Seminario
Cali - Septiembre 13 de 2019

Classification Using Naive Bayes (Lantz)

Kevin García[1,a], Alejandro Vargas[1,b]

[1]Departamento de estadística, Universidad del Valle, Cali, Colombia

Resumen

Las redes neuronales en la actualidad son los algoritmos por excelencia del Machine Learning, arrojando muy buenos resultados. En el siguiente informe se verán presentados definiciones y conceptos sobre Neural Networks y Deep Learning que tratarán de responder a las preguntas ¿qué son? y ¿cómo funcionan?, también se verán las funciones de activación, el algoritmo Backpropagation y el descenso del gradiente, conceptos claves para comenzar a entender la anatomía de una red neuronal y su funcionamiento.

1. Introducción

El ser humano en su búsqueda por entender la realidad que lo rodea, se ha empeñado en desarrollar modelos que logran interpretar en planos mas "sencillos" dicha realidad, de forma que sea entendible para todos. Las redes neuronales son el conjunto de algoritmos de Machine Learning más populares; el reconocimiento de caracteres, de imágenes, de voz, generación de texto, traducción de idiomas, predicción de fraude, conducción autónoma y pronóstico de enfermedades son algunos de los muchos ejemplos que existen, en los cuales se hace uso de estos potentes algoritmos con los que podemos modelar comportamientos "inteligentes".

2. ¿Qué son las redes neuronales?

Para entender de manera sencilla este concepto, comenzaremos dando respuesta a la pregunta: ¿qué es una neurona? Una neurona es la unidad básica de procesamiento dentro de una red neuronal, le hace honor a su nombre, ya que al igual que una neurona biológica, esta tiene unas conexiones de entrada por las cuales recibe "estímulos externos", que conocemos como valores de entrada, y con base en estos valores, la neurona realizará un cálculo y arrojará un valor de salida, así pues, una neurona no deja de ser más que una suma ponderada de los valores de entrada $Y = W_1X_1 + W_2X_2 + W_3X_3 + b$, esto lo conocemos como regresión lineal.

Una regresión separa con una linea recta grupos de puntos pero, ¿qué sucede cuando no es posible separar una nube de puntos con una linea?, bueno, este problema se resuelve agregando más neuronas; de aquí surge el concepto de red neuronal, estas pueden ser cada vez más grandes y abstractas, entre más capas y neuronas sean agregadas más complejo y abstracto es el aprendizaje de la red.

[a]Universidad del Valle. E-mail: kevin.chica@correounivalle.edu.co
[b]Universidad del Valle. E-mail: jose.alejandro.vargas@correounivalle.edu.co

