Trabajo Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica

Artificial intelligence techniques and their application to time series forecast

Autor: Antonio Luis González Hernández

Tutor: Juan Antonio Becerra González

Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024

Departamento de Teoría de la Señal y Comunicaciones

Trabajo Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica

Artificial intelligence techniques and their application to time series forecast

Autor:

Antonio Luis González Hernández

Tutor:

Juan Antonio Becerra González

Profesor Titular

Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024

Trabajo Fin de Grado: Artificial intelligence techniques and their application to time series forecast

Autor: Antonio Luis González Hernández
Tutor: Juan Antonio Becerra González

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Índice Abreviado

Contents

1 Ejemplo de Capítulo

Una de las virtudes del ingeniero es la eficiencia.

GUANG TSE

El *formato de capítulo* abarca diversos factores. Un capítulo puede incluir, además de texto, los siguientes elementos:

- *Figuras*
- *Tablas*
- *Ecuaciones*
- *Ejemplos*
- *Resúmenes*, con recuadros en gris, por ejemplo
- *Lemas*, *corolarios*, *teoremas*,... y sus demostraciones
- *Cuestiones*
- *Problemas* propuestos
- ...

En este capítulo se propone incluir ejemplos de todos estos elementos, para que el usuario pueda modificarlos fácilmente para su uso. Consulte el código suministrado, para ver cómo se escriben en LaTeX.

1.1 Ejemplo de sección

En la Figure 1.1 se incluye a modo de ejemplo la imagen del logo de la ETSI [1]. El código para que aparezca dicha imagen se muestra en el cuadro siguiente:

Si nos detenemos en los comandos que hemos utilizado, con `width` se controla el ancho, y se escala así el tamaño de la imagen. En LaTeXexisten diversas opciones para situar la figura en la página: con `t` o `b` se le indica que las incluya arriba o abajo (top/bottom) y con `!` se le pide que la deje dónde está, tras el texto anterior.

Code 1.1 Código para incluir una figura.

```
\begin{figure}[htbp]
\centering
\includegraphics[width=3 cm]{capituloLibroETSI/figuras/logoESI.pdf}
\caption{Logo de la ETSI}
\label{fig:figura1}
\end{figure}
```

Para dar énfasis a algún texto, usamos `\emph`. Así, por ejemplo,

[1] Se usa aquí el package de acrónimos, que la primera vez define el acrónimo y ya luego sólo incluye el mismo. Esto facilita luego generar de forma automática la lista de acrónimos.

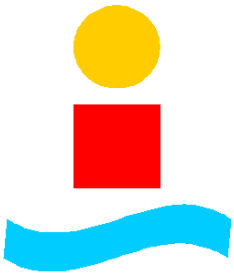

Figure 1.1 Logo de la ETSI.

No olvide intentar utilizar este formato en sus publicaciones de la ETSI

hace aparecer el anterior texto en itálica. Pero si escribiésemos, por ejemplo,

No olvide intentar utilizar este formato siempre *en sus publicaciones de la ETSI*

vemos cómo hemos destacado la palabra "siempre" en torno a su contexto. Para ello, hemos escrito, realmente, `\emph{siempre}` dentro de la frase original.

1.1.1 Ejemplo de subsección

Si se usaba `\section` para indicar una sección, se utiliza `\subsection` para una subsección.

1.2 Elementos del texto

1.2.1 Figuras

Además del tipo de figura que vimos anteriormente, el normal, podemos desear incluir una figura en modo apaisado ocupando toda la página. Para ello utilizamos el entorno de figura siguiente `\begin{sidewaysfigure}`, cuyo resultado se puede observar en la Figure 1.2.

Aunque puede optar por la forma que desee, en el fichero `notacion.sty` se incluyen definiciones para que pueda usar `\LABFIG{etiqueta}` y `\FIG{etiqueta}` para poner una etiqueta y hacer referencia a la misma luego. Además, está definido para que `\FIG{etiqueta}` incluya por delante el término Figura.

1.2.2 Tablas

A modo de ejemplo, Table 1.1 incluye un ejemplo de tabla. Al igual que con figura, si usa `notacion.sty` puede usar `\LABTAB{etiqueta}` y `\TAB{etiqueta}` para poner una etiqueta y una referencia, y el `\TAB{etiqueta}` ya incluye el nombre Tabla por delante.

