LINEAR PROGRAM DIFFERENTIATION FOR SINGLE-CHANNEL SPEECH SEPARATION

Barak A. Pearlmutter[*]

Hamilton Institute
National University of Ireland Maynooth
Co. Kildare, Ireland

Rasmus K. Olsson[†]

Informatics and Mathematical Modelling
Technical University of Denmark
2800 Lyngby, Denmark

ABSTRACT

Many apparently difficult problems can be solved by reduction to linear programming. Such problems are often subproblems within larger systems. When gradient optimisation of the entire larger system is desired, it is necessary to propagate gradients through the internally-invoked LP solver. For instance, when an intermediate quantity $\mathbf{z}$ is the solution to a linear program involving constraint matrix $\mathbf{A}$, a vector of sensitivities $dE/d\mathbf{z}$ will induce sensitivities $dE/d\mathbf{A}$. Here we show how these can be efficiently calculated, when they exist. This allows algorithmic differentiation to be applied to algorithms that invoke linear programming solvers as subroutines, as is common when using sparse representations in signal processing. Here we apply it to gradient optimisation of overcomplete dictionaries for maximally sparse representations of a speech corpus. The dictionaries are employed in a single-channel speech separation task, leading to 5 dB and 8 dB target-to-interference ratio improvements for same-gender and opposite-gender mixtures, respectively. Furthermore, the dictionaries are successfully applied to a speaker identification task.

1. INTRODUCTION

Linear programming solvers (LP) are often used as subroutines within larger systems, in both operations research and machine learning [1, 2]. One very simple example of this is in sparse signal processing, where it is common to represent a vector as sparsely as possible in an overcomplete basis; this representation can be found using LP, and the sparse representation is then used in further processing [3–9].

To date, it has not been practical to perform end-to-end gradient optimisation of algorithms of this sort. This is due to the difficulty of propagating intermediate gradients (adjoints) through the LP solver. We show below how these adjoint calculations can be done: how a sensitivity of the output can be manipulated to give a sensitivity of the inputs. As usual in Automatic Differentiation (AD), these do not require much more computation than the original primal LP calculation—in fact, rather unusually, here they may require considerably less.

We first introduce our notational conventions for LP, and then give a highly condensed introduction to, and notation for, AD. We proceed to derive AD transformations for a simpler subroutine than LP: a linear equation solver. (This novel derivation is of independent interest, as linear equations are often constructed and solved within larger algorithms.) Armed with a general AD transformation for linear equation solvers along with suitable notation, we find the AD transformations for linear program solvers simple to derive. This is applied mechanically to yield AD rules for a linearly-constrained L_1-optimiser.

The problem of finding an overcomplete signal dictionary tuned to a given stimulus ensemble, so that signals drawn from that ensemble will have sparse representations in the constructed dictionary, has received increasing attention, due to applications in both neuroscience and in the construction of efficient practical codes [10]. Here we derive a gradient method for such an optimisation, and apply it to learn a sparse representation of speech.

Single-channel speech separation, where the objective is to estimate the speech sources of the mixture, is a relevant task in hearing aids, as a speech recognition pre-processor, and in other applications which might benefit from better noise reduction. For this reason, there has been a flurry of interest in the problem [9, 11–17]. We encode the audio mixtures in the basis functions of the combined personalised dictionaries, which were adapted using the devised gradient method. The sparse code separates the signal into its sources, and reconstruction follows. Furthermore, we show that the dictionaries are truly personal, meaning that a given dictionary provides the sparsest fit for the particular speaker, which it was adapted to. Hence, we are able to correctly classify speech signals to their speaker.

[*]Supported by Science Foundation Ireland grant 00/PI.1/C067 and the Higher Education Authority of Ireland.

[†]Thanks to Oticon Fonden for financial support for this work.

2. BACKGROUND AND NOTATION

We develop a convenient notation while briefly reviewing the essentials of linear programming (LP) and algorithmic differentiation (AD).

2.1. Linear Programming

In order to develop a notation for LP, consider the general LP problem

$$\arg\min_{\mathbf{z}} \ \mathbf{w}^\top \mathbf{z} \text{ s.t. } \mathbf{A}\mathbf{z} \le \mathbf{a} \text{ and } \mathbf{B}\mathbf{z} = \mathbf{b} \qquad (1)$$

We will denote the linear program solver lp, and write the solution as $\mathbf{z} = \mathsf{lp}(\mathbf{w}, \mathbf{A}, \mathbf{a}, \mathbf{B}, \mathbf{b})$. It is important to see that $\mathsf{lp}(\cdot)$ can be regarded as either a mathematical function which maps LP problems to their solutions, or as a computer program which actually solves LP problems. Our notation deliberately does not distinguish between these two closely related notions.

