

기계학습

3.6강. 앙상블

김용대¹

서울대학교 통계학과¹



목차

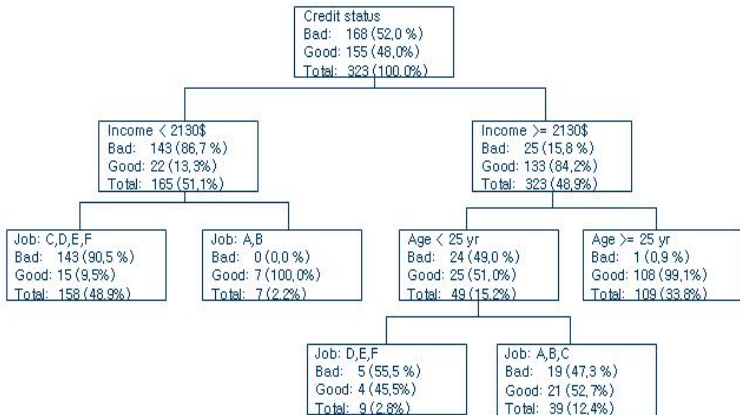
- 1절: 의사결정나무
- 2절: 배깅
- 3절: 랜덤포레스트
- 4절: 부스팅

1절: 의사결정나무

의사결정나무

- Decision tree
- 회귀분석이나 분류분석을 위한 모형
- "if-then"으로 구성된 규칙 생성
- SQL같은 DB언어로 규칙이 쉽게 표현
- 좋은 해석력

예제



의사결정나무 구축을 위한 4단계

partial Correlation
???

- 1 성장: 반복적으로 데이터를 분할하여 나무를 성장시킨다.
정지규칙을 만족하면 성장을 정지한다.
- 2 가지치기: 너무 큰 나무는 과적합으로 인하여 예측력이
저하되기 때문에 성장한 나무를 적당한 크기로 줄인다.
Validation표본이나 교차확인방법을 이용한다.
- 3 확인: 구축된 의사결정나무의 예측력을 확인하다 (예:gain
chart, lift chart)
- 4 해석 및 예측

의사결정나무의 장점

- 규칙이 이해가 쉽다
- 연속변수 및 범주형변수도 쉽게 처리된다.
- 중요한 변수를 찾아준다.
- 입력변수의 이상치에 강건하다 (robust).
- 모형화 작업이 필요없다.

의사결정나무의 장점

- 규칙이 이해가 쉽다
- 연속변수 및 범주형변수도 쉽게 처리된다.
- 중요한 변수를 찾아준다.
- **입력변수의 이상치에 강건하다 (robust).**
- 모형화 작업이 필요없다.

의사결정나무의 단점

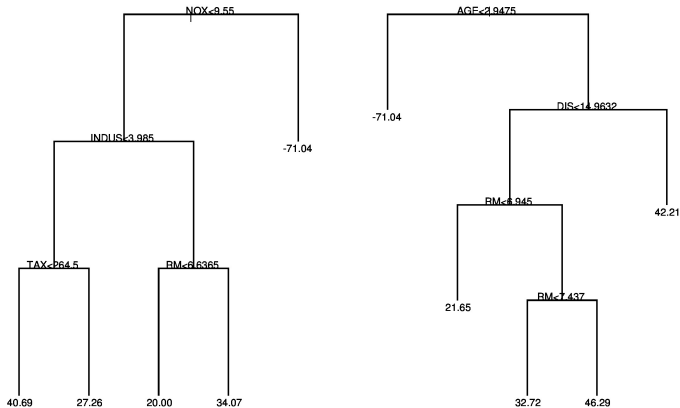
- 회귀모형에서 예측력이 떨어진다
- 나무의 크기가 크면 해석력이 떨어진다.
- 불안정하다.
- 주효과를 쉽게 파악할 수 없다.
- 비사각영역의 추정이 어렵다.

의사결정나무의 단점

- 회귀모형에서 예측력이 떨어진다
- 나무의 크기가 크면 해석력이 떨어진다.
- 불안정하다.
- 주효과를 쉽게 파악할 수 없다.
- 비사각영역의 추정이 어렵다.

