Sifting Common Information from Many Variables

Greg Ver Steeg Shuyang Gao Kyle Reing Aram Galstyan
University of Southern California, Information Sciences Institute
gregv@isi.edu, gaos@usc.edu, reing@usc.edu, galstyan@isi.edu

Abstract

Measuring the relationship between any two variables is a rich and active area of research at the core of the scientific enterprise. In contrast, characterizing the common information among a group of observed variables has remained a speculative undertaking producing no practical methods for high-dimensional data. A promising solution would be a multivariate generalization of the famous Wyner common information, but this approach relies on solving an apparently intractable optimization problem. We formulate an incremental version of this problem called the information sieve that not only admits a simple fixed-point solution, but also empirically exhibits an exponential rate of convergence. We use this scalable method to demonstrate that common information is a useful concept for machine learning. The sieve outperforms standard methods on dimensionality reduction tasks, solves a blind source separation problem involving Gaussian sources that cannot be solved with ICA, and accurately recovers structure in brain imaging data.

1 Introduction

One of the most fundamental measures of the relationship between two random variables, $X_1, X_2$, is given by the mutual information, $I(X_1; X_2)$. While mutual information measures the strength of a relationship, the “common information” provides a concrete representation, $Y$, of the information that is shared between two variables. According to Wyner [33], if $Y$ contains the common information between $X_1, X_2$, then we should have $I(X_1; X_2 | Y) = 0$, i.e., $Y$ makes the variables conditionally independent. The challenge is to find the most succinct representation of $Y$ that has this property. We can extend this idea to many variables using the multivariate generalization of mutual information called total correlation [32], so that conditional independence is equivalent to the condition $TC(X_1, \ldots, X_n | Y) = 0$. Common information has many potential applications for distributed source coding and cryptography [17]. Here we suggest that common information also has promising applications in machine learning which have been overlooked due to the intractability of recovering common information in general.

Our main contribution is a method to iteratively extract common information between a potentially large number of variables, culminating in the decomposition described in Cor. 3.2. This decomposition proceeds by searching for a single latent factor that reduces the conditional dependence as much as possible. Then the data is transformed to remove this dependence and the “remainder information” trickles down to the next layer. The process is repeated until all the common information has been extracted and the remainder contains nothing but independent noise. Our second main contribution is to show that under the assumptions of linearity and Gaussianity this optimization has a simple fixed-point solution (Eq. 8) whose computational complexity is linear in the number of variables and which empirically converges to a local optimum at an exponential rate. Non-linearity and non-Gaussianity can be accommodated in a principled way. Furthermore, our approach is manifestly invariant to the scale of the data.

Our final contribution is to use these new computationally efficient tools to show that the principle of common information extraction is useful for a variety of machine learning problems. While PCA
finds components that explain the most variation, the sieve discovers components that explain the most dependence and we show that this is useful for exploratory data analysis. Common information can be used to solve a natural class of blind source separation problems that are impossible to solve using independent component analysis (ICA) due to the presence of Gaussian sources. Finally, we show that common information outperforms standard approaches for dimensionality reduction and recovers meaningful structure in fMRI data.

2 Preliminaries

Using standard notation \[7\], capital \(X_i\) denotes a continuous random variable whose instances are denoted in lowercase, \(x_i\). We abbreviate multivariate random variables, \(X \equiv X_1, n \equiv X_1, \ldots, X_n\), with an associated probability density function, \(p_X(X_1 = x_1, \ldots, X_n = x_n)\), which is typically abbreviated to \(p(x)\), with vectors in bold. We will index different groups of multivariate random variables with superscripts, \(X^k\), as defined in Fig.1. We let \(X^0\) denote the original observed variables and we omit the superscript for readability when no confusion results.

Differential entropy is defined as \(H(X) \equiv \langle \log 1/p(x) \rangle\), where we use brackets for expectation values. We use natural logarithms so that the units of information are "nats". The mutual information (MI) between two groups of random variables, \(X^k\) and \(Y\), can be written in terms of entropy as the reduction of uncertainty in one variable, given information about the other, \(I(X; Y) \equiv H(X) − H(X|Y)\). Multivariate mutual information, or total correlation, is defined as follows.

\[
TC(X) \equiv D_{KL} \left( p(x) \bigg\| \prod_{i=1}^{n} p(x_i) \right) = \sum_{i=1}^{n} H(X_i) − H(X)
\]

This quantity is non-negative and zero if and only if all the \(X_i\)’s are independent. We can also define a conditional TC, after conditioning on some other single factor, \(Y\), as \(TC(X|Y) \equiv D_{KL}(p(x|y)||\prod_{i=1}^{n} p(x_i|y))\). If \(Y\) were the hidden source of all dependence in \(X\), then \(TC(X|Y) = 0\). Therefore, we consider the problem of searching for a factor \(Y\) that minimizes \(TC(X|Y)\). Equivalently, we can define the reduction in TC after conditioning on \(Y\) as follows.

\[
TC(X; Y) \equiv TC(X) − TC(X|Y) = \sum_{i=1}^{n} I(X_i; Y) − I(X; Y).
\]

We refer to \(TC(X; Y)\) as the amount of total correlation in \(X\) that is explained by \(Y\). Next, we demonstrate a way to incrementally decompose \(TC(X)\) in terms of a sum of non-negative contributions coming from different latent factors, \(Y_1, \ldots, Y_r\).

3 Linear sieve decomposition to extract common information

For \(Y\) to contain the common information in \(X\), we need \(TC(X|Y) = 0\). Following Wyner [33], we would like find the smallest set of latent factors that satisfy this condition. To that end, we begin with a single latent factor and optimize it to minimize \(TC(X|Y)\) [19]. This optimization can be written equivalently as follows.

\[
\max_{y=f(x)} \ TC(X; Y)
\]

For now, assume that we have an efficient way to solve this optimization problem for a single latent factor, \(Y\). We would like to leverage this solution to iteratively extract more and more of the common information in \(X\). We now introduce such a scheme.

