▣ **강의명:** CS109A: 데이터 과학 입문

▣ **주차:** Lecture 24

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

▣ **목적:** Lecture 24의 핵심 개념 학습

Contents

AdaBoost (Adaptive Boosting)

약한 모델을 모아 강한 모델을 만드는 앙상블 기법

1 개요

□ 핵심 요약

핵심 요약 AdaBoost는 '실수로부터 배운다'는 철학을 가진 머신러닝 알고리즘입니다. 아주 간단한 모델(Weak Learner)을 여러 번 순차적으로 학습시키되, **이전 단계에서 틀린 문제(오분류 데이터)에 더 큰 가중치**를 부여하여 다음 모델이 그 문제를 집중적으로 공부하게 만듭니다. 최종적으로 이들을 합쳐 강력한 예측 성능을 냅니다.

학습 목표:

- AdaBoost가 샘플의 가중치(Weight)를 업데이트하는 원리를 이해합니다.
- 왜 AdaBoost가 지수 손실(Exponential Loss) 함수를 최소화하는지 파악합니다.
- Gradient Boosting과 AdaBoost의 수학적, 절차적 차이를 구분합니다.
- 실제 데이터 분석 프로젝트(과제 등)에서 과적합(Overfitting)을 피하는 방법을 익힙니다.

2 필수 용어 정리

처음 접하는 분들을 위해 핵심 용어를 정리했습니다.

용어	원어	쉬운 설명	비고
약한 학습기	Weak Learner	무작위 추측보다 조금 더 나은 수준의 아주 단순한 모델.	주로 Stump 사용
스텀프	Stump	뿌리만 있는 나무. 질문(분기)이 딱 하나인 결정 트리.	Node 1개, Leaf 2개
앙상블	Ensemble	여러 모델의 의견을 종합하여 결론을 내리는 기법.	집단 지성
가중치	Weight (w)	해당 데이터가 얼마나 중요한지를 나타내는 점수.	틀리면 높아짐
학습률	Learning Rate (λ)	한 모델(스텀프)의 발언권 세기.	신뢰도에 비례
지수 손실	Exponential Loss	정답과 예측이 다를 때 페널티를 급격히(지수적으로) 주는 함수.	$e^{-y\hat{y}}$

Table 1: *AdaBoost* 핵심 용어 비교표

3 핵심 개념과 원리

3.1 1) 기본 아이디어: 오답 노트 만들기

AdaBoost의 작동 방식은 **수험생이 오답 노트를 공부하는 과정**과 매우 비슷합니다.

- **1단계:** 모의고사를 봅니다. 쉬운 문제는 맞히고 어려운 문제는 틀립니다.
- **2단계:** 틀린 문제(오분류 데이터)를 별표(⋆) 쳐서 강조합니다. (가중치 증가)
- **3단계:** 다음 공부 때는 별표 친 문제 위주로 공부합니다.
- **4단계:** 이 과정을 반복한 뒤, 여러 번의 시험 결과를 종합해 최종 실력을 냅니다.

3.2 2) 약한 학습기(Weak Learner)로서의 스텀프(Stump)

AdaBoost는 복잡한 트리를 쓰지 않고, 아주 단순한 **스텀프(Stump)**를 사용합니다.

- **구조:** 질문 하나(Node), 결과 두 갈래(Leaves). (예: "x_1이 4.6보다 큰가?")
- **이유:** 모델이 너무 복잡하면 과적합되기 쉽습니다. 아주 단순한 모델을 여러 개 쌓아 올리는 것이 더 효율적입니다.

3.3 3) 레이블의 변경: 0과 1 대신 -1과 +1

보통 분류 문제에서 클래스를 0과 1로 두지만, AdaBoost는 수학적 편의를 위해 **-1과 +1**을 사용합니다.

