▣ **강의명:** CS109A: 데이터 과학 입문

▣ **주차:** Lecture 25

▣ **교수명:** Pavlos Protopapas, Kevin Rader, Chris Gumb

▣ **목적:** Lecture 25의 핵심 개념 학습

Contents

CS109A 수업 노트: 앙상블 학습의 확장

(Blending, Stacking, and Mixture of Experts)

1 개요: 왜 더 복잡한 앙상블이 필요한가?

이 문서는 기존의 랜덤 포레스트나 부스팅(Bagging/Boosting)을 넘어, 서로 다른 종류의 모델들을 결합하여 성능을 극대화하는 고급 앙상블 기법을 다룹니다.

□ 핵심 요약

핵심 목표 및 요약

- **한계 극복:** 단일 모델(Decision Tree 등)이나 동일한 모델의 집합(Random Forest)만으로는 해결하기 어려운 복잡한 데이터 패턴을 학습합니다.
- **이종 결합:** 선형 회귀, KNN, SVM 등 성격이 완전히 다른 모델들을 하나로 합쳐 각 모델의 장점만 취합니다.
- **메타 학습:** 모델들의 예측값(Output)을 다시 학습 데이터로 사용하여 최종 결론을 내리는 '관리자 모델(Meta-model)'을 도입합니다.

2 필수 용어 정리

본격적인 학습 전에 아래 용어를 숙지하면 이해가 훨씬 빠릅니다.

용어 (한글/영문)	설명	비고
기반 모델 (Base Model)	1차적으로 데이터를 학습하여 예측을 수행하는 개별 모델들	전문가 팀원
메타 모델 (Meta Model)	기반 모델들의 예측값을 입력받아 최종 판단을 내리는 모델	팀장/관리자
동질성 (Homogeneous)	같은 종류의 알고리즘(예: 트리)만 모아서 앙상블 하는 것	Random Forest 등
이질성 (Heterogeneous)	서로 다른 알고리즘(예: 선형회귀+KNN)을 섞어서 사용하는 것	Blending/Stacking
홀드아웃 세트 (Holdout Set)	메타 모델을 학습시키기 위해 따로 떼어둔 검증용 데이터	커닝 방지용
게이팅 (Gating Network)	데이터의 특성에 따라 '어떤 전문가 모델'을 쓸지 결정하는 망	상담원/분류기

Table 1: 앙상블 심화 학습을 위한 핵심 용어

3 핵심 개념: 앙상블의 진화

우리가 이전에 배운 Bagging과 Boosting은 강력하지만, 주로 **'같은 종류의 모델(주로 결정 트리)'**을 사용하는 한계가 있습니다. Blending과 Stacking은 **'서로 다른 모델'**을 결합한다는 점에서 차이가 있습니다.

어벤져스 팀 구성하기

- **Bagging (Random Forest):** 100명의 의사가 모여서 진단하고 다수결로 결정합니다. (모두가 의사라는 점에서 동질적임)
- **Blending/Stacking:** 의사, 변호사, 회계사가 모입니다. 그리고 이들의 의견을 종합해서 최종 결정을 내리는 '판사(Meta Model)'가 있습니다. (서로 다른 직업군이 모여 이질적임)

구분	구성 모델	결합 방식	특징
Bagging	동질적 (주로 트리)	병렬 학습 → 평균/투표	분산(Variance) 감소, 과적합 방지
Boosting	동질적 (약한 모델)	순차 학습 → 가중치 합	편향(Bias) 감소, 오답 집중 학습
Blending	**이질적 (다양함)**	데이터 분할 → 메타 모델 학습	구현이 쉬우나 데이터 낭비 발생
Stacking	**이질적 (다양함)**	교차 검증(CV) → 메타 모델 학습	데이터 효율 높음, 연산 비용 큼

Table 2: 다양한 앙상블 기법 비교

4 Blending: 단순하고 직관적인 결합

4.1 기본 원리

Blending은 데이터를 여러 조각으로 나누어, 일부는 기반 모델(Base Model)을 학습시키고, 나머지는 그 모델들을 평가(예측)하여 메타 모델(Meta Model)을 학습시키는 방식입니다.

4.2 수행 절차 (Step-by-Step)

1. **데이터 분할:** 전체 데이터를 Train / Validation / Holdout / Test 세트로 나눕니다.
2. **기반 모델 학습:** Train 세트로 여러 모델(선형회귀, 결정트리, KNN 등)을 학습시킵니다.
3. **예측 생성 (1차):** 학습된 기반 모델들로 Validation 세트를 예측합니다.
4. **메타 모델 학습:**
   - 입력(X): 기반 모델들이 내놓은 예측값들 (필요시 원본 데이터도 포함 가능)
   - 정답(Y): Validation 세트의 실제 정답
   - 위 데이터를 사용해 메타 모델(보통 간단한 선형 모델)을 학습시킵니다.
5. **최종 평가:** Test 세트로 최종 성능을 확인합니다.

