Solutions for ST340 Lab 8

2019–20

1: OR

Here is an example of using R's `optim` function to learn the OR function. `optim` works best when you provide a function to calculate the gradient, but we will be lazy for now: look at `?optim`.

```
logistic <- function(x) {
  1/(1+exp(-x))
}

(or.x <- matrix(c(0,0,1,1,0,1,0,1),4,2))
```

```
##      [,1] [,2]
## [1,]    0    0
## [2,]    0    1
## [3,]    1    0
## [4,]    1    1
```

```
(or.y <- c(0,1,1,1))
```

```
## [1] 0 1 1 1
```

```
ann <- function(x,theta) {
  w=theta[1:2]
  b=theta[3]
  logistic( x %*% w + b ) # The coefficient multiplying '1' is often called the 'bias';
  # here denoted b
}

cost=function(theta) {
  o=ann(or.x,theta)
  sum((o-or.y)^2)
}

(theta0=rep(0,3))
```

```
## [1] 0 0 0
```

```
(theta=optim(theta0,cost)$par)
```

```
## [1] 123.46667 200.00000 -42.43333
```

```
ann(or.x,theta) # Should be approximately or.y
```

```
##              [,1]
## [1,] 3.72767e-19
## [2,] 1.00000e+00
## [3,] 1.00000e+00
## [4,] 1.00000e+00
```

2: XOR

Adapt the above to learn the XOR function. Hint: you can write a two layer network in the efficient form

```
logistic(logistic(x %*% w + b) %*% w2 + b2)
```

where `w` is a matrix, `b` is a vector, `w2` is a vector, `b2` is a scalar, and the inner `logistic` is applied componentwise to a vector. You will probably find that the architecture for the OR example is not sufficiently big here; try adding a unit in the intermediate layer (then `theta` has length 13), and also randomising `theta0`.

```
xor.x <- matrix(c(0,0,1,1,0,1,0,1),4,2)
xor.y <- c(0,1,1,0)

ann <- function(x,theta) { # See accompanying note to explain this choice of architecture
  w1 <- matrix(theta[1:6],2,3)
  b1 <- theta[7:9]
  w2 <- matrix(theta[10:12],3,1)
  b2 <- theta[13]
  output <- logistic(logistic(x %*% w1 + b1) %*% w2 + b2)
  return(output)
}

cost <- function(theta) {
  o <- ann(xor.x,theta)
  sum((o-xor.y)^2)
}

(theta0 <- runif(13,-1,1))
```

```
##  [1] -0.225899730  0.006204632  0.549619001 -0.947594133  0.997113558
##  [6] -0.541831142  0.931370878  0.469137451 -0.127952222 -0.467712312
## [11]  0.826159131 -0.557704251 -0.765511549
```

```
(theta <- optim(theta0,cost)$par)
```

```
##  [1]   8.8520975  -0.1841422   6.7310228  10.1027018   9.6864065
##  [6] -23.3447865  39.1635734 -70.3254779  14.8896643 -11.6193788
## [11]  46.2702200 -22.4079728  10.6102367
```

```
ann(xor.x,theta) #Should be approximately xor.y
```

```
##              [,1]
## [1,] 6.762068e-11
## [2,] 1.000000e+00
## [3,] 1.000000e+00
## [4,] 1.069079e-08
```

3: ANNs

(a) Read through the following code to see how a basic ANN can be implemented in `R`.

```
# Used to store the data as it travels through the network
layer <- function() {
  a=new.env()
  # a$x will be used to store the input and hidden layers
  # a$dx stores the derivatives of the cost function with respect to l$x
```

```r
  a
}

# Implement a 'layer' to apply the logistic function
logistic<-function() {
  e=new.env()
  e$forward<-function(a,z,train) {
# a is the input to the layer
# z is the output from the layer
# "train" is "true" during training, and false when the network is being tested on
# new data; most layers will ignore the "train" variable, but it is needed for consistency
z$x <- 1/(1+exp(-a$x))
  }
  e$backward<-function(a,z,learning.rate) {
a$dx <- z$dx * z$x * (1-z$x) # Backpropagation just through the logistic function. Check!
  }
  e
}

# Implement a fully connected layer: each output depends on every input.
fully.connected<-function(nIn,nOut) {
  # nIn is size of the input layer
  # nOut is the size of the output layer
  shape=c(nIn,nOut)
  e=new.env()
  e$w=array(runif(prod(shape),-0.1,0.1),shape) # Network parameters - connection weights
  e$mw=array(0,shape) # w-momentum
  shape[1]=1
  e$b=array(0,shape) # Network parameters - bias term
  e$mb=array(0,shape) # b-momentum
  e$forward <- function(a,z,train) {
z$x <- a$x %*% e$w + e$b[rep(1,dim(a$x)[1]),]
  }
  e$backward <- function(a,z,learning.rate) {
a$dx <- z$dx %*% t(e$w) # Backpropagation just through the 'linear combination' step. Check!
