# Final

Saturday, December 18, 2021 5:57 PM

Tim Demetriades 12/19/2021

## EE628WS Final

**Name:**

Q1.
- What are the main tasks that autoencoders are used for?
- What are the advantages for the attention mechanism?

•Autoencoders are a type of unsupervised neural network that can be used to learn an encoding of a set of data & then reconstruct that data to be as close as possible to the original input data. It is a form of dimensionality reduction where the encoded representation is of a much lower dimension & therefore size compared to the original. In this way it can be used to represent the data (such as an image) in a more compact way. One application of an autoencoder is for de-noising, where an image with noise is encoded & then the reconstructed image will have a majority of the noise removed. Autoencoders can also be used for pretraining a model to help improve performance.

•The attention mechanism is used in Encoder-Decoder networks (a variant of RNNs) to solve some of its limitations. Some of the limitations include the fact that the only connections between encoder & decoder are the fixed length semantic vectors, the fact that the encoder has to convert the information of the whole sequence set into a fixed length semantic vector, & the fact that the semantic vector may not carry the full information from the sequence. The attention mechanism is able to solve these limitations by utilizing inputting different semantic vectors at different time slots, & each one will automatically select the best context information for the current output. In this way it is able to enhance some parts of the input while diminishing other parts to help the network focus of the most important parts of the data.

Q2.
- What is difference between PCA and auto-encoder?
- Why does a variational auto-encoder look for a Gaussian distribution for the latent factors?

• Although similar, there are several differences between PCA + auto-encoders. for one, auto-encoders are much more flexible than PCA. Also, PCA can only represent linear transformations, whereas since auto-encoders are neural networks + neural networks contain activation functions, the encoding can contain non-linearities. Lastly, since auto-encoders are neural networks, they can be stacked to form a deep network unlike with PCA.

• Variational auto-encoders are similar to vanilla auto-encoders, but instead of directly producing a coding for a given input, the encoder produces a mean coding + standard deviation which are then sampled randomly from a Gaussian distribution. By using Gaussian distributions VAE's are able to model the true distribution $P(z|X)$, where $z$ is the latent variable, as a simpler distribution that is easier to evaluate as a result of the many nice properties of Gaussian distributions. Some of these properties include being able to measure the KL divergence in the variational loss + being able to use the 'reparametrization trick' for efficient gradient computation. Furthermore, one VAE's allow generation of new samples by sampling

the 'reparametrization trick' for efficient gradient computation. Furthermore, one VAE's allow generation of new samples by sampling in the latent space, which is much easier if it follows a Gaussian distribution.



Q3.
- What are the differences between variational auto-encoder with vanilla auto-encoder?
- What are the two parts of cost function with variational auto-encoder? Explain the meanings with these cost function losses.
- What is KL divergence metric for? What is the formula for KL divergence?

- Although similar to a vanilla auto-encoder, a variational auto-encoder adds a hidden coding layer by instead of directly producing a coding for a given input, it produces a mean coding & standard deviation. The actual coding is then sampled randomly from a Gaussian distribution. After that the decoder just decodes the sampled part normally, similar to a vanilla auto-encoder. A VAE attempts to find the parameters (mean & standard deviation) from the underlying distribution of the input data.
- For a variational auto-encoder, the cost function consists of 2 parts. First is the reconstruction loss: $\Delta = \|x_i - \tilde{x}_i\|^2$. This is how similar the auto-encoder's output was to its input. A vanilla auto-encoder also uses this part. However, with a VAE, there is also the latent loss: $(\mu + \sigma \sim 2) \sim N(0, I)$. This is how close its hidden nodes were to a normal distribution. The goal of a VAE is to minimize these 2 cost functions, but there is a trade-off between the

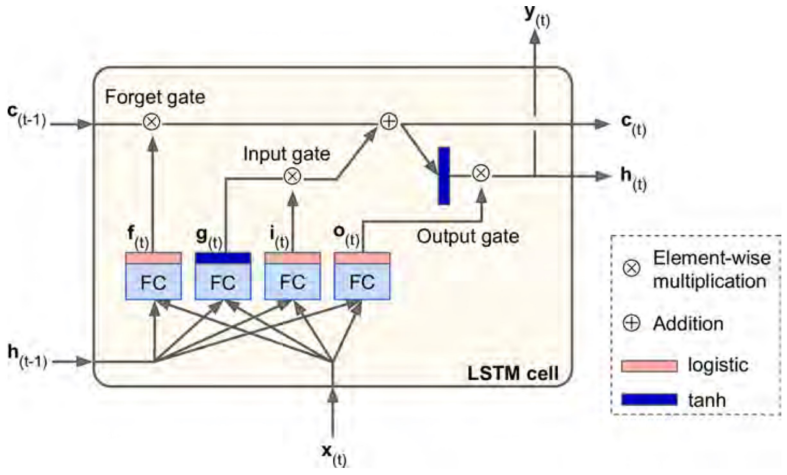
how close its hidden nodes were to a normal distribution. The goal of a VAE is to minimize these 2 cost functions, but there is a trade-off between the 2 & therefore a nice equilibrium needs to be found.

