LEARNING RELEVANT FEATURES FOR STATISTICAL INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce an new technique to learn correlations between two types of data. The learned representation can be used to directly compute the expectations of functions over one type of data conditioned on the other, such as Bayesian estimators and their standard deviations. Specifically, our loss function teaches two neural nets to extract features representing the singular probability vectors of highest singular value for the stochastic map (set of conditional probabilities) implied by the joint dataset, relative to the inner product defined by the Fisher information metrics evaluated at the marginals. We test the approach using a synthetic dataset, analytical calculations, and inference on occluded MNIST images. Surprisingly, when applied to supervised learning (one dataset consists of labels), this approach automatically provides regularization and faster convergence compared to the cross-entropy objective. We also explore using this approach to discover salient independent features of a single dataset.

1 INTRODUCTION

Further progress in artificial intelligence requires algorithms which can learn to model unlabeled data (unsupervised learning). Here we assume that the data naturally comes in two parts, such as past and future histories, visual and auditory inputs, actions and their effects, etc. Our goal is to produce a useful model of the correlations which exist between both variables. By contrast, existing techniques perform well in assymetric situation, to predict one variable that has a very small number of possible values (such as labels in supervised learning) or that depends almost deterministically on the other.

An existing option would be to use an autoencoder with a probabilistic decoder, such as a variational autoencoder (VAE) (Kingma & Welling, 2013) or further refinements of the concept (Higgins et al., 2017; Chen et al., 2018), trained to produce one variable given the other instead of the same variable (such as in Iten et al. (2018)).

Instead, we propose here a new approach which has very different characteristics. We will refer to it as *relevant feature analysis* (RFA). Some of the 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 doesn't directly enable sampling from it). Moreover, (iv) the gradient need not be propagated through latent variables, which ought to ease training.

This last point results from the fact that we do not need to learn a generative model (decoder). Instead, we use two deterministic encoders, one for each variable. The individual output neurons of these feed-forwards networks 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 of the relevant features of one variable conditioned on the other variable taking a specific value.

Moreover, if the number of learned features is sufficiently large for the problem, then one can estimate the conditional expectation of any "smooth"[1] function over the conditional probabilities.

2 THEORY

Our approach is based on a theory introduced in Bény & Osborne (2013; 2015b) for understanding the role of information in effective quantum field theories, but restricted here to the classical non-quantum setting. Relations to deconvolution Bény (2018b) and kernel PCA Bény (2018c) were also previously explored.

We formalize the problem by assuming that our data was sampled from an unknown joint distribution $p(x, y)$ over two random variables X and Y.

Let V_X and V_Y denotes the linear spaces spanned by the probability distributions over X and Y respectively.

The joint distribution $p(x, y)$ yields conditionals $p(y|x)$ and $p(x|y)$, which can be understood as the components of stochastic matrices, defining linear maps $\mathcal{N}$ and $\mathcal{N}^*$ between V_X and V_Y. If $\mu \in V_X$ and $\nu \in V_Y$, then the images $\mathcal{N}(\mu) \in V_Y$ and $\mathcal{N}^*(\nu) \in V_X$ are defined by

$$\mathcal{N}(\mu)(y) = \sum_x p(y|x)\mu(x) \quad \text{and} \quad \mathcal{N}^*(\nu)(x) = \sum_y p(x|y)\nu(y). \tag{1}$$

These stochastic maps $\mathcal{N}$ and $\mathcal{N}^*$ perform inference of one variable given some (possibly imperfect) knowledge about the other, with priors given by the marginals $p(x)$ or $p(y)$ of $p(x, y)$ depending on the direction of the inference.

We want to approximate $\mathcal{N}$ and $\mathcal{N}^*$ by setting their smaller *singular values* to zero.

However, in order to defined the singular value decomposition, we need inner products on V_X and V_Y. For that purpose, we use the Fisher information metrics evaluated at the points $p(x)$ and $p(y)$ respectively (marginals of $p(x, y)$), that is,

$$\langle \mu, \mu' \rangle_X := \sum_x \frac{\mu(x)\mu'(x)}{p(x)} \quad \text{and} \quad \langle \nu, \nu' \rangle_Y := \sum_y \frac{\nu(y)\nu'(y)}{p(y)} \tag{2}$$

for any probability distributions $\mu, \mu' \in V_X$ and $\nu, \nu' \in V_Y$.

A beauty of this choice is that it makes $\mathcal{N}^*$ the *transpose* of $\mathcal{N}$:

$$\langle \nu, \mathcal{N}(\mu) \rangle_Y = \langle \mathcal{N}^*(\nu), \mu \rangle_X. \tag{3}$$

It follows that $\mathcal{N}$ and $\mathcal{N}^*$ have the same set of singular values, which are the square roots of the eigenvalues of $\mathcal{N}^*\mathcal{N}$. The singular vectors of $\mathcal{N}$ and $\mathcal{N}^*$ are respectively the left- and right- eigenvectors of $\mathcal{N}^*\mathcal{N}$. Moreover, the singular values are all in the interval $[0, 1]$ because the Fisher metric contracts under any stochastic map.

