Memorial de Encantamentos: Runas antigas de Redes Neurais e Algoritmos Genéticos

A Guerreira Escuridão: Ana Luz pereira Mendes

Muli, Escola de Encantamentos Ciêntíficos: Ilum - Escola de Ciências

Prof. Responsável: Dr. Daniel Cassar

Agradecimentos

Em nome da sabedoria antiga e das magias dessas runas antigas, ofereço minhas mais sinceras graças à todos que buscam o saber oculto, em especial aos guerreiros que me ajudaram a lutar contra Feras e Monstros:

- Ana Luiza Poletto Loss
- Gabriel Viégas Ribeiro
- Caio Matheus Leão Dantas
- Rafael Anis Shaikhzadeh Santos
- Maria Emily Nayla Gomes da Si

Ao mestre, Dr. Daniel Cassar, que veio antes de mim, cujos ensinamentos sussurram na páginas deste pequeno livro. Aos Encantamentos que deram certo e aqueles que deram errado!

Fica aqui o meu carinho!

Contracapa

"Nada na vida deve ser temido,
apenas compreendido.
Agora é hora de compreender mais,
para que possamos temer menos."
— Marie Curie

1 Grafo: O Monstro

Nome do Monstrinho: 3.1 *"Eu podia jurar que não veria grafos nunca mais na minha vida" — aluno da turma 2024 que não quis se identificar*

Missão: faça em uma folha sulfite o grafo computacional da expressão abaixo.
$L = (y - (ab + cd + ef + g))^2$

Encantamento: Primeiro comecei calculando cada operação matemática como multiplicação e soma, criando dois operadores para cada. Com esses operadores, foi possível realizar a subtração, por meio de uma soma apos uma multiplicação por -1. O segundo passo consiste em realizar a derivada parcial de cada termo em relação a um valor final. Esse processo consiste em analisar o peso de cada termo no valor final, o que ajuda a compreender onde e como é possível mudar algum valor inicial para aumentar ou diminuir o resultado.

Efeito e aprendizado: Essa atividade foi essencial para compreender os princípios básico de uma rede neural, como os cálculos de transmissão de informação ocorre e representar eles com grafos foi bastante útil para isso. Além disso, ferramentas como o Backpropagation foram melhor compreendidas depois dessa missão.

Onde estão os pergaminhos?: Enviei pelo canal da disciplina, o nome do arquivo é "Monstrinho_3-1.ipynb".

2 Um Monstro feito de Bolinhas e Pauzinhos

Nome do Monstrinho: 3.2 *"Átomos não são bolinhas e ligações químicas não são pauzinhos" — Prof. Julio*

Missão: Utilize classes de Python para modelar elementos químicos e moléculas.

Encantamento: Primeiramente, criei uma classe *Elemento* com três atributos (*s_elemento, n_atomico, p_atomico*). Em seguida, construí uma classe *Molécula* utilizando os objetos criados anteriormente na classe *Elemento* para estruturar novos métodos, dentro da classe *Molécula*, em que um método calcula o peso e um outro a formula química. Depois de diversas discussões com colegas de classe, como o José Vitor e a Júlia Guedes cheguei em um formato mais sintético, uma vez que utilizei melhor das chaves do dicionário (está fofinho, vale a pena da uma olhadinha!).

Efeito e aprendizado: Esse caderno é muito importante para consolidar o conceito de classe e entender como elas podem interagir ente si. Além disso, aqui também foi explorado como podemos usar dicionários feitos por objetos de classes e como isso facilita a escrita do código.

Onde estão os pergaminhos?: Encontra-se na minha Cesta de Pergaminhos dentro do GitHub a seguir: https://github.com/LuzMendes/Memorial-de-Encantamentos-Runas-antigas-de-Redes-Neurais-e-Algoritmos-Geneticos?tab=readme-ov-file.

