

High-Dimensional Probability Estimation with Deep Density Models

Oren Rippel*
 Massachusetts Institute of Technology,
 Harvard University
 rippel@math.mit.edu

Ryan Prescott Adams†
 Harvard University
 rpa@seas.harvard.edu

Abstract

One of the fundamental problems in machine learning is the estimation of a probability distribution from data. Many techniques have been proposed to study the structure of data, most often building around the assumption that observations lie on a lower-dimensional manifold of high probability. It has been more difficult, however, to exploit this insight to build explicit, tractable density models for high-dimensional data. In this paper, we introduce the *deep density model* (DDM), a new approach to density estimation. We exploit insights from deep learning to construct a bijective map to a representation space, under which the transformation of the distribution of the data is approximately factorized and has identical and known marginal densities. The simplicity of the latent distribution under the model allows us to feasibly explore it, and the invertibility of the map to characterize contraction of measure across it. This enables us to compute normalized densities for out-of-sample data. This combination of tractability and flexibility allows us to tackle a variety of probabilistic tasks on high-dimensional datasets, including: rapid computation of normalized densities at test-time without evaluating a partition function; generation of samples without MCMC; and characterization of the joint entropy of the data.

1 Introduction

Many core machine learning tasks are concerned with density estimation and manifold discovery. Probabilistic graphical models are a dominating approach for constructing sophisticated density estimates, but they often present computational difficulties in practice. For example, undirected models, such as the Boltzmann machine [Smolensky, 1986, Hinton et al., 2006] are able to achieve compact and efficiently-computed latent variable representations at the cost of only providing unnormalized density estimates. Directed belief networks [Pearl, 1988, Neal, 1992, Adams et al., 2010], on the other hand, enable one to specify *a priori* marginals of hidden variables and are easily normalized, but require costly inference procedures. Bayesian nonparametric density estimation (e.g., [Escobar and West, 1995, Rasmussen, 2000, Adams et al., 2009]) is another flexible approach, but it often requires costly inference procedures and does not typically scale well to high-dimensional data.

Manifold learning provides an alternative way to implicitly characterize the density of data via a low-dimensional embedding, e.g., locally-linear embedding [Roweis and Saul, 2000], IsoMap [Tenenbaum et al., 2000], the Gaussian process latent variable model [Lawrence, 2005], kernel PCA [Schölkopf et al., 1998], and t-SNE [Van der Maaten and Hinton, 2008]. Typically, however, these methods have emphasized visualization as the primary motivation. A notable exception is the autoencoder neural network [Cottrell et al., 1987, Hinton and Salakhutdinov, 2006], which seeks embeddings in representation spaces that themselves can be high dimensional. Unfortunately, the autoencoder does not have a clear probabilistic interpretation (although see [Rifai et al., 2012] for a discussion).

*<http://math.mit.edu/~rippel>

†<http://people.seas.harvard.edu/~rpa>

Some approaches, such as manifold Parzen windows [Vincent and Bengio, 2002], have attempted to tackle the combined problem of density estimation and manifold learning directly, but have faced difficulties due to the curse of dimensionality. Other approaches, such as the Bayesian GP-LVM [Titsias and Lawrence, 2010], characterize the manifold implicitly in terms of a nonlinear mapping from a representation space to the observed space. To define a coherent probabilistic model, however, it is necessary to find an invertible map between these spaces so that the density of a datum can be evaluated in the latent space without integrating over the pre-image. It has proven difficult to flexibly parameterize the space of such invertible maps, however, let alone find a transformation that results in a tractable density on the representation space. Independent components analysis [Bell and Sejnowski, 1995] and the related idea of a density network [MacKay and Gibbs, 1997] are examples of bijective models that exploit invertible linear transformations; these, however, have rather limited expressiveness. DiffeoMap [Walder and Schölkopf, 2008] establishes a bijection close to a lower-dimensional subspace, and then projects to it. Other approaches, such as the the back-constrained GP-LVM [Lawrence and Candela, 2006] attempt to approximate this bijection.

In this work, we introduce the *deep density model* (DDM), an approach that bridges manifold discovery and density estimation. We exploit ideas from deep learning to introduce a rich and flexible class of bijective transformations of the observed space. We optimize over these transformations to obtain a map under which the implied distribution on the representation space has an approximately factorized form with known marginals. The invertibility of the map ensures that measure is not collapsed across the transformation, and as such, the determinant of the Jacobian can be computed. This leads to fully-normalized probability densities without a partition function.

The combination of rich bijective transformations with density estimation enables us to explore a variety of modeling directions for high-dimensional data. As the approach is generative, we can easily sample data from a trained model without Markov chain Monte Carlo (MCMC). We present a variety of applications to the CIFAR and MNIST datasets, for proof-of-concept. The deep density model also provides new possibilities for supervised learning by building Bayesian classifiers that have well-calibrated class-conditional probabilities. This additionally permits to exploit densities of unlabeled data to perform unsupervised learning, by constructing mixtures of models and training them coherently with expectation maximization.

In developing the deep density model, we also provide insight into a variety of fundamental concepts for latent variable models. Using information theoretic tools, we identify important connections between sparsity and the independence of the latent dimensions. These connections allow finding a map leads to an approximately factorized latent distribution. By understanding the distribution of the data in representation space and the transformation that gives rise to it, we can characterize the entropy of the distribution over data in the observed space. This enables making informed choices in model selection.

