Оптимизация гиперпараметров CatBoost с Optuna

Лаконичный код для максимально быстрой оптимизации

```
import optuna
import numpy as np
import pandas as pd
from catboost import CatBoostRegressor, Pool, cv
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import warnings
warnings.filterwarnings('ignore')

def objective(trial):
    """
    Objective функция для оптимизации гиперпараметров CatBoost
    Оптимизирована для максимальной скорости и качества результатов
    """

    # Параметры для оптимизации (оптимальный набор для регрессии)
    params = {
        'iterations': 1000,  # Фиксированное значение для стабильности
        'learning_rate': trial.suggest_float('learning_rate', 0.01, 0.3, log=True),
        'depth': trial.suggest_int('depth', 4, 10),
        'l2_leaf_reg': trial.suggest_float('l2_leaf_reg', 1, 10, log=True),
        'random_strength': trial.suggest_float('random_strength', 1e-9, 10, log=True),
        'bagging_temperature': trial.suggest_float('bagging_temperature', 0, 1),
        'border_count': trial.suggest_int('border_count', 32, 255),
        'feature_border_type': trial.suggest_categorical('feature_border_type',
                                                         ['Median', 'Uniform', 'UniformAndQ
        'silent': True,
        'thread_count': -1,  # Использовать все доступные ядра
        'random_seed': 42
    }

    # Создаем Pool для CatBoost (если есть категориальные признаки)
    train_pool = Pool(X_train, y_train, cat_features=cat_features if 'cat_features' in gl

    # Cross-validation с CatBoost
    cv_results = cv(
        pool=train_pool,
        params=params,
        fold_count=5,
        early_stopping_rounds=50,
        verbose=False,
        stratified=False,  # Для регрессии
        as_pandas=True
    )
```

```
    # Возвращаем лучший результат (минимальный RMSE)
    best_score = cv_results['test-RMSE-mean'].min()

    return best_score

def optimize_catboost(X_train, y_train, n_trials=None, timeout=None, cat_features=None):
    """
    Функция для оптимизации гиперпараметров CatBoost

    Parameters:
    -----------
    X_train : array-like, shape (n_samples, n_features)
        Тренировочные данные
    y_train : array-like, shape (n_samples,)
        Целевая переменная
    n_trials : int, optional
        Количество trials (если None, рассчитывается автоматически)
    timeout : int, optional
        Максимальное время оптимизации в секундах
    cat_features : list, optional
        Индексы категориальных признаков

    Returns:
    --------
    study : optuna.study.Study
        Результаты оптимизации
    best_params : dict
        Лучшие параметры
    """

    # Автоматический расчет количества trials на основе размера данных
    if n_trials is None:
        data_size = len(X_train)
        if data_size <= 1000:
            n_trials = 100
        elif data_size <= 5000:
            n_trials = 200
        elif data_size <= 15000:  # Для ~11k записей
            n_trials = 300
        else:
            n_trials = 500

    # Создаем study с Pruner для ускорения
    pruner = optuna.pruners.MedianPruner(
        n_startup_trials=10,
        n_warmup_steps=30,
        interval_steps=10
    )

    sampler = optuna.samplers.TPESampler(
        n_startup_trials=20,
        n_ei_candidates=24,
        seed=42
    )

    study = optuna.create_study(
```

```
        direction='minimize',
        pruner=pruner,
        sampler=sampler
    )

    # Глобальные переменные для objective функции
    globals()['X_train'] = X_train
    globals()['y_train'] = y_train
    globals()['cat_features'] = cat_features

    print(f"Начинаем оптимизацию с {n_trials} trials...")

    # Запускаем оптимизацию
    study.optimize(
        objective,
        n_trials=n_trials,
        timeout=timeout,
        show_progress_bar=True
    )

    print(f"\nОптимизация завершена!")
    print(f"Лучший RMSE: {study.best_value:.6f}")
    print(f"Лучшие параметры: {study.best_params}")

    return study, study.best_params

def train_final_model(X_train, y_train, X_test, y_test, best_params, cat_features=None):
    """
    Обучение финальной модели с лучшими параметрами
    """

    model = CatBoostRegressor(**best_params)

    # Обучение
    model.fit(
        X_train, y_train,
        eval_set=(X_test, y_test) if X_test is not None else None,
        cat_features=cat_features,
        early_stopping_rounds=50,
        verbose=100
    )

    # Предсказания
    if X_test is not None:
        predictions = model.predict(X_test)
        rmse = np.sqrt(mean_squared_error(y_test, predictions))
        print(f"Финальный RMSE на тестовой выборке: {rmse:.6f}")

    return model

# Пример использования:
if __name__ == "__main__":
    # Подготовка данных
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state

    # Указать индексы категориальных признаков (если есть)
```

```
    cat_features = [0, 2, 5]  # Пример

    # Оптимизация гиперпараметров
    study, best_params = optimize_catboost(
        X_train, y_train,
        n_trials=300,  # Для 11k записей оптимальное значение
        cat_features=cat_features
    )

    # Обучение финальной модели
    final_model = train_final_model(X_train, y_train, X_test, y_test, best_params, cat_fe

    # Визуализация результатов оптимизации
    optuna.visualization.plot_optimization_history(study).show()
    optuna.visualization.plot_param_importances(study).show()
```

