FACTORIZATION OF OVERLAPPING HARMONIC SOUNDS USING APPROXIMATE MATCHING PURSUIT

Steven K. Tjoa
Imagine Research
San Francisco, CA 94114 USA
steve@imagine-research.com

K. J. Ray Liu
University of Maryland
College Park, MD 20742 USA
kjrliu@umd.edu

ABSTRACT

Factorization of polyphonic musical signals remains a difficult problem due to the presence of overlapping harmonics. Existing dictionary learning methods cannot guarantee that the learned dictionary atoms are semantically meaningful. In this paper, we explore the factorization of harmonic musical signals when a fixed dictionary of harmonic sounds is already present. We propose a method called approximate matching pursuit (AMP) that can efficiently decompose harmonic sounds by using a known predetermined dictionary. We illustrate the effectiveness of AMP by decomposing polyphonic musical spectra with respect to a large dictionary of instrumental sounds. AMP executes faster than orthogonal matching pursuit yet performs comparably based upon recall and precision.

1. INTRODUCTION

Dictionary learning, sparse coding, and constrained factorization algorithms have recently revolutionized the way we perform music transcription and source separation. Many researchers have reported success when decomposing simple musical signals using nonnegative matrix factorization (NMF) [23] or methods based upon sparse coding such as K-SVD [1,2]. Unfortunately, problems remain for intricate, polyphonic musical signals. When musical notes overlap in time and frequency, the separation and transcription performance of these basic dictionary learning methods diminishes rapidly. In such a case, the algorithm will usually learn a dictionary where each individual atom contains information from multiple musical sources, thus hindering our attempts at decomposition.

Researchers have slowly improved upon the original dictionary learning methods by adding constraints to the learning process. By restricting the dictionary atoms to reside within a predetermined feasible set, we can ensure that the learned atoms will be useful at the conclusion of the learning process. For example, existing solutions include adding constraints to the dictionary learning process such as harmonicity [3,25] or smoothness [3,26].

Another solution is to add structure to the dictionary. For example, one can construct and use a large, predefined, overcomplete dictionary where each atom is already labeled and assumed to contain information from only one musical source. Instead of learning an optimal dictionary for a given musical signal, it may suffice to match the signal to this large set of precomputed, labeled dictionary atoms. Then, by decomposing a signal with respect to this fixed dictionary, classification is easily achieved by simply reading the label of the atom. As musical databases become more available, construction of predefined dictionaries will become easier, thus reducing the need for adaptive dictionary learning.

Of course, the performance of such an algorithm depends upon the breadth of the dictionary. When atoms from more musical sources are added to the dictionary, the dictionary's ability to decompose polyphonic music will improve. However, dictionary growth introduces concerns related to scalability and computational complexity. While the aforementioned algorithms have significantly advanced the state of the art, they remain slow and difficult to scale as the dictionary size increases. Most of the original factorization methods such as matching pursuit (MP) [18] and NMF with multiplicative updates [17] have complexity that is linear in the size of the dictionary. As a result, when dictionary sizes grow, the transcription efficiency of these algorithms diminishes.

To summarize the problem: how can we make use of a large, precomputed, overcomplete dictionary to factorize overlapping harmonic sounds accurately and efficiently?

We address this problem by proposing a variant of MP called *approximate matching pursuit* (AMP). Unlike MP and NMF, AMP can decompose signals into a sparse combination of atoms with complexity that is *sublinear* in the dictionary size while maintaining accuracy. To do this, AMP uses

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page.
© 2011 International Society for Music Information Retrieval.

an approximate nearest neighbor (ANN) method to find approximate matches to the signal residual at each iteration. The ANN method that we choose in this work is locality sensitive hashing (LSH), a probabilistic hash algorithm that places similar, yet not identical, observations into the same bin. LSH can retrieve near neighbors with a complexity that is sublinear in the dictionary size.

