Visualising kernel spaces

Lech Szymanski and Brendan McCane
Department of Computer Science
University of Otago
PO Box 56
Dunedin, New Zealand
Email: {lechszym,mccane}@cs.otago.ac.nz

***Abstract*—Classification in kernel machines consists of a non-linear transformation of input data into a feature space, followed by a separation with a linear hyperplane. This transformation is expressed through a kernel function, which is capable of computing similarities between two data points in an abstract geometric space for which individual point vectors are computationally intractable. In this paper we combine the notion of kernel distance and methods for data dimensionality reduction to obtain visualisations for such kernel spaces.**

I. Introduction

Despite being a linear classifier, a Support Vector Machine (SVM) [1] is capable of classifying non-linearly separable data due to the *kernel trick* [2], [3]. The idea is to pass the classifier's input through a non-linear transformation into a feature space where the points are likely to become linearly (or nearly linearly) separable. This gives the SVM's linear separating hyperplane a chance of classifying complex data patterns. The *trick* is to optimise the SVM cost function using its dual form which depends on the dot product of the data vectors in the feature space. Thus a desired transformation can be made with a kernel function, which is capable of computing such a dot product for spaces where individual point coordinates are not computable. This allows for classification in abstract geometric spaces with a large (even infinite) number of dimensions. We will refer to these spaces as kernel spaces.

Understanding the internal representation of data in a kernel space is important since there are a number of kernel functions to choose from when performing classification. Different kernel functions are best suited for different problems (although certain kernels work remarkably well on a wide range of datasets). In addition, each function has a number of parameters that must be set prior to classification. The choice of these parameters determines the outcome of the classification, and often the generalisation capabilities of the classifier. We propose a visualisation technique that gives an idea of the data distribution in the kernel space. A similar approach has been used to compare medical images [4], we just expand its use to the analysis of the inner workings of various kernel functions and to applications in classification.

The visualisations presented in this paper are based on the concept of kernel distance, which is explained in section II. In section III, we show how this distance is used to create visualisations with multidimensional scaling and include a number of examples.

II. Kernel distance

The kernel function computes the dot product of two points in a kernel space. Thus, given two data vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, where $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^m$ of arbitrary dimension m, the kernel function expresses the following relation:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j), \tag{1}$$

where $\Phi(\mathbf{x}_i)$ is the projection of $\mathbf{x}_i$ into the kernel space. The value of the kernel function relates the *similarity* of two vectors in the kernel space.

For our visualisations we need to convert this similarity to a distance measure. Such a measure has already been proposed and it is called the *kernel distance* [5], [6]. It is defined as the magnitude of the difference vector between two points in the kernel space:

$$d_{ij} = ||\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)||, \tag{2}$$

where $||\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)|| = \sqrt{\Big[\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\Big]^T \Big[\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\Big]}$. The kernel distance is the Euclidean distance between the two vectors in the kernel space. It is fairly trivial to expand equation 2 to get the following relation:

$$d_{ij} = \sqrt{\Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_i) + \Phi(\mathbf{x}_j)^T \Phi(\mathbf{x}_j) - 2\Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)}, \tag{3}$$

which is the same as

$$d_{ij} = \sqrt{K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j)}. \tag{4}$$

The fact that kernel distance can be computed in terms of the kernel function is of great importance, because for many kernels, a given vector $\Phi(\mathbf{x}_i)$ is not computable, despite the fact that $K(\mathbf{x}_i, \mathbf{x}_j)$ is. Note that kernel distance is defined only for kernels that produce positive definite kernel distance matrices.

III. Visualisation

With the kernel distance computed using equation 4, we can apply a dimensionality reduction technique to create a dataset of arbitrary dimension that approximates the distribution of the data in the kernel space. In this work we use multidimensional scaling (MDS) [7], which attempts to preserve relative distances between points. For a dataset of N points $\mathbf{x}_1, ..., \mathbf{x}_N$, the representation of their distribution in the kernel space is

