05 - Gradient Descent – REDES NEURONALES - (Parte 5) - Transcrip

Hasta este momento hemos definido una función la cual llamamos función de costo j que nos permite evaluar qué tan bueno es nuestro modelo para un bloque de imágenes o muestras.

Y este indicador de qué tan bueno es el modelo lo que realmente representa es que tan adecuados son los parámetros W y B que tenemos actualmente.

Por lo tanto en este vídeo vamos a continuar con el proceso de entrenamiento de la red neuronal y lo que vamos a ver es cuál es el algoritmo que nos va a permitir ajustar dichos parámetros W y B para que dicha función de costo vaya minimizándose.

Este algoritmo es conocido como descenso por gradientes o gradient descent y su fundamento principal es el cálculo diferencial.

Empezamos.

Hasta el momento hemos estudiado una red neuronal artificial muy sencilla de una sola capa donde esta capa también es la capa de salida la cual está integrada de tres neuronas artificiales.

Cada una de estas neuronas tiene asociada una serie de pesos sinápticos las cuales afectan cada una de las entradas que que llegan a la neurona o a la red neuronal.

En este ejemplo muy sencillo nuestras imágenes de entrada son imágenes sintéticas de un gato de un perro y de un ave y para fines de simplicidad las representamos como si solamente tuvieran cuatro pixeles y cada uno de estos pixeles son forma las entradas a nuestra red neuronal.

Es importante mencionar que cada una de estas unidades o neuronas artificiales también es conocida como perceptron.

Así entonces tenemos que los pesos sinápticos asociados con nuestra red neuronal pueden ser representados en una matriz llamada W y las imágenes de entrada pueden ser representadas en una matriz donde cada columna de la matriz es decir cada columna representa los pixeles de cada una de las imágenes.

De esta forma al realizar la operación de Wx más elvallas obtenemos un score que representa la viabilidad de cada una de las imágenes.

De igual forma estudiamos que si hace resultado a ese score le aplicamos la función softmax la cual está representada por esta fórmula aquí en la pantalla en la parte de arriba obtenemos una salida que representa probabilidades de cada una de las imágenes de entrada para pertenecer a una clase en particular.

En este caso recordemos que las clases pueden ser gato perro y ave entonces al aplicar la función softmax obtenemos esta matriz de resultados donde nos indican las probabilidades de cada una de estas imágenes.

Si a los resultados de cada imagen de entrada por ejemplo esta columna nos representa las probabilidades para una imagen de entrada de tipo gato si a este vector le aplicamos la función de pérdida mostrada aquí tenemos que la pérdida para este ejemplo es de 0.94 la pérdida para el perro es de 1.77 y la pérdida para el ave es de 0.89.