Our experiments demonstrate that AMP is as capable as orthogonal matching pursuit (OMP) [20] for decomposing polyphonic musical spectra into combinations of atoms from a large dictionary of over 17,000 labeled musical spectra. Meanwhile, AMP requires less computation and factorizes more quickly than OMP.

2. RELATED WORK

Computation of sparse coefficients with respect to a large, overcomplete dictionary is often accomplished by pursuit algorithms such as MP [18]. This greedy algorithm directly addresses the issue of sparsity by decomposing a signal, $\mathbf{x}$, into a linear expansion of waveforms that are selected from a redundant dictionary of functions. When stopped after a few iterations, this algorithm yields a signal approximation using only a few atoms. After each iteration of the MP algorithm, the residual, $\mathbf{r}$, is orthogonal to the previously selected vector, $\mathbf{a}_k$, but not necessarily orthogonal to the dictionary vectors selected earlier.

Pati et al. proposed OMP, an improvement over MP which ensures that the residual is orthogonal to all previously selected dictionary vectors [20]. After dictionary atoms are selected for inclusion into the decomposition, an extra orthogonalization step is performed by solving a least-squares problem. Researchers have shown that OMP provides a dramatic improvement over MP [20]. In many cases, when an input signal is known to be k-sparse, OMP converges in k iterations, while MP will require many more iterations to converge.

Pursuit algorithms have been applied to MIR in many ways. The most popular applications are music transcription and source separation. Harmonic matching pursuit (HMP) has been used to decompose an audio signal into Gabor or harmonic (i.e., sums of Gabor) atoms [15]. Dictionaries of atoms can also be adapted and learned to fit the data [9]. To resolve instances when harmonics from separate notes overlap, some algorithms impose smoothness constraints [4]. Similar sparse coding methods have been used for genre recognition [19]. In the neurological signal processing literature, pursuit methods for generic acoustic signals have been applied for coding purposes [24].

Cotton and Ellis [10] also use LSH together with MP, however that work addresses a fundamentally different problem – content-based search of whole acoustic events, e.g., the sound made by a horse's hoof. There, the sparse representation produced by MP is stored using LSH. On the other hand, our proposed method addresses the problems of transcription and source separation. As shown later, we enhance MP by embedding LSH *within* MP to make it faster and more scalable. Also, we use a massive dictionary of real-world musical spectra, not synthetic Gabor atoms as in [10].

3. PROBLEM FORMULATION

Given the magnitude spectrum of an input signal, $\mathbf{x} \in \mathbb{R}^M$, and a dictionary, $\mathbf{A} = [\mathbf{a}_1\, \mathbf{a}_2\, ...\, \mathbf{a}_K] \in \mathbb{R}^{M \times K}$, the problem is to find a vector of coefficients $\mathbf{s} \in \mathbb{R}^K$ that minimizes $||\mathbf{x} - \mathbf{A}\mathbf{s}||_2$.

When $M < K$, the dictionary is called *overcomplete*, and there are infinitely many solutions for $\mathbf{s}$. However, by imposing a sparsity constraint on $\mathbf{s}$, the solution space diminishes greatly, possibly to a unique solution. In particular, if the input is truly a sparse linear combination of dictionary atoms, i.e., $\mathbf{x} = \mathbf{A}\mathbf{s}_0$, where $\mathbf{s}_0$ is a sparse vector, then the problem becomes finding an optimal set of coefficients, $\hat{\mathbf{s}} = \mathrm{argmin}_{\mathbf{s}}\, ||\mathbf{x} - \mathbf{A}\mathbf{s}||_2$, that is equal to the input coefficients, i.e., $\hat{\mathbf{s}} = \mathbf{s}_0$.

An exhaustive search for the sparsest solution is NP-hard [12]. However, suboptimal greedy algorithms such as OMP often work well in practice. Unfortunately, OMP requires at least K inner products to computed during each iteration, thus creating a complexity that is at least linear in K. Because this complexity is too slow for large dictionaries, the problem becomes solving for $\hat{\mathbf{s}} = \mathbf{s}_0$ using an algorithm that has complexity that is sublinear in the dictionary size, K.

