# 15 Monaural Speech Separation by Support Vector Machines: Bridging the Divide Between Supervised and Unsupervised Learning Methods

Sepp Hochreiter[1] and Michael C. Mozer[2]

[1] Institute of Bioinformatics
Johannes Kepler University
4040 Linz, Austria
[2] Department of Computer Science
University of Colorado
Boulder, CO 80309

**Abstract.** We address the problem of identifying multiple independent speech sources from a single signal that is a mixture of the sources. Because the problem is ill-posed, standard independent component analysis (ICA) approaches which try to invert the mixing matrix fail. We show how the unsupervised problem can be transformed into a supervised regression task which is then solved by support-vector regression (SVR). It turns out that the linear SVR approach is equivalent to the sparse-decomposition method proposed by [1, 2]. However, we can extend the method to *nonlinear ICA* by applying the "kernel trick." Beyond the kernel trick, the SVM perspective provides a new interpretation of the sparse-decomposition method's hyperparameter which is related to the input noise. The limitation of the SVM perspective is that, for the nonlinear case, it can recover only whether or not a mixture component is present; it cannot recover the strength of the component. In experiments, we show that our model can handle difficult problems and is especially well suited for speech signal separation.

## 15.1 Introduction

Independent component analysis (ICA) [3–5] attempts to recover multiple source signals from one or more observed signals that are mixtures of the sources. ICA algorithms rely on the assumption that the sources are statistically independent, and most make the further assumption that mixtures are linear combinations of the sources. Well known ICA methods like "infomax" [6, 7], maximum likelihood approaches [8], entropy and cumulant based methods [9–11] have the restriction that the number of source and mixture signals must be equal. The restriction stems from the fact that these methods recover the sources by inverting the mixing matrix. To avoid ill-posed problems, the inversion requires at least as many observations as sources.

In many real world applications, only one observation is available. Speech processing is an important example in which this situation is often true. However, many nonspeech acoustic signals, such as bird songs, music, and



*S. Makino et al. (eds.), Blind Speech Separation,* 411–432.

traffic, are recorded with only one or two microphones. Other technological applications are based on a single mixture time series, such as mobile communication signals with direct sequence code division multiple access (DS-CDMA) [12]. Standard ICA approaches cannot be used in these cases.

The human auditory system is unmatched in its ability to robustly distinguish among multiple sound sources from two mixtures—the ears. It can even extract sources from monaural recordings. In some cases, separation of speech signals is easy because the signals occur in different frequency bands. However, when simple physical characteristics do not distinguish the sources, the task is extremely difficult.

As one realizes when listening to an orchestra, the human auditory system is able to separate complex mixtures from just two sources. The conductor is able to isolate individual melody lines, instruments, or even musicians from the ensemble, whereas a naive audience member may not. The difference between the conductor and the audience member is the conductor's knowledge and familiarity with the sound patterns that constitute the performance. One could even imagine that the conductor has a dictionary of sound *atoms*—canonical or prototypical musical phrases and timbres—and identification of components comes by isolating the atoms from the mixture. This argument carries over to speech processing. Our conjecture is that people can separate speech so well because of the accumulation of speech experience since birth. These experiences allow for the construction of a dictionary of sounds corresponding to vowels, words, etc. The dictionary contains not only one entry per vowel or word but multiple entries which allow to distinguish female from male and help to identify words spoken in a dialect or foreign accent.

Several ICA approaches have adopted the idea of using a dictionary to extract multiple sources from a smaller number of mixtures, or even one mixture [1, 2, 13, 14]. The dictionary can be composed of primitive functions (e.g., Fourier bases, wavelet packages, or Gabor functions) [1, 2], it can be predefined based on prior knowledge, or it can be trained to fit the problem [14–17]. Zibulevsky and Pearlmutter [1, 2] specify not only a dictionary, but also a prior that enforces sparseness—i.e., an expectation as to how many sources will be present simultaneously. All these approaches are restricted to mixtures consisting of linearly superimposed dictionary atoms; this restriction is necessary to avoid ambiguity in the problem.

In this chapter we show that the *sparse-decomposition method* of Zibulevsky and Pearlmutter can be reinterpreted as $\epsilon$-support vector regression ($\epsilon$-SVR) [18–23] when there is a single mixture and a Laplacian prior.

By drawing the connection between ICA—an unsupervised learning task—and support-vector regression—traditionally used for supervised learning—we show how an unsupervised task can be transformed into a supervised framework. This transformation may be useful to other problems as well. The key idea of the transformation is to use a predefined set of examples—the dictionary atoms—for generating target values for the

supervised learning framework. In our approach, each unsupervised example is paired with each atom, and a fixed function is used to generate a "target" value for the example, i.e., the target for the SVR. Typically, the target value might be a measure of similarity between the dictionary atom and the example, such as a dot product. Using this approach for speech signal processing, which may contain a mixture of multiple speakers, we break the input signal into short segments which serve as the unsupervised examples. Using a prespecified dictionary of atomic speech signals [16], we use our method to determine, for each segment, which combination of the atoms is present. Our approach may also be useful for DS-CDMA mobile communication, where the dictionary consists of spreading sequences of the users.

The $\epsilon$-SVR framework allows us to obtain sparse solutions. The sparsity constraint reflects the fact that at any point in the input signal, only a few mixture components (i.e., a few speakers) should be present. Within the SVR framework, support vectors will correspond to exactly those dictionary atoms that are correlated with the input signal, but which are mutually decorrelated. The support-vector machine determines which target values are spurious, which are produced by superimposition, and which indicate that a dictionary entry is present in the input signal.

The idea of applying SVR to unsupervised problems was also suggested for filtering [24], and recently for estimating missing values in cDNA microarrays [25].

By drawing an analogy between $\epsilon$-SVR and the sparse-decomposition method for ICA, we obtain a new interpretation to the sparse-decomposition method's hyperparameter that determines the degree of sparseness. Further, the analogy yields a generalization of the sparse-decomposition method to allow for nonlinear transformations of the sources before they are mixed, and to allow for a further nonlinear transformation in the process of identifying dictionary atoms in the mixture.

We demonstrate our approach with experiments using noisy mixtures of speech with a single microphone. As we show, our approach incorporates nonlinear transformations of the dictionary atoms that achieve a degree of robustness and invariance to irrelevant characteristics of the speech signal. Examples of transformations and the corresponding invariants that we consider include: taking the local variance of the absolute value or square of the waveform to reduce sensitivity to the sign of the waveform combined with adding a constant value to the waveform (a nonlinear transformation, and a nonlinear invariant); or the power spectrum which is invariant to temporal shifts of the waveform (linear transformation and nonlinear invariant). These transformations are highly relevant for speech separation where, for example, shift invariance avoids the need to segment the speech signal.