3 Batalha EletroDunder: A Origem

Nome do Monstrinho: 3.3 *Classes em Python não pagam imposto sobre herança.*

Missão: Modelar algum conceito científico utilizando herança de classes.

Encantamento: O conceito cientifico modelado foi o de componentes elétricos. Portanto, criei uma classe mãe chamada *Componentes_Elétricos* com os principais atributos como corrente, resistência e potencia. Além disso, ainda na classe mãe, instanciei um método em comum em que retorna a carga que passou pelo componente elétrico dependendo do tempo em que ele ficou ligado. Após isso, criei três outras classes especificas para cada tipo de componente elétrico, em que serão herdadas algumas características e novas surgiram:

1. Capacitor (novo atributo: capacitância; novo método: *Carga_Acumulada*);
2. Gerador (novo atributo: potência; novo método: *Energia_gerada*)
3. Resistor (novo atributo: potência; novo método: *Energia_dissipada*)

Efeito e aprendizado: Com esse exercício podemos compreender uma forma muito útil de representar conceitos científicos. Esse processo facilita trabalhar e realizar testes com o conceito escolhido. Assim, foi possível concretiza, a partir desse estudo a ideia de herança.

Onde estão os pergaminhos?: Encontra-se na minha Cesta de Pergaminhos dentro do GitHub a seguir: https://github.com/LuzMendes/Memorial-de-Encantamentos-Runas-antigas-de-Redes-Neurais-e-Algoritmos-Geneticos?tab=readme-ov-file.

4 Batalha EletroDunder: O retorno

Nome do Monstrinho: 3.4 `__dunder__`

Missão: Criar uma classe que contenha pelo menos 3 métodos dunder que não foram apresentados no material de aula.

Encantamento: Usei as classes criadas no Monstrinho 3.3, para poder construir circuitos elétricos a partir dos componentes criados anteriormente. Os métodos dunder que implementei foram:

1. __and__ (x & y): Que significa a associação em serie entre um circuito e um novo componente elétrico.

2. __or__ (x | y): Que significa a associação em paralelo entre um circuito e um novo componente elétrico.

3. __matmul__ (x@ y): Que retorna uma tabela com o valor da diferença de potencial e da resistência dos dois circuitos.

Efeito e aprendizado: Esse monstrinho foi muito importante para conhecer novos métodos dunder e usar a criatividade para atribuir novos significados e ações a eles. Como pensamentos futuros fica como ideia criar os métodos __rand__ e __ror__.

Onde estão os pergaminhos?: Encontra-se na minha Cesta de Pergaminhos dentro do GitHub a seguir: https://github.com/LuzMendes/Memorial-de-Encantamentos-Runas-antigas-de-Redes-Neurais-e-Algoritmos-Geneticos?tab=readme-ov-file.

5 A Fera com classe

Nome da Fera: 4.1 *Quem classifica a classe classificadora?*

Legião: Guerreira Escuridão.

Missão: Alterar a rede neural feita em Python puro para resolver um problema de classificação.

Encantamento: Na rede neural feita em Python puro, foi necessário usar como função de ativação uma sigmoide. Segundo, converter as saídas para que retornem tanto a probabilidade quanto as classes.

$$\begin{cases} verde, \text{se } probabilidade > 0.5 \\ velmelho, \text{se } probabilidade \leq 0.5 \end{cases}$$

Por ultimo, para calcular cada perda usei a função de custo *binary cross-entropy*, em que implementei a seguinte formula:

$H_p(q) = -(1/N) \times \sum[y_i \times log(p(y_i)) + (1 - y_i) \times log(1 - p(y_i))]$

Efeito e aprendizado: Podemos perceber que entendendo os diversos passos que uma rede neural possui, temos a liberdade de trabalhar com sua estrutura para resolver diferentes problemas. Esse trabalho foi interessante também para perceber que é necessário utilizar diferentes funções de perda para se adequar melhor ao problema em questão. Para aprender sobre isso foi necessário consultar algumas referências (a referência [1] foi a que achei mais interessante).

