



Ensemble methods

Sang Yup Lee



Ensemble approach

- Ensemble method
 - To combine several (weak) learners into a stronger learner
 - Approaches
 - Bagging (Bootstrap Aggregating)
 - Resampling => many subsamples
 - Random forest
 - Boosting
 - 이전 learner의 부족한 부분을 이후의 learner가 보완
 - Gradient boosting
 - Adaptive boosting (AdaBoosting)



Bagging

- 참고논문
 - Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2), 123-140.
- 기본 원리
 - 우리가 갖고 있는 학습 데이터(하나의 sample)를 이용하여 여러개의 subsample data 를 만들고, 각 subsample에서 예측값을 계산하고 그 값들을 이용하여 최종 예측을 하는 방법
 - subsample size = size of original data (usually)
 - (새로운 관측치에 대한) 예측
 - 회귀문제: 평균
 - 분류문제: majority voting (즉, 최빈값 사용)



Bagging

- Subsampling 방법

- Bootstrapping

- Random sampling with replacement (중복을 허용한 sampling 방법)
 - 중복을 허용했기 때문에, 2번이상 추출된 관측치 존재
 - 같은 sample size라고 하더라도 포함된 관측치들이 어느정도 다르다.

- Example

- Original dataset (original sample) =>
($dp_1, dp_2, dp_3, dp_4, dp_5, dp_6, dp_7, dp_8, dp_9, dp_{10}$)
 - \hat{y}_1 → sample1 => ($dp_3, dp_6, dp_7, dp_3, dp_1, dp_{10}, dp_6, dp_9, dp_5, dp_2$)
 - \hat{y}_2 → sample2 => ($dp_5, dp_2, dp_1, dp_1, dp_{10}, dp_9, dp_8, dp_7, dp_7, dp_4$)

- Subsample size => hyperparameter



Bagging

- 주효과
 - 모형의 일반화 가능성 증가 \Rightarrow unseen data에 대해서 모형의 성능 증가
- Python code
 - “Bagging.ipynb”



Random Forest

- 기본원리
 - Bagging 의 한계
 - 각 decision node에서 모든 feature를 고려해서 data를 split하게 되는데, 여러개의 tree를 사용할 지라도 매번 비슷한 feature 사용 (subsample 간의 차이 미미) \Rightarrow 즉, 학습데이터 특성에 영향을 많이 받는다 \Rightarrow overfitting problem
 - Bagging의 확장
 - 각 decision node에서 데이터를 split하는데 모든 feature를 사용하는 것이 아니라 random하게 선택된 일부의 feature들만을 사용
 - 사용되는 feature의 수 $= \sqrt{p}$
 - p = 전체 feature의 수



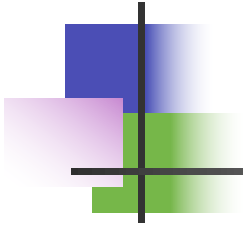
Random Forest

- Python code
 - “Bagging.ipynb”



Random Forest

- `oob_score` parameter
 - `oob_score`: bool, default=False
 - Whether to use out-of-bag samples to estimate the generalization score. Only available if `bootstrap=True`.

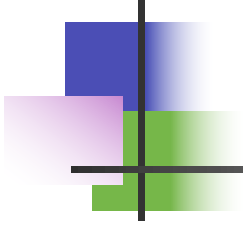


Boosting



Boosting

- 기본원리
 - 학습 데이터에 대해서 tree (혹은 learner)를 순차적으로 (sequentially) 적용
 - 각 단계의 tree (혹은 learner)의 목적은 이전 tree가 잘 설명하지 못한 부분을 보완하는 것
 - 종류
 - adaptive boosting (AdaBoosting)
 - gradient boosting



AdaBoosting

AdaBoosting (Adaptive boosting)

■ 기본원리

- Adaboosting은 예측이 제대로 되지 않은 관측치들에 가중치를 주어서 그 다음 learner가 해당 관측치들을 더 많이 고려하게 하는 방법
- Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780), 1612.
- 보통 분류문제에 적용



AdaBoosting

- 기본원리

- 이전 learner에 의해서 예측이 제대로 잘 되지 않은 data points의 weight 값을 크게해서 다음 단계에서는 그 data points를 더 많이 고려하게끔 만드는 것
- 틀린 것은 가중치를 많이주고, 맞은 것은 가중치를 적게 주겠다라는 뜻
- weight가 큰 관측치의 경우 다음 학습에서 더 많이 고려 됨



