The Information Sieve

Greg Ver Steeg   GREGV@ISI.EDU

University of Southern California, Information Sciences Institute, Marina del Rey, CA 90292 USA

Aram Galstyan   GALSTYAN@ISI.EDU

University of Southern California, Information Sciences Institute, Marina del Rey, CA 90292 USA

Abstract

We introduce a new framework for unsupervised learning of representations based on a novel hierarchical decomposition of information. Intuitively, data is passed through a series of progressively fine-grained sieves. Each layer of the sieve recovers a single latent factor that is maximally informative about multivariate dependence in the data. The data is transformed after each pass so that the remaining unexplained information trickles down to the next layer. Ultimately, we are left with a set of latent factors explaining all the dependence in the original data and remainder information consisting of independent noise. We present a practical implementation of this framework for discrete variables and apply it to a variety of fundamental tasks in unsupervised learning including independent component analysis, lossy and lossless compression, and predicting missing values in data.

The hope of finding a succinct principle that elucidates the brain's information processing abilities has often kindled interest in information-theoretic ideas (Barlow, 1989; Simoncelli & Olshausen, 2001). In machine learning, on the other hand, the past decade has witnessed a shift in focus toward expressive, hierarchical models, with successes driven by increasingly effective ways to leverage labeled data to learn rich models (Schmidhuber, 2015; Bengio et al., 2013). Information-theoretic ideas like the venerable InfoMax principle (Linsker, 1988; Bell & Sejnowski, 1995) can be and are applied in both contexts with empirical success but they do not allow us to quantify the *information value* of adding depth to our representations. We introduce a novel incremental and hierarchical decomposition of information and show that it defines a framework for unsu-

pervised learning of deep representations in which the information contribution of each layer can be precisely quantified. Moreover, this scheme automatically determines the structure and depth among hidden units in the representation based only on local learning rules.

The shift in perspective that enables our information decomposition is to focus on how well the learned representation explains multivariate mutual information in the data (a measure originally introduced as "total correlation" (Watanabe, 1960)). Intuitively, our approach constructs a hierarchical representation of data by passing it through a sequence of progressively fine-grained sieves. At the first layer of the sieve we learn a factor that explains as much of the dependence in the data as possible. The data is then transformed into the "remainder information", which has this dependence extracted. The next layer of the sieve looks for the largest source of dependence in the remainder information, and the cycle repeats. At each step, we obtain a successively tighter upper and lower bound on the multivariate information in the data, with convergence between the bounds obtained when the remaining information consists of nothing but independent factors. Because we end up with independent factors, one can also view this decomposition as a new way to do independent component analysis (ICA) (Comon, 1994; Hyvärinen & Oja, 2000). Unlike traditional methods, we do not assume a specific generative model of the data (i.e., that it consists of a linear transformation of independent sources) and we extract independent factors incrementally rather than all at once. The implementation we develop here uses only discrete variables and is therefore most relevant for the challenging problem of ICA with discrete variables, which has applications to compression (Painsky et al., 2014).

After providing some background in Sec. 1, we introduce a new way to iteratively decompose the information in data in Sec. 2, and show how to use these decompositions to define a practical and incremental framework for unsupervised representation learning in Sec. 3. We demonstrate the versatility of this framework by applying it first to independent component analysis (Sec. 4). Next, we use the sieve

as a lossy compression to perform tasks typically relegated to generative models including in-painting and generating new samples (Sec. 5). Finally, we cast the sieve as a lossless compression and show that it beats standard compression schemes on a benchmark task (Sec. 6).

1 Information-theoretic learning background

Using standard notation (Cover & Thomas, 2006), capital X_i denotes a random variable taking values in some domain and whose instances are denoted in lowercase, x_i. In this paper, the domain of all variables are considered to be discrete and finite. We abbreviate multivariate random variables, $X \equiv X_{1:n} \equiv X_1, \ldots, X_n$, with an associated probability distribution, $p_X(X_1 = x_1, \ldots, X_n = x_n)$, which is typically abbreviated to $p(x)$. 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 often omit the superscript in this case for readability.

Entropy is defined in the usual way as $H(X) \equiv \mathbb{E}_X[\log 1/p(x)]$. We use base two logarithms so that the unit of information is bits. Higher-order entropies can be constructed in various ways from this standard definition. For instance, the mutual information between two groups of random variables, X and Y can be written as the reduction of uncertainty in one variable, given information about the other, $I(X;Y) = H(X) - H(X|Y)$.

The "InfoMax" principle (Linsker, 1988; Bell & Sejnowski, 1995) suggests that for unsupervised learning we should construct Y's to maximize their mutual information with X, the data. Despite its intuitive appeal, this approach has several potential problems (see (Ver Steeg et al., 2014) for one example). Here we focus on the fact that the InfoMax principle is not very useful for characterizing "deep representations", even though it is often invoked in this context (Vincent et al., 2008). This follows directly from the data processing inequality (a similar argument appears in (Tishby & Zaslavsky, 2015)). Namely, if we start with X, construct a layer of hidden units Y^1 that are a function of X, and continue adding layers to a stacked representation so that $X \to Y^1 \to Y^2 \ldots Y^k$, then the information that the Y's have about X cannot increase after the first layer, $I(X;Y^{1:k}) = I(X;Y^1)$. From the point of view of mutual information, Y^1 is a copy and Y^2 is just a copy of a copy. While a coarse-grained copy might be useful, the InfoMax principle does not quantify how or why.

