Hiperparâmetros são configurações de um modelo de machine learning que são definidas antes do treinamento começar e **controlam** o processo de aprendizado.

Eles determinam as características e o comportamento do modelo, diferente dos parâmetros (como pesos e vieses), que são aprendidos durante o treinamento.

Teorema do Bias-Variância:

$$\text{Erro Total} = \text{Bias}^2 + \text{Variância} + \text{Erro Irredutível}$$

- Bias (Viés): Erro devido a suposições simplificadoras do modelo
- Variância: Erro devido à sensibilidade a pequenas flutuações no conjunto de treinamento
- Trade-off: Modelos complexos (baixo bias, alta variância) vs Modelos simples (alto bias, baixa variância)

Erro de Generalização:

```
text
```

$$E\_gen = E\_emp + \Omega(\text{model complexity})$$

Onde:

- E_gen: Erro em dados não vistos
- E_emp: Erro empírico (dados de treinamento)
- Ω: Termo de regularização que penaliza complexidade

Exemplos comuns incluem a taxa de aprendizado, o tamanho do lote (batch size) e a arquitetura do modelo (como o número de camadas em uma rede neural).

Características principais

- **Definidos antes do treinamento:** Os hiperparâmetros não são aprendidos dos dados, mas sim configurados manualmente antes que o modelo seja treinado.
- **Controlam o processo:** Eles influenciam como o modelo aprende e o desempenho final. Um ajuste inadequado pode levar a problemas como overfitting (quando o modelo se ajusta demais aos dados de treinamento) ou underfitting.

- **Exemplos comuns:**
  - **Taxa de aprendizado:** Controla o tamanho do "passo" dado para otimizar a função de perda em cada iteração de treinamento.
  -
  - **Tamanho do lote (Batch Size):** O número de exemplos de dados que o modelo processa antes de atualizar seus parâmetros.
  -
  - **Número de épocas (Epochs):** O número de vezes que o modelo percorre todo o conjunto de dados de treinamento.
  -
  - **Arquitetura do modelo:** Em redes neurais, isso pode incluir o número de camadas ocultas e o número de nós em cada camada.

- **Exemplos comuns:**
  - **Número de camadas ocultas e neurônios (Redes Neurais)**: Controlam a complexidade do modelo.
  - **Profundidade máxima da árvore (Árvores de Decisão)**: Limita o número de divisões para evitar o sobreajuste (*overfitting*).
  - **Número de vizinhos (KNN)**: O 'k' no algoritmo K-Vizinhos Mais Próximos.

Importância do ajuste

- **Otimização:** O processo de encontrar os hiperparâmetros ideais é chamado de "otimização de hiperparâmetros" ou "tuning de hiperparâmetros".
- **Equilíbrio:** O objetivo é encontrar um equilíbrio entre o viés e a variância do modelo, garantindo que ele não seja nem muito simples (subajustado) nem muito complexo (superajustado) para os dados.
- **Desempenho:** A escolha correta dos hiperparâmetros é crucial para o sucesso de um modelo de machine learning, pois impacta diretamente sua capacidade de generalizar para dados novos e não vistos.

Ajuste de hiperparâmetros

Encontrar a melhor combinação de hiperparâmetros é um processo chamado de ajuste de hiperparâmetros. Isso é crucial para otimizar o desempenho do modelo. Métodos comuns incluem:

- ***Grid Search***: Testa todas as combinações possíveis dentro de um intervalo de valores pré-definido.
- ***Random Search***: Testa combinações aleatórias, sendo frequentemente mais eficiente que o *grid search*.
- **Otimização Bayesiana**: Usa um modelo probabilístico para guiar a busca, tornando-a mais eficiente.

Ajuste de hiperparâmetros

- O ajuste de hiperparâmetros é a identificação e seleção de variáveis usadas no treinamento de modelos de machine learning, tem como foco obter um melhor desempenho dos modelos em suas tarefas.
- Quando realizado corretamente, o ajuste de hiperparâmetros minimiza a função de perda de um modelo de aprendizado de máquina, o que significa que o desempenho do modelo é treinado para ser o mais preciso possível.

Ajuste de hiperparâmetros

- O ajuste de hiperparâmetros é uma prática experimental em que cada iteração testa diferentes valores de hiperparâmetros até que os melhores sejam identificados. Esse processo é crítico para o desempenho do modelo, pois os hiperparâmetros regem seu processo de aprendizado.
- A quantidade de neurônios em uma rede neural, a taxa de aprendizado de um modelo de IA generativa e o tamanho do kernel de uma máquina de vetores de suporte são exemplos de hiperparâmetros.

