CSE 5523: Lecture Notes 22
Expectation maximization

Contents

22.1 Expectation maximization [Dempster et al., 1977] 1
22.2 Sample EM code 2
22.3 Continuous observations (Gaussian mixture model) 3
22.4 Evaluation of unsupervised models 4

Sometimes we have unlabeled data and want to divide it into classes that statistically explain it.

For example, heights and weights of animals on a farm can be explained using a set of species.

Message passing can help discover classes that maximize posterior probability of unlabeled data.

22.1 Expectation maximization [Dempster et al., 1977]

Optimizing parameters $\mathbf{M}$ *and* missing data labels X isn't closed-form or gradient solvable.

But, we can start with random $\mathbf{M}^{(0)}$'s and iterate solving for $X^{(0)}$'s, then $\mathbf{M}^{(1)}$'s, then $X^{(1)}$'s, etc.

Assume N training examples, each with V variables $X_{n,v}$, only some of which are observed.

(And remember C_v are conditioned-on variables, $\mathbf{f}_{v,u}$ and $\mathbf{b}_{v,w}$ are forward and backward messages.)

Randomly initialize distributions for random variables X_v over $|X_v|$ values for $\prod_{X_u \in C_v} |X_u|$ cases:

$$\mathbf{M}_v^{(0)} \sim \text{Dirichlet}(\mathbf{1}^{(\prod_{X_u \in C_v} |X_u|) \times |X_v|})$$

Then for several iterations i, calculate **expected** distributions $\left(\mathbf{x}_{n,v}^{(i)}\right)^\top$ over each hidden variable X_v:

$$\left(\mathbf{x}_{n,v}^{(i)}\right)^\top = \underbrace{\left(\bigotimes_{X_u \in C_v} \left(\mathbf{f}_{n,u,v}^{(i)}\right)^\top\right)}_{\text{joint of conditioned-on variables}} \mathbf{M}_v^{(i-1)} \bigodot_{w | X_v \in C_w} \text{diag}(\mathbf{b}_{n,w,v}^{(i)})$$

This is called an **expectation step** or **E step**.

Then calculate the **maximum** a posteriori estimate of the model $\mathbf{M}_v^{(i)}$ for each variable X_v:

$$\mathbf{M}_v^{(i)} = \sum_n \left(\bigotimes_{X_u \in C_v} \mathbf{x}_{n,u}^{(i)}\right) \left(\mathbf{x}_{n,v}^{(i)}\right)^\top$$

and normalize so rows sum to one. Tied or stationary models are summed over all tied instances.

This is called a **maximization step** or **M step**.

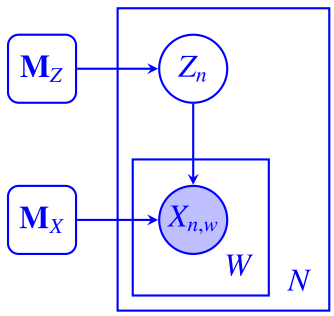
This algorithm is called **expectation maximization (EM)**.

It is guaranteed to converge on a ***local*** maximum: both E and M step decrease KL divergence.

22.2 Sample EM code

Here's example code where one of K 'topics' is chosen for each of N W-word documents.

This fits parameters and hidden variable values for the following plate diagram:



(NOTE: here each backward message $\mathbf{b}_{n,X,Z}$ is a *product* of backward messages from X's to Z.)

```
import sys
import numpy as np
import pandas as pd

X = pd.read_csv( sys.argv[1], sep=' ' )                                 ## read data
N = len(X)                                                              ## number of documents
W = len(X.columns)                                                      ## doc length in words
V = np.unique(X)                                                        ## vocab of word types
K = 2                                                                   ## number of topics

M_Z = pd.DataFrame( np.random.dirichlet( np.ones( K ) ) ).T            ## initialize models
M_X = pd.DataFrame( np.random.dirichlet( np.ones( len(V) ), K ), columns=V )

xT = {}                                                                 ## word Kronecker deltas
for n in range(N):                                                      ## for each document
  for w in X:                                                           ## for each word token
    xT[n,w] = pd.DataFrame( np.zeros((1,len(V))), columns=V )
    xT[n,w][ X[w][n] ] += 1                                             ## one-hots w. std cols

for i in range(3):                                                      ## for each EM iter

  b_XZ = [ np.multiply.reduce( [ M_X @ xT[n,w].T for w in X ] )        ## backward messages
           for n in range(N) ]

  zT = {}                                                               ## E step, update vars
  for n in range(N):                                                    ## for each document
    d = M_Z @ pd.DataFrame( np.diagflat(b_XZ[n]), index=range(K), columns=range(K) )
```

```
    zT[n] = d / (d @ np.ones((K,K)) )

  M_Z = pd.DataFrame( np.add.reduce( [ zT[n]/N for n in range(N) ] ), ## M step, update mdls
                      columns=range(K) )
  M_X = pd.DataFrame( np.add.reduce( [ zT[n].T @ xT[n,w] for n in range(N) for w in X ] ),
                      columns=V )
  M_X = M_X / ( M_X @ pd.DataFrame( np.ones((len(V),len(V))), index=V, columns=V ) )

  print( M_Z )
  print( M_X )
```

