# Guia do Aventureiro de Lumi

## 1 Contextualização

Parabéns, agora você é um membro oficial da sua Guilda escolhida! Como todos os membros das Guildas, é esperado que você proteja o Reino de Lumi de criaturas mal intencionadas. Além disso, alguns membros também dividem seus conhecimentos adquiridos com os demais nas tão famosas *LUMI Talks* que ocorrem às 8:00 de terça-feira na Taverna do Sol.

Os primeiros seis meses de quem se junta a uma Guilda são conhecidos como o *período probatório*, onde é esperado que os membros mostrem proatividade e desenvolvam diversas atividades em prol da sociedade e de seu desenvolvimento pessoal.

Ao final do período probatório, cada membro descreve as tarefas realizadas no famoso Memorial que será avaliado pelos membros do Comitê das Guildas.

## 2 As atividades

Você tem em mãos um guia com diversas atividades da disciplina de Redes Neurais e Algoritmos Genéticos. Sua tarefa é realizar uma certa quantidade dessas atividades para exercitar os conceitos apresentados na disciplina bem como estudar novos conceitos relacionados.

Ao final da disciplina, cada estudante deverá entregar o seu Memorial. Este é um documento pdf relatando as tarefas realizadas e onde elas podem ser encontradas (nome do arquivo ou link do repositório). O Memorial é como se fosse uma "cola textual" mostrando as atividades realizadas, ele não é um relatório científico.

Assim como em disciplinas passadas, esta disciplina também segue a filosofia "escolha sua própria aventura". Temos abaixo diversas atividades com diversos níveis de dificuldade (e pontuações) diferentes. Algumas atividades podem ser realizadas individualmente, algumas em duplas e algumas em trios. Observe as regras em cada tipo de atividade. Veja na tabela abaixo a pontuação máxima de cada tipo de atividade.

| Atividade | Pontuação máxima |
|---|---:|
| Monstrinhos | 5 |
| Feras formidáveis | 15 |
| Tarrasque | 30 |
| Tiamat | 30 |
| LUMI Talk | 10 |

A pontuação deste semestre é um pouco *fuzzy*, porém o mínimo esperado é que cada estudante realize corretamente um conjunto de atividades que some 50 pontos. Uma boa performance gira em torno de 90 pontos e uma excelente performance gira em torno de 120 pontos.

Assim como é requisito de diversas publicações científicas, atividades realizadas em duplas ou em trios **devem** conter uma seção relatando a contribuição de cada membro. É esperado que todos os membros contribuam de maneira significativa para cada entrega. Em especial, "apenas" escrever o README de um repositório não é uma contribuição suficiente; é necessário contribuir mais do que isso para ser considerado um dos autores da atividade.

**Não é necessário fazer *todas* as atividades.**

Observe que *todas* as atividades da disciplina estão descritas abaixo. Algumas delas só farão sentido após certas aulas.

Todas as entregas realizadas devem conter o nome dos autores, a descrição da contribuição de cada autor (se a tarefa for em equipe), uma introdução sobre o trabalho e uma discussão/conclusão ao final. Todas as entregas devem guiar o leitor com relação aos passos tomados.

Todas as entregas realizadas em repositório tipo git devem conter as seguintes informações no README (não necessariamente nesta ordem):

- o nome dos autores
- o nome da instituição onde foi realizado o trabalho
- o nome do professor responsável da disciplina
- descrição da contribuição de cada autor para o trabalho (se a tarefa foi em equipe)
- agradecimentos a todos que ajudaram (se pertinente)
- texto explicativo breve sobre o que tem no repositório e como executar/navegar os arquivos

# 3 Monstrinhos

Todos os monstrinhos são atividades **individuais**.