Una alternativa al uso de estos comandos está representado por el uso del comando `\autoref{etiqueta}` que, en conjunción con el paquete `babel` genera automáticamente los nombres de Figura o Tabla, en función de la etiqueta correspondiente.

Table 1.1 Valores de parámetros.

Definición	notación	valor
Potencia transmitida (entregada a antena)	P_{et}	-5 a 20 dBm
Ganancia antenas	G	40.5 dBi

1.2.3 Listados de programas

Es muy habitual en nuestros documentos que tengamos que incluir listados de programas. Para ello, se propone la utilización de un paquete denominado `listings`. Se obtiene con él un listado como el mostrado en el Code 1.2 de MATLAB® siguiente:

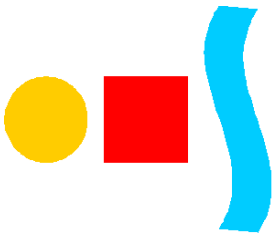

Figure 1.2 Logo de la ETSI.

Code 1.2 Representación de la función rect$(t - T/2)$.

```
clear all
close all
T = 1;
A = 1;
L = 100;
tstep = T/L;
t = 0:tstep:T-tstep;
g_t = A*ones(1,L);
figure(1);
subplot(211);
h=plot(t,g_t); axis( [0 T -A-0.1 A+0.1]);
set(h,'linewidth', 1.0);
ylabel('g(t)'), xlabel('t[s]'); grid on;

g_n = g_t;
subplot(212);
h=stem(g_n, '.', 'filled'); axis( [1 L -0.1 A+0.1]);
set(h,'linewidth', 1.0);
ylabel('g(n)'), xlabel('n');
```

También se puede generar en este caso una relación de los códigos usados en nuestro documento, de manera equivalente a la relación de figuras o tablas. Para ello, observar la correspondiente codificación en el fichero principal.

1.2.4 Ecuaciones

Para escribir expresiones matemáticas, como por ejemplo $2+2=4$, sólo hace falta que meta la expresión entre símbolos `$`. En el fichero notacion.sty se incluyen muchas definiciones para facilitar la escritura de estas expresiones y de ecuaciones. Para escribir una ecuación, con una o más líneas, se aconseja utilizar `align`, como en el siguiente ejemplo, en las ecuaciones (1.1)-(1.3),

$$T = kT_b, \tag{1.1}$$

$$R_b = \frac{1}{T_b}, \tag{1.2}$$

$$D = \frac{1}{T} = \frac{R_b}{k} = \frac{R_b}{\log_2 M}. \tag{1.3}$$

Si no quiere numerar una línea, utilice las instrucción \nonumber antes de poner \\ para escribir la siguiente línea. Y con & puede alinear las ecuaciones.

Un ejemplo más complejo de ecuaciones sería el siguiente: decimos que el vector aleatorio $\mathbf{Z}$ es gaussiano si su función densidad de probabilidad conjunta viene dada por:

$$f_{\mathbf{Z}}(\mathbf{z}) = \frac{1}{(2\pi)^N |\mathbf{C_Z}|^{1/2}} \mathrm{e}^{-\frac{1}{2}(\mathbf{z}-\mathbf{m_Z})^\top \mathbf{C_Z}^{-1}(\mathbf{z}-\mathbf{m_Z})} \tag{1.4}$$

con el vector media la matriz $\mathbf{m_Z}$ y la matriz de covarianza real $\mathbf{C_Z}$ $(2N \times 2N)$ simétrica definida positiva dado por:

$$\mathbf{m_Z} = \begin{bmatrix} \mathbf{m_X} \\ \mathbf{m_Y} \end{bmatrix}, \quad \mathbf{C_Z} = \begin{bmatrix} \mathbf{C_X} & \mathbf{C_{XY}} \\ \mathbf{C_{YX}} & \mathbf{C_Y} \end{bmatrix} \tag{1.5}$$

con el vector $\boldsymbol{\omega}$ dado por:

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_N \\ \omega_{N+1} \\ \vdots \\ \omega_{2N} \end{bmatrix} \tag{1.6}$$

Si no desea que se numere una ecuación puede poner asterisco, tanto en el entorno `equation` como `align`.

1.2.5 Ejemplos

Para incluir un ejemplo, utilize el entorno \ejmp, usando el entorno \begin{ejmp} y \end{ejmp}, y para la solución el entorno \begin{sol}.

