Generative Moment Matching Networks

Yujia Li[1] YUJIALI@CS.TORONTO.EDU
Kevin Swersky[1] KSWERSKY@CS.TORONTO.EDU
Richard Zemel[1,2] ZEMEL@CS.TORONTO.EDU
[1]Department of Computer Science, University of Toronto, Toronto, ON, CANADA
[2]Canadian Institute for Advanced Research, Toronto, ON, CANADA

Abstract

We consider the problem of learning deep generative models from data. We formulate a method that generates an independent sample via a single feedforward pass through a multilayer perceptron, as in the recently proposed generative adversarial networks (Goodfellow et al., 2014). Training a generative adversarial network, however, requires careful optimization of a difficult minimax program. Instead, we utilize a technique from statistical hypothesis testing known as maximum mean discrepancy (MMD), which leads to a simple objective that can be interpreted as matching all orders of statistics between a dataset and samples from the model, and can be trained by backpropagation. We further boost the performance of this approach by combining our generative network with an auto-encoder network, using MMD to learn to generate codes that can then be decoded to produce samples. We show that the combination of these techniques yields excellent generative models compared to baseline approaches as measured on MNIST and the Toronto Face Database.

1. Introduction

The most visible successes in the area of deep learning have come from the application of deep models to supervised learning tasks. Models such as convolutional neural networks (CNNs), and long short term memory (LSTM) networks are now achieving impressive results on a number of tasks such as object recognition (Krizhevsky et al., 2012; Sermanet et al., 2014; Szegedy et al., 2014), speech recognition (Graves & Jaitly, 2014; Hinton et al., 2012a), image caption generation (Vinyals et al., 2014; Fang et al., 2014; Kiros et al., 2014), machine translation (Cho et al., 2014; Sutskever et al., 2014), and more. Despite their successes, one of the main bottlenecks of the supervised approach is the difficulty in obtaining enough data to learn abstract features that capture the rich structure of the data. It is well recognized that a promising avenue is to use unsupervised learning on unlabelled data, which is far more plentiful and cheaper to obtain.

A long-standing and inherent problem in unsupervised learning is defining a good method for evaluation. Generative models offer the ability to evaluate generalization in the data space, which can also be qualitatively assessed. In this work we propose a generative model for unsupervised learning that we call generative moment matching networks (GMMNs). GMMNs are generative neural networks that begin with a simple prior from which it is easy to draw samples. These are propagated deterministically through the hidden layers of the network and the output is a sample from the model. Thus, with GMMNs it is easy to quickly draw independent random samples, as opposed to expensive MCMC procedures that are necessary in other models such as Boltzmann machines (Ackley et al., 1985; Hinton, 2002; Salakhutdinov & Hinton, 2009). The structure of a GMMN is most analogous to the recently proposed generative adversarial networks (GANs) (Goodfellow et al., 2014), however unlike GANs, whose training involves a difficult minimax optimization problem, GMMNs are comparatively simple; they are trained to minimize a straightforward loss function using backpropagation.

The key idea behind GMMNs is the use of a statistical hypothesis testing framework called maximum mean discrepancy (Gretton et al., 2007). Training a GMMN to minimize this discrepancy can be interpreted as matching all moments of the model distribution to the empirical data distribution. Using the kernel trick, MMD can be represented as a simple loss function that we use as the core training objective for GMMNs. Using minibatch stochastic gradient descent, training can be kept efficient, even with large datasets.

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

As a second contribution, we show how GMMNs can be used to bootstrap auto-encoder networks in order to further improve the generative process. The idea behind this approach is to train an auto-encoder network and then apply a GMMN to the code space of the auto-encoder. This allows us to leverage the rich representations learned by auto-encoder models as the basis for comparing data and model distributions. To generate samples in the original data space, we simply sample a code from the GMMN and then use the decoder of the auto-encoder network.

Our experiments show that this relatively simple, yet very flexible framework is effective at producing good generative models in an efficient manner. On MNIST and the Toronto Face Dataset (TFD) we demonstrate improved results over comparable baselines, including GANs. Source code for training GMMNs is available at `https://github.com/yujiali/gmmn`.

2. Maximum Mean Discrepancy

Suppose we are given two sets of samples $X = \{x_i\}_{i=1}^N$ and $Y = \{y_j\}_{j=1}^M$ and are asked whether the generating distributions $P_X = P_Y$. Maximum mean discrepancy is a frequentist estimator for answering this question, also known as the two sample test (Gretton et al., 2007; 2012a). The idea is simple: compare statistics between the two datasets and if they are similar then the samples are likely to come from the same distribution.

