Machine Learning Homework 3*

Hongzhou Liu
517030910214
deanlhz@sjtu.edu.cn

1 SVM vs. Neural Networks

In this section, I did experiments on the SVM and MLP using following two datasets:

Table 1: Datasets

Dataset	Classes	Size	Features
breast-cancer	2	683	10
dna	3	3186	180

I tried both binary and multiclass classification tasks on the two classifiers. The size and features of the two datasets is different. Thus, we can compare the performance of SVM and MLP in multiple perspectives.

1.1 SVM

1.1.1 Experiment on data preprocessing

In this part, I try to preprocess the datasets and compare the performance of SVM between preprocessed datasets and un-preprocessed datasets. The result is shown in Table 3 and the setting of other hyper-parameters in Table 2. The preprocessing method is simple Z-score normalization here. The Z-score normalization converts the raw data into the standard score by

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

where μ is the mean of dataset and σ is the standard deviation of the dataset. As the result shows, the effect of normalization is larger when the dataset has less samples. It's probably because that when samples are inefficient, the mean and the deviation will be affected largely by some outliers while in big datasets, it will not occur.

Table 2: Hyper-parameters

Dataset	training size	C	kernel	dimension
breast-cancer	0.6	1.0	rbf	10
dna	0.6	1.0	rbf	180

1.1.2 Experiment on sample size of training set

In this part, I split the datasets into training and test sets with different ratios. As the results of non-preprocessed data seem better than preprocessed data. I choose not to normalize the data, other hyper-parameters is shown in Table 4. The result in Table 5 shows that the train-test ratio affects the performance a lot. The best performance of both datasets occur when the ratio is set to 0.8. The test

*GitHub repo: https://github.com/DeanAlkene/CS420-MachineLearning/tree/master/A3

Table 3: Experiment on data preprocessing

scale	breast-cancer accuracy	dna accuracy
True	95.26%	95.37%
False	**97.81%**	**95.84%**

accuracy even hits 99% in breast-cancer dataset. It shows that the problem of overfitting occurs here because the breast-cancer dataset consists of little samples, the classifier will easily overfits. The results also told us to train a classifier with abundant training data but not too much, because it will hurts the generalization ability of the classifier.

Table 4: Hyper-parameters

Dataset	scale	C	kernel	dimension
breast-cancer	False	1.0	rbf	10
dna	False	1.0	rbf	180

Table 5: Experiment on sample size of training set

training size	breast-cancer accuracy	dna accuracy
0.9	95.65%	93.41%
0.8	**99.27%**	**97.65%**
0.7	97.07%	95.86%
0.6	96.35%	95.29%
0.5	97.66%	96.30%

1.1.3 Experiment on penalty factor C

As we all know, the primal form of SVM is

$$\begin{array}{c} \min_{\boldsymbol{w},b,\xi_i} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{s.t. } y_i\left(\boldsymbol{w}\cdot\boldsymbol{x}_i + b\right) \geq 1 - \xi_i \\ \xi_i \geq 0, i = 1, 2, \ldots, N \end{array} \tag{2}$$

As we can see, there's a penalty factor C before the slack variables. The larger the C is, the heavier the penalty adds to the misclassified data points. In this part, I did experiment on different C's, the other hyper-parameters is shown in Table 6 and the results is in Table 7. Actually, when using rbf kernel, we need to set C a little bit larger while when using linear kernel, small C should be set. As we can see, when C is small, the performance of SVM on both datasets is quite bad. However, the best C on breast-cancer dataset is smaller than it on dna dataset. It's probably due to the number of classes and the size of the dataset increased the risk of misclassification on dna dataset, thus a larger penalty factor is needed. Also, a large C will produce a small margin hyperplane allowing less misclassifications.

1.1.4 Experiment on kernel of SVM

In this part, I compared the performance of SVM with different kernels on the datasets. The kernel I used are:

- Linear kernel: $K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y}$
- Polynomial kernel: $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + c)^d$
- RBF kernel: $K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right)$
- Sigmoid kernel: $K(\mathbf{x}, \mathbf{y}) = \tanh(\alpha\mathbf{x}^T\mathbf{y} + c)$

The results is shown in Table 9 and other hyper-parameters in 8. As we can see, on breast-cancer dataset, polynomial kernel works best and linear is the second. On dna dataset, RBF kernel works

Table 6: Hyper-parameters

Dataset	scale	training size	kernel	dimension
breast-cancer	False	0.6	rbf	10
dna	False	0.6	rbf	180

Table 7: Experiment on penalty factor

C	breast-cancer accuracy	dna accuracy
0.001	64.23%	52.70%
0.005	68.98%	52.70%
0.01	95.26%	49.65%
0.05	**97.08%**	52.55%
0.1	96.72%	68.31%
0.5	96.72%	95.45%
1.0	96.35%	**96.63%**
5.0	96.72%	95.76%
10.0	96.72%	95.69%

best. It reveals that the decision bound in breast-cancer is more likely a linear or polynomial one, while the dna dataset may be linear separable after mapping it into infinite dimension space (by RBF kernel). It reminds us that the decision bound varies a lot among different datasets.

