TECNOLÓGICO NACIONAL DE MÉXICO CAMPUS

CULIACÁN

INTELIGENCIA ARTIFICIAL

ALGORITMO DE BACKPROPAGATION

KAREN DANIELA CHIQUETE MARTÍNEZ

M.C JOSÉ MARIO RÍOS FÉLIX

Contenido

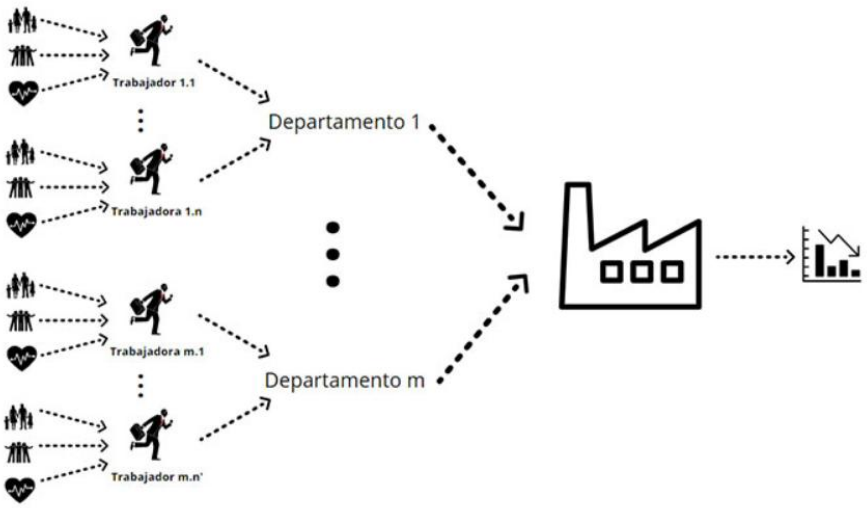
INTRODUCCIÓN

El algoritmo de *Backpropagation* constituye uno de los elementos centrales en el ámbito de la Inteligencia Artificial, debido a su papel esencial en el entrenamiento de redes neuronales artificiales. Este método permite ajustar los pesos internos de una red mediante la minimización del error, empleando gradientes obtenidos a través de la regla de la cadena. Gracias a dicha capacidad, las redes neuronales pueden aprender patrones complejos, realizar tareas de clasificación, detección y predicción, y resolver problemas no lineales presentes en diversas áreas tecnológicas.

Si bien los fundamentos teóricos de la retropropagación de errores surgieron en la década de los setenta, fue hasta el trabajo de Rumelhart, Hinton y Williams en 1986 cuando el algoritmo fue formalizado y difundido ampliamente. Este hito permitió entrenar redes neuronales multicapa de manera efectiva, superando las limitaciones del perceptrón simple, el cual no podía resolver problemas no linealmente separables, como la conocida operación XOR. La introducción del Backpropagation marcó un punto de inflexión que impulsó el resurgimiento del interés en las redes neuronales y sentó las bases del aprendizaje profundo moderno.

A pesar de los avances en métodos de optimización, como Adam, RMSProp y Adagrad, la retropropagación continúa siendo el fundamento matemático sobre el cual operan todos los algoritmos contemporáneos de aprendizaje profundo. Sin este mecanismo, arquitecturas actualmente indispensables como las redes convolucionales, las redes recurrentes y los modelos tipo transformer no podrían entrenarse de manera eficiente.

Por ello, el Backpropagation no solo se considera una técnica de entrenamiento, sino también un componente esencial para comprender la evolución y el funcionamiento de la Inteligencia Artificial actual.



FUNDAMENTOS TEÓRICOS

El algoritmo de *Backpropagation* se sustenta en diversos conceptos fundamentales de las redes neuronales artificiales y del cálculo diferencial. Estos elementos permiten comprender el proceso mediante el cual una red aprende a partir de datos y ajusta sus parámetros internos.

