Zhao and Guan *Big Data Analytics* (2017) 2:10
DOI 10.1186/s41044-017-0025-5

Big Data Analytics

RESEARCH Open Access

A subspace recursive and selective feature transformation method for classification tasks

Xuan Zhao* and Steven Sheng-Uei Guan

*Correspondence:
xuan.zhao@xjtlu.edu.cn
Department of Computer Science and Software Engineering, Xi'an Jiaotong-Liverpool University, 111 Ren'ai Road, 215123 Suzhou, China

Abstract

Background: Practitioners and researchers often found the intrinsic representations of high-dimensional problems has much fewer independent variables. However such intrinsic structure may not be easily discovered due to noises and other factors. A supervised transformation scheme RST is proposed to transform features into lower dimensional spaces for classification tasks. The proposed algorithm recursively and selectively transforms the features guided by the output variables.

Results: We compared the classification performance of linear classifier and random forest classifier on the original data sets, data sets being transformed with RST and data sets being transformed by principle component analysis and linear discriminant analysis. On 7 out 8 data sets RST shows superior classification performance with linear classifiers but less ideal with random forest classifiers.

Conclusions: Our test shows the proposed method's capability to reduce features dimensions in general classification tasks and preserve useful information using linear transformations. Some limitations of this method are also pointed out.

Keywords: Machine learning, Feature transformation, Feature selection, Classification, Subspace learning

Background

In machine learning tasks the intrinsic representations of high-dimensional data may have much fewer independent variables, as suggested by Hastie [1] in hand written recognitions, the motion of objects [2], and array signal processing [3]. Most of methods trying to solve this problem is domain-specific, like in image processing, the learning of representation often relies on the locality and smoothness assumptions [4]. We are interested in a generally applicable transformation method to transform feature into lower dimensions while preserving useful information for classification tasks.

The proposed algorithm reduces the dataset dimensionality by selectively projecting data points on to the decision plane determined by fitted linear discriminative model. The algorithm is able to run recursively to make better projections.

Notation

Bold lower-case letters denote vectors, e.g. $\boldsymbol{\nu}$, and capital letters for matrices, e.g. M. A vector $\boldsymbol{\nu}$'s $L1$ and $L2$ norm are denoted by $\|\boldsymbol{\nu}\|$ and $\|\boldsymbol{\nu}\|_2$ respectively. The ith row and jth

column of a matrix M are denoted by $\boldsymbol{m}(i)$ and $\boldsymbol{m}_j$, respectively. We use *sample* and *data point* interchangeably to refer to an observation in the data set.

Classification tasks

Given a dataset $X \in \boldsymbol{R}^{m\times n}$ (m samples and n features) and its corresponding class $\boldsymbol{y} \in \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_c\}^m$. The classes are $\boldsymbol{c}$; $\boldsymbol{y}_i$ is sample $X(i)$'s ground truth class. Let $\boldsymbol{x}$ be a row/sample in X, θ be the model parameters.

A classification task is to construct classifier $h(\theta) : \boldsymbol{x} \mapsto \boldsymbol{y}$ from the seen examples $(X, \boldsymbol{y})$ such that for an unseen set of examples X_{pred}, $h(X_{pred}, \theta)$ will be as close as possible to $\boldsymbol{y}_{pred}$. The modelling process involves minimizing the empirical error between $\boldsymbol{y}$ and $h(X, \theta)$, denoted as $\hat{E}(\boldsymbol{y}, h(X, \theta))$. To avoid h over-fits $(X, \boldsymbol{y})$, the complexity of h is penalized in terms of its some kind of norm $||h(\cdot)||_p$. Therefore a classifier can be trained by solving:

$$argmin_\theta \, \hat{E}\left(\boldsymbol{y}, h(X, \theta)\right) + \|h(X, \theta)\|_p \tag{1}$$

Support vector machine

Support vector machines (SVM) is a classic classification algorithm. In the case of two-class classification task ($\boldsymbol{y} \in \{c_1, c_2\}^m$), it minimizes $\hat{E}(\boldsymbol{y}, h(X, \theta))$ by attempting to place a hyperplane, $f(X) = \boldsymbol{w}X + w_0$, between data points from class c_1 and c_2. The classifier $h(X_{pred})$ returns a vector of positive or negative signs to indicate the labels of X_{pred}.