1.1.5 Experiment on dimension

In this part, I did experiment on the dimension of features of both datasets. I utilized PCA to reduce dimension of feature vectors to get multiple dimension settings. Other hyper-parameters setting is in Table 10 and the results of SVM is in Table 11 and Table 12. As seen, when not reducing dimensions the performance is the best. For sure, if we reduce the dimension, some information will be lost and thus hurt the performance. However, on breast-cancer dataset, when the dimension is 5, it can achieve accuracy of 97.45%, only 0.36% less than the best performance. Meanwhile on dna dataset, there's a "valley" between dimension of 20 and dimension of 100. It shows that, there's still redundant information among dimensions. Sometimes, when the dataset is large, dimension reduction is a way to avoid dimension explosion and save time while not lossing much accuracy.

1.2 MLP

1.2.1 Experiment on data preprocessing

In this part, I will test the effect of Z-score normalization on MLP performance. The hyper-parameter setting is shown in Table 13 and the performance of MLP on the two dataset is in Table 14. The result is the same as it in the SVM part, which is, the Z-score normalization will not help to improve the performance of the classifier on those two datasets.

1.2.2 Experiment on sample size of training set

Again in this part, I will discuss the effect of train-test ratio on the performance of MLP on the two datasets. You can find hyper-parameter settings in Table 15 and testing accuracy in Table **??**. The situation is similar to the part of experiment of SVM. The best accuracy occurs when the ratio is about 0.8 or 0.9. But we can also get a satisfied result by setting the ratio as 0.5.

Table 8: Hyper-parameters

Dataset	scale	training size	C	dimension
breast-cancer	False	0.6	1.0	10
dna	False	0.6	1.0	180

Table 9: Experiment on kernel of SVM

kernel	breast-cancer accuracy	dna accuracy
linear	96.35%	92.39%
poly	**96.72%**	94.35%
rbf	95.62%	**95.45%**
sigmoid	92.70%	93.41%

Table 10: Hyper-parameters

Dataset	scale	training size	C	kernel
breast-cancer	False	0.6	1.0	rbf
dna	False	0.6	1.0	rbf

Table 11: Experiment on dimension

dimension	breast-cancer accuracy
2	97.08%
3	97.45%
5	97.45%
8	96.72%
10	**97.81%**

Table 12: Experiment on dimension

dimension	dna accuracy
2	76.39%
5	88.94%
10	91.37%
20	93.18%
50	92.39%
100	93.73%
150	95.69%
180	**96.39%**

Table 13: Hyper-parameters

Dataset	training size	hidden layers	activation	α	dimension
breast-cancer	0.6	(100)	relu	0.0001	10
dna	0.6	(100)	relu	0.0001	180

Table 14: Experiment on data preprocessing

scale	breast-cancer accuracy	dna accuracy
True	97.45%	93.80%
False	**97.81%**	**94.20%**

Table 15: Hyper-parameters

Dataset	scale	hidden layers	activation	α	dimension
breast-cancer	False	(100)	relu	0.0001	10
dna	False	(100)	relu	0.0001	180

Table 16: Experiment on sample size of training set

training size	breast-cancer accuracy	dna accuracy
0.9	97.10%	**95.30%**
0.8	**97.81%**	94.83%
0.7	96.59%	94.56%
0.6	96.72%	94.67%
0.5	97.66%	94.98%

1.2.3 Experiment on network structure

In this part, I will discuss the effect of network structure on the performance of MLP. By setting the number of layers and the number of neurons in each layer, we can get different network structures. They can be either wide or narrow, or shallow or deep. We first fix the hyper-parameters in Table 17 and modify the network structure. The test accuracy is shown in 18. As we can see, for breast-cancer dataset which is small and with short feature vectors, a narrow MLP with some depth is suitable. A wider network may not help except for adding depth. For dna dataset, which is larger and with more features, a wider network helps. Thus, we know that, the size of each layer in an MLP should match the number of features of the data. We should not add too much layers when the dataset is simple, which may lead to overfitting problem and will consume more time to train.

