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

Greg Durrett

Administrivia

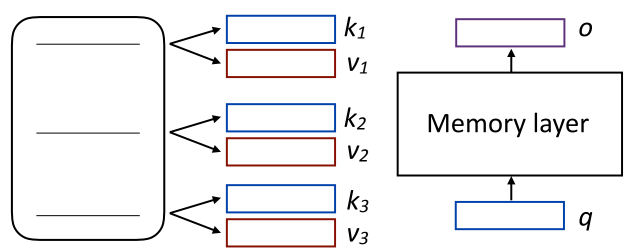
- Final project proposals due today
- Project 3 grades back tonight

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

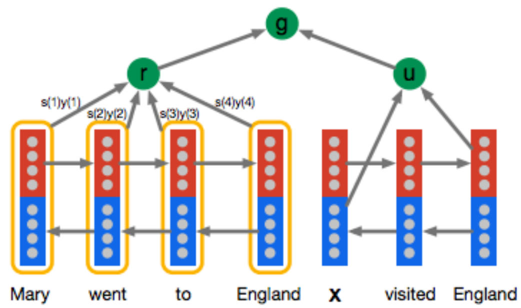


$$o = \sum_i \alpha_i v_i$$
$$\alpha = \mathrm{softmax}(e)$$
$$e_i = q \cdot k_i$$

Sukhbaatar et al. (2015)

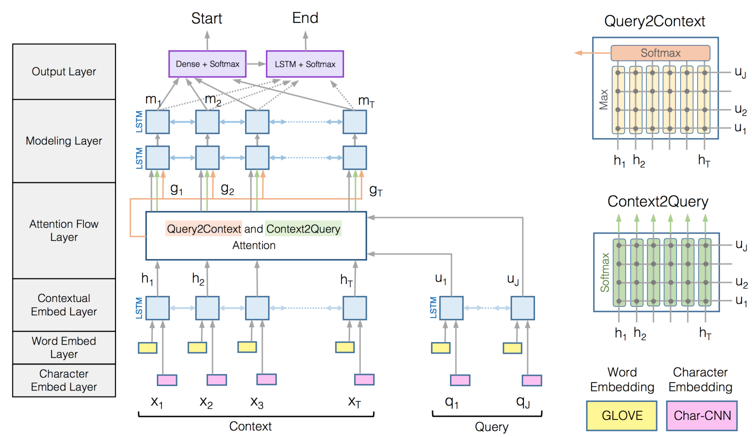
Recall: Attentive Reader

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



Hermann et al. (2015)

Recall: Bidirectional Attention Flow



Seo et al. (2016)

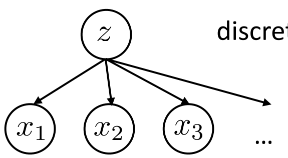
This Lecture

- 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

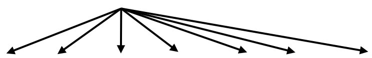
- Discrete class bag-of-words model:



discrete class: {1, 2, …, n} $P(z)$

document words drawn from the vocabulary $P(x|z)$

Science



too many drug trials too few patients

$$P(z = \text{Science})P(\text{too}|z = \text{Science})P(\text{many}|z = \text{Science})\dots$$

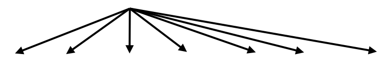
Recall: EM for Generative Models

- Unsupervised learning: find parameters to maximize marginal likelihood

$$\log P(\mathbf{x}|\theta) = \log \sum_z P(\mathbf{x}, z|\theta)$$

- EM is a technique for doing this maximization

Science



too many drug trials too few patients

$$P(z = \text{Science})P(\text{too}|z = \text{Science})P(\text{many}|z = \text{Science})\dots$$

Recall: EM

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

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

- Variational approximation *q*

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

- Jensen's inequality (uses concavity of log)

