Artificial Intelligence Cource Project

CS410

Ruijie Wang
Shanghai Jiao Tong University
515021910338
Wjerry5@sjtu.edu.cn

Zhenghui Wang
Shanghai Jiao Tong University
515021910120
felixwzh@sjtu.edu.cn

Abstract—**This report is the project report in CS410. In the project, we firstly selected high-quality data with different type of labels and do the dimensionality reduction using PCA and autoencoder. Then we do the multiple classification task and binary classification task using Logistic regression, SVM, random forest, deep forest and deep neural network. In this report, firstly we will introduce the basic theory of our models. Then we will introduce how our experiments are designed. Next, in result and discussion part, we will display our experiment result and compare and analyze the results.**

Index Terms—**gene-classification, machine learning, deep learning**

I. INTRODUCTION

The human genome is the complete set of nucleic acid sequences for humans. Although the sequence of the human genome has been (almost) completely determined by DNA sequencing, it is not yet fully understood. Most (though probably not all) genes have been identified by a combination of high throughput experimental and bioinformatics approaches, yet much work still needs to be done to further elucidate the biological functions of their protein and RNA products. And the popular machine learning methods can solve many biogenome task, which can in turn improve the human knowledge towards AI algorithm.

Some traditional, classical classification models have vertified their ability and robust in supervised learning. Especially in classification tasks which aim to classify examples into given set of categories, classical machine learning like SVM, logistic regression can interpret a specific task into statistical classification. Moreover, some new model such as random forest and deep forest have gained good performance on such kind of tasks.

Our code is available at https://github.com/Wjerry5/SJTU-CS410

Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised. Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics and drug design, where they have produced results comparable to and in some cases superior to human experts.

In the project, we are given some gene and their feature observation, some of which comes from healthy cells and others comes from cancer cells. We firstly selected high-quality data with different type of labels and do the dimensionality reduction using PCA and autoencoder. Then we do the multiple classification task and binary classification task using Logistic regression, SVM, random forest, deep forest and deep neural network. In this report, firstly we will introduce the basic theory of our models. Then we will introduce how our experiments are designed. Next, in result and discussion part, we will display our experiment result and compare and analyze the results.

II. METHOD

In this section, we mainly introduce the machine learning method used in our experiment, including

PCA, autoencoder, Logistic regression, SVM, random forest, deep forest and deep neural network with greedy layer-wise pre training. And the basic experiment idea is also mentioned.

A. *Data Processing*

Since the given dataset is a large p small n problem, dimensionality reduction is required at first. In this part, we will introduce two dimensionality reduction methods: PCA and Autoencoder. Later we compare both their performances and their individual result under different dimension using a same SVM classifier.

1) PCA: Principal components analysis(PCA)[10] aims to perform a linear mapping of the data to a lower-dimensional space in such a way that the variance of the data in the low-dimensional representation is maximized. Suppose we are given data matrix X with size $n \times p$, where n is the quality of sampling and p is the dimension of feature. The PCA algorithm is shown as follow:

- Normalize the data : $x = x - \sum_i^N x_i$;
- Calculate the covariance matrix $C = \frac{X^T X}{n-1}$. Then diagonalize it : $C = VLV^T$, where V is the matrix of eigenvectors and L is diagonal matrix with eigenvalues λ_i .
- Choose d rows ,the sum of whose variance reach some percentage of original variance. The criterion should be followed Eq(1).

$$\frac{\sum_i^d \lambda_i}{\sum_i^n \lambda_i} \geq Threshold \tag{1}$$

The final p is determined by $Threshold$. PCA is based on the idea that the feature has large variance but the noise has low variance.

2) Autoencoder: An autoencoder neural network is an unsupervised learning algorithm that applies back propagation, setting the target values to be equal to the inputs. The structure of autoencoder is show as Fig(1):

Now suppose we have a set of unlabeled training examples $X = [x_1, x_2, x_3, \ldots]$, we want $X = H_{W,b}(X)$. Then the lower-dimension output of hidden layer L2 is the encoder of the raw data. This output is a nonlinear dimensionality reduction, which may overcome the weakness of PCA that it is only a linear dimensionality reduction method.

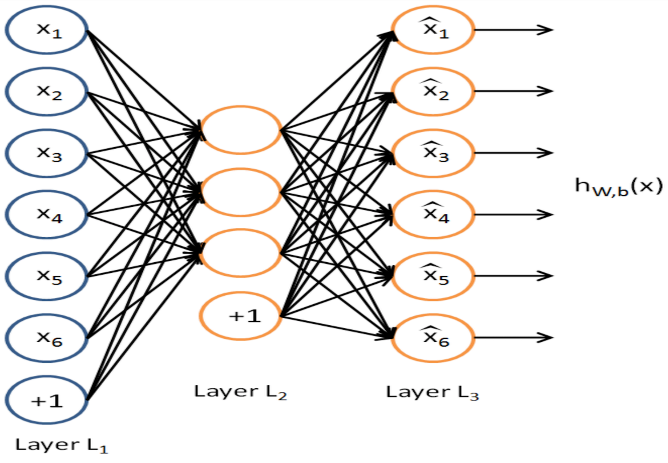

Fig. 1. structure of autoencoder

B. *Classical Methods*

1) Logistic Regression: Logistic regression is a classical supervised linear classifier. Generally speaking, logistic regression firstly calculates the boundary among different classes, then it can predict the possibility of test data class based on the calculated data boundary.

