UNIVERSIDAD NACIONAL DE CÓRDOBA

FACULTAD DE MATEMÁTICA, ASTRONOMIA, FÍSICA Y COMPUTACIÓN.

Métodos computacionales para el cálculo de la volatilidad implícita del modelo de Black Scholes

Tesis realizada por Diego Juan Nasareno Lupi para la Licenciatura en Ciencias de la Computación en la Universidad Nacional de Córdoba

---

Dirigida por:
Doctora Noemí Patricia Kisbye
Licenciado Pedro Ángel Pury

Agradecimientos

- completar
- completar
- completar

Resumen

Las finanzas cuantitativas constituyen, desde hace varias décadas, un área particular de estudio dentro de la matemática. Esta nueva disciplina surge de la necesidad de encontrar modelos cuantitativos que permitan describir el comportamiento aleatorio de activos financieros y, en particular, valorar los productos llamados derivados financieros.
Si la hipótesis sobre la dinámica de los activos es que estos siguen un proceso estocástico lognormal, con tendencia y volatilidad constante, entonces la valoración de una opción call sobre el activo está dada por la fórmula de Black Scholes. Ahora bien, dado que la volatilidad no es observable en el mercado, se define la volatilidad implícita del activo como aquella que iguala la prima del mercado con el valor dado por la fórmula.
La obtención de este parámetro de volatilidad implícita permite luego valorar otros derivados financieros como así también comprender movimientos propios del mercado.

Por otra parte, la determinación de la volatilidad implícita requiere de la aplicación de métodos numéricos, puesto que se trata de resolver una ecuación no lineal sin una solución cerrada. En los últimos años, a su vez, han aparecido propuestas de uso de métodos de aprendizaje automático para modelar de manera empírica la función que provee la volatilidad implícita.

Este trabajo incluye la exploración bibliográfica referida al concepto de volatilidad implícita y sus implicancias, y de métodos computacionales factibles de ser implementados para su cálculo. Además se realizará la implementación efectiva en computadora de algunas soluciones y se hará un análisis comparativo de la eficiencia de los distintos métodos estudiados.

Índice general

Capítulo 1

Introducción

Para el desarrollo del trabajo que presentamos en este informe, es necesario estudiar y comprender varias herramientas y técnicas que usaremos a lo largo del proyecto. A continuación presentamos un resumen de los temas que abordaremos.

1.1. Motivación

La volatilidad implícita en estos días ha adquirido una gran importancia. Se puede utilizar para el calculo de superficies de volatilidad. Las superficies de volatilidad son usadas para valuar opciones mediante modelos matématicos. Otra utilidad es como parametro de riesgo para inversores, dependiendo de la volatilidad implícita se pondera algunas estrategias sobre otras para disminuir el riesgo y aumentar su posibilidad de rentabilidad.
Por eso es indispensable poder calcular la volatilidad implícita, generalmente se utilizan métodos iterativos para estimarla, pero estos métodos suelen ser lentos. Por lo tanto se propone el cálculo de la volatilidad implícita mediante redes neuronales para desminuir considerablemente el tiempo de calculo.

1.2. Objetivos del trabajo

El objetivo del trabajo es determinar la volatilidad implícita sobre call europeas, mediante métodos numéricos, o el uso de métodos de aprendizaje automático para modelar de manera empírica la función que provee la volatilidad implícita. Ya que se trata de resolver una ecuación no lineal sin una solución cerrada.

1.3. Estructura de la Tesis

La tesis esta estructurada de la siguiente manera:

- Nociones Preliminares: Se presentan los conceptos básicos utilizados a lo largo del trabajo.
- Implicancias de la volatilidad implícita: Se muestra la importancia de su calculo mediante aplicaciones de la misma.
- Implementación númerica del método de Bisección y de Brent: Se implementa el método de Bisección y del método brent para aproximar la volatilidad ímplicita utilizando la fórmula de Black Scholes.
- Implementación con Red Neuronal Feed-Forward: Se implementa una red neuronal para aproximar la volatilidad implicita utilizando la fórmula de Balck Scholes.
- Resultados: Se presentan los resultados obtenidos comparando los métodos numéricos con la red nueronal.
- Conclusiones: Se presentan conclusiones del trabajo y posibles mejoras.

Capítulo 2

Nociones Preliminares

2.1. Conceptos Financieros elementales

2.1.1. Los tipos de Interés

Si depositamos dinero en una cuenta bancaria, al cabo de un cierto tiempo este capital se incrementa en un determinado monto, llamado **interés**.

1. **Interés Simple**:

   $V = V_0(1 + r)^t$

   Donde V representa el valor de un depósito de valor inicial V_0 transcurrido un tiempo t y r es la tasa de interés anual.

2. **Interés Compuesto**:

   $V = V_0(1 + \frac{r}{n})^{nt}$

   Donde que V , V_0 y t tienen las mismas condiciones que para el interés simple, pero una tasa r de interés compuesto n veces al año.

3. **Interés Continuo**:

   El interés continuo puede verse como el interés compuesto para $n \to \infty$, es decir:

   $\lim_{n\to\infty} V_0(1 + \frac{r}{n})^{nt} = V_0 e^{rt}$

2.1.2. Tasa libre de riesgo

A los efectos de valoración de derivados se asume la existencia de una tasa llamada **tasa libre de riesgo**. Se trata de una tasa de referencia que no

tiene riesgo crediticio, es decir, que un inversor sabe que invirtiendo a esa tasa podrá recuperar el capital. Por ejemplo, los bonos del tesoro(Bonar 2025), o alguna tasa a la cual el propio estado ofrece para la devolución del préstamo(LEBAC, LETES). [1]

2.1.3. Productos básicos

Denominamos productos básicos a aquellos instrumentos financieros cuyo valor no depende de otro activo. Entre ellos están las acciones, los índices, las monedas, los commodities y lnos bonos.

2.1.4. Derivados

Un derivado puede ser definido como un instrumento financiero cuyo valor depende o deriva de los valores de otros activos subyacentes. Estos activos subyacentes podrían ser activos básicos u otros derivados.
En términos generales, un derivado consiste en un contrato entre dos partes para comprar o vender el subyacente. Las características y condiciones de estos contratos dan lugar a diferentes tipos de derivados, tales como contratos forward, futuros y opciones.

2.1.5. Opciones

Las opciones son contratos que dan derecho a una de sus partes a comprar (o vender) el subyacente, a un precio determinando en un tiempo futuro. Las opciones que dan derecho a compra se denominan **calls** y las que dan derecho a venta se denominan **puts**. Estos contratos tienen un valor inicial denominado prima, que es el precio al cual compra el contrato quien adquiere el derecho a compra o venta.
Quien compra una opción está en posición long sobre el contrato, y quien la vende está en posición short.
El agente que este en posición long tiene derecho a ejercerla o no. En cambio el que este en posición short tiene una obligación sobre lo que haga su contraparte.
Las opciones que se negocian en mercados formales se denominan *opciones vanilla o estándar*. En el mercado OTC se negocia una variedad mucho mayor de opciones y se denominan en general *opciones exóticas*. [1]
Dentro de las opciones vanilla existen dos tipos:

- **Opciones Europeas**: Opciones europeas: son aquellas cuyo ejercicio ocurre sólo en la fecha de madurez.

- **Opciones americanas**: son aquellas que pueden ser ejercidas en cualquier momento previo a la madurez.

2.1.6. Opcion Europea

En cada contrato se fija entonces un precio de ejercicio o **strike**, que es el precio pactado al cual se comprará o venderá el subyacente, en la fecha de expiración o también llamada **madurez** del contrato. Así, un inversor que negocia una opción adquiere una de las siguientes posiciones en el contrato:

- posición long: quien compra la opción, y por lo tanto tiene derecho a ejercerla.
- posición short: quien vende la opción o suscriptor, y por lo tanto contrae una obligación.

2.1.7. Mercados financieros

Los instrumentos financieros se comercializan en el mercado financiero. En la práctica existe un mercado formal u organizado y un mercado extrabursátil, denominado también over *the counter market* (OTC). En el caso del mercado formal, las negociaciones son multilaterales, es decir, existen múltiples agentes que actúan como compradores o vendedores de productos financieros. Los instrumentos que se comercializan están estandarizados, y existe una regulación de este tipo de mercados para proteger a los inversores de posibles incumplimientos de las contrapartes. En el caso del mercado extrabursátil, las negociaciones son bilaterales, es decir, entre dos partes. En estos casos los contratos suelen acordarse entre las partes en cuanto a la cantidad y características del subyacente. No existe una regulación formal sino que se basa en un principio de confianza entre partes.

El tamaño del segmento de mercados OTC es varias veces mayor que el de los mercados regulados, especialmente en los contratos de tipos de interés y de divisas. Según un estudio de ICAP[3], estima que cada día se producen 2 millones de transacciones en los mercados OTC por un nominal de 5 billones de dólares.

2.1.8. Movimiento Browniano

Sea $(\Omega, \mathcal{F}, \mathbb{P})$ un espacio de probabilidad. Si para cada $\omega \in \Omega$ existe una función continua $W : \mathbb{R} \to \mathbb{R}$ que depende de ω, entonces el proceso $\{W(t), t \geq$

0} se denomina Movimiento browniano con tendencia μ y volatilidad σ si satisface:

- $W(0) = 0$
- si $0 = t_0 < t_1 < ... < t_n$, entonces

$W(t_1) = W(t_1) - W(0), \qquad W(t_2) - W(t_1), \qquad W(t_3) - W(t_2),$
$..., \qquad W(t_n) - W(t_{n-1}),$
son variables aleatorias independientes, y cada uno de estos incrementos está normalmente distribuido con media y varianza dada por:

- $E(W(t_i) - W(t_{i-1})) = \mu(t_i - t_{i-1})$,
- $Var(W(t_i) - W(t_{i-1}) = \sigma^2(t_i - t_{i-1})$.

2.1.9. Movimiento Geométrico Browniano

Sea $(\Omega, \mathcal{F}, \mathbb{P})$ un espacio de probabilidad. Si para cada $\omega \in \Omega$ existe una función continua $S : \mathbb{R} \to \mathbb{R}$ que depende de ω, entonces el proceso $\{S(t), t \geq 0\}$ se denomina Movimiento Geométrico Browniano con tendencia μ y volatilidad σ (MGB) si se cumple que:

$\log\left(\frac{S(t)}{S(0)}\right)$, es un movimiento browniano con tendencia μ y volatilidad σ [1]

2.1.10. Valoración de una call europea con Black-Scholes Fómula

Sea c la prima de una opción call europea, con strike K y madurez T , sobre un activo cuyo precio sigue un movimiento geométrico browniano con tendencia $r - \sigma^2/2$ y volatilidad σ bajo las probabilidades de riesgo neutral. Sea r la tasa libre de riesgo. Entonces, bajo una hipótesis de no arbitraje se cumple que:

$c = S(0)\Phi(d_1) - Ke^{-rT}\Phi(d_2)$,

donde:
Φ es la distribución normal estándar acumulada,[4]

$$d_1 = \frac{Log\left(\frac{S(0)}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}} \qquad d_2 = d_1 - \sigma\sqrt{T}$$

con:

- $S(0)$: Precio inicial del activo.
- r: tasa libre de riesgo.
- K: Strike.
- T: Madurez de la opción.
- σ: volatilidad del activo.

[1]
Para simplificar la notación vamos a utilizar S_n en vez de $S(n)$, $\forall n > 0$

2.1.11. Volatilidad

La volatilidad mide la incertidumbre acerca del precio futuro de un activo. En teoría, la volatilidad se calcula continuamente, para valuar las opciones, tal como se dan los cambios en el valor de S. Black Scholes asume σ constante, esto implica una previa estimación estadística de σ, por ejemplo medir el comportamiento de los precios en los últimos meses y usar estimadores de varianza(volatilidad Histórica).
Como vimos en la sección 2.1.10 la fórmula de Black Scholes para una opción call depende de distintos parámetros, el precio inicial del activo, la tasa libre de riesgo, el stike, la madurez de la opción y su volatilidad. Pero esta ultima no te la brinda el mercado.