2.1 Redes Neuronales Artificiales

Las redes neuronales artificiales (ANN, por sus siglas en inglés) son modelos computacionales inspirados en el funcionamiento biológico del cerebro. Están compuestas por unidades llamadas neuronas artificiales, organizadas en capas de entrada, capas ocultas y una capa de salida. Cada neurona recibe señales, las pondera mediante pesos, aplica una función de activación y produce una salida. Estas estructuras permiten modelar relaciones complejas entre variables y resolver problemas no lineales (Haykin, 2009).

2.2 Neuronas Artificiales y Funciones de Activación

Una neurona artificial opera realizando una combinación lineal de sus entradas y aplicando posteriormente una función de activación. Esta función introduce no linealidad al modelo, lo que facilita que la red pueda aprender patrones complejos. Entre las funciones de activación más utilizadas se encuentran la sigmoide, la tangente hiperbólica, la ReLU (Rectified Linear Unit) y la Softmax para tareas de clasificación (Goodfellow et al., 2016). La elección de la función de activación influye directamente en la eficiencia del aprendizaje y en la propagación del gradiente.

2.3 Función de Pérdida

Para entrenar una red neuronal es necesario medir qué tan bien está realizando sus predicciones. Esto se logra mediante una función de pérdida o error que cuantifica la diferencia entre la salida producida por la red y la salida deseada. Las funciones de pérdida más comunes incluyen el error cuadrático medio (MSE) y la entropía cruzada (cross-entropy), siendo esta última ampliamente utilizada en problemas de clasificación (Goodfellow et al., 2016).

2.4 Descenso del Gradiente

El aprendizaje en redes neuronales se basa en la optimización iterativa de los pesos utilizando el método del descenso del gradiente. Este método calcula las derivadas parciales de la función de pérdida respecto a cada peso y ajusta dichos valores en la dirección en la que el error disminuye. El tamaño del paso realizado en cada iteración depende de la tasa de aprendizaje (*learning rate*), un parámetro crítico que determina la velocidad y estabilidad del entrenamiento (Haykin, 2009).

2.5 Regla de la Cadena

La base matemática del algoritmo de Backpropagation es la regla de la cadena, una propiedad del cálculo diferencial que permite obtener derivadas de funciones compuestas. En redes neuronales, esta regla facilita el cálculo del gradiente de la función de pérdida con respecto a los parámetros de cada capa, propagando el error desde la salida hacia las capas internas. Esta propiedad es esencial para entrenar redes profundas de manera eficiente (Rumelhart et al., 1986).

2.6 Redes Multicapa

Las redes neuronales multicapa (MLP) surgieron como una extensión del perceptrón simple, otorgándoles la capacidad de aprender problemas no linealmente separables. Estas arquitecturas se componen de una o varias capas ocultas y requieren un método eficiente para ajustar sus pesos. El Backpropagation fue el primer algoritmo que permitió entrenar redes multicapa de forma práctica, demostrando que podían resolver problemas complejos como el caso XOR (Rumelhart et al., 1986).

En conjunto, estos fundamentos establecen las bases teóricas que permiten comprender el comportamiento y la eficiencia del algoritmo de Backpropagation en el aprendizaje automático.

¿QUÉ ES BACKPROPAGATION?

El *Backpropagation* es un algoritmo de aprendizaje supervisado utilizado para entrenar redes neuronales artificiales mediante la actualización iterativa de los pesos internos del modelo. Su objetivo principal es minimizar la función de pérdida calculando cómo cada peso contribuye al error total, para posteriormente ajustarlos en la dirección que reduzca dicho error. Este procedimiento se basa en el cálculo eficiente de gradientes y constituye la base del entrenamiento de redes multicapa modernas.

Este algoritmo opera propagando el error desde la capa de salida hacia las capas intermedias, proceso que da origen a su nombre, ya que el error "retropropaga" a través de la red. Para lograrlo, emplea la regla de la cadena, lo que permite obtener derivadas parciales de funciones compuestas de manera eficiente, incluso cuando la red posee múltiples capas con miles de parámetros (Rumelhart et al., 1986).

