https://doi.org/10.2352/ISSN.2470-1173.2017.12.IQSP-238
© 2017, Society for Imaging Science and Technology

Feature ranking and selection used in a machine learning framework for predicting uniformity of printed pages

Minh Q. Nguyen [a, b] and Jan P. Allebach [b]
[a] *School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906, U.S.A.*
[b] *Duos Technologies, Jacksonville, FL 32216, U.S.A.*

Abstract

In our previous work [1,2], we presented a block-based technique to analyze printed page uniformity both visually and metrically. In this paper, we introduce a new sets of tools for feature ranking and selection. The features learned from the models are then employed in a Support Vector Machine (SVM) framework to classify the pages into one of the two categories of acceptable and unacceptable quality. We utilize three methods in feature ranking including F-score, Linear-SVM weight, and Forward Search. The first two methods are filter methods while the last is categorized as a wrapper approach. We use the result from the wrapper method and information from the filter methods as confidence scores in our feature selection framework

Feature Selection

Feature selection [3, 4, 6] or feature ranking plays an important role in machine learning [5, 7]. The question always being asked is which features are actually relevant to classification. Does having as many features as possible improve the accuracy? It turns out that it doesn't. In fact, only a number of features are crucial to classification while the others may have a negative impact. The accuracy will start decreasing after a certain amount of features have been added. Not to mention more features means adding more computation cost and data acquisition. In addition, it increases the chance of overfitting. The selection of relevant features also means reducing dimensionality and facilitating data visualization, data understanding and interpretation.

There are three primary approaches [3, 9, 10] in feature selection: filter approach, wrapper approach, and embedded approach. Filter approach uses statistical method to filter out the low score features before classification. As a preprocessing method, it doesn't incorporate learning process [9] and it is finished before stepping into classification. Its drawback is ignoring feature dependencies. Wrapper methods [5, 11] combine feature selection and pattern classification in finding features which are evaluated based on classification results. The evaluation process uses accuracy as a metric to select the subset of features that achieve the least error rate. Wrapper methods typically involve cross-validation during its evaluation process. The last approach is embedded methods, which select features during the process of building the model. The process of learning and selecting features are incorporated. Embedded approach is less computationally expensive and less prone to overfitting than wrapper methods [4].

In this section, we review three methods in feature ranking including F-score, Linear-SVM weight, and Forward Search. The first two methods are filter methods while the third belongs to wrapper approach. We use the result from the wrapper method and information from the filter methods as confident scores in our feature selection framework.

F-score

F-score [12,13] is a simple method to compute the separation between two different classes. It belongs to the filter approach. The score is the ratio of between-class variance divided by within-class variance. The idea of F-score bears a great resemblance to the idea of the very well-known thesholding method: the Otsu's method [14]. Given two classes $(+)$ and $(-)$ with n_+ data points of the positive class and n_- data points of the negative class, F-score is computed as follows:

$$F(j) = \frac{\left(\bar{x}_j^{(+)} - \bar{x}_j\right)^2 + \left(\bar{x}_j^{(-)} - \bar{x}_j\right)^2}{\frac{1}{n_+ - 1}\sum_{k=1}^{n_+}\left(x_{k,j}^{(+)} - \bar{x}_j^{(+)}\right)^2 + \frac{1}{n_- - 1}\sum_{k=1}^{n_-}\left(x_{k,j}^{(-)} - \bar{x}_j^{(-)}\right)^2} \quad (1)$$

$$j = 1, 2, ..., P$$

where $\bar{x}_j$, $\bar{x}_j^{(+)}$, $\bar{x}_j^{(-)}$ denote the average of feature j of the entire data, positive data and negative data respectively; $x_{k,j}^{(+)}$ is the k^{th} positive data of the j^{th} feature , and $x_{k,j}^{(-)}$ is the k^{th} negative data of the j^{th} feature, P is the number of features.

From eqn. (1), the numerator denotes the variability between two classes while the denominator shows the within-class variability. In order words, F-score is the metric of separation between classes:

$$F(j) = \frac{\texttt{between-class variance of feature j}}{\texttt{within-class variance of feature j}} \quad (2)$$

As simple as its formulation may seem, F-score, however, contains a serious drawback when it comes to mutual information among features. In [15], the author used an example to show the case where F-scores of each single feature is low but the a combination of the two shows a great separation between the two classes. In this case, F-score fails to reveal the information among features. Figure 1 shows an example of two features whose F-scores are low. The features, however, are appropriate for classification since F-scores fail to reveal mutual information among the two features. It turns out that a feature with a low F-score doesn't necessarily mean an inappropriate feature for classification. This is an important characteristic that one needs to pay attention to when using F-score as a tool in feature ranking.

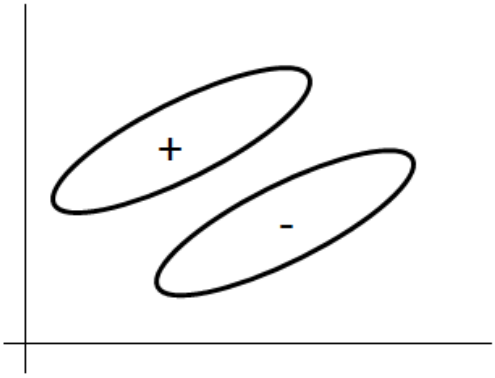


Figure 1. *An example of two low F-scores corresponding to two features [15]. In spite of the scores, the features are excellent for classification. F-scores, in this case, fail to reveal mutual information between features.*

Despite its drawback, F-score is still very widely used in practice due to its simplicity. F-score can be calculated in solely two loops. The inner loop contains the calculations of F-score for each feature, which include the means of the two classes, the mean of the entire data, and stddevs of the two classes. The outer loop is to run the entire features of our model. In a typical scenario, features with lowest scores will be dropped in an adaptive thresholding i.e. the selection of F-score thresholding is based on the error rate once the features have been dropped. The threshold results in the least error rate will be accepted. We use F-score as a confidence measure in comparison with other methods of feature ranking. F-score itself doesn't suffice in a feature selection framework.

