3ª Emisión

Módulo 08

Introducción al Deep Learning

Edgar Morales Palafox

DGTIC
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Dirección General de Cómputo y de Tecnologías de información y Comunicación
Dirección de Docencia en TIC
Educación Continua
1971 - 2021

Objetivo

El participante identificará las características de las redes neuronales profundas y las técnicas para evitar el sobreajuste y mejorar su entrenamiento.

Contenido

- Redes neuronales profundas
  - Motivación de redes neuronales profundas
  - Sobreajuste y regularización
  - Explosión y desvanecimiento del gradiente
  - Capas de normalización

Motivación de redes neuronales profundas

Imagenet Large Scale Visual Recognition Challenge (ILSVRC)

Es un concurso anual de visión por computadora desarrollado sobre un subconjunto de un conjunto de datos de visión por computador disponible públicamente llamado ImageNet.

2010

- 1000 categorías
- 1.4 millones de imágenes

https://image-net.org/

2011, 74.3% Método clasico CV
2012, 84.3% Convolutional NN
2015, 96.4% (Imagenet lo consideró resuelto)

Hoy en día, convnets (convolutional neural network) es la base de los algoritmos de visión por computadora

Sobreajuste y regularización

Modelo de liner

Un modelo de liner es el que sigue una línea recta en el modelo de predicción . Puede tener un sólo atributo para predecir el valor, o múltiples atributos para predecir el valor, y la ecuación se ve así:

Para una variable $$h(X) = \theta_0 + \theta_1 X$$

Para múltiples variables $$h(X) = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \theta_3 X_3 \ldots . \theta_n X_n$$

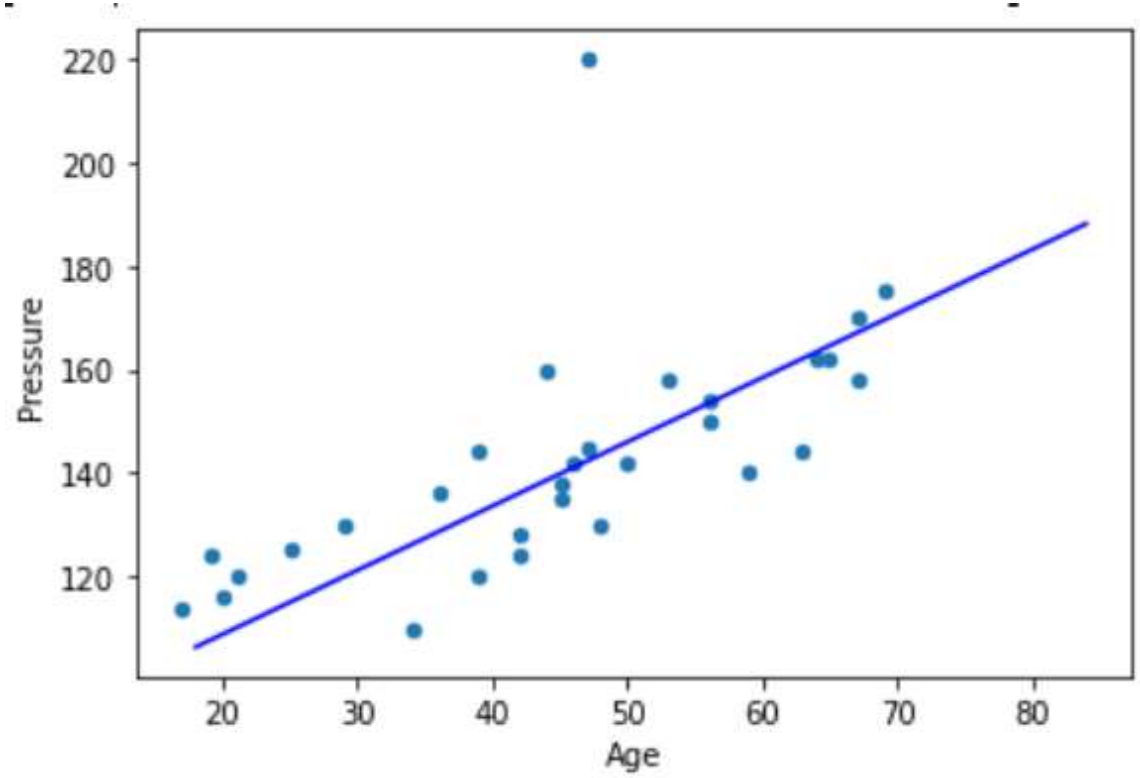
Theta-0 es el **intercepto** y theta-1 a theta-n son los pendientes correspondientes a su atributo X-1 a Xn

Regresión lineal

La regresión lineal, en ocasiones puede ser representado gráficamente como una línea recta, pero hay que considerar que hay una hipótesis polinomial, esta línea podría ser una curva. En cualquier caso, modela las relaciones entre una variable dependiente escalar ”**y**“ y uno o más valores explicativos denotados por ”**x**”.

