# Feature Ranking Using Linear SVM

**Yin-Wen Chang** B92059@CSIE.NTU.EDU.TW
**Chih-Jen Lin** CJLIN@CSIE.NTU.EDU.TW
*Department of Computer Science, National Taiwan University*
*Taipei 106, Taiwan*

**Editors:** I. Guyon, C. Aliferis, G. Cooper, A. Elisseeff, J.-P. Pellet, P. Spirtes, and A. Statnikov

## Abstract

Feature ranking is useful to gain knowledge of data and identify relevant features. This article explores the performance of combining linear support vector machines with various feature ranking methods, and reports the experiments conducted when participating the Causality Challenge. Experiments show that a feature ranking using weights from linear SVM models yields good performances, even when the training and testing data are not identically distributed. Checking the difference of Area Under Curve (AUC) with and without removing each feature also gives similar rankings. Our study indicates that linear SVMs with simple feature rankings are effective on data sets in the Causality Challenge.

**Keywords:** SVM, feature ranking.

## 1. Introduction

The Causality Challenge (Guyon et al., 2008) aims at investigating situations where the training and testing sets might have different distributions. The goal is to make predictions on manipulated testing sets, where some features are disconnected from their natural cause. Applications of the problem include predicting the effect of a new policy or predicting the effect of a new drug. In both examples, the experimental environment and the real environment differ.

In order to make good predictions on manipulated testing sets, we use several feature ranking methods to gain knowledge of the data. Among existing approaches to evaluate the relevance of each feature, some are related to certain classification methods, but some are more general. Those independent of classification methods are often based on statistic characteristics. For example, we experimented with Fisher-score, which is the correlation coefficient between one of the features and the label. In this work, we select Support Vector Machines (SVMs) (Boser et al., 1992) as the classifier, and consider one feature ranking method specific to SVM (Guyon et al., 2002).

This article is organized as follows. In Section 2 we introduce support vector classification. Section 3 describes several feature ranking strategies. Section 4 presents experiments conducted during the development period of the competition, our competition results, and some post-challeng analysis. Closing discussions are in Section 5.





## 2. Support Vector Classification

Support vector machines (SVMs) are useful for data classification. It finds a separating hyperplane with the maximal margin between two classes of data. Given a set of instance-label pairs $(\boldsymbol{x}_i, y_i), \boldsymbol{x}_i \in R^n, y_i \in \{1, -1\}, i = 1, \ldots, l$, SVM solves the following unconstrained optimization problem:

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\xi(\boldsymbol{w}, b; \boldsymbol{x}_i, y_i), \tag{1}$$

where $\xi(\boldsymbol{w}, b; \boldsymbol{x}_i, y_i)$ is a loss function, and $C \geq 0$ is a penalty parameter on the training error. Two common loss functions are:

$$\max(1 - y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b), 0) \text{ and } \max(1 - y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b), 0)^2, \tag{2}$$

where $\phi$ is a function that mapped training data into higher dimensional space. The former is called L1-loss SVM, and the latter is L2-loss SVM. When participating in the challenge, we choose the L2-loss function. Post-challenge experiments show that the two loss functions result in similar performances. We give detailed results of using both loss functions in Section 4.3.

For any testing instance $\boldsymbol{x}$, the decision function (predictor) is

$$f(\boldsymbol{x}) = \operatorname{sgn}\left(\boldsymbol{w}^T\phi(\boldsymbol{x}) + b\right). \tag{3}$$

Practically, a kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$ may be used to train the SVM. A linear SVM has $\phi(\boldsymbol{x}) = \boldsymbol{x}$ so the kernel function is $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T\boldsymbol{x}_j$. Another popular kernel is the radial basis function (RBF):

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2), \text{ where } \gamma > 0. \tag{4}$$

We use linear SVM for both feature ranking and classification in the challenge. We also conduct some post-challenge experiments using SVM with RBF kernel as the classifier. The results will be discussed in Section 4.3.

We use grid search to determine the penalty parameter $C$ for linear SVM, and both $C$ and $\gamma$ for SVM with RBF kernel. For each value of $C$ or $(C, \gamma)$, we conduct five-fold cross validation on the training set, and choose the parameters leading to the highest accuracy.