Suppose now that we are given all of the training dataset annotated with labels, while $\mathbf{x}$ denotes the feature vector and $\mathbf{Y}$ denotes label, then the calculated class possibility is defined as Eq(2):

$$P(Y = y_i | x_i) = \frac{e^{\omega x_i + b}}{1 + \sum_i e^{\omega x_i + b}} \tag{2}$$

where ω denotes the regression weights and b denotes the regression bias. Based on the labels training data, we can determine the value of ω and b using Maximum Likelihood Estimation(MLE) method. Assume $h_\omega(x_i)$ denoting the possibilities, the log likelihood function is defined as Eq(3):

$$L(\omega) = \sum_i^N [y_i log h_\omega(x_i) + (1 - y_i) log(1 - h_\omega(x_i))] \tag{3}$$

Various gradient-based optimize algorithm can be used to determine the value of ω and b. In our work, we use logistic regression in both multiple classification task and binary classification task and we compare its performance under various optimization and regularization methods.

C. Support Vector Machine

Support vector machine (SVM) is a supervised leanring model proposed by [4]. Its main idea is to find a hyper plane in the feature space to separate different samples into different categories, Different from other linear model like linear regression, SVM aims to maximize the *margin* between two different categories' samples and the samples on the boundary are so-called *support vectors*.

However, sometimes the smaples are not linear separable, then [5] introduce so-called kernel trick, to project samples with finite dimensions to a high dimension or even infinite dimension space implicitly. More formally, let $\phi(\mathbf{x}_i)$ be the projected high dimension vector and $\mathbf{x}_i$ the original vector, we have:

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) =< \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) >= \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \quad (4)$$

Some of the common kernel functions are as follows:

1) linear kernel:

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j \quad (5)$$

2) polynomial kernel:

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x}_j + r)^d, d > 1 \quad (6)$$

3) RBF (Gaussian) kernel:

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma ||\mathbf{x}_i - \mathbf{x}_j||^2), \gamma > 0 \quad (7)$$

4) sigmoid kernel:

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i^T \mathbf{x}_j + r), \gamma > 0, r < 0 \quad (8)$$

To better tailor SVM for multi-class classification problems, several methods are proposed. [9] use so-called *one-against-all* approach, while [8] use a more robust approach which is called *one-against-one*. Briefly speaking, if we have n classes, then $\frac{1}{2}n(n-1)$ classifiers are constructed. For each individual classifier, it trains data from two classes. There are also many fancy methods to construct a multi-class classifier using SVM like leveraging binary decision tree [11], But we use the one-against-one approach in our experiments.

D. Random Forest

Random Forest [1] is one of the ensemble machine leanring models [16] which could be both applied on classification and regression problems. As it is called ensemble methods, its basic components are decision trees [14] and that the reason why it's called *forest*. An illustration for random forest is shown in Figure 2. We first introduce decision tree and then random forest.

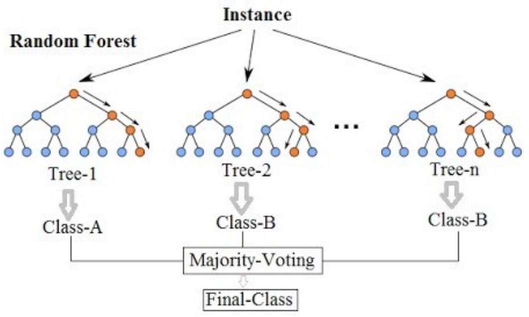

Fig. 2. An illustration for random forest.

A decision tree is a non-parametric supervised learning method which conduct a classification problem by make many decisions in the tree. It learns the structure of the tree by exploring all the possible plit point of every feature, and measure the quality of the split point by some *purity* metrics such as information entropy $Ent(D)$:

$$Ent(D) = -\sum_{k=1}^{|\mathcal{Y}|} p_k \log_2 p_k \quad (9)$$

where D is the current sample set and p_k is the ratio of the k-th class in D $(k = 1, 2, ..., \mathcal{Y})$. Obviously, D is more pure if $Ent(D)$ is smaller. Some other purity metric includes *information gain*, *gain ratio* and *Gini index* in CART (Classification and Regression Tree) [2].

Although we can use some techniques like control the minimal size of leaf node to reduce the risk of over fitting, it is still easy for decision tree to over fit. Thus, random forest (RF) is proposed [1]. It main idea is that the performance of many *weak* classifiers is equal to the performance of a strong classifier. Because the diversity of many weak classifiers boost the whole model's (forest's) generalization ability. More specificly, for each split node of each tree in RF, we do not search all the possible split points to find the *best* split point. Instead, we randomly select some of the features' possible split points to find a possible best split point. This is the most important part of RF, and this is the origin of RF's diversity. More formally, suppose we have d different features for each sample, we only take k features into account when constructing a node. The recommend value for k is $\log_2 d$ [1]. If $k = 1$, we randomly select only one feature. When $k = d$, the tree is a traditional decision tree.

