Generative Moment Matching Networks

Yujia Li, Kevin Swersky and Richard Zemel

Feb.12, 2015

- We want to learn generative models
  - Generate nice samples
  - Find structure in data
  - Extract features for other tasks like classification
  - Make use of unlabeled data

- Generative models in deep learning
  - Undirected models
    - Boltzmann machines, RBMs, DBMs
  - Directed models
    - Neural Autoregressive Distribution Estimator (NADE)
    - Sigmoid belief nets
    - Deep belief nets (hybrid)
    - Recent advances in using neural nets to do inference for these models
  - Auto-Encoders used as generative models
    - Methods for recovering density models from auto-encoders

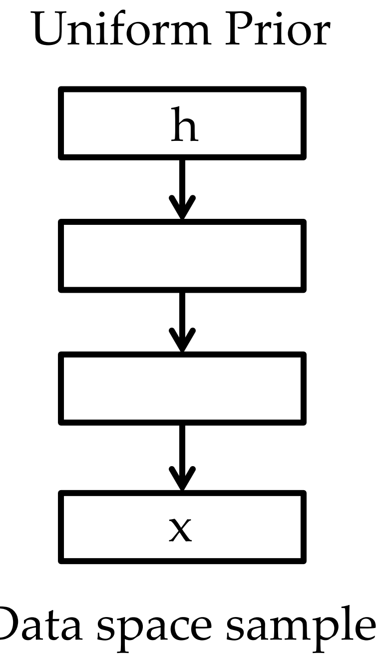
- The model we consider here:
  - Uniform prior

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

  - Deterministic mapping defined by a neural net

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

  - p(h) and x=*f*(h; w) jointly define a distribution of x



- Very easy to generate samples
  - h ~ p(h), then pass h through the net to get x
  - Not easy to estimate probability

- Similar models studied in (Mackay, 1995) and (Magdon-Ismail and Atidya, 1998)

- Recently used in generative adversarial nets (Goodfellow et al., 2014)
  - Training formulated as minimax optimization
  - Alternating optimization

- We train them with a simple objective MMD
  - Can be interpreted as moment matching
  - Trainable by direct backpropagation
  - "Generative moment matching networks" (GMMN)

Moment Matching

- Moments:
  - Mean (1^{st} order), variance (2^{nd} order), skewness (3^{rd} order), …
- Under mild conditions: if all moments of two distributions p and q are the same, then p = q.
- Training generative models with moment matching:
  - Make model moments match data moments

- Define mapping function ϕ
  - Then $\frac{1}{N}\sum_{i=1}^{N}\phi(\mathbf{x}_i)$ is the sample moment
  - Examples:

$$\phi(\mathbf{x}) = \mathbf{x}$$ 1st order moment

$$\phi(\mathbf{x}) = \text{vec}(\mathbf{x}, \mathbf{x}\mathbf{x}^\top)$$ 1st and 2nd order moments

$$= (x_1, ..., x_d, x_1x_1, x_1x_2, ..., x_dx_d)$$

- Moment matching objective:
  - Data set $\{\mathbf{x}_1^d, ..., \mathbf{x}_N^d\}$, model samples $\{\mathbf{x}_1^s, ..., \mathbf{x}_M^s\}$

$$\left\| \frac{1}{M}\sum_{i=1}^{M}\phi(\mathbf{x}_i^s) - \frac{1}{N}\sum_{j=1}^{N}\phi(\mathbf{x}_j^d) \right\|^2$$

- Problem with this approach
  - There are infinite many moments
  - High order moments require exponentially many terms in ϕ: n-th order moments contain d^n terms

- Kernel trick
  - ϕ may contain infinite many terms, but we don't need to write them down

$$\left\| \frac{1}{M} \sum_{i=1}^{M} \phi(\mathbf{x}_i^s) - \frac{1}{N} \sum_{j=1}^{N} \phi(\mathbf{x}_j^d) \right\|^2$$

$$= \frac{1}{M^2} \sum_{i=1}^{M} \sum_{j=1}^{M} \phi(\mathbf{x}_i^s)^\top \phi(\mathbf{x}_j^s) - \frac{2}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \phi(\mathbf{x}_i^s)^\top \phi(\mathbf{x}_j^d) + \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i=1}^{N} \phi(\mathbf{x}_i^d)^\top \phi(\mathbf{x}_j^d)$$

$$= \frac{1}{M^2} \sum_{i=1}^{M} \sum_{j=1}^{M} k(\mathbf{x}_i^s, \mathbf{x}_j^s) - \frac{2}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} k(\mathbf{x}_i^s, \mathbf{x}_j^d) + \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i=1}^{N} k(\mathbf{x}_i^d, \mathbf{x}_j^d)$$

