JACOBS
UNIVERSITY

Vytenis Šakėnas

Distortion Invariant Feature Extraction with Echo State Networks

Technical Report No. 24
October 2010

School of Engineering and Science

Distortion Invariant Feature Extraction with Echo State Networks

Vytenis Šakėnas

School of Engineering and Science
Jacobs University Bremen gGmbH
Campus Ring
28759 Bremen
Germany

E-Mail: v.sakenas@jacobs-university.de

Abstract

In complex pattern recognition tasks data usually exhibits many local distortions which significantly disturb the recognition process. A method for extracting temporal features from a signal that are invariant to these distortions is presented in this report. The idea is to use Echo State Network to generate a rich high-dimensional representation of data. Temporal features are then extracted by finding projections of the high-dimensional representation that are minimally influenced by the selected distortions while still carrying most of the information about the underlying signal required for the performed task. The algorithm performance is analyzed on synthetic signals as well as on high-dimensional handwriting data for shift and scale distortions. It is shown that the algorithm is capable to extract a low dimensional feature set from a reservoir which is invariant to the selected distortions and relevant to the performed task.

1 Introduction

Pattern recognition and signal processing of complex real world data, such as speech or handwriting, is a difficult task for computational intelligence. One of the reasons for it is the inherent abundance of distorting variations in data. In speech processing such variations come from changes in pitch, accents and intonations of different speakers. Similarly, variations in handwriting are caused by a different slant, character size, pen width, writer style etc. Modeling all distortions for such a high-dimensional data is a daunting task facing the curse of dimensionality.

In the current state-of-art speech and handwriting recognizers data distortions are handled from two different aspects. First, the data is heavily preprocessed by multiple algorithms. These algorithms are usually carefully handcrafted for a very specific distortion. For example, in handwriting recognition text is aligned, centered, deslanted and normalized using several specialized preprocessing routines. The second aspect for dealing with variations in data is designing the recognizer that inherently achieves invariance to some of the distortions. Good examples of such models are convolutional neural networks [LBBH98] and multidirectional multidimensional recurrent neural networks [GS09].

In this report I present a method for learning temporal features which are invariant to certain distortions in the input data. The goal of the proposed algorithm is to process complex data (such as handwriting or speech) and output features that contain only relevant information to the task (e.g. classification) with filtered-out distortions. The following list summarizes the main principles of the proposed algorithm:

- Extracted features should contain information about the underlying signal which is needed for the performed task. For example, in a classification task information relevant for determining the label of a pattern should be maximally retained.

- Variation that is caused by selected distortions should be filtered out in the produced features. That is the same pattern with different distortions should produce similar temporal features.

- Data processing is performed with an Echo State Network (ESN) [Jae01]. It is a recurrent neural network with a random and fixed internal connectivity matrix.

- The features are learned by a linear readout from the reservoir. This is the core paradigm of reservoir computing. The reservoir acts only as an excitable medium.

- The training data should contain pairs of examples that differ only by the distortion that the algorithm should learn to remove in the extracted features.

- Learning of the readout weights is performed by solving an optimization problem which minimizes the mean square error between the outputs when original and transformed instances of the same pattern are processed by the reservoir.

- To enforce the features to represent useful information, additional constraints are added. The learned outputs should be orthogonal, have a zero mean and unit variance.

- Learning of such features is efficient, i.e. of the same complexity as the traditional output learning of the ESN using linear regression.

In the next section I describe the proposed algorithm for extracting transformation invariant features from the reservoir. The performance of it will be analyzed on simple synthetic data and real handwriting. A brief analysis of a local receptive field reservoir layer which utilizes the described learning method will be performed in the following section. The report will conclude with a discussion and future directions of the research.

2 Model for extracting distortion invariant features

Before going in details of the proposed learning algorithm, lets briefly recap the basic ESN equations and fix the notation. In this report I will be dealing with a standard discrete time ESNs with leaky integrator units which are updated by:

$$\mathbf{x}(t+1) = (1-\lambda)\mathbf{x}(t) + \lambda \tanh(\mathbf{W}_{in}\,\mathbf{u}(t+1) + \mathbf{W}\,\mathbf{x}(t)). \tag{1}$$

Here $\mathbf{x}(t)$ is the network unit activation at the discrete time step t, $\mathbf{W}_{in}$ and $\mathbf{W}$ are the randomly generated input and internal reservoir weight matrices, $\mathbf{u}(t)$ is the network input and λ is the leaking rate of a neuron. The hyperbolic tangent is used as a nonlinear activation function for the internal units of the reservoir. The output of the ESN is read out by:

$$\mathbf{y}(t) = \mathbf{W}_{out}\,\mathbf{x}(t), \tag{2}$$

where $\mathbf{W}_{out}$ is the output weight matrix. In our case, learning of $\mathbf{W}_{out}$ is the target of the learning algorithm.

As already pointed out, the input and internal weights for ESNs are generated randomly. Thus the input is embedded by the reservoir in a (usually) high dimensional space in a very complex manner. The rich expansion of the input allows learning very powerful models for different kind of temporal data. In our case we want to exploit the rich expansion of the input signal to separate the distortions from information which is meaningful to the task. The hope is that the distortions are most strongly represented in just a few dimensions of the reservoir. In this case we want to learn a projection from the reservoir of $K < N$ dimensions (where N is the size of the reservoir) where we omit $N - K$ directions in which the distortions are pronounced most.