Another useful fact is the following: if we write $\mu(x) = p(x)f(x)$ and $\nu(y) = p(y)g(y)$ for all x, y (we call the function f and g *features*), then the inner products become simple covariances:

$$\langle \mu, \mu' \rangle_X := \sum_x p(x)f(x)f'(x) \quad \text{and} \quad \langle \nu, \nu' \rangle_Y := \sum_y p(y)g(y)g'(y), \tag{4}$$

Moreover, it is easy to check that

$$\langle \nu, \mathcal{N}(\mu) \rangle_Y := \sum_{xy} p(x, y)f(x)g(y). \tag{5}$$

These three covariances can be approximated straightforwardly by averages on the data.

If we have neural network producing some arbitrary functions $f_1(x), \ldots, f_k(x)$ and $g_1(y), \ldots, g_k(y)$, then we can use the above three equations to compute the components of $\mathcal{N}$ in the

[1] In the sense that their expectation values over the training data should be reliable. E.g., they cannot be peaked on a single data point.

subspaces spanned by the corresponding functions $\mu_i(x) = p(x)f_i(x)$ and $\nu_j(y) = p(y)g_j(y)$, i.e., the numbers N_{ij} such that $\mathcal{N}(\mu_j) = \sum_j N_{ij}\nu_i$. Indeed, using $A_{ij} = \langle \nu_i, \mathcal{N}(\mu_j)\rangle_Y$ and $L_{ij} = \langle \nu_i, \nu_j\rangle_Y$, we have $N = L^{-1}A$ (using the matrix inverse and matrix product). Similarly we can define the components N^*_{ij} of $\mathcal{N}^*$, which are given by $N^* = K^{-1}A^\top$ where $K_{ij} = \langle \mu_i, \mu_j\rangle_X$.

We observe that $\mathrm{Tr}\,(N^*N)$ is the sum of the square of the singular values of $\mathcal{N}$ restricted to the span of the functions μ_i and ν_i. Maximizing $\mathrm{Tr}\,(N^*N)$ over the parameters of the networks then yields the subspaces of V_X and V_Y spanned by the k singular vectors of $\mathcal{N}$ with largest singular values. We refer to the corresponding features as the k *most relevant features*.

Since we know explicitly the components of $\mathcal{N}$ and $\mathcal{N}^*$ with respect to these subspaces, we can also perform any inference using these projected forms of the channels.

Of course, This approach can produce a faithful representation of the correlations only if $\mathcal{N}$ is actually close to being of rank k (see Appendix B for a more precise statement). If we interpret the relevant subspace as a space of probability over latent variable, this means that our latent variables have at most k discrete states.

However, even if the rank-k projection on $\mathcal{N}$ is a not a good approximation, this strategy allows us to do *exact* inference on certain random variables, namely those which are in the span of the relevant features!

Indeed, $\mathcal{N}$ maps any of the most k relevant functions $\mu_i(x)$ to the span of the most k relevant functions $\nu_i(y)$, i.e., for any $i = 1, \dots, k$, $\mathcal{N}(\mu_i) = \sum_{j=1}^k c_j\nu_j$. Hence the expectation value of the feature $f_i(x) = \frac{1}{p(x)}\mu_i(x)$ conditioned on $Y = y^*$ is given by

$$\begin{aligned}\overline{f_i} &= \sum_x p(x|y^*)f_i(x) = \sum_x \frac{1}{p(x)}p(x|y^*)\mu_i(x) = \langle \mathcal{N}^*(\delta_{y^*}), \mu_i\rangle_X = \langle \delta_{y^*}, \mathcal{N}(\mu_i)\rangle_Y \\ &= \frac{1}{p(y^*)}\mathcal{N}(\mu_i)(y^*) = \sum_{j=1}^k c_j g_j(y^*).\end{aligned} \tag{6}$$

The role of the variables X and Y can be exchanged in this result by the swaps $\mathcal{N} \leftrightarrow \mathcal{N}^*$, $f_i \leftrightarrow g_i$.

For instance, if $p(x, y)$ is Gaussian, the singular vectors can be computed analytically following Bény & Osborne (2013); Bény (2018a) (Appendix A). Notably, for any two dimensional Gaussian, the space of k most relevant features is simply spanned by the moments $f_n(x) = x^n$ and $g_n(y) = y^n$ for $n = 0, \dots, k-1$. Hence, in this case the first k moments can be inferred exactly using only $k+1$-dimensional subspaces of features (See Appendix C).

