A short introduction to boosting - 2021년 P-sat 논문스터디 8주차

김원구, 김재희, 오정민

2021년 3월 23일

'A short introduction to boosting' 논문을 바탕으로 AdaBoost에 대한 전반적인 내용을 소개하는 자료입니다.

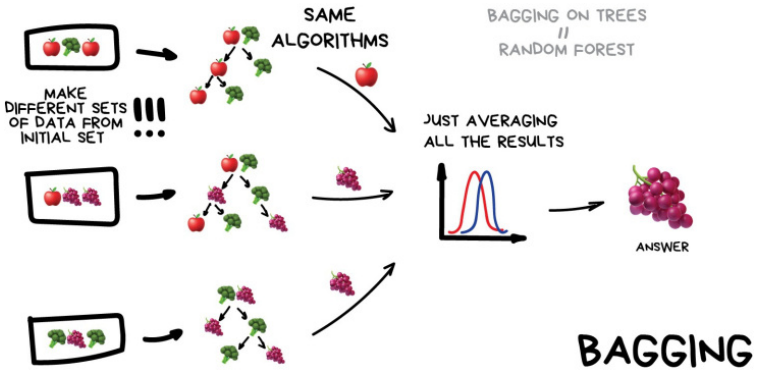
1. Introduction

- 본 논문에서는 부스팅 개념을 설명하기 위해 경마 알고리즘을 예시로 든다. 경마에서 이기는 컴퓨팅 알고리즘을 개발한다고 하면, 전문 도박사도 완벽한 알고리즘을 제시하지는 못한다. 하지만, 가장 유리한 조건의 말에 베팅 혹은 최근 승률이 가장 좋은 말에 베팅 등의 경험 법칙(rules of thumbs)이 나람대로 효과가 좋을 수 있다. 물론 이런 경험 법칙은 부정확하지만 적어도 랜덤 추측(random guessing)보다는 나을 것이다. 이러한 최소한의 성능을 보장하는 경험 법칙을 많이 모으면 어떨까? weak learner들을 모아 strong learner를 만들어보자!
- 그러나 여기에는 2가지 챌린지가 있다. 1) 어떤 weak learner들을 골라야 하나? 2) 어떻게 그것들을 하나의 고성능 single model로 통합할 수 있나? 이 2가지 챌린지는 boosting 기법으로 해결할 수 있다! 본 논문에서는 부스팅 모델 중 하나인 AdaBoost 모델을 설명하고, 왜 boosting 모델이 overfit에 강건한지를 밝힌다.

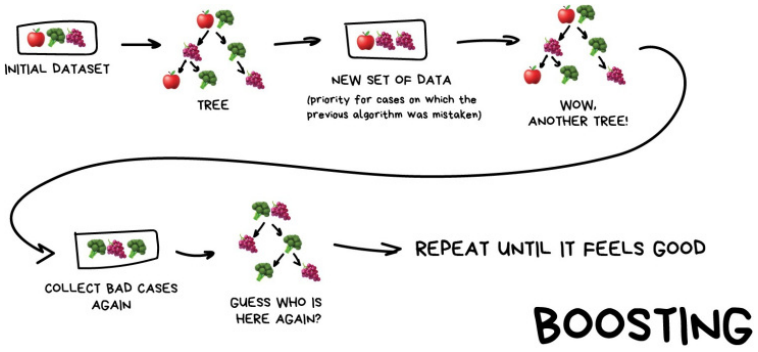
2. AdaBoost

2.1 Boosting 개요

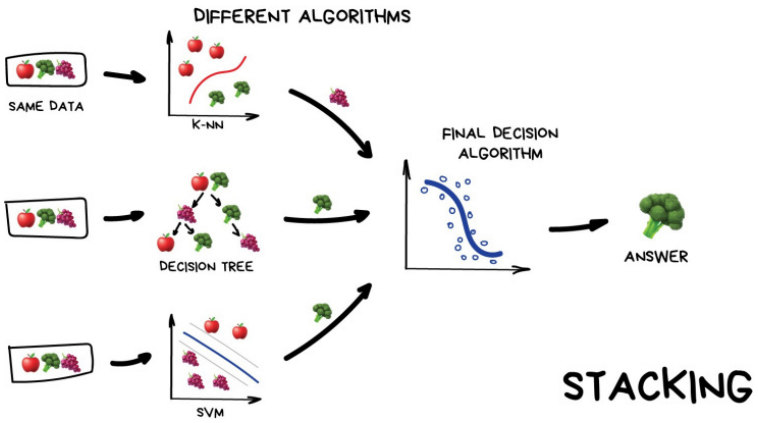
- 부스팅은 여러 모델의 예측을 결합함으로써 보다 정확한 예측을 도출하는 앙상블 기법 중 하나이다. 랜덤포레스트로 익숙한 배깅 앙상블은 각 weak learner를 학습 후 그 결과를 aggregating해 최종 결과를 도출한다.



