ACH2016 - INTELIGÊNCIA ARTIFICIAL – 2021
ESPECIFICAÇÃO DO PRIMEIRO TRABALHO
Profa. Patrícia Oliveira

Data de apresentação: 04/11/2021

Entrega no e-disciplinas: até o dia 03/11 às 23:59. Devem ser postados o código fonte, o código executável (junto com eventuais instruções de execução do mesmo) e a apresentação em PowerPoint. Basta uma postagem por grupo. Não se esqueçam de colocar a identificação dos integrantes nos arquivos editáveis.

Observação 1: Este trabalho deve ser realizado por um grupo de no mínimo 3 e no máximo 4 alunos.

Observação 2: Pode ser utilizada qualquer linguagem de programação (Java, Python, etc). Fica vedado o uso de softwares e pacotes utilitários que já contenham algoritmos de aprendizado de máquina implementados (por exemplo, Scikit learn e WEKA).

Observação 3: Plágios, mesmo que parciais, serão punidos com nota zero para os copiadores e copiados envolvidos. O mesmo acontecerá caso seja detectado plágio com arquivos disponíveis na internet.

==========

Esse trabalho tem como objetivo geral a implementação da rede neural Perceptron Multicamadas (MPL), com o algoritmo de aprendizagem Backpropagation.

A tarefa a ser desempenhada por essa rede consiste em reconhecer 10 (dez) classes de imagens binárias 32x32, que representam dígitos manuscritos.

Os dados a serem processados pertencem à base de dados da UCI (University of Califórnia, Irvine) e o download destes pode ser feito a partir do seguinte endereço:

https://archive.ics.uci.edu/ml/machine-learning-databases/optdigits/

Duas versões desse conjunto de dados estão disponíveis:

1) Os dados no formato original são encontrados nos arquivos com prefixo **optdigits-orig**. Observe esses dados para entender de que se trata o problema. Entretanto, para reduzir o número de entradas da rede, vocês devem utilizar os dados no formato pré-processado, conforme descrito no item seguinte.

2) Os dados pré-processados são encontrados nos arquivos **optdigits.tra e optdigits.tes**.

Para construir uma representação reduzida dos dados, cada uma das imagens 32x32 foi dividida em blocos 4x4 disjuntos e o número de pixels com 1´s foi contado para cada um desses blocos.

Esse procedimento gerou, para cada imagem, uma matriz 8x8 em que cada elemento é um inteiro entre 0 e 16.

Nos arquivos **optdigits.tra** e **optdigits.tes**, cada linha representa uma dessas matrizes (imagens pré-processadas), cujas linhas foram concatenadas. Vocês devem trabalhar, portanto, com esses dois conjuntos.

Parte 1: Preparação dos dados

a) Junte os dois conjuntos **optdigits.tra** e **optdigits.tes**, de modo a formar um único conjunto de dados, com o nome **optdigits.dat**.

b) Normalize o conjunto de dados, tal que todos os valores apresentados à rede estejam no intervalo (-1, 1). Uma maneira bastante simplificada de normalizar os dados consiste em somar todos os valores de um determinado exemplo e dividir cada um desses valores pela soma obtida. Ou seja, para cada linha do conjunto de dados, some todos os valores presentes nessa linha e divida cada um desses valores pela soma obtida. **OBS:** a última coluna (rótulo de classe) não deve ser levada em conta nesse processo, afinal, esse valor não compõe a entrada da rede.

c) Represente a saída desejada para cada exemplo como um vetor binário com 10 elementos, no qual só um desses elementos (com índice correspondente à classe) recebe o valor 1 (um), e ao restante é atribuído o valor 0 (zero).

Parte 2: Escolha da arquitetura e de outros parâmetros da rede

Implemente o algoritmo backpropagation para uma rede MLP com uma única camada oculta para resolver o problema de classificação dos dados descritos acima.

Nessa etapa, devem ser realizados experimentos preliminares que permitam que o grupo escolha a melhor arquitetura para a rede (número de neurônios na camada oculta) e a taxa de aprendizado adequada e decida se deve ou não incorporar o termo momentum à regra de atualização dos pesos da rede.

