Probabilistic Graphical Models

Inference & Learning in DL

Zhiting Hu

Lecture 19, March 29, 2017

Reading:
Deep Generative Models

- Explicit probabilistic models
  - Provide an explicit parametric specification of the distribution of $x$
  - Tractable likelihood function $p_\theta(x)$

- E.g.,

$$p(x, z | \alpha) = p(x | z)p(z | \alpha)$$
Deep Generative Models

- Explicit probabilistic models
  - Provide an explicit parametric specification of the distribution of $x$
  - Tractable likelihood function $p_\theta(x)$

- E.g., Sigmoid Belief Nets

$$
p(v_{kn} = 1|w_k, h_n^{(1)}, c_k) = \sigma(w_k^T h_n^{(1)} + c_k)
$$

$$
p(h_{in}^{(1)} = 1|w_i, h_n^{(2)}, c_i) = \sigma(w_i^T h_n^{(2)} + c_i)
$$

$h_n^{(2)} = \{0,1\}^J$

$h_n^{(1)} = \{0,1\}^I$

$v_n = \{0,1\}^K$
Deep Generative Models

- Explicit probabilistic models
  - Provide an explicit parametric specification of the distribution of $x$
  - Tractable likelihood function $p_\theta(x)$
  - E.g., Deep generative model parameterized with NNs (e.g., VAEs)

\[
p_\theta(x|z) = N(x; \mu_\theta(z), \sigma)
\]

\[
p(z) = N(z; 0, I)
\]
Deep Generative Models

- Implicit probabilistic models
  - Defines a stochastic process to simulate data $x$
  - Do not require tractable likelihood function
  - Data simulator
  - Natural approach for problems in population genetics, weather, ecology, etc.

- E.g., generate data from a deterministic equation given parameters and random noise (e.g., GANs)

\[
\begin{align*}
x_n &= g(z_n; \theta) \\
z_n &\sim N(0, I)
\end{align*}
\]
Recap: Variational Inference

- Consider a probabilistic model $p_\theta(x, z)$
- Assume variational distribution $q_\phi(z | x)$
- Lower bound for log likelihood

$$\log p(x) = KL \left( q_\phi(z | x) \| p_\theta(z | x) \right) + \int_z q_\phi(z | x) \log \frac{p_\theta(x, z)}{q_\phi(z | x)}$$

$$\geq \int_z q_\phi(z | x) \log \frac{p_\theta(x, z)}{q_\phi(z | x)}$$

$$\coloneqq \mathcal{L}(\theta, \phi; x)$$

- Free energy

$$F(\theta, \phi; x) = -\log p(x) + KL(q_\phi(z | x) \| p_\theta(z | x))$$
Wake Sleep Algorithm

- Consider a generative model $p_{\theta}(x|z)$
  - E.g., sigmoid brief nets
- Variational bound:
  $$\log p(x) \geq \int q_{\phi}(z|x) \log \frac{p_{\theta}(x,z)}{q_{\phi}(z|x)} := \mathcal{L}(\theta, \phi; x)$$
- Use a inference network $q_{\phi}(z|x)$
- Maximize the bound w.r.t. $p_{\theta} \rightarrow \text{Wake phase}$
  - $\max_{\theta} \mathbb{E}_{q(z|x)} [\log p_{\theta}(x|z)]$
  - Get samples from $q(z|x)$ through bottom-up pass
  - Use the samples as targets for updating the generator
Wake Sleep Algorithm

- [Hinton et al., Science 1995]
- Generally applicable to a wide range of generative models by training a separate inference network

- Consider a generative model $p_\theta(x|z)$, with prior $p(z)$
  - E.g., multi-layer brief nets

- Free energy
  $$F(\theta, \phi; x) = -\log p(x) + KL(q_\phi(z|x) \parallel p_\theta(z|x))$$

- Inference network $q_\phi(z|x)$
  - a.k.a. recognition network
Wake Sleep Algorithm

● Free energy:

\[ F(\theta, \phi; x) = -\log p(x) + KL(q_\phi(z|x) \| p_\theta(z|x)) \]

● Minimize the free energy w.r.t. \( p_\theta \) \( \rightarrow \) Wake phase
  
  ● \( \max_\theta E_{q(z|x)}[\log p_\theta(x|z)] \)
  
  ● Get samples from \( q \) through bottom-up pass on training data
  
  ● Use the samples as targets for updating the generator

[Figure courtesy: Maei’s slides]
Wake Sleep Algorithm

- Free energy:
  \[ F(\theta, \phi; x) = -\log p(x) + KL(q_\phi(z|x) \| p_\theta(z|x)) \]