Example 1.2.1 Calcule $2+2$.

Solution. Para resolver esto se puede utilizar que $1+1=2$, de la siguiente forma

$$2+2=(1+1)+(1+1)=4,$$

donde se ha contado, pruebe a utilizar los dedos de su mano, a cuatro.

Observad que antes de comenzar el ejemplo y tras su finalización se han incluido unos *filetes* a modo de resalte en el texto. En el caso de una serie de ejemplos, los entornos \begin{ejmpn} y \begin{soln}, junto con los entornos de cierre correspondientes, permiten que no existan estos filetes entre los ejemplos y soluciones intermedias de la serie.

1.2.6 Lemas, teoremas y similares

Se incluyen ejemplos de estos elementos de texto. Empezamos con la Definición 1.2.1 y la Propiedad 1.2.1:

Definition 1.2.1 (Suma) La suma es la operación que permite contar sobre un número, otro.

Property 1.2.1 (Suma) *Los números enteros se pueden sumar.*

Lemma 1.2.1 (Suma de 1 y 1) *La suma* $1+1$ *es igual a* 2.

Proof. Ponga un dedo a la vista, junto a otro, y cuéntelos. ■

Theorem 1.2.1 (Suma) *La suma de cualquier número y dos es igual a la suma del mismo número más uno más uno.*

Proof. Por inducción y el Lema 1.2.1. ■

Corollary 1.2.1.1 (Contables) *Los números enteros son contables.*

Proof. Por el Teorema 1.2.1. ■

1.2.7 Resúmenes

Para incluir un resumen de una sección o un conjunto de secciones o en cualquier otro punto que consideremos interesante, se utiliza el entorno \begin{Resumen}, que admite como parámetro opcional un nombre que queramos asignarle al resumen. Por defecto, se denomina "Resumen". Observar que se ha modificado la cabecera de las páginas impares. Una vez finalizado el resumen, con el comando \end{Resumen}, se recupera la anterior cabecera automáticamente. Los resúmenes que se deseen incluir aparecen en la tabla de contenidos como una sección sin numeración, con el nombre elegido o el nombre por defecto de Resumen. En el siguiente ejemplo hemos utilizado este parámetro opcional de nombre.

Resumen de Teoría de Información

Debido al considerable número de definiciones, teoremas y propiedades que hemos descrito en los apartados anteriores, vamos a presentar un resumen de los principales resultados, no necesariamente en el mismo orden que el expuesto anteriormente. Supondremos en este resumen que las variables aleatorias X, Y y Z son discretas, definidas en el alfabeto $\mathscr{X}$, $\mathscr{Y}$ y $\mathscr{Z}$ respectivamente.

Entropía de una variable aleatoria discreta

Se define la entropía $H(X)$ de una variable aleatoria discreta X , con función masa de probabilidad $p(x)$, en la forma:

$$H(X) = -\sum_{x\in\mathscr{X}} p(x)\log p(x) = \mathbb{E}[-\log p(X)]$$

1. Se cumple:

$$0 \leqslant H(X) \leqslant \log|\mathscr{X}|$$

con la igualdad en la izquierda si y sólo si $p_i = 1$ para algún $x_i \in \mathscr{X}$ y con la igualdad a la derecha si y sólo si la variable aleatoria está uniformemente distribuida; esto es, $p_i = 1/|\mathscr{X}|$ para todo i.

2. $H(X) = 0$ si y sólo si X es determinista.

3. $H(X) = H(p(x))$ es una función cóncava en $p(x)$.

4. Se define la *Función de Entropía Binaria* en la forma:

$$h_b(p) \stackrel{\text{def}}{=} -p\log p - (1-p)\log(1-p)$$

5. La función entropía binaria $h_b(p)$ es una función cóncava en p.

6. Si X y $\hat{X}$ son dos variables aleatorias estadísticamente independientes igualmente distribuidas,

$$\Pr\left(X = \hat{X}\right) \geqslant 2^{-H(X)}$$

con la igualdad si y sólo si X tiene una distribución uniforme.

