LightGBMのlinear_treeパラメータを触ってみる

2022-08-29 09:00　Morita Ryo
Tech, DataAnalytics

こんにちは、データソリューション部の森田です。

普段はデータサイエンティストとして社内の生産・調達領域のデータ利活用の取り組みに携わっています。

kaggleなどのデータ分析コンペに参加するのが趣味なので、今回はデータ分析コンペで良く用いられるLightGBMというアルゴリズムを使って少し遊んでみます。

目次 [非表示]

1.はじめに
2.勾配ブースティング決定木の仕組みとその拡張
3.実験
4.おわりに
5.参考

はじめに

kaggleをはじめとするデータ分析コンペティションでは、勾配ブースティング決定木(GBDT , Gradient Boosting Decision Tree)と呼ばれる機械学習アルゴリズムが高精度な予測が可能であるために多くのタスクに対して適用されています。

特にLightGBMは学習速度が比較的高速でありながら高精度の予測ができることが多く、参加者の多くが使用するポピュラーなアルゴリズムです。

また、LightGBMはデータ分析コンペだけではなく実務において非常に強力なアルゴリズムであり、弊社でも使用実績があります。(FORXAI Recognitionの骨格検出アルゴリズムによる MFP組み立て工程改善)

勾配ブースティング決定木の仕組みとその拡張

LightGBMをはじめとする勾配ブースティング決定木ですが、弱学習器に決定木モデルが使用されています。

勾配ブースティング決定木は、1. ある決定木を学習させる, 2. 1つ前の決定木が学習できなかった残差を次の決定木に学習させる、という処理を精度が頭打ちになるまで繰り返し行うことで学習がされます。

このように複数の決定木を順々に学習させ、残差を小さくしていくことで高精度な予測を実現しています。

通常の場合、勾配ブースティング決定木モデルにおける各決定木の葉ノードの予測値は定数値になっていますが、これを線形モデルに置き換える拡張(GBDT-PL)が提案されています。

各葉ノードで定数による予測をしていたところを線形モデルに予測させることになるので、より柔軟にデータにフィットさせることができ、精度向上に繋がりそうです。

LightGBMの場合にはlinear_treeというハイパーパラメータをTrueに設定する(デフォルトはFalse)ことで、これが実現できます(参考リンク : 公式ドキュメント, github pull request)。 ハイパーパラメータを変えるだけなので非常に簡単ですね。

以下ではlinear_treeがFalseの場合とTrueの場合で精度の比較をしてみます。

実験

今回データはTabular Playground Series - Feb 2021のデータを使用して実験します。

本データでは目的変数は連続値であるため、回帰問題としてRMSEを評価関数とします。また説明変数は10個のカテゴリ変数と14個の連続変数の計24変数から構成されています(今回、特徴量エンジニアリングは特にせず。連続変数に関しては標準化有り無しの両方で検証)。

学習・検証はtrain.csvに対して実施しています。

LighGBMのハイパーパラメータはlinear_treeのTrue/False以外は以下の値で固定しました。

```
MODEL_PARAMS = {
    "boosting_type" : "gbdt",
    "objective" : "rmse",
    "max_depth" : 6,
    "num_leaves" : 40,
    "learning_rate": 0.1,
    "seed" : 42,
    "linear_tree": False, # ここをTrue/Falseで比較
}
```

参考までに学習用関数も載せておきます。

```
def fit_lgb(train_X, train_y, model_params, train_params,cv, cat_cols = None):
    oof_pred = np.zeros(len(train_y))

    # クロスバリデーション
    for fold, (train_idx, val_idx) in enumerate(cv):
        print("*" * 100)
        print(f"FOLD : {fold + 1} / {len(cv)}")

        # train/val分割
        tra_x, tra_y = train_X.iloc[train_idx], train_y[train_idx]
        val_x, val_y = train_X.iloc[val_idx], train_y[val_idx]

        tra_data = lgb.Dataset(tra_x, label = tra_y, categorical_feature= cat_cols)
        val_data = lgb.Dataset(val_x, label = val_y, categorical_feature= cat_cols)

        # 学習
        model = lgb.train(
            params = model_params,
            train_set = tra_data,
            valid_sets = [tra_data, val_data],
            **train_params)

        # 検証データの予測
        val_pred = model.predict(val_x)
        oof_pred[val_idx] = val_pred

    # 全体のRMSEスコア計算
    whole_rmse= mean_squared_error(train_y, oof_pred, squared = True)
    print("*" * 20,"FINISHED", "*" * 20)
    print(f"whole_rmse : {whole_rmse:.5f}")

    return whole_rmse
```

さて肝心の結果ですが、以下のようになりました。

実験	linear_Tree	説明変数変数の標準化	RMSEスコア	学習時間(秒)
0	FALSE	なし	0.7139	41.7
1	FALSE	あり	0.7140	45.9
2	TRUE	なし	0.7138	50
3	TRUE	あり	0.7140	47.2

今回のデータセットに対しては精度が大きく向上することはありませんでした。また学習時間も短縮されない結果でした(ただしbest iterationまでの木の本数は減っていたので、葉の予測値を線形回帰モデルにしたことで1本1本の木はより柔軟になっているといえそうです。) また、今回のデータが各変数間でスケールに差がないものだったため標準化を実施しなくても大きな問題はない結果でしたが、基本的には標準化は実施するのが良いでしょう(公式ドキュメントでも標準化が推奨されている)。

おわりに

なんだかパっとしない結果になってしまいました。

ただ、ハイパーパラメータを変えるだけでLightGBM版のGBDT-PLが使えるというのは気軽で良いと思います。 学習時間や精度に対する影響はデータの種類や量によって効果の程度に差はありそうなので、また他のデータでも試してみようかなと思います。

以上です。

参考

コニカミノルタは画像IoTプラットフォームFORXAIを通じて、お客様やパートナー様との共創を加速させ、技術・ソリューションの提供により人間社会の進化に貢献してまいります。

コニカミノルタでは、一緒に働く仲間を募集しています。本記事で興味を持っていただけた方は下記のリンクより応募いただけると大変うれしいです。

新卒採用に関する情報については以下の採用情報ページをご覧ください

新卒採用情報 – 採用情報 | コニカミノルタ

コニカミノルタの新卒採用サイトです。募集要項や募集職種などの採用情報から、プロジェクト紹介、社員インタビューなどを掲載しています。ぜひご覧ください。

KONICA MINOLTA

中途採用に関する情報については以下の採用情報ページをご覧ください。

キャリア採用情報 – 採用情報 | コニカミノルタ

コニカミノルタキャリア採用情報　現在の募集職種にはこちらからエントリー可能です。募集要項、先輩インタビュー、人事部からのメッセージなど掲載。

KONICA MINOLTA

Morita Ryo

データサイエンスセンター データソリューション部所属。2018年コニカミノルタへ新卒入社し、入社後3年半はプロダクションプリンターの製品開発に従事。その後データサイエンティストに転身。現職では、生産・調達領域のデータ利活用テーマを推進しており、故障検知・要因分析、製造パラメータの最適化等のテーマ等を担当。

前の記事

技術開発本部 2021年度入社 新人発表会　～運営編～

次の記事

新規事業での品質保証評価について