## 3. Feature Ranking Strategies

In this section, we describe several feature ranking strategies that we experiment with in the challenge. All methods assign a weight to each feature and rank the features accordingly.

### 3.1 F-score for Feature Ranking

F-score (Fisher score) is a simple and effective criterion to measure the discrimination between a feature and the label. Based on statistic characteristics, it is independent of the classifiers. Following Chen and Lin (2006), a variant of F-score is used. Given training

**Algorithm 1** Feature Ranking Based on Linear SVM Weights

**Input**: Training sets, $(\boldsymbol{x}_i, y_i), i = 1, \ldots, l$.
**Output**: Sorted feature ranking list.

1. Use grid search to find the best parameter $C$.
2. Train a L2-loss linear SVM model using the best $C$.
3. Sort the features according to the absolute values of weights in the model.

instances $\boldsymbol{x}_i, i = 1, \ldots, l$, the F-score of the $j$th feature is defined as:

$$F(j) \equiv \frac{\left(\bar{\boldsymbol{x}}_j^{(+)} - \bar{\boldsymbol{x}}_j\right)^2 + \left(\bar{\boldsymbol{x}}_j^{(-)} - \bar{\boldsymbol{x}}_j\right)^2}{\frac{1}{n_+ - 1}\sum_{i=1}^{n_+}\left(x_{i,j}^{(+)} - \bar{\boldsymbol{x}}_j^{(+)}\right)^2 + \frac{1}{n_- - 1}\sum_{i=1}^{n_-}\left(x_{i,j}^{(-)} - \bar{\boldsymbol{x}}_j^{(-)}\right)^2}, \tag{5}$$

where $n_+$ and $n_-$ are the number of positive and negative instances, respectively; $\bar{\boldsymbol{x}}_j$, $\bar{\boldsymbol{x}}_j^{(+)}$, $\bar{\boldsymbol{x}}_j^{(-)}$ are the average of the $j$th feature of the whole, positive-labeled, and negative-labeled data sets; $x_{i,j}^{(+)}/x_{i,j}^{(-)}$ is the $j$th feature of the $i$th positive/negative instance. The numerator denotes the inter-class variance, while the denominator is the sum of the variance within each class. A larger F-score indicates that the feature is more discriminative.

A known deficiency of F-score is that it considers each feature separately and therefore cannot reveal mutual information between features. However, F-score is simple and generally quite effective.

### 3.2 Linear SVM Weight for Feature Ranking

After obtaining a linear SVM model, $\boldsymbol{w} \in R^n$ in (1) can be used to decide the relevance of each feature (Guyon et al., 2002). The larger $|w_j|$ is, the $j$th feature plays a more important role in the decision function (3). Only $\boldsymbol{w}$ in linear SVM model has this indication, so this approach is restricted to linear SVM. We thus rank features according to $|w_j|$. The procedure is in Algorithm 1.

### 3.3 Change of AUC with/without Removing Each Feature

We determine the importance of each feature by considering how the performance is influenced without that feature. If removing a feature deteriorates the classification performance, the feature is considered important. We select the cross validation AUC as the performance measure. Features are ranked according to the AUC difference.

This performance-based method has the advantage of being applicable to all classifiers. The disadvantage is that it takes a huge amount of time to train and predict when the number of features is large. Besides, by removing only one feature at a time, the method does not take into account how features affect each other.





Table 1: Challenge data sets. All of them have two classes.

| Dataset | Feature type | # Feature | # Training | # Testing |
|---|---|---|---|---|
| REGED | numerical | 999 | 500 | 20,000 |
| SIDO | binary | 4,932 | 12,678 | 10,000 |
| CINA | mixed | 132 | 16,033 | 10,000 |
| MARTI | numerical | 1,024 | 500 | 20,000 |
| LUCAS | binary | 11 | 2,000 | 10,000 |
| LUCAP | binary | 143 | 2,000 | 10,000 |

### 3.4 Change of Accuracy with/without Removing Each Feature

This method is the same as the one described in Section 3.3, except that the measure of performances is the accuracy rate.

