**CHAPTER** **9:** **PREDICTIVE** **NEURAL** **NETWORKS**

* Creación de neuronas de entrada y salida para predicción
* Cómo crear conjuntos de entrenamiento para una red neuronal predictiva
* Predicción de la ola sinusoidal

Las redes neuronales son particularmente buenas para reconocer patrones. El reconocimiento de patrones se puede utilizar para predecir patrones futuros en los datos. Una red neuronal utilizada para predecir patrones futuros se llama red neuronal predictiva o temporal. Una red neuronal predictiva se puede utilizar para predecir eventos futuros, como tendencias del mercado de valores y ciclos de manchas solares.

Este capítulo introducirá redes neuronales predictivas a través de un ejemplo de trabajo neto codificado para predecir la onda sinusoidal. El siguiente capítulo presentará una red neuronal que intenta predecir el S&P 500.

# Cómo predecir con una red neuronal

Muchos tipos diferentes de redes neuronales se pueden utilizar para la predicción. Este libro utilizará la red neuronal feedforward para intentar aprender patrones en los datos y predecir valores futuros. Al igual que todos los problemas aplicados a las redes neuronales, la predicción es una cuestión de determinar inteligentemente cómo configurar las neuronas de entrada e interpretación de salida para un problema.

Hay muchas maneras de modelar problemas de predicción. Este libro se centrará en una técnica específica, que consiste en tomar datos conocidos, particionarlo en conjuntos de entrenamiento y aplicarlo a las neuronas de entrada. Un número menor de neuronas de salida luego repre envió los datos futuros. A continuación, se muestra una discusión sobre cómo configurar las neuronas de entrada y salida para una predicción simple.

## Configuración de neuronas de entrada y salida para predicción

Considere una serie simple de números, como la secuencia que se muestra aquí: 1, 2, 3, 4, 3, 2, 1, 2, 3, 4, 3, 2, 1

Una red neuronal que predice números de esta secuencia podría usar tres neuronas de entrada y una sola neurona de salida. Por ejemplo, un conjunto de entrenamiento podría parecerse a la Tabla 9.1.

### Tabla 9.1: Conjuntos de entrenamiento de muestra para una red neuronal predictiva

|  |  |  |
| --- | --- | --- |
| **set** | **Input (Porciones de tiempo)** | Ideal Output |
| 1 | 1,2,3 | 4 |
| 2 | 2,3,4 | 3 |
| 3 | 3,4,3 | 2 |
| 4 | 4,3,2 | 1 |

Como puede ver, la red neuronal está preparada para recibir varias muestras de datos en una secuencia. La neurona de salida predice entonces cómo se continuará la secuencia. La idea es que ahora se puede alimentar cualquier secuencia de tres números, y el trabajo de la red neuronal predecirá el cuarto número. Cada punto de datos se denomina sector de tiempo. Por lo tanto, cada neurona de entrada representa una rebanada de tiempo conocida. Las neuronas de salida representan futuras rebanadas de tiempo.

Por supuesto, este es un ejemplo muy simple. Es posible incluir datos adicionales que puedan influir en el siguiente valor. En el capítulo 10, intentaremos predecir el S&P 500 e incluiremos la tasa de interés actual.

## Selección de datos para predicción

Hay varios métodos para seleccionar datos de entrada para la predicción. En el ejemplo anterior, cada valor del conjunto de entrenamiento se produjo directamente junto al siguiente. Esto no siempre es posible. La cantidad de datos de entrada puede producir demasiados conjuntos de entrenamiento. Este será el caso de la red neuronal presentada en el próximo capítulo, ya que tendremos fácil acceso a la información financiera del S&P 500 que se remonta a 1950, así como a las tasas de interés preferencia. Serán más datos de los que serán prácticos trabajar para entrenar la red neuronal.

En el ejemplo presentado en la tabla anterior, los valores reales se introducen simplemente en la red neuronal. Para un ejemplo simple, como la onda sinusoidal, esto funciona muy bien. Sin embargo, es posible que desee alimentar el porcentaje de red neuronal aumenta y hacer que las neuronas de salida predigan aumentos o disminuciones porcentuales futuros. Este tecnique se ampliará en el capítulo 10 cuando se aplique una red neuronal predictiva al S&P 500.

