Cell Type and Disease State Inference

Yifeng Chen, *CSE, SJTU*

May 20, 2017

***Abstract*— Nowadays,machine learning techniques have been broadly used in various applications.Though ML techniques have proved its magic power in solving prediction problems, in fields like bioinformatics,we have to face an inevitable dilemma that the feature space's dimension is way much larger than the number of observations, thus bring up the contradiction between bias and variance.In this project, we propose two models,one with robust pca dimension reduction and simple linear model and the other with multilayer perceptrons.We evaluate the performance of these two models by training them to infer cell types and disease state from cell array.**

I. Preliminaries

A. Dataset Insight

E-TABM-185, a dataset provided by the European Bioinformatics Institute (EMBL-EBI).In our project,we exploit two relevant files of this dataset.

MicroArray contains the gene expression of 5896 cells.Each cell corresponds to a 22283-length vector,with each scalar representing the expression extent of this cell to different genes.

E-TABM-185 is a hgu133a integrated dataset integrated from ArrayExpress and GEO.For each cell in microarray,the dataset provides us with its detailed information,such as CellType,gender,BioSource,etc.

B. Labels Selection

Since some of the records in E-TABM-185 is vacant,it is necessary to select proper labels for out inference.To keep as many cells as we can,we select disease state since about 4k cells have corresponding records. Meanwhile,to further test the generalizability of our methods,we also choose Cell Type as label,which shrink numbers of observations to about 1k.

C. Dataset Division

After we select the label,we have to divide the dataset into two parts,one for train and the other for test.Before the division,a filter operation is applied to eliminate those types that occurs for less than 20 times. Finally,we get 1077 cells of 19 different cell types and 3053 cells of 43 different disease states.

Since the dataset itself is extremely unbalanced ,we can't randomly divide the dataset into two parts to do training and test. For each label type,we deliberately choose 20% for testing and leave the rest for training so that the test set's distribution is the same as the train set.

II. Traditonal Approach

As introduced before, we are facing with a so-called "tiny n,huge p" problem where feature space significantly exceed the number of observations.

Through traditional approach,we hope to use a simple linear classifier to do the inference.However, directly applying a simple model on such a high-dimension feature space is likely to suffer from the under-fitting problem,so we need to do dimension reduction in lieu of feeding the raw data into simple models.

A. PCA

Principle Component Analysis(PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.Mathematically,PCA can be considered as an constrained optimization problem:

$$\begin{gathered}\min ||M - MXX^T||_F^2 \\ \text{s.t. } X^TX = I_p\end{gathered}$$

where $M \in R^{n\times m}$ is the original Matrix,$X \in R^{m\times p}$ is its principal components.X can be calculated by solving SVD of M:

$$M = U\Sigma V^T$$

Σ contains singluar values, and we do a hard cut to shrink the dimension to p with 95% of the singular values get kept.Then we use $MV \in R^{n\times p}$ to approximate M.

B. Robust PCA

One drawback of classical PCA is that it is very sensitive to possible outliers and corrupted data.Robust PCA tackles this problem by regularizing PCA by a $\ell 1$ penalty.Mathematically,robust pca can be written as an optimization problem as

$$\begin{gathered}\min ||L||_* + \gamma||S||_1 \\ \text{s.t. } to\ L + S = M\end{gathered}$$

To solve this,we use Alternating direction method of multipliers (ADMM)[2] algorithm.To convert the problem to standard ADMM form,we rewrite it into the augmented Lagrangian form:

$$\begin{aligned}L_{\frac{1}{\tau}(L,S,y)} = {} & ||L||_* + \gamma||S||_1 + \\ & \frac{1}{\tau} < y, M - L - S > + \\ & \frac{1}{2\tau}||M - L - S||_F^2\end{aligned}$$

Then we apply the algorithm as follows,where $SVT_{\gamma}X = Udiag(\{(\sigma_i - \gamma)_+\})V^T$.

Algorithm 1 ADMM For Robust PCA

```
1: $S_0 \leftarrow \hat{S}, y_0 \leftarrow \hat{y}, k \leftarrow 1$
2: $\tau \leftarrow \hat{\tau} > 0$
3: **while** not converge **do**
4:     $L_k \leftarrow SVT_{\tau}(M - S_{k-1} + y_{k-1})$
5:     $S_k \leftarrow ET_{\gamma\tau}(M - L_k + y_{k-1})$
6:     $y_k \leftarrow y_{k-1} + \frac{1}{\tau}(M - L_k - S_K)$
7:     $k \leftarrow k + 1$
```

Use the Low Rank approxiamtion L we get,we apply PCA again on L and use its subspace to project the original matrix M on.

