Institute of Paper Science and Technology
Atlanta, Georgia

IPST Technical Paper Series Number 957

On Proper Handling of Multicollinear Inputs and Errors-in-Variables with Explicit and Implicit Neural Models

S. Karrila and N. Euliano

November 2002

Submitted to
FLAIRS - 2003
May 11-15, 2003
St. Augustine, Florida

INSTITUTE OF PAPER SCIENCE AND TECHNOLOGY
PURPOSE AND MISSIONS

The Institute of Paper Science and Technology is an independent graduate school, research organization, and information center for science and technology mainly concerned with manufacture and uses of pulp, paper, paperboard, and other forest products and byproducts. Established in 1929 as The Institute of Paper Chemistry, the Institute provides research and information services to the wood, fiber, and allied industries in a unique partnership between education and business. The Institute is supported by 41 member companies. The purpose of the Institute is fulfilled through four missions, which are:

- to provide a multidisciplinary graduate education to students who advance the science and technology of the industry and who rise into leadership positions within the industry;
- to conduct and foster research that creates knowledge to satisfy the technological needs of the industry;
- to provide the information, expertise, and interactive learning that enable customers to improve job knowledge and business performance;
- to aggressively seek out technological opportunities and facilitate the transfer and implementation of those technologies in collaboration with industry partners.

ACCREDITATION

The Institute of Paper Science and Technology is accredited by the Commission on Colleges of the Southern Association of Colleges and Schools (1866 Southern Lane, Decatur, Georgia 30033-4097: Telephone number 404-679-4501) to award the master of science and doctor of philosophy degrees.

NOTICE AND DISCLAIMER

The Institute of Paper Science and Technology (IPST) has provided a high standard of professional service and has put forth its best efforts within the time and funds available for this project. The information and conclusions are advisory and are intended only for internal use by any company who may receive this report. Each company must decide for itself the best approach to solving any problems it may have and how, or whether, this reported information should be considered in its approach.

IPST does not recommend particular products, procedures, materials, or service. These are included only in the interest of completeness within a laboratory context and budgetary constraint. Actual products, materials, and services used may differ and are peculiar to the operations of each company.

In no event shall IPST or its employees and agents have any obligation or liability for damages including, but not limited to, consequential damages arising out of or in connection with any company's use of or inability to use the reported information. IPST provides no warranty or guaranty of results.

On proper handling of multicollinear inputs and errors-in-variables with explicit and implicit neural models

Seppo J. Karrila and Neil R. Euliano

Submitted to
FLAIRS-2003
The 16th International FLAIRS Conference, Special Track on Neural Network Applications
St. Augustine, FL, May 11-15, 2003

Abstract

When model input variables appear redundant, it is common practice to simply drop some of them until the redundancy is removed, prior to model identification. As a result, the final model has forgotten the interdependency in the original input data, which may be an essential condition for model validity.

We provide a practical approach to neural network modeling, such that the final model will also incorporate a "memory" of the multicollinearity in training inputs and provide a check on new input vectors for consistency with this pattern.

We approach this problem stepwise, pointing out the benefits achieved or lost at each step when model complexity is increased. The steps lead in a natural way to building implicit models, which also handle noise in inputs in close resemblance to total least squares. The practical tool for this is a feedforward network of specifically selected configuration.

Introduction

We will denote the vector of perceived independent variables by $\mathbf{x}$, with components $x_1, \ldots, x_m$, and the perceived dependent variables (to be modeled) by $\mathbf{y}$, with components $y_1, \ldots, y_n$. While it is conventional to use $\mathbf{x}$ as the inputs and $\mathbf{y}$ as the outputs of a neural model, we will see that other arrangements have their benefits.

Feature selection and feature extraction reduce the dimensionality of the model input vector to reach a parsimonious model that can be identified from the available limited size empirical data set (Jain et al. 2000).

Consider data in which the input variables x_1 and x_2 contain essentially the same information; let $x_1=u(x_2)$ as a good approximation. We then say that x_1 and x_2 are collinear, even when u is not a linear mapping; a more proper term would be "functionally related" or "redundant."

The alternative models $\mathbf{y}=f(\mathbf{x})=f(u(x_2),x_2)=f(x_1,u^{-1}(x_1))$ seem equivalent, but neither of the last two models is valid when $x_1 \neq u(x_2)$; this condition needs to be retained with the model. The only exception is when, say, x_1 is only a dummy argument in $f(\mathbf{x})$ and $\mathbf{y}$ is independent of it.