2 Bijections and Normalized Densities

We are interested in learning a distribution over data in a high-dimensional space $\mathcal{Y} \subseteq \mathbb{R}^K$. We denote this (unknown) distribution as $p_{\mathbf{Y}}(\cdot)$. An axiomatic assumption in machine learning is that the data contain structure, and this corresponds to $p_{\mathbf{Y}}(\cdot)$ distribution having most of its probability mass on a lower-dimensional, but very complicated, manifold in \mathcal{Y} . Tractably parametrizing the space of such manifolds and then fitting the resulting distributions to data is a significant challenge.

Similarly, studying this distribution directly in the observed space presents both theoretical and computational difficulties. Instead, we consider how the data might be the result of a transformation from an unobserved *representation space* $\mathcal{X} \subseteq \mathbb{R}^K$. We denote the distribution on this space as $p_{\mathbf{X}}(\cdot)$, and we assume the observed data arise from a transformation $\mathbf{f} : \mathcal{X} \rightarrow \mathcal{Y}$. We assume that the latent distribution $p_{\mathbf{X}}(\cdot)$ has a simple factorized form:

$$p_{\mathbf{X}}(\mathbf{x}) = \prod_{k=1}^K p_{X_k}(x_k), \quad (1)$$

where the marginal factors $p_{X_k}(\cdot)$ have a simple and known univariate form.

We further make the assumption that $\mathbf{f}(\cdot)$ is bijective and that $\mathbf{f}^{-1}(\cdot)$ is available analytically. In this case, the probability density for a point $\mathbf{y} \in \mathcal{Y}$ can be computed:

$$p_{\mathbf{Y}}(\mathbf{y}) = \prod_{k=1}^K p_{X_k}([\mathbf{f}^{-1}(\mathbf{y})]_k) \left| \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{f}^{-1}(\mathbf{y})}. \quad (2)$$

In this paper, we introduce the *deep density model*, which discovers rich bijective transformations to map from simple latent distributions into complex observed densities. By optimizing over a large and flexible class of such bijective transformations, it is possible to discover structure in high-dimensional data sets while still having a manageable, normalized density estimator.

Bijectivity is critical for ensuring that the density in Eq. (2) is normalized. This bijectivity is in contrast to many neural network approaches to latent representation where the latent space is often smaller (for an information bottleneck in, e.g., an autoencoder) or larger (for an *overcomplete* representation) than the observed space. When the representation space is smaller, then $\mathbf{f}(\cdot)$ cannot be surjective and so multiple points in \mathcal{Y} may map to the same point in \mathcal{X} , leading to overestimates of the density. In the overcomplete case, we also sacrifice bijectivity since $\mathbf{f}^{-1}(\cdot)$ cannot be surjective; in other words, the image of \mathcal{Y} under $\mathbf{f}^{-1}(\cdot)$ will not span \mathcal{X} . As such, $p_{\mathbf{X}}(\cdot)$ will have support beyond $\mathbf{f}^{-1}(\mathcal{Y})$. That is, the latent normalization includes mass that appears in $\mathcal{X} \setminus \{\mathbf{f}^{-1}(\mathcal{Y})\}$. However, taking this mass into account is a very challenging problem, whose difficulty unfortunately increases with the richness of $\mathbf{f}(\cdot)$. Thus, by using $p_{\mathbf{X}}(\cdot)$ instead of the normalized density

$$\frac{p_{\mathbf{X}}(\mathbf{x}) \mathbb{I}_{\mathbf{x} \in \mathbf{f}^{-1}(\mathcal{Y})}}{\int_{\mathbf{x} \in \mathbf{f}^{-1}(\mathcal{Y})} p_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}} < p_{\mathbf{X}}(\mathbf{x}), \quad (3)$$

we underestimate the true density and are in a situation similar to that in energy-based models with partition functions.

Thus, when learning a representation of data, we must make a choice between overcompleteness and bijectivity. Our model will be able to use bijectivity to produce normalized density estimates, but at the cost of requiring the representation space to be of equal dimension to the observed space. Overcomplete representation spaces are thought to allow for greater parametric flexibility, but we view the ability to produce a normalized density estimate as a significant win.

3 The Deep Density Model

Our objective is to learn a density estimate on the space \mathcal{Y} from a set of N training examples $\mathbf{Y} := \{\mathbf{y}_n\}_{n=1}^N \subset \mathcal{Y}$. We seek to produce an invertible parametric map $\mathbf{f}_{\Theta} : \mathcal{X} \rightarrow \mathcal{Y}$ such that the observed data under $\mathbf{f}_{\Theta}^{-1} : \mathcal{Y} \rightarrow \mathcal{X}$ are well-described by a simple, factorized distribution. We denote this function as the *decoder*, and the representation-space data as $\mathbf{X} := \{\mathbf{x}_n\}_{n=1}^N \subset \mathcal{X}$. We construct the map \mathbf{f}_{Θ} via a sequence of layers, each of which applies an invertible linear transformation followed by an elementwise nonlinearity. This nonlinearity must be bijective; we use the standard sigmoid (logistic) function $\sigma(z) = 1/(1 + e^{-z})$. The parameters for layer m are square matrix $\Omega_m \in \mathbb{R}^{K \times K}$ and bias vector $\omega_m \in \mathbb{R}^K$. Layer m performs transformation $\mathbf{S}_{\Omega_m, \omega_m}(\mathbf{y}) := \sigma(\Omega_m \mathbf{y} + \omega_m)$. Given M layers, we write the entire map as the composition

$$\mathbf{f}_{\Theta} := \bigcirc_{m=1}^M \mathbf{S}_{\Omega_m, \omega_m}(\mathbf{x}), \quad (4)$$

where \bigcirc is the composition operator and $\Theta := \bigcup_{m=1}^M \{\Omega_m, \omega_m\}$.

