Perceptron

Rui Wang

10/19/2018

Perceptron(Rosenblatt 1958)

Basic Theory

1. Seperation happens
2. hyperplane is represented by a vector that is perpendicular to it $\hat{y} = sign(f(x)) = sign(\vec{\omega} \cdot \vec{x})$; $\hat{y}$ is prediction, f is a hyperplane
3. Margin (how far away we are from decision boundary in the correct direction) $= y \cdot f(x) = y \cdot (\vec{\omega} \cdot \vec{x} + b)$

Algorithm

- Input: $\{\vec{x}_i, y_i\}_{i=1}^n$ Initialize: $\vec{\omega}^{(1)} = (0, ..., 0)$ for $t = 1, 2, ...,$ locate i s.t. $y_i(\vec{\omega}^{(t)} \cdot \vec{x}_i) \leq 0$ if none, stop and output $\vec{\omega}^{(t)}$ else $\vec{\omega}^{(t+1)} = \vec{\omega}^{(t)} + y_i\vec{x}_i$ end
- Intuition: if point is incorrectly classified, move decision boundary toward it

```
#Perceptron function
perceptron <- function(data = data, I = I){
  x = data[,-dim(data)[2]]
  y = data[,dim(data)[2]]
  w = rep(0,dim(x)[2])
  W = matrix(0,ncol = dim(x)[2],nrow = I)
  for (e in 1:I){
    k=0
    for (i in 1:length(y)){
      if(y[i] * (x[i,] %*% w) <= 0){
    # update w
     w = w + y[i] * x[i,]
      }else{
        k = k+1  #if converge break
      }
    }
    W[e,] = w
    if(k == length(y)){
      break
    }
  }
  return(list(w=w,k=k,e=e,W=W))
}
```

Implementation

```
#load data
library(R.matlab)
```

```
## R.matlab v3.6.2 (2018-09-26) successfully loaded. See ?R.matlab for help.
```

```
##
## Attaching package: 'R.matlab'

## The following objects are masked from 'package:base':
##
##     getOption, isOpen
```

```
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
pathname = file.path("/home/grad/rw175/STA561Homework/Machine-Learning", "mnist_all.mat")
data = readMat(pathname)

train4 = data$train4
label4 = matrix(rep(1,nrow(train4)),byrow = TRUE)
train9 = data$train9
label9 = matrix(rep(-1,nrow(train9)),byrow = TRUE)
train_X = rbind(train4,train9)
train_y = rbind(label4,label9)
test4 = data$test4
test9 = data$test9
test_X = rbind(test4,test9)
test_y = rbind(matrix(rep(1,nrow(test4)),byrow = TRUE),matrix(rep(-1,nrow(test9)),byrow = TRUE))

###normalize data
min_train = apply(train_X,2,min)
max_train = apply(train_X,2,max)
normalize_fun = function(x){
  for(i in 1:ncol(x)){
    if(max_train[i] == min_train[i]){
      next
    }else{
      x[,i] = (x[,i] - min_train[i])/(max_train[i] - min_train[i])
    }
  }
  return(x)
}
train_X_norm = normalize_fun(train_X)
train_data = cbind(train_X_norm,train_y)

test_X_norm = normalize_fun(test_X)
test_data = cbind(test_X_norm,test_y)


#reorder
set.seed(2018)
order = sample(1:length(train_y),size = length(train_y),replace = FALSE)
train_data = train_data[order,]

result_per = perceptron(data=train_data,I=10)
```