Chloe Drazen
Aaron Zhou

Congressional Deciphers

Data and Problem

The Congressional Voting Records dataset contains the results of 16 votes for each of the 435 U.S. House of Representatives Congressmen in 1984. Each vote is classified as either a yea (+1), nay (-1), or unknown (0). The original data includes each Congressman's party affiliation, either Democrat (+1) or Republican (-1), along with their votes on the following bills:

1. handicapped-infants
2. water-project-cost-sharing
3. adoption-of-the-budget-resolution
4. physician-fee-freeze
5. el-salvador-aid
6. religious-groups-in-schools
7. anti-satellite-test-ban
8. aid-to-nicaraguan-contras
9. mx-missile
10. immigration
11. synfuels-corporation-cutback
12. education-spending
13. superfund-right-to-sue
14. crime
15. duty-free-exports
16. export-administration-act-south-africa

The problem of this assignment is to determine if a given congressmen is a Democrat or a Republican given his voting record on the above 16 bills. In mathematical terms, this translates into a two-class classification problem.

Before we began analyzing the data, we first converted the data file from strings of 'yes' and 'no', 'democratic' and 'republican' to the numbers numerical values described above in the parentheses using the following Bash Shell Script:

```
sed 's/democrat/1/;s/republican/-1/;s/y/1/g;s/n/-1/g;s/?/0/g' <house-votes-84.data.txt >votes.csv
```

Methods

To classify the data, we made use of three techniques studied in class: soft Support Vector Machine (SVM), AdaBoost, and Principal Component Analysis (PCA).

As an appetizer, we used PCA to see what combinations of votes have the greatest amount of variation. Note that the direction in which the greatest variation occurs need not be the direction along which party affiliation is determined.

We represented our data as a matrix of dimensions 435 by 16. Each data point x is row vector in $\{+1, -1\}^{16}$. We used soft SVM because the data was likely not linearly separable. We used the soft SVM program from the lab to classify the data, where X is each congressman's voting record, t is the congressman's true party affiliation, and gamma is 0.005. Once we performed the Soft SVM, we analyzed the corresponding weight vector and correctness of the classification, which is discussed in the Results section.

We also made use of AdaBoost to predict the party affiliations of the congress members.

For both SVM and AdaBoost, we first performed the classification task without reducing the dimension of the data by PCA, after which we used SVM and AdaBoost on the PCA-dimensionally reduced dataset.

Using a Python script we generated a training dataset consisting of 80% of the original data, i.e. 348 out of 435 voting patterns of the members of Congress, and used the remaining 20% of the data as the testing dataset. The accuracies of all our predictions are measured with respect to the testing dataset.

In order to visualize the spread of the data for AdaBoost, we reduced the data from 16 dimensions to 2. Once we found the first two principal components and projected the original dataset, we performed the AdaBoost algorithm to classify the data into two classes. Those results are discussed below in the Results and Interpretation section.

Results and Interpretation

<u>Principal Component Analysis</u>

In order to visualize the high-dimensional data, we performed PCA. The idea behind the PCA is that we want to reduce the original dimension of the data, which is 16, to a lower dimension, which is 2 in our case, such that we capture the greatest amount of variation in the dataset.

We noticed that when the data is projected onto its first two principal components, it is very divided between the blue points (Democrats) and red points (Republicans) along the first principal component, meaning that the two parties are very polarized in their opinions. But they are significantly less so along the second principal component. The corresponding plot is shown below in Figure 1.

