Kernel and Nonlinear Canonical Correlation Analysis

Pei Ling Lai and Colin Fyfe
Applied Computational Intelligence Research Unit
Department of Computing and Information Systems
The University of Paisley, Scotland
peiling.lai@paisley.ac.uk, colin.fyfe@paisley.ac.uk

Abstract

We have previously [4] derived a neural network implementation of the statistical technique of Canonical Correlation Analysis (CCA). We extend this to nonlinear CCA either by adding a nonlinearity to our neural method or by nonlinearly transforming the data to a feature space and then performing linear CCA in this feature space. We give comparative results on both artificial and real data sets.

1 Introduction

We have previously [2] introduced an artificial neural network which finds canonical correlations between two data sets: Canonical Correlation Analysis (CCA) ([3],p281) is used when we have two data sets which we believe have some underlying correlation. Consider two sets of input data, from which we draw iid samples to form a pair of input vectors, $\mathbf{x}_1$ and $\mathbf{x}_2$. Then in classical CCA, we attempt to find the linear combination of the variables which gives us maximum correlation between the combinations. Let

$$y_1 = \mathbf{w}_1\mathbf{x}_1 = \sum_j w_{1j}x_{1j} \quad (1)$$

$$y_2 = \mathbf{w}_2\mathbf{x}_2 = \sum_j w_{2j}x_{2j} \quad (2)$$

Then we wish to find those values of $\mathbf{w}_1$ and $\mathbf{w}_2$ which maximise the correlation between $\mathbf{y}_1$ and $\mathbf{y}_2$. We must have some constraint on the maximisation otherwise the weights could simply increase without bounds; typically, we limit the variance of $\mathbf{y}_1$ and $\mathbf{y}_2$. Let $\mathbf{x}_1$ have mean μ_1 and $\mathbf{x}_2$ have mean μ_2. Then the standard method (see [3],p281) uses

$$\begin{aligned}\Sigma_{11} &= E\{(\mathbf{x}_1-\mu_1)(\mathbf{x}_1-\mu_1)^T\}\\ \Sigma_{22} &= E\{(\mathbf{x}_2-\mu_2)(\mathbf{x}_2-\mu_2)^T\}\\ \Sigma_{12} &= E\{(\mathbf{x}_1-\mu_1)(\mathbf{x}_2-\mu_2)^T\}\\ \text{and } K &= \Sigma_{11}^{-\frac{1}{2}}\Sigma_{12}\Sigma_{22}^{-\frac{1}{2}} \quad (3)\end{aligned}$$

where T denotes the transpose of a vector. We then perform a Singular Value Decomposition of

$$K = (\alpha_1, \alpha_2, ..., \alpha_k)D(\beta_1, \beta_2, ..., \beta_k)^T \quad (4)$$

where α_i and β_i are the standardised eigenvectors of KK^T and K^TK respectively and D is the diagonal matrix of eigenvalues. Then the first canonical correlation vectors (those which give greatest correlation) are given by

$$\mathbf{w}_1 = \Sigma_{11}^{-\frac{1}{2}}\alpha_1 \quad (5)$$

$$\mathbf{w}_2 = \Sigma_{22}^{-\frac{1}{2}}\beta_1 \quad (6)$$

0-7695-0619-4/00 $10.00 © 2000 IEEE

with subsequent canonical correlation vectors defined in terms of the subsequent eigenvectors, α_i and β_i.

In this paper, we review a neural implementation of Canonical Correlation Analysis, extend it to nonlinear CCA (NLCCA) and then derive a Kernel based version of CCA (KCCA). We compare these methods on artificial and real data sets.

2 The Canonical Correlation Network

Consider the input data as two vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ which are fed forward through weights, $\mathbf{w}_1$ and $\mathbf{w}_2$ to give outputs $\mathbf{y}_1$ and $\mathbf{y}_2$. We wish to maximise the correlation $E(\mathbf{y}_1\mathbf{y}_2)$ where $E()$ denotes the expectation which will be taken over the joint distribution of $\mathbf{x}_1$ and $\mathbf{x}_2$. As above in standard CCA, we add the constraint that $E(y_1^2 = 1)$ and similarly with y_2 . Using the method of Lagrange multipliers, this yields the constrained optimisation function,

$$J = E\{(y_1y_2) + \frac{1}{2}\lambda_1(1-y_1^2) + \frac{1}{2}\lambda_2(1-y_2^2)\}$$

