Data&Analytics
INNOVACIÓN Y TECNOLOGÍA

Data&Analytics
INNOVACIÓN Y TECNOLOGÍA

Fundamentos de Deep Learning

M.Sc. Angelo Jonathan Diaz Soto

2025

Data Science – Business Intelligence – Big Data – Machine Learning – Artificial Intelligence – Innovation and Technology
(+51) 976 760 803
www.datayanalytics.com info@datayanalytics.com

Índice

- Gradiente Descendente
- Perceptron Multicapa
- Redes Neuronales
- Redes *Feedforward* y *Recurrent Neural Networks*
- Pesos, parámetros sesgo y activación
- Funciones de activación: Sigmoidal, ReLu y Softmax
- Gradiente descendiente y Backpropagation

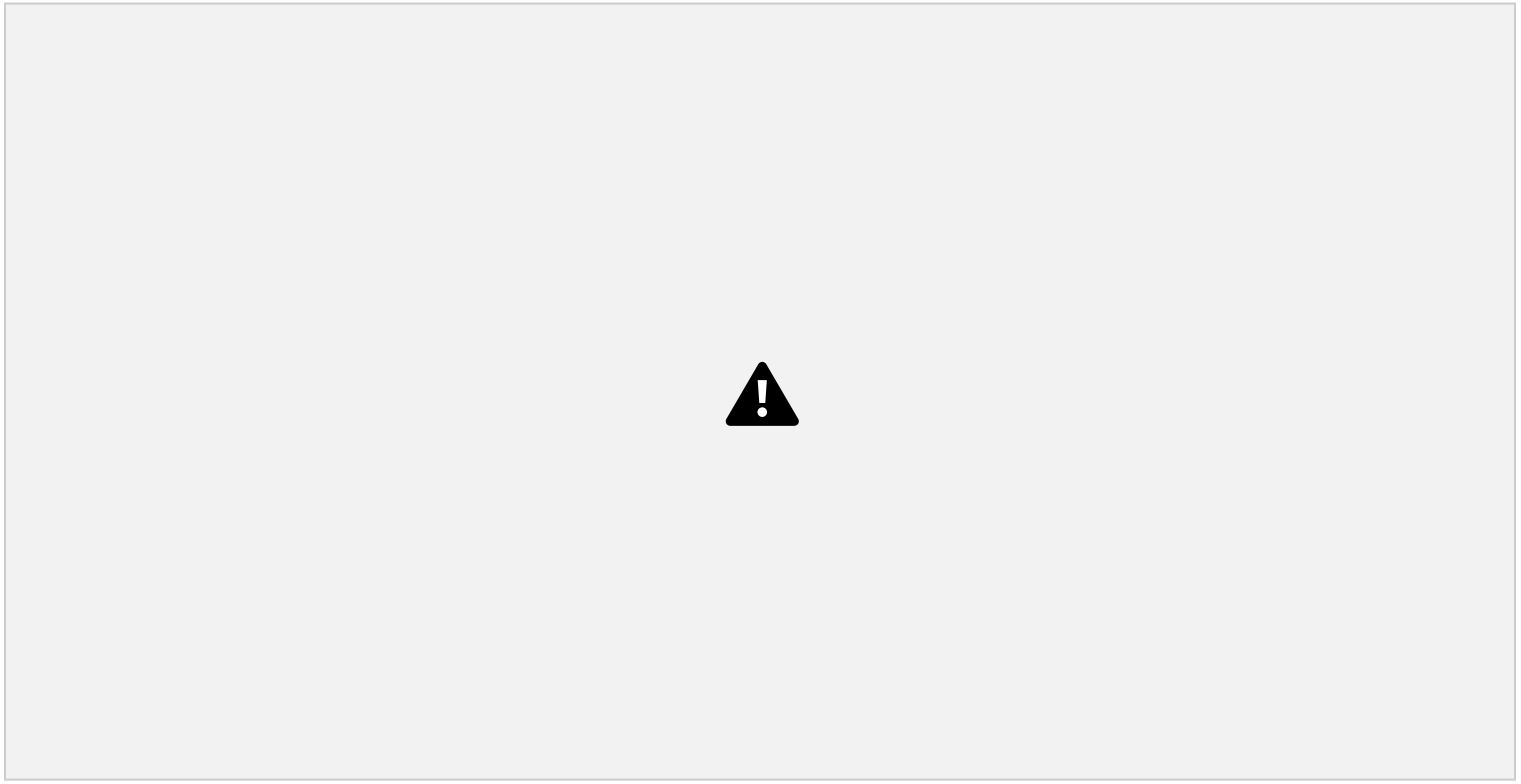
(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

Data&Analytics
INNOVACIÓN Y TECNOLOGÍA

1.Red neuronal artificial

surgen inicialmente
simular el
de las redes neuronales

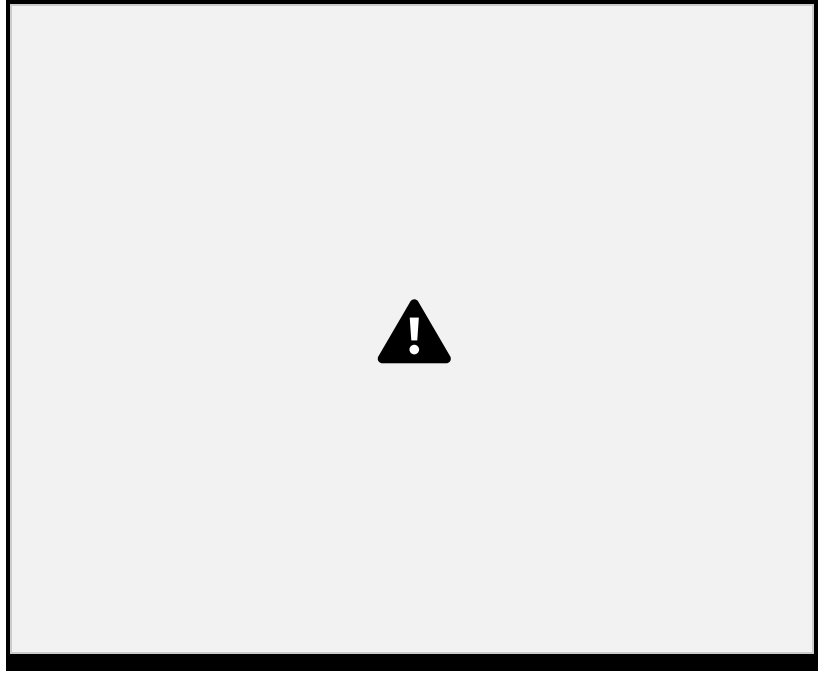
Las Rns artificiales
como un modo de
comportamiento
naturales.

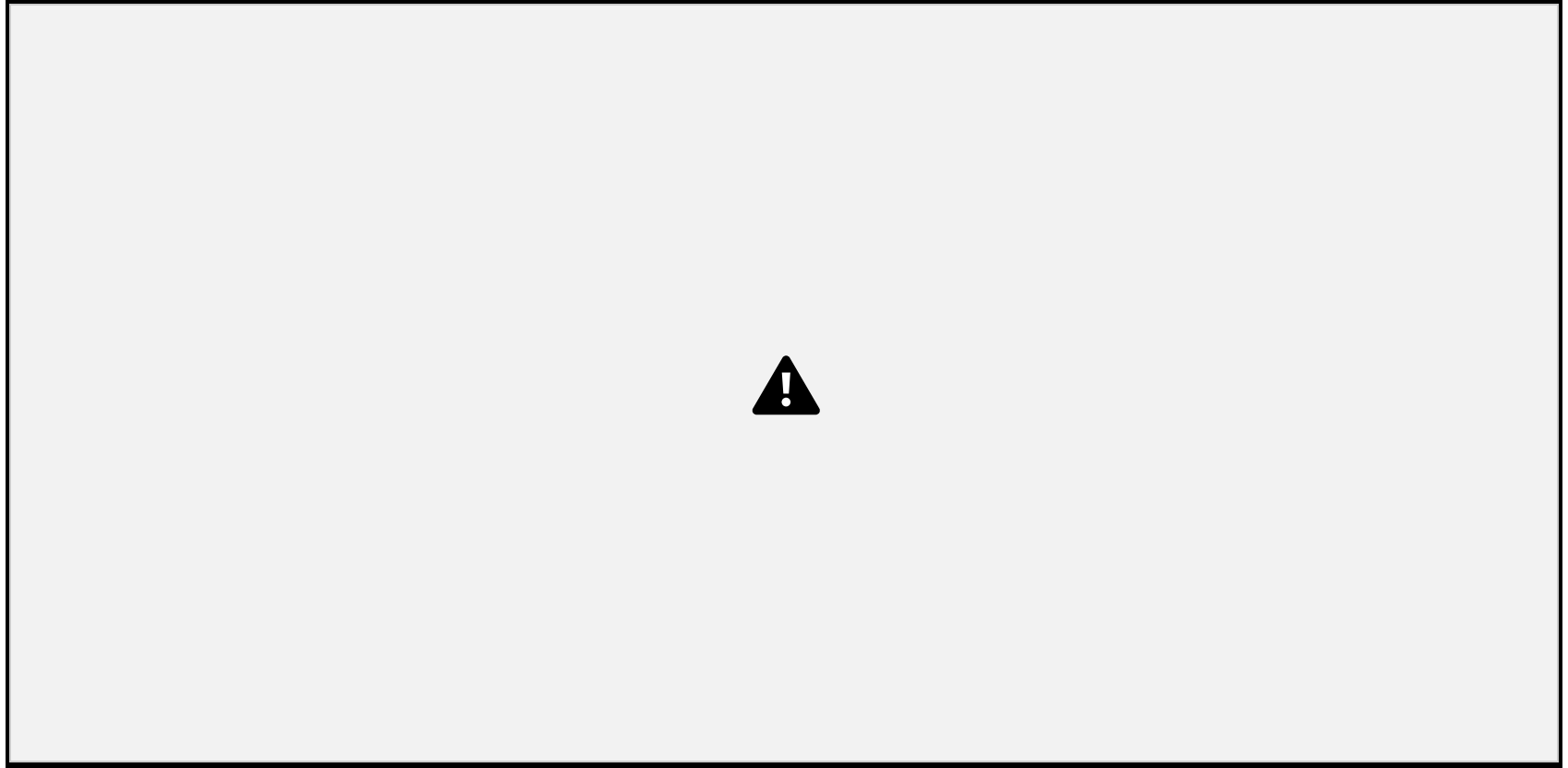


(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

2.Red neuronal

Red formada por varias capas de neuronas artificiales interconectadas.

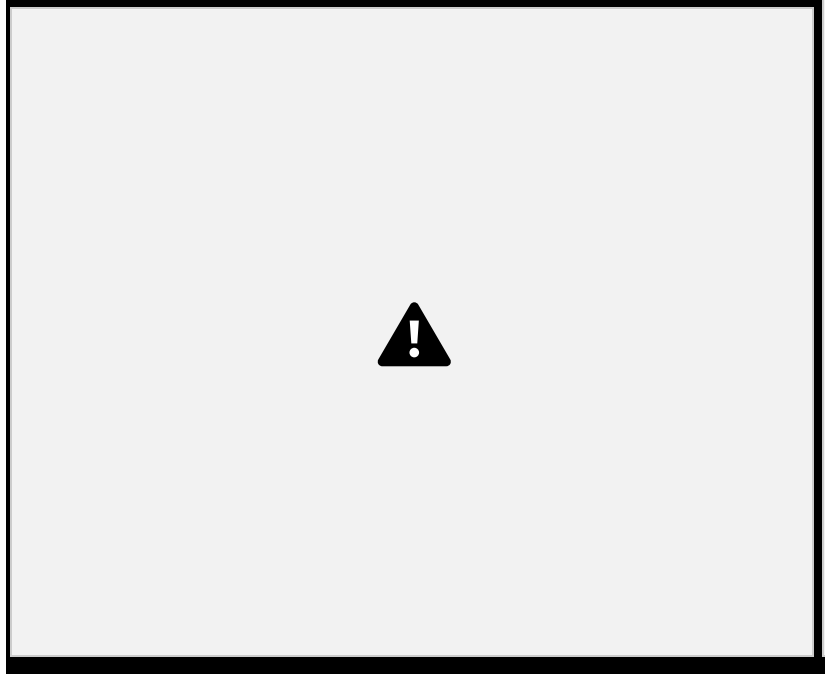


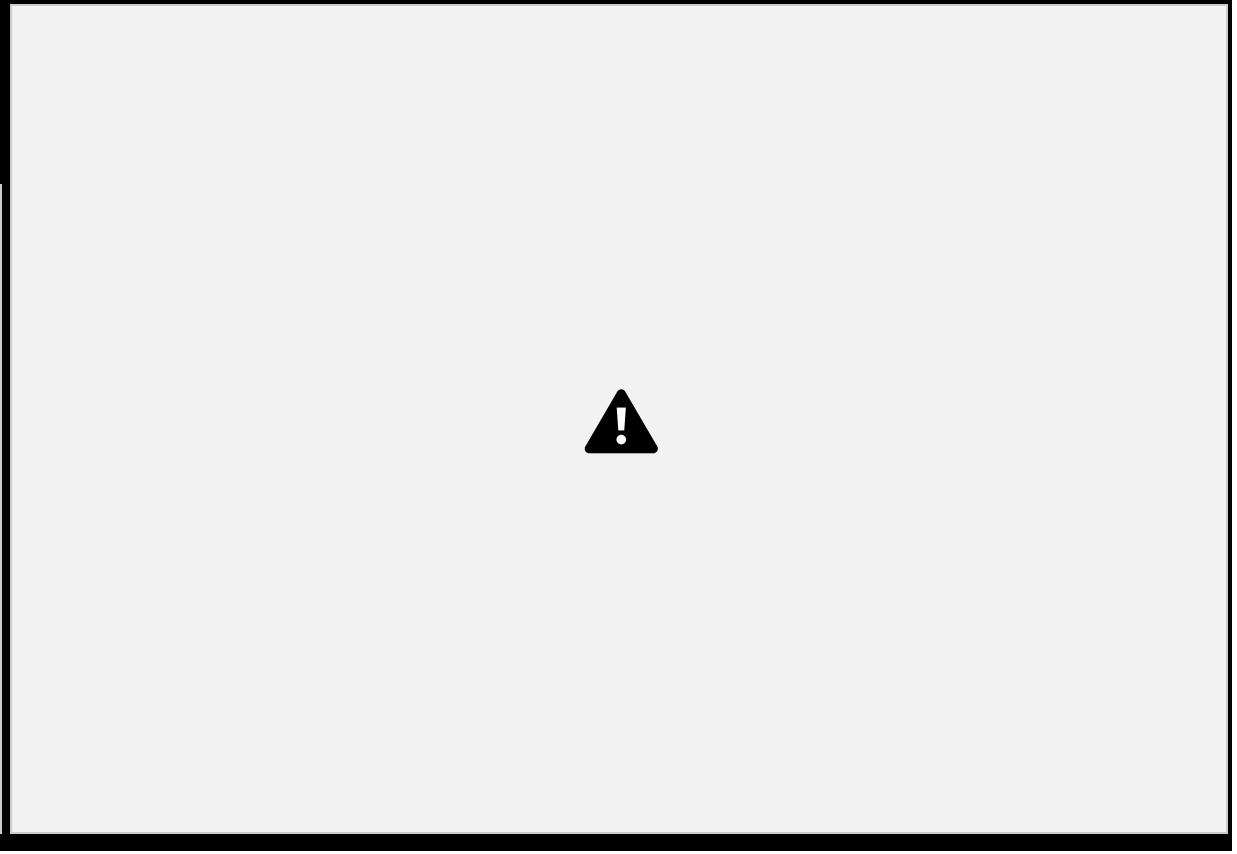


(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

2.Red neuronal

Red formada por varias capas de neuronas artificiales interconectadas.

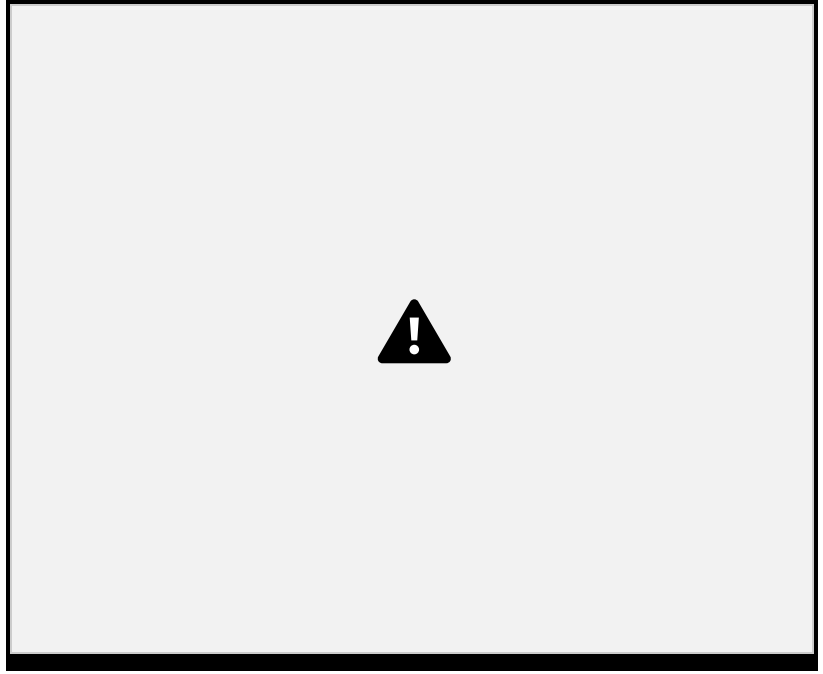


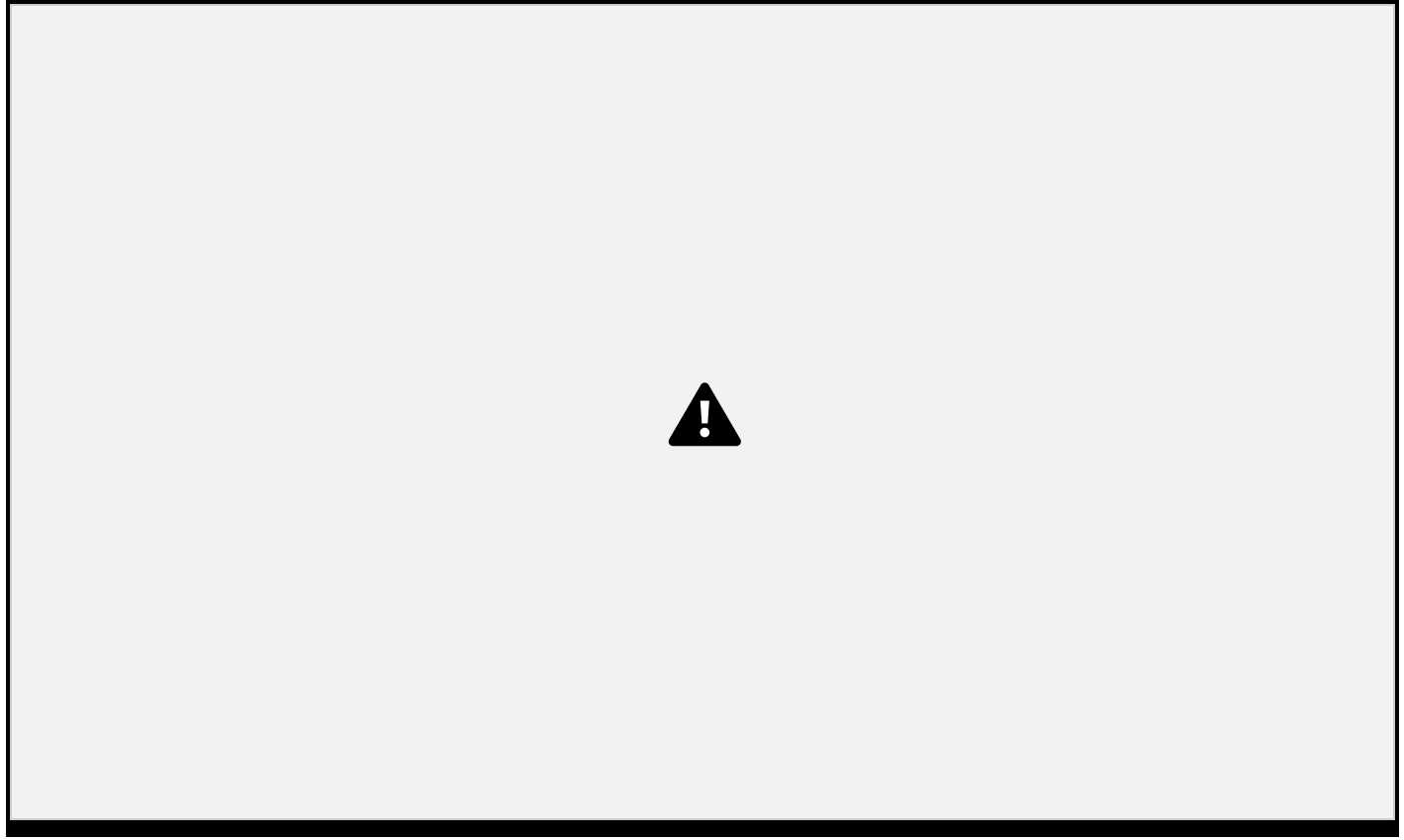


(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

2.Red neuronal

Red formada por varias capas de neuronas artificiales interconectadas.

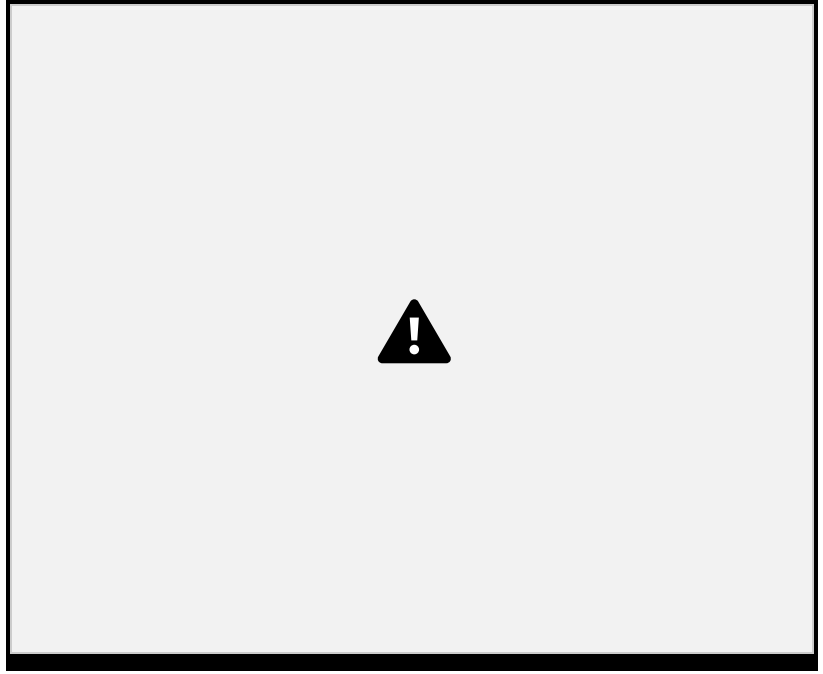


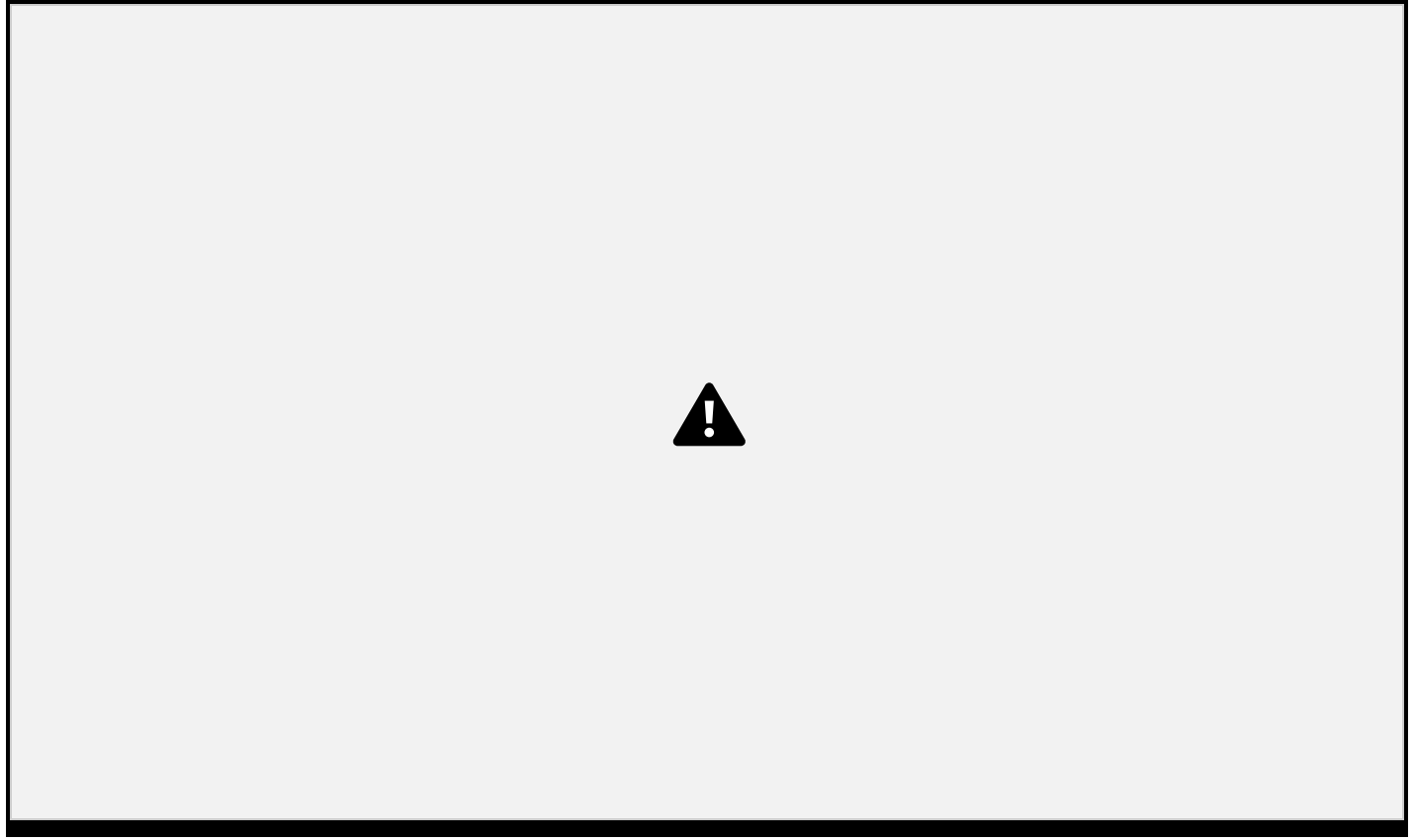


(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

2.Red neuronal

Red formada por varias capas de neuronas artificiales interconectadas.

