Unsupervised Learning

Part 2

Topics

- How to determine the *K* in *K*-means?
- Hierarchical clustering
- Soft clustering with Gaussian mixture models
- Expectation-Maximization method
- Review for quiz Tuesday

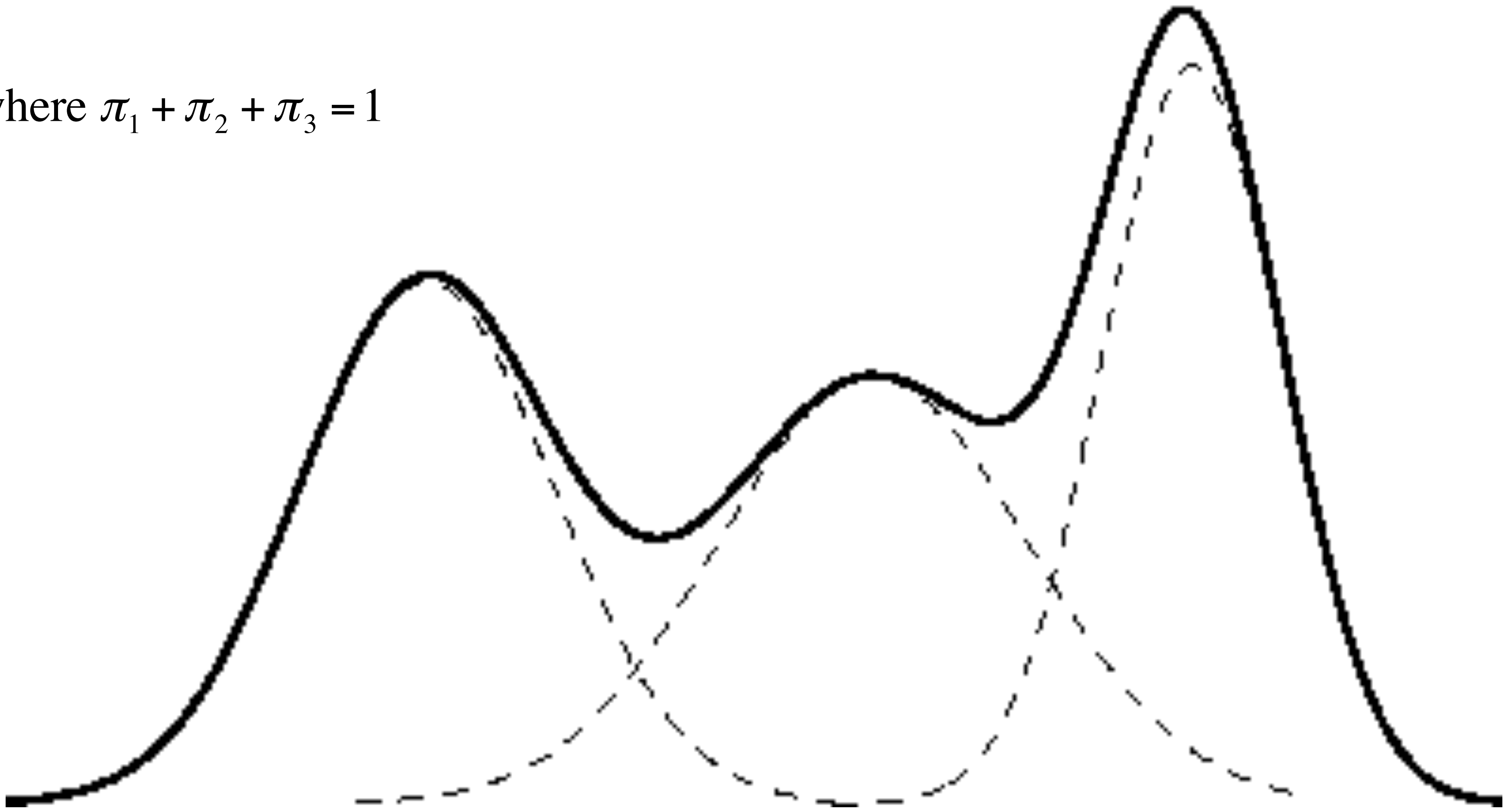
Clustering via finite Gaussian mixture models

- A "soft", generative version of K-means clustering
- **Given:** Training set S = $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, and K.
- **Assumption:** Data is generated by sampling from a "mixture" (linear combination) of K Gaussians.