We have 176 features in our entire data set of 347 sample pages [1, 2]. We first linearly scale each feature j separately to $[0 \div 1]$ using the the formula:

$$f_i^{scaled}(j) = \frac{f_i(j) - m_j}{M_j - m_j}, i = 1,2,....,n; j = 1,2,..,P \qquad (3)$$

where

$$M_j = max\{f_i(j)\}_{i=1}^{n} \quad \texttt{and} \quad m_j = min\{f_i(j)\}_{i=1}^{n}$$

n is the number of the entire data instances i.e. $n = n_+ + n_-$
P is the number of features.

The linear scaling serves two purposes. First, it helps make data compact and the visualization (if applicable) efficient. Second, it makes the selection of sigma in our Radial Basis Kernel or Gaussian kernel in SVM more efficient. After the linear scaling, we compute F-score for each feature using our training dataset in eqn. (1). We then normalize F-scores using the following equations:

$$F^{scaled}(j) = \frac{F(j) - m}{M - m}, j = 1,2,..,P \qquad (4)$$

where

$$M = max\{F(j)\}_{j=1}^{P} \quad \texttt{and} \quad m = min\{F(j)\}_{j=1}^{P}$$

P is the number of features.

The results of F-score and its normalization are shown in Fig. 2 and Fig. 3. The highest scores are listed in Table . Table 4 shows full meaning of feature expressions.

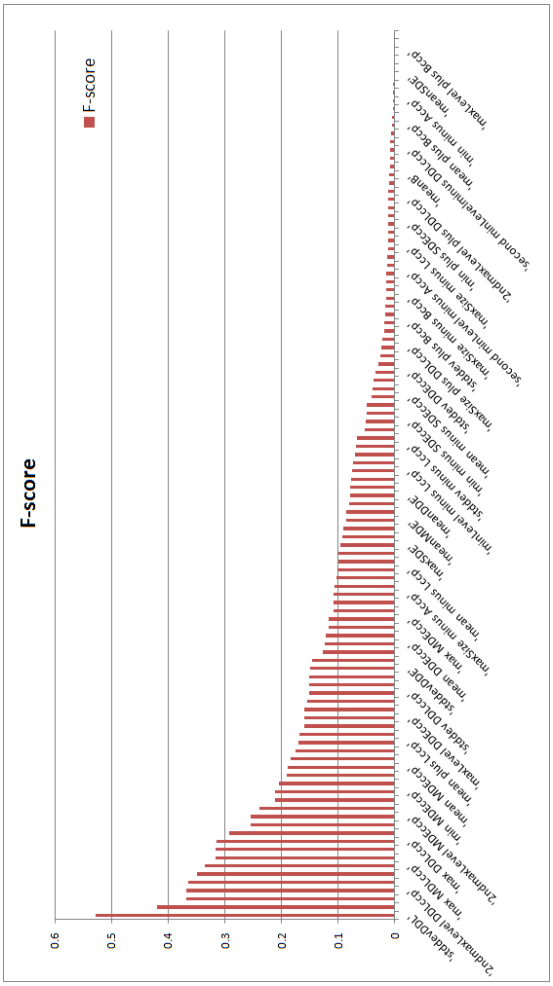


Figure 2. *F-score of 176 features.*

Linear Support Vector Machine Weight

Support Vector Machine (SVM) [16] is a popular method in machine learning. It is considered one of the best-known techniques in a binary classification [17]. SVM uses linear or non-linear kernels to construct linear or non-linear discriminant boundary. The idea of using SVM as a means to ranking features has been well investigated in different references [18, 19]. In order to find weights of each feature, a linear SVM is used to find the vector $\mathbf{w}$:

$$\mathbf{w} = (w_1, w_2, ..., w_P)$$

where P is number of features. The rank of each feature j is measured by its absolute value of w_j.

$$\text{Rank of feature j} = |w_j|$$

The intuition behind this method lies in the classification of SVM. In the solution of linear SVM, an instance $\mathbf{x}_i$, $i =$

