Learning relevant features for statistical inference

Cédric Bény

1Department of Applied Mathematics, Hanyang University (ERICA), 55 Hanyangdaehak-ro, Ansan, Gyeonggi-do, 426-791, Korea.

We introduce an algorithm that learns correlations between two datasets, in a way which can be used to infer one type of data given the other. The approach allows for the computation of expectation values over the inferred conditional distributions, such as Bayesian estimators and their standard deviations. This is done by learning feature maps which span hyperplanes in the spaces of probabilities for both types of data, optimized to optimally represent correlations. When applied to supervised learning, this yields a new objective function which automatically provides regularization and results in faster convergence. We propose that, in addition to many applications where two correlated variables appear naturally, this approach could also be used to identify dominant independent features of a single dataset in an unsupervised fashion: in this scenario, the second variables should be produced from the original data by adding noise in a manner which defines an appropriate information metric.

I. INTRODUCTION

We consider the fundamental problem of modeling the correlations between two types of data, such as past and future, visual and auditory inputs, actions and their effects, etc.

Existing techniques perform very well at this task when one variable has a very small number of possible values (such as labels in supervised learning). They also perform relatively well in some examples where one variable is assumed to depend deterministically on the other, such as machine translation.

Here we are interested in the more general case where the amount of correlation may represent only a small part of the variability. Let us think of a dataset as consisting of independent samples from the joint distribution of two random variables $X$ and $Y$. If the mutual information $I(X:Y)$ is much smaller than the entropy of either variables, we should be able to find a compact representation of the correlated features for each variables. These representations should allow one to efficiently infer one variable given the other.

An existing option would be to use an autoencoder with a probabilistic decoder, such as a variational autoencoder (VAE) [1] or further refinements of the concept [2,3], trained to produce one variable given the other instead of the same variable (such as in Ref. [1]).

Instead, we propose here a new approach which has very different characteristics. We will refer to it as relevant feature analysis (RFA). The main design advantages over a VAE are: (i) there is no tunable variable in the objective function, (ii) it does not require parametric models of the conditional probability distributions, (iii) it allows for a direct evaluation of expectation values over the conditional distributions (but does not enable sampling from it). Moreover, (iv) it ought to converge more easily, because the gradient need not be propagated through latent variables.

Our algorithm provides a numerical solution of the theory that was introduced in Refs. [5,6] in order to understand the role of information in effective quantum field theories (in the classical setting, which is a special case of the original quantum formulation). Relations to deconvolution [7] and kernel PCA (principal component analysis) [8] were also explored by the author. Our approach may also be potentially viewed as a non-linear version of canonical correlation analysis [9]. It is also similar to the information bottleneck method [10,11] at the conceptual level, but we do not know of a formal connection so far.

One essential difference over an autoencoder is reflected in the fact that we do not need to produce a generative model (decoder). Instead, we use two “encoders”, one for each variable. The individual output neurons of these feed-forwards network represent functions over the data which we call features. Our learning objective, or loss function, effectively maximizes the amount of correlations between these features.

The resulting relevant features can then be used to do inference in both directions. Specifically, they allow one to directly evaluate the expectation of any smooth function over the conditional probabilities, such as a Bayesian estimator. In contrast, it does not naturally provide a way of sampling from the posterior.

One essential aspect of this algorithm is that the learned features do not represent just independent (i.e., “disentangled”) variables, but instead form a basis of the space of function over such variables (up to some cutoff). We conjecture that this “relaxation” of the problem makes learning easier.

RFA can also be used for supervised classification problems, where it reduces to new objective function which depends on the minibatch statistics. Remarkably, we found that it performs dramatically better than cross-entropy on unregularized networks, both in terms of convergence speed an final accuracy. In our tests, it also outperformed a network regularized using dropout and trained with cross-entropy. Adding BatchNorm layers, however, erased the advantage of RFA over cross-entropy.

The reader may jump to Section [13] for a straightforward description of the algorithm. The theoretical background and justification for the approach are provided...
in Section II. In order to provide some intuition on the nature of the approximation, we provide an analytical (gaussian) example in Section IV. Experiments with the algorithm are reported in Section V for supervised learning V A and inference V B and as autoencoder V C. We end with a general discussion and outlook in Section VI.

II. THEORY

We consider two correlated random variables $X$ and $Y$ with a joint probability distribution $p(x, y)$. We assume that we are able to numerically evaluate expectations with respect to this distribution, for instance because we can sample from it. We want to use this ability in order to compute expectations with respect to the conditional distributions $p_{X|Y}(x|y) = p(x, y)/p_Y(y)$ and $p_{Y|X}(y|x) = p(x, y)/p_X(x)$, where $p_X(x) = \sum_y p(x, y)$ and $p_Y(y) = \sum_x p(x, y)$ are the marginals of $p$. Below we sometime remove the subscripts $X$, $X|Y$ or $Y|X$ if there is no ambiguity.

For instance, suppose we generated samples of $y$ given $x$, through explicit knowledge of $p_{Y|X}$. Then the evaluation of expectations with respect to $p_{X|Y}$ is the subject of Bayesian inference. However, this is generally done in a context where the variable $X$ has low dimensionality and parameterizes a hand-crafted model. Our approach, however, is free of such a model and the variable $X$ can be of very high dimensionality.