First, let us define what we call invariant features. Suppose $\phi(\mathbf{u})$ is the transformation that the learned features should be invariant to (e.g. it shifts the given input vector $\mathbf{u}$ by a random amount). Let the $\mathbf{u}(t)$ be the temporal pattern and $\tilde{\mathbf{u}}(t) = \phi(\mathbf{u}(t))$ its transformed version using the above defined transformation. Then we will call a feature f invariant to transformation $\phi(\mathbf{u})$ if $f(\mathbf{u}(t)) \approx f(\tilde{\mathbf{u}}(t))$.

This directly gives the criteria that should be optimized in learning invariant features. That is, given an input signal $\mathbf{u}(t)$ and its distorted version $\tilde{\mathbf{u}}(t)$, we need to find functions $f_j(\mathbf{u})$ such that the output signals $y_j := f_j(\mathbf{u}(t))$ and $\tilde{y}_j := f_j(\tilde{\mathbf{u}}(t))$ minimize the following quantity:

$$\text{minimize } \Delta := \left\langle (y_j - \tilde{y}_j)^2 \right\rangle_t \tag{3}$$

where $\langle \cdot \rangle_t$ denotes averaging over time. Additionally we add the following constraints:

$$\langle y_j \rangle_t = 0 \tag{4}$$

$$\left\langle y_j^2 \right\rangle_t = 1 \tag{5}$$

$$\forall i < j : \langle y_i y_j \rangle_t = 0 \tag{6}$$

These constraints force the extracted features to have a zero mean (equation 4), unit variance (equation 5) and be orthogonal to each other (equation 6). The first two constrains are added to avoid trivial solutions such as a constant output which is independent from the input signal $\mathbf{u}(t)$. The last constraint forces different features to represent different information. It also imposes order, meaning that i-th feature will be more invariant to the distortion than j-th.

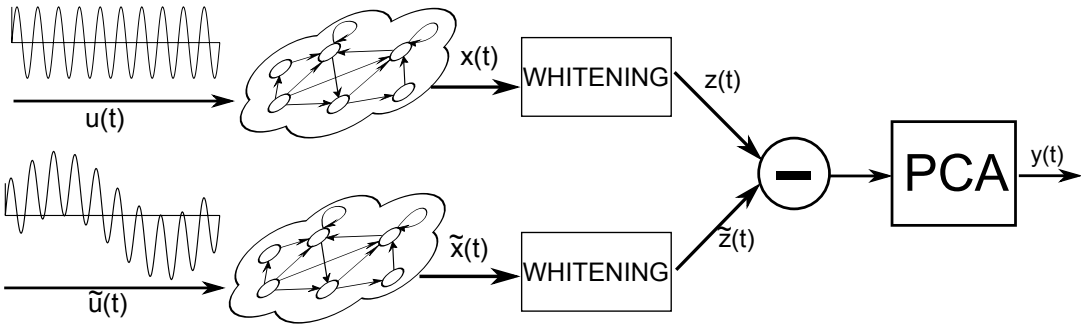

Figure 1: The basic scheme for extracting transformation invariant features. The ESN is run separately with the input signal $\mathbf{u}(t)$ and its transformation $\tilde{\mathbf{u}}(t)$. This gives two state vector sequences $\mathbf{x}(t)$ and $\tilde{\mathbf{x}}(t)$ which are passed through a whitening filter to produce whitened state vector sequences $\mathbf{z}(t)$ and $\tilde{\mathbf{z}}(t)$. The difference $\mathbf{z}(t) - \tilde{\mathbf{z}}(t)$ is then computed and principle components of the differences are extracted using PCA. The resulting feature vector $\mathbf{y}(t)$ is the K principle components with smallest eigenvalues.

A reader familiar with Slow Feature Analysis (SFA) [WS02] will notice that this is very similar to the way slow features are defined. The difference is that here we minimize the difference between outputs for two signals differing by a transformation $\phi(\mathbf{u})$ while SFA minimizes the derivative of the output signal with respect to time. Hence we also use a very similar learning procedure to solve

the above optimization problem. The algorithm for extracting the transformation invariant features can be split to the following steps:

1. **Input expansion.** First we drive the Echo State Network with data and collect the reservoir states. In this step we want to obtain a rich and high-dimensional temporal representation of data. The core idea behind the algorithm is the hope that in this representation there will be only a few directions in which distortions are represented the most. These directions then can be ignored in further processing of data. To find such directions we have to collect the states during two separate runs. First, the ESN should be driven with the original data. Second, it should be driven with the same data having a particular distortion applied. We denote the collected state sequences as $\mathbf{x}(t)$ and $\tilde{\mathbf{x}}(t)$. Note that quality of the input expansion can be controlled by several parameters of the reservoir, such as: input scaling, reservoir size, spectral radius and leaking rate. For details on how these parameters affect the reservoir dynamics see [Jae01] and [LJ09].

