Unsupervised learning in NLP

WSD: do context words naturally cluster?

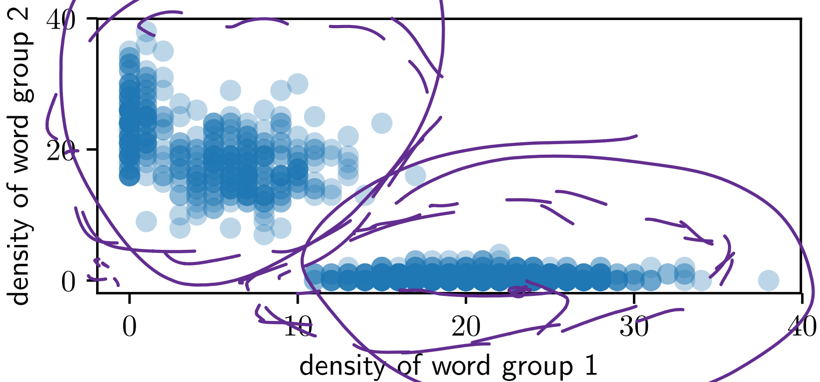


Figure 5.1: Counts of words from two different context groups

"bank"

1. *financial, deposits, credit, lending, capital, markets, regulated, reserve, liquid, assets*
2. *land, water, geography, stream, river, flow, deposits, discharge, channel, ecology*

Unsup. Learning in NLP

- Motivation: there's a LOT more unlabeled than labeled data!
- Do documents or words naturally cluster?
    - WSD: context words cluster around senses
    - Documents: words cluster around topics
- Uses of unsup. NLP
    - 1. Exploratory analysis
    - 2. Unsupervised transfer: usually we have lots of unlabeled data, but little labeled data.
        - Learn language representations (word clusters, embeddings) from unlabeled data, apply to supervised model.

A few methods

- Count-based, no "learning": Word-to-word co-ocurrence in unlabeled data
  - Pointwise mutual information (Church and Hanks 1990)
- Count model-based: EM algorithm to unsupervisedly learn Naive Bayes (related: K-Means for GMMs)
- Gradient-based: word embedding models (next week) and neural language models

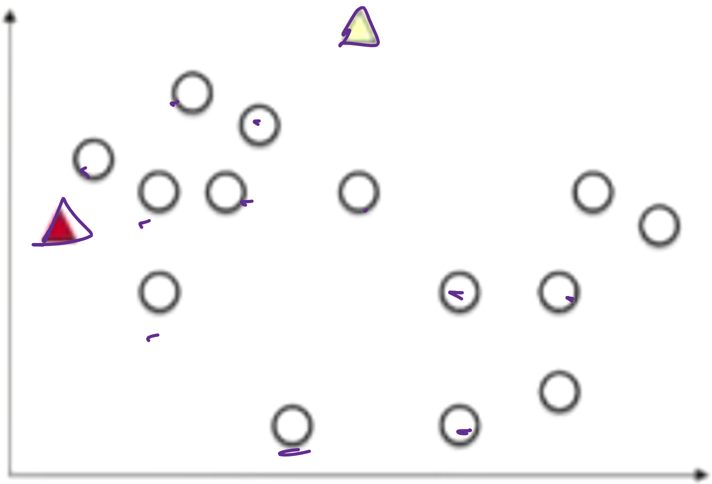
Clustering with (hard) EM

- How to learn a *model* without training data? How about fake it:
  - Initialize: Randomly guess labels
  - ** Learn model parameters as if those labels were true.
  - Make predictions.
  - Go back to ** and iterate.

