Juan Moreno y Diego Cano

MEMORIA PRÁCTICA 2

Apartado 1.1

La implementación de nuestro algoritmo de retropropagación está dividida en 3 partes:

1. Creación de la red neuronal y su arquitectura
2. Por cada época, entrenar la red neuronal con los datos de entrenamiento (sacamos gráficas de ecm y porcentaje de aciertos una vez acabe el entrenamiento)
3. Una vez acabadas las épocas, probar el modelo con el conjunto de test (sacamos el score global del modelo y si se ejecutar el problema_real2 para predecir, sacamos sus etiquetas)

1. Para la creación de la red, realizamos los siguientes pasos

- Creamos una red neuronal
- Creamos las 3 capas principales (entrada, oculta y salida)
- Añadimos las neuronas de entrada a su capa correspondiente
- Añadimos las neuronas ocultas a su capa correspondiente
- Añadimos las neuronas de salida a su capa correspondiente
- Conectamos la capa de entrada con la capa oculta inicializando los pesos a valores aleatorios entre -0.5 y 0.5
- Conectamos la capa oculta con la capa de salida inicializando los pesos a valores aleatorios entre -0.5 y 0.5
- Añadimos las 3 capas principales a la red

2. Para el funcionamiento principal del algoritmo hemos tenido en cuenta los siguientes pasos

- Por cada época ejecutamos todo el conjunto de entrenamiento
- Por cada ejemplo en el conjunto de entrenamiento se realiza lo siguiente:
    - Damos valor a las neuronas de entrada con el ejemplo de entrenamiento e inicializamos el sesgo (1.0)
    - Disparamos los valores de la capa de entrada, inicializamos la capa oculta (y su sesgo) y propagamos la capa de entrada.
    - Disparamos los valores de la capa oculta, inicializamos la capa de salida y propagamos la capa oculta.
    - Disparamos la capa de salida y ya tendríamos disponibles los valores de salida de la red neuronal.
    - Realizamos algunas inicializaciones para cálculos posteriores y también realizamos cálculos necesarios para poder sacar el ecm y el porcentaje de aciertos.
    - Comienza la retropropagación del error

- Calculamos el error de cada neurona de salida que haya en la red
- Vemos las conexiones entre la capa oculta y la capa de salida y calculamos las correcciones de los pesos, aprovechamos para actualizar esos pesos y también vamos calculando los delta_in para poder seguir retropropagando el error.
- Calculamos los errores de las neuronas de la capa oculta
- Retropropagamos el error calculando las correcciones de peso entre la capa de entrada y la capa de salida. Aprovechamos para actualizar pesos
- Una vez actualizados los pesos, se realiza el mismo proceso para el siguiente ejemplo de entrenamiento

3. Una vez acabado el proceso de entrenamiento, se procede a calcular la salida de la red con el conjunto de test:

Para cada ejemplo de prueba
- Damos valor a las neuronas de entrada con el ejemplo de prueba e inicializamos el sesgo (1.0)
- Disparamos los valores de la capa de entrada, inicializamos la capa oculta (y su sesgo) y propagamos la capa de entrada.
- Disparamos los valores de la capa oculta, inicializamos la capa de salida y propagamos la capa oculta.
- Disparamos la capa de salida y ya tendríamos disponibles los valores de salida de la red neuronal.
- Los valores de salida de la red tenemos que pasarlos por una función de test (se le asigna 1.0 al valor más grande de la salida y -1.0 al resto)
- Comparamos los valores obtenidos con la salida esperada y vemos si ha acertado la red neuronal o no.

Finalmente calculamos el porcentaje de aciertos global con el conjunto de entrenamiento.

Apartado 2.1

Tras realizar varios experimentos, hemos concluido que una de las mejores configuraciones para los distintos problemas es la siguiente (modo de lectura 1 con 70% de entrenamiento y 30% de test):

- Constante de aprendizaje = 0.1
- Número de neuronas en la capa oculta: 10
- Número de épocas = 1000

Los experimentos se han realizado con la constante de aprendizaje fija (0.1) y variando el número de épocas(200, 500, 1000) y el número de neuronas en la capa oculta (5, 10, 15, 20). Un total de 12 ejecuciones y teniendo en cuenta el tiempo de ejecución y resultados obtenidos, la mejor configuración es la mencionada anteriormente. A continuación se muestran las curvas de error y porcentaje de aciertos para cada uno de los problemas con los parámetros anteriores