- For a universal kernel like Gaussian

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{2\sigma}\|\mathbf{x} - \mathbf{y}\|^2\right)$$

  - Using Taylor expansion, the implicit feature map $\phi(\mathrm{x})$ contains terms of all orders (weighted differently)
  - Polynomial kernels are not universal
    - $\phi(\mathrm{x})$ only contains terms up to the order of the kernel
    - But this is still a much more compact representation than writing out $\phi(\mathrm{x})$ explicitly.

$$\mathcal{L}_{\mathrm{MMD}^2} = \frac{1}{M^2}\sum_{i=1}^{M}\sum_{j=1}^{M} k(\mathbf{x}_i^s, \mathbf{x}_j^s) - \frac{2}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} k(\mathbf{x}_i^s, \mathbf{x}_j^d) + \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i=1}^{N} k(\mathbf{x}_i^d, \mathbf{x}_j^d)$$

- This is an empirical estimate of the kernel Maximum Mean Discrepancy (MMD) between data and model distributions

An Alternative Interpretation of MMD

- MMD originally came from the hypothesis testing literature
  - Suppose we have access to only samples from two distributions $X \sim P_A$ and $Y \sim P_B$
  - Can we tell if $P_A = P_B$?
  - Two-sample test problem

- Theorem: p and q are probability measures, then p = q iff

$$\max_{f \in \mathcal{F}} |\mathbb{E}_p[f(x)] - \mathbb{E}_q[f(x)]| = 0$$

  $\mathcal{F}$ is the class of bounded continuous functions.

  – If p ≠ q, then we can always construct some *f* that picks up the difference between p and q.
  – $\mathrm{MMD} = \max_{f \in \mathcal{F}} |\mathbb{E}_p[f(x)] - \mathbb{E}_q[f(x)]|$
  – If MMD is small, we say p = q, otherwise p ≠ q
  – Problem: $\mathcal{F}$ is a huge class

- (Gretton et al., 2007) showed that $\mathcal{F}$ can be just a RKHS associated with a universal kernel *k* and we can use

$$\mathrm{MMD}^2 \triangleq \|\mathbb{E}_p[\phi(x)] - \mathbb{E}_q[\phi(x)]\|^2$$

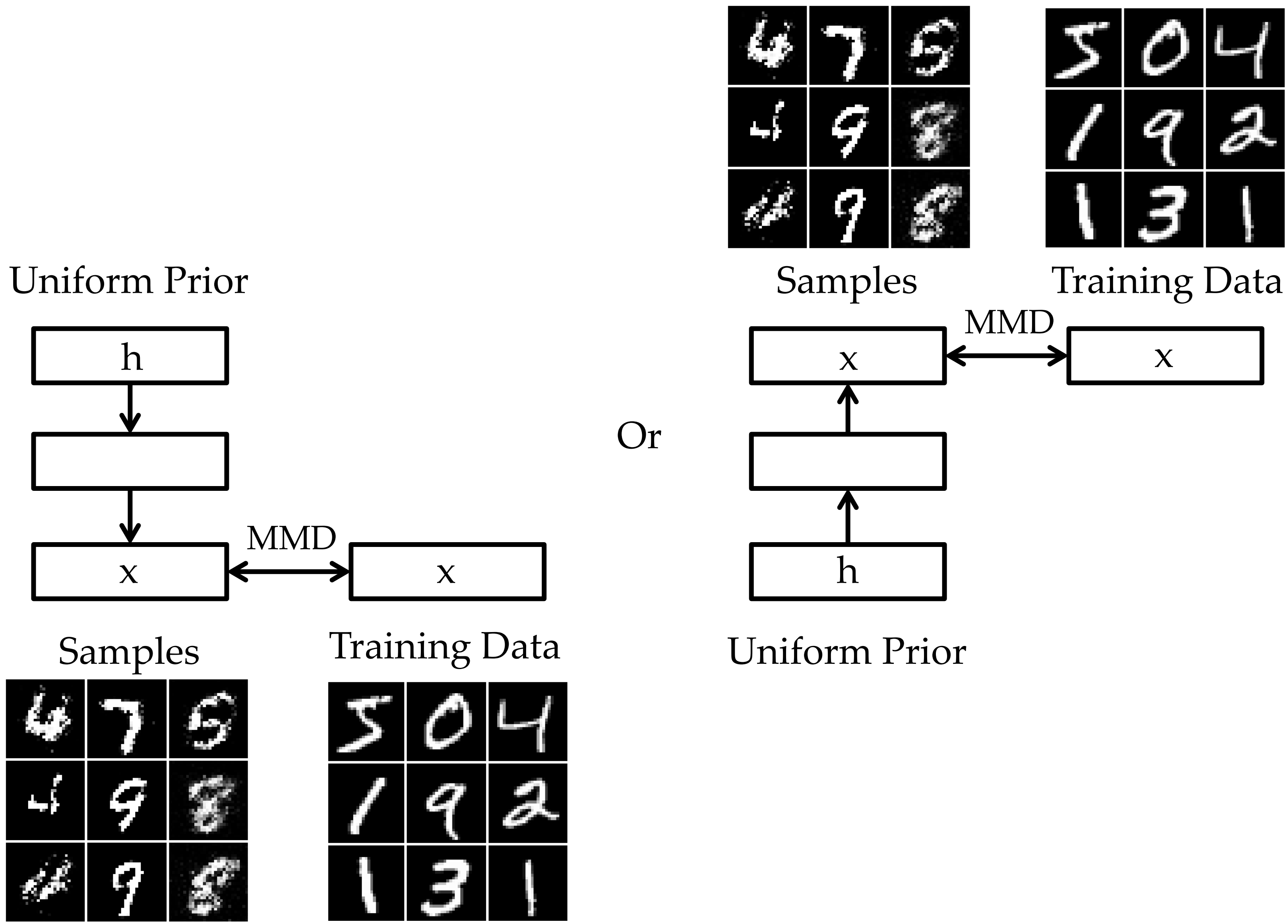
  - $\phi(\mathrm{x})$ is the kernel feature map for *k*
  - p = q iff $\mathrm{MMD}^2 = 0$