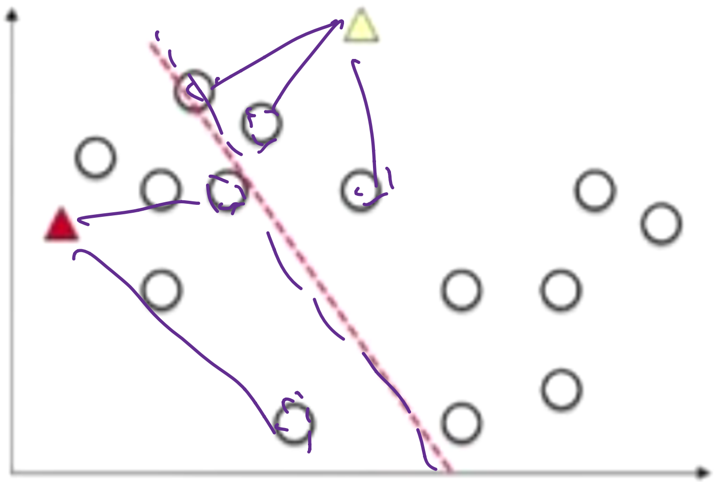
- K-Means is an example for continuous data
  - 1. Randomly initialize cluster centroids
  - 2. Alternate until convergence:
    - ("E"): Assign each example to closest centroid
    - ("M"): Update centroids to means of these newly assigned examples

- K-Means is an instance of a probabilistic unsupervised learning algorithm (Gaussian Mixture Model)