Mixture of three Gaussians (one dimensional data)

$$p(x) = \pi_1 \mathcal{N}(x \mid \mu_1, \sigma_1) + \pi_2 \mathcal{N}(x \mid \mu_2, \sigma_2) + \pi_3 \mathcal{N}(x \mid \mu_3, \sigma_3)$$

where $\pi_1 + \pi_2 + \pi_3 = 1$



Clustering via finite Gaussian mixture models

- **Clusters:** Each cluster will correspond to a single Gaussian. Each point **x** ∈ S will have some probability distribution over the *K* clusters.

- **Goal:** Given the data, find the Gaussians! (And the coefficients of their sum)

  I.e., Find parameters $\{\theta_K\}$ of these *K* Gaussians such $P(S \mid \{\theta_K\})$ is maximized.

- This is called a **Maximum Likelihood** method.
  - *S* is the data
  - $\{\theta_K\}$ is the "hypothesis" or "model"
  - $P(S \mid \{\theta_K\})$ is the "likelihood".

General form of one-dimensional (univariate) Gaussian Mixture Model

$$p(x) = \sum_{i=1}^{K} \pi_i \mathcal{N}(x \mid \mu_i, \sigma_i)$$

where $\sum_{i=1}^{K} \pi_i = 1$

Simple Case:
Maximum Likelihood for Single Univariate Gaussian

- Assume training set S has *N* values generated by a univariant Gaussian distribution:

$$S = \{x_1, \ldots, x_N\}, \text{ where}$$

$$\mathcal{N}(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- Likelihood function: probability of data given model (or parameters of model)

$$p(S \mid \mu, \sigma) = \prod_{i=1}^{N} \mathcal{N}(x_i \mid \mu, \sigma)$$

- How to estimate parameters μ and σ from S?

- Maximize the likelihood function with respect to μ and σ .

$$\text{Maximize}: \; p(S \mid \mu,\sigma) = \prod_{i=1}^{N} \mathcal{N}(x_i \mid \mu,\sigma)$$

$$= \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- Problem: Individual values of $\mathcal{N}(x_i \mid \mu,\sigma)$ are typically very small. (Can underflow numerical precision of computer.)

- Solution: Work with log likelihood instead of likelihood.

$$\ln p(S \mid \mu,\sigma) = \ln\left(\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}\right)$$

$$= \sum_{i=1}^{N} \ln\left(\frac{e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}}\right) = \sum_{i=1}^{N}\left(\ln\left(e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}\right) - \ln\left(\sqrt{2\pi\sigma^2}\right)\right)$$

$$= -\sum_{i=1}^{N}\left(\frac{(x_i-\mu)^2}{2\sigma^2} + \frac{1}{2}\ln(2\pi) + \frac{1}{2}\ln(\sigma^2)\right)$$

$$= -\frac{1}{2\sigma^2}\sum_{i=1}^{N}(\mathbf{x}_i-\mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi)$$

Now, find maximum likelihood parameters, μ and σ².

First, maximize

$$\ln p(S \mid \mu,\sigma) = -\frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-\mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi)$$

with respect to μ.

$$\frac{d}{d\mu}\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-\mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi)\right]$$

$$= \frac{d}{d\mu}\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-\mu)^2\right]$$

$$= -\frac{1}{2\sigma^2}\left[\sum_{i=1}^{N}-2(x_i-\mu)\right]$$

$$= \frac{1}{\sigma^2}\left[\left(\sum_{i=1}^{N}x_i\right) - N\mu\right] = 0$$

Result:

$$\boxed{\mu_{ML} = \frac{1}{N}\sum_{i=n}^{N}x_n}$$

(ML = “Maximum Likelihood”)

