1장. 신경망 "neuralnet"

인공신경망(artificial neural network; 신경망)은 뇌의 구조와 기능을 모형화한 것으로, 입력 층과 은닉 층(hidden layer), 출력 층의 뉴런들, 그리고 이들을 잇는 연결선(synapse)으로 구성된다. 신경망은 지도학습(supervised learning)의 한 방법으로 역전파 알고리즘(back-propagation algorithm) 또는 이것의 변형으로 적합되어 예측분석(predictive analytics)에 사용된다. 이 장에서는 R 팩키지 "neuralnet"을 활용하여 신경망 분석을 할 것이다.

1. 인공신경망

인공신경망(신경망, neural network)은 입력 값 $x_1, \cdots, x_p$로부터 출력 값 y에 이르는 과정을 뇌(腦, brain)의 구조와 기능으로부터 유추한다. 즉 입력 값 변수(→ 입력 노드)와 출력 값 변수(→ 출력 노드) 사이에 그림 1과 같이 몇 개의 노드를 중개자로 넣어 활용한다. 앞으로 다음 몇 개의 용어를 자주 쓰게 될 것이다.

- 입력 층(input layer): 입력 노드들 $x_1, \cdots, x_p$의 통칭
- 출력 층(output layer): 출력 노드(들) y의 통칭
- 은닉 층(hidden layer): 입력 층과 출력 층 사이에 낀 "은닉" 노드들의 통칭.

그림 1의 예에서는 은닉 층이 1개이지만 은닉 층이 2개 이상일 수도 있다. 소위 딥러닝(deep learning)은 은닉 층의 수가 "많은" (2개 이상인) 신경망이다.

일단 은닉 층이 1개이고 그 내에 q개의 노드(변수)가 있다고 하자. 그리고 이들을 $z_1, \cdots, z_q$로 표기하자. 은닉 변수들은 실제 관측되는 변수들이 아닌 "잠재변수(潛在變數, latent variable)"이다. 각 은닉 변수 z_k는 입력 변수 $x_1, \cdots, x_p$와 다음 관계에 있다고 상정한다 $(k = 1, \cdots, q)$.

$$z_k = f_1\left(w_{0k} + \sum_{j=1}^{p} w_{jk} x_j\right), \tag{1}$$

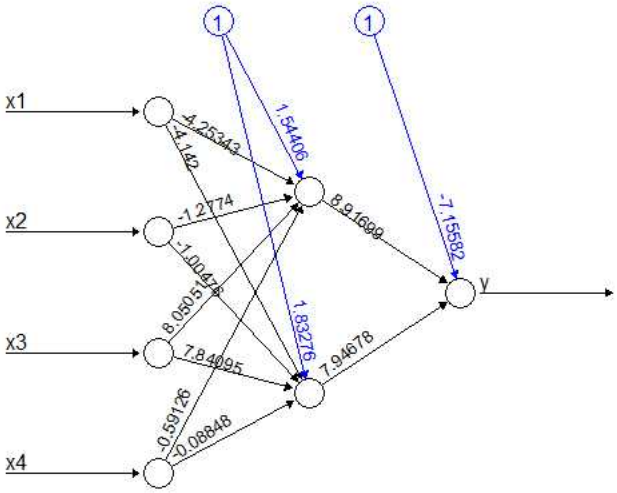


그림 1. 신경망 모형의 보기

여기서 w_{jk}는 (입력)노드 j에서 (은닉)노드 k로 가는 연결 줄 (시냅스 synapse)에 붙은 계수이고 w_{0k}는 노드 k에 영향을 주는 상수항이고[1] $f_1(u)$는 활성화 함수(activation function)라고 하는 선형예측식 u_k의 한 변환이다.[2] 활성화 함수의 예는

- 로지스틱(logistic) 함수: $f_1(u) = \dfrac{e^u}{1+e^u}, \ -\infty < u < \infty.$
- hyperbolic tangent 함수: $f_1(u) = \dfrac{e^u - e^{-u}}{e^u + e^{-u}}, \ -\infty < u < \infty.$
- 무변환: $f_1(u) = u, \ -\infty < u < \infty$

등이다. 다시 말하여 입력 변수 $x_1, \cdots, x_p$의 선형결합과 상수의 합 u_k에 로지스틱과 같은 단조변환을 취하여 잠재변수 z_k가 결정된다고 보는 것이다. 활성화 함수 $f_1(u_k)$는 투입값 u_k의 양쪽 끝 부분을 완화시키는 역할을 한다.

