**An Introduction to Statistical Learning with Applications in R**

**(R의 응용 프로그램을 사용한 통계 학습 소개)**

Second Edition (두번째 버전)

**Start Date** : 23.01.01

**Student** : Rae Kyeong Jeong

**Professor** : Yujin Chung

**E-mail** : [yujinchung@kyonggi.ac.kr](mailto:yujinchung@kyonggi.ac.kr)

**Member**

~~정재형(17)~~ ~~김가영(19)~~ 김상철(18) 김현주(20) ~~송기현(20)~~ ~~박선현(21)~~ 정래경(19) 배진현(18) 오승태(19)

**Research Assistant Time**

Friday 17:00 ~ 19:00 [기존 ISLR , 데분논]

**Activity Report**

Every week Wednesday and Sunday

**PDF Download Site**

<https://www.statlearning.com/>

**Note**

1. 미리미리 번역이라도 해놓기!

2.

3.

4.

**Study Range**

8.2.2 Random Forests ~ 8.2.3 Boosting

-- 새로 시작 23.03.12 –

9.1.4 Construction of the Maximal Margin Classifier ~ 9.2.2 Details of the Support Vector Classifier

10 ~ 10.1 Deep learning

**8.2.2 Random Forests (랜덤 포레스트) ~ 8.2.3 Boosting (부스팅) p343 ~ p348**

Random Forests 랜덤 포레스트

Random forests provide an improvement over bagged trees by way of a small tweak that decorrelates the trees.

랜덤 포레스트는 트리들을 비연관시키는 작은 변화를 통해 배깅한 트리들보다 개선을 제공합니다.

As in bagging, we build a number of decision trees on bootstrapped training samples.

배깅에서와 마찬가지로, 우리는 부트스트랩 훈련 샘플에 많은 의사 결정 트리를 구축합니다.

But when building these decision trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors.

그러나 이러한 의사 결정 트리를 구축할 때, 트리의 분할을 고려할 때마다, m 예측 변수의 랜덤 표본이 p 예측 변수의 전체 집합에서 분할 후보로 선택됩니다.

The split is allowed to use only one of those m predictors.

분할은 이러한 m 예측 변수 중 하나만 사용할 수 있습니다.

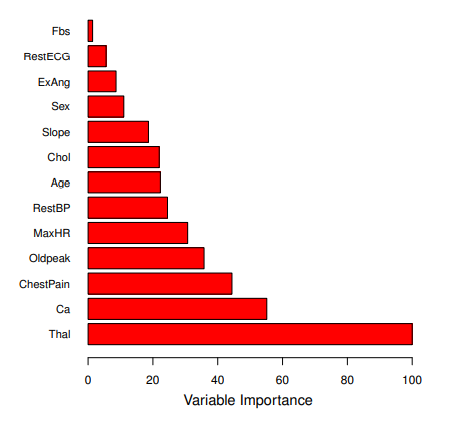
A fresh sample of m predictors is taken at each split, and typically we choose m ≈ √p

각 분할에서 m 예측 변수의 새로운 샘플이 추출되며, 그리고 일반적으로 우리는 m ≈ √p를 선택합니다.

that is, the number of predictors considered at each split is approximately equal to the square root of the total number of predictors. (4 out of the 13 for the Heart data) (심장 데이터의 경우 13개 중 4개).

즉, 각 분할에서 고려되는 예측 변수의 수는 총 예측 변수 수의 제곱근과 거의 같습니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ[ FIGURE 8.9. 그림 8.9. ]ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ



A variable importance plot for the Heart data.

심장 데이터에 대한 변수 중요도 그림입니다.

Variable importance is computed using the mean decrease in Gini index, and expressed relative to the maximum.

변수 중요도는 지니 지수의 평균 감소를 사용하여 계산되고, 그리고 최대값에 상대적으로 표현됩니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

In other words, in building a random forest, at each split in the tree, the algorithm is not even allowed to consider a majority of the available predictors.

즉, 트리의 각 분할에서 랜덤 포레스트를 구축할 때, 알고리즘은 사용 가능한 예측 변수의 대부분을 고려할 수 없습니다.

This may sound crazy, but it has a clever rationale.

미친 소리처럼 들릴 수도 있지만, 하지만 영리한 근거가 있습니다.

Suppose that there is one very strong predictor in the data set, along with a number of other moderately strong predictors.

데이터 집합에 다른 적당히 강한 예측 변수와 함께 매우 강한 예측 변수가 하나 있다고 가정합니다.

Then in the collection of bagged trees, most or all of the trees will use this strong predictor in the top split.

그런 다음 배깅한 트리의 집합에서, 대부분 또는 모든 트리가 맨 위 분할에서 이 강력한 예측 변수를 사용합니다.

Consequently, all of the bagged trees will look quite similar to each other.

결과적으로, 모든 배깅한 트리들은 서로 상당히 비슷하게 보일 것입니다.

Hence the predictions from the bagged trees will be highly correlated.

따라서 배깅한 트리들의 예측은 높은 상관관계가 있을 것입니다.

Unfortunately, averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities.

안타깝게도, 상관 관계가 높은 많은 양을 평균화하는 것은 상관 관계가 없는 많은 양을 평균화하는 것만큼 큰 분산 감소로 이어지지 않습니다.

In particular, this means that bagging will not lead to a substantial reduction in variance over a single tree in this setting.

특히, 이 설정에서 배깅이 단일 트리에 대한 분산의 상당한 감소로 이어지지 않음을 의미합니다.

Random forests overcome this problem by forcing each split to consider only a subset of the predictors.

랜덤 포리스트는 각 분할이 예측 변수의 하위 집합만 고려하도록 하여 이 문제를 해결합니다.

Therefore, on average (p − m)/p of the splits will not even consider the strong predictor, and so other predictors will have more of a chance.

따라서, 분할의 평균 (p − m)/p는 강력한 예측 변수도 고려하지 않으므로, 다른 예측 변수가 더 많은 확률을 갖습니다.

We can think of this process as decorrelating the trees, thereby making the average of the resulting trees less variable and hence more reliable.

우리는 이 과정을 트리와 비관련된 장식으로 생각할 수 있으며, 따라서 결과적인 트리의 평균을 덜 가변적이게 만들고 따라서 더 신뢰할 수 있게 만듭니다.

8.2 Bagging, Random Forests, Boosting, and Bayesian Additive Regression Trees

8.2 배깅, 랜덤 포레스트, 부스팅, 그리고 베이지안 가법 회귀 트리

The main difference between bagging and random forests is the choice of predictor subset size m.

배깅과 랜덤 포레스트의 주요 차이점은 예측 변수 부분 집합 크기 m을 선택하는 것입니다.

For instance, if a random forest is built using m = p, then this amounts simply to bagging.

예를 들어, m = p를 사용하여 랜덤 포레스트를 구축하는 경우, 그러면 이는 단순히 배깅에 해당합니다.

On the Heart data, random forests using m = √p leads to a reduction in both test error and OOB error over bagging (Figure 8.8).

Heart 데이터에서 m = √p를 사용하는 랜덤 포리스트는 테스트 오류와 OOB 오류 오버 배깅(그림 8.8)을 모두 감소시킵니다.

Using a small value of m in building a random forest will typically be helpful when we have a large number of correlated predictors.

랜덤 포리스트를 작성할 때 m의 작은 값을 사용하는 것은 우리가 일반적으로 상관된 예측 변수의 수가 많을 때 유용합니다.

We applied random forests to a high-dimensional biological data set consisting of expression measurements of 4,718 genes measured on tissue samples from 349 patients.

349명의 환자의 조직 샘플에서 측정된 4,718개 유전자의 발현 측정으로 구성된 고차원 생물학적 데이터 세트에 랜덤 포레스트를 적용했습니다.

There are around 20,000 genes in humans, and individual genes have different levels of activity, or expression, in particular cells, tissues, and biological conditions.

인간에게는 약 20,000개의 유전자가 있고, 개별 유전자는 특히 세포, 조직, 생물학적 조건에서 다른 수준의 활동 또는 발현을 가지고 있습니다.

In this data set, each of the patient samples has a qualitative label with 15 different levels: either normal or 1 of 14 different types of cancer.

이 데이터 세트에서, 각 환자 검체에는 정상 또는 14가지 암 유형 중 하나와 같은 15가지 수준의 정성 레이블이 있습니다.

Our goal was to use random forests to predict cancer type based on the 500 genes that have the largest variance in the training set.

우리의 목표는 랜덤 포레스트를 사용하여 훈련 세트에서 가장 큰 차이를 가진 500개의 유전자를 기반으로 암 유형을 예측하는 것이었습니다.

We randomly divided the observations into a training and a test set, and applied random forests to the training set for three different values of the number of splitting variables m.

우리는 관찰을 훈련과 테스트 세트로 무작위로 나누고, 그리고 분할 변수 m의 세 가지 다른 값에 대해 훈련 세트에 랜덤 포레스트를 적용했습니다.

The results are shown in Figure 8.10.

그 결과는 그림 8.10에 나와 있습니다.

The error rate of a single tree is 45.7 %, and the null rate is 75.4 %.

단일 트리의 오류율은 45.7%이고 null 비율은 75.4%입니다.

We see that using 400 trees is sufficient to give good performance, and that the choice m = √p gave a small improvement in test error over bagging (m = p) in this example.

우리는 400개의 트리를 사용하는 것이 좋은 성능을 제공하기에 충분하며, 그리고 이 예에서 m = √p 선택은 배깅(m = p)에 비해 테스트 오류가 약간 개선되었음을 알 수 있습니다.

As with bagging, random forests will not overfit if we increase B, so in practice we use a value of B sufficiently large for the error rate to have settled down.

배깅과 마찬가지로, 만약 우리가 B를 늘린다면 랜덤 포레스트가 과적합되지 않으므로, 실제로는 오류율이 안정될 정도로 충분히 큰 B 값을 사용합니다.

**8.2.3 Boosting 8.2.3 부스팅**

We now discuss boosting, yet another approach for improving the predictions resulting from a decision tree.

우리는 이제 의사 결정 트리에서 발생하는 예측 결과를 개선하기 위한 또 다른 접근법인 부스팅에 대해 논의합니다.

Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification.

배깅과 마찬가지로, 부스팅은 회귀 또는 분류를 위한 많은 통계적 학습 방법에 적용될 수 있는 일반적인 접근법입니다.

Here we restrict our discussion of boosting to the context of decision trees.

