**57**

Capítulo 6

Inicialización de peso

* Pesos aleatorios a distancia
* Redes neuronales entrenadas y no entrenadas
* Nguyen-Widrow Weight Init

Las redes neuronales deben comenzar con sus pesos inicializados a números aleatorios. Estos pesos aleatorios proporcionan a los algoritmos de entrenamiento un punto de partida para los pesos. Si todos los pesos de una red neuronal se establecieron en cero, la red neuronal nunca se entrenaría a un nivel de error aceptable. Esto se debe a que una matriz de peso a cero colocaría la red neuronal en un mínimo local del que nunca puede escapar.

A menudo, los pesos de las redes neuronales simplemente se inicializan con números aleatorios entre un rango específico. El rango -1 a +1 es muy popular. Estos pesos aleatorios cambiarán a medida que la red neuronal esté entrenada para producir salidas aceptables. En la siguiente sección echaremos un vistazo a cómo son las redes neuronales entrenadas y no entrenadas.

Más adelante en este capítulo veremos que estos pesos aleatorios se pueden modificar para ayudar a la red neuronal a entrenar más rápido. El algoritmo de inicialización de peso Nguyen-Widrow es una técnica popular para ajustar estos pesos iniciales. El algoritmo de inicialización de peso Nguyen-Widrow pone los pesos en una posición que es mucho más propicia para el entrenamiento. Esto significa menos iteraciones de entrenamiento para llevar la red neuronal a una tasa de error aceptable.

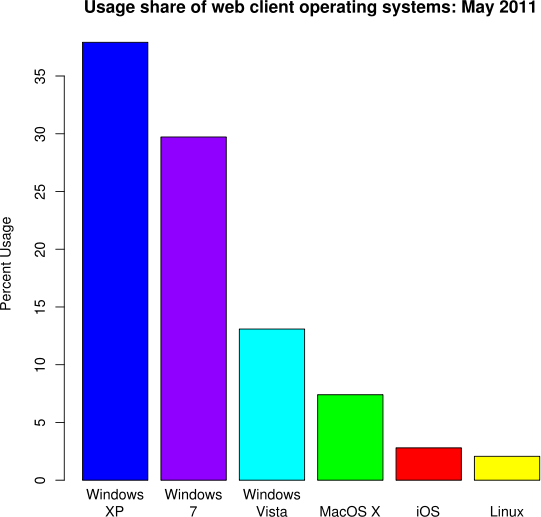
**Inicialización de**  58 pesos

* 1. **Mirando**  **los**  **pesos**

En capítulos anteriores hemos visto los pesos de una red neuronal como una serie de números. Normalmente, no puede echar un vistazo a una matriz de peso y ver ningún tipo de patrón significativo. Sin embargo, si los pesos se representan gráficamente comienzan a surgir patrones.

Una forma común de ver los pesos de una red neuronal es con un tipo especial de gráfico llamado histograma. Un histograma se compone de barras verticales que cuentan el número de ocurrencias en una población. Probablemente has visto histogramas muchas veces antes. Por ejemplo, la Figura 6.1 muestra la popularidad de los sistemas operativos.

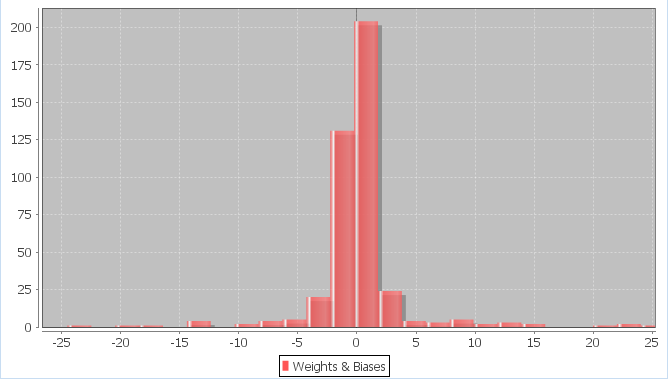
**Figura 6.1:** Histograma de popularidad del sistema operativo (de Wikipedia)



El eje y muestra el número de apariciones de cada uno de los grupos en el eje y. Podemos usar un histograma para ver los pesos de una red neuronal. Por lo general, puede distinguir a una red neuronal no entrenada mirando este histograma. Figura 6.2 shows una red neuronal entrenada.