---

1) 이것을 신경망에서는 ‘bias’라고 하고 그림 1에서는 ①에서 은닉노드 k로 가는 시냅스에 붙은 계수이다.

2) $u_k = w_{0k} + \sum_{j=1}^{p} w_{jk} x_j, \ k = 1, \cdots, q.$

이렇게 잠재변수 $z_1, \cdots, z_q$가 정해지면 이들과 bias의 결합의 단조변환으로 출력변수 y가 결정된다고 본다.

$$y = f_2(w_{0y} + \sum_{k=1}^{q} w_{ky} z_k), \tag{2}$$

여기서 $f_2(v_y)$는 선형예측식 $v_y = w_{0y} + \sum_{k=1}^{q} w_{ky} z_k$의 활성화 변환이다. 출력변수 y가 이항형(0,1)인 경우에는 로지스틱 변환이, 연속형인 경우 무변환이 적정하다.

은닉 노드 z_k는 입력 변수 $x_1, \cdots, x_p$ 또는 출력 변수 y와 달리 직접 관측되지 않는다. 이것은 현상의 배후에 자리 잡은 추상적 개념(abstract concept)이다. 다시 말하여, 신경망은 입력 변수 $x_1, \cdots, x_p$에 대한 추상화를 거쳐 출력 변수 y로 간다고 보는 것이다. 예컨대,

x1: 국어, x2: 영어, x3: 수학, x4: 과학, y: 대학입학(0, 1)

에서

z1: “언어능력”, z2: “수리능력”

과 같은 것이다. 그러나 신경망은 각 은닉 노드에 대해 이름 짓기(naming)을 스스로 하지 않는다.

출력 변수 y의 결정 식 (1)에 오차가 없을 수 없다. y가 연속형 수치인 경우에는 오차는 관측값과 모형값 간 차이이고, y가 이항형(0,1)인 경우에는 실제 값과 모형확률 사이의 괴리가 오차인데 이것들의 총합을 다음과 같이 규정한다.[3)]

- 오차총합 SSE (sum of squared errors): 출력 변수 y가 연속형 수치인 경우

$$E = \frac{1}{2} \sum_{i=1}^{n} (y_i^{\text{obs}} - y_i^{\text{net}})^2 .$$

- 오차총합 CE (cross-entropy): 출력 변수 y가 이항형 (0,1)인 경우

$$E = -\sum_{i=1}^{n} \left\{ y_i^{\text{obs}} \log_e y_i^{\text{net}} + (1 - y_i^{\text{obs}}) \log_e (1 - y_i^{\text{net}}) \right\}.$$

이제 남은 문제는 오차총합 E가 최소화되는 시냅스 계수들을 찾아내는 것이다.

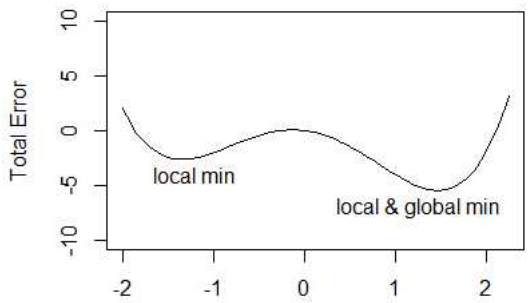
---

3) y_i^{obs}는 개체 $i(=1, \cdots, n)$의 y 관측값이고 y_i^{net}는 개체 i의 네트워크 y 산출값이다.

2. 역전파 알고리즘

신경망 모형에는 많은 수의 시냅스가 있는데 이것에 적정 “weight”를 붙이는 것은 쉽지 않은 일이다. 다음 작업이 필요하다.