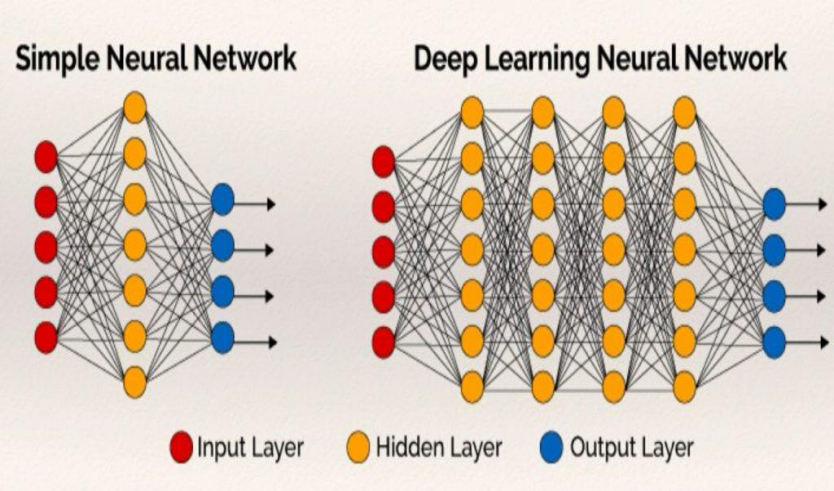

FIGURE 1: Gráfico de Redes Neuronales

Como podemos observar en la imagen anterior, una red neuronal está compuesta por neuronas o nodos de procesamiento, y estos a su vez están divididos en grupos que se llaman "capas". Existen 3 tipos de capas: capas de entrada, capas ocultas y capas de salida; las conexiones se establecen entre los nodos de cada capa adyacente, la capa de entrada está formada por nodos de entrada, que reciben la información directamente del exterior; mientras que la capa de salida representa la respuesta de la red a una entrada dada, siendo esta información transferida al exterior. Una red neuronal también la podemos ver de la siguiente forma:

$$a_l = g(\omega_{\ell 0}^{(1)} + \sum_{j=1}^{p} \omega_{\ell j}^{(1)} X_j)$$

Donde, a_l es la conexión a la capa de entrada a través de un vector de parámetros o pesos $\omega_{\ell j}^{(1)}$ (El (1) hace referencia al número de la capa, en este caso sería la primera, y ℓj hace referencia a la $j - esima$ variable y la $\ell - esima$ unidad). Los términos $\omega_{\ell 0}^{(1)}$ son los sesgos o interceptos, y la función g es llamada función de activación, usualmente se utilizan funciones sigmoides en las capas de salidas, ya que estas arrojan valores entre 0 y 1 ($g = \frac{1}{(1-e^{-t})}$), en el resto de capas se suelen utilizar funciones ReLU (Rectified Linear Unit Function).

3. ¿Cómo funcionan?

Como ya observamos, al poner dos neuronas de forma secuencial, una de ellas recibe la información procesada por la neurona anterior, esto trae una ventaja muy importante para el aprendizaje de la red y es que la red podrá aprender conocimiento jerarquizado, lo que sería en palabras simples, que mis neuronas en las primeras capas sean capaces de procesar cosas simples y mis neuronas en capas posteriores hagan procesos más complejos con la información suministrada por las primeras capas. Con un ejemplo, esto sería: Un fin de semana perfecto es estar con amigos y tomar cerveza, no es posible que sea perfecto si una de estas condiciones no se cumple; con una regresión simple

podría modelar si mi fin de semana va a ser perfecto o no, en función de si tengo amigos y cerveza, pero, y si lo que quiero es conocer ¿cuál será mi calificación en el examen final del lunes? Podríamos pensar que la respuesta a esto está relacionada con el interés por la asignatura, pero también estaría relacionada con si mi fin de semana fue perfecto o no. Las neuronas en mi primera capa se encargarían de procesar si mi fin de semana fue perfecto o no y si mi interés por la clase es positivo o negativo; se pensaría entonces que si mi fin de semana fue perfecto y mi interés por la clase es negativo entonces mi calificación final será baja. De esta manera jerarquizada, mi red neuronal puede aprender a distinguir qué personas van a aprobar su examen final el lunes. Bueno, entre más capas tengo, más complejo es el aprendizaje de mi red, y esta profundidad en la cantidad de capas es la que da nombre al Deep Learning.

Para alcanzar un aprendizaje profundo con mi red se requieren conectar múltiples neuronas de forma secuencial, pero como cada neurona a final de cuentas resuelve un único problema de regresión lineal, y lo que termino haciendo es concatenar múltiples regresiones lineales; matemáticamente se puede comprobar que el resultado de estas conexiones o mejor dicho, la suma de mucha lineas rectas es al final una única linea recta, por lo que mi red neuronal terminaría convirtiéndose en una regresión lineal. Para que esto no suceda, es necesario conseguir que los resultados arrojados por cada neurona sean distintos a una linea recta, esto se logra con las funciones de activación.

Función de activación: Una función de activación resuelve el problema, de forma que mis valores de salida pasan primero por dicha función, la cual le añade deformaciones no lineales y hace que mi red no colapse en una única regresión lineal. Estas deformaciones dependen de la función de activación utilizada, existen varios tipos, pero las más usadas son:

1. **Función sigmoide:** La función sigmoide, denotada de la forma $G(x) = \frac{1}{(1-e^{-x})}$ hace que los valores muy altos se saturen en 1 y los valores bajos en 0, por lo tanto, esta función añade dicha deformación no lineal, y además representa muy bien probabilidades.

