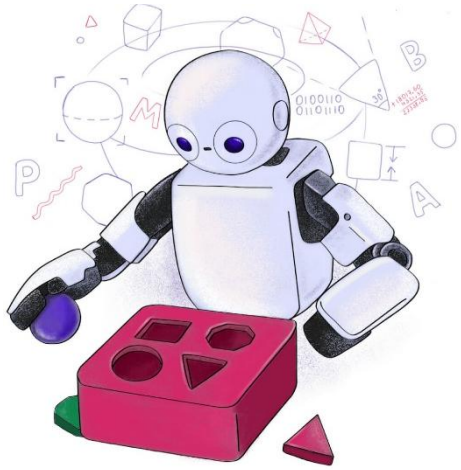
TP558 - Tópicos avançados em Machine Learning:

Auto-Pytorch: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL



Inatel

Anderson Reis Rufino Marins
anderson.rufino@dtel.Inatel.br

Introdução

- Nos últimos anos, o aprendizado de máquina (ML) conquistou enorme popularidade, resultado de uma década de avanços que permitiram atingir desempenhos elevados em muitas tarefas.
- . Ainda assim, projetar soluções de ML manualmente continua sendo um processo trabalhoso, que atrasa a aplicação em novos domínios.
- Para enfrentar esse desafio, surgiu o campo do **aprendizado de máquina automatizado (AutoML)**, cujo objetivo é auxiliar pesquisadores e desenvolvedores na criação de pipelines de ML de alto desempenho sem a necessidade de intervenção manual.

Introdução

- O **AutoPyTorch** é uma biblioteca de *AutoML* (Automated Machine Learning) em Python voltada para **redes neurais e aprendizado profundo**. Ele automatiza várias etapas do processo de construção de modelos de *deep learning* e *machine learning*, como:
- Pré-processamento de dados
- Seleção de arquitetura de rede neural
- Escolha de hiperparâmetros
- Treinamento e validação
- Avaliação de desempenho
- Ou seja, o AutoPyTorch busca, de forma **automática**, a melhor combinação de **pipeline de ML + arquitetura de rede + hiperparâmetros**.

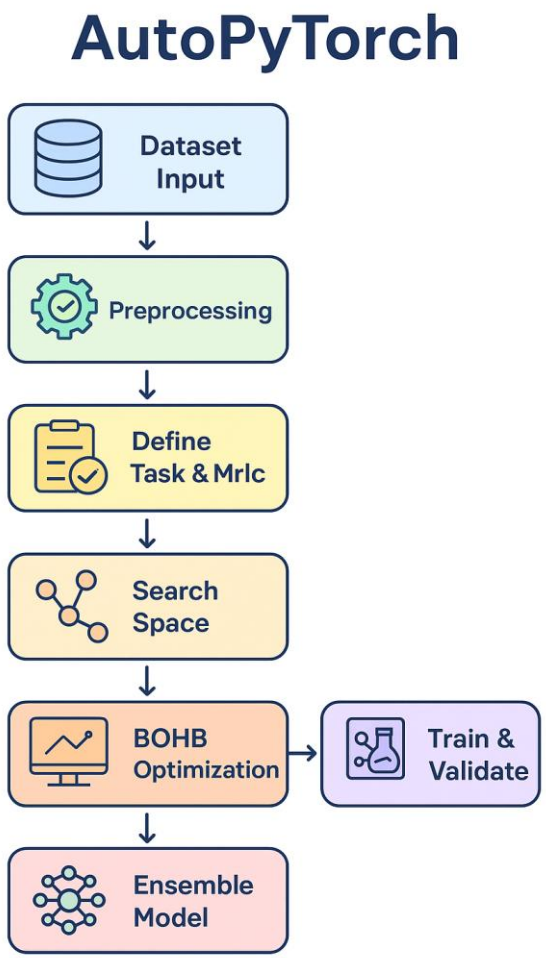
Fundamentação teórica

- Nos primeiros anos do AutoML, as ferramentas se concentraram em **pipelines tradicionais de aprendizado de máquina**, já que os dados tabulares eram — e continuam sendo — muito relevantes em aplicações práticas.
- Exemplos incluem *Auto-WEKA*, *hyperopt-sklearn*, *auto-sklearn* e *TPOT*. Esses sistemas exploravam abordagens como **otimização Bayesiana**, **estratégias evolucionárias** e **aprendizado por reforço** para resolver não apenas a **otimização de hiperparâmetros**, mas também o problema combinado de **seleção de algoritmo e configuração (CASH)**.