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

- Can optimize this lower-bound on log likelihood instead of log-likelihood

Adapted from Leon Gu

Recall: EM

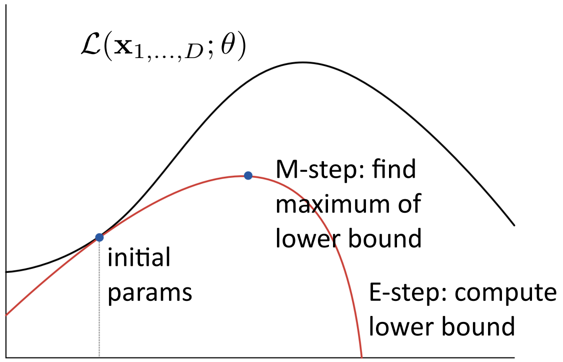
$$\log \sum_z P(\mathbf{x}, z|\theta) \geq \mathbb{E}_{q(z)} \log P(\mathbf{x}, z|\theta) + \text{Entropy}[q(z)]$$

- If $q(z) = P(z|\mathbf{x}, \theta)$, equality is achieved

- Expectation-maximization: alternating maximization of the lower bound over *q* and θ
  - Current timestep = *t*, have parameters θ^{t-1}
  - E-step: maximize w.r.t. *q*; that is, $q^t = P(z|\mathbf{x}, \theta^{t-1})$
  - M-step: maximize w.r.t. θ; that is, $\theta^t = \text{argmax}_\theta \mathbb{E}_{q^t} \log P(\mathbf{x}, z|\theta)$

Recall: EM

$$\mathcal{L}(\mathbf{x}_{1,\ldots,D}) = \sum_{i=1}^{D} \log \sum_{\mathbf{y}} P(\mathbf{y}, \mathbf{x}_i)$$

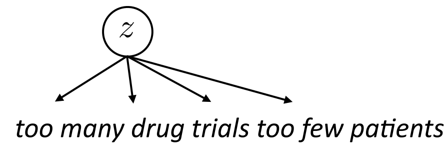


EM for Generative Models

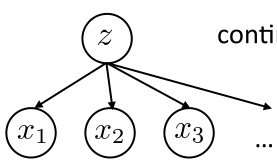
- What form does q take? q is a multinomial, so just a bunch of numbers

$$q(z) = P(z|\mathbf{x},\theta) \propto P(z)\prod_i P(x_i|z)$$

- Easy to compute, easy to represent
- M-step: supervised learning problem with fractional annotation; possible because we can take the expectation: $\theta^t = \text{argmax}_\theta \mathbb{E}_{q^t} \log P(\mathbf{x}, z|\theta)$



Deep Generative Models



continuous vector $P(z)$

document words drawn from the vocabulary

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

- 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…

Miao et al. (2015)

EM for Deep Generative Models

Expectation-maximization: alternating maximization over q and θ

E-step: maximize w.r.t. q; that is, $q^t = \cancel{P(z|\mathbf{x}, \theta^{t-1})}$

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

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

- P(z|**x**) is now a complicated distribution, can't simply use it for q

EM for Deep Generative Models

Expectation-maximization: alternating maximization over *q* and θ

E-step: maximize w.r.t. *q*; that is, $q^t = \text{argmin}_q \text{KL}(q(z) \| P(z|\mathbf{x}))$

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

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

- 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

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

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

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

- Different arrangement of terms: KL between *q* and *prior* + conditional likelihood term

Comparison of Objectives

- EM: $\mathbb{E}_{q(z|\mathbf{x})}[\log P(\mathbf{x}, z|\theta)] + \text{Entropy}[q(z|\mathbf{x})] + KL(q(z|\mathbf{x}) \| P(z|\mathbf{x}, \theta))$

  "make the data likely under q"
  (generative)

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

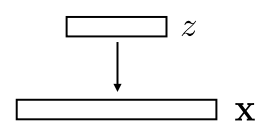
  "make the data likely under q" "make q close to the prior"
  (discriminative)