Ideally the collinearity and variable range conditions in the learning data will be embedded in the model, and still the practical cost of creating and using such improved model should not be prohibitive. We provide an approach that matches these goals.

As side products we will gain a simple saliency indicator for the inputs (for feature selection, reverting back to dropping variables which is occasionally justified), a procedure for reduction in the number of model parameters without dropping any inputs, and a practical approach close to nonlinear "total least squares" modeling that allows errors (unbiased noise) in the independent variables.

Our ultimate models are of Non-Linear Factor Analysis (NLFA) type (Karrila and Rezak 2002, Karrila 2002), providing functional constraints in implicit form between $\mathbf{x}$ and $\mathbf{y}$. Such models are symmetric in the sense of not distinguishing between inputs and outputs, and allow multivalued x#y relationships, at the cost of requiring an iterative solution for each evaluation.

The basis for all of the above is simple: a linear bottleneck as the first hidden layer is employed with otherwise conventional feedforward nets, which can be trained with backpropagation. Hetero- and autoassociation, as well as a hybrid or mixed scheme between these, are used in network training to effect dimensionality reduction (Verveer and Duin 1995) that removes collinearity.

Theory

The removal of collinearity is based on dimensionality reduction, familiar to most in the linear setting of PCA or factor analysis. Original data are represented in a lower dimensional coordinate system (encoded), so that reconstruction error between decoded and real data is tolerable.

Principal Component Analysis (PCA) is a well-established method, mathematically related to the Singular Value Decomposition or Polar Decomposition of a matrix. It finds the unique affine subspace of given dimensionality,

such that the orthogonal projection of data into this subspace retains a maximal fraction of the variance; the extracted features are orthogonal projections to the axes spanning this subspace. Factor Analysis can be considered postprocessing of the PCA results by selecting a new, possibly oblique, set of axes (approximately) spanning the same subspace, often so that each of the feature values (called scores) is referred back to a small subset of the original variables.

On performing PCA all variables are treated symmetrically; there are no inputs and outputs. Still the final result can be viewed as a linear, total least squares model – the squared deviations from a linear (or affine) subspace in its normal direction have been minimized, and some coordinates can be solved given others, based on the "implicit model" that data lie in this subspace.

We will pursue similar traits in a nonlinear setting by using neural networks as the engine that performs the minimization of error.

A feedforward neural network processes a data vector sequentially layer by layer. The outputs of any layer can be viewed as encoded values, mapped to the outputs of the network by the remaining layers. If the output target values are equal to the inputs, the network is autoassociative. Then the encoded values are approximately mapped back to the inputs; the same network also embeds the decoding mapping in its structure.

If an autoassociative neural network (AANN) is successfully trained, it has learned encoding-decoding mappings that preserve the original data vectors with only a small reconstruction error (and when a validation set is used, the ability to interpolate has also been tested during training and network selection). If the AANN has a bottleneck layer with a small number of nodes, the encoding at this layer reduces the dimensionality of the data vectors. This is how the bottleneck AANN structure functions as a tool for data reduction.

An AANN with linear activation functions and only one hidden bottleneck layer performs PCA. It finds the same subspace as PCA would find for a given reduction of dimensionality, but the factor loadings (weight vectors of linear encoding) will not be orthogonal without postprocessing or special network constructs. An extensive review is provided in a recent book (Diamantaras and Kung 1996).

Kramer's nonlinear PCA (NLPCA) learns nonlinear mappings f and g to perform encoding and decoding. These mappings are each represented by an NN with (at least) one hidden sigmoidal layer, and when the two networks are combined at the bottleneck so that the output layer of f is the input layer of g, the AANN has (at least) three hidden layers. The universal approximation property of NN ensures that three hidden layers with enough nodes in the first and third will always suffice, and this is the configuration that Kramer originally presented (Kramer 1991). NN training becomes more difficult as network depth is increased and even with convergence the weights may be stuck to a local suboptimal error minimum; this encourages limiting the network depth, but on occasions increasing the number of layers is useful (Villiers and Barnard 1992).

