Artificial Intelligence Laboratory 4: Learning

DT8012 (HT19) Halmstad University

17th of December 2019

This lab is designed to introduce you the use of machine learning algorithms. By the end of this lab session you will learn how to use supervised learning algorithms to solve:

- classification problems: learning categories
- regression problems: learning function values

For this lab, you will use learning algorithms in two different domains:

- **Vehicle fleet management:** using signal data collected from different sensors in vehicles, you will be able to classify the driver performance (classification), and also finding a suitable model for fuel consumption (regression).
- **Poker Player**: using historical data of player's betting actions you can predict his current action.

Before you arrive at the lab:

You should read through the entire document and look up anything you don't quite remember from the lectures.

After the lab:

Within three weeks after this lab session you must hand in your report. Your report should explain your results, and how you achieved them. Make sure you include any relevant information to your explanation.
The deadline for your report submission is Jan 8th. The reports which are sent after Jan 8th will be graded at the end of February or at the end of May. This means there are only three fixed dates for grading. If you submit your reports after these dates, you need to wait until the next time that the course is given to get your grade.

Environment setup

The only required software tool is Python. You need to install 'scikit-learn' library for this lab. Scikit-learn is a library for implementing a number of machine learning algorithms in Python. If you are using IDE like PyCharm, it can be easily installed within 'Preference -> Project Interpreter' menu.

For more information about scikit-learn see: http://scikit-learn.org/stable/ and http://scikit-learn.org/stable/user_guide.html

Task 1: Vehicle fleet management

Introduction

In this task, you will use vehicle diagnostic data ***Lab4Data.csv*** as an input to your learning algorithms. The data is a .csv file that contains the attributes name in the first row, followed by samples (2000 samples). The attributes are some of the various signals that can be collected from a vehicle. All attributes are numeric (Figure 1 shows part of this data file).

Acceleration	Gender	Age	LateralAcceleration	VehicleSpeed	GearChanges	EngineSpeed	FuelConsum	Weight	DriverPerformance
0.052966	1	36.308	0.24189	83.546	5.5275	1393.9	33.878	32.026	2
0.083355	2	29.51	0.1991	75.305	2.0625	1264.1	28.645	27.234	3
0.16115	2	25.452	0.39129	78.012	7.0441	1512.3	35.743	30.146	1
0.088723	2	31.077	0.12482	75.598	1.2205	1032.6	31.902	32.151	3
0.18522	1	30.559	0.30736	82.278	2.0518	1745.3	29.192	27.475	2
0.019916	1	34.094	0.038242	83.967	1.174	1336.2	31.275	31.387	3
-0.0002927	1	29.806	0.1244	76.461	2.473	1226.3	31.166	29.024	3
0.016495	1	31.95	0.24034	73.093	0.83174	1413.8	30.851	30.669	3
0.024025	2	35.121	0.19802	69.101	1.7063	1230.4	31.949	30.101	3

Figure 1- Part of the input data file (Lab5Data.csv)

In this data file, the last column ("DriverPerformance") represents the concept that you need to learn from for classifying the driver performance. It contains three values corresponding to three different class labels. In addition, the column "FuelConsumption" is the concept that you need to learn from to find a regression model for fuel consumption.

You will be using a number of machine learning techniques such as support vector machine (SVM), neural network (NN), nearest neighbor, linear regression, etc. They all use inductive learning to construct a model that can be used for predicting variables of interest based on other attributes. However the models and/or ways to find the models differ significantly.

Example.py contains a simple code about how to use this library and SVM to classify the driver performance (see Figure 1).

Figure 1.A: The data is split into training set (80% of the data) and test set (20% of the data).

Train_set, Test_set = train_test_split(Data, test_size=0.2)

Figure 1.B: Then SVM is used to classify the driver performance using support vector classifier (SVC) with a linear kernel. This function takes the attribute values (all the columns except the "DriverPerformance") and returns a "decision" that estimate the output values of the driver performance.

svc = svm.SVC(kernel='linear', C=1.0).fit(Input_train, Target_train)

Figure 1.C: In order to evaluate the performance of the classifier the function "predict" is used on test set data. The outcomes of the predictions are compared with the true values in "Target_test" to evaluate the accuracy of the prediction method.

PredictedOutcome = svc.predict(Input_test)

```python
import numpy as np
from sklearn.cross_validation import train_test_split
from sklearn import svm

# import data
Data = np.loadtxt(open("Lab5Data.csv", "rb"), delimiter=";", skiprows=1)
print ('****************************************')
print('Length of Total Data:', len(Data))

Train_set, Test_set = train_test_split(Data, test_size=0.2)
Input_train = Train_set[:, :8]     # input features
Target_train = Train_set[:, 9]  # output labels

Input_test = Test_set[:, :8]
Target_test = Test_set[:, 9]
print ('****************************************')
print('Length of Train Data:', len(Train_set))
print('Length of Test Data', len(Test_set))
```

A

```python
svc = svm.SVC(kernel='linear', C=1.0).fit(Input_train, Target_train)
#svc = svm.SVC(kernel='rbf', degree=3, C=1.0).fit(Input_train, Target_train)
print ('****************************************')
print(svc)
```

B

```python
PredictedOutcome = svc.predict(Input_test)
Number_of_Correct_Predictions = len([i for i, j in zip(PredictedOutcome, Target_test) if i == j])

print ('****************************************')
print('Number of Correct Predictions:', Number_of_Correct_Predictions, 'Out_of:', len(PredictedOutcome),
      'Number of Test Data')
print('Accuracy of Prediction in Percent:', (Number_of_Correct_Predictions/float(len(PredictedOutcome)))*100)
```

C

Parameter tuning

"In the context of machine learning, hyperparameter optimization or model selection is the problem of choosing a set of hyperparameters for a learning algorithm, usually with the goal of optimizing a measure of the algorithm's performance on an independent data set. Often cross-validation is used to estimate this generalization performance. Hyperparameter optimization contrasts with actual learning problems, which are also often cast as optimization problems, but optimize a loss function on the training set alone." https://en.wikipedia.org/wiki/Hyperparameter_optimization

Parameter tuning of a machine learning algorithm is a process which one goes through to optimize the hyperparameters (e.g. the number of neighbors in k-Nearest Neighbor) that improve the accuracy. In this case, we are not trying to improve the learning algorithm but to optimize the hyperparameters value that gives the lower average error (higher accuracy) using the same algorithm.