의사결정나무의 불안정성

- 두개의 bootstrap 자료로 부터 만든 두개의 의사결정나무



의사결정나무의 불안정성

- 두 나무에서 쓰인 변수가 거의 다르다 (거의 마지막에 쓰인 'RM'만 예외).
- 이런 이유는, 첫번째 분할에 쓰인 변수가 달라지면 완전 다른 나무가 구축되기 때문이다.
- 두개의 변수가 비슷하게 출력변수를 설명하는 경우 이런 현상이 자주 생긴다.
- 의사결정나무의 불안정성을 해결하기 위하여 개발된 방법이 앙상블(Ensemble)이다.

2절: 배깅 (Bagging)

앙상블 소개

- 앙상블이란 하나의 자료에 대해서 여러 개의 예측모형을 만든 후, 이를 결합하여 최종예측모형을 만드는 방법을 통칭한다.
- 앙상블 방법의 예
 - Bagging (Breiman 1996)
 - Boosting (Freund and Schapire 1997)
 - Random Forest (Breiman 2004)
- 실증적으로 앙상블 방법이 의사결정나무 보다 훨씬 좋은 예측력을 갖는 것이 밝혀졌다.

배깅 알고리즘

- **Bootstrap aggregating**

- \mathcal{L} : 학습자료

- 알고리즘

- 1 B 개의 붓스트랩 표본 $\{\mathcal{L}^{(b)}, b = 1, \dots, B\}$ 을 만든다 (data sets obtained by with replacement sampling from \mathcal{L}).
- 2 각각의 붓스트랩 표본에 대해서 예측모형 $\{f(\mathbf{x}, \mathcal{L}^{(b)}), b = 1, \dots, B\}$ 을 구축한다.
- 3 y 가 연속형변수이면 평균예측모형 $f_B(\mathbf{x}) = \text{av}_b f(\mathbf{x}, \mathcal{L}^{(b)})$ 를 사용하고
- 4 y 가 범주형이면 (class label), majority vote 방법을 이용하여 다음과 같이 배깅예측모형을 만든다: $f_B(\mathbf{x}) = \text{argmax}_j N_j$ where $N_j = \#\{b : f(\mathbf{x}, \mathcal{L}^{(b)}) = j\}$.

예측력 비교

- Bagging Classification trees

| Data Set | Samples | Variables | Classes | \bar{e}_S | \bar{e}_B | Decrease |
|-----------------|----------------|------------------|----------------|-------------|-------------|-----------------|
| waveform | 300 | 21 | 3 | 29.0 | 19.4 | 33% |
| heart | 1395 | 16 | 2 | 10.0 | 5.3 | 47% |
| breast cancer | 699 | 9 | 2 | 6.0 | 4.2 | 30% |
| ionosphere | 351 | 34 | 2 | 11.2 | 8.6 | 23% |
| diabetes | 1036 | 8 | 2 | 23.4 | 18.8 | 20% |
| glass | 214 | 9 | 6 | 32.0 | 24.9 | 22% |
| soybean | 307 | 35 | 19 | 14.5 | 10.6 | 27% |

예측력 비교

- Bagging Regression trees

| Data Set | \bar{e}_S | \bar{e}_B | Decrease |
|-----------------|-------------|-------------|-----------------|
| Boston Housing | 19.1 | 11.7 | 39% |
| Ozone | 23.1 | 18.0 | 22% |
| Friedman # 1 | 11.4 | 6.2 | 46% |
| Friedman # 2 | 30,800 | 21,700 | 30% |
| Friedman # 3 | .0403 | .0249 | 38% |

배깅의 원리

- 배깅의 원리는 불안정한 예측모형의 분산을 평균을 사용해서 줄이는 것이다.
- 따라서, 배깅에 의해서 예측력이 향상되는 예측모형은 분산은 크고 편이는 작아야 한다.
- 즉, 일부러 과적합된 예측모형이 배깅을 적용하였을 때 큰 효과를 볼 수 있다.
- 이 원리를 의사결정나무에 적용하면, 가지치기를 하지 않은 나무 모형이 배깅하고 가장 잘 어울린다.
- 의사결정나무 구축에서 가장 시간이 많이 드는 부분이 가지치기인데, 그 이유는 모형선택(얼마나 많은 가지를 쳐낼 것인지를 결정)이 필요하기 때문이다.
- 즉, 배깅은 최적의 의사결정나무 구축보다 계산속도가 빠르다!