E. Deep Forest

Deep forest (DF) is a decision tree ensemble approach with highly competitive performance to deep neural networks. It is recently proposed by [17]. It is well known that when we talk about the term *deep*, we usually mean many layers of neural networks such as the very deep convolutional neural networks' variant – ResNet [6] or the deep recurrent neural networks in temporal sequence like long short-term memory (LSTM) [7]. We rarely construct some kind of *deep* model for traditional models, especially for those models which do not use a gradient based learning approach like random forest.

However, [17] successfully construct a multi-layer ensemble approach using traditional models – random forest. More specificly, deep forest use so-called multi-grained sacnning (illustrated in Figure 3) to do representation learning. It is similar to the convolution operation but with a decision tree based approach.

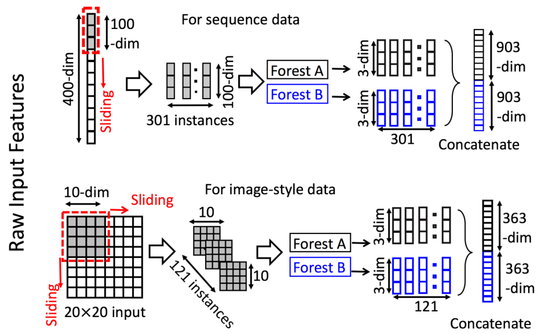

Fig. 3. Illustration of feature re-representation using multi-grained sacnning. Suppose there are three classes, raw features are 400-dim, and sliding window is 100-dim.

After the multi-grained sacnning layer, a cascade forest structure is leveraged to learn better representation in a level-by-level way. Each level is an ensemble fo decision tree forest. And the *diversity* is encouraged by incluing different types of forest like 2 random forests with $k = \sqrt{d}$ and 2 completely-random tree forests with $k = 1$. The split is guided by the *gini* value. The construction of the levels are adaptive. Each level will generate a class vector as a leared representation for next level and each level's input are the origin feature vector concatenated with early level's output class vector. For each level, we evaluate the performance with k-fold cross validation. If there is no significant improvement of the performance, we stop the training. The illustration for both multi-grained sacnning and cascade forest structure are shown in Figure 4, which is the whole model's pipline.

F. Deep Learing Based Methods

Deep learning[15] has been shown as a successful machine learning method for a variety of tasks. In our work, we use a deep neural network with fully-connected multilayer structure both on PCA-processed data and on raw data with **greedy layer-wise pre training**. Besides, we add L2 regularization in each fully-connected layer to avoid overfitting and control the model complexity.

1) network structure: Since the features do not have any locality or direct relationship, we dont use convolutions.

- **nn on pca-processed data**
  In this model, firstly we use pca-processed data (500-dimension) as our network input. Then we add hidden layer1(256-dimension), hidden layer2(512-dimension), hidden layer3(256-dimension), finally a output layer. The structure is shown as Fig(5) network i. For weight initialization, we initialize each neurons weight vector as a random vector sampled from a multi-dimensional gaussian. For bias initialization, we set the bias vectors all to 0. We initialize our weight with the consideration that the size of our network is limited and it can be trained easily according to the network ii.
- **nn on raw data**
  In this model, we use the raw data as

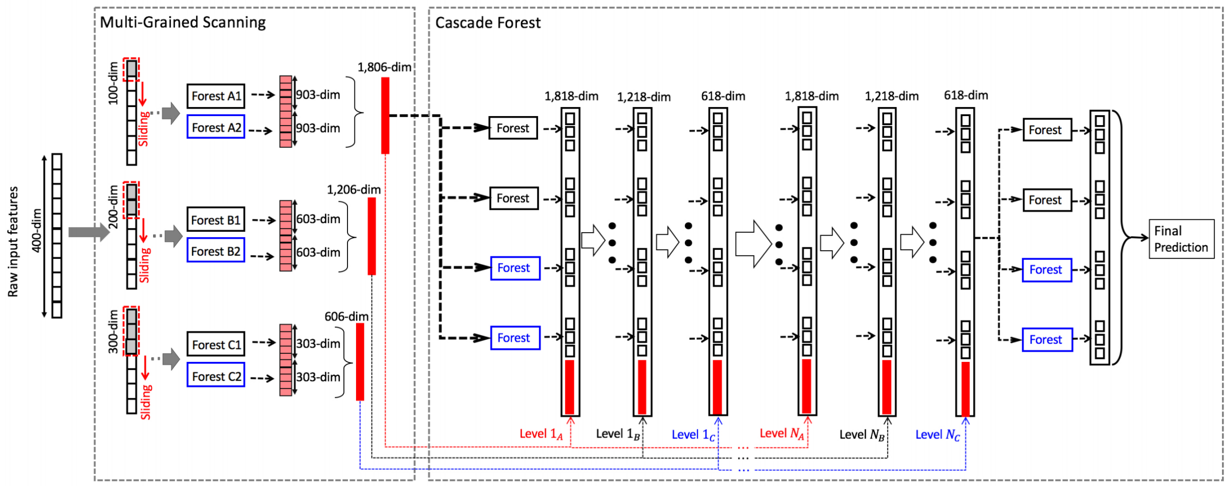

Fig. 4. The overall procedure of deep forest. Suppose there are three classes to predict, raw features are 400-dim, and three sizes of sliding windows are used.

the network inputs. Then we add hidden layer1(8192-dimension), hidden layer2(512-dimension), hidden layer3(128-dimension), finally a output layer. The structure is shown as Fig(5) network ii. Since we did not use the low-dimension data, the training become much harder according to out experiment. So the network initialization is very import. We use greedy layer-wise pre training , which is a pre-train method based on the idea of autoencoder, to initialize the network.