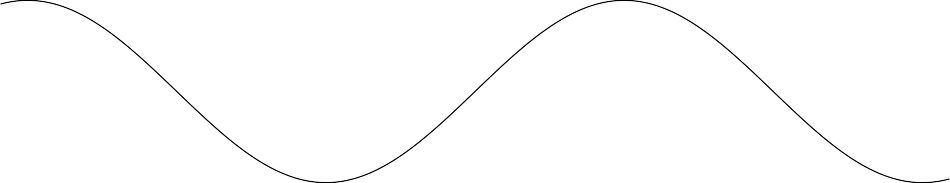
Otra consideración para los datos del conjunto de entrenamiento es dejar suficientes datos para poder evaluar el rendimiento de la red neuronal predictiva. Por ejemplo, si el objetivo es entrenar una red neuronal para la predicción del mercado de valores, tal vez sólo los datos anteriores a 2005 deben utilizarse para entrenar la red neuronal. Todo lo que pasó en 2005 se puede utilizar para evaluar el rendimiento de la red neuronal entrenada.

# Predicción de la ola sinusoidal

El ejemplo de este capítulo es relativamente simple. Se presenta una red neuronal que predice la onda sinusoidal. La onda sinusal es matemáticamente predecible, por lo que en muchos sentidos no es un buen ejemplo para ilustrar una red neuronal predictiva; sin embargo, la onda sinusal se entiende fácilmente y varía con el tiempo. Esto hace que sea un buen introducción a las redes neuronales predictivas. El capítulo 10 se expandirá sobre esto y utilizará una red neuronal para tratar de proporcionar una idea de la predicción del mercado de valores.

La onda sinusoidal se puede ver trazando la función sinusoidal trigonométrica. La Figura 9.1 muestra la onda sinusoidal.

### Figura 9.1: La onda sinusoidal.



3.2

2.4

1.6

0.8

-4.8

-4

-3.2

-2.4

-1.6

-0.8

0

0.8

1.6

2.4

3.2

4

4.8

-0.8

-1.6

-2.4

-3.2

La función de onda sinusoidal se utiliza para comenzar a entrenar la red neuronal. El algoritmo backpropagation se utiliza para entrenar para el predictor de ondas sinusoidales. Cuando se ejecute por primera vez el ejemplo de onda sinusoidal, verá los resultados del proceso de entrenamiento. La lista 9.1 muestra la salida típica del proceso de entrenamiento del predictor de ondas sinusoidales.

### Listado 9.1: Training the Sine Wave Predictor

Iteración #1 Error:0.48120350975475823 Iteración #2 Er- ror:0.3675344576855236 Iteración #3 Error:0.3212066601426759

Iteración #4 Error:0.2952410514715732 Iteración #5 Er- ror:0.2780102928778258 Iteración #6 Error:0.26556861969786527

Iteración #7 Error:0.25605359706505776 Iteración #8 Er-

ror:0.24842242500053566 Iteración #9 Error:0.24204767544134156 It-

eración #10 Error:0.23653845782593882

...

Iteración #4990 Error:0.0231939762897425 Iteration #4991 Er- ror:0.0231931093486356 Iteración #4992 Error:0.023192222246688515

Iteración #4993 Error:0.02319137532183077 Iteración #4994 Er- ror:0.023190508573672858 Iteración #4995 Error:0.02318964200159761

Iteración #4996 Error:0.0231877560498862 Ración ite#4997 Er- ror:0.02318790938322986 Iteración #4998 Error:0.023187043335705867 Iteración #4999 Error:0.023186177461801745

Al principio, la tasa de error es bastante alta en 48%. Esto rápidamente comienza a caer al 36,7% en la segunda iteración. En el momento en que se ha producido la iteración número 4.999, la tasa error ha caído al 2,3%. El programa está diseñado para detenerse antes de llegar a la iteración número 5.000. Esto logra reducir la tasa de error a menos del 3%.

La capacitación adicional produciría una mejor tasa de error; sin embargo, al limitar las iteraciones, el programa es capaz de terminar en sólo unos minutos en un ordenador normal. Este programa tardó unos dos minutos en ejecutarse en un Intel Core2 Dual 2mghtz com- puter.

Una vez completado el entrenamiento, la onda sinusoidal se presenta a la red neuronal para su predicción. La salida de esta predicción se muestra en el listado 9.2.

### Listado 9.2: Predicción de la ola sinusoidal

5:Actual=0.76604:Predicted=0.7892166200864351:Difference=2.32% 6:A ctual=0.86602:Predicted=0.8839210963512845:Diferencia=1.79% 7:Ac tual=0.93969:Predicted=0.934526031234053:Difference=0.52% 8:Act ual=0.9848:Predicted=0.9559577688326862:Difference=2.88% 9:Actu al=1.0:Predicted=0.961556601973113:Difference=3.84% 10:Actual= 0.9848:Predicted=0.9558060932656686:Difference=2.90% 11:Actual= 0.93969:Predicted=0.9354447787244102:Difference=0.42% 12:Real

=0.86602:Predicted=0.8894014978439005:Difference=2.34% 13:Actua l=0.76604:Predicted=0.801342405700056:Difference=3.53% 14:Actua l=0.64278:Predicted=0.6633506809125252:Difference=2.06% 15:Actu al=0.49999:Predicted=0.4910483600917853:Difference=0.89% 16:Act ual=0.34202:Predicted=0.31286152780645105:Difference=2.92% 17:A ctual=0.17364:Predicted=0.14608325263568134:Difference=2.76% 18:Actual=0.0:Predicted=-0.008360016796238434:Difference=0.84% 19:Actual=-0.17364:Predicted=-0.15575381460132823:Difference=1.79% 20:Actual=-0.34202:Predicted=-0.3021775158559559:Difference=3.98%

...

490:Actual=-0.64278:Predicted=-0.6515076637590029:Diferencia=0.87% 491:Actual=-0.76604:Predicted=-0.813333939237001:Difference=4.73%492:Actual=-0.86602:Predicted=-0.9076496572125671:Difference=4.16% 493:Actual=-0.93969:Predicted=-0.9492579517460149:Difference=0.96%

494:Actual=-0.9848:Predicted=-0.9644567437192423:Difference=2.03% 495:Actual=-1.0:Predicted=-0.9664801515670861:Diferencia=3.35% 496:Actual=-0.9848:Predicted=-0.9579489752650393:Difference=2.69% 497:Actual=-0.93969:Predied=-0.9340105440194074:Diferencia=0.57% 498:Actual=-0.86602:Predicted=-0.8829925066754494:Diferencia=1.70% 499:Actual=-0.76604:Predicted=-0.7913823031308845:Diferencia=2.53%

Como puede ver, se muestran los valores reales y predichos para cada elemento. La red neuronal sólo fue entrenada para los primeros 250 elementos; sin embargo, el trabajo de la red neuronal es capaz de predecir mucho más allá de estos primeros 250. También observará que el diferencia encaje entre los valores reales y los valores predichos rara vez supera el 3%.

**Obtención de datos de ondas** **sinusoidales**

Se proporciona una clase denominada **ActualData** que mantendrá los datos "calculados" reales para la onda sinusoidal. La clase **ActualData** también proporciona métodos de conveniencia que se pueden usar para construir los conjuntos de entrenamiento. La clase **ActualData** se muestra en el listado 9.3.

### Listado 9.3: Datos de onda sinusoidal reales (Datos reales.java)

paquete com.heatonresearch.book.introneuralnet.ch9. predecir;

/\*\*

* Capítulo 9: Redes neuronales predictivas

\*

* RealData: Contiene valores de la onda sinusoidal.

\*

* @author Jeff Heaton
* @version 2,1

\*/

clase pública RealData {

doble sinDEG estático público( doble deg final) { double rad final = deg \* (Math.PI / 180); doble resultado final = Math.sin(rad);

retorno ((int) (resultado \* 100000.0)) / 100000.0;

}

final privado doble real[]; entrada final privadaSize ;

entrada final privadaSize;

RealDatapúblico(tamañofinal int, entrada finalSize, final int outputSize) {

this.actual = nuevo doble[tamaño]; this.inputSize = inputSize; this.outputSize = outputSize;

ángulo int = 0;

para (int i = 0; i < this.real.length; i++) { this.real[i] = sinDEG(ángulo);

ángulo += 10;

}

}

público void getInputData(desplazamientofinal int, final double target[]) {

para (int i = 0; i < this.inputSize; i++) { target[i] = this.real[offset + i];

}

}

público void getOutputData(desplazamiento final int, final double target[]) {

para (int i = 0; i < this.outputSize; i++) { target[i]= this.real[offset + this.inputSize + i];

}

}

}

El constructor de la clase **ActualData** inicializa las variables internas en

los valores de la onda sinusoidal. La firma del constructor para el **ActualData**

clase se muestra aquí:

Public ActualData(tamañofinal int, entrada finalSize,

salida final intSize)

Tres argumentos se pasan al **actualData** constructor. El **tamaño** argumenta- ment define cuántos valores de la onda sinusoidal serán considerados. El argumento **inputSize** especifica el número de neuronas de entrada. El **outputSize** argument speci- fies el número de neuronas de salida.

En primer lugar, se asigna una matriz **doble** que es lo suficientemente grande como para hold todos los valores de onda sinusoidal solicitados. A continuación, los argumentos se copian en variables de instancia.

this.actual = nuevo doble[tamaño]; this.inputSize = inputSize; this.outputSize = outputSize;

Los datos de onda sinusoidal comienzan en el ángulo cero.

ángulo int = 0;

A continuación, recorremos el número de valores solicitados.

para (int i = 0; i < this.real.length; i++) {

Para cada valor solicitado, el seno se calcula utilizando un ángulo expresado en de- grees.

this.real[i] = sinDEG(ángulo); ángulo += 10;

}

El bucle continúa con un incremento de ángulo de 10 grados.

## Construcción de conjuntos de entrenamiento para la ola sinusoidal

Como recordará de capítulos anteriores, puede proporcionar conjuntos de entrenamiento para entrenar la red neuronal feedforward. Estos conjuntos de entrenamiento constan de dos matrices. El primer ar- ray especifica los valores de entrada para la red neuronal. La segunda matriz especifica las salidas ideales para cada uno de los valores de entrada. La clase **ActualData** proporciona dos métodos para recuperar cada una de estas matrices. El primer método, denominado **getInputData**, se muestra aquí:

valor void público getInputData(desplazamientofinal int, objetivo doble final[])

La variable **offset** especifica el desplazamiento en los datos reales en los que se copian las ginebras. La matriz de **destino** recibirá los valores reales. La meth- od **getInputData** simplemente continúa copiando los valores adecuados en la matriz de **destino.**

para (int i = 0; i < this.inputSize; i++) { target[i] = this.real[offset + i];

}

El método **getOutputData** recupera los valores ideales con los que la neural

red será entrenado. La firma para el método **getOutputData** se muestra aquí:

público void getOutputData(desplazamiento final int, objetivo doble final[])

Como se ha mencionado anteriormente, la variable **de desfase** especifica el desplazamiento en los datos reales en los que se va a empezar a copiar valores ideales. La matriz **de destino** recibirá los valores ideales. El método **getInputData** simplemente continúa copiando los valores adecuados en la matriz de **destino.**

para (int i = 0; i < this.outputSize; i++) {

objetivo[i] = this.real[offset + this.inputSize + i];

}

El bucle anterior rellenará la matriz de destino con los valores reales. Tabla 9.2

muestra estos valores.

### Tabla 9.2: Datos de entrenamiento sine wave

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Entrada 1** | **Entrada 2** | **Entrada 3** | **Entrada 4** | **Entrada 5** | **Salida 1** |
| 0.0 | 0.17364 | 0.34202 | 0.49999 | 0.64278 | 0.76604 |
| 0.17364 | 0.34202 | 0.49999 | 0.64278 | 0.76604 | 0.86602 |
| 0.34202 | 0.49999 | 0.64278 | 0.76604 | 0.86602 | 0.93969 |
| 0.49999 | 0.64278 | 0.76604 | 0.86602 | 0.93969 | 0.9848 |
| 0.64278 | 0.76604 | 0.86602 | 0.93969 | 0.9848 | 1.0 |
| 0.76604 | 0.86602 | 0.93969 | 0.9848 | 1.0 | 0.9848 |
| 0.86602 | 0.93969 | 0.9848 | 1.0 | 0.9848 | 0.93969 |
| 0.93969 | 0.9848 | 1.0 | 0.9848 | 0.93969 | 0.86602 |
| 0.9848 | 1.0 | 0.9848 | 0.93969 | 0.86602 | 0.76604 |
| 1.0 | 0.9848 | 0.93969 | 0.86602 | 0.76604 | 0.64278 |
| 0.9848 | 0.93969 | 0.86602 | 0.76604 | 0.64278 | 0.49999 |
| 0.93969 | 0.86602 | 0.76604 | 0.64278 | 0.49999 | 0.34202 |
| 0.86602 | 0.76604 | 0.64278 | 0.49999 | 0.34202 | 0.17364 |
| 0.76604 | 0.64278 | 0.49999 | 0.34202 | 0.17364 | 0.0 |
| 0.64278 | 0.49999 | 0.34202 | 0.17364 | 0.0 | -0.17364 |
| 0.49999 | 0.34202 | 0.17364 | 0.0 | -0.17364 | -0.34202 |
| 0.34202 | 0.17364 | 0.0 | -0.17364 | -0.34202 | -0.5 |
| 0.17364 | 0.0 | -0.17364 | -0.34202 | -0.5 | -0.64278 |
| 0.0 | -0.17364 | -0.34202 | -0.5 | -0.64278 | -0.76604 |
| -0.17364 | -0.34202 | -0.5 | -0.64278 | -0.76604 | -0.86602 |
| -0.34202 | -0.5 | -0.64278 | -0.76604 | -0.86602 | -0.93969 |
| -0.5 | -0.64278 | -0.76604 | -0.86602 | -0.93969 | -0.9848 |
| -0.64278 | -0.76604 | -0.86602 | -0.93969 | -0.9848 | -1.0 |
| -0.76604 | -0.86602 | -0.93969 | -0.9848 | -1.0 | -0.9848 |
| -0.86602 | -0.93969 | -0.9848 | -1.0 | -0.9848 | -0.93969 |
| -0.93969 | -0.9848 | -1.0 | -0.9848 | -0.93969 | -0.86602 |
| -0.9848 | -1.0 | -0.9848 | -0.93969 | -0.86602 | -0.76604 |

Estos valores de entrenamiento se utilizarán en la siguiente sección para entrenar la red neuronal.

## Entrenamiento del Predictor de Ondas Sinusoidales

El predictor de ondas sinusoidales utiliza una función de activación tangente hiperbólica, en lugar de la función de activación sigmoide que muchas de las redes neuronales de este libro han utilizado. Esto se debe a que la función sinusoidales devuelve números entre -1 y 1. Esto se puede ver en la Tabla 9.2. La función sigmoide sólo puede manejar números entre 0 y 1, y nos fallaría cuando se nos presentan los valores de la función sinusoidal.

Como se mencionó anteriormente, la red neuronal se entrena con una contrapropagación al-gorithm. La contrapropagación fue cubierta en el capítulo 5, "Feedforward Backpropagation Neural Networks". Cuando se utiliza la tangente hiperbólica como función de activación, es importante utilizar una baja tasa de aprendizaje e impulso. De lo contrario, los ajustes serán demasiado grandes y la red puede no converger en una tasa de error aceptable.

El entrenamiento se produce en el método **trainNetworkBackprop.** La firma para **trainNetworkBackprop** se muestra aquí:

tren vacío privadoNetworkBackprop()

Este método comienza creando un objeto **Backpropagation** para entrenar el trabajo de red. Se utiliza una tasa de aprendizaje de 0,001 y un impulso de 0,1. Estos son lo suficientemente pequeños como para entrenar correctamente esta red neuronal.

tren final = nueva backpropagation (this.network, this.input, this.ideal, 0.001, 0.1);

Se crea una variable local denominada **época** para contar el número de épocas de entrenamiento.

int epoch = 1;

Se introduce el bucle y, a continuación, para cada época de entrenamiento se llama al método **de iteración** en el objeto **train.**

hacer { tren.iteration();

System.out.println("Iteración #" + epoch + " Error:"

+ train.getError()); epoch++;

} mientras ((epoch < 5000) && (train.getError() > 0.01));

}

Esto continúa hasta que hayan pasado 5.000 épocas, o la tasa de error sea inferior a

1%.

## Predecir el futuro de la ola sinusoidal

Para ver cuán eficaz es la red neuronal para predecir el futuro de la onda sinusoidal, se debe llamar al método **de visualización.** La firma para el método **de visualización** se muestra aquí:

pantalla de vacío privado() {

Los números mostrados de la onda sinusoidal se redondearán a dos decimales.

Usamos la clase Java **NumberFormat** para lograr esto.

numberformat final percentFormat = NumberFormat. getPercentInstance();

percentFormat.setMinimumFractionDigits(2);

Es necesario crear matrices de entrada y salida para contener la entrada a la red neuronal, así como su salida.

entrada doble final[] = nuevo doble[SinWave.INPUT\_SIZE]; doble salida final[] = nuevo doble[SinWave.OUTPUT\_SIZE];

A continuación, recorremos todos los datos reales. La red neuronal no fue entrenada en todos los datos reales, por lo que parte de esto será predicción.

para (int i = SinWave.INPUT\_SIZE; i < SinWave.ACTUAL\_SIZE; i++) {

Obtenemos los datos de entrada y salida de la matriz de datos real. Los datos de salida son los valores ideales. Comparamos la salida real de la red neuronal con estos datos.

this.real.getInputData(i - SinWave.INPUT\_SIZE,entrada); this.real.getOutputData(i - SinWave.INPUT\_SIZE, salida);

Se crea **un StringBuilder** para contener los datos con formato.

final StringBuilder str = nuevo StringBuilder(); str.append(i);

str.append(":Actual=");

para (int j = 0; j < output.length; j++) { if (j > 0) {

str.append(',');

}

str.append(salida[j]);

}

La red neuronal está llamada para calcular su predicción.

doble predicción final[] = this.network.computeOutputs(entrada);

El valor predicho se muestra con el valor real.

str.append(":P redicted=");

para (int j = 0; j < output.length; j++) { if (j > 0) {

str.append(',');

}

str.append(predecir[j]);

}

La diferencia también se muestra como un porcentaje.

str.append(":D ifference=");

error finalCalculation = nuevo ErrorCalculation(); error.updateError(predecir, salida); str.append(percentFormat.format(error.calculateRMS()));

System.out.println(str.toString());

}

Por último, la línea para este valor real es salida. Esta red neuronal hace una razón-

un buen trabajo para predecir los valores futuros. Hay muchas maneras diferentes en los datosde entrada se pueden presentar para la predicción. En el siguiente capítulo exploraremos cómo crear redes neuronales que intenten predecir los mercados financieros.

# Resumen del capítulo

La red neuronal feedforward es muy experta en reconocer patrones. Apoyo utilizado- erly, una red neuronal feedforward se puede utilizar para predecir patrones futuros. Estas redes neuronales se denominan redes neuronales predictivas o temporales. Un trabajo de red neuronal predictiva no es un tipo específico de red neuronal; más bien, es cualquier tipo de red neuronal utilizada para la predicción. Este libro utiliza redes neuronales feedforward para todos los predictien ex-amples.

La implementación de una red neuronal feedforward que predice es simplemente una cuestión de construir correctamente las neuronas de entrada y salida. El tiempo se divide en varios bloques, llamados sectores de tiempo. Por ejemplo, una red neuronal puede tener cinco sectores de tiempo conocidos seguidos de un sector de tiempo desconocido. Esto produciría una red neuronal con cinco neuronas de entrada y una neurona de salida. Dicha red neuronal se entrenaría utilizando datos reales conocidos en grupos de seis sectores de tiempo. Las primeras cinco rebanadas de cada grupo serían las neuronas de entrada. La sexta sería la salida ideal. Para que la red neuronal prediga, simplemente proporcionaría cinco rebanadas de tiempo conocidas a las neuronas de entrada. La salida de la neurona de salida sería la predicción de la red neuronal para el sexto corte de tiempo.

Un libro entero podría escribirse fácilmente en redes neuronalespredictivas. Este capítulo introdujo redes neuronales predictivas a un nivel básico prediciendo la onda sinusoidal. Tal vez una de las aplicaciones más comunes de las redes neuronales predictivas es predecir el movimiento en los mercados financieros. El siguiente capítulo proporcionará una troduccióna la programación de redes neuronales para las predicciones del mercado financiero.

# vocabulario

Datos reales

Onda sinusoidal de red neuronal predictiva

Rebanada temporal de tiempo de red neuronal

# Preguntas para revisión

1. **¿Se puede utilizar un mapa autoorganizado para la predicción? Si es así, ¿cómo configuraría las neuronas de entrada y salida?**
2. **Cuando se utiliza una red neuronal feedforward para la predicción, ¿qué representan las neuronas de entrada? ¿Qué representan las neuronas de salida?**
3. **Tiene 500 intervalos de información. ¿Cómo podría utilizar estos datos para entrenar y probar una red neuronal predictiva?**
4. **¿Cómo elegiría entre el uso de una tangente hiperbólica y una función sigmoide como la función de activación para una red neuronal predictiva?**

Tangente hipervolica, dado que trabaja con valores entre el rango [1, -1]. Mientras que la function sigmoide solo funciona para valore positivos.

1. **¿Por qué podrías usar más de una neurona de salida en un trabajo predictivo de redes neuronales?**

Graficar prediccion del programa

Graficar el error