주의사항

Blending의 단점: 데이터 효율성 Blending은 Validation과 Holdout 세트를 따로 떼어놓아야 하므로, 실제로 모델 학습에 사용할 수 있는 데이터의 양이 줄어듭니다. 데이터가 아주 많다면 괜찮지만, 데이터가 적다면 성능 저하가 올 수 있습니다.

5 Stacking: 데이터 낭비 없는 정교한 결합

5.1 Blending과의 차이점

Stacking은 Blending과 비슷하지만, **교차 검증(Cross-Validation)**을 사용하여 데이터를 낭비하지 않고 모든 데이터를 학습에 활용합니다. 조금 더 복잡하지만 성능은 일반적으로 더 좋습니다.

5.2 수행 절차 (Step-by-Step)

1. **데이터 분할:** 전체 데이터를 Train과 Test로 나눕니다.
2. **교차 검증 (CV) 수행:** Train 데이터를 K개의 폴드(Fold)로 나눕니다 (예: 3-Fold).
3. **예측값 생성 (Out-of-Fold Prediction):**
   - Fold 1, 2로 학습하고 → Fold 3을 예측합니다.
   - Fold 1, 3으로 학습하고 → Fold 2를 예측합니다.
   - Fold 2, 3으로 학습하고 → Fold 1을 예측합니다.
   - 이렇게 하면 모든 Train 데이터에 대한 '예측값'을 얻을 수 있습니다.
4. **메타 데이터 생성:** 위에서 얻은 예측값들을 모아서 새로운 입력 데이터셋(X_{meta})을 만듭니다.
5. **메타 모델 학습:** X_{meta}와 실제 정답(Y)을 이용해 메타 모델을 학습시킵니다.
6. **최종 추론:** Test 데이터를 입력하면, 기반 모델들이 예측값을 내놓고, 이를 메타 모델이 받아 최종 결과를 출력합니다.

Stacking 과정 비유 팀원들이 돌아가며 모의고사를 봅니다.

- A가 시험 볼 때 B, C가 공부한 걸로 채점해주고,
- B가 시험 볼 때 A, C가 도와주는 식입니다.
- 결과적으로 팀원 모두의 모의고사 성적표(예측값)가 모이게 되고, 선생님(메타 모델)은 "철수는 수학을 잘하고 영희는 영어를 잘하네"라는 패턴을 학습하여 실전 수능(Test Set)에 대비합니다.

5.3 Passthrough (패스스루)

메타 모델을 학습시킬 때, 단순히 기반 모델의 예측값($\hat{y}$)만 쓰는 것이 아니라, **원본 데이터(X)도 함께 넣어주는 기법**입니다.

- **장점:** 메타 모델이 "어떤 데이터 상황에서 어떤 모델을 믿어야 할지" 더 잘 판단할 수 있습니다.
- **주의:** 데이터 차원이 커지고 복잡해질 수 있습니다.

6 Mixture of Experts (MoE)

6.1 개념: 협력이 아닌 분업

지금까지의 앙상블은 모든 모델이 힘을 합쳐(평균, 투표) 결과를 냈습니다. 하지만 MoE는 **"이 문제는 네가 전문가니까 네가 처리해"**라고 맡기는 방식입니다. 최근 대형 언어 모델(LLM)에서도 효율성을 위해 많이 사용되는 핵심 개념입니다.

6.2 구조: 전문가와 게이트

- **전문가 (Experts):** 특정 데이터 영역이나 패턴에 강한 모델들입니다. (예: $X < 0$일 때는 선형회귀 A, $X \geq 0$일 때는 선형회귀 B)
- **게이팅 네트워크 (Gating Network):** 입력 데이터(X)를 보고 어떤 전문가에게 일을 맡길지 결정하는 '분류기'입니다. 주로 Softmax 함수를 사용하여 각 전문가에 대한 가중치(확률)를 출력합니다.

$$\text{최종 예측 } \hat{y} = \sum_{i=1}^{K} g_i(x) \cdot f_i(x) \tag{1}$$

- $g_i(x)$: 게이팅 네트워크가 부여한 i번째 전문가의 가중치 (합은 1)
- $f_i(x)$: i번째 전문가의 예측값

병원 진료 시스템 환자(데이터)가 병원에 오면, 접수처 직원(Gating Network)이 증상을 듣고 정형외과(Expert 1)로 보낼지, 내과(Expert 2)로 보낼지 결정합니다. 모든 의사가 다 같이 진료하는 것이 아니라, 가장 적합한 의사가 주도적으로 치료합니다.

