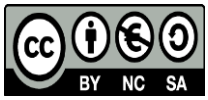
uc3m | Universidad Carlos III de Madrid

OPENCOURSEWARE
REDES DE NEURONAS ARTIFICIALES
Ricardo Aler



Tema 7. Introducción a Deep Learning

En este tema se introducen los conceptos principales de Deep Learning. Para introducir los contenidos, se sigue la siguiente línea argumental. En primer lugar, se intenta entender cual es la función de la capa oculta en las redes de neuronas más comunmente utilizadas (una sola capa oculta). En segundo lugar, se motiva la introducción de más capas ocultas, lo que daría lugar a redes « profundas ». Pero en este caso, se muestra que la existencia de muchas capas ocultas lleva aparejado el problema del « vanishing gradient » (desvanecimiento del gradiente) para el algoritmo típico de aprendizaje (backpropagation). O lo que es lo mismo, que el gradiente (el cual determina la velocidad de aprendizaje) se hace muy pequeño para las capas cercanas a la entrada, lo que implica que prácticamente no se produce aprendizaje en dichas capas. El resto del tema explica algunas de las soluciones a este problema: pre-entrenamiento no supervisado (unsupervised pre-training), las « rectified linear units » o RELUs, y las redes convolucionales.

7.1. Introducción

Se comienza el tema motivando las técnicas de Deep Learning por sus resultados en los últimos años. Uno de los más conocidos, ocurridos en Marzo de 2016, es cuando el sistema Alpha Go, desarrollado por Google DeepMind ganó a Lee Sedol, uno de los maestros del juego del Go. Recordemos que un sistema de inteligencia artificial venció a Gary Kasparov en 1997 al ajedrez, pero el juego del Go quedaba todavía fuera del alcance de estos sistemas. Alpha Go utiliza varias técnicas de inteligencia artificial. En concreto, árboles de búsqueda guiados por una « value network » y una « policy network ». Ambas son redes de neuronas profundas, inicialmente entrenadas para imitar el juego humano (disponible en bases de datos con aproximadamente 30 millones de movimientos), y posteriormente auto-entrenadas jugando partidas contra sí mismas. A destacar que el hardware utilizado disponía de 1920 CPUs y 280 GPUs. Se comenta también que algunos de los mejores resultados en reconocimiento de imágenes, (dominio ImageNet), reconocimiento del habla o reconocimiento de caracteres.

Se comienza recordando la arquitectura de una red estándar con una sola capa oculta y su estructura de entrada / capa oculta / salida ; y cual es el procesamiento que ocurre en cada una de las neuronas : suma ponderada de las entradas con los pesos de la red seguida de la función de activación, típicamente sigmoidal. Se recuerda que las redes de neuronas con una

capa oculta son aproximadores universales, siempre que se disponga de un número de neuronas ocultas suficientes. Si esto es así ¿por qué puede tener sentido añadir más capas ocultas?

La siguiente parte de la clase intenta motivar esta cuestión, a través de tres puntos:

1. Las RRNN con una capa oculta clasifican en dos etapas. Puede ser interesante añadir más etapas.
2. Algunos problemas se pueden resolver de manera más sencilla usando más capas.
3. El cerebro parece hacer un procesamiento secuencial de la información.

En primer lugar, se explica que el proceso que las redes de neuronas siguen para resolver problemas de clasificación consta de dos etapas: proyección del espacio original de los datos (feature space) a otro espacio (primera etapa) seguido de la clasificación lineal en este nuevo espacio (segunda etapa). Se va a intentar ilustrar este proceso con una red de neuronas con sólo dos neuronas a la entrada y dos en la capa oculta, porque esto permite visualizar los espacios origen y destino. Cada neurona oculta u_i está representada por la ecuación