We also need to construct a non-bijective *encoder* function $\mathbf{g}_{\Psi} : \mathcal{Y} \rightarrow \mathcal{X}$, which maps the observed data into the representation space. This function will be optimized to imbue the latent distribution $p_{\mathbf{X}}(\cdot)$ with the properties outlined in Section 2; the necessity of $\mathbf{g}_{\Psi}(\cdot)$ will be expanded upon in Subsection 3.4.2. $\mathbf{g}_{\Psi}(\cdot)$ is composed of J layers, and the j -th layer of $\mathbf{g}_{\Psi}(\cdot)$ has K_j hidden units (with $K_J = K$) and parameters Γ_j

and γ_j . We denote the parameters for \mathbf{g} as $\Psi := \bigcup_{j=1}^J \{\Gamma_j, \gamma_j\}$ and use the notational shortcuts above to write

$$\mathbf{g}_\Psi(\mathbf{y}) = \bigcirc_{j=1}^J \mathbf{S}_{\Gamma_j, \gamma_j}(\mathbf{y}). \quad (5)$$

3.1 Regularizing the Transformations

We will train the model on data by minimizing an objective composed of several parts:

Divergence Penalty $\mathcal{D}(\Psi)$: This determines the fit of the current encoding transformation. It forces the marginal densities of the empirical distribution of the representation-space data to match a target distribution of our choice, by penalizing divergence from it.

Invertibility Measure $\mathcal{I}(\Theta)$: This ensures the invertibility of $\mathbf{f}_\Theta(\cdot)$ by penalizing poorly-conditioned transformations.

Reconstruction Loss $\mathcal{R}(\Theta, \Psi)$: This jointly penalizes the encoder $\mathbf{g}_\Psi(\cdot)$ and decoder $\mathbf{f}_\Theta(\cdot)$ to ensure that $\mathbf{g}_\Psi(\mathbf{y}) \approx \mathbf{f}_\Theta^{-1}(\mathbf{y})$ on the data.

Each of these participates in the overall objective given by:

$$C(\Theta, \Psi) = \mu_{\mathcal{D}}\mathcal{D}(\Theta) + \mu_{\mathcal{I}}\mathcal{I}(\Psi) + \mu_{\mathcal{R}}\mathcal{R}(\Theta, \Psi), \quad (6)$$

where $\mu_{\mathcal{I}}, \mu_{\mathcal{D}}, \mu_{\mathcal{R}} \in \mathbb{R}$ are the weights of each term. We will examine each of these terms in more detail in the proceeding sections.

3.2 Divergence Penalty: Sculpting the Latent Marginals

In order to fit the deep density model to data, it is necessary to specify a measure of distance between the empirical distribution and the model distribution. We achieve this via a divergence penalty, which forces the model to distribute the mass of the data in the representation space so as to be similar to a distribution chosen *a priori*. This construction has several advantages: it 1) results in a known, fixed distribution on the representation space that can be used to generate fantasy data, 2) enables sparsity to be enforced as a constraint rather than a penalty weighed against the reconstruction cost, and 3) combats overfitting by explicitly requiring that some of the data have low probability under the model. This third advantage is subtle, but critical: some data must live in the tail of the distribution, in contrast to the maximum of the posterior which is simply a weighing of the MLE against the mode of the prior. See Figure 1 for a comparison of distributions produced by a various regularization techniques.

Concretely, we assume that the representation space is a unit hypercube, i.e., $\mathcal{X} = [0, 1]^K$. As before, we assume there are N data $\{\mathbf{y}_n\}_{n=1}^N$, which (for a given Ψ) are mapped into \mathcal{X} to give $\{\mathbf{x}_n = \mathbf{g}_\Psi(\mathbf{y}_n)\}_{n=1}^N$. Ideally, for representation dimension $k \in 1, \dots, K$, the divergence penalty would measure the difference between the marginal empirical distribution

$$\hat{p}_{X_k}(x) = \frac{1}{N} \sum_{n=1}^N \delta(x - [\mathbf{x}_n]_k) \quad (7)$$

and a target univariate distribution $q(\cdot)$ that we define. In practice, we approximate $\hat{p}_{X_k}(\cdot)$ by finding the best fit of a tractable parametric family and then computing the symmetrized Kullback–Liebler divergence:

$$T(p(\cdot) \parallel q(\cdot)) = D(p(\cdot) \parallel q(\cdot)) + D(q(\cdot) \parallel p(\cdot)), \quad (8)$$

where

$$D(p(\cdot) \parallel q(\cdot)) = \int_{\mathbf{x}} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x}. \quad (9)$$