dw    <- t(a$x) %*% z$dx
db    <- apply(z$dx,2,sum)

w.new <- e$w - 0.1*learning.rate*dw + 0.9*e$mw
e$mw <- w.new - e$w
e$w <- w.new
b.new <- e$b - 0.1*learning.rate*db + 0.9*e$mb
e$mb <- b.new - e$b
e$b <- b.new
  }
  e
}

softmax.nll.classifier <- function(a,y) {
  weights <- exp(a$x-apply(a$x,1,max))  # (subtract column sums: better numerically)
  C <- apply(weights,1,sum)
  softmax <- weights/C
  predictions <- apply(softmax,1,which.max)-1
  errors <- sum(predictions!=y)
  target <- diag(dim(softmax)[2])[y+1,] # one-hot encoding of the true label
  a$dx <- softmax - target
  cost <- sum(-target*log(softmax),na.rm=TRUE) # negative log likelihood
  list(errors=errors,cost=cost)
}

train.classification <- function(nn,train.X,train.labels,batch.size,learning.rate) {
  errors <- 0
  cost<- 0
  layers <- replicate(length(nn)+1,layer())
  n=length(nn)
  n.reps=ceiling(dim(train.X)[1]/batch.size)
  for (rep in 1:n.reps) {
p=sample(dim(train.X)[1],batch.size)
if (length(dim(train.X))==2)
  layers[[1]]$x <- train.X[p,,drop=FALSE]
if (length(dim(train.X))==4)
  layers[[1]]$x <- train.X[p,,,,drop=FALSE]
y=train.labels[p]
for (i in 1:n) {
  nn[[i]]$forward(layers[[i]],layers[[i+1]],TRUE)
}
s <- softmax.nll.classifier(layers[[n+1]],y)
errors <- errors + s$errors
cost <- cost + s$cost
for (i in n:1) {
  nn[[i]]$backward(layers[[i]],layers[[i+1]],learning.rate)
}
  }
  print(paste("Training errors:",errors/n.reps/batch.size*100,"%   Cost:",
          cost/n.reps/batch.size))
}

test.classification <- function(nn,test.X,test.labels,batch.size) {
  errors <- 0
  layers <- replicate(length(nn)+1,layer())
  n=length(nn)
  n.test=dim(test.X)[1]
  n.reps=ceiling(n.test/batch.size)
  for (rep in 1:n.reps) {
p=(batch.size*(rep-1)+1):min(batch.size*rep,n.test)
if (length(dim(test.X))==2)
  layers[[1]]$x <- test.X[p,,drop=FALSE]
if (length(dim(test.X))==4)
  layers[[1]]$x <- test.X[p,,,,drop=FALSE]
y=test.labels[p]
for (i in 1:n) {
  nn[[i]]$forward(layers[[i]],layers[[i+1]],FALSE)
}
s <- softmax.nll.classifier(layers[[n+1]],y)
errors <- errors + s$errors
  }
  print(paste("Test errors:",errors/dim(test.X)[1]*100,"%"))
```

```
}
```

(b) Run this code on the MNIST data using a small, fully-connected network.

```r
load("mnist.RData")
train.X <- train.X/255
test.X <- test.X/255
ls()
```

```
##  [1] "ann"                       "cost"
##  [3] "fully.connected"           "layer"
##  [5] "logistic"                  "nClasses"
##  [7] "or.x"                      "or.y"
##  [9] "softmax.nll.classifier" "test.classification"
## [11] "test.labels"               "test.X"
## [13] "test.Y"                    "theta"
## [15] "theta0"                    "train.classification"
## [17] "train.labels"              "train.X"
## [19] "train.Y"                   "xor.x"
## [21] "xor.y"
```

```r
input.dim <- dim(train.X)[2] #784
n.classes <- max(train.labels)+1 #10
hidden.layer.size <- 100
batch.size <- 100
learning.rate <- 0.001
nn=list(
  fully.connected(input.dim,hidden.layer.size),
  logistic(),
  fully.connected(hidden.layer.size,hidden.layer.size),
  logistic(),
  fully.connected(hidden.layer.size,hidden.layer.size),
  logistic(),
  fully.connected(hidden.layer.size,n.classes)
)
for (i in 1:100) { # <- Increase this if you have time
  train.classification(nn,train.X,train.labels,batch.size,learning.rate)
  # (Expect 90% error initially)
  test.classification(nn,test.X,test.labels,batch.size)
}
```

```
## [1] "Training errors: 89.42 %   Cost: 2.30628867303512"
## [1] "Test errors: 89.68 %"
## [1] "Training errors: 88.7566666666667 %   Cost: 2.30232524861235"
## [1] "Test errors: 90.18 %"
## [1] "Training errors: 86.7516666666667 %   Cost: 2.29223804414518"
## [1] "Test errors: 89.72 %"
## [1] "Training errors: 74.7216666666667 %   Cost: 2.14679975350814"
## [1] "Test errors: 63.13 %"
## [1] "Training errors: 51.325 %   Cost: 1.52746923198789"
## [1] "Test errors: 35.3 %"
## [1] "Training errors: 28.965 %   Cost: 0.894663815104113"
## [1] "Test errors: 24.59 %"
## [1] "Training errors: 20.8716666666667 %   Cost: 0.693967967757182"
## [1] "Test errors: 18.21 %"
## [1] "Training errors: 16.7316666666667 %   Cost: 0.591214701347811"
## [1] "Test errors: 14.89 %"
## [1] "Training errors: 13.9833333333333 %   Cost: 0.515656467618919"
## [1] "Test errors: 13.02 %"
## [1] "Training errors: 12.41 %   Cost: 0.466720782401637"
## [1] "Test errors: 11.7 %"
## [1] "Training errors: 11.1633333333333 %   Cost: 0.422168889361964"
## [1] "Test errors: 10.57 %"
## [1] "Training errors: 10.1216666666667 %   Cost: 0.381461646836066"
## [1] "Test errors: 9.52 %"
## [1] "Training errors: 9.29 %   Cost: 0.343291115126596"
## [1] "Test errors: 8.8 %"
## [1] "Training errors: 8.31666666666667 %   Cost: 0.309045711214649"
## [1] "Test errors: 7.89 %"
## [1] "Training errors: 7.715 %   Cost: 0.283750585712768"
## [1] "Test errors: 7.14 %"
## [1] "Training errors: 6.94333333333333 %   Cost: 0.253417762311801"
## [1] "Test errors: 6.77 %"
## [1] "Training errors: 6.40666666666667 %   Cost: 0.236524510785134"
## [1] "Test errors: 6.24 %"
## [1] "Training errors: 5.76166666666667 %   Cost: 0.212320004051349"
## [1] "Test errors: 5.89 %"
## [1] "Training errors: 5.52166666666667 %   Cost: 0.200515618953447"
## [1] "Test errors: 5.92 %"
## [1] "Training errors: 5.08833333333333 %   Cost: 0.186611507365171"
## [1] "Test errors: 5.55 %"
## [1] "Training errors: 4.59166666666667 %   Cost: 0.171288569665205"
## [1] "Test errors: 5.43 %"
## [1] "Training errors: 4.45333333333333 %   Cost: 0.162734179506488"
## [1] "Test errors: 5.13 %"
## [1] "Training errors: 4.21833333333333 %   Cost: 0.150464938810977"
## [1] "Test errors: 5.04 %"
## [1] "Training errors: 4.10166666666667 %   Cost: 0.148659669797706"
## [1] "Test errors: 4.81 %"
## [1] "Training errors: 3.78333333333333 %   Cost: 0.139082361375048"
## [1] "Test errors: 4.69 %"
## [1] "Training errors: 3.765 %   Cost: 0.13685528404309"
## [1] "Test errors: 4.52 %"
## [1] "Training errors: 3.69333333333333 %   Cost: 0.130991057661733"
## [1] "Test errors: 4.47 %"
## [1] "Training errors: 3.33 %   Cost: 0.119674086417139"
## [1] "Test errors: 4.18 %"
## [1] "Training errors: 3.08833333333333 %   Cost: 0.111186123915743"
## [1] "Test errors: 4.47 %"
## [1] "Training errors: 3.13833333333333 %   Cost: 0.11271411808614"
## [1] "Test errors: 4.29 %"
## [1] "Training errors: 3.05833333333333 %   Cost: 0.110349859928642"
## [1] "Test errors: 3.93 %"
## [1] "Training errors: 2.865 %   Cost: 0.103530978924325"
## [1] "Test errors: 3.97 %"
## [1] "Training errors: 2.805 %   Cost: 0.0995758843992099"
## [1] "Test errors: 4.06 %"
## [1] "Training errors: 2.68166666666667 %   Cost: 0.0945666802538752"
## [1] "Test errors: 3.99 %"
## [1] "Training errors: 2.51166666666667 %   Cost: 0.0913461361189316"
## [1] "Test errors: 3.81 %"
## [1] "Training errors: 2.52666666666667 %   Cost: 0.0898665037669896"
## [1] "Test errors: 3.6 %"
## [1] "Training errors: 2.215 %   Cost: 0.0828085010715441"
## [1] "Test errors: 3.74 %"
## [1] "Training errors: 2.34166666666667 %   Cost: 0.0839214782535235"
## [1] "Test errors: 3.55 %"
## [1] "Training errors: 2.21333333333333 %   Cost: 0.0802185009100318"
## [1] "Test errors: 3.49 %"
## [1] "Training errors: 2.21333333333333 %   Cost: 0.0798540139917258"
## [1] "Test errors: 3.58 %"
## [1] "Training errors: 2.13 %   Cost: 0.0769383196432137"
## [1] "Test errors: 3.44 %"
## [1] "Training errors: 2.07833333333333 %   Cost: 0.0735279193344968"
## [1] "Test errors: 3.53 %"
## [1] "Training errors: 1.88 %   Cost: 0.071213527044199"
## [1] "Test errors: 3.46 %"
## [1] "Training errors: 1.89166666666667 %   Cost: 0.0680894486130021"
## [1] "Test errors: 3.42 %"
## [1] "Training errors: 1.66666666666667 %   Cost: 0.0622779635191207"
## [1] "Test errors: 3.39 %"
## [1] "Training errors: 1.725 %   Cost: 0.0632116972923098"
## [1] "Test errors: 3.37 %"
## [1] "Training errors: 1.71333333333333 %   Cost: 0.0611730691230323"
## [1] "Test errors: 3.28 %"
## [1] "Training errors: 1.65833333333333 %   Cost: 0.0608586368784184"
## [1] "Test errors: 3.41 %"
## [1] "Training errors: 1.60166666666667 %   Cost: 0.0588421542352415"
## [1] "Test errors: 3.14 %"
## [1] "Training errors: 1.51 %   Cost: 0.0542482150494959"
## [1] "Test errors: 3.16 %"
## [1] "Training errors: 1.46666666666667 %   Cost: 0.0541855015353294"
## [1] "Test errors: 3.29 %"
## [1] "Training errors: 1.35833333333333 %   Cost: 0.0522278004167682"
## [1] "Test errors: 3.44 %"
## [1] "Training errors: 1.31333333333333 %   Cost: 0.0519152491022742"
## [1] "Test errors: 3.27 %"
## [1] "Training errors: 1.31166666666667 %   Cost: 0.0486720278568313"
## [1] "Test errors: 3.05 %"
## [1] "Training errors: 1.11166666666667 %   Cost: 0.0454442037741745"
## [1] "Test errors: 3.15 %"
## [1] "Training errors: 1.24333333333333 %   Cost: 0.04694173443582"
## [1] "Test errors: 3.14 %"
## [1] "Training errors: 1.14333333333333 %   Cost: 0.0442982069093186"
## [1] "Test errors: 3.18 %"
## [1] "Training errors: 1.17333333333333 %   Cost: 0.0452623675700245"
## [1] "Test errors: 3.2 %"
## [1] "Training errors: 1.08666666666667 %   Cost: 0.0429451261982234"
## [1] "Test errors: 3.08 %"
## [1] "Training errors: 1.03333333333333 %   Cost: 0.0416541029960118"
## [1] "Test errors: 3.14 %"
## [1] "Training errors: 1.00166666666667 %   Cost: 0.0403088041692722"
## [1] "Test errors: 3.09 %"
## [1] "Training errors: 0.941666666666667 %   Cost: 0.03679872283225"
## [1] "Test errors: 3.14 %"
## [1] "Training errors: 0.898333333333333 %   Cost: 0.0369811747460958"
## [1] "Test errors: 3.04 %"
## [1] "Training errors: 0.86 %   Cost: 0.0345364138182158"
## [1] "Test errors: 3.09 %"
## [1] "Training errors: 0.841666666666667 %   Cost: 0.033860446460191"
## [1] "Test errors: 3.03 %"
## [1] "Training errors: 0.833333333333333 %   Cost: 0.0325228780172081"
## [1] "Test errors: 2.92 %"
## [1] "Training errors: 0.771666666666667 %   Cost: 0.0334923390335821"
## [1] "Test errors: 3.11 %"
## [1] "Training errors: 0.728333333333333 %   Cost: 0.0306616379213242"
## [1] "Test errors: 2.92 %"
## [1] "Training errors: 0.725 %   Cost: 0.0304806034759264"
## [1] "Test errors: 3.05 %"
## [1] "Training errors: 0.718333333333333 %   Cost: 0.0300095667782351"
## [1] "Test errors: 3.01 %"
## [1] "Training errors: 0.706666666666667 %   Cost: 0.0292444142073081"
## [1] "Test errors: 3.06 %"
## [1] "Training errors: 0.578333333333333 %   Cost: 0.0275527445457048"
## [1] "Test errors: 3.03 %"
## [1] "Training errors: 0.62 %   Cost: 0.0266602100039143"
## [1] "Test errors: 3.08 %"
## [1] "Training errors: 0.518333333333333 %   Cost: 0.0242543150016683"
## [1] "Test errors: 3.02 %"
## [1] "Training errors: 0.501666666666667 %   Cost: 0.0237475196049736"
## [1] "Test errors: 3.06 %"
## [1] "Training errors: 0.465 %   Cost: 0.0237907711719201"
## [1] "Test errors: 2.97 %"
## [1] "Training errors: 0.466666666666667 %   Cost: 0.022442985201601"
## [1] "Test errors: 3.01 %"
## [1] "Training errors: 0.458333333333333 %   Cost: 0.0217771486410223"
## [1] "Test errors: 2.98 %"
## [1] "Training errors: 0.42 %   Cost: 0.0225215447053523"
## [1] "Test errors: 2.92 %"
## [1] "Training errors: 0.436666666666667 %   Cost: 0.0230607213662056"
## [1] "Test errors: 3 %"
## [1] "Training errors: 0.385 %   Cost: 0.0209462200454468"
## [1] "Test errors: 2.85 %"
## [1] "Training errors: 0.385 %   Cost: 0.018817329386149"
## [1] "Test errors: 2.85 %"
## [1] "Training errors: 0.351666666666667 %   Cost: 0.0186943808558771"
## [1] "Test errors: 2.98 %"
## [1] "Training errors: 0.383333333333333 %   Cost: 0.0187474398696415"
## [1] "Test errors: 3.03 %"
## [1] "Training errors: 0.311666666666667 %   Cost: 0.0179591200754011"
## [1] "Test errors: 2.98 %"
## [1] "Training errors: 0.268333333333333 %   Cost: 0.0168955411242329"
## [1] "Test errors: 2.91 %"
## [1] "Training errors: 0.251666666666667 %   Cost: 0.0159579433632576"
## [1] "Test errors: 2.96 %"
## [1] "Training errors: 0.293333333333333 %   Cost: 0.0160291985723202"
## [1] "Test errors: 2.96 %"
## [1] "Training errors: 0.291666666666667 %   Cost: 0.0153425450570044"
## [1] "Test errors: 2.97 %"
## [1] "Training errors: 0.248333333333333 %   Cost: 0.015118725782837"
## [1] "Test errors: 2.96 %"
## [1] "Training errors: 0.203333333333333 %   Cost: 0.0140452568451477"
## [1] "Test errors: 2.96 %"
## [1] "Training errors: 0.24 %   Cost: 0.0140722024889596"
## [1] "Test errors: 2.92 %"
## [1] "Training errors: 0.19 %   Cost: 0.0131709333821148"
## [1] "Test errors: 2.97 %"
## [1] "Training errors: 0.17 %   Cost: 0.0135797648305143"
## [1] "Test errors: 2.89 %"
## [1] "Training errors: 0.156666666666667 %   Cost: 0.0119909640858352"
## [1] "Test errors: 2.94 %"
## [1] "Training errors: 0.151666666666667 %   Cost: 0.0122632731579011"
## [1] "Test errors: 2.91 %"
## [1] "Training errors: 0.143333333333333 %   Cost: 0.0119853263560905"
## [1] "Test errors: 2.9 %"
## [1] "Training errors: 0.15 %   Cost: 0.0113133144096897"
## [1] "Test errors: 2.88 %"
## [1] "Training errors: 0.14 %   Cost: 0.0109742949729251"
## [1] "Test errors: 2.81 %"
## [1] "Training errors: 0.148333333333333 %   Cost: 0.0111769040979632"
## [1] "Test errors: 2.88 %"
```

(c) Run this code on the CIFAR-10 subset using a small, fully-connected network.

```r
load("frog-horse.RData")
train.X <- train.X/255
test.X <- test.X/255
ls()
```

```
##  [1] "ann"                  "batch.size"
##  [3] "cost"                 "fully.connected"
##  [5] "hidden.layer.size"    "i"
##  [7] "input.dim"            "layer"
##  [9] "learning.rate"        "logistic"
## [11] "n.classes"            "nClasses"
## [13] "nn"                   "or.x"
## [15] "or.y"                 "softmax.nll.classifier"
## [17] "test.classification"  "test.labels"
## [19] "test.X"               "test.Y"
## [21] "theta"                "theta0"
## [23] "train.classification" "train.labels"
## [25] "train.X"              "train.Y"
## [27] "xor.x"                "xor.y"
```

```r
input.dim <- dim(train.X)[2] #3072
n.classes <- max(train.labels)+1 #2
hidden.layer.size <- 100
batch.size <- 100
learning.rate <- 0.001
nn=list(
  fully.connected(input.dim,hidden.layer.size),
  logistic(),
  fully.connected(hidden.layer.size,hidden.layer.size),
  logistic(),
  fully.connected(hidden.layer.size,hidden.layer.size),
  logistic(),
  fully.connected(hidden.layer.size,n.classes)
)
for (i in 1:100) { # <- Increase this if you have time
  train.classification(nn,train.X,train.labels,batch.size,learning.rate)
  # (Expect 50% error rate initially)
  test.classification(nn,test.X,test.labels,batch.size)
}
```

```