3 RFA Algorithm

Let us explicit the algorithm resulting from the above analysis.

We assume that we are given independent samples $(x_1, y_1), (x_1, y_2), \dots$ from the otherwise unknown joint distribution $p(x, y)$.

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

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

$$C = k_0 - \mathrm{Tr}\,(K^{-1}A^\top L^{-1}A), \tag{7}$$

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

$$K_{ij} = \frac{1}{N}\sum_{n=1}^N f_i(x_n)f_j(x_n), \quad L_{ij} = \frac{1}{N}\sum_{n=1}^N g_i(x_n)g_j(x_n), \quad A_{ij} = \frac{1}{N}\sum_{n=1}^N g_i(x_n)f_j(x_n). \tag{8}$$

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 it's

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

$$\Theta_j = \frac{1}{N_{\text{full}}} \sum_{n=1}^{N_{\text{full}}} \Theta(x_n) f_j(x_n), \tag{9}$$

where the average is to be taken on the full training batch (of size N_{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 $\Theta(x) = x$ of the data, and possibly higher moments to gain more knowledge about the shape of the posterior distribution, such as the second moments $\Theta'(x) = x^2$, etc.

Inference can then be performed with new data using

$$\overline{\Theta} = \sum_x p(x|y)\, \Theta(x) \;\approx\; \sum_{i,j=1}^{k_0} (K^{-1} A^\top L^{-1})_{ji} \Theta_j g_i(y). \tag{10}$$

Moreover, the accuracy of these predictions does not depend on the rank k_0 if Θ is taken in the span of the relevant features, i.e., $\Theta(x) = \sum_{i=1}^{k} c_i f_i(x)$, for which $\Theta_j = \sum_{i=1}^{k_0} c_i K_{ij}$.

The reverse inference formulas are obtained simply by the exchanges $K \leftrightarrow L$, $A \leftrightarrow A^\top$, and $g_i \leftrightarrow f_i$.

4 Experiments

For all our experiments, we used the Flux package (Innes, 2018) for Julia, and the built-in ADAM optimizer with default parameters.

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. (7). 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. (10), we use the covariances A, L, K and expectations Θ_j (Equ. (9)) built from the training data only.

4.1 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 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 the reverse of Eq. (10) for function $\Theta(y) = y$, and use the maximum component of expected value $\overline{y}$ to infer the labels from the data.

We tested this approach on the MNIST and CIFAR10 datasets, and compared the results to the standard cross-entropy objective (Fig. 1).

We found that, without regularization, simply changing the objective from cross-entropy to RFA provided a large improvement both of convergence speed and final accuracy for both models.

On MNIST, RFA alone also outperformed cross-entropy with dropout. (Dropout did not yield any improvement in conjunction with RFA). However, adding batch-normalization layers on the CIFAR example, erased any distinction between RFA and cross-entropy.

The code containing all parameters used can be found in (Bény), including a Tensorflow/Keras implementation.

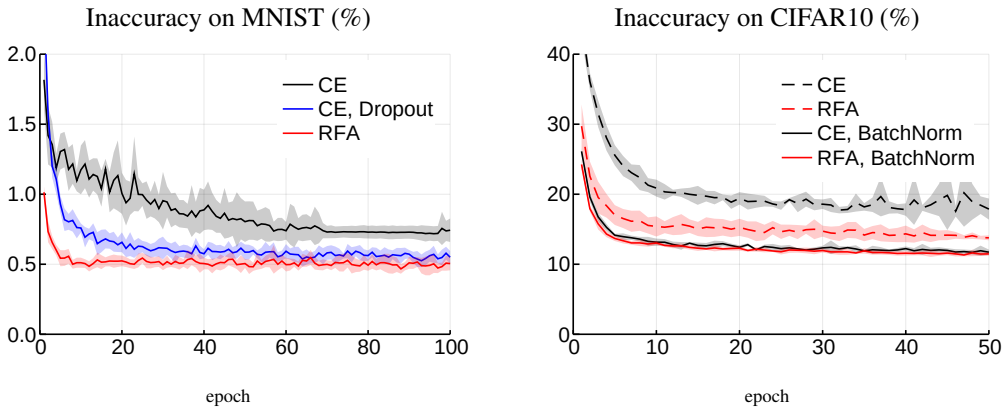

Figure 1: Loss and inaccuracy (error rate) on test sets for two classification tasks. The models were trained either using the cross-entropy (CE) or our objective (RFA), with or without regularization layers. On the MNIST dataset, we used an "all CNN" network, and for the CIFAR10 dataset we used a short 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).

4.2 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.

After training the features, we used the training dataset to also compute the expected pixel gray value as well as their covariance for each feature using Eq. (9).

