Supervised learning with information penalties

Artemy Kolchinsky
Santa Fe Institute
Massachusetts Institute of Technology
artemy@santafe.edu

David H. Wolpert
Santa Fe Institute
Massachusetts Institute of Technology
Arizona State University
david.h.wolpert@gmail.com

Abstract

We propose to find models that have low prediction error about targets while also making their predictions using compressed intermediate representation of inputs. We call our approach "supervised learning with information penalties" [SLIP]. SLIP can be useful when intermediate representations are transmitted over a low-bandwidth channel from a sender to a receiver, after which the receiver must make statistical predictions about the targets. We show a fundamental connection between SLIP and the information bottleneck method, and argue that our approach provides a novel way to perform information bottleneck in continuous, non-linear spaces. We implement SLIP using a noisy neural network in combination with a differentiable non-parametric entropy estimator. This implementation is demonstrated on simple problems using feed-forward and recurrent networks.

1 Introduction

Broadly speaking, the goal of supervised learning is to learn a mapping from inputs to outputs, such that it is then possible to make accurate predictions about outputs when given novel inputs. In this paper, we introduce an extension of supervised learning in which the goal is to learning mappings that not only offer accurate predictions, but also do so by using a compressed intermediate representation of the inputs. We call this extension *supervised learning with information penalties* [SLIP].

SLIP is useful when inputs are collected at one location, predictions are made at another location, and the two locations are connected by a low-capacity channel. As an illustration, consider a remote weather station which makes detailed recordings of various meteorological variables (wind-speed, precipitation, etc.). These measurements are used by a central server to make predictions about the probabilities of different weather conditions for the next day. If the channel between the weather station and server has low capacity, there are three ways one may attempt to implement this scenario:

1. The weather station compresses the meteorological data using some (lossless or lossy compression) algorithm. We assume that the compression is not tailored to the prediction task, but rather to accurate reconstruction of the meteorological data. This compressed description is then sent to the server, where it is uncompressed. Finally, the server runs a weather-prediction model that takes meteorological data as input.

2. The weather station compresses the meteorological data into some intermediate representation. This compressed intermediate representation is created with the prediction task in mind; thus, the compression algorithm will throw away details of the meteorological data that are irrelevant for prediction. The intermediate representation is transmitted to the server, and the server runs a weather-prediction model that uses the intermediate representation as input. This is the SLIP approach.

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

3. The weather station runs locally a weather-prediction model which uses the raw meteorological data as input. It then transmit to the server a compressed description of the resulting prediction.

In Appendix A, we show that approach #3 is formally equivalent to SLIP (approach #2). We also show that SLIP will be better (and in most cases, strictly better) than approach #1 in terms of the amount of information that needs to be transmitted while offering optimal weather predictions.

SLIP also has applications for prediction of temporal sequences, where recurrent neural networks [RNNs] are often employed [1–4]. To use the example above, we might consider the case where the server runs a dynamical system in order to predict weather trajectories across multiple days, with initial conditions determined by data received from the weather station. Here SLIP provides a way to find an RNN that offers accurate prediction of weather trajectories on the server, while minimizing the number of bits that need to be transmitted by the weather station.

SLIP has other advantages besides predictions constrained by an intermediate low-capacity channel. For example, because it only extracts information from the input that is relevant for predicting the output, it can be used for feature selection and extraction, and for exploratory analysis of data. Furthermore, for finite training data, SLIP has the additional advantage of functioning as an information-theoretic regularizer; in some cases, this leads to improved generalization performance, relative to supervised learning without regularization. However, in this work we do not fully explore these aspects of SLIP.

In the next section, we discuss the conceptual approach behind SLIP.

We then provide details of our implementation of SLIP for finite, continuous-valued data. Our implementation combines several important elements, including adaptive levels of stochastic noise and a differentiable non-parametric estimator of entropy.

In section 4, we relate SLIP to previous work in machine learning, including research auto-encoders and regularization. We also discuss the close relationship between SLIP and the *information bottleneck* [IB] method [5]. The goals of the two approaches are similar: like SLIP, IB takes a distribution over inputs and outputs and produces a compressed intermediate representation which is good for predicting outputs. The major difference is the following: while both approaches make use of a 'decoding map' from intermediate representations to output predictions, in SLIP finding a good decoding map is part of the optimization problem, while IB assumes that the globally-optimal decoding map is available. This difference is important because the globally-optimal decoding map may be very hard to compute, making IB impractical for some important scnearios. Possibly for this reason, IB has so far only been implemented for random variables that are discrete-valued [5] or continuous-valued but linearly-related [6]. As we show, optimizing the SLIP cost function is practical, and formally equivalent to minimizing an upper bound on the IB cost function. Thus, SLIP can be used to perform IB over continuous-valued, non-linearly-related random variables, which has never been done before.

In section 5, we provide some preliminary results of SLIP on a digit classification task.

2 Proposed approach

2.1 Definitions

Capital letters X usually indicate random variables, though can also indicate sets of outcomes, constants, or functions, as will be clear from context. Latin letters x indicate outcomes of random variables; lowercase Greek letters α are used to indicate parameters.

For information-theoretic functions, $H(\cdot)$ indicates Shannon entropy, $H(\cdot|\cdot)$ indicates conditional entropy, $C(\cdot\|\cdot)$ indicates the cross-entropy function, and $D(\cdot\|\cdot)$ indicates Kullback-Leibler [KL] divergence. Given conditional distributions $P_1(A = a|B = b)$ and $P_2(A = a|B = b)$, we use $C_{A|B}(P_1\|P_2)$ to indicate conditional cross-entropy, and $D_{A|B}(P_1\|P_2)$ to indicate conditional KL divergence. $MI_P(A; B)$ indicates the mutual information between random variables A and B which are jointly distributed according to $P(A, B)$ (subscript P is left off when clear from context). All logs are base-2 and all information-theoretic quantities are in units of bits.

