Examen de Métodos de Inteligencias Artificial

Equipo:

- Sharon Nicole Manzano
- Samuel Gómez
- Ana Paula Moreno
- Ana Sofía Avila

Instrucciones. *Lee cuidadosamente cada pregunta y contesta lo que se te pide. No olvides escribir subir todos los archivos .m elaborados para cada inciso que requiera de uso de software y pegar el código de programa indentado y comentado.*

1. **(5 puntos) En caso de realizar un pronóstico a 3 días a futuro con el Perceptrón Multicapa y con el modelo Adaline ¿Qué diferencia existiría?**

Adaline (modelo lineal):

- El modelo generaría una proyección estrictamente lineal, ya que solo puede aprender relaciones lineales entre las variables.
- El resultado sería poco flexible, incapaz de capturar patrones complejos o no lineales en la serie.
- Tiende a producir pronósticos más suavizados y con menor capacidad de adaptación a cambios bruscos.

Perceptrón Multicapa – MLP (modelo no lineal):

- Puede modelar relaciones no lineales gracias a sus capas ocultas y funciones de activación.
- Puede capturar patrones reales más complejos, estacionalidades o comportamientos no-lineales presentes en los datos.
- Produce una proyección más ajustada a la dinámica real de la serie, especialmente cuando ésta presenta curvaturas o interacciones entre variables.

2. **(5 puntos) Determine la cantidad de pesos de una red multicapa con las siguientes características:**

   6 variables de entrada
   4 capas en donde la capa uno tiene 8 neuronas, la capa dos tiene 5 neuronas, la capa 3 cuenta con 10 neuronas y la capa de salida con 3 neuronas.

$$Capa\ 1\ -> (6 + 1\ bias) * 8\ neuronas = 56$$
$$Capa\ 2\ -> (8 + 1\ bias) * 5\ neuronas = 45$$
$$Capa\ 3\ -> (5 + 1\ bias) * 10\ neuronas = 60$$

Examen de Métodos de Inteligencias Artificial

$$Capa\ 4\ \ -> (10\ +\ 1\ bias)\ *\ 3\ neuronas\ =\ 33$$
La red tiene un total de 194 *pesos*

3. **(5 puntos) Explica brevemente qué propusieron Levenberg-Marquardt para mejorar la función de costo en una red multicapa.**

Lo que propusieron con el método de Levenberg-Marquardt que es una mejora del algoritmo tradicional de entrenamiento basado en gradiente, fue combinar el método de Gauss–Newton con el descenso por gradiente, creando un algoritmo híbrido que ajusta los pesos de manera más eficiente. Esto dio como resultado, un modelo que:

- Acelera la convergencia cuando la red está cerca de una buena solución (como Gauss–Newton)
- Evita inestabilidades cuando la red está lejos de la solución, actuando como un descenso por gradiente
- Modifica la actualización de pesos usando un factor de amortiguamiento (λ) que controla si el método se comporta más como Gauss–Newton o como gradiente

4. **(5 puntos) Explica las ventajas y desventajas que hay en un perceptrón multicapa al tener una o muchas neuronas a la salida en un problema de clasificación supervisada.**

1 neurona:

- **Ventajas:** Simple, pocos pesos, menos riesgo de sobreajuste, rápida
- **Desventajas**: Solo binaria, no da probabilidades por clase

Varias neuronas:

- **Ventajas:** Multiclase, da probabilidades, flexible
- **Desventajas:** Más compleja, riesgo de overfitting

5. **(30 puntos)** Considere la base de datos contenida en el archivo **Emisiones2015.xlsx**. El conjunto de datos cuenta con 7384 datos (En la **hoja gt_2015**), 9 variables de entrada (AT, AP, AH, AFDPP, GTEP, TIT, TAT, TEY, CDP) y 2 variables de salida (CO y NOX).
   La información de la base (para evitar hacer omisiones en la traducción) de datos puede consultarse en:
   https://archive.ics.uci.edu/dataset/551/gas+turbine+co+and+nox+emission+data+set

   Referencia: Gas Turbine CO and NOx Emission Data Set [Dataset]. (2019). UCI Machine Learning Repository. https://doi.org/10.24432/C5WC95.

