Ex5_NeuralNetworksInKeras

October 11, 2019

1 Load dependencies

In [1]:
```
import numpy as np
# Better to fix the seed in the beginning:
seed = 666
np.random.seed(seed)

import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split, KFold
from sklearn.datasets import load_wine
from sklearn.preprocessing import scale

from keras import backend
from keras.models import Sequential
from keras.losses import categorical_crossentropy
from keras.callbacks import LearningRateScheduler
from keras.optimizers import SGD
from keras.utils import to_categorical
from keras.layers import Dense
```

```
Using TensorFlow backend.
```

2 Utility functions (data preprocessing and KFold cross-validation)

In [2]:
```
### Scale and center features, transform labels into a one-hot encoding vector:
def preprocess_data(X, y):
### TO DO ###
    X_out = scale(X)
    y_out = to_categorical(y)
    return X_out, y_out

### Training history plot function: (this function is finished, nothing to add !)
def print_training_history(training_history, fig_idx):
```

```
    epoch_absciss = range(1, len(training_history.history['loss'])+1)
    plt.figure(fig_idx, figsize=(10, 5))
    plt.suptitle("MLP model assessment")
    plt.subplot(1, 2, 1)
    plt.plot(epoch_absciss, training_history.history['loss'])
    plt.plot(epoch_absciss, training_history.history['val_loss'])
    plt.title("Train/Validation loss")
    plt.ylabel('Loss')
    plt.xlabel('Epochs')
    plt.legend(['Train loss', 'Validation loss'], loc='best')
    plt.subplot(1, 2, 2)
    plt.plot(epoch_absciss, training_history.history['accuracy'])
    plt.plot(epoch_absciss, training_history.history['val_accuracy'])
    plt.title("Train/Validation accuracy")
    plt.ylabel('Accuracy')
    plt.xlabel('Epochs')
    plt.legend(['Train accuracy', 'Validation accuracy'], loc='best')
    plt.show()

### Select a MLP model on a list of hyper-parameters instances, via Kfold cross-valida
def KFold_model_selection(X, y, fixed_hyper_parameters, hyper_parameters_instances, nu
### TO DO ###
    def KFold_split(X, Y, num_folds, seed):
        KFold_splitter = KFold(n_splits=num_folds, shuffle=True, random_state=seed)
        X_train_folds = []
        X_val_folds = []
        Y_train_folds = []
        Y_val_folds = []
        for (kth_fold_train_idxs, kth_fold_val_idxs) in KFold_splitter.split(X, Y):
            X_train_folds.append(X[kth_fold_train_idxs])
            X_val_folds.append(X[kth_fold_val_idxs])
            Y_train_folds.append(Y[kth_fold_train_idxs])
            Y_val_folds.append(Y[kth_fold_val_idxs])
        return X_train_folds, X_val_folds, Y_train_folds, Y_val_folds

    X_train_val, X_test, Y_train_val, Y_test = train_test_split(X, y, test_size = 0.3,
    X_train_folds, X_val_folds, Y_train_folds, Y_val_folds = KFold_split(X_train_val,
    mean_val_MSEs = []
    for hyper_parameters_instance in hyper_parameters_instances:
        print("\nNow preprocessing hyper-parameter instance", hyper_parameters_instanc
        mean_val_MSE = perform_KFold_CV(X_train_folds, X_val_folds, Y_train_folds, Y_va
                                        fixed_hyper_parameters,
                                        hyper_parameters_instance)
        print("Mean validation MSE:", mean_val_MSE)
        mean_val_MSEs.append(mean_val_MSE)
    best_instance_idx = mean_val_MSEs.index(min(mean_val_MSEs))
    best_hyper_parameters_instance = hyper_parameters_instances[best_instance_idx]
    print("\n\nBest hyper-parameter instance:", best_hyper_parameters_instance)
```

```
    best_model_test_MSE = assess_MLP(X_train_val, X_test, Y_train_val, Y_test,
                                     fixed_hyper_parameters,
                                     hyper_parameters_instances[best_instance_idx])
    print("Test MSE:", best_model_test_MSE)

    return

### KFold cross-validation of a MLP model with given hyper-parameters:
def perform_KFold_CV(X_train_folds, X_val_folds, Y_train_folds, Y_val_folds, fixed_hyp
