CMSC 25025 / STAT 37601

Machine Learning and Large Scale Data Analysis

Assignment 3 Out: Thursday, April 18 Theoretical part 1-4, due: Thursday, April 25, 2018 at 2:00 pm.

Experimental part 5-6, due: Thursday, May 2, 2018a at 2:00 pm.

Please hand in this Homework in 5 files:

1. A pdf of the theoretical homework (problems 2-4).
2. A pdf of your jupyter notebook for problem 5.
3. The ipynb file for problem 5.
4. A pdf of your jupyter notebook for problem 5.
5. The ipynb file for problem 5.

1. *Trouble with algorithms* (5 points)

   Read this article and be prepared to discuss in class. What problem in the previous homework is it related to?

   `https://arstechnica.com/information-technology/2016/02/the-nsas-skynet-program-may-be-killing-thousands-of-innocent-people/`

   Comment on the following article on Youtube's algorithm for recommending the next movie you watch:

   `https://www.theguardian.com/technology/2018/feb/02/how-youtubes-algorithm-distorts-truth`

   Add your answers to the discussion group *Trouble with machine learning algorithms II*.

2. *Lasso* (5 points)

   Show that the minimizer of the function $f(\beta) = -t\beta + \frac{1}{2}\beta^2 + \lambda|\beta|$ is given by $sign(t)[|t| - \lambda]_+$

3. *Logistic regression* (10 points)

   (a) Give a detailed derivation of the Newton algorithm for logistic regression and show that each iteration corresponds to solving a weighted least square problem. Hint: Compute the gradient ∇L and the Hessian H of the loss L at θ. Given current θ write the Newton iteration for $\theta^{new} - \theta$. Note that $H\theta$ simplifies.

   (b) Assume the data are perfectly linearly separable, i.e. there exist θ such that $x_i^t\theta < 0$ if $y_i = 0$ and $x_i^t\theta > 0$ if $y_i = 1$. Show that the conditional maximum likelihood estimator for the logistic regression model does not exist. Comment on the behavior of the iteratively reweighted least squares algorithm. Hint: If θ is a perfect separator then

$\alpha\theta$ is also for any $\alpha > 0$. It may be easier to work with the likelihood instead of the log-likelihood.

4. *Bernoulli mixtures:* (10 points)

   (a) Assume your observation are binary in R^d, i.e. $X = (X_1, \dots, X_d), X_j \in \{0,1\}, j = 1, \dots, d$. We assume the d variables are independent with joint distribution $P(X_1 = x_1, \dots, X_d = x_d) = P(X_1 = x_1) \cdots P(X_d = x_d)$, and $P(X_j = x) = p_j^x (1-p_j)^{1-x}$. Given a sample $X_1, \dots, X_n$ from this distribution write the log-likelihood, write the score equation for each parameter $p_j, j = 1, \dots, d$ and write the maximum likelihood estimate $\widehat{p}_j$.

   (b) Assume now that the data $X_1, \dots, X_n$ come from a mixture model with M components and each component a product of d independent Bernoulli variables as in (a):

   $$f(x;\theta) = \sum_{m=1}^{M} \pi_m f_m(x;\theta_m), \quad f_m(x;\theta_m) = \prod_{j=1}^{d} p_{j,m}^{x_j}(1-p_{j,m})^{(1-x_j)},$$

   where $\theta = (\pi_1, \dots, \pi_M, \theta_1, \dots, \theta_M)$ and $\theta_m = (p_{1,m}, \dots, p_{d,m})$.
   Derive the details of the EM algorithm for this model.

   i. Write out the precise formula for computing the responsibilities $w_{mi}, m = 1, \dots, M, i = 1, \dots, n$ in terms of the current estimates $\widehat{\pi}_m, \theta_m, m = 1, \dots, M$.

   ii. Once you have the responsibilities. Write the formula for computing the new estimates $\widehat{\pi}_m^{new}, p_{j,m}^{new}$.

5. *Bernoulli mixtures for mnist:* (30 points)

   Read in the mnist data set from the previous homework and binarize it by setting the value to 1 if there is some ‘ink’ over that pixel. Implement the EM algorithm on this data *for each class* with $M = 1, 3, 5$. So in total you will have $M \cdot 10$ models. And, for $M = 1$ you don’t need the EM algorithm.