These were used into Eq. (10) to compute the mean pixel gray values 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_{l^2} over the conditional distribution, where $d^2_{l^2}(x, y) = \sum_i (x_i - y_i)^2$, where $x_i \in [0, 1]$ is the value of the ith pixel. (This is equivalent to minimizing the mean square error).

The results are shown in Fig. 2. For each example, 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).

The graph of the singular values shows that the rank cutoff of 120 is too low to capture all of the relevant features, but the results are reasonable nevertheless.

An IPython notebook reproducing this result and more can be found in (Bény).

4.3 Feature discovery

We mentioned in Section 2 that if $p(x, y)$ is a two-dimensional Gaussian distribution with zero mean, then the n most relevant features of X are the first n powers of X itself, independently of the covariance matrix. This implies that the singular features are the Hermite polynomials in X (which results from applying the Gram-Schmidt procedure to the basis $\{1, x, x^2, \dots\}$).

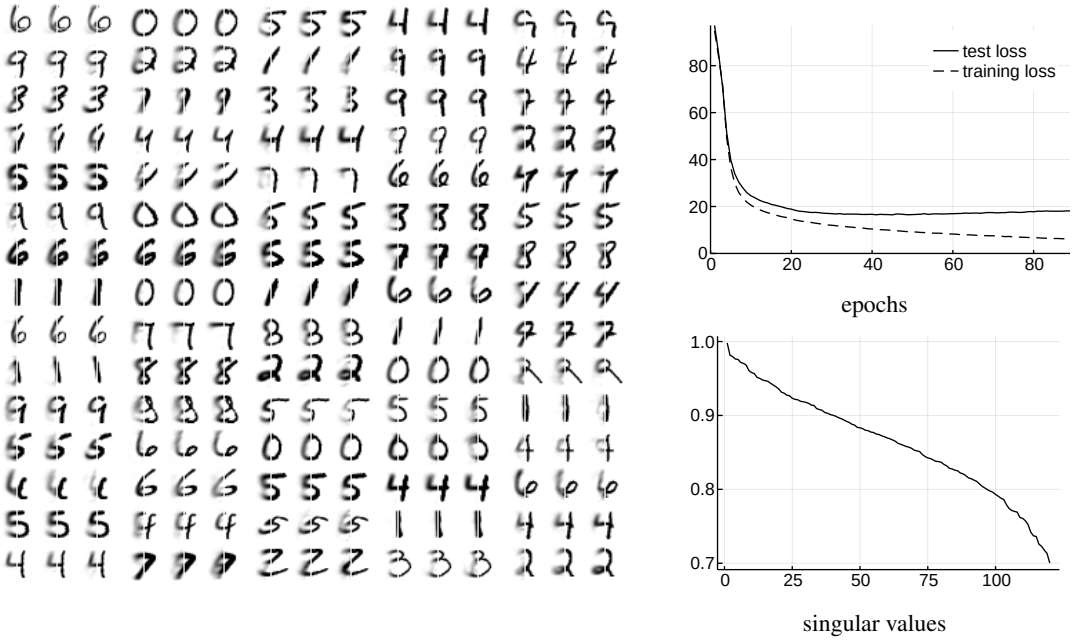

Figure 2: The left halves of the MNIST digits in the test set are inferred from the right half. Three images are shown for each random sample. The middle image represents the mean over pixel intensity of the inferred condition distribution. Left and right images corresponds to a plus and minus one standard 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. We used the features for which the test loss was smallest (epoch 45, with a test loss of 16.43). The average l^2 distance between the original and mean image over all $10,000$ test images is 3.276.

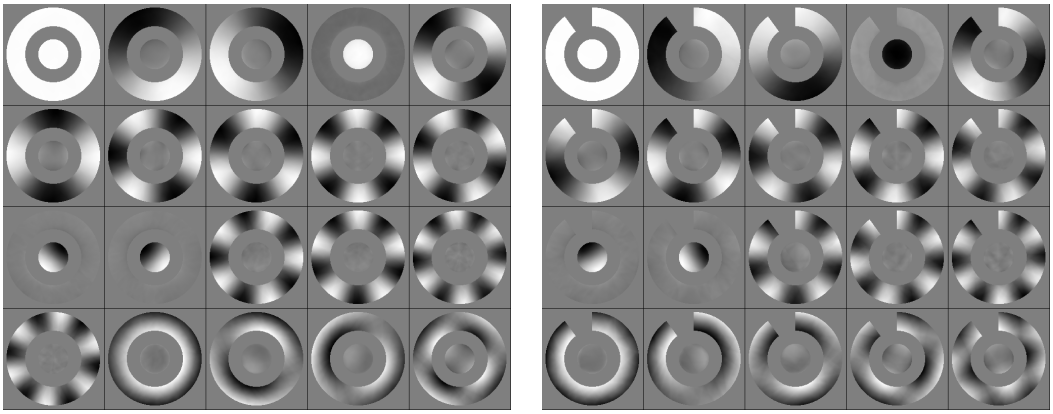

Figure 3: *Top-left*: First 20 eigenrelevance features (on X) determined by our algorithm for a system where X consists of two coordinates uniformly sampled over a circle and a surrounding ring, and Y consists of the same points but shifted by a small normally distributed vector. The features are arranged 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).