- The formula for KL divergence is:

$$D_{KL}[Q(z|X)||P(z|X)] = \underset{\text{expectation}}{E}[\log Q(z|X) - (\log P(X|z) + \log P(z)) + \log P(X)]$$

Put simply, it is the difference of one probability distribution compared to another one. It is used to minimize the amount of information loss we have when approximating a distribution, which is what a variational auto-encoder does.






Q4. a Why do people use encoder-decoder RNNs rather than plain sequence-to-sequence RNNs for automatic translation?

b What are the word embeddings? Why do people use this?

c For above LSTM cell, explain the functions of every gate, input and output items.

- Rather than directly inputing the input sequence to the output sequence when using plain sequence-to-sequence RNNs, with Encoder-Decoder RNNs we are converting the input data into a context semantic vector, which is the word embedding. This is a form of preprocessing that lowers the dimensions of the data into something that is more meaningful. This allows the model to correctly correlate different vectors in order to group similar data (like words) together.

- Word embeddings are used to group similar words together. With word embeddings, words are represented by real-valued vectors in a pre-defined vector space. Each word is mapped to one vector which are then learned. It is a form of dimensionality reduction. One use is with encoder-decoder RNNs, where they allow the model to be able to handle different sequences of varying lengths.

- The LSTM consists of 3 gates. First is the input gate which controls which parts of $g_{(t)}$ should be added to the long-term state. Next is the forget gate, which controls which parts of the long-term state should be erased, essentially deciding which information can be ignored + which need attention. Finally, there is the output gate, which controls which parts of the long-term state should be read + output at this time to $h_{(t)}$ + $y_{(t)}$.

Inputs: $x_{(t)}$ = current input, $h_{(t-1)}$ = previous short term state, $c_{(t-1)}$ = previous long term state
Outputs: $y_{(t)} = h_{(t)}$ = current short term state output, $c_{(t)}$ = current long term state

Q5. a An attention model is a typical "word embedding" or "seq2seq" model?

b Consider a task with input sequence of fixed length. Could RNN architecture still be useful for such task?

c Consider an RNN for a language generation task. $y_t$ is an output of this RNN at each time step, $L$ is a length of the input sequence, $N$ is a number

ture still be useful for such task?

c Consider an RNN for a language generation task. $y_t$ is an output of this RNN at each time step, $L$ is a length of the input sequence, $N$ is a number of words in the vocabulary.

1. $y_t$ is a vector of length $N$.
2. $y_t$ is a vector of length $(L - t)$.
3. $y_t$ is a vector of length $L \times N$.
4. Each element of $y_t$ is either 0 or 1.
5. Each element of $y_t$ is a number from 0 to 1.
6. Each element of $y_t$ is a number from 0 to $N$.

- An attention model is a 'seq2seq' model. This is because it is mostly used for task such as machine translation where sequences are generated based on an input sequence.

- Although an RNN architecture could be used for a task with in input sequence of fixed length, there are other architectures that could also be used. For instance, if we are trying to make an output sequence based on nearby words a CNN can be used since there is a fixed length input sequence. However, RNNs would still be useful for this task even though they strive with variable length input sequences.



Q6. The following equation captures the computation in a ResNet block. What goes into the two blanks above (where $g(\cdot)$ is the activation function)?

$a[l+2] = g(W[l+2] * g(W[l+1]a[l] + b[l+1]) + b[l+2] + \underline{0}) + \underline{a[l]}$

Q6. The following equation captures the computation in a ResNet block. What goes into the two blanks above (where $g(\cdot)$ is the activation function)?

$a[l+2] = g\left(W[l+2] * g\left(W[l+1]a[l] + b[l+1]\right) + b[l+2] + \underline{0}\right) + \underline{a[l]}$

(a) $a[l]$ and 0, respectively

(b) 0 and $z[l+1]$, respectively

(c) $z[l]$ and $a[l]$, respectively

(d) 0 and $a[l]$, respectively

- W[l+2] = weights conv 2
- W[l+1] = weights conv 1
- a[l] = previous output
- b[l+1] = bias for conv 1
- b[l+2] = bias for conv 2

Q7. Which ones of the following statements on Inception Networks are true? (Check all that apply.)

(a) Inception networks incorporates a variety of network architectures (similar to dropout, which randomly chooses a network architecture on each step) and thus has a similar regularizing effect as dropout.

(b) Inception blocks usually use 1x1 convolutions to reduce the input data volumes size before applying 3x3 and 5x5 convolutions.

(c) Making an inception network deeper (by stacking more inception blocks together) should not hurt training set performance.

(d) A single inception block allows the network to use a combination of 1x1, 3x3, 5x5 convolutions and pooling.