Assuming feasibility, boundedness, and uniqueness, the solution to this LP problem will satisfy a set of linear equalities consisting of a subset of the constraints: the *active* constraints [18–20]. An LP solver calculates two pieces of information: the solution itself, and the identity of the active constraints. We will find it convenient to refer to the active constraints by defining some very sparse matrices that extract the active constraints from the constraint matrices. Let $\alpha_1 < \cdots < \alpha_n$ be the indices of the rows of $\mathbf{A}$ corresponding to active constraints, and $\beta_1 < \cdots < \beta_m$ index the active rows of $\mathbf{B}$. Without loss of generality, we assume that the total number of active constraints is equal the dimensionality of the solution, $n + m = \dim \mathbf{z}$. We let $\mathbf{P}_\alpha$ be a matrix with n rows, where the i-th row is all zeros except for a one in the α_i-th column, and $\mathbf{P}_\beta$ similarly have m rows, with its i-th row all zeros except for a one in the β_i-th column. So $\mathbf{P}_\alpha \mathbf{A}$ and $\mathbf{P}_\beta \mathbf{B}$ hold the active rows of $\mathbf{A}$ and $\mathbf{B}$, respectively. These can be combined into a single matrix,

$$\mathbf{P} \equiv \begin{bmatrix} \mathbf{P}_\alpha & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_\beta \end{bmatrix}$$

Using these definitions, the solution $\mathbf{z}$ to (1), which presumably is already available having been computed by the algorithm that identified the active constraints, must be the unique solution of the system of linear constraints

$$\mathbf{P} \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} \mathbf{z} = \mathbf{P} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}$$

or

$$\mathsf{lp}(\mathbf{w}, \mathbf{A}, \mathbf{a}, \mathbf{B}, \mathbf{b}) = \mathsf{lq}(\mathbf{P} \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}, \mathbf{P} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}) \qquad (2)$$

where lq is a routine that efficiently solves a system of linear equations, $\mathsf{lq}(\mathbf{M}, \mathbf{m}) = \mathbf{M}^{-1}\mathbf{m}$. For notational convenience we suppress the identity of the active constraints as an output of the lp routine. Instead we assume that it is available where necessary, so any function with access to the solution $\mathbf{z}$ found by the LP solver is also assumed to have access to the corresponding $\mathbf{P}$.

2.2. Algorithmic Differentiation

AD is a process by which a numeric calculation specified in a computer programming language can be mechanically transformed so as to calculate derivatives (in the differential calculus sense) of the function originally calculated [21]. There are two sorts of AD transformations: forward accumulation [22] and reverse accumulation [23]. (A special case of reverse accumulation AD is referred to as backpropagation in the machine learning literature [24].) If the entire calculation is denoted $\mathbf{y} = h(\mathbf{x})$, then forward accumulation AD arises because a perturbation $d\mathbf{x}/dr$ induces a perturbation $d\mathbf{y}/dr$, and reverse accumulation AD arises because a gradient $dE/d\mathbf{y}$ induces a gradient $dE/d\mathbf{x}$. The Jacobian matrix plays a dominant role in reasoning about this process. This is the matrix $\mathbf{J}$ whose i, j-th entry is dh_i/dx_j. Forward AD calculates $\acute{\mathbf{y}} = \mathbf{J}\acute{\mathbf{x}} = \overrightarrow{h}(\mathbf{x}, \acute{\mathbf{x}})$, and reverse AD calculates $\grave{\mathbf{x}} = \mathbf{J}^\top \grave{\mathbf{y}} = \overleftarrow{h}(\mathbf{x}, \grave{\mathbf{y}})$. The difficulty is that, in high dimensional systems, the matrix $\mathbf{J}$ is too large to actually calculate. In AD the above matrix-vector products are found directly and efficiently, without actually calculating the Jacobian.

The central insight is that calculations can be broken down into a chained series of assignments $v := g(u)$, and transformed versions of these chained together. The transformed version of the above internal assignment statement would be $\acute{v} := \overrightarrow{g}(u, \acute{u}, v)$ in forward mode [22], or $\grave{u} := \overleftarrow{g}(u, v, \grave{v})$ in reverse mode [23]. The most interesting property of AD, which results from this insight, is that the time consumed by the adjoint calculations can be the same as that consumed by the original calculation, up to a small constant factor. (This naturally assumes that the transformations of the primitives invoked also obey this property, which is in general true.)

