COS 424
Homework #3
Due Tuesday, March 30th

See the course website for important information about collaboration and late policies, as well as where and when to turn in assignments.

Program and data files

This homework uses again the Reuters21578 dataset. Please download the file "`hw3data.tar.gz`" which is available from `http://www.cs.princeton.edu/courses/archive/spring10/cos424/w/hw3`. This archive contains the following files:

- `reuters_train.txt`, `reuters_test.txt` contain the preprocessed training and testing sets. Although we still use the "ModApte" split, these files are not exactly the same as those you were supposed to prepare during homework 2. Each line represents a single document and contains the numerical document ID followed by a sparse vector containing the number of occurences of each term in each document. These counts are not normalized.
- `reuters_dict.txt` provides the association between term index and the term themselves. Each line contains the term index, the term string, the total count of occurrences of this term in the training set, and the number of training documents containing the term. We have discarded all terms that were less than four characters long or appeared in less than three training documents. The total number of terms should therefore be slightly lower than the number of terms in the homework 2 files.
- A couple C++ files and a `Makefile` contain the almost complete implementation of two programs named `umix` and `humix`. The main source files are `umix.cpp` and `humix.cpp`. Part of the homework consists of completing and running these programs.
- The ancillary files `vector.h`, `vector.cpp`, `util.h`, `util.cpp`, and `wrapper.h` contain utility routines that you do not need to modify. The appendix of this document provides a brief description of the vector classes implemented by files `vector.h` and `vector.cpp`.
- File `homework.tex` contains the LaTeX source of this document.

Part 1 – Unigram mixture model

We wish to group the Reuters documents into semantically meaningful clusters that we call *topics*. We plan to achieve this using a mixture model whose k components represent k distinct topics. Let random variables X and Y represent a document and its topic. The unigram mixture model can then be written as

$$\begin{aligned} P_\theta(X) &= \textstyle\sum_{y=1}^{k} P_\theta(Y=y) P_\theta(X|Y=y)\,, \\ P_\theta(Y=y) &= \lambda_y \\ P_\theta(X = t_1 t_2 \ldots t_L | Y=y) &= \beta_L \, p_{y t_1} \, p_{y t_2} \, \cdots \, p_{y t_L}\,. \end{aligned}$$

In the above expressions, $t_1 t_2 \ldots t_L$ represents the sequence of terms in the document X. The parameters θ of the model are (i) the prior probabilities λ_y of each component, (ii) the probabilities β_L that a document contains L terms, and (iii) the probability p_{yt} of occurrence of term t for component y. These parameters satisfy the normalization constraints $\sum_{y=1}^{k} \lambda_y = 1$, $\sum_{L=1}^{\infty} \beta_L = 1$, and $\forall y \ \sum_t p_{yt} = 1$. This model makes the assumption that the document length does not depend on the document topic because we use the same parameters β_L for all components. We do this because our model would too easily cluster documents by length instead of by content.

Question 1a Show that the log likelihood $\log L(\theta) = \sum_i \log P_\theta(X = x_i)$ can be written as the sum of an expression that depends only on the β_L parameters and an expression that depends only on the λ_y and p_{yt} parameters.

A simple derivation answers the question.

$$\begin{aligned}
\log L(\theta) &= \sum_{i=1}^{n} \log P_\theta(X = x_i) = \sum_{i=1}^{n}\sum_{y=1}^{k} \log P_\theta(Y = y)\, P_\theta(X = x_i | Y = y) \\
&= \sum_{i=1}^{n} \log \left(\sum_{y=1}^{k} \lambda_y \, \beta_{L_i} \, p_{yt_1} \, p_{yt_2} \, \cdots \, p_{yt_{L_i}} \right) \\
&= \sum_{i=1}^{n} \log \left(\beta_{L_i} \sum_{y=1}^{k} \lambda_y \, p_{yt_1} \, p_{yt_2} \, \cdots \, p_{yt_{L_i}} \right) \\
&= \sum_{i=1}^{n} \log \beta_{L_i} + \sum_{i=1}^{n} \log \left(\sum_{y=1}^{k} \lambda_y \prod_t p_{yt}^{n_{it}} \right)
\end{aligned}$$

These two parts of the log likelihood can be optimized separately. Since we are not interested in modeling the document length, we can simply ignore the β_L and simply write

$$P_\theta(X = x_i | Y = y) = \prod_t p_{yt}^{n_{it}} ,$$

where n_{it} represents the number of occurrences of term t in document x_i.