**Incremental decomposition** We begin with some input data, \(X\), and then construct \(Y_1\) to minimize \(TC(X|Y_1)\) or, equivalently, maximize \(TC(X; Y_1)\). After doing so, we would like to transform the original data into the remainder information, \(X^1\), so that we can use the same optimization to learn a factor, \(Y_2\), that extracts more common information that was not already captured by \(Y_1\). We diagram this construction at layer \(k\) in Fig.1 and show in Thm [5.1] the requirements for constructing the remainder information. The result of this procedure is encapsulated in Cor. [5.2] which says that we can iteratively apply this procedure and \(TC(X|Y_1, \ldots, Y_k)\) will be reduced at each layer until it reaches zero and \(Y\) captures all the common information.
A practical problem immediately arises, which is that if we take, e.g., Theorem 3.1. Incremental decomposition of common information

\[ TC(X^{k-1}) = TC(X^k) + TC(X^{k-1}; Y_k), \]

as long as the remainder information \( X^k \) satisfies two properties.

1. Invertibility: there exist functions \( g, h \) so that \( x_i^{k-1} = g(x_i^k, y) \) and \( x_i^k = h(x_i^{k-1}, y_k) \)
2. Remainder contains no information about \( Y_k \): \( \forall i, I(X_i^k; Y_k) = 0 \)

The decomposition above was originally introduced for discrete variables as the “information sieve” [30]; the continuous formulation we introduce here differs in the conditions required and the proof for this new version is in Sec. [3]. Note that because we can always find non-negative solutions for \( TC(X^{k-1}; Y_k) \), it must be that \( TC(X^k) \leq TC(X^{k-1}) \). In other words, the remainder information is more independent than the input data. This is consistent with the intuition that the sieve is sifting out the common information at each layer.

Corollary 3.2. Iterative decomposition of common information We construct a hierarchical representation where each \( Y_k \) is a function of \( X^{k-1} \) and \( X^k \) is the remainder information as defined in Thm 3.1.

\[ TC(X) = TC(X^r) + \sum_{k=1}^{r} TC(X^{k-1}; Y_k) \]

\[ TC(X|Y_1, r) \leq TC(X^r) = TC(X) - \sum_{k=1}^{r} TC(X^{k-1}; Y_k) \]

Eq. 3 follows from repeated application of Eq. 4. \( TC(X) \) is a constant that depends on the data. For high-dimensional data, it can be very difficult to estimate, but by learning latent factors extracting progressively more common information, we get better bounds. The second line, Eq. 6 is proved in Sec. 4 and shows that each \( Y_k \) that we add progressively reduces the conditional dependence in \( X \) from \( TC(X) \) down to zero. In other words, we keep adding and optimizing latent factors until they contain all the common information so that the observations are conditionally independent.

Optimization It follows from Eq. 5 that to extract more common information, we simply have to solve a series of optimization problems of the form of Eq. 4. We now introduce a tractable version of this optimization. We consider the case where \( Y \) is a linear function of \( X \), i.e. \( y = w \cdot x \).

Due to a special invariance of mutual information [7], this objective is unchanged if we instead let \( y = \sigma(w \cdot x) \), where \( \sigma \) represents a smooth invertible transformation (as is the case for many neural network formulations). Therefore, the nonlinearity is irrelevant and we focus on the linear case.

Next, we consider a further simplification where \( X \) is a multivariate normal distribution and therefore the joint distribution over \( X \) and \( Y \) is also normal. W.l.o.g., we assume the data is drawn from \( \mathcal{N}(0, \Sigma_X) \). Estimating the covariance matrix for large \( n \) is itself a challenging research problem, but, luckily, in our approach this will ultimately not be required.

A practical problem immediately arises, which is that if we take, e.g., \( Y = X_i \), then, formally the mutual informations \( I(X_i; Y) \) and \( I(X; Y) \) diverge and we are confronted with subtracting two infinities in our objective. This reflects the fact that these continuous factors may have infinitely many bits of precision. We know this to be unphysical, so strictly for optimization purposes we add a small amount of noise to \( Y \). We replace the appropriate expressions for Gaussian distributions in Eq. 3 using Eq. 2 to get the following.

\[ \max_{y = w \cdot x + \epsilon} -\frac{1}{2} \sum_i \log \left( 1 - \frac{\langle X_i Y \rangle^2}{\langle X_i^2 \rangle \langle Y^2 \rangle} \right) - \frac{1}{2} \log \frac{\langle Y^2 \rangle}{\eta^2} \]

Here \( \epsilon \) is a Gaussian random variable with constant variance \( \eta^2 \). The value of \( \eta \) is arbitrary, however, it sets the scale of \( Y \) (otherwise \( Y \) would be invariant under scaling and could, in principle be very
large or small with no effect on the objective). Next, we take the derivative with respect to \( w_j \). After some tedious manipulations (see Sec. 5) we arrive at the following fixed-point expression:

\[
  w_j = \eta^2 \frac{\langle X_j Y \rangle}{\langle X_j^2 \rangle \langle Y^2 \rangle - \langle X_j Y \rangle^2}
\]

Interestingly, we arrive at a novel nonlinear twist on the classic Hebbian learning rule [2]. If \( X_j \) and \( Y \) “fire together they wire together”, but this objective strongly prefers correlations that are nearly maximal, in which case the denominator becomes small and the weight becomes large. This optimization of \( TC(X; Y) \) for continuous random variables \( X \) and \( Y \) is, to the best of our knowledge, the first tractable solution except for a special case discussed by Op’t Veld and Gastpar [19] where a \( Y \) exists so that \( TC(X|Y) = 0 \).

A final consideration is the construction of remainder information consistent with the requirements in Thm. 3.1. In the discrete formulation of the sieve, constructing remainder information is a major problem that ultimately imposes a bottleneck on its usefulness because the state space of remainder information can grow quickly. In the linear case, however, the construction of remainder information is a simple linear transformation reminiscent of incremental PCA. We define the remainder information with a linear transformation, \( X^*_k = X^*_{k-1} - \langle X^*_{k-1} Y_k \rangle / \langle Y_k^2 \rangle Y_k \). This transformation is clearly invertible, and it can be checked that \( \langle X^*_k Y_k \rangle = 0 \) which implies \( I(X^*_k; Y_k) = 0 \).