La regresión lineal es el algoritmo que aprende la dependencia entre cada uno de los ya conocidos ”**x**” y ”**y**”, tanto así que podemos usarlo para predecir ”**y**” para una muestra no conocida de ”**x**”. Dicha dependencia estadística se llama una *hipótesis* y usualmente se denota por **h(θ)**.

Predecir la presión arterial de una persona, según su edad



Cada punto azul representa nuestra data de muestra y la línea azul es la hipótesis que se debe aprender
Suponiendo que nuestra hipótesis es h(x)=84+1.24xh(x)=84+1.24x,

¿Cómo encontrar los dos parámetros θ0 y θ1?

Necesitamos definir una *función de costo*. Esencialmente, lo que hace una función de costo es calcular el error de raíz al cuadrado entre el modelo de predicción y la salida como tal.

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Por ejemplo, nuestra hipótesis predice que para alguien que tiene 48 años, su presión arterial debería ser de **h(48)=84+1.24∗48=143mmHg**; sin embargo, en nuestra muestra de entrenamiento, tenemos el valor de **130mmHg**.Por lo tanto, el error es **(143−130)2 =169**. Ahora necesitamos calcular este error en cada entrada en nuestro conjunto de datos de entrenamiento, luego sumarlo todo.

$$(\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2)$$

De la sumatoria se tomar el valor significativo. Esto nos da un único número escalar, el cual representa el costo de la función. Para el ejemplo se obtiene:

```
J(Theta) = 151.38
```

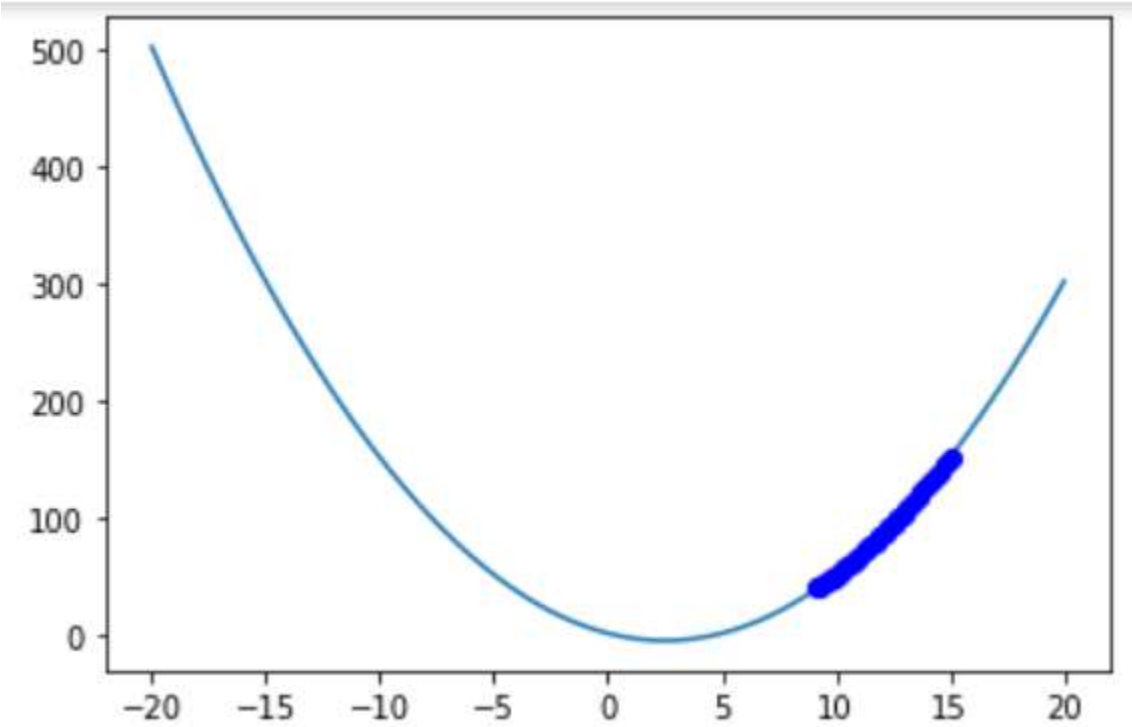
Ahora, necesitamos encontrar los valores de **θ** tanto que el valor de nuestra *función de costo* es mínimo.