Onde estão os pergaminhos?:Encontra-se na minha Cesta de Pergaminhos dentro do GitHub a seguir: https://github.com/LuzMendes/Memorial-de-Encantamentos-Runas-antigas-de-Redes-Neurais-e-Algoritmos-Geneticos?tab=readme-ov-file.

6 A Fera Mascarada

Nome da Fera: 4.3 *Derrube pra fora.*

Legião: Guerreira Escuridão, Ana Luiza Poletto Loss. (A discussão acerca do tema e o código foram realizados em conjunto pelas duas autoras).

Missão: Implemente o regularizador dropout na rede neural feita em Python puro.

Encantamento: Primeiramente, com base na rede neural feita em Python puro, foi necessário usar como função de ativação uma sigmoide. Segundo, converter as saídas para que retornem tanto a probabilidade quanto as classes.

$$\begin{cases} verde, \text{se } probabilidade > 0.5 \\ velmelho, \text{se } probabilidade \leq 0.5 \end{cases}$$

Por ultimo, para calcular cada perda usamos a função de custo *binary cross-entropy*, que segue a seguinte formula:

$H_p(q) = -(1/N) \times \sum[y_i \times log(p(y_i)) + (1 - y_i) \times log(1 - p(y_i))]$

Efeito e aprendizado: Essa Fera Formidável proporcionou, principalmente, uma compreensão mais ampla de como uma rede neural funciona e como podemos realizar alterações nela. Entender sobre o *Dropout*, neste trabalho, permitiu perceber que não é necessário utilizar todos os neurônios de uma rede, mostrando que apenas alguns realmente possuem importância para a predição. Foi interessante observar que o *y_true* e o *y_pred*, não houve uma diferença significativas, ou seja, o desligamento aleatório de alguns neurônios durante o treinamento foi importante para aumentar a capacidade de generalização!

Onde estão os pergaminhos?: Encontra-se na minha Cesta de Pergaminhos dentro do GitHub a seguir: https://github.com/LuzMendes/Memorial-de-Encantamentos-Runas-antigas-de-Redes-Neurais-e-Algoritmos-Geneticos?tab=readme-ov-file.

7 Uma Fera mais profunda (com camadas...)

Nome da Fera: 4.6 *E se meus dados forem imagens?*

Legião: Guerreira Escuridão, Caio Leão Dantas. (A discussão acerca do tema e o código foram realizados em conjunto pelos dois autores)

Missão: implementar uma rede neural convolucional (CNN) utilizando PyTorch ou lightning.

Encantamento: O primeiro passo é entender que uma CNN é composta por diferentes tipos de camadas, geralmente organizadas em blocos convolucionais seguidos por camadas densas (fully connected). Então, usando o Pytorch construímos uma rede neural, entendendo cada tipo de camada, usamos como função de ativação a *nn.ReLU()* e o *nn.MaxPool2d(kernel_ size=2)*. Apos repetir essa ordem para as próximas camadas, adicionamos uma saída *nn.Linear()*. Nessa atividade optamos por dividir nossos dados em lotes para que seja mais fácil convergir os pesos. Além disso, usamos otimizador Adam e a GPU para rodar o código.

Efeito e aprendizado: Essa Fera foi muito importante para entender as principais diferenças entre uma rede neural tradicional e uma CNN, olhando para a estrutura e para a eficiência no processamento de imagens. Além de aprender sobre o otimizador Adam e como usar o Pytorch para construir uma rede neural se preocupando com a sequência de uma CNN.

Onde está os pergaminhos?: Encontra-se na minha Cesta de Pergaminhos dentro do GitHub a seguir: https://github.com/LuzMendes/Memorial-de-Encantamentos-Runas-antigas-de-Redes-Neurais-e-Algoritmos-Geneticos?tab=readme-ov-file.