## [1] "Training errors: 48.6 %   Cost: 0.830880850311418"
## [1] "Test errors: 50 %"
## [1] "Training errors: 50.7 %   Cost: 0.708939420978511"
## [1] "Test errors: 50 %"
## [1] "Training errors: 51.7 %   Cost: 0.6989761603805"
## [1] "Test errors: 50 %"
## [1] "Training errors: 44.6 %   Cost: 0.692758603366839"
## [1] "Test errors: 50 %"
## [1] "Training errors: 50.3 %   Cost: 0.696142419661487"
## [1] "Test errors: 50 %"
## [1] "Training errors: 51.4 %   Cost: 0.696026810355934"
## [1] "Test errors: 50 %"
## [1] "Training errors: 50.9 %   Cost: 0.697785453259224"
## [1] "Test errors: 50 %"
## [1] "Training errors: 48.5 %   Cost: 0.697712948913749"
## [1] "Test errors: 50 %"
## [1] "Training errors: 47.8 %   Cost: 0.696214589778707"
## [1] "Test errors: 50 %"
## [1] "Training errors: 48.9 %   Cost: 0.696860600524977"
## [1] "Test errors: 50 %"
## [1] "Training errors: 48.6 %   Cost: 0.698088665137251"
## [1] "Test errors: 50 %"
## [1] "Training errors: 49.4 %   Cost: 0.696468824993814"
## [1] "Test errors: 50 %"
## [1] "Training errors: 49.6 %   Cost: 0.694682037104658"
## [1] "Test errors: 50 %"
## [1] "Training errors: 48.8 %   Cost: 0.69265278985215"
## [1] "Test errors: 50 %"
## [1] "Training errors: 50 %   Cost: 0.697058402418517"
## [1] "Test errors: 50 %"
## [1] "Training errors: 51.4 %   Cost: 0.701178559810666"
## [1] "Test errors: 50 %"
## [1] "Training errors: 50.6 %   Cost: 0.696497128741878"
## [1] "Test errors: 50 %"
## [1] "Training errors: 49.6 %   Cost: 0.695395578099508"
## [1] "Test errors: 50 %"
## [1] "Training errors: 51.1 %   Cost: 0.697629964940563"
## [1] "Test errors: 50 %"
## [1] "Training errors: 50.8 %   Cost: 0.694489318296211"
## [1] "Test errors: 50 %"
## [1] "Training errors: 47.8 %   Cost: 0.694478410241528"
## [1] "Test errors: 50 %"
## [1] "Training errors: 48.7 %   Cost: 0.695131212701767"
## [1] "Test errors: 50 %"
## [1] "Training errors: 47.8 %   Cost: 0.694148905103666"
## [1] "Test errors: 50 %"
## [1] "Training errors: 45.6 %   Cost: 0.691617366074611"
## [1] "Test errors: 50 %"
## [1] "Training errors: 49.3 %   Cost: 0.697079518304986"
## [1] "Test errors: 50 %"
## [1] "Training errors: 48.2 %   Cost: 0.694065145759491"
## [1] "Test errors: 50 %"
## [1] "Training errors: 53.6 %   Cost: 0.708630461888969"
## [1] "Test errors: 50 %"
## [1] "Training errors: 48 %   Cost: 0.69988847152443"
## [1] "Test errors: 50 %"
## [1] "Training errors: 53.3 %   Cost: 0.700773725080626"
## [1] "Test errors: 50 %"
## [1] "Training errors: 51.6 %   Cost: 0.696310625953336"
## [1] "Test errors: 50 %"
## [1] "Training errors: 49.8 %   Cost: 0.694676854407156"
## [1] "Test errors: 50 %"
## [1] "Training errors: 53.5 %   Cost: 0.696210033043789"
## [1] "Test errors: 50 %"
## [1] "Training errors: 49 %   Cost: 0.69959841605299"
## [1] "Test errors: 50 %"
## [1] "Training errors: 46.8 %   Cost: 0.69681845221449"
## [1] "Test errors: 50 %"
## [1] "Training errors: 48 %   Cost: 0.695723426613555"
## [1] "Test errors: 50 %"
## [1] "Training errors: 49.1 %   Cost: 0.694025267369244"
## [1] "Test errors: 50 %"
## [1] "Training errors: 46.9 %   Cost: 0.693430571981582"
## [1] "Test errors: 50 %"
## [1] "Training errors: 48.5 %   Cost: 0.700701905406417"
## [1] "Test errors: 50 %"
## [1] "Training errors: 51.8 %   Cost: 0.704722391137596"
## [1] "Test errors: 50 %"
## [1] "Training errors: 49.8 %   Cost: 0.69582915910377"
## [1] "Test errors: 50 %"
## [1] "Training errors: 49.6 %   Cost: 0.692850549296623"
## [1] "Test errors: 50 %"
## [1] "Training errors: 47.6 %   Cost: 0.694403735887494"
## [1] "Test errors: 50 %"
## [1] "Training errors: 46.4 %   Cost: 0.700296503327139"
## [1] "Test errors: 50 %"
## [1] "Training errors: 50.6 %   Cost: 0.693684152538626"
## [1] "Test errors: 50 %"
## [1] "Training errors: 48.2 %   Cost: 0.691971197435997"
## [1] "Test errors: 50 %"
## [1] "Training errors: 52 %   Cost: 0.699005572040548"
## [1] "Test errors: 47.9 %"
## [1] "Training errors: 46 %   Cost: 0.690324196237647"
## [1] "Test errors: 50 %"
## [1] "Training errors: 45.1 %   Cost: 0.692801022183844"
## [1] "Test errors: 50 %"
## [1] "Training errors: 50.4 %   Cost: 0.696192575149686"
## [1] "Test errors: 50 %"
## [1] "Training errors: 48.2 %   Cost: 0.693218391991845"
## [1] "Test errors: 50 %"
## [1] "Training errors: 50.5 %   Cost: 0.696575436581726"
## [1] "Test errors: 50 %"
## [1] "Training errors: 46.1 %   Cost: 0.689366391314268"
## [1] "Test errors: 50 %"
## [1] "Training errors: 51.7 %   Cost: 0.69548338769438"
## [1] "Test errors: 50 %"
## [1] "Training errors: 50.6 %   Cost: 0.695411884075211"
## [1] "Test errors: 50 %"
## [1] "Training errors: 48.1 %   Cost: 0.694338986548993"
## [1] "Test errors: 50 %"
## [1] "Training errors: 49.4 %   Cost: 0.694331834916029"
## [1] "Test errors: 34.2 %"
## [1] "Training errors: 45.4 %   Cost: 0.692064067622899"
## [1] "Test errors: 41.2 %"
## [1] "Training errors: 44.1 %   Cost: 0.690981616300054"
## [1] "Test errors: 38.5 %"
## [1] "Training errors: 47.1 %   Cost: 0.692517688164393"