Instead of looking for a Y that memorizes the data, we shift our perspective to searching for a Y so that the X_i's are as independent as possible conditioned on this Y. Essentially, we are trying to reconstruct the latent factors that are the cause of the dependence in X_i. To formalize this, we introduce the multivariate mutual information which was first introduced as "total correlation" (Watanabe, 1960).

$$
\begin{aligned}
TC(X) &\equiv D_{KL}\left(p(x) \| \prod_{i=1}^{n} p(x_i)\right) \\
&= \sum_{i=1}^{n} H(X_i) - H(X)
\end{aligned}
\tag{1}
$$

This quantity reflects the dependence in X and is zero if and only if the X_i's are independent. Just as mutual information is the reduction of entropy in X after conditioning on Y, we can define the reduction in multivariate information in X after conditioning on Y.

$$
\begin{aligned}
TC(X;Y) &\equiv TC(X) - TC(X|Y) \\
&= \sum_{i=1}^{n} I(X_i;Y) - I(X;Y).
\end{aligned}
\tag{2}
$$

That $TC(X)$ can be hierarchically decomposed in terms of short and long range dependencies was already appreciated by Watanabe (Watanabe, 1960) and has been used in applications such as hierarchical clustering (Kraskov et al., 2005). This provides a hint about how higher levels of hierarchical representations can be useful: more abstract representations should reflect longer range dependencies in the data. Our contribution below is to demonstrate a tractable approach for learning a hierarchy of latent factors, Y, that exactly capture the multivariate information in X.

2 Incremental information decomposition

We consider any set of probabilistic functions of some input variables, X, to be a "representation" of X. Looking at Fig. 1(a), we consider a representation with a single learned latent factor, Y. Then, we try to save the information in X that is not captured by Y into the "remainder information", $\bar{X}$. The final result is encapsulated in Cor. 2.4 which says that we can repeat this procedure iteratively (as in Fig. 1(b)) and $TC(X)$ decomposes into a sum of non-negative contributions from each Y_k. Note that $X^{(k)}$ includes Y_k, so that Y's at subsequent layers can depend on latent factors learned at earlier layers.

Theorem 2.1. Incremental Decomposition of Information *Let Y be some (deterministic) function of $X_1, \ldots, X_n$ and let $\bar{X}_i$ be a probabilistic function of X_i, Y, for each $i = 1, \ldots, n$. Then the following upper and lower bounds on $TC(X)$ hold:*

$$
A \leq TC(X) - \big(TC(\bar{X}) + TC(X;Y)\big) \leq B
$$

$$
A = -\sum_{i=1}^{n} I(\bar{X}_i;Y), \quad B = \sum_{i=1}^{n} H(X_i|\bar{X}_i,Y)
\tag{3}
$$

$$X^0 : \mathbf{X_1} \ldots \mathbf{X_n}$$
$$X^1 : X_1^1 \ldots X_n^1 \ Y_1$$
$$X^2 : X_1^2 \ldots X_n^2 \ Y_1^2 \ Y_2$$
$$\ldots$$
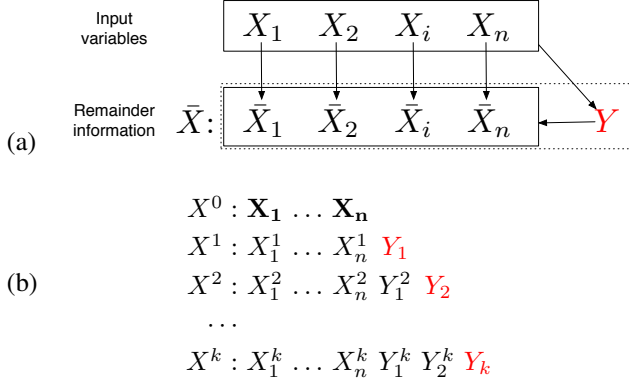$$X^k : X_1^k \ldots X_n^k \ Y_1^k \ Y_2^k \ Y_k$$

Figure 1. (a) This diagram describes one layer of the information sieve. In this graphical model, the variables in the top layer (X_i's) represent (observed) input variables. Y is some function of all the X_i's that is optimized to be maximally informative about multivariate dependence in X. The remainder information, $\bar{X}_i$ depends on X_i and Y and is set to contain information in X_i that is not captured by Y. (b) Summary of variable naming scheme for multiple layers of the sieve. The input variables are in bold and the learned latent factors are in red.

A proof is provided in App. B. Note that the remainder information, $\bar{X} \equiv \bar{X}_1, \ldots, \bar{X}_n, Y$, includes Y. Bounds on $TC(X)$ also provide bounds on $H(X)$ by using Eq. 1. Next, we point out that the remainder information, $\bar{X}$, can be chosen to make these bounds tight.

