10장 기계학습 알고리즘1

jihyun

2019 8 28

4. 신경망

신경망은 입력층, 은닉층, 출력층으로 구분된다. 신경망은 예측 성능이 우수하다고 알려져 있고, 은닉층에서 입력 값이 조합되므로, 비선형적인 문제를 해결할 수 있는 특징이 있다. 그러나 의사결정나무 모델과 비교했을때 만들어진 모델을 직관적으로 이해하기가 어렵고, 수작업으로 모델을 수정하기 어렵다는 단점이 있다.

R의 nnet 패키지는 신경망 모델 작성을 위한 함수이다. nnet패키지에서 신경망의 파라미터는 엔트로피 또는 SSE(Sum of Square Error) 를 고려해 최적화 된다. 출력 결과는 softmax함수를 사용해 확률과 같은 형태로 변환 할 수 있다. 과적합을 막기 위한 방법으로 가중치 감소(weight decay)를 제공한다.

아이리스 데이터에 대한 신경망 모델을 만들어보자

```
#install.packages('nnet')
library(nnet)
m <- nnet(Species ~ ., data = iris, size = 3) #        = 3
```

```
## # weights:  27
## initial  value 245.437423
## iter  10 value 37.949098
## iter  20 value 7.335906
## iter  30 value 6.257029
## iter  40 value 5.617721
## iter  50 value 5.006940
## iter  60 value 4.925056
## iter  70 value 4.924976
## iter  70 value 4.924976
## iter  70 value 4.924976
## final  value 4.924976
## converged
```

train 과 test를 따로 분리하지 않았으므로, iris데이터 자신에 대한 예측 결과는 다음과 같다.

```
head(predict(m, newdata = iris))
```

```
##   setosa   versicolor     virginica
## 1      1 2.829245e-33 1.679241e-191
## 2      1 2.829245e-33 1.679241e-191
## 3      1 2.829245e-33 1.679241e-191
## 4      1 2.829245e-33 1.679241e-191
## 5      1 2.829245e-33 1.679241e-191
## 6      1 2.829245e-33 1.679241e-191
```

만약 예측 결과로 부터 확률값이 아니라 분류된 결과를 알고자 한다면 type 에 class 를 지정해 주면 된다.

```
head(predict(m, newdata = iris, type = 'class'))
```

```
## [1] "setosa" "setosa" "setosa" "setosa" "setosa" "setosa"
```

지금까지의 방법은 nnet()에 포뮬러를 지정한 형태였다. 포뮬러를 지정 할 경우, nnet()은 다음과 같이 동작하도록 되어있다. 1. 출력층에서 시그모이드 함수 사용 2. 분류 레벨(출력층)의 수가 2개 라면 엔트로피를 사용하여 가중치 수정, 분류레벨이 3개이상이면 SSE 를 이용하여 가중치가 수정된다.

좀더 빠른 속도를 원하거나, nnet에서 각종 파라미터가 자동으로 지정되는 것을 원하지 않는 경우 x,y 를 직접 지정하는 형태로 nnet()을 호출 할 수 있다.

그러기 위해서는 가장 먼저 y 를 지시 행렬(indicator matrix)로 변경해해야 한다. 지시행렬로 변경하는 방법은 class.ind를 사용하면 된다.

```
head(class.ind(iris$Species))
```

```
##      setosa versicolor virginica
## [1,]      1          0         0
## [2,]      1          0         0
## [3,]      1          0         0
## [4,]      1          0         0
## [5,]      1          0         0
## [6,]      1          0         0
```

지시 행렬을 이용하여 다음과 같이 모델을 만들 수 있다.

```
m2 <- nnet(iris[,1:4], class.ind(iris$Species), size = 3, softmax = TRUE)
```

```
## # weights:  27
## initial  value 171.750905
## iter  10 value 69.314723
## final  value 69.314718
## converged
```

m2 모델의 예측결과는 다음과 같다.

```
head(predict(m2, newdata = iris[,1:4], type = "class" ))
```

```
## [1] "setosa" "setosa" "setosa" "setosa" "setosa" "setosa"
```

5. 서포트 벡터 머신.(SVM)

서포트 벡터 머신은 (SVM) 서로 다른 분류에 속한 데이터 간에 간격이 최대가 되는 선(또는 평면)을 찾아 이를 기준으로 데이터를 분류하는 모델이다.