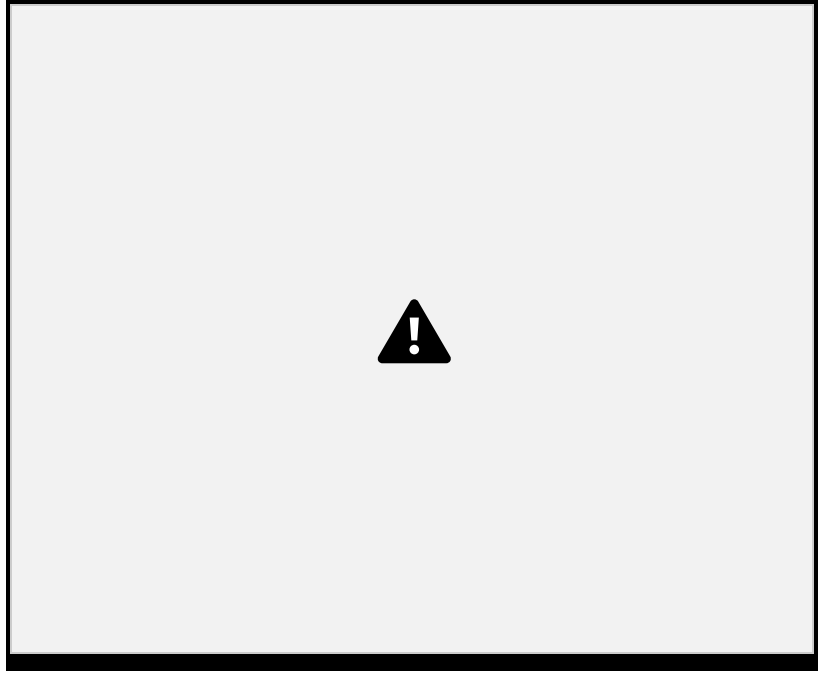


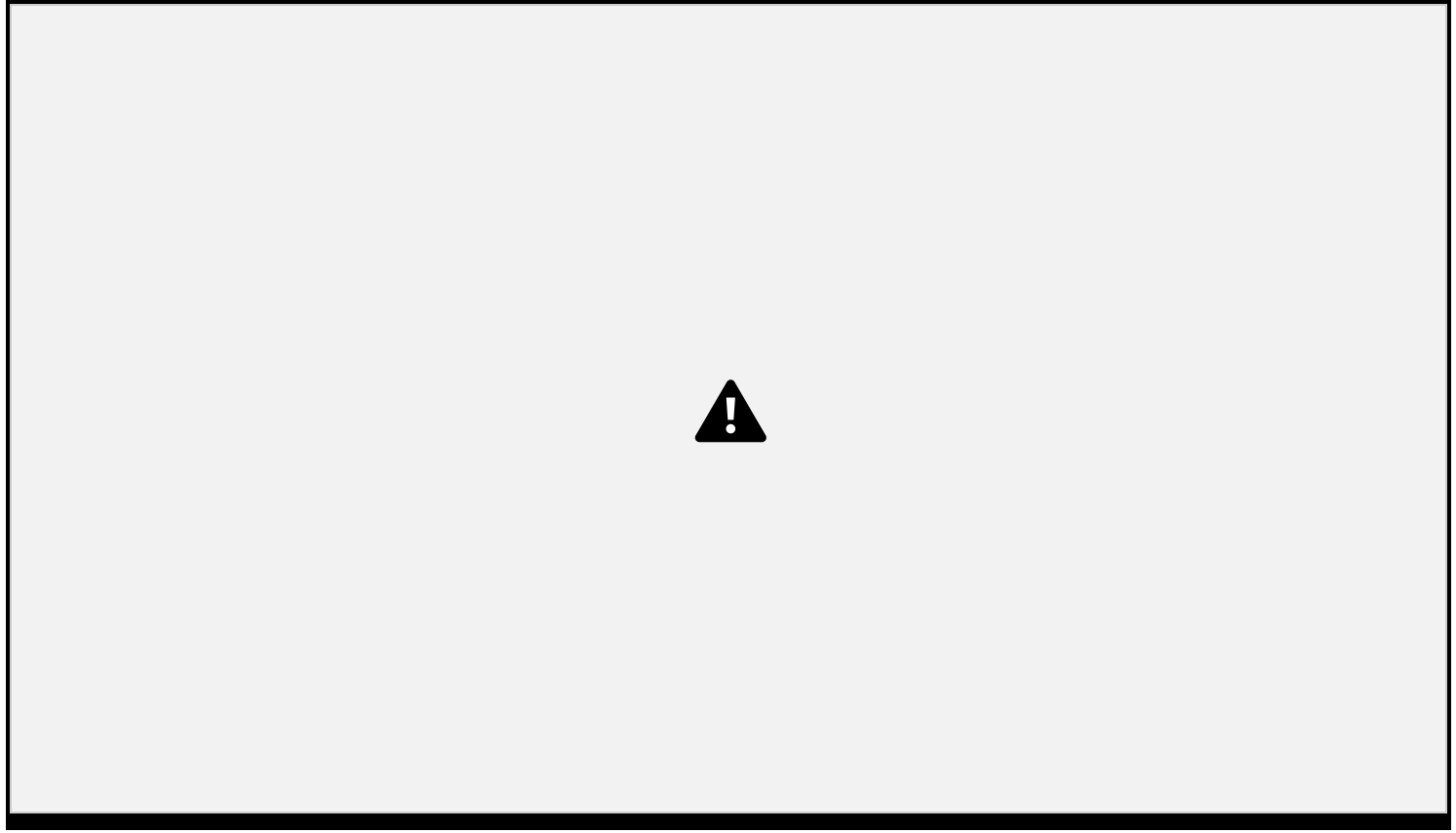


(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

2.Red neuronal

Red formada por varias capas de neuronas artificiales interconectadas.

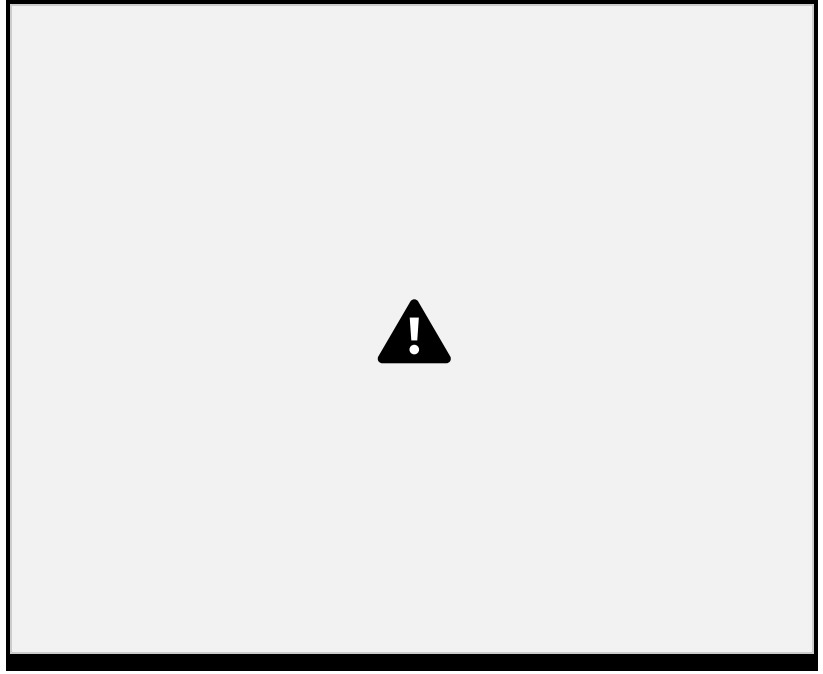


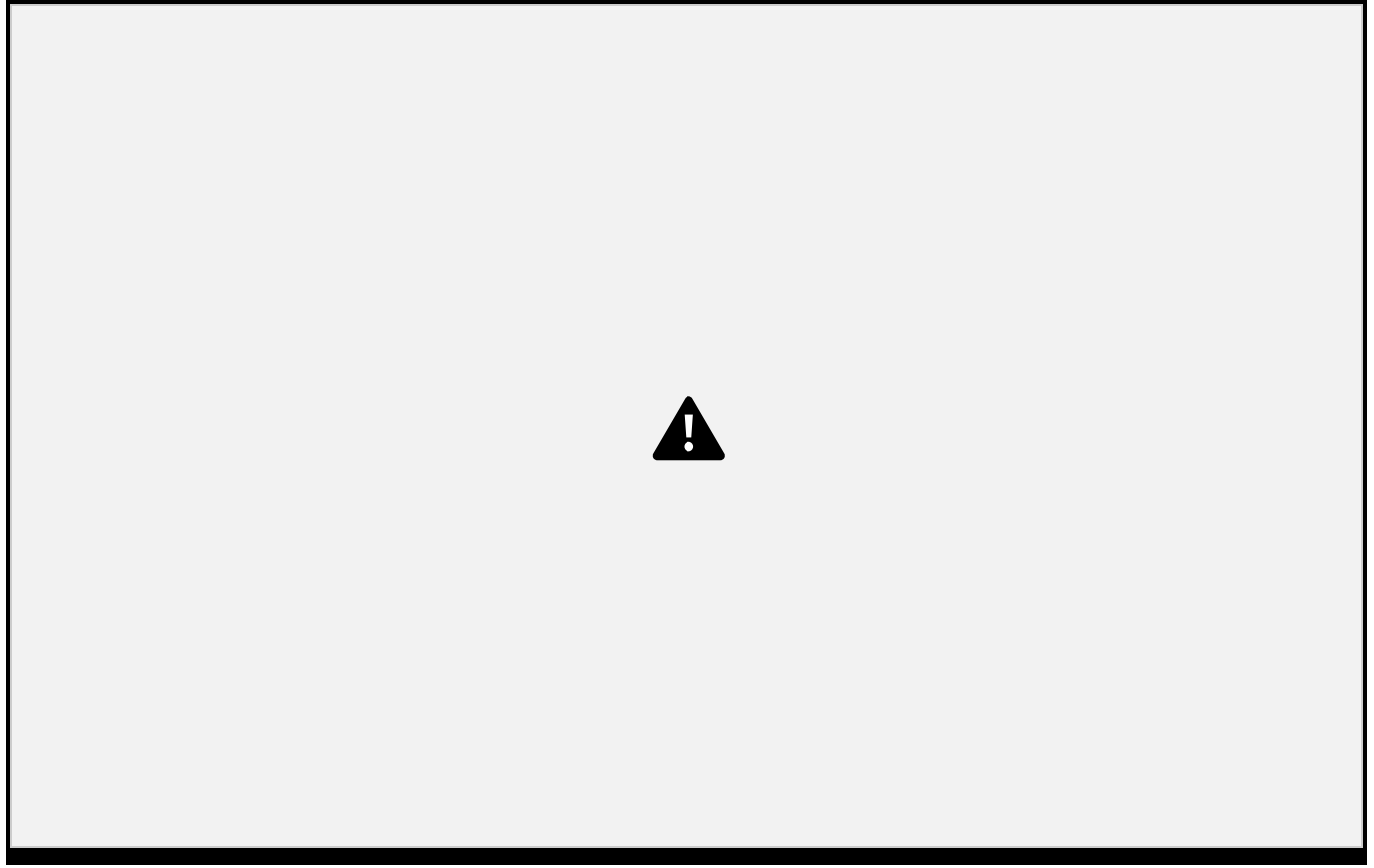


(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

2.Red neuronal

Red formada por varias capas de neuronas artificiales interconectadas.

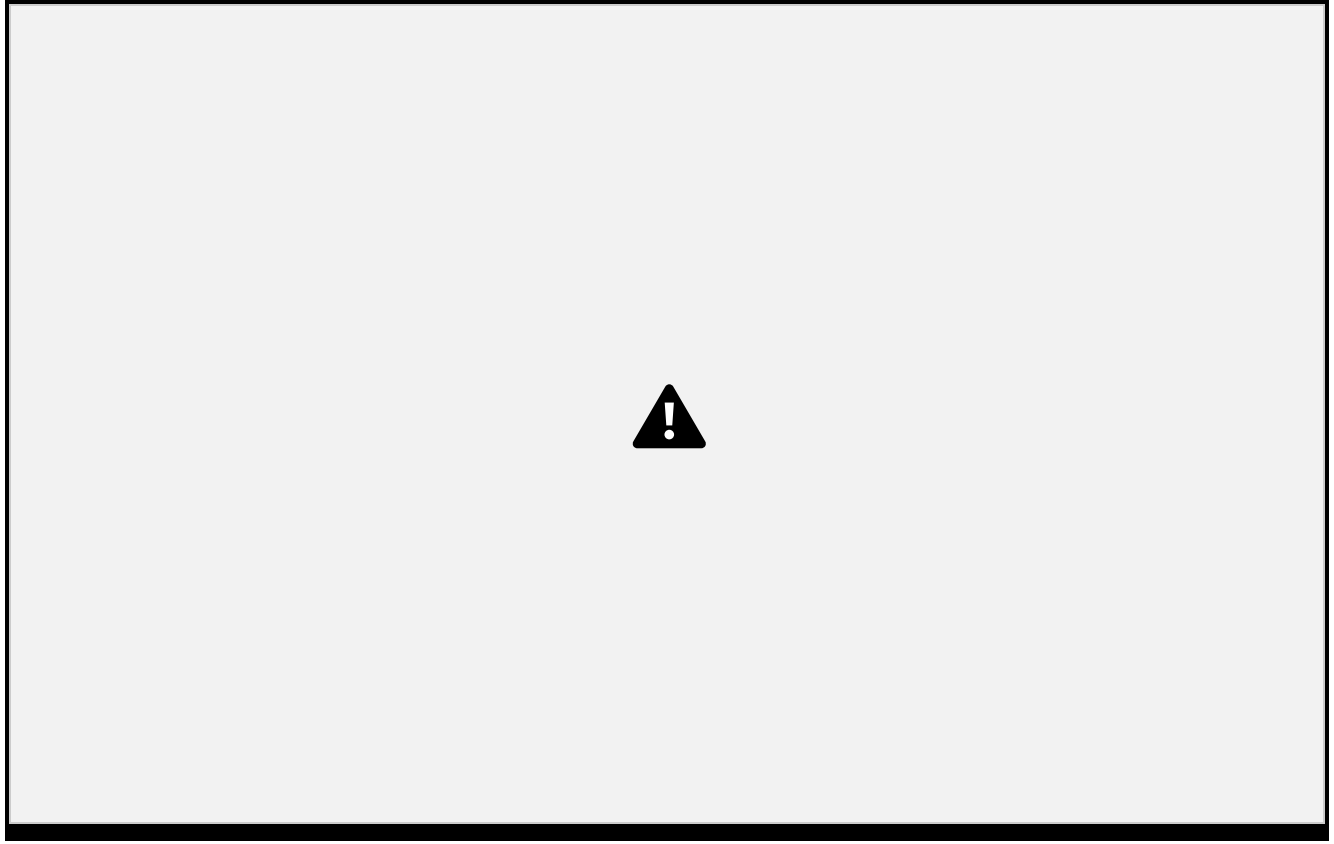


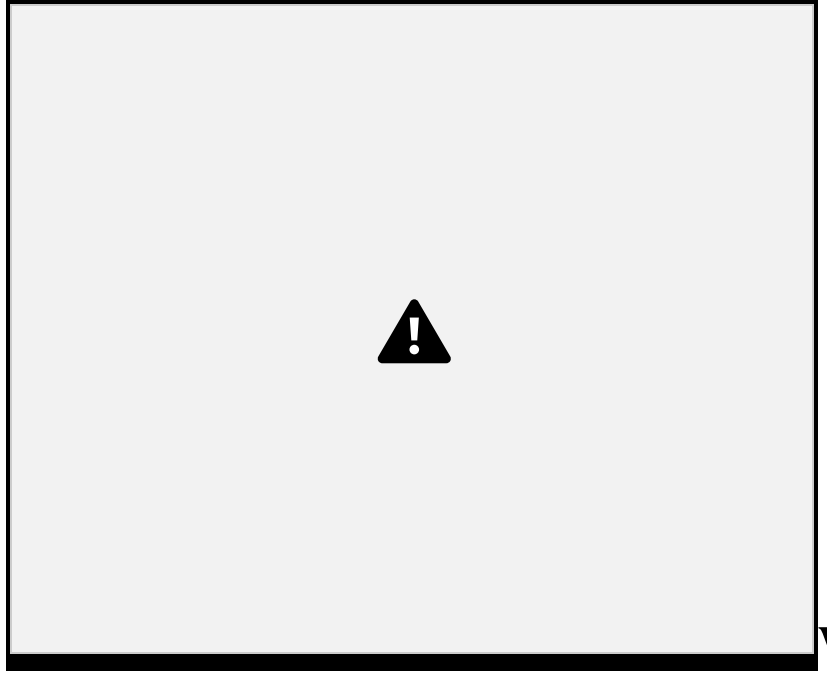


(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

2.Red neuronal

Red formada por varias capas de neuronas artificiales interconectadas.





$w^{(2)}$

(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

2.Red neuronal

Red formada por varias capas de

nterconectadas.

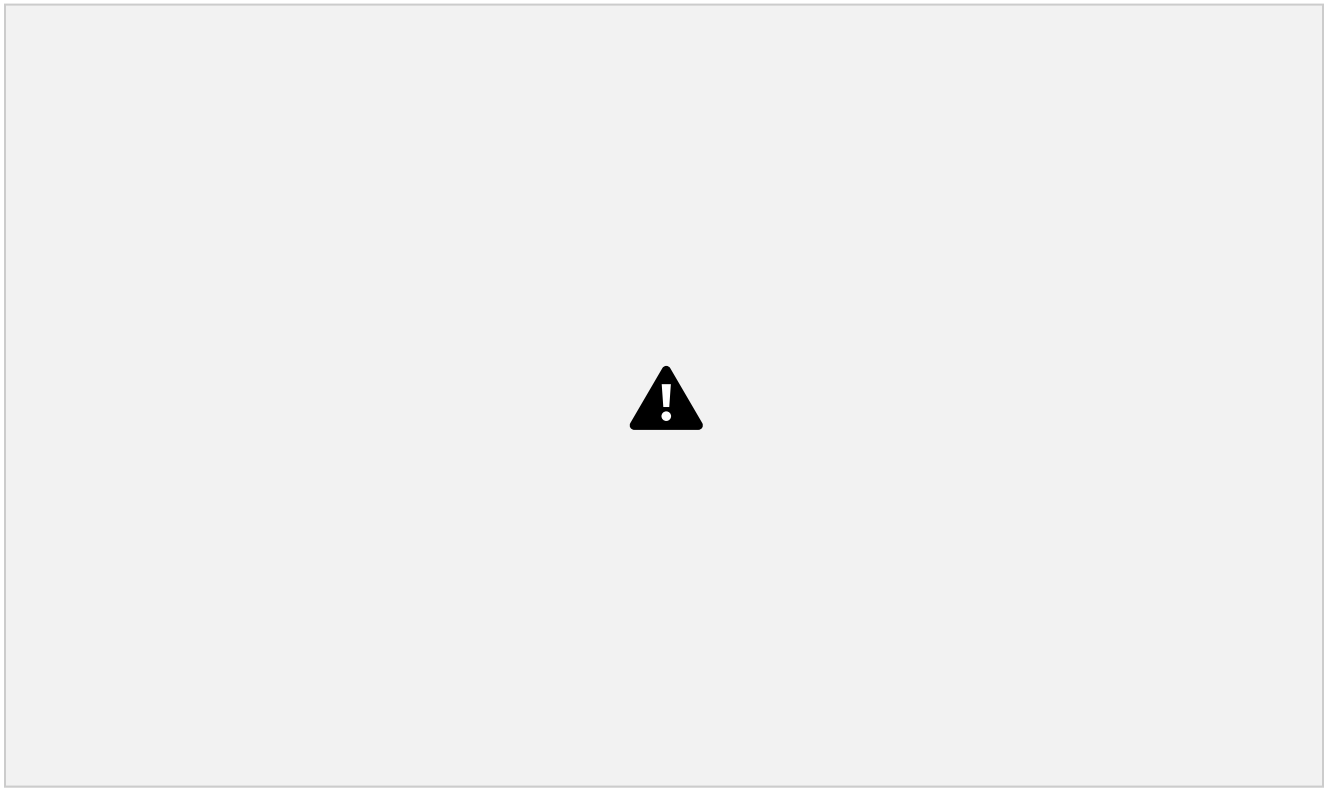
$w^{(1)}$

$$w^{(2)}$$

(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

3.Multilayer Perceptrón

Hay muchos tipos de redes basados en neuronas artificiales, el más clásico se denomina **Perceptrón multicapa**.



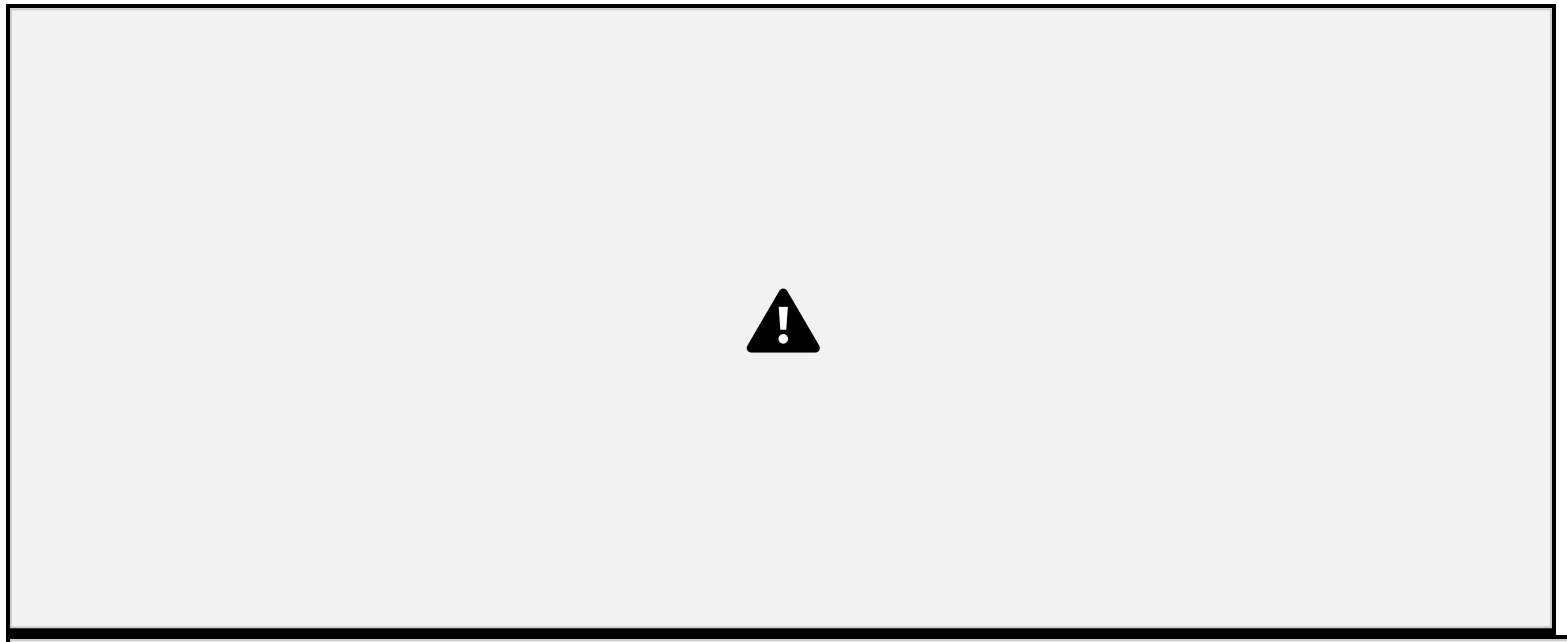
(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

4.Aplicaciones

Son la base de las **redes profundas**, actualmente muy utilizadas con mucho éxito.