Lemma 2.2. Construction of perfect remainder information *For discrete, finite random variables X_i, Y drawn from some distribution, $p(X_i, Y)$, it is possible to define another random variable $\bar{X}_i \sim p(\bar{X}_i | X_i, Y)$ that satisfies the following two properties:*

(i) $I(\bar{X}_i; Y) = 0$ *Remainder contains no Y info*

(ii) $H(X_i | \bar{X}_i, Y) = 0$ *Original is perfectly recoverable*

We give a concrete construction in App. C. We would like to point out one caveat here. The cardinality of $\bar{X}_i$ may have to be large to satisfy these equalities. For a fixed number of samples, this may cause difficulties with estimation, as discussed in Sec. 3. With perfect remainder information in hand, our decomposition becomes exact.

Corollary 2.3. Exact decomposition *For Y a function of X and perfect remainder information, $\bar{X}_i, i = 1, \ldots, n$, as defined in Lemma 2.2, the following decomposition holds:*

$$TC(X) = TC(\bar{X}) + TC(X; Y) \tag{4}$$

The above corollary follows directly from Eq. 3 and the definition of perfect remainder information. Intuitively, it states that the dependence in X can be decomposed into a piece that is explained by Y, $TC(X; Y)$, and the remaining

dependence in $\bar{X}$. This decomposition can then be iterated to extract more and more information from the data.

Corollary 2.4. Iterative decomposition *Using the variable naming scheme in Fig. 1(b), we construct a hierarchical representation where each Y_k is a function of X^{k-1} and X^k includes the (perfect) remainder information from X^{k-1} according to Lemma 2.2.*

$$TC(X) = TC(X^r) + \sum_{k=1}^{r} TC(X^{k-1}; Y_k) \tag{5}$$

It is easy to check that Eq. 5 results from repeated application of Cor. 2.3. We show in the next section that the quantities of the form $TC(X^{k-1}; Y_k)$ can be estimated and optimized over efficiently, despite involving high-dimensional variables. As we add the (non-negative) contributions from optimizing $TC(X^{k-1}; Y_k)$, the remaining dependence in the remainder information, $TC(X^k)$, must decrease because $TC(X)$ is some data-dependent constant. Decomposing data into independent factors is exactly the goal of ICA, and the connections are discussed in Sec. 4.

3 Implementing the sieve

Because this learning framework contains many unfamiliar concepts, we consider a detailed analysis of a toy problem in Fig. 2 while addressing concrete issues in implementing the information sieve.

$$Y_1 = \arg \max_{Y=f(X)} TC(X; Y)$$
$$X_1^1 = X_1 + Y_1 \mod 2$$
$$X_2^1 = X_2 + Y_1 \mod 2$$
$$X_3^1 = X_3$$

Data			Remainder			
X_1	X_2	X_3	Y_1	X_1^1	X_2^1	X_3^1
0	0	1	0	0	0	1
0	0	0	0	0	0	0
1	1	0	1	0	0	0
1	1	1	1	0	0	1

Figure 2. A simple example for which we imagine we have samples of X drawn from some distribution.

Step 1: Optimizing $TC(X^{k-1}; Y_k)$ First, we construct a variable, Y_k, that is some arbitrary function of X^{k-1} and that explains as much of the dependence in the data is possible. Note that we have to pick the cardinality of Y_k and we will always use binary variables. Dropping the layer indices, k, the optimization can be written as follows.

$$\max_{p(y|x)} \sum_i I(X_i; Y) - I(X; Y) \tag{6}$$

Here, we have relaxed the optimization to allow for probabilistic functions of X. If we take the derivative of this expression (along with the constraint that $p(y|x)$ should be normalized) and set it equal to zero, the following simple

fixed point equation emerges.

$$p(y|x) = \frac{p(y)}{Z(x)} \prod_{i=1}^{n} \frac{p(x_i|y)}{p(x_i)}$$

The state space of X is exponentially large in n, the number of variables. Fortunately, this fixed point equation tells us that we can write the solution in terms of a linear number of terms which are just marginal likelihood ratios. Details of this optimization are discussed in Sec. A. Note that the optimization provides a probabilistic function which we round to a deterministic function by taking the most likely value of Y for each X. In the example in Fig. 2, $TC(X; Y_1) = 1$ bit, which can be verified from Eq. 2.

Surprisingly, we did not need to restrict or parametrize the set of possible functions; the simple form of the solution was implied by the objective. Furthermore, we can also use this function to find labels for previously unseen examples or to calculate Y's for data with missing variables (details in Sec. A). Not only that, but a byproduct of the procedure is to give us a value for the objective $TC(X^{k-1}; Y_k)$, which can be estimated even from a small number of samples.

Step 2: Remainder information   Next, the goal is to construct the remainder information, X_i^k, as a probabilistic function of X_i^{k-1}, Y_k, so that the following conditions are satisfied: $(i) I(X_i^k; Y_1) = 0$ and $(ii) H(X_i^{k-1}|X_i^k, Y_k) = 0$. This can be done exactly and we provide a simple algorithm in Sec. C. Solutions for this example are given in Fig. 2. Concretely, we estimate the marginals, $p(x_i^{k-1}, y_k)$ from data and then write down a conditional probability table, $p(x_i^k|x_i^{k-1}, y_k)$, satisfying the conditions. The example in Fig. 2 was constructed so that the remainder information had the same cardinality as the original variables. This is not always possible. While we can always achieve perfect remainder information by letting the cardinality of the remainder information grow, it might become difficult to estimate marginals of the form $p(X_i^{k-1}, Y_k)$ at subsequent layers of the sieve, as is required for the optimization in step 1. In results shown below we allow the cardinality of the variables to increase by only one at each level to avoid state space explosion, even if doing so causes $I(X_i^{k-1}; Y_k) > 0$. We keep track of these penalty terms so that we can report accurate lower bounds using Eq. 3.

