AMP: ASSEMBLY MATCHING PURSUIT

S. BISWAS

Department of Statistics and Operations Research, University of North Carolina at Chapel Hill
Chapel Hill, North Carolina, USA

V. JOJIC*

Department of Computer Science, University of North Carolina at Chapel Hill
Chapel Hill, North Carolina, USA
**E-mail: vjojic@cs.unc.edu*

This paper is submitted to the Pacific Symposium on Biocomputing 2013 session **Personalized medicine: from genotypes and molecular phenotypes towards therapy**. The paper contains contains original, unpublished results, and is not currently under consideration elsewhere. All co-authors concur with the contents of the paper.

AMP: ASSEMBLY MATCHING PURSUIT*

S. BISWAS

Department of Statistics and Operations Research, University of North Carolina at Chapel Hill
Chapel Hill, North Carolina, USA
E-mail: sbiswas@live.unc.edu

V. JOJIC*

Department of Computer Science, University of North Carolina at Chapel Hill,
Chapel Hill, North Carolina, USA
**E-mail: vjojic@cs.unc.edu*

Metagenomics, the study of the total genetic material isolated from a biological host, promises to reveal host-microbe or microbe-microbe interactions that may help to personalize medicine or improve agronomic practice. We introduce a method that discovers metagenomic units (MGUs) relevant for phenotype prediction through sequence-based dictionary learning. The method aggregates patient-specific dictionaries and estimates MGU abundances in order to summarize a whole population and yield universally predictive biomarkers. We analyze the impact of Gaussian, Poisson, and Negative Binomial read count models in guiding dictionary construction by examining classification efficiency on a number of synthetic datasets and a real dataset from Ref. 1. Each outperforms standard methods of dictionary composition, such as random projection and orthogonal matching pursuit. Additionally, the predictive MGUs they recover are biologically relevant.

1. Introduction

Advances in bioinformatics, refinements in DNA amplification, and the proliferation of computational power have greatly aided the analysis of DNA sequences recovered from environmental microbiomes. Early metagenomic studies focused on the sequencing of the 16S-rRNA sequence in an attempt to discover trends at a genus level.[2] Most reported large species diversity even between related hosts, and it is now becoming clear that metagenomic correlations may be better studied in other units such as genes or functional groups.[3] This requires the study of the full metagenome, a more complex task than 16S sequence study. In general, comparative metagenomics examines how the microbial composition of metagenomic samples correlates with host properties. If we can identify bacterial taxa, genes, or operons that are consistently predictive of disease, then these biological signatures could be used to build models that aid in diagnosis and treatment. For example, such approaches would have medical implications for diseases such as Inflammatory Bowel Disease (IBD) that may be treated via modulation of the gut microbiota.[4]

Our approach to summarization of metagenomic datasets is based on adaptive dictionary learning. In this framework, a signal (e.g. a set of DNA sequencing reads) is succinctly represented in terms of a small number of dictionary elements, sometimes called atoms or words. The history of dictionary learning is rich and varied, tracing back to projection pursuit.[5] Famous dictionaries, such as the Fourier basis and wavelets, have been successfully used to

*Code and supplemental material available from: `http://www.cs.unc.edu/~vjojic/amp`

decompose and denoise a variety of signals.[6] Algorithms for efficient discovery of sparse representations in such dictionaries have swept through the statistical, machine learning, signal processing, and computer vision communities.[7–9] The advent of locality sensitive hashing[10] and random projection[11] have additionally made the task of handling large datasets feasible, if not trivial. Indeed, the name of the game is random projection as any projection of the data seems to be informative. However, as we show, pure random projections do not always work efficiently.

Here we demonstrate how short-read metagenomic sequencing data can be decomposed into a sequence-based dictionary assembled on the fly. The dictionary contigs composed from reads prioritized by our simple probabilistic models turn out to be discriminative. Patient-specific dictionaries can then be merged together and processed to discover a short universal dictionary that is predictive of phenotype across a population. Finally, we contrast the performances of our Assembly Matching Pursuit algorithms with the performance of a standard Random Projection method[11] and the popular short-read assembler, SOAPdenovo.[12]

1.1. *Notation and primitive sequence operations*

We will denote the ℓ_2 norm as $\|x\|_2 = \sqrt{\sum_i x_i^2}$. Given a matrix D and an index set of its columns, I, we will use D_I to denote a matrix consisting of only those columns. Similarly, for a vector w and an index set of its coordinates I we will use w_I to denote a vector composed only of those coordinates. Finally, $D_{i,:}$ denotes the i^{th} row of D.

We define `kmers(Seq,k)` as the set of all k long contiguous substrings of `Seq`; we assume that k is set ahead of time and simply use `kmers(Seq)`.

