AE-MNIST.R

evegas

2022-03-15

```
# Autoenconder  (AE) digits from MNIST dataset

library(keras)


#### Data
mnist <- dataset_mnist()
```

```
## Loaded Tensorflow version 2.7.0
```

```
x_train <- mnist$train$x
y_train <- mnist$train$y
x_test <- mnist$test$x
y_test <- mnist$test$y

# reshape
dim(x_train) <- c(nrow(x_train), 784)
dim(x_test) <- c(nrow(x_test), 784)

######################## Selection several digits

cifra <- c(0:9)

x_train_cifra<-x_train[which(y_train %in% cifra),]
y_train_cifra<-y_train[which(y_train %in% cifra)]

x_test_cifra<-x_test[which(y_test %in% cifra),]
y_test_cifra<-y_test[which(y_test %in% cifra)]


dim(x_train_cifra)
```

```
## [1] 60000   784
```

```
length(y_train_cifra)
```

```
## [1] 60000
```

```
dim(x_test_cifra)
```

```
## [1] 10000   784
```

```
length(y_test_cifra)
```

```
## [1] 10000
```

```
sort(unique(y_train_cifra))
```

```
##  [1] 0 1 2 3 4 5 6 7 8 9
```

```
sort(unique(y_test_cifra))
```

```
##  [1] 0 1 2 3 4 5 6 7 8 9
```

```
######################### rescale

x_train_cifra <- x_train_cifra / 255
x_test_cifra <- x_test_cifra / 255
#y_train_cifra <- to_categorical(y_train_cifra, 10)
#y_test_cifra <- to_categorical(y_test_cifra, 10)


############################### Autoencoder

#### Encoder

model_enc <- keras_model_sequential()
model_enc %>%
  layer_dense(units = 128, activation = "relu", input_shape =  ncol(x_train)) %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = 32, activation = "relu")
summary(model_enc)
```

```
## Model: "sequential"
## _________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## =================================================================================
##  dense_2 (Dense)                    (None, 128)                     100480
##
##  dense_1 (Dense)                    (None, 64)                      8256
##
##  dense (Dense)                      (None, 32)                      2080
##
## =================================================================================
## Total params: 110,816
## Trainable params: 110,816
## Non-trainable params: 0
## _________________________________________________________________________________
```

```
#### Decoder

model_dec <- keras_model_sequential()
model_dec %>%
  layer_dense(units = 64, activation = "relu", input_shape =  c(32)) %>%
  layer_dense(units = 128, activation = "relu") %>%
  layer_dense(units = ncol(x_train), activation = "relu")
summary(model_dec)
```

```
## Model: "sequential_1"
## _________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## =================================================================================
```

```
##  dense_5 (Dense)                    (None, 64)                    2112
##
##  dense_4 (Dense)                    (None, 128)                   8320
##
##  dense_3 (Dense)                    (None, 784)                   101136
##
## ================================================================================
## Total params: 111,568
## Trainable params: 111,568
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
#### Autoencoder

model<-keras_model_sequential()
model %>%model_enc%>%model_dec
```

```
## Model: "sequential_2"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                  Param #
## ================================================================================
##  sequential (Sequential)            (None, 32)                    110816
##
##  sequential_1 (Sequential)          (None, 784)                   111568
##
## ================================================================================
## Total params: 222,384
## Trainable params: 222,384
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
############################################################

summary(model)
```

```
## Model: "sequential_2"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                  Param #
## ================================================================================
##  sequential (Sequential)            (None, 32)                    110816
##
##  sequential_1 (Sequential)          (None, 784)                   111568
##
## ================================================================================
## Total params: 222,384
## Trainable params: 222,384
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
######################################## Training


model %>% compile(
  loss = "mean_squared_error",
  #optimizer = optimizer_rmsprop(),
  optimizer = "adam",
```

```
  metrics = c("mean_squared_error")
)

history <- model %>% fit(
  x= x_train_cifra, y = x_train_cifra,   # Autoencoder
  epochs = 15, batch_size = 128,
  validation_split = 0.2
)

######## Prediction

# Autoencoder
output_cifra<-predict(model,x_test_cifra)
dim(output_cifra)
```

```
## [1] 10000   784
```

```
# From input to encoder
enc_output_cifra<-predict(model_enc,x_test_cifra)
dim(enc_output_cifra)
```