서포트 벡터 머신에는 커널함수가 있다. 커널 함수는, 데이터의 분포가 직선으로 나눌 수 없는 형태일 때(곡선으로만 나눠지는 경우) 데이터의 차원을 적절한 고차원으로 옮긴 뒤, 변환된 차원에서 서포트 벡터 머신을 이용해 초평면을 찾는다.

서포트 벡터 머신 학습을 위한 패키지는 e1071, kernlab 등이 있다.

아이리스 데이터에 포뮬러를 사용하여 svm 모델을 만들어 보자.

```
#install.packages("kernlab")
library(kernlab)
(m<- ksvm(Species~., data=iris))
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.448371390571469
##
## Number of Support Vectors : 56
##
## Objective Function Value : -3.6253 -3.6918 -22.0222
## Training error : 0.026667
```

만들어진 모델로부터의 예측에는 predict()를 사용한다.

```
head(predict(m, newdata = iris))
```

```
## [1] setosa setosa setosa setosa setosa setosa
## Levels: setosa versicolor virginica
```

ksvm()은 가우시안 커널을 기본으로 사용한다. 만약 커널함수를 바꾸고 싶다면 kernel파라미터에 원하는 함수를 지정하면 된다.

e1071패키지는 tune()함수를 사용해 모델을 튜닝할 수 있다. (커널함수의 최적의 상수값을 찾아준다)

```
#install.packages("e1071")
library(e1071)
tune(svm, Species~., data = iris, gamma = 2^(-1:1), cost = 2^(2:4))
```

```
##
## Error estimation of 'svm' using 10-fold cross validation: 0.04666667
```

6. 클래스 불균형 해소

구분할 각 분류에 해당하는 데이터의 비율이 동일하지 않은 경우, 훈련 데이터 내 비율이 높은 분류 쪽으로 결과를 내놓는 모델이 될 수 있다. 이런 상황을 클래스 불균형이라고 한다.

예를 들어 다음의 유방암 데이터를 살펴보자.

```
#install.packages('mlbench')
library(mlbench)
data(BreastCancer)
table(BreastCancer$Class)
```

```
##
##    benign malignant
##       458       241
```

유방암 데이터의 양성(benign) 수는 458, 악성(maligant) 수는 241이다. 이런 경우 모델이 maligant의 특징을 학습하지 않을 확률이 높다. 클래스 불균형을 해소하는 한가지 방법은 관측치가 적은 쪽에 더 큰 가중치를 두거나, 데이터가 적은 쪽을 잘못 분류했을때 더 많은 비용을 들게 하는 방법이다. 보통 모델링 함수의 param, loss, cost 등의 파라미터에 이런 값을 지정할 수 있다.

또 다른 방법은 모델의 훈련데이터를 직접 조절하는 방법이다. 이에는 업샘플링, 다운샘플링, SMOTE가 있다.

6-1 업샘플링, 다운샘플링

업샘플링은 데이터가 적은 표본으로 더 많이 추출하는 방법이며(데이터가 큰 쪽에 수를 맞춘다.), 다운 샘플링은 데이터가 많은 쪽을 적게 추출하는 방법이다(데이터가 작은 쪽으로 수를 맞춘다).

caret패키지의 upSample(), downSample() 이 이러한 표본 추출 방법을 지원한다.