$$h(X) = sign\left(\boldsymbol{w}X + w_0\right) \tag{2}$$

The hyperplane, often known as decision plane, is then optimized to maximize it's minimal distance to data points from c_1 and c_2 (also known as the margin). Vapnik showed that $\hat{E}(y, h(X))$ can be minimized by maximizing the margin [5]. A penalizing term C is used to control the magnitude of the decision plane misplacing a particular sample $x_i \in X$ on the other side. The model complexity can be penalized by minimizing $\boldsymbol{w}$'s L_1 norm $||\boldsymbol{w}||$ or L_2 norm $||\boldsymbol{w}||_2$ [6]

$$argmin_{\boldsymbol{w},w_0} \, \frac{\|\boldsymbol{w}\|_p}{2} \boldsymbol{w} \cdot X + w_0 = 1 + C \sum_1^m \xi_i \tag{3}$$

In the proposed algorithm $\boldsymbol{w}$ are used to indicate the feature's contribution to a discriminative model. Since $\boldsymbol{w}$ defines the direction of the hyperplane separating two classes of data points in the n dimensional space, a $w_i \in \boldsymbol{w}(i = 1, \ldots, n)$ closing to zero indicates the plane is nearly orthogonal to the i^{th} axis. This intuition has been used in Recursive Feature Elimination [7] method. Figure 1 is a plot of such weights .

Recursive feature elimination

Recursive Feature Elimination (RFE) is a supervised feature ranking and selection technique. It use a classifier's feature-related weights as the feature importance metrics, such $\boldsymbol{w}$ in SVMs or the coefficients in Fischer's linear discriminator [7, 8].

With a desired numbers of features fixed and number of features s to be removed in each step, RFE starts by training a classifier on all training samples. Recursively it eliminates features with the s lowest importance until the desired numbers of features has been reached.

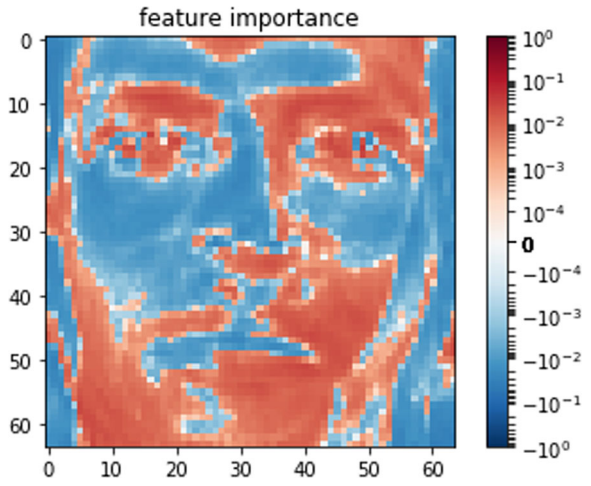

Fig. 1 Weights from the first label of **faces** data set Darker color indicates higher importance. Positive importances is the magnitude of feature's contribution in classifying this label against the rest, negative is for the rest against this label

Principle component analysis

Principle Component Analysis (PCA) is a feature transformation method that finds a sets of orthogonal components that explain the maximum variance out of the samples. It can also be used to reduce dimensionality of the samples by projecting them to a lower dimensional space (principle component) [9].

PCA algorithm starts with subtracting the mean of X for all $x_i \in X$. Then it computes a covariance matrix $\frac{Cov(X,X)}{m}$ by singular value decomposition. Afterwards it finds the eigenvectors with $k(k < m)$ greatest eigen value as the orthogonal components. For a transformation matrix A of size $m \times k$ using k eigenvectors. The projection of X to a k-dimensional space is XA.

Feature extraction with SVM

Tajiri et al. proposed a method based for feature extraction using the weight coefficients of a learnt linear SVM classifier for binary classification problems [10]. The intuition is that assume the decision boundary of a linear SVM perfectly discriminates both classes, then the orthogonal hyper plane of the decision boundary, which is determined by the weight coefficients, is an ideal projection plane for the data points to be projected on to embed the discriminative information into the transformed data points.