Cada **capa extrae**

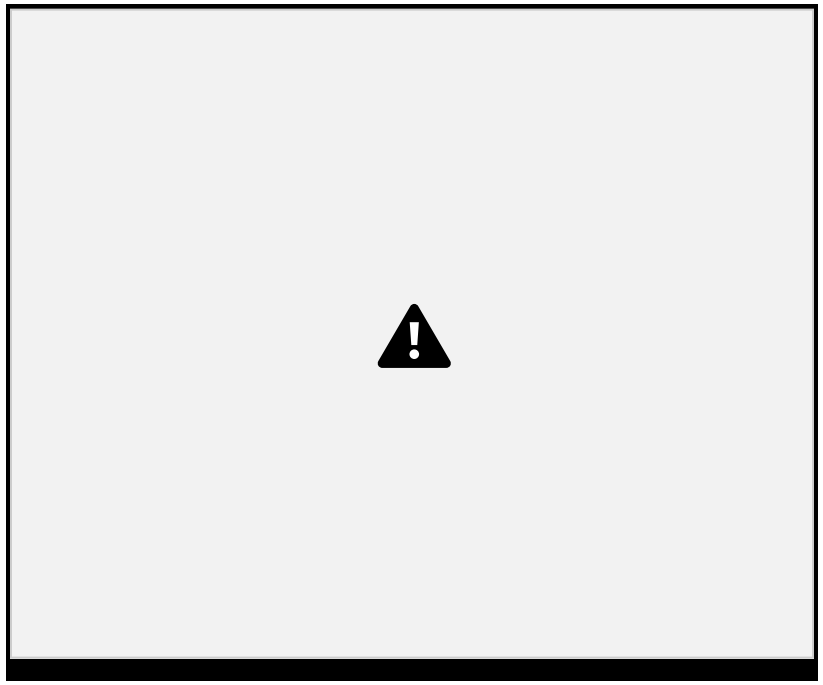
características cada vez más complejas

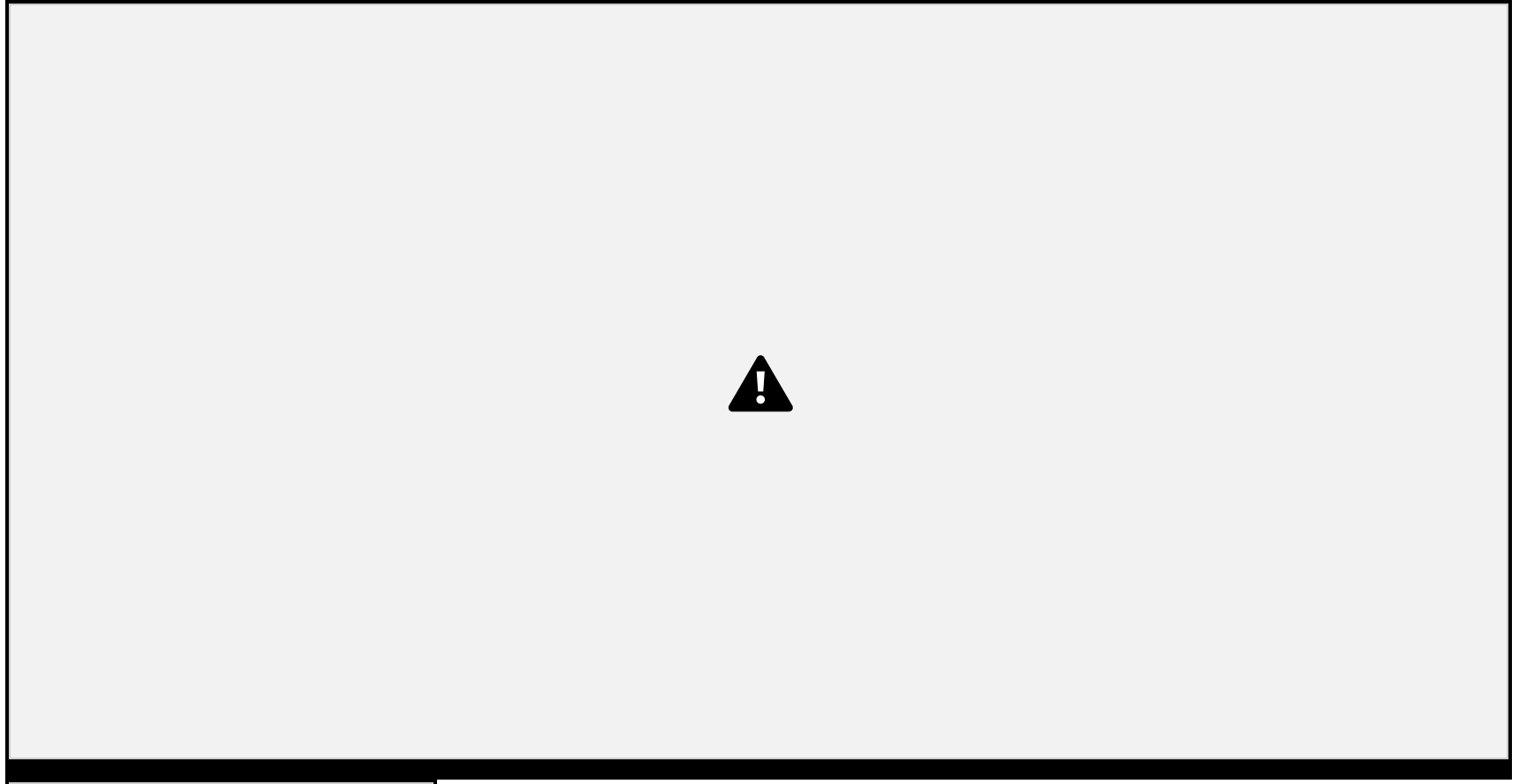


(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

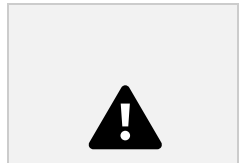
5.Propagación hacia delante

Forward

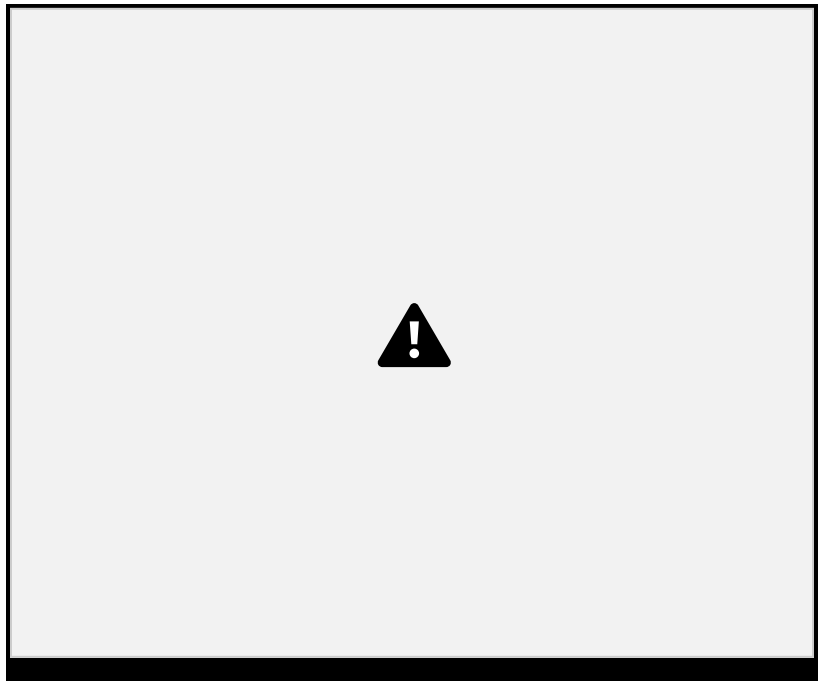


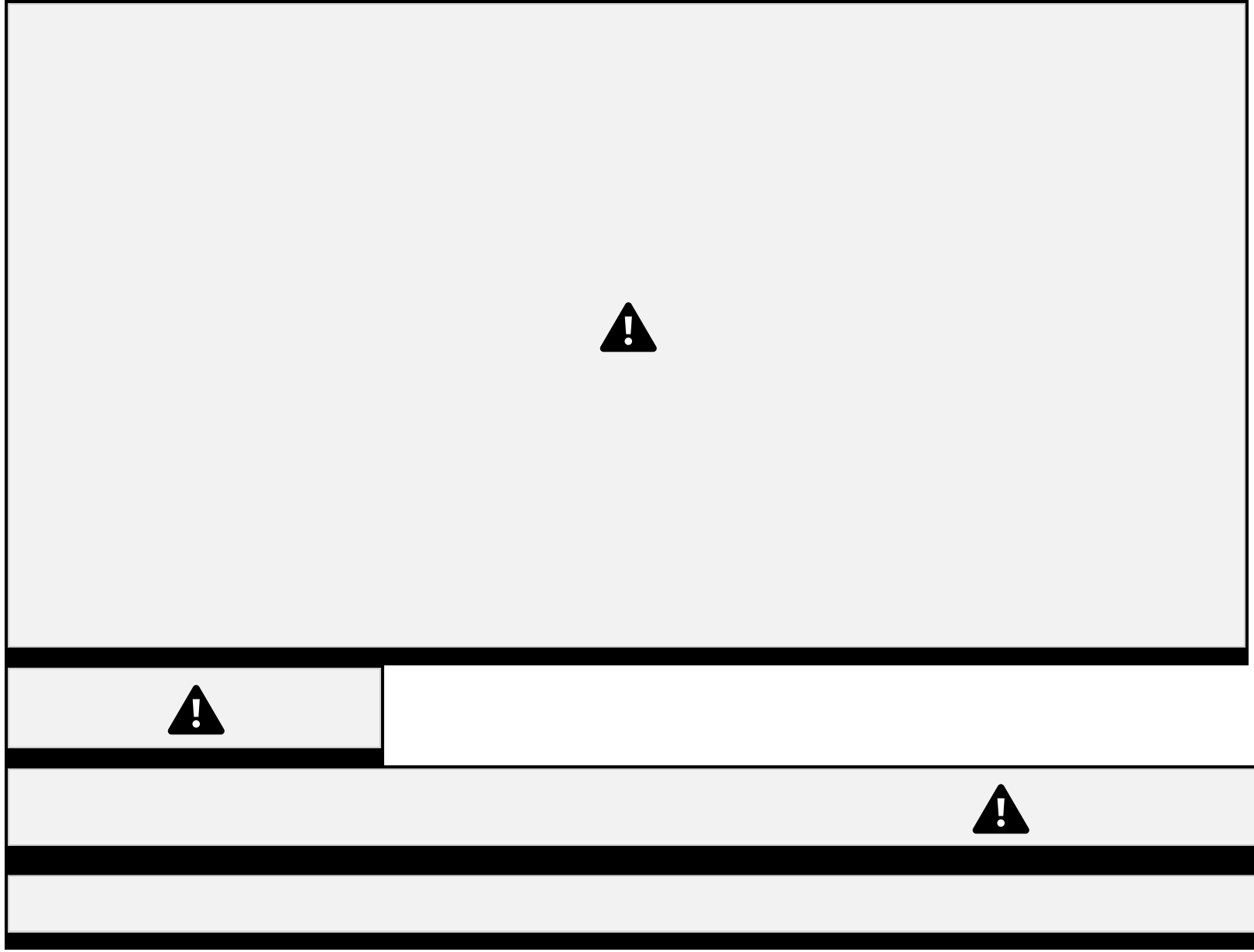


5.Propagación hacia delante



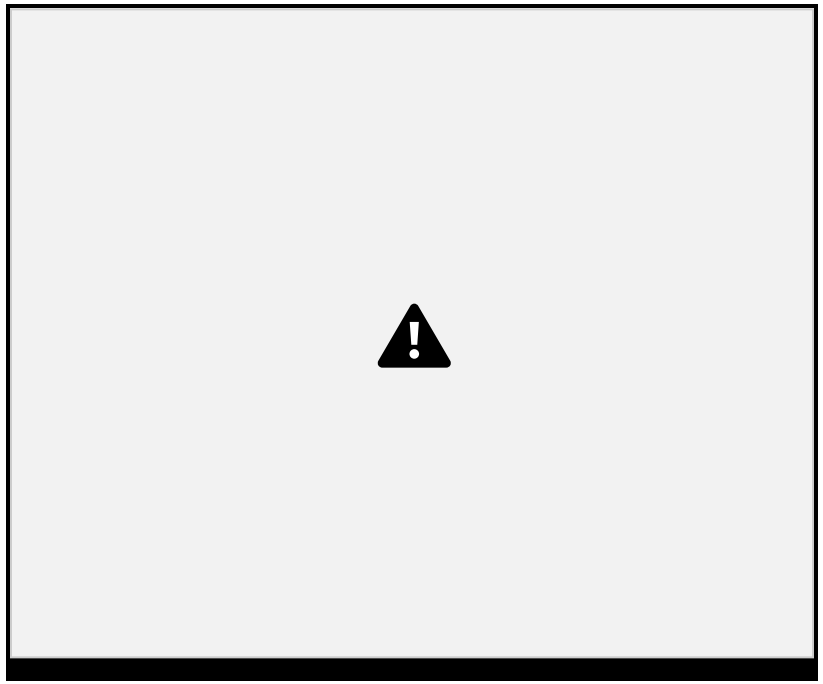
Forward

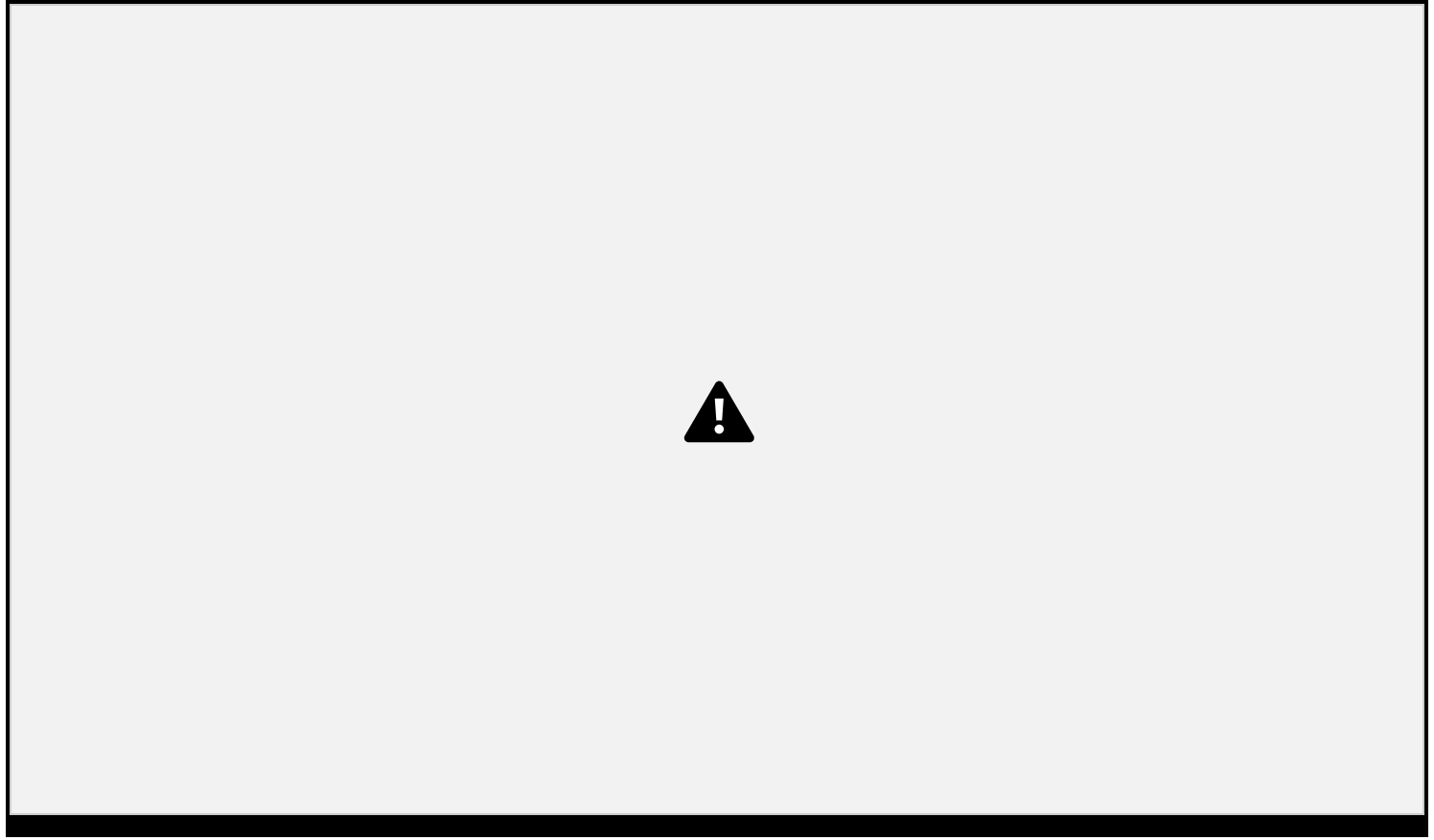


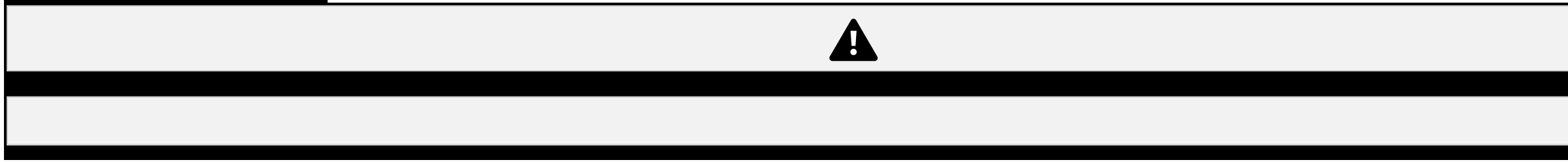


5.Propagación hacia delante

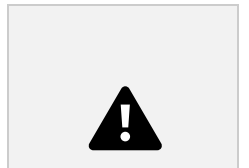
Forward



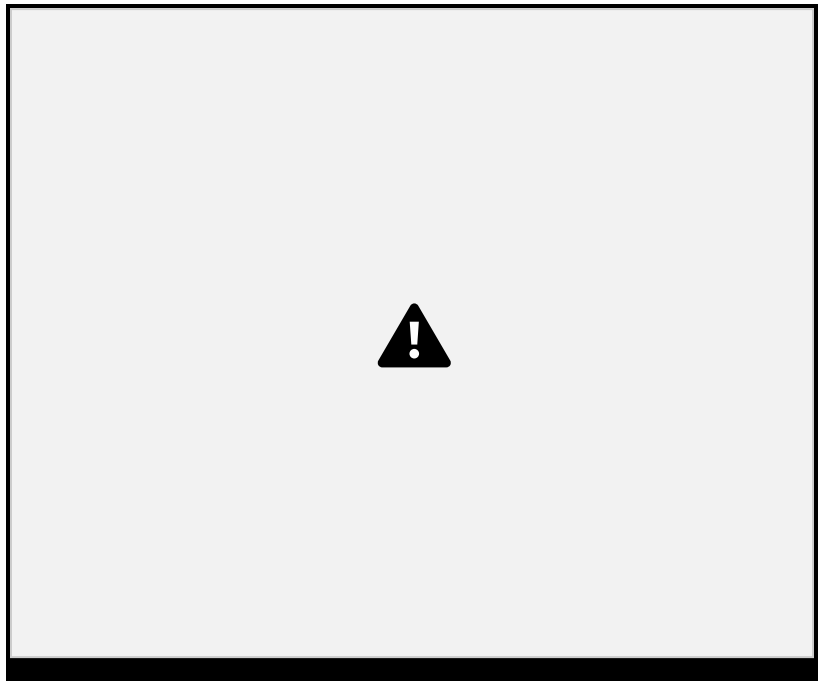


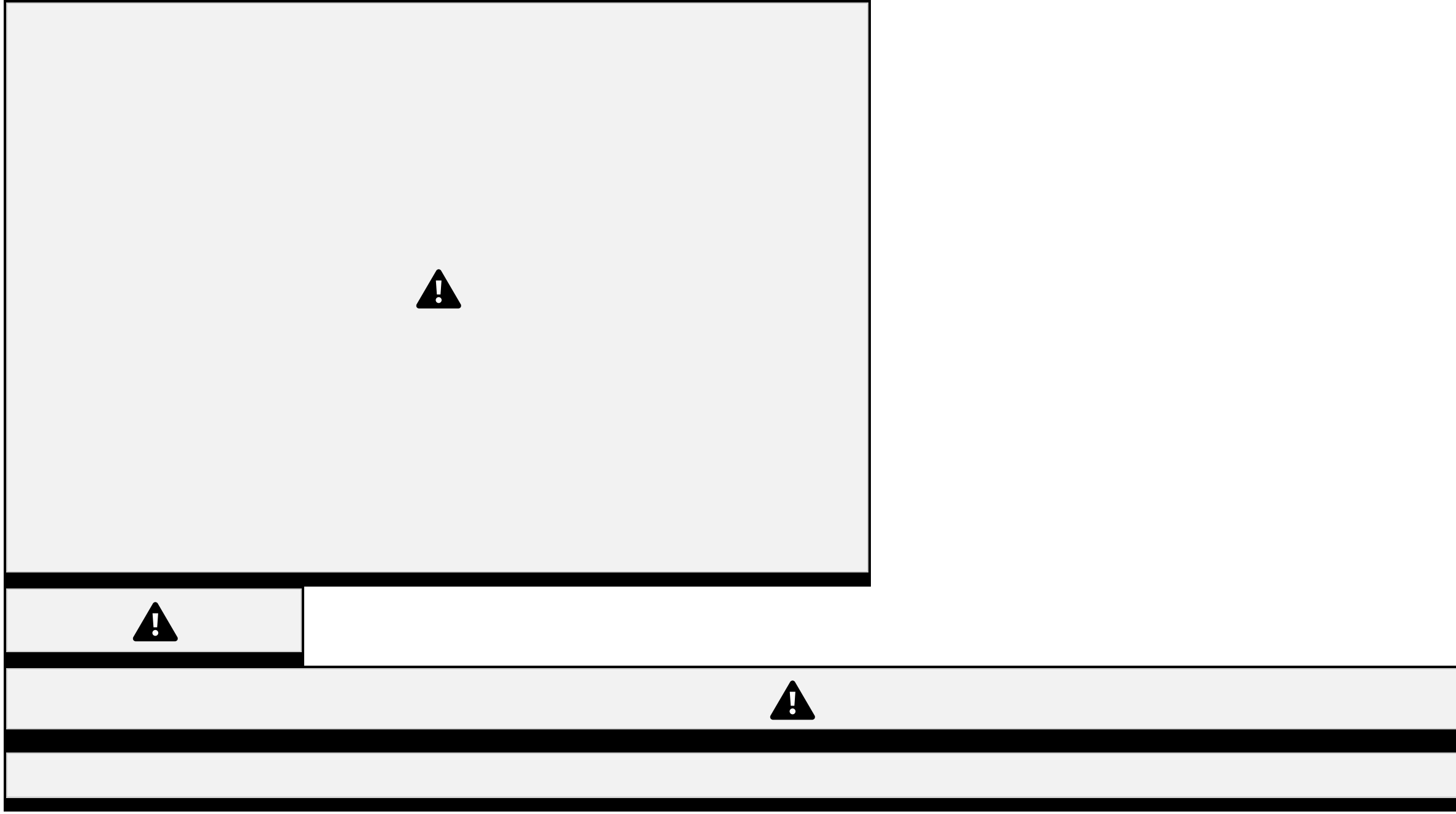


5. Propagación hacia delante



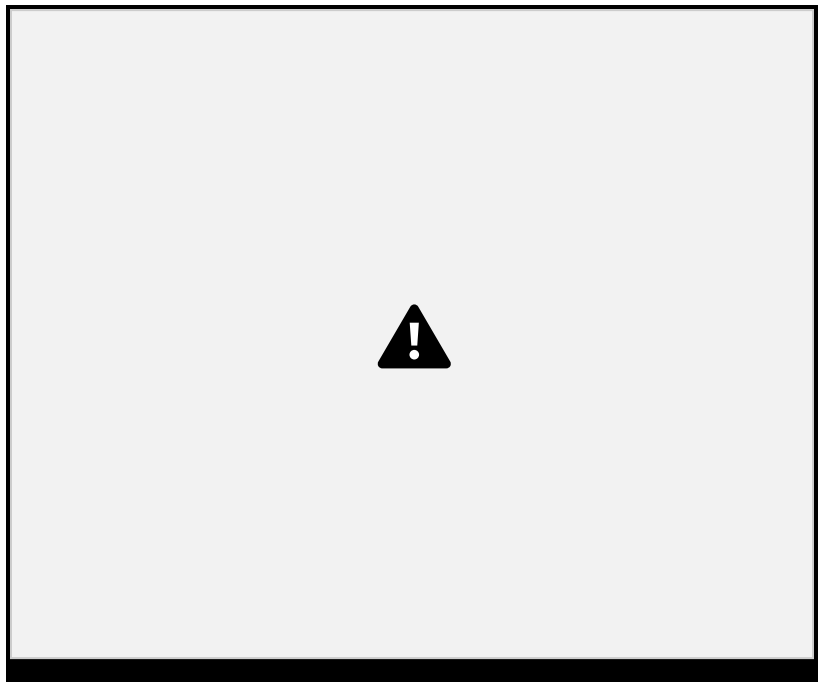
Forward

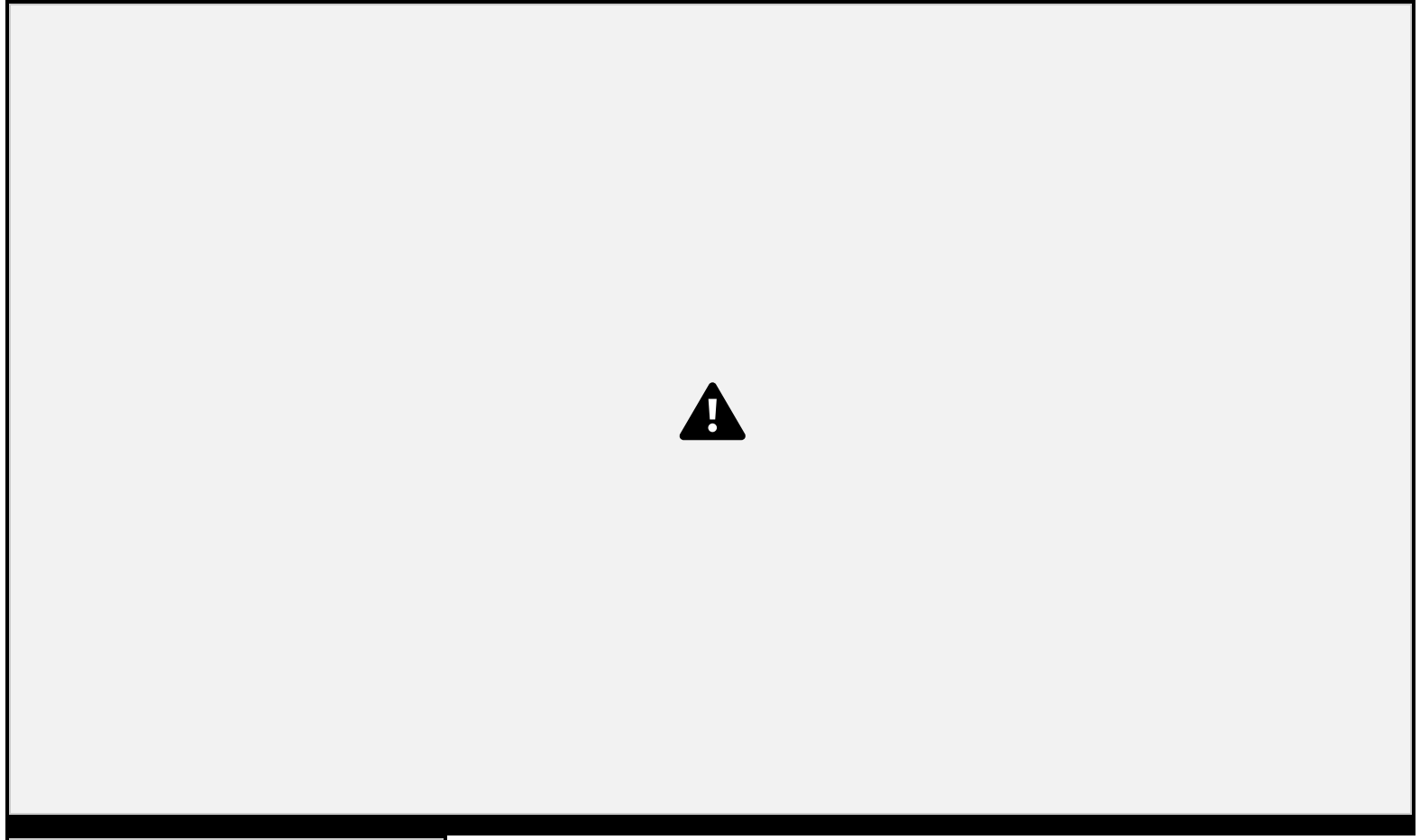




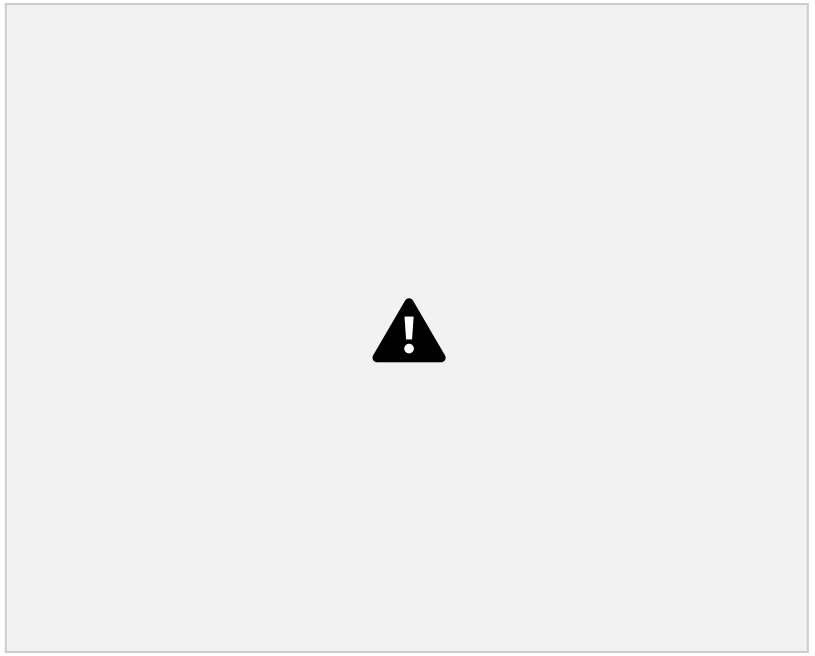
5. Propagación hacia delante

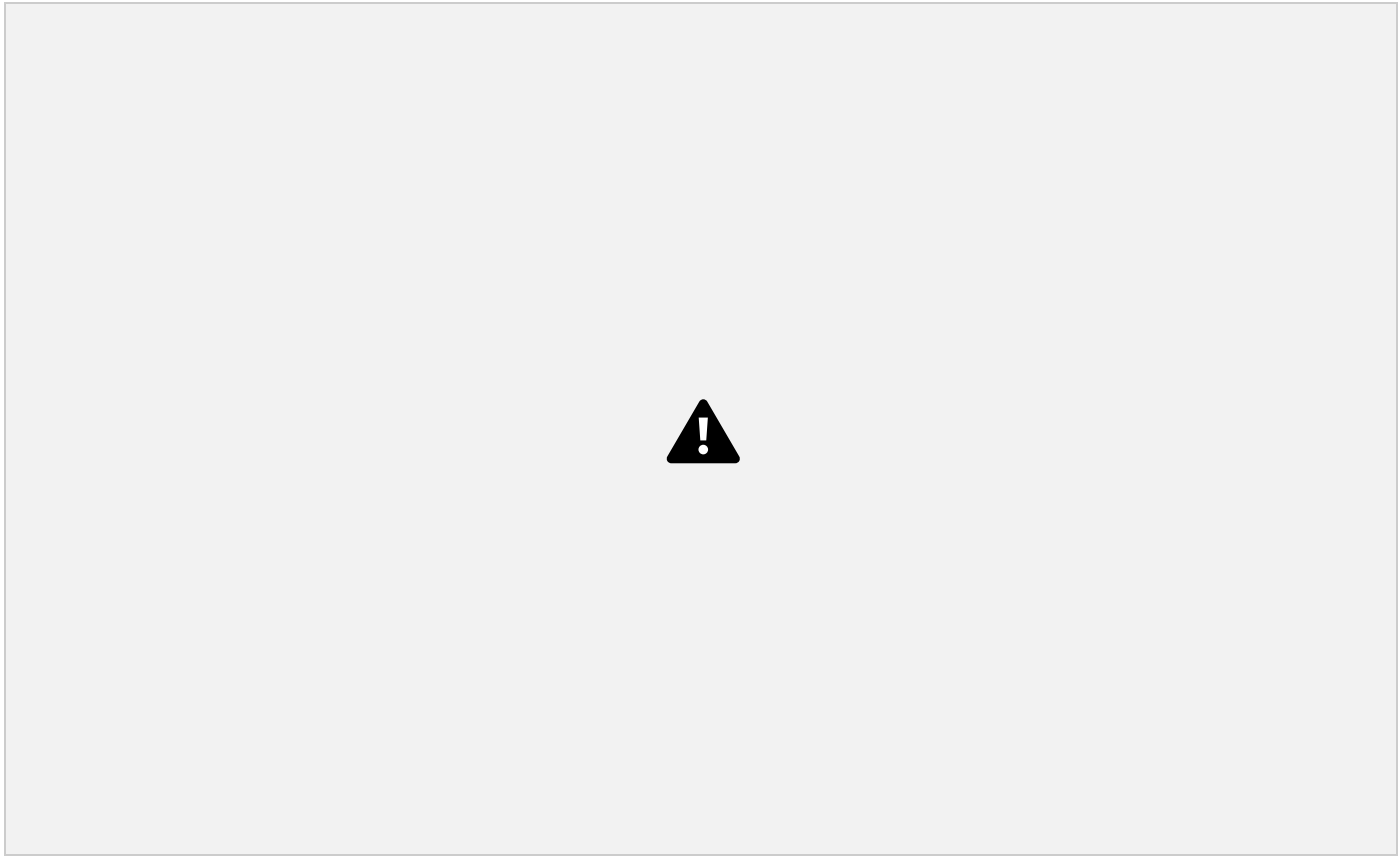
Forward





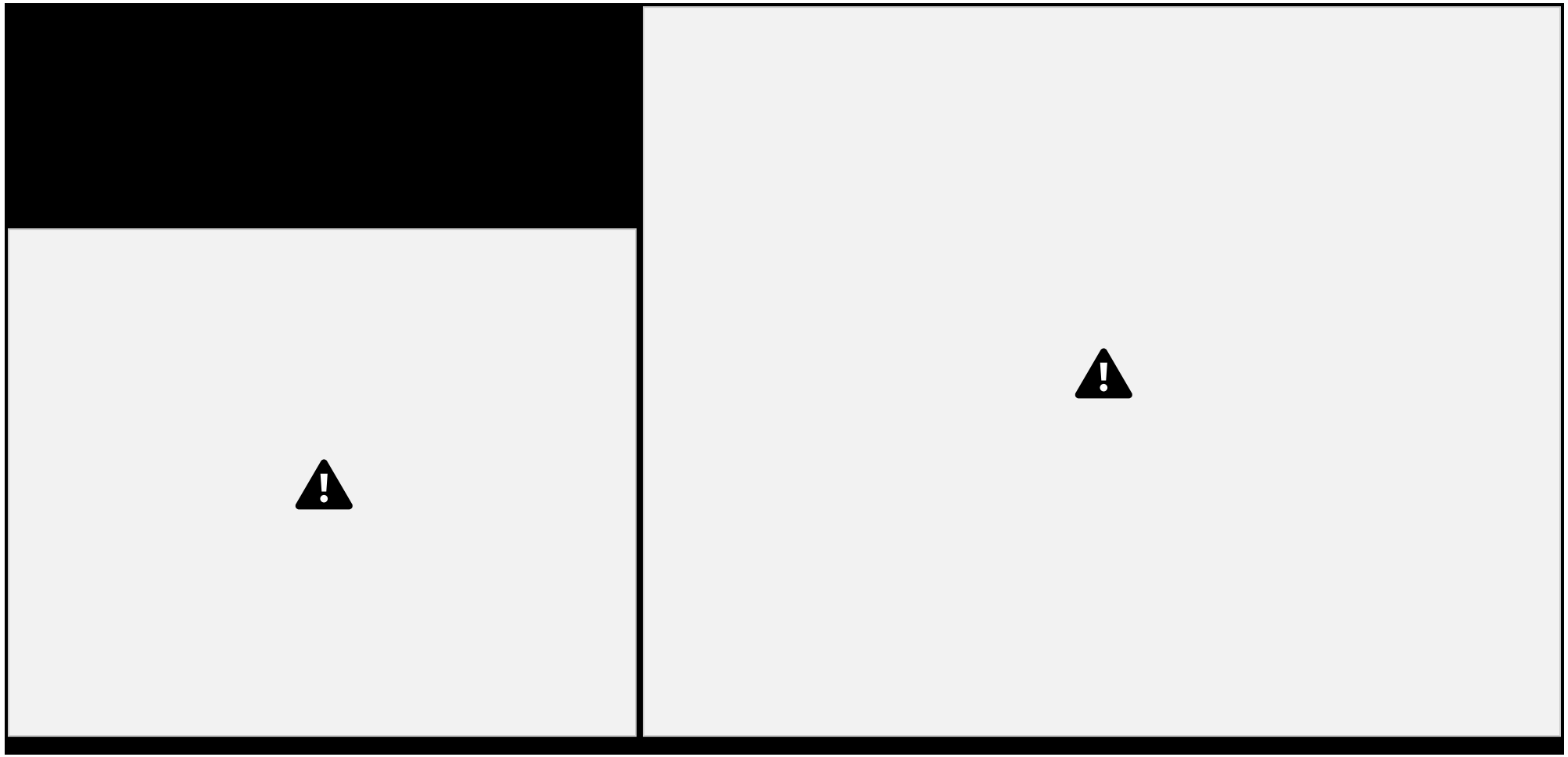
6.TensorFlow



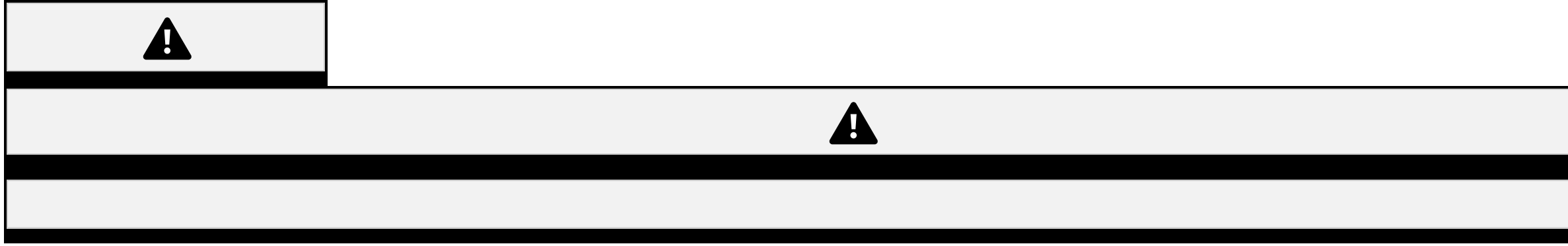


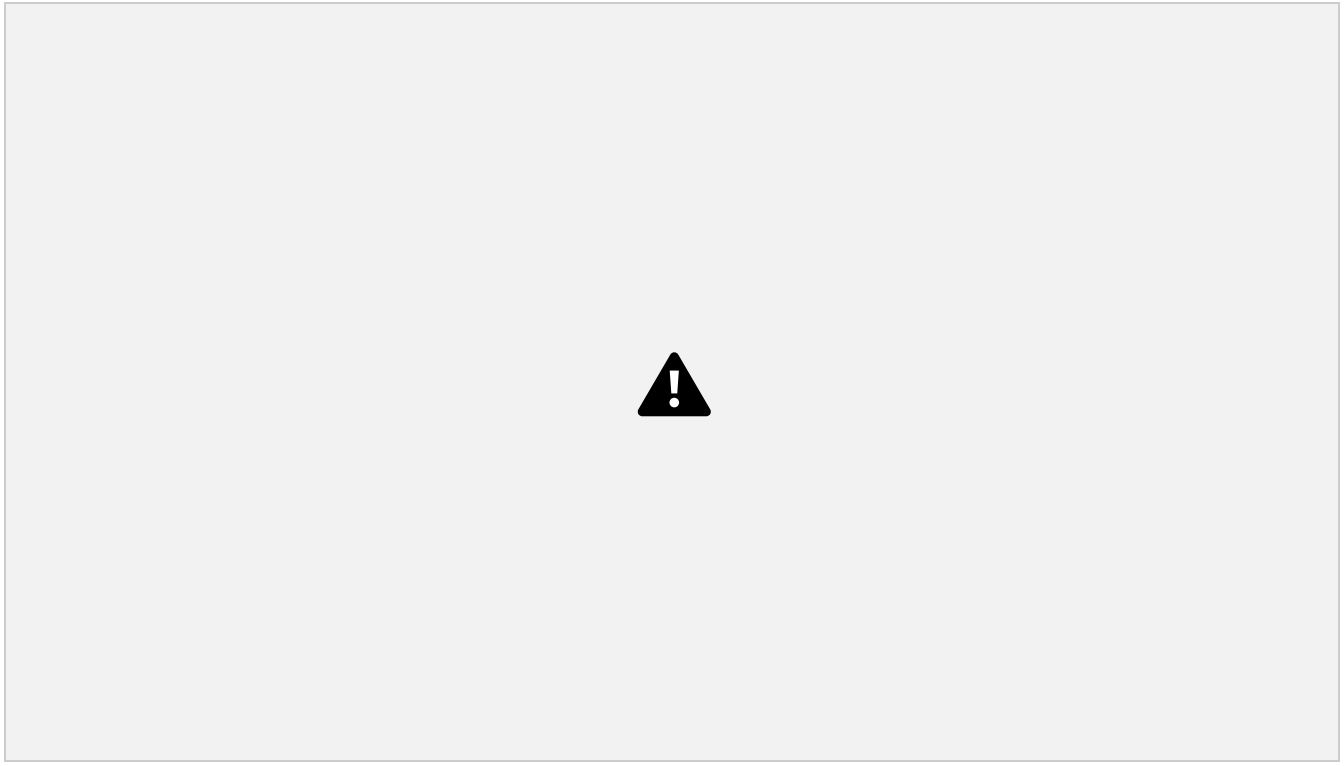
7.Clasificación multi etiqueta

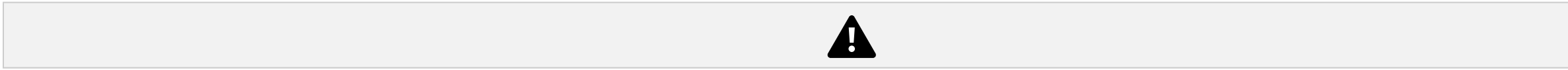
Uno vs Todos: Múltiples unidades de salida



8.Aprendizaje en RNs







9.scikit-learn

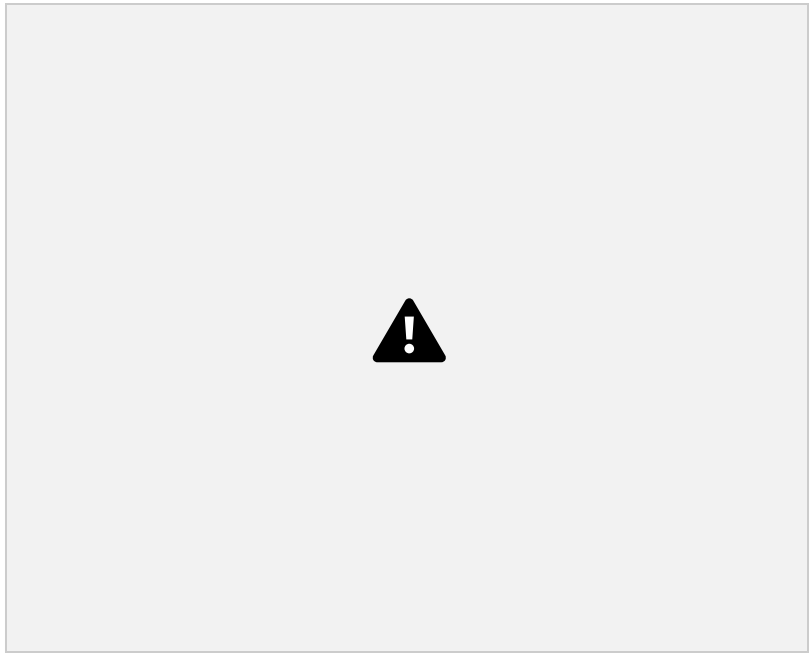
Contiene dos clases principales para RNAs

- *neuronal_network. MLPClassifier*. -
- *neuronal_network. M*

Ambas comparten varios parámetros:

- *Alpha*. Parámetro que rige la regularización.
- *Activation*. Función de activación:
  - *'identity'*(tenemos la *z=a*, no hay función de activación), *'logistic'*, *'tanh'* o *'relu'(es lineal)*