### Aleatorización adistancia 59

**Figura 6.2:** Una red neuronal entrenada



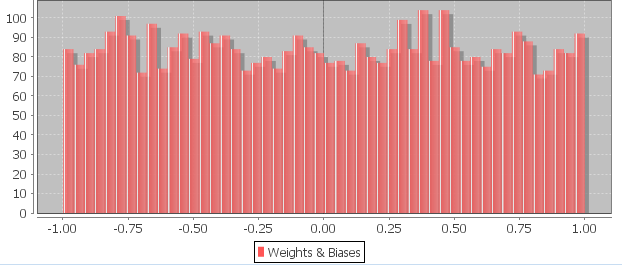
Un histograma de red neuronal es el mismo que el concepto que el histograma del sistema operativo mostrado anteriormente. El eje y especifica cuántos pesos cayeron en los rangos especificados por los números del eje x. Esto le permite ver la distribución de los pesos.

Las redes neuronales entrenadas por Mo st se verán algo así como el gráfico anterior. Sus pesos estarán muy apretados alrededor de cero. Una red neuronal entrenada normalmente se verá como una curva gaussiana estrecha varían.

# Aleatorización a distancia

En la última sección vimos cómo es una red neuronal entrenada en un histograma de peso. Las redes neuronales no entrenadas pueden tener una variedad de apariencias. Cómo se verá el histograma de peso estará determinado por el método de inicialización de peso utilizado.

**Figura 6.3:** Una aleatorización a distancia



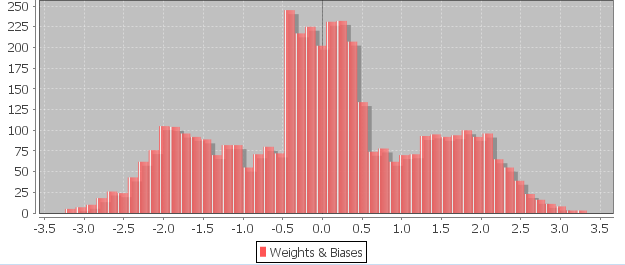
La aleatorización de rango produce un gráfico de aspecto muy simple. Cuantos más pesos haya, más plana será la parte superior. Esto se debe a que el generador de números aleatorios debe estar dando

una distribución uniforme de los números. Si usted está aleatorizando al rango de -1 a 1, usted esperaría tener aproximadamente el mismo número de pesos por encima de cero como abajo.

# Uso de Nguyen-Widrow

Ahora veremos el método de inicialización de peso Nguyen-Widrow. El método Nguyen-Widrow comienza al igual que el método aleatorio de rango. Los valores aleatorios se eligen en -0.5 y +0.5. Sin embargo, se emplea un algoritmo especial para modificar las ponderaciones. El histograma de una initialización de peso Nguyen-Widrow se parece a la Figura 6.4.

**Figura 6.4:** La inicialización de Nguyen-Widrow



Como se puede ver, la inicialización De Nguyen-Widrow tiene un patrón muy distintivo. Hay una gran distribución de pesos entre -0.5 y 0.5. Sin embargo, luego sube gradualmente y luego cae rápidamente a alrededor de -3.0 y +3.0.

## Actuación de Nguyen-Widrow

Tal vez te estés preguntando cuánta ventaja puede tener Nguyen-Widrow. el a continuación muestra el número medio de iteraciones de entrenamiento necesarias para entrenar una red neuronal Inicializado Por Gama Aleatorio Y Nguyen-Widrow.

Promedio i t e r a t i o n s ( inferior I s b e T te r ) Rango Aleatorio : 502 . 86

Nguyen*−*Widrow : 454 . 88

Como se puede ver en la información anterior, el Nguyen-Widrow supera al aleatorizador de rango.

## Implementación de Nguyen-Widrow

Esta técnica fue inventada por Derrick Nguyen y Bernard Widrow. Se introdujo por primera vez en su artículo "Mejorar la velocidad de aprendizaje de las redes neuronales de 2 capas eligiendo los valores iniciales de los pesos adaptativos" en proceedings of the International Joint Conference on Neural Networks, 3:21-26, 1990.

