📙

A estratégia de Validação Holdout

Ciclo	Ciclo 05: As garantias de aprendizado
# Aula	28
Created	@February 16, 2023 4:33 PM
Done	☐
Ready	☑

Objetivo da Aula:

- [ ] A estratégia da validação holdout
- [ ] O modo de separação
- [ ] A sequência prática
- [ ] Os cuidados
- [ ] Prós & Contras
- [ ] Resumo
- [ ] Próxima Aula

Conteúdo:

▼ 1. A estratégia da validação holdout

Uma solução comum para o problema da adaptação do algoritmo, durante o experimento que busca o melhor valor para o parâmetro é a validação holdout.

Na validação holdout, você simplesmente separa um terceiro conjunto de dados, que será usado exclusivamente para selecionar os parâmetros que trazem a melhor performance ao algoritmo.

2. O modo de separação

A estratégia mais eficiente para garantir a generalização dos algoritmos de Machine Learning é a separação do conjunto de dados original em 3 subconjuntos: Treinamento, validação e teste.

2.1 O modo de separação

A separação dos dados em conjunto de treinamento e teste deve ser feita de maneira aleatória, mantendo a proporção original dos exemplos entre as classes. Por exemplo:

1. Conjunto de dados original (100% dos dados)
    a. 25 colunas e 10.000 linhas
    b. 60% classe A e 40% classe B
2. Conjunto de dados de treino (70% dos dados originais)
    a. 25 colunas e 7.000 linhas
    b. 60% classe A e 40% classe B
3. Conjunto de dados de validação (15% dos dados originais)
    a. 25 colunas e 1.500 linhas
    b. 60% classe A e 40% classe B
4. Conjunto de dados de teste (15% dos dados originais)
    a. 25 colunas e 1.500 linhas
    b. 60% classe A e 40% classe B

▼ 2.2 A proporção da separação

As proporções mais comuns da separação são:

1. 70% treinamento, 15% validação e 15% teste.
2. 80% treinamento, 10% validação e 10% teste.

▼ 3. A sequência prática

Ao coletar um conjunto de dados para treinar, validar e testar um algoritmo de Machine Learning, siga os seguintes passos:

1. Defina a proporção de dados para os subconjuntos de treinamento, validação e teste. Por exemplo, 70% para treinamento, 15% para validação e 15% para teste.
2. Separe a % dos dados de teste logo no início do projeto, antes mesmo de fazer qualquer análise e guarde em algum local.
3. Separe a % dos dados de validação logo no início do projeto, antes mesmo de fazer qualquer análise e guarde em algum local.
4. Com a % dos dados de treinamento, realize o treinamento do algoritmo e valide sua performance usando a % dos dados de validação.
5. Teste inúmeros valores para os principais parâmetros do algoritmo que está sendo treinado. Escolha um valor para o parâmetro, treine o algoritmo, valide sua performance e guarde o resultado. Após testar todos os parâmetros disponíveis, definia os melhores valores.
6. Definido os melhores valores para os parâmetros, junte os dados de treinamento com os dados de validação e treine novamente o algoritmo com a união desses dados, usando os melhores parâmetros.
7. O resultado dos passos de 1 a 5, fornecerá um algoritmo treinado com os melhores valores para os parâmetros.
8. Por último, avalie a capacidade de generalização desse modelo final, utilizando os dados de teste separados no começo do projeto, para estimar o erro de generalização.

9. Se a performance do algoritmo treinado com os dados treinamento e validado com os dados de validação, for muito maior do que a performance sobre os dados de teste, que ele nunca viu, o seu algoritmo pode estar “overfitado”. Por exemplo, acurárcia do algoritmo na validação é de 80% e a acurácia do algoritmo no teste é de 60%.
10. Se a performance do algoritmo treinado com os dados treinamento e validado com os dados de validação, não for muito tão diferente da que a performance sobre os dados de teste, que ele nunca viu, o seu algoritmo está generalizando bem. Por exemplo, acurárcia do algoritmo na validação é de 80% e a acurácia do algoritmo no teste é de 75%.
11. Unir os dados de treinamento, validação e teste, retreinar o algoritmo com os melhores valores para os principais parâmetros e publicar em produção.

▼ 4. Os cuidados

1. Se o conjunto de dados de validação é muito pequeno, a avaliação da performance pode ser imprecisa: você pode acabar não selecionando o melhor modelo.
2. Se o conjunto de dados de validação é muito grande, então o conjunto para treinar o algoritmo será menor do que aquele utilizando para avaliar sua performance. É como se você estivesse treinando o algoritmo para correr 100m rasos e avaliando sua performance em uma maratona de 25 Km.

Uma forma de resolver esse problema é utilizando uma técnica chamada Cross-Validation, onde cada modelo é avaliado com uma parte do conjunto de dados e treinado com o resto. A cada iteração, uma porção dos dados de treinamento se tornam os novos dados para validação enquanto o conjunto usado na validação durante a iteração anterior é incorporado aos dados de treinamento.

Depois da última iteração, também chamado de “fold”, é realizada a média e o desvio-padrão dos valores das performance de cada iteração.

▼ 5. Prós & Contras

▼ 5.1 Benefícios da Estratégia Train-Validation-Test

A estratégia Train-Validation-Test tem diversos benefícios, incluindo:

- Melhorar a precisão do modelo de Machine Learning.
- Ajudar a evitar o overfitting e o underfitting.
- Prevenir o uso de dados de teste para ajustar os parâmetros do modelo.
- Fornecer uma boa estimativa do desempenho do modelo no mundo real.

▼ 5.2 Desvantagens da Estratégia Train-Validation-Test

No entanto, a estratégia Train-Validation-Test também tem algumas desvantagens, incluindo:

- Pode ter custos computacionais elevados.
- O conjunto de dados pode não ser suficientemente grande para treinar e validar.
- O resultado pode não ser generalizável para dados não vistos anteriormente.

▼ 6. Resumo

▼ 7. Próxima aula

O problema do Overfitting