A. Inner product on probability vectors

In order to define our strategy, we need to equip the spaces of probability distributions for $X$ and $Y$ with an inner product structure. Let us focus on $X$, and assume that it takes discrete values to avoid unnecessary technicalities. The set of probability vectors is a convex subset of the real linear space $V_X = \mathbb{R}^n$. Let us equip this space with the product

$$\langle \mu, \mu' \rangle_X := \sum_x \frac{\mu(x)\mu'(x)}{p_X(x)}$$

for any $\mu, \mu' \in V_X$. We also write $\|\mu\|_X^2 = \langle \mu, \mu \rangle_X$. Importantly, this depends explicitly on the fixed probability vector $p_X(x)$, which we took to be the marginal of $p(x, y)$. If $p_X$ has full support, this makes $V_X$ into a real inner product space. The same can be done for the variable $Y$, yielding the inner product $\langle \nu, \nu' \rangle_Y$ for $\nu, \nu' \in V_Y$.

Had we interpreted $\mu$ and $\mu'$ as tangent vectors to $V_X$, considered as a manifold, this would be the Fischer information (Riemannian) metric, as in Ref. [6]. But this quantity is also meaningful for finite vectors: the induced norm distance between $p_X$ and any probability vector $q$ is the $\chi^2$-divergence:

$$\chi^2(q, p_X) = \langle q - p_X, q - p_X \rangle_X.$$  

It measures how distinguishable $q$ is from $p_X$ in the contexts of the Pearson $\chi^2$-test. Specifically, it quantifies how easy it is to reject the null hypothesis that the state is $p_X$ when it is actually $q$, based on the empirical distribution obtained from independent samples.

The set of conditional probability distributions $p_{Y|X}$ form a stochastic map, i.e., a linear map $\mathcal{N}: V_X \rightarrow V_Y$, $\mu \mapsto \mathcal{N}(\mu)$, where

$$\mathcal{N}(\mu)(y) = \sum_x p_{Y|X}(y|x)\mu(x)$$

for any $\mu \in V_X$.

It is straightforward to check that the stochastic map $\mathcal{N}^*$ defined by

$$\mathcal{N}^*(\nu)(x) = \sum_x p_{X|Y}(x|y)\nu(x)$$

is the transpose $\mathcal{N}^*$ of $\mathcal{N}$ with respect to the inner products we defined [12], i.e., for all $\nu \in V_Y$ and $\mu \in V_X$,

$$\langle \nu, \mathcal{N}(\mu) \rangle_Y = \langle \mathcal{N}^*(\nu), \mu \rangle_X.$$  

Also, we observe that $\mathcal{N}(p_X) = p_Y$ and $\mathcal{N}^*(p_Y) = p_X$.

B. Eigen-relevance decomposition

We can use the inner products on $V_X$ and $V_Y$ to define a singular value decomposition of the stochastic map $\mathcal{N}$. That is, there is an orthonormal family $u_1, \ldots, u_k$ of $V_X$ and an orthonormal family $v_1, \ldots, v_k$ of $V_Y$, such that

$$\mathcal{N}(u_j) = \eta_j v_j,$$

for $j = 1, \ldots, k$. For each $j$, $\eta_j$ is a singular value of $\mathcal{N}$, whose square we call the relevance of the vector $v_j$. Moreover $\eta_j \in [0, 1]$ since the $\chi^2$ divergence is contractive under any stochastic map. Given that $\mathcal{N}^*$ is the transpose of $\mathcal{N}$:

$$\mathcal{N}^*(v_j) = \eta_j u_j.$$  

Equivalently, $u_j$ is an eigenvector of $\mathcal{N}^* \circ \mathcal{N}$ and $v_j$ is an eigenvector of $\mathcal{N} \circ \mathcal{N}^*$, both with eigenvalue $\eta_j^2$.

Because $\mathcal{N}$ maps $p_X$ to $p_Y$, we always have the dual eigenvectors $u_0 = p_X$ and $v_0 = p_Y$ with eigenvalue 1.

C. Low-rank approximation

Typically, the dimension $k$ of the space of probabilities is more than astronomically large. For instance, if the values of $X$ consists of small 256 gray level images of $28 \times 28$ pixels, then $k = 256^{28^2} \approx 10^{1888}$. However, in many case, only very few of these dimensions may be relevant for the purpose of inferring other variables.

The core of our approach is to approximate $\mathcal{N}$ and $\mathcal{N}^*$ by restricting them to the span of the first $k_0$ eigenvectors
u_j and v_j with largest singular values \( \eta_j \). That is, if we order the singular values \( \eta_j, j = 1, \ldots, k \) in decreasing order, we propose to use the approximations
\[
\mathcal{N}_0(\mu) = \sum_{j \leq k_0} \eta_j (u_j, \mu) X v_j
\]
with
\[
\mathcal{N}_0^*(\nu) = \sum_{j \leq k_0} \eta_j (v_j, \nu) Y u_j
\]
to \( \mathcal{N} \) and \( \mathcal{N}^* \) respectively, for some \( k_0 \) typically much smaller than \( k \), and any \( \mu \in V_X, \nu \in V_Y \).

We denote the components of \( \mathcal{N}_0 \) and \( \mathcal{N}_0^* \) by \( q(y|x) \) and \( q(x|y) \), e.g.,
\[
\mathcal{N}_0(\mu)(y) = \sum_x q(y|x) \mu(x).
\]

Since \( \mathcal{N}_0 \) and \( \mathcal{N}_0^* \) are adjoint, we can define \( q(x, y) = q(x|y)p_X(y) = q(y|x)p_X(x) \). Although the marginals of \( q(x, y) \) are the probability distributions \( p_X \) and \( p_Y \), the numbers \( q(x, y) \) are not necessarily positive.

The quality of this approximation for a given \( k_0 \) does not directly depend on the dimensionality of \( X \) and \( Y \), but only on the amount of correlations between the two variables. Our aim is to use a \( k_0 \) small enough that the components of \( \mathcal{N}_0 \) and \( \mathcal{N}_0^* \) can be computed explicitly.