We will refer to the adjoints of original variables introduced in forward accumulation (perturbations) using a forward-leaning accent $v \mapsto \acute{v}$; to the adjoint variables introduced in the reverse mode transformation (sensitivities) using a reverse-leaning accent $v \mapsto \grave{v}$; and to the forward- and reverse-mode transformations of functions using forward and reverse arrows, $h \mapsto \overrightarrow{h}$ and $h \mapsto \overleftarrow{h}$. A detailed introduction to AD is beyond the scope of this paper, but one form appears repeatedly in our derivations. This is $\mathbf{V} := \mathbf{A}\mathbf{U}\mathbf{B}$ where $\mathbf{A}$ and $\mathbf{B}$ are constant matrices and $\mathbf{U}$ and $\mathbf{V}$ are matrices as well. This transforms to $\acute{\mathbf{V}} := \mathbf{A}\acute{\mathbf{U}}\mathbf{B}$ and $\grave{\mathbf{U}} := \mathbf{A}^\top \grave{\mathbf{V}} \mathbf{B}^\top$.

2.3. AD of a Lin. Eq. Solver

We first derive AD equations for a simple implicit function, namely a linear equation solver. We consider a subroutine lq which finds the solution $\mathbf{z}$ of $\mathbf{M}\mathbf{z} = \mathbf{m}$, written $\mathbf{z} = \mathsf{lq}(\mathbf{M}, \mathbf{m})$. This assumes that $\mathbf{M}$ is square and full-rank, just as a division operation $z = x/y$ assumes that $y \neq 0$. We will derive formulae for both forward mode AD (the $\acute{\mathbf{z}}$ induced by $\acute{\mathbf{M}}$ and $\acute{\mathbf{m}}$) and reverse mode AD (the $\grave{\mathbf{M}}$ and $\grave{\mathbf{m}}$ induced by $\grave{\mathbf{z}}$).

For forward propagation of perturbations, we will write $\acute{\mathbf{z}} = \overrightarrow{\mathsf{lq}}(\mathbf{M}, \acute{\mathbf{M}}, \mathbf{m}, \acute{\mathbf{m}}, \mathbf{z})$. Because $(\mathbf{M} + \acute{\mathbf{M}})(\mathbf{z} + \acute{\mathbf{z}}) = \mathbf{m} + \acute{\mathbf{m}}$ which reduces to $\mathbf{M}\acute{\mathbf{z}} = \acute{\mathbf{m}} - \acute{\mathbf{M}}\mathbf{z}$, we conclude that

$$\overrightarrow{\mathsf{lq}}(\mathbf{M}, \acute{\mathbf{M}}, \mathbf{m}, \acute{\mathbf{m}}, \mathbf{z}) = \mathsf{lq}(\mathbf{M}, \acute{\mathbf{m}} - \acute{\mathbf{M}}\mathbf{z}).$$

Note that lq is linear in its second argument, where the perturbations enter linearly. For reverse propagation of sensitivities, we will write

$$\begin{bmatrix}\grave{\mathbf{M}} & \grave{\mathbf{m}}\end{bmatrix} = \overleftarrow{\mathsf{lq}}(\mathbf{M}, \mathbf{m}, \mathbf{z}, \grave{\mathbf{z}}). \tag{3}$$

First observe that $\mathbf{z} = \mathbf{M}^{-1}\mathbf{m}$ and hence $\grave{\mathbf{m}} = \mathbf{M}^{-\top}\grave{\mathbf{z}}$ so

$$\grave{\mathbf{m}} = \mathsf{lq}(\mathbf{M}^\top, \grave{\mathbf{z}}).$$

For the remaining term we start with our previous forward perturbation $\acute{\mathbf{M}} \mapsto \acute{\mathbf{z}}$, namely $\acute{\mathbf{z}} = -\mathbf{M}^{-1}\acute{\mathbf{M}}\mathbf{z}$, and note that the reverse must be the transpose of this linear relationship, $\grave{\mathbf{M}} = -\mathbf{M}^{-\top}\grave{\mathbf{z}}\mathbf{z}^\top$, which is the outer product