- **이점:** 정답(y)과 예측($\hat{y}$)을 곱했을 때의 부호로 정답 여부를 바로 알 수 있습니다.
  - 정답($y = +1$), 예측($\hat{y} = +1$) $\rightarrow y \cdot \hat{y} = 1$ (양수, 정답)
  - 정답($y = -1$), 예측($\hat{y} = -1$) $\rightarrow y \cdot \hat{y} = 1$ (양수, 정답)
  - 틀린 경우 (하나가 +1, 하나가 -1) $\rightarrow y \cdot \hat{y} = -1$ (음수, 오답)
- 이 속성은 가중치 업데이트 수식에서 매우 중요하게 사용됩니다.

4 AdaBoost의 동작 절차 (Step-by-Step)

데이터 포인트가 N개 있을 때, AdaBoost는 다음 과정을 반복합니다.

1. **초기화:** 모든 데이터의 가중치(w_n)를 똑같이 $\frac{1}{N}$로 설정합니다.
2. **반복 (Iteration):** 멈춤 조건(예: 50번 반복)이 될 때까지 다음을 수행합니다.
   (a) **스텀프 학습:** 현재 가중치(w_n)를 고려하여, 데이터를 가장 잘 분류하는 스텀프(S)를 찾습니다. 가중치가 높은 데이터를 틀리면 페널티가 크므로, 모델은 가중치가 높은 데이터를 맞히려고 노력합니다.
   (b) **에러 계산 (ϵ):** 스텀프가 틀린 데이터들의 가중치 합을 구합니다.
   (c) **스텀프의 중요도(λ) 산출:**

$$\lambda = \frac{1}{2} \ln \left(\frac{1 - \epsilon}{\epsilon} \right)$$

   - 에러(ϵ)가 작을수록(잘 맞힐수록) λ는 커집니다. (발언권이 세짐)

- 에러가 0.5(무작위)면 λ는 0이 됩니다. (의견 무시)

(d) **데이터 가중치 업데이트:**

$$w_{new} \leftarrow w_{old} \times \exp(-\lambda \cdot y \cdot S(x))$$

- **맞췄을 때 ($y \cdot S(x) > 0$):** $e^{-\lambda}$ (1보다 작음)를 곱해 가중치 **감소**.
- **틀렸을 때 ($y \cdot S(x) < 0$):** e^{λ} (1보다 큼)를 곱해 가중치 **증가**.

(e) **앙상블 업데이트:** 지금까지 만든 모델에 현재 스텀프를 더합니다.

$$T_{new} = T_{old} + \lambda \cdot S$$

3. **최종 예측:** 모든 스텀프의 예측값에 각자의 중요도(λ)를 곱해 더한 뒤, 부호(Sign)를 봅니다. (양수면 +1, 음수면 -1)

5 직관적 예제: 오렌지색 원과 파란색 삼각형

가상의 2차원 데이터를 분류하는 상황을 상상해 봅시다.

1. **Step 1:** 첫 번째 스텀프가 $x_1 > 4.6$이라는 기준을 세워 데이터를 나눕니다. 몇몇 파란색 삼각형은 맞히지만, 일부 오렌지색 원을 틀립니다.
2. **Weight Update:** 틀린 오렌지색 원들의 크기(가중치)를 키웁니다. 이제 이 원들은 아주 '중요한' 데이터가 되었습니다.
3. **Step 2:** 두 번째 스텀프는 커진 오렌지색 원들을 맞히기 위해 $x_2 > 8$이라는 새로운 기준을 가져옵니다. 이번에는 이전에 맞았던 파란색 삼각형 몇 개를 틀릴 수도 있습니다.
4. **Weight Update:** 이번에 틀린 데이터들의 크기를 다시 키웁니다.
5. **Step 3:** 세 번째 스텀프가 또 다른 기준($x_1 > 5.7$)으로 남은 어려운 문제들을 해결하려 합니다.
6. **결과:** 이 세 개의 단순한 직선(스텀프)들이 합쳐져서, 복잡한 곡선 형태의 경계면을 만들어냅니다.

