SFA classification with few training data: Improvements with parametric bootstrap

Wolfgang Konen
Cologne University of Applied Sciences
wolfgang.konen@fh-koeln.de

February 8, 2012

Slow Feature Analysis (SFA) is a versatile algorithm to find stable features or slow-varying signals in multidimensional data. It is capable of finding highly relevant features for classification tasks. This paper deals with the *marginal training data problem* which appears in SFA classification when the number of training records is too low. We derive a quantitative condition between training set size and SFA configuration parameters which allows to predict whether the marginal training data problem will occur. We analyze the reasons for the problem and propose several strategies to avoid it. Among these strategies, the *parametric bootstrap* approach, which augments the training data with virtual training patterns drawn from an estimated distribution, successfully solves the marginal training data problem. We report first evidence, that parametric bootstrap is also beneficial for non-marginal SFA and for other machine learning algorithms like Random Forests.

Contents

1 Introduction

Slow Feature Analysis (SFA) is a new learning algorithm emerging from neuroscience which is capable of learning new features or 'concepts' from multidimensional (time) signals in an unsupervised fashion. SFA has been originally developed in context of an abstract model of unsupervised learning of invariances in the visual system of vertebrates by Laurenz Wiskott [Wis98] and is described in detail in [WS02, Wis03]. Although SFA is inspired from neuroscience, it does not have the drawbacks of conventional ANNs (Artificial Neural Networks) such as long training times or strong dependencies on initial conditions. A MATLAB implementation of SFA is available as open-source toolkit `sfa-tk` [Ber03] and [Ber05] extended the SFA algorithm originally developed for time series to be applicable to classification tasks as well. [Kon09] extended `sfa-tk` with a new variant SVD-SFA with better numerical stability on arbitrary input signals and [KKH10] successfully applied `sfa-tk` to a gesture recognition test case where it has shown a similar performance to state-of-the-art machine learning algorithms like Random Forest (RF) [Bre01]. SFA is fast in training and it has the potential to find hidden features in multidimensional signals, as has been shown impressively by [Ber05] for handwritten-digit recognition.

However, it was observed in [KKH10] that the SFA classification algorithm shows suboptimal behavior if the number of training patterns is too small. We will refer to this effect as *marginal training data* problem. After a very short introduction to SFA in Sec. 2 it is the main purpose of this paper to describe the marginal training data problem and the circumstances under which it may appear in more detail (Sec. 3), to propose a new solution for this problem based on *parametric bootstrap* (Sec. 4) and to show some results on classification datasets (Sec. 5).

2 SFA

We give here a very short introduction to the main ingredients of SFA. For a more comprehensive discussion of SFA and SFA-based classification, the reader is referred to [WS02, Wis03, Ber05, Kon09].

In a nutshell, the SFA approach is defined as follows: For a signal $\vec{x}(t)$ where t indicates time and $\vec{x}$ is a vector of dimension n_{pp}, find the set of real-valued output functions $g_1(\vec{x}), g_2(\vec{x}), ..., g_J(\vec{x})$, such that each output signal

$$y_j(t) = g_j(\vec{x}(t)), \qquad j = 1, \ldots, J, \tag{1}$$

minimally changes in time:

$$\langle \Delta {y_j}^2 \rangle_t \text{ is minimal.} \tag{2}$$

Here, $\langle \cdot \rangle_t$ means average over time and Δy indicates the time difference operator in the case of time series and the intra-class pattern difference vector in the case of classification [Ber05].

To exclude trivial solutions we add some constraints:

$$\langle y_j \rangle_t = 0 \text{ (zero mean)} \qquad \text{and} \qquad \langle y_j^2 \rangle_t = 1 \text{ (unit variance)}. \tag{3}$$

For arbitrary functions this problem is difficult to solve, but SFA finds a solution by expanding the n_{pp}-dimensional input signal in a nonlinear function space of certain basis functions, e.g. monomials of degree 2:

$$\vec{z} = \left[x_1, \ldots, x_{n_{pp}}, x_1^2, x_1 x_2, \ldots, x_{n_{pp}}^2 \right]^T \tag{4}$$

This expanded signal $\vec{z}$ is transformed to a sphered expanded signal

$$\vec{v} = \boldsymbol{S}(\vec{z} - \langle \vec{z} \rangle_t) \tag{5}$$

where the sphering matrix $\boldsymbol{S}$ is choosen such that the constraint $\langle \vec{v}\, \vec{v}^{\,T} \rangle_t = \boldsymbol{I}$ (unit matrix) is fulfilled. Then SFA calculates the time difference of the sphered expanded signal and determines from its covariance matrix

$$\boldsymbol{C} = \mathrm{Cov}(\Delta\vec{v})$$

the normalized eigenvectors $\vec{w}_1, \ldots, \vec{w}_J$ with the smallest eigenvalues. Finally the sphered expanded signal is projected onto these eigenvectors to obtain the slow output signals

$$y_j(t) = \vec{w}_j \cdot \vec{v}, \qquad j = 1, \ldots, J.$$

It is easy to show that due to the sphering of Eq. (5) the output signals fulfill automatically the constraints of Eq. (3).

In SFA classification [Ber05] each pair of patterns from the same class level is used to form a mini time series of length 2. Thus, if we have P patterns for a certain class level, we can form $P(P-1)/2$ mini time series. The optimization goal is now to minimize the variation of (2) on average over *all* such mini time series. The goal of SFA to find a 'slow' signal transformation $y_j(t)$ then translates to finding a transformation with small intra-class variation. Such a transformation is suitable for subsequent classification by a simple classifier, e.g. a Gaussian classifier.