C. LIBLINEAR

As for the linear classifier,We use the LIBLINEAR[1] library provide by Machine Learning Group at National Taiwan University to do classification.LIBLINEAR supports two popular binary linear classifiers: LR and linear SVM. WE use the LR mode here.Given a set of instance-label pairs $(x_i, y_i), i = 1, ..., l, x_i \in R_n,\ y_i \in \{1, +1\}$, LIBLINEAR solve the following unconstrained optimization problem:

$$min_w \frac{1}{2} w^T w + C \sum_{i=1}^{l} log(1 + exp(-y_i w^T x_i))$$

For a multi-class problem,LIBLINEAR uses a one-versus-rest strategy.

III. MLP Approach

A multilayer perceptron (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs. An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training the network.[3][4].MLP is a modification of the standard linear perceptron and can distinguish data that is not linearly separable[5].

In our method,we propose a 4-layer MLP model,with 2k,1k,500 hidden units layerwise.Due to the constraint of our GPU,the original dimension(22283) will exhaust the GPU resources.So we use the dimension-reduced(1100) approximation in lieu of raw data here.

Since we'are already using the reduced-dimension,it increases the risk of overfitting greatly.In order to control the complexity,we use $\ell 2$ norm,$\ell 1$ norm and dropout techniques to constrain the model.

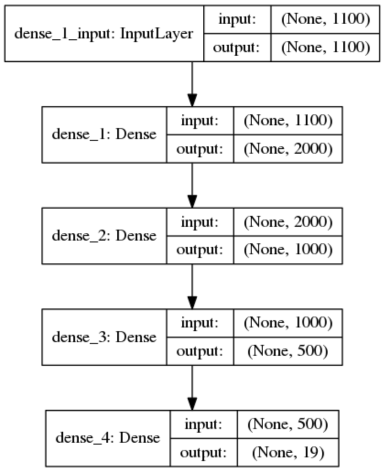


Fig. 1: Model for Cell Type Inference

A. Relu

We choose Relu as the activation function.Relu can be viewed as a gate that let positive activations pass and block out all activations.Our motivation to use Relu here is to avoid vanishing gradient problem which we find will occur when using sigmoid.One thing tricky when using Relu is the initialization of parameters.For example,when we are initializing parameters using normal distribution,we have to shift the distributions a little towards positive.Otherwise,some units may get killed initially and won't be trained again.

B. Pre-learning

The initialization of parameters has always been a non-trival problem for deep learning.A bad initialization may make the model trapped in some local minimum and not properly trained. An interesting approach to initialize the parameters is through an unsupervised learning process called Encoding & Decoding.In our project,we replace the output layer in our model with another layer to reconstruct the input.Then we tune the parameters to minimize the reconstruction error.By training the model to decompose and ensemble the input, we hope that it can learn to extract vital components and higher-level features from the input .The reconstruction error we used is the euclidean distance between input and output signal.

C. $\ell 2$ Norm

$$||M||_2 = \sqrt{\sum_{i,j} M_{i,j}^2}$$

$\ell 2$ Norm is applied in all layers except the output layer to control model complexity.$\ell 2$ Norm is computational efficient because it is a convex constraint.The l2-norm solution will tend to have relatively fewer large residuals (since large residuals incur a much larger penalty in l2-norm approximation).

D. $\ell 1$ Norm

$$||M||_1 = \sum_{i,j} |M_{i,j}|$$

The amplitude distribution of the optimal residual for the $\ell 1$-norm approximation problem will tend to have more zero and very small residuals , compared to the l2-norm approximation solution.

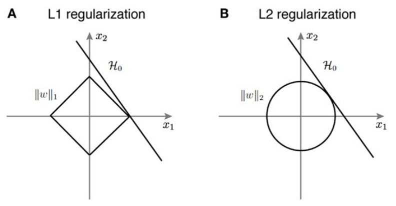


Fig. 2: $\ell 1$,$\ell 2$ illustration

Consider the 2D case in Fig.2,in the case of L2 regularization, in most cases, the hypothesis is tangential to the $||w||_2$(which is a circle),the point of intersection has both x1 and x2 components. On the other hand, in L1, due to the nature of $||w||_1$, the viable solutions are limited to the corners, which are on one axis only. Extend this to higher dimensions and we can see why L1 regularization leads to solutions to the optimization problem where many of the variables have value 0.

