AIM-3 Scalable Data Science (SS 2017)

Homework Assignment 3

Due on July 14, 2017 at 14:15

Instructions. For exercises 1-3, be sure to show all work, to receive full credit. The instructions for exercise 4 are described on page 3. Be aware you may be asked to meet with one of the instructors to run your codes later. **Note**: Source code stubs for exercise 4 may be found in our TU Berlin GitLab repository. You will need to upload your solutions as a single **PDF** file (HW3_lastname.pdf) to ISIS by no later than July 14, 2017 at 14:15. Also, you will need to drop off a stapled, printed copy of your homework assignment at the start of the lecture held on July 14. Lastly, be certain to work individually, you are free to drop some hints. However, you must solve these problems on your own.

1. (5 points) Dimensionality Reduction

Dimensionality reduction methods plays a vital role in performing machine learning on high-dimensional data. In class, we learned about the singular value decomposition (SVD). In MMDS, Section 11.3, the authors introduce CUR Decomposition.

(i) What is CUR decomposition? Briefly describe how it works. When is it useful? (2 points)
(ii) Solve Exercise 11.4.2 in the MMDS book. (3 points)

2. (6 points) Item Based Collaborative Filtering

You are given a dataset of ratings as depicted in the table below:

User	Movie A	Movie B	Movie C	Movie D
Ryan	1	5	3	5
Stavros			2	
Brahma		4	1	1
Brodie	4	3		2
Zosimus		5	1	

(i) Compute the item similarity matrix (denoted by R) using Pearson Correlation as a measure.
(ii) Compute the item similarity matrix (denoted by S) using Jaccard Coefficient as a measure.
(iii) Use matrices R and S to compute the corresponding ratings that Zosimus would give to Movies A & D.
(iv) Use matrices R and S to compute the corresponding ratings that Stavros would give to Movies A & B.

For each sub-problem (i) through (iv), show how you arrived at your solution.

3. (9 points) MapReduce

For each of the exercises listed below, *justify* your answer for credit.

(i) Suppose we have the following relations

R		S	
A	B	B	C
0	1	0	1
1	2	1	2
2	3	2	3

and we take their natural join by the algorithm of MMDS Section 2.3.7. Apply the Map function to the tuples of these relations. Then, construct the elements that are input to the Reduce function. Identify one of these elements in the list below

a) (3, (R,2))
b) (2, [(S,3)])
c) (1, (S,2))
d) (3, [(R,2)]

(ii) Suppose our input data to a MapReduce operation consists of integer values (the keys are not important). The *map* function takes an integer i and produces the list of pairs (p,i), such that p is a prime divisor of i. For example, map(12) = [(2,12), (3,12)]. The *reduce* function is addition. That is, reduce $(p, [i_1, i_2, \ldots, i_k])$ is $(p, i_1+i_2+\ldots+i_k)$. Compute the output, if the input is the set of integers 15, 21, 24, 30, 49. Then, identify, in the list below, one of the pairs in the output.

a) (5,30)
b) (3,90)
c) (2,47)
d) (2,102)

(iii) Using the matrix-vector multiplication method described in MMDS Section 2.3.1, applied to the matrix and vector:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1
2
3
4

apply the *map* function to this matrix and vector. Then, identify in the list below, one of the key-value pairs that are output by *map*.

a) (4,48)
b) (1,2)
c) (1,9)
d) (1,30)

4. (10 points) Naïve Bayes Classification in Apache Flink

In this exercise, you will conduct a practical programming exercise using Apache Flink and employ the Naive Bayes algorithm to solve a real-world problem.