Formally, the following MMD measure computes the mean squared difference of the statistics of the two sets of samples.

$$\mathcal{L}_{\text{MMD}^2} = \left\| \frac{1}{N}\sum_{i=1}^{N}\phi(x_i) - \frac{1}{M}\sum_{j=1}^{M}\phi(y_j) \right\|^2 \tag{1}$$

$$= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'=1}^{N}\phi(x_i)^\top\phi(x_{i'}) - \frac{2}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\phi(x_i)^\top\phi(y_j) + \frac{1}{M^2}\sum_{j=1}^{M}\sum_{j'=1}^{M}\phi(y_j)^\top\phi(y_{j'}) \tag{2}$$

Taking ϕ to be the identity function leads to matching the sample mean, and other choices of ϕ can be used to match higher order moments.

Written in this form, each term in Equation (2) only involves inner products between the ϕ vectors, and therefore the kernel trick can be applied.

$$\mathcal{L}_{\text{MMD}^2} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'=1}^{N}k(x_i, x_{i'}) - \frac{2}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}k(x_i, y_j) + \frac{1}{M^2}\sum_{j=1}^{M}\sum_{j'=1}^{M}k(y_j, y_{j'}) \tag{3}$$

The kernel trick implicitly lifts the sample vectors into an infinite dimensional feature space. When this feature space corresponds to a universal reproducing kernel Hilbert space, it is shown that asymptotically, MMD is 0 if and only if $P_X = P_Y$ (Gretton et al., 2007; 2012a).

For universal kernels like the Gaussian kernel, defined as $k(x, x') = \exp(-\frac{1}{2\sigma}|x - x'|^2)$, where σ is the bandwidth parameter, we can use a Taylor expansion to get an explicit feature map ϕ that contains an infinite number of terms and covers all orders of statistics. Minimizing MMD under this feature expansion is then equivalent to minimizing a distance between *all* moments of the two distributions.

3. Related Work

In this work we focus on generative models due to their ability to capture the salient properties and structure of data. Deep generative models are particularly appealing because they are capable of learning a latent manifold on which the data has high density. Learning this manifold allows smooth variations in the latent space to result in non-trivial transformations in the original space, effectively traversing between high density modes through low density areas (Bengio et al., 2013a). They are also capable of disentangling factors of variation, which means that each latent variable can become responsible for modelling a single, complex transformation in the original space that would otherwise involve many variables (Bengio et al., 2013a). Even if we restrict ourselves to the field of deep learning, there are a vast array of approaches to generative modelling. Below, we outline some of these methods.

One popular class of generative models used in deep learning are undirected graphical models, such as Boltzmann machines (Ackley et al., 1985), restricted Boltzmann machines (Hinton, 2002), and deep Boltzmann machines (Salakhutdinov & Hinton, 2009). These models are normalized by a typically intractable partition function, making training, evaluation, and sampling extremely difficult, usually requiring expensive Markov-chain Monte Carlo (MCMC) procedures.

Next there is the class of fully visible directed models such as fully visible sigmoid belief networks (Neal, 1992) and the neural autoregressive distribution estimator (Larochelle & Murray, 2011). These admit efficient log-likelihood calculation, gradient-based learning and efficient sampling, but require that an ordering be imposed on the observable variables, which can be unnatural for domains such as images and cannot take advantage of parallel computing methods due to their sequential nature.

More related to our own work, there is a line of research devoted to recovering density models from auto-encoder networks using MCMC procedures (Rifai et al., 2012; Bengio

et al., 2013b; 2014). These attempt to use contraction operators, or denoising criteria in order to generate a Markov chain by repeated perturbations during the encoding phase, followed by decoding.

Also related to our own work is the class of deep, variational networks (Rezende et al., 2014; Kingma & Welling, 2014; Mnih & Gregor, 2014). These are also deep, directed generative models, however they make use of an additional neural network that is designed to approximate the posterior over the latent variables. Training is carried out via a variational lower bound on the log-likelihood of the model distribution. These models are trained using stochastic gradient descent, however they either require that the latent representation is continuous (Kingma & Welling, 2014), or require many secondary networks to sufficiently reduce the variance of gradient estimates in order to produce a sufficiently good learning signal (Mnih & Gregor, 2014).

