

# baseline\_\_main

Yufei Zhao

2017/10/29

```
print(R.version)

##
## platform      x86_64-w64-mingw32
## arch          x86_64
## os            mingw32
## system        x86_64, mingw32
## status
## major         3
## minor         4.1
## year          2017
## month         06
## day           30
## svn rev       72865
## language      R
## version.string R version 3.4.1 (2017-06-30)
## nickname      Single Candle

if(!require("gbm")){
  install.packages("gbm")
}

library(gbm)
```

## Step 1: set up controls for evaluation experiments.

In this chunk, we have a set of controls for the evaluation experiments.

- (T/F) cross-validation on the training set
- (number) K, the number of CV folds
- (T/F) run evaluation on an independent test set

```
run.cv=TRUE # run cross-validation on the training set
K <- 3 # number of CV folds
run.test=TRUE # run evaluation on an independent test set
```

## Step 2: perform model selection by 5 folds cross-validation

Using cross-validation or independent test set evaluation, we compare the performance of different classifiers or classifiers with different specifications. In this example, we use GBM with different **depth**. In the following chunk, we list, in a vector, setups (in this case, **depth**) corresponding to models that we will compare. In your project, you maybe comparing very different classifiers. You can assign them numerical IDs and labels specific to your project.

```
model_values <- seq(3, 11, 4)
model_labels = paste("GBM with depth =", model_values)
```

## Read in data

```
# lables(0 for muffin, 1 for chicken, 2 for dog)
labels <- read.csv("../data/label_train.csv",header=TRUE)
colnames(labels)[2] <- "labels"
sift_data <- read.csv("../data/sift_train.csv",header=TRUE, stringsAsFactors = FALSE)

# prepare data
label_train <- labels[,-1]
dat_train <- sift_data[,-1]
```

## load train and test method

```
source("../lib/baseline_train.R")
source("../lib/baseline_test.R")
```

## Model selection with cross-validation

- Do model selection by choosing among different values of training model parameters, that is, the interaction depth for GBM in this example.

```
source("../lib/baseline_cv.R")

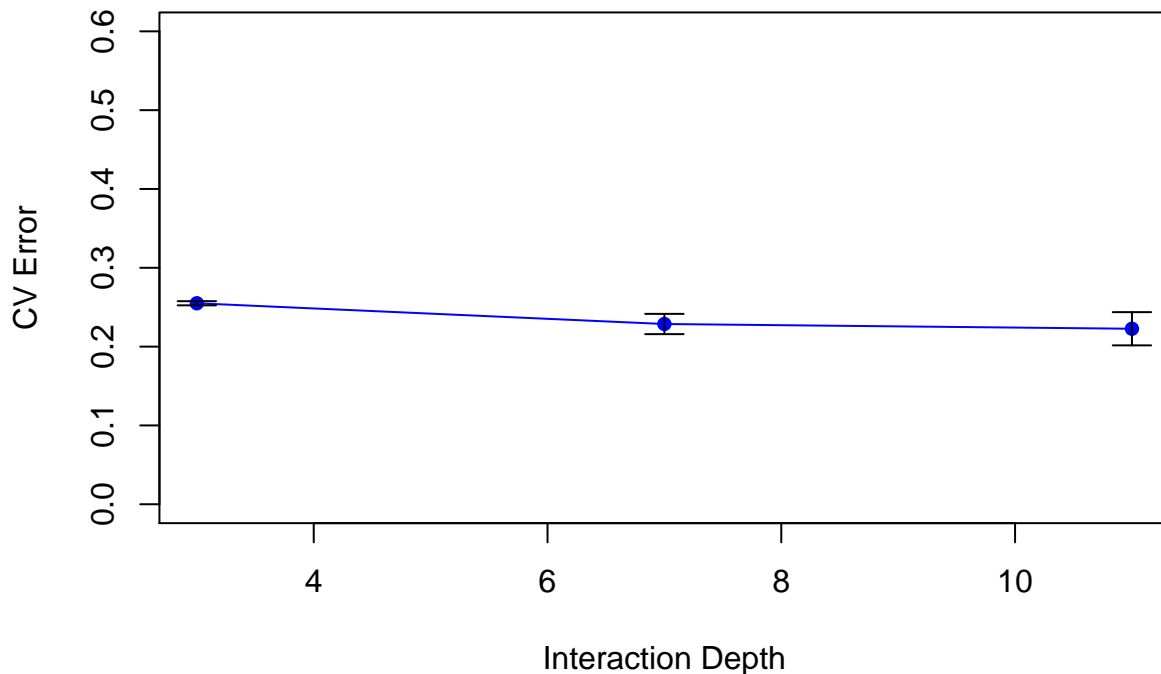
if(run.cv){
  err_cv <- array(dim=c(length(model_values), 2))
  for(k in 1:length(model_values)){
    cat("k=", k, "\n")
    err_cv[k,] <- cv.function(dat_train, label_train, model_values[k], K)
  }
  save(err_cv, file="../output/baseline_err_cv.RData")
}
```

```
## k= 1
## k= 2
## k= 3
```

## Step 3: Visualize cross-validation results.

```
if(run.cv){
  load("../output/baseline_err_cv.RData")
  #pdf("../fig/cv_results.pdf", width=7, height=5)
  plot(model_values, err_cv[,1], xlab="Interaction Depth", ylab="CV Error",
       main="Cross Validation Error", type="n", ylim=c(0, 0.6))
  points(model_values, err_cv[,1], col="blue", pch=16)
  lines(model_values, err_cv[,1], col="blue")
  arrows(model_values, err_cv[,1]-err_cv[,2], model_values, err_cv[,1]+err_cv[,2],
        length=0.1, angle=90, code=3)
  #dev.off()
}
```

## Cross Validation Error



- Choose the “best” parameter value

```
model_best=model_values[1]
if(run.cv){
  model_best <- model_values[which.min(err_cv[,1])]
}
```

```
par_best <- list(depth=model_best)
cat(model_best)
```

```
## 11
```

- Train the model with the entire training set using the selected model (model parameter) via cross-validation.

```
tm_train=NA
tm_train <- system.time(fit_train <- train(dat_train, label_train, par_best))
```

```
## Warning in gbm.perf(fit_gbm, method = "OOB", plot.it = FALSE): OOB
## generally underestimates the optimal number of iterations although
## predictive performance is reasonably competitive. Using cv.folds>0 when
## calling gbm usually results in improved predictive performance.
```

```
save(fit_train, file="./output/baseline_fit_train.RData")
```

### Step 4: Make prediction

Feed the final training model with the completely holdout testing data.

```
{r} #dat_test <- read.csv("../data/sift_test.csv", header=TRUE)
#dat_test <- dat_test[,-1] #

{r test} #tm_test=NA #if(run.test){ # load(file="../output/baseline_fit
# tm_test <- system.time(pred_test <- test(fit_train, dat_test))
# save(pred_test, file="../output/baseline_pred_test.RData")
#} #
```

### Summarize Running Time

Prediction performance matters, so does the running times for constructing features and for training the model, especially when the computation resource is limited.

```
cat("Time for training model=", tm_train[1], "s \n")
```

```
## Time for training model= 3944.14 s
```

```
#cat("Time for making prediction=", tm_test[1], "s \n")
```