We define `overlap(Seq,Kmers,m)` the subset of k-mers in the ordered set, `Kmers`, that overlaps with at least `m` letters of either terminus of the sequence, `Seq`. A k-mer is also included in this overlapping set if its reverse and complement overlaps with `Seq`. We assume that `overlap(EMPTY,Kmers,m)` returns `Kmers`, and that `overlap(Seq,K,m)` returns `EMPTY` when no k-mer in `K` overlaps with `Seq`.

We define `count(Kmers`$_i$`, Seq)` as the number of times the i^{th} k-mer $\in$ `Kmers` occurs in `Seq`.

We define `extend(Seq,Kmer)` to be the sequence constructed by appending `Kmer` – either as given or reversed and complemented – to the overlapping end of `Seq`. This is done by removing the overlapping segment of `Seq`, and concatenating the remaining part of `Seq` with `Kmer`. We assume that `extend(Seq,EMPTY) = Seq` and `extend(EMPTY,Kmer)= Kmer`.

1.2. *Dictionaries for metagenomic read sets*

We introduce a representation of the read sets in terms of dictionaries meant to capture the k-mer profile of the sample. Given k and bound on genome length of any given microbiome's member, S, we can construct an exponentially large matrix $D : 4^k \times N(S)$ defined as $D_{rs} =$ `count(`r`, `s`)`, where r is a k-mer and s is a sequence of length S. Here $N(S) = (4^{S+1}-4)/(4-1)$, the number of sequences with length at most S.

Given the dictionary matrix, D, and vector of abundances of sequences in the microbiome, w, we can describe the observed k-mer profile, y, as a noisy version of the true k-mer profile

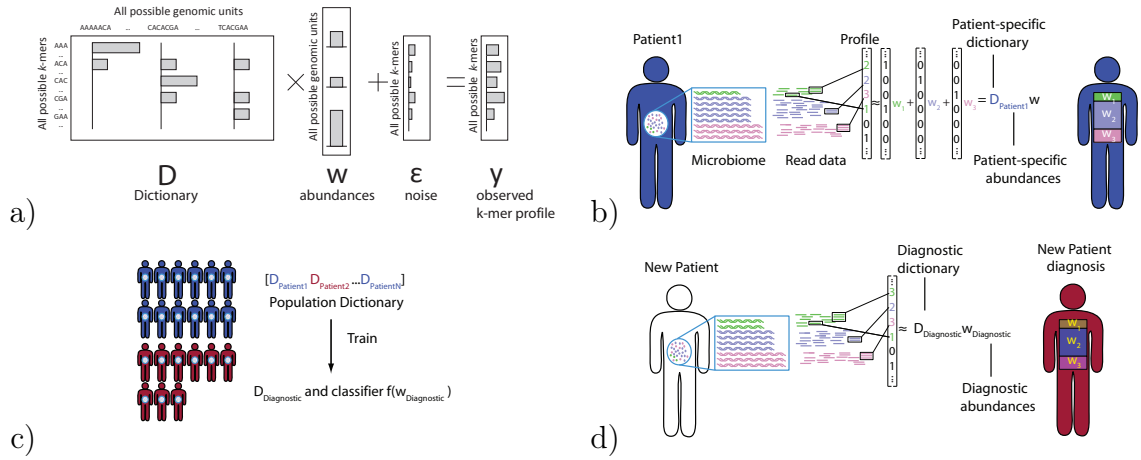

Fig. 1. a) A sketch of a generative model of a read set profile of a metagenomic set using an exponentially sized **super-dictionary**. b) Learning of a **patient-specific dictionary** from a single patient's data discovers dictionary elements that represent the most abundant MGUs. c) The patient-specific dictionaries from both healthy and sick patients are aggregated into a **population dictionary** and a set of dictionary elements predictive of phenotype are selected yielding a **diagnostic dictionary**. d) Prediction of a new patient's phenotype from abundances of diagnostic MGUs.

by noting that $y = Dw + \epsilon$, see Fig. 1a.

In order to disambiguate from the later dictionaries, we call this exponentially sized dictionary the **super-dictionary**. In fact, any MGU dictionary will be a subset of the super-dictionary.

2. Methods

2.1. *Dictionary hierarchy*

We will be constructing dictionaries that are subsets of the **super-dictionary**. A **patient-specific dictionary** is a set of MGUs used to represent a particular patient's k-mer profile. A **population dictionary** is an aggregate of patient-specific dictionaries meant to represent k-mer profiles of multiple patients. We note that the MGUs found in one patient may be representative of other patients' k-mer profiles. Finally, a **diagnostic dictionary** is a subset of the population dictionary that is relevant for predicting a phenotype.