Volatilidad Histórica

La volatilidad histórica como su nombre lo indica me muestra el riesgo histórico de un periodo de tiempo de hoy hacia atrás. Se calcula midiendo las variaciones que han tenido los rendimiento del activo en cierto periodo de tiempo, que puede ser 20 días, 50 días o 200 días o el que cada analista considere mejor y esta volatilidad por lo regular se presenta anualizada. La metodología mas común para calcularla es calculando las desviaciones estándar de los rendimientos del activo.

Volatilidad Implícita

La volatilidad implícita muestra cual es el riesgo que están percibiendo los inversionistas de hoy en adelante, al contrario de la volatilidad histórica, esta es una volatilidad futura. Es calculada midiendo implícitamente como se están valorando o a que precios se están vendiendo los contratos de opciones de cierto activo.

Como vimos en la sección 2.1.10 para valorar una call europea necesitamos conocer, el precio inicial del subyacente, la tasa libre de riesgo, el strike, el tiempo de madurez de la opción y la volatilidad del subyacente. Los primeros cuatro parámetros son conocidos al momento de iniciar la opción. En cambio σ representa una volatilidad del activo en el período de vigencia de la opción, y por lo tanto es desconocido. Más aún, se está suponiendo constante cuando en la práctica puede ser un valor variable e incluso estocástico. Dado que las opciones call cotizan en el mercado, también es conocida la prima de la opción. Por ello se denomina volatilidad implícita al valor de σ que iguala la prima de la opción con la correspondiente fórmula de Black-Scholes.

2.1.12. Superficie de volatilidad

Para valorar las opciones los agentes usan la superficie de volatilidad, ya que la volatilidad depende del strike y de la madurez.
Una superficie de volatilidad es una representación tridimensional de las volatilidades implícitas de un subyacente en relación con los diferentes precios de ejercicio y las diferentes fechas de vencimiento.
La superficie de volatilidad combina las sonrisas de volatilidad con la estructura temporal de la volatilidad para tabular las volatilidades adecuadas. De este modo poder valorar una opción con cualquier precio de ejercicio y fecha de vencimiento. Un ejemplo de superficie de volatilidad que puede ser usada para opciones sobre divisas se muestra en el Cuadro 2.1. En este caso asumimos que la sonrisa es medida como la relación entre la volatilidad y $\frac{K}{S_0}$.
Una dimensión de la tabla es $\frac{K}{S_0}$ y la otra es el tiempo de madurez. Los valores de la tabla son las volatilidades implícitas calculadas con el método de Black-Scholes. Y las volatilidades que no se encuentren en la tabla son calculadas mediante interpolación, por lo general se utiliza Spline.[2]

2.2. Teorema del valor intermedio

Sea f una funcion continua en [a,b] sea d entre f(a) y f(b) entonces existe $c \in [a, b]$ tal que$f(c) = d$

Cuadro 2.1: Superficie de volatilidad

	K/S_0				
	0.9	0.95	1.00	1.05	1.10
1 mes	14.2	13.0	12.0	13.1	14.5
3 meses	14.0	13.0	12.0	13.1	14.2
6 meses	14.1	13.3	12.5	13.4	14.3
1 año	14.7	14.0	13.5	14.0	15.1
2 años	15.0	14.4	14.0	14.5	15.1
5 años	14.8	14.6	14.4	14.7	15.0

2.3. Método de Bisección

El método de bisección se basa en el teorema de valor intermedio para f funcion continua. Si f es continua en el intervalo [a,b] y f(a)*f(b)< 0, entonces f tiene al menos una raiz en (a,b).

Si el algoritmo de bisección se aplica a una función continua f en un intervalo [a, b], donde $f(a)f(b) < 0$, entonces, después de n pasos se habrá calculado una raíz aproximada con un error a lo más de $\frac{(b-a)}{2^{n+1}}$ [5]

El Algoritmo 2.1 aplica el metodo de bisección.

2.4. Método de Brent

El método de Brent es un algoritmo de búsqueda de raíces que combina el método de bisección, el método secante y la interpolación cuadrática inversa. Este método converge siempre que los valores de la función sean computables dentro de una región dada que contiene una raíz.
A diferencia del método de Bisección, el método de brent converge haciendo menos iteraciones sin perder la robustez del método de bisección. Ya que para ciertos casos utiliza el método de bisección, sin caer en los problemas de divergencia que tienen el metodo de interpolación cuadrática inversa, o del método de la Secante.

Método de la Secante:

Método de Newton: $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$

En el método de la secante imita el de Newton pero evita el cálculo de derivadas remplazamos $f'(x_n)$ de la formula de Newton por una aproximación de su derivada:
$f'(x) = \lim_{h\to 0} \dfrac{f(x+h) - f(x)}{h} \approx \dfrac{f(x+h) - f(x)}{h}$ [6]

Obteniendo como resultado:

$$x_{n+1} = x_n - f(x_n)\frac{x_n x_{n-1}}{f(x_n)f(x_{n-1})}$$

interpolación cuadrática inversa:

Es un metodo iterativo, que aplica la fórmula de interpolación de lagrange para hacer una interpolación cuadrática en el inverso de f, para hallar $f(x) = 0$:
$x_{k+1} = \dfrac{f_{n-1}f_n}{(f_{n-2} - f_{n-1})(f_{n-2} - f_n)}x_{n-2} + \dfrac{f_n f_{n-2}}{(f_{n-1} - f_n)(f_{n-1} - f_{n-2})}x_{n-1} \dfrac{f_{n-1}f_{n-2}}{(f_n - f_{n-1})(f_n - f_{n-2})}x_n$
[7]

Donde $f_k = f(x_k)$. El algoritmo 2.2 aplica el metodo de brent.

2.5. Redes neuronales

Una red neuronal intentará estimar los parámetros optimos para una función $f(x, W, b)$, a partir de un conjunto de entrenamiento $\{(x_1, y_1), ..., (x_n, y_n)\}$, tal que $f(x_i, W, b) \approx y_i$, para $i = 1, ..., n$.
Donde $x = [x_1, ..., x_n]$, $y = [y_1, ..., y_n]$

2.5.1. Perceptrón

La unidad básica de una red neuronal son los perceptrones (o neuronas). Un perceptrón toma un vector de entradas de valor real, calcula la combinación lineal de estas entradas con respecto a los pesos, luego genera una salida, dependiendo de la función de activación. Más precisamente:

$$o(x_1, ..., x_n) = \psi(w_0 + x_1 w_1 + ... + x_n w_n)$$

Algorithm 2.1: Bisección

Input : f, a, b, tol
Output: c
Precondition: f(a)f(b) < 0 **and** a < b

```
1  err := b - a;
2  while tol < err do
3  |   c := (a+b)/2;
4  |   err := err/2;
5  |   fc := f(c);
6  |   if fc = 0 then
7  |   |   return;
8  |   end
9  |   if fcf(a) < 0 then
10 |   |   b := c;
11 |   else
12 |   |   a := c;
13 |   end
14 end
15 return;
```

donde cada w_i es una constante de valor real, o peso, y ψ es la función de activación [13], como muestra la siguiente figura:

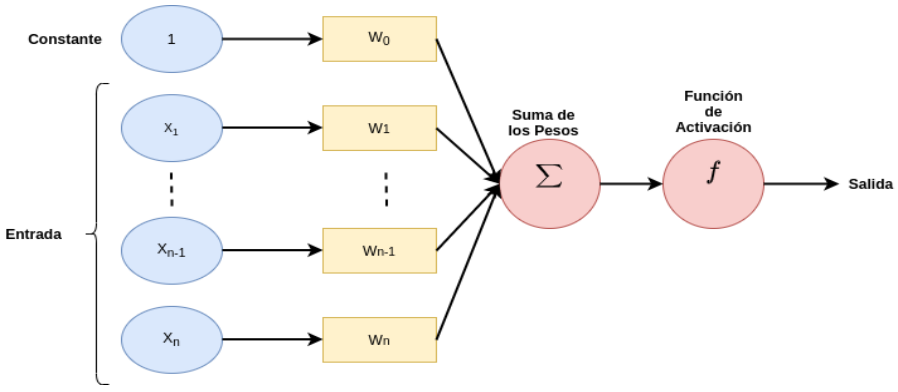

Figura 2.1: Perceptrón

2.5.2. Redes Feed-Forward

Este tipo de algoritmo recibe el nombre de "red" porque se construye componiendo funciones (perceptrones).

Algorithm 2.2: Brent

Input : f, a, b, tol
Output: b
Precondition: f(a)f(b) < 0

if $|f(a)| < |f(b)|$ **then**
  swap(a,b);
end
c := a;
mflag := True;
while *tol* $< |b - a|$ ***or*** *f(b) = 0* **do**
  if *f(a)* $\neq$ *f(c) and f(b)* $\neq$ *f(c)* **then**
    A := $\dfrac{af(b)f(c)}{(f(a)-f(b))(f(a)-f(c))}$;
    B := $\dfrac{bf(a)f(c)}{(f(b)-f(a))(f(b)-f(c))}$;
    C := $\dfrac{cf(a)f(b)}{(f(c)-f(a))(f(c)-f(b))}$;
    s := A+B+C; (interpolacion cuadrática inversa)
  else
    s := $b - f(b)\dfrac{b-a}{f(b)-f(a)}$; (metodo de secante)
  end
  if ***not*** $\left(\dfrac{3a+b}{4} < s < b\right)$ ***or*** *(mflag* ***and*** $|s-b| \geq |b-c|/2$*)* ***or***
   *(****not*** *mflag* ***and*** $|s-b| \geq |c-d|/2$*)* ***or*** *(mflag* ***and*** $|b-c| < tol$*)* ***or***
   *(****not*** *mflag* ***and*** $|c-d| < tol$*)* **then**
    mflag := True;
    s := $\dfrac{a+b}{2}$; (metodo de bisección)
  else
    mflag := False;
  end
  d := c;
  c := b;
  if *f(a)f(s) < 0* **then**
    b := s
  else
    a := s
  end
  if $|f(a)| < |f(b)|$ **then**
    swap(a,b);
  end
end
return;

La arquitectura de este tipo de modelos se puede dividir en tres partes principales, denominadas capas, tal como se muestra en la Figura 2.2. Las capas en este tipo de redes se conocen como completamente conectadas (o fully connected) debido a que cada neurona de una capa se conecta con todas las neuronas en la capa siguiente pero nunca con neuronas de la misma capa.
En la Figura 2.2 podemos observar las distintas capas que componen una red neuronal. En primer lugar tenemos la capa de entrada, donde se define cuántos valores de entrada tomará nuestro modelo, estos serán luego enviados a la primer capa oculta. Después de esto puede venir una o más capas ocultas, seguidas de una capa de salida generando una aproximación del resultado deseado, por ende, si una red neuronal Feed-Forward tiene N capas, entonces tiene N-2 capas ocultas. Cada unidad (o input) alimenta solo las unidades de la siguiente capa. Las unidades en intermedio las capas a menudo se llaman capas ocultas(hidden units) porque no tienen conexión directa con el datos, tanto de entrada como de entrada ni salida. [14]

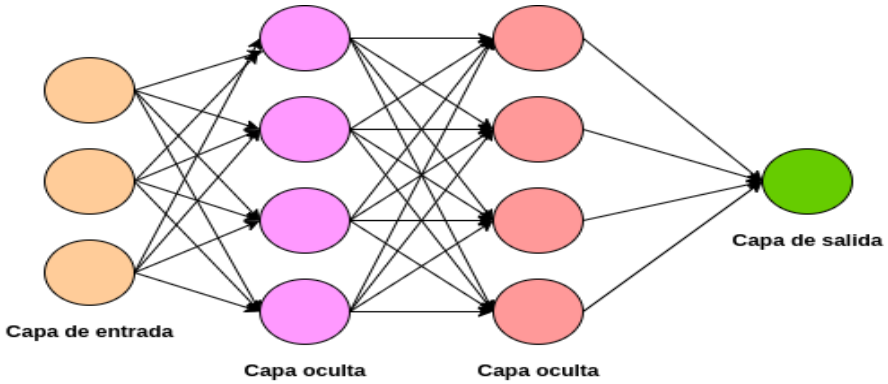

Figura 2.2: Feed-Forward

2.5.3. Funciones de Activación

La función de activación en las redes neuronales son las encargadas de enviar señales al siguiente nivel o siguiente capa. Es crucial elegir la función de activacion correcta para no caer en el *problema de desvanecimiento de gradiente*.
Otra importante caractirística que debe tener la función de activación es ser diferenciable. Ya que al aplicar el algoritmo del *descenso del gradiente*, cuando se propaga para hacia atras, calcula los gradientes de error con respecto a los pesos, para calcular los nuevos pesos acordemente. [9]
El Figura 2.2 muestra las funciones de activación que vamos a utilizar, en nuestro caso el α de la Elu es 1.