- 반면, 부스팅은 각 weak learner를 순차적으로 학습하며 잘못 예측한 데이터에 가중치를 부여해 오류를 개선해 나가며 학습한다.



- 스태킹은 여러 base model의 output을 meta model의 input으로 사용해 결과를 도출한다.



2.2 AdaBoost 개요

- AdaBoost는 기존에서는 개별 모델에 가중치를 별도로 주는 개념이 추가된 방식이다. (원래 부스팅은 동일한 가중치 부여)
- cf) 이후에 등장하는, XGBoost의 이전 모델인 GBM은 가중치 업데이트를 Gradient Descent를 통해 한다는 점이 AdaBoost와의 큰 차이

- AdaBoost 알고리즘의 핵심은 training set 전체에 대해 분포와 가중치 셋을 유지한다는 점이다. t번째 라운드의 training example i의 분포는 $D_t(i)$이며, 초기 가중치는 동일하게 적용된다. (ex. 처음에는 모두 $1/m$으로 유지, m은 샘플 크기) 가중치는 매 라운드마다 오분류 샘플에 대해 증가해 모델이 분류하기 어려운 샘플에 더 집중하게 한다.

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error

$$\epsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right].$$

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right)$.
- Update:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \end{aligned}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

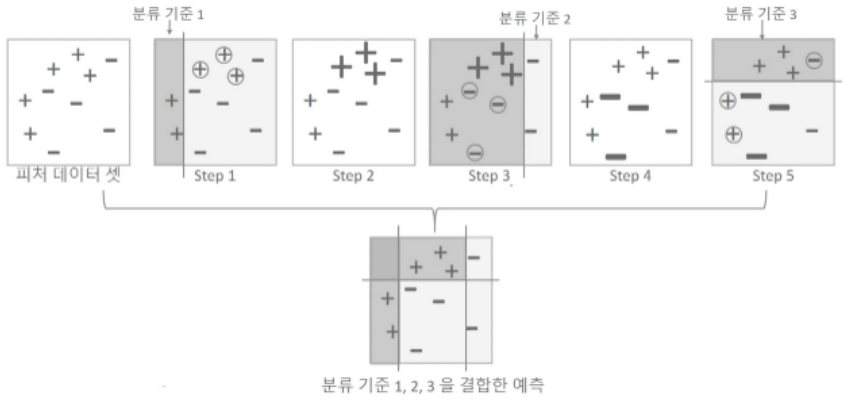
Figure 1: The boosting algorithm AdaBoost.

- weak learner의 error는 아래 식으로 측정된다. error는 각 라운드 t 마다 weak learner가 학습되는 분포 D_t에 대해 측정된다.

$$\epsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i: h_t(x_i) \neq y_i} D_t(i)$$

- 여기서, α_t는 해당 weak모델 H_t의 중요성을 측정하는데, $\epsilon_t \leq 1/2$라면 $\alpha_t \geq 0$이며 α_t가 커질수록 ϵ_t는 작아진다. 왜냐하면 ϵ_t가 $1/2$ 이하라는 것은 weak leaner의 성능이 기본 조건인 "오분류율 50% 이하"를 만족하고 작을수록 성능이 더 좋다는 의미이기 때문이다.
- 최종 모델은 H는 weak learners 집합 T의 majority vote이다.
- 아래는 AdaBoost의 학습 조감도이다. (출처: 출처)

