International Journal of Neural Systems, Vol. 10, No. 5 (2000) 365–377
© World Scientific Publishing Company

KERNEL AND NONLINEAR CANONICAL CORRELATION ANALYSIS

P. L. LAI and C. FYFE
Applied Computational Intelligence Research Unit,
The University of Paisley, Scotland

Received 10 April 2000
Revised 4 August 2000
Accepted 23 September 2000

We review a neural implementation of the statistical technique of Canonical Correlation Analysis (CCA) and extend it to nonlinear CCA. We then derive the method of kernel-based CCA and compare these two methods on real and artificial data sets before using both on the Blind Separation of Sources.

1. Introduction

We have previously introduced[5,4] a neural implementation of Canonical Correlation Analysis (CCA). In this paper, we compare two extensions to the method:

1. We first consider a nonlinear extension to the neural method and show how it performs Nonlinear CCA.
2. We then use the kernel methods popularised by Support Vector Machines (SVMs) to create a Kernel CCA method.

Canonical Correlation Analysis[7] 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

$$y_1 = \mathbf{w}_1\mathbf{x}_1 = \sum_j w_{1j}x_{1j} \tag{1}$$

$$y_2 = \mathbf{w}_2\mathbf{x}_2 = \sum_j w_{2j}x_{2j} \tag{2}$$

Then we wish to find those values of $\mathbf{w}_1$ and $\mathbf{w}_2$ which maximise the correlation between y_1 and y_2. We must constrain the solutions to ensure a finite solution and so we ensure that the variance of y_1 and y_2 are both 1.

Let $\mathbf{x}_1$ have mean μ_1 and $\mathbf{x}_2$ have mean μ_2. Then the standard statistical method (see Ref. 7) lies in defining

$$\Sigma_{11} = E\{(\mathbf{x}_1 - \mu_1)(\mathbf{x}_1 - \mu_1)^T\} \tag{3}$$

$$\Sigma_{22} = E\{(\mathbf{x}_2 - \mu_2)(\mathbf{x}_2 - \mu_2)^T\} \tag{4}$$

$$\Sigma_{12} = E\{(\mathbf{x}_1 - \mu_1)(\mathbf{x}_2 - \mu_2)^T\} \tag{5}$$

$$\text{and} \quad K = \Sigma_{11}^{-\frac{1}{2}}\Sigma_{12}\Sigma_{22}^{-\frac{1}{2}} \tag{6}$$

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

$$K = (\alpha_1, \alpha_2, \ldots, \alpha_k)D(\beta_1, \beta_2, \ldots, \beta_k)^T \tag{7}$$

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

$$\mathbf{w}_1 = \Sigma_{11}^{-\frac{1}{2}}\alpha_1 \tag{8}$$

$$\mathbf{w}_2 = \Sigma_{22}^{-\frac{1}{2}}\beta_1 \tag{9}$$

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

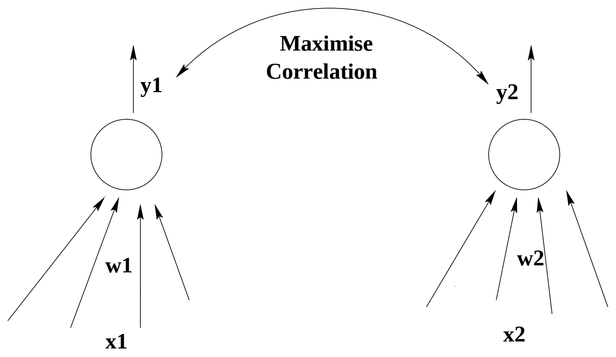

Fig. 1. The CCA Network. By adjusting weights, w_1 and w_2, we maximise correlation between y_1 and y_2.

2. The Canonical Correlation Network

Let us consider CCA in artificial neural network terms. The input data comprises two vectors $\mathbf{x}_1$ and $\mathbf{x}_2$. Activation is fed forward from each input to the corresponding output through the respective weights, $\mathbf{w}_1$ and $\mathbf{w}_2$ (see Fig. 1 and Eqs. (1) and (2)) to give outputs y_1 and y_2.

We have previously[5] derived an objective function for the maximisation of this correlation under the constraint that the variance of y_1 and y_2 should be 1 as

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

where the constraints have been implemented using Lagrange multipliers, λ_1 and λ_2.

Using gradient ascent with respect to $\mathbf{w}_i$ and descent with respect to λ_i gives the learning rules

$$\begin{aligned} \Delta w_{1j} &= \eta x_{1j}(y_2 - \lambda_1 y_1) \\ \Delta\lambda_1 &= -\eta_0(1-y_1^2) \\ \Delta w_{2j} &= \eta x_{2j}(y_1 - \lambda_2 y_2) \\ \Delta\lambda_2 &= -\eta_0(1-y_2^2) \end{aligned} \tag{10}$$

where w_{1j} is the jth element of weight vector, $\mathbf{w}_1$ etc.

However, just as a neural implementation of Principal Component Analysis (PCA) e.g. Ref. 9 may be very interesting but generally not a useful method of finding Principal Components, so a neural implementation of CCA may be only a curiosity. We therefore extend this to nonlinear CCA.[10,3]

3. Nonlinear Canonical Correlation Analysis

We wish to test whether the network can find nonlinear correlations between the two data sets, $\mathbf{x}_1$ and $\mathbf{x}_2$, and test whether such correlations are greater than the maximum linear correlations. Therefore we introduce a nonlinearity to our network (a tanh( ) function). This is a somewhat more *ad hoc* procedure than that used to derive Kernel CCA. Thus we calculate y_1 and y_2 using

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

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

To maximise the correlation between y_1 and y_2, we again use the objective function

$$J_1 = E\left\{(y_1y_2) + \frac{1}{2}\lambda_1(1-y_1^2) + \frac{1}{2}\lambda_2(1-y_2^2)\right\}$$

whose derivatives give us

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

where in this last equation, all the vector multiplications are to be understood as being on an element by element basis. Similarly with $\mathbf{w}_2$, $\mathbf{v}_2$, and λ_2. This gives us a method of changing the weights and the Lagrange multipliers on an online basis. We therefore have the joint learning rules

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

and similarly with the second set of weights.

4. Kernel Canonical Correlation Analysis

In this section, we extend CCA by nonlinearly transforming the data to a feature space and then performing linear CCA in this feature space.