## 3.1 "Eu podia jurar que não veria grafos nunca mais na minha vida" — aluno da turma 2024 que não quis se identificar

**Objetivo**: faça em uma folha sulfite o grafo computacional da expressão abaixo.

$$L = (y - (a \cdot b + c \cdot d + e \cdot f + g))^2$$

Realize o forward pass considerando que $L$ é o vértice folha e os seguintes valores

- $y = -50$
- $a = -2$
- $b =$ último dígito da sua matrícula da Ilum
- $c = 8$
- $d = -5$
- $e =$ último dígito do dia do seu aniversário
- $f =$ seu número primo favorito menor que 10
- $g = -10$

Compute os gradientes locais de cada vértice numérico realizando o backpropagation a partir do vértice folha. Indique para cada vértice qual é a respectiva derivada parcial e como a regra da cadeia é aplicada (faça de maneira similar ao que fizemos em sala).

**Comentário**: é necessário que a entrega seja feita à mão em folha sulfite mesmo, não é para fazer digital. Tire uma foto legível ou escaneie para anexar ao seu Memorial e guarde o original.

## 3.2 "Átomos não são bolinhas e ligações químicas não são pauzinhos" — Prof. Julio

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

**Considerações do experimento**: Crie uma classe chamada `Elemento`. *Todo* elemento químico utilizado nesta tarefa deve ser uma instância desta classe. Toda instância da classe `Elemento` deve ter os seguintes atributos: símbolo do elemento, número atômico do elemento e peso atômico do elemento (fique à vontade para incluir outras informações se quiser). Crie uma classe chamada `Molecula`. Esta classe deve receber um dicionário onde as chaves representam os elementos e os valores representam a quantidade do respectivo elemento. A classe `Molecula` deve ter um método capaz de exibir o peso atômico

da molécula criada e um método capaz gerar uma fórmula química (em string) para esta molécula.

**Observação**: não é necessário que a fórmula química siga as regras oficiais da química, basta representar os elementos existentes e suas quantidades.

## 3.3 Classes em Python não pagam imposto sobre herança

**Objetivo**: Modele algum conceito científico utilizando herança de classes.

**Considerações do experimento**: O uso da herança de classes *deve fazer sentido* dentro do contexto científico escolhido, isto é, deve haver uma justificativa bem embasada para o uso de herança de classes na sua entrega. Certifique-se que a classe mãe tem pelo menos um método que não seja dunder para ser herdado pela classe filha. Garanta que a classe filha tem pelo menos um método (dunder ou não) que *justifique* a sua criação.

## 3.4 `__dunder__`

**Objetivo**: Se informe sobre métodos dunder que *não* foram utilizados no material de aula e crie uma classe que contenha pelo menos 3 destes métodos dunder. Faça códigos onde cada um destes métodos dunder seja acessado sem os chamar explicitamente (exemplo: não é para rodar `a.__add__(b)` mas sim `a + b` para o caso do dunder `__add__`).

**Considerações do experimento**: A classe deve conter pelo menos 3 métodos dunder que não foram vistos no material da disciplina. Sua classe deve fazer sentido, isto é, não crie uma classe "sem pé nem cabeça" apenas para a entrega. Reflita sobre uma classe onde os métodos dunder propostos realmente fazem sentido. Na sua entrega, explique brevemente o que fazem os métodos dunder que escolheu e mostre eles em ação com uma instância da sua classe.

## 3.5 Forma, função e ativação

**Objetivo**: implemente 3 novas funções de ativação na rede neural feita em Python puro nos vídeos da disciplina. Escreva brevemente sobre estas 3 funções de ativação, mostrando a equação delas e comentando a diferença com relação à função de ativação sigmoidal. Mostre que seu código funciona rodando alguns testes simples.

**Comentário**: aqui não é o lugar de *inventar* funções de ativação. Busque por funções de ativação já existentes utilizadas em redes neurais.

**Comentário 2**: observe que o enunciado diz claramente que é para realizar a tarefa na rede neural feita em Python puro nos vídeos da disciplina. Se você está usando o `PyTorch`, `numpy`, `tensorflow`, `keras`, `lightning` ou qualquer outra biblioteca pronta, você está no caminho errado!

## 3.6 A curva de aprendizado

**Objetivo**: Implemente o registro da curva de aprendizado tanto dos dados de treino quanto de validação no código de redes neurais feito em Python puro nesta disciplina.

Além de implementar, treine um modelo de rede neural em um conjunto de dados qualquer e mostre e interprete o gráfico das curvas de aprendizado.