AdaBoost의 학습/예측 프로세스



3. 학습 오차 분석

- 위에서 정의했던 h_t의 error ϵ을 $\frac{1}{2} - \gamma_t$로 정의해보자. Binary problem case에서 random guess를 했을 때, error rate는 $\frac{1}{2}$일 것이다. 그러면 우리가 새롭게 정의한 ϵ, 즉 $\frac{1}{2} - \gamma_t$에서 γ_t가 의미하는 바는 h_t에 대한 prediction에 대해서 random guess보다 얼마나 더 잘 맞추었는가에 대한 지표가 된다.
- Final hypothesis H에 대한 training error의 upper bound는 다음과 같다.

$$\prod_t 2\sqrt{\epsilon_t(1-\epsilon_t)} = \prod_t \sqrt{1-4\gamma_t^2} \leq exp\left(-2\sum_t \gamma_t^2\right) \leq exp\left(-2t\gamma^2\right)$$

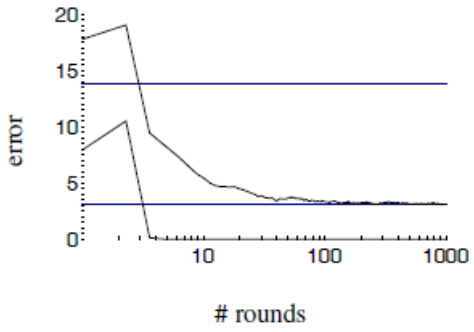
- 그러므로 각각의 weak hypothesis가 random보다 약간이라도 더 좋은 성능을 가진다면, 즉 γ_t가 0보다 큰 어떤 γ보다 크거나 같은 값을 가지게 된다면($\gamma_t \geq \gamma, \exists\, \gamma > 0$), training error는 T가 증가함에 따라 exponentially fast하게 감소한다.

4. 일반화 오차

- 앞에서는 training error에 대해 살펴보았는데, 이번에는 알고리즘이 new test data에 대한 성능을 살펴보고자 generalization error에 대해서 다룬본다.
- generalization error를 앞에서 우리가 구했던 training error의 term을 이용하면 다음과 같이 표현이 가능하다.

$$\hat{Pr}[H(x) \neq y] + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right)$$

- 먼저 generalization error와 관련해서 해당 결과값에 대한 유도가 너무 어렵고 복잡하여, 따로 추가하지 않은 점 양해바란다…! (해당 논문에는 유도가 나와있지는 않다…)
- 위의 식에 앞에서 구한 training error의 upper bound를 대입하면 generalization error의 upper bound를 쉽게 도출할 수 있다. 여기서 T는 # of boosting rounds, m은 sample size, d는 VC-dimension of the weak hypothesis space를 의미한다. 여기서 $\tilde{O}$는 원래의 Big-O notation에서 log term을 생략한 표현법이다.
- 해당식을 보면 알 수 있는 점이 T가 커짐에 따라 training error term은 감소하고 $\tilde{O}$는 증가함으로 trade-off 관계에 있음을 알 수 있다. 즉, T는 무한정하게 커지게 만들면 결국 generalization error가 커지게 되어 overfitting이 일어난다는 추론을 할 수 있다. Occam's Razor principle에 의하면 이런 trade-off에 최적화된 generalization error를 최소화 할 수 있는 T를 찾을 수 있다고 한다.
- 하지만 실제로는 이런 overfitting 발생하는 경우가 많지 않다고 한다. training error가 0에 도달한 이후에도 generalization error가 계속해서 감소하는 경우가 많이 발견되었는데, 그 이유는 hypothesis h에 대하여 단순히 'right or wrong'의 개념 뿐만 아니라 'margin'에 의해 측정되는 hypothesisd의 'confidence' 정도 또한 영향을 주기 때문이다. training error가 0에 도달한 이후에도 confidence는 rounds가 계속됨에 따라 증가하게 되고, 이는 결국 generalization error를 감소하게 만든다.
- 아래는 'letter' dataset에 대해서 training error와 test error의 변화를 보여준다.

