Support Vector Machine

Zhoutao Pei

This program fits an SVM model through stochastic gradient descent, using **hinge loss with L2 regularization** as loss function.

The data used in this program come from UC Irvine machine learning data repository https://archive.ics.uci.edu/ml/datasets/Adult. Only numeric attributes are used.

Data preprocessing

```
train <- read.table(params$trainFile, sep=',', header=FALSE, stringsAsFactors=F)
test <- read.table(params$testFile, sep=',', header=FALSE, stringsAsFactors=F, skip=1)
train <- train[,c(1,3,5,11,12,13,15)]
test <- test[,c(1,3,5,11,12,13,15)]
train$label <- -1
test$label <- -1
train$label[train$V15 == " >50K"] = 1
test$label[test$V15 == " >50K."] = 1
train$V15 <- NULL
test$V15 <- NULL
colnames(train)[1:6] <- c("Age","Fnlwgt","EduNum","CapGain","CapLoss","Hours")
colnames(test)[1:6] <- c("Age","Fnlwgt","EduNum","CapGain","CapLoss","Hours")
trainX <- train[, 1:6]
trainy <- train[, 7]
testX <- test[, 1:6]
testy <- test[, 7]
```

Function to train an SVM model by SGD

```
SVM <- function(trainX, trainy, epoch=500, batch_size=100, lambda=1e-5, lr=0.1) {
  # transform trainX so that each column ranges from 0 to 1
  colmin <- apply(trainX, 2, min)
  colrange <- apply(trainX, 2, function(x) range(x)[2]-range(x)[1])
  trainX$intercept <- 1 # add a column for intercept
  colmin <- c(colmin, 0)
  colrange <- c(colrange, 1)
  trainX_mat <- t((t(trainX) - colmin)/colrange)
  accuracy <- numeric(0)
  n_batch <- floor(nrow(train)/batch_size)
  w <- numeric(ncol(trainX))
  for (i in 1:epoch) {
    lr_epoch <- lr/(i*0.05+1)
    idx_shuffle <- sample(1:nrow(trainX), nrow(trainX))
    for (j in 1:n_batch) {
      idx_batch <- idx_shuffle[((j-1)*batch_size+1):(j*batch_size)]
      X_batch <- trainX_mat[idx_batch,]
      y_batch <- trainy[idx_batch]
```

```
      # calculation of gradient depends on the sign of (1-y(wx+b))
      # if (1-y(wx+b)) > 0, gradient(w) = -yx + lambda*w, gradient(b) = -y
      # if (1-y(wx+b)) <= 0, gradient(w) = lambda*w, gradient(b) = 0
      notCorrect <- (1-as.numeric(X_batch %*% w)*y_batch) > 0
      gradient <- -as.numeric(t(X_batch) %*% (y_batch*as.numeric(notCorrect))) /
                  batch_size + lambda*w
      w <- w - lr_epoch*gradient
    }
    pred <- as.numeric(trainX_mat %*% w)
    accuracy_batch <- sum(pred*trainy > 0)/length(trainy)
    accuracy <- c(accuracy, accuracy_batch)
  }
  return(list(w=w, colmin=colmin, colrange=colrange, train_accuracy=accuracy))
}
```