Without loss of generality, we assume that the dictionary is overcomplete, $M < K$; this assumption is not strictly necessary for AMP to operate. We also assume that the true sparsity of any input signal, $||\mathbf{s}_0||_0$, is less than the dimensionality, M. For musical signals, this assumption usually holds in practice. For example, even in highly polyphonic music, the number of simultaneous sounds will likely be significantly less than the dimensionality of our spectra, i.e., the number of frequency bins. If not, then we increase the FFT size to produce longer spectra.

4. PROPOSED ALGORITHM: APPROXIMATE MATCHING PURSUIT

One drawback of existing pursuit methods such as MP and OMP is their complexity. When the dictionary size, K, becomes very large (e.g., over one million), these methods may require an unacceptably large amount of computation to find an answer. For example, in each iteration of MP, K inner products must be computed between the residual $\mathbf{r}$ and every atom in the dictionary – a complexity of order $O(MK)$. Here, we introduce a simple variation of these pursuit methods that uses an ANN algorithm in place of computing K inner products as done in MP. As a result, we can reduce the complexity to be sublinear in K.

The approximate matching pursuit (AMP) algorithm is described in Algorithm 1. This algorithm is similar to OMP except that it addresses the main computational bottleneck for large dictionaries – nearest neighbor search – by allowing any adequately near neighbor to be selected as a component.

Algorithm 1 Approximate Matching Pursuit [Tjoa and Liu]

```
Input: $\mathbf{x} \in \mathbb{R}^M$; $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_K] \in \mathbb{R}^{M \times K}$ s.t. $||\mathbf{a}_k||_2 = 1$ for all $k$.
Output: $\hat{\mathbf{s}} \in \mathbb{R}^K$
Initialize: $\mathcal{S} \leftarrow \emptyset$; $\mathbf{s} \leftarrow \mathbf{0}$; $\mathbf{r} \leftarrow \mathbf{x}$; $\epsilon > 0$.
while $||\mathbf{r}|| > \epsilon$ do
    Find any $k$ such that $\mathbf{a}_k$ and $\mathbf{r}$ are near neighbors.
    $\mathcal{S} \leftarrow \mathcal{S} \cup k$
    Solve for $\{s_j | j \in \mathcal{S}\}$: $\min_{s_j|j \in \mathcal{S}} ||\mathbf{x} - \sum_{j \in \mathcal{S}} \mathbf{a}_j s_j||_2$
    $\mathbf{r} \leftarrow \mathbf{x} - \mathbf{A}\mathbf{s}$
$\hat{\mathbf{s}} \leftarrow \mathbf{s}$
```

AMP intentionally resembles MP and OMP. Like OMP, AMP is capable of providing a sparse decomposition in far fewer iterations than MP. If the ANN retrieval method were instead changed to a nearest-neighbor (NN) method, then AMP would yield identical results to OMP. Also, AMP is flexible in the sense that any ANN method could be used as long as it performs retrieval in sublinear time. Therefore, AMP can also be considered as a modular framework of algorithms.

Despite its simplicity, AMP embodies a fundamentally different philosophy to signal factorization. AMP is a data-driven algorithm, not a model- or knowledge-based algorithm. With such an abundance of available musical data, we use side information, not rigid mathematical models, to represent test data. Algorithmic advances such as AMP, coupled with technological advances in computing, are making data-driven algorithms more computationally feasible for problems in MIR such as transcription and source separation.