- *hidden_layer_sizes*. Tupla que contiene el número de neuronas en cada capa intermedia.

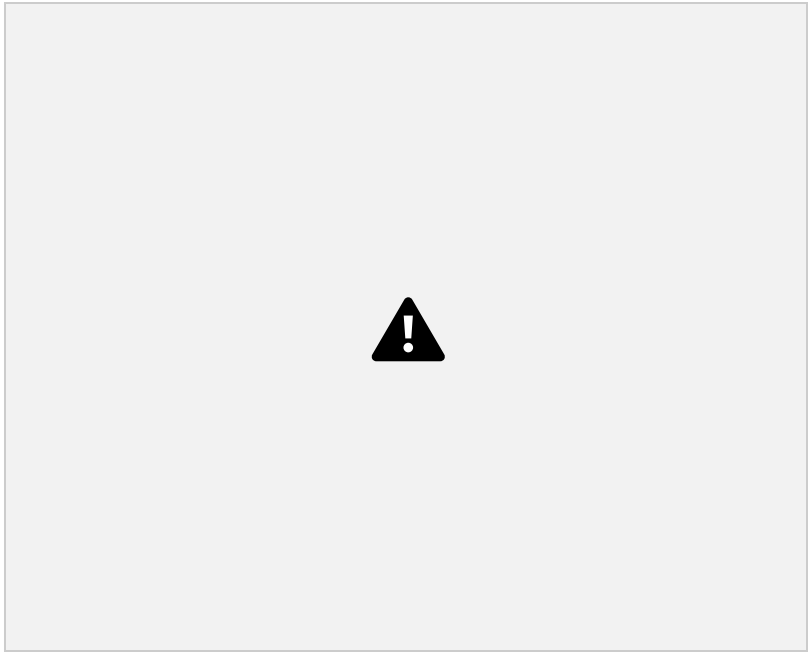
- *Solver*. Algoritmo de optimización:
  - *'lbfgs', 'sgd'* (gradiente descendiente estocástico) o *'adam'*



1.Contexto 

- Dado que el coste representa lo alejado que estamos del resultado deseado, queremos **Minimizar** dicho valor y, portanto, alcanzar su valor mínimo (idealmente 0): - Usamos Gradiente Descendiente
  - **Hipótesis**: el valor a reducir decrece rápidamente si en cada parámetro de la Fórmula, aplicamos un corrector negativo.

    → Este corrector o gradiente se obtiene a partir de la derivada parcial en el peso a establecer computado.

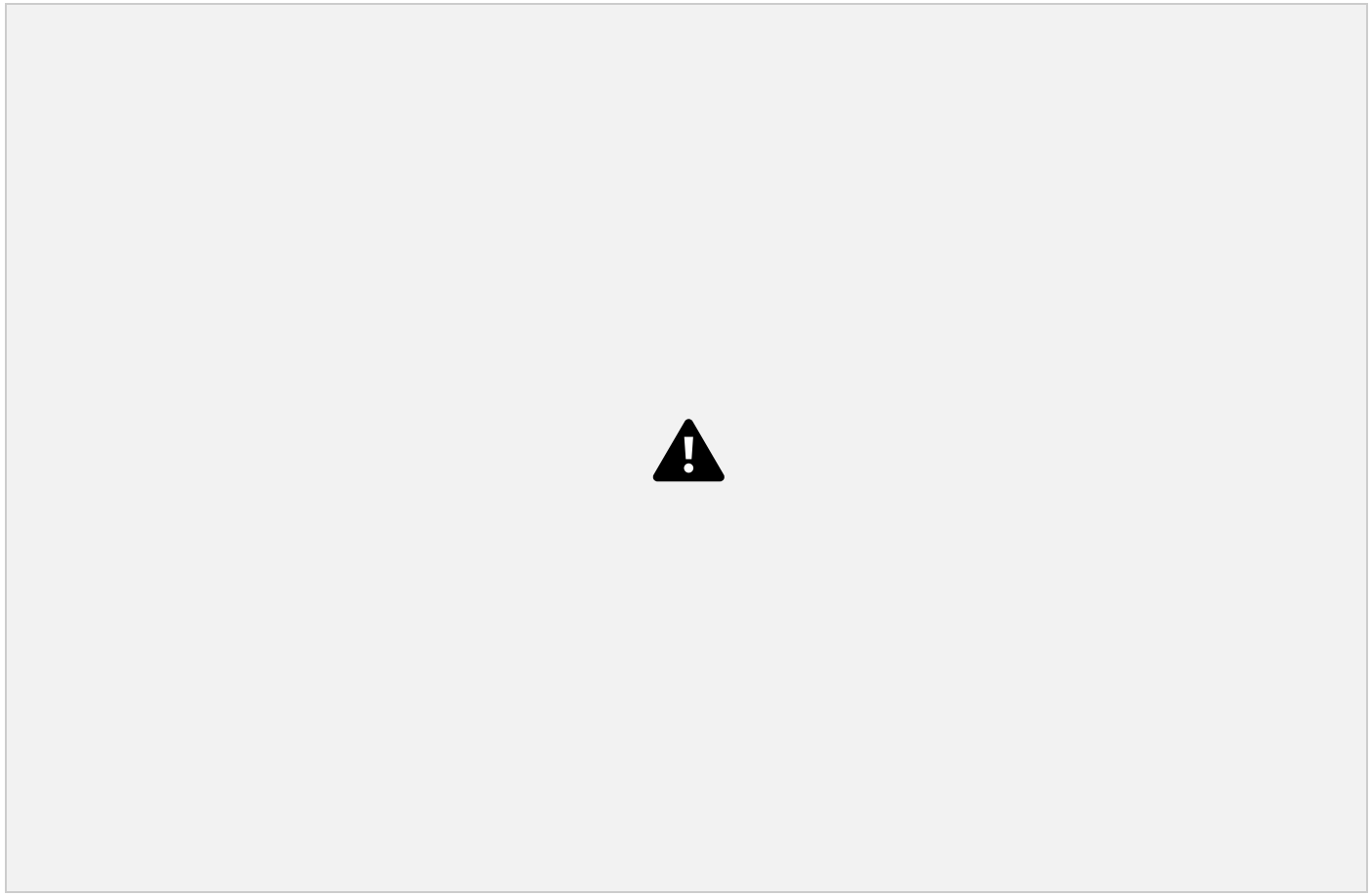


- La aplicación de cada gradiente (al completo) sobre cada peso no es una buena idea, ya que aumenta la posibilidad de caer en **óptimos locales**.
  - En su lugar, se aplica un **ratio de aprendizaje** de forma que en cada iteración, el valor de un peso se actualiza:

$$wi_{t+1} = wi_t * (1 - ratio) - ratio * gradient$$

(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

2.Ejemplo



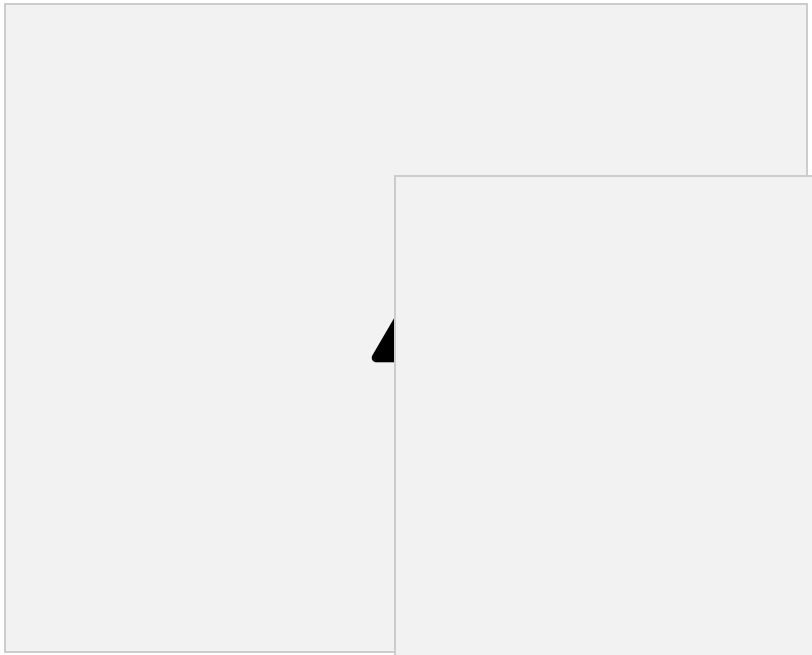
https://medium.com/@pytholabs/multivariate-linear-regresion-from-scratch-in-python-5c4f219be6a

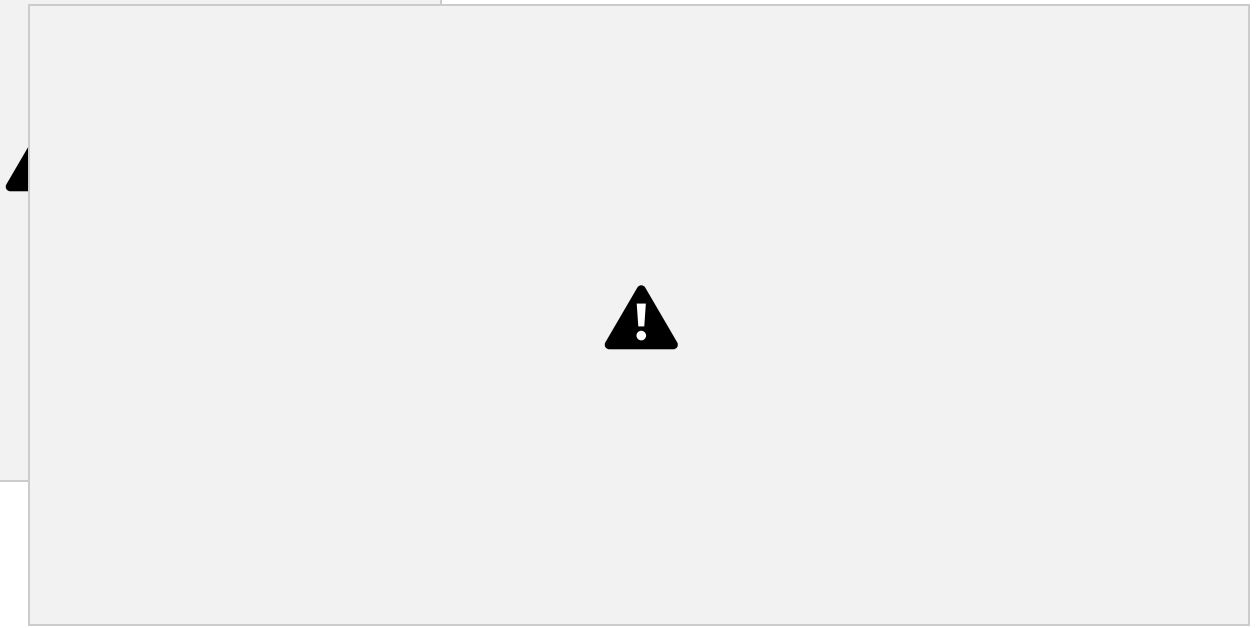
3.El gradiente descendiente (I)

El método gradiente descendiente permite minimizar una función. Para ello, básicamente parte de unos valores para los parámetros elegidos aleatoriamente, y avanza en pasos sucesivos hacia un mínimo.

Utilizando como ejemplo una función cualquiera: $F(\theta_0,\theta_1)$.