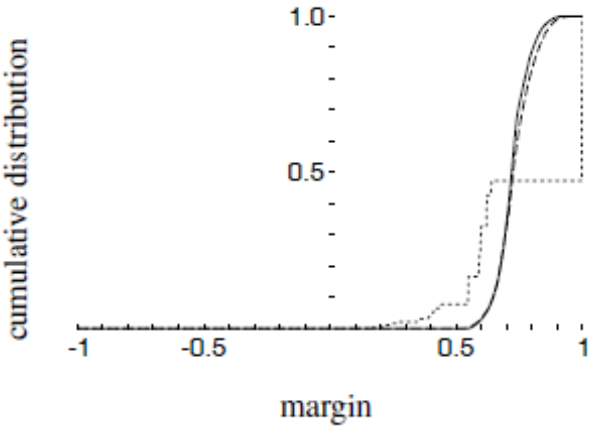


- 이번에는 margin의 관점에서 살펴보도록하자. margin은 다음과 같이 표현 가능하다.

$$
\begin{aligned}
margin(x,y) &= \frac{y\sum_t \alpha_t h_t(x)}{\sum_t \alpha_t} \\
&= \frac{\sum_t \alpha_t y h_t(x)}{\sum_t \alpha_t} \\
&= \frac{\sum_{t:h_t(x)=y} \alpha_t - \sum_{t:h_t(x)\neq y} \alpha_t}{\sum_t \alpha_t}
\end{aligned}
$$

- 분자에 대해서 생각을 해보면 올바르게 분류한 weights와 올바르게 분류하지 못한 weights들의 차이가 된다. 따라서, margin은 분류문제에서 confidence의 측정정도로 생각할 수 있다.

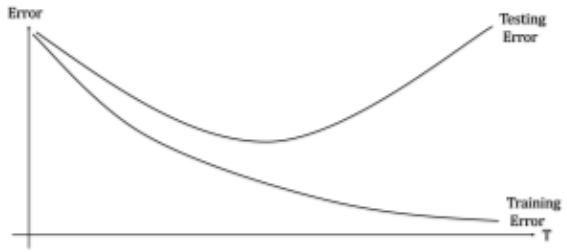
number of rounds	5	100	1000
training error	0.0	0.0	0.0
test error	8.4	3.3	3.1
% margins ≤ 0.5	7.7	0.0	0.0
minimum margin	0.14	0.52	0.55

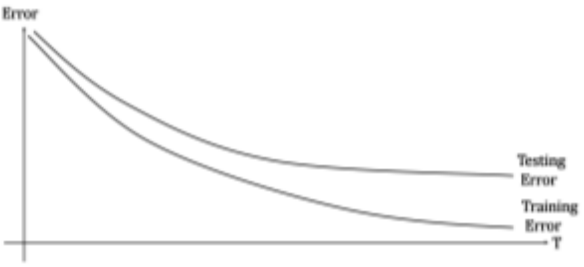


- 위의 표를 보면 알 수 있듯이 training error가 0이 되었음에도 rounds가 increase할수록 minimum margin 값 또한 커지게 되고 test error는 계속해서 감소함을 알 수 있다.
- margin의 관점에서 generation error를 표현하면 다음과 같이 표현이 가능하다.

$$
\hat{Pr}[margin(x,y) \leq \theta] + \tilde{O}\left(\sqrt{\frac{d}{m\theta^2}}\right) \quad for\ any\ \theta > 0
$$

- 식에서 1번째 term은 T가 커짐에 따라 0으로 수렴하고, 2번째 term은 T로 부터 독립적이다. 따라서 이 식은 Adaboost에 대한 generation error가 T가 커짐에 따라 감소하는 경향이 있음을 보여준다. 그 결과 우리가 아는 일반적인 형태의 trade-off 관계가 아닌 지속적으로 감소하는 형태가 나오게 된다.





5. SVM과 Adaboost의 관계

- 이 장에선 SVM과 Adaboost의 관계에 대해 다루고 있습니다. 결론부터 이야기하면, "서로 다른 듯 보이지만 Adaboost는 SVM의 장점을 가져오고자 했다." 입니다. 여기서 장점이란 과적합에 강건하다는 점입니다. 이 점을 염두에 두고 시작하겠습니다.
- margin theory에선 부스팅 모델들과 SVM의 공통점을 알 수 있습니다. 여기선 부스팅 모델이 약한 분류기를 이미 찾았고, 각각의 분류기를 엮어서 강한 분류기를 만들 때 필요한 α를 찾는 단계라고 생각해봅시다.
- 합리적으로 생각해보면, 부스팅 모델이 일반화가 잘 될 수 있도록 α를 선택해야 할 것 입니다. 이를 위해 앞에서 언급한 식인

$$\hat{\Pr}\left[\text{margin}(x,y) \leq \theta\right] + \tilde{O}\left(\sqrt{\frac{d}{m\theta^2}}\right)$$

을 최소화하면 될 것입니다. 이 식이 나타내는게 부스팅 모델의 일반화 성능의 최대값이니까요(maximum error bound).