5. LOCALITY SENSITIVE HASHING

AMP allows the use of any ANN algorithm that can perform retrieval in sublinear time. For this work, we focus on locality-sensitive hashing (LSH), a category of algorithms that places nearby points in a high-dimensional space into the same bin in a hash table. Because of its simplicity, robustness, and low complexity, LSH has become popular for solving many high-level problems beyond MIR such as search and retrieval of text and images. The robustness of LSH is desirable for problems in MIR where queries are often distorted due to environmental or musical variation, and therefore, learned dictionary atoms will rarely match predefined dictionary atoms exactly. Ryynänen and Klapuri used LSH to perform query-by-humming (QBH) by constructing a hash table from pitch contour vectors [21]. Yu et al. use LSH and order statistics to store chroma features in a hash table for audio content retrieval [28]. Cotton and Ellis use LSH to store landmarks in audio that correspond to meaningful acoustic events [10]. Casey and Slaney have used LSH to store features called audio shingles for computing various levels of musical similarity between songs [5–7].

However, LSH has rarely been used for signal-level problems like music transcription. To our knowledge, this work is among the first in MIR to use LSH for low-level tasks such as sparse coding and music transcription.

While other ANN algorithms can be used within AMP instead of LSH, such as those that use space partitioning like the kd-tree and hierarchical k-means, these algorithms do not work well in high-dimensional spaces, i.e., dimensionality over 100. In fact, all current indexing techniques based on space partitioning degrade to linear search for sufficiently high dimensions [11, 14, 27]. Therefore, we only consider LSH in this work.

In this work, for $i \in \{1, 2, ..., k\}$ and $\ell \in \{1, 2, ..., L\}$, we define the function h_i^ℓ to be

$$h_i^\ell(\mathbf{q}) = \text{sign}\langle \mathbf{p}_i^\ell, \mathbf{q} \rangle \tag{1}$$

where $\mathbf{p}_i^\ell$ is a zero-mean, unit variance, Gaussian random vector with independent elements. As illustrated later, the parameters k and L adjust the tradeoff between recall and precision of the dictionary atoms.

It has been shown that this choice of distribution on $\mathbf{p}_i^\ell$ will hash points together whose angle,

$$\theta(\mathbf{q}, \mathbf{r}) = \arccos \frac{\langle \mathbf{q}, \mathbf{r} \rangle}{||\mathbf{q}|| ||\mathbf{r}||}, \tag{2}$$

is small [8]. Specifically, it can be shown that, for any i and ℓ, the probability that $h_i^\ell(\mathbf{q}) = h_i^\ell(\mathbf{r})$ is equal to

$$P(h_i^\ell(\mathbf{q}) = h_i^\ell(\mathbf{r})) = 1 - \frac{\theta(\mathbf{q}, \mathbf{r})}{\pi}. \tag{3}$$

We claim that two hashes are equal, $h(\mathbf{q}) = h(\mathbf{r})$, if and only if there exists an ℓ such that, for all $i \in \{1, 2, ..., k\}$, $h_i^\ell(\mathbf{q}) = h_i^\ell(\mathbf{r})$. In other words, the following events are equivalent:

$$\{h(\mathbf{q}) = h(\mathbf{r})\} = \cup_{\ell=1}^L \cap_{i=1}^k \{h_i^\ell(\mathbf{q}) = h_i^\ell(\mathbf{r})\}. \tag{4}$$

From (3) and (4), it can be shown that the probability that $h(\mathbf{q}) = h(\mathbf{r})$ is equal to

$$P(h(\mathbf{q}) = h(\mathbf{r})) = 1 - \left(1 - \left(1 - \frac{\theta(\mathbf{q}, \mathbf{r})}{\pi}\right)^k\right)^L. \tag{5}$$