## 15.2 Viewing Sparse Decomposition as $\epsilon$-SVR

In this section, we review the sparse-decomposition method introduced by Zibulevsky and Pearlmutter [1], focusing on the case of a single mixture signal. We will also describe the relation between sparse decomposition and $\epsilon$-support vector regression ($\epsilon$-SVR).

### 15.2.1 The Sparse-Decomposition Method

Denote the mixture signal by $\boldsymbol{x} \in \mathbb{R}^L$. In the case of speech, the signal might correspond to a window of $L$ consecutive samples from a longer input stream. We assume a dictionary matrix, $\boldsymbol{S} \in \mathbb{R}^{L \times P}$, whose columns consist of the $P$ atomic signals of length $L$. We assume a generative process in which the mixture is created by first choosing a set of dictionary atoms and then combining them linearly with noise:

$$\boldsymbol{x} \;=\; \boldsymbol{S}\,\boldsymbol{c} \;+\; \boldsymbol{\nu} \;=\; \sum_{i=1}^{P} c_i\; \boldsymbol{s}^i \;+\; \boldsymbol{\nu}, \tag{15.1}$$

where $\boldsymbol{c} \in \mathbb{R}^P$ is a vector of weighting coefficients, one per atom, $\boldsymbol{\nu} \sim N\left(0, \sigma^2\right)$ is an $L$-dimensional i.i.d. additive Gaussian noise vector, and $\boldsymbol{s}^i$ the $i$th atom in the dictionary: $\boldsymbol{s}^i \;=\; [\boldsymbol{S}]_i$. Figure 15.1 illustrates the generative process that produces the mixture.

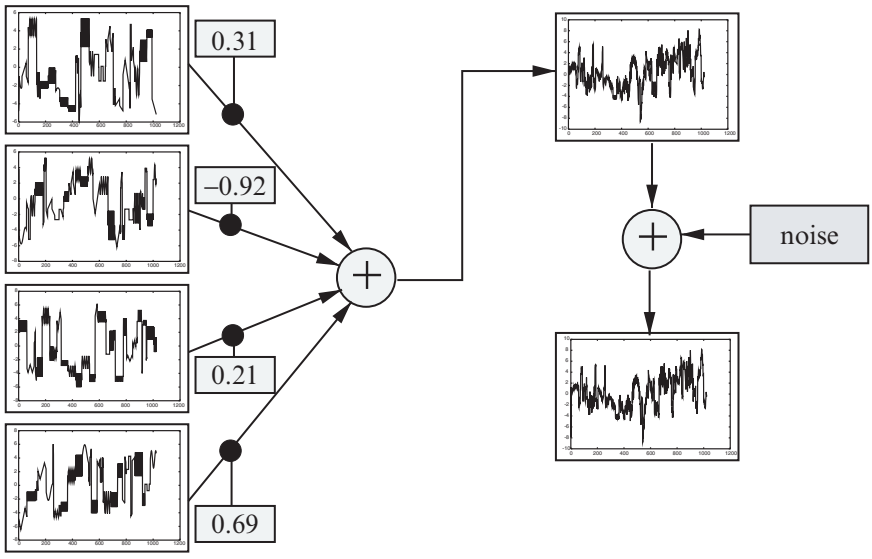


**Fig. 15.1.** The data generation process. Four atoms from the dictionary $\boldsymbol{S}$ are weighted by real nonzero coefficients and are added together with noise resulting in the mixture. The goal is to recover the nonzero weighting factors, or at least to detect whether an atom is present in the mixture.

The goal of the sparse-decomposition method is to determine the coefficient vector, $\hat{\boldsymbol{c}}$, that satisfies two properties: (I) $\boldsymbol{x}$ must be well approximated by $\hat{\boldsymbol{c}}$, and (II) $\hat{\boldsymbol{c}}$ is sparse, i.e., it has elements of small magnitude. These two properties are achieved by a Bayesian approach in which property 1 is characterized by the likelihood, $p(\boldsymbol{x} \mid \boldsymbol{c}, \boldsymbol{S})$, and property 2 is characterized by the prior, $p(\boldsymbol{c})$. Thus, the approach tries to maximize the posterior

$$p(\boldsymbol{c} \mid \boldsymbol{x}, \boldsymbol{S}) \;\propto\; p(\boldsymbol{x} \mid \boldsymbol{c}, \boldsymbol{S})\; p(\boldsymbol{c}),$$

where we use "$\propto$" because we omit the constant normalization factor in the denominator of Bayes rule. Given the Gaussian noise model, the likelihood is

$$p(\boldsymbol{x} \mid \boldsymbol{c}, \boldsymbol{S}) \;\propto\; \exp\left(-\frac{1}{2\,\sigma^2}\,(\boldsymbol{x} \;-\; \boldsymbol{S}\;\boldsymbol{c})^2\right).$$

To enforce sparseness of the coefficients, a Laplacian prior for $\boldsymbol{c}$ is used with $\|\boldsymbol{c}\|_1 = \sum_{i=1}^{P} |c_i|$ we have:

$$p(\boldsymbol{c}) \;\propto\; \exp\left(-\frac{\epsilon}{\sigma^2}\,\|\boldsymbol{c}\|_1\right).$$

Consequently, the posterior is

$$p(\boldsymbol{c} \mid \boldsymbol{x}, \boldsymbol{S}) \propto \exp\left(-\frac{1}{\sigma^2}\left[\frac{1}{2}\,(\boldsymbol{x} \;-\; \boldsymbol{S}\;\boldsymbol{c})^2 \;+\; \epsilon\;\|\boldsymbol{c}\|_1\right]\right).$$

The solution, $\hat{\boldsymbol{c}}$, is obtained by maximum a posteriori (MAP) search. Taking the log of the posterior, flipping its sign, and ignoring irrelevant constant terms and factors, we obtain the minimization problem

$$\hat{\boldsymbol{c}} \;=\; \arg\min_{\boldsymbol{c}}\; \frac{1}{2}\,(\boldsymbol{x} \;-\; \boldsymbol{S}\;\boldsymbol{c})^2 \;+\; \epsilon\;\sum_{i=1}^{P} |c_i|.$$

This unconstrained optimization problem can be turned into a constrained optimization problem in which $\boldsymbol{c}$ is split into two vectors such that $\boldsymbol{c} = \boldsymbol{c}^+ - \boldsymbol{c}^-$, where $\boldsymbol{c}^+$ and $\boldsymbol{c}^-$ contain the magnitudes of the positive and negative coefficients of $\boldsymbol{c}$, respectively. The MAP solution $\{\hat{\boldsymbol{c}^+}, \hat{\boldsymbol{c}^-}\}$ is

$$\begin{aligned}
\arg\min_{\boldsymbol{c}^+,\boldsymbol{c}^-}\;\; & \frac{1}{2}\left(\boldsymbol{c}^+ - \boldsymbol{c}^-\right)^T \boldsymbol{S}^T \boldsymbol{S}\left(\boldsymbol{c}^+ - \boldsymbol{c}^-\right) \;- \\
& \boldsymbol{x}^T \boldsymbol{S}\left(\boldsymbol{c}^+ - \boldsymbol{c}^-\right) \;+\; \epsilon\;\boldsymbol{1}^T\left(\boldsymbol{c}^+ + \boldsymbol{c}^-\right) \\
\text{s.t.}\;\; & 0 \le c_i^+, c_i^- \le C, \qquad (15.2)
\end{aligned}$$

where $T$ is the transposition operator, $\boldsymbol{1}$ is the vector of ones, and $C$ is an upper bound that can serve as an additional constraint on the solution (which was not part of the original formulation by Zibulevsky and Pearlmutter).