We wish to find the optimal solution using gradient ascent and so we find the derivative of the instantaneous version of this function with respect to both the weights, $\mathbf{w}_1$ and $\mathbf{w}_2$, and the Lagrange multipliers, λ_1 and λ_2. By changing the Lagrange multipliers in proportion to the derivates of J we are changing the relative strength of the constraint compared to the function we are optimising; this allows us to smoothly maximise that function in the region in which we are satisfying the constraint. We thus derive the rules [4]

$$\begin{aligned} \Delta w_{1j} &= \eta x_{1j}(y_2 - \lambda_1 y_1) \\ \Delta\lambda_1 &= \eta_0(1-y_1^2) \end{aligned} \tag{7}$$

and similarly with $\mathbf{w}_2$ and λ_2. We have reported simulations confirming that the above rules do in fact find those linear filters of greatest correlation. However it is difficult to justify the use of linear CCA neural networks (as with linear PCA networks) since there are more efficient non-neural methods which can be used for this task. We therefore consider first a nonlinear network and then a Kernel based network.

2.1 Non-linear Correlations

Now consider another pair of outputs, $\mathbf{y}_1$ and $\mathbf{y}_2$ which are calculated from

$$y_1 = \textstyle\sum_j w_{1j}\tanh(v_{1j}x_{1j}) = \mathbf{w}_1.\mathbf{f}_1 \text{ and} \tag{8}$$

$$y_2 = \textstyle\sum_j w_{2j}\tanh(v_{2j}x_{2j}) = \mathbf{w}_2\mathbf{f}_2 \tag{9}$$

This gives us another pair of adaptable parameters, which may be changed to maximise the correlation. In particular, since the weights, $\mathbf{v}_1$ and $\mathbf{v}_2$ are used prior to the use of the nonlinearity, this gives us extra flexibility in maximising correlations. Using

$$\begin{aligned} \frac{\partial J_1}{\partial \mathbf{w}_1} &= \mathbf{f}_1 y_2 - \lambda_1 y_1 \mathbf{f}_1 = \mathbf{f}_1(y_2 - \lambda_1 y_1) \\ \frac{\partial J_1}{\partial \mathbf{v}_1} &= \mathbf{w}_1 y_2(1-\mathbf{f}_1^2)\mathbf{x}_1 - \lambda_1\mathbf{w}_1(1-\mathbf{f}_1^2)\mathbf{x}_1 y_1 \\ &= \mathbf{w}_1(1-\mathbf{f}_1^2)\mathbf{x}_1(y_2-\lambda_1 y_1) \end{aligned} \tag{10}$$

and similarly with respect to $\mathbf{w}_2$, $\mathbf{v}_2$, and λ_2, we derive

$$\begin{aligned} \Delta\mathbf{w}_1 &= \eta\mathbf{f}_1(y_2-\lambda_1 y_1) \\ \Delta\mathbf{v}_{1i} &= \eta\mathbf{x}_{1i}\mathbf{w}_{1i}(y_2-\lambda_1 y_1)(1-\mathbf{f}_1^2) \end{aligned} \tag{11}$$

We will later show that this rule can find correlations not accessible by the linear rule.

2.2 Kernel Principal Component Analysis

Kernel methods are a recent innovation based on the methods developed for Support Vector Machines. To introduce the use of Kernel methods, we review Kernel Principal Component Analysis [1] which is the most common Kernel method in unsupervised learning and also a method which will underly our Kernel Canonical Correlation Analysis.

PCA finds the eigenvectors and corresponding eigenvalues of the covariance matrix of a data set. Let $\chi = \mathbf{x}_1, ..., \mathbf{x}_M$ be iid (independent, identically distributed) samples drawn from a data source. If each $\mathbf{x}_i$ is n-dimensional, $\exists$ at most n eigenvalues/eigenvectors. Let C be the covariance matrix of the data set; then C is n*n. Then the eigenvectors, $\mathbf{e}_i$ are n dimensional vectors which are found by solving

$$C\mathbf{e} = \lambda \mathbf{e} \tag{12}$$

where λ is the eigenvalue corresponding to $\mathbf{e}$. We will assume the eigenvalues and eigenvectors are arranged in non-decreasing order of eigenvalues and each eigenvector is of length 1. We will use the sample covariance matrix as though it was the true covariance matrix and so

