APUNTES INTELIGENCIA ARTIFICIAL

C. Axel

[1] *Tecnológico de Costa Rica, Escuela de Ingeniería en Computación*
Inteligencia Artificial - IC6200 - Grupo 2

E–mail: *axelchaves.r@estudiantec.cr*

1. INTRODUCCIÓN

A continuación se presenta una investigación con el propósito de mostrar una síntesis de los temas vistos en la clase de Inteligencia Artificial del martes 9 de abril de 2024 a cargo del profesor Steven Pacheco Portuguez. Entre estos temas, se destaca: las respuestas al quiz realizado durante la clase, funciones de activación, tangente hiperbólica, ReLU, Backpropagation, entre otros. Además, se pretende reforzar el material de clase con investigaciones externas.

2. RESPUESTAS DEL QUIZ

1. Describa las fórmulas para calcular el Accuracy, Recall, Precision y F1-Score:

Accuracy: Coresponde a la proporción de predicciones correctas respecto al total de predicciones realizadas.

$$Accuracy = \frac{Número\ de predicciones\ correctas}{Total\ de\ predicciones}$$

Recall: Mide la proporción de positivos reales que fueron correctamente identificados por el modelo.

$$Recall = \frac{Verdaderos\ positivos}{Verdaderos\ positivos + Falsos\ Negativos}$$

Precision: Indica la proporción de ejemplos clasificados como positivos que fueron correctamente clasificados.

$$Precision = \frac{Verdaderos\ positivos}{Verdaderos\ positivos + Falsos\ Positivos}$$

F1-Score: Es la media armónica entre Precision y Recall. Es útil cuando hay un desbalance entre las clases.

$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$

2. Justifique por qué no es conveniente evaluar un modelo basándose solamente en el resultado de la métrica Accuracy.

Existen distintos motivos, entre los más destacados se encuentran: puede existir un desbalance de clases en ciertos casos, el costo de los errores de predicción dado un único parámetro de medición, la falta de interpretación del modelo por parte del accuracy, entre otras. Es por esto que es importante utilizar métricas adicionales como Precision, Recall y F1-Score para evaluar de manera más completa el rendimiento de un modelo.

3. Describa en qué consiste la técnica de GridSearch.

Se refiere a una técnica utilizada en el aprendizaje automático donde se busca la mejor combinación de hiperparámetros para un determinado modelo. Asimismo, una práctica común con esta técnica es definir una cuadrícula de posibles valores para cada hiperparámetro y luego evaluar todas estas combinaciones. Finalmente, se seleccionará la combinación de hiperparámetros que produzca el mejor rendimiento.

4. Mencione tres funciones de activación.

Función Sigmoide, Función Tangente Hiperbólica y Función ReLU.

3. FUNCIONES DE ACTIVACIÓN

Se refieren a funciones matemáticas que se aplican a la salida de cada neurona en una red neuronal artificial. Asimismo, estas permiten agregar no linealidad al modelo, lo que le permite a la red aprender y modelar relaciones no lineales en los datos. Por otra parte, las funciones de activación agregan comportamiento a la red, nos permiten controlar el rango de salida de la misma y conceden acceso a la señal para que pase a la siguiente capa [1].

3.1. Función Sigmoide

Es también conocida como función logística y se encarga de mapear los valores de entrada en un rango entre cero y uno.

Esta función es utilizada en las redes neuronales especialmente en la capa de salida de modelos de clasificación binaria, cuya salida se interpreta como la probabilidad de que una instancia sea parte de una clase particular.

Su forma es la siguiente:

$\sigma(x) = \frac{1}{1+e^{-x}}$

Asimismo, es importante destacar que la función Sigmoide es siempre positiva y estrictamente creciente. Además, esta cuenta con el problema del desvanecimiento del gradiente.

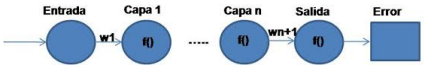

Fig. 1: Representación gráfica del desvanecimiento del gradiente.

3.2. Tangente Hiperbólica

La tangente hiperbólica es especialmente utilizada en las capas ocultas de las redes neuronales. Además, al igual que la funciójn Sigmoide, es diferenciable, lo que la hace adecuada para la propagación hacia atrás (backpropagation).

Su forma es la siguiente:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{1}$$

Además, es importante resaltar que la tangente hiperbólica tiene su activación entre -1 y 1, que puede ser positiva o negativa, está centrada en el origen y es estrictamente creciente. Asimismo, esta es utilizada en modelos de lenguaje LSTM (Long short-term memory) y es diferenciable en cualquier punto.