Examen de Métodos de Inteligencias Artificial

"Predicting CO and NOx emissions from gas turbines: novel data and a benchmark PEMS." Turkish Journal of Electrical Engineering and Computer Sciences 27.6 (2019): 4783-4796

a) **(10 puntos)** Utilice el 80% de los datos para entrenar el modelo neurona multicapa. Simule la totalidad de los datos. (Un modelo aceptable debe tener un J menor a 8.1, al simular todos los datos). Indique si se propuso un modelo conjunto para las dos salidas, o tuvo que realizar modelos independientes por salida para alcanzar este desempeño.

b) **(10 puntos)** Proporcione el desempeño global del modelo utilizado en su estimación y el desempeño por cada variable de salida.

c) **(5 puntos)** Señale las características de su modelo (Número de neuronas, capas, variables usadas) y calcule cuantos pesos tiene.

d) **(5 puntos)** Estime las emisiones de CO y NOX para la turbina de gas con los datos proporcionados en la hoja con el nombre **predecir**.

Código:

```
clear all; close all; clc;
%% Load data
% Cargo los datos desde el archivo de Excel
nombreArchivo = 'Emisiones2015.xlsx';
T = readtable(nombreArchivo, 'Sheet', 'gt_2015');
% X = variables de entrada (9)
% Y = variables de salida (CO y NOX)
X = T{:, {'AT','AP','AH','AFDP','GTEP','TIT','TAT','TEY','CDP'}};
Y = T{:, {'CO','NOX'}};
%% Separación de datos
% Mezclo los datos para evitar orden temporal
N = size(X,1);
idxPerm = randperm(N);
X = X(idxPerm,:);
Y = Y(idxPerm,:);
%% Partición (80% - 20% con cvpartition, igual que tu DNA)
cv = cvpartition(N, 'HoldOut', 0.20);
Xtrain = X(training(cv), :);
Ytrain = Y(training(cv), :);
Xtest  = X(test(cv), :);
Ytest  = Y(test(cv), :);
%% Normalización (muy importante en regresión)
muX = mean(Xtrain,1);
sigmaX = std(Xtrain,0,1);
sigmaX(sigmaX == 0) = 1;
Xtrain_norm = (Xtrain - muX)./sigmaX;
Xtest_norm  = (Xtest  - muX)./sigmaX;
%% Modelo
% Red con arquitectura fija (tu mejor modelo)
red = feedforwardnet([60 40 30 10 10]);
% Función de entrenamiento LM (regresión)
red.trainFcn = 'trainlm';
% División interna para entrenamiento / validación / prueba
red.divideFcn = 'dividerand';
red.divideParam.trainRatio = 0.7;
red.divideParam.valRatio   = 0.15;
red.divideParam.testRatio  = 0.15;
% Entrenamiento
red = train(red, Xtrain_norm', Ytrain');
%% Simulación (TRAIN)
Ygtrain = (red(Xtrain_norm'))';   % salida de la red en TRAIN
```

Examen de Métodos de Inteligencias Artificial

```
% Medidas de desempeño (regresión solo J)
Jtrain_global = perform(red, Ytrain', Ygtrain');
Jtrain_CO     = perform(red, Ytrain(:,1)', Ygtrain(:,1)');
Jtrain_NOX    = perform(red, Ytrain(:,2)', Ygtrain(:,2)');
[Jtrain_global  Jtrain_CO  Jtrain_NOX]
%% Test
Ygtest = (red(Xtest_norm'))';  % salida en TEST
Jtest_global = perform(red, Ytest', Ygtest');
Jtest_CO     = perform(red, Ytest(:,1)', Ygtest(:,1)');
Jtest_NOX    = perform(red, Ytest(:,2)', Ygtest(:,2)');
[Jtest_global  Jtest_CO  Jtest_NOX]
%% Simulación con TODOS los datos (para el criterio J < 8.1)
X_all_norm = (X - muX)./sigmaX;
Yg_all = (red(X_all_norm'))';
J_global = perform(red, Y', Yg_all');
J_CO     = perform(red, Y(:,1)', Yg_all(:,1)');
J_NOX    = perform(red, Y(:,2)', Yg_all(:,2)');
disp('Desempeño TODOS LOS DATOS')
[J_global  J_CO  J_NOX]
if J_global < 8.1
   disp('>>> Cumple criterio J < 8.1');
else
   disp('>>> NO cumple criterio J < 8.1');
end
%% Arquitectura y número de pesos
nPesos = sum(red.numWeightElements);
disp(['Numero total de pesos: ' num2str(nPesos)])
%% Predicciones para hoja "predecir"
T_pred = readtable(nombreArchivo, 'Sheet', 'predecir');
X_pred = T_pred{:, {'AT','AP','AH','AFDP','GTEP','TIT','TAT','TEY','CDP'}};
X_pred_norm = (X_pred - muX)./sigmaX;
Y_pred = (red(X_pred_norm'))';
T_pred.CO_pred  = Y_pred(:,1);
T_pred.NOX_pred = Y_pred(:,2);
disp('Predicciones hoja predecir')
disp(T_pred)
```

