Least-Squares Regression with Unitary Constraints for Network Behaviour Classification

Antonio Robles-Kelly[1,2](✉)

[1] DATA61 - CSIRO, Tower A, 7 London Circuit, Canberra ACT 2601, Australia
antonio.robles-kelly@data61.csiro.au

[2] College of Engineering and Computer Science, Australian National University, Canberra, Australia

Abstract. In this paper, we propose a least-squares regression method [2] with unitary constraints with applications to classification and recognition. To do this, we employ a kernel to map the input instances to a feature space on a sphere. In a similar fashion, we view the labels associated with the training data as points which have been mapped onto a Stiefel manifold using random rotations. In this manner, the least-squares problem becomes that of finding the span and kernel parameter matrices that minimise the distance between the embedded labels and the instances on the Stiefel manifold under consideration. We show the effectiveness of our approach as compared to alternatives elsewhere in the literature for classification on synthetic data and network behaviour log data, where we present results on attack identification and network status prediction.

1 Introduction

The literature on classification is vast, comprising approaches such as Linear Discriminant Analysis, Projection Pursuit and kernel methods [20]. All these algorithms treat the input data as vectors in high dimensional spaces and look for linear or non-linear mappings to the feature space, often with reduced dimensionality, leading to statistically optimal solutions.

For computer network behaviour, misuse strategies are often use to guarantee security and privacy, whereby breaches and intrusions are recognised using network states or traffic patterns. This can be viewed as a classification over the network behaviour data. Hence, classification methods, such as support vector machines (SVMs) [19] have been traditionally adapted and applied to network behaviour analysis.

One of the most popular classification methods is Linear Discriminant Analysis (LDA) [16]. LDA is a classical method for linear dimensionality reduction which can utilise label information for purposes of learning a lower dimensional space representation suitable for feature extraction, supervised learning and classification. Both LDA and the closely related Fisher's Linear Discriminant (FDA) [13] are concerned with learning the optimal projection direction for binary classes. Further, various extensions and improvements to LDA have been

DOI: 10.1007/978-3-319-49055-7_3

proposed in the literature. For instance, non-parametric discriminant analysis (NDA) [16] incorporates boundary information into between-class scatter. Mika *et al.* [23] and Boudat and Anour [3] have proposed kernel versions of LDA that can cope with severe non-linearity of the sample set. In a related development, Maximum Margin Criterion (MMC) [21] employs an optimisation procedure whose constraint is not dependent on the non-singularity of the within-class scatter matrix.

Indeed, these methods are closely related to manifold learning techniques [22], where a distance metric is used to find a lower dimensional embedding of the data under study. For instance, ISOMAP [29], focuses on the recovery of a lower-dimensional embedding of the data which is quasi-isometric in nature. Related algorithms include locally linear embedding [25], which is a variant of PCA that restricts the complexity of the input data using a nearest neighbour graph and the Laplacian eigenmap method [4] which constructs an adjacency weight matrix for the data-points and projects the data onto the principal eigenvectors of the associated Laplacian matrix. In [15], a pairwise discriminant analysis method with unitary constraints is minimised making use of unconstrained optimisation over a Grassmann manifold.

Note that LDA, PCA and their kernel and regularised versions correspond to particular cases of least-squares reduced rank regression [2,10]. Here, we present a least-squares regression method for classification. We note that the classification labels can be embedded in a Stiefel manifold making use of a matrix of random rotations. This treatment allows us to arrive to a formulation in which the feature vector for an instance to be classified is mapped onto the Stiefel manifold making use of a kernel mapping. The matching of these features and the labels mapped onto the Stiefel manifold can then be effected via a regularised least-squares formulation. This yields a least-squares approach where the aim of computation are given by the parameter matrix for the kernel function and the span of the resulting vector on the Stiefel manifold.

2 Stiefel Manifolds and Least Squares

In this section, we provide some background on Stiefel manifolds [1,6,31] and review the concepts used throughout the paper. With these formalisms, we introduce, later on in the paper, our least-squares approach. To this end, we depart from a linear regression setting and view the j^{th} coefficient $\mathbf{z}_j$ of target vector $\mathbf{z}$ arising from the label $\mathbf{y}$ corresponding to the instance $\mathbf{x}$ in the training set Ψ as a linear combination of k kernel functions $K(\mathbf{A}_i, \mathbf{x})$ such that

$$\mathbf{z}_j = \sum_{i=1}^{k} b_{j,i} K(\mathbf{A}_i, \mathbf{x}) + \epsilon_{\mathbf{x}} \tag{1}$$

where $\epsilon_{\mathbf{x}}$ is the residual for $\mathbf{x}$, $\mathbf{A}_i$ are the parameters for the i^{th} kernel function K and $b_{j,i}$ are mixture coefficients.