Recursive Selective Feature Transformation (RST) for classification tasks

Feature importance

$\boldsymbol{w}$ from the linear SVM model can be seen as how much contribution does each of feature make to form the decision plane.

From a geometric perspective, if $\boldsymbol{w}_i = 0$, the axis of dimension i is then parallel to the decision plane. Such situation indicates that the decision plane does not split the samples into two classes in dimension i; in other words, with feature i the SVM model cannot discriminate the samples.

If $\boldsymbol{w}_i = -1$ or 1, the axis of dimension i is then orthogonal to the decision plane, which indicates the only by feature i can the decision plane discriminate the samples. In most cases $\boldsymbol{w}_i$ does not reach -1 and 1.

Feature importance vector v

In a binary classification setting where $|\boldsymbol{c}| = 2$, our feature importance vector $\boldsymbol{v} \in [0, 1]^n$ takes the absolute values of weight vector $\boldsymbol{w} \in [-1, 1]^n$.

In a multi-class setting $|\boldsymbol{c}| > 2$, a k-class classification problem is solved by One-versus-Rest [11] scheme, which is an ensemble of k classifiers with each of them trained by discriminating training data from $c_i \in \boldsymbol{c}$ against the rest $(\boldsymbol{c}/c_i)$. In a k class setting there are k $\boldsymbol{w}$s, denoted as $W \in [-1, 1]^{k \times n}$ from all sub classifiers in the ensemble, then $\boldsymbol{v}$ is taken as the mean vector of absolute values of W.

$$\boldsymbol{v} = \begin{cases} abs(\boldsymbol{w}), & \text{if } |\boldsymbol{c}| = 2 \\ \frac{1}{k} \sum_1^k abs(W(i)), & \text{if } |\boldsymbol{c}| = k > 2 \end{cases} \tag{4}$$

Recursive Selective Feature Transformation (RST)

Consider the hypothetical dataset X has m rows of examples and n columns of features.

We adopted an improved version of the SVM feature extraction method by Tajiri et al. [10]. In short, Tajiri, et al.'s method projects data points on the linear decision plane's orthogonal plane. Naturally projecting the data points to a hyper-plane in the feature space reduces dimensionality by 1, in other words the projected data points have dimensionality of $n - 1$, thus this approach maintains data points' inter-class separation while reduces dimensionality.

Therefore reducing the dimensionality to a much smaller number, say k, requires data points being projected $n - k$ times, which involves training SVM model $n - k$ times. Since the approximated time complexity of training a linear SVM model is $O(max(m, n)min(m, n)^2)$ [12], assume data sets normally have much more rows than columns, that is $m >> n$, the time complexity of Tajiri, et al.'s method in this scenario is around $O(m(n - k)n^2)$, which is expensive to compute for high dimensional data.

To speed up the dimensionality reduction process, RST uses SVM's weight vector to selected top-k *important* features from the *feature importance vector* (Eq. 4) to reduce the weight vector to length k, and project data points onto a k-dimensional feature subspace determined by the reduced weight vector. Thus k can be used as a parameter to determine the desired dimensionality. (RST Step 1)

Projecting data points using weights corresponding to those *less important* features is not ideal since these features contributed less to form current decision plane. They may represent redundant or other non-informative information and therefore can normally be discarded as seen in RFE. A linear decision plane may not generalise on those less important features, therefore RST performs dimensionality reduction on these features with RST Step 1. (RST Step 2).

To further improve the transformation quality while reduce the dimensionality, RST Step 1 and RST Step 2 will run recursively. The time complexity of a combined step of RST step 1 and 2 is around $O(mn^2)$. In implementation we substitute linear SVM [13] for logit model (l2-regularised, stochastic gradient descent solver with hinge loss) on problems with over 10,000 samples to cope with the potential cache issue of SVM.

