

APPLICATIONDRIVENLEARNING.JL: UM PACOTE JULIA PARA TREINAMENTO DE MODELOS PREDITIVOS NO CONTEXTO DE TOMADA DE DECISÃO

Giovanni Amorim

PUC-Rio

Rua Marquês São Vicente 225, Rio de Janeiro, RJ

giovanni.amorim@elgroup.com.br

Alexandre Street

PUC-Rio

Rua Marquês São Vicente 225, Rio de Janeiro, RJ

street.@ele.puc-rio.br

Joaquim Dias Garcia

PSR

Pr. De Botafogo 370, Rio de Janeiro, RJ

kimmldg@gmail.com

RESUMO

O framework *Application-Driven Learning* propõe uma abordagem que integra o treinamento de modelos preditivos com processos de tomada de decisão, otimizando os modelos especificamente para o contexto da aplicação. Neste artigo, apresentamos o *ApplicationDrivenLearning.jl*, uma biblioteca de código aberto que, a partir dos ecossistemas de otimização e aprendizado de máquina da linguagem Julia, permite a representação flexível e treinamento eficiente desses modelos. A interface permite ao usuário modelar de forma precisa os processos de tomada de decisão, com restrições e funções objetivo de diversas naturezas, bem como utilizar modelos preditivos de tamanhos e estruturas variadas. O treinamento pode ser realizado a partir de uma formulação exata de otimização binível ou uma abordagem heurística utilizando métodos de otimização com Nelder-Mead ou gradiente descendente, uma contribuição secundária deste trabalho. O pacote reduz drasticamente as barreiras de acesso a essa nova tecnologia combinando facilidade de uso e métodos de solução de alta performance.

PALAVRAS CHAVE. Aprendizado Guiado pela Aplicação. Otimização Binível. Modelagem Preditiva

Tópicos: MOI-Métodos de Otimização sob Incerteza. EN&OG-PO na Área de Energia, Óleo e Gás.

ABSTRACT

The *Application-Driven Learning* framework proposes an approach that integrates the training of predictive models with decision-making processes, optimizing the models specifically for the context it will be applied. In this paper, we present *ApplicationDrivenLearning.jl*, an open-source library that leverages Julia's optimization and machine learning ecosystems to enable flexible representation and efficient training of such models. The interface allows users to accurately model decision-making processes, including a wide range of constraints and objective functions, as well as to incorporate predictive models of varying sizes and structures. Training can be performed using either an exact bilevel optimization formulation or a heuristic approach based on optimization methods such as Nelder-Mead or gradient descent—a secondary contribution of this work. The package significantly lowers the barriers to adopting this new technology by combining ease of use with high-performance solution methods.

KEYWORDS. Application Driven Learning. Bilevel Optimization. Predictive Models.

Paper topics: MOI-Optimization Under Uncertainty Methods. EN&OG-OP on Energy, Oil and Gas.

1. Introdução

A tomada de decisão sob incerteza é um tema central em diversas aplicações no campo da pesquisa operacional. Tradicionalmente, a separação entre previsão e decisão tem guiado o desenvolvimento de modelos operacionais neste contexto: primeiro, estimativas pontuais são geradas com base em dados históricos; em seguida, decisões são planejadas com base nessas previsões; e, por fim, ajustes operacionais são feitos conforme as incertezas se realizam. Esse paradigma, chamado também de “malha aberta”, é amplamente utilizado, mas sofre a limitação do desalinhamento entre acurácia preditiva e custo de decisão [Dias Garcia et al., 2021].

Neste trabalho, apresentamos o pacote **ApplicationDrivenLearning.jl**, uma biblioteca de código aberto desenvolvida na linguagem Julia que viabiliza o treinamento de modelos preditivos integrados a problemas de decisão. A biblioteca foi concebida com foco em flexibilidade e desempenho, aproveitando os ricos ecossistemas de otimização (JuMP.jl) e aprendizado de máquina (Flux.jl) da linguagem Julia. Sua estrutura permite representar diferentes etapas do processo decisório — previsão, planejamento e avaliação — e aplicar métodos exatos e heurísticos implementados para o treinamento dos modelos preditivos. Também propomos um novo método de treinamento baseado em gradientes, voltado ao treinamento eficiente de modelos em larga escala, como redes neurais profundas.

Nosso objetivo com esta implementação é fornecer uma ferramenta prática para pesquisadores e operadores que desejam desenvolver modelos preditivos mais aderentes à realidade das decisões operacionais. Este artigo detalha a estrutura do pacote, discute seus métodos de treinamento e apresenta resultados experimentais em problemas teóricos e práticos. Neste artigo aplicamos o método ao problema de despacho econômico, que pode ser executado pelo operador a cada hora ou 15 minutos para programar a operação ótima das centrais de geração com alta acurácia e segurança.