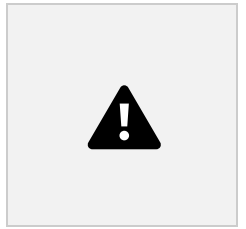
- Se pretende minimizar $F(\theta_0,\theta_1)$
- Se parte de unos valores θ_0 y θ_1 elegidos aleatoriamente.
- Se actualizan θ_0 y θ_1 mientras se reduzca $F(\theta_0,\theta_1)$

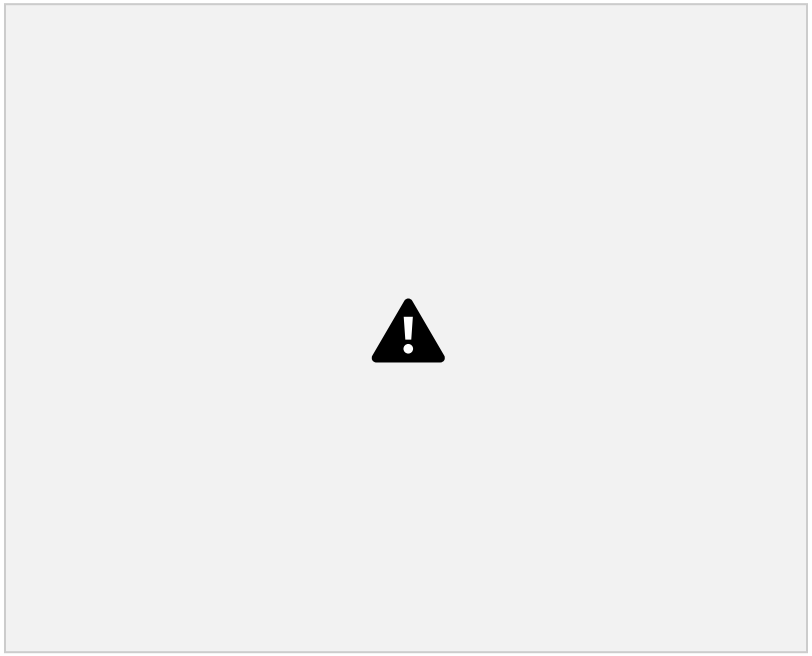


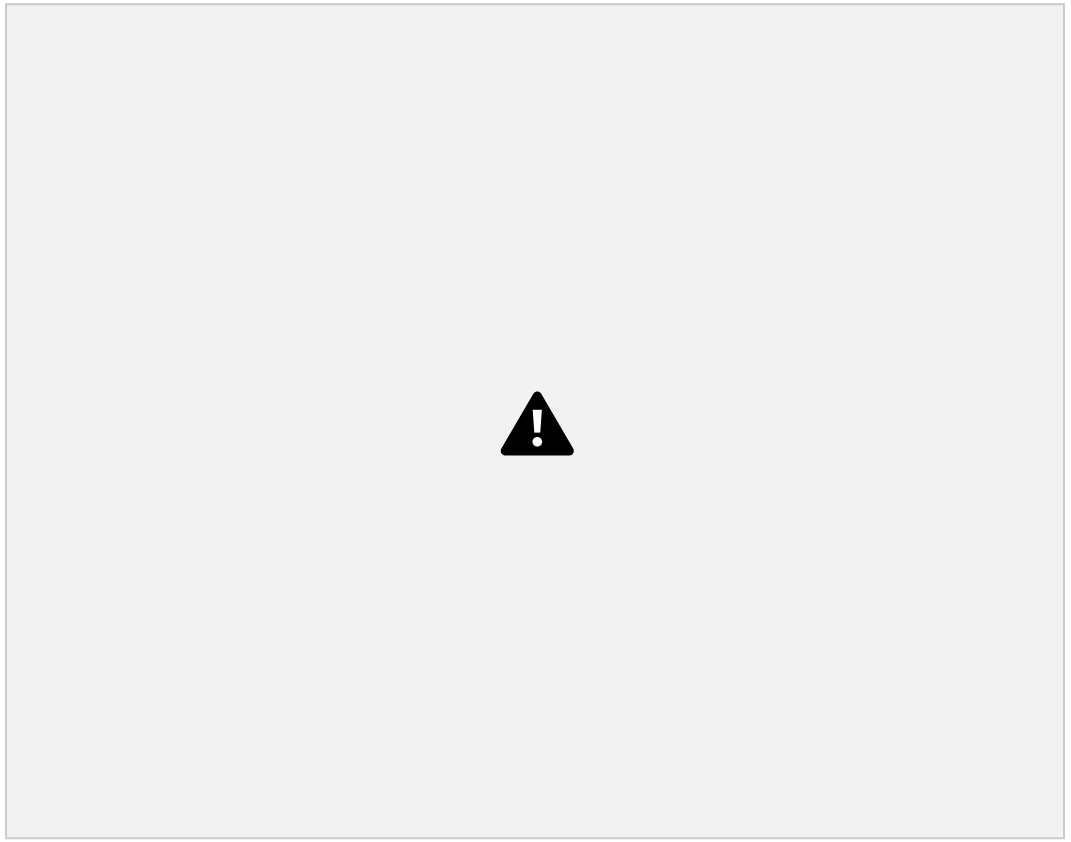


(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

3.El gradiente descendiente (I)

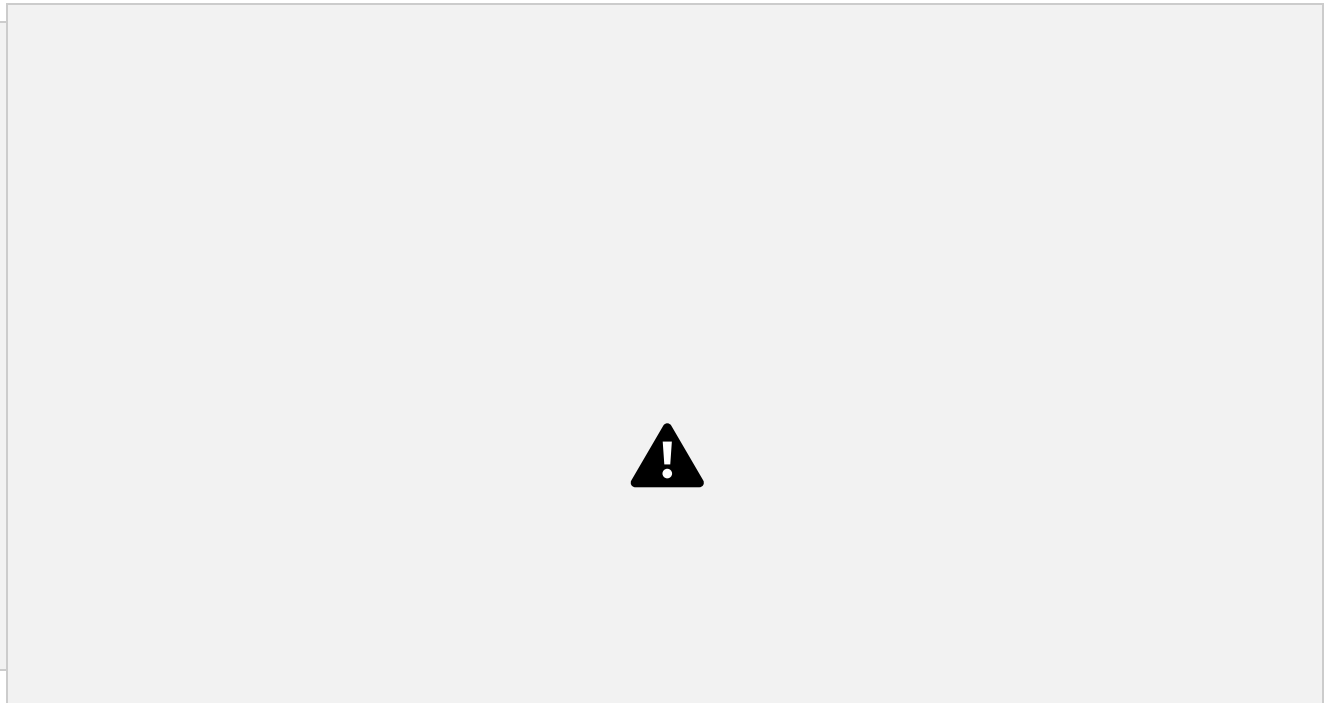






3.El gradiente descendiente (III)

- Suponiendo una funci
  cualquiera de **una** var
  $F(\theta_1)$
  El signo de la
  derivada nos
- dirá si $\theta_1$crece o



decrece. Cuando la derivada de la

tangente es 0 se estanca en un valor.

(+51) 976 760 803 w[redacted]ytics.com info@datayanalytics.com

3.El

gradiente descendente

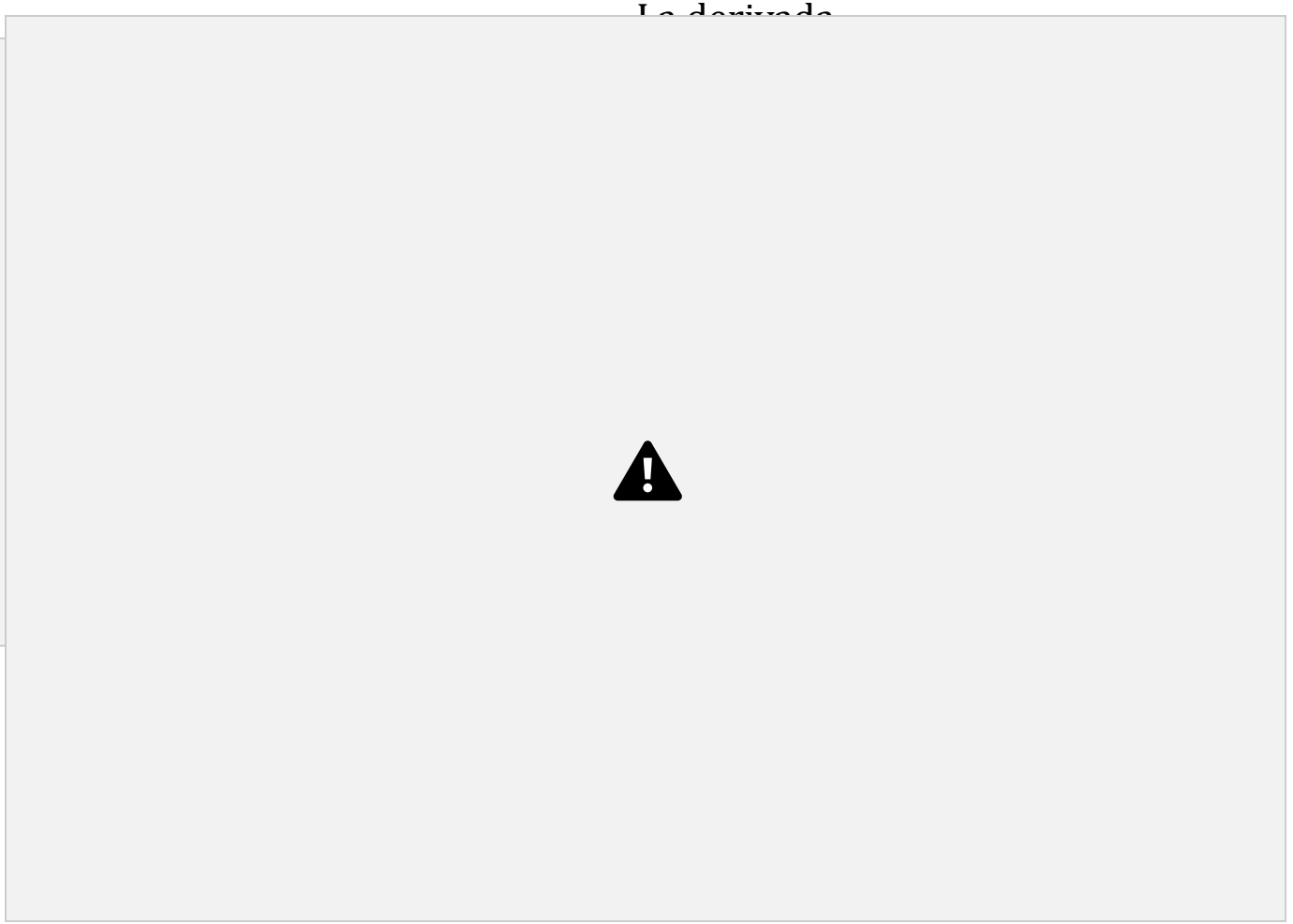
endi ente (III)

La derivada

Suponiendo una Func
cualquiera de **Una** var

El signo de la derivada

nos dirá si θ_1 crece o decrece.



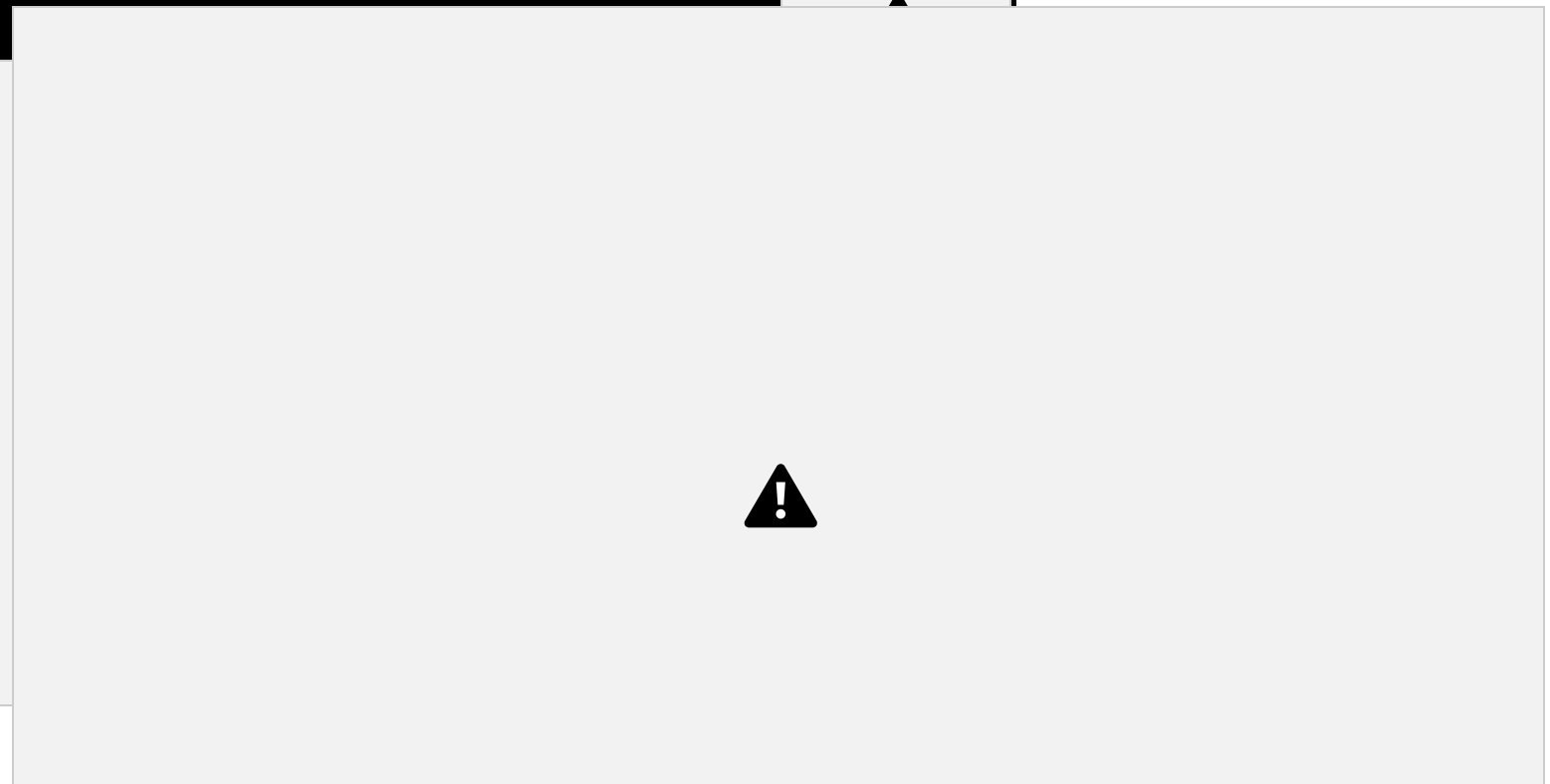
Cuando la derivada de la

tangente es 0 se estanca
en un valor.

(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

3.El gradiente

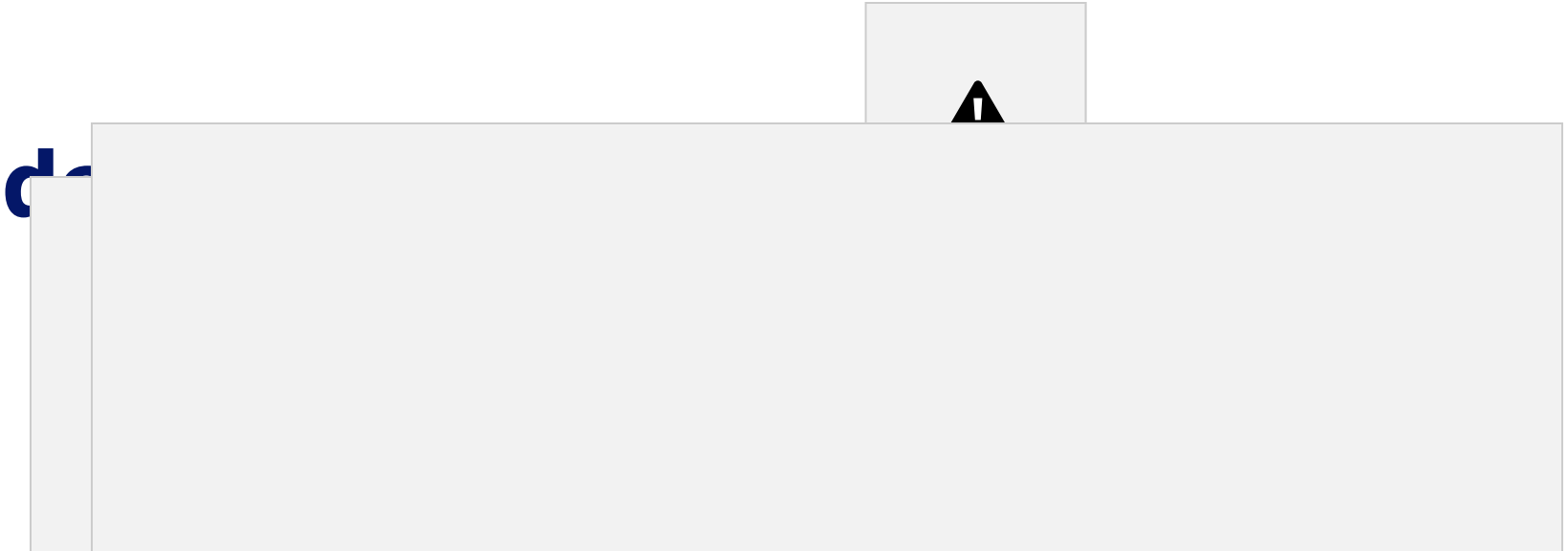
Como la función crece la
derivada es positiva



(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

3.El gradiente d

Como la función crece la derivada es positiva



Por tanto, decrece y se acerca
al punto de convergencia

(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

3.El gradiente descendiente (III)

Como la función crece la derivada es
positiva

Por tanto, decrece y se acerca al punto de convergencia

Ahora la pendiente es negativa, porque la función decrece.
Ahora la derivada es negativa y al restar un número negativo con **-α** entonces θ_1 crecerá y será positivo (se desplaza a la derecha)

(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

3.El gradiente descendiente(IV)

Cuando el valor de los

parámetros se

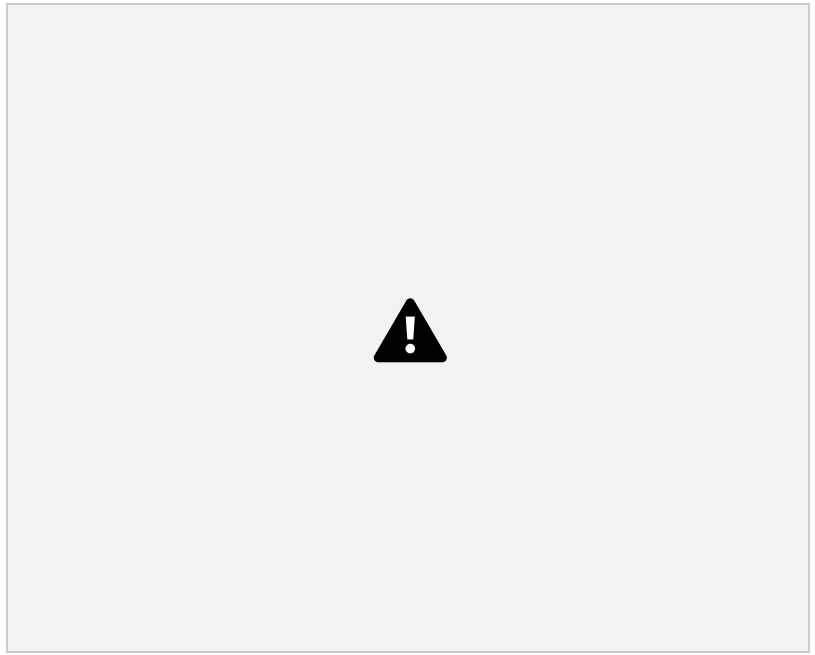
aproxima a un mínimo, el algoritmo da pasos más pequeños de manera automática, ya que $_{d\theta 1}{}^{d}F(\theta_1)$ es menor.

Portanto, no es necesario decrementar α a lo largo de las

iteraciones. Debemos fijar un valor aceptable.

3.El gradiente descendiente (V)

El valor α se denomina razón de aprendizaje.



Siαesdemasiadopequeña,la convergenciapuedeserdemasiadolenta.

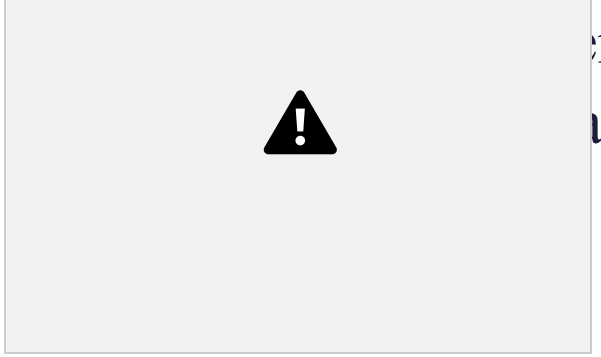
Si α esdemasiadogrande,elalgoritmo puedenoconverger,einclusodivergir.

(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

3.El gradiente descendiente (V)

El valor α se denomina razón de aprendizaje.

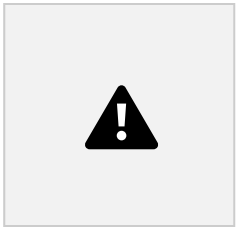
Si α es demasiado



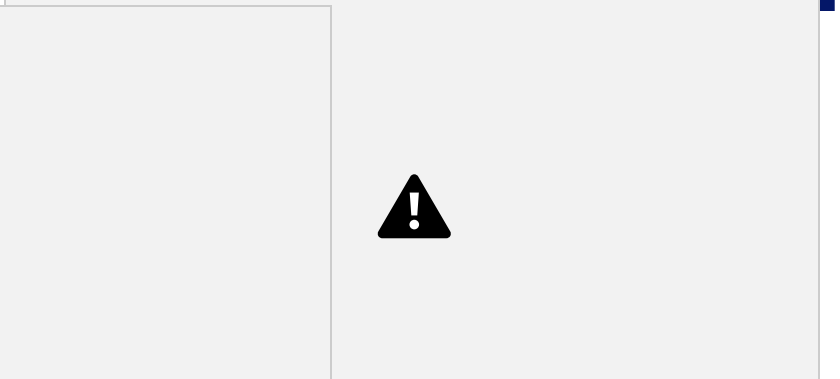
ergencia
ado lenta.

Si α es demasiado grande, el algoritmo puede no converger, e incluso divergir.

(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com



3 El gradiente descendiente (VI)



El método gradiente descendiente siempre converge a un óptimo local. En

algunos casos, este óptimo coincide con el global o no.

El punto al que converge el algoritmo depende de los valores iniciales de los parámetros.

descendiente (VI)

El método gradiente descendiente siempre converge a un óptimo local. En algunos casos,

este óptimo coincide con el global o no.

Fijarse que hay otro óptimo mínimo.

El punto al que converge el algoritmo depende de los valores iniciales de los parámetros.

(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

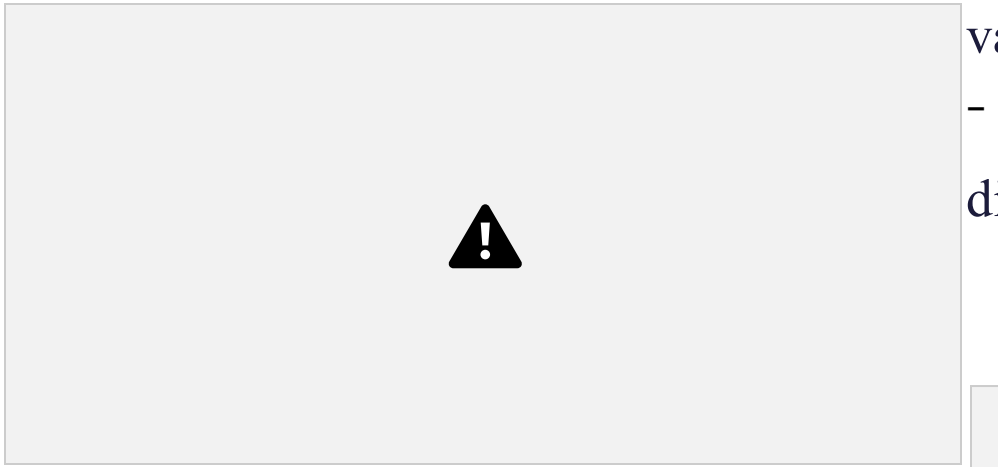
3 El gradiente descendiente (VI)

El método gradiente descendiente siempre converge a un óptimo local. En algunos casos,

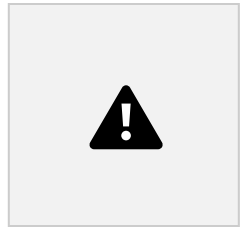
este óptimo coincide con el global.

