CS11-747 Neural Networks for NLP

Models w/ Latent Random Variables

Graham Neubig

Carnegie Mellon University

Language Technologies Institute

Site

https://phontron.com/class/nn4nlp2017/

Discriminative vs. Generative Models

- **Discriminative model:** calculate the probability of output given input P(Y|X)
- **Generative model:** calculate the probability of a variable P(X), or multiple variables P(X,Y)
- Which of the following models are discriminative vs. generative?
  - Standard BiLSTM POS tagger
  - Globally normalized CRF POS tagger
  - Language model

Types of Variables

- Observed vs. Latent:
  - **Observed:** something that we can see from our data, e.g. X or Y
  - **Latent:** a variable that we assume exists, but we aren't given the value
- Deterministic vs. Random:
  - **Deterministic:** variables that are calculated directly according to some deterministic function
  - **Random (stochastic):** variables that obey a probability distribution, and may take any of several (or infinite) values

Quiz: What Types of Variables?

- In the an attentional sequence-to-sequence model using MLE/teacher forcing, are the following variables observed or latent? deterministic or random?
  - The input word ids **f**
  - The encoder hidden states **h**
  - The attention values **a**
  - The output word ids **e**

Variational Auto-encoders

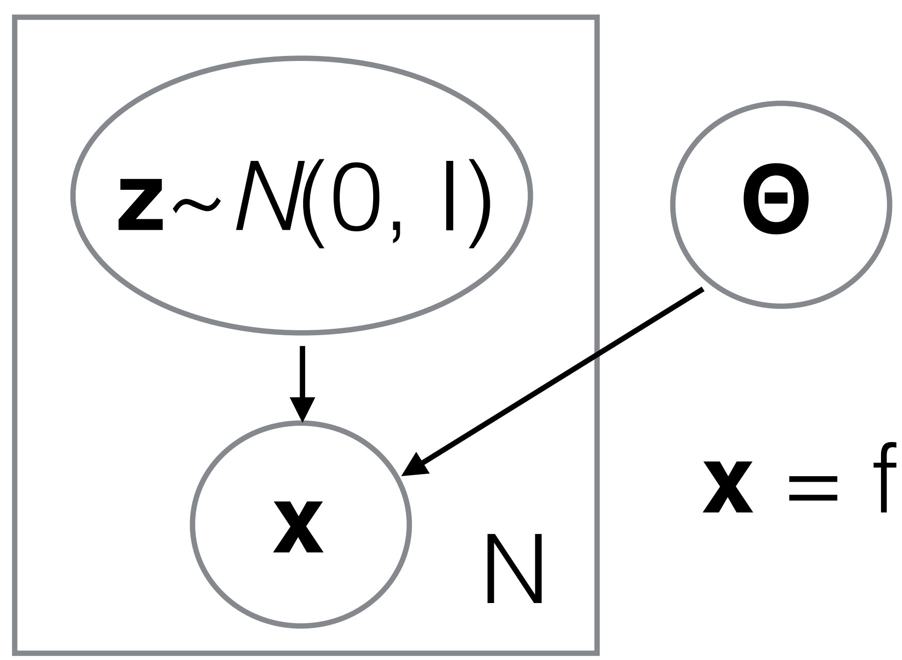
(Kingma and Welling 2014)

Why Latent Random Variables?

- We believe that there are underlying latent factors that affect the text/images/speech that we are observing
  - What is the content of the sentence?
  - Who is the writer/speaker?
  - What is their sentiment?
  - What words are aligned to others in a translation?
- All of these have a correct answer, *we just don't know what it is.* Deterministic variables cannot capture this ambiguity.

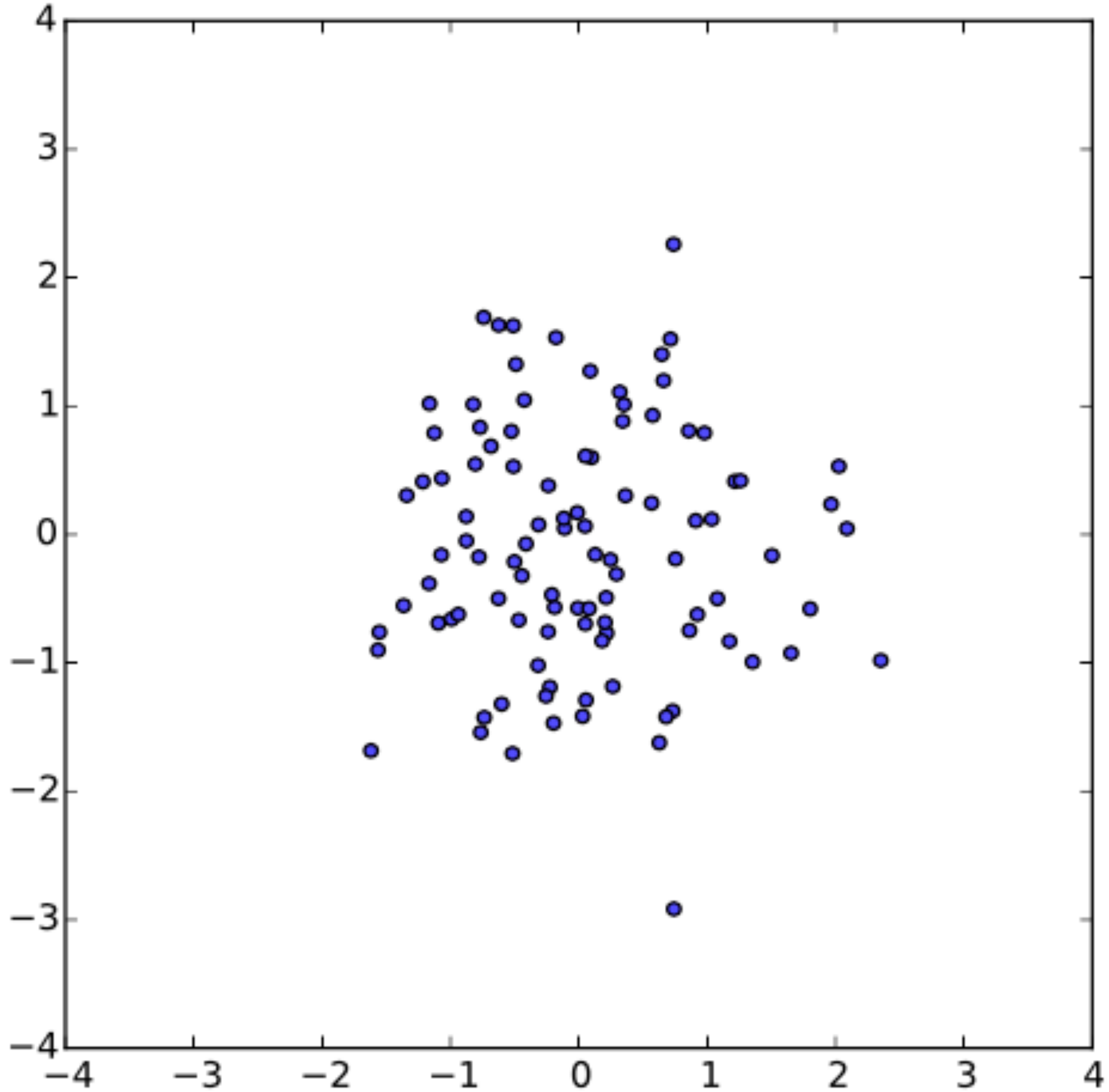
A Latent Variable Model

- We observed output **x** (assume a continuous vector for now)
- We have a latent variable **z** generated from a Gaussian
- We have a function f, parameterized by **Θ** that maps from **z** to **x**, where this function is usually a neural net