- An empirical estimate is

$$\mathrm{MMD}^2 = \frac{1}{M^2}\sum_{i=1}^{M}\sum_{j=1}^{M} k(\mathbf{x}_i^p, \mathbf{x}_j^p) - \frac{2}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} k(\mathbf{x}_i^p, \mathbf{x}_j^q) + \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i=1}^{N} k(\mathbf{x}_i^q, \mathbf{x}_j^q)$$

Using MMD to Learn GMMNs

- It's simple!
  - Generate a set of $\{h_1, \ldots, h_M\}$ from uniform prior $p(h)$
  - Compute corresponding $X^s = \{x_1, \ldots, x_M\}$ by $x = f(h; w)$
  - Use MMD between X^s and training set X^d as a loss function, backprop through the MMD loss and the neural net f to update w


Uniform Prior
h
x
MMD
x
Samples
Training Data
Or
Samples
Training Data
MMD
x
x
h
Uniform Prior

GMMN in Auto-Encoder Code Space

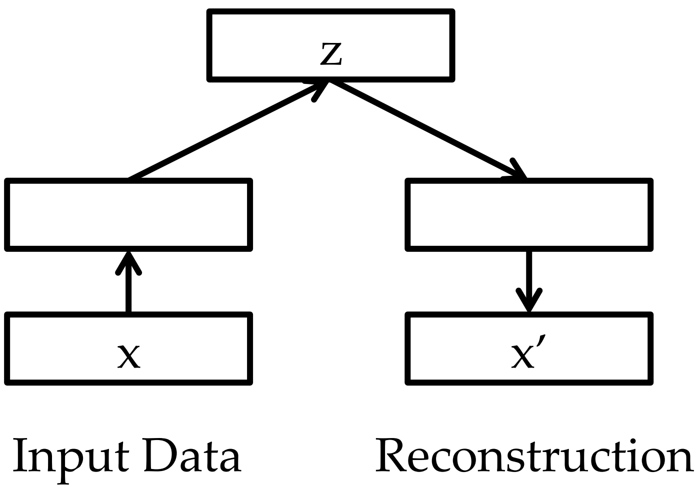
- Auto-Encoders
  - Easier to train
  - Good at recovering a low-dimensional manifold in high-dimensional space
  - Disentangle factors of variations (Bengio et al., 2013)
  - If we transform the distribution in the original input space to a distribution in the code space then it looks much nicer!

