Feature network models for proximity data : statistical inference, model selection, network representations and links with related models

Frank, L.E.

Chapter 4

Feature Selection in Feature Network Models: Finding Predictive Subsets of Features with the Positive Lasso [1]

Abstract

A set of features is the basis for the network representation of proximity data achieved by Feature Network Models (FNM). Features are binary variables that characterize the objects in an experiment, with some measure of proximity as response variable. Sometimes features are provided by theory and play an important role in the construction of the experimental conditions. In some research settings, the features are not known a priori. This paper shows how to generate features in this situation and how to select an adequate subset of features that takes into account a good compromise between model fit and model complexity, using a new version of Least Angle Regression that restricts coefficients to be nonnegative, called the Positive Lasso. It will be shown that features can be generated efficiently with Gray codes that are naturally linked to the FNM. The model selection strategy makes use of the fact that FNM can be considered as a univariate multiple regression model. A simulation study shows that the proposed strategy leads to satisfactory results if the number of objects $\leqslant 22$. If the number of objects is larger than 22, the number of features selected by our method exceeds the true number of features in some conditions.

4.1 Introduction

Feature Network Models or FNM (Heiser, 1998) are graphical models that represent proximity data in a discrete space while using the same formalism that is the basis of least squares methods used in multidimensional scaling. A typical application area for FNM would be cognitive psychology where one studies how human cognition

[1] This chapter has been accepted for publication as: Frank, L. E. & Heiser, W. J. (in press). Feature selection in Feature Network Models: finding predictive subsets of features with the Positive Lasso. *British Journal of Mathematical and Statistical Psychology.* With an exception for the notes in this chapter and Figure 4.2, which are reactions to remarks made by the members of the promotion committee.

processes stimuli by analyzing the ratings of perceived (dis)similarity of these objects. If N respondents evaluate the dissimilarity of m objects and T binary features characterize these objects, the number of features in which two objects are distinct yields a dissimilarity coefficient that can be used as a structural model to be fitted to the data. The additivity properties of networks make it possible to consider the model as a univariate multiple linear regression problem with positivity restrictions on the parameters. The positivity restrictions are necessary because the parameters represent edge lengths in the network representation of the models.

Least squares and multiple linear regression estimates have been frequently applied in models that are related to FNM, like, for example, extended similarity trees (Corter & Tversky, 1986) and additive clustering with ADCLUS (Shepard & Arabie, 1979) or with MAPCLUS (Arabie & Carroll, 1980). However, the possibilities offered by multiple linear regression have not been fully explored in the context of these models. For instance, statistical inference is not common practice for these clustering and tree models. Recently, theoretical standard errors were introduced and used to construct confidence intervals for the parameters of the FNM (Frank & Heiser, in press a) and related additive trees (Frank & Heiser, 2004) using the theory of nonnegative least squares. In this article, we use the multiple linear regression framework for the selection of a subset of features that constitutes a good compromise between model fit and model complexity.

Before introducing the feature selection strategy proposed in this work, more has to be said about the nature of the features and the feature sets that are used to represent the proximities. The concept of a feature was introduced in psychology by Tversky (1977) who proposed the Contrast Model (CM) to describe the similarity between two objects in terms of a linear combination of the features they share (common features) and the features that distinguish between them (distinctive features). The Contrast Model in its most general form has been used in practice with a priori features only (Gati & Tversky, 1984; Keren & Baggen, 1981), but many models have been developed since, which search for either the common features part or the distinctive features part of the model, or a combination of both. Models based on common features are additive similarity trees (Sattath & Tversky, 1977) and additive clustering (ADCLUS, Shepard & Arabie, 1979; MAPCLUS, Arabie & Carroll, 1980; CLUSTREES, Carroll & Corter, 1995). The distinctive features are used for the extended similarity trees (EXTREE) proposed by Corter and Tversky (1986). A model that has the closest relation to the CM is the Modified Contrast Model (MCM) developed by Navarro and Lee (2004) that aims at finding a set of both common and distinctive features that best describes the data. This model comprises an implementation of Tversky's Contrast Model as well as the Common Features (CF) and the Distinctive Features (DF) models, that are special cases of both CM and MCM. FNM is based on the distinctive features only.

All aforementioned methods aim at finding a set of features that does not necessarily have a nested structure as required in hierarchical trees and additive trees. Rather, a less restricted structure of possibly overlapping clusters or features is sought. The FNM is the only model that represents this overlapping feature structure by a network representation. To find such a feature structure, we propose a strategy that is related to the predictor selection problem in the multiple regression

framework. The basic idea is to generate a very large number of features (or if possible, the complete set of features) first, and then select the best set of features with a subset selection algorithm. We used the Lasso option of the Least Angle Regression (LARS) algorithm (Efron et al., 2004), a recently developed efficient model selection algorithm that is less greedy than the traditional forward selection methods used in the multiple linear regression context, in the sense that the traditional methods have the tendency to eliminate useful predictors that happen to be correlated with the predictor selected in the previous step. We modified the Lasso option of this algorithm into a Positive Lasso to meet the positivity constraints of our model.

The large number of features presented to the subset selection algorithm is generated using the Gray code, a cyclic permuted version of the usual binary code, which will be explained below. Gray codes have a natural link with the network representation of the feature profiles for the objects in the FNM. In addition, the symmetry property of distinctive features leads to a very efficient use of the Gray codes because only half of the total number of codes is sufficient to enumerate the set of all possible distinctive features. This property allows in practice for enumerating the total number of features for numbers of objects m smaller or equal to 22. If m exceeds 22 and complete enumeration is no longer possible, we propose the use of a very large sample of Gray codes combined with a filter technique to reduce the number of features before using subset selection.

The strategy proposed here is different from the algorithms for the other methods because it approaches the problem of finding an adequate set of features from a different angle: most methods search for sets of features while fixing the number of features in advance. Typically, several solutions with different numbers of features are generated, and the best set of features is selected based on criteria such as goodness-of-fit and interpretability. The first application of FNM used a cluster differences scaling algorithm (Heiser, 1998) with number of clusters equal to two, which constitutes a one-dimensional MDS problem with the coordinates restricted to form a bipartition. It is still a hard combinatorial problem, and, therefore the implementation uses a nesting of several random starts together with K-means type of reallocations. The strategy proposed in this paper incorporates model selection criteria during the search process, leading to a set of features that is not necessarily optimal in the current data, but that has predictive value with a balanced trade-off between goodness-of-fit and prediction accuracy. Prediction accuracy or prediction error, which can be assessed with closed form formulas or can be approximated with cross-validation techniques, has not been used yet in this context, except for the Modified Contrast Model (Navarro & Lee, 2004) that uses a forward feature selection method and a model selection criterion related to the BIC criterion.

The remainder of the article is organized as follows. The second section presents the theory of the Feature Network Models and the generation of binary features using Gray codes. The section ends with an application on a data set and a comparison of features provided by theory and features selected by the strategy we propose. The third section shows the results of a simulation study that evaluates the performance of our strategy, and the last section provides concluding remarks.

Table 4.1: Matrix of 16 English consonants, their pronunciation and phonetic features

Consonants		F_1^*	F_2	F_3	F_4	F_5	F_6	F_7
p	(**p**ie)	0	0	0	0	0	1	0
t	(**t**ie)	0	0	0	0	1	0	0
k	(**k**ite)	0	0	0	0	0	0	1
f	(**f**ie)	0	0	1	0	0	1	0
θ	(**th**igh)	0	0	1	0	1	0	0
s	(**s**igh)	0	0	1	1	1	0	0
ʃ	(**sh**y)	0	0	1	1	0	0	1
b	(**b**uy)	1	0	0	0	0	1	0
d	(**d**ie)	1	0	0	0	1	0	0
g	(**g**uy)	1	0	0	0	0	0	1
v	(**v**ie)	1	0	1	0	0	1	0
ð	(**th**y)	1	0	1	0	1	0	0
z	(**Z**ion)	1	0	1	1	1	0	0
ʒ	(vi**s**ion)	1	0	1	1	0	0	1
m	(**m**y)	1	1	0	0	0	1	0
n	(**n**igh)	1	1	0	0	1	0	0

*F_1 = *voicing*; F_2 = *nasality*; F_3 = *affrication*; F_4 = *duration*; F_5 = *place, middle*; F_6 = *place, front*; F_7 = *place, back*.

4.2 Theory

Feature Network Models

Feature Network Models (FNM) are graphical structures that represent proximity data in a discrete space. The properties of these models will be explained using a well known data set, the perceptual confusions among 16 English consonants collected by Miller and Nicely (1955). These 16 phonemes can be described by 7 articulatory features[2]: *voicing*, *nasality*, *affrication*[3], *duration*[4] and three *places of articulation* (see Table 4.1). The authors were particularly interested in which articulatory features are important in distinguishing the consonants when affected by varying signal to noise conditions. The original data consist of 17 matrices in which each cell contains the frequencies of confusion between the spoken phoneme (the rows) and the phoneme written down by the participants (the columns). Shepard (1972) pooled the data from the first six original matrices (representing 6 different signal-

[2]It should be noted that the feature set consists of 7 features instead of the 6 features used for the same data in Chapter 2. The articulatory feature *place of articulation* has three levels (*front, middle, back*) and is represented in Table 4.1 by the three binary features F_5, F_6 and F_7 as a result of dummy coding. Representing the three levels by three variables leads to multicollinearity, and as a result, the third level has been left out from the feature set in Chapter 2. In the present chapter, the technique of the (Positive) Lasso is robust to multicollinearity and therefore, the complete feature set is used.

[3]At present, phonetic experts would call this feature *friction*.

[4]The feature *duration* is not a proper phonetic feature and has been adopted arbitrarily by Miller & Nicely (1955) to distinguish the difference between {s, ʃ, z, ʒ} and the remaining consonants.

to-noise conditions) collected by Miller and Nicely and coverted the pooled data to a symmetric matrix of similarities with the transformation $\varsigma_{ij} = (f_{ij} + f_{ji})/(f_{ii} + f_{jj})$, where f denotes the frequencies of confusion. For our study, the similarities were further transformed into dissimilarities δ_{ij} by the transformation $\delta_{ij} = -\log(\varsigma_{ij})$, assuming that the similarity measures decay exponentially with distance.

The data are illustrative for the use of features provided by theory, i.e., phonetic theory describes the articulatory properties of the phonemes. In many situations, no theory is available about the objects. Features are binary variables indicating for each object whether a particular characteristic is present or absent. Features are not always intrinsically binary: any ordinal or even interval variable if categorised can be transformed into a set of binary features, using dummy coding. For example, the place of articulation has three categories to indicate the place in the mouth where the phonemes are pronounced: front, middle and back. Dummy coding produces the three features *place, front*, *place, middle*, and *place, back* (Table 4.1).