## 4. Experimental Results

In the Causality Challenge, there are four competition tasks (REGED, CINA, SIDO and MARTI) and two small toy examples (LUCAS and LUCAP). All tasks have three versions of data sets, each with the same training set, and different testing sets. Testing sets with digit zero indicates unmanipulated testing set, while digit one and two denote manipulated testing sets. Table 1 shows the data set descriptions. Details can be found at `http://www.causality.inf.ethz.ch/challenge.php`.

We preprocess data via scaling, instance-wise normalization, and Gaussian filtering. We scale each feature of REGED and CINA to $[0, 1]$, and apply the same scaling parameter to their testing sets. In contrast, training and testing sets in MARTI are separately scaled to $[-1, 1]$ for each feature, since this way results in a better performance. Another reason is that the training data in MARTI are perturbed by noises, while the testing data are free of noises. After applying a Gaussian filter on the training set to filter out the noises, there is an unknown bias value that we would like to substrate or add. We might use information from the distribution of testing data to gain knowledge of the unknown bias value, and then scale the training and testing data using the same scaling parameter. Alternatively, we can ignore the bias value, and scale the training and testing data separately. For SIDO, LUCAS, and LUCAP, the range of their features are already in $[0, 1]$. We normalize each instance of these three problems to have the unit length.

According to the data set description, two kinds of noise are added to MARTI. First, to obtain 1,024 features, 999 features in REGED are complemented by 25 calibrant features, each of which has a value zero plus a small Gaussian noise. Second, the training set is perturbed by a zero-mean correlated noise. Since we cannot get into the first quartile of the competition results without regarding the noise, we use a Gaussian filter to eliminate the low frequency noise in the training set before scaling. For each instance, we rearrange the 1,024 features into a 32×32 array and apply the Gaussian filter, according to the fact that neighboring positions are similarly affected. The low pass spatial Gaussian filter is defined

**Algorithm 2** Training and Prediction

**Input**: Training sets, testing sets.

**Output**: predictions on nested subsets.

1. Use a feature ranking algorithm to compute the sorted feature list $f_j, j = 1, \ldots, n$.
2. For each feature size $m \in \{1, 2, 4, \ldots, 2^i, \ldots, n\}$.
   (a) Generate the new training set that has only the first $m$ features in the sorted feature list, $f_j, j = 1, \ldots, m$.
   (b) Use grid search to find the best parameter $C$.
   (c) Train the L2-loss linear SVM model on the new training set.
   (d) Predict the testing set using the model.

as:

$$g(x_0) = \frac{1}{G(x_0)} \sum_x e^{-\frac{1}{2}(\frac{\|x-x_0\|}{\sigma})^2} f(x), \text{ where } G(x_0) = \sum_x e^{-\frac{1}{2}(\frac{\|x-x_0\|}{\sigma})^2} \quad (6)$$

where $f(x)$ is the value at position $x$ in the 32×32 array. For each position $x$, we take the Gaussian weighted average of all values in the array. The resulting $g(x)$ is the approximated low frequency noise we derive, and $f'(x) = f(x) - g(x)$ is the feature value that we would like to use. The $\sigma$ is set to 3.2 after experimenting with several values.

Since testing sets may not follow the same distribution as training sets, and it is intended to hide their distributions, no validation sets are provided during the development period, which is the time between the start and the termination of the challenge. Instead, an on-line submission page shows which quartile that submission belongs to among all submissions. Besides, testing AUC of toy examples are available.

The linear SVM classifier that we use is LIBLINEAR[1] (Fan et al., 2008), and we use LIBSVM[2] (Chang and Lin, 2001) for SVM with RBF kernel. While LIBSVM can handle linear kernel as well, we use LIBLINEAR due to its special design for linear SVM. Our implementation extends from the framework by Chen and Lin (2006)[3]. All sources for our experiments are available at `http://www.csie.ntu.edu.tw/~cjlin/papers/causality`.

We experiment with the feature ranking methods described in Section 3. We use F-score, W, D-AUC, D-ACC to denote the methods in Sections 3.1-3.4, respectively. The linear SVM weights are derived from LIBLINEAR model files. The procedure is described in Algorithm 2.