Resultados:

```
ans =

   5.3457    1.4884    9.2029


ans =

   9.9964    2.8123   17.1805
Desempeño TODOS LOS DATOS

ans =

   6.2753    1.7530   10.7976

>>> Cumple criterio J < 8.1
Numero total de pesos: 4712
--- Predicciones hoja predecir ---
```

AT	AP	AH	AFDP	GTEP	TIT	TAT	TEY	CDP	CO_pred	NOX_pred
5.1391	1021.5	59.337	3.9638	23.215	1077.3	550.09	134.44	11.873	2.397	62.406
10.692	1006.9	78.953	3.8344	32.126	1098.4	536.59	159.27	13.714	2.7224	61.211
-6.2348	1035.8	63.962	2.9854	26.928	1078.1	549.88	134.24	11.783	4.5931	80.287

Conclusión del ejercicio:

La red neuronal que entrenamos funciona bien para predecir las emisiones de CO y NOX de la turbina de gas. Usamos el 80% de los datos para entrenar y al evaluar todo el conjunto, el modelo

obtuvo un error global de **J =6.2**, que está por debajo del límite de **8.1**, así que cumple con lo que se pide. No fue necesario hacer dos modelos separados, uno para CO y otro para NOX; un solo modelo logró predecir ambos valores.

Además, las predicciones para los datos nuevos de la hoja predecir son coherentes y tienen valores similares a los que aparecen en los datos reales. En pocas palabras, el modelo es correcto, funciona bien y da resultados confiables.

6. **(50 puntos)** Cargue el conjunto de datos con nombre **forest_training.csv** La base cuenta con 198 datos de bosques, basados en análisis espectral de imágenes tomadas vía satélite. El conjunto de datos cuenta con 27 variables y una variable de salida asociada al tipo de bosque. La fuente original puede consultarse en:
   https://archive.ics.uci.edu/ml/datasets/Forest+type+mapping

   Referencia: Johnson, B., Tateishi, R., Xie, Z., 2012. Using geographically-weighted variables for image classification. Remote Sensing Letters, 3 (6), 491-499.

```
clear all; close all; clc;

%% Load Data

load Data_6.mat

%% Partición

% Ys

clases = forest_training{:,1};  % string

Ytrain = zeros(size(clases));

Ytrain(clases == "d") = 1;

Ytrain(clases == "h") = 2;

Ytrain(clases == "s") = 3;

Ytrain(clases == "o") = 4;

clases = forest_testing{:,1};  % string

Ytest = zeros(size(clases));

Ytest(clases == "d") = 1;

Ytest(clases == "h") = 2;

Ytest(clases == "s") = 3;

Ytest(clases == "o") = 4;

% Xs
```

```
Xtrain = table2array(forest_training(:, 2:end));  % Features for training
Xtest = table2array(forest_testing(:, 2:end));    % Features for testing
%% Modelo
% red = feedforwardnet([10 10 10 10 10]);  % Tipo de red
% red.trainFcn = 'trainrp';
% red = train(red, Xtrain', Ytrain');
load red_6a.mat
%% Simulación
Ygtrain = (red(Xtrain'))';
Ygtrain(Ygtrain > 4,:) = 4;
Ygtrain(Ygtrain < 1,:) = 1;
Ygtrain = round(Ygtrain);
%% Medidas de desempeño
A = confusionmat(Ytrain, Ygtrain);
figure(1)
confusionchart(A, [1,2,3,4])
exatrain = sum(diag(A)) / sum(A(:));
pretrain = mean(diag(A) ./ sum(A, 2));
rectrain = mean(diag(A) ./ sum(A, 1)');
Jtrain = perform(red, Ytrain, Ygtrain)
Train_metrics = [exatrain, pretrain, rectrain]
%% Test
Ygtest = (red(Xtest'))';
Ygtest(Ygtest > 4,:) = 4;
Ygtest(Ygtest < 1,:) = 1;
Ygtest = round(Ygtest);
B = confusionmat(Ytest, Ygtest);
figure(2)
confusionchart(B, [1,2,3,4])
exatest = sum(diag(B)) / sum(B(:));
pretest = mean(diag(B) ./ sum(B, 2));
```

Examen de Métodos de Inteligencias Artificial

```
rectest = mean(diag(B) ./ sum(B, 1)');

Jtest = perform(red, Ytest, Ygtest);

Test_metrics = [exatest pretest rectest]
```