Now, maximize

$$\ln p(S \mid \mu,\sigma) = -\frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-\mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi)$$

with respect to σ^2.

$$\frac{d}{d\sigma^2}\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-\mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi)\right]$$

$$= \frac{d}{d\sigma^2}\left[-\frac{1}{2}\left(\sigma^2\right)^{-1}\sum_{i=1}^{N}(x_i-\mu)^2 - \frac{N}{2}\ln\sigma^2\right]$$

$$= \frac{1}{2}\left(\sigma^2\right)^{-2}\sum_{i=1}^{N}(x_i-\mu)^2 - \frac{N}{2\sigma^2} = \frac{1}{\left(\sigma^2\right)^2}\sum_{i=1}^{N}(x_i-\mu)^2 - \frac{N}{\sigma^2}$$

$$= \frac{\sum_{i=1}^{N}(x_i-\mu)^2 - N\sigma^2}{\left(\sigma^2\right)^2} = 0 \quad \Rightarrow \boxed{\sigma^2_{ML} = \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu_{\mathrm{ML}})^2}$$

- The resulting distribution is called a "generative model" because it can generate new data values.

$$\mathcal{N}(x \mid \mu_{ML}, \sigma_{ML}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu_{ML})^2}{2\sigma_{ML}^2}}$$

- We say that

$$\theta = \{\mu_{ML}, \sigma_{ML}\}$$

parameterizes the model.

- In general, θ is used to denote the (learnable) parameters of a probabilistic model

Finite Gaussian Mixture Models

- Back to finite Gaussian mixture models

- **Assumption:** Data is generated from mixture of *K* Gaussians. Each cluster will correspond to a single Gaussian. Each point **x** will have some probability distribution over the *K* clusters.

Multivariate Gaussian Distribution

Multivariate (D-dimensional):

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

where $\boldsymbol{\mu}$ is a D-dimensional mean vector, $\boldsymbol{\Sigma}$ is a $D \times D$ covariance matrix, and $\left|\boldsymbol{\Sigma}\right|$ is the determinant of $\boldsymbol{\Sigma}$.

- Let S be a set of multivariate data points (vectors):
  $S = \{\mathbf{x}_1, ..., \mathbf{x}_m\}$.

- General expression for finite Gaussian mixture model:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \mu_k, \Sigma_k)$$

- That is, **x** has probability of "membership" in multiple clusters/classes.

More Complicated Case: Maximum Likelihood for Gaussian Mixture Model

- Goal: Given S = $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, and given K, find the Gaussian mixture model (with K Gaussians) for which S has maximum log-likelihood.

- Log likelihood function:

$$\ln P(S \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k) \right)$$

- Given S, we can maximize this function to find

$$\{\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}_{ML}$$

- But no closed form solution (unlike simple case in previous slides)

- In this multivariate case, can most efficiently maximize this function using the "Expectation / Maximization" (EM) algorithm.

Expectation-Maximization (EM) algorithm

- **General idea:**
  - Choose random initial values for means, covariances and mixing coefficients. (Analogous to choosing random initial cluster centers in *K*-means.)

  - Alternate between E (expectation) and M (maximization) step:

    - **E step:** use current values for parameters to evaluate posterior probabilities, or "responsibilities", for each data point. (Analogous to determining which cluster a point belongs to, in *K*-means.)

    - **M step:** Use these probabilities to re-estimate means, covariances, and mixing coefficients. (Analogous to moving the cluster centers to the means of their members, in *K*-means.)

  Repeat until the log-likelihood or the parameters θ do not change significantly.

More detailed version of EM algorithm

1. Let **X** be the set of training data. Initialize the means μ$_k$, covariances Σ$_k$, and mixing coefficients π$_k$, and evaluate initial value of log likelihood.

$$\ln p(\mathbf{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

2. **E step**. Evaluate the "responsibilities" using the current parameter values

$$r_{n,k} = \frac{\boldsymbol{\pi}_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \boldsymbol{\pi}_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

where $r_{n,k}$ denotes the "responsibilities" of the *k*th cluster for the *n*th data point.