Furthermore, multi-class problem can be difficult for feature extraction based on linear SVM models. The usual approach dealing with multi-class problem is One-versus-Rest (OvR) ensemble [11]. Through this approach, a k class problem leads to k decision functions, i.e., k weight vectors and k corresponding intercepts, each of them represents the model for each "one class versus the rest of classes" problems. Obviously averaging the weight vectors, as in the linear models, brings little benefits in terms of generalisation. In RST the i-th ($i \in [1, k]$) projection matrix is learnt from the i-th weight vectors in the ensemble. Thus upon training a classifier, the i-th sub-classifier in the OvR ensemble learns on data set that has been transformed by the i-th projection matrix, see Fig. 2 for illustration. Upon predicting an unknown sample, RST transforms the sample will be by the k projection matrices into k transformed samples for the ensemble to predict.

Methods

Evaluation procedures

The experiments runs for 32 iterations. At each iteration the data set is randomly split with train-test ratio of 3:2. Transformations are learnt by RST, linear discriminant analysis (LDA) [14] and PCA on the training set, then both of the training and testing set are transformed with the learnt transformer. RST runs for 6 recursions. Transformation learnt from each recursion were benchmarked. For comparison the number of output features from PCA and LDA is set the same as the number of dimensions of each transformed data set by RST. Due to the nature of LDA, the number of features after dimensionality reduction is strictly less than the number of classes.

The values of data sets has been scaled to [0, 1] without any centering, no missing values or outliers (within +-1.5 IQR) present. Random forest classier and Linear SVM classifier with L_2 norm penalty and hinge loss is used as the benchmark classifier.

We evaluate the classification performance via multi-class logarithmic loss [15]:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(p_{i,c}) \tag{5}$$

where N is the number of samples being tested, C is the number of classes, *log* is the natural logarithm, $y_{i,c}$ is 1 if observation i is in class c and 0 otherwise, and $p_{i,c}$ is the predicted probability that sample i is in class c. This metric takes into account the uncertainty of the classifier's prediction according to how much it varies from the actual class label. That

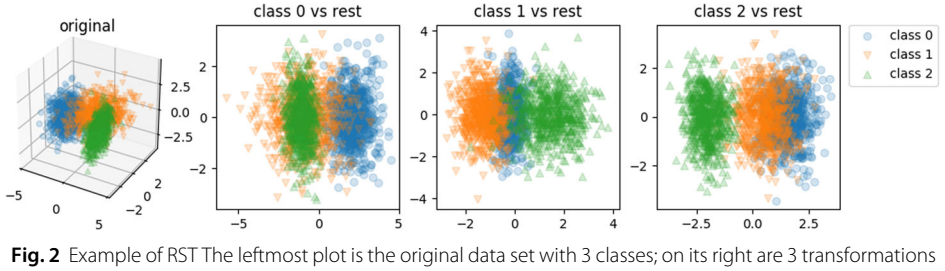

Fig. 2 Example of RST The leftmost plot is the original data set with 3 classes; on its right are 3 transformations learnt by RST. Each of the classes can be seen separated from the rest of the two classes. The classifier will be an ensemble of 3 sub classifiers. Upon training a classifier, the i-th sub-classifier in the ensemble learns on data set that has been transformed by the i-th projection matrix. Upon predicting an unknown sample, the sample will be transformed by the 3 projection matrices into 3 transformed samples for the ensemble to predict

is, lower log loss indicates higher confidence that a classifier makes a correct prediction. Incorrect predictions or uncertain predictions will yield higher log loss.

However naturally linear SVM does not make probabilistic prediction, we obtained it by calibrating our linear SVM model using Niculescu-Mizil's method [16] via a 3-fold cross-validation.

Data sets

Eight data sets are used to evaluate the proposed method:

1. *otto group classification* [17] (61878 samples, 93 dimensions, 9 classes)
2. *mnist digits recognition* [18] (70000 samples, 784 dimensions, 10 classes)
3. *olivetti faces recognition* [19] (400 samples, 4096 dimensions, 40 classes)
4. *sonar: rock vs mine sensory readings* [20] (207 samples, 60 dimensions, 2 classes)
5. *hand written digits* [21] (1797 samples, 64 dimensions, 10 classes)
6. *hand written letter recognition* [22] (20000 samples, 16 dimensions, 26 classes)
7. *glass identification* [23] (214 samples, 9 dimensions, 6 classes)
8. *iris* [24] (150 samples, 4 dimensions, 3 classes)

Results

The comparison of Random Forest and SVM performance on original datasets, linear discriminate analysis (LDA) transformed data sets, PCA transformed data sets and RST transformed data sets is in Tables 1 and 2.

