CPSC 532P / LING 530A: Deep Learning for Natural Language Processing (DL-NLP)

Muhammad Abdul-Mageed

muhammad.mageed@ubc.ca

Natural Language Processing Lab

The University of British Columbia

Table of Contents

Autoencoders

Autoencoders: Networks That Copy Their Input to Their Output

- A neural network is trained to copy its input to its output.

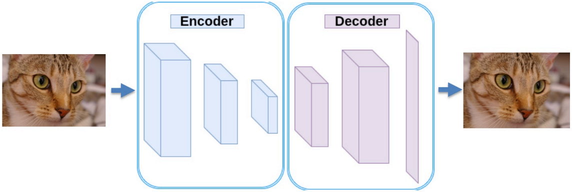

Figure: An Autoencoder.

Autoencoders: Machines With Bottleneck

Autoencoders: Bottleneck in Latent Space Representation

- Has a hidden layer h that describes a **code** representing the input.
- Has an **encoder** function $h = f^{(x)}$ and a **decoder** that produces a reconstruction $r = g^{(h)}$.

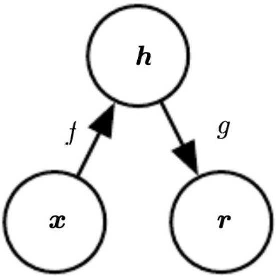

Figure: [Goodfellow et al., 2016]

Autoencoders: Stochastic Mappings

- We are not interested in trivial copying.
- We can force the model to learn **only useful properties** of the data.
- We can do this e.g., by constraining **h** to have smaller dimension than **x** (**undercomplete autoencoder**).
- Autoencoders are **stochastic** mappings between:

$$p_{encoder}(h|x)$$

and

$$p_{decoder}(x|h).$$

Rgularizing Autoencoders

There are at least three methods for preventing AEs from copying:

And Three For The Road...

1. **Limiting the model capacity** by keeping the encoder and decoder shallow
2. **Keeping the code size small** (i.e., using an h with smaller dimension than x)
3. **Regularization**

Variational Autoencoders

VAEs

- **Variational autoencoders** are generative models.
- They now have a wide range of applications such as generating **sound** (e.g., music), **images** (e.g., faces), or **text** (sometimes carrying some style like sentiment).

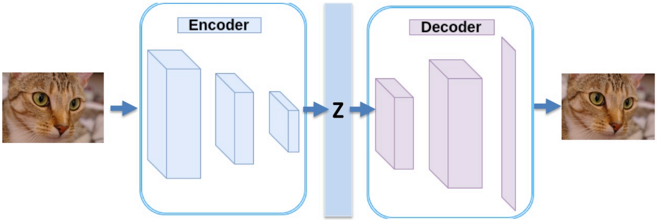

Figure: A variational autoencoder. It has an **encoder** (left) and a **decoder** (right).

Encoder

- The encoder takes an **input** datapoint x and **outputs** a hidden representation z, and has **weights** and biases θ.
- z **has less dimensions than** x (**bottleneck**).
- Encoder needs to **learn good compression of the data**

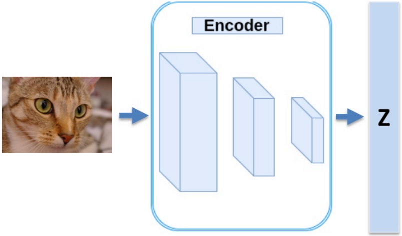

Figure: The **encoder** part of an VAE.

VAE: Encoder II

Encoder

- We denote the **encoder** as $q_\theta(z|x)$.
- The encoder produces **two vectors:** μ (a vector of means) and σ (a vector of standard deviations).
- These two vectors are used to **sample a vector** of random variables of length $n = len(z)$.
- **The** *ith* **element of the sampled encoded vector has the** μ **and** σ **of the corresponding** *ith* **elements** in each of the two vectors.

VAE: Encoder Illustrated

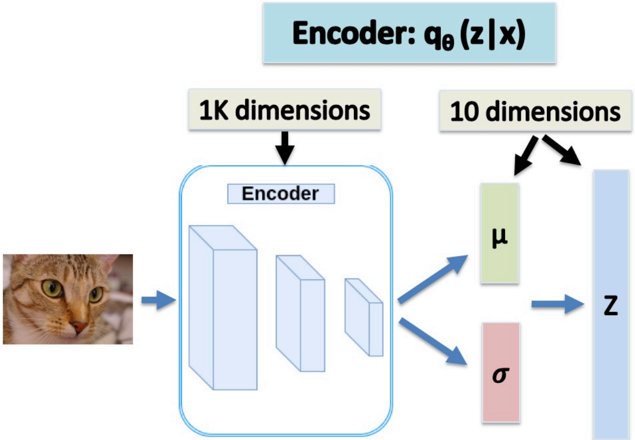

Figure: The encoder works such that it produces **two vectors:** μ (a vector of means) and σ (a vector of standard deviations).

Kullback–Leibler Divergence (KL Divergence)

KL Divergence

- To enforce this, we introduce **KL divergence**.
- **Measures of how one probability distribution is different from a second probability distribution**.
- **Always greater than or equal to zero**: A smaller KL divergence value means we can expect more similar behavior of the two distributions.
- For distributions P and Q of a continuous random variable, **KL divergence is defined as**:

1: KL Divergence

$$D_{KL}(P||Q) = \int_{-\inf}^{\inf} p(x) \log \frac{p(x)}{q(x)} dx$$

How is KL Divergence Useful?