2. **Signal whitening.** In the second step we normalize these signals $\mathbf{x}(t)$ and $\tilde{\mathbf{x}}(t)$ by a whitening (or sphering) transformation. During whitening, the mean of the signal is subtracted and a linear transformation is then applied such that the resulting signal has a unit variance in all directions. This is done by using principal component analysis. This step is important to make the learned features obey constraints 4-6. In our case the whitening is performed on the concatenation $(\mathbf{x}(t), \tilde{\mathbf{x}}(t))$. The result of this step is the whitening matrix $\mathbf{Z}$.

3. **Calculate whitened state difference.** After computing the whitening matrix in the previous step we acquire the whitened reservoir state sequences $\mathbf{z}(t) = \mathbf{Z} \cdot \mathbf{x}(t)$ and $\tilde{\mathbf{z}}(t) = \mathbf{Z} \cdot \tilde{\mathbf{x}}(t)$. In this step we calculate the difference $\mathbf{d}(t) = \mathbf{z}(t) - \tilde{\mathbf{z}}(t)$.

4. **Find directions of least variance.** In the final learning step we have to find directions of least variance of the whitened state difference $\mathbf{d}(t)$. This will solve the initial optimization problem defined by the 3, because in these directions on average the square of the signal difference is minimized. Again these directions can be found using principal component analysis. Principal components of $\mathbf{d}(t)$ will be the directions we are searching for. Constraint 6 will be satisfied because principal components are orthogonal while constraints 4 and 5 are satisfied because of the whitening. The result of this step is the matrix $\mathbf{P}$ containing the K principal components of $\mathbf{d}(t)$ with the smallest eigenvalues.

As we can see the training algorithm produces two matrices: the whitening matrix $\mathbf{Z}$ and the projection matrix $\mathbf{P}$. The output of the ESN in the exploitation

phase is then obtained by using the output matrix $\mathbf{W}_{out} = \mathbf{P} \cdot \mathbf{Z}$. The scheme described above is illustrated in Figure 1.

From a computational perspective the learning boils down to the following steps:

1. Run ESN with $\mathbf{u}(t)$ and $\tilde{\mathbf{u}}(t)$ according to Equation 1. Obtain the reservoir state sequences $\mathbf{x}(t)$ and $\tilde{\mathbf{x}(t)}$ (ignoring the initial washout phase). Let $\bar{\mathbf{x}}(t) := (\mathbf{x}(t), \tilde{\mathbf{x}(t)})$, i.e. temporal concatenation of the reservoir state sequences.

2. Subtract the temporal mean of reservoir states: $\bar{\mathbf{x}}_0(t) = \bar{\mathbf{x}}(0) - \langle \mathbf{x}(t) \rangle_t$. Let $\bar{\mathbf{X}}_0 = [\bar{\mathbf{x}}_0(1) \cdots \bar{\mathbf{x}}_0(M)]$.

3. Compute the correlation matrix $\mathbf{C} = \bar{\mathbf{X}}_0 \bar{\mathbf{X}}_0^T$ and perform SVD on it, getting $(\mathbf{U}, \mathbf{S}, \mathbf{V}) = svd(\mathbf{C})$.

4. Compute the whitening matrix by $\mathbf{Z} = \mathbf{S}^{-\frac{1}{2}} \cdot \mathbf{U}$ and obtain the whitened reservoir state sequences $\mathbf{z}(t) = \mathbf{Z}\mathbf{x}(t)$ and $\tilde{\mathbf{z}}(t) = \mathbf{Z}\tilde{\mathbf{x}}(t)$.

5. Compute the whitened state difference signal $\mathbf{d}(t) = \mathbf{z}(t) - \tilde{\mathbf{z}}(t)$.

6. Subtract the temporal mean from the whitened state difference $\mathbf{d}_0(t) = \mathbf{d}(t) - \langle \mathbf{d}(t) \rangle_t$. Let $\mathbf{D}_0 = [\mathbf{d}_0(1) \cdots \mathbf{d}_0(M)]$.

7. Compute the correlation matrix $\mathbf{C}' = \mathbf{D}_0 \mathbf{D}_0^T$ and perform SVD on it, getting $(\mathbf{U}', \mathbf{S}', \mathbf{V}') = svd(\mathbf{C}')$.

8. The feature projection matrix is $\mathbf{P} = \mathbf{U}'(N - K + 1 : N, :)$ (i.e. last K rows of the $\mathbf{U}'$ matrix)

In the above description step (1) corresponds to input expansion, (2)-(4) to the whitening phase, (5) to the state difference calculation and (6)-(8) to the invariant feature extraction. After the training in the exploitation phase the distortion invariant features are obtained simply by using the learned whitening and feature projection matrices ($\mathbf{Z}$ and $\mathbf{P}$) on the states of the reservoir. The described learning algorithm is fast (i.e. of the same computational complexity as the traditional supervised ESN learning by performing linear regression). This feature extraction is also stable because in the spirit of reservoir computing the feature extraction does not modify the reservoir or influence the dynamics in any way.