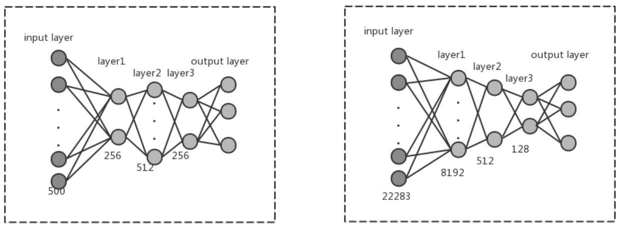

Fig. 5. a.network i; b, network ii;

2) Regularization: We use L2 regularization in each fullt-connected layer.It can be implemented by penalizing the squared magnitude of all parameters directly in the objective. For every weight w in the network, we add the term $1/2\lambda w^2$ to the objective, where λ is the regularization strength. This can help control the capacity of neural networks to prevent overfitting.

Furthermore, we employ Dropout[12] to prevent overfit- ting. While training, dropout is implemented by only keep- ing a neuron active with some probability p, or setting it to zero otherwise.

3) Pre-train: In network ii, the training get very hard because of the high dimension of the network. We use **greedy layer-wise pre training** to pre-train. It will determine the initial weights of each layer layer by layer with the idea of autoencoder. Since each layer has been local optimal solution, the training get much easier .

More specifically, to determine weights of first hidden layer, we use the network shown as Fig(1). Once we get $X = h_{\omega,b}(X)$, it means the hidden layer can encode the input properly, at the same time it can pass these extracted feature to the following layer. Seemingly, when we initialize the second hidden layer, we use the output of the first hidden layer as the input of network in Fig(1). Thus we can determine all the weights. The process is show as Fig(6). The network ivin Fig(6) is we want, so to determine the 3 layers weights, we use network i-network iii(which are same as Fig(1)) to train them.

III. Experiment

In this section, we describe the experiment settings, which includes the data processing, parameter settings for each model and other details.

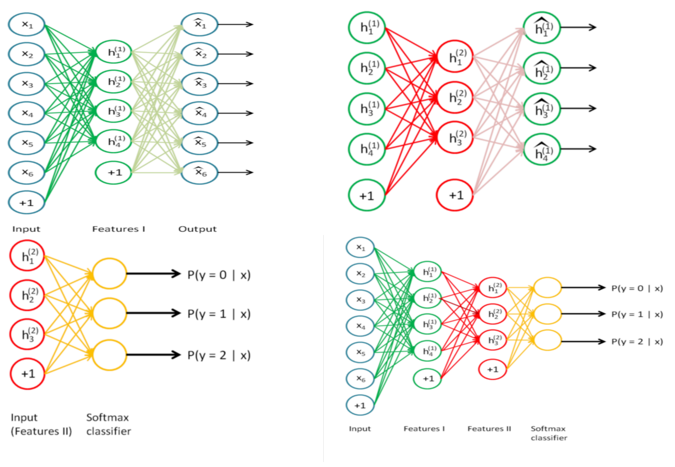

Fig. 6. a.network i; b, network ii; c, network iii; d, network iv

A. Label Selection

Since there many labels in the dataset, we have to choose some of them to be the classification objective. The first label we choose is *MaterialType*, for every sample has this label. We also choose *Sex* (female & male) for it is also very important label though many of the samples' Sex label is empty. The third and fourth labels we choose are *DiseaseState* and *BioSourceType*, and the statistics of this two labels are show in Figure 7 and 8. We can see that both labels are long-tailed distributed, so we only choose those labels which has more than 50 samples.

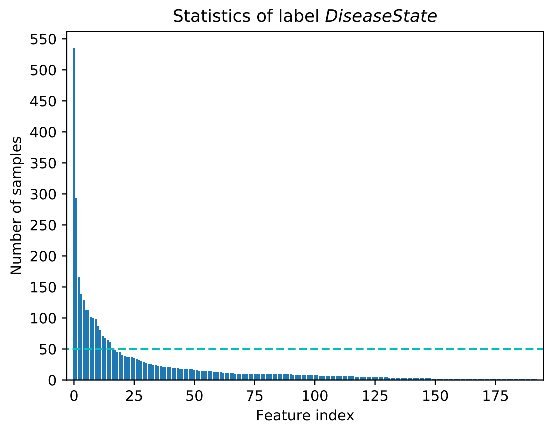

Fig. 7. Statistics of label *DiseaseState*

we choose 4 of the labels as our classification objective. The statistics of the 4 labels are shown in Tabel I

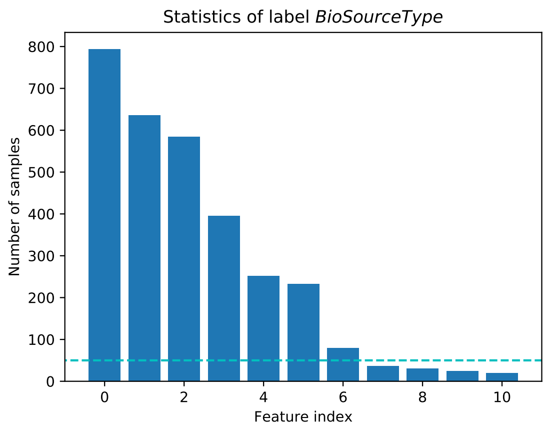

Fig. 8. Statistics of label *BioSourceType*

Label Type	# Category	# Samples
MaterialType	2	5896
Sex	2	1536
DiseaseState	16	3493
BioSourceType	7	2293

TABLE I
STATISTICS OF THE 4 LABELS USED IN OUR EXPERIMENTS.