3 Marginal Training Data

3.1 Observation

Sometimes data is rare, e.g. for classification with few observations. Normally, the performance of a classifier will slowly degrade if we slowly diminish the number of training patterns. This is for example the case for the RF classifier in Fig. 1(a). SFA classification, on the other hand, shows on the same dataset a problematic behaviour: When only a small number of training data N is available for classification (more specifically: if $N < D_x$ where D_x is the dimension of the expanded function space) then the SFA classifier is not better than random guessing on new test data.

The situation was studied in more detail with the following experiment: We applied SFA and RF as reference method to the same set of data (5-class gesture data set from [KKH10]). The ratio of training and test set was varied starting with a high number of training data and low number of test data and then decreasing the training data while increasing the test data, respectively. For each number of training data, ten runs with different random seeds were carried out and in Fig. 1 we show the mean values and the standard deviations (error bars) for this experiment.

Both plots show that the classification rates of the two classifiers are promising with enough training data available (e. g. > 120 training patterns). However, with fewer training patterns the SFA detection rates become quite unsatisfying. The error rate increases immensely for < 90 training patterns, while the error on the training set constantly stays at zero level. We therefore see a case of severe overfitting. The RF predictor has with the same data no overfitting problem, the training set error is always a good predictor of the test set error and both errors are only moderately rising when the number of training patterns is diminished.

3.2 Explanation

Why has SFA classification such an overfitting problem with marginal training data? The answer to this question lies in the way how SFA computes the output signal. The algorithm computes the covariance matrix of the expanded input signal and determines its eigenvalues. We show in Sec. 3.2.2 that a certain covariance matrix becomes rank deficient if too few training examples are available. This leads to an underdetermined linear system to be solved by the algorithm. As a necessary precondition to avoid rank deficiency in SFA, we show below in Eq. (8) that the constraint

$$N \geq D_x + K \tag{6}$$

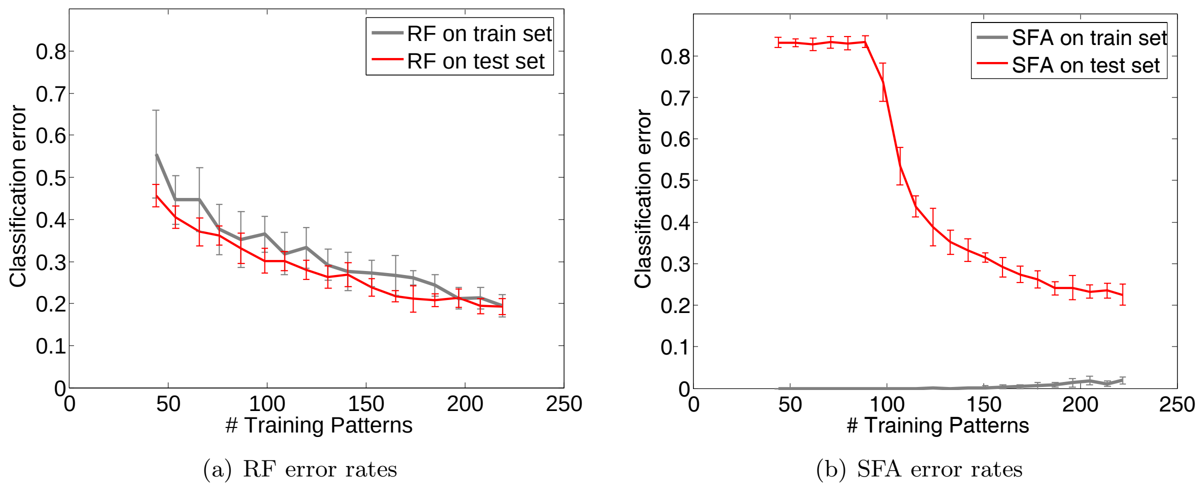


Figure 1: **Error rates on a gesture classification benchmark when varying the number of training patterns. (a) RF (Random Forest): The test set error on unseen data increases only slowly as we move from right to left into the regime of fewer training patterns. The training set error (OOB in the case of RF) is a good predictor for the test set error. (b) SFA with $n_{pp} = 12$: The test set error suddenly increases when the number of patterns is too small for a sufficient rank of the covariance matrix ($<$ 90 patterns), while the training set error stays constantly near zero.**

has to be met. Here N is the number of patterns available for training and K is the number of classes and D_x is the dimension of the nonlinear expanded function space used by SFA.

3.2.1 A Simple Example

As a simple example consider the case of $N = 4$ training patterns $\vec{v}$ in a $(D_x = 3)$-dimensional expanded function space. We assume $K = 2$ classes with 2 patterns in class 1 and 2 patterns in class 2. The two patterns of class 1 necessarily lie on a line l_1, their difference vector $\Delta\vec{v}_1$ forms an eigenvector direction of $\boldsymbol{C} := \text{Cov}(\Delta\vec{v})$. Similarly, the two patterns of class 2 have another difference vector $\Delta\vec{v}_2$ parallel to line l_2. The subspace spanned by both vectors $\Delta\vec{v}_1$ and $\Delta\vec{v}_2$ is a 2-dimensional plane and it contains the two eigenvectors of $\boldsymbol{C}$ with the smallest non-zero eigenvalues.