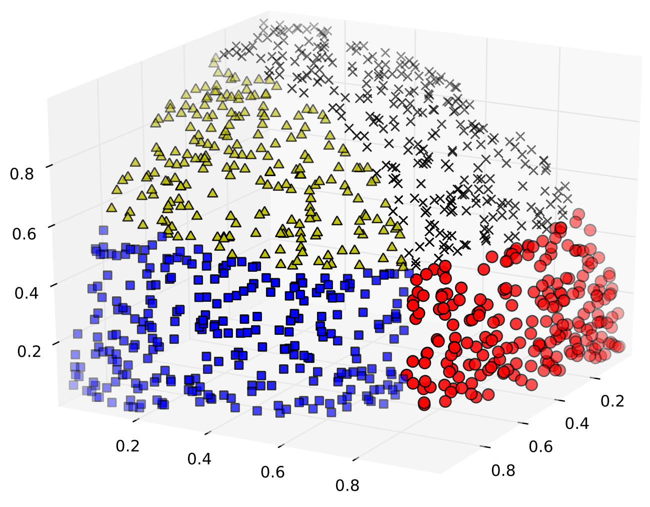

Figure 1. LSH example with $k = 2$. Points on the unit sphere are separated into $2^k = 4$ bins.

To construct the LSH table, we initialize L empty tables. For each atom $\mathbf{a}$ in the dictionary $\mathbf{A}$, and for each $\ell \in \{1, 2, ..., L\}$, its hash is computed as a k-tuple:

$$h^\ell(\mathbf{a}) = (h^\ell_1(\mathbf{a}), h^\ell_2(\mathbf{a}), ..., h^\ell_k(\mathbf{a})), \qquad (6)$$

and $\mathbf{a}$ is placed into bin $h^\ell(\mathbf{a})$ of table ℓ. Finally, to perform a query for point $\mathbf{r}$, for all ℓ, we retrieve all of the points in bin $h^\ell(\mathbf{r})$ of table ℓ. Among these retrieved points that share a bin with $\mathbf{r}$, we perform exhaustive search to find the nearest neighbor among them. As indicated by Eq. 5, through the proper choice of k and L, one can achieve any desired amount of similarity between any two input vectors.

An example of LSH is shown in Figure 1 when $k = 2$. Points on the unit sphere are hashed, and those points that reside in the same bin share the same marker. We notice that points in the same bin are close together.

There are many theoretical results for LSH that are beyond the scope of this paper. For detailed discussion and proofs, please see [11, 14, 22, 27].

6. EXPERIMENTS

To illustrate the performance of AMP, we factorize polyphonic spectra as sparse combinations of atoms from a dictionary of real piano sounds. First, we discuss how to build a dictionary. For this work, our data comes from the University of Iowa database of musical instrument samples [13]. Each file in the data set is labeled by pitch and loudness, e.g., “Piano C4 mf”, and contains a signal of an isolated note sampled at 44100 Hz. We only consider the subset of piano sounds.

For each signal, we compute a short-time Fourier transform with a frame size of 92.9 milliseconds (i.e., 4096/44100) and a hop of 10 milliseconds. To discard silent segments, we detect any spectrum whose power is below a threshold. The remaining spectra are normalized to have unit Euclidean norm and are saved along with their pitch labels. These normalized spectra constitute the dictionary, $\mathbf{A}$, and the pitch labels are used later to evaluate matches among dictionary atoms. In total, we use a dictionary of 17,753 spectra of piano sounds covering the entire piano keyboard (i.e., MIDI values 21 through 108).

For the following experiments, the input to AMP is a vector $\mathbf{x} \in \mathbb{R}^M$, a magnitude spectrum containing overlapping harmonic sounds, where $\mathbf{x} = \mathbf{A}\mathbf{s}_0$. $\mathbf{A}$ is the dictionary of size M-by-K described earlier, and $\mathbf{s}_0$ is a synthetically generated sparse vector of length K containing λ ones in uniformly random locations. In other words, λ determines the number of overlapping sounds at any moment. We vary λ in the following experiments.

The LSH structure accepts parameters L and k, where L is the number of LSH tables and k is the length of each key. The dictionary, $\mathbf{A}$, is used to populate each of the L LSH tables as described in Section 5. Finally, given the input $\mathbf{x}$ and the LSH tables, AMP produces a sparse coefficient vector, $\hat{\mathbf{s}}$.