$$\grave{\mathbf{M}} = -\grave{\mathbf{m}}\mathbf{z}^\top.$$

2.4. AD of Linear Programming

We apply equation (3) followed by some bookkeeping, yields

$$\begin{aligned}\begin{bmatrix}\grave{\mathbf{A}} & \grave{\mathbf{a}}\\ \grave{\mathbf{B}} & \grave{\mathbf{b}}\end{bmatrix} &= \overleftarrow{\mathsf{lp}}(\mathbf{w}, \mathbf{A}, \mathbf{a}, \mathbf{B}, \mathbf{b}, \mathbf{z}, \grave{\mathbf{z}})\\ &= \mathbf{P}^\top \overleftarrow{\mathsf{lq}}(\mathbf{P}\begin{bmatrix}\mathbf{A}\\ \mathbf{B}\end{bmatrix}, \mathbf{P}\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix}, \mathbf{z}, \grave{\mathbf{z}})\\ \grave{\mathbf{w}} &= \mathbf{0}\end{aligned}$$

Forward accumulation is similar, but is left out for brevity.

2.5. Constrained L_1 Optimisation

We can find AD equations for linearly constrained L_1-norm optimisation via reduction to LP. Consider

$$\arg\min_{\mathbf{c}} \|\mathbf{c}\|_1 \text{ s.t. } \mathbf{D}\mathbf{c} = \mathbf{y}.$$

Although $\|\mathbf{c}\|_1 = \sum_i |c_i|$ is a nonlinear objective function, a change in parametrisation allows optimisation via LP. We name the solution $\mathbf{c} = \mathsf{L1opt}(\mathbf{y}, \mathbf{D})$ where

$$\mathsf{L1opt}(\mathbf{y}, \mathbf{D}) = \begin{bmatrix}\mathbf{I} & -\mathbf{I}\end{bmatrix} \mathsf{lp}(\mathbf{1}, -\mathbf{I}, \mathbf{0}, \mathbf{D}\begin{bmatrix}\mathbf{I} & -\mathbf{I}\end{bmatrix}, \mathbf{y})$$

in which $\mathbf{0}$ and $\mathbf{1}$ denote column vectors whose elements all contain the indicated number, and each $\mathbf{I}$ is an appropriately sized identity matrix. The reverse-mode AD transformation follows immediately,

$$\begin{aligned}&\overleftarrow{\mathsf{L1opt}}(\mathbf{y}, \mathbf{D}, \mathbf{c}, \grave{\mathbf{c}}) = \begin{bmatrix}\grave{\mathbf{D}} & \grave{\mathbf{y}}\end{bmatrix} =\\ &\begin{bmatrix}\mathbf{0}' & \mathbf{I}\end{bmatrix} \overleftarrow{\mathsf{lp}}(\mathbf{1}, -\mathbf{I}, \mathbf{0}, \mathbf{D}\begin{bmatrix}\mathbf{I} & -\mathbf{I}\end{bmatrix}, \mathbf{y}, \mathbf{z}, \begin{bmatrix}\mathbf{I}\\ -\mathbf{I}\end{bmatrix}\grave{\mathbf{c}}) \begin{bmatrix}\mathbf{I} & \mathbf{0}\\ -\mathbf{I} & \mathbf{0}\\ \mathbf{0}^\top & 1\end{bmatrix}\end{aligned}$$

where $\mathbf{z}$ is the solution of the internal LP problem and $\mathbf{0}'$ is an appropriately sized matrix of zeros.

3. DICTIONARIES OPTIMISED FOR SPARSITY

A major advantage of the LP differentiation framework, and more specifically the reverse accumulation of the constrained L_1 norm optimisation, is that it provides directly a learning rule for learning sparse representation in overcomplete dictionaries.

We assume an overcomplete dictionary in the columns of $\mathbf{D}$, which is used to encode a signal represented in the column vector $\mathbf{y}$ using the column vector of coefficients $\mathbf{c} = \mathsf{L1opt}(\mathbf{y}, \mathbf{D})$ where each dictionary element has unit L_2 length. A probabilistic interpretation of the encoding as a maximum posterior (MAP) estimate naturally follows from two assumptions: a Laplacian prior $\mathrm{p}(\mathbf{c})$, and a noise-free observation model $\mathbf{y} = \mathbf{D}\mathbf{c}$. This gives

$$\mathbf{c} = \arg\max_{\mathbf{c}'} \mathrm{p}(\mathbf{c}'|\mathbf{y}, \mathbf{D})$$