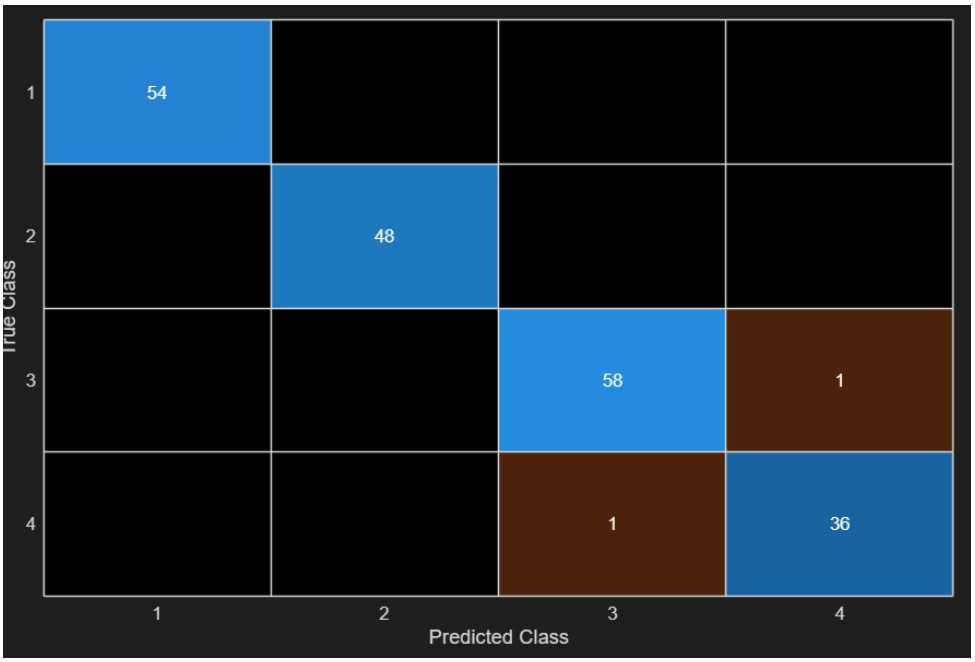
a) (5 puntos) Transforme las categorías de texto a categorías numéricas.

```
clases = forest_training{:,1};  % string
Ytrain = zeros(size(clases));

Ytrain(clases == "d") = 1;
Ytrain(clases == "h") = 2;
Ytrain(clases == "s") = 3;
Ytrain(clases == "o") = 4;

clases = forest_testing{:,1};  % string
Ytest = zeros(size(clases));

Ytest(clases == "d") = 1;
Ytest(clases == "h") = 2;
Ytest(clases == "s") = 3;
Ytest(clases == "o") = 4;
```

b) (5 puntos) Obtenga un modelo neuronal multicapa en donde J sea menor o igual a 0.09

```
Jtrain =

    0.0101
```

c) (5 puntos) Obtenga y proporcione la matriz de confusión del entrenamiento.

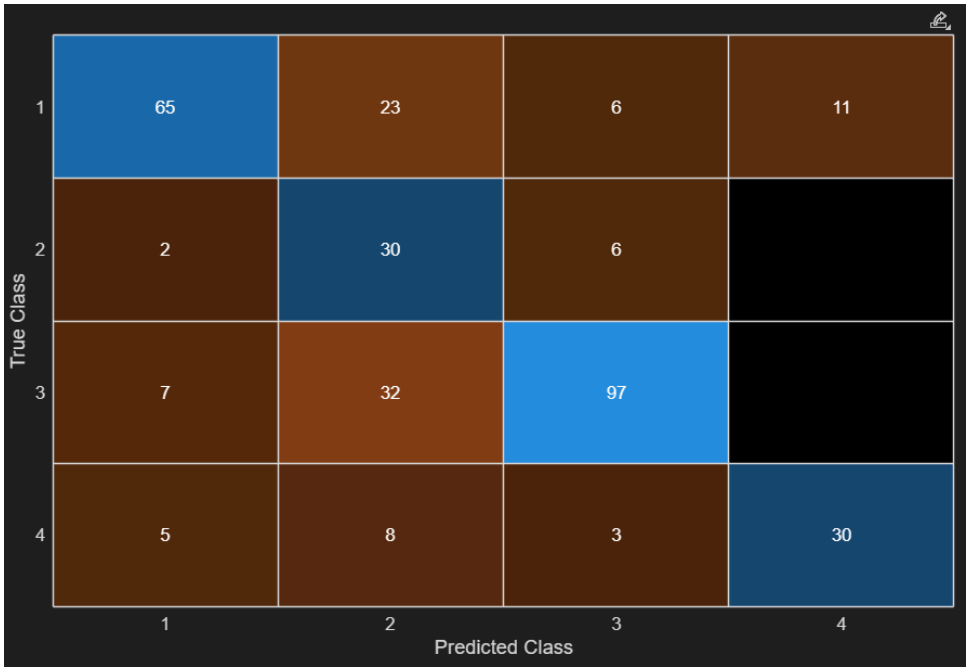


Examen de Métodos de Inteligencias Artificial

d) (5 puntos) Clasifique los datos contenidos en el forest_test.csv

```
Ygtest = (red(Xtest'))';

Ygtest(Ygtest > 4,:) = 4;
Ygtest(Ygtest < 1,:) = 1;

Ygtest = round(Ygtest);
```