8 A Fera Secreta

Nome da Fera: 4.9 *A senha de tamanho variável.*

Legião: Guerreira Escuridão, Gabriel Viegas Ribeiro. (A discussão acerca do tema e o código foram realizados em conjunto pelos dois autores)

Missão: Resolver o problema da senha de forma que você não forneça a informação do tamanho da senha para a função que gera a população. Considere que a senha pode ser uma string de 1 até 30 caracteres.

Encantamento: Para esse problema, modificamos 4 aspectos principais do *script*:

1. Criação da população: neste problema, modificamos a função que gera a população (*populacao_senha_s_tamanho*) para gerar populações com tamanhos diferentes de senha usando a função *random.randint* para definir o tamanho da senha.

2. Função objetivo: mudamos a função objetivo para adicionar o valor da diferença entre a senha verdadeira como componente da distância que forma o *fitness*.

3. Cruzamento: decidimos pelo cruzamento de ponto simples, e adicionamos a verificação para que o máximo do corte (*range*) seja escolhido com base no tamanho da menor senha. Colocamos ainda uma condicional que define que se o tamanho máximo for menor que um, não há cruzamento.

4. Mutação: criamos uma nova função mutação baseada na mutação por salto que muda o tamanho da senha, aumentando ou diminuindo.

Os outros aspectos mantemos igual ao algoritmo feito em sala.

Efeito e aprendizado: Essa Fera, foi importante para aprender como os algoritmo genéticos são flexíveis para modelar diversos problemas. Ela também foi crucial para se habituar a como construir a função objetiva, pensando não apenas no seu objetivo, mas no formato do candidato. Além disso, trabalhar com um candidato final desconhecido foi bem desafiador e podemos dizer que não saber a senha não deixou a convergência mais demorada.

Onde está os pergaminhos?: Encontra-se na minha Cesta de Pergaminhos dentro do GitHub a seguir: https://github.com/LuzMendes/Memorial-de-Encantamentos-Runas-antigas-de-Redes-Neurais-e-Algoritmos-Geneticos?tab=readme-ov-file.

9 A Fera Preciosa

Nome da Fera: 4.13 *A liga ternária mais cara do mundo*

Legião: Guerreira Escuridão, Maria Emily Nayla Gomes da Silva.(Ana Luz: Implementação das funções para criar a população, cruzamento e mutação de massas, além da escrita do notebook. No GitHub realizou o Upload do notebook principal; Emily Gomes: Implementação das funções objetivo e de mutação de elemento, além da escrita do notebook. No GitHub realizou o README e o upload do script.)

Missão: : Encontre uma liga de três elementos que tenha o maior custo possível.

Encantamento: Nesta Fera, foi adaptado as funções de algoritmo genético para descrever nosso problema. Então, optamos por criar candidatos que não sejam inválidos, mas para isso usamos a estrutura do candidato como *[Elemento1, Elemento2, Elemento3, Peso1, Peso2, Peso3]*. Já na função objetivo definimos que o *fitness* seria calculado a com diferença entre o custo da liga e seu peso molecular, pois desejamos o maior valor possível e o menor peso atômico. Assim, a diferença cria um sistema de penalidade, ou seja, o melhor desempenho do indivíduo será quando o peso da liga for menor. Pensando no cruzamento, optamos por cruzar apenas a primeira parte do canditado, ou seja, os elementos. Isso porque, com a restrição do peso máximo essa troca poderia gerar muitos indivíduos inválidos. Por fim, fizemos dois tipos de mutação, uma que atua na massa e outra no elemento, fazendo pequenas mudanças. Na mutação do peso, fixamos um dos pesos e trocamos os outros dois de modo a manter o candidato válido, já na mutação do elemento, trocamos aleatoriamente um dos elementos.

