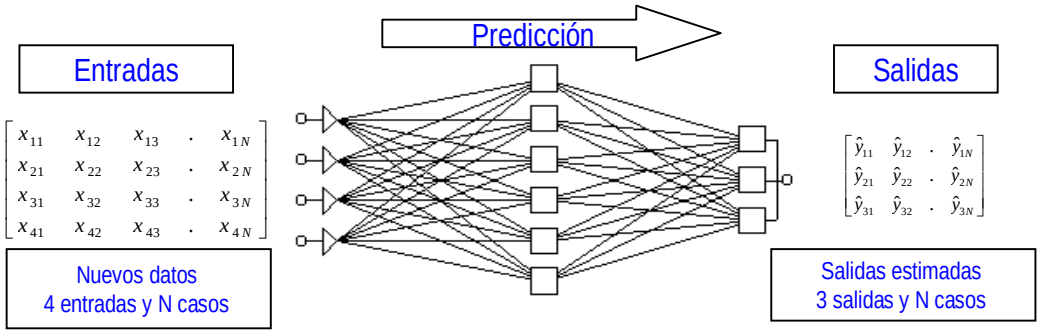
CAPITULO 3

ALGUNOS COMENTARIOS SOBRE GENERALIZACION EN BACKPROPAGATION

En una RN entrenada, si las salidas calculadas por la red con nuevos ejemplos están próximas a los valores deseados, hay generalización (Haykin, 1995).



De este modo, entendamos por generalización, la habilidad de una red neuronal de almacenar en sus pesos sinápticos características que le son comunes a todos los patrones de entrenamiento que fueron usados durante la fase de entrenamiento.

Cuando una nueva entidad es almacenada, las fortalezas de las conexiones existentes representadas por los pesos se modifican suavemente. Si estas modificaciones refuerzan los pesos previamente almacenados entonces ocurre generalización.

Las entradas a la RN durante el entrenamiento deben contener suficiente información perteneciente al dominio. Una buena generalización ocurre cuando los ejemplos de entrenamiento son los suficientemente representativos del dominio del cual forman parte.

El número de casos utilizados para el entrenamiento deben ser suficientes y además representativos del conjunto de casos que se desea generalizar

Si el problema es de clasificación, los ejemplos que formen parte del entrenamiento deben representar todas las categorías. La red puede aprender mejor con una mayor cantidad de ejemplos de todas las categorías.

Si es de aproximación (vale para predicción), se deben incluir ejemplos de valores continuos a través de todo el rango. Es decir, de la nube de ejemplos de entrenamiento, la muestra debe contener ejemplos de los bordes y del centro para un mejor entrenamiento.

Estas dos características principales vigiladas durante el entrenamiento resultan vitales para lograr una buena generalización.

Los patrones generalizados son la suma de todas las modificaciones separadas que han ocurrido. Es decir, en la medida que un patrón de entrenamiento es presentado a la red, ellos pueden generalizar debido al cambio ocurrido en sus pesos.

RECOMENDACIONES PARA EL ENTRENAMIENTO Y GENERALIZACION DE UNA RED EN BACKPROPAGATION

Existen cuatro fases fundamentales en la construcción de modelo de red neuronal. A continuación se mencionarán algunas sugerencias que podrían ser tomadas en cuenta cuando se intente el desarrollo completo de un modelo. Estas sugerencias son válidas desde cualquier aplicación que se emplee como herramienta de desarrollo. Sin embargo, alguna de estas sugerencias son resueltas ya desde algunas aplicaciones. Por ejemplo desde Matlab, Predict, Statistical

NN, SAS NN, pueden hacerse ensayos previos de la arquitectura de la red.

Arquitectura:

Como ya se mencionó, un comienzo de construcción puede ser el de asumir que el número de neuronas en la capa oculta sean menos de la mitad de la suma del número de entradas más el de las salidas.

Si no es suficiente porque el entrenamiento es pobre, incrementar con pocas unidades y proceder a entrenar nuevamente.

Este proceso se puede continuar hasta tanto no se observe mejoras en las pruebas del entrenamiento.

Sin embargo, hay que mantener clara la siguiente premisa: "La mejor red neuronal entrenada no es aquella cuyos error continuamente está bajando". Esto puede conducir a memorización.

Existe una relación donde se puede determinar la cantidad de ejemplos de entrenamiento necesarios para un error deseado y un tamaño de red establecido. Esto es, $N > W/E$, donde N es la cantidad de ejemplos de entrenamiento, W la cantidad de nodos ocultos y E el error deseado.

Entrenamiento:

Después de haber definido una buena arquitectura para la red, en esta fase, se deben atender algunos pasos que son importantes para un buen desempeño del entrenamiento.

El primero de ello es la inicialización de la red. El inicio del entrenamiento requiere de que los pesos tengan un valor inicial. Varios mecanismos existen para la determinación de estos pesos. De

manera aleatoria, por aproximación mediante regresión múltiple son algunos de los métodos empleados. Sin embargo, si el proceso de entrenamiento fue iniciado y deba ser suspendido, no se requiere que la red sea entrenada desde sus comienzos. Los últimos pesos serían los pesos iniciales cuando se desee continuar el entrenamiento con nuevos grupos de datos de entrenamiento.
La constante de aprendizaje juega un importante papel durante la definición de los parámetros para el entrenamiento. Por ejemplo, en Matlab este valor puede flotar en el sentido que en la medida que requiera cambios, el algoritmo automáticamente lo altera a los fines de darle una mayor eficiencia al entrenamiento medido a través de la minimización del error cuadrático. Sin emabargo, se repite la premisa: “La mejor red neuronal entrenada no es aquella cuyos error continuamente está bajando”.

Validación:

Es conveniente añadir ruido a los pesos a través de ejemplos que permitan continuar el proceso cuando se conoce que el entrenamiento ha caído en un mínimo local.

Se debe repetir este proceso hasta conseguir que los parámetros sean satisfechos:
error deseado siguiendo un patrón de caída aceptable,
tiempos de entrenamiento no muy extensos. El tiempo crece exponencialmente de acuerdo a número de entradas.

Guardar al menos el 10% de los datos (entradas y salidas) para hacer pruebas de la eficiencia de la red.

Generalización del modelo:

El mejor desempeño que pueda tener el modelo recién entrenado es que cuando sean presentados ejemplos nunca vistos por la red, pueda clasificarlos o aproximarlos con el mínimo error ajuste.

Un mal ajuste de estos nuevos ejemplos y un entrenamiento supuestamente bueno, implica una memorización de los patrones de entrenamiento.

Por tanto, se debe estar seguro de que no existan contradicciones en los datos y para ello se debe vigilar la preparación de los datos como parte de los recursos para un buen desempeño de una red.