Since, in this case the representation space is the unit hypercube, we choose our objective distribution to be a member of the Beta family:

$$q(x; \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1} (1-x)^{\beta-1}. \quad (10)$$

Given the data representations $\{\mathbf{x}_n\}_{n=1}^N$, we estimate the empirical distribution of each dimension with a Beta distribution, using moment-matching to approximate its parameters:

$$\hat{\alpha}_k = \hat{\mu}_k \left[\frac{\hat{\mu}_k (1 - \hat{\mu}_k)}{\hat{\sigma}_k^2} - 1 \right] \quad (11)$$

$$\hat{\beta}_k = (1 - \hat{\mu}_k) \left[\frac{\hat{\mu}_k (1 - \hat{\mu}_k)}{\hat{\sigma}_k^2} - 1 \right], \quad (12)$$

where $\hat{\mu}_k$ and $\hat{\sigma}_k^2$ are the sample mean and variance, respectively. We note that a Beta distribution with parameter $\alpha < 1$ produces a very sharp peak at 0, and as such allows to pursue sparsity in distribution (under the assumption that elements of small magnitude cannot be distinguished from each other).

With these in hand, we get a closed-form expression for our divergence penalty:

$$T(\hat{p}_{X_k}(\cdot) \| q(\cdot)) = (\hat{\alpha} - \alpha) [\psi(\hat{\alpha}) - \psi(\alpha)] + (\hat{\beta} - \beta) [\psi(\hat{\beta}) - \psi(\beta)] \quad (13)$$

$$- (\hat{\alpha} - \alpha + \hat{\beta} - \beta) [\psi(\hat{\alpha} + \hat{\beta}) - \psi(\alpha + \beta)], \quad (14)$$

where $\psi(z) = d \log \Gamma(z) / dz$ is the digamma function.

Furthermore, for each example, we impose an example divergence penalty, denoted as $\hat{p}_{X_n}(\cdot)$, which penalizes the distance between our objective distribution and the empirical distribution over the elements of that particular example:

Finally, our total divergence penalty is

$$\mathcal{D}(\Psi) = \frac{1}{K} \sum_{k=1}^K T(\hat{p}_{X_k}(\cdot) \| q(\cdot)) + \frac{1}{N} \sum_{n=1}^N T(\hat{p}_{X_n}(\cdot) \| q(\cdot)).$$

3.2.1 Sparsity in Distribution

The traditional pursuit of sparsity entails the application of an L_1 -type regularization that directly penalizes the activations in the representation space. This has several undesirable properties. First, the penalty does not differentiate between the cases where the activated units are distributed evenly among examples, and where a fixed set of units is always activated at all examples while others never are. Furthermore, it is discomforting that, in the limit of small reconstruction cost in the objective function, the regularization term is optimized if and only if all examples are identically mapped to the same point, namely zero. This forces *all* the activations to be small in order to have some of them vanish; it artificially forces the activation distribution to be contained in a small region around zero. Another implication of direct activity penalization is that we must search the parameter space of the regularization coefficient in order to attain our desired sparsity structure.

Instead of inducing sparsity directly, we achieve it in *distribution*, across examples. In practice, this arises from penalizing the KL-divergence between the *empirical distribution* — the actual distribution in the representation space for the given set of observations — and an appropriately-chosen target distribution $q(\cdot)$, which has a peak at 0. The difference between this and the traditional L_1 approach to sparsity can be understood by considering the optimization problem as the dualization of the sparsity constraints. In the case of an L_1 -type regularization, these constraints directly bound the space under the prescribed distance metric. In the L_1 case, we thus have a situation with two different points on the same contour of these constraints, one of which a more desirable of a solution than the other. On the other hand, the constraints that emerge from the divergence penalization are imposed within the probability simplex. The advantage is that a contour of these constraints corresponds to a locus of distributions with similar sparsity structures and thus similar desirability.

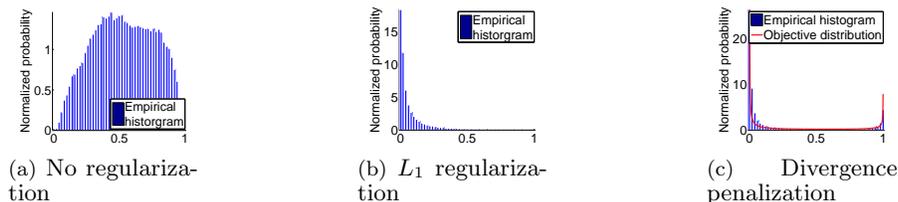


Figure 1: Histograms of the empirical distribution at $\hat{p}_{X_1}(\cdot)$, $k = 1$ with the specified methods of regularization, upon training on MNIST. The objective distribution was taken as $\text{Beta}(\cdot; 0.02, 0.2)$.