We summarize the methods that we experiment with:

- F-score: feature ranking using F-score described in Section 3.1.
- W: feature ranking using linear SVM weights described in Section 3.2.

1. `http://www.csie.ntu.edu.tw/~cjlin/liblinear`
2. `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
3. `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools`





Table 2: Best five-fold cross validation AUC and the corresponding feature size. The best feature ranking approach is bold-faced.

| Dataset | REGED | | SIDO[4] | | CINA | | MARTI | |
|---|---|---|---|---|---|---|---|---|
| F-score | 0.9998 | (16) | 0.9461 | (2,048) | 0.9694 | (132) | 0.9210 | (512) |
| W | **1.0000** | (32) | **0.9552** | (512) | **0.9710** | (64) | 0.9632 | (128) |
| D-AUC | **1.0000** | (16) | – | | 0.9699 | (128) | **0.9640** | (256) |
| D-ACC | 0.9998 | (64) | – | | 0.9694 | (132) | 0.8993 | (32) |

- D-AUC: feature ranking by checking the change of AUC with/without removing each feature. Detailes are in Section 3.3.
- D-ACC: feature ranking by checking the change of accuracy with/without removing each feature. Details are in Section 3.4.

### 4.1 Development Period

During the development period, we took into account the cross validation AUC on training sets, the testing AUC of toy examples, and the quartile information to decide the method for the final submission.

Since we did not develop a strategy to deal with different training/testing distributions, we use the same model to predict each task's three testing sets. We did not use the provided information of the manipulated features in REGED and MARTI, and the 25 calibrant features in MARTI.

With AUC being the evaluation criterion, we submitted the decision values of linear SVM predictions. Nested subsets according to sorted feature lists are used since their performances are better. That is, one of the predictions based on a subset outperforms the one based on the whole feature set.

For F-score and W, it takes less than one minute to train all the models for nested-subset submissions for REGED, CINA, and MARTI, while it takes about 13 minutes for SIDO. Excluding preprocessing, the time required to predict one testing set is around five minutes for REGED and MARTI, 16 seconds for CINA, and three minutes for SIDO. SIDO is more computational costly to train and predict due to a larger number of features and training instances. For D-AUC and D-ACC, it takes a few hours to get the feature rank.

We submitted totally 60 entries before the challenge ended. Among methods we have tried, W has testing AUC in the first quartile for all data sets. This result seems to indicate that it is better than others. We used cross-validation with AUC in order to get more definitive conclusions.

Table 2 shows the five-fold cross validation AUC of using the best feature size. We conduct cross validation on all feature size $\in \{1, 2, 4, \ldots, 2^i, \ldots, n\}$, where $n$ is the total number of features. W and D-AUC seem to perform better than other methods, while D-ACC is the worst.

4. D-AUC and D-ACC are infeasible for SIDO due to the large number of features of SIDO.

Table 3: Comparisons of the performance on toy examples. The testing AUC is showed. Sorted feature list and nested subsets on it are used.

| Dataset | LUCAS 0 | LUCAS 1 | LUCAS 2 | LUCAP 0 | LUCAP 1 | LUCAP 2 |
|---|---|---|---|---|---|---|
| F-score | 0.9208 | 0.8989 | 0.7446 | **0.9702** | 0.8327 | 0.7453 |
| W | 0.9208 | 0.8989 | **0.7654** | **0.9702** | **0.9130** | **0.9159** |
| D-AUC | 0.9208 | 0.8989 | **0.7654** | 0.9696 | 0.8648 | 0.8655 |
| D-ACC | 0.9208 | 0.8989 | 0.7446 | 0.9696 | 0.7755 | 0.6011 |

We find that D-ACC differs most from others, while the other three methods are more similar. Especially, the top ranked features chosen by W and D-AUC are alike. For example, W and D-AUC have exactly the same top four features for CINA, and the same set of top eight features with slightly different rankings for REGED.

In Table 3, we compare different feature ranking methods according to the testing AUC of the toy examples, LUCAS and LUCAP. We can see that W still outperforms others. It is much better than other methods especially on manipulated testing data sets (see LUCAP 1 and LUCAP 2). Similar to the cross validation results, D-ACC is the worst.