- 모든 입력변수를 표준화한다. weight를 단위무관(unit-free)하게 할 필요가 있기 때문이다. 표준화의 방법은 범위 표준화와 평균-표준편차 표준화가 있다. 전자는 최소값이 0, 최대값이 1이 되도록 하는 선형변환이고, 후자는 평균이 0, 표준편차가 1이 되도록 하는 선형변환이다.
- 모든 가중치를 정규분포 임의 값으로 초기화한다. 임시로 신경망이 설정되면 입력 층에서 은닉 층을 거쳐 출력 층의 노드 값을 생성한다. 그 결과로 오차총합 E를 얻는다.
- 가중치에 의한 오차총합 E의 gradient(편미분 벡터)를 산출한다. “역(逆)전파 알고리즘”(back-propagation algorithm)은 gradient를 감소시키는 방향으로 weight들을 약간 이동시키는 방법이다. 이때 이동의 크기는 학습률(learning rate) α에 의하여 제어된다. 탄력적(resilient) 역전파 알고리즘은 학습률 α의 크기를 네트워크의 적합이 진행됨에 따라 점차 작게 조정하고 gradient의 반영 방식에 변화를 준 알고리즘으로 R의 “neuralnet” 팩키지에 구현되어 있다.
- 모든 역전파 알고리즘에 의한 해는 대역적 최저점(global minimum)이 아닌 국소적 최저점(local minimum)일 수 있다 (실망스럽게도). 이런 문제를 다소 완화하기 위하여, 다수의 초기점에서 출발하여 알고리즘에 의해 최종 도달한 국소적 최저점 중에서 오차총합이 가장 작은 해를 취하는 방법을 시도할 수 있다.



3. 신경망의 구조

이 장에서 다루는 신경망은 다층퍼셉트론모형(multi-layer perceptron model)으로 은닉 층의 수와 은닉 층 내 노드의 수가 관건이다. 출력의 연속함수로 표현되는 경우 Cybenko의 “universal approximation theorem”에 의해 은닉 층은 1개여도 된다. 그러나 이 정리가 몇 개의 노드가 있으면 되는지를 말하지는 않는다. 복잡할 것 같지 않은 사회적 현상에 대한 모형으로서의 신경망은 작은 수(7-8개 이내)의 은닉 노드로 충분히 표현할 수 있다. 은닉 노드들은 각각 추상적 개념의 표상인데 이것들을 위계적 구조화할 생각이라면 은닉 층의 수를 2개 이상으로 두어야 하고 그럴 생각이 없다면 은닉 층의 수는 1개로 둔다.

경험적으로 적정한 신경망 모형은 AIC 등 모형선택 기준을 적용하여 찾거나 독립적인 테스트 데이터에서 예측 오차를 평가하여 찾는다.

4. R 팩키지 “neuralnet”

R 팩키지 “neuralnet”은 다층 퍼셉트론 신경망을 적합하여 다수의 공변량과 반응 간의 함수적 관계를 추정해낸다. 모형적합은 역전파 알고리즘과 3종의 탄력적 역전파 알고리즘으로 할 수 있고 사용자가 활성화 함수와 오차총합을 지정할 수 있다. 세부적 설명과 사항은 Gunther and Fritsch (2010)의 R Journal 논문과 CRAN의 “neuralnet” 매뉴얼을 참조하기 바란다.

“neuralnet” 팩키지의 주요 함수는 neuralnet()이다. 용법은 다음과 같다.

```
neuralnet(formula, data, hidden = 1, rep = 1,
          algorithm = "rprop+",
          err.fct = "sse", act.fct = "logistic",
          linear.output = TRUE, likelihood = FALSE)
```

여기서 `formula`는 출력변수와 입력변수의 지정 문이고, `data`는 신경망 적합에 쓰일 데이터 프레임이다. 그리고

`hidden` 은닉 층의 수와 층별 노드 수를 벡터로 지정한다. 예컨대 `c(4,2)`는 은닉 층 1에 노드 4개, 은닉 층 2에 노드 2개가 있음을 말한다.

`rep` 모형의 적합 횟수를 말한다. 이것을 늘리면 국소적 해에서 벗어날 가능성이 커진다.

`algorithm` 역전파 알고리즘 `"backprop"` 외에 이것의 탄력적 버전인 `"rprop+"`를 쓸 수 있다 (디폴트). 그 외에 `"rprop-"` 등이 있다.

`err.fct` 오차총합의 지정. 오차 제곱 합 `"sse"`과 교차 엔트로피 `"ce"`가 대표적이다.

