Deep Generative Models and a Probabilistic Programming Library

Jun Zhu
dcszj@mail.tsinghua.edu.cn
Department of Computer Science and Technology
Tsinghua University

VALSE, Dalian, 2018
Discriminative (Deep) Learning

Learn a (differentiable) function mapping from input to output

\[ x \rightarrow f(x; \theta) \rightarrow y \]

- Gradient back-propagation
Generative Modeling

- Have training examples
  \[ x \sim p_{\text{data}}(x) \]

- Want a model that can draw samples:
  \[ x' \sim p_{\text{model}}(x) \]
  where \( p_{\text{model}}(x) \approx p_{\text{data}}(x) \)

\[ x \sim p_{\text{data}}(x) \quad \xrightarrow{\text{model}} \quad x' \sim p_{\text{model}}(x) \]
Why generative models?

- Leverage unlabeled datasets, which are often much larger than labeled ones
  - Unsupervised learning
    - e.g., clustering, density estimation, feature extraction, dimension reduction, data generation
  - Semi-supervised learning
    - e.g., classification, information extraction, learning-to-rank, network analysis, opinion mining

- Conditional generative models
  - Speech synthesis: Text ⇒ Speech
  - Machine Translation: French ⇒ English
  - Image captioning: Image ⇒ Text
Generative models are everywhere …

- (hierarchical) Language models for text
- Gaussian mixture models for clustering/density estimation
- Probabilistic PCA/FA/ICA for dimension reduction
- Probabilistic matrix factorization (PMF) for recommendation
- Deep belief networks (Hinton et al., 2016)
Two ways to build deep generative models

- Traditional one
  - Hierarchical Bayesian methods

- More modern one
  - Deep generative models
Hierarchical Bayesian Modeling

- Build a hierarchy through distributions in analytical forms


**Simple, Local Factors**: a conditional probability distribution
Deep Generative Models

- More flexible by using differential function mapping between random variables

- If \( z \) is uniformly distributed over \((0, 1)\), then \( y = f(z) \) has the distribution

\[
p(y) = p(z) \left| \frac{dz}{dy} \right|
\]

- where \( p(z) = 1 \)

- This trick is widely used to draw samples from exponential family distributions (e.g., Gaussian, Exponential)
Deep Generative Models

- More flexible by using differential function mapping between random variables

- DGMs learn a function transform with deep neural networks

![Diagram showing a function transform with deep neural networks](image)

- **Z**: Cause = Disease
- **X**: Topics = Docs, Objects = Images, Words = Phonemes

\[ f_{DNN} \]
An example with MLP

- 1D latent variable $z$; 2D observation $x$
- **Idea**: NN + Gaussian (or Bernoulli) with a diagonal covariance

\[
\begin{align*}
\mu_1, \sigma_1^2, \mu_2, \sigma_2^2 & \\
\mathbf{z} & \sim \mathcal{N}(0, 1) \\
\mathbf{x}_i | \mathbf{z} & \sim \mathcal{N}(\mu_i, \sigma_i^2)
\end{align*}
\]
Implicit Deep Generative Models

Generate data with a stochastic process whose likelihood function is not explicitly specified (Hartig et al., 2011)
Deep Generative Models

[Image Generation: Generative Adversarial Nets, Goodfellow13 & Radford15]
Learning Deep Generative Models

Given a set \( D \) of unlabeled samples, learn the unknown parameters (or a distribution)

\[
\text{data } D = \{x_i\}
\]

\[
\text{models } p(x|\theta)
\]

Find a model that minimizes

\[
\mathbb{D}(\text{data } \{x_i\}_{i=1}^n, \text{model } p)
\]
Learning Deep Generative Models

- Maximum likelihood estimation (MLE):
  \[ \hat{\theta} = \arg \max p(D|\theta) \]
  - has an explicit likelihood model

- Minimax objective (e.g., GAN)
  - a two-player game to reach equilibrium

- Moment-matching:
  - draw samples from \( p \):
    \[ \tilde{D} = \{y_i\}_{i=1}^M, \text{where } y_i \sim p(x|\theta) \]
  - Kernel MMD:
    \[ L_{MMD^2} = \left\| \frac{1}{N} \sum_{i=1}^{N} \phi(x_i) - \frac{1}{M} \sum_{j=1}^{M} \phi(y_i) \right\|_{H}^2 \]
    - rich enough to distinguish any two distributions in certain RKHS
Related Work at VALSE 2018