Fundamentação teórica

- O **AutoNet**, baseado em *Lasagne* e *Theano*, foi uma das primeiras abordagens a otimizar arquitetura e hiperparâmetros **conjuntamente** em grande escala, o que resultou em sucesso no primeiro desafio de AutoML.
- . O presente trabalho dá continuidade a essa linha, mas de forma mais sistemática e poderosa, apresentando o **Auto-PyTorch**, que combina **otimização multifidelidade**, **ensembles**, **meta-aprendizado para warm-start** e um **espaço de configuração eficiente**.

Arquitetura e funcionamento



Treinamento e otimização

1. Entrada dos dados

- Você fornece o dataset (tabular, imagens ou séries temporais).

Treinamento e otimização

2. Pré-processamento

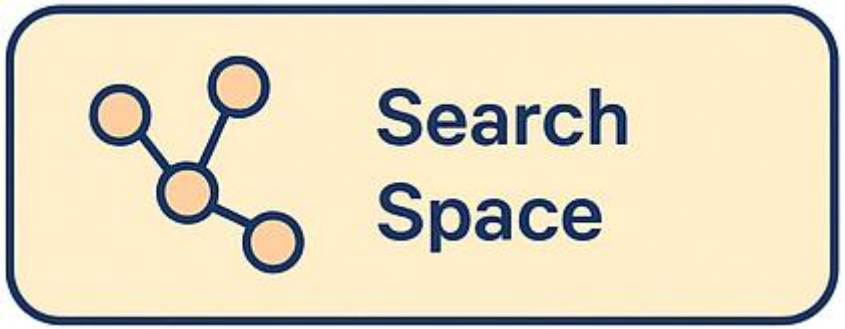
- O AutoPyTorch faz o pré-processamento automaticamente (normalização, codificação de categorias, divisão em treino/teste, etc.).

Treinamento e otimização

3. Configuração da busca

- Você define:
  - O tipo de tarefa (classificação, regressão, multiclasse).
  - Tempo máximo para busca (por ex., 1 hora).
  - Métrica de avaliação (accuracy, RMSE, F1-score, etc.).

Treinamento e otimização



4. Espaço de busca

- Define o espaço de possíveis **pipelines**:
  - Diferentes arquiteturas de rede (MLP, CNN, etc.).
  - Diferentes otimizadores (Adam, SGD...).
  - Diferentes técnicas de regularização (Dropout, BatchNorm...).

Treinamento e otimização

5. Otimização Bayesiana

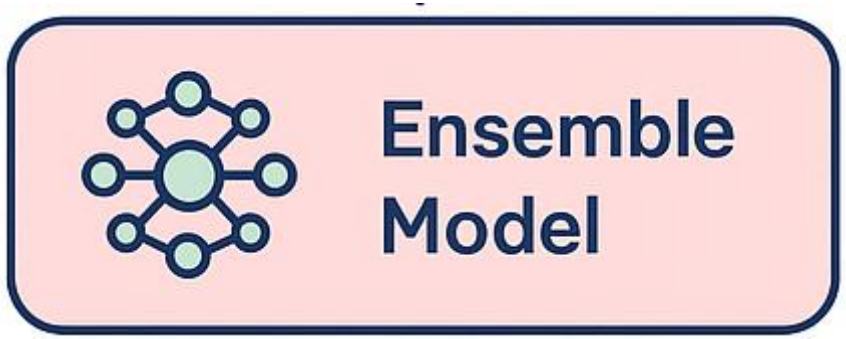
- Utiliza **BOHB (Bayesian Optimization + HyperBand)**:
  - **HyperBand** → avalia modelos rapidamente, descartando os ruins.
  - **Bayesian Optimization** → escolhe de forma inteligente novos hiperparâmetros a testar.

Treinamento e otimização

6. Treinamento e Validação

- Os modelos mais promissores são treinados em maior profundidade.
- Os ruins são descartados cedo (early stopping).

Treinamento e otimização



7. Construção de Ensemble

- No final, o AutoPyTorch monta um **ensemble** dos melhores modelos encontrados para aumentar a robustez e melhorar a performance.

Treinamento e otimização

8. Resultado final

- Retorna o **melhor modelo** ou um **ensemble de modelos**, pronto para ser usado em previsão.