Some set theoretic properties of the binary feature matrix lead to the estimation of a distance measure that approximates the observed dissimilarities. For example, the phoneme g has feature {*voicing, place back*} and phoneme v has the features {*voicing, affrication, place front*}. The difference between the union and the intersection (= the symmetric set difference) expresses which feature g has that v does not have and vice versa: $(g \cup v) - (g \cap v) = \{$*affrication, place front, place back*$\}$. Following Goodman (1951, 1977) and Restle (1959, 1961), a distance measure that satisfies the metric axioms can be expressed as a simple count τ of the elements of the symmetric set difference, a count of the non common elements, between the stimuli O_i and O_j and becomes the *feature distance*: $d(O_i, O_j) = \tau[(O_i \cup O_j) - (O_i \cap O_j)]$.

If $\mathbf{E}$ is a binary matrix of order $m \times T$ that indicates which features t describe the m objects, as in Table 4.1, the re-expression of the feature distance in terms of coordinates is as follows (Heiser, 1998):

$$\begin{aligned} d(O_i, O_j) &= \tau[(O_i \cup O_j) - (O_i \cap O_j)] \\ &= \sum_t |\mathbf{e}_{it} - \mathbf{e}_{jt}|, \end{aligned} \tag{4.1}$$

This re-expression of the feature distance in terms of binary coordinates is also known as the *Hamming distance*. The feature distance used in FNM is a weighted version of the distance in Equation 4.1:

$$d(O_i, O_j) = \sum_t \eta_t |\mathbf{e}_{it} - \mathbf{e}_{jt}|, \tag{4.2}$$

where the weights η_t express the relative contribution of each feature.

If we string out the dissimilarities into a vector, we can use a univariate multiple linear regression model for the dissimilarities:

$$\boldsymbol{\delta} = \mathbf{X}\boldsymbol{\eta} + \boldsymbol{\epsilon}, \tag{4.3}$$

where $\boldsymbol{\delta}$ is a $n \times 1$ vector with dissimilarities, $\mathbf{X}$ is a known $n \times T$ binary $(0, 1)$ matrix of rank T, with n equal to all possible pairs of m objects, i.e., $\frac{1}{2}m(m-1)$, $\boldsymbol{\eta}$ is a $T \times 1$

vector with feature discriminability parameters, and $\boldsymbol{\epsilon}$ is a $T \times 1$ vector. We assume that $\boldsymbol{\epsilon}$ is a $n \times 1$ random vector that follows a normal distribution,

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mu, \sigma^2 \mathbf{I}), \tag{4.4}$$

where $\mathbf{I}$ is an identity matrix of rank n, and where it is assumed that σ^2 is small enough to ensure the occurrence of negative dissimilarities to be negligible. The feature parameters are estimated by minimizing the following nonnegative least squares loss function:

$$\min_{\boldsymbol{\eta}} = \|\boldsymbol{\delta} - \mathbf{X}\boldsymbol{\eta}\|^2 \qquad \text{subject to } \boldsymbol{\eta} \geq 0, \tag{4.5}$$

where the feature parameters $\boldsymbol{\eta}$ are constrained to be positive because they represent edge lengths in the network representation of the network, as will be explained in the next paragraph. To be able to express the loss function of the FNM in a more convenient multiple regression problem as done in Equation 4.5, the original matrix $\mathbf{E}$ must be transformed first. The matrix $\mathbf{X}$ is obtained by applying the following transformation on the rows of matrix $\mathbf{E}$ for each pair l of the total of n pairs of objects, where the elements of $\mathbf{X}$ are defined by:

$$x_{lt} = |e_{it} - e_{jt}|, \tag{4.6}$$

where the index $l = 1, \cdots, n$ varies over all pairs (i, j). The result is the binary $(0, 1)$ matrix $\mathbf{X}$, where each row contains the featurewise distances for each pair of objects, with 1 meaning that the feature is distinctive for a pair of objects. It is important to notice that features become truly distinctive features only after this transformation, while the features in the matrix $\mathbf{E}$ are not inherently common or distinctive. The weighted sum of these featurewise distances is the fitted distance for each pair of objects and is equal to $\hat{\mathbf{d}} = \mathbf{X}\hat{\boldsymbol{\eta}}$. Transforming the objects $\times$ feature matrix to the object pairs $\times$ features matrix is necessary to apply the multiple regression approach and, at the same time, provides a considerable reduction of the number of features to be generated, as will become clear later.

The multiple regression approach has been used earlier in the context of the common features model (Arabie & Carroll, 1980), and for tree models (Corter, 1996). However, the nonnegative least squares method has not been used by these authors, although they were aware of the problem. Only Arabie and Carroll (1980) address the problem by implementing a subroutine in the MAPCLUS algorithm that encourages the weights to become positive. These authors explain that the use of nonnegative least squares has been avoided explicitly because in the context of the iterative algorithm that is the basis of the MAPCLUS algorithm, it would reduce the number of clusters in the solution. We implemented the nonnegative least squares option in PROXGRAPH (the program used to fit FNM), not during the feature selection procedure, but for the situation where the features are supplied by the user. In that case, the use of nonnegative least squares has a considerable advantage because it opens the way to statistical inference by providing theoretical standard errors for the feature parameters (Frank & Heiser, in press a).

Table 4.2: Feature parameters ($\hat{\eta}$), standard errors, and 95% confidence intervals for Feature Network Model on *consonant* data with $R^2 = 0.61$

Features	$\hat{\eta}$	$\hat{\sigma}_\eta$	95% CI	
Constant	2.11	0.13	1.85	2.37
Voicing	1.22	0.11	1.01	1.43
Nasality	0.81	0.13	0.56	1.06
Affrication	0.12	0.11	-0.11	0.34
Duration	0.32	0.12	0.10	0.55
Place, middle	0.00	0.00	0.00	0.00
Place, front	0.10	0.07	-0.04	0.24
Place, back	0.26	0.10	0.06	0.45

Table 4.2 shows the feature discriminability parameters that result from minimizing the loss function in Equation 4.5, as well as the corresponding standard errors and 95% confidence intervals. The method to compute the standard errors and 95% *t*-intervals for inequality constrained feature parameters in the context of Feature Network Models has been described in (Frank & Heiser, in press a). The model with seven features has an $R^2 = 0.61$, and the values of the feature parameters lead to the conclusion that the most important categorizing criteria used by the participants were the following: *voicing*, *nasality*, *duration*, and *place, back*. The features *affrication*, *place, middle*, and *place, front* do not play an important role as follows from the 95% *t*-confidence intervals that show that the feature parameters of these features do not significantly differ from zero (see Table 4.2).

The feature distance parallels the path-length distance in a valued graph if one of the metric axioms, the triangle inequality, is reaching its limiting additive form $d_{ij} = d_{il} + d_{jl}$ when l is on the shortest path from i to j (Flament, 1963; Heiser, 1998). Hence, sorting out the additivities in the fitted feature distances and excluding edges that are sums of other edges results in a parsimonious subgraph of the complete graph. Figure 4.1 shows the Feature Network representation that results from the fitted distances on the *consonant* data. The phonemes are the vertices in the network and the estimated feature distances ($\hat{\mathbf{d}} = \mathbf{X}\hat{\boldsymbol{\eta}}$) are represented as additive counts of edge lengths in the graph, where the edge lengths are the feature parameters $\hat{\boldsymbol{\eta}}$. For display purposes the 7-dimensional feature network has been embedded in 3-dimensional Euclidean space using PROXSCAL[5] (a multidimensional scaling program distributed as part of the Categories package by SPSS, Meulman & Heiser, 1999). The solution of the common space was restricted by a linear combination of the feature variables, to be able to represent the features as vectors in the same space. The final network representation was obtained using the default options for 3-D plotting in Matlab. The three most important features (*voicing*, *nasality*, and *duration*) are represented as vectors in Figure 4.1, leading from the origin through the point with coordinates equal to the correlations of each feature with each of the three dimensions. The network clearly shows the importance of the *voicing* feature:

[5] with the interval transformation option and initialized with the simplex solution

all voiced consonants are on the left part of the network and well separated from the unvoiced consonants on the right part. The second important feature, *nasality*, separates the consonants *m* and *n* from the other consonants. The consonants *s*, ʃ, *z* and ʒ form a group with the shape of a rectangle and differ from the remaining 12 consonants because of the length of their pronunciation, described by the feature *duration*. The plus and minus signs on each vector designate the projection onto the vector of the centroids of the consonants that possess the feature (+) and the consonants that do not possess that feature (−).

Generating features with Gray codes

Given that features can be viewed as binary variables, a very straightforward way to produce all possible binary (0,1) features for m objects is to generate the binary codes for m bits of the integers 0 to $2^m - 1$, as illustrated in Table 4.3 for $m = 4$. Another, more restrictive way to produce the binary features, is to use the Gray code (Gray, 1953). A Gray code represents each number in the sequence of integers $\{0 \cdots 2^m - 1\}$

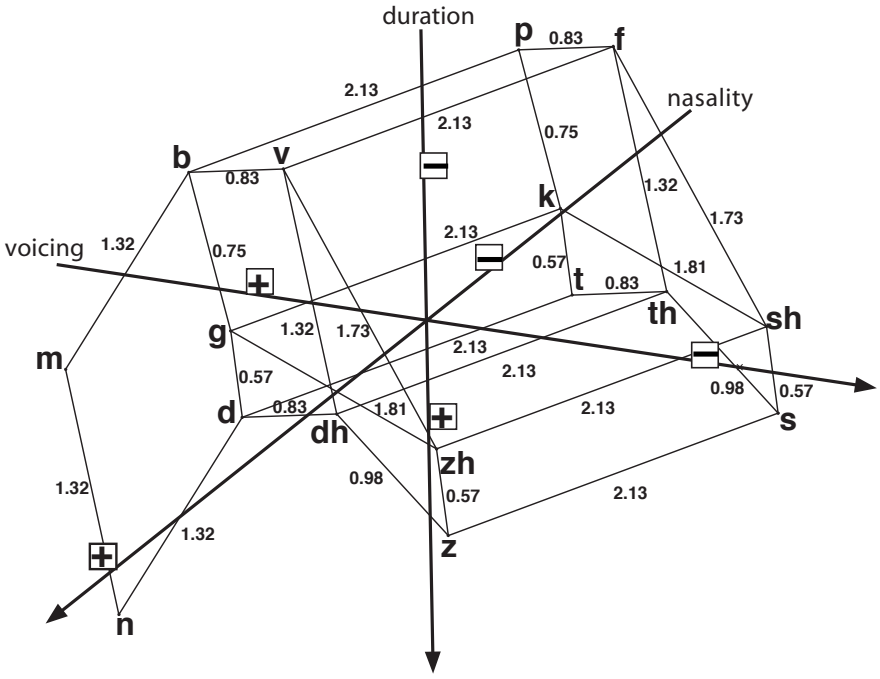

Figure 4.1: Feature Network representation for the *consonant* data with the three most important features (*voicing*, *nasality*, and *duration*) represented as vectors. The plus and minus signs designate the projections onto the vector of the centroids of the objects that possess the feature (+) and the objects that do not have that feature (-). (dh = ð; zh = ʒ; th = θ; sh = ʃ).