- Duplex GAN for Domain Adaptation (DGM)
  - Hu, Kan, Shan, Chen

- Multimodal Possion Gamma Belief Networks (DGM)
  - Wang, Chen, Zhou

- WHAI: Weibull Hybrid Autoencoding Inference for Deep Topic Models (DGM)
  - Zhang, Chen, Guo, Zhou

- The Assumed Parameter Filter (probabilistic programming)
  - Erol, Wu, Li, Russell

- …
**Our work on DGMs**

- **Learning algorithms and theories**
  - Max-margin variational auto-encoder (Li et al., NIPS 2015)
  - Learning to generate with memory (Li et al., ICML 2016)
  - Conditional moment-matching generative networks (Ren et al., NIPS 2016)
  - Population matching discrepancy (Chen et al., NIPS 2017)
  - Implicit variational inference (Shi et al., ICLR 2018)

- **Semi-supervised learning & Style transfer**
  - Max-margin variational auto-encoder for SSL (Li et al., PAMI 2017)
  - Triple generative adversarial networks (Li et al., NIPS 2017)
  - Structured generative adversarial networks (Deng et al., NIPS 2017)
  - Learning to write styled Chinese characters by reading a handful of examples (Sun et al., IJCAI 2018)
  - Smooth neighbors for SSL (Luo et al., CVPR 2018)

- **Programming library**
  - ZhuSuan (Shi et al., arXiv 2017)
Outline

- Deep Generative Models
- Semi-supervised Learning & Style transfer
- ZhuSuan: a Probabilistic Programming library
- Conclusions & QA
Semi-supervised Learning

A toy example
Representation Matters

- t-SNE embedding of learned representations by different DGM models on CIFAR10
## Triple Generative Adversarial Nets

- A minimax game for semi-supervised learning
  - GAN is for unsupervised learning  \( p_{\text{model}}(x) = p_{\text{data}}(x) \)
  - We aim to learn the joint distribution  \( p_{\text{model}}(x, y) = p_{\text{data}}(x, y) \)

- A simple insight
  - factorization form with conditionals

\[
\begin{align*}
p(x, y) &= p(x)p(y|x) \\
&= p(y)p(x|y)
\end{align*}
\]

- We need three players
  - Two generators to generate \((x, y)\)
  - A discriminator to distinguish fake \((x, y)\)

[Li et al., NIPS 2017]
Triple-GAN

- The network architecture

- Both C and G are generators
- D is the discriminator
- CE: cross-entropy loss for learning classifier

[Li et al., NIPS 2017]
A minimax game

The optimization problem

\[
\min_{C,G} \max_D U(C, G, D) = E_p[\log D(x, y)] + \alpha E_{p_c}[\log(1 - D(x, y))] + (1 - \alpha) E_{p_g}[\log(1 - D(x, y))]
\]

- The hyper-parameter \( \alpha \) is often set at 1/2

- The standard supervised loss can be incorporated

\[
\min_{C,G} \max_D \tilde{U}(C, G, D) = U(C, G, D) + E_p[-\log p_c(y|x)]
\]

[Li et al., NIPS 2017]
Major theoretical results

**Theorem**

The equilibrium of $\tilde{U}(C, G, D)$ is achieved if and only if $p(x, y) = p_g(x, y) = p_c(x, y)$ with $D_{C,G}^*(x, y) = \frac{1}{2}$ and the optimum value is $-\log 4$.

**Lemma**

For any fixed $C$ and $G$, the optimal discriminator $D$ is:

$$D_{C,G}^*(x, y) = \frac{p(x, y)}{p(x, y) + p_\alpha(x, y)},$$

where $p_\alpha(x, y) := (1 - \alpha)p_g(x, y) + \alpha p_c(x, y)$. 
Some Practical Tricks for SSL

- Pseudo discriminative loss: using $(x, y) \sim p_g(x, y)$ as labeled data to train $C$
  - Explicit loss, equivalent to $KL(p_g(x, y) \| p_c(x, y))$
  - Complementary to the implicit regularization by $D$

- Collapsing to the empirical distribution $p(x, y)$
  - Sample $(x, y) \sim p_c(x, y)$ as true data for $D$
  - Biased solution: target shifting towards $p_c(x, y)$

