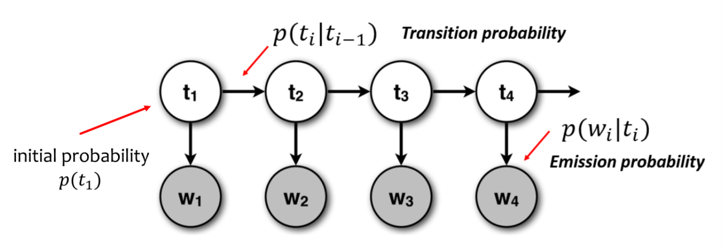


Our goal is to infer the cluster assignments z_i.

Latent Variable Example: HMM

Our data is a collection of sentences $s_1 \ldots s_N$.

For each sentence, we observe the words $w_1 \ldots w_M$.
We have a model of the data:



Our goal is to infer the latent (unobserved) labels $t_1 \ldots t_M$ for each word.

Latent Variables in Graphical Models

In general, we are interested in learning with partially observed data in graphical models.

$\mathbf{x}$ is a **collection of observed random variables**, and $\mathbf{z}$ is a **collection of unobserved (latent) random variables**.
We have a joint model $p(\mathbf{x}, \mathbf{z}; \theta)$.
We would like to learn the parameters that maximize the data:

$$\begin{aligned}\hat{\theta} &= \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{N} p(\mathbf{x}_i; \theta) \\ &= \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{N} \sum_{\mathbf{z}} p(\mathbf{x}_i, \mathbf{z}; \theta)\end{aligned}$$

We also might want to compute the most probable assignment of the latent variables with the learned model.

EM Algorithm Motivation

We have a chicken and egg problem:
If we knew the variable assignments, we could solve for the parameters. (Maximization)
If we knew the parameters, we could find the best assignment of latent variables (using our inference algorithms). (Expectation)

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

Estimate unobserved variables

MLE given the estimated values of unobserved variables

(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

Estimate unobserved variables

MLE given the estimated values of unobserved variables

Example: Hard EM vs. Soft EM for Gaussian Mixture Models

Algorithm 1 Hard EM for GMMs

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

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

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

Algorithm 1 Soft EM for GMMs

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

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

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:

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

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

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

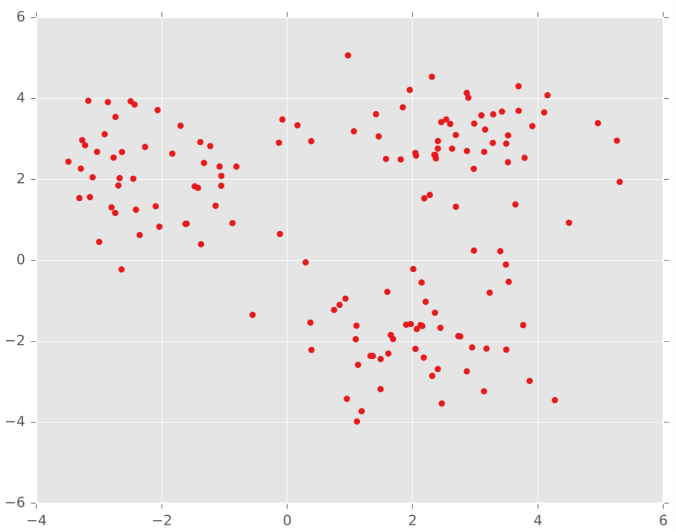
return $(\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$

(Here I'm using ϕ_k instead of π_k for the mixture proportion.)

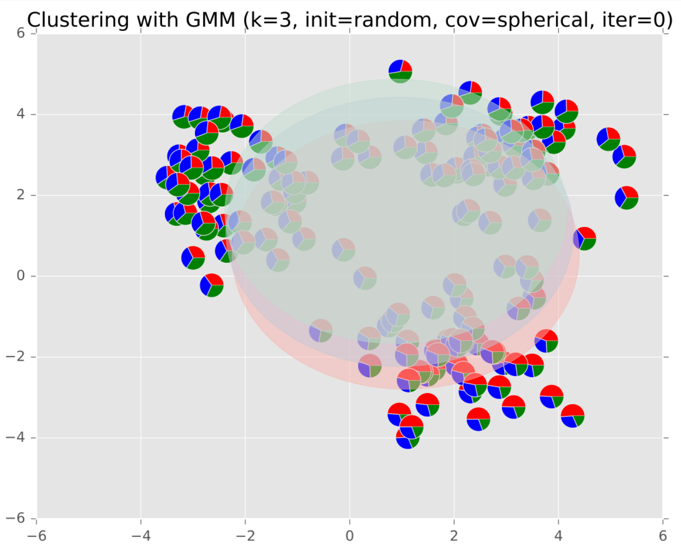
K-Means is Hard EM with Σ_k and ϕ_k fixed to 1

