Scalable Machine Learning and Deep Learning
Review Questions 5

Group 21 - Marco

Boffo Marco
Dei Rossi Marco

December 5, 2020

Abstract

In the following document, you can find the answers to the Research Questions 5 Assignment.

QUESTION 1 (1 point)

Assume we have a stacked autoencoder with three hidden layers $h1$, $h2$, and $h3$, in which each layer applies the following functions respectively, $h1 = f1(x)$, $h2 = f2(h1)$, and $h3 = f3(h2)$, and the output of the network will be $y = f4(h3)$. Do you think it is a good autoencoder if it generates $f4(f3(f2(f1(x)))) = x$ for all input instances x. How can we improve it?

A stacked-autoencoder (or deep-autoencoder) neural network model (SAE model) is an unsupervised learning network composed of multiple layers of sparse autoencoders. If it generates, as output, the same instance used in input, there is a great risk that the network is just focusing on copying the input and not on finding patterns in the data. Having an autoencorder that generates the same output as the input, could be desirable although may sound useless. We are typically not interested in the output of the decoder. Doing so, we hope that training the autoencoder to perform the input copying task will result in the initial encoder taking on useful properties.

A way to force the autoencoder to learn useful features is to add gaussian noise to its inputs and training it to recover the original noise-free inputs. Moreover, we can also implement a dropout effect by adding an appropriate term to the cost function to push the autoencoder to reducing the number of active neurons in the coding layer.

QUESTION 2 (1 point)

How does Gibbs sampling work? When do we need to use Gibbs sampling?

The Gibbs Sampling is a Monte Carlo-Markov Chain method for obtaining a sequence of observations which are approximated from a specified multivariate probability distribution. In Restricted Boltzmann Machines direct sampling is difficult because it is non trivial to calculate the joint probability due to the huge number of possible combination.

Much easier is the calculation of the conditional probabilities of hidden state given the visible state and vice versa. Gibbs Sampling iteratively draws an instance from the distribution of each variable, conditional on the current values of the other variables in order to estimate complex joint distributions. Each random variable is iteratively resampled from its conditional distribution given the remaining variables.

Gibbs sampling is applicable when the joint distribution is not known explicitly, but the conditional distribution of each variable is known. The Gibbs sampling algorithm is used to generate an instance from the distribution of each variable in turn, conditional on the current values of the other variables.

QUESTION 3 (1 point)

How do you tie weights in a stacked autoencoder? What is the point of doing so?

When the autoencoder is symmetrical, it is a common practice to use tying weights. In a symmetric architecture, we can tie the weights of the decoder layers of the weights of the encoder layers.

This is nothing but tying the weights of the decoder layer to the weights of the encoder layer. To implement tying weights, we need to create a custom layer to tie weights between the layer. This custom layer acts as a regular dense layer, but it uses the transposed weights of the encoder's dense layer, however having its own bias vector.

This reduces the number of weights of the model almost to half of the original, thus reducing the risk of over-fitting and speeding up the training process.

QUESTION 4 (1 point)

What are minibatch standard deviation layers? Why will they help training GANs?

Generative adversarial networks has a tendency to capture only little variation from training data. Sometimes all input noise vectors generate similar looking images. This problem is also known as 'mode collapse'. To add a little variation to generated images, authors of the progressive gans have used minibatch standard deviation.

Here standard deviation of each feature in the activation map is calculated and then averaged over the specific minibatch to get a single value. Through this new activation, an extra feature map is added to each instance in the batch and filled with the computed value. These new activation maps are added at the end of the discriminator network.

This idea is related to the lack of variation evident in many GAN models. Thanks to this new layer, the discriminator will know if the generator produced a batch with a small standard deviation across the feature maps. This will encourage the generator to produce more diverse outputs, reducing the risk of mode collapse.

QUESTION 5 (1 point)

What is Nash Equilibrium? How does it relate to GANs?

In Game theory, a Nash Equilibrium is reached where each player does not want to change their actions given the other players actions, since no player has anything to gain by changing only their own strategy.

GAN is based on the zero-sum non-cooperative game. It converges when the discriminator and the generator reach a Nash equilibrium, a state in which nor the generator or the discriminator can improve its individual gain by choosing a different strategy.

For a generator expressive enough to reproduce the distribution of observed samples, Nash equilibrium exists for that generator function producing the data distribution, and the discriminator having to choose between the real and fake samples with a 50-50 probability.

Nash equilibrium may not always exist in GAN games, i.e. there is no guarantee that the equilibrium will ever be reached.