여기서 우리는 우리의 부스팅에 대한 논의를 의사 결정 트리의 맥락으로 제한합니다.

Recall that bagging involves creating multiple copies of the original training data set using the bootstrap, fitting a separate decision tree to each copy, and then combining all of the trees in order to create a single predictive model.

배깅에는 부트스트랩을 사용하여 기존의 훈련 데이터 세트의 여러 복사본을 만들고, 각 복사본에 별도의 의사 결정 트리를 맞춘 다음, 단일 예측 모델을 만들기 위해 모든 트리를 결합하는 작업이 포함되는 것을 기억합시다.

Notably, each tree is built on a bootstrap data set, independent of the other trees.

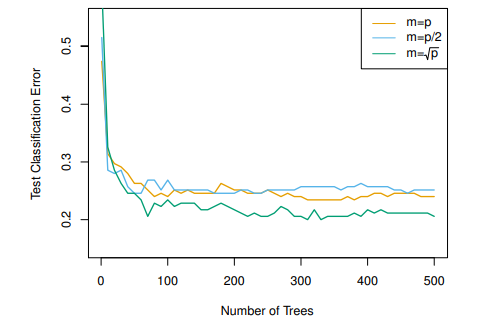
특히, 각 트리는 다른 트리와 독립적으로 부트스트랩 데이터 세트를 기반으로 구축됩니다.

Boosting works in a similar way, except that the trees are grown sequentially: each tree is grown using information from previously grown trees.

부스팅은 트리가 순차적으로 성장한다는 점을 제외하고는 유사한 방식으로 작동합니다: 각 트리는 이전에 성장한 트리의 정보를 사용하여 성장합니다.

Boosting does not involve bootstrap sampling; instead each tree is fit on a modified version of the original data set.

부스팅은 부트스트랩 샘플링을 포함하지 않으며; 대신 각 트리는 기존의 데이터 세트의 수정된 버전에 적합합니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ[ FIGURE 8.10. 그림 8.10 ] ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

Results from random forests for the 15-class gene expression data set with p = 500 predictors.

p = 500 예측 변수가 있는 15개의 클래스 유전자 발현 데이터 집합에 대한 랜덤 포리스트의 결과입니다.

The test error is displayed as a function of the number of trees.

검정 오류는 트리 수의 함수로 표시됩니다.

Each colored line corresponds to a different value of m, the number of predictors available for splitting at each interior tree node.

각 색상 선은 각 내부 트리 노드에서 분할할 수 있는 예측 변수의 수인 m의 서로 다른 값에 해당합니다.

Random forests (m<p) lead to a slight improvement over bagging (m = p).

랜덤 포레스트 (m<p)는 배깅 (m = p)보다 약간 개선됩니다.

A single classification tree has an error rate of 45.7 %.

단일 분류 트리의 오류율은 45.7%입니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

Consider first the regression setting. Like bagging, boosting involves combining a large number of decision trees, **ˆf^1 ,..., ˆf^B**.

먼저 회귀 설정을 고려합니다. 배깅과 마찬가지로, 부스팅은 다수의 의사 결정 트리인 **ˆf^1 ,..., ˆf^B**를 결합하는 것을 포함합니다.

Boosting is described in Algorithm 8.2.

부스팅은 알고리즘 8.2에 설명되어 있습니다.

What is the idea behind this procedure?

이 절차의 배경에 있는 아이디어는 무엇일까요?

Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead learns slowly.

단일 대규모 의사 결정 트리를 데이터에 적합시키는 것과 달리, 데이터를 엄격하게 적합시키고 잠재적으로 과적합시키는 것과 마찬가지로, 부스팅 접근 방식은 학습 속도가 느립니다.

Given the current model, we fit a decision tree to the residuals from the model.

현재 모형이 주어지면, 결정 트리를 모형의 잔차에 적합시킵니다.

That is, we fit a tree using the current residuals, rather than the outcome Y , as the response.

즉, 결과 Y가 아닌 현재 잔차를 반응으로 사용하여 트리를 적합시킵니다.

We then add this new decision tree into the fitted function in order to update the residuals.

우리는 그런 다음 잔차를 업데이트하기 위해 적합 함수에 이 새로운 결정 트리를 추가합니다.

Each of these trees can be rather small, with just a few terminal nodes, determined by the parameter d in the algorithm.

이러한 각 트리는 알고리즘의 매개 변수 d에 의해 결정되는 몇 개의 터미널 노드와 함께 다소 작을 수 있습니다.

By fitting small trees to the residuals, we slowly improve **ˆf** in areas where it does not perform well.

작은 트리를 잔차에 적합시킴으로써, 우리는 성능이 좋지 않은 영역에서 천천히 **ˆf**를 개선합니다.

The shrinkage parameter λ slows the process down even further, allowing more and different shaped trees to attack the residuals.

수축 모수 λ를 **사용하면** 공정 속도가 더욱 느려지므로, 더 많은 다른 모양의 트리가 잔차를 공격할 수 있습니다.

In general, statistical learning approaches that learn slowly tend to perform well.

일반적으로, 천천히 배우는 통계적 학습 접근법은 잘 수행되는 경향이 있습니다.

Note that in boosting, unlike in bagging, the construction of each tree depends strongly on the trees that have already been grown.

배깅과 달리 부스팅에서는 각 트리의 구성이 이미 자란 트리에 크게 의존한다는 점에 유의해야 합니다.

We have just described the process of boosting regression trees.

우리는 방금 부스팅 회귀 트리의 과정을 설명했습니다.

Boosting classification trees proceeds in a similar but slightly more complex way, and the details are omitted here.

부스팅 분류 트리는 비슷하지만 조금 더 복잡한 방식으로 진행되며, 자세한 내용은 여기서 생략합니다.

Boosting has three tuning parameters:

부스팅에는 세 가지 튜닝 매개 변수가 있습니다:

8.2 Bagging, Random Forests, Boosting, and Bayesian Additive Regression Trees

8.2 배깅, 랜덤 포레스트, 부스팅, 그리고 베이지안 가법 회귀 트리

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

Algorithm 8.2 Boosting for Regression Trees

회귀 트리에 대한 알고리즘 8.2 부스팅

1. Set ˆf(x) = 0 and ri = yi for all i in the training set.

1. 훈련 세트의 모든 i에 대해 ˆf(x) = 0 및 ri = yi를 설정합니다.

2. For b = 1, 2,...,B, repeat:

2. b = 1, 2, ..., B에 대해 다음을 반복합니다:

(a) Fit a tree **ˆf^b** with d splits (d + 1 terminal nodes) to the training data (X, r).

(a) d 분할 (d + 1 단자 노드)이 있는 트리 **ˆf^b** 를 훈련 데이터 (X, r)에 맞춥니다.

(b) Update **ˆf** by adding in a shrunken version of the new tree:

(b) 축소된 새 트리 버전을 추가하여 **ˆf**를 업데이트합니다:



(c) Update the residuals,

(c) 잔차를 업데이트합니다,

텍스트이(가) 표시된 사진

자동 생성된 설명

3. Output the boosted model,

3. 부스트된 모델을 출력합니다,

1. The number of trees B.

1. 트리의 수 B입니다.

Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select B.

배깅 및 랜덤 포레스트와 달리, 만약 B가 너무 크면 부스팅이 과적합될 수 있지만, 이러한 과적합은 조금이라도 느리게 발생하는 경향이 있습니다. 우리는 교차 검증을 사용하여 B를 선택합니다.

2. The shrinkage parameter λ, a small positive number.

2. 작은 양수인 수축 매개변수 λ입니다.

This controls the rate at which boosting learns.

이것은 부스팅이 학습하는 속도를 제어합니다.

Typical values are 0.01 or 0.001, and the right choice can depend on the problem.

일반적인 값은 0.01 또는 0.001이며, 올바른 선택은 문제에 따라 달라질 수 있습니다.

Very small λ can require using a very large value of B in order to achieve good performance.

매우 작은 λ는 우수한 성능을 달성하기 위해 매우 큰 B 값을 사용해야 할 것을 요구합니다.

3. The number d of splits in each tree, which controls the complexity of the boosted ensemble.

3. 각 트리의 분할 수 d로, 부스트 앙상블의 복잡성을 제어합니다.

Often d = 1 works well, in which case each tree is a stump, consisting of a single split.

종종 d = 1이 잘 작동하며, 이 경우 각 트리는 단일 분할로 구성된 그루터기입니다.

In this case, the boosted ensemble is fitting an additive model, since each term involves only a single variable.

이 경우, 각 항에는 단일 변수만 포함되기 때문에, 부스트 앙상블이 가법 모형을 적합시킵니다.

More generally d is the interaction depth, and controls the interaction order of the boosted model, since d splits can involve at most d variables.

더 일반적으로 d는 교호작용 깊이이며, d 분할에는 대부분의 d 변수가 포함될 수 있기 때문에, 부스트 모형의 교호작용 순서를 제어합니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

In Figure 8.11, we applied boosting to the 15-class cancer gene expression data set, in order to develop a classifier that can distinguish the normal class from the 14 cancer classes.

그림 8.11에서, 우리는 14개의 암 등급들로부터 정상적인 등급을 구별할 수 있는 분류기를 개발하기 위해, 15-class 암 유전자 발현 데이터 세트에 부스팅을 적용했습니다.

We display the test error as a function of the total number of trees and the interaction depth d.

우리는 검정 오류를 총 트리 수와 상호 작용 깊이 d의 함수로 표시합니다.

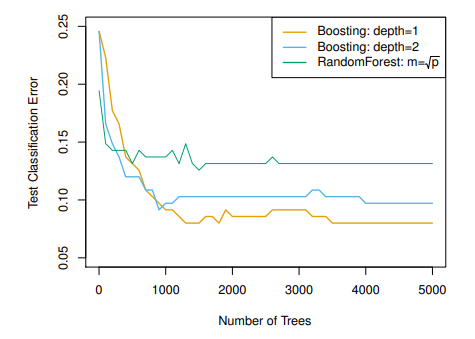
We see that simple stumps with an interaction depth of one perform well if enough of them are included.

우리는 상호 작용 깊이가 1인 단순한 그루터기가 충분히 포함되면 성능이 우수하다는 것을 알 수 있습니다.

This model outperforms the depth-two model, and both outperform a random forest.

이 모델은 깊이 2 모델보다 성능이 우수하며, 둘 다 랜덤 포레스트보다 성능이 우수합니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ[ FIGURE 8.11. 그림 8.11 ] ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ



Results from performing boosting and random forests on the 15-class gene expression data set in order to predict cancer versus normal.