**Theorem 1.** \( \mathcal{N}_0 \) is the map of rank \( k_0 \) which minimizes the average distance
\[
\sum_x p(x) \| \mathcal{N}_0(\delta_x) - \mathcal{N}(\delta_x) \| _F^2 = \sum_{x,y} \frac{(q(x, y) - p(x, y))^2}{p(x)p(y)}.
\]

**Proof.** The low rank approximation \( \mathcal{N}_0 \) minimizes the distance \( \| \mathcal{N} - \mathcal{N} \| _F \) where
\[
\| \mathcal{M} \| _F^2 = \text{Tr}(\mathcal{M}^* \mathcal{M})
\]
is the Hilbert-Schmidt (or Frobenius) norm [13]. This follows from the fact that this is also the \( l^2 \)-norm of the vector of singular values of \( \mathcal{M} \). Let us find the explicit form of the trace. Each possible value \( x \) of the variable \( X \) is associated with a probability distribution \( \delta_x(y) = 1 \) when \( x = y \) and zero otherwise. These distributions form an orthogonal basis of \( V_X \), and have norms \( \langle \delta_x, \delta_x \rangle = 1/p_X(x) \). Therefore,
\[
\text{Tr}(\mathcal{M}^* \mathcal{M}) = \sum_x p_X(x) \langle \delta_x, \mathcal{M}^* \mathcal{M}(\delta_x) \rangle_Y = \sum_x p_X(x) \| \mathcal{M}(\delta_x) \| _Y^2.
\]

**D. Features**

We express the elements \( \mu \in V_X \) and \( \nu \in V_Y \) in terms of the marginals \( p_X \) and \( p_Y \) as simple products:
\[
\mu(x) = p_X(x)f(x) \quad \text{and} \quad \nu(y) = p_Y(y)g(y)
\]
for all \( x, y \), where \( f \) and \( g \) are real functions of \( x \) and \( y \) which we call features.

The inner products then simply become correlations among features. Using also \( \mu' = p_X f' \) and \( \nu' = p_Y g' \), we obtain
\[
\langle \mu, \mu' \rangle_X = \sum_x p_X(x)f(x)f'(x) = \bar{f}\bar{f}',
\]
\[
\langle \nu, \nu' \rangle_Y = \sum_y p_Y(y)g(y)g'(y) = \bar{g}\bar{g}'.
\]

These are simple expectation values with respect to \( p \), which we assumed is the type of quantity we can evaluate for arbitrary functions \( f, f', g, g' \).

Since \( \mathcal{N}^* \mathcal{N} \) is self-adjoint in terms of this inner product, its eigenvectors \( u_i \) are orthogonal, and hence the corresponding features \( a_i \) defined by \( u_i(x) = p_X(x)a_i(x) \) are uncorrelated. Indeed,
\[
\langle u_i, u_j \rangle_X = 0 \quad \text{for all } i, j.
\]

for all \( i, j \). Moreover, accounting for the eigenvector \( u_0 = p_X \) (corresponding to the constant feature \( a_0(x) = 1 \) for all \( x \)),
\[
\bar{a}_i = 0
\]
\[
\text{for all } i \neq 0. \quad \text{Hence we trivially have}
\]
\[
\langle u_i, u_j \rangle = \bar{a}_i \bar{a}_j
\]
\[
\text{for all } i, j \neq 0.
\]

Likewise for the eigenvectors of \( \mathcal{N}^* \mathcal{N} \). If \( v_i(y) = p_Y(y)b_i(y) \):
\[
\langle v_i, v_j \rangle_Y = 0 = \bar{b}_i \bar{b}_j.
\]

for all \( i, j \neq 0 \).

Importantly, this does not mean that the features \( u_1, u_2, \ldots \) nor \( v_1, v_2, \ldots \) are “disentangled”, i.e., they are not statistically independent. These features represent components in the space of probability vectors, rather than the “sample” space. They should be understood as spanning a subspace of the space of functions over the relevant independent variables. We discuss this in more detail in Section [IV.C].

**E. Corners of \( \mathcal{N} \) and loss function**

The final piece of puzzle we need, is the ability to express the components (corners) of \( \mathcal{N} \) and \( \mathcal{N}^* \) in the span of possible non-orthogonal families of features.
Let us therefore consider two arbitrary families $f_1, \ldots, f_{k_0}$ and $g_1, \ldots, g_{k_0}$ of features, which respectively represent the vectors $p_X f_j \in V_X$ and $p_Y g_j \in V_Y$.

Firstly, we need matrices representing the components of the inner products on $V_X$ and $V_Y$. Those are the symmetric matrices

$$K_{ij} = \langle p_X f_i, p_X f_j \rangle_X = \delta_{ij},$$

$$L_{ij} = \langle p_Y g_i, p_Y g_j \rangle_Y = \delta_{ij}. \tag{21}$$

The components $N_{ij}$ of $N$ are defined by

$$N(p_X f_j) = \sum_i N_{ij} p_Y g_i. \tag{23}$$

Taking the inner product with $p_Y g_k$, we obtain

$$\langle p_Y g_k, N(p_X f_j) \rangle = \sum_i N_{ij} L_{ki}. \tag{24}$$

The left-hand side can be computed using Equ. 3 It is the matrix

$$A_{kj} = \langle p_Y g_k, N(p_X f_j) \rangle = \sum_{x,y} \frac{p_Y(y) g_k(y) p_Y|X(y|x)p_X(x) f_j(x)}{p_Y(y)} \tag{25}$$

$$= \sum_{x,y} p(x, y) g_k(y) f_j(x) = \langle g_k f_j \rangle.$$

Therefore, in matrix notation, Equ. 24 is $A = LN$, or

$$N = L^{-1} A. \tag{26}$$

The components $N^*_{ij}$ of $N^*$ are obtained by just swapping $X$ and $Y$, yielding