### 4.2 Competition Results

Table 4 shows the results of our final submission. Fnum is the best number of features to make prediction. It is determined by the organizers according to the nested-subset submissions. Fscore indicates how good the ranking is according to the causal relationships known only to the organizers. Tscore is the testing AUC. Top Ts is the maximal score of the last entry made by all participants, and Max Ts is the best score reachable, estimated using causal relationship knowledge not available to participants.

We explain that on CINA 2, our method might benefit from good feature ranking. Our result is the best among all submissions. The four features used might be the direct cause of the label. As mentioned earlier, W and D-AUC identify exactly the same top four features. Similarly for MARTI 2 and REGED 2, Fnum is small and W and D-AUC select the same set of features, although the rankings are slightly different.

We also observe that the Fnums of the final submission are similar to the best feature size given by the cross validation results on the training data. However, we benefit from the nested-subset submission, since we do not select the best feature size. According to the rule, the best feature size is selected according to the testing AUC, so the testing set information is used indirectly.

Although the challenge is designed in a way that casual discovery is required to make good predictions, our simple feature ranking method performs rather well. It is interesting that our simple method outperforms some more complicated casual discovery methods.

However, the good performances do not indicate that the highly ranked features are important causes. Our methods rank the features according to their relevance, not their causal importance, and, thus, they do not enhance our knowledge of the underlying causal relationships between features.





Table 4: The results of our final submission in the Causality Challenge. We obtain feature ranking using linear SVM weights. The column "Fnum" shows the best feature size to make prediction and the total number of features.

| Dataset | Fnum | Fscore | Tscore | Top Ts | Max Ts | Rank |
|---|---|---|---|---|---|---|
| REGED 0 | 16/999 | 0.8526 | 0.9998 | 1.0000 | 1.0000 | |
| REGED 1 | 16/999 | 0.8566 | 0.9556 | 0.9980 | 0.9980 | |
| REGED 2 | 8/999 | 0.9970 | 0.8392 | 0.8600 | 0.9543 | |
| mean | | | 0.9316 | | | 1 |
| SIDO 0 | 1,024/4,932 | 0.6516 | 0.9432 | 0.9443 | 0.9467 | |
| SIDO 1 | 4,096/4,932 | 0.5685 | 0.7523 | 0.7532 | 0.7893 | |
| SIDO 2 | 2,048/4,932 | 0.5685 | 0.6235 | 0.6684 | 0.7674 | |
| mean | | | 0.7730 | | | 2 |
| CINA 0 | 64/132 | 0.6000 | 0.9715 | 0.9788 | 0.9788 | |
| CINA 1 | 64/132 | 0.7053 | 0.8446 | 0.8977 | 0.8977 | |
| CINA 2 | 4/132 | 0.7053 | 0.8157 | 0.8157 | 0.8910 | |
| mean | | | 0.8773 | | | 1 |
| MARTI 0 | 256/1,024 | 0.8073 | 0.9914 | 0.9996 | 0.9996 | |
| MARTI 1 | 256/1,024 | 0.7279 | 0.9209 | 0.9470 | 0.9542 | |
| MARTI 2 | 2/1,024 | 0.9897 | 0.7606 | 0.7975 | 0.8273 | |
| mean | | | 0.8910 | | | 3 |

Our linear SVM classifier has excellent performances on version 0 on all tasks. Our Tscore is close to Top Ts. However, compared with the best performance by other participants, the performance on version 1 is slightly worse, and the performance on version 2 is still worse than that on version 1. As the ranking for each task is determined according to the average of the performances on the three testing sets, we might take the advantage of good performances on version 0, where the testing and training distributions are the same.

Figure 1 shows the profile of the selected features (i.e., top Fnum features). This figure is provided by the organizers. The noise filtering method we used might not be good enough since for MARTI 0 and MARTI 1, the ratios of "direct causes" features are low compared with other methods. Besides, our feature ranking method ranks both direct causes and direct effects in the front of the list. They together make up most of the features on version 0. This result is reasonable since our methods do not consider causal relationships and therefore not necessarily rank true causes on the top. In Table 4, we have excellent performances on version 0 of all tasks. On manipulated testing sets, the ratio of unrelated features become higher, and our performance of these two versions are not as good as version 0. The only exception is CINA 2, where we did not obtain any unrelated features.