We would like to improve $\mathbf{D}$ for a particular distribution of signals, meaning change $\mathbf{D}$ so as to maximise the sparseness of the codes assigned. With $\mathbf{y}$ drawn from this distribution, an ideal dictionary will minimise the average code length, giving maximally sparse coefficients. We will update $\mathbf{D}$ so as to minimise $E = \langle\|\mathsf{L1opt}(\mathbf{y}, \mathbf{D})\|_1\rangle$ while keeping the columns of $\mathbf{D}$ at unit length. This can be regarded a special case of Independent Component Analysis [25], where measures of independence across coefficients are optimised. We wish to use a gradient method so we calculate $\nabla_{\mathbf{D}} E_{\mathbf{y}}$ where $E_{\mathbf{y}} = \|\mathsf{L1opt}(\mathbf{y}, \mathbf{D})\|_1$ making $E = \langle E_{\mathbf{y}}\rangle$. Invoking AD,

$$\begin{aligned}\nabla_{\mathbf{D}} E_{\mathbf{y}} = \grave{\mathbf{D}} &= \begin{bmatrix}\grave{\mathbf{D}} & \grave{\mathbf{y}}\end{bmatrix}\begin{bmatrix}\mathbf{I}\\ \mathbf{0}^\top\end{bmatrix}\\ &= \overleftarrow{\mathsf{L1opt}}(\mathbf{y}, \mathbf{D}, \mathbf{c}, \operatorname{sign}(\mathbf{c}))\begin{bmatrix}\mathbf{I}\\ \mathbf{0}^\top\end{bmatrix}\end{aligned} \tag{4}$$

where $\operatorname{sign}(x) = +1/0/-1$ for x positive/zero/negative, and applies elementwise to vectors.

We are now in a position to perform stochastic gradient optimisation [26], modified by the inclusion of a normalisation step to maintain the columns of $\mathbf{D}$ at unit length and non-negative.

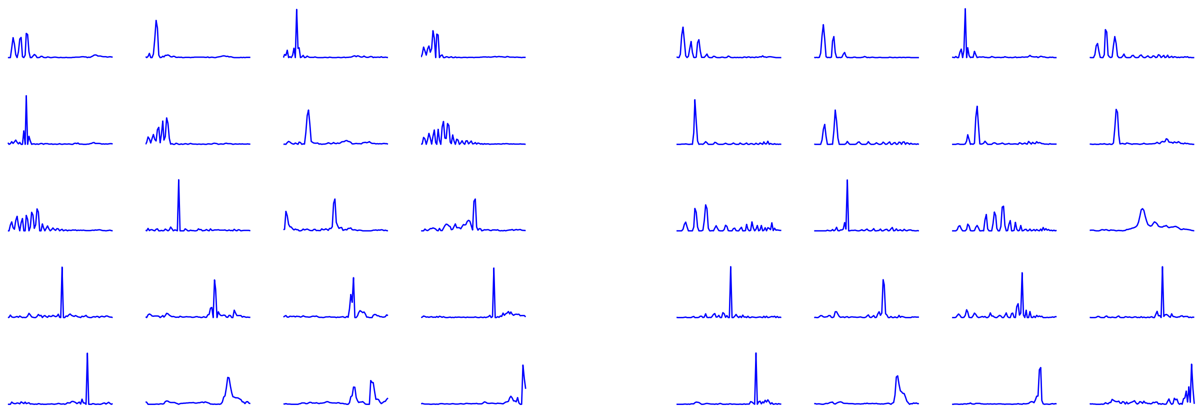

Fig. 1. A sample of learnt dictionary entries for male (left) and female (right) speech in the Mel spectrum domain. Clearly, harmonic features emerge from the data but some broad and narrow noise spectra can also be seen. The dictionaries were initialised to $N = 256$ delta-like pulses, length $L = 80$ and were adopted from $T = 420\,\mathrm{s}$ of speech.

1. Draw $\mathbf{y}$ from signal distribution.
2. Calculate $E_{\mathbf{y}}$.
3. Calculate $\nabla_{\mathbf{D}} E_{\mathbf{y}}$ by (4).
4. Step $\mathbf{D} := \mathbf{D} - \eta \nabla_{\mathbf{D}} E_{\mathbf{y}}$.
5. Set any negative element of $\mathbf{D}$ to zero.
6. Normalise the columns $\mathbf{d}_i$ of $\mathbf{D}$ to unit L_2 norm.
7. Repeat to convergence of $\mathbf{D}$.