$$N^* = K^{-1} A^\top. \tag{27}$$

Hence the singular values of the corner of $N$ defined by the features $f_j$ and $g_j$ are just the square-root of the eigenvalues of the matrix $N^* N = K^{-1} A^\top L^{-1} A$. In order to find the features $f_j$ and $g_j$ with the same span as the first $k_0$ eigenvectors $u_i, v_j$, we just need to maximize all the eigenvalues of $N^* N$. A simple way to do this is to use (minus) the trace of $N^* N$ as loss function, since it is the sum of the square of the singular values. We call Tr ($N^* N$) the relevance of the subspaces defined by the features $f_j$ ad $g_i$ for all $i,j$. This yields the loss/cost function:

$$C = k_0 - \text{Tr} (N^* N) = k_0 - \text{Tr} (K^{-1} A^\top L^{-1} A). \tag{28}$$

Once optimal features have been found, one can obtain the components of the eigenvectors in the span of $f_1, \ldots, f_{k_0}$ through standard numerical diagonalization of $N^* N$.

### F. Inference

The features minimizing $C$ can be used to infer one variable from the other. For instance, given $y$, the inferred probability distribution over $x$ is given by $p_{X|Y}(x|y) = N^*(\delta_y)(x)$, where $\delta_y(y') = 1$ when $y = y'$ and zero otherwise. In order to compute this, we first need the components of the distribution $\delta_y$ in terms of the family $p_Y g_1, \ldots, p_Y g_{k_0}$, i.e., the real numbers $(\delta_y)_j$ such that

$$\delta_y(y') = p_Y(y') \sum_{i=1}^{k_0} (\delta_y)_i g_i(y') + r(y'), \tag{29}$$

where $\langle r, p_Y \delta_i \rangle_Y = 0$ for all $i$. Taking the inner product with $p_Y g_j$, we obtain

$$\langle p_Y g_j, \delta_y \rangle_Y = \sum_{i=1}^{k_0} (\delta_y)_i L_{ji}. \tag{30}$$

Therefore the components of $\delta_y$ are explicitly

$$(\delta_y)_i = \sum_j (L^{-1})_{ij} g_j(y). \tag{32}$$

It follows that

$$p_{X|Y}(x|y) = N^*(\delta_y)(x) \approx N_{0i}^*(\delta_y)(x)$$

$$= \sum_{i,j,k} N^*_{ki}(L^{-1})_{ij} g_j(y) f_k(x). \tag{33}$$

Then, for instance, the expected inferred value of $X$ is

$$\overline{x} = \sum_{i,j,k} N^*_{ki}(L^{-1})_{ij} g_j(y) \sum_x p_X(x) x f_k(x). \tag{34}$$

For the inference of $Y$ from $x$, we have

$$p_{Y|X}(y|x) \approx \sum_{i,j,k} N_{ki}(K^{-1})_{ij} f_j(x) g_k(y). \tag{35}$$

### III. ALGORITHM

Let us briefly summarize the algorithm resulting from the above analysis.

We assume that we have the ability to compute expectation values of functions with respects to a joint distribution $p(x, y)$. For concreteness, let us assume that we can do so by sampling from this distribution, or that we have access to a dataset of samples.

We need two independent deterministic feed-forward neural networks. The first maps $x$ to a set of $k_0$ real-valued features $f_1(x), \ldots, f_{k_0}(x)$. The second maps $y$ to a different set of $k_0$ features $g_1(y), \ldots, g_{k_0}(y)$. 

The parameters of the neural networks are to be set to minimize the loss function

$$C = k_0 - \text{Tr} \left( K^{-1} A^T L^{-1} A \right)$$  \hspace{1cm} (36)$$

where the matrices $K, L, A$ can be approximated over a mini-batch $(x_n, y_n), n = 1, \ldots, N$ via

$$K_{ij} = \frac{1}{N} \sum_{n=1}^{N} f_i(x_n) f_j(x_n)$$  \hspace{1cm} (37)$$

$$L_{ij} = \frac{1}{N} \sum_{n=1}^{N} g_i(x_n) g_j(x_n)$$  \hspace{1cm} (38)$$

$$A_{ij} = \frac{1}{N} \sum_{n=1}^{N} g_i(x_n) f_j(x_n).$$  \hspace{1cm} (39)$$

To compute the gradient of $C$, we use the fact that for any parameter $w$ used to compute $L$, the gradient of $L^{-1}$ is given by the matrix product

$$\frac{\partial}{\partial w} L^{-1} = -L^{-1} (\frac{\partial}{\partial w} L) L^{-1}.$$  \hspace{1cm} (40)$$

Once the features have been learned, we still need to use the training data in a second step. Indeed, suppose that we wish to use our model to infer the value of some function $\Theta(x)$, i.e., to compute its approximate expectation value in terms of the conditional distribution $x \mapsto p(x|y)$. Then we need to store, for each feature $j = 1, \ldots, k_0$, the quantities

$$\Theta_j = \frac{1}{N_{\text{full}}} \sum_{n=1}^{N_{\text{full}}} \Theta(x_n) f_j(x_n).$$  \hspace{1cm} (41)$$

Where the average is to be taken on the full training batch (of size $N_{\text{full}}$). The same can be done exchanging $x$ with $y$ and $f_j$ with $g_j$ for the reverse inference.

For instance, if we are interested in the Bayesian estimator for an $L^2$ distance, then we need at least the expectation values of the real components $\Theta(x) = x^\alpha$ of the data, and possibly higher moments to gain more knowledge about the shape of the posterior distribution, such as the second moments $\Theta(x) = x^\alpha x^\beta$, etc.