   Implementation clues:

   (a) You want to avoid estimates of $p_{j,m}$ that are zero or one. Same for π. One way to do that is to say add 1 to the numerator and add 2 to the denominator of any ratio you compute as an estimate of the probabilities

   (b) Initialize the EM as follows. Assign each example at random to one of the M components. Compute the probability estimates $\widehat{p}_{j,m}, \widehat{\pi}_m$ according to this assignment and use them as initial values to compute the first round of responsibilities.

(c) Beware of multiplying lots of probabilities. You will quickly get below the rounding error of the computer. Work with sums of logs.

$$f_m(X_i; \theta_m) = \exp[\sum_j X_{ij} \log p_{j,m} + (1 - X_{ij}) \log(1 - p_{j,m})].$$

When you are computing the responsibility for example X_i and cluster m

$$w_{mi} = \frac{f_m(X_i, \widehat{\theta}_m)\pi(m)}{\sum_{m'=1}^{M} f_{m'}(X_i, \widehat{\theta}_{m'})}, m = 1, \dots, M.$$

First compute the $\log$ of each of the terms in the denominator sum. Call them $\ell_m(x), m = 1, \dots, M$. Find the largest one - L. Then compute:

$$w_{mi} = \frac{\exp(\ell_m - L)}{\sum_{m'} \exp(\ell_{m'} - L)}.$$

Can you explain why this is important?

(d) Sum the log-denominators of the responsibilities over all examples

$$L = \sum_{i=1}^{n} \log \sum_{m'=1}^{M} f_{m'}(X_i, \widehat{\theta}_{m'}).$$

This gives the current log likelihood of the data. This quantity should decrease with each iteration. One criterion to stop the iterations is when this quantity doesn't change much.

(e) After you estimate the different components show the probability vectors you obtain as an image, i.e. reshape $p_{j,m}, j = 1, \dots, 28^2$ as a 28×28 array. What do you see?

(f) Classify the test set data using Bayes' rule with your estimated class mixture models and report your results for M=1,3,5.

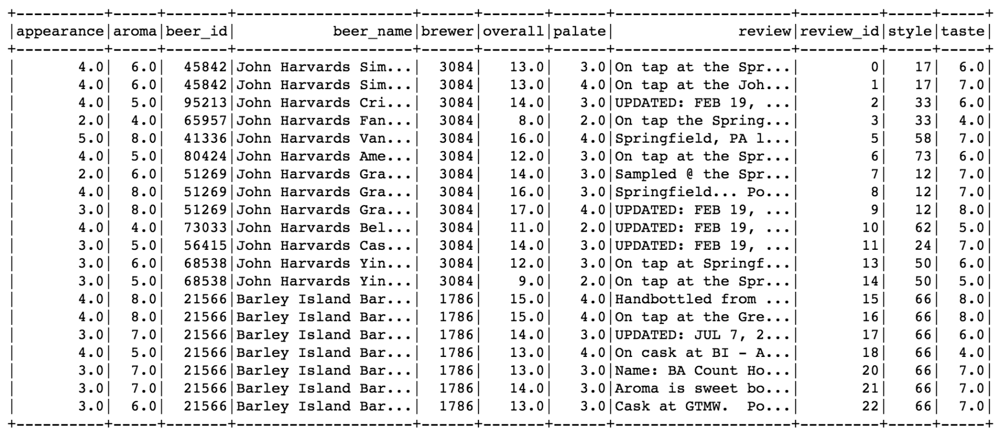
LSDA

6. *Stochastic gradient descent on beer reviews* (40 points)

In this problem, you will work with 2,924,163 beer reviews from `https://www.ratebeer.com`. One entry contains the text of the consumer's review together with ratings of *appearance, aroma, palate, style, taste*, and an *overall* score from 0 to 20. The data also include the name, id and brewer of the beer. You will give reviews binary label: 1 for a (*positive*) review with overall score at least 14, and otherwise 0 for a (*negative*) review.

Part 1: Data inspection.