This procedure can be regarded as a very efficient exact maximum likelihood treatment of the posterior integrated using a Gaussian approximation [7]. However, the formulation here can be easily and mechanically generalised to other objectives.

A set of personalised speech dictionaries were learnt by sparsity optimisation in the Grid Corpus [27] which is available at http://www.dcs.shef.ac.uk/spandh/gridcorpus. This corpus contains 1000×34 utterances of 34 speakers, confined to a limited vocabulary. The speech was preprocessed and represented to (essentially) transform the audio signals into a Mel time-frequency representation, as presented and discussed by Ellis and Weiss [14]. The data was downsampled to $8\,\mathrm{kHz}$ and high-pass filtered to bias our objective towards more accuracy in the high-end of the spectrum. The short-time Fourier transform was computed from windowed data vectors of length $32\,\mathrm{ms}$, corresponding to $K = 256$ samples, and subsequently mapped into $L = 80$ bands on the Mel scale. From $T = 420\,\mathrm{s}$ of audio from each speaker, the non-zero time-frames were extracted for training and normalised to unity L2 norm. The remainder of the audio ($> 420\,\mathrm{s}$) was set aside for testing. The stochastic gradient optimisation of the linearly constrained L_1 norm was run for 40,000 iterations. The step-size η was decreased throughout the training. The $N = 256$ columns of the dictionaries were initialised with narrow pulses distributed evenly across the spectrum and non-negativity was enforced following each iteration. In Figure 1 is displayed a randomly selected sample of learnt dictionary elements of one male and one female speaker. The dictionaries clearly capture a number of characteristics of speech, such as quasi-periodicity and dependencies across frequency bands.

3.1. Source Separation

This work was motivated by a particular application: single-channel source separation.[1] The aim is to recover R source signals from a one-dimensional mixture signal. In that context, an important technique is to perform a linearly constrained L_1-norm optimisation in order to fit an observed signal using a sparse subset of coefficients over an overcomplete signal dictionary. A single column of the mixture spectrogram is the sum of the source spectra: $\mathbf{y} = \sum_i^R \mathbf{y}_i$. In the interest of simplicity, this model assumes a $0\,\mathrm{dB}$ target-to-masker ratio (TMR). Generalization to general TMR by the inclusion of weighting coefficients is straightforward.

As a generative signal model, it is assumed that $\mathbf{y}_i$ can be represented sparsely in the overcomplete dictionary $\mathbf{D}$, which is the concatenation of the source dictionaries: $\mathbf{D} = \begin{bmatrix} \mathbf{D}_1 & \ldots & \mathbf{D}_i & \ldots & \mathbf{D}_R \end{bmatrix}$. Assuming that the $\mathbf{D}_i$ are different in some sense, it can be expected that a sparse representation in the overcomplete basis $\mathbf{D}$ coincides with the separation of the sources, *i.e.* we compute

$$\mathbf{c} = \begin{bmatrix} \mathbf{c}_1^\top & \ldots & \mathbf{c}_i^\top & \ldots & \mathbf{c}_R^\top \end{bmatrix}^\top = \texttt{L1opt}(\mathbf{y}, \mathbf{D})$$

where the $\mathbf{c}_i$ are the coefficients pertaining to the ith source. The source estimates in the Mel spectrum domain are then re-synthesised as $\hat{\mathbf{y}}_i = \mathbf{D}_i \mathbf{c}_i$. The conversion back to the time-domain consists of mapping to the amplitude spectro-

[1]The INTERSPEECH 2006 conference hosts a special session on this issue, based on the GRID speech corpus. See www.dcs.shef.ac.uk/~martin/SpeechSeparationChallenge.htm.

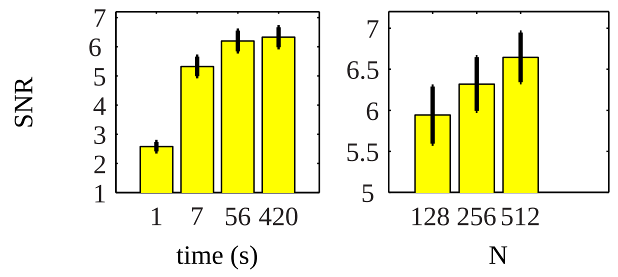

Fig. 2. Dependency of the separation performance measured as signal-to-noise ratio (SNR) as a function of the data volume (left), and, the dictionary size, N (right). Only $T = 7\,\mathrm{s}$ of speech is needed to attain near-optimal performance. The performance increases about 0.5 dB per doubling of N.