Discussion

Among all the transformer (original - no transformation, RST, LDA, PCA) + classifier (Random Forest, Linear SVM) combinations, SVM combined with RST transformation has the best classification performance (lowest logarithmic loss) over all other combinations on all the data sets except for **sonar** and **letter**. It is noteworthy to mention that none of the transformations improves the classification performance over the original (un-transformed) data set. While on **letter** data set RST+SVM performs better than original+SVM by a small margin but being outperformed by original+RandomForest.

Generally RST works better with linear SVM classifiers than with the non-linear Random Forest classifiers. With non-linear Random Forest Classifier RST, along with other two feature transformers even deteriorates the classification performance. This is not surprising since Random Forest's feature extraction techniques is to use multiple random feature subspaces, i.e. multiple random splits over features; if the set of extracted features are optimised to be useful and compact, its random subspace is surely a less useful representation.

In most of the cases RST reduces the dimensionality and steadily reduces the log loss at the same time, up to a point where the log loss stops to decrease or starts to increase. Therefore in practice is preferable to add a sub set of data to validate the feature transformer learnt by RST while training RST so that RST stops learning when not seeing any improvement over log loss.

Conclusion

The results show that RST is able to extract a fraction of the features from high dimensional data set that improves the classification performance from our empirical

Table 1 Cross validated (32 runs of 3:2 train-test split) linear SVM and random forest performance on original datasets, RST transformed data sets, LDA transformed data sets and PCA transformed data sets

		Original	rst	lda	pca
	No. features	Log loss ± stdev	Log loss ± stdev	Log loss ± stdev	Log loss ± stdev
Otto group					
RandomForest	93 (original)	1.60 ± 0.03			
	80		2.12 ± 0.04		1.69 ± 0.03
	68		3.44 ± 0.08		1.71 ± 0.03
	58		3.39 ± 0.08		1.69 ± 0.02
	49		3.36 ± 0.08		1.74 ± 0.02
	41		3.36 ± 0.15		1.72 ± 0.03
	34		3.25 ± 0.18		1.74 ± 0.03
	8			2.12 ± 0.06	
LinearSVM	93 (original)	0.96 ± 0.04			
	80		0.87 ± 0.04		0.78 ± 0.01
	68		0.77 ± 0.05		0.79 ± 0.03
	58		**0.70 ± 0.01**		0.80 ± 0.03
	49		**0.70 ± 0.01**		0.81 ± 0.02
	41		**0.70 ± 0.01**		0.84 ± 0.03
	34		**0.70 ± 0.01**		0.88 ± 0.03
	8			0.94 ± 0.02	
mnist					
RandomForest	784 (original)	0.46 ± 0.00			
	684		1.24 ± 0.04		1.55 ± 0.04
	597		1.86 ± 0.04		1.50 ± 0.04
	454		1.85 ± 0.05		1.53 ± 0.07
	396		1.87 ± 0.03		1.46 ± 0.03
	345		1.98 ± 0.14		1.33 ± 0.02
	288		1.26 ± 0.20		1.29 ± 0.03
	9			1.04 ± 0.02	
LinearSVM	784 (original)	1.85 ± 0.10			
	684		1.17 ± 0.06		1.33 ± 0.05
	597		0.60 ± 0.09		1.33 ± 0.04
	521		0.68 ± 0.13		1.37 ± 0.11
	396		0.71 ± 0.07		1.35 ± 0.07
	345		0.55 ± 0.09		1.30 ± 0.06
	288		**0.41 ± 0.11**		1.37 ± 0.07
	9			0.52 ± 0.01	
Olivetti faces					
RandomForest	4096 (original)	4.29 ± 1.09			
	720		6.45 ± 0.58		9.58 ± 1.43
	509		3.67 ± 0.40		8.24 ± 0.38
	430		3.32 ± 0.53		8.04 ± 0.75
	375		3.62 ± 0.39		8.13 ± 1.05
	327		4.42 ± 0.34		7.29 ± 0.52
	288		3.96 ± 0.83		7.05 ± 0.83
	39			4.15 ± 0.26	
LinearSVM	4096 (original)	1.98 ± 0.04			
	720		1.21 ± 0.05		1.51 ± 0.09
	509		**1.07 ± 0.04**		1.50 ± 0.04
	430		**1.07 ± 0.04**		1.53 ± 0.08
	375		**1.07 ± 0.04**		1.51 ± 0.05
	327		**1.07 ± 0.04**		1.52 ± 0.09
	285		**1.07 ± 0.04**		1.52 ± 0.06
	39			1.53 ± 0.07	