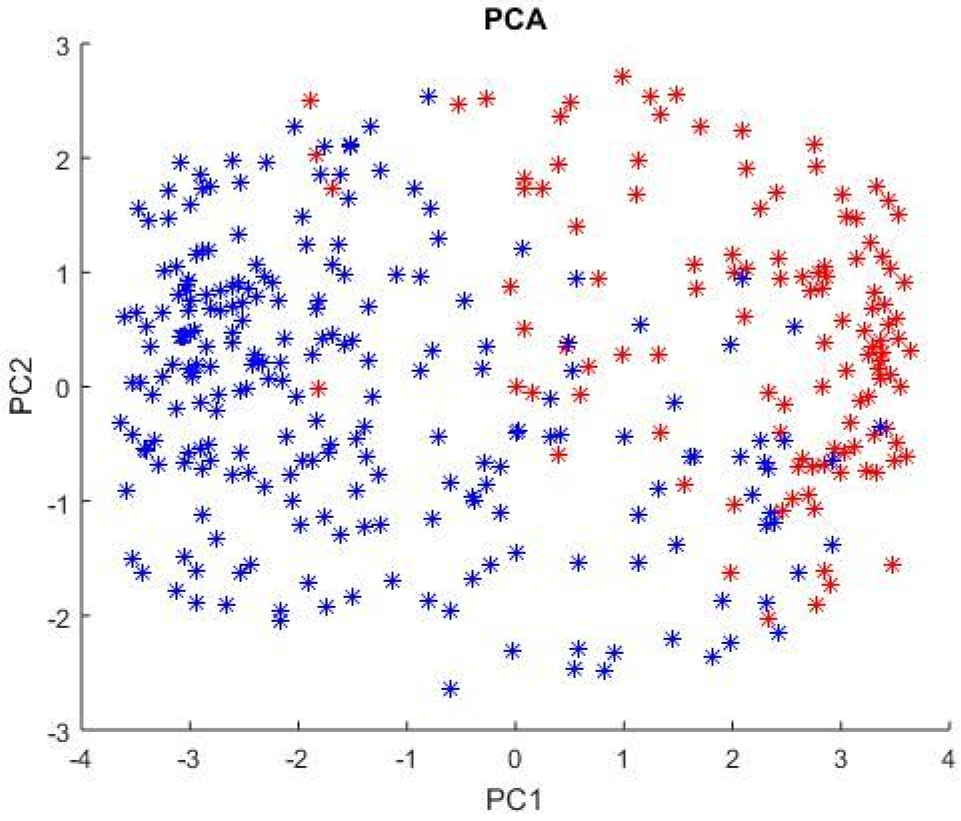

Figure 1

To further examine the first principal component, we found the element with the greatest magnitude to be the 5th entry in the PC1 vector, which corresponds to el-salvador-aid, which means that the two parties hold a very polarized view on this issue. In contrast, the least decisive party indicator was the immigration bill, which means that the Congressmen were not very polarized along party lines for this bill.

On the other hand, the 2nd element has the greatest magnitude in the PC2 vector, which corresponds to water-project-cost-sharing. This implies that although the congressional members' opinions on this issue vary a lot, the variations aren't based on their party affiliations.

<u>Soft Support Vector Machine</u>

First of all we used SVM on the entire dataset instead of the training dataset. We did this because we wanted to examine the weights to get some idea of which voting issue is the most significant in determining a member's party affiliation.

We found that 4.76% of Republicans were incorrectly classified (8 out of 168), and 6.0% of the Democrats were incorrectly classified (16 out of 267). Overall, 5.51% of the data was incorrectly classified using the Soft SVM method. The SVM classified data is shown below in Figure 2. The points on x = -1 whose y-value is above 0 are the eight incorrectly classified Republicans. Similarly, the points on x = +1 whose y-value is below 0 represent the sixteen incorrectly classified Democrats.