2. **Función Tanh:** La función tangente hiperbólica, cuya forma es similar a la sigmoide, pero su rango varía entre -1 y 1 y está dada por $G(x) = tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$

3. **Función ReLU:** Esta función se comporta de forma lineal cuando es positiva, y constante a 0 cuando el valor de entrada es negativo, se puede denotar como $G(x) = max(0, x)$

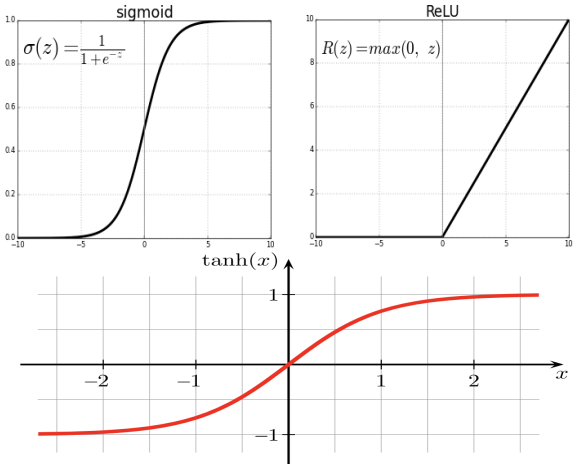

FIGURE 2: Gráficos de las funciones de activación

Como observamos, cada función tiene su peculiaridad, pero todos resuelven el problema añadiendo estas deformaciones no lineales. Cada función es utilizada en situaciones particulares donde se les requiera, así, podemos lograr encadenar varias neuronas. Un ejemplo, podría ser el de separar la siguiente nube de puntos de forma circular, esto no es posible lograrlo con una linea recta, ni siquiera con dos, es necesario implementar múltiples neuronas con múltiples funciones de activación para conseguirlo.

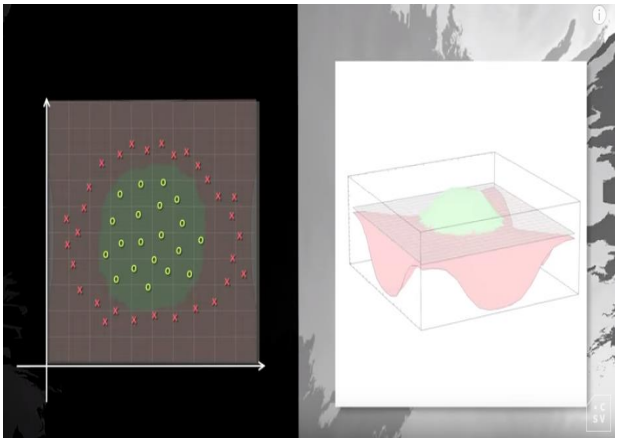

FIGURE 3: Ejemplo funciones de activación

En la figura anterior, podemos apreciar cómo utilizando redes neuronales es posible separar la nube de puntos en dos grupos distintos, en la práctica, este problema podría ser el de identificar células cancerígenas en una imagen, se utilizaron 4 neuronas en la capa oculta, una entrada y una salida; al implementar funciones sigmoides en la capa oculta para cada una de las neuronas, obtenemos la forma geométrica que separa ambos grupos, un plano con un bulto en el medio. De esta forma, las redes neuronales son capaces de desarrollar soluciones muy complejas gracias a la unión de todas estas neuronas.

Como se dijo anteriormente, las redes neuronales son un conjunto de algoritmos de machine learning, así pues, lo que nos interesa es que todas estas cosas que puede realizar una red neuronal, las debe aprender a hacer por si misma, en otras palabras, la red neuronal debe ajustar sus parámetros por si misma a partir de los datos que se le suministren, a esto se le denomina "aprendizaje automático".

