Hyperparameter tuning

how do we tune hyperparameter

1. 가장 influential한 parameter를 정하자.
2. 그들을 바꿀 때마다 training에 어떤 영향을 주는지 파악한다.
3. Tune them! : manually or automatically

result

hyperparameter를 fitting한 결과로

- underfitting
- good
- overfitting

이다. 결과에 따라 적절히 수정해나간다.

tree

- GBDT : XGBoost, LightGBM, CatBoost
  - GBDT는 상대적으로 hyperparameter에 민감하다.
- Random forest

GBDT

- 기본적인 hyperparameter를 알아보자.
- 아래 XGBoost, LightGBM 순서대로 매칭되는 hyperparameter
- XGBoost
  - `max_depth` : tree 만들 때 깊이 (7로 시작하길 추천한다고 한다, 크게하면 시간이 너무 걸린다고 한다)
  - `subsample` : object를 sampling하여 사용
  - `colsample_bytree, colsample_bylevel`
  - `min_child_weight` : 0으로 갈수록 less constrained (가장 중요한 parameter 중 하나라고 한다), `lambda, alpha`
  - `eta` : learning rate (클수록 train에 빠르게 fit되겠다, overfitting 주의, 너무 크면 converge 안할수도 있겠다), `num_round`
- LightGBM
  - `max_depth/num_leaves`
  - `bagging_fraction`
  - `feature_fraction`
  - `min_data_in_leaf, lambda_l1, lambda_l2`
  - `learning_rate, num_iterations`
- sklearn.RandomForest
  - GB랑 다르게 각 tree가 독립이니까 tree 많이 만들어도 overfitting되는 정도는 약하다. 대신에 굳이 필요없는 training 시간을 피하는게 좋을 것이다.
  - `N_estimators`
  - `max_depth` : 7로 시작하길 추천 (GBDT보다는 더 깊게 해도 된다)
  - `max_features`
  - `min_samples_leaf` : GBDT에서 min_child_weight와 같은 역할 (regularization)
  - criterion : Gini, Entropy인데 Gini가 더 좋은 것 같다고 한다.
  - RF에서 tree 갯수 정할때 사용 코드 : fit하고 `rf.estimators_` 이용

NN (fully connected)

- layer의 neurons 갯수
- layer의 갯수
- optimizer
  - SGD+momentum, Adam, Adadelta, Adagrad...
- Batch size : 클수록 overfitting쪽
- Learning rate
- Regularization
  - dropout, L1, L2...

Linear model

- tuning할게 거의 없다.
- 주로 Regularization parameter

practical guide

- competition goal을 정해라.
  - problem 해결책
  - software 사용법
  - medal hunt
- after enter competition
  - organize idea in some structure
  - select the most important and promising ideas
  - try to understand the reasons why something does/doesn't work
- data loading
  - csv/txt file을 hdf5/npy로 바꾸면 더 빠른 loading이 가능하다.
  - pandas가 default로 data를 높은 bit에 저장하니 가능하면 낮추자.
  - Large dataset은 chunk로 나눠서 다룰수 있다.
- 주로 LightGBM으로 빠르게 시작한다. 처음부터 tuning까지 하는 loop를 할 필요없다. 처음에는 빠르게 모델링해서 이해해보자.
- Fast and dirty always better
- Initial pipeline
  - simple하게 시작해서 전체적인 pipeline을 만들자.
  - From simple to complex
- reproducible
  - Fix random seed, resuse code
- Read paper
- Start with EDA and a baseline
  - check validation is stable
- Feature을 많이 만든다. 그리고 hyperparameter등으로 regularization한다.
- jupyter notebook에 매크로로 주요 coding 만들어라.

pipeline tip

- 전체적인 순서
  - 문제 이해
  - EDA
  - cv strategy 정하기
  - FE
  - Modeling
  - Ensemble

- EDA
  - 각 변수 histogram
  - train, test가 비슷한지 확인
  - binning numerical features
  - correlation
- cross validation strategy
  - 매우 중요
  - time ? Time-based validation
  - different entity? Stratified validation
  - Random validation
  - 이런거 다 섞어서
- FE
  - 다양하다!!
- Modelling
  - 다양하다!!
- Ensemble
  - low-correlated prediction을 average하는게 좋은 성능을 낸다.
  - 다양한 model을 사용

