Now, let’s use the model to make predictions and  
see how and if we can improve the model. Then, let’s train the model on the whole data.

**Step 1: prepare the data**

The first step consists in importing the test data and preparing it. The test data need not be large  
and thus can be imported and worked on in R.

I need to remove the target column from the test set, or else it will be used to make predictions.  
If you do not remove this column the accuracy of the model will be very high, but it will be wrong  
since, of course, you do not have the target column at running time… because it is the column  
that you want to predict!

library("tidyverse")

library("yardstick")

small\_test <- read\_delim("data\_split/small\_test.txt", "|",

escape\_double = FALSE, col\_names = FALSE,

trim\_ws = TRUE)

small\_test %>%

mutate(X1= " ") %>%

write\_delim("data\_split/small\_test2.txt", col\_names = FALSE, delim = "|")

I wrote the data in a file called small\_test2.txt and can now use my model to make predictions:

system2("/home/cbrunos/miniconda3/bin/vw", args = "-t -i vw\_models/small\_oaa.model data\_split/small\_test2.txt -p data\_split/small\_oaa.predict")

The predictions get saved in the file small\_oaa.predict, which is a plain text file. Let’s add these  
predictions to the original test set:

small\_predictions <- read\_delim("data\_split/small\_oaa.predict", "|",

escape\_double = FALSE, col\_names = FALSE,

trim\_ws = TRUE)

small\_test <- small\_test %>%

rename(truth = X1) %>%

mutate(truth = factor(truth, levels = c("1", "2", "3", "4", "5")))

small\_predictions <- small\_predictions %>%

rename(predictions = X1) %>%

mutate(predictions = factor(predictions, levels = c("1", "2", "3", "4", "5")))

small\_test <- small\_test %>%

bind\_cols(small\_predictions)

**Step 2: use the model and test data to evaluate performance**

We can use the several metrics included in {yardstick} to evaluate the model’s performance:

conf\_mat(small\_test, truth = truth, estimate = predictions)

accuracy(small\_test, truth = truth, estimate = predictions)

Truth

Prediction 1 2 3 4 5

1 51 15 2 10 1

2 11 6 3 1 0

3 0 0 0 0 0

4 0 0 0 0 0

5 0 0 0 0 0

# A tibble: 1 x 3

.metric .estimator .estimate

1 accuracy multiclass 0.570

We can see that the model never predicted class 3, 4 or 5. Can we improve by adding some  
regularization? Let’s find out!

**Step 3: adding regularization**

Before trying regularization, let’s try changing the cost function from the logistic function to the  
hinge function:

# Train the model

hinge\_oaa\_fit <- system2("/home/cbrunos/miniconda3/bin/vw", args = "--oaa 5 -d data\_split/small\_train.txt --loss\_function hinge -f vw\_models/hinge\_oaa.model", stderr = TRUE)

system2("/home/cbrunos/miniconda3/bin/vw", args = "-i vw\_models/hinge\_oaa.model -t -d data\_split/small\_test2.txt -p data\_split/hinge\_oaa.predict")

predictions <- read\_delim("data\_split/hinge\_oaa.predict", "|",

escape\_double = FALSE, col\_names = FALSE,

trim\_ws = TRUE)

test <- test %>%

select(-predictions)

predictions <- predictions %>%

rename(predictions = X1) %>%

mutate(predictions = factor(predictions, levels = c("1", "2", "3", "4", "5")))

test <- test %>%

bind\_cols(predictions)

conf\_mat(test, truth = truth, estimate = predictions)

accuracy(test, truth = truth, estimate = predictions)

Truth

Prediction 1 2 3 4 5

1 411 120 45 92 1

2 355 189 12 17 0

3 11 2 0 0 0

4 36 4 0 1 0

5 3 0 3 0 0

# A tibble: 1 x 3

.metric .estimator .estimate

1 accuracy multiclass 0.462

Well, didn’t work out so well, but at least we now know how to change the loss function. Let’s go  
back to the logistic loss and add some regularization. First, let’s train the model:

regul\_oaa\_fit <- system2("/home/cbrunos/miniconda3/bin/vw", args = "--oaa 5 --l1 0.005 --l2 0.005 -d data\_split/small\_train.txt -f vw\_models/small\_regul\_oaa.model", stderr = TRUE)

Now we can use it for prediction:

system2("/home/cbrunos/miniconda3/bin/vw", args = "-i vw\_models/small\_regul\_oaa.model -t -d data\_split/test2.txt -p data\_split/small\_regul\_oaa.predict")

predictions <- read\_delim("data\_split/small\_regul\_oaa.predict", "|",

escape\_double = FALSE, col\_names = FALSE,

trim\_ws = TRUE)

test <- test %>%

select(-predictions)

predictions <- predictions %>%

rename(predictions = X1) %>%

mutate(predictions = factor(predictions, levels = c("1", "2", "3", "4", "5")))

test <- test %>%

bind\_cols(predictions)

We can now use it for predictions:

conf\_mat(test, truth = truth, estimate = predictions)

accuracy(test, truth = truth, estimate = predictions)

