Chapter 7 Nonlinear multivariate statistical analysis

7.1 Nonlinear PCA (NLPCA)

If the data are in the form $\mathbf{x}(t) = [x_1, \ldots, x_l]$, where each variable $x_i, (i = 1, \ldots, l)$, is a time series containing n observations, the PCA method looks for u, a linear combination of the x_i, and an associated vector $\mathbf{a}$, with

$$u(t) = \mathbf{a} \cdot \mathbf{x}(t)\,, \tag{1}$$

so that

$$\langle \|\mathbf{x}(t) - \mathbf{a}u(t)\|^2 \rangle \text{ is minimized,} \tag{2}$$

where $\langle \cdots \rangle$ denotes a sample or time mean. Here u, called the first principal component (PC), is a time series, while $\mathbf{a}$, the first eigenvector of the data covariance matrix, (also called an empirical orthogonal function, EOF), often describes a spatial pattern.

The fundamental difference between NLPCA and PCA is that NLPCA allows a nonlinear mapping from $\mathbf{x}$ to u whereas PCA only allows a linear mapping. To perform NLPCA, a nonlinear mapping is made, i.e.

$$u(t) = f(\mathbf{x}(t), \mathbf{w}) \tag{3}$$

where f denotes the nonlinear mapping function from the data space to the u (the nonlinear PC) space, and $\mathbf{w}$, the parameters determining the f structure inherent to the dataset. Denoting g as the inverse mapping function from u to the data space, we have

$$\mathbf{x}'(t) = g(u, \tilde{\mathbf{w}}) \tag{4}$$

where g is the f-adjoint operator. For linear PCA, g is simply the transpose of f. $\mathbf{x}'(t)$ is the approximation to dataset $\mathbf{x}(t)$, when the 1-D PC space is used to describe the dataset.

As in linear PCA, the cost function defined by the error between $\mathbf{x}(t)$ and $\mathbf{x}'(t)$ is used to determine the parameters $\mathbf{w}$ and $\tilde{\mathbf{w}}$, i.e.

$$\langle \|\mathbf{x}(t) - \mathbf{x}'(t)\|^2 \rangle \text{ is minimized.} \quad (5)$$

An important issue in NLPCA is how to derive the nonlinear operators f and g from the inherent structure of the dataset. This has been implemented by neural networks (NN) (Kramer 1991), since NN can simulate any nonlinear continuous functions (Cybenko 1989).

The NLPCA is basically a standard feed-forward NN with 4-layers of transfer functions mapping from the inputs to the outputs.