A similar property holds for multivariate Gaussians, namely, the less relevant singular values are polynomials in the more relevant ones. If this is true more generally, it should be possible to further compress and organize the latent space extracted with RFA by finding a minimal set of generators, which ought to also be in the span of the most relevant features.

We applied RFA to a synthetic dataset to explore this idea, shown in Fig. 3. Here, X consists of two real numbers, distributed uniformly within a ring and a disk. The variable Y is obtained by adding a random Gaussian shift to X with a small standard deviation. The more relevant features ought to be those which are more robust to such small random displacement. This formalizes the idea that we are interested in extracting "large-scale" features.

We would expect the relevant independent variables to be: the binary variable indicating whether the point is in the disk or the ring and the angle around the ring, followed by the radial component in the ring, and finally the Cartesian coordinates inside the disk. This is precisely what we see in Fig. 3.

Indeed—if we put aside for now the fact that the angle itself is not directly represented—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 six 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 most relevant features are the Cartesian coordinates inside the disk. This is followed by additional moments of the angle, down to a wavelength equal to the ring's thickness, at which point we see the radius in the ring appear.

As mentioned, we see that the angle itself is not represented, likely because it is discontinuous. However, as shown also in Fig. 3, creating a gap in the ring allows for the angle to emerge as most relevant variable. This suggests that this approach may be able to automatically learn intrinsic coordinates of the latent variable manifold.

4.4 Disentangled features and generative model

If we postulate that the independent (or disentangled) relevant latent variables 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*). The purpose of this linear layer is to find the independent variables. These latent variables are then processed by an arbitrarily complex generative network to produce a possible value of the variable X. As objective function, we may us an appropriate measure of similarity between the output and the data element from which the features were obtained.

We tested this idea as follows. We took X to consist of the MNIST digits, and produced Y by randomly permuting neighboring pixels in the image, until the mean displacement per pixel is of order 1. In addition, we added independent Gaussian noise to the pixel values. (Hence the noise map $\mathcal{N}$ simulates the coarse-graining channel introduced in Bény & Osborne (2015a)).

As in the previous experiment, we do so to implement our intuition that the more relevant features ought to be the ones which are of larger scale, or more robust to local perturbations.

The features of the clean images were produced by the same convolutional neural network as in Section 4.1, 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*.

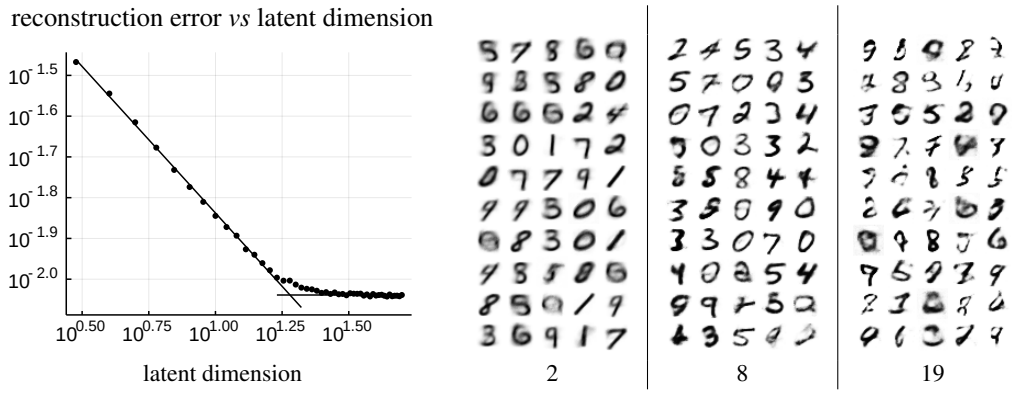

Figure 4: Left: Best mean squared error for images reconstructed from a subspace of the features, as a function of the subspace dimension. This logarithmic plot shows that improvements stop once the dimension reaches 19 (where the two lines cross). Right: images produced by the generator from latent variables sampled according to the best Gaussian fit in latent space, for feature subspaces of dimensions 2, 8 and 19.

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. 4, as function of the latent dimension. Here we see a distinct change of polynomial scaling law 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 twenty 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. 4. Below dimenson 20, most generated image can be recognized as a specific digit.

5 Outlook

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 regularizing effect for supervised learning also warrants further study.

We have not explored potential applications of one of the salient aspect of RFA, namely the fact that it provides the functions which can be most reliably predicted. To see why this is potentially significant, we observe that a central feature of scientific exploration is that the properties of a system which can be predicted and understood are not given a priori, but must be discovered.

