Otimização de Hiperparâmetros

Conceito

As técnicas de otimização de hiperparâmetros são peças fundamentais dentro de um projeto de dados. É através delas que conseguimos extrair o melhor desempenho do modelo e conseguimos elevar o nível da solução de dados que estamos construindo.

É importante salientar e clarear as características entre parâmetros e hiperparâmetros dentro dos processos de modelagem que devido utilizarem a mesma palavra, são facilmente confundidos e definidos como a mesma coisa de maneira errada.

Os parâmetros, por sua vez, são ajustados diretamente por processos de aprendizado dos algoritmos e possuem influência direta no desempenho deles. As fronteiras de vizinhos no KNN, os pesos definidos dentro de redes neurais, coeficientes de regressão linear, todos estes são exemplos de parâmetros que se ajustam no momento do treinamento do modelo com base em um conjunto de dados.

Por outro lado, os hiperparâmetros são variáveis do algoritmo que são definidas antes do treinamento. Os hiperparâmetros são características construtivas do algoritmo. Neste caso, utilizando os exemplos citados para exemplificar os parâmetros, nos hiperparâmetros consideramos a quantidade de vizinhos definido como K no KNN, a quantidade de camadas que vamos utilizar nas redes neurais e as métricas de desempenho de regressões lineares. Os hiperparâmetros têm forte influência no desempenho dos modelos e por isso, a escolha e o valor agregado aos mesmos devem ser definidos de maneira criteriosa.

Para produzir a otimização de um determinado *hiperparâmetro* é necessário realizar o treinamento do modelo diversas vezes. Somente assim, podemos realizar os testes e validar se o desempenho do nosso modelo obteve melhorias com base nos hiperparâmetros e valores definidos para eles. Estes processos de treinar várias vezes o modelo e posteriormente testar os hiperparâmetros do algoritmo treinado possui alto custo computacional e tende a ser um pouco demorado.

A fim de resolver esses gargalos e problemas no processo de melhoria da modelagem, os algoritmos de otimização foram criados para auxiliar-nos no desenvolvimento de processos de definição de hiperparâmetros e seus respectivos valores que são tão importantes em projetos de dados.

Principais Algoritmos de Otimização

Abaixo, são compartilhados os principais algoritmos de otimização de hiperparâmetros:

- *Grid Search*: Possivelmente o caso mais ingênuo e mais simples. Este algoritmo de busca recebe um conjunto de valores de um ou mais hiperparâmetros e testa todas as combinações possíveis. Desta maneira, ele tabula o desempenho de cada configuração e ao final, indica a melhor opção de configuração para ser utilizada pelo algoritmo.

- *Random Search*: O *Random Search* é bastante semelhante ao *Grid Search*. A principal diferença entre eles é que ao invés de testar todas as combinações possíveis de configuração entre hiperparâmetros e valores que ele pode assumir, o *Random Search* estabelece o teste de combinações aleatórias de acordo com um número especificado de amostras. Quando a quantidade de dados envolvidos e a quantidade de parâmetros for muito grande, este algoritmo se torna uma melhor opção do que *Grid Search*.

- *Bayes Search*: Utilizando a busca bayesiana, o algoritmo *Bayes Search* estima a melhor combinação ou configuração de hiperparâmetros fundamentados nas distribuições criadas de combinações testadas anteriormente. Internamente, este algoritmo encontra as regiões de menor confiança na distribuição gerada e dentro destas regiões ele verifica qual das mesmas contém um valor mais elevado para o desempenho do algoritmo. Desta maneira, a cada iteração da busca bayesiana, as configurações ou combinações de melhor desempenho são aplicadas aos hiperparâmetros e por sua vez, a distribuição do desempenho do modelo é frequentemente atualizada. Em termos práticos, o ganho de desempenho proporcionado por este algoritmo é muito superior quando comparado com o *Grid Search* e o *Random Search*. Assim, ele é um algoritmo indicado para tarefas de otimização mais complexas. No entanto, é válido destacar que o *Bayes Search* consome mais tempo do que os algoritmos *Grid Search* e *Random Search*.