- Unlabeled data loss on $C$
  - Confidence (Springenberg [2015])
  - Consistence (Laine and Aila [2016])
Some Results

Semi-supervised classification

Table 1: Error rates (%) on partially labeled MNIST, SHVN and CIFAR10 datasets. The results with † are trained with more than 500,000 extra unlabeled data on SVHN.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MNIST $n = 100$</th>
<th>SVHN $n = 1000$</th>
<th>CIFAR10 $n = 4000$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M1+M2$ [11]</td>
<td>3.33 (±0.14)</td>
<td>36.02 (±0.10)</td>
<td></td>
</tr>
<tr>
<td>VAT [18]</td>
<td>2.33</td>
<td></td>
<td>24.63</td>
</tr>
<tr>
<td>Ladder [23]</td>
<td>1.06 (±0.37)</td>
<td></td>
<td>20.40 (±0.47)</td>
</tr>
<tr>
<td>Conv-Ladder [23]</td>
<td><strong>0.89</strong> (±0.50)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADGM [17]</td>
<td>0.96 (±0.02)</td>
<td>22.86 †</td>
<td></td>
</tr>
<tr>
<td>SDGM [17]</td>
<td>1.32 (±0.07)</td>
<td>16.61 (±0.24)†</td>
<td></td>
</tr>
<tr>
<td>MMCVA [15]</td>
<td>1.24 (±0.54)</td>
<td><strong>4.95</strong> (±0.18)†</td>
<td></td>
</tr>
<tr>
<td>CatGAN [26]</td>
<td>1.39 (±0.28)</td>
<td></td>
<td>19.58 (±0.58)</td>
</tr>
<tr>
<td>Improved-GAN [25]</td>
<td>0.93 (±0.07)</td>
<td>8.11 (±1.3)</td>
<td>18.63 (±2.32)</td>
</tr>
<tr>
<td>ALI [5]</td>
<td></td>
<td>7.3</td>
<td>18.3</td>
</tr>
<tr>
<td>Triple-GAN (ours)</td>
<td><strong>0.91</strong> (±0.58)</td>
<td><strong>5.77</strong> (±0.17)</td>
<td><strong>16.99</strong> (±0.36)</td>
</tr>
</tbody>
</table>
Some Results

- Class-conditional generation

(d) Dog  (e) Horse  (f) Ship
Some Results

- Disentangle class and style

Figure: Same $y$ for each row. Same $z$ for each column.
Some Results

- Latent space interpolation on MNIST
Some Results

- Latent space interpolation on SVHN
Some Results

Latent space interpolation on CIFAR10
Structured GAN

- Structural extensions to Triple-GAN
  - Learn joint distribution $p(x, y, z)$
  - Ability to infer the posterior $p(y, z \mid x)$, $p(y \mid x)$, $p(z \mid x)$
  - Structural bias to disentangle latent factors $y$ and $z$ by information regularization on $p(y \mid x)$ and $p(x \mid x)$

[Deng et al., NIPS 2017]
Structured GAN

- Produces even better results than Triple-GAN

<table>
<thead>
<tr>
<th>Method</th>
<th>MNIST</th>
<th>SVHN</th>
<th>CIFAR-10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$n = 20$</td>
<td>$n = 50$</td>
<td>$n = 1000$</td>
</tr>
<tr>
<td>Ladder [22]</td>
<td>-</td>
<td>-</td>
<td>0.89 ± 0.50</td>
</tr>
<tr>
<td>VAE [12]</td>
<td>-</td>
<td>3.33 ± 0.14</td>
<td>36.02 ± 0.10</td>
</tr>
<tr>
<td>CatGAN [28]</td>
<td>-</td>
<td>1.39 ± 0.28</td>
<td>-</td>
</tr>
<tr>
<td>ALI [5]</td>
<td>-</td>
<td>-</td>
<td>7.3</td>
</tr>
<tr>
<td>ImprovedGAN [27]</td>
<td>16.77 ± 4.52</td>
<td>2.21 ± 1.36</td>
<td>8.11 ± 1.3</td>
</tr>
<tr>
<td>TripleGAN [15]</td>
<td>5.40 ± 6.53</td>
<td>1.59 ± 0.69</td>
<td>5.83 ± 0.20</td>
</tr>
<tr>
<td>SGAN</td>
<td>4.0 ± 4.14</td>
<td>1.29 ± 0.47</td>
<td>5.73 ± 0.12</td>
</tr>
</tbody>
</table>