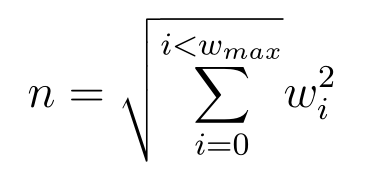
Para implementar un ión aleatorizat Nguyen-Widrowprimero inicialice la red neuronal con valores de peso aleatorios en el rango -0.5 a +0.5. Esta es exactamente la misma técnica que se describió anteriormente en este artículo para los números aleatorios a distancia.

La técnica de aleatorización Nguyen-Widrow es eficiente sin embargo asigna cada neurona oculta a una gama del problema. Para ello debemos mapear las neuronas de entrada a las neuronas ocultas. Calculamos un valor, llamado beta, que establece estos rangos. Puede ver el cálculo de beta en la ecuación 6.1.

*β* = 0*.*7*h i* (6.1)

1

La variable **h**  representa el número de neuronas ocultas en la primera capa oculta, mientras que la variable **i** representa el número de neuronas de entrada. Calcularemos los pesos que toman cada neurona oculta de uno en uno. Para cada neurona oculta calculamos la norma euclidiana para todas las entradas a la neurona oculta actual. Esto se hace con la ecuación 6.2.



(6.2)

Beta seguirá siendo la misma para cada neurona oculta. Sin embargo, la norma debe recalcularse para cada neurona oculta. Una vez calculados los valores beta y norma, el pesos se pueden ajustar. La siguiente ecuación muestra cómo se ajustan los pesos usando el Previamente Calculado Valores.

*wt*+1

= *βwt*

*N*

(6.3)

Todos los pesos entrantes a la neurona oculta actual se están ajustando usando la misma norma. Este mismo proceso se repite para cada neurona oculta.

Es posible que haya notado que solo estamos especificando cómo calcular las ponderaciones entre la capa de entrada y la primera capa oculta. El método Nguyen-Widrow no especifica

cómo calcular las ponderaciones entre una capa oculta y la salida. Del mismo modo, el método Nguyen- Widrow no especifica cómo calcular las ponderaciones entre varias capas ocultas. Todos los pesos fuera de la primera capa y la primera capa oculta están simplemente inicializando a un valor entre -0.5 y +0.5.

## Nguyen-Widrow en acción

Ahora caminaremos a través de la inicialización de peso aleatorio para una pequeña red neuronal. Veremos una red neuronal que tiene dos neuronas de entrada y una sola neurona de salida. Hay una capa oculta single con dos neuronas. Las neuronas de sesgo están presentes en la entrada y capas ocultas.

Comenzamos inicializando los pesos a números aleatorios en el rango -0.5 a +0.5. el Comenzando Pesos Son Mostrado Aquí.

Peso 0 : H1*−>*O1 :

Peso 1 : H2*−>*O1 :

Peso 2 : B2*−>*O1 :

Peso 3 : I1*−>*H1 :

Peso 4 : I2*−>*H1 :

0 . 23773012320107711

0 . 2200753094723884

0 . 121696910737914

0 . 5172524211645029

Peso 5 : B1*−>*H1 :

*−*0.5258712726818855

Peso 6 : I1*−>*H2 :

0 . 8891383322123643

Peso 7 : I2*−>*H2 :

*−*0.007687742622070948

Peso 8 : B1*−>*H2 :

*−*0.6610227585583137

*−*0.48985643968339754

Primero debemos calcular beta, que se da en la ecuación 6.1. Este cálculo se muestra aquí.

Beta = 0 . 7 ∗ (Neuronas ocultaciones ˆ ( 1 . 0 / inputCount )

Relleno En el Variables Nosotros have.

Beta = 0 . 7 ∗ ( 2 . 0 ˆ ( 1 . 0 / 2 . 0 ) ) = 0 . 9899

Ahora estamos listos para modificar los pesos. Sólo modificaremos los pesos de tres a ocho. Los pesos de cero a dos no están cubiertos por el algoritmo Nguyen-Widrow, y simplemente se establecen en valores aleatorios entre -0.5 y 0.5.