`act.fct` 활성화 함수의 지정. 로지스틱 `"logistic"`과 hyperbolic tangent `"tanh"` 중에서 선택할 수 있다. 출력 노드에서 이것을 막으려면 `linear.output`을 `FALSE`로 놓는다.

`likelihood` 이것을 `TRUE`로 놓으면 아카이케 정보량 기준 `AIC`와 베이즈 정보량 기준 `BIC`가 산출된다.

이 밖에 유용한 함수는 `compute()`와 `plot.nn()`이다.

`compute(net, covariate)`의 용법:

데이터셋 `covariate`의 각 개체에 대하여 신경망 `net`의 출력값을 산출한다.

`plot.nn(net, show.weights = TRUE, information=TRUE)`의 용법:

신경망 `net`을 플롯한다. `show.weights`는 시냅스 가중치의 출력을 제어하고 `information`은 오차총합과 계산 스텝 수의 출력을 제어한다.

다음 절에서 데이터 사례에 적용한 R 스크립트를 제시하니 보기를 통해 익숙해지길 바란다.

5. 활용사례

5.1 infert 데이터

이 데이터는 infertility(불임성)에 관한 연구에서 나왔다. 불임 83명(case=1)과 비(非)불임 165명(case=1)에 대하여 age, parity, induced, spontaneous 등의 공변량이 조사되었다. 따라서 case를 이항형 출력변수로, age를 포함하는 4개 공변량을 입력변수로 볼 수 있다. 자세한 변수설명은 help(infert)를 참조 바란다.

```
# binary classification
library(neuralnet)
data(infert)
X <- infert[,c("age","parity","induced","spontaneous")]
y <- infert["case"]
df <- data.frame(scale(X),y)   # 신경망모형 적합전 scaling이 필요하다.
set.seed(1)  # 신경망은 random seed에 의존한다.
nn <- neuralnet(y ~ age+parity+induced+spontaneous, data=df,
                hidden=2,           # 1개 은닉층에 2개의 뉴런
                err.fct="ce",       # cross entropy 오차 적용
                linear.output=F)    # 출력노드에서 선형 활성화 비적용
print(nn)     # threshold는 편미분 변화량에 적용되는 파라미터이다.
  1 repetition was calculated.
    Error Reached  Threshold       Steps
  1 118.9899631    0.008828907722  4943
plot(nn)
```

그림 2가 적합된 신경망 모형이다. plot.nn() 함수로 얻었다. 1개 은닉층에 2개 노드가 있는 것을 볼 수 있다. 최종 오차총합은 119.0 (=cross-entropy)이고 이 값에 이르는 데 4943회의 업데이팅이 있었다.

그림 3은 신경망의 적합 횟수를 10회로 지정한 결과이다. 최종 오차총합이 115.6으로 앞의 것보다 작다. 이것도 대역적 최소점이 아닐 수도 있지만 앞의 것은 대역적 최소점이 아닌 것이 확실하다. 물론 계산 시간은 더 걸린다.