3절: 랜덤 포레스트 (Random Forest)

서론

- 랜덤포레스트는 여러개의 의사결정나무를 랜덤하게 만든 후, 이를 결합하여 최종 예측모형을 만드는 방법이다.
- 배깅은 랜덤포레스트의 일종인데, 랜덤포레스트는 배깅보다 더 많은 무작위성을 사용한다.
- 랜덤포레스트의 장점은
 - 예측력이 배깅보다 좋으며
 - 이상치에 강건하고
 - 계산속도가 상대적으로 빠르며
 - 사용하기 쉽다. (초보자도 쉽게 사용 가능)

알고리즘

- RF1: 나무를 성장시킬 때, 각 노드에서 변수를 임의로 뽑아서 사용한다.
- RF2: 각 노드에서 k 개의 변수를 임의로 뽑고, 이 변수들 중 가장 불순도를 크게 감소시키는 변수로 나무를 성장시킨다.
- RF-L
 - L 개의 변수를 임의로 뽑는다.
 - L 개의 변수를 이용하여 F 개의 선형결합을 임의로 만든다 (가중치를 임의로 정한다).
 - F 개의 새로운 변수중 가장 불순도를 크게 감소시키는 변수로 나무를 성장시킨다.
- 가지치기는 사용하지 않고, 부스트랩 표본을 사용한다.

예측력비교

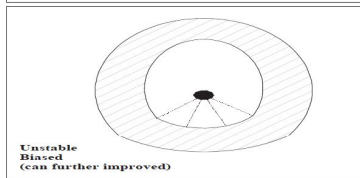
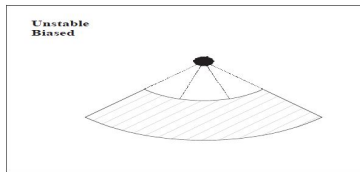
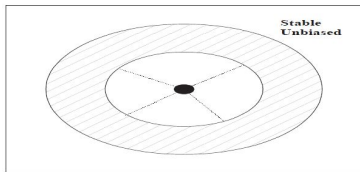
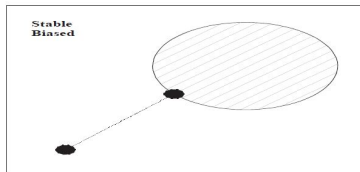
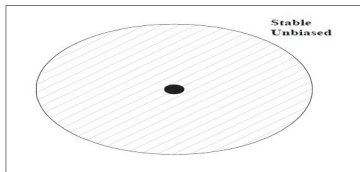
| Data Set | Single | Bagging | AdaBoost | RF1 | RF1-L |
|-----------------|--------|---------|----------|------|-------|
| waveform | 29.0 | 19.4 | 18.2 | 17.2 | 16.1 |
| breast cancer | 6.0 | 5.3 | 3.2 | 2.9 | 2.9 |
| ionosphere | 11.2 | 8.6 | 5.9 | 7.1 | 5.7 |
| diabetes | 23.4 | 18.8 | 20.2 | 24.2 | 23.1 |
| glass | 32.0 | 24.9 | 22.0 | 20.6 | 23.5 |

이상치 강건성

- 5%의 자료를 임의로 뽑아서 속한 그룹을 임의로 바꾼다.
- 오차율의 증가량 (%)

| Data Set | AdaBoost | RF1 |
|-----------------|-----------------|------------|
| breast cancer | 43.2 | 1.8 |
| ionosphere | 27.7 | 3.8 |
| diabetes | 6.8 | 1.8 |
| glass | 1.6 | 0.4 |

기하학적 설명



Remark

- 조율모수가 전혀 없어서 초보자도 쉽게 사용할 수 있다.
- 계산면에서도 RF는 매우 효율적인데, 완벽한 분산처리가 가능하기 때문이다.