Entropía conjunta y entropía condicional

Definimos la *entropía conjunta* de las variables aleatorias X e Y, $H(X,Y)$ en la forma:

$$H(X,Y) = \sum_{x\in\mathscr{X}}\sum_{y\in\mathscr{Y}} p(x,y)\log\frac{1}{p(x,y)} = \mathbb{E}[-\log p(X,Y)]$$

Definimos la *entropía condicional* $H(X\mid Y)$ en la forma:

$$\begin{aligned} H(X\mid Y) &= \sum_{y\in\mathscr{Y}} p(y)H(X\mid Y=y) = \\ &= -\sum_{x\in\mathscr{X}}\sum_{y\in\mathscr{Y}} p(x,y)\log p(x\mid y) = \\ &= \mathbb{E}[-\log p(X\mid Y)] \end{aligned}$$

1.

$$H(X,Y) \leqslant H(X) + H(Y)$$

con la igualdad si y sólo si X e Y son estadísticamente independientes.

2.

$$\begin{aligned} H(X\mid Y) &\leqslant H(X) \\ H(Y\mid X) &\leqslant H(Y) \end{aligned}$$

con la igualdad si y sólo si X e Y son estadísticamente independientes.

3. $H(X \mid Y) = 0$ si y sólo si X es una función de Y.

4.
$$H(X \mid X) = 0$$

5.
$$H(X,Y) = H(Y) + H(X \mid Y)$$

6.
$$H(X,Y) = H(X) + H(Y \mid X)$$

7.
$$H(X,Y \mid Z) = H(X \mid Z) + H(Y \mid X,Z)$$

8. *Desigualdad de Fano* Sean X y $\hat{X}$ dos variables aleatorias que toman valores en el mismo alfabeto $\mathscr{X}$. Se verifica:
$$H\left(X \mid \hat{X}\right) \leqslant h_b(p_e) + p_e \log\left(|\mathscr{X}| - 1\right)$$

Reglas de las cadenas

Sea $\mathbf{X}$ un vector formado por las N variables aleatorias $X_i, i = 1,2,\ldots,N$.

1. *Regla de la cadena para la entropía*

$$\begin{aligned} H(X_1,X_2,\ldots,X_N) &= \sum_{i=1}^{N} H\left(X_i \mid X_1,\ldots,X_{i-1}\right) = \\ &= H(X_1) + H(X_2 \mid X_1) + \cdots + H\left(X_N \mid X_1,\ldots,X_{N-1}\right) \end{aligned}$$

2. *Regla de la cadena para la entropía condicional*

$$H(X_1,X_2,\ldots,X_N \mid Y) = \sum_{i=1}^{N} H\left(X_i \mid X_1,\ldots,X_{i-1},Y\right)$$

3. *Regla de la cadena para la información mutua*

$$I(X_1,X_2,\ldots,X_N;Y) = \sum_{i=1}^{N} I\left(X_i;Y \mid X_1,\ldots,X_{i-1}\right)$$

4. *Regla de la cadena para la Información Mutua Condicional*

$$I(X_1,X_2,\ldots,X_N;Z \mid Y) = \sum_{i=1}^{N} I\left(X_i;Z \mid X_1,\ldots,X_{i-1},Y\right)$$

1.3 Una nueva sección después del resumen

Problemas Propuestos

Esto es un ejemplo de cómo incluir cuestiones y/o problemas al final de un capítulo, con o sin solución. Para poner un problema o cuestión, usar \begin{prob} y \end{prob}. Para incluir la solución, a continuación usar \begin{soln} seguido del texto terminado en \end{soln}.

1.1 Sean A, B y C sucesos de un cierto experimento con probabilidades dadas por:

$$\begin{aligned}
\Pr(A) &= \frac{1}{3} \\
\Pr(B) &= \frac{1}{4} \\
\Pr(C) &= \frac{1}{5} \\
\Pr(A \cap B) &= \frac{1}{12} \\
\Pr(A \cap C) &= \frac{1}{15} \\
\Pr(B \cap C) &= \frac{1}{20} \\
\Pr(A \cap B \cap C) &= \frac{1}{30}
\end{aligned}$$

a) ¿Son los sucesos independientes dos a dos? ¿Son estadísticamente independientes?

b) Encontrar $\Pr(A \cup B)$.

c) Encontrar $\Pr(A \cup B \cup C)$.

