Feature Engineering in Machine Learning for Survival Analysis

Lefei Chen, Gary Lai, Ziyuan Wang, Juncheng Yang and Bashan Zuo
{lefei.chen, gary.lai, ziyuan.wang, juncheng.yang, bashan.zuo}@emory.edu

Department of Mathematics and Computer Science, Emory University

ABSTRACT

(by Gary Lai, Juncheng Yang)

Cancer data typically contains huge amount of features but lack enough complete samples. How to obtain useful information from both complete and incompletet data is our focus in this report. Specifically, we address this problem by focusing on data processing and feature engineering in survival analysis, including data pre-processing, dimensionality reduction and feature selection. After data is pre-propossed, we apply and compare different strategies including genetic algorithm, PCA, autoencoder and sparse autoencoder, and we show that genetic algorithm is superior but slower, then we also tried hybrid approach of genetic algorithm with PCA/autoencoder/sparse autoencoder to achieve the balance between efficiency and precison.

1 Introduction

(by Ziyuan Wang)

Feature engineering is a process of generating new dataset from the original one to make machine learning efficient and effective, which is an essential part in machine learning. Typical process of feature engineering includes cleaning, preprocessing, dimension reduction and feature selection. Each process has various strategies to achieve the effect.

2 Background

(by Lefei Chen)

There have been many approaches dealing with redundant and high dimensional data. In feature selection, genetic algorithm a good solution to many clinical problems.

Yang et al.[1] have conducted genetic algorithm to solve the problem of discovering the Core Effective Formulae (CEF) in Traditional Chinese Medicine (TCM). They used binary to represent choosing herbs or not. By measuring fitness with coreness and effectiveness, they generated core herbs in prescriptions.

Jambek et al.[2] have utilized genetic algorithm to assess fetal well-being, in which PCA was first used to reduce dimensionality, genetic algorithm was applied afterwards to choose the best features to discern BabyâĂŹs Fetal Heart Rate(FHR) signals of the cardiotocography (CTG) records.

Anastassopoulos et al.[3] have addressed a similar problem as ours, in which they proposed an Artificial Neural Network to extract diagnostic features on Osteoporosis Clinical Data. They mainly focused on the detection of redundancy in dataset by genetic algorithm.

In dimension reduction, autoencoder, as an unsupervised learning, is a common approach especially in neural networks. Ithapu et al.[4] have designed a deep learning model to predict Alzheimerd́isease, in which randomized denoising autoencoder was used to identify the

biomarker in neuroimaging, which regresses the training data while keeping the variance.

3 Experiments Setup

(by Bashan Zuo, Juncheng Yang)

Our experiments are conducted on two dataset, brain and breast cancer datasets. Each dataset contains features of five categories, including Clinical, CNV, Mutation, mRNA, and Protein. Same pre-processing method has been applied to each categories to guarantee the constant input of different feature engineering strategies. After analyzing each feature type in the datasets separately, we pick out the most important feature types in each category and combine them together to see the performance of features integrally.

Five strategies are utilized to study features, including genetic algorithm, autoencoder, sparse-autoencoder, PCA, and the combinations of feature selection and dementionality reduction. We compare both the accuracy and efficiency of each strategy as the final results, which will be discussed later in Evaluation.

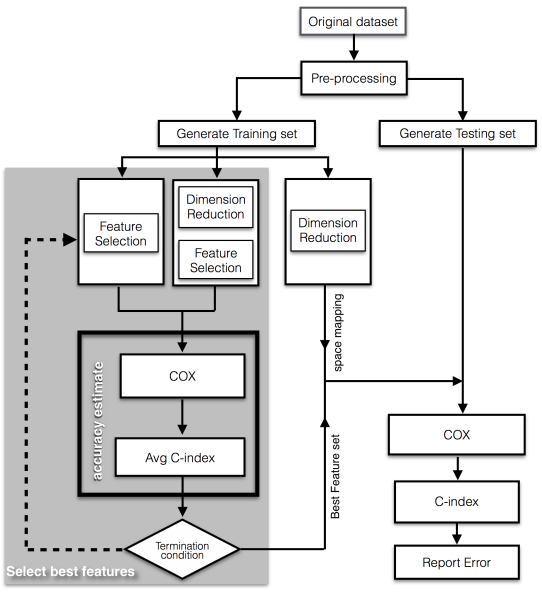


Figure 1: Flowchart of model process

3.1 Vallidation

(by Lefei Chen)