Los pesos se recalcularán en dos fases. Primero recalcularemos todos los pesos, desde el sesgo y las neuronas de entrada hasta las neuronas ocultas. Para ello debemos calcular la norma euclidiana, o magnitud, para la neurona oculta. De la ecuación 6.2, tenemos lo siguiente.

Norma Oculto 1 = s Q R T ( ( peso 3 ) ˆ2 + ( peso 4 ) ˆ2 + ( peso 5 ) ˆ 2 )

Relleno En Pesos tenemos el Siguientes.

Norma Oculta 1 = s Q R T ( ( 0 . 5 1 7 2 ˆ2 ) + (  *−*0 .5258 ) ˆ2 + ( 0 . 8 8 9 1 ˆ2 ) ) = 1 . 155

Ahora veremos cómo calcular el primer peso.

Nuevo peso =( Beta ∗ peso Antiguo ) /Norma oculta 1

Relleno En Valores Nosotros have.

Nuevo peso =( 0 . 9899 ∗ 0 . 5 1 7 2 ) / 1 . 155 = 0 . 4432

Los tres pesos que alimentan oculta la primera neurona se transforman de esta manera.

0 . 5172524211645029  *−>* 0 . 44323151482681195 ( como J u s T Visto )

0 . 8891383322123643  *−>* 0 . 7618990530577658

*−*0.5258712726818855  *−*

*> −*0.4506169739523903

Ahora repetimos el mismo proceso para la neurona dos. El valor de beta sigue siendo el mismo. Sin embargo la magnitud debe ser recalculada. La magnitud recalculada para la neurona dos se da Aquí.

Neurona Dos Magnitud = 0 . 8227815750355376

Usando esto ahora podemos recalcular pesos de seis a ocho.

*−*0.6610227585583137  *−>*  *−*0.79532547274779

*−*0.48985643968339754  *−>*  *−*0.5893825884595142

*−*0.007687742622070948  *−>*  *−*0.009249692928269318

éste Resultados En el final Valores Para todo De el Pesos.

Peso 0 : H1*−>*O1 :

Peso 1 : H2*−>*O1 :

Peso 2 : B2*−>*O1 :

Peso 3 : I1*−>*H1 :

Peso 4 : I2*−>*H1 :

0 . 23773012320107711

0 . 2200753094723884

0 . 121696910737914

0 . 44323151482681195

Peso 5 : B1*−>*H1 :

*−*0.4506169739523903

Peso 6 : I1*−>*H2 :

0 . 7618990530577658

Peso 7 : I2*−>*H2 :

*−*0.009249692928269318

Peso 8 : B1*−>*H2 :

*−*0.79532547274779

*−*0.5893825884595142

Estos pesos, aunque todavía aleatorios, mejor utilizar las neuronas ocultas. Estos son los pesos con los que empezaremos a entrenar.

# Resumen del capítulo

En este capítulo vimos cómo se inicializan los pesos de una red neuronal. Estos pesos deben establecerse en valores aleatorios. Si todos los pesos se establecieran en cero, la red neuronal nunca entrenaría correctamente. Hay varias maneras diferentes de que los pesos de una red neuronal se inicializan comúnmente.

La primera es la aleatorización de rangos. Este proceso inicializa cada peso de la red neuronal- trabajar a un valor aleatorio en un rango específico. El rango utilizado es típicamente -1 a 1 o 0 para

1. Aunque la aleatorización de rango puede proporcionar muy buenos resultados para el entrenamiento de la red neuronal, hay mejores métodos.

La inicialización del peso de Nguyen-Widrow puede proporcionar tiempos de entrenamiento significativamente mejores que la aleatorización simple del rango. La técnica de aleatorización Nguyen-Widrow es eficiente, ya que asigna cada neurona oculta a una gama del problema.

Hasta ahora, todo el training presentado se ha centrado en intentar minimizar el error. En el siguiente capítulo veremos el algoritmo de entrenamiento de Levenberg Marquardt (LMA). LMA a menudo puede entrenar en menos iteraciones que la propagación resistente (RPROP).

### 6.4 Resumen del capítulo 65