CSE514 Project- Feature Selection

-Weiqing Wang, Ziyang Liu, Fan Zhang

**1. Overview**

Feature selection is a process of selecting a subset of relevant features for use in model construction. Feature selection techniques are used for three reasons:

* simplification of models to make them easier to interpret by us.
* shorter training times.
* enhanced generalization by the reduction of variance.

In this project, three feature selection methods were applied: SVD/PCA, PCA & ICA and Autoencoder on the Alzheimer disease genes dataset to find out whether data selection works in the way we anticipated and the different between three methods.

The preliminary results are generated as compare groups to see what feature selection can bring to the table. The classification/cluster methods chosen are decision tree, random forest, SVM and k-means. After applying all three methods to the whole dataset, the results are as follows:

|  |  |
| --- | --- |
| Decision Tree | ~ 61% |
| Random Forest | ~ 67% |
| SVM | ~ 65% |
| k- means | ~ 59% |

**2. Feature selection**

*2.1 SVD/PCA*

*2.1.1 Intuition and methods*

The algorithm used in this project for SVD first compute the leverage score for each dimension and then selects the first 20 features with the largest weights, some description of the algorithm is as follows: first, using MATLAB’s svd method to compute the principal components. The key idea here is that we want to get the dimensions having the dominant eigenvectors, e.g. representing 95% percent of the data. Then by computing the leverage score and sorting them, we can get the ranking of each features. Finally, the most significant features can be selected.

Similar to SVD, using PCA we can get the eigenvectors, by the same method, we can get the dominant eigenvectors, computing the leverage score and get the rank of them, we can choose the most representative features we want.

*2.1.2 Results*

First, by using three classification methods mentioned above and k-means, we can get the accuracy results below:





The right ones are k-means results.

*2.1.3 Analyze*

We can see that, by feature selection, methods like SVM and random forest get a performance increase and significantly runtime deduction, but decision tree method doesn’t perform well with the selected dataset. The reason is that since the dataset is not linear separable, random forest and SVM can handle it well. But decision tree depend much more on the variety of the dataset, so only 20 significant features cannot get the satisfied result.

*2.2 Feature selection base on ICA*

*2.2.1 Brief introduction to ICA*
 ICA aims to find an entirely new coordinate system of statistically independent directions, the basic ICA model can be expressed as x = As, A is the mixing matrix and S is the source matrix.
For example:
x1 = a11s1 + a12s2+a13s3
x2 = a21s1 + a22s2+a23s3
s1 = w11x1 + w12x2
s2 = w21x1 + w22x2
s3 = w31x1 + w32x2
We use the Fast\_ICA mentioned in the paper, it takes 146.695615s to calculate and extract ICA feature and use the k-means as the classifier, the result is listed below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Number of ICA | K-means | Decision Tree | Random Forest | SVM |
| 2 | 53% | 53.47% | 53.57% | 54.39% |
| 20 | 56.59% | 50.1% | 54.57% | 54.1% |
| 200 | 54.12% | 52.74% | 51.0989% | 54.12% |

 Obviously, the Independent component vectors which extracted by our algorithm don’t have the information we want to have.

*2.2.2 Alternative method*

 So we turn to use the another method which was proposed by H.K Ekenel, using ICA1 architecture:
 Prior to ICA, PCA is performed on the AD set, then we keep M largest eigenvectors in the matrix V, perform ICA analysis on V’, then we can get the ICA representation coefficients of the AD set, {a1,a2….ak}:
For a given xt, rt = xtV’, at = rtA, overall the projected Matrix can be expressed as: V’A.
We did this combination in two ways.

1) We keep top-k eigenvectors in the V’, then set the number of IC equal to the number of eigenvectors, then use k-means to do the clustering:

 

The accuracy is around 0.60, best at k = 67 with 61.87% accuracy, the other see below:
SVM 68.95% Random Forest 68.4%

2) We fixed the number of eigenvectors (200), then set the number of IC from 2 – 200:



 In this case, we can see with the number of IC increasing, the accuracy increasing, which suggested that using ICA to reduce the dimension in this dataset is not a good idea.
The best performance is listed below:
k-means 62.09% Random forest 69.7802% SVM 68.8352%

 Compare the above table with the results we have in the PCA and origin (total feature), the PCA-ICA slightly increasing the behavior of this 3 classifier, but I still don’t think the ICA can be applied on this data set, because the idea that decompose the gene matrix to the linear combination of several source is required the knowledge of the biology, and if we don’t know the number of source, the results would be meaningless, and 360+ number of samples is definitely not support us to find the best way to decode those gene with pure mathematics.