암에 비해 정상을 예측하기 위해 15개의 클래스 유전자 발현 데이터 세트에 대해 부스팅 및 랜덤 포레스트를 수행한 결과입니다.

The test error is displayed as a function of the number of trees.

검정 오류는 트리 수의 함수로 표시됩니다.

For the two boosted models, λ = 0.01. 부스트된 두 모형의 경우, λ = 0.01입니다.

Depth-1 trees slightly outperform depth-2 trees, and both outperform the random forest, although the standard errors are around 0.02, making none of these differences significant.

깊이-1 트리는 깊이-2 트리를 약간 능가하며, 두 트리 모두 랜덤 포레스트를 능가하지만, 표준 오차는 약 0.02이므로 이러한 차이는 중요하게 만들지 않습니다.

The test error rate for a single tree is 24 %.

단일 트리에 대한 검정 오류율은 24%입니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

This highlights one difference between boosting and random forests: in boosting, because the growth of a particular tree takes into account the other trees that have already been grown, smaller trees are typically sufficient.

이것은 부스팅과 랜덤 포레스트 사이의 한 가지 차이점을 강조합니다. 부스팅에서, 특정 나무의 성장은 이미 자란 다른 나무들을 고려하기 때문에 일반적으로 작은 나무들로 충분합니다.

Using smaller trees can aid in interpretability as well; for instance, using stumps leads to an additive model.

작은 트리를 사용하면 해석 가능성에도 도움이 될 수 있습니다; 예를 들어, 그루터기()를 사용하면 가법 모형을 만들 수 있습니다.

\* Stumps 그냥 스텀프라고 읽기 [<https://velog.io/@woooa/TIL-%EC%88%9C%EC%97%B4-%EC%A4%91%EC%9A%94%EB%8F%84%EC%99%80-XGBoost>]

**9.1.4 Construction of the Maximal Margin Classifier ~ 9.2.2 Details of the Support Vector Classifier p372 ~ p379**

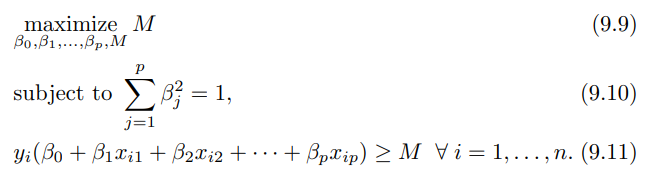
**9.1.4 Construction of the Maximal Margin Classifier 9.1.4 최대 마진(Margin) 분류기의 구성**

We now consider the task of constructing the maximal margin hyperplane based on a set of n training observations **x1,...,xn ∈ Rp** and associated class labels **y1,...,yn ∈ {−1, 1}.**

우리는 이제 n개의 훈련 관찰 데이터 **x1,...,xn ∈ Rp** 및 관련 클래스 레이블 **y1,...,yn ∈ {−1, 1}** 세트를 기반으로 최대 마진 초평면을 구성하는 작업을 고려합니다.

Briefly, the maximal margin hyperplane is the solution to the optimization problem

간단히 말해서, 최대 마진 초평면은 최적화 문제에 대한 해결책입니다.



This optimization problem (9.9)–(9.11) is actually simpler than it looks.

이 최적화 문제 (9.9)–(9.11)는 보기보다 실제로 더 간단합니다.

First of all, the constraint in (9.11) that guarantees that each observation will be on the correct side of the hyperplane, provided that M is positive.

우선, (9.11)의 제약 조건은 M이 양수인 경우 각 관측값이 초평면의 올바른 쪽에 있음을 보장(보증)합니다.

(Actually, for each observation to be on the correct side of the hyperplane we would simply need **yi(β0 + β1xi1 + β2xi2+···+βpxip) > 0**, so the constraint in (9.11) in fact requires that each observation be on the correct side of the hyperplane, with some cushion, provided that M is positive.)

(실제로, 각 관측치가 초평면의 올바른 쪽에 있기 위해서는 **yi(β0 + β1xi1 + β2xi2+··+βpxip) > 0**이면 되므로, (9.11)의 제약 조건은 실제로 M이 양의 경우 각 관측치가 일부 쿠션을 포함하여 초평면의 올바른 쪽에 있어야 합니다.)

Second, note that (9.10) is not really a constraint on the hyperplane, since if **β0 + β1xi1 + β2xi2 + ··· + βpxip = 0** defines a hyperplane, then so does **k(β0 +β1xi1 +β2xi2 +··· +βpxip) = 0** for any **k ̸= 0**.

둘째로, (9.10)은 초평면에서 실제로 제약이 되지 않습니다. 왜냐하면 **β0 + β1xi1 + β2xi2 + · · + βpxip = 0**이 초평면을 정의하면, **k ̸= 0**에 대하여 **k(β0 + β1xi1 + β2xi2 + · · + βpxip) = 0**도 초평면을 정의하기 때문입니다.

However, (9.10) adds meaning to (9.11); one can show that with this constraint the perpendicular distance from the ith observation to the hyperplane is given by ….

그러나, (9.10)는 (9.11)에 의미를 추가합니다; 이 제약 조건을 사용하여 i번째 관측에서 초평면까지의 수직 거리가 …에 의해 주어진다는 것을 보여줄 수 있습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Therefore, the constraints (9.10) and (9.11) ensure that each observation is on the correct side of the hyperplane and at least a distance M from the hyperplane.

따라서, 제약 조건 (9.10)과 (9.11)은 각 관측치가 초평면의 올바른 쪽에 있고 초평면으로부터 최소 거리 M에 있도록 확신합니다.

Hence, M represents the margin of our hyperplane, and the optimization problem chooses **β0, β1,..., βp** to maximize M.

따라서, M은 초평면의 마진을 나타내며, 최적화 문제는 M을 최대화하기 위해 **β0, β1, ..., βp**를 선택합니다.

This is exactly the definition of the maximal margin hyperplane!

이것이 바로 최대 마진 초평면의 정의입니다!

The problem (9.9)–(9.11) can be solved efficiently, but details of this optimization are outside of the scope of this book.

문제 (9.9)–(9.11)는 효율적으로 해결할 수 있지만, 이러한 최적화에 대한 자세한 내용은 이 책의 범위를 벗어납니다.

**9.1.5 The Non-separable Case 9.1.5 분리 불가능한 케이스**

The maximal margin classifier is a very natural way to perform classification, if a separating hyperplane exists.

최대 마진 분류기는 만약 분리 초평면이 존재하는 경우, 분류를 수행하는 매우 자연스러운 방법입니다.

However, as we have hinted, in many cases no separating hyperplane exists, and so there is no maximal margin classifier.

그러나, 우리는 암시한 바와 같이, 많은 경우 분리 초평면이 존재하지 않으므로, 최대 마진 분류기가 없습니다.

In this case, the optimization problem (9.9)–(9.11) has no solution with **M > 0**.

이러한 경우, 최적화 문제 (9.9)–(9.11)에는 **M > 0**인 해가 없습니다.

An example is shown in Figure 9.4.

그림 9.4에 예가 나와 있습니다.

In this case, we cannot exactly separate the two classes.

이러한 경우, 우리는 두 클래스를 정확히 구분할 수 없습니다.

However, as we will see in the next section, we can extend the concept of a separating hyperplane in order to develop a hyperplane that almost separates the classes, using a so-called soft margin.

그러나, 우리는 다음 섹션에서 살펴보겠지만, 소프트 마진이라고 불리는 것을 사용하여, 클래스를 거의 분리하는 초평면을 개발하기 위해 분리 초평면의 개념을 확장할 수 있습니다.

The generalization of the maximal margin classifier to the non-separable case is known as the support vector classifier.

분리 불가능한 경우에 대한 최대 마진 분류기의 일반화는 지원(서포트) 벡터 분류기로 알려져 있습니다.

**9.2 Support Vector Classifiers 9.2 서포트 벡터 분류기. [발표 여기서부터 다시 시작]**

**9.2.1 Overview of the Support Vector Classifier 9.2.1 서포트 벡터 분류기의 개요**

In Figure 9.4, we see that observations that belong to two classes are not necessarily separable by a hyperplane.

그림 9.4에서, 우리는 두 클래스에 속하는 관측값이 반드시 초평면으로 분리될 수 있는 것은 아님을 알 수 있습니다.

In fact, even if a separating hyperplane does exist, then there are instances in which a classifier based on a separating hyperplane might not be desirable.

실제로, 분리 초평면이 존재하더라도, 분리 초평면에 기반한 분류기가 바람직하지 않을 수 있는 경우(예시)가 있습니다.

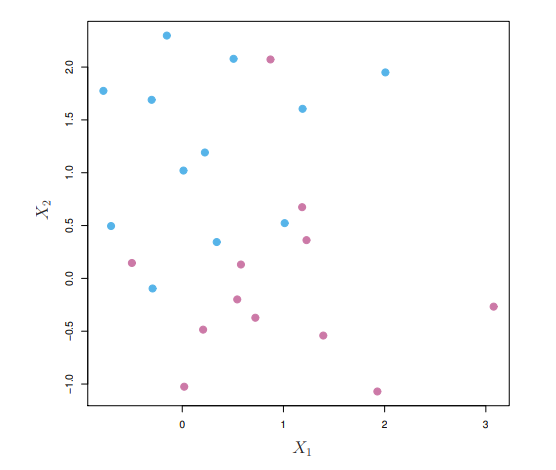
A classifier based on a separating hyperplane will necessarily perfectly classify all of the training observations;

분리 초평면을 기반으로 하는 분류기는 반드시 모든 학습 관찰을 완벽하게 분류합니다;

this can lead to sensitivity to individual observations.

이것은 개별 관찰에 대한 민감성을 초래할 수 있습니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ[ FIGURE 9.4. 그림 9.4. ] ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ



There are two classes of observations, shown in blue and in purple.

파란색과 보라색으로 표시된, 두 가지 관찰 등급이 있습니다.

In this case, the two classes are not separable by a hyperplane, and so the maximal margin classifier cannot be used.

이러한 경우, 두 클래스는 초평면으로 분리할 수 없으므로, 최대 마진 분류기를 사용할 수 없습니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

An example is shown in Figure 9.5. 그림 9.5에 예가 나와 있습니다.

The addition of a single observation in the right-hand panel of Figure 9.5 leads to a dramatic change in the maximal margin hyperplane.

그림 9.5의 오른쪽 패널에 단일 관찰을 추가하면 최대 마진 초평면에서 극적인 변화가 발생합니다.