1.2 Suponer que tenemos una moneda con las siguientes características: cuando se lanza, la probabilidad de que salga cara es $\Pr(c) = p$ y la probabilidad de que salga cruz $\Pr(+) = q$. Lanzamos dos veces la moneda y queremos conocer acerca de la independencia estadísticade los siguientes sucesos:

$$\begin{aligned}
A &= \text{Sale c en la primera tirada} \\
B &= \text{En las dos tiradas sale lo mismo} \\
C &= \text{Sale c en la segunda tirada}
\end{aligned}$$

1.3 Determinar la media, la autocorrelación y el espectro densidad de potencia de la salida de un sistema con respuesta impulsiva dada por:

$$h(n) = \begin{cases} 1 & n = 0,2 \\ -2 & n = 1 \\ 0 & \text{en cualquier otro caso} \end{cases}$$

cuando la señal de entrada es un ruido blanco $X(n)$ con varianza σ_X^2.

Solution. Si el ruido es blanco, su valor esperado será cero y su espectro densidad de potencia será una constante: $S_X(\Omega) = C$. Calculemos su autocorrelación. Se tiene:

$$R_X(k) = Ł[F]^{-1}[S_X(\Omega)] = Ł[F]^{-1}[C] = \frac{1}{2\pi}\int_{-\pi}^{\pi} Ce^{jk\Omega}\mathrm{d}\Omega = \begin{cases} k \neq 0 & \frac{C}{2\pi}\frac{\mathrm{e}^{jk\Omega}}{jk}\Big|_{-\pi}^{\pi} \\ k = 0 & C \end{cases} \Rightarrow$$

$$R_X(k) = C\delta(k) = \begin{cases} C & k = 0 \\ 0 & k \neq 0 \end{cases}$$

Ahora bien: (...) □

Anexo

Si quiere introducir un separador dentro de un capítulo puede utilizar la instrucción subchapter. Esto le puede interesar, por ejemplo, para introducir alguna información adicional al final de un capítulo como un anexo al mismo.

En las secciones que siguen vamos a repasar algunas materias que utilizamos ampliamente a lo largo del texto y que sostienen de forma rigurosa el estudio de las comunicaciones digitales.

1.4 Señales: definición y clasificación

Una puede definirse como una función que transmite información generalmente sobre el estado o el comportamiento de un sistema físico, [**?**]. Aunque las señales puedan representarse de muchas maneras, en todos los casos la información está contenida en la variación de alguna magnitud física. Matemáticamente se representan como una función de una o más variables independientes. Por ejemplo, una señal de voz se puede representar como una función del tiempo y una imagen fotográfica puede representarse como una variación de la luminosidad respecto a dos parámetros espaciales. En cualquier caso, es una práctica común denotar como tiempo, t, a la variable independiente, en el caso de una variación continua de la variable independiente, y n en caso contrario.

1.4.1 Clasificación de señales

Establezcamos a continuación una clasificación de las señales atendiendo a diversos puntos de vista.

Señales deterministas y aleatorias

Una señal se clasifica como determinista cuando no hay incertidumbre alguna acerca del valor que tiene en cualquier instante de tiempo. Estas señales pueden modelarse como una función matemática, por ejemplo, $g(t) = 10\cos\left(4\pi t^2\right)$.

Una señal aleatoria es aquella para la que existe cierta incertidumbre respecto a su valor. Matemáticamente vamos a modelarla como una función muestra de un proceso aleatorio.

Para que una señal transmita información debe tener un carácter aleatorio, [**?**].

Señales periódicas y no periódicas.

Una señal $g(t)$ es periódica, con periodo T_0, si existe una cantidad $T_0 > 0$ tal que:

$$g(t) = g(t+T_0) \quad \forall t \tag{1.7}$$

siendo T_0 el valor más pequeño que cumple esta relación. Una señal que no cumpla (1.7) se denomina no periódica.

Señales analógicas, discretas, muestreadas y digitales

Una señal analógica $g(t)$ es aquella que está definida para todo t. Una señal discreta sólo está definida en un conjunto numerable[2] de valores del tiempo. Una señal muestreada está definida para todo instante de tiempo, aunque sólo puede tomar valores en un conjunto numerable y una señal digital es aquella que sólo está definida en un conjunto numerable de valores del tiempo y toma valores en un conjunto numerable.

[2] Un conjunto es numerable o contable cuando sus elementos pueden ponerse en correspondencia uno a uno con el conjunto de los números naturales. Con posterioridad veremos que el concepto de conjunto numerable o contable juega un papel importante en el desarrollo de numerosos aspectos de la teoría.