We will show that *this formulation is $\epsilon$-support vector regression ($\epsilon$-SVR)* [18], but in order to do so, we must first give a brief overview of $\epsilon$-SVR.

### 15.2.2 $\epsilon$-Support Vector Regression

$\epsilon$-SVR is a supervised approach to regression in which we are given training data $\left\{\left(\boldsymbol{s}^i, y_i\right), \ldots, \left(\boldsymbol{s}^P, y_P\right)\right\}$, where $\boldsymbol{s}^i \in \mathbb{R}^L$ and $y_i$ is a scalar. The goal is to produce a function, $h$, such that $h\left(\boldsymbol{s}^i\right)$ closely approximates $y_i$. In the linear formulation of $\epsilon$-SVR, $h\left(\boldsymbol{s}^i\right) = \left\langle \boldsymbol{w}, \boldsymbol{s}^i \right\rangle$, where $\boldsymbol{w} \in \mathbb{R}^L$, and $\langle ., . \rangle$ denotes the dot product. In the nonlinear formulation, $h\left(\boldsymbol{s}^i\right) = \left\langle \boldsymbol{w}, \boldsymbol{\theta}\left(\boldsymbol{s}^i\right)\right\rangle$, where $\boldsymbol{\theta}$ is a mapping from $\mathbb{R}^L$ to $\mathbb{R}^L$.

Note we consider regression functions through the origin. Consequently, the formulation excludes a constant offset term. The $\epsilon$-SVR attempts to obtain a "flat" function by minimizing $\frac{1}{2}\|\boldsymbol{w}\|^2$, but subject to the constraint that the fit is good enough, as quantified by the constraint

$$\left|y_i \; - \; h\left(\boldsymbol{s}^i\right)\right| \; < \; \epsilon \; + \; \xi_i \tag{15.3}$$

for all $i$. The parameter $\epsilon$ is a measure of how accurate the fit needs to be, or intuitively, a measure of the noise in the data. The *slack variables* $\xi_i \geq 0$ allow for the fact that it may not be possible to find an $h$ that satisfies the $\epsilon$ accuracy criterion.

$$\xi_i \; = \; \max\{0, \; \left|y_i \; - \; h\left(\boldsymbol{s}^i\right)\right| \; - \; \epsilon\}.$$

However, to ensure that the deviations are minimal, the optimization attempts to minimize the magnitude of the slack variables as well. Specifically, the constrained optimization is over the objective function

$$\frac{1}{2}\|\boldsymbol{w}\|^2 + C \; \|\boldsymbol{\xi}\|_1,$$

where $C$ determines the trade off between the flatness of the function and the tolerance of prediction errors and

$$\|\boldsymbol{\xi}\|_1 = \sum_i \xi_i = \sum_i \max\{0 \; , \; \left|y_i - h\left(\boldsymbol{s}^i\right)\right| - \epsilon\} = \left|y_i - h\left(\boldsymbol{s}^i\right)\right|_\epsilon . \tag{15.4}$$

We divide (15.3) into an inequality set for which $\left|y_i - h\left(\boldsymbol{s}^i\right)\right| = y_i - h\left(\boldsymbol{s}^i\right)$, where $\xi_i$ in (15.3) is denoted by $\xi_i^+$, and into an inequality set for which $\left|y_i \; - \; h\left(\boldsymbol{s}^i\right)\right| \; = \; h\left(\boldsymbol{s}^i\right) \; - \; y_i$, where $\xi_i$ in (15.3) is denoted by $\xi_i^-$. We obtain for the linear case $h\left(\boldsymbol{s}^i\right) \; = \; \boldsymbol{w}^T \boldsymbol{s}^i$ as $\epsilon$-SVR optimization problem

$$\begin{aligned} \min_{\boldsymbol{w}, \boldsymbol{\xi}^+, \boldsymbol{\xi}^-} \quad & \frac{1}{2} \; \|\boldsymbol{w}\|^2 \; + \; C \; \sum_{i=1}^{l} \left(\xi_i^+ \; + \; \xi_i^-\right) \\ \text{s.t.} \quad & y_i \; - \; \boldsymbol{w}^T \boldsymbol{s}^i \; \leq \; \epsilon \; + \; \xi_i^+, \\ & \boldsymbol{w}^T \boldsymbol{s}^i \; - \; y_i \; \leq \; \epsilon \; + \; \xi_i^-, \\ & \xi_i^+ \; \geq \; 0 \; \text{ and } \; \xi_i^- \; \geq \; 0, \end{aligned} \tag{15.5}$$

It is known that the primal $\epsilon$-SVR optimization problem (15.5) possesses an alternative but entirely equivalent formulation, the *dual* formulation:

$$\begin{aligned} \min_{\boldsymbol{c}^+,\boldsymbol{c}^-} \quad & \frac{1}{2}\left(\boldsymbol{c}^+ - \boldsymbol{c}^-\right)^T \boldsymbol{S}^T\boldsymbol{S}\left(\boldsymbol{c}^+ - \boldsymbol{c}^-\right) - \\ & \boldsymbol{y}^T\left(\boldsymbol{c}^+ - \boldsymbol{c}^-\right) + \epsilon\,\mathbf{1}^T\left(\boldsymbol{c}^+ + \boldsymbol{c}^-\right) \\ \text{s.t.} \quad & 0 \le c_i^+, c_i^- \le C, \end{aligned} \tag{15.6}$$

where $\boldsymbol{S}$ is the matrix formed by $\boldsymbol{s}^i$: $\boldsymbol{s}^i = [\boldsymbol{S}]_i$. The coefficient $c_i$ are the Lagrange multipliers for the primal constraints in (15.5) which are split into positive and negative components, $c_i^+$ and $c_i^-$: $c_i = c_i^+ - c_i^-$. The $\boldsymbol{s}^i$ for which $c_i \neq 0$ are called *support vectors* or *support sources* or *support signals*. Note that

$$\left(\boldsymbol{c}^+ - \boldsymbol{c}^-\right)^T \boldsymbol{S}^T\boldsymbol{S}\left(\boldsymbol{c}^+ - \boldsymbol{c}^-\right) = \frac{1}{2}\sum_{i,j=1}^{P} c_i\, c_j \left\langle \boldsymbol{s}^i, \boldsymbol{s}^j \right\rangle.$$