GMM Example

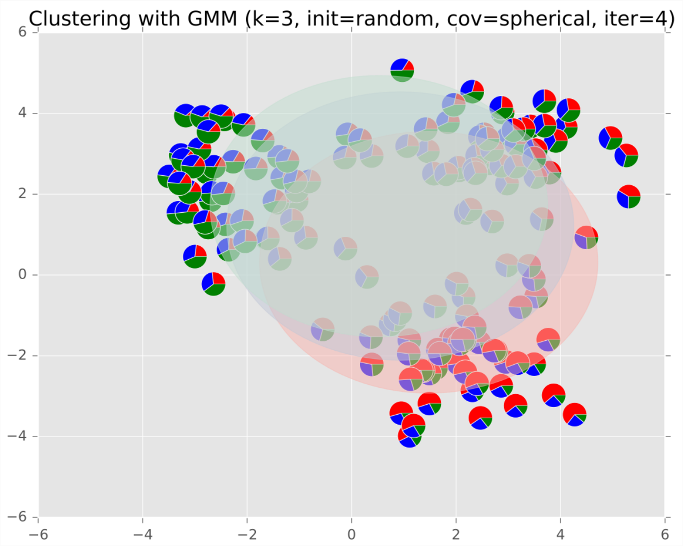
GMM Example



GMM Example

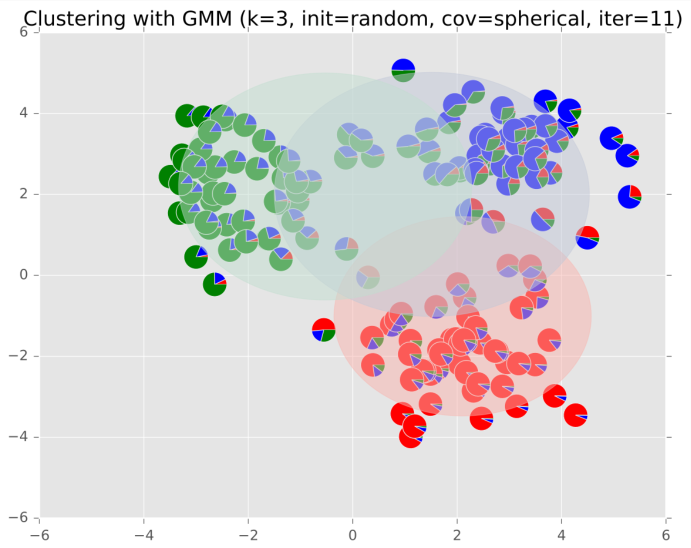

Clustering with GMM (k=3, init=random, cov=spherical, iter=0)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

GMM Example

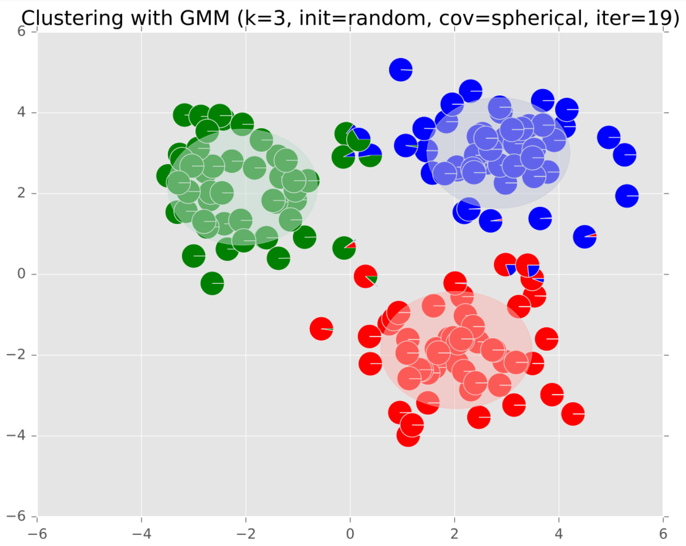


Clustering with GMM (k=3, init=random, cov=spherical, iter=4)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

GMM Example


Clustering with GMM (k=3, init=random, cov=spherical, iter=11)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