Inference can then be performed with new data following Equ. (34):

$$\bar{\Omega} = \sum_x p(x|y) \Theta(x) \approx \sum_{j=1}^{k_0} \Theta_j (K^{-1} A^T L^{-1} g(y))_j,$$  \hspace{1cm} (42)$$

where $g(y)$ denotes the column vector with components $g_1(y), \ldots, g_{k_0}(y)$.

Conversely, the expected value of a function $\Sigma$ of the variable $y$ given $x$ is given by

$$\bar{\Sigma} = \sum_y p(y|x) \Sigma(y) \approx \sum_{j=1}^{k_0} \Sigma_j (L^{-1} A K^{-1} f(x))_j,$$  \hspace{1cm} (43)$$

where $\Sigma_j = \frac{1}{N_{\text{full}}} \sum_{n=1}^{N_{\text{full}}} \Sigma(x_n) g_j(y_n)$.

### A. Alternative interpretation of the loss

If we write $F_{ij} := f_j(x_i)$ and $G_{ij} := g_j(y_i)$ for the value of the features on the dataset, then $K = \frac{1}{N} F^T F$, $L = \frac{1}{N} G^T G$ and $A = \frac{1}{N} G^T F$. The relevance can then be written as

$$\text{Tr} \left( K^{-1} A^T L^{-1} A \right) = \text{Tr} \left( P Q \right)$$

where $P = F(F^T F)^{-1} F^T$ and $Q = G(G^T G)^{-1} G^T$ are the projectors on the ranges of $F$ and $G$ respectively. Hence, we are maximizing the overlap between those ranges (which represents possible linear combinations of datapoints, respectively determined from features of one or the other correlated variables.)

This makes it clear that the size of the minibatches needs to be larger than the number of features.

### B. Heuristic

**Constant features**—The loss function $C$ takes value between 0 and $k_0 - 1$ because the constant feature always has relevance 1. The constant feature could be enforced a priori rather than learned, which, due to the objective, automatically forces the learned features to have zero expectation values (be orthogonal to the constant feature). This might have advantages in certain circumstances, but in our experiments we found that this sometime hindered convergence.

**Invertibility issues**—The covariance matrices $K$ and $L$ can be ill-conditioned, potentially causing the gradient to “explode” because of the inverses $K^{-1}$ et $L^{-1}$ involved in the loss function. This can be avoided by replacing $K^{-1}$ by $(K + \epsilon I)^{-1}$ in the loss for some small positive number $\epsilon$, which may be decreased or set to zero in later epochs—and likewise for $L^{-1}$.

**Symmetries in the loss function**—The loss $C$ only depends on the span of the features $f_i$ and $g_j$, hence it has a very large group of symmetries. In particular, it is invariant under a change of the norm of each features independently from each other. Because of that, these norms can slowly drift far apart, eventually causing numerical instabilities. Hence, for long runs it may be necessary to also add a small extra cost to prevent that. For instance, one may use the loss function

$$C' = C + \alpha \sum_i |K_{ii} - 1| + \alpha \sum_j |L_{jj} - 1|.$$  \hspace{1cm} (44)$$

We found however that this can interfere with convergence (maybe by trapping the model in local minima), so that $\alpha$ needs to be very small.

**Regularization**—In all our tests, dropout had no beneficial effect. In fact, our objective seems to already provide a form of regularization, as shown in Section.
IV. GAUSSIAN EXAMPLE

If \( p(x, y) \) is gaussian, everything can be computed analytically. For instance, let us consider the probability distribution

\[
p_X(x) \propto \exp\left(-\frac{x^2}{2\sigma^2}\right),
\]

and the conditional

\[
p_{Y|X}(y|x) \propto \exp\left(-\frac{(y-x)^2}{2\sigma^2}\right).
\]

That is, \( y \) is equal to \( x \) but with some added gaussian noise. This gives

\[
p_{X|Y}(x|y) \propto \exp\left(-\frac{(x-\gamma y)^2}{2\tau^2(1-\gamma)}\right),
\]

where

\[
\gamma := \frac{\tau^2}{\sigma^2 + \tau^2}.
\]

It was show in Ref. [5], that the most relevant subspace of dimension \( k_0 \) on the variable \( X \) is simply spanned by the features

\[
f_n(x) = x^n, \quad n = 0, \ldots, k_0 - 1.
\]

Similarly for \( Y \):

\[
g_n(y) = y^n.
\]

This independance of the relevant features on the detailed parameters of \( p \) is a general property of gaussian joint distributions.

This means, for instance, that the most relevant feature (\( n = 1 \)) for predicting the value of \( X \) given \( Y = y \) is simply \( Y \) itself. The higher order features have to do with inferring extra aspects of the probability distribution over \( X \).

As set of orthogonal features can be obtain from the Gram-Schmidt procedure, which, if done from small to large \( n \) much necessarily yield the eigenvectors \( u_n \) and \( v_n \). For illustration purpose, let us work with the non-orthogonal vectors \( f_n \) and \( g_n \), keeping only the first \( k_0 = 3 \) vectors.

