HOW-TO: Classification Model Creation for ANGEL

Nigel Williams
Centre for Advanced Internet Architectures (CAIA). Technical Report 070131A
Swinburne University of Technology
Melbourne, Australia
niwilliams@swin.edu.au

***Abstract*— This paper briefly outline the steps required to produce a classification model for the Angel Flow Classifier. This paper assumes some prior knowledge of Machine Learning techniques.**

Keywords- **ANGEL, Traffic Classification, Machine Learning.**

I. INTRODUCTION

The Angel Flow Classifier employs Machine Learning (ML) techniques to classify traffic flows into classes (e.g. game, non-game). A pre-made classification model is used to perform class prediction. This document details the process of producing such a classification model for use with a Naïve Bayes classifier (the process remains essentially the same for other algorithm implementations, however). It is assumed that the reader is at least somewhat familiar with ML and network metering. For a more thorough introduction to ML techniques and algorithms in the context of network traffic classification refer to the Dynamic Self-Learning Traffic Classification (DSTC/netAI) [1] website at CAIA.

II. TOOLS REQUIRED

The process of creating a classification model for use in the Angel Flow Classifier involves three major steps: Statistics Generation, Dataset Preparation and Model Induction. Each of these stages currently requires the use of a separate tool. For statistics generation we use NetMate, for dataset preparation we use Weka and its file format ARFF, and for model induction with use BayesInducer, part of the ANGEL distribution.

A.*Statistics Generation*

When generating flow statistics, we prefer to use the NetMate accounting tool [2]. NetMate is an extensible metering tool with the ability to export common network flow statistics such as packet lengths and inter-arrival times. More exotic statistics such as Fourier transforms etc can be user-implemented. Note that the statistics used by ANGEL for classification are supported in the default NetMate distribution. If not using NetMate, you will need to ensure that your meter software is able to output the feature listed in the Appendix.

Example usage of the NetMate tool for generating flow statistics from a tcpdump file can be found in the netAI 'getting started' guide [3]. Information on the approach to flow statistics generation (flow timeouts, bi-directionality etc) can be found in the following papers [4], [5] and [6].

The Datasets provided with ANGEL were generated using the approach in [5] and [6].

B.*Dataset Preparation and Pre-processing*

The raw statistics created by NetMate need to be formatted before any pre-processing or model induction takes place. As we currently use the Weka [7] Machine Learning suite for pre-processing tasks we prepare our data in the default file format of Weka, called ARFF (Attribute Relation File Format) [8].

Converting the NetMate output into ARFF primarily involves the addition of ARFF headers to the statistics file (see the ARFF Reference for header guidelines). The algorithm implemented in the basic ANGEL distribution is a Naïve Bayes Classifier for Discretised data (that is feature values should be binned prior to model induction and before classification). We use the Supervised Discretisation filter within Weka to translate the numeric statistics into nominal intervals.

C.*Model Induction*

The BayesInducer tool is used to create a

From February 2007 and July 2010 this report was a confidential deliverable to the Smart Internet Technologies CRC. With permission it has now been released to the wider community.

classification model that can be loaded into the Angel Flow Classifier. It takes the pre-processed ARFF file and outputs a text file containing probability tables that can be used for Naïve Bayes classification.

III. Generating Statistics

The datasets supplied with ANGEL were generated using the sub-flow pair technique described in [5] and [6]. We prefer to use NetMate for statistics calculation, but any suitable meter software should fit the role. Note that some of the steps in the following section may differ depending on the meter software used.

IV. Dataset Preparation

This section describes how to prepare NetMate's output into the ARFF format for Weka.

Below is a set of comma-separated statistics as typically output from NetMate, Each line provides a statistical summary of a single flow.

```
27960,17,13,559,12,3660,43,43
27960,17,12,3660,13,559,305,305
993,6,13,1391,12,958,73,107
```

In this example NetMate has output eight columns of data, in the order: destination port, protocol, forward packets, forward bytes, back packets, back bytes, min forward packet length, mean forward packet length. Typically you would want thousands of flows output, rather than the three used in this example. Also, the Angel Flow Classifier by default uses a larger number of features, detailed in the Appendix.

Notice that the 'class' of the flow is not indicated on each line – adding a class column is the first step. Before assigning classes the actual class of each flow should have been determined. In the above example, we have already confirmed that flows with destination port `27960` are game flows, and those with any other destination port are not game flows. The prior classification of flows can be done in a number of ways, though the best is usually to have control over the original trace capture or to do some kind of manual packet inspection.

