CS395T: Structured Models for NLP  
Lecture 21: Deep Generative Models I  
Greg Durrett

Administrivia

- Final project proposals due today
- Project 3 grades back tonight

Project 3 Results

- Tanya Goyal: 82.46
  - Ensemble of 2-layer LSTMs with mean pooling, two-step training procedure: fine-tune vectors, then fix vectors and learn LSTM
- Su Wang: 81.89
  - 2-channel CNN with 100 feature maps, batch = 64, 25 epochs, L2 regularization and aggressive learning rate decay (0.95 per 100 epochs)
- Aditya Gupta: 81.80
  - Ensemble of (Bi?)LSTMs, hidden state dim = 200, different learning rates for different pieces of the model
- Elisa Ferracane: 81.23
  - BiLSTM with hidden state dim = 32, batch = 16, epochs = 10

Recall: Memory Networks

- Memory networks let you reference input in an attention-like way
- Memorize input items into two vectors: a key and a value
- Keys compute attention weights given a query, weighted sum of values gives the output

$$ e_i = q \cdot k_i $$
$$ o = \sum_i \alpha_i v_i $$
$$ \alpha = \text{softmax}(e) $$

Sukhbaatar et al. (2015)
Recall: Attentive Reader

- Attentive reader: encode query, encode sentence, use attention to compute document representation, make prediction
- Uses fixed-size representations for the final prediction, multiclass classification

Recall: Bidirectional Attention Flow

This Lecture

- Variational autoencoders as deep generative models
  - Induce latent structure from the data by training in an unsupervised way
  - Can sample from them to produce examples
- Variational autoencoders as autoencoders

Deep Generative Models
Generative Models

- Discrete class bag-of-words model:
  \[ P(z) \]
  \[ P(x | z) \]
  \[ \text{discrete class: } \{1, 2, \ldots, n\} \]
  \[ \text{document words drawn from the vocabulary} \]
  \[ P(x | z) \]

Recall: EM for Generative Models

- Unsupervised learning: find parameters to maximize marginal likelihood
  \[ \log P(x | \theta) = \log \sum_z P(x, z | \theta) \]
- EM is a technique for doing this maximization

Recall: EM

\[
\log \sum_z P(x, z | \theta) \geq \mathbb{E}_{q(z)} \log P(x, z | \theta) + \text{Entropy}[q(z)]
\]
- Variational approximation \( q \)
- Jensen’s inequality (uses concavity of log)
- Can optimize this lower-bound on log likelihood instead of log-likelihood

Recall: EM

- If \( q(z) = P(z | x, \theta) \), equality is achieved
- Expectation-maximization: alternating maximization of the lower bound over \( q \) and \( \theta \)
  - Current timestep = \( t \), have parameters \( \theta^{t-1} \)
  - E-step: maximize w.r.t. \( q \); that is, \( q^t = P(z | x, \theta^{t-1}) \)
  - M-step: maximize w.r.t. \( \theta \); that is, \( \theta^t = \arg\max_{\theta} \mathbb{E}_{q^t} \log P(x, z | \theta) \)
Recall: EM

\[ L(x_1, \ldots, D) = \sum_{i=1}^{D} \log \sum_y P(y, x_i) \]

EM for Generative Models

- What form does \( q \) take? \( q \) is a multinomial, so just a bunch of numbers
  \[ q(z) = P(z|x, \theta) \propto P(z) \prod_i P(x_i|z) \]
- Easy to compute, easy to represent
- M-step: supervised learning problem with fractional annotation; possible
  because we can take the expectation:
  \[ \theta^t = \arg\max_{\theta} \mathbb{E}_{q(t)} \log P(x, z|\theta) \]

Deep Generative Models

- document words drawn from the vocabulary
  \[ P(x|z) = \text{softmax}(\text{emb}(x)^T z - b_x) \]
- Neural document model: probability of a word is depends on its dot
  product with the topic vector \( z \) (or use an even more complicated NN)
- What is \( P(z) \)? Let’s just say \( N(0, 1) \) for now...

EM for Deep Generative Models

- Expectation-maximization: alternating maximization over \( q \) and \( \theta \)
  
  E-step: maximize w.r.t. \( q \); that is, \( q^t = P(z|x, \theta^{t-1}) \)
  
  M-step: maximize w.r.t. \( \theta \); that is, \( \theta^t = \arg\max_{\theta} \mathbb{E}_{q(t)} \log P(x, z|\theta) \)

\[ P(z) = N(0, 1) \quad P(x|z) = \text{softmax}(\text{emb}(x)^T z - b_x) \]

- \( P(z|x) \) is now a complicated distribution, can’t simply use it for \( q \)
**EM for Deep Generative Models**

Expectation-maximization: alternating maximization over \( q \) and \( \theta \)

