Chapter 7 Code

Xi Liang

5/26/2017

Example 1. Modeling the strength of concrete with ANNs

Exploring and preparing the data

```
concrete <- read.csv('data/concrete.csv')
```

```
str(concrete)
```

```
## 'data.frame':    1030 obs. of  9 variables:
##  $ cement      : num  540 540 332 332 199 ...
##  $ slag        : num  0 0 142 142 132 ...
##  $ ash         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ water       : num  162 162 228 228 192 228 228 228 228 228 ...
##  $ superplastic: num  2.5 2.5 0 0 0 0 0 0 0 0 ...
##  $ coarseagg   : num  1040 1055 932 932 978 ...
##  $ fineagg     : num  676 676 594 594 826 ...
##  $ age         : int  28 28 270 365 360 90 365 28 28 28 ...
##  $ strength    : num  80 61.9 40.3 41 44.3 ...
```

Normalizing the data.

```
normalize <- function(x) {
  return ((x-min(x)) / (max(x) - min(x)))
}
```

```
concrete_norm <- as.data.frame(lapply(concrete, normalize))
```

```
summary(concrete_norm$strength)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.2664  0.4001  0.4172  0.5457  1.0000
```

training a model on the data

```
concrete_train <- concrete_norm[1:773, ]
concrete_test <- concrete_norm[774:1030, ]
```

```
library(neuralnet)
```

```
set.seed(10)
concrete_model <- neuralnet(strength ~ cement + slag + ash + water + superplastic + coarseagg + fineagg
```

```
plot(concrete_model)
```

evaluating model performance

```
model_results <- compute(concrete_model, concrete_test[1:8])
predicted_strength <- model_results$net.result
```

```
cor(predicted_strength, concrete_test$strength)
```

```
##              [,1]
## [1,] 0.7218542561
```

improving model performance

```
set.seed(10)
concrete_model2 <- neuralnet(strength ~ cement + slag + ash + water + superplastic + coarseagg + fineag
                             data = concrete_train, hidden = 5)
```

```
plot(concrete_model2)
```

```
model_results2 <- compute(concrete_model2, concrete_test[1:8])
predicted_strength2 <- model_results2$net.result
cor(predicted_strength2, concrete_test$strength)
```

```
##              [,1]
## [1,] 0.8111905868
```

Example 2. Peforming OCR with SVMs

Exploring and preparing the data

```
letters <- read.csv("data/letterdata.csv")
```

```
str(letters)
```

```
## 'data.frame':    20000 obs. of  17 variables:
##  $ letter: Factor w/ 26 levels "A","B","C","D",..: 20 9 4 14 7 19 2 1 10 13 ...
##  $ xbox  : int  2 5 4 7 2 4 4 1 2 11 ...
##  $ ybox  : int  8 12 11 11 1 11 2 1 2 15 ...
##  $ width : int  3 3 6 6 3 5 5 3 4 13 ...
##  $ height: int  5 7 8 6 1 8 4 2 4 9 ...
##  $ onpix : int  1 2 6 3 1 3 4 1 2 7 ...
##  $ xbar  : int  8 10 10 5 8 8 8 8 10 13 ...
##  $ ybar  : int  13 5 6 9 6 8 7 2 6 2 ...
##  $ x2bar : int  0 5 2 4 6 6 6 2 2 6 ...
##  $ y2bar : int  6 4 6 6 6 9 6 2 6 2 ...
##  $ xybar : int  6 13 10 4 6 5 7 8 12 12 ...
##  $ x2ybar: int  10 3 3 4 5 6 6 2 4 1 ...
##  $ xy2bar: int  8 9 7 10 9 6 6 8 8 9 ...
##  $ xedge : int  0 2 3 6 1 0 2 1 1 8 ...
##  $ xedgey: int  8 8 7 10 7 8 8 6 6 1 ...
##  $ yedge : int  0 4 3 2 5 9 7 2 1 1 ...
##  $ yedgex: int  8 10 9 8 10 7 10 7 7 8 ...
```

Training a model on the data

```r
library(kernlab)
```

```r
letters_train <- letters[1:16000, ]
letters_test <- letters[16001:2000, ]
```

```r
letter_classifier <- ksvm(letter ~., data = letters_train,
                          kernal = "vanilladot")
```

```r
letter_classifier
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.0460515263229448
##
## Number of Support Vectors : 8705
##
## Objective Function Value : -43.7045 -34.2373 -59.9812 -27.5964 -35.1291 -47.6352 -68.0306 -39.7344 -
## Training error : 0.053375
```

Evaluating model performance

```r
letter_predictions <- predict(letter_classifier, letters_test)
head(letter_predictions)
```

```
## [1] U C E G R L
## Levels: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