For the nonlinear formulation we define the Gram matrix $\boldsymbol{K}$ as

$$K_{ij} = k(\boldsymbol{s}^i, \boldsymbol{s}^j) = \left\langle \boldsymbol{\theta}\left(\boldsymbol{s}^i\right), \boldsymbol{\theta}\left(\boldsymbol{s}^j\right)\right\rangle.$$

The Gram matrix can be written as $\boldsymbol{K} = \boldsymbol{\theta}(\boldsymbol{S})^T\boldsymbol{\theta}(\boldsymbol{S})$, where $\boldsymbol{\theta}(\boldsymbol{S})$ is the matrix formed by $\boldsymbol{\theta}(\boldsymbol{S})_i = \boldsymbol{\theta}(\boldsymbol{s}^i)$. The nonlinear dual formulation is therefore

$$\begin{aligned} \min_{\boldsymbol{c}^+,\boldsymbol{c}^-} \quad & \frac{1}{2}\left(\boldsymbol{c}^+ - \boldsymbol{c}^-\right)^T \boldsymbol{K}\left(\boldsymbol{c}^+ - \boldsymbol{c}^-\right) - \\ & \boldsymbol{y}^T\left(\boldsymbol{c}^+ - \boldsymbol{c}^-\right) + \epsilon\,\mathbf{1}^T\left(\boldsymbol{c}^+ + \boldsymbol{c}^-\right) \\ \text{s.t.} \quad & 0 \le c_i^+, c_i^- \le C. \end{aligned} \tag{15.7}$$

It can be seen that in (15.6) only $\boldsymbol{S}^T\boldsymbol{S}$ is replaced by $\boldsymbol{K}$ to obtain a nonlinear formulation.

As a consequence of the transformation from the primal to the dual formulation, the primal vector $\boldsymbol{w}$ can be expressed through the coefficients $c_i$ and the training data $\boldsymbol{s}^i$:

$$\boldsymbol{w} = \sum_{i=1}^{P} c_i\, \boldsymbol{s}^i$$

for the linear version and

$$\boldsymbol{w} = \sum_{i=1}^{P} c_i\, \boldsymbol{\theta}\left(\boldsymbol{s}^i\right)$$

for the nonlinear version. Consequently, for the linear formulation

$$h(\boldsymbol{s}) = \sum_{i=1}^{P} c_i \left\langle \boldsymbol{s}^i, \boldsymbol{s}\right\rangle = \boldsymbol{s}^T\boldsymbol{S}\,\boldsymbol{c}.$$

For the nonlinear formulation we obtain

$$h(\boldsymbol{s}) \;=\; \sum_{i=1}^{P} c_i \left\langle \boldsymbol{\theta}\left(\boldsymbol{s}^i\right), \boldsymbol{\theta}\left(\boldsymbol{s}\right)\right\rangle \;=\; \boldsymbol{\theta}(\boldsymbol{s})^T \boldsymbol{\theta}(\boldsymbol{S})\; \boldsymbol{c} \tag{15.8}$$

which can reformulated with a kernel as

$$h(\boldsymbol{s}) \;=\; \sum_{i=1}^{P} c_i \; k\left(\boldsymbol{s}^i, \boldsymbol{s}\right) \;=\; \boldsymbol{k}(s, \boldsymbol{S})^T \boldsymbol{c},$$

where $\boldsymbol{k}(s, \boldsymbol{S})$ is the vector

$$\boldsymbol{k}(s, \boldsymbol{S}) \;=\; \boldsymbol{\theta}(\boldsymbol{s})^T \boldsymbol{\theta}(\boldsymbol{S}) \;=\; \left(k(s, s^1), k(s, s^2), \ldots, k(s, s^P)\right).$$

### 15.2.3 The Relationship Between the Sparse-Decomposition Method and $\epsilon$-SVR

The sparse-decomposition method and $\epsilon$-support vector regression are equivalent: Optimization problem (15.2) is *identical* to optimization problem (15.6). To spell out the mapping, consider framing the sparse-decomposition method as $\epsilon$-SVR. The data for the $\epsilon$-SVR consists of $P$ training examples, where the input for example $i$ is the dictionary atom $\boldsymbol{s}^i$, $\boldsymbol{s}^i \in \mathbb{R}^L$, and the target for the example, $\boldsymbol{y}_i$, is the dot product of the mixture $\boldsymbol{x}$ and dictionary atom $\boldsymbol{s}^i$: $y_i = \boldsymbol{x}^T \boldsymbol{s}^i = \left\langle \boldsymbol{x}, \boldsymbol{s}^i \right\rangle$.

The $\epsilon$-SVR formulation gives an interpretation to the hyperparameter $\epsilon$ in the sparse-decomposition method. It is a measure of the noise level in the data, and it indirectly affects the number of $\hat{c}_i$ that are significantly nonzero. As depicted in Fig. 15.2, each example will have a target, $y_i$, that either

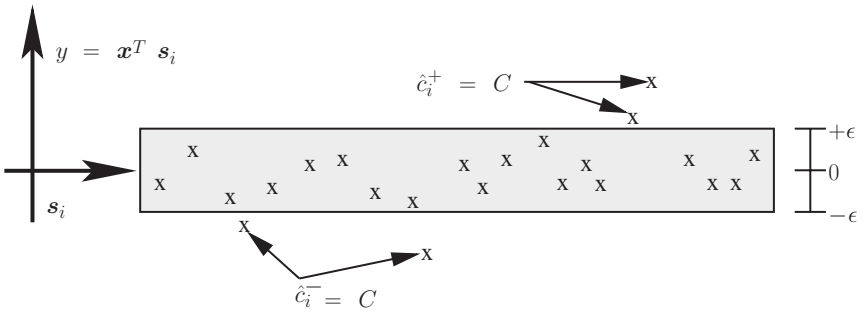


**Fig. 15.2.** The linear $\epsilon$–support vector regression corresponding to the sparse-decomposition method. Each "x" in the figure denotes a single training example in the $\epsilon$-SVR model. The horizontal axis is a one-dimensional depiction of the input space, and the vertical axis is the target output. The grey area, the $\epsilon$-tube, specifies the range of target outputs that are not significantly different from zero. The examples $i$ that lie outside the $\epsilon$-tube will have $|\hat{c}_i| = C$.

lies inside or outside the $\epsilon$-*tube*. The closer a target $y_i$ is to zero, the more nearly orthogonal is the mixture $\boldsymbol{x}$ to atom $\boldsymbol{s}^i$, and the less likely atom $i$ is to be present in the mixture. Thus, the $\epsilon$-tube distinguishes atoms that are likely to be relevant from those likely to be irrelevant. It turns out that any example $i$ lying outside the $\epsilon$-*tube* will have either $\hat{c}_i = C$ or $\hat{c}_i = -C$. In the sparse-decomposition formulation, $c_i$ indicates the degree to which a dictionary atom $i$ is present.