Unsupervised Kernel methods are a recent innovation based on the methods developed for Support Vector Machines.[15,1] Support Vector regression for example, performs a nonlinear mapping of the data set into some high-dimensional feature space in which we may then perform linear operations. Since the original mapping was nonlinear, any linear operation in this feature space corresponds to a nonlinear operation in data space. The most popular unsupervised kernel method to date has been Kernel Principal Component Analysis.[13]

4.1. *Kernel canonical correlation analysis*

Consider mapping the input data to a high-dimensional (perhaps infinite-dimensional) feature space, F. Now the covariance matrices in feature space are defined by

$$\Sigma_{11} = E\{(\Phi(\mathbf{x}_1) - \mu_1)(\Phi(\mathbf{x}_1) - \mu_1)^T\}$$

$$\Sigma_{22} = E\{(\Phi(\mathbf{x}_2) - \mu_2)(\Phi(\mathbf{x}_2) - \mu_2)^T\}$$

$$\Sigma_{12} = E\{(\Phi(\mathbf{x}_1) - \mu_1)(\Phi(\mathbf{x}_2) - \mu_2)^T\}$$

where now $\mu_i = E(\Phi(\mathbf{x}_i))$ for $i = 1, 2$. Let us assume for the moment that the data has been centred in feature space (we actually will use the same trick as Ref. 12 to centre the data later(see Appendix A)). Then we define

$$\Sigma_{11} = E\{\Phi(\mathbf{x}_1)\Phi(\mathbf{x}_1)^T\}$$

$$\Sigma_{22} = E\{\Phi(\mathbf{x}_2)\Phi(\mathbf{x}_2)^T\}$$

$$\Sigma_{12} = E\{\Phi(\mathbf{x}_1)\Phi(\mathbf{x}_2)^T\}$$

and we wish to find those values $\mathbf{w}_1$ and $\mathbf{w}_2$ which will maximise $\mathbf{w}_1^T\Sigma_{12}\mathbf{w}_2$, subject to the constraints $\mathbf{w}_1^T\Sigma_{11}\mathbf{w}_1 = 1$ and $\mathbf{w}_2^T\Sigma_{22}\mathbf{w}_2 = 1$.

In practise we will approximate Σ_{12} with $\frac{1}{M}\sum_i \Phi(\mathbf{x}_{1i})\Phi(\mathbf{x}_{2i})$, the sample average and similarly with Σ_{11} and Σ_{22}.

At this stage we can see the similarity with our nonlinear CCA: if we consider an instantaneous gradient-based algorithm, we would derive precisely our NLCCA algorithm for the particular nonlinearity involved. However the kernel methods adopt a different approach.

Now an alternative method[11] of finding the canonical correlation directions is to solve the generalised eigenvalue problem

$$\begin{bmatrix} 0 & \Sigma_{12} \\ \Sigma_{21} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \end{bmatrix} = \rho \begin{bmatrix} \Sigma_{11} & 0 \\ 0 & \Sigma_{22} \end{bmatrix} \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \end{bmatrix} \tag{11}$$

and so we may use the arguments of Ref. 13 to show that $\mathbf{w}_1$ and $\mathbf{w}_2$ exist in the feature space which is spanned by $\{\Phi(\mathbf{x}_{11}), \Phi(\mathbf{x}_{12}), \ldots, \Phi(\mathbf{x}_{1M}), \Phi(\mathbf{x}_{21}), \ldots, \Phi(\mathbf{x}_{2M})\}$ and therefore can be expressed as

$$\mathbf{w}_1 = \sum_{i=1}^{M} \alpha_{1i}\Phi(\mathbf{x}_{1i}) + \sum_{i=1}^{M} \alpha_{2i}\Phi(\mathbf{x}_{2i})$$

$$\mathbf{w}_2 = \sum_{i=1}^{M} \beta_{1i}\Phi(\mathbf{x}_{1i}) + \sum_{i=1}^{M} \beta_{2i}\Phi(\mathbf{x}_{2i})$$

With some abuse of the notation we will use $\mathbf{x}_i$ to be the ith instance from the set of data i.e. from either the set of values of $\mathbf{x}_1$ or from those of $\mathbf{x}_2$ and write

$$\mathbf{w}_1 = \sum_{i=1}^{2M} \alpha_i\Phi(\mathbf{x}_i)$$

$$\mathbf{w}_2 = \sum_{i=1}^{2M} \beta_i\Phi(\mathbf{x}_i)$$

Therefore substituting this in the criteria we wish to optimise, we get

$$(\mathbf{w}_1^T\Sigma_{12}\mathbf{w}_2) = \frac{1}{M}\sum_{k,i} \alpha_k \cdot \Phi^T(\mathbf{x}_k)\Phi(\mathbf{x}_{1i}) \sum_{l} \beta_l \Phi^T(\mathbf{x}_{2i})\Phi(\mathbf{x}_l) \tag{12}$$

where the sums over i are to find the sample means over the data set. Similarly with the constraints and so

$$\mathbf{w}_1^T\Sigma_{11}\mathbf{w}_1 = \frac{1}{M}\sum_{k,i} \alpha_k \cdot \Phi^T(\mathbf{x}_k)\Phi(\mathbf{x}_{1i}) \cdot \sum_{l} \alpha_l \Phi^T(\mathbf{x}_{1i})\Phi(\mathbf{x}_l)$$

$$\mathbf{w}_2^T\Sigma_{22}\mathbf{w}_2 = \frac{1}{M}\sum_{k,i} \beta_k \cdot \Phi^T(\mathbf{x}_k)\Phi(\mathbf{x}_{2i}) \cdot \sum_{l} \beta_l \Phi^T(\mathbf{x}_{2i})\Phi(\mathbf{x}_l)$$

Using $(K_1)_{ij} = \Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_{1j})$ and $(K_2)_{ij} = \Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_{2j})$ we then have that we require to maximise $\alpha^T K_1 K_2^T \beta$ subject to the constraints $\alpha^T K_1 K_1^T \alpha = 1$ and $\beta^T K_2 K_2^T \beta = 1$. Therefore if we define $\Gamma_{11} = K_1 K_1^T$, $\Gamma_{22} = K_2 K_2^T$ and $\Gamma_{12} = K_1 K_2^T$ we solve the problem in the usual way: by forming matrix $K = \Gamma_{11}^{-\frac{1}{2}} \Gamma_{12} \Gamma_{22}^{-\frac{1}{2}}$ and performing a singular value decomposition on it as before to get

$$K = (\gamma_1, \gamma_2, \ldots, \gamma_k) D(\theta_1, \theta_2, \ldots, \theta_k)^T \tag{13}$$

where γ_i and θ_i are again the standardised eigenvectors of KK^T and $K^T K$ respectively and D is the diagonal matrix of eigenvalues.[a]

Then the first canonical correlation vectors in feature space are given by

$$\alpha_1 = \Gamma_{11}^{-\frac{1}{2}} \gamma_1 \tag{14}$$

$$\beta_1 = \Gamma_{22}^{-\frac{1}{2}} \theta_1 \tag{15}$$

with subsequent canonical correlation vectors defined in terms of the subsequent eigenvectors, γ_i and θ_i.

Now for any new values $\mathbf{x}_1$, we may calculate

$$\mathbf{w}_1 \cdot \Phi(\mathbf{x}_1) = \sum_i \alpha_i \Phi(\mathbf{x}_i) \Phi(\mathbf{x}_1) = \sum_i \alpha_i K_1(\mathbf{x}_i, \mathbf{x}_1) \tag{16}$$

which then requires to be centered as before. We see that we are again performing a dot product in feature space (it is actually calculated in the subspace formed from projections of $\mathbf{x}_i$).

There are four particular aspects of the above algorithm which should be pointed out:

1. The optimal weight vectors are vectors in a feature space which we may never determine. We are simply going to calculate the appropriate matrices using the kernel trick — e.g., we may use Gaussian kernels[13] so that

$$K_1(\mathbf{x}_{1i}, \mathbf{x}_{1j}) = \exp(-(\mathbf{x}_{1i} - \mathbf{x}_{1j})^2) \tag{17}$$

which gives us a means of calculating K_1 without ever having had to calculate $\Phi(\mathbf{x}_{1i})$ or $\Phi(\mathbf{x}_{1j})$ explicitly.