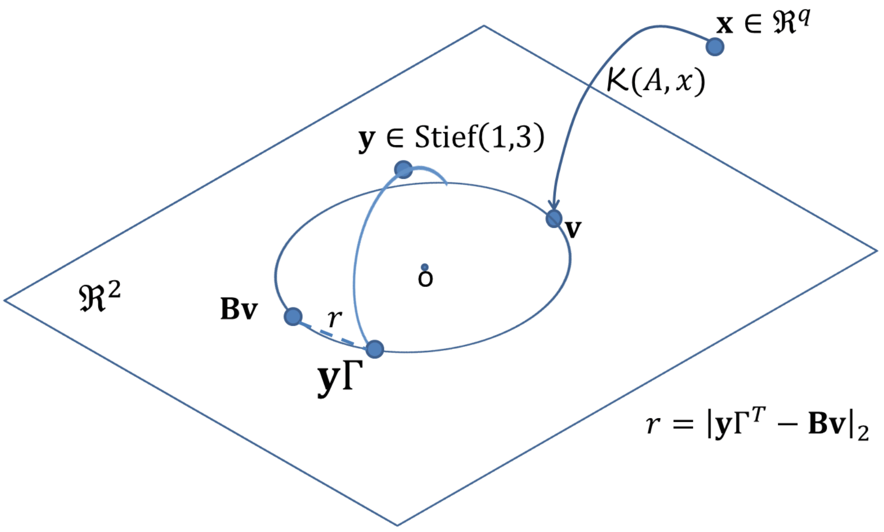

Fig. 1. Graphical illustration of our least-squares approach. In the figure, we have denoted the origin of the example $\Re^2$ space as o and used the notation introduced in Sect. 2.

Here, we aim at minimising the residual by considering the case where the targets $\mathbf{z}$ are p-dimensional vectors in a real Stiefel manifold [18]. Recall that a Stiefel manifold Stief(p, n) is the set of all orthonormal p-dimensional frames in $\Re^n$, where n and p be positive integers such that $p \leq n$. This formalism is important since, with these ingredients, we can define the span Span$(\mathbf{v})$, where $\mathbf{v} \in$ Stief(p, n), in a straightforward manner as a linear operation on a p-dimensional unit vector in $\Re^n$. Moreover, in practice, we can consider the span Span$(\mathbf{v})$ to be a matrix $\mathbf{B}$ and denote the set of vectors spanned by all the possible choices of $\mathbf{B}$ as $\mathcal{V}_{\mathbf{B}}$.

To formulate our least-squares cost function, we consider the Stiefel manifold Stief$(1, n)$, *i.e.* the manifold comprised by the 1-frames in $\Re^n$, and view it as the frame of a unit ball of dimension n. In Fig. 1, we use the case where $n = 2$ to elaborate further on the approach taken here and provide a better intuition of the concepts presented previously in the section. In the figure, we show a vector $\mathbf{v}$ in the manifold Stief$(1, 2)$. Note that we have used the kernel map $\mathcal{K} : \Re^n \times \Re^m \times \Re^m \rightarrow \mathcal{S}^n$ to embed a vector $\mathbf{x}$ in $\Re^m$ onto the unit n-sphere $\mathcal{S}^n$ such that $\mathbf{v} = \mathcal{K}(\mathbf{A}, \mathbf{x})$, where $\mathbf{A}$ is a set of parameters in $\Re^n \times \Re^m$. It is worth noting in passing that we have chosen to define the parameters of the function $\mathcal{K}$ in this manner to be consistent with the linear case in Eq. 1, *i.e.* $\mathbf{v} = \frac{\mathbf{Ax}}{|\mathbf{Ax}|_2}$ where $\mathbf{A}$ is a matrix whose i^{th} column is $\mathbf{A}_i$ and $| \cdot |_2$ denotes the L-2 norm.