The three matrices (correlators) we need can be easily computed:

\[
K = \begin{pmatrix}
1 & 0 & \tau^2 \\
0 & \tau^2 & 0 \\
\tau^2 & 0 & 3\tau^4
\end{pmatrix},
\]

\[
L = \begin{pmatrix}
1 & 0 & \tau^2 + \sigma^2 \\
0 & \tau^2 + \sigma^2 & 0 \\
\tau^2 + \sigma^2 & 0 & 3(\tau^2 + \sigma^2)^2
\end{pmatrix},
\]

\[
A = \begin{pmatrix}
1 & 0 & \tau^2 \\
0 & \tau^2 & 0 \\
\tau^2 + \sigma^2 & 0 & \tau^2(\sigma^2 + 3\tau^2)
\end{pmatrix}.
\]

We obtain

\[
M = K^{-1}A^\top L^{-1}A = \begin{pmatrix}
1 & \tau^2(1 - \gamma^2) \\
0 & \gamma & 0 \\
0 & 0 & \gamma^2
\end{pmatrix}.
\]

The eigenvalues of \( M \) can be read on the diagonal, and the corresponding eigenvectors are \((1, 0, 0), (0, 1, 0)\) and \((-\tau^2, 0, 1)\), which means that the eigen-features are in order \( u_0(x) = 1, v_1(x) = x \) and \( u_2(x) = x^2 - \tau^2 \).

Let us see how the approximate inference using only the three most relevant features fares. Given the value \( y \) for \( Y \), the inferred distribution over \( X \) is, from Eq. (34),

\[
N_0^Y(\delta_Y)(x) = p_X(x)p_Y(y) \sum_{j,k=0}^{2}(K^{-1}A^\top L^{-1})_{jk}y^jx^k.
\]

The approximately inferred first and second moments of \( X \) is given by integrating the above times \( x \) (resp. \( x^2 \)) over \( x \). We obtain

\[
\bar{x} = \gamma y \quad \text{and} \quad \bar{x^2} = \gamma^2 y^2 + (1 - \gamma)\tau^2,
\]

which are actually exact: they are equal to the first two moments of \( X \) over \( p_{X|Y} \) as given in Eq. (47). I this case, this would be true for the first \( k_0 - 1 \) moments had we kept the \( k_0 \) most relevant features.

V. EXPERIMENTS

For all our experiments, we used the Flux package [14] for Julia, and the built-in ADAM optimizer with default parameters (\( \eta = 0.001, \beta_1 = 0.9, \beta_2 = 0.999 \)).

As usual the data is divided into a training set and a testing set. No aspect of the testing set is used during training. The loss function refers to Eq. (44). In order to monitor overfitting, we compute a “test loss” and a “training loss”. The test loss is computed from the trained features using only the test data, and accordingly, the training loss is computed purely using the training data.

Moreover, when performing inference on test data using Eq. (42), we use the covariances \( A, L, K \) built from the training data only.

A. Supervised learning

In the context of a supervised classification task, one of the dataset (the labels) is of sufficiently low dimensionality that we can use a complete basis over its probability space as our features. This serves as a good first sanity test for our approach. Surprisingly, we find that RFA converges faster than standard approaches, and without the need for regularization.

Let the variable \( Y \) stands for the labels, with values in \( \{1, \ldots, n\} \). The probability space consists of vectors
FIG. 1. Loss and inaccuracy (error rate) on test sets for two classification tasks. The models were trained either using the cross-entropy (CE) or our objective (RFA), with or without regularization layers. On the MNIST dataset, we used an “all CNN” network, and for the CIFAR10 dataset we used a short VGG variation with 10 convolutions and 3 fully connected layers. In the regularized form, post-activation BatchNorm layers were placed after each convolutional layers on the VGG network. What is shown is the mean over 10 independent runs for MNIST and 5 runs for CIFAR10. The shaded area spans the standard deviation. ADAM with default parameters was used in all cases. No data augmentation was used except for horizontal flips for CIFAR10 (resulting in epochs of 100,000 images).

with \( n \) real components. The canonical basis corresponds to the “one hot” encoding \( g_i(j) = \delta_{ij} \) (Kronecker delta). All we need is a neural network to encode \( n \) features \( f_1, \ldots, f_n \) on \( X \). After learning the most relevant features \( f_i \). We apply Eq. (13) with \( \Sigma(y) = y \), and use the maximum component of expected value \( \mathbf{y} \) to infer the labels from the data.

We tested this approach on the standard MNIST dataset, using a small convolutional net which performs well using standard supervised learning with a cross entropy objective. Specifically, it reaches near 0.5% accuracy using dropout regularization, which is optimal without expansion of the training data or model averaging [15]. The same network (but without dropout) can be used in our approach (RFA); what differs is the nature of the loss function, and the method to obtain predictions (Fig. 1 top row).

We also compared our approach to BatchNorm regularization on the CIFAR10 dataset [16], using a variation of the network VGG16, namely with the last 3 convolutional layers and the last max pool removed to accommodate the small size of the images (Fig. 1 bottom row).

We found that simply changing the objective from cross-entropy to RFA provided a very significant improvement both of convergence speed and final accuracy for both models.

On MNIST, the unregularized network also converged faster with RFA than the same network with our best dropout configuration and cross-entropy objective.

In all our tests, dropout would always degrade performance in conjunction with the RFA objective. Using BatchNorm layers on the CIFAR example, however, essentially erased any distinction between RFA and cross-entropy.

These tests shows that RFA has a strong regularizing effect, and appears to have no disadvantages over using the cross entropy classification objective.

The code containing all parameters used can be found in [17].

B. Inference

In this next experiment, we used the left and right halves of the MNIST digit images as correlated variables \( X \) and \( Y \). The goal is to obtain the expected left halves given the right halves, or vice versa.

The features were represented by two identical convolutional neural networks with the same architecture as in the previous section. The losses during training, and optimal eigenvalues are represented in Fig. 3.

These were used into Eq. (42) to compute the mean pixel gray value and their covariances over the conditional probability of \( X \) given \( Y \) on test data. This mean is the Bayesian estimator for the \( l^2 \) distance between half images, i.e., it should minimize the expected distance \( d_{\Sigma}^2 \) over the conditional distribution, where \( d_{\Sigma}^2(x,y) = \sum_i(x_i - y_i)^2 \), where \( x_i \in [0,1] \) is the value of the \( i^{th} \) pixel. (This is equivalent to minimizing the mean square error).

