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
    \[ p(h) = \prod_{j=1}^{H} U(h_j) \]
  – Deterministic mapping defined by a neural net
    \[ x = f(h; w) \]
  – p(h) and x=f(h; w) jointly define a distribution of x

• Very easy to generate samples
  – h \sim 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\textsuperscript{st} order), variance (2\textsuperscript{nd} order), skewness (3\textsuperscript{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 $\phi$
  
  – Then $\frac{1}{N} \sum_{i=1}^{N} \phi(x_i)$ is the sample moment

  – Examples:
    
    \[
    \phi(x) = x \quad \text{1st order moment}
    \]
    
    \[
    \phi(x) = \text{vec}(x, xx^\top) \quad \text{1st and 2nd order moments}
    \]
    
    \[
    = (x_1, \ldots, x_d, x_1 x_1, x_1 x_2, \ldots, x_d x_d)
    \]

• Moment matching objective:
  
  – Data set $\{x_1^d, \ldots, x_N^d\}$, model samples $\{x_1^s, \ldots, x_M^s\}$

  \[
  \left\| \frac{1}{M} \sum_{i=1}^{M} \phi(x_i^s) - \frac{1}{N} \sum_{j=1}^{N} \phi(x_j^d) \right\|^2
  \]
• Problem with this approach
  – There are infinite many moments
  – High order moments require exponentially many terms in $\phi$: $n$-th order moments contain $d^n$ terms

• Kernel trick
  – $\phi$ may contain infinite many terms, but we don’t need to write them down

$$
\left\| \frac{1}{M} \sum_{i=1}^{M} \phi(x^s_i) - \frac{1}{N} \sum_{j=1}^{N} \phi(x^d_j) \right\|^2
= \frac{1}{M^2} \sum_{i=1}^{M} \sum_{j=1}^{M} \phi(x^s_i)^\top \phi(x^s_j) - \frac{2}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \phi(x^s_i)^\top \phi(x^d_j) + \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \phi(x^d_i)^\top \phi(x^d_j)
$$

$$
= \frac{1}{M^2} \sum_{i=1}^{M} \sum_{j=1}^{M} k(x^s_i, x^s_j) - \frac{2}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} k(x^s_i, x^d_j) + \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} k(x^d_i, x^d_j)
$$
• For a universal kernel like Gaussian

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

– Using Taylor expansion, the implicit feature map \( \phi(x) \)
contains terms of all orders (weighted differently)

– Polynomial kernels are not universal
  • \( \phi(x) \) only contains terms up to the order of the kernel
  • But this is still a much more compact representation than writing out \( \phi(x) \) explicitly.
\[ \mathcal{L}_{\text{MMD}}^2 = \frac{1}{M^2} \sum_{i=1}^{M} \sum_{j=1}^{M} k(x_i^s, x_j^s) - \frac{2}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} k(x_i^s, x_j^d) + \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} k(x_i^d, x_j^d) \]

- 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: If 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 \( p \neq q \), then we can always construct some \( f \) that picks up the difference between \( p \) and \( q \).

– 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 \( p \neq q \)

– 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

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

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

• An empirical estimate is

$$
\text{MMD}^2 = \frac{1}{M^2} \sum_{i=1}^{M} \sum_{j=1}^{M} k(x^p_i, x^p_j) - \frac{2}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} k(x^p_i, x^q_j) + \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} k(x^q_i, x^q_j)
$$
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

Samples

Training Data

MMD

Samples

Training Data

Or

MMD

Uniform Prior

h

x

x

x
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

Input Data | Reconstruction

Auto-Encoder
Training GMMN+AE

GMMN

Uniform Prior

\[ h \]

\[ \text{MMD} \]

\[ z \]

\[ x \]

\[ x' \]

Input Data

Reconstruction
Training GMMN+AE

Uniform Prior

Encoder

MMD

Input Data
Generating Samples

Uniform Prior

Code Samples

Decoder

Input Samples
Practical Considerations

• Bandwidth parameter $\sigma$ in the kernel
  – We can treat them as hyperparameters
  – Or use heuristics to set them
  – For most cases we used multiple kernels with fixed $\sigma$
    $$k(x, y) = \sum_i k_{\sigma_i}(x, y)$$
  – For example fix $\sigma_i = 1, 2, 5, 10, \ldots$
  – Matching distributions at multiple scales
  – Covers the range of possible $\sigma$
- **Square root loss**

\[
L_{\text{MMD}} = \sqrt{L_{\text{MMD}}^2}
\]

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

\[
\frac{\partial L_{\text{MMD}}}{\partial w} = \frac{1}{2\sqrt{L_{\text{MMD}}^2}} \frac{\partial L_{\text{MMD}}^2}{\partial w}
\]

- 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.

```
1   while Stopping criterion not met do
2      Get a minibatch of data $X^d \leftarrow \{x_{i_1}^d, \ldots, x_{i_b}^d\}$
3      Get a new set of samples $X^s \leftarrow \{x_1^s, \ldots, x_b^s\}$
4      Compute gradient $\frac{\partial L_{MMD}}{\partial w}$ on $X^d$ and $X^s$
5      Take a gradient step to update $w$
6   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
Uniform Prior

GMMN+AE architecture for MNIST

Sigmoid

ReLU

ReLU

ReLU

ReLU

Sigmoid

MMD

Input Data

Reconstruction

Sigmoid

Sigmoid

Sigmoid

Sigmoid
• 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

<table>
<thead>
<tr>
<th>Model</th>
<th>MNIST</th>
<th>TFD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBN</td>
<td>138 ± 2</td>
<td>1909 ± 66</td>
</tr>
<tr>
<td>Stacked CAE</td>
<td>121 ± 1.6</td>
<td>2110 ± 50</td>
</tr>
<tr>
<td>Deep GSN</td>
<td>214 ± 1.1</td>
<td>1890 ± 29</td>
</tr>
<tr>
<td>Adversarial nets</td>
<td>225 ± 2</td>
<td>2057 ± 26</td>
</tr>
<tr>
<td>GMMN</td>
<td>147 ± 2</td>
<td>2085 ± 25</td>
</tr>
<tr>
<td>GMMN+AE</td>
<td>282 ± 2</td>
<td>2204 ± 20</td>
</tr>
</tbody>
</table>

• 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

<table>
<thead>
<tr>
<th></th>
<th>MNIST</th>
<th>TFD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GMMN</td>
<td>GMMN+AE</td>
</tr>
<tr>
<td>2 weeks ago</td>
<td>~135</td>
<td>~270</td>
</tr>
<tr>
<td>Last week</td>
<td>147</td>
<td>282</td>
</tr>
</tbody>
</table>
• Surprising architectures
  – Example: GMMN on TFD

Uniform Prior:
- 10
- 64
- 256
- 256
- 1024
- 48x48

Manually tuned: 1900~2000

Uniform Prior:
- 260
- 57
- 1500
- 1500
- 10
- 48x48

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