⚖️ **Resumo Comparativo**

Vantagens e desvantagens

Aspecto	Vantagem	Desvantagem
Facilidade de uso	Automatiza grande parte do pipeline	Exige conhecimento inicial de AutoML
Desempenho	Busca arquiteturas e hiperparâmetros ótimos	Pode demandar alto tempo de execução
Flexibilidade	Integra PyTorch e Auto-Sklearn	Difícil incluir arquiteturas personalizadas
Resultados	Ensemble melhora a precisão	Depuração complexa
Aplicações	Ideal para dados tabulares	Suporte limitado a imagens/NLP

Exemplo(s) de aplicação

Resultados: {'accuracy': 0.9777777777777777}			
Modelos:	Preprocessing	Estimator	Weight
---:	:---	:---	---:
0	None	CBLearner	0,24
1	None	ETLearner	0,2
2	None	RFLearner	0,18
3	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,NoScaler,FastICA	no embedding,ResNetBackbone,FullyConnectedHead,nn.Sequential	0,14
4	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,RobustScaler,FeatureAgglomeration	no embedding,ShapedResNetBackbone,FullyConnectedHead,nn.Sequential	0,1
5	None	SVMLearner	0,1
6	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,StandardScaler,RandomTreesEmbedding	no embedding,ShapedMLPBackbone,FullyConnectedHead,nn.Sequential	0,04

Exemplo(s) de aplicação

	Wine - Resultados: {'accuracy': 1.0}		
	Preprocessing	Estimator	Weight
---:	:----------	:----------	---------:
0	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,StandardScaler,NoFeaturePreprocessing	no embedding,ShapedMLPBackbone,FullyConnectedHead,nn.Sequential	0,14
1	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,MinMaxScaler,FeatureAgglomeration	no embedding,MLPBackbone,FullyConnectedHead,nn.Sequential	0,1
2	None	ETLearner	0,1
3	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,RobustScaler,KitchenSink	no embedding,MLPBackbone,FullyConnectedHead,nn.Sequential	0,08
4	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,NoScaler,Nystroem	no embedding,ShapedMLPBackbone,FullyConnectedHead,nn.Sequential	0,08
5	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,NoScaler,Nystroem	no embedding,ShapedMLPBackbone,FullyConnectedHead,nn.Sequential	0,08
6	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,NoScaler,SPC	no embedding,ShapedMLPBackbone,FullyConnectedHead,nn.Sequential	0,06
7	None	RFLearner	0,06
8	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,QuantileTransformer,RandomTreesEmbedding	no embedding,ShapedMLPBackbone,FullyConnectedHead,nn.Sequential	0,06
9	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,RobustScaler,KitchenSink	no embedding,ShapedResNetBackbone,FullyConnectedHead,nn.Sequential	0,04
10	None	LGBMLearner	0,04
11	None	KNNLearner	0,04
12	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,StandardScaler,NoFeaturePreprocessing	no embedding,ShapedMLPBackbone,FullyConnectedHead,nn.Sequential	0,04
13	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,MinMaxScaler,FeatureAgglomeration	no embedding,ShapedMLPBackbone,FullyConnectedHead,nn.Sequential	0,02
14	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,QuantileTransformer,RandomTreesEmbedding	no embedding,ShapedResNetBackbone,FullyConnectedHead,nn.Sequential	0,02
15	None	CBLearner	0,02
16	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,StandardScaler,NoFeaturePreprocessing	no embedding,ShapedMLPBackbone,FullyConnectedHead,nn.Sequential	0,02

Exemplo(s) de aplicação

	Breast Cancer - Resultados: {'accuracy': 0.9736842105263158}		
	Preprocessing	Estimator	Weight
---:	:----------	:----------	---------:
0	None	ETLearner	0,12
1	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,StandardScaler,NoFeaturePreprocessing	no embedding,ShapedMLPBackbone,FullyConnectedHead,nn.Sequential	0,12
2	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,StandardScaler,NoFeaturePreprocessing	no embedding,ShapedMLPBackbone,FullyConnectedHead,nn.Sequential	0,1
3	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,RobustScaler,FeatureAgglomeration	no embedding,ShapedResNetBackbone,FullyConnectedHead,nn.Sequential	0,08
4	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,MinMaxScaler,SRC	no embedding,MLPBackbone,FullyConnectedHead,nn.Sequential	0,08
5	None	CBLearner	0,08
6	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,StandardScaler,PCA	no embedding,MLPBackbone,FullyConnectedHead,nn.Sequential	0,06
7	None	LGBMLearner	0,06
8	None	RFLearner	0,06
9	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,StandardScaler,NoFeaturePreprocessing	no embedding,ShapedMLPBackbone,FullyConnectedHead,nn.Sequential	0,06
10	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,QuantileTransformer,KitchenSink	no embedding,MLPBackbone,FullyConnectedHead,nn.Sequential	0,04
11	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,NoScaler,FastICA	no embedding,ResNetBackbone,FullyConnectedHead,nn.Sequential	0,04
12	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,PowerTransformer,FastICA	no embedding,MLPBackbone,FullyConnectedHead,nn.Sequential	0,04
13	None	KNNLearner	0,04
14	SimpleImputer,Variance Threshold,NoCoalescer,NoEncoder,QuantileTransformer,KitchenSink	no embedding,ShapedResNetBackbone,FullyConnectedHead,nn.Sequential	0,02