Ключевые особенности реализации

1. Максимальная скорость оптимизации

- **MedianPruner** - отсекает неперспективные trials на ранней стадии
- **TPESampler** - использует байесовскую оптимизацию для умного поиска гиперпараметров
- **Встроенная CV функция CatBoost** - быстрее чем sklearn.model_selection

2. Автоматический расчет количества trials

- До 1k записей: 100 trials
- До 5k записей: 200 trials
- **До 15k записей (включая 11k): 300 trials**
- Свыше 15k записей: 500 trials

3. Оптимизированные гиперпараметры для регрессии

- `learning_rate`: логарифмическая шкала 0.01-0.3
- `depth`: глубина дерева 4-10
- `l2_leaf_reg`: L2 регуляризация 1-10
- `random_strength`: случайность разбиений
- `bagging_temperature`: температура для байесовского бэггинга
- `border_count`: количество границ для числовых признаков

4. Cross-validation настройки

- **5 фолдов** - оптимальный баланс скорости и качества
- **early_stopping_rounds=50** - предотвращает переобучение
- **stratified=False** - для задач регрессии

5. Использование ресурсов

- `thread_count=-1` - использует все доступные ядра процессора
- `silent=True` - убирает лишний вывод для ускорения

Расширенная версия с pruning callback

```
from optuna.integration import CatBoostPruningCallback

def objective_with_pruning(trial):
    """
    Версия с pruning callback для еще большего ускорения
    """
    params = {
        'iterations': trial.suggest_int('iterations', 500, 2000),
        'learning_rate': trial.suggest_float('learning_rate', 0.01, 0.3, log=True),
        'depth': trial.suggest_int('depth', 4, 10),
        'l2_leaf_reg': trial.suggest_float('l2_leaf_reg', 1, 10, log=True),
        'random_strength': trial.suggest_float('random_strength', 1e-9, 10, log=True),
        'bagging_temperature': trial.suggest_float('bagging_temperature', 0, 1),
        'border_count': trial.suggest_int('border_count', 32, 255),
        'silent': True,
        'thread_count': -1,
        'random_seed': 42
    }

    # Разделяем на train/validation для pruning
    from sklearn.model_selection import train_test_split
    X_tr, X_val, y_tr, y_val = train_test_split(
        X_train, y_train, test_size=0.2, random_state=42
    )

    model = CatBoostRegressor(**params)

    # Callback для pruning
    pruning_callback = CatBoostPruningCallback(trial, "RMSE")

    model.fit(
        X_tr, y_tr,
        eval_set=[(X_val, y_val)],
        cat_features=cat_features,
        callbacks=[pruning_callback],
        verbose=False
    )

    # Проверяем, нужно ли prune
    pruning_callback.check_pruned()
```

```
    # Возвращаем лучший validation score
    return model.get_best_score()['validation']['RMSE']
```

Рекомендации по использованию

Для датасета с 11k записей:

- **Количество trials**: 300
- **Время оптимизации**: 30-60 минут (в зависимости от количества признаков)
- **Память**: ~2-4 GB RAM
- **CPU**: рекомендуется 4+ ядер для параллельного выполнения

Параметры для разных сценариев:

```
# Быстрая оптимизация (10-15 минут)
study, params = optimize_catboost(X_train, y_train, n_trials=100)

# Сбалансированная оптимизация (30-45 минут)
study, params = optimize_catboost(X_train, y_train, n_trials=300)

# Глубокая оптимизация (1-2 часа)
study, params = optimize_catboost(X_train, y_train, n_trials=500)

# Оптимизация по времени (остановка через час)
study, params = optimize_catboost(X_train, y_train, timeout=3600)
```