CS3308 Homework 3

UNIkeEN

June 7, 2024

Contents

1 SVM vs. Neural Networks

1.1 Requirements

Select at least two data sets from the link (Data A) below, and then investigate classification performances of Support Vector Machine (SVM) and neural networks (e.g., MLP) on the selected data sets. You may try different experimental settings, e.g., varying the sample size of the training set, trying data sets with different dimensions, and other configurations that may affect the performance in your mind. You may also try different kernels for SVM.

Links to the data sets:

- Data A: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

1.2 Models

1.2.1 SVM

A Support Vector Machine (SVM) is a supervised learning algorithm widely used for classification tasks. It seeks to find the hyperplane that best separates different classes in the feature space. The optimal hyperplane is the one that maximizes the margin between the nearest points of the classes (called 'support vectors'). The mathematical formulation of a linear SVM aiming to classify data points can be described as follows:

Given a dataset of n points of the form $(\mathbf{x}_i, y_i)$ where $\mathbf{x}_i$ is a feature vector and y_i is the class label (with $y_i \in \{-1, 1\}$), the goal is to find the parameters $\mathbf{w}$ and b that solve the optimization problem:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i \tag{1}$$

subject to

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n. \tag{2}$$

This formulation emphasizes the linear case. The parameter C is a regularization parameter that controls the trade-off between maximizing the margin and minimizing the classification error. The ξ_i are slack variables introduced to allow misclassification of difficult or noisy cases, thus providing flexibility in the decision boundary for non-linearly separable cases.

However, not all data can be linearly separated in the original feature space. To handle such cases, SVMs use a technique called the "kernel trick" to transform the data into a higher-dimensional space where it becomes more likely to find a linear separation. This transformation is done implicitly, meaning that we do not need to compute the coordinates of the data in the high-dimensional space explicitly. Instead, we define a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ that computes the dot product of two data points in the transformed feature space. This allows us to apply the same linear classification algorithm to non-linear problems.

1. Linear Kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j \tag{3}$$

This is equivalent to the linear SVM described above.

2. Polynomial Kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + c)^d \tag{4}$$

where c is a constant and d is the degree of the polynomial. This kernel allows for curved decision boundaries.

3. Radial Basis Function (RBF) Kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2) \tag{5}$$

where γ is a parameter that defines the width of the Gaussian function. This kernel can handle very complex boundaries.

4. Sigmoid Kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\alpha \mathbf{x}_i \cdot \mathbf{x}_j + c) \tag{6}$$

where α and c are kernel parameters. This kernel is similar to neural networks.

The choice of kernel function and its parameters can significantly affect the performance of the SVM. Selecting the right kernel and tuning its parameters typically requires experimentation and cross-validation.

By using these kernels, SVMs can efficiently perform classification tasks even when the data is not linearly separable in the original feature space. This flexibility makes SVMs a powerful tool for a wide range of classification problems.

1.2.2 MLP

A Multi-Layer Perceptron (MLP) is a type of artificial neural network consisting of multiple layers of nodes. Shown as Fig. 1, a typical MLP with three types of layers: the input layer, one hidden layer, and the output layer. Each node (or neuron) in the input layer $(x_1, \cdots, x_i)$ receives the input features of the data. These inputs are then passed to the hidden layer $(h_1, \cdots, h_j)$, where each node applies a linear transformation followed by a non-linear activation function to the inputs.

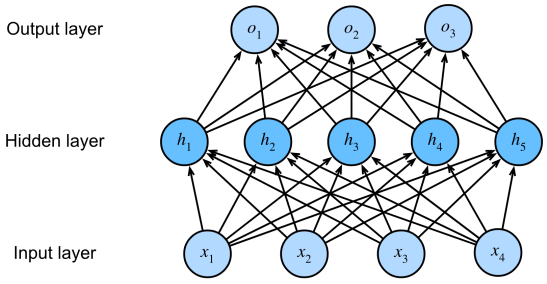


Fig. 1. MLP

The nodes in the hidden layer are fully connected to both the input layer and the output layer $(o_1, \cdots, o_k)$. This means that every node in one layer is connected to every node in the subsequent layer, allowing for complex interactions and representations of the data. The final layer, the output layer, produces the predictions of the network. For a classification task, these outputs might be transformed using a softmax function to yield class probabilities. The training process involves adjusting the weights of these connections to minimize the error between the predicted outputs and the actual target values using backpropagation and an optimization algorithm like gradient descent.