3.3 Invertibility: condition number penalty

Invertibility of \mathbf{f}_{Θ} is critical to providing a normalized density. A standard autoencoder contracts volumes around observed examples only, due to the reconstruction penalty approximating an invertible map at the observations. A true bijection, however, will guarantee conservation of volume not just at the data, but also at points we have never seen before. This will allow computation of the determinant of the Jacobian of $\mathbf{f}_{\Theta}(\cdot)$, and precisely specify how probability mass is reshuffled by the transformation. To ensure invertibility of the transformation, we must ensure the invertibility of each layer. As the nonlinear activation functions are fixed, invertibility is determined by the condition numbers of the $\mathbf{\Omega}_m$ matrices in $\mathbf{f}_{\Theta}(\cdot)$. We therefore introduce a regularization term that ensures invertibility:

$$\mathcal{I}(\Theta) = \frac{1}{M} \sum_{m=1}^M \log \left(\frac{\lambda_{\max}(\mathbf{\Omega}_m)}{\lambda_{\min}(\mathbf{\Omega}_m)} \right). \quad (15)$$

Here, $\lambda_{\max}(\mathbf{A})$ and $\lambda_{\min}(\mathbf{A})$ are the maximum and minimum eigenvalues of \mathbf{A} . In this case, the curse of dimensionality becomes a blessing of dimensionality: in high dimensions, not only orthogonality is easily attained, it is difficult to escape. We find that in practice the invertibility requirement is easily satisfied, and does not constrain the algorithm at all.

3.4 Reconstruction and the independence of latent dimensions

Since we now have the ability to dictate the marginal distributions in the representation space, we can shed light on the connection between entropy, sparsity and independence as a function of the transformation to the representation space. In order to attain a tractable distribution in the representation space, we must eliminate dependencies between the latent dimensions. We refer to this process as *diversification*, as it reduces the overlap of information learnt by distinct dimensions; see the effects of this process in Figure 2.

Diversification arises naturally when considering the effect of simultaneously increasing sparsity in the latent representation, while decreasing the entropy of the target marginals. The reconstruction penalty demands that information be preserved in the representation space. However, as the entropy decreases, the information capacity of the marginals decreases; it is necessary for the dimensions of the latent distribution to become more independent in order to reduce redundancy and continue to reconstruct successfully. Hence, we increase sparsity by limiting the capacity of the encoder, until we get to the point of minimum marginal entropy under sufficient conservation of information (which we measure by reconstruction of the observations). At this point, we expect the marginals to be approximately independent.

In the second line below, notice that we may write the observed space entropy in terms of the representation entropy and a term that accounts for the contraction of volumes under the transformation. We then write the joint entropy in terms of the marginal entropies and the mutual information information

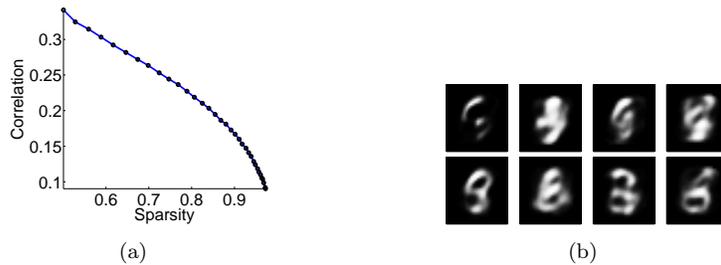


Figure 2: Effects of diversification. (a) Decrease of first-order dependencies as function of sparsity. (b) Generations from the model by sampling independently from the latent marginals. First row: generations from a model with high-entropy marginals $\text{Beta}(\cdot; 0.02, 0.2)$ with strong dependence; images are incoherent. Second row: generations from a model with diversified marginals with a final distribution $\text{Beta}(\cdot; 0.004, 0.2)$.

between the joint distribution and the independent marginal factorization:

$$\begin{aligned}
 \mathcal{H}(p_{\mathbf{Y}}(\cdot)) &= - \int_{\mathbf{y}} p_{\mathbf{Y}}(\mathbf{y}) \log p_{\mathbf{Y}}(\mathbf{y}) d\mathbf{y} \\
 &= \mathcal{H}(p_{\mathbf{X}}(\cdot)) + \mathbb{E}_{\mathbf{x}} \left[\log \left| \frac{\partial \mathbf{f}_{\Theta}(\cdot)}{\partial \mathbf{y}} \right| \right] \\
 &= \sum_{k=1}^K \mathcal{H}(p_{X_k}(\cdot)) - D \left(\prod_{k=1}^K p_{X_k}(\cdot) \parallel p_{\mathbf{X}}(\cdot) \right) + \mathbb{E}_{\mathbf{x}} \left[\log \left| \frac{\partial \mathbf{f}_{\Theta}(\cdot)}{\partial \mathbf{y}} \right| \right]. \quad (16)
 \end{aligned}$$

The latent dimensions are independent if and only if the mutual information is zero, and as such, we seek to minimize it. The entropy in the observed space is fixed, but we have control of the marginal entropies, and we can decrease $\mathbb{E}_{\mathbf{x}} \left[\log \left| \frac{\partial \mathbf{f}(\cdot)}{\partial \mathbf{y}} \right| \right]$ by placing an information bottleneck on $\mathbf{f}(\cdot)$. Thus, by minimizing both these terms, we can minimize latent dependencies.