In the threedimensional expanded function space there is necessarily a direction (perpendicular to the eigenvector plane) for which all training patterns show no intra-class variation. (Analoguously, if the feature space is 4-dimensional, there are two directions with no intra-class variation at all.) Therefore $\boldsymbol{C}$ will have at least one eigenvalue exactly 0 or close to 0 within machine accuracy. The associated eigenvector direction is usually not a good direction for discriminating the underlying class distributions: It is just an artefact of the specific set of training patterns (see Fig. 2). If different training patterns (different two points in the 3D feature space) are selected, usually a different direction of the first eigenvector will emerge.

Let us depict in Fig. 3 a concrete case where l_1 coincides with the x-axis and l_2 is a line parallel to the y-axis but passing through the point $(x, y, z) = (0, 0, 2)$. Thus, for the patterns of class 1 and 2 we have $v_z = 0$ and $v_z = 2$, resp. The mean vector $\vec{\mu}$ of all training patterns has $\mu_z = 1$. The two lines l_1 and l_2 do not intersect, but they have a common direction perpendicular to them which is the direction of the z-axis. This direction will emerge as an eigenvector of $\boldsymbol{C} := \text{Cov}(\Delta\vec{v})$ with eigenvalue 0, because all the intra-class difference vectors have no component in this direction. Therefore the z-axis will be the SFA eigenvector with the slowest signal: the slowest signal y_1 is simply the projection of each vector $(\vec{v} - \vec{\mu})$ on the z-axis, which is in our case constantly -1 for class-1 patterns and $+1$ for class-2 patterns.

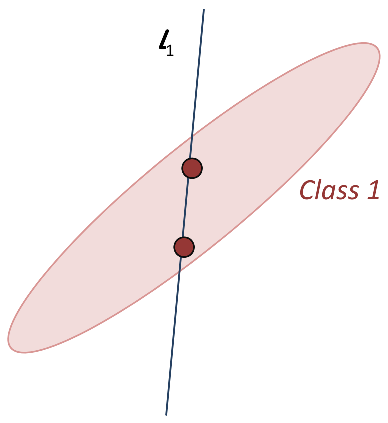


Figure 2: **If there are only two training patterns for class 1, the corresponding eigenvector direction will be along line l_1. This may be a bad characterization of the principal axis of the underlying distribution (ellipsoid).**

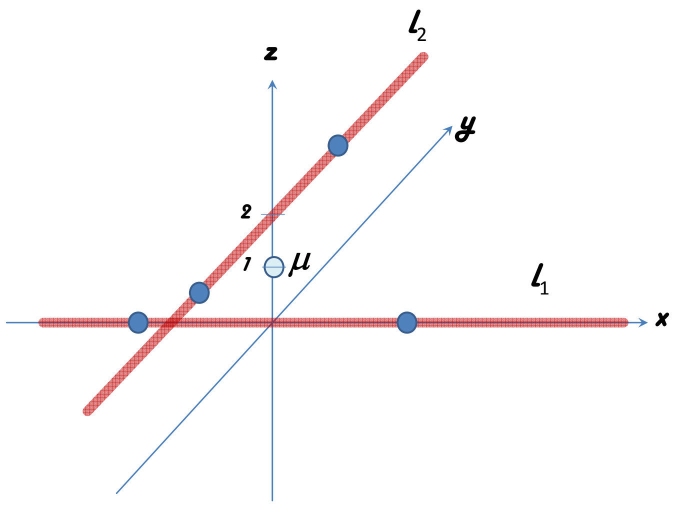


Figure 3: **A very simple 2-class problem.**

If now a new test pattern arrives, it will have almost always a z-projection different from -1 and $+1$. Even more, the z-projection will not be very significant for the class membership, since the z-direction was an artefact of the training data. Nevertheless, SFA will rely completely on this direction and therefore it makes with high probability a wrong classification.

This example should give the reader a basic and visual understanding for the reasons why SFA classification will fail on marginal training data. We will now transfer the arguments to arbitrary dimensions (Sec. 3.2.2) and to realistic datasets (Sec. 3.2.3).

3.2.2 The General Case

Given a classification problem with N training patterns $\vec{x}^{(n)}$, where each training pattern belongs to one of the K classes $m = 1, \dots, K$. Each pattern $\vec{x}^{(n)}$ is transformed to a point $\vec{v}^{(n)}$ in the nonlinear, D_x-dimensional expanded function space used by SFA. The SFA matrix $\boldsymbol{C} = \mathrm{Cov}(\Delta\vec{v})$ is formed from all intra-class difference vectors $\Delta\vec{v} = \vec{v}^{(i)} - \vec{v}^{(j)}$, where $\vec{v}^{(i)}$ and $\vec{v}^{(j)}$ are patterns belonging to the same class m. We show here that matrix $\boldsymbol{C}$ is rank deficient as soon as $N < D_x + K$.