Once we know the class of each flow, simply add a class column to the end of each line. After adding the class column, the statistics will look like this:

```
27960,17,13,559,12,3660,43,43,game
27960,17,12,3660,13,559,305,305,game
993,6,13,1391,12,958,73,107,Non-game
```

It is critical that the class names 'game' and 'Non-game' are used, as these are the class names that the BayesInducer will look for. It is also necessary that the class column is the lasts column of the data. We then add an ARFF header to the data. For the example data, we would construct the header as follows:

```
@relation 'our example dataset'

@attribute dstport   numeric
@attribute protocol  numeric
@attribute fpackets  numeric
@attribute fbytes    numeric
@attribute bpackets  numeric
@attribute bbytes    numeric
@attribute minfpktl  numeric
@attribute meanfpktl numeric

@attribute class {game, Non-game}

@data
27960,17,13,559,12,3660,43,43,game
27960,17,12,3660,13,559,305,305,game
993,6,13,1391,12,958,73,107,Non-game
```

The order of the attributes listed should match the order of the attributes on each line. It is also important that the last 'attribute' (@attribute class) appears as it does in this example. The relation tag is optional, but it is usually helpful to include a short description of the dataset.

We first want to filter out any features that are not going to be used in classification. For this example we want to remove the destination port and protocol. This is done using the filter in the pre-process window of the Weka GUI.

The next step is to 'discretise' the numeric data into nominal values, as the Naïve Bayes classifier provided with the Angel Flow Classifier uses only nominal values. This step might not be required for other classification algorithm implementations that are able to use numeric values (C4.5 for instance). To discretise the data, load the ARFF file into Weka, and use the supervised discretisation filter, and save the filtered file.

After discretisation the output should look something like the following:

```
@attribute meanfpktl {'\'(-inf-14]\'',…
etc … ,'\'(1336.5-inf)\''}

@data
'\'(42.5-43.5]\'','\'(42.5-
43.5]\'','\'(40.5-43.5]\'', … etc …,
game
```

The main observation here is that the numeric values have been replaced with a nominal value indicating an interval. This process makes each line significantly longer, hence the truncated example. The dataset can now be run through the BayesInducer to produce the classification model.

V. Creating Model with BayesInducer

Using BayesInducer is relatively straightforward. Simply run the executable with the prepared file as an argument and pipe the output to a file.

```
$./BayesInducer example.arff > out_model_file.model
```

The resultant file can then be used in the Angel Flow Classifier as-is (by specifying the filename in the conf file).

VI. Conclusions

This how-to provides a brief overview on the process and tools used to create classification models for the Angel Flow Classifier. It is highly recommended that the reader familiarise themselves with the reference material before attempting to create a new classification model.

Acknowledgement

This work was supported from 2005 to early 2007 by the Smart Internet Technology Cooperative Research Centre, http://www.smartinternet.com.au.

References

[1] DSTC Project http://caia.swin.edu.au/urp/dstc/ (as of January 2007)
[2] NetMate, http://sourceforge.net/projects/netmate-meter/ (as of January 2007).
[3] netAI Homepage, http://caia.swin.edu.au/urp/dstc/netai/ (as of January 2007)
[4] SOME DSTC PAPER
[5] T.T.T. Nguyen, G. Armitage, **"Training on multiple sub-flows to optimise the use of Machine Learning classifiers in real-world IP networks,"** IEEE 31st Conference on Local Computer Networks, Tampa, Florida, USA, November 2006.
[6] T.T.T. Nguyen, G. Armitage, **"Synthetic Sub-flow Pairs for Timely and Stable IP Traffic Identification,"** Australian Telecommunication Networks and Application Conference 2006, Melbourne, Australia 4-6 December 2006
[7] Weka Homepage: http://www.cs.waikato.ac.nz/ml/weka/ (as of January 2007)
[8] Attribute-Relation File Format http://www.cs.waikato.ac.nz/~ml/weka/arff.html (as of January 2007)

Appendix

The following table lists the features used by the Angel Flow Classifier to perform classification. If using the default distribution of ANGEL, then these features must be included in the classification model.

Feature	**Description**
minfpktl	Minimum Forward Packet Length
meanfpktl	Mean Forward Packet Length
maxfpktl	Maximum Forward Packet Length
stdfpktl	Std Deviation of Forward Packet Lengths
minbpktl	Minimum Forward Packet Length
meanbpktl	Mean Backwards Packet Length
maxbpktl	Max Backwards Packet Length
stdbpktl	Std Deviation Backwards Packet Lengths
minfiat	Minimum Forward Inter-Arrival Time
meanfiat	Mean Forward Inter-Arrival Time
maxfiat	Maximum Forward Inter-Arrival Time
stdfiat	Std Deviation Forward Inter-Arrival Time
minbiat	Minimum Backward Inter-Arrival Time
meanbiat	Mean Backward Inter-Arrival Time
maxbiat	Maximum Backward Inter-Arrival Time
stdbiat	Std Deviation Backward Inter-Arrival