Throughout this work, we assume that inputs and outputs are distributed according to some joint distribution $Q(x, y)$, with marginals indicated by $Q(y)$ and $Q(m)$. We use the conditional probability distribution $P_\theta(m|x)$ to indicate the *encoding map* from inputs from *messages*, where θ is a vector of parameters, and the conditional probability distribution $P_\phi(y|m)$ to indicate the *decoding map* from inputs from messages, where ϕ is a vector of parameters (these are described in more detail in the next section).

We make use of the following notation:

$$Q_\theta(y, m) := \int Q(x, y)\, P_\theta(m|x)\, dx \tag{1}$$

$$Q_\theta(x, m) := Q(x) P_\theta(m|x) \tag{2}$$

$$Q_\theta(m) := \int Q(x) P_\theta(m|x)\, dx. \tag{3}$$

2.2 SLIP

Consider the following scenario: inputs $x \in X$ and outputs $y \in Y$ are distributed according to joint distribution $Q(x, y)$, and there are two agents, a *sender* and a *receiver*, connected by a communication channel. A sample input-output pair x, y is drawn from Q; then, the input x is provided to the sender, while neither the sender nor the receiver are provided with y. The sender chooses a message m according to the encoding map $P_\theta(m|x)$, and then transmits m to the receiver over the communication channel. The receiver uses a decoding map $P_\phi(y|m)$ (parameterized by ϕ) to make a prediction about the probabilities of different values of y.

For a given input x, the values of y will be distributed according to $Q(y|x)$. The error of the receiver's predictions $P_\phi(y|m)$ can be measured via the cross-entropy of the two distributions:

$$C(Q(Y|x) \| P_\phi(Y|m)) := -\int Q(y|x) \log P_\phi(y|m)\, dy \tag{4}$$

For a fixed distribution Q, cross-entropy is minimized when $P_\phi(y|m) = Q(y|x)$ for almost all y. We also define the *expected* error of the receiver's prediction using the *conditional cross-entropy*, i.e., the expectation of the cross-entropy in Eq. (4) across all inputs x and messages m:

$$\begin{aligned} C_{Y|M}(Q_\theta \| P_\phi) &:= \int Q(x)\ P_\theta(m|x)\ C(Q_\theta(Y|x) \| P_\phi(Y|m))\, dx\, dm \\ &= -\int Q_\theta(y, m) \log P_\phi(y|m)\, dy\, dy. \end{aligned} \tag{5}$$

Traditionally, one would choose parameters θ and ϕ so as to minimize prediction error:

$$\theta^\star, \phi^\star = \arg\min_{\theta,\phi} C_{Y|M}(Q_\theta \| P_\phi) \tag{6}$$

In SLIP, however, we are interested in minimizing not only the prediction error, but also the *transmission cost*, i.e., the number of bits that need to be sent when the sender transmits the message to the receiver. Let $T(\theta)$ indicate the transmission cost, as function of encoding map parameters θ (note that the transmitted bits will depend only on the encoding map and not the decoding map). We propose to minimize

$$\theta^\star, \phi^\star = \arg\min_{\theta,\phi} C_{Y|M}(Q_\theta \| P_\phi) + \alpha T(\theta) \tag{7}$$

where α is a hyper-parameter that controls the balance between prediction error and transmission cost.

Minimizing both prediction error and transmission cost presents an interesting trade-off. Lowest prediction error can be achieved by making the message m a copy of input x, but this scheme will carry a large transmission cost, since each x will need to be transmitted exactly. On the other hand, sending no bits at all will eliminate transmission cost, but will have high prediction error, since the receiver's predictions cannot depend on m (nor, indirectly, x). The specifics of the trade-off will depend on the distribution $Q(x, y)$; for example, if y is independent of x, then the no-transmission scenario is optimal.

2.3 Transmission cost

There are, in fact, multiple ways of defining the transmission cost, $T(\theta)$, depending on the particular communication scenario. We consider two possibilities.

In the first scenario, the sender receives some input x, uses the encoding map to convert this a given message m, and then losslessly transmits m to receiver. The receiver then uses the decoding map to make a prediction, achieving average prediction accuracy $C_{Y|M}(Q_\theta \| P_\phi)$. Using optimal coding, the transmission of a single m will require on average $H(P_\theta(M))$ bits per input, where H is the Shannon entropy [7] and $P_\theta(m) = \int Q(x) P_\theta(m|x)\, dx$. Thus, in this 'single-message' regime, transmission cost is naturally defined as [1]

$$T_{\text{single}}(\theta) := H(P_\theta(M))$$

In the second scenario, the sender first receives a block of n inputs $\boldsymbol{x}^{(n)} := \langle x_1, \ldots, x_n \rangle$ sampled IID from $Q(x)$, and transmits to the receiver a block of n messages $\boldsymbol{m}^{(n)} := \langle m_1, \ldots, m_n \rangle$. If the sender were to select the transmitted block of messages by sampling from $P_\theta(\boldsymbol{m}^{(n)}|\boldsymbol{x}^{(n)}) := \prod_{i=1}^n P_\theta(m_i|x_i)$, this would require $nH(P_\theta(M))$ bits and again achieve average error $C_{Y|M}(Q_\theta \| P_\phi)$. However, 'block-coding' results in information-theory [7] show that as $n \to \infty$, it is possible to achieve average error $C_{Y|M}(Q_\theta \| P_\phi)$ while only transmitting $nI_{Q_\theta}(X; M)$ bits, where

$$I_{Q_\theta}(X; M) := \int Q_\theta(x, m) \log \frac{P_\theta(m|x)}{\int Q_\theta(x', m)\, dx'}\, dx\, dm$$

is the mutual information between inputs and messages.

Thus, in the 'block-coding' regime, transmission cost is naturally defined as

$$T_{\text{block}}(\theta) := I_{Q_\theta}(X; M)$$

bits per input; a sketch of the relevant information-theoretic argument is found in Appendix B.

In this paper, we use T_{block} for the transmission cost in Eq. (7), leading to the following SLIP cost function:

$$\mathcal{L}_{\text{SLIP}}(\theta, \phi) := C_{Y|M}(Q_\theta \| P_\phi) + \alpha I_{Q_\theta}(X; M) \tag{8}$$

However, it is important to note several points of comparison between T_{single} and T_{block}.