For validation, we use five folds leave-one-out cross validation. The model runs as follow steps. We first divide the dataset, randomly choose 80% as training, while the remaining 20% as testing dataset. Then we part the training set as five folds, taking four folds as training, one fold as validation. We perform the leave-one-out for five times and get the even training result. Then we use the best strategy on the testing dataset, and get the test result. We run the model for five times in total, in order to get a more general result.

3.2 Preprocessing

(by Juncheng Yang, Bashan Zuo)

We used the following procedures to preprocess the dataset. In the first step, we delete all zero variance, and features that have more than five hundred Nans. In the second step, we split the whole dataset into five smaller datasets, each corresponds to a feature type, Clinical, CNV, Mutation, mRNA and Protein. Within dataset of each category, we selectively delete samples with Nans. Instead of deleting samples when there is Nan on the whole dataset, we delete samples within each feature type, thus we can have more samples for each data type.

4 Methods

(by Juncheng Yang, Bashan Zuo)

Our primary method is feature selection, through which we figure out which features are important factors in survival analysis. In the feature selection part, we implement two methods by ourselves: Bootstrap and Stepwise/wrapper/Genetic Algorithm. In our secondary method, we turn to dimensionality reduction, through which the data is projected into a lower dimension as new features to represent the original data. In this work, we use PCA, autoencoder and sparse-autoencoder as our methods for dimensionality reduction.

4.1 Bootstrap

(by Bashan Zuo, Juncheng Yang)

Feature selection is an important part of data pre-processing. The reason of performing feature selection are based on the following two facts. The first reason is the curse of dimensionality. The curse of dimensionality refers to a certain learning algorithms may perform poorly in high-dimensional data, which know as $n << p$ problem. If model building only depends on part of features, then $n << p$ problem will be optimized. Second, weak and irrelevant features may degrade the predictive ability of the model, thus it is important to select features before pipe into any model.

We notice that if we choose one feature at a time and use it to calculate CI score, then in effect, we use CI score as a classifier to determine the effectiveness of a feature. If a feature gives a CI score around 0.50, it indicates this feature is not a decisive feature, thus providing little insight for the final prediction and we can delete it. By deleting all these features, we can reduce the number of dimension of our dataset, meanwhile we reduce the impact of unimportant features which can become a burden on important features when considered equally. In this way, we can filter out a large number of features and provide a cleaner and useful dataset.

4.2 Genetic Algorithm

(by Bashan Zuo, Lefei Chen, Ziyuan Wang)

The genetic algorithm is a simulation of evolution. In nature, good individuals will survive according to the law of natural selection. In this project, we utilized Genetic Algorithm as the heuristic function to select subset features that most adapted to the model. The basic idea of Genetic Algorithm is to selection two best subset features as the parent that have the highest c-score among a group of subset, and use them to produce a new group of subsets which is called children in the genetic algorithm. The producing process involves two mechanisms: crossover and mutation. During the crossover process, we conduct the crossing with another sequence by the sub-sequence from a selected gene index to end. During the mutation process, we reverse the binary value at a randomly selected gene index. The probability of occurrence of these two events is different; crossover is set to happen most of time. Through those two mechanisms, good feature index will be inherited, while the irrelevant feature index will be eliminated. After reaching a certain number of generation, one of the best feature set will be selected among all of the feature sets, we then use the best one for model building. The following flow chart shows the basic process of producing a new population.