### TO DO ###
    val_fold_MSEs = []
    # For each fold, assess a surrogate model with fixed hyper-parameters:
    cmpt = 0
    for X_train_fold, X_val_fold, Y_train_fold, Y_val_fold in zip(X_train_folds, X_val
        val_fold_MSE = assess_MLP(X_train_fold, X_val_fold, Y_train_fold, Y_val_fold,
        cmpt += 1
#         print("Surrogate model", str(cmpt) + "/" + str(len(X_val_folds)), "validatio
        val_fold_MSEs.append(val_fold_MSE)
    # Compute the mean validation MSE between all the folds:
    mean_val_MSE = np.mean(val_fold_MSE)
    return mean_val_MSE
### Fit and evaluate a MLP model with given hyper-parameters:
def assess_MLP(X_train, X_test, y_train, y_test, fixed_hyper_parameters, hyper_paramet
### TO DO ###
    in_shape = X_train.shape[1]
    num_y_classes = y_train.shape[1]
    myMLP = build_MLP(in_shape, num_y_classes, hyper_parameters_instance)
    myMLP.fit(X_train, y_train,\
              batch_size=fixed_hyper_parameters["train batch size"],\
              epochs=fixed_hyper_parameters["epochs"])
    mytest_loss, mytest_accuracy = myMLP.evaluate(X_test, y_test, fixed_hyper_parameter
    return mytest_loss
```

3 MLP (multi-layer perceptron) builder

In [3]:
```
### Build a simple fully-connected MLP with SGD model:
def build_MLP(input_shape, num_classes, hyper_parameters_instance=None): #add hyper pa
    MLP = Sequential()
    # Hidden layers (fully connected/dense):
    if hyper_parameters_instance==None:
        MLP.add(Dense(10, activation='relu'))
    else:
        if hyper_parameters_instance["HiddenLayerActivationRelu"]==True:
            MLP.add(Dense(10, activation='relu'))
            if hyper_parameters_instance["Flag"]==True:
                MLP.add(Dense(10, activation='relu'))
        else:
```

```
                MLP.add(Dense(10, activation='sigmoid'))
                if hyper_parameters_instance["Flag"]==True:
                    MLP.add(Dense(10, activation='sigmoid'))
            # Output layer (fully-connected/dense):
            MLP.add(Dense(units=num_classes, activation='softmax'))
        #     sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
            MLP.compile(loss='categorical_crossentropy', optimizer="SGD", metrics=['accuracy'])
            return MLP
```

4 Load and preprocess the Wine dataset

```
In [4]: # Load the Wine dataset:
        X = load_wine().data
        y = load_wine().target
        # Get the shape of the individual feature vectors in the dataset:
        input_shape = X.shape[1]
        # Get the number of classes:
        num_classes = (np.unique(y)).shape[0]
        # Preprocess data: (implement the preprocess_data function)
        X, y = preprocess_data(X, y)
```

5 Train, validate and evaluate a MLP model, and plot the results:

```
In [5]: # Number of epochs:
        num_epochs = 20
        # Train batch size:
        train_batch_size = 16
        # Split data into train/val/test sets:
        X_train_val, X_test, Y_train_val, Y_test = train_test_split(X, y, test_size = 0.3, rand

        X_train, X_val, Y_train, Y_val = train_test_split(X_train_val, Y_train_val, test_size =

        # Load an MLP:
        model = build_MLP(input_shape, num_classes)
        # print(model.summary())
        # Train and validate MLP, store the training history in a variable:
        training_history = model.fit(X_train, Y_train,
                                     batch_size = train_batch_size, epochs = num_epochs,
                                     validation_data = [X_val , Y_val])
        model.summary()
        # Evaluate the model:
        test_loss, test_accuracy = model.evaluate(X_test, Y_test, train_batch_size)
        print("Test loss:", test_loss)
        print("Test accuracy:", test_accuracy)
        # Plot training history:
        # print_training_history(training_history, fig_idx=1)
```

```
Train on 99 samples, validate on 25 samples
Epoch 1/20
