Improvement of speech recognition by nonlinear noise reduction

Krzysztof Urbanowicz and Holger Kantz

Max Planck Institute for Physics of Complex Systems
Nöthnitzer Str. 38
D–01187 Dresden, Germany

(Dated: March 17, 2006)

Abstract

The success of nonlinear noise reduction applied to a single channel recording of human voice is measured in terms of the recognition rate of a commercial speech recognition program in comparison to the optimal linear filter. The overall performance of the nonlinear method is shown to be superior. We hence demonstrate that an algorithm which has its roots in the theory of nonlinear deterministic dynamics possesses a large potential in a realistic application.

PACS numbers: 05.45.Tp,05.40.Ca

Keywords: Chaos, noise reduction, time series, speech recognition

*Electronic address: urbanow@pks.mpg.de
†Electronic address: kantz@pks.mpg.de
It is a common nuisance that in signal recording, signal transmission, and signal storage, perturbations occur which distort the original signal. In many cases these perturbations are purely additive, so that in principle they could be removed once identified. If these perturbations are sufficiently uncorrelated and appear to be random, they are called noise. Noise reduction is hence an important data processing task. Traditionally, the distinction of signal and noise is done in the frequency domain, where noise contributes with a broad background to a signal which should occupy distinguished frequency bands only. If the signal itself is irregular in time and hence has a broad power spectrum, spectral methods have a poor performance. Nonlinear methods for noise reduction assume and exploit constraints in the clean signal which are beyond second order statistics and hence are most efficiently captured by non-parametric dynamical modelling of the signal. In this paper it is shown that a particular such algorithm is able to denoise a recorded human voice signal so that the recognition rate of a commercial speech recognition program is significantly enhanced. Hence, we have not only a more suitable quantifier for the success of noise reduction on human voice, but we also verify that noise reduction algorithms do what they should do, namely enhance the intelligibility of the signal. This latter aspect is highly non-trivial, since every noise reduction algorithm when removing noise also distorts the signal. Finding merely a positive gain hence does not guarantee that a human (or algorithm) really understands better the meaning of the signal.

I. INTRODUCTION

Noise reduction and source separation are ubiquitous data analysis and signal processing tasks. In the analysis of chaotic data, noise reduction became a prominent issue about 15 years ago. Since the analysis of chaotic data in terms of dimensions, entropies and Lyapunov exponents requires an access to the small length scales (small scale fluctuations of the signal), already a moderate amount of measurement noise on data is known to be destructive. On the other hand, a deterministic source of a signal, albeit potentially chaotic, supplies redundancy which enables one to distinguish between signal
and noise and eventually to remove the latter to some extend. Noise reduction schemes which exploit such dynamical constraints were proposed in [1, 2, 3, 4, 5, 6, 7, 8, 9] and where tested on many data sets. Since such algorithms were designed to treat chaotic data, they do not make use of spectral properties of data and can, in principle, even remove in-band noise, which is noise whose high frequency spectrum is identical to the spectrum of the signal.

Human voice is a typical non-stationary signal, where noise reduction is a relevant issue. In telecommunication, in the development of hearing aids, and in automatic speech recognition, noise contamination of the speech signal poses severe problems. Having multiple simultaneous recordings, noise reduction is also known as blind source separation. However, most often a single recording only is available. In a previous study [10, 11] we demonstrated that nonlinear noise reduction can cope with noise on human speech data and has a performance, which is comparable to advanced spectral adaptive filter banks. In order to look through the main difference in concepts of linear and nonlinear methods of noise reduction that are presented in this paper see Table I.

The performance of a noise reduction scheme is usually measured as gain in dB. For this purpose, one starts with a clean signal $\tilde{x}(t)$, numerically adds noise to $\tilde{x}(t)$ resulting in $x(t)$, and then applies the noise reduction scheme without making use of $\tilde{x}(t)$. When we call the result of the noise reduction $y(t)$, then the gain is defined as the logarithm of the ratio of the noise power before and after the noise reduction, which is

$$
\text{gain} = 10 \log \left( \frac{\langle (x(t) - \tilde{x}(t))^2 \rangle}{\langle (y(t) - \tilde{x}(t))^2 \rangle} \right),
$$

where $\langle \ldots \rangle$ indicate the time average. This quantity has three drawbacks, namely i) it cannot be computed without knowledge of the clean signal $\tilde{x}(t)$, ii) it can be negative if the initial noise level is low, since distortion of the signal by the filter can be stronger than the reduction of noise, and iii) it does not quantify whether the intelligibility of the signal was improved by the noise reduction. Therefore, we employ in this paper a commercial speech recognition program as a quantifier of the success of noise reduction. The relative number of words which are not correctly recognized is taken as a quantifier for the signal corruption, regardless of whether this is noise or some systematic distortion which might be introduced by the noise reduction scheme.

