Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 1

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data classification:

1. Add new neuron activation function: leaky Relu.
2. Add new functions for data augmentation with the use of the random noise signal. The function should change inputs by adding or subtracting small random value of the input vector. The random value should be different for each sample and each element of the input vector. The random value should be generated with use standard normal distribution and scaled with additional parameter "rand_scale" (rand_scale*numpy.random.randn()). Add new argument "rand_scale" to training function, which specify a scaling factor for data augmentation.
3. Change training function to mini-batch training function. Add new argument "batch_size" to training function, which specify the batch size. If the batch size is equal to 1 than training will be using the stochastic gradient descent. If the batch size is equal to size of training dataset than training will be using the full batch training.

Task 2

Evaluate classification accuracy of neural network: 1 input layer, 2 hidden layers with 12 units and leaky Relu activation, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 1000,
- learning rate = 0.001,
- batch size = 8,
- rand_scale = 0.1.

Training dataset:

```
import numpy as np
n = 40

X1_1 = 1 + 4 * np.random.rand(n, 1)
X1_2 = 1 + 4 * np.random.rand(n, 1)
class1 = np.concatenate((X1_1, X1_2), axis=1)
Y1 = np.ones(n)

X0_1 = 3 + 4 * np.random.rand(n, 1)
X0_2 = 3 + 4 * np.random.rand(n, 1)
class0 = np.concatenate((X0_1, X0_2), axis=1)
Y0 = np.zeros(n)

train_x = np.concatenate((class1, class0))
train_y = np.concatenate((Y1, Y0))
```

Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 2

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data regression:

1. Create a new function for saving and loading trained model to file with 3 arguments. The first argument will be "file" with file name/path. The second argument will be "operation" with option 'save' for saving a network model to file and 'load' to load network model from file. The third optional argument of new function will be "model", where we send neural network model dictionary. The function will return model of neural network.
2. Update network training function to return loss function history for each iteration and each epochs. You can use any structure of loss history. Create a new function "history_plot", which plot changes of loss function based on loss history in two subplots: first plot of loss function in each epoch (total loss) and second plot of loss function in each iteration in each epoch. Add necessary description of the charts (labels, axis description, titles).
3. Add and integrate early stopping function in the training process. The early stop function allows to stop training process and return best network model, if the difference in loss function is small enough in few epochs or loss function is increasing value in few epochs. The function parameter will be "accuracy" and "patience". The accuracy control the difference between two successive epochs to stop training process. The patience parameter control number of epochs with no improvements to stop training process. The function should return the best neural network model, which means it returns the model with the lowest loss function value in the patient range. Add new arguments "accuracy" and "patience" to the training function to control early stop criterion.

Task 2

Evaluate loss function changes of neural network: 1 input layer, 2 hidden layers with 8 units and hyperbolic tangent, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 10000,
- learning rate = 0.001,
- early stopping: accuracy = 0.01, patience = 5 epochs

Training dataset:

```
import numpy as np
n = 20

X = np.linspace(-5, 5, n).reshape(-1, 1)
data_noise = np.random.normal(0, 0.2, n).reshape(-1, 1)
xaxis = np.sin(2 * X)
yaxis = np.cos(X)
x1, x2 = np.meshgrid(xaxis, yaxis)
y = x1 * x2 + data_noise

train_x = [np.reshape(x1, (x1.size)), np.reshape(x2, (x2.size))]
train_y = np.reshape(y, (y.size))
```

Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 3

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data regression:

1. Add new neuron activation function: ELU.
2. Add new function for changing the learning rate with use Darken and Moody method. Integrate new function with training function of multi-layer neural network. Add new argument "lrate_method = 'darken_moody'" to to select this method of learning rate changing during training. The argument "l_rate" will be describing the initial learning rate value.
3. Add new function for changing the learning rate with the use Adagrad method. Integrate new function with training function of multi-layer neural network. Add new argument "lrate_method = 'adagrad'" to select this method of learning rate changing during training. The argument "l_rate" will be describing initial learning rate value of all weights.