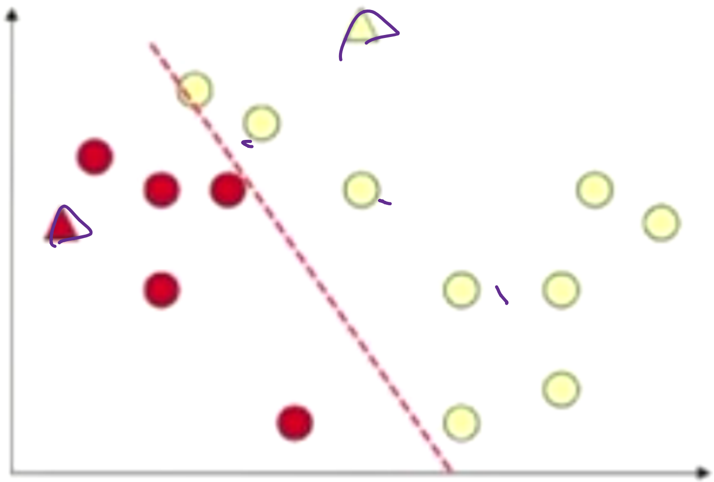
K-means clustering example



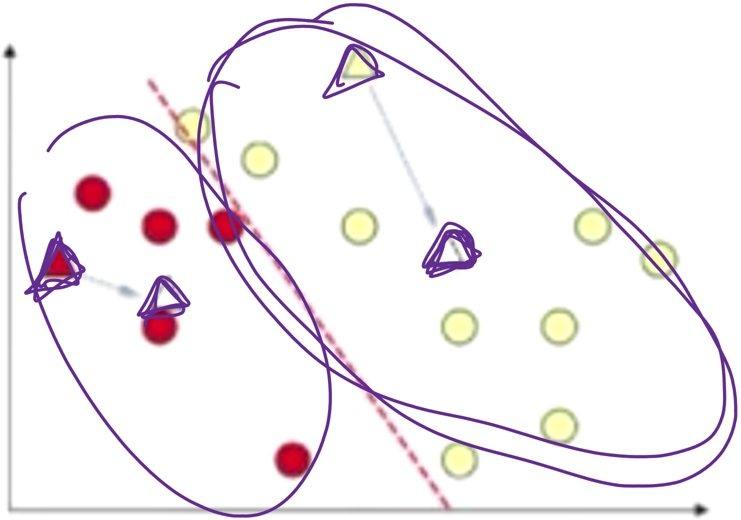
K-means clustering example



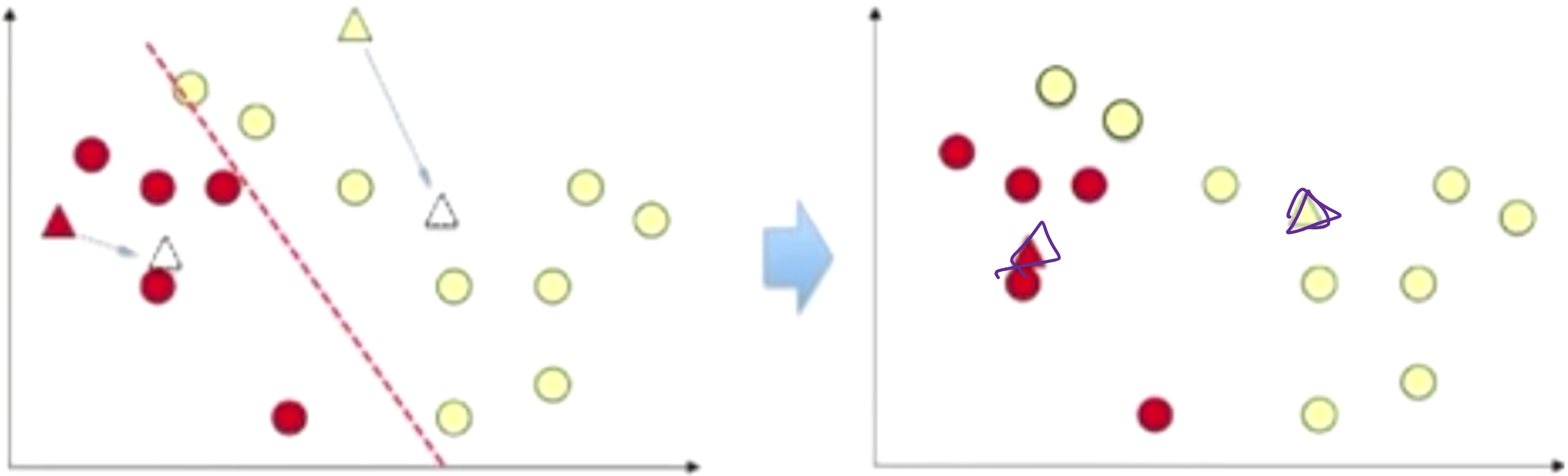
K-means clustering example



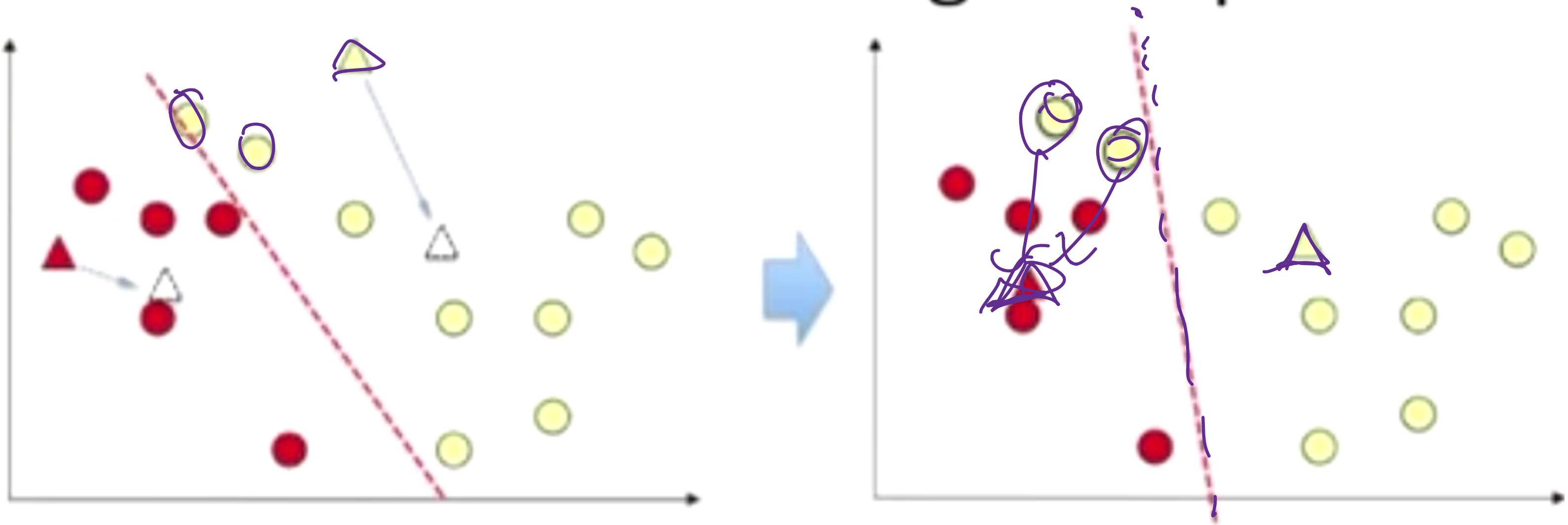
K-means clustering example



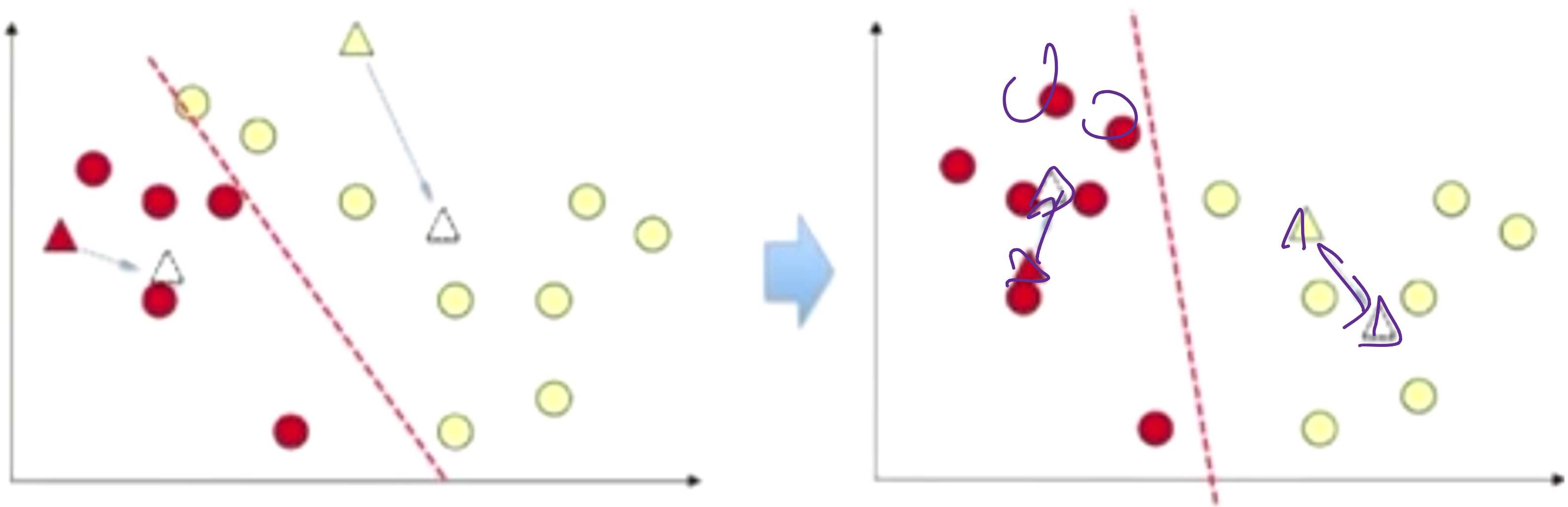
K-means clustering example



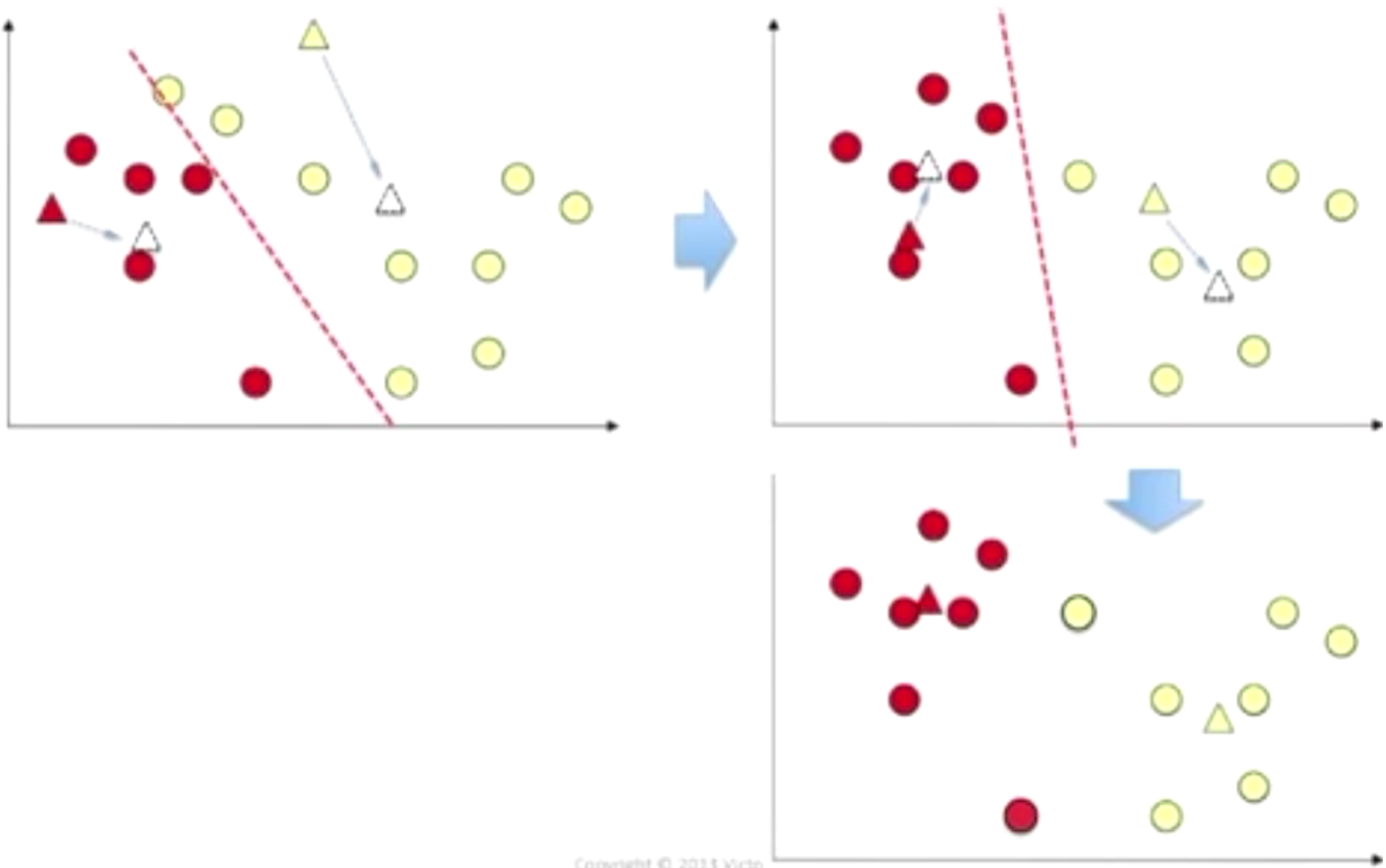
K-means clustering example



K-means clustering example



K-means clustering example



Latent-variable generative models

Text

$$P(w, z \mid \theta)$$

Parameters

(Sometimes) latent quantity to help explain the language you see

- Document category
- World context
- Grammatical category
- Semantic structure
- Real-valued embedding

Easy stuff

- Supervised learning: $\mathbf{argmax_\theta\ P(w^{train}, z^{train} \mid \theta)}$
- Prediction (via posterior inference): $\mathbf{P(z \mid w^{input}, \theta)}$

Unsupervised stuff with *marginal inference*

- Latent (unsupervised) learning: $\mathbf{argmax_\theta\ P(w^{train} \mid \theta)}$
- Language modeling (via marginal inference): $\mathbf{P(w^{input} \mid \theta)}$

Multinomial Naive Bayes

- Parameters
  - $\boldsymbol{\phi_k}$ word distribution for each class $\mathbf{k}$
  - $\boldsymbol{\mu}$ prior distribution over labels
- Generative story. for $P(w,z|\boldsymbol{\mu},\boldsymbol{\phi})$
  For each document *d*:
  - P(z): Draw label $\mathbf{z_d \sim Categ(\mu)}$
  - P(w|z): For t=1,2,...: Draw next word $\mathbf{w_{d,t} \sim}$ $\mathbf{Categ(\phi_z)}$