First, block-coding results generally hold asymptotically as block lengths get large. In a situation where inputs arrive over time as a data stream, the sender may have a long delay before a sufficient number of inputs are available to make a block. Therefore, in cases where real-time predictions are necessary and delays are not acceptable, it may be more appropriate to use the first transmission scenario of transmission (leading to transmission cost T_{single}), or possibly other scenarios based on finite-blocklength coding [8–10].

Second, we can write the mutual information as:

$$I_{Q_\theta}(X; M) = H(Q_\theta(M)) - H(P_\theta(M|X))$$

By the non-negativity of conditional entropy, it can be seen that the mutual information is never larger than $H(P_\theta(M))$, i.e. $T_{\text{single}}(\theta) \geq T_{\text{block}}(\theta)$. When encoding map P_θ is deterministic, $H(P_\theta(M|X)) = 0$, and single-message and block-coding transmission costs become equivalent. More generally, if the noise level $H(P_\theta(M|X))$ is held fixed, then single-message and block-coding optimization problems (Eq. (7)) differ by a constant, and are equivalent. If, however, the amount of noise is allowed to vary, then noisy (i.e., non-deterministic) encoding maps can be advantageous for block-coding, since they decrease transmission cost T_{block}, with noise acting as an adaptively-chosen quantization. On the other hand, in the single-message regime, adding noise can only increase transmission cost T_{single}; therefore, the SLIP cost function will be optimized by deterministic encoding maps.

2.4 Recurrent neural networks

It is straightforward to generalize SLIP to the case of sequence models implemented by recurrent neural networks [RNNs]. For simplicity, we consider a basic case in which a RNN is trained to predict future trajectories of a temporal sequence given initial states.

[1] Note that if m is continuous-valued, quantization will also be required, though here we ignore these details.

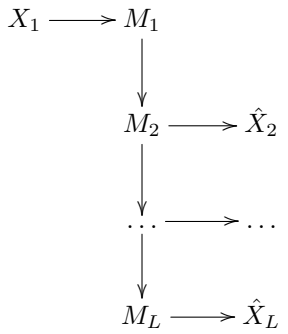

Figure 1: The organization of the RNN. Initial state X_1 is provided as input. This state is mapped to a middle layer M_1 according to $P_\theta(m|x)$. This middle layer updates itself over L-steps. At step t, the network outputs a prediction $\hat{X}^t$ according to $P_\phi(x|m)$.

Let $\vec{x} := \langle x_1, \ldots, x_L \rangle$ indicate a length-L sequence of states and assume a dataset $\mathcal{D} = \{\vec{x}_1, \ldots, \vec{x}_N\}$ of N such sequences sampled from some 'true' distribution $Q(\vec{X} = \vec{x})$. We imagine a network where initial states serve as input X_1, which then feeds into the initial middle layer state M_1 according to map $P_\theta(m|x)$. Middle layer states update themselves in a recurrent manner for M_t for $t = 2..L$ according to map $P_\gamma(m_{t+1}|m_t)$, while generating future predictions $\hat{X}_t$ for $t..2$ according to map $P_\phi(x|m)$. (The flowchart of this RNN is diagrammed in Fig. 1. This topology is sometimes called the "one-to-many" RNN)

Assuming as above that the map from X_1 to M_1 is stochastic, and that predictions at all points from X_2 to X_L contribute to log likelihood, the RNN analogue of the $\mathcal{L}_{\text{SLIP}}$ cost function is:

$$\mathcal{L}_{\text{SLIP}}(\theta, \phi) = \alpha I_{Q_\theta}(X_1; M_1) + \sum_{t=2}^{L} C_{X_t|M_t}(Q_\theta \| P_\phi) \tag{9}$$

where

$$Q_\theta(M_t = m, X_t = x) = Q(X_t = x) P_\theta(m|x) \; .$$

Minimizing this cost function will identify networks that offer good predictions of future trajectories, while minimizing the information transferred from initial input state to initial middle layer state.

3 Implementation

3.1 Overview

We now demonstrate an implementation of SLIP which learns optimal encoding and decoding maps from finite data. We take advantage of the fact that SLIP can be naturally interpreted and implemented using a neural network architecture.

We consider a feed-forward network with three continuous-valued layers: an input layer, a middle layer, and an output layer.

The SLIP 'encoding map' corresponds to the mapping from input layer to middle layer states, $P_\theta(m|x)$. Here, θ are neural network parameters specifying the mapping, typically including biases and weights of connections to the middle layer. Unlike most neural network implementations, in our case the mapping from the input to the middle layer can be stochastic. We define the mapping to be a deterministic differentiable function $f_\theta(x)$, plus Gaussian noise of variance σ^2:

$$P_\theta(m|x) := f_\theta(x) + \mathcal{N}\left(0, \sigma^2 \mathbf{I}\right) \tag{10}$$

Note that σ is a trainable parameter (i.e., one of the elements of θ), and is optimized during the learning process to minimize $\mathcal{L}_{\text{SLIP}}$ (Eq. (8)). This parameter sets the noise levels so as to minimize mutual information between input and middle layer activity, while still preserving prediction accuracy.

The SLIP 'decoding map' corresponds to the mapping from the middle layer states to predictions about outputs, $P_\phi(y|m)$. We first let the mapping from middle layer to output layer states be a

deterministic differentiable function $f_\phi(m)$, with parameter vector ϕ typically specifying biases and weights of connections to the output layer. Then, we define the decoding map as a Gaussian with unit variance centered on $f_\phi(m)$:

$$P_\phi(y|m) := f_\phi(m) + \mathcal{N}(0, \mathbf{I}) \ .$$

3.2 Optimization

The neural network learns optimal encoding and decoding maps from a finite training dataset. We assume that the training dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ represents N input-output pairs sampled IID from $Q(x, y)$. In addition, each time a network is iterated (for example, during each iteration of training, as well when testing the network on novel inputs), we stochastically sample a middle-layer activation m_i for each x_i from $P_\theta(m|x_i)$.