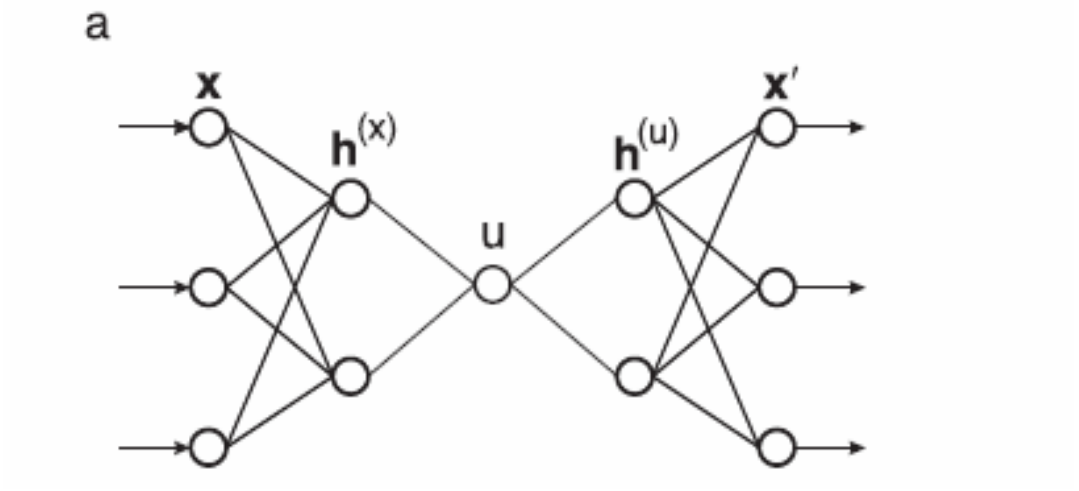


Fig. 1:
(a) A schematic diagram of the NN model for calculating nonlinear PCA (NLPCA). There are 3 'hidden' layers of variables or 'neurons' (denoted by circles) sandwiched between the input layer $\mathbf{x}$ on the left and the output layer $\mathbf{x}'$ on the right. Next to the input layer is the encoding layer, followed by the 'bottleneck' layer (with one neuron u), which is then followed by the decoding layer. A nonlinear function maps from the higher dimension input space to the lower dimension bottleneck space, followed by an inverse transform mapping from the bottleneck space back to the original space represented by the outputs, which are to be as close to the inputs as possible by minimizing the cost function $J = \langle \|\mathbf{x} - \mathbf{x}'\|^2 \rangle$. Data compression is achieved by the bottleneck, with the bottleneck neuron giving u, the nonlinear principal component (NLPC).

One can view the NLPCA network as composed of two standard 2-layer feed-forward NNs placed one after the other. The first 2-layer network maps from the inputs $\mathbf{x}$ through a hidden layer to the bottleneck layer with only one neuron u, i.e. a nonlinear mapping $u = f(\mathbf{x})$. The next 2-layer feedforward NN inversely maps from the nonlinear PC (NLPC) u back to the original higher dimensional $\mathbf{x}$-space, with the objective that the outputs $\mathbf{x}' = \mathbf{g}(u)$ be as close as possible to the inputs $\mathbf{x}$, where $\mathbf{g}(u)$ nonlinearly generates a curve in the $\mathbf{x}$-space, hence a 1-dimensional approximation of the original data. Because the target data for the output neurons $\mathbf{x}'$ are simply the input data $\mathbf{x}$, such networks are called *auto-associative* NNs. To minimize the MSE of this approximation, the objective function $J = \langle \|\mathbf{x} - \mathbf{x}'\|^2 \rangle$ is minimized to solve for the weight and offset parameters of the NN. Squeezing the input information through a bottleneck layer with only one neuron accomplishes the dimensional reduction.

In Fig.1, the transfer function f_1 maps from x, the input column vector of length *l,* to the first hidden layer (the encoding layer), represented by $\boldsymbol{h}^{(x)}$, a column vector of length m, with elements

$$h_k^{(x)} = f_1((\mathbf{W}^{(x)}\mathbf{x} + \mathbf{b}^{(x)})_k)\,, \tag{6}$$

where $\mathbf{W}^{(x)}$ is an $m \times l$ weight matrix, $\mathbf{b}^{(x)}$, a column vector of length m containing the offset parameters, and $k = 1, \ldots, m$. Similarly, a second transfer function f_2 maps from the encoding layer to the bottleneck layer containing a single neuron, which represents the nonlinear principal component u,

$$u = f_2(\mathbf{w}^{(x)} \cdot \mathbf{h}^{(x)} + \overline{b}^{(x)})\,. \tag{7}$$

The transfer function f_1 is generally nonlinear (usually the hyperbolic tangent or the sigmoidal function, though the exact form is not critical), while f_2 is usually taken to be the identity function.

Next, a transfer function f_3 maps from u to the final hidden layer (the decoding layer) $\mathbf{h}^{(u)}$,

$$h_k^{(u)} = f_3((\mathbf{w}^{(u)} u + \mathbf{b}^{(u)})_k) \,, \tag{8}$$

$(k = 1, \ldots, m)$; followed by f_4 mapping from $\mathbf{h}^{(u)}$ to $\mathbf{x}'$, the output column vector of length l, with

$$x_i' = f_4((\mathbf{W}^{(u)} \mathbf{h}^{(u)} + \overline{\mathbf{b}}^{(u)})_i) \,. \tag{9}$$

The objective function $J = \langle \|\mathbf{x} - \mathbf{x}'\|^2 \rangle$ is minimized by finding the optimal values of $\mathbf{W}^{(x)}$, $\mathbf{b}^{(x)}$, $\mathbf{w}^{(x)}$, $\overline{b}^{(x)}$, $\mathbf{w}^{(u)}$, $\mathbf{b}^{(u)}$, $\mathbf{W}^{(u)}$ and $\overline{\mathbf{b}}^{(u)}$. The MSE (mean square error) between the NN output $\mathbf{x}'$ and the original data $\mathbf{x}$ is thus minimized. The NLPCA was implemented using the hyperbolic tangent function for f_1 and f_3, and the identity function for f_2 and f_4, so that

$$u = \mathbf{w}^{(x)} \cdot \mathbf{h}^{(x)} + \overline{b}^{(x)} \,, \tag{10}$$

$$x_i' = (\mathbf{W}^{(u)} \mathbf{h}^{(u)} + \overline{\mathbf{b}}^{(u)})_i \,. \tag{11}$$