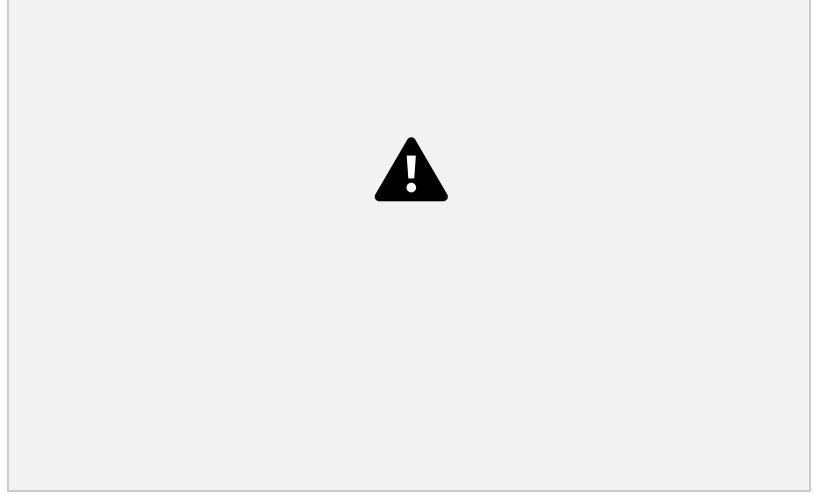
Fijarse que hay otro óptimo mínimo.

El punto al que converge el algoritmo depende de los valores iniciales de los parámetros.



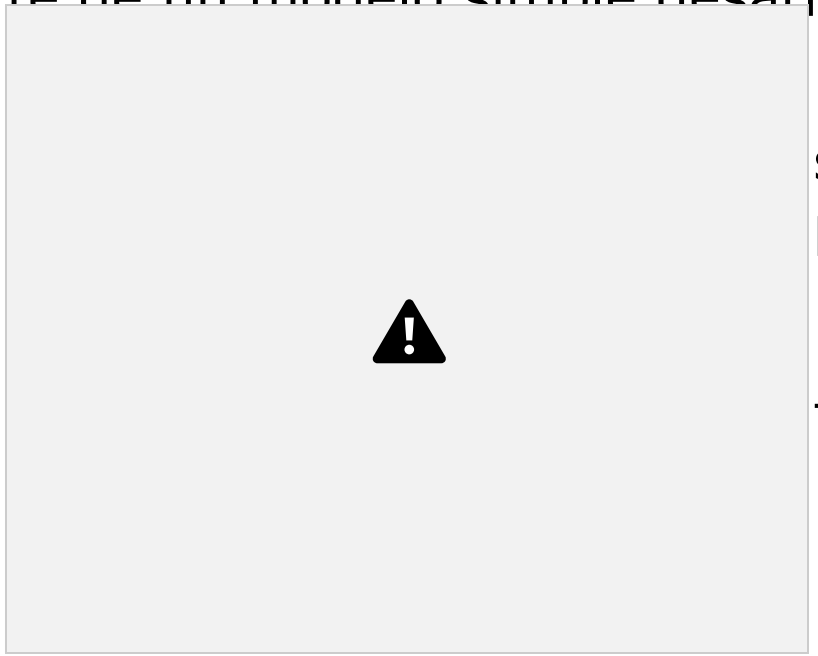
- Aquí tenemos dos puntos de partida que son diferentes en el mínimo de convergencia.





1.Perceptron multicapa

Perceptrón multicapa parte de un modelo simple desarrollado en la década de los 50: **Perceptrón**



Este modelo toma una serie de N **entradas** $(x_1,...,x_n)$ y genera una salida a partir de una función que utiliza las entradas, una serie de **pesos** $(w_1,...,w_n)$, y un factor adicional: **sesgo**(b)

(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

2.Conceptos clave

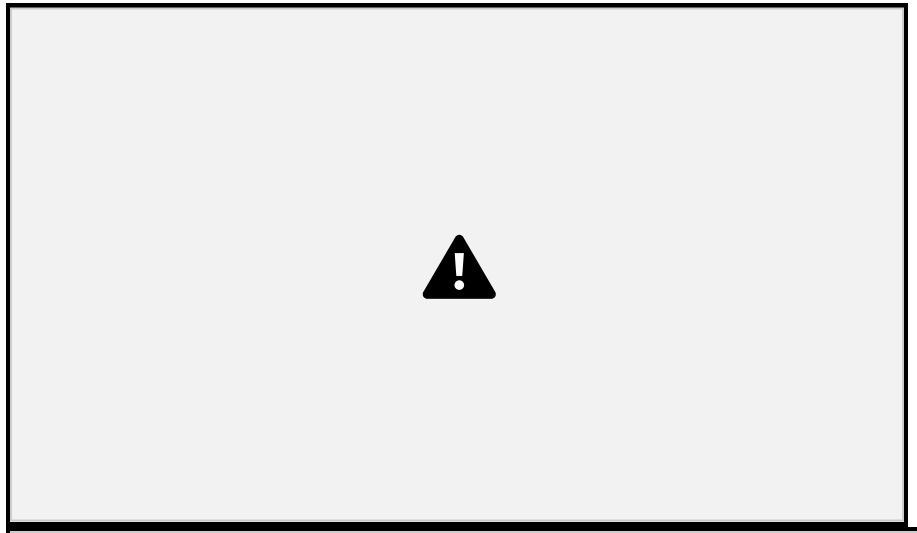
Cómo se combinan las entradas→ pesos*(w)*

Cómo se determina la salida→ función de activación

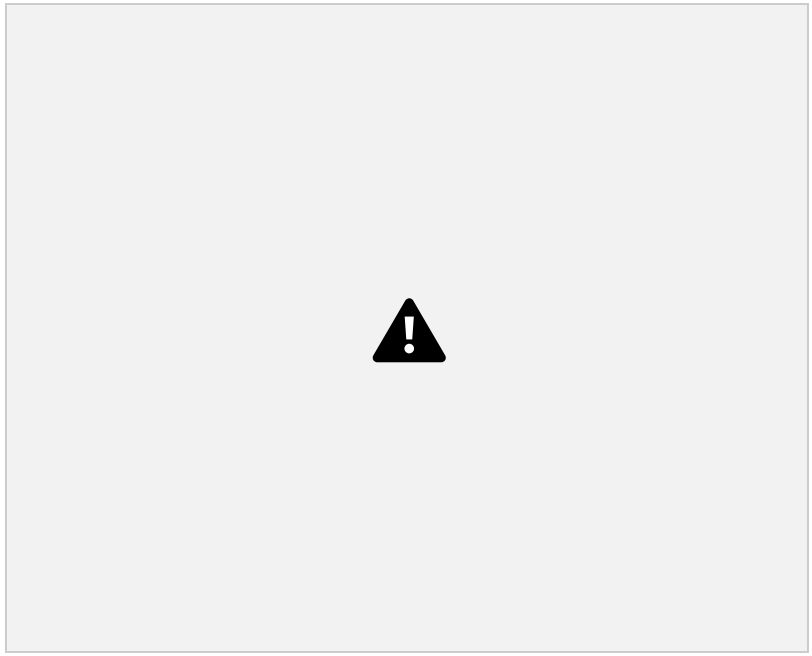
- La función de activación suele establecerse como:

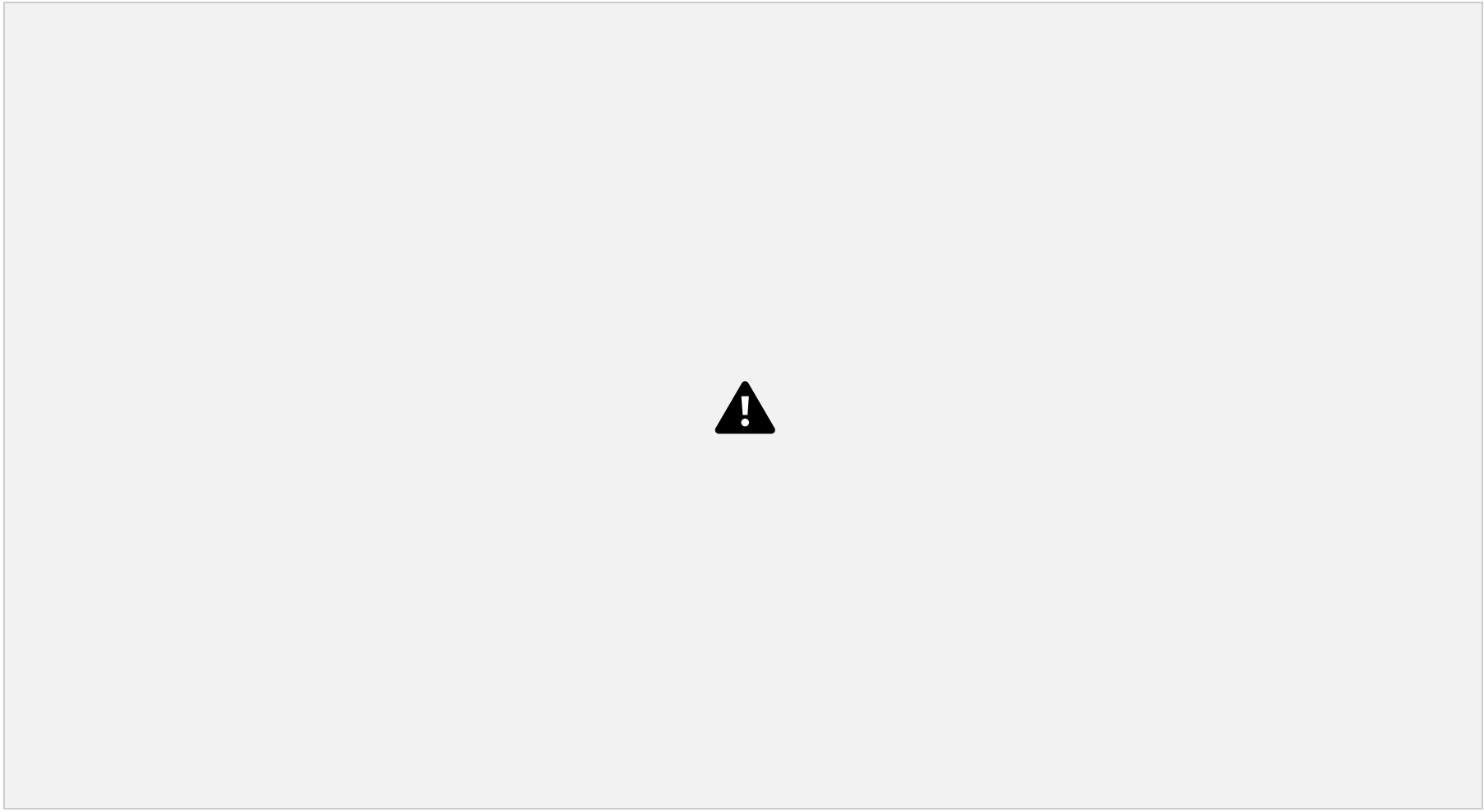
Cómo se establecen estos parámetros→ aprendizaje

Qué otros parámetros intervienen→ sesgo



3.Arquitectura



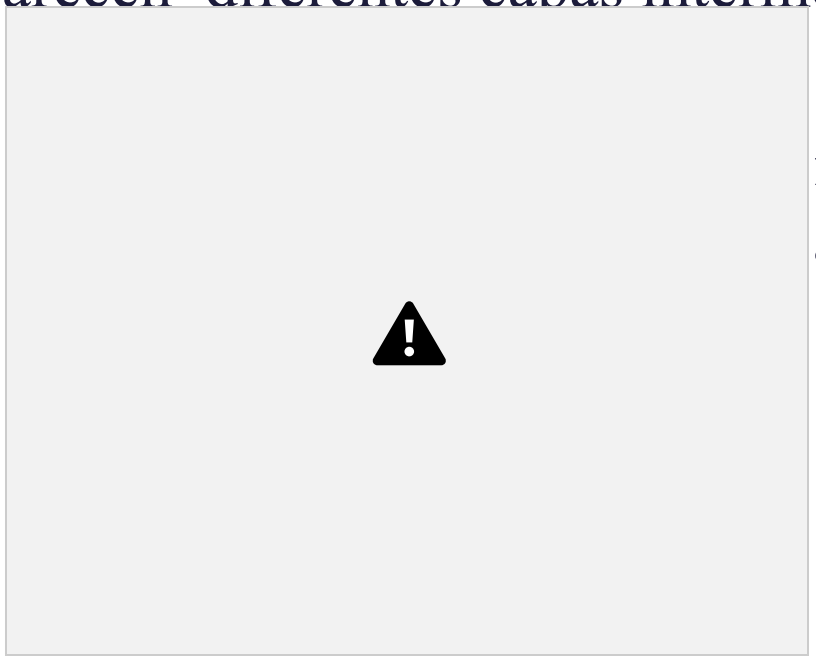


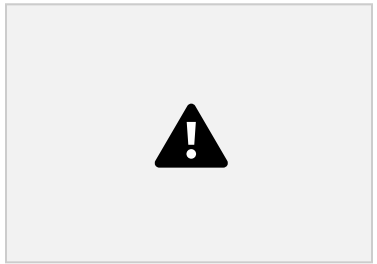
4.Capas

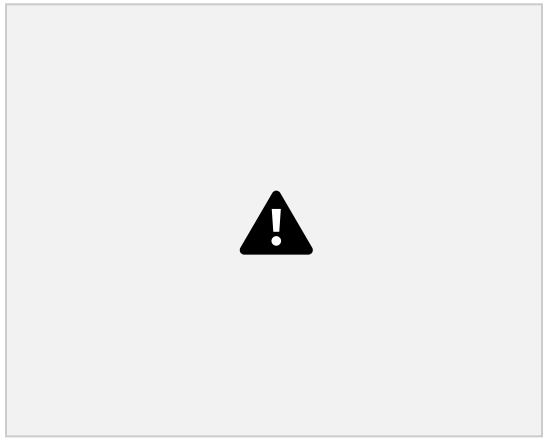
El MLP se entiende como la evolución del Perceptrón Simple donde aparecen diferentes capas intermedias (llamadas **ocultas**) entre la entrada y salida.

En estas capas intermedias podemos reemplazar la función anterior por una

sigmoidal*







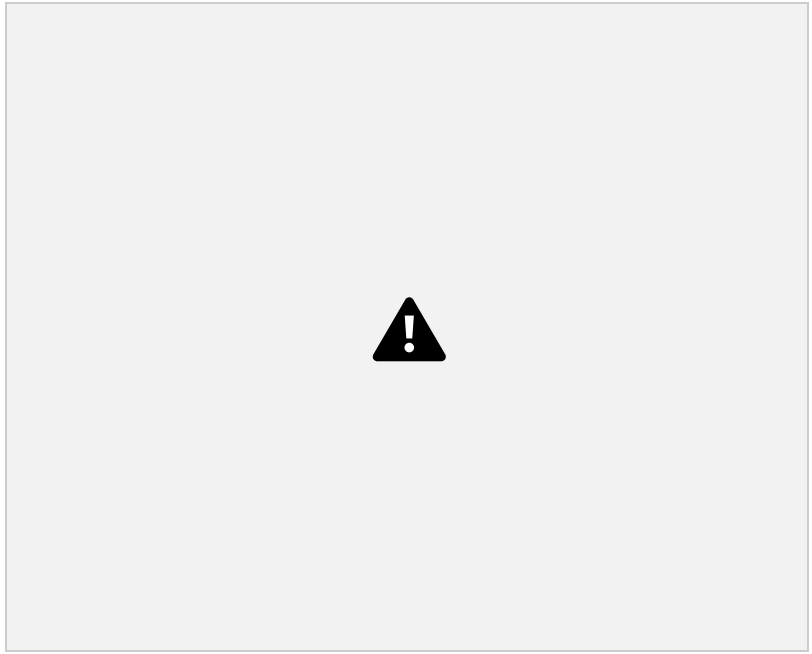
*Algunos autores limitan el uso del término MLP al uso de funciones básicas y al introducir sigmoidales pasan al término de Redes Neuronales.

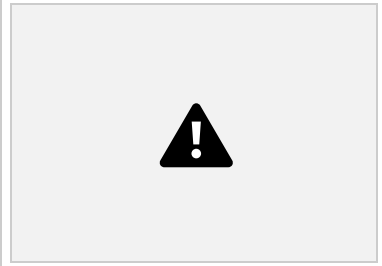
(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

5.Función sigmoidal

El uso de esta función introduce ciertos cambios:

- La generación de **salidas** que ya no son **binarias** sino en el Rango [0,1].
- La capacidad de resolución de **problemas no lineales.**





(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

6.Características

El número de capas ocultas aumer ed

→ **Afecta a** , pero sobre todo **aumenta** la zaje.

El proceso de

- aprendizaje puede verse como la búsqueda de
- reducir un valor llamado **Coste.**
- Este valor Coste representa lo lejos que está nuestra red de
- producir un **resultado perfecto** sobre un conjunto de entrenamiento.

(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

7.Aprendizaje

El **aprendizaje** es clave→ computación del **coste**: clave

En lugar de

computar el error absoluto, se buscan **funciones de coste** que varíen

ligeramente ante pequeños cambios en los parámetros (pesos o sesgo).

En este proceso se hace uso de dos técnicas:

- **Back propagation**
- **Gradiente Descendente**

El aprendizaje es un proceso iterativo donde se introduce un parámetro:

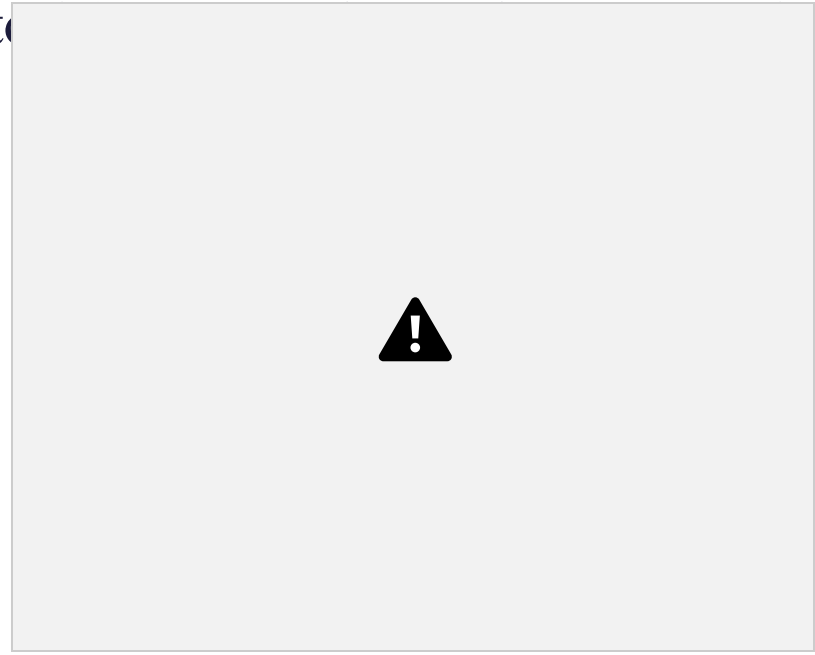
- Factor o Ratio de Aprendizaje
- Factores altos aumentan la velocidad, pero también el **riesgo** de caer en Óptimos locales.

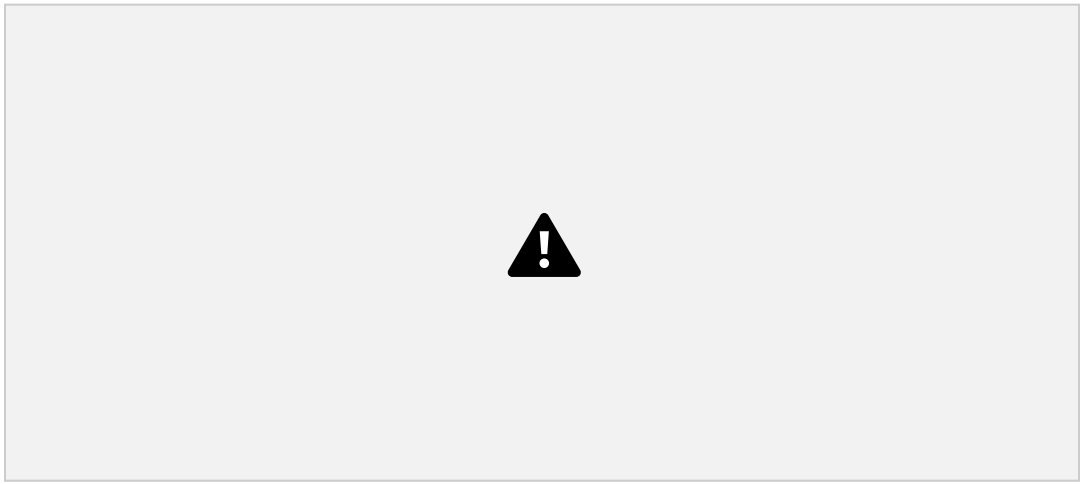
8.Proceso de Aprendizaje (I)

Consta de dos etapas **forward** y **backward**, partiendo de una red con pesos inicializados (normalmente de forma aleat nto con valores de entrada y salida esperada.

Forward: a partir de unos valores de entrada, se realizan los **cálculos intermedios**, se almacenan para cálculos posteriores, y se obtiene un resultado.

- Se compara con el **esperado** y se computa el error**.**

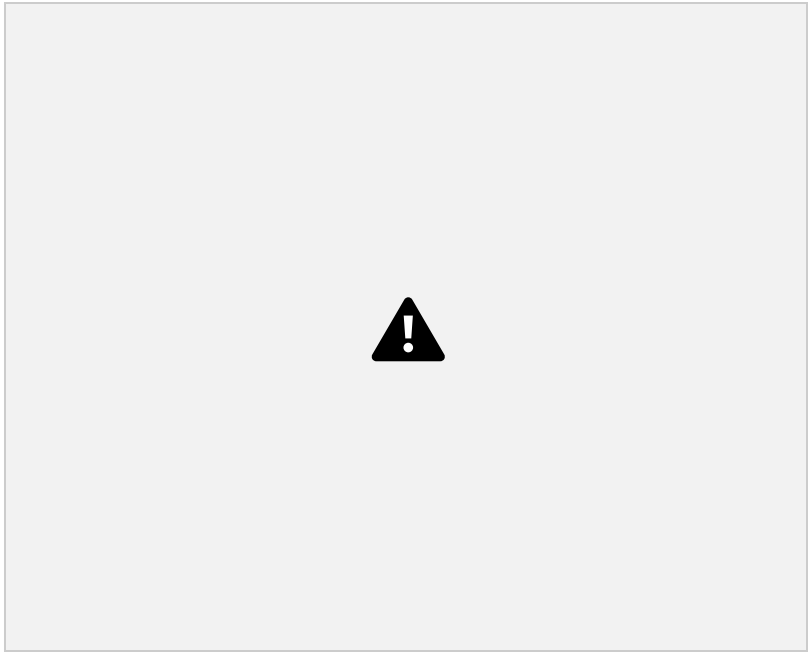




(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

8.Proceso de Aprendizaje (II)

Backward: Se aplica la regla de la cadena de forma que en cada neurona se actualicen sus parámetros asociados (pesos) en base a



El ratio de aprendizaje: Suele ser una constante (normalmente 0.05) 1) El gradiente computado para esta neurona: Es la **derivada parcial** del error total (descomponiendo la fórmula) con respecto al peso de la neurona a actualizar.

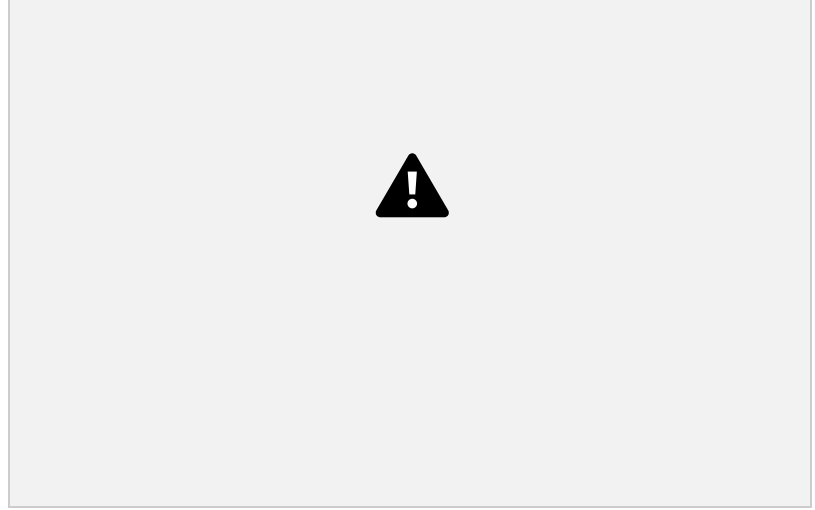
(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

9.Estimar el error

Para una red con N neuronas de salida, podemos estimar el error usando el error cuadrático medio.

… pero luego veremos que otras fórmulas de error son más apropiadas.

Más adelante profundizaremos en la formulación y los conceptos de gradiente descendiente y *Back propagation*.

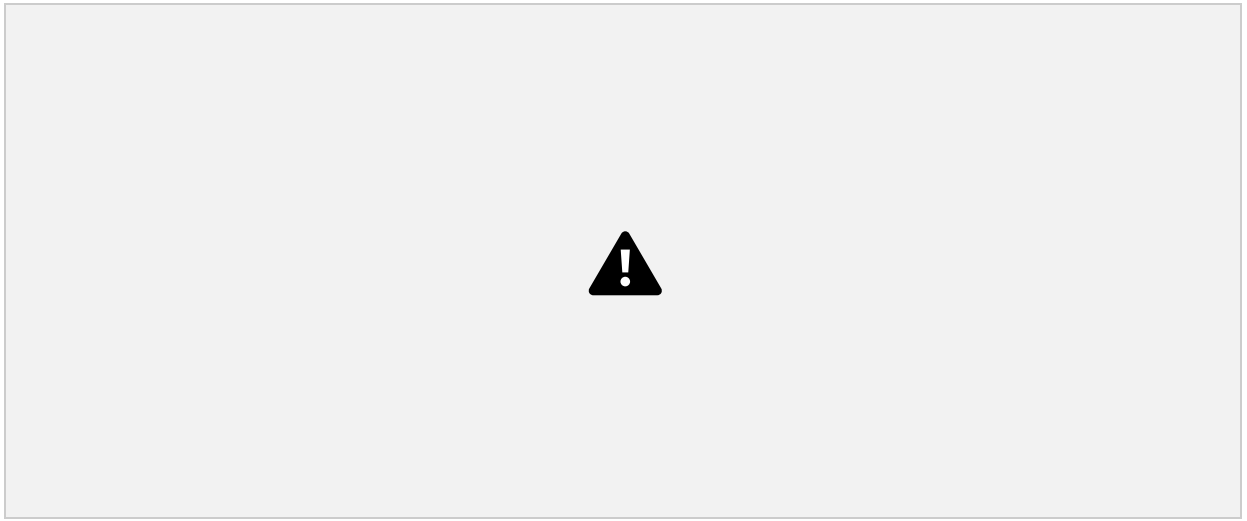


1.Fed Forward

A pesar de que se verán en la última sesión, es importante conocer l ipales **topologías** de redes.

Las llamadas **Feed Forward** se corresponden con el modelo tradicional, y se definen como **redes sin ciclos** ni **bucles** durante la clasificación.

- El flujo de datos pasa de las capas de entrada a las intermedias, y finalmente a la capa de salida donde se obtiene el resultado final.