Evaluate loss function changes of neural network: input layer, 1 hidden layers with 32 units and Elu activation, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 1000,
- initial learning rate = 0.01,
- test: fixed learning rate, Darken and Moody method, and Adagrad method.

Training dataset:

```
import numpy as np

X = np.linspace(-5, 5, n).reshape(-1, 1)
y = np.sin(2 * X) + np.cos(X) + 5
data_noise = np.random.normal(0, 0.2, n).reshape(-1, 1)
Y = y + data_noise

train_x = x
train_y = Y
```

Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 4

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data classification:

1. Create a new function to calculate classification quality measures: F1, confusion matrix for binary classification problems (see sklearn library). The new function should take two arguments "y_predict" with predict values by neural network and "y_true" with true/desired values. The function should print values of f1, precision, recall, and display confusion matrix.
2. Integrate K-fold cross-validation functionality in neural network training function. Add new argument "k_fold" to the training function. If the argument "k_fold" is 0 than training will not use cross validation methods and for positive values will performer K-fold cross-validation with "k_fold" divisions. The training function at the end of training should display minimum, average, and maxim values of quality measures f1.
3. Integrate momentum method functionality in neural network training function. The momentum method use in weight updating process a history of previous updates. Add new argument "alpha" to training function, which is a positive number called the momentum constant.

Task 2

Evaluate classification accuracy of neural network: 1 inpu t layer, 1 hidden layers with 32 units and ReLu activation, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 1000,
- learning rate = 0.001,
- K-fold cross-validation
- alpha = 0.1

Training dataset:

```
import numpy as np
n = 40

X1_1 = 1 + 2 * np.random.rand(n, 1)
X1_2 = 1 + 4 * np.random.rand(n, 1)
class1 = np.concatenate((X1_1, X1_2), axis=1)
Y1 = np.ones(n)

X0_1 = 3 + 2 * np.random.rand(n, 1)
X0_2 = 3 + 4 * np.random.rand(n, 1)
class0 = np.concatenate((X0_1, X0_2), axis=1)
Y0 = np.zeros(n)

train_x = np.concatenate((class1, class0))
train_y = np.concatenate((Y1, Y0))
```

Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 5

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data classification:

1. Add new neuron activation function: leaky Relu.
2. Add new functions for data augmentation with the use of the random noise signal. The function should change inputs by adding or subtracting small random value of the input vector. The random value should be different for each sample and each element of the input vector. The random value should be generated with use standard normal distribution and scaled with additional parameter "rand_scale" (rand_scale*numpy.random.randn()). Add new argument "rand_scale" to training function, which specify a scaling factor for data augmentation.
3. Change training function to mini-batch training function. Add new argument "batch_size" to training function, which specify the batch size. If the batch size is equal to 1 than training will be using the stochastic gradient descent. If the batch size is equal to size of training dataset than training will be using the full batch training.

Task 2

Evaluate classification accuracy of neural network: 1 input layer, 2 hidden layers with 12 units and leaky Relu activation, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 1000,
- learning rate = 0.001,
- batch size = 8,
- rand_scale = 0.1.

Training dataset:

```
import numpy as np
n = 40

X1_1 = 0 + 4 * np.random.rand(n, 1)
X1_2 = 0 + 4 * np.random.rand(n, 1)
class1 = np.concatenate((X1_1, X1_2), axis=1)
Y1 = np.ones(n)

X0_1 = 3 + 3 * np.random.rand(n, 1)
X0_2 = 3 + 3 * np.random.rand(n, 1)
class0 = np.concatenate((X0_1, X0_2), axis=1)
Y0 = np.zeros(n)

train_x = np.concatenate((class1, class0))
train_y = np.concatenate((Y1, Y0))
```

Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 6

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data regression:

1. Create a new function for saving and loading trained model to file with 3 arguments. The first argument will be "file" with file name/path. The second argument will be "operation" with option 'save' for saving a network model to file and 'load' to load network model from file. The third optional argument of new function will be "model", where we send neural network model dictionary. The function will return model of neural network.
2. Update network training function to return loss function history for each iteration and each epochs. You can use any structure of loss history. Create a new function "history_plot", which plot changes of loss function based on loss history in two subplots: first plot of loss function in each epoch (total loss) and second plot of loss function in each iteration in each epoch. Add necessary description of the charts (labels, axis description, titles).
3. Add and integrate early stopping function in the training process. The early stop function allows to stop training process and return best network model, if the difference in loss function is small enough in few epochs or loss function is increasing value in few epochs. The function parameter will be "accuracy" and "patience". The accuracy control the difference between two successive epochs to stop training process. The patience parameter control number of epochs with no improvements to stop training process. The function should return the best neural network model, which means it returns the model with the lowest loss function value in the patient range. Add new arguments "accuracy" and "patience" to the training function to control early stop criterion.

