Drug Sensitivity Prediction for Cancer Cell Lines

EN. 600.438 Computational Genomics: Data Analysis
Final Write Up

Members:
Yu-chi (Kiki) Chang
Mariya Kazachkova
Min Geol Joo (Kevin)

Introduction:

Drug sensitivity prediction is an essential goal of research in translational sciences. This can be useful in precision medicine, in which doctors and healthcare providers can know which drug is most effective and how effective the target drug is given a patient's genetic information. In this project, we focus on predicting drug response based on genomic profiling data sets measured in human cancer cell lines. We attempt to predict the sensitivity of AEW541 drug given a cancer cell lines' SNP array data or gene expression data. Since AEW541 is a kinase inhibitor drug, we used its IC50 value, the drug dosage needed to reach 50% tumor growth inhibition, as the indicator of the drug's sensitivity and effectiveness. We analyze the drug's IC50 value against a total of ~400 cell lines using various regression models and compare and assess the fit quality of all models. Our result suggests that linear regression model is the best model for SNP array data, and Lasso is the best for gene expression data. We also found that gene expression data provides much better predictive power of the drug's sensitivity given an individual cancer cell line than SNP array. This study suggests methods for utilizing genomic data for drug sensitivity prediction. Given more clinical data, the project can be expanded upon predicting other drugs' sensitivity for a given cancer cell line.

Related Work:

We were able to find a few articles that dealt with using gene expression data in order to predict the sensitivity of cancer cell lines to anticancer drugs. One article in particular made use of IC50 values (just as we intended to do) and gene expression data to examine relationships of a specific gene with the sensitivity of a cancer cell line to a specific anti-cancer drug. This approach used Spearman's rank correlation coefficients in order to find correlations between specific genes and resistance to various inhibiting drugs. Their method had previously been used to identify various gene expression correlations with resistance to BRAF(V600E), an inhibitor drug [1].

Another interesting article we found dealt with using the Cancer Cell Line Encyclopedia to help predict anticancer drug sensitivity. This particular lab used a categorical modeling approach that used both a categorical modeling approach (with Bayes classification) and a net regression analysis. Furthermore, this lab used 10-fold cross validation in order to evaluate performance (an idea that we also chose to use) [2].

Note: While we were looking up gene array data we also found a SNP array dataset from the same lab. This is how we decided to also include SNP data in our project and compare it to gene data. We were unfortunately unable to find articles that tried to do anticancer drug sensitivity prediction using SNP data (and given our results it seems that there may be a reason behind this).

<u>Data:</u>

Drug Info

Compound	Target(s)	Mechanism of action	Class	Highest Phase	Organization
AEW541	IGF-1R	IGF-1R Inhibitor	Kinase inhibitor	Preclinical	Novartis

We used three data sets for this project: GSE36138_series_matrix (SNP data), GSE36133_series_matrix (gene data), and CCLE_NP24.2009_Drug_data_2015.02.24 (cancer cell lines with corresponding IC50 values). The SNP and gene array data was collected from homosapians. Our original data had the following dimensions: SNP data -- 22,419 features by 947 cancer cell lines, gene data -- 18,926 features by 917 cancer cell lines, IC50 -- 504 cancer cell lines by one IC50 value (note, this data set actually had around 504 cell lines per each of 24 drugs, but we only used one of the drugs; noted above). This posed a problem in that there was not a consistent set of cell lines through all of the data. The cell lines needed to be matched up with one another in order to put our data into a format that could be useful, so since the cell lines were labeled in all three data sets we knew we could use the names to match them up. This, however, was made trickier by the fact that the IC50 data gave the cell lines as actual words (ex: 1321N1_CENTRAL_NERVOUS_SYSTEM), whereas the SNP data sets and gene datasets used a different format (ex: GSM887898 and GSM886835, respectively). To solve this and make our data usable we used the following process:

1.) Use BeautifulSoup (library for doing HTML scraping in python) to navigate to the NCBI website (specifically https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc= followed by the cell line in question from the SNP and gene data set)
2.) Pull all of the text off of the html page and parse in order to pull the title and primary site off of the page (this is the information needed in order to match up with the way cell lines are labeled in the IC50 data set)
3.) After uniforming the way cell lines are are named create three new datasets that will be used for the rest of the project (these new datasets are SNP data, gene data, and IC50 values with the same number of cell lines and where row i in each dataset corresponds to the same cell line across all three datasets).