2. The method requires a dot product between members of the data set $\mathbf{x}_1$ and $\mathbf{x}_2$ and therefore the vectors must be of the same length. (This is not a requirement of the standard statistical method[7] nor of our neural methods (see Secs. 2 and 3).) Therefore in the exam data (see below), we must discard one set of exam marks.
3. The method requires a matrix inversion and the data sets may be such that one data point may be repeated (or almost) leading to a singularity or badly conditioned matrices. One solution (suggested to us by Darryl Charles) is to add noise to the data set; this is effective in the exam data set, is a nice solution if we were to consider biological information processors but need not always work. An alternative is to add μI, where I is the identity matrix to Γ_{11} and Γ_{22} — a method which was also used in Ref. 8. This gives robust and reliable solutions.
4. Whereas our neural implementation of nonlinear CCA is inevitably an iterative process, the kernel methods requires just one transformation into feature space followed by the standard linear operation in this space. This makes the kernel method computationally attractive.

5. Results

In this section, we compare the correlations found by these two methods on artificial and real data before using them on the problem of "Blind Separation of Sources".

5.1. *Experiment 1: simple artificial data*

Let us generate data according to the prescription:

$$x_{11} = 1 - \sin\theta + \varepsilon_1 \tag{18}$$

$$x_{12} = \cos\theta + \varepsilon_2 \tag{19}$$

$$x_{21} = \theta + \varepsilon_3 \tag{20}$$

$$x_{22} = \theta + \varepsilon_4 \tag{21}$$

where θ is drawn from a uniform distribution in $[-\pi, \pi]$ and ε_i, $i = 1, \ldots, 4$ are drawn from the zero mean Gaussian distribution N(0, 0.1). Equations (18) and (19) define a circular manifold in the

[a]This optimisation is applicable for all symmetric matrices (Theorem A.9.2, Ref. 7).

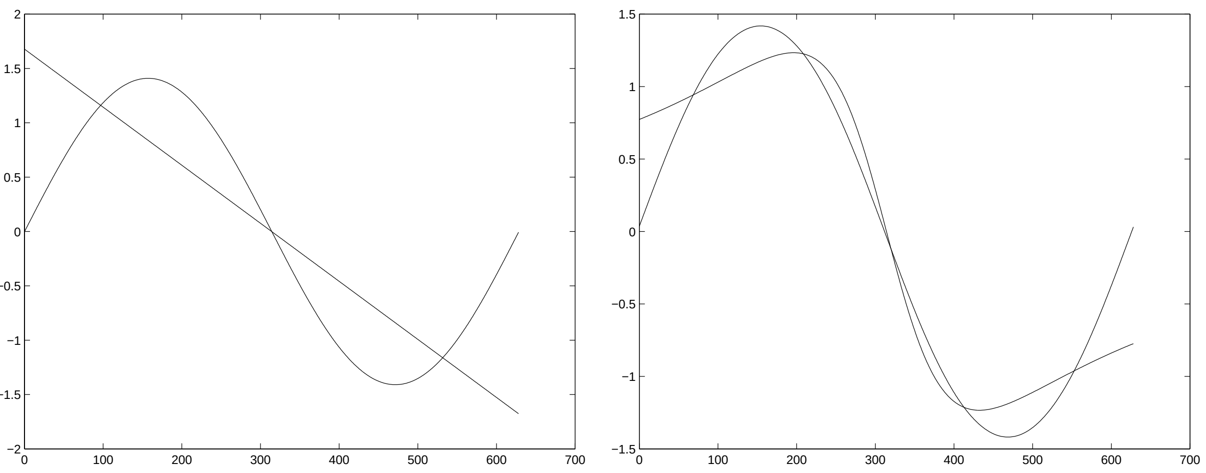

Fig. 2. The left diagram shows the outputs of the linear network, while the right diagram shows the outputs of the nonlinear network. Visual inspection would suggest that the outputs from the nonlinear network are more correlated. The actual sample correlation values achieved were 0.623 (linear) and 0.865 (nonlinear).

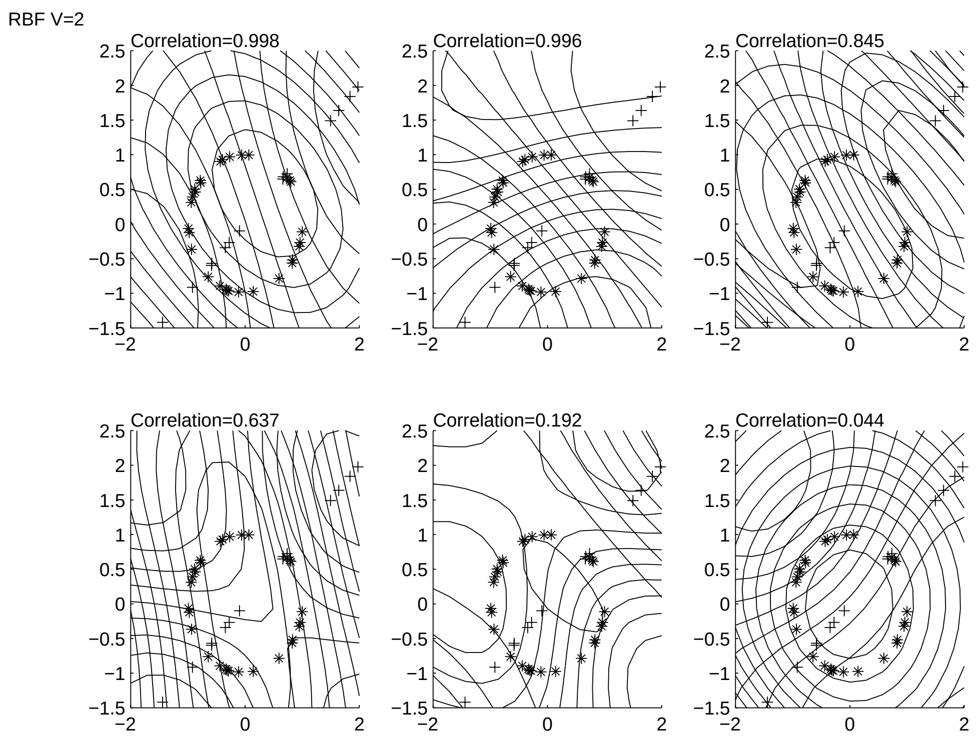

Fig. 3. Each figure shows the contours of equal correlation of each of the data sets with the points on the other. The actual correlations found are given in the title to each figure.

two-dimensional input space while Eqs. (20) and (21) define a linear manifold within the input space where each manifold is only approximate due to the presence of noise (ε_i, $i = 1, \ldots, 4$).

Thus $\mathbf{x}_1 = \{x_{11}, x_{12}\}$ lies on or near the circular manifold while $\mathbf{x}_2 = \{x_{21}, x_{22}\}$ lies on or near the line. We use a learning rate of 0.001 for all weights and learn over 100,000 iterations. The learning rates are decreased to zero during the course of the simulation.

The learned network finds a linear correlation between the data sets of 0.6233 while the nonlinear neurons have a correlation of 0.865479. Clearly by 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 Fig. 2 in which we graph the output values of the trained network from each pair of neurons against inputs where the θ values in Eqs. (18)–(21) range from -3 to 3 and, of course, we do not add noise. 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 linear network does not have enough degrees of freedom to make the line bend toward the curves. We have similar results matching data from a noisy surface of a sphere with data from a noisy plane.