1.3 Experiments

1.3.1 Datasets and Platform

We selected the following datasets:

Table 1: Datasets

Name	Class	Data	Feature Dimension
colon-cancer	2	62	2000
diabetes	2	768	8
fourclass	2	862	2
vehicle	4	846	18

Among these datasets, `diabetes` and `fourclass` have a similar number of samples and both have 2 classes, but they differ in the number of feature dimensions. The `colon-cancer` dataset also has 2 classes, with a smaller number of samples but a very high number of feature dimensions. The `vehicle` dataset, on the other hand, has four classes.

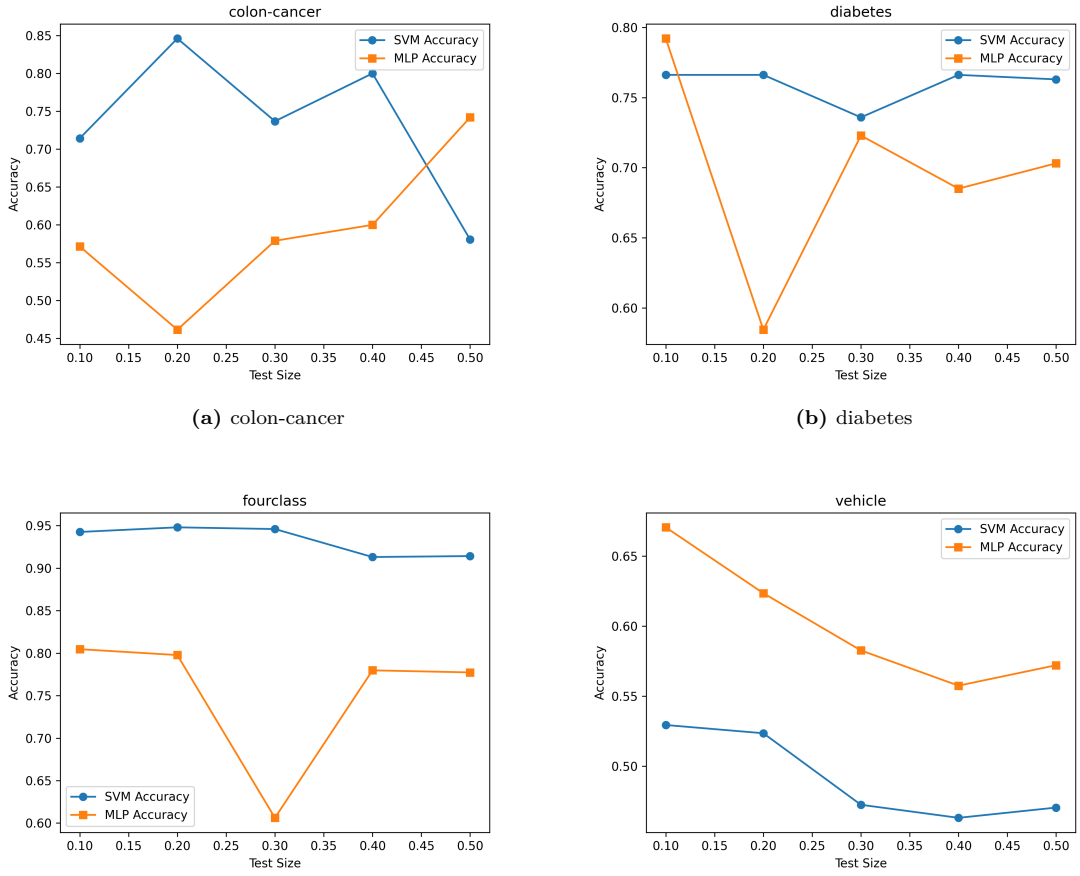
The testing platform utilized an Intel(R) Core(TM) i5-13600KF CPU.