Data sources:

SNP: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE36138 (series matrix file at bottom.This file is included)

Gene: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE36133 (series matrix file at bottom. This file is included)

IC50: (this file is included)
https://portals.broadinstitute.org/ccle/data/browseData?conversationPropagation=begin

Methods:

Preprocessing:
We split the 393 samples into 314 samples for the training set, and 79 samples for the test sets using Scikit-Learn train_set_split method. For SNP array data, each cancer cell line (sample) has 22419 SNPs (features); for gene expression data, each cancer cell line has 18926 features. The datasets being provided have been normalized, ence no normalization step was done in this project.

Feature Selection:
In an attempt to improve the fitting quality of the predictive models, we want to remove uninformative features in our datasets. To this end, we used the Scikit-Learn Variance Threshold method to remove extreme values, since features that have low variance value suggest that they remain nearly constant throughout all samples, and features that have high variance may indicate technical noise.
We select features by observing variance vs samples charts (Figures 6, 7). For SNP array data, we exclude the top 50 features by variance, and select the next 5450 ranked by variance in descending order; for gene expression data, we selected the top 6000 features ranked by variance in descending order. We then construct new training set and test set with the selected features, reducing the dataset dimensions while preserving all samples.

Principal Component Analysis (PCA):
We ran PCA after feature selection on our SNP and gene expression training data to reduce dimensionality and the impact of noisy data. We chose the top 30 principal components (ex: SNP array was originally 314 x 22000, and 314 x 30 after PCA). The same was done to our test sets, to project them onto the same principle components as the training sets.

Regression Model Training:
We explored various regression and generalized linear models, including linear regression, ridge regression, lasso, and support vector regression. We select parameters that gave the best possible R^2 value. For each model, we calculate the R^2 value and the 10-fold cross validation scores for data before and after features selection and PCA.

10-fold Cross Validation:
To assess the predictive quality of the training models, a cross validation (CV) technique was used. A total of 314 cell lines were provided in the training set of drug response. The cell lines were split into 10 disjoint sets of approximately equal sizes, with one set being designated the test set and the remaining cell lines in the training set. This is done such that each set has an equal chance of belonging to the test set.

Model Prediction Testing:
We calculated the mean squared error of the prediction for each model and each dataset for a total of 8 mean squared error values.

<u>Results:</u>
We separated our data into a training set and a test set. 80% of the data was allocated to training, and 20% to test using Scikit-Learn's train_test_split method for both SNP array data and gene expression data. After running linear, ridge, lasso, and support vector regression on these two datasets, we were able to get fit quality scores (R^2 value) for each model on each dataset (Figure 1).

We see a high degree of similarity between the scores for each regression, but it was concerning to see negative R^2 values. Similarly, when performing 10-fold cross validations on each datasets, the validation accuracy scores are all negative values. (Figure 3). After exploring possible sources of this error, it seemed the most reasonable to conclude that our model was fitting the data worse than a constant function (i.e., a horizontal line). The most likely reason for this seemed to be the presence of noise in the data, and the possibility that we were overfitting the data with its noise. To tackle this problem, we decided to try feature selection by removing features with extreme variances. The first step was to get a holistic picture of the distribution of variances, which we saw in Figures 6 and 7.

Based on these figures, we decided to do feature selection as mentioned in our methods. We then proceeded to run PCA to project our data onto the top 30 principal components. We re-trained our models on the post-selection data and overall ended up with better results. The pre-selection $R^{2.}$ values can be seen in Figure 1, which are mostly negative. Figure 2 shows that post-selection, several of the R^2 values become positive, and most of one still negative are closer to 0. Figures 3 and 4 also show that the 10-fold cross validation score improved after features selection, for almost all results. These improvements seems to support our initial suspicions that our models were overfitting noisy data.

During training the data, we improved the robustness of lasso by testing different alpha values and finding the optimal one. A higher alpha value is more suitable for datasets that have higher dimensionality, and this affects the reductions that lasso conducts.

We then took our newly fitted models and ran prediction tests on them with the post-PCA test datasets. Mean squared errors for the prediction results can be seen in Figure 5. The best model for SNP data is linear regression with a MSE value of 6.4136037562, and the best model for gene expression data is lasso with a MSE value of 6.80620129491.