**Comentário**: observe que o enunciado diz claramente que é para realizar a tarefa na rede neural feita em Python puro nos vídeos da disciplina. Se você está usando o `PyTorch`, `numpy`, `tensorflow`, `keras`, `lightning` ou qualquer outra biblioteca pronta, você está no caminho errado!

## 3.7 Comparando as performances

**Objetivo**: Compare o tempo de execução de três algoritmos diferentes de otimização (busca aleatória, busca em grade e algoritmos genéticos) para resolver o problema das caixas binárias. Para este exercício, considere como critério de parada para o algoritmo genético e para a busca aleatória o ato de encontrar a resposta do problema. Mantenha o algoritmo de busca em grade com o critério de parada visto na disciplina (isto é, testar todas as possibilidades).

**Lembrete**: nós estudamos como medir o tempo de execução de algoritmos na disciplina Lógica Computacional.

**Dica**: O enunciado do objetivo não definiu o número de caixas. É esperado de um cientista que ele entenda que não adianta resolver esse problema para apenas *um* valor de $n$ caixas, mas sim buscar uma tendência resolvendo esse problema para alguns valores de $n$ diferentes em busca de encontrar um padrão. Considere pelo menos 10 valores de $n$ diferentes na sua resolução.

**Dica 2**: É esperado que um cientista entenda que medir o tempo de execução exige rodar a mesma busca algumas vezes a fim de computar alguma estatística, como a média dos tempos, por exemplo.

## 3.8 A função de Himmelblau

**Objetivo**: Use um algoritmo genético para encontrar as coordenadas $(x, y)$ dos mínimos globais da função de Himmelblau abaixo.

$$f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$$

## 3.9 Alterando os parâmetros da busca

**Objetivo**: Escolha um problema de otimização e proponha um algoritmo genético para resolvê-lo. Altere os parâmetros de busca do algoritmo genético e relate o que observar. Faça pelo menos 5 testes com parâmetros de busca diferentes.

**Dica**: Existem diversos parâmetros que você pode variar no seu algoritmo genético como, por exemplo, o tamanho da população, a taxa de mutação, a chance de cruzamento, o operador de seleção, etc. O problema não especificou qual ou quais parâmetros você deve alterar e também não especificou como fazer essa alteração. Em problemas científicos, sempre que possível costumamos manter todas as variáveis constantes e alterar apenas uma delas de cada vez para ver seu efeito. Se você alterar duas variáveis ao mesmo tempo ficará difícil entender o efeito de cada uma.

## 3.10 Praticamente um X-man!

**Objetivo**: Crie e evolua um algoritmo genético para resolver um problema de otimização que tenha mais do que um operador de mutação. Fica a seu critério se irá permitir que um mesmo indivíduo possa passar por mais de um operador de mutação a cada geração ou se ele deverá passar por apenas um deles caso seja sorteado para mutar. É necessário que os operadores de mutação sejam diferentes para fazer sentido ter mais de um (isto é, não é para usar um mesmo operador e apenas alterar as probabilidades de mutação).

## 3.11 "Criatividade requer coragem" — Henri Matisse

**Objetivo**: Sua tarefa é criar três novos operadores genéticos, um de seleção, um de cruzamento e um de mutação e observar eles em ação. Realize esta atividade no código de algoritmo genético desenvolvido nesta disciplina (isto é, não é para usar o `DEAP`). Conte para o leitor sobre o funcionamento dos novos operadores que criou.

## 3.12 Otimização de hiperparâmetros

**Objetivo**: use algoritmos genéticos para encontrar um bom conjunto de hiperparâmetros em um experimento de aprendizado de máquina. Escolha um algoritmo de aprendizado de máquina que tenha pelo menos 3 hiperparâmetros para serem otimizados.

**Dica**: o enunciado não especificou o problema de aprendizado de máquina. Neste caso, você pode ficar à vontade para escolher o problema que quiser. Se estiver sem ideias, sugiro o dataset dos pinguins e o algoritmo das árvores de decisão que estudamos na disciplina Aprendizado de Máquina.