ReLu(z) = max(z,0)	$Tahn(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Elu(z) = $\begin{cases} \alpha(e^z - 1) & \text{si } z < 0 \\ z & \text{si } 0 \leq z \end{cases}$

Figura 2.3: Función de activación

2.5.4. Aprendizaje por Descenso del Gradiente

Como hemos mencionado anteriormente el objetivo de una red neuronal es estimar los parámetros óptimos para una función $f(x, W, b)$. Una forma de estimar esos parámetros es reducir el error. Sea
$J(x, W, b, Y) = \frac{1}{n}\sum_{i=1}^{n}(f(x_i, W, b) - y_i)^2$
una función de costo.

Para reducir dicho error se busca W tal que minimize la funcion $J(x, W, b, Y)$, se puede utilizar el descenso del gradiente.
Sea $w = [W, b]$, (para simplificar la notación). El gradiente de J en w, denotado como $\nabla J(w)$, es el vector de derivadas parciales de J, es decir, $\nabla J(w) = \left(\frac{\partial J(w)}{\partial w_1}, \frac{\partial J(w)}{\partial w_2}, ..., \frac{\partial J(w)}{\partial w_n}\right)$
El descenso del gradiente es un algoritmo iterativo, empezamos con valor inicial de w^0 (por ejemplo $w^0 = [0, 0, ..., 0]$) y entonces en cada iteración damos un paso en la dirección negativa del gradiente en el punto actual. Esto es, $w^{t+1} = w^t - \eta\nabla J(w^t)$, donde $\eta > 0$, conocido como el learning rate, es el encargado de decidir el tamaño del paso que damos. Ya que $\nabla J(x, w^t, Y, b) > 0$ cuando J es creciente en w^t, y $\nabla J(x, w^t, Y, b) < 0$ cuando J es decreciente en w^t (como muestra la imagen 2.4), se obtiene $J(x, w^{t+1}, b, Y) \leq J(x, w^t, b, Y)$, siempre y cuando η no sea muy grande(Observar Figura 2.5).
El algoritmo es el siguiente:

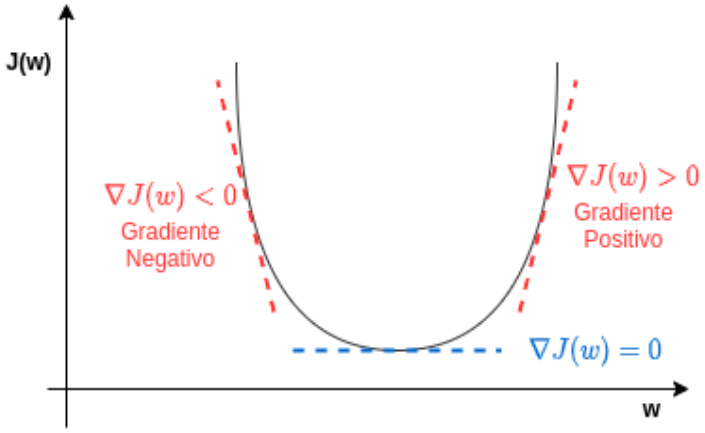

Figura 2.4: Descenso Del Gradiente

Algorithm 2.3: Descenso del Gradiente

Input : x, Y, η, T
Output: w
Precondition: d = len(w)

initialize w^0;
t := 0;
while $t < T$ **do**
    i := 0;
    while $i < d$ **do**
        $w_i^{t+1} = w_i^t - \eta \nabla J(x, w_i^t, Y, b)$;
        i := i+1;
    end
    t := t+1;
end
return;

2.5.5. Learning Rate

El learning rate determina el tamaño del paso en cada iteración mientras se mueve hacia un mínimo de una función de error. Un learning rate grande conduce rápidamente al un mínimo local, pero con riesgo de diverger. En cambio un learning rate pequeño no diverge pero necesita muchas iteraciones para llegar al mínimo local(Figura 2.5). Comúnmente se utiliza un learning rate grande y se lo va decrementando por cada epoca hasta encontrar un buen resultado.

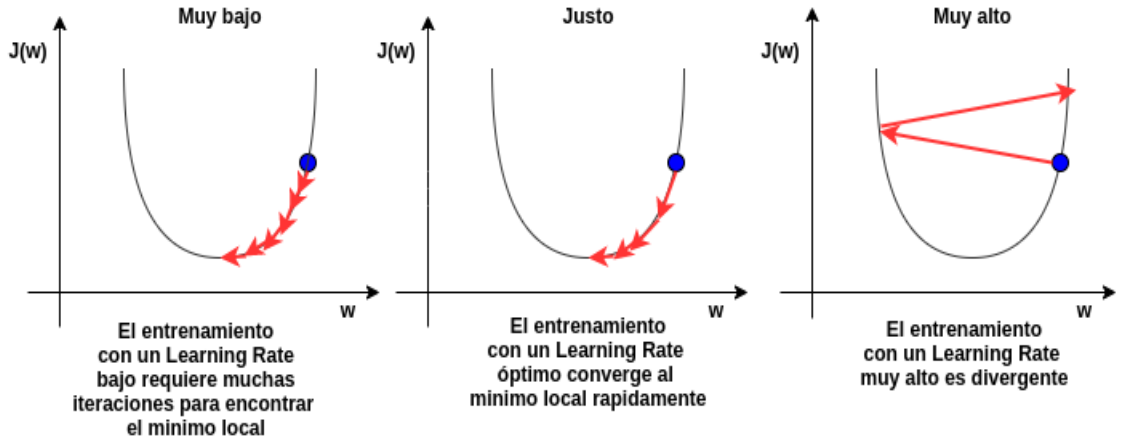

Figura 2.5: Learning Rate

Algoritmos de decrecimiento del Learning Rate:

Time-Based Decay: $base\_lr \frac{1}{1 + decay * epoch}$ [8]

Step Decay: $base\_lr * dacay^{floor\left(\frac{1+epoch}{epoch\_drop}\right)}$ [8]

Exponential Decay: $base\_lr * e^{(-k*epoch)}$ [10]

Donde epoch representa la epoca en cual la red se encuentra, base_lr es el valor inicial del learning rate.

Ciclycal Learning Rate:

En este caso el learning rate varia entre un mínimo y máximo, creciendo y decreciendo como muestra la Figura 2.6.
Para obtener dicho maximo y minimo utilizaremos un método similiar al de Smith [11] para determinar el learning rate, a diferencia del método de Smith, iremos subiendo el learning rate exponencialmente por cada batch. Empezaremos con un learning rate de 10^{-10} hasta llegar a 1, comparando la tasa de error contra el learning rate.

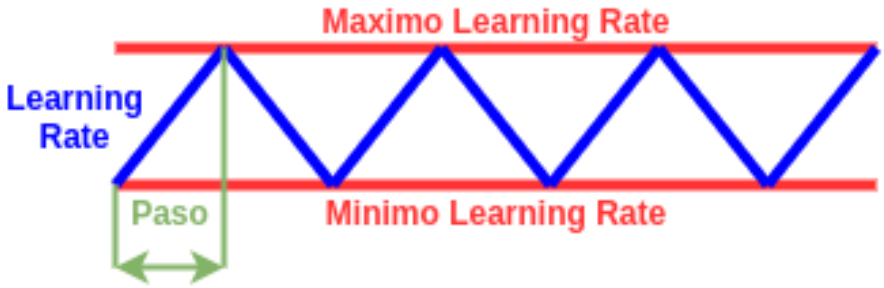

Figura 2.6: Cycical Learnig Rate

2.6. k-fold cross validation

K-fold cross validation es una técnica utilizada para evaluar modelos propuestos, con el fin de encontrar el mejor modelo. En K-fold cross validation los datos de prueba se dividen en K subconjuntos. Uno de los subconjuntos se utiliza como datos de test y el resto (K-1) como datos de entrenamiento. El proceso de cross validation es repetido durante K iteraciones, con cada uno de los posibles subconjuntos de datos de test. Finalmente se realiza el promedio de los resultados de cada modelo(cada modelo tiene K resultados), guardando el modelo que obtuvo mejor promedio. Este método es muy preciso puesto que evaluamos a partir de K combinaciones de datos de entrenamiento y de test. [12]

Capítulo 3

Implicancias de la volatilidad implícita

En esta sección describiremos diferentes usos de la volatilidad implícita.
El primer uso es en el calculo de superficies de volatilidad, que esta sirve para valorar opciones.
El segundo uso es utilizar la volatilidad implícita como indicador de riesgo de mercado, de esta manera poder ejercer estrategias, ya sean de cobertura o de especulación.

3.1. Superficie de Volatilidad para valorar opciones

Para valorar las opciones europeas, lookback, asiaticas, entre otras. Existen modelos matemáticos para calcular el valor la opción. A continuación listaremos una serie de modelos para calcular las distintas tipos de opciones. [15]

- Black-Scholes(1973): Modelo para valorar opciones europeas.
- Rubinstein (1991): Modelo para valorar opciones chooser simples.
- Conze y Viswanathan(1991): Modelo para valorar opciones lookbacks con precio de ejercicio fijo.
- Goldman, Sosin y Gatto(1979): Modelo para valorar opciones lookbacks con precio de ejercicio flotante.
- Kemma y Vorst (1990): Modelo para valorar opciones asiáticas con media geométrica.

- Levy (1992): Modelo para valorar opciones asiáticas con media aritmética.
- Margrabe (1978): Modelo para valorar opciones sobre el intercambio de dos activos.

Todos los modelos anteriormente mencionacionados para ser aplicados necesitan la volatilidad como parámetro, entre otros parámetros. Pero el mercado no nos brinda el valor de la volatilidad, una manera de estimarla es mediante la superficie de volatilidad, definida en el capítulo anterior.

3.2. Estrategias dependiendo de la volatilidad implícita

Entender el concepto de volatilidad es esencial para tener éxito en la comercialización de opciones. Un inversor que puede reconocer cuando una opción o series de opciones estan baratas o caras tiene una gran ventaja a la hora de invertir. Esto quiere decir que si la volatilidad implícita es baja, entonces el precio de la opción va a ser baja. En cambio si la volatilidad implícita es alta, el precio de la opción va a ser alta.
Ahora bien, como reconocer cuando la volatilidad implícita es alta o baja. Una manera es comparar la volatilidad histórica con la volatilidad implícita de la opción. Esto sería calcular las volatilidades históricas diarias de la opción sobre 1 o 2 años, y compararlas con la volatilidad implícta. Por ejmplo si la volatilidad implícita es menor al 80 % de las volatilidades históricas de los ultimos 2 años, entonces la volatilidad implícita es baja.

Ahora nombraremos algunas estrategias la cual se hace uso de la volatilidad implícita.[16]

3.2.1. Backspread

Las claves para aplicar la estrategia.

- Esperamos un movimiento particular del mercado, pero teniendo un poco de protección en caso de equivocarnos.
- Buscamos volatilidad implícita baja al momento de aplicar, con esperanza que suba en el futuro.