Truth

Prediction 1 2 3 4 5

1 816 315 60 110 1

2 0 0 0 0 0

3 0 0 0 0 0

4 0 0 0 0 0

5 0 0 0 0 0

# A tibble: 1 x 3

.metric .estimator .estimate

1 accuracy multiclass 0.627

So accuracy improved, but the model only predicts class 1 now… let’s try with other hyper-parameters values:

regul\_oaa\_fit <- system2("/home/cbrunos/miniconda3/bin/vw", args = "--oaa 5 --l1 0.00015 --l2 0.00015 -d data\_split/small\_train.txt -f vw\_models/small\_regul\_oaa.model", stderr = TRUE)

conf\_mat(test, truth = truth, estimate = predictions)

accuracy(test, truth = truth, estimate = predictions)

Truth

Prediction 1 2 3 4 5

1 784 300 57 108 1

2 32 14 3 2 0

3 0 1 0 0 0

4 0 0 0 0 0

5 0 0 0 0 0

# A tibble: 1 x 3

.metric .estimator .estimate

1 accuracy multiclass 0.613

So accuracy is lower than previously, but at least more categories get correctly predicted. Depending  
on your needs, you should consider different metrics. Especially for classification problems, you might  
not be interested in accuracy, in particular if the data is severely unbalanced.

Anyhow, to finish this blog post, let’s train the model on the whole data and measure the time it  
takes to run the full model.

**Step 4: Training on the whole data**

Let’s first split the whole data into a training and a testing set:

nb\_lines <- system2("cat", args = "text\_fr.txt | wc -l", stdout = TRUE)

system2("split", args = paste0("-l", floor(as.numeric(nb\_lines)\*0.995), " text\_fr.txt data\_split/"))

system2("mv", args = "data\_split/aa data\_split/train.txt")

system2("mv", args = "data\_split/ab data\_split/test.txt")

The whole data contains 260247 lines, and the training set weighs 667MB, which is quite large. Let’s train  
the simple multiple classifier on the data and see how long it takes:

tic <- Sys.time()

oaa\_fit <- system2("/home/cbrunos/miniconda3/bin/vw", args = "--oaa 5 -d data\_split/train.txt -f vw\_models/oaa.model", stderr = TRUE)

Sys.time() - tic

Time difference of 4.73266 secs

Yep, you read that right. Training the classifier on 667MB of data took less than 5 seconds!

Let’s take a look at the final object:

oaa\_fit

[1] "final\_regressor = vw\_models/oaa.model"

[2] "Num weight bits = 18"

[3] "learning rate = 0.5"

[4] "initial\_t = 0"

[5] "power\_t = 0.5"

[6] "using no cache"

[7] "Reading datafile = data\_split/train.txt"

[8] "num sources = 1"

[9] "average since example example current current current"

[10] "loss last counter weight label predict features"

[11] "1.000000 1.000000 1 1.0 2 1 253"

[12] "0.500000 0.000000 2 2.0 2 2 499"

[13] "0.250000 0.000000 4 4.0 2 2 6"

[14] "0.250000 0.250000 8 8.0 1 1 2268"

[15] "0.312500 0.375000 16 16.0 1 1 237"

[16] "0.250000 0.187500 32 32.0 1 1 557"

[17] "0.171875 0.093750 64 64.0 1 1 689"

[18] "0.179688 0.187500 128 128.0 2 2 208"

[19] "0.144531 0.109375 256 256.0 1 1 856"

[20] "0.136719 0.128906 512 512.0 4 4 4"

[21] "0.122070 0.107422 1024 1024.0 1 1 1353"

[22] "0.106934 0.091797 2048 2048.0 1 1 571"

[23] "0.098633 0.090332 4096 4096.0 1 1 43"

[24] "0.080566 0.062500 8192 8192.0 1 1 885"

[25] "0.069336 0.058105 16384 16384.0 1 1 810"

[26] "0.062683 0.056030 32768 32768.0 2 2 467"

[27] "0.058167 0.053650 65536 65536.0 1 1 47"

[28] "0.056061 0.053955 131072 131072.0 1 1 495"

[29] ""

[30] "finished run"

[31] "number of examples = 258945"

[32] "weighted example sum = 258945.000000"

[33] "weighted label sum = 0.000000"

[34] "average loss = 0.054467"

[35] "total feature number = 116335486"

Let’s use the test set and see how the model fares:

conf\_mat(test, truth = truth, estimate = predictions)

accuracy(test, truth = truth, estimate = predictions)

Truth

Prediction 1 2 3 4 5

1 537 175 52 100 1

2 271 140 8 9 0

3 1 0 0 0 0

4 7 0 0 1 0

5 0 0 0 0 0

# A tibble: 1 x 3

.metric .estimator .estimate

1 accuracy multiclass 0.521

Better accuracy can certainly be achieved with hyper-parameter tuning… maybe the subject for a  
future blog post? In any case I am very impressed with Vowpal Wabbit and am certainly looking forward  
to future developments of {RVowpalWabbit}!