2.2. *Matching pursuit and greedy algorithms*

The matching pursuit algorithm[13] finds a representation of a signal by greedily selecting dictionary elements that best explain the signal's residual (see Algorithm 3.1). In our case, the signal corresponds to the k-mer profile and dictionary elements correspond to MGU sequences.

The most relevant observation about the matching pursuit algorithms is that *each* dictionary element is examined in order to find the one that best correlates with the residual. Given the exponential size of the super-dictionary, the matching pursuit search requirement is not feasible in polynomial time. Therefore, we turn to the area of weak greedy algorithms, in which asymptotic convergence is guaranteed even if the chosen dictionary element in each iteration does not correlate optimally with the residual.[14] This permits the use of randomized

schemes that sample and accept dictionary elements if they explain a prespecified fraction of the residual signal.

The probabilistic matching pursuit (PMP) algorithm[15] leverages this intuition by probabilistically sampling dictionary elements. By avoiding an exhaustive search, PMP enables the use of large dictionaries; however, PMP methods do not explicitly optimize a likelihood, and sampling is restricted to conditioning on an element's correlation with the residual. We require a more flexible framework, and so present a generalized PMP (GPMP) algorithm (Algorithm 3.2). GPMP iteratively chooses dictionary elements that increase the likelihood of the data, $p(y|D, w)$, by sampling from a proposal distribution, $q(j|y, D, w)$.

3. Algorithm

Our patient-specific dictionary construction algorithm follows the GPMP framework. To specify the algorithm, we must select a likelihood $p(y|D, w)$ to optimize and proposal distribution $q(j|y, D, w)$ for sampling the dictionary.

3.1. *Likelihood*

The Gaussian distribution with fixed unit variance is a common choice of likelihood in matching pursuit applications,

$$\log p(y|D, w) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{n}(y_i - D_{i,:}w)^2. \tag{1}$$

This likelihood corresponds to linear regression, and its primary benefit is the computational efficiency with which it can be optimized.

A second choice of likelihood, corresponding to Poisson regression, is

$$\log p(y|D, w) = \sum_{i=1}^{n} y_i(D_{i,:}w) - \exp\{D_{i,:}w\} - \log(y_i!). \tag{2}$$

In contrast to linear regression, which treats both positive and negative observations, Poisson regression is meant to model non-negative data, such as read counts. Notably, the expected value and variance of a Poisson random variable are equal.

A third choice of likelihood, corresponding to Negative Binomial regression, is

$$\log p(y|D, w) = \sum_{i=1}^{n} y_i(D_{i,:}w) + \frac{1}{\alpha}\log(1 - \exp\{D_{i,:}w\}) + \log\frac{\Gamma(y_i + (1/\alpha))}{\Gamma(y_i + 1)\Gamma(1/\alpha)}. \tag{3}$$

Like Poisson count models, Negative Binomial models have been used to model non-negative, integral data; however, with the additional dispersion parameter α, they can model count data with less constricted mean-variance relationships.[16]

3.2. *Dictionary element proposal*

We wish to propose a sequence j whose k-mer profile is likely to increase the objective $\log p(y|D_{I\cup j}, w_{I\cup j})$ compared to $\log p(y|D_I, w_I)$. Given this goal we can easily construct a forward sampling algorithm that will produce a reasonable candidate sequence. Specifically, we

Algorithm 3.1. *Matching Pursuit*

Input: D, c
Output: w, *such that* $\|y - Dw\|_2 \leq c$
initialize $w_j = 0, \forall j$
while$(\|y - Dw\|_2 \leq c)$
  $R = y - Dw$
  $c_j = \left| \frac{\langle R, D_k \rangle}{\langle D_k, D_k \rangle} \right|, \forall j$
  $k = \text{argmax}_j \, c_j$
  $I = I \cup \{k\} \quad w_k = c_k$

Algorithm 3.2. *Generalized Probabilistic Matching Pursuit*

Input: D, y, c
Output: I *and* w *such that* $\log p(y|D, w) > c$
$I = \emptyset, w_j = 0, \forall j$
while$(\log p(y|D, w) \leq c)$
  sample $\{j\}$ *from* $q(j|y, D, w)$
  $I = I \cup j$
  $w_I = \text{argmax}_v \log p(y|D_I, v)$
return I, w

Algorithm 3.3. *Dictionary element proposal*

Input: $D, y, w, m,$ `Orthogonal`,
  set of all observed k*-mers* K
Output: *a candidate dictionary element* s
$s =$ `EMPTY`$, I = \{i | w_i \neq 0\}$
repeat
  $K_s =$ `overlap`$(s, K, m) \cup \{$`EMPTY`$\}$
  if(`Orthogonal`)
    $K_s = K_s - \cup_{i \in I}$ `kmers`(i)
  foreach$(l \in K_s)$
    $s' =$ `extend`(s,l)
    $I' = I \cup \{s'\}$
    $w_{I'} = \text{argmax}_w \log p(w|y, D_{I'})$
    $\pi_l = p(y|D_{I'}, w_{I'})$
  sample l^* *from normalized* π
  $s =$ `extend`(s, l^*)
until$(l^* =$ `EMPTY`$)$