3.3. Rectifier Linear Unit (ReLU)

La función ReLu se encarga de mapear cualquier valor de entrada negativo a cero y deja cualquier valor positivo sin cambios. Es decir, activa la neurona solo si la entrada es positiva; en caso de que esta sea negativa, la salida es cero.

ReLu se define de la siguiente manera:

$$\text{ReLU}(x) = \max(0, x)$$

(2)

En adición, ReLU está acotada debajo del cero, no está acotada en los positivos y puede tener neuronas muertas.

3.4. Leaky ReLU

Corresponde a una variante de la función ReLU, la cual pretende resolver el problema de las neuronas muertas asociado con ReLU. De este modo, Leaky ReLU en lugar de colocar ceros para los valores de entrada negativos, tiene una pequeña pendiente positiva, lo cual permite que la información fluya incluso si el valor de entrada es negativo, evitando así que la neurona se muera.

Leaky ReLU se define de la siguiente manera:

$$\text{Leaky ReLU}(x) = \begin{cases} x, & \text{si } x > 0 \\ \alpha x, & \text{si } x \leq 0 \end{cases}$$

3.5. Parametric ReLU

Parametric ReLU también corresponde a una variante de la función ReLU que pretende resolver el problema de las neuronas muertas. Es así como la pendiente para los valores de entrada negativos en Parametric ReLU es una constante predefinida. Asimismo, esta pendiente es un parámetro aprendido durante el entrenamiento de la red neuronal.

Parametric ReLu permite aprender un parámetro para dejar que la señal continue. Además, se define de la siguiente manera:

$$\text{PReLU}(x) = \begin{cases} x, & \text{si } x > 0 \\ \alpha x, & \text{si } x \leq 0 \end{cases}$$

3.6. Softmax

La función Softmax se utiliza en la capa de salida de una red neuronal, especialmente en problemas de clasificación multiclase.

Softmax toma como entrada un vector de números reales y produce otro vector de la misma dimensión, donde cada elemento está en el rango (0,1) y la suma de todos los elementos es igual a 1. Es decir, se convierten los números reales en probabilidades, lo que hace más sencilla la interpretación de la salida de la red como una distribución de probabilidad sobre las clases.

Softmax se define de la siguiente manera:

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$$

3.7. ¿Cuál Función de Activación Utilizar?

Escoger una función de activación para una red neuronal puede conllevar a decisiones complejas, sin embargo, una etapa de pruebas es clave para la selección. No obstante, existen ciertas recomendaciones generales.

Por ejemplo, si se quiere resolver un problema relacionado al desvanecimiento del gradiente, es bueno utilizar una función como Sigmoid. Por otro lado, la función ReLU es sencilla de computar y se utiliza mucho para Deep Learning, no obstante, hay que tener especial cuidado con el problema de las neuronas muertas y en utilizar ReLU únicamente en las capas ocultas y no en las capas de salida.

Como estos, existen muchos casos de aplicación de las funciones de activación, lo cual también debe estar ligado al criterio de quien desarrolla y de los resultados que se obtienen tras las pruebas.

4. ENTROPÍA CRUZADA (CROSS ENTROPY)

La entropía cruzada es una medida utilizada el problemas de clasificación para cuantificar la diferencia entre dos distribuciones de probabilidad. Por tanto, la entropía cruzada se utiliza como una función de pérdida para entrenar modelos de clasificación [2].

La fórmula de la entropía cruzada es la siguiente:

$$\text{CrossEntropy}(p, q) = -\sum_{i} p(i) \log(q(i))$$

La entropía cruzada penaliza las discrepancias entre las probabilidades predichas y las probabilidades reales. Cuanto mayor sea la diferencia entre las distribuciones de probabilidad, mayor será el valor de la entropía cruzada.

5. BACKPROPAGATION

Backpropagation corresponde a un algoritmo fundamental utilizado para entrenar redes neuronales con el fin de ajustar los pesos de las conexiones entre neuronas [3].

El proceso de backpropagation se lleva a cabo en dos fases principales: la propagación hacia adelante (forward propagation) y la propagación hacia atrás (backward propagation).

5.1. Forward Propagation

Durante la forward propagation, las entradas se alimentan a la red neuronal, y las activaciones se calculan en cada capa de la red utilizando los pesos actuales. Las activaciones se propagan capa por capa hasta que se obtienen las salidas finales de la red.