(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

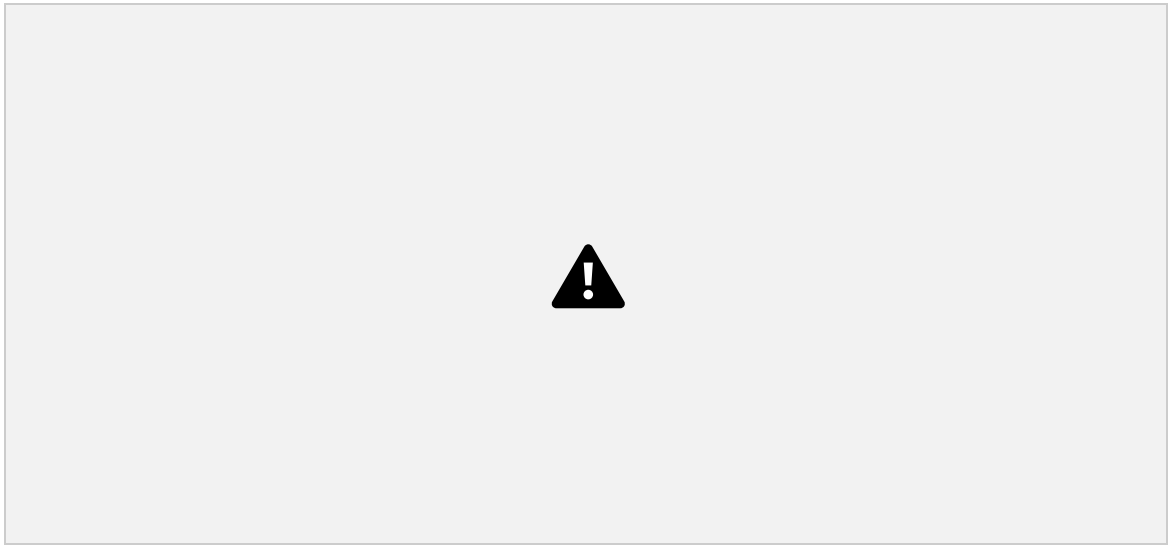
2. Recurrent Neural Networks

Las redes Recurrent Neural Networks (RNNs), por el contrario, permiten la **aparición de bucles** en su arqu

Estos bucles tienen diferentes implicaciones, como la capacidad de aprovechar la temporalidad de

una secuencia de entrada a través de un **estado.**

- Podemos decir que una red puede procesar una imagen en el instante *t*, y utilizar este último estado para procesar la siguiente imagen capturada en el instante *t+1*.



3.Características de FFN y RNN

Los resultados usando RNNs han sido especialmente buenos hasta el momento en el reconocimiento d

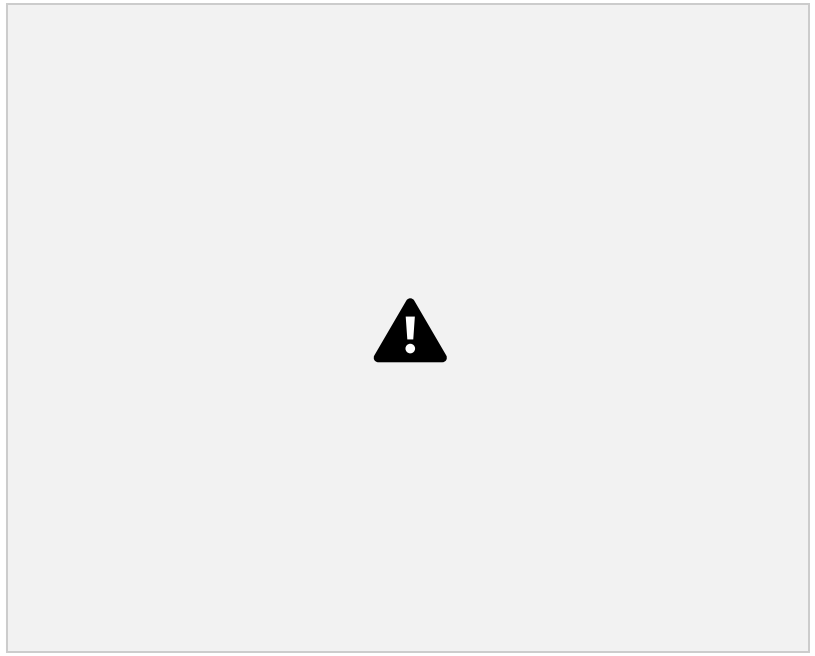
- Línea De Inv Abierta

Sinembargo, son **difíciles de entrenar y en muchos casos no** resulta práctico su uso ante problemas que puedan resolverse con redes Feed Forward.

- Recomendable únicamente para problemas que requieran esa potencia.

Existe un tercer tipo de red con enlaces no dirigidos llamadas

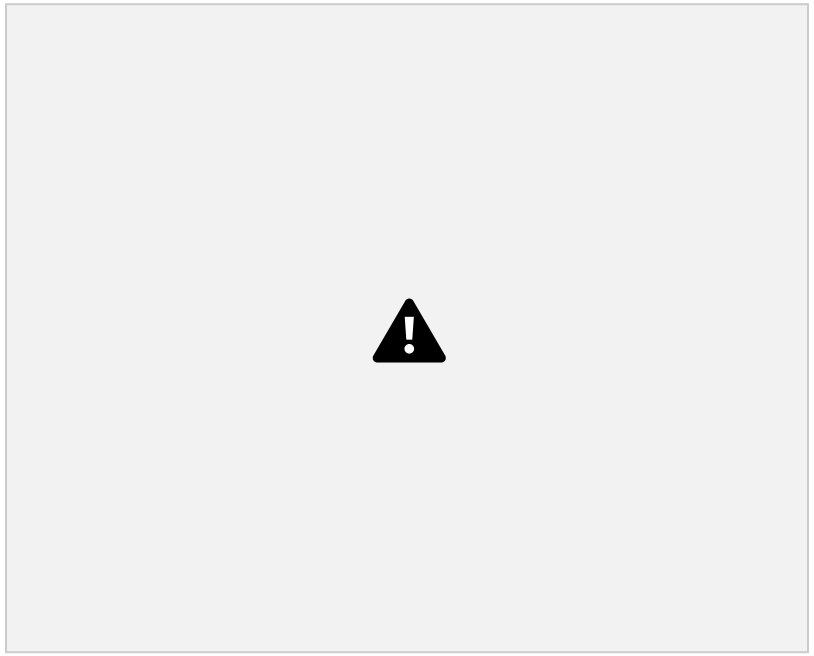
Symmetric o **Restricted Boltzmann Machines**.



1.Contexto

Portanto, el **resultado**, a nivel de neurona, se

computa como:
- Valores de entrada
  - Pesos asociados
  - Parámetro de sesgo



Entonces, para una neurona conectada con *n* neuronas de una capa anterior tendremos:

- ***n* pesos**, pero
- un **único** parámetro de **sesgo** *(b)*

(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

2.Aprendizaje y activación

Junto a su arquitectura, los **pesos** y **sesgo** codifican el conocimiento aprendido por una red neuronal:

- El proceso de aprendizaje tiene por objetivo el establecimiento de estos parámetros.
- Para que este proceso fluya correctamente

debemos prestar atención al **proceso de activación**

- La **activación** de una neurona determina el valor computado que esta neurona propaga a las siguientes capas.

(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

3.Función de activación

La función de activación sirve como **abstracción** de qué neuronas **intervienen** en el proceso de clasificación, aunque tiene otros objetivos ¯ Simplifica y orienta el proceso de aprendizaje

Partiendo del modelo más sencillo (Perceptrón), vimos que las neuronas únicamente propagaban valores binarios, pero a continuación veremos **diversas funciones** (y sus implicaciones)

(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

3.Función de activación

Sigmoidal(I)

La función de activación usada en los perceptrones parte de la

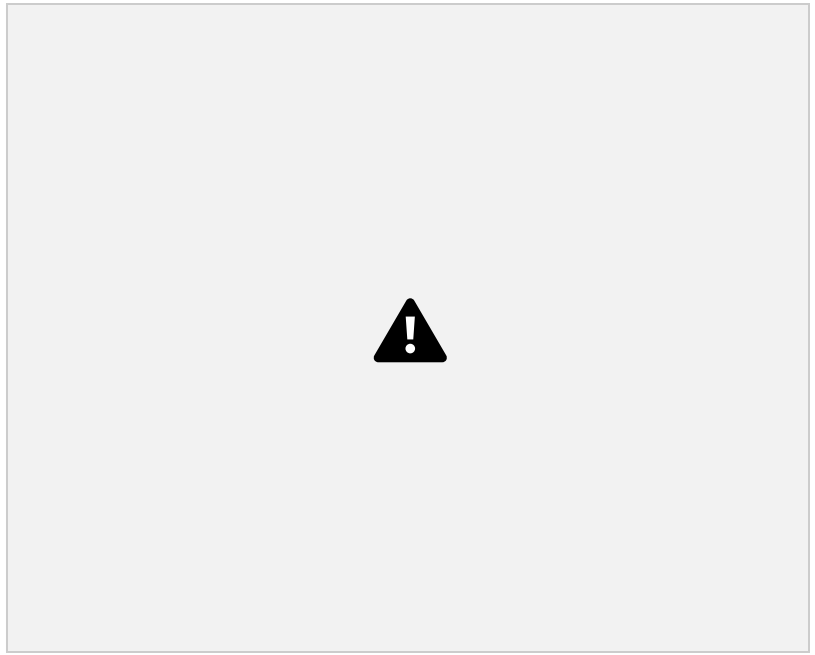
limitación de resolver **problemas lineales**.

Además de esta función, se ha visto la **sigmoidal**, la cual aporta una mayor versatilidad para resolver problemas complejos y no solamente problemas lineales.

3.Función de activación

Sigmoidal(I)

Una de las principales ventajas de la función sigmoidal es la **acotación** de la salida en el rango[0,1].



Sin embargo, esta función tiende a tener **pequeñas variaciones**

ante los cambios en los parámetros de entrada.

(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

3.Función de activación

Sigmoidal(II)

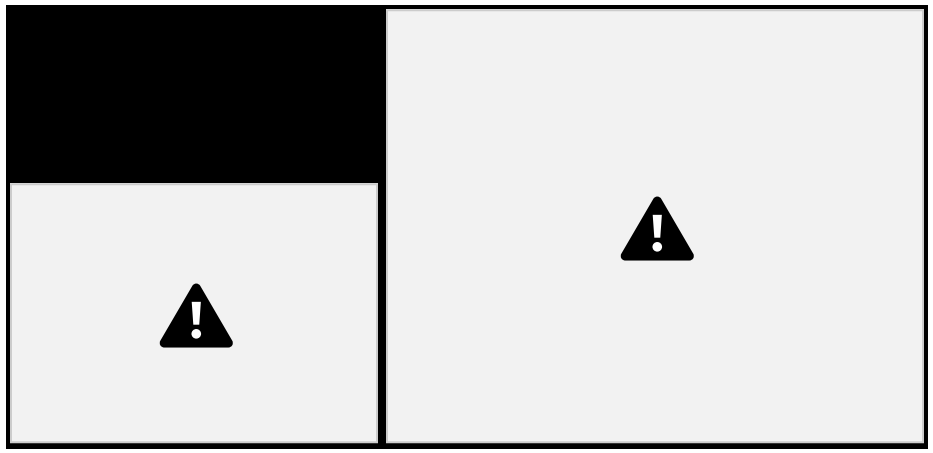
Este hecho afecta al problema conocido como *vanishing gradient*, dificultando el proceso de aprendizaje.

- ***Vanishing gradient:*** durante el aprendizaje algunas variaciones en los parámetros no se traducen en diferencias lo suficientemente significativas como para tomarse en cuenta.

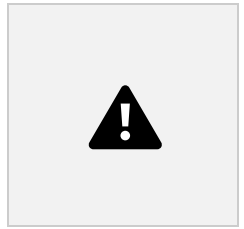
●Parecido a problemas de optimización, si el cambio

que hacemos no tiene grandes efectos en el resultado vamos a descartar el cambio.

A pesar de estas limitaciones, la función sigmoidal sigue siendo ampliamente utilizada.

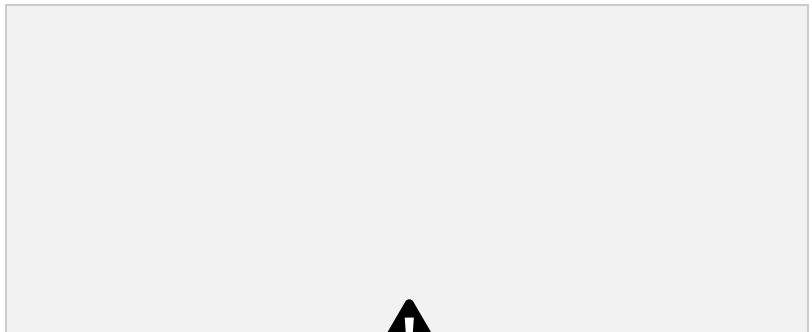


3.Función de activación



ReLu(I)

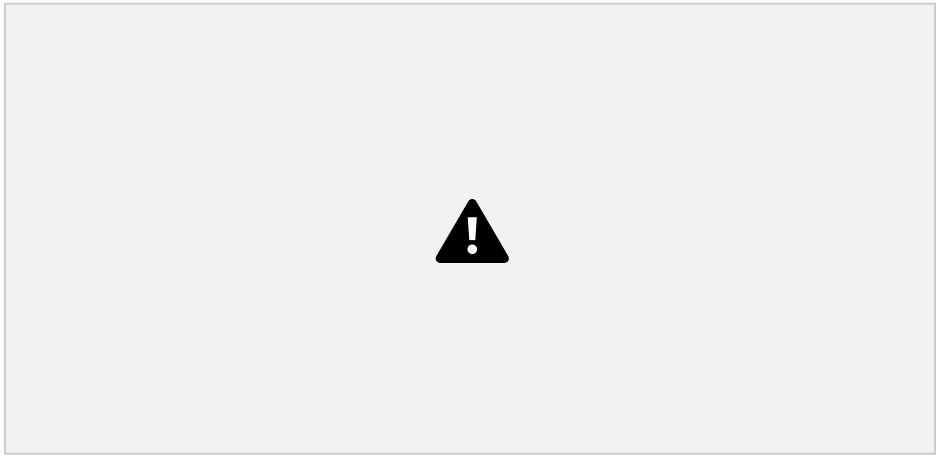
La función de activación más usada en Deep Learning es la conocida como **ReLu** (*Rectified Linear Unit*)**.**



- Valor de entrada si es positivo, o 0 si es cero o negativo.

Puede pensarse que presenta las mismas restricciones que las funciones lineales…

pero ReLu **no es lineal** y sus combinaciones tampoco lo son.

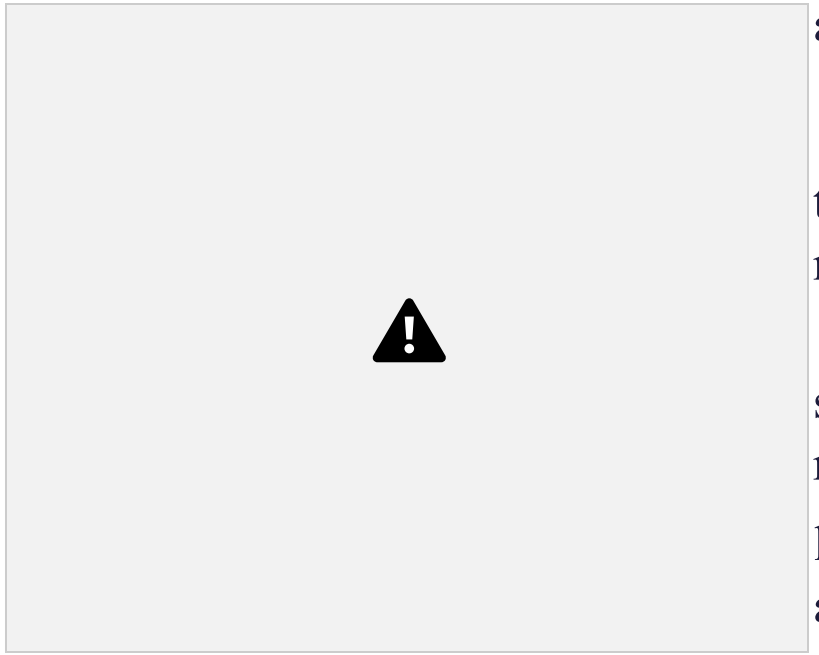


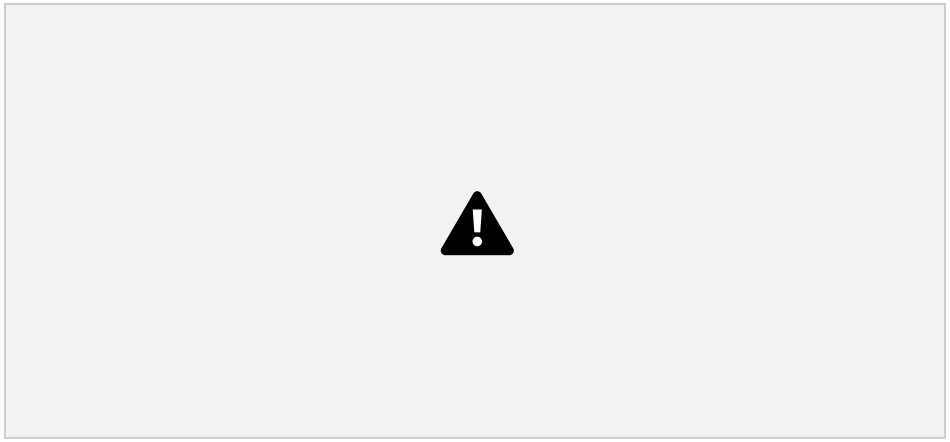
3.Función de activación

ReLu(I)

La principal ventaja de ReLu es su **aportación al proceso de clasificación y aprendizaje:**

- Funciones como la sigmoidal se traducen en clasificaciones dónde la mayor parte de las neuronas intervienen en el proceso→ **activaciones densas**.
- Ineficiente: las clasificaciones sobre subconjuntos de datos de neuronas obtienen mejores resultados.
- ReLu fuerza la no activación de las neuronas con valores negativos, lo que favorece **activaciones dispersas**→ se obtiene el mismo resultado pero más rápido.

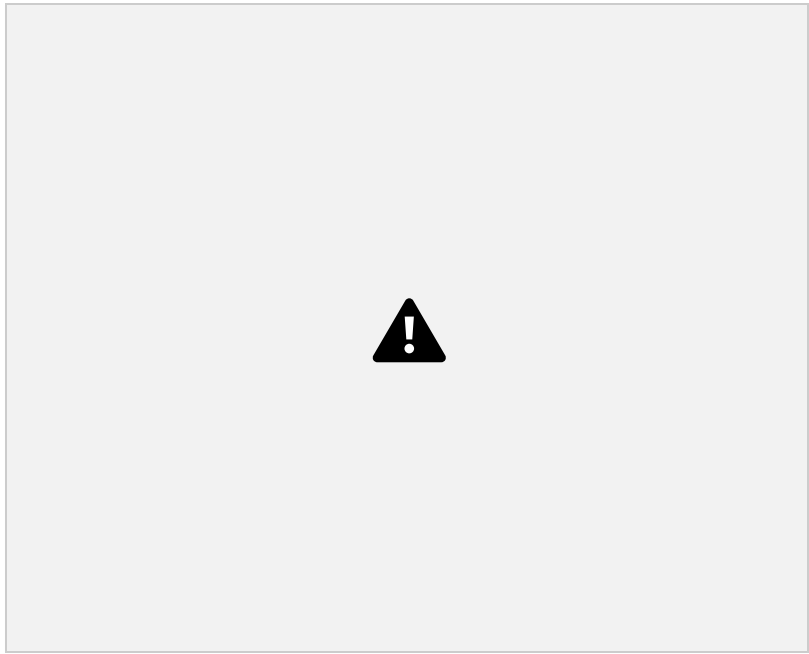




3.Función de activación

ReLu(II)

A pesar de la simplicidad de ReLu, su introducción en las redes neuronales ha tenido un **enorme impacto**.



- Algunos trabajos muestran cómo el aprendizaje se **acelera hasta 6 veces** al utilizar ReLu en lugar de la función sigmoidal o variaciones de la misma.
- La computación también es más eficiente.

También posee algunas desventajas, como el llamado ***Dying ReLu problem.***

- Algunas neuronas dejan de aprender y quedan bloqueadas en un estado de no activación.

(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

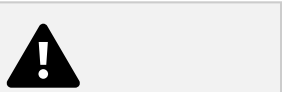
3.Función de activación

ReLu(II)

- A pesar de la simplicidad de ReLu, su introducción en las redes neuronales ha tenido un **enorme impacto**.
  - Algunos trabajos muestran cómo el aprendizaje se **acelera hasta 6 veces** al utilizar ReLu en lugar de la función sigmoidal o variaciones de la misma.
  - La computación también es más eficiente.
- También posee algunas desventajas, como el llamado ***Dying ReLu problem.***

y quedan bloqueadas en un estado de no activación.

Estasneuronasdejan deaprender



1.Contexto

- La capa final (*output*) se utiliza para **interpretar el resultado** del proceso de clasificación.

Además de conocer la categoría o clase que debemos asociar a la entrada proporcionada, la interpretación se refiere a las **diferencias** entre valores obtenidos.

- Tanto si utilizamos la función Sigmoidal como ReLu, las neuronas de la capa final presentarán unos valores que no son válidos para una interpretación directa.

(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

2.Interpretar resultados

Si estamos utilizando una función de activación **ReLu**, cada neurona tendrá valores en el **rango [0,inf]** y podemos pensar en seleccionar directamente la neurona final con **mayor activación** como clase.

Si utilizamos la función **Sigmoidal** partimos de un escenario similar… pero controlado en el **rango [0,1].**

El problema lo tenemos si queremos interpretar el resultado como **probabilidad de pertenencia** de la entrada a cada una de las clases.

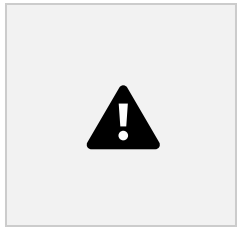
(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

3.Softmax

Para solvertar esta carencia se i bién llamada *Multinomial Logistic Maximun*

Esta regresión parte de la siguiente premisa:

- Todas las **clases** o categorías son **excluyentes entre sí**; es ampliamente utilizada en otros campos como SVMs

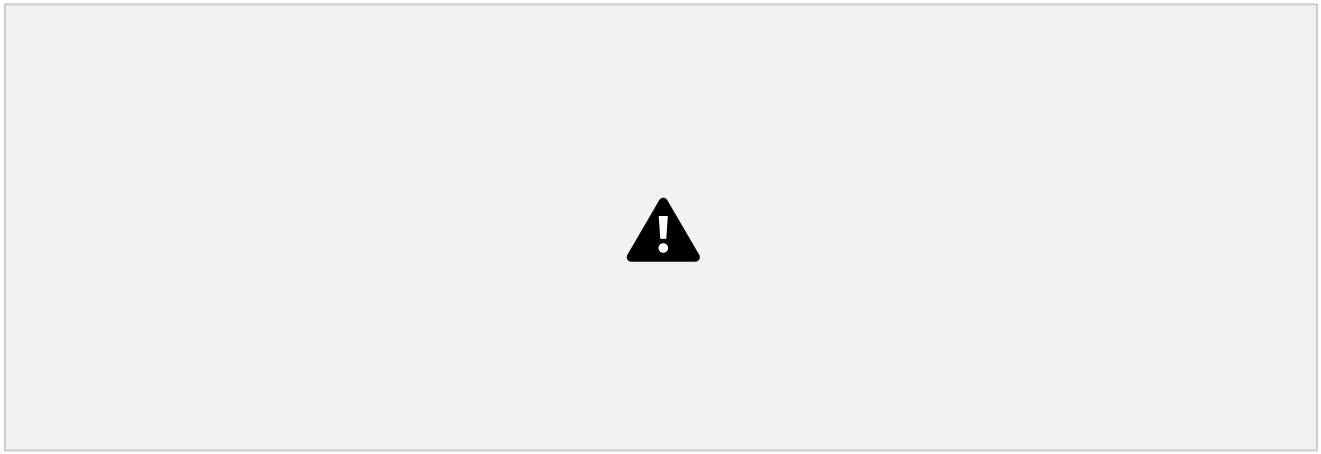


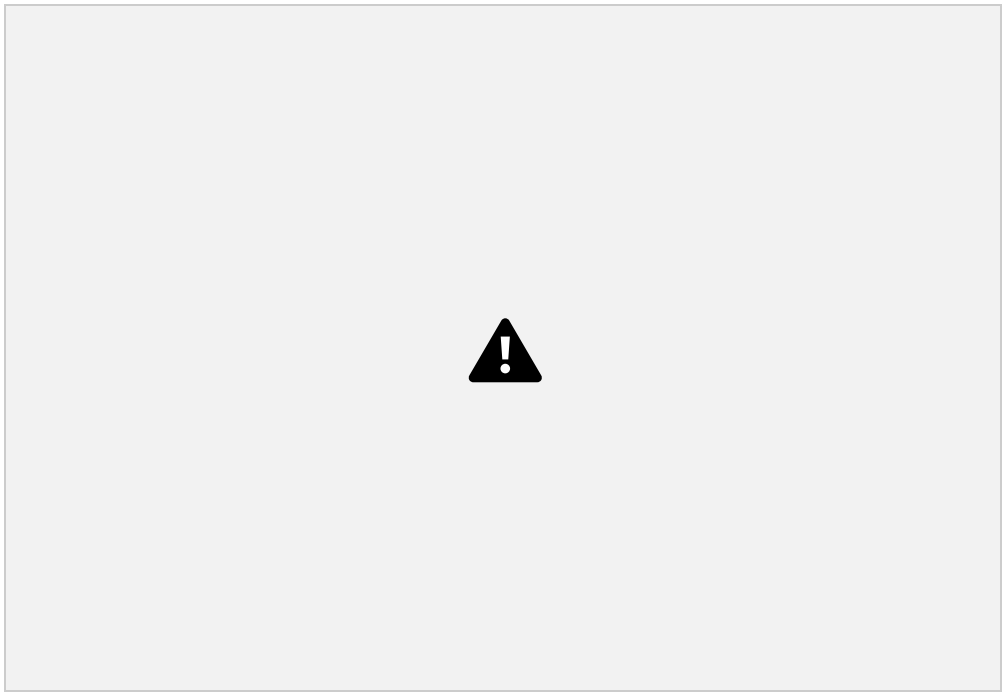
4.Clases de salida

La fórmula

parte de

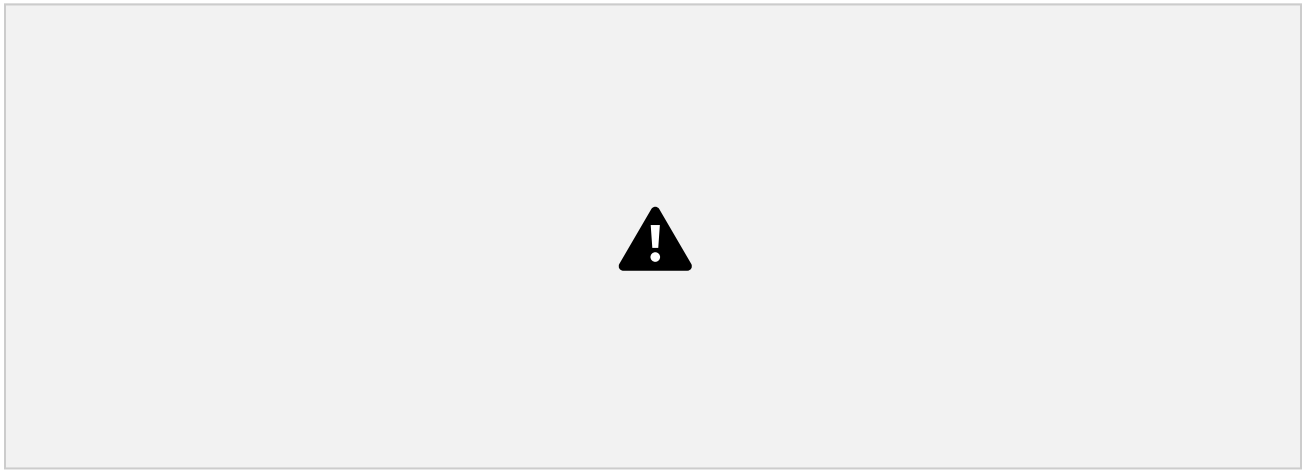
valores de salida, los cuales se coresponden con las posibles clases del dominio y generan un nuevo **vector normalizado.**