---

initialize a new dictionary element (contig) by sampling a k-mer based on the increase in likelihood if that k-mer, alone, were to enter the model as an element. When extending a contig, an overlapping k-mer is sampled according to the change in likelihood that would result if it were added to the growing element. If no k-mer sufficiently improves the likelihood, then the algorithm may sample the `EMPTY` k-mer (i.e. choose to terminate extension), and thus complete an iteration. The likelihood biases the algorithm toward sampling k-mers that occur with high frequency.

Another proposal distribution produces orthogonal dictionary elements, and is accordingly used by orthogonal matching pursuit algorithms. Since the entries in our dictionary are always nonnegative, two dictionary elements will be orthogonal, $(\sum_k D_{k,i} D_{k,j} = 0)$, if and only if their corresponding MGUs do not share any k-mers. Algorithm 3.3 implements both of these choices specified by argument `Orthogonal`.

3.3. *The AMP algorithms*

We introduce four algorithms based on choices of likelihood and dictionary element proposal.

(1) Gaussian Assembly Matching Pursuit (GAMP) combines the likelihood from (1) and the non-orthogonal dictionary proposal.

(2) Poisson Assembly Matching Pursuit (PAMP) combines the likelihood from (2) and the non-orthogonal dictionary proposal.
(3) Negative Binomial Assembly Matching Pursuit (NAMP) combines the likelihood from (3) and the non-orthogonal dictionary proposal.
(4) Orthogonal Assembly Matching Pursuit (OAMP) uses the orthogonal dictionary proposal.

Because the orthogonal dictionary proposal strongly constrains dictionary construction, the choice of likelihood is irrelevant.

3.4. *Population dictionary construction and patient summarization*

Given a learned patient-specific dictionary we are tasked with constructing a dictionary that can be used universally across the whole patient population, the **population dictionary**. We can construct this dictionary by pooling all patient specific dictionaries, but here we face two challenges:

(1) How do we estimate abundances of the population dictionary elements in each patient?
(2) Which of the population dictionary elements are diagnostically relevant?

Abundance estimation A patient's k-mer profile may be regressed onto the population dictionary in order to estimate MGU abundances. We utilize Negative Binomial (NB) regression due to its flexibility in modeling potentially overdispersed data, such as read counts.[16] NB models have been fruitfully applied in RNA-Seq data analysis,[17] and we have found that abundances estimated by NB regression – regardless of dictionary origin – are more accurate than those estimated using other likelihoods (data not shown).

Using abundances as predictors MGU abundance estimates can be directly used as predictors of phenotype. In terms of interpretability, logistic regression is most appealing. In our experiments we use an efficient implementation of sparse logistic regression.[18] The sparsity inducing, ℓ_1-penalty selects only a small portion of the features to participate in phenotype prediction from an otherwise large population dictionary. Because the optimal scale, λ, of the ℓ_1-penalty is unknown, it must be estimated from the data. The data are therefore split into training, validation, and test sets – the validation set is used to determine the λ parameter. The sparsest model that classifies the validation set statistically as well as the best model is chosen. The classification accuracy of this logistic regression model is then evaluated on the test set.

The chosen set of MGUs that are predictive of phenotype correspond to parts of the population dictionary that can be diagnostically useful. Thus, they compose the **diagnostic dictionary**.

4. Implementation

We customized an implementation of a succinct suffix trie[19] to store suffix and prefix k-mer tries. The counts of each k-mer are also stored during trie construction. While the AMP algorithms' implementation is straightforward, here we draw attention to two issues relating

to likelihood optimization and read storage and querying.

The AMP assemblers are string-based and rely on greedy extension. However, extension is stochastic and is guided by the likelihood of the observed k-mer profile (Algorithms 3.2 & 3.3). When updating the weight of a growing contig to a conditional maximum likelihood value (i.e. computing $w_j = \text{argmax}_{v_j} \log p(y|D, v_j)$), GAMP equates the first partial derivative of the likelihood (with respect to the contig's weight) to zero and solves for w. NAMP and PAMP, on the other hand, utilize Newton-Raphson updates to find a w that maximizes the likelihood (a closed-form solution for w_I does not exist when equating the gradient of the negative binomial or Poisson likelihoods to zero).