**Generalizing to the non-Gaussian, nonlinear case** While the optimization of \( TC(X; Y) \) for the linear, Gaussian case has a solution in Eq. 8 it is not immediately clear that it will be useful for real-world data. To build this case, we first point out that we do not actually have to require that the data, \( X \), is drawn from a joint normal distribution to get meaningful results. It turns out that if each of the individual marginals is Gaussian, then the expression for mutual information for Gaussians provides a lower bound for mutual information [9]. As we pointed out, the objective is invariant under invertible transformations of the marginals [7]. Therefore, to ensure that the optimization that we solved (Eq. 7) is a lower bound for the optimization of interest, Eq. 3, we should transform the marginals to be individually Gaussian distributed. Several parametric transformations to Gaussianize one-dimensional data exist, including a recent method that works well for long-tailed data [11]. Alternatively, the brute force solution is to empirically Gaussianize the data based on the rank [26].

A problem remains; we argued that as long as the marginals are Gaussian, we can use the Gaussian MI as a lower bound for the true MI. However, if \( Y \) is a linear function of \( X \) and the marginals, \( X_i \), are (individually but not jointly) Gaussian, then this does not guarantee that \( Y \) is also Gaussian. Generally, we ignore this issue and take Eq. 7 as our optimization problem. However, in principle, one could introduce a Gaussianizing nonlinearity for \( Y \) as part of the optimization. Alternatively, we could follow ICA [16] and add a term to the optimization that models the non-Gaussianity of \( Y \). Despite ignoring this key ingredient of ICA, we show in Sec. 5 that the sieve outperforms ICA for some blind source separation problems.

### 4 Implementation details

**A single layer** A concrete implementation of one layer of the sieve transformation is straightforward and the algorithm is summarized in Alg. 1 in Supplementary Material (SM). Our implementation is available online [27]. The minimal preprocessing of the data is to subtract the mean of each variable. Optionally, further Gaussianizing preprocessing can be applied. Our fixed point optimization requires us to start with some weights, \( w^0 \) and we iteratively update \( w \) using Eq. 8 until we reach a fixed point. This only guarantees that we find a local optima so we typically run the optimization 10 times and take the solution with the highest value of the objective. We initialize \( w^0 \) to be drawn from a normal with zero mean and scale \( \eta / (\sqrt{n} \sigma_x) \). The scale of \( w^0 \) is set by \( \eta \) and we also scale each \( w^0 \) by the standard deviation of each marginal so that one variable does not strongly dominate the random initialization, \( y = w^0 \cdot x \).

The iteration proceeds by estimating marginals and then applying Eq. 8. We do not actually add noise when calculating \( y \), because \( \langle X_i Y \rangle \) does not depend on the noise and we can analytically correct the variance of \( Y \), \( \langle Y^2 \rangle = (\sum_i w_i x_i^2) + \eta^2 \). Estimating the covariance at each step is the main computational burden, but the steps are all linear. If we have \( N \) samples and \( n \) variables, then we calculate labels for each data point, \( y = w \cdot x \), which amounts to \( N \) dot products of vectors with length \( n \). Then we calculate the covariance, \( \langle X_i Y \rangle \), which amounts to \( n \) dot products of vectors of
length $N$. These are the most intensive steps and could be easily sped up using GPUs or mini-batches if $N$ is large. Convergence is determined by checking when changes in the objective of Eq.\[\text{[7]}\] fall below a certain threshold, we used $10^{-8}$ in our experiments.

**Multiple layers** After training one layer of the sieve, it is trivial to take the remainder information and feed it again through Alg.\[\text{[1]}\] Each layer contributes $TC(X^{k-1}; Y_k)$ in our decomposition of $TC(X)$, so we can stop when these contributions become negligible. This occurs when the variables in $X^k$ become independent. In that case, $TC(X|Y_1:k) = TC(X^k) = 0$ and since $TC(X^k) \geq TC(X^k; Y_{k+1})$, we get no more positive contributions from optimizing $TC(X^k; Y_{k+1})$.

## 5 Results

We begin with some benchmark results on a synthetic model. We use this model to show that the sieve can uniquely recover the hidden sources, while other methods fail to do so.

![Diagram](image.png)

**Figure 2:** (Left) This is the generative model used for synthetic experiments. Each independent source, $Z_i$, is drawn from a unit normal distribution. Each observed variable, $X_i = Z_j + \epsilon_i$, combines its parent with additive white Gaussian noise. (Right) A sample of data where rows are samples and columns are $X_i$’s generated according to the model on the left with $m = 1, k = 100, C = 50, N = 50$, and then transformed using the sieve. The noise magnitude is not uniform so columns are standardized for visualization.

**Data generating model** For the synthetic examples, we consider data generated according to a model defined in Fig.\[\text{[2]}\]. We have $m$ sources, each with unit variance, $Z_i \sim \mathcal{N}(0, 1)$. Each source has $k$ children and the children are not overlapping. Each channel is an additive white Gaussian noise (AWGN) channel defined as $X_i = Z_j + \epsilon_i$. The noise has some variance that may be different for each observed variable, $\epsilon_i \sim \mathcal{N}(0, \epsilon_i^2)$. Each channel can be characterized as having a capacity, $C_i = 1/2 \log(1 + 1/\epsilon_i^2)$ \[\text{[7]}\], and we define the total capacity for each source, $C = \sum_{i=1}^{k} C_i$. For experiments, we set $C$ to be some constant, and we set the noise so that the fraction, $C_i/C$, allocated to each variable, $X_i$, is drawn from the uniform distribution over the probability simplex. Note that this capacity only gives an upper bound on performance. It may not be achievable because we also have to infer the relationships between variables and sources.