- Code space also helps MMD – as MMD is better in lower-dimensional spaces (Ramdas et al., 2015)

Training GMMN+AE

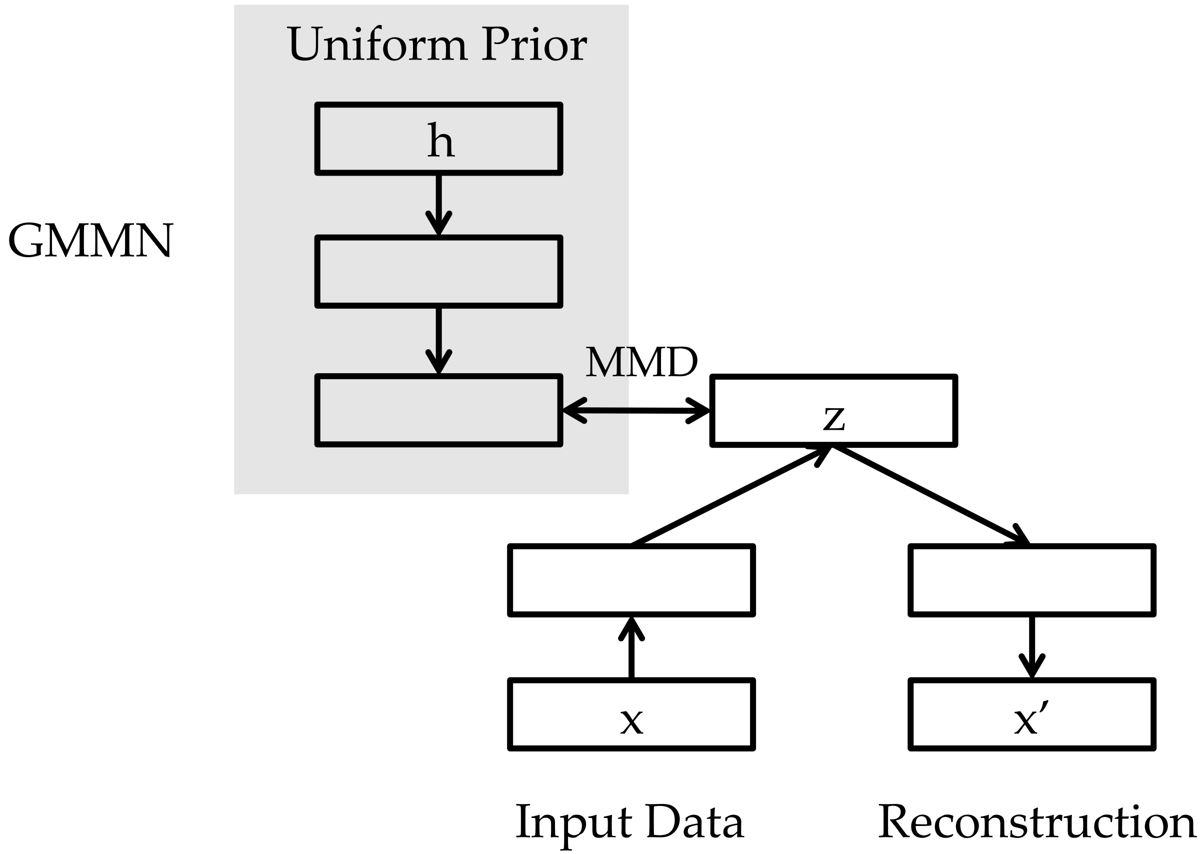
Trained with layer-wise pretraining + fine-tuning,