- 여기서 식이 너무 복잡하니까, 앞의 항은 0이라고 간주합시다. 즉, 모델이 예측한 레이블이 모두 정확하다고 간주하는 겁니다(과적합 상황 가정). 뒷 항은 분류기의 margin에 대한 항이므로 이제 SVM처럼 각 데이터 중 margin이 가장 작은 데이터의 margin을 최대화하는 문제가 됩니다.
- 여기서 $h(x) =< h_1(x), h_2(x), ..., h_N(x) >$
  $\alpha =< \alpha_1, \alpha_2, ..., \alpha_N >$

이런 식으로 벡터화하여 표현하겠습니다.

- $h(x)$는 각 약한 분류기의 예측값이 될 것이고, α는 각 약한 분류기의 예측값에 대한 가중치가 될 것입니다.
- 위에서 확인했던 margin에 대한 식과, 여기서 정의한 두 벡터를 이용하면 margin은 다음과 같이 표현할 수 있습니다.

$$\max_{\boldsymbol{\alpha}} \min_{i} \frac{(\boldsymbol{\alpha} \cdot \mathbf{h}(x_i))y_i}{||\boldsymbol{\alpha}||\ ||\mathbf{h}(x_i)||}$$

- 이 식을 해석해보면, margin이 제일 작은 데이터(row, instance)의 margin을 최대화 하는 α를 찾는 것입니다.
- 그리고 여기서 l1 norm은 각 원소의 절대값의 합이고, lmax norm은 각 원소 중 최대값이므로 다음과 같은 식으로 표현할 수 있습니다.

$$||\boldsymbol{\alpha}||_1 \doteq \sum_t |\alpha_t|, \qquad ||\mathbf{h}(x)||_\infty \doteq \max_t |h_t(x)| \ .$$

- 그런데 $h_t(x)$는 -1 혹은 1의 값을 반환하는 함수이므로 $\|h(x)_\infty\| = 1$일 것입니다.
- 하지만 SVM은 다음과 같은 식을 사용하여 각 벡터를 정의합니다.

$$||\boldsymbol{\alpha}||_2 \doteq \sqrt{\sum_t \alpha_t^2}, \qquad ||\mathbf{h}(x)||_2 \doteq \sqrt{\sum_t h_t(x)^2} \ .$$

- 부스팅 모델과 다른 점은 두 항 모두 l2 norm을 이용하여 유클리드 거리를 사용한다는 점입니다.
- 부스팅 모델을 SVM처럼 보면, 이렇게 사용하는 norm의 차이 외에는 큰 차이가 없는 것으로 보입니다. 하지만 이 작은 차이로 인해 다음과 같은 3가지의 큰 차이점이 발생합니다.

(1) 다른 norm의 사용 = margin의 큰 차이

- 차원 수가 크지 않을 때, norm의 차이는 사실 크지 않습니다. 2차원에 l1 norm을 사용하나 l2 norm을 사용하나 대각선과 직선 정도의 차이를 가져올 뿐입니다. 하지만 차원이 커질 수록 norm에 따른 차이는 급격히 커지게 됩니다.
- 예를 들어, 1x1 짜리 정사각형을 상상해봅시다. l1 norm을 사용할 경우 2지만 l2 norm을 사용하면 $\sqrt{2}$로 그 차이는 $2-\sqrt{2}$ 입니다. 하지만 1x1x1 짜리 정육면체에선 l1 norm 은 3, l2 norm은 $\sqrt{3}$으로 두 차이는 $3-\sqrt{3}$으로 2차원에서보다 커지게 됩니다.
- 부스팅 모델이든 SVM이든 데이터의 차원수는 큰 것이 일반적인 경우입니다. 특히, 변수 간의 관계가 높을수록, 즉 α가 매우 sparse한 벡터일 경우 이 문제는 커지게 됩니다. (변수 간 관계가 높다면 하나의 변수가 다른 변수의 많은 부분을 설명하기 때문에 α가 대부분 작은 값을 가지는 벡터가 됩니다.) 이 경우 l_{max} norm(부스팅)에선 단순히 큰 값을 취하면 되지만 l2 norm(SVM)에선 모든 원소를 제곱합하게 됩니다. 차원수가 많다면 두 norm 사이의 차이는 커질 수 밖에 없습니다.