Furthermore, we adopt the normalization conditions that $\langle u \rangle = 0$ and $\langle u^2 \rangle = 1$. These conditions are approximately satisfied by modifying the objective function to

$$J = \langle \|\mathbf{x} - \mathbf{x}'\|^2 \rangle + \langle u \rangle^2 + (\langle u^2 \rangle - 1)^2 \,. \tag{12}$$

The total number of (weight and offset) parameters used by the NLPCA is $2lm + 4m + l + 1$, though the number of effectively free parameters is two less due to the constraints on $\langle u \rangle$ and $\langle u^2 \rangle$.

The choice of m, the number of hidden neurons in both the encoding and decoding layers, follows a general principle of parsimony. A larger m increases the nonlinear modelling capability of the network, but could also lead to overfitted solutions (i.e. wiggly solutions which fit to the noise in the data). If f_4 is the identity function, and m=1, then (11) implies that all x_i' are linearly

related to a single hidden neuron, hence there can only be a linear relation between the x_i' variables. Thus, for nonlinear solutions, we need to look at $m \geq 2$. Actually, one can use different numbers of neurons in the encoding layer and in the decoding layer; however, keeping them both at m neurons gives roughly the same number of parameters in the forward mapping from $\mathbf{x}$ to u and in the inverse mapping from u to $\mathbf{x}'$. It is also possible to have more than one neuron at the bottleneck layer. For instance, with two bottleneck neurons, the mode extracted will span a 2-D surface instead of a 1-D curve.

The transfer function tanh has the property that given x in the interval $[-L, L]$, one can find a small enough weight w, so that $\tanh(wx) \approx wx$, i.e. the transfer function is almost linear. Similarly, one can choose a large enough w, so that tanh approaches a step function, thus yielding Z-shaped solutions. If we can penalize the use of excessive weights, we can limit the degree of nonlinearity in the NLPCA solution. This is achieved with a modified objective function

$$J = \langle \|\mathbf{x} - \mathbf{x}'\|^2 \rangle + \langle u \rangle^2 + (\langle u^2 \rangle - 1)^2 + P \sum_{ki} (W_{ki}^{(x)})^2 \,, \tag{13}$$

where P is the weight penalty parameter. A large P increases the concavity of the objective function, and forces the weights $\mathbf{W}^{(x)}$ to be small in magnitude, thereby yielding smoother and less nonlinear solutions than when P is small or zero. Hence, increasing P also reduces the number of effectively free parameters of the model. We have not penalized other weights in the network. In principle, $\mathbf{w}^{(u)}$ also controls the nonlinearity in the inverse mapping from u to $\mathbf{x}'$. However if the nonlinearity in the forward mapping from $\mathbf{x}$ to u is already being limited by penalizing $\mathbf{W}^{(x)}$, then there is no need to further limit the weights in the inverse mapping.