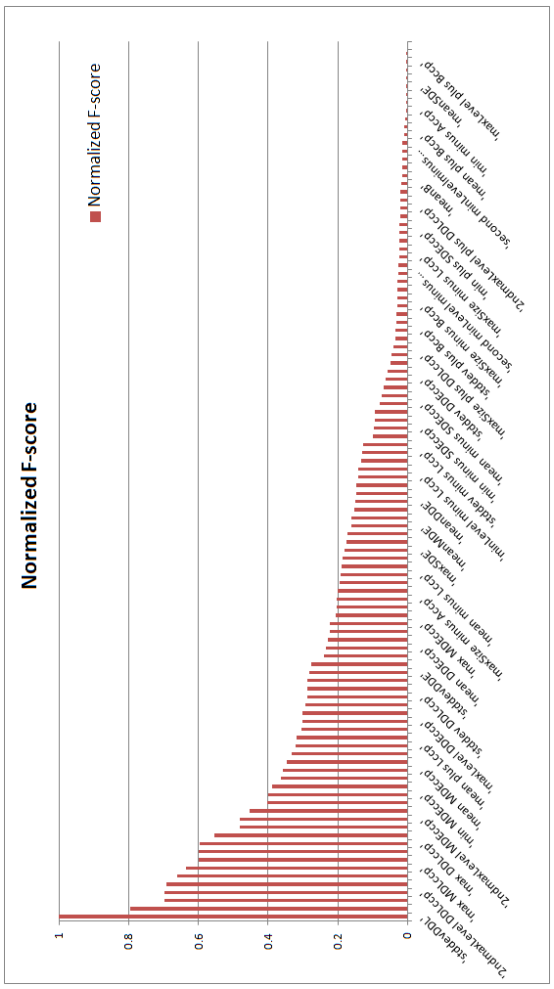


Figure 3. *Normalized F-score of 176 features.*

$\{1,2,...,n\}$ is classified according to the sign of the equation:

$$y(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x} + b \tag{5}$$

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{6}$$

$$y_i = \{+1,-1\} \text{ and b is bias}$$

$\mathbf{x}_i$'s with $\alpha_i \neq 0$ are support vectors. In fact, only support vectors contribute to creating decision boundary while the rest (with α_i = 0) do not. When y($\mathbf{x}$) > 0, $\mathbf{x}$ is classified to the (+) class. Otherwise, it belongs to the (-) class. It is noticed from eqn. (6) that only support vectors contribute to formulate $\mathbf{w}$. Each support vector is $\mathbb{R}^P$. Equation (5) apparently reveals $\mathbf{w}\in\mathbb{R}^P$ contributes to decision making. The features whose $|w_j|$ is small does not contribute significantly to the classification (eqn. 5); whereas the features with greater value of $|w_j|$ will play an important role in the final classification. In other words, features with greater $|w_j|$ are more dominant features and considered relevant features while those with lighter $|w_j|$ are irrelevant and can be removed in feature selection. This theory seems very intuitive [18] and can be used as a preprocessing method in feature ranking. In reality,

30 highest F-scores and normalized F-scores (nFscore) of all the features.

Fscore	nFscore	Feature
0.5281	1	'stddevDDL'
0.4193	0.7939	'stddevMDL'
0.3683	0.6974	'maxLevel DDLccp'
0.3683	0.6974	'2ndmaxLevel DDLccp'
0.3650	0.6911	'maxDDL'
0.3489	0.6606	'maxMDL'
0.3354	0.6351	'max MDLccp'
0.3159	0.5981	'maxLevel MDLccp'
0.3159	0.5981	'2ndmaxLevel MDLccp'
0.3144	0.5953	'max DDLccp'
0.2919	0.5527	'stddev MDLccp'
0.2541	0.4811	'maxLevel MDEccp'
0.2541	0.4811	'2ndmaxLevel MDEccp'
0.2389	0.4523	'mean DDLccp'
0.2118	0.4010	'mean MDLccp'
0.2107	0.3989	'min MDEccp'
0.2046	0.3874	'min DDLccp'
0.1911	0.3618	'maxSize DDLccp'
0.1891	0.3580	'mean MDEccp'
0.1832	0.3469	'maxSize MDLccp'
0.1749	0.3311	'min MDLccp'
0.1696	0.3211	'mean plus Lccp'
0.1685	0.3190	'meanDDL'
0.1604	0.3037	'max plus Lccp'
0.1595	0.3020	'maxLevel DDEccp'
0.1595	0.3020	'2ndmaxLevel DDEccp'
0.1543	0.2921	'meanMDL'
0.1518	0.2874	'stddev DDLccp'
0.1513	0.2864	'2ndmaxLevel plus Lccp'
0.1511	0.2861	'min plus Lccp'

features with $|w_j|$ greater than some threshold value are retained. The threshold is set by the sparsity criteria [18].

In many references [18–20], linear-SVM is used to rank features. Although the usage of non-linear SVM is doable, the result of the usage is questionable. For instance, from the current space of $\mathbf{x} = (x_1, x_2)$, a non-linear SVM is used to map to a higher dimensional space $\mathbf{x'} = (x_1, x_2, x_1 \times x_2, x_1^2 + x_2)$. The question is whether the weights of $x_1 \times x_2$ or $x_1^2 + x_2$ serve to acknowledge the weights of the original features i.e. x_1 and x_2. The answer is not. That's the reason why in most literature [18–20], a linear-SVM is widely utilized to find weights of features in filter approach.

In many any applications, an SVM Recursive Feature Elimination (SVM RFE) [21] is often discussed. This method works in the same manner of Linear-SVM weight except it is manipulated in many iterations. At the completion of each iteration, a feature is dropped if the value of its $|w_j|$ is the minimum among all the remaining features' $|w_j|$. The number of iterations (equivalent to the number of desired features) is specified by user. In fact, the number of features dropped in each iteration is also user-specified.

Figure 4 show the results of linear-SVM weights and its normalized values, in which C = 5 is chosen. In fact, a change in C will not impact $|w_j|$ too much. C is tested with different values but the results look quite similar. In [21], C = 100 is recommended. The corresponding values of the 30 highest weights are provided in Table . As a comparison with F-score ranking, there exist 13 common features among the 30 highest weights from the two methods of using F-score and linear-SVM weights. Apparently, the two methods yield different results with differ-

ent weights. However, a consistency of weights of a few features can be seen such as *stddevDDL, stddevMDL, maxMDL, maxDDL*. Obviously, some information from the two preprocessing methods (F-score and Linear-SVM weight) can be applicable in feature ranking and incorporated into the next method (Forward Search) which is discussed in the section to follow.

