04 - Función de Costo - Cross Entropy - REDES NEURONALES - (Parte 4) Transcript

En este vídeo continuamos con nuestro ejemplo de Redner, una ala artificial de una sola capa.

En el vídeo anterior presentamos la función softmax que nos permite convertir las salidas provenientes de una transformación lineal a probabilidades que nos indican la viabilidad de que una determinada imagen de entrada pertenezca a una clase.

En este vídeo vamos a presentar una función de pérdida que nos permite evaluar qué tan buenos son los parámetros actuales que nuestra redner una al tiene y con base en este indicador que nos da el número de la función de pérdida vamos a poder realizar ajustes a nuestros parámetros tratando de minimizar dicha función de pérdida.

Empezamos.

Antes de mostrar la función loss function y cómo vamos a calcular este error podemos procesar varias imágenes de entrada a la vez en una especie de batch o mini lote como vimos anteriormente y las salidas que nos van a dar van a ser los puntajes para cada una de las clases como ya habíamos visto en una diapositiva anterior y podemos aplicar directamente computacionalmente la función exponencial a todos estos elementos de la matriz de salida donde nos dan estos resultados y luego lo único que hacemos es dividir entre la suma de cada una de estas columnas entre la vamos a llamarle suma y así tenemos la probabilidad para cada una de las imágenes de entrada donde por ejemplo vemos que la probabilidad de que sea gato se trata de 0.39 la probabilidad de que sea perro 0.28 y la probabilidad de que sea ave es 0.33 para la primera imagen la primera esta columna está indicando las probabilidades para esta imagen de entrada la segunda columna es la que acabamos de ver para los perros para el caso de la imagen de entrada perro y la tercera columna para esta imagen de entrada es importante notar que estos resultados son para estas imágenes en particular vemos que en general al menos para este gato en particular funciona más o menos bien o sea funciona bien la probabilidad más alta es para el gato pero notamos que la probabilidad del ave también está muy cercana para el perro para este perro en particular no funciona nada bien ya que la probabilidad más baja es la probabilidad arrojada para el perro para la clase perro y en el caso del ave pues funciona también bien porque la probabilidad más alta es la arrojada para para la clase ave pero estamos hablando de que solamente para estas imágenes y lo que queremos es tener un conjunto tener un set de parámetros w y b que sean los óptimos para cualquier imagen por lo tanto tienen que ser ajustados y para esto hablábamos que necesitamos encontrar una función que mira qué tan buenos son estos parámetros a manera de resumen podemos entonces decir que para convertir los scores los puntajes arrojados por nuestro modelo lineal a probabilidades lo que hacemos es el resultado de haber aplicado la multiplicación de la matriz de pesos con la matriz de entradas más vallas aplicar la función softmax la cual ya acabamos de describir y está y está dada por esta ecuación que está aquí en la parte de arriba y así tenemos la salida para la probabilidad arrojada para cada uno de los datos de entrada que en este caso fueron ejemplos de un perro perdón de un gato de un perro y de un ave y nuevamente el punto que estoy al que quiero llegar y es lo con lo que vamos a continuar es que ya teniendo estas probabilidades tenemos que evaluar qué tan buenos son los valores que tenemos en nuestros parámetros w y b para clasificar nuestros objetos o nuestras imágenes de entrada porque recordemos que esto todo esto lo podemos decir que es estar representado en la función neural networks que los y lo que recibe es la x w y b solamente podemos ajustar los parámetros asociados con cada neuronas decir los pesos sinápticos de cada neurona y las vallas o sesgos de cada neurona los cual también son parámetros de la neurona y decíamos que entonces tenemos que definir una función matemática de error que nos permita cuantificar qué tan buenos son los valores actuales para esto presentamos el concepto de los function o función de pérdida la cual para casos de clasificación multiclase se utiliza esta fórmula y vamos a analizar ahorita por qué esta fórmula es la correcta pero lo que nos está diciendo es que a la salida vamos a recordar que en este caso y con gorrito es este y con gorrito es el es la salida de la red neuronal que tenemos después de haber aplicado la función softmax es decir las probabilidades que nos está arrojando y lo que nos está diciendo esta fórmula es que a dichas probabilidades les vamos a calcular el logaritmo natural y el negativo de esa de esa función va a ser la función de error o la función de pérdida y lo que vamos a analizar ahora es por qué esta función es la función que nos va a permitir obtener esto y vemos también que la función originalmente está dada por y multiplicada por el logaritmo