The gene expression data showed more clear distinctions in the MSE values, which suggested to us that Lasso had a measurable impact during its coefficient reductions. Lasso tries to minimize the sum of squared errors, bound on $\sum|b_j| <= s$, for a tuning parameter s. It reduces the coefficients, usually to zero, depending on the significance of the feature. We believe the reason why Lasso worked best for the gene expression data is related to feature selection. About 6,000 features were selected for use from 18,000 in the gene expression dataset, and there is still probably noise or less significant data that is being fit. Our hypothesis is that Lasso, through regularization, was able to create a less overfitted model that ended up being able to better predict our test data. For SNP data, the differences in the MSE for linear regression, ridge regression, and Lasso were relatively small. Similar to our explanation for gene expression, since about 5,000 features were selected from 22,000, regularization might have been unnecessary since the features selection was more selective compared to SNP data. Thus, both ridge regression and lasso might not have been good models for the data. Our result did not match our expectations based on what we learned in class, as we still expected lasso to be the best model here. One possibility we considered is that a different tuning parameter *s* would have created a more appropriate coefficient matrix that would have made our model have greater prediction accuracy.

Conclusions:

It is difficult to conclude anything regarding the relationship between drug sensitivity and cancer cell lines due to the low quality of our fits and prediction tests. We attribute a lot of these missed expectations on the nature of our data as well as the robustness of our model fits.

There were several difficulties in obtaining and processing the data. While searching for data we found ourselves looking through numerous datasets that were not particularly well labeled and often struggled to understand what the data represented. Additionally, properly using the html scraper took a while to figure out, and we felt that it made our project feel more like tedious data manipulation than a machine learning project (this being said, it also appears that tediously manipulating data is a large part of research).

Overall, this project was a good way to apply several concepts we learned in class on a real dataset and a real problem. Although our results were inconclusive, it was highly interesting to learn the process from obtaining data to training and testing our models and analyzing the results. Furthermore, we learned about the importance of trying several adaptations during each step in the process, and the flexible nature of this research. As students aiming to be future biostatisticians, this felt like a great introduction to the field.

Course Project Statements:

This project did not relate to any sort of outside research projects by any of the members of the group.

Individual contributions:

Kiki: Feature selection, PCA, model selections, training models on pre-feature selection data, and analysis

Mariya: Data processing (html scraping), contributions to model-making process as well as testing

Kevin: Feature selection, adding to model training and testing, analysis of results

Acknowledgements

We would like to thank Benj Shapiro for explaining numerous concepts to us during office hours as well as Professor Battle for teaching this class.

Figures

	SNP	Gene Expression
Linear Regression	-0.901256804495	-0.172076965315
Ridge Regression	-0.894554561435	-0.171934191368
Lasso (alpha =1)	-0.000285941760211	0.0395188856111
Support Vector Regression	-0.272463170758	-0.173651354437

Figure 1: Fit quality score before feature selection, as R^2 values

	SNP	Gene Expression
Linear Regression	0.0717900634794	-0.0197331293489
Ridge Regression	0.0717888186608	-0.0197299268899
Lasso (alpha =1)	0.0199087570419	0.0149713153409
Support Vector Regression	-0.223260236884	-0.233322249723

Figure 2: Fit quality score after feature selection, as R^2 values

	SNP	Gene Expression
Linear Regression	-0.92 (+/- 0.73)	-0.18 (+/- 0.36)
Ridge Regression	-0.91 (+/- 0.73)	-0.18 (+/- 0.36)
Lasso (alpha =1)	-0.05 (+/- 0.07)	-0.03 (+/- 0.11)
Support Vector Regression	-0.32 (+/- 0.44)	-0.17 (+/- 0.38)

Figure 3: 10-fold Cross Validation Score before feature selection

	SNP	Gene Expression
Linear Regression	-0.11 (+/- 0.23)	-0.02 (+/- 0.42)
Ridge Regression	-0.11 (+/- 0.23)	-0.02 (+/- 0.42)
Lasso (alpha =1)	-0.03 (+/- 0.11)	0.05 (+/- 0.30)
Support Vector Regression	-0.22 (+/- 0.23)	-0.22 (+/- 0.23)

Figure 4: 10-fold Cross Validation Score after feature selection

	SNP	Gene Expression
Linear Regression	6.4136037562	7.0459967852
Ridge Regression	6.41361235746	7.04597465734
Lasso (alpha =1)	6.77208531167	6.80620129491
Support Vector Regression	8.4522974183	8.52182238329