To optimize $\mathcal{L}_{\text{SLIP}}$, we perform gradient descent in parameter space according to:

$$\langle \theta, \phi \rangle_{t+1} \leftarrow \langle \theta, \phi \rangle_t - \gamma \nabla_{\theta,\phi} \left(-\frac{1}{N} \sum_i \log P_\phi(y_i|m_i) + \alpha \bar{I}_{\theta,\mathcal{D}}(X; M) \right) \tag{11}$$

where γ is a learning rate parameter and $\bar{I}_{\theta,\mathcal{D}}(X; M)$ is a finite-sample estimator of mutual information $I_{Q_\theta}(X; M)$.

Accounting for the stochasticity of the mapping from x to m, the *expected* step direction will correspond to the expected gradient:

$$\nabla_{\theta,\phi} \left(-\frac{1}{N} \mathbb{E}_{P_\theta} [\, \log P_\phi(y_i|M_i) \,] + \alpha \bar{I}_{\theta,\mathcal{D}}(X; M) \right)$$

and as the training dataset size $N \to \infty$, the expected gradient steps converge to:

To estimate the mutual information, we first note that $I_{Q_\theta}(X; M) = I(f_\phi(X); M)$. Let d indicate the dimensionality of $f_\theta(x)$. We approximate the distribution of $f_\theta(X)$ as a mixture of Gaussians with variance h, with one Gaussian centered at each $f_\theta(x_i)$ for $i = 1..N$. h is chosen using cross validation to maximize the leave-one-out log likelihood of the data [11]:

$$h = \arg\max_s \sum_i \log \left(\frac{1}{n-1} \sum_{j \neq i} \frac{1}{\sqrt{2\pi} s^{d/2}} \exp\left(-\frac{\|f_\theta(x_i) - f_\theta(x_j)\|_2^2}{2s} \right) \right) \tag{12}$$

Note that we approximate the distribution of $f_\theta(X)$ as a mixture of Gaussians with variance h, and that the channel from $f_\theta(X)$ to middle layer activity M is an Additive White Noise Gaussian channel with noise variance σ^2 (Eq. (10)). This allows us to use a 'variational' upper bound on the mutual information (described in detail in [12]):

$$I(f_\phi(X); M) \leq -\frac{1}{N} \sum_i \log \frac{1}{N} \sum_j \exp\left(-\frac{\|f_\theta(x_i) - f_\theta(x_j)\|_2^2}{2(\sigma^2 + h)} \right) - d \log \frac{\sigma^2}{\sigma^2 + h} \tag{13}$$

We equate $\bar{I}_{\theta,\mathcal{D}}(X; M)$ with the RHS of Eq. (13). Note that in a situation where $f_\theta(x_i)$ are mapped into several well-separated clusters (a commonly-encountered solution to the optimization problem posed here), this bound becomes tight [12].

Thus, performing gradient descent according to Eq. (11) minimizes an upper bound on $\mathcal{L}_{\text{SLIP}}$ of Eq. (8). We carry out the following alternating procedure:

1. Holding θ fixed, we use an optimizer to select the best value of h according to Eq. (12).
2. Holding h, we take several stochastic gradient descent steps according to Eq. (11). Note that step 2 also updates the noise level σ^2. It also relies on the fact that the mutual information estimator in Eq. (13) is differentiable in σ^2 and other parameters in θ.

The optimization is performed using off-the-shelf deep learning frameworks [13, 14].

See [15–19] for related ideas about using neural networks to optimize non-parametric estimates of information-theoretic functions.

4 Relation to existing approaches

We discuss the relation of our method to existing approaches in auto-encoders, regularization, and the information bottleneck.

4.1 Auto-encoders

Auto-encoders can be consider as a special case of the supervised learning, in which the output Y is a copy of the input X. It is natural to penalize the coding length of middle layers as a way of compressing data in a way that allows accurate reconstruction, and in fact this was suggested in early work on auto-encoders [20, 21]. More recent work has looked at *denoising auto-encoders* [22], which – like our approach – find more robust encodings by injecting noise into the map between input and middle layers.

However, there are also several differences between our approach and existing auto-encoder work. First, with the exception of [23], these ideas have generally not been applied to the supervised learning context. Furthermore, existing treatments treat middle layer state space as a discrete-valued, limiting the flexibility of possible middle-layer representations and mappings from input layers to middle layers (SLIP generally allows middle layer states to be either discrete- or continuous-valued; here we implement the latter case). Finally, existing work has looked at either penalizing middle layer encoding length (analogous to our 'transmission cost'), *or* injecting noise into the map, rather than combing the two as we do here. Because of this, denoising autoencoders do not have a notion of "optimal" noise level on the training data (since less noise would improve prediction error on the training data), and cannot directly adapt the noise level.

4.2 Regularization by injecting noise

Adding noise to neural network activity has been shown to be an effective regularization technique in terms of improving generalization performance (e.g., [24]). Unlike these approaches, however, our primary motivation for adding noise is not regularization, but rather to control the amount of information passed from inputs to middle layers. For this reason, unlike typical regularization techniques, we add noise both during training and testing regimes. However, as our results show, in some cases our approach can also improve generalization performance.

In the RNN context, it has been argued that noise should only be added to the map from inputs to middle states, rather than in the recurrent connections (where the noise can amplify itself and drown out all signal) [25]. This is concordant with our proposal to constrain the information transferred from input to middle states, rather than over the recurrent connections (section 2.4).

4.3 Information bottleneck

The *information bottleneck* [IB] [5, 26, 27] is a method for extracting information in one random variable that is relevant for predicting another random variable.

The IB takes a joint distribution over two random variables X and Y with joint distribution $Q(x, y)$ and posits a "bottleneck" variable M. M is defined to be a stochastic function of X, while also being conditionally independent of Y given X. Using our previous notation, IB searches for maps $P_\theta(m|x)$ which minimize mutual information between X and M, while maximizing mutual information between M and Y:

$$\arg\min_\theta -I_{Q_\theta}(M;Y) + \alpha I_{Q_\theta}(X;M) =: \mathcal{L}_{\text{IB}}$$

where we use $\mathcal{L}_{\text{IB}}$ to indicate the information bottleneck cost function, and the first mutual information term is computed using Eq. (1) and the second mutual information using Eq. (2).

