Artificial Intelligence Laboratory 5: Learning

DT8012 (HT16) Halmstad University

12th of December 2016

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 two weeks after this lab session you must hand in your **report** and **source code**. Your report should explain your results, and **how** you achieved them. Make sure you include any relevant information to your explanation. Your code, developed in order to solve the exercises proposed here, should be well commented and self-explanatory.

The deadline for your report and code submission is **December 27**. Send your report as a pdf document and your code as one zip file to hassan.nemati@hh.se. The name of your files should include your first and last names, e.g. HassanNematiReport.pdf, HassanNematiCode.zip. Write the name of the lab in the title of your email e.g. Lab 5. Do not forget to write your name and your group mate full name.

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 ***Lab5Data.csv*** as an input to your learning algorithms. The data is a .csv file that contains feature name in first row, followed by samples (2000 samples). The features are some examples of various signals that can be collected from a vehicle. All features 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 in three classes. 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 experience (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.

In the **Example.py** 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)

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)

In order to evaluate the performance of the classifier the function "predict" is used on test set data.

PredictedOutcome = svc.predict(Input_test)

And the outcomes of the predictions are compared with the true values in "Target_test" to evaluate the accuracy of the prediction method.

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). You need to explain the algorithm and write the evaluation results of the classification algorithm in your report.

Task 1 "B": Investigate different classification algorithms

- Try 3 different classification algorithms from scikit-learn library. You need to write the evaluation results of each classification algorithm in your report.

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. You need to write the parameters you changed with the corresponding results in your report. 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 **Lab5Data.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). 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. 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. 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. **Lab5_poker_data.txt** contains 200 poker game samples. Each sample is a part of betting between two agents. Data using 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.

Task 2: Poker hand strength estimation

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.

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. You need to write the evaluation results of each learning algorithm in your report.

C) Use cross-validation and try different parameters for k-NN classifiers: change the number of neighbors and change the distance measurement method. You need to write the parameters you changed with the corresponding results in your report. Specify which parameters give the best result?

D) **(extra credit)** Compare the results (task 2B with the results in Task 2C) both in terms of accuracy and also finding a point which is classified differently using each of the algorithms. Explain your results.

Grading Criteria

- Pass: Complete all the tasks except task 1D, 1H, and 2D.
- Extra credit: Complete the task 1D, 1H, and 2D (within the 2 weeks deadline).
- Submission deadline December 27.