In this paper we briefly recall the algorithm including its adaptation for the treatment of
TABLE I: The conceptional comparison between linear methods, LP methods and Hybrid method.

<table>
<thead>
<tr>
<th>Methods/Concepts</th>
<th>Linear methods (eg. low pass filter)</th>
<th>Nonlinear methods (eg. GHKSS)</th>
<th>Hybrid methods (eg. LPNCF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finding neighbours</td>
<td>in time</td>
<td>in space</td>
<td>in time and in space</td>
</tr>
<tr>
<td>Finding corrections</td>
<td>smoothing in time</td>
<td>smoothing in phase space</td>
<td>smoothing using the information in time and in space</td>
</tr>
<tr>
<td>Noise estimation</td>
<td>in Fourier space</td>
<td>violation of constraints in phase space</td>
<td>violation of constraints in time and in space</td>
</tr>
</tbody>
</table>

voice data, which are nonstationary limit cycle-like signals with embedded noisy segments (stemming from the fricatives). We then apply it to data samples which without added noise are perfectly recognized by the speech recognition software. We demonstrate that our optimised noise reduction scheme does not reduce the recognition rate when it is applied to clean data, and that it improves the recognition rate when it is applied to noisy data, which is comparable with a reduction of the noise amplitude by about 1/2. Noise level $\%N$ is defined as 100 times the ratio of the standard deviation of noise $\sigma$ and standard deviation of data $\sigma_{data}$: $\%N = 100\frac{\sigma}{\sigma_{data}}$.

II. METHOD

For the purposes of noise reduction (NR) in voice signals we use a combination of the GHKSS [9] and the LPNCF [12] methods. The GHKSS method is one version of the Local Projective (LP) [1, 2, 3, 4, 5, 6, 7, 8, 9] method that was developed for chaotic signals corrupted by noise. Assuming that the clean data are confined to some deterministic attractor in a reconstructed state space, which itself is locally a subset of a smooth manifold, the method aims at identifying this local manifold in linear approximation and to project the noisy state vector (which due to noise is not confined to this hyperplane) onto the local manifold. Algorithmically, this means to identify a neighbourhood in the delay embedding...
space and to perform a Singular Value Decomposition of a particular covariance matrix. Some refinements are described in [9].

The LPNCF method, which was particularly developed for chaotic flows, makes use of nonlinear constraints which appear because of the time continuous character of the flow. These constraints are functionals of the state vectors which assume the value 0 for dense sampling of deterministic data. Let \( \{x_i\} \) for \( i = 1, 2, \ldots, N \) be the time series. The corresponding clean signal we denote as \( \{\tilde{x}_i\} \), so when the measurement noise \( \{\eta_i\} \) is present we come to the formula \( x_i = \tilde{x}_i + \eta_i \) for \( i = 1, 2, \ldots, N \). We can define the time delay vectors \( x_i = (x_i, x_{i-1}, \ldots, x_{i-(d-1)}) \) as our points in the reconstructed phase space. Then we can choose two neighbours \( x_k, x_j \in X_{nN}^N \) to vector \( x_n \) (\( X_{nN}^N \) is the set neighbours of the point \( x_n \)). Let us introduce the following function [13]

\[
G_n(s) = x_{n-s}(x_{k+1-s} - x_{j+1-s}) + x_{k-s}(x_{j+1-s} - x_{n+1-s}) + x_{j-s}(x_{n+1-s} - x_{k+1-s}),
\]

for \( s = 0, 1, \ldots, d - 1 \).

The function \( G_n(s) \) vanishes for clean one-dimensional systems because it appears after eliminating \( a \) and \( b \) from following equations:

\[
\begin{align*}
\tilde{x}_{n+1} &= a\tilde{x}_n + b \\
\tilde{x}_{k+1} &= a\tilde{x}_k + b \\
\tilde{x}_{j+1} &= a\tilde{x}_j + b.
\end{align*}
\]

In the case of higher dimensional systems function \( G_n(s) \) does not always vanish but is altering slowly in time for dense sampling.