**Empirical convergence rates** We examine how quickly the objective converges by plotting the error at the $t$-th iteration. The error is defined as the difference between $TC$ at each iteration and the final $TC$. We take the final value of $TC$ to be the value obtained when the magnitude of changes falls below $10^{-14}$. We set $C = 1$ for these experiments. In Fig.\[\text{[3]a}\], we look at convergence for a few different settings of the generative model and see exponential rates of convergence, with a coefficient that seems to depend on problem details. The slowest convergence (though still exponential) comes from data where each $X_i$ is generated from an independent normal distribution (i.e., there is no common information). We get a more nuanced view in Fig.\[\text{[3]b}\] from looking at some real world datasets that are described in detail in Sec.\[\text{[A]}\]. While the convergence rate still looks exponential, the rate is slower. Also, we see oscillation in the error (which appears as banding on the plots).

**Recover a single source from common information** As a first test of performance, we consider a simple version of the model in Fig.\[\text{[2]}\] in which we have just a single source and we have $k$ observed variables that are noisy copies of the source. For this experiment, we set the capacity bound to $C = 4$ nats. By varying $k$, we are spreading this capacity across a larger number of noisier variables. We use the sieve to recover a single latent factor, $Y$, that captures as much of the dependence as possible (Eq.\[\text{[3]}\]), and then we test how close this factor is to the true source, $Z$. We also compare to various other standard methods described in Sec.\[\text{[A]}\]. The results are summarized in Fig.\[\text{[4]a}\].

This problem seems relatively simple, and for a small number of variables almost any technique suffices to recover the source. As the number of variables rises, however, intuitively reasonable methods fail and only the sieve maintains high performance. The first component of PCA, for instance, is really the projection with the largest variance. Because of the asymmetries in the noise model, this
5.1 Blind source separation based on common information

In the generative model in Fig. 2, we have $m$ independent sources that are each Gaussian distributed. We could imagine applying an orthonormal rotation, $R$, to the vector of sources and call these $\tilde{Z}_j = \sum_k R_{jk} Z_k$. Because of the Gaussianity of the original sources, $\tilde{Z}$ also represent $m$ independent Gaussian sources. We can write down an equivalent generative model for the $X_i$’s, but each $X_i$ now depends on all the $\tilde{Z}$ (i.e., $X_i = \sum_j R_{ij}^{-1} \tilde{Z}_j + \epsilon_i$). From a generative model perspective, our original model is unidentifiable and therefore independent component analysis cannot recover it \cite{8}. On the other hand, the original generating model is special because the common information about the $X_i$’s are localized in individual sources, while in the rotated model, you need to combine information from all the sources to predict any individual $X_i$. The sieve is able to uniquely recover the true sources because they represent the optimal way to sift out common information.

To measure our ability to recover the independent sources in our model, we consider a model with $m = 10$ sources and varying numbers of noisy observations. The results are shown in Fig. 4(b). We learn 10 layers of the sieve and check how well $Y_1, \ldots, Y_{10}$ recover the true sources. We also specify 10 components for the other methods shown for comparison. As predicted, ICA does not recover the independent sources. The data generating model conforms to the assumptions underlying Factor Analysis (FA), so its failure on such an intuitive example is also surprising. Again, there are many FA models that are equally good generative models of the data so it cannot pick out the original. In contrast, common information provides a simple and effective principle for uniquely identifying the most informative sources in this case.

\footnote{To some extent this can be characterized analytically by noting the first PCA component is the largest eigenvector of the covariance matrix, $\Sigma_X$. In this case, $\Sigma_X$ is a diagonal matrix with a rank-one perturbation and the eigenvectors of this type of system have been well-studied \cite{3}.}
Source separation in fMRI data  To demonstrate that our approach is practical for blind source separation in a more realistic scenario, we applied the sieve to recover spatial brain components from fMRI data. This data is generated according to a synthetic but biologically motivated model that incorporates realistic spatial modes and heterogeneous temporal signals [8]. We show in Fig. 5(b) that we recover components that match well with the true spatial components. For comparison, we show ICA’s performance in Fig. 5(c) which looks qualitatively worse. ICA’s poor performance for recovering spatial MRI components is known and various extensions have been proposed to remedy this [1]. This preliminary result suggests that the concept of “common information” may be a more useful starting point than “independent components” as an underlying principle for brain imaging analysis.

![Figure 5: Colors represent different spatial components. (a) The spatial map of 27 components used to generate fMRI data. (b) 27 spatial components recovered by the information sieve. (c) 27 spatial components recovered by ICA where components visualize the recovered mixing matrix.](image)

5.2 Dimensionality reduction and exploratory data analysis

The sieve can be viewed as a dimensionality reduction (DR) technique. Therefore, we apply various DR methods to two standard datasets and use a Support Vector Machine with a Gaussian kernel to compare the classification accuracy after dimensionality reduction. The two datasets we studied were GISETTE and MADELON and consist of 5000 and 500 dimensions respectively. Details about the datasets and the standard techniques used for comparison are in Sec. A. For each method and dataset, we learn a low-dimensional representation on training data and then transform held-out test data and report the classification accuracy on that. The results are summarized in Fig. 6.

![Figure 6: (a) Validation accuracy for GISETTE dataset (b) Validation accuracy for MADELON dataset. All the scores are averaged by running 20 trials.](image)

For the GISETTE dataset, we see factor analysis, the sieve, and PCA performing the best, producing low dimensional representations with similar quality using a relatively small number of dimensions. For the MADELON dataset, the sieve representation gives the best accuracy with factor analysis and PCA resulting in accuracy drops of about five and ten percent respectively. Interestingly, all three techniques peak at five dimensions, which was intended to be the correct number of latent factors embedded in this dataset [13].
The first component of PCA explains the most variance in the data. Analogously, the first component of the sieve extracts the largest source of common information first. The ranking of components for the sieve can be used in a similar way as PCA. In Sec. we compare the top components learned on the Olivetti faces dataset to those learned by PCA. We also show that we can recover good lossy approximations of the images based on the sieve’s low dimensional representation.