- Maximize the free energy w.r.t. \( q_\phi(z|x) \)?
  - computationally expensive / high variance

- Instead, maximize w.r.t. \( q_\phi(z|x) \) \( \Rightarrow \) Sleep phase
  \[ F'(\theta, \phi; x) = -\log p(x) + KL(p(z|x) \| q_\phi(z|x)) \]

  - \( \max_\phi \mathbb{E}_{p(z,x)} \log q_\phi(z|x) \)
  - “Dreaming” up samples from \( p \) through top-down pass
  - Use the samples as targets for updating the recognition network
Wake Sleep Algorithm

- **Wake phase:**
  - Use recognition network to perform a bottom-up pass in order to create samples for layers above (from data)
  - Train generative network using samples obtained from recognition model

- **Sleep phase:**
  - Use generative weights to reconstruct data by performing a top-down pass
  - Train recognition weights using samples obtained from generative model

- **KL is not symmetric**
- **Doesn’t optimize a well-defined objective function**
- **Not guaranteed to converge**
Variational Auto-encoders (VAEs)

- [Kingma & Welling, 2014]
- Enjoy similar applicability with wake-sleep algorithm
  - Not applicable to discrete latent variables
- Optimize a variational lower bound on the log-likelihood
- Reduce variance through reparameterization of the recognition distribution
  - Alternatives: use control variates as in reinforcement learning [Mnih & Gregor, 2014]
Variational Auto-encoders (VAEs)

- Generative model $p_\theta(x|z)$, with prior $p(z)$
  - a.k.a. decoder
- Inference network $q_\phi(z|x)$
  - a.k.a. encoder, recognition network

Variational lower bound

$$\log p(x) \geq E_{q_\phi(z|x)}[\log p_\theta(x,z)] - KL\left(q_\phi(z|x) \parallel p(z)\right)$$

$$:= \mathcal{L}(\theta, \phi; x)$$
Variational Auto-encoders (VAEs)

- Variational lower bound

\[
\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x, z)] - \text{KL}(q_\phi(z|x) \| p(z))
\]

- Optimize \(\mathcal{L}(\theta, \phi; x)\) w.r.t. \(p_\theta(x|z)\)
  - the same with the wake phase

- Optimize \(\mathcal{L}(\theta, \phi; x)\) w.r.t. \(q_\phi(z|x)\)
  - Directly computing the gradient with MC estimation

\[
\nabla_\phi \mathbb{E}_{q_\phi(z)} [f(z)] = \mathbb{E}_{q_\phi(z)} [f(z) \nabla_{q_\phi(z)} \log q_\phi(z)] \approx \frac{1}{L} \sum_{l=1}^{L} f(z) \nabla q_{\phi}(z^{(l)}) \log q_{\phi}(z^{(l)})
\]

a REINFORCE-like update rule which suffers from high variance [Mnih & Gregor 2014]

(Next lecture for more on REINFORCE)

- VAEs use a reparameterization trick to reduce variance
VAEs: Reparameterization Trick

\[ z \sim q_\phi(z|x) \quad \Rightarrow \quad \epsilon \sim p(\epsilon) \]
\[ \text{deterministic variable} \quad z = g_\phi(\epsilon, x) \]

Example:
\[ z \sim p(z|x) = \mathcal{N}(\mu, \sigma^2) \quad \Rightarrow \quad \epsilon \sim \mathcal{N}(0, 1) \]
\[ z = \mu + \sigma \epsilon \]
VAEs: Reparameterization Trick

\[ q_\phi(z^{(i)}|x^{(i)}) = \mathcal{N}(z^{(i)}; \mu^{(i)}, \sigma^2(i)I) \]

\[ z^{(i,l)} = \mu^{(i)} + \sigma^{(i)} \odot \epsilon^{(l)} \]

\[ z = z_\phi(\epsilon) \text{ is a deterministic mapping of } \epsilon \]

[Figure courtesy: Chang’s slides]
VAEs: Reparameterization Trick

- Variational lower bound
  \[
  \mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_{\phi}(z|x)}[\log p_{\theta}(x, z)] - \text{KL}\left(q_{\phi}(z|x) \parallel p(z)\right)
  \]
  \[
  \mathbb{E}_{q_{\phi}(z|x)}[\log p_{\theta}(x, z)] = \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I)} \left[\log p_{\theta}(x, z_{\phi}(\epsilon))\right]
  \]