Later on, we comment further on our choice of function $\mathcal{K}$. For now, we continue our discussion on Figure 1. Note that, in the figure, we also show a label vector $\mathbf{y} \in$ Stief$(1, 3)$ and the target induced by the matrix Γ onto

Stief$(1, 2)$. Here, we focus on matrices Γ that correspond to rotations about the origin, *i.e.* $\Gamma\Gamma^T = \mathbf{I}$, where $\Gamma^{-1} = \Gamma^T$ and $\mathbf{I}$ is the identity matrix. As can be observed from the figure, this treatment leads in a straightforward manner to a geometric interpretation of the distances in the ambient space and the consequent formulation of the least-squares cost function. With the notation and concepts introduced above, we can write the cost function as follows

$$f = \sum_{\mathbf{x}\in\Psi} \epsilon_{\mathbf{x}}^2 = \sum_{\mathbf{x}\in\Psi} |\mathbf{y}\Gamma - \mathbf{B}\mathcal{K}(\mathbf{A}, \mathbf{x})|_2^2 \tag{2}$$

where the target for the instance $\mathbf{x}$ is $\mathbf{z} = \mathbf{y}\Gamma$ and the entries of the matrix $\mathbf{B}$ are given by $b_{i,j}$.

3 Implementation and Discussion

3.1 Optimisation and Initialisation

As a result, the aim of computation becomes that of recovering $\mathbf{A}$ and $\mathbf{B}$ such that f is minimum subject to $|\mathbf{B}\mathcal{K}(\mathbf{A}, \mathbf{x})|_2 = 1$, where $|\cdot|_2$ denotes the L-2 norm. Note that, the minimisation of the cost function in Eq. 2 can be rewritten as a maximisation by using the inner product and adding an L-2 norm term so as to induce sparsity. Thus, here we solve

$$\Theta = \max_{\mathbf{A},\mathbf{B}} \left\{ \sum_{\mathbf{x}\in\Psi} \langle \mathbf{y}\Gamma, \mathbf{B}\mathcal{K}(\mathbf{A}, \mathbf{x})\rangle - \frac{\lambda}{|\Psi|}|\mathbf{B}|_2^2 \right\} \tag{3}$$

where λ is a constant that controls the influence of the regularisation term in the maximisation process, as before $|\cdot|_2$ denotes the matrix L-2 norm and we have used Θ as a shorthand for the set of parameters comprised by the maximisers $\mathbf{A}^*$ and $\mathbf{B}^*$ of the cost function over the space of solutions for $\mathbf{A}$ and $\mathbf{B}$, *i.e.* $\Theta = \{\mathbf{A}^*, \mathbf{B}^*\}$.

In practice, the i^{th} coefficient of the label vector for the feature $\mathbf{x}$ can be defined as

$$\mathbf{y}_i = \begin{cases} 1 & \text{if } \mathbf{x} \text{ belongs to the class indexed } i \text{ in } \Psi \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

whereas the rotation matrix Γ can be computed exploiting the group structure of all rotations about the origin. We do this by using the algorithm in [11] as follows. We commence by constructing a 2×2 rotation matrix, *i.e.* a two-dimensional rotation. With this matrix in hand, we recursively increase the dimensionality from q to $q + 1$ by augmenting the current $q \times q$ rotation matrix making use of the embedding of a vector in an $q+1$ dimensional sphere. The idea underpinning this procedure is that a vector on an q-dimensional sphere should correspond to the embedding induced by the rotation matrix operating on an analogous vector on a sphere in $q + 1$ dimensions.

For the optimisation, we employ a coordinate descent scheme [14]. The idea is to optimise the multivariable cost function in Eq. 3 by maximising it along one coordinate direction at a time while keeping all other coordinates fixed. Thus, the multivariable problem is in fact solved by computing a sequence of two optimisation subproblems, *i.e.* one for $\mathbf{A}$ and another one for $\mathbf{B}$. For each subproblem, we perform line search optimisation in the steepest gradient descent direction.

Our choice of this optimisation scheme resides in the notion that, when maximising in the coordinate direction spanned by $\mathbf{A}$, the problem in hand turns into a standard non-linear least-squares one which can be solved, in a straightforward manner, using an interior point trust region method [8]. Once the matrix $\mathbf{A}^*$ for the current iteration is in hand, computing the iterate of $\mathbf{B}^*$ can be done by viewing the problem as a standard regularised linear least squares one. We iterate until convergence is reached, which is when the norm difference between $\mathbf{A}^*$ and $\mathbf{B}^*$ for consecutive iterations is less or equal to a predefined tolerance τ.