I. Naive Bayes

Implement a Naive Bayes classifier using Apache Flink. Evaluate your implementation on a data set of postings from 20 newsgroups (check out `http://qwone.com/~jason/20Newsgroups/` for a description of the data). Clone the classification source code from the GitLab assignment repository (Assignments/Assignment3) and have a look at *de.tuberlin.dima.aim3.classification*. Be sure to adjust the configuration file, i.e., *de.tuberlin.dima.aim3.classification.Config* to point to the location of your local project. You will need to complete the three provided classes *Training*, *Classification* and *Evaluator*.

a. In *Training.java*, complete the two flink data flows that should compute the conditional counts of words per label (<label, word, count>) as well as the summed counts of all word counts per label (<label, counts>).

b. In *Classification.java*, implement the classification function.

c. In *Evaluator.java*, complete a function that computes the accuracy of your classifier on a provided test data set. Your classifier should produce an accuracy of well over 80%. If not you may have forgotten to introduce a smoothing parameter in your classifier.

We have also provided a test set without labels (*secrettest.dat*). Run your trained classifier and write the output to a file. It should contain the number, label, and probability assigned to each data point in a tab separated manner (e.g., [1] talk.politics.guns<tab>0.123).

Submit (i) a description of your model and experiments, including assumptions, (ii) the source code corresponding to the key methods, (iii) a snapshot of the run, and (iv) a plot of the generated results file, depicting the label and corresponding probability, where the x-axis is the label and the y-axis is the corresponding probability value. Be sure to include the smoothing factor you selected ('k') in the plot title.

II. Background Notes

a. **Naive Bayes Classifier: Theory**

As discussed in class, a naive Bayes classifier models a joint distribution over a label Y and a set of observed random variables or features, $(F_1, F_2 \dots F_n)$ using the assumption that the full joint distribution can be factored as follows (features are assumed to be conditionally independent given the label):

$$P(F_1, F_2 \dots F_n) = P(Y) \prod_i P(F_i|Y)$$

To classify a datum, we can find the most probable label given the feature values for each pixel, using Bayes theorem:

$$P(y|f_1, \dots, f_m) = \frac{P(f_1, \dots, f_m|y)P(y)}{P(f_1, \dots, f_m)} = \frac{P(y)\prod_{i=1}^{m} P(f_i|y)}{P(f_1, \dots, f_m)}$$

$$argmax_y P(y|f_1 \dots f_m) = argmax_y \frac{P(y)\prod_{i=1}^{m} P(f_i|y)}{P(f_1, \dots, f_m)} = argmax_y P(y) \prod_{i=1}^{m} P(f_i|y)$$

Given that multiplying many probabilities can result in underflow, instead we will compute log probabilities which have the same argmax:

$$argmax_y \log P(y|f_1 \dots f_m) = argmax_y \left\{ \log P(y) + \sum_{i=1}^{m} \log P(f_i|y) \right\}$$

To compute logarithms, use `Math.log()` function. Put simply, you will have to compute the most likely label given the data (text in the posting to be classified). This means you have to compute the probability for each label and then return the label which recieves the highest probability from your model for the document in question.

b. **Parameter Estimation**

Our naive Bayes model has several parameters to estimate. One parameter is the prior distribution over labels (the names of the newsgroups), $P(Y)$. for simplicity we assume a uniform prior accross all classes. The term $\log P(y)$ can thus be ignored in the argmax computation. The other parameters to estimate are the conditional probabilities of our features given each label y: $P(F_i|Y = y)$, where is the conditional probability of a word w occurring in a posting of class(topic) y. We do this for each possible feature value (each word in the vocabulary).

$$\hat{P}(w|Y = y) = \frac{count(w, y)}{\sum_{w'} count(w', y)}$$

where $count(w, y)$ is the number of times word w occured in the training examples of label y.

c. **Smoothing**

Your current parameter estimates are unsmoothed, that is, you are using the empirical estimates for the parameters. These estimates are rarely adequate in real systems. Minimally, we need to make sure that no parameter ever receives an estimate of zero, but good smoothing can boost accuracy quite a bit by reducing overfitting. In this assignment, we use Laplace smoothing, which adds k counts to every possible observation value:

$$P(w|Y=y) = \frac{count(w,y)+k}{\sum_{w'}(count(w',y)+k)}$$

If $k = 0$, the probabilities are unsmoothed. As k grows larger, the probabilities are smoothed more and more.