Lecture 15 Quiz

Quiz, 7 questions

7/7 points (100%)

✓ **Congratulations! You passed!**

Next Item

✓ 1 / 1 points

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.

It makes sense to learn a mapping from x to some target t for solving a classification or regression problem, but why do we want to learn a mapping that takes x to x ? It seems like a silly thing to do!

[✓ While the objective is to reconstruct x, what we really care about is learning good representations in the hidden layers which come as a by-product of doing this.]

Correct

This is the most important reason why we want to learn autoencoders.

- Due to the constraints in the network, x will never be mapped exactly to x but to something close-by, say $\hat{x}$. We can then use $\hat{x}$ as a better representation of x.
- While the objective is to learn f such that $f(x) = x$, we might end up learning a mapping to a totally different space in which $f(x)$ might be a much better representation of the input.
- We want to have a decoder network, so that it is easy to get a reconstruction of the data.

✓ 1 / 1 points

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 is more compressible than the input representation.

○ The intermediate representation actually has more information than the inputs.

[✓ 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.]

Correct

○ The intermediate representation has more noise.

✓ 1 / 1 points

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

○ Using high learning rate and momentum.

[✓ Adding noise to the inputs.]

Correct

○ Using a squared error loss function for the reconstruction.

○ Using large minibatches for stochastic gradient descent.

✓ 1 / 1 points

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 least** 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.]

Correct
An autoencoder is just a neural net trying to reconstruct its input. There is hardly any restriction on the kind of encoder or decoder to be used.

- ○ Brian is correct, as long as the decoder network has the same number of parameters as the encoder network.

✓ 1 / 1 points

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.
- [✓ Autoencoders are smooth functions and map nearby points in input space to nearby points in code space. They are also invariant to small fluctuations in the input. In this sense, this hashing is locality-sensitive, whereas general hash functions are not locality-sensitive.]

○ Autoencoders have smooth objective functions whereas standard hash functions have no concept of an objective function.

○ 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.

✓ 1 / 1 points

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.

Correct
The objective function for RBMs is log-likelihood. This is intractable to compute due to the partition function. Its gradients are hard to compute for similar reasons and computing them approximately requires CD or other MCMC methods. Autoencoders usually have tractable objectives such as squared loss or cross entropy which are easy to compute and differentiate.

☐ 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.

Correct
RBMs define a joint density $P(v, h)$. Given v, we can compute $P(h|v)$. Autoencoders define a function $h = f(v)$.

☐ RBMs are undirected graphical models, but autoencoders are feed-forward neural nets.

Correct

☐ RBMs work only with binary inputs but autoencoders work with all kinds of inputs.

Un-selected is correct

✓ 1 / 1 points

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 ?

[✓ If the input data comes with a lot of noise, the autoencoder is being forced to reconstruct noisy input data. Being a deterministic mapping, it has to spend a lot of capacity in modelling the noise in order to reconstruct correctly. That capacity is not being used for anything semantically valuable, which is a waste.]

Correct
Note that this is different from denoising autoencoders. In that case, clean inputs are available which are used as targets. The noise is only in the inputs.

○ Autoencoders cannot work with discrete-valued inputs.

○ The inference process for finding states of hidden units given the input is intractable for autoencoders.

○ The hidden representations are noisy.