O pacote já está disponível online em:
github.com/LAMPSPUC/ApplicationDrivenLearning.jl

2. Aprendizado Orientado pela Aplicação

Um processo genérico de tomada de decisão pode ser descrito em três etapas: **previsão**, **planejamento** e **avaliação**. A etapa de previsão consiste na aplicação de um modelo preditivo com parâmetros e informações estruturadas x_t com o objetivo de gerar uma estimativa y_t do cenário futuro $\Psi(x_t, \theta)$.

Na etapa de planejamento, uma decisão z_t^* é calculada com base no cenário previsto $\hat{y}_t$ por meio da resolução de um problema de minimização do custo de planejamento G_p , considerando a região viável Z_p .

Por fim, a etapa de avaliação consiste na verificação da decisão planejada z_t^* frente aos valores realizados y_t por meio de uma função de custo de avaliação G_a , permitindo ajustes limitados sobre a decisão planejada z_t^* , restritos por uma nova região viável possivelmente condicionada à decisão planejada $Z_a(z_t^*, y_t)$.

Nosso objetivo é resolver o problema unificado de encontrar os parâmetros do modelo preditivo $\theta \in \Theta$ que minimizem o custo de avaliação ao longo de um conjunto de dados $x_1, y_1, \dots, x_T, y_T$. Este problema unificado pode ser representado por um problema de otimização binível, em que o problema de nível inferior reflete a etapa de planejamento das decisões por etapa e o problema de nível superior engloba as etapas de avaliação das decisões e definição dos parâmetros do modelo preditivo:

$$\begin{aligned} \min_{\theta \in \Theta, z_1^*, \dots, z_T^*} & \sum_{t=1}^T G_a(z_t^*, y_t) \\ \text{s. t.: } & \hat{y}_t = \Psi(x_t, \theta) \quad \forall t \in 1, \dots, T \\ & z_t^* \in \arg \min_{z \in Z_p} G_p(z, \hat{y}_t) \quad \forall t \in 1, \dots, T \end{aligned}$$

Diversas estratégias foram propostas na literatura para solucionar problemas de otimização binível como esse. Por mais que soluções exatas possam ser obtidas por formulações de programação inteira com uso das condições de KKT, os métodos podem ser tornar computacionalmente inviáveis quando aplicados a problemas de planejamento com muitas variáveis, longos horizontes temporais, não linearidade na formulação da tomada de decisão ou modelos preditivos complexos.

Para lidar com os desafios de escalabilidade, um método heurístico foi proposto [Garcia et al., 2024] para resolver iterativamente o problema do treinamento, permitindo atualizações eficientes dos parâmetros por meio de otimização numérica dentro do arcabouço de aprendizado orientado pela aplicação. Esse método parte da inicialização dos parâmetros do modelo de previsão por alguma regra, que pode ter natureza aleatória ou considerar alguma dinâmica conhecida do modelo e do problema. Em seguida, a etapa de atualização altera os valores dos parâmetros executando os passos de planejamento e avaliação, utilizando os dados disponíveis e os valores de custo resultantes, e finalmente verifica-se um critério de convergência.

Essa descrição é propositalmente genérica, pois diversos métodos de otimização numérica já consolidados podem ser utilizados para conduzir essa busca. O trabalho em [Garcia et al., 2024] concentra-se no uso do algoritmo de Nelder-Mead, um método de otimização local, para implementar esse procedimento, demonstrando sua aplicabilidade em problemas nos quais a abordagem binível exata não é viável. No entanto, ainda há espaço para aprimoramentos, como a utilização da estrutura específica do problema para orientar de forma mais eficiente a etapa de atualização.

3. Método Baseado em Gradiente

Este trabalho propõe uma versão do método heurístico descrito acima com base em métodos de gradiente, aproveitando métodos conhecidos para extrair informações de gradientes a partir da estrutura do modelo e aplicando a regra da cadeia para atualizar os parâmetros do modelo preditivo de forma adequada.