Lemma:

$$\mathrm{rank}(\boldsymbol{C}) \leq \min(D_x, N-K) \tag{7}$$

Proof. Each of the N patterns belongs to one class m. If N_m is the number of patterns belonging to class $m = 1, \dots, K$, we have

$$N_1 + N_2 + \ldots + N_K = N.$$

The difference vectors $\Delta\vec{v} = \vec{v}_i^{(m)} - \vec{v}_j^{(m)}$ for class m will span at most an $(N_m - 1)$-dimensional subspace, since the N_m points $\vec{v}_i^{(m)}$ can not span more than $N_m - 1$ dimensions.[1] The matrix $\boldsymbol{C}$

[1]Note that this is true even though the number of difference vectors $\Delta\vec{v}$ entering into the calculation of $\boldsymbol{C}$ is larger than N_m, namely $N_m(N_m-1)/2$ in the case of monomials of degree 2.

is formed from these subspaces and thus can not have a rank larger than the direct sum of these subspaces:

$$\text{rank}(\boldsymbol{C}) \leq N_1 - 1 + N_2 - 1 + \ldots + N_K - 1 = N - K.$$

Since on the other hand $\boldsymbol{C}$ is a square matrix with D_x rows and columns, it can not have a rank larger than D_x. In combination this proves the Lemma above. □

Thus, if $N - K < D_x$ then $\boldsymbol{C}$ is rank deficient. There remain at least $D_x - (N - K)$ dimensions perpendicular to all difference vectors. These dimensions will have associated eigenvalues exactly 0 or close to 0 within machine accuracy, but they are not stable, since a different selection of N training patterns will lead to other directions. Therefore we will get 0% training set error (by construction), but with high probability a large test set error. In other words,

$$N \geq D_x + K \tag{8}$$

is a necessary condition to avoid rank deficiency and overfitting.

To see more clearly why we get 0% training set error by construction, we have to look at the way how SFA processes training and test data for classification (cf. Fig. 3): Each pattern is transformed into the sphered expanded function space yielding a vector $\vec{v}$. The mean vector $\vec{\mu}$ of all expanded training vectors is subtracted. Then several projections on the slowest SFA directions take place, which are in the case of rank deficiency parallel to eigenvectors $\vec{w}_j$ of $\boldsymbol{C}$ with eigenvalue 0. All training records of one class are in a hyperplane with a constant offset to $\vec{\mu}$, therefore the projection of $\vec{v} \cdot \vec{w}_j$ will be the same number for all training patterns of the same class and a usually different number for each different class. Any classifier trained on such data will have 0% training set error by construction.

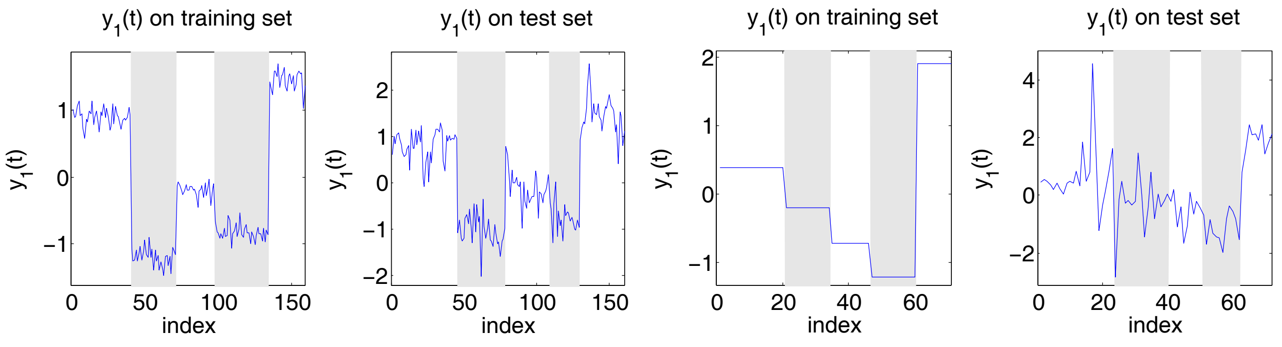
3.2.3 Real Data Example

Fig. 4(a) shows the results from a gesture recognition experiment [KKH10] where SFA with $D_x = 90$ was used on a sufficient number of 159 training records which is enough for $D_x + K = 95$ as required by Eq. (8). The slowest SFA signal y_1 shows some intra-class variation and a sharp inter-class separation. A Gaussian classifier trained on the four slowest SFA signals $y_1, \ldots, y_4$ can learn quite robustly to separate the 5 classes; the test set errror is with 8% quite low. In contrast, Fig. 4(b) shows the results from SFA when there are too few training records. Here the number of $N = 71$ training records is smaller than $D_x + K = 95$. Consequently we get a rank-deficient covariance matrix and the SFA ouput signal shows absolutely *no intra-class variation* on the training data (severe overfitting). The SFA model will select any of the $95 - 71$ dimensions with eigenvalue zero, and almost surely this dimension will be meaningless for the test data. As a result we get a high error rate of 77% on independent test data, which is not better than random guessing.

3.2.4 Effect on the Gaussian Classifier

What is the effect of a rank-deficient matrix $\boldsymbol{C}$ on the subsequent classifier, which is in our SFA case usually a Gaussian classifier? The effect of zero or very small (10^{-7} and below) eigenvalues on the Gaussian classifier is very drastical: As explained in the preceeding section, the projection of the training data on the corresponding eigenvector dimensions will yield a zero variance of the SFA output in these dimensions. Consequently, the Gaussian classifier will model a Gaussian distribution which is completely flat in these dimensions.

Now if new test data arrive for the classifier, they will usually not have zero projections in these dimensions. The classifier will consider these data so far outside from any trained distribution that it will respond with a completely arbitrary class label. As a consequence, the classifier will not be better than random guessing, the error rate becomes usually as high as $1 - 1/K$.