There is a fundamental connection between the SLIP cost function $\mathcal{L}_{\text{SLIP}}$ and the information-bottleneck cost function $\mathcal{L}_{\text{IB}}$. We first write the SLIP cost function (Eq. (8)):

$$\begin{aligned}\mathcal{L}_{\text{SLIP}} &= C_{Y|M}(Q_\theta \| P_\phi) + \alpha I_{Q_\theta}(X;M) \\ &= H\left(Q_\theta(Y|M)\right) + D_{Y|M}(Q_\theta \| P_\phi) + \alpha I_{Q_\theta}(X;M)\end{aligned}$$

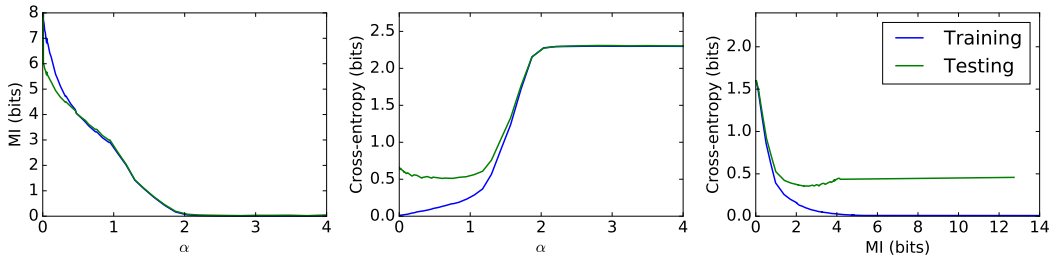

Figure 2: Values of α, prediction error (cross-entropy), and compression (MI) using a feedforward-network on the MNIST dataset. Training set results shown in blue and testing in green.

Note that the minimizing $\mathcal{L}_{\text{SLIP}}$ is equivalent to minimizing $\mathcal{L}_{\text{SLIP}} + \text{const}$. Thus, we add the constant $-H(Q(Y))$ to $\mathcal{L}_{\text{SLIP}}$ to define the equivalent:

$$\begin{aligned}\mathcal{L}'_{\text{SLIP}} &= \mathcal{L}_{\text{SLIP}} - H\left(Q(Y)\right) \\ &= -I_{Q_\theta}(Y;M) + D_{Y|M}(Q_\theta \| P_\phi) + \alpha I_{Q_\theta}(X;M) \\ &= \mathcal{L}_{\text{IB}} + D_{Y|M}(Q_\theta \| P_\phi) \end{aligned} \tag{14}$$

Thus, minimizing $\mathcal{L}_{\text{SLIP}}$ is equivalent to minimizing an upper bound on $\mathcal{L}_{\text{IB}}$, where the middle layer functions as the bottleneck variable. More specifically, $\mathcal{L}'_{\text{SLIP}}$ is composed of two terms: the IB cost function $\mathcal{L}_{\text{IB}}$, and a penalty term that pushes the map $P_\phi\left(y|m\right)$ to correspond to the conditional distribution $Q_\theta\left(y|m\right)$ (as defined in Eq. (1)).

Note that computing the integral in Eq. (1) needed to find the exact decoding map may be intractable, and may encounter estimation issues if Q is fit using a finite training dataset. SLIP addresses this by finding the best encoding *and* decoding maps within a parametric family, in this way providing a tractable upper bound to IB.

Connections between neural networks and the IB have been previously explored [28], but by penalizing the information between input and middle layers, we develop a much more direct link. In addition, the IB has been previously developed only for discrete-valued random variables [5], or for continuous-valued random variables that are related in a linear fashion [6]. Our approach provides a general way to perform non-linear IB on continuous-valued random variables.

5 Results

5.1 Feedforward

As a preliminary demonstration of the feasibility of SLIP, we demonstrate it on a simple feed forward network, with a middle layer containing 10 nodes. This network was trained using 2000 randomly sampled images from the MNIST dataset of 10 digits, with the output being the digit class. The network was trained to minimize $\mathcal{L}_{SLIP}$ and then tested on 2000 other randomly sampled images.

Figure 2 show values of α, prediction error (cross-entropy), and compression (MI) on the MNIST dataset, for both training and testing datasets. By varying α, we can smoothly tradeoff between prediction error and compression. First, when $\alpha = 0$ (no information-theoretic regularization is performed), the MI between input and middle layer is approximately ~15 bits (not shown). However, even for extremely small positive α, MI decreases to ~8 bits with no loss in performance, showing that middle layer activity was previously storing a a large amount of information irrelevant for prediction.

As α increases further, the expected monotonic relationship holds for the training data: MI decreases, while prediction performance worsens. For testing data, however, there is a regularization effect: for small values of α, MI values decrease while prediction error *also decreases*, reaching it's minimum at $\alpha \approx 1$, MI ≈ 3 bits (notice that to store one of 10 equiprobable options requires ~ 3.32 bits). Thus, some amount of information-theoretic regularization improves generalization performance on testing data. Once MI is decreased further, the monotonic relationship that was observed in the training set also arises on the training dataset: prediction error increases as MI decreases.

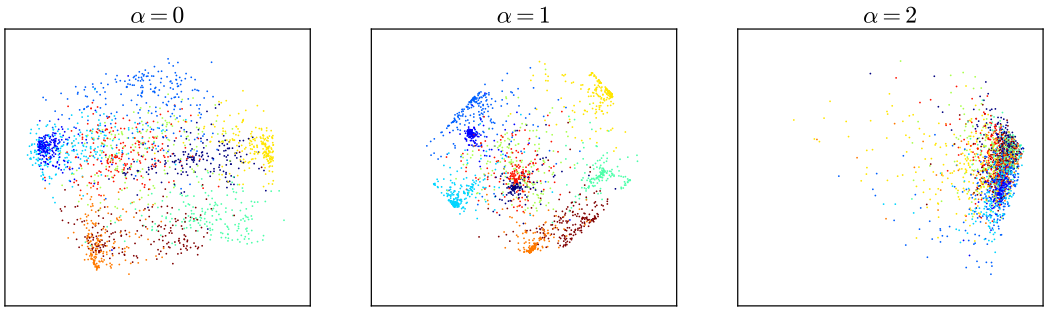

Figure 3: Middle layer activity for three values of α on the MNIST dataset.