5. Results and Discussion

To assess the ability of our methods to produce discriminative diagnostic dictionaries, we turned to synthetic and real data experiments. We put particular focus on the efficiency with which our AMP methods could produce representations relevant for phenotype prediction.[b]

In all synthetic experiments we worked with k-mers of fixed read length. The real dataset consisted of a mix of 75bp and 44bp read datasets. Hence we used k-mer length of 44bp, using shorter reads directly as k-mers. From each longer 75bp read we constructed 3 44-mers with 16bp spaced starting offsets. In our dictionary element proposal algorithm we required that a k-mer achieve an overlap of 20bp with a growing contig to be considered a candidate for appending.[c] Finally, to estimate the classification accuracy we performed a 10-fold cross-validation with an inner cross-validation on the training and validation sets to select λ (the held out data in the outer fold were not used during training).

5.1. *Baselines*

For comparision, we chose to analyze the quality of dictionaries produced by SOAPdenovo[12] and a pure random projection method.[11]

For synthetic experiments, SOAPdenovo was run on each sample using a single thread and a minimum k-mer overlap (option `-K`) of 21 for extension purposes. For the real data from Ref. 1, we used the SOAPdenovo contigs already generated in their paper.[d] Because SOAPdenovo's assembly is not likelihood driven, the order in which contigs are produced is not interpretable. Thus, the longest SOAPdenovo contigs with high coverage were added in a random fashion when evaluating successively larger population dictionaries.

Random projections (RP) summarize a set of points by projecting them into a lower dimensional subspace defined by a intelligently chosen, but random basis. If done properly, the relative distances between points before and after projection will be, on average, approximately preserved. In the case of metagenomic samples, we treat each sample's k-mer profile as a K dimensional point. Application of RP produces a new, smaller set of features that are sums

[b]Efficiency refers to the size of the population dictionary required to produce a diagnostic dictionary capable of achieving a particular classification accuracy.
[c]This amounts to using `m=20` in Algorithm 3.3.
[d]They used `-K` 21 and `-K` 23 for 44bp and 75bp reads, respectively.

of randomly weighted k-mer profiles, each with dimension $C < K$. If we have N samples then, $w^{\mathrm{RP}} = PY$ where Y is the $K \times N$ matrix of k-mer profiles from all samples, P is the $C \times K$ random projection matrix, and w^{RP} is the resulting $C \times N$ projected form of Y. These new C dimensional features are roughly akin to abundances produced by the AMP methods and are treated as such during our classification step. Indeed, an implicitly constructed dictionary matrix can be defined as matrix D that satisfies $P = (D^T D)^{-1} D^T$. The matrix P is constructed using the method described in Ref. 11 and refer to the algorithm as ARP.

5.2. *Synthetic data generation*

A/T SNP A 10Kb sequence was randomly generated and duplicated. The 5000th base in one duplicated copy was changed to an 'A' and the 5000th base in the other copy was changed to a 'T'. We then generated 100 synthetic metagenomic samples, 50 of which were phenotypically 'sick' and 50 of which were phenotypically 'healthy'. For each of the 100 samples, 20000 75bp reads with 2% noise were simulated from the 10Kb templates. A 50/50 and 33/67 ratio of the two variants were maintained for 'healthy' and 'sick' sample, respectively.

Distinct species For this synthetic experiment 40 10Kb sequences were randomly generated. From this true dictionary, we generated 100 synthetic metagenomic samples, 50 of which were phenotypically 'sick' and 50 of which were phenotypically 'healthy'. For each of the 100 samples, 40000 75bp reads with 3% noise were simulated from the 10Kb templates with varying coverage. Average baseline mixing proportions of the templates followed an exponential decay; however for 'sick' samples, the relative abundances of the 7th, 13th, and 24th most abundant templates were altered by 1%, 0.67%, and 0.33%, respectively (see Supp. Info. Fig. 1 for exact abundances).

Synthetic community The Genome Institute at the Washington School of Medicine has produced many draft-quality genomes of various human gut microbes.[e] We selected 31 microbial species' genomes to represent actual genera and phyla found in the human gut. From this true dictionary, we generated 100 samples, 50 'healthy' and 50 'sick', each with 10 million, 75bp reads with 3% noise. Baseline mixing proportions of each microbe in all samples were set in accordance with relative abundances reported in Ref. 1 and Ref. 20 based on the genus and phylum they represent; however, the relative abundances of 3 microbes in 'sick' samples were altered by 1%, 2%, and 3% (see Supp. Info. Fig. 2 for exact abundances).

5.3. *Synthetic data results*

Fig. 2 shows the performance of the methods in classifying 'healthy' and 'sick' samples from the synthetic experiments.

In the A/T SNP experiment, the discriminative abundances are driven by the reads spanning the SNP position. Without noise, all methods converge quickly and classification is trivial for SOAPdenovo and OAMP as orthogonality requirements ensure that none of the shared

[e]Freely available from `http://genome.wustl.edu/pub/organism/Microbes/Human_Gut_Microbiome/`.