(2) 컴퓨터 연산량 차이

- SVM은 기본적으로 연산량을 많이 필요로 합니다. 커널에 따라 다르지만 보통 $O(N^2)$ 혹은 $O(N^3)$의 연산량을 필요로 합니다. 하지만 부스팅 모델은 (우리가 사용하는 한) 기본적인 트리 모델을 사용하기 때문에 $O(N)$으로 훨씬 적은 연산량을 가지고 있습니다. 맞아요 논문에서 자기들이 잘했다고 자랑하는 겁니다.

(3) 고차원 데이터 사용

- SVM과 부스팅 모델은 고차원 데이터를 효율적으로 사용하기 위해 다루는 탐색 방식이 다릅니다. SVM은 고차원 데이터를 커널을 이용해 저차원으로 맵핑하여 문제를 해결하지만, 부스팅은 greedy search(탐욕 알고리즘)을 이용하여 문제를 해결합니다. 탐욕 알고리즘이란 최종적인 결과를 고려하지 않고(보통 최종 결과를 고려할 수 없을 때 사용합니다.) 현재 상황에서 최선의 결과를 선택하는 것을 의미합니다. adaboost는 이전의 학습기가 제대로 예측하지 못한 오류만 고려하므로 탐욕 알고리즘을 채택했다고 볼 수 있습니다.

6. 다중분류 문제

- 이전까지 이진 분류 문제를 가정하여 adaboost를 설명했습니다. 그렇다면 다중 분류 문제에선 adaboost를 어떻게 적용할 수 있을까요. 앞으로 몇가지 방법을 소개하도록 하겠습니다.

(1) AdaBoost.MH

- 이 모델은 n개 클래스에 대한 다중 분류 문제를 n개의 이진 분류 문제로 전환하여 해결합니다. 예를 들어 개, 고양이, 사자, 호랑이에 대한 분류 문제가 있다면, MH 모델은

(1) 개인가 다른 동물인가
(2) 고양이인가 다른 동물인가
(3) 사자인가 다른 동물인가
(4) 호랑이인가 다른 동물인가 의 문제로 전환하여 문제를 해결합니다.

(2) AdaBoost.M2

- 이 모델은 MH 모델에 비해 조금 복잡하게 문제를 해결합니다. 위의 예시를 이용해보자면

(1) 개인가, 고양이인가
(2) 개인가, 사자인가
(3) 개인가, 호랑이인가
(4) 고양이인가, 사자인가
(5) 사자인가, 호랑이인가
(6) 사자인가, 호랑이인가 의 문제로 변환합니다.

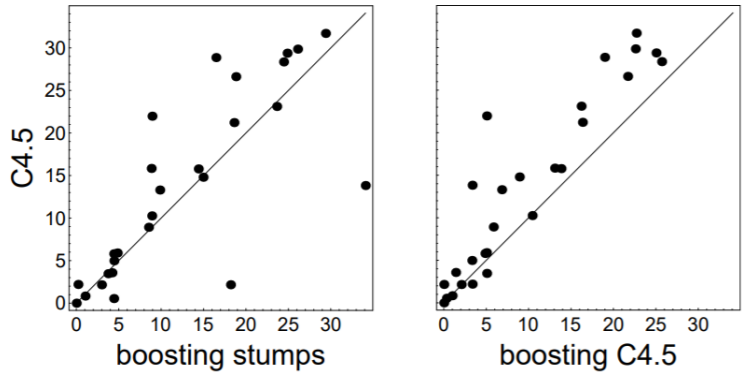
- 이렇게 각각의 이진분류문제로 전환하여 예측값을 산출하면 MH 모델은 4개, M2 모델은 6개의 예측값이 나오는데 이것들을 어떻게 합칠까요? 가장 단순한 방법은 voting, 즉 다수결을 통해 해결합니다. 혹은, margin 계산하기. 각 이진분류문제마다 margin을 계산할 수 있습니다. 이 margin을 이용하여 가장 margin이 먼 클래스로 분류하면 됩니다.