Another issue to note is that in general there may not be a unique choice for the remainder information. In the example, $I(X_3; Y) = 0$ already so we choose $X_3^1 = X_3$, but $X_3^1 = X_3 + Y_1 \mod 2$ would also have been a valid choice. If the identity transformation, $X_i^k = X_i^{k-1}$ satisfies the conditions, we will always choose it.

Step 3: Repeat until the end   At this point we repeat the procedure, putting the remainder information back into step 1 and searching for a new latent factor that explains any remaining dependency. In this case, we can see by inspection

that $TC(X^1) = 0$ and, using Eq. 5, we have $TC(X) = TC(X^1) + TC(X; Y_1) = 1$ bit. Generally, in high-dimensional spaces it may be difficult to verify that the remainder information is truly independent. When the remainder information is independent, the result of attempting the optimization $\max_{p(y_k|x^{k-1})} TC(X^{k-1}; Y_k) = 0$. In practice, we stop our hierarchical procedure when the optimization in step 1 stops producing positive results because it means our bounds are no longer tightening. Code implementing this entire pipeline is available (Ver Steeg).

Prediction and compression   Note that our condition for the remainder information that $H(X_i^{k-1}|X_i^k, Y_k) = 0$ implies that we can perfectly reconstruct each variable X_i^{k-1} from the remainder information at the next layer. Therefore, we can in principle reconstruct the data from the representation at the last layer of the sieve. In the example, the remainder information requires two bits to encode each variable separately, while the data requires three bits to encode each variable separately. The final representation has exploited the redundancy between X_1, X_2 to create a more succinct encoding. A use case for lossy compression is discussed in Sec. 5. Also note that at each layer some variables are almost or completely explained (X_1^1, X_2^1 in the example become constant). Subsequent layers can enjoy a computational speed-up by ignoring these variables that will no longer contribute to the optimization.

4   Discrete ICA

If X represents observed variables then the entropy, $H(X)$, can be interpreted as the average number of bits required to encode a single observation of these variables. In practice, however, if X is high-dimensional then estimating $H(X)$ or constructing this code requires detailed knowledge of $p(x)$, which may require exponentially many samples in the number of variables. Going back at least to Barlow (Barlow, 1989), it was recognized that if X is transformed into some other basis, Y, with the Y's independent ($TC(Y) = 0$), then the coding cost in this new basis is $H(Y) = \sum_j H(Y_j)$, i.e., it is the same as encoding each variable separately. This is exactly the problem of independent component analysis: transform the data into a basis for which $TC(Y) = 0$, or is minimized (Comon, 1994; Hyvärinen & Oja, 2000).

While our method does not directly minimize the total correlation of Y, Eq. 5 shows that, because $TC(X)$ is a data-dependent constant, every increase in the total correlation explained by each latent factor directly implies a reduction in the dependence of the resulting representation,

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

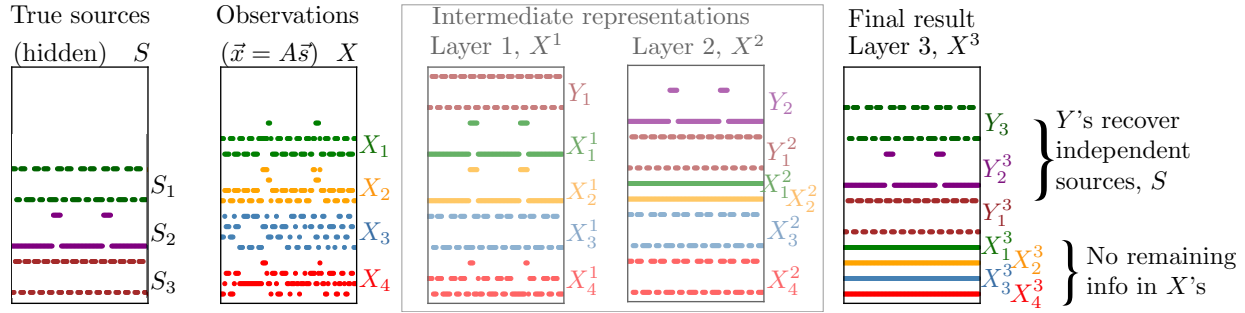Since the terms in the sum are optimized (and always non-

Figure 3. On the far left, we consider three independent binary random variables, S_1, S_2, S_3. The vertical position of each signal is offset for visibility. From left to right: Independent source data is linearly mixed, $\vec{x} = A\vec{s}$. This data, X, is fed into the information sieve. After going through some intermediate representations, the final result is shown on the far right. Consult Fig. 1(b) for the variable naming scheme. The final representation recovers the independent input sources.

negative), the dependence is decreased at each level. That independence could be achieved as a byproduct of efficient coding has been previously considered (Hochreiter & Schmidhuber, 1999). An approach that leading to "less dependent components" for continuous variables has also been shown (Stögbauer et al., 2004).