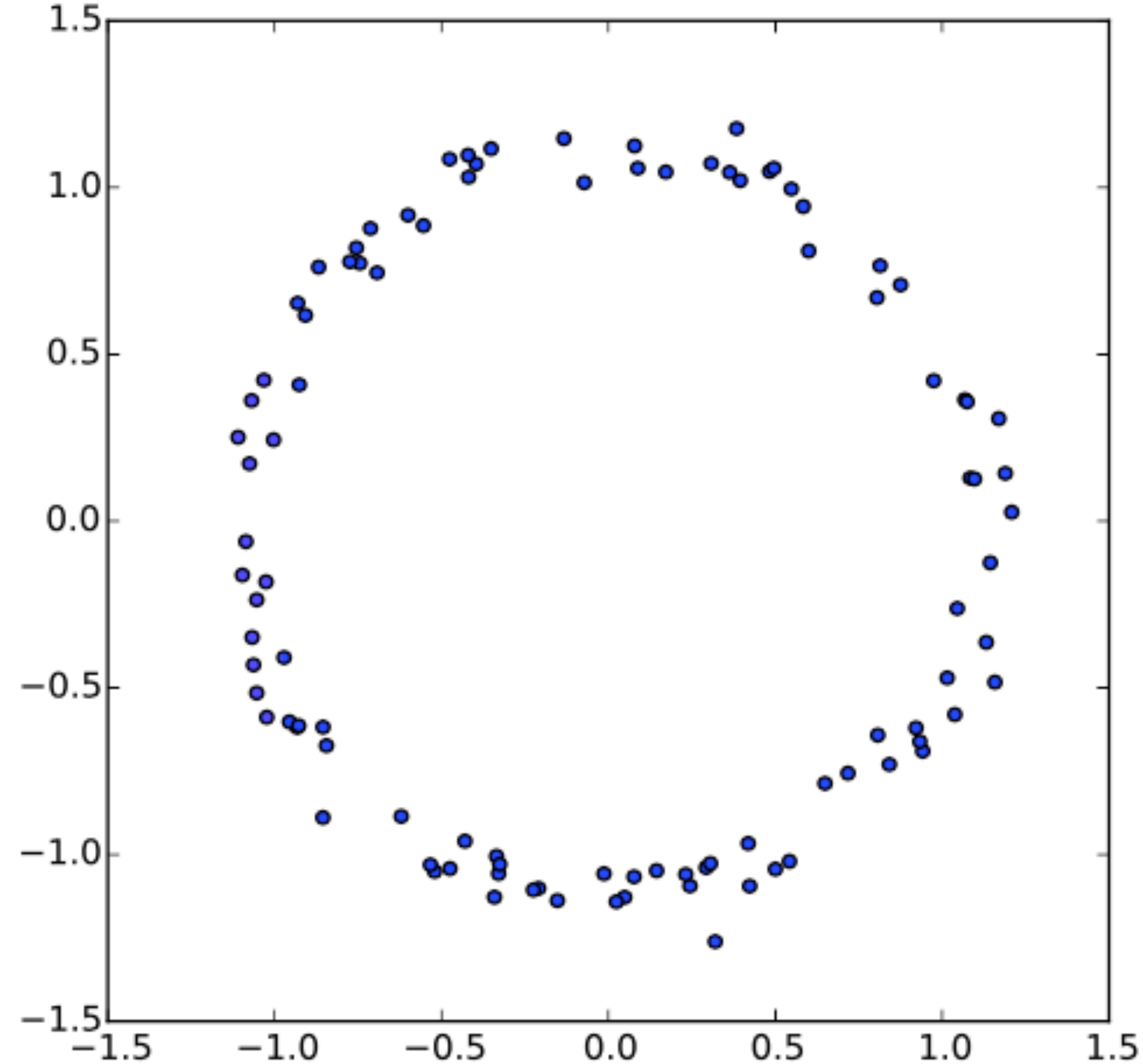


$\mathbf{x} = f(\mathbf{z}; \mathbf{\Theta})$

An Example (Goersch 2016)



z



x

What is Our Loss Function?

- We would like to maximize the corpus log likelihood

$$\log P(\mathcal{X}) = \sum_{\boldsymbol{x} \in \mathcal{X}} \log P(\boldsymbol{x}; \theta)$$

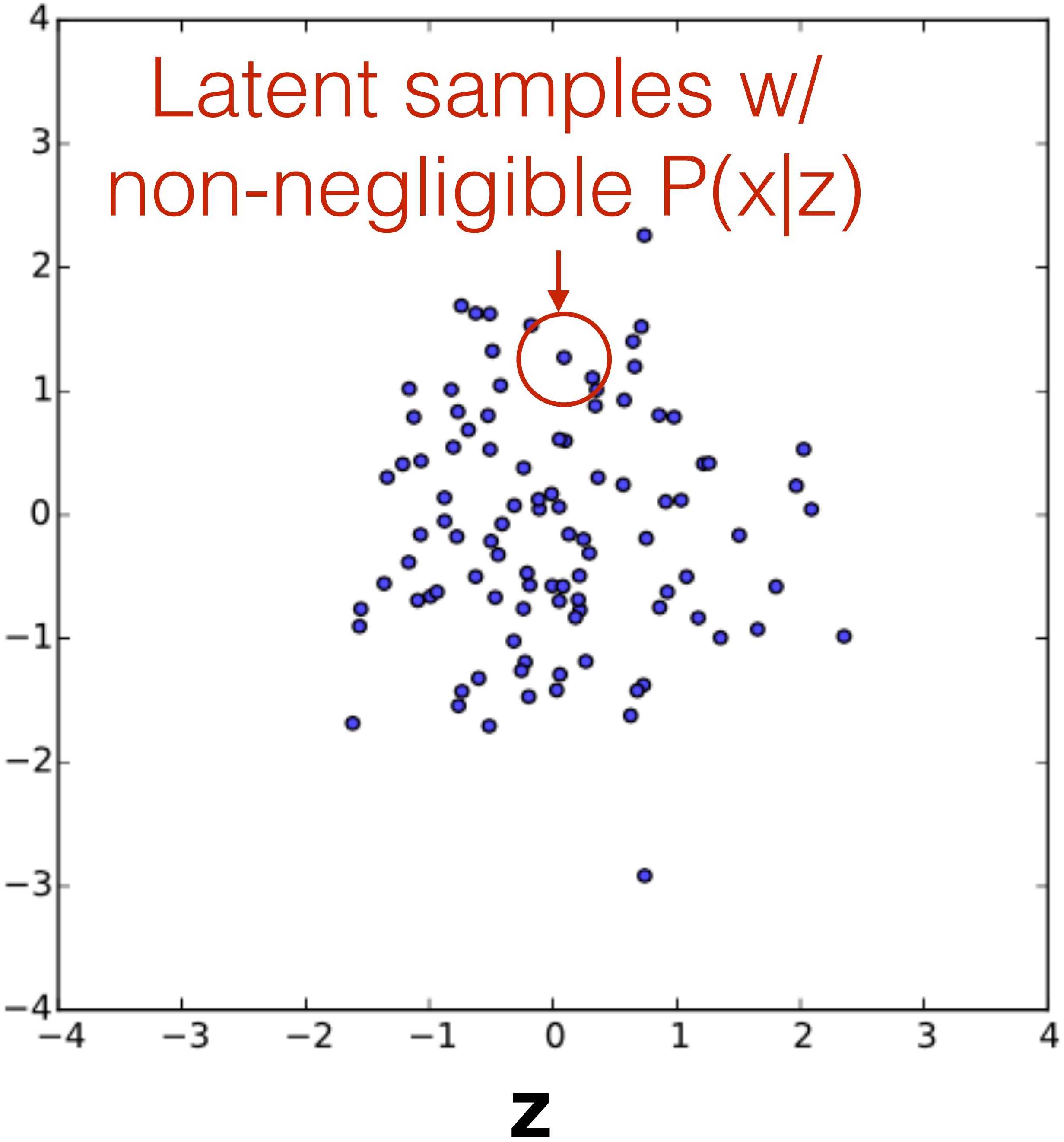
- For a single example, the marginal likelihood is