Given the output, $\hat{\mathbf{s}}$, we count the number of hits, misses, and false alarms. A hit occurs if an element in $\mathbf{s}_0$ matches an element in $\hat{\mathbf{s}}$. A miss occurs if an element in $\mathbf{s}_0$ does not match any element in $\hat{\mathbf{s}}$. A false alarm occurs if an element in $\hat{\mathbf{s}}$ does not match any element in $\mathbf{s}_0$. A match occurs when two coefficients share the same pitch label.

All source code is written in Python using the NumPy, SciPy, and Matplotlib packages [16].

In Figure 2, we compare AMP against another pursuit method, OMP. For each algorithm, using the number of hits, misses, and false alarms, we plot the recall, precision, and F-measure. Recall is defined as R = hits/(hits + misses), precision is defined as P = hits/(hits + false alarms), and F-measure is defined as $F = 2PR/(P + R)$. We also monitor the execution time and number of M-dimensional inner products computed by each algorithm. All quantities are averaged over twenty independent trials.

From Figure 2, we see that the recall, precision, and F-measure are all relatively similar for both algorithms. The recall for AMP is nearly as high as that of OMP. The gap in precision between the algorithms is slightly larger. In practice, the stopping criterion can affect the tradeoff between recall and precision. When convergence occurs early, $\hat{\mathbf{s}}$ is more sparse; therefore, recall decreases and precision increases. When convergence occurs late, $\hat{\mathbf{s}}$ is less sparse; therefore, recall increases and precision decreases. For this work, we simply fix the stopping criterion such that the ratio of the residual norm to the input norm, $||\mathbf{r}||/||\mathbf{x}||$, is equal to 0.25. A more sophisticated stopping criterion may be able to improve this tradeoff.

Next, we plot the execution time. Results show that AMP executes approximately two to four times faster than OMP. The parameters used in LSH, (L, k), affect execution time. When the length of the key, k, is low, then there are fewer keys and more elements per bin. Therefore, the candidate set of spectra is larger. When k is high, there are more keys and fewer elements per bin resulting in a smaller candidate set. The number of tables, L, has the opposite effect of k. When L is high, the size of the candidate set increases. When L is low, the candidate set size decreases.

Finally, we plot the number of M-dimensional inner products computed by both algorithms. This measure describes the primary source of computational effort. We see that OMP requires far more inner products than AMP. For OMP, each iteration requires K inner products because the residual is matched against every dictionary atom. For AMP, each iteration requires far fewer than K inner products because LSH only retrieves those dictionary atoms that are likely close to the residual vector. However, we notice that the gap in the number of inner products computed by OMP and AMP is larger than the gap in execution time. This discrepancy is largely caused by overhead required of LSH, for example, key computation, data subset retrieval, etc. Optimizing these operations at a lower level could further widen the gap in execution time between AMP and OMP.

7. CONCLUSION

We have proposed AMP, a pursuit algorithm that can decompose overlapping harmonic spectra as well as OMP while executing in less time and requiring fewer computations. We have shown that the recall, precision, and F-measure for AMP is comparable with that of OMP. Unlike OMP which has complexity that is linear in the size of the dictionary, AMP has sublinear complexity and is therefore much faster. The simple modification of using LSH in place of exhaustive linear search makes previously infeasible techniques feasible once again. Previously, LSH has primarily been used to solve high-level tasks such as song or document retrieval; here, we use LSH for the signal-level tasks of factorization and separation.

AMP, like many recently proposed machine learning algorithms, uses real data rather than contrived models and constraints to describe musical spectra. We hope that this simple algorithm inspires a new class of methods that intelligently exploit the abundant musical data that already exists among public collections rather than chasing gains in fully unsupervised algorithms where little progress is left to be made.

The dictionary itself has a significant impact on the decomposition. Therefore, future work will include proper dictionary design, i.e., how to create dictionary atoms from musical data sets for maximum accuracy and efficiency. Dictionary design also affects the proper choice of LSH parameters, L and k. A careful analysis of pairwise distances among dictionary atoms can reveal which set of LSH parameters minimizes the probability of error.