Dropout on encoder layers during training

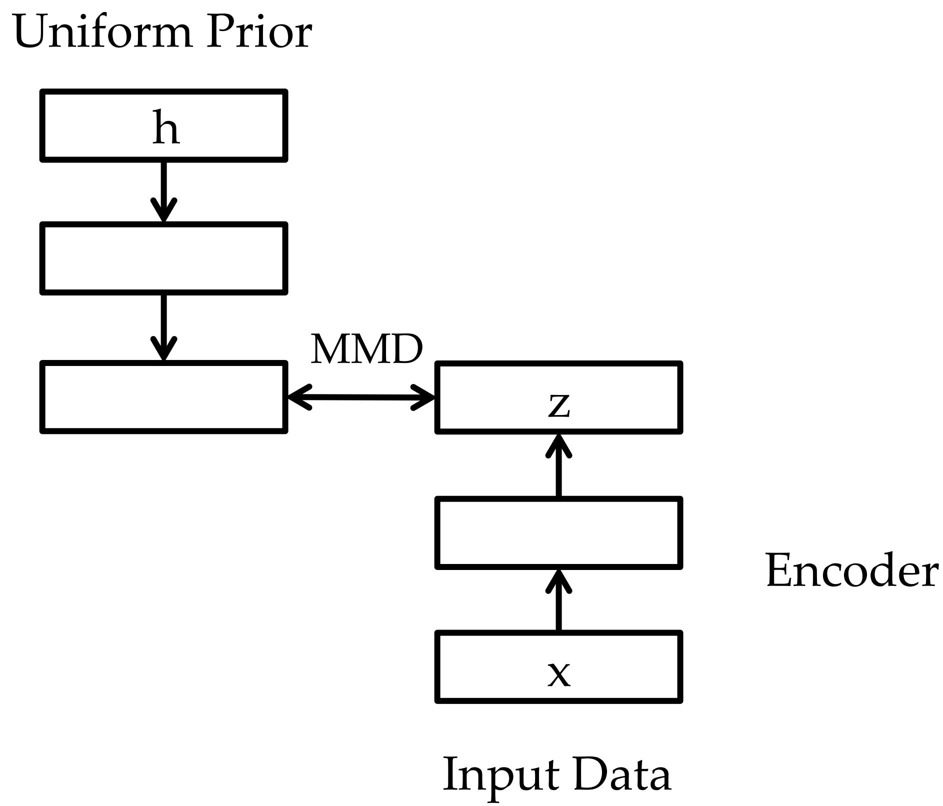


Auto-Encoder

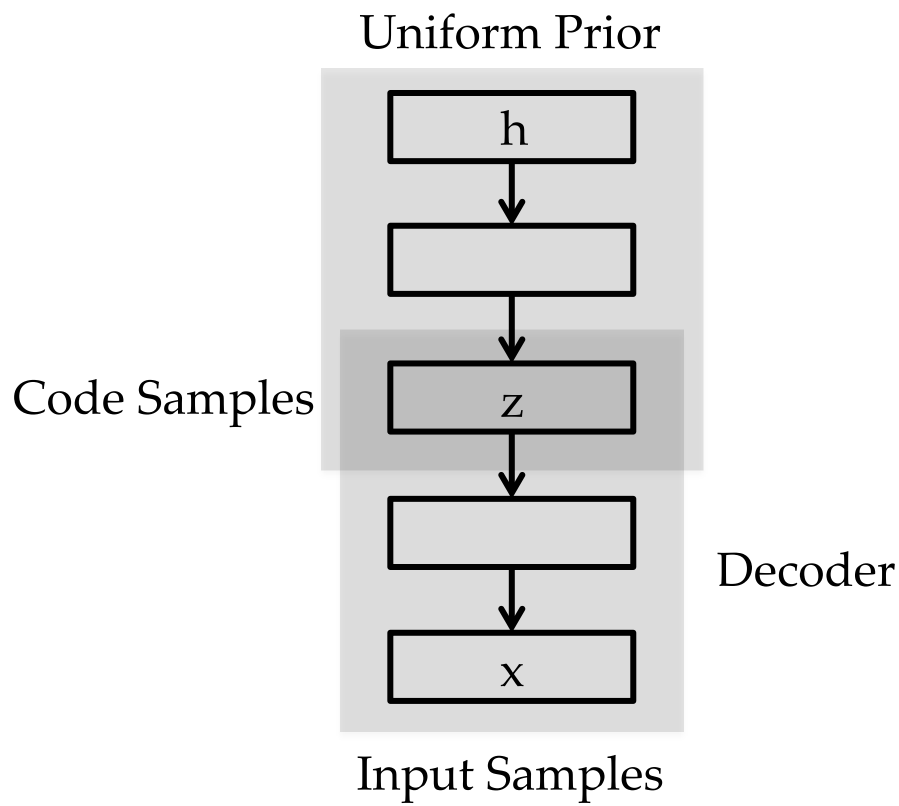
Training GMMN+AE



Training GMMN+AE



Generating Samples



Practical Considerations

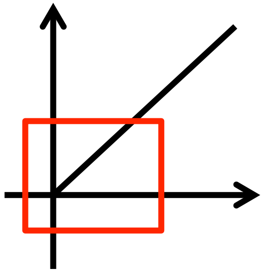
- Bandwidth parameter σ in the kernel
  - We can treat them as hyperparameters
  - Or use heuristics to set them
  - For most cases we used multiple kernels with fixed σ

$$k(x, y) = \sum_i k_{\sigma_i}(x, y)$$

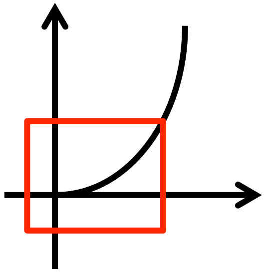
  - For example fix $\sigma_i = 1, 2, 5, 10, \ldots$
  - Matching distributions at multiple scales
  - Covers the range of possible σ

- Square root loss

$$\mathcal{L}_{\mathrm{MMD}} = \sqrt{\mathcal{L}_{\mathrm{MMD}^2}}$$



v.s.