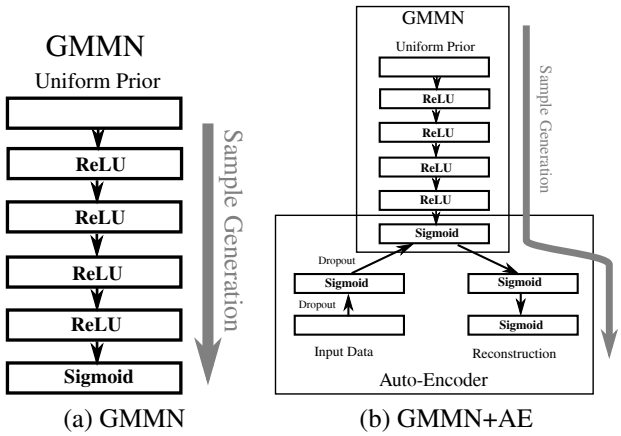
Finally there is some early work that proposed the idea of using feed-forward neural networks to learn generative models. MacKay (1995) proposed a model that is closely related to ours, which also used a feed-forward network to map the prior samples to the data space. However, instead of directly outputing samples, an extra distribution is associated with the output. Sampling was used extensively for learning and inference in this model. Magdon-Ismail & Atiya (1998) proposed to use a neural network to learn a transformation from the data space to another space where the transformed data points are uniformly distributed. This transformation network then learns the cumulative density function. In recent independent work, Dziugaite et al. (2015) proposed an idea very similar to our's, which trains a feed-forward neural network generative model by optimizing MMD. A more thorough comparison with their approach is left to future work.

4. Generative Moment Matching Networks

4.1. Data Space Networks

The high-level idea of the GMMN is to use a neural network to learn a deterministic mapping from samples of a simple, easy to sample distribution, to samples from the data distribution. The architecture of the generative network is exactly the same as a generative adversarial network (Goodfellow et al., 2014). However, we propose to train the network by simply minimizing the MMD criterion, avoiding the hard minimax objective function used in generative adversarial network training.

More specifically, in the generative network we have a stochastic hidden layer $\mathbf{h} \in \mathbb{R}^H$ with H hidden units at the top with a prior uniform distribution on each unit independently,



(a) GMMN (b) GMMN+AE

Figure 1. Example architectures of our generative moment matching networks. (a) GMMN used in the input data space. (b) GMMN used in the code space of an auto-encoder.

$$p(\mathbf{h}) = \prod_{j=1}^{H} U(h_j) \tag{4}$$

Here $U(h) = \frac{1}{2}\mathbf{I}[-1 \leq h \leq 1]$ is a uniform distribution in $[-1, 1]$, where $\mathbf{I}[.]$ is an indicator function. Other choices for the prior are also possible, as long as it is a simple enough distribution from which we can easily draw samples.

The $\mathbf{h}$ vector is then passed through the neural network and deterministically mapped to a vector $\mathbf{x} \in \mathbb{R}^D$ in the D dimensional data space.

$$\mathbf{x} = f(\mathbf{h}; \mathbf{w}) \tag{5}$$

f is the neural network mapping function, which can contain multiple layers of nonlinearities, and $\mathbf{w}$ represents the parameters of the neural network. One example architecture for f is illustrated in Figure 1(a), which has 4 intermediate ReLU (Nair & Hinton, 2010) nonlinear layers and one logistic sigmoid output layer.

The prior $p(\mathbf{h})$ and the mapping $f(\mathbf{h}; \mathbf{w})$ jointly defines a distribution $p(\mathbf{x})$ in the data space. To generate a sample $\mathbf{x} \sim p(\mathbf{x})$ we only need to sample from the uniform prior $p(\mathbf{h})$ and then pass the sample $\mathbf{h}$ through the neural net to get $\mathbf{x} = f(\mathbf{h}; \mathbf{w})$.

Goodfellow et al. (2014) proposed to train this network by using an extra discriminative network, which tries to distinguish between model samples and data samples. The generative network is then trained to counteract this in order to make the samples indistinguishable to the discriminative network. The gradient of this objective can be backpropagated through the generative network. However, because of the minimax nature of the formulation, it is easy to get

stuck at a local optimum. So the training of generative network and the discriminative network must be interleaved and carefully scheduled. By contrast, our learning algorithm simply involves minimizing the MMD objective.

Assume we have a dataset of training examples $\mathbf{x}_1^d, ..., \mathbf{x}_N^d$ (d for data), and a set of samples generated from our model $\mathbf{x}_1^s, ..., \mathbf{x}_M^s$ (s for samples). The MMD objective $\mathcal{L}_{\text{MMD}^2}$ is differentiable when the kernel is differentiable. For example for Gaussian kernels $k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{2\sigma}||\mathbf{x} - \mathbf{y}||^2\right)$, the gradient of $\mathbf{x}_{ip}^s$ has a simple form

$$\frac{\partial \mathcal{L}_{\text{MMD}^2}}{\partial \mathbf{x}_{ip}^s} = \frac{2}{M^2} \sum_{j=1}^{M} \frac{1}{\sigma} k(\mathbf{x}_i^s, \mathbf{x}_j^s)(\mathbf{x}_{jp}^s - \mathbf{x}_{ip}^s) - \frac{2}{MN} \sum_{j=1}^{N} \frac{1}{\sigma} k(\mathbf{x}_i^s, \mathbf{x}_j^d)(\mathbf{x}_{jp}^d - \mathbf{x}_{ip}^s) \quad (6)$$

This gradient can then be backpropagated through the generative network to update the parameters $\mathbf{w}$.