La importancia del Backpropagation radica en que fue el primer método que permitió entrenar redes neuronales multicapa de forma efectiva. Antes de su introducción, el perceptrón multicapa no podía ajustarse adecuadamente, ya que no existía un procedimiento sistemático para calcular el impacto de los pesos internos sobre el error. Con este algoritmo, las redes pudieron aprender representaciones jerárquicas y resolver tareas previamente imposibles para modelos lineales, como el problema XOR y otras funciones no linealmente separables (Haykin, 2009).

En la práctica, el Backpropagation no actúa solo, sino que se combina con algoritmos de optimización como el descenso del gradiente o sus variantes modernas. En cada iteración del entrenamiento, el algoritmo calcula el error de la red, obtiene los gradientes de cada parámetro y actualiza los pesos con base en dichos gradientes. De esta manera, la red ajusta progresivamente sus representaciones internas hasta converger hacia una solución óptima o suficientemente buena (Goodfellow et al., 2016).

FUNCIONAMIENTO DEL BACKPROPAGATION PASO A PASO

El funcionamiento del algoritmo de Backpropagation se desarrolla a través de una serie de etapas que permiten entrenar una red neuronal ajustando sus pesos para reducir el error en las predicciones. Este proceso se lleva a cabo en ciclos llamados *epochs*, en los cuales la red procesa los datos una y otra vez hasta lograr una convergencia aceptable. El procedimiento puede dividirse en cuatro fases principales: *forward pass*, cálculo del error, retropropagación del error y actualización de pesos.

4.1 Forward Pass

En la primera etapa, la red neuronal recibe una entrada y la procesa capa por capa hasta obtener una salida. Cada neurona realiza una combinación lineal de sus entradas y aplica una función de activación, generando así la respuesta final del modelo. Este proceso permite evaluar la predicción actual con los pesos existentes, pero no implica aún ningún ajuste (Goodfellow et al., 2016).

4.2 Cálculo del Error

Una vez obtenida la salida de la red, se calcula la diferencia entre la predicción y el valor esperado mediante una función de pérdida. Esta función cuantifica qué tan bien o mal está desempeñándose la red. Las funciones de pérdida más empleadas son el error cuadrático medio (MSE) para problemas de regresión y la entropía cruzada (cross-entropy) para tareas de clasificación. El error calculado se utiliza como base para retropropagar las correcciones necesarias (Haykin, 2009).

4.3 Retropropagación del Error

En esta fase, el error obtenido en la salida se propaga hacia atrás a través de la red, desde la capa final hasta las capas anteriores. Utilizando la regla de la cadena, el algoritmo calcula las derivadas parciales de la función de pérdida respecto a cada uno de los pesos. Este procedimiento permite identificar qué tan responsable es cada peso del error total. Así, cada

neurona recibe un valor que indica la magnitud del ajuste requerido para mejorar el desempeño del modelo (Rumelhart et al., 1986).

4.4 Actualización de Pesos

Con los gradientes calculados, la red procede a modificar los pesos y sesgos mediante un algoritmo de optimización. La forma más básica consiste en aplicar el descenso del gradiente, actualizando los pesos de acuerdo con la siguiente regla:

$$w_{nuevo} = w_{actual} - \eta \frac{\partial E}{\partial w}$$

donde η es la tasa de aprendizaje y $\frac{\partial E}{\partial w}$ representa el gradiente del error respecto al peso. La elección adecuada de la tasa de aprendizaje es esencial para garantizar que la red avance correctamente hacia una solución óptima sin divergir ni estancarse.

En implementaciones modernas, este proceso puede realizarse utilizando variantes del descenso del gradiente como Stochastic Gradient Descent (SGD), Adam o RMSProp, que mejoran la estabilidad y velocidad del entrenamiento (Goodfellow et al., 2016).

Las funciones de activación desempeñan un papel fundamental en el funcionamiento de una red neuronal, ya que introducen no linealidad en el modelo y permiten que la red aprenda relaciones complejas entre las variables de entrada y salida. Su elección influye directamente en la eficiencia del proceso de Backpropagation, afectando la propagación del gradiente, la velocidad de aprendizaje y la capacidad del modelo para generalizar.