GMM Example



Clustering with GMM (k=3, init=random, cov=spherical, iter=19)
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

K-Means vs. GMM

Convergence:

K-Means tends to **converge** much faster than a **GMM**

Speed:

Each iteration of **K-Means** is **computationally less intensive** than each iteration of a **GMM**

Initialization:

To **initialize** a **GMM**, we typically first run **K-Means** and use the resulting cluster centers as the means of the Gaussian components

Output:

A **GMM** yields a **probability distribution** over the cluster assignment for each point; whereas **K-Means** gives a single **hard assignment**

Board Work: Hard-EM for HMMs

- (3 min) Write down how you would apply hard-EM for unsupervised POS induction. Things to consider:
  - What are the examples?
  - What is the E-step and M-step?
  - Where does the loop over the data go in your code?
- (3 min) Discuss with a partner

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

Estimate unobserved variables

MLE given the estimated values of unobserved variables

(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

Estimate unobserved variables

MLE given the estimated values of unobserved variables

Other latent variable models in NLP

Latent Variable Example: Word alignment for MT

Our data is a collection of pairs of sentences.

We observe the words in the original sentence (the source) and the words in the translated sentence (the target).

Our goal is to infer the latent (unobserved) alignments $t_1 \dots t_M$ between the words in the source and the words in the target.

IBM Alignment models

If we had explicit word alignments we could estimate translation tables directly from them.

mi lasciate in pace

Leave me in peace

Lasciate i monti

Leave the mountains

But we don’t have word alignments — just sentence alignments!

IBM Alignment models

Unsupervised models for aligning words and phrases in parallel sentences.

mi lasciate in pace

Leave me in peace

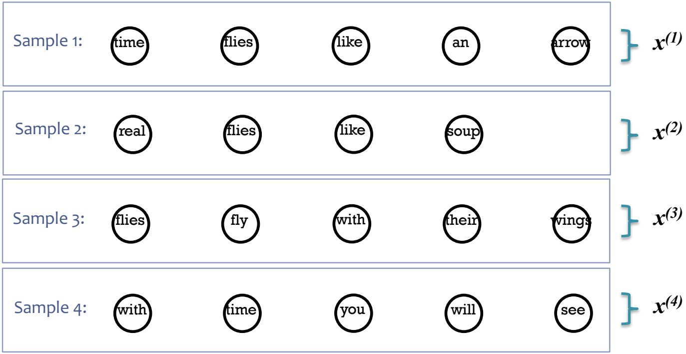
Lasciate i monti

Leave the mountains

Brown, Peter F. (1993). “The mathematics of statistical machine translation: Parameter estimation,” Computational Linguistics

Grammar Induction

Training Data: Sentences only, without parses



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

Grammar Induction

Question: Can maximizing (unsupervised) marginal likelihood produce useful results?

Answer: Let's look at an example…

- **Babies** learn the syntax of their **native language** (e.g. English) just by **hearing** many sentences
- Can a **computer** similarly learn syntax of a **human language** just by looking at lots of example sentences?
  - This is the problem of Grammar Induction!
  - It's an unsupervised learning problem
  - We try to recover the **syntactic structure** for each sentence without any supervision

End

We stopped here.

Soft-EM for HMMs: the Forward-Backward algorithm

Learning Problem

- Given HMM with unknown parameters $\theta = \{\{\pi_i\}, \{p_{ij}\}, \{q_i^k\}\}$ and observation sequence $\mathbf{O} = \{O_t\}_{t=1}^T$

  find parameters that maximize likelihood of observed data

  $$\arg\max_\theta p(\{O_t\}_{t=1}^T|\theta)$$

  But likelihood doesn't factorize since observations not i.i.d.

  hidden variables – state sequence $\{S_t\}_{t=1}^T$

  EM (Baum-Welch) Algorithm:

  E-step – Fix parameters, find expected state assignments

  M-step – Fix expected state assignments, update parameters

The Forward-Backward Algorithm

- Also called Baum-Welch algorithm
- A special case of EM algorithm
  - Repeat until converge
    - E-step:
      - Expected state occupancy count $\gamma_t(j) = P(y_t = j|\boldsymbol{x}, \lambda)$
        - Probability of being in state j at time t
      - Expected state transition count $\xi_t(i, j) = P(y_t = i, y_{t+1} = j|\boldsymbol{x}, \lambda)$
        - Probability of being in state i at time t and in state j at time t+1
    - M-step:
      - Estimate π_i, a_{ij}, b_{ik}