참고링크

- sklearn hyperparameter tuning
- hyperopt hyperparameter tuning
- jupyter-notebook-tips-tricks

FE

- Statictics and distance based features
  - Groupby
    - user_id로 groupby
    - standard deviation 등등
  - Neighbors
    - 거리와 관련한 feature를 이용 예를 들어, 500m안에 지하철 몇개 있는지
    - mean encoding해서 KNN 등등
- Matrix factorization
  - recommendation
  - SVD, PCA
  - PCA할때,
    - train, test set을 합쳐서 pca.fit하고 각자 transform한다.
- Feature interaction
  - categorical feature의 combination
    - combination만들고 one-hot encoding해도 되고
    - one-hot하고 column끼리 곱해도 된다.
  - 곱, 합, 차, 나누기 다 사용해도 된다!
  - 여러개의 feature를 만들고 feature selection을 한다.
  - tree model에서 만드는 interaction을 이용
    - sklearn : `tree_model.apply()`
    - xgboost : `booster.predict(pred_leag=True)`
- t-sne
  - manifold learning methods
  - hyperparameter(`Perplexity`)를 잘 선택
  - EDA, FE에 사용가능

참고링크

- facebook Research's paper about extracting categorical features from trees

Ensembling

- intro
  - averaging : 각 model의 output을 average하는 것
    - averaging (or blending)
    - weighted averaging
    - conditional averaging
  - bagging
  - boosting
  - stacking
  - stacknet
- bagging
  - RF같이 different version의 같은 model을 average
  - how?
    - seed를 바꾸기
    - row sampling or bootstrapping
    - shuffling
    - column sampling
    - model specific parameter
    - control number of models
- boosting
  - weighted averaging of model in sequentially
  - main type
    - weight based : error에 따라 각 sample에 weight를 준다.
      - parameter
        - learning rate (eta) : 각 model을 average할 때, 앞에 곱해지는 값
        - number of estimators : 이게 커지면 learning rate는 줄여야겠다.
      - adaboost
    - residual based : error를 새로운 target 으로 만든다.
      - parameter
        - learning rate (eta)
        - number of estimators
        - sampling(row, column)
      - Xgboost, Lightgbm, H2O's GBM, Catboost, sklearn's GBM
- stacking
  - making predictions of a number of models in a hold-out sete and then using a dfferent model to train on these predictions
  - how? (meta model)
    1. train set을 두 개로 나눈다.
    2. 첫번째 set으로 여러개의 base model을 train 시킨다.
    3. 이것들로 두번째 set을 predict한다.
    4. 해당 prediction 값들을 input으로 하여 highter level model을 train 한다.
  - time sensetive data인 경우는 주의해야겠군
  - 모델의 다양성도 중요하다.
- Stacknet
  - neural net에서 각 node는 simple linear model이다. 이를 대신해서 다양한 모델을 사용할 수 있다는 아이디어를 이용한다.
  - How?

        - backpropagation을 사용하지 않는다.
- Ensembling tips
    - 1st level tips
        - diversity on algorithm!
            - 2~3개 GBDT
            - 2~3개 neural net
            - 1~2개 RF
            - 1~2개 linear model, svm
            - 1~2개 KNN
            - 1개 nonlinear kernel svm
        - diversity on input data
            - categorical feature : one hot, label, target ..encoding
            - numerical : outlier, binning, derivative, percetiles...
            - interaction : col끼리 연산, groupby, unsupervised
        - subsequent level tip
            - simpler algorithm 사용
            - FE
                - pairwise difference between meta features
                - row-wise statictics
                - feature selection
            - for every 7.5 model in previous level we add 1 in meta
            - data leakage 조심

Catboost

- categorical feature
    - one hot encoding 기능 제공
    - number of apprearace encoding 기능제공
    - data의 random permutation하여 target encoding보다 더 좋은 encoding 제공
    - feature combination 기능 제공 (numerical도 가능)
- symmetric decision trees
    - predcition speed가 빠르다고 한다.
    - hyperparameter의 영향을 덜 받는 형태
- ordered boosting으로 overffiting을 막는다.
    - 다른 boosting은 leaf의 모든 object의 gradient를 average하는데
    - catboost는 past objects를 permutation하여 사용
- overfitting detector
- evaluating custom metrics during training
- Nan features support
- cross validation
- feature importance
- training parameter
    - number of tree + learning rate
    - tree depth
    - L2
    - bagging temperature