Campos de Aplicação

O uso da técnica de otimização de hiperparâmetros pode e deve ser aplicado em qualquer projeto de dados. Independente do problema que esteja sendo tratado, é importante explorar os algoritmos de *machine learning* como também os hiperparâmetros que os constituem para que possam existir meios de realizar o *tuning* ou melhorias do desempenho do modelo sendo testado e assim, alterando o seu comportamento padrão.

Diversos algoritmos aplicados a problemas de aprendizado supervisionado, não supervisionado, por reforço e *deep learning* possuem tanto parâmetros para realização da modelagem adotando um comportamento padrão como também hiperparâmetros que podem ser ajustados para adequar o desempenho dele ao seu problema específico.

É importante destacar que para modificações em poucos parâmetros esse processo pode ser feito manualmente. Porém, à medida que a quantidade de hiperparâmetros aumenta, a quantidade de testes que temos que fazer aumentará exponencialmente e o uso das técnicas exploradas nesta sessão serão uteis para proporcionar o melhor uso do tempo de modelagem como também produzir o melhor desempenho possível dos algoritmos utilizados.

Aplicação Prática

Através do link abaixo, pode ser feito o download do(s) script(s) python criado(s) para exemplificar uma abordagem do uso prático da otimização de hiperparâmetros como também o(s) respectivo(s) conjunto(s) de dados utilizados.

Script Python

```
### Otimização de Hiperparâmetros
```

```
# Definindo variável seed
SEED = 123456

#### Importando a base dados da própria biblioteca Sklearn
from sklearn.datasets import load_breast_cancer

# Carregando dataset
df = load_breast_cancer()

# Importando biblioteca pandas
import pandas as pd

# Formatando dataset em um DataFrame e apresentado a 5 primeiras linhas
do mesmo
df_feature = pd.DataFrame(data=df['data'], columns=df['feature_names'])
df_feature.head()

# Definindo a variável target
df_targets = pd.Series(data=df['target'], name='benign')

# Apresentando os valores únicos da variável target
df_targets.unique()

# Atribuindo features na variável X e target na variável y
X = df_feature
y = df_targets

#### Carregando o algoritmo de Árvores de Decisão para ser o algoritmo
na qual vamos aplicar a otimização
# Carregando as bibliotecas
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_validate
import numpy as np

# Criando o modelo
modelo_tree = DecisionTreeClassifier()

# Aplicando a técnica de cross validade
results = cross_validate(modelo_tree, X, y, cv=5,
                scoring=('accuracy'),
                return_train_score=True)
print(f"mean_train_score {np.mean(results['train_score']):.2f}")
print(f"mean_test_score {np.mean(results['test_score']):.2f}")

#### Abordando o uso da otimização através do Grid Search
# Carregando a biblioteca GridSearchCV
```

```
from sklearn.model_selection import GridSearchCV

# Definindo parâmetros
relacao_parametros = {
  "max_depth" : [3, 5],
  "min_samples_split" : [32, 64, 128],
  "min_samples_leaf" : [32, 64, 128],
  "criterion" : ["gini", "entropy"]
}

# Criando modelo
modelo_tree = DecisionTreeClassifier()

# Aplicando o algoritmo com parâmetros definidos anteriormente
clf = GridSearchCV(modelo_tree, relacao_parametros, cv=5,
return_train_score=True, scoring='accuracy')

# Treinando o modelo
search = clf.fit(X, y)

# Capturando os resultados e os índices dos melhores parâmetros
results_GridSearchCV = search.cv_results_
indice_melhores_parametros = search.best_index_

# Apresentando a média de score de treino e teste produzida
print(f"mean_train_score {results_GridSearchCV['mean_train_score']
[indice_melhores_parametros]:.2f}")
print(f"mean_test_score {results_GridSearchCV['mean_test_score']
[indice_melhores_parametros]:.2f}")

# Apresentação dos parâmetros
results_GridSearchCV['params'][indice_melhores_parametros]

#### Abordando o uso da otimização através do Random Search
# Carregando as variáveis
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint

# Definindo relação de parâmetros
relacao_parametros_2 = {
    "max_depth" : randint(1, 10),
    "min_samples_split" : randint(32, 129),
    "min_samples_leaf" : randint(32, 129),
    "criterion" : ["gini", "entropy"]
}
```

```
# Criação do modelo
modelo_tree = DecisionTreeClassifier()

clf = RandomizedSearchCV(modelo_tree, relacao_parametros_2,
random_state=SEED, cv=5, return_train_score=True, n_iter=10,
scoring='accuracy')
search = clf.fit(X, y)
results_RandomizedSearchCV = search.cv_results_
indice_melhores_parametros = search.best_index_

# Apresentando a média de score de treino e teste produzida
print(f"mean_train_score
{results_RandomizedSearchCV['mean_train_score']
[indice_melhores_parametros]:.2f}")
print(f"mean_test_score {results_RandomizedSearchCV['mean_test_score']
[indice_melhores_parametros]:.2f}")

# Apresentação dos parâmetros
results_RandomizedSearchCV['params'][indice_melhores_parametros]
```