https://colab.research.google.com/drive/1bWpXIehqiP2lAmxLFffy8vVX3X7wY__J#scrollTo=QpCe_p4xtCkX

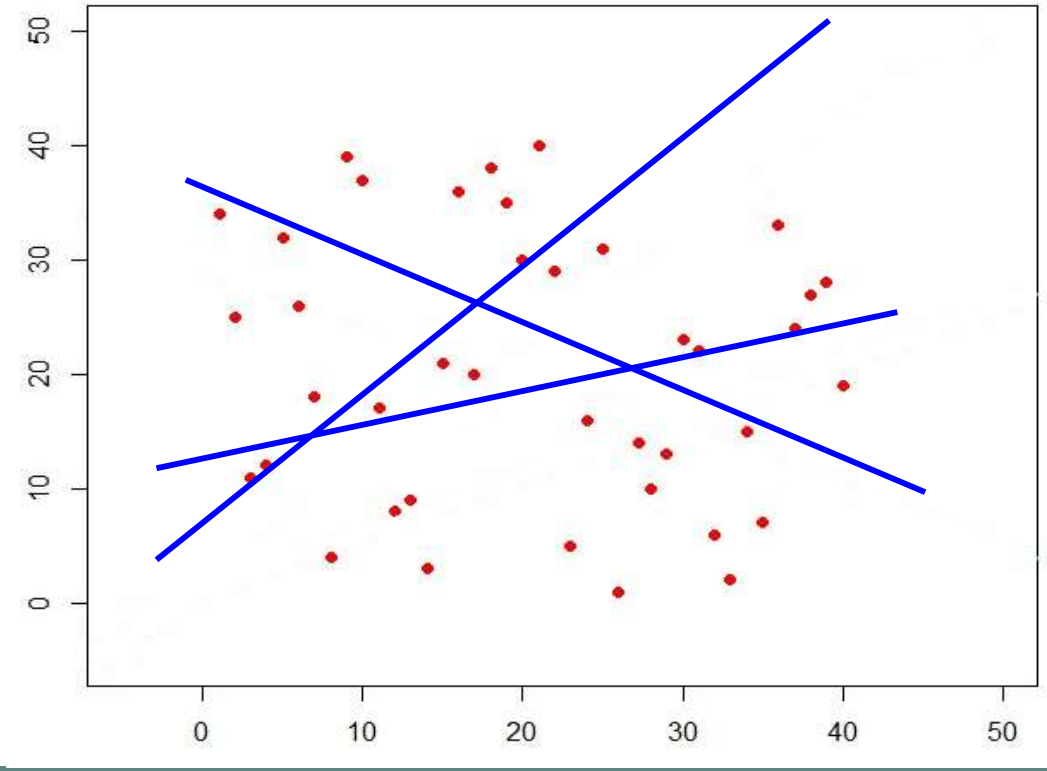
¿Pero cómo podemos hacer eso?

Hay muchos algoritmos posibles, pero el más popular es *gradiente por descenso*.

Se encuentra un mínimo local



Podría haber tantas posibilidades para ajustar una línea recta en este tipo de conjunto de datos