The resulting maximal margin hyperplane is not satisfactory ㅡ for one thing, it has only a tiny margin.

결과적인 최대 마진 초평면은 만족스럽지 않습니다. ㅡ 한 가지는, 마진이 아주 작다는 점입니다.

This is problematic because as discussed previously, the distance of an observation from the hyperplane can be seen as a measure of our confidence that the observation was correctly classified.

이전에 논의한 바와 같이, 초평면에서 관찰까지의 거리는 관찰이 올바르게 분류되었다는 확신의 척도로 볼 수 있기 때문에 이것은 문제가 됩니다.

Moreover, the fact that the maximal margin hyperplane is extremely sensitive to a change in a single observation suggests that it may have overfit the training data.

게다가, 최대 마진 초평면이 단일 관찰의 변화에 극도로 민감하다는 사실은 훈련 데이터에 과적합되었을 수 있음을 제안합니다.

In this case, we might be willing to consider a classifier based on a hyperplane that does not perfectly separate the two classes, in the interest of …

이러한 경우, 우리는 …의 이익을 위해, 두 클래스를 완벽하게 분리하지 않는 초평면에 기반한 분류기를 고려할 수 있습니다.

\* Greater robustness to individual observations, and 개별 관찰에 대한 더 큰 견고성 및

\* Better classification of most of the training observations. 대부분의 훈련 관찰에 대한 더 나은 분류.

That is, it could be worthwhile to misclassify a few training observations in order to do a better job in classifying the remaining observations.

즉, 나머지 관찰을 더 잘 분류하기 위해 몇 가지 훈련 관찰을 잘못 분류하는 것이 가치가 있을 수 있습니다.

The support vector classifier, sometimes called a soft margin classifier, does exactly this.

소프트 마진 분류기라고도 하는 서포트 벡터 분류기가 정확히 이 작업을 수행합니다.

Rather than seeking the largest possible margin so that every observation is not only on the correct side of the hyperplane but also on the correct side of the margin, we instead allow some observations to be on the incorrect side of the margin, or even the incorrect side of the hyperplane.

모든 관측치가 초평면의 올바른 쪽에 있을 뿐만 아니라(또한) 마진의 정확한 쪽에 있을 수 있도록 가능한 가장 큰 마진을 찾는 대신, 우리는 일부 관측치가 마진의 잘못된 쪽에 있거나, 심지어 초평면의 잘못된 쪽에 있을 수 있도록 허용합니다.

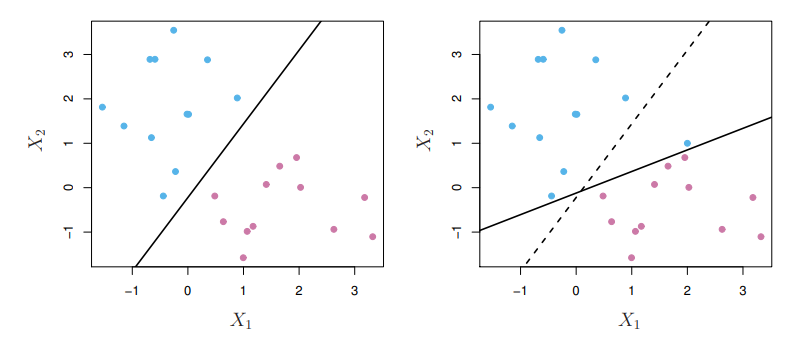
(The margin is soft because it can be violated by some of the training observations.)

(마진은 훈련 관찰 중 일부에 의해 위반될 수 있기 때문에 소프트(soft)입니다.)

An example is shown in the left-hand panel of Figure 9.6.

그림 9.6의 왼쪽 패널에 예가 나와 있습니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ[ FIGURE 9.5. 그림 9.5. ] ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ



Left: Two classes of observations are shown in blue and in purple, along with the maximal margin hyperplane.

왼쪽: 최대 마진 초평면과 함께 두 가지 관찰 클래스가 파란색과 보라색으로 표시됩니다.

Right: An additional blue observation has been added, leading to a dramatic shift in the maximal margin hyperplane shown as a solid line.

오른쪽: 추가적인 파란색 관찰이 추가되어, 실선으로 표시된 최대 마진 초평면에서 극적인 이동이 발생했습니다.

The dashed line indicates the maximal margin hyperplane that was obtained in the absence of this additional point.

점선은 이 추가 점이 없을 때 얻은 최대 마진 초평면을 나타냅니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

Most of the observations are on the correct side of the margin.

대부분의 관찰은 마진의 올바른 쪽에 있습니다.

However, a small subset of the observations are on the wrong side of the margin.

그러나, 관측치의 작은 하위 집합이 마진의 잘못된 쪽에 있습니다.

An observation can be not only on the wrong side of the margin, but also on the wrong side of the hyperplane.

관찰은 마진의 잘못된 쪽에 있을 뿐만 아니라 초평면의 잘못된 쪽에 있을 수도 있습니다.

In fact, when there is no separating hyperplane, such a situation is inevitable.

사실, 분리 초평면이 없을 때 그러한 상황은 불가피합니다.

Observations on the wrong side of the hyperplane correspond to training observations that are misclassified by the support vector classifier.

초평면의 잘못된 쪽에 있는 관측치는 서포트 벡터 분류기에 의해 잘못 분류된 훈련 관측치에 해당합니다.

The right-hand panel of Figure 9.6 illustrates such a scenario.

그림 9.6의 오른쪽 패널은 그러한 시나리오를 보여줍니다.

**9.2.2 Details of the Support Vector Classifier 9.2.2 서포트 벡터 분류기의 세부 사항**

The support vector classifier classifies a test observation depending on which side of a hyperplane it lies.

서포트 벡터 분류기는 초평면의 어느 쪽에 있는지에 따라 테스트 관찰을 분류합니다.

The hyperplane is chosen to correctly separate most of the training observations into the two classes, but may misclassify a few observations.

초평면은 대부분의 훈련 관찰을 두 클래스로 올바르게 분리하기 위해 선택되지만, 몇 가지 관찰을 잘못 분류할 수 있습니다.

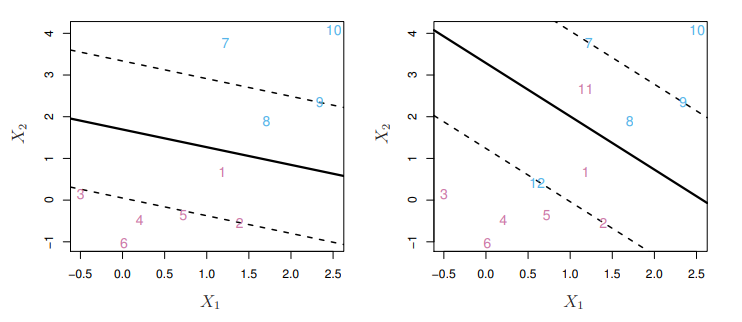
It is the solution to the optimization problem where C is a nonnegative tuning parameter.

C가 음이 아닌 조정 매개변수인 최적화 문제에 대한 솔루션(해결방법)입니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ[ FIGURE 9.6. 그림 9.6. ] ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

텍스트이(가) 표시된 사진

자동 생성된 설명



Left: A support vector classifier was fit to a small data set.

왼쪽: 서포트 벡터 분류기가 작은 데이터 세트에 적합했습니다.

The hyperplane is shown as a solid line and the margins are shown as dashed lines.

초평면은 실선으로 표시되고 마진은 파선(점선)으로 표시됩니다.

Purple observations: Observations 3, 4, 5, and 6 are on the correct side of the margin, observation 2 is on the margin, and observation 1 is on the wrong side of the margin.

보라색 관찰: 관찰 3, 4, 5, 6은 여백의 올바른 쪽에 있고, 관찰 2는 마진에 있고, 관찰 1은 마진의 잘못된 쪽에 있습니다.

Blue observations: Observations 7 and 10 are on the correct side of the margin, observation 9 is on the margin, and observation 8 is on the wrong side of the margin.

파란색 관측치: 관측치 7과 10은 마진의 올바른 쪽에 있고, 관측치 9는 마진에 있고, 관측치 8은 마진의 잘못된 쪽에 있습니다.

No observations are on the wrong side of the hyperplane.

초평면의 잘못된 쪽에 관측치가 없습니다.

Right: Same as left panel with two additional points, 11 and 12.

오른쪽: 11과 12라는 두 개의 추가 포인트가 있는 왼쪽 패널과 동일합니다.

These two observations are on the wrong side of the hyperplane and the wrong side of the margin.

이 두 관찰은 초평면의 잘못된 쪽에 있고 마진의 잘못된 쪽에 있습니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

As in (9.11), M is the width of the margin; we seek to make this quantity as large as possible.

(9.11)에서와 같이, M은 마진의 너비입니다; 우리는 이 양을 가능한 한 많이 만들기 위해 노력합니다.

In (9.14), **ϵ1,..., ϵn** are slack variables that allow individual observations to be on the wrong side of the margin or the hyperplane; we will explain them in greater detail momentarily.

(9.14)에서 **ϵ1,..., ϵn**은 개별적인 관찰이 마진 또는 초평면의 잘못된 쪽에 있을 수 있도록 허용하는 여유(완만한) 변수입니다; 잠시 후에 더 자세히 설명하겠습니다.

Once we have solved (9.12)–(9.15), we classify a test observation **x∗** as before, by simply determining on which side of the hyperplane it lies.

(9.12)~(9.15)를 풀면(~하면), 우리는 초평면의 어느 쪽에 있는지 간단히 결정하여 테스트 관찰 **x\***를 이전과 같이 분류합니다.

That is, we classify the test observation based on the sign of **f(x∗) = β0 + β1x∗1 + ··· + βpx∗p**.

즉, 우리는 **f(x∗) = β0 + β1x∗1 + ··· + βpx∗p**의 부호를 기반으로 테스트 관찰을 분류합니다.

The problem (9.12)–(9.15) seems complex, but insight into its behavior can be made through a series of simple observations presented below.

문제 (9.12)~(9.15)는 복잡해 보이지만, 아래에 제시된 일련의 간단한 관찰을 통해 그 동작에 대한 통찰력을 얻을 수 있습니다.

First of all, the slack variable **ϵi** tells us where the ith observation is located, relative to the hyperplane and relative to the margin.

우선, 여유(완만한) 변수 **ϵi**는 초평면과 마진을 기준으로 i번째 관측값이 어디에 있는지 우리에게 알려줍니다.