Materiais complementares

- Abhishek Thakur channel at Youtube: Hyperparameter Optimization: This Tutorial Is All You Need
- BERGSTRA, James et al. Algorithms for hyper-parameter optimization. Advances in neural information processing systems, v. 24, 2011.
- BARTMANN, Nico et al. Applied Predictive Process Monitoring and Hyper Parameter Optimization in Camunda. In: International Conference on Advanced Information Systems Engineering. Springer, Cham, 2021. p. 129-136.
- SUI, Guoxin; YU, Yong. Bayesian contextual bandits for hyper parameter optimization. IEEE Access, v. 8, p. 42971-42979, 2020.
- GUPTA, Jayesh. Exploration Study of Ensembled Object Detection models and Hyper Parameter Optimization. Exploration Study of Ensembled Object Detection models and Hyper Parameter Optimization, [S. l.], p. 2-9, 2 nov. 2021.
- OTIMIZACAO EVOLUTIVA DE HIPERPARAMETROS PARA MODELOS DE SERIES TEMPORAIS NEBULOSAS. Anais do 14º Simpósio Brasileiro de Automação Inteligente, [S. l.], p. 2-7, 1 out. 2019.
- SMITH, Michael R.; MARTINEZ, Tony; GIRAUD-CARRIER, Christophe. The potential benefits of filtering versus hyper-parameter optimization. arXiv preprint arXiv:1403.3342, 2014.

Referências

- BERGSTRA, James et al. Algorithms for hyper-parameter optimization. Advances in neural information processing systems, v. 24, 2011.
- BARTMANN, Nico et al. Applied Predictive Process Monitoring and Hyper Parameter Optimization in Camunda. In: International Conference on Advanced Information Systems Engineering. Springer, Cham, 2021. p. 129-136.
- SUI, Guoxin; YU, Yong. Bayesian contextual bandits for hyper parameter optimization. IEEE Access, v. 8, p. 42971-42979, 2020.

- GUPTA, Jayesh. Exploration Study of Ensembled Object Detection models and Hyper Parameter Optimization. Exploration Study of Ensembled Object Detection models and Hyper Parameter Optimization, [S. l.], p. 2-9, 2 nov. 2021.
- OTIMIZACAO EVOLUTIVA DE HIPERPARAMETROS PARA MODELOS DE SERIES TEMPORAIS NEBULOSAS. Anais do 14º Simpósio Brasileiro de Automação Inteligente, [S. l.], p. 2-7, 1 out. 2019.
- SMITH, Michael R.; MARTINEZ, Tony; GIRAUD-CARRIER, Christophe. The potential benefits of filtering versus hyper-parameter optimization. arXiv preprint arXiv:1403.3342, 2014.

Tópico anterior

Próximo Tópico