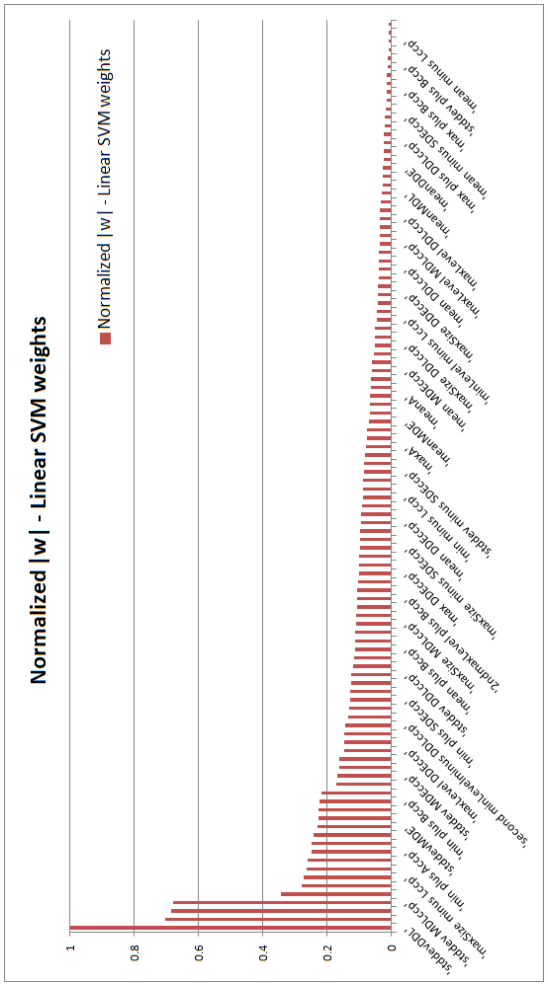


Figure 4. *Normalized linear-SVM weight.*

Forward Search

One of the core drawbacks of filter methods is the lack of taking feature correlation or feature dependency into account. One solution is to measure a group of features at a time through exhaustive search. This method, however, involves exhaustive computation due to the possible combinations of features i.e. 2^N combinations where N is the number of features. It is therefore barely employed in practice. A feasible solution is Forward Search and Backward Elimination which offer a good balance between computational cost and accuracy. The methods are also called consecutive search methods [22].

Forward search or Sequential Forward Search (SFS) [23] is a well-known method in feature selection. The feature subset of added features is initially empty. At each iteration, a single feature is added to the set. This feature is the one whose presence in the set will yield the most accuracy increase among all the yet-to-be-added features. The process is evaluated in a cross-validation. In [23], Mao incorporated Gram–Schmidt orthogonal transform into SFS and each feature can be then evaluated independently. One characteristic of Forward Search is that once a feature is added to the feature subset, it will not be removed later. This is actually a drawback of Forward Search since it makes it unable to correct any error it may have created in earlier iterations. Contrary to Forward Search, Backward Elimination starts with a full set of features [24]. At each stage, a single feature is dropped such that its disappearance will cause the least accuracy drop. Backward Elimination isn't feasible when the number of features is huge. In fact, the number of retained features isn't huge in practice. Plus, the evaluation through cross-validation is computationally expensive and inefficient in Backward Elimination when the number of features is large. Zhang used a method called "Adaptive Forward-Backward Greedy Algorithm" in [25] to fix the problem. At every step of forward search, a backward elimination is conducted to remove the least relevant features in the feature subset. By doing backward elimination, error of adding irrelevant features in previous steps can be excluded during the forward search process. In spite of its error correction functionality, this method is computationally expensive due to the backward process when the number of added features becomes bigger. In this paper, we conduct Forward Search in a number of times with different randomized order of the data. We will then inspect those features with high frequency of appearance for feature ranking.

30 highest linear-SVM weights (LSVMW) and its normalized weights (nLSVMW) of all the features.

LSVMW	nLSVMW	Feature
2.1885	1	'stddevDDL'
1.5412	0.7042	'stddevMDL'
1.4996	0.6852	'stddevDDE'
1.4831	0.6776	'stddev MDLccp'
0.7562	0.3455	'maxMDL'
0.6124	0.2798	'maxLevel plus Bccp'
0.5991	0.2737	'maxSize minus Lccp'
0.5801	0.2650	'maxDDL'
0.5734	0.2620	'2ndmaxLevel plus Lccp'
0.5443	0.2487	'min plus Accp'
0.5422	0.2477	'maxSize plus Lccp'
0.5305	0.2424	'max MDLccp'
0.5045	0.2305	'stddevMDE'
0.4981	0.2275	'second minLevel minus Lccp'
0.4977	0.2274	'min DDLccp'
0.4917	0.2246	'min plus Bccp'
0.4763	0.2176	'min MDEccp'
0.3751	0.1713	'min plus Lccp'
0.3668	0.1676	'stddev MDEccp'
0.3566	0.1629	'stddevA'
0.3547	0.1620	'maxLevel plus Lccp'
0.3213	0.1468	'maxLevel DDEccp'
0.3213	0.1468	'2ndmaxLevel DDEccp'
0.3200	0.1462	'min DDEccp'
0.3190	0.1457	'second minLevelminus DDLccp'
0.2930	0.1338	'maxDDE'
0.2916	0.1332	'mean MDLccp'
0.2830	0.1293	'min plus SDEccp'
0.2808	0.1283	'min minus Bccp'
0.2740	0.1251	'max minus Lccp'