If **ϵi = 0** then the ith observation is on the correct side of the margin, as we saw in Section 9.1.4.

만약 **ϵi = 0**이면 섹션 9.1.4에서 본 것처럼, 우리는 i번째 관측값은 마진의 올바른 쪽에 있습니다.

If **ϵi > 0** then the ith observation is on the wrong side of the margin, and we say that the ith observation has violated the margin.

만약 **ϵi > 0**이면 i번째 관측치가 마진의 잘못된 쪽에 있으며, 우리는 i번째 관측치가 마진을 위반했다고 말합니다.

If ϵ**i > 1** then it is on the wrong side of the hyperplane.

만약 **ϵi > 1**이면 초평면의 잘못된 쪽에 있는 것입니다.

We now consider the role of the tuning parameter C.

우리는 이제 튜닝 매개변수 C의 역할을 고려합니다.

In (9.15), C bounds the sum of the **ϵi**’s, and so it determines the number and severity of the violations to the margin (and to the hyperplane) that we will tolerate.

(9.15)에서, C는 **ϵi**의 합을 제한하므로, 우리가 허용할 마진(및 초평면)에 대한 위반의 수와 심각도를 결정합니다.

We can think of C as a budget for the amount that the margin can be violated by the n observations.

우리는 C를 n개의 관찰에 의해 마진이 위반될 수 있는 양에 대한 예산으로 생각할 수 있습니다.

If C = 0 then there is no budget for violations to the margin, and it must be the case that **ϵ1 = ··· = ϵn = 0**, in which case (9.12)–(9.15) simply amounts to the maximal margin hyperplane optimization problem (9.9)–(9.11).

만약 C = 0인 경우 마진의 위반에 대한 예산이 없으며, 이 경우 **φ1 = · · = φn = 0**이어야 합니다. 이 경우 (9.12)– (9.15)는 단순히 최대 마진 초평면 최적화 문제 (9.9)– (9.11)에 해당합니다. **φ PI**

(Of course, a maximal margin hyperplane exists only if the two classes are separable.)

(물론, 최대 여백 초평면은 두 클래스가 분리 가능한 경우에만 존재합니다.)

For **C > 0** no more than C observations can be on the wrong side of the hyperplane, because if an observation is on the wrong side of the hyperplane then **ϵi > 1**, and (9.15) requires that **합n i=1 ϵi ≤ C**.

**C > 0**의 경우, 만약 관측치가 초평면의 잘못된 쪽에 있으면 **σi > 1**이 되고, (9.15)는 **합n i=1 ϵi ≤ C**가 필요하기 때문에, 초평면의 잘못된 쪽에 C 이상의 관측치가 있을 수 없습니다.

As the budget C increases, we become more tolerant of violations to the margin, and so the margin will widen.

예산 C가 증가함에 따라, 우리는 마진에 대한 위반에 대해 더 관대해지며, 그리고 마진이 넓어집니다.

Conversely, as C decreases, we become less tolerant of violations to the margin and so the margin narrows.

반대로, C가 감소함에 따라, 우리는 마진의 위반에 대한 관용이 줄어들어 마진이 좁아집니다.

An example is shown in Figure 9.7.

그림 9.7에 예가 나와 있습니다.

In practice, C is treated as a tuning parameter that is generally chosen via cross-validation.

실제로, C는 일반적으로 교차 검증을 통해 선택되는 튜닝 매개변수로 취급됩니다.

As with the tuning parameters that we have seen throughout this book, C controls the bias variance trade-off of the statistical learning technique.

이 책 전체에서 본 튜닝 매개변수와 마찬가지로, C는 통계 학습 기법(기술)의 편향 분산 균형(트레이드 오프)를 제어합니다.

When C is small, we seek narrow margins that are rarely violated;

C가 작을 때, 우리는 거의 위반되지 않는 좁은 마진을 찾습니다;

this amounts to a classifier that is highly fit to the data, which may have low bias but high variance.

이것은 편향은 낮지만 분산은 높은 데이터에 매우 적합한 분류기에 해당합니다.

On the other hand, when C is larger, the margin is wider and we allow more violations to it;

반면에, C가 크면, 마진이 더 넓어지고 더 많은 위반이 허용됩니다;

this amounts to fitting the data less hard and obtaining a classifier that is potentially more biased but may have lower variance.

이것은 데이터를 덜 어렵게 적합(피팅)하고 잠재적으로 더 편향되지만 더 낮은 분산을 가질 수 있는 분류기를 얻는 것과 같습니다.

The optimization problem (9.12)–(9.15) has a very interesting property:

최적화 문제 (9.12)–(9.15)는 매우 흥미로운 속성을 가지고 있습니다:

it turns out that only observations that **either lie** on the margin or that violate the margin will affect the hyperplane, and hence the classifier obtained.

오직 마진에 있거나 마진을 위반하는 관측값만 초평면에 영향을 미치므로 따라서 분류기가 획득됩니다.

In other words, an observation that **lies** strictly on the correct side of the margin does not affect the support vector classifier!

즉, 정확히 마진의 올바른 쪽에 있는 관측값은 서포트 벡터 분류기에 영향을 미치지 않습니다!

Changing the position of that observation would not change the classifier at all, provided that its position remains on the correct side of the margin.

그 위치가 마진의 올바른 쪽에 남아 있다면, 해당 관측치의 위치를 변경해도 분류기는 전혀 변경되지 않습니다.

Observations that lie directly on the margin, or on the wrong side of the margin for their class, are known as support vectors.

마진에 직접적으로 놓이거나, 해당 클래스의 마진의 잘못된 쪽에 있는 관측값을 서포트 벡터라고 알려져 있습니다.

These observations do affect the support vector classifier.

이러한 관찰은 서포트 벡터 분류기에 영향을 미칩니다.

The fact that only support vectors affect the classifier is in line with our previous assertion that C controls the bias-variance trade-off of the support vector classifier.

서포트 벡터만 분류기에 영향을 미친다는 사실은 C가 서포트 벡터 분류기의 편차-분산 맞바꿈(트레이드 오프)를 제어한다는 이전 주장과 일치합니다.

When the tuning parameter C is large, then the margin is wide, many observations violate the margin, and so there are many support vectors.

튜닝 매개변수 C가 크면, 마진이 넓고, 많은 관측치가 마진을 위반하므로, 서포트 벡터가 많습니다.

In this case, many observations are involved in determining the hyperplane.

이러한 경우, 초평면을 결정하는 데 많은 관찰이 포함됩니다.

The top left panel in Figure 9.7 illustrates this setting:

그림 9.7의 왼쪽 상단 패널은 이 설정을 설명합니다:

this classifier has low variance (since many observations are support vectors) but potentially high bias.

이 분류기는 분산이 낮지만 (많은 관측 값이 서포트 벡터이기 때문에) 잠재적으로 높은 편향이 있습니다.

In contrast, if C is small, then there will be fewer support vectors and hence the resulting classifier will have low bias but high variance.

반대로, C가 작으면 서포트 벡터가 적어지므로 따라서 결과 분류기는 편향는 낮지만 분산은 높습니다.

The bottom right panel in Figure 9.7 illustrates this setting, with only eight support vectors.

그림 9.7의 오른쪽 아래 패널은 8개의 서포트 벡터만 있는 이 설정을 보여줍니다.

The fact that the support vector classifier’s decision rule is based only on a potentially small subset of the training observations (the support vectors) means that it is quite robust to the behavior of observations that are far away from the hyperplane.

서포트 벡터 분류기의 결정 규칙이 훈련 관찰 (서포트 벡터)의 잠재적으로 작은 하위 집합에만 기반한다는 사실은 초평면에서 멀리 떨어진 관찰의 동작에 상당히 견고하다는 것을 의미합니다.

This property is distinct from some of the other classification methods that we have seen in preceding chapters, such as linear discriminant analysis.

이 속성(소유물?)은 선형 판별 분석과 같이 이전 장에서 본 다른 분류 방법과 구별됩니다.

Recall that the LDA classification rule depends on the mean of all of the observations within each class, as well as the within-class covariance matrix computed using all of the observations.

LDA 분류 규칙은 각 클래스 내의 모든 관측값의 평균과 게다가 모든 관측값을 사용하여 계산된 클래스 내 공분산 행렬에 따라 불리어집니다.

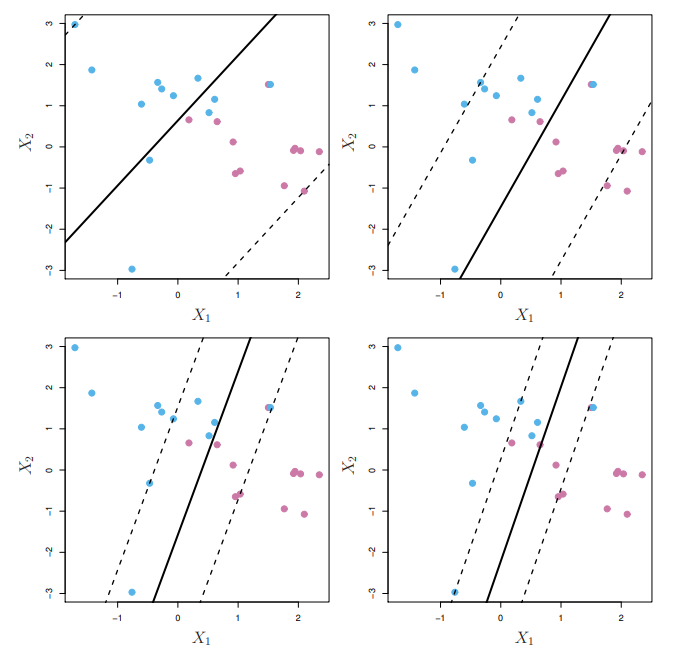
In contrast, logistic regression, unlike LDA, has very low sensitivity to observations far from the decision boundary.

대조적으로, 로지스틱 회귀는, LDA와 달리, 결정 경계에서 멀리 떨어진 관측치에 대해 매우 낮은 민감도를 갖습니다.

In fact we will see in Section 9.5 that the support vector classifier and logistic regression are closely related.

사실 우리는 서포트 벡터 분류기와 로지스틱 회귀가 밀접하게 관련되어 있음을 섹션 9.5에서 볼 수 있습니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ[ FIGURE 9.7. 그림 9.7. ] ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ



A support vector classifier was fit using four different values of the tuning parameter C in (9.12)–(9.15).

