HW8

The *successive halving* hyperparameter search technique is as follows:

0. Specify a hyperparameter grid.
1. Sample a set of hyperparameter combinations from this grid.
2. Train models for T epochs under each hyperparameter combination.
3. Assess the validation losses for each combination.
4. Stop training for the half of combinations which have the worst validation losses.
5. Continue 2–4 until only one combination remains.

Complete the code to implement this procedure for the `hidden_layer_sizes` of `sklearn`'s `MLPClassifier` as applied to the `iris` dataset. Print out the selected `hidden_layer_sizes` when you are done.

In [51]:
```
import numpy as np
from sklearn import datasets
from sklearn.neural_network import MLPClassifier
import random
from sklearn.metrics import log_loss
from sklearn.model_selection import train_test_split

# The training and validation sets.
iris = datasets.load_iris()
X = iris.data
y = iris.target
X_train, X_val, y_train, y_val = train_test_split(
    X, y, test_size=0.3, shuffle=True, stratify = y)
```

Steps 0–1

In [105…
```
# Step 0: Specify the grid.
grid = [(2**k,)*2**l for k in np.arange(0, 4) for l in np.arange(0,4)]
print("Search Space:", grid)

# Step 1: Sample.
random.seed(553)
hyper = random.sample(grid, 8)
print("The Random Sample:", hyper)
```

```
Search Space: [(1,), (1, 1), (1, 1, 1, 1), (1, 1, 1, 1, 1, 1, 1, 1), (2,), (2, 2), (2, 2, 2, 2), (2, 2, 2, 2, 2, 2, 2, 2), (4,),
(4, 4), (4, 4, 4, 4), (4, 4, 4, 4, 4, 4, 4, 4), (8,), (8, 8), (8, 8, 8, 8), (8, 8, 8, 8, 8, 8, 8, 8)]
The Random Sample: [(8, 8, 8, 8, 8, 8, 8, 8), (1,), (2, 2, 2, 2, 2, 2, 2, 2), (1, 1, 1, 1), (8, 8), (8, 8, 8, 8), (2, 2, 2, 2),
(4, 4)]
```

Steps 2—5

To train an `MLPClassifier` for an epoch, use the `partial_fit(X, y, classes=None)` method. Make sure to read the description of the `classes` parameter in the documentation.

The loss to evlauate on the validation set is `sklearn.metrics.log_loss`.

Take $T = 100$ as the number of epochs for step 2.

In [144…
```
# Steps 2–5:
T = 100 # Number of epochs for step 2.

# YOUR CODE HERE:
hyperparameters = []
for i in range(len(hyper)):
    clf = MLPClassifier(hidden_layer_sizes = (hyper[i]))
    hyperparameters.append([0, clf, hyper[i]])
while (len(hyperparameters) > 1):
    for h in range(len(hyperparameters)):
        clf = hyperparameters[h][1]
        for t in range(T):
            clf.partial_fit(X_train, y_train, classes=np.unique(y))
        hyperparameters[h]=([log_loss(y_val, clf.predict_proba(X_val)), clf, hyperparameters[h][2]])
    hyperparameters.sort()
    hyperparameters = hyperparameters[:len(hyperparameters)//2]
print(hyperparameters[0][2])
```

```
(8, 8, 8, 8, 8, 8, 8, 8)
```