2.1 Spatial feature extraction from a single reservoir

The first demo example is a proof-of-principle and uses rather simple synthetic input data. For this purpose I use a simple sine wave as an undistorted signal. To generate distorted versions of this simple core signal three different transformations will be analyzed: shift variation, scaling variation and both shift and scaling

variations together. The distorted data in three described settings was generated according the following expressions:

$$u_{shift}(t) = \sin(at) + f\sin(bt + c) + \nu$$
$$u_{scale}(t) = \sin(at) \cdot g \cdot (2 + \sin(dt + e)) + \nu$$
$$u_{shift+scale}(t) = \sin(at) \cdot \sin(dt + e) + m\sin(bt + c) + \nu$$

Here $\sin(at)$ is the core undistorted signal that we are interested in recovering. The generated distortions are on a significantly slower time scale than the signal. In the experiments the constants were selected from the following ranges: $a \in [0.09; 0.11]$, $b, d \in [0.09; 0.011]$, $f, g \in [2; 3]$, $e, c \in [0; 2\pi]$. Also low amplitude Gaussian additive noise ν is added to the distorted signal. Examples of clean and distorted signals are shown in Figure 2. In this demo I use a standard ESN (Equation 1) with a reservoir of size 50. The reservoir weight matrix has a spectral radius of 0.9 and the input weights are sampled form a uniform distribution in [-0.5,0.5]. The leaking constant λ is set to 0.7. Precise values are not very significant because similar results can be achieved in a very wide parameter range.

To learn features invariant to a particular transformation I first generate two signals according to expressions given above. It is important that the driving signal in both of them remains the same (i.e. $\sin(at)$ part) while the distortion is different (e.g. parameters b,c and f for shift distorted signal). These signals are used to run the ESN and learn the invariant features using the algorithm described in the previous section. To test the outcome of the training I generate a third signal having the same type of distortion, however, here I use a different driving signal (i.e. change the parameter a). This is done to ensure that the learned features generalize well for slightly different signals than the ones that were seen during training.

The features generated by the trained model for a specific distortion are shown in Figure 2. The plots show 5 features that are most invariant to the selected distortion. We see that, as expected, these features show almost no sign of severe distortions that are exhibited by the input signal. Moreover, from visual inspection it can be seen that these features also contain a lot of information about the undistorted driving signal.

It is important to quantify how well the extracted features represent the underlying undistorted signal (i.e. $\sin(at)$). For this purpose I use them to reconstruct the clean signal by using standard linear regression. The regression weights are computed on the training data. The reconstruction quality is measured by computing the normalized root mean square error (NRMSE) of the reconstructed signal and the actual clean signal during the testing run. It is interesting to see how the reconstruction quality changes when more and more distortion invariant features are used. For this purpose the reconstruction was evaluated with all possible numbers of features. The results of this study are shown in Figure 3. As a comparison, the clean signal was also reconstructed from distorted input