Table 4.3: Binary code and Gray code for 4 bits

Integer	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

as a binary vector of length m in an order such that adjacent integers have Gray code representations that differ only in one bit position, meaning that the transition from one integer to the next in the order requires changing just one bit at a time, which is called is called the *adjacency property* (*cf.* Gardner, 1972).

Table 4.3 shows the Gray coding corresponding to the integers based on 4 bits. There is a specific relationship between the Gray code and the binary code: Gray codes are binary codes arranged in a special order. Table 4.3 clearly shows the pattern of the flipping of one bit at a time, compared to the binary codes, where the transition from one integer to the next in the order is not restricted to the change of one bit. There are many ways to produce a binary sequence that has the adjacency property. The most common way to produce such a sequence is the so called *binary reflected Gray code* that starts with all bits zero and successively flips the right-most bit that produces a new string. Generating the binary reflected Gray codes works as follows. For m bits the list L_m starts with L_1 and produces the list $0, 1$. For $m > 1$ bits, L_m is formed by taking the first half of the list, L_{m-1} prepending a 0 to every number, then following that list by the reverse of L_{m-1} with a 1 prepended to every number (*cf.* Savage, 1997). For example, to obtain L_2, the list L_1 (0,1) is written forwards and backwards, producing 0, 1, 1, 0, and, prepending 0's to the first half and 1's to the second half, yields the L_2 list 00, 01, 11, 10.

In contrast to the more arbitrary binary codes, the Gray codes are directly related to the Feature Network Models. An m-bit Gray code is equal to a Hamiltonian cycle on an m-dimensional hypercube (Gilbert, 1958; Savage, 1997). It represents all the possible feature combinations for m objects. It is a cycle that visits each combination only once. The feature distance is a city-block metric on the binary coordinates of this same space. Since adjacent Gray codes differ by only one bit, feature distances

of three consecutive Gray codes are additive. The binary coordinates represent the feature pattern of each object. Based on their feature pattern, the objects have their place in this m-dimensional hypercube.

The Gray code shares with the binary code the nice property that the second half of the list of the codes is the complement of the first half. Table 4.3 shows this property: the pattern of the Gray codes (and the binary code) representing the integers 8 to 15 is the negative of the pattern of the integers 0 to 7. The fact that the second half of the Gray codes for m objects is the complement of the first half constitutes a useful property in the context of the Feature Network Models. Since complementary features yield the same $\mathbf{X}$-matrix and consequently the same $\hat{\eta}$ values as the original features, only half of the number of 2^m features needs to be generated. This property holds for the distinctive features only, and not for the common features, where it would be necessary to generate the complete Gray code. Further reductions can be obtained by discarding the feature with zeros only, because it has no meaning in the Feature Network Models. The feature with ones only (the universal feature) is always located in the second half of the Gray code and will therefore not be part of the set of generated features. However, PROXGRAPH, the program used to fit FNM (programmed in Matlab), has the option of adding the universal feature to the model. A remark has to be made about a special category of features, the unique features, which describe only one object (having a 1 for that particular object and zero values for the remaining objects). In the common features model the presence of one or more unique features in the object $\times$ features matrix $\mathbf{E}$ leads to a zero feature product in the predictor set and is one of the problems to be avoided in, for example, the MAPCLUS algorithm (Arabie & Carroll, 1980). The FNM that use featurewise distances does not have this inconvenience and therefore all Gray codes representing the unique features can be part of the complete feature set.

Summarizing, complete enumeration of the distinctive features for m objects amounts to generating $\frac{1}{2}(2^m) - 1$ Gray codes, using the integers $\{1 \cdots \frac{1}{2}(2^m) - 1\}$, and forming the complete set of featurewise distances $\mathcal{D}$ that contains a total number of $T_{\mathcal{D}} = \frac{1}{2}(2^m) - 1$ predictors. Taking the set $\mathcal{D}$ as the starting point of the feature subset selection process constitutes a considerably smaller problem than would be the case for the generation of predictors in a univariate multiple regression problem with arbitrary binary predictors. In that case the number of predictors to be enumerated amounts to 2^n, where n is equal to $\frac{1}{2}m(m-1)$, which shows the proportion of additional predictors that are needed. The possibility of using the transformation from the matrix $\mathbf{E}$ to the matrix $\mathbf{X}$ allows for this reduction of the number of predictors to be generated.

However, there are limitations to the total number of distinctive features that can be handled because the set $T_{\mathcal{D}}$ of predictors grows considerably with increasing number of objects m. For example, for $m = 20$ the set $T_{\mathcal{D}}$ contains $\frac{1}{2}(2^m) - 1 = \frac{1}{2}(2^{20}) - 1$ or about a half million distinctive features, growing to about 1 million for $m = 21$, becoming more than 2 million for $m = 22$ and exceeding 4 million for $m = 23$. A set of $\frac{1}{2}(2^{22}) - 1$ (= about 2 million) predictors is the maximum number that the current implementation of the predictor selection algorithm, the Positive Lasso, can handle simultaneously.

To generate the Gray codes within PROXGRAPH, we used a Matlab transcription by Burkardt of the original algorithms for generating Gray codes in Nijenhuis and Wilf (1978) (see http://www.csit.fsu.edu/ burkardt/, Fortran and C++ files of the same algorithms are also available at this site). Both binary code and Gray code have the convenient attribute that features can be (re)produced by a simple integer or rank number. This property saves computer memory because it is not necessary to save the entire sequence of $\frac{1}{2}(2^m) - 1$ features, since the original feature set can be retrieved by simply keeping track of the corresponding integer or rank number. Another advantage of saving the integer or rank numbers is the possibility of getting back the original features after transformations to featurewise distances have been applied on those features. In the Feature Network Models one important transformation performed on the features is the transformation from the $(m \times T)$ matrix $\mathbf{E}$ representing the T features that describe the m objects to the matrix $\mathbf{X}$ of size $(n \times T)$, that contains the symmetric set difference for each of the $n = \frac{1}{2}m(m-1)$ object pairs (Equation 4.6). This matrix $\mathbf{X}$ is also the format for the features when submitted to the feature selection algorithm. The problem with this transformation is that it is not reversible because the results are not unique. In the simple example of one feature and two objects the result 0 can come from $\mathbf{x}_{12} = |1-1|$, where both objects have the feature, or from $\mathbf{x}_{12} = |0-0|$, where neither of the objects possesses the feature. The result 1 is not unique either. It means that one of the two objects has the feature, but it is not clear which object has the feature. Therefore, saving the rank numbers of the features in the set before applying the transformation, makes it possible to reproduce the original feature matrix at the end of the entire feature selection process.

Selecting a subset of features with the Positive Lasso

Above we have shown that the FNM can be considered as a univariate multiple linear regression problem with positivity constraints on the feature discriminability parameters (see Equations 4.3, 4.4, and 4.5). When the features are known in advance and their number is reasonably small, the feature discriminability parameters can be obtained directly by minimizing the nonnegative least squares loss function of Equation 4.5. However, in the case of unknown features, the Gray codes are used to generate a very large number of features, and, as a result, the simple nonnegative least squares loss function cannot be used. The large number of features calls for a variable selection method. There are many methods available for variable selection, see for example a recent review by Guyon and Elisseeff (2003). Given the multiple regression context of the FNM, we have chosen the least absolute shrinkage and selection operator (Lasso). The Lasso is a constrained version of ordinary least squares (OLS) and minimizes the residual sum of squares subject to the sum of the absolute value of the coefficients being less than a constant (Tibshirani, 1996). Let $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T$ be n-vectors representing the T featurewise distances with n equal to the number of unique pairs of objects, and $\boldsymbol{\delta}$ the n vector of dissimilarities. It is assumed that the featurewise distances have been standardized to have mean 0 and

unit length and that the response variable ($\boldsymbol{\delta}$) has mean 0:

$$\sum_{l=1}^{n} \delta_l = 0, \quad \sum_{l=1}^{n} x_{nt} = 0 \quad \text{and} \quad \sum_{l=1}^{n} x_{lt}^2 = 1 \quad \text{for} \quad t = 1, 2, \cdots, T. \tag{4.7}$$

Applied to the context of the FNM, the Lasso loss function can be written in the following way:

$$\min_{\boldsymbol{\eta}_{\text{L}}} = \|\boldsymbol{\delta} - \mathbf{X}\boldsymbol{\eta}_{\text{L}}\|^2 \qquad \text{subject to} \quad G(\boldsymbol{\eta}_{\text{L}}) \leqslant b, \tag{4.8}$$

where the constraint $G(\boldsymbol{\eta}_{\text{L}}) = \sum_{t=1}^{T} |\eta_t|$ and $b \geq 0$ is the tuning parameter that controls the amount of shrinkage. If $\hat{\boldsymbol{\eta}}^0$ is the vector of ordinary least squares estimates and $b_0 = \sum_{t=1}^{T} |\hat{\eta}_t^0|$, the Lasso estimates become the ordinary least squares estimates for values of $b > b_0$. On the other hand, values of $b < b_0$ will cause shrinkage of the solutions toward 0, and some of the coefficients will become exactly equal to 0. This effect constitutes the parsimony property that characterizes the Lasso compared to ridge regression, which is related to the Lasso and is probably more generally known.

For any given constraint value b in the path of Lasso solutions[6], only a subset of the features has non-zero values of the regression coefficients $\hat{\eta}$. While ridge regression also shrinks coefficients, it does not, however, set any coefficients to 0 and, as a result, does not lead to more simple models. The differences in the nature of shrinkage between the Lasso and ridge regression result from the constraints used in both methods. Both methods use the residual sum of squares loss function, but where the Lasso uses the constraint $\sum_{t=1}^{T} |\eta_t|$, ridge regression uses $\sum_{t=1}^{T} \eta_t^2$ instead (see for more details: Hastie et al., 2001; Tibshirani, 1996). From the viewpoint of geometry, the Lasso constraint leads to a constraint region with corners and flat edges, while ridge regression leads to round shaped constraint regions, see Figure 4.2. The residual sum of squares function has elliptical contours and both methods find the first point where these elliptical contours hit the constraint region. In the case of the Lasso, when the elliptical contours hit a corner, some of the estimated parameters become exactly 0, while in the case of ridge regression the estimated parameters will never become 0 because the elliptical contours will never hit a corner.