Table 17: Hyper-parameters

Dataset	scale	training size	activation	α	dimension
breast-cancer	False	0.6	relu	0.0001	10
dna	False	0.6	relu	0.0001	180

Table 18: Experiment on network structure

hidden layers	breast-cancer accuracy	dna accuracy
(10)	97.08%	93.49%
(100)	97.08%	94.43%
(10, 10)	**97.81%**	93.65%
(100, 100)	95.26%	94.59%
(200, 200)	95.62%	**94.90%**
(100, 200, 100)	97.45%	94.20%

1.2.4 Experiment on activation function

In this section, I did experiment about activation functions of MLP on the two datasets. We all know that activation function is an important component in Neural Networks. I did experiment with the following activation functions:

- Identity: $f(x) = x$
- Logistic: $f(x) = \dfrac{1}{1+e^{-x}}$
- Tanh: $f(x) = \tanh x$
- ReLU: $f(x) = \max(0, x)$

The test accuracy is shown in Table 20 based on the setting of hyper-parameters in Table 19. As we can see, on breast-cancer dataset, identity function can lead to a good performance while on dna dataset, we need to choose tanh as activation function to achieve a better performance. It is because, breast-cancer dataset is a simple dataset, even not introducing much complexity during training, MLP will get a good performance. We should also notice that ReLU also worked well in both situations.

Table 19: Hyper-parameters

Dataset	scale	training size	hidden layers	α	dimension
breast-cancer	False	0.6	(100)	0.0001	10
dna	False	0.6	(100)	0.0001	180

1.2.5 Experiment on regularization factor α

Regularization is a method to prevent model from overfitting. While using regularization technique, one should add a regularization term after the original loss function. There is a factor to control the

Table 20: Experiment on activation function

activation	breast-cancer accuracy	dna accuracy
identity	**97.45%**	91.45%
logistic	96.35%	93.96%
tanh	94.16%	**94.59%**
relu	97.08%	94.20%

scale of regularization. In this part, I tested two different factors in MLP on the two datasets. The results is shown in Table 22 and hyper-parameters is shown in Table 21. The result differs between two datasets. While using small dataset, a larger factor leads to a better performance while a smaller factor leads to a better performance when the dataset is larger. The effect of the regularization factor is hard to predict, we need to carefully adjust it to get a better performance.

Table 21: Hyper-parameters

Dataset	scale	training size	hidden layers	activation	dimension
breast-cancer	False	0.6	(100)	relu	10
dna	False	0.6	(100)	relu	180

Table 22: Experiment on regularization factor

α	breast-cancer accuracy	dna accuracy
0.001	**96.35%**	93.41%
0.0001	95.99%	**94.04%**

1.2.6 Experiment on dimension

Finally, we will try different dimensions and observe the performance of those datasets on MLP. As usual, we utilize PCA to get different dimensions. The other hyper-parameters is shown in Table 23. The results shown in Table 24 and Table 25 is quite different from what we get in SVM. The best performance on breast-cancer dataset occurs when the dimension is 3 and 8, which means the MLP can extract information from reduced data more efficiently. It also reveals that, there's quite much redundant in the dataset. For the dna dataset, we will get the best performance when the dimension is 150, the reason is the same as before.

1.3 SVM on Big Datasets

In this part, I will try SVM on big dataset. I chose CIFAR-10 as the dataset. The CIFAR-10 1 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. Thus, the feature vector is of 3072 dimensions. There are 50000 training images and 10000 test images.

I did 5 experiments, the experiment setting is shown in Table 26. As you can see, I also tried Z-score normalization and PCA dimension reduction. The result of those experiments in shown in Table 27. As you can see, training SVM on a big dataset is really time consuming. Fortunately, we can utilize PCA to reduce dimensions. When we apply PCA on the original dataset to reduce its dimension to 50 then run SVM, the total training time (including PCA) is about 50 times less than the SVM on the original dataset. However, it will, for sure, decreases the accuracy. I also compared our SVM performance with the deep learning algorithm benchmarks in `https://paperswithcode.com/sota/image-classification-on-cifar-10`. The comparison is shown in Table 28. Compared with those deep learning methods, the performance of SVM is quite bad. Thus, the strengths and weaknesses of SVM on big datasets can be conclude as:

Strength

- The optimization problem of SVM is convex, thus we can always find global optima, while local optima problem will occur when MLP is used.