Es muy importante notar que estas pérdidas son específicas para cada una de las imágenes que estamos utilizando en el proceso de entrenamiento por lo tanto la pérdida estrictamente lo que representa es que también funciona nuestro modelo de red neuronal para clasificar una imagen determinada mencionábamos que en el caso del gato vemos que la probabilidad de que esta imagen sea de un gato de esta imagen de un gato sea reconocida por nuestra red neuronal como un gato es de 0.39 y nos da la probabilidad más alta entre las probabilidades de esta imagen ya que muestra que 0.28 para que sea perro y 0.33 para que sea un ave no obstante en el caso del perro de esta imagen del perro la probabilidad de que esta imagen de perro sea reconocida por nuestra red neuronal como un perro es simplemente solamente de 0.17 es decir la probabilidad es muy baja por lo tanto es lógico que la pérdida asociada a esta imagen sea mayor sea de 1.77 mayor que la del gato entonces está reconociendo bien el gato pero no está reconociendo bien el perro y si agrupamos todas estas pérdidas de todas las imágenes que están siendo procesadas por nuestra red neuronal entonces tenemos nuestra función de costo dado por j que es 1.2 por lo tanto la función de costo mire que también funciona nuestro modelo de red neuronal para todas las imágenes que están siendo procesadas en este caso solamente estamos simulando 3 imágenes pero en un proceso de entrenamiento real pueden ser muchas más imágenes de hecho van a ser muchas más imágenes estamos hablando de que podríamos tener miles de imágenes para entrenar nuestra red neuronal y así los únicos parámetros sobre los cuales tenemos control son precisamente los parámetros de nuestra red neuronal que son los pesos sinápticos w y los vayases no podemos cambiar las entradas ya que estas son las imágenes que queremos aprender a clasificar y entonces lo que tenemos que hacer es encontrar una forma de cómo ajustar estos parámetros w y b para reducir esta función de costo para hacerla lo más pequeña posible por cierto es importante mencionar que estrictamente dado que la función de costo depende de w y w no es un solo valor sino es una matriz de valores de pesos sinápticos asociados a nuestra red neuronal la función w debe ser una función de w mayúscula y el vallas también es un vector de valores uno por cada neurona que tenemos por lo tanto debemos decir que este es un vector y w es una matriz nuestro siguiente paso es encontrar una forma en la que podamos hacer de manera sistemática el ajuste de los valores w y b para reducir nuestra función de costo j y así tenemos que utilizar utilizar una herramienta matemática que nos permita encontrar qué tanto varía la función j con respecto a w y b este proceso es un proceso de optimización y una forma de visualizar la función de costo es imaginarla como si tuviéramos un paisaje de montañas y estamos en un lugar de en algún lugar en una cima de una montaña y lo que queremos hacer es llegar al mínimo llegar a este valle donde la altitud a la cual nos encontramos es la función de costo es que tan grande es nuestra función de costo y lo que queremos es ir dando pasos en el sentido de mayor inclinación de esta montaña para llegar lo más rápido posible a este mínimo por lo tanto vamos ir dando pasitos en un sentido en el sentido que la mayor de la mayor razón de cambio de de la montaña es decir el gradiente hasta llegar a este mínimo volviendo nuestros ejemplos de forma gráfica una forma muy común de representar nuestro proceso de minimización es asumir que nuestra función de costo vamos a llamar la j que solamente depende de dos dimensiones podemos decir que depende de w de una sola w y de un vallas y tiene esta forma como de un tazón y nos encontramos en alguna parte del tazón acá y lo que queremos nuevamente es ir encontrando la forma de llegar lo más rápido que podamos al mínimo del tazón que está dado por este punto de color azul si estamos viendo por ejemplo este mismo tazón desde arriba veríamos que tal vez estamos empezando por acá en este punto y lo que queremos hacer es llegar al mínimo que está aquí en el centro del tazón entonces vamos a ir calculando mediante el gradiente de la función de costo con respecto a w y a b para poder tomar un paso en la dirección de mayor razón de cambio que nos aproxima lo más rápido posible a este valor mínimo por ejemplo podemos tomar un paso de aquí en este sentido luego llegamos a este punto de nueva cuenta y evaluamos el gradiente y encontramos la mayor el punto de mayor razón de cambio y tomamos otro paso por acá y así mismo llegamos a este punto repetimos el proceso tomamos este punto este paso así nos vamos hasta que lleguemos a este valor que minimiza a este proceso dado que estamos tomando un paso en el sentido del gradiente se le llama descenso por gradientes o en inglés gradient descent y es el proceso de optimización más utilizado para realizar el proceso de entrenamiento de una red normal que nos va a permitir minimizar el valor de la función de costo y al hacer esto vamos a ajustar nuestros parámetros para que nuestra red normal tenga un mejor desempeño como menciona anteriormente el gradiente indica la dirección de mayor cambio por lo tanto esto es importante cuando actualicemos nuestros parámetros tenemos que considerar el negativo del gradiente ya que queremos minimizar la función de costo si quisiéramos maximizar una función entonces tomaríamos el gradiente como positivo pero como queremos minimizarla tomamos el negativo de gradiente en la siguiente imagen se va a ver más claro esto entonces de nueva cuenta si ahora simplificamos aún más nuestra función de costo y reducimos las dimensiones de este espacio que dependía de dos variables j y b a un espacio que son donde solamente depende de una variable w y nuestra función j entonces depende de esa variable lo que tendríamos es que imaginemos que este es el valor actual de w este punto que tengo aquí que estoy marcando lo que vamos a hacer entonces es evaluar el gradiente en este punto que es la pendiente y si nos acordamos de cálculo la pendiente de ese punto estaría dada por la derivada la función de costo con respecto a en este caso sería de w con respecto a w y lo que esto nos representa es esta pendiente por lo tanto lo que queremos hacer para minimizar nuestra