For discrete variables, which are the focus of this paper, performing ICA is a challenging and active area of research. Recent state-of-the-art results lower the complexity of this problem to *only* a single exponential in the number of variables (Painsky et al., 2014). Our method represents a major leap for this problem as it is only linear in the number of variables, however, we only guarantee extraction of components that are *more independent*, while the approach of Painsky et. al. guarantees a global optimum.

The most commonly studied scenario for ICA is to consider a reconstruction problem where some (typically continuous) and independent source variables are linearly mixed according to some unknown matrix (Comon, 1994; Hyvärinen & Oja, 2000). The goal is to recover the matrix and unmix the components (back into their independent sources). Next we demonstrate our discrete independent component recovery on an example reminiscent of traditional ICA examples.

An ICA example   Fig. 3 shows an example of recovering independent components from discrete random variables. The sources, S, are hidden and the observations, X, are a linearly mixture of these sources. The mixing matrix used in this example is

$$A = ((1, 1, 1), (2, 0, -1), (1, 2, 0), (-1, 1, 0)).$$

The information sieve continues to add layers as long as it increases the tightness of the information bounds. The intermediate representations at each layers are also shown. For instance, layer 1 extracts one independent component, and then removes this component from the remainder information. After three layers, the sieve stops because X^3 con-

sists of independent variables and therefore the optimization of $\max TC(X^3; Y_4) = 0$.

In this case, the procedure correctly stops after three latent factors are discovered. Naively, three layers makes this a "deep" representation. However, we can examine the functional dependence of Y's and X's by looking at the strength of the mutual information, $I(Y_k; X_i^{k-1})$, as shown in Fig. 4. This allows us to see that none of the learned latent factors (Y's) depend on each other so the resulting model is actually, in some sense, shallow. The example in the next section, for contrast, has a deep structure where Y's depend on latent factors from previous layers. Note that the structure in Fig. 4 perfectly reflects the structure of the mixing matrix (i.e., if we flipped the arrows and changes the Y's to S's, this would be an accurate representation of the generative model we used).
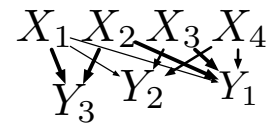

Figure 4. This visualizes the structure of the learned representation for the ICA example in Fig. 3. The thickness of links is proportional to $I(Y_k; X_i^{k-1})$.

While the sieve is guaranteed to recover independent components in some limit, there may be multiple ways to decompose the data into independent components. Because our method does not start with the assumption of a linear mixing of independent sources, even if such a decomposition exists we might recover a different one. While the example we showed happened to return the linear solution that we used to generate the problem, there is no guarantee to find a linear solution, even if one exists.

5. Lossy compression on MNIST digits

The information sieve is not a generative probabilistic model. We construct latent factors that are functions of the data in a way that maximizes the (multivariate) information that is preserved. Nevertheless, because of the way the remainder information is constructed, we can run the sieve in reverse and, if we throw away the remainder information and keep only the Y's, we get a lossy compression. We can use this lossy compression interpretation to perform tasks that are usually achieved using generative models including in-painting and generating new examples (the converse, interpreting a generative model as lossy compression, has also been considered (Hinton & Salakhutdinov, 2006)).

We illustrate the steps for lossy compression and in-painting in Fig. 5. Imagine that we have already trained a sieve model. For lossy compression, we first transform some data using the sieve. The sieve is an invertible transformation, so we can run it in reverse to exactly recover the inputs. Instead we store only the labels, Y, throwing the remainder information, $X_{1:n}^k$, away. When we invert the sieve, what values should we input for $X_{1:n}^k$? During training, we estimate the most likely value to occur for each variable, X_i^k. W.l.o.g., we relabel the symbols so that this value is 0. Then, for lossy recovery, we run the sieve in reverse using the labels we stored, Y, and setting $X_{1:n}^k = 0$.

In-painting proceeds in essentially the same way. We take advantage of the fact that we can transform data even in the presence of missing values, as described in Sec. A. Then we replace missing values in the remainder information with 0's and invert the sieve normally.
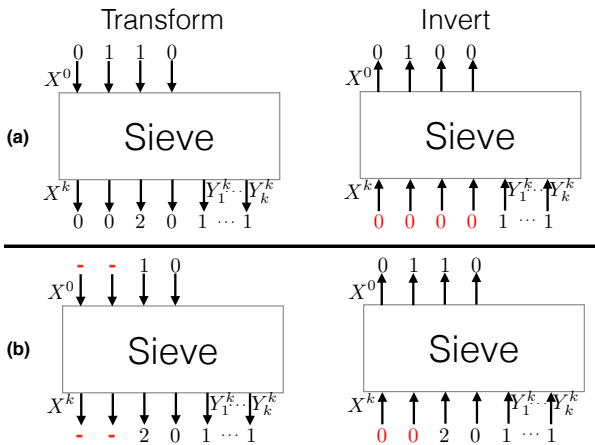

Figure 5. (a) We use the sieve to transform data into some labels, Y plus remainder information. For lossy recovery, we invert the sieve using only the Y's, setting X's to zero. (b) For in-painting, we first transform data with missing values. Then we invert the sieve, again using zeros for the missing remainder information.