The results from a Kernel CCA with Gaussian kernels on the same data is shown in Fig. 3: one set of contours are contours of equal correlation (with points on the line) in the space defined by points on (or near) the circle; the others 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. In addition, the kernel method enables us to find more than two canonical correlation filters.

5.2. *Real data*

Our second experiment uses a data set reported in Ref. 7 (p. 290); it comprises 88 students' marks on five module exams. The exam results can be partitioned into two data sets: two exams were given as closed-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). We thus split the five variables (exam marks) into two sets — the closed-book exams (x_{11}, x_{12}) and the opened-book exams (x_{21}, x_{22}, x_{23}). One possible quantity of interest here is how highly a student's ability on closed-book exams is correlated with his ability on opened-book exams. Table 1 shows the results from each of the methods and from the standard statistical method. We see that the nonlinear neural network performs slightly better than the linear network but that the kernel method performs best of all. This result however should be treated with care: firstly the kernel method requires that the length of each of the data samples is the same; we have had to remove one data column so that both $\mathbf{x}_1$ and $\mathbf{x}_2$ are the same length. Secondly and more importantly, it is possible to get a correlation of 1 on any data set using radial kernels by simply having an infinitely wide Gaussian kernel! This is clearly a non-informative solution and can be thought of as being caused by over-regularising the CCA operation. Current research into selecting reduced sets of points for Kernel PCA[14] and into the most effective kernels for any particular problem may throw light on this problem.

5.3. *The blind separation of sources*

Independence Component Analysis (ICA) networks are often thought of as extensions of PCA networks. ICA and the related problem of blind source separation have recently gained much interest[2] due to many applications. The objective of this section is to present neural networks for separation of mixed independent source signals. Let there be N independent non-Gaussian signals (s_1, $s_2, \ldots, s_N$) which are mixed using a (square) mixing matrix A to get N vectors, x_i, each of which is an unknown mixture of the independent signals,

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{22}$$

Table 1. The converged weights from the neural network and Kernel method are compared with the values reported from a standard statistical technique.[7]

Standard Statistics Maximum Correlation	0.6630
Linear Neural Network Maximum Correlation	0.6630
Nonlinear Neural Network Maximum Correlation	0.6740
Kernel CCA Maximum Correlation	0.761

Then the aim is to use an artificial neural network to retrieve the original input signals when the only information presented to the network is the unknown mixture of the signals. Note that the outputs, $\mathbf{y}$ are to be the elements of the original signal in some order i.e., we are not insisting that the first output of our neural network is equal to the first signal, the second equal to the second signal and so on. We merely insist that neuron i's output is one of the N original signals not mixed with any of the other signals. In this paper, we will consider more difficult mixtures in which the mixing is done by nonlinear means or by nonstationary matrices. Our aim is still the "blind separation of sources" — we assume no prior knowledge of the mixing process nor of the signals which are mixed.

If we wish to extract more than one signal, we may attempt to maximise the correlation between each pair of outputs in $\mathbf{y}_1$ and $\mathbf{y}_2$ (i.e., matching the elements of each in order) while at the same time ensuring that subsequent pairs of outputs are not finding the same correlations. To do this we have the constraint that all elements in each of the output vectors $\mathbf{y}_1$ and $\mathbf{y}_2$ have zero correlation with all other elements in the respective vectors while still retaining the constraint that each output should have variance 1. This gives a combined objective function of

$$J = E\left\{(\mathbf{y_1^T y_2}) + \frac{\mathbf{1}}{\mathbf{2}}\Lambda_\mathbf{1}(\mathbf{I} - \mathbf{y_1 y_1^T}) \right.$$
$$\left. + \frac{1}{2}\Lambda_2(I - \mathbf{y_2 y_2^T})\right\} \tag{23}$$

with Λ_i, $i = 1, 2$ now a matrix of Lagrange multipliers. This gives us the learning rules

$$\triangle\mathbf{w_1} = \eta\mathbf{f_1}(\mathbf{y_2} - \Lambda_\mathbf{1}\mathbf{y_1})$$
$$\triangle\mathbf{v_1} = \eta\mathbf{w_1}(\mathbf{y_2} - \Lambda_\mathbf{1}\mathbf{y_1})\mathbf{x_1}(\mathbf{1} - \mathbf{f_1^2})$$
$$\triangle\Lambda_1 = \eta_0(I - \mathbf{y_1 y_1^T})$$

where

$$\mathbf{y_1} = \mathbf{w_1^T f}(\mathbf{v_1 x_1}) \tag{24}$$

with $\mathbf{w}_1$ the m*n matrix of weights connecting $\mathbf{x_1}$ to $\mathbf{y_1}$. Similarly with $\mathbf{w}_2$, Λ_2.

This model may be viewed as an abstraction of two data streams (e.g., sight and sound) identifying an entity in the environment by identifying the maximal correlations between the data streams. We

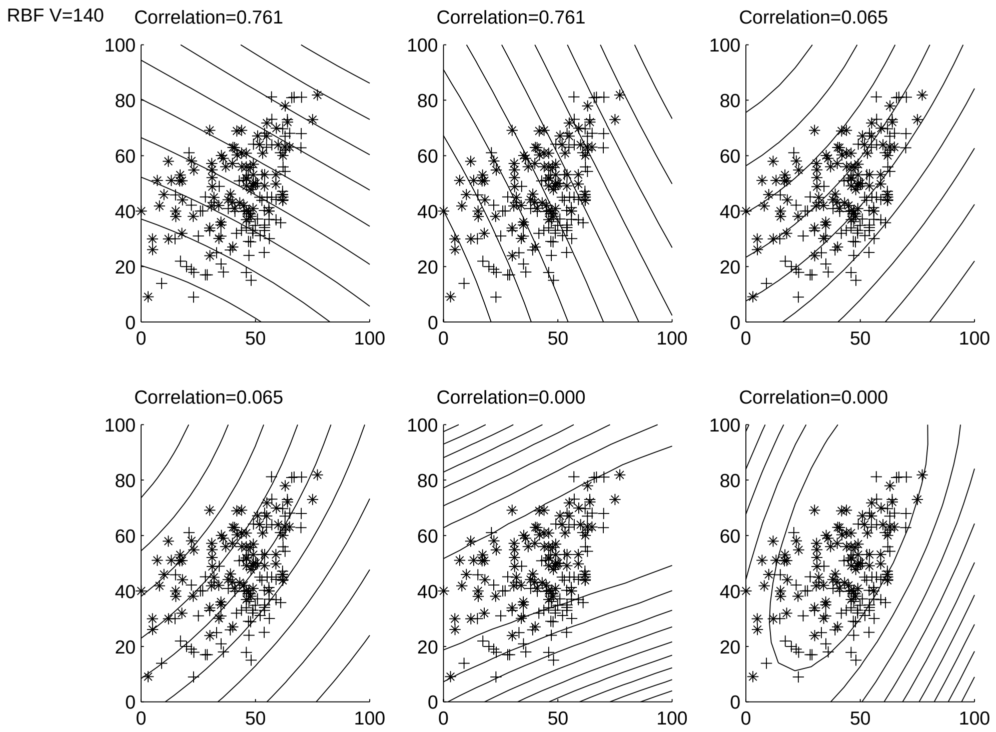

Fig. 4. The kernel canonical correlation directions found using radial kernels. The contour lines are lines of equal correlation. Each pair of diagrams shows the equal correlation contours from the perspective of one of the data sets.