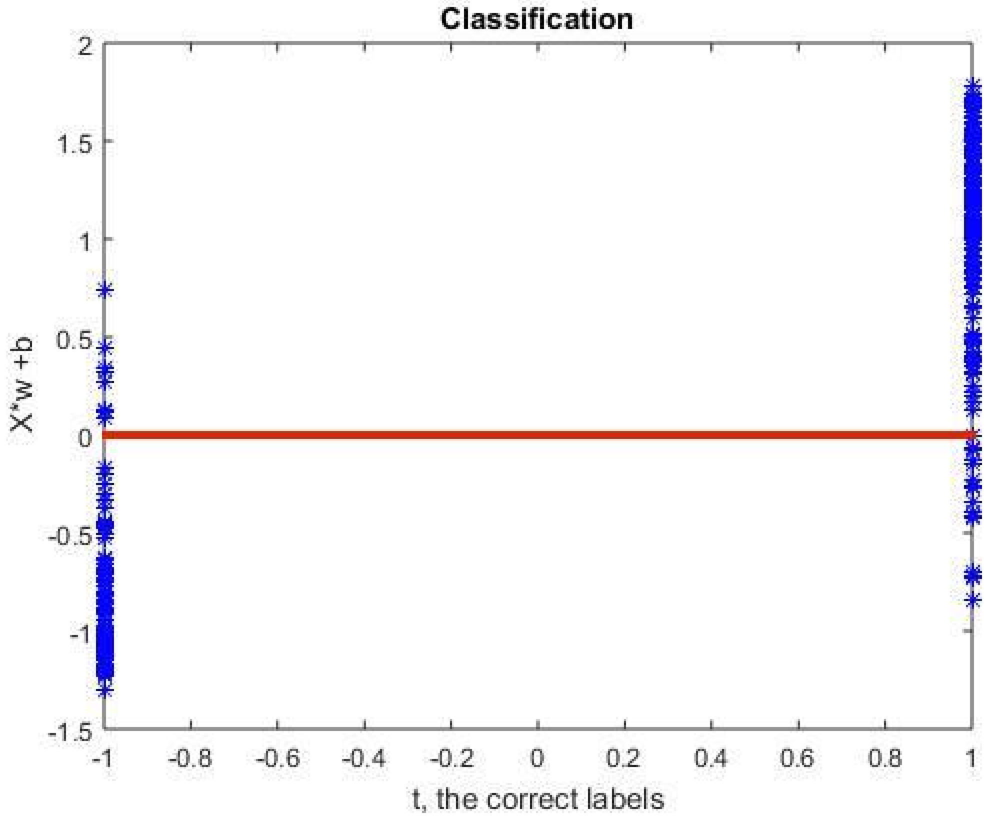

Figure 2

Using SVM, we learned the optimal weight vector for w is:
w = [0.0762; 0.0118; 0.1909; -0.4809; -0.1359; 0.0151; -0.0353; 0.0447; 0.1054; -0.0326; 0.1320; -0.1404; -0.0518; -0.0623; 0.0557; -0.0269];

Each value of w corresponds to how much weight each of the 16 votes has in determining party affiliation. By finding the minimum and maximum of the magnitude of these values, we found that the most discriminative vote was at index 4, which was physician_fee_freeze. Interestingly, the probability that a Congressman is a Democrat, given that he voted no on this bill, is 0.99. On the other hand, the probability that a Congressman is a Republican, given that he voted yes on this bill is 0.92. The results obviously indicate that this bill was very decisively split down party lines.

We also found that the least decisive vote corresponds to the second index, water_project_cost_sharing. For this vote, Democrats were very split in how they voted. For example, the probability that a Congressman voted yes, given that he is a Democrat is only 0.45. Conversely, the probability that a Congressman voted no, given that he is a Democrat is also 0.45 (the remaining 0.10 is for abstained votes). Therefore, this vote is not a good indicator of party affiliation when Congressmen in the same party are so divided on the issue.

Comparing these results to those of the most and least decisive votes obtained through PCA, we see that in PCA our most decisive vote is el_salvador_aid, our third most decisive vote is the physician_fee_freeze (which is the most decisive vote according to SVM). When we compare the magnitude of these two votes, the different in magnitude is only 0.014672. Compared to the total difference in magnitude between the most and least decisive votes of the PCA weight vector of 0.11318, it is clear that the two votes were very close for most decisive, and both are still a good indicator of party affiliation.

Similarly comparing our results for the least decisive vote obtained through PCA (immigration) and SVM (water_project_cost_sharing), the difference between the PCA weight vector's first two most decisive bills is only a 0.0025508 difference in magnitude. Therefore, although water_project_cost_sharing is not the least decisive vote for PCA as it is for SVD, it is the second least decisive.

In order to see how accurate SVM is in predicting party affiliations, we trained our SVM model on 80% of the data and then used the corresponding weight vector to test the remaining 20%. We were pleased to discover that our model achieved an accuracy of 95.4% on the testing dataset.