To that end, we approximate

$$\mathbb{E}_{\mathbf{x}} \left[\log \left| \frac{\partial \mathbf{f}_{\Theta}(\cdot)}{\partial \mathbf{y}} \right| \right] \approx \frac{1}{N} \sum_{n=1}^N \log \left| \frac{\partial \mathbf{f}_{\Theta}(\mathbf{x}_n)}{\partial \mathbf{y}} \right|. \quad (17)$$

Note that an inherent property of the sigmoid nonlinearity is its asymptotic flatness as its output approaches zero. As such, the decoder \mathbf{f}_{Θ} becomes more unable to distinguish between points as their magnitude decreases. Thus, for each representation dimension, by increasing the α parameter of our objective distribution $q(\cdot; \alpha, \beta)$, we can shift more probability mass towards zero: this not only decreases the marginal entropies $\sum_{k=1}^K \mathcal{H}(p_{X_k}(\cdot))$, but also increases the information bottleneck. We add that the flatness of the sigmoid still does not imply true sparsity: a deep transformation can distinguish between small but nonzero values, and as such "undo" the sparsity effect—information is not gone, only bit-shifted. As such, in order to ensure true information loss in the neighbourhood of zero, we also introduce a threshold to the encoder, that kills any representation element less than some $\varepsilon > 0$; namely, we use the encoder $\mathbb{I}_{\mathbf{g}_{\Psi}(\mathbf{y}) \geq \varepsilon} \mathbf{g}_{\Psi}(\mathbf{y})$.

3.4.1 Entropy characterization

A direct consequence of the above is our ability to now place an upper bound on $\mathcal{H}(p_{\mathbf{Y}}(\cdot))$, and, once the mutual information is minimized, to characterize it. This now allows us to make informed choice of a representation space—for example, we understand the interplay between the latent dimensionality and sparsity. Minimizing the mutual information in the way demonstrated above corresponds to selecting this

space to be maximally sparse under the constraint of retaining the information encapsulated in the input examples.

We furthermore note that the entropy of model can be thought of as the "effective number of configurations" that data drawn from its distribution can take. As such, it governs how mass is allocated between the training examples and out-of-sample data: thus, the diversification procedure described dictates the generalizability of the model in a very transparent way.

3.4.2 On why we need $g_{\Psi}(\cdot)$

We can now understand the motivation for introducing $g_{\Psi}(\cdot)$ even with a bijective map. We need $g_{\Psi}(\cdot)$ to induce points in the observed space to be close together in \mathcal{X} , both to control the information capacity as well as to determine the marginals in \mathcal{X} . On the other hand, $f_{\Theta}(\cdot)$ should not expand measure in mapping from \mathcal{X} back to \mathcal{Y} ; this ensures that there is an information bottleneck even under bijectivity. However, asking $f_{\Theta}(\cdot)$ and $f_{\Theta}^{-1}(\cdot)$ to serve this double purpose is contradictory, as one function exactly undoes the contraction of measure performed by the other function. Furthermore, $g_{\Psi}(\cdot)$ is helpful from a computational point of view. In the process of shaping the latent distribution, we must define our requirements as penalties on \mathcal{X} , which we proceed to back-propagate. Since asymptotic flatness of $f_{\Theta}(\cdot)$ means asymptotic steepness of $f_{\Theta}^{-1}(\cdot)$, performing numerical optimization on this function—let alone demanding representation points to be clustered in this asymptotically steep regime, as the divergence penalty requires—is clearly numerically unstable.

4 Training

We train the model on a GPU cluster. In cases of continuous input dimensions, we pre-process the data by whitening and normalizing it. We additionally use PCA to reduce the dimensionality of CIFAR.

In the pretraining stage, we recursively train single-layer deep density models with stochastic gradient descent, where we take the representation space examples of one iteration as the observed space examples of the next.

In the fine-tuning stage, we optimize the objective in Eq. (6) by performing block coordinate descent: we iterate through the layers, and for each layer we take a step to minimize the objective as a function only of that layer’s parameters. Since different gradient magnitudes of distinct layers are not mixed here, this side-steps the problem of having the gradient exponentially decay during back-propagation.

The optimization procedure is designed to have few free parameters: most are actively adapted in the process of optimization. The step size is chosen adaptively via an inexact Armijo’s rule line search. Exact line search on the overall objective is computationally expensive and not possible using mini-batches of data. Secondly, the coefficients $\mu_{\mathcal{I}}, \mu_{\mathcal{D}}, \mu_{\mathcal{R}}$ are adapted to maintain a specified ratio of gradient magnitudes, as the step direction is a mixture of the various penalties and is thus dictated by the relative proportions of their gradients.

We sprinkle masking noise onto the inputs to attain robust solutions, as presented in Vincent et al. [2008]. We also add momentum, but implicitly: we define a window size, and sweep it across the shuffled indices of the training examples, in increments that are a fraction of the window size. As such, each example will be presented in several minibatches in a row, but with each minibatch still introducing new training examples into the objective. We found this implicit momentum technique gives the algorithm a more stable convergence.

