FGV EMAp
ESCOLA DE MATEMÁTICA APLICADA

Técnicas e Algoritmos em Ciência de Dados
Tarefa 3

Este trabalho deve ser enviado até o dia 17 de maio de 2024, às 10hs.
Envios atrasados serão penalizados em 10% por hora de atraso.

Tópicos avaliados

Esta tarefa oferece uma oportunidade emocionante para os alunos colocarem em prática seus conhecimentos adquiridos em sala de aula, usando redes neurais para resolver problemas do mundo real, tanto na classificação quanto na regressão. Os alunos aplicarão os conceitos que aprenderam para construir, treinar e otimizar redes neurais, usando um conjunto de validação para ajustar hiperparâmetros. Os alunos também se acostumarão a gerar gráficos importantes durante o treinamento para analisar o comportamento dos modelos. Ao final do projeto, os alunos terão adquirido experiência prática na implementação de redes neurais.

Instruções

Identificador
Por favor, use o **mesmo número aleatório de 6 dígitos que você usou para as TAREFAS 1,2** e escreva-o na primeira célula do notebook. Certifique-se de manter uma cópia desse número, pois ele será usado para fornecer o feedback.

Submissão
Envie seus arquivos através do ECLASS. Os arquivos que você envia não podem ser substituídos por mais ninguém e não podem ser lidos por nenhum outro aluno. No entanto, você pode substituir seu envio quantas vezes quiser, reenviando, embora apenas a última versão enviada seja mantida.

Se você tiver problemas, no último minuto, envie sua tarefa por e-mail como um anexo em alberto.paccanaro@fgv.br com o assunto "URGENTE – SUBMISSÃO TAREFA 3". No corpo da mensagem, explique o motivo de não enviar através do ECLASS.

IMPORTANTE

- Seu envio consistirá em um único notebook do Python implementando suas soluções.
- **O nome do arquivo será o número aleatório que o identifica (por exemplo, 568423.ipynb)**
- Esta tarefa consiste em 2 partes. Certifique-se de que o código de ambas as partes é colocado nas células de código relevantes no notebook.
- NÃO ENVIE NENHUM CONJUNTO DE DADOS, apenas o código.
- Qualquer função auxiliar que você usará deve ser incluída no notebook – não envie scripts adicionais.

Todo o trabalho que você enviar deve ser exclusivamente seu próprio trabalho. As submissões serão verificadas para isso.

Critérios de avaliação

Esta tarefa é obrigatória e vale 10% da sua nota final total para este curso. Para obter notas máximas para cada pergunta, você deve respondê-la corretamente, mas também completamente. Daremos pontos para códigos bem estruturados.

TAREFA

Parte 1 – Regressão com Redes Neurais (valor: 50%)

Download a partir do ECLASS:

- Tarefa_3_template.ipynb
- energy_efficiency.csv

Descrição do Conjunto de Dados e Problema: neste exercício, você usará o conjunto de dados de Previsão de Eficiência Energética. Este conjunto contém informações sobre a eficiência energética de edifícios com base em oito características, incluindo o tamanho do edifício, a orientação e o tipo de materiais de construção utilizados. O conjunto inclui dois alvos: carga térmica e carga de resfriamento, que representam a energia necessária para aquecer e resfriar o edifício, respectivamente.
Este conjunto de dados é útil para construir redes neurais que preveem a eficiência energética dos edifícios, o que é um problema importante no campo da energia sustentável. O conjunto foi utilizado em vários trabalhos de pesquisa em aprendizado de máquina e apresenta um desafio de regressão interessante.

Descrição do Exercício: Previsão de Eficiência Energética com Redes Neurais Neste exercício, você usará o conjunto de dados de Previsão de Eficiência Energética fornecido. Você construirá e treinará uma rede neural para prever a carga térmica (coluna rotulada como y1 no conjunto de dados) e a carga de resfriamento (coluna rotulada como y2) de um edifício com base em suas características de eficiência energética.

Para completar este exercício, você escreverá código para construir e treinar redes neurais para este problema:

1. Divida o conjunto de dados em conjuntos de treinamento, validação e teste usando uma proporção de 70:15:15.

2. Use numpy para construir uma rede neural que receba as características de eficiência energética como entrada e preveja as cargas térmica e de resfriamento como saídas. Você escolherá o número de neurônios por camada e o número de camadas, mas cada camada terá o mesmo número de neurônios. Esses dois valores devem ser parâmetros de entrada para sua rede neural. Ou seja, você não pode codificar cada camada de forma fixa, o que significa que você terá que escrever código que possa trabalhar com um número variável de camadas e neurônios.

3. Codifique o passo direto e o algoritmo de retropropagação para aprender os pesos da rede neural. Use o conjunto de treinamento para treinar a rede neural e atualize os pesos usando o método de descida de gradiente estocástico. Para as camadas ocultas,

use a função de ativação sigmoidal. Você precisará regularizar sua rede neural usando weight decay, ou seja, você incluirá um termo de regularização na sua função de erro.

4. Monitore o treinamento plotando as perdas de treinamento e validação ao longo das épocas.

O desempenho de sua rede neural será diferente dependendo do número de camadas, número de neurônios por camada e o valor de λ que controla a quantidade de weight decay. Você experimentará com 3 valores de λ: 0 (sem weight decay), 0.001 e 0.0001.
Para escolher a melhor configuração de rede e avaliar seu desempenho, você:

1. Escolherá 3 possíveis valores do número de camadas ocultas (por exemplo, de 1 a 3) e 3 diferentes valores de neurônios por camada (por exemplo, 100, 200 e 300), mas você também pode escolher valores diferentes.