다음은 BreastCancer 데이터를 대상으로 upSample()을 수행한 예다.

```
#install.packages('caret')
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:kernlab':
##
##     alpha
```

```
x <- upSample(subset(BreastCancer, select = -Class), BreastCancer$Class)
table(x$Class)
```

```
##
##    benign malignant
##       458       458
```

이전에는 458 : 241 이던 비율이 458 : 458 로 맞춰진 것을 알 수 있다.

다음은 upSampling()을 훈련데이터에 적용한 경우와 그렇지 않은 경우 각각 의사 결정 나무의 성능을 비교한 코드이다

```
library(party)
```

```
## Loading required package: grid
```

```
## Loading required package: mvtnorm
```

```
## Loading required package: modeltools
```

```
## Loading required package: stats4
```

```
##
## Attaching package: 'modeltools'

## The following object is masked from 'package:kernlab':
##
##     prior

## Loading required package: strucchange

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich
```

```
library(rpart)
data <- subset(BreastCancer, select = -Id)
parts <- createDataPartition(data$Class, p = .8)
data.train <- data[parts$Resample1, ]
data.test <- data[-parts$Resample1, ]
m <- rpart(Class ~., data = data.train)
confusionMatrix(data.test$Class,
                predict(m, newdata = data.test, type = "class"))
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign        90         1
##   malignant      4        44
##
##                Accuracy : 0.964
##                  95% CI : (0.9181, 0.9882)
##     No Information Rate : 0.6763
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9193
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.9574
##             Specificity : 0.9778
##          Pos Pred Value : 0.9890
##          Neg Pred Value : 0.9167
##              Prevalence : 0.6763
##          Detection Rate : 0.6475
##    Detection Prevalence : 0.6547
```

```
##       Balanced Accuracy : 0.9676
##
##        'Positive' Class : benign
##
```

upSampling()을 적용하지 않았을 때 정확도가 0.9209로 나왔다.

```
#   train data  upSample         data.up.train    .
data.up.train <- upSample(subset(data.train, select = -Class),
                          data.train$Class)

m<-rpart(Class~., data = data.up.train)
confusionMatrix(data.test$Class,
                predict(m, newdata = data.test, type = "class"))
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign        87         4
##   malignant      2        46
##
##                Accuracy : 0.9568
##                  95% CI : (0.9084, 0.984)
##     No Information Rate : 0.6403
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9055
##
##  Mcnemar's Test P-Value : 0.6831
##
##             Sensitivity : 0.9775
##             Specificity : 0.9200
##          Pos Pred Value : 0.9560
##          Neg Pred Value : 0.9583
##              Prevalence : 0.6403
##          Detection Rate : 0.6259
##    Detection Prevalence : 0.6547
##       Balanced Accuracy : 0.9488
##
##        'Positive' Class : benign
##
```

upSampling을 적용했을때는 정확도가 0.8993이 나왔다. 정확도는 감소했지만 Sensitivity는 증가했다.

6-2 SMOTE

SMOTE는 비율이 낮은 분류의 데이터를 만들어 내는 방법이다. SMOTE는 먼저 분류 개수가 적은 쪽의 데이터 샘플을 취한디 이 샘플에 knn의 적용하여 비슷 한 특징을 가진 샘플들을 만들어 낸다.

iris 데이터에 SMOTE를 적용해 보자. 종류의 비율을 다르게 하기 위해 Setosa는 rare로 , 다른것은 common으로 class 를 만들었다.

```
data(iris)
data <- iris[,c(1,2,5)]
data$Species <- factor(ifelse(data$Species == "setosa", "rare", "common"))
table(data$Species)
```

```
##
## common   rare
##    100     50
```

이 데이터에 SMOTE를 적용하여, common 과 rare를 맞춘 결과는 다음과 같다.

```
#install.packages("DMwR")
library(DMwR)
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
newData <- SMOTE(Species ~ ., data, per.over = 600, perc.under = 100)
table(newData$Species)
```

```
##
## common   rare
##    100    150
```

```
#   ???
```