In general, shrinkage improves prediction accuracy, trading off decreased variance for increased bias, Hastie et al. (2001). For the special case of the Lasso, shrinkage leads to more parsimonous models because some coefficients become exactly zero. Another advantage of the Lasso, especially useful for the FNM context, is that it does not suffer from overfit or highly correlated settings because it avoids the explicit use of the OLS estimates. This means that the design matrix $\mathbf{X}$ need not be of full rank, which is very convenient in a situation with a very large number of featurewise distances.

[6] In contrast to ordinary least squares, the Lasso does not yield a single solution but a path of solutions depending on the values of the tuning parameter b. Typically, Lasso solutions are computed for several values of b, ranging from $b = 0$ to $b = b_0$. An example of a path of Lasso solutions can be viewed in Figure 4.3, starting with $b = 0$, which forces all coefficients to become zero, and ending with the value of $b = b_0$ equal to the sum of the coefficients of the ordinary least squares solution.

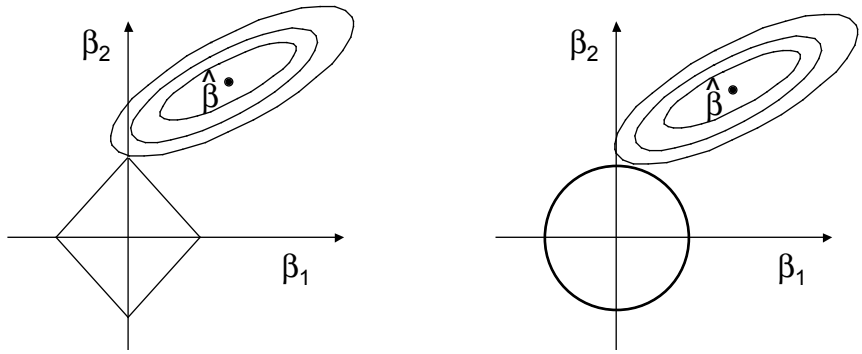

Figure 4.2: Graphs of estimation for the Lasso (left) and ridge regression (right) with contours of the least squares error functions (the ellipses) and the constraint regions, the diamond for the Lasso and the disk for ridge regression. The corresponding constraint functions are equal to $|\beta_1| + |\beta_2| \leqslant b$ for the Lasso and $\beta_1^2 + \beta_2^2 \leqslant b^2$ for ridge regression . It is clear that only the constraint function of the Lasso can force the $\hat{\beta}$-values to become exactly equal to 0. (The graphs are adapted from Hastie et al. (2001), p. 71).

The computation of Lasso solutions is a quadratic programming problem, and can be solved by numerical analysis algorithms, but the LARS or Least Angle Regression (Efron et al., 2004) is a better approach. The LARS algorithm works as follows. It starts with all feature parameters of the vector $\hat{\boldsymbol{\eta}} = (\hat{\eta}_1, \hat{\eta}_2, \cdots, \hat{\eta}_T)'$ equal to 0. The next step is to find the predictor $\mathbf{x}_t$ most correlated with response $\boldsymbol{\delta}$, which is added into the model. The residuals $\mathbf{r} = \hat{\mathbf{d}} - \boldsymbol{\delta}$ are calculated and the parameter $\hat{\eta}_t$ is increased in the direction of the sign of its correlation with $\boldsymbol{\delta}$ until some other feature $\mathbf{x}_k$ has as much correlation with the current residual vector as does $\mathbf{x}_t$. The feature parameters $(\hat{\eta}_t, \hat{\eta}_k)$ are increased in their joint least squares direction, until some other feature $\mathbf{x}_q$ has as much correlation with the current residual. The just described steps are repeated until all features have been entered in the model and the process stops when $\text{corr}(\mathbf{r}, \mathbf{x}_t) = 0 \; \forall \; t$, which corresponds to the OLS solution. In the situation where the number of predictors T exceeds the number of observations n, the LARS algorithm terminates at the saturated least squares fit after $n - 1$ predictors have entered the active set (see, for more details, Efron et al., 2004, p. 444). The number $n - 1$ follows from mean centering the columns of the matrix of predictors $\mathbf{X}$ which results in a row-rank equal to $n - 1$. It should be noted however, that although the model contains no more than $n - 1$ predictors, the number of *different* predictors that have entered the model during the complete sequence of solutions is typically greater than $n - 1$.

LARS provides an efficient way to compute the Lasso sequence of solutions simultaneously for all values of b, as b varies from 0 to infinity by applying the following modification: if a non-zero parameter becomes zero, it is removed from the active set of features and the joint direction is recomputed. The implementation

of LARS in *R* also allows for the transformation of the Lasso into a Positive Lasso necessary for the FNM where all feature discriminability parameters should be positive. We applied the procedure described in Efron et al. (2004, section 3.4, p. 421) to the LARS algorithm programmed in *R*. The result is the solution of the following minimization function:

$$\min_{\boldsymbol{\eta}_{\text{PL}}} = \|\boldsymbol{\delta} - \mathbf{X}\boldsymbol{\eta}_{\text{PL}}\|^2 \quad \text{subject to} \quad G(\boldsymbol{\eta}_{\text{PL}}) \leqslant b \quad \text{and all} \quad \eta_t \geqslant 0, \tag{4.9}$$

where the constraint in Equation 4.8 is extended with a positivity constraint for the feature discriminability parameters.

Selecting the number of features with an AIC criterion

The Lasso and the Positive Lasso do not yield a single solution $\hat{\boldsymbol{\eta}}$, but a path of possible solutions defined by the continuum depending on the values of the tuning parameter b, which represents the amount of shrinkage. Choosing a value for b leads automatically to the choice of the number of features in the model, i.e. the number of features with nonzero $\hat{\eta}$-values. The problem is to choose a good value for the a priori unknown b, such that the corresponding model minimizes the prediction error.

Efron et al. (2004) proved that for some LARS estimators, the best value for b can be found with an adaptation of Mallows' C_p statistic (Mallows, 1973, 1995), where the step number of the LARS algorithm is used as an estimate for the degrees of freedom of the corresponding model. For the Lasso and the Positive Lasso estimators, the step number of the LARS algorithm cannot be used as an estimate for the degrees of freedom because the total number of steps can exceed the total number of predictors in the full model. However, recently, Zou, Hastie, and Tibshirani (2006) showed that the number of non-zero coefficients is an unbiased estimate for the degrees of freedom for the Lasso, an informative measurement of model complexity, with no special assumptions on the predictors. This estimate for the degrees of freedom in the Lasso can be used to estimate the prediction error of each of the models along the path of Lasso solutions by the following AIC criterion, derived especially for the Lasso (Zou et al., 2006):

$$AIC_{\text{L}} = \frac{\|\boldsymbol{\delta} - \hat{\mathbf{d}}\|^2}{n} + \frac{2}{n}\widehat{df}(\hat{\mathbf{d}})\sigma^2_{\text{L}}, \tag{4.10}$$

where $\hat{\mathbf{d}} = \mathbf{X}\hat{\boldsymbol{\eta}}_{\text{L}}$, and the error variance σ^2_{L}, if unknown, is replaced with an estimate based on the largest model. In the case where the number of predictors exceeds the number of observations, the largest model in the total sequence of Lasso solutions resulting from the LARS algorithm, involves at maximum $n-1$ predictors. Since the largest model is a (nearly) saturated model, the error variance is very close to zero. Therefore, we estimated σ^2_{L} in Equation 4.10 by taking the mean of the error variances of all models in the sequence of Lasso solutions.

The AIC_{L} criterion, which approximates the Mallows' C_p statistic (Mallows, 1973, 1995) closely, has been shown to offer substantially better accuracy than cross-validation and related nonparametric methods, if one is willing to assume the model

is correct (Efron et al., 2004; Zou et al., 2006). We used the AIC_{L} criterion to select the best model for the Positive Lasso solutions and to assess the prediction error, assuming that the theory about the effective number of non-zero parameters applies for the Positive Lasso as well. To our knowledge it is the best criterion available at the moment.

Figure 4.3 shows the results of the modifications of the Lasso-LARS algorithm into the Positive Lasso as in Equation 4.9 using the theoretical (phonetic) features of the *consonant* data (Table 4.1). The left top panel shows the paths of the Lasso estimates of the feature discriminability parameters against the degrees of freedom expressing the effective number of nonzero parameters. Feature 5 (*place, middle*) obtains negative feature discriminability parameters along the path. The right top

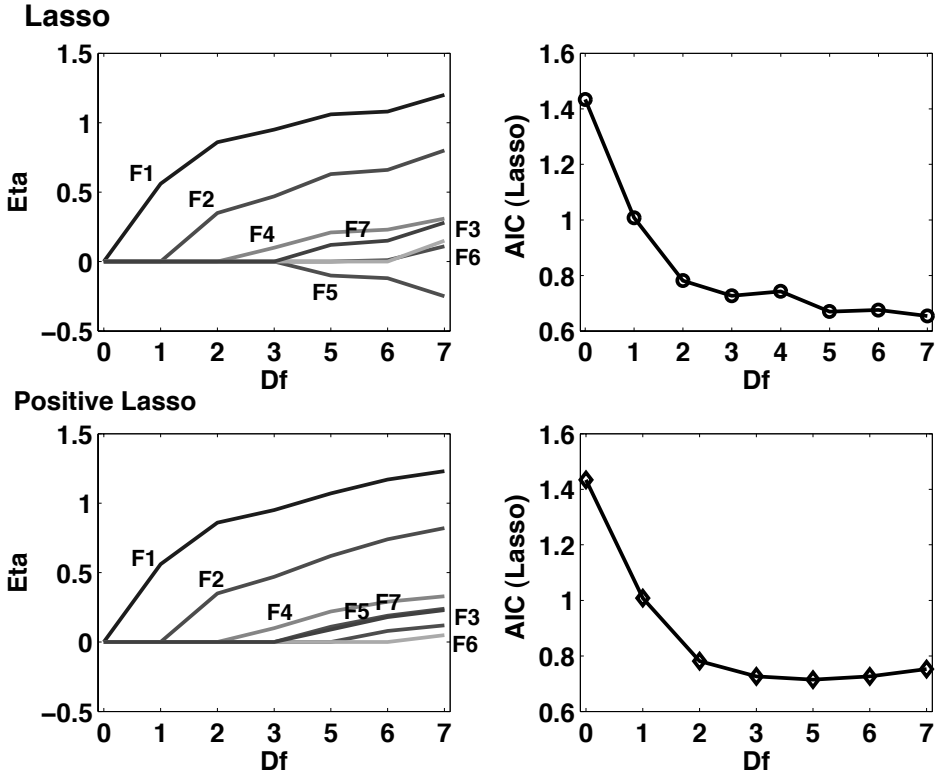

Figure 4.3: Estimates of feature parameters for the *consonant data*. *Top panels*: trajectories of the Lasso estimates $\hat{\eta}_{\mathrm{L}}$ (left panel) and the AIC_{L} values plotted against the effective number of parameters (= df) of the Lasso algorithm (right panel). The model with lowest AIC_{L} value (= 0.65) contains all 7 features. *Lower panels*: trajectories of the Positive Lasso estimates $\hat{\eta}_{\mathrm{PL}}$ (left panel) and the adjusted AIC_{L} values plotted against the effective number of parameters (= df) of the Positive Lasso algorithm (right panel). The model with lowest AIC_{L} value (= 0.71) has 5 features.