3. **M step.** Re-estimate the parameters θ using the current responsibilities.

$$\boldsymbol{\mu}_k^{new} = \frac{1}{\sum_{n=1}^{N} r_{n,k}} \sum_{n=1}^{N} \left(r_{n,k}\, \mathbf{x}_n \right)$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{1}{\sum_{n=1}^{N} r_{n,k}} \sum_{n=1}^{N} \left(r_{n,k}\, (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T \right)$$

$$\boldsymbol{\pi}_k^{new} = \frac{\sum_{n=1}^{N} r_{n,k}}{N}$$

4. Evaluate the log likelihood with the new parameters

$$\ln p(\mathbf{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

and check for convergence of either the parameters or the log likelihood. If not converged, return to step 2.

- EM much more computationally expensive than *k-means*

- **Common practice:** Use *k*-means to set initial parameters, then improve with EM.

-
  - **Initial means:** Means of clusters found by *k*-means

  - **Initial covariances:** Sample covariances of the clusters found by *k*-means algorithm.

  - **Initial mixture coefficients:** Fractions of data points assigned to the respective clusters.

- Can prove that EM finds local maxima of log-likelihood function.

- EM is very general technique for finding maximum-likelihood solutions for probabilistic models

Case Study:

Text classification from labeled and unlabeled documents using EM

(Nigam et al., *Machine Learning*, 2000)

- Method for combining small amount of labeled data with large amount of unlabeled data to increase classification accuracy

- Unlabeled text data: can provide information about joint probability distribution over words
  - E.g., suppose "homework" indicates positive class
  - Suppose "homework" appears frequently with "lecture" in unlabeled data
  - Then increase probability that "lecture" (on its own) indicates positive class

Overall approach

- Train classifier using labeled documents
- Use classifier to assign probabilistically weighted class labels to unlabeled documents
- Then train new classifier using all the documents
- Iterate

Text classification from labeled and unlabeled documents using EM

K. Nigam et al., *Machine Learning*, 2000

- Big problem with text classification: need labeled data.
- What we have: lots of unlabeled data.
- Question of this paper: Can unlabeled data be used to increase classification accuracy?
- I.e.: Any information implicit in unlabeled data? Any way to take advantage of this implicit information?

General idea: A version of EM algorithm

- Train a classifier with small set of available labeled documents.

- Use this classifier to assign probabilisitically-weighted class labels to unlabeled documents by calculating expectation of missing class labels.

- Then train a new classifier using all the documents, both originally labeled and formerly unlabeled.

- Iterate.

Probabilistic framework

- Assumes data are generated with Gaussian mixture model
- Assumes one-to-one correspondence between mixture components and classes.
- “These assumptions rarely hold in real-world text data”

Probabilistic framework

Let $C = \{c_1, ..., c_K\}$ be the classes / mixture components

Let $\theta = \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_K\} \cup \{\boldsymbol{\Sigma}_1, ..., \boldsymbol{\Sigma}_K\} \cup \{\pi_1, ..., \pi_K\}$ be the mixture parameters.

Assumptions: A document d_i is created by first selecting a mixture component according to the mixture weights π_j, then having this selected mixture component generate a document according to its own parameters, with distribution

$$p(d_i \mid c_j; \theta).$$

- Likelihood of document d_i:

$$p(d_i \mid \theta) = \sum_{j=1}^{k} \pi_k p(d_i \mid c_j; \theta)$$

- Now, we will apply EM to a naive Bayes Classifier

  Recall naive Bayes classifier: Assume each attribute is conditionally independent, given c_j.

$$\text{Let } \mathbf{x} = (a_1, a_2, \ldots, a_D)$$

We have:

$$p(a_1, a_2, \ldots, a_D \mid c_j) = p(a_1 \mid c_j)\, p(a_2 \mid c_j) \cdots p(a_D \mid c_j)$$

$$p(c_j \mid \mathbf{x}) = p(c_j) \prod_i p(a_i \mid c_j),\ i = 1, \ldots, D;\ j = 1, \ldots, K$$

To “train” naive Bayes from labeled data, estimate

$$p(c_j) \text{ and } p(a_i \mid c_j),\ j = 1,\ldots,K;\ i = 1,\ldots,D$$

These values are estimates of the parameters in θ. Call these values $\hat{\theta}$.

