ST340 Programming for Data Science

Assignment 3

Released: Monday week 8, 2019-11-18; Deadline: 12:00 on Monday week 11, 2019-12-09.

Instructions

- Work individually.
- Specify your student number and name on your assignment.
- Any programming should be in R. Your report should be created using R markdown. Submit a single knitted pdf document which includes any code you have written.

Q1 Gradient descent

Here is a function that does gradient descent with a fixed number of iterations to find local minima:

```
gradient.descent <- function(f, gradf, x0, iterations=1000, eta=0.2) {
  x<-x0
  for (i in 1:iterations) {
    cat(i,"/",iterations,": ",x," ",f(x),"\n")
    x<-x-eta*gradf(x)
  }
  x
}
```

Example:

```
f <-function(x) { sum(x^2) }
gradf<-function(x) { 2*x }
gradient.descent(f,gradf,c(10,20),10,0.2)
```

(a) Write a *short* function that uses `gradient.descent` to find a local *maximum*. (For the purpose of this question, `gradient.descent` is a "black box". Don't worry about the printed output, just the return value matters.)

i.e.

```
gradient.ascent <- function(f, df, x0, iterations=1000, eta=0.2) {
# ... use gradient.descent(...) here ...
}
f <-function(x) { (1+x^2)^(-1) }
gradf<-function(x) { -2*x*(1+x^2)^(-2) }
gradient.ascent(f,gradf,3,40,0.5)
```

(b) Consider the function $f : \mathbb{R}^2 \to \mathbb{R}$ given by

```
f <- function(x) (x[1]-1)^2 + 100*(x[1]^2-x[2])^2
```

i) Give a short mathematical proof that f has a unique minimum.

ii) Write a function `gradf` to calculate ∇f, i.e.
`gradf <- function(x) { # ... use x[1] and x[2] ... }`

iii) Starting from the point `x0=c(3,4)`, try to find the minimum using gradient descent. `gradient.descent(f,gradf,c(3,4), ... , ...)`

(c) Write a function to do gradient descent with momentum. Starting from the point `x0=c(3,4)`, use your function to find the minimum of the function from part (b).

Q2 Support vector machines

Run the following code to load the tiny MNIST dataset:

```
load("mnist.tiny.RData")
train.X=train.X/255
test.X=test.X/255
```

and then show some digits:

```
library(grid)
grid.raster(array(aperm(array(train.X[1:50,],c(5,10,28,28)),c(4,1,3,2)),c(140,280)),
              interpolate=FALSE)
```

(a) Use three-fold cross validation on the training set to compare SVMs with linear kernels, polynomial kernels and RBF kernels, i.e.

```
library(e1071)
svm(train.X,train.labels,type="C-classification",kernel="linear",cross=3)$tot.accuracy
svm(train.X,train.labels,type="C-classification",kernel="poly",
    degree=2,coef=1,cross=3)$tot.accuracy
```

etc. (The flag warning=FALSE is helpful here. What is the suppressed warning message warning you about?)

(b) For the RBF kernels, write a grid search function that takes two lists, `log.C.range` and `log.gamma.range`, and for each pair `(lc,lg)` of entries in the pair of lists attempts cross-validation with parameters `cost = exp(lc)` and `gamma=exp(lg)`. Once you have found the model with the best cross-validation error, train it on the full `tiny' training set and then test it on the`tiny' test set.