SVM Forward Search Algorithm for Feature Selection

1. Model Selection
   C (in non-linearly separable SVM) and σ (in RBF kernel) need to be assigned specific values before conducting Forward Search. C and σ are chosen in an exhaustive search in a RBF-SVM with a 5-fold cross validation utilizing full set of features. Again, our data consists of 347 pages each of which has 176 features. Each test page has been graded by a print quality expert (golden eye) and is classified into one of four categories: A (good PQ – pass), B (fairly good PQ – pass), C (bad PQ – fail), and D (very bad PQ – fail). This is our ground truth. In fact, we only take pass and fail into consideration which means a binary classification. We randomize data, and split it into training and testing set in a 5-fold cross validation. We use 4 folds for training and the remaining fold for testing. Selected C and σ are the ones that yield the best average accuracy (in a 5-fold cross-validation) though exhaustive search of C = $[0.1, 0.2, ..., 1, 2, ..., 10, 20, ..., 100, 200, ..., 10,000]$ and σ = $[.01, .02, ..., 1]$. Due to linear data scaling to $[0 \div 1]$, restricting the search for σ in $[.01, .02, ..., 1]$ is rational. A refined search for C is also conducted in the proximity of the newly found C. When C and σ are selected, their values remain unchanged during feature selection. Figure 5 shows the result of our analysis. C = 0.2 and σ = .83 are chosen at the best accuracy of 77%.
2. Repeat 7 times with different random generators
   - Split data into training and testing set into 5 folds. Use 4 folds for training and the remaining fold for testing
   - Initially set feature subset S = $\emptyset$
   - Add a feature to S such that the average accuracy in a 5-fold cross-validation is the maximum in a SVM classification (best average accuracy)
   - Stop when accuracy doesn't improve or start falling or when the number of features approaches a predefined cutoff
3. Find the mean and stddev of accuracy according to the number of added features
4. Rank features according to their frequency and median order of occurrences in Forward Search. For instance, feature i appears 7 out of 7 times and its orders are $A = \{1, 1, 4, 2, 3, 4, 4\}$. It will be placed in the top positions of most frequently appeared features (7 out of 7) and then ranked based upon its relative order of appearances (= median(A) = 3) when it is compared with features of the same number of occurrences. Select features until accuracy starts dropping

Figure 6(a) shows the performance of Forward Search in 7 trials. As can be seen, the accuracy attains its maximum around .85 when the number of added features in approximately 20. Afterwards, the accuracy starts declining gradually. Figure 6(b) shows the mean and mean plus/minus stddev of all the curves in Fig. 6(a). Apparently, the two figures provide information about the number of features that is sufficient to achieve the best result in classification.

We next need information about the highest ranked features. Using results of 7 trials of Forward Search, we then:

1. record the order of each feature's occurrence in the feature subset
2. count the number of occurrence in 7 trials
3. find the median order of all occurrences each feature
4. rank feature according to their number of occurrence (first priority) and then their median order of all occurrences (second priority)

Figure 7 and 8 demonstrate the results of our analysis in raw data and normalized data (using eqn. (3)) in Forward Search. Information about F-score and Linear-SVM weights is also incorporated. Horizontal axes are features ranked according to their number of occurrence (first priority) and then their median order of all occurrences (second priority) (from left to right: highest to lowest rank). As can be seen, features with highest F-scores and Linear-SVM weights appear in the first orders. Also, a few features with low scores still occur i.e. emphasizing the drawback of the filter methods (F-score and Linear-SVM weight). Table provides the details of all the data of Fig. 7 and Fig. 8. It is noticeable that feature #4 have the best scores of F-score and Linear-SVM weight but it is ranked 4th in Forward Search, while feature #1 has second best in F-score and Linear-SVM weight. Undoubtedly, these two features are among the most important features in the feature set. Highly ranked features will be employed in specific application [8].

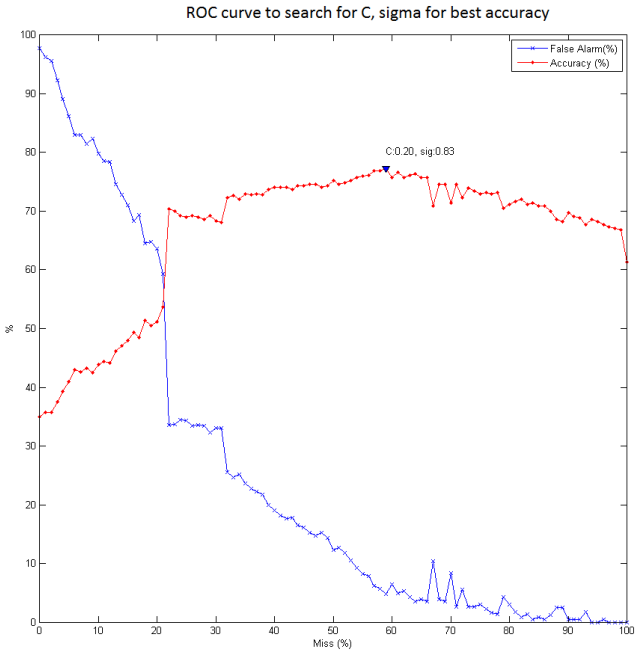


Figure 5. *Receiver Operating Characteristic (ROC) curve to search for C, σ in RBF-SVM for best accuracy using full set of features. The values of C, σ are then kept unchanged during Forward Search.*