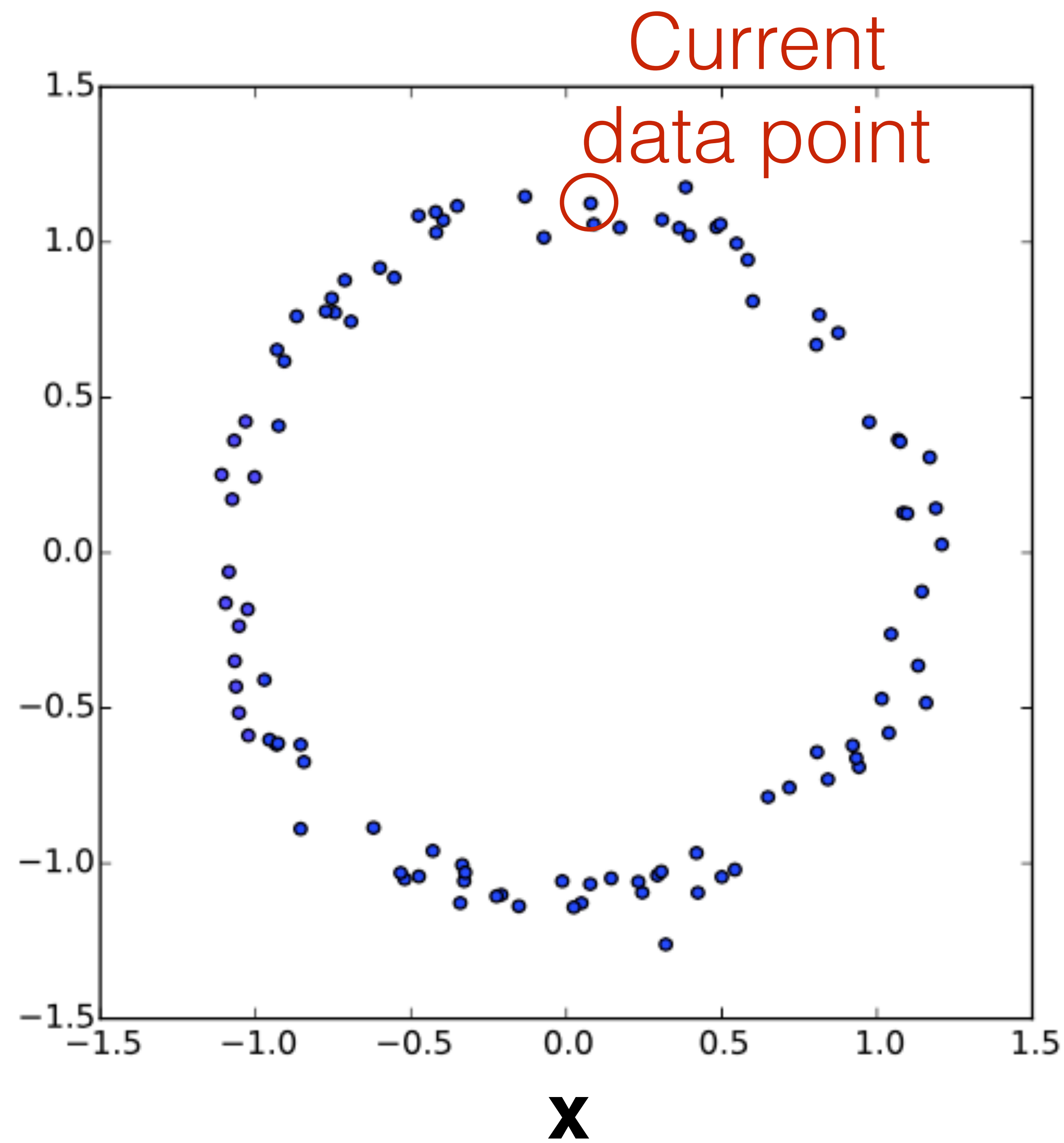
$$P(\boldsymbol{x}; \theta) = \int P(\boldsymbol{x} \mid \boldsymbol{z}; \theta) P(\boldsymbol{z}) d\boldsymbol{z}$$

- We can approximate this by sampling **z**s then summing

$$P(\boldsymbol{x}; \theta) \approx \sum_{\boldsymbol{z} \in S(\boldsymbol{x})} P(\boldsymbol{x} | \boldsymbol{z}; \theta) \quad \text{where} \quad S(\boldsymbol{x}) := \{\boldsymbol{z}'; \boldsymbol{z}' \sim P(\boldsymbol{z})\}$$

Problem: Straightforward Sampling is Inefficient

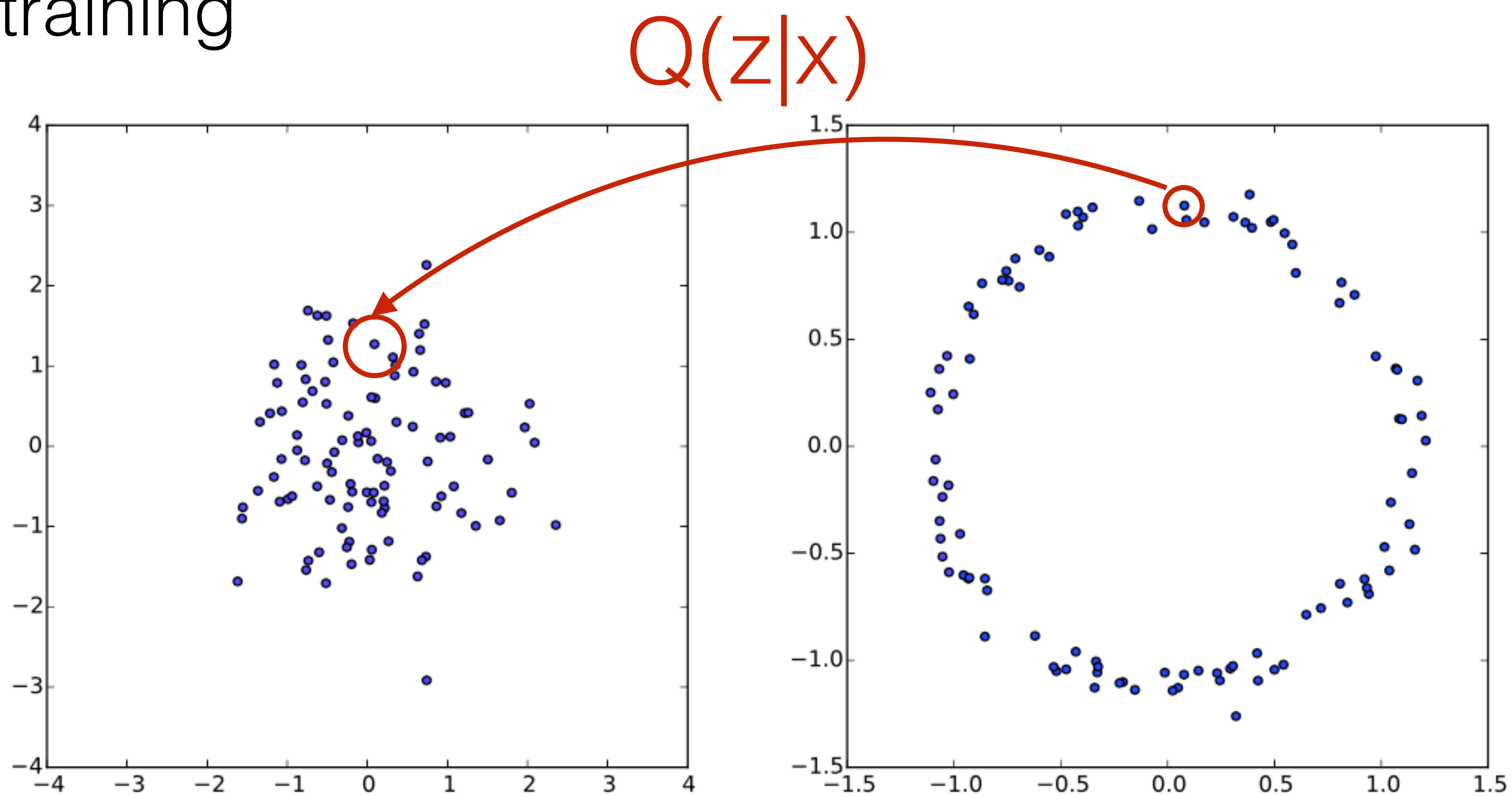




Solution: “Inference Model”

- Predict which latent point produced the data point using inference model Q(**z**|**x**)

- Acquire samples from inference model’s conditional for more efficient training



- Called variational auto-encoder because it “encodes” with the inference model, “decodes” with generative model

Disconnect Between Samples and Objective

- We want to optimize the expectation

$$\begin{aligned} P(\boldsymbol{x};\theta) &= \int P(\boldsymbol{x} \mid \boldsymbol{z};\theta)P(\boldsymbol{z})d\boldsymbol{z} \\ &= \mathbb{E}_{\boldsymbol{z}\sim P(\boldsymbol{z})}[P(\boldsymbol{x} \mid \boldsymbol{z};\theta)] \end{aligned}$$

- But if we sample according to Q, we are actually approximating

$$\mathbb{E}_{\boldsymbol{z}\sim Q(\boldsymbol{z}|\boldsymbol{x};\phi)}[P(\boldsymbol{x} \mid \boldsymbol{z};\theta)]$$

- How do we resolve this disconnect?