Note that Naive Bayes can be thought of as a generative mixture model.

Document d_i is represented as a vector of word frequencies (w_1, ..., $w_{|V|}$), where V is the vocabulary (all known words).

There is an assumed probability distribution over words associated with each class, parameterized by θ.

We need to estimate $\hat{\theta}$ to determine what probability distribution document $d_i = (w_1, ..., w_{|V|})$is most likely to come from.

Applying EM to Naive Bayes

- We have a small number of labeled documents S_{labeled}, and a large number of unlabeled documents, $S_{\text{unlabeled}}$.

- The initial parameters $\hat{\theta}$ are estimated from the labeled documents S_{labeled}.

- **Expectation step:** The resulting classifier is used to assign probabilistically-weighted class labels $p(c_j \mid \mathbf{x})$ to each unlabeled document $\mathbf{x} \in S_{\text{unlabeled}}$.

- **Maximization step:** Re-estimate $\hat{\theta}$ using $p(c_j \mid \mathbf{x})$ values for $\mathbf{x} \in S_{\text{unlabeled}} \cup S_{\text{unlabeled}}$

- Repeat until $p(c_j \mid \mathbf{x})$ or $\hat{\theta}$ has converged.

Augmenting EM

What if basic assumptions (each document generated by one component; one-to-one mapping between components and classes) do not hold?

They tried two things to deal with this:

(1) Weighting unlabeled data less than labeled data

(2) Allow multiple mixture components per class:

A document may be comprised of several different sub-topics, each best captured with a different word distribution.

Data

- 20 UseNet newsgroups
- Web pages (WebKB)
- Newswire articles (Reuters)

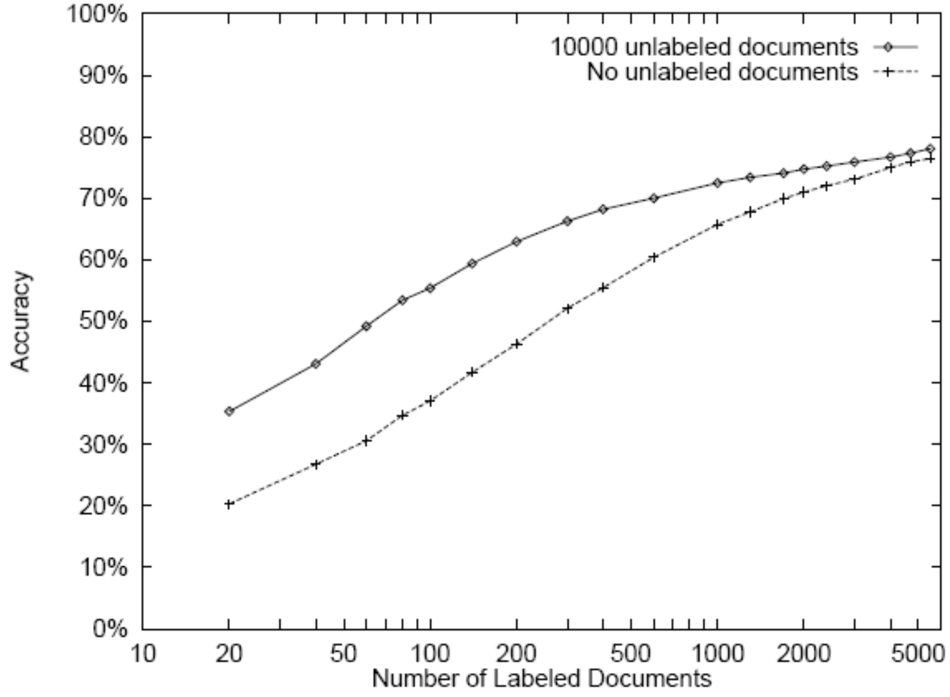


Figure 2. Classification accuracy on the 20 Newsgroups data set, both with and without 10,000 unlabeled documents. With small amounts of training data, using EM yields more accurate classifiers. With large amounts of labeled training data, accurate parameter estimates can be obtained without the use of unlabeled data, and the two methods begin to converge.