Besides, we visualize the given data with selected labels into 2-D figure using PCA method, which is shown in Figure 9. By observation, data with Bio-7 label, Material-2 label and Disease-16 have clear boundary while it is very hard to identify the data with Sex-2 label in 2-D space.

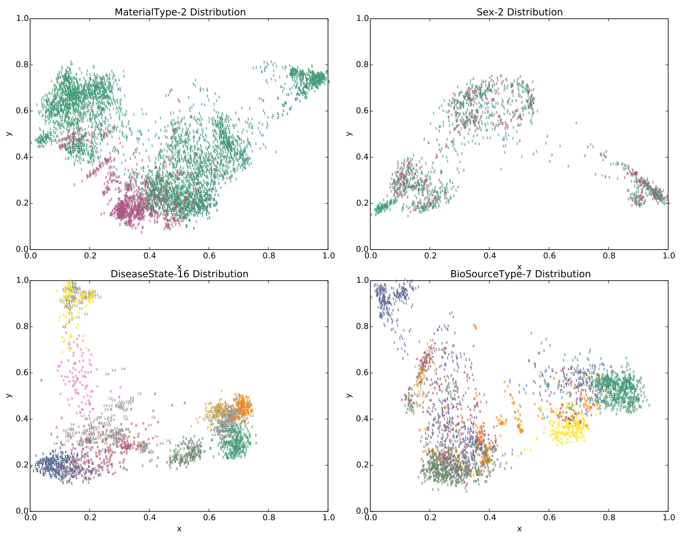

Fig. 9. a.Material-2 Distribution; b, Sex-2 Distribution2; c, Disease-16 Distribution; d,Bio-7 Distribution

B. Experiment Details

1) Cross Validation: In order to get a promising results, all the experiments are conducted with the same 5-fold separation for cross validation.

2) Problem formulation: For label *MaterialType* and *Sex*, since there are only two categories, we conduct binary classification. For label *DiseaseState* and *BioSourceType*, we conduct multi-class classification.

3) Evaluation Metric: As shown in *DiseaseState* and *BioSourceType*, the distribution of different categories is not balanced, so simply use accuracy is not a realistic. Instead, we tend to choose F_1-score as our evaluation metric, because it take both precision P and recall R into account. We have:

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

where TP is True Positive, FP is False Positive and FN is False Negative. And F_1 is defined as the harmonic mean of P and R :

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R}$$

When there are multiple classes, we use so-called Micro F_1-score as the evaluation metric.

4) Implementation: Our deep neural networks is implemented based on *Keras* [3]. The deep forest model is based on the release by [17], and rest of the model are implemented based on scikit-learn [13].

C. Dimensionality Reduction

The original data is a large p small n problem. So we have to do some dimensionality reduction and we employ PCA to do so. We reduced the original data (22283 dimension) to different dimensions (100,200,400,500,1000,1200,2000,3400) and the ratio of reserved variance are shown in Tabel II.

Dimension	Reserved Variance (%)
100	80.64
200	85.43
400	89.33
500	90.47
1000	93.78
1200	94.65
2000	96.99
3400	99.00

TABLE II
RESERVED VARIANCE (%) OF DIFFERENT REDUCED DIMENSION.

To determine the suitable dimension considering both feature representation ability and computation complexity, we evaluate these different dimension data in all the 4 tasks using a SVM classifier with linear kernel. The result is shown in Figure 10 and we find that when dimension is greater than 500, the performance of SVM in all the 4 tasks will meet a bottle neck. So we choose to reduce the original data into 500 dimensions and this 500 dimensions data are used for experiments of SVM, LR, RF and DF.

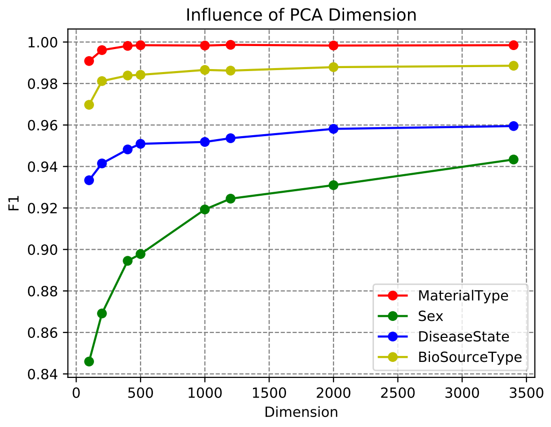

Fig. 10. The performance of SVM in 4 tasks with different data reduced dimensions.

IV. RESULT AND DISCUSSION

In this section, the experiment result and comparison result will be displayed, including the F1 score of each model, the influence of some special parameters in each model and the comparison of different models. Additionally, the required discussion is also included here, mainly analysis about the data distribution and the model performance.