Genders	SNR (dB)
M/M	4.9 ± 1.2
M/F	7.8 ± 1.3
F/F	5.1 ± 1.4

Table 1. Monaural two-speaker signal-to-noise separation performance (mean±stderr of SNR), by speaker gender. The simulated test data consisted of all possible combinations, $T = 6\,\mathrm{s}$, of the 34 speakers.

gram and subsequently reconstructing the time-domain signal using the noisy phase of the mixture. Due to the sparsity of speech in the transformed domain, the degree of overlap of the sources is small, which causes the approximation to be fairly accurate. Useful software in this connection is available at http://www.ee.columbia.edu/~dpwe/. In the following, the quality of $\hat{\mathbf{y}}_i$ are evaluated in the time-domain simply as the ratio of powers of the target to reconstruction error, henceforth termed the signal-to-noise ratio (SNR).

In order to assess the convergence properties of the algorithm, the SNR was computed as a function of the amount of training data, see figure 2. It was found that useful results could be achieved with a few seconds of training data, whereas optimal performance was only obtained after a few minutes. It was furthermore investigated how the SNR varies as a function of the number of dictionary elements, N. Each doubling of N brings an improvement, indicating the potential usefulness of increased computing power. The above results were obtained by simulating all possible mixtures of 8 speakers (4 male, 4 female) at $0\,\mathrm{dB}$ and computing the SNR's on $6\,\mathrm{s}$ segments. Performance figures were computed on the complete data set of 34 speakers, amounting to 595 combinations, with $N = 256$ and $T = 420\,\mathrm{s}$; see Table 1. The test data is available at www2.imm.dtu.dk/~rko/single_channel.

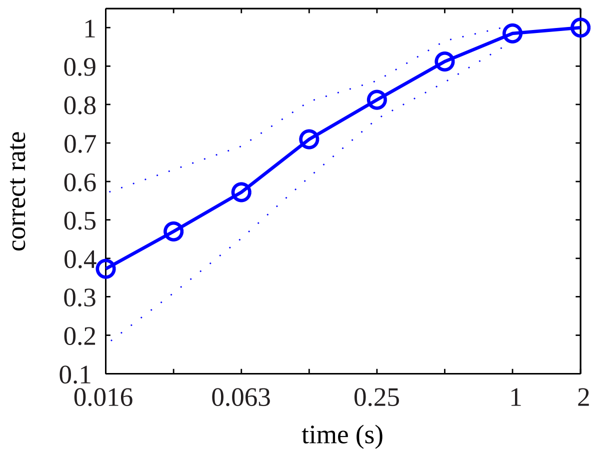

Fig. 3. The maximum-likelihood correct-classification rate as computed in a $T = 2\,\mathrm{s}$ test window on all combination of the 34 speakers and 34 dictionaries. If all time-frames are included into computation, the classification is perfect, but the performance decreases as smaller windows are used.

3.2. Speaker identification

In many potential applications of source separation the speakers of the mixture would be novel, and have to be estimated from the audio stream. In order to perform truly *blind* separation, the system should be able to automatically apply the appropriate dictionaries. Here we attack a simpler subproblem: speaker identification in an audio signal with only a single speaker. Our approach is straightforward: select the dictionary that yields the sparsest code for the signal. Again, this can be interpreted as maximum-likelihood classification. Figure 3 displays the percentage of correctly classified sound snippets. The figures were computed on all combinations of speakers and dictionaries, that is $34 \times 34 = 1156$ combinations. The complete data $(2\,\mathrm{s})$ resulted in all speakers being correctly identified. Shorter windows carried a higher error rate. For the described classification framework to be successful in a source separation task, it is required that each speaker appears exclusively in parts of the audio signal. This is not at all unrealistic in normal conversation, depending on the politeness of the speakers.

4. CONCLUSION AND OUTLOOK

Linear programming is often viewed as a black-box solver, which cannot be fruitfully combined with gradient-based optimisation methods. As we have seen, this is not the case. LP can be used as a subroutine in a larger system, and perturbations can be propagated forwards and sensitivities propagated backwards through the LP solver. The only caution is that LP is by nature only piecewise differentiable, so care must be taken with regard to crossing through such discontinuities.

The figures carry evidence that the adapted Mel scale dictionaries to a large extent perform the job, and that the generalisation of the results to spontaneous speech depends to a large extent on designing a sensible scheme for providing the algorithm with a balanced training data. Furthermore, the system should be able to manage some difficult aspects of real-room conditions, in particular those in which the observed signal is altered by the room dynamics. We feel that a possible solution could build on the principles laid out in previous work [9], where a head-related transfer function (HRTF) is used to provide additional contrast between the sources.

We found that using spectrogram patches rather than power spectra improved the results only marginally, in agreement with previous reports using a related approach [16].

References

[1] O. L. Mangasarian, W. N. Street, and W. H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, July-Aug. 1995.

[2] P. S. Bradley, O. L. Mangasarian, and W. N. Street. Clustering via concave minimization. In *Adv. in Neu. Info. Proc. Sys. 9*, pages 368–374. MIT Press, 1997.

[3] I. F. Gorodnitsky and B. D. Rao. Sparse signal reconstruction from limited data using FOCUSS: A reweighted minimum norm algorithm. *IEEE Trans. Signal Proccessing*, 45(3):600–616, 1997.

[4] M. Lewicki and B. A. Olshausen. Inferring sparse, overcomplete image codes using an efficient coding framework. In *Advances in Neural Information Processing Systems 10*, pages 815–821. MIT Press, 1998.

[5] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

[6] 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*, 4(5):87–90, 1999.

[7] M. S. Lewicki and T. J. Sejnowski. Learning overcomplete representations. *Neu. Comp.*, 12(2):337–65, 2000.

[8] M. Zibulevsky and B. A. Pearlmutter. Blind source separation by sparse decomposition in a signal dictionary. *Neu. Comp.*, 13(4):863–882, Apr. 2001.

[9] B. A. Pearlmutter and A. M. Zador. Monaural source separation using spectral cues. In *ICA*, pages 478–485, 2004.

[10] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T.-W. Lee, and T. J. Sejnowski. Dictionary learning algorithms for sparse representation. *Neu. Comp.*, 15 (2):349–396, 2003.

[11] S. T. Roweis. One microphone source separation. In *Adv. in Neu. Info. Proc. Sys. 13*, pages 793–799. MIT Press, 2001.

[12] G.-J. Jang and T.-W. Lee. A maximum likelihood approach to single-channel source separation. *J. of Mach. Learn. Research*, 4:1365–1392, Dec. 2003.

[13] M. N. Schmidt and M. Mørup. Nonnegative matrix factor 2-D deconvolution for blind single channel source separation. In *ICA*, pages 123–123, 2006.

[14] D. P. W. Ellis and R. J. Weiss. Model-based monaural source separation using a vector-quantized phase-vocoder representation. In *ICASSP*, 2006.

[15] M. N. Schmidt and R. K. Olsson. Single-channel speech separation using sparse non-negative matrix factorization. In *Interspeech*, 2006, submitted.

[16] S. T. Roweis. Factorial models and refiltering for speech separation and denoising. In *Eurospeech*, pages 1009–1012, 2003.

[17] F. Bach and M. I. Jordan. Blind one-microphone speech separation: A spectral learning approach. In *Advances in Neural Information Processing Systems 17*, pages 65–72, 2005.

[18] G. B. Dantzig. Programming in a linear structure. USAF, Washington D.C., 1948.

[19] S. I. Gass. *An Illustrated Guide to Linear Programming*. McGraw-Hill, 1970.

[20] R. Dorfman. The discovery of linear programming. *Annals of the History of Computing*, 6(3):283–295, July–Sep. 1984.

[21] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Number 19 in Frontiers in Appl. Math. SIAM, Philadelphia, PA, 2000. ISBN 0–89871–451–6.

[22] R. E. Wengert. A simple automatic derivative evaluation program. *Commun. ACM*, 7(8):463–464, 1964.

[23] B. Speelpenning. *Compiling Fast Partial Derivatives of Functions Given by Algorithms*. PhD thesis, Department of Computer Science, University of Illinois, Urbana-Champaign, Jan. 1980.

[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back–propagating errors. *Nature*, 323:533–536, 1986.

[25] A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neu. Comp.*, 7(6):1129–1159, 1995.

[26] H. Robbins and S. Monro. A stochastic approximation method. *Ann. Mat. Stats.*, 22:400–407, 1951.

[27] M. P. Cooke, J. Barker, S. P. Cunningham, and X. Shao. An audio-visual corpus for speech perception and automatic speech recognition, 2005. Submitted.