1.3.2 Testset Ratio

Table 2: Accuracy(%) vs. Testset Ratio

Test Ratio		0.1	0.2	0.3	0.4	0.5
colon-cancer	SVM	71.43	84.62	73.68	80.00	58.06
	MLP	57.14	46.15	57.89	60.00	74.19
diabetes	SVM	76.62	76.62	73.59	76.62	76.30
	MLP	79.22	58.44	72.29	68.51	70.31
fourclass	SVM	94.25	94.80	94.59	91.30	91.42
	MLP	80.46	79.77	60.62	77.97	77.73
vehicle	SVM	52.94	52.35	47.24	46.31	47.04
	MLP	67.06	62.35	58.27	55.75	57.21

We selected datasets that do not have pre-split training and testing sets. Therefore, we used the `train_test_split` function provided by `sklearn` to split the data at different ratios. We trained and evaluated the accuracy of the models on the test sets using the SVM and MLP classifiers provided by `sklearn` without presetting any parameters. The results are as shown in Table 2 and Fig. 2.



(a) colon-cancer

(b) diabetes

(c) fourclass

(d) vehicle

Fig. 2. Accuracy vs. Testset Ratio

We can see that in most cases, the model performs best when the test ratio is 0.1. For the `diabetes` and `fourclass` datasets, which are relatively balanced, the models generalize well and exhibit minimal SVM performance fluctuations with changes in the test ratio. For the `vehicle` dataset, the model performances are lower. This could be due to underfitting caused by the preset model configuration. For convenience, we will use **0.2** as the testset ratio for subsequent experiments. This ratio ensures relatively good model performance and, compared to the lower ratio of 0.1, it better evaluates the model's generalization ability and is more representative.

1.3.3 Hypermeter and Kernel of SVM

As previously mentioned, the C value is a crucial hyperparameter for SVM. As the C value increases, the SVM will fit the training data more strictly. We conducted experiments using SVM with **RBF, polynomial and sigmoid kernels**, with the C value ranging from [0.01, 0.03, 0.1, 0.3, 1.0, 10.0, 100.0, 1000, 10000].

Table 3: Accuracy(%) vs. C in RBF Kernel SVM

C	0.01	0.03	0.1	0.3	1	10	100	1000	10000
colon-cancer	61.54	61.54	61.54	61.54	84.62	76.92	76.92	76.92	76.92
diabetes	64.29	64.29	66.23	75.32	76.62	75.97	74.03	70.78	69.48
fourclass	68.79	80.92	83.82	88.44	94.80	98.84	99.42	99.42	99.42
vehicle	17.65	34.12	36.47	40.59	52.35	67.65	74.71	82.35	85.88

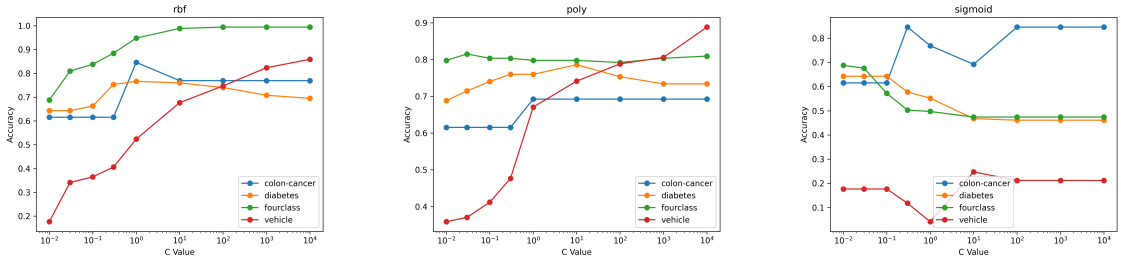
Table 4: Accuracy(%) vs. C in Polynomial Kernel SVM

C	0.01	0.03	0.1	0.3	1	10	100	1000	10000
colon-cancer	61.54	61.54	61.54	61.54	69.23	69.23	69.23	69.23	69.23
diabetes	68.83	71.43	74.03	75.97	75.97	78.57	75.32	73.38	73.38
fourclass	79.77	81.50	80.35	80.35	79.77	79.77	79.19	80.35	80.92
vehicle	35.88	37.06	41.18	47.65	67.06	74.12	78.82	80.59	88.82