Projecting the SVM Decision Boundary to the Principal Components.

We projected the SVM decision boundary, which consists of the normal vector w and an offset value b, to the first two principal components of the dataset.

While the original decision boundary is a hyperplane in the 16-dimensional data space where each dimension corresponds to one voting issue, the projection brings the the decision boundary down to a 1-dimensional affine hyperplane in the 2-dimensional principal component subspace.

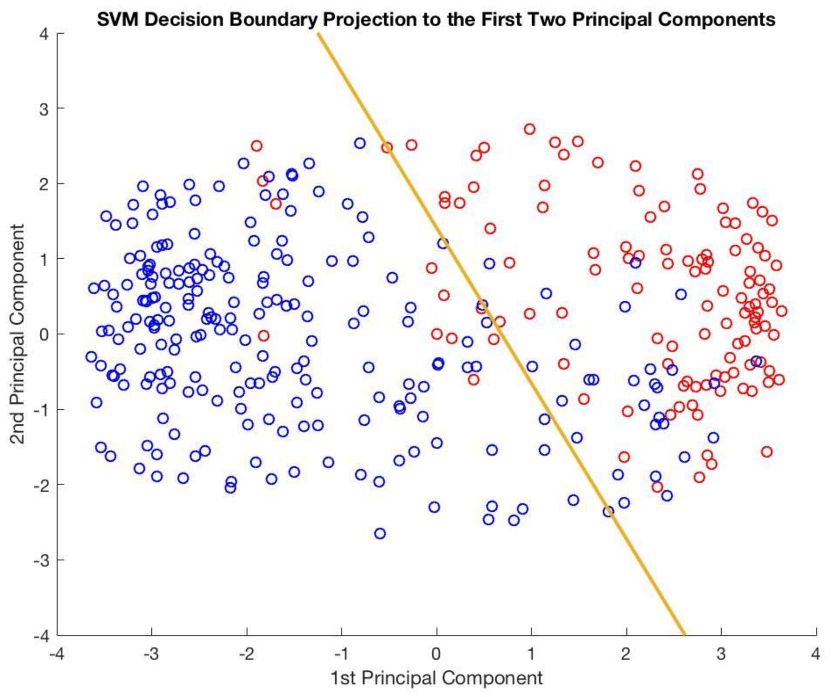

Figure 3

This projected boundary in general would not be the same as the boundary we would get if we perform SVM on the dimensionally reduced data, because we'd be ignoring certain information when we operate on a lower dimensional space.

One interesting observation is that in this dimensionally reduced plot, we have more misclassified data points than if we were to plot the decision boundary and the data points in the original 16-dimensional space. This shows that in lower dimensions we are less able to linearly separate the binary data points than in higher dimensions.

<u>AdaBoost</u>

We performed Adaboost two times. The first time is on the original dataset in 16 dimensions. The second time is on the dimensionally reduced dataset.

In the first instance, after training 80% of the data using AdaBoost, 95.11% of the training data points were correctly classified (331 out of 348). We then used the results from the training data to test the remaining 20% of the data. Out of 87 test points, 97.94% were correctly classified (only 2 points misclassified).

Then we took the whole dataset and reduced its dimension to 2 by projecting the data to the leading two principal component subspace. We obtained the leading principal components by performing SVD on the covariance matrix of the whole dataset.

On the set of 348 training data points, the training accuracy for AdaBoost is 327/348=94.0%
On the set of 87 testing data points, the testing accuracy is 78/87=89.7%

We see that the classification accuracies are lower for the dimensionally reduced dataset, although we can achieve better visualizations of the data and a more compact dataset as a result of the dimensionality reduction.

We performed this reduction in order to better visualize whether the data points were correctly classified. The results of our AdaBoost predictions are displayed in Figure 4 below.
Also, we've sampled 10,000 grid points on in the subspace to figure out where the decision boundary lies.