(a) Sufficient training data (159 records) (b) Too few training data (71 records)

Figure 4: **Output of the slowest SFA signal y_1 for $D_x = 90$ on the 5-class gesture classification problem. The x-axis shows the training (test) record index. For better visualization the records are ordered: first class-1 records (leftmost white area), then class-2 records (grey area) and so on up to class 5 (rightmost white area). (a) Sufficient training data: The SFA signals on training and test set (160 records) show a roughly similar behaviour, only with a somewhat larger intra-class variation in the test set. (b) Too few training data (at least $D_x + K = 95$ records are required): The output on the training data shows severe overfitting. The SFA signal on the test set (72 records) is completely different, and the test set error (77%) is not better than random guessing.**

Two options can be considered to improve the situation, but they both do not really solve the problem:

- Exclude all SFA output dimensions (eigenvectors of of $\boldsymbol{C}$) where the ratio λ/λ_{max} of the corresponding eigenvalue λ is below $\epsilon_c = 10^{-7}$. This does not solve anything, because this throws away the important dimensions, which are *somewhere* in the subspace spanned by the eigenvectors with zero eigenvalue. (Remember, the eigenvalues would not be zero, if sufficient training data from the underlying distribution were available.)
- Regularize the covariance matrix $\boldsymbol{\Sigma}$ of the Gaussian classifier. If $\boldsymbol{\Sigma}$ has diagonal elements smaller than $\epsilon_d = 0.01$, replace them by 0.01 before calculating the inverse $\boldsymbol{\Sigma}^{-1}$. This improves the situation somewhat, e. g. in a 5-class-problem the error rate may drop from 80% down to 30%-40%, but it is still not a good classifier.

Thus, the only way to deal successfully with marginal training data in SFA classification is to avoid the rank deficiency of matrix $\boldsymbol{C}$ right from the beginning, and this is what we will discuss in the next section.

4 Solving the Marginal Training Data Problem

While SFA works well in classification experiments with sufficient training data, where it achieves results comparable to the well-known Random Forest classifier, it shows severe limitations on cases with marginal training data. Are there possibilities to overcome these limitations? We have seen in Sec. 3.2, Eq. (8), that the necessary condition to avoid overfitting is

$$N \geq D_x + K$$

There are two options to cure a situation where Eq. (8) is initially violated:

1. **Fewer features:** Decrease D_x to a value below $N - K$.

2. **Parametric bootstrap:** Increase N by augmenting the training set with additional samples drawn from an approximating distribution.

4.1 Fewer Features

For monomials of degree 2 the relation

$$D_x = n_{pp} + \frac{n_{pp}(n_{pp}+1)}{2} \tag{9}$$

holds. Therefore, one option is to decrease n_{pp} until the constraint of Eq. (8) is fulfilled. An example is shown in Fig. 5, where the task was to classify the gestures of one person: For each cross validation run we have 66 or 67 training data (90% of 74 gesture records in 10-fold cross validation). This leads to the necessary condition $D_x \leq 66 - 5 = 59$ or $n_{pp} \leq 9$ acc. to Eq. (9), which is confirmed by the steep incline of the red curve between $n_{pp} = 9$ and 10 in Fig. 5. Best results are obtained with $n_{pp} \in \{5, 6\}$. – This option works, but it has the drawback that the amount of information transferable to the classification algorithm is quite severely limited to 5 or 6 input dimensions.

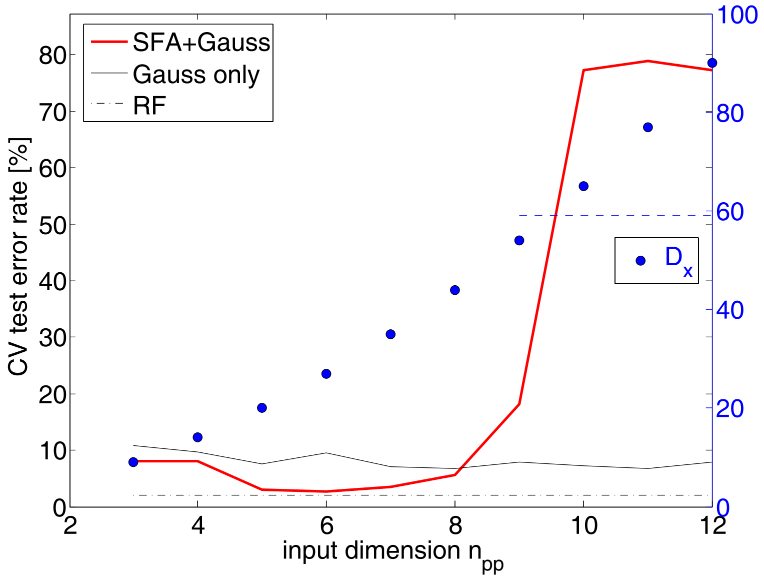


Figure 5: **SFA with marginal data: Too high n_{pp} yield unsatisfactory error rates. Experimental settings: Cross validation with 10 folds, each fold ha 66 or 67 training data. Shown is the average out of 5 runs with different seeds for the fold generation. The dashed blue line shows the critical value for D_x: with $D_x < 59$ we get satisfactory error rates.**

4.2 Parametric Bootstrap

Another option is to keep n_{pp} at its desired value, but to enrich the training data by *parametric bootstrap* [HTF01, pp. 264]: This method increases the number of training instances by adding N_{copies} new records which are 'noisy copies' of original training records $\vec{x}$.