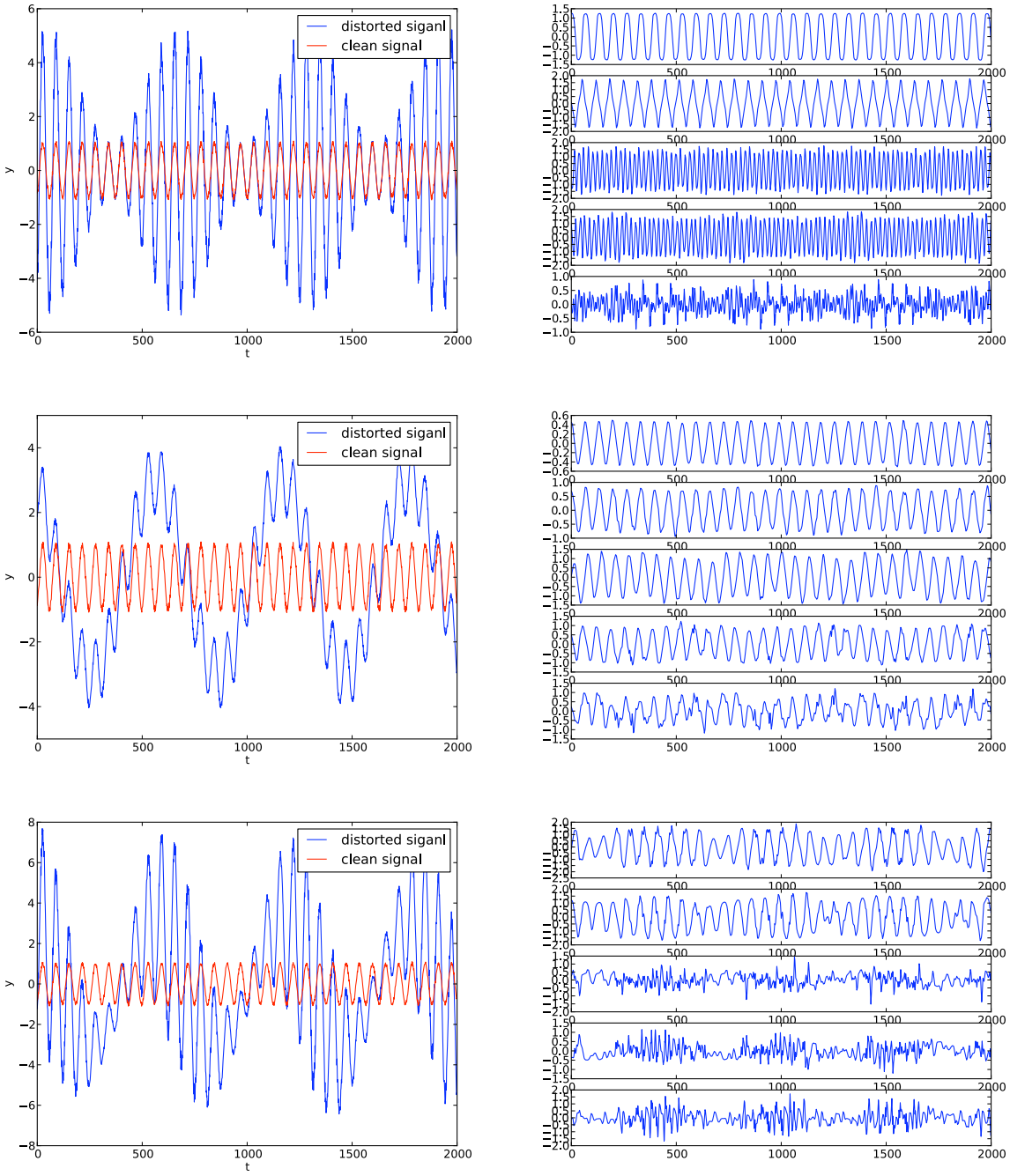

Figure 2: On the left: the driver signal (in red) and the distorted signal (in blue) for different distortions. On the right: five features with least information about the distortion (ordered from most to less invariant). Vertically: signal with shift distortion, signal with scaling distortion, signal with both shift and scaling distortions applied.

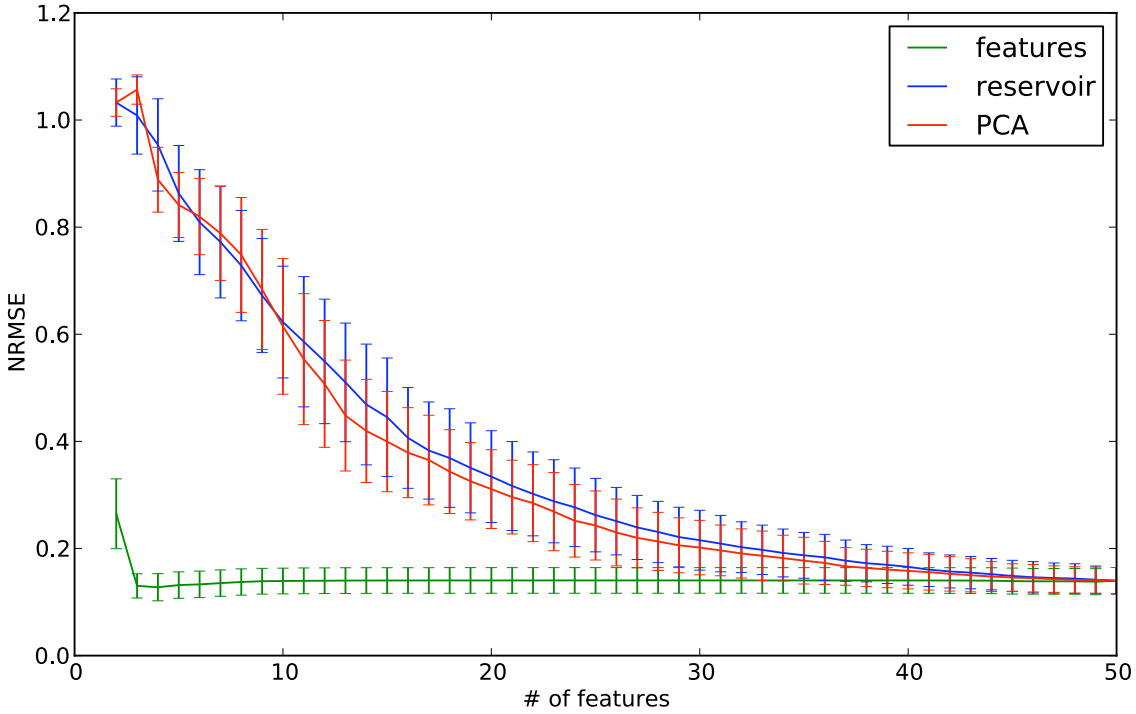

Figure 3: The NRMSE of reconstruction of the driver signal from the extracted features. On x axis - the number of features used, on y axis - the reconstruction error. The data is corrupted with shift distortion.

using different numbers of raw reservoir states and different numbers of principal components of raw reservoir states. We see that using the invariant features the reconstruction error drops very quickly and is optimal when using only 4 most invariant features. Conversely, the reconstruction error when using an increasing number of the raw reservoir states or principal components decreases slowly and reaches the optimal value only when the whole 50 components (i.e. use the whole reservoir as readout) are used. We may conclude that the learned features indeed efficiently represent the desired information and can be used to learn a low dimensional distortion-invariant representation of data.