VAE Objective

- We can create an optimizable objective matching our problem, starting with KL divergence

$$\mathcal{KL}[Q(\boldsymbol{z} \mid \boldsymbol{x}) || P(\boldsymbol{z} \mid \boldsymbol{x})] = \mathbb{E}_{\boldsymbol{z} \sim Q(\boldsymbol{z}|\boldsymbol{x})}[\log Q(\boldsymbol{z} \mid \boldsymbol{x}) - \log P(\boldsymbol{z} \mid \boldsymbol{x})]$$

Bayes's Rule

$$\mathcal{KL}[Q(\boldsymbol{z} \mid \boldsymbol{x}) || P(\boldsymbol{z} \mid \boldsymbol{x})] = \mathbb{E}_{\boldsymbol{z} \sim Q(\boldsymbol{z}|\boldsymbol{x})}[\log Q(\boldsymbol{z} \mid \boldsymbol{x}) - \log P(\boldsymbol{x} \mid \boldsymbol{z}) - \log P(\boldsymbol{z})] + \log P(\boldsymbol{x})$$

Rearrange/negate

$$\log P(\boldsymbol{x}) - \mathcal{KL}[Q(\boldsymbol{z} \mid \boldsymbol{x}) || P(\boldsymbol{z} \mid \boldsymbol{x})] = \mathbb{E}_{\boldsymbol{z} \sim Q(\boldsymbol{z}|\boldsymbol{x})}[\log P(\boldsymbol{x} \mid \boldsymbol{z})] - \mathbb{E}_{\boldsymbol{z} \sim Q(\boldsymbol{z}|\boldsymbol{x})}[\log Q(\boldsymbol{z} \mid \boldsymbol{x}) - \log P(\boldsymbol{z})]$$

Definition of KL divergence

$$\log P(\boldsymbol{x}) - \mathcal{KL}[Q(\boldsymbol{z} \mid \boldsymbol{x}) || P(\boldsymbol{z} \mid \boldsymbol{x})] = \mathbb{E}_{\boldsymbol{z} \sim Q(\boldsymbol{z}|\boldsymbol{x})}[\log P(\boldsymbol{x} \mid \boldsymbol{z})] - \mathcal{KL}[Q(\boldsymbol{z} \mid \boldsymbol{x}) || P(\boldsymbol{z})]$$

Interpreting the VAE Objective

$$\log P(\boldsymbol{x}) - \mathcal{KL}[Q(\boldsymbol{z} \mid \boldsymbol{x})||P(\boldsymbol{z} \mid \boldsymbol{x})] = \mathbb{E}_{\boldsymbol{z} \sim Q(\boldsymbol{z}|\boldsymbol{x})}[\log P(\boldsymbol{x} \mid \boldsymbol{z})] - \mathcal{KL}[Q(\boldsymbol{z} \mid \boldsymbol{x})||P(\boldsymbol{z})]$$

- Left side is **what we want to optimize**
  - Marginal likelihood of x
  - Accuracy of inference model
- Right side is **what we can optimize**
  - Expectation according to Q of likelihood P(x|z) (approximated by sampling from Q)
  - Penalty for when Q diverges from prior P(z), calculable in closed-form for Gaussians

Problem!
Sampling Breaks Backprop

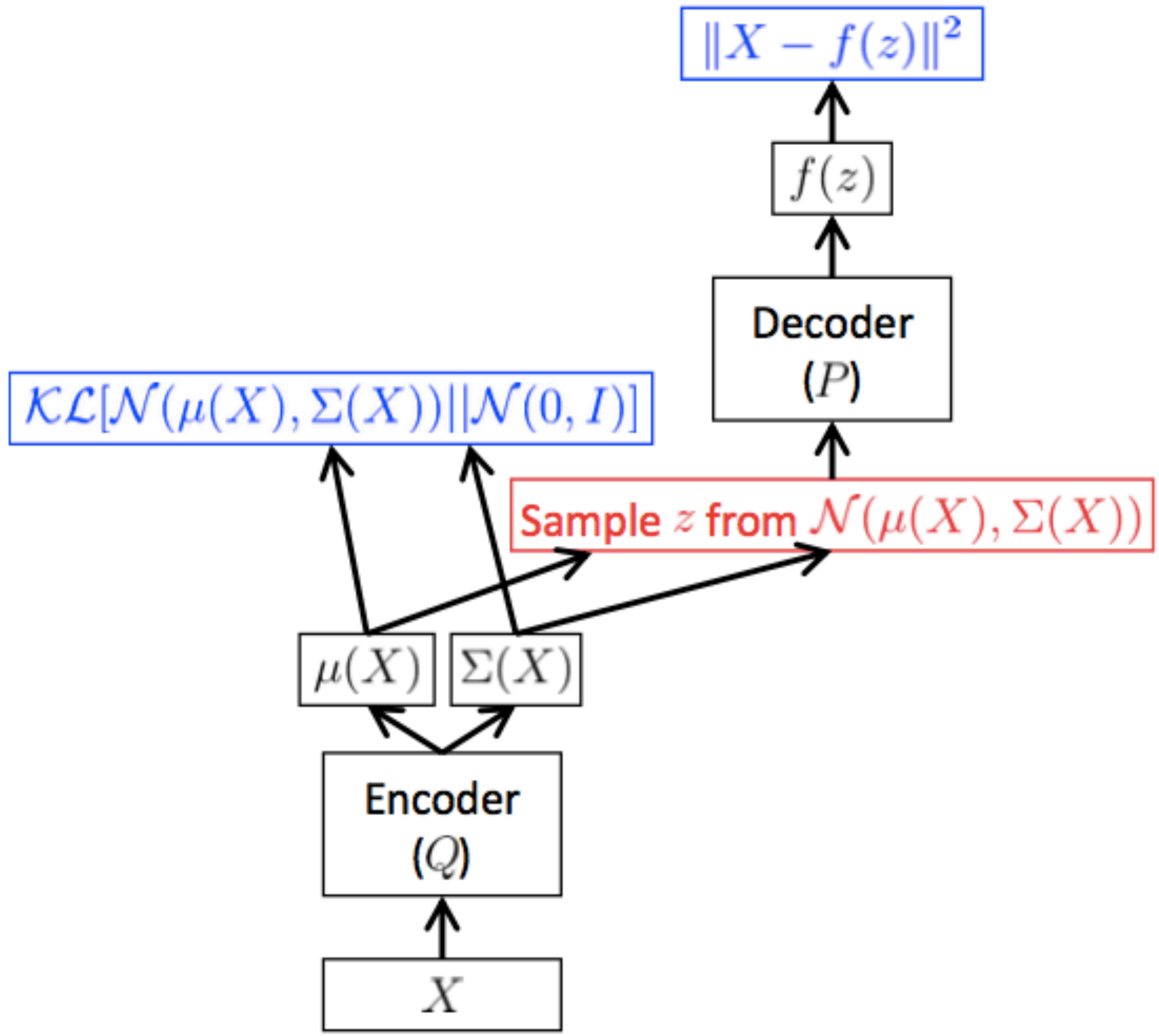


Figure Credit: Doersch (2016)