## 15.3 Nonlinear Formulation

The $\epsilon$-SVR framework also provides for a nonlinear approximation of $\boldsymbol{y}$ by introducing a nonlinear kernel $k(\boldsymbol{a}, \boldsymbol{b})$, where $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^L$. The dot products $\langle \boldsymbol{s}^i, \boldsymbol{s}^j \rangle$ in the $\epsilon$-SVR are replaced by

$$k\left(\boldsymbol{s}^i, \boldsymbol{s}^j\right) \;=\; \left\langle \boldsymbol{\theta}\left(\boldsymbol{s}^i\right), \boldsymbol{\theta}\left(\boldsymbol{s}^j\right) \right\rangle,$$

or in matrix notation $\boldsymbol{S}^T\boldsymbol{S}$ is replaced by the kernel matrix $\boldsymbol{K}$ with $K_{ij} = k\left(\boldsymbol{s}^i, \boldsymbol{s}^j\right)$. The interpretation of this kernel is that the $P$ vectors $\boldsymbol{s}^i$ are mapped by a function $\boldsymbol{\theta}$ into a feature space. The kernel is the dot product in the feature space: $k(\boldsymbol{a}, \boldsymbol{b}) = \langle \boldsymbol{\theta}(\boldsymbol{a}), \boldsymbol{\theta}(\boldsymbol{b}) \rangle$.

Exploiting the correspondence we have established between the sparse-decomposition method and the $\epsilon$-SVR, we can generalize the linear sparse-decomposition method to a nonlinear method by analogy to the nonlinear $\epsilon$-SVR. We present two different nonlinear extensions of the sparse-decomposition method. In both extensions, the dictionary atoms $\boldsymbol{s}^i$ are mapped into a feature space using a nonlinear kernel. In the first extension we assume that the mixture $\boldsymbol{x}$ arises from a nonlinear combination of atoms $\boldsymbol{s}^j$. In the second extension we assume that mixture is linear but prior to being mixed, the atoms undergo a nonlinear transformation. We consider two subclasses of the second extension: (a) we directly approximate the nonlinear transformation by applying this transformation to the atom waveform sources as well, and (b) we extract invariances of the nonlinear transformation through the feature map which maps into a space of components which are invariant under the nonlinear transformation. In this space of invariances, the dot products extract similarities between the mixture and the dictionary atoms.

### 15.3.1 Nonlinear Mixtures

In our first extension, we assume that the mixture $\boldsymbol{x}$ results from a nonlinear combination of the atoms $\boldsymbol{s}^1, \boldsymbol{s}^2, \ldots, \boldsymbol{s}^P$:

$$\boldsymbol{x} \;=\; g\left(\boldsymbol{s}^1, \boldsymbol{s}^2, \ldots, \boldsymbol{s}^P\right).$$

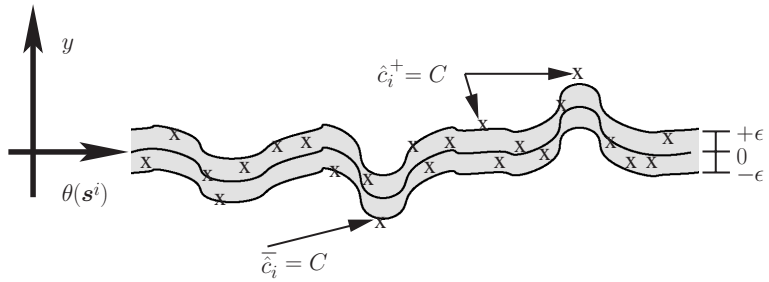


**Fig. 15.3.** The nonlinear $\epsilon$-SVR. The dictionary atoms $\boldsymbol{s}^i$ are mapped by $\boldsymbol{\theta}$ into a feature space.

We further assume that the nonlinear mixture $g$ can be expressed as a linear mixture in some feature space:

$$k\left(\boldsymbol{s}^i, \boldsymbol{x}\right) \;=\; k\left(\boldsymbol{s}^i, g\left(\boldsymbol{s}^1, \boldsymbol{s}^2, \ldots, \boldsymbol{s}^P\right)\right) \;=\; \sum_{j=1}^{P} k\left(\boldsymbol{s}^i, \boldsymbol{s}^j\right)\; c_j.$$

The feature space that allows for the linear mixing is defined by the mapping $\boldsymbol{\theta}(.)$, yielding the kernel $k$ already mentioned above:

$$k\left(\boldsymbol{s}^i, \boldsymbol{s}^j\right) \;=\; \left\langle \boldsymbol{\theta}\left(\boldsymbol{s}^i\right), \boldsymbol{\theta}\left(\boldsymbol{s}^j\right)\right\rangle.$$

The target values for the $\epsilon$-SVR are $y_i \;=\; k\left(\boldsymbol{s}^i, \boldsymbol{x}\right)$.

Finally we apply nonlinear support-vector regression to obtain $\hat{\boldsymbol{c}}$. The presence of dictionary atom $i$ in the mixture is indicated by $\hat{c}_i = C$ or $\hat{c}_i = -C$ (see Fig. 15.3). Optimization of the kernel parameters for an SVR-kernel may be used to obtain $\boldsymbol{\theta}$. However, $\boldsymbol{\theta}$ cannot be estimated explicitly, because it is represented only implicitly via its correspondence with a kernel function.

### 15.3.2 Nonlinear Transformation of Atomic Sources

Another generalization of the sparse-decomposition method is obtained by allowing each atomic source $\boldsymbol{s}^i$ to be explicitly transformed by $\boldsymbol{\theta}$—a mapping from $\mathbb{R}^L$ into $\mathbb{R}^L$—before the sources are linearly superimposed to produce the mixture. In contrast to the previous section, where $\boldsymbol{\theta}$ was implicitly specified via a kernel function, here we must specify $\boldsymbol{\theta}$ explicitly. Given a choice of $\boldsymbol{\theta}$, the atomic sources are first transformed by $\boldsymbol{\theta}$, and the linear theory is applied to the transformed sources.