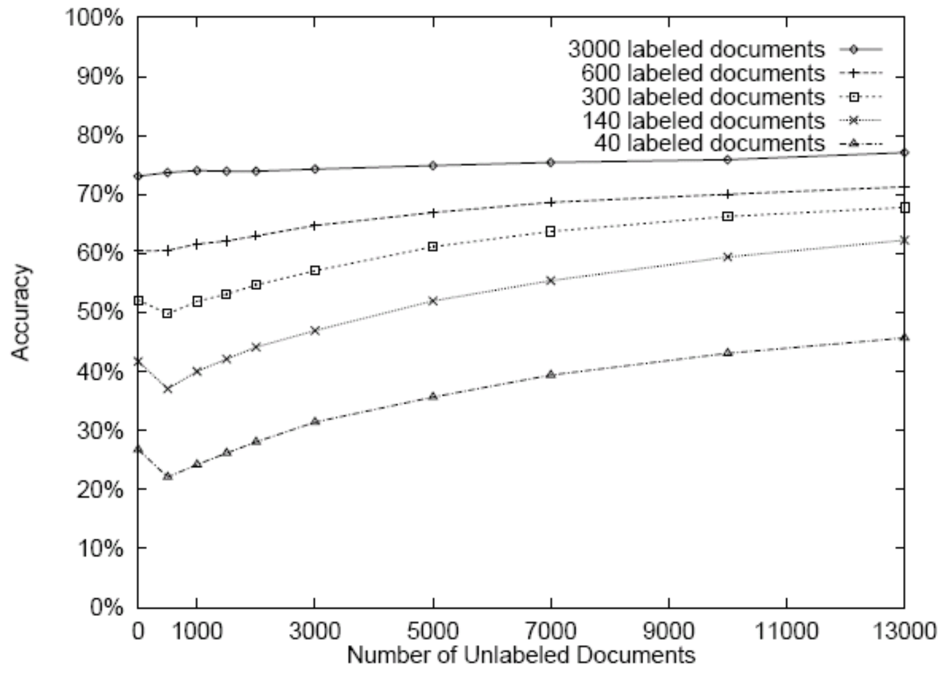


Figure 3. Classification accuracy while varying the number of unlabeled documents. The effect is shown on the 20 Newsgroups data set, with 5 different amounts of labeled documents, by varying the amount of unlabeled data on the horizontal axis. Having more unlabeled data helps. Note the dip in accuracy when a small amount of unlabeled data is added to a small amount of labeled data. We hypothesize that this is caused by extreme, almost 0 or 1, estimates of component membership, $\mathrm{P}(c_j|d_i,\hat{\theta})$, for the unlabeled documents (as caused by naive Bayes' word independence assumption).