- Inception score:
  - TripleGAN: 5.08 ± 0.09
  - Improved-GAN: 3.87 ± 0.03
Style Transfer for Chinese Characters

- Build a generative model for Chinese characters generation and writing style transfer tasks

One-shot Generalization  Disentanglement

[Sun et al., IJCAI 2018]
Style-Transfer VAE

Style Bank of Characters

Character Structure Knowledge

Convolution Network

Style Inference

Knowledge Embedding

Content Code

Inference

Convolution Network

Character Recognition

Style Feature

Deconvolution Network

Character Reconstruction

Generated Characters

Generation

Style Specified

Style Inference Network

Style Feature

Generation Network

Content Code

Character Recognition Network

Chinese Characters

Style Specified Characters
Structural Information

Content Code (Knowledge)

- Instead one-hot label to identify unique Chinese characters with human knowledge.
- Reuse the structure information sharing in all Chinese characters.

Pairwise Training:

我 你 他 → 我 你 他

Reconstruction
One-Shot Generation

Too simple characters can’t provide sufficient style information.
Low-Shot Generation

Sometimes low-shot can capture more detailed style information.
Style Generation

Interpolation
Outline

- Deep Generative Models
- Semi-supervised Learning
- ZhuSuan: a Probabilistic Programming library
- Conclusions & QA
Bayesian inference

\[ p(z|x) = \frac{p(x, z)}{p(x)} \]

Given Disease, what is the Cause?
Given Object, what are the Components?
Given Docs, what are the Topics?

Find Cause of Disease
Extract Topics from Docs
Identify Objects from Images
Recognize Words in Speeches
Inference in Old Days

Variational Inference
(Too much math!!!)

$$p(z|x) \quad q_\phi(z) \in \text{some family}$$

$$\min_{\phi} \text{KL}(q_\phi(z) \| p(z|x))$$

MCMC
(Many dynamics!!!)

Extremely painful for a single model!
Inference in ZhuSuan

Turn painful math deviations into Easy and Intuitive (Probabilistic) Programming

Computers are good at following instructions, but not at reading your mind.

~ Donald Knuth
ZhuSuan is a python library for generative models, built upon TensorFlow.

Unlike existing DL libraries, which are mainly for supervised tasks, ZhuSuan is featured for:

- its deep root into Bayesian Inference
- supporting various kinds of generative models: traditional hierarchical Bayesian models & recent deep generative models.
With ZhuSuan, users can enjoy

- powerful fitting and multi-GPU training of deep learning

- while at the same time they can use generative models to
  - model the complex world
  - exploit unlabeled data
  - deal with uncertainty by performing principled Bayesian inference
  - generate new samples
Model Primitives: BayesianNet

- A DAG representing a Bayesian Network
- Two types of nodes:
  - Deterministic nodes: Can be composed of any Tensorflow operations.
  - Stochastic nodes: Use StochasticTensor's from ZhuSuan's library.
- Start a BayesianNet environment

```python
import zhusuan as zs
with zs.BayesianNet() as model:
    # build the model
```
Example: Variational Autoencoders

\[ z \sim N(z|0, I) \]
\[ x_{logits} = f_{NN}(z) \]
\[ x \sim \text{Bernoulli}(x|\text{sigmoid}(x_{logits})) \]

```python
import tensorflow as tf
from tensorflow.contrib import layers
import zhusuan as zs

with zs.BayesianNet() as model:
    z_mean = tf.zeros([n, n_z])
    z_logstd = tf.zeros([n, n_z])
    z = zs.Normal('z', z_mean, z_logstd)
    h = layers.fully_connected(z, 500)
    x_logits = layers.fully_connected(h, n_x, activation_fn=None)
    x = zs.Bernoulli('x', x_logits)
```
With zs.BayesianNet() as variational:
    # build variational ...
qz_samples, log_qz = variational.query('z', outputs=True, local_log_prob=True)

lower_bound = zs.sgvb(log_joint, observed={'x': x},
                       latent={'z': [qz_samples, log_qz]})

optimizer = tf.train.AdamOptimizer(learning_rate=0.001)
run_op = optimizer.minimize(-lower_bound)
With tf.Session() as sess:
    for iter in range(iters):
        sess.run(run_op)
Example: Variational Autoencoders

Structured variational posterior as a BayesianNet ALSO.

```python
import tensorflow as tf
from tensorflow.contrib import layers
import zhusuan as zs

with zs.BayesianNet() as variational:
x = tf.placeholder([None, n_x], tf.float32)
h = layers.fully_connected(x, 500)
z_mean = layers.fully_connected(h, n_z, activation_fn=None)
z_logstd = layers.fully_connected(h, n_z, activation_fn=None)
z = zs.Normal('z', z_mean, z_logstd)
```

Variational Inference Algorithms

ZhuSuan supports a broad class of variational objectives, ranging from widely used evidence lower bounds to recent state-of-arts.