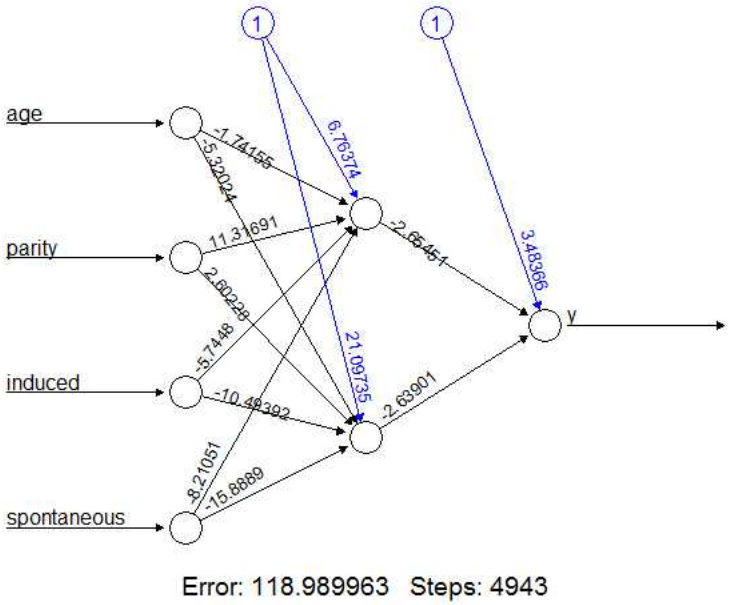


그림 2. infert 신경망: 1개 은닉층, 2개 노드가 적용되었다.

```
set.seed(1)
nn.10 <- neuralnet(y ~ age+parity+induced+spontaneous, data=df,
          hidden=2, err.fct="ce", linear.output=F,
          rep=10)      # rep 옵션 사용
print(nn.10)           # 오차총합 순서로 정렬되어 제시됨
10 repetitions were calculated.
        Error Reached Threshold Steps
6  115.6472588    0.009276981432  1527
5  115.6840964    0.009216245441  1646
2  116.5741874    0.009668204840 10914
8  117.1388501    0.009530284941  3757
1  118.9899631    0.008828907722  4943
```

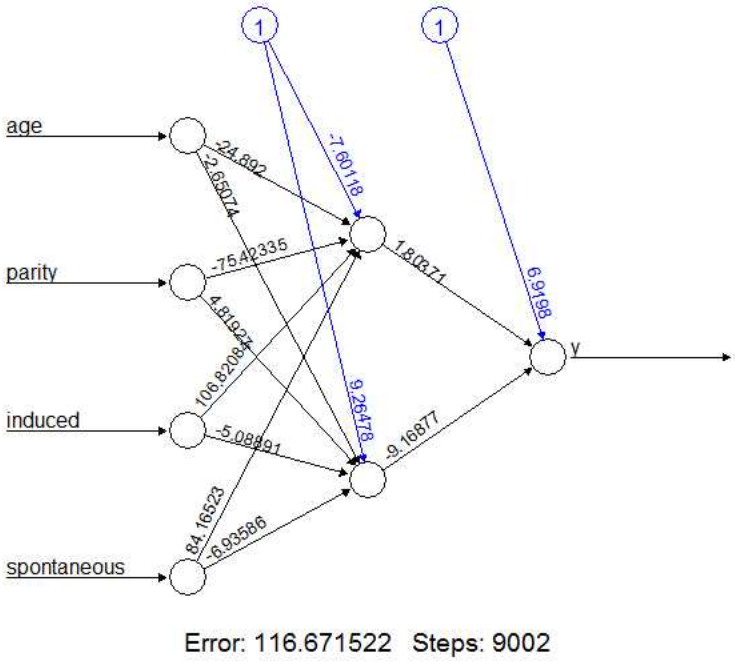


그림 3. infert 신경망: 10회 반복 적합

```
7  118.9910290    0.009069222091  3619
10 118.9988215    0.009798017162  4407
9  119.0314854    0.009669834070  4901
3  119.0530412    0.009758661881  3958
4  119.3103828    0.009899005744 28241
plot(nn.10, rep="best")              # rep 옵션을 "best"로 놓음
```

이제부터는 전체 데이터를 훈련 데이터와 테스트 데이터로 분할하여 신경망 적합은 훈련 데이터로 하고 적합 모형을 테스트 데이터에 적용하여 예측오차를 공정하게 평가하는 과정에 들어갈 것이다. 훈련 데이터와 테스트 데이터의 비율은 3:1이다.

```
n <- nrow(infert)
n.1 <- round(n*0.75)
set.seed(2)
index <- sample(1:n, n.1)
X.1 <- infert[index,c("age","parity","induced","spontaneous")]
y.1 <- infert[index,"case"]
X.2 <- infert[-index,c("age","parity","induced","spontaneous")]
y.2 <- infert[-index,"case"]
df.1 <- data.frame(scale(X.1),y.1)
mean.1 <- apply(X.1, 2, mean)
sd.1 <- apply(X.1, 2, sd)
set.seed(4)
nn.1 <- neuralnet(y.1 ~ age+parity+induced+spontaneous, data=df.1,
            hidden=2, err.fct="ce", linear.output=F)
pred.1 <- compute(nn.1, df.1[,1:4])$net.result     # 모형 적합값 산출
```