The results are shown in Fig. 2. For each examples, we also computed the images obtained by adding plus or minus one standard deviation along the direction of greatest variance in the space of features. This reveals the main ambiguities (such as between 8 and 3 or 7 and 9 which share a similar right half).

In this figure, we clipped the pixel values between what corresponds to white and black. In reality, since our approach enforces no constraint on the pixel values, the predictions typically err below or above those values.

C. Feature extraction

Beyond modeling the correlations between two different types of data, we examine if, following an idea proposed in Ref. [8], this approach could be used on a single dataset in the manner of an autoencoder in order to ex-
FIG. 2. The left halves of the MNIST digits are inferred from the right half. For each of a random samples of test digits, the middle image represents the mean over pixel intensity of the inferred condition distribution. Left and right images correspond to a plus and minus one standard deviation from the mean in the direction of largest covariance (in the space of half-images). We used two independent convolutional neural nets to learn 120 pairs of features. The average \( l^2 \) distance between the original and mean image over all 10,000 test images is 3.276.

FIG. 3. Left: loss during training for the features used in Fig. 2. We used the features for which the test loss was smallest (epoch 45, with a test loss of 16.43). Right: relevances for the learned 120 features, in decreasing order.

FIG. 4. Top-left: First 20 eigenrelevance features (on \( X \)) determined by our algorithm for a system where \( X \) consists of two coordinates uniformly sampled over a circle and a surrounding ring, and \( Y \) consists of the same points but shifted by a small normally distributed vector. The features are arranged from left-to-right and top-to-bottom in order of decreasing relevance. Top-right: the same features multiplied by the marginal \( p_X \). Bottom row: introducing a gap in the ring allows for a monotonous function of the angle to serve as second most relevant variable (instead of the sine/cosine couple). Hence the angle is automatically “disentangled” from the other variables. (Mid-gray represents the value 0).

Indeed, the \( \chi^2 \) metric alone is purely information-theoretic and does not “know” anything about the nature of the data. In fact, it assigns an infinite distance between any data points since they corresponds to distributions with disjoint support.

A reasonable metric on pixel intensity can be accounted for in the information metric \( d_\pi \) if we use a map \( \pi \) which randomizes pixel intensity by addition gaussian noise for instance. Similarly \( d_\pi \) can be made to account for locality in an image by making \( \pi \) also randomize pixel location, randomly swapping nearby pixels.

In this setting, the RFA algorithm determines those features of \( X \) which lose the least distinguishability under the coarse-graining \( \pi \). For instance, the example of Section IV can be interpreted in this setting, where \( X \) is a normally distributed real variable, and \( \pi \) adds a gaussian noise to this variable, encoding the real metric into the problem. RFA then selects the low-order polynomials in \( X \) as most relevant variable, which matches our intuition that the low-order moments are most robust to this kind of noise. Moreover, the most relevant variable is \( x \) itself.

A more complex numerical example is shown in Fig. 4. Here, \( X \) consists of two real numbers, distributed uniformly within a ring and a disk. The stochastic map \( \pi \) introduces the Euclidean distance by adding a random gaussian noise to \( X \). The inner and outer radii of the

\[
d_\pi(p, q) := \chi^2(\pi(p), \pi(q)). \tag{57}
\]
disk are 1.5 and 2.5. The radius of the inner disk is 0.8, and the standard deviation of the noise is 0.2.

We would expect the relevant independent variables to be: the binary variable indicated whether the point is in the disk or the ring and the angle around the ring, followed by the radial component inside the ring, and finally the cartesian coordinates inside the disk. This is precisely what we see in Fig. 4. As in the gaussian example, the relevant features span the algebra of functions of those variables, and are ordered by how fast they vary in the euclidean metric (the wavelength encodes the degree of precision of the corresponding variable).

Let us put aside for now the salient fact that the angle itself is not represented (likely because it is discontinuous). Besides the constant function, the two most relevant variables are the sine and cosine of the angle, followed by the binary variable separating the disk from the ring.

But these features ought to span the space of probabilities over the relevant variables, not just the variables themselves. Hence the next 6 features are sines and cosines of smaller wavelength, which can encode probability distributions which are increasingly more precisely localized, down to a precision (wavelength) comparable with the diameter of the inner disk. Accordingly, the next two features are the Cartesian coordinates inside the disk. This is followed by additional moments of the angle down a wavelength equal to the ring’s thickness, at which point we see the radius in the ring appear.

In the bottom row of Fig. 4, we also show what happens if we let a gap in the ring that is bigger than the standard deviation of the noise: this allows for the angle along the ring (or at least a monotous function of it) to appear as most relevant variables (rather than its sine and cosine).

From the purpose of obtaining statistically independent—or disentangled—features of the dataset, this structure is a little bit of a problem: all these moments are uncorrelated by not independent. Technically, the features form an orthogonal basis of an algebra of function over the independent random variables. What we are looking for are independent generators of the algebra. Since we are able to concretely compute the inner product between all features, we could attempt to find those variables through algebraic methods. For instance, we can numerically evaluate all of the structure constants. However we did not so far find a satisfactory method to extract clean generators in this way.

One potential issue demonstrated by the complete ring example, is that the independent generators themselves may not always be part of—or even in the span of—the most relevant features. Indeed, in that example, the sine and cosine of the angle are not statistically independent. We would need instead the angle itself, which, due to its discontinity, is a linear combination of waves of arbitrarily small wavelengths. However, a good enough approximation to the angle can—if measured in terms of our metric—be necessarily obtained as linear combination of our relevant features.