Solution: Re-parameterization Trick

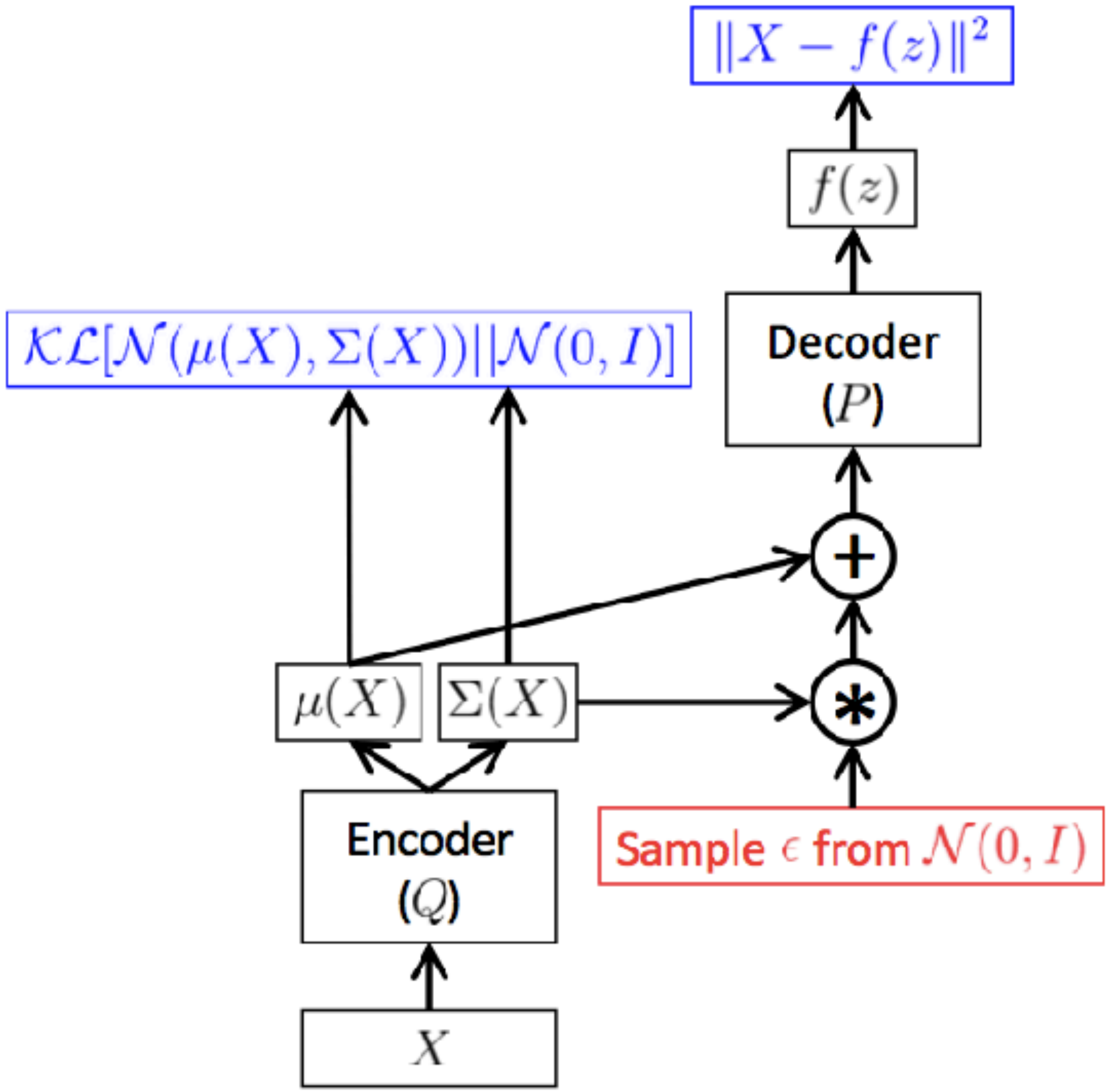


Figure Credit: Doersch (2016)

An Example: Generating Sentences w/ Variational Autoencoders

Generating from Language Models

- **Remember:** using ancestral sampling, we can generate from a normal language model

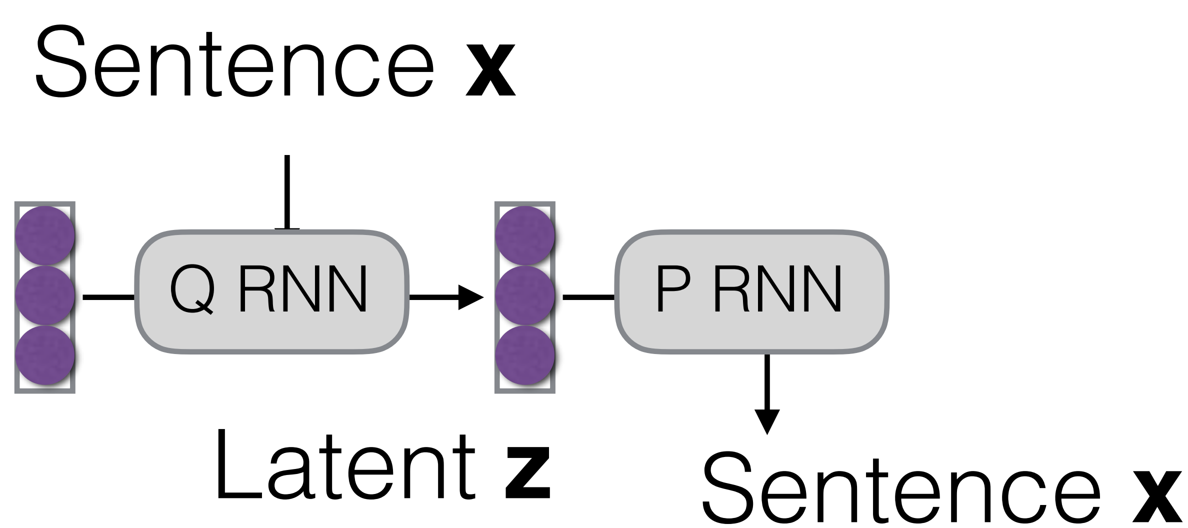
```
while x_{j-1} != "</s>":
  x_j ~ P(x_j | x_1, …, x_{j-1})
```

- We can also generate conditioned on something P(**y**|**x**) (e.g. translation, image captioning)

```
while y_{j-1} != "</s>":
  y_j ~ P(y_j | X, y_1, …, y_{j-1})
```

Generating Sentences from a Continuous Space (Bowman et al. 2015)

- The VAE-based approach is conditional language model that conditions on a latent variable **z**
- Like an encoder-decoder, but latent representation is latent variable, input and output are identical



Motivation for Latent Variables

- Allows for a **consistent latent space** of sentences?
    - e.g. interpolation between two sentences

Standard encoder-decoder

i went to the store to buy some groceries .
i store to buy some groceries .
i were to buy any groceries .
horses are to buy any groceries .
horses are to buy any animal .
horses the favorite any animal .
horses the favorite favorite animal .
horses are my favorite animal .

VAE

" 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 .

- **More robust to noise?** VAE can be viewed as standard model + regularization.

Let’s Try it Out!