- 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)…

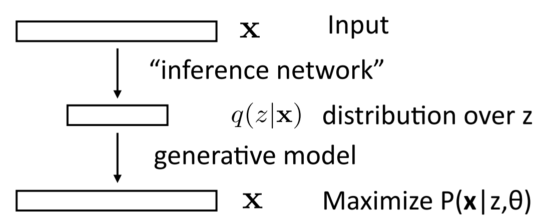
Variational Autoencoders

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

Generative model (test):



Autoencoder (training):



Miao et al. (2015)

Training VAEs

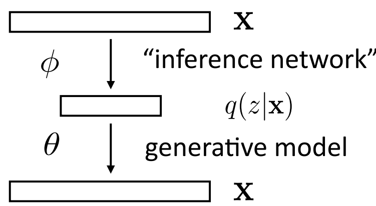
$$\mathbb{E}_{q(z|\mathbf{x})}[\log P(\mathbf{x}|z,\theta)] + \mathrm{KL}(q(z|\mathbf{x})\|P(z))$$

- Choose *q* to be Gaussian with parameters that are computed from **x**

$$q = N(\mu(\mathbf{x}), \mathrm{diag}(\sigma^2(\mathbf{x})))$$

- mu and sigma are computed from one-layer feedforward networks over **x**, call their parameters ϕ

Autoencoder (training):



- How to handle the expectation? Just sample!

Miao et al. (2015)

Training VAEs

For each example **x**

Compute *q* (run forward pass to compute mu and sigma)

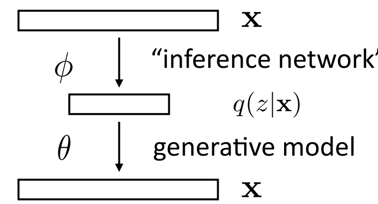
For some number of samples

Sample z ~ *q*

Compute P(**x**|z) and compute loss

Backpropagate to update phi, theta

Autoencoder (training):

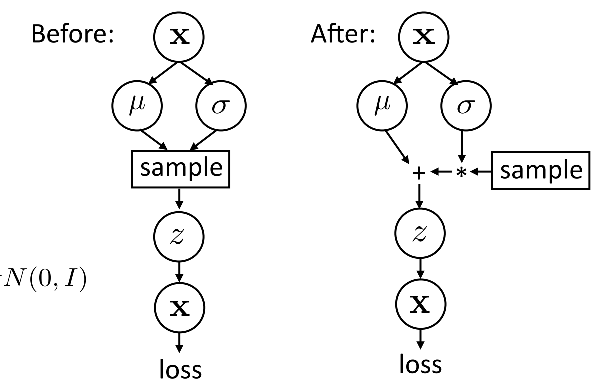


Reparameterization Trick

- Can't backpropagate through a sampling operation

- Recall that

$$N(\mu, \mathrm{diag}(\sigma^2)) = \mu + \sigma N(0, I)$$



Kingma and Welling (2013)

Training VAEs

For each example **x**

Compute *q* (run forward pass to compute mu and sigma)

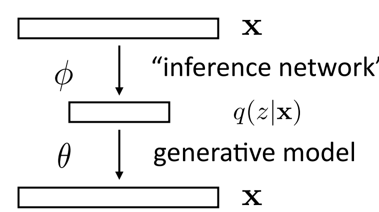
For some number of samples

Sample z ~ *q*

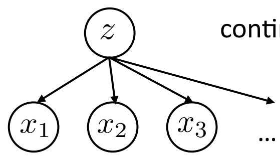
Compute P(**x**|z) and compute loss

Backpropagate to update phi, theta

Autoencoder (training):



VAEs as Deep Generative Models



continuous vector $P(z)$

document words drawn from the vocabulary

$$P(x|z) = \mathrm{softmax}(\mathrm{emb}(x)^\top z - b_x)$$

- 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

Miao et al. (2015)

Neural Variational Document Model

Model	Dim	20News	RCV1
LDA	50	1091	1437
LDA	200	1058	1142
RSM	50	953	988
docNADE	50	896	742
SBN	50	909	784
fDARN	50	917	724
fDARN	200	—	598
NVDM	50	**836**	563
NVDM	200	852	**550**

(a) Perplexity on test dataset.