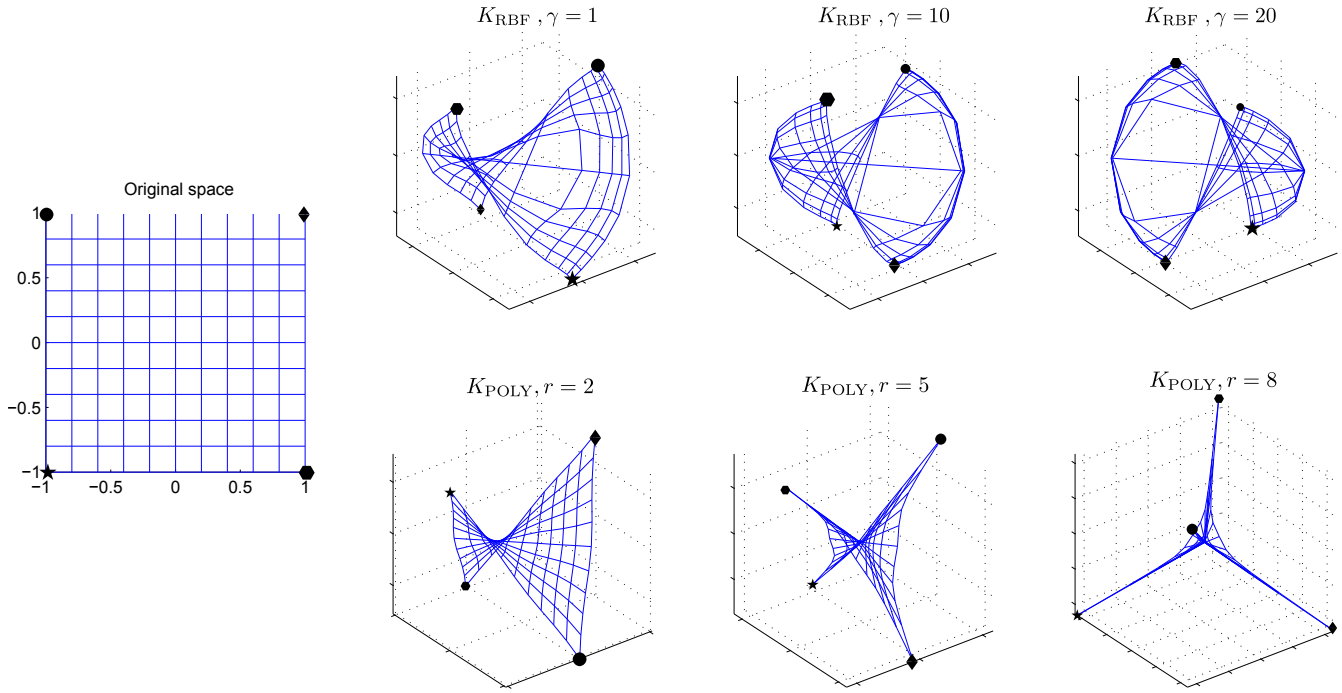

Fig. 1. 3D visualisation of the transformation of a 2D mesh, centred at the origin, (shown on the left) to RBF and polynomial kernel spaces with various parameter settings; the corners of the mesh in the original and the kernel space are marked with different symbols; coordinates of the visualisation are not shown, since they are arbitrary – it's the relationship between points that is of importance.

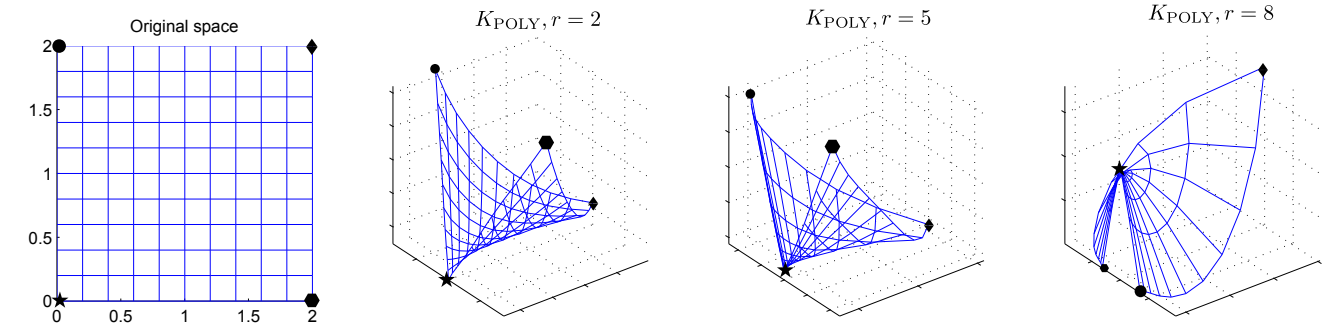

Fig. 2. 3D visualisation of the transformation of a 2D mesh, offset from the origin, (shown on the left) to polynomial kernel spaces with various parameter settings; the corners of the mesh in the original and the kernel space are marked with different symbols.

