Universidade Federal do Rio Grande do Sul **Inteligência Artificial**

Instituto de Informática **Prof. Joel Luís Carbonera**

Departamento de Informática Aplicada

Trabalho 1
Aprendizado supervisionado

Instruções preliminares

As implementações devem ser realizadas em Python 3. Caso precise instalar bibliotecas adicionais (assuma que, quando o ambiente de execução não for mencionado, os códigos serão executados em uma máquina virtual com uma instalação padrão do Ubuntu e interpretador Python 3.8 com Miniconda, pip, numba, numpy, pandas e Tensorflow 2.1), descreva as bibliotecas adicionais no seu `Readme.md`.

O trabalho deve ser desenvolvido em equipes com **3 membros** preferencialmente, embora equipes de **2 membros** sejam aceitas também. O trabalho **não pode ser realizado individualmente**.

1. Regressão linear

O objetivo deste exercício é a implementação do algoritmo de gradiente descendente / descida de gradiente para regressão linear.

Você irá implementar regressão linear para prever o preço de fazendas em Alegrete, a partir de um conjunto de dados fictício. O conjunto é um arquivo com valores separados por vírgula (`alegrete.csv`), onde a primeira coluna contém a área do terreno, em hectares, e a segunda contém o preço, em milhares de reais.

As funções da regressão linear devem ser implementadas no arquivo `alegrete.py`. Nas funções a seguir, considere o parâmetro b como representando o coeficiente linear, e o parâmetro w como representando o coeficiente angular. As funções a serem implementadas são:

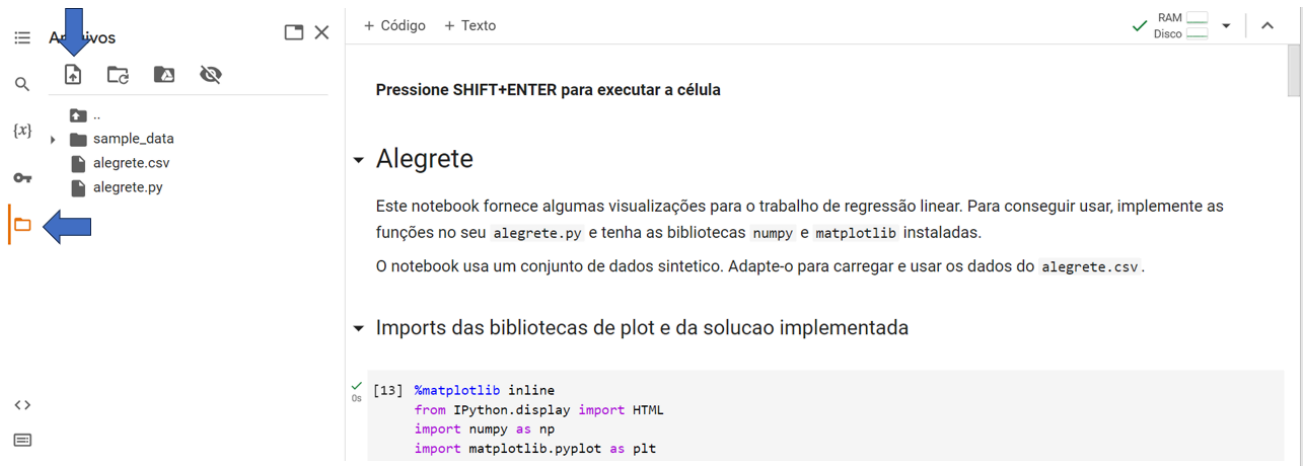
- `compute_mse(b, w, data):` recebe os parâmetros da regressão (coeficiente linear e coeficiente angular) e o conjunto de dados (matriz com a área do terreno na primeira coluna e o preço na segunda coluna) e retorna o erro quadrático médio (MSE – mean squared error) da regressão.
- `step_gradient(b, w, data, alpha):` recebe os parâmetros da regressão (coeficiente linear e angular), o conjunto de dados (matriz com a área do terreno na primeira coluna e o preço na segunda coluna) e a taxa de aprendizado. Executa **uma** atualização por descida do gradiente e retorna os valores atualizados para `b` e `w`.
- `fit(data, b, w, alpha, num_iterations):` recebe o conjunto de dados (matriz com a área do terreno na primeira coluna e o preço na segunda coluna), os valores

iniciais dos parâmetros da regressão (coeficiente linear e angular), a taxa de aprendizado e o número de épocas/iterações. Para cada época/iteração, executa uma atualização por descida do gradiente e registra os valores atualizados de `b` e `w`. Ao final, retorna duas listas, uma com os `b` e outra com os `w` obtidos ao longo da execução (o último valor das listas deve corresponder à última época/iteração).