$$C \doteq \frac{1}{M}\sum_{j=1}^{M} \mathbf{x}_j \mathbf{x}_j^T \tag{13}$$

Now each eigenvector lies in the span of χ; i.e. the set $\chi = \mathbf{x}_1, ..., \mathbf{x}_M$ forms a basis set (normally overcomplete since $M > n$) for the eigenvectors. So each $\mathbf{e}_i$ can be expressed as

$$\mathbf{e}_i = \sum_j \alpha_{ij} \mathbf{x}_j \tag{14}$$

Now if we wish to find the principal components of a new data point, $\mathbf{x}$, we project it on the eigenvectors previously found: the first principal component is $\mathbf{x}.\mathbf{e}_1$, the second is $\mathbf{x}.\mathbf{e}_2$ etc. These are the coordinates of $\mathbf{x}$ in the eigenvector basis. There are only n eigenvectors (at most) and so there can only be n coordinates in the new system: we have merely rotated the data set. Now consider projecting one of the data points from χ on the eigenvector $\mathbf{e}_i$:

$$\mathbf{x}_k.\mathbf{e}_1 = \mathbf{x}_k.\sum_j \alpha_{1j}\mathbf{x}_j = \alpha_1.\sum_j \mathbf{x}_k \mathbf{x}_j \tag{15}$$

Now let $\mathbf{K}$ be the matrix of dot products. Then $K_{ij} = \mathbf{x}_i \mathbf{x}_j$. Multiplying both sides of (12) by $\mathbf{x}_k$ we get

$$\mathbf{x}_k C \mathbf{e}_1 = \lambda \mathbf{e}_1.\mathbf{x}_k \tag{16}$$

and using the expansion for $\mathbf{e}_1$ and the definition of the sample covariance matrix, C, gives

$$\frac{1}{M} K^2 \alpha_1 = \lambda_1 K \alpha_1 \tag{17}$$

It can be shown that the solutions of this equation are the eigenvectors of the K matrix. Now this so far only provides an alternative means of calculating principal components. However we now preprocess the data using a $\Phi : \chi \to F$. So F is now the space (the feature space) spanned by $\phi(\mathbf{x}_1), ..., \phi(\mathbf{x}_M)$. The above arguments all hold for $K_{ij} = \phi(\mathbf{x}_i)\phi(\mathbf{x}_j)$. But now the Kernel Trick: provided we can calculate K we don't need the individual terms, $\phi(\mathbf{x}_1), ..., \phi(\mathbf{x}_M)$. And the above argument shows that any operation which can be defined in terms of dot products can be Kernelised. We thus Kernelise CCA.

2.3 Kernel Canonical Correlation Analysis

Now the standard statistical method of performing CCA relies wholly on dot products (3)-(6). We therefore repeat these dot products but now in feature space: we define

$$\begin{aligned}
(\Sigma_{11})_{ij} &= K(\mathbf{x}_{1i}, \mathbf{x}_{1j}) = \exp(-(\mathbf{x}_{1i} - \mathbf{x}_{1j})^2/\sigma^2) \\
(\Sigma_{22})_{ij} &= K(\mathbf{x}_{2i}, \mathbf{x}_{2j}) = \exp(-(\mathbf{x}_{2i} - \mathbf{x}_{2j})^2/\sigma^2) \\
(\Sigma_{12})_{ij} &= K(\mathbf{x}_{1i}, \mathbf{x}_{2j}) = \exp(-(\mathbf{x}_{1i} - \mathbf{x}_{2j})^2/\sigma^2)
\end{aligned}$$

where now $K(ab) = \phi(a)\phi(b)$ forms a nonlinear mapping to some unknown feature space. The Kernel trick allows us to operate in the feature space even though we do not know the mapping $\phi()$. We centre the ensure data points using the online algorithm described in [1] and then calculate the matrix

$$K = \Sigma_{11}^{-\frac{1}{2}}\Sigma_{12}\Sigma_{22}^{-\frac{1}{2}} \tag{18}$$

on which we perform a Singular Value Decomposition to get

$$K = (\alpha_1, \alpha_2, ..., \alpha_k)D(\beta_1, \beta_2, ..., \beta_k)^T \tag{19}$$

We note that the unknown feature space actually is formed from the particular data points chosen and the Kernel function which we use to operate operate upon them. Normally the emphasis is upon the kernel function, since we typically draw samples iid from the underlying distribution. However here we are drawing samples from two distributions and so are actually creating two different feature spaces. This will become important in the following discussion.

