Final project Drug Sensitivity Ranking

Akhan Ismailov
Department of Computer Science
HKUST
404akhan@gmail.com

Abstract

In this project, I will show methods and ideas I used for making 95.533% accuracy on Drug Sensitivity-3 kaggle contest with username 404akhan.

1 Outline

In report, first I will describe the data. Show performance of basic linear method trained with gradient descent. Introduce a new idea for generating relevant features. Show how unsupervised clustering of data and separate prediction on each cluster improves performance. Then exploit transitive property of ranking $a > b, b > c \Rightarrow a > c$ by completing acyclic graph and generating new training data.

2 Description of data

The dataset provides 265 Drugs with experiments on 990 cancer cell lines. Drug sensitivity is measured by IC50. The training data file contains 3M pairwise comparisons of drug sensitivity in the following format: every row has 4 columns as "CellLineID Drug1 Drug2 Comparison", "C683665 D211 D221 1". Comparison takes value -1, 0, 1 which stands for Drug1 less/equal/more sensitive than Drug2 for cell line id. Among the 3M samples, 1/5 (.6M) comparison values are withdrawn from the training data, the purpose of this contest is to predict those .6M comparison values. It also provides cell line features, where every cell line is described by binary genetic features of dimensionality 1250. We remove 187 features that only take value of zero, and convert a feature cancer type that takes 32 values, given as string to one hot representations of 32 values. We end up with number of features for cells being 1095=1250-187+32. Training data split into 1,920,000 training data and 480,000 held out validation, with ratio 4:1.

$$total\_drug = 265, total\_cells = 990, total\_features = 1095$$

3 Linear model, gradient descent

I use $\theta(i) = \beta_0(i) + x(k) \cdot \beta_1(i)$ for drug ranking score. Where k is cell line id with feature vector $x(k)$, i is drug id with coefficients vector $\beta_1(i)$ and bias $\beta_0(i)$ that need to be learned. $0 \leq k < total\_cells$, $0 \leq i < total\_drug$. Size of $x(k)$ initially given 1095, later will be added more features. Pairwise comparison value $y(cell\_line_k, drug_i, drug_j) \in \{-1, 0, 1\}$ is modeled by $F(\beta_0(i) - \beta_0(j) + x(k) \cdot (\beta_1(i) - \beta_1(j))$, where F chosen to be tanh. The loss function for this model is mean square error, which is differentiable with respect to $\beta_0(i), \beta_1(i)$ for any i. Thus we minimize loss with mini-batch gradient descent with batch size 10000 and exponentially decayed learning rate, with ten epochs (complete iterations over data). During prediction if tanh value > 0 we predict 1, else -1. We do not predict 0 at all, which is only 0.18% of the data. This model gave 88.311% accuracy of categorization on test set.

4 Generating features

After above linear model, that had number of parameters equal to $total\_drug * 1095$ seven times smaller than training set size. I generated new features by multiplication of given. Where I create multiplication by multiplying only important features, where I measure importance of feature f_j, by looking at $\sum_{i=1}^{n} |\beta_1(i)(j)|$. Which stands for each drug i that have coefficient $\beta_1(i)(j)$ for feature j, we take absolute value and sum over all drugs i. The higher that value the more important that feature is. If we initially had 1095 features, looking at top 70 features, measured by importance defined above, producing new 2415=70*69/2 features by multiplication, which increased number of parameters to $total\_drug * 3510$, which is 3.5 times bigger. And eventually applying linear model from 3, on increased number of features gave performance 90.98% (+2.6%) on test set. Parameter top 70 resulted from tuning.

5 Cluster specific prediction

After adding 70*69/2 features I left only top 500 important of them, importance again is defined in the same way. Despite performance drop by 1%. I have done it to allow increase of parameters in another way. Here I divide all $cell\_lines$ into 2 clusters with k-means according to there features, so that depending on which cluster $cell\_line_k$ belongs to $y(cell\_line_k, drug_i, drug_j)$ predicted using different $\beta_0(i), \beta_1(i), \beta_0(j), \beta_1(j)$. Which leads to increasing of number parameters by two, which is equal to $total\_drug * 3190$. This lead to performance of 91.4% on held out validation with tuned L1 penalty, which is bigger than model from section 4 with tuned L1 penalty. We will combine this with section 6. Gaussian mixtures as clustering method tried as well and cluster size resulted from tuning.

6 Graph completion

As described we have data $y(cell\_line_k, drug_i, drug_j) = \{-1, 0, 1\}$ which stands for $drug_i$ less/equal/more sensitive than $drug_j$ in cell line k. We will use $a > b, b > c \Rightarrow a > c$ to generate new data from given. So we have directed acyclic graph for each $cell\_line_k$. I checked that it is acyclic, this fact was not obvious from given data, in fact from beginning I thought they could have been $(a > b, b > c, c > a)$ triples due to measurement errors.

So, in directed acyclic graph for any two nodes v_i, v_j that have directed path from v_i to v_j, we need to generate a directed edge from v_i to v_j. Which means we infer that $drug_i$ more sensitive than $drug_j$ for $cell\_line_k$, if there are intermediate inequalities that are given in original data. I used Floyd-Warshall algorithm [1] for fast finding of all pairs that are connected with directed paths. The process generated new data of form $y(cell\_line_k, drug_i, drug_j) = \{-1, 1\}$, which increased amount of data by around 8 times. In fact this was very clean generation of data, when I used this process on training data, some generated edges matched to edges from held out data in fact 60% matched and beautifully all of them were in the same direction.

So, 60% of data from held out of size 480000, can now be predicted with 100% accuracy. Despite that by pure combining these 60% accurate predictions, with model from 5 that is trained on original train data gave only 92.657% on test set which was not as high improvement as expected (+1.3%). After analyzing I found that those 60% accurate predictions have been predicted almost accurately by model from 5 as well, so this oracle 60% predictions did not improve performance much, because linear model was able to figure out most of them as well. Despite that my idea of generation data from this prior knowledge about ranking data will be used to better train the model from section 5.

As said we were able to increase training size from 1,920,000 to 16,000,000. And model from section 5 trained with this additional 8 times more data gave performance on held out set of 94.6% and on test set 94.547% (+3.2% from pure model section 5). And performing all this pipeline not only on just partitioned training data, but on whole gave performance of 95.533% accuracy of categorization on test data, which is the best result in the contest by 21 May.

7 Results

Table 1: Accuracy of Categorization on Test data

Model	Accuracy
tanh + linear	88.31%
tanh + linear + generated features	90.98%
tanh + linear + generated features top500 + cluster specific	91.4%
tanh + linear + generated features top500 + cluster specific + graph completion	95.53%

8 Conclusion

My final model gave performance of 95.533% accuracy of categorization on test data. It is linear over features, uses L1 penalty, uses tanh as converter of value from real line to $(-1, 1)$. That uses an approach I have not seen before for generating features by multiplications of the most important, the ones that gets higher absolute coefficients. That divides cell lines into two clusters with k-means and uses different coefficients for each cluster. That is trained on enlarged training data by 8 times, which is done by completion of acyclic graph. Where training was done with mini-batch gradient descent.

Acknowledgments

This work was done by Akhan Ismailov.

References

[1] Floyd-Warshall algorithm. (2017, April 26). In Wikipedia, The Free Encyclopedia. Retrieved 19:44, May 21, 2017, from `https://en.wikipedia.org/w/index.php?title=Floyd%E2%80%93Warshall_algorithm&oldid=777285627`