FUNCIONES DE ACTIVACIÓN Y SU IMPACTO EN BACKPROPAGATION

5.1 Importancia de la No Linealidad

Sin funciones de activación no lineales, una red neuronal multicapa se comportaría de manera equivalente a un perceptrón simple, incapaz de resolver problemas no linealmente separables. La no linealidad permite que la red identifique patrones complejos como bordes, formas, variaciones temporales o relaciones abstractas. Además, hace posible que las redes profundas aprendan representaciones jerárquicas a lo largo de múltiples capas (Goodfellow et al., 2016).

5.2 Funciones de Activación Comunes

a) Sigmoide

La función sigmoide produce valores entre 0 y 1 y fue una de las más utilizadas en las primeras redes neuronales. Sin embargo, su derivada es muy pequeña cuando la entrada es grande o pequeña, lo que genera el problema conocido como *vanishing gradient* (gradiente desvanecido) (Haykin, 2009).

b) Tangente Hiperbólica (tanh)

Similar a la sigmoide, pero con un rango de salida entre –1 y 1. Presenta una saturación menos severa que la sigmoide, pero sigue sufriendo del problema del gradiente desvanecido.

c) ReLU (Rectified Linear Unit)

La función ReLU definió un avance importante en el aprendizaje profundo al permitir que el gradiente fluyera de manera más eficiente y evitar saturación en la parte positiva. Su simplicidad y eficacia la han convertido en la función estándar en redes profundas (Goodfellow et al., 2016).

d) Funciones variantes de ReLU

Incluyen Leaky ReLU, ELU y SELU. Estas variantes buscan resolver el problema del "neurona muerta", donde algunas neuronas dejan de activarse completamente al quedar atrapadas en valores negativos constantes.

e) Softmax

Utilizada principalmente en la capa de salida para tareas de clasificación multiclase. Convierte un vector de valores en una distribución de probabilidades, facilitando la interpretación de los resultados.

5.3 Impacto en el Backpropagation

La elección de la función de activación afecta directamente cómo se propaga el gradiente durante el entrenamiento. Algunas de las consecuencias más relevantes son:

• Gradiente Desvanecido

Ocurre cuando las funciones de activación saturan sus valores, lo que provoca que sus derivadas sean cercanas a cero. En este caso, los gradientes que deben propagarse hacia capas profundas se vuelven muy pequeños, impidiendo un aprendizaje adecuado. Este fenómeno afectó durante décadas la capacidad de entrenar redes profundas, especialmente con sigmoide y tanh (Rumelhart et al., 1986).

• Gradiente Explosivo

Contrario al gradiente desvanecido, en algunas arquitecturas los gradientes pueden aumentar de manera exponencial, generando inestabilidad numérica. Aunque menos común, debe controlarse mediante técnicas de normalización o *gradient clipping*.

• Velocidad de Convergencia

Funciones como ReLU permiten un flujo de gradiente más estable y aceleran el aprendizaje, logrando que las redes profundas converjan más rápido que con funciones tradicionales.

• Capacidad de Generalización

La función de activación también influye en la forma en que el modelo abstrae patrones. Por ejemplo, ReLU facilita la extracción de características en redes convolucionales, mientras que softmax es ideal para probabilidades en clasificación.

5.4 Consideraciones para Seleccionar una Función de Activación

La selección adecuada depende del tipo de red y del problema a resolver. En general:

- ReLU y sus variantes son preferidas en redes profundas debido a su estabilidad y eficiencia.
- Sigmoide suele reservarse para salidas binarias.
- Tanh se utiliza cuando se requiere una salida centrada en cero.
- Softmax es indispensable para clasificación multiclase.

Una elección inapropiada puede ralentizar el aprendizaje, causar estancamiento o impedir la convergencia.

VENTAJAS Y DESVENTAJAS DEL BACKPROPAGATION

El algoritmo de Backpropagation es una herramienta esencial en el entrenamiento de redes neuronales, y su adopción masiva se debe a los múltiples beneficios que ofrece en términos de eficiencia y precisión. Sin embargo, también presenta limitaciones importantes que deben considerarse al diseñar y entrenar modelos de aprendizaje profundo.