The provided C++ program `umix` maximizes the likelihood of this model using the Expectation Maximization algorithm. The main source code `umix.cpp` contains numerous comments. However there is a critical expression missing in the middle of the M step of the EM algorithm. The missing expression is clearly marked with text `INSERT_CORRECT_CODE_HERE`.

Question 1b Complete the missing expression in the M step. Provide a listing of that segment of the code showing your completion.

```
for (int i=0; i<n; i++)          // loop over examples
{
  double qij = qj[i];            // cache qij
  for (const SVector::Pair *p = docs[i].data; p->i >= 0; p++)
    c.p[p->i] += qij * p->v;
}
```

Here is the program output for computing a mixture with 5 components:

```
$ ./umix reuters_train.txt reuters_dict.txt 5 reuters_clus5.txt
Read 9603 documents
Read 7303 terms (maxid 7303)
Initializing 5 components
```

```
Pass 66 logl=-4.90179e+06
Saving components
0.339047  share offer issu corp debentur common dividend stock compani qtly
0.241021  loss profit billion year oper januari quarter februari rose from
0.186173  tonn wheat export market grain coffe crop usda sugar quota
0.137562  bank trade that brazil countri japan debt minist japanes foreign
0.0961973  mine barrel gold ship refineri crude said compani feet plant
```

The output summarizes each component on a line displaying its λ_y coefficient and its ten most distinctive terms.

Question 1c Run `umix` on the Reuters data for mixtures of 15 and 30 components. Can you recognize topics corresponding to some of the manual labels provided with the Reuters21578 dataset?

Here are the outputs of `umix` on 15 and 30 components.

```
$ ./umix reuters_train.txt reuters_dict.txt 15 reuters_clus15.txt
Read 9603 documents
Read 7303 terms (maxid 7303)
Initializing 15 components
Pass 142 logl=-4.80869e+06
Saving components
0.170492  offer share issu bond debentur will said manag moodi debt
0.149343  loss profit oper year note gain includ reuter exclud sale
0.11302  said compani earn mine will quarter gold barrel ounc expect
0.0968848  billion bank januari februari monei rose bill rate rise fell
0.0893807  dollar bank that said rate opec currenc market econom economi
0.0872163  said merger acquir compani corp acquisit complet agreement share approv
0.0850336  bank that debt brazil trade countri japan said offici japanes
0.0832011  qtly dividend record april prior split march payout quarterli payabl
0.0711536  coffe produc export tonn wheat grain agricultur crop usda that
0.0535016  tonn wheat export sugar grain shipment import corn maiz soybean
0.000773633  bundesbank unchang elev reagan polici metromedia lombard answer enquiri poll
0  ardeshir avaj transship cigra shex nowruz subproduct peavei laydai
0  ardeshir avaj transship cigra shex nowruz subproduct peavei laydai
0  ardeshir avaj transship cigra shex nowruz subproduct peavei laydai
0  ardeshir avaj transship cigra shex nowruz subproduct peavei laydai

$ ./umix reuters_train.txt reuters_dict.txt 30 reuters_clus30.txt
Read 9603 documents
Read 7303 terms (maxid 7303)
Initializing 30 components
Pass 58 logl=-4.80749e+06
Saving components
0.196676  said share compani offer acquir merger acquisit common corp sharehold
0.16346  bank billion rate dollar rise monei januari februari rose market
0.153436  loss profit oper year note gain reuter sale includ exclud
0.117372  issu bond debentur coupon manag will underwrit debt rate offer
0.0898152  bank trade that debt brazil countri japan offici foreign would
0.0836767  qtly dividend record april prior march split payout quarterli payabl
0.062613  tonn coffe wheat crop export agricultur produc grain usda price
0.0475118  tonn wheat export sugar grain shipment corn import maiz barlei
0.0442977  opec barrel crude price refineri ecuador said that sugar product
0.0185196  gold ounc mine feet grade drill reserv barrel cubic said
0.00916865  copper mine rainbow south african progress miner magma steel plant
```

```
0.00614447  bank prime rate rais lira effect immedi dollar lend trust
0.0022709  reserv repurchas custom temporari feder economist agreement indirectli enter fund
0.0022709  reserv repurchas custom temporari feder economist agreement indirectli enter fund
0.00147766  hillard tesco stake siemen ordinari telecom tsco meston jacob campbel
0.00055293  bundesbank unchang reagan lombard enquiri polici poll answer disapprov council
0.000315927  human geigi ciba clearwat drug trial hillsdown cell clinic food
0.000105291  metromedia distanc switch nearest dedic unveil competitor custom link introduc
0.000105281  quota alloc import sugar mozambiqu paraguai gabon mauritiu papua tobago
0.000105271  medar automat owen mdxr farmington vision inspect toledo illinoi devic
0.000105209  nobel contraven incid regrett nobl erik unaccept weapon singapor histori
0  hillard clbga bancgroup votuporanga collieri paranavai catanduva simao grootvlei
0  hillard clbga bancgroup votuporanga collieri paranavai catanduva simao grootvlei
0  hillard clbga bancgroup votuporanga collieri paranavai catanduva simao grootvlei
0  hillard clbga bancgroup votuporanga collieri paranavai catanduva simao grootvlei
0  hillard clbga bancgroup votuporanga collieri paranavai catanduva simao grootvlei
0  hillard clbga bancgroup votuporanga collieri paranavai catanduva simao grootvlei
0  hillard clbga bancgroup votuporanga collieri paranavai catanduva simao grootvlei
0  hillard clbga bancgroup votuporanga collieri paranavai catanduva simao grootvlei
0  hillard clbga bancgroup votuporanga collieri paranavai catanduva simao grootvlei
```