4절:부스팅 (Boosting)

서론

- 부스팅은 약한 예측모형 (random guess보다 조금 좋은 모형) 여러개를 결합하여 매우 정확한 예측모형을 만드는 방법
- Freund and Schapire (1997)가 처음으로 실제 자료분석에 쓸 수 있는 부스팅알고리즘을 개발하였으며 "AdaBoost (Adaptive Boost)"로 명명함.
- 그 이후로 다양한 종류의 부스팅 알고리즘이 개발됨.
- 본 장에서는 다음의 내용을 다룸
 - ① AdaBoost 알고리즘
 - ② 알고리즘에 대한 통계학적인 이해
 - ③ 로지스틱회귀모형을 위한 부스팅: Gradient 부스팅
 - ④ 부스팅과 라쏘의 관계

예제

- 경마장에 100명의 자칭 전문가들이 있다고 하자.
- 전문가들은 많은 시간과 비용을 사용했기 때문에, 일반인 보다는 예측의 정확도가 높다. 그러나 많이 높지는 않다.
- 이제 문제는 “이 100명 전문가의 예측결과를 합쳐서 매우 정확한 예측을 할 수는 없을까?”이다.
- 놀랍게도 이것이 가능하다는 것이 밝혀졌으며, 이 이론을 *weak learnability*로 부른다.
- AdaBoost는 weak learnability를 실제 자료분석을 위하여 구현한 최초의 알고리즘이다.

예측력 비교

| Data Set | Single | AdaBoost | Decrease | Bagging |
|-----------------|---------------|-----------------|-----------------|----------------|
| waveform | 29.0 | 18.2 | 37% | 19.4 |
| breast cancer | 6.0 | 3.2 | 47% | 5.3 |
| ionosphere | 11.2 | 5.9 | 47% | 8.6 |
| diabetes | 23.4 | 20.2 | 14% | 18.8 |
| glass | 32.0 | 22.0 | 31% | 24.9 |

AdaBoost의 통계학적 이해

- M-추정방법론의 적용
- 예측모형구축의 목적은 최적의 예측모형

$$f^0 = \arg \min_{f \in \mathcal{F}} \mathbb{E}_{(Y, \mathbf{X})} l(Y, f(\mathbf{X}))$$

를 구축하는 것이다.

- M-추정방법은 f^0 를 \hat{f} 로 추정한다:

$$\hat{f} = \arg \min_{f \in \mathcal{F}} \sum_{i=1}^n l(y_i, f(\mathbf{x}_i)) / n.$$

- 즉, empirical risk를 최소화 하는 예측모형을 찾는다.

= , MLE, - log likelihood, logistic loss

Steepest descent 알고리즘으로의 AdaBoost

weak learner (=base learner,)

- \mathcal{H} 를 기저예측모형의 집합이라 하자.
- AdaBoost는 exponential 손실함수 $l(y, f(\mathbf{x})) = \exp(-yf(\mathbf{x}))$, where $y \in \{-1, 1\}$ 를 이용한 M-추정방법으로 이해할 수 있다.
- $F_m = \sum_{k=1}^m c_k f_k$ 을 현재의 앙상블모형이라 하자.
- 그러면, AdaBoost는

$$(c_{m+1}, f_{m+1}) = \operatorname{argmin}_{c \in [0, \infty), f \in \mathcal{H}} \sum_{i=1}^n \exp[-y_i \{F_m(\mathbf{x}_i) + cf(\mathbf{x}_i)\}]$$

인 (c_{m+1}, f_{m+1}) 를 구한 후, 앙상블 모형을
 $F_{m+1} = F_m + c_{m+1}f_{m+1}$ 로 갱신한다.

- Schapire and Singer (1999)는 이러한 steepest descent 방법이 정확히 AdaBoost알고리즘과 같다는 것을 증명하였다.

Remark on exponential 손실함수



Fisher consistent

~ ~ ~

•

$$f^*(\mathbf{x}) = \operatorname{argmin}_f \mathbb{E}_{(Y, \mathbf{X})} \exp(-Yf(\mathbf{X}))$$

라 하면

$$f^*(\mathbf{x}) = \frac{1}{2} \log p(\mathbf{x}) / (1 - p(\mathbf{x}))$$

exp adaboost
logistic regression

이고 $p(\mathbf{x}) = \Pr(Y = 1 | \mathbf{X} = x)$ 임이 증명가능하다.

- 즉, 로지스틱회귀모형과 유사하다.

| | |
|------------------------------|---|
| <p>??</p> <p>= *1.5</p> | <p>yc average(2</p> <p>... 5%</p> <p>)</p> |
|------------------------------|---|

Gradient 부스팅 (Friedman 2001)

- Friedman (2001)은 AdaBoost 알고리즘을 다양한 손실함수로 확장하였다.
- 하지만, exponential 손실함수 이외에는 steepest descent 방법을 적용할 수 없다.
- Friedman (2001)은 functional gradient descent 방법을 사용하여 다양한 부스팅 알고리즘을 개발하였다.

