

Integrantes: Manuel Carrizo, Franco Lehmann y Máximo Pérez.

1. Que la aplicación corra sin presentar errores y que se entregue un documento de respaldo con las explicaciones pertinentes sobre el trabajo realizado. (20 pts)

La aplicación se corre con el comando:

```
python app.py
```

* Running on http://127.0.0.1:5000

* Running on http://10.0.1.81:5000

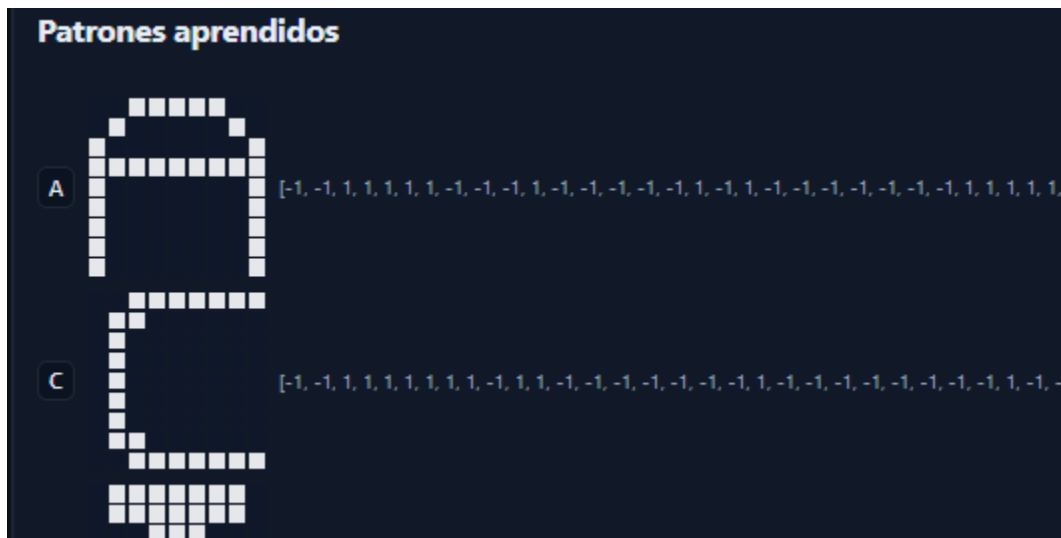
La aplicación va a correr en esas direcciones.

2. Que la aplicación cuente con una interfaz amigable que permita: (25 pts)

- a. Poder consumir imágenes como entrada y no vectores (por supuesto luego la imagen debe ser transformada en vector para que la red la pueda procesar).
- b. Ver las imágenes de entrenamiento y test.
- c. Ver los vectores de entrenamiento y test.
- d. Transformar una imagen en vector y viceversa
- e. El producto correspondiente de cada entrada de test con la matriz W en cada paso.
- f. Valores de la función activación en cada paso.

Respuestas:

- a. Las imágenes que consume como entrada es la matriz que se muestra a la izquierda, que mientras se marcan las posiciones deseadas se va modificando el vector que es lo que se utiliza para procesar el resultado. Luego lo que se predice que es un vector se muestra en otra matriz (interfaz).
- b. Las imágenes de entrenamiento se pueden ver en la sección de patrones aprendidos, en donde están las letras que se aprendió con los vectores correspondientes al lado.



- c. En la imagen anterior se ve que se muestran los vectores de cada letra que se entrenó.

-

- $s = \text{sgn}(h)$
- [-1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1,
-1, 1, 1, -1, -1, -1, -1,
-1, -1, -1, 1, 1, -1, -1,
-1, -1, -1, -1, -1, 1, 1,
-1, -1, -1, -1, -1, -1, -1,
1, 1, -1, -1, -1, -1, -1,
-1, -1, 1, 1, -1, -1, -1, 1,
1, -1, -1, 1, 1, -1, -1, -1,
1, 1, 1, 1, 1, -1, -1, -1,
1, 1, 1, 1, 1, 1, -1]

Ejemplo: Deseamos incluir la letra M, por lo que la etiqueta será "M" y su matriz:



4. Que la aplicación permita “reconocer” instancias totalmente desconocidas (que nunca fueron entrenadas). (15 pts)

La red de Hopfield, al ser autoasociativa, puede recuperar patrones almacenados a partir de entradas incompletas o con ruido.

En el caso de instancias totalmente desconocidas, el modelo no encuentra un mínimo exacto correspondiente, pero converge hacia el patrón más cercano en el espacio de energía, estimando así la clase más probable.

5. Determinar (al menos desde un punto de vista teórico) que sucede cuando los patrones de entrenamiento son muy similares. Se espera que se propongan caminos (o explicaciones) de solución. (15 pts)

Cuando los patrones de entrenamiento son muy parecidos entre sí, la red de Hopfield comienza a mostrar dificultades para almacenar y recuperar información correctamente. Esto ocurre especialmente cuando la matriz es pequeña, como en una red de 4×4 (16 neuronas), donde la capacidad máxima teórica es:

$$\text{Cantidad de patrones} = 0,14 * \text{cant de neuronas}$$

Por lo que con 16 neuronas puede almacenar aproximadamente 2 patrones.

En el caso nuestro queremos guardar al menos 10 patrones (10 letras), por lo que el camino que nosotros propusimos consiste en aumentar el tamaño de la matriz a 9×9 , donde vamos a tener 81 neuronas, por lo que vamos a poder guardar aproximadamente 11 patrones. También elegimos las letras que tengan patrones diferenciados.

Los resultados experimentales mostraron que este cambio mejora notablemente el desempeño: la red puede distinguir los patrones entrenados y tiende a converger hacia alguna de las letras aprendidas. Sin embargo, incluso así, observamos que:

- La red no siempre recupera exactamente el patrón original,
- Requiere un patrón de entrada bastante parecido,
- Aún existe leve distorsión en los estados finales.

Esto se explica porque estamos operando muy cerca del límite de capacidad teórica, donde aparecen mínimos locales no deseados y el proceso de recuperación se vuelve menos estable.

Para comparar, al entrenar la misma red de 81 neuronas con solo 4 patrones, pudimos comprobar que funciona de forma excelente: incluso con ruido significativo en la entrada, la red recupera correctamente la letra aprendida. Esto confirma que el problema no está en la implementación sino en la saturación de la memoria.

En conclusión, cuando los patrones son muy similares o cuando el número de patrones se aproxima al límite de capacidad, la red comienza a mostrar errores en la recuperación. Una matriz más grande (con más neuronas) permitiría almacenar patrones con mayor

independencia y mejorar significativamente el desempeño. Por eso, para un sistema más robusto, sería recomendable utilizar una matriz aún mayor o aplicar técnicas alternativas de preprocesamiento y reducción de similitudes entre patrones.