The printout of the most discriminant words clearly suggests some of the official Reuters categories such as `acq`, `moneyfx`, `grain`, `crude`, etc.

Note that some clusters have probability zero. The list of discriminant words is then meaningless since the unigram probabilities for these clusters is uniform. This is in fact caused by a numerical weakness of the program `umix` that manifests itself when the λ coefficient of a cluster approaches zero. It is possible to fix it by using the `q[j][i]` variable to store $Q_{ij}\lambda_j^{-1}$ instead of Q_{ij}. Here is an updated program fragment of the E step:

```
// Compute P(X,Y=j) * exp(-maxlogqi) and normalize in order to get Qij
// * We do not compute P(X|Y=j) directly to avoid underflow.
double qnorm = 0;
for (int j=0; j<k; j++)
  {
    Component &c = component[j];
    double unormQij = exp(logqi[j] - maxlogqi);
    q[j][i] = unormQij;
    qnorm += c.lambda * unormQij;
  }
for (int j=0; j<k; j++)
  q[j][i] = q[j][i] / qnorm;
// At this point Qij = c[j].lambda * q[j][i].
```

and an updated program fragment from the M step

```
// Recompute lambda for component j
// * The formula says lambda_j = (sum_i Qij) / (sum_i sum_y Qiy)
//   but we know that sum_y Qiy = 1
c.lambda = c.lambda * qjsum / n;
```

With these changes the output of `umix` becomes:

```
$ ./scratch/umix reuters_train.txt reuters_dict.txt 15 reuters_clus15.txt
Read 9603 documents
Read 7303 terms (maxid 7303)
Initializing 15 components
Pass 107 logl=-4.77042e+06
```

```
Saving components
0.12664  share offer said debentur common compani file usair debt gencorp
0.11913  said that dollar bank opec market rate currenc price econom
0.113495  loss profit oper year note gain includ exclud discontinu extraordinari
0.10661  said compani earn will ounc quarter expect gold share sale
0.0892497  that brazil bank said trade debt countri japan would offici
0.0698412  issu bond coupon manag will eurobond lead underwrit franc bank
0.0686624  tonn wheat crop produc coffe export agricultur grain usda said
0.0640027  tonn wheat export januari februari rose sugar import fell rise
0.0586635  billion bank monei bill reserv surplu rose market januari deficit
0.0513699  qtly prior record april march payout dividend reuter quarterli qtrly
0.0376861  acquir said acquisit complet corp bank asset save agreement merger
0.0346841  dividend split share stock declar payabl record quarterli common sharehold
0.0341977  reuter march sale year corp dilut nine billion toronto calif
0.0182802  rate prime bank rais crude effect post price grade barrel
0.00748814  trade deficit volcker yeutter baker gasolin distil bill stop treasuri
```

The empty clusters have been replaced by small clusters whose content is relatively meaningful. However these clusters are typically not very stable: a different initialization could easily give very different small clusters. This illustrates how tricky the EM numerics can be.