Exemplo(s) de aplicação

California Housing - Resultados: {'r2': 0.8484231464130146}			
California Housing - Modelos:	Preprocessing	Estimator	Weight
---:	:----------	:----------	---------:
0	None	CBLearner	0,64
1	None	LGBMLearner	0,36

Exemplo(s) de aplicação

- 🏦 **1. Finanças e Bancos**
- **Exemplos:**
- **Previsão de inadimplência (credit scoring):** prever se um cliente vai ou não pagar um empréstimo.
- **Detecção de fraudes em transações bancárias:** classificação binária com muitos atributos categóricos e numéricos.
- **Previsão de demanda de caixa ou fluxo financeiro.**
- **Por que usar AutoPyTorch?**
- 👉 Ele busca automaticamente o modelo e os hiperparâmetros ideais para dados tabulares financeiros (normalmente complexos e não lineares).

Exemplo(s) de aplicação

- 🩺 **2. Saúde e Biomedicina**
- **Exemplos:**
- **Classificação de pacientes com base em sintomas ou exames.**
- **Previsão de doenças ou riscos clínicos (por exemplo, diabetes, hipertensão).**
- **Análise de biomarcadores e dados laboratoriais.**
- **Por que usar AutoPyTorch?**
- 👉 Permite combinar modelos clássicos (como árvores) e redes neurais para encontrar automaticamente o melhor pipeline — ideal quando o domínio médico gera muitos parâmetros e variáveis correlacionadas.

Exemplo(s) de aplicação

- 🏭 **3. Indústria e Manufatura**
- **Exemplos:**
- **Manutenção preditiva:** prever falhas em equipamentos a partir de sensores.
- **Otimização de processos industriais.**
- **Controle de qualidade automatizado (classificação de peças defeituosas).**
- **Por que usar AutoPyTorch?**
- 👉 É útil quando há grandes volumes de dados de sensores e é necessário ajustar muitos hiperparâmetros de modelos preditivos.

Exemplo(s) de aplicação

- 🛒 **4. Varejo e Marketing**
- **Exemplos:**
- **Previsão de vendas.**
- **Recomendação de produtos (com dados tabulares de usuários e produtos).**
- **Análise de churn (clientes que vão cancelar um serviço).**
- **Por que usar AutoPyTorch?**
- 👉 Automatiza o ajuste fino de modelos de previsão de demanda e comportamento, sem exigir conhecimento avançado de tuning manual.

Exemplo(s) de aplicação

- ⚙️ **5. Energia e Engenharia**
- **Exemplos:**
- **Previsão de consumo de energia elétrica.**
- **Modelagem de geração solar/eólica.**
- **Otimização de parâmetros em sistemas de controle.**
- **Por que usar AutoPyTorch?**
- 👉 Excelente para modelar séries temporais tabulares ou sistemas com dados contínuos e categóricos mistos.

Exemplo(s) de aplicação

- 🧠 **6. Pesquisa e Prototipagem Científica**
- **Exemplos:**
- Pesquisadores que querem testar rapidamente **diversas configurações de rede neural**.
- **Prototipagem de modelos** sem precisar escrever manualmente todos os testes de hiperparâmetros.
- **Por que usar AutoPyTorch?**
- 👉 Acelera o ciclo de experimentação científica, deixando o pesquisador focar em análise dos resultados em vez de tuning manual.

Exemplo(s) de aplicação

- 🚙 **7. Transporte e Logística**
- **Exemplos:**
- **Previsão de tempo de entrega e rotas ótimas.**
- **Manutenção preditiva de frotas.**
- **Classificação de riscos em viagens ou trajetos.**
- **Por que usar AutoPyTorch?**
- 👉 Automatiza a criação de modelos preditivos baseados em múltiplos parâmetros (distância, tempo, clima, tráfego etc.).

