Machine Learning Lab 8

Multilayer Perceptron on MNIST

1. In this lab, we will use `MLPClassifier` from Scikit-learn for classifying MNIST dataset with multilayer perceptron. To use `MLPClassifier`, you can do the following, as an example:

```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(50,), max_iter=10,
alpha=1e-4, solver='sgd', verbose=10, tol=1e-4, random_state=1,
learning_rate_init=.1)
mlp.fit(X_train, y_train)
mlp.predict(X_test)
```

   Note the `MLPClassifier` takes various parameters. For example `hidder_layer_sizes` indicates number of hidden nodes in each hidden layer (in this case, we put one hidden layer with 50 nodes). `max_iter` indicates maximum number of iterations are allowed for the training (10 in this case). `solver` indicates the algorithm for solving weight updates (`sgd` means stochastic gradient descent, the one we learned in the lecture). For more explanation of these parameters, see the documentation at https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier . After creating the class, you use `fit` and `predict` for training and testing of the classifier. Read the documentation and try to understand how to use `MLPClassifier`.
2. Try to use multilayer perception to classify digit '2' and digit '5'. Calculate the accuracy of your classifier. You should play around with different parameters, e.g. number of hidden nodes and hidden layers, number of iterations, learning rate, etc, to see how good your classifier can be. You can use first 1000 images only if the training time for all the images are too slow.
3. `MLPClassifier` can work with multi-class classification directly. Try to use it over all the digits and try to tune the parameters for the best performance.
4. (Optional) You can also visualize all the weights for all the perceptrons in the network. The weights can be obtained through the attribute of `coef_` of `MLPClassifier. coef_` is a list whose ith element represents the weight

matrix corresponding to layer i. For more help of how to visualize those weights, see online tutorial https://scikit-learn.org/stable/auto_examples/neural_networks/plot_mnist_filters.html#sphx-glr-auto-examples-neural-networks-plot-mnist-filters-py .