Table 23: Hyper-parameters

Dataset	scale	training size	hidden layers	activation	α
breast-cancer	False	0.6	(100)	relu	0.0001
dna	False	0.6	(100)	relu	0.0001

Table 24: Experiment on dimension

dimension	breast-cancer accuracy
2	96.35%
3	**98.18%**
5	96.72%
8	98.18%
10	97.44%

Table 25: Experiment on dimension

dimension	dna accuracy
2	77.10%
5	89.02%
10	90.27%
20	89.18%
50	91.76%
100	92.31%
150	**96.31%**
180	94.51%

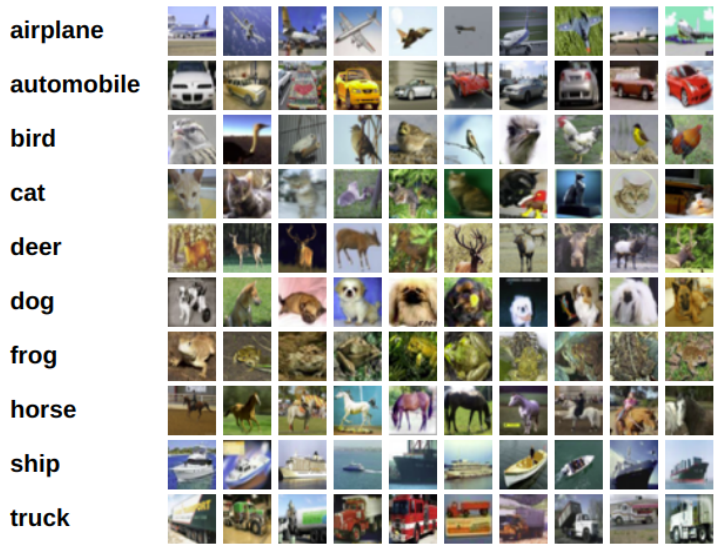

Figure 1: CIFAR-10

Table 26: Hyper-parameters

No.	Dataset	scale	C	kernel	dimension
0	CIFAR-10	False	5.0	rbf	3072
1		True			3072
2		False			50
3		False			500
4		False			1500

Table 27: SVM on CIFAR-10

No.	accuracy	time
0	57.21%	17106.87s
1	**57.29%**	19272.48s
2	53.93%	**452.99s**
3	54.78%	9371.59s
4	44.46%	30315.99s

Table 28: SVM on CIFAR-10

method	accuracy
EfficientNet-B7	98.90%
DenseNet	96.54%
SimpleNetv1	95.51%
ResNet	93.57%
SVM	**57.29%**

- The optimization objective can be solved by classical algorithms and the iterations needed to converge is less than MLP, though there's big time consumption for each iteration.
- The kernel trick is a good and easy method to achieve non-linearity.

Weakness

- The noisy data in the dataset can affect the hyperplane easily, especially when utilizing kernels. Some noisy data points will be considered as support vectors and thus lead to overfitting problem.
- The optimization algorithm of objective function of SVM is a quadratic programming algorithm. The time complexity is $\mathcal{O}(mn^2)$ when the size of the dataset is m and the number of features is n. Thus, SVM is too time consuming on big datasets.
- The SVM is originally a binary classifier. However, the datasets nowadays are always multi-labeled. The soultion to convert SVM into a multi-class classifier is combining multiple SVMs by one-vs-one or one-vs-rest methods. The performance of multi-class SVM is worse than binary SVM in this way.

2 Causal discovery algorithms

2.1 LiNGAM

In this part, I chose LiNGAM as the causal discovery algorithm. LiNGAM was one of the first of the algorithms that assumed linearity among the variables and non-Gaussianity of error term, and still one of the best for smaller models. The idea is to use the Independent Components Analysis (ICA) algorithm to check all permutations of the variables to find one that is a causal order—that is, one in which earlier variables can cause later variables but not vice-versa. The method is clever. First, since we assume the model is a directed acyclic graph (DAG), there must be some permutation of the variables for which the main diagonal of the inverse of the weight matrix contains no zeros. This gives us a permuted estimate of the weight matrix. Then we look for a permutation of this weight matrix that is lower triangular. There must be one, since the model is assumed to be a DAG. But a lower triangular weight matrix just gives a causal order, so we're done.

2.1.1 Model

The Linear, Non-Gaussian, Acyclic Model (LiNGAM) is a model with the following three properties.

1. The observed variables $x_i, i \in \{1, \cdots, m\}$ can be arranged in a causal order, such that no later variable causes any earlier variable. We denote such a causal order by $k(i)$. That is, the generating process is recursive, meaning it can be represented graphically by a directed acyclic graph (DAG).
2. The value assigned to each variable x_i is a linear function of the values already assigned to the earlier variables, plus a "disturbance" (noise) term e_i, and plus an optional constant term c_i, that is
$$x_i = \sum_{k(j)<k(i)} b_{ij} x_j + e_i + c_i \tag{3}$$
3. The disturbances e_i are all continuous-valued random variables with non-Gaussian distributions of non-zero variances, and the e_i are independent of each other, that is, $p(e_1, \cdots, e_m) = \prod_i p_i(e_i)$.