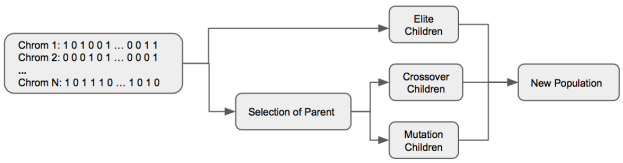


Figure 2: Genetic Algorithm

Table 1 shows the configuration of the GA in this project. Mutation rate and Crossover rate are the probability of those two events happens. Population size determines the number of individuals in each generation. Generation limit is set to be 50 due to computational time constraint. EliteCount is the top number of children with the highest c-Index score value, those children are automatically pushed into next generation.

Parameter	Value
Population size	30
Generation limit	50
Mutation rate	0.1
Crossover rate	0.8
EliteCount	2
Fitness Function	C-Index

Table 1: GA Parameter

To select the best feature index, we divided entire data set into 5 folders. 5 set of best feature index will be generated by performing GA on 4 sequentially selected folders. The final feature index will be selected based on the cumulative number.

4.3 PCA

(by Juncheng Yang)

Principal component analysis (PCA), one of the most important dimensionality reduction methods, is employed in our project to reduce the number of features. As a linear unsupervised learning technique, the goal of PCA is to linear map from high dimensional space where observations are possibly correlated to low dimensional space where variables are linearly uncorrelated such that the variance of the data is maximized.

In our experiments, we tried reducing feature dimensionality to 5, 10, 15, 20. As we shown in Table 2 and Table 3, for both datasets, Clinical features shows superior performance using C-index score as standard. While for other feature types, their C-index scores fluctuate a lot with varying dimensions, and for most dimensions, they perform worse than without reduction and their results can be thought as random and not a single number can be proposed for an ideal dimensionality. The reason for this is that, the features are not very relevant to final results and are not most suitable for PCA.

4.4 Autoencoder

(by Gary Lai)

The architecture of autoencoder is a neural network, in which the output values can be very similar with the input values, if its performance is good. According to Hinton and Salakhutdinov (2016)[5], if the performance of autoencoder is good, the hidden layer can represent the original data well; in other words, the data is projected into a lower dimension containing important information extracted from original data. Now, we know how to reduce the dimensionality of the data. The coming problem is how to decide the dimension the data is going to be reduced? We therefore not only implement the autoencoder method but also want to answer this question in our autoencoder method.

In our project, we build an autoencoder with three layers: an input layer, a hidden layer and an output layer. Between each layer, sigmoid function is used for the activation function. In each training process, we run through 1,000 epochs and use mean square error as our performance estimation.

The next step is to decide the dimension need to be reduced. To decide the dimensionality, we run the autoencoder with the neuron number in hidden layer from 1 to the number of features. After the iteration, we can see the mean square error converge at certain point, and we select the point as the number inside hidden layer. In figure 3, it shows the convergence of each type. The following table 4 shows the dimension we decide.

	Data Source	
Data Type	BRCA	GBMLGG
Clinical	32 -> 15	7 -> 5
CNV	40 -> 20	80 -> 40
Mutation	28 -> 28	52 -> 36
Protein	87 -> 40	67 -> 31

Table 4: Dimensional decided by Autoencoder

4.5 Sparse-autoencoder

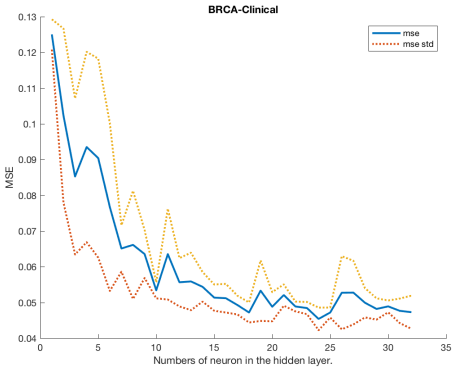
(by Lefei Chen)

In the previous part, we have talked about autoencoder, in which we limit the number of hidden units, compressed the representation of the original data. Now, we turn to wonder if any further restriction could be applied to autoencoder to explore more about the dataset.