2 Transformer

2.1 What is a Transformer?

2.1.1 Origin and Basic Concept of Transformers

Transformers were introduced by Vaswani et al. in their seminal 2017 paper titled "Attention is All You Need" [5]. This groundbreaking work marked a significant departure from the traditional architectures used in natural language processing (NLP) and opened up new possibilities for handling language-related tasks. The primary innovation of the Transformer model is its use of a mechanism called self-attention or scaled dot-product attention. This mechanism allows the model to weigh the importance of different words in a sentence regardless of their position, addressing the limitations inherent in previous models like recurrent and convolutional neural networks.

Recurrent Neural Networks (RNNs) and their variants, such as Long Short-Term Memory networks (LSTMs), process sequences of data one step at a time, maintaining a hidden state that carries forward contextual information. This sequential processing, however, makes it difficult for RNNs to handle long-range dependencies efficiently, as the information must be passed along a chain of cells, which can lead to the vanishing gradient problem. Convolutional Neural Networks (CNNs), on the other hand, use a sliding window approach to capture local features, but they struggle with global context due to their fixed-size receptive fields.

The self-attention mechanism in Transformers allows the model to evaluate relationships between all words in a sentence simultaneously. This means that each word can directly attend to every other word, regardless of their distance from one another in the sequence. This is achieved through a process where the model generates three vectors for each word: a query, a key, and a value. The attention scores are calculated as the dot products of the query with all keys, scaled and passed through a softmax function to obtain weights, which are then used to compute a weighted sum of the values. This enables the model to focus on the most relevant parts of the input sequence when producing an output, greatly enhancing its ability to understand context over longer distances.

2.1.2 Fundamental Differences from Other Neural Models

Transformers differ fundamentally from Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs) in several key aspects, particularly in how they process input data. RNNs and LSTMs process inputs sequentially, one time step at a time, which inherently limits their ability to parallelize computations. This sequential nature means that during training and inference, each step depends on the computation of the previous step, leading to slower processing times, especially for long sequences.

In contrast, Transformers process the entire sequence of data simultaneously, thanks to their reliance on the self-attention mechanism. This parallelization significantly speeds up training and inference times, making Transformers particularly well-suited for handling large datasets and complex tasks. By evaluating the entire context of a sentence at once, Transformers can capture dependencies between distant words more effectively. This is in stark contrast to RNNs and LSTMs, which might struggle with long-range dependencies due to their sequential processing.

Moreover, the self-attention mechanism enables Transformers to capture dependencies between distant words more effectively than Convolutional Neural Networks (CNNs). CNNs are typically used for tasks like

image recognition, where local features are paramount. They use layers of convolutions to progressively capture larger spatial contexts, but they still face limitations when it comes to capturing global context due to their inherently local receptive fields. Transformers, however, do not suffer from this limitation because their self-attention mechanism allows each word in the input sequence to directly attend to every other word, regardless of their position.

Another notable advantage of Transformers over RNNs and LSTMs is their ability to handle variable-length inputs and outputs efficiently. While RNNs and LSTMs can also manage variable-length sequences, their performance often degrades as the sequence length increases. Transformers, however, maintain consistent performance across different sequence lengths due to their parallel processing capabilities and self-attention mechanism.

2.2 Basic Structure of a Transformer

2.2.1 Attention Mechanism

The core component of the Transformer architecture is the attention mechanism, which fundamentally changes how models handle relationships between elements in a sequence. This mechanism allows the model to dynamically focus on different parts of the input sequence when producing each element of the output sequence, enabling a more nuanced understanding of the data.

The attention mechanism operates by computing a weighted sum of input values, known as the *values* (V), where the weights are derived from the similarity between a query (Q) and corresponding keys (K). The similarity scores determine how much focus to place on each input value when producing the output. Mathematically, this process is expressed by the following equation:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Here, Q represents the queries, K represents the keys, V represents the values, and d_k is the dimensionality of the keys. The dot products of the queries and keys are scaled by $\sqrt{d_k}$ to stabilize the gradients and then passed through a softmax function to obtain the attention weights. These weights are then used to compute a weighted sum of the values, producing the final output.

