Fundamentals of Machine Learning - 2022

Practice 4 - Aritificial Neural Networks (ANN)
September 20 2022

This practice is intended for your own exercise and **does not** need to be turned in.

1 Questions

1. Name three popular activation functions. Can you draw them? What can you say about their derivatives, and why this is important for training a neural net?
2. Describe the role of the bias parameter and the learning rate in the context of ANNs.
3. Name different aspects you can change in a neural network arquitecture.
4. What is "early stoping"? Whay is "dropout"? How can you implement these in the Keras API?
5. What does exactly the backpropagation algorithm computes? What is the precise relation to gradient descent?
6. List the necessary steps to enable GPU usage in Google Colab. What approximate speedup can you expect over CPU usage?

2 Problems

2.1 Your first ANN

1. Using the Python's CuPy library train an ANN to perform parity checks of 4 bits arrays.
   (a) Build an ANN that has 4 unit imputs, a fully connected layer with 4 nodes as a hidden layer, and a single output layer to return the result. (You can ommit the bias paramaters for this network).
   (b) Select an activation function g, such as tanh, ReLu, sigmoid, etc.
   (c) Build a training set with the set of all 16 combinations of the 4 bits arrays.
   (d) Code the backpropagation algorithm for this network.

(e) Using a MSE cost function, train your network for 5000 epochs and plot the evolution of the training error.
(Note: This is the only time in this course that you just need to evaluate the training error. The reason is that you are training your model with the WHOLE data universe, so there is no generalization further to do and thus no need for a test error.)

(f) Finally, compare your results with a NN built with Keras with the same arquitecture and parameters.

2.2 A real life problem

2. You have a dataset with short newswires (data) and their 46 corresponding topics (labels). You have to classfify the newswires according to their topic using artificial neural networks. This problem aims at using GPUs so we recomend you to use GoogleColab, unless you have a GPU on your computer.

   (a) Open a new GoogleColab .ipynb notebook file. Remember that you can load your files from your google drive to see them in colab, doing:

   ```
   from google.colab import drive
   drive.mount('/content/drive')
   ```

   like this you can upload the pickles files to your drive and load them from there.

   (b) Inside google Colab, load the dataset `newswires_dict.p` provided as a pickle, which contains the train as test datasets stored in the form:
   (traindata, trainlabels), (testdata, testlabels)

   (c) Load a dictionary ("wordindex") with the index corresponding to each word, that is in the file `newswires_wordindex.p` also provided as a pickle.
   Suggestion: to understand how the message encoding/decoding works, try:

   ```
   reversewordindex = dict([(value, key) for (key, value) in wordindex.items()])
   decodednewswire = " ".join([reversewordindex.get(i - 3, "?") for i in traindata[0]])
   ```

   and print the decoded newswire.

   (d) Design a neural network to classify each of the news in only one of the 46 categories.

   (e) How many components will the input tensor of the network have and what will be the meaning of each dimension? Vectorize the input vectors accordingly, using a function built from scratch, and also using the built-in way to do this in Keras with the `to-categorical` function.

   (f) Now, once the input and output tensors are vectorized, you will design the neural network. Which architecture will your network have? How many inner layers? How many dimensions will each layer have?

(g) How many dimensions should the last layer of the network have? Which kind of activation should you use and what does that means in terms of the probability distribution over the 46 different output classes?

(h) What is an appropriate loss function to use in this case, and what does it meassure?

(i) Perform the validation. Set apart 1000 samples in the training data to use as a validation set and try a model fit with 20 epochs and batch-size 512.

(j) Plot the loss function vs the number of epochs, what do you see?

(k) Retrain the model from scratch using the minimum number of epochs that avoids overfitting and evaluate the network over the test set. Which accuracy do you get? How does it compare with a random classification? (you might perform a `np.random.shuffle` over the labels to estimate that).

(l) Try using larger or smaller layers: 32 units, 128 units. What happens with the validation accuracy? Why does that happen?

(m) You used two hidden layers. Now try using a single hidden layer, or three hidden layers. Are those results better than the one with two hidden layers?