An invertible mapping applied to the reduced values can be used to form new encoding-decoding pairs, so this nonlinear reduction is not unique (or user independent) like linear PCA is. While Kramer discusses the application of information theoretic principles to select the reduced dimensionality, using the validation set seems to be a good practical approach for avoiding over-fitting also with AANN.

The NLPCA seems not to be popular in applications and we take a step back from it. In the following we explore the benefits of linear encoding, combined with nonlinear processing of the reduced inputs.

Adding a linear bottleneck to an ordinary I/O network; parsimony and saliency benefits

Feature selection is a particular case of linear mapping $\mathbf{x}\#A\mathbf{x}$, with some rows of the identity matrix I dropped to form A; preprocessing with a linear mapping not of full rank is common in this form. The purpose is to reduce the number of parameters in a neural model and ensure that the available training data are sufficient for "identification."

The number of hidden nodes may be dictated by the nature of the output, as each "ridge" in it requires at least one sigmoidal hidden node. We consider the number of sigmoidal hidden nodes r fixed.

Insert a linear bottleneck layer (i.e., the nodes pass on their net activation as such) with p nodes after the input layer, without biases so the mapping is linear and not general affine. Prior to the sigmoidal layer there are now $mp+pr+r$ parameters, compared with earlier $mr+r$ parameters. We benefit if $p<mr/(m+r)$, or equivalently $2p<H$ where H is the harmonic mean of m and r (the geometric mean $G=\sqrt{(mr)}$ σ H is a useful bound if m and r are of the same order of magnitude). The bottleneck should then be tried with p about H/2 (or G/2), and if small training error is reached, smaller values of p can be tried. This constriction in the network can be viewed as one sort of regularization.

The activations of the $p<m$ bottleneck nodes can be considered "indicators" that show the joint effects of inputs. If the input variables have been normalized, the weights to the bottleneck indicate relative strengths of effects on the output(s). The weights act as a saliency measure.

Mixed association to embed collinearity rules in model and achieve total least squares fitting

If, as in the Introduction, $x_1=u(x_2)$, then the linear bottleneck $A\mathbf{x}=x_2$ does not lose information from $\mathbf{x}$; the reconstruction of $\mathbf{x}=(u(x_2),x_2)$ will require a nonlinear mapping. The universal approximation capability of the

network after the bottleneck ensures that $\mathbf{x}$ can be (approximately) reconstructed (Barron 1993). Let us then assign $\mathbf{x} \oplus \mathbf{y}$ as the targeted output, i.e., add output nodes to the previous model type.

The mapping u is now part of the network model, so the input collinearity is embedded; the model also calculates "corrected independent variables" that obey this collinearity, enabling checking or alerts if deviation from given new independent variables (inputs to model) is large. This "mixed" model between auto- and heteroassociation reaches our first goal. If the input nodes clip the input ranges according to the span of the training data, the model will also alert to extrapolation.

Neglecting the outputs y and restricting the mapping after the bottleneck to linear, we recover the neural computation of principal components of $\mathbf{x}$. The support of A, namely $A^T(AA^T)^{-1}A$, projects $\mathbf{x}$ into that subspace which spans maximal variance with dimensionality reduction to rank(A). Alternatively, this subspace provides a linear model that has been fit to the data in the total least squares sense.

When a total least squares model is desired, the errors in independent variables need to be observed in fitting the model. With feedforward networks this only happens when the independent variables are included in the outputs – an error sum is formed only at the output layer.

In the next section we will further discuss what the linear "encoding" bottleneck does in combination with a nonlinear "decoding."

Implicit models and Non-Linear Factor Analysis

Consider full autoassociation with a linear bottleneck as the first hidden layer. Now $\mathbf{x} \oplus \mathbf{y}$ is both the input and targeted output of the neural network. For brevity, we will denote this set of variables by only $\mathbf{x}$ (as if dropping y from the previous model type). The perceived independent and dependent variables are now treated symmetrically. In recent publications this network topology has been given the name Non-Linear Factor Analysis (NLFA), as it encodes to "score values" with linear "factor loadings" in matrix A, and decodes nonlinearly (Karrila and Rezak 2002, Karrila 2002).