## [1] "Test errors: 50 %"
## [1] "Training errors: 47.7 %   Cost: 0.690747730948283"
## [1] "Test errors: 50 %"
## [1] "Training errors: 45.4 %   Cost: 0.689220712665581"
## [1] "Test errors: 50 %"
## [1] "Training errors: 51.5 %   Cost: 0.702509649518792"
## [1] "Test errors: 50 %"
## [1] "Training errors: 49.3 %   Cost: 0.692454558253825"
## [1] "Test errors: 50 %"
## [1] "Training errors: 49.4 %   Cost: 0.695432651189264"
## [1] "Test errors: 50 %"
## [1] "Training errors: 49.3 %   Cost: 0.692673939391012"
## [1] "Test errors: 50 %"
## [1] "Training errors: 48.9 %   Cost: 0.691547827039657"
## [1] "Test errors: 50 %"
## [1] "Training errors: 49.3 %   Cost: 0.691987901616453"
## [1] "Test errors: 49.8 %"
## [1] "Training errors: 50.2 %   Cost: 0.692898320107878"
## [1] "Test errors: 50 %"
## [1] "Training errors: 48.4 %   Cost: 0.693161791513851"
## [1] "Test errors: 50 %"
## [1] "Training errors: 48.5 %   Cost: 0.691727484975368"
## [1] "Test errors: 50 %"
## [1] "Training errors: 49.4 %   Cost: 0.693231631940529"
## [1] "Test errors: 35.5 %"
## [1] "Training errors: 44.3 %   Cost: 0.689916400065053"
## [1] "Test errors: 42.5 %"
## [1] "Training errors: 44.2 %   Cost: 0.68895735623173"
## [1] "Test errors: 50 %"
## [1] "Training errors: 45.9 %   Cost: 0.68752170616084"
## [1] "Test errors: 50 %"
## [1] "Training errors: 51.1 %   Cost: 0.699142398425796"
## [1] "Test errors: 50 %"
## [1] "Training errors: 47.5 %   Cost: 0.691363597568982"
## [1] "Test errors: 50 %"
## [1] "Training errors: 48.8 %   Cost: 0.692235468849833"
## [1] "Test errors: 33.9 %"
## [1] "Training errors: 37.7 %   Cost: 0.687786897351194"
## [1] "Test errors: 38.4 %"
## [1] "Training errors: 46.3 %   Cost: 0.689273753348594"
## [1] "Test errors: 50 %"
## [1] "Training errors: 46.6 %   Cost: 0.689580851049864"
## [1] "Test errors: 50 %"
## [1] "Training errors: 50.5 %   Cost: 0.699478815010315"
## [1] "Test errors: 50 %"
## [1] "Training errors: 47 %   Cost: 0.693974950642922"
## [1] "Test errors: 50 %"
## [1] "Training errors: 46.7 %   Cost: 0.694415692052353"
## [1] "Test errors: 50 %"
## [1] "Training errors: 47.6 %   Cost: 0.686844090333783"
## [1] "Test errors: 50 %"
## [1] "Training errors: 49.6 %   Cost: 0.692046395446968"
## [1] "Test errors: 45.5 %"
## [1] "Training errors: 49.5 %   Cost: 0.692851970404746"
## [1] "Test errors: 50 %"
## [1] "Training errors: 50 %   Cost: 0.693928138257113"
## [1] "Test errors: 49.1 %"
## [1] "Training errors: 43.4 %   Cost: 0.684472240326902"
## [1] "Test errors: 50 %"
## [1] "Training errors: 45.1 %   Cost: 0.685515069428027"
## [1] "Test errors: 49.9 %"
## [1] "Training errors: 45.5 %   Cost: 0.68665022812545"
## [1] "Test errors: 48 %"
## [1] "Training errors: 45.6 %   Cost: 0.685971323655239"
## [1] "Test errors: 49.7 %"
## [1] "Training errors: 47 %   Cost: 0.683534556681701"
## [1] "Test errors: 44.5 %"
## [1] "Training errors: 40.8 %   Cost: 0.68332933075079"
## [1] "Test errors: 39.4 %"
## [1] "Training errors: 42.7 %   Cost: 0.685950880180453"
## [1] "Test errors: 48.1 %"
## [1] "Training errors: 45.6 %   Cost: 0.683627541117316"
## [1] "Test errors: 50 %"
## [1] "Training errors: 46.6 %   Cost: 0.680910388912685"
## [1] "Test errors: 50 %"
## [1] "Training errors: 47.5 %   Cost: 0.686833367891105"
## [1] "Test errors: 33.6 %"
## [1] "Training errors: 39.2 %   Cost: 0.68091585490522"
## [1] "Test errors: 50 %"
## [1] "Training errors: 39.2 %   Cost: 0.679629364268368"
## [1] "Test errors: 37 %"
## [1] "Training errors: 45.9 %   Cost: 0.681888197420286"
## [1] "Test errors: 38.5 %"
```

4: Alternative activation functions

Fill in the gaps below to create two functions that can be used instead of the logistic function defined above.

```
Tanh<-function() { # tanh nonlinearity
  e=new.env()
  e$forward<-function(a,z,train) {
    z$x <- tanh(-a$x)  # Use tanh instead of the logistic function
  }
  e$backward<-function(a,z,learning.rate) {
    a$dx <- z$dx * (1+z$x) * (1-z$x)
  }
  e
}

relu<-function() { # Rectified Linear Units -- positive part function nonlinearity
  e=new.env()
  e$forward<-function(a,z,train) {
    z$x <- a$x * (a$x>0)
  }
  e$backward<-function(a,z,learning.rate) {
    a$dx <- z$dx * (a$x>0)
  }
  e
}

load("mnist.RData")
train.X <- train.X/255
test.X <- test.X/255
ls()
```

```
##  [1] "ann"                     "batch.size"
##  [3] "cost"                    "fully.connected"
##  [5] "hidden.layer.size"       "i"
##  [7] "input.dim"               "layer"
##  [9] "learning.rate"           "logistic"
## [11] "n.classes"               "nClasses"
## [13] "nn"                      "or.x"
## [15] "or.y"                    "relu"
## [17] "softmax.nll.classifier" "Tanh"
## [19] "test.classification"     "test.labels"
## [21] "test.X"                  "test.Y"
## [23] "theta"                   "theta0"
## [25] "train.classification"    "train.labels"
## [27] "train.X"                 "train.Y"
## [29] "xor.x"                   "xor.y"
```