Conclusion

This paper discusses feature ranking and selection in different methods. In filter methods, we explained how to use F-score and linear-SVM weight. Though filter methods are easy to implement, their main drawback is the inability to reveal mutual information among features. The other mentioned method is wrapper

27 highest ranked features according to their number of occurrence and their median order of occurrence in 7 trials in Forward Search

#	Feature	#Occur	Fscore	LSVMW	nFscore	nLSVMW
1	'stddevMDL'	7/7	0.4193	1.5412	0.7939	0.7042
2	'maxLevel MDLccp'	7/7	0.3159	0.0823	0.5981	0.0376
3	'stddev MDLccp'	7/7	0.2919	1.4831	0.5527	0.6776
4	'stddevDDL'	7/7	0.5281	2.1885	1	1
5	'meanL'	7/7	0.0261	0.1406	0.0494	0.0642
6	'maxL'	7/7	0.0114	0.0749	0.0215	0.0342
7	'maxMDE'	7/7	0.1263	0.0358	0.2391	0.0163
8	'stddev minus Lccp'	7/7	0.0689	0.2639	0.1304	0.1205
9	'stddev plus Accp'	7/7	0.0157	0.0852	0.0297	0.0389
10	'meanMDE'	7/7	0.0856	0.1571	0.1620	0.0717
11	'stddevMDE'	7/7	0.1081	0.5045	0.2046	0.2305
12	'meanDDE'	7/7	0.0792	0.0589	0.1499	0.0269
13	'max MDEccp'	7/7	0.1169	0.1143	0.2213	0.0522
14	'stddev minus SDEccp'	6/7	0.0499	0.1887	0.0944	0.0862
15	'max plus Accp'	6/7	0.0160	0.2191	0.0302	0.1001
16	'min minus Lccp'	6/7	0.1000	0.1956	0.1893	0.0893
17	'maxDDE'	6/7	0.1172	0.2930	0.2219	0.1338
18	'maxSize plus DDLccp'	5/7	0.0238	0.1317	0.0450	0.0601
19	'stddev minus Bccp'	5/7	0.0367	0.1991	0.0694	0.0909
20	'stddev MDEccp'	5/7	0.0666	0.3668	0.1261	0.1676
21	'mean MDLccp'	4/7	0.2118	0.2916	0.4010	0.1332
22	'meanMDL'	4/7	0.1543	0.0687	0.2921	0.0313
23	'max DDEccp'	4/7	0.1077	0.2210	0.2039	0.1009
24	'min MDLccp'	4/7	0.1749	0.2341	0.3311	0.1069
25	'min minus SDEccp'	4/7	0.0515	0.1933	0.0975	0.0883
26	'maxSize plus Bccp'	4/7	0.0488	0.1666	0.0924	0.0761
27	'stddevDDE'	4/7	0.1488	1.4996	0.2817	0.6852

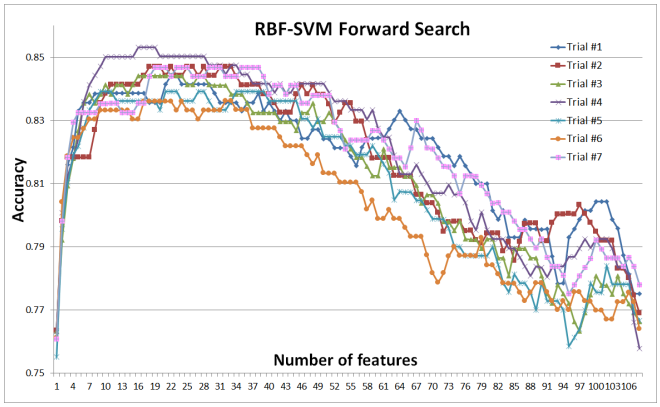
Abbreviations of features in Figures. 2, 3, 4, 7, 8 and in the tables

Expression	Meaning
stddev	standard deviation
ccp	connected component
max	maximum value
min	minimum value
maxSize	maximum size in connected component map
maxLevel	highest level in connected component map
2ndmaxLevel	second highest level in connected component map
plus	positive part in uniformity maps
minus	negative part in uniformity maps
n	normalized
nFscore	normalized F-score
LSVMW	Linear Support Vector Machine Weight
nLSVMW	normalized Linear Support Vector Machine Weight

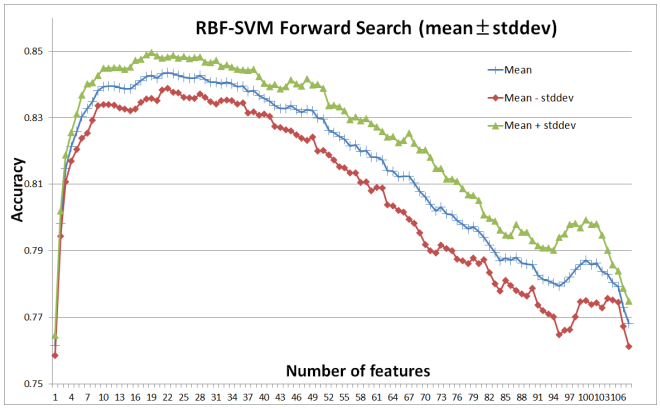
method, which is shown to make up for this drawback. Wrapper methods combine feature selection and pattern classification in finding features which are evaluated based on classification results. The evaluation process uses accuracy as a metric to select the subset of features with the least error rate. One of the wrapper methods used to rank features is Forward Search. We explained how Forward Search worked and implemented the method in 7 trials with different randomized orders of the data set. In each trial, we used 5-fold cross validation in each step of adding a new feature to the feature subset. The added feature is the one that yields the highest average accuracy in a 5-fold cross validation. We repeated the process 7 times with 7 different randomized orders. Due to the different results of subsets (in 7 trials), we ranked features according to their number of occurrence and their median order in 7 trials. Finally, we demonstrated our result of Forward Search incorporated with features' F-scores and linear-SVM weights. Highly ranked features are then used in the construction of a significantly better Receiver Operating Characteristic (ROC) curves when compared with using all features [8] in our application.