Ajuste de hiperparâmetros

- Um bom ajuste de hiperparâmetros significa um desempenho geral mais sólido do modelo de aprendizado de máquina de acordo com as métricas da tarefa pretendida. É por isso que o ajuste de hiperparâmetros também é conhecido como *otimização de hiperparâmetros*.

Métodos de ajuste de hiperparâmetros

- Os cientistas de dados têm uma variedade de métodos de ajuste de hiperparâmetros à sua disposição, cada um com seus respectivos pontos fortes e fracos. O ajuste de hiperparâmetros pode ser realizado manualmente ou automatizado como parte de uma estratégia de AutoML (aprendizado de máquina automatizado).

Pesquisa em grade

A pesquisa em grade é um método abrangente e exaustivo de ajuste de hiperparâmetros. Depois que os cientistas de dados estabelecem todos os valores possíveis para cada hiperparâmetro, uma pesquisa em grade constrói modelos para todas as configurações possíveis desses valores discretos de hiperparâmetros. Esses modelos são avaliados quanto ao desempenho e comparados entre si, com o melhor modelo selecionado para treinamento.

Dessa forma, a pesquisa em grade é semelhante a forçar um PIN de forma bruta, inserindo todas as combinações possíveis de números até que a sequência correta seja descoberta. Embora permita que os cientistas de dados considerem todas as configurações possíveis no espaço de hiperparâmetros, a pesquisa em grade é ineficiente e consome muitos recursos computacionais.

Grid Search:

- Busca exaustiva em grade pré-definida
- Garante encontrar ótimo na grade
- Custo computacional exponencial

Pesquisa aleatória

A pesquisa aleatória difere da pesquisa em grade porque os cientistas de dados fornecem distribuições estatísticas em vez de valores discretos para cada hiperparâmetro. Uma pesquisa aleatória extrai amostras de cada intervalo e constrói modelos para cada combinação. Ao longo de várias iterações, os modelos são ponderados uns contra os outros, até que o melhor modelo seja encontrado.

A pesquisa aleatória é preferível à pesquisa em grade em situações em que o espaço de pesquisa de hiperparâmetros contém grandes distribuições — simplesmente exigiria muito esforço testar cada valor discreto. Os algoritmos de pesquisa aleatória podem retornar resultados comparáveis à pesquisa em grade em consideravelmente menos tempo, embora não seja garantido que descubram a configuração de hiperparâmetros mais ideal.

Random Search (Bergstra & Bengio, 2012):

- Amostragem aleatória do espaço
- Mais eficiente que Grid Search para espaços de alta dimensão
- Teorema: Probabilidade de encontrar região ótima cresce exponencialmente

Otimização bayesiana

A otimização bayesiana é um algoritmo de otimização baseada em modelo sequencial (SMBO), no qual cada iteração de testes melhora o método de amostragem do próximo. Tanto a grade quanto as pesquisas aleatórias podem ser realizadas simultaneamente, mas cada teste é realizado isoladamente — os cientistas de dados não podem usar o que aprenderam para informar os testes subsequentes.

Com base nos testes anteriores, a otimização bayesiana seleciona probabilisticamente um novo conjunto de valores de hiperparâmetros que provavelmente fornecerá melhores resultados. O modelo probabilístico é chamado de *substituto* da função objetiva original. Como os modelos substitutos são eficientes em termos de computação, eles geralmente são atualizados e melhorados cada vez que a função objetiva é executada.

Quanto melhor o substituto for na previsão de hiperparâmetros ideais, mais rápido o processo se tornará, com menos testes de função objetiva necessários. Isso torna a otimização bayesiana muito mais eficiente do que os outros métodos, já que não há perda de tempo em combinações inadequadas de valores de hiperparâmetros.

O processo de determinar estatisticamente a relação entre um resultado (neste caso, o melhor desempenho do modelo) e um conjunto de variáveis é conhecido como *análise de regressão*. Os processos gaussianos são um desses SMBO populares entre cientistas de dados.

Hyperband

Lançado em 2016, o Hyperband foi projetado para melhorar a pesquisa aleatória, truncando o uso de configurações de treinamento que não fornecem resultados sólidos e, ao mesmo tempo, alocando mais recursos para configurações positivas.

Essa "parada antecipada" é obtida por meio da redução sucessiva pela metade, um processo que reduz o conjunto de configurações removendo a metade de pior desempenho após cada rodada de treinamento. Os 50% melhores de cada lote são levados para a próxima iteração, até que reste uma configuração de hiperparâmetros ideal.

