Machine Learning BITS F464

Assignment #2 Report

BITS Pilani

November 30, 2015

Problem Statement: #2 Active Learning in a Spam Filter

Dataset: SpamBase

Submitted By: Abhinav Agarwal, 2013A7PS124P

1 Introduction

Active Learning is a machine learning technique that is used widely to create classification systems in absence of large numbers of labeled examples. It involves iteratively training a classifier form an unlabelled training set using an information source i.e. an oracle (usually a human expert) to label those examples which are most informative to the training process.
Tools used: R language, RStudio, R Packages (caret, MASS, kernlab)

2 Data

2.1 Source & Preprocessing

The dataset used is the publically available SpamBase dataset which consists of 4601 instances of 57 attributes. 48 of the attributes are frequency percentage of a particular word like 'sell', 'buy' and so on. The last attribute is the nominal class attribute of type spam which denotes whether the email was considered spam(1) or not(0). This has been modified to contain instead words 'spam' to denote being spam and 'ham' to denote being non-spam for sake of easier processing.
The classification model is a SVM with a Radial Basis Function Kernel and two preprocessing steps are done to use the same.
method = 'center' subtracts the mean from the predictor values
method = 'scale' divides by standard deviation

2.2 Justification

A spam filter is a tool used by email hosts to filter unsolicited emails including mass junk email, offers and newsgroup subscriptions from the user's inbox. Conventionally the email host decides for the user what kinds of emails he assumes should be spam i.e. the data is labeled by the host. In an active learning spam filter the user can customize the filter to own needs. The 'oracle' is now not the host but the user himself. We have a set of unlabelled data which is the set of emails that the user actually gotten. We query the user dynamically and train the classifier using this labeled data. So we get a spam filter

personalized to the user. I have tried to simulate the same by considering a set of emails which we have labels for. We first remove the labels from a subset of this set and then use active learning algorithms to decide which instances to query the user for i.e. extract from the superset labeled data.

3 Algorithms

svmRadial: The classifier used was a Support Vector Machine with a Radial Basis Function which is used so as to make non linearly separable features linearly separable after mapping them to a high dimensional feature space. The kernel has the formula:

$$\begin{aligned} K\left(x^{(i)}, x^{(j)}\right) &= \phi(x^{(i)})^T \phi(x^{(j)}) \\ &= \exp\left(-\gamma \left\| x^{(i)} - x^{(j)} \right\|^2\right), \qquad \gamma > 0 \end{aligned}$$

Active learning: It is a machine learning paradigm for optimally choosing unlabeled observations in a training data set to query for their true labels. The framework is particularly useful when there are very few labelled observations relative to a large number of unlabeled observations, and the user seeks to determine as few true labels as possible to achieve highly accurate classifiers. We use a pool learning strategy where a pool of labelled data is queried at once instead of a stream based strategy where one instance is queried at a time. There are many approaches to deciding the querying strategy. We use a query by committee which utilizes a committee of C classifiers that are each trained on the labelled training data.

Query by bagging: The 'query by bagging' approach to active learning applies bootstrap aggregating (bagging) by randomly sampling with replacement C times from the training data to create a committee of C classifiers. We "query the oracle" with the observations that have the maximum disagreement among the C trained classifiers. To determine maximum disagreement among bagged committee members, we use the vote entropy approach wherein we seek to query those observations which maximize the vote entropy among all committee members.

To calculate this committee disagreement, we use the formula from Dr. Burr Settles' "Active Learning Literature Survey". Unlabeled observations in y are assumed to have NA for a label.