Functional gradient descent 알고리즘

- F_{m-1} 를 현재의 앙상블모형이라 하자.
- $C(F) = \sum_{i=1}^n l(y_i, F(\mathbf{x}_i))$ 는 경험위험함수 (Empirical risk)이다.
- 경험위험함수를 현재 앙상블 모형 F_{m-1} 에서 얻은 gradient를 g_m 일 하자:

$$g_m(\mathbf{x}) = \left[\frac{\partial C(F)}{\partial F(\mathbf{x})} \right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}.$$

- 상수 ρ_m 을 다음과 같이 구한다:

$$\rho_m = \operatorname{argmin}_{\rho \in \mathbb{R}} C(F_{m-1} + \rho g_m).$$

- 앙상블 모형을 $F_m = F_{m-1} + \rho g_m$ 로 갱신한다.

Gradient 부스팅 알고리즘

- 하지만, g_m 은 대부분 기저예측모형 \mathcal{H} 에 포함되지 않는다.
- g_m 대신 기저위함수 중에서 g_m 과 가장 가까운 모형 f_m 을 사용한다:

$$f_m = \operatorname{argmin}_{f \in \mathcal{H}} \sum_{i=1}^n \{g_m(\mathbf{x}_i) - f(\mathbf{x}_i)\}^2.$$

- 양상블 모형을 $F_m = F_{m-1} + \rho f_m$ 로 갱신하다. 여기서, ρ 는 다음과 같이 구한다:

$$\rho_m = \operatorname{argmin}_{\rho \in R} C(F_{m-1} + \rho f_m).$$

로짓부스팅

- 로지스틱 손실함수 (로지스틱모형의 negative 로그 가능도함수)와 의사결정나무를 기저예측모형으로 사용한 gradient 부스팅 알고리즘
- 로지스틱 손실함수: $l(y, f) = \log\{1 + \exp(-2yf)\}$ for $y \in \{-1, 1\}$.
- Gradient:

$$g_m(\mathbf{x}_i) = \frac{2y_i \exp(-2y_i F_{m-1}(\mathbf{x}_i))}{1 + \exp(2y_i F_{m-1}(\mathbf{x}_i))}.$$

- Gradient값을 출력변수로 만든 가상자료 $\{(\mathbf{x}_i, g_m(\mathbf{x}_i)), i = 1, \dots, n\}$ 를 이용하여 L 개의 마지막 노드를 갖는 의사결정나무 T_m 을 구축
- 앙상블모형을 $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + T_m(\mathbf{x})$ 으로 갱신.

Regularization

- 부스팅 알고리즘은 경험위험함수를 가장 작게 하는 기저예측모형의 선형결합을 찾는 알고리즘으로 이해할 수 있다.
- 기저예측모형은 단순해도 기저예측모형의 선형결합은 매우 복잡한 함수도 근사시킬 수 있다.
- 예를 들면, $p + 1$ 개의 마지막 노드를 갖는 의사결정나무는 p 차원의 어떤 함수도 근사시킬 수 있음을 Breiman (2000)에 증명하였다.
- M-추정방법은 고려하는 모형이 복잡하지 않을 때만 좋은 예측력을 보인다 (과적합문제).
- 부스팅 알고리즘도 과적합 문제로 인하여 예측력이 나쁠 것으로 예상할 수 있다.
- 부스팅의 예측력을 향상시키기 위해서는 적절한 regularization 이 필요하다.

축소 (Shrinkage, Frieman 2001)

- $F_m = F_{m-1} + T_m$ 으로 갱신하는 대신, 적당한 작은 상수 $\gamma \in (0, 1)$ 에 대해서 $F_m = F_{m-1} + \gamma T_m$ 으로 양상블 모형을 갱신한다.
- Friedman (2001)은 $\gamma = 0.1$ 일때 예측력이 좋다는 것을 실증적으로 보였다.
- 이 방법은 축소추정량과 밀접한 관련이 있다.
- 이해하기 쉽지 않은 부분은 축소모수 γ 를 작게 할 수록 예측력이 계속 좋아진다는 것이다.
- 놀랍게도, 이러한 이상한 현상을 LASSO로 설명할 수 있다!