앞의 마지막 줄에서 산출된 pred.1은 훈련 데이터에서 모형 적합값으로 확률로 표현되어 있다. 이것이 훈련 데이터에서의 범주 0 비율보다 크면 범주 1로 분류하고 그렇지 않으면 범주 0으로 분류하기로 하자.[4)]

```
p.1 <- table(y.1)
cut.1 <- quantile(pred.1, prob=p.1[1]/(p.1[1]+p.1[2]+1))
tab.1 <- addmargins(table(pred.1 > cut.1, df.1$y.1))

tab.1
         0   1 Sum           # 훈련 데이터에 재적합하여 얻은 오류율은
  FALSE 101  20 121          # 41/186, 즉 22%이다.
  TRUE   21  44  65
  Sum   122  64 186
```

다음은 테스트 데이터에 적합모형을 적용하여 공정한 오류율을 산출하는 과정이다. 이를 위하여 테스트 데이터의 공변량을 훈련 데이터의 평균과 표준편차에 준

4) 다른 방법도 있다. 간단하게 이 확률이 0.5보다 크면 범주 1로 분류하고 그렇지 않으면 범주 0으로 분류하는 방법이 그렇다.

거하여 표준화할 필요가 있다.

```
df.2 <- data.frame(scale(X.2, center=mean.1, scale=sd.1), y.2)
pred.2 <- compute(nn.1, df.2[,1:4])$net.result
tab.2 <- addmargins(table(pred.2 > cut.1, df.2$y.2))
tab.2
        0  1 Sum             # 테스트 데이터에 적용하여 얻은 오류율은
  FALSE 34  8  42           # 13/62, 즉 27%
  TRUE   9 11  20
  Sum   43 19  62
```

참고로, 로지스틱 모형의 경우 테스트 오류율은 21/62, 즉 34%이다. 로지스틱 모형은 은닉 층이 없는 신경망 모형으로 볼 수 있는 바, 이 사례에서는 신경망 모형이 로지스틱 모형에 비해 우수하다고 볼 수 있다.

5.2 Boston 데이터

“MASS” 팩키지의 Boston 데이터는 보스턴 지역의 블록 단위 주택가격에 관한 자료로서 반응변수 medv (중간값 주택가격)와 13개 공변량으로 구성되어 있다. 관측 수는 506이다. 훈련 데이터 대 테스트 데이터의 비율은 3:1로 한다.

```
library(MASS)
data(Boston)
data <- Boston
index <- sample(1:nrow(data),round(0.75*nrow(data)))
train <- data[index,]
test <- data[-index,]
```

다음은 선형회귀분석 부분이다.

```
lm.fit <- glm(medv ~ ., data=train)
summary(lm.fit)
pr.lm <- predict(lm.fit,test)
MSE.lm <- sum((pr.lm - test$medv)^2)/nrow(test)
```

다음은 신경망 모형 투입이전에 해야 할 공변량 표준화 부분이다. 최소값이 0, 최대값이 1인 범위 표준화를 적용하였다.

```
maxs <- apply(train, 2, max)
mins <- apply(train, 2, min)
train_ <- as.data.frame(scale(train, center=mins,
          scale= maxs-mins))
test_ <- as.data.frame(scale(test, center=mins, scale= maxs-mins))
```

다음은 신경망 구조 설정 부분이다. 은닉 층을 2개로 하였는데 층 1에는 5개의 노드를, 층 2에는 3개의 노드를 넣었다. 모형적합은 2회 반복으로, 출력노드에서는 무변환(선형출력)을 지정하였다. 반응변수가 연속형이기 때문이다.

```
library(neuralnet)
n <- names(train_)
f <- as.formula(paste("medv ~", paste(n[!n %in% "medv"],
     collapse = " + ")))
nn <- neuralnet(f, data=train_, hidden=c(5,3), rep=2,
      linear.output=T)
print(nn)
2 repetitions were calculated.
        Error Reached Threshold Steps
1 0.3857463901    0.009849349026  1567
2 0.4518236956    0.008682433666  2556
```

다음은 테스트 데이터에서 모형 예측값을 산출하는 과정이다. 더불어 평균제곱오차 MSE를 산출하였다.

```
pr.nn <- compute(nn, test_[, 1:13])
pr.nn_ <- pr.nn$net.result * (max(train$medv)-min(train$medv))
          + min(train$medv)
test.r <- (test_$medv)
           * (max(train$medv)-min(train$medv)) + min(train$medv)
MSE.nn <- sum((test.r - pr.nn_)^2)/nrow(test_)
```

```
round(c(MSE.lm,MSE.nn),1)
[1] 22.2  9.8
```