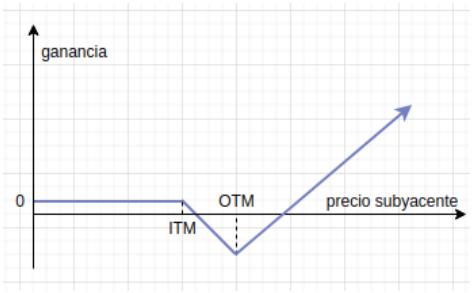

Figura 3.1: Call Backspread.

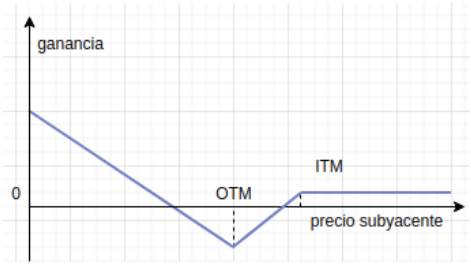

Figura 3.2: Put Backspread.

La estrategia de comprar una Backspread involucra escribir opcines call o put, en at-the-money o in-the-money(primas caras). Simultaneamente comprar un número mayor de opciones out-of-the-money(primas baratas). Idealmente se busca obtener que el valor de las opciones escritas sea mayor al valor de las opciones compradas.

En el caso que tengamos expectativas de un movimiento alcista del subyacente entonces la estrategía será Call Backspread, que consiste en vender y comprar opciones call. El objetivo es que el precio del subyacente sube o baje drasticamente para obtener ganancia, con preferencia alcista como se puede observar en la imagen 3.1.

En el caso que tengamos expectativas de un movimiento bajista del subyacente entonces la estrategía será Put Backspread, que consite en vender y comprar opciones put. El objetivo es el mismo que el Call Backspread, con preferencia bajista como se puede observar en la imagen 3.2.

3.2.2. Long Straddle o Long Strangle

Las claves para aplicar la estrategia.

- Esperamos que el valor del subyacente tenga un gran cambio, sin importar si el precio suba o baje.
- Buscamos volatilidad implícita baja al momento de aplicar.
- Idealmente opciones out-of-the-money.
- Tiempo adecuado antes de la expiración, tratar que no sean contratos cortos.

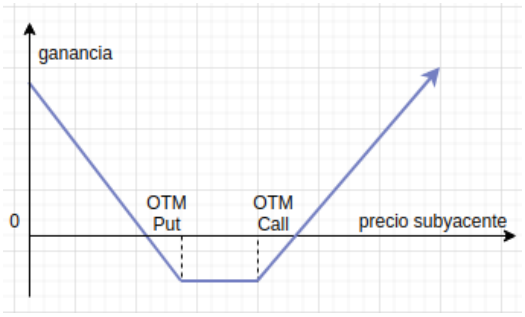

Figura 3.3: Long Strangle.

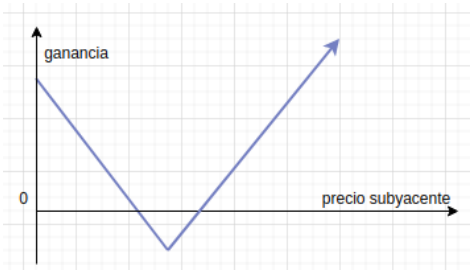

Figura 3.4: Long Straddle.

La estrategia de comprar una Straddle o Strangle consiste en comprar una opción call y una opción put al mismo tiempo, con el mismo tiempo de madurez, puede ser con el mismo strike(Straddle) o no(Strangle). Ideamente se busca que al momento de comprar las opciones, la volatilidad implícita sea baja(comprar opciones baratas), y que el precio del subayente tenga un abrupto cambio(Guerra comercial entre Estados Unidos y China, Pandemia(Precio Petroleo, entre otros)) sin importar su dirección, puede ser alcista o bajista. Por eso el tiempo de madurez es clave en este tipo de estrategias. La perdida en este tipo de estrategias es limitada, es el valor de las primas. En cambio la ganancia puede ser ilimitada.
La Figura 3.3 muestra la estrategia de Strangle, y la Figura 3.4 muestra la estrategia de Straddle.

3.2.3. Long Butterfly Spread

Las claves para aplicar la estrategia.

- Esperamos que el valor del subyacente no varie mucho.
- Buscamos volatilidad implícita alta al momento de aplicar, mientras mas alta mejor.
- Opciones cortas, tiempo de expiración menor a 60 dias.

La estrategia Long Butterfly Spread puede utilizarse tanto con opciones call, como con opciones put.
En caso que nuestra estrategia utilice opciones call, la estrategia consiste en comprar una opción call a un determinado strike, vender dos call con un strike mayor al strike de la opción nombrada anteriormente, y comprar otra opción call con un strike mayor a todos los anteriores.

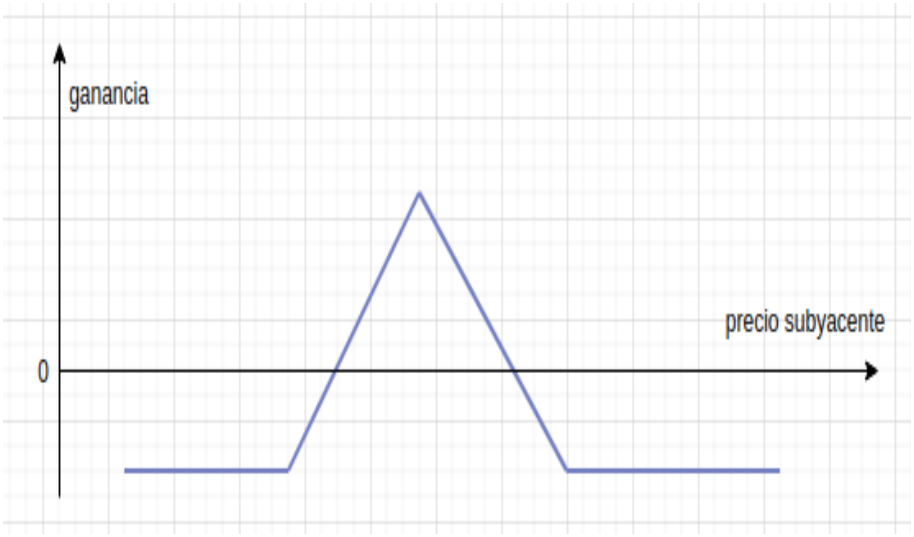

Figura 3.5: Long Butterfly Spread

En caso que nuestra estrategia utilice opciones put, la estrategia consiste en comprar una opción put a un determinado strike, vender dos put con un strike menor al strike de la opción nombrada anteriormente, y comprar otra opción put con un strike menor a todos los anteriores.
Esta estrategia la mayor ganacia se encuentra cuando el precio del subyacente se encuentre próximo al strike del medio(opciones vendidas). Pero al ser la volatilidad alta, el precio del subyacente es muy variable, siendo asi esta estrategia riesgosa. Una buena medida es salir cuando obtengamos una buena ganancia, sin esperar que el precio del subyacente llegue al pico, como se observa en la Figura 3.5.

Capítulo 4

Implementación númerica del método de Bisección y de Brent

En esta sección vamos a implementar el método de Bisección y el método brent para aproximar la volatilidad implícita utilizando la fórmula de Black Scholes.

4.1. Introducción

Tanto el método de Bisección como el método de Brent son algoritmos de búsqueda de raíces de funciones. Luego como nuestro objetivo es encontrar el valor de la volatilidad implícita.
La función 4.1 va a ser la función a la cual aplicaremos los métodos matemáticos.
Sea c la prima de una opción call europea.

$$g(\hat{\sigma}) = S(0)\Phi(d_1) - Ke^{-rT}\Phi(d_2) - c \tag{4.1}$$

donde:

$$d_1 = \frac{Log\left(\frac{S(0)}{K}\right) + \left(r + \frac{\hat{\sigma}^2}{2}\right)T}{\hat{\sigma}\sqrt{T}} \qquad d_2 = d_1 - \hat{\sigma}\sqrt{T}$$

4.2. g es monótona creciente con respecto a σ

Sea $B(\sigma) = S(0)\Phi(d_1) - Ke^{-rT}\Phi(d_2)$, donde d_1 y d_2 son definidas en la sección anterior. Luego veremos que B es una función monótona creciente respecto a σ. En primer lugar notemos que B es continua con respecto a σ, por lo cual restará analizar que su derivada respecto a σ es positiva:

$$B'(\sigma) = \frac{d}{d\sigma}B(\sigma) > 0$$

Tenemos que:

$$B'(\sigma) = S(0)\Phi'(d_1)d_1'(\sigma) - Ke^{-rT}\Phi'(d_2)d_2'(\sigma)$$

Ahora, dado que $d_2 = d_1 - \sigma T$, entonces $d_2' = d_1' - T$. Luego:

$$\begin{aligned} B'(\sigma) &= S(0)\Phi'(d_1)d_1'(\sigma) - Ke^{-rT}\Phi'(d_2)d_2'(\sigma) \\ &= (S(0)\Phi'(d_1)(\sigma) - Ke^{-rT}\Phi'(d_2))d_1'(\sigma) + K\Phi'(d_2)e^{-rT}\sqrt{T} \end{aligned}$$

El primer término de la última expresión es 0. En efecto, dado que Φ es la densidad de la normal estándar, entonces

$$\begin{aligned} S_0\Phi'(d_1) - Ke^{-rT}\Phi'(d_2) &= \frac{1}{\sqrt{2\pi}}\left(S_0^{-d_1^2/2} - Ke^{-rT-}\frac{(d_1-\sigma\sqrt{T})^2}{2}\right) \\ &= \frac{1}{\sqrt{2\pi}}\left(S_0^{-d_1^2/2} - Ke^{-rT-}\frac{d_1^2}{2} + d_1\sigma\sqrt{T} - \frac{\sigma^2 T}{2}\right) \\ &= \frac{1}{\sqrt{2\pi}}e^{d_1^2/2}\left(S_0 - Ke^{-rT+d_1\sigma\sqrt{T}-}\frac{\sigma^2 T}{2}\right) \end{aligned} \tag{4.2}$$

El exponente en la última expresión es:

$$-rT + d_1\sigma\sqrt{T} - \frac{\sigma^2 T}{2} = -rT + ln(\frac{S_0}{K}) + (r + \frac{\sigma^2}{2})T - \frac{\sigma^2 T}{2} = ln(\frac{S_0}{K})$$

Por lo tanto resulta:

$$S_0\Phi'(d_1) - Ke^{-rT}\Phi'(d_2) = \frac{1}{\sqrt{2\pi}}e^{-d_1^2/2}(S_0 - Ke^{ln(S_0/k)}) = 0$$

Volviendo al cálculo de $B'(\sigma)$ y usando que $\Phi'(x) > 0$ para cualquier x, tenemos entonces que

$$B'(\sigma) = Ke^{-rT}\Phi'(d_2)\sqrt{T} > 0 \tag{4.3}$$

que es una expresión positiva para todo $\sigma > 0$.
Como sabemos que B es una función monótonamente creciente con respecto σ, entonces

$$g(\sigma) = B(\sigma) - c \tag{4.4}$$

tambien lo es. [1]

4.3. Inicialización de intervalo para método de bisección y de brent

Como sabemos de la sección anterior, $g : (0, \infty) \to \mathbb{R}$ es una función monótona creciente. Podemos postular $a = \epsilon$, con $\epsilon \to 0$, entonces $g(a) < 0$.
Ahora para postular b utilizaremos el siguiente algoritmo:

Algorithm 4.1: Encontrar b

Input : g
Output: b

```
1 b := 1;
2 while g(b) < 0 do
3 |   b := b*10;
4 end
5 return;
```

4.4. Aplicación del Método de Bisección

Sea g la función definida en la sección 4.1.
Inicializamos el intervalo [a,b] para aplicar el método de bisección, con $a < b$ y $g(a) < 0 < g(b)$.
Luego aplicar el algoritmo de bisección hasta hallar un ξ, tal que $g(\xi) = 0$ ó $|b - a| < \varepsilon$, a ε lo llamaremos tolerancia.