References

[1] Nguyen, Minh Q, Jessome, Renee, Astling, Steve, Maggard, Eric, Nelson, Terry, Shaw, Mark, Allebach, Jan P, "Perceptual metrics and visualization tools for evaluation of page uniformity", Proc. IS&T/SPIE Electronic Imaging, 9016, doi:10.1117/12.2038752,



(a)



(b)

Figure 6. *(a) RBF-SVM Forward Search in 7 trials of different randomized orders of data; b) RBF-SVM Forward Search (mean ± sdttdev) from the 7 trials.*

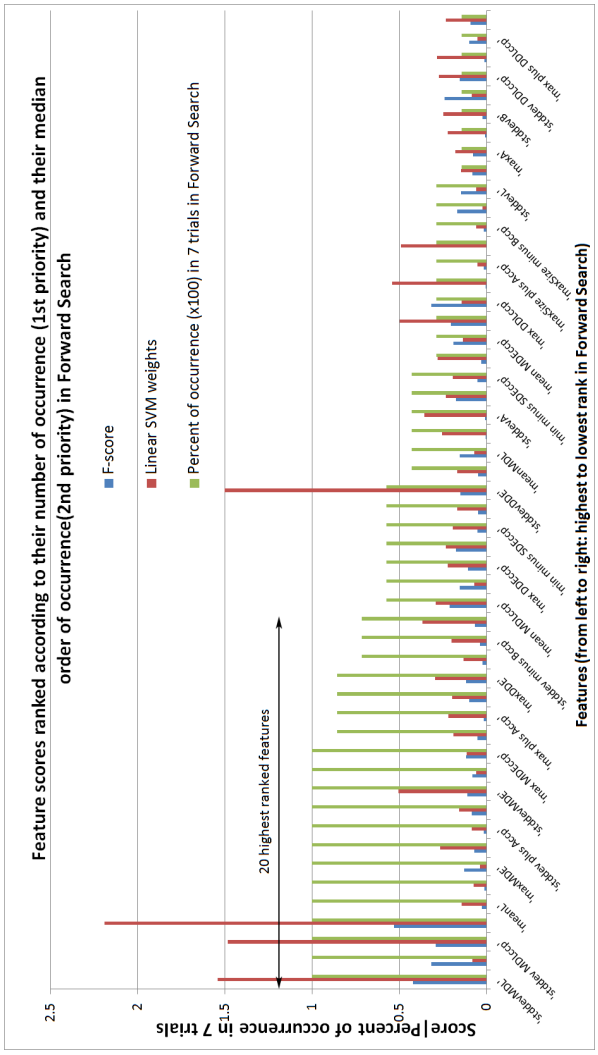


Figure 7. *Feature Ranking according to their number of occurrence and their median order in 7 trials in Forward Search. Information of F-score and Linear-SVM weight is also incorporated.*

901608-901608-13, 2014

[2] Nguyen, Minh Q, Allebach, Jan P, "Controlling misses and false alarms in a machine learning framework for predicting uniformity of printed pages", Proc. IS&*T*/SPIE Electronic Imaging, 9396, doi:10.1117/12.2083162, 93960I-93960I-12, 2015

[3] Avrim L. Blum and Pat Langley. 1997. Selection of relevant features and examples in machine learning. Artif. Intell. 97, 1-2 (December 1997), 245-271. DOI=http://dx.doi.org/10.1016/S0004-3702(97)00063-5

[4] Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. J. Mach. Learn. Res. 3 (March 2003), 1157-1182

[5] Karegowda, Asha Gowda, M. A. Jayaram, and A. S. Manjunath. "Feature subset selection problem using wrapper approach in supervised learning." International journal of Computer applications 1.7 (2010): 13-17

[6] González, Antonio, and Raúl Pérez. "Selection of relevant features in a fuzzy genetic learning algorithm." IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 31.3 (2001): 417-425.

[7] Carbonell, Jaime G., Ryszard S. Michalski, and Tom M. Mitchell. "An overview of machine learning." Machine learning. Springer Berlin Heidelberg, 1983. 3-23

[8] Nguyen, Minh Q, "Perceptual metrics, visualization tools, and machine-learning-based quality prediction for evaluation of page uniformity", Dissertation/thesis number 10106211, ISBN 9781339696614, Purdue University, 2015

[9] Lal, Thomas Navin, et al. "Embedded methods." Feature extraction. Springer Berlin Heidelberg, 2006. 137-165

[10] Maldonado, Sebastián, and Richard Weber. "A wrapper method for feature selection using support vector machines." Information Sciences 179.13 (2009): 2208-2217

[11] Wang, Hongzhi, et al. "A learning-based wrapper method to correct systematic errors in automatic image segmentation: consistently improved performance in hippocampus, cortex and brain segmentation." NeuroImage 55.3 (2011): 968-985