El cómo entrenar una red neuronal para que aprenda a realizar ciertas tareas, es un problema que se trató desde el año 1986, pero no fue hasta varios años después, con el avance tecnológico, que se dio solución a este problema. El trabajo firmado por Rumelhart, Hinton y Williams muestra experimentalmente como usando un "nuevo" algoritmo de aprendizaje se podría conseguir que una red neuronal auto ajustara sus parámetros, para así aprender una representación interna de la información que se está procesado, este trabajo lleva el nombre de Learning representations by back - propagatin errors; de aquí el nombre del algoritmo utilizado en el aprendizaje de cualquier red neuronal: Backpropagation.

Los modelos de regresión lineal ajustan sus parámetros basados en el método de MC (mínimos cuadrados), para con este encontrar el punto mínimo de la función de coste, y así nuestra recta de regresión será tal que minimice los errores, lo que sucede es que los MC están limitados al modelo de regresión y a su función de coste, esto es porque el método supone que la función de coste es convexa, por lo que de encontrar un mínimo, estaremos totalmente seguros de que este es un mínimo global,

el problema surge cuando tenemos funciones no convexas, este es el caso de las redes neuronales; en funciones no convexas es posible tener mínimos locales, por lo que al derivar la función e igualar a 0 vamos a obtener múltiples ecuaciones a resolver y no solo esto, si no además podemos tener otras zonas de la función que poseen pendiente nula, estas pueden ser máximos locales, puntos de inflexión o puntos silla. Un sistema de ecuaciones enorme que es ineficiente de resolver. Para darle solución al problema de encontrar los mínimos de mi función de coste, se hará uso de otro algoritmo importante en el aprendizaje de una red neuronal, el descenso del gradiente.

Gradient Descent: Para entender de forma intuitiva este algoritmo, imaginemos que estamos en la punta de una montaña y queremos descender al punto más bajo, sin embargo, vamos con los ojos vendados, lo mas lógico es ir tanteando la inclinación del terreno y desplazarnos por donde la pendiente descienda con mayor intensidad, los pasos a seguir serían evaluar la inclinación para conocer la dirección con la mayor pendiente, luego caminamos una distancia en dicha dirección y nos detenemos, volvemos a repetir hasta llegar a un lugar plano.

Matemáticamente, esto sería calcular las derivadas de la función de coste para cada parámetro en un punto aleatorio, esta nos dirá la pendiente en ese punto para cada parámetro, el vector de estas derivadas se denomina vector gradiente $\triangledown W$, y nos indica la dirección hacia la que la pendiente asciende, pero como lo que queremos es descender, es lógico trabajar con $-\triangledown W$ la dirección contraria, nos desplazamos en esa dirección y repetimos el proceso, calculamos la derivadas parciales para el nuevo punto y volvemos a movernos iterativamente hasta llegar a un punto donde movernos no suponga una variación notable en el coste, es decir que la pendiente sea próxima a nula. El algoritmo del descenso del gradiente lo podemos denotar como $\theta := \theta - \alpha \nabla W$, donde θ es mi parámetro, el cual se va a actualizar, y α es el ratio de aprendizaje y define cuánto afecta el gradiente a la actualización de nuestros parámetros en cada iteración, así pues, un ratio muy bajo hará que se tengan que hacer mas iteraciones para converger y esto repercute en un coste de tiempo mayor, por otra parte, un ratio muy alto tendrá el problema de que se harán saltos muy largos y nuestra función de coste nunca convergerá a un mínimo quedando en un bucle infinito, así que la elección de este ratio es fundamental.