4.1 Distribution sequencing and initialization

Enforcing the divergence penalty immediately after initialization results in a highly nonconvex and thus a very challenging optimization problem. The diversification procedure often terminates with an objective distribution Beta $(\cdot; \alpha, \beta)$ with $\alpha \ll 1$. As such, once the algorithm settles near a solution whose empirical distribution in the representation space takes this shape, it will be very difficult for it to “reshuffle” the representation data to attain an alternative solution also with the same distribution. To that end, instead

Dataset	MNIST	CIFAR-10
Training examples	3301.5	-41.8
Test examples (TE)	3343.7	-45.5
Test examples rotated by 90^0	(TE) - 63.9	(TE) - 1.9
TE with 10% of elements corrupted	(TE) - 310.6	(TE) - 123.1

Table 1: Mean log-probabilities of points in the observed space. The test examples are assigned similar probability as the training examples; the rotated test examples are assigned slightly less probability than the test examples; and the corrupted examples are assigned significantly lower probability. Both models were taken to have 3 layers.

of penalizing directly with the final objective distribution, we penalize through a family of distributions $q_j(\cdot) = \text{Beta}(\cdot; \alpha_j, \beta_j)$ over iterations j , where we have $\alpha_0, \beta_0 > 1$ as an “easy” initial problem, and then have the parameter sequences converge as $\alpha_j \rightarrow \alpha, \beta_j \rightarrow \beta$. The transitions between consecutive elements in these sequences are also adaptive: the objective distribution parameters are updated once the current empirical distribution is sufficiently close to the objective distribution.

In addition, we choose the sequence of hyperparameters such that $\frac{\alpha_j}{\alpha_j + \beta_j} = \frac{\alpha}{\alpha + \beta} = \mathbb{E}[\text{Beta}(\cdot; \alpha, \beta)]$, as maintaining a constant expectation improves stability. In accordance with this, we initialize the biases as $\log\left(\frac{\alpha}{\beta}\right)$, which similarly gives rise to a consistent initial expectation. In order to initialize the weight matrices to satisfy the invertibility constraint, we select them to be random orthonormal matrices, with a scaling such that the representation distribution approximately matches in variance the initial objective distribution $\text{Beta}(\cdot; \alpha_0, \beta_0)$.

5 Empirical results

In this section, we examine the properties of the model and learning algorithm on benchmark data: the MNIST digits (60,000 28×28 binary handwritten numerals)¹ and the CIFAR-10 image data² (50,000 32×32 color images). In particular, we examine the global and local properties of the learned densities, via generation and perturbation. We emphasize that the density estimates we report here arise are fully normalized due to the tractability of the Jacobian determinant of our transformation.

5.1 Density evaluation

We start by conducting a variety of tests to examine the quality of the density estimates produced by the DDM.

First, for a probability model to be useful, it should not overfit by distributing most of the mass to training examples, but also assign high probability to unseen out-of-sample data. Similarly, it should assign low density to points in data space that resemble real observations, which in fact are not.

In Table 1, we compute the probabilities assigned by models to their training examples, test examples, and distortions of the test examples.

As an interpretable example, we train a model on examples from MNIST’s digit 9 class, and consider the log-probability it assigns the digit 6 under rotation. Intuitively, we expect the highest density to be assigned to the upside-down 6; see Figure 3. We also add a test-set 9 to demonstrate the density calibration.

We also investigate the marginal entropy of MNIST. We train the model to have a final diversification marginals $\text{Beta}(\cdot; 0.01, 0.4)$. This distribution is extremely peaked at 0 and 1, which justifies approximating the MNIST representation elements as Bernoullis by rounding off to 0 and 1 as $[x_n]_k = \lceil [x_n]_k - \frac{1}{2} \rceil$,

¹<http://yann.lecun.com/exdb/mnist/>

²<http://www.cs.toronto.edu/~kriz/cifar.html>

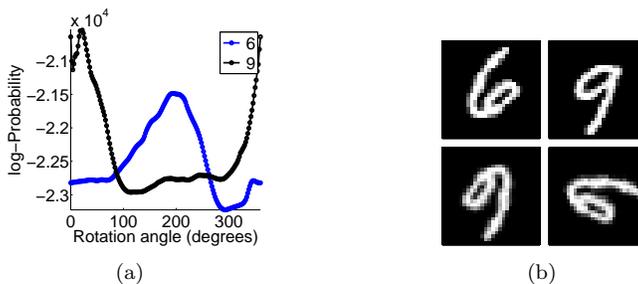


Figure 3: Probabilities of a rotated 6 under a model trained only on 9’s. A real 9 is more probable than an inverted 6. (a) log-densities of a 6 and a 9 as a function of the angle of rotation. (b) Respectively: the original 6; a 9; the rotation of the 6 that the 9-model assigned the highest probability to; and the rotation that the 9-model assigned the lowest probability to.

which corresponds to Bernoulli parameter $p = \int_{1/2}^1 \frac{\Gamma(0.01+0.4)}{\Gamma(0.01)\Gamma(0.4)} x^{0.01-1} (1-x)^{0.4-1} dx \approx 0.0224$. We now note that, by the law of large numbers, as $K \rightarrow \infty$, we expect the mean log-probability of the Bernoulli test examples to approach $-KH(\text{Bern}(\cdot; 0.0465)) = 21.0181\dots$. Indeed, we compute that $\frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K \log \text{Bern}([x_n]_k; 0.0465) = 20.7217\dots$. Thus, we see that the marginal entropy of the model is close to our expectation of it.

5.2 Generation

The approximate independence due to the process of diversification described in Subsection 3.4 combined with the invertibility property of the decoder allows us to produce real samples from the density estimation on the observed space, by sampling within the representation space. Such instantiations of the distribution provide us with visualizations of regions of high density under it; see Figure 4.