## 3.13 Reconstrução de imagens

**Objetivo**: Implemente um algoritmo de reconstrução de imagens usando algoritmos genéticos.

**Dica**: Veja o capítulo 11 do livro do Eyal Wirsansky [3]. O desafio desse experimento é entender cada etapa da implementação, não simplesmente entregar a implementação em si (pois ela está no livro…)

# 4 Feras formidáveis

Todas as feras formidáveis são atividades que podem ser realizadas **individualmente** ou em **duplas**.

Em caso de entrega em dupla, é necessário que todos os membros do grupo tenham contribuído significativamente para a entrega e saibam defender o trabalho realizado para pontuarem. Não se esqueçam de acrescentar uma seção relatando a contribuição de cada membro para a entrega.

## 4.1 Quem classifica a classe classificadora?

**Objetivo**: altere a rede neural feita em Python puro para resolver um problema de classificação. Treine uma rede neural em um dataset simples de classificação para mostrar que funciona.

**Comentário**: aqui é necessário se informar sobre as diferenças de uma rede neural classificadora com relação a uma rede neural regressora. A função de perda, por exemplo, não poderá ser mais a função de perda dos resíduos quadrados.

**Comentário 2**: observe que o enunciado diz claramente que é para realizar a tarefa na rede neural feita em Python puro nos vídeos da disciplina. Se você está usando o `PyTorch`, `numpy`, `tensorflow`, `keras`, `lightning` ou qualquer outra biblioteca pronta, você está no caminho errado!

## 4.2 Stop right now, thank you very much

**Objetivo**: implemente uma estratégia de Parada Antecipada (*Early Stopping*) no processo de treino da rede neural feita em Python puro ou no processo de treino da rede neural feita em `PyTorch`

**Comentário**: esta não é para resolver com o módulo `lightning`.

## 4.3 Derrube pra fora

**Objetivo**: implemente o regularizador *dropout* na rede neural feita em Python puro.

**Comentário**: algumas tarefas vão apresentar palavras e conceitos que ainda não vimos em sala. Parte do desafio é justamente se informar sobre estes conceitos.

**Comentário 2**: observe que o enunciado diz claramente que é para realizar a tarefa na rede neural feita em Python puro nos vídeos da disciplina. Se você está usando o `PyTorch`, `numpy`, `tensorflow`, `keras`, `lightning` ou qualquer outra biblioteca pronta, você está no caminho errado!

## 4.4 No alto daquele monte vivia Carlos, o incerto

**Objetivo**: implemente e compute a incerteza de previsão de uma rede neural utilizando a estratégia de Monte Carlo Dropout.

**Sugestão**: faça a tarefa "Derrube pra fora" antes de fazer esta.

## 4.5 Um momento, por favor!

**Objetivo**: implemente o otimizador de Descida do Gradiente com Momento (*Gradient Descent with Momentum*) na rede neural feita em Python puro.

**Comentário**: observe que o enunciado diz claramente que é para realizar a tarefa na rede neural feita em Python puro nos vídeos da disciplina. Se você está usando o `PyTorch`, `numpy`, `tensorflow`, `keras`, `lightning` ou qualquer outra biblioteca pronta, você está no caminho errado!

## 4.6 E se meus dados forem imagens?

**Objetivo**: implementar uma rede neural convolucional (CNN) utilizando `PyTorch` ou `lightning`. Treine esta rede neural em um conjunto de dados de imagens. Explique para o leitor como funciona a camada de convolução de uma CNN e o motivo de utilizarmos este tipo de arquitetura quando estudamos imagens.

**Dica**: um dos mais famosos conjuntos de dados de imagens é o MNIST.

## 4.7 Preciso de um espaço (latente) para pensar

**Objetivo**: implemente uma rede neural do tipo autoencoder utilizando `PyTorch` ou `lightning`. Treine esta rede no conjunto de dados que desejar e comente sobre o que é o codificador, o que é o decodificador e o que é o espaço latente. Representações visuais são bem-vindas se possível.