approximated by N vectors $\mathbf{y}_1, ...\mathbf{y}_N$ of arbitrary dimension by minimising the following cost function with respect to $\mathbf{y}_1, ..., \mathbf{y}_N$:

$$J = \sum_{i=1}^{N} \sum_{j>i}^{N} (||\mathbf{y}_i - \mathbf{y}_j|| - d_{ij})^2. \quad (5)$$

Other distance preserving techniques, such as Local Linear Embedding [8] for instance, should work just as well.

In this paper we feature two kernel functions:

- the radial basis function kernel (RBF), often referred to as the gaussian kernel, defined as

$$K_{\mathrm{RBF}}(\mathbf{x}_i, \mathbf{x}_j) = \exp^{-\gamma||\mathbf{x}_i - \mathbf{x}_j||}, \quad (6)$$

with variable parameter $\gamma > 0$,

- the polynomial kernel, defined as

$$K_{\mathrm{POLY}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + 1)^r, \quad (7)$$

with variable parameter r.

A. Visualising 2D surfaces in kernel spaces

To get an idea of the transformation that a geometric space undergoes in a kernel space, we created 3D visualisations of a uniformly sampled 2D surface after being passed into different kernel spaces (see Figures 1 and 2). We connected the points into a mesh to give a sense of how the surface morphs after the transformation. The reason why the MDS scaling is done in 3D rather than 2D is to emphasise that the kernel transformation adds dimensionality to the data.

For the RBF kernel, increasing the γ value tends to push points towards the edge of the space such that close neighbours