e) (5 puntos) Obtenga y proporcione la matriz de confusión de la prueba.



f) (5 puntos) Calcule la exactitud del modelo, tanto del entrenamiento como de la validación.

f) (10 puntos) Calcule la precisión y el recall del modelo, sólo del entrenamiento.

Métricas (exa, pre, rec)

Training	Test
Train_metrics = 0.9899 0.9890 0.9890	Test_metrics = 0.6831 0.6935 0.6858

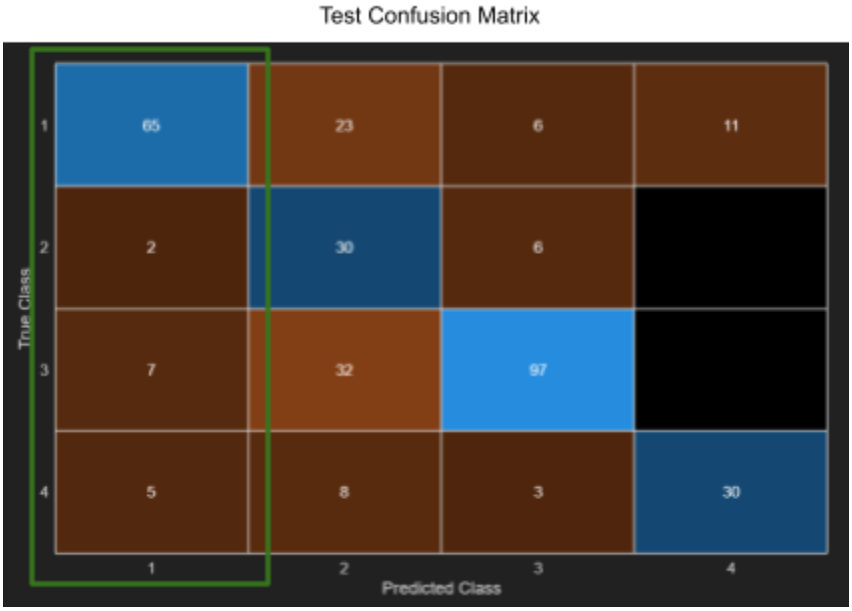
g) (5 puntos) Si se cambia a 4 salidas binarias y se hace un nuevo modelo, ¿Cambiaría el resultado obtenido? Justifique su respuesta

El resultado sí cambiaría. En el modelo actual se utiliza una sola salida numérica (1–4), por lo que la red trabaja como un modelo de regresión y posteriormente se redondea la predicción para asignar la clase. Esto implica que la red aprende una relación artificial de orden entre las clases, lo cual no es del todo correcto en teoría.

Al utilizar 4 salidas binarias, la red se transforma en un clasificador multiclase adecuado. En este caso, cada clase cuenta con su propia neurona de salida y la decisión se basa en la probabilidad más alta que dé, no en un valor numérico continuo.

Por estas razones, las métricas de desempeño sí cambiarán, y en la mayoría de los casos el desempeño mejora debido a que esta alternativa está mejor alineada con la naturaleza del problema.

h) (5 puntos) Tome una columna de una de las matrices de confusión y proporcione la interpretación de los números que contiene.



La primera columna de la matriz de confusión representa todas las veces que el modelo clasificó como clase 1, sin importar cuál haya sido su clase verdadera. Los valores indican cuántas observaciones de cada clase real fueron asignadas a esta categoría por el modelo:

- 65 casos cuya clase verdadera era 1 fueron correctamente clasificados como 1 (Los Verdaderos Positivos)
- 2 casos de la clase verdadera 2 fueron clasificados incorrectamente como 1.
- 7 casos de la clase verdadera 3 fueron clasificados incorrectamente como 1.

- 5 casos de la clase verdadera 4 fueron también clasificados incorrectamente como 1.

Referencia:

Levenberg-Marquardt. (n.d.). ScienceDirect. https://www.sciencedirect.com/topics/computer-science/levenberg-marquardt