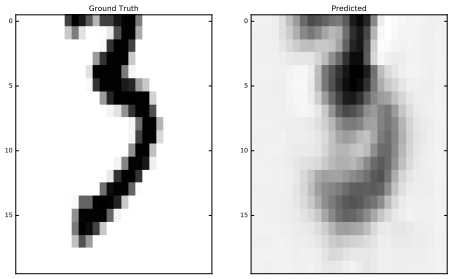

Figure 4: Demonstration of the RNN task. The networks were given the four rows of a MNIST image as input, and trained to predict the rest of the image. An example ground truth sequence is on the left, and the corresponding predicted sequence is on the right.

Some insight into these networks can be derived by plotting the activity of the middle layer for different values of α. Fig. 3 shows activity for α=0, 1, and 2. Each point represents the activation of the middle layer for a given input sample in the training datset. The 10-dimensional points are projected to 2 dimensions using PCA, and points are colored according to their true output (i.e. digit). For $\alpha = 0$, the points corresponding to each digits are located in relatively diffuse clouds. For $\alpha = 1$, points corresponding to different digits are packed in relatively-compact clusters. For $\alpha = 2$, there is little structure in the middle layer activity.

5.2 Recurrent neural networks

We also implemented a simple SLIP for RNNs, as a preliminary demonstration of the feasibility of the approach in this domain.

Our architecture consisted of a one-to-many RNN (Fig. 1) that learned to predict 5-long sequences of 112-dimensional vectors. The recurrent middle layer consisted of 15 nodes.

We again used MNIST data, sampling 2000 images for training dataset and 2000 images for the testing dataset. However, instead of predicting the digits, we predicted image content. Specifically, each image in MNIST consists of 28x28 grayscale pixels. We discarded the first 8 rows, and then grouped the remaining rows into sets of 4, converting each input image into 5x112 matrix. The network was given the first row of this matrix as input, and then asked to predict rows 2,3,4 and 5 using a recurrent architecture. The task is demonstrated diagrammed in Fig. 4.

Figure 5 show values of α, prediction error (cross-entropy), and compression (MI) on this task, for both training and testing datasets. As in the feedforward network, we again observe that by varying α, we can smoothly trade-off between prediction error and information storage. Interestingly, we do not see a regularization effect for small values of α as before; understanding the cause of this requires further investigation.

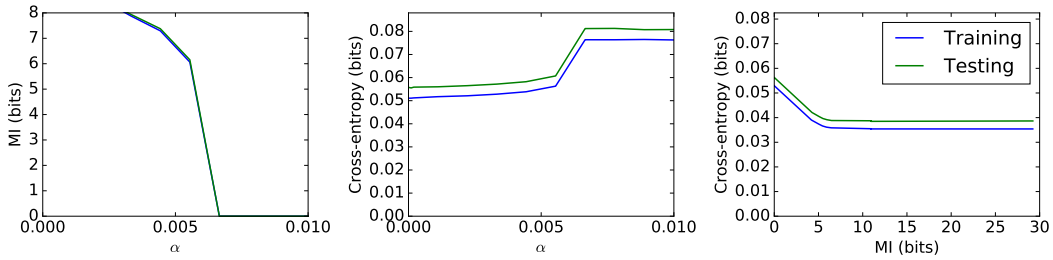

Figure 5: Values of α, prediction error (cross-entropy), and compression (MI) using an RNN. Training set results shown in blue and testing in green.

Acknowledgments

We would like to thank the Santa Fe Institute for helping to support this research. This work was made possible through the support of AFOSR MURI on multi-information sources of multi-physics systems under Award Number FA9550-15-1-0038.

References

[1] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006.

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

[3] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] N. Tishby, F. Pereira, and W. Bialek. The information bottleneck method. In *37th Allerton Conf on Communication*, 1999.

[6] Felix Creutzig, Amir Globerson, and Naftali Tishby. Past-future information bottleneck in dynamical systems. *Physical Review E*, 79(4):041925, 2009. URL `http://journals.aps.org/pre/abstract/10.1103/PhysRevE.79.041925`.

[7] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. John Wiley & Sons, 2006.

[8] Yury Polyanskiy, H Vincent Poor, and Sergio Verdú. Channel coding rate in the finite blocklength regime. *IEEE Transactions on Information Theory*, 56(5):2307–2359, 2010.

[9] Victoria Kostina and Sergio Verdú. Fixed-length lossy compression in the finite blocklength regime. *IEEE Transactions on Information Theory*, 58(6):3309–3338, 2012.

[10] Victoria Kostina and Sergio Verdú. Lossy joint source-channel coding in the finite blocklength regime. *IEEE Transactions on Information Theory*, 59(5):2545–2575, 2013.

[11] Peter Hall and Sally C Morton. On the estimation of entropy. *Annals of the Institute of Statistical Mathematics*, 45(1):69–88, 1993.

[12] Artemy Kolchinsky, Brendan D. Tracey, and David H. Wolpert. An upper bound on the entropy and mutual information of gaussians mixtures. 2017. URL `arxiv:topost`.

[13] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[14] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org*.

[15] Nicol Norbert Schraudolph. *Optimization of entropy with neural networks*. PhD thesis, Citeseer, 1995. URL `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.165.2068&rep=rep1&type=pdf`.

[16] Nicol N. Schraudolph. Gradient-based manipulation of nonparametric entropy estimates. *Neural Networks, IEEE Transactions on*, 15(4):828–837, 2004. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1310356.

[17] Sarit Shwartz, Michael Zibulevsky, and Yoav Y. Schechner. Fast kernel entropy estimation and optimization. *Signal Processing*, 85(5):1045–1058, 2005. URL http://www.sciencedirect.com/science/article/pii/S0165168405000216.