$$u_i = S(x_1 * w_{1i} + x_2 * w_{2i} + w_i)$$

donde los w_{ji} son los pesos y S es la función de activación (comúnmente, la sigmoide). $x_1 * w_{1i} + x_2 * w_{2i} + w_i = 0$ es la ecuación de una recta en un espacio de dos dimensiones. La cantidad $x_1 * w_{1i} + x_2 * w_{2i} + w_i$ será positiva o negativa según el punto (x_1, x_2) esté a la izquierda o derecha de la recta, y será mayor en valor absoluto cuanto más lejos esté de dicha recta. La aplicación de la sigmoide es una transformación monótona no lineal de dicha cantidad. De esta manera, se puede entender el espacio transformado representado por las dos neuronas ocultas, el cual representa fundamentalmente si el dato original estaba a la izquierda o a la derecha de las rectas que representan a cada una de las dos neuronas ocultas. Por último, la capa de salida intenta hacer una separación lineal de los datos proyectados en el espacio transformado. Aunque una separación lineal en el espacio original no era posible (para el conjunto de datos de ejemplo), en el espacio transformado por las neuronas ocultas sí que lo es (o casi). Dado que la introducción de una capa oculta permite simplificar el problema de clasificación, nos podemos preguntar si la inclusión de más capas ocultas permitiría resolver problemas de clasificación incluso más complejos, y esta es la primera motivación para las redes profundas. En las transparencias se muestra un ejemplo del dominio SEMEION, cuyo objetivo es clasificar dígitos escritos a mano (10 clases, dígitos 0 a 9). El espacio de los datos (feature space) consta de 16x16 entradas. Se muestra que incluso con sólo dos neuronas ocultas, la red es capaz de agrupar los datos en grupos más fácilmente separables.

En segundo lugar se muestra como para algunos problemas de reconocimiento de imágenes, descritos en el siguiente artículo, es preferible incrementar el número de capas que el número de neuronas dentro de una única capa, para mejorar la capacidad de generalización:

> Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V. (2014). Multi-digit number recognition from Street View imagery using deep convolutional neural networks. In *International Conference on Learning Representations*

En tercer lugar, se ilustra que la neurología de la visión muestra que el procesamiento ocurre también de manera secuencial, desde lo más concreto y cercano al sentido, hasta lo más general, pasando por múltiples procesamientos intermedios.

7.2. El problema del « Vanishing Gradient »

Se comienza explicando que el algoritmo de backpropagation funciona bien con una o unas pocas capas ocultas, pero no a medida que se incrementa el número de capas. De manera empírica, y partiendo de un problema de reconocimiento de dígitos, se muestra como el gradiente se va haciendo más y más pequeño a medida que nos alejamos de la capa de salida. Dado que el gradiente representa la velocidad a la que se actualizan los pesos, gradientes muy pequeños implican que los pesos de las capas lejanas de la capa de salida (y cercanas a la capa de entrada), prácticamente no se actualizan. Es decir, casi no se produce aprendizaje para las capas cercanas a la de entrada. Se muestran varios ejemplos donde se puede ver que en una red con tres capas ocultas y una de salida, la capa más cercana a la entrada aprende aproximadamente 100 veces más lento que la capa más cercana a la salida.

¿ Cuál es la razón de este fenómeno? Una posible explicación es que el gradiente de los pesos de la primera capa se calcula mediante la regla de la cadena:

$$dE/dw \sim S'(\ldots) * S'(\ldots) * S'(\ldots) * \ldots$$

donde S'(…) es la derivada de la función de activación. Cuando esta función es la sigmoide, su derivada es $S'(x) = S(x) * (1-S(x))$, cuyo máximo es 0.25 que ocurre para el valor cero (para el resto de valores es incluso más pequeño). Se puede ver cómo la aplicación de la regla de la cadena repetidamente con valores menores que uno, puede llegar a dar lugar a valores muy pequeños para el gradiente. Más pequeños, cuanto más lejos estemos de la capa de salida. El resultado es que las capas más cercanas a la entrada casi no tienen aprendizaje (o este es muy lento) y no es útil añadir más capas.

7.3. Pre-entrenamiento no supervisado

Una primera solución al problema del « vanishing gradient » es entrenar los pesos de las capas ocultas de manera no supervisada. Es decir, de manera que su entrenamiento no dependa del resultado deseado, que sólo está disponible de manera directa en la capa de salida. Para inicializar los pesos de la capa oculta de manera no supervisada, se puede utilizar un tipo de rede de neuronas conocido como autoencoder. El objetivo de un autoencoder es reconstruir la entrada (es decir, la salida es idéntica a la entrada), pero usando menos neuronas en la capa oculta que en la capa de entrada. De esta manera, se fuerza al autoencoder a encontrar la información relevante en las neuronas de entrada. Es una especie de compresión de la información presente en la entrada. En las transparencias se puede ver como un autoencoder con sólo dos neuronas ocultas es capaz de encontrar agrupamientos de los 10 dígitos del dominio de reconocimiento de dígitos SEMEION. Los pesos aprendidos por el autoencoder se pueden utilizar para inicializar los pesos de la primera capa oculta de la red profunda. Esto se puede hacer repetidas veces, pero ahora tomando como entrada la salida de la primera capa oculta. De esta manera, se van inicializando los pesos de las sucesivas capas ocultas. A esta aproximación se la denomina Stacked Autoencoders. La capa final ya se puede entrenar de manera supervisada, con backpropagation, dado que está en contacto con la salida. Finalmente, toda la red se puede entrenar con backpropagation. Dado que el pre-entrenamiento no supervisado ya encontró una buena inicialización de los pesos de las capas ocultas, el backpropagation puede limitarse a hacer un ajuste fino de dichos pesos. Una aproximación similar usa Restricted Boltzmann Machines (RBS's) en lugar de autoencoders para construir las denominadas Deep Belief Networks (DBN's).