[12] Jaganathan, P., N. Rajkumar, and R. Kuppuchamy. "A comparative study of improved F-score with support vector machine and RBF network for breast cancer classification." International Journal of Machine Learning and Computing 2.6 (2012): 741

[13] Polat, Kemal, and Salih Güneş. "A new feature selection method on classification of medical datasets: Kernel F-score feature selection." Expert Systems with Applications 36.7 (2009): 10367-10373

[14] Otsu, Nobuyuki. "A threshold selection method from gray-level his-

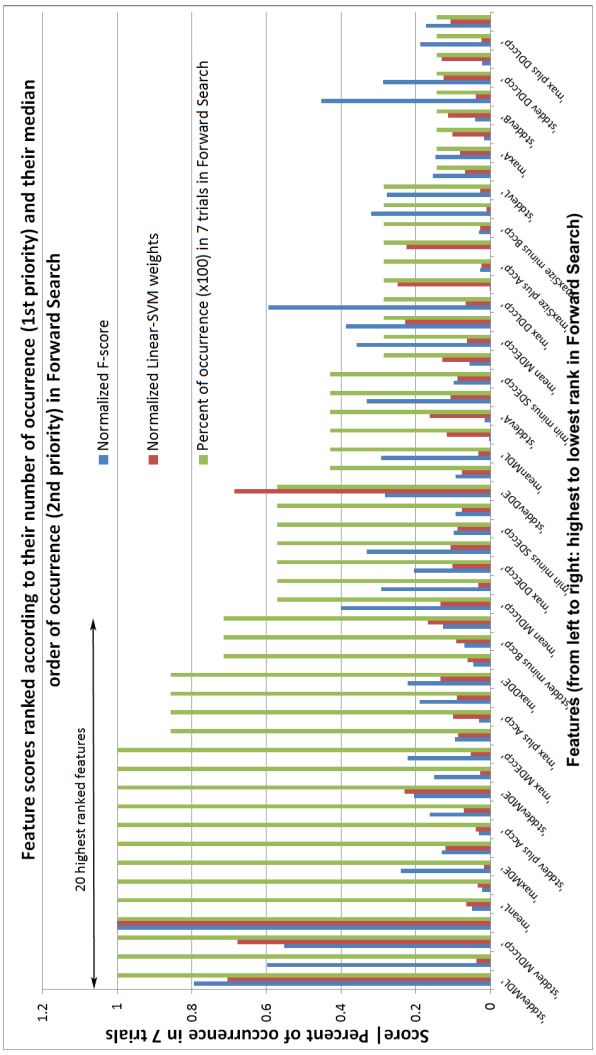


Figure 8. *Feature Ranking according to their number of occurrence and their median order in 7 trials in Forward Search (Normalized version of Fig. 7). Information of F-score and Linear-SVM weight is also incorporated.*

tograms." Automatica 11.285-296 (1975): 23-27

[15] Chen, Yi-Wei, and Chih-Jen Lin. "Combining SVMs with various feature selection strategies." Feature extraction. Springer Berlin Heidelberg, 2006. 315-324

[16] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297

[17] Cristianini, Nello, and John Shawe-Taylor. An introduction to support vector machines and other kernel-based learning methods. Cambridge university press, 2000

[18] Brank, Janez, et al. "Feature selection using linear support vector machines." Proceedings of the 3rd International Conference on Data Mining Methods and Databases for Engineering. 2002

[19] Chang, Yin-Wen, and Chih-Jen Lin. "Feature ranking using linear SVM." WCCI Causation and Prediction Challenge. 2008

[20] Mladenić, Dunja, et al. "Feature selection using linear classifier weights: interaction with classification models." Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2004

[21] Guyon, Isabelle, et al. "Gene selection for cancer classification using support vector machines." Machine learning 46.1-3 (2002): 389-422

[22] Kung, Sun Yuan. Kernel methods and machine learning. Cambridge University Press, 2014

[23] Mao, K. Z. "Orthogonal forward selection and backward elimination algorithms for feature subset selection." IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 34.1 (2004): 629-634

[24] Hall, Mark A., and Lloyd A. Smith. "Feature subset selection: a correlation based filter approach." (1997)

[25] Zhang, Tong. "Adaptive forward-backward greedy algorithm for learning sparse representations." IEEE transactions on information theory 57.7 (2011): 4689-4708

Author Biography

Minh Q. Nguyen *received his B.S. in Electrical Engineering at the University of Technology, Ho Chi Minh City, Vietnam, his M.S., Ph.D. in Electrical and Computer Engineering at Purdue University, USA in 2009, 2011, and 2015 respectively. He worked under the guidance of Prof. Jan P. Allebach during his time at Purdue. After graduation, Minh joined Duos Technologies as a Computer Vision software engineer with a focus on rail road inspection imaging systems. His research interests include image processing, computer vision, image quality, color science, and machine learning.*

Jan P. Allebach *is Hewlett-Packard Distinguished Professor of Electrical and Computer Engineering at Purdue University. Allebach is a Fellow of the IEEE, the National Academy of Inventors, the Society for Imaging Science and Technology (IS&T), and SPIE. He was named Electronic Imaging Scientist of the Year by IS&T and SPIE, and was named Honorary Member of IS&T, the highest award that IS&T bestows. He has received the IEEE Daniel E. Noble Award, and is a member of the National Academy of Engineering. He currently serves as an IEEE Signal Processing Society Distinguished Lecturer (2016-2017).*