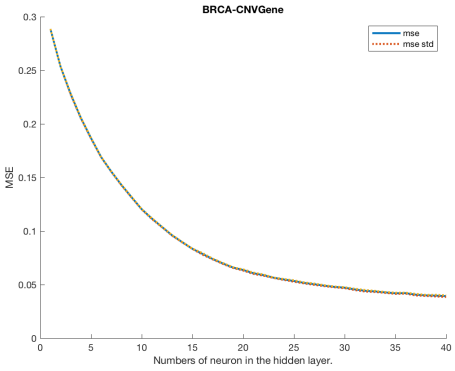
We put constraint on hidden layer that most units should not stimulated when processing information, which means the average activation of each hidden neuron should be a very small figure. For example, if the regulation is 0.05, under which case, the activations of neurons in the hidden layer are almost 0, while only few ones are around 1.

We introduce a $a_k^2(x)$ to denote the activation of this hidden unit when the network is given a specific input x. Further, define $\hat{p_j} = \frac{1}{m}\Sigma[a_k^2(x^i)]$ as the average activation of hidden layer. We like to constrain $\hat{p} = p$.

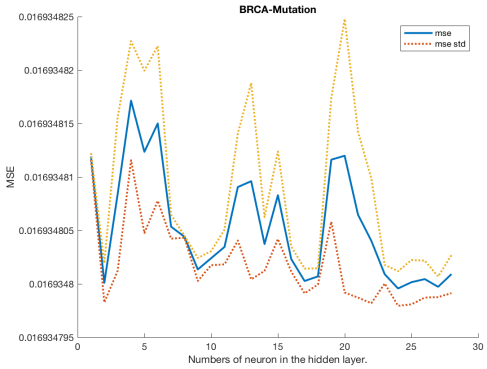
We use Kullback-Leibler (KL) divergence, which is a standard function to measure the difference between a Bernoulli random variable with mean and a Bernoulli random variable