O arquivo `alegrete.ipynb` é um Jupyter Notebook com funções para você visualizar os resultados da sua implementação. Note que o **carregamento dos dados** de entrada deve ser realizado **neste notebook e não no arquivo** `alegrete.py`. Divirta-se com a animação com a sequência de retas encontradas por seu algoritmo e a curva de custo final.

No seu `Readme.md`, coloque os valores iniciais de `b`, `w`, valores de `alpha` e `num_iterations` que resultem na melhor execução da sua regressão linear. Coloque também o erro quadrático médio obtido. Se sua implementação estiver boa, não se preocupe em encontrar bons `b` e `w` iniciais. Você pode inicializá-los aleatoriamente que o algoritmo irá encontrar uma reta que faça um bom ajuste.

Ao trabalhar no ambiente colab, você pode abrir o arquivo `alegrete.ipynb` nele. Para isso, no ambiente colab, vá em Arquivo>Abrir notebook>upload e selecione o arquivo. Depois de abrir o notebook `alegrete.ipynb` no ambiente, para que seja possível executar o código do arquivo `alegrete.py` com os dados do arquivo `alegrete.csv`, você deve carregá-los no ambiente para que fiquem disponíveis na sessão do colab. Para isso, você deve acessar o os arquivos da sessão do colab, na barra lateral esquerda. E depois utilizar a funcionalidade de upload na parte superior da área que se abre, conforme indicado na figura abaixo.



Curiosidade: Você pode pensar que a regressão linear, conforme estudada em aula, pode ser realizada por um único perceptron, que recebe uma única entrada e que tem um único peso na conexão com esta entrada, além do bias e utilizando uma função de ativação linear. O peso representa o coeficiente angular e o bias representa o coeficiente linear.

2. Tensorflow/Keras

O objetivo deste exercício é avaliar uma das ferramentas consagradas de aprendizado profundo, avaliando seu desempenho em conjuntos de dados simples e as dificuldades observadas em problemas mais desafiadores.

Você deve utilizar como base, o notebook chamado `Trabalho_redes_neurais.ipynb`, fornecido com o kit do trabalho. Este notebook já mostra uma particularidade para trabalhar com dados de imagens coloridas, que é o fato de possuírem 3 canais de cores.

Você deve preencher as funções determinadas no notebook para retornarem a rede neural com a melhor configuração que você tiver encontrado (camadas e neurônios, principalmente, embora outros hiperparâmetros possam ser testados também) para os datasets MNIST, Fashion MNIST, CIFAR-10 e CIFAR-100.

Não é obrigatório atingir um nível pré-definido de acurácia. O que importa é a análise dos fatores que levam a um desempenho bom ou ruim. Técnicas que melhoram o desempenho das redes neurais em datasets mais desafiadores estão além do escopo do trabalho, mas se você conseguir pesquisar e usar essas técnicas, parabéns (e pontinho extra)!

Coloque no seu `Readme.md` as características de cada dataset: quantas classes, quantas amostras e qual o tamanho das imagens (altura x largura x canais de cor). Para cada dataset, faça pelo menos 5 testes com configurações diferentes e busque responder as seguintes questões

1) Investigue e reflita sobre os fatores que tornam os problemas de classificação de cada dataset mais ou menos complexos em cada dataset. Pense em uma relação de ordem de complexidade/dificuldade dos datasets (os datasets em ordem dos menos complexos/difíceis para os mais complexos/difíceis) e justifique a resposta. O mais importante nesta questão é a investigação e a reflexão e não o fato de a resposta estar precisa. O esperado nesta questão é a sequência de datasets em ordem de complexidade/dificuldade, seguido de uma hipótese de quais características do dataset influenciaram o seu ranking. Exemplo:

Dataset_1: É o mais simples/fácil pelos motivos x, y e x.
Dataset_2: É mais difícil que o Dataset_1 pelos motivos a,b e c.
Dataset_3: É mais difícil que os demais, por conta de x, y,z.
…

2) Qual a maior acurácia obtida em cada dataset e quais mudanças (em termos de configurações da estrutura da rede ou outros hiperparâmetros) fizeram a performance melhorar (ou pior, caso tenha ocorrido piora em relação a alguma performance já avaliada).