99/99 [==============================] - 0s 594us/step - loss: 1.0801 - accuracy: 0.4141 - val_
Epoch 2/20
99/99 [==============================] - 0s 91us/step - loss: 1.0000 - accuracy: 0.4646 - val_l
Epoch 3/20
99/99 [==============================] - 0s 81us/step - loss: 0.9341 - accuracy: 0.5556 - val_l
Epoch 4/20
99/99 [==============================] - 0s 90us/step - loss: 0.8765 - accuracy: 0.6061 - val_l
Epoch 5/20
99/99 [==============================] - 0s 100us/step - loss: 0.8231 - accuracy: 0.6364 - val_
Epoch 6/20
99/99 [==============================] - 0s 92us/step - loss: 0.7735 - accuracy: 0.6667 - val_l
Epoch 7/20
99/99 [==============================] - 0s 90us/step - loss: 0.7332 - accuracy: 0.7273 - val_l
Epoch 8/20
99/99 [==============================] - 0s 91us/step - loss: 0.6946 - accuracy: 0.7778 - val_l
Epoch 9/20
99/99 [==============================] - 0s 91us/step - loss: 0.6590 - accuracy: 0.7879 - val_l
Epoch 10/20
99/99 [==============================] - 0s 110us/step - loss: 0.6262 - accuracy: 0.8283 - val_
Epoch 11/20
99/99 [==============================] - 0s 111us/step - loss: 0.5974 - accuracy: 0.8788 - val_
Epoch 12/20
99/99 [==============================] - 0s 101us/step - loss: 0.5700 - accuracy: 0.8889 - val_
Epoch 13/20
99/99 [==============================] - 0s 100us/step - loss: 0.5446 - accuracy: 0.9091 - val_
Epoch 14/20
99/99 [==============================] - 0s 91us/step - loss: 0.5211 - accuracy: 0.9192 - val_l
Epoch 15/20
99/99 [==============================] - 0s 101us/step - loss: 0.5005 - accuracy: 0.9192 - val_
Epoch 16/20
99/99 [==============================] - 0s 101us/step - loss: 0.4798 - accuracy: 0.9192 - val_
Epoch 17/20
99/99 [==============================] - 0s 91us/step - loss: 0.4600 - accuracy: 0.9192 - val_l
Epoch 18/20
99/99 [==============================] - 0s 101us/step - loss: 0.4419 - accuracy: 0.9192 - val_
Epoch 19/20
99/99 [==============================] - 0s 110us/step - loss: 0.4245 - accuracy: 0.9192 - val_
Epoch 20/20
99/99 [==============================] - 0s 103us/step - loss: 0.4090 - accuracy: 0.9192 - val_
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 10)                140
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 33
```

```
=================================================================
Total params: 173
Trainable params: 173
Non-trainable params: 0
_________________________________________________________________
54/54 [==============================] - 0s 55us/step
Test loss: 0.38749685883522034
Test accuracy: 0.9814814925193787
```

6 Model selection of our MLP

```
In [7]: # Number of folds in KFold cross-validation:
        num_folds = 5
        # Number of epochs:
        num_epochs = 5
        # Train batch size:
        train_batch_size = 16
        # Create the list of hyper-parameters instances:
        hyper_parameters_instances = [{"Flag": True, "HiddenLayerActivationRelu": True},
                                      {"Flag": True, "HiddenLayerActivationRelu": False},
                                      {"Flag": False, "HiddenLayerActivationRelu": True},
                                      {"Flag": False, "HiddenLayerActivationRelu": False}]
        # Also store the fixed hyper-parameters:
        fixed_hyper_parameters = {"epochs": num_epochs,
                                  "train batch size": train_batch_size}
        # Select model with KFold cross-validation:
        KFold_model_selection(X, y, fixed_hyper_parameters, hyper_parameters_instances, num_fo
```

```
Now preprocessing hyper-parameter instance {'Flag': True, 'HiddenLayerActivationRelu': True}
Epoch 1/5
99/99 [==============================] - 0s 525us/step - loss: 1.1335 - accuracy: 0.3232
Epoch 2/5
99/99 [==============================] - 0s 60us/step - loss: 1.0959 - accuracy: 0.4242
Epoch 3/5
99/99 [==============================] - 0s 81us/step - loss: 1.0611 - accuracy: 0.4646