In practice, for our gradient descent approach above, we require an initial estimate of $\mathbf{A}$ prior to the recovery of $\mathbf{B}$ for our first iteration. For our initial estimate of $\mathbf{A}$, we have used randomly generated values in the interval $[-1, 1]$ and normalised the columns of $\mathbf{A}$ to unity.

3.2 Classification

Note that, so far, we have not constrained the dimensionality of the feature vector or the vectors given by $\mathbf{y}\Gamma$. Its clear, however, that if the feature vector dimensionality is m and that of the vector $\mathbf{Ax}$ is n, the matrix $\mathbf{A}$ should be of size $n \times m$. Similarly, for a training set containing k classes, the vector $\mathbf{y}$ is expected to be of length k and, hence, the matrix Γ is of size $k \times k$. Further, since we have imposed the constraint $\mathbf{BB}^T = \mathbf{I}$, $\mathbf{B}$ should also be $k \times k$.

Nonetheless at first glance this fixes the dimensionality of the matrices $\mathbf{A}$ and $\mathbf{B}$ to correspond to $k \times m$ and $k \times k$, respectively, note the length of the vector $\mathbf{y}$ can be set to any length $m \geq k$. This implies that those coefficients whose indexes are greater than k will be identical to zero. The main advantage of this treatment strives in the fact that the dimensionality of both matrices, $\mathbf{A}$ and $\mathbf{B}$, can be rendered independent of the number of classes.

To classify a testing instance $\hat{\mathbf{x}}$, we employ a nearest-neighbour classifier [12] with respect to the vectors $\mathbf{y}\Gamma$ as yielded by $\mathbf{B}\mathcal{K}(\mathbf{A}, \hat{\mathbf{x}})$. This is consistent with the notion that the cost function being minimised at training is, in effect, the Euclidean distance of the feature vectors with respect to their label vectors embedded in the Stiefel manifold under consideration. For the kernel function, here we use both, a Radial Basis function (RBF) [5] and a linear kernel. For the linear kernel, the i^{th} element of the vector $\mathcal{K}(\mathbf{A}, \mathbf{x})$ is given by

$$K_L(\mathbf{A}_i, \mathbf{x}) = \frac{\langle \mathbf{A}_i, \mathbf{x} \rangle}{\mid \mathbf{Ax} \mid_2} \tag{5}$$

where $\mathbf{A}_i$ denotes the i^{th} column of the matrix $\mathbf{A}$. Similarly, for a matrix $\mathbf{A}$ with n columns the RBF kernel is given by

$$K_{RBF}(\mathbf{A}_i, \mathbf{x}) = \frac{\exp\big(-\rho|\mathbf{A}_i - \mathbf{x}|_2^2\big)}{\Big|\sum_{i=1}^{n} \exp\big(-\rho|\mathbf{A}_i - \mathbf{x}|_2^2\big)\Big|_2} \tag{6}$$

where, as before, $|\cdot|_2$ denotes the vector L-2 norm, ρ^{-1} is the bandwidth of the kernel function and $\langle\cdot,\cdot\rangle$ accounts for the dot product.

3.3 Relation to Neural Networks and Support Vector Machines

As mentioned above, here we use RBF and linear kernels. These are kernels commonly used in neural networks [17] and Support Vector Machines (SVMs) [9]. Further, the nearest neighbour classifier as used here is reminiscent of the mean-squared error soft-max linear classifier in feed forward neural networks [24]. Despite these apparent similarities, there are a number of differences between our method and single-layer neural networks and SVMs.

Regarding SVMs, we commence by noting that, for binary classification, the dual form of the support vector machine depends on a single set of parameters, *i.e.* the alpha-weights, where the dividing hyperplane is given by $\langle \mathbf{w}, \mathbf{x}\rangle + b = 0$ where b is the intersect of the hyperplane and $\mathbf{w}$ is its normal vector.