서포트 벡터 분류기는 (9.12)-(9.15)에서 튜닝 매개변수 C의 4가지 다른 값을 사용하여 적합했습니다.

The largest value of C was used in the top left panel, and smaller values were used in the top right, bottom left, and bottom right panels.

C의 가장 큰 값은 왼쪽 상단 패널에 사용되었고, 더 작은 값은 오른쪽 상단, 왼쪽 하단 및 오른쪽 하단 패널에 사용되었습니다.

When C is large, then there is a high tolerance for observations being on the wrong side of the margin, and so the margin will be large.

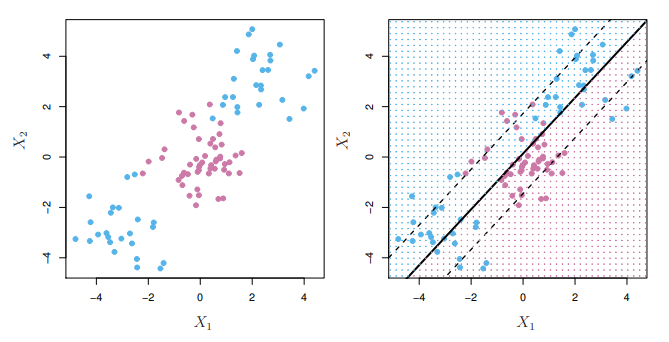
C가 크면, 마진의 잘못된 쪽에 있는 관측치에 대한 높은 허용오차가 있으므로 마진이 커집니다.

As C decreases, the tolerance for observations being on the wrong side of the margin decreases, and the margin narrows.

C가 감소하면, 마진의 잘못된 쪽에 있는 관측치에 대한 허용오차가 감소하고, 마진이 좁아집니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ[ FIGURE 9.8. 그림 9.8. ] ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ



Left: The observations fall into two classes, with a non-linear boundary between them.

왼쪽: 관측값은 두 클래스 사이에 비선형 경계가 있는 두 클래스로 나뉩니다.

Right: The support vector classifier seeks a linear boundary, and consequently performs very poorly.

오른쪽: 서포트 벡터 분류기는 선형 경계를 찾고, 결과적으로 성능이 매우 낮습니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

**10 Deep Learning딥러닝 p403 ~ p407**

This chapter covers the important topic of deep learning.

이 장에서는 딥러닝의 중요한 주제를 다룹니다.

At the time of writing (2020), deep learning is a very active area of research in the machine learning and artificial intelligence communities.

집필 시점 (2020)에서 딥러닝은 머신러닝과 인공지능 커뮤니티에서 매우 활발한 연구 영역입니다.

The cornerstone of deep learning is the neural network.

딥러닝의 초석(기초)은 신경망입니다.

Neural networks rose to fame in the late 1980s.

신경망은 1980년대 후반에 명성을 얻었습니다.

There was a lot of excitement and a certain amount of hype associated with this approach, and they were the impetus for the popular Neural Information Processing Systems meetings (NeurIPS, formerly NIPS) held every year, typically in exotic places like ski resorts.

이 접근법과 관련하여 많은 관심과 어느 정도의 과대 광고가 있었고, 그것들은 매년 스키 리조트와 같은 이국적인 장소에서 개최되는 인기 있는 신경 정보 처리 시스템 학회 (NeurIPS, 이전 NIPS)에서 추진력을 얻었습니다.

This was followed by a synthesis stage, where the properties of neural networks were analyzed by machine learners, mathematicians and statisticians; algorithms were improved, and the methodology stabilized.

이후 기계 학습자, 수학자 및 통계학자가 신경망의 특성을 분석하는 합성 단계가 이어졌습니다; 알고리즘이 개선되고 방법론이 안정화되었습니다.

Then along came SVMs, boosting, and random forests, and neural networks fell somewhat from favor.

그 후 SVM, 부스팅 및 랜덤 포레스트가 등장했으며 신경망은 다소 인기가 떨어졌습니다.

Part of the reason was that neural networks required a lot of tinkering, while the new methods were more automatic.

그 이유 중 일부는 신경망이 많은 수리를 필요로 하는 반면, 새로운 방법은 더 자동적이었기 때문입니다.

Also, on many problems the new methods outperformed poorly-trained neural networks.

또한, 많은 문제에서 새로운 방법은 제대로 훈련되지 않은 신경망을 능가합니다.

This was the status quo for the first decade in the new millennium.

이것은 새로운 천년(밀레니엄)의 첫 10년 동안의 현상이었습니다. -> 2000년 ~ 2010년을 말하는 듯

All the while, though, a core group of neural-network enthusiasts were pushing their technology harder on ever-larger computing architectures and data sets.

그러나, 그 동안, 신경망 애호가들의 핵심 그룹은 점점 더 큰 컴퓨팅 구조와 데이터 세트에 대해 그들의 기술을 더 강하게 밀어붙였습니다.

Neural networks resurfaced after 2010 with the new name deep learning, with new architectures, additional bells and whistles, and a string of success stories on some niche problems such as image and video classification, speech and text modeling.

신경망은 2010년 이후 새로운 아키텍처, 추가적인 매력적인 부가기능[동의어], 이미지 및 비디오 분류, 음성 및 텍스트 모델링과 같은 틈새 문제에 대한 일련의 성공 사례와 함께 딥러닝이라는 새로운 이름으로 다시 부상했습니다.

Many in the field believe that the major reason for these successes is the availability of ever-larger training datasets, made possible by the wide-scale use of digitization in science and industry.

현장의 많은 사람들은 이러한 성공의 주요 원인이 과학 및 산업에서 디지털화를 광범위하게 사용함으로써 가능해진 점점 더 큰 훈련 데이터 세트의 가용성이라고 생각합니다.

In this chapter we discuss the basics of neural networks and deep learning, and then go into some of the specializations for specific problems, such as convolutional neural networks (CNNs) for image classification, and recurrent neural networks (RNNs) for time series and other sequences.

이 장에서는 우리는 신경망과 딥러닝의 기본 사항에 대해 논의한 다음, 이미지 분류를 위한 컨볼루션 신경망 (CNN)과 시계열 및 기타 시퀀스를 위한 반복 신경망 (RNN)과 같은 특정 문제에 대한 일부 전문화(전문적으로)에 대해 알아봅니다.

We will also demonstrate these models using the R package keras, which interfaces with the tensorflow deep-learning software developed at Google.

우리는 또한 구글에서 개발한 텐서플로 딥러닝 소프트웨어와 상호작용하는 R 패키지 케라를 사용하여 이러한 모델을 시연할 것입니다.

The material in this chapter is slightly more challenging than elsewhere in this book.

이 장의 내용은 이 책의 다른 곳보다 조금 더 도전적입니다.

**10.1 Single Layer Neural Networks 10.1 단일 레이어(계층) 신경망**

A neural network takes an input vector of p variables X = (X1, X2,...,Xp) and builds a nonlinear function f(X) to predict the response Y .

신경망은 p개의 변수 X = (X1, X2, ..., Xp)의 입력 벡터를 취하고 비선형 함수 f(X)를 구축하여 반응 Y를 예측합니다.

We have built nonlinear prediction models in earlier chapters, using trees, boosting and generalized additive models.

우리는 트리, 부스팅 및 일반화된 가산(가법) 모델을 사용하여 이전 장에서 비선형 예측 모델을 구축했습니다.

What distinguishes neural networks from these methods is the particular structure of the model.

신경망과 이러한 방법을 구별하는 것은 모델의 특정 구조입니다.

Figure 10.1 shows a simple feed-forward neural network for modeling a quantitative response using p = 4 predictors.

그림 10.1은 p = 4인 예측 변수를 사용하여 정량적 반응을 모델링하기 위한 간단한 피드포워드 신경망을 보여줍니다.

\* 피드포워드 : 피드백의 반대말이라고 생각하면 될듯

In the terminology of neural networks, the four features X1,...,X4 make up the units in the input layer.

신경망 용어를 보면, X1, ..., X4의 네 가지 특징은 입력 층(레이어)의 단위를 구성합니다.

The arrows indicate that each of the inputs from the input layer feeds into each of the K hidden units (we get to pick K; here we chose 5).

화살표들은 입력 층(레이어)의 각 입력이 K개의 숨겨진 단위로 공급됨을 나타냅니다 (우리는 K를 선택합니다. 여기서 5개를 선택했습니다).

The neural network model has the form (10.1).

신경망 모델의 형태는 (10.1)입니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

It is built up here in two steps.

이것은 두 단계로 여기에 지어집니다.

First the K activations Ak, k = 1, . . . , K, in the hidden layer are computed as functions of the input features X1,...,Xp, where g(z) is a nonlinear activation function that is specified in advance.

먼저 hidden 레이어[은닉층]의 K개의 활성화 Ak, k = 1, . ., K는 입력(input) 특징 X1, ..., Xp의 함수로 계산됩니다. 여기서 g(z)는 사전에 지정된 비선형 활성화 함수입니다.



We can think of each Ak as a different transformation hk(X) of the original features, much like the basis functions of Chapter 7.

우리는 각 Ak를 7장의 기본 함수와 마찬가지로, 원래 특징의 다른 변환 hk(X)로 생각할 수 있습니다.

\* 7장에서 단순한 선형 모델을 알아봤었습니다.

These K activations from the hidden layer then feed into the output layer, resulting in (10.3) a linear regression model in the K = 5 activations.

hidden 레이어[은닉층]의 이러한 K개의 활성화는 출력 층(레이어)로 공급되어 K = 5인 활성화에서 (10.3) 선형 회귀 모델을 생성합니다.

텍스트이(가) 표시된 사진

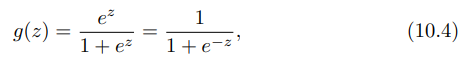
자동 생성된 설명

All the parameters β0,..., βK and w10,...,wKp need to be estimated from data.

모든 모수 β0, ..., βK 및 w10, ..., wKp는 데이터에서 추정해야 합니다.

In the early instances of neural networks, the sigmoid activation function was favored, (10.4) which is the same function used in logistic regression to convert a linear function into probabilities between zero and one (see Figure 10.2).

신경망의 초기 사례에서, 선형 함수를 0과 1 사이의 확률로 변환하기 위해 로지스틱 회귀에서 사용되는 것과 동일한 함수인 sigmoid 활성화 함수(10.4)가 선호되었습니다 (그림 10.2 참조).



ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ[ FIGURE 10.1. 그림 10.1. ] ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

도표이(가) 표시된 사진

자동 생성된 설명

Neural network with a single hidden layer.