The percentage of the variance explained by the NLPCA mode is given by

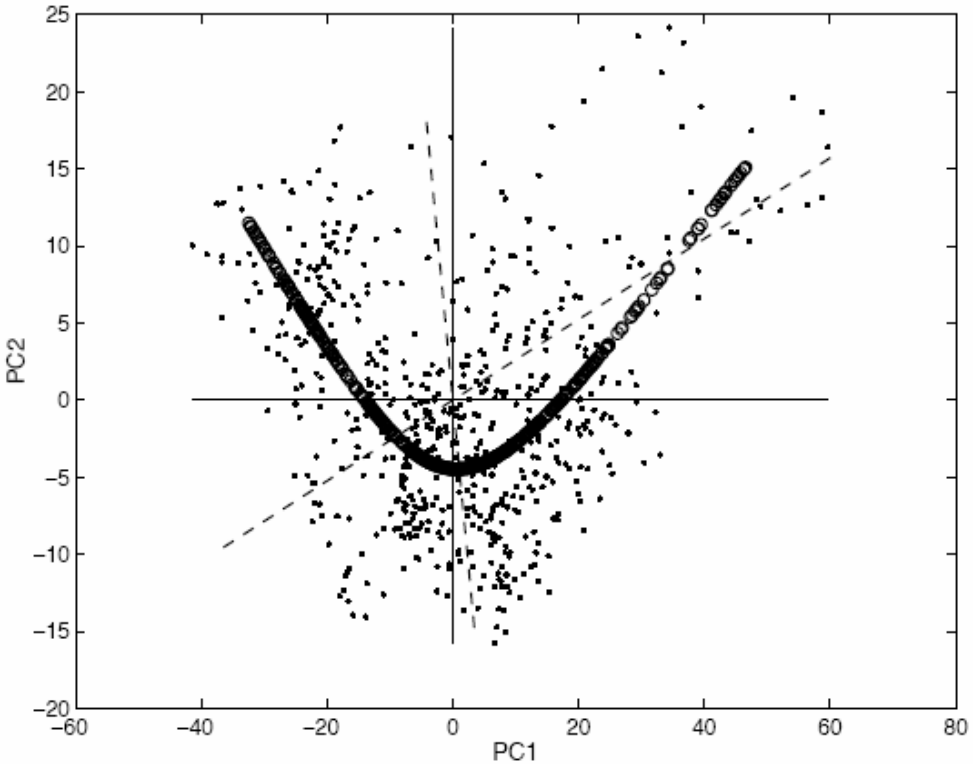
$$100\% \times \left(1 - \frac{\langle \|\mathbf{x} - \mathbf{x}'\|^2 \rangle}{\langle \|\mathbf{x} - \overline{\mathbf{x}}\|^2 \rangle} \right) , \tag{14}$$

with $\overline{\mathbf{x}}$ being the mean of $\mathbf{x}$.

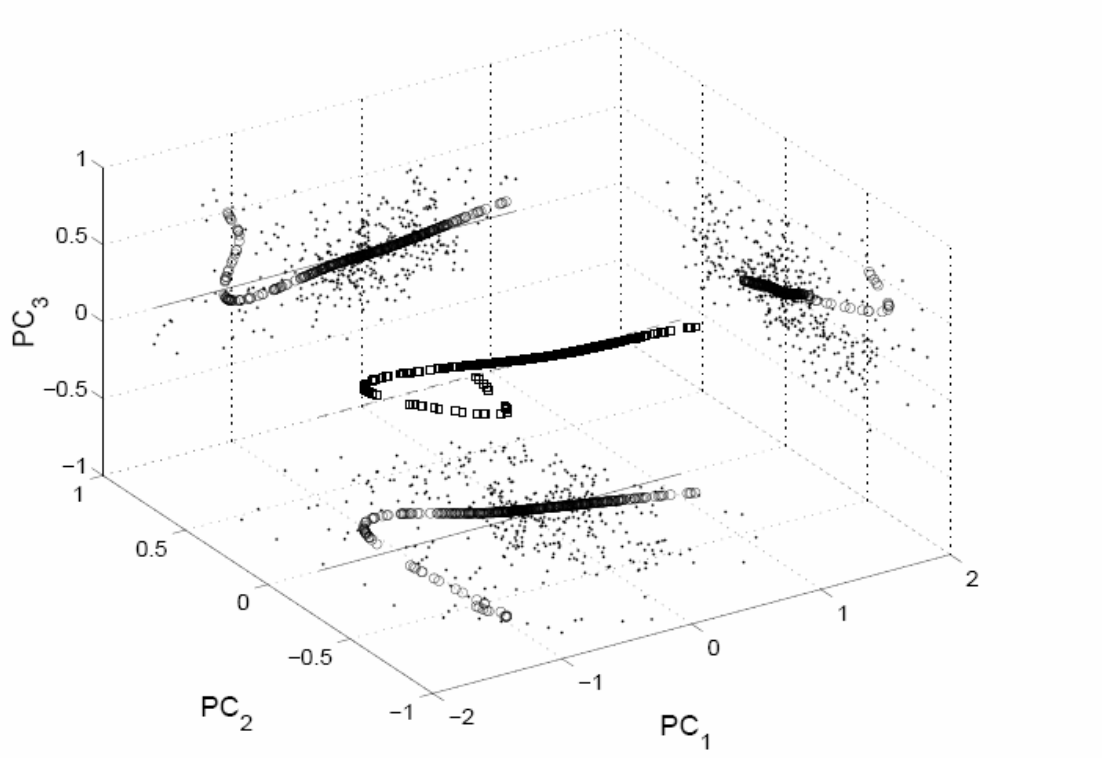
In effect, the linear relation ($u = \mathbf{e} \cdot \mathbf{x}$) in PCA is now generalized to $u = f(\mathbf{x})$, where f can be any nonlinear continuous function representable by a feed-forward NN mapping from the input layer to the bottleneck layer; and $\langle \|\mathbf{x} - \mathbf{g}(u)\|^2 \rangle$ is minimized. The residual, $\mathbf{x} - \mathbf{g}(u)$, can be input into the same network to extract the second NLPCA mode, and so on for the higher modes.