Table 5: Accuracy(%) vs. C in Sigmoid Kernel SVM

C	0.01	0.03	0.1	0.3	1	10	100	1000	10000
colon-cancer	61.54	61.54	61.54	84.62	76.92	69.23	84.62	84.62	84.62
diabetes	64.29	64.29	64.29	57.79	55.19	46.75	46.10	46.10	46.10
fourclass	68.79	67.63	57.23	50.29	49.71	47.40	47.40	47.40	47.40
vehicle	17.65	17.65	17.65	11.76	4.12	24.71	21.18	21.18	21.18

Due to running time constraints, we did not conduct experiments with linear SVM. The results show that the four datasets achieved maximum test accuracies of **84.62%, 78.57%, 99.42%, and 88.82%** under different conditions. An interesting observation is that for the `vehicle` dataset, accuracy has a positive correlation with the C value under both RBF and polynomial kernels. This indicates the presence of nonlinear and complex patterns in the dataset, which these kernels can fit well. However, the performance is significantly worse with the sigmoid kernel, which has a relatively simple decision boundary. On the other hand, the `fourclass` dataset achieved **near-perfect accuracy** with RBF kernel SVM at higher C values, suggesting the presence of minimal noise and small feature dimensions in the dataset, preventing overfitting and underfitting by the model.



(a) RBF **(b)** Polynomial **(c)** Sigmoid

Fig. 3. Accuracy vs. C in Kernel SVM

1.3.4 Structure of MLP

For the MLP model, we fixed the activation function as `ReLU` and the optimizer as `Adam` with an automatic learning rate. We trained the model with different hidden layer structures, and the results are as follows:

Table 6: Accuracy(%) vs. MLP's Structure

Hidden Layers	50	100	100,50	100,100	100,200	100,200,100	100,200,100,50
colon-cancer	53.85	46.15	61.54	69.23	61.54	61.54	61.54
diabetes	64.29	58.44	66.23	64.94	64.94	66.23	56.49
fourclass	79.19	79.77	75.14	80.35	74.57	75.72	74.57
vehicle	61.18	62.35	62.35	65.29	69.41	66.47	69.41