6 Related work

Although the sieve is linear, the information objective that is optimized is nonlinear so the sieve substantially differs from methods like PCA. Superficially, the sieve might seem related to methods like Canonical Correlation Analysis (CCA) that seek to find a $Y$ that makes $X$ and $Z$ independent, but that method requires some set of labels, $Z$. One possibility would be to make $Z$ a copy of $X$, so that $Y$ is reducing dependence between $X$ and a copy of itself. However, this objective differs from common information as can be seen by considering the case where $X$ consists of independent variables. In that case the common information within $X$ is zero, but $X$ and its copy still have dependence. The concept of “common information” has largely remained restricted to information-theoretic contexts. The common information in $X$ that is about some variable, $Z$, is called intersection information and is also an active area of research.

Insofar as the sieve reduces the dependence in the data, it can be seen as an alternate approach to independent component analysis that is more directly comparable to “least dependent component analysis”. As an information theoretic learning framework, the sieve could be compared to the information bottleneck, which also has an interesting Gaussian counterpart. The bottleneck requires labeled data to define its objective. In contrast, the sieve relies on an unsupervised objective that fits more closely into a recent program for decomposing information in high-dimensional data, except that work is restricted to discrete latent factors.

Finally, the sieve could be viewed as a new objective for projection pursuit based on common information. The sieve stands out from standard pursuit algorithms in two ways. First, an information based “orthogonality” criteria for subsequent projections naturally emerges and, second, new factors may depend on factors learned at previous layers (note that in Fig. each learned latent factor is included in the remainder information that is optimized over in the next step).

7 Conclusion

We introduced the information sieve for continuous variables, a new scheme for incrementally extracting common information from high-dimensional data. The foundation of the sieve method is an information theoretic optimization that finds latent factors that capture as much information about multivariate dependence in the data as possible. We showed for the first time that this optimization could be efficiently carried out for continuous variables. Not only is our method linear in the number of variables, empirically it converges to a fixed point at an exponential rate.

With a practical method for extracting common information from high-dimensional data in hand, we were able to explore new applications of common information in machine learning. Besides promising results for exploratory data analysis and dimensionality reduction, common information seems to provide a particularly compelling new approach to blind source separation.

While the results here relied on assumptions of linearity and Gaussianity, the invariance of the objective under nonlinear marginal transforms, a common ingredient in deep learning schemes, suggests a straightforward way to generalize these methods. We leave concrete implementations of the nonlinear sieve to future work. The greedy nature of the sieve construction may be a limitation so another potential direction would be to jointly optimize several latent factors at once. Sifting out common information in high-dimensional data seems to provide a powerful and practical paradigm for unsupervised learning and the sieve provides a promising tool for that purpose.

Acknowledgments

GV thanks Sanjoy Dasgupta, Lawrence Saul, and Yoav Freund for suggesting exploration of the linear, Gaussian case for decomposing multivariate information. This work was supported in part by DARPA grant W911NF-12-1-0034 and IARPA grant FA8750-15-C-0071.
References

[1] E. A. Allen, E. B. Erhardt, Y. Wei, T. Eichele, and V. D. Calhoun. Capturing inter-subject variability with group independent component analysis of fmri data: a simulation study. Neurorimage, 59(4), 2012.

[2] P. Baldi and P. Sadowski. The ebb and flow of deep learning: a theory of local learning. arXiv preprint arXiv:1506.06472, 2015.

[3] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen. Rank-one modification of the symmetric eigenproblem. Numerische Mathematik, 31(1):31–48, 1978.

[4] R. B. Cattell. Factor analysis: an introduction and manual for the psychologist and social scientist. 1952.

[5] G. Chechik, A. Globerson, N. Tishby, and Y. Weiss. Information bottleneck for gaussian variables. In Journal of Machine Learning Research, pages 165–188, 2005.

[6] P. Comon. Independent component analysis, a new concept? Signal processing, 36(3):287–314, 1994.

[7] T. M. Cover and J. A. Thomas. Elements of information theory. Wiley-Interscience, 2006.

[8] E. B. Erhardt, E. A. Allen, Y. Wei, T. Eichele, and V. D. Calhoun. Simtb, a simulation toolbox for fmri data under a model of spatiotemporal separability. Neurorimage, 59(4):4160–4167, 2012.

[9] D. V. Foster and P. Grassberger. Lower bounds on mutual information. Phys. Rev. E, 83(1):010101, 2011.

[10] J. H. Friedman. Exploratory projection pursuit. Journal of the American Stat. Assoc., 82(397), 1987.

[11] G. M. Goerg. The lambert way to gaussianize heavy-tailed data with the inverse of tukey’s h transformation as a special case. The Scientific World Journal: Special Issue on Probability and Statistics with Applications in Finance and Economics, 2014.

[12] V. Griffith, E. K. Chong, R. G. James, C. J. Ellison, and J. P. Crutchfield. Intersection information based on common randomness. Entropy, 16(4):1985–2000, 2014.

[13] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror. Result analysis of the nips 2003 feature selection challenge. In Advances in neural information processing systems, pages 545–552, 2004.

[14] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM review, 53(2):217–288, 2011.

[15] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, 313(5786):504–507, 2006.

[16] A. Hyyrinen and E. Oja. Independent component analysis: algorithms and applications. Neural networks, 13(4):411–430, 2000.

[17] G. R. Kumar, C. T. Li, and A. El Gamal. Exact common information. In Information Theory (ISIT), 2014 IEEE International Symposium on. IEEE, 2014.

[18] C.-J. Lin. Projected gradient methods for nonnegative matrix factorization. Neural computation, 19(10):2756–2779, 2007.

[19] G. Op’t Veld and M. C. Gastpar. Caching gaussians: Minimizing total correlation on the gray–wyner network. In 50th Annual Conference on Information Systems and Sciences (CISS), 2016.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. The Journal of Machine Learning Research, 12:2825–2830, 2011.

[21] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. Science, 290(5500):2323–2326, 2000.

[22] D. Sculley. Web-scale k-means clustering. In Proceedings of the 19th international conference on World wide web, pages 1177–1178. ACM, 2010.

[23] D. Sculley. Linear information sieve code. http://github.com/gregversteeg/LinearSieve.

[24] G. Ver Steeg and A. Galstyan. Discovering structure in high-dimensional data through correlation explanation. Advances in Neural Information Processing Systems (NIPS), 2014.

[25] G. Ver Steeg and A. Galstyan. Maximally informative hierarchical representations of high-dimensional data. In Artificial Intelligence and Statistics (AISTATS), 2015.

[26] G. Ver Steeg and A. Galstyan. The information sieve. In International Conference on Machine Learning (ICML), 2016.

[27] M. Wang, F. Sha, and M. I. Jordan. Unsupervised kernel dimension reduction. In Advances in Neural Information Processing Systems, pages 2379–2387, 2010.

[28] S. Watanabe. Information theoretical analysis of multivariate correlation. IBM Journal of research and development, 4(1):66–82, 1960.

[29] A. D. Wyner. The common information of two dependent random variables. Information Theory, IEEE Transactions on, 21(2):163–179, 1975.
Supplementary Material for “Sifting Common Information from Many Variables”

A Datasets and Methods

For completeness, we provide additional details about datasets and methods used in our experiments. We also provide open source code implementing the sieve [27]. Algorithm 1 gives pseudo-code describing one layer of the sieve.

Table 1 describes the datasets used. GISETTE and MADELON are the two largest data sets from the NIPS 2003 feature selection challenge [13]. GISETTE is based on handwritten digit data and the classification problem is to distinguish 4’s and 9’s. MADELON is an artificially constructed dataset that includes many distractor variables. The Olivetti faces dataset is a standard machine learning dataset consisting of grayscale images of faces of 40 subjects each in ten different poses.

Table 1: Dataset Statistics

| Dataset   | # Features | # Train | # Validation |
|-----------|------------|---------|--------------|
| GISETTE   | 5000       | 6000    | 1000         |
| MADELON   | 500        | 2000    | 600          |
| Olivetti  | 4096       | 400     |              |

We compared with a variety of dimensionality reduction methods. We choose the desired number of components and use each technique to produce an appropriately sized representation. In many cases, this entails a straightforward application of standard techniques including Principle Component Analysis (PCA) [14], Independent Component Analysis (ICA) [16], Non-Negative Matrix Factorization (NMF) [18], and Factor Analysis [4] (FA). For manifold embedding techniques, it is also standard to pick the number of components that you would like to use to define a low-dimensional manifold for the data, including Local Linear Embedding (LLE) [21] and Isometric Mapping (Isomap) [24]. Restricted Boltzmann Machines (RBMs) [15] require inputs to be binary or in the range [0, 1]. Therefore we scale and translate each variable to fit in this range. For k-Means clustering [22], we fix the number of components to be \( k \), the number of cluster centers used. Then we transform each data point to a vector of distances from each of the cluster centers. All methods were run using standard implementations in the scikit library [20].

\[
\text{Data: Data matrix, } N \text{ iid samples of vectors, } x \in \mathbb{R}^n \\
\text{Result: Weights, } w, \text{ so that } y = w \cdot x \text{ optimizes } TC(X; Y) \text{ and remainder information, } \bar{x}. \\
\text{Set } \eta^2 = 1, \text{ or some arbitrary constant to set scale; } \\
\text{Subtract mean from each column of data; } \\
\text{Optionally, Gaussianize marginals of data; } \\
\text{Initialize } w_i \sim \mathcal{N}(0, \eta^2 \sqrt{\sigma_{x_i}}); \\
\text{while not converged do } \\
\quad \text{Calculate } y = w \cdot x \text{ for each sample; } \\
\quad \text{Calculate moments from data, } \langle X_j Y \rangle, \langle Y^2 \rangle, \langle X_j^2 \rangle; \\
\quad \forall j, w_j \leftarrow \eta^2 \langle X_j Y \rangle / (\langle Y^2 \rangle \langle X_j^2 \rangle - \langle X_j Y \rangle^2); \\
\text{end } \\
\text{Calculate remainder info, } \bar{x} = x - \frac{\langle X_j Y \rangle}{\langle Y^2 \rangle} y; \\
\text{Algorithm 1: Algorithm to learn one layer of the sieve.}
\]

B Proof of Theorem 3.1

We will closely follow the proof for the discrete version of the sieve [30], with the main deviations appearing in the statement of the invertibility criterion and in the last two paragraphs.

Definition The random variables \( Y \equiv Y_1, \ldots, Y_m \) constitute a representation of \( X \) if the joint distribution factorizes, \( p(x, y) = \prod_{j=1}^m p(y_j | x)p(x), \forall x \in X, \forall j \in \{1, \ldots, m\}, \forall y_j \in Y_j \). A
We use the chain rule again to write

\[ \sum_{i=1}^{n} I(Y : X_i) - \sum_{j=1}^{m} I(Y_j : X), \]  

then the following bound and decomposition holds.

\[ TC(X) \geq TC(X; Y) = TC(Y) + TC_L(X; Y) \]  

Theorem B.2. Incremental decomposition of common information \( Y \) a function of \( X \), the following decomposition holds,

\[ TC(X) = TC(\bar{X}) + TC(X; Y), \]  

as long as the remainder information \( \bar{X} \) satisfies two properties.

1. Invertibility: there exist functions \( g, h \) so that \( x_i = g(\bar{x}_i, y) \) and \( \bar{x}_i = h(x_i, y) \)

2. Remainder contains no information about \( Y \): \( \forall i, I(\bar{X}_i; Y) = 0 \)

Proof. We refer to Fig. 1(a) for the structure of the graphical model. We set \( \bar{X} = \bar{X}_1, \ldots, \bar{X}_n, Y \) and we write \( \bar{X}_1, \ldots, \bar{X}_n \) to pick out all terms except \( Y \). Note that because \( Y \) is a deterministic function of \( X \), we can view \( \bar{X}_i \) as a probabilistic function of \( X_i, Y \) or of \( X \) (as required by Thm. B.1). Applying Thm. B.1, we have

\[ TC(X; \bar{X}) = TC(\bar{X}) + TC_L(X; \bar{X}), \]

On the LHS, note that \( TC(X; \bar{X}) = TC(X) - TC(X|\bar{X}) \), so we can re-arrange to get

\[ TC(X) - (TC(\bar{X}) + TC(X; Y)) = TC(X|\bar{X}) + TC_L(X; \bar{X}) - TC(X; Y). \]  

The LHS is the quantity we are trying to bound, so we focus on expanding the RHS and bounding it. First we expand \( TC_L(X; \bar{X}) = \sum_{i=1}^{n} I(X_i; \bar{X}_i) - \sum_{i=1}^{n} I(\bar{X}_i; X_i) - I(Y; X). \) Using the chain rule for mutual information we expand the first term.

\[ TC_L(X; \bar{X}) = \sum_{i=1}^{n} I(X_i; Y) + \sum_{i=1}^{n} I(X_i; \bar{X}_{1:n}Y) - \sum_{i=1}^{n} I(\bar{X}_i; X_i) - I(Y; X). \]

Rearranging, we take out a term equal to \( TC(X; Y) \).

\[ TC_L(X; \bar{X}) = TC(X; Y) + \sum_{i=1}^{n} I(X_i; \bar{X}_{1:n}Y) - \sum_{i=1}^{n} I(\bar{X}_i; X_i). \]

We use the chain rule again to write \( I(X_i; \bar{X}_{1:n}Y) = I(X_i; \bar{X}_iY) + I(X_i; \bar{X}_i|Y\bar{X}_i) \), where \( \bar{X}_i \equiv \bar{X}_1, \ldots, \bar{X}_n \) with \( \bar{X}_i \) (and \( Y \)) excluded.

\[ TC_L(X; \bar{X}) = TC(X; Y) + \sum_{i=1}^{n} (I(X_i; \bar{X}_iY) + I(X_i; \bar{X}_i|Y\bar{X}_i) - I(\bar{X}_i; X_i)). \]

The conditional mutual information, \( I(A; B|C) = I(A; BC) - I(A; C) \). We expand the first instance of CMI in the previous expression.

\[ TC_L(X; \bar{X}) = TC(X; Y) + \sum_{i=1}^{n} (I(\bar{X}_i; X_iY) - I(\bar{X}_i; Y) + I(X_i; \bar{X}_i|YX_i) - I(\bar{X}_i; X_i)). \]

Since \( Y = f(X) \), the first and fourth terms cancel. Finally, this leaves us with

\[ TC_L(X; \bar{X}) = TC(X; Y) - \sum_{i=1}^{n} I(\bar{X}_i; Y) + \sum_{i=1}^{n} I(X_i; \bar{X}_i|Y\bar{X}_i). \]
Now we can replace all of this back in to Eq. 12 noting that the $TC(X; Y)$ terms cancel.

$$TC(X) - (TC(\bar{X}) + TC(X; Y)) = TC(X|\bar{X}) - \sum_{i=1}^{n} I(\bar{X}_i; Y) + \sum_{i=1}^{n} I(X_i; \bar{X}_i|Y\bar{X}_i).$$  \hspace{1cm} (13)

The middle term is zero because of assumption 2 and the final term is zero by assumption 1. It just remains to show that $TC(X|\bar{X}) = 0$.

$$TC(X|\bar{X}) = D_{KL}(p(x|\bar{x})||\prod_i p(x_i|\bar{x}))$$

Looking at the definition, we see that by assumption 1, each $x_i$ is a deterministic function of $\bar{x}$ (which by definition includes $y$). Therefore, $p(x|\bar{x})$ factorizes and this expression is zero.  \hfill \qed

\section{Proof of Corollary 3.2}

\textbf{Corollary C.1. Iterative decomposition of common information} \quad \text{We construct a hierarchical representation where each $Y_k$ is a function of $X^{k-1}$ and $X^k$ is the remainder information as defined in Thm 3.1.}

$$TC(X) = TC(X^r) + \sum_{k=1}^{r} TC(X^{k-1}; Y_k)$$

$$TC(X|Y_{1:r}) \leq TC(X^r) = TC(X) - \sum_{k=1}^{r} TC(X^{k-1}; Y_k)$$

\begin{align*}
X^0 : & \quad X_1 \ldots X_n \\
X^1 : & \quad X_1^r \ldots X_n Y_1 \\
X^2 : & \quad X_1^2 \ldots X_n^2 Y_2^r \\
& \quad \vdots \\
X^r : & \quad X_1^r \ldots X_n^r Y_1^r Y_2^r \ldots Y_r
\end{align*}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{variable_naming.png}
\caption{Summary of variable naming scheme for multiple layers of the sieve.}
\end{figure}

\textit{Proof.} The top line follows from repeated application of Eq. 12. The second line follows once we show that $TC(X|Y_{1:r}) \leq TC(X^r)$, where $X^r$ is defined according to the remainder information criteria in Thm 3.1. We use Fig. C.1 as a reminder of the variable naming conventions. In particular, we point out that each latent factor, $Y_k$ is also transformed and added to subsequent layers. For remainder information, we have that $x_i^{k-1} = f_{i,k}(x_i^r, y_k)$, for some function $f_{i,k}$. If we repeated this definition at each level we find $x_i^0 = f_{i,1}(x_i^1, y_1) = f_{i,2}(f_{i,1}(x_i^1, y_2), y_1)$, etc. Letting $k = 0$ represent the input layer and $k = r$ represent the final layer, we can write the function connecting the input layer and layer $r$ succinctly as $x_i = h_i(x_i^r, y_{1:r})$. For succinctness, we write the inverse of this function as $g \equiv h^{-1}$, $x_i^r = g_i(x_i, y_{1:r})$. By the same argument, we have functions $y_k = c_k(y_{k:r}^r), y_k^r = d_k(y_{k:r})$. Using the change of variables formulate,

$$p(x_{1:n}, y_{1:r}) = p(x_{1:n}, y_{1:r}) \frac{\partial(g, d)}{\partial(x, y)}$$

The determinant of the Jacobian has a special form that can be reduced using the Schur complement to the following expression: $\prod_{i=1}^{n} g_i'(x_i, y) \prod_{k=1}^{r} d_k'(y_{k:r})$. We can also see that $p(x, y_{1:r}) = p(x^r, y_{1:r}) g_i(x, y) \prod_{k=1}^{r} d_k'(y_{k:r})$ via the same argument and $p(y_{1:r}) = p(y_{1:r}) \prod_{k=1}^{r} d_k'(y_{k:r})$. Then $p(x_{1:n}|y_{1:r}) = p(x_{1:n}^r | y_{1:r}^r) \prod_{i=1}^{n} g_i'(x_i, y)$.
Now we can calculate $TC(X_{1:n}|Y_{1:r})$ using these expressions.

$$TC(X_{1:n}|Y_{1:r}) = \left\{ \log \prod_{i=1}^n p(x_i|y_{1:r}) \right\} = \left\{ \log \prod_{i=1}^n p(x_i|y_{1:r}) \right\}$$

After some rearranging, the last inequality follows from the non-negativity of TC and mutual information.

### D Derivation of Eq. \[14\]

We are attempting to optimize the following expression.

$$\max_{y} \frac{1}{2} \sum_{i} \log \left( 1 - \frac{\langle X_i Y \rangle^2}{\langle X_i^2 \rangle \langle Y^2 \rangle} \right) - \frac{1}{2} \log \frac{\langle Y^2 \rangle}{\eta^2}$$

Here $\epsilon$ is a Gaussian random variable with constant variance $\eta^2$. The value of $\eta$ is arbitrary, however, it sets the scale of $Y$ (otherwise $Y$ would be invariant under scaling and could, in principle be very large or small with no effect on the objective). Next, we take the derivative of the objective, $\mathcal{O}$, with respect to $w_j$. We will use $\partial_j = \frac{\partial}{\partial w_j}$. Note that $w_j$ only appears within $Y$ and $\partial_j Y = X_j$.

$$\partial_j \mathcal{O} = \frac{1}{2} \sum_{i} \partial_j \frac{\langle X_i Y \rangle^2}{\langle X_i^2 \rangle \langle Y^2 \rangle} - \frac{\langle X_i Y \rangle^2}{\langle X_i^2 \rangle} \frac{\langle X_i Y \rangle}{\langle X_i^2 \rangle \langle Y^2 \rangle} = \sum_i \frac{\langle X_i Y \rangle^2}{\langle X_i^2 \rangle \langle Y^2 \rangle} - \frac{\langle X_i Y \rangle^2}{\langle X_i^2 \rangle} \frac{\langle X_i Y \rangle}{\langle X_i^2 \rangle \langle Y^2 \rangle} - \frac{\langle X_i Y \rangle}{\langle Y^2 \rangle} = 0$$

This gives us a set of $n$ equations that are set equal to zero. We analytically multiply this set of equations by $\Sigma_{XX}^{-1}$ to transform them into the following fixed point equations for each $j$. Note that $\sum_j \Sigma_{XX}^{-1} \Sigma_{XX;j} = \delta_{j,j}$ and $\sum_j \Sigma_{XX}^{-1} \langle X_i Y \rangle = w_i$. We are not guaranteed that $\Sigma_{XX}$ is invertible. Because this step is formal, not numerical, we could imagine adding a tiny amount of iid noise to the $X_i$'s, far below the experimental precision of the data, to guarantee invertibility.

$$w_j = \frac{\langle Y^2 \rangle}{1 + d} \cdot \frac{\langle X_i Y \rangle}{\langle X_i^2 \rangle \langle Y^2 \rangle - \langle X_i Y \rangle^2}$$

Here, $d = \sum_r \beta_r^2/(1 - \beta_r^2)$ with $\beta_r = \langle X_i, Y \rangle^2 / (\langle X_i^2 \rangle \langle Y^2 \rangle)$. We can simplify this fixed point equation by multiplying by $\langle X_i Y \rangle$ and summing over $i$. This yields the identity, $\langle Y^2 \rangle/(1 + d) = \eta^2$. Re-writing the update equation, we get the following expression where we see explicitly how $\eta$ simply sets the scale for $Y$.

$$w_j = \eta^2 \frac{\langle X_i Y \rangle}{\langle X_i^2 \rangle \langle Y^2 \rangle - \langle X_i Y \rangle^2}$$

### E Component analysis on Olivetti faces

PCA is often used for exploratory data analysis. The first component of PCA captures the dimension of greatest variance in the data. In contrast, the first sieve component explains the most dependence.
in the data. While the usefulness of different methods of component analysis for exploratory analysis is largely qualitative, we show one example comparing the top components for both PCA and the sieve on the Olivetti faces dataset in Fig. E.1.

First of all, the similarity of the components, including the ranking, is striking. This similarity is despite the fact that sieve is only linear while PCA is quadratic in the number of variables. However, there are some apparent differences in the results. The darkness of the blues and reds for PCA suggest that most weights are near to the maximum weight. The strongest weights for the sieve are more localized. For instance, the weights around the eyes in the fifth component and on the forehead for the sixth component are more highlighted for the sieve.

We can also use the sieve for reconstructing data from a small number of learned factors. Note that the sieve transform is invertible so that $X_i = X_1^r + \langle X_1^{k+1} Y_k \rangle / \langle Y_k^2 \rangle Y_k$. If we have a sieve transformation with $r$ layers, then we can continue this expansion to get the following.

$$X_i = X_1^r + \sum_{k=1}^{r} \langle X_1^{k+1} Y_k \rangle / \langle Y_k^2 \rangle Y_k$$

If we knew the remainder information, $X_1^r$, this transformation would be perfect. However, we can simply set the $X_1^r = 0$ and we will get a prediction for $X_i$ based only on the learned factors, $Y$. The result of this procedure on the Olivetti faces is shown in Fig. E.2. We compare to reconstruction with PCA. The results subtly differ but neither appears obviously preferable for reconstructing the image.

**Figure E.1:** The top 6 components for the Olivetti faces dataset using the information sieve (top) and PCA (bottom). Red and blue correspond to negative and positive weights respectively.

**Figure E.2:** We take Olivetti faces (middle row) and then try to reconstruct them using the top 20 components from the sieve (top row) or PCA (bottom row).