That the classical PCA is indeed a linear version of this NLPCA can be readily seen by replacing all the transfer functions with the identity function, thereby removing the nonlinear modelling capability of the NLPCA. Then the forward map to u involves only a linear combination of the original variables as in the PCA.

Example1:



Example2:



7.2 Nonlinear complex PCA (NLCPCA)

Complex principal component analysis (CPCA) is PCA applied to complex variables. In the first type of application, a 2-dimensional vector field such as the wind (u,v) can be analyzed by applying CPCA to w=u+iv (Chap. 4). In the second type of application, a real time-varying field can be complexified by the Hilbert transfer and analyzed by CPCA, often called Hilbert PCA (Chap. 5) to distinguish from the first type of application.

Earlier this chapter, we have examined a feed-forward NN for performing Nonlinear PCA. In this section, we will discuss how the same approach can be applied to complex variables, giving rise to nonlinear complex PCA.

The NLCPCA model uses basically the same architecture (Fig. 1) as the NLPCA model as discussed above (with three layers of hidden neurons where the middle layer is the bottle-neck layer), except all the input variables, and the weight and offset parameters are now complex-values.

The cost function is defined as

$$J = \langle \|\mathbf{z} - \mathbf{z}'\|^2 \rangle + P \sum_j |w_j|^2 , \tag{15}$$

where $\mathbf{z}$ is the model output, $\mathbf{z}'$, the target data, w_j, the individual weights from hidden layers 1, 2 and 3, and P, the weight penalty parameter.

Since the objective function J is a real function with complex weights, the optimization of J is equivalent to finding the vanishing gradient of J with respect to the real and the imaginary parts of the weights. All the weights (and offsets) in the model are combined into a single weight vector w. Hence the gradient of the objective function with respect to the complex weights can be split into

$$\frac{\partial J}{\partial \mathbf{w}} = \frac{\partial J}{\partial \mathbf{w}^{\mathrm{R}}} + i \frac{\partial J}{\partial \mathbf{w}^{\mathrm{I}}} \tag{16}$$

where $\mathbf{w}^{\mathrm{R}}$ and $\mathbf{w}^{\mathrm{I}}$ are the real and the imaginary components of the weight vector. The two components can be put into a single real parameter vector during nonlinear optimization using an algorithm for real variables.

7.3 Nonlinear singular spectrum analysis

In Chapter 5, we have learned that by incorporating time lagged version of the dataset, the PCA method can be extended to the singular spectrum analysis (SSA) method. We have also learned earlier in this chapter, that the leading PCs of a dataset can be nonlinearly combine by an NN to produce nonlinear PCA. We will see in this section that the leading PCs from an SSA can also be nonlinearly combined to produce nonlinear SSA (NLSSA).

The NLSSA procedure is as follows. First SSA is applied to the dataset as a prefilter, and after discarding the higher modes, we retain the leading SSA PCs, $\boldsymbol{x}(t) = [\boldsymbol{x}_1, ..., \boldsymbol{x}_l]$, where each variable $\boldsymbol{x}_i$, $(\mathrm{i} = 1, ..., \mathrm{l})$, is a time series of length n. The variables x are the inputs to the NLPCA model. For details, See Heish' s paper.

7.4 Nonlinear canonical correlation analysis (NLCCA)

Consider two vector variables $\mathbf{x}$ and $\mathbf{y}$, each with n samples. CCA looks for linear combinations

$$u = \mathbf{f}^{\mathrm{T}}\mathbf{x}, \quad \text{and} \quad v = \mathbf{g}^{\mathrm{T}}\mathbf{y}, \tag{17}$$

where the canonical variates u and v have maximum correlation, i.e. the weight vectors $\mathbf{f}$ and $\mathbf{g}$ are chosen such that $\mathrm{cor}(u, v)$, the Pearson correlation coefficient between u and v, is maximized (see chapter 4). For NLCCA, the nonlinear maps f and g, and their inverse maps, are replaced below by nonlinear mapping function using NNs.