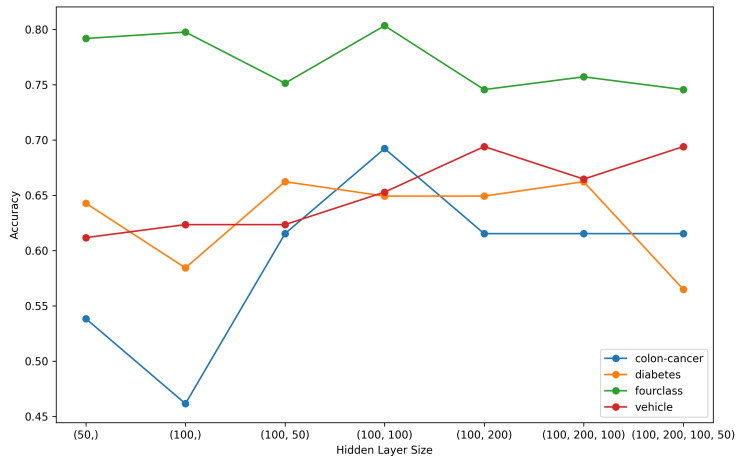


Fig. 4. Accuracy vs. MLP's Structure

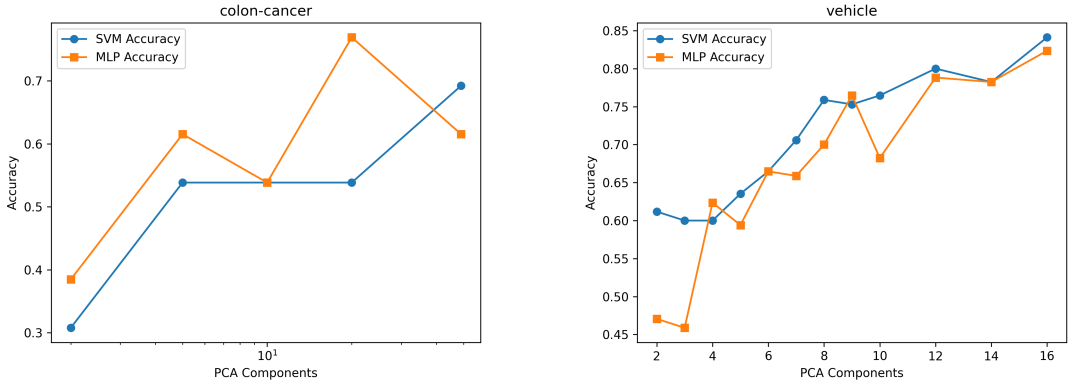
From the results above, it is evident that for the `fourclass` dataset, which has a relatively small feature dimension, the performance on the MLP model is significantly better than the other

datasets. For the `colon-cancer` dataset, it is noticeable that as the model complexity increases, the test accuracy initially rises and then falls.

1.3.5 Data Dimension

We used PCA and LDA methods to conduct dimensionality reduction experiments on the `colon-cancer` and `vehicle` datasets, as they have the highest dimensions among the selected datasets.

For the `vehicle` dataset, the target dimensions for PCA were set to [2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16]; for the `colon-cancer` dataset, which originally has 2000 dimensions, the target dimensions for PCA were set to [2, 5, 10, 20, 49] (the training set size after splitting is 49, so the PCA `n_components` cannot exceed this value). The parameters for SVM and MLP were taken from the best parameters of the previous experiments on these two datasets. The results are as follows:

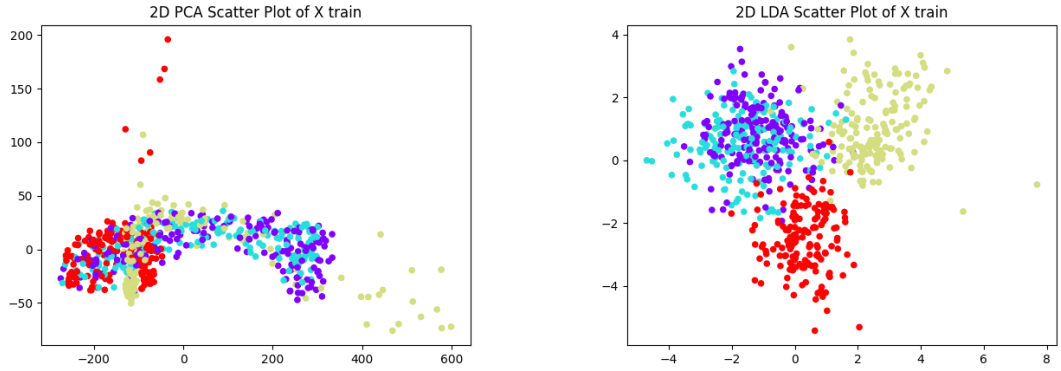


(a) colon-cancer **(b)** diabetes

Fig. 5. Accuracy vs. PCA Target Dimension

From the Fig. 5, it can be observed that the performance of SVM generally improves with the increase in data dimensions, indicating that PCA dimensionality reduction, while reducing complexity, also loses some necessary information. The performance of MLP fluctuates and surpasses SVM at certain dimensions, possibly suggesting that it has a stronger ability to extract patterns from fewer dimensions.

We also conducted LDA experiments on the four-class `vehicle` dataset. According to the principles of LDA, it can reduce the dimensions to at most the number of classes minus one, which is 3 in this case. We set `n_components` to 2, and the test accuracies of the SVM and MLP models with LDA were **72.35%** and **75.88%**, respectively, significantly higher than the results with PCA at 2 dimensions (**60.00%** and 4**5.88%**). We also give the 2-components scattering figure of PCA and LDA, shown in Fig. 6, which supports LDA have better classes distance than PCA.