부스팅 알고리즘과 고차원 회귀모형

- 주어진 학습자료 \mathcal{L} 에 대해서 유한개의 기저예측모형만이 다른 값을 가질 수 있다 (예: 학습자료에서 서로 다른 값을 갖는 의사결정나무는 유한개다)
- 즉, $\mathcal{H} = \{T_1, \dots, T_q\}$ 라 할 수 있는데, 여기서 기저예측모형의 개수 q 는 매우 크기는 하지만 유한이다.
- 로짓부스팅은 다음의 고차원 로지스틱 회귀모형에서 회귀계수를 추정하는 문제로 이해할 수 있다:

$$\log \frac{\Pr(Y = 1|\mathbf{x})}{\Pr(Y = -1|\mathbf{x})} = \beta_0 + \sum_{j=1}^q \beta_j T_j(\mathbf{x}).$$

부스팅 알고리즘과 고차원 회귀모형 (계속)

- 즉, 로짓부스팅은 다음의 두 단계로 구성되어 있다:
 - ① 입력변수를 의사결정나무를 이용하여 새로운 변수로 변환한다,
 - ② 변환된 입력변수들의 선형 로지스틱 회귀모형을 고려하고, 회귀계수를 추정한다.
- 이러한 면에서, 로짓부스팅은 다항회귀와 유사하다고 볼 수 있다:

$$\log \frac{\Pr(Y = 1|\mathbf{x})}{\Pr(Y = -1|\mathbf{x})} = \beta_0 + \sum_{j=1}^p \beta_j x_j + \sum_{j,k} \beta_{jk} x_j x_k + \sum_{j,k,l} \beta_{jkl} x_j x_k x_l + \cdots$$

- 의사결정나무를 이용하여 입력변수를 변환하는 방법의 큰 장점은 입력변수의 잡음에 강건하다는 것이다!

부스팅과 LASSO

- Regularized 부스팅 알고리즘에서 ϵ 을 0으로 보내면, LASSO 알고리즘과 같아진다.
- 즉, Regularized gradient 부스팅알고리즘은 의사결정나무로 변환된 입력변수를 사용하여 로지스틱 모형의 회귀계수를 LASSO로 구한 알고리즘이라고 이해할 수 있다.

일반화 가법 모형

- 일반화 가법 모형 (Generalized additive model, GAM)은 구조화된 비모수함수모형이다.
- 선형모형 = 가
 - $Y = \beta_0 + \sum_{i=1}^p \beta_i x_i + \epsilon.$
- 일반화 가법 모형
 - $Y = \beta_0 + \sum_{j=1}^p f_j(x_j) + \epsilon.$
- 예제
 - $Y = \exp(-1 + 2x_1) + \sin(2\pi x_2) + \epsilon.$

부스팅과 일반화 가법 모형

- 기저예측모형 \mathcal{H} 가 노드가 두개인 의사결정나무로 구성되어 있다고 하자.
- Gradient 부스팅을 통하여 구축한 예측모형은 다음과 같이 쓸 수 있다:

$$F(\mathbf{x}) = \beta_0 + \sum_{T_m \in \mathcal{H}} \beta_m T_m(\mathbf{x}).$$

- $v(m)$ 을 나무 T_m 에서 사용된 변수라고 하자.
- 그러면 앙상블 모형 F 는 다음과 같이 쓸 수 있다:
 $F(\mathbf{x}) = \beta_0 + \sum_{j=1}^p f_j(x_j)$ 이고

$$f_j(x_j) = \sum_{T_m \in \mathcal{H}} \beta_m T_m(\mathbf{x}) I(v(m) = j)$$

이다.

- 즉, 부스팅은 일반화 가법 모형에서 각 component 함수 f_j 를 의사결정나무의 선형결합으로 추정한 것이다.

교호작용 추정

- \mathcal{H} 가 3개의 노드를 갖는 의사결정나무로 구성되어 있다고 하자.
- 즉, 각 의사결정나무는 두개의 변수를 사용한다.
- 따라서, 앙상블 모형을 다음과 같이 쓸 수 있다:

$$F(\mathbf{x}) = \beta_0 + \sum_{j,k} f_{jk}(x_j, x_k), \text{ 이고}$$

$$f_{jk}(x_j, x_k) = \sum_{T_m \in \mathcal{H}} \beta_m T_m(\mathbf{x}) I(v(m) = \{j, k\}).$$

- 즉, 기저예측모형의 노드수를 결정하는 것은 앙상블모형의 교호작용의 차수를 결정하는 것이다.

해석

- 의사결정나무의 선형결합을 어떻게 해석할 수 있을까?
- 입력변수의 상대적 중요도 (Relative importance)와 부분의존도그림 (Partial dependency plot)를 이용

상대적 중요도

- 설명을 쉽게 하기 위하여, \mathcal{H} 가 노드수가 2개인 의사결정나무로 구성되어 있다고 하자.
- 주어진 나무 T 에 대해서 $s(T)$ 를 어미노드와 자식노드들 사이의 불순도 측정치의 차이로 하자 (불순도의 차이가 클수록 중요한 변수임).
- 변수 j 의 상대적 중요도는 다음과 같이 구한다:

$$RI_j = \sum_{m=1}^M s(T_m) I(v(m) = j).$$

부분의존도 그림

- 주어진 앙상블 모형 $F(\mathbf{x})$ 에서 j 번째 변수의 부분의존도 그림은 다음과 같이 정의된다:

$$f_j(x_j) = \frac{1}{n} \sum_{i=1}^n F(x_{i1}, \dots, x_{i(j-1)}, x_j, x_{i(j+1)}, \dots, x_{ip}).$$

예제

Example

- Experiment settings

- J = 6 terminal nodes
- Learning rate: $v = 0.1$

- Data set: California Housing

- Response Variable: Median house value
- Eight Predictor Variables:
 - ◆ *median Income (MedInc), housing density (Population), average occupancy (AveOccup), location (Longitude and Latitude), average number of rooms (AveRooms) and bedrooms (AveBedrms), age of the house (HouseAge)*

- Results

- Training and test error
 - Relative importance and partial dependence plots
-

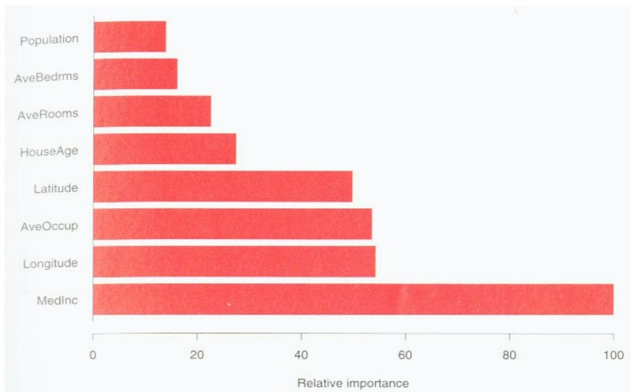
예제

Training and Test Error



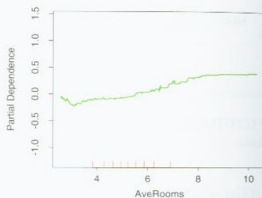
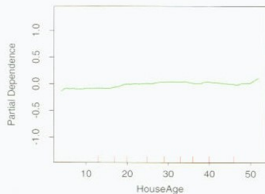
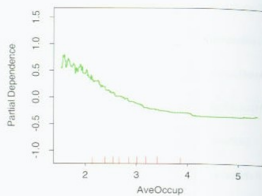
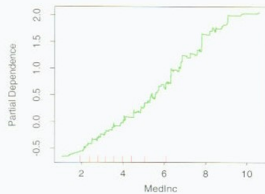
예제

Relative Importance of Predictor Variables



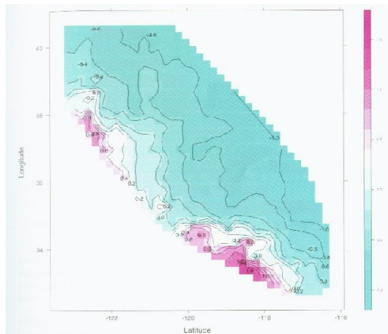
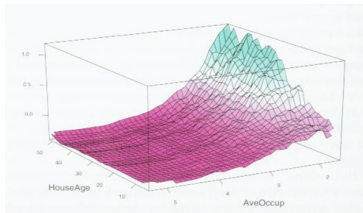
예제

Partial Dependence Plots (1-dim)



예제

Partial Dependence Plots (2-dim)



Thank you!!