[18] Kari Torkkola. Feature extraction by non-parametric mutual information maximization. *Journal of machine learning research*, 3(Mar):1415–1438, 2003.

[19] Katerina Hlavávcková-Schindler, Milan Palus, Martin Vejmelka, and Joydeep Bhattacharya. Causality detection based on information-theoretic approaches in time series analysis. *Physics Reports*, 441(1):1–46, 2007. URL http://www.sciencedirect.com/science/article/pii/S0370157307000403.

[20] Geoffrey E. Hinton and Richard S. Zemel. Autoencoders, minimum description length, and Helmholtz free energy. *Advances in neural information processing systems*, pages 3–3, 1994. URL http://www.cs.toronto.edu/~hinton/absps/cvq.pdf.

[21] Geoffrey E. Hinton and Richard S. Zemel. Minimizing description length in an unsupervised neural network. *Preprint*, 1997. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.111.1574&rep=rep1&type=pdf.

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

[23] G. Deco, W. Finnoff, and H. G. Zimmermann. Elimination of Overtraining by a Mutual Information Network. In Stan Gielen and Bert Kappen, editors, *ICANN '93*, pages 744–749. Springer London, 1993. ISBN 978-3-540-19839-0 978-1-4471-2063-6. URL http://link.springer.com/chapter/10.1007/978-1-4471-2063-6_208. DOI: 10.1007/978-1-4471-2063-6_208.

[24] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[25] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[26] Alexander G. Dimitrov and John P. Miller. Neural coding and decoding: communication channels and quantization. *Network: Computation in Neural Systems*, 12(4):441–472, 2001. URL http://www.tandfonline.com/doi/abs/10.1080/net.12.4.441.472.

[27] Inés Samengo. Information loss in an optimal maximum likelihood decoding. *Neural computation*, 14(4):771–779, 2002. URL http://www.mitpressjournals.org/doi/abs/10.1162/089976602317318947.

[28] Naftali Tishby and Noga Zaslavsky. Deep Learning and the Information Bottleneck Principle. *arXiv preprint arXiv:1503.02406*, 2015. URL http://arxiv.org/abs/1503.02406.

Appendix A Comparison of SLIP to other approaches

In the introduction, we argued that:

(A) SLIP is formally equivalent to a scenario in which a sender first uses an input to make a probabilistic prediction about the output, and then sends a compressed description of the prediction to the receiver.

(B) SLIP is better than a scheme in which the sender compresses the input using a some general-purpose compression algorithm (one that is not tailored toward the prediction task), sends the compressed input to the receiver, and the receiver then uncompresses the input and makes a prediction.

Here we provide formal justification for these claims.

First consider scenario (A). Assume the sender maps each input x to a probabilistic prediction of the output, indicated by $P_x(Y = y)$. The sender convert P_x to some transmittable description s (e.g., s could be a binary string) and transmits s to the receiver. The receiver gets s and decodes from it some probabilistic prediction about outputs Y, i.e., a probability distribution over Y. Formally, the sender's actions can be represented by a map from inputs to description-of-predictions d according to some $P(s|x)$, and the receiver's action can be represented by some $P(y|s)$ that uncovers a set of probabilistic prediction about Y given s. In this sense, scenario A is formally equivalent to SLIP, with s, the description-of-predictions, acting as the 'message' m as formulated in SLIP.

We now compare scenario (B) to SLIP. Assume that the encoding maps for scenario (B) and for SLIP is chosen from the same space of possible encoding maps Θ, and similarly that the decoding maps are chosen from the same space of possible decoding maps Φ. The difference is that in SLIP the encoding map is chosen in order to minimize the overall objective of minimizing transmission cost while lowering prediction error, while in scenario (B) the encoding map is chosen to minimize transmission cost while lowering some arbitrary distortion function. In both cases, the decoding maps are chosen to minimize prediction error. We show that, for a given number of transmitted bits, SLIP will generally achieve lower prediction error.

To begin, let $\theta^\star, \phi^\star$ be a solution to the SLIP optimization problem:

$$\theta^\star, \phi^\star = \underset{\theta\in\Theta,\phi\in\Phi}{\arg\min}\, C_{Y|M}(Q_\theta \| P_\phi) + \alpha I_{Q_\theta}(X; M)$$

and let $E(\theta)$ indicate the best prediction error achievable given fixed encoding map P_θ:

$$E(\theta) := \min_{\phi\in\Phi} C_{Y|M}(Q_{\theta'} \| P_\phi)$$

Note that the incurred prediction error, $E_{\text{SLIP}} := C_{Y|M}(Q_{\theta^\star} \| P_{\phi^\star})$, is a solution to the minimization problem:

$$\begin{aligned} E_{\text{SLIP}} = \min_{\theta\in\Theta} & E(\theta) \\ \text{subject to } & I_{Q_\theta}(X; M) \leq R \end{aligned} \tag{15}$$

where $R := I_{Q_{\theta^\star}}(X; M)$.

Now assume that scenario (B) maps input x to compressed description m according to some $P_{\theta'}(m|x)$, which is chosen using rate-distortion minimization:

$$\begin{aligned} \theta' = \underset{\theta\in\Theta}{\arg\min}\ & \mathbb{E}_{Q_\theta}[d(x, m)] \\ \text{subject to } & I_{Q_\theta}(X; M) \leq R \end{aligned} \tag{16}$$

where d is some arbitrary distortion function and R is defined as above. The prediction error for scenario (B) is given by the best decoding map ϕ', given that the encoding map is specified by θ': $E_{\text{COMP}} = E(\theta')$.

Note that generally

$$E_{\text{COMP}}(\theta') \geq E_{\text{SLIP}} \tag{17}$$

since the former is achieved by optimizing only the decoding map ϕ, while the latter is achieved by optimizing both the encoding map θ and decoding map ϕ (note that the rate constraint which is explicit in Eq. (15) is also satisfied by θ', by definition in Eq. (16)).