Backpropagation: El descenso del gradiente me da la solución al problema de encontrar los mínimos por medio de mi vector gradiente; en una regresión lineal simplemente era responder a la pregunta de ¿Cómo varía el coste frente a un cambio del parámetro W?, así, solo era calcular las derivadas parciales $\frac{\delta C}{\delta \omega}$. Para una red neuronal no es tan sencillo, ya que tenemos múltiples parámetros y la forma en cómo varía un parámetro puede afectar el resultado final, y por lo tanto, afectar el error de la red, es decir, si pongo a variar un parámetro en las primeras capas, no me dirá a ciencia cierta si el error disminuyo debido a ese cambio o a alguno en las capas posteriores. El algoritmo de Backpropagation nos dirá el valor de $\frac{\delta C}{\delta \omega}$, es decir nos ayudará a responder la pregunta de ¿cómo varia el coste cuando variamos alguno de los parámetros?

Para entender de manera intuitiva el algoritmo de Backpropagation, imaginemos una cadena de responsabilidades, un caficultor se encarga de cosechar granos de café según un proceso que él mismo ha elegido, estos granos son enviados a una de muchas fabricas las cuales tuestan, muelen y envasan, luego, estos paquetes viajarán a uno de los muchos puntos de venta, donde nosotros nos decidiremos a comprar específicamente esa marca, llegaremos a la casa y nos prepararemos una tasa de café que nos dará energía para afrontar el examen final; terminado el día nos damos cuenta de que perdimos el examen, ¿es acaso esto culpa de la tasa de café?, tal vez no era la marca adecuada o tal vez fue culpa de la empresa que no molió bien los granos, o quizás el caficultor utilizó un proceso inadecuado, ¿de quién es la culpa?. Esta cadena de responsabilidades se puede asemejar a una red neuronal, donde cada nodo es una neurona con una tarea determinada, el examen es el resultado de salida o Output layer y la evaluación que obtenemos la podemos ver como nuestra función de coste, así, ese resultado desfavorable en el examen genera una fuerte señal de error, lo que se hace

es analizar toda la cadena de responsabilidades que ha afectado el resultado y si encontramos una neurona que tuviera influencia en el error obtenido, entonces debemos responsabilizarla con parte de ese error, este análisis debe realizarse hacia atrás, desde la señal de error hacia las primeras capas, y esto es así, ya que en una red neuronal el error de las capas anteriores depende del error de las capas posteriores, por ejemplo, si nos damos cuenta que el resultado del examen no depende mucho de si la calidad del café es buena o mala, entonces el error en las fases anteriores, como el método de molido o el método de cosechado también afectan poco al error final.

Con el algoritmo de Backpropagation obtenemos el error que tiene cada neurona en el error final, y con esto, podemos utilizar el descenso del gradiente para minimizar el error en cada una de las neuronas como si se tratara de una única regresión.

Estos algoritmos se pueden escribir de forma que:

$$\nabla W^k = \frac{1}{n}\sum_{i=1}^{n}\frac{\delta L[y_i, f(x_i; W)]}{\delta W^k}$$

$$W^k := W^k - \alpha \nabla W^k$$

4. Conclusiones

Podemos decir que aunque su funcionamiento es complejo y en algunos casos abstracto, las redes neuronales son útiles para resolver problemas en muchas áreas del conocimiento y la tecnología, automatizando procesos, logrando replicar comportamientos inteligentes y dando cabida a un mundo de posibilidades.

1. Las funciones de activación resuelven el problema de la linealidad dentro de las redes neuronales de manera optima, dando soluciones a problemas complejos que no son fáciles de modelar con regresiones lineales.
2. El algoritmo del descenso del gradiente y backpropagation cumplen una función importante en la anatomía de una red neuronal, ya que son el centro del aprendizaje de la misma.
3. Las redes neuronales son capaces de modelar comportamientos tan abstractos, que muchas de estas son completas cajas negras, y logran encontrar relaciones o asociaciones entre variables las cuales un humano no imaginaría.

References

Efron, B. & Hastie, T. (2017), *Computer age statistical inference. Algorithms, Evidence, and Data Science*, Cambridge.

Lantz, B. (2013), *Machine learning with R*, Packt Publishing.

Vega, C. S. (n.d.), 'Aprendiendo inteligencia artificial'.
*https://www.youtube.com/playlist?list=PL-Ogd76BhmcDxef4liOGXGXLL-4h65bs4