A. Logistic Regression

We apply logistic regression to four task. Besides, we try different regularization method and solver with different regularization strength C . We want to choose the best parameter settings and compare the preformence of different regularization method and solver. The result with the change of C is shown as Fig(11).It can be seen that the performance of

Label Type	$F_1(\%)$	C	solver	penalty
MaterialType	99.86	4	lbfgs	l2
Sex	94.96	0.001	sag	l2
DiseaseState	95.72	0.001	lbfgs	l2
BioSourceType	98.42	0.01	lbfgs	l2

TABLE III
BEST RESULTS AND PARAMETER SETTINGS FOR LOGISTIC REGRESSION.

LR did not vary too much with the change of C, which may mean the data distribution is ideal and the regularization of this model is not necessary.

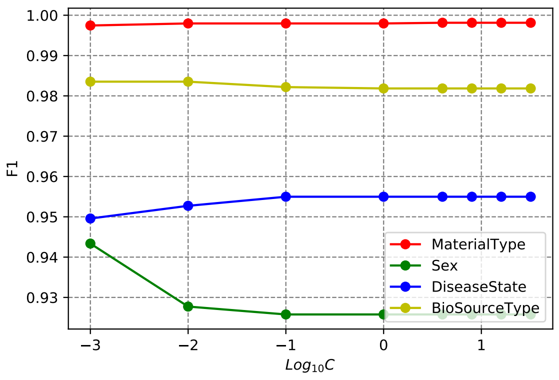

Fig. 11. The performance of LR in 4 tasks with different regularization strength.

And the best results on 4 tasks with selected parameters is shown in Table(III).

B. SVM

We try different kernels (polynomial, RBF and sigmoid) with different C and γ for SVM model, because we want to find not only the best parameter settings but also how the parameters influence the model performance.

We first evaluate the influence of γ in polynomial, RBF and sigmoid kernel and the results are shown in Figure 12, 13 and 14. We can find that polynomial kernel is more stable to γ, while RBF kernel and sigmoid kernel are much more sensitive to γ. This also indicates that the choice of kernel and parameter tuning is vary important in SVM and some kernels (polynomial kernel here) are more robust in some conditions.

We then set γ as the best value and further evaluate the influence of C in different kernels. the results are shown in Figure 15, 16 and 17. We can find that for all the 3 kernels and all the 4

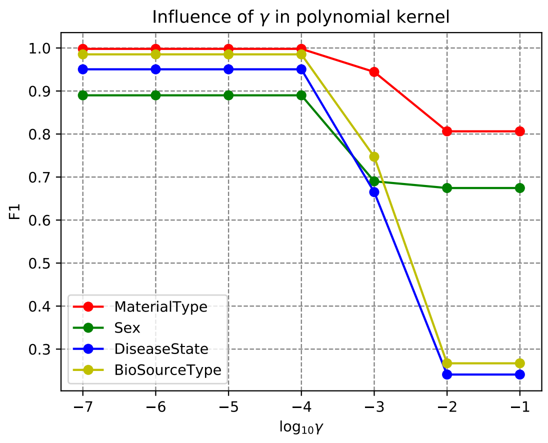

Fig. 12. Influence of γ in polynomial kernel.

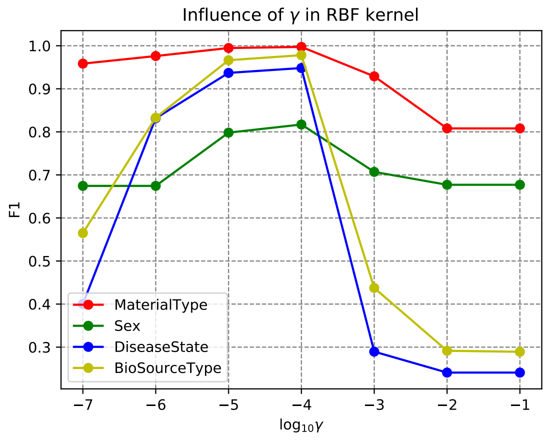

Fig. 13. Influence of γ in RBF kernel.

tasks, when C is heavily small, the performance is extremely poor. This is because when C is very small, we allow many of the sample to be classified to the wrong class, which may hurt model's performance. As C grows, the performance of both polymomial and RBF kernels grows consistently while the performance of sigmoid kernel has a small drop in the end.

After extensive experiments on SVM, we get the best parameter settings for the four tasks shown in Tabel IV.

C. Random Forest

For random forest, we adjust many hyperparameters incluing: maximum feature number, maximum tree depth, minimal samples in split node and minimal samples in leaf node. Extensive experiments show that most of the hyperparameters have less or no influence on the results. We argue that this is

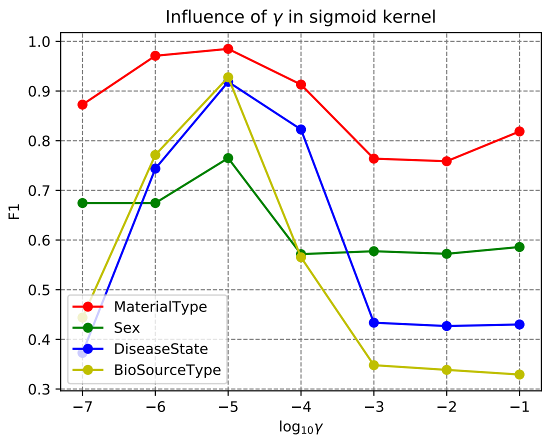

Fig. 14. Influence of γ in sigmoid kernel.