natural de y con gorrito donde y con gorrito es la predicción del modelo y y sin gorrito es la el valor actual de la salida y vamos a ver por qué porque funciona de esta manera y en este momento es importante recordar el concepto de aprendizaje supervisado que vimos en el primer vídeo donde recordemos que la implicación de este tipo de aprendizaje es que además de los datos de entrada para entrenar al modelo también serán las salidas para cada entrada es decir a qué clase o qué tipo de objeto es cada una de las entradas por ejemplo en este caso se trata de un gato le tenemos que decir o sea nosotros vemos que es un gato pero de alguna manera cuantitativa tenemos que indicarle al modelo que en realidad se trata de un gato eso es lo que representa el aprendizaje supervisado y esto lo utilizamos para evaluar la función de pérdida porque la función de pérdida está dada por y lo voy a poner aquí el producto de la clase correcta con el logaritmo natural de la predicción del modelo y aquí no voy a utilizar sub índices para hacer esto más más fácil de visualizar y después esto se va a sumar para la predicción de cada clase y vamos a ver por qué funciona cómo será esta información al modelo de red normal y con esto también vamos a ver por qué este yé desaparece de la fórmula y se convierte y esto se va a reducir bueno esto tiene un signo negativo y esto se va a reducir simplemente a menos el logaritmo natural de la predicción del modelo de red normal entonces vamos a ver por qué funciona esto de esta manera aquí por cierto también esta predicción aquí aquí es importante colocarlo esta predicción en esta fórmula debe tener aquí el índice j también porque estamos hablando de que se trata de la clase correcta por el logaritmo natural de la predicción arrojada por el modelo para la clase correcta esto es en este caso sería el logaritmo natural por ejemplo para el gato para esta predicción diríamos que la pérdida es el logaritmo natural de el valor arrojado por la red normal para el gato en este caso este valor de acá y bueno y por qué si originalmente decíamos que es la multiplicación de yé con el logaritmo natural de de yé concorrido por qué se reduce a esto y esto se reduce a esta fórmula porque la clase correcta para cada imagen se da en forma de vector 1 hot vector donde lo que se pasa es un vector del mismo tamaño del número de clases es decir un vector del mismo tamaño de la salida que va a tener el modelo donde se pasa un 1 para la clase correcta en este caso sería la primera clase tendría un 1 la segunda clase tendría un 0 y la tercera clase tendría otro 0 este sería el vector de salida con las etiquetas para cada dato con el que se va a evaluar el modelo con el que se va a entrenar el modelo por ejemplo para el caso para el caso del perro el vector yé estaría dado de la forma 0 porque el primer elemento es un gato el segundo elemento es un perro ahí sería 1 y el tercer elemento es una vez sería 0 y para el caso del pájaro o el ave sería un vector yé que estaría dado con la fórmula con la forma 0 0 1 ya que el primer elemento sería para la clase gato para la clase perro y finalmente en uno señalamos la clase la clase ave ahora si lo que hacemos implementamos esta fórmula si implementamos esta fórmula para calcular cuál es la función de pérdida cuál es la pérdida arrojada por evaluar esta imagen del gato lo que tendríamos es que nos está diciendo que es el negativo de la sumatoria para todas las clases de cada elemento cada elemento arrojado por nuestra red neuronal entonces vamos a hacer esto esto sería igual a la suma vamos a dejar el negativo aquí afuera y sería la suma de yé el valor correcto por el logaritmo natural de el valor predicho para ese mismo para esa misma clase entonces estamos hablando de que sería 1 por el logaritmo natural de 0.39 más y luego estaríamos hablando de que sería 0 por el logaritmo natural de 0.28 más nuevamente 0 por el logaritmo natural de 0.33 observamos estamos aplicando la fórmula estamos multiplicando la yé correcta por el logaritmo natural de el valor arrojado para esa misma clase por nuestro modelo que en este caso sería 0.39 luego sería 0 por el logaritmo natural de 0.28 este valor y luego 0 por el logaritmo natural de 0.33 por lo tanto todos estos valores dado que los que las clases se identifican con este vector 1 y este es el motivo 1 de los motivos por los que las clases se identifican así donde 1 hot significa que es 1 para la clase correcta y 0 para todas las demás todos estos valores van a ser 0 por lo tanto el único valor que va a quedar va a ser el de la clase correcta y esto se va a reducir a menos el logaritmo natural de 0.39 el cual es menos el logaritmo natural de la predicción o yé con gorrito para la clase correcta de acuerdo por lo tanto por eso esto se reduce a esta fórmula menos el logaritmo natural de las probabilidades arrojadas por el modelo de las salidas arrojadas por el modelo ahora va la pena