Generative model—Alternatively, if we postulate that the independent relevant parameters can be found in the linear span of the relevant features, we can attempt to extract them by optimizing a neural network composed of two parts. Firstly, a linear layer maps the relevant features to a small number of outputs (equal to the latent dimension). These latent variables are then processed by a generative network to produce a possible value of the variable $X$. As objective function, we may use an appropriate measure of similarity between the output and the data element from which the features were obtained. This optimization ought to extract a minimal generative subspace as output of the first linear layer.

We tested this idea as follows. First, we used RFA on the MNIST images as samples of the variable $X$, together with noisy versions of the same images for the variable $Y$. The noise map $\pi$ that we used simulates the coarse-graining channel introduced in Ref. 13. It consists of two noise components: (a) Firstly, pairs of random neighbours (pixels which share an edge) are selected randomly and permuted, up to a number of independent permutation equal to the total number of pixels, and (b) a gaussian noise is added to the pixel values, with standard deviation $\sigma = 10\%$ of the normal range, and independently for each pixel. The $\chi^2$ metric applied to these coarse-grained images results in a metric which accounts for spatial proximity and pixel-value proximity.

The features or the original images were produced by the same convolutional neural network as in Section V A, while the features of the coarse-grained images were extracted by a network of the same geometry, but with half the number of filters and neurons.

We extracted the 1000 most relevant out of 1200 learned features in this way. (The least relevant features in this system happen to be highly dependent on the total number of features and hence cannot be trusted to be correct). As a second step, we trained a linear layer coupled to a network composed of 5 fully-connected layers of 800 hidden neurons each. We refer to the number
of output neurons in the first linear layer as the latent dimension.

As input, this network received the features extracted from MNIST images using the above convolutional neural net (after it was fully trained using RFA), and was trained to minimize the mean square error between its output and the original MNIST digit.

The resulting best mean square errors are shown in Fig. 5 as function of the latent dimension. In a logarithmic plot, the slope gives the largest exponent of polynomial. Here we see a distinct change of behaviour (slope) at dimension 19. Increasing the dimension further provides no improvement. This behaviour is compatible with our hypothesis that the extra features are just functions of those first dozen features (functions which are effectively re-implemented by the generative network).

Images generated by sampling from a gaussian approximation of the latent distribution for different latent dimensions are shown in Fig. 6. Below dimension 20, most generated image can be recognized as a specific digit, but they are significantly worse looking than samples of the original dataset. This could be caused by the generators of the algebra not being statistically independent (such as in the ring example). However, there are many other parameters at play such as: the power of the neural networks used, the appropriateness of the coarse-graining for this task, an insufficient number of features, gaussians being a bad approximations of the latent distributions, etc.

VI. DISCUSSION

We only brushed a few potential applications of RFA. More detailed studies will be needed to see if it can be used to obtain state-of-the art results for inference, disentangled feature extraction or for data generation. The surprising regularizing effect in supervised learning also warrants further study.

One of the important features of RFA that we not stressed in our experiments is the fact that the resulting model allows for the direct evaluation of the expectation values in the posterior distribution without sampling. In particular this allows for the evaluation of credible intervals. Hence it should be especially suited to scientific applications where the ability to quantify uncertainty is essential.

The fact that RFA is grounded in a consistent information-theoretical analysis which comes with a large class of analytically solvable examples (namely all Gaussian joint distributions, using the methods from Ref. [19]), presents exciting potentials for further developments of the approach.

Another interesting aspect of this approach is that the theory it is based on has a complete quantum-theoretical formulation [5]. Hence it would be natural to find extensions of the present work to contexts where the data is quantum.

ACKNOWLEDGMENTS

We would like to thank Joël Bény and Raban Iten for helpful suggestions. This work was supported by the National Research Foundation of Korea (NRF-2018R1D1A1A02048436).

[1] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. (arXiv:1312.6114) 2013.
[2] Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In International Conference on Learning Representations, volume 3, 2017. [semanticscholar.org].
[3] Ricky TQ Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in vae's. 2018.
[4] Raban Iten, Tony Metger, Henrik Wilming, Lidia Del Río, and Renato Renner. Discovering physical concepts with neural networks. (arXiv:1807.10300) 2018.
[5] Cédric Bény and Tobias J Osborne. Renormalisation as an inference problem. (arXiv:1310.3188) 2013.
[6] Cédric Bény and Tobias J Osborne. The renormalisation group via statistical inference. New J. Phys., 17:083005, 2015. (arXiv:1402.4949).
[7] Cédric Bény. Quantum deconvolution. Quantum Information Processing, 17(2):26, 2018. (arXiv:1708.03215).
[8] Cédric Bény. Inferring relevant features: from qft to pca. International Journal of Quantum Information,
[9] Harold Hotelling. Relations between two sets of variates. In *Breakthroughs in statistics*, pages 162–190. Springer, 1992.

[10] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. (arXiv:physics/0004057) 2000.

[11] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE, 2015. (arXiv:1503.02406)

[12] M. Ohya and D. Petz. *Quantum entropy and its use*. Springer Verlag, 2004.

[13] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[14] Mike Innes. Flux: Elegant machine learning with julia. *Journal of Open Source Software*, 2018.

[15] Yann LeCun, Corinna Cortes, and Christopher J C Burges. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/ Accessed: 2019-04-12.

[16] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, 2009. [www.cs.toronto.edu]

[17] Cédric Bény. RFA examples source code. http://github.com/cbeny/RFA Accessed: 2019-04-19.

[18] C. Bény and T. J. Osborne. Information geometric approach to the renormalisation group. *Phys. Rev. A*, 92:022330, 2015. (arXiv:1206.7004)

[19] Cédric Bény. Coarse-grained distinguishability of field interactions. *Quantum*, 2:67, 2018. (arXiv:1509.03249)