Lecture 15 Quiz

Quiz, 7 questions

1 point

1.
The objective function of an autoencoder is to reconstruct its input, i.e., it is trying to learn a function f, such that $f(x) = x$ for all points x in the dataset. Clearly there is a trivial solution to this. f can just copy the input to the output, so that $f(x) = x$ for all x. Why does the network not learn to do this ?

- The network has constraints, such as bottleneck layers, sparsity and bounded activation functions which make the network incapable of copying the entire input all the way to the output.
- Since all the hidden units in an autoencoder are linear, the model is not powerful enough to learn the identity transform, unless the number of hidden units in each layer is not less than the number of input dimensions.
- Optimization algorithms that are used to train the autoencoder are not exact. So the network cannot easily learn to copy the input to the output.
- The objective function used to train an autoencoder is to minimize reconstruction error if x lies in the dataset but maximize it if x does not belong to the dataset. This prevents it from copying the input to the output for all x.

1 point

2.
The process of autoencoding a vector seems to lose some information since the autoencoder cannot reconstruct the input exactly (as seen by the blurring of reconstructed images reconstructed from 256-bit codes). In other words, the intermediate representation appears to have less information than the input representation. In that case, why is this intermediate representation more useful than the input representation ?

- The intermediate representation may take much less memory to store and much less time to do comparisons with compared to the input. That makes matching queries to a database faster.
- The intermediate representation is more compressible than the input representation.
- The intermediate representation actually has more information than the inputs.
- The intermediate representation has more noise.

1 point

3.
What are some of the ways of regularizing deep autoencoders?

- Using large minibatches for stochastic gradient descent.
- Using a squared error loss function for the reconstruction.
- Using high learning rate and momentum.
- Adding noise to the inputs.

Lecture 15 Quiz

Quiz, 7 questions

1
point

4.
In all the autoencoders discussed in the lecture, the decoder network has the same number of layers and hidden units as the encoder network, but arranged in reverse order. Brian feels that this is not a strict requirement for building an autoencoder. He insists that we can build an autoencoder which has a very different decoder network than the encoder network. Which of the following statements is correct?

- ○ Brian is correct, as long as the decoder network has **at most** as many parameters as the encoder network.
- ○ Brian is mistaken. The decoder network must have the same architecture. Otherwise backpropagation will not work.
- ○ Brian is correct. We can indeed have any decoder network, as long as it produces output of the same shape as the data, so that we can compare the output to the original data and tell the network where it's making mistakes.
- ○ Brian is correct, as long as the decoder network has the same number of parameters as the encoder network.

1
point

5.
Another way of extracting short codes for images is to hash them using standard hash functions. These functions are very fast to compute, require no training and transform inputs into fixed length representations. Why is it more useful to learn an autoencoder to do this ?

- ○ Autoencoders have several hidden units, unlike hash functions.
- ○ For an autoencoder, it is possible to invert the mapping from the hashed value to the reconstruct the original input using the decoder, while this is not true for most hash functions.
- ○ Autoencoders have smooth objective functions whereas standard hash functions have no concept of an objective function.
- ○ Autoencoders can be used to do **semantic** hashing, where as standard hash functions do not respect semantics , i.e, two inputs that are close in meaning might be very far in the hashed space.

1
point

6.
RBMs and single-hidden layer autoencoders can both be seen as different ways of extracting one layer of hidden variables from the inputs. In what sense are they different ?

- ☐ The objective function and its gradients are intractable to compute exactly for RBMs but can be computed efficiently exactly for autoencoders.
- ☐ RBMs work only with binary inputs but autoencoders work with all kinds of inputs.
- ☐ RBMs are undirected graphical models, but autoencoders are feed-forward neural nets.
- ☐ RBMs define a probability distribution over the hidden variables conditioned on the visible units while autoencoders define a deterministic mapping from inputs to hidden variables.

Lecture 15 Quiz

1 point

Quiz, 7 questions

7.
Autoencoders seem like a very powerful and flexible way of learning hidden representations. You just need to get lots of data and ask the neural network to reconstruct it. Gradients and objective functions can be exactly computed. Any kind of data can be plugged in. What might be a limitation of these models ?

- ○ The hidden representations are noisy.
- ○ There is no simple way to incorporate uncertainty in the hidden representation $h = f(v)$. A probabilistic model might be able to express uncertainty better since it is being made to learn $P(h|v)$.
- ○ Autoencoders cannot work with discrete-valued inputs.
- ○ The inference process for finding states of hidden units given the input is intractable for autoencoders.