AdaBoosting

- 종속변수가 취할 수 있는 값 $\Rightarrow +1, -1$
- 작동순서
 - 1. 학습데이터에 있는 모든 관측치의 weight를 동일하게 설정 $\Rightarrow 1/n$
 - 가중치의 합 = 1이 되게 함
 - 2. 각 base learner에 대해서 아래 과정 반복
 - Subsampling with replacement using the weights
 - 가중치가 높을수록 subsample에 뽑힐 확률이 큼
 - 학습 with the subsample
 - 학습 결과를 original dataset에 적용해서 error 정도를 계산
 - 전체 에러 계산
 - 전체 에러 (Total error) = # of misclassified instances / n
 - 모형의 성능 계산 (모형 t)
 - $\alpha_t = \frac{1}{2} \ln\left(\frac{1-TE}{TE}\right)$
 - Update the weights



AdaBoosting

- 작동순서 (cont'd)
 - Update the weights
 - For misclassified instances: $w_t = w_{t-1} \cdot e^{\alpha_t}$
 - For correctly classified instances $w_t = w_{t-1} \cdot e^{-\alpha_t}$
 - 의미: 잘못 예측한 것의 가중치를 더 크게 한다.
 - $t = t + 1$
 - 다시 처음단계로 (앞의 2 step)
- 새로운 관측치에 대한 종속변수 값 예측
 - $\text{sign}(\sum_{t=1}^T \alpha_t \cdot M_t(x))$



AdaBoosting

- Python code
 - “AdaBoosting_example.ipynb”



Review

- Decision Tree
 - Weak learner
 - 하나의 모형을 사용하는 경우, 모형의 성능이 좋지 못하다.
- Ensemble 방법
 - 여러 개의 weak learner 를 사용하는 방법
 - 종류
 - Bagging
 - Random Forest
 - Boosting
 - Adaptive boosting
 - Gradient boosting



oob_score

- Out of Bag data points
 - 각 subsample에 포함되지 않은 관측치들
 - 이를 평가데이터로 사용 가능
 - Example dataset

나이	도시거주여부	흡연여부	폐암 여부
50	Yes	Yes	Yes
41	No	No	No
34	No	Yes	Yes
25	Yes	No	No

oob_score

- Example (cont'd)
 - 첫 번째 subsample: dp1, dp1, dp2, dp3
 - Then oob data point(s): dp4
 - Suppose that we use five subsamples, that is, five DT models, and dp4 is oob data point in the first, second, and fifth subsamples.
 - Suppose that the dependent variable of dp4 in each subsample (i.e., 1st, 2nd, and 5th) is predicted as below:

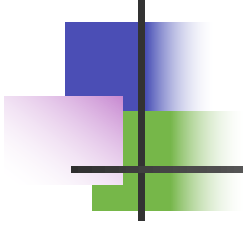
Decision Tree	DV of dp4
1	No
2	No
5	Yes

최종 예측 = No



oob_score

- 다른 oob data points 에 대해서도 동일한 작업 수행
- $\text{oob_score} = \# \text{ of 정확하게 예측된 oob data points} / \# \text{ of total oob data points}$



Gradient boosting



Gradient boosting

- 기본원리

- 각 단계에서의 tree (혹은 learner)가 이전 단계의 tree가 설명하지 못한 부분, 즉 나머지 (residual)를 추가적으로 설명하는 것
- Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189-1232.

- 설명

- 회귀문제를 예로 설명
- 분류문제의 경우도 비슷하게 작동



Gradient boosting

- 1단계
 - Fit the model (i.e., a decision tree) into the data
 - Obtain predicted values for data points
 - $\hat{y}_{1,i}$ 이것이 첫번째 단계에서의 예측치가 됨
- 2 단계에서의 예측치
 - 앞 단계의 모형이 설명하지 못한 부분(나머지)을 구함
 - Residuals: $r_{1,i} = y_i - \hat{y}_{1,i}$
 - 위에서 얻은 r을 종속변수로 다시 DT 모형을 적용 => 예측치를 구함, $\hat{r}_{1,i}$
 - 이전 단계에서의 예측치 (즉, $\hat{y}_{1,i}$)에 $\hat{r}_{1,i}$ 를 더해서 종속변수의 예측치를 업데이트 함
 - 즉, $\hat{y}_{2,i} = \hat{y}_{1,i} + \hat{r}_{1,i}$
 - 이러한 과정을 반복
- 최종 예측치: $\hat{y}_i = \hat{y}_{1,i} + \sum_{t=1}^T \hat{r}_{t,i}$
 - Learning rate를 적용하는 경우: $\hat{y}_i = \hat{y}_{1,i} + \eta \sum_{t=1}^T \hat{r}_{t,i}$ (learning rate를 작게 하는 경우, 보통은 n_estimator의 값을 크게 하는 것이 필요)



