Winter 2022 **CSC 566:** Advanced Data Mining Alexander Dekhtyar

Assignment 3: Support Vector Machines

Due date: Tuesday, February 17, end of day.

Note: Extra credit parts of the assignment will be open for another week (through February 25), but **Assignment 4** will be out on or before February 17.

Assignment

This is a team assignment. Break out in teams of 3-4 people.

Main Task. Your main task is to implement the Support Vector Machines using gradient descent and compare your implementation to that of `sklearn`. This assignment, however, comes with several strings attached.

Datasets and Data Acquisitions. For this assignment you are repsonsible for constructing your own datasets for the primary task (as well as for the secondary/extra credit tasks). The data for the datasets that you need to construct is widely and publicly available. Constructing the dataset with the specific set of features you like is **part of the assignment**.

Classification Problem. Our test problem is going to be the prediction of the winner in a US Presidential Election for each US county. There are 3006 counties in the USA[1], which gives us a reasonably sized dataset to test the work of Support Vector Machines. We will use 2012 (Obama-Romney), 2016(Trump-Clinton) and 2020 (Biden-Trump) elections as our main testing grounds, but you are welcome to build and use additional presidential elections datasets for your work.

Your datasets shall consist of the following:

1. The county-by-county winner of the presidential election in each election year above. The county-by-county elections data can be found in a variety of places (including, I believe, Kaggle).

2. The county-by-county demographic data. You can use the US Census Bureau data to obtain the demographic data for each county. Your demographic data must be for a time that roughly matches the election date - so, for 2020 election, you need 2020 data, but for 2016 elections you need data from 2016, or, failing that 2015.

   You can select what demographic attributes to include in your dataset. Feature engineering - i.e., finding the best features that predict the outcomes (winner of election) is an integral part of this assignment, and it allows you to engage in some creativity, and test multiple ideas against each other.

3. Any additional county-by-county data you may choose to include. (As a somewhat frivolous example, a team may decide to include weather information: temperature, rainfall/snowfall, windchill factor, etc., on election day for each county as extra features).

   Note: this is optional. I believe decent results can be had with some very simple demographic data (i.e., you can probably drop a lot of the Census data out of your final dataset and still be very accurate), but I do not wawnt to stymie your creativity. In fact, **the most accurate models** submitted to me based on data that is above and beyond demographic (and improves on the pure demographic predictions) will be considered for additional credit.

[1] There are also boroughs, parishes, census areas etc in several states that may increase the count a bit, but they are all reported in the same way in the datasets that you will be using.

Machine Learning. Implement Linear Kernal Support Vector Machines via gradient descent and use your implementation to classify your datasets. Compare your SVM implementation to the scikit-learn implementation of Linear Kernal Support Vector Machines. Ensure that they are comparable and find the same/very similar models.

Note: Support Vector Machines have one hyper-parameter: C, the importance of the slack penalty. You need to tune this hyperparameter as part of your SVM implementation - the final accuracy can depend on the actual value of C.

You can use something like 80-20 train-test split (ensuring that both the training and the testing data have counties from each state) for testing purposes, however, when running the final evaluation of the SVM classifier and when comparing its work to the work of other classifiers, **use 10-fold cross validation.** You can set up the cross-validation manually, or use scikit-learn's cross-validation implementation.

Compare and contrast. In addition to comparing your SVM implementation to scikit-learn's implementation, compare the results of your SVM classifier to the results of running Logistic Regression and Linear Discriminant Analysis. In doing so, please note:

- **Same featureset comparisons:** the most straightforward way is to compare SVM, LDA and Logistic Regression on the same dataset - so, you could, for example, figure out the best features for SVM (your primary task), and then run LDA and Logistic Regression on the same data. This however potentiall puts the other two methods at a disadvantage, because for LDA and/or Logistic Regression, there may be different importances assigned to the features than for SVMs.
- **Best featureset comparisons:** you could also optimize the features for each method separately, and compare the performance of the individual classification methods on their "preferred" datasets.

Report. Create a report describing your work. The report should be formatted as an academic technical report (you can, at your desire, format it as an ACM or IEEE paper, but this is optional), and shall include the following:

- Title
- Names and email addresses and affiliations of all authors (use alphabetical order by last name)
- Short abstract
- Introduction section that provides a brief overview of your work.
- Description of the datasets you pulled the data from (where you got the data, and so on).
- Description of your feature engineering. Describe the dataset/datasets you created, and the features you engineered.
- Description of your SVM implementation.
- Description of your testing/hyperparameter tuning.
- Description of all comparative studies (SVM vs SVM, SVM vs LDA vs Logistic Regression) you conducted.
- Results: accuracy results/confusion matrices for all models you want to highlight in your comparative studies, and the side-by-side comparisions.
- Discussion.

Extra Credit

For extra credit in the course, you can do one or more of the following.

Use Non-Linear Kernels. Implement a generalized version of SVMs with non-linear kernels, test the implementation (you can use something like Iris for testing/debugging your implementations), figure out what non-linear kernels you want to try, and run non-linear SVMs it on your elections datasets. Compare the results of linear and non-linear SVMs in your report.

Extend SVMs to the case of regression. You can use something like Ratio of Election winner to Election loser votes as value you want to estimate (this make it look like logistic regression, but your internal mechanism still will be SVMs). Test your solution, report on its accuracy.

Extend SVMs to the case of multivariate classification. To test, you can construct and use datasets for Canada's, France's, UK's or Israel's parliamentary elections (caution - Israel uses a different system, but there is still a notion of "which party got the most votes in a specific census-designated place; for all other elections, you need to predict the party of the winner of the electoral district). Again, for code validation purposes, Iris (it's a three class dataset) may be a good choice.

Deliverables. Submit all your code (you can submit Jupyter notebooks directly) and your report. Use the following `handin` command. I would appreciate it if the PDF of your report is submitted separately from your code, and your code is submitted as a zip or a .tar.gz file.

```
handin dekhtyar 566-a04 <files>
```

GOOD LUCK!