Baum-Welch (EM) Algorithm

- Start with random initialization of parameters
- **E-step** – Fix parameters, find expected state assignments

$$\gamma_i(t) = p(S_t = i|O,\theta) = \frac{\alpha_t^i \beta_t^i}{\sum_j \alpha_t^j \beta_t^j} \qquad \mathbf{O} = \{O_t\}_{t=1}^{T}$$

Forward-Backward algorithm

$$\xi_{ij}(t) = p(S_{t-1} = i, S_t = j|O,\theta)$$

$$= \frac{p(S_{t-1} = i|O,\theta)p(S_t = j, O_t, \ldots, O_T|S_{t-1} = i, \theta)}{p(O_t, \ldots, O_T|S_{t-1} = i, \theta)}$$

$$= \frac{\gamma_i(t-1)\ p_{ij}\ q_j^{O_t}\ \beta_t^j}{\beta_{t-1}^i}$$

Baum-Welch (EM) Algorithm

- Start with random initialization of parameters
- **E-step**

$$\gamma_i(t) = p(S_t = i | O, \theta)$$

$$\xi_{ij}(t) = p(S_{t-1} = i, S_t = j | O, \theta)$$

$\sum_{t=1}^{T} \gamma_i(t)$ = expected # times in state i

$\sum_{t=1}^{T\text{-}1} \gamma_i(t)$ = expected # transitions from state i

$\sum_{t=1}^{T-1} \xi_{ij}(t)$ = expected # transitions from state i to j

- **M-step**

$$\pi_i = \gamma_i(1)$$

$$p_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

$$q_i^k = \frac{\sum_{t=1}^{T} \delta_{O_t = k} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}$$

Marginal Inference

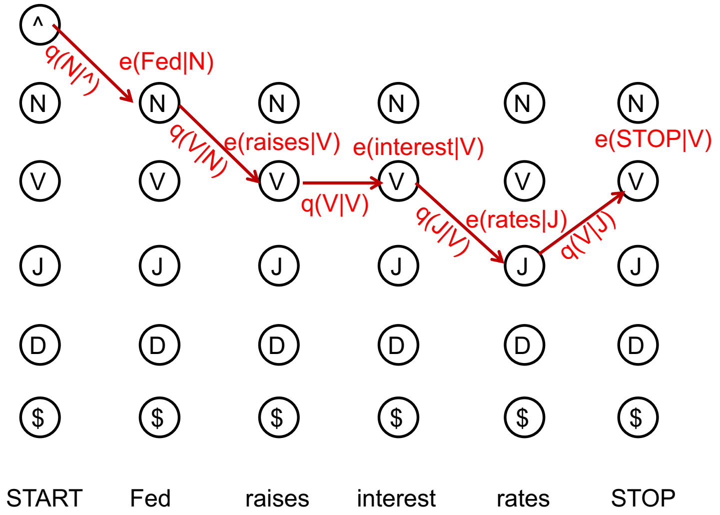
- Problem: find the marginal probability of each tag for y_i

$$p(x_1 \ldots x_n, y_i) = \sum_{y_1 \ldots y_{i-1}} \sum_{y_{i+1} \ldots y_n} p(x_1 \ldots x_n, y_1 \ldots y_{n+1})$$

Compare it to "Viterbi Inference"

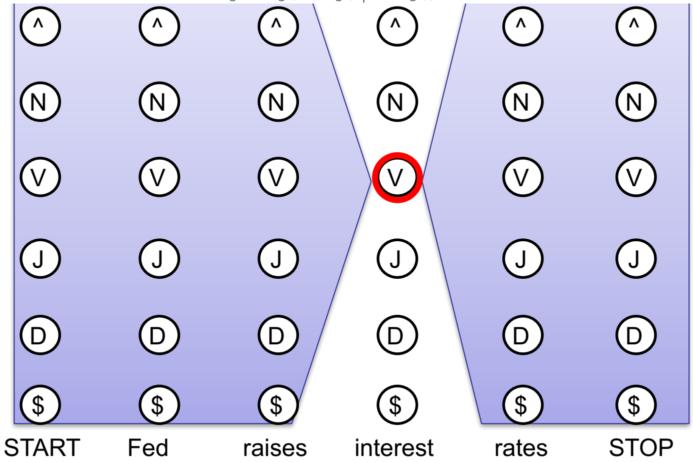
$$\pi(i, y_i) = \max_{y_1 \ldots y_{i-1}} p(x_1 \ldots x_i, y_1 \ldots y_i)$$