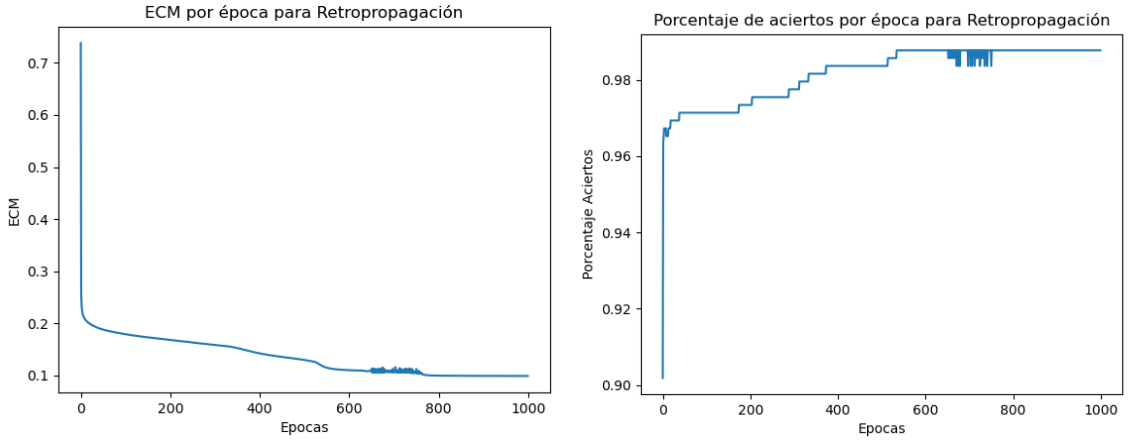

Figura 1. ECM y aciertos para el problema_real1

Porcentaje de aciertos final: **0.9714**

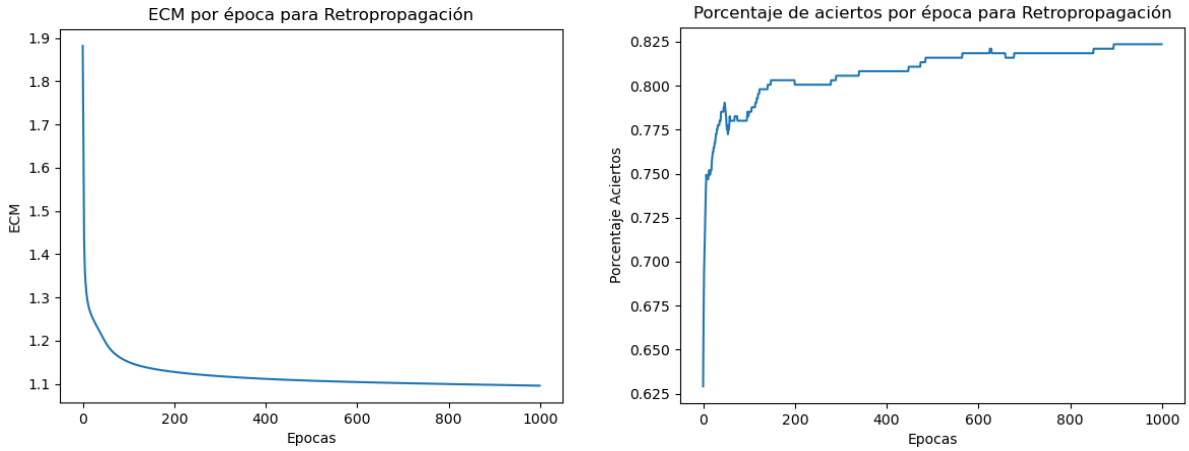

Figura 2. ECM y aciertos para el problema_real2

Porcentaje de aciertos final: **0.7619**

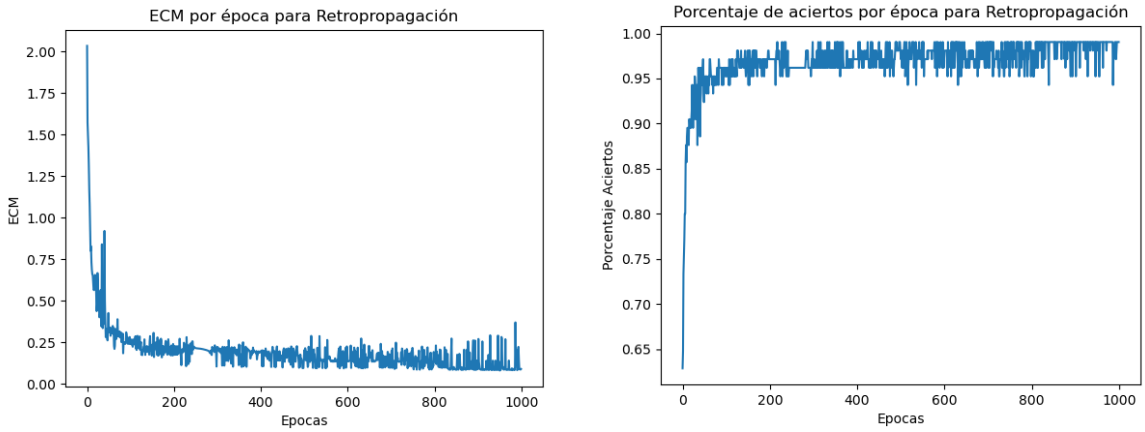

Figura 3. ECM y aciertos para el problema_real3

Porcentaje de aciertos final: **0.9555**

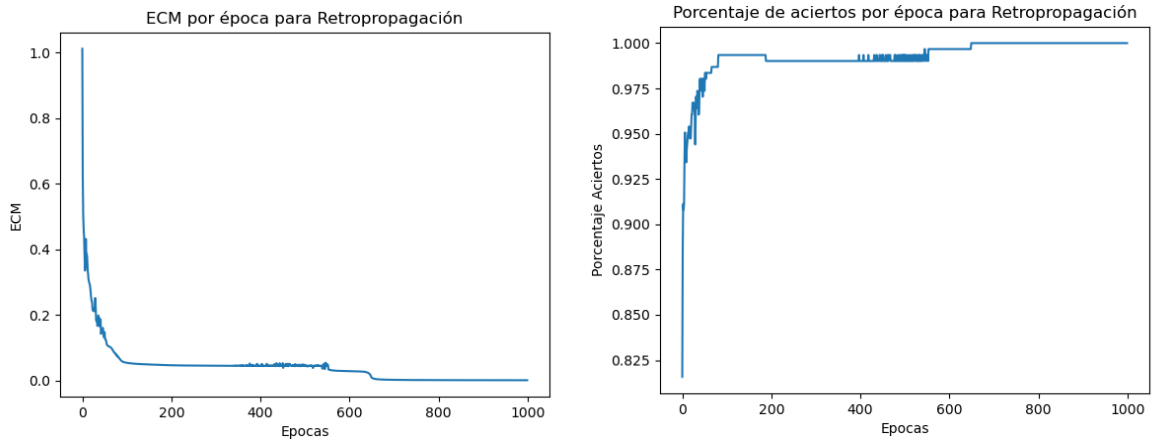

Figura 4. ECM y aciertos para el problema_real5

Porcentaje de aciertos final: **0.9541**

Como se puede observar en las gráficas, todos los problemas tienen tendencias muy parecidas. En las figuras del ECM comienza con un error más alto y a medida que aumentan las épocas disminuye considerablemente hasta quedarse casi constante. Con el porcentaje de aciertos ocurre que en las primeras épocas es menor hasta que llegado un punto va aumentando y se vuelve casi constante también.

El problema que tiene más variaciones es el ***problema_real3***, ya que en las curvas se puede observar que va variando mucho más el ECM y el porcentaje de aciertos. Esto puede deberse a que el número de ejemplos para entrenar es mucho menor que en los demás problemas. El ***problema_real3*** define unos 150 ejemplos y los demás del orden de 500-600 ejemplos. Además de tener 3 salidas distintas, eso puede estar provocando que en épocas distintas varíen los resultados más fácilmente. De todas formas la tendencia de las dos curvas es bastante buena.

Apartado 3.1

Para los problemas 4 y 6 hay que aplicar una normalización de los datos por los rangos de valores entre los distintos atributos. Se puede observar fácilmente que hay algunas columnas con valores muy elevados y diferentes a otras. Por lo tanto hay que proceder a normalizar los datos en estos ejemplos.

Nosotros hemos utilizado el StandardScaler de la librería scikit-learn. Nos permite normalizar de 2 formas distintas: con la media y con la desviación estándar. Simplemente hay que instanciar el StandardScaler y llamar al método fit_transform pasándole el array numpy que contiene los datos a normalizar. Hay que normalizar tanto las entradas de entrenamiento como las de test. Una vez hecho esto, los problemas 4 y 6 pueden solucionarse sin problema.

A continuación se muestran las curvas de ECM y porcentaje de acierto para el problema_real6 con los parámetros indicados en el enunciado:

- Número de neuronas en la capa oculta = 20
- Tasa de aprendizaje = 0.1
- Normalización de atributos
- Porcentaje de entrenamiento = 70%
- 5000 épocas