Nesse método de treinamento por gradiente, a etapa de atualização funciona modificando os parâmetros com base no gradiente do custo calculado para cada instante de tempo, $\partial C_t / \partial \theta$, seguindo uma lógica de gradiente descendente. Esse gradiente pode ser obtido aplicando a regra da cadeia:

$$\frac{\partial C_t}{\partial \theta} = \frac{\partial C_t}{\partial z_t^*} \times \frac{\partial z_t^*}{\partial \hat{y}_t} \times \frac{\partial \hat{y}_t}{\partial \theta}$$

A formulação resulta em três termos principais: 1) O gradiente do custo de avaliação em relação à decisão planejada, que pode ser extraído diretamente da solução dual do problema de avaliação, uma vez que esta entra como lado direito de restrições de igualdade no modelo de avaliação; 2) gradiente da decisão de planejamento em relação à saída do modelo preditivo, que pode ser calculado com técnicas de otimização diferenciável [Besançon et al., 2024]; e 3) gradiente da previsão em relação aos seus parâmetros, que pode ser obtido com base na estrutura do modelo preditivo.

4. Estrutura do Pacote

O pacote `ApplicationDrivenLearning.jl` foi criado para integrar de forma eficiente a definição e treinamento de modelos preditivos com o processo de otimização em duas etapas descrito pela lógica de malha fechada, mantendo uma interface simples e intuitiva para o usuário. É totalmente implementado na linguagem Julia e foi construído com base no pacote `JuMP.jl` para a descrição, manipulação e solução de modelos de otimização e no pacote `Flux.jl` para a construção de modelos preditivos. Os componentes chave são:

1. **Modelo preditivo:** Define o mapeamento entre variáveis de entrada no conjunto de dados e as previsões declaradas no processo de tomada de decisão utilizando um ou mais modelos Flux parametrizados.

2. Modelo de planejamento: Define o modelo de otimização responsável pela decisão de primeiro nível, que utiliza diretamente as previsões realizadas.
3. Modelo de avaliação: Define o modelo de otimização responsável pela decisão de segundo nível, que modifica, com base em sua estrutura, a decisão de primeiro nível com base nos valores realizados, presentes no conjunto de dados de entrada.
4. Modos de otimização: Algoritmos de otimização dos parâmetros do modelo preditivo com base nos valores resultantes dos modelos de planejamento e avaliação. Entre eles, a formulação binível (BilevelMode), o algoritmo Nelder-Mead (NelderMeadMode) e o novo método do gradiente apresentado na seção 3.0 (GradientMode).

5. Experimentos

Nos experimentos realizados, o modelo preditivo é responsável por gerar previsões de demanda D_t e os requisitos de reserva de subida e descida Rup_t e Rdn_t , que alimentam o modelo de planejamento de despacho e re-despacho de um sistema elétrico. Por fim, o modelo de avaliação é responsável por realizar os ajustes na geração a partir das reservas alocadas. A descrição completa do problema de otimização resultante, assim como as configurações dos sistemas de 118 e 300 barras, pode ser encontrada [Garcia et al., 2024].

Utilizamos como dados históricos um conjunto de séries sintéticas, geradas a partir de um processo auto-regressivo e o modelo treinado é um AR-1, que utiliza o lag de uma etapa como variável auxiliar. Note que o pacote permitiria a utilização de outras informações relevantes, como outros lags, variáveis climáticas, de calendário ou quaisquer outras informações que pudessem ser úteis.

As séries históricas de demanda utilizadas foram de 1200 observações para treino e 360 para teste. Os modos de solução do pacote ApplicationDrivenLearning.jl utilizados foram o NelderMeadMPI e o GradientMPI. Ambos partiram de uma solução inicial resultante do treinamento do modelo preditivo utilizando o método padrão de mínimos quadrados. Cada treinamento teve o tempo máximo total de 10 minutos. A tabela 1 sintetiza os resultados dessa comparação, mostrando a redução de custo dos modelos dirigidos pela aplicação.

Tabela 1. Resultados para sistemas elétricos. Cada experimento é avaliado pelo custo médio de avaliação no conjunto de dados utilizado.

		Treino			Teste		
Barras	Modelo	MQ	Nelder-Mead	Gradiente	MQ	Nelder-Mead	Gradiente
118	Linear (AR-1)	11995	11556	11524	12413	12095	12114
300	Linear (AR-1)	90562	86877	85405	106683	101856	103078

Referências

DIAS GARCIA, Joaquim Dias et al. Application-Driven Learning via Joint Prediction and Optimization of Demand and Reserves Requirement. arXiv preprint arXiv:2102.13273, 2021.

GARCIA, Joaquim Dias; BODIN, Guilherme; STREET, Alexandre. BilevelJuMP. jl: Modeling and solving bilevel optimization problems in Julia. INFORMS Journal on Computing, v. 36, n. 2, p. 327-335, 2024.

BESANÇON, Mathieu et al. Flexible differentiable optimization via model transformations. INFORMS Journal on Computing, v. 36, n. 2, p. 456-478, 2024.