Machine Learning Group
Fakulty IV, Technische Universität Berlin
Prof. Dr. Klaus-Robert Müller
Email: klaus-robert.mueller@tu-berlin.de

Problem Set 4: Support Vector Machines, Neural Networks

Part 1: Implementation

Assignment 1 (20 points)

Implement a feedforward neural network with ReLU activations,

```
class neural_network(layers, lr=.1, p=.1, lam=.1)
```

where `layers` is a list of positive integers that represent the dimensionality of the respective layer. At initialization, all weights and biases are sampled independently from $\mathcal{N}(\mu = 0, \sigma^2 = 10^{-1})$. The parameters are stored in two parameter lists `weights` $= [\mathbf{W}_1, \ldots, \mathbf{W}_L]$, `biases` $= [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_L]$ for each layer $1, \ldots, L$. `lr` is the learning rate, `p` is the dropout rate, `lam` is the weight decay coefficient. The class has the following functions:

- `relu(X, W, b)` with `X` $\in \mathbb{R}^{n \times \text{in}}$, `W` $\in \mathbb{R}^{\text{in} \times \text{out}}$ and `b` $\in \mathbb{R}^{\text{out}}$ with in, out $\in \mathbb{N}$ being the dimensions of ReLU inputs and outputs. The function computes the ReLU activations in the next layer during the forward pass. Dropout is also implemented in this function (see guide).
- `softmax(X, W, b)` computes the softmax activation in the last layer,
- `forward(X)` computes the forward pass in a for-loop over ReLU layers and a final softmax layer
- `loss(output, y)` computes and returns the cross-entropy loss where `output = forward(X)` is the model output and `y` are the true labels,
- `fit(X, y, nsteps=1000, bs=100, plot=False)` trains the objective for `nsteps` steps and batch size `bs`. When `plot` is set to `True`, plot the training loss, validation loss and validation accuracy as a function of the number of iterations after training. Use `torch.optim.SGD` as an optimizer.
- `predict(X)` returns `forward(X)` for compatibility with your cross-validation class from Problem Set 3.

You have to install the `torch` package (`pip install torch`) for this assignment. You may only use `numpy` capabilities (no pre-implemented layers, activation functions or losses) as well as optimizers and autograd.

Assignment 2 (5 points)

Implement a plotting method

```
plot_boundary_2d(X, y, model)
```

that takes as input a $(n \times 2)$ array `X`, ± 1 labels `y` and a fitted classifier. It should

- plot the points in `X` as circles in different colors for both classes,
- mark support vectors with a cross (for SVMs only) and
- plot the separating hyperplane.

Hint: To plot the separating hyperplane, evaluate your classifier on each point of a grid that stretches all data points and then plot the level zero contour line.

Assignment 3 (10 points)

Write the SVM dual optimization problem as a quadratic programming (QP) problem in the form of

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \frac{1}{2}\boldsymbol{x}^\top \mathbf{P}\boldsymbol{x} + \boldsymbol{q}^\top \boldsymbol{x} \\ \text{s.t.} \quad & \mathbf{G}\boldsymbol{x} \preceq \boldsymbol{h} \\ & \mathbf{A}\boldsymbol{x} = \boldsymbol{b} \end{aligned}$$

Describe the variables in your report. Implement this in the class `svm_qp` with the help of `cvxopt.solvers.qp` (see stubs). The class should have the functions `fit(X, y)` and `predict(X)`. The labels in `y` will be ± 1. Your `fit` method should only save the support vectors, not all data points.. You might have to install the `cvxopt` package via `pip install cvxopt`.

Part 2: Application

Assignment 4 (25 points)

Use your SVM QP implementation to train classifiers on the `easy_2d` dataset from the ISIS site.

1. Find the optimal parameters C and σ for a Gaussian kernel and plot the results. Use your cross validation method. If you do not have a running cross validation method, contact us.
2. Train one model for a σ and C that obviously overfit and for a σ and C that obviously underfit the data. Plot the results.
3. For optimal C and σ, plot a receiver operator characteristics (ROC) curve by varying the bias parameter b of your SVM model.

Assignment 5 (15 points)

In this assignment you will work on the UCI Iris dataset[1]. This is a 4-dimensional dataset with 150 instances in 3 classes. Use the `iris.npz` file from the ISIS course page. Which classes are linearly separable from the two other classes and which classes are not? Are they separable with a non-linear classifier? Describe what you tested. Provide the found hyperparameters and classification accuracies.

Assignment 6 (25 points)

We would now like to compare SVM and neural networks on a 'real' dataset, the USPS dataset, which we saw on earlier sheets. For SVM, one trains to classify one digit against the rest of the digits (one-vs-rest classification). The neural network has one output for each class.

1. Use 5-fold cross-validation to find the best kernel and kernel parameters for each digit. Report the best kernel and kernel parameter as well as the estimated test error for each digit.
2. Use 5-fold cross-validation to find neural network parameters (`layers`, `p`, `lam` and `lr`). You may fix some reasonable values for the `layers` if you encounter long runtimes. Report the best hyperparameters and describe how you found them. Also report test errors for each class.
3. Plot, for each digit, 5 randomly chosen Support Vectors from the two classes.
4. Plot 100 weight vectors from the first layer of your neural network as grayscale images. Do these plots say anything about the test error of the model? Does the weight decay parameter have an effect on these plots?
5. What problem do we have when using one-vs-rest classification with the SVM implemented via the guide book pseudo-code (that would not exist with one-vs-one classification or in neural networks)? How could we change the SVM method to fix this? (You only have to describe it, you do not have to implement it.)

[1] `http://archive.ics.uci.edu/ml/datasets/Iris`