  - Square root loss helps to drive the loss to zero
  - Much larger gradients when close to 0

$$\frac{\partial \mathcal{L}_{\mathrm{MMD}}}{\partial \mathbf{w}} = \frac{1}{2\sqrt{\mathcal{L}_{\mathrm{MMD}^2}}} \frac{\partial \mathcal{L}_{\mathrm{MMD}^2}}{\partial \mathbf{w}}$$

  - Easy to implement, simply scale the learning rate

- Minibatch training
  - MMD requires $O(N^2)$ computation
  - Linear time MMD variants available
  - We can also use random features to get linear time approximations
  - But for all our experiments we simply did minibatch training.

while *Stopping criterion not met* **do**

  Get a minibatch of data $\mathbf{X}^d \leftarrow \{\mathbf{x}^d_{i_1}, ..., \mathbf{x}^d_{i_b}\}$

  Get a new set of samples $\mathbf{X}^s \leftarrow \{\mathbf{x}^s_1, ..., \mathbf{x}^s_b\}$

  Compute gradient $\frac{\partial \mathcal{L}_{\mathrm{MMD}}}{\partial \mathbf{w}}$ on $\mathbf{X}^d$ and $\mathbf{X}^s$

  Take a gradient step to update $\mathbf{w}$

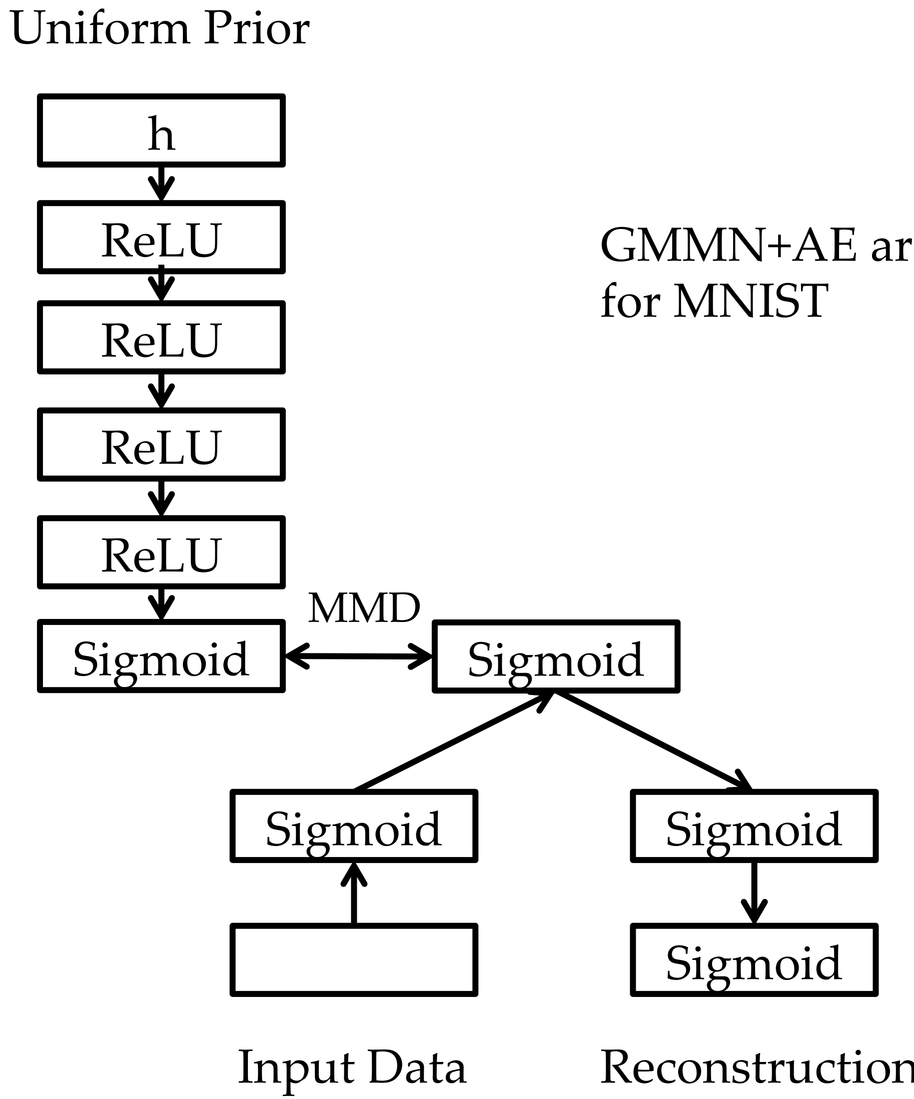
end

Experiments

- Datasets
  - MNIST: 60,000 training images (55,000 train, 5,000 validation), 10,000 test images, 32x32 (standard)
  - Toronto Face Dataset (TFD): ~100k images, 48x48, same training/test sets as in (Goodfellow et al., 2014)
  - Preprocessing: scale input image to [0,1]