4.2. Auto-Encoder Code Space Networks

Real-world data can be complicated and high-dimensional, which is one reason why generative modelling is such a difficult task. Auto-encoders, on the other hand, are designed to solve an arguably simpler task of reconstruction. If trained properly, auto-encoder models can be very good at representing data in a code space that captures enough statistical information that the data can be reliably reconstructed.

The code space of an auto-encoder has several advantages for creating a generative model. The first is that the dimensionality can be explicitly controlled. Visual data, for example, while represented in a high dimensional space, often exists on a low-dimensional manifold. This is beneficial for a statistical estimator like MMD because the amount of data required to produce a reliable estimate grows with the dimensionality of the data (Ramdas et al., 2015). The second advantage is that each dimension of the code space can end up representing complex variations in the original data space. This concept is referred to in the literature as disentangling factors of variation (Bengio et al., 2013a).

For these reasons, we propose to bootstrap auto-encoder models with a GMMN to create what we refer to as the GMMN+AE model. These operate by first learning an auto-encoder and producing code representations of the data, then freezing the auto-encoder weights and learning a GMMN to minimize MMD between generated codes and data codes. A visualization of this model is given in Figure 1(b).

Our method for training a GMMN+AE proceeds as follows:

1. Greedy layer-wise pretraining of the auto-encoder (Bengio et al., 2007).
2. Fine-tune the auto-encoder.
3. Train a GMMN to model the code layer distribution using an MMD objective on the final encoding layer.

We found that adding dropout to the encoding layers can be beneficial in terms of creating a smooth manifold in code space. This is analogous to the motivation behind contractive and denoising auto-encoders (Rifai et al., 2011; Vincent et al., 2008).

4.3. Practical Considerations

Here we outline some design choices that we have found to improve the peformance of GMMNs.

Bandwidth Parameter. The bandwidth parameter in the kernel plays a crucial role in determining the statistical efficiency of MMD, and optimally setting it is an open problem. A good heuristic is to perform a line search to obtain the bandwidth that produces the maximal distance (Sriperumbudur et al., 2009), other more advanced heuristics are also available (Gretton et al., 2012b). As a simpler approximation, for most of our experiments we use a mixture of K kernels spanning multiple ranges. That is, we choose the kernel to be:

$$k(x, x') = \sum_{q=1}^{K} k_{\sigma_q}(x, x') \quad (7)$$

where k_{σ_q} is a Gaussian kernel with bandwidth parameter σ_q. We found that choosing simple values for these such as $1, 5, 10$, etc. and using a mixture of 5 or more was sufficient to obtain good results. The weighting of different kernels can be further tuned to achieve better results, but we kept them equally weighted for simplicity.

Square Root Loss. In practice, we have found that better results can be obtained by optimizing $\mathcal{L}_{\text{MMD}} = \sqrt{\mathcal{L}_{\text{MMD}^2}}$. This loss can be important for driving the difference between the two distributions as close to 0 as possible. Compared to $\mathcal{L}_{\text{MMD}^2}$ which flattens out when its value gets close to 0, $\mathcal{L}_{\text{MMD}}$ behaves much better for small $\mathcal{L}_{\text{MMD}}$ values. Alternatively, this can be understood by writing down the gradient of $\mathcal{L}_{\text{MMD}}$ with respect to $\mathbf{w}$

$$\frac{\partial \mathcal{L}_{\text{MMD}}}{\partial \mathbf{w}} = \frac{1}{2\sqrt{\mathcal{L}_{\text{MMD}^2}}} \frac{\partial \mathcal{L}_{\text{MMD}^2}}{\partial \mathbf{w}} \quad (8)$$

The $1/(2\sqrt{\mathcal{L}_{\text{MMD}^2}})$ term automatically adapts the effective learning rate. This is especially beneficial when both $\mathcal{L}_{\text{MMD}^2}$ and $\frac{\partial \mathcal{L}_{\text{MMD}^2}}{\partial \mathbf{w}}$ become small, where this extra factor can help by maintaining larger gradients.