*2.2.3 Data visualization*
Despite my doubts, I think there may have something information we can get from our results,
If I use 2 IC, then I can draw the data points on the graph:



Although most of points tangle with each other’s, we still can see some outliers, so in this case, I think the k-nn classifier would be fit for this data, so we tried:
Using the 200 vector – ICA, using 7-nn accuracy: 76.19% compare to the origin data 64%, which is a great improvement, however, because I used this model after I analyzing the data, so I would not trust the performance I get, since the Eout is no longer bounded by hoeffding.

2.2.4 *Another interpretation of the ICA and future works*
 Basically, the project matrix can be seen as the efficiencies of the origin feature, from the IC vectors, we can know how much the origin feature contribute to the Independent vectors.
 So, with this idea in mind, we analyzed the 2-IC, after normalized and chose the coefficient based on the absolute value > 4(with is the 0.0025% of the origin features), we selected two groups of genes, then tried to use DAVID to further study those genes.

*2.3 Stacked Denoise Autoencoder (SDAE)*

*2.3.1 Tasks & Background:*

Consider the form of data we have and the paper included in the project, we decided to use the Stacked Denoise Autoencoder (SDAE) to reduce the dimension of our data while extracting useful infomation.

The procedure of training of an SDA can typically be divided into two phases:

- Pertaining

Unsupervised layer-wise training phase. Train one layer (autoencoder) at a time while fix parameters of other layers by minimizing the reconstruction error.

- Fine-tuning

Supervised training phase. Apply an input, get the value loss function on the output layer, then reflect back the loss as weight updates to each layers, from the output to the input.

*2.3.2 Effect of Parameters during pertaining*

During the training, we may add noise to the input data in order to increase the robustness of the produced SDAE.

We plan to train an SDAE, remove its classification layer on the output, then use the remain part to reduce the feature of our data while pertaining useful information.

The performance of a stacked denoise autoencoder training algorithm may related to at least some or all of these parameters:

1. The configuration of the network

 - Number of hidden layers

 - Number of variables ( dimension ) of each layers

1. The stopping conditions during pertaining
2. The stopping conditions when training single layer
3. The noise added during training (could be different on each layer)
4. The step size of the SDG method during training

Consider the amount of parameters, we start from training a single autoencoder, using the Matlab built-in trainAutoencoder, want to see how the number of iterations and the number of features reflects its information extracting abilities. We apply k-means algorithm on the data went through pertained autoencoder and treat its accuracy as the quality of the data we got. (both on encoded data from hidden layer and decoded data from the output layer). We got these two plots

 

In the left figure, we can see that with the increase of number of iterations, the quality of data first increase, then begin to decrease. We think the reason is that the AE preprocess here works similar as PCA. With the effect of transfer and sigmoid function together, some the information was dropped. But the most important information is not likely to be dropped in this case. So the quality of data improved at first. But after some point, we loss too much information and the quality started to decrease.

In the right figure, the input dataset we used only 50 features. From the figure we can see that the quality of encoded data is at max when it “fixes” the input. It might be good to let input layer to have the same dimension as the data. One of the disadvantage of this is its running time. Our dataset have 8560 features so doing this may have big overhead.

After this, we use Matlab to stack same autoencoders to see how it performs regarding the number of layers.



We can see that with too much layer our quality will be reduced. Actually in later layers, the training will stop very fast due to convergence. This indicates that there is no more useful information.