After getting positive results on the simple dataset, I tested the algorithm on difficult real world data. For this purpose I used the USPS handwritten digit dataset [1]. This dataset contains 16x16 pixel size grayscale images of handwritten digits from zero to nine. To generate a temporal data from it a random sequence of single digit images were concatenated. The goal of this example is to learn vertical shift invariant features. Therefore, a random vertical shift distortion of ± 2 pixels was added to every digit. This resulted in a 20-dimensional signal of handwritten

[1] Can be downloaded from `http://cs.nyu.edu/~roweis/data/usps_all.mat`

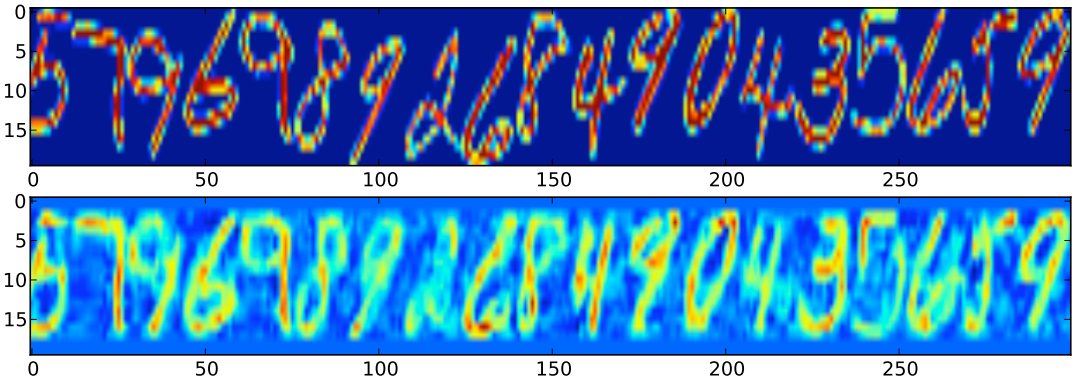

Figure 4: The shift distorted handwritten digit data (top) and the reconstruction from 200 most invariant features with the shift variation undone (bottom).

digits (see Figure 4). Similarly to the previous demo, three separate signals were generated: two for training (with the same underlying digits, but different shift transformations) and one for testing (with different digits).

In this example we are dealing with a high-dimensional and difficult data, thus a large reservoir of 500 units is used. The internal weight matrix is scaled to a spectral radius of 0.9. The input weights are generated by sampling from a uniform distribution in [-0.1,0.1] and the leaking constant λ is set to 0.7.

Because of the complex nature of the input, it is virtually impossible to get an intuition on what the extracted raw features represent. For this purpose I generate a feature correlation with a past input window diagram. Namely, I compute $FC_i = \langle y_i(t) \cdot [\mathbf{u}(t-k+1)...\mathbf{u}(t)] \rangle_t$, where index i denotes the i-th feature in the feature vector $\mathbf{y}$. This gives an idea to what parts of input the feature responds most. The feature correlation diagrams for handwriting data are shown in Figure 5. We see that the most shift invariant features indeed extract low spatial frequency components from data which are least affected by vertical shift. On the other end, the least invariant features show little order in their response pattern and are greatly influenced in even small variations in the input (e.g. single pixel shift).

Again it is interesting to quantify how well the extracted features represent the data. We can do it by performing the reconstruction of the input data *without* the shift distortion from the features. Similarly to the first demo, the reconstruction weights are learned using linear regression on the training data and tested on the reconstruction of the test data. In this case one additional trick was employed. Due to the nature of data it is essentially impossible to remove the shift distortion instantaneously. For this reason the reconstruction is delayed by 4 timesteps, i.e. at time t the undistorted input at time $t-4$ is produced.

The sample reconstruction of input with shift removed is shown in Figure 4. 200 most shift invariant features were used in this case. It is evident that the reconstruction is far from perfect, which is confirmed by the reconstruction NRMSE