`vae-lm.py`

Difficulties in Training

- Of the two components in the VAE objective, the KL divergence term is much easier to learn!

$$\underbrace{\mathbb{E}_{\boldsymbol{z}\sim Q(\boldsymbol{z}|\boldsymbol{x})}[\log P(\boldsymbol{x} \mid \boldsymbol{z})]}_{\text{Requires good generative model}} - \underbrace{\mathcal{KL}[Q(\boldsymbol{z} \mid \boldsymbol{x})||P(\boldsymbol{z})]}_{\text{Just need to set the mean/variance of Q to be same as P}}$$

- Results in the model learning to rely solely on decoder and ignore latent variable

Solution 1: KL Divergence Annealing

- Basic idea: Multiply KL term by a constant λ starting at zero, then gradually increase to 1
- Result: model can learn to use **z** before getting penalized

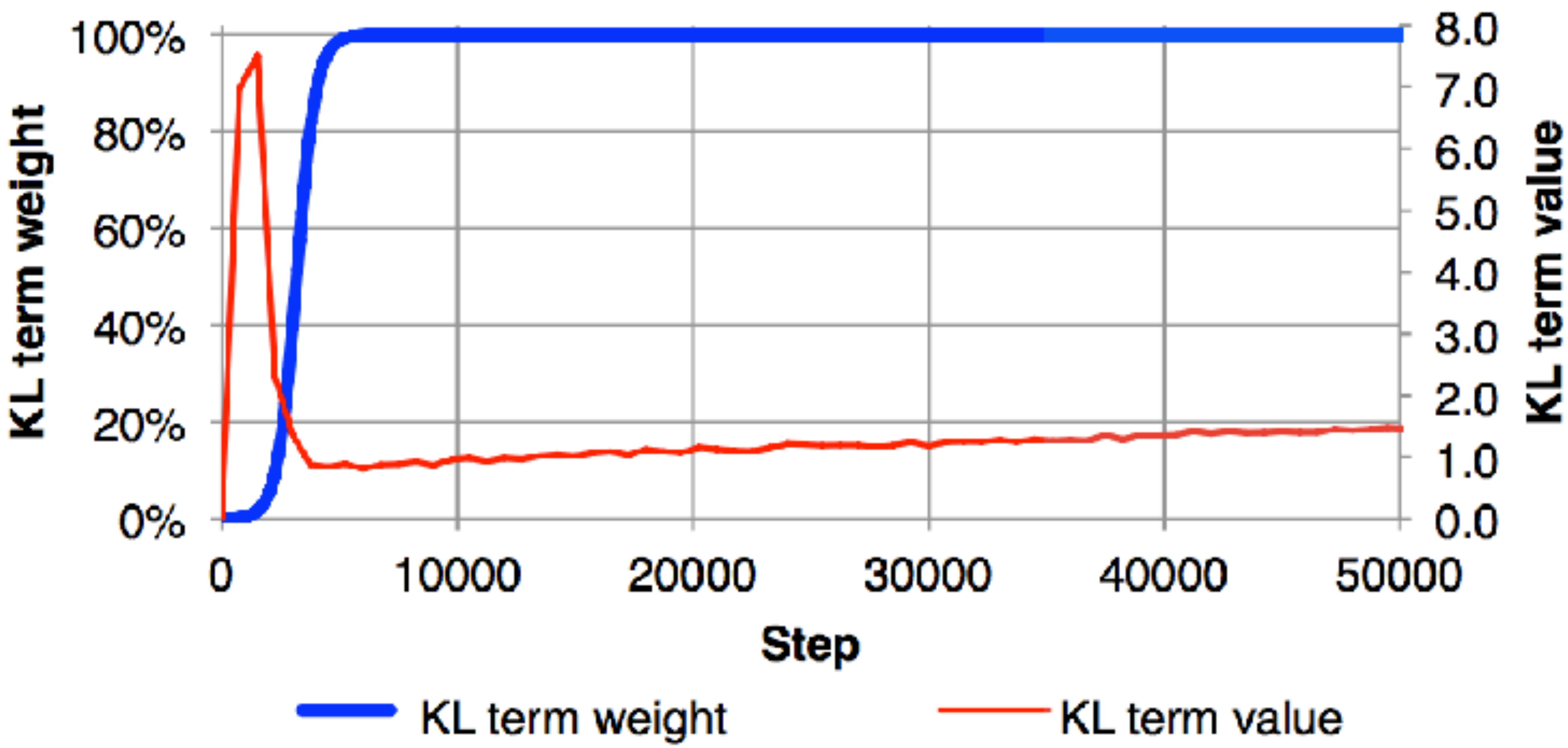


Figure Credit: Bowman et al. (2017)

Solution 2: Weaken the Decoder

- But theoretically still problematic: it can be shown that the optimal strategy is to ignore **z** when it is not necessary (Chen et al. 2017)

- Solution: weaken decoder P(**x**|**z**) so using **z** is essential

  - Use word dropout to occasionally skip inputting previous word in **x** (Bowman et al. 2015)

  - Use a convolutional decoder w/ limited context (Yang et al. 2017)

Handling Discrete Latent Variables

Discrete Latent Variables?

- Many variables are better treated as discrete
  - Part-of-speech of a word
  - Class of a question
  - Speaker traits (gender, etc.)
- How do we handle these?

Method 1: Enumeration

- For discrete variables, our integral is a sum

$$P(\boldsymbol{x};\theta) = \sum_{\boldsymbol{z}} P(\boldsymbol{x} \mid \boldsymbol{z};\theta)P(\boldsymbol{z})$$

- If the number of possible configurations for **z** is small, we can just sum over all of them

Method 2: Sampling

- Randomly sample a subset of configurations of **z** and optimize with respect to this subset
- Various flavors:
  - Marginal likelihood/minimum risk (previous class)
  - Reinforcement learning (next class)
- **Problem:** cannot backpropagate through sampling, resulting in very high variance

Method 3: Reparameterization

(Maddison et al. 2017, Jang et al. 2017)

- Reparameterization also possible for discrete variables!

Original Categorical Sampling Method:

$$\hat{\boldsymbol{z}} = \text{cat-sample}(P(\boldsymbol{z} \mid \boldsymbol{x}))$$

Reparameterized Method

$$\hat{\boldsymbol{z}} = \text{argmax}(\log P(\boldsymbol{z} \mid \boldsymbol{x}) + \text{Gumbel(0,1)})$$

where the Gumbel distribution is

$$\text{Gumbel}(0, 1) = -\log(-\log(\text{Uniform(0,1)}))$$

- Backprop is still not possible, due to argmax

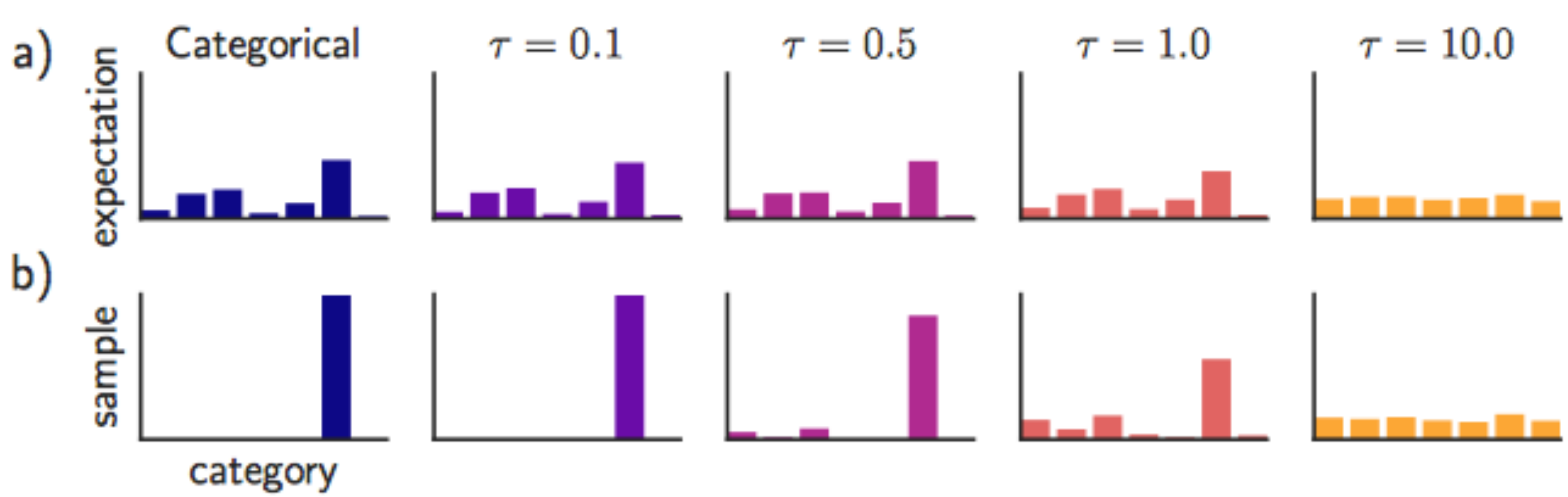
Gumbel-Softmax

- A way to soften the decision and allow for continuous gradients

- Instead of argmax, take softmax with temperature τ

$$\hat{\boldsymbol{z}} = \text{softmax}((\log P(\boldsymbol{z} \mid \boldsymbol{x}) + \text{Gumbel}(0,1))^{1/\tau})$$