The State Lattice / Trellis: Viterbi



The State Lattice / Trellis: Marginal

$$p(x_1 \dots x_n, y_i) = \sum_{y_1 \dots y_{i-1}} \sum_{y_{i+1} \dots y_n} p(x_1 \dots x_n, y_1 \dots y_{n+1})$$



Dynamic Programming!

$$p(x_1 \dots x_n, y_i) = p(x_1 \dots x_i, y_i) p(x_{i+1} \dots x_n | y_i)$$

- Sum over all paths, on both sides of each y_i

$$\alpha(i, y_i) = p(x_1 \dots x_i, y_i) = \sum_{y_1 \dots y_{i-1}} p(x_1 \dots x_i, y_1 \dots y_i)$$

$$= \sum_{y_{i-1}} e(x_i | y_i) q(y_i | y_{i-1}) \alpha(i-1, y_{i-1})$$

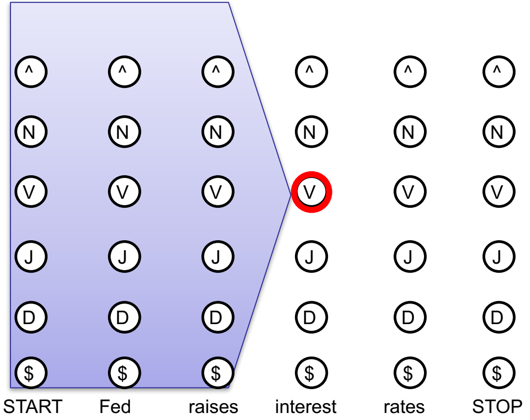
$$\beta(i, y_i) = p(x_{i+1} \dots x_n | y_i) = \sum_{y_{i+1} \dots y_n} p(x_{i+1} \dots x_n, y_{i+1} \dots y_{n+1} | y_i)$$

$$= \sum_{y_{i+1}} e(x_{i+1} | y_{i+1}) q(y_{i+1} | y_i) \beta(i+1, y_{i+1})$$

The State Lattice / Trellis: Forward

$$\alpha(i, y_i) = p(x_1 \dots x_i, y_i) = \sum_{y_1 \dots y_{i-1}} p(x_1 \dots x_i, y_1 \dots y_i)$$

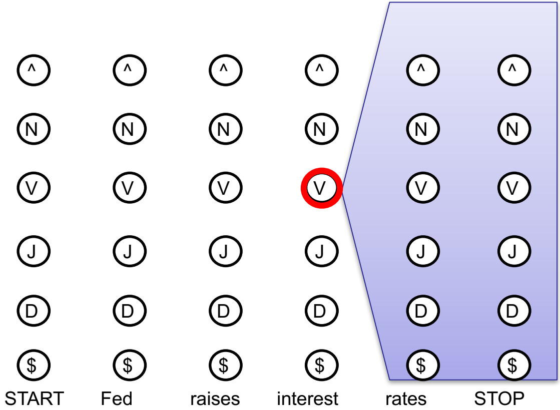
$$= \sum_{y_{i-1}} e(x_i|y_i) q(y_i|y_{i-1}) \alpha(i-1, y_{i-1})$$



The State Lattice / Trellis: Backward

$$\beta(i, y_i) = p(x_{i+1} \dots x_n | y_i) = \sum_{y_{i+1} \dots y_n} p(x_{i+1} \dots x_n, y_{i+1} \dots y_{n+1} | y_i)$$

$$= \sum_{y_{i+1}} e(x_{i+1} | y_{i+1}) q(y_{i+1} | y_i) \beta(i+1, y_{i+1})$$



Forward Backward Algorithm

- Two passes: one forward, one back
  - Forward:

$$\alpha(0, y_0) = \begin{cases} 1 \text{ if } y_0 == START \\ 0 \text{ otherwise} \end{cases}$$

    - For i = 1 … n

$$\alpha(i, y_i) = \sum_{y_{i-1}} e(x_i|y_i)q(y_i|y_{i-1})\alpha(i-1, y_{i-1})$$

  - Backward:

$$\beta(n, y_n) = \begin{cases} q(y_{n+1}|y_n) \text{ if } y_{n+1} = \text{STOP} \\ 0 \text{ otherwise} \end{cases}$$

    - For i = n-1 ... 0

$$\beta(i, y_i) = \sum_{y_{i+1}} e(x_{i+1}|y_{i+1})q(y_{i+1}|y_i)\beta(i+1, y_{i+1})$$

Question

Does the Forward-Backward algorithm remind you of an algorithm we recently talked about?