3.1 Code

3.1.1 Passive Learner

```
#Load the two files into R:
dataset <- read.csv("spamdata.csv",header=FALSE,sep=";")
names <- read.csv("spamnames.csv",header=FALSE,sep=";")

#Set the names of the dataset dataframe:
names(dataset) <- sapply((1:nrow(names)),function(i) toString(names[i,1]))

#make column y a factor variable for binary classification (spam or non-spam)
dataset$y <- as.factor(dataset$y)

#get a sample of 1000 rows
sample <- dataset[sample(nrow(dataset), 1000),]

require(caret)
require(kernlab)

#Split the data in dataTrain and dataTest
trainIndex <- createDataPartition(sample$y, p = .8, list = FALSE, times = 1)
dataTrain <- sample[ trainIndex,]
dataTest  <- sample[-trainIndex,]
levels(dataTrain$y) <- list(ham="0", spam="1")
levels(dataTest$y) <- list(ham="0", spam="1")
#Create the SVM model:

### finding optimal value of a tuning parameter
sigDist <- sigest(y ~ ., data = dataTrain, frac = 1)
### creating a grid of two tuning parameters, .sigma comes from the earlier line. we are trying to find
best value of .C
svmTuneGrid <- data.frame(.sigma = sigDist[1], .C = 2^(-2:7))

x <- train(y ~ .,
           data = dataTrain,
           method = "svmRadial",
           preProc = c("center", "scale"),
           trControl = trainControl(method = "repeatedcv", repeats = 5,
                                    classProbs =  TRUE))

#Evaluate the model
pred <- predict(x,dataTest[,1:57])

acc <- confusionMatrix(pred,dataTest$y)
```

3.1.2 Active Learner

```
require(itertools2)
require(caret)
require(entropy)
require(kernlab)

#Computes the disagreement measure for each of the unlabeled observations based on the either the
predicted class label
vote_entropy <- function(x, type='class', entropy_method='ML') {
  it <- do.call(itertools2::izip, x)
  disagreement <- sapply(it, function(obs) {
    entropy(table(unlist(obs)), method=entropy_method)
  })
  disagreement
}

# Returns a vector of indices of unlabeled observations.
which_unlabeled <- function(y) {
  which(is.na(y))
}

# Returns a vector of indices of labeled observations.
which_labeled <- function(y, return_logical = FALSE) {
  which(!is.na(y))
}

# Splits a matrix and its class labels into labeled and unlabeled pairs.
split_labeled <- function(x, y) {
  x <- as.matrix(x)
  y <- factor(y)

  unlabeled_i <- which_unlabeled(y)
  list(x_labeled=x[-unlabeled_i, ],
       y_labeled=y[-unlabeled_i],
       x_unlabeled=x[unlabeled_i, ],
       y_unlabeled=y[unlabeled_i])
}

# Automatic query of an oracle.
query_oracle <- function(i, y_truth) {
  as.vector(y_truth[i])
}

# The 'query by bagging' approach to active learning applies bootstrap aggregating (bagging) by randomly
sampling with replacement
query_bagging <- function(x, y, fit_f, predict_f,
                          disagreement="vote_entropy", "post_entropy",
                          num_query=1, C=50, ...) {

  disagreement <- match.arg(disagreement)
```

```
  x <- as.matrix(x)
  y <- factor(y)
  p <- ncol(x)
  split_out <- split_labeled(x, y)

  bag_control <- bagControl(
      fit=fit_f,
      predict=predict_f,
      aggregate=disagree_f,
      oob=FALSE,
      allowParallel=TRUE
  )

  bag_out <- try(
    bag(x=split_out$x_labeled,
        y=split_out$y_labeled,
        B=C, vars=p, bagControl=bag_control, ...),
    silent=TRUE
  )

  if (inherits(bag_out, "try-error")) {
    stop("The following error occurred while training the bagged classifiers:\n",
         bag_out)
  }

  disagreement <- predict(bag_out, split_out$x_unlabeled)

  # Determines the order of the unlabeled observations by disagreement measure.
  query <- head(order(disagreement, decreasing=TRUE), n=num_query)

  list(query=query, disagreement=disagreement)
}

#Load the two files into R:
dataset <- read.csv("spamdata.csv",header=FALSE,sep=";")
names <- read.csv("spamnames.csv",header=FALSE,sep=";")

#Set the names of the dataset dataframe:
names(dataset) <- sapply((1:nrow(names)),function(i) toString(names[i,1]))

#make column y a factor variable for binary classification (spam or non-spam)
dataset$y <- as.factor(dataset$y)

#get a sample of 1000 rows
sample <- dataset[sample(nrow(dataset), 1000),]

require(caret)
require(kernlab)

#Split the data in dataTrain and dataTest
trainIndex <- createDataPartition(sample$y, p = .8, list = FALSE, times = 1)
dataTrain <- sample[ trainIndex,]
dataTest  <- sample[-trainIndex,]
levels(dataTrain$y) <- list(ham="0", spam="1")
```

```
levels(dataTest$y) <- list(ham="0", spam="1")
#Create the SVM model:

### finding optimal value of a tuning parameter
sigDist <- sigest(y ~ ., data = dataTrain, frac = 1)
### creating a grid of two tuning parameters, .sigma comes from the earlier line. we are trying to find
best value of .C
svmTuneGrid <- data.frame(.sigma = sigDist[1], .C = 2^(-2:7))

fit_f <- function(x, y, ...) {
  caret::train(x, y, method = "svmRadial",
               preProc = c("center", "scale"),
               trControl = trainControl(method = "repeatedcv", repeats = 5,
                                        classProbs =  TRUE))
}

predict_f <- function(object, x) {
  getElement(predict(object, x), "y")
}

toquery <- query_bagging(x=x, y=y, fit_f=fit_f, predict_f=predict_f, C=10,
             disagreement="vote_entropy", num_query=100)
queried <- query_oracle(toquery, y)

auxind <- match(queried$y, y$y)
yy <- (rbind(y[,], queried)[-auxind,])
yy <- yy[order(yy$y),]
y[,] <- yy

x <- train(x, y, method = "svmRadial",
               preProc = c("center", "scale"),
               trControl = trainControl(method = "repeatedcv", repeats = 5,
                                        classProbs =  TRUE))

#Evaluate the model
pred <- predict(x,dataTest[,1:57])

acc <- confusionMatrix(pred,dataTest$y)
```