Table 4.4: Estimates of feature discriminability parameters ($\hat{\eta}_{\text{ICLS}}$ = ICLS, $\hat{\eta}_{\text{L}}$ = Lasso, and $\hat{\eta}_{\text{PL}}$ = Positive Lasso) for the *consonant data*

Features	$\hat{\eta}_{\text{ICLS}}$	$\hat{\eta}_{\text{L}}$	$\hat{\eta}_{\text{PL}}$
intercept	2.11	2.22	2.39
Voicing	1.22	1.20	1.07
Nasality	0.81	0.80	0.62
Affrication	0.12	0.11	0.00
Duration	0.32	0.31	0.22
Place, middle	0.00	-0.25	0.11
Place, front	0.10	0.15	0.00
Place, back	0.26	0.28	0.09

panel shows the AIC_{L} values at each step of the iterations, plotted against the estimated degrees of freedom, represented by the effective number of non-zero parameters as in Equation 4.10. The AIC_{L} curve, which also represents the estimates of prediction error, shows that the best model occurs at 7 df, the model that contains all features. The complete model corresponds to the ordinary least squares solution because the Lasso always converges to it. The lower left panel shows the path of the Positive Lasso estimates of the feature discriminability parameters and it is clear that all trajectories stay in the positive part of the parameter space. The AIC_{L} curve in the lower right panel indicates that the best model occurs at $df = 5$ with only 5 of the 7 features present in the model.

Table 4.4 displays the estimates of feature discriminability parameters according to the best Lasso model and the best Positive Lasso model based on the AIC_{L} curves compared to the inequality constrained least squares estimates obtained with Equation 4.5. In this case the Lasso estimates $\hat{\eta}_{\text{L}}$ are equal to the ordinary least squares estimates, without positivity constraints, and yield a negative coefficient value for feature *place, middle*. The Positive Lasso estimates $\hat{\eta}_{\text{PL}}$ show that the two features *affrication* and *place, front* have coefficient values equal to zero as a result of activated positivity constraints, and consequently, these two features are not part of the model. This finding confirms the results of the 95% CI presented before (Table 4.2) showing that the feature parameters ($\hat{\eta}_{\text{ICLS}}$) of these two features do not significantly differ from zero[7].

[7]It should be noted that from the perspective of phonetic theory, it is rather unusual that the feature *affrication* disappears from the model and should probably be ascribed to the experimental conditions used by Miller and Nicely (1955). The authors presented the consonants under 6 different signal-to-noise conditions and, as a result, the non-fricative consonants become contaminated with noise and are no longer distinguishable from the fricatives. The same experimental conditions could also explain the fact that *voicing* has so much influence, while it is known as a phonetic feature that is easily lost during perception.

Generating features by taking a random sample combined with a filter

When the number of objects m exceeds 22, it is not possible to generate the complete set of distinctive features. In that case, we propose to take a sample from the total number of rank numbers representing the Gray codes associated with the number of objects. This sample might still be too large to be submitted to the Positive Lasso algorithm and necessitates some preselection strategy. Preselection is often used as a preprocessing step before variable subset selection. A review on this topic has been given by Guyon and Elisseeff (2003). A common preselection strategy is the ranking method that performs this preprocessing step by selecting variables that have high values on a scoring function, usually the coefficient of determination or R^2 (Guyon & Elisseeff, 2003). The variables are sorted in decreasing order based on their values on the scoring function. To build a predictor, nested subsets are constructed which incorporate progressively more variables of decreasing relevance. Other scoring functions are the correlation, where positively correlated variables are top ranked and negatively correlated variables bottom ranked. We propose the use of the regression coefficient, or, in the context of FNM, discriminability parameter $\hat{\eta}$, which is a scaled version of the correlation coefficient as can be seen in the following relation (*cf.* Draper & Smith, 1998, p. 42):

$$\hat{\eta} = \frac{s_{\boldsymbol{\delta}}}{s_{\mathbf{x}}} r_{\boldsymbol{\delta}\mathbf{x}}, \tag{4.11}$$

where $s_{\boldsymbol{\delta}}$ and $s_{\mathbf{x}}$ are the standard deviations of the dependent variable $\boldsymbol{\delta}$ and the predictor variable $\mathbf{x}$, and $r_{\boldsymbol{\delta}\mathbf{x}}$ is the correlation between the dependent variable and the predictor variable. It is clear that when both $\boldsymbol{\delta}$ and $\mathbf{x}$ are standardized, as required for the Positive Lasso, the regression coefficient is equal to the correlation.

Example of feature generation and selection on the *consonant* data

The previous section showed the results of the Positive Lasso on the a priori phonetic features of the *consonant* data. In many data analytic situations, the features are not given by theory. This section shows an example of feature generation using Gray codes followed by feature selection with the Positive Lasso on the same *consonant* data. First, all possible distinctive features were generated with the number of Gray codes equal to $\frac{1}{2}(2^{16}) - 1 = 32,767$ because there are 16 consonants, yielding a $16 \times 32,767$ matrix of objects by features. After transformation of this matrix into the $120 \times 32,767$ matrix $\mathbf{X}$ using Equation 4.6, the complete set of featurewise distances was analyzed with the Positive Lasso algorithm.

Figure 4.4 shows that the AIC_{L} curve attains its lowest value ($= 0.51$) at the model with 7 features. Table 4.5 shows the values of the feature discriminability parameters for the feature matrix obtained from phonetic theory and for the feature matrix resulting from the Positive Lasso algorithm. The model resulting from the Positive Lasso has higher fit ($R^2 = 0.70$) and lower prediction error values compared to the model based on phonetic theory.

Table 4.6 displays the features of the model selected by the Positive Lasso algorithm juxtaposed to the 7 features based on phonetic theory. Comparing the features

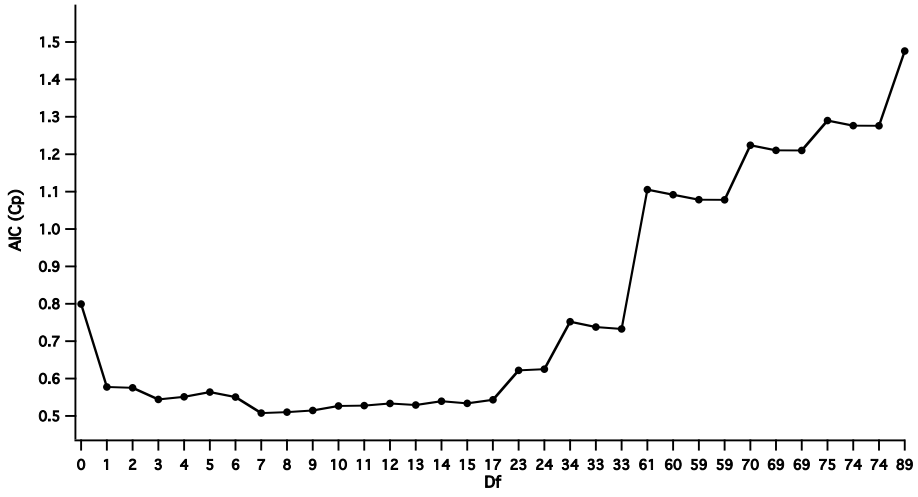

Figure 4.4: AIC_{L}-plot for the *consonant* data using all possible features generated with Gray codes ($T = 32,767$). The lowest AIC_{L} value ($= 0.51$) points to a model with 7 features.

from phonetic theory to the features resulting from the Positive Lasso, leads to the conclusion that the two feature sets are very different from each other, except for the first feature that represent the phonetic property *voicing*. The remaining features selected by the Positive Lasso do not seem to be related to the theoretic phonetic properties of the consonants. However, the network representation of the feature set selected by the Positive Lasso displayed in Figure 4.5 does make sense in terms of phonetics: there is a clear distinction between the voiced consonants (*m*, *n*, *b*, *d*, ð, *g*, *v*, *z*, ʒ) on the left part of the configuration and the unvoiced consonants on the right part. The nasals (*m*, *n*) form a distinct cluster, showing the importance of the phonetic property *nasality*. The cluster (*s*, *θ*) represents middle voiceless consonants and the cluster (ð, *b*, *v*) front and middle voiced consonants. Another cluster that can be distinguished comprise the voiceless plosives, (*p*, *t*, *k*) opposed to the three voiced plosives (*b*, *d*, *g*)[8]. The clusters just described correspond to clusters found with ADCLUS by Shepard and Arabie (1979) and with MAPCLUS by Arabie and Carroll (1980).

[8]From the perspective from phonetics, the same remarks that were made in footnote[7] for the solution of the set of 7 theoretic features in Figure 4.3 and Table 4.4 also apply to this solution.

4.3 Simulation study

In this section we report a Monte Carlo experiment that evaluated the performance of our method given that the true feature structure (the true model) is known in advance. The first question that will be addressed is the following: does the Positive Lasso select the correct subset of features given that the true feature set is known, under different data analytic conditions such as error, n/T ratio (number of object pairs compared to the number of features), and the size of the feature discriminability parameters? The performance criterion of this study is the proportion of recovery of the true features, measured by Gray code rank number. We verified the baseline condition of the proportion of recovery on error-free data, which resulted in complete recovery of the correct features for the experimental conditions. Another question addressed by the simulation study is: how does the method of random sampling from Gray codes combined with a filter perform compared to the complete enumeration method? Proportion of recovery of true features is not a useful performance criterion in this situation because random samples of features are taken and that would merely result in testing the performance of the pseudo-random number generator, instead of testing the performance of the method. Instead, the following measures serve as outcome: the effective number of parameters (Df) and the prediction error (AIC_{L}).