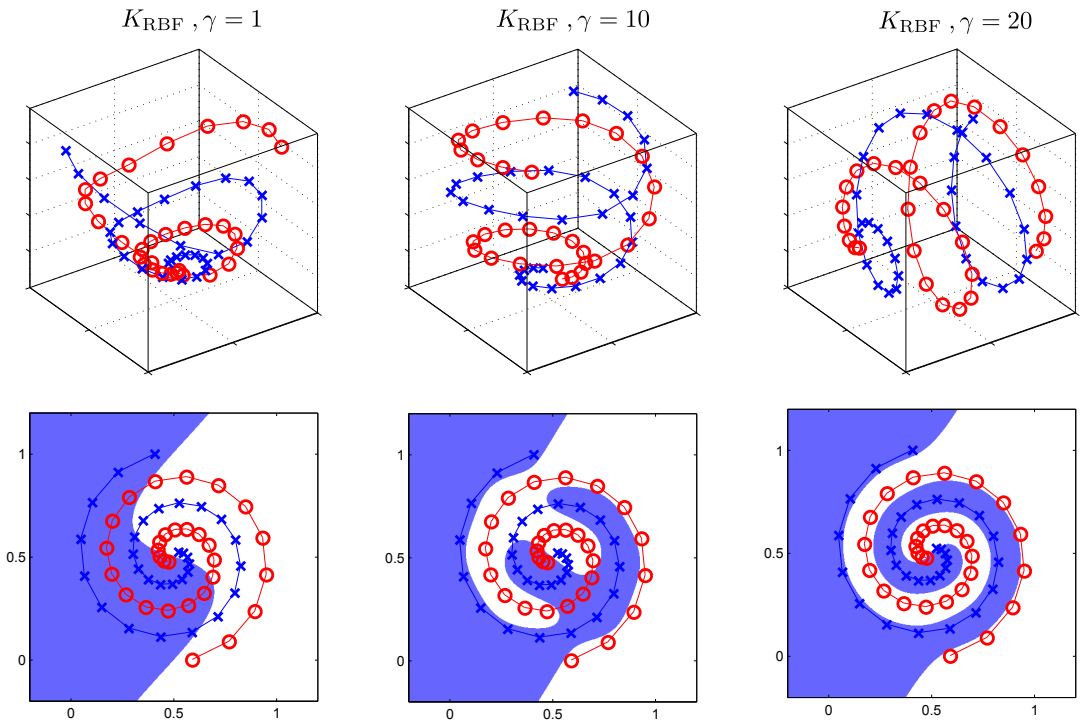

Fig. 3. A 3D visualisation of a two-dimensional spiral pattern in an RBF kernel space with various values of γ (top row) and the corresponding SVM classification (below); the desired class membership for a given point is indicated by marking it with a red circle or a blue cross; the classification results after training on an SVM are shown as classification regions in blue (for crosses) and white (for circles); the points have been joined with a line following the spiral to help spot the pattern in the 3D representation of the kernel space.

become even closer, whereas points beyond a certain distance threshold (inversely proportional to the value of γ) become identically distant. This creates an increasing void in the middle of the space as γ increases. The polynomial kernel transformation, on the other hand, warps the space so that the corners become stretched towards the edge of the space while the middle points push towards the centre. In addition, the warping is dependent on the magnitude of individual vectors in the original space, and so the polynomial kernel transformation is different for datasets with different offsets from the origin (as shown in Figure 2). Conversely, the RBF kernel depends only on the pair-wise distances between the data points, and so a shift of the dataset in the original space does not affect its distribution in the RBF kernel space. Hence, the RBF kernel visualisations of the mesh, regardless of the offset from the origin, are all identical.

B. Visualising data patterns in kernel spaces

Kernel space visualisation can be used to get an idea of the data representation with respect to classification. Figure 3 shows a 3D visualisation of a spiral binary classification problem in a number of RBF kernel spaces. These plots are accompanied by the results of the SVM classification for the corresponding kernel settings[1]. The visualisations show what happens to a complex pattern, the spiral, under the RBF kernel transformation as the value of γ increases. As the mid-space void grows in the kernel space and neighbouring points get closer, the spiral unwinds. It is not possible to see a complex pattern, like the spiral, unwind completely in the 3D visualisation, because it takes an infinite amount of dimensions of the RBF kernel space to achieve that. However, Figure 3 shows a glimpse of how the pattern becomes more separable.

[1]All classifications results presented in this paper were obtained with the LIBSVM library [9].

C. Aiding classification with kernel space visualisations

Finally, we present a scenario where kernel space visualisation can aid in choosing the appropriate kernel parameters for SVM classification. For this demonstration we use the splice-junction gene sequence (SJGS) dataset sourced from the UCI Machine Learning Repository [10]. This dataset consists of over 3000 DNA sequences that contain *spliced out* regions, sections of DNA removed during the process of protein creation, categorised into three classes. These sequences are formed into 240-dimensional vectors that constitute the input to the SVM classifier. One third of the data is used for training while the other two thirds is withheld for later testing of the classifier's performance. Figure 4 shows a 3D visualisation of a 200-point sample chosen at random from the training set in an RBF kernel space with different settings of parameter γ. The diagrams indicate that out of three choices shown, the kernel with $\gamma = 0.001$ tends to group data best. As the value of γ increases, the three classes seem more mixed in the kernel space. This suggests that the RBF kernel with $\gamma = 0.001$ gives