Amongst those variants of the SVMs elsewhere in the literature, the least-squares support vector machine [30] imposes a quadratic penalty function which is probably the one that most closely resembles our formulation. Here, we limit our discussion to the linear case. We do this without any loss of generality and for the sake of clarity. Further, extending the following to the "kernelised" setting is a straightforward task. For non-linearly separable training data, the minimisation in hand for the linear least-squares SVM is given by [27].

$$\min_{\mathbf{w}} \left\{ \langle \mathbf{w}, \mathbf{w}\rangle + \varsigma \sum_{\mathbf{x}\in\Psi} \big(y - (\langle \mathbf{w}, \mathbf{x}\rangle + b)\big)^2 \right\} \tag{7}$$

where $\langle\cdot,\cdot\rangle$ denotes, as usual, the dot product, $y \in -1, 1$ is the instance label and the hyperplane normal vector is

$$\mathbf{w} = \sum_{\mathbf{x}\in\Psi} y\alpha\mathbf{x} \tag{8}$$

ς is a hyperparameter and α is a positive scalar, *i.e.* a slack variable, such that

$$\sum_{\mathbf{x}\in\Psi} y\alpha = 0 \tag{9}$$

In the equations above, we have written y instead of $\mathbf{y}$ to reflect the fact that $\mathbf{y}$ is a vector, which in the binary case where the length of the vector equals the number of classes in the training set is given by $[1,0]^T$ or $[0,1]^T$ whereas y is a scalar in $\{-1,1\}$. In addition to this, from inspection, its clear the two cost functions, ours in Eq. 3 and that of the least-squares SVM in Eq. 7 are quite distinct. Also, note that there is no constraint on the number of support vectors in Eq. 7, *i.e.* the number of feature vectors $\mathbf{x}$ contributing to the hyperplane normal vector $\mathbf{w}$. This contrasts with our method, where the complexity of evaluating the kernel is given by the number of rows in $\mathbf{A}$. This is a major difference with respect to SVMs, where non-linear kernels can be quite cumbersome to evaluate when the number of support vectors is large.

Now we turn our attention to the multiple class case and comment on the similarities of our method with single-layer feed forward networks [17]. Note that the matrix $\mathbf{A}$ can be viewed as the weights of the hidden layer of the network and the $\mathbf{B}$ as the coefficients of the all-connected layer. That said, there are two major differences. The first of these pertains the cost function itself, whereby, there is no evident analogue of our rotation matrix Γ in neural networks. The second relates to the optimisation scheme used here. Recall that, in neural networks, the training is often achieved by back propagation [26]. In contrast, our approach is an iterative coordinate descent one which imposes unitary constraints on $\mathbf{B}\mathcal{K}(\mathbf{A}, \mathbf{x})$. For our approach, the final classification step via a nearest-neighbour classifier is akin to the classification layer often used in neural networks. This opens-up the possibility of using well known techniques on soft-max and entropy-based classification layers as an alternative to our nearest-neighbour classifier.

4 Experiments

In this section, we show classification results yielded by our approach and provide comparison with alternative methods elsewhere in the literature. To this end, we commence by presenting qualitative results using synthetic data and, later on in the section, we focus our attention on intrusion detection using a widely available network behaviour data set.

4.1 Synthetic Data

To illustrate the utility of our method for classification, we have used two synthetically generated point clouds in 2D. The first of these accounts for a binary classification setting comprising 750 points arranged in an annulus where the centre corresponds to one class and the periphery to another one. The second of these data sets is given by 1500 points describing two annuli distributed into four well defined clusters.

For both points clouds, we have used an $\mathbf{A}$ matrix with 10 rows, *i.e.* $n = 10$, and a rotation Γ and $\mathbf{B}$ matrices of size 10×10. For purposes of comparison we have used a least-squares SVM (LS-SVM) [30] for binary classification and, for the multiclass annuli data, an stochastic feed forward neural network [28] with a single hidden layer composed of 10 neurons.

In Fig. 2, we show the results yielded by our method and the alternatives. In the left-hand column, we show the input point clouds. In the middle and right-hand columns we show the results yielded by our method and the alternatives, respectively. Note that, from the figure, we can note that our method delivers very similar results to those obtained by the LS-SVM. Nonetheless these classification boundaries are very similar, the LS-SVM yields 116 support vectors. This contrasts with our approach, which employs a matrix $\mathbf{A}$ with 10 rows. Thus, for