7. 실험 및 적용

- Adaboost는 빠르고, 단순하며, 코딩하기도 쉽습니다. 크게 보면 round 수를 제외하면 튜닝해야할 파라미터도 없습니다. 심지어 약한 분류기를 어떻게 합쳐야 할지 고민할 필요도 없죠. 즉, 충분한 데이터와 충분한 약한 분류기를 확보할 수 있다면 최소한의 성능을 확보할 수 있는 모델입니다. 전체 변수, 전체 데이터를 잘 맞추는 모델을 만들 필요 없이 50% 이상의 정확도를 가지는 약한 학습기만 만들면 되니까요.
- 반대로 말하면, 데이터가 충분하지 못하거나, 학습기의 성능을 50%이상 확보하지 못하면, 절대로 제대로 학습하지 못하는 모델입니다. (현재 상황에서 생각하면 사실 고려할 필요도 없을 정도의 문제점이죠) 하지만 이외에도 단순한 트리 기반 + 탐욕 알고리즘의 조합으로 인해 노이즈에 예민하다는 문제점도 존재합니다.
- 그러면 실험은 어떻게 되었나 살펴보도록 합시다.

(1) UCI benchmark Dataset + C4.5

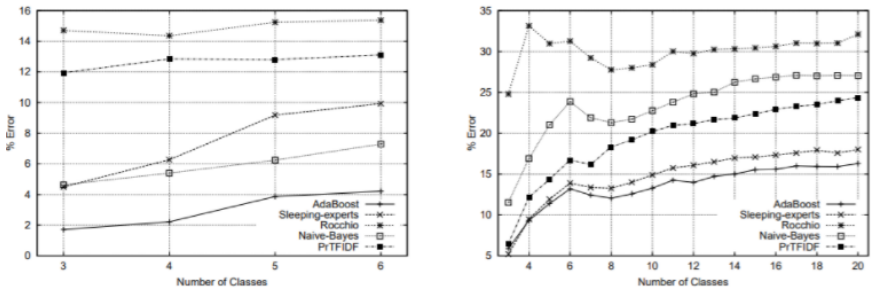
- UCI 데이터셋은 UC 어바인 대학교에서 제공하는 머신러닝을 위한 다양한 데이터셋입니다. 이 논문에선 각각의 데이터셋에 대해 적용하여 결과를 살펴보았습니다.
- 사용한 모델은 C4.5라는 트리 모델과 단순히 하나의 가지만 치는 stump 모델이었습니다. C4.5 모델은 가지치기(prunning)이 제공되는 트리 모델입니다. 그 결과는 다음과 같습니다.



- y 축은 C4.5 모델만 사용한 것이고, x 축은 stump나 C4.5 모델을 부스팅한 결과입니다. 정확도를 scatter plot 으로 나타낸 것이기 때문에, 부스팅을 사용하였을 때, 대부분의 경우에 C4.5보다 좋은 성능을 보이는 것을 알 수 있습니다.

(2) Text Cateogorization

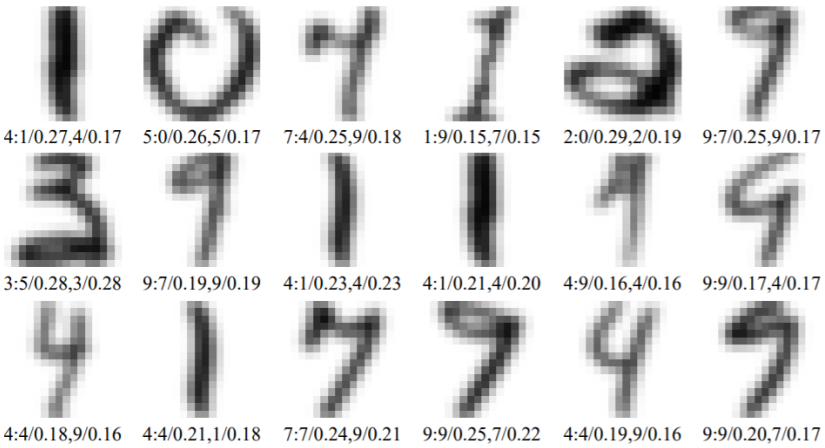
- 자연어 처리에서 학습 시 사용되지 않은 단어나 문장이 테스트 시 나타나는 것은 큰 문제입니다(보통 Out of Bag, OOB 문제라고 합니다). 이를 고려하여 문서 분류 문제에선 테스트 시의 성능을 평가하였습니다. 결과는 아래와 같습니다.