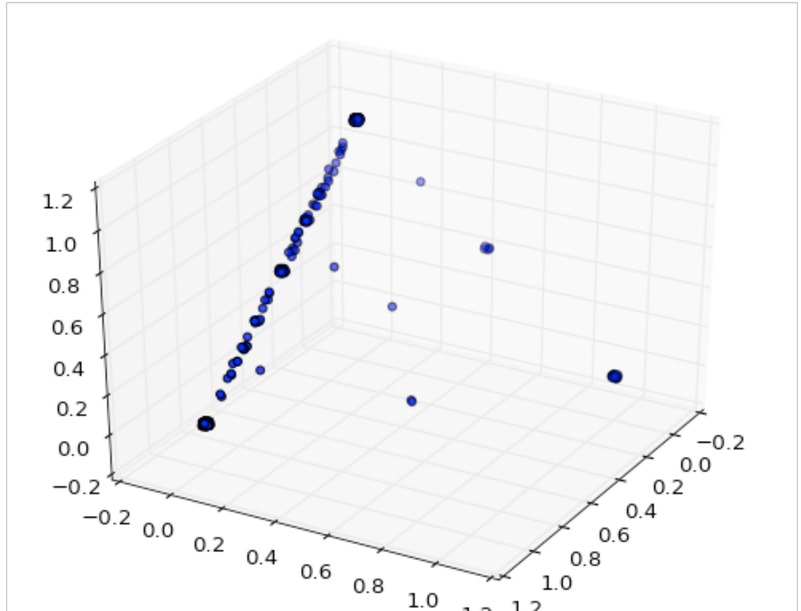
Easy stuff

- Supervised learning: $\mathbf{argmax_\theta\ P(w^{train}, z^{train} \mid \theta)}$
- Prediction (via posterior inference): $\mathbf{P(z \mid w^{input}, \theta)}$

Unsupervised stuff with *marginal inference*

- Latent (unsupervised) learning: $\mathbf{argmax_\theta\ P(w^{train} \mid \theta)}$
- Language modeling (via marginal inference): $\mathbf{P(w^{input} \mid \theta)}$

- Supervised classification with MNB:
    - Training: known (w,z), learn params
    - Testing: fix params, known w, want z
- Unsupervised learning (soft clustering)
    - known w, jointly learn z and params
    - Can learn latent structure in data



1987 NYT data
one point per doc
"congress", "religious", "reagan"
probabilities per doc (normalized)

EM for Unsup. NB

- Iterate
  - (E step): Infer $Q(z) := P(z \mid w, \mu,\phi)$
    - *Predict doc category posterior, from current model*
  - (M step): Learn new
    $\mu,\phi := \text{argmax}_{\mu,\phi}\ E_Q[\log P(w,z \mid \mu,\phi)]$
    - *From **weighted** relative frequency counting!*

EM performance

- Guaranteed to find a locally maximum likelihood solution. Guaranteed to converge.
  - But can take a while
- Dependent on initialization

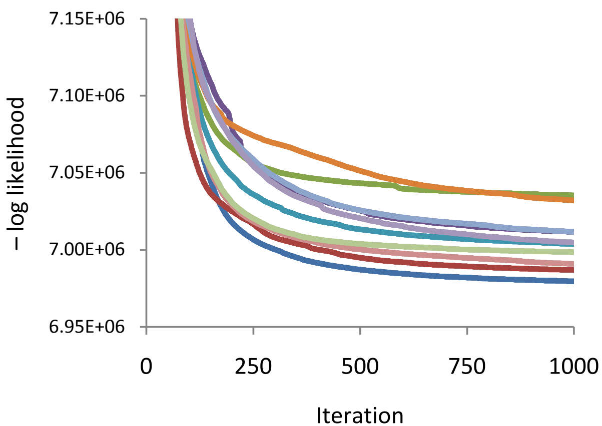


Figure 1: Variation in negative log likelihood with increasing iterations for 10 EM runs from different random starting points.

Johnson 2007, "Why doesn't EM find good HMM POS-taggers?"

EM pros/cons

- Works best for a simple model with rapid E/M-step inference - like Naive Bayes
- Requires probabilistic modeling assumptions
- Dependent on initialization
    - Many alternative methods (e.g. MCMC), but can similar issues with local optima
- EM used for lots in NLP, esp. historically
    - Machine translation
    - HMM-based speech recognition
    - Topic modeling, doc clustering
- At the moment, gradient-based learning for non-probabilistic models (vanilla NNs or matrix factorization) is more common. Note EM and prob. models can be mixed with neural networks (cutting edge research area).