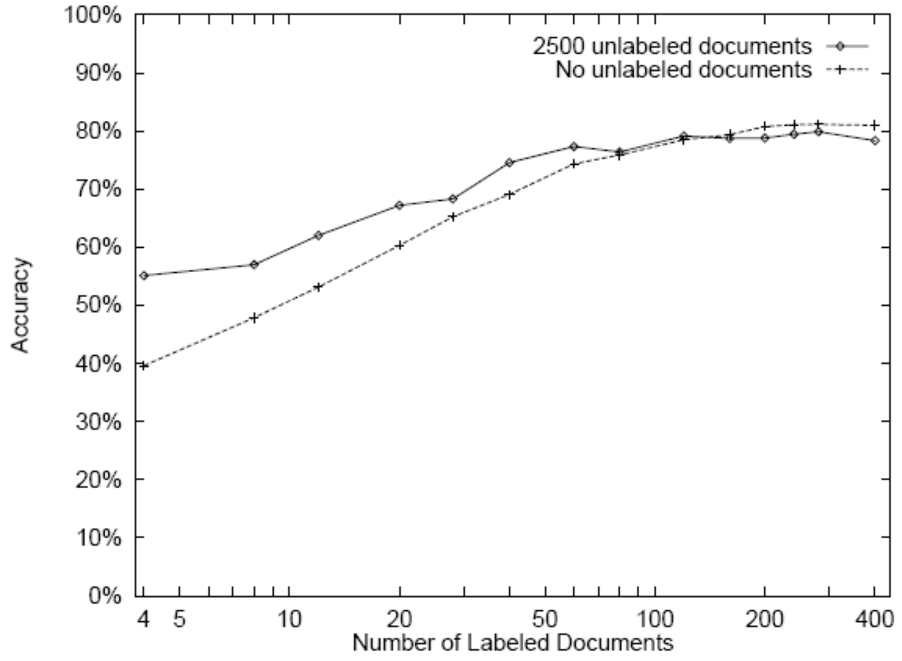


Figure 4. Classification accuracy on the WebKB data set, both with and without 2500 unlabeled documents. When there are small numbers of labeled documents, EM improves accuracy. When there are many labeled documents, however, EM degrades performance slightly—indicating a misfit between the data and the assumed generative model.

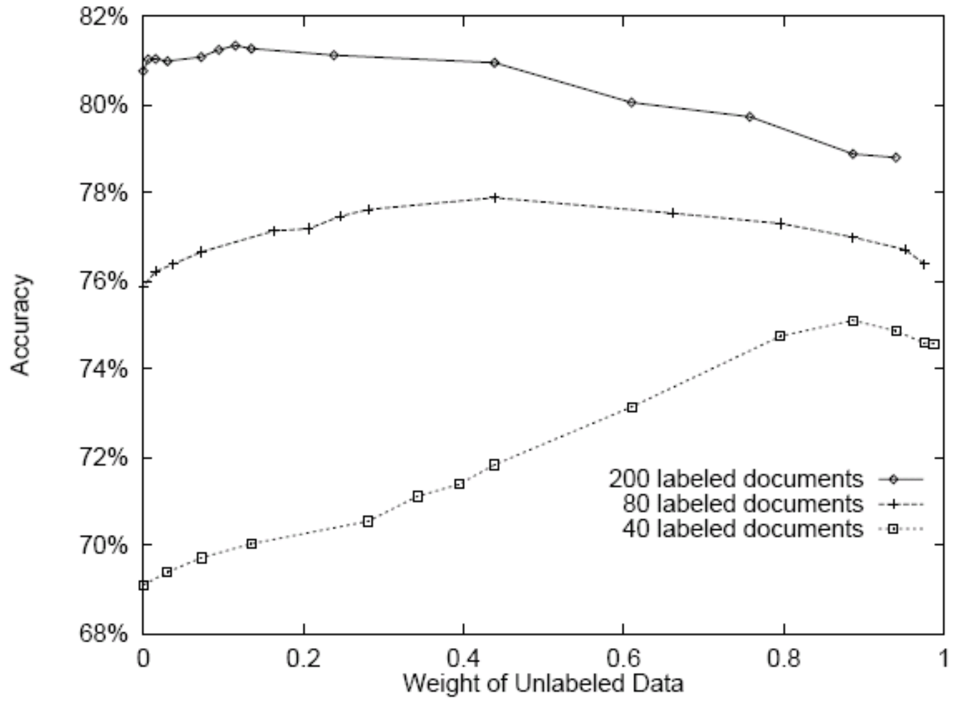


Figure 5. The effects of varying λ, the weighting factor on the unlabeled data in EM-λ. These three curves from the WebKB data set correspond to three different amounts of labeled data. When there is less labeled data, accuracy is highest when more weight is given to the unlabeled data. When the amount of labeled data is large, accurate parameter estimates are attainable from the labeled data alone, and the unlabeled data should receive less weight. With moderate amounts of labeled data, accuracy is better in the middle than at either extreme. Note the magnified vertical scale.