Epoch 4/5
99/99 [==============================] - 0s 70us/step - loss: 1.0278 - accuracy: 0.5354
Epoch 5/5
99/99 [==============================] - 0s 80us/step - loss: 0.9964 - accuracy: 0.5657
25/25 [==============================] - 0s 640us/step
Epoch 1/5
99/99 [==============================] - 0s 465us/step - loss: 1.2935 - accuracy: 0.3737
Epoch 2/5
99/99 [==============================] - 0s 75us/step - loss: 1.2173 - accuracy: 0.3737
Epoch 3/5
```

```
99/99 [==============================] - 0s 81us/step - loss: 1.1460 - accuracy: 0.3535
Epoch 4/5
99/99 [==============================] - 0s 91us/step - loss: 1.0842 - accuracy: 0.3434
Epoch 5/5
99/99 [==============================] - 0s 81us/step - loss: 1.0268 - accuracy: 0.3636
25/25 [==============================] - 0s 637us/step
Epoch 1/5
99/99 [==============================] - 0s 499us/step - loss: 1.3209 - accuracy: 0.2020
Epoch 2/5
99/99 [==============================] - 0s 81us/step - loss: 1.2384 - accuracy: 0.2020
Epoch 3/5
99/99 [==============================] - 0s 71us/step - loss: 1.1746 - accuracy: 0.2727
Epoch 4/5
99/99 [==============================] - 0s 81us/step - loss: 1.1287 - accuracy: 0.2727
Epoch 5/5
99/99 [==============================] - 0s 90us/step - loss: 1.0901 - accuracy: 0.3333
25/25 [==============================] - 0s 637us/step
Epoch 1/5
99/99 [==============================] - 0s 488us/step - loss: 1.0993 - accuracy: 0.3333
Epoch 2/5
99/99 [==============================] - 0s 71us/step - loss: 1.0279 - accuracy: 0.4848
Epoch 3/5
99/99 [==============================] - 0s 60us/step - loss: 0.9782 - accuracy: 0.6263
Epoch 4/5
99/99 [==============================] - 0s 71us/step - loss: 0.9322 - accuracy: 0.6970
Epoch 5/5
99/99 [==============================] - 0s 80us/step - loss: 0.8877 - accuracy: 0.7273
25/25 [==============================] - 0s 639us/step
Epoch 1/5
100/100 [==============================] - 0s 480us/step - loss: 1.2854 - accuracy: 0.1000
Epoch 2/5
100/100 [==============================] - 0s 60us/step - loss: 1.2288 - accuracy: 0.1400
Epoch 3/5
100/100 [==============================] - 0s 70us/step - loss: 1.1825 - accuracy: 0.1700
Epoch 4/5
100/100 [==============================] - 0s 70us/step - loss: 1.1448 - accuracy: 0.2000
Epoch 5/5
100/100 [==============================] - 0s 79us/step - loss: 1.1106 - accuracy: 0.2300
24/24 [==============================] - 0s 621us/step
Mean validation MSE: 1.0684965054194133

Now preprocessing hyper-parameter instance {'Flag': True, 'HiddenLayerActivationRelu': False}
Epoch 1/5
99/99 [==============================] - 0s 453us/step - loss: 1.1183 - accuracy: 0.4040
Epoch 2/5
99/99 [==============================] - 0s 76us/step - loss: 1.1093 - accuracy: 0.4040
Epoch 3/5
99/99 [==============================] - 0s 70us/step - loss: 1.1021 - accuracy: 0.4040
```

```
Epoch 4/5
99/99 [==============================] - 0s 83us/step - loss: 1.0950 - accuracy: 0.4040
Epoch 5/5
99/99 [==============================] - 0s 81us/step - loss: 1.0881 - accuracy: 0.4040
25/25 [==============================] - 0s 798us/step
Epoch 1/5
99/99 [==============================] - 0s 463us/step - loss: 1.1907 - accuracy: 0.4242
Epoch 2/5
99/99 [==============================] - 0s 80us/step - loss: 1.1770 - accuracy: 0.3737
Epoch 3/5
99/99 [==============================] - 0s 71us/step - loss: 1.1673 - accuracy: 0.4040
Epoch 4/5
99/99 [==============================] - 0s 81us/step - loss: 1.1543 - accuracy: 0.4242
Epoch 5/5
99/99 [==============================] - 0s 76us/step - loss: 1.1409 - accuracy: 0.4444
25/25 [==============================] - 0s 640us/step
Epoch 1/5
99/99 [==============================] - 0s 556us/step - loss: 1.2032 - accuracy: 0.4242
Epoch 2/5
99/99 [==============================] - 0s 81us/step - loss: 1.1823 - accuracy: 0.4242
Epoch 3/5