4 Results

The accuracy of the passive learning svmRadial classifier was 0.8794.

```
Confusion Matrix and Statistics

          Reference
Prediction ham spam
      ham  113   16
      spam   8   62

               Accuracy : 0.8794
                 95% CI : (0.8259, 0.9212)
    No Information Rate : 0.608
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7423
 Mcnemar's Test P-Value : 0.153

            Sensitivity : 0.9339
            Specificity : 0.7949
         Pos Pred Value : 0.8760
         Neg Pred Value : 0.8857
             Prevalence : 0.6080
         Detection Rate : 0.5678
   Detection Prevalence : 0.6482
      Balanced Accuracy : 0.8644

       'Positive' Class : ham
```

The accuracy of the active learning classifier with query by bagging with 10 instances of the svmRadial classifier in the committee was 0.9397 which is better than the passive learning classifier.

```
Confusion Matrix and Statistics

          Reference
Prediction ham spam
      ham  111    4
      spam   8   76

               Accuracy : 0.9397
                 95% CI : (0.897, 0.9685)
    No Information Rate : 0.598
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8756
 Mcnemar's Test P-Value : 0.3865

            Sensitivity : 0.9328
            Specificity : 0.9500
         Pos Pred Value : 0.9652
         Neg Pred Value : 0.9048
             Prevalence : 0.5980
         Detection Rate : 0.5578
```

```
Detection Prevalence : 0.5779
   Balanced Accuracy : 0.9414

      'Positive' Class : ham
```

5 Further Scope

A comparison of query by bagging and boosting can be done with different classifiers like LDA etc. The disagreement technique can be changed to Kullback divergence and post entropy.

6 References

[1] Active Learning Literature Survey, Burr Settles
http://burrsettles.com/pub/settles.activelearning.pdf

[2] SVM Based Spam Filter with Active and Online Learning, Qiang Wang et. al.
http://trec.nist.gov/pubs/trec15/papers/hit.spam.final.final.pdf

[3] Active Learning with Boosting for Spam Detection, Nikhila Arkalgud
https://classes.soe.ucsc.edu/cmps242/Winter08/proj/nikhila_report.pdf

[4] Online Active Learning Methods for Fast Label-Efficient Spam Filtering, D. Sculley
http://axon.cs.byu.edu/Dan/778/papers/Active%20Learning/sculley2*.pdf

[5] https://cran.r-project.org/web/packages/MASS/MASS.pdf

[6] https://cran.r-project.org/web/packages/caret/caret.pdf

[7] https://cran.r-project.org/web/packages/kernlab/kernlab.pdf