Algorithm 1: GMMN minibatch training

Input : Dataset $\{\mathbf{x}_1^d, ..., \mathbf{x}_N^d\}$, prior $p(\mathbf{h})$, network $f(\mathbf{h}; \mathbf{w})$ with initial parameter $\mathbf{w}^{(0)}$

Output: Learned parameter $\mathbf{w}^*$

while *Stopping criterion not met* **do**
  Get a minibatch of data $\mathbf{X}^d \leftarrow \{\mathbf{x}_{i_1}^d, ..., \mathbf{x}_{i_b}^d\}$
  Get a new set of samples $\mathbf{X}^s \leftarrow \{\mathbf{x}_1^s, ..., \mathbf{x}_b^s\}$
  Compute gradient $\frac{\partial \mathcal{L}_{\text{MMD}}}{\partial \mathbf{w}}$ on $\mathbf{X}^d$ and $\mathbf{X}^s$
  Take a gradient step to update $\mathbf{w}$
end

Minibatch Training. One of the issues with MMD is that the usage of kernels means that the computation of the objective scales quadratically with the amount of data. In the literature there have been several alternative estimators designed to overcome this (Gretton et al., 2012a). In our case, we found that it was sufficient to optimize MMD using minibatch optimization. In each weight update, a small subset of data is chosen, and an equal number of samples are drawn from the GMMN. Within a minibatch, MMD is applied as usual. As we are using exact samples from the model and the data distribution, the minibatch MMD is still a good estimator of the population MMD. We found this approach to be both fast and effective. The minibatch training algorithm for GMMN is shown in Algorithm 1.

5. Experiments

We trained GMMNs on two benchmark datasets: MNIST (LeCun et al., 1998) and the Toronto Face Dataset (TFD) (Susskind et al., 2010). For MNIST, we used the standard test set of 10,000 images, and split out 5000 from the standard 60,000 training images for validation. The remaining 55,000 were used for training. For TFD, we used the same training and test sets and fold splits as used by (Goodfellow et al., 2014), but split out a small set of the training data and used it as the validation set. For both datasets, rescaling the images to have pixel intensities between 0 and 1 is the only preprocessing step we did.

On both datasets, we trained the GMMN network in both the input data space and the code space of an auto-encoder. For all the networks we used in this section, a uniform distribution in $[-1, 1]^H$ was used as the prior for the H-dimensional stochastic hidden layer at the top of the GMMN, which was followed by 4 ReLU layers, and the output was a layer of logistic sigmoid units. The auto-encoder we used for MNIST had 4 layers, 2 for the encoder and 2 for the decoder. For TFD the auto-encoder had 6 layers in total, 3 for the encoder and 3 for the decoder. For both auto-encoders the encoder and the decoder had mirrored architectures. All layers in the auto-encoder network used sigmoid nonlinearities, which also guaranteed that the code space dimensions lay in $[0, 1]$, so that they could match the GMMN outputs. The network architectures for MNIST are shown in Figure 1.

The auto-encoders were trained separately from the GMMN. Cross entropy was used as the reconstruction loss. We first did standard layer-wise pretraining, then fine-tuned all layers jointly. Dropout (Hinton et al., 2012b) was used on the encoder layers. After training the auto-encoder, we fixed it and passed the input data through the encoder to get the corresponding codes. The GMMN network was then trained in this code space to match the statistics of generated codes to the statistics of codes from data examples. When generating samples, the generated codes were passed through the decoder to get samples in the input data space.

For all experiments in this section the GMMN networks were trained with minibatches of size 1000, for each minibatch we generated a set of 1000 samples from the network. The loss and gradient were computed from these 2000 points. We used the square root loss function $\mathcal{L}_{\text{MMD}}$ throughout.

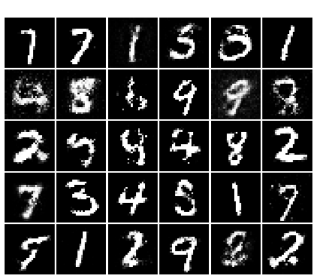
Evaluation of our model is not straight-forward, as we do not have an explicit form for the probability density function, it is not easy to compute the log-likelihood of data. However, sampling from our model is easy. We therefore followed the same evaluation protocol used in related models (Bengio et al., 2013a), (Bengio et al., 2014), and (Goodfellow et al., 2014). A Gaussian Parzen window (kernel density estimator) was fit to 10,000 samples generated from the model. The likelihood of the test data was then computed under this distribution. The scale parameter of the Gaussians was selected using a grid search in a fixed range using the validation set.

The hyperparameters of the networks, including the learning rate and momentum for both auto-encoder and GMMN training, dropout rate for the auto-encoder, and number of hidden units on each layer of both auto-encoder and GMMN, were tuned using Bayesian optimization (Snoek et al., 2012; 2014)[1] to optimize the validation set likelihood under the Gaussian Parzen window density estimation.

The log-likelihood of the test set for both datasets are shown in Table 1. The GMMN is competitive with other approaches, while the GMMN+AE significantly outperforms the other models. This shows that despite being relatively simple, MMD, especially when combined with an effective decoder, is a powerful objective for training good generative models.