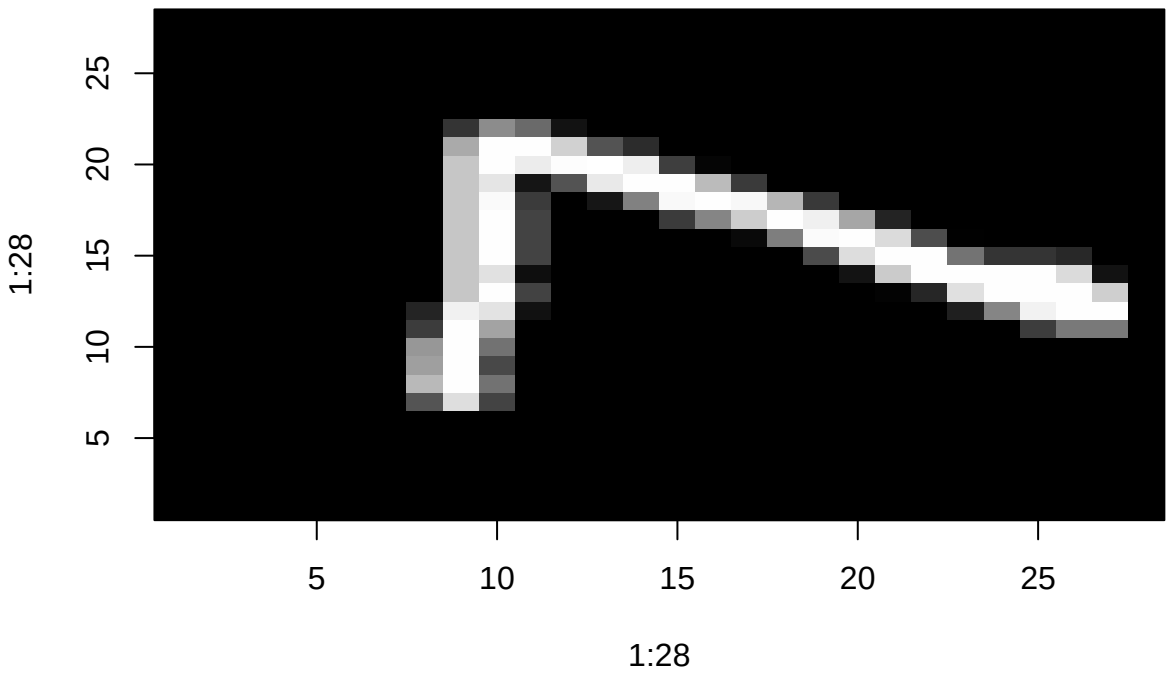
```
## [1] 10000    32
```

```
# From encoder to decoder
dec_output_cifra<-predict(model_dec,enc_output_cifra)
dim(dec_output_cifra)
```

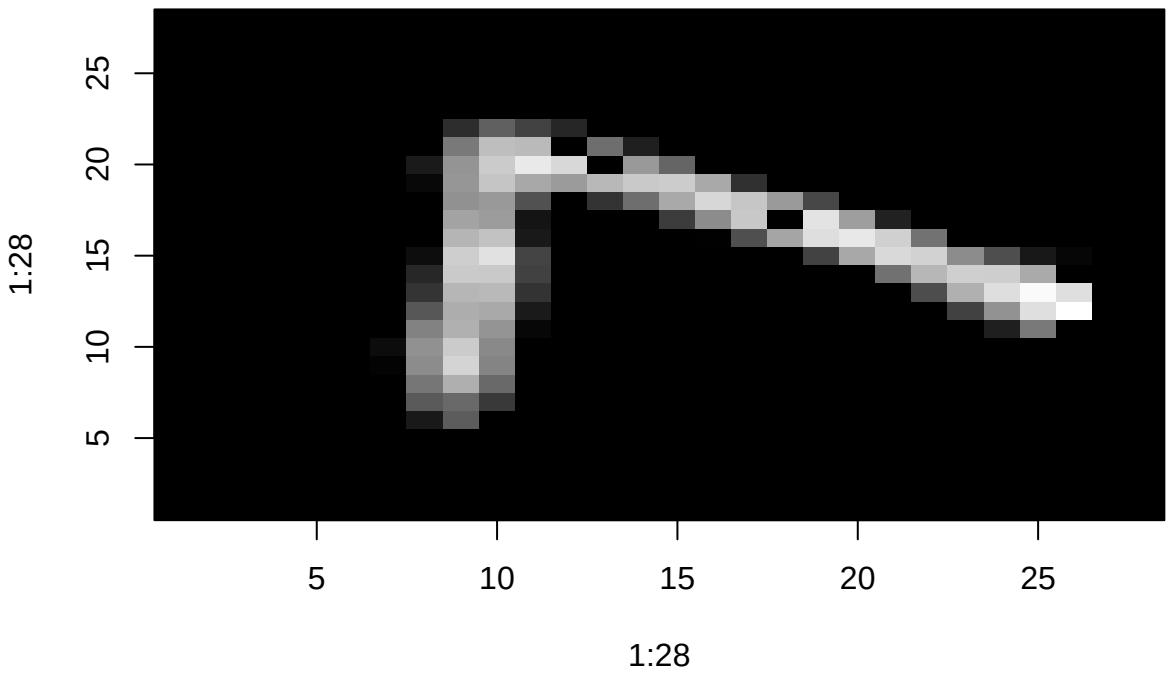
```
## [1] 10000   784
```

```
# Check

idx<-1
#x_test_cifra[idx,]
im<-matrix(x_test_cifra[idx,], nrow=28, ncol=28)
image(1:28, 1:28, im, col=gray((0:255)/255))
```