- Optimize \(\mathcal{L}(\theta, \phi; x)\) w.r.t. \(q_{\phi}(z|x)\)
  \[
  \nabla_{\phi} \mathbb{E}_{q_{\phi}(z|x)}[\log p_{\theta}(x, z)] = \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I)} \left[\nabla_{\phi} \log p_{\theta}(x, z_{\phi}(\epsilon))\right]
  \]
  - Uses the gradients w.r.t. the latent variables
  - For Gaussian distributions, \(\text{KL}\left(q_{\phi}(z|x) \parallel p(z)\right)\) can be computed and differentiated analytically
VAEs: Training

**Algorithm 1** Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

\[
\begin{align*}
\theta, \phi &\leftarrow \text{Initialize parameters} \\
\text{repeat} & \\
& \quad X^M \leftarrow \text{Random minibatch of } M \text{ datapoints (drawn from full dataset)} \\
& \quad \epsilon \leftarrow \text{Random samples from noise distribution } p(\epsilon) \\
& \quad g \leftarrow \nabla_{\theta,\phi} \tilde{L}^M(\theta, \phi; X^M, \epsilon) \text{ (Gradients of minibatch estimator (8))} \\
& \quad \theta, \phi \leftarrow \text{Update parameters using gradients } g \text{ (e.g. SGD or Adagrad [DHS10])} \\
\text{until convergence of parameters } (\theta, \phi) & \\
\text{return } \theta, \phi
\end{align*}
\]
VAEs: Results

\[ N_{\text{train}} = 1000 \]

\[ N_{\text{train}} = 50000 \]

- Marginal log-likelihood
- # Training samples evaluated (millions)

Graphs showing the performance of different methods with varying training sample sizes.
VAEs: Results

Figure 12. Generated CIFAR images. The rightmost column shows the nearest training examples to the column beside it.

[Gregor et al., 2015]

Generated MNIST images
VAEs: Limitations and variants

- **Element-wise reconstruction error**
  - For image generation, to reconstruct every pixel
  - Sensitive to irrelevant variance, e.g., translations
  - Variant: feature-wise (perceptual-level) reconstruction [Dosovitskiy et al., 2016]
    - Use a pre-trained neural network to extract features of data
    - Generated images are required to have similar feature vectors with the data
  - Variant: Combining VAEs with GANs [Larsen et al., 2016] (more later)

Reconstruction results with different loss
VAEs: Limitations and variants

- Not applicable to discrete latent variables
  - Differentiable reparameterization does not apply to discrete variables
  - Wake-sleep algorithm/GANs allow discrete latents
  - Variant: marginalize out discrete latents [Kingma et al., 2014]
    - Expensive when the discrete space is large
  - Variant: use continuous approximations
    - Gumbel-softmax [Jang et al, 2017] for approximating multinomial variables
  - Variant: combine VAEs with wake-sleep algorithm [Hu et al., 2017]
VAEs: Limitations and variants

- Usually use a fixed standard normal distribution as prior
  - $p(z) = \mathcal{N}(z; 0, I)$
  - For ease of inference and learning
  - Limited flexibility: converting the data distribution to fixed, single-mode prior distribution
- Variant: use hierarchical nonparametric priors [Goyal et al., 2017]
  - E.g., Dirichlet process, nested Chinese restaurant process (more later)
  - Learn the structures of priors jointly with the model

$$p(z) = \mathcal{N}(z; 0, I)$$
VAEs: Limitations and variants

Usually use a fixed standard normal distribution as prior for ease of inference and learning. Limited flexibility: converting the data distribution to fixed, single-mode prior distribution.

Variant: use hierarchical nonparametric priors [Goyal et al., 2017]. For example, Dirichlet process, nested Chinese restaurant process (more later) to learn the structures of priors jointly with the model.
Deep Generative Models

- Implicit probabilistic models
  - Defines a stochastic process to simulate data $x$
  - Do not require tractable likelihood function
  - Data simulator
  - Natural approach for problems in population genetics, weather, ecology, etc.