We may now take new points and project them upon the feature space by finding the dot product of the new points and the previous points which formed each feature space and projecting them upon their respective directions of maximum correlation (See Figure 2)

3 Simulations

3.1 Artificial Data

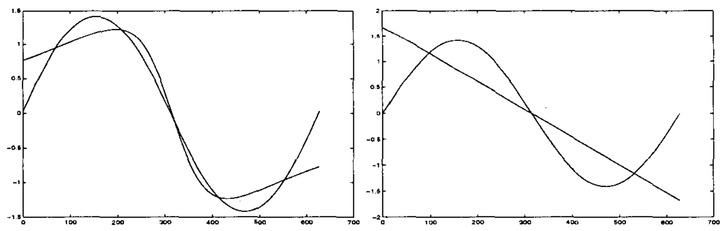

Figure 1: The left diagram shows the outputs of the nonlinear network, while the right diagram shows the outputs of the linear network. Visual inspection would suggest that the outputs from the nonlinear network are more correlated. The actual sample correlation values achieved were 0.623(linear) and 0.865(nonlinear).

We first generate two sets of two dimensional data according to the prescription:

$$(x_{11}, x_{12}) = (1 - \sin\theta + \mu_1, cos\theta + \mu_2) \tag{20}$$

$$(x_{21}, x_{22}) = (\theta + \mu_3, \theta + \mu_4) \tag{21}$$

where θ is drawn from a uniform distribution in $[-\pi, \pi]$ and $\mu_i, i = 1, ..., 4$ are drawn from the zero mean Gaussian distribution N(0, 0.1). Equation (20) defines a circular manifold in the two dimensional input space while equation (21) defines a linear manifold within the input space where each manifold is only approximate due to the presence of noise ($\mu_{i,i=1,...,4}$). Thus $\mathbf{x}_1 = \{x_{11}, x_{12}\}$ lies on or near the circular manifold $x_{11}^2 + x_{12}^2 = 1$ while $\mathbf{x}_2 = \{x_{21}, x_{22}\}$ lies on or near the line $\mathbf{x}_{21} = \mathbf{x}_{22}$.

We wish to compare the linear CCA network, with the nonlinear CCA network and the Kernel CCA method.

Standard Statistics : Maximum correlation	0.6630		
$\mathbf{w}_1$	0.0260	0.0518	
$\mathbf{w}_2$	0.0824	0.0081	0.0035
Neural Network:Maximum correlation	0.6962		
$\mathbf{w}_1$	0.0264	0.0526	
$\mathbf{w}_2$	0.0829	0.0098	0.0041

Table 1: The converged weights from the neural network are compared with the values reported from a standard statistical technique [3].

Thus we train one pair of weights $\mathbf{w}_1$ and $\mathbf{w}_2$ using rules (7)(the linear network), a second pair use the nonlinear rules (11), and the final method is the KCCA.

For the neural methods, we use a learning rate of 0.001 for all weights and learn over 100000 iterations. The network finds a sample linear correlation between the data sets of 0.623 (equal to the correlation between $\mathbf{y}_1$ and $\mathbf{y}_2$ while the nonlinear neurons have a sample correlation of 0.865. In putting the data sets through the nonlinear tanh() function, we have created a relationship whose correlations are greater than the linear correlations of the original data set. We show a test of the outputs from both the linear and nonlinear networks in Figure 1 in which we graph the output values of the trained network from each pair of neurons against inputs where the θ values in equations (20)- (21) range from -3 to 3. We see that the linear network is aligning the outputs as well as may be expected but the nonlinear network's outputs are very closely aligned with each other over much more of the data set.

The results from a Kernel CCA networks with rbf Gaussian kernels on the same data is shown in the left part of Figure 2: the solid contours are contours of equal correlation (with points on the line) in the space defined by points on (or near) the circle; the dotted lines are contours of equal correlation (with points on the circle) in the space defined by points on or near the line. While the results are not so easily interpreted as before, we see that areas of high correlation tend to be local areas in each space.