Works for continuous latent variables
- \texttt{zs.sgvb}: Stochastic gradient variational Bayes.
- \texttt{zs.iwae}: Importance weighted lower bounds.

Works for both continuous and discrete latent variables
- \texttt{zs.nvil}: Variance reduced score function estimator/REINFORCE.
- \texttt{zs.vimco}: Variance reduced multi-sample score function estimator.

* This is like optimization algorithms (SGD, momentum, Adam, etc.) in deep learning software. Users need not dive into the technical details of these algorithms because ZhuSuan provides easy-to-use APIs for users to directly try on their generative models.
# like creating the variable to optimize over.
z = tf.Variable(0.)

# like optimizer = tf.train.AdamOptimizer(…)
hmc = zs.HMC(step_size=1e-3, n_leapfrogs=10)

# like optimize_op = optimizer.minimize(…)
sample_op, hmc_info = hmc.sample(
    log_joint, observed={'x': x}, latent={'z': z})

with tf.Session() as sess:
    for iter in range(iters):
        # like sess.run(optimize_op)
        _ = sess.run(sample_op)

- Create the variable to store samples
- Initialize HMC
- Call sample() method to return a sample operation
- Run the sample operation like an optimizer!
Applications: ZhuSuan as a Research Platform

ZhuSuan is featured for both Bayesian Statistics and Deep Learning. State-of-the-Art models can be found in ZhuSuan’s examples.

- Bayesian Logistic Regression
- Bayesian Neural Nets for Multivariate Regression
- (Convolutional) Variational Autoencoders (VAE)
- Semi-supervised learning for images with VAEs
- Deep Sigmoid Belief Networks
- Generative Adversarial Networks (GAN)
- Gaussian processes (GPs)
- Topic Models
- More to come …
ZhuSuan: GitHub Page

Welcome to ZhuSuan

ZhuSuan is a python probabilistic programming library for advantages of Bayesian methods and deep learning. ZhuSu, which are mainly designed for deterministic neural network primitives and algorithms for building probabilistic models, algorithms include:

- Variational inference with programmable variational pc (SGVB, REINFORCE, VIMCO, etc.).
- Importance sampling for learning and evaluating models, with programmable proposals.
- Hamiltonian Monte Carlo (HMC) with parallel chains, and optional automatic parameter tuning.

Supported Inference

(Stochastic) Variational Inference (VI & SVI)

- Kinds of variational posteriors we support:
  - Mean-field posterior: Fully-factorized.
  - Structured posterior: With user specified dependencies.
- Variational objectives we support:
  - SGVB: Stochastic gradient variational Bayes
  - IWAIE: Importance weighted objectives
  - NVIL: Score function estimator with variance reduction
  - VIMCO: Multi-sample score function estimator with variance

Contributions & Stars
Welcome!

github.com/thu-ml/zhusuan
Summary

- Deep Generative models are powerful tools
  - hierarchical Bayesian models
  - deep generative models
  - semi-supervised learning & style transfer

- ZhuSuan provides a Python programming library
  - deep root into Bayesian inference
  - support deep generative models
DL still long way to go …


NIPS 2017 Adversarial Attack & Defense, first-places in all three tasks.
What VALUE taught me?

The 10th Asian Conference on Machine Learning, Beijing

Beijing, China

November 14 - 16, 2018

ACML 2018

Welcome to the 10th Asian Conference on Machine Learning (ACML 2018). The conference will take place on November 14 - 16, 2018 at Beijing Jiaotong University, Beijing, China. The conference aims to provide a leading international forum for researchers in machine learning and related fields to share their new ideas, progresses and achievements. Submissions from regions other than the Asia-Pacific are also highly encouraged.
Thanks!


**Online Documents:** [http://zhusuan.readthedocs.io/](http://zhusuan.readthedocs.io/)