Task 2

Evaluate loss function changes of neural network: 1 input layer, 2 hidden layers with 8 units and hyperbolic tangent, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 10000,
- learning rate = 0.001,
- early stopping: accuracy = 0.01, patience = 5 epochs

Training dataset:

```
import numpy as np
n = 20

X = np.linspace(-5, 5, n).reshape(-1, 1)
data_noise = np.random.normal(0, 0.2, n).reshape(-1, 1)
xaxis = np.sin(2 * X)
yaxis = np.cos(X)
x1, x2 = np.meshgrid(xaxis, yaxis)
y = x1 * x2 + data_noise + 7

train_x = [np.reshape(x1, (x1.size)), np.reshape(x2, (x2.size))]
train_y = np.reshape(y, (y.size))
```

Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 7

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data regression:

1. Add new neuron activation function: ELU.
2. Add new function for changing the learning rate with use Darken and Moody method. Integrate new function with training function of multi-layer neural network. Add new argument "lrate_method = 'darken_moody'" to to select this method of learning rate changing during training. The argument "l_rate" will be describing the initial learning rate value.
3. Add new function for changing the learning rate with the use Adagrad method. Integrate new function with training function of multi-layer neural network. Add new argument "lrate_method = 'adagrad'" to select this method of learning rate changing during training. The argument "l_rate" will be describing initial learning rate value of all weights.

Evaluate loss function changes of neural network: input layer, 1 hidden layers with 32 units and Elu activation, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 1000,
- initial learning rate = 0.01,
- test: fixed learning rate, Darken and Moody method, and Adagrad method.

Training dataset:

```
import numpy as np

X = np.linspace(-5, 5, n).reshape(-1, 1)
y = np.sin(2 * X) + np.cos(X) + 3
data_noise = np.random.normal(0, 0.2, n).reshape(-1, 1)
Y = y + data_noise

train_x = x
train_y = Y
```

Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 8

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data classification:

1. Create a new function to calculate classification quality measures: F1, confusion matrix for binary classification problems (see sklearn library). The new function should take two arguments "y_predict" with predict values by neural network and "y_true" with true/desired values. The function should print values of f1, precision, recall, and display confusion matrix.
2. Integrate K-fold cross-validation functionality in neural network training function. Add new argument "k_fold" to the training function. If the argument "k_fold" is 0 than training will not use cross validation methods and for positive values will performer K-fold cross-validation with "k_fold" divisions. The training function at the end of training should display minimum, average, and maxim values of quality measures f1.
3. Integrate momentum method functionality in neural network training function. The momentum method use in weight updating process a history of previous updates. Add new argument "alpha" to training function, which is a positive number called the momentum constant.

Task 2

Evaluate classification accuracy of neural network: 1 inpu t layer, 1 hidden layers with 32 units and ReLu activation, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 1000,
- learning rate = 0.001,
- K-fold cross-validation
- alpha = 0.1

Training dataset:

```
import numpy as np
n = 40

X1_1 = 1 + 4 * np.random.rand(n, 1)
X1_2 = 0 + 4 * np.random.rand(n, 1)
class1 = np.concatenate((X1_1, X1_2), axis=1)
Y1 = np.ones(n)

X0_1 = 0 + 4 * np.random.rand(n, 1)
X0_2 = 3 + 4 * np.random.rand(n, 1)
class0 = np.concatenate((X0_1, X0_2), axis=1)
Y0 = np.zeros(n)

train_x = np.concatenate((class1, class0))
train_y = np.concatenate((Y1, Y0))
```

Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 9

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data classification:

1. Add new neuron activation function: leaky Relu.
2. Add new functions for data augmentation with the use of the random noise signal. The function should change inputs by adding or subtracting small random value of the input vector. The random value should be different for each sample and each element of the input vector. The random value should be generated with use standard normal distribution and scaled with additional parameter "rand_scale" (rand_scale*numpy.random.randn()). Add new argument "rand_scale" to training function, which specify a scaling factor for data augmentation.
3. Change training function to mini-batch training function. Add new argument "batch_size" to training function, which specify the batch size. If the batch size is equal to 1 than training will be using the stochastic gradient descent. If the batch size is equal to size of training dataset than training will be using the full batch training.

Task 2

Evaluate classification accuracy of neural network: 1 input layer, 2 hidden layers with 12 units and leaky Relu activation, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 1000,
- learning rate = 0.001,
- batch size = 8,
- rand_scale = 0.1.

Training dataset:

```
import numpy as np
n = 40

X1_1 = 0 + 5 * np.random.rand(n, 1)
X1_2 = 0 + 5 * np.random.rand(n, 1)
class1 = np.concatenate((X1_1, X1_2), axis=1)
Y1 = np.ones(n)

X0_1 = 4 + 6 * np.random.rand(n, 1)
X0_2 = 4 + 6 * np.random.rand(n, 1)
class0 = np.concatenate((X0_1, X0_2), axis=1)
Y0 = np.zeros(n)

train_x = np.concatenate((class1, class0))
train_y = np.concatenate((Y1, Y0))
```

Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 10

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data regression:

1. Create a new function for saving and loading trained model to file with 3 arguments. The first argument will be "file" with file name/path. The second argument will be "operation" with option 'save' for saving a network model to file and 'load' to load network model from file. The third optional argument of new function will be "model", where we send neural network model dictionary. The function will return model of neural network.
2. Update network training function to return loss function history for each iteration and each epochs. You can use any structure of loss history. Create a new function "history_plot", which plot changes of loss function based on loss history in two subplots: first plot of loss function in each epoch (total loss) and second plot of loss function in each iteration in each epoch. Add necessary description of the charts (labels, axis description, titles).
3. Add and integrate early stopping function in the training process. The early stop function allows to stop training process and return best network model, if the difference in loss function is small enough in few epochs or loss function is increasing value in few epochs. The function parameter will be "accuracy" and "patience". The accuracy control the difference between two successive epochs to stop training process. The patience parameter control number of epochs with no improvements to stop training process. The function should return the best neural network model, which means it returns the model with the lowest loss function value in the patient range. Add new arguments "accuracy" and "patience" to the training function to control early stop criterion.

Task 2

Evaluate loss function changes of neural network: 1 input layer, 2 hidden layers with 8 units and hyperbolic tangent, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 10000,
- learning rate = 0.001,
- early stopping: accuracy = 0.01, patience = 5 epochs

Training dataset:

```
import numpy as np
n = 20

X = np.linspace(-5, 5, n).reshape(-1, 1)
data_noise = np.random.normal(0, 0.2, n).reshape(-1, 1)
xaxis = np.sin(2 * X)
yaxis = np.cos(X)
x1, x2 = np.meshgrid(xaxis, yaxis)
y = x1 * x2 + data_noise + 10

train_x = [np.reshape(x1, (x1.size)), np.reshape(x2, (x2.size))]
train_y = np.reshape(y, (y.size))
```

Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 11

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data regression:

1. Add new neuron activation function: ELU.
2. Add new function for changing the learning rate with use Darken and Moody method. Integrate new function with training function of multi-layer neural network. Add new argument "lrate_method = 'darken_moody'" to to select this method of learning rate changing during training. The argument "l_rate" will be describing the initial learning rate value.
3. Add new function for changing the learning rate with the use Adagrad method. Integrate new function with training function of multi-layer neural network. Add new argument "lrate_method = 'adagrad'" to select this method of learning rate changing during training. The argument "l_rate" will be describing initial learning rate value of all weights.