*2.3.3 Train SDAE*

Finally we modify Theano Python Package to train SDAE and use it to preprocess our data. Different configures was applied. (80% for training and validation, 20% for test)

Hidden Layer Structure tried: RF SVM - rbf LR-Train LR-Test

raw data 65.3846 % 63.4615 %

[250,150,50 ] 64.001% 63.7673% 62.5% 64.1%

[250,150,50 ] each+ 0.1 noise 62.3626% 63.1868% 59.6% 57.5%

[250,150,70,150,250] 63.1868 % 51.6484 % 64.3 % 55.8%

[250,150,70,150,250] each+ 0.1 noise 64.2857 % 51.6484 % 65.1% 60.4%

We can see that SDAE is highly flexible and its configurations will greatly effect its performance.

On the above examples, we can see that with same configuration [250,150,50], same dataset, adding 0.1 noise reduced all the performance. While in configuration [250,150,70,150,250], its performance doesn’t reduced, and it also generalize better.

We think this is because by adding big layers on the back ,the power of Find-tuning will be boosted, since the output layer has more power to distribute the weight update. But introducing these layers may capture useless info and make a bad bound, which make SVM bad. We can see that SDAE have a lot of potential, but choosing the right settings is not easy.

**3. Summary**

*3.1 Solution quality*

The best results for each selection methods using different classifiers are shown below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| classifier | decision tree | random forest | SVM | k-means |
| PCA | ~60% | ~65% | ~68% | ~62% |
| SVD | ~59% | ~68% | ~68 | ~60% |
| PCA-ICA | ~65% | ~68% | ~69% | ~62% |
| auto-encoder | ~60% | ~64% | ~64% | ~63% |

Besides the PCA-ICA works very well using 7-NN classifier ~76%

We can see from the results that PCA-ICA is a quite competitive method, giving that it shows rather stable and good performance overall. As for the performance of different classifiers, SVM and random forest have great performance for all four selected datasets. We think that’s due to the property of the AD set, which is non-linearly separated. Autoencoder currently do not have a compelling results. But we can see that it has very big potential. Its parameter tuning is very tricky. Its performance and properties can be greatly effected by its structure and other parameters. So it has great flexibility.

*3.2 Modeling complexity*

SVD has the highest speed and PCA is slower. SVD runs in the time range of several seconds and PCA in several minutes. ICA alone is in the thied place, it runs in about 120 seconds, after combining it with PCA and reduce its dimension, it became much faster. Autoencoder is relatively expensive if some layer have large dimension. It is better to run it on GPU. Most of its time were spent on pertaining.

Regarding the difficulty of implementing all the methods we use, manipulating SVD, PCA and ICA relying on more on your math ability, since we can use MATLAB and do simple operations. But Autoencoder requires much more configuration, coding and debugging, which brings quite amount of load.

*3.3 Lessons learned and possible future work*

We concluded that different learning algorithm is sensitive to different attributes of the dataset. For example, SVM is very sensitive to poor boundary. By applying different algorithms ,we can get implication about how our dataset was changed.

In the future, using the top features we selected, by tools like DAVID, we can get what each genes’ name, but due to the lack of biomedical knowledge, we would not know what these genes represent, but we suspect they would be related to those key factors that may cause Alzheimer disease

*3.4 Work distribution*

SVD/PCA - Fan Zhang; PCA & ICA- Weiqing Wang; Autoencoder Ziyang Liu.

**4 Reference**

1. Xing, Chen, Li Ma, and Xiaoquan Yang. "Stacked Denoise Autoencoder Based Feature Extraction and Classification for Hyperspectral Images." Journal of Sensors 2016 (2016): 1-10. Web.

2. Masci, Joathan, Ueli Meier, Dan CireÅan, and JÃrgen Schmidhuber. "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction." Lecture Notes in Computer Science Artificial Neural Networks and Machine Learning â ICANN 2011 (2011): 52-59. Web.

3. Prasad, Mithun, Arcot Sowmya, and Inge Koch. "Efficient Feature Selection Based on Independent Component Analysis." Proceedings of the 2004 Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004. (n.d.): n. pag. Web.

4. Chakroborty, Sandipan, and Goutam Saha. "Feature Selection Using Singular Value Decomposition and QR Factorization with Column Pivoting for Text-independent Speaker Identification." Speech Communication 52.9 (2010): 693-709. Web.