99/99 [==============================] - 0s 90us/step - loss: 1.1700 - accuracy: 0.4242
Epoch 4/5
99/99 [==============================] - 0s 91us/step - loss: 1.1558 - accuracy: 0.4242
Epoch 5/5
99/99 [==============================] - 0s 81us/step - loss: 1.1447 - accuracy: 0.4242
25/25 [==============================] - 0s 640us/step
Epoch 1/5
99/99 [==============================] - 0s 505us/step - loss: 1.1748 - accuracy: 0.2828
Epoch 2/5
99/99 [==============================] - 0s 80us/step - loss: 1.1560 - accuracy: 0.2828
Epoch 3/5
99/99 [==============================] - 0s 91us/step - loss: 1.1380 - accuracy: 0.2828
Epoch 4/5
99/99 [==============================] - 0s 71us/step - loss: 1.1280 - accuracy: 0.2828
Epoch 5/5
99/99 [==============================] - 0s 82us/step - loss: 1.1177 - accuracy: 0.2828
25/25 [==============================] - 0s 780us/step
Epoch 1/5
100/100 [==============================] - 0s 464us/step - loss: 1.0787 - accuracy: 0.4300
Epoch 2/5
100/100 [==============================] - 0s 59us/step - loss: 1.0784 - accuracy: 0.4300
Epoch 3/5
100/100 [==============================] - 0s 79us/step - loss: 1.0770 - accuracy: 0.4300
Epoch 4/5
100/100 [==============================] - 0s 70us/step - loss: 1.0771 - accuracy: 0.4300
Epoch 5/5
100/100 [==============================] - 0s 90us/step - loss: 1.0759 - accuracy: 0.4300
```

```
24/24 [==============================] - 0s 715us/step
Mean validation MSE: 1.1514487266540527

Now preprocessing hyper-parameter instance {'Flag': False, 'HiddenLayerActivationRelu': True}
Epoch 1/5
99/99 [==============================] - 0s 390us/step - loss: 1.1256 - accuracy: 0.4040
Epoch 2/5
99/99 [==============================] - 0s 60us/step - loss: 1.0582 - accuracy: 0.4444
Epoch 3/5
99/99 [==============================] - 0s 70us/step - loss: 0.9996 - accuracy: 0.5152
Epoch 4/5
99/99 [==============================] - 0s 60us/step - loss: 0.9506 - accuracy: 0.5657
Epoch 5/5
99/99 [==============================] - 0s 80us/step - loss: 0.9015 - accuracy: 0.6061
25/25 [==============================] - 0s 621us/step
Epoch 1/5
99/99 [==============================] - 0s 424us/step - loss: 1.4813 - accuracy: 0.3636
Epoch 2/5
99/99 [==============================] - 0s 80us/step - loss: 1.3551 - accuracy: 0.4040
Epoch 3/5
99/99 [==============================] - 0s 71us/step - loss: 1.2429 - accuracy: 0.4545
Epoch 4/5
99/99 [==============================] - 0s 81us/step - loss: 1.1510 - accuracy: 0.4949
Epoch 5/5
99/99 [==============================] - 0s 70us/step - loss: 1.0688 - accuracy: 0.5354
25/25 [==============================] - 0s 646us/step
Epoch 1/5
99/99 [==============================] - 0s 388us/step - loss: 1.3316 - accuracy: 0.4242
Epoch 2/5
99/99 [==============================] - 0s 71us/step - loss: 1.2398 - accuracy: 0.4444
Epoch 3/5
99/99 [==============================] - 0s 71us/step - loss: 1.1485 - accuracy: 0.4646
Epoch 4/5
99/99 [==============================] - 0s 91us/step - loss: 1.0758 - accuracy: 0.4848
Epoch 5/5
99/99 [==============================] - 0s 60us/step - loss: 1.0191 - accuracy: 0.5354
25/25 [==============================] - 0s 645us/step
Epoch 1/5
99/99 [==============================] - 0s 444us/step - loss: 1.4097 - accuracy: 0.3838
Epoch 2/5
99/99 [==============================] - 0s 91us/step - loss: 1.3167 - accuracy: 0.4040
Epoch 3/5
99/99 [==============================] - 0s 70us/step - loss: 1.2373 - accuracy: 0.4141
Epoch 4/5
99/99 [==============================] - 0s 90us/step - loss: 1.1693 - accuracy: 0.4646
Epoch 5/5
99/99 [==============================] - 0s 90us/step - loss: 1.1058 - accuracy: 0.4747
25/25 [==============================] - 0s 639us/step
```

```
Epoch 1/5
100/100 [==============================] - 0s 479us/step - loss: 1.4633 - accuracy: 0.5200
Epoch 2/5
100/100 [==============================] - 0s 89us/step - loss: 1.3201 - accuracy: 0.5700
Epoch 3/5