```r
table(letter_predictions, letters_test$letter)
```

```
##
## letter_predictions   A   B   C   D   E   F   G   H   I   J   K   L   M   N
##                  A 545   1   0   0   0   0   0   0   1   2   0   0   1   0
##                  B   0 522   0   7   6   2   1  11   1   0   1   0  10   2
##                  C   1   0 483   0   1   0   1   0   1   0   3   2   0   0
##                  D   1   5   0 521   0   2  10  18   3   3   5   0   0   5
##                  E   0   3  11   0 506   1   2   0   0   1   1  14   0   0
##                  F   0   0   0   0   1 531   1   0   6   1   0   0   0   0
##                  G   0   0   8   0  19   0 500   7   0   0   0  11   5   0
##                  H   0   2   1   8   0   3   0 410   0   2   5   1   4   3
##                  I   0   0   0   0   0   1   0   0 474   9   0   0   0   0
##                  J   0   0   0   0   0   0   0   0  18 474   0   0   0   0
##                  K   1   0   1   0   1   0   1   8   0   0 468   1   0   0
##                  L   0   0   0   0   0   0   1   1   0   0   0 499   0   0
##                  M   4   0   1   0   0   0   0   0   0   0   1   0 532   2
##                  N   0   0   0   3   0   1   0   1   0   2   0   0   2 508
##                  O   0   0   8   3   0   1   7   5   0   2   0   1   1  10
##                  P   0   0   0   0   0   3   0   1   1   0   0   0   0   0
##                  Q   0   0   0   0   2   0   4   3   0   0   0   1   0   0
```

```
##                  R   0   7   1   5   3   0   3  26   0   0  22   3   0   7
##                  S   0   2   0   0   1   1   0   0   2   6   0   4   0   0
##                  T   0   0   0   0   0   9   0   0   0   0   0   0   0   0
##                  U   0   0   0   2   0   0   0   3   0   0   1   0   0   0
##                  V   0   0   0   0   0   0   1   0   0   0   0   0   0   0
##                  W   0   0   0   0   0   0   2   0   0   0   1   0   2   1
##                  X   0   1   0   0   1   1   0   0   4   2   7   4   0   0
##                  Y   1   0   0   0   0   0   0   0   0   0   0   0   0   0
##                  Z   0   0   0   0   2   0   0   0   1   1   0   0   0   0
##
## letter_predictions   O   P   Q   R   S   T   U   V   W   X   Y   Z
##                  A   0   0   0   1   1   0   1   0   0   0   1   0
##                  B   0   1   2  19   5   2   0  18   3   1   0   0
##                  C   0   0   0   0   0   0   0   0   0   0   0   0
##                  D   8   1   0   3   0   1   0   0   0   4   0   0
##                  E   0   3   3   0   3   0   0   1   0   7   0   5
##                  F   0  23   0   0   4   4   0   1   0   1   1   0
##                  G   2   6   2   2   0   2   0   1   1   0   0   0
##                  H   1   3   0   3   1   7   1   1   2   0   1   0
##                  I   0   0   0   0   0   0   0   0   0   1   0   0
##                  J   0   0   0   0   0   0   0   0   0   0   0   1
##                  K   0   0   0   5   0   2   2   0   0   7   0   0
##                  L   0   0   0   0   0   0   0   0   0   0   0   0
##                  M   0   0   0   0   0   0   5   1   8   0   1   0
##                  N   0   0   0   2   0   0   0   1   0   0   0   0
##                  O 510   2   9   0   0   0   2   0   0   2   0   0
##                  P   0 505   0   0   0   0   0   0   0   0   0   0
##                  Q   6   1 527   1   1   0   0   0   0   1   0   7
##                  R   4   0   1 498   1   1   0   2   1   1   0   0
##                  S   0   0   2   0 484   0   0   0   0   0   0   5
##                  T   0   0   0   0   0 544   0   0   0   0   1   0
##                  U   2   0   0   0   0   1 559   0   3   0   1   0
##                  V   0   0   0   0   0   0   2 517   0   0   2   0
##                  W   9   0   1   0   0   0   3   6 537   0   0   0
##                  X   0   0   0   1   1   3   0   0   0 517   0   1
##                  Y   0   6   0   0   0   0   0   0   0   1 559   0
##                  Z   0   0   0   0   2   0   0   0   0   0   0 495
```

```r
agreement <- letter_predictions == letters_test$letter
prop.table(table(agreement))
```

```
## agreement
##       FALSE        TRUE
## 0.05549207256 0.94450792744
```

Improving model performance

```r
letter_classifier_rbf <- ksvm(letter ~., data = letters_train,
                              kernel = "rbfdot")
```

```r
letter_predictions_rbf <- predict(letter_classifier_rbf, letters_test)
```

```r
agreement_rbf <- letter_predictions_rbf == letters_test$letter
table(agreement_rbf)
```

```
## agreement_rbf
## FALSE  TRUE
##   750 13252
```

```r
prop.table(table(agreement_rbf))
```

```
## agreement_rbf
##        FALSE         TRUE
## 0.0535637766 0.9464362234
```