Another important feature of RFA 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), presents exciting potentials for further refinements and developments of the approach.

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

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).

REFERENCES

C. Bény and T. J. Osborne. Information geometric approach to the renormalisation group. *Phys. Rev. A*, 92:022330, 2015a. doi: 10.1103/PhysRevA.92.022330. (arXiv:1206.7004).

Cédric Bény. RFA examples source code. `http://github.com/cbeny/RFA`. Accessed: 2019-09-22.

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

Cédric Bény. Quantum deconvolution. *Quantum Information Processing*, 17(2):26, 2018b. (arXiv:1708.03215).

Cédric Bény. Inferring relevant features: from qft to pca. *International Journal of Quantum Information*, 16:1840012, 2018c. (arXiv:1802.05756).

Cédric Bény and Tobias J Osborne. Renormalisation as an inference problem. *(arXiv:1310.3188)*, 2013.

Cédric Bény and Tobias J Osborne. The renormalisation group via statistical inference. *New J. Phys.*, 17:083005, 2015b. doi: 10.1088/1367-2630/17/8/083005. (arXiv:1402.4949).

Ricky TQ Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in vaes. 2018.

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

Irina Higgins, Loic 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).

Mike Innes. Flux: Elegant machine learning with julia. *Journal of Open Source Software*, 2018. doi: 10.21105/joss.00602.

Raban Iten, Tony Metger, Henrik Wilming, Lídia Del Rio, and Renato Renner. Discovering physical concepts with neural networks. *(arXiv:1807.10300)*, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *(arXiv:1312.6114)*, 2013.

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

A EXTRA INFORMATION ABOUT THE ALGORITHM

A.1 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 $K = \frac{1}{N}F^\top F$, $L = \frac{1}{N}G^\top G$ and $A = \frac{1}{N}G^\top F$. The relevance can then be written as

$$\mathrm{Tr}\,(K^{-1}A^\top L^{-1}A) = \mathrm{Tr}\,(PQ)$$

where $P = F(F^\top F)^{-1}F^\top$ and $Q = G(G^\top G)^{-1}G^\top$ 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.)

A.2 HEURISTIC

Batch size—In our experiments, we observed that the batch size during training needs to be an order of magnitude larger than the number of features (rank cutoff). When the batch size was too small, learning seemed to converge normally in terms of training and test loss, but resulted in features which yield dramatically different losses when evaluated on larger batches, and yield spurious predictions.

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 either by using the Moore-Penrose pseudo-inverse, or by replacing K^{-1} by $(K + \epsilon \mathbf{1})^{-1}$ in the loss for some small positive number ϵ, 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, it is preferable not to have a linear last layer. Using a hyperbolic tangent as last nonlinearity worked in our experiments.

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 4.1.

B THEORY IN MORE DETAILS

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 $p_{X|Y}(x|y) = p(x, y)/p_X(x)$ and $p_{Y|X}(y|x) = p(x, y)/p_Y(y)$, 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.

B.1 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

$$\langle \mu, \mu' \rangle_X := \sum_x \frac{\mu(x)\mu'(x)}{p_X(x)} \tag{11}$$

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 μ and μ' as tangent vectors to V_X, considered as a manifold, this would be the Fisher information (Riemannian) metric, as in Bény & Osborne (2015b). But this quantity is also meaningful for finite vectors: the induced norm distance between p_X and any probability vector q is the χ^2-divergence:

$$\chi^2(q, p_X) = \langle q - p_X, q - p_X \rangle_X. \tag{12}$$

It measures how distinguishable q is from p_X in the contexts of the Pearson χ^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 $\mathcal{N} : V_X \to V_Y$, $\mu \mapsto \mathcal{N}(\mu)$, where

$$\mathcal{N}(\mu)(y) = \sum_x p_{Y|X}(y|x)\mu(x) \tag{13}$$

for any $\mu \in V_X$.

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

$$\mathcal{N}^*(\nu)(x) = \sum_x p_{X|Y}(x|y)\nu(x) \tag{14}$$

is the *transpose* $\mathcal{N}^*$ of $\mathcal{N}$ with respect to the inner products we defined (Ohya & Petz, 2004), i.e., for all $\nu \in V_Y$ and $\mu \in V_X$,

$$\langle \nu, \mathcal{N}(\mu) \rangle_Y = \langle \mathcal{N}^*(\nu), \mu \rangle_X. \tag{15}$$