The combination of linear encoding to reduced dimensionality and nonlinear decoding back to approximate reconstruction seems to have been neglected prior to publication of the NLFA method. Viewing the encoding and decoding as inverse mappings we might assume that if one is linear so should be the other. Indeed, if the *decoding* is linear, the encoding is given by its pseudoinverse. However, our earlier example of $\mathbf{x}\#x_2\#\mathbf{x}=(u(x_2),x_2)$ provides a natural linear *encoding* with nonlinear decoding. It is easy to construct examples of this type that will show the shortcomings of linear PCA with nonlinear data.

We define the optimal result of linear dimensionality reduction as follows, for the intrinsic smooth functional relationship between all variables in $\mathbf{x}$. To get a rigorous definition, we consider a continuum of points (not a noisy and discrete set of data records) that obey the underlying relationship and form a manifold.

Definition of LRID. A manifold M in ∇^m is linearly reducible to dimension p, if there is a linear mapping A into ∇^p such that the restriction of A to M is bijective (one-to-one). The linearly reducible intrinsic dimensionality (LRID) of M is the smallest of such values p.

If $p<m$, then data reduction is performed by the encoding-decoding pair $\mathbf{x} \# A\mathbf{x} \# \mathbf{x}$. Denoting the (nonlinear) decoding by g, $\mathbf{x}=g(A\mathbf{x})$ for all $\mathbf{x}$ in M. If p=LRID, then A must be onto ∇^p and of full row rank, and AA^T is invertible, so the projection to support (orthogonal complement of null space) can be calculated as given earlier.

Going backwards from Kramer's NLPCA, if we restrict the encoding to be linear we have NLFA, and if we further restrict also the decoding to be linear we have PCA. Clearly the capacity to reduce data decreases with each restriction, so NLFA is a compromise in complexity and capacity between the two earlier methods – a semilinear method between fully linear and fully nonlinear. There are cases where the intrinsic dimensionality is strictly less than the LRID, but in many practical cases the nonlinearity is "mild" and NLFA works very well.

To visualize a comparison between these methods, Figure 1 provides a taxonomy, which refines the conventional classification of data reduction methods to linear and nonlinear. It is appropriate to categorize methods based on the encoding and decoding types separately. With the addition of the NLFA method to the arsenal, an approach exists for every useful slot in this taxonomy.

Note that the NLFA provides a model $\mathbf{x}=g(A\mathbf{x})=g(v)$, which is implicit and can be used as follows. Given enough components of $\mathbf{x}$ as vector $B\mathbf{x}$, we can numerically solve $B\mathbf{x}=Bg(v)$ for v and get the remaining components from g(v). Only g needs to be stored for model application, and the collinearity check is whether a satisfactory v can be found. Separate checking of ranges for components of $\mathbf{x}$ can of course be done. When v lives in ∇^2 we can visualize g as contour plots.

DECODING \ ENCODING	Linear	Nonlinear
Linear	Principal component analysis (PCA)	Useless, revert to PCA
Nonlinear	NLFA	Kramer's NLPCA

Figure 1. Types of encoding and decoding are used to create taxonomy of some constructive data reduction methods. NLFA stands for Non-Linear Factor Analysis, while NLPCA refers to Non-Linear PCA with neural networks.

In the following summary the linear operator A serves as an indication of a linear bottleneck layer with p nodes in the first hidden position. Output approximating, say, target **x** is denoted by **x'**. The independent and dependent variables are components of **x** and **y**, respectively.

Model type	NN dissected	Benefit and Cost
I/O plain	**x# y'** **Heteroassociative**	Baseline
I/O With linear feature extraction	**x#Ax# y'** **Heteroassociative**	+ parsimony + saliency estim. - "extra" layer to train
I/O With embedded NLFA for indep. variables only	**x#Ax# x'⊕y'** **Mixed association**	+ handles multicollinearity with new inputs + errors-in-vars - apparent complexity up
Implicit (NLFA)	**x⊕y#A(x⊕y) # x'⊕y'** **Autoassociative**	+ all of the above benefits + **x# y'** can be many-valued in an implicit model - use of model requires iterations

Table 1. Comparison of the model types discussed in the order of increasing model complexity.

A numerical example; laboratory refining data

Wood pulp for making paper is mechanically beaten or refined to soften and break the fibers. The type and amount of refining can be observed through its effects on measured properties of paper made in a "standard way." First principle modeling of these phenomena is intractable, and empirical modeling is needed to gain insights.

