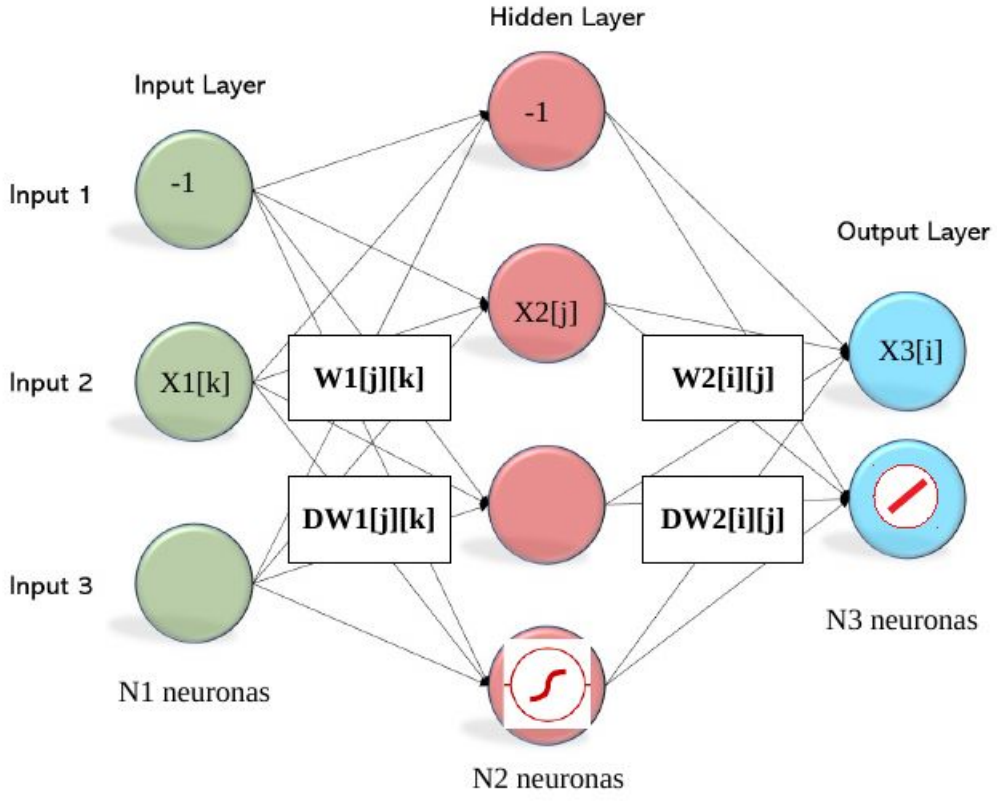
Generalidades:

- El programa bp.c implemente el entrenamiento de una red neuronal de 3 capas (entrada-intermedia-salida) utilizando descenso por el gradiente estocástico con momentum.
- El entrenamiento se realiza un número fijo de épocas. Durante todo el entrenamiento se monitorea el error de validación y se conserva la red correspondiente al número de épocas que produjo el mínimo error de validación. Al terminar el entrenamiento se devuelve dicha red que produjo el mínimo error, junto con las predicciones en el conjunto de test que produce esa red de mínimo error en validación.
- El entrenamiento se realiza minimizando el error cuadrático medio. Este error se estima periódicamente para los tres conjuntos (entrenamiento, validación y test) y se guardan en el archivo .mse
- La estimación de los errores se realiza cada un número fijo de épocas que es introducido por el usuario en el archivo de parámetros. Esto se hace para evitar carga computacional.
- El programa resuelve siempre un problema de regresión (ajustar las salidas lo máximo posible a los valores correctos en el sentido de mínimo error cuadrático). El mismo programa se puede usar para aprender problemas de clasificación binaria (0-1). Por esta razón el código estima también errores de clasificación sobre los conjuntos de entrenamiento, validación y test (en las mismas épocas que estima errores de regresión) y los guarda en el archivo .mse
- Se asume que en los problemas de clasificación las dos clases corresponden a 0-1. Para medir error de clasificación el programa discretiza la salida de la red, considerando como salida 0 a todo valor menor a 0.5, y como salida 1 a todo valor mayor o igual a 0.5
- El código no pregunta si el problema es regresión o clasificación y calcula siempre todos los errores. Cuando nuestro problema es de regresión debemos mirar los errores cuadráticos medios, cuando es de clasificación los errores porcentuales.
- Aunque no lo utilizamos generalmente, el código tiene la posibilidad de continuar un entrenamiento de una red a partir de la red guardada en el disco.
- Para realizar un entrenamiento estocástico los patrones se tienen que presentar a la red ordenados al azar (para evitar “ciclos”). Para hacer eso el código utiliza un index (seq[ ]) de acceso a los datos, que se re-ordena al azar en cada época.