Task 1 "A": Implement k-NN classifier

- Implement k-Nearest Neighbor (k-NN) algorithm to classify driver performance in **Lab5Data.csv**. For this task, you are **not allowed** to use the k-NN function in scikit-learn library (you need to implement this classification algorithm by your own using python).

In your report, you need to: a) explain the algorithm and b) write the evaluation results of the classification algorithm.

Task 1 "B": Investigate different classification algorithms

- Try 3 different classification algorithms from scikit-learn library. For this task, you are allowed to use the available functions in scikit-learn library.

In your report, you need to write the evaluation results of each classification algorithm.

Task 1 "C": Parameter tuning

- Use cross-validation and try different parameters for k-NN classifiers: change the number of neighbors and change the distance measurement method.

In your report, you need to write the parameters you have changed with the corresponding. Specify which parameters give the best result?

Task 1 "D":

- **(extra credit)** Compare the results (task 1B with the results in Task 1C) both in terms of accuracy and also finding a point which is classified differently using each of the algorithms. Explain your results.

Task 1 "E": Implement k-NN regression

- Modify your K-Nearest Neighbor (k-NN) algorithm in task 1A to find a model to fit the "FuelConsumption" data in **Lab4Data.csv.** For this task you are **not allowed** to use the k-NN function in scikit-learn library (you need to implement this regression model by your own using python).

In your report, you need to explain the algorithm and write the evaluation results of the regression model in your report.

Task 1 "F": Investigate different regression algorithms

- Try 3 different regression algorithms from scikit-learn library. For this task, you are allowed to use the available functions in scikit-learn library.

In your report, you need to write the evaluation results of each regression model in your report.

Task 1 "G": Parameter tuning

- Use cross-validation and try different parameters for k-NN regression.

In your report, you need to write the parameters you changed with the corresponding results in your report. Specify which parameters give the best result?

Task 1 "H":

- **(extra credit)** Compare the results (task 1F with the results in Task 1G) in terms of accuracy. Explain your results.

Task 2: Poker Project

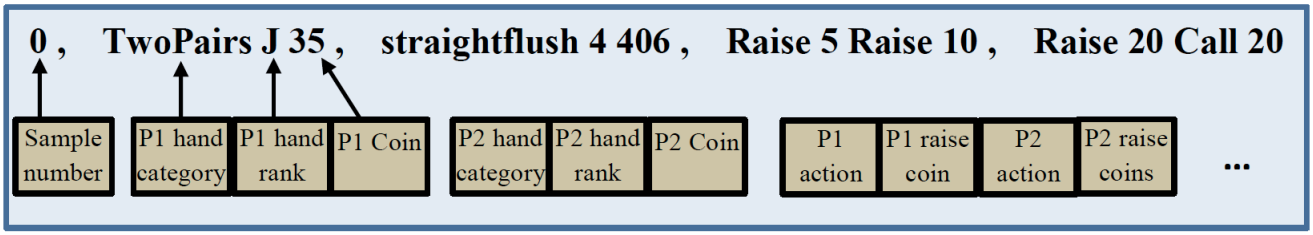
Introduction

You can also apply machine learning algorithms to the poker agent project. **Lab4PokerData.txt** contains 200 poker game samples. Each sample is a part of betting between two agents. Data recorded in the following format:

#Sample, P1Hand P1HandRank P1Coin, P2Hand P2HandRank P2Coin, P1Action P1BettingAmount P2Action P2BettingAmount, P1Action P1BettingAmount P2Action P2BettingAmount,...

Each line starts with sample number. **P1Hand** represents player1's hand category, **P1HandRank** represents the rank of the hand category (e.g if the hand is 'highcard', **P1HandRank** is highest card within; if the hand is 'OnePair', P1HandRank is the rank of the pair), **P1Coin** represents the amount of money that player1 possess. **P2Hand**, **P2HandRank**, **P2Coin** are the same information for p2. This is followed by the action sequence of two agents.

Example:



Your task in this part of the lab is to use the data for predicting the agents' final actions (call or fold). In order to do this, you need to create attributes from the available information such as hand category, hand rank, amount of money, etc. The action of the agents represents the concept that you need to learn from the data.

Task 2 "A":

- Come up with at least two more attributes (numerical attributes) e.g. one of the attributes can be the average amount of money raise by an agent.

Task 2 "B":

- Try using at least 3 different learning algorithms (k-NN and two other algorithms) to predict the final action of the other player, i.e., call, or fold.

  In your report, you need to write the evaluation results of each learning algorithm in your report.

Task 2 "C":

- Use cross-validation and try different parameters for k-NN classifiers: change the number of neighbors and change the distance measurement method.

  In your report, you need to write the parameters you have changed with the corresponding results in your report. Specify which parameters give the best result?

Grading Criteria

- Pass: Complete all the tasks except tasks 1D and 1H.
- Extra credit: Complete all the tasks including the tasks 1D and 1H.