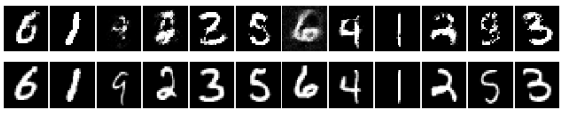
Some samples generated from the GMMN models are

[1]We used the service provided by https://www.whetlab.com

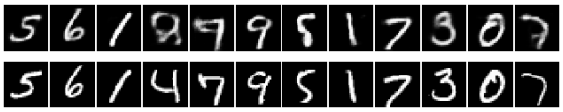


(a) GMMN MNIST samples

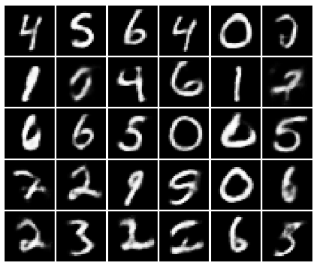
(b) GMMN TFD samples



(e) GMMN nearest neighbors for MNIST samples



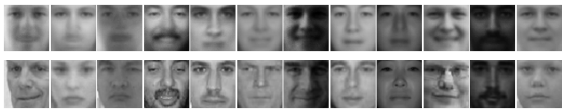
(f) GMMN+AE nearest neighbors for MNIST samples



(c) GMMN+AE MNIST samples

(d) GMMN+AE TFD samples

(g) GMMN nearest neighbors for TFD samples



(h) GMMN+AE nearest neighbors for TFD samples

Figure 2. Independent samples and their nearest neighbors in the training set for the GMMN+AE model trained on MNIST and TFD datasets. For (e)(f)(g) and (h) the top row are the samples from the model and the bottom row are the corresponding nearest neighbors from the training set measured by Euclidean distance.

Model	MNIST	TFD
DBN	138 ± 2	1909 ± 66
Stacked CAE	121 ± 1.6	2110 ± 50
Deep GSN	214 ± 1.1	1890 ± 29
Adversarial nets	225 ± 2	2057 ± 26
GMMN	147 ± 2	2085 ± 25
GMMN+AE	$\mathbf{282 \pm 2}$	$\mathbf{2204 \pm 20}$

Table 1. Log-likelihood of the test sets under different models. The baselines are Deep Belief Net (DBN) and Stacked Contractive Auto-Encoder (Stacked CAE) from (Bengio et al., 2013a), Deep Generative Stochastic Network (Deep GSN) from (Bengio et al., 2014) and Adversarial nets (GANs) from (Goodfellow et al., 2014).

shown in Figure 2(a-d). The GMMN+AE produces the most visually appealing samples, which are reflected in its Parzen window log-likelihood estimates. The likely explanation is that any perturbations in the code space correspond to smooth transformations along the manifold of the data space. In that sense, the decoder is able to "correct" noise in the code space.

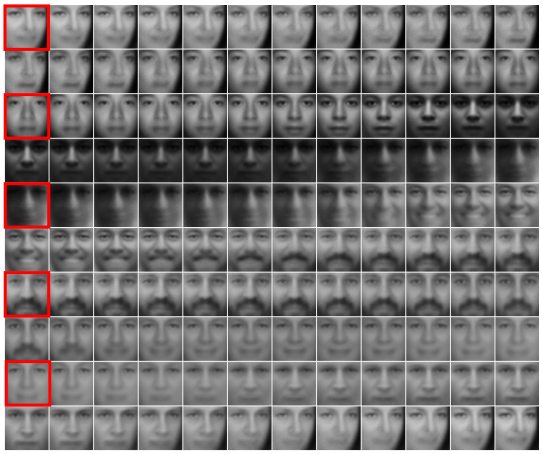
To determine whether the models learned to merely copy the data, we follow the example of (Goodfellow et al., 2014) and visualize the nearest neighbour of several samples in terms of Euclidean pixel-wise distance in Figure 2(e-h). By this metric, it appears as though the samples are not merely data examples.

One of the interesting aspects of a deep generative model such as the GMMN is that it is possible to directly explore the data manifold. Using the GMMN+AE model, we randomly sampled 5 points in the uniform space and show their corresponding data space projections in Figure 3. These points are highlighted by red boxes. From left to right, top to bottom we linearly interpolate between these points in the uniform space and show their corresponding projections in data space. The manifold is smooth for the most part, and almost all of the projections correspond to realistic looking data. For TFD in particular, these transformations involve complex attributes, such as the changing of pose, expression, lighting, gender, and facial hair. More results can be found at `http://www.cs.toronto.edu/~yujiali/proj/gmmn.html`.

6. Conclusion and Future Work

In this paper we provide a simple and effective framework for training deep generative models called generative moment matching networks. Our approach is based off of optimizing maximum mean discrepancy so that samples generated from the model are indistinguishable from data examples in terms of their moment statistics. As is standard with MMD, the use of the kernel trick allows a GMMN to avoid explicitly computing these moments, resulting in a simple

(a) MNIST interpolation



(b) TFD interpolation

Figure 3. Linear interpolation between 5 uniform random points from the GMMN+AE prior projected through the network into data space for (a) MNIST and (b) TFD. The 5 random points are highlighted with red boxes, and the interpolation goes from left to right, top to bottom. The final two rows represent an interpolation between the last highlighted image and the first highlighted image.

training objective, and the use of minibatch stochastic gradient descent allows the training to scale to large datasets.