- GMMN & GMMN+AE Architectures
  - GMMN has 5 layers, 4 intermediate ReLU layers and 1 sigmoid output layer – same across all experiments
  - MNIST AE: 2 encoder layers, 2 decoder layers, all sigmoid
  - TFD AE: 3 encoder layers, 3 decoder layers, all sigmoid

GMMN+AE architecture for MNIST



- Evaluation
  - Computing likelihood is hard
  - Generating samples is easy, so
    - We generated 10,000 samples from the model
    - Use kernel density estimator to estimate the density
    - Compute log-likelihood of data under this estimated density
  - Same protocol used in previous work like (Goodfellow et al., 2014)

- Results

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	$\mathbf{282 \pm 2}$	$\mathbf{2204 \pm 20}$

  - DBN and Stacked CAE from (Bengio et al., 2013)
  - Deep GSN from (Bengio et al., 2014)
  - Adversarial nets from (Goodfellow et al., 2014)

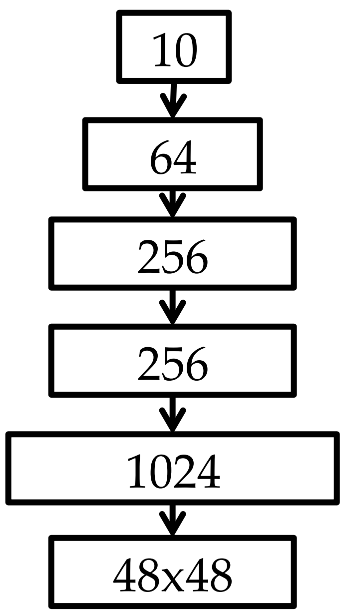
- Significant step forward over baselines
- GMMN+AE much better than GMMN

- Power of Bayesian Optimization
  - Number of hidden units, learning rate, momentum, dropout rate optimized on validation set using BO.
  - Switched from manual tuning to Bayesian Optimization a week before ICML deadline

	MNIST		TFD	
	GMMN	GMMN+AE	GMMN	GMMN+AE
2 weeks ago	~135	~270	1900~2000	~2100
Last week	147	282	**2085**	**2204**

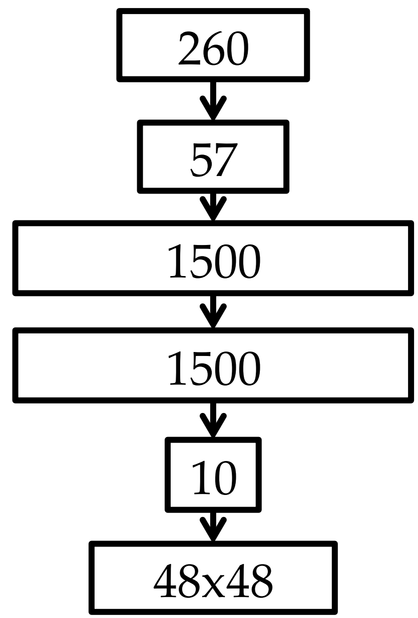
- Surprising architectures
  - Example: GMMN on TFD

Uniform Prior



Manually tuned: 1900~2000

Uniform Prior



Bayesian Optimization: 2085

- Not so surprising settings:
  - Auto-Encoder code space dimensionality much smaller than data dimensionality
  - Large dropout for the encoder