3.2 Real data

Our second experiment uses a data set reported in [3],p281; it comprises 88 students' marks on 5 module exams. The exam results can be partitioned into two data sets: two exams were given as close book exams (C) while the other three were opened book exams (O). The exams were on the subjects of Mechanics(C), Vectors(C), Algebra(O), Analysis(O), and Statistics(O). Thus, in our first simulations, we split the five variables (exam marks) into two sets - the closed-book exams and the opened-book exams . One possible quantity of interest here is how highly a student's ability on closed-book exams is correlated with his ability on open-book exams. Alternatively, one might try to use the open-book exam results to predict the closed-book results (or vice versa). The responses of the linear and nonlinear network sare shown in Table 1. The correlations found by the kernel method are shown in Figure 2; again we see that radial kernels give us a local area of maximal correlation within the data set.

4 Discussion

We have extended a neural implementation of Canonical Correlation Analysis and by introducing a non-linearity have shown that the neural method can find correlations which linear CCA cannot find. We have also introduced Kernel CCA and compared the methods on both real and artificial data sets. It is possible to use the tanh() function as a kernel so giving us equivalent methods in both domains. Comparative studies of these are ongoing.

However the Kernel method was noted to be wanting in two respects:

- The method requires a dot product to be performed between the data points in classes 1 and 2. Thus we require to equalise the number of elements in the two data samples. In this data set, we did this by merely using 2 of the three results in the closed book examination data set.

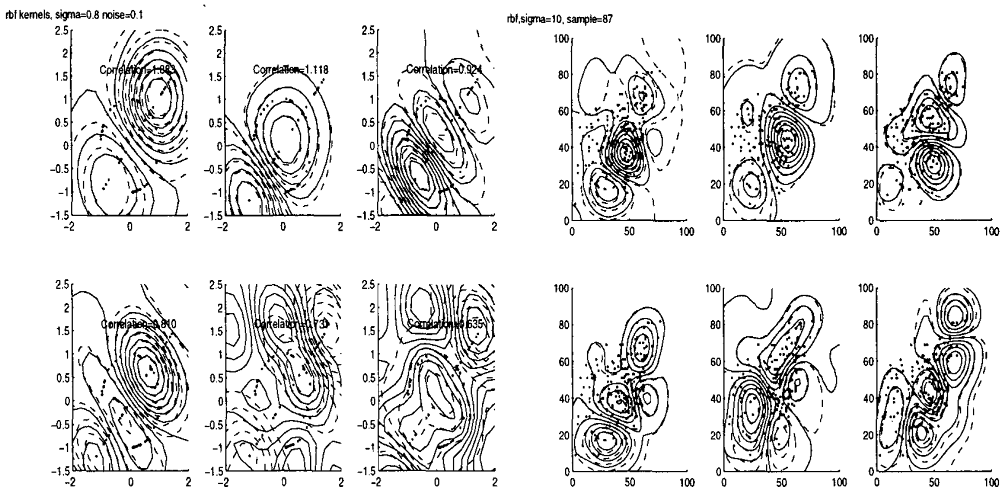

Figure 2: The first 6 directions of principal correlation are shown here: in the left diagram, all points on each contour have equal correlation with the points on the line. In the right diagram, the solid lines show lines of equal correlation with the second data set; the dashed lines have points of equal correlation with the first data set.

- Figures for correlations are not useful for comparison with the other methods. We see that correlations can be extremely high locally; indeed, there are occasions when, because we are using structures in two different spaces, we find correlations which exceed 1.
- The method is prone to singularities: if two students have the same values in their exams, then σ_{11} and/or σ_{22}will be singular with an obvious concomitant effect on (18).

Future work will investigate alternative kernels and nonlinearities.

References

[1] C. J. C. Burges P. Knirsch K-R. Muller G. Ratsch B. Scholkopf, S. Mika and A. J. Smola. Input space vs feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10:1000–1017, 1999.

[2] P. L. Lai and C. Fyfe. Canonical correlation analysis using artificial neural networks. In *European Symposium on Artificial Neural Networks, ESANN98*, 1998.

[3] K. V. Mardia, J.T. Kent, and J.M. Bibby. *Multivariate Analysis*. Academic Press, 1979.

[4] P.L.Lai and C. Fyfe. A neural network implementation of canonical correlation analysis. *Neural Networks*, 12(10):1391–1397, Dec. 1999.