Now one can check that for a highly sampled clean dynamics there can be derived such a constraint

\[
C^m_n = \sum_{k=0}^{m-1} (-1)^l G_n(k) \approx 0,
\]

\[
l = \begin{cases} 
0 & \text{if } k = 0, \\
 k + \sum_{s=1}^\text{int}(\log_2(k)) \text{int}(k/2^s) & \text{if } k > 0,
\end{cases}
\]

where \( \text{int}(z) \) is a integer part of \( z \) and \( \log_2(z) \) is a logarithm with a base 2 from \( z \). Similarly as in LP methods the constraints [4] are satisfied in this approach by application of the method.
of Lagrange multipliers to an appropriate cost function. Since we expect that corrections to noisy data should be as small as possible, the cost function is assumed to be the sum of squared corrections $S = \sum_{s=1}^{N} (\delta x_s)^2$, where $\delta x_n$ are the corrections of the NR method connected to $x_n$, such that resulting time series of the NR method $y_n$ is set as $y_n = x_n + \delta x_n$ ($n = 1, 2, \ldots, N$). The method is a compromise between time and space integration methods. In the constraints there appear neighbours in space together with their pre-images. Hence it combines spatial and temporal vicinity and can perform better than standard time averaging or standard LP methods. This is because the size of the neighbourhood in time and in space is smaller in the LPNCF method than in standard methods which use only time or space averaging. Empirically we found the optimal noise reduction a method which links the LPNCF and the GHKSS methods. Corrections coming from the GHKSS and the LPNCF methods are added and divided by half.

It is known that the voice signal in the most of the time has many similarities with a flow \cite{10}, which means it represents smooth anharmonic oscillations with a typical frequency of about 200 Hz, for men. However, articulated human voice is a concatenation of different phonemes, so that the frequency, amplitude and, most importantly, the wave form of the oscillation varies tremendously on time scales of about 50 to 200 ms, causing that this signal is highly nonstationary. A qualitatively different component in articulated human voice is due to fricatives and sibilants, which are high frequency broad band noise-like parts of the signal. Such a sound starts around $n=41200$ in Fig. \textbf{1} (a). All nonlinear noise reduction schemes are very suitable for removing noise from anharmonic oscillations but they have the tendency to suppress strongly the fricatives and sibilants. Since the latter are of utmost relevance for a correct recognition by a speech recognition algorithm, we have to take special care of these. Finally, there are pauses in the speech which are pure noise after noise contamination of the signal. It is important to remove the noise during the speech pauses, so that the beginning and ending of words is correctly identified by the recognition algorithm. A particular challenge lies in these two opposing requirements: noise like fricatives should not be suppressed, whereas noise during speech pauses must be eliminated.

So the important modification of the standard algorithms for stationary data here is to identify the fricatives/sibilants and to treat them in a different way. As a first step we compute the auto-correlation function in a gliding window analysis (using windows of 300 sample points, which is about 14 ms). The location of the first maximum serves as a rough
FIG. 1: The voice time series of the word ”München”, recorded with a sampling rate of 22050 Hz (a time step is 1/22050 of second). The top panel is the clean signal, the next one shows the signal with added noise (%$N = 35\%$), the bottom the signal after noise reduction (the parameter $m$ calculated by the algorithm is as follows: from 35799 time step $m=124$, from 36501 $m=168$, from 41122 $m=7$, and from 42716 $m=127$).

estimate of the dominant period in the signal. We can then define windows in time during which the dominant frequency is rather constant. Obtaining the autocorrelation function is rather fast because we use previous calculations in next windows.

The estimated period inside a window is used to fix almost all of the parameters of the algorithm, e.g., the embedding dimension and embedding delay in nearest neighbourhood searching, the maximal embedding projection, a maximal range of neighbourhood in time etc. We also optimize some initial low-pass filtering of the signal by simple averaging on time windows which are about 1/40 of the local period determined by the auto-correlation function. The most important parameters of the algorithm are embedding dimension in neighbourhood searching $m$ (this parameter is equal to the period), maximal number of nearest neighbours (in our calculations it is 12), embedding dimension $q$ on which there is a projection (here we have the spectrum of $q$ which is 2-6 for LPNCF method and 3-12 for GHKSS method).