4.5. Aplicación del Método de Brent

Sea g la función definida en la sección 4.1.
Inicializamos el intervalo [a,b] para aplicar el método de brent, con $a < b$ y $g(a) < 0 < g(b)$.
Luego aplicar el algoritmo de brent hasta hallar un ξ, tal que $g(\xi) = 0$ ó $|b - a| < \varepsilon$, a ε lo llamaremos tolerancia.

4.6. Problemas Númericos

En la muestra hay aproximado un 0.05 % de casos que no cumplen con la condición $g(a)g(b) < 0$, para $a \to 0$ y $b \to \infty$, donde g es la función definida en la sección 4.1. Por lo tanto no se puede aplicar ninguno de los métodos vistos anteriormente.
El problema radica en el cálculo de la fórmula de Black-Scholes definida en la sección 2.1.10, básicamente en $\Phi(d_1)$ y $\Phi(d_2)$. Por definición $\Phi(n) < 1$, $\forall n \in \mathbb{R}$. Pero Python usa aritmética de punto flotante IEEE-754, donde su precisión es $2^{(-56)}$, dando asi una precisión aproximada de 15,95 dijitos decimales. Luego usando la función de Scipy para el cálculo de la distribución normal acumulada definida en [17], obtenemos $\Phi(n) = 1$, para $n > 8{,}3$.

Volviendo a la fórmula de Black-Scholes, sabemos que $\Phi(d_1) > \Phi(d_2)$, ya que $o, T > 0$, pero hay casos en que el extremo inferior, definido para aplicar el método de bisección(o brent), obtengo $\Phi(d_1) = \Phi(d_2)$ por el problema de precisión, ocasionando el error antes mencionado.
Veamos un caso en particular. Sea:

- c : 5.983489610184446
- S : 15.752756180327959
- k : 10
- r : 0.09010364215460305
- T : 0.2590760904347537

La Figura 4.1 muestra la comparación entre g(n), para $n \in [0{,}01, 1]$, y la volatilidad implícita (linea roja punteada). Se puede observar que no se puede hallar el intervalo [a,b] para aplicar el método de bisección(o brent), ya que para este caso g(n) es positivo $\forall n < 0{,}01$.

Si tomo $o = 0{,}01$(extremo inferior del intervalo), obtengo:

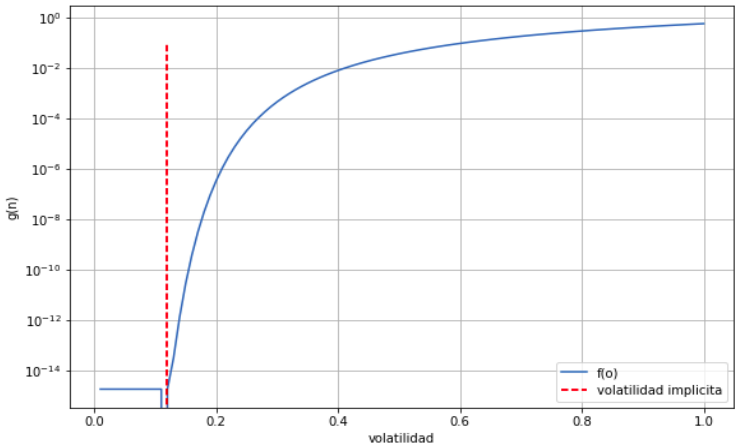

Figura 4.1: Función g

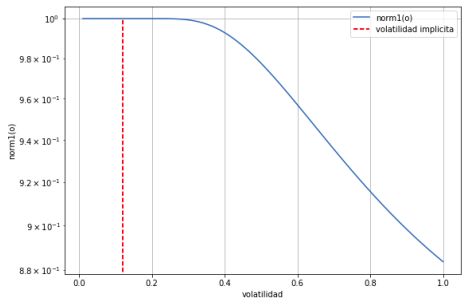

Figura 4.2: $\Phi(d_1)$.

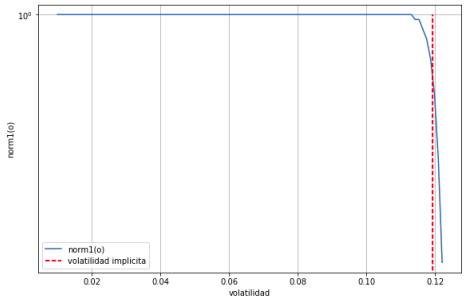

Figura 4.3: $\Phi(d_1)$.

- $\Phi(d_1) = 1$, $\Phi(d_2) = 1$

$g(0{,}01) = 1{,}7763568394002505e - 15$

Pero si aplico la normal con la volatilidad implícita $o = 0{,}11928197090875538$, obtengo:

- $\Phi(d_1) = 0{,}9999999999999986$, $\Phi(d_2) = 0{,}9999999999999982$

$g(0{,}11928197091) = 0$, contradiciendo la propiedad de monótona creciente.

Capítulo 5

Implementación con Red Neuronal Feed-Forward

En esta sección presentaremos el cálculo de la volatilidad implícita mediante redes neuronales.
Todo cálculo realizado en este cápitulo es sobre GPU(GeForce GTX 1080/PCIe/SSE2), RAM(15.6 GiB).

5.1. Calculo de prima de opción call con ratio

En el paper [18] se demostró que las redes neuronales estiman mejor el precio de la opción mediante el ratio(S/K), entonces obtenemos:

$$\frac{c}{K} = \frac{S(0)}{K}\Phi(d_1) - e^{-rT}\Phi(d_2) = B'(S(0)/K, T, r, \sigma).$$

donde:

$$d_1 = \frac{Log\left(\frac{S(0)}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}} \qquad d_2 = d_1 - \sigma\sqrt{T}.$$

5.2. Generación de muestra

Como hemos visto en la sección anterior, el nuevo cálculo de de prima nos dará un valor $\frac{c}{K}$. En nuestro caso buscamos estimar la volatilidad implícita, pero usaremos el ratio para generar la muestra ya que en la práctica la red estima mejor.

Luego vamos a generar 2 muestras, una muestra amplia y una muestra estrecha.
Las variables de la muestra ya sea amplia o estrecha, pertenecerán a un ambiente previamente definido, pero las variables de la muestra estrecha sera un subconjunto de la amplia.

	Parametros	muestra amplia	muestra estrecha
Entrada	precio ratio(S_0/K)	$[A^1,B^1]$	$[A^1+C_1^1,B^1-C_2^1]$
	Tiempo de madurez(τ)	$[A^2,B^2]$	$[A^2+C_1^2,B^2-C_2^2]$
	volatilidad(σ)	$[A^3,B^3]$	$[A^3+C_1^3,B^3-C_2^3]$
	Tasa libre de riesgo(r)	$[A^4,B^4]$	$[A^4+C_1^4,B^4-C_2^4]$
Salida	Precio de Call(c/K)	(O^1,L^1)	(O^2,L^2)

donde $A^i+C_1^i < B^i-C_2^i$, $C_j^i > 0$, $O^j < L^j$, para $i = 1,2,3,4$, $j = 1,2$.

Luego para generar una muestra de tamaño N, aplicaremos N veces el siguiente algoritmo, donde:

- **numero_aleatorio:** Es una función que dado un rango genera un número aleatorio perteneciente a dicho rango.
- **B′:** Es la fórmula de Black-Scholes dada en la sección anterior.
- **rango_ratio:** Es el rango de valores de el ratio(S_0/K).
- **rango_T:** es el rango de valores de el tiempo de maturez(τ).
- **rango_σ:** es el rango de valores de la volatilidad implícita (σ).
- **rango_r:** es el rango de valores de la tasa libre riesgo(r).

Algorithm 5.1: Generación muestra

Input : rango_ratio, rango_T, rango_σ, rango_r
Output: C

```
1 ratio := numero_aleatorio(rango_ratio);
2 T := numero_aleatorio(rango_T);
3 σ := numero_aleatorio(rango_ratio);
4 r := numero_aleatorio(rango_r);
5 C := B′(ratio, T, σ, r);
6 return;
```

Observar que C es c/K.

5.3. k-fold cross validation

En nuestro caso utilizaremos 8-fold cross validation para estimar los siguientes hiperparámetros.
Primero definiremos la estructura de la red neuronal entre 1 y 10 capas ocultas, variando entre 50 a 1000 neuronas por capa oculta. Luego definiremos la función de activación [19], la inicialización de pesos de la red [20] y el algoritmo de optimización del decenso del gradiente [21] en simultaneo(haciendo todas las convinaciones posibles). Siguiendo por determinar la mejor función de error [22].
Luego determinaremos el dropout. Y por ultimo el tamaño del batch.
Los hiperparámetros propuesto para aplicar 8-fold cross validation están definidos en el Cuadro 5.2.

Mientras se vayan definiendo los hiperparámetros óptimos, se usarán los hiperparámetros por defecto que utiliza Keras Sequential [23], excepto el tamaño del batch que será de 1024 como se observa en el Cuadro 5.1.

Cuadro 5.1: Hiperparametros por defecto

Parametros	Opciones
Función de error	ECM
Función de activación	ReLu
Inicialización de pesos	glorot_uniform
Algoritmo de optimización	SGD
Dropout	0
Tamaño de batch	1024
Learning rate	0.001
epocas	200

Sea:

$$\text{ECM} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

$$\text{EAM} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

$$\text{EPAM} = 100\frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{y_i}$$

$$\overline{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

Cuadro 5.2: Estimación de hiperparametros

Parametros	Opciones o Rango
Capas	[1,10]
Neuronas	[50,1000]
Función de error	ECM, EAM, EPAM
Función de activación	ReLu, Elu, tanh
Inicialización de pesos	uniform, glorot_uniform, he_uniform
Algoritmo de optimización	SGD, RMSprop, Adam
Dropout	[0,0.2]
Tamaño de batch	[256, 2048]

Cuadro 5.3: Hiperparametros

Parametros	Opciones
Capas	3
Neuronas	950
Función de error	ECM
Función de activación	ReLu
Inicialización de pesos	random_uniform
Algoritmo de optimización	Adam
Dropout	0
Tamaño de batch	1024

$$SS_{tot} = \sum_{i=1}^{N} (y_i - \overline{y})^2$$

$$R^2 = 1 - \frac{ECM}{SS_{tot}}$$

Donde y_i es el valor esperado de salida de la red, $\hat{y}_i$ es el valor que predice la red y N es el tamaño de muestra.

En el Cuadro 5.3 se observan los hiperparámetros que utilizaremos en el entrenamiento de la red.

Learning Rate

Anteriormente hemos usado un **learning rate** fijo(10^{-3}).
Tomando los hiperparámetros del Cuadro 5.3 utilizaremos un método similiar al de Smith [11] para determinar el learning rate, a diferencia del método de

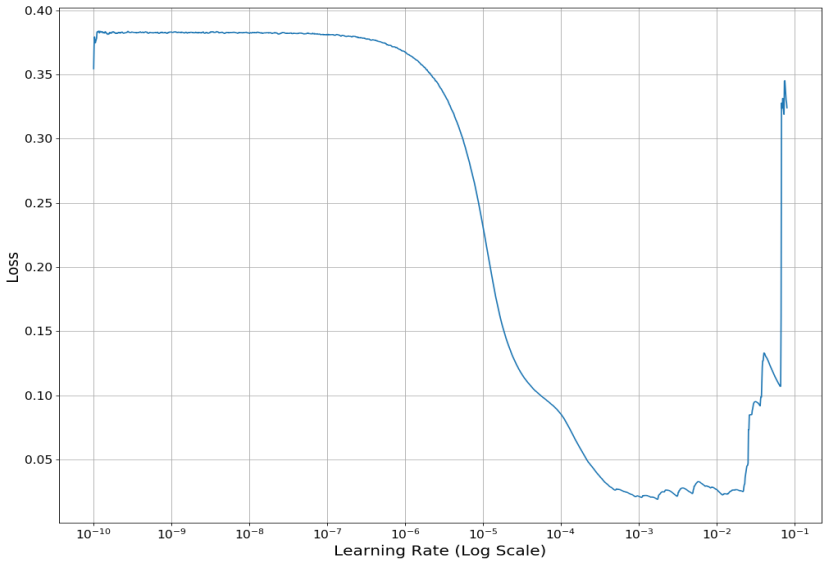

Figura 5.1: Método de Smith

Smith, iremos subiendo el learning rate exponencialmente por cada batch. Empezaremos con un learning rate de 10^{-10} hasta llegar a 1, comparando ECM contra el learning rate.
Como se puede observar en la Figura 5.1, el rango de learning rate óptimo se encuentra entre $5 * 10^{-6}$ y $5 * 10^{-3}$.