The mappings from $\mathbf{x}$ to u and $\mathbf{y}$ to v are represented by the double-barreled NN on the left hand side of Fig. 2.

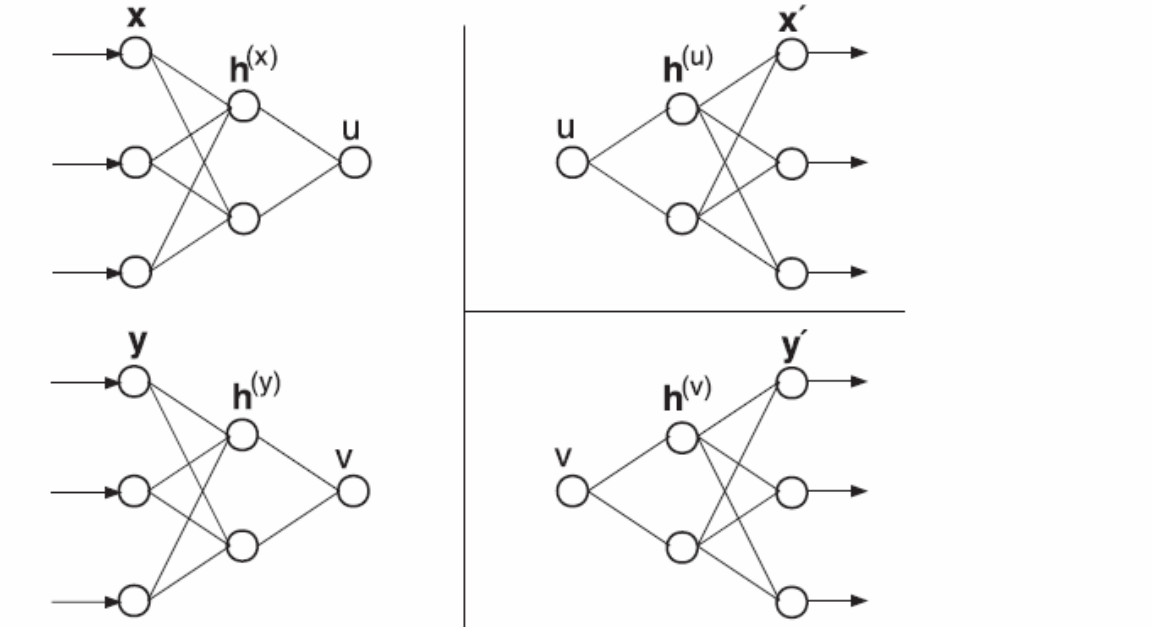


Fig.2

The three feed-forward NNs used to perform NLCCA. The double-

barreled NN on the left maps from the inputs $\mathbf{x}$ and $\mathbf{y}$ to the canonical variates u and v. The objective function J forces the correlation between u and v to be maximized. On the right side, the top NN maps from u to the output layer $\mathbf{x}'$. The objective function J_1 basically minimizes the MSE of $\mathbf{x}'$ relative to $\mathbf{x}$. The third NN maps from v to the output layer $\mathbf{y}'$. The objective function J_2 basically minimizes the MSE of $\mathbf{y}'$ relative to $\mathbf{y}$.

By minimizing the cost function $J = -cor(u,v)$, one finds the parameters which maximize the correlation cor(u,v). After the forward mapping with the double-barreled NN has been solved, inverse mapping from the canonical variates u and v to the original variables, as represented by the two standard NNs on the right of Fig. 2, are to be solved, where the NN of their outputs x' and y' are minimized with respect to x and y, respectively.

In Fig.2, the input x and y are mapped to the neurons in the hidden layer:

$$h_k^{(x)} = \tanh((\mathbf{W}^{(x)}\mathbf{x} + \mathbf{b}^{(x)})_k), \quad h_n^{(y)} = \tanh((\mathbf{W}^{(y)}\mathbf{y} + \mathbf{b}^{(y)})_n), \tag{18}$$

where $\mathbf{W}^{(x)}$ and $\mathbf{W}^{(y)}$ are weight matrices, and $\mathbf{b}^{(x)}$ and $\mathbf{b}^{(y)}$, the offset or bias parameter vectors. The dimensions of $\mathbf{x}$, $\mathbf{y}$, $\mathbf{h}^{(x)}$ and $\mathbf{h}^{(y)}$ are l_1, m_1, l_2 and m_2 respectively.