The way how we control the parameters of the noise reduction method through the observed period is such that for large periods the outcoming signal after reduction is much smoother than for short periods. This means that large corrections are made on those parts
of the signal where large periods are detected. If we detect a short autocorrelation time and a low variance, both less than some threshold, we consider this part of the signal as pure noise. We hence set by hand the period to its maximal value, in order to flatten the signal to zero. Sounds like 's', 'tch', 'h' are like noise with very little periodicity but the energy flow (here variance) is much higher than for the noise (see the Fig. 1a) starting from n=41200 begins 'chen'). In order to prevent the algorithm from removing these parts of the signal, we do not do any corrections when the detected period is less than 6 sample intervals and the variance is high. All parameters were optimized empirically and might depend on the voice recognition software, on the language spoken, and perhaps even on the speaker.

In order to speed up the algorithm, we make use of the observation that a delay vector $x_j$ which is a neighbour of a point $x_n$ gives rise to a delay vector $x_{j+s}$ which most surely is also neighbour of $x_{n+s}$, if $s < 30$. Hence, the neighbour search is only done every 30 time steps, whereas in between the “old” neighbourhood is simply translated in time. Due to the rather large embedding dimensions $q$, it is sufficient to perform the correction for every 6th point only, since the final time series is the arithmetic mean of all corrected vectors $x_n$, so that corrections of unprocessed data come from processing the points consecutive in time.

III. LINEAR FILTER

In order to do a comparison to the well-known linear filtering we apply the Wiener filter to the noisy signals. Since we use white noise, the noise spectrum if fully determined by the noise level. Knowing the noise level we can employ a perfect linear filter, but in practice we have to estimate the noise level from the data using the power spectrum. For the purposes of our linear filter we estimate the noise standard deviation $\sigma$ using the upper 40% percent of the of frequency domain, where the power spectrum is flat. Then the Wiener filter can be described as follows. If $S_{signal}^k$ is the amplitude of Fourier Transform of the noisy signal, additivity of the noise and independence of the noise and signal tell us:

$$ (S_{signal}^k)^2 = (S_{noise}^k)^2 + (S_{clean}^k)^2. \quad (5) $$

The action of the optimal linear filter for white noise consists in rescaling the amplitudes in Fourier space of the signal by the use of noise variance:

$$ S_{after}^k = \sqrt{(S_{signal}^k)^2 - \sigma^2}. \quad (6) $$
The inverse Fourier transform on the corrected amplitudes $S_k^{after}$ keeping phases from Fourier transform of the noisy signal yields the new signal. One can prove that knowing the exact value of noise level such an algorithm is the optimal linear method of noise reduction [14].

IV. RESULTS AND CONCLUSIONS

The speech recognition is done by the commercial software program Linguatec ViaVoice Pro release 10 for German, which is based on the IBM recognition algorithm viavoice. The difficulty in speech recognition lies in the required training of the algorithm in order to adapt to a specific speaker. In order to make our results reproducible, we downloaded the sample sentences together with the speaker specific auxiliary data files, from the distributor [15].

We were working on the following recorded sentences in German:


As outlined above, the method was optimized not only on performance and maximal gain but also for speed. The algorithm is rather fast. The data recorded with 22050 Hz (which is required by the recognition software) gives about a million data for 45 seconds, but nevertheless the method works only ca. 130 seconds, which is 3 times slower than on-line noise reduction, on a 2 GHz processor AMD Athlon. In order to create noisy signals, we first convert the speech stored in the wave-format (.wav) into real numbers, representing the time series of the sound amplitude. We add independently drawn Gaussian random numbers of the desired variance and apply the back-conversion into the wave-format for the determination of the recognition rate. The noise reduction is done on the real valued time series, again with a successive conversion for recognition. In Fig. we show the signal which corresponds to the word "München". In the upper panel (a) there is a clean signal. In the middle, part (b), the noisy signal with standard deviation (SD) of noise equal 0.009 and in the bottom, part (c), the noisy signal after NR. As pointed out before, around n=41200 is the fricative “ch” (pronounced as [ç]). The autocorrelation function suggests a period of 4, and the variance is much larger than for noise on a pause which can be seen on the
beginning of the signal (b). The signal (a) and (c) are recognized well by the program but
the noisy signal (b) is badly represented in the recognition of the full text. Hence, although
the noise level appears to be small, the recognition software is considerably confused and
the recognition rate drops significantly.