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

B.2 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

$$\mathcal{N}(u_j) = \eta_j v_j, \tag{16}$$

for $j = 1, \ldots, k$. For each j, η_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 χ^2 divergence is contractive under any stochastic map. Given that $\mathcal{N}^*$ is the transpose of $\mathcal{N}$:

$$\mathcal{N}^*(v_j) = \eta_j u_j. \tag{17}$$

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

B.3 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 28×28 pixels, then $k = 256^{28^2} \simeq 10^{1888}$. 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 η_j. That is, if we order the singular values η_j, $j = 1, \ldots, k$ in decreasing order, we propose to use the approximations

$$\mathcal{N}_0(\mu) = \sum_{j \leq k_0} \eta_j \langle u_j, \mu \rangle_X v_j \tag{18}$$

$$\mathcal{N}_0^*(\nu) = \sum_{j \leq k_0} \eta_j \langle v_j, \nu \rangle_Y u_j \tag{19}$$

$$\tag{20}$$

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.,

$$\mathcal{N}_0(\mu)(y) = \sum_x q(y|x)\mu(x). \tag{21}$$

Since $\mathcal{N}_0$ and $\mathcal{N}_0^*$ are adjoint, we can define $q(x,y) = q(x|y)p_Y(y) = q(y|x)p_X(x)$. 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*

$$\sum_x p(x)\|\mathcal{N}_0(\delta_x) - \mathcal{N}(\delta_x)\|_Y^2 = \sum_{xy} \frac{(q(x,y) - p(x,y))^2}{p(x)p(y)}. \tag{22}$$

Proof. The low rank approximation $\mathcal{N}_0$ minimizes the distance $\|\mathcal{N}_0 - \mathcal{N}\|_{\mathrm{F}}$ where

$$\|\mathcal{M}\|_F^2 = \mathrm{Tr}\,(\mathcal{M}^*\mathcal{M}) \tag{23}$$

is the Hilbert-Schmidt (or Frobenius) norm (Eckart & Young, 1936). 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,

$$\begin{aligned}\mathrm{Tr}\,(\mathcal{M}^*\mathcal{M}) &= \sum_x p_X(x)\langle \delta_x, \mathcal{M}^*\mathcal{M}(\delta_x)\rangle_Y \\ &= \sum_x p_X(x)\|\mathcal{M}(\delta_x)\|_Y^2\end{aligned}$$

□

B.4 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:

$$\mu(x) = p_X(x)f(x) \quad \text{and} \quad \nu(y) = p_Y(y)g(y) \tag{24}$$

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

$$\langle \mu, \mu' \rangle_X = \sum_x p_X(x)f(x)f'(x) = \overline{ff'}, \tag{25}$$

$$\langle \nu, \nu' \rangle_Y = \sum_y p_Y(y)g(y)g'(y) = \overline{gg'}. \tag{26}$$

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,

$$\overline{a_i a_j} = \langle u_i, u_j \rangle_X = 0, \tag{27}$$

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),

$$\overline{a}_i = 0 \tag{28}$$

for all $i \neq 0$. Hence we trivially have

$$\overline{a_i a_j} = \overline{a}_i \overline{a}_j \tag{29}$$

for all $i, j \neq 0$.

Likewise for the eigenvectors of $\mathcal{N}\mathcal{N}^*$. If $v_i(y) = p_Y(y) b_i(y)$:

$$\overline{b_i b_j} = \langle v_i, v_j \rangle_Y = 0 = \overline{b}_i \overline{b}_j. \tag{30}$$

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 4.3.

B.5 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

$$K_{ij} = \langle p_X f_i, p_X f_j \rangle_X = \overline{f_i f_j}, \tag{31}$$

$$L_{ij} = \langle p_Y g_i, p_Y g_j \rangle_Y = \overline{g_i g_j}. \tag{32}$$

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

$$\mathcal{N}(p_X f_j) = \sum_i N_{ij} p_Y g_i. \tag{33}$$

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

$$\langle p_Y g_k, \mathcal{N}(p_X f_j) \rangle = \sum_i N_{ij} L_{ki}. \tag{34}$$

The left-hand side can be computed using Equ. 13. It is the matrix

$$\begin{aligned} A_{kj} &= \langle p_Y g_k, \mathcal{N}(p_X f_j) \rangle \\ &= \sum_{x,y} \frac{p_Y(y) g_k(y) p_{Y|X}(y|x) p_X(x) f_j(x)}{p_Y(y)} \\ &= \sum_{x,y} p(x,y) g_k(y) f_j(x) = \overline{g_k f_j}. \end{aligned} \tag{35}$$

Therefore, in matrix notation, Equ. (34) is $A = LN$, or

$$N = L^{-1} A. \tag{36}$$

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

$$N^* = K^{-1} A^\top. \tag{37}$$

Hence the singular values of the corner of $\mathcal{N}$ 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_j, 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 $\mathrm{Tr}\,(N^* N)$ the *relevance* of the subspaces defines by the features f_j ad g_i for all i, j. This yields the loss/cost function:

$$C = k_0 - \mathrm{Tr}\,(N^* N) = k_0 - \mathrm{Tr}\,(K^{-1} A^\top L^{-1} A). \tag{38}$$

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 $N^* N$.