Efeito e aprendizado: Um desafio do projeto foi identificar como seria calculado o *fitness*, então optamos por um sistema de penalização, pensado que queriamos máximizar uma característica e minimizar outra. Esse trabalho foi muito interessante para compreender como a função objetiva pode trabalhar com mais de um objetivo e entender a relação entre eles. A criação da estrutura do candidato também foi uma aspecto importante que permeou todas as funções, desde qual parte seria usada na seleção, até como realizar as mutações. Consideramos que obtivemos um excelente resultado, visto que o código convergiu para ligas compostas por elementos de altíssimo valor associado e massas que, embora não sejam as menores, também não estão muito distantes da massa de outros elementos.

Onde está os pergaminhos?: Encontra-se na minha Cesta de Pergaminhos dentro do GitHub a seguir: https://github.com/LuzMendes/Memorial-de-Encantamentos-Runas-antigas-de-Redes-Neurais-e-Algoritmos-Geneticos?tab=readme-ov-file.

Encantamentos Perigosos: Um caminho para as profundezas computacionais

Atenção pequena criatura que está desbravando o mundo das magias. Após passar desta página não terá mais volta, o Reino de Lumi deixará de ser o mesmo!!!

10 LUMI Talks

Nome do Perigo: 6 *Interpretabilidade e explicabilidade de Redes Neurais*

Legião: Guerreira Escuridão, Caio Matheus Leão Dantas e Rafael Anis Shaikhzadeh Santos.(A discussão acerca do tema e o código foram realizados em conjunto pelos três autores).

Missão: Realizar um seminário de 10 a 15 minutos sobre um tema relevante para a disciplina.

Encantamento: No desenvolvimento dessa apresentação, foram abordados os conceitos de interpretabilidade e explicabilidade de redes neurais, modelos frequentemente considerados como "caixas-pretas" e aplicação de três técnicas principais de XAI (*Explainable Artificial Intelligence*): Grad-CAM, LIME e SHAP.

- O Grad-CAM (*Gradient-weighted Class Activation Mapping*) foi apresentado como uma técnica específica para redes neurais convolucionais (CNNs) com resolução baseada na geração de "mapas de calor" que destacam as regiões de uma imagem de entrada que foram mais importantes para uma determinada classificação.

- O LIME (*Local Interpretable Model-agnostic Explanations*) foi introduzido como um método agnóstico de modelo, aplicável a qualquer modelo de Machine Learning com resolução pautada no aprendizado de um modelo interpretável em torno de uma predição específica. Isso é feito através da perturbação dos dados de entrada e observação de como essas perturbações afetam a probabilidade da predição do modelo original.

- Já o SHAP (*SHapley Additive exPlanations*) foi descrito como uma abordagem baseada nos valores de Shapley (advindos da teoria dos jogos) para interpretar predições. A resolução do SHAP consiste em calcular a contribuição marginal média de cada feature para a predição, considerando todas as ordens possíveis de entrada

Efeito e aprendizado: O estudo da interpretabilidade e explicabilidade de redes neurais permitiu o aprendizado sobre algumas das principais técnicas para compreender as decisões de modelos de IA, transformando-os de "caixas-pretas" em sistemas mais transparentes e confiáveis.

O principal aprendizado foi a distinção entre três abordagens proeminentes: LIME, Grad-CAM e SHAP, e as suas vantagens e desvantagens nos processos de explicação e/ou interpretação do modelo.

Onde está os pergaminhos?: Encontra-se na minha Cesta de Pergaminhos dentro do GitHub a seguir: https://github.com/LuzMendes/Memorial-de-Encantamentos-Runas-antigas-de-Redes-Neurais-e-Algoritmos-Geneticos?tab=readme-ov-file.

11 O Tarrasque

Nome do Perigo: 5.1 *Modelagem Preditiva da Resposta Dielétrica de Perovskitas*

Legião: Guerreira Escuridão, Caio Matheus Leão Dantas e Rafael Anis Shaikhzadeh Santos.(A discussão acerca do tema e o código foram realizados em conjunto pelos três autores).