Linear factor analysis has been applied (Howard, Poole, and Page 1994) to a published collection of data (Cottral et al. 1954), of which we only inspect a subset. Up to the point of rotating the principal components the results of such analysis are user independent. Further background on linear factor analysis (and principal component analysis) can be found in the references of their publication.

They found three major factors and gave an interpretation for each. With any method (including NLFA) one could dispute where the cutoff in required reproduction error should be; equally well four or five factors could have been retained from PCA, but the available interpretations happened to have a good match with only three factors.

Our data come from Appendix I, Table 16, of the same reference dating back to 1954. These data are for a bleached sulfite pulp labeled "extra strong (green)." Six different types of beater were applied to the same pulp for various periods of time, by different laboratories; due to a duplication of the Lampen mill, there are seven "beating curves."

We model ten pulp and paper characteristics with the NLFA method. These are freeness (of pulp), breaking length, Mullen burst factor, tear factor, Bekk porosity (transformed with natural logarithm), Schopper fold (transformed with natural logarithm), % stretch to failure, light scattering coefficient, contrast ratio, and apparent density.

Two state variables are found to conserve the measured data within reasonable reproduction error. The lowest R^2 – value 0.88 is found on predicting stretch, mainly due to inaccuracy in the measurement of this variable. With the exception of natural logarithm of porosity (value 0.96) and tear (0.97), the remaining variables have R^2 –values around 0.99.

Only eight sigmoidal nodes are needed in the second hidden layer; the first has two linear nodes as stated above. There are 40 data records, and 20% of these were held as validation data during training. This is a typical case where the number of data records is seemingly small compared with the number of variables.

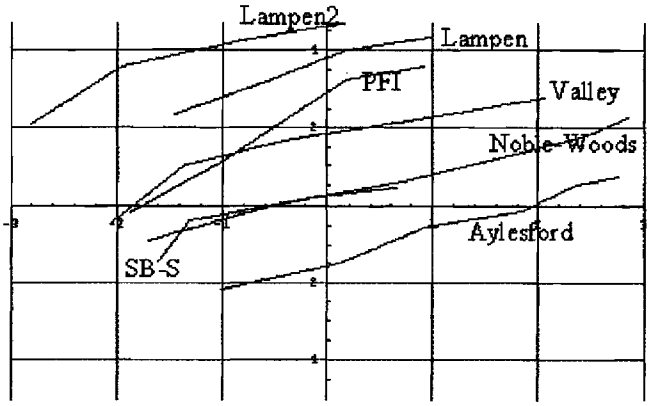

Figure 2. The reduced coordinates (**v** in the theory section) trace distinct, nearly parallel curves for each type of beater device. Along each curve the amount of beating is varied.

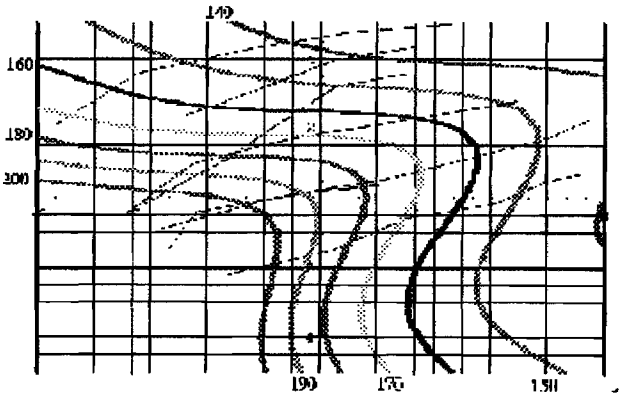

Figure 3. The tear factor is shown as contours superposed on the curves of Figure 2.

Graphics such as Figure 3 also serve as a check that no sharp kinks or other anomalies have been introduced in the model.

In terms of the fraction of variance explained, the NLFA with reduction to two dimensions performs slightly better than PCA with reduction to three dimensions. This is to be expected when the functional dependencies between variables are nonlinear. As an illustration of what goes on here, the canvas of an umbrella can be poorly fit with a linear subspace, so PCA will claim it is three-dimensional; but only two parameters are needed to generate this surface – a two-dimensional manifold. We expect that a semilinear method such as NLFA can gain acceptance and simultaneously far exceed the power of linear PCA in practical applications.