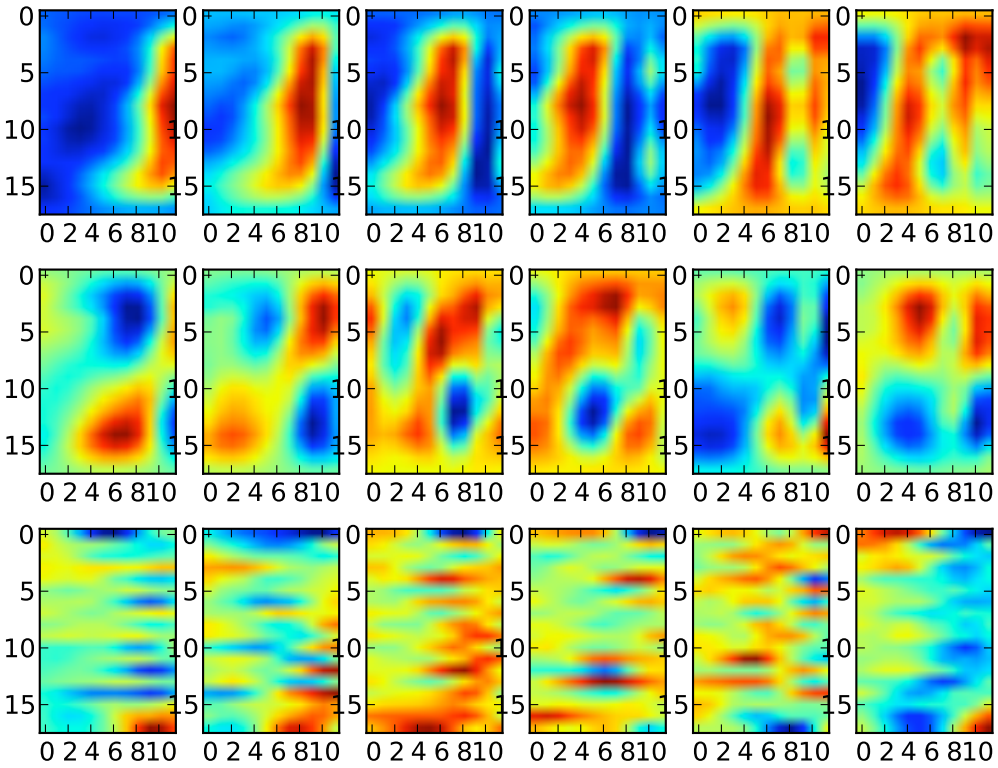

Figure 5: Feature correlation with the last 12 timesteps of input. Top two rows: the 12 most invariant features. They show a clear organization responding primarily to the vertical and diagonal lines in the input. Bottom row: the 6 least invariant features. They are highly dependent on pixelwise vertical shift of the input.

of 0.618. In Figure 6 the reconstruction performance with different number of features is compared. Again, as in the first demo example the reconstruction error drops much faster when using the shift invariant features compared to the raw reservoir states or principal components of the reservoir. Here, though, the optimal performance is achieved with a 150-180 feature set (i.e. 30-40% of the reservoir compared to 10% in the previous demo). Another important observation is that the optimal reconstruction NRMSE using the raw states (or principal components) of the reservoir is worse (0.650). Although, improvement of the reconstruction performance when using the shift invariant features is not drastic, it is still statistically significant.

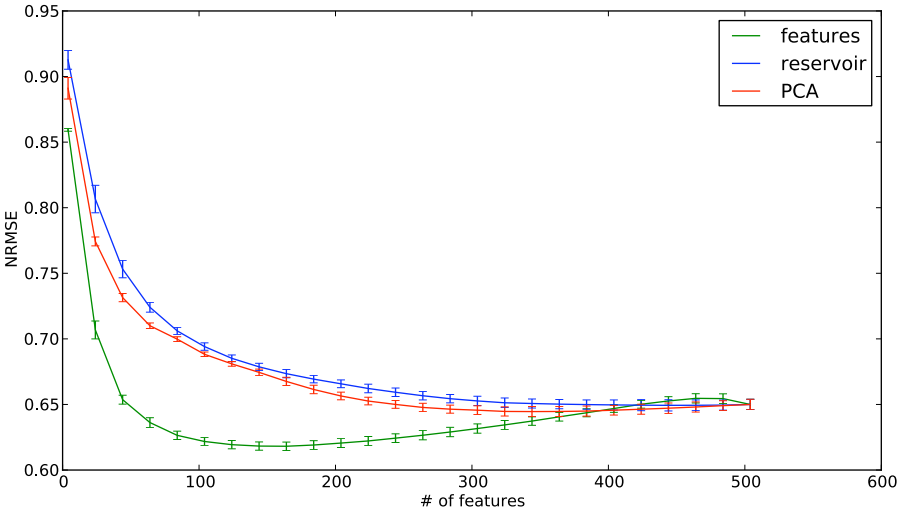

Figure 6: The NRMSE of the reconstruction of the handwriting signal without the shift from extracted features. On x axis - the number of features used, on y axis - the reconstruction error.

2.2 Model for processing multidimensional data

There are two shortcomings of the described algorithm that I want to briefly address in this section. One is that it requires two instances of data that differ only by the desired distortion. Second, that it does not work as well for complex high-dimensional data as for simple ones. The idea to try to circumvent this is to have a modular reservoir, where each module (i.e. reservoir) has local receptive field designed in a such way that adjacent modules get adjacent data dimensions of the input (see Figure 7). In addition, input and internal weights of the modules are shared between all of them. In this case if we look at any two adjacent modules we see that they differ only by the received input which is shifted by a few dimensions. Thus we can directly apply the described feature learning algorithm using states of the adjacent modules and learn locally shift invariant features.

For training of such modular reservoir one run with the original data suffices. The shift invariant features are learned due to the internal structure of the model. In addition, the fact that every module is processing only a few dimensions of the data might allow to learn a more powerful model in the long run. Finally, there is one additional benefit of using such architecture. Namely, we can learn not only the vertical shift invariant features in an easy way. Let the output feature vector of the i-th module at time t be $\mathbf{y}^i(t)$. Then minimizing $\left\langle (y^i_j(t) - y^{i+1}_j(t))^2 \right\rangle_t$ give the vertical shift invariant features as in previous examples, but we can also choose to minimize $\left\langle (y^i_j(t-1) - y^i_j(t))^2 \right\rangle_t$ which would give the horizontal shift invariant

features[2] or $\left\langle (y_j^i(t-1) - y_j^{i+1}(t))^2 \right\rangle_t$ giving features invariant to a diagonal shift. Indeed this learning scheme perfectly fits for learning outputs of the modules in this architecture because (a) the learned features are invariant to local shifts (b) a low dimensional output is enough to capture the underlying data (c) the learning is unsupervised, i.e. a teacher signal is not needed. In fact, the core idea of the algorithm was found when thinking on how to train such modular architecture shown in Figure 7.