GPIA - Grupo de Pesquisa em Informática Aplicada
DIT - Departamento de Informática e Turismo
Me. UGO HENRIQUE PEREIRA DA SILVA

GridSearchCV e RandomizedSearchCV

Conceito: Métodos de busca para encontrar a combinação ótima de hiperparâmetros de um modelo.

Tópico	Conteúdo Principal
Ajuste de Hiperparâmetros	O que são hiperparâmetros? (Ex: `n_estimators` em RF, `C` e `gamma` em SVM). Como o ajuste afeta o *trade-off* Vies-Variância.
GridSearchCV	**Busca Exaustiva:** Testa *todas* as combinações possíveis de hiperparâmetros em uma grade definida. **Vantagem:** Garante que a melhor combinação da grade seja encontrada. **Desvantagem:** Alto custo computacional ($\mathcal{O}(n_1 \times n_2 \times \dots \times n_k)$).
RandomizedSearchCV	**Busca Aleatória:** Testa combinações aleatórias da grade. **Vantagem:** Mais eficiente computacionalmente, especialmente para grandes espaços de busca. **Desvantagem:** Não garante a melhor combinação, mas geralmente chega perto rapidamente.
Validação Cruzada (`cv`)	Essencial em ambos os métodos para garantir a robustez da avaliação (Ex: k-Fold Cross-Validation).

GPIA - Grupo de Pesquisa em Informática Aplicada
DIT - Departamento de Informática e Turismo
Me. UGO HENRIQUE PEREIRA DA SILVA

INSTITUTO FEDERAL
São Paulo
Campus
São Paulo

Regularização L1/L2 e Impacto nos Coeficientes

Conceito: Adição de um termo de penalidade à função de custo (ou *loss function*) do modelo para desencorajar coeficientes grandes e, assim, simplificar o modelo e reduzir o *overfitting*.

Tópico	Conteúdo Principal
Overfitting & Função de Custo	Revisão de *overfitting* (alta variância). Introdução da Função de Custo (Ex: MSE) e o objetivo de minimizá-la.
Regularização L2 (Ridge)	**Penalidade:** Soma do quadrado dos coeficientes ($\sum \beta_i^2$). **Impacto:** Reduz a magnitude dos coeficientes em direção a zero, mas **não os zera completamente**. Útil quando *todas* as *features* são relevantes.
Regularização L1 (Lasso)	**Penalidade:** Soma do valor absoluto dos coeficientes ($\sum
Elastic Net	Combinação de L1 e L2. Permite os benefícios de ambos.
Hiperparâmetro α (ou λ)	Controla a força da regularização. $\alpha=0$ (sem regularização), $\alpha \rightarrow \infty$ (coeficientes $\rightarrow 0$).

GPIA - Grupo de Pesquisa em Informática Aplicada
DIT - Departamento de Informática e Turismo
Me. UGO HENRIQUE PEREIRA DA SILVA

INSTITUTO FEDERAL
São Paulo
Campus São Paulo

Pipeline e Evitar Vazamento de Dados (*Data Leakage*)

Conceito: O `Pipeline` do Scikit-learn encadeia etapas de processamento (transformações) e o estimador final (o modelo) em um único objeto.

Tópico	Conteúdo Principal
O Problema do Vazamento de Dados	O que é? Ocorre quando informações do conjunto de teste "vazam" para o treinamento. Ex: Ajustar um `StandardScaler` ou fazer uma seleção de *features* usando todo o conjunto de dados (*train* + *test*) antes da divisão e validação cruzada.
Solução: O Objeto `Pipeline`	O `Pipeline` garante que cada etapa de pré-processamento (como escalonamento, codificação, imputação) só use dados do *fold* de treinamento atual durante a Validação Cruzada.
Sintaxe e Implementação	Como construir um *pipeline* (sequência de tuplas: `('nome_da_etapa', objeto_transformador)`).
Integração com GridSearchCV	Como o `Pipeline` se torna o estimador (modelo) que o `GridSearchCV` ajusta. Os hiperparâmetros são nomeados com o prefixo da etapa (Ex: `'modelo__alpha'`).

oções de AutoML (Visão Geral)

Conceito: O *Automated Machine Learning* visa automatizar as tarefas repetitivas e demoradas de um *workflow* de ML, como engenharia de *features*, seleção de modelos e ajuste de hiperparâmetros.

Tópico	Conteúdo Principal
Motivação	Por que automatizar? Acelerar o desenvolvimento, reduzir a necessidade de *expertise* profunda em todas as áreas, democratizar o ML.
Componentes Chave	**Auto Feature Engineering**, **Auto Model Selection** (seleção do algoritmo mais adequado) e **Auto Hyperparameter Tuning** (usando buscas mais avançadas como a Bayesiana, em vez de Grid/Random).
Bibliotecas/Ferramentas	Menção a ferramentas populares (Ex: Google Vertex AI, Azure Machine Learning, H2O.ai, Auto-Sklearn, TPOT).
Limitações	Custo computacional, dificuldade em explicar o modelo final (*black-box*), ainda requer *expertise* humana para definir o problema e interpretar os resultados.