With the transformation $\boldsymbol{\theta}$, (15.1) becomes

$$\boldsymbol{x} \;=\; \sum_{j=1}^{P} c_i\; \boldsymbol{\theta}\left(\boldsymbol{s}^i\right) \;=\; \boldsymbol{\theta}\left(\boldsymbol{S}\right)\; \boldsymbol{c}.$$

Multiplying the equation from left with $\boldsymbol{\theta}(\boldsymbol{s})^T$ gives

$$\boldsymbol{\theta}(\boldsymbol{s})^T \boldsymbol{x} \;=\; \boldsymbol{\theta}(\boldsymbol{s})^T \boldsymbol{\theta}(\boldsymbol{S})\;\; \boldsymbol{c}.$$

This equation is exactly the definition of the nonlinear $\epsilon$-SVR regression function, $h(\boldsymbol{S})$—see (15.8)—with $\boldsymbol{\theta}$ as the feature space mapping and the target

$$y_i \;=\; h(\boldsymbol{s}^i) = \;\boldsymbol{\theta}\left(\boldsymbol{s}^i\right)^T \boldsymbol{x}.$$

### 15.3.3 Nonlinear Transformed Atomic Sources: Inhomogeneous Case

In the previous section, we assumed that each atomic source $\boldsymbol{s}^i$ is transformed by the same function $\boldsymbol{\theta}$. Extending this *homogeneous* case, we now consider a more general *inhomogeneous* case in which each atomic source, $\boldsymbol{s}^i$, can be transformed by a distinct function, $f_i$ (it will become clear shortly why we use the notation $f_i$ rather than $\theta_i$), where the $\{f_i\}$ represent a family of functions parameterized by a single variable $\tau_i$, i.e., $f_i(\boldsymbol{s}) = g(\boldsymbol{s};\tau_i)$. For example, if $\boldsymbol{x}$ represents a time series, $g$ might shift the input in time by $\tau_i$ elements, and $f_i$ would then correspond to a particular shift.

In the situation we consider, we wish to extract the *invariant feature*, e.g., a representation of the time series independent of its particular shift. Suppose that we can determine a function of the input space, $\boldsymbol{\theta}(.)$, that is invariant under and linear in $f_i$, i.e.,

$$\boldsymbol{\theta}(a\; f_i(\boldsymbol{s})) \;\propto\; \boldsymbol{\theta}(\boldsymbol{s}).$$

For example, if $\boldsymbol{\theta}$ computes the power spectrum, it would be invariant under shifts in the signal and $q(a) = a$. This function $\boldsymbol{\theta}$ then serves to extract the invariant feature from the signal.

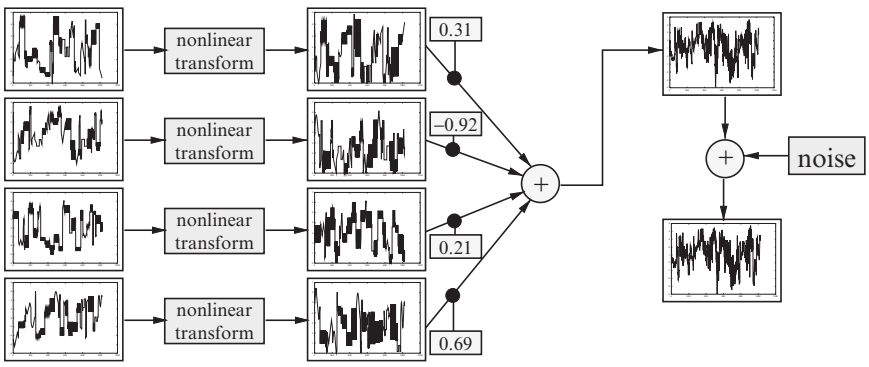


**Fig. 15.4.** The data generation process allowing for transformations of the atomic sources prior to mixing, in contrast with the original sparse-decomposition method (see Fig. 15.1).

Our goal is to approximate the feature of the mixture, $\boldsymbol{\theta}(\boldsymbol{x})$, in terms of a linear combination of the features of the atoms, i.e.,

$$\boldsymbol{\theta}(\boldsymbol{x}) \;=\; \sum_{i=1}^{P} t(c_i)\; \boldsymbol{\theta}\left(f_i\left(\boldsymbol{s}^i\right)\right) \;=\; \sum_{i=1}^{P} c_i\; \boldsymbol{\theta}\left(\boldsymbol{s}^i\right).$$

Under this formulation, the unknown transformations $f_i$ are eliminated from the approximation problem. $\hat{c_i}$ indicates whether mixture $\boldsymbol{x}$ and atom $\boldsymbol{s}^i$ share the same features or not: large $\hat{c_i}$ implies that $\boldsymbol{x}$ and $\boldsymbol{s}^i$ are mapped to similar (correlated) feature vectors.

The approach in this subsection is equivalent to the approach in Sect. 15.3.1 if we set

$$k\left(\boldsymbol{s}^i, \boldsymbol{s}^j\right) \;=\; \left\langle \boldsymbol{\theta}\left(\boldsymbol{s}^i\right), \boldsymbol{\theta}\left(\boldsymbol{s}^j\right)\right\rangle.$$

However, here $\boldsymbol{\theta}$ is designed for the $f_i$ and via $t(c_i)$, the magnitudes of the components $c_i$ can be computed. In contrast, in Sect. 15.3.1, the $c_i$ indicate only the presence of source atoms.

## 15.4 Experiments

### 15.4.1 Nonlinear Approximation of the Linear Correlations

We use a dictionary consisting of 64 atoms of length 128. The atoms vary in their frequencies and their shape (e.g., sinuidal, triangular, rectangular, and asymmetric triangular). On average, we chose 4.5 dictionary atoms to be present in one mixture. The nonzero coefficients $c_i$ are randomly chosen from $[0.1, 1.0] \cup [-1.0, -0.1]$. To generate training data, we then (1) added Gaussian noise with variance 1.0 to each mixture component, (2) added noise frequencies to the whole mixture (sinuidal with amplitude form $[0; 0.2]$), and (3) shifted the phase of each atom by a random amount chosen from a uniform distribution corresponding to a phase shift of 0–20% of the period. The values for hyperparameters $\epsilon$ and $C$ are chosen through a validation set. They are adjusted so that the average number of sources which are not recognized is below 0.5. In doing so, we impose an upper bound on the error that results from failing to detect sources that are present. To evaluate the performance, we count the number of sources which were wrongly detected.

The linear sparse-decomposition method leads to an error of 35.32. Table 15.1 shows the result for the nonlinear kernel $k(\boldsymbol{a}, \boldsymbol{b}) = (\gamma + \boldsymbol{a} \cdot \boldsymbol{b})^d$ with different values for $\gamma$ and $d$. The nonlinear kernel obtains results comparable to the linear sparse-decomposition method. This simulation demonstrates that nonlinear kernels work even for linear problems. We assume the reason for the good performance of the nonlinear kernels is that they are more robust against noise or against the specific noise which is present in the data if appropriate parameters are chosen.