Luego vamos a proponer 3 métodos de decrecimiento del learning rate. Utilizaremos grid-search para encontrar los parámetros óptimos de los algoritmos de decrecimiento del learning rate. Tomando los hiperparámetros del Cuadro 5.3. En el Cuadro 5.4 se definen los intervalos sobre los cuales aplicaremos grid-search, usando como referencia el método de Smith. Dando como resultado el Cuadro 5.5.

Cuadro 5.4: Grid-Search

Parametros	Step Decay	Exponential Decay	Time-Based Decay
base_lr	$[10^{-2}, 10^{-4}]$	$[10^{-2}, 10^{-4}]$	$[10^{-2}, 10^{-4}]$
decay	[0.9, 0.95]	[0.007, 0.002]	[0.01, 8]
epoch_drop	5, 10, 20, 40, 50	-	-

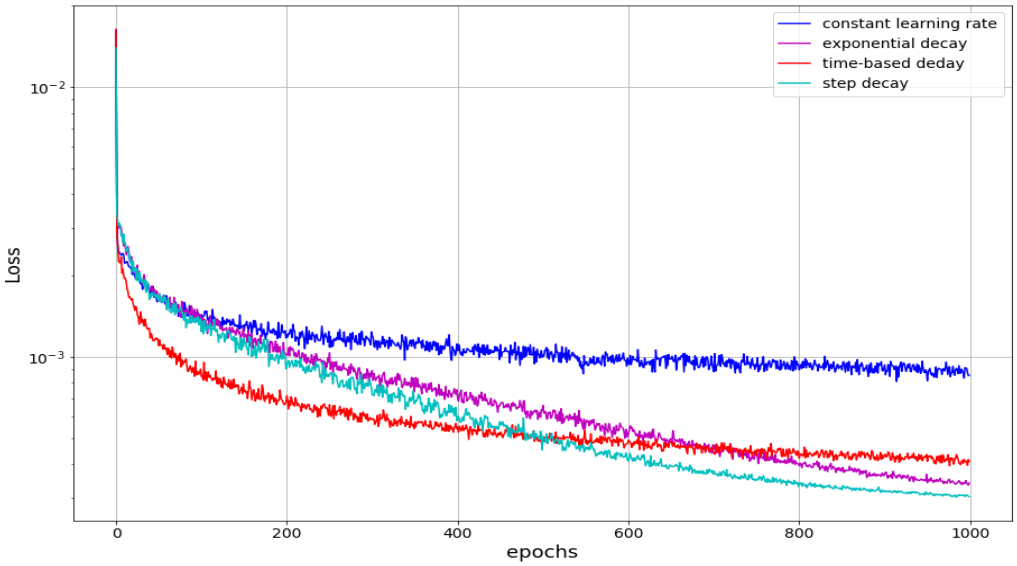

Figura 5.2: Learning Rate Decay

Cuadro 5.5: Parametros de los Algoritmos de Decrecimiento

Parametros	Step Decay	Exponential Decay	Time-Based Decay
base_lr	$5 * 10^{-4}$	0.0005	0.005
decay	0.9	0.002	0.875
epoch_drop	20	-	-

Observando los parámetros del Cuadro 5.5 entrenaremos la red por 1000 épocas ulizando los algoritmos de decrecimiento del learning rate y los hiperpárametros del Cuadro 5.3 antes mencionados. Los vamos a comparar mediante su ECM, para obtener el mejor algoritmo de decrecimiento. Como se puede observar en la Figura 5.2 Step Decay es el algoritmo que mejor resultado obtuvo.

Luego comparamos Step Decay con Cyclical Decay [11], donde el learning rate varia entre un máximo de $5 * 10^{-3}$ y un mínimo de $5 * 10^{-6}$, con un paso cada 8 epocas aproximadamente(ver Figura 2.6). Observando la Figura 5.3,

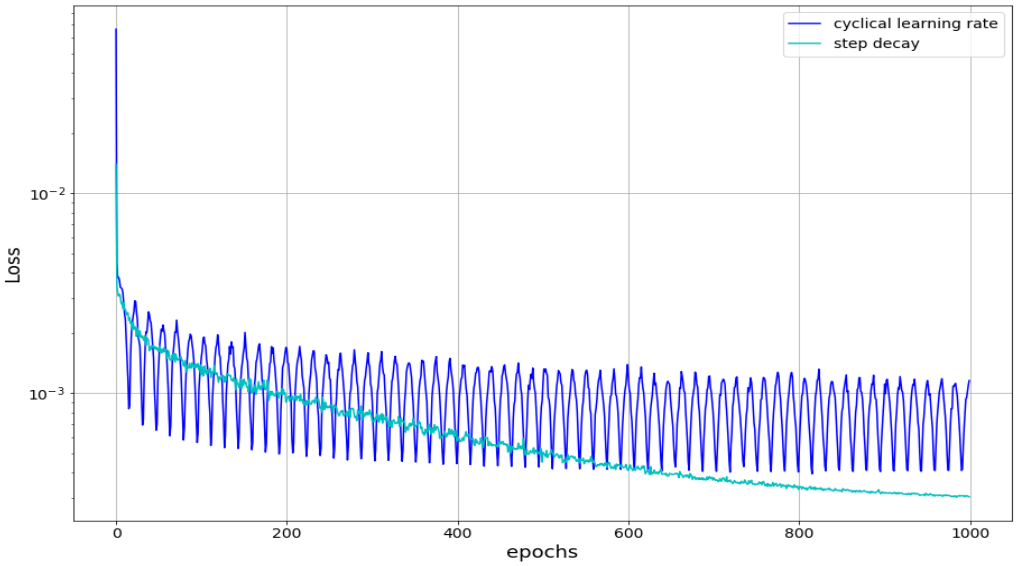

Figura 5.3: Cyclical LR vs Step Decay

podemos concluir que Step Decay da un mejor resultado.

5.4. Optimización

Notar que si sigma es grande, la derivada de la call respecto de sigma es muy pequeña, esto puedo ocasionar un problema de gradiente pronunciado. Por lo tanto se propone un aplanamiento del gradiente para manejar este problema. Primero, cada opción puede ser dividida entre el valor intrínseco y un valor de tiempo, luego sustraemos el valor intrínseco de la siguiente manera:

$$\tilde{C} = C - max(S_0 - Ke^{-r\tau}, 0)$$

El nuevo cálculo propone superar el problema, logrando reducir el gradiente pronunciado aplicando una transformación logarítmica sobre el valor de la opción [24]. En nuestro caso sería:

$$\frac{\tilde{C}}{K} = \frac{C}{K} - max(ratio - e^{-r\tau}, 0) \tag{5.1}$$

Capítulo 6

Resultados

En este capítulo vamos a analizar la efectividad de los modelos propuestos. El experimiento se realizó en CPU(Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz)

6.1. Primera Red

Ya tenemos los hiperparámetros óptimos del capítulo anterior, entonces vamos a definir la muestra de entrenamiento, validación y test. Crearemos 2 muestras, una amplia y una estrecha. En el cuadro 6.1 se define el ambiente de las muestras, y para generar la muestra se usa el algoritmo 5.1.

Cuadro 6.1: Hiperparametros de la Muestra

	Parametros	muestra amplia	muestra estrecha
Entrada	precio ratio(S_0/K)	[0.4, 1.6]	[0.5, 1.5]
	Tiempo de madurez(τ)	[0.2, 1.1]	[0.3, 0.95]
	volatilidad(σ)	[0.01, 1]	[0.02, 0.9]
	Tasa libre de riesgo(r)	[0.02, 0.1]	[0.03, 0.08]
Salida	Precio de Call(c/K)	(0, 0.9)	(0, 0.73)

Una vez obtenida la muestra, la entrada de la red será {c/K, S_0/K, r, τ} y la salida {σ}. Vamos a entrenar la red durante 1000 épocas utilizando la muestra amplia y los hiperparámetros del obtenidos capítulo anterior.
Una vez entrenada la red, en el Cuadro 6.2 y en las Figuras 6.1 y 6.2 se pueden observar los resultados.

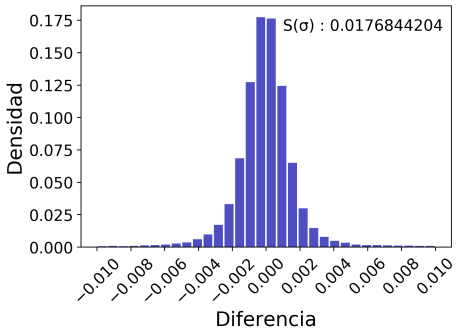

Figura 6.1: Predicción muestra amplia.

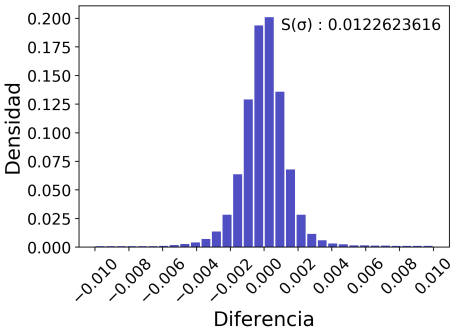

Figura 6.2: Predicción muestra estrecha.

Cuadro 6.2: Error Red

	ECM	EAM	EPAM	R^2
Muestra Amplia	$3{,}21e(-4)$	$6{,}09e(-3)$	9,30	0.996031
Muestra Estrecha	$1{,}47e(-4)$	$4{,}07e(-3)$	5,46	0.997714

6.2. Optimización

Ya tenemos los hiperparámetros óptimos del capítulo anterior, entonces vamos a definir la muestra de entrenamiento, validación y test. Crearemos 2 muestras, una amplia y una estrecha. A diferencia de la muestra de la sección anterior, en esta se va a modificar utilizando la ecuación 5.1 del cápitulo anterior. En el cuadro 6.3 se define el ambiente de la muestra, y para generar las muestras se usa el algoritmo 5.1.

Cuadro 6.3: Hiperparametros de la Muestra

	Parametros	muestra amplia	muestra estrecha
Entrada	precio ratio(S_0/K)	[0.4, 1.6]	[0.5, 1.5]
	Tiempo de madurez(τ)	[0.2, 1.1]	[0.3, 0.95]
	volatilidad(σ)	[0.01, 1]	[0.02, 0.9]
	Tasa libre de riesgo(r)	[0.02, 0.1]	[0.03, 0.08]
Salida	Precio de Call($\log(\tilde{C}/K)$)	[-16.12,-0.94]	[-16.12,-0.94]

Una vez obtenida la muestra, la entrada de la red será $\{\log(\tilde{C}/K)$, S_0/K, r, $\tau\}$ y la salida $\{\sigma\}$. Vamos a entrenar la red durante 1000 épocas.

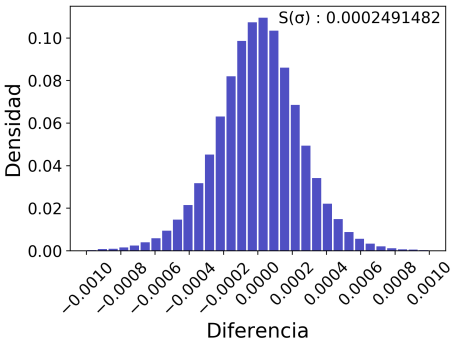

Figura 6.3: Predicción muestra amplia.