- 그 결과 AdaBoost의 성능이 다른 어떤 모델보다 항상 좋은 것을 알 수 있습니다. 부스팅 방법론은 이외에도 문서 필터링(스팸 처리), 랭킹 시스템(추천), 분류 문제에서 광범위하게 사용되고 있다고 합니다.

(3) 이상치 식별(identification of outlier)

- Adaboost의 장점 중 하나는 이상치를 식별한다는 것입니다. 우리가 흔히 사용하는 이상치 탐지가 아니라 식별한다고 번역한 것은 이상치(클래스가 잘못 분류되어 있거나, 데이터가 분류하기에 모호한 데이터일 경우)를 의미합니다. Adaboost는 round를 거듭하면서 분류하기 데이터에 집중하기 때문에 가중치가 큰 데이터는 종종 이상치일 수 있습니다. 다음을 살펴봅시다.



- 위 데이터는 각각 round가 4, 12, 25일 때 큰 웨이트를 가지고 있는 데이터입니다. 그림 밑에 써져 있는 숫자는 $d : l_1/w_1, l_2/w_2$ 입니다. d는 실제 클래스, l_n은 n번째로 높은 weight를 가진 클래스 $weight_n$은 해당 웨이트 입니다. 각각의 데이터들은 w_1과 w_2가 거의 동일한 것을 볼 수 있는데요. Adaboost에서 이를 분류하기 힘들어하는 것을 의미합니다.
- 이렇게 이상치에 민감하게 반응하다 보니, 이상치의 개수가 Adaboost의 성능에 큰 영향을 미치게 됩니다. 이를 해결하기 위해 이상치에 적은 가중치를 부여하는 Gentle AdaBoost가 제시되었다고 합니다!

Appendix

training error

step 1. $f(x) = \sum_t \alpha_t h_t(x)$라 가정하고, $D_{t+1}(i)$를 구한다.

$$\begin{aligned} D_{T+1}(i) &= D_1(i)\frac{exp(-\alpha_1 y_i h_1(x_i))}{Z_1}...\frac{exp(-\alpha_T y_i h_T(x_i))}{Z_T} \\ &= \frac{1}{m}D_1(i)\frac{exp(-y_i\sum_t \alpha_t h_t(x_i))}{\prod_t Z_t} \\ &= \frac{1}{m}D_1(i)\frac{exp(-y_i f(x_i))}{\prod_t Z_t} \end{aligned}$$

step 2. H의 triaing error의 계산가능한 범위($\prod_t Z_t$)를 구한다.

$$\begin{aligned} training\,error(H) &= \frac{1}{m}\sum_i \begin{cases} 1 & if\ \ y_i \neq H(x_i) \\ 0 & else \end{cases} \\ &= \frac{1}{m}\sum_i \begin{cases} 1 & if\ \ y_i f(x_i) \leq 0 \\ 0 & else \end{cases} \\ &\leq \frac{1}{m}\sum_i exp(-y_i f(x_i)) \\ &= \sum_i D_{T+i}(i)\prod_t Z_t \\ &= \prod_t Z_t \end{aligned}$$

step 3. Z_t를 계산

$$\begin{aligned} Z_t &= \sum_i D_t(i) \times \begin{cases} e^{-\alpha_t} & if\ \ h(x_i) = y_i \\ e^{\alpha_t} & if\ \ h(x_i) \neq y_i \end{cases} \\ &= \sum_{i:h_t(x_i)=y_i} D_t(i)e^{-\alpha_t} + \sum_{i:h_t(x_i)\neq y_i} D_t(i)e^{\alpha_t} \\ &= e^{-\alpha_t}\sum_{i:h_t(x_i)=y_i} D_t(i) + e^{\alpha_t}\sum_{i:h_t(x_i)\neq y_i} D_t(i) \\ &= e^{-\alpha_t}(1-\epsilon_t) + e^{\alpha_t}\epsilon_t \\ &= 2\sqrt{\epsilon_t(1-\epsilon_t)} \\ &= \sqrt{1-4\gamma_t^2} \\ &\leq e^{-2\gamma_t^2} \end{aligned}$$