**Table 15.1.** Average over 100 trials of wrongly detected atoms (false positives) for kernels $k(\boldsymbol{a}, \boldsymbol{b}) = (\gamma + \boldsymbol{a} \cdot \boldsymbol{b})^d$ (the linear result was 35.32).

| $d$ | $\gamma = 10^4$ | $\gamma = 10^5$ | $\gamma = 10^6$ | $\gamma = 10^7$ |
|---|---|---|---|---|
| 2 | 35.14 | 35.21 | 35.23 | 35.23 |
| 5 | 35.22 | 35.20 | 35.23 | 35.23 |
| 10 | 35.30 | 35.20 | 35.22 | 35.23 |
| 20 | 35.25 | 35.14 | 35.21 | 35.23 |
| 30 | 35.40 | 35.19 | 35.21 | 35.23 |

### 15.4.2 Transformed Atomic Sources

**Artificial Data** We consider a case where the mixture components have been corrupted by a class of componentwise transformations that produce nonnegative values: $[f(\boldsymbol{s})]_j = |s_j|^w$, where $w$ is a scalar that parameterizes $f$. We wish to extract an invariance that mitigates the effect of $w$ for any $w$. The *local variance* is a measure that achieves this goal. The local variance of $\boldsymbol{s}$ is defined as:

$$(2\ l\ +\ 1)^{-1} \sum_{t=j-l}^{j+l} (s_t - \bar{s}_j)^2,$$

where

$$\bar{s}_j := (2\ l\ +\ 1)^{-1} \sum_{t=j-l}^{j+l} s_t$$

and $l$ is a parameter that characterizes the neighborhood locality.

The local variance serves as an invariance extractor ($\boldsymbol{\theta}$). For $w = 1$ the invariance is exact, because the curve is only mirrored at the $y = 0$ axis. The mirroring is just a shift of the mean signal, a base-line-shift, which occurs for many measurement devices (it is a huge problem for EEG measurements). The variance as a central moment corrects the means to zero and eliminates base-line-shifts. However in one signal the base-line-shifts may occur multiple, therefore the variance should be taken locally. Also local frequency measures are invariant to base-line-shifts but the local variance combines the frequency of a whole frequency band.

For $w > 1$, the local variance is very robust against values of $w$, and serves to define a similarity measure (signals which have high local variance in the same position may be similar). For our tests, we used three different neighborhood sizes: $l = 8$ (referred to as AV1), $l = 10$ (AV2), $l = 20$ (AV3).

We generated 100 dictionary atoms of length 1024. To produce an atom we segmented the 1024 length vector into random segments of length between 1 and 64. Each segment consists of a scaled (from $[-0.8, -0.2] \cup [0.2, 0.8]$) periodic function from previous experiment. To each segment component a

**Table 15.2.** Best linear results out of 100 trials.

| Task | False negative | False positive |
|---|---|---|
| 1: $s_i^2$ | 0.94 | 9.20 |
| 2: $\|s_i\|$ | 1.03 | 20.97 |
| 3: $\|s_i\|^w$ | 1.00 | 20.02 |

**Table 15.3.** Average over 100 mixtures of the number of wrongly detected atoms (false positives) for the linear sparse-decomposition method ("linear") and three different methods measuring the local variance ("AV1"–"AV3"). "Failed" means that we were not able to push the average number of undetected sources (false negatives) below 0.4.

| Task | Linear | AV1 | AV2 | AV3 |
|---|---|---|---|---|
| 1: $s_i^2$ | Failed | 0.63 | 0.72 | 0.99 |
| 2: $\|s_i\|$ | Failed | 5.41 | 7.38 | Failed |
| 3: $\|s_i\|^w$ | Failed | 0.55 | 0.84 | 2.37 |

constant between $[-ac, +ac]$ is added. Figure 15.4 depicts the data generation process.

Task 1 uses $w = 1$, task 2 uses $w = 2$, and task 3 uses $|s_j|^w$ with $w$ randomly chosen from $[0.5; 2.0]$ as transformation. We set $ac = 5.0$ for task 1 and $ac = 0.5$ for task 2 and 3. The transformations are mixed as in previous experiment and Gaussian noise has $\sigma = 0.01$. As in the previous experiment, we ensure that the average number of undetected sources falls below a bound which is here set to 0.4. The best linear results are shown in Table 15.2 and the average results of our new method are shown in Table 15.3. The nonlinear mapping by the local variance formulas was able to extract the invariant and, therefore, could classify an atom as being present or not. The linear model failed at the task.

The results show that extracting invariant features (here the local variance) and measuring similarities between these invariant features instead of the original waveforms can help to robustly detect sources in a one-dimensional mixture. However prior knowledge is necessary to extract appropriate features.

**Speech Data** We considered transformations that shift the dictionary atoms, where each atom can have a different shift. As an invariant we use the power spectrum. The dictionary entries are 5 spoken words ("hid", "head", "had", "hud", and "hod") spoken by 20 different speakers, yielding a dictionary size of 100 atoms. The data was obtained from `areas/speech/database/hvd/` in the AI-Repository at `cs.cmu.edu`. The speech is sampled at 10 kHz.

**Table 15.4.** Average number of wrongly detected atoms for the linear sparse-decomposition method ("linear"), and three nonlinear transformation into the power spectrum ("PS1" to "PS3"). The error values are an average of 100 mixtures. "Failed" means that we were not able to push the missed atom signals below a threshold (see text for details).

| | Linear | PS1 | PS2 | PS3 |
|---|---|---|---|---|
| Task 1 | Failed | 1.82 | 1.72 | 1.50 |
| Task 2 | Failed | 5.06 | 4.82 | 5.10 |

We did not restrict the shifts of the atoms. The coefficients $c_i$ are chosen from $[0.2, 0.8]$. The power spectrum is obtained by using fast Fourier transformation with shifting Hanning window of size 256. The power spectrum is often used in speech processing and it is suited to define various "speech kernels". The lowest twenty frequencies were set to zero. The additive Gaussian noise had standard deviation of $\sigma = 0.05$ for task 1 (T1) and $\sigma = 0.2$ for task 2 (T2).

We compared three methods: PS1 is the power spectrum of the original mixture, PS2 is the power spectrum of the mixture where absolute mixture values smaller than 0.05 are set to zero, and PS3 is the power spectrum of the mixture where absolute mixture values smaller that 0.1 are set to zero. As in previous experiments we keep the average number of not detected atoms below a certain bound: 0.64 for T1 and 0.74 for T2.

The linear approach completely failed to solve the task. The results for the nonlinear transformation (power spectrum) are given in Table 15.4. Figure 15.5 shows an example of atomic source detection for PS3.

## 15.5 Conclusions

In this chapter, we reinterpreted the sparse-decomposition method for a single mixture as $\epsilon$-support vector regression ($\epsilon$-SVR). The $\epsilon$-SVR analogy supplied a new view on the sparse-decomposition method's hyperparameter and allowed us to introduce family of similar algorithms of which the sparse-decomposition method is one member. This family includes methods that allow for nonlinear transformations of the sources before they are mixed, and other nonlinear transformations in the process of identifying dictionary atoms in the mixture. One benefit of the nonlinearities incorporated into the approach is that they can be used to achieve some degree of invariance to irrelevant characteristics of signals. We demonstrated our approach with experiments using noisy single mixtures and speech datasets.