Given a classification problem with a training set $S = \{\vec{x}^{(n)} | \ n = 1, \ldots, N\}$ of size N, where each training pattern belongs to one of the classes $m = 1, \ldots, K$. The function $c(n) = m$ denotes for each pattern $\vec{x}^{(n)}$ its class membership. Let

$$f_m = N_m / N \tag{10}$$

denote the relative frequency of class m in the training set where N_m is the absolute frequency of class m. Each pattern vector $\vec{x}^{(n)}$ has F components (features) $x_i^{(n)}$, $i = 1, \ldots, F$, and we define the following quantities:

$$\mu_i^{(m)} = E\left[x_i^{(n)} \middle| c(n) = m \right] \tag{11}$$

$$\sigma_i^{(m)} = \sqrt{E\left[\left(x_i^{(n)} - \mu_i^{(m)} \right)^2 \middle| c(n) = m \right]} \tag{12}$$

which are the mean and the standarad deviation, resp., of the i^{th} feature when averaging over all patterns n with $c(n) = m$, i. e. when using all patterns of class m.

Then the parametric bootstrap algorithm can be formulated as follows:

Parametric_bootstrap_algo(S, N_{copies}):

```
1: Initialization: Compute $\mu_i^{(m)}$, $\sigma_i^{(m)}$ and $f_m$          ▷ Eqs. (10), (11) and (12).
2: for ($c = 1, \ldots, N_{copies}$) do                                    ▷ Loop over bootstrap samples.
3:     Select class $m$ from $\{1, \ldots, K\}$ with probability $f_m$
4:     for ($i = 1, \ldots, F$) do
5:         Draw a random variable $Z \prec N(0, \sigma_i^{(m)})$ and form $X_i^{(c)} = \mu_i^{(m)} + \sigma_{nc} Z$
6:     end for
7: end for
8: Return the augmented training set $S^+ = S \cup \{\vec{X}^{(c)} | c = 1, \ldots, N_{copies}\}$
```

The augmented training set S^+ will have the same basic statitistic properties (class frequency, class centroids $\vec{\mu}^{(m)}$) as the basic set S. This is true in the statistical sense, i. e. for sufficiently large N_{copies} or when averaging over multiple S^+ productions. σ_{nc} is a free strength parameter, which is usually choosen close to 1. For $\sigma_{nc} = 1$ the set S^+ has on average the same class standard deviations $\sigma_i^{(m)}$ as the basic set S.

The number N_{copies} of the bootstrap samples can either be given by the user or it is automatically set by the following simple heuristic:

Automatic_bootstrap_heuristic(S):

```
1: if $N <= 1.2 * (D_x + K)$ then
2:     $N_{copies} = \alpha_{nc} * (D_x + K) - N$
3:     $S$ = Parametric_bootstrap_algo($S$, $N_{copies}$)
4: end if
5: Return $S$
```

The parameter 1.2 in the above heuristic can be considered as rule of thumb. $\alpha_{nc} = 1.5$ is a usual parameter value for the desired training set size ratio, but higher values might be also beneficial for the overall accuracy. SFA is not very sensitive to the precise value as long as the number of training records is well above the border $D_x + K$.

5 Results

Fig. 6 shows the resulting CV error rates as a function of augmented training set size $|S^+|$. We expect to need at least $|S^+| = D_x + K = 90 + 5$ records and we find from Fig. 6 that the steepest decline of the red curve is near this value. However, to get a low error rate, $|S^+|$ should be higher, between 200 and 350. Note that the parametric bootstrap affects only the training data, and no changes to the test data are made. Therefore the CV test error rate remains realistic. – The option *parametric bootstrap* allows to put more of the original training information into the SFA model since we can work with n_{pp} =12 or higher and are not restricted to n_{pp} =6. We name this enhanced algorithm SFA+PB.

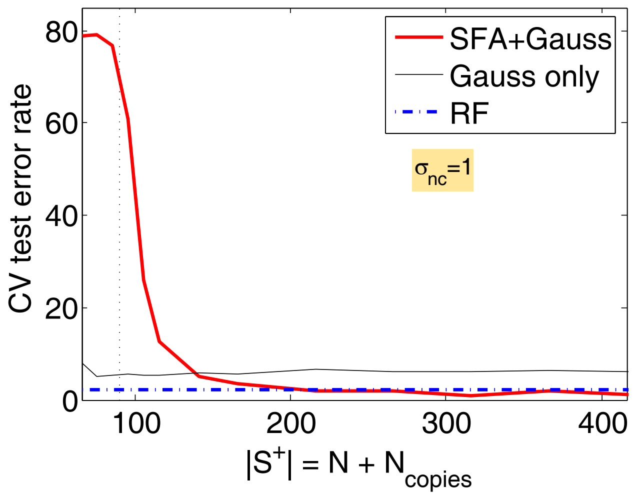


Figure 6: **Bootstrapping SFA in the case of marginal data: SFA with too few training data ($N = 66$, $N_{copies} = 0$) gets very high test set errors (approx. 80%). By applying a parametric bootstrap, which augments the training data to the new size $|S^+| = N + N_{copies}$, we decrease the SFA test set error dramatically to values around 2% (thick line "SFA+Gauss"). This is by a factor of 3 smaller than the test set error from a Gaussian classifier (thin line "Gauss only") and comparable to the state-of-the-art classification algorithm Random Forest (RF, 2.1%). Dotted vertical line: the critical value $|S^+| = D_x + K = 90 + 5$. Experimental settings: Same as Fig. 5, additionally $n_{pp} = 12$ (hence $D_x = 90$) and $\sigma_{nc} = 1.0$.**