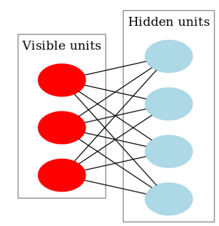
- Randomly sample a dimension of z, see what words score highest along that axis, manually label that dimension

Space	*Religion*	*Encryption*	*Sport*	*Policy*
orbit	muslims	rsa	goals	bush
lunar	worship	cryptography	pts	resources
solar	belief	crypto	teams	charles
shuttle	genocide	keys	league	austin
moon	jews	pgp	team	bill
launch	islam	license	players	resolution
fuel	christianity	secure	nhl	mr
nasa	atheists	key	stats	misc
satellite	muslim	escrow	min	piece
japanese	religious	trust	buf	marc

Miao et al. (2015)

History: Restricted Boltzmann Machines

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



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

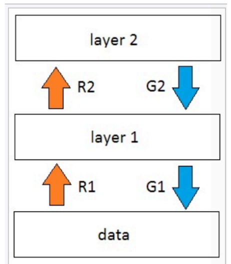
- Contrastive divergence:
  given x, compute P(z|x), sample z
  sample x' ~ P(x|z), sample z' ~ P(z'|x)
  update towards (z, x) away from (z', x')

"inference network"
"generative network"

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

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



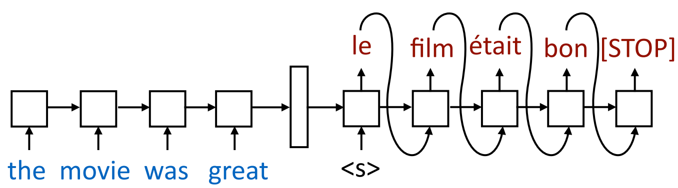
- One layer of this trained end-to-end looks like VAEs

  data -> layer1 -> data
  R1 G1

Hinton et al. (1995)

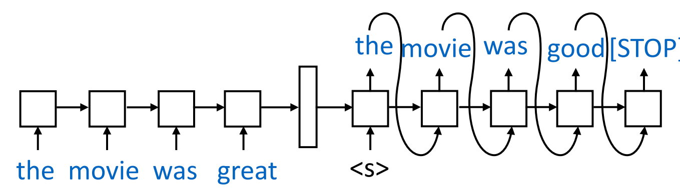
VAEs as Autoencoders

Encoder-Decoder Models



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

Autoencoders



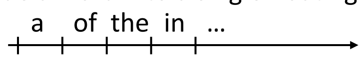
- Encoder-decoder models without attention: compress the input into a single &*#! vector, unfold it to produce output
- Autoencoder: encode input **x** into a vector z, produce **x** given z

$$\underbrace{P(\mathbf{x}'|\mathbf{x}) = P(z|\mathbf{x})}_{\text{encoder}} \underbrace{\prod_i P(x'_i|z, \mathbf{x}'_{<i})}_{\text{decoder}}$$