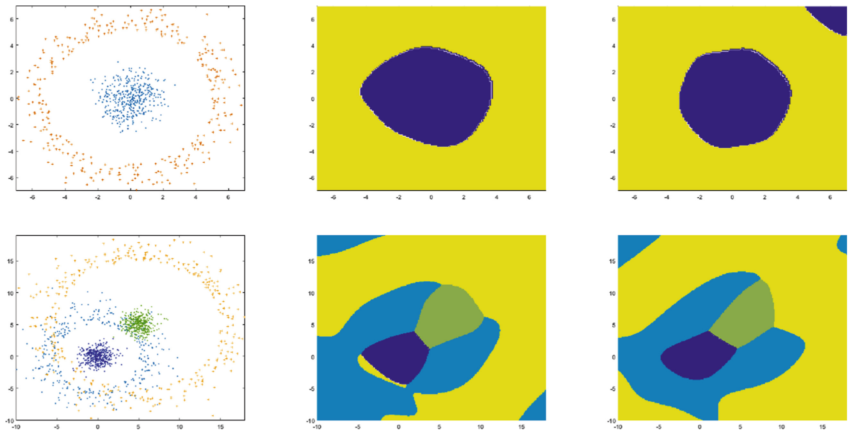

Fig. 2. Classification results on synthetic data. Left-hand column: input point clouds; Middle column: results yielded by our approach; Right-hand column: results delivered by a LS-SVM (top panel) and an RBF neural network (bottom panel).

testing, the evaluation of the kernel for our method is approximately an order of magnitude less computationally costly. Regarding multiclass classification, note the boundaries yielded by the neural network are smoother than those obtained by our method. As a result, the region comprising the small group of points next to the centre of the second annulus are completely misclassified. In contrast, the decision boundary yielded by our algorithm does classify these points correctly.

4.2 Network Behaviour Classification

We now turn our attention to the task of cyber attack discrimination in network behaviour data. Here, we view this task as a classification one where the aim is to discern attacks from legitimate connections in the network. For our experiments, we have used the sanitized Kyoto University benchmark data[1]. These benchmark dataset is comprised of the traffic net logs for the Kyoto University's honeypots from November 2006 to August 2009[2]. The data contains 24 features including the attack label and the connection state identifier and is arranged by day per calendar month.

For our experiments, we have used the data corresponding to November 2006 and trained our method on the connections established on the even days of the month. We have then tested our method and the alternatives on the data corresponding to each of the odd days. For our method, we have used a matrix **A** with 30 rows and, for the neural network, whenever applicable, have set to 30 the number of hidden neurons.

[1] The data is accessible at http://www.takakura.com/Kyoto_data/.

[2] The description of the data can be found at http://www.takakura.com/Kyoto_data/BenchmarkData-Description-v5.pdf.

Table 1. Average false classification rate and std. deviation (in percentage) for the attack detection task on the Kyoto University benchmark data.

	Mean	St. dev
Our method (linear)	1.056	1.011
Our method (RBF)	0.967	0.974
LS-SVM	4.839	3.331

Table 2. Average false classification rate and std. deviation (in percentage) for the attack-connection state combinations on the Kyoto University benchmark data.

	Mean	St. dev
Our method (linear)	14.273	3.893
Our method (RBF)	9.027	2.941
Neural network	11.746	4.342
LDA	33.656	19.379

Here, we make use of our method for both, attack detection and attack-connection state classification. The former of these tasks is a binary classification one whereas the latter is a multiple class one. The data contains a label entry which indicates whether the connection under study was an attack. It also contains a connection state variable that indicates the status of the connection, generally when it ends. Since the dataset provides 13 different connection states, this yields 26 classes, *i.e.* for each state, there is the possibility of the connection being an attack or not.

In Tables 1 and 2 we show the mean false classification rate and the corresponding std. deviation for our method and the alternatives for both, attack detection and attack-connection state classification. Since our method employs a random initialisation for the matrix $\mathbf{A}$, we have effected ten trails and averaged accordingly before computing the mean and std. deviation results shown in the tables. In our experiments, for the attack detection, we have again compared our results to those yielded by the LS-SVM. For the attack-connection state classification task, we have compared our results with those delivered by the neural network in [28] and those yielded by linear discriminant analysis (LDA) [7]. From the tables, its clear that our method outperforms the alternatives. Moreover, when an RBF is used, our method can correctly predict an attack 99.033% of the time and can account for the status of the network at termination of the connection with 90.973% accuracy. It also delivers a lower standard deviation and is quite competitive even when a linear kernel is used.

5 Conclusions

In this paper, we have proposed a least-squares reduced rank regression method on with applications to classification. Departing from the notion that a kernel function can be used to map an input instance to a Steifel manifold, we have used a rotation matrix to pose the problem of recovering the span and kernel parameter matrices in a least-squares minimisation setting. We have discussed the link between our approach and support vector machines and feed forward neural networks and shown qualitative results on synthetic data. We have also performed experiments on a widely available network behaviour dataset and compared our approach to alternatives elsewhere in the literature.