Noções de AutoML

AutoML: Automatização completa do processo de machine learning, incluindo:

- Engenharia de features
- Seleção de modelo
- Ajuste de hiperparâmetros
- Avaliação de modelo

Ferramenta	Linguagem	Destaques
Auto-sklearn	Python	Baseada em scikit-learn
TPOT	Python	Usa algoritmos genéticos
H2O AutoML	Multi	Interface gráfica e CLI
Google AutoML / Azure AutoML	Cloud	Escalável e fácil de integrar

Busca em Espaço Combinatório:

O espaço AutoML inclui:

- Algoritmos $A \in \{A_1, A_2, ..., A_□\}$
- Pré-processadores $P \in \{P_1, P_2, ..., P_□\}$
- Hiperparâmetros $\lambda \in \Lambda$

O problema AutoML:

$$(A^*, P^*, \lambda^*) = \text{argmin } E[L(y, f_{A,P,\lambda}(x))]$$

- A*: Melhor algoritmo de machine learning
- P*: Melhor pipeline de pré-processamento
- λ*: Melhores hiperparâmetros
- E[]: Valor esperado (esperança matemática)
- L(): Função de perda (loss function)
- y: Variável target (resposta verdadeira)
- f_{A,P,λ}(x): Predição do modelo
- x: Variáveis de entrada (features)

A equação `(A*, P*, λ*) = argmin E[L(y, f_{A,P,λ}(x))]` significa:

"Encontre a tripla ideal (algoritmo, pré-processamento, hiperparâmetros) que minimiza o erro esperado do modelo ao prever a variável target a partir das features de entrada, considerando a incerteza através de múltiplas validações."

"Teste sistematicamente todas as combinações possíveis de modelos, transformações de dados e configurações, e escolha aquela que produz os melhores resultados de forma mais consistente em dados não vistos."

Esta formulação captura a essência do AutoML: a automação completa do processo de seleção e otimização de modelos de machine learning.

```
from tpot import TPOTRegressor

tpot = TPOTRegressor(generations=5, population_size=20, verbosity=2)

tpot.fit(X_train, y_train)

print(tpot.score(X_test, y_test)
```

```
import autosklearn.classification

# AutoML com auto-sklearn

automl = autosklearn.classification.AutoSklearnClassifier(

    time_left_for_this_task=300,  # 5 minutos

    per_run_time_limit=60,

    n_jobs=-1

)

automl.fit(X_train, y_train)

print(automl.show_models())
```

```
import h2o

from h2o.automl import H2OAutoML

h2o.init()

h2o_df = h2o.H2OFrame(pd.concat([X, y], axis=1))


aml = H2OAutoML(max_runtime_secs=300)

aml.train(y='target', training_frame=h2o_df)


lb = aml.leaderboard

print(lb.head())
```

Boas Práticas e Considerações Finais

1. Sempre use Pipeline para evitar data leakage
2. Comece com RandomizedSearchCV para explorar o espaço de parâmetros
3. Use GridSearchCV para refinar os melhores parâmetros
4. Regularização L1 para seleção de features, L2 para coeficientes pequenos
5. Validação cruzada estratificada para dados desbalanceados
6. AutoML para problemas complexos ou quando há pouco tempo
7. Sempre valide no conjunto de teste separado

Atividade Prática:

1. **Conjunto de Dados:** Utilize um conjunto de dados real (Ex: Boston Housing, Titanic, ou um dataset de classificação/regressão do Scikit-learn).
2. **Modelo Base:** Escolha um modelo com hiperparâmetros importantes (Ex: Random Forest, SVM, Regressão Logística/Ridge).
3. **Etapas:**
   - **Pré-Processamento Básico:** (Ex: `StandardScaler` ou `OneHotEncoder`).
   - **Criar o `Pipeline`:** Encadear o pré-processamento e o modelo.
   - **Definir a Grade de Busca:** Criar um dicionário de hiperparâmetros para o `GridSearchCV` ou `RandomizedSearchCV`.
   - **Executar o Ajuste:** Treinar o `GridSearchCV`/`RandomizedSearchCV` usando o *pipeline* e a validação cruzada.
   - **Avaliação:** Analisar o `best_params_` e o `best_score_`.
   - **Modelo Final:** Avaliar a *performance* do melhor modelo no conjunto de teste (separado no início).