The attention mechanism allows the model to weigh the importance of each word in the input sequence based on its relevance to the current word being processed, regardless of their position in the sequence. This ability to capture long-range dependencies and context is a significant advantage over previous models that relied on fixed-size receptive fields or sequential processing.

2.2.2 Encoder and Decoder

The Transformer architecture consists of two main components: the *encoder* and the *decoder*. Both the encoder and decoder are composed of multiple layers that work together to process the input and generate the output sequences.

Encoder

The encoder is a stack of identical layers, each comprising two primary sub-layers:

- **Multi-Head Self-Attention Mechanism**: This sub-layer allows the encoder to attend to different positions of the input sequence simultaneously. By using multiple heads, the model can capture various aspects of relationships between words. Each head performs an attention function independently, and their outputs are concatenated and linearly transformed to form the final output.
- **Position-Wise Fully Connected Feed-Forward Network**: This sub-layer consists of two linear transformations with a ReLU activation in between. It processes each position independently and identically, enhancing the model's ability to learn complex patterns.

To facilitate training, each of these sub-layers is surrounded by layer normalization and residual connections. The residual connections help in preventing the vanishing gradient problem and enable deeper networks by providing a direct path for gradients during backpropagation. Layer normalization ensures that the inputs to the sub-layers have a stable distribution, which accelerates convergence.

Decoder

The decoder is also composed of a stack of identical layers, but it includes an additional attention sub-layer:

- **Masked Multi-Head Self-Attention Mechanism**: Similar to the encoder's self-attention, but with a masking operation to prevent attending to future positions in the sequence. This ensures that predictions for position i can depend only on the known outputs at positions less than i.
- **Encoder-Decoder Attention**: This sub-layer attends to the encoder's output, allowing the decoder to condition its predictions on the entire input sequence. It helps the decoder focus on relevant parts of the input sequence while generating each word of the output.
- **Position-Wise Fully Connected Feed-Forward Network**: Identical to the one used in the encoder, providing additional non-linearity and complexity to the model's predictions.

Similar to the encoder, each of these sub-layers in the decoder is wrapped with layer normalization and residual connections to facilitate training. The combination of self-attention, encoder-decoder attention, and feed-forward networks allows the decoder to generate accurate and contextually relevant outputs by leveraging the information encoded in the input sequence.

In summary, the Transformer's attention mechanism and its encoder-decoder structure provide a powerful framework for processing sequences. The self-attention mechanism captures relationships across the entire input sequence, while the encoder-decoder architecture enables complex and context-aware generation of output sequences, making Transformers highly effective for a wide range of NLP tasks.

2.3 Examples of Applications in Other Fields

Transformers have found applications beyond their original scope in Natural Language Processing (NLP). They are now widely used in various domains:

Natural Language Processing (NLP): Models such as BERT [2] and GPT-3 [1] leverage the Transformer architecture for tasks including language modeling, translation, and text generation. BERT (Bidirectional Encoder Representations from Transformers) is designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers, which makes it particularly powerful for understanding the nuances of language. GPT-3 (Generative Pre-trained Transformer 3), on the other hand, is a state-of-the-art autoregressive language model capable of generating human-like text based on a given prompt, demonstrating the effectiveness of Transformers in creating coherent and contextually relevant text.

Computer Vision: Vision Transformers (ViTs) [3] apply the Transformer architecture to image classification tasks, achieving state-of-the-art performance by treating image patches as sequences of tokens. ViTs divide an image into a sequence of fixed-size patches, linearly embed each patch, add position embeddings, and then feed the resulting sequence of embeddings to a standard Transformer encoder. This approach allows ViTs to capture the global context of an image more effectively than traditional convolutional neural networks (CNNs), leading to impressive results on image recognition benchmarks.

Audio Processing: In the field of audio processing, Transformers are used for tasks such as speech recognition, audio generation, and music transcription. By treating audio signals as sequences, similar to text, Transformers can model long-range dependencies and capture temporal patterns more effectively than conventional methods.

Protein Structure Prediction: Transformers have made significant strides in the field of bioinformatics, particularly in protein structure prediction. The AlphaFold model [4] by DeepMind utilizes a Transformer-based architecture to predict the 3D structure of proteins from their amino acid sequences. This breakthrough has the potential to revolutionize biological research and drug discovery by providing accurate models of protein structures.