- As τ approaches 0, will approach max



Application Examples in NLP

Variational Models of Language Processing (Miao et al. 2016)

- Present models with random variables for document modeling and question-answer pair selection

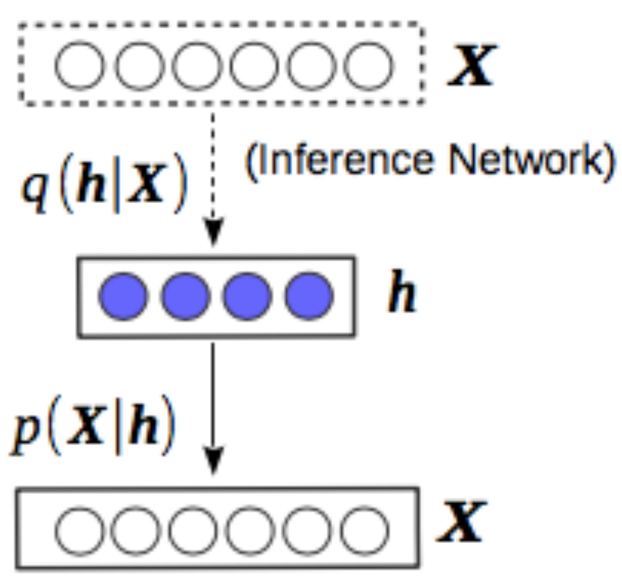


Figure 1. NVDM for document modelling.

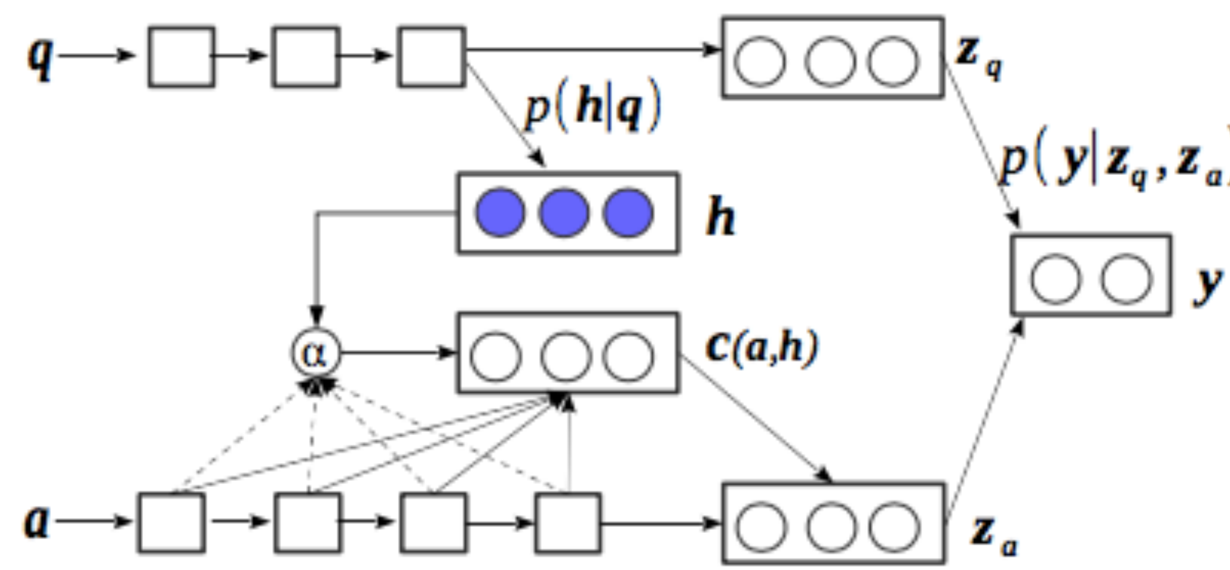


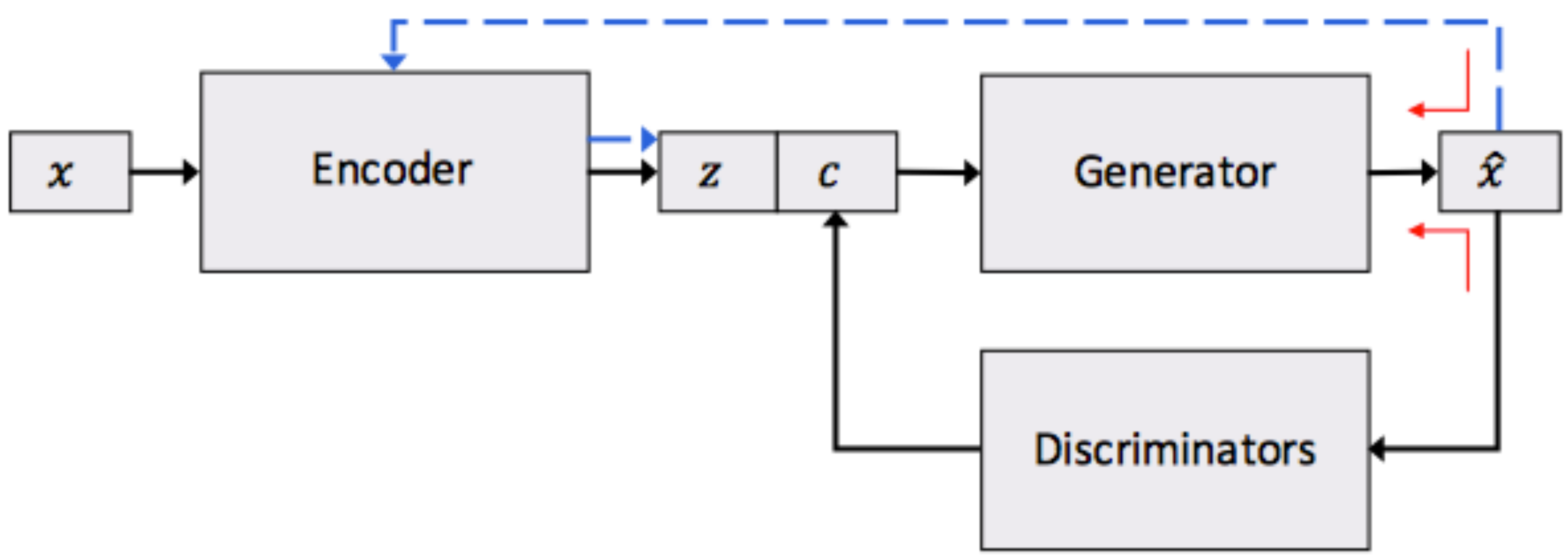
Figure 2. NASM for question answer selection.

- Why random variables? Documents: more consistent space, question-answer more regularization?

Controllable Text Generation

(Hu et al. 2017)

- Creates a latent code **z** for content, and another latent code **c** for various aspects that we would like to control (e.g. sentiment)