Our second contribution combines MMD with auto-encoders for learning a generative model of the code layer. The code samples from the model can then be fed through the decoder in order to generate samples in the original space. The use of auto-encoders makes the generative model learning a much simpler problem. Combined with MMD, pretrained auto-encoders can be readily bootstrapped into a good generative model of data. On the MNIST and Toronto Face Database, the GMMN+AE model achieves superior performance compared to other approaches. For these datasets, we demonstrate that the GMMN+AE is able to discover the implicit manifold of the data.

There are many interesting directions for research using MMD. One such extension is to consider alternatives to the standard MMD criterion in order to speed up training. One such possibility is the class of linear-time estimators that has been developed recently in the literature (Gretton et al., 2012a).

Another possibility is to utilize random features (Rahimi & Recht, 2007). These are randomized feature expansions whose inner product converges to a kernel function with an increasing number of features. This idea was recently explored for MMD in (Zhao & Meng, 2014). The advantage of this approach would be that the cost would no longer grow quadratically with minibatch size because we could use the original objective given in Equation 2. Another advantage of this approach is that the data statistics could be pre-computed from the entire dataset, which would reduce the variance of the objective gradients.

Another direction we would like to explore is joint training of the auto-encoder model with the GMMN. Currently, these are treated separately, but joint training may encourage the learning of codes that are both suitable for reconstruction as well as generation.

While a GMMN provides an easy way to sample data, the posterior distribution over the latent variables is not readily available. It would be interesting to explore ways in which to infer the posterior distribution over the latent space. A straightforward way to do this is to learn a neural network to predict the latent vector given a sample. This is reminiscent of the recognition models used in the wake-sleep algorithm (Hinton et al., 1995), or variational auto-encoders (Kingma & Welling, 2014).

An interesting application of MMD that is not directly related to generative modelling comes from recent work on learning fair representations (Zemel et al., 2013). There, the objective is to train a prediction method that is invariant to a particular sensitive attribute of the data. Their solution is to learn an intermediate clustering-based representation. MMD could instead be applied to learn a more powerful, distributed representation such that the statistics of the representation do not change conditioned on the sensitive variable. This idea can be further generalized to learn representations invariant to known biases.

Finally, the notion of utilizing an auto-encoder with the GMMN+AE model provides new avenues for creating generative models of even more complex datasets. For example, it may be possible to use a GMMN+AE with convolutional auto-encoders (Zeiler et al., 2010; Masci et al., 2011;

Makhzani & Frey, 2014) in order to create generative models of high resolution color images.

Acknowledgements

We thank David Warde-Farley for helpful clarifications regarding (Goodfellow et al., 2014), and Charlie Tang for providing relevant references. We thank CIFAR, NSERC, and Google for research funding.

References

Ackley, David H, Hinton, Geoffrey E, and Sejnowski, Terrence J. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

Bengio, Yoshua, Lamblin, Pascal, Popovici, Dan, Larochelle, Hugo, et al. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

Bengio, Yoshua, Mesnil, Grégoire, Dauphin, Yann, and Rifai, Salah. Better mixing via deep representations. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 2013a.

Bengio, Yoshua, Yao, Li, Alain, Guillaume, and Vincent, Pascal. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, pp. 899–907, 2013b.

Bengio, Yoshua, Thibodeau-Laufer, Eric, Alain, Guillaume, and Yosinski, Jason. Deep generative stochastic networks trainable by backprop. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2014.

Cho, Kyunghyun, van Merrienboer, Bart, Gulcehre, Caglar, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

Dziugaite, Gintare Karolina, Roy, Daniel M., and Ghahramani, Zoubin. Training generative neural networks via maximum mean discrepancy optimization. In *Uncertainty in Artificial Intelligence (UAI)*, 2015.

Fang, Hao, Gupta, Saurabh, Iandola, Forrest, Srivastava, Rupesh, Deng, Li, Dollár, Piotr, Gao, Jianfeng, He, Xiaodong, Mitchell, Margaret, Platt, John, Zitnick, C. Lawrence, and Zweig, Geoffrey. From captions to visual concepts and back. *arXiv preprint arXiv:1411.4952*, 2014.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

Graves, Alex and Jaitly, Navdeep. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1764–1772, 2014.

Gretton, Arthur, Borgwardt, Karsten M, Rasch, Malte, Schölkopf, Bernhard, and Smola, Alex J. A kernel method for the two-sample-problem. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

Gretton, Arthur, Borgwardt, Karsten M, Rasch, Malte J, Schölkopf, Bernhard, and Smola, Alexander. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012a.