We now show that in many situations, the inequality in Eq. (17) will be strict. Assume for simplicity that Θ is the the space of all possible *deterministic* maps from x to m, and let $f_\theta(x)$ indicate the message m corresponding to input x when the deterministic map indexed by θ is used. Note that the assumption of deterministic maps means that the mutual information is equal to the entropy:

$$I_{Q_{\theta'}}(X; M) = H(Q_{\theta'}(M))$$

where $Q_{\theta'}(M)$ is defined according to Eq. (3).

Let θ' be chosen according to Eq. (16), and m_1, m_2 be any two messages such that the marginal probabilities of the two messages obey

$$Q_{\theta'}(m_1) \leq Q_{\theta'}(m_2)\ . \tag{18}$$

Let $\hat{x}$ be some input such that $f_{\theta'}(\hat{x}) = m_1$, and let $\theta'_{\hat{x}\to m_2}$ indicate a slight modification of the map θ', where input $\hat{x}$ is mapped to message m_2 rather than m_1:

$$f_{\theta'_{\hat{x}\to m_2}}(x) = \begin{cases} f_{\theta'}(x) & \text{if } x \neq \hat{x} \\ m_2 & \text{otherwise} \end{cases}$$

It is easy to show that, due to Eq. (18), $H(Q_{\theta'_{\hat{x}\to m_2}}(M)) < H(Q_{\theta'}(M))$. Since we also assumed that θ' was chosen to minimize distortion, this means that $d(\hat{x}, m_1) \leq d(\hat{x}, m_2)$ – otherwise $\theta'_{\hat{x}\to m_2}$ would be chosen as the solution to Eq. (16), given that it would obey that rate constraint and also have lower distortion.

It is quite possible, however, that $E(\theta') > E(\theta'_{\hat{x}\to m_2})$ – in other words, a lower prediction error is possible by switching $\hat{x}$ from m_1 to m_2. Then,

$$E(\theta') > E(\theta'_{\hat{x}\to m_2}) \geq E_{\text{SLIP}}$$

making the inequality Eq. (17) strict.

To summarize, given the assumptions stated above, a sufficient condition for the inequality to be strict is for there to exist an $\hat{x}$ and an m_2 such that:

$$\begin{aligned} Q_{\theta'}(f_{\theta'}(\hat{x})) &\leq Q_{\theta'}(m_2) \\ &\text{and} \\ d(\hat{x}, f_{\theta'}(\hat{x})) &\leq d(\hat{x}, m_2) \\ E(\theta') &> E(\theta'_{\hat{x}\to m_2}) \end{aligned}$$

As the number of inputs and messages grows, the existence of such a pair $\hat{x}$ and an m_2 becomes increasingly likely.

Appendix B Mutual information as transmission cost

In the main text, we consider the situation in which the sender receives an entire block of inputs $\boldsymbol{x}^{(n)} := \langle x_1, \ldots, x_n \rangle$ and transmits to the receiver an entire block of messages $\boldsymbol{m}^{(n)} := \langle m_1, \ldots, m_n \rangle$. We claim that for any θ, as $n \to \infty$ it is sufficient for the sender to transmit to the receiver $I_{Q_\theta}(X; M)$ bits *per input*, and still achieve average error $C_{Y|M}(Q_\theta \| P_\phi)$ *per input*.

The proof is based on standard rate-distortion arguments, specifically the achievability of the rate-distortion function. The central idea is the following: if the sender were to sample a block of messages from $P_\theta(\boldsymbol{m}^{(n)}|\boldsymbol{x}^{(n)}) := \prod_{i=1}^n P_\theta(m_i|x_i)$ and transmit this block exactly, transmission would require $nH(P_\theta(M))$ bits and achieve some average error $C_{Y|M}(Q_\theta \| P_\phi)$. Instead, for each $\boldsymbol{x}^{(n)}$, the sender can deterministically select one particular block of messages, which is defined to have the same statistics as most other samples from $P_\theta(\boldsymbol{m}^{(n)}|\boldsymbol{x}^{(n)})$. Sending this particular block takes on average $nI_{Q_\theta}(X; M)$ bits, but still allows the receiver to achieve the error $C_{Y|M}(Q_\theta \| P_\phi)$ per input.

Slightly more formally, we sketch a proof based on [7, section 10.5]. Define a deterministic function $f : X^n \to M^n$ which maps each $\boldsymbol{x}^{(n)}$ into a block of messages $f(\boldsymbol{x}^{(n)})$. Using a random code, f is defined so that each $f(\boldsymbol{x}^{(n)})$ is jointly *distortion typical* with $\boldsymbol{x}^{(n)}$. This means that the pair of blocks $\langle \boldsymbol{x}^{(n)}, f(\boldsymbol{x}^{(n)}) \rangle$ is approximately the same as most samples from $P_\theta(\boldsymbol{m}^{(n)}|\boldsymbol{x}^{(n)})$ in terms of the frequencies of the occurrence of each pair of outcomes (x, m), and in terms of the average prediction error $C(Q(Y|x) \| P_\theta(Y|m))$ (Eq. (4)) per co-occurring (x, m) combination.

Now assume that the cardinality of the image of f is $2^{n(I_{Q_\theta}(X;M)+\epsilon)}$. As $n \to$, it can be shown that the expected prediction error across the block

$$\int Q(\boldsymbol{x}^n) \frac{1}{n} \sum_{i=1}^n C(Q(Y|X = [\boldsymbol{x}^n]_i) \| P_\theta(Y|M = [f(\boldsymbol{x}^{(n)})]_i))\, d\boldsymbol{x}^n \leq C_{Y|M}(Q_\theta \| P_\phi) + \epsilon$$

where $[\cdot]_i$ indicates the i^{th} element in a block. At the same time, the transmission cost reflects the expected number of bits needed to transmit some block of messages $f(\boldsymbol{x}^{(n)})$:

$$H(f(\boldsymbol{X}^{(n)})) \leq n(I_{Q_\theta}(X; M) + \epsilon)$$

where ϵ can be made as small as desired.

The above assumed that M is a discrete-valued random variable. If M is continuous-valued, the same results can be recovered by quantizing M to a resolution Δ, then taking the limit of $\Delta \to \infty$.