Evaluate loss function changes of neural network: input layer, 1 hidden layers with 32 units and Elu activation, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 1000,
- initial learning rate = 0.01,
- test: fixed learning rate, Darken and Moody method, and Adagrad method.

Training dataset:

```
import numpy as np

X = np.linspace(-5, 5, n).reshape(-1, 1)
y = np.sin(2 * X) + np.cos(X) + 6
data_noise = np.random.normal(0, 0.2, n).reshape(-1, 1)
Y = y + data_noise

train_x = x
train_y = Y
```

Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 12

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data classification:

1. Create a new function to calculate classification quality measures: F1, confusion matrix for binary classification problems (see sklearn library). The new function should take two arguments "y_predict" with predict values by neural network and "y_true" with true/desired values. The function should print values of f1, precision, recall, and display confusion matrix.
2. Integrate K-fold cross-validation functionality in neural network training function. Add new argument "k_fold" to the training function. If the argument "k_fold" is 0 than training will not use cross validation methods and for positive values will performer K-fold cross-validation with "k_fold" divisions. The training function at the end of training should display minimum, average, and maxim values of quality measures f1.
3. Integrate momentum method functionality in neural network training function. The momentum method use in weight updating process a history of previous updates. Add new argument "alpha" to training function, which is a positive number called the momentum constant.

Task 2

Evaluate classification accuracy of neural network: 1 inpu t layer, 1 hidden layers with 32 units and ReLu activation, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 1000,
- learning rate = 0.001,
- K-fold cross-validation
- alpha = 0.1

Training dataset:

```
import numpy as np
n = 40


X1_1 = 1 + 5 * np.random.rand(n, 1)
X1_2 = 1 + 5 * np.random.rand(n, 1)
class1 = np.concatenate((X1_1, X1_2), axis=1)
Y1 = np.ones(n)

X0_1 = 3 + 5 * np.random.rand(n, 1)
X0_2 = 3 + 5 * np.random.rand(n, 1)
class0 = np.concatenate((X0_1, X0_2), axis=1)
Y0 = np.zeros(n)

train_x = np.concatenate((class1, class0))
train_y = np.concatenate((Y1, Y0))
```

Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 13

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data classification:

1. Add new neuron activation function: leaky Relu.
2. Add new functions for data augmentation with the use of the random noise signal. The function should change inputs by adding or subtracting small random value of the input vector. The random value should be different for each sample and each element of the input vector. The random value should be generated with use standard normal distribution and scaled with additional parameter "rand_scale" (rand_scale*numpy.random.randn()). Add new argument "rand_scale" to training function, which specify a scaling factor for data augmentation.
3. Change training function to mini-batch training function. Add new argument "batch_size" to training function, which specify the batch size. If the batch size is equal to 1 than training will be using the stochastic gradient descent. If the batch size is equal to size of training dataset than training will be using the full batch training.

Task 2

Evaluate classification accuracy of neural network: 1 input layer, 2 hidden layers with 12 units and leaky Relu activation, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 1000,
- learning rate = 0.001,
- batch size = 8,
- rand_scale = 0.1.

Training dataset:

```
import numpy as np
n = 40

X1_1 = 1 + 4 * np.random.rand(n, 1)
X1_2 = 2 + 4 * np.random.rand(n, 1)
class1 = np.concatenate((X1_1, X1_2), axis=1)
Y1 = np.ones(n)

X0_1 = 4 + 4 * np.random.rand(n, 1)
X0_2 = 5 + 4 * np.random.rand(n, 1)
class0 = np.concatenate((X0_1, X0_2), axis=1)
Y0 = np.zeros(n)

train_x = np.concatenate((class1, class0))
train_y = np.concatenate((Y1, Y0))
```

Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 14

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data regression:

1. Create a new function for saving and loading trained model to file with 3 arguments. The first argument will be "file" with file name/path. The second argument will be "operation" with option 'save' for saving a network model to file and 'load' to load network model from file. The third optional argument of new function will be "model", where we send neural network model dictionary. The function will return model of neural network.
2. Update network training function to return loss function history for each iteration and each epochs. You can use any structure of loss history. Create a new function "history_plot", which plot changes of loss function based on loss history in two subplots: first plot of loss function in each epoch (total loss) and second plot of loss function in each iteration in each epoch. Add necessary description of the charts (labels, axis description, titles).
3. Add and integrate early stopping function in the training process. The early stop function allows to stop training process and return best network model, if the difference in loss function is small enough in few epochs or loss function is increasing value in few epochs. The function parameter will be "accuracy" and "patience". The accuracy control the difference between two successive epochs to stop training process. The patience parameter control number of epochs with no improvements to stop training process. The function should return the best neural network model, which means it returns the model with the lowest loss function value in the patient range. Add new arguments "accuracy" and "patience" to the training function to control early stop criterion.

Task 2

Evaluate loss function changes of neural network: 1 input layer, 2 hidden layers with 8 units and hyperbolic tangent, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 10000,
- learning rate = 0.001,
- early stopping: accuracy = 0.01, patience = 5 epochs

Training dataset:

```
import numpy as np
n = 20

X = np.linspace(-5, 5, n).reshape(-1, 1)
data_noise = np.random.normal(0, 0.2, n).reshape(-1, 1)
xaxis = np.sin(2 * X)
yaxis = np.cos(X)
x1, x2 = np.meshgrid(xaxis, yaxis)
y = x1 * x2 + data_noise + 1

train_x = [np.reshape(x1, (x1.size)), np.reshape(x2, (x2.size))]
train_y = np.reshape(y, (y.size))
```

Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 15

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data regression:

1. Add new neuron activation function: ELU.
2. Add new function for changing the learning rate with use Darken and Moody method. Integrate new function with training function of multi-layer neural network. Add new argument "lrate_method = 'darken_moody'" to to select this method of learning rate changing during training. The argument "l_rate" will be describing the initial learning rate value.
3. Add new function for changing the learning rate with the use Adagrad method. Integrate new function with training function of multi-layer neural network. Add new argument "lrate_method = 'adagrad'" to select this method of learning rate changing during training. The argument "l_rate" will be describing initial learning rate value of all weights.

Evaluate loss function changes of neural network: input layer, 1 hidden layers with 32 units and Elu activation, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 1000,
- initial learning rate = 0.01,
- test: fixed learning rate, Darken and Moody method, and Adagrad method.

Training dataset:

```
import numpy as np

X = np.linspace(-5, 5, n).reshape(-1, 1)
y = np.sin(2 * X) + np.cos(X) + 2
data_noise = np.random.normal(0, 0.2, n).reshape(-1, 1)
Y = y + data_noise

train_x = x
train_y = Y
```

Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 16

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data classification:

1. Create a new function to calculate classification quality measures: F1, confusion matrix for binary classification problems (see sklearn library). The new function should take two arguments "y_predict" with predict values by neural network and "y_true" with true/desired values. The function should print values of f1, precision, recall, and display confusion matrix.
2. Integrate K-fold cross-validation functionality in neural network training function. Add new argument "k_fold" to the training function. If the argument "k_fold" is 0 than training will not use cross validation methods and for positive values will performer K-fold cross-validation with "k_fold" divisions. The training function at the end of training should display minimum, average, and maxim values of quality measures f1.
3. Integrate momentum method functionality in neural network training function. The momentum method use in weight updating process a history of previous updates. Add new argument "alpha" to training function, which is a positive number called the momentum constant.

Task 2

Evaluate classification accuracy of neural network: 1 inpu t layer, 1 hidden layers with 32 units and ReLu activation, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 1000,
- learning rate = 0.001,
- K-fold cross-validation
- alpha = 0.1

Training dataset:

```
import numpy as np
n = 40

X1_1 = 1 + 3 * np.random.rand(n, 1)
X1_2 = 1 + 3 * np.random.rand(n, 1)
class1 = np.concatenate((X1_1, X1_2), axis=1)
Y1 = np.ones(n)

X0_1 = 3 + 3 * np.random.rand(n, 1)
X0_2 = 3 + 3 * np.random.rand(n, 1)
class0 = np.concatenate((X0_1, X0_2), axis=1)
Y0 = np.zeros(n)

train_x = np.concatenate((class1, class0))
train_y = np.concatenate((Y1, Y0))
```

Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 17

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data classification:

1. Add new neuron activation function: leaky Relu.
2. Add new functions for data augmentation with the use of the random noise signal. The function should change inputs by adding or subtracting small random value of the input vector. The random value should be different for each sample and each element of the input vector. The random value should be generated with use standard normal distribution and scaled with additional parameter "rand_scale" (rand_scale*numpy.random.randn()). Add new argument "rand_scale" to training function, which specify a scaling factor for data augmentation.
3. Change training function to mini-batch training function. Add new argument "batch_size" to training function, which specify the batch size. If the batch size is equal to 1 than training will be using the stochastic gradient descent. If the batch size is equal to size of training dataset than training will be using the full batch training.

Task 2

Evaluate classification accuracy of neural network: 1 input layer, 2 hidden layers with 12 units and leaky Relu activation, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 1000,
- learning rate = 0.001,
- batch size = 8,
- rand_scale = 0.1.

Training dataset:

```
import numpy as np
n = 40

X1_1 = 1 + 4 * np.random.rand(n, 1)
X1_2 = 1 + 4 * np.random.rand(n, 1)
class1 = np.concatenate((X1_1, X1_2), axis=1)
Y1 = np.ones(n)

X0_1 = 5 + 4 * np.random.rand(n, 1)
X0_2 = 5 + 4 * np.random.rand(n, 1)
class0 = np.concatenate((X0_1, X0_2), axis=1)
Y0 = np.zeros(n)

train_x = np.concatenate((class1, class0))
train_y = np.concatenate((Y1, Y0))
```

Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 18

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data regression:

1. Create a new function for saving and loading trained model to file with 3 arguments. The first argument will be "file" with file name/path. The second argument will be "operation" with option 'save' for saving a network model to file and 'load' to load network model from file. The third optional argument of new function will be "model", where we send neural network model dictionary. The function will return model of neural network.
2. Update network training function to return loss function history for each iteration and each epochs. You can use any structure of loss history. Create a new function "history_plot", which plot changes of loss function based on loss history in two subplots: first plot of loss function in each epoch (total loss) and second plot of loss function in each iteration in each epoch. Add necessary description of the charts (labels, axis description, titles).
3. Add and integrate early stopping function in the training process. The early stop function allows to stop training process and return best network model, if the difference in loss function is small enough in few epochs or loss function is increasing value in few epochs. The function parameter will be "accuracy" and "patience". The accuracy control the difference between two successive epochs to stop training process. The patience parameter control number of epochs with no improvements to stop training process. The function should return the best neural network model, which means it returns the model with the lowest loss function value in the patient range. Add new arguments "accuracy" and "patience" to the training function to control early stop criterion.

Task 2

Evaluate loss function changes of neural network: 1 input layer, 2 hidden layers with 8 units and hyperbolic tangent, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 10000,
- learning rate = 0.001,
- early stopping: accuracy = 0.01, patience = 5 epochs

Training dataset:

```
import numpy as np
n = 20

X = np.linspace(-5, 5, n).reshape(-1, 1)
data_noise = np.random.normal(0, 0.2, n).reshape(-1, 1)
xaxis = np.sin(2 * X)
yaxis = np.cos(X)
x1, x2 = np.meshgrid(xaxis, yaxis)
y = x1 * x2 + data_noise + 3

train_x = [np.reshape(x1, (x1.size)), np.reshape(x2, (x2.size))]
train_y = np.reshape(y, (y.size))
```

Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 19

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data regression:

1. Add new neuron activation function: ELU.
2. Add new function for changing the learning rate with use Darken and Moody method. Integrate new function with training function of multi-layer neural network. Add new argument "lrate_method = 'darken_moody'" to to select this method of learning rate changing during training. The argument "l_rate" will be describing the initial learning rate value.
3. Add new function for changing the learning rate with the use Adagrad method. Integrate new function with training function of multi-layer neural network. Add new argument "lrate_method = 'adagrad'" to select this method of learning rate changing during training. The argument "l_rate" will be describing initial learning rate value of all weights.