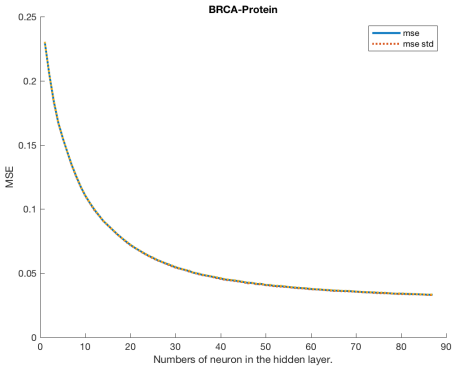
(a) BRCA Data: Clinical



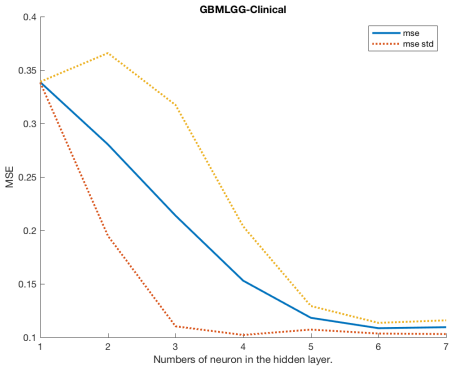
(b) BRCA Data: CNVGene



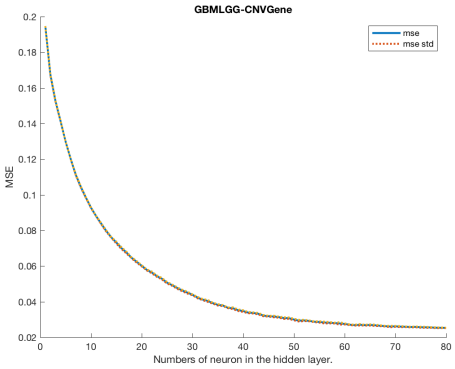
(c) BRCA Data: Mutation



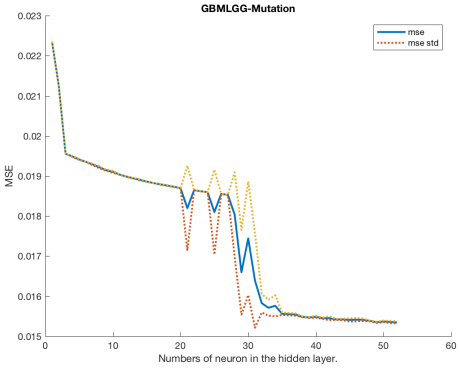
(d) BRCA Data: Protein



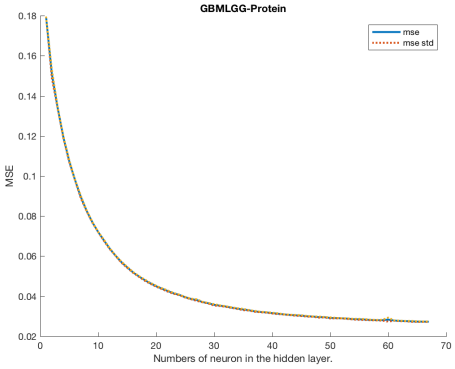
(e) GBMLGG Data: Clinical



(f) GBMLGG Data: CNVGene



(g) GBMLGG Data: Mutation



(h) GBMLGG Data: Protein

Figure 3: Result of Autoencoder

Table 2: PCA: BRCA

Dataset	BRCA			
Feature Type /Dimension	Clinical(32)	Mutation(28)	CNV(40)	Protein(87)
5	0.068	0.086	0.044	0.077
10	0.727	0.595	0.595	0.542
15	0.731	0.607	0.592	0.576
20	0.709	0.613	0.579	0.586
25	-	-	0.580	0.590

Table 3: PCA: GBMLGG

Dataset	GBMLGG			
Feature Type /Dimension	Clinical(7)	Mutation(52)	CNV(80)	Protein(67)
5	0.728	0.502	0.511	0.531
10	-	0.516	0.516	0.524
15	-	0.527	0.511	0.529
20	-	0.533	0.507	0.547
25	-	0.535	0.497	0.538

with mean $\hat{p_j}$.

$$KL(p||\hat{p_j}) = \sum_{j=1}^{s_2} p \log \frac{p}{\hat{p}_j} + (1-p) \log \frac{1-p}{1-\hat{p}_j}$$

In order to keep this sparsity of the hidden layer, we add an extra penalty term penalizing $\hat{p}_j$ deviating significantly from in the cost function with β controls the weight of the sparsity penalty term.

$$J_{sparse}(W,b) = J(W,b) + \beta \sum_{j=1}^{s_2} KL(p||\hat{p_j})$$

In order to keep accordance with autoencoder, we set the same hidden layer dimension as that in autoencoder. Another parameter should be mentioned is the sparsity parameter, which defines how sparse the hidden layer should be. As we can see from table 2, the

	Data Source	
Data Type	BRCA	GBMLGG
Clinical	0.4/32	0.4/7
CNV	0.5/40	0.5/80
Mutation	0.3/28	0.3/52
Protein	0.5/87	0.3/67

Table 5: Sparsity Parameter

5 Evaluation

In the following sections, we provide results from experiments ran on different feature types. We conduct the experiments on both BRCA and GBMLGG.

5.1 Bootstrap

(by Ziyuan Wang)

In the first step, we use our bootstrapping method to filter our features. We have tried on all dataset of categories of two datasets, here we only take as an example. As we can see in the table 5, different features exhibit different CI scores wen being evaluated individually, some of features show high CI index score, like age, while some features show low CI score around 0.50. A CI score of 0.50 indicates the feature does not give much insight to the final prediction. Therefore, we selectively drop these features which have CI score around 0.50. As can be seen later in the section, this method is effective in dimension reduction and feature selections.