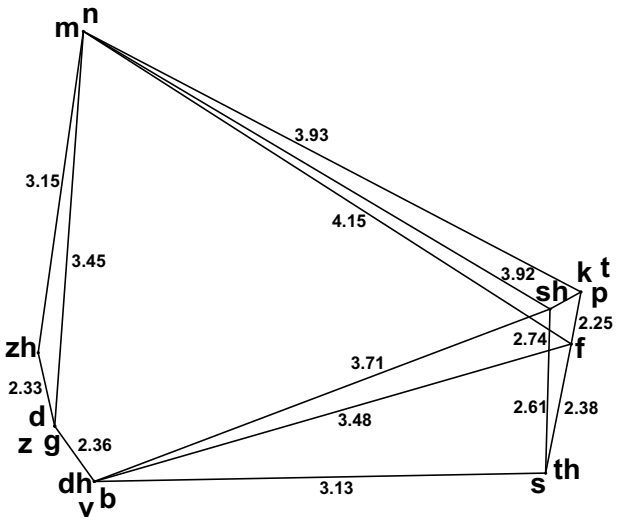

Figure 4.5: Feature Network representation for the *consonant* data based on the feature matrix selected by the Positive Lasso displayed in Table 4.6. (dh = ð; zh = ʒ; th = θ; sh = ʃ).

Table 4.5: Positive Lasso estimates, R^2, and prediction error (K-fold cross-validation) for the features from phonetic theory (left) and for the features selected from the complete set of distinctive features (right)

Features from phonetic theory		*Features selected from the complete set*	
Features	$\hat{\eta}_{\text{PL}}$	Features	$\hat{\eta}_{\text{PL}}$
intercept	2.39	*intercept*	2.03
Voicing	1.07	F1	0.91
Nasality	0.62	F2	0.33
Affrication	0.00	F3	0.30
Duration	0.22	F4	0.35
Place, middle	0.24	F5	0.22
Place, front	0.00	F6	0.36
Place, back	0.09	F7	0.19
Model fit			
R^2	0.56		0.70
Prediction error (standard error) based on K-fold cross-validation			
K=5	0.35 (0.08)		0.29 (0.04)
K=10	0.35 (0.07)		0.27 (0.05)

Method for simulation study

The experimental conditions of this simulation study result from the cross classification of five experimental variables with two levels each. For each experimental condition, a total of 50 simulation samples were generated. The first experimental variable is the *number of objects* m = 12 or 24. The second experimental variable is the ratio of the number of observations n and the number of features T and has two levels: $n/T = 16$ and $n/T = 8$. Given the two levels of *number of objects*, the number of observations n is equal to the number of object pairs $n = \frac{1}{2}m(m-1)$. For the 12 objects condition, which has $n = 66$, the number of features needed to obtain the two n/T ratios is 4 and 8. For the 24 objects condition (with n equal to 276) the number of features needed is equal to 17 and 35. The third experimental variable is the *size of the feature discriminability parameters* with two levels: medium values (M) and a combination of small and large values (S + L). Depending on the number of features needed the following patterns of 4 feature discriminability parameters is repeated. For the medium values conditions the pattern is $\{2.0, 2.5, 1.5, 3.0\}$ and for the small + large values the pattern is $\{6.0, 0.2, 0.5, 0.3\}$. The fourth experimental variable is the *amount of error* added to the data, and comes in two levels: 0.05 (low) and 0.35 (high). The fifth experimental variable is the *feature generation strategy*, which consists of either generating the whole set of possible features using half of the complete Gray code sequence, or a set of the 100 best features based on the filter criterion of largest separate $\hat{\eta}$ value selected from a large random sample (30%) of all possible features.

Using the whole set of possible features is only feasible for the 12 objects condition. As a result, the random sample strategy was applied to all 16 combinations of the experimental variables (*number of objects*, n/T ratio, *size of the feature discriminability parameters*, *amount of error*), and the complete set of features strategy was used for the 12 objects condition only, leading to 6 experimental conditions obtained from the combination of n/T ratio, *size of the feature discriminability parameters* and *amount of error*. The 50 simulation samples in each experimental condition were submitted to the Positive Lasso algorithm and the performance of the selected models was evaluated by the following measures that serve as outcome variables: the effective number of features (*Df* = the number of nonzero parameters) and the prediction error AIC_{L} obtained with Equation 4.10.

Generating true configurations and sampling dissimilarities

The true configurations are defined by the three experimental variables *number of objects*, n/T ratio, and *size of the feature discriminability parameters*. The combination of the levels of these three experimental variables results into 8 experimental conditions. For each of these conditions, a feature matrix was created by taking a random sample from the total number of feature patterns for a given number of features (4, 8, 17, and 35). This method generates features for the number of bits equal to the number of features and results in a matrix of size $[\frac{1}{2}(2^T) - 1] \times T$ that represents the

Table 4.6: Matrices of features based on phonetic theory (left) and of features selected by the Positive Lasso (right)

	Features from phonetic theory							*Features selected by Positive Lasso*						
Consonants	F1	F2	F3	F4	F5	F6	F7	*F1*	*F2*	*F3*	*F4*	*F5*	*F6*	*F7*
p	0	0	0	0	0	1	0	1	1	1	0	0	1	0
t	0	0	0	0	1	0	0	1	1	1	0	0	1	0
k	0	0	0	0	0	0	1	1	1	1	0	0	1	0
f	0	0	1	0	0	1	0	1	1	1	0	1	1	0
θ	0	0	1	0	1	0	0	1	1	1	1	1	1	0
s	0	0	1	1	1	0	0	1	1	1	1	1	1	0
ʃ	0	0	1	1	0	0	1	1	1	1	1	0	0	0
b	1	0	0	0	0	1	0	0	1	1	1	1	1	1
d	1	0	0	0	1	0	0	0	0	1	1	1	1	1
g	1	0	0	0	0	0	1	0	0	1	1	1	1	1
v	1	0	1	0	0	1	0	0	1	1	1	1	1	1
ð	1	0	1	0	1	0	0	0	1	1	1	1	1	1
z	1	0	1	1	1	0	0	0	0	1	1	1	1	1
ʒ	1	0	1	1	0	0	1	0	0	0	1	1	1	1
m	1	1	0	0	0	1	0	0	0	0	0	0	0	0
n	1	1	0	0	1	0	0	0	0	0	0	0	0	0

Table 4.7: Feature matrices for 12 objects and rank numbers used to construct the true configurations for the simulation study

4 features condition				*8 features condition*							
F1	F2	F3	F4	F1	F2	F3	F4	F5	F6	F7	F8
0	1	0	1	1	1	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0
1	0	1	1	1	0	0	0	0	0	0	1
0	0	1	1	0	1	0	0	0	1	0	1
0	1	1	1	1	0	0	1	0	0	1	0
1	1	1	1	1	1	1	1	0	0	0	0
1	1	1	0	1	0	0	0	0	0	1	0
0	1	1	0	1	1	0	1	1	1	1	1
1	0	1	0	1	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0
1161	322	688	86	691	415	1921	444	1533	1568	1729	495

complete lattice of possible feature patterns for the given number of features. Selecting at random 12 or 24 (corresponding to the number of objects) feature patterns yields a connected network. It should be noted that this method is slightly different from the way the Gray codes are used to create the complete set of distinctive features, where Gray codes are generated for the number of bits equal to the number of *objects* instead of the number of *features*. Table 4.7 shows the resulting feature matrices for 12 objects and 4 or 8 features with on the bottom row the corresponding Gray code rank numbers. The distances of the true configurations were computed using the levels of the experimental variable *size of the feature discriminability parameters*. The network representations for the 12 objects with 4 and 8 features and the two different levels of the sizes of feature discriminability parameters are displayed in Figure 4.6.

The true distances $\mathbf{d}$ for these configurations were obtained with $\mathbf{d} = \mathbf{X}\boldsymbol{\eta}$, where $\boldsymbol{\eta}$ represents the experimental values of the feature discriminability parameters, and $\mathbf{X}$ results from the transformation from Equation 4.6 applied on the feature matrices displayed in Table 4.7. The feature matrices and configurations for the 24 objects condition were obtained in exactly the same way. For each of the experimental conditions 50 samples of dissimilarities were obtained with the two levels of error using the binomial distribution to ensure positive dissimilarity values that follow a normal distribution. The details of the method of sampling from the binomial distribution are described in Frank and Heiser (in press a).

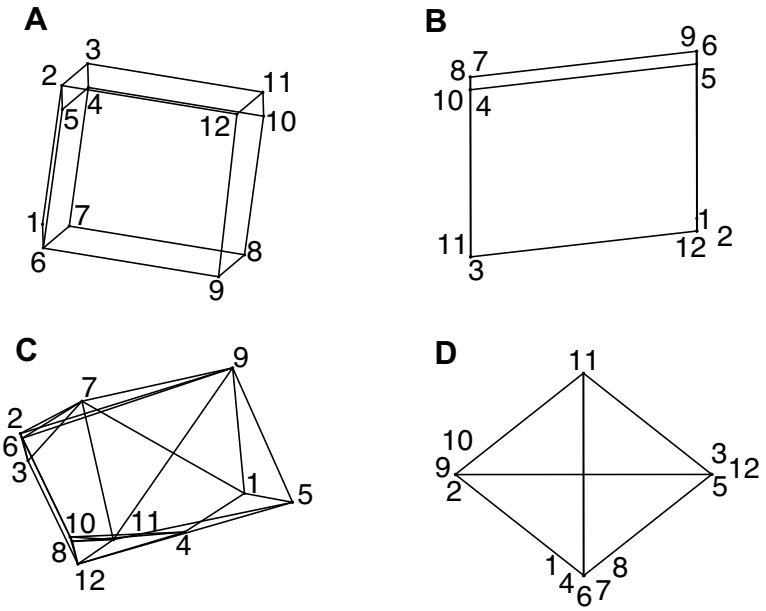

Figure 4.6: Feature network plots for the experimental conditions for 12 objects. A = 4 features, medium $\boldsymbol{\eta}$; B = 4 features, small + large $\boldsymbol{\eta}$; C = 8 features, medium $\boldsymbol{\eta}$; D = 8 features, small + large $\boldsymbol{\eta}$.

Results simulation study

Simulation results of the strategy using the complete set of Gray codes

This section answers the question whether the Positive Lasso selects the correct subset of features given that the true feature set is known, under different data analytic conditions. Table 4.8 shows the proportions of correctly recovered true feature rank numbers for the 50 simulation samples under the experimental conditions defined by combined levels of error, number of features and feature parameter size. In general, the true features are better recovered in the medium feature parameter values condition, compared to the combination of small and large feature parameter values.