The algorithm of the recognition program enforces it to generate reasonable words only,
which is, it only generates words form an internal dictionary. Therefore, misunderstanding
by the recognition software cannot lead to wrong letters inside words, but only to the
replacement of correct words by some other words. Rarely, the wrongly recognised word
resembles in sound the original word. If the system is strongly misled, it can generate a long
wrong word out of several short ones or vice versa, such that the number of words is not
conserved. However, the total number of letters is more or less unchanged. Hence, in order
to do statistics on the recognised sentences, we use the following indicator: We identify the
correctly recognised words and those words which are not part of the original sentences, and
then count the numbers of letters inside these two groups of words.

In Fig. 3 we show these differences and similarities as a function of the amount of noise
added. Without noise reduction, a standard deviation of the added noise of more than 0.003
leads to mis-interpretations of the speech recognition software. If one takes into account that
every wrongly recognised letter requires a correction by hand, the recognition is useless when
more than half of the number of characters has to be replaced. This situation occurs for
noise levels above 0.007 (%N = 28%). After noise reduction, the recognition rate increases
considerably. However, for very low noises distortions of the signal introduced by the noise
filtering leads to a small number of wrongly recognised letters.

The results can be interpreted in an even better way, when we compute the noise level
after noise reduction as \( \langle (y(t) - \bar{x}(t))^2 \rangle \) and present the recognition success as a function
of the latter. This is shown in Fig. 4. In this figure the two lines with squares and circles
representing the result without noise reduction are the same as in Fig. 3. The two lines with
triangles in Fig. 4 representing the results after noise reduction, now lie almost on the same
curve as without noise reduction. We can therefore conclude that the remaining distortion
of the signal after noise reduction has the same affect as white noise with a corresponding
amplitude, and that the effective noise level \( \langle (y(t) - \bar{x}(t))^2 \rangle \) (which unfortunately can only
be computed in the unrealistic situation where the clean signal is available) determines the
recognition rate uniquely. This happens so only if we use the same parameters for all noise
levels in the noise reduction and if the noise magnitude is similar in whole of data set after noise reduction.

For comparison the results of the best linear filter which was described in section III are shown in Fig. 2. We see that such a filter reduces the recognition rate for small noise levels and improves for high ones. For low noise levels, these are overestimated from the power spectrum, so that the high-frequency components of the signal are strongly reduced. In the case of high noise levels their estimation is rather good so reduction is well done improving the recognition ability.

The gain parameter corresponding to Figs 2, 3 is presented in Fig. 5. One can see that the efficiency of nonlinear and linear noise reduction is comparable in these both cases and not very high especially for small noise levels. For low noise levels, the gain is even negative because the clean signal is distorted by either of the noise reduction schemes. The nonlinear method introduces some distortion everywhere where the voice is not well represented by a flow and is not sufficiently smooth (see Fig. 1). Also for larger noise levels the gain is small compared to gains obtained in [11], which reflects that the data structures which must be preserved for a good recognition cannot be directly translated into gain. Also, however, the noise levels which are relevant in the present study are much smaller than those considered in [11], since larger noise levels completely destroy the speech recognition. We see that the gain parameter is not a good indicator of the recognition rate. We most well see this property when we compare the recognition rate for small noise levels for nonlinear and linear filters. The similar gains leads to apparent recognition rates in favor of nonlinear NR. Even if the gain is negative the program for speech recognition is not much mislead in nonlinear NR case. On the above examples it is well seen that noise removing using chaos like features improve the recognition rate especially for intermediate noise levels and does not destroy the signal when the noise level is small.
FIG. 2: Plot of similarities and differences in letters of the correct text and texts recognized from noisy signals (squares and circles) or texts recognized from noisy signals after optimal linear noise reduction (open triangles). Standard deviation of added noise appears at the x-axis.

FIG. 3: Plot of similarities and differences in letters of the correct text and texts recognized from noisy signals (squares and circles) or texts recognized from noisy signals after nonlinear noise reduction (triangles). Standard deviation of added noise appears at the x-axis.

FIG. 4: Plot of similarities and differences in letters of the correct text and texts recognized from noisy signals (squares and circles) or texts recognized from noisy signals after nonlinear noise reduction (triangles). Standard deviation of left noise after noise reduction appears at the x-axis.

FIG. 5: Plot of the gain parameter versus SD of added noise in the signal.