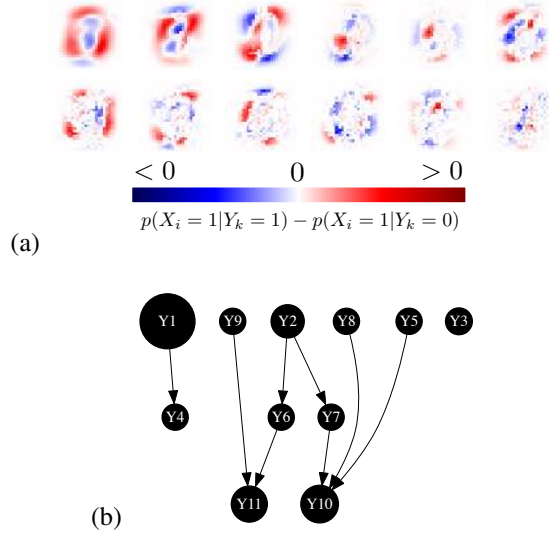For the following tasks, we consider 50k MNIST digits

Figure 6. (a) We visualize each of the learned components, arranged in reading order. (b) The structural relationships among the latent factors is based on $I(Y_k; Y_j^{k-1})$. The size of a node represents the magnitude of $TC(X^{k-1}; Y_k)$.

that were binarized at the normalized grayscale threshold of 0.5. We include no prior knowledge about spatial structure or invariance under transformations through convolutional structure or pooling, for instance. The 28×28 binarized images are treated as binary vectors in a 784 dimensional space. The digit labels are also not used in our analysis. We trained the information sieve on this data, adding layers as long as the bounds were tightening. This led to a 12 layer representation and a lower bound on $TC(X)$ of about 40 bits. It seems likely that more than 12 layers could be effective but the growing size of the state space for the remainder information increases the difficulty of estimation with limited data. A visualization of the learned latent factors and the relationships among them appears in Fig. 6. Unlike the ICA example, the latent factors here have exhibit multi-layered relationships.

The middle row of Fig. 7 shows results from the lossy compression task. We use the sieve to transform the original digits into 12 binary latent factors, Y, plus remainder information for each pixel, $X_{1:784}^{12}$, and then we use the Y's alone to reconstruct the image. In the third row, the Y's are estimated using only pixels from the top half. Then we reconstruct the pixels on the bottom half from these latent factors. Similar results on test images are shown in Sec. D, along with examples of "hallucinating" new digits.

6. Lossless compression

Given samples of X drawn from $p(x)$, the best compression we can do in theory is to use an average of $H(X)$ bits
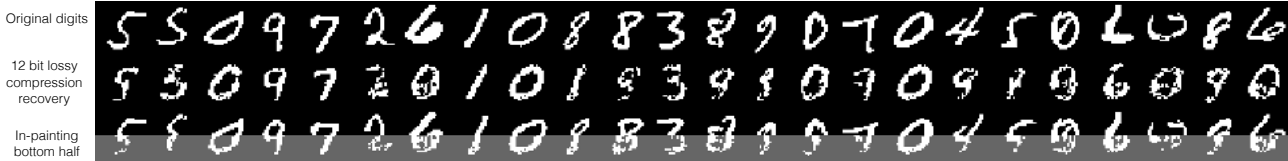
Figure 7. The top row are randomly selected MNIST digits. In the second row, we compress the digits into the 12 binary variables, Y_k, and then attempt to reconstruct the image. In the bottom row, we learn Y's using just the pixels in the top half and then recover the pixels in the bottom half.

for our compressed representation (Shannon, 1948). However, in practice, if X is high-dimensional then we cannot accurately estimate $p(x)$ to achieve this level of compression. We consider alternate schemes and compare them to the information sieve for a compression task.

Benchmark   For a lossless compression benchmark, we consider a set of 60k of binarized digits with 784 pixels, where the order of the pixels has been randomly permuted (the same unknown permutation is applied to each image). Note that we have made this task artificially more difficult than the straightforward task of compressing digits because many compression schemes exploit spatial correlations among neighboring pixels for efficiency. The information sieve is unaffected by this permutation since it does not make any assumptions about the structure of the input space (e.g. the adjacency of pixels). We use 50k digits as training for models, and report compression results on the 10k test digits.

Naively, these 28 by 28 binary pixels would require 784 bits per digit to store. However, some pixels are almost always zero. According to Shannon, we can compress pixel i using just $H(X_i)$ bits on average (Shannon, 1948). Because the state space of each individual bit is small, this bound is actually achievable (using arithmetic coding (Cover & Thomas, 2006), for example). Therefore, we should be able to store the digits using $\sum_i H(X_i) \approx 297$ bits/digit on average.

We would like to make the data more compressible by first transforming it. We consider a simplified version of the sieve with just one layer. We let Y take m possible values and then optimize it according to our objective. For the remainder information, we use the (invertible) function $\bar{x}_i = |x_i - \arg\max_z p(X_i = z|Y = y)|$. In other words, $\bar{X}$ represents deviation from the most likely value of x_i for a given value of y. The cost of storing a digit in this new representation will be $\log_2 m + \sum_{i=1}^{784} H(\bar{X}_i)$, where $\log_2 m$ bits are used to store the value of Y.

