Numerical Recipes

Report 2

Data classification using machine learning.

The hand-in deadline for this report is:

Friday 1st December @ 23.59

Aim

Demonstrate the use of **machine learning classification** to predict the *Weather Type* based on a sample of collected ground observations.

Introduction

You will be using two weeks of observation data from 24th October to 7th November (inclusive) collected from the **UK Met Office DataPoint** service. Observations are made hourly at each of the 126 observation stations.

See the page `https://www.metoffice.gov.uk/datapoint` for more information.

Features

Each data point consists of the following observations, or *Features*:

- Temperature (degrees Celsius)
- Wind direction (16 point compass)
- Wind speed (mph)
- Wind gust (mph)
- Dew Point (degrees Celsius)
- Screen Relative Humidity (%)
- Visibility (m)
- Pressure (hPa)
- Pressure Tendency (Pa/s)

For more information on the collection of this data see the following page `https://www.metoffice.gov.uk/datapoint/product/uk-hourly-site-specific-observations`.

Weather Types

The *Weather Type* is also collected at the time of the observation. This is an index of the 30 types of possible weather as described at `https://www.metoffice.gov.uk/datapoint/support/documentation/code-definitions`.

It may be very difficult (and may be impossible) to apply a machine learning classification to all 30 types. To simplify the problem for this assignment the data has been reorganised into two sets:

- Basic : each data point is labelled as one of 3 Weather Types - *Clear*, *Cloudy* and *Precipitation*
- Advanced (11 types - *Clear*, *Partly Cloudy*, *Mist*, *Fog*, *Cloudy*, *Overcast*, *Rain*, *Sleet*, *Hail*, *Snow*, and *Thunder*)

You should start by using the *Basic* stream. If you able to get decent classification accuracy then try the *Advanced* stream.

Task

You will try to predict the Weather Type for any given observation by using a machine learning classification method as covered in the lectures.

Up to approximately two-thirds of the marks can be obtained for an excellent report which describes a solution which includes the following steps:

- Create suitable training and testing data samples based upon a suitable selection of Features from the data
- Select the *Decision Tree* classification method
- Fit the *training* data
- Predict the Weather Type using the *testing* data
- Evaluate the prediction performance
- Visualise the classification process

Additional marks can be obtained by going further and optimising the classification, including but not limited to:

- Trying a different selection of features by modifying the selection made in `FeatureExtract.py`
- Using different classification methods and comparing the relative performance
- Optimising your chosen classifier(s)
- Add your own input features based on the available observation data
- Results using the *Advanced.txt* file

Please refer to the *Resources* section for suggestions on how to visualise the classification process and on how to proceed with classification optimisation.

Supplied Code and Data

Observation Data

All the observation data are available as text files `basic.txt` and `advanced.txt`, with one observation entry per line and the feature data separated by spaces. Each line commences with a numeric Station ID and name of each weather station. When loading the data you have the option to just read a selected number of lines from the top of the file for initial evaluation.

The Weather class

Routines for loading observation data and the manipulation of the dataset are provided for you through the `Weather` class. You are not expected to completely understand the class implementation. You are welcome to extend the class implementation if this helps with your chosen solution.

Feature extraction

The supplied python file `FeatureExtract.py` performs the following steps:

- Loads the weather data from file
- Displays information about the dataset
- Recovers any incomplete data with null observations
- Discards any data that cannot be recovered
- Converts the data into an appropriate format for classification
- Appends new features derived from the data
- Selects features to be exported for classification

Please review `FeatureExtract.py` to understand how the above steps are implemented. Examples of how to access a selection of the dataset are also included in the code. Feel free to copy the examples and adapt them for your purposes. You may wish to modify the contents of `FeatureExtract.py` and add your own code as part of your optimisation work.

Note: You will need to execute the main steps in `FeatureExtract.py` to produce the *pickle* format data file containing your selected *Features* so that they can be read for your classification task.

Machine Learning Template

You will be using the **sklearn** package to devise your machine learning classification method. The template file `MachineLearning.py` will guide you through all the steps required for the minimal solution outlined in the *Task* section. This template files only serves as a guide - you can start from an empty file if you wish.

Notebooks

IPython notebooks for the `FeatureExtract.py` and `MachineLearning.py` code have also been provided if you wish to use them. Using notebooks is not a requirement for the assessment and are only provided as an aid to iterate through the various steps.

Software Requirements

The relevant software packages should already be installed on the Lab PCs. You should be able to download the supplied code and run `FeatureExtract.py` and then `MachineLearning.py` without errors. Please report any problems you have with running the **supplied** code (i.e a freshly downloaded version without any modifications)

If you would like to use your laptop then ensure that the packages `numpy`, `scipy` and `sklearn` are installed. Please ensure you are running at least version of *0.18* of sklearn. See `http://scikit-learn.org/stable/install.html` for more details. As always it is your responsibility to maintain a working copy on the Lab PCs which you could submit in case of any laptop problems.

Report

You should submit a report describing your approach and chosen classifier(s).

The basic style of the report should be:

- An introduction describing the context and the problem
- Description of the data
- Description of the chosen classification methods and any further work
- Presentation of results, performance and any comparisons
- Summary and conclusions
- Appendix which must contain all code written by you. All code must be well documented in the code itself.
- Visualisation images may also be included in an appendix

The report, excluding appendices, should not exceed 8 pages. The report is to be written so that it would be understandable by a scientist who has no knowledge of the course. Therefore it must describe the context, problem, methods applied and results in a self-standing way, which does not rely upon any other knowledge of the problem. The report should be submitted on LEARN as a pdf file. Report marking criteria include:

- Structure of report
- Quality of communication to reader with no prior knowledge
- Quality and completeness of presentation of results including adequate explanation
- Depth to which the problem has been addressed and comparisons made
- Quality of conclusions
- Quality of code

This project and report must be your own work, and must contain a declaration to this effect.

Further Resources

Please see the supplementary Q&A document for more discussion on the task.

Visualisation

The matplotlib package and functions within sklearn can be used to visualise the classification process. Many examples with supplied code are shown on the sklearn examples page (http://scikit-learn.org/stable/auto_examples/index.html). Here are two examples for the Decision Tree classification method:

- Decision surface plots of a decision tree: http://edin.ac/2zKBKeE
- Export graphviz (for decision trees): http://edin.ac/2hvWfBq

Optimisation

Please refer to Section 3 of the sklearn user guide (http://scikit-learn.org/stable/model_selection.html) for ideas on how to improve your initial classification result. In particular, explore the use of *cross-validation* and *grid search* techniques to find the best result for a given estimator. An application of the grid search technique is demonstrated here: http://edin.ac/2zKB4pC.