107061112 王昊文 2021/12/31

COM526000 Deep Learning Assignment 3

1. Consider the following variant of radial basis function networks in which the hidden units take on either 0 or 1 values. The hidden unit takes on the value of 1, if the distance to a prototype vector is less than σ. Otherwise it takes on the value of 0. Discuss the relationship of this method to RBF networks, and its relative advantages/ disadvantages.

   Solution. This kind of kernel function is similar to a step function. The advantage of using a step function is that it is computationally cheap, as compared to calculating the Gaussian RBF kernel function, which contains exponential computaions. However, the disadvantage is that if our prototype vector is not well-chosen, that the distance from a given point in the dataset to all the prototype vectors are less than σ, that will make the activations to the linear layers all be 0, this usually leads to the network not be able to make a prediction.

2. Discuss why an RBF network is a supervised variant of a nearest-neighbor classifier.

   Solution. For example, we can make the prototype vectors be some data from the training set. The distance function is the RBF function, we can think of it as the distance function from the nearest-neighbor classifier. At the last layer we simply use a dense layer to learn the weights from each prototype vector neurons, this is equivalent to doing a weighted voting on nearest-neighbor classifier, but the voting procedure is learned in a supervised way.

3. Suppose that you change your RBF network so that you keep only the top-k activations in the hidden layer, and set the remaining activations to 0. Discuss why such an approach will provide improved classification accuracy with limited data.

   Solution. With limited data, we are very likely to encounter overfitting when using the RBF network, as it has a higher variance as compared to linear classifiers like SVM and logistic regression. Hence, the RBF may not be able to learn the correct distribution of the data. However, the approach that was mentioned is like

performing drop-out, which in some sense, is like a regularization on the hidden layer, this gives the model lower variance and will be able to obtain a better testing accuracy.

4. How would your architecture for the previous question change if you were given a training database in which the mutation positions in each sequence were tagged, and the test database was untagged?

   Solution. We can use a RNN/LSTM/transformer network for the memory and learning for the text, and at the output layer, we simply use a cross-entropy loss to predict whether the current input contains mutation or not. Once we trained our RNN/LSTM/transformer network with the training labels, we can use the testing set for performace evaluation.

5. Recommend possible methods for pre-training the input and output layers in the machine translation approach with sequence-to-sequence learning.

   Solution. One can utilize transformer, an attention encoder-decoder architecture desinged by Google. We can first use *Word2vec* to encode a sentence (by using average of the encoded words), or use *Bert* to encode a sentence as our input. We need two of the embedding models so that we are able to understand what is the decoded output. Once we have the embeddings of the two models, we can use supervised model to train the **transformer** to learn in a seq2seq fashion.

6. Consider a setting in which you have a large database of pairs of sentences in different languages. Although you have sufficient representation of each language, some pairs might not be well represented in the database. Show how you can use this training data to (i) create the same universal code for a particular sentence across all languages, and (ii) have the ability to translate even between pairs of languages not well represented in the database.

   Solution. We can use an autoencoder like architecture to accomplish this. Assume that we want to translate English into Chinese. We can construct a seq2seq model, for example, transforers, for the translation part. But furthermore, we try to add another transformer after the the Chinese output, and make it translate back to English. We utilize two losses to train the network, one calculating the accuracy

of Chinese, another one is how similar are the English output and our generated English. This can ensure our translation quality, and make sure that we are actually learning the structure of a language. So for pairs that is not represented. So assume that we have an English word but we don't have the corresponding Chinese in the database, we input the English word into the pretrain model and get the middle output, which is the Chinese word, we then check the output from the transformer after the Chinese word and match it with our input, this way we can check the quality of our model.

7. For a one-dimensional time series of length L and a filter of size F, what is the length of the output? How much padding would you need to keep the output size to a constant value?