The canonical variate neurons u and v are calculated from a linear combination of the hidden neurons $\mathbf{h}^{(x)}$ and $\mathbf{h}^{(y)}$, respectively, with

$$u = \mathbf{w}^{(x)} \cdot \mathbf{h}^{(x)} + \overline{b}^{(x)}, \quad v = \mathbf{w}^{(y)} \cdot \mathbf{h}^{(y)} + \overline{b}^{(y)}. \tag{19}$$

These mappings are standard feedforward MLP NNs, and are capable of representing any continuous functions mapping from $\mathbf{x}$ to u and from $\mathbf{y}$ to v to any given accuracy, provided large enough l_2 and m_2 are used.

To maximize $\mathrm{cor}(u, v)$, the objective function $J = -\mathrm{cor}(u, v)$ is minimized by finding the optimal values of $\mathbf{W}^{(x)}$, $\mathbf{W}^{(y)}$, $\mathbf{b}^{(x)}$, $\mathbf{b}^{(y)}$, $\mathbf{w}^{(x)}$, $\mathbf{w}^{(y)}$, $\overline{b}^{(x)}$ and $\overline{b}^{(y)}$. The constraints $\langle u \rangle = 0 = \langle v \rangle$, and $\langle u^2 \rangle = 1 = \langle v^2 \rangle$ are also used, which are approximately satisfied by modifying the objective function to

$$J = -\mathrm{cor}(u, v) + \langle u \rangle^2 + \langle v \rangle^2 + (\langle u^2 \rangle^{1/2} - 1)^2 + (\langle v^2 \rangle^{1/2} - 1)^2 \,. \tag{20}$$

On the right side of Fig. 2, the top NN maps from u to x' in two steps:

$$h_k^{(u)} = \tanh((\mathbf{w}^{(u)} u + \mathbf{b}^{(u)})_k) \,, \quad \text{and} \quad \mathbf{x}' = \mathbf{W}^{(u)} \mathbf{h}^{(u)} + \overline{\mathbf{b}}^{(u)} . \tag{21}$$

The objective function $J_1 = \langle \|\mathbf{x}' - \mathbf{x}\|^2 \rangle$ is minimized by finding the optimal values of $\mathbf{w}^{(u)}$, $\mathbf{b}^{(u)}$, $\mathbf{W}^{(u)}$ and $\overline{\mathbf{b}}^{(u)}$. The MSE between the NN output $\mathbf{x}'$ and the original data $\mathbf{x}$ is thus minimized.

Similarly, the bottom NN on the right side of Fig. 2 maps from v to y':

$$h_n^{(v)} = \tanh((\mathbf{w}^{(v)} v + \mathbf{b}^{(v)})_n) \,, \quad \text{and} \quad \mathbf{y}' = \mathbf{W}^{(v)} \mathbf{h}^{(v)} + \overline{\mathbf{b}}^{(v)} , \tag{22}$$

with the objective function $J_2 = \langle \|\mathbf{y}' - \mathbf{y}\|^2 \rangle$ minimized. The total number of parameters used by the NLCCA is $2(l_1 l_2 + m_1 m_2) + 4(l_2 + m_2) + l_1 + m_1 + 2$, though the number of effectively free parameters is four less due to the constraints on $\langle u \rangle$, $\langle v \rangle$, $\langle u^2 \rangle$ and $\langle v^2 \rangle$.

A number of runs mapping from $(\mathbf{x}, \mathbf{y})$ to (u, v), using random initial parameters, was performed. The run attaining the highest $\mathrm{cor}(u, v)$ was selected as the solution. Next a number of runs (mapping from u to $\mathbf{x}'$) was used to find the solution with the smallest MSE in $\mathbf{x}'$. Finally, a number of runs was used to find the solution yielding the smallest MSE in $\mathbf{y}'$. After the first NLCCA mode has been retrieved from the data, the method can be applied again to the residual to extract the second mode, and so forth.