그 결과는 선형회귀 모형의 경우엔 MSE가 22.2이고 신경망 모형의 경우는 MSE가 9.8로 나타났다. 신경망 모형의 성능이 훨씬 좋다는 것을 의미한다. 다음은 모형성능을 적합값(수직축) 대 실제값(수평축)의 산점도로 보여주는 과정이다. 그림 4에서 좌우를 비교해보라.

```
par(mfrow=c(1,2))
plot(test$medv, pr.lm, xlim=c(0,60), ylim=c(0,60), col='blue',
     main='Real vs predicted lm',pch=18, cex=0.7)
abline(0,1,lwd=2)
legend('bottomright', legend='LM', pch=18, col='blue', cex=.8)
plot(test$medv, pr.nn_, xlim=c(0,60), ylim=c(0,60), col='red',
     main='Real vs predicted NN', pch=18, cex=0.7)
abline(0,1,lwd=2)
legend('bottomright', legend='NN', pch=18, col='red', cex=0.8)
```

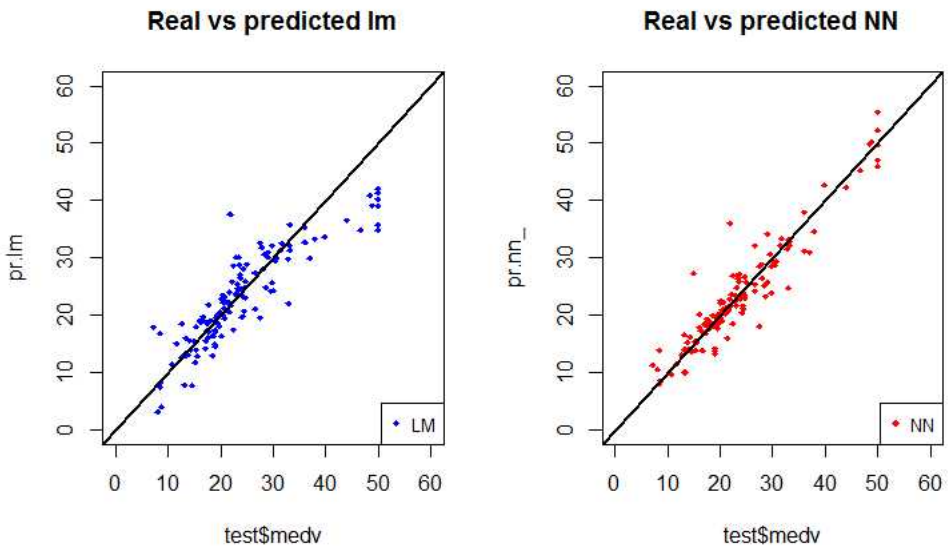


그림 4. 적합값(수직축) 대 실제값(수평축): 선형회귀모형[좌], 신경망모형[우]

6. 딥러닝

딥러닝(deep learning)은 은닉 층의 수가 많은 신경망 모형이다. 그렇다고 단순히 은닉 층을 높이 (깊게) 쌓는 것은 아니다. 대상 데이터의 특성을 살리는 특정한 구조의 노드를 체계적으로 끼워 넣은 신경망 모형이다. 이런 딥러닝은 이미지 처리와 사운드 분석 등 공학 분야의 지도학습에서 특히 좋은 성능을 발휘한다.

딥러닝을 위한 R 팩키지로는 “darch”가 있지만 실제적인 딥러닝을 위해서는 h2o 또는 TensorFlow와 같은 전문 소프트웨어를 활용할 필요가 있다.

딥러닝은 워낙 넓고 깊은 토픽이므로 이 장에서 개괄이라도 하는 것은 무리이다. 이 분야 개괄에 관하여는 다음 책을 보라.

- Jeff Heaton (2015). Artificial Intelligence for Humans Volume 3: Deep Learning and Neural Networks.
- 김의중 (2016). 인공지능, 머신러닝, 딥러닝 입문.

탐구문제

1. R의 “gclus” 팩키지에서 제공하는 spam 데이터에 대하여 신경망 모형을 구축하고 예측 오류율을 평가하라.