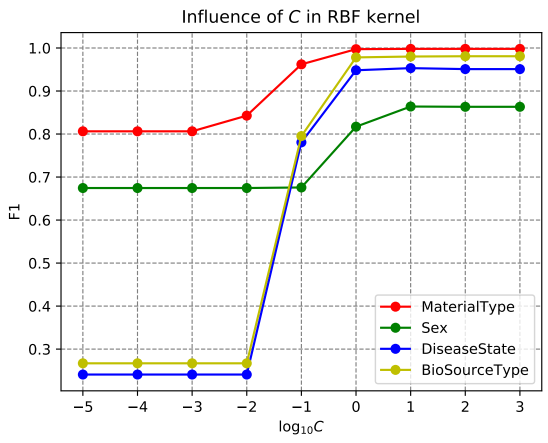

Fig. 16. Influence of C in RBF kernels

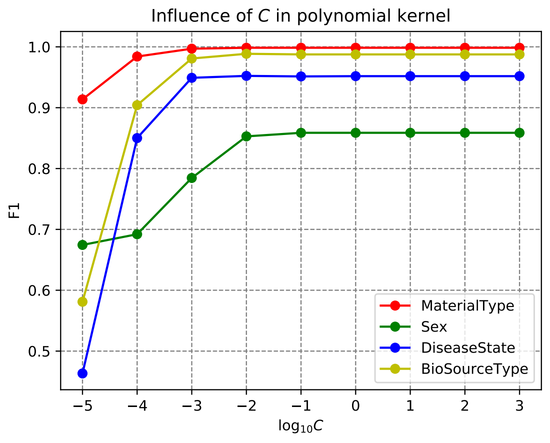

Fig. 15. Influence of C in polynomial kernels

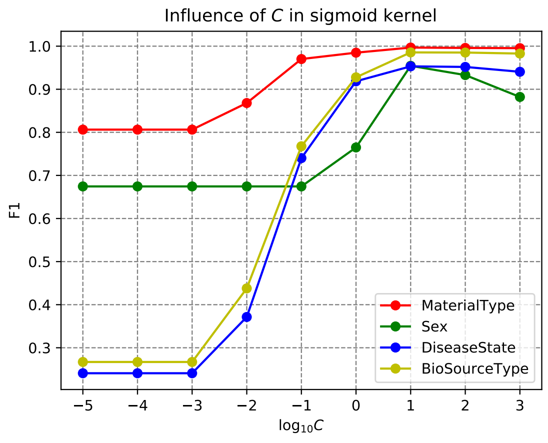

Fig. 17. Influence of C in sigmoid kernels

because the data is quite linear separable, thus such ensemble methods can not leverage its best ability.

Only one hyper parameter have influence on the results, which is minimal samples in leaf node (*min_samples_leaf*) and is shown in Figure 18. We find that with the growing of *min_samples_leaf*, the performance of all the 4 tasks decrease consistently. We think this is because random forest is a ensemble method and each weak classifier (decision tree here) should be somehow overfitted to the data (*min_samples_leaf*), which enhance the whole model's diversity.

The best results for 4 tasks are shown in Tabel V.

D. Deep Forests

Since the features do not have any locality or direct relationship, we only employ the cascade forest structure with 1 random forest (10 trees) and 1 completely-random tree forests (10 trees) in each

Label Type	$F_1(\%)$	Kernel	γ	C
MaterialType	99.85	polynomial	1e-3	1
Sex	95.44	sigmoid	1e-5	10
DiseaseState	95.31	sigmoid	1e-5	10
BioSourceType	98.76	polynomial	1e-3	1

TABLE IV
BEST PARAMETER SETTINGS FOR SVM MODEL.

Label Type	$F_1(\%)$	*min_samples_leaf*
MaterialType	99.35	1
Sex	80.14	1
DiseaseState	94.41	1
BioSourceType	97.20	1

TABLE V
BEST PARAMETER SETTINGS FOR RANDOM FOREST MODEL.

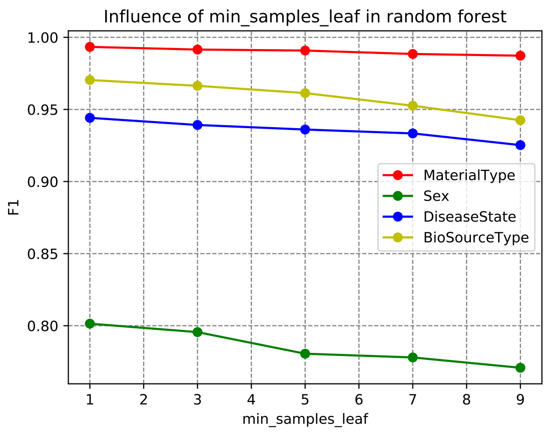

Fig. 18. Influence of *min_samples_leaf* in random forest.