팁: 스텀프는 어떻게 가중치를 반영하나요? 결정 트리 알고리즘(Gini Index 등)을 계산할 때, 단순히 데이터 개수를 세는 것이 아니라 **가중치의 합(Weighted Sum)**을 사용하면 됩니다. 또는, 가중치가 높은 데이터를 가중치만큼 복사(Resampling)하여 데이터 셋을 늘린 뒤 학습하는 방법도 있습니다.

6 비교: AdaBoost vs Gradient Boosting

두 알고리즘 모두 '이전 모델의 실수를 바로잡는다'는 점은 같지만, 접근 방식이 다릅니다.

비교 항목	Gradient Boosting (GBM)	AdaBoost
목표 함수 (Loss)	주로 MSE (회귀), Log Loss (분류)	**Exponential Loss** ($e^{-y\hat{y}}$)
실수 처리 방식	잔차(Residual)를 직접 학습	오분류 데이터의 **가중치(Weight)**를 증가
학습률 (λ)	사용자가 지정 (하이퍼파라미터)	**수식으로 자동 계산** (Optimal λ)
최적화 방식	손실 함수의 기울기(Gradient)를 따라감	Exponential Loss를 최소화하는 해를 분석적으로 찾음

Table 2: *Gradient Boosting*과 *AdaBoost*의 차이점

- **AdaBoost의 특징:** 학습률(λ)을 튜닝할 필요 없이, 매 단계마다 수학적으로 최적의 λ를 찾아냅니다. (물론 구현체에 따라 추가적인 학습률 조정 옵션이 있을 수 있습니다.)
- **Gradient 관점:** AdaBoost도 넓은 의미에서 보면 Exponential Loss에 대한 Gradient Descent를 수행하는 것으로 해석할 수 있습니다.

7 주의사항: 과적합(Overfitting)

7.1 1) Boosting은 과적합될 수 있다

Random Forest와 같은 배깅(Bagging) 기법은 나무를 많이 심을수록 에러가 줄어들고 안정화되는 경향이 있습니다. 반면, **Boosting 계열(AdaBoost 포함)은 너무 많이 반복하면 과적합됩니다.**

- **이유:** 계속해서 틀린 문제(노이즈나 이상치일 수도 있음)에 집착하여 가중치를 높이기 때문에, 나중에는 데이터의 노이즈까지 외워버리게 됩니다.
- **해결:** 검증 데이터(Validation Set)의 에러가 다시 증가하기 시작하는 시점에서 학습을 멈추는 **조기 종료(Early Stopping)**가 필수적입니다.

7.2 2) 주요 하이퍼파라미터 (Hyperparameters)

프로젝트나 과제 진행 시 조정해야 할 값들입니다. (Scikit-Learn 기준)

- `n_estimators`: 스텀프의 개수(반복 횟수). 너무 크면 과적합.
- `learning_rate`: 각 스텀프의 영향력을 전체적으로 줄이는 비율. (기본값 1.0). 값을 줄이면 `n_estimators`를 늘려야 합니다.
- `base_estimator`: 약한 학습기의 종류. (기본값: Depth 1짜리 Decision Tree). 깊이를 늘리면 모델이 복잡해져 과적합 위험이 커집니다.

8 실전 가이드 및 자주 묻는 질문(FAQ)

8.1 1) 언제 어떤 라이브러리를 써야 하나요?

- **Scikit-Learn (AdaBoostClassifier):** 학습용, 데이터가 작을 때, 기본 원리 이해용.
- **XGBoost / LightGBM / CatBoost:** 실무 프로젝트, 대용량 데이터, 높은 성능이 필요할 때. (속도 최적화, 결측치 처리, 정규화 기능이 포함됨)

8.2 2) FAQ: 초심자가 자주 하는 오해

Q1. 왜 하필 지수 손실(Exponential Loss)을 쓰나요?
A1. 지수 손실은 미분 가능하며, 0-1 손실(맞으면 0, 틀리면 1)의 상한선(Upper Bound) 역할을 합니다. 즉, 지수 손실을 줄이면 분류 에러도 자연스럽게 줄어듭니다. 또한, 잘못된 예측에 대해 벌점을 아주 크게 주기 때문에 학습 방향을 잡기 좋습니다.

