FastML

Machine learning made easy

- RSS

Search

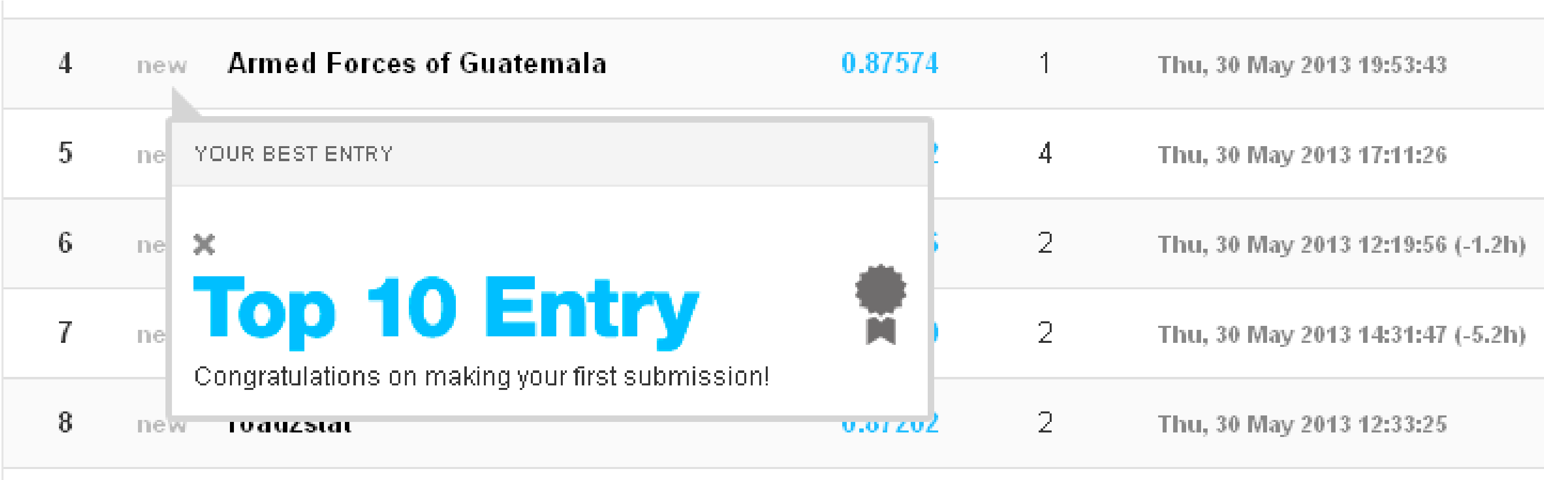
Navigate…

- Home
- Contents
- Links
- About

Amazon aspires to automate access control

2013-06-01 23:52

This is about Amazon access control challenge at Kaggle. Either we're getting smarter, or the competition is easy. Or maybe both. You can beat the benchmark quite easily and with AUC of 0.875 you'd be comfortably in the top twenty percent at the moment. We scored fourth in our first attempt - the model was quick to develop and back then there were fewer competitors.



Traditionally we use [Vowpal Wabbit](). Just simple binary classification with the logistic loss function and 10 passes over the data.

It seems to work pretty well even though the classes are very unbalanced: there's only a handful of negatives when compared to positives. Apparently Amazon employees usually get the access they request, even though sometimes they are refused.

Let's look at the data. First a label and then a bunch of IDs.

```
1,39353,85475,117961,118300,123472,117905,117906,290919,117908
1,17183,1540,117961,118343,123125,118536,118536,308574,118539
1,36724,14457,118219,118220,117884,117879,267952,19721,117880
```

We will count unique values in each column. That's R, by the way.

```
count_unique = function( column ) { length( unique( column )) }
apply( train, 2, count_unique )

          ACTION         RESOURCE           MGR_ID    ROLE_ROLLUP_1    ROLE_ROLLUP_2
               2             7518             4243              128              177
   ROLE_DEPTNAME       ROLE_TITLE ROLE_FAMILY_DESC      ROLE_FAMILY        ROLE_CODE
             449              343             2358               67              343
```

Altogether, it's approximately 17k binary features, with 33k examples in the training set. Categorical variables have way too many values for a random forest. Linear model seems like a good fit.

We'll skip converting features to zeros and ones for now; Vowpal Wabbit doesn't need this. We'll just create a namespace for each feature and let the Wabbit figure out the rest.

```
1 |e0 _39353 |e1 _85475 |e2 _117961 |e3 _118300 |e4 _123472 |e5 _117905 |e6 _117906 |e7 _290919
1 |e0 _17183 |e1 _1540 |e2 _117961 |e3 _118343 |e4 _123125 |e5 _118536 |e6 _118536 |e7 _308574
1 |e0 _36724 |e1 _14457 |e2 _118219 |e3 _118220 |e4 _117884 |e5 _117879 |e6 _267952 |e7 _19721
```

Namespaces are called e0…e7, for no particular reason. You may notice that originally there was one more column. That's because we got rid of ROLE_CODE, which is identical with ROLE_TITLE - you don't need both.

We prefix IDs with underscores so that VW knows that they are strings and need to be hashed. Actually, you can skip prefixing and just use numbers, and it produces similiar results.

We provide two Python scripts: one for converting from CSV to VW, and one for converting VW predictions to a submission format. There's also a script measuring AUC, for validation. That's all.

`csv2vw.py train.csv train.vw`
`csv2vw.py test.csv test.vw`

`vw -d train.vw -c -k -f model --loss_function logistic --passes 10` `vw -t -d test.vw -i model -p p.txt`

`vw2sub.py p.txt p_sub.txt`

Normally we'd run VW's predictions through a sigmoid function to get probabilities. But you can do without this step here, because AUC metric only cares about ranking of predictions. For the same reason you could use VW's quantile loss function instead of the logistic. More on this below.

VW loss functions

Vowpal Wabbit supports four loss functions: squared, logistic, hinge and quantile. Squared is for regression, logistic and hinge for classification, quantile for ranking.

This competition can be viewed either as a classification or a ranking task, so one might choose between logistic and quantile losses. The downside of the quantile function is that you need to tune an additional parameter, `--quantile_tau`. On the other hand, it converges in fewer passes.

`vw -d train.vw -c -k -f model --loss_function quantile --quantile_tau 0.15 --passes 3`

The results with logistic and quantile functions are similiar.