KL Divergence

- Minimizing the KL divergence here means optimizing the probability distribution parameters μ and σ (the encoder's distribution $q_\theta(z|x)$) to closely resemble that of the target distribution ($p(x)$) (In our case, represented as a prior over the latent variable / the model prior ($p(z)$).

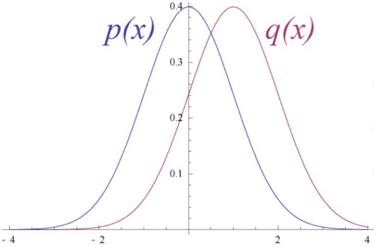

Figure: [From Wikipedia].

VAE: Decoder

Decoder

- The decoder takes as input the representation z, and outputs the **parameters** to the probability distribution of the data ϕ.

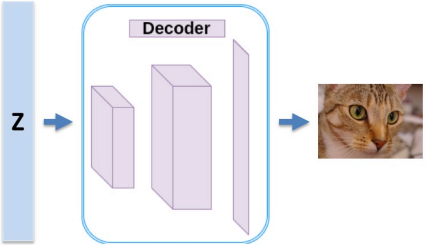

Figure: The **decoder** part of an VAE.

VAE: Putting it All Together

Loss

- **Information is lost** as we generated from the compressed data (z).
- No global representations shared across all data points, so we can **decompose the loss l of the whole dataset into several losses** (each as l_i).
- Recall: **encoder** denotes as $q_\theta(z|x)$, and the **decoder** as the joint likelihood of the visible and hidden variables $p_\phi(x|z)$.
- For loss, we use the reconstruction **joint log-likelihood** $logp_\phi(x|z)$ under the **approximate posterior over the latent variables** $\mathbb{E}_z \sim q(z|x)$.
- As mentioned, we also use **KL divergence** between the encoder's distribution $q_\theta(z|x)$ and $p(z)$.

VAE: Loss Function

2: VAE Loss

$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i|z)] + KL(q_\theta(z|x_i)||p(z)).$$

- The **first term** encourages the decoder to reconstruct the data, causing it to incur cost if it doesn't achieve good construction.
- **The Kullback-Leibler** divergence measures how much info. is lost when using q to represent p.

(Variational) Auto-Encoders for Sentiment and Affect

Applications of VAE to SAA

- **Semi-supervised learning** (e.g., Dai et al., 2015)
- **Controlled text generation** (sentiment-carrying text) (e.g., Hu et al., 2018)

Semi-supervised learning of Sentiment With VAE

- Dai et al. (2015) train a **sequence auto-encoder** on two sentiment datasets (IMDB and Rotten Tomatoes).
- The auto-encoder predicts its own input and as a by-product the network has **weights** which they use to initialize a supervised sentiment classifier (LSTM).

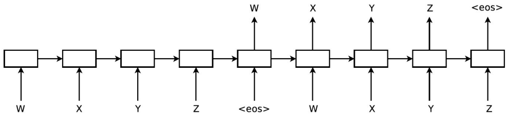

Figure 1: The sequence autoencoder for the sequence "WXYZ". The sequence autoencoder uses a recurrent network to read the input sequence in to the hidden state, which can then be used to reconstruct the original sequence.

Figure: [From Dai et al., 2015].

Sequence Autoencoders - LSTMs (SA-LSTMs)

- Found SA-LSTMs to outperform ConvNets and paragraph vectors (previous SOTA).

Table 1: A summary of the error rates of SA-LSTMs and previous best reported results.

Dataset	**SA-LSTM**	**Previous best result**
IMDB	7.24%	7.42%
Rotten Tomatoes	16.7%	18.5%
20 Newsgroups	15.6%	17.1%
DBpedia	1.19%	1.74%

Figure: [From Dai et al., 2015].

SA-LSTMs > LSTMs

- Also found SA-LSTMs to be more stable to train than LSTMs.
- For example, **increasing hidden units causes overfitting with LSTMs but not with SA-LSTMs** (which seem better suited to long documents).

Table 2: Performance of models on the IMDB sentiment classification task.

Model	Test error rate
LSTM with tuning and dropout	13.50%
LSTM initialized with word2vec embeddings	10.00%
LM-LSTM (see Section 2)	7.64%
SA-LSTM (see Figure 1)	7.24%
SA-LSTM with linear gain (see Section 3)	9.17%
SA-LSTM with joint training (see Section 3)	14.70%
Full+Unlabeled+BoW [22]	11.11%
WRRBM + BoW (bnc) [22]	10.77%
NBSVM-bi (Naïve Bayes SVM with bigrams) [36]	8.78%
seq2-bow*n*-CNN (ConvNet with dynamic pooling) [12]	7.67%
Paragraph Vectors [19]	7.42%

Figure: [From Dai et al., 2015].

Controlled Sentiment Generation

- Hu et al. (2018): "Toward Controlled Generation of Text"

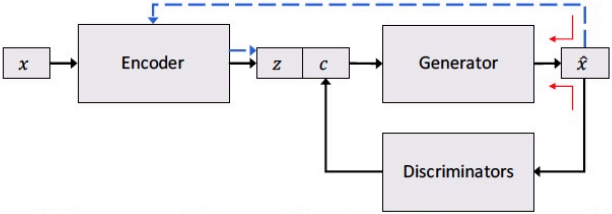

Figure 1. The generative model, where z is unstructured latent code and c is structured code targeting sentence attributes to control. Blue dashed arrows denote the proposed independency constraint (section 3.2 for details), and red arrows denote gradient propagation enabled by the differentiable approximation.

Figure: [From Hu et al., 2018].