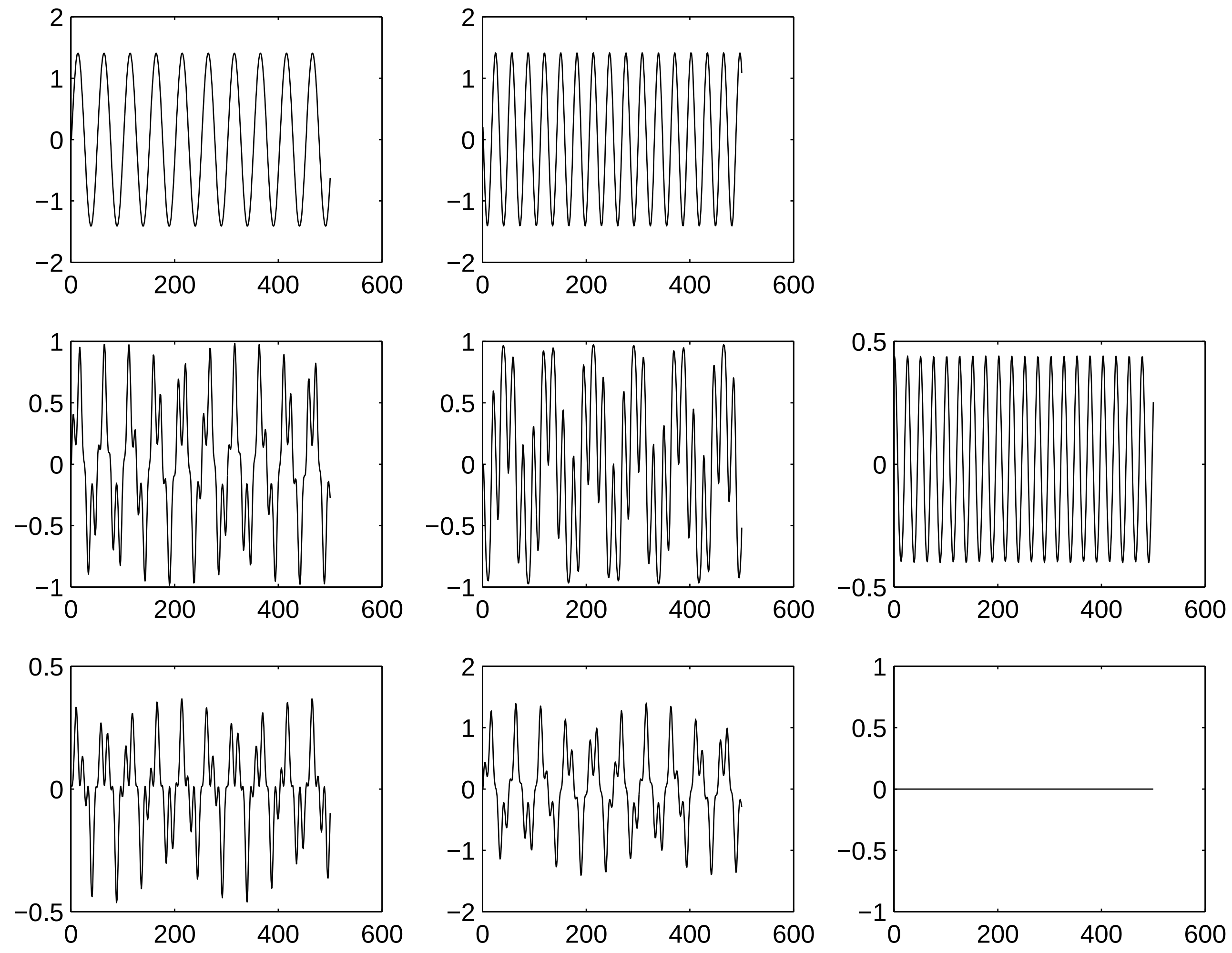

Fig. 5. The top line shows the underlying sines which we wish to recover. The second line shows the signals when they are embedded on the surface of a sphere, $\mathbf{x_1}$. The third line shows the signals as coordinates on the plane, $\mathbf{x_2}$.

have previously considered[6] an alternative use of CCA in this context: we have attempted to identify independent components of data streams by minimising the nonlinear correlations between data sets so that each output is as independent as possible from the others but we will not pursue this method in this paper.

With our neural method, we wish to emulate biological information processing in that e.g., auditory information cannot choose which pathway it uses. All early processing is done by the same neurons. In other words, these neurons see the information at each time instant. We therefore structure our data set so that

$$\mathbf{x}_1 = \{m_{1,1}(t),\, m_{1,2}(t), \ldots,\, m_{1,1}(t+1), \ldots,\, m_{1,2}(t+P_1)\}$$

$$\mathbf{x}_2 = \{m_{2,1}(t),\, m_{2,2}(t), \ldots,\, m_{2,1}(t+1), \ldots,\, m_{2,2}(t+P_2)\}$$

Good extraction of single sinusoids from a mixture of sinusoids (e.g., for $2*2$ square mixes $\mathbf{m_1}$, $\mathbf{m_2}$) have been obtained using only

$$\mathbf{x}_1 = \{m_{1,1}(t),\, m_{1,2}(t),\, m_{1,1}(t+P_1),\, m_{1,2}(t+P_1)\}$$
$$\mathbf{x}_2 = \{m_{2,1}(t),\, m_{2,2}(t),\, m_{2,1}(t+P_2),\, m_{2,2}(t+P_2)\}$$

where P_1 and P_2 are the time delays which need not be equal.

To create a data set, we have embedded two sine waves on the surface of a sphere and on the plane (recall that we previously were able to maximise