Gradient boosting

- sklearn class: GradientBoostingRegressor
 - <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html>
- Python code
 - See "Gradient_Boosting_example.ipynb"
 - Dataset:
<https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset>
 - 분석 단위: 도시
 - 종속변수: 일별 자전거 대여수



Gradient boosting

- Important hyperparameters
 - learning_rate (also known as shrinkage)
 - It shrinks the contribution of individual trees so that no tree has too much influence when building the model.
 - learning_rate_values = [0.001, 0.01, 0.05, 0.1, 0.15, 0.2, 0.3, 0.5, 1.0]
 - subsample
 - The fraction of samples to be used for fitting the individual base learners.
 - subsamples = [1, 0.9, 0.8, 0.7, 0.6, 0.5]
 - subsample 을 사용하면 (보통) 일반화 정도가 커진다.



Gradient boosting

- Gradient boosting for classification
 - The overall process is similar to that of a regression problem.
 - Refer to the following blogs, if you are interested.
 - <https://tyami.github.io/machine%20learning/ensemble-5-boosting-gradient-boosting-classification/>
 - <https://blog.paperspace.com/gradient-boosting-for-classification/>



Variants of gradient boosting

- Variants of gradient boosting
 - XGBoosting
 - Light GBM
 - Cat Boosting



XGBoosting

- XGBoosting (eXtreme Gradient Boosting)
 - Presented by Tianqi Chen in 2016
 - <https://arxiv.org/pdf/1603.02754.pdf>
 - <https://xgboost.readthedocs.io/en/latest/tutorials/model.html>
 - Gradient Boosting의 확장을 통한 속도와 성능의 개선
 - 중요 개선 부분: Regularization (L1 or L2)



XGBoosting

- XGBoosting (Extreme Gradient Boosting)
 - Regularization term을 포함
 - 각 단계에서의 objective function
 - $\text{obj}^{(t)} = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t)}) + \Omega(f_t)$
 - $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$
 - $l(y_i, \hat{y}_i^{(t)})$ 은 일반적으로 squared error 즉, $(y_i - \hat{y}_i^{(t)})^2$
 - At t, find f_t that minimizes the objective function.
 - $\Omega(f_t)$ 은 모형의 복잡성과 각 Tree의 기여도 관련된 것



XGBoosting

- Python code
 - Installation
 - pip install xgboost
 - “XGBoost.ipynb”
 - Parameter 관련 내용
 - <https://xgboost.readthedocs.io/en/stable/parameter.html>

Light GBM

- Light GBM (Light Gradient Boosting Model)

- Released by Microsoft in 2017 (XGBoosting의 확장)
- 자세한 설명은 <https://lightgbm.readthedocs.io/en/latest/> 참고
- 주요 특징

- Leaf-wise split

모든 leaf node 중에서 loss reduction에 가장 큰 기여를 하는 노드를 split



- Gradient-based One-Side Sampling (GOSS)

- GOSS keeps all the instances with large gradients and performs random sampling on the instances with small gradients. 즉, 에러 정도가 큰 관측치는 모두 사용하고, 에러 정도가 작은 관측치의 일부만 random sampling 방법을 통해서 사용
 - Large gradient => large error => (모델의 성능을 개선하는데 있어서) 더 중요한 역할



Light GBM

- 설치 필요
 - `pip install lightgbm`
- Python code
 - `"lightgbm.ipynb"`



CatBoost

- CatBoost
 - Category Boost
 - It was released by a company named Yandex in 2017.
 - Extension of gradient boosting
 - Works well with categorical variables
 - <https://catboost.ai/docs>



[참고] 서로 다른 모형들 사용하기

- 분류문제: VotingClassifier
 - <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html>
 - See “Voting_classifier_example.ipynb”
- 회귀문제: VotingRegressor
 - <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingRegressor.html>