- **E-step**: maximize w.r.t. \( q \); that is, \( q^t = \arg\min_{q} KL(q(z) \| P(z|x)) \)
- **M-step**: maximize w.r.t. \( \theta \); that is, \( \theta^t = \arg\max_{\theta} \mathbb{E}_q \log P(x, z|\theta) \)

\[
P(z) = N(0, 1) \quad P(x|z) = \text{softmax}(\text{emb}(x)^T z - b_x)
\]

- \( P(z|x) \) is now a complicated distribution, can’t simply use it for \( q \)
- **E-step**: choose a family of distributions \( q \), find the best \( q \) in that family
  \( q = N(\mu, \text{diag}(\sigma^2)) \)
  - Even computing the best mu and sigma for an example is hard!
- **M-step**: now we need to take an expectation over a continuous distribution


**Deep Generative Models**

- EM doesn’t seem to be helping...let’s start over with the objective

\[
\log \sum_z P(x, z|\theta) = \log \sum_z q(z) \frac{P(x, z|\theta)}{q(z)} \geq \sum_z q(z) \log \frac{P(x, z|\theta)}{q(z)}
\]

- Different arrangement of terms: KL between \( q \) and prior + conditional likelihood term

**Comparison of Objectives**

- **EM**: \( \mathbb{E}_{q(z|x)}[\log P(x, z|\theta)] + \text{Entropy}[q(z|x)] + KL(q(z|x) \| P(z|x, \theta)) \)
  - “make the data likely under \( q \)” (generative)
- **VAE**: \( \mathbb{E}_{q(z|x)}[\log P(x|z, \theta)] - KL(q(z|x) \| P(z)) \)
  - “make the data likely under \( q \)” “make \( q \) close to the prior” (discriminative)
- Approximate \( q \) with a separate set of parameters, optimize \( q \) and \( \theta \) jointly with gradient descent
- Still need to reckon with that expectation over a continuous \( q(z) \)...

**Variational Autoencoders**

\[
\mathbb{E}_{q(z|x)}[\log P(x|z, \theta)] + KL(q(z|x) \| P(z))
\]

**Generative model (test):**

1. Input \( x \)
2. “inference network” \( z \)
3. \( q(z|x) \) distribution over \( z \)
4. generative model
5. Maximize \( P(x|z, \theta) \)

**Autoencoder (training):**

- Input \( x \)
- “inference network” \( q(z|x) \) distribution over \( z \)
- generative model
- Maximize \( P(x|z, \theta) \)

Miao et al. (2015)
Training VAEs

\[
\mathbb{E}_{q(z|x)}[\log P(x|z, \theta)] + \text{KL}(q(z|x) \parallel P(z))
\]

- Choose \(q\) to be Gaussian with parameters that are computed from \(x\)
  \[q = N(\mu(x), \text{diag}(\sigma^2(x)))\]
- \(\mu\) and \(\sigma\) are computed from one-layer feedforward networks over \(x\), call their parameters \(\phi\)
- How to handle the expectation? Just sample!

Miao et al. (2015)

Reparameterization Trick

- Can’t backpropagate through a sampling operation
  \[N(\mu, \text{diag}(\sigma^2)) = \mu + \sigma N(0, I)\]
- Recall that

Kingma and Welling (2013)

Training VAEs

For each example \(x\)

- Compute \(q\) (run forward pass to compute \(\mu\) and \(\sigma\))
- For some number of samples
  - Sample \(z \sim q\)
  - Compute \(P(x|z)\) and compute loss
  - Backpropagate to update \(\phi, \theta\)

Autoencoder (training):

\[\phi \xrightarrow{\text{"inference network"}} q(z|x)\]
\[\theta \xrightarrow{\text{generative model}} x\]
VAEs as Deep Generative Models

- **Continuous vector** $P(z)$
- **Document words drawn from the vocabulary**
  
  $$P(x|z) = \text{softmax}(\text{emb}(x)^\top z - b_x)$$

- We’ve seen a way to train this real-valued bag-of-words model in a fully unsupervised way
- “Encoder network” looks like the E-step of EM (but has distinct parameters), backpropagate end-to-end through encoder and decoder

Neural Variational Document Model

- Train this generative model on 20NewsGroups (online newsgroups) and RCV1 (newswire)
- Unsupervised learning: how to evaluate?
  - Data likelihood (perplexity)
  - See if interesting latent structure comes out

Randomly sample a dimension of $z$, see what words score highest along that axis, manually label that dimension

Neural generative model with hidden (boolean) variables $z$ and observed variables $x$

$$P(x, z) = \frac{1}{z} \exp(x^\top W z)$$

Contrastive divergence:
given $x$, compute $P(z|x)$, sample $z$
sample $x' \sim P(x|z)$, sample $z' \sim P(z'|x)$
update towards $(z, x)$ away from $(z', x')$
History: Wake-Sleep Algorithm