Gretton, Arthur, Sejdinovic, Dino, Strathmann, Heiko, Balakrishnan, Sivaraman, Pontil, Massimiliano, Fukumizu, Kenji, and Sriperumbudur, Bharath K. Optimal kernel choice for large-scale two-sample tests. In *Advances in Neural Information Processing Systems*, pp. 1205–1213, 2012b.

Hinton, Geoffrey E. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14 (8):1771–1800, 2002.

Hinton, Geoffrey E, Dayan, Peter, Frey, Brendan J, and Neal, Radford M. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

Hinton, Geoffrey E., Deng, Li, Yu, Dong, Dahl, George E., Mohamed, Abdel-rahman, Jaitly, Navdeep, Senior, Andrew, Vanhoucke, Vincent, Nguyen, Patrick, Sainath, Tara N., and Kingsbury, Brian. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.*, 29(6):82–97, 2012a.

Hinton, Geoffrey E, Srivastava, Nitish, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012b.

Kingma, Diederik P. and Welling, Max. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

Kiros, Ryan, Salakhutdinov, Ruslan, and Zemel, Richard S. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

Larochelle, Hugo and Murray, Iain. The neural autoregressive distribution estimator. In *roceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

MacKay, David JC. Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 354(1):73–80, 1995.

Magdon-Ismail, Malik and Atiya, Amir. Neural networks for density estimation. In *NIPS*, pp. 522–528, 1998.

Makhzani, Alireza and Frey, Brendan. A winner-take-all method for training sparse convolutional autoencoders. In *NIPS Deep Learning Workshop*, 2014.

Masci, Jonathan, Meier, Ueli, Cireşan, Dan, and Schmidhuber, Jürgen. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Artificial Neural Networks and Machine Learning–ICANN 2011*, pp. 52–59. Springer, 2011.

Mnih, Andriy and Gregor, Karol. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*, 2014.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, pp. 807–814, 2010.

Neal, Radford M. Connectionist learning of belief networks. *Artificial intelligence*, 56(1):71–113, 1992.

Rahimi, Ali and Recht, Benjamin. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

Ramdas, Aaditya, Reddi, Sashank J, Poczos, Barnabas, Singh, Aarti, and Wasserman, Larry. On the decreasing power of kernel and distance based nonparametric hypothesis tests in high dimensions. In *The Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, 2015.

Rezende, Danilo Jimenez, Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pp. 1278–1286, 2014.

Rifai, Salah, Vincent, Pascal, Muller, Xavier, Glorot, Xavier, and Bengio, Yoshua. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 833–840, 2011.

Rifai, Salah, Bengio, Yoshua, Dauphin, Yann, and Vincent, Pascal. A generative process for sampling contractive auto-encoders. In *International Conference on Machine Learning (ICML)*, 2012.

Salakhutdinov, Ruslan and Hinton, Geoffrey E. Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, 2009.

Sermanet, Pierre, Eigen, David, Zhang, Xiang, Mathieu, Michaël, Fergus, Rob, and LeCun, Yann. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations*, 2014.

Snoek, Jasper, Larochelle, Hugo, and Adams, Ryan P. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, 2012.

Snoek, Jasper, Swersky, Kevin, Zemel, Richard S., and Adams, Ryan P. Input warping for bayesian optimization of non-stationary functions. In *International Conference on Machine Learning*, 2014.

Sriperumbudur, Bharath K, Fukumizu, Kenji, Gretton, Arthur, Lanckriet, Gert RG, and Schölkopf, Bernhard. Kernel choice and classifiability for rkhs embeddings of probability distributions. In *Advances in Neural Information Processing Systems*, pp. 1750–1758, 2009.

Susskind, Joshua, Anderson, Adam, and Hinton, Geoffrey E. The toronto face dataset. Technical report, Department of Computer Science, University of Toronto, 2010.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc VV. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pp. 3104–3112, 2014.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

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

Vinyals, Oriol, Toshev, Alexander, Bengio, Samy, and Erhan, Dumitru. Show and tell: A neural image caption generator. *arXiv preprint arXiv:1411.4555*, 2014.

Zeiler, Matthew D, Krishnan, Dilip, Taylor, Graham W, and Fergus, Robert. Deconvolutional networks. In *Computer Vision and Pattern Recognition*, pp. 2528–2535. IEEE, 2010.

Zemel, Richard, Wu, Yu, Swersky, Kevin, Pitassi, Toni, and Dwork, Cynthia. Learning fair representations. In *International Conference on Machine Learning*, pp. 325–333, 2013.

Zhao, Ji and Meng, Deyu. Fastmmd: Ensemble of circular discrepancy for efficient two-sample test. *arXiv preprint arXiv:1405.2664*, 2014.