At a broader level, this chapter provided a means of mapping an unsupervised ICA problem into a supervised framework. To achieve this mapping, we introduced the notion of a dictionary of predefined atoms. In our approach, each unsupervised example is paired with each atom, and a fixed function

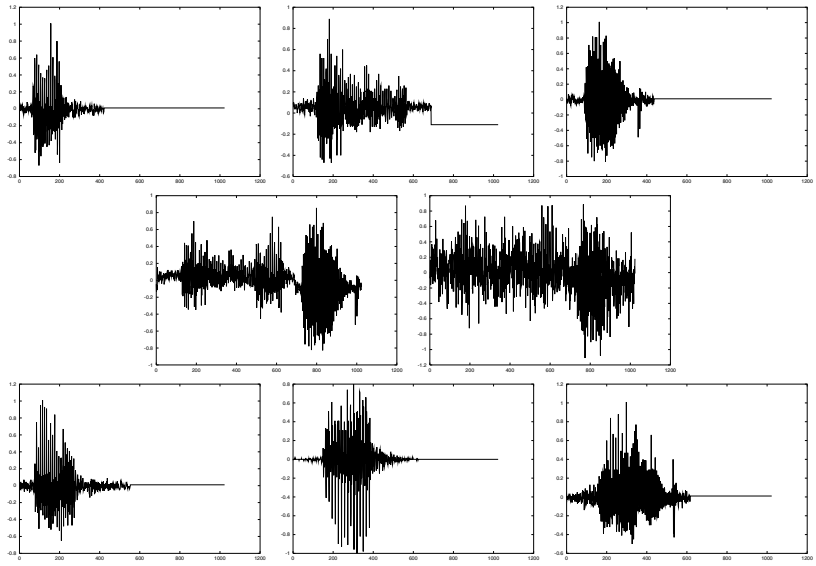

**Fig. 15.5.** Example for the method PS3. It detected 6 dictionary atoms in mixture: three were correctly and three wrongly detected. First line: three dictionary entries which are present (but shifted) in the mixture. Second line: (left) the mixture without noise and (right) the mixture. Third line: wrongly detected dictionary entries out of 100.

is used to generate a "target" value for the example. This target value can then be used within a supervised learning framework. Typically, the target value might be a measure of similarity between the dictionary atom and the example, such as a dot product. This mapping idea may well be useful in other unsupervised learning problems, where the target values from different unsupervised examples may be combined.

## References

1. M. Zibulevsky and B. A. Pearlmutter, "Blind source separation by sparse decomposition," *Neural Computation*, vol. 13, no. 4, pp. 863–882, 2001.
2. B. A. Pearlmutter and A. M. Zador, "Monaural source separation using spectral cues," in *Proc. of the Fifth International Conference on Independent Component Analysis and Blind Signal Separation*, C. G. Puntonet and A. Prieto, Eds. Springer Berlin/Heidelberg, 2004, pp. 478–485.
3. A. Cichocki, R. Unbehauen, L. Moszczynski, and E. Rummert, "A new on-line adaptive algorithm for blind separation of source signals," in *Proc. Int. Symposium on Artificial Neural Networks, ISANN-94*, 1994, pp. 406–411.
4. A. Hyvärinen, "Survey on independent component analysis," *Neural Computing Surveys*, vol. 2, pp. 94–128, 1999.

5. C. Jutten and J. Herault, "Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture," *Signal Processing*, vol. 24, no. 1, pp. 1–10, 1991.
6. A. J. Bell and T. J. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural Computation*, vol. 7, no. 6, pp. 1129–1159, 1995.
7. B. A. Pearlmutter and L. C. Parra, "Maximum likelihood blind source separation: A context-sensitive generalization of ICA," in *Advances in Neural Information Processing Systems 9*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds. MIT Press, Cambridge, MA, 1997, pp. 613–619.
8. H. Attias and C. E. Schreiner, "Blind source separation and deconvolution: The dynamic component analysis algorithm," *Neural Computation*, vol. 10, no. 6, pp. 1373–1424, 1998.
9. S. Amari, A. Cichocki, and H. Yang, "A new learning algorithm for blind signal separation," in *Advances in Neural Information Processing Systems 8*, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds. MIT Press, Cambridge, MA, 1996, pp. 757–763.
10. P. Comon, "Independent component analysis – a new concept?" *Signal Processing*, vol. 36, no. 3, pp. 287–314, 1994.
11. J.-F. Cardoso and A. Souloumiac, "Blind beamforming for non Gaussian signals," *IEE Proceedings-F*, vol. 140, no. 6, pp. 362–370, 1993.
12. T. Tanaka, "Analysis of bit error probability of direct-sequence CDMA multiuser demodulators," in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. MIT Press, Cambridge, MA, 2001, pp. 315–321.
13. G. Cauwenberghs, "Monaural separation of independent acoustical components," in *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems (ISCAS'99)*, vol. 5. IEEE, 1999, pp. 62–65.
14. T.-W. Lee, M. S. Lewicki, M. Girolami, and T. J. Sejnowski, "Blind source separation of more sources than mixtures using overcomplete representations," *IEEE Signal Processing Letters*, 1998.
15. S. T. Roweis, "One microphone source separation," in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. MIT Press, Cambridge, MA, 2001, pp. 793–799.
16. M. S. Lewicki and T. J. Sejnowski, "Learning overcomplete representations," *Neural Computation*, vol. 12, no. 2, pp. 337–365, 2000.
17. ——, "Learning nonlinear overcomplete representations for efficient coding," in *Advances in Neural Information Processing Systems 10*, M. I. Jordan, M. J. Kearns, and S. A. Solla, Eds. MIT Press, Cambridge, MA, 1998, pp. 556–562.
18. V. Vapnik, *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.
19. C. Cortes and V. N. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.
20. B. Schölkopf and A. J. Smola, *Learning with kernels – Support Vector Machines, Reglarization, Optimization, and Beyond*. MIT Press, Cambridge, 2002.
21. B. Schölkopf, P. L. Bartlett, A. J. Smola, and R. Williamson, "Support vector regression with automatic accuracy control," in *Proceedings of ICANN'98*, ser. Perspectives in Neural Computing, L. Niklasson, M. Bodén, and T. Ziemke, Eds. Berlin: Springer Verlag, 1998, pp. 111–116.

22. ——, "Shrinking the tube: a new support vector regression algorithm," in *Advances in Neural Information Processing Systems 11*, M. S. Kearns, S. A. Solla, and D. A. Cohn, Eds. Cambridge, MA: MIT Press, 1999, pp. 330–336.
23. A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, pp. 199–222, 2004. Also: NeuroCOLT Technical Report NC-TR-98-030.
24. R. Vollgraf, M. Scholz, I. Meinertzhagen, and K. Obermayer, "Nonlinear filtering of electron micrographs by means of support vector regression," in *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, Massachusetts, 2004, pp. 717–724.
25. X. Wang, A. Li, Z. Jiang, and H. Feng, "Missing value estimation for DNA microarray gene expression data by support vector regression imputation and orthogonal coding scheme," *BMC Bioinformatics*, vol. 7, p. 32, 2006.