In bold are the best score (lowest log loss) on the corresponding data set. Results for *otto group classification*, *mnist digits recognition* and *olivetti faces recognition*

Table 2 Cross validated (32 runs of 3:2 train-test split) linear SVM and random forest performance on original datasets, RST transformed data sets, LDA transformed data sets and PCA transformed data sets

		Original	rst	lda	pca
	No. features	Log loss ± stdev	Log loss ± stdev	Log loss ± stdev	Log loss ± stdev
Sonar					
RandomForest	60 (original)	**0.49 ± 0.04**			
	51		1.17 ± 0.38		0.77 ± 0.18
	43		4.83 ± 1.42		0.60 ± 0.03
	36		6.17 ± 1.31		0.57 ± 0.02
	30		6.18 ± 1.98		0.65 ± 0.15
	25		5.78 ± 1.54		0.64 ± 0.15
	20		6.03 ± 1.16		0.78 ± 0.37
	1			9.23 ± 1.21	
LinearSVM	60 (original)	0.52 ± 0.04			
	51		0.56 ± 0.06		0.55 ± 0.04
	43		0.82 ± 0.22		0.55 ± 0.04
	36		0.85 ± 0.21		0.57 ± 0.02
	30		0.85 ± 0.20		0.55 ± 0.04
	25		0.85 ± 0.20		0.55 ± 0.04
	20		0.85 ± 0.20		0.56 ± 0.03
	1			1.00 ± 0.18	
Digits					
RandomForest	64 (original)	0.46 ± 0.07			
	54		1.00 ± 0.22		0.82 ± 0.07
	46		0.92 ± 0.36		0.81 ± 0.17
	39		0.98 ± 0.20		0.73 ± 0.08
	33		0.93 ± 0.19		0.74 ± 0.04
	27		0.87 ± 0.15		0.61 ± 0.08
	22		0.97 ± 0.20		0.59 ± 0.05
	9			0.61 ± 0.04	
LinearSVM	64 (original)	0.25 ± 0.02			
	54		0.29 ± 0.01		0.36 ± 0.02
	46		0.24 ± 0.02		0.38 ± 0.01
	39		**0.23 ± 0.02**		0.36 ± 0.03
	33		0.24 ± 0.02		0.39 ± 0.03
	27		**0.23 ± 0.02**		0.39 ± 0.01
	22		0.24 ± 0.02		0.41 ± 0.02
	9			0.28 ± 0.02	
Letter					
RandomForest	16 (original)	0.64 ± 0.01			
	12		2.78 ± 0.11	1.36 ± 0.05	1.05 ± 0.02
	9		7.17 ± 0.05	1.61 ± 0.05	1.20 ± 0.03
	5		7.08 ± 0.05	2.28 ± 0.07	1.96 ± 0.06
	2		7.12 ± 0.16	4.11 ± 0.07	4.98 ± 0.09
LinearSVM	16 (original)	1.33 ± 0.01			
	12		1.33 ± 0.01	1.62 ± 0.02	1.45 ± 0.01
	9		1.37 ± 0.01	1.89 ± 0.02	1.66 ± 0.01
	5		1.33 ± 0.03	2.20 ± 0.04	2.05 ± 0.02
	2		1.28 ± 0.01	2.51 ± 0.01	2.44 ± 0.01
Glass					
RandomForest	9 (original)	2.43 ± 0.83			
	5		4.33 ± 1.61	3.53 ± 0.83	2.44 ± 0.88
	2		5.80 ± 0.45	3.82 ± 0.61	3.18 ± 1.15
LinearSVM	9 (original)	1.35 ± 0.04			
	5		1.23 ± 0.04	1.20 ± 0.03	1.18 ± 0.03
	2		**1.10 ± 0.04**	1.22 ± 0.03	1.21 ± 0.05
Iris					
RandomForest	4 (original)	0.65 ± 0.36			
	2		0.39 ± 0.68	0.25 ± 0.22	0.28 ± 0.27
LinearSVM	4 (original)	0.46 ± 0.04			
	2		**0.30 ± 0.04**	0.43 ± 0.02	0.48 ± 0.05