To gain this sparsity,we need far more computation than $\ell 2$ due to the non-convex nature of $\ell 1$,but we believe that it can enhance the generalizability of the model due to the intrinsic selective nature of gene expression.

E. Dropout

Dropout[6] is a technique that prevents overfitting and provides a way of approximately combining exponentially many different neural network architectures efficiently. During training stage,we randomly drop some of the connection during one forward pass and only train some part of the network.One can view this trained part as a new model so that during train process we are actually training hundreds of thousands of models together.During prediction stage,we keep all the connections,which can be viewed as an ensemble of the previous models trained.

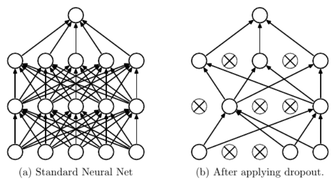


Fig. 3: Dropout

We apply dropout in all layers except output layer and try out different keep ratios:0.6,0.8,1.0(no dropout).

F. Adam

Using naive gradient-descent may cause severe oscillation due to the unbalance distribution,so we use Adam[7] to optimize our MLP.Adam is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. Adam is straightforward to implement,and is invariant to diagonal rescaling of the gradients,and is well suited for problems that are large in terms of data and/or parameters.

Algorithm 2 Adam Optimizer

```
1: Require: $\alpha$ Stepsize                        ▷ usually set to 1e-3
2: Require: $\beta_1, \beta_2 \in [0, 1)$           ▷ usually set to 0.9,0.999
3: Require: $f_\theta(x)$                            ▷ Object function
4: Require: $\theta_0$
5: $m_0 \leftarrow 0$
6: $v_0 \leftarrow 0$
7: $t \leftarrow 0$
8: while $\theta_t$ not converge do
9:     $t \leftarrow t + 1$
10:    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta 1) g_t$
11:    $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta 2) g_t^2$
12:    $\hat{m_t} \leftarrow m_t/(1 - \beta_1^t)$
13:    $\hat{v_t} \leftarrow v_t/(1 - \beta_2^t)$
14:    $\theta_t \leftarrow \theta_{t-1} - \alpha \hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)$   ▷ $\epsilon$ usually set to 1e-8.
```

IV. Experiments & Results

A. Traditoinal Approach

We try three different models in Traditional Approach:RAW LIBLINEAR, PCA LIBLINEAR, RPCA(robust pca) LIBLINEAR. The result is as follows(dim:input feature dimension):

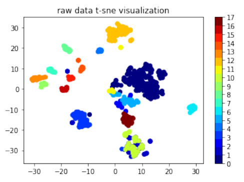
	Model	Dim	Test Accuracy
Cell Type	RAW	22283	**88.94**%
	PCA	1002	71.15%
	RPCA	**100**	86.05%
Disease State	RAW	22283	87.05%
	PCA	2322	**87.73**%
	RPCA	**100**	84.32%

Using Raw LIBILNEAR here can actually achieve a decent performance in both two tasks,indicating that features lie in a relatively low space where LibLinear can handle.Traditional PCA leads to great performance shrinking when classifying cell types.However,this method works when classifying disease Disease states.This instability may be explained by one of traditional pca's flaws:being extremely sensitive to outliers. An astonishing fact is that we can use RPCA to enormously reduce the feature space into 100 dimensions with a little performance drop(2%-3%).Using $\ell 1$ norm enables our model the magic ability of feature selecting.

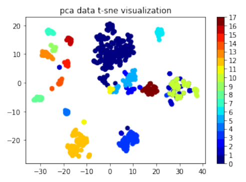
B. *MLP Approach*

	Dropout	Norm	Accuracy
Cell Type	0.6	L2	91.23%
	0.6	L1+L2	**93.26%**
	0.8	L2	89.15%
	0.8	L1+L2	90.05%
	1.0	L2	87.89%
	1.0	L1+L2	88.57%
Disease State	0.6	L2	91.24%
	0.6	L1+L2	92.05%
	0.8	L2	91.09%
	0.8	L1+L2	**92.09%**
	1.0	L2	88.24%
	1.0	L1+L2	88.05%

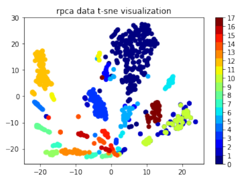
Clearly the composition of dropout and $\ell 1$ outperforms others in both tasks.A MLP of 4 layers can learn the decomposition and composition of features automatically,thus leading to a higher accuracy(4%-5%) in test data set.Since we'are using the traditional PCA approximation as input here.This means MLP learns much better(20%) than linear classifier when classifying cell type,indicating a stronger generalizability.To prove this,we use t-sne to visualize the representation generated by RAW, PCA, RPCA and MLP.PCA and RPCA both fail to separate class 10(green points) and class 2(blue points),while MLP does much better than them.