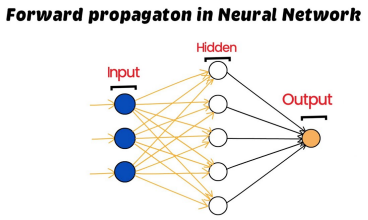

Fig. 2: Representación gráfica del la propagación hacia adelante.

5.2. Backward Propagation

Durante la propagación hacia atrás, el algoritmo calcula los gradientes de la función de pérdida con respecto a los pesos de la red utilizando la regla de la cadena de cálculo. El proceso de cálculo de gradientes y actualización de pesos se repite iterativamente durante el entrenamiento de la red neuronal hasta que la función de pérdida converja a un valor mínimo o hasta que se alcance un número máximo de iteraciones.

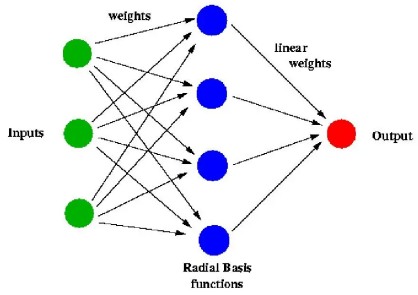

Fig. 3: Representación gráfica del la propagación hacia atrás.

6. OPTIMIZACIÓN DEL GRAFO

Se refiere a optimización de operaciones en un grafo computacional, el cual es una representación abstracta de las operaciones matemáticas realizadas en un modelo de aprendizaje automático. En este contexto, los modelos se pueden representar como grafos computacionales, donde los nodos del grafo representan operaciones y los bordes representan los datos que fluyen entre estas operaciones [4].

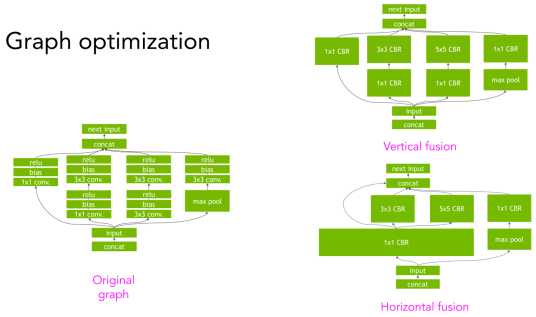

Fig. 4: Descripción del proceso de optimización de un grafo.

Para llevar a cabo la optimización de un grafo, se pueden seguir estos pasos:

1. Representación del modelo como un grafo: el modelo se representa como un grafo computacional, donde los nodos del grafo representan operaciones y los bordes los datos que fluyen entre estas operaciones.
2. Análisis del grafo: Se realiza un análisis detallado del grafo con el fin de identificar posibles ineficiencias y oportunidades de optimización. En esta parte es posible utilizar una función de transferencia y encontrar derivadas parciales con el uso de la regla de la cadena.

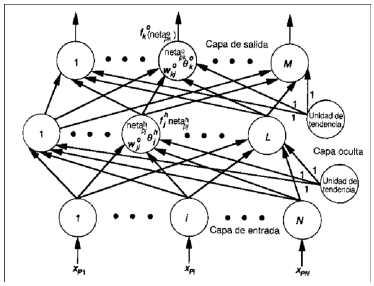

Fig. 5: Ejemplo de grafo con posibilidad de optimización.

3. Aplicar técnicas de optimización: Al pasar por las entradas del modelo, revisar los pesos de cada nodo, aplicar la función de transferencia y así obtener una función de activación, se puede generar una salida, la cual corresponde a la optimización que se estaba buscando.

REFERENCIAS

[1] P. Saucedo de Miguel, Funciones de activación ruidosas en redes neuronales recurrentes.

[2] Z.I. Botev, D.P. Kroese, R.Y. Rubinstein and P. L'Ecuyer, '"The cross-entropy method for optimization,", vol. 31, pp. 35-59.

[3] B.J. Wythoff, '"Backpropagation neural networks: a tutorial," Chemometrics Intellig.Lab.Syst., vol. 18, no. 2, pp. 115-155.

[4] A. Cedeño, '"APLICACIÓN DE LA TEORÍA DE GRAFOS A LA OPTIMIZACIÓN DE REDES NEURONALES ARTIFICIALES," Revista SOCIENCYTEC, vol. 1, no. 2, pp. 51-69.