How about this algorithm?

Input: a factor graph with no cycles
Output: exact marginals for each variable and factor

Algorithm:

1. Initialize the messages to the uniform distribution.

$$\boxed{\mu_{i \to \alpha}(x_i) = 1} \quad \boxed{\mu_{\alpha \to i}(x_i) = 1}$$

1. Choose a root node.
2. Send messages from the **leaves** to the **root**.
   Send messages from the **root** to the **leaves**.

$$\boxed{\mu_{i \to \alpha}(x_i) = \prod_{\alpha \in \mathcal{N}(i) \setminus \alpha} \mu_{\alpha \to i}(x_i)} \quad \boxed{\mu_{\alpha \to i}(x_i) = \sum_{\boldsymbol{x}_{\boldsymbol{\alpha}} : \boldsymbol{x}_{\boldsymbol{\alpha}}[i] = x_i} \psi_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}}) \prod_{j \in \mathcal{N}(\alpha) \setminus i} \mu_{j \to \alpha}(\boldsymbol{x}_{\boldsymbol{\alpha}}[i])}$$

1. Compute the beliefs (unnormalized marginals).

$$\boxed{b_i(x_i) = \prod_{\alpha \in \mathcal{N}(i)} \mu_{\alpha \to i}(x_i)} \quad \boxed{b_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}}) = \psi_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}}) \prod_{i \in \mathcal{N}(\alpha)} \mu_{i \to \alpha}(\boldsymbol{x}_{\boldsymbol{\alpha}}[i])}$$

2. Normalize beliefs and return the **exact** marginals.

$$\boxed{p_i(x_i) \propto b_i(x_i)} \quad \boxed{p_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}}) \propto b_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}})}$$

It computes the marginals for each variable!

How about this algorithm?

Input: a factor graph with no cycles
Output: exact marginals for each variable and factor

Algorithm:

1. Initialize the messages to the uniform distribution.

$$\boxed{\mu_{i \to \alpha}(x_i) = 1} \quad \boxed{\mu_{\alpha \to i}(x_i) = 1}$$

1. Choose a root node.
2. Send messages from the **leaves** to the **root**.
   Send messages from the **root** to the **leaves**.

$$\boxed{\mu_{i \to \alpha}(x_i) = \prod_{\alpha \in \mathcal{N}(i) \setminus \alpha} \mu_{\alpha \to i}(x_i)} \quad \boxed{\mu_{\alpha \to i}(x_i) = \sum_{\boldsymbol{x}_{\boldsymbol{\alpha}} : \boldsymbol{x}_{\boldsymbol{\alpha}}[i] = x_i} \psi_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}}) \prod_{j \in \mathcal{N}(\alpha) \setminus i} \mu_{j \to \alpha}(\boldsymbol{x}_{\boldsymbol{\alpha}}[i])}$$

1. Compute the beliefs (unnormalized marginals).

$$\boxed{b_i(x_i) = \prod_{\alpha \in \mathcal{N}(i)} \mu_{\alpha \to i}(x_i)} \quad \boxed{b_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}}) = \psi_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}}) \prod_{i \in \mathcal{N}(\alpha)} \mu_{i \to \alpha}(\boldsymbol{x}_{\boldsymbol{\alpha}}[i])}$$

2. Normalize beliefs and return the **exact** marginals.

$$\boxed{p_i(x_i) \propto b_i(x_i)} \quad \boxed{p_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}}) \propto b_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}})}$$

25

It computes the marginals for each variable!

This is **Sum-product belief propagation** from last time

Forward-Backward Algorithm as BP

- The forward-backward algorithm is belief propagation (sequential version)

Alternative to EM: Direct Maximization of the Marginal

$\mathbf{x}$ is a collection of observed RVs

$\mathbf{z}$ is a colleciton of latent RVs

We would like to learn the parameters that maximize the data:

$$
\begin{aligned}
\hat{\theta} &= \operatorname*{argmax}_{\theta} \prod_{i=1}^{N} p(\mathbf{x}_i; \theta) \\
&= \operatorname*{argmax}_{\theta} \prod_{i=1}^{N} \sum_{\mathbf{z}} p(\mathbf{x}_i, \mathbf{z}; \theta)
\end{aligned}
$$

If we can compute $\sum_{\mathbf{z}}$, we can use backprop and gradient ascent to maximize over θ directly. This can work well in practice.