		BRCA					GBMLGG				
Method		Clinical	Mutation	CNV	Protein	mRNA	Clinical	Mutation	CNV	Protein	mRNA
o feature selectio	C-index score	0.702249	0.603707	0.540647	0.608988	Not Available	0.730858	0.544566	0.513417	0.530184	Not Available
	std	0.118172	0.0477282	0.0541554	0.071727	Not Available	0.0823538	0.0372471	0.0638936	0.0322798	Not Available
	dimension	32 --> 32	28 --> 28	40 --> 40	87 --> 87	Not Available	7 --> 7	52 --> 52	80 --> 80	67 --> 67	Not Available
GA	C-index score	0.748364	0.63115	0.584562	0.657632	Not Available	0.730858	0.547354	0.530091	0.571428	Not Available
	std	0.0572807	0.0368157	0.073863	0.03609	Not Available	0.0823538	0.0364366	0.0720761	0.0469127	Not Available
	dimension	32 --> 18	28 --> 23	40 --> 15	87 --> 42	Not Available	7 --> 7	52 --> 35	80 --> 43	67 --> 34	Not Available
AE	C-index score	0.677434	0.607398	0.635924	0.587493	Not Available	0.714892	0.513327	0.502347	0.529388	Not Available
	std	0.0608848	0.0530347	0.0668008	0.0712472	Not Available	0.0752878	0.0295687	0.0743885	0.0531355	Not Available
	dimension	32 --> 15	28 --> 28	40 --> 20	87 --> 40	Not Available	7 --> 5	52 --> 36	80 --> 40	67 --> 31	Not Available
SAE	C-index score	0.65545	0.573401	0.558178	0.617607	Not Available	0.650293	0.537437	0.530328	0.526382	Not Available
	std	0.116799	0.0505581	0.0496106	0.0266926	Not Available	0.0929361	0.0278778	0.0425958	0.0273215	Not Available
	dimension	32 --> 15	28 --> 28	40 --> 20	87 --> 40	Not Available	7 --> 5	52 --> 36	80 --> 40	67 --> 31	Not Available
PCA	C-index score	0.73114	0.603831	0.5794	0.619616	Not Available	0.728092	0.542949	0.524111	0.531094	Not Available
	std	0.0657416	0.0477557	0.0627528	0.0287119	Not Available	0.0859935	0.0429919	0.0539169	0.0408591	Not Available
	dimension	32 --> 15	28 --> 28	40 --> 20	87 --> 40	Not Available	7 --> 5	52 --> 36	80 --> 40	67 --> 31	Not Available
AE+GA	C-index score	0.681657	0.541039	0.636068	0.559427	Not Available	0.662029	0.50556	0.493959	0.530704	Not Available
	std	0.0817883	0.0733332	0.0756376	0.0604589	Not Available	0.137867	0.0386674	0.0834081	0.0262651	Not Available
	dimension	32 --> 15 --> 7	28 --> 28 --> 12	40 --> 20 --> 11	87 --> 40 --> 15	Not Available	7 --> 5 --> 2	52 --> 36 --> 4	80 --> 40 --> 21	67 --> 31 --> 13	Not Available
SAE+GA	C-index score	0.681404	0.562342	0.604837	0.578772	Not Available	0.62862	0.535573	0.554267	0.538906	Not Available
	std	0.0736032	0.0640146	0.0324853	0.0516147	Not Available	0.0750076	0.0415947	0.0515888	0.0127673	Not Available
	dimension	32 --> 15 --> 8	28 --> 28 --> 14	40 --> 20 --> 9	87 --> 40 --> 18	Not Available	7 --> 5 --> 3	52 --> 36 --> 17	80 --> 40 --> 23	67 --> 31 --> 20	Not Available
PCA+GA	C-index score	0.721963	0.610601	0.571342	0.628769	Not Available	0.728092	0.52698	0.519936	0.540849	Not Available
	std	0.0878612	0.0842583	0.0907784	0.0170992	Not Available	0.0859935	0.0447712	0.06192	0.0295038	Not Available
	dimension	32 --> 15 --> 8	28 --> 28 --> 10	40 --> 20 --> 10	87 --> 40 --> 18	Not Available	7 --> 5 --> 5	52 --> 36 --> 14	80 --> 40 --> 18	67 --> 31 --> 15	Not Available