- 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

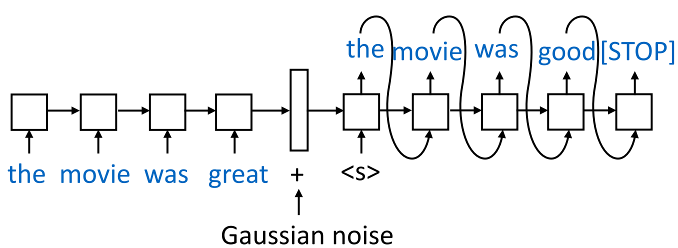


- Map a sentence of length *k* into a *k*-dimensional z

Autoencoders

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

Autoencoders

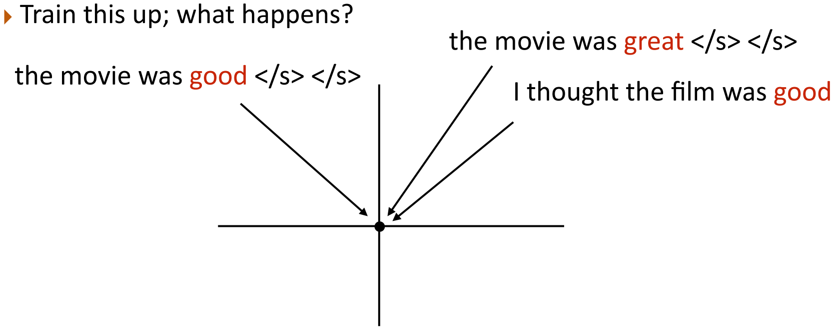


- During training, add Gaussian noise and force the network to predict
- Same computation graph as VAE with reparameterization, add KL term to make the objective the same
- Inference network (q) is the encoder and generator is the decoder

LSTM VAEs

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

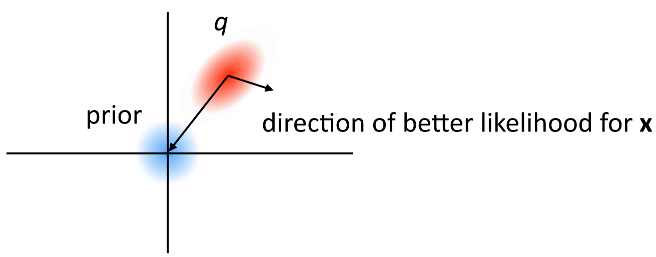
- Train this up; what happens?



KL Collapse

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

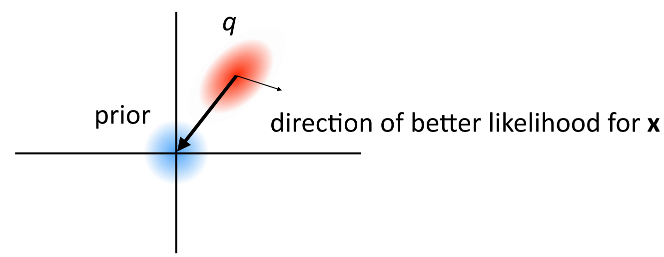
- What does gradient encourage latent space to do?



KL Collapse

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

- What does gradient encourage latent space to do?

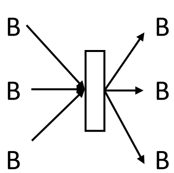


- In reality, the likelihood signal is very weak, z is set to 0

A Tale of Two Decoders

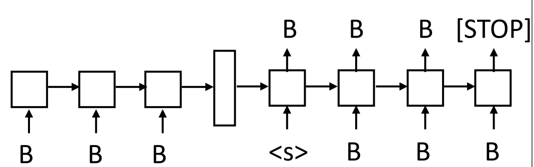
$$\mathbb{E}_{q(z|\mathbf{x})}[\log P(\mathbf{x}|z,\theta)] + \mathrm{KL}(q(z|\mathbf{x})\|P(z))$$

- 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



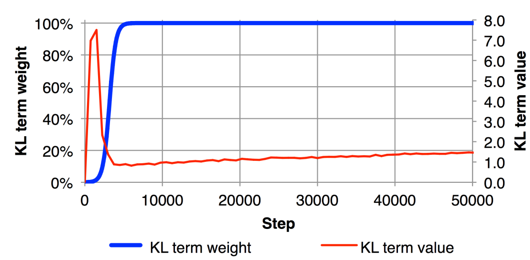
LL = 1/3 if input is ignored

- LSTM: can get decent likelihood ignoring z entirely

A Tale of Two Decoders

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

- 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)

Model	Standard			
	Train NLL	Train PPL	Test NLL	Test PPL
RNNLM	100 –	95	**100** –	**116**
VAE	98 (2)	100	101 (2)	119

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

Bowman et al. (2016)

Results

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

- 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 .

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 .

- 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