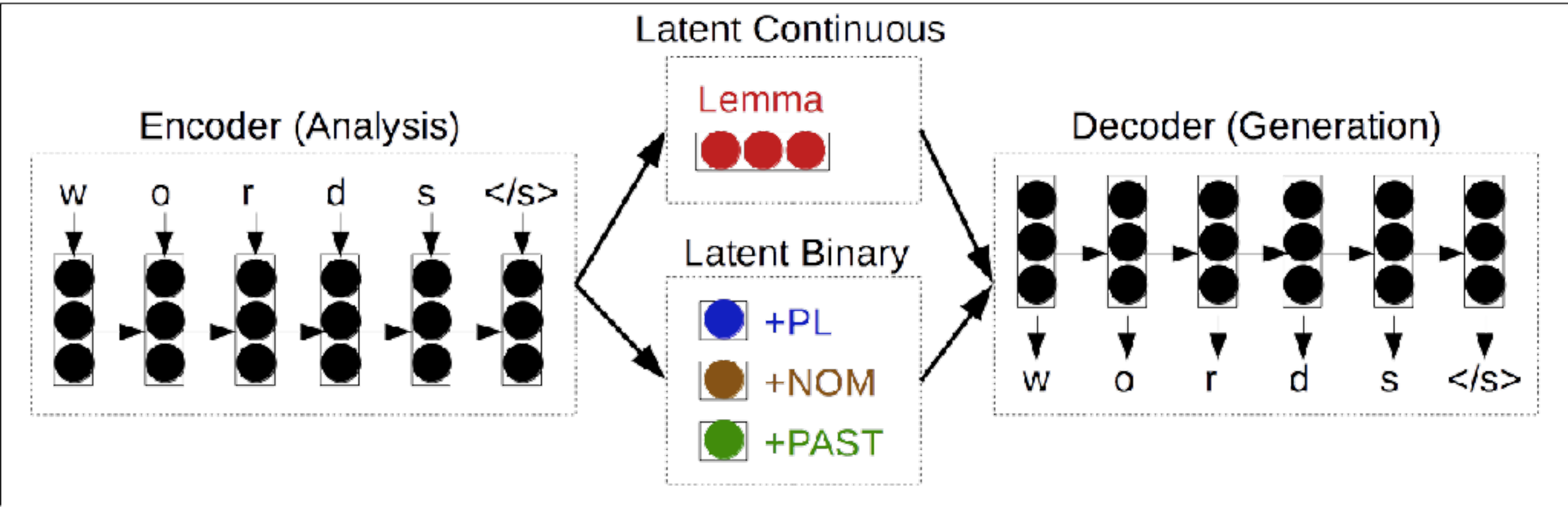


- Both **z** and **c** are continuous variables

Controllable Sequence-to-sequence

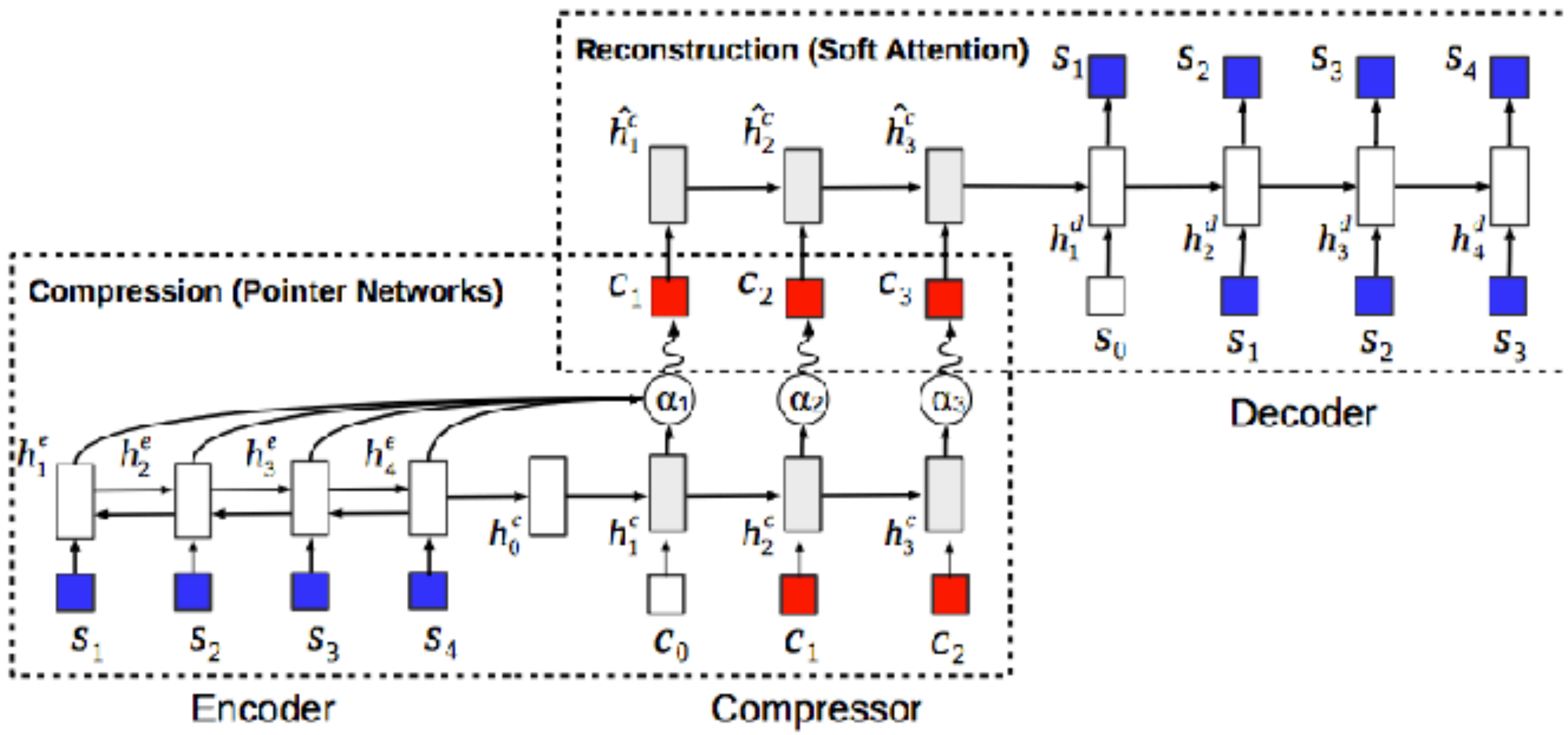
(Zhou and Neubig 2017)

- Latent continuous and discrete variables can be trained using auto-encoding or encoder-decoder objective



Symbol Sequence Latent Variables (Miao and Blunsom 2016)

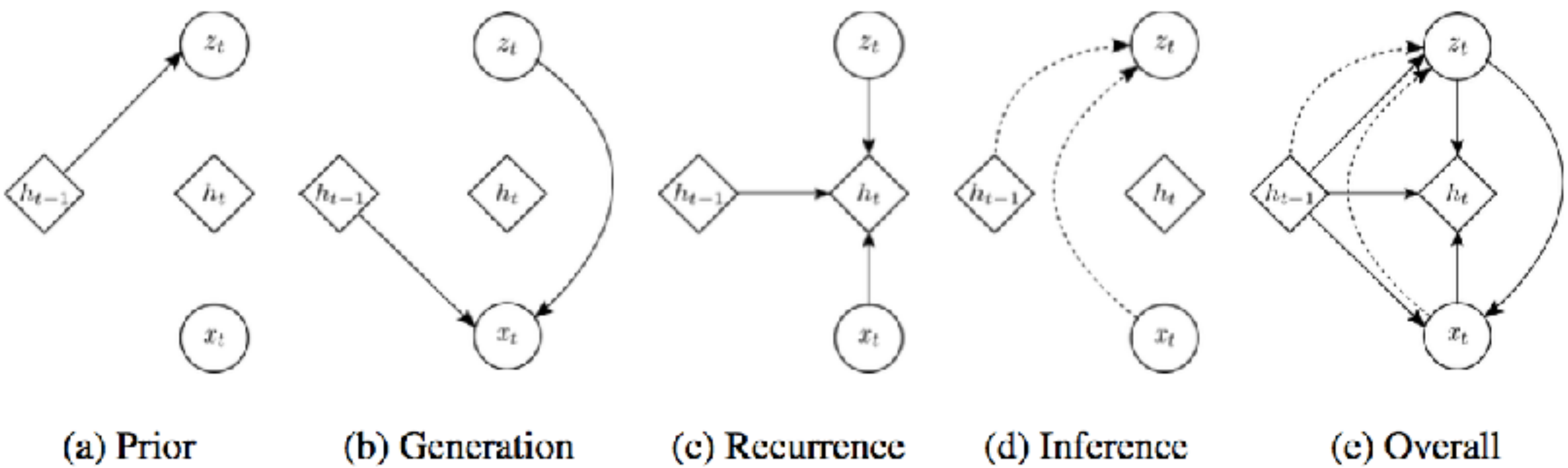
- Encoder-decoder with a sequence of latent symbols



- Summarization in Miao and Blunsom (2016)
- Attempts to "discover" language (e.g. Havrylov and Titov 2017)
  - But things may not be so simple! (Kottur et al. 2017)

Recurrent Latent Variable Models (Chung et al. 2015)

- Add a latent variable at each step of a recurrent model



Questions?