Notes on practical application

The methods presented are based on conventional feedforward networks with at least two hidden layers. The first hidden layer is a linear layer with fewer nodes than there are inputs. When total least squares-type models are generated, the perceived independent variables are included as targeted outputs. In the mixed-type (explicit) modeling, this may have a "hints" or "multitask learning" effect, which aids the network in learning to model the dependent variables (Caruana 1997).

It has been theoretically shown recently that bypass connections from the bottleneck to the output layer can improve reconstruction of the inputs (Karrila 2002). Flexible neural software that allows arbitrary network configurations, based on Wan's transposed network for backpropagation, can be useful for further experimentation (Wan and Beaufays 1996; Principe, Euliano, and Lefebvre 2000).

The number of nodes needed in the bottleneck needs to be found by experimentation. While PCA can from a single calculation show the effect of any linearly reduced dimensionality on reconstruction error, NLFA requires separate (and perhaps repeated) training for each degree of reduction. Software that allows running a "batch," where the number of nodes in a hidden layer is sequentially varied, and each network configuration is trained multiple times, makes this process easy and automated for the user.

Conclusions

A linear bottleneck layer that learns to preprocess the inputs to an NN during its training has several benefits, especially when available data are limited and possibly plagued by functional dependencies between variables. Without autoassociation the benefits are in model parsimony, linearly constructed reduced (indicator) variables whose construction is easy to document, and in evaluation of the saliency of the independent variables for feature selection.

Explicit models that calculate the dependent variables in one pass, given the independent ones, can be so constructed that multicollinearity in training data is modeled and can be automatically checked for when the model is applied with new input data. At the same time noise in the input variables is observed during training, and the fitting is reminiscent of total least squares.

Implicit models can be built by autoassociation of NLFA type, when it is not clear that some variables depend on others as single-valued functions. The application of implicit models requires iterative numerical solution. However, with continuous increase in computing power, such cost of model evaluation decreases in proportion to the value of better performing models. We anticipate that implicit modeling will gain in popularity in the near future.

The progression from simple to more complex models can serve as a step-wise approach to exploration in practical data-driven modeling.

References

Barron, A. 1993. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. Information Theory* 39(3):930-945

Caruana, C. 1997. Multitask Learning. *Machine Learning*, 28:41-75

Cottral, L., Stephansen, E., Tyden, A., Rosen, W., and Nybergh, B., 1954. Comparative beating investigation with different laboratory beating apparatus for pulp evaluation. *Proc. Tech. Sect., BPBMA* 35(3):359

Diamantaras, K., and Kung, S. 1996. *Principal Component Neural Networks: Theory and Applications*, NY: Wiley.

Howard, R., Poole, R., and Page, D. 1994. Factor analysis applied to the results of a laboratory beating investigation. *J. Pulp and Paper Science* 20(5):J137

Jain, A.K., Duin, R.P.W., Mao, J. 2000. Statistical Pattern Recognition: A Review. *IEEE Trans. Pattern Analysis and Machine Intelligence* 22(1):4-37

Karrila, S. 2002. Non-Linear Factor Analysis (NLFA) with Feedforward Networks Performs Non-Linear Data Reduction with Extraction of Linear Scores. In *Computational Intelligence and Applications*, 9-15. Atlanta, GA: Dynamic Publishers, Inc.

Karrila, S., Rezak, S. 2002. Review, Developments and Pulp and Paper Research Applications of Data Reduction with Neural Networks. In *Proceedings of TAPPI Paper Summit Meeting*, Atlanta, GA, March 4-6.

Kramer, M. 1991. Nonlinear principal component analysis using autoassociative neural networks. *AIChE J.*, 37(2):233-243.

Principe, J., Euliano, N., and Lefebvre, W., 2000. *Neural and Adaptive Systems*. John Wiley and Sons.

Verveer, P., Duin, P. 1995. An evaluation of intrinsic dimensionality estimators. *IEEE Trans. Pattern Analysis and Machine Intelligence* 17(1):81-86

Villiers, J. de, Barnard, E. 1992. Backpropagation neural nets with one and two hidden layers. *IEEE Trans. Neural Networks* 4(1):136-141

Wan, E., and Beaufays, F. 1996. Diagrammatic derivation of gradient algorithms for neural networks. *Neural Computation* 8(1):182-201