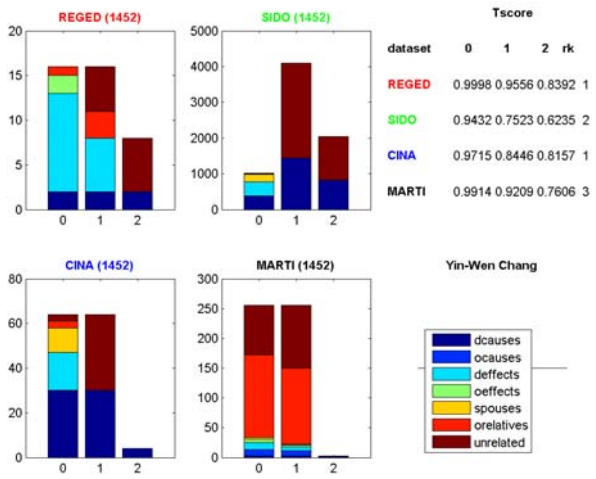


Figure 1: Profile of features selected (provided by the competition organizers). dcause: direct cause, deffect: direct effects, ocauses: other causes, oeffects: other effects, spouses: parent of a direct effect, orelatives: other relatives, unrelated: completely irrelevant.

### 4.3 Post-Challenge Experiments

After the challenge, the testing AUC values of our past submissions are available. We are able to compare the results of all methods, including L2-loss linear SVM with different feature ranking methods and a direct use of SVM without feature ranking. We also conduct post-challenge experiments to compare the feature ranking methods using L1-loss SVM. Besides, in order to see if nonlinear SVMs help to improve the performance, we apply the feature rankings obtained from L2-loss linear SVM to nonlinear SVM with the RBF kernel.

Table 5 shows the testing AUC revealed after the challenge ended. LINEAR stands for a direct use of L2-loss linear SVM. It is worth noticing that similar to Tables 2 and 3, W is generally the best. This result is interesting as for testing AUC in Table 5, training and testing sets are from different distributions. An exception where F-score has better testing AUC than W is REGED. D-ACC is still the worst though the difference to other methods becomes much smaller.

In order to understand the difference between using L1-loss and L2-loss functions, we experiment with L1-loss linear SVM to rank features and classify data instances. The results are in Table 6. In general, the testing AUC values do not differ much from those of L2-loss SVM in Table 5. However, here we do not have a solid conclusion that W outperforms other methods. Instead, most methods win on some data sets.





Table 5: Comparison of different feature ranking methods using L2-loss linear SVM. It shows testing AUC and the corresponding Fnum, revealed after the challenge has ended. We did not run D-AUC and D-ACC on SIDO, so some slots in this table are blank.

| | Feature ranking methods | | | | | | | | SVM |
|---|---|---|---|---|---|---|---|---|---|
| Dataset | F-score | | W | | D-AUC | | D-ACC | | LINEAR |
| REGED 0 | **0.9998** | (64) | **0.9998** | (16) | 0.9997 | (16) | 0.9987 | (128) | 0.9970 |
| REGED 1 | 0.9555 | (32) | **0.9556** | (16) | 0.9528 | (16) | 0.9438 | (999) | 0.9438 |
| REGED 2 | **0.8510** | (8) | 0.8392 | (8) | 0.8392 | (8) | 0.8113 | (32) | 0.7442 |
| mean | **0.9354** | | 0.9316 | | 0.9306 | | 0.9179 | | 0.8950 |
| SIDO 0 | 0.9430 | (4096) | **0.9432** | (1024) | | | | | 0.9426 |
| SIDO 1 | 0.7515 | (4932) | **0.7523** | (4096) | | | | | 0.7515 |
| SIDO 2 | 0.6184 | (4096) | **0.6235** | (2048) | | | | | 0.6143 |
| mean | 0.7710 | | **0.7730** | | | | | | 0.7695 |
| CINA 0 | 0.9706 | (132) | **0.9715** | (64) | 0.9712 | (128) | 0.9706 | (132) | 0.9706 |
| CINA 1 | 0.8355 | (128) | **0.8446** | (64) | 0.8416 | (128) | 0.8348 | (132) | 0.8348 |
| CINA 2 | 0.6108 | (64) | **0.8157** | (4) | **0.8157** | (4) | 0.8140 | (8) | 0.6095 |
| mean | 0.8057 | | **0.8773** | | 0.8761 | | 0.8732 | | 0.8050 |
| MARTI 0 | 0.9899 | (512) | **0.9914** | (256) | 0.9860 | (1024) | 0.9903 | (512) | 0.9860 |
| MARTI 1 | 0.8960 | (1024) | **0.9209** | (256) | 0.9134 | (32) | 0.8960 | (1024) | 0.8960 |
| MARTI 2 | 0.7571 | (4) | **0.7606** | (2) | **0.7606** | (2) | 0.7282 | (1024) | 0.7282 |
| mean | 0.8810 | | **0.8910** | | 0.8867 | | 0.8715 | | 0.8701 |