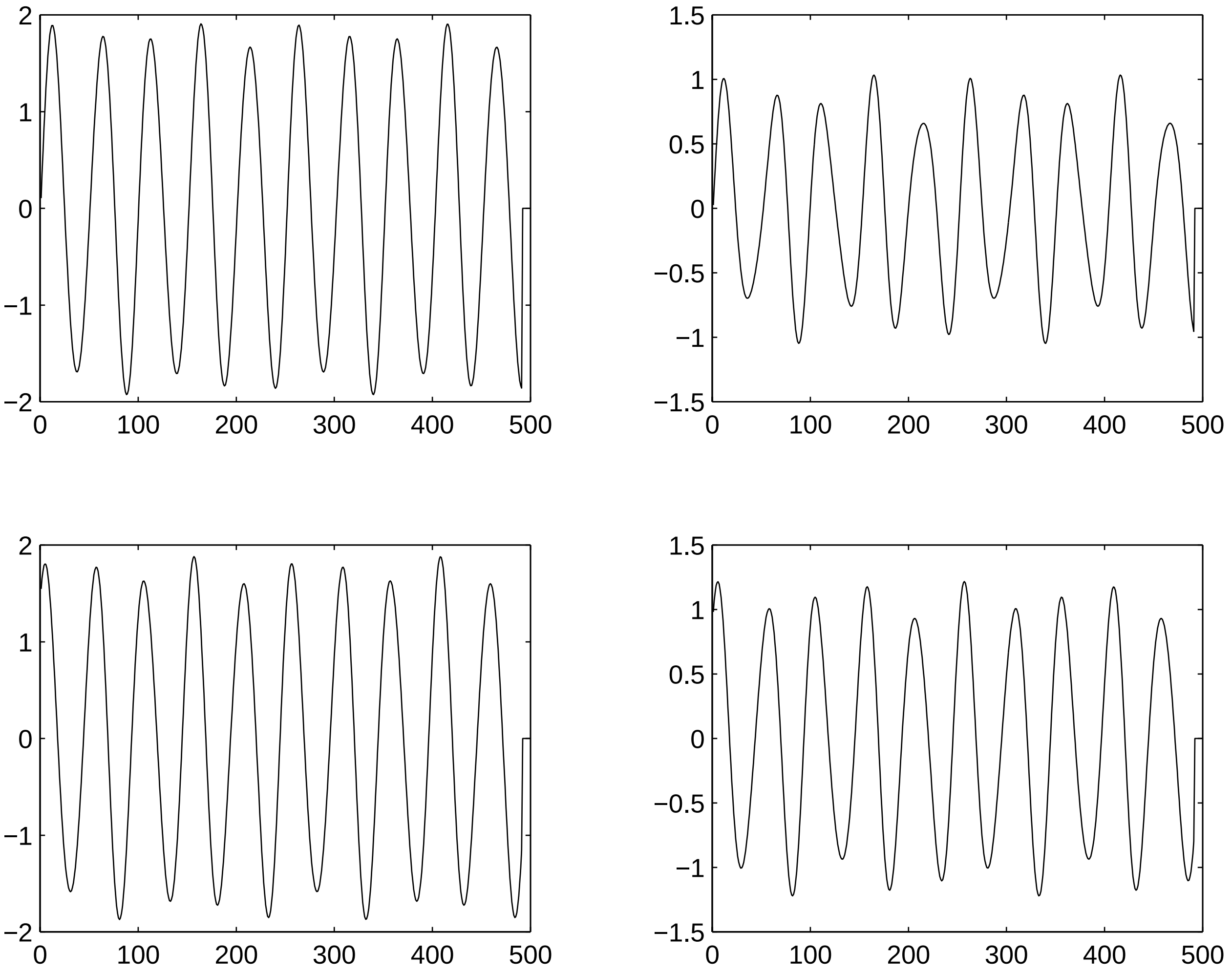

Fig. 6. One sine wave has been recovered from the nonlinear mixtures of the sines. The top line shows a pair of $\mathbf{y_1}$ neurons, the bottom line the corresponding pair of $\mathbf{y_2}$ neurons.

correlations when we selected random, but corresponding, points on the surface of the sphere and on the plane) and generate artificial data according to the prescription (see Fig. 5):

$$x_{11} = \sin(s_1) * \cos(s_2) \qquad (25)$$

$$x_{12} = \sin(s_1) * \sin(s_2) \qquad (26)$$

$$x_{13} = \cos(s_1) \qquad (27)$$

$$x_{21} = s_1 + s_2 \qquad (28)$$

$$x_{22} = s_2 - s_2 \qquad (29)$$

where e.g., $s_1(t) = \sin(t/8) * \pi$, $s_2(t) = \sin\{(t/5 + 3) * \pi\}$ for each of $t = 1, 2, \ldots, 500$. The results from the nonlinear mixture, $\mathbf{x_1}$ and $\mathbf{x_2}$, are shown in Fig. 6. We see that the sine wave has been recovered. However, the quality of the recovered signal deteriorates when both $\mathbf{x_1}$ and $\mathbf{x_2}$ are nonlinear mixtures.

5.4. *Blind separation using KCCA*

The Kernel CCA method of Sec. 4 also retrieves the individual sources from similar mixtures to those above but has the advantage that it requires only a small number of data points to create the appropriate kernel matrices. Typically around 80 samples gives as good or better identification of individual sinusoids as the nonlinear CCA method. There are two other situations which we now discuss where the Kernel method outperforms the nonlinear neural method.

5.4.1. *More signals than mixtures*

The data set we use comprises three sinusoids in noise mixed linearly with a random mixing matrix; we keep the number of mixtures to two so that we do not stray too far from biological plausibility and so the second line of Fig. 7 has only two images. We use 80 samples and linear kernels. Each sample contains the mixture at time t and the mixture at times $t-1, \ldots, t-9$ so that a vector of ten samples from the mixture is used at any one time. The underlying signals are shown in the top line of Fig. 7 and the correlation filtered data in the bottom line of that figure. The mixing matrix is shown at the top of the figure. We see that the three sinusoids have been separated with great accuracy. Similar results have been achieved with Gaussian and sigmoid kernels. There is a little beating in the higher frequency sinusoids which is not apparent when we separate two signals from two mixtures. This particular mixture was chosen since there is very little of the signal s_2 in the first mixture. This case is very readily treated with the technique of Minor Component Analysis (Ref. 6) but is the most difficult case for KCCA.