Q2. 스텀프(Stump)는 너무 단순하지 않나요?
A2. 바로 그 단순함이 핵심입니다! 모델 하나하나는 약하지만(무작위보다 조금 나은 수준), 이것들이 수십, 수백 개 모여 서로의 단점을 보완하면 매우 복잡하고 정교한 경계면을 만들어냅니다.

Q3. 결정 트리는 파라메트릭인가요, 논파라메트릭인가요?
A3. 면접이나 시험에 나올 수 있는 질문입니다. 정답은 **"둘 다 볼 수 있다"**입니다.

- **Parametric 관점:** 분기 기준(Threshold)과 같은 파라미터를 학습하므로.
- **Non-parametric 관점:** 데이터가 많아질수록 트리 구조가 계속 커지고 복잡해지므로(파라미터 수가

고정되어 있지 않음). 통계학적으로는 주로 Non-parametric으로 분류합니다.

9 학습 체크리스트 (시험 및 프로젝트 대비)

주의사항

과제/프로젝트 주의사항 XGBoost와 같은 최신 라이브러리는 과제(Homework)에서는 사용이 제한될 수 있으니(직접 구현 요구 등), 공지사항을 꼭 확인하세요. 하지만 기말 프로젝트에서는 성능을 위해 적극 권장됩니다.

- ☐ **개념:** Weak Learner가 모여 Strong Learner가 되는 과정을 문장으로 설명할 수 있는가?
- ☐ **수식:** 가중치 업데이트 식에서 $y \cdot \hat{y}$의 부호에 따라 가중치가 어떻게 변하는지 설명할 수 있는가? (맞으면 감소, 틀리면 증가)
- ☐ **계산:** λ 값을 구하는 공식 $\frac{1}{2}\ln(\frac{1-\epsilon}{\epsilon})$을 알고 있는가?
- ☐ **비교:** Random Forest(배깅)와 AdaBoost(부스팅)의 과적합 성향 차이를 아는가?
- ☐ **구현:** Scikit-Learn을 사용하여 모델을 학습시키고, 하이퍼파라미터(`n_estimators`)를 튜닝할 수 있는가?
- ☐ **평가:** Validation Curve를 그려보고, 언제 학습을 멈춰야 할지(Early Stopping) 판단할 수 있는가?

10 한 페이지 요약 (Cheat Sheet)

AdaBoost Quick Review

1. **정의:** 약한 학습기(Stump)를 순차적으로 연결하여 강한 예측기를 만드는 앙상블 기법.
2. **핵심 메커니즘:**
   - 이전 모델이 틀린 데이터 → 가중치(w) 증가 → 다음 모델이 집중 학습.
   - 잘 맞히는 모델 → 중요도(λ) 증가 → 최종 투표에서 큰 목소리.
3. **손실 함수:** Exponential Loss ($e^{-y\hat{y}}$)를 최소화.
4. **장점:** 구현이 쉽고, 과적합에 강한 편(하지만 영원히 강하진 않음), 파라미터 튜닝이 비교적 적음.
5. **단점:** 노이즈 데이터나 이상치(Outlier)에 민감함 (계속 가중치를 키우려다 망가질 수 있음).
6. **코드 예시 (Python/sklearn):**

```python
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier

# 깊이가인 1 약한학습기 (Stump) 생성
stump = DecisionTreeClassifier(max_depth=1)

# 아다부스트모델생성번   (50 반복)
ada = AdaBoostClassifier(
    base_estimator=stump,
    n_estimators=50,
    learning_rate=1.0
)

# 학습및예측
ada.fit(X_train, y_train)
y_pred = ada.predict(X_test)
```