We applied the L2-loss SVM with RBF kernel on the list of features given by L2-loss linear SVM in order to clarify the performance in the case if feature rankings are combined with a nonlinear kernel. The results are shown in Table 7. Approach W still outperforms other methods when using a nonlinear SVM classifier. For these challenge data sets, W seems to be a good method regardless of the classifier used. Note that the Fnum values are not always the same in Tables 5 and 7, even though the same feature rankings are applied.

We also tried to incorporate Recursive Feature Elimination (RFE) (Guyon et al., 2002). For a given set of features, we use linear SVM weights to obtain the rankings, output ranks of those in the second half, and continue the same procedure on the first half features. To be more precise, subsets $S_j$ of size $|S_j| \in \{n, 2^{\lfloor \log n \rfloor}, \ldots, 2^i, \ldots, 2, 1\}$ are generated, where $n$ is the total number of features and $j = 0, \ldots, \lceil \log n \rceil$. After we train on subset $S_j$, we use the linear SVM weights to rank features in $S_j$ and let $S_{j+1}$ include the first half features. The results are not very different from that without RFE.

## 5. Discussion and Conclusions

In this challenge, we have experimented with several feature ranking methods. Among them, feature ranking based on F-score is independent from classifiers, feature ranking based on

Table 6: Comparison of different feature ranking methods using L1-loss linear SVM. It shows testing AUC and the corresponding Fnum. We did not run D-AUC and D-ACC on SIDO, so some slots in this table are blank.

| Dataset | Feature ranking methods: F-score | | W | | D-AUC | | D-ACC | | SVM LINEAR |
|---|---|---|---|---|---|---|---|---|---|
| REGED 0 | 0.9996 | (32) | **0.9997** | (16) | 0.9991 | (16) | 0.9981 | (256) | 0.9964 |
| REGED 1 | 0.9528 | (32) | **0.9558** | (64) | 0.9392 | (256) | 0.9551 | (64) | 0.9348 |
| REGED 2 | 0.8562 | (8) | 0.8419 | (8) | 0.8504 | (8) | **0.8777** | (16) | 0.7396 |
| mean | 0.9362 | | 0.9325 | | 0.9296 | | **0.9436** | | 0.8903 |
| SIDO 0 | 0.9407 | (4096) | **0.9419** | (512) | | | | | 0.9397 |
| SIDO 1 | 0.7588 | (4932) | **0.7590** | (4096) | | | | | 0.7588 |
| SIDO 2 | 0.6687 | (4932) | **0.6701** | (2048) | | | | | 0.6687 |
| mean | 0.7894 | | **0.7903** | | | | | | 0.7891 |
| CINA 0 | 0.9713 | (132) | 0.9713 | (132) | **0.9716** | (128) | 0.9713 | (132) | 0.9713 |
| CINA 1 | 0.8373 | (128) | 0.8369 | (132) | **0.8425** | (128) | 0.8369 | (132) | 0.8369 |
| CINA 2 | 0.6377 | (128) | 0.6377 | (128) | **0.8094** | (4) | 0.6347 | (132) | 0.6347 |
| mean | 0.8154 | | 0.8153 | | **0.8745** | | 0.8143 | | 0.8143 |
| MARTI 0 | 0.9872 | (512) | 0.9896 | (256) | **0.9933** | (512) | 0.9916 | (512) | 0.9858 |
| MARTI 1 | 0.8950 | (1024) | 0.9046 | (512) | **0.9168** | (512) | 0.9078 | (512) | 0.8950 |
| MARTI 2 | 0.7694 | (8) | **0.7790** | (4) | 0.7710 | (2) | 0.7369 | (8) | 0.7299 |
| mean | 0.8839 | | 0.8911 | | **0.8937** | | 0.8787 | | 0.8703 |