Tipo de red entrenada y nombres de las variables en el código



Como se muestra en la figura, la red tiene 3 capas. En la primera simplemente se repiten los valores del patrón que se quiere propagar, agregando un valor fijo en la primer neurona. La primer capa tiene un número de N1+1 neuronas (N1 neuronas que cargan los N1 inputs y una neurona fija). Lo mismo ocurre con la segunda capa, que tiene N2+1 neuronas. En la segunda capa las neuronas tienen una activación sigmoidea. En la tercer capa se repite el esquema, pero esta capa implementa una activación lineal (como la del perceptrón no acotado).

Los valores que tomas las neuronas de las tres capas se guardan en arrays de números de doble precisión (double) que se llaman X1, X2 y X3. Los valores fijos se colocan en X1[0] y en X2[0].

Las conexiones (pesos) entre la primer y segunda capa se guardan en una matriz de dos dimensiones, llamada W1. Las conexiones entre la segunda y tercera capa se guardan en W2.

Para poder implementar el momentum es necesario guardar las correcciones a cada peso que se realizaron en la época de entrenamiento anterior. Para esto se usan las matrices DW1 y DW2.

Archivo de parámetros

El archivo con los parámetros del entrenamiento tiene la extensión .net y tiene los valores en este orden:

N1: NEURONAS EN CAPA DE ENTRADA
N2: NEURONAS EN CAPA INTERMEDIA
N3: NEURONAS EN CAPA DE SALIDA
PTOT: cantidad TOTAL de patrones en el archivo .data
PR: cantidad de patrones de ENTRENAMIENTO
PTEST: cantidad de patrones de test (archivo .test)
ITER: Total de Iteraciones
ETA: learning rate
u: Momentum
NERROR: graba error cada NERROR iteraciones
WTS: numero de archivo de sinapsis inicial
SEED: semilla para el rand()
CONTROL:verbosity

Comentarios sobre los parámetros:

- N1, N2 y N3 son enteros >=1. N1 coincide con la cantidad de dimensiones de los datos de entrada. N2 regula la complejidad de la red. En general para N3 usamos 1 (1 valor a aprender en regresión o un valor binario en clasificación), aunque el código está preparado para valores mayores.
- PTOT es el total de datos que lee del conjunto .data. De esos valores, PR los usa para entrenar y el resto (PTOT-PR) como conjunto de validación. Esta división en entrenamiento y validación se hace al azar (NO se toman los primeros para entrenar y los últimos para validar a menos que se lo especifique con SEED).
- ETA y u son doubles positivos
- ITER y NERROR son enteros positivos. La cantidad de veces que se grabaran los errores en el archivo .mse (la cantidad de líneas que tendrá el .mse) es ITER/NERROR.
- Los últimos tres parámetros (WTS, SEED y CONTROL) no se deberían cambiar de los valores por defecto.
- WTS es el archivo inicial para los pesos de la red. WTS=0 implica empezar la red con valores al azar. Si se coloca un valor entero >0 el código buscará el archivo .wts con ese valor (por ejemplo 1.wts) y leerá los valores iniciales de cada peso de ese archivo. Esto posibilita continuar entrenamientos cuando se lo desea.
- SEED regula cómo se toma la semilla del rand() y cómo se reparte en validación y train. Esto sirve para continuar un entrenamiento anterior usando un archivo .wts. Para poder usar la misma partición en validación se debe proporcionar aca la misma semilla que se usó en ese entrenamiento. Hay dos valores especiales:
    - SEED: -1: No mezclar los patrones: usar los primeros PR para entrenar y el resto para validar.
    - SEED: 0: Seleccionar semilla con el reloj, y mezclar los patrones.
    - SEED>0: Usa el número como semilla, y mezcla los patrones.
- CONTROL: 0:resumen, 1:0 + pesos, 2:1 + datos