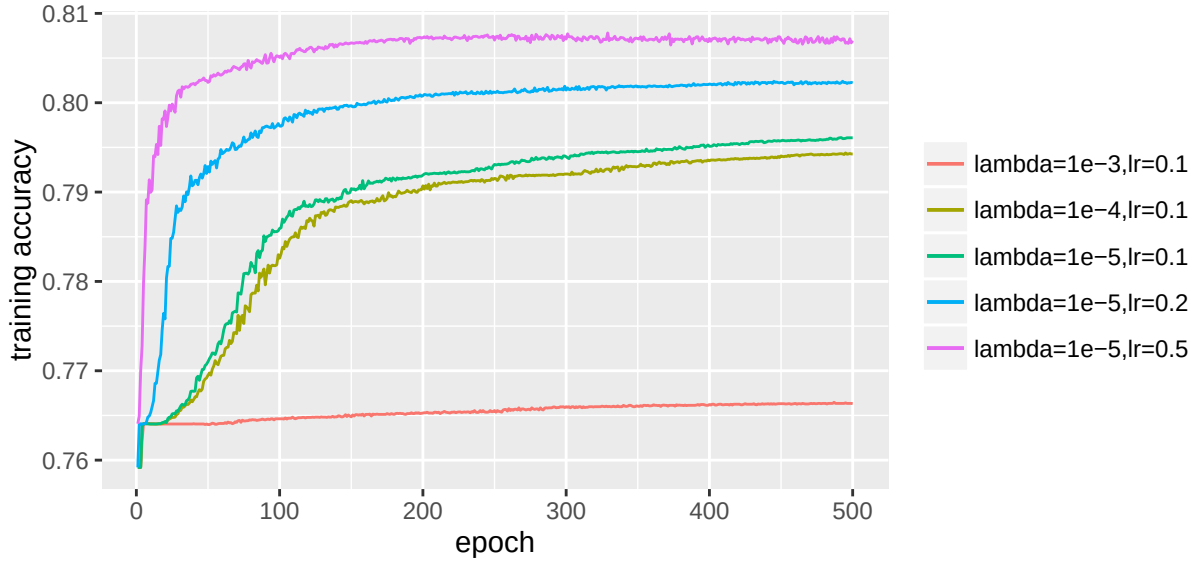
function to make prediction and calculate model accuracy

```
predict_SVM <- function(model, testX) {
  testX$intercept <- 1
  testX_mat <- t((t(testX) - model$colmin)/model$colrange)
  pred <- as.numeric(testX_mat %*% model$w)
  pred[pred > 0] = 1
  pred[pred <= 0] = -1
  return(pred)
}

get_accuracy <- function(pred, testy) {
  return (sum(pred == testy)/length(testy))
}
```

test model with different parameters

```
model1 <- SVM(trainX, trainy, lambda=1e-3)
model2 <- SVM(trainX, trainy, lambda=1e-4)
model3 <- SVM(trainX, trainy, lambda=1e-5)
model4 <- SVM(trainX, trainy, lambda=1e-5, lr=0.2)
model5 <- SVM(trainX, trainy, lambda=1e-5, lr=0.5)
library(ggplot2)
train_acc <- data.frame(model1 = model1$train_accuracy,
                        model2 = model2$train_accuracy,
                        model3 = model3$train_accuracy,
                        model4 = model4$train_accuracy,
                        model5 = model5$train_accuracy)

ggplot(train_acc) +
  geom_line(aes(x=1:500, y=model1, color="lambda=1e-3,lr=0.1"))+
  geom_line(aes(x=1:500, y=model2, color="lambda=1e-4,lr=0.1"))+
  geom_line(aes(x=1:500, y=model3, color="lambda=1e-5,lr=0.1"))+
  geom_line(aes(x=1:500, y=model4, color="lambda=1e-5,lr=0.2"))+
  geom_line(aes(x=1:500, y=model5, color="lambda=1e-5,lr=0.5"))+
```

```
  labs(x = "epoch", y = "training accuracy")+
  theme(legend.title=element_blank())
```




```
pred1 <- predict_SVM(model1, testX)
pred2 <- predict_SVM(model2, testX)
pred3 <- predict_SVM(model3, testX)
pred4 <- predict_SVM(model4, testX)
pred5 <- predict_SVM(model5, testX)
test_acc1 <- get_accuracy(pred1, testy)
test_acc2 <- get_accuracy(pred2, testy)
test_acc3 <- get_accuracy(pred3, testy)
test_acc4 <- get_accuracy(pred4, testy)
test_acc5 <- get_accuracy(pred5, testy)
```

```
##  Model1, lambda=1e-3, lr=0.1, test accuracy = 0.771144278606965
##  Model2, lambda=1e-4, lr=0.1, test accuracy = 0.79522142374547
##  Model3, lambda=1e-5, lr=0.1, test accuracy = 0.797555432712978
##  Model4, lambda=1e-5, lr=0.2, test accuracy = 0.802591978379706
##  Model5, lambda=1e-5, lr=0.5, test accuracy = 0.805233093790308
```