CSE 158/258, Fall 2019: Homework 2

Instructions

Please submit your solution **by the beginning of the week 5 lecture (Oct 28).** Submissions should be made on **gradescope**. Please complete homework **individually**.

This specification includes both questions from the undergraduate (CSE158) and graduate (CSE258) classes. You are welcome to attempt questions from both classes but will only be graded on those for the class in which you are enrolled. MGTA495 students should attempt CSE258 questions.

You will need the following files:

Polish Bankruptcy data : `https://archive.ics.uci.edu/ml/datasets/Polish+companies+bankruptcy+data`

Code examples : `http://jmcauley.ucsd.edu/code/week2.py` (classification) and `http://jmcauley.ucsd.edu/code/week3.py` (clustering/communities)

Executing the code requires a working install of Python 2.7 or Python 3 with the scipy packages installed. **Please include the code of (the important parts of) your solutions.**

Tasks — Diagnostics (week 2):

In the first homework, we had two issues with the classifiers we built. Namely (1) the data were not shuffled, and (2) the labels were highly imbalanced. Both of these made it difficult to effectively build an accurate classifier. Here we'll try and correct for those issues using the *Bankruptcy* dataset.

1. Download and parse the bankruptcy data. We'll use the `5year.arff` file. Code to read the data is available in the stub. Train a logistic regressor (e.g. `sklearn.linear_model.LogisticRegression`) with regularization coefficient $C = 1.0$. Report the accuracy and Balanced Error Rate (BER) of your classifier (1 mark).

2. **(CSE158 only)** Retrain the above model using the `class_weight='balanced'` option. Report the accuracy and BER of your new classifier (1 mark).

3. Shuffle the data, and split it into training, validation, and test splits, with a 50/25/25% ratio. Using the `class_weight='balanced'` option, and training on the training set, report the training/validation/test *accuracy* and *BER* (1 mark).

4. Implement a complete regularization pipeline with the balanced classifier. Consider values of C in the range $\{10^{-4}, 10^{-3}, \ldots, 10^{3}, 10^{4}\}$. Report (or plot) the train, validation, and test BER for each value of C. Based on these values, which classifier would you select (in terms of generalization performance) and why (1 mark)?

5. **(CSE158 only)** Compute the F_β scores for $\beta = 1$, $\beta = 0.1$, and $\beta = 10$ for the above classifier, using $C = 1$ (on the test set) (1 mark).

6. **(CSE258 only)** The `sample_weight` option allows you to manually build a balanced (or imbalanced) classifier by assigning different weights to each datapoint (i.e., each label y in the training set). For example, we would assign equal weight to all samples by fitting:

   ```
   weights = [1.0] * len(ytrain)
   mod = linear_model.LogisticRegression(C=1, solver='lbfgs')
   mod.fit(Xtrain, ytrain, sample_weight=weights)
   ```

   (note that you should use the `lbfgs` solver option, and need not set `class_weight='balanced'` in this case). Assigning larger weights to (e.g.) positive samples would encourage the logistic regressor to optimize for the True Positive Rate. Using the above code, compute the F_β score (on the test set) of your (unweighted) classifier, for $\beta = 1$ and $\beta = 10$. Following this, identify weight vectors that yield better performance (compared to the unweighted vector) in terms of the F_1 and F_{10} scores (2 marks).[1]

[1]Note that on our (small) dataset, these measures can be quite sensitive to the assignment of datapoints to the train/test splits. You may want to re-shuffle your data if you are encountering degenerate solutions.

Tasks — Dimensionality Reduction (week 3):

Next we'll consider using PCA to build a lower-dimensional feature vector to do prediction.

7. Following the stub code, compute the PCA basis on the training set. Report the first PCA component (i.e., `pca.components_[0]`) (1 mark).

8. Next we'll train a model using a low-dimensional feature vector. By representing the data in the above basis, i.e.:

```
Xpca_train = numpy.matmul(Xtrain, pca.components_.T)
Xpca_valid = numpy.matmul(Xvalid, pca.components_.T)
Xpca_test = numpy.matmul(Xtest, pca.components_.T)
```

   compute the validation and test BER of a model that uses just the first N components (i.e., dimensions) for $N = 5, 10, \ldots, 25, 30$. Again use `class_weight='balanced'` and $C = 1.0$ (2 marks).