- Deep generative model with generation parameters G and “recognition” parameters R
- “Wake” phase: take data, encode it “upwards” using R, train G in a supervised way
- “Sleep” phase: generate top-down, train R in a supervised way
- One layer of this trained end-to-end looks like VAEs
  - data -> layer1 -> data
  - $R_1$ $G_1$

Hinton et al. (1995)

VAEs as Autoencoders

Encoder-Decoder Models

- Encoder-decoder models without attention: compress the input into a single &*#! vector, unfold it to produce output

Autoencoders

- Encoder-decoder models without attention: compress the input into a single &*#! vector, unfold it to produce output
- Autoencoder: encode input $x$ into a vector $z$, produce $x$ given $z$

$$P(x'|x) = P(z|x) \prod_i P(x_i'|z, x_{<i}')$$

- What semantics do we want the latent space to have?
Autoencoders

- What semantics do we want the latent space to have?
- What semantics does the latent space actually have?
- Can encode a word into a single floating-point value
  a of the in ...
- Map a sentence of length $k$ into a $k$-dimensional $z$

<table>
<thead>
<tr>
<th>$z_1$</th>
<th>$z_2$</th>
<th>$z_3$</th>
<th>$z_4$</th>
<th>$z_5$</th>
<th>$z_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$z_1$</th>
<th>$z_2$</th>
<th>$z_3$</th>
<th>$z_4$</th>
<th>$z_5$</th>
<th>$z_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Can an LSTM learn to do this?
- Yes!
- Want continuous semantic structure in the latent space: nearby points should have similar meaning

LSTM VAEs

$E_{q(z|x)}[\log P(x|z, \theta)] + KL(q(z|x)||P(z))$

- Train this up; what happens?
  the movie was good </s> </s>
  the movie was great </s> </s>
  I thought the film was good

- During training, add Gaussian noise and force the network to predict
- Same computation graph as VAE with reparameterization, add KL term to make the objective the same
- Inference network ($q$) is the encoder and generator is the decoder
**KL Collapse**

\[ \mathbb{E}_{q(z|x)}[\log P(x|z, \theta)] + \text{KL}(q(z|x) \| P(z)) \]

- What does gradient encourage latent space to do?

<table>
<thead>
<tr>
<th>prior</th>
<th>direction of better likelihood for ( x )</th>
</tr>
</thead>
</table>

- In reality, the likelihood signal is very weak, \( z \) is set to 0

**A Tale of Two Decoders**

\[ \mathbb{E}_{q(z|x)}[\log P(x|z, \theta)] + \text{KL}(q(z|x) \| P(z)) \]

- Suppose vocab is \{A, B, C\}. Sentences are either AAA, BBB, or CCC

<table>
<thead>
<tr>
<th>B</th>
<th>B</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
</tbody>
</table>

- LL = \((1/3)^3\) if input is ignored
- LL = \(1/3\) if input is ignored

- NVDM: using \( z \) can help a lot
- LSTM: can get decent likelihood ignoring \( z \) entirely

**Solution:** anneal KL term during learning

- Model initially uses \( z \) a lot (\( q \) gets far from the prior), then as KL term is turned up the prior balances it more

Bowman et al. (2016)
Results

- Train autoencoder on the Penn Treebank (pretty small corpus for language modeling purposes)

<table>
<thead>
<tr>
<th>Model</th>
<th>Train NLL</th>
<th>Train PPL</th>
<th>Test NLL</th>
<th>Test PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNNLM</td>
<td>100</td>
<td>95</td>
<td>100</td>
<td>116</td>
</tr>
<tr>
<td>VAE</td>
<td>98 (2)</td>
<td>100</td>
<td>101 (2)</td>
<td>119</td>
</tr>
</tbody>
</table>

- Doesn’t really improve perplexities over RNNLM: confirms that RNN is pretty good at modeling the space

Bowman et al. (2016)

Results

- Encode sentence, sample from q, generate from those samples

  - we looked out at the setting sun.
  - i went to the kitchen.
  - how are you doing?
  - we were laughing at the same time.
  - i went to my apartment.
  - what are you doing?
  - i’ll see you in the early morning.
  - i looked around the room.
  - are you sure?
  - i looked up at the blue sky.
  - i turned back to the table.
  - what are you doing?
  - it was down on the dance floor.

- Encode two samples, generate from points interpolated between the two samples

Bowman et al. (2016)

Takeaways

- VAE is a framework for training deep generative models
- VAE can be seen as either a principled variational method motivated by a lower bound or simply an ad-hoc trick to make latent spaces more continuous
- Some tricks to get these models to train well
- Generative objective ensures that the latent space z has interesting and coherent semantics; lets us sample from z and generate instances from the data manifold