**Comentário**: diversos exemplos usam datasets de imagens para mostrar como uma rede tipo autoencoder funciona. Quem sabe seja interessante se informar sobre camadas convolucionais caso opte por esta estratégia.

**Dificultando um pouco mais**: existem arquiteturas chamadas de autoencoder variacionais. São um pouco mais complicadas que as não-variacionais, porém são capazes de gerar novos dados. Veja se acha interessante abraçar este desafio adicional.

## 4.8 "... o futuro, assim como o passado, seriam presente diante de seus olhos" — Pierre-Simon de Laplace

**Objetivo**: implementar uma rede neural recorrente (RNN) utilizando `PyTorch` ou `lightning`. Treine esta rede neural em um conjunto de dados sequenciais. Explique para o leitor como funciona a recorrência de uma RNN e o motivo de utilizarmos este tipo de arquitetura quando estudamos dados sequenciais.

## 4.9 A senha de tamanho variável

**Objetivo**: 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.

**Dica**: A função objetivo terá que quantificar em sua métrica tanto se o candidato acertou as letras quanto se acertou o tamanho da senha.

**Dica 2**: Você pode criar diferentes estratégias de mutação, não precisa ser apensa uma! Quem sabe uma função de mutação pode alterar letras e a outra pode alterar o tamanho da senha? Ver o exercício "Praticamente um X-man!".

**Dica 3**: Observe que você terá que pensar um pouco sobre como fará o cruzamento no caso de senhas de tamanhos diferentes. Quem sabe tenha que fazer alguma consideração adicional sobre quais são os valores possíveis para o ponto de corte…

## 4.10 O caixeiro que prefere cidades ímpares

**Objetivo**: Encontre o caminho de menor distância no problema do caixeiro viajante que prefere cidades ímpares e mostre ele de forma gráfica.

**Considerações do experimento**: Considere um número $n \geq 7$ de coordenadas $(x, y)$ de cidades (cada cidade ocupa uma posição $(x, y)$ diferente). Você pode gerar as coordenadas de forma aleatória ou simplesmente usar as coordenadas que desejar. O caixeiro só anda em linha reta e apenas entre duas cidades. O caixeiro começa e termina seu trajeto na mesma cidade e, fora a cidade inicial, ele não visita nenhuma outra cidade mais de uma vez. Além disso, atribua um número inteiro para cada uma das $n$ cidades que o caixeiro irá visitar, iniciando a contagem pelo número zero e aumentando esse número de 1 em 1. O caixeiro deverá necessariamente visitar primeiro as cidades com números ímpares antes das cidades com números pares. A cidade de número zero deve ser a cidade inicial.

## 4.11 Eles estão se multiplicando!

**Objetivo**: Encontre o caminho de menor distância no problema do caixeiro viajante considerando que existe mais do que um caixeiro viajante.

**Considerações do experimento**: Considere um número $n \geq 10$ de coordenadas $(x, y)$ de cidades (cada cidade ocupa uma posição $(x, y)$ diferente). Você pode gerar as coordenadas de forma aleatória ou simplesmente usar as coordenadas que desejar. Os caixeiros só andam em linha reta e apenas entre duas cidades. Todos os caixeiros começam em cidades diferentes e jamais visitam cidades já visitadas por outros caixeiros.

## 4.12 Novos palíndromos

**Objetivo**: Encontre pelo menos 10 palíndromos diferentes de 5 letras. Estes palíndromos devem ter pelo menos uma vogal. Não é necessário que eles formem palavras válidas em português ou qualquer outro idioma.

**Dica**: Nada te impede de inventar um novo operador de mutação. Existe uma forma de melhorar bastante a convergência deste problema com um operador de mutação especializado para esta tarefa.

**Comentário**: o código deve de alguma forma gerar os 10 palíndromos diferentes, mas ninguém disse que eles devem ser encontrados na mesma evolução de um algoritmo genético. Quem sabe evoluir um algoritmo mais de uma vez possa ser uma estratégia válida.