(a) PCA **(b)** LDA

Fig. 6. 2-Components Scattering Figure

1.4 Conclusion

In this experiment, we first established a training-to-test set ratio of 8:2, which has shown good results in both SVM and MLP experiments, laying a foundation for subsequent experiments. The following experiments fully demonstrated that different datasets require different SVM and MLP parameter configurations. Among them, the `fourclass` dataset, which has the fewest features, was able to achieve nearly perfect performance. SVM generally performs well in solving classification problems and is more versatile for different types of problems. On the other hand, MLP has more potential than SVM, with significant room for improvement. Regarding dimensionality reduction, LDA exhibited better feature projection performance than PCA in this experiment.

2 Causal Discovery Algorithms

2.1 Requirements

Apply one causal discovery algorithm on a real world problem. You need to specify the details of the problem, collect the data by yourself or from a public website, briefly summarize what algorithm you use, and explain the results. You may use any causal discovery algorithm described in the paper [1] and use the software packages in Page 26 of the paper.

2.2 Introduction

Testing large-scale software systems is a vast and systematic project. In China, the back-end servers of many internet companies use the Springboot framework and microservice architecture based on Java. The high number of concurrent requests heavily relies on performance tuning of the Java runtime. In my work at Game-play Department of SJTU Minecraft Club, I have encountered various Minecraft servers and plugins, all implemented in Java. Understanding the causal relation-

ships between JDK parameters, memory usage, compilation time, and package size is particularly important in the small-scale, multi-instance server environment of a student club. To explore these issues, we used datasets from `https://groups.cs.umass.edu/kdl/causal-eval-data` to implement causal discovery algorithms.

2.3 Algorithm

2.3.1 Peter-Clark Algotirhm

The PC algorithm (named after its creators Peter and Clark) is a popular method for causal discovery in the field of statistics and machine learning. It aims to infer the causal structure from observational data by identifying the underlying causal relationships between variables. The algorithm is based on conditional independence tests and operates in two main phases: the skeleton phase and the orientation phase.

In the skeleton phase, the algorithm starts with a fully connected undirected graph, where each node represents a variable. It then iteratively removes edges by performing conditional independence tests. The basic idea is to test whether two variables X and Y are conditionally independent given a set of other variables Z. If X and Y are found to be conditionally independent given Z, the edge between X and Y is removed. This phase is crucial for reducing the complexity of the graph and identifying the potential direct causal links.

Mathematically, the conditional independence test can be represented as:

$$X \perp Y \mid Z \tag{7}$$

where $\perp$ denotes independence. If $P(X, Y \mid Z) = P(X \mid Z)P(Y \mid Z)$, then X and Y are conditionally independent given Z, and the edge $X - Y$ is removed.

Once the skeleton (the undirected graph) is established, the algorithm proceeds to the orientation phase. In this phase, the edges are directed based on certain rules to infer causal relationships. One key rule is the v-structure rule, which states that if there are edges $X - Z$ and $Y - Z$ but no edge $X - Y$, and X and Y are not conditionally independent given any subset that excludes Z, then the edges are oriented as $X \rightarrow Z \leftarrow Y$.

Another important rule is the preservation of the acyclic nature of the graph, ensuring no directed cycles are formed. The orientation phase continues until all edges are either oriented or cannot be further oriented without violating the acyclic constraint.

The PC algorithm has certain limitations, such as sensitivity to the sample size and the choice of conditional independence tests. Small sample sizes can lead to unreliable conditional independence tests, resulting in incorrect causal structures. Additionally, the algorithm assumes the absence of hidden confounders and that the data-generating process can be represented by a Directed Acyclic Graph (DAG).

Despite these limitations, the PC algorithm remains a widely used and powerful tool for causal discovery, particularly when dealing with high-dimensional data where other methods may be computationally infeasible.