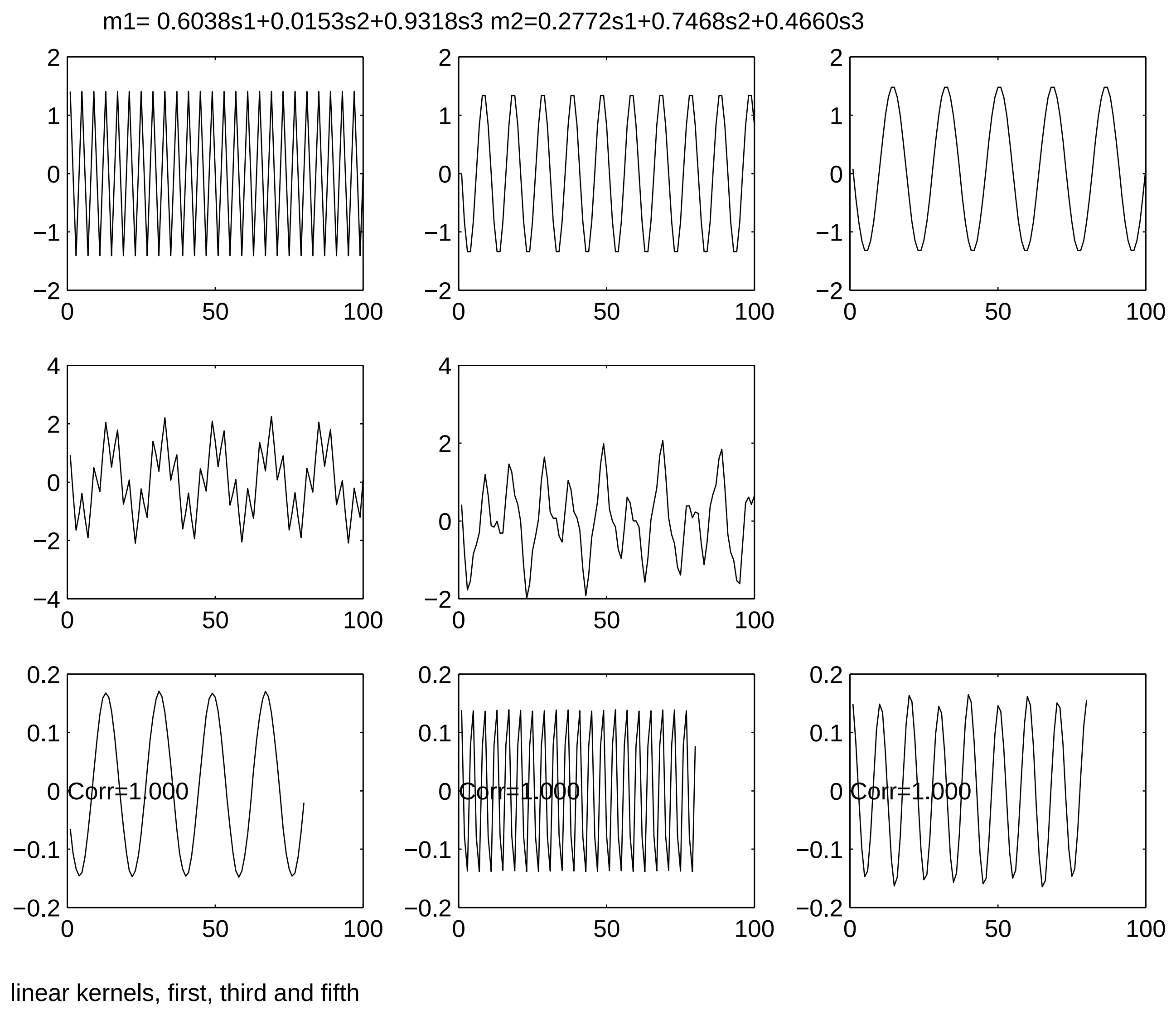

Fig. 7. The top line shows the three underlying sinusoids. The second line shows the two linear mixtures of these sinusoids presented to the algorithm. The bottom line shows three filters produced.

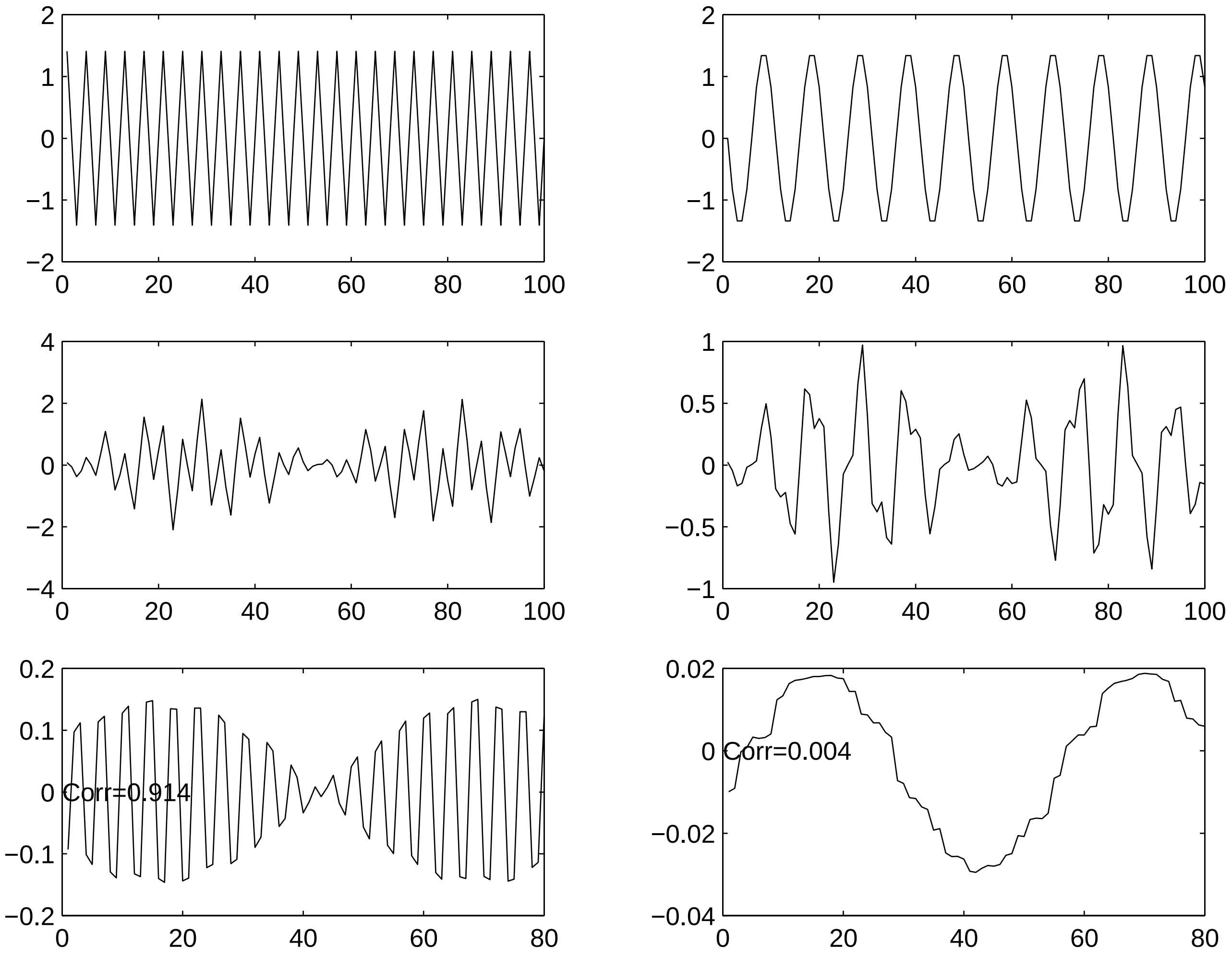

Fig. 8. The Gaussian kernel finds the signals but also finds the underlying slowly changing mixing process in its seventh projection. The top line shows the two underlying signals. The middle line shows the non-stationary mixtures. The bottom line shows two of the outputs from the Kernel CCA method: the first is one of the signals, the second is the mixing frequency.

5.4.2. *Time-varying mixtures*

Non-stationary mixtures are shown in Fig. 8. Now we create mixtures, m_1 and m_2 using

$$m_1 = 0.95\,\sin(t/17)s_1 + 0.61\,\sin(t/17)s_2$$

$$m_2 = 0.23\,\sin(t/17)s_1 + 0.49\,\sin(t/17)s_2$$

where t is a parameter denoting time. The underlying frequencies can be extracted by linear kernels. The rbf kernel, however, (Fig. 8) is also able to extract the mixing frequency in its 7th filter. This also happens when we use alternate sines and cosines (and with different frequencies) in our mixing matrix. The linear kernels also finds the individual sines but does not find the mixing frequencies.

6. Conclusion

In this paper, we have introduced a nonlinearity to our neural implementation of CCA. The nonlinear CCA rules were shown to find correlations greater in magnitude than could be found by linear CCA. We have also introduced kernel CCA and shown that again it can find correlations greater than linear correlations.

However we should add the caveat that it is entirely possible to create correlations using these methods and any use of such methods should be accompanied with a strong dose of common sense.