100/100 [==============================] - 0s 90us/step - loss: 1.1973 - accuracy: 0.5900
Epoch 4/5
100/100 [==============================] - 0s 90us/step - loss: 1.0910 - accuracy: 0.6100
Epoch 5/5
100/100 [==============================] - 0s 86us/step - loss: 1.0013 - accuracy: 0.6400
24/24 [==============================] - 0s 705us/step
Mean validation MSE: 0.8710221449534098

Now preprocessing hyper-parameter instance {'Flag': False, 'HiddenLayerActivationRelu': False}
Epoch 1/5
99/99 [==============================] - 0s 398us/step - loss: 1.2492 - accuracy: 0.2525
Epoch 2/5
99/99 [==============================] - 0s 71us/step - loss: 1.2046 - accuracy: 0.2525
Epoch 3/5
99/99 [==============================] - 0s 61us/step - loss: 1.1662 - accuracy: 0.2525
Epoch 4/5
99/99 [==============================] - 0s 80us/step - loss: 1.1338 - accuracy: 0.2626
Epoch 5/5
99/99 [==============================] - 0s 50us/step - loss: 1.1050 - accuracy: 0.2626
25/25 [==============================] - 0s 598us/step
Epoch 1/5
99/99 [==============================] - 0s 394us/step - loss: 1.4934 - accuracy: 0.3030
Epoch 2/5
99/99 [==============================] - 0s 70us/step - loss: 1.4151 - accuracy: 0.3030
Epoch 3/5
99/99 [==============================] - 0s 60us/step - loss: 1.3494 - accuracy: 0.3030
Epoch 4/5
99/99 [==============================] - 0s 90us/step - loss: 1.2814 - accuracy: 0.2929
Epoch 5/5
99/99 [==============================] - 0s 60us/step - loss: 1.2263 - accuracy: 0.2929
25/25 [==============================] - 0s 659us/step
Epoch 1/5
99/99 [==============================] - 0s 435us/step - loss: 1.4090 - accuracy: 0.2828
Epoch 2/5
99/99 [==============================] - 0s 61us/step - loss: 1.3642 - accuracy: 0.2727
Epoch 3/5
99/99 [==============================] - 0s 70us/step - loss: 1.3155 - accuracy: 0.2828
Epoch 4/5
99/99 [==============================] - 0s 71us/step - loss: 1.2736 - accuracy: 0.3030
Epoch 5/5
99/99 [==============================] - 0s 70us/step - loss: 1.2349 - accuracy: 0.3434
25/25 [==============================] - 0s 598us/step
Epoch 1/5
```

```
99/99 [==============================] - 0s 444us/step - loss: 1.1224 - accuracy: 0.1919
Epoch 2/5
99/99 [==============================] - 0s 70us/step - loss: 1.1050 - accuracy: 0.2424
Epoch 3/5
99/99 [==============================] - 0s 70us/step - loss: 1.0891 - accuracy: 0.2727
Epoch 4/5
99/99 [==============================] - 0s 60us/step - loss: 1.0731 - accuracy: 0.2929
Epoch 5/5
99/99 [==============================] - 0s 81us/step - loss: 1.0562 - accuracy: 0.3030
25/25 [==============================] - 0s 678us/step
Epoch 1/5
100/100 [==============================] - 0s 390us/step - loss: 0.9159 - accuracy: 0.5100
Epoch 2/5
100/100 [==============================] - 0s 60us/step - loss: 0.9079 - accuracy: 0.5300
Epoch 3/5
100/100 [==============================] - 0s 60us/step - loss: 0.9010 - accuracy: 0.5400
Epoch 4/5
100/100 [==============================] - 0s 80us/step - loss: 0.8929 - accuracy: 0.5600
Epoch 5/5
100/100 [==============================] - 0s 60us/step - loss: 0.8859 - accuracy: 0.5700
24/24 [==============================] - 0s 665us/step
Mean validation MSE: 0.8928737640380859


Best hyper-parameter instance: {'Flag': False, 'HiddenLayerActivationRelu': True}
Epoch 1/5
124/124 [==============================] - 0s 338us/step - loss: 1.2985 - accuracy: 0.3468
Epoch 2/5
124/124 [==============================] - 0s 64us/step - loss: 1.1620 - accuracy: 0.4113
Epoch 3/5
124/124 [==============================] - 0s 56us/step - loss: 1.0444 - accuracy: 0.4758
Epoch 4/5
124/124 [==============================] - 0s 72us/step - loss: 0.9464 - accuracy: 0.5323
Epoch 5/5
124/124 [==============================] - 0s 64us/step - loss: 0.8625 - accuracy: 0.6290
54/54 [==============================] - 0s 295us/step
Test MSE: 0.8413145630447952
```