2.4 Experiments

2.4.1 Datasets and Platform

This dataset contains a sample of Maven-enabled Java projects from GitHub. The researcher from UMass Amherst compiled and ran the unit tests of each project, varying JDK options and monitoring runtime behavior. Here are the details from the dataset website:

Table 7: JDK Dataset Details

Field Name	**Category**	**Description**
repo_name	subject identifier	Name of the GitHub repository containing the experimentation code
trial	trial identifier	
debug	treatment	Indicates whether debug symbols were requested during compilation
obfuscate	treatment	Indicates whether a code obfuscator was run on the final JAR file
parallelgc	treatment	Indicates whether a parallel garbage collection was employed during execution (instead of serial garbage collection)
num_bytecode_ops	outcome	Number of bytecode instructions in the compiled code
total_unit_test_time	outcome	Number of seconds required to execute unit tests
allocated_bytes	outcome	Number of bytes allocated during execution of unit tests
jar_file_size_bytes	outcome	Size of JAR file after compilation (and possibly obfuscation)
compile_time_ms	outcome	Number of milliseconds to compile the source
source_ncss	covariate	Number of non-comment source statements in the source code
test_classes	covariate	Number of Java classes in the unit test source
test_functions	covariate	Number of functions in the unit test source
test_ncss	covariate	Number of non-comment source statements in the test source
test_javadocs	covariate	Number of JavaDoc comments in the test source

The testing platform utilized an Intel(R) Core(TM) i5-13600KF CPU.

2.4.2 Result

We use `causallearn` package to implement the algorithms. Here is the result, arrow in the picture means causality.

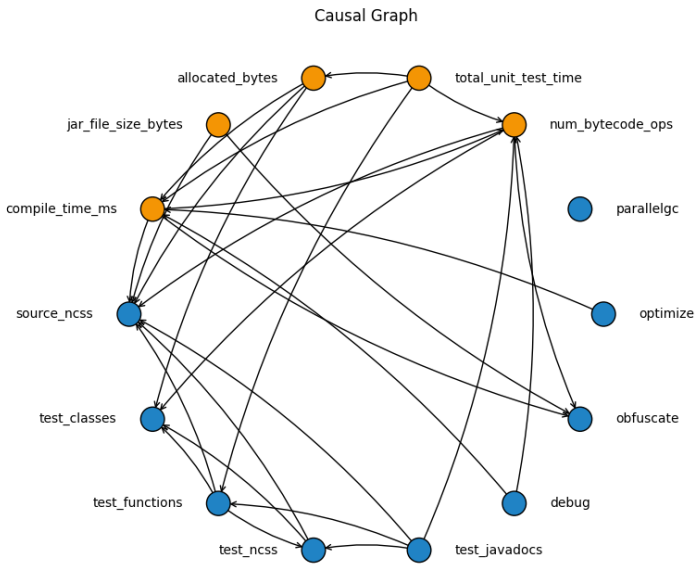


Fig. 7. PC Result

From the graph, it can be seen that compile time is influenced by many factors (such as debug and optimize) and is not directly related to the size of the packaged jar file. However, there are many test-related covariates that receive arrows from the outcome, This result is somewhat confusing to me, Indicating that the causal inference of the model may be not effective in this regard.

Furthermore, we also tried another algorithm GES, here is the result, it's similar to the result of PC on test-related data:

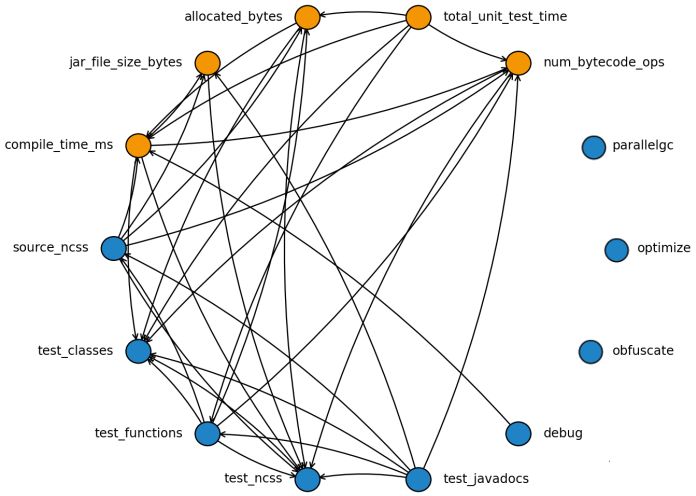


Fig. 8. GES Result

References

[1] P. Spirtes and K. Zhang, "Causal discovery and inference: concepts and recent methodological advances," in *Applied informatics*, vol. 3, pp. 1–28, Springer, 2016.