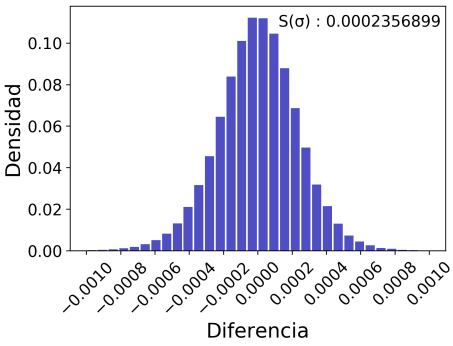

Figura 6.4: Predicción muestra estrecha.

Una vez entrenada la red, en el Cuadro 6.4 y en las Figuras 6.3 y 6.4 se pueden observar los resultados.

Cuadro 6.4: Error Red Optimizada

	ECM	EAM	EPAM	R^2
Muestra Amplia	$6{,}24e(-8)$	$1{,}92e(-4)$	0,059	0.999999104
Muestra Estrecha	$5{,}58e(-8)$	$1{,}84e(-4)$	0,062	0.999999020

6.2.1. Cambio en función de decrecimiento del Learning Rate

Observando la Figura 5.1 podemos concluir que el entrenamiento de la red con un learning rate menor a 10^{-7} prácticamente no actualizaría los pesos, entonces si queremos hacer mas épocas en el entrenamiento de la red, con el algoritmo que usamos en la sección anterior tendríamos en la época 3000 un learning rate aproximado de $1{,}87e^{-17}$, y en la época 1000 el learning rate aproximado sería de $2{,}65e^{-8}$. Una posible solución a este problema es modificar los hiperparámetros óptimos aumentando el *base_lr*, aumentar el *decay*, y aumentar el *epoch_drop*, para que el learnig rate descienda mas lentamente. Otra solución sería cuando el learning rate sea menor a un umbral, aumentar el *base_lr*. El umbral puede ser variable.
El algoritmo 6.1 muestra el método de decrecimiento de learning rate que utilizaremos:

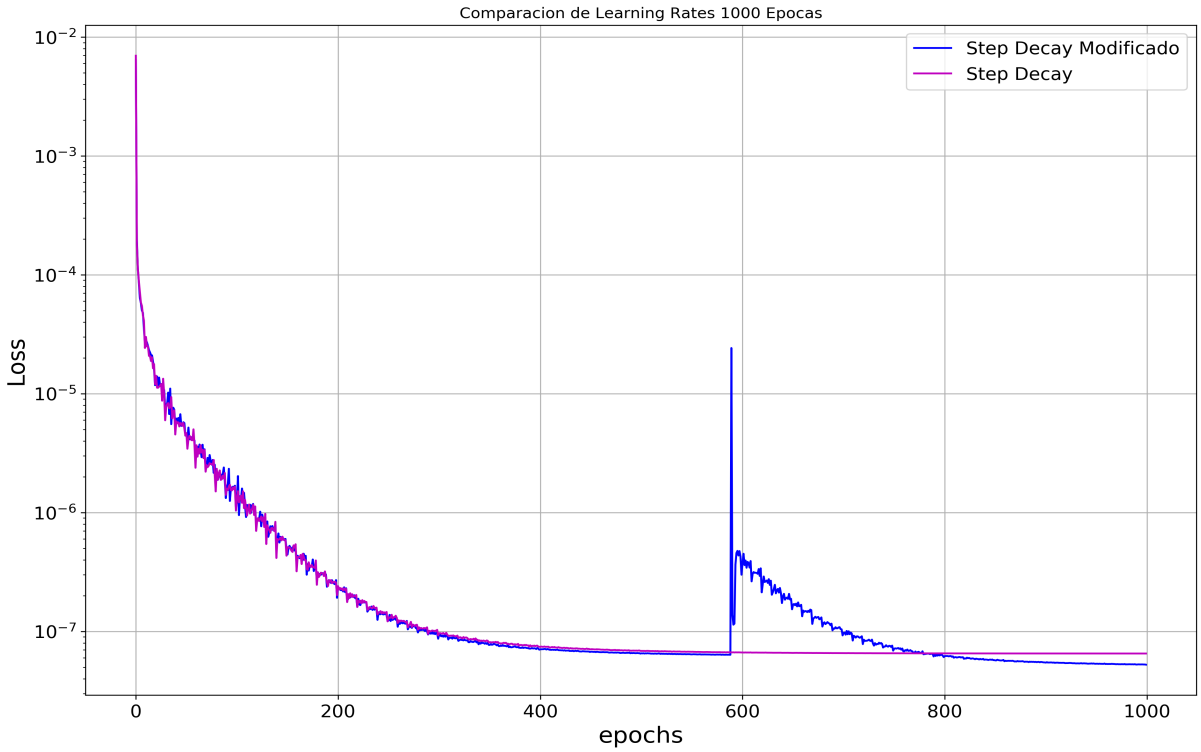

Figura 6.5: Step Decay Modificado

Algorithm 6.1: Step Decay Modificado

Input : epoch
Output: lrate

```
i := 0;
step := floor((1+epoch)/10);
lrate := 0.0005 * pow(0.9, step);
while lrate < 10^-(6+i) do
    lrate := 100*lrate;
    i = i + 0.5;
end
return;
```

La Figura 6.5 muestra la comparación entre el step decay usado anteriormente y el step decay modificado.
Las Figuras 6.6 y 6.7 muestan el learning rate resultante con los diferentes algoritmos.

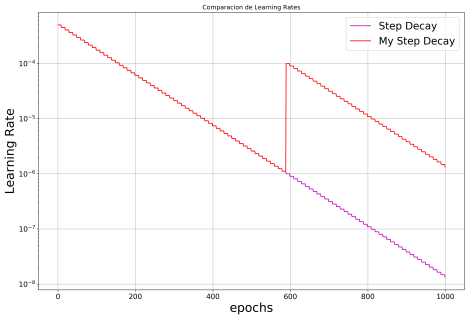

Figura 6.6: Learning Rate sobre 1000 epocas.

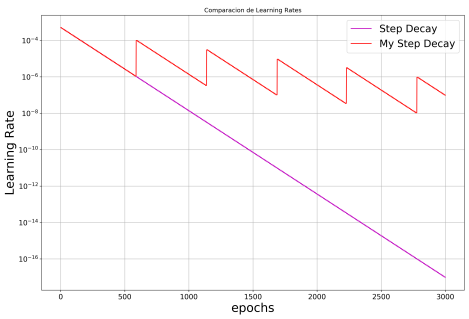

Figura 6.7: Learning Rate sobre 3000 epocas.

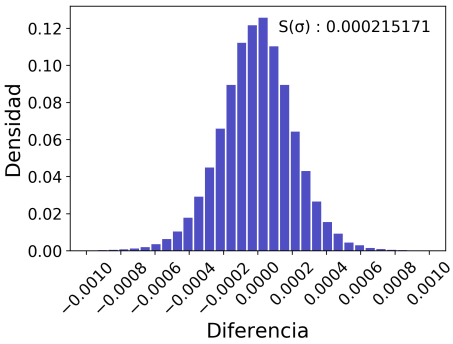

Figura 6.8: Predicción muestra amplia.

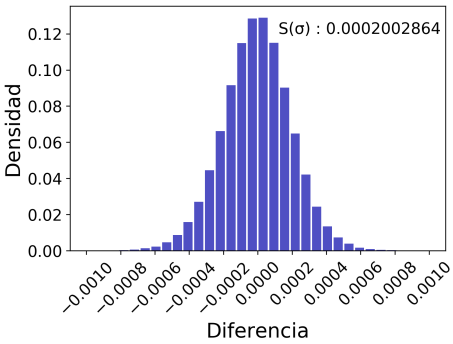

Figura 6.9: Predicción muestra estrecha.

Luego vamos a entrenar la red durante 3000 epocas usando step decay modificado. En las Figuras 6.8, 6.9, y en el Cuadro 6.5 se muestran los resultados.

Cuadro 6.5: Error Red Optimizada

	ECM	EAM	EPAM	R^2
Muestra Amplia	$4{,}67e(-8)$	$1{,}66e(-4)$	0,0515	0.999999329
Muestra Estrecha	$4{,}02e(-8)$	$1{,}57e(-4)$	0,0526	0.999999295

6.3. Métodos Numéricos

En el capítulo 4 vimos que hay casos en los que no se puede aplicar los métodos numéricos, exceptuando esos casos, vamos a analizar su precisión. Ademas en el capítulo 4 se vió que lo máximo que podemos aspirar es un error absoluto de 2^{-56}. Ahora vamos a analizar si en la práctica, la tolerancia de los métodos corresponden con los resultados obtenidos de aplicar los mismos. En el método de Brent utilizamos el de la librería Scipy. [25]
En el cuadro 6.6 se muestran los resultados.

Cuadro 6.6: Error brent y Bisección

	ECM	EAM	EPAM	R^2
Bisección	$1{,}10e(-8)$	$2{,}89e(-6)$	0,00469	0.99999986
Brent	$8{,}17e(-6)$	$9{,}18e(-5)$	0,35894	0.99989551

6.3.1. Peso de la Volatilidad sobre el precio de la opción

Observando el Cuadro 6.6, el error absoluto medio del método de bisección es $2{,}89e(-6)$ y de brent es $9{,}18e(-5)$, siendo la tolerancia $2e(-56)$, esto es por el problema de presición y el impacto que tiene la volatilidad sobre el precio de la opción utilizando la fórmuala de Black Scholes(definida en el sección 2.1.10). En la Figura 6.10, que compara el precio de la opción con respecto a la variación de volatilidad, utilizando diferentes ratios, se puede observar que mientras más diferencia hay entre S_0 y K la volatilidad tiene menos impacto sobre el precio en especial en los casos OTM y en las volatilidades cercanas a 0. Luego en la Figura 6.11, que compara el precio de la opción con respecto a la variación de volatilidad, utilizando diferentes tasas de interés, se observa practicamente un desplazamiento. Por último en la Figura 6.12, que compara el precio de la opción con respecto a la variación de volatilidad, utilizando diferentes tiempos de madurez, se puede observa que mientras los contratos sean más largos la volatilidad tiene un mayor impacto sobre el precio.

Ahora bien observando los gráficos podemos concluir que la volatilidad tiene menos impacto en los contratos cortos(en nuestro ambiente), en OTM y con volatilidades menores a 0.2, pero en la practica los casos con mas error son ITM con $\Phi_1(\sigma)$ y $\Phi_2(\sigma)$ cercanos a 1, esto se debe a la densidad de puntos flotates [26]. Esto es la cantidad de números representables en un rango. El