There is a case to be made that Kernel CCA and NLCCA are the same type of operation in that a kernel can be created out of any real-valued (nonlinear) function; we have for example shown that both can be used for Blind Separation of Sources. The NLCCA method has adaptive nonlinearities via the $\mathbf{v}$ weights while the kernel methods use a fixed nonlinear transformation to feature space; this would seem to potentially provide the NLCCA method with greater power than the kernel method. Further interpretation of the results of the kernel method is not always obvious. However the kernel approach has not been found wanting when we compare experimental results and seems to offer a new means of finding nonlinear and non-stationary correlations and one which is very promising for future research.

Appendix A
Centering in High-Dimensional Space

Given any Φ and any set of observations $\mathbf{x_1}, \ldots, \mathbf{x_M}$, the points

$$\tilde{\Phi}(\mathbf{x_i}) := \Phi(\mathbf{x_i}) - \frac{\mathbf{1}}{\mathbf{M}} \sum_{\mathbf{i=1}}^{\mathbf{M}} \Phi(\mathbf{x_i}) \qquad (30)$$

are centered. Thus, we go on to define the covariance matrix and $\tilde{\mathbf{K}}_{\mathbf{ij}} = (\tilde{\Phi}(\mathbf{x_i}) \cdot \tilde{\Phi}(\mathbf{x_j}))$ in F. We arrive at the already familiar eigenvalue problem,

$$\tilde{\lambda}\tilde{\alpha} = \tilde{\mathbf{K}}\tilde{\alpha} \qquad (31)$$

with $\tilde{\alpha}$ being the expansion coefficients of an eigenvector (in F). In terms of the points in (30), $\tilde{V} = \sum_{i=1}^{M} \tilde{\alpha}_i \tilde{\Phi}(\mathbf{x}_i)$. Because we do not have the centered data (see Eq. (30)), we cannot compute $\tilde{K}$ directly; however, we can express it in terms of its non-centered counterpart K. In the following, we shall use $K_{ij} = (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j))$ and the notations $\mathbf{1}_{ij} = 1$ for all $i, j, (1_M)_{ij} := \frac{1}{M}$, to compute $\tilde{K}_{ij} = (\tilde{\Phi}(\mathbf{x}_i) \cdot \tilde{\Phi}(\mathbf{x}_j))$:

$$\begin{aligned}\tilde{K}_{ij} &= \left(\left(\Phi(\mathbf{x}_i) - \frac{1}{M}\sum_{m=1}^{M}\Phi(\mathbf{x}_m)\right)\right. \\ &\qquad \left.\cdot \left(\Phi(\mathbf{x}_j) - \frac{1}{M}\sum_{n=1}^{M}\Phi(\mathbf{x}_n)\right)\right) \\ &= K_{ij} - \frac{1}{M}\sum_{m=1}^{M} 1_{im}K_{mj} - \frac{1}{M}\sum_{n=1}^{M} K_{in}1_{nj} \\ &\qquad + \frac{1}{M^2}\sum_{m,n=1}^{M} 1_{im}K_{mm}1_{nj} \\ &= (K - 1_M K - K 1_M + 1_M K 1_M)_{ij} \qquad (32)\end{aligned}$$

We thus can compute $\tilde{K}$ from K and then solve the eigenvalue problem (see Eq. (31)). The solutions $\tilde{\alpha}^{\mathbf{k}}$ are normalized by normalizing the corresponding vectors $\tilde{\mathbf{V}}^k$ in F, which translates into $\tilde{\lambda}_{\mathbf{k}}(\tilde{\alpha}^{\mathbf{k}} \cdot \tilde{\alpha}^{\mathbf{k}}) = \mathbf{1}$.

For feature extraction, we compute projections of centered Φ-images of test patterns t onto the eigenvectors of the covariance matrix of the centered points.

$$(\tilde{\mathbf{V}}^k \cdot \tilde{\Phi}(t)) = \sum_{i=1}^{M} \tilde{\alpha}_i^k (\tilde{\Phi}(x_i) \cdot \tilde{\Phi}(t)) \qquad (33)$$

Consider a set of test point $\mathbf{t_1}, \ldots, \mathbf{t_L}$, and define two $L \times M$ matrices by $\mathbf{K}_{ij}^{\text{test}} = (\Phi(t_i) \cdot \Phi(x_j)$ and $\tilde{\mathbf{K}}_{ij}^{\text{test}} = (\Phi(\mathbf{t_i}) - \frac{\mathbf{1}}{\mathbf{M}}\sum_{\mathbf{m=1}}^{\mathbf{M}} \Phi(\mathbf{x_m}))(\Phi(\mathbf{x_j}) - \frac{\mathbf{1}}{\mathbf{M}}\sum_{\mathbf{n=1}}^{\mathbf{M}} \Phi(\mathbf{x_n}))$. As in Eq. (32), we express $\tilde{K}^{\text{test}}$ in terms of K^{test}, and arrive at $\tilde{K}^{\text{test}} = K^{\text{test}} - 1'_M K - K^{\text{test}} 1_M + 1'_M K 1_M$, where $1'_M$ is the $L \times M$ matrix with all entries equal to $\frac{1}{M}$.

References

1. C. J. C. Burges 1988, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery* **2**(2), 1–43.
2. C. Jutten and J. Herault 1991, "Blind separation of sources, Part 1: An adaptive algorithm based on neuromimetic architecture," *Signal Processing* **24**, 1–10.
3. J. Karhunen and J. Joutsensalo 1995, "Generalizations of principal component analysis, optimization problems, and neural networks," *Neural Networks* **8**(4), 549–562.
4. P. L. Lai and C. Fyfe 1998, "Canonical correlation analysis using artificial neural networks," in *European Symposium on Artificial Neural Networks, ESANN98.*
5. P. L. Lai and C. Fyfe 1999, "A neural network implementation of canonical correlation analysis," *Neural Networks* **12**(10), 1391–1397.
6. P. L. Lai, D. Charles and C. Fyfe 2000, "Seeking independence using biologically inspired artificial neural networks," in *Developments in Artificial Neural Network Theory: Independent Component Analysis and Blind Source Separation*, Springer-Verlag.
7. K. V. Mardia, J. T. Kent and J. M. Bibby 1979, *Multivariate Analysis* (Academic Press).
8. S. Mika, B. Scholkopf, A. Smola, K.-R. Muller, M. Scholz and G. Ratsch 1999, "Kernel pca and de-noising in feature spaces," *Advances in Neural Processing Systems* **11**, 536–542.
9. E. Oja 1982, "A simplified neuron model as a principal component analyser," *Journal of Mathematical Biology* **16**, 267–273.

10. E. Oja 1997, "The nonlinear pca learning rule in independent component analysis," *Neurocomputing* **17**, 25–45.
11. J. O. Ramsay and B. W. Silverman 1997, *Functional Data Analysis* (Springer).
12. B. Scholkopf, S. Mika, C. Burges, P. Knirsch, K.-R. Muller, G. Ratsch, and A. J. Smola 1999, "Input space vs feature space in kernel-based methods," *IEEE Transactions on Neural Networks* **10**, 1000–1017.
13. B. Scholkopf, A. Smola and K.-R. Muller 1998, "Muller nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation* **10**, 1299–1319.
14. A. J. Smola, O. L. Mangasarian and B. Scholkopf 1999, "Sparse kernel feature analysis," *Technical Report 99-04*, University of Wiscosin, Madison.
15. V. Vapnik 1995, *The Nature of Statistical Learning Theory* (Springer Verlag, New York).