References

1. Absil, P.A., Mahony, R., Sepulchre, R.: Riemannian geometry of Grassmann manifolds with a view on algorithmic computation. Acta Applicandae Math. **80**(2), 199–220 (2004)
2. Anderson, T.W.: Estimating linear restrictions on regression coefficients for multivariate normal distributions. Ann. Math. Stat. **22**(3), 327–351 (1951)
3. Baudat, G., Anouar, F.: Generalized discriminant analysis using a kernel approach. Neural Comput. **12**(10), 2385–2404 (2000)
4. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Neural Information Processing Systems, vol. 14, pp. 634–640 (2002)
5. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2006)
6. Boothby, W.M.: An Introduction to Differentiable Manifolds and Riemannian Geometry. Academic Press, Cambridge (1975)
7. Borg, I., Groenen, P.: Modern Multidimensional Scaling, Theory and Applications. Springer Series in Statistics. Springer, Heidelberg (1997)
8. Coleman, T.F., Li, Y.: An interior, trust region approach for nonlinear minimization subject to bounds. SIAM J. Optim. **6**, 418–445 (1996)
9. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines. Cambridge University Press, Cambridge (2000)
10. De la Torre, F.: A least-squares framework for component analysis. IEEE Trans. Pattern Anal. Mach. Intell. **34**(6), 1041–1055 (2012)
11. Diaconis, P., Shahshahani, M.: The subgroup algorithm for generating uniform random variables. Probab. Eng. Inf. Sci. **1**, 15–32 (1987)
12. Duda, R.O., Hart, P.E.: Pattern Classification. Wiley, Hoboken (2000)
13. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Ann. Eugenics **7**, 179–188 (1936)
14. Friedman, J., Hastie, T., Tibshirani, R.: Regularization paths for generalized linear models via coordinate descent. J. Stat. Softw. **33**(1), 1–22 (2010)
15. Fu, Z., Robles-Kelly, A., Tan, R.T., Caelli, T.: Invariant object material identification via discriminant learning on absorption features. In: Object Tracking and Classification in and Beyond the Visible Spectrum (2006)
16. Fukunaga, K.: Introduction to Statistical Pattern Recognition, 2nd edn. Academic Press, Cambridge (1990)
17. Hassoun, M.H.: Fundamentals of Artificial Neural Networks. MIT Press, Cambridge (1995)
18. James, I.M.: The Topology of Stiefel Manifolds. Cambridge University Press, Cambridge (1976)
19. Khan, L., Awad, M., Thuraisingham, B.: A new intrusion detection system using support vector machines and hierarchical clustering. Int. J. Very Large Data Bases **16**(4), 507–521 (2007)
20. Landgrebe, D.: Hyperspectral image data analysis. IEEE Sig. Process. Mag. **19**, 17–28 (2002)

21. Li, H., Jiang, T., Zhang, K.: Efficient and robust feature extraction by maximum margin criterion. In: Neural Information Processing Systems, vol. 16 (2003)
22. Ma, Y., Fu, Y.: Manifold Learning Theory and Applications. CRC Press, Inc., Boca Raton (2011)
23. Mika, S., Ratsch, G., Weston, J., Scholkopf, B., Muller, K.: Fisher discriminant analysis with kernels. In: IEEE Neural Networks for Signal Processing Workshop, pp. 41–48 (1999)
24. Neal, R.M.: Bayesian Learning for Neural Networks. Springer, New York (1996)
25. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**, 2323–2326 (2000)
26. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**, 533–536 (1986)
27. Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B., Vandewalle, J.: Least Squares Support Vector Machines. World Scientific, Singapore (2002)
28. Tang, Y., Salakhutdinov, R.R.: Learning stochastic feedforward neural networks. In: Advances in Neural Information Processing Systems, pp. 530–538 (2013)
29. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290**(5500), 2319–2323 (2000)
30. Van Gestel, T., Suykens, J.A.K., De Brabanter, J., Lambrechts, A., De Moor, B., Vandewalle, J.: Bayesian framework for least-squares support vector machine classifiers, Gaussian processes, and kernel fisher discriminant analysis. Neural Comput. **14**(5), 1115–1147 (2002)
31. Wong, Y.C.: Differential geometry of Grassmann manifolds. Proc. Natl. Acad. Sci. United States Am. **57**(3), 589–594 (1967)