- E.g., generate data from a deterministic equation given parameters and random noise (e.g., GANs)

$$x_n = g(z_n; \theta)$$

$$z_n \sim N(0, I)$$
Generative Adversarial Nets (GANs)

- [Goodfellow et al., 2014]
- Assume implicit generative model
- Learn cost function jointly
- Interpreted as a mini-max game between a generator and a discriminator
- Generate sharp, high-fidelity samples
Generative Adversarial Nets (GANs)

- **Generator** \( x_n = G(z_n, \theta_g) \) \( z_n \sim N(0, I) \)
  - Maps noise variable to data space
- **Discriminator** \( D(x, \theta_d) \in [0, 1] \)
  - Outputs the probability that \( x \) came from the data rather than the generator

**Learning**
- Train \( D \) to maximize the probability of assigning the correct label to both training examples and generated samples
- Train \( G \) to fool the discriminator

\[
x \sim \mathcal{N}(0, I)
\]

[Figure courtesy: Kim’s slides]
GANs: Learning

For $D$: binary cross entropy with label 1 for real, 0 for fake

For $G$: $\min_G E_z [\log 1 - D(G(z))]$

Alternate training of $D$ and $G$

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim P_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim P(z)} [\log (1 - D(G(z)))].$$
GANs: Theoretical results

- For \( G \) fixed, the optimal discriminator \( D \) is

\[
D^*_G(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}
\]

- Plug in the minimax objective

\[
C(G) = \max_D V(G, D)
\]

\[
= \mathbb{E}_{x \sim p_{\text{data}}} [\log D^*_G(x)] + \mathbb{E}_{z \sim p_z} [\log (1 - D^*_G(G(z)))]
\]

\[
= \mathbb{E}_{x \sim p_{\text{data}}} [\log D^*_G(x)] + \mathbb{E}_{x \sim p_g} [\log (1 - D^*_G(x))]
\]

\[
= \mathbb{E}_{x \sim p_{\text{data}}} \left[ \log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \right] + \mathbb{E}_{x \sim p_g} \left[ \log \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)} \right]
\]

\[
C(G) = -\log(4) + KL \left( p_{\text{data}} \left\| \frac{p_{\text{data}} + p_g}{2} \right\| \right) + KL \left( p_g \left\| \frac{p_{\text{data}} + p_g}{2} \right\| \right)
\]

\[
C(G') = -\log(4) + 2 \cdot JSD \left( p_{\text{data}} \left\| p_g \right\| \right)
\]

- \( C^* = -\log 4 \) is the global minimum and the only solution is \( p_g = p_{\text{data}} \)
GANs in Practice

- Optimizing $D$ to completion in inner loop is computational prohibitive
  - Alternate between $k$ steps of optimizing $D$ and one step of optimizing $G$

- Optimizing $G$
  - $\min_G E_z[\log 1 - D(G(z))]$ suffers from vanishing gradient when $D$ is too strong
  - Uses $\max_G E_z[\log D(G(z))]$ in practice
GANs in Practice

- Instability of training
  - Requires careful balance between the training of $D$ and $G$
  - [Arjovsky & Botton, 2017]: under certain conditions, $\mathbb{E}_z[\log D(G(z))]$ is a centered Cauchy distribution with infinite expectation and variance

- Mode collapsing
  - Generated samples are often from only a few modes of the data distribution

- A set of heuristics attempting to fix the problems
  - e.g., [Salimans et al., 2016]: minibatch discrimination, one-sided label smoothing, …
GANs: Applications & results

- Generating images
GANs: Applications & results

- Generating images
- Translating images (e.g., Isola et al., 2016)
GANs: Applications & results

- Generating images
- Translating images (e.g., Isola et al., 2016)
- Domain adaptation (e.g., Purushotham et al., 2017)
- Imitation learning (e.g., Ho & Ermon 2016)
- …
GANs vs. VAEs

- Variational Auto-encoders:
  - Probabilistic graphical model framework
  - Allow efficient Bayesian inference
  - Generated samples tend to be blurry
    - An issue of maximum likelihood training
  - Do not support discrete latent variables

- GANs:
  - Generate sharp images
  - Do not support inference \((x \rightarrow z)\)
  - Do not support discrete visible variables
GANs: Limitations & variants

- Do not support inference
  - No mechanism to inferring $z$ from $x$
  - Variants: additionally learn an inference network
    - [Dumoulin, et al., 2016, Donahue et al., 2016]
GANs: Limitations & variants

- Do not support discrete visible variables
  - Non-differentiability of samples hinders gradients backprop
  - Variant: treats generator training as policy learning [Yu et al., 2017]
    - High variance, slow convergence
  - Variant: continuous approximations
    - Gumbel-softmax [Kusner & Hernndez-Lobato, 2016]
    - Only qualitative results with small discrete space
  - Variant: combines VAEs with GANs-like discriminators [Hu et al., 2017]
    - Uses VAEs to handle discrete visibles, GANs to handle discrete latents
    - Wake-sleep style learning
GANs: Limitations & variants