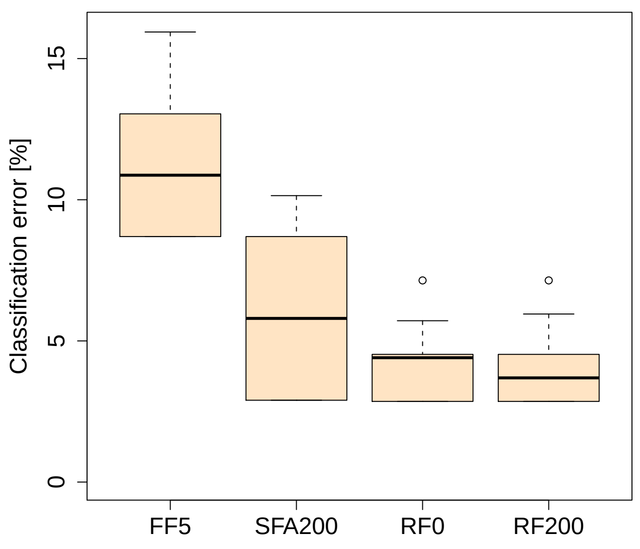
Parameter sensitivity We ran several tests with other values of parameter σ_{nc}. The algorithm is not very sensitive to this parameter, since we get nearly identical results if we halve or double the value. But the right order of maginitude is important: With too small values, e.g. $\sigma_{nc} = 0.1$, the convergence as a function of N_{copies} is very slow, while with too large values, e.g. $\sigma_{nc} = 3.2$, the error rates rise after a small dip quickly to unsatisfactory high error rates of 25% and above.

Strategies to handle marginal data Fig. 7 compares different strategies to cope with marginal data. SFA with parametric bootstrap (SFA200) is always better than using fewer features (FF5). Random Forest (RF0, RF200) is comparable to SFA200. RF200 is the test whether RF will benefit from parametric bootstrap samples although it does not have a critical marginal training data set size. The results from our experiments show that RF200 is slightly better than RF0, but further experiments on other data would be needed to verify whether this slight performance increase is significant.

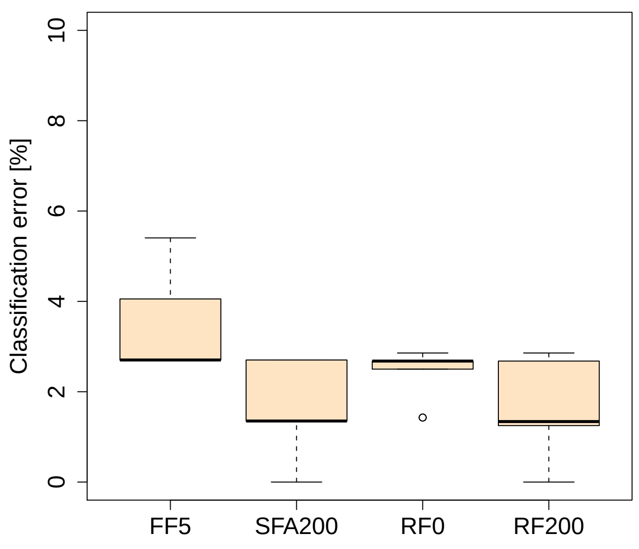
6 Conclusion and Outlook

In this paper we have studied the effect of small training set size (marginal data) on the SFA classification algorithm. We identified the reason why SFA has to fail with too few training data, gave in Eq. (8) a quantitative condition for *too few training data* and offered several strategies to overcome this problem (regularization of the Gaussian classifier, fewer feature dimensions, parametric bootstrap).

Among these strategies, parametric bootstrap (PB) is most versatile (applicable to differently sized problems) and gives the best results in terms of classification errors on independent test sets. It has been shown that parametric bootstrap is also beneficial for other classification algorithms like Random Forest.



(a) Person 121 (63 training patterns)



(b) Person 122 (67 training patterns)

Figure 7: **Comparision of different strategies to handle marginal training data: FF5: fewer features, i.e. set $n_{pp} = 5$ and do not use parametric bootstrap. SFA200: use $n_{pp} = 12$ and augment each training set with 200 parametric bootstrap patterns. FF5 and SFA200 are the SFA-related strategies. For comparision we process the same cross-validation data with random forest (RF) as well. RF0: plain RF with 500 trees and `mtry`=3; RF200: augment each training set with 200 parametric bootstrap patterns. Each box plot shows the results from 10 cross-validation experiments, each with 10 random folds and each fold using different random parametric bootstrap samples.**

What is the reason that parametric bootstrap has a beneficial effect on most classification tasks? Without PB and with too few training data, certain dimensions in the expanded space will be overemphasized. These dimensions might obscure the important dimensions. With PB no dimension will have a zero eigenvalue in $\boldsymbol{C}$ (at least if each input dimension in the expanded space has a non-zero standard deviation). The removal of the zero eigenvalue directions makes it possible for SFA to 'find' in the data those directions with slow variations originating from the data itself (and not from a meaningless consequence of the training record selection process).

But even in the case of non-marginal data it can be beneficial to augment the training set with PB patterns. In [KKH10] it was shown that the difficult task of gesture classification on unseen persons has originally a test set error of 17.7%. With PB on the training set, the test set error improved by 10% to 15.3%. It is our opinion that with PB the distribution of the training data is explored more thoroughly. This lets SFA concentrate on those dimensions which show small variation in the training data and as a consequence it allows SFA to generalize better on new gestures of persons not seen during training.