darnos cuenta de por qué estamos utilizando esta función del logaritmo natural ya vimos por qué se reduce simplemente a esta función a menos el logaritmo natural si pensamos que tenemos un modelo de red normal que está arrojando valores de forma correcta tendríamos que la probabilidad de clasificar esta imagen de entrada de un gato como correctamente como un gato tendría que ser muy próxima a 1 si la función logaritmo natural en este caso esta gráfica de acá nos muestra menos logaritmo natural de yé prima vamos a decirlo así entonces si la yé prima o yé con gorrito es un valor cercano a 1 vamos a ver que la pérdida va a estar cercana a 0 entonces valores que sean aquí de 0.9 y mayores van a tener un valor de pérdida bajito en cuando sean igual a 1 pues va a ser de 0 la pérdida va a ser de 0 y realmente dado que son probabilidades el único valor que nos importa el único rango de valores yé con gorrito que nos interesan son entre 0 y 1 porque son probabilidades no pueden ser valores negativos y no pueden ser valores mayores a 1 entonces una probabilidad cercana a 1 nos va a dar una función logaritmo natural cercana a 0 lo que implica un error pequeño un costo pequeño de forma contraria vamos a considerar ahora la imagen de entrada del perro donde la predicción es 0.17 en el caso del perro la función de pérdida de li aquí indica cada uno de los elementos de las imágenes de entrada en el lote de entrada que estamos manejando en el minibatch va a estar dado simplemente por menos el logaritmo natural de la predicción para la imagen que en este caso sería 0.17 y si vemos en la gráfica 0.17 está tal vez por acá tiene una función tiene una pérdida mucho más alta que para el caso del gato por lo tanto cuando evaluamos la función de pérdida para esta imagen de entrada de tipo perro vamos a tener que es una pérdida mucho más grande que para el gato en este caso nos va a dar un valor de 1.77 en el caso del gato es de 0.94 y de forma similar por ejemplo para el caso del ave vamos a llamarla el de el ave va a ser igual menos el logaritmo natural en este caso de 0.41 y si buscamos por aquí 0.41 nos da por acá lo cual nos representa exactamente una pérdida de 0.89 y así vemos que mientras más lejana sea la predicción de nuestro modelo para la clase correcta es decir mientras menos se aproxima a uno para la clase correcta en este caso el ejemplo del perro que nos da una probabilidad de 0.17 solamente mayor va a ser la pérdida sin embargo nosotros no queremos que nuestro modelo nada más aprenda a predecir correctamente perros o gatos nosotros queremos un modelo que evalúe correctamente la mayor cantidad de imágenes de entrada sin importar a qué clase pertenecen por lo tanto el paso final es utilizar todas estas pérdidas que tenemos aquí y agruparlas en un solo valor para lo cual lo que se va a hacer es implementar esta función que tenemos aquí donde lo único que nos dice es que va a ser la sumatoria desde 1 hasta m y en este caso m representa el número de ejemplos en este caso es el número de imágenes donde nos lo que vamos a hacer es entonces hacer la sumatoria dividida por el número de ejemplos y esto es otra cosa más que el promedio de todas estas pérdidas entonces la función de costo final para este lote de imágenes sería 1 sobre 3 de 0.94 más 1.77 más 0.89 con esta fórmula calculamos cuál es la función de pérdida de todas estas imágenes que es aproximadamente de 1.2 este valor nos va a dar 1.2 por eso lo que queremos es encontrar parámetros doble u y b que minimizen esta función de costo a éstas le llama función de costo o cost function usualmente se representa con la letra j que en este caso en este ejemplo habíamos dicho que es 1.2 nuestro problema ahora va a ser cómo minimizar esta función de costo de qué manera sistemática podemos encontrar y ajustar los valores doble u y b que resultan en el mejor modelo de clasificación porque después de todo nosotros no podemos cambiar los datos de entrada solamente tenemos control sobre doble u y sobre b en el próximo vídeo vamos a ver qué herramienta matemática podemos aplicar para que nos ayude a minimizar esta función de costo con respecto a los valores doble u y b muchas gracias en este vídeo presentamos la función de pérdida que nos permite evaluar qué tan buenos son los parámetros actuales de nuestra red neuronal artificial por cierto esta función de pérdida se llama cross entropy ahora lo que nos necesitamos es encontrar una forma de ajustar los parámetros de nuestra red neuronal para que la función de pérdida sea minimizada por lo tanto tenemos que pensar en una herramienta matemática que nos permita minimizar una función la función de pérdida con respecto a los parámetros y por supuesto dicha herramienta es el cálculo por lo tanto en el siguiente vídeo vamos a presentar el algoritmo que nos permite realizar de forma iterativa ajustes a nuestros parámetros para reducir la función de costo continuamos [MÚSICA] [MÚSICA]