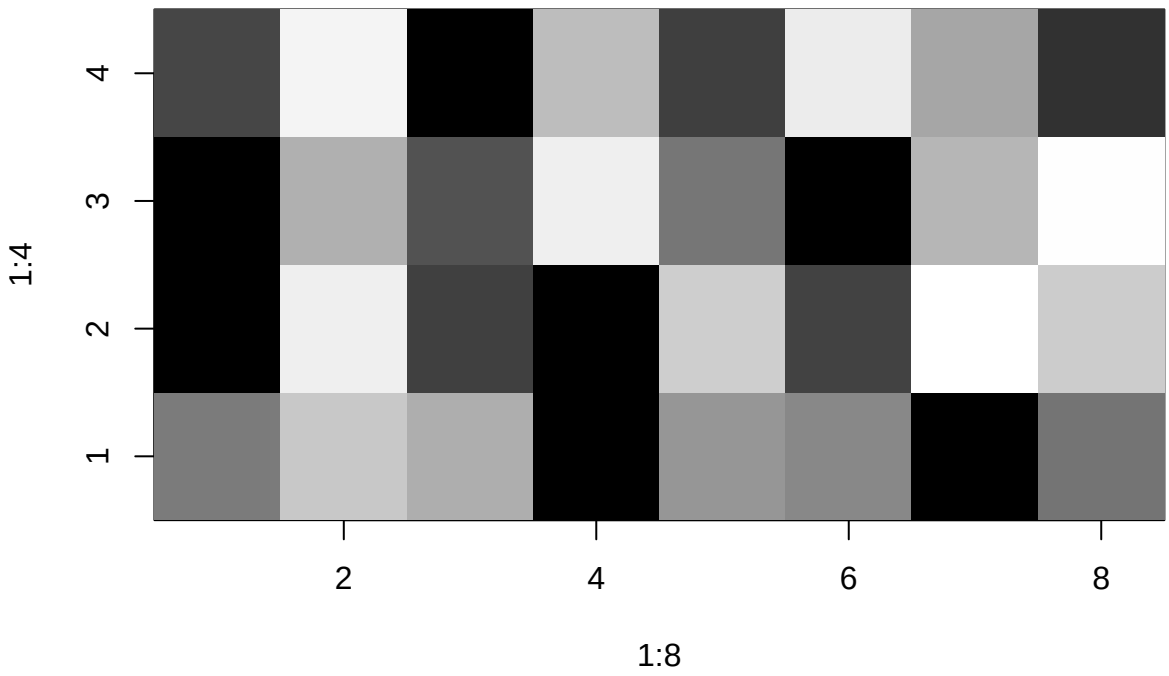

```
#output_cifra[idx,]
im<-matrix(output_cifra[idx,], nrow=28, ncol=28)
image(1:28, 1:28, im, col=gray((0:255)/255))
```




```
#dec_output_cifra[idx,]
im<-matrix(dec_output_cifra[idx,], nrow=28, ncol=28)
image(1:28, 1:28, im, col=gray((0:255)/255))

#which.max(enc_output_cifra[idx,])
#which.min(enc_output_cifra[idx,])
#which(enc_output_cifra[idx,]==cifra)

# Encoder results
im<-matrix(enc_output_cifra[idx,], nrow=8, ncol=4)
image(1:8, 1:4, im, col=gray((0:255)/255))
```




```
#############################################################


#dim(x_train_cifra)
#dim(x_test_cifra)

# Save encode digit in a Rdata file

save(enc_output_cifra, y_test_cifra, file=paste0("Encod_", paste(cifra, collapse = ""), ".RData"))
```