Observação: Você podem explorar aspectos extras (normalização, decaimento de taxa de aprendizagem, outras operações nas redes convolucionais, etc).

3. Entrega

Você deve entregar um arquivo `.zip` contendo:

- O arquivo `alegrete.py` com as funções da regressão linear preenchidas.
- O arquivo `alegrete.ipynb` com o código adequado para carregar o dataset. Para exportar o notebook no Colab, vá em Arquivo → Fazer Download → Fazer download do .ipynb.
- O arquivo `Trabalho_redes_neurais.ipynb` com a implementação da rede neural com a melhor performance para cada dataset
- Outros arquivos de código fonte que forem necessários;
- Um arquivo `Readme.md` em texto sem formatação ou Markdown com o seguinte conteúdo:

Nomes, cartões de matrícula e turma (turma da manhã é A, e turma da tarde é B) dos integrantes do grupo;

- Valores iniciais de `b`, `w`, valores de `alpha` e `num_iterations` que resultem na melhor execução da sua regressão linear.
- Especifique também o melhor erro quadrático médio obtido na sua implementação da regressão linear.
- Sua análise dos datasets (quantas classes, quantas amostras, qual o tamanho das imagens (altura x largura x canais de cor).
- Suas conclusões considerando as questões do Exercício 1.
- Suas conclusões considerando as questões do Exercício 2.
- Documentação do(s) extra(s) implementado(s), se aplicável.

4. Critérios de avaliação

O trabalho será avaliado de acordo com os seguintes critérios:

Item	**Percentual da nota**
Regressão Linear	
Implementação: está de acordo com a especificação?	20
Execução: código executa conforme especificação?	10
Readme.md: Informações sobre a regressão linear estão completas?	5
Readme.md: Parâmetros reportados realmente atingem o MSE reportado?	10
Tensorflow/Keras	
Implementação: está de acordo com a especificação?	20
Execução: resultados obtidos ao executar consistentes com o reportado? Não é esperado que sejam idênticos, devido ao não determinismo do processo, quando não controlamos as seeds de randomização, mas caso os valores sejam muito discrepantes, o professor pode requisitar esclarecimentos.	20
Interpretação: análise dos resultados está adequada?	15
Geral	
Extras, e.g.: normalização de features na regressão linear, técnicas para melhorar o desempenho no Tensorflow, etc. Documentar no Readme.md!	10

Total	**110**

5. Observações gerais

- Apesar do total somar 110, a nota satura em 100. Os 10% extras podem ajudar a compensar eventuais perdas de notas nos demais critério.
- O trabalho deve ser feito em grupos de 2 a 3 integrantes.
- Fiquem atentos à política de plágio!

Política de Plágio

Grupos poderão apenas discutir questões de alto nível relativas a resolução do problema em questão. Poderão discutir, por exemplo, questões sobre as estruturas de dados utilizadas, técnicas para visualizar os dados, etc. Não é permitido que os grupos utilizem quaisquer códigos-fonte provido por outros grupos, ou encontrados na internet.

Pode-se fazer consultas na internet ou em livros apenas para estudar o modo de funcionamento das técnicas de IA, e para analisar o pseudo-código que as implementa. Não é permitida a cópia de implementações concretas (em quaisquer linguagens de programação) das técnicas. O objetivo deste trabalho é justamente descobrir as dificuldades envolvidas na resolução do problema em questão. Toda e qualquer fonte consultada pelo grupo (tanto para estudar os métodos a serem utilizadas, quanto para verificar a estruturação da técnica) precisa obrigatoriamente ser citada no Readme.md.

Usamos rotineiramente um sistema anti-plágio que compara o código-fonte desenvolvido pelos grupos com soluções enviadas em edições passadas da disciplina, e também com implementações disponíveis online.

Qualquer nível de plágio (ou seja, utilização de implementações que não tenham sido 100% desenvolvidas pelo grupo) poderá resultar em nota zero no trabalho. Caso a cópia tenha sido feita de outro grupo da disciplina, todos os envolvidos (não apenas os que copiaram) serão penalizados.