Rectified Linear Units (RELU) y dropout

Otras dos técnicas que se utilizan para hacer más tratable el entrenamiento de redes profundas son la RELUs y dropout. Las RELUs son funciones de activación a usar en substitución de la sigmoide. Dado que una RELU es lineal para valores mayores que cero, y su derivada en ese rango es 1, no sufre del desvanecimiento del gradiente en ese rango (aunque para valores menores que cero su valor es constante e igual a cero).

Dropout es una técnica para reducir el problema de sobreaprendizaje, que se puede producir por la gran cantidad de parámetros (pesos) presente en redes profundas. En cada etapa de entrenamiento, y con cierta probabilidad, se elimina un conjunto de neuronas que no estará disponible en esa etapa. Dado que esto va cambiando con las etapas de aprendizaje, se fuerza a que las neuronas no se sobreadapten a las salidas de otras neuronas concretas.

Comentar por último que otra manera en la que se está abordando actualmente el problema del desvanecimiento del gradiente es por fuerza bruta, usando Graphical Processing Units (GPUs) que permiten incrementar enormemente la capacidad de proceso necesaria para el entrenamiento de las redes.

7.4. Redes Convolucionales (CNN)

Las redes convolucionales son arquitecturas especializadas de redes profundas, orientadas al reconocimiento de imágenes (aunque también se las puede utilizar en dominios de sonido o texto). Entre otras cuestiones, las CNNs intentan superar dos cuestiones presentes en las redes de neuronas estándar : dotar de invarianza translacional y reducir el número de pesos a ajustar.

La arquitectura de las CNNs es una alternancia de capas de convolución y capas de submuestreo, siendo la última etapa de la red una red estándar. Las capas de convolución consiguen la invarianza translacional usando filtros que analizan regiones reducidas de la imagen. Por ejemplo, si la imagen es de 8x8 pixels, un filtro de 3x3 se centraría en regiones de menor tamaño, con la característica de que dicho filtro se aplicaría a todas las regiones de la imagen. El objetivo es localizar determinada característica (por ejemplo, un segmento vertical) en cualquier lugar de la imagen (de ahí la invarianza translacional). Un filtro es equivalente a una red de neuronas, pero con una única neurona. Es decir, el filtro hace una suma ponderada de la entrada por los pesos del filtro y después les aplica una función de activación (sigmoide, RELU, …). El filtro se va desplazando por la imagen y generando un mapa de características. Por ejemplo, a partir de una imagen de 8x8 y un filtro de 3x3, se puede generar un mapa de características de 6x6. Cada punto del mapa indica si se detectó la característica representada por el filtro en la zona de la imagen correspondiente. Típicamente, cada capa de convolución tiene múltiples mapas, cada uno con su propio filtro (y por tanto, cada mapa tiene filtros distintos con pesos distintos). Por ejemplo, un mapa podría detectar segmentos verticales, otro horizontales, otro oblicuos, etc.

El objetivo de las capas de subsampling es reducir la resolución espacial. Consiguen esto aplicando filtros de nxn sobre la salida de la capa de convolución, y aplicando la función máximo (o media). El máximo indica que la característica identificada por la anterior capa de convolución se identificó en al menos uno de los nxn valores.

Tras varias capas de convolución y subsampling, se cierra la arquitectura con una red de neuronas estándar con varias capas ocultas. La red convolucional completa se entrena directamente con backpropagation. El uso de filtros en la capa de convolución permite disminuir drásticamente el número de pesos a ajustar : si el filtro es de 3x3, un mapa de características de la capa de convolución tendrá que ajustar meramente 9 pesos (compárese con una red estándar cuya entrada fuera una imagen de 8x8, en la que habría que entrenar 8*8+1 = 65 pesos).