linear SVM weights require a linear SVM classifier, and the other two performance-based methods can use any classifier.

We focus on simple methods, so in this competition we can conduct quite complete validation procedures to select good models. However, although we have excellent performance on predictions, our methods do not provide information on the underlying causal relationships between features. Without causal discovery, the performance of our methods on manipulated data sets are not as good as that on unmanipulated data sets. Our methods might be improved by using causality, and how it can be done will need more investigations.

## Acknowledgments

This work was supported in part by grants from the National Science Council of Taiwan.

## References

Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.





Table 7: Comparison of different feature ranking methods using L1-loss SVM with RBF kernel as classifier. It shows testing AUC and the corresponding Fnum. We did not run D-AUC and D-ACC on SIDO, so some slots in this table are blank.

| Dataset | Feature ranking methods: F-score | | W | | D-AUC | | D-ACC | | SVM RBF |
|---|---|---|---|---|---|---|---|---|---|
| REGED 0 | **0.9997** | (64) | 0.9995 | (16) | **0.9997** | (16) | 0.9989 | (64) | 0.9968 |
| REGED 1 | 0.9709 | (32) | **0.9753** | (32) | 0.9748 | (16) | 0.9531 | (128) | 0.9419 |
| REGED 2 | **0.8881** | (8) | 0.8676 | (8) | 0.8676 | (8) | 0.8189 | (32) | 0.7459 |
| mean | **0.9529** | | 0.9475 | | 0.9474 | | 0.9236 | | 0.8949 |
| SIDO 0 | 0.9339 | (4096) | **0.9444** | (4096) | | | | | 0.9259 |
| SIDO 1 | 0.7339 | (4096) | **0.7634** | (4096) | | | | | 0.7124 |
| SIDO 2 | 0.5862 | (4096) | **0.6255** | (4096) | | | | | 0.5686 |
| mean | 0.7513 | | **0.7778** | | | | | | 0.7357 |
| CINA 0 | 0.9732 | (64) | **0.9754** | (32) | 0.9716 | (32) | 0.9718 | (128) | 0.9683 |
| CINA 1 | 0.8387 | (64) | **0.8646** | (32) | 0.8306 | (4) | 0.8383 | (128) | 0.8249 |
| CINA 2 | 0.6855 | (64) | **0.8358** | (4) | **0.8358** | (4) | 0.8164 | (8) | 0.6739 |
| mean | 0.8325 | | **0.8919** | | 0.8793 | | 0.8755 | | 0.8224 |
| MARTI 0 | 0.9883 | (512) | **0.9916** | (256) | 0.9848 | (1024) | 0.9896 | (512) | 0.9848 |
| MARTI 1 | 0.8877 | (1024) | **0.9181** | (256) | 0.9057 | (32) | 0.8877 | (1024) | 0.8877 |
| MARTI 2 | **0.7659** | (8) | 0.7616 | (16) | 0.7609 | (2) | 0.7308 | (1024) | 0.7308 |
| mean | 0.8806 | | **0.8904** | | 0.8838 | | 0.8694 | | 0.8678 |

Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Yi-Wei Chen and Chih-Jen Lin. Combining SVMs with various feature selection strategies. In Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti Zadeh, editors, *Feature extraction, foundations and applications*. Springer, 2006.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9: 1871–1874, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/liblinear.pdf`.

Isabelle Guyon, Constantin Aliferis, Greg Cooper, André Elisseeff, Jean-Philippe Pellet, Peter Spirtes, and Alexander Statnikov. Design and analysis of the causation and prediction challenge. *JMLR: Workshop and Conference Proceedings*, 2008.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.