PROJECT REPORT

AKSHAY KULKARNI kulka182@umn.edu

I have uploaded the files that contain highest weighted feature corresponding to each representation and relevant graphs for summary here (https://goo.gl/9QzSIF). Please copy and paste the url in browser. I did this because document was getting very big. Please don't cut my marks for this.

DATA STRUCTURES

Input data was stored in **CSR (compressed storage format)** format. In this format, a n*n sparse matrix A that has k non-zero entries is stored using 3 arrays:

- Two integer arrays **rowptr** and **colind**
- One array for real **values**.

The array **rowptr** is of size n+1 and other two arrays are of size of K. The **colind** stores column indices of the non-zero entries in A, and the array value stores column indices of non-zero entries in A. The **rowptr** is used to determine where the storage of a row starts and ends in the arrays **colind** and **values**.

Along with this for Ridge Regression matrix was stored in **CSC (compressed storage format)**. This was necessary of efficient computation of dot-product($\mathbf{X}_i \mathbf{w}_i$) of X(input) and w(Weights) without including column.

After storing the input, it was divided into training, validation and testing set. Then training set was also transformed into the format (tf, tfidf, binary, sqrttf) provided on command-line. Then, training, validation and testing was normalized.

CENTROID BASED CLASSIFIER

- Firstly, centroid of all classes (20 classes) was computed by assigning the training documents to that class.

- For each class two types of centroid were computed i.e one centroid was computed by using training documents that belonged to that class and other

centroid was computed by using training documents that didn't belonged to this class.

- Maximum F1(+ve) score for each class was computed using the approach mention in slides (108 and 109). Also score of each document with respect to each class kept and it was assigned to class with maximum score.

NEAREST NEIGHBOR BASED CLASSIFIER

- For this classifier no initial training was needed. For every test documents its distance(similarity) with all training documents were computed and K documents with highest similarity were stored.

- Maximum F1(+ve) score for each class was computed using the approach mention in slides (108 and 109). Also score of each document with respect to each class kept and it was assigned to class with maximum score.

NON-NEGATIVE RIDGE REGRESSION

- Firstly, input was stored in stored in both CSR and CSC format for efficient computation of weights.

- Coordinate descent was used to estimate weights. Initial value of weights were set to **0.0.** During each iteration of coordinate descent, weights were optimized for each feature at a time keeping all other features fixed. Also, it was ensured weights never were below zero.

- Let **c1** be cost before start of outer iteration and **c2** be cost after outer iteration. Algorithm was terminated when **c2-c1 < 1e-6**.

- Validation set was used to estimate best **lambda** for each class. Again the approach mentioned in slides (108 and 109) was used.

- For each document in testing set, its score with each class was calculated using best **lambda** obtained in previous step. And it was assigned to the class with maximum score.

RUN-TIMES

Classifier	Input	Representation	K	Time(in Secs)
regression	word	tf	0	10.98453
regression	word	binary	0	13.977722
regression	word	sqrt	0	14.02533
regression	word	tfidf	0	14.378153
regression	word	binaryidf	0	17.796307
regression	word	sqrtidf	0	16.812634
regression	char	tf	0	105.725587
regression	char	binary	0	120.83657
regression	char	sqrt	0	120.841283
regression	char	tfidf	0	126.672709
regression	char	binaryidf	0	134.188465
regression	char	sqrtidf	0	141.802982
nearestneighbor	word	tf	5	8.615155
nearestneighbor	word	tf	10	8.006252
nearestneighbor	word	tf	15	8.006184
nearestneighbor	word	tf	20	8.130364
nearestneighbor	word	binary	5	8.009557
nearestneighbor	word	binary	10	8.008636
nearestneighbor	word	binary	15	8.02076
nearestneighbor	word	binary	20	8.158345
nearestneighbor	word	sqrt	5	8.009008
nearestneighbor	word	sqrt	10	8.061373
nearestneighbor	word	sqrt	15	8.014221
nearestneighbor	word	sqrt	20	8.026738
nearestneighbor	word	tfidf	5	8.01536
nearestneighbor	word	tfidf	10	8.055559
nearestneighbor	word	tfidf	15	8.86675
nearestneighbor	word	tfidf	20	8.01625
nearestneighbor	word	binaryidf	5	8.02215
nearestneighbor	word	binaryidf	10	8.011985
nearestneighbor	word	binaryidf	15	8.010751
nearestneighbor	word	binaryidf	20	8.021953
nearestneighbor	word	sqrtidf	5	8.014917
nearestneighbor	word	sqrtidf	10	8.024229
nearestneighbor	word	sqrtidf	15	8.088419

nearestneighbor	word	sqrtidf	20	8.015441
nearestneighbor	char	tf	5	39.703287
nearestneighbor	char	tf	10	39.768529
nearestneighbor	char	tf	15	39.846036
nearestneighbor	char	tf	20	39.721157
nearestneighbor	char	binary	5	39.696426
nearestneighbor	char	binary	10	39.705634
nearestneighbor	char	binary	15	39.788562
nearestneighbor	char	binary	20	39.713182
nearestneighbor	char	sqrt	5	39.782128
nearestneighbor	char	sqrt	10	39.744096
nearestneighbor	char	sqrt	15	39.732814
nearestneighbor	char	sqrt	20	40.093722
nearestneighbor	char	tfidf	5	39.715058
nearestneighbor	char	tfidf	10	39.708872
nearestneighbor	char	tfidf	15	40.314403
nearestneighbor	char	tfidf	20	39.722433
nearestneighbor	char	binaryidf	5	39.758721
nearestneighbor	char	binaryidf	10	39.707711
nearestneighbor	char	binaryidf	15	39.707894
nearestneighbor	char	binaryidf	20	40.791372
nearestneighbor	char	sqrtidf	5	39.718182
nearestneighbor	char	sqrtidf	10	39.763174
nearestneighbor	char	sqrtidf	15	39.74164
nearestneighbor	char	sqrtidf	20	40.183928
centroid	word	tf	0	0.089441
centroid	word	binary	0	0.088153
centroid	word	sqrt	0	0.077738
centroid	word	tfidf	0	0.124083
centroid	word	binaryidf	0	0.077001
centroid	word	sqrtidf	0	0.079146
centroid	char	tf	0	0.431907
centroid	char	binary	0	0.676075
centroid	char	sqrt	0	0.418978
centroid	char	tfidf	0	0.381415
centroid	char	binaryidf	0	0.448622
centroid	char	sqrtidf	0	0.400304