For comparison, we consider an analogous benchmark introduced in (Gregor & LeCun, 2011). For this benchmark, we just choose m random digits as representatives (from the training set). Then for each test digit, we store the identity of the closest representative (by Hamming distance),

along with the error which we will also call $\bar{X}_i$, so that we can recover the original digit. Again, the number of bits per digit will just by $\log_2 m$ plus the cost of storing the errors for each pixel according to Shannon.

Figure 8. This shows $p(x_i = 1|y = k)$ for $k = 1, \ldots, 20$ for each pixel, x_i, in an image.

Consider the single layer sieve with $Y = 1, \ldots, m$ and $m = 20$. After optimizing, Fig. 8 visualizes the components of Y. As an exercise in unsupervised clustering the results are somewhat interesting; the sieve basically finds clusters for each digit and for slanted versions of each digit. In Fig. 9 we explicitly construct the remainder information (bottom row), i.e. the deviation between the most likely value of each pixel conditioned on Y (middle row) and the original (top row).
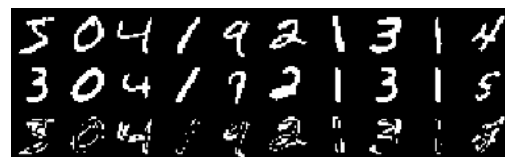

Figure 9. The top row shows the original digit, the middle row shows the most likely values of the pixels conditioned on the label, $y = 1, \ldots, 20$, and the bottom row shows the remainder or residual error, $\bar{X}$.

The results of our various compression benchmarks are shown in Table 1. For comparison we also show results from two standard compression schemes, gzip, based on Lempel-Ziv coding (Ziv & Lempel, 1977), and Huffman coding (Huffman et al., 1952). We take the better compression result from storing and compressing the 784×50000 data array in column-major or row-major order with these (sequence-based) compression schemes. Note that the sieve and random representative benchmark that we described require a codebook of fixed size whose contribution is asymptotically negligible and is not included in the results.

Table 1. Summary of compression results. Results with a "*" are reported based on empirical compression results rather than Shannon bounds.

Method	Bits per digit
Naive	784
Huffman* (Huffman et al., 1952)	376
gzip* (Ziv & Lempel, 1977)	328
Bitwise	297
20 random representatives	293
50 random representatives	279
100 random representatives	267
20 sieve representatives	266
50 sieve representatives	252
100 sieve representatives	243

Discussion First of all, sequence-based compression schemes have a serious disadvantage in this setup. Because the pixels are scrambled, to take advantage of correlations would require longer window sizes than is typical. The random compression scheme does significantly better. Despite the scrambled pixels, at least it uses the fact that the data consist of iid samples of length 784 pixels. However, the sieve leads to much better compression; for instance, 20 sieve representatives are as good as 100 random ones. The idea behind "factorial codes" (Barlow, 1989) is that if we can transform our data so that the variables are independent, and then (optimally) compress each variable separately, we will achieve a globally optimal compression. The compression results shown here are promising, but are not state-of-the-art. The reason is that our discovery of discrete independent components comes at a cost of increasing the cardinality of variables at each layer of the sieve. To define a more practical compression scheme, we would have to balance the trade-off between reducing dependence and controlling the size of the state space. We leave this direction for future work.

7 Related work

The idea of decomposing multivariate information as an underlying principle for unsupervised representation learning has been recently introduced (Ver Steeg & Galstyan, 2015; 2014) and used in several contexts (Pepke & Ver Steeg, 2016; Madsen et al., 2016). While bounds on $TC(X)$ were previously given, here we provided an exact decomposition. Our decomposition also introduces the idea of remainder information. While previous work required fixing the depth and number of latent factors in the representation, remainder information allows us to build up the representation incrementally, learning the depth and number of factors required as we go. Besides providing a more flexible approach to representation and structure learning, the invertibility of the information sieve makes it more naturally suited to a wider variety of tasks including lossy and lossless compression and prediction. Another

interesting related result showed that positivity of the quantity $TC(X; Y)$ (the same quantity appearing in our bounds) implies that the X's share a common ancestor in any DAG consistent with $p_X(x)$ (Steudel & Ay, 2015). A different line of work about information decomposition focuses on distinguishing synergy and redundancy (Williams & Beer, 2010), though these measures are typically impossible to estimate for high-dimensional systems. Finally, a different approach to information decomposition focuses on the geometry of the manifold of distributions defined by different models (Amari, 2001).

Connections with ICA were discussed in Sec. 4 and the relationship to InfoMax was discussed in Sec. 1. The information bottleneck (IB) (Tishby et al., 2000) is another information-theoretic optimization for constructing representations of data that has many mathematical similarities to the objective in Eq. 6, with the main difference being that IB focuses on supervised learning while ours is an unsupervised approach. Recently, the IB principle was used to investigate the value of depth in the context of supervised learning (Tishby & Zaslavsky, 2015). The focus here, on the other hand, is to find an information-theoretic principle that justifies and motivates deep representations for unsupervised learning.