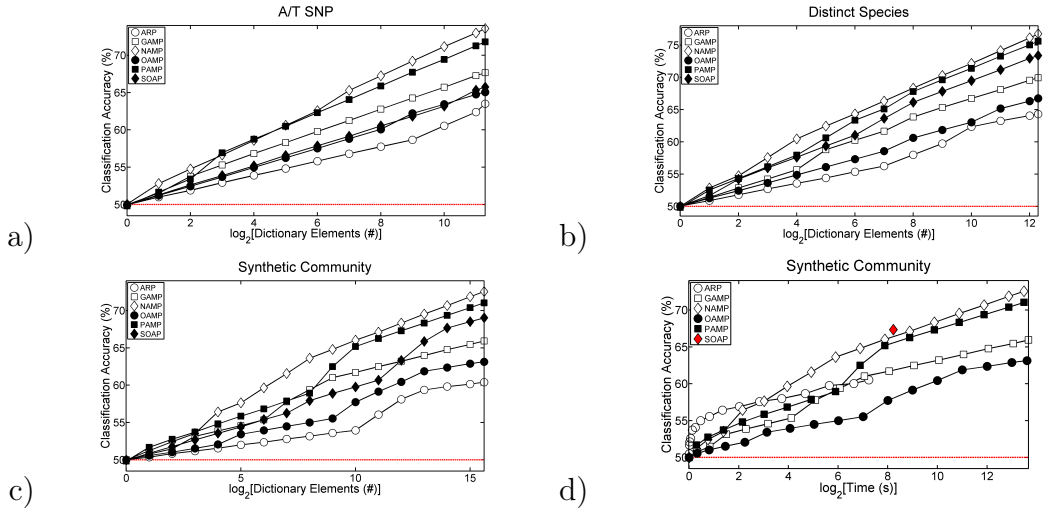

Fig. 2. a,b,c) Mean classification accuracies of the methods on synthetic datasets. The dashed red line corresponds the performance expected when always predicting 'healthy'. Plots with confidence bands around the mean can be seen in Supp. Info. Fig. 3. d) Performance comparison with respect to running time of each method on the synthetic community experiment. SOAPdenovo performance is marked with a single red diamond since the order in which it produces contigs is not interpretable.

reads between species are available for the second contig (data not shown). However, in a more realistic, noisy setting contig construction is more difficult. Nevertheless, NAMP, PAMP and GAMP begin to discover sequences containing the discriminative SNP within the first 8 contigs, before the other methods.

In the distinct species experiment, species' k-mer profiles are nearly orthogonal. Without noise, OAMP and SOAPdenovo reconstruct the true dictionary within the first 40 iterations (data not shown). With noise, OAMP spends more iterations constructing subcontigs of the true dictionary elements that are non-discriminitave. By exploring only well-supported edges in its De Bruijn graph construction, SOAPdenovo better handles the noise, constructs longer contigs, and thereby discovers significant features more quickly.

Interestingly, in all scenarios, including the synthetic community, there is a steady and consistent difference in performance between GAMP, PAMP, and NAMP. This illustrates clear ordering between the three choices of likelihood: NAMP $\succ$ PAMP $\succ$ GAMP. Additionally, NAMP and PAMP discover discriminative features sooner than the other methods,[f] and with the exception of the A/T SNP experiment, SOAPdenovo consistently outperforms GAMP. These results suggest that given the appropriate read count model, likelihood driven assembly can direct the early discovery of predictive features. Finally, the subpar performance of ARP on all experiments demonstrates the benefit of computing abundances of sensible contigs in a manner consistent with the nature of the data.

[f]This comparison is not directly applicable to SOAPdenovo as its assembly is not order dependent.

5.4. *Human Gut Metagenome Analysis*

In addition to synthetic experiments, we tested our method on data from Ref. 1. This data set contains 576 gigabases of sequence data obtained from the fecal samples of 124 Spanish and Danish individuals, 25 of whom have inflammatory bowel syndrome (IBD). Population dictionary pools for the AMP methods were constructed by aggregating the first 1000 dictionary contigs greater than 500 bp of each patient. For SOAPdenovo's pool we took the longest 124000 contigs of the roughly 6.6 million contigs greater than 500 bp produced by SOAPdenovo in Ref. 1. ARP's pool was constructed by producing 1000 random projections for each of the 124 patients, since ARP does not have a concept of a patient specific dictionary. From each of their respective pools, successively larger population dictionaries were constructed in order to evaluate each method's classification accuracy. Length distributions for the contigs used in each population dictionary can be found in Supp. Info. Fig. 4.

