ST340 Programming for Data Science - Assignment 3

Matthew Nunes & Oliver Harvey

Friday 8 December 2017

Student number(s): 1508437 &

Student name(s): Matthew Nunes, Oliver Harvey

Q1

Here is a function that does gradient descent to find local minima:

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

Example:

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

Q1a

Write a *short* function that uses `gradient.descent` to find a local *maxima*. (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, alpha=0.2) {
... use gradient.descent(...) here ...
}
f <-function(x) { (1+x^2)^(-1) }
df<-function(x) { -2*x*(1+x^2)^(-2) }
gradient.ascent(f,df,3,40,0.5)
```

Q1b

Consider the function $f : R^2 \to R$

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

(1) Give a short mathematical proof that f has a a unique minima.
(2) Write a function df to calculate the gradient of f, i.e.
    `df <- function(x) { ... use x[1] and x[2] ...}`

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

Q1c

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

Q2

Load the tiny MNIST dataset:

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

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=F)
```

Use 3-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, etc.

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 training set and then test it on the test set.