## 4.13 A liga ternária mais cara do mundo

**Objetivo**: Encontre uma liga de três elementos que tenha o maior custo possível. A liga ternária deve ser da forma $x$A.$y$B.$z$C sendo que $x + y + z = 100\,\text{g}$, $x \geq 5\,\text{g}$, $y \geq 5\,\text{g}$, $z \geq 5\,\text{g}$ e "A", "B" e "C" são elementos químicos diferentes. Utilize o preço dado abaixo [1]. Considere que qualquer composto com 3 elementos químicos é chamado de liga.

**Dica**: Pode ser interessante criar uma função que gere a população inicial para garantir que $x + y + z = 100\,\text{g}$.

**Dica 2**: Pode ser interessante criar uma ou mais funções de cruzamento e mutação neste problema de forma que todas elas garantam que, ao final do processo do operador, os indivíduos mantenham a característica de ter $x + y + z = 100\,\text{g}$.

```
# preço em dólares por quilograma
preco = {
    "H": 1.39,
    "He": 24,
    "Li": 85.6,
    "Be": 857,
    "B": 3.68,
    "C": 0.122,
    "N": 0.14,
    "O": 0.154,
    "F": 2.16,
    "Ne": 240,
    "Na": 3.43,
    "Mg": 2.32,
    "Al": 1.79,
    "Si": 1.7,
    "P": 2.69,
    "S": 0.0926,
    "Cl": 0.082,
    "Ar": 0.931,
    "K": 13.6,
    "Ca": 2.35,
    "Sc": 3460,
    "Ti": 11.7,
    "V": 385,
    "Cr": 9.4,
    "Mn": 1.82,
    "Fe": 0.424,
    "Co": 32.8,
    "Ni": 13.9,
    "Cu": 6,
    "Zn": 2.55,
    "Ga": 148,
    "Ge": 1010,
    "As": 1.31,
    "Se": 21.4,
    "Br": 4.39,
    "Kr": 290,
    "Rb": 15500,
```

```
    "Sr": 6.68,
    "Y": 31,
    "Nb": 85.6,
    "Mo": 40.1,
    "Tc": 100000,
    "Ru": 10600,
    "Rh": 147000,
    "Pd": 49500,
    "Ag": 521,
    "Cd": 2.73,
    "In": 167,
    "Sn": 18.7,
    "Sb": 5.79,
    "Te": 63.5,
    "I": 35,
    "Xe": 1800,
    "Cs": 61800,
    "Ba": 0.275,
    "La": 4.92,
    "Ce": 4.71,
    "Pr": 103,
    "Nd": 57.5,
    "Pm": 460000,
    "Sm": 13.9,
    "Eu": 31.4,
    "Gd": 28.6,
    "Tb": 658,
    "Dy": 307,
    "Ho": 57.1,
    "Er": 26.4,
    "Tm": 3000,
    "Yb": 17.1,
    "Lu": 643,
    "Hf": 900,
    "Ta": 312,
    "W": 35.3,
    "Re": 4150,
    "Os": 12000,
    "Ir": 56200,
    "Pt": 27800,
    "Hg": 30.2,
    "Tl": 4200,
    "Pb": 2,
    "Bi": 6.36,
    "Po": 49200000000000,
    "Ac": 29000000000000,
    "Th": 287,
    "Pa": 280000,
    "U": 101,
    "Np": 660000,
    "Pu": 6490000,
    "Am": 750000,
```

```
    "Cm": 160000000000,
    "Bk": 185000000000,
    "Cf": 185000000000,
}
```