- Unable to control the attributes of generated samples
  - Uninterpretability of the input latent vector $z$
  - An Issue shared with VAEs and other DNN methods
  - Variants: add a mutual-information regularizer to enforce disentangled hidden codes [Chen et al., 2016]
    - Unsupervised
    - Semantic of each dimension is observed after training, rather than designated by users in a controllable way

(a) Varying $c_1$ on InfoGAN (Digit type)

(a) Rotation
GANs: Limitations & variants

- Unable to control the attributes of generated samples
  - Uninterpretability of the input latent vector $z$
  - An Issue shared with VAEs and other DNN methods
  - Variants: add a mutual-information regularizer to enforce disentangled hidden codes [Chen et al., 2016]
    - Unsupervised
    - Semantic of each dimension is observed after training, rather than designated by users in a controllable way
  - Variants: use supervision information to enforce designated semantics on certain dimensions of $z$ [Odena et al., 2017; Hu et al., 2017]

<table>
<thead>
<tr>
<th>Varying the code of sentiment</th>
<th>Varying the code of tense</th>
</tr>
</thead>
<tbody>
<tr>
<td>this movie was awful and boring .</td>
<td>this was one of the outstanding thrillers of the last decade</td>
</tr>
<tr>
<td>this movie was funny and touching .</td>
<td>this is one of the outstanding thrillers of the all time</td>
</tr>
<tr>
<td>jackson is n’t very good with documentary</td>
<td>this will be one of the great thrillers of the all time</td>
</tr>
<tr>
<td>jackson is superb as a documentary productions</td>
<td>i thought the movie was too bland and too much</td>
</tr>
<tr>
<td>you will regret it</td>
<td>i guess the movie is too bland and too much</td>
</tr>
<tr>
<td>you will enjoy it</td>
<td>i guess the film will have been too bland</td>
</tr>
</tbody>
</table>
Harnessing DNNs with Logic Rules

- [Hu et al., 2016]
- Deep NNs
  - Heavily rely on massive labeled data
  - Uninterpretable
  - Hard to encode human intention/domain knowledge
Harnessing DNNs with Logic Rules

- How humans learn
  - Learn from *concrete* examples (as DNNs do)
  - Learn from *general* knowledge and rich experiences [Minksy 1980; Lake et al., 2015]
  - E.g., the past tense of verbs\(^1\):

  **Examples:**
  - add -> added
  - accept -> accepted
  - ignore -> ignored
  - end -> ended
  - block -> blocked
  - love -> loved
  - ...

  **Rule:**
  - regular verbs –d/-ed

\(^1\) [https://www.technologyreview.com/s/544606/can-this-man-make-aimore-human](https://www.technologyreview.com/s/544606/can-this-man-make-aimore-human)
DNNs + knowledge

- Logic rule
  - A flexible declarative language
  - Expresses structured knowledge
  - E.g., sentence sentiment analysis

  \[ S \text{ has 'A-but-B' structure} \Rightarrow \left( \text{sentiment}(S) \Leftrightarrow \text{sentiment}(B) \right) \]

- Input-target space: \((X, Y)\)
- Soft first-order logic (FOL) rules: \((r, \lambda)\)
  - \(r(X, Y) \in [0,1]\)
  - \(\lambda\): confidence level of the rule
Rule knowledge distillation

- Neural network \( p_\theta(y|x) \)

at iteration \( t \):

\[
\theta^{(t+1)} = \arg\min_{\theta \in \Theta} \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, \sigma_\theta(x_n))
\]
Rule knowledge distillation

- Neural network $p_\theta(y|x)$
- Train to imitate the outputs of a rule-regularized teacher network (i.e. distillation)

at iteration $t$:

$$\theta^{(t+1)} = \arg \min_{\theta \in \Theta} \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, \sigma_\theta(x_n))$$

true hard label

soft prediction of $p_\theta$

soft prediction of the teacher network
Rule knowledge distillation

- Neural network $p_\theta(y|x)$
- Train to imitate the outputs of a rule-regularized teacher network (i.e. distillation)

at iteration $t$:

$$
\theta^{(t+1)} = \arg\min_{\theta \in \Theta} \frac{1}{N} \sum_{n=1}^{N} (1 - \pi) \ell(y_n, \sigma_\theta(x_n)) + \pi \ell(s_n^{(t)}, \sigma_\theta(x_n)),
$$