Table 6: Model performance

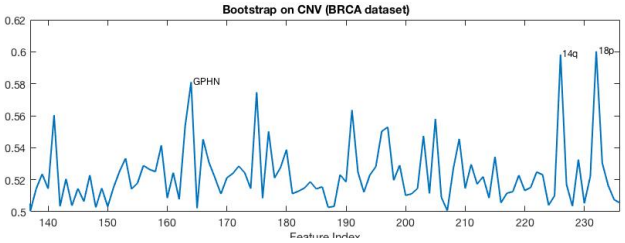


Figure 4: Bootstrap result of CNV dataset in BRCA

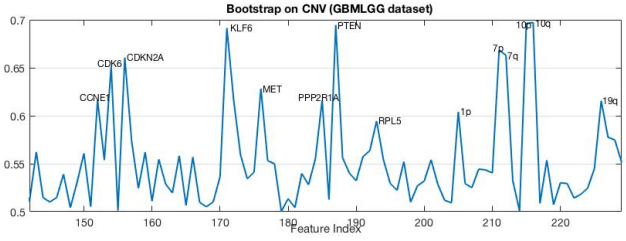


Figure 5: Bootstrap result of CNV dataset in GBMLGG

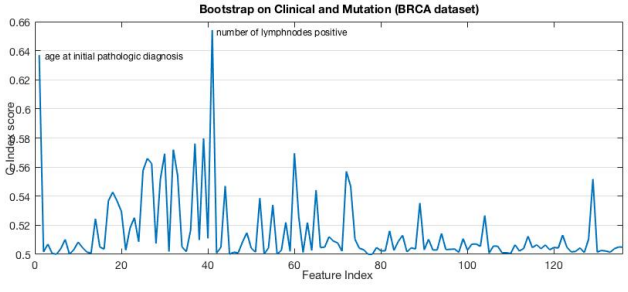


Figure 6: Bootstrap result of Clinical and Mutation dataset in BRCA

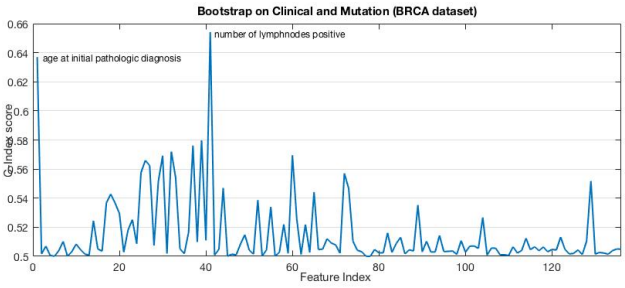


Figure 7: Bootstrap result of Clinical and Mutation dataset in GBMLGG

5.2 Clinical

(by Ziyuan Wang)

For clinical data of BRCA, we get 32 features and 1039 samples after pre-processing. The mean C-index of these 32 features is 70.56. The experiment results with different strategies are as the table 5. Through the genetic algorithm, we choose the most useful n features, and drop the rest, although the rest may hold some useful information. We get the mean C-index of 72.34. For auto-encoder and sparse auto-encoder, the C-index values are lower than 70 all the time. It maybe because that we cannot drop the unuseful datas more efficiently through the autoencoder, but sparse-autoencoder still performs a little better than normal auto-encoder. As PCA, it has the highest C-index when reduce to 10 dimensions, and others' C-index are also high. Therefore, PCA is a great strategy on this dataset. Fi-

nally we try the combination strategy, which is the combination of auto-encoder or other dimension reduction and genetic algorithm in this case. The reason for this strategy is the genetic algorithm cost many times on high dimensional data, so we reduce the dimension first. We can find this strategy can also generate a high C-index. However, we also need more experiments to adjust the both hyperparameters.