Fig. 3a) shows the method performances in classifying individuals based on their health status (IBD or healthy). We see that all methods discover relevant contigs, but at a different rate. The leading algorithm is NAMP, followed closely by PAMP, and thereafter SOAPdenovo. We see that GAMP and OAMP are relatively close in terms of performance but for different reasons. The OAMP is affected by the orthogonality requirement while Gaussian likelihood is overly greedy, driven by the quadratic cost on the residual.

From the final GAMP, SOAPdenovo, PAMP, and NAMP dictionaries, 11, 18, 18, and 19 metagenomic units, respectively, were found to have non-zero weight, suggesting their importance as potential biomarkers for IBD (Fig. 3b)). We obtained KO (KEGG orthologous groups) numbers for each of these features using KAAS, an annotation server that queries the KEGG database.[21] For discovered enzymes we additionally mined the KEGG BRITE database to obtain a functional annotation. Finally, as a measure of consistency between our method and an independent biological study, we noted any commonalities between our annotations and those of Ref. 22 (see Supp. Info., Fig. 5). Of our 48 features, 10 were found to be either enriched or depleted in the Ref. 22 analysis. In particular, 4 were related to the PTS, a system important for sugar transport into the cell and recently found to include biomarkers for IBD.[23] We additionally found nitrate reductase among our significant features. Nitrate reductase plays an important role in the conversion of nitrate to nitrite and nitric oxide, neither of which can be synthesized by human DNA. Unsurprisingly then, elevated levels of nitric oxide have been found to correlate with IBD.[24] Finally, we noted the presence of vanillate monooxygenase, an agent that may play a role in xenobiotic degredation of phenolic compounds, such as p-cresol, another correlate of IBD.[25]

5.5. *Time and memory*

Fig. 2a,b,c) and 3 describe the efficiency of the various methods in terms of accuracy gained per added dictionary element. To gauge computational efficiency, it is important to consider efficiency with respect to running time.

Fig. 2d) illustrates the performance of the various methods with respect to time in the synthetic community experiment and corresponds with the accuracies depicted in Fig. 2c). For the AMP methods, time required to achieve a particular classification accuracy was calculated

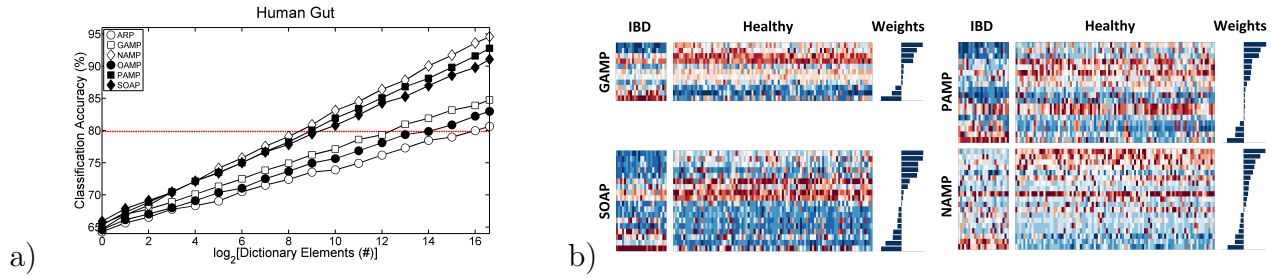

Fig. 3. a) Mean classification accuracy of each method on the real dataset. Dashed red line is the performance expected by always predicting 'healthy'. Plots with confidence bands around the mean can be seen in Supp. Info. Fig. 3. b)Most predictive dictionary element abundances for the healthy and sick patients stemming from the different methods (GAMP, SOAP, PAMP, NAMP) as well as weights of these abundances in a sparse logistic regression trained model.

as the sum total of the times required to generate each contig used in the corresponding population dictionary. For the ARP the performance curve is parameterized by the number of rows in the projection matrix. Each successive point on the AMP and ARP curves corresponds to a two-fold increase in the number of contigs over the previous point. For SOAPdenovo we measured the total running time required to assemble all synthetic samples (2.99×10^4 seconds) and noted how many contigs were produced (5.00×10^6). Thus, we extrapolated the time required to generate a final population dictionary of size 50000 to be $50000 \div (5.00 \times 10^6) \times (2.99 \times 10^4) = 299$ seconds. SOAPdenovo's performance is depicted as a single point since the contigs it produces are not necessarily order dependent.

SOAPdenovo reaches its final 67% accuracy before NAMP and PAMP, and handily outperforms GAMP and OAMP. The AMP methods are not as time efficient due to the expensive floating point arithmetic (e.g. computing exponents and logarithms) associated with the likelihood computations. However, NAMP and PAMP offset these inefficiencies by nearly reaching the same accuracy as SOAPdenovo in the same time and with a dictionary $1/25^{th}$ the size. Ultimately, with equally large dictionaries as SOAPdenovo, NAMP and PAMP provide superior performance by classifying 4-5% more accurately.