2.1.2 Algorithm

The causal discovery algorithm for LiNGAM is shown in 1

Algorithm 1: LiNGAM discovery algorithm

1 Given an $m \times n$ data matrix $\mathbf{X}$ ($m \ll n$), where each column contains one sample vector $\mathbf{x}$, first subtract the mean from each row of $\mathbf{X}$, then apply an ICA algorithm to obtain a decomposition $\mathbf{X} = \mathbf{AS}$ where $\mathbf{S}$ has the same size as $\mathbf{X}$ and contains in its rows the independent components. From here on, we will exclusively work with $\mathbf{W} = \mathbf{A}^{-1}$

2 Find the one and only permutation of rows of $\mathbf{W}$ which yields a matrix $\widetilde{\mathbf{W}}$ without any zeros on the main diagonal. In practice, small estimation errors will cause all elements of $\mathbf{W}$ to be non-zero, and hence the permutation is sought which minimizes $\sum_i 1/|\widetilde{\mathbf{W}}_{ii}|$.

3 Divide each row of $\widetilde{\mathbf{W}}$ by its corresponding diagonal element, to yield a new matrix $\widetilde{\mathbf{W}}'$ with all ones on the diagonal.

4 Compute an estimate $\widehat{\mathbf{B}}$ of $\mathbf{B}$ using $\widehat{\mathbf{B}} = \mathbf{I} - \widetilde{\mathbf{W}}'$

5 Finally, to find a causal order, find the permutation matrix $\mathbf{P}$ (applied equally to both rows and columns) of $\widehat{\mathbf{B}}$ which yields a matrix $\widetilde{\mathbf{B}} = \mathbf{P}\widehat{\mathbf{B}}\mathbf{P}^T$ which is as close as possible to strictly lower triangular. This can be measured for instance using $\sum_{i \leq j} \widetilde{\mathbf{B}}_{ij}^2$.

2.2 Experiment

I did experiment on the dataset LUCAS from `http://www.causality.inf.ethz.ch/data/LUCAS.html`. LUCAS (LUng CAncer Simple set) contains toy data generated artificially by causal Bayesian networks with binary variables. It has 11 variables which are Smoking, Yellow_Fingers, Anxiety, Peer_Pressure, Genetics, Attention_Disorder, Born_an_Even_Day, Car_Accident, Fatigue, Allergy and Coughing. The result is shown in Figure 2. As we can see, the performance is not so good. Some egdes are totally reversed which are not reasonable. For instance, Fatigue should be the cause of Car_Accident instead of what you can see in the figure. Also, Anxiety and Peer_Pressure may cause someone Smoking but the relationships are reversed in the figure. Those unreasonable directions show that LiNGAM is not so good. But it can still found some cause-result relationships, for example, Smoking and Genetics both lead to Lung_cancer. Whatever, it is a very first method to do causal discovery, which still groundbreaking.

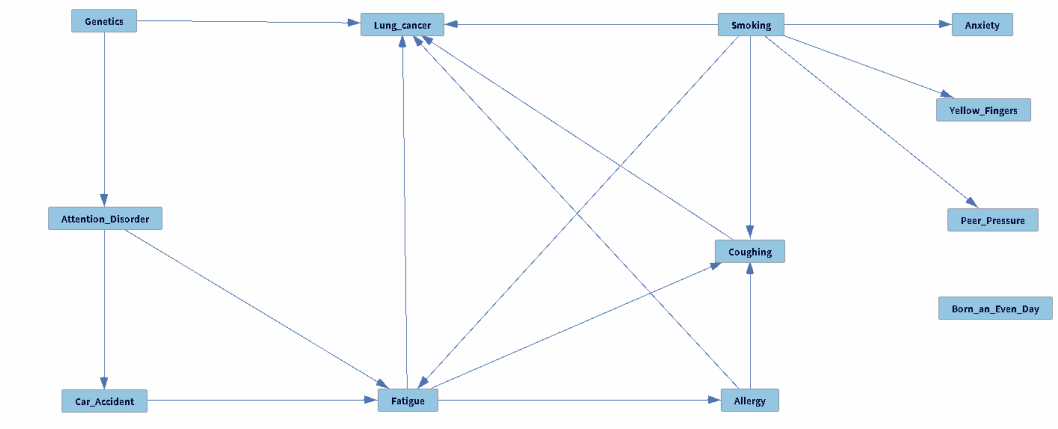

Figure 2: LiNGAM Result