DSC 255: Machine Learning - Homework 10

Mathematical and conceptual exercises

1. **Neural net parametrization.** A feedforward neural net has an input layer with 10 nodes, followed by four hidden layers with 1000 nodes each, and an output layer with one node. Each layer is fully connected to the previous layer. What is the total number of parameters, approximately? You may disregard the offset term at each node (the "b" in the linear function) if you like.

2. **Softmax probabilities.** The output layer of a particular neural net has four nodes y_1, y_2, y_3, y_4, representing four labels in a classification problem. For some input x, these nodes end up with the values

$$y_1 = 1.0, \quad y_2 = 0.0, \quad y_3 = -1.0, \quad y_4 = 0.0$$

and these are converted into probabilities using a softmax. What is the most likely label, and what probability is assigned to it?

3. **Exclusive-OR.** The XOR Boolean function, $\{0,1\}^2 \to \{0,1\}$, has the following behavior:

x_1	x_2	XOR(x_1, x_2)
0	0	0
0	1	1
1	0	1
1	1	0

Show how to implement this function using a neural net with two input units (x_1, x_2), an output unit, and two hidden units with ReLU activation.

4. **Decipher the net.**

   (a) Which function $\{0,1\}^2 \to \{0,1\}$ is realized by the following neural net with two input units x_1, x_2, two hidden units h_1, h_2, and one output unit y?

   - $h_1 = \text{ReLU}(x_1 + x_2)$
   - $h_2 = \text{ReLU}(x_1 + x_2 - 1)$
   - $y = h_1 - h_2$

(b) Which function $\mathbb{R} \to \mathbb{R}$ is realized by the following neural net with one input unit x, two hidden units h_1, h_2, and one output unit y?

- $h_1 = \text{ReLU}(x)$
- $h_2 = \text{ReLU}(-x)$
- $y = h_1 + h_2$

Programming exercises

Before undertaking these problems, install PyTorch on your computer and download the archive files `simple-nn.zip` and `hymenoptera.zip` from the course website.

1. **Neural net experiments.**

   When you unzip `simple-nn.zip`, you will see a Jupyter notebook called `simple-nn.ipynb`. Look through it and run the cells in Sections 1 and 2. These load in a data set called `data1.txt`, train a neural net on it, and print the decision boundary.

   (a) Look at the training code. You will see that the neural net has one hidden layer with H nodes. Train the net on the five data sets `data1.txt` through `data5.txt`. For each, try two values of H, and train multiple times (at least five), picking the lowest-error solution. For each of the data sets, plot the best model you found, state what value of H produced it, and state the number of iterations till convergence.

   (b) Now move to Section 3 of the notebook. This generates a new data set that is highly noisy. To deal with it, copy over the neural net code from above and modify it to train a net with two hidden layers, each with H nodes. Train the net a bunch of times and print the best boundary you find. You should aim for < 100 errors on the training set (this should be possible even with $H = 4$).

2. **A computer vision classification task.** In this problem, we'll train a classifier that takes as input an image of an ant or a bee, and determines which it is.

   When you unzip `hymenoptera.zip`, you will see a Jupyter notebook, called `ants-bees.ipynb` and a directory called `hymenoptera_data`.

   (a) The `hymenoptera_data` directory has a subdirectory with the training set (`train`) and another with the test set (`val`). Take a look at some of the images. They are of varying sizes, with diverse backgrounds. And there are only 244 training images: not an easy learning problem!

Run the first few cells of the notebook to see how the images are normalized in size. Pick an image from the directory and show both the original version and the normalized version.

(b) Run the remaining cells of the notebook to see how the ResNet50 network is used to produce a 2048-dimensional representation for each image, and how a logistic regression classifier is constructed on top of this. Report the test accuracy of the logistic regression classifier.

(c) Now, use this same 2048-d representation to construct a k-nearest neighbor classifier. Give the test accuracies obtained for $k = 1, 3, 5$. Note: If you use `sklearn.neighbors.KNeighborsClassifier`, you might want to set `algorithm='kd_tree'`.