Label Type	$F_1(\%)$	Depth
MaterialType	99.83	5
Sex	92.90	9
DiseaseState	95.58	8
BioSourceType	98.62	5

TABLE VI
BEST PARAMETER SETTINGS FOR RANDOM FOREST MODEL.

Label Type	$F_1(\%)$	learning rate	*batchsize*
MaterialType	99.30	0.001	32
Sex	92.92	0.001	50
DiseaseState	94.38	0.001	32
BioSourceType	95.87	0.001	16

TABLE VII
BEST RESULTS AND PARAMETER SETTINGS FOR NN MODEL ON PCA DATA.

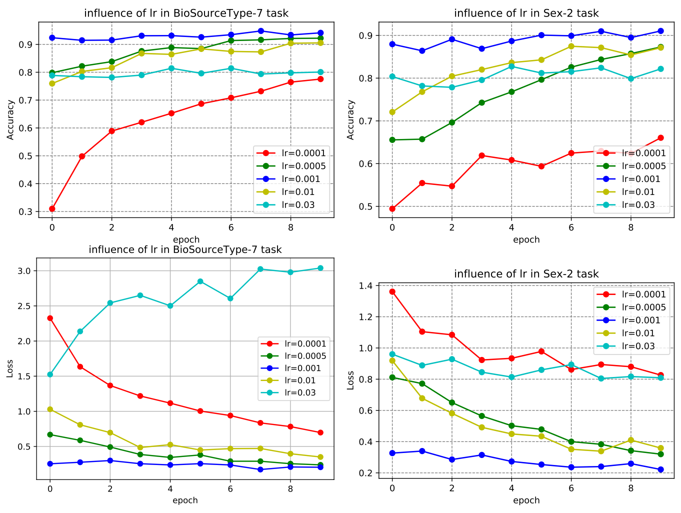

Fig. 19. a.accuracy in Bio task; b, accuracy in Sex task; c, loss in Bio task; d, loss in Sex task

level. As a extension to the original deep forest [17], we also add a LR model in each level to boost the performance of it.

We find that when applied on 4 different tasks, the final levels of deep forest varies. The results are shown in Tabel VI. We can find that Deep forest beat random forest in all the 4 tasks, which indicates that such *deep* variant of traditional works. We also noticed that the performance of deep forest is compatible to SVM, and we argue that this is because of the data's distribution.

E. Deep Learning based Method

1) nn on pca-processed data: At first we try deep neural network on pca-processed data. The dimension is 500, and the network is shown in Fig5). Besides, we use L2 regularization and dropout to avoid overfitting. The dropout rate is 0.50. In the end, we use batch normalization in each layer and it greatly improves the performance. The results and selected parameters are shown in Table(VII).

Besides, we visualize the training process of Sex and Bio tasks with the change of learning. It is shown in Fig(19), and it can be seen $lr = 0.001$ is the best result in four task.

2) nn on raw data: To improve the nn' performance, we try deep neural network on raw data. Because it is a nonlinear classifier, it may solve the relevant of raw data feature by itself. The dimension is 22283, and the network is shown in Fig5). The setting of L2 regularization, dropout and batch normalization is same as the above experiment. But without the proper initialization (e.g. random initialization), the performance is not better than nn above. On the contrary, the accuracy is only at around 70%, for it is easily influenced by local optimization. So we use greedy layer-wise pre training method to initialize the weights using 3 different neural network (which have the same structure as autoencoder) . Then the performance is improved a lot, besides the train process becoming faster. The best result and seleted parameters is shown in Table(VIII).

Compared with Table(VII), it can be seen that MaterialType task and BioSourceType task has been

Label Type	$F_1(\%)$	learning rate	$batchsize$
MaterialType	99.78	0.001	32
Sex	92.41	0.01	50
DiseaseState	94.37	0.001	32
BioSourceType	98.35	0.001	32

TABLE VIII
BEST RESULTS AND PARAMETER SETTINGS FOR NN MODEL ON RAW DATA.

improved, but the Sex task and DiseaseState task did not become higher. Besides, there are some differences among the optimal parameters.

ACKNOWLEDGMENT

This work is successfully produced thanks to the kind guidance from and discussion with Prof. Bo Yuan and TAs and the assistant from my partner Zhenghui Wang.

REFERENCES

[1] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[2] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[3] François Chollet et al. Keras, 2015.

[4] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[5] Nello Cristianini and John Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. 1999.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[8] Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. *Neurocomputing: algorithms, architectures and applications*, 68(41-50):71, 1990.

[9] Yi Liu and Yuan F Zheng. One-against-all multi-class svm classification using reliability measures. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 2, pages 849–854. IEEE, 2005.

[10] A. Rechtsteiner M. E. Wall and L. M. Rocha. Singular value decomposition and principal component analysis. *In A prac- tical approach to microarray data analysis*, pages 91–109, Springer, 2003.

[11] Gjorgji Madzarov, Dejan Gjorgjevikj, and Ivan Chorbev. A multi-class svm classifier utilizing binary decision tree. *Informatica*, 33(2), 2009.

[12] A. Krizhevsky I. Sutskever N. Srivastava, G. Hinton and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[14] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[15] Y. Bengio Y. LeCun and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[16] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[17] Zhi-Hua Zhou and Ji Feng. Deep forest: Towards an alternative to deep neural networks. *arXiv preprint arXiv:1702.08835*, 2017.