하나

하나(한 줄)의 hidden 레이어 [은닉층]가 있는 신경망입니다.

The hidden layer computes activations Ak = hk(X) that are nonlinear transformations of linear combinations of the inputs X1, X2,...,Xp.

hidden 레이어 [은닉층]는 입력(input)들 X1, X2, ..., Xp의 선형 조합의 비선형 변환인 활성화 Ak = hk(X)를 계산합니다.

Hence these Ak are not directly observed.

따라서 이러한 Ak는 직접적으로 관찰되지 않습니다.

The functions hk(·) are not fixed in advance, but are learned during the training of the network.

함수 hk(·)는 사전에 고정된 것이 아니지만, 네트워크 훈련 중에 학습됩니다.

The output layer is a linear model that uses these activations Ak as inputs, resulting in a function f(X).

출력(output) 층(레이어)는 이러한 활성화 Ak를 입력으로 사용하여 함수 f(X)를 생성하는 선형 모델입니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

The preferred choice in modern neural networks is the ReLU (rectified linear unit) activation function, which takes the form (10.5)

현대 신경망에서 선호되는 선택은 ReLU(수정된 선형 단위) 활성화 함수이며 형식은 (10.5)입니다

텍스트이(가) 표시된 사진

자동 생성된 설명

A ReLU activation can be computed and stored more efficiently than a sigmoid activation.

ReLU 활성화는 시그모이드 활성화보다 더 효율적으로 계산되고 저장될 수 있습니다.

Although it thresholds at zero, because we apply it to a linear function (10.2) the constant term wk0 will shift this inflection point.

비록 이 값이 0에서 임계값이지만, 선형 함수 (10.2)에 적용되기 때문에 상수항 wk0가 이 변곡점으로 이동합니다.

So in words, the model depicted in Figure 10.1 derives five new features by computing five different linear combinations of X, and then squashes each through an activation function g(·) to transform it.

즉, 그림 10.1에 나타낸 모델은 X의 5개의 서로 다른 선형 조합을 계산하여 5개의 새로운 특징을 도출한 다음, 활성화 함수 g(·)를 통해 각각을 압축하여 변환합니다.

The final model is linear in these derived variables.

최종 모형은 이러한 파생 변수에서 선형이 됩니다.

The name neural network originally derived from thinking of these hidden units as analogous to neurons in the brain — values of the activations Ak = hk(X) close to one are firing, while those close to zero are silent (using the sigmoid activation function).

신경망이라는 이름은 원래 이러한 숨겨진 단위가 뇌의 뉴런과 유사하다고 생각하는 데서 유래되었습니다. ㅡ Ak = hk(X) 활성화 값은 1에 가까운 것이 발화(?)하는 반면, 0에 가까운 것은 침묵(?)합니다 (S 시그모이드 활성화 함수 사용).

\* 그냥 0과 1사이를 가진다고 해석

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ[ FIGURE 10.2. 그림 10.2. ] ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

차트이(가) 표시된 사진

자동 생성된 설명

Activation functions. 활성화 함수입니다.

The piecewise-linear ReLU function is popular for its efficiency and computability.

부분 선형 ReLU 함수는 효율성과 계산 가능성으로 유명합니다.

We have scaled it down by a factor of five for ease of comparison.

우리는 비교하기 쉽도록 5배로 축소했습니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

The nonlinearity in the activation function g(·) is essential, since without it the model f(X) in (10.1) would collapse into a simple linear model in X1,...,Xp.

활성화 함수 g(·)의 비선형성은 필수적입니다. 왜냐하면, 이것이 없으면 (10.1)의 모델 f(X)가 X1, ..., Xp의 단순 선형 모델로 축소되기 때문입니다.

Moreover, having a nonlinear activation function allows the model to capture complex nonlinearities and interaction effects.

게다가, 비선형 활성화 함수를 사용하면 모델이 복잡한 비선형성과 상호 작용 효과를 보존할 수 있습니다. [가질 수 있다.]

Consider a very simple example with p = 2 input variables X = (X1, X2), and K = 2 hidden units h1(X) and h2(X) with g(z) = z2.

p = 2일때 입력 변수 X = (X1, X2) 및 K = 2일때 숨겨진 단위 h1(X) 및 h2(X)를 g(z) = z2로 사용하는 매우 간단한 예제를 생각해 보려고 합니다.

We specify the other parameters as (10.6).

우리는 다른 모수를 (10.6)과 같이 지정합니다.

텍스트, 시계, 손목시계이(가) 표시된 사진

자동 생성된 설명

From (10.2), this means that (10.7). (10.2)에서 이는 (10.7)을 의미합니다.

텍스트, 손목시계, 게이지이(가) 표시된 사진

자동 생성된 설명

Then plugging (10.7) into (10.1), we get (10.8). 그런 다음 (10.7)을 (10.1)에 연결하면 (10.8)이 됩니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

So the sum of two nonlinear transformations of linear functions can give us an interaction!

따라서 선형 함수의 두 비선형 변환의 합은 우리에게 상호 작용을 제공할 수 있습니다!

In practice we would not use a quadratic function for g(z), since we would always get a second-degree polynomial in the original coordinates X1,...,Xp.

실제로 우리는 원래 좌표 X1, ..., Xp에서 항상 2차 다항식을 얻기 때문에, g(z)에 대해 2차 함수를 사용하지 않을 것입니다.

The sigmoid or ReLU activations do not have such a limitation.

시그모이드 또는 ReLU 활성화에는 이러한 제한이 없습니다.

Fitting a neural network requires estimating the unknown parameters in (10.1).

신경망을 적합시키려면 (10.1)에서 알 수 없는 매개 변수를 추정해야 합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

For a quantitative response, typically squared-error loss is used, so that the parameters are chosen to minimize (10.9).

정량적 반응의 경우, 일반적으로 제곱 오차 손실이 사용되므로, 모수가 최소화(10.9)되도록 선택됩니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ[ FIGURE 10.3. 그림 10.3. ] ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

텍스트이(가) 표시된 사진

자동 생성된 설명

Examples of handwritten digits from the MNIST corpus.

MNIST corpus의 손으로 쓴 숫자의 예입니다.

Each grayscale image has 28 × 28 pixels, each of which is an eight-bit number (0–255) which represents how dark that pixel is.

각 그레이스케일(회색조) 이미지에는 28 × 28 픽셀이 있으며, 각 픽셀은 해당 픽셀의 어두운 정도를 나타내는 8비트 숫자 (0-255)입니다.

The first 3, 5, and 8 are enlarged to show their 784 individual pixel values.

처음 3, 5, 8은 784개의 개별 픽셀 값을 표시하도록 확대됩니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

Details about how to perform this minimization are provided in Section 10.7.

이러한 최소화를 수행하는 방법에 대한 자세한 내용은 섹션 10.7에 나와 있습니다.

**10.5 Recurrent Neural Networks 순환(반복) 신경망 p421 ~ p424**

- 순차 데이터나 시계열 데이터를 이용하는 인공 신경망 유형임. 이 딥러닝 알고리즘은 언어변환, 자연어 처리, 음성 인식, 이미지 캡션과 같은 순서 문제나 시간 문제에 흔히 사용됩니다.

Many data sources are sequential in nature, and call for special treatment when building predictive models.

많은 데이터 소스는 본질적으로 순차적이며, 예측 모델을 구축할 때 특별한 처리가 필요합니다.

Examples include:

예를 들어 다음과 같습니다:

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

원, 도표, 스크린샷이(가) 표시된 사진

자동 생성된 설명

FIGURE 10.12.

그림 10.12.

Schematic of a simple recurrent neural network.

간단한 반복 신경망의 도식.

The input is a sequence of vectors **{Xl}L1** , and here the target is a single response.

입력은 벡터 **{Xl}L1**의 시퀀스이며, 여기서 타겟은 단일 반응입니다.

The network processes the input sequence **X** sequentially; each **Xl** feeds into the hidden layer, which also has as input the activation vector **Al−1** from the previous element in the sequence, and produces the current activation vector **Al**.

네트워크는 입력 시퀀스 **X**를 순차적으로 처리합니다; 각 **Xl**은 은닉층으로 공급되며, 은닉층은 시퀀스의 이전 요소에서 활성화 벡터 **Al-1**을 입력하고 현재 활성화 벡터 **Al**을 생성합니다.

The same collections of weights **W**, **U** and **B** are used as each element of the sequence is processed.

시퀀스의 각 요소가 처리될 때 동일한 가중치 **W**, **U** 및 **B** 컬렉션이 사용됩니다.

The output layer produces a sequence of predictions **Ol** from the current activation **Al**, but typically only the last of these, **OL**, is of relevance.

출력층은 현재 활성화 **Al**로부터 일련의 예측 **Ol**을 생성하지만, 일반적으로 마지막 예측 **OL**만 관련이 있습니다.

To the left of the equal sign is a concise representation of the network, which is unrolled into a more explicit version on the right.

등호 왼쪽에는 네트워크의 간결한 표현이 있으며, 오른쪽에는 보다 명확한 버전으로 전개됩니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

- Documents such as book and movie reviews, newspaper articles, and tweets.

서적 및 영화 리뷰, 신문 기사, 트윗 등의 문서.

The sequence and relative positions of words in a document capture the narrative, theme and tone, and can be exploited in tasks such as topic classification, sentiment analysis, and language translation.

문서에서 단어의 시퀀스와 상대적 위치는 서술, 주제 및 어조를 포착하며 주제 분류, 감정 분석 및 언어 번역과 같은 작업에 활용될 수 있습니다.

- Time series of temperature, rainfall, wind speed, air quality, and so on.

온도, 강우량, 풍속, 대기질 등의 시계열입니다.

We may want to forecast the weather several days ahead, or climate several decades ahead.

우리는 며칠 앞으로 날씨를 예측하거나, 수십 년 앞으로 기후를 예측하고 싶을 수도 있습니다.

- Financial time series, where we track market indices, trading volumes, stock and bond prices, and exchange rates.

시장 지수, 거래량, 주식 및 채권 가격, 환율을 추적하는 금융 시계열입니다.

Here prediction is often difficult, but as we will see, certain indices can be predicted with reasonable accuracy.

여기서 예측은 종종 어렵지만, 우리가 보게 될 것처럼, 특정 지수는 합리적인 정확도로 예측할 수 있습니다.

- Recorded speech, musical recordings, and other sound recordings.

녹음된 음성, 음악 녹음 및 기타 사운드 녹음.