To warm up, check the mean, median and standard deviation of the overall ratings for each beer and brewer. Do you think people have similar taste?



```
+----------+-----+-------+--------------------+------+-------+------+--------------------+---------+-----+-----+
|appearance|aroma|beer_id|           beer_name|brewer|overall|palate|              review|review_id|style|taste|
+----------+-----+-------+--------------------+------+-------+------+--------------------+---------+-----+-----+
|       4.0|  6.0|  45842|John Harvards Sim...|  3084|   13.0|   3.0|On tap at the Spr...|        0|   17|  6.0|
|       4.0|  6.0|  45842|John Harvards Sim...|  3084|   13.0|   4.0|On tap at the Joh...|        1|   17|  7.0|
|       4.0|  5.0|  95213|John Harvards Cri...|  3084|   14.0|   3.0|UPDATED: FEB 19, ...|        2|   33|  6.0|
|       2.0|  4.0|  65957|John Harvards Fan...|  3084|    8.0|   2.0|On tap the Spring...|        3|   33|  4.0|
|       5.0|  8.0|  41336|John Harvards Van...|  3084|   16.0|   4.0|Springfield, PA l...|        5|   58|  7.0|
|       4.0|  5.0|  80424|John Harvards Ame...|  3084|   12.0|   3.0|On tap at the Spr...|        6|   73|  6.0|
|       2.0|  6.0|  51269|John Harvards Gra...|  3084|   14.0|   3.0|Sampled @ the Spr...|        7|   12|  7.0|
|       4.0|  8.0|  51269|John Harvards Gra...|  3084|   16.0|   3.0|Springfield... Po...|        8|   12|  7.0|
|       3.0|  8.0|  51269|John Harvards Gra...|  3084|   17.0|   4.0|UPDATED: FEB 19, ...|        9|   12|  8.0|
|       4.0|  4.0|  73033|John Harvards Bel...|  3084|   11.0|   2.0|UPDATED: FEB 19, ...|       10|   62|  5.0|
|       3.0|  5.0|  56415|John Harvards Cas...|  3084|   14.0|   3.0|UPDATED: FEB 19, ...|       11|   24|  7.0|
|       3.0|  6.0|  68538|John Harvards Yin...|  3084|   12.0|   3.0|On tap at Springf...|       13|   50|  6.0|
|       3.0|  5.0|  68538|John Harvards Yin...|  3084|    9.0|   2.0|On tap at the Spr...|       14|   50|  5.0|
|       4.0|  8.0|  21566|Barley Island Bar...|  1786|   15.0|   4.0|Handbottled from ...|       15|   66|  8.0|
|       4.0|  8.0|  21566|Barley Island Bar...|  1786|   15.0|   4.0|On tap at the Gre...|       16|   66|  8.0|
|       3.0|  7.0|  21566|Barley Island Bar...|  1786|   14.0|   3.0|UPDATED: JUL 7, 2...|       17|   66|  6.0|
|       4.0|  5.0|  21566|Barley Island Bar...|  1786|   13.0|   4.0|On cask at BI - A...|       18|   66|  4.0|
|       3.0|  7.0|  21566|Barley Island Bar...|  1786|   13.0|   3.0|Name: BA Count Ho...|       20|   66|  7.0|
|       3.0|  7.0|  21566|Barley Island Bar...|  1786|   14.0|   3.0|Aroma is sweet bo...|       21|   66|  7.0|
|       3.0|  6.0|  21566|Barley Island Bar...|  1786|   13.0|   3.0|Cask at GTMW.  Po...|       22|   66|  7.0|
+----------+-----+-------+--------------------+------+-------+------+--------------------+---------+-----+-----+
```

```
df = spark.read.json('/project/cmsc25025/beer_review/labeled.json')
df.show()
-----------------------------------------------------
```

Part 2: Sentiment analysis.

Your first task is to classify the sentiment of reviews from the text. Text can have neutral, mixed, sarcastic, or otherwise ambiguous sentiment. *Sentiment analysis*[1] can be challenging even for people.

Here is one example review that you will be working on:

```
I got this one in Indianapolis. The body was dark brown. The aroma
was caramel and chocolate with some pancakes. The taste was pancakes
with coffee and chocolate. Well made, but a little boring.
```

Other reviews seem easier to classify according to positive or negative sentiment:

```
I was surprised by this one. A really nice local sour beer.
Pours a great looking brown with a slight red hue. Not much head,
but a good amount of lacing. Aroma is nutty, with cherries and a
tartness to it. Very earthy. Flavor is nice and sour, lots of red
fruits and nuts, with just a hint of yeast. The wood really comes
into play here, and works nicely with the other elements. This one
is tasty for sure.
```

After loading the labeled data, proceed as follows.