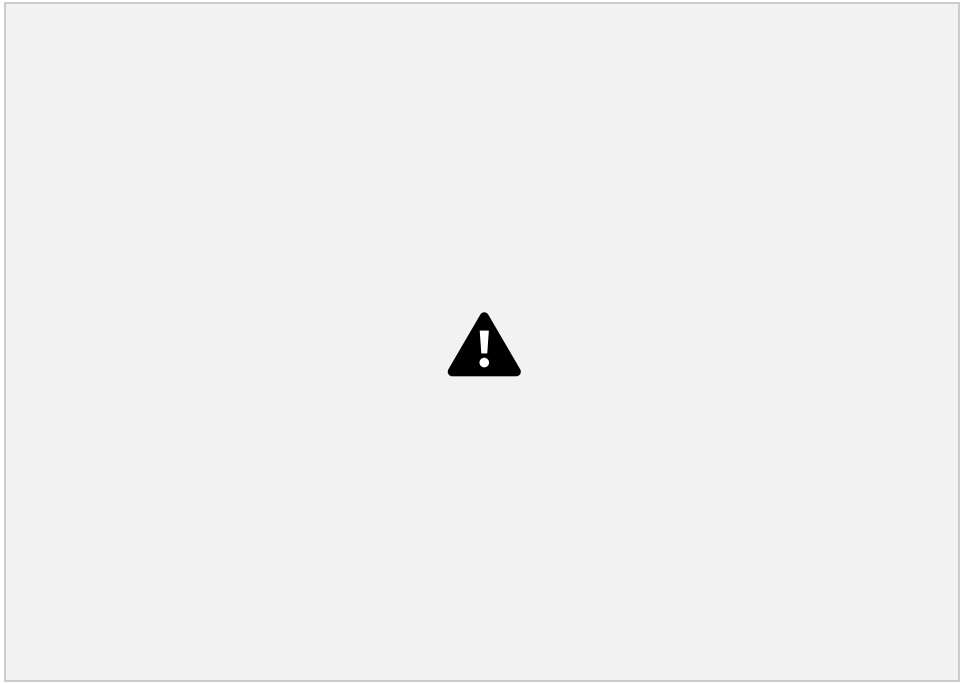




4.Clases de salida

La fórmula parte de valores de salida, los cuales se coresponden con las posibles clases del dominio y generan un nuevo **vector normalizado.**





(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

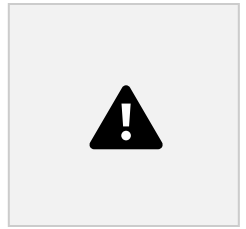
5.Interpretación de salida

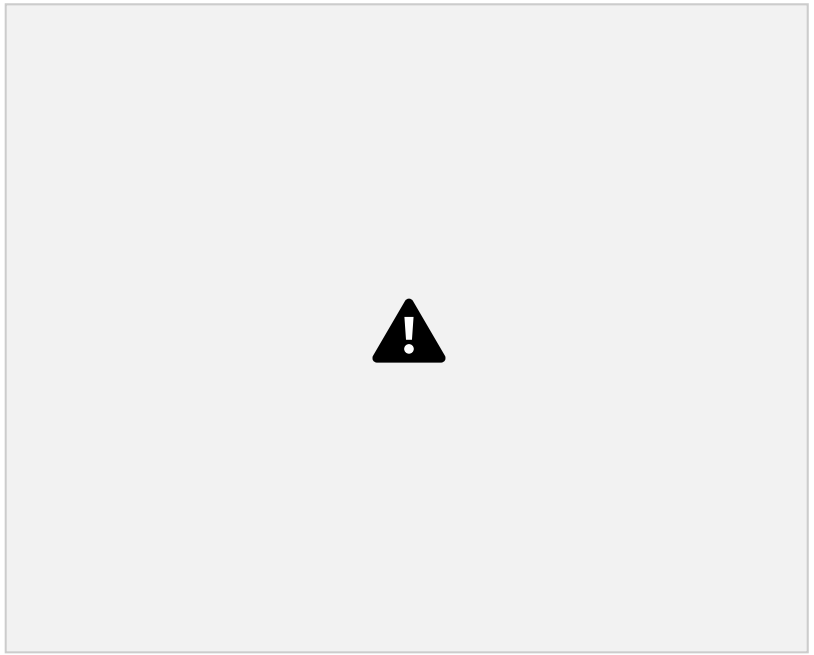
El nuevo vector obtenido tras el uso de Softmax ya puede **interpretarse** como la probabilidad de pertenencia a cada una de las clases del dominio.

- Para que el uso de Softmax sea efectivo, debemos trabajar con la **función de coste** ***cross-entropy***,la cual se puede integrar fácilmente en la formulación para realizar el aprendizaje.

En esta función, el término O_i representa la salida I tras el uso de Softmax, y T representa el valor real (1 ó 0).

(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com





1.Contexto

En lugar de aprender entrada a entrada, se suele utilizar **pequeños conjuntos de entrada**, lo

t descent.

Backpropagation consiste de forma iterativa la corrección de parámetros con Gradiente Descendiente:

- Para optimizar los cálculos se utiliza la regla de la cadena.

La aplicación de *Back propagation* se realiza **capa a capa** desde el final, de forma que los primeros cálculos

afectan a la capa de salida y la última de las capas ocultas.

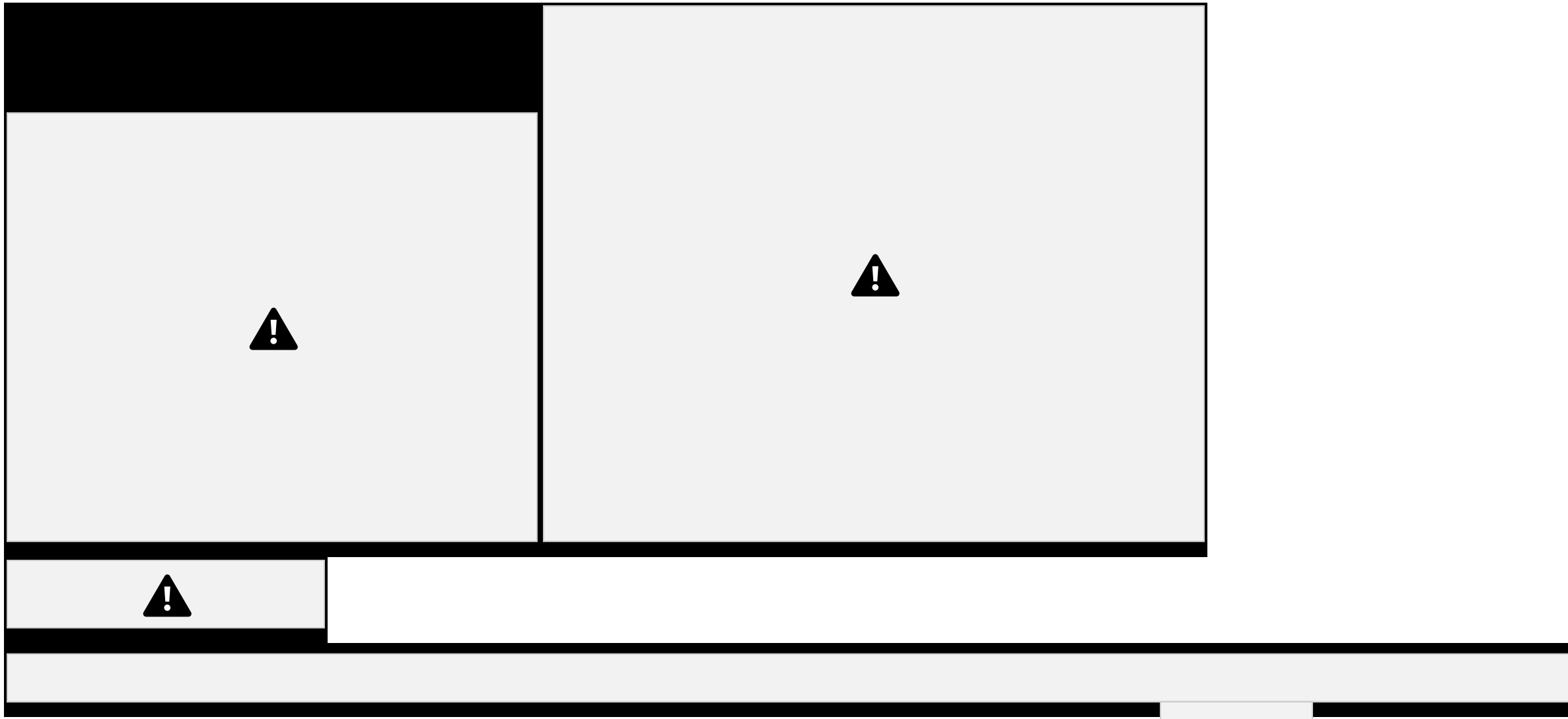
Para entender sus implicaciones, nos remitiremos a un ejemplo usando la topología de la siguiente página,

donde se parte de unos pesos y sesgo ya establecidos.

- Usamos la función de activación **Sigmoidal** y el **error cuadrático medio** como función de Coste.
- https:/mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

2.Ejemplo

Arquitectura de la red



2.Ejemplo

Arquitectura de la red

-Dos neuronas de entrada *(i)*

-Dos neuronas ocultas *(h)*

-Dos posibles categorías *(o)*

-*w* son los pesos

-Abajo están los bías o coste que son únicos

(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

2.Ejemplo

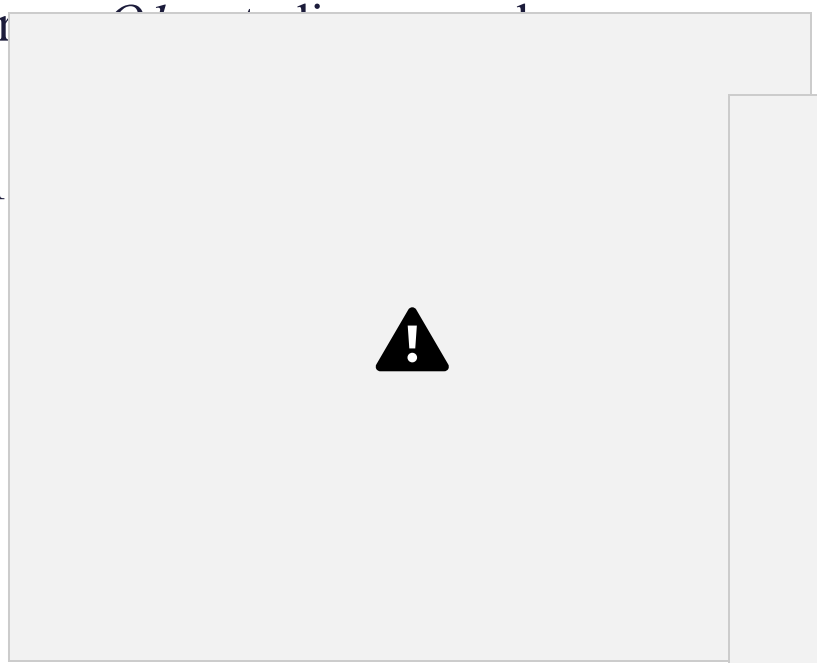
Valores de capas ocultas

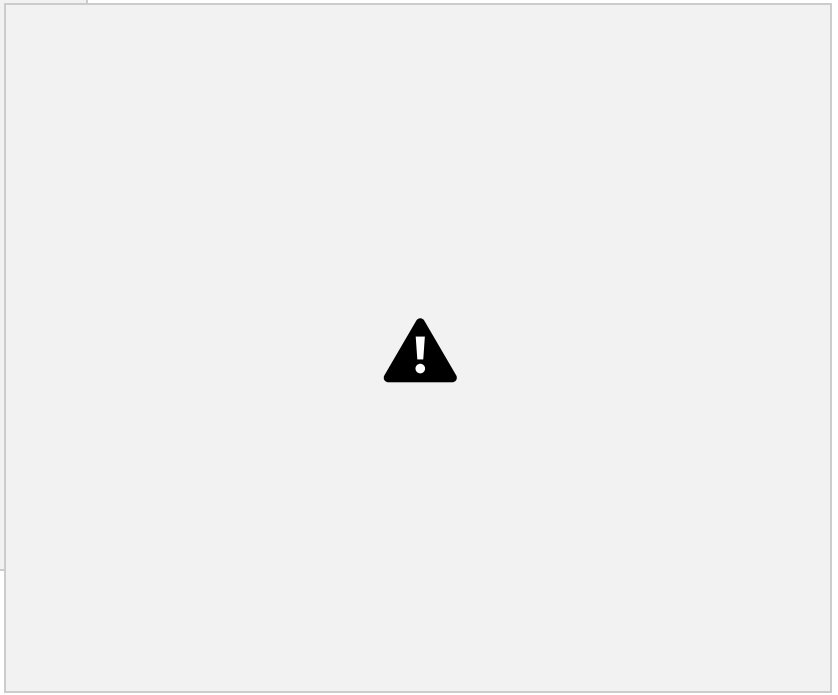
Centrándonos en la neur

de aprendizaje a partir

salida esperada es[0.01

El paso ***forward***, los valores de *h1* y *h2* se establecen a





partir de la suma de pesos por la entrada más sesgo, y en

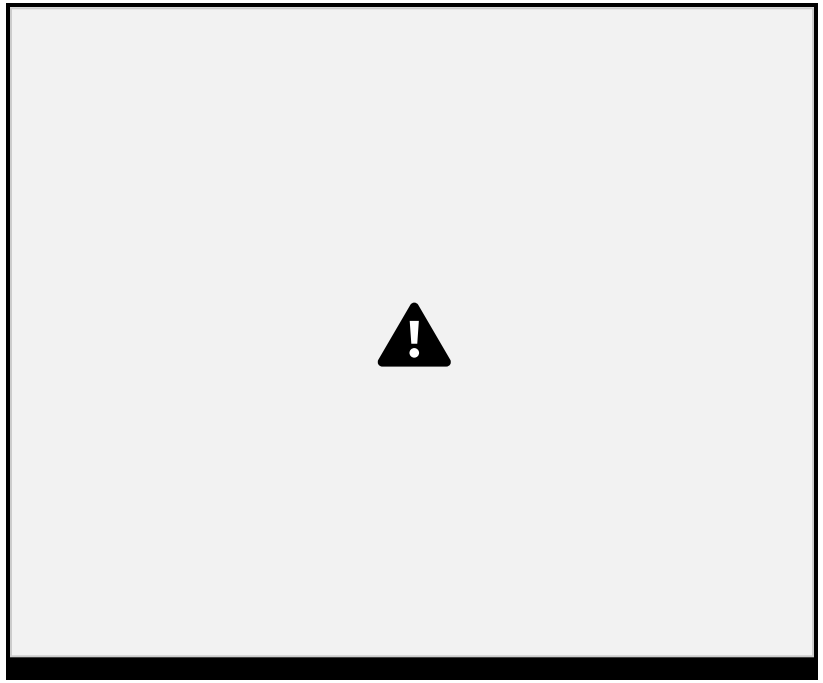
este caso usamos la **función de activación** Sigmoidal:

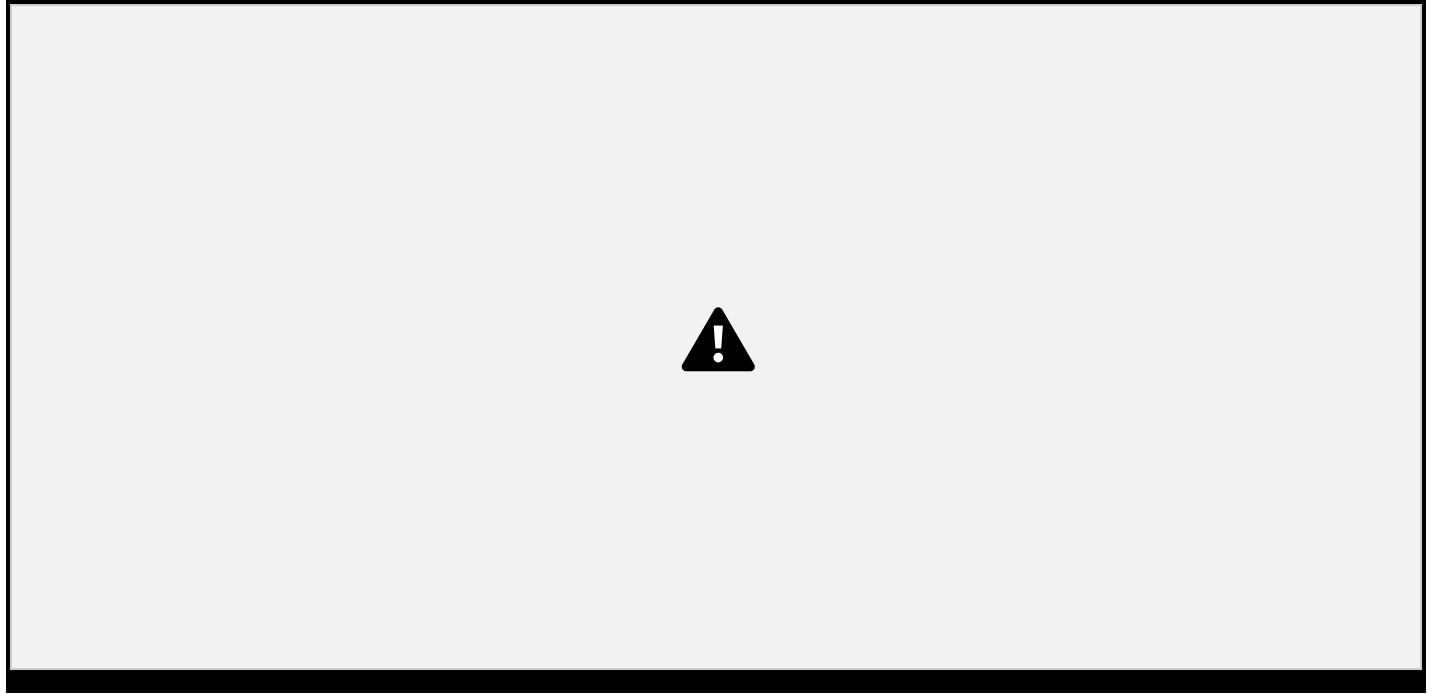
- f(h1)=sigmoid(0.15*0.05+0.2*0.10+0.35)
- f(h1)=sigmoid(0.3775)=0.593269992
- f(h2)=sigmoid(0.25*0.05+0.3*0.10+0.35)
- f(h2)=sigmoid(0.3925)=0.5958843378

(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com

2.Ejemplo

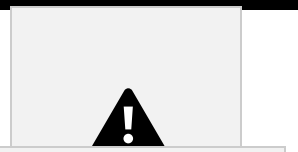
Calcular el error

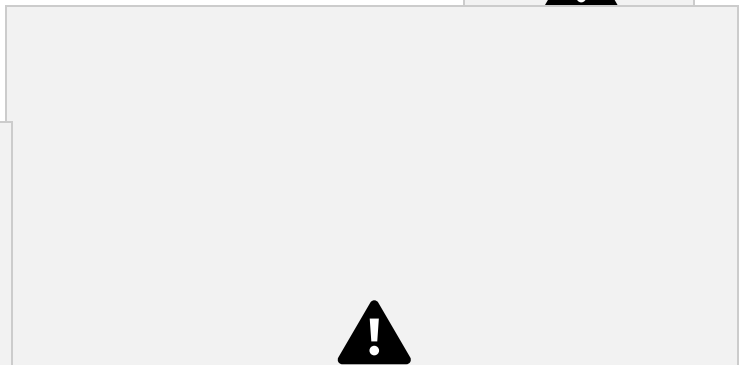




2.Ejemplo

Calcular el error





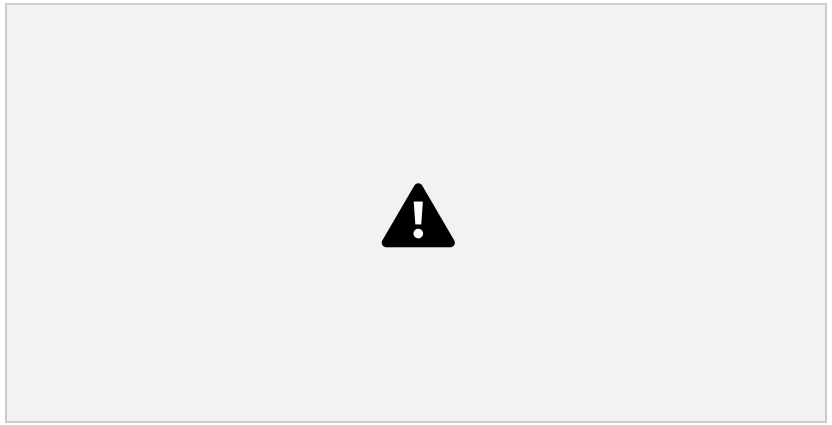
- Tras ello pasamos a calcular los valores reales

obtenidos en la Salidade *O1* y *O2*:

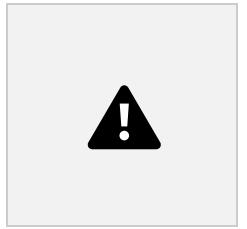
- f(O1)=sigmoid(0.4*0.593269 92+0.45*0.5958 43 78+0.6)
- f(O1)=sigmoid(1.105905967)=0.75136507
- f(O2)=sigmoid(0.5*0.593269 92+0.5 *0.5958 43 78+0.6)
- f(O2)=sigmoid(1.2 437138)=0.7 2928465

- Y calculamos el **error** (utilizando el error cuadrático) ya que no es lo esperado del target:

  - Error total=E(O1)+E(O2)
  - $E(O1)=½\ (0.01–0.75136507)^2=0.27481\ 083$
  - $E(O2)=½\ (0.9\ –0.7\ 2928465)^2=0.0.0235\ 60\ 26$
  - Error total=0.298371 09



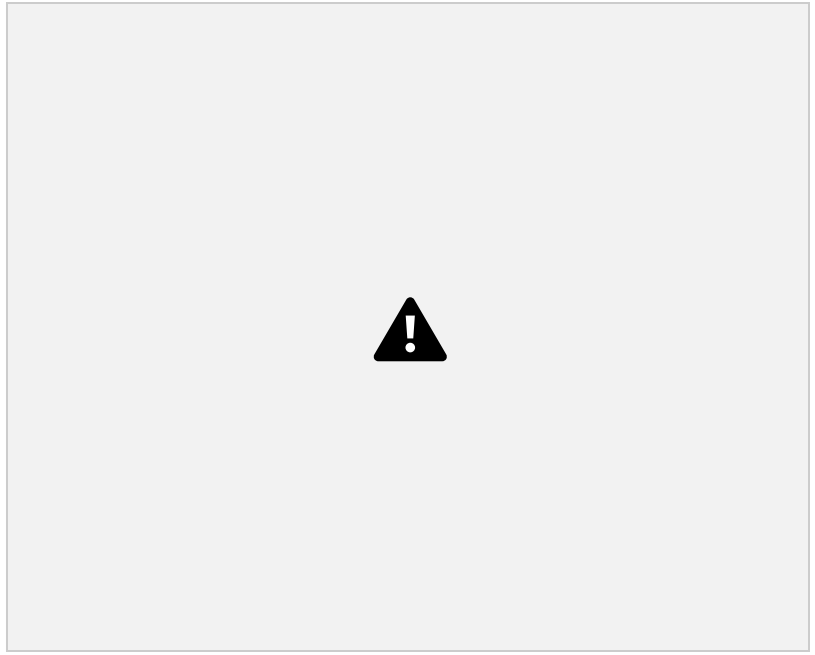
(+51) 976 760 803 www.datayanalytics.com info@datayanalytics.com



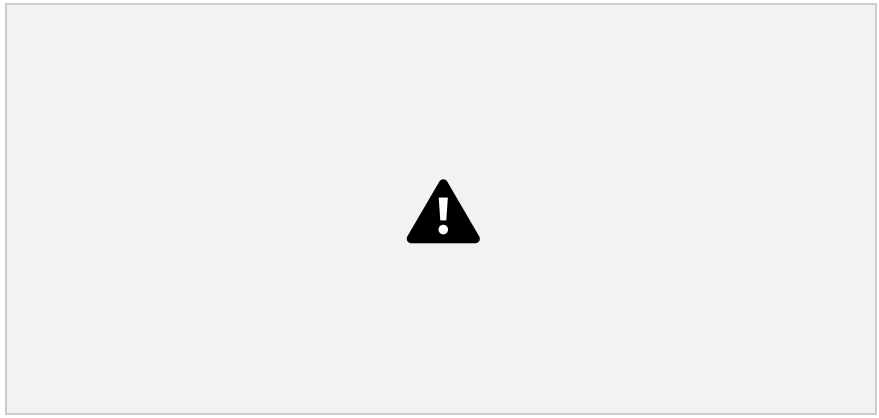
2.Ejemplo

Etapa Backwards

- Como no hemos tenido los valores esperados tenemos que



- corregir la red con una rectificación para poder aproximarnos al erro resperado (target establecido)[0.1,0.9]
- Calculamos el **gradiente** para *w5* (valor inicial 0.4), la cual



- afecta únicamente al cálculo de O1.
- Hacemos uso de la regla de la cadena para descomponer, y
- obtenemos lo siguiente:

El objetivo es **simplificar** el cálculo en pequeñas partes que también se pueden **reutilizar** en otros cálculos