A forma com a qual os experimentos foram conduzidos, além dos resultados e impressões obtidos nessa fase devem ser descritos na apresentação deste trabalho. Mais detalhes sobre o conteúdo dessa apresentação serão expostos no final desta especificação.

Importante: os valores dos pesos das conexões que chegam em cada neurônio também devem estar normalizados entre (-1, 1). Por exemplo, se no neurônio 1, chegam conexões

com pesos w_{11}, w_{21} e w_{31}, esses valores devem ser somados ($S=w_{11} + w_{21} + w_{31}$) e cada um desses valores deve ser dividido pela soma obtida ($w_{11}=w_{11}/S$, $w_{21}=w_{21}/S$, $w_{31}=w_{31}/S$).

Parte 3: Treinamento e Avaliação de Desempenho da rede MLP utilizando a Abordagem Holdout.

Aqui deve ser feita a escolha dos hiperparâmetros do modelo. O programa para a Parte 3 deve ser implementado para execução via linha de comando, tendo como argumentos os nomes dos arquivos de entrada (conjunto de dados) e de saída (resultados obtidos), e então:

a) Carregar o conjunto de dados.

b) Dividir, aleatoriamente, os dados em três conjuntos de tamanhos iguais: treinamento, validação e teste.

c) Inicializar a rede com valores aleatórios (entre -1 e 1).

d) Treinar a rede usando o conjunto de treinamento para atualizar os pesos e o conjunto de validação para conferir a convergência do algoritmo de aprendizagem.

e) Para cada época, registrar, no arquivo de saída, o erro quadrático para o conjunto de treinamento, conjunto de validação e conjunto de teste. Gerar um gráfico com os erros obtidos (curva de aprendizado).

f) Registrar, no arquivo de saída, o número de épocas para a convergência da rede e o erro verdadeiro do modelo.

g) Utilizando o conjunto de teste, calcular e registrar, no arquivo de saída, a matriz de confusão 10x10 para a rede.

O programa pode ser escrito em Java, C/C++, ou qualquer outra linguagem que gere código executável.

Parte 4: Treinamento e Avaliação de Desempenho da rede MLP utilizando a Abordagem Crossvalidation para o melhor modelo encontrado na Parte 3.

O programa para a Parte 4 deve ser implementado para execução via linha de comando, tendo como argumentos os nomes dos arquivos de entrada (conjunto de dados) e de saída (resultados obtidos) e, então:

a) Carregar o conjunto de dados.

b) Aleatoriamente, dividir esse conjunto em 10 (dez) partições (folds).

c) Executar o treinamento (e teste) da rede utilizando o método Crossvalidation.

d) Registrar, no arquivo de saída, os erros verdadeiros obtidos pela rede para cada uma das 10 etapas do Crossvalidation e a estimativa final para o desempenho do modelo.

O programa pode ser escrito em Java, C/C++, ou qualquer outra linguagem que gere código executável.

Devem ser entregues ao final deste trabalho:

a) Todos os códigos fontes e executáveis referentes às Partes 3 e 4.

b) Uma apresentação em Powerpoint que inclua:

   i) **Para a Parte 2:** a descrição dos experimentos realizados para escolha da melhor arquitetura de rede para o problema, incluindo ainda discussões sobre a escolha da taxa de aprendizado e a utilização ou não do termo momentum.

   ii) **Para a Parte 3:** uma discussão sobre o desempenho da rede, levando em conta os erros obtidos pelo modelo para os conjuntos de treinamento, validação e teste e a matriz de confusão calculada. Apresente os gráficos de curva de aprendizado que reforcem essa discussão. Para os resultados obtidos, o que vocês podem concluir ao relacionar o erro sobre o conjunto de treinamento e o desempenho estimado do modelo?

   iii) **Para a Parte 4:** uma análise que discuta os erros obtidos pela rede para cada uma das 10 etapas do Crossvalidation e a estimativa final para o desempenho do modelo.

   iv) Suas impressões gerais e conclusões.