true hard label

soft prediction of $p_\theta$

balancing parameter

soft prediction of the teacher network
Teacher network construction

- Teacher network: \( q(Y|X) \)
  - Comes out of \( p \)
  - Fits the logic rules: \( E_q[r(X,Y)] = 1 \), with confidence \( \lambda \)

\[
\min_{q, \xi \geq 0} \ KL(q || p_\theta(Y|X)) + C \sum_l \xi_l \\
\text{s.t. } \lambda_l (1 - E_q[r_l(X,Y)]) \leq \xi_l \\
l = 1, \ldots, L
\]

Closed-form solution:

\[
q^*(Y|X) \propto p_\theta(Y|X) \exp \left\{ - \sum_l C\lambda_l (1 - r_l(X,Y)) \right\}
\]
Method summary

- At each iteration
  - Construct a teacher network through posterior constraints
  - Train the NN to emulate the predictions of the teacher
Application: sentiment classification

- Sentence => positive/negative
- Base network: CNN [Kim, 2014]

- Rule knowledge:
  - sentence S with structure A-but-B: => sentiment of B dominates
## Results

- **accuracy (%)**

<table>
<thead>
<tr>
<th>Model</th>
<th>SST2</th>
<th>MR</th>
<th>CR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 CNN (Kim, 2014)</td>
<td>87.2</td>
<td>81.3±0.1</td>
<td>84.3±0.2</td>
</tr>
<tr>
<td>2 CNN-Rule-(p)</td>
<td>88.8</td>
<td>81.6±0.1</td>
<td>85.0±0.3</td>
</tr>
<tr>
<td>3 CNN-Rule-(q)</td>
<td>89.3</td>
<td><strong>81.7±0.1</strong></td>
<td><strong>85.3±0.3</strong></td>
</tr>
<tr>
<td>4 MGNC-CNN (Zhang et al., 2016)</td>
<td>88.4</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>5 MVCNN (Yin and Schutze, 2015)</td>
<td><strong>89.4</strong></td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>6 CNN-multichannel (Kim, 2014)</td>
<td>88.1</td>
<td>81.1</td>
<td>85.0</td>
</tr>
<tr>
<td>7 Paragraph-Vec (Le and Mikolov, 2014)</td>
<td>87.8</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>8 CRF-PR (Yang and Cardie, 2014)</td>
<td>–</td>
<td>–</td>
<td>82.7</td>
</tr>
<tr>
<td>9 RNTN (Socher et al., 2013)</td>
<td>85.4</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>10 G-Dropout (Wang and Manning, 2013)</td>
<td>–</td>
<td>79.0</td>
<td>82.1</td>
</tr>
</tbody>
</table>
References

- [Dosovitskiy et al., 2016] “Generating Images with Perceptual Similarity Metrics based on Deep Networks”, NIPS’16
- [Larsen et al., 2016] “Autoencoding beyond pixels using a learned similarity metric”, ICML’16
- [Kingma et al., 2014] “Semi-supervised learning with deep generative models”, NIPS’14
- [Jang et al., 2017] “Categorical Reparameterization with Gumbel-Softmax”, ICLR’17
- [Hu et al., 2017] “Controllable Text Generation”, 2017
- [Goodfellow et al., 2014] “Generative Adversarial Nets”, NIPS’14
- [Salimans et al., 2016] “Improved Techniques for Training GANs”, NIPS’16
- [Isola et al., 2016] “Image-to-Image Translation with Conditional Adversarial Networks”, 2016
- [Purushotham et al., 2017] “Variational Recurrent Adversarial Deep Domain Adaptation”, ICLR’17
- [Ho & Ermon 2016] “Generative Adversarial Imitation Learning”, NIPS’16
- [Dumoulin, et al., 2016] “Adversarially Learned Inference”, ICLR’17
- [Donahue et al., 2016] “Adversarial Feature Learning”, 2016
- [Yu et al., 2017] “SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient”, AAAI’17
- [Chen el al., 2016] “InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets”, NIPS’16
- [Odena et al., 2017] “Conditional image synthesis with auxiliary classifier GANs”, 2017
- [Hu et al., 2016] “Harnessing DNNs with logic rules”, ACL’2016
- Hamid Reza Maei, Wake-Sleep algorithm for Representational Learning
- Mark Chang, Variational Autoencoder
- Namju Kim, Generative Adversarial Networks (GAN)