6.1 Ventajas del Backpropagation

a) Eficiencia computacional

El algoritmo permite calcular gradientes de manera rápida y precisa gracias al uso de la regla de la cadena. Este enfoque reduce significativamente el costo computacional en comparación con métodos de entrenamiento previos, lo que facilita trabajar con redes neuronales de varias capas (Goodfellow et al., 2016).

b) Capacidad para entrenar redes profundas

Backpropagation hizo posible entrenar redes multicapa, permitiendo que estas redes aprendieran representaciones jerárquicas complejas. Este avance fue clave para el surgimiento del aprendizaje profundo y el desarrollo de arquitecturas modernas como CNN, RNN y Transformers (Haykin, 2009).

c) Flexibilidad

El algoritmo es compatible con diversas arquitecturas neuronales, funciones de activación y métodos de optimización. Su versatilidad permite adaptarse tanto a tareas de clasificación como de regresión y análisis de secuencias.

d) Mejora continua basada en gradientes

A través de la minimización de funciones de pérdida diferenciables, Backpropagation permite un aprendizaje continuo, ajustando los pesos conforme la red se enfrenta a nuevos datos durante el entrenamiento.

6.2 Desventajas del Backpropagation

a) Gradiente desvanecido

Una de sus principales limitaciones es el problema del *vanishing gradient*, donde los gradientes se vuelven extremadamente pequeños en redes profundas, dificultando el aprendizaje en capas iniciales. Este problema es más frecuente con funciones de activación sigmoide o tanh (Rumelhart et al., 1986).

b) Dependencia de la diferenciabilidad

El algoritmo solo funciona con funciones de activación y pérdida que sean diferenciables. Esto limita el uso de ciertas funciones que podrían ser útiles, pero no cumplen con esta propiedad matemática.

c) Sensibilidad a la inicialización y a la tasa de aprendizaje

Una incorrecta inicialización de pesos o una tasa de aprendizaje inadecuada puede causar problemas como convergencia lenta, oscilaciones o falta de convergencia. Esto obliga a recurrir a técnicas adicionales como inicialización Xavier/He o ajustes adaptativos del aprendizaje.

d) Riesgo de sobreajuste (overfitting)

Backpropagation puede ajustar demasiado la red a los datos de entrenamiento si no se emplean técnicas de regularización como dropout, early stopping o normalización. Esto reduce la capacidad de generalización del modelo (Goodfellow et al., 2016).

e) Necesidad de grandes cantidades de datos

Para lograr un aprendizaje estable, las redes entrenadas por Backpropagation requieren grandes volúmenes de datos etiquetados, lo cual en algunos casos puede ser costoso o difícil de obtener.

6.3 Consideración Final

A pesar de sus limitaciones, Backpropagation continúa siendo el método estándar y más utilizado para el entrenamiento de redes neuronales. Sus ventajas, especialmente relacionadas con la eficiencia y el poder representativo que otorga a las redes profundas, superan ampliamente sus desventajas en la mayoría de las aplicaciones actuales de Inteligencia Artificial.

REFERENCIAS BÍBLIOGRAFICAS

- **Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986).** *Learning representations by back-propagating errors.* Nature, 323(6088), 533–536.
- **Goodfellow, I., Bengio, Y., & Courville, A. (2016).** *Deep Learning.* MIT Press.
- **Haykin, S. (1999).** *Neural Networks: A Comprehensive Foundation* (2nd ed.). Prentice Hall.
- **Bishop, C. M. (2006).** *Pattern Recognition and Machine Learning.* Springer.
- **Rojas, R. (1996).** *Neural Networks: A Systematic Introduction.* Springer.
- **Nielsen, M. A. (2015).** *Neural Networks and Deep Learning.* Determination Press. (Libro gratuito en línea muy citado).
- **Mitchell, T. M. (1997).** *Machine Learning.* McGraw-Hill.
- **LeCun, Y., Bengio, Y., & Hinton, G. (2015).** *Deep learning.* Nature, 521(7553), 436–