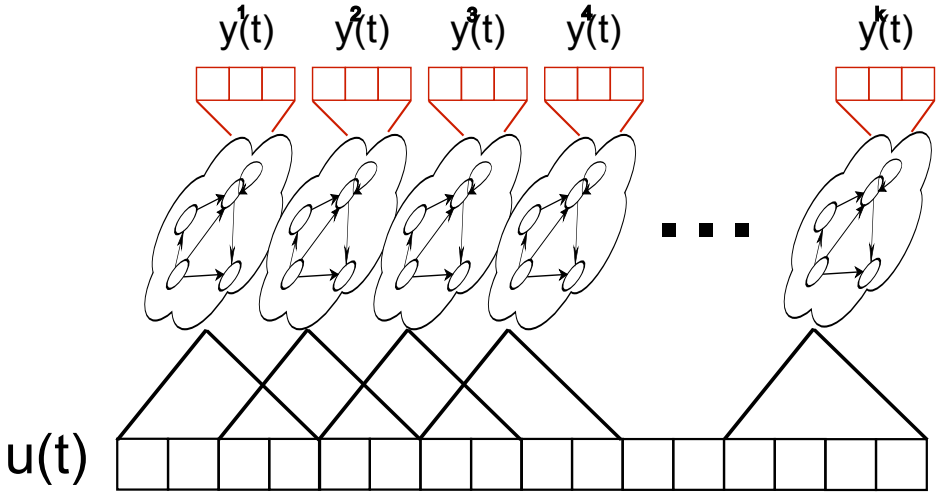

Figure 7: A modular reservoir for processing high-dimensional input. Each reservoir sees only a small window of the input. Input and internal weights are shared by all reservoirs. The output feature vectors $\mathbf{y}^i(t)$ (in red) should be learned.

A single layer of the modular reservoir was tested on the same handwritten digit shift removing task as described in the previous section. Reservoirs of 50 neurons were used for the modules. Each module received input from 6 input dimensions with adjacent modules having the input shifted by a single dimension resulting in 15 modules in total. Despite a substantially different setup the performance of this model was very similar to the single reservoir. The best reconstruction NRMSE (0.634) was achieved using 30 features while the reconstruction error dynamics and feature correlation diagrams were almost identical to ones achieved by the single reservoir (figures 5 and 6). This indicates that although the learning algorithm integrates nicely into the modular reservoir the modularity alone does not influence performance much (which is expected). The full potential of such a modular architecture, however, should come from processing the extracted features in a multiple hierarchical layers. The details on how multiple such layers should be integrated to make use of the extracted features in a most efficient way remain to be worked out.

[2]This is actually the standard Slow Feature Analysis.

3 Discussion

An algorithm for learning distortion invariant features from a reservoir is outlined in this report. The core idea of the approach is to find linear projections from the reservoir such that the projected reservoir states vary minimally when the same signal with a distortion is processed. The learning process of finding such projections boils down to solving an optimization problem by using principal component analysis first for whitening the reservoir states and then for finding the optimal projections. Due to the constraints in the optimization problem, the resulting features are uncorrelated and ordered, i.e. from the most invariant to the least invariant to the selected distortion. The algorithm was tested on a simple and complex data. In both cases the extracted features showed invariance to the selected distortion and carried a high amount of information about the core signal. Namely, 10-35% of the most invariant features were enough to optimally reconstruct the underlying undistorted signal.

There are several target applications for the proposed algorithm. First and foremost it allows one to generate a rich temporal feature set of a complex signal while filtering for distortions. This approach could be used in the early stages of speech or handwriting processing architectures. The second application of the algorithm is to use it for obtaining a low dimension readout from a reservoir. As seen in the examples the relatively small amount of invariant features are capable to capture the information about the driving signal (contrary to PCA, which does not seem to work well). Finally, the described algorithm will hopefully be a core part of the currently developed architecture for multidimensional data processing briefly described in Section 2.2.

This report contains initial findings on learning distortion invariant features from reservoirs. There are multiple open questions that have to be further investigated:

- The current algorithm simply reads out features from the reservoir, thus fully relying on it to obtain nice separation of the driving signal and distortion components. Potentially much better features could be extracted if the reservoir itself would be adapted. Thus possibility of having feedback connections from the extracted features or internal adaptation of the reservoir is currently under investigation.

- One of the shortcomings of the current algorithm is that two signals are required for the training phase; they must have an identical driving signal and should differ only by the distortion that is applied. It is interesting to investigate how this constraint could be lifted allowing to learn, for example, time warping invariant features.

- In the described handwriting example the two training data sequences were generated by using the same digit instances with a random shift added. It

is interesting to analyze if algorithm can work if different instances of the same digits were used instead.

- Current learning is performed offline, however for some applications it might be beneficial to obtain an online version of it.

4 Acknowledgments

The research that led to the results reported in this paper was funded through the EU FP7 project ORGANIC (grant agreement number 231267). The author would like to thank Prof. Herbert Jaeger and Mantas Lukoševičius for valuable discussions and advice.

References

[GS09] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 21, pages 545–552. MIT Press, 2009.

[Jae01] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks. *GMD Report 148, German National Research Center for Information Technology*, 2001.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LJ09] Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 08/2009 2009.

[WS02] Laurenz Wiskott and Terrence Sejnowski. Slow Feature Analysis: Unsupervised Learning of Invariances. *Neural Computation*, 14(4):715–770, 2002.