Figure 5: Mean Squared Error after feature selection

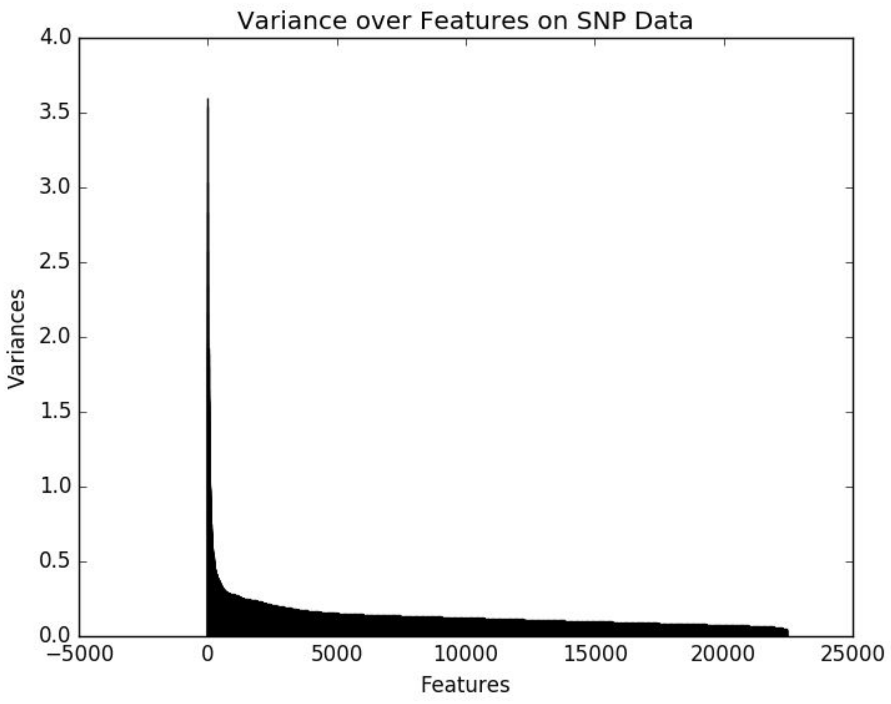

Figure 6: Features of SNP array data sorted by variance. The sharp change in curvature around 1000 led us to further investigation in Figure 3.

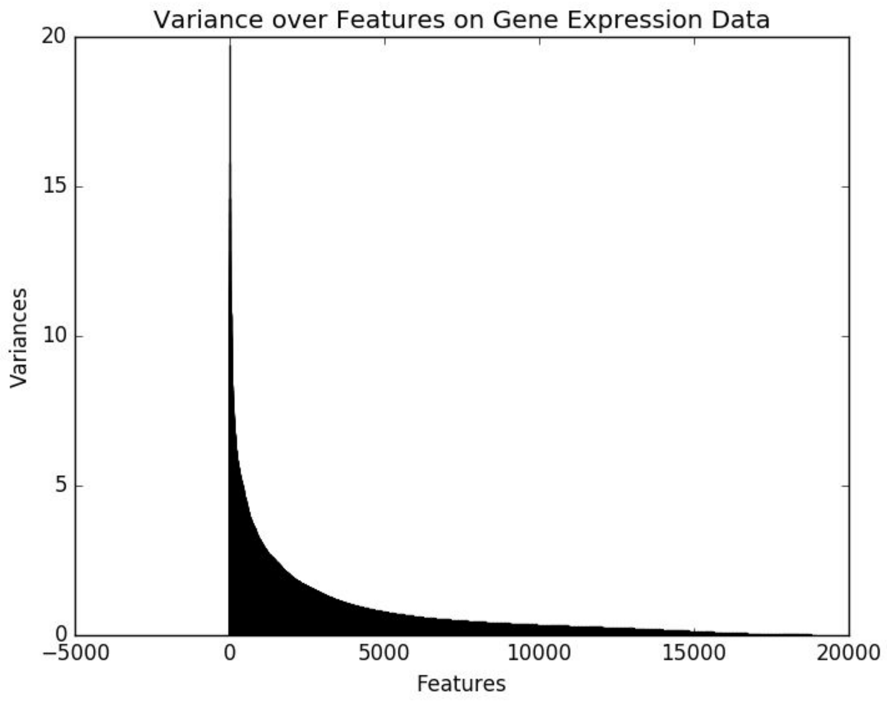

Figure 7: A closer look of Figure 2. It seems that the first 300-400 features seem to be the most interesting,

References:

[1] http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0176763#sec005
[2] http://sagecongress.org/WP/wp-content/uploads/2012/04/CellLineEncyclopedia.pdf