Question 2 – Hierarchical Clustering for UMIX

We now would like to organize the clusters hierarchically. Program `humix` follows an agglomerative clustering strategy. It starts from the topics computed by `umix` and repeatedly merges the two closest topics until all topics have been merged into a single component.

The distance between two topics is determined using a heuristic formula implemented by the function `mergeDistance()` in file `humix.cpp`. Given two topics g and j, the function first computes the term distribution p_{*t} of the merged topic

$$p_{*t} = \frac{\lambda_g p_{gt} + \lambda_j p_{jt}}{\lambda_g + \lambda_j}$$

and returns

$$\lambda_g H(p_{gt}, p_{*t}) + \lambda_j H(p_{jt}, p_{*t})$$

where the Hellinger distance is

$$H(p_t, q_t) = \sum_t \left(\sqrt{p_t} - \sqrt{q_t}\right)^2 .$$

Program `humix` also computes the likelihood on both the training set and a validation set. This is achieved by function `computeLogLikelihood()` in file `humix.cpp`. However this function lacks a couple critical expressions clearly marked with text `INSERT_CORRECT_CODE_HERE`.

Question 2a Complete the missing parts of function `computeLogLikelihood()`. Provide a listing of that segment of the code showing your completion. There are strong similarities between this code and the E step of the `umix` program.

The program fragments are indeed very close to the `umix` E-step.

```
// Store log P(X=x_i|Y=j) into logpij[j]
for (int j=0; j<k; j++)
  {
    double sum = 0;
```

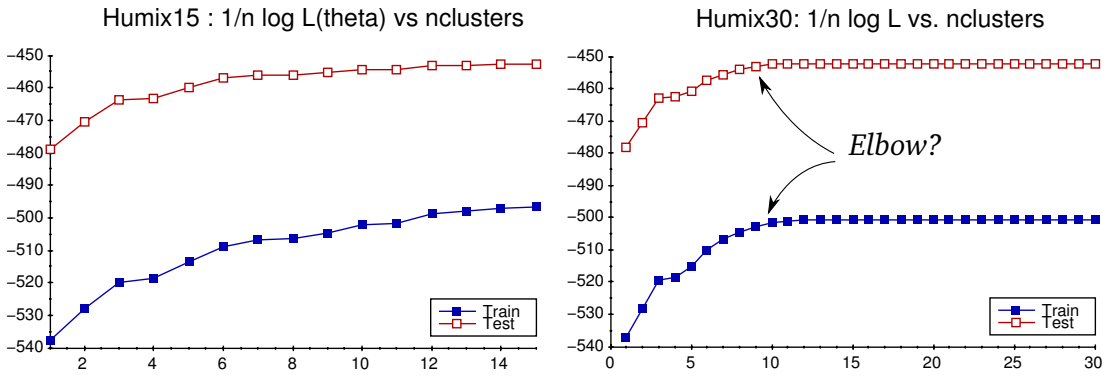
```
        for (const SVector::Pair *p = doc.data; p->i >= 0; p++)
          sum += p->v * comps[j].logp[p->i];
        logpij[j] = sum;
      }
     // Compute max_j log P(X=x_i|Y=j) into maxlogpj
    VFloat maxlogpij = logpij[0];
    for (int j=0; j<k; j++)
      maxlogpij = max(maxlogpij, logpij[j]);
    // Compute sum_j lambda_j P(X=x_i|Y=j) * exp(-maxlogpij) into piscaled
    // Attention: computing exp(logpij[j]) directly would underflow!
    double piscaled = 0;
    for (int j=0; j<k; j++)
      piscaled += comps[j].lambda * exp(logpij[j]-maxlogpij);
    // Add log(sum_j lambda_j P(X=x_i|Y=j)) to logl
    logl += log(piscaled) + maxlogpij;
```