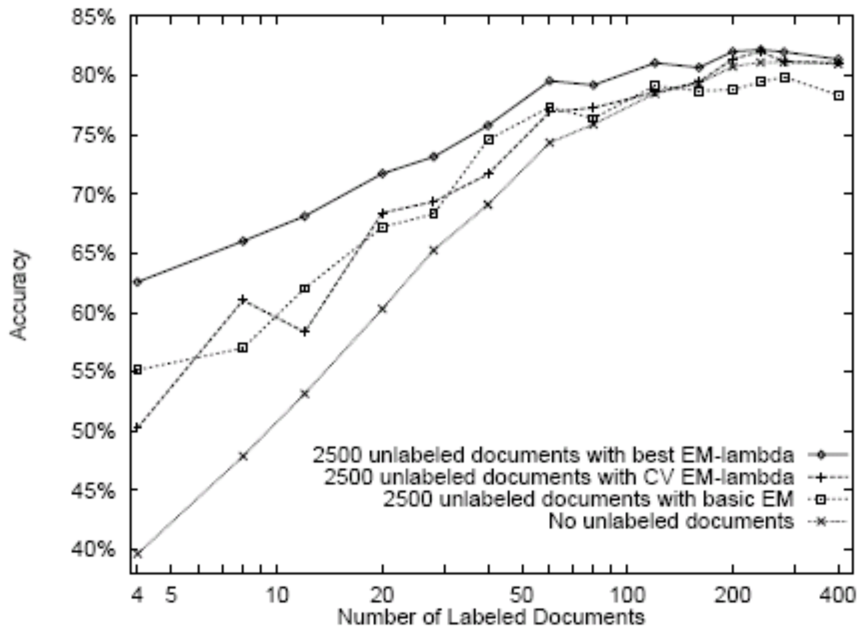


Figure 6. Classification accuracy on the WebKB data set, with modulation of the unlabeled data by the weighting factor λ. The top curve shows accuracy when using the best value of λ. In the second curve, λ is chosen by cross-validation. With small amounts of labeled data, the results are similar to basic EM; with large amounts of labeled data, the results are more accurate than basic EM. Thanks to the weighting factor, large amounts of unlabeled data no longer degrades accuracy, as it did in Figure 4, and yet the algorithm retains the large improvements with small amounts of labeled data. Note the magnified vertical axis to facilitate the comparisons.

Table 4. Precision-recall breakeven points showing performance of binary classifiers on Reuters with traditional naive Bayes (NB1), multiple mixture components using just labeled data (NB*), basic EM (EM1) with labeled and unlabeled data, and multiple mixture components EM with labeled and unlabeled data (EM*). For NB* and EM*, the number of components is selected optimally for each trial, and the median number of components across the trials used for the negative class is shown in parentheses. Note that the multi-component model is more natural for Reuters, where the negative class consists of many topics. Using both unlabeled data and multiple mixture components per class increases performance over either alone, and over naive Bayes.

Category	NB1	NB*	EM1	EM*	EM* vs NB1	EM* vs NB*
acq	69.4	74.3 (4)	70.7	83.9 (10)	+14.5	+9.6
corn	44.3	47.8 (3)	44.6	52.8 (5)	+8.5	+5.0
crude	65.2	68.3 (2)	68.2	75.4 (8)	+10.2	+7.1
earn	91.1	91.6 (1)	89.2	89.2 (1)	-1.9	-2.4
grain	65.7	66.6 (2)	67.0	72.3 (8)	+6.3	+5.7
interest	44.4	54.9 (5)	36.8	52.3 (5)	+7.9	-2.6
money-fx	49.4	55.3 (15)	40.3	56.9 (10)	+7.5	+1.6
ship	44.3	51.2 (4)	34.1	52.5 (7)	+8.2	+1.3
trade	57.7	61.3 (3)	56.1	61.8 (3)	+4.1	+0.5
wheat	56.0	67.4 (10)	52.9	67.8 (10)	+11.8	+0.4