## 4.14 A liga ternária leve mais cara do mundo

**Objetivo**: Encontre uma liga de três elementos que tenha o maior custo e o menor peso atômico. A liga ternária deve ser da forma $x$A.$y$B.$z$C sendo que $x + y + z = 100\,\text{g}$, $x \geq 5\,\text{g}$, $y \geq 5\,\text{g}$, $z \geq 5\,\text{g}$ e "A", "B" e "C" são elementos químicos diferentes. Utilize o preço dado no exercício anterior e peso atômico dados abaixo [3]. Considere que qualquer composto com 3 elementos químicos é chamado de liga.

```
# peso atômico em gramas por mol
peso_atomico = {
    "H": 1.008,
    "He": 4.002602,
    "Li": 6.94,
    "Be": 9.0121831,
    "B": 10.81,
    "C": 12.011,
    "N": 14.007,
    "O": 15.999,
    "F": 18.998403163,
    "Ne": 20.1797,
    "Na": 22.98976928,
    "Mg": 24.305,
    "Al": 26.9815385,
    "Si": 28.085,
    "P": 30.973761998,
    "S": 32.06,
    "Cl": 35.45,
    "Ar": 39.948,
    "K": 39.0983,
    "Ca": 40.078,
    "Sc": 44.955908,
    "Ti": 47.867,
    "V": 50.9415,
    "Cr": 51.9961,
    "Mn": 54.938044,
    "Fe": 55.845,
    "Co": 58.933194,
    "Ni": 58.6934,
    "Cu": 63.546,
    "Zn": 65.38,
    "Ga": 69.723,
    "Ge": 72.63,
    "As": 74.921595,
    "Se": 78.971,
    "Br": 79.904,
    "Kr": 83.798,
```

```
    "Rb": 85.4678,
    "Sr": 87.62,
    "Y": 88.90584,
    "Nb": 92.90637,
    "Mo": 95.95,
    "Tc": 97.90721,
    "Ru": 101.07,
    "Rh": 102.9055,
    "Pd": 106.42,
    "Ag": 107.8682,
    "Cd": 112.414,
    "In": 114.818,
    "Sn": 118.71,
    "Sb": 121.76,
    "Te": 127.6,
    "I": 126.90447,
    "Xe": 131.293,
    "Cs": 132.90545196,
    "Ba": 137.327,
    "La": 138.90547,
    "Ce": 140.116,
    "Pr": 140.90766,
    "Nd": 144.242,
    "Pm": 144.91276,
    "Sm": 150.36,
    "Eu": 151.964,
    "Gd": 157.25,
    "Tb": 158.92535,
    "Dy": 162.5,
    "Ho": 164.93033,
    "Er": 167.259,
    "Tm": 168.93422,
    "Yb": 173.045,
    "Lu": 174.9668,
    "Hf": 178.49,
    "Ta": 180.94788,
    "W": 183.84,
    "Re": 186.207,
    "Os": 190.23,
    "Ir": 192.217,
    "Pt": 195.084,
    "Hg": 200.592,
    "Tl": 204.38,
    "Pb": 207.2,
    "Bi": 208.9804,
    "Po": 209.0,
    "Ac": 227.0,
    "Th": 232.0377,
    "Pa": 231.03588,
    "U": 238.02891,
    "Np": 237.0,
    "Pu": 244.0,
```

```
    "Am": 243.0,
    "Cm": 247.0,
    "Bk": 247.0,
    "Cf": 251.0,
}
```

## 4.15 Vai pra lá ou vem pra cá!

**Objetivo**: Implemente o operador genético de migração no código de algoritmo genético desenvolvido nesta disciplina (isto é, não é para usar o `DEAP`). Conte para o leitor sobre como a sua implementação funciona e mostre ela em ação.

## 4.16 "Se não está funcionando, aumenta a temperatura que dá certo" — Autor anônimo

**Objetivo**: Proponha um problema de otimização com restrição. Implemente uma rotina de **recozimento simulado** (*simulated annealing*) no código de algoritmo genético desenvolvido nesta disciplina (isto é, não é para usar o `DEAP`). Utilize de um algoritmo genético com recozimento simulado para resolver o problema com restrição escolhido. Aproveite e conte para o leitor como funciona o processo de recozimento simulado que você implementou.

## 4.17 Regressão simbólica

**Objetivo**: Faça um algoritmo de regressão simbólica e resolva um problema com ele.

**Sugestão**: É fortemente recomendado usar o módulo `DEAP` para resolver esse, já que ele já tem alguns métodos implementados. Veja um exemplo de como seguir em frente neste link: https://deap.readthedocs.io/en/master/examples/gp_symbreg.html. É importante não apenas fazer funcionar, mas entender o que cada parte está fazendo (esse é o desafio). Não vale usar módulos prontos de regressão simbólica pra resolver esse problema (mas se algum dia você precisar, busque pelo `pysr` que é excelente!). Este é um problema de programação genética, que é um subtipo de algoritmos genéticos. Veja o capítulo 12 do livro do Eyal Wirsansky [3].