función de costo es que estamos acá encontramos la pendiente en ese punto y lo que vamos a hacer es tomar es tomar un paso en esa dirección y llegamos a este punto que nos representa un nuevo valor w originalmente nuestro valor w era este valor de aquí y ahora una vez que llegamos a este punto de nueva cuenta calculamos el gradiente tomamos la que sería la pendiente y volvemos a ajustar nuestro valor de w yendo hacia este punto de nueva cuenta que ajustamos y nos vamos a otro valor de w los valores de w van ajustándose aquí sería otro valor de w hasta que lleguemos a este valor mínimo de forma muy intuitiva y de forma muy a grandes rasgos este es el proceso de actualización de los valores de nuestros parámetros de la red neuronal para minimizar el gradiente por lo tanto lo que queremos hacer es la operación donde nuestro parámetro w se actualiza de la forma w actual menos la derivada de la función de costo con respecto a w es decir este este gradiente y ahora dependiendo de qué tan rápido queramos llegar podemos cambiar este paso que tan grande es el paso que tomamos hacia hacia nuestra siguiente valor w por ejemplo si estamos en este punto tal vez nos convenga tomar un paso de este tamaño en el sentido del gradiente y llegamos más rápido o podríamos tomar un paso más pequeñito que nos lleve nada más acá y lleguemos a este valor de w y así irnos con pasos muy pequeñitos que nos vayan llegando poco a poco a nuestro valor de w por lo tanto este valor del gradiente se multiplica por un hiperparámetro de hecho alfa que indica qué tan grande es el paso por lo tanto por lo tal se le conoce como este size también se le conoce como la razón de aprendizaje o learning rate vamos a ver después cuando estamos haciendo la implementación que pytorch por ejemplo maneja el concepto de la ley y se refiere a esto a este tamaño del paso es un punto que tenemos con el cual tenemos que tener cuidado porque dependiendo del punto en el que nos encontremos si este valor de la ley es muy grande corremos el riesgo de hacer un brinco que tal vez nos lleve acá a este otro extremo y luego si es muy grande tomar otro paso por acá de tal forma que no se vaya acercando al valor mínimo de nuestra función de costo de forma contraria si para el valor de la ley tomamos un número muy pequeñito puede ser que la forma en la que vaya aproximándose hacia este valor mínimo sea con pasos muy pequeños y tome mucho tiempo en llegar entonces una parte de crítica de nuestro proceso de entrenamiento de la red neuronal es elegir qué tan grande debe ser este paso que vamos a tomar para llegar a nuestro mínimo y de esta forma este es el proceso que vamos a seguir para actualizar los valores de nuestra red neuronal por lo tanto lo que vamos a hacer es un proceso iterativo donde en cada paso realicemos una de estas operaciones y así es como vamos a ajustar los valores de nuestra matriz w y los valores de b para reducir las pérdidas o la función de costo también tenemos que realizar esta operación para b b menos ese mismo learning rate la razón de aprendizaje por la derivada de nuestra función de costo con respecto a b por lo que vamos a estar calculando comúnmente la derivada de la función de costo con respecto a estos parámetros ahora algo que es importante tomar en cuenta es que en este modelo que solamente tenemos una neurona es relativamente trivial calcular la derivada de la función si aquí tuviéramos nuestra función de costo calcular la derivada de esta función con respecto a w y a b sería relativamente sencillo sin embargo nuestros modelos van a ser como este o aún más grandes con muchas capas donde si lo que quisiéramos es calcular la derivada vamos a decir que por aquí tenemos la pérdida la función de pérdida si quisiéramos calcular la derivada de la pérdida con respecto a todos los parámetros w y notemos que aquí estoy utilizando la anotación con esta con esta letra delta en vez de una d porque estamos indicando que ya estamos hablando de cálculo con muchas variables ya la pérdida la función de pérdida o de costo depende de absolutamente todos estos parámetros w recordemos que cada conexión aquí en la red normal cada una de estas conexiones representa una w un peso sináptico de cada una de las neuronas por lo tanto aquí tenemos muchísimos pesos sinápticos y la pérdida o el costo vamos a llamarlo costo y vamos a decir es que es j y la el costo puede ser afectado por cualquiera de esos valores w por lo tanto lo que tenemos que hacer es cada una de estas neuronas la vamos a simplificar aún más pensando la como primero en una operación de multiplicación y su salida va a una función de suma en vez de hacer la operación directamente de w por x + b y hacer tomar toda esta derivada la forma en la que vamos a implementar estos problemas y la forma en la que los frameworks de deep learning funcionan es construyendo una gráfica computacional donde cada operación está representada por un nodo y vamos a simplificar aún más vamos a suponer que solamente tenemos una entrada una entrada x la cual está asociada con un peso sináptico w y además tenemos un vallas vamos a suponer que es una neurona completa y aquí tenemos un vallas esto lo vamos a representar de forma que tenemos nuestra entrada x w ahora también lo vamos a ver como una entrada estas dos valores van a ser multiplicados y luego este resultado va a pasar por un operador de suma donde se le va a sumar el valor b esta es la forma en la que usualmente se representa se realiza el proceso de operaciones mediante una implementación de gráficas computacionales en este vídeo hemos presentado el algoritmo de gradient descent o de senso por gradientes que nos permite ajustar los parámetros w y b de nuestra red neuronal para así ir minimizando nuestra función de costo en el siguiente vídeo vamos a ilustrar de forma intuitiva con un ejemplo muy sencillo como efectivamente al cambiar nuestros parámetros w o b por un valor pequeño h la función el valor de la función a la salida es afectado de una manera que depende de la derivada de dicha salida con respecto a dichos parámetros esto nos va a permitir visualizar un poco de forma más clara porque el algoritmo de gradient descent basado en cálculo funciona para ir minimizando la función de costo con respecto a los parámetros entonces vamos a ¡Seguid!

[SILENCIO] [MÚSICA]