```
input.dim <- dim(train.X)[2] #784
n.classes <- max(train.labels)+1 #10
hidden.layer.size <- 100
batch.size <- 100
learning.rate <- 0.001
nn=list(
  fully.connected(input.dim,hidden.layer.size),
  relu(),
```

```r
  fully.connected(hidden.layer.size,hidden.layer.size),
  relu(),
  fully.connected(hidden.layer.size,hidden.layer.size),
  relu(),
  fully.connected(hidden.layer.size,n.classes)
)
for (i in 1:100) { # <- Increase this if you have time
  train.classification(nn,train.X,train.labels,batch.size,learning.rate)
  # (Expect 90% error initially)
  test.classification(nn,test.X,test.labels,batch.size)
}
```

```
## [1] "Training errors: 26.0783333333333 %   Cost: 0.833275580264324"
## [1] "Test errors: 8.69 %"
## [1] "Training errors: 6.565 %   Cost: 0.22564408528024"
## [1] "Test errors: 5.63 %"
## [1] "Training errors: 4.575 %   Cost: 0.156593229565857"
## [1] "Test errors: 4.62 %"
## [1] "Training errors: 3.56166666666667 %   Cost: 0.12214277774866"
## [1] "Test errors: 3.65 %"
## [1] "Training errors: 2.86666666666667 %   Cost: 0.0939465889025541"
## [1] "Test errors: 3.5 %"
## [1] "Training errors: 2.395 %   Cost: 0.0798230275682762"
## [1] "Test errors: 3.15 %"
## [1] "Training errors: 1.99833333333333 %   Cost: 0.0690395203723024"
## [1] "Test errors: 3 %"
## [1] "Training errors: 1.81166666666667 %   Cost: 0.0601272023494485"
## [1] "Test errors: 3.02 %"
## [1] "Training errors: 1.625 %   Cost: 0.0529172512419231"
## [1] "Test errors: 2.73 %"
## [1] "Training errors: 1.48833333333333 %   Cost: 0.0497731070610757"
## [1] "Test errors: 2.75 %"
## [1] "Training errors: 1.26666666666667 %   Cost: 0.0398133080122701"
## [1] "Test errors: 2.7 %"
## [1] "Training errors: 1.11833333333333 %   Cost: 0.0352207113520533"
## [1] "Test errors: 2.7 %"
## [1] "Training errors: 0.893333333333333 %   Cost: 0.0308976859196564"
## [1] "Test errors: 2.56 %"
## [1] "Training errors: 0.831666666666667 %   Cost: 0.027235024005367"
## [1] "Test errors: 2.56 %"
## [1] "Training errors: 0.748333333333333 %   Cost: 0.0267236211636909"
## [1] "Test errors: 2.57 %"
## [1] "Training errors: 0.628333333333333 %   Cost: 0.0221809903669898"
## [1] "Test errors: 2.51 %"
## [1] "Training errors: 0.555 %   Cost: 0.019486290543646"
## [1] "Test errors: 2.37 %"
## [1] "Training errors: 0.531666666666667 %   Cost: 0.0166386092422425"
## [1] "Test errors: 2.46 %"
## [1] "Training errors: 0.42 %   Cost: 0.0153564083548673"
## [1] "Test errors: 2.37 %"
## [1] "Training errors: 0.366666666666667 %   Cost: 0.0131888207831851"
## [1] "Test errors: 2.95 %"
## [1] "Training errors: 0.346666666666667 %   Cost: 0.0129024770960134"
## [1] "Test errors: 2.42 %"
## [1] "Training errors: 0.226666666666667 %   Cost: 0.00929433957501798"
## [1] "Test errors: 2.5 %"
## [1] "Training errors: 0.235 %   Cost: 0.00873583946989753"
## [1] "Test errors: 2.47 %"
## [1] "Training errors: 0.135 %   Cost: 0.00674136887412882"
## [1] "Test errors: 2.43 %"
## [1] "Training errors: 0.118333333333333 %   Cost: 0.00539870829771726"
## [1] "Test errors: 2.44 %"
## [1] "Training errors: 0.095 %   Cost: 0.00537619681906716"
## [1] "Test errors: 2.34 %"
## [1] "Training errors: 0.108333333333333 %   Cost: 0.00509664974329139"
## [1] "Test errors: 2.53 %"
## [1] "Training errors: 0.085 %   Cost: 0.00438856422767854"
## [1] "Test errors: 2.38 %"
## [1] "Training errors: 0.0666666666666667 %   Cost: 0.00303881232290465"
## [1] "Test errors: 2.3 %"
## [1] "Training errors: 0.035 %   Cost: 0.00232770479971732"
## [1] "Test errors: 2.28 %"
## [1] "Training errors: 0.0166666666666667 %   Cost: 0.00158158874248727"
## [1] "Test errors: 2.23 %"
## [1] "Training errors: 0.0116666666666667 %   Cost: 0.00159907899944696"
## [1] "Test errors: 2.25 %"
## [1] "Training errors: 0.00833333333333333 %   Cost: 0.00135503307080787"
## [1] "Test errors: 2.25 %"
## [1] "Training errors: 0.00166666666666667 %   Cost: 0.000914594856064568"
## [1] "Test errors: 2.32 %"
## [1] "Training errors: 0 %   Cost: 0.000755252280339335"
## [1] "Test errors: 2.21 %"
## [1] "Training errors: 0 %   Cost: 0.00068300118813833"
## [1] "Test errors: 2.27 %"
## [1] "Training errors: 0 %   Cost: 0.0005954504920323"
## [1] "Test errors: 2.31 %"
## [1] "Training errors: 0 %   Cost: 0.000564856904552608"
## [1] "Test errors: 2.28 %"
## [1] "Training errors: 0.00333333333333333 %   Cost: 0.000590635706017061"
## [1] "Test errors: 2.35 %"
## [1] "Training errors: 0 %   Cost: 0.000488788370505069"
## [1] "Test errors: 2.27 %"
## [1] "Training errors: 0 %   Cost: 0.000442180633687957"
## [1] "Test errors: 2.32 %"
## [1] "Training errors: 0 %   Cost: 0.000404368711124711"
## [1] "Test errors: 2.23 %"
## [1] "Training errors: 0 %   Cost: 0.000408216938877965"
## [1] "Test errors: 2.27 %"
## [1] "Training errors: 0 %   Cost: 0.000368407034586523"
## [1] "Test errors: 2.26 %"