In bold are the best score (lowest log loss) on the corresponding data set. Results for sonar: rock vs mine sensory readings, hand written digits, hand written letter recognition, glass identification and iris

experiments. However it is performance is not up to the state-of-art. For instance, in Otto Group Classification Challenge data set [17], our result is only comparable to the results from the second quartile, which is typically from gradient boosted tree models with feature engineering.

It is noteworthy to mention that three major limitations of RST needs further improvement. Firstly, the process of feature selection via SVM weights is not a smooth process, thus optimising RST via efficient numeric methods, for instances, stochastic gradient descent, is infeasible. Secondly, RST algorithm essentially stacks multiple linear transformations. Although fitting data with linear models are fast and efficient, stacked linear transformations are still not capable enough capture the non-linearity in the feature space. Thirdly, RST embeds discriminative information into the input space (feature space) from the output space (class labels). If some outputs (classes) are similar [25] or some samples are mislabelled, RST will make less ideal transformations.

Abbreviations
LDA: Linear discriminate analysis; RFE: Recursive feature elimination; RST: Recursive selective feature transformation; SVM: Support vector machine; PCA: Principle component analysis

Acknowledgments
Not applicable.

Funding
This research is supported by Jiangsu Provincial Science and Technology under Grant No. BK20131182, China.

Availability of data and materials
All the data are available from online open data repositories:

1. *iris* [24]
   https://archive.ics.uci.edu/ml/datasets/iris
2. *hand written letter recognition* [22]
   https://archive.ics.uci.edu/ml/datasets/letter+recognition
3. *glass identification* [23]
   https://archive.ics.uci.edu/ml/datasets/glass+identification
4. *sonar: rock vs mine sensory readings* [20]
   http://archive.ics.uci.edu/ml/datasets/connectionist+bench+(sonar,+mines+vs.+rocks)
5. *hand written digits* [21]
   http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html
6. *mnist digits recognition* [18]
   http://yann.lecun.com/exdb/mnist/
7. *olivetti faces recognition* [19]
   http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
8. *Otto Group Product Classification Challenge* [17]
   https://www.kaggle.com/c/otto-group-product-classification-challenge/data

Authors' contributions
Conception and design of study: XZ, SS-UG. Experiment: XZ. Drafting the manuscript: XZ. Revising the manuscript critically for important intellectual content: Xuan Zhao, Steven Sheng-Uei Guan. Both authors read and approved the final manuscript.

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Received: 22 June 2017 Accepted: 18 October 2017
Published online: 02 December 2017