- Samples

(a) GMMN MNIST samples

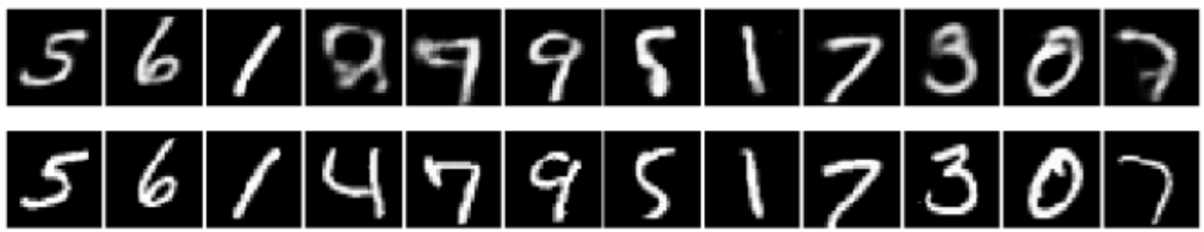
(b) GMMN TFD samples

(c) GMMN+AE MNIST samples

(d) GMMN+AE TFD samples

- Closest training examples to generated samples

(e) GMMN nearest neighbors for MNIST samples



(f) GMMN+AE nearest neighbors for MNIST samples

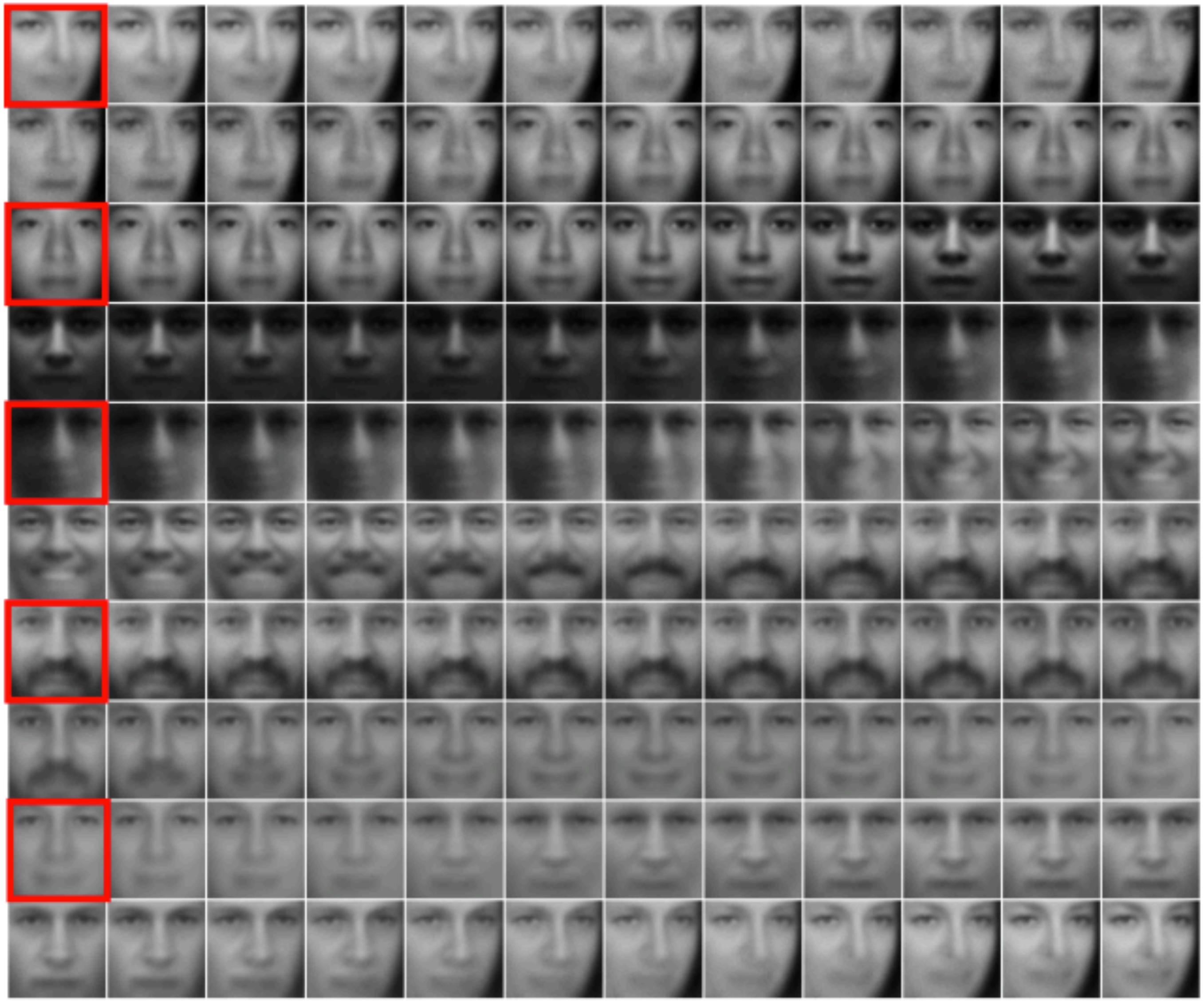
- Closest training examples to generated samples

(g) GMMN nearest neighbors for TFD samples

(h) GMMN+AE nearest neighbors for TFD samples

- Exploring the learned space

- Exploring the learned space



- Videos

How We Started to Work on This

- Fairness
- Domain adaptation
- Learning invariant features
- Learning features robust to noise

Future Directions

- Generate larger, more realistic images
- Generate image labels like segmentation masks
- Conditional generation

Take-Aways

- MMD offers a much simpler objective for training this type of networks
- Auto-Encoders can be readily bootstrapped into part of a good generative model

Q & A

Generative Moment Matching Networks

Yujia Li, Kevin Swersky and Richard Zemel

Feb.12, 2015