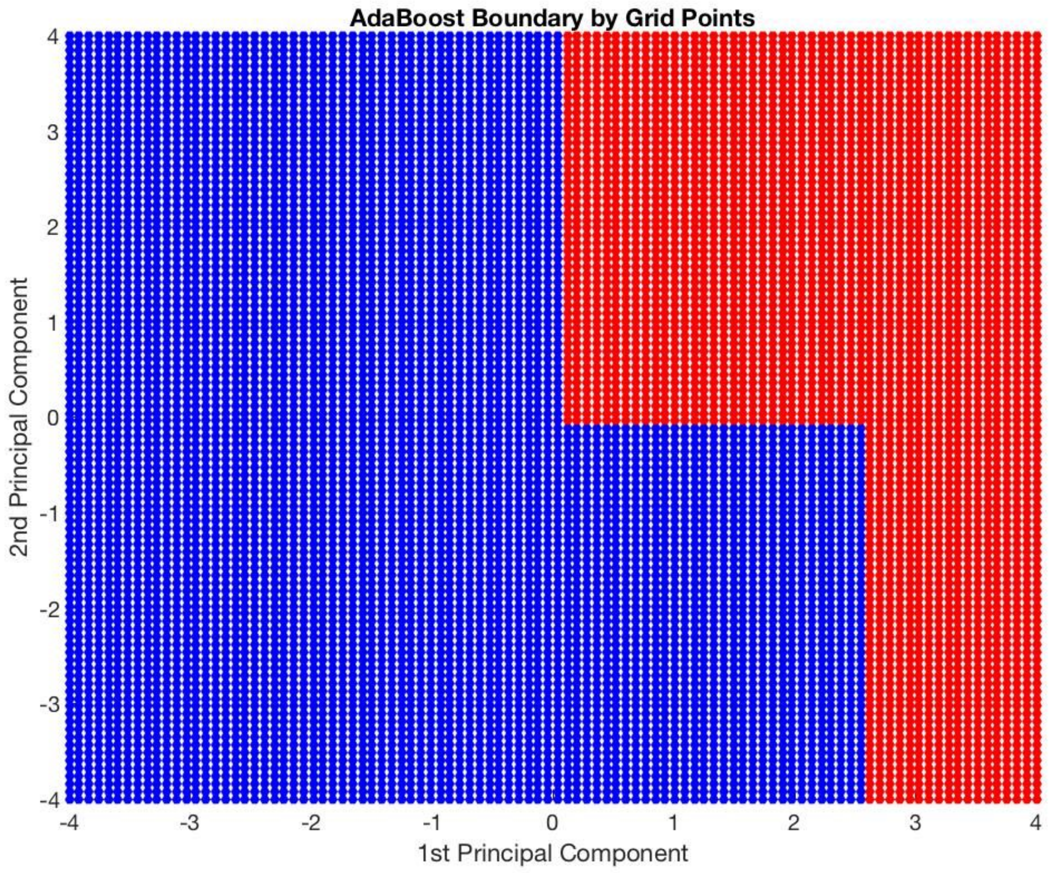

Figure 4

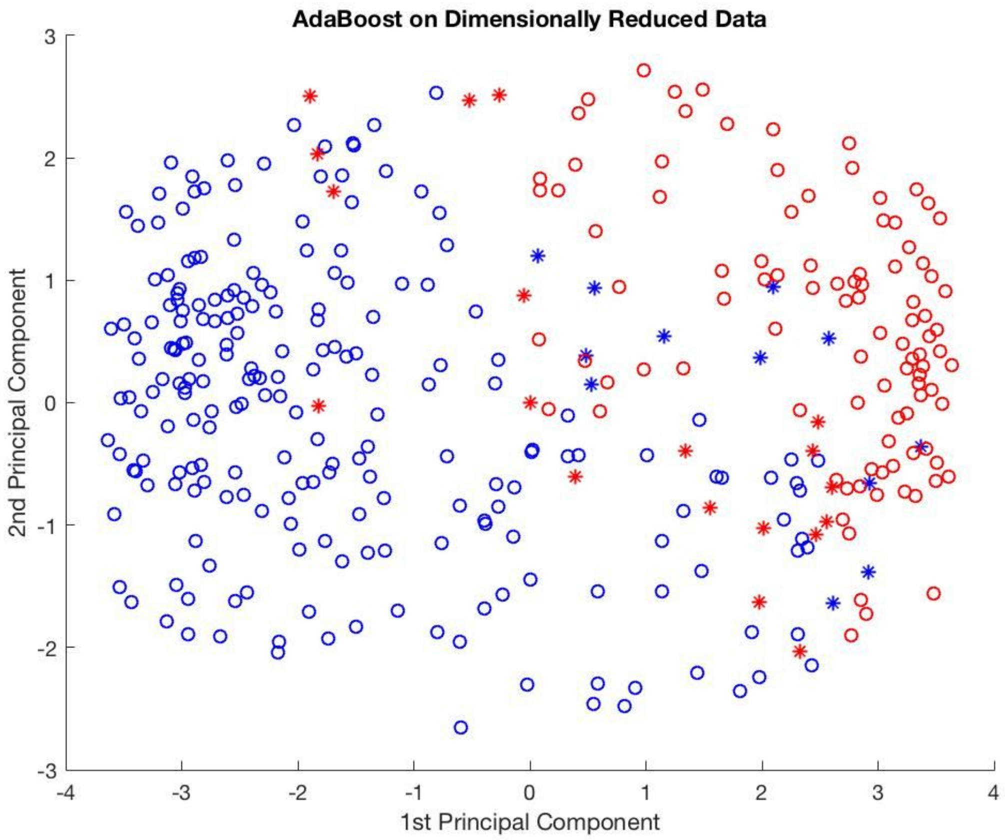

Figure 5

In Figure 5 above, the blue and red colors represent the predictions, blue for Democrat, red for Republicans; a circle means the prediction is correct and an asterisk means the point is misclassified. This plot is done on AdaBoost for the entire dataset. We did this plot on the entire dataset because we already knew the accuracy of the model.

<u>Comparison between SVM, AdaBoost, along with PCA</u>

In the case where we didn't reduce the dimension of the data, AdaBoost had an accuracy of 97.9% on the testing dataset, whereas SVM had an accuracy of 95.4%.

In the case where the data was reduced to the leading two principal components, AdaBoost had an accuracy of 89.7%, whereas SVM had an accuracy of 88.5%.

We see that in both cases, AdaBoost outperforms SVM by a certain amount. But while SVM has a lower accuracy in terms of prediction, it gives us more information by providing us with the weights associated with each vote and the offset, so that we have an explicit description of the decision boundary, whereas we only managed to find the decision boundary for AdaBoost in the 2-dimensional subspace.

A Slightly Philosophical Rambling

If we are only interested in making accurate predictions, we should rely on AdaBoost. But if we want to understand more why the predictions are made and what the underlying reasons are for those predictions, then we should explore the data with SVM, because it would yield more structural information about the data in higher dimensional spaces.

PCA is a great tool in reducing the dimensions of the data. Dimensionality reduction would speed up computations and make visualizations possible. However, in our case, we only have a meager 435 data points in 16 dimensions, so we can ignore the computational cost issue.

An interesting observation in the case of AdaBoost is that when we increased the number of weak learners the accuracy didn't improve accordingly. Actually, the number of testing errors always stayed the same no matter how many learners we added. In fact, we achieved maximum accuracy with just one weak learner.

This might mean that there are certain testing data points that are intrinsically unclassifiable into the correct labels that we provide. What we mean by this is that if we only look at the structure of the testing data points then the intrinsically correct classifications we would associate those data points with are not the same as the correct labels that originally came with those data points. This makes sense because voting is a human activity and therefore the way people vote may not always conform to a set pattern. Our models' ability to predict rely on the regularity of the behavior of our objects of interest. Indeed, if everybody were to vote randomly, we would not have any predictable accuracy at all. So, if our prediction accuracy showed anything at all, it showed that the congressmen were more or less doing their job in representing their parties.

Dataset Source:
https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records
Retrieved May 19, 2016
Github repository:
https://github.com/AaronZhouQian/SVM_AdaBoost_PCA_on_Congressional_Data