Missão: Identifique e otimize os hiperparâmetros de uma rede neural do tipo MLP para resolver um problema de regressão ou de classificação de interesse científico.

Encantamento: Com esse notebook buscamos realizar uma predição da constante dielétrica de diferentes perovskitas. Fizemos uma otimização de hiperparâmetros por busca em grade, testando 432 arquiteturas diferentes, variando hiperparâmetros como funções de ativação, número de neurônios por camada e taxa de aprendizado. Visto que mantemos o número de épocas constante, decidimos implementar a estratégia de *Early Stopping* para evitar *overfitting*.

Por fim, realizamos uma otimização e avaliamos nosso modelo através do calculo do RSME. Obtivemos um RMSE de 15,44 que não é considerado ideal, pois as constantes dielétricas do nosso dataset também estão na ordem de dezenas. Uma possível explicação para tal valor obtido está nos dados de atributos analisados, que não conseguem capturar exatamente as relações tridimensionais de vizinhança atômica da perovskita. Portanto, esse aspecto pode influenciar na falta de conhecimento do modelo acerca do arranjo do material e variação na taxa de erros. Para melhorar a performance da nossa rede poderíamos tentar diversas estratégias como utilizar reguladores e normalizadores.

Efeito e aprendizado: Esse trabalho foi muito importante para entender, na prática, como usar as ferramentas aprendidas em outros notebooks, mas em um problema real prático. Foi especialmente interessante aprender a estruturar um projeto completo de Redes Neurais, levando em conta as especificidades do problema e observando como diferentes técnicas se comportam diante dele.

Onde está os pergaminhos?: O notebook e todos as arquivos usados pode ser encontrado em nosso GitHub: “https://github.com/LuzMendes/Tarrasque-Modelagem-Preditiva-da-Resposta-Dieletrica-de-Perovskitas.”.

12 Reflexões sobre a jornada

Muito se desvelou, ao longo desta jornada de estudos do Mágicos, sobre as Runas Ocultas das Redes Neuronais Artificiais, que simulam a mente dos magos, e sobre também os Encantamentos das profundezas dos Feitiços dos Algoritmos Genéticos, que otimizam nossos ataques. Revisitando todos os pergaminhos na construção desse memorial percebi o quão foi proveitoso e interessante aprender esses ensinamentos. Com ânimo, confesso: compreendi os fundamentos destes encantamentos, e vislumbro suas aplicações nas batalhas que vão além do Reino da Lumi.

Essas magias computacionais foram muito gratificantes, por que, mesmo que às vezes passei certa raiva no momento de entender alguns conceitos que foram surgindo(dropout kkk), foi muito satisfatório usar a criatividade para lutar contra essas Feras e Monstrinhos. Dessas criaturas místicas, fica no meu coração especialmente duas batalhas: a Batalha EletroDunder, pois é umas das primeiras e mais divertidas de perceber a ciência começar a tomar forma por meio das classes; a Fera Preciosa, que fiquei contente com a maneira como conseguimos contornar os problemas em questão, em especial o uso da ideia de penalização na função objetiva.

Contudo, nem toda jornada é feita apenas de vitórias. O último feitiço lançado não foi suficiente para derrotar o terrível Tarrasque, mas foi forte o bastante para que crescesse em mim a sabedoria e o ânimo para lutar mais batalhas como essa. Obrigada mestre Cassar por essa jornada mística!

13 Referências

[1]ENSINA.AI. Uma explicação visual para função de custo Binary Cross-Entropy (ou Log Loss). Medium, 2017. Disponível em: https://medium.com/ensina-ai/uma-explica%C3%A7%C3%A3o-visual-para-fun%C3%A7%C3%A3o-de-custo-binary-cross-entropy-ou-log-loss-eaee662c396c. Acessoem: 26maio2025.