NNS vs. kNN MNIST dataset

Intro

This is a quick note on comparing the output of the NNS clustering, regression and classification technique to the already well documented k-nearest neighbors (kNN) approach.

Load Required Packages in R NNS (>= 4.1)

```
require(devtools); install_github('OVVO-Financial/NNS', ref = "NNS-Beta-Version")
require(NNS)
require(data.table)
require(caret)
require(R.utils)
```

Download the MNIST dataset

The following routines download and load the MNIST dataset.

```
get.mnist <- function(dir=NULL) {
  # dir: the path containing the extracted files:
  # train-images-idx3-ubyte ; train-labels-idx1-ubyte
  # t10k-images-idx3-ubyte ; t10k-labels-idx1-ubyte
  if(is.null(dir)) {

    dir <- getwd()
    u <- c('http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz',
           'http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz',
           'http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz',
           'http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz')
    sapply(u, function(x) {
      download.file(x, f <- file.path(dir, basename(x)))
      gunzip(f) })
  }
}



load_mnist <- function() {
  load_image_file <- function(filename) {
    ret = list()
    f = file(filename,'rb')
    readBin(f,'integer',n=1,size=4,endian='big')
    ret$n = readBin(f,'integer',n=1,size=4,endian='big')
    nrow = readBin(f,'integer',n=1,size=4,endian='big')
    ncol = readBin(f,'integer',n=1,size=4,endian='big')
    x = readBin(f,'integer',n=ret$n*nrow*ncol,size=1,signed=F)
    ret$x = matrix(x, ncol=nrow*ncol, byrow=T)
    close(f)
    ret
```

```
  }
  load_label_file <- function(filename) {
    f = file(filename,'rb')
    readBin(f,'integer',n=1,size=4,endian='big')
    n = readBin(f,'integer',n=1,size=4,endian='big')
    y = readBin(f,'integer',n=n,size=1,signed=F)
    close(f)
    y
  }
  train <<- load_image_file('train-images-idx3-ubyte')
  test <<- load_image_file('t10k-images-idx3-ubyte')

  train$y <<- load_label_file('train-labels-idx1-ubyte')
  test$y <<- load_label_file('t10k-labels-idx1-ubyte')
}
train <- data.frame()
test <- data.frame()

# Load data.
load_mnist()
```

NNS.reg

NNS will treat this as a multivariate regression with each pixel as a feature, or regressor. There is no need to normalize the intensities to a 255 grey-scale.

NNS.reg(..., order = "max", n.best = 1,...) is similar to a kNN=1 methodology, but we can test to see how many clusters provide the best estimate by using the NNS.stack routine which iteratively checks the n.best parameter (analogous to the k in kNN).

```
start.time= Sys.time()
NNS.stack(IVs.train = train$x[1:6000,], DV.train = train$y[1:6000], method = 1, order='max',
                            obj.fn = expression ( mean(round(predicted)==as.numeric(actual)) ),
                            objective = "max")
```

```
## $NNS.reg.n.best
## [1] 1
##
## $OBJfn.reg
## [1] 0.8914667
##
## $NNS.dim.red.threshold
## [1] NA
##
## $OBJfn.dim.red
## [1] -Inf
##
## $reg
## NULL
##
## $dim.red
## [1] NA
##
## $stack
```

```
## NULL
```

```
print(Sys.time()-start.time)
```

```
## Time difference of 13.39868 mins
```

`n.best = 1` is our best result, so we will use it now in the `NNS.reg` on the full dataset, then check our accuracy.

```
start.time=Sys.time()

nns.predictions = NNS.reg(train$x, train$y, point.est = test$x,
         order='max', n.best = 1, plot = FALSE, residual.plot = FALSE)$Point.est


# Make sure estimates are between 0 and 9
nns.predictions = pmin(pmax(nns.predictions,0),9)

mean(round(nns.predictions)==test$y)
```

```
## [1] 0.9691
```

```
print(Sys.time()-start.time)
```

```
## Time difference of 29.33359 mins
```

Comments

The goal of this demonstration of `NNS` vs. `k means` for the MNIST dataset is to show the wide range of capability of `NNS` multivariate regression in machine learning applications.

`NNS` is not a one-trick pony, as it has been demonstrated to excel in time-series forecasting, nonlinear continuous regressions, and provide solutions for econometric applications. See the following examples:

- `NNS` Forecasting Presentation Download the .pdf file here: https://ssrn.com/abstract=3382300
- `NNS` Forecasting vs. KERAS LSTM Deep Learning View and download the .html file here: https://htmlpreview.github.io/?https://github.com/OVVO-Financial/NNS/blob/NNS-Beta-Version/examples/Sunspots_example.html
- `Classification Using NNS Clustering Analysis` https://ssrn.com/abstract=2864711
- `The 7 Reasons Most Econometric Investments Fail - NNS Contributions Towards Solutions` View and download the .html file here: https://htmlpreview.github.io/?https://github.com/OVVO-Financial/NNS/blob/NNS-Beta-Version/examples/7_Econometic_Reasons.html

I look forward to further discussions and collaboration with those equally as passionate about these issues, and open to embracing alternative solutions. If you found this presentation interesting or useful, please feel free to reach out via e-mail: ovvo.financial.systems@gmail.com

Thanks for your interest!