8. REFERENCES

[1] M. Aharon, Michael Elad, and Alfred Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans. Signal Processing*, 54(11):4311–4322, November 2006.

[2] Michal Aharon, Michael Elad, and Alfred M. Bruckstein. K-SVD and its non-negative variant for dictionary design. In *Proc. SPIE Conf. Wavelets*, volume 5914, pages 327–339, July 2005.

[3] N. Bertin, R. Badeau, and E. Vincent. Enforcing harmonicity and smoothness in bayesian non-negative matrix factorization applied to polyphonic music transcription. *IEEE Trans. Audio, Speech, and Language Processing*, 18(3):538–549, March 2010.

[4] FJ Cañadas Quesada, P. Vera-Candeas, N. Ruiz-Reyes, R. Mata-Campos, and JJ Carabias-Orti. Note-event detection in polyphonic musical signals based on harmonic matching pursuit and spectral smoothness. *J. New Music Research*, 37(3):167–183, 2008.

[5] M. Casey, C. Rhodes, and M. Slaney. Analysis of minimum distances in high-dimensional musical spaces. *IEEE Trans. Audio, Speech, and Language Processing*, 16(5):1015–1028, 2008.

[6] M. Casey and M. Slaney. Song intersection by approximate nearest neighbor search. In *Proc. ISMIR*, volume 6, pages 144–149, 2006.

[7] M. Casey and M. Slaney. Fast recognition of remixed music audio. In *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, volume 4, pages IV–1425–IV–1428. IEEE, April 2007.

[8] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proc. ACM Symp. Theory of Computing*, pages 380–388. ACM, 2002.

[9] N. Cho, Y. Shiu, and C. C. J. Kuo. Efficient music representation with content adaptive dictionaries. In *Proc. IEEE Int. Symp. Circuits and Systems*, pages 3254–3257, 2008.

[10] C. Cotton and D.P.W. Ellis. Finding similar acoustic events using matching pursuit and locality-sensitive hashing. In *IEEE Workshop Appl. Signal Proc. Audio and Acoustics*, pages 125–128. IEEE, 2009.

[11] M. Datar, N. Immorlica, P. Indyk, and V.S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. 20th Annual Symposium on Computational Geometry*, pages 253–262, 2004.

[12] D.L. Donoho and J. Tanner. Thresholds for the recovery of sparse solutions via l1 minimization. In *40th Annual Conf. Information Sciences and Systems*, pages 202–206, March 2006.

[13] Lawrence Fritts. Musical instrument samples, 1997–.

[14] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. Int. Conf. Very Large Data Bases*, pages 518–529, 1999.