# 5 Os terríveis Tarrasque e Tiamat

Os terríveis Tarrasque e Tiamat são atividades que podem ser realizadas em **duplas** ou em **trios**.

É necessário que todos os membros do grupo tenham contribuído significativamente para a entrega e saibam defender o trabalho realizado para pontuarem. Não se esqueçam de acrescentar uma seção relatando a contribuição de cada membro para a entrega.

## 5.1 Tarrasque

**Objetivo**: Identifique e otimize os hiperparâmetros de uma rede neural do tipo MLP para resolver um problema de regressão ou de classificação de interesse científico. Espera-se o uso de boas práticas em ciências de dados assim como os discutidos nesta disciplina e os já apresentados em disciplinas anteriores. Teste ao menos 100 diferentes arquiteturas de rede durante sua otimização de hiperparâmetros. Os dados para treinar o modelo devem ser dados tabulados (isto é, não podem ser imagens, textos, áudios ou vídeos) e não podem ser dados sequenciais (como, por exemplo, séries temporais). Evite o uso de datasets didáticos como o dos pinguins ou o do titanic. É necessário apresentar o problema escolhido ao professor para aprovação (pode enviar uma mensagem no teams se preferir).

**Nota**: Esta tarefa requer um repositório git individual apenas para ela.

## 5.2 Tiamat

**Objetivo**: Identifique um problema de otimização de interesse científico (este deve ser aprovado pelo professor, pode enviar uma mensagem no teams se preferir). Proponha e evolua um algoritmo genético com o propósito de resolver este problema. Utilize pelo menos um operador genético que não tenha sido apresentado no material da disciplina. Explique como este operador genético funciona na sua entrega. Se tiver interesse, pode criar seu próprio operador genético. Não se esqueça de discutir sobre o(s) resultado(s) obtido(s).

**Nota**: Esta tarefa requer um repositório git individual apenas para ela.

# 6 LUMI Talks

Os *LUMI Talks* são atividades que devem ser realizadas em **trios**.

Tratam-se de seminários de 10 a 15 minutos sobre temas relevantes para a disciplina.

Alguns temas pré-aprovados para os *LUMI Talks* estão abaixo. Caso deseje apresentar algum outro tema, é necessário antes aprovação do professor.

1. Redes Neurais sob a óptica da Álgebra Linear
2. Engenharia de atributos (*feature engineering*)
3. Métodos de regularização de redes neurais (*dropout*, *batch normalization*, etc.)
4. Transferência de aprendizado (*transfer learning*)
5. Interpretabilidade e explicabilidade de Redes Neurais
6. Outros otimizadores (AdaGrad, RMSProp e/ou Adam)
7. Redes Neurais Convolucionais (CNNs)
8. Redes Neurais Recorrentes (RNNs)
9. Redes Neurais LSTM
    - Este tema deve ocorrer após o tema 8 ter sido apresentado
10. Redes Neurais tipo AutoEncoders
11. Redes Neurais Geracionais (GANs)
12. Redes Neurais de Grafos
13. Redes Neurais de Kolmogorov-Arnold

É necessário estar presente durante a apresentação para receber a pontuação. Infelizmente, como não existe uma entrega física dos *LUMI Talks*, não é possível pontuar no caso de não ter participado da apresentação oral.

# 7 Referências


1. Material suplementar de FORREST, Robert M.; GREER, A. Lindsay. Evolutionary design of machine-learning-predicted bulk metallic glasses. Digital Discovery, 2023.
2. MENTEL, Łukasz. mendeleev – A Python resource for properties of chemical elements, ions and isotopes. Disponível em: https://github.com/lmmentel/mendeleev.
3. EYAL WIRSANSKY. Hands-On Genetic Algorithms with Python: Applying genetic algorithms to solve real-world deep learning and artificial intelligence problems, 2020.