Evaluate loss function changes of neural network: input layer, 1 hidden layers with 32 units and Elu activation, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 1000,
- initial learning rate = 0.01,
- test: fixed learning rate, Darken and Moody method, and Adagrad method.

Training dataset:

```
import numpy as np

X = np.linspace(-5, 5, n).reshape(-1, 1)
y = np.sin(2 * X) + np.cos(X) + 12
data_noise = np.random.normal(0, 0.2, n).reshape(-1, 1)
Y = y + data_noise

train_x = x
train_y = Y
```

Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 20

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data classification:

1. Create a new function to calculate classification quality measures: F1, confusion matrix for binary classification problems (see sklearn library). The new function should take two arguments "y_predict" with predict values by neural network and "y_true" with true/desired values. The function should print values of f1, precision, recall, and display confusion matrix.
2. Integrate K-fold cross-validation functionality in neural network training function. Add new argument "k_fold" to the training function. If the argument "k_fold" is 0 than training will not use cross validation methods and for positive values will performer K-fold cross-validation with "k_fold" divisions. The training function at the end of training should display minimum, average, and maxim values of quality measures f1.
3. Integrate momentum method functionality in neural network training function. The momentum method use in weight updating process a history of previous updates. Add new argument "alpha" to training function, which is a positive number called the momentum constant.

Task 2

Evaluate classification accuracy of neural network: 1 inpu t layer, 1 hidden layers with 32 units and ReLu activation, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 1000,
- learning rate = 0.001,
- K-fold cross-validation
- alpha = 0.1

Training dataset:

```
import numpy as np
n = 40

X1_1 = 0 + 4 * np.random.rand(n, 1)
X1_2 = 0 + 4 * np.random.rand(n, 1)
class1 = np.concatenate((X1_1, X1_2), axis=1)
Y1 = np.ones(n)

X0_1 = 3 + 4 * np.random.rand(n, 1)
X0_2 = 3 + 4 * np.random.rand(n, 1)
class0 = np.concatenate((X0_1, X0_2), axis=1)
Y0 = np.zeros(n)

train_x = np.concatenate((class1, class0))
train_y = np.concatenate((Y1, Y0))
```

Activity Task 2: Neural Networks for Classification and Identification, 2021

Set number: 21

Instruction: Write solution in Python to given problem. Task 1.1: 0.2 point, Task 1.2: 0.2 point, Task 1.3: 0.3 point, Task 2: 0.3 point. Deadline: 09.12.2021

Task 1

Create multi-layer neural network training procedure for data classification:

1. Add new neuron activation function: leaky Relu.
2. Add new functions for data augmentation with the use of the random noise signal. The function should change inputs by adding or subtracting small random value of the input vector. The random value should be different for each sample and each element of the input vector. The random value should be generated with use standard normal distribution and scaled with additional parameter "rand_scale" (rand_scale*numpy.random.randn()). Add new argument "rand_scale" to training function, which specify a scaling factor for data augmentation.
3. Change training function to mini-batch training function. Add new argument "batch_size" to training function, which specify the batch size. If the batch size is equal to 1 than training will be using the stochastic gradient descent. If the batch size is equal to size of training dataset than training will be using the full batch training.

Task 2

Evaluate classification accuracy of neural network: 1 input layer, 2 hidden layers with 12 units and leaky Relu activation, 1 output layer. Select the correct activation function for the output layer, loss function and other training parameters. The training parameters:

- epochs = 1000,
- learning rate = 0.001,
- batch size = 8,
- rand_scale = 0.1.

Training dataset:

```
import numpy as np
n = 40

X1_1 = 2 + 4 * np.random.rand(n, 1)
X1_2 = 1 + 4 * np.random.rand(n, 1)
class1 = np.concatenate((X1_1, X1_2), axis=1)
Y1 = np.ones(n)

X0_1 = 4 + 4 * np.random.rand(n, 1)
X0_2 = 3 + 4 * np.random.rand(n, 1)
class0 = np.concatenate((X0_1, X0_2), axis=1)
Y0 = np.zeros(n)

train_x = np.concatenate((class1, class0))
train_y = np.concatenate((Y1, Y0))
```