2. Calculará a perda para cada configuração no conjunto de validação.

3. Gerará 3 heatmaps, um para cada valor do parâmetro de regularização λ, mostrando a perda no conjunto de validação ao plotar o número de camadas e número de neurônios em uma grade. Isso ajudará você a visualizar a melhor configuração para a rede neural.

4. Treinará seu modelo final selecionando a melhor combinação de hiperparâmetros e avaliará o desempenho final da rede neural usando o conjunto de teste e o erro quadrático médio da raiz como métrica e relatará isso.

Importante:
- Treine por 50 epochs.
- Defina a taxa de aprendizado η para 0,01.

Parte 2 – Classificação com Redes Neurais (valor: 50%)

Baixe os dados do ECLASS:

- drug_side_effects.csv
- drug_features.csv

Descrição do Conjunto de Dados: neste exercício, você construirá e treinará uma rede neural para prever a ocorrência de efeitos colaterais de medicamentos. O conjunto de dados é derivado do conjunto de dados SIDER, contendo efeitos colaterais relativamente comuns que podem ocorrer para pelo menos 50 medicamentos. Isso produz um total de 740 medicamentos e 256 efeitos colaterais. As características representam várias propriedades moleculares, incluindo peso molecular, número de átomos, número de anéis, número de doadores e aceitadores de ligação de hidrogênio, logP, área de superfície polar topológica (TPSA), número de ligações rotativas, número de anéis aromáticos, número de anéis alifáticos, número de anéis saturados e número de heteroátomos.

Lembre-se de que cada medicamento pode causar muitos efeitos colaterais, não apenas um. *Sinta-se livre para explorar o conjunto de dados e verificar os potenciais efeitos colaterais de medicamentos populares!*

Para completar este exercício, siga estas etapas:

1. Carregue o conjunto de dados e divida-o em conjuntos de treinamento, validação e teste usando uma proporção de 80:10:10.

2. Padronize os features removendo a média e escalonando para variância unitária. Para fazer isso, realize o seguinte para cada feature (coluna) no conjunto de dados:
    a. Calcule a média e o desvio padrão em todo o conjunto de treinamento para esse feature.
    b. Subtraia a média de cada valor nesse feature e divida pelo desvio padrão.
    c. Aplique a mesma transformação aos conjuntos de validação e teste usando a média e o desvio padrão calculados a partir do conjunto de treinamento.

    Observação: você precisa codificar esta parte, não é permitido usar o scikit-learn.

    A normalização dos features é importante para redes neurais porque:
    - *Garante que todas as características tenham a mesma escala, evitando que certas características dominem o processo de aprendizagem devido à sua maior magnitude.*
    - *Melhora a estabilidade numérica do processo de treinamento, tornando a rede neural menos sensível à escolha da taxa de aprendizado e outros hiperparâmetros.*

3. Construa uma rede neural usando NumPy que receba as características como entrada e preveja a ocorrência de efeitos colaterais. Você escolherá o número de neurônios por camada e o número de camadas. Você fornecerá essas informações como uma lista de

entrada, onde o comprimento da lista determina o número de camadas ocultas e cada elemento é o número de neurônios dessa camada oculta. Por exemplo, uma lista `layers = [64, 128, 256]` deveria produzir uma rede com 4 camadas, sendo 3 camadas ocultas com 64, 128 e 256 neurônios cada. Para as camadas ocultas, use a função de ativação sigmoidal. Você precisará regularizar sua rede neural usando weight decay, ou seja, você incluirá um termo de regularização na sua função de erro.

4. Codifique o passo direto e o algoritmo de retropropagação para aprender os pesos da rede neural. Use o conjunto de treinamento para treinar a rede neural e atualize os pesos usando o método de descida de gradiente estocástico. Não se esqueça dos vieses.

5. Monitore o treinamento plotando as perdas de treinamento e validação ao longo das épocas.

    a. *Observação: certifique-se de que a perda diminua durante o treinamento; valores aceitáveis estão entre 0.2 e 2.8 aproximadamente. Esses valores dependem da escolha dos diferentes hiperparâmetros. Teste apenas valores sensatos, levando em conta o conjunto de dados, ou seja, número de características, medicamentos, efeitos colaterais.*

O desempenho de sua rede neural será diferente dependendo do número de camadas, número de neurônios por camada e o valor de λ que controla a quantidade de weight decay. Você experimentará com 3 valores de λ: 0 (sem decaimento de peso), 1 e 0.01.
Para escolher a melhor configuração de rede e avaliar seu desempenho, você:

1. Para cada valor de λ, selecione 3 configurações diferentes de camada (note que neste exercício o número de neurônios por camada não precisa ser o mesmo para cada camada).

2. Calcule a perda para cada configuração no conjunto de validação.

3. No final deste processo, você terá 9 valores de perda (um para cada configuração). Treine seu modelo final selecionando a melhor combinação de hiperparâmetros e avalie o desempenho final da rede neural usando o conjunto de teste e a Área Sob a Curva ROC (AUROC) com a função fornecida no Jupyter notebook.

    a. *Observação: não espere valores impressionantes de AUROC, pois este é um problema altamente complexo que não pode ser resolvido facilmente com uma rede neural simples com features padrão. Espere valores na faixa de 0.55 a 0.75.*

Importante:
- Treine por 50 epochs.
- Defina a taxa de aprendizado η para 0.01.