(a) *Generating features.* You need to represent text reviews in terms of a vector of features (covariates). One simple but effective representation is to use membership in a fixed vocabulary. Suppose the vocabulary contains p words. For a given review, you normalize the text,

[1]http://en.wikipedia.org/wiki/Sentiment_analysis

and separate it into space-delimited tokens. For each of the tokens, if it is in the dictionary you have a one for the corresponding word in the feature vector, and you ignore it otherwise.

For example, suppose the review is

```
"I like this beer!  Thanks for sharing!  Yay!"
```

and after normalization you process this into the list

```
["i", "like", "this", "beer", "thanks", "for", "sharing", "yay"].
```

If `"i"`, `"this"`, `"for"` and `"yay"` are not in your dictionary, but the other words are, you have four active features

```
["like", "beer", "thanks", "sharing"].
```

So, this review would be a p-dimensional vector where four features are set to 1, and the rest are set to 0. You should drop a review if none of its words is in the dictionary.

We have processed a few vocabularies for you. The file `vocab_50.json` contains `word:id` pairs for 30,009 words. Those words appear at least 50 times across all the reviews. Similarly for `vocab_{10,20,30}.json`. The most common 50 words are thrown out. You are more than welcome to use your own vocabulary. To load our vocabulary, run

```
with open('/project/cmsc25025/beer_review/vocab_50.json', 'r') as f:
    vocab = json.load(f)
```

Note that your feature vectors are going to be very sparse, as most reviews have very few words. In order to speed up the computation, using a sparse vector representation for high dimensional data is crucial. We recommend `scipy.sparse.csr_matrix`[2] since it's easy to construct from scratch. To construct these features for every document You will have to loop over every document and find which vocabulary words are in the document. Try your function on a few documents to see how long it takes. It could take a few minutes to process the whole data set. Split your data into training, validation and testing sets with proportion 0.7:0.15:0.15.

(b) *Logistic regression using Newton's method.* Train an ℓ_2-regularized logistic regression classifier using the `sklearn.linear_model.LogisticRegression` class. To select the regularization parameter $C = 1/\lambda$, you should try different values on the validation set. Pick the best. How long does it take to train?

Do the same thing using the `LinearSVC` class in `sklearn.svm`. Use `loss='hinge'`. Compare the results of the logistic loss to the hinge loss. Is there a difference?

[2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_matrix.html

(c) *Stochastic gradient descent.* Your next job is to train an ℓ_2-regularized logistic regression classifier using stochastic gradient descent. Recall the SGD framework that was covered in class using minibatches.

i. Initialize the model with $\theta = 0$ (uniform).

ii. Randomly split the training data into mini-batches. Make one pass of the data, processing one mini-batch in every iteration. This is called one training epoch.

iii. Repeat the last step a few times.

Remember the role that the learning rate plays in SGD. Tiny learning rates lead to slow convergence, while large rates can impede convergence—it might make the objective value oscillate. Try fixed vs. decreasing learning rates.

To select the regularization parameter λ, you should try different values on the validation set (see the next question for the proportion). Select those that yield the smallest validation error. You might want to do this on a smaller training set than the full one. Once you've chosen the values of these parameters train the model Plot the error rate and negative log-likelihood for both training and validation set as a function of iterations. Finally report the error on the test set. How does it compare in terms of time and error rate to the Logistic regression function in the previous item.

Part 3: Scores versus text.

In addition to text reviews, the users also scored *appearance, aroma, palate, style, taste* of a beer. In this problem, you will check whether those scores could reflect people's opinion better than text.

Train another logistic regression model using those features. You should use the same SGD algorithm as before. Compare the model using score features with that using review text. Again, use the validation set to tune the regularization parameters, and retrain the model on the union of training and validation set. Finally, compute the prediction error on the testing set.

Which model predicts better? Is the representation you constructed for text more powerful, or are the scores? Why? Comment on your findings and discuss your thinking.