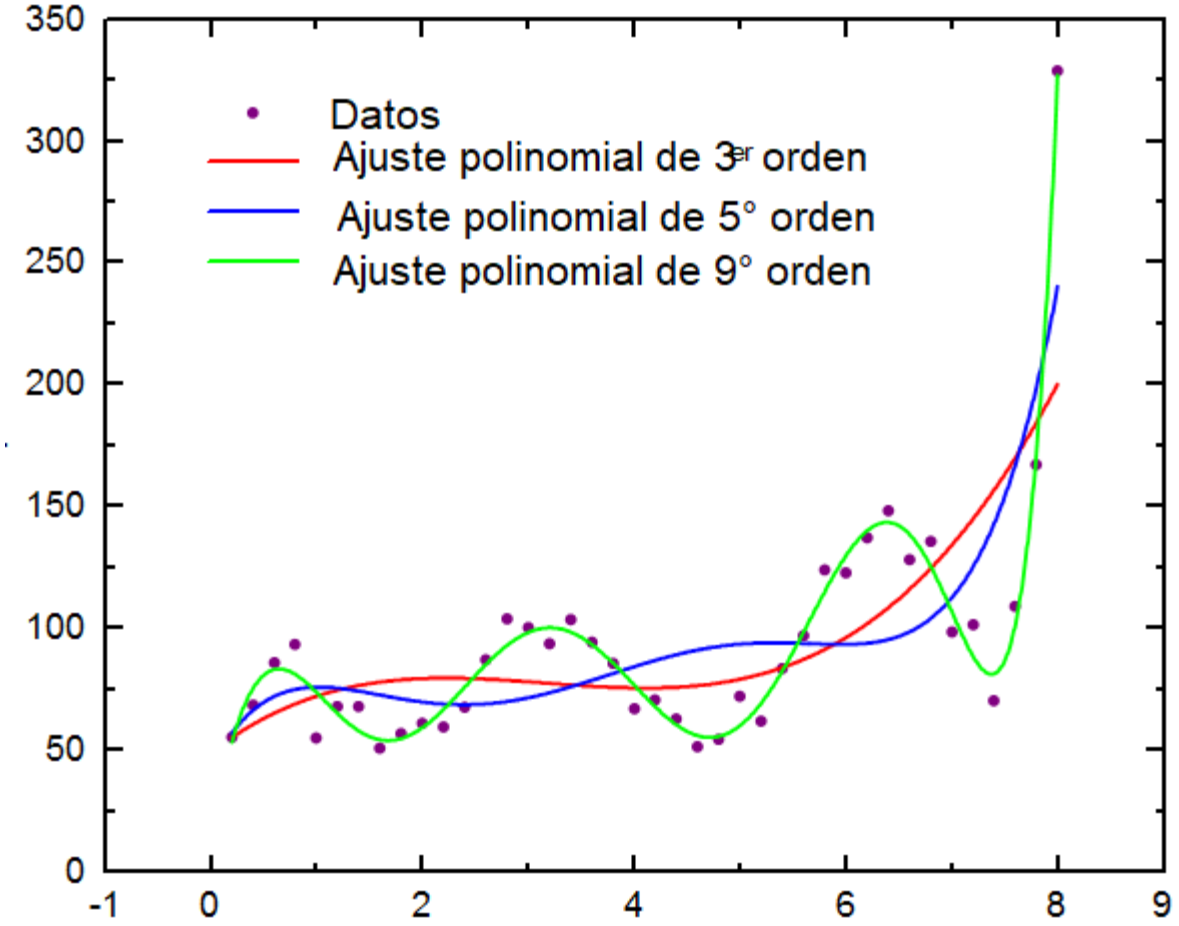


Por lo tanto, comenzamos a usar una ecuación polinomial de la forma que se muestra a continuación:

$$h(X) = \theta_0 + \theta_1 X^1 + \theta_2 X^2 + \theta_3 X^3 \ldots . \theta_n X^n$$

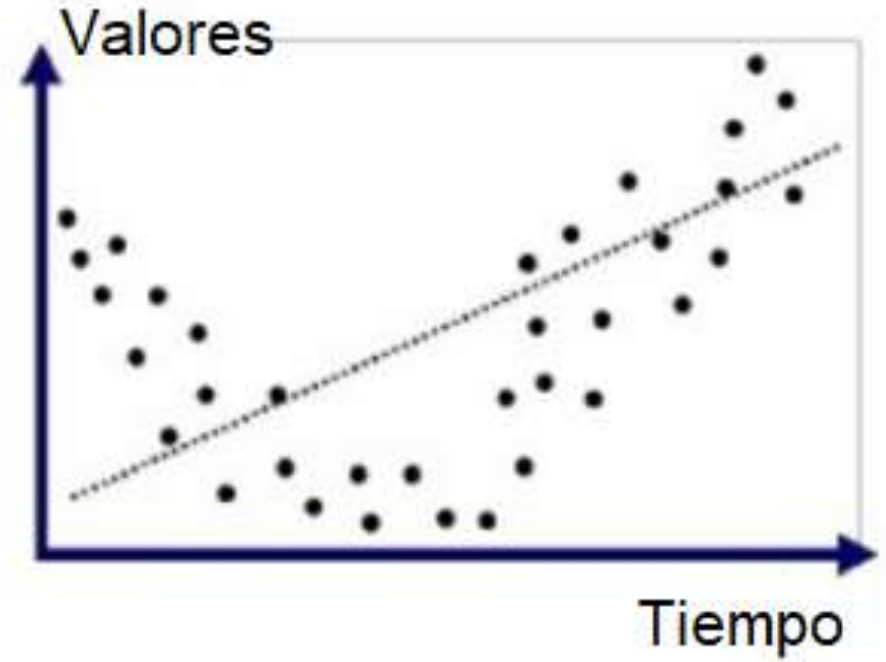
Donde n es el grado de la ecuación, Ɵ es la pendiente