8 Conclusion

We introduced the information sieve, which provides a decomposition of multivariate information for high-dimensional (discrete) data that is also computationally feasible. The extension of the sieve to continuous variables is nontrivial but appears to result in algorithms that are more robust and practical (Ver Steeg et al., 2016). We established here a few of the immediate implications of the sieve decomposition. First of all, we saw that a natural notion of "remainder information" arises and that this allows us to extract information in an incremental way. Several distinct applications to fundamental problems in unsupervised learning were demonstrated and appear promising for in-depth exploration. The sieve provides an exponentially faster method than the best known algorithm for discrete ICA (though without guarantees of global optimality). We also showed that the sieve defines both lossy and lossless compression schemes. Finally, the information sieve suggests a novel conceptual framework for understanding unsupervised representation learning. Among the many deviations from standard representation learning a few properties stand out. Representations are learned incrementally and the depth and structure emerge in a data-driven way. Representations can be evaluated information-theoretically and the decomposition allows us to separately characterize the contribution of each hidden unit in the representation.

Acknowledgments

References

Amari, Shun-ichi. Information geometry on hierarchy of probability distributions. *Information Theory, IEEE Transactions on*, 47(5):1701–1711, 2001.

Barlow, Horace. Unsupervised learning. *Neural computation*, 1 (3):295–311, 1989.

Bell, Anthony J and Sejnowski, Terrence J. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159, 1995.

Bengio, Yoshua, Courville, Aaron, and Vincent, Pascal. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

Comon, Pierre. Independent component analysis, a new concept? *Signal processing*, 36(3):287–314, 1994.

Cover, Thomas M and Thomas, Joy A. *Elements of information theory*. Wiley-Interscience, 2006.

Gregor, Karol and LeCun, Yann. Learning representations by maximizing compression. *arXiv preprint arXiv:1108.1169*, 2011.

Hinton, Geoffrey E and Salakhutdinov, Ruslan R. Reducing the dimensionality of data with neural networks. *Science*, 313 (5786):504–507, 2006.

Hochreiter, Sepp and Schmidhuber, Jürgen. Feature extraction through lococode. *Neural Computation*, 11(3):679–714, 1999.

Huffman, David A et al. A method for the construction of minimum redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

Hyvärinen, Aapo and Oja, Erkki. Independent component analysis: algorithms and applications. *Neural networks*, 13(4):411–430, 2000.

Kraskov, Alexander, Stögbauer, Harald, Andrzejak, Ralph G, and Grassberger, Peter. Hierarchical clustering using mutual information. *EPL (Europhysics Letters)*, 70(2):278, 2005.

Linsker, Ralph. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988.

Madsen, Sarah, Ver Steeg, Greg, Daianu, Madelaine, Mezher, Adam, Jahanshad, Neda, Nir, Talia M., Hua, Xue, Gutman, Boris A., Galstyan, Aram, and Thompson, Paul M. Relative value of diverse brain mri and blood-based biomarkers for predicting cognitive decline in the elderly. In *SPIE Medical Imaging*, 2016.

Painsky, Amichai, Rosset, Saharon, and Feder, Meir. Generalized binary independent component analysis. In *Information Theory (ISIT), 2014 IEEE International Symposium on*, pp. 1326–1330. IEEE, 2014.

Pepke, Shirley and Ver Steeg, Greg. Multivariate information maximization yields hierarchies of expression components in tumors that are both biologically meaningful and prognostic. *bioRxiv*, 2016. doi: 10.1101/043257. URL http://www.biorxiv.org/content/early/2016/03/11/043257.

Schmidhuber, Jürgen. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

Shannon, C.E. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379423, 1948.

Simoncelli, Eero and Olshausen, Bruno. Natural image statistics and neural representation. *Annu. Rev. Neurosci.*, 24, 2001.

Steudel, Bastian and Ay, Nihat. Information-theoretic inference of common ancestors. *Entropy*, 17(4):2304–2327, 2015.

Stögbauer, Harald, Kraskov, Alexander, Astakhov, Sergey A, and Grassberger, Peter. Least-dependent-component analysis based on mutual information. *Physical Review E*, 70(6):066123, 2004.

Tishby, Naftali and Zaslavsky, Noga. Deep learning and the information bottleneck principle. *arXiv preprint arXiv:1503.02406*, 2015.

Tishby, Naftali, Pereira, Fernando C, and Bialek, William. The information bottleneck method. *arXiv:physics/0004057*, 2000.

Ver Steeg, Greg. Open source project implementing the discrete information sieve. http://github.com/gregversteeg/discrete_sieve.

Ver Steeg, Greg and Galstyan, Aram. Discovering structure in high-dimensional data through correlation explanation. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

Ver Steeg, Greg and Galstyan, Aram. Maximally informative hierarchical representations of high-dimensional data. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015. http://arxiv.org/abs/1410.7404.

Ver Steeg, Greg, Galstyan, Aram, Sha, Fei, and DeDeo, Simon. Demystifying information-theoretic clustering. In *International Conference on Machine Learning*, 2014.

Ver Steeg, Greg, Gao, Shuyang, Reing, Kyle, and Galstyan, Aram. Sifting common information from many variables. 2016.

Vincent, Pascal, Larochelle, Hugo, Bengio, Yoshua, and Manzagol, Pierre-Antoine. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103. ACM, 2008.

Watanabe, Satosi. Information theoretical analysis of multivariate correlation. *IBM Journal of research and development*, 4(1): 66–82, 1960.

Williams, P.L. and Beer, R.D. Nonnegative decomposition of multivariate information. *arXiv:1004.2515*, 2010.

Ziv, Jacob and Lempel, Abraham. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343, 1977.