We may want to give a text transcription of a speech, or perhaps a language translation.

우리는 아마도 연설의 텍스트 녹음이나 언어 번역을 제공하고 싶을 것입니다.

We may want to assess the quality of a piece of music, or assign certain attributes.

우리는 음악의 질을 평가하거나 특정 속성을 할당하고 싶을 수도 있습니다.

- Handwriting, such as doctor’s notes, and handwritten digits such as zip codes.

의사 소견서와 같은 필기 및 우편 번호와 같은 필기 숫자.

Here we want to turn the handwriting into digital text, or read the digits (optical character recognition).

여기서는 필기를 디지털 텍스트로 변환하거나 숫자(광학 문자 인식)를 읽으려고 합니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

In a recurrent neural network (RNN), the input object **X** is a sequence.

순환(반복) 신경망(RNN)에서 입력 객체 **X**는 시퀀스입니다.

Consider a corpus of documents, such as the collection of **IMDb** movie reviews.

**IMDB** 영화 리뷰 모음과 같은 문서 말뭉치를 고려합니다.

\* IMDB: 미국의 영화 정보 모음 사이트

Each document can be represented as a sequence of **L** words, so **X = {X1, X2, . . . , XL}**, where each **Xl** represents a word.

각 문서는 **L** 개의 단어의 시퀀스로 표시될 수 있으므로 **X = {X1, X2, ., XL}**, 여기서 각 **Xl**은 단어를 나타냅니다.

The order of the words, and closeness of certain words in a sentence, convey semantic meaning.

단어의 순서와 문장에서 특정 단어의 근접성은 의미있는 뜻를 전달합니다.

**RNN**s are designed to accommodate and take advantage of the sequential nature of such input objects, much like convolutional neural networks accommodate the spatial structure of image inputs.

**RNN**은 컨볼루션 신경망이 이미지 입력의 공간 구조를 수용하는 것과 마찬가지로, 이러한 입력 객체의 순차적 특성을 수용하고 활용하도록 설계되었습니다.

The output **Y** can also be a sequence (such as in language translation), but often is a scalar, like the binary sentiment label of a movie review document.

출력 **Y**는 시퀀스(언어 번역 등)일 수도 있지만, 종종 영화 리뷰(검토) 문서의 이진 감성 레이블과 같은 스칼라입니다.

Figure 10.12 illustrates the structure of a very basic **RNN** with a sequence **X = {X1,X2,...,XL}** as input, a simple output **Y,** and a hidden-layer sequence **{Al}L1 = {A1, A2, . . . , AL}**.

그림 10.12는 시퀀스 **X = {X1,X2,...,XL}**을 입력으로 하고, 간단한 출력 **Y**를 가지며, 은닉층 시퀀스 **{Al}L1 = {A1,A2, ., AL}**을 갖는 매우 기본적인 **RNN**의 구조를 보여줍니다.

Each **Xl** is a vector; in the document example **Xl** could represent a one-hot encoding for the **l**th word based on the language dictionary for the corpus (see the top panel in Figure 10.13 for a simple example).

각 **Xl**은 벡터입니다; 문서 예제에서 **Xl**은 말뭉치를 위한 언어 사전을 기반으로 **l**번째 단어에 대한 원핫 인코딩을 나타낼 수 있습니다(단순한 예는 그림 10.13의 상단 패널 참조).

\* 원-핫 인코딩은 단어 집합의 크기를 벡터의 차원으로 하고, 표현하고 싶은 단어의 인덱스에 1의 값을 부여하고, 다른 인덱스에는 0을 부여하는 단어의 벡터 표현 방식입니다.

As the sequence is processed one vector **Xl** at a time, the network updates the activations **Al** in the hidden layer, taking as input the vector **Xl** and the activation vector **Al−1** from the previous step in the sequence.

시퀀스가 한 번에 하나의 벡터 **Xl**로 처리됨에 따라, 네트워크는 시퀀스의 이전 단계에서 벡터 **Xl**과 활성화 벡터 **Al-1**을 입력하여 은닉층의 활성화 **Al**을 업데이트합니다.

Each **Al** feeds into the output layer and produces a prediction **Ol** for **Y**.

각 **Al**은 출력층으로 공급되고 **Y**에 대한 예측 **Ol**을 생성합니다.

**OL**, the last of these, is the most relevant.

이 중 마지막 **OL**이 가장 관련성이 높습니다.

In detail, suppose each vector **Xl** of the input sequence has **p** components **XlT = (Xl1, Xl2, . . . , Xlp)**, and the hidden layer consists of **K** units **ATl = (Al1, Al2, . . . , AlK )**.

자세한 내용은 입력 시퀀스의 각 벡터 **Xl**이 **p** 성분 **XlT = (Xl1, Xl2, ., Xlp)**이고 은닉층이 **K** 단위 **ATl = (Al1, Al2, ., AlK)**로 구성되어 있다고 가정합니다.

As in Figure 10.4, we represent the collection of **K × (p+1)** shared weights **wkj** for the input layer by a matrix **W**, and similarly **U** is a **K × K** matrix of the weights **uks** for the hidden-to-hidden layers, and **B** is a **K + 1** vector of weights **βk** for the output layer.

그림 10.4에서와 같이, 우리는 입력 계층에 대한 **K × (p+1)** 공유 가중치 **wkj**의 집합을 행렬 **W**로 나타내며, 마찬가지로 **U**는 은닉층->은닉층 대한 가중치 **uks**의 **K × K** 행렬이고, **B**는 출력 계층에 대한 가중치 **βk**의 **K + 1** 벡터입니다.

Then 그리고나서

폰트, 텍스트, 화이트, 친필이(가) 표시된 사진

자동 생성된 설명

and the output **Ol** is computed as (10.16)과 출력 **Ol**은 다음과 같이 계산됩니다

텍스트, 폰트, 화이트, 친필이(가) 표시된 사진

자동 생성된 설명

for a quantitative response, or with an additional sigmoid activation function for a binary response, for example.

예를 들어 정량적 응답의 경우, 이진 반응에 대한 추가적인 시그모이드 활성화 함수를 사용할 수 있습니다.

Here **g(·)** is an activation function such as **ReLU**.

여기서 **g(·)**는 **ReLU**와 같은 활성화 기능입니다.

Notice that the same weights **W**, **U** and **B** are used as we process each element in the sequence, i.e. they are not functions of **l**.

시퀀스의 각 요소를 처리할 때 동일한 가중치 **W**, **U** 및 **B**가 사용됩니다. 즉, 그것들은 **l**의 함수가 아닙니다.

This is a form of weight sharing used by **RNN**s, and similar to the use of filters in convolutional neural networks (Section 10.3.1).

이것은 **RNN**에 의해 사용되는 가중치 공유의 한 형태이며, 컨볼루션 신경망에서의 필터의 사용과 유사합니다(섹션 10.3.1).

As we proceed from beginning to end, the activations **Al** accumulate a history of what has been seen before, so that the learned context can be used for prediction.

처음부터 끝까지 진행함에 따라 활성화 **Al**은 이전에 본 것의 이력을 축적하여 학습된 문자를 예측에 사용할 수 있습니다.

For regression problems the loss function for an observation (**X, Y**) is

회귀 문제의 경우 관측치(**X, Y**)에 대한 손실 함수는 다음과 같습니다

폰트, 타이포그래피, 화이트, 텍스트이(가) 표시된 사진

자동 생성된 설명

which only references the final output **OL = β0+)Kk=1 βkALk**. (10.17 위의 식 참고)

이는 최종 출력 **OL = β0+)Kk = 1 βALK**만 참조합니다.

Thus **O1, O2, . . . , OL−1** are not used.

따라서 **O1, O2, . . OL-1**은 사용되지 않습니다.

When we fit the model, each element **Xl** of the input sequence **X** contributes to **OL** via the chain (10.16), and hence contributes indirectly to learning the shared parameters **W**, **U** and **B** via the loss (10.18).

우리가 모델을 맞출 때, 입력 시퀀스 **X**의 각 요소 **Xl**은 체인(10.16)을 통해 **OL**에 기여하므로, 손실(10.18)을 통해 공유 매개 변수 **W**, **U** 및 **B**를 학습하는 데 간접적으로 기여합니다.

With **n** input sequence/response pairs (**xi,yi**), the parameters are found by minimizing the sum of squares (10.19).

**n**개의 입력 시퀀스/반응 쌍(**xi,yi**)을 사용하면 제곱합을 최소화하여 모수를 찾을 수 있습니다

폰트, 텍스트, 화이트, 친필이(가) 표시된 사진

자동 생성된 설명

Here we use lowercase letters for the observed **yi** and vector sequences **xi = {xi1, xi2, . . . , xiL}**, as well as the derived activations.

여기서 우리는 관찰된 **yi** 및 벡터 시퀀스 **xi = {xi1, xi2, ., xiL}**에 대해 소문자를 사용하고 유도된 활성화도 사용합니다.

Since the intermediate outputs **Ol** are not used, one may well ask why they are there at all.

중간 출력 **Ol**은 사용되지 않기 때문에, 왜 그것들이 거기에 있는지 물어볼 수 있습니다.

First of all, they come for free, since they use the same output weights **B** needed to produce **OL**, and provide an evolving prediction for the output.

우선, **OL**을 생성하는 데 필요한 동일한 출력 가중치 **B**를 사용하고 출력에 대한 진화하는 예측을 제공하기 때문에 무료로 제공됩니다.

Furthermore, for some learning tasks the response is also a sequence, and so the output sequence **{O1, O2, . . . , OL}** is explicitly needed.

또한, 일부 학습 과제의 경우 응답도 시퀀스이므로 출력 시퀀스 **{O1, O2, ., OL}**이(가) 명시적으로 필요합니다.

When used at full strength, recurrent neural networks can be quite complex.

최대 강도로 사용할 경우, 반복 신경망은 상당히 복잡할 수 있습니다.

We illustrate their use in two simple applications.

우리는 두 가지 간단한 응용 프로그램에서 이러한 응용 프로그램의 사용을 설명합니다.

In the first, we continue with the **IMDb** sentiment analysis of the previous section, where we process the words in the reviews sequentially.

첫 번째로, 우리는 리뷰의 단어를 순차적으로 처리하는 이전 섹션의 **IMDB** 감정 분석을 계속합니다.

In the second application, we illustrate their use in a financial time series forecasting problem.

두 번째 응용 프로그램에서, 우리는 재무 시계열 예측 문제에서 그것들의 사용을 설명합니다.