Lo que hace es comenzar a formar una línea curva que pueda representar mejor los puntos de datos en comparación con una línea recta. Cuando solo teníamos una theta, eso significa que solo teníamos una pendiente de la dirección y, por lo tanto, tenemos una línea recta, pero si tenemos muchas tetas significa que hay muchas pendientes y, por lo tanto, nuestra línea puede cambiar de dirección de muchas maneras diferentes.

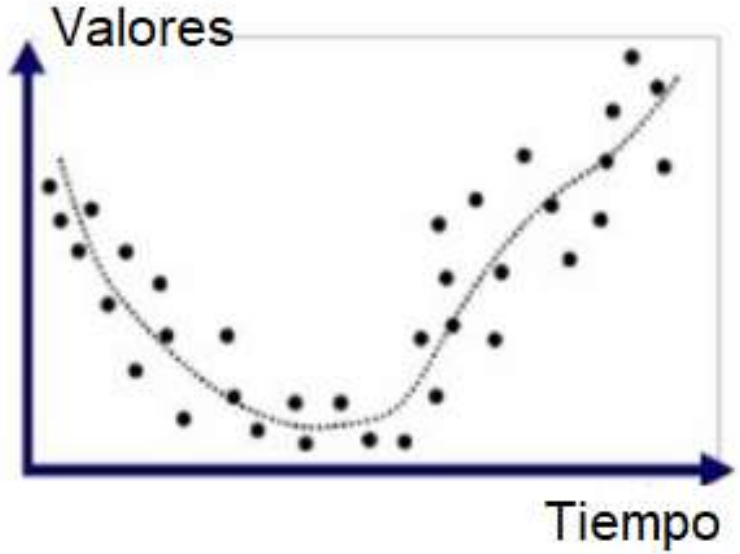

Datos
Ajuste polinomial de 3er orden
Ajuste polinomial de 5° orden
Ajuste polinomial de 9° orden
350
300
250
200
150
100
50
0
-1
0
2
4
6
8
9

¿Para qué nos sirve la regularización?

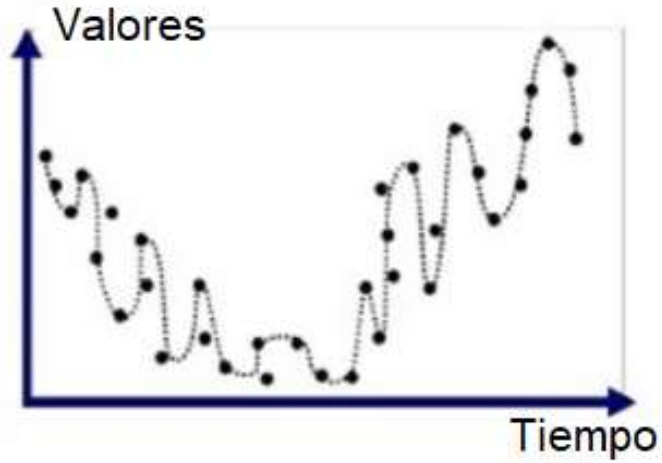
Underffited: La línea hipotética que dibujamos no sigue realmente la misma tendencia que los puntos. De esta manera, nuestro modelo no ofrece una imagen detallada de nuestros datos.



Goodfitted : mediante el uso de una ecuación polinomial , agrega complejidad a la línea que puede tomar diferentes tipos de formas, mientras que, si tiene una sola variable, diga ‘X’ y predice ‘Y’, entonces solo está creando una sola línea.



Overfitting : Al hacer que la ecuación sea polinomial puede configurarla para que coincida con sus puntos de datos, sin embargo, si está configurando una línea hipotética en la medida en que trata de pasar cada punto de datos posible, entonces dice que su modelo es overfit.



Regularización

Si el modelo tiene demasiada precisión, se sugiere eliminar el grado más alto. Esto es eliminar los términos del final de la ecuación.

Eliminar

$$h(X) = \theta_0 + \theta_1 X^1 + \theta_2 X^2 + \theta_3 X^3 \ldots . \theta_n X^n$$

$$h(X) = \theta_0 + \theta_1 X^1 + \theta_2 X^2$$