Exemplo(s) de aplicação

- 📜 **8. Governo e Setor Público**
- **Exemplos:**
- **Análise de dados socioeconômicos.**
- **Detecção de irregularidades em licitações.**
- **Previsão de arrecadação tributária.**
- **Por que usar AutoPyTorch?**
- 👉 Permite construir modelos analíticos de alta qualidade com baixo esforço técnico em equipes que não são especialistas em ML.

Comparação com outros algoritmos

Critério	AutoPyTorch	Auto-WEKA	hyperopt-sklearn	auto-sklearn	TPOT
Linguagem base	Python (PyTorch)	Java (WEKA)	Python (scikit-learn + Hyperopt)	Python (scikit-learn)	Python (scikit-learn + DEAP)
Bibliotecas principais	PyTorch, Sklearn, Optuna, ConfigSpace	WEKA, SMAC	scikit-learn, Hyperopt	scikit-learn, SMAC3	scikit-learn, DEAP (Evolução genética)
Tipo de tarefa	Classificação, Regressão, Time Series (em versões recentes)	Classificação, Regressão	Classificação, Regressão	Classificação, Regressão	Classificação, Regressão
Abordagem de otimização	BOHB (Bayesian Optimization + Hyperband)	SMAC (Bayesian Optimization)	TPE (Tree-structured Parzen Estimator)	SMAC (Bayesian Optimization)	Algoritmos genéticos (evolutivos)
Suporte a pipelines personalizados	✅ Sim, com preprocessamento e tuning integrado	✅ Limitado (WEKA filters)	✅ Sim	✅ Sim	✅ Sim (geneticamente evoluídos)

Comparação com outros algoritmos

Critério	AutoPyTorch	Auto-WEKA	hyperopt-sklearn	auto-sklearn	TPOT
Integração com frameworks modernos	Alta (PyTorch, Numpy, Optuna)	Baixa (Java-based)	Média (scikit-learn)	Alta (scikit-learn)	Alta (scikit-learn)
Desempenho em dados grandes	Alto (GPU opcional, paralelismo)	Médio	Médio	Médio	Baixo-Médio
Curva de aprendizado / Facilidade de uso	Moderada (configurações complexas)	Difícil (interface WEKA tradicional)	Fácil (simples em Python)	Fácil	Muito fácil (pipeline intuitivo)
Reprodutibilidade	Alta (controla seeds e configurações)	Alta	Média	Alta	Média

Comparação com outros algoritmos

Critério	AutoPyTorch	Auto-WEKA	hyperopt-sklearn	auto-sklearn	TPOT
Comunidade / Suporte	Média (ativa em pesquisa AutoML)	Baixa (antigo)	Média	Alta	Alta
Vantagem principal	Combina AutoML + Deep Learning com eficiência	Integração direta no WEKA	Simples e leve	Robusto, vencedor de benchmarks	Interpretação fácil via pipelines evolutivos
Desvantagem principal	Complexidade maior e dependência do PyTorch	Interface limitada e linguagem Java	Poucos modelos e suporte menor	Sem suporte a redes neurais	Lento em datasets grandes

Comparação com outros algoritmos

- 💡 **Resumo Geral**
- **AutoPyTorch** → Melhor opção para **tarefas com redes neurais** e datasets grandes (usa GPU e PyTorch).
- **Auto-WEKA** → Voltado a usuários WEKA (Java), pouco usado em projetos modernos.
- **hyperopt-sklearn** → Simples e leve, ideal para **protótipos rápidos**.
- **auto-sklearn** → Forte e estável, ótimo para **modelos clássicos supervisionados**.
- **TPOT** → Foco em **interpretação e pipelines evolutivos**, ideal para iniciantes em AutoML.

Perguntas?

Referências

- [1] L. Zimmer, M. Lindauer and F. Hutter, "Auto-Pytorch: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 9, pp. 3079-3090, 1 Sept. 2021, doi: 10.1109/TPAMI.2021.3067763.
- [2] https://www.automl.org/automl-for-x/tabular-data/autoweka
- [3] https://hyperopt.github.io/hyperopt-sklearn
- [4] https://automl.github.io/auto-sklearn/master
- [5] https://github.com/telekom-security/tpotce

Obrigado!

Quis:
https://docs.google.com/forms/d/e/1FAIpQLSf1ClZTZNyTFw5c6nVhG3A00XVJeNWGcITg6DGmKdlFShzRDA/viewform?usp=header