Solution. If we mark the first element of the series as one, and the last element of the series as L, than initially, our filter is placed at 1 through F. We continue to slide the filter left by one unit until the filter is placed at $L - F$ through L. So the procedure will happen $L - F + 1$ times from the observation above, so it will result in an output size of $L - F + 1$.
In order to keep a constant output size, that is, of size L, we need to add padding to both size of the output sequence. Assume that the pad length is P, we get:

$$L - F + 1 + 2P = L$$

By solving the equation, we get $P = \frac{(F-1)}{2}$.

8. Compute the convolution of the input volume in the upper-left corner of Figure 8.2 with the horizontal edge detection filter of Figure 8.1(b). Use a stride of 1 without padding.

Solution. The edge detecting filter looks like this:

1	1	1
0	0	0
-1	-1	-1

And the input volume looks like this:

6	3	4	4	5	0	3
4	7	4	0	4	0	4
7	0	2	3	4	5	2
3	7	5	0	3	0	7
5	8	1	2	5	4	2
8	0	1	0	6	0	0
6	4	1	3	0	4	5

So essentially, when we are sliding, we sum the three values from the first row that the kernel is covering and subtract it by the sum of the last row that the kernel is covering. The result is:

4	6	4	-3	-3
0	-1	0	1	-2
-5	-6	1	-1	0
6	11	1	-3	4
3	3	4	4	2

9. Perform a 4×4 pooling at stride 1 of the input volume in the upper-left corner of Figure 8.4.

Solution. The input volume looks like:

6	3	4	4	5	0	3
4	7	4	0	4	0	4
7	0	2	3	4	5	2
3	7	5	0	3	0	7
5	8	1	2	5	4	2
8	0	1	0	6	0	0
6	4	1	3	0	4	5

Now, assuming that we are using maxpooling, that is we choose the max value from every kernel size. For this question, the pooling kernel size is 4 by 4. So, the resulting output is:

7	7	5	7
8	8	5	7
8	8	6	7
8	8	6	7

10. Throughout this chapter, a neural network, referred to as the policy network, has been used in order to implement the policy gradient. Discuss the importance of the choice of network architecture in different settings.

Solution. The choice of the network architecture is going to greatly affect the ability to represent the learned function. The policy gradient-based following the PPO style will generally discard and don't store the expierences as compared to other deep learning method, say DQN, or some policy gradient medthods. A neural network usually can have more complex representation and usually perform better than traditional shallow models.

11. You have two slot machines, each of which has an array of 100 lights. The probability distribution of the reward from playing each machine is an unknown (and possibly machine-specific) function of the pattern of lights that are currently lit up. Playing a slot machine changes its light pattern in some well-defined but unknown way. Discuss why this problem is more difficult than the multi-armed bandit problem. Design a deep learning solution to optimally choose machines in each trial that will maximize the average reward per trial at steady-state.

Solution. This problem is more difficult than the multi-armed bandit problem, as the probability distribution for the multi-armed bandit problem is fixed, however the probability of the problem changes state-by-state.
Fortuanately, the pattern is well-defined, as it is representatable by a complex function, We can deploy a deep learner, which also takes in the state as an consideration, and perform, for example, DQN.

12. Consider the well-known game of rock-paper-scissors. Human players often try to use the history of previous moves to guess the next move. Would you use a Q-learning or a policy-based method to learn to play this game? Why? Now consider a situation in which a human player samples one of the three moves with a probability that is an

unknown function of the history of 10 previous moves of each side. Propose a deep learning method that is designed to play with such an opponent.Would a well-designed deep learning method have an advantage over this human player? What policy should a human player use to ensure probabilistic parity with a deep learning opponent?

Solution. Q-learning is well-suitable for this game, due to its stochastic nature, like in the multi-armed bandit problem. If we want to utilize deep learning, We can use DQN, to learn the distribution from our opponent.
However, as a human, stand a chance is to use a complete random strategy, since any complex function you use, is exploitable by a deep learner, the only way to ensure probabilistic parity is to use a random strategy.