Exercise 4

Deep Learning Lab

October 17, 2022

1 MNIST Digit Classification Using Feed-Forward Neural Networks

Using the code presented in the lecture (at the end of Sec. 2.2, "Put them all together") as a starting point, in this task, you will build a digit classification system using feed-forward neural networks.

1. The validation set was missing in the code presented in the lecture. Create a validation set by using the first 55000 samples of the original training data for training and the rest for validation. Hint: Use `torch.utils.data.SubsetRandomSampler` and `sampler` option of `torch.utils.data.DataLoader`.

2. Add the necessary lines of code to monitor the training progress (i.e. training and validation losses). Remember, you should only look at training and validation losses during training!

3. Make sure that your code works with the points above. Now try to find a better set of hyper-parameters. You may try different learning rates, batch sizes, number of epochs, sizes of hidden layers, number of hidden layers, activation functions, and optimization algorithm.

4. Once you settle on a final model, compute the test set accuracy.

2 Binary Classification

1. Consider the following function that creates a classification dataset where observations are drawn from multivariate Gaussian distributions:

   Listing 1: Multivariate Gaussians dataset.

```python
def create_dataset(means, std, sample_size, seed=None):
    random_state = np.random.RandomState(seed)

    X = np.zeros((sample_size, len(means[0])), dtype=np.float32)
    Y = np.zeros((sample_size, len(means)), dtype=np.float32)

    cov = np.eye(len(means[0]))*(std**2)

    for i in range(sample_size):
        c = random_state.randint(len(means))
        X[i] = random_state.multivariate_normal(means[c], cov)
        Y[i, c] = 1.

    return X, Y
```

   (a) Using this function, create a dataset with two classes. Place one mean at $(-1, 1)$ and another at $(1, -1)$. Let the standard variation be 0.5, the sample size 500, and the seed 0.

(b) Use plt.scatter to plot the observations in your dataset, coloring the points according to their classes. Tip: use the colors 0 and 1 to represent the classes, and set the colormap (cmap) to *plt.cm.RdBu*.

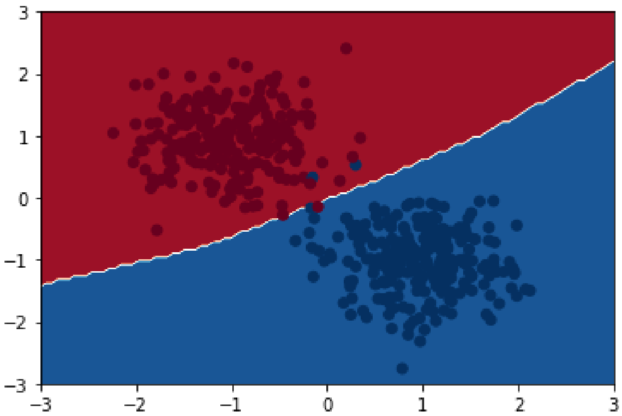
(c) Train a multilayer perceptron using your dataset.

(d) Use your multilayer perceptron to predict the class of observations in a grid on the set $[-3, 3] \times [-3, 3]$. Visualize the decision boundary by creating a contour plot based on these predictions (see Fig. 2). Tip: use np.meshgrid to create a grid of observations and plt.contourf to create the contour plot. Here is an example.

(e) Observe the consequences of changing the parameters that you used to create your dataset (means, standard deviation, sample size).

(f) Observe the consequences of changing model hyperparameters.

Figure 1: Example decision boundary.



(g) Do the same on the following dataset:

Listing 2: Generate "moon"-form datapoints.

```
from sklearn.datasets import make_moons

X, y = make_moons(n_samples=500, noise=0.2)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.RdBu)
```

Figure 2: Dataset with two moons.