The AMP methods additionally require less memory than SOAPdenovo. On average, SOAPdenovo requires 2358 bytes per 75 bp read, whereas the AMP methods require 2037 bytes per 75 bp read (Supp. Info. Fig. 6). These reads were taken from the synthetic community experiment.

6. Conclusion

We introduced the Assembly Matching Pursuit family of methods for metagenomic dataset summarization and analysis. Our AMP methods follow a novel generalized matching pursuit paradigm, which guides dictionary construction using likelihood based principles. Within this framework, we explored the appropriateness of popular likelihood choices for modeling read counts and accordingly derived the GAMP, PAMP, and NAMP assemblers. In investigating an alternative proposal distribution, we derived the OAMP assembler, which enforces orthogonality among its contigs.

We also introduced a simple abundance estimation protocol that directly regresses k-mer

profiles of any read sample on a set of dictionary sequences. Indeed, a dictionary does not have to be composed of contigs from our AMP methods. It may generated by SOAPdenovo, any another assembler, or in the future, set to be a large sequence database.

By coupling AMP assembly with a negative binomial based abundance estimator, we have put forth a simple method of aggregating sample dictionaries into a population dictionary from which learned abundances can be leveraged as predictors of phenotype. In both synthetic and real datasets we show that this new family of methods does significantly better in phenotype discrimination than random projections. Further, due to their simplicity, the methods easily handle large scale datasets, such as in Ref. 1, which spans 0.6 terabases. Finally, while we focused on medical applications as an illustration, the method is applicable to other metagenomic, and in principle, RNA-seq studies.

References

1. J. Qin, R. Li, J. Raes, M. Arumugam *et al.*, *Nature* **464**, 59 (March 2010).
2. P. Hugenholtz, B. M. Goebel and N. R. Pace, *J. Bacteriol.* **180**, 4765 (Sep 1998).
3. C. Burke, P. Steinberg, D. Rusch, S. Kjelleberg and T. Thomas, *Proc. Natl. Acad. Sci. U.S.A.* **108**, 14288 (Aug 2011).
4. D. Knights, L. Parfrey, J. Zaneveld, C. Lozupone and R. Knight, *Cell Host & Microbe* **10**, 292 (October 2011).
5. J. H. Friedman and J. W. Tukey, *IEEE Trans. Comput.* **23**, 881 (1974).
6. S. Mallat, *A wavelet tour of signal processing*Wavelet Analysis and Its Applications Series, Wavelet Analysis and Its Applications Series (Academic Press, 1999).
7. D. L. Donoho, *IEEE Transactions on Information Theory* **41**, 613 (1995).
8. E. J. Candès and T. Tao, *IEEE Transactions on Information Theory* **52**, 5406 (2006).
9. J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang and S. Yan, *Proceedings of the IEEE* **98**, 1031 (June 2010).
10. A. Gionis, P. Indyk and R. Motwani, Similarity search in high dimensions via hashing1997.
11. D. Achlioptas, *Journal of Computer and System Sciences* **66**, 671 (June 2003).
12. R. Li, H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G. Shan, K. Kristiansen, S. Li, H. Yang, J. Wang and J. Wang, *Genome Res.* **20**, 265 (Feb 2010).
13. S. G. Mallat and Z. Zhang, *IEEE Trans. Signal Process.* **41**, 3397 (1993).
14. V. Temlyakov, *Greedy Approximation* (Cambridge University Press, 2011).
15. S. E. Ferrando, E. J. Doolittle, A. J. Bernal and L. J. Bernal, *Signal Processing* **80**, 2099 (October 2000).
16. J. Hilbe, *Negative Binomial Regression*, 2nd edn. (Cambridge University Press, 2011).
17. S. Anders and W. Huber, *Genome Biol.* **11**, p. R106 (2010).
18. R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang and C.-J. Lin, *Journal of Machine Learning Research* **9**, 1871 (2008).
19. D. Okanohara, ux-trie `http://code.google.com/p/ux-trie`, (2012).
20. M. Arumugam, J. Raes, E. Pelletier *et al.*, *Nature* **473**, 174 (May 2011).
21. Y. Moriya, M. Itoh, S. Okuda, A. C. Yoshizawa and M. Kanehisa, *Access* **35**, 182 (2007).
22. S. Greenblum, P. J. Turnbaugh and E. Borenstein, *Proc. Natl. Acad. Sci. U.S.A.* **109**, 594 (Jan 2012).
23. A. L. Francl, T. Thongaram and M. J. Miller, *BMC Microbiology* (2010).
24. G. Kolios, V. Valatas and S. G. Ward, *Immunology* (2004).
25. M. H. van Nuenen, K. Venema, J. van der Woude and E. J. Kuipers, *Digestive Diseases* **49**, 485 (2004).