Run on simple set of ‘documents’, each with three words:

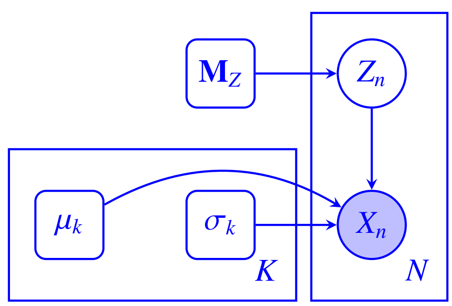
```
x1 x2 x3
a b a
c b c
b a a
```

It correctly identifies word distributions for the different topics:

```
          0         1
0  0.666667  0.333333
              a         b             c
0  6.666667e-01  0.333333  1.517833e-09
1  5.677142e-08  0.333333  6.666666e-01
```

22.3 Continuous observations (Gaussian mixture model)

EM can also model continuous downstream observations (e.g. mixtures of Gaussians):



Here each observation X_n is drawn from a mixture Z_n of K different Gaussian components.

In this case the backward message still contains a likelihood of child values for each parent value:

$$(\mathbf{b}_{n,X,Z})_{[k]} = \mathcal{N}_{\mu_k,\sigma_k}(x_n)$$

and the M step still sets the model’s parameters weighted by the forward message:

$$\mu_k = \frac{1}{N} \sum_n (\mathbf{x}_{n,Z})_{[k]} \, x_n$$

$$\sigma_k = \frac{1}{N} \sum_n (\mathbf{x}_{n,Z})_{[k]} \, (x_n - \mu_k)^2$$

22.4 Evaluation of unsupervised models

Unsupervised models produce arbitrarily labeled '**clusters**' (estimates for categorical variables).

We typically evaluate these against human-labeled '**classes**' using information-theoretic measures:

$$\text{Homogeneity}(z, \hat{z}) = 1 - \frac{\mathsf{H}(z \mid \hat{z})}{\mathsf{H}(z)} = 1 - \frac{\sum_c \sum_k \overbrace{\mathsf{P}_{z_{1..N},\hat{z}_{1..N}}(z{=}c, \hat{z}{=}k) \log_2 \mathsf{P}_{z_{1..N},\hat{z}_{1..N}}(z{=}c \mid \hat{z}{=}k)}^{\text{conditional entropy of predicting classes from clusters}}}{\sum_c \underbrace{\mathsf{P}_{z_{1..N},\hat{z}_{1..N}}(z{=}c) \log_2 \mathsf{P}_{z_{1..N},\hat{z}_{1..N}}(z{=}c)}_{\text{entropy of predicting classes}}}$$

$$\text{Completeness}(z, \hat{z}) = 1 - \frac{\mathsf{H}(\hat{z} \mid z)}{\mathsf{H}(\hat{z})} = 1 - \frac{\sum_k \sum_c \overbrace{\mathsf{P}_{z_{1..N},\hat{z}_{1..N}}(\hat{z}{=}k, z{=}c) \log_2 \mathsf{P}_{z_{1..N},\hat{z}_{1..N}}(\hat{z}{=}k \mid z{=}c)}^{\text{conditional entropy of predicting clusters from classes}}}{\sum_k \underbrace{\mathsf{P}_{z_{1..N},\hat{z}_{1..N}}(\hat{z}{=}k) \log_2 \mathsf{P}_{z_{1..N},\hat{z}_{1..N}}(\hat{z}{=}k)}_{\text{entropy of predicting clusters}}}$$

$$\text{V-measure}(z, \hat{z}) = \frac{2\,\text{Homogeneity}(z, \hat{z})\,\text{Completeness}(z, \hat{z})}{\text{Homogeneity}(z, \hat{z}) + \text{Completeness}(z, \hat{z})}$$

It's the log of the accuracy we'd get if we trained a statistical classifier on some held-out data.

But how can we calculate significance for these aggregate measures using permutation testing?

These can be permutation tested by accounting per-instance 'heterogeneity' and 'incompleteness':

$$\mathsf{PIH}(n, z_{1..N}, \hat{z}_{1..N}) = \frac{\log_2 \mathsf{P}_{z_{1..N},\hat{z}_{1..N}}(z_n \mid \hat{z}_n)}{\sum_{n'} \log_2 \mathsf{P}_{z_{1..N},\hat{z}_{1..N}}(z_{n'} \mid \hat{z}_{n'})}$$

$$\mathsf{PII}(n, z_{1..N}, \hat{z}_{1..N}) = \frac{\log_2 \mathsf{P}_{z_{1..N},\hat{z}_{1..N}}(\hat{z}_n \mid z_n)}{\sum_{n'} \log_2 \mathsf{P}_{z_{1..N},\hat{z}_{1..N}}(\hat{z}_{n'} \mid z_{n'})}$$

These are then summed over the set of items in each permutation, and subtracted from one.

References

[Dempster et al., 1977] Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39 (Series B):1–38.