5.3 Mutation

(by Ziyuan Wang)

For mutation data of BRCA, we get 28 features after pre-processing. The mean C-index of these 28 features is 60.37. The single GA can get the highest C-index on this dataset. The combination strategy of PCA with GA can also get a high score relatively, which costs less time. In GBMLGG, we have 52 features left. The result of mutation data is slightly better than 0.5, which also means the importance of the mutation in GBMLGG. Moreover, we always get less features left after the combination strategy, like 4, 17, and 14, while more than 30 features left after single dimension reduction or feature selection. Therefore, we can draw a conclusion that the combination strategy can generate conciser datasets because the difference between C-index is under 0.01.

5.4 CNV

(by Ziyuan Wang)

For CNV data of BRCA, we get 40 features and 1078 features after pre-processing. The mean C-index of these 40 features is 55.38. The experiment results with different strategies are as the table. Through the single genetic algorithm, we get the mean C-index of 60.18, which is high in this dataset but lower than the C-index of Clinical data. This is because that the CNV is less important than Clinical in some extent. As the table shows, the three dimension reduction strategies have similar C-index results from 58 to 60. As the combination strategy, we get the best C-index value as 63.07, which is reduced to 20 dimensions by auto-encoder and 11 dimensions by genetic algorithm. In this combination strategy, it takes less time for feature selection, and also includes some useful information of the totally dropped features in single genetic algorithm.

5.5 mRNA

(by Ziyuan Wang)

For mRNA data, we get 6387 features and 1092 samples after pre-processing. Here we only get the PCA results because the high dimension cost too much time. As PCA, we get the highest C-index 56.20, which is not so strong. From this we can draw the conclusion that mRNA is not important or too many noises included.

6 Conclusion

(by Juncheng Yang, Bashan Zuo)

We show that certain feature types are more important than others, like clinical data, which makes sense in our experiment setup. We also show that genetic algorithm as an advanced feature selection algorithm provides the best or similar to the best performance compared to normal feature dimensionality reduction method. However, it has high time complexity, to overcome this shortcoming, we devised several hybrid approaches that combine dimensionality reduction and genetic algorithm, which show superior precision and also provide relatively low time complexity

References

[1] M. Yang and J. Poon and S. Wang and L. Jiao and S. Poon and L. Cui and P. Chen and D. M. Sze and L. Xu *Application of Genetic Algorithm for Discovery of Core Effective Formulae in TCM* 2013: Computational and Mathematical Methods in Medicine.

[2] S. Racindran and A. B. Jambek and H. Muthusamy and S. Neoh *A Novel Clinical Decision Support System Using Improved Adaptive Genetic Algotithm for the Assessment Fetal Well-being* January 2014:Com-

putational and Mathematical Methods in Medicine.

[3] G. C. Anastassppoulos and A. Anastassppoulos and G. Drosos and K. Kazakos and H. Papadopoulos *Diagnostic Feature Extraction on Osteoporpsis Clinical Data Using Genetic Algorithm* 2013:International Federation for Information Processing.

[4] V. K. Ithapu and V. Singh and O. Okonkwo and S. C. Johnson *AuDis: an automatic CRF-enhanced disease normalization in biomedical text* 2014:Med Image Comput Comput Assist Interv.17(01)470-478.

[5] G.E. Hinton and R.R. Salakhutdinov *Reducing the Dimensionality of Data with Neural Networks* July 2006:Science.

[6] H. Li and Y. Luan *Kernel Cox Regression Models for Linking Gene Expression Profiles to Censored Survival Data.*

[7] N. Simon and J.Friedman and T. Hastie and R. Tibshirani *Regularization paths for Cox's Proportional Hazards Model via Coordinate Descent* March 2011: J Stat Softw 1-13.

[8] S. Yousefi and C. Song and N. Nauata and L. Cooper *Learning Genomic Representations to Predict Clinical Outcomes in Cancer* 2016:ICLR.

[9] R. Ranganath and A. Perotte and N. Elhadad and D. Blei *Deep Survival Analysis*2016:Proceddings of Machine Learning for Healthcare.

[10] S. Agrawal *https://github.com/siddharth-agrawal/Sparse-Autoencoder.*