When the feature parameter values all have medium size, the recovery is mainly affected by the ratio of the number of features compared to the number of observations (n/T). When the true number of features is small (n/T=16), there is perfect recovery of the features, regardless of the error level. In the condition of larger number of features compared to the number of observations (n/T=8) the true number of features is less well recovered. In the low error condition the proportions range from 0.86 to 1.00, with perfect recovery for the features with the highest true feature parameter values. In the high error condition, the features with the highest true fea-

ture parameter values are perfectly recovered, while the features with the lower true feature parameter values are less well recovered with proportions ranging from 0.00 to 0.62. The condition of combined small and large feature parameter values shows a different pattern: the features with large feature parameter values are perfectly recovered in all conditions formed by the combination of error level and the ratio of number of features compared to the number of observations. The features associated with small feature parameter values are recovered with small proportions in the condition of small number of features compared to the number of observations (n/T=16), and are almost never recovered in the condition of larger number of features compared to the number of observations (n/T=8).

Additional information on fit and effective number of features in the selected models is displayed in Figure 4.7, which shows the distributions of 50 simulation samples on 12 objects for all experimental conditions, using the complete set of distinctive features. The panels on the first row represent the effective number of features (= Df) selected by the Positive Lasso for each simulation sample and the true

Table 4.8: Proportion of correctly recovered features from the complete set of distinctive features under combined levels of error (L = low; H = high), the ratio of the number of object pairs and the number of features (= n/T ratio), and feature parameter (η) sizes, medium and small + large.

$n/T = 16$		*medium η values*							
		2.0	2.5	1.5	3.0				
	Error								
	L	1.00	1.00	1.00	1.00				
	H	1.00	1.00	1.00	1.00				
		small + large η values							
		6.0	0.2	0.5	0.3				
	Error								
	L	1.00	0.02	1.00	0.36				
	H	1.00	0.00	0.22	0.00				
$n/T = 8$		*medium η values*							
		2.0	2.5	1.5	3.0	2.0	2.5	1.5	3.0
	Error								
	L	0.98	1.00	0.94	1.00	0.94	1.00	0.86	1.00
	H	0.62	1.00	0.16	1.00	0.42	1.00	0.00	1.00
		small + large η values							
		6.0	0.2	0.5	0.3	6.0	0.2	0.5	0.3
	Error								
	L	1.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
	H	1.00	0.00	0.00	0.00	1.00	0.02	0.00	0.00

number of features is represented as a dashed line. The panels on the second row show the associated AIC_L values, which are measures of prediction error for the selected models. Each of the eight panels represents the two error levels, low (L) and high (H). The four panels on the left correspond to the condition with medium η values and the four panels on the right (first row and second row) correspond to the condition with small + large η values.

Since the pattern of the outcomes differs in these two levels of η values, we describe the results separately, beginning with the medium condition. The panel on the left of the first row shows the results for true number of features equal to 4. When

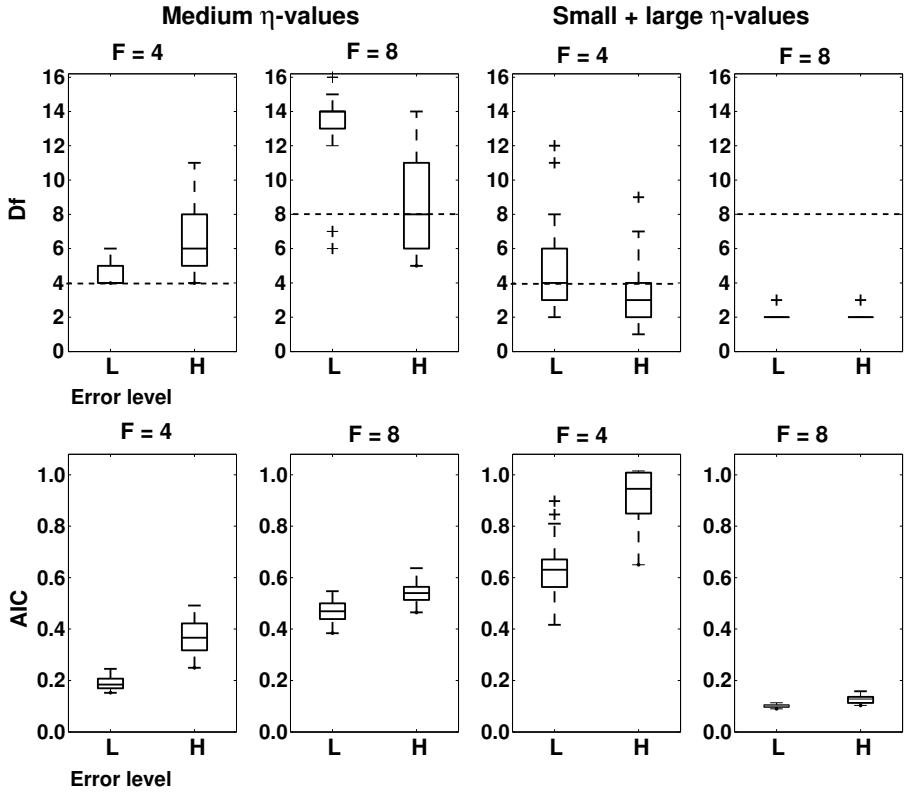

Figure 4.7: Boxplots showing the distributions of 50 simulation samples on 12 objects using the complete set of Gray codes. The experimental conditions are medium (left panels) and small + large (right panels) η values, two error conditions, low (L) and high (H), and two levels of true number of features (4 and 8) corresponding to two levels of n/T ratio equal to 16 and 8. The top panels show the effective number of features selected for each sample (= *Df*) with the true number of features represented as a dashed line. The lower panels show the associated AIC_L values.

error is low, the true number features are well recovered by the Positive Lasso, with a little overfitting for a small proportion of samples. The high error condition clearly shows some overfitting, because most of the models selected by the Positive Lasso contain 6 features. The corresponding AIC_L values show that prediction error is lower when error is low. The next panel shows the results for true number of features equal to 8, meaning that there are more features compared to the number of observations than in the previous condition of 4 features. In the low error condition, there is a considerable amount of overfitting because most of the selected models contain 14 features. The high error condition shows that most of the selected models contain the right number of 8 features. The associated AIC_L values show that the prediction error is higher in the high error condition, and that, despite the overfitting in the low error condition, the prediction error is still very acceptable.

The results for the small + large η values show a different pattern. In the model with 4 features, most of the samples in the low error condition recover models with 4 features, but some overfitting is clearly present. Most of the models selected in the high error condition, have fewer than 4 features. The prediction error is lower in the low error condition than in the high error condition. In the model with 8 features, the selected models all have 2 features, regardless of the error level. This substantial amount of underfitting does not have an effect on the prediction error, which is very low in both conditions. Combining these results with the findings in Table 4.8, we know that, in almost all the samples, the two features with the highest η values are selected.

Summarizing, the true number of features are best recovered in the 4 features model, for both medium and small + large η values. The 8 features model, shows some overfitting for the medium η values, and some underfitting for the small + large η values. In all conditions, the prediction error is satisfactory.

Simulation results on random sample of Gray codes + Filter

The model based on 12 objects allows for comparing the strategy of taking a random sample of features from all possible Gray codes combined with the use of a filter with the strategy of using the whole set of possible distinctive features obtained with all possible Gray codes. To assess the performance of the random sample strategy, the same simulation samples for the 12 objects used with the complete set of distinctive features, were analyzed again using the random sample strategy. The results of the random sample strategy are displayed in Figure 4.8. Since the same samples are used, Figure 4.8 can be compared directly with the results of Figure 4.7 that is based on the complete set of distinctive features.

The results in Figure 4.8 show that in the majority of the experimental conditions, the number of features selected by the Positive Lasso are equal to, or very close to the number of features in the true model. The best results in terms of recovered number of features are obtained when the true feature parameter values are a combination of small and large values. The best prediction error values occur when the true model has smaller number of features (the 4 features model) for both the medium and the small + large η values. When the true number of features is not recovered, there is, in general, a tendency towards overfitting. In particular, the condition with medium

η values and 4 features as the true model, shows a considerable amount of overfitting. However, the associated prediction error values are still the lowest of all the experimental conditions. In conclusion, compared to the method using the complete set of distinctive features, the random sample strategy has higher prediction error, but, in general succeeds in finding subsets of features of (about) the same number as the true models, besides a tendency to overfit in some conditions.

The simulation results for the model based on 24 objects provided additional information on the random sample strategy. Figure 4.9 clearly shows that in this case, the number of features selected by the Positive Lasso exceeded the true number of features considerably, even more when the true number of features is equal to 35,

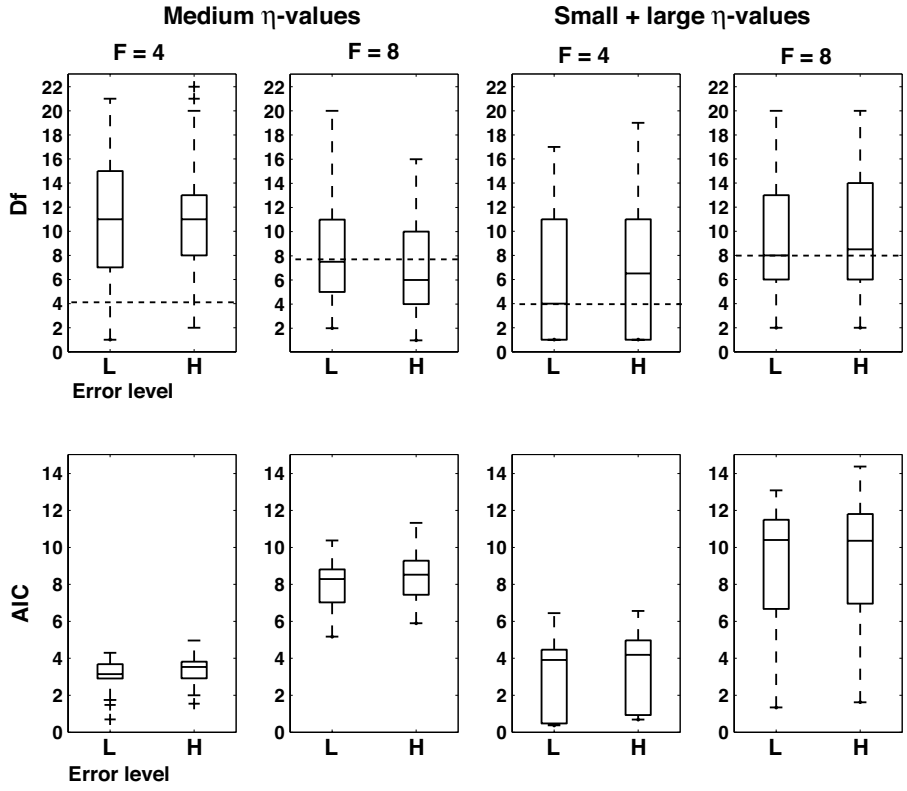

Figure 4.8: Boxplots showing the distributions of 50 simulation samples on 12 objects using a large random sample of the complete set of Gray codes combined with a filter. The experimental conditions are medium (left panels) and small + large (right panels) η values, two error conditions, low (L) and high (H), and two levels of true number of features (4 and 8) corresponding to two levels of n/T ratio equal to 16 and 8. The top panels show the effective number of features selected for each sample (= *Df*) with the true number of features represented as a dashed line. The lower panels show the associated AIC_{L} values.