6.3 주요 이슈: 전문가 붕괴 (Expert Collapse)

학습 과정에서 특정 전문가 한 명만 너무 똑똑해지거나, 게이팅 네트워크가 한쪽으로만 데이터를 몰아주는 현상이 발생할 수 있습니다.

- **증상:** 모든 데이터가 Expert 1에게만 가고, 나머지 Expert들은 놉니다.
- **원인:** 초기에 성능이 조금이라도 좋은 쪽에 몰아주다 보니 빈익빈 부익부 현상이 발생함.
- **해결:** 학습 시 손실 함수(Loss Function)를 조정하여 전문가들이 골고루 학습되도록 유도해야 합니다.

7 구현 및 활용 가이드 (Python/Scikit-Learn)

Stacking은 'sklearn' 라이브러리를 통해 쉽게 구현할 수 있습니다. 아래는 개념적인 의사 코드(Pseudo-code) 흐름입니다.

Stacking Regressor 구현 흐름

```
from sklearn.ensemble import StackingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor

# 1. 기반 모델(Experts) 정의
estimators = [
    ('rf', RandomForestRegressor(n_estimators=10)),
    ('knn', KNeighborsRegressor(n_neighbors=5)),
    ('lr', LinearRegression())
]

# 2. 메타 모델(Manager) 정의 - 보통 단순한 모델 사용
meta_model = LinearRegression()

# 3. Stacking 모델 구성 (cv=5는 5-Fold 교차검증 의미)
# passthrough=True 옵션: 원본 데이터도 메타 모델에 전달
clf = StackingRegressor(
    estimators=estimators,
    final_estimator=meta_model,
    cv=5,
    passthrough=True
)

# 4. 학습 및 예측
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

7.1 언제 어떤 모델을 써야 할까? (의사결정)

- **데이터가 적고 빠른 결과가 필요하다:** → 단일 모델 또는 Random Forest
- **성능을 0.1%라도 더 올리고 싶다 (Kaggle 등):** → Stacking (다양한 모델 결합)
- **데이터의 특성이 영역별로 확연히 다르다:** → Mixture of Experts (또는 Tree 모델)
- **해석(Interpretation)이 중요하다:** → Stacking을 쓰되, 메타 모델로 선형 회귀를 써서 각 기반 모델의 가중치(계수)를 확인한다.

8 학습 점검 체크리스트 & FAQ

☐ **기반 모델의 다양성 확보:** 서로 비슷한 모델(예: 트리 3개)만 섞지 않았는가? (KNN, 선형, 트리를 섞는 것이 좋음)

☐ **데이터 누수(Leakage) 주의:** Stacking 시 Test 데이터를 학습 과정에 실수로 포함시키지 않았는가?

☐ **메타 모델의 단순성:** 메타 모델을 너무 복잡하게(예: 딥러닝) 만들어서 과적합되지 않았는가? (보통 선형 모델 권장)

☐ **전처리 일관성:** KNN 같은 거리 기반 모델을 섞을 때는 스케일링(StandardScaler)을 했는가? (트리는 필요 없지만 KNN은 필수)

8.1 자주 묻는 질문 (FAQ)

Q1. Stacking이 Random Forest보다 무조건 좋은가요?
A. 아닙니다. Stacking은 연산량이 훨씬 많고 복잡합니다. 데이터에 따라 Random Forest 같은 단일 앙상블 기법이 더 효율적일 수 있습니다. Stacking은 '마지막 성능 쥐어짜기' 용도로 주로 쓰입니다.

Q2. Passthrough는 언제 켜야 하나요?
A. 기반 모델들이 데이터의 모든 정보를 다 캡처하지 못했다고 판단될 때 켭니다. 하지만 원본 데이터의 차원(Feature 수)이 너무 많으면 메타 모델이 과적합될 수 있으니 주의해야 합니다.

Q3. Mixture of Experts에서 '전문가'는 사람이 정해주나요?
A. 아닙니다. 데이터 학습 과정(Gradient Descent)을 통해 모델이 스스로 "이 데이터 영역은 내가 처리할게"라고 학습하게 됩니다. 하지만 초기 설정이나 손실 함수 설계가 잘못되면 한 명의 전문가만 일하는 쏠림 현상이 발생할 수 있습니다.

학습 팁: 처음에는 Random Forest나 Gradient Boosting 같은 검증된 단일 앙상블 기법을 먼저 마스터하세요. 그 후 성능의 한계를 느낄 때, 서로 다른 모델들을 섞는 Stacking을 시도해보는 것이 가장 효율적인 학습 순서입니다.