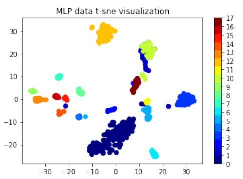
(a) raw data



(b) pca data



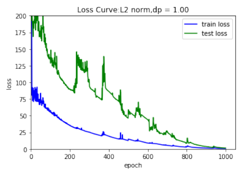
(c) rpca data



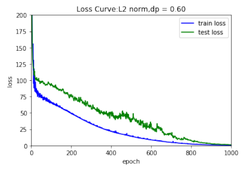
(d) MLP data

Fig. 4: t-sne visualization of gene representations

Fig.5 shows the power of dropout during train time.In Fig.5(a), we are suffering from a severe oscillation in test set and a large performance gap without dropout.In Fig.5(b), we can barely see any severe oscillation and the performance gap between train set and test set is much smaller.This shows that dropout can work as a perfect regularizer and improve the problem known as "variance". Fig.6 shows the same result by accuracy performance.The accuracy of the model with dropout is not only higher but also steadier than the one without dropout.

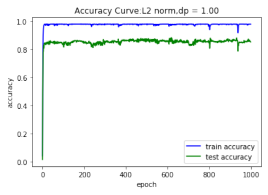


(a) dropout=1

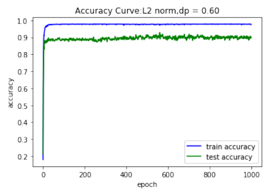


(b) dropout=0.6

Fig. 5: Dropout: Loss Comparision



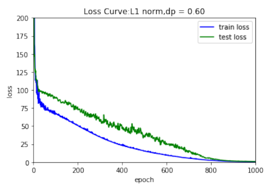
(a) dropout=1



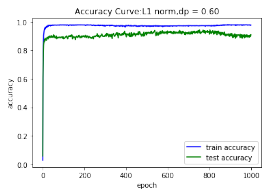
(b) dropout=0.6

Fig. 6: Dropout:Accuracy Comparision

Fig.7 is the loss and accuracy curve with L1+L2 norm and dropout=0.6.Compared to Fif.5(b),Fig.6(b),it is slightly better than using L2 alone.However,adding L1 is not as effective as dropout.This may because that MLP has the power of learning features naturally,so adding L1 in the last layer to explictly achieve feature selection may not be necessary here.



(a) loss



(b) accuracy

Fig. 7: L1+L2,dropout=0.6

V. Conclusion

In this project,we propose two approachs to do inference of cell types and disease states on cell arrays,which is a "tiny n a large p problem".In the traditional approach,we use RPCA to project the data into a lower dimension subspace and a linear classifier to solve it.In the MLP approach,we construct a 4-layer MLP with dropout and $\ell 1 + \ell 2$ normalization,which turns out to outperform all other models.A MLP is able to learn the feature automatically and hierarchically which enables it a strong generalizability.

Acknowledgment

Genuine thanks to my teammate Ruiji Wang for his consistent assistance and contribution in this project.

Special thanks to Prof.Yuan and T.A.for their patient guide in this project and devotion in course CS410.

References

[1] Rong-En Fan,Kai-Wei Chang,Cho-Jui Hsieh,Xiang-Rui Wang,Chih-Jen Lin *LIBLINEAR: A Library for Large Linear Classification* Journal of Machine Learning Research 9 (2008) 1871-1874

[2] Stephen Boyd, Neal Parikh, Eric Chu,Borja Peleato,Jonathan Eckstein *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers* Foundations and Trends in Machine Learning Vol. 3, No. 1 (2010) 1122

[3] Rosenblatt, Frank. x. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Spartan Books, Washington DC, 1961.

[4] Rumelhart, David E., Geoffrey E. Hinton, and R. J. Williams. *Learning Internal Representations by Error Propagation* Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundations. MIT Press, 1986.

[5] Cybenko, G. Approximation by superpositions of a sigmoidal function Mathematics of Control, Signals, and Systems, 2(4), 303-314.

[6] Nitish Srivastava,Geoffrey Hinton,Alex Krizhevsky ,Ilya Sutskever,Ruslan Salakhutdinov. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting* Journal of Machine Learning Research 15 (2014) 1929-1958

[7] Diederik P. Kingma, Jimmy Ba. *Dropout: Adam: A Method for Stochastic Optimization* conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015