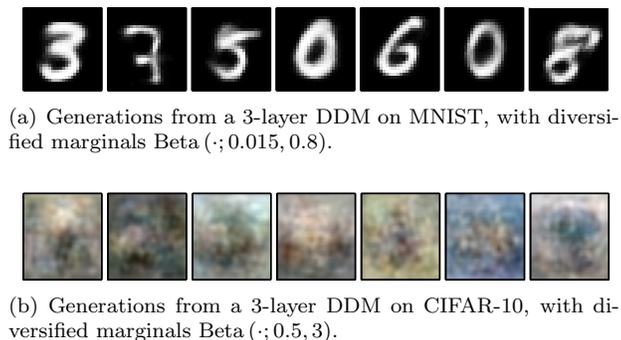


Figure 4: Generations from DDMs.

6 Discussion

In this paper, we have proposed a new way to construct normalized probability density estimates from high-dimensional data. Our approach borrows ideas from deep learning, differential geometry, and information theory to ensure that the learned distributions are rich, but tractable and normalizable.

There are several interesting directions that we believe such a density model opens up. One advantage of a fully-normalized density model is that it enables Bayesian probabilistic classifiers to be constructed to provide calibrated conditional distributions over class membership. If the data are drawn from a mixture

of C classes, we may compute $p(\cdot; c)$, the likelihood that an element is drawn from class c by training a model with only the examples under that class. It is then possible to, e.g., estimate the most likely class:

$$c_n^* = \arg \max_c p(\mathbf{y}_n; c) \quad \forall n = 1, \dots, N.$$

Moreover, since we now have class-conditional densities, we may reject classification of examples if no class model is confident enough to own them. Namely, we can set a threshold parameter λ , and reject classification of example \mathbf{y}_n if $p(\mathbf{y}_n; c) < \lambda \forall c = 1, \dots, C$.

Lastly, we can not only control the density at the observations, but also the distribution of mass in the rest of the latent space. Specifically, instead of letting each class model see only its own examples, we can expose it to training examples from other classes, and demand it assigns them as little probability as possible. This not only teaches the model to recognize examples that lie on the manifold of its class, but also identify differences from other examples by mapping them away from this manifold.

We tested the above ideas on MNIST, and found that a raw mixture of Bayesian classifiers gives us an error rate of 9.5%. However, penalizing density assigned to foreign examples results in a model that achieves a much lower error rate of 1.614%. As the model provides calibrated probabilistic predictions, it is also able to assess its confidence when making classifications. Among examples in which the model is confident (approximately 95% of the test data), the Bayesian DDM classifier achieves 0.45% error. Although these are not state-of-the-art rates, they show the flexibility of classifier construction in the DDM setting and how the normalized density can be leveraged *across* separately trained models, something not typically possible for energy-based probabilistic approaches.

We can extend these ideas further to use the deep density model even if only a small fraction of the observations are labelled. That is, we can use density estimates to extract useful information from unlabeled data by leveraging our knowledge of the empirical density in the representation space. One possible approach is to run the expectation-maximization algorithm and train the DDM on weighted data as part of a mixture model.

References

- P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. 1986.
- G.E. Hinton, S. Osindero, and Y.W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.
- J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.
- Radford M. Neal. Connectionist learning in belief networks. *Artificial Intelligence*, 56:71–113, July 1992.
- Ryan P. Adams, Hanna M. Wallach, and Zoubin Ghahramani. Learning the structure of deep sparse graphical models. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010.
- Michael D. Escobar and Mike West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430):577–588, June 1995.
- Carl Edward Rasmussen. The infinite Gaussian mixture model. In *Advances in Neural Information Processing Systems 12*, pages 554–560, 2000.
- Ryan P. Adams, Iain Murray, and David J. C. MacKay. The Gaussian process density sampler. In *Advances in Neural Information Processing Systems 21*, 2009.
- S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

- J.B. Tenenbaum, V. De Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.
- N. Lawrence. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research*, 6:1783–1816, 2005.
- B. Schölkopf, A. Smola, and K.R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.
- L. Van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.
- Garrison W. Cottrell, Paul Munro, and David Zipser. Learning internal representations from gray-scale images: An example of extensional programming. In *Conference of the Cognitive Science Society*, 1987.
- Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
- Salah Rifai, Yoshua Bengio, Yann Dauphin, and Pascal Vincent. A generative process for sampling contractive auto-encoders. In *International Conference on Machine Learning*, 2012.
- Pascal Vincent and Yoshua Bengio. Manifold Parzen windows. In *Advances in Neural Information Processing Systems 15*, pages 825–832. MIT Press, 2002.
- M. Titsias and N. Lawrence. Bayesian Gaussian process latent variable model. In *International Conference on Artificial Intelligence and Statistics*, 2010.
- A.J. Bell and T.J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159, 1995.
- David J.C. MacKay and Mark N. Gibbs. Density networks. In *Proceedings of Society for General Microbiology Edinburgh meeting*, 1997.
- Christian Walder and Bernhard Schölkopf. Diffeomorphic dimensionality reduction. In *Advances in Neural Information Processing Systems 22*, 2008.
- Neil D. Lawrence and Joaquin Quiñonero Candela. Local distance preservation in the GP-LVM through back constraints. In *International Conference on Machine Learning*, pages 513–520, 2006.
- 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, 2008.