We used here a very simple – however broadly applicable – parametric model for the bootstrap, namely a model based on a Gaussian noise distribution. For the gesture classification task we plan to investigate as future work a more specific parametric model where the creation of virtual patterns is based on gesture-specific geometric operators, e.g. random rotations of real class patterns, or timeline operators like shift of start and stop point. We expect that with such virtual patterns the generalization capabilities, especially towards unseen persons, can be enhanced.

In summary, the neuro-inspired algorithm SFA has shown to be fast and precise on classification tasks and it needs only few parameters. Due to its simple projection approach, the application of the trained model is 3–6 times faster than the already fast RF method [KKH10]. With SFA+PB, our new parametric-bootstrap extension (see Appendix for details), the algorithm can deal also with few training data, which was not possible for plain SFA.

Acknowledgement

This work has been supported by the Bundesministerium für Bildung und Forschung (BMBF) under the grant SOMA (AIF FKZ 17N1009, "Ingenieurnachwuchs") and by the Cologne University of Applied Sciences under the research focus grant COSA.

Appendix: Implementation Details `sfa-tk` (MATLAB)

The parametric bootstrap algorithm is part of the MATLAB toolkit `sfa-tk`, version V2.8 and higher, which is available for download from `http://gociop.de/downloads/`. `sfa-tk` V2.8 is an extension of Pietro Berkes' `sfa-tk` V1.0 [Ber03].

The parametric bootstrap algorithm is implemented in file `sfa/add_noisy_copies.m` in three slightly different versions:

`pars.ncmethod`	description
1	Each training pattern in turn is centroid; for each feature only one standard deviation (over all classes)
2	One centroid per class (see Eq. (11)); same standard deviation as in case 1
3	One centroid per class (see Eq. (11)); one standard deviation per class and feature (see Eq. (12));

Case 3 is exactly the algorithm described in Sec. 4.2 and it is the recommended case, because it models the approximating distribution most accurately. (Case 1 and 2 are predecessors which approximate the underlying distribution with less accuracy. However, in our experiments undertaken so far, all three cases gave very similar results.)

`sfa/sfa_classify.m` checks with `sfa/sfaPBootstrap.m` whether the number of training patterns is too small. If so, it automatically generates a sufficient amount of parametric bootstrap patterns and works with the union of all patterns as augmented trainig set. Use `pars.ncalpha` $= \alpha_{nc}$ to set the training set size ratio (see Sec. 4.2) to values different from the default value 1.5.

The automatic augmentation with parametric bootstrap patterns can be disabled if the user sets `pars.doPB=0`.

The Gaussian classifier can be regularized by a parameter `opts.CL.epsD` $= \epsilon_d$ in order to suppress too small diagonal elements (see Sec. 3.2.4). If omitted, `gaussClassifier.m` will set the default `opts.CL.epsD=0.01`.

`opts.epsC` $= \epsilon_c$ is an optional parameter to exclude eigenvector dimensions with eigenvalue ratio $\lambda/\lambda_{max} \leq \epsilon_c$. As described in Sec. 3.2.4 it is recommended for classification to leave this option inactive by setting `opts.epsC` $= 0$ (the default).

A simple example script for SFA classification may look like this (see also `demo/class_demo2.m`):

```
pars = parsDefault();
opts = optsDefault();
[x,realclass,xtst,realc_tst,opts] = dataLoad(opts,pars);
res = sfa_classify(x,realclass,xtst,realc_tst,opts,pars);
[confmat,classerr] = mk_confmat(realc_tst,res.predT,opts.nclass,opts.classes);
fprintf('Confusion matrix & class errors on test set\n    --- true class ---\n')
disp(confmat);
disp(num2str(classerr,'%7.3f'));
```

References

[Ber03] P. Berkes. `sfa-tk`: Slow Feature Analysis Toolkit for MATLAB (v.1.0.1). http://people.brandeis.edu/~berkes/software/sfa-tk, 2003. 2, 12

[Ber05] P. Berkes. Pattern recognition with slow feature analysis. Cognitive Sciences EPrint Archive (CogPrint) 4104, http://cogprints.org/4104/, 2005. 2, 3

[Bre01] L. Breiman. Random forests. In *Machine Learning*, pages 5–32, 2001. 2

[HTF01] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001. 8

[KKH10] P. Koch, W. Konen, and K. Hein. Gesture recognition on few training data using slow feature analysis and parametric bootstrap. In *Proceedings of IEEE World Congress on Computational Intelligence 2010, Barcelona*, page to appear, July 2010. 2, 3, 6, 12

[Kon09] W. Konen. On the numeric stability of the SFA implementation `sfa-tk`. arXiv.org e-Print archive, http://arxiv.org/abs/0912.1064, December 2009. 2

[Wis98] L. Wiskott. Learning invariance manifolds. In *Proc. of the 5th Joint Symp. on Neural Computation*, volume 8, pages 196–203, San Diego, CA, 1998. Univ. of California. 2

[Wis03] L. Wiskott. Estimating driving forces of nonstationary time series with slow feature analysis. arXiv.org e-Print archive, http://arxiv.org/abs/cond-mat/0312317/, December 2003. 2

[WS02] L. Wiskott and T. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002. 2