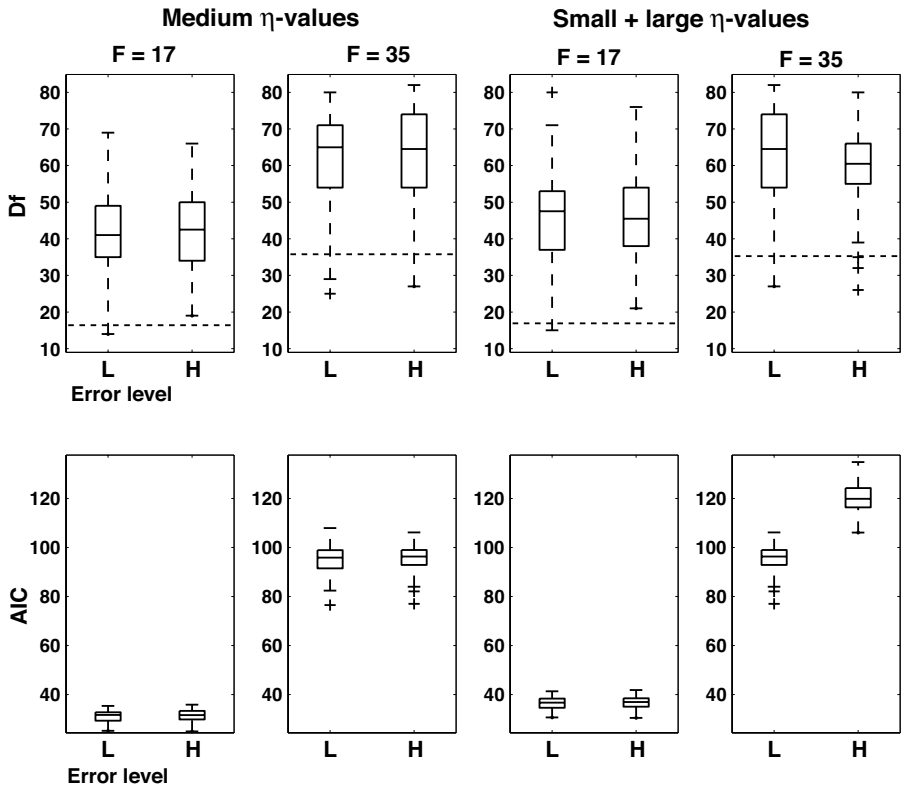

Figure 4.9: Boxplots showing the distributions of 50 simulation samples on 24 objects using a large random sample of the complete set of Gray codes. The experimental conditions are medium (left panels) and small + large (right panels) η values, two error conditions, low (L) and high (H), and two levels of true number of features (17 and 35) corresponding to two levels of n/T ratio equal to 16 and 8. The top panels show the effective number of features selected for each sample (= Df) with the true number of features represented as a dashed line. The lower panels show the associated AIC_{L} values.

the condition where the number of features is larger compared to the number of observations ($n/T = 8$ condition). The prediction error is much lower for the 17 features model (the condition $n/T = 8$). It is clear that many more features are needed to obtain models with acceptable levels of prediction error.

4.4 Discussion

This paper introduces a method to generate and select a subset of features for the Feature Network Models. It combines feature enumeration using Gray codes with a predictor selection algorithm, the Positive Lasso. The fact that FNM can be considered as a univariate multiple regression problem allows for the use of this type of predictor selection algorithm. The advantages of the multiple regression framework had not been fully explored in earlier feature models related to the FNM. In the following we discuss the two basic elements that constitute our method.

The first element is the enumeration of features with Gray codes. The enumeration of all possible subsets makes use of the adjacency property of the Gray codes, the property that successive numbers differ exactly in one bit. In our study, we did not explicitly use the adjacency property in the feature selection strategy. Instead of exploiting the adjacency property of the Gray codes, we used the codes to efficiently list all possible features where each feature is generated exactly once. The gain of using the Gray code results from its combination with the transformation from the objects $\times$ features matrix $\mathbf{E}$ to the object pairs $\times$ featurewise distances matrix $\mathbf{X}$. This transformation limits the search for predictors to the set of truly distinctive features. To our knowledge, the explicit search in the space of distinctive features has not been used as a predictor generation and selection strategy before. The adjacency property proved to be useful in the simulation study where we needed connected networks to represent the true network configuration. The list of Gray codes for the number of bits equal to the number of features (instead of equal to the number of objects) results in a lattice, or the complete network of all possible feature patterns. Selecting m feature patterns from this matrix of size $[\frac{1}{2}(2^T) - 1] \times T$ ensures a connected network representation for a given number of objects.

In this context, it should be noted that features can be generated using Gray codes in two ways. The first method amounts to generating Gray codes for the number of bits equal to the number of objects. This method is suitable in the situation where there is no a priori knowledge about the possible number of features suitable for the data at hand. Features generated in this way yield a feature matrix of size $m \times [\frac{1}{2}(2^m) - 1]$, which obviously leads to a feature selection problem because there are far more features than observations. The second method is more appropriate for the situation where there is some knowledge available on the number of features that would be reasonable for the data. In that case, features could be generated for the number of bits equal to the number of features instead of equal to the number of objects, resulting in a matrix of size $[\frac{1}{2}(2^T) - 1] \times T$. Again, the symmetric property of the Gray code allows for discarding the second half of the code. It is no longer a feature selection problem because the number of features is known in advance, but rather a problem of finding the right feature pattern for each object, a problem related to the travelling salesman problem. The set of all possible feature patterns given a fixed number of features can be viewed as a lattice, or complete network of all possible feature patterns. The best selection of feature patterns to describe a given number of objects should correspond to one of the possible shortest path routes on this lattice.

For the situation where one searches for sets of fixed numbers, Gray codes have been frequently used to list all p-element subsets of a q-element set in such a way that consecutive sets differ by exactly only one element (*cf.* Nijenhuis & Wilf, 1978). Applying this particular use of Gray codes to the FNM would lead to the complete enumeration of all possible subsets of features. It is well known that the number of subsets grows exponentially with the number of objects m and the number of features, limiting the strategy to a very small number of objects and features (a small explorative study showed that in the context of FNM complete enumeration of all subsets is limited to number of objects $\leqslant 5$ and number of features $\leqslant 9$). Attacking the problem of selecting a subset of features in this way would be an NP-hard problem (NP = nondeterministic polynomial problem, a category of problems that cannot be solved exactly in polynomial time, but for which the verification of the solution can be accomplished in polynomial time). Even if optimal solutions could be obtained, they would not necessarily yield a good compromise between model complexity and prediction accuracy. Since our method attempts to achieve this compromise it is difficult to compare it to conceptually different techniques like the aforementioned and the cluster differences scaling algorithm that has been used earlier in FNM.

Using our method, the generation of all possible distinctive features with Gray codes is feasible for $m \leqslant 22$. Simple binary codes could have been used instead, although, the generation of successive objects that differ in only one bit, might be faster (*cf.* Savage, 1997). If the number of objects exceeds 22, features are generated by taking a very large sample from the whole set of Gray codes followed by a filter technique that selects the features with the highest separate discriminability parameters. The results of the simulation study show that in this case, the number of features selected by our method exceeds the true number of features in some conditions. Therefore, feature generation for $m > 22$ must be further improved.

Given a very large set of features, obtained with the complete list of Gray codes or a large sample of this list, the second element of our method consists of selecting the best subset of features using the Positive Lasso, an adaptation of the Lasso algorithm to meet the positivity constraints of FNM. The results obtained with the Positive Lasso are in accordance with results obtained with the Lasso: the Lasso tends to perform best with a combination of small and large parameter values (Friedman & Popescu, 2006; Tibshirani, 1996). The results of our simulation study shows that the best results are obtained in the condition formed by the combination of small and large feature discriminability parameters. The Positive Lasso is also useful in the situation of features given by theory or provided by the experimental conditions, and helps to select the relevant features. We used the Positive Lasso as a tool, but it certainly merits to be studied in its own right since all its properties are not exactly known yet.

To select the amount of shrinkage, we used an AIC-like criterion adapted for the Lasso context. It is well known that AIC, in contrast to BIC, has a tendency towards overfitting (*cf.* Zou et al., 2006) and this property probably explains part of the overfitting observed in our simulation study. However, it is also known that AIC and BIC possess different asymptotic optimality (*cf.* Zou et al., 2006): if the true regression function is not in the candidate models, the model selected by AIC

asymptotically achieves the smallest average squared error among the candidates. BIC on the other hand is known for its consistency in selecting the true model: if the true model is among the candidate models, the probability of selecting the true model with BIC approaches 1 as the sample size approaches infinity. Given our setting, when the number of objects exceeds 22 and as a consequence we have to take a large random sample from the total set of features, we know in advance that the true model might not be present among the candidate models, which motivates the choice for the AIC criterion. Yang (2005) recently explored the possibilities of combining both strengths of BIC and AIC in the context of regression estimation and concluded that there are some theoretical and empirical results in support of adaptive model selection, but that it is still not clear whether it can really combine the strength of AIC (= prediction optimality or minimax-rate optimality) with the strength of BIC (consistency).

The combination of the two elements (the enumeration of features with Gray codes and the selection of the best subset of features) leads to a method that aims at selecting a subset of features that constitutes a good compromise between model fit and model complexity. The Gray codes allow for defining a finite solution space, which can be further reduced by restricting the search to the distinctive features only. In fact, features are generated from a model instead of being the result of collecting empirical data. Next, the Positive Lasso algorithm selects the best subset regardless of the number of features in the set. Instead of searching for the optimal solution in the distinctive features space, we prefer a suboptimal solution, selected with the Positive Lasso, that has better generalizability properties.

The method described in this paper can be applied directly to the common features model, with one restriction. Given that the total set of common features is larger than the total set of distinctive features, the limits of complete enumeration of the set of common features will be reached earlier than with 22 objects, the limit for the distinctive features model.