With three NNs in NLCCA, overfitting can occur in any of the three networks. With noisy data, the three objective functions are modified to:

$$J = -\mathrm{cor}(u,v) + \langle u\rangle^2 + \langle v\rangle^2 + (\langle u^2\rangle^{1/2} - 1)^2 + (\langle v^2\rangle^{1/2} - 1)^2 + P\left[\sum_{ki}(W_{ki}^{(x)})^2 + \sum_{nj}(W_{nj}^{(y)})^2\right],$$

$$J_1 = \langle \|\mathbf{x}' - \mathbf{x}\|^2\rangle + P_1 \sum_k (w_k^{(u)})^2 ,$$

$$J_2 = \langle \|\mathbf{y}' - \mathbf{y}\|^2\rangle + P_2 \sum_n (w_n^{(v)})^2 ,$$

where P, P_1 and P_2 are nonnegative weight penalty parameters. Since the nonlinearity of a network is controlled by the weights in the hyperbolic tangent transfer function, only those weights are penalized.

Example:

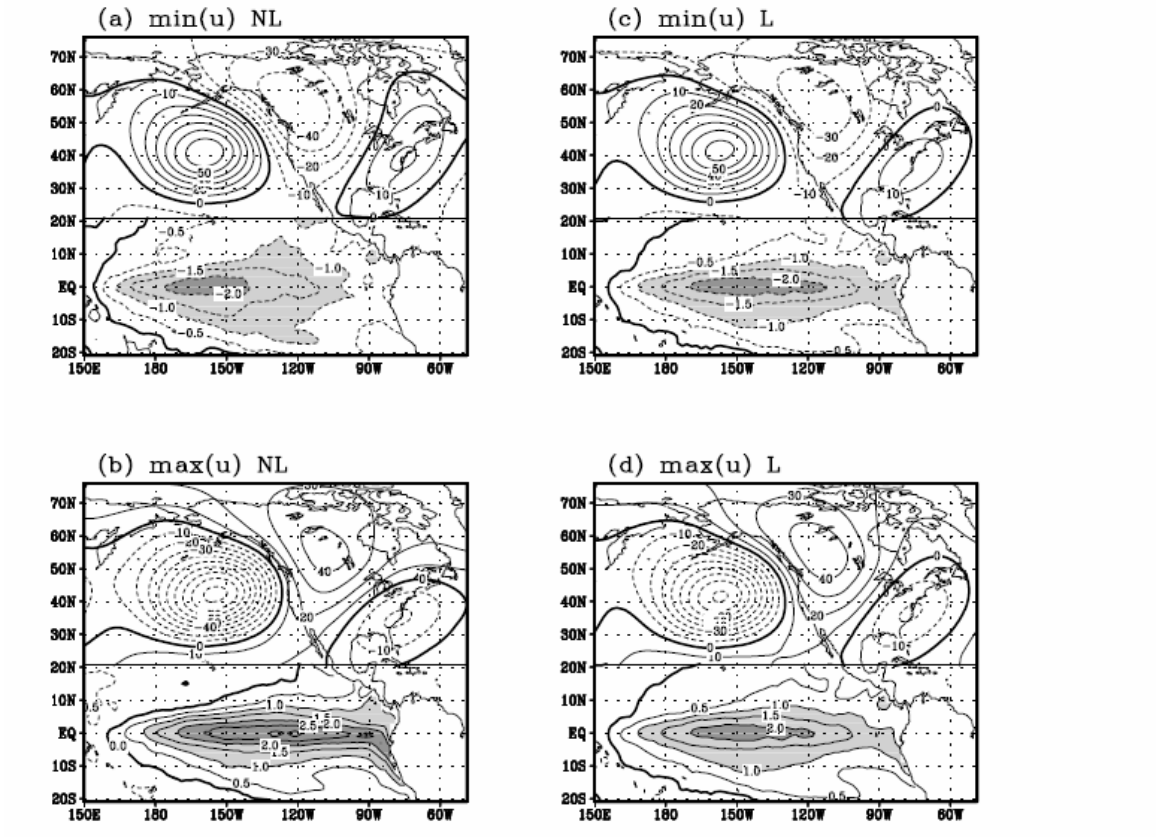


Fig.3

The spatial patterns for the first NLCCA mode between the winter Z500 anomalies and the tropical Pacific SST anomalies as the canonical variate u takes its (a) minimum value and (b) maximum value. The Z500 anomalies with contour intervals of 10m are shown north of 20°N. SST anomalies with contour intervals of 0.5°C are displayed south of 20°N. The SST anomalies greater than +1°C or less than -1°C are shaded, and heavily shaded if greater than +2°C or less than -2°C. The linear CCA mode 1 is shown in panels (c) and (d) for comparison.