Here is the beginning of the program output for the 5 components mixture:

```
$ ./humix reuters_train.txt reuters_test.txt reuters_dict.txt reuters_clus5.txt
Read 9603 documents
Read 3299 documents
Read 7303 terms (maxid 7303)
Read 5 components
----------------------  components: 5
Train logL = -4.90179e+06
Valid logL = -1.5112e+06
  #0: 0.339047  share offer issu corp debentur common dividend stock compani qtly
  #1: 0.241021  loss profit billion year oper januari quarter februari rose from
  #2: 0.186173  tonn wheat export market grain coffe crop usda sugar quota
  #3: 0.137562  bank trade that brazil countri japan debt minist japanes foreign
  #4: 0.0961973  mine barrel gold ship refineri crude said compani feet plant
Merging #2 and #3 as #1
----------------------  components: 4
Train logL = -4.93902e+06
Valid logL = -1.51722e+06
  #0: 0.339047  share offer issu corp common debentur stock qtly bond april
  #1: 0.323735  tonn export bank trade that market countri wheat brazil minist
  #2: 0.241021  loss profit year billion oper januari quarter februari rose from
  #3: 0.0961973  mine barrel gold ship refineri said crude feet plant grade
Merging #1 and #3 as #0
----------------------  components: 3
Train logL = -4.97703e+06
Valid logL = -1.52663e+06
...
```

Question 2b Run `humix` on the 15 and 30 components mixtures from question 1c. For each of these runs, produce a plot of the training and validation log likelihoods as a function of the number of components. Clearly mark the "elbow" if you can see it.

The following graphs plot the mean training and validation log likelihoods. That is, the log likelihoods have been divided by the number of examples in the training and testing sets respectively.



If the training and testing data had been selected randomly, we would expect to see a smaller mean log likelihood on the testing set. The reuters dataset convention is to train on the earlier stories and test on the latest stories. The higher mean log likelihood on the testing set suggests that the latest Reuters stories were more neatly clustered than the training stories.

Locating the elbow on the `Humix15` curves is anyone's guess. Things are a bit clearer on the `Humix30` curves. In both cases one gets about a dozen meaningful clusters.

Question 2c Could we make `humix` run faster? How?

Program `humix` computes all the Hellinger distances between clusters at each iteration. Since the distance between clusters not affected by the merging operation does not change from one iteration to the next, it is possible to avoid recomputing them. Such a speedup only matters if we are dealing with lots of clusters.

If you are interested by topic models and want to learn much more sophisticated approaches, you should definitely discuss with Sean Gerrish and see Professor David Blei...

Appendix – Vector classes

The provided source code relies extensively on two vector classes defined by files `vector.h` and `vector.cpp`. Class `FVector` and `SVector` respectively represent a dense vector and a sparse vector. Both classes define operators `<<` and `>>` to read and write a textual representation of the vector. This is useful for loading and saving the data files. Both classes provide a collection of nearly identical functions:

`x.size()`	Return the size of vector `x`.
`x.get(i)`	Return the `i`-th coefficient of vector `x`.
`x.set(i,v)`	Set the `i`-th coefficient of vector `x`.
`x.add(s)`	Add scalar `s` to vector `x`.
`x.add(y)`	Add vector `y` to vector `x`.
`x.scale(s)`	Multiplies all coefficients of `x` by `s`.
`dot(x,y)`	Return the dot product of vectors `x` and `y`.
`combine(x,a,y,b)`	Return the vector $a\mathbf{x} + b\mathbf{y}$.

When the index is larger than the vector size, function `x.get()` returns 0 and function `x.set()` automatically increases the vector size. You can also use the standard bracket syntax `x[i]` to quickly access the `i`-th coefficient of a `FVector x`. You must however be certain that the vector size is larger than `i`.

Sparse vector are implemented as a sequence of records corresponding to the nonzero coefficient. Each record contains the coefficient index and the coefficient value. Records are sorted by increasing index value. The following idiom can be used to iterate over all the nonzero coefficients of a `SVector x`:

```
for (const SVector::Pair *p = x; p->i >= 0; p++) {...}
```

In the body of the loop, `p->i` represents the coefficient index, and `p->v` represents the coefficient value.