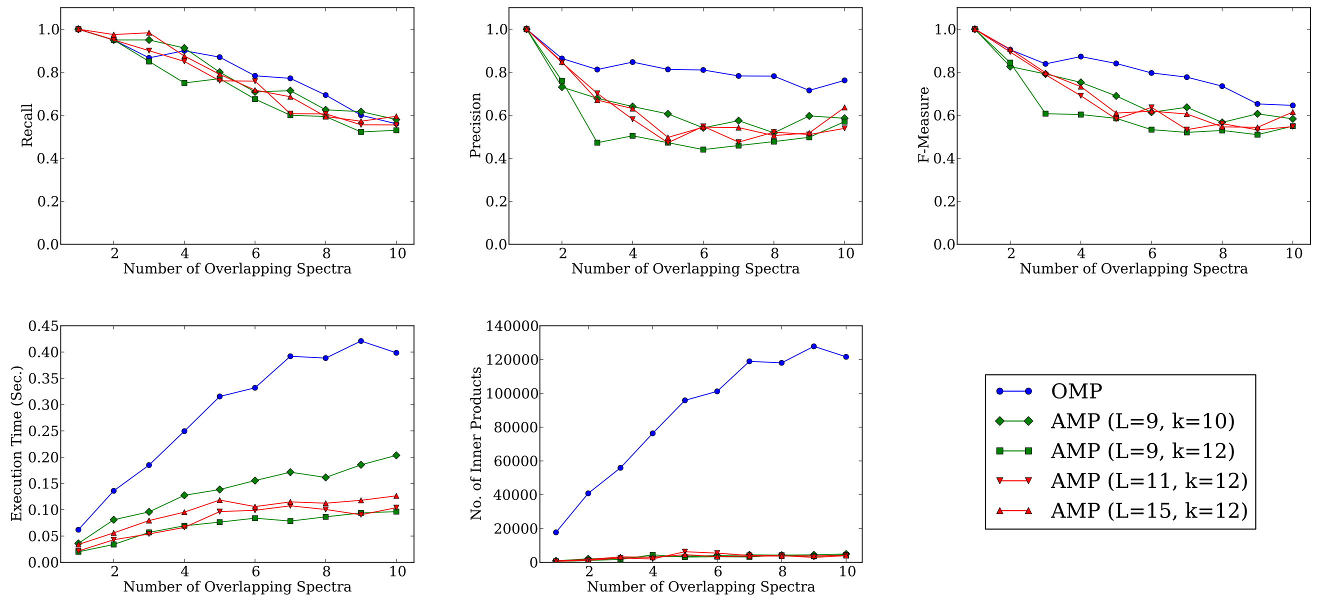

Figure 2. Recall, precision, F-measure, execution time, and number of inner products computed for OMP and AMP after decomposing an input spectrum containing overlapping sounds. Dictionary contains 17,753 spectra of piano sounds. All quantities are averaged over twenty trials.

[15] R. Gribonval and E. Bacry. Harmonic decomposition of audio signals with matching pursuit. *IEEE Trans. Signal Processing*, 51(1):101–111, 2003.

[16] Eric Jones, Travis Oliphant, Pearu Peterson, et al. Scipy: Open source scientific tools for python, 2001–.

[17] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[18] S.G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Trans. Signal Processing*, 41(12):3397–3415, 1993.

[19] Pierre-Antoine Manzagol, Thierry Bertin-Mahieux, and Douglas Eck. On the use of sparse time-relative auditory codes for music. In *Proc. Intl. Soc. Music Information Retrieval Conf.*, pages 603–608, 2008.

[20] Y.C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Asilomar Conf. Signals, Systems and Computers*, pages 40–44, 1993.

[21] M. Ryynänen and A. Klapuri. Query by humming of midi and audio using locality sensitive hashing. In *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pages 2249–2252, 2008.

[22] Malcolm Slaney and Michael Casey. Locality-sensitive hashing for finding nearest neighbors. *IEEE Signal Processing Mag.*, 25(2):128–131, March 2008.

[23] P. Smaragdis and J. C. Brown. Non-negative matrix factorization for polyphonic music transcription. In *Proc. IEEE Workshop on Appl. Signal Processing to Audio and Acoustics*, pages 177–180, New Paltz, NY, October 2003.

[24] E. Smith and M. S. Lewicki. Efficient coding of time-relative structure using spikes. *Neural Computation*, 17(1):19–45, 2005.

[25] Steven K. Tjoa, Matthew C. Stamm, W. Sabrina Lin, and K. J. Ray Liu. Harmonic variable-size dictionary learning for music source separation. In *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, pages 413–416, Dallas, TX, March 2010.

[26] Tuomas Virtanen. Monaural sound source separation by non-negative matrix factorization with temporal continuity and sparseness criteria. *IEEE Trans. Audio, Speech, and Language Processing*, 15(3):1066–1074, March 2007.

[27] R. Weber, H.J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proc. Int. Conf. Very Large Data Bases*, pages 194–205, 1998.

[28] Y. Yu, M. Crucianu, V. Oria, and E. Damiani. Combining multi-probe histogram and order-statistics based lsh for scalable audio content retrieval. In *ACM Int. Conf. Multimedia*, pages 381–390. ACM, 2010.