B.6 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 $p_{X|Y}(x|y) = \mathcal{N}^*(\delta_y)(x)$, where $\delta_y(y')$ is 1 when $y = y'$ and zero otherwise. In order to compute this, we first need the components of the distribution δ_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

$$\delta_y(y') = p_Y(y') \sum_{i=1}^{k_0} (\delta_y)_i g_i(y') + r(y'), \tag{39}$$

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

$$\langle p_Y g_j, \delta_y \rangle_Y = \sum_{i=1}^{k_0} (\delta_y)_i L_{ji}, \tag{40}$$

where the left hand side is also just

$$\langle p_Y g_j, \delta_y \rangle_Y = g_j(y). \tag{41}$$

Therefore the components of δ_y are explicitly

$$(\delta_y)_i = \sum_j (L^{-1})_{ij} g_j(y). \tag{42}$$

It follows that

$$\begin{aligned} p_{X|Y}(x|y) &= \mathcal{N}^*(\delta_y)(x) \approx \mathcal{N}_0^*(\delta_y)(x) \\ &= \sum_{ijk} N^*_{ki} (L^{-1})_{ij} g_j(y) f_k(x). \end{aligned} \tag{43}$$

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

$$\overline{x} = \sum_{ijk} N^*_{ki} (L^{-1})_{ij} g_j(y) \sum_x p_X(x) x f_k(x). \tag{44}$$

For the inference of Y from x, we have

$$p_{Y|X}(y|x) \approx \sum_{ijk} N_{ki} (K^{-1})_{ij} f_j(x) g_k(y). \tag{45}$$

C ANALYTICAL EXAMPLE

When $p(x, y)$ is any multivariate Gaussian distribution, everything can be computed analytically. Let us consider here the one-dimensional case. We use $p(x) \propto \exp\left(-x^2/2\tau^2\right)$, and the conditional $p(y|x) \propto \exp\left(-(y-x)^2/2\sigma^2\right)$. That is, y is equal to x but with some added Gaussian noise. This gives

$$p_{X|Y}(x|y) \propto \exp\left(-\frac{(x-\gamma y)^2}{2\tau^2(1-\gamma)}\right), \quad \text{where} \quad \gamma = \frac{\tau^2}{\sigma^2+\tau^2}. \tag{46}$$

It was show in Bény & Osborne (2013), that the most relevant subspace of dimension k_0 on the variable X is simply spanned by the features

$$f_n(x) = x^n, \tag{47}$$

$n = 0, \ldots, k_0 - 1$. Similarly for Y;

$$g_n(y) = y^n. \tag{48}$$

This independence 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.

A 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:

$$K = \begin{pmatrix} 1 & 0 & \tau^2 \\ 0 & \tau^2 & 0 \\ \tau^2 & 0 & 3\tau^4 \end{pmatrix} \qquad L = \begin{pmatrix} 1 & 0 & \tau^2+\sigma^2 \\ 0 & \tau^2+\sigma^2 & 0 \\ \tau^2+\sigma^2 & 0 & 3(\tau^2+\sigma^2)^2 \end{pmatrix} \tag{49}$$

$$A = \begin{pmatrix} 1 & 0 & \tau^2 \\ 0 & \tau^2 & 0 \\ \tau^2+\sigma^2 & 0 & \tau^2(\sigma^2+3\tau^2) \end{pmatrix}. \tag{50}$$

We obtain

$$M = K^{-1}A^\top L^{-1}A = \begin{pmatrix} 1 & 0 & \tau^2(1-\gamma^2) \\ 0 & \gamma & 0 \\ 0 & 0 & \gamma^2 \end{pmatrix}. \tag{51}$$

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 $u_0(x) = 1$, $u_1(x) = x$ and $u_2(x) = x^2 - \tau^2$.

Because we are working with continuous variables, the true rank of $\mathcal{N}$ is infinite, even for any finite cutoff on the singular values. Nevertheless, it is instructive to see how the approximate inference fares for rank $k_0 = 3$. Given the value y for Y, the inferred distribution over X is

$$\mathcal{N}_0^*(\delta_y)(x) = p_X(x)p_Y(y)\sum_{j,k=0}^{2}(K^{-1}A^\top L^{-1})_{kj}y^j x^k. \tag{52}$$

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

$$\overline{x} = \gamma\, y \quad \text{and} \quad \overline{x^2} = \gamma^2 y^2 + (1-\gamma)\tau^2, \tag{53}$$

which are actually exact: they are equal to the first two moments of X over $p_{X|Y}$ as given in Eq. (46).

In fact, it is easy to see that this would be true for the first $k_0 - 1$ moments had we kept the k_0 most relevant features.