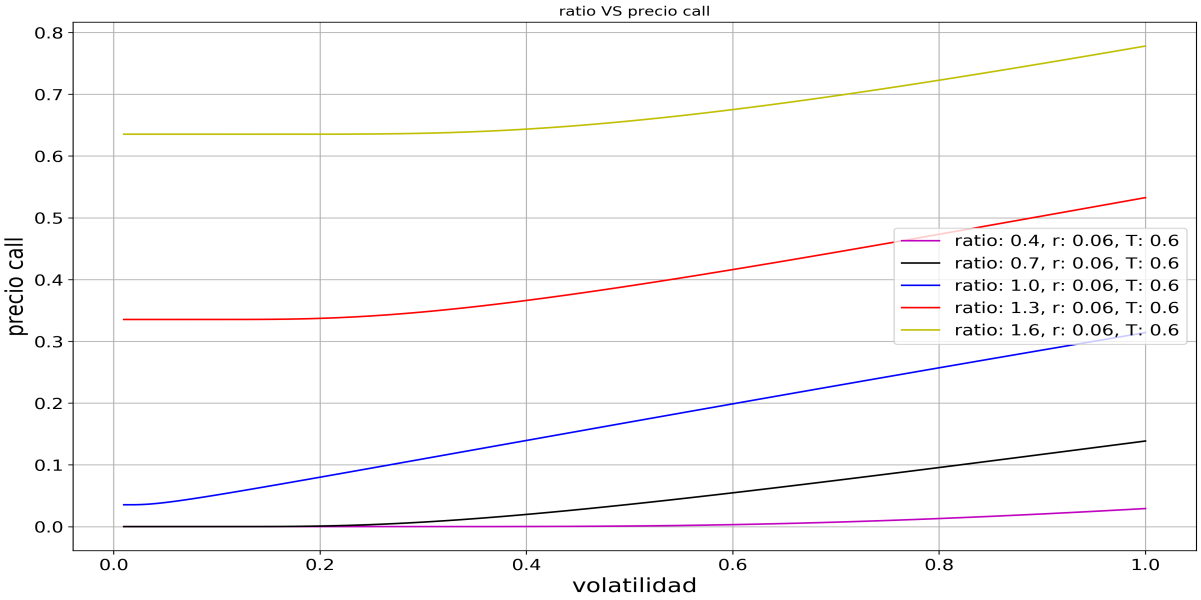

Figura 6.10: Volatilidad implícita vs precio call con distiontos ratios

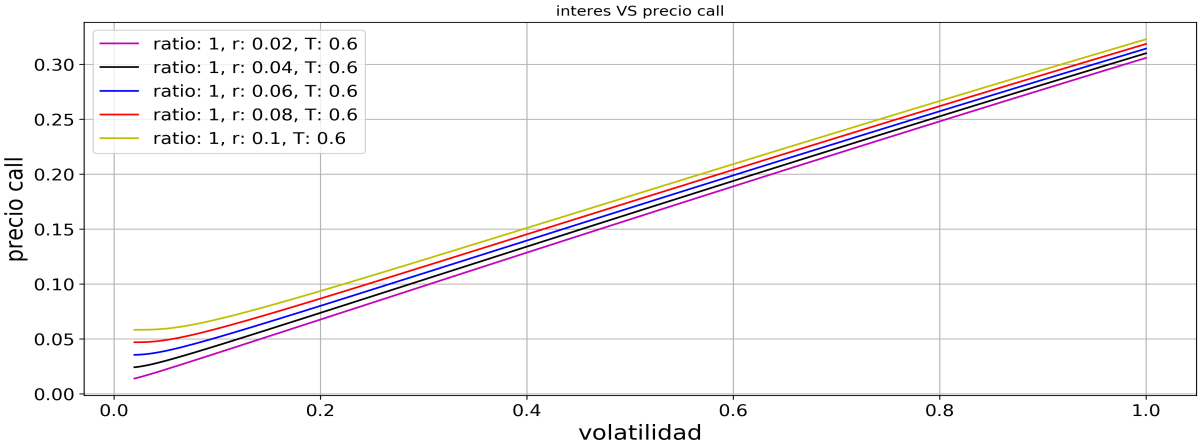

Figura 6.11: Volatilidad implícita vs precio call con distiontas tasas de interés

formato se escribe con un significando que tiene un bit entero implícito de valor 1 (excepto para los números especiales). Con los 52 bits de la mantisa, la precisión total es por lo tanto de 53 bits (es decir de $53log_{10}(2) \approx 15{,}955$ que se redondea a 16 dígitos decimales). El exponente de este formato está sesgado o desplazado en 1023 unidades, ya que como el máximo valor representado

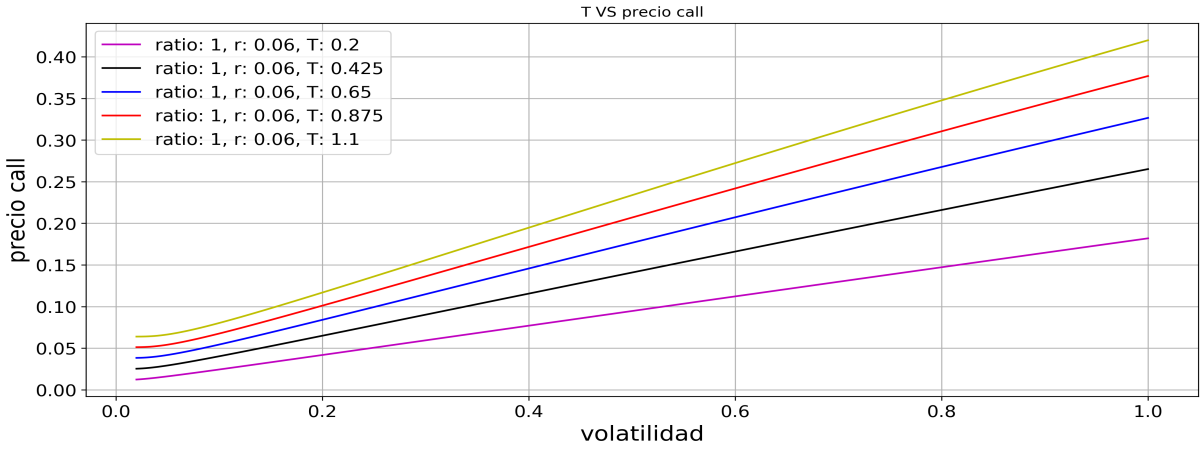

Figura 6.12: Volatilidad implícita vs precio call con distiontos tiempos de madurez

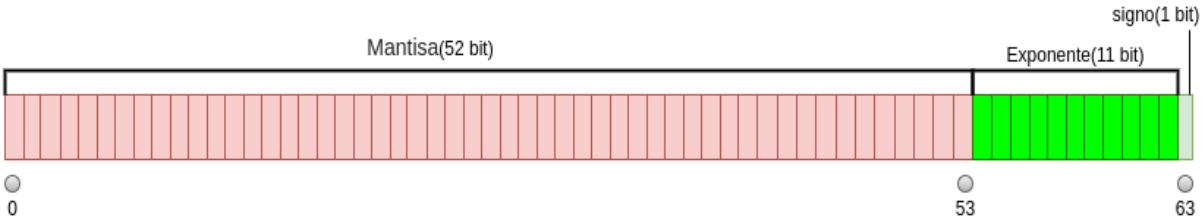

Figura 6.13: Estructura de un número en formato de coma flotante de doble precisión

por 11 bits es $2^{11} - 1 = 2047$, es la mitad de este rango la que representa exponentes positivos y la otra, exponentes negativos. Observar Figura 6.13.

Lo cual esto nos da que hay 2^{52} numeros entre 2^n y 2^{n+1} para todo n entero en [-1023, 1022].
Luego como el ancho de 11 bits del exponente permite la representación de números en el rango comprendido entre 2^{-1023} y 2^{1023} (10^{-308} y 10^{308}). Esto nos permite concluir q hay la misma cantidad de numeros representables entre 10^{-308} y 10^{-307} que en 0,1 y 1. Por lo tanto los $\Phi_1(\sigma)$ y $\Phi_2(\sigma)$ cercanos a 0 en OTM van a tener mas precisión aunque el peso de la volatilidad sobre el precio de la opción sea menor.

6.4. Tiempo de Ejecución y Robustez

Una vez entrenada la red (la busqueda de hiperpámetros fue 10.08 horas, la busqueda del algoritmo de learning rate fue 11.72 horas y entrenamiento de la red 3000 epocas fue 8.83 horas), vamos a observar el tiempo ejecución de evaluación de la misma sobre una muestra de 10000. Ademas el tiempo de ejecución de los métodos de brent y bisección sobre la misma muestra en una CPU(Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz).(CAMBIAR)
Tambien analizaremos su robustez, esto sería, si para todo elemento de la muesta el modelo genera una salida.
En el Cuadro 6.8 se observan el tiempo de ejecución y la Robustez.

Cuadro 6.7: Robustez y Tiempo de Ejecución

	Red Neuronal	Método de Brent	Método de Bisección
Tiempo de Ejecución(segundos)	7,34	157,05	318,31
Robustez	Si	No	No

Capítulo 7

Conclusiones

En este trabajo se presentó la importancia del calculo de la volatilidad implícita. Ademas se propuso una manera diferente del cálculo que usualmente se utiliza, usando redes neuronales feedforward.
Si bien el método de bisección estima mejor la volatilidad implícita, no es una mejora significativa con respecto a la red neuronal o al método de brent, sin embargo el tiempo de ejecución la red neuronal es 21.12 veces mas rápido que el método de Brent y 43.37 veces mas rápido que el método de bisección.
En caso de tener datos atípicos, por ejemplo precio de strike tres veces mas grande que el precio del subyancente, o tasas libre de riesgo superiores al 20 %, o contratos muy largos, en esos casos es conveniente usar los métodos numéricos, ya que la red no va a estimar bien, porque la red no entrenó con esos datos. Sin embargo hay casos en los que los métodos numéricos no pueden ser aplicados.
En caso que la presición sea imprescindible, entonces se utilizará el método de bisección.

Dentro de las posibilidades de continuar con el trabajo tenemos, uso de GPU en la implementacón de los modelos matématicos propuestos, presentar soluciones para el problema numérico del capítulo 4.6.
En este trabajo se planteo el cálculo de la volatilidad implícita mediante opciones call europeas, ademas se podrían proponer otros modelos obtener la volatilidad impícita mediante puts europeas.

Bibliografía

[1] Patricia Kisbye: *Modelos Matemáticos en Finanzas Cuantitativas*, (2019)

[2] John C. Hull: *Options, Futures, and Other Derivatives - 7th edition*, 381-389 (2009).

[3] ICAP: *The Future of the OTC Markets*, (2008) `https://thehill.com/sites/default/files/ICAP_TheFutureoftheOTCMarkets_0.pdf`

[4] Jay L. Devore: *Probabilidad y Estadística para Ingeniería y Ciencias - Séptima Edición*, 144-148 (2008).

[5] Ward Cheney y David Kincaid: *Métodos numéricos y computación - Sexta Edición*, 76-82(2011)

[6] Ward Cheney y David Kincaid: *Métodos numéricos y computación - Sexta Edición*, 111-112(2011)

[7] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery: *NUMERICAL RECIPES The Art of Scientific Computing - Third Edition*, 454-456(2007)

[8] Jason Brownlee: *Deep Learning With Python*, 109-112(2016)

[9] Wei Di, Anurag Bhardwaj, Jianing Wei: *Deep Learning Essentials*, 61-62(2018)

[10] Wei Di, Anurag Bhardwaj, Jianing Wei: *Deep Learning Essentials*, 219(2018)

[11] Leslie N. Smith: *Cyclical Learning Rates for Training Neural Networks*, (2015) `https://arxiv.org/abs/1506.01186`

[12] Payam Refaeilzadeh, Lei Tang, Huan Liu: *Cross-Validation* (2008) `http://leitang.net/papers/ency-cross-validation.pdf`

[13] Tom M. Mitchell: *Machine Learning*, 86-88(1997)

[14] John A. Hertz, Anders S. Krogh, Richard G. Palmer: *Introduction To The Theory Of Neural Computation*, 90-92(1991)

[15] Prosper Lamothe Fernández, Miguel Pérez Somalo: *Opciones Financieras y Productos Estructurados*, 319-336(2003)

[16] Jay Kaeppel: *The Option Trader's Guide to Probability, Volatility, And Timing*, (2002)

[17] Distribución Normal Acumulada: `https://docs.scipy.org/doc/scipy-0.16.0/reference/generated/scipy.stats.norm.html`

[18] Rene H Garcia, Ramazan Gencay: *Pricing and hedging derivative securities with neural networks and a homogeneity hint*, (2000)

[19] Función de Activación: `https://keras.io/activations/`

[20] Inicialización de Pesos: `https://keras.io/initializers/`

[21] Optimizadores del Descenso del Gradiente: `https://keras.io/optimizers/`

[22] Función de Error: `https://keras.io/losses/`

[23] Modelo Secuencial: `https://keras.io/api/models/sequential/`

[24] Shuaiqiang Liu , Cornelis W. Oosterlee, Sander M.Bohte *Pricing options and computing implied volatilities using neural networks*, 14-15(2018)

[25] Método de Brent: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.brentq.html`

[26] Densidad Punto Flotante: `https://web.archive.org/web/20160806053349/http://www.csee.umbc.edu/~tsimo1/CMSC455/IEEE-754-2008.pdf`