References

1. Hastie T, Simard PY. Metrics and models for handwritten character recognition. Stat Sci. 1998;13(1):54–65. doi:10.1214/ss/1028905973.
2. Tomasi C, Kanade T. Shape and motion from image streams under orthography: a factorization method. Int J Comput Vis. 1992;9(2):137–54.
3. Markovsky I. Low Rank Approximation London: Springer London; 2012. doi:10.1007/978-1-4471-2227-2. http://link.springer.com/10.1007/978-1-4471-2227-2.
4. Bengio Y, Courville A, Vincent P. Representation learning: A review and new perspectives. IEEE Trans Pattern Anal Mach Intell. 2013;35(8):1798–828.
5. Vapnik VN, Vol. 1. Statistical Learning Theory. 1998.
6. Weston J, Elisseeff A, Schölkopf B, Tipping M. Use of the zero-norm with linear models and kernel methods. J Mach Learn Res. 2003;3:1439–61.
7. Gysels E, Renevey P, Celka P. Svm-based recursive feature elimination to compare phase synchronization computed from broadband and narrowband eeg signals in brain–computer interfaces. Signal Process. 2005;85(11):2178–89.
8. Louw N, Steel S. Variable selection in kernel fisher discriminant analysis by means of recursive feature elimination. Comput Stat Data Anal. 2006;51(3):2043–55.
9. Jolliffe I. Principal Component Analysis. New York: Springer-Verlag; 2002.
10. Tajiri Y, Yabuwaki R, Kitamura T, Abe S. Feature Extraction Using Support Vector Machines Vol 6444, LNCS. 2010. p. 108–15.
11. Rocha A, Goldenstein SK. Multiclass from binary: Expanding one-versus-all, one-versus-one and ecoc-based approaches. IEEE Trans. Neural Netw Learn Syst. 2014;25(2):289–302.
12. Chapelle O. Training a Support Vector Machine in the Primal. Neural Computation. 2007;19(5):1155–78. doi:10.1162/neco.2007.19.5.1155. http://www.mitpressjournals.org/doi/10.1162/neco.2007.19.5.1155.
13. Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ. LIBLINEAR: A Library for Large Linear Classification. J Mach Learn Res. 2008;9(2008):1871–4. doi:10.1038/oby.2011.351.
14. Yu H, Yang J. A direct LDA algorithm for high-dimensional data – with application to face recognition. Pattern Recogn. 2001;34(10):2067–70. doi:10.1016/S0031-3203(00)00162-X.
15. Rosasco L, Vito ED, Caponnetto A, Piana M, Verri A. Are Loss Functions All the Same? Neural Comput. 2004;16(5):1063–76. doi:10.1162/089976604773135104.
16. Niculescu-Mizil A, Caruana R. Predicting good probabilities with supervised learning. In: Proceedings of the 22Nd International Conference on Machine Learning. ICML '05. New York: ACM Press; 2005. p. 625–32. doi:10.1145/1102351.1102430. http://doi.acm.org/10.1145/1102351.1102430.
17. Kaggle. Otto Group Product Classification Challenge. 2015. https://www.kaggle.com/c/otto-group-product-classification-challenge/data. Accessed 05 June 2017.
18. LeCun Y, Cortes C, Burges CJC. Mnist handwritten digit database. AT&T Labs. 2010;2. [Online]. Available: http://yann.lecun.com/exdb/mnist.
19. Ahonen T, Hadid A, Pietikäinen M. Face recognition with local binary patterns. Computer vision-eccv 2004. 2004;469–481. doi:10.1007/978-3-540-24670-1_36. http://link.springer.com/10.1007/978-3-540-24670-1_36.
20. Gorman RP, Sejnowski TJ. Analysis of hidden units in a layered network trained to classify sonar targets. Neural Netw. 1988;1(1):75–89.
21. Denker JS, Gardner W, Graf HP, Henderson D, Howard R, Hubbard WE, Jackel LD, Baird HS, Guyon I. Neural network recognizer for hand-written zip code digits. In: NIPS. 1988. p. 323–31.
22. Frey PW, Slate DJ. Letter recognition using holland-style adaptive classifiers. Mach Learn. 1991;6(2):161–82.
23. Evett IW, Spiehler EJ. Rule induction in forensic science. In: Knowledge based systems. Halsted Press; 1988. p. 152–60. http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/papers/evett_1987_rifs.pdf.
24. Hertz Ja, Krogh AS, Palmer RG, Weigend AS. Introduction to the Theory of Neural Computation. Artificial Intelligence. 1993;I(June):1–17.
25. Zhao X, Guan S, Man KL. An Output Grouping Based Approach to Multiclass Classification Using Support Vector Machines. In: Advanced Multimedia and Ubiquitous Engineering. 2016;389–395. doi:10.1007/978-981-10-1536-6_51. http://link.springer.com/10.1007/978-981-10-1536-6_51.