## [1] "Training errors: 0 %   Cost: 0.000365995031228206"
## [1] "Test errors: 2.25 %"
## [1] "Training errors: 0 %   Cost: 0.000354563151762167"
## [1] "Test errors: 2.33 %"
## [1] "Training errors: 0 %   Cost: 0.000350855379470883"
## [1] "Test errors: 2.3 %"
## [1] "Training errors: 0 %   Cost: 0.000332827353510768"
## [1] "Test errors: 2.3 %"
## [1] "Training errors: 0 %   Cost: 0.000306107256422866"
## [1] "Test errors: 2.31 %"
## [1] "Training errors: 0 %   Cost: 0.000290711250353721"
## [1] "Test errors: 2.29 %"
## [1] "Training errors: 0 %   Cost: 0.000296386673943092"
## [1] "Test errors: 2.3 %"
## [1] "Training errors: 0 %   Cost: 0.000278161220210979"
## [1] "Test errors: 2.32 %"
## [1] "Training errors: 0 %   Cost: 0.000256925565571378"
## [1] "Test errors: 2.28 %"
## [1] "Training errors: 0 %   Cost: 0.000275143190910884"
## [1] "Test errors: 2.34 %"
## [1] "Training errors: 0 %   Cost: 0.000243203320103584"
## [1] "Test errors: 2.29 %"
## [1] "Training errors: 0 %   Cost: 0.000237831509158986"
## [1] "Test errors: 2.34 %"
## [1] "Training errors: 0 %   Cost: 0.000218201049171018"
## [1] "Test errors: 2.29 %"
## [1] "Training errors: 0 %   Cost: 0.000210663301472316"
## [1] "Test errors: 2.31 %"
## [1] "Training errors: 0 %   Cost: 0.000206464000123127"
## [1] "Test errors: 2.26 %"
## [1] "Training errors: 0 %   Cost: 0.000196342397233113"
## [1] "Test errors: 2.28 %"
## [1] "Training errors: 0 %   Cost: 0.00020623196946417"
## [1] "Test errors: 2.27 %"
## [1] "Training errors: 0 %   Cost: 0.000188660247365854"
## [1] "Test errors: 2.28 %"
## [1] "Training errors: 0 %   Cost: 0.000191793277825873"
## [1] "Test errors: 2.28 %"
## [1] "Training errors: 0 %   Cost: 0.000189558336729317"
## [1] "Test errors: 2.32 %"
## [1] "Training errors: 0 %   Cost: 0.00017481584691016"
## [1] "Test errors: 2.3 %"
## [1] "Training errors: 0 %   Cost: 0.000178760145474712"
## [1] "Test errors: 2.27 %"
## [1] "Training errors: 0 %   Cost: 0.000168204238444165"
## [1] "Test errors: 2.29 %"
## [1] "Training errors: 0 %   Cost: 0.000171667605590897"
## [1] "Test errors: 2.29 %"
## [1] "Training errors: 0 %   Cost: 0.000156416379828935"
## [1] "Test errors: 2.31 %"
## [1] "Training errors: 0 %   Cost: 0.000156071252683869"
## [1] "Test errors: 2.32 %"
## [1] "Training errors: 0 %   Cost: 0.000158757522307593"
## [1] "Test errors: 2.28 %"
## [1] "Training errors: 0 %   Cost: 0.000151485716328826"
## [1] "Test errors: 2.29 %"
## [1] "Training errors: 0 %   Cost: 0.000146344031453487"
## [1] "Test errors: 2.3 %"
## [1] "Training errors: 0 %   Cost: 0.000152266575372985"
## [1] "Test errors: 2.32 %"
## [1] "Training errors: 0 %   Cost: 0.000150441539663955"
## [1] "Test errors: 2.24 %"
## [1] "Training errors: 0 %   Cost: 0.000142149583874049"
## [1] "Test errors: 2.31 %"
## [1] "Training errors: 0 %   Cost: 0.000135223886813932"
## [1] "Test errors: 2.28 %"
## [1] "Training errors: 0 %   Cost: 0.00013320069584191"
## [1] "Test errors: 2.28 %"
## [1] "Training errors: 0 %   Cost: 0.000133817149656269"
## [1] "Test errors: 2.31 %"
## [1] "Training errors: 0 %   Cost: 0.000127678150287801"
## [1] "Test errors: 2.3 %"
## [1] "Training errors: 0 %   Cost: 0.000124956078543356"
## [1] "Test errors: 2.29 %"
## [1] "Training errors: 0 %   Cost: 0.000124002861507011"
## [1] "Test errors: 2.29 %"
## [1] "Training errors: 0 %   Cost: 0.000123959562833895"
## [1] "Test errors: 2.33 %"
## [1] "Training errors: 0 %   Cost: 0.000121074710173936"
## [1] "Test errors: 2.31 %"
## [1] "Training errors: 0 %   Cost: 0.000116270276002672"
## [1] "Test errors: 2.3 %"
## [1] "Training errors: 0 %   Cost: 0.00011675828442296"
## [1] "Test errors: 2.3 %"
## [1] "Training errors: 0 %   Cost: 0.000113310043878216"
## [1] "Test errors: 2.3 %"
## [1] "Training errors: 0 %   Cost: 0.0001115298413774"
## [1] "Test errors: 2.27 %"
## [1] "Training errors: 0 %   Cost: 0.000107525961902307"
## [1] "Test errors: 2.28 %"
## [1] "Training errors: 0 %   Cost: 0.000103859345479702"
## [1] "Test errors: 2.31 %"
## [1] "Training errors: 0 %   Cost: 0.000105615887908058"
## [1] "Test errors: 2.31 %"
## [1] "Training errors: 0 %   Cost: 0.000105033881654928"
## [1] "Test errors: 2.33 %"
## [1] "Training errors: 0 %   Cost: 0.000100700995238772"
## [1] "Test errors: 2.28 %"
## [1] "Training errors: 0 %   Cost: 0.000102935913941069"
## [1] "Test errors: 2.31 %"
## [1] "Training errors: 0 %   Cost: 0.000101516875732526"
## [1] "Test errors: 2.29 %"
## [1] "Training errors: 0 %   Cost: 0.000102977961552913"
## [1] "Test errors: 2.32 %"
## [1] "Training errors: 0 %   Cost: 9.7229353225549e-05"
## [1] "Test errors: 2.27 %"
## [1] "Training errors: 0 %   Cost: 9.65366327625232e-05"
## [1] "Test errors: 2.3 %"
## [1] "Training errors: 0 %   Cost: 9.16251397715346e-05"
## [1] "Test errors: 2.31 %"
## [1] "Training errors: 0 %   Cost: 9.35781771168592e-05"
## [1] "Test errors: 2.3 %"
```