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

Greg Durrett
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

\[
\begin{align*}
q & = \text{softmax}(e) \\
o & = \sum_i \alpha_i v_i \\
e_i & = q \cdot k_i
\end{align*}
\]

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

Seo et al. (2016)
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:

  \[ P(z) \]
  \[ P(x|z) \]

  discrete class: \( \{1, 2, ..., n\} \)

  document words drawn from the vocabulary

  \[ P(z = \text{Science}) \]
  \[ P(\text{too}|z = \text{Science}) \]
  \[ P(\text{many}|z = \text{Science}) \]
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

Science

too many drug trials too few patients

\[P(z = \text{Science})P(\text{too}|z = \text{Science})P(\text{many}|z = \text{Science}) \ldots\]
Recall: EM

$$\log \sum_z P(x, z|\theta)$$

$$= \log \sum_z q(z) \frac{P(x, z|\theta)}{q(z)}$$  \hspace{10pt} \text{Variational approximation } q

$$\geq \sum_z q(z) \log \frac{P(x, z|\theta)}{q(z)}$$  \hspace{10pt} \text{Jensen’s inequality (uses concavity of log)}

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

\hspace{10pt} \text{Can optimize this lower-bound on log likelihood instead of log-likelihood}

Adapted from Leon Gu
Recall: EM

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

- 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

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

**M-step:** find maximum of lower bound

**E-step:** compute lower bound

initial params

M-step: find maximum of lower bound

E-step: compute lower bound
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:

$$\theta^t = \arg\max_\theta \mathbb{E}_{q^t} \log P(x, z|\theta)$$

Too many drug trials too few patients
Deep Generative Models

![Diagram]

- **Continuous vector** $P(z)$
- $P(x|z) = \text{softmax}(\text{emb}(x)^\top z - b_x)$

- 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 $\mathcal{N}(0, 1)$ for now...

Miao et al. (2015)
EM for Deep Generative Models

Expectation-maximization: alternating maximization over $q$ and $\theta$

E-step: maximize w.r.t. $q$; that is, $q^t = P(z|x, \theta^t \perp)$

M-step: maximize w.r.t. $\theta$; that is, $\theta^t = \text{argmax}_{\theta} \mathbb{E}_{q^t} \log P(x, z|\theta)$

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

- $P(z|x)$ is now a complicated distribution, can’t simply use it for $q$
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, $\theta^t = \arg\max_\theta \mathbb{E}_{q^t} \log P(x, z|\theta)$

$P(z) = N(0, 1)$  
$P(x|z) = \text{softmax}(\text{emb}(x)^\top z - b_x)$

- $P(z|x)$ 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

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

Jensen

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

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

- 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

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

Generative model (test):

Input

```
  z
  ↓
  x
```

Autoencoder (training):

```
  x  Input
  ↓
“inference network”

  q(z|x) distribution over z
  ↓
  generative model

  x  Maximize $P(x|z, \theta)$
```

Miao et al. (2015)
Training VAEs

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

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

- How to handle the expectation? Just sample!

Autoencoder (training):

- Input: \( x \)
- Inference network: \( q(z|x) \)
- Generative model: \( x \)

Miao et al. (2015)
Training VAEs

For each example \( \mathbf{x} \)

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

Autoencoder (training):

\[
\begin{align*}
  &\mathbf{x} \\
  &\phi \quad \text{“inference network”} \\
  &q(z|x) \\
  &\theta \quad \text{generative model} \\
  &\mathbf{x}
\end{align*}
\]
Reparameterization Trick

- Can’t backpropagate through a sampling operation

- Recall that
  \[ N(\mu, \text{diag}(\sigma^2)) = \mu + \sigma N(0, I) \]
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):

\[
\begin{align*}
\phi & \quad \text{“inference network”} \\
q(z|x) & \\
\theta & \quad \text{generative model}
\end{align*}
\]
VAEs as Deep Generative Models

- 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.

\[
P(x|z) = \text{softmax}(\text{emb}(x)^\top z - b_x)
\]
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

Miao et al. (2015)
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 $(z', x')$

"inference network"
"generative network"

Smolensky (1986), Carreira-Perpiñán and Hinton (2005)
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

R1  G1

Hinton et al. (1995)
VAEs as Autoencoders
Encoder-decoder models without attention: compress the input into a single &*#! vector, unfold it to produce output

the movie was great

le film était bon [STOP]
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
- Map a sentence of length $k$ into a $k$-dimensional $z$

```
the movie was good </s> </s>  
the movie was great </s> </s>  
I thought the film was good
```

$z_1$  $z_2$  $z_3$  $z_4$  $z_5$  $z_6$
Autoencoders

the movie was **good** </s> </s>  

the movie was **great** </s> </s>  

I thought the film was **good**  

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

- 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
LSTM VAEs

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

- Train this up; what happens?
  - the movie was good</s></s>
  - the movie was great</s></s>
  - I thought the film was good
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?
What does gradient encourage latent space to do?

In reality, the likelihood signal is very weak, z is set to 0
A Tale of Two Decoders

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

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

  - LL = $(1/3)^3$ if input is ignored
  - NVDM: using $z$ can help a lot

  - LL = 1/3 if input is ignored
  - LSTM: can get decent likelihood ignoring $z$ entirely
A Tale of Two Decoders

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

- 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>RNLM</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 RNLM: confirms that RNN is pretty good at modeling the space

Bowman et al. (2016)
Results

<table>
<thead>
<tr>
<th>INPUT</th>
<th>MEAN</th>
<th>SAMP. 1</th>
<th>SAMP. 2</th>
<th>SAMP. 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>we looked out at the setting sun</td>
<td>they were laughing at the same time</td>
<td>ill see you in the early morning</td>
<td>i looked up at the blue sky</td>
<td>it was down on the dance floor</td>
</tr>
<tr>
<td>i went to the kitchen</td>
<td>i went to the kitchen</td>
<td>i went to my apartment</td>
<td>i looked around the room</td>
<td>i turned back to the table</td>
</tr>
<tr>
<td>how are you doing?</td>
<td>what are you doing?</td>
<td>“are you sure?”</td>
<td>what are you doing?</td>
<td>what are you doing?</td>
</tr>
</tbody>
</table>

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

  “i want to talk to you.”
  “i want to be with you.”
  “i do n’t want to be with you.”
  i do n’t want to be with you.
  she did n’t want to be with him.

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

  he was silent for a long moment.
  he was silent for a moment.
  it was quiet for a moment.
  it was dark and cold.
  there was a pause.
  it was my turn.

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