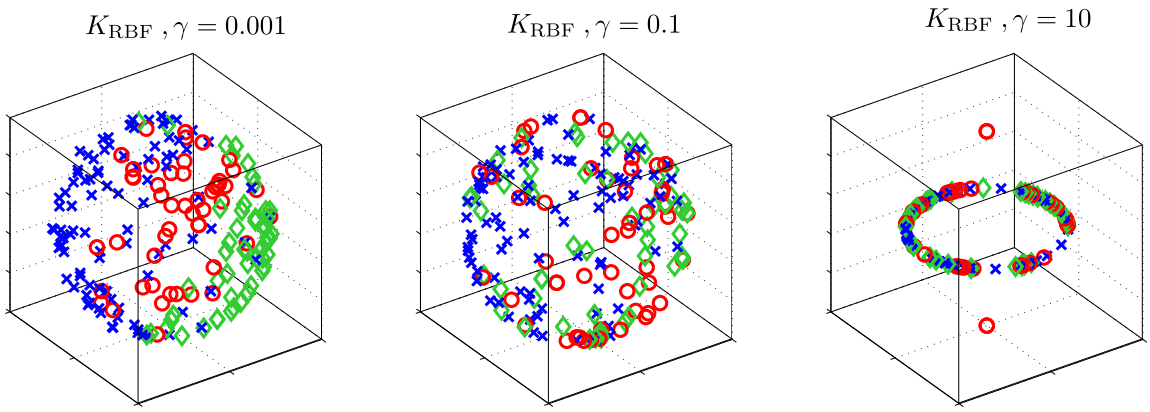

Fig. 4. Visualisation of the SJGS dataset sample in an RBF kernel space with varying values of γ; the desired class membership for a given point is indicated by marking it with a red circle,blue cross, or a green diamond.

TABLE I
RESULTS OF THE CLASSIFICATION OF SJGS DATASET USING AN SVM CLASSIFIER WITH DIFFERENT KERNEL FUNCTIONS

Kernel	K_{RBF} $\gamma = 0.001$	K_{RBF} $\gamma = 0.1$	K_{RBF} $\gamma = 10$	K_{LIN}
Train error (%)	0.10	0.00	0.00	0.00
Test error (%)	6.9	42.6	45.0	7.9
Number of support vectors	356	947	947	326

a more appropriate internal representation with respect to class separation than the other two.

This is confirmed by the results of the SVM classification shown in Table I. The table gives the test and train error, which relate the percentage of points misclassified by the classifier out of the entire train and test datasets respectively. The number of support vectors relates the complexity of the model – the fewer support vectors, the less complex the classifier.

Despite a tiny train error, the SVM classifier with the RBF kernel with $\gamma = 0.001$ gives the lowest test error, hence the best generalisation. In addition, it uses the least amount of support vectors (as compared to other RBF kernels). This means that RBF kernel with $\gamma = 0.001$ does not lead the SVM to overtrain and is indeed the best choice, as suggested by the visualisations, of the three RBF kernels examined.

We can go even further. Relying on the visualisations of the kernel transformations in Figure 1, we can infer that the smaller the value of γ in the RBF kernel, the less severe the distortion of the original data. Thus for $\gamma = 0.001$, the transformation into the kernel space must be nearly linear. This prompts us to test the SVM classification of the SJGS dataset with a linear kernel,

$$K_{\mathrm{LIN}}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j. \tag{8}$$

The performance of the classifier with this kernel is shown in the last column of Table I. It does almost as well as K_{RBF} with $\gamma = 0.001$, beating it on simplicity (in terms of support vectors used), but losing slightly on the generalisation.

IV. CONCLUSION

We have presented a technique for the visualisation of data distribution in kernel spaces. This method combines the kernel distance measure with dimension reduction algorithms that approximate the geometry of a dataset by preserving pairwise distances. These visualisations give an intuitive idea of what happens to data when it undergoes a non-linear transformation through a given kernel function. Although perhaps not all that practical for use in the state of the art classification of very large and complex datasets, nevertheless they do offer insight into the world of abstract kernel spaces, and should prove useful for educational purposes.

REFERENCES

[1] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
[2] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.
[3] B. Schölkopf, C. J. C. Burges, and A. J. Smola, *Advances in Kernel Methods: Support Vector Learning*. The MIT Press, 1998.
[4] J. Camargo, J. Caicedo, and F. González, "Kernel-based visualization of large collections of medical images involving domain knowledge," in *X Congreso Internacional de Interaccin Persona-Ordenador*, 2009.
[5] J. M. Phillips and S. Venkatasubramanian, "A gentle introduction to the kernel distance," *CoRR*, vol. abs/1103.1625, 2011.
[6] S. Joshi, R. V. Kommaraji, J. M. Phillips, and S. Venkatasubramanian, "Comparing distributions and shapes using the kernel distance," in *Proceedings of the 27th annual ACM symposium on Computational geometry*, ser. SoCG '11. New York, NY, USA: ACM, 2011, pp. 47–56.
[7] I. Borg and P. J. F. Groenen, *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2010.
[8] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
[9] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," 2001. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm
[10] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml/