The sparse decomposition of sound in the time domain using non-negative quadratic programming.

Conor Houghton

***Abstract*— Non-negative matrix deconvolution and sparse decomposition are useful tools for source separation and acoustic object recognition. Here, a new algorithm for calculating a sparse decomposition of sound in the time domain is derived using non-negative quadratic programming.**

I. Introduction

A signal is said to be sparse if its distribution has a sharp peak and a fat tail: a sparse signal has both small and large values more often than a Gaussian distributed signal with the same variance. Recently, it has been realized that it can be useful to decompose sound over a sparse basis, that is, a basis whose components are sparse. Because different sparse components can often be treated as if they were different channels it is easier to separate sources and perform acoustic object recognition on the sparse coded signal [1], [2], [3], [4], [5]. Here, an algorithm for finding sparse representations of sound is presented. It is based on non-negative matrix factorization and non-negative matrix deconvolution [6], [7]; however, while non-negative matrix deconvolution is a decomposition of the spectrographic representation of sound, the algorithm presented here uses a recent result on non-negative quadratic programming (NQP) [8], [9], [10], to decompose sound in the time domain.

In [8], [9], [10], Sha and coworkers consider the basic problem of quadratic programming with non-negative constraints, minimizing the potential

$$F(\mathbf{v}) = \frac{1}{2}\mathbf{v}^T\mathbf{A}\mathbf{v} + \mathbf{b}^T\mathbf{v} \tag{1}$$

subject to $\mathbf{v} \geq \mathbf{0}$. They discover an elegant multiplicative update rule which converges monotonically. There is no non-negativity constraint on $\mathbf{A}$: non-negative matrices $\mathbf{A}^+$ and $\mathbf{A}^-$ consisting of the positive and negative elements are defined by

$$A^{\pm}_{ij} = \begin{cases} \pm A_{ij} & \pm A_{ij} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

so that $\mathbf{A} = \mathbf{A}^+ - \mathbf{A}^-$. Now, the potential is decomposed into three parts:

$$\begin{aligned} F_a(\mathbf{v}) &= \frac{1}{2}\mathbf{v}^T\mathbf{A}^+\mathbf{v} \\ F_b(\mathbf{v}) &= \mathbf{b}^T\mathbf{v} \\ F_c(\mathbf{v}) &= \frac{1}{2}\mathbf{v}^T\mathbf{A}^-\mathbf{v} \end{aligned} \tag{3}$$

so $F = F_a + F_b - F_c$. Three gradients are defined

$$\begin{aligned} a_i &= \frac{\partial F_a(\mathbf{v})}{\partial v_i} = A^+_{i\bar{j}}v_{\bar{j}} \\ b_i &= \frac{\partial F_b(\mathbf{v})}{\partial v_i} = b_i \\ c_i &= \frac{\partial F_c(\mathbf{v})}{\partial v_i} = A^-_{i\bar{j}}v_{\bar{j}} \end{aligned} \tag{4}$$

where, for clarity, we have used a summation convention in which repeated over-lined indices are summed so that

$$A^+_{i\bar{j}}v_{\bar{j}} = \sum_j A^+_{ij}v_j. \tag{5}$$

The multiplicative update rule is

$$v_i \leftarrow \left[\frac{-b_i + \sqrt{b_i^2 + 4a_ic_i}}{2a_i}\right] v_i \tag{6}$$

and this converges to the minimum [10].

Non-negative matrix factorization (NMF) is very similar to NQP [6]. Non-negative matrix factorization seeks to minimize the Froebenius error

$$E = \|\mathbf{N} - \mathbf{MF}\|^2 \tag{7}$$

for an approximate factorization of a non-negative $n \times m$ matrix $\mathbf{N}$ into

$$\widetilde{\mathbf{N}} = \mathbf{MF} \tag{8}$$

where $\mathbf{M}$ and $\mathbf{F}$ are $n \times r$ and $r \times m$ matrices which are constrained to being non-negative. Expanding out E gives

$$E = N_{\bar{i}\bar{j}}N_{\bar{i}\bar{j}} - 2M_{\bar{i}\bar{j}}F_{\bar{j}\bar{k}}N_{\bar{i}\bar{k}} + M_{\bar{i}\bar{j}}F_{\bar{j}\bar{k}}M_{\bar{i}\bar{l}}F_{\bar{l}\bar{k}}. \tag{9}$$

Manuscript received 30 March 2008. Science Foundation Ireland grant 06/RFP/BIM020 and support by the Mathematics Applications Consortium for Science and Industry are acknowledged.

School of Mathematics, Trinity College Dublin, Dublin 2, Ireland e-mail: houghton@maths.tcd.ie

Now, if the $\mathbf{F}$ matrix is regarded as fixed, NQP gives the update rule

$$M_{ij} \leftarrow \left[\frac{N_{i\bar{k}}F_{j\bar{k}}}{\widetilde{N_{i\bar{l}}F_{j\bar{l}}}}\right] M_{ij} \tag{10}$$

and, if $\mathbf{M}$ is fixed, it gives

$$F_{ij} \leftarrow \left[\frac{M_{\bar{k}i}N_{\bar{k}j}}{\widetilde{M_{\bar{l}i}N_{\bar{l}j}}}\right] F_{ij}. \tag{11}$$

In fact, in NMF, neither matrix is fixed and both updates are performed simultaneously. NMF has weaker convergence properties than NQP: each iteration reduces E, but convergence is difficult to establish [11], [12], [13].

NMF has proved effective at extracting features from data sets. For example, one of the illustrative applications given in the original paper [6] is a corpus of faces and the NMF features are recognizable of parts of faces. Non-negative matrix deconvolution (NMD), an extension of NMF in which factorization is replaced by deconvolution [7] is useful for decomposing sound spectrograms. The purpose of this paper is to use the NQP formula to introduce an NMD-like algorithm for decomposing sound in the time domain. The purpose of the algorithm is to find sparse components of sound without using the spectrogram, thereby avoiding the down-sampling and reconstruction errors typical of spectrogram methods. It is also particularly easy to add a sparseness term to the objective function in this algorithm.

II. The algorithm

The problem considered here is to approximately decompose a sound waveform $s(t)$ as

$$\tilde{s}(t) = \int_0^T d\tau\, h_{\bar{i}}(t-\tau)a_{\bar{i}}(\tau) \tag{12}$$

where the compactly-supported $a_i(\tau)$ are the sound basis and the $h_i(t)$ are the components over that basis; for later use let N be the number of components, so the sum over i is

$$\tilde{s}(t) = \sum_{i=1}^{N}\int_0^T d\tau\, h_i(t-\tau)a_i(\tau). \tag{13}$$

As a constraint we require $h(t) \geq 0$. The main point of this algorithm is to calculate a decomposition with a non-negative component. The reason that this might be desirable is two fold; first, having a non-negative component is useful for the potential applications in source separation and acoustic object recognition and, second, sparsification is particularly straight-forward if the component is non-negative.

Now, we want to minimize the error

$$E = \int_0^L dt(s-\tilde{s})^2 \tag{14}$$

where $t \in [0, L]$ is the domain of the sample $s(t)$. In practice both the waveform will be discretely sampled and all the integrals will be sums over time steps. This means the problem is to minimize

$$E = S_{\bar{i}}S_{\bar{i}} - 2S_{\bar{j}}H_{\bar{i}\bar{j}\bar{k}}A_{\bar{i}\bar{k}} + H_{\bar{i}\bar{j}\bar{k}}A_{\bar{i}\bar{k}}H_{\bar{l}\bar{j}\bar{m}}A_{\bar{i}\bar{m}} \tag{15}$$

where

$$\begin{aligned} S_i &= s(i\delta t) \\ H_{ij} &= h_i(j\delta t) \\ H_{ijk} &= H_{i(j-k)} = h_i(j\delta t - k\delta t) \\ A_{ij} &= a_i(j\delta t) \end{aligned} \tag{16}$$

δt is the time step and the non-negative constraint means $H_{ijk} > 0$.

For fixed basis A_{ij} the NQP algorithm can be applied. In principle this is a straightforward calculation, but from a notational point of view it is complicated by the need to find an update for H_{ij} rather than the H_{ijk} that appear in the expression for E. This requires a change of index changing the ordinary convolutions to forward convolutions. Thus, for example.

$$-2S_{\bar{j}}H_{\bar{i}\bar{j}\bar{k}}A_{\bar{i}\bar{k}} \equiv -2\int dt \int d\tau s(t)h_{\bar{i}}(t-\tau)a_{\bar{i}}(\tau) \tag{17}$$

becomes

$$-2S_{\bar{j}\bar{k}}H_{\bar{i}\bar{j}}A_{\bar{i}\bar{k}} \equiv -2\int dt \int d\tau s(t+\tau)h_{\bar{i}}(t)a_{\bar{i}}(\tau) \tag{18}$$

after the change of index, where $S_{jk} = s(j\delta t + k\delta t)$. Now

$$H_{ij} \leftarrow \left[\frac{-b_{ij} + \sqrt{b_{ij}^2 + 4a_{ij}c_{ij}}}{2a_{ij}}\right] H_{ij} \tag{19}$$

where

$$\begin{aligned} a_{ij} &= A^+_{i\bar{k}}\tilde{S}^+_{j\bar{k}} + A^-_{i\bar{k}}\tilde{S}^-_{j\bar{k}} \\ b_{ij} &= -S_{j\bar{k}}A_{i\bar{k}} \\ c_{ij} &= A^-_{i\bar{k}}\tilde{S}^+_{j\bar{k}} + A^+_{i\bar{k}}\tilde{S}^-_{j\bar{k}} \end{aligned} \tag{20}$$

where

$$\tilde{S}_i = H_{\bar{j}i\bar{k}}A_{\bar{j}\bar{k}} \tag{21}$$

and $\tilde{S}_{ik} = \tilde{S}_{i+k}$. The superscript $\pm$ refer to the positive or negative parts, as in (2) above.

In contrast, since the basis is unconstrained, it can be updated exactly by minimizing E: by differentiating, the minimizing A_{ij} satisfies

$$S_{\bar{k}}H_{i\bar{k}j} = H_{i\bar{k}j}H_{\bar{l}\bar{k}\bar{m}}A_{\bar{l}\bar{m}} \tag{22}$$

and, although the formulation of the problem has created lots of indices, this is basically a matrix equation:

$$V_I = M_{I\bar{J}} A_{\bar{J}} \tag{23}$$

where we have vectorized by setting $I = Ni + j$, $J = Nl+m$, $V_I = S_{\bar{k}} H_{i\bar{k}j}$, $M_{IJ} = H_{i\bar{k}j} H_{l\bar{k}m}$ and $A_J = A_{lm}$. M will generally be invertible provided $N < L/\delta t$. In practical tests on sample data, an algorithm which alternates between the NQP (22) updates of the components and the least squares updates of the basis (23) finds a good approximation to the original sound waveform. However, the corresponding components, $h_i(t)$, are not particularly sparse.

Obviously the best way to ensure sparseness is to add a sparseness term to E. This is particularly easy because h is non-negative:

$$E = \int_0^L dt(s-\tilde{s})^2 + 2\lambda \sum_i \int_0^L dt\, h_i \tag{24}$$

where λ is a parameter fixing the relative importance of accuracy and sparseness; the factor of two is a notational convenience. Now, since $a_i(t) \leftarrow \sigma a_i(t)$, $h_i(t) \leftarrow h_i(t)/\sigma$ does not alter $\tilde{s}(t)$, but will, for $\sigma > 1$, reduce E, trying to minimizing this E will lead to the components getting smaller and smaller and the basis larger and larger. One way to stop this is to fix the size of the $a_i(t)$. Thus, the new objective function is

$$\begin{aligned} E &= \int_0^L dt(s-\tilde{s})^2 + 2\lambda \sum_i \int_0^L dt\, h_i \\ &\quad + \sum_i \mu_i \left(\int_0^T dt\, a_i^2 - 1 \right) \end{aligned} \tag{25}$$

where the μ_i are Lagrange multipliers. Converting this to matrix notation, the NQP step becomes

$$H_{ij} \leftarrow \left[\frac{-(b_{ij}-\lambda) + \sqrt{(b_{ij}-\lambda)^2 + 4a_{ij}c_{ij}}}{2a_{ij}} \right] H_{ij} \tag{26}$$

with a_{ij}, b_{ij} and c_{ij} unchanged from before (20).

The least squares update is now more difficult: it is now a constrained quadratic programming problem and can not be solved exactly. However numerically, it just means solving the Newton equations for

$$\begin{aligned} E &= -2V_{\bar{I}}A_{\bar{I}} + A_{\bar{I}}M_{\bar{I}\bar{J}}A_{\bar{J}} + \sum_i \mu_i \sum_{I=N(i-1)}^{Ni-1} (A_{\bar{I}}A_{\bar{I}} - 1) \\ &\quad + \text{terms independent of } A_I. \end{aligned} \tag{27}$$

The Newton equations for the extremum within the constrained space are

$$\begin{aligned} \frac{\partial L}{\partial A_I} &= 0 \\ \frac{\partial L}{\partial \mu_i} = \sum_{I=N(i-1)}^{Ni-1} A_{\bar{I}}A_{\bar{I}} - 1 &= 0 \end{aligned} \tag{28}$$

and these can be solved using a numerical root finder.

From this, two algorithms can be formulated. In the first the NPQ update is iterated until the objective function equilibrates, reaching a minimum for that value of the basis a_is.

Algorithm 1

Initialize A_{ij} and H_{ij}.

Until the objective function E, (25), equilibrates:

NQP update:
Until E equilibrates:
Calculate $\tilde{S}_i$
Update H_{ij} using (26)

Least squares update:
Calculate V_I and M_{IJ}.
Minimize E by numerically solving (28)

In the second the NPQ update is only iterated for a small, fixed number of iterations before the basis a_is are changed in the least squares update.

Algorithm 2

Initialize A_{ij} and H_{ij}.

Until the objective function E, (25), equilibrates:

NQP update:
For a fixed number of iterations:
Calculate $\tilde{S}_i$
Update H_{ij} using (26)

Least squares update:
Calculate V_I and M_{IJ}.
Minimize E by numerically solving (28)

In either case, the NPQ and least squares update alternate until the objective function E equilibrates. In fact, for

the sample described in the next section, Sect. III, Algorithm 2 is much more effective, equilibrating faster.

III. Results

As an example the Algorithm 2 has been applied to recorded speech. The sample[1] was taken from librivox, a public domain collection of poetry read and recorded by amateur volunteers.[2] The sample used was two minutes long. It was downloaded as ogg vorbis and converted to a waveform down-sampled to 8kHz using `sox`. [3] The number of components N is set to 20, δt is set equal the sample rate, so $\delta t = .125$ ms and the width of the basis functions $a_i(\tau)$ is 2.5 ms, meaning that $T/\delta t$ is also 20. In each iteration the NQP update was iterated four times, the a_i were then updated using the Newton Equation routine described in [14]. The basis were initialized as

$$a_i(t) = \sin \frac{\pi t}{T} \sin f_i t \tag{29}$$

where the $f_0 = \pi/2\delta t$, $f_{19} = 2\pi/T$ and the others are evenly spaced in between. The components were initialized randomly with each H_{ij} assigned a random number between zero and 0.03. It is likely that the speed of the algorithm could be improved if some more sample-specific choice of initial H_{ij} values was made, for example, varying the range of the random values used does effect the run time. Increasing the number of components and the width of a_i improves the final result, but increases the time each iteration takes.

If the sparseness parameter is zero, $\lambda = 0$, the estimated sound $\tilde{s}(t)$ approaches the recording $s(t)$. For $\lambda = .0005$ convergence requires 48 NPQ updates. The resulting approximation is not particularly good:

$$\sqrt{\frac{\int dt\,(s-\tilde{s})^2}{\int dt\, s^2}} \approx .098 \tag{30}$$

This is because the approximation underestimates the recording in order to increase sparseness.

Where the algorithm does succeed is in producing sparse components. The $h_i(t)$ are very sparse. Across the twenty components, 0.86 of the H_{ij} have values less than 10^{-6}; the average value of the H_{ij} which are greater than 10^{-6} is .0025. I believe that sparse components of this sort will be useful in applications of sparse-coding based methods.

Acknowledgment

B. Pearlmutter is thanked for useful conversation.

[1] The Ballad of Reading Gaol by Oscar Wilde, read by John Gonzales.

[2] `http://librivox.org/`

[3] `http://sourceforge.net/projects/sox/`

References

[1] P. D. O'Grady, B. A. Pearlmutter, and S. T. Rickard, "Survey of sparse and non-sparse methods in source separation," *International Journal of Imaging Systems and Technology*, no. 1, pp. 18–33, 2005.

[2] S. A. Abdallah and M. D. Plumbley, "Unsupervised analysis of polyphonic music using sparse coding," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 179–196, January 2006.

[3] H. Asari, B. A. Pearlmutter, and A. M. Zador, "Sparse representations for the cocktail party problem," *Journal of Neuroscience*, vol. 26, no. 28, pp. 7477–7490, 2006.

[4] H. Asari, R. Olsson, B. Pearlmutter, and A. Zador, *Sparsification for monaural source separation*. Springer Verlag, 2007.

[5] M. G. Jafari, S. A. Abdallah, M. D. Plumbley, and M. E. Davies, "Sparse coding for convolutive blind audio source separation," in *Proceedings of the 6th International Conference on Independent Component Analysis and Blind Source Separation (ICA 2006)*, March 2006, pp. 132–139.

[6] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–91, 1999.

[7] P. Smaragdis, "Discovering auditory objects through non-negativity constraints," in *Statistical and Perceptual Audio Processing (SAPA)*, 2004.

[8] F. Sha, L. K. Saul, and D. D. Lee, "Multiplicative updates for nonnegative quadratic programming in support vector machines," in *Advances in Neural Information Processing Systems 15*, S. T. S. Becker and K. Obermayer, Eds. Cambridge, MA,: MIT Press, 2003, pp. 1041–1048.

[9] ——, "Multiplicative updates for large margin classifiers," in *Proceedings of the Sixteenth Annual Conference on Computational Learning Theory (COLT)*, Washington D.C., USA, 2003.

[10] F. Sha, Y. Lin, L. K. Saul, and D. D. Lee, "Multiplicative updates for nonnegative quadratic programming," *Neural Computation*, vol. 19, no. 8, pp. 2004–2031, 2007.

[11] L. Finesso and P. Spreij, "Approximate Nonnegative Matrix Factorization via Alternating Minimization," *ArXiv Mathematics e-prints*, Feb. 2004.

[12] E. F. Gonzales and Y. Zhang, "Accelerating the lee-seung algorithm for non-negative matrix factorization," Dept. Comput. Appl. Math., Rice University, Houston, TX, Tech. Rep., 2005.

[13] C.-J. Lin, "On the convergence of multiplicative update algorithms for nonnegative matrix factorization," *Neural Networks, IEEE Transactions on*, vol. 18, no. 6, pp. 1589–1596, Nov. 2007.

[14] W. T. V. William H. Press, Saul A. Teukolsky and B. P. Flannery, *Numerical recipes 3rd edition: the art of scientific computing*, 3rd ed. Cambridge University Press, 2007.