Reinforcement Learning: In reinforcement learning, Transformers are employed to model the sequential nature of decision-making processes. By leveraging their ability to handle long-range dependencies and parallelize computations, Transformers enhance the performance of reinforcement learning algorithms in various applications, from game playing to robotic control.

Time-Series Forecasting: Another significant application of Transformers is in the field of time-series forecasting. By treating time-series data as sequential input, Transformers can effectively model temporal dependencies and capture patterns over long time horizons. This makes them particularly well-suited for predicting future values in various domains, such as finance, weather forecasting, and inventory management.

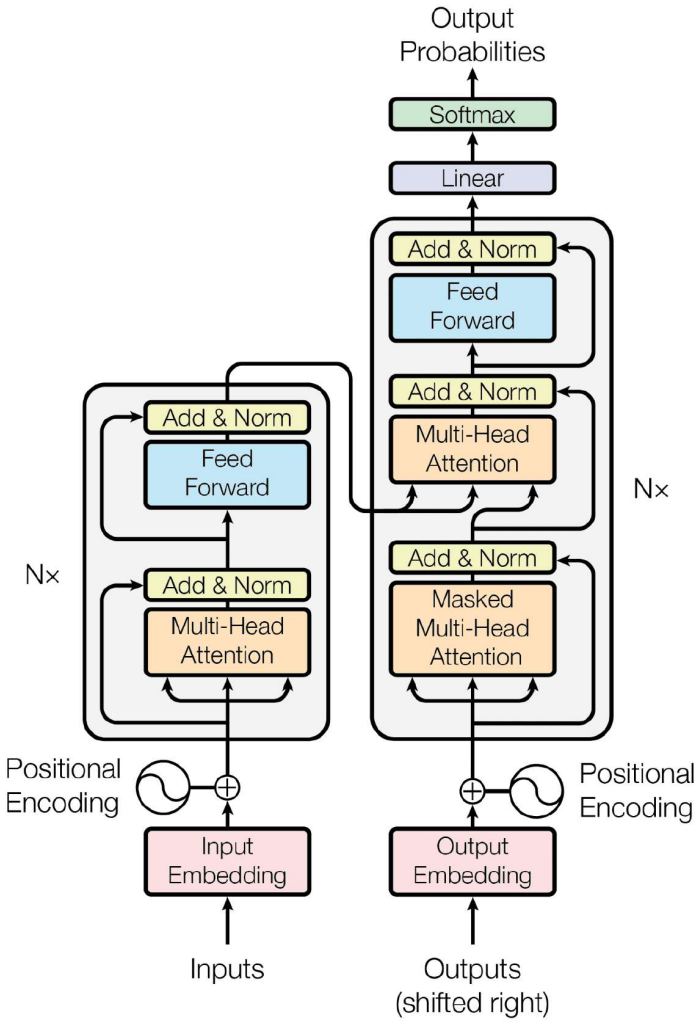

Figure 2.1 The encoder-decoder structure of the Transformer architecture. Taken from “Attention Is All You Need".

3 Time series Transformer

3.1 Overview

3.1.1 Enhancing Time Series Models with Transformers

Transformers, through their use of multi-head attention, have the potential to significantly enhance time series models' ability to manage long-term dependencies. This approach offers distinct advantages over current methodologies. To illustrate the effectiveness of transformers in handling long-term dependencies, consider how ChatGPT generates extensive and detailed responses in language models. By applying multi-head attention to time series, similar benefits can be achieved: one attention head can concentrate on long-term dependencies, while another focuses on short-term dependencies. This division of attention enables a more comprehensive analysis of the data. Consequently, we believe that transformers could enable time series models to forecast up to 1,000 data points into the future, or possibly even more.

3.1.2 The Challenge of Quadratic Complexity

Transformers face a significant challenge in calculating multi-head self-attention for time series. Since each data point in the series must interact with every other data point, adding more data points exponentially increases the computation time required. This phenomenon, known as quadratic complexity, results in a computational bottleneck when processing lengthy sequences.

List of Figures

List of Tables

List of Codes

Bibliography

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei, *Language models are few-shot learners*, 2020.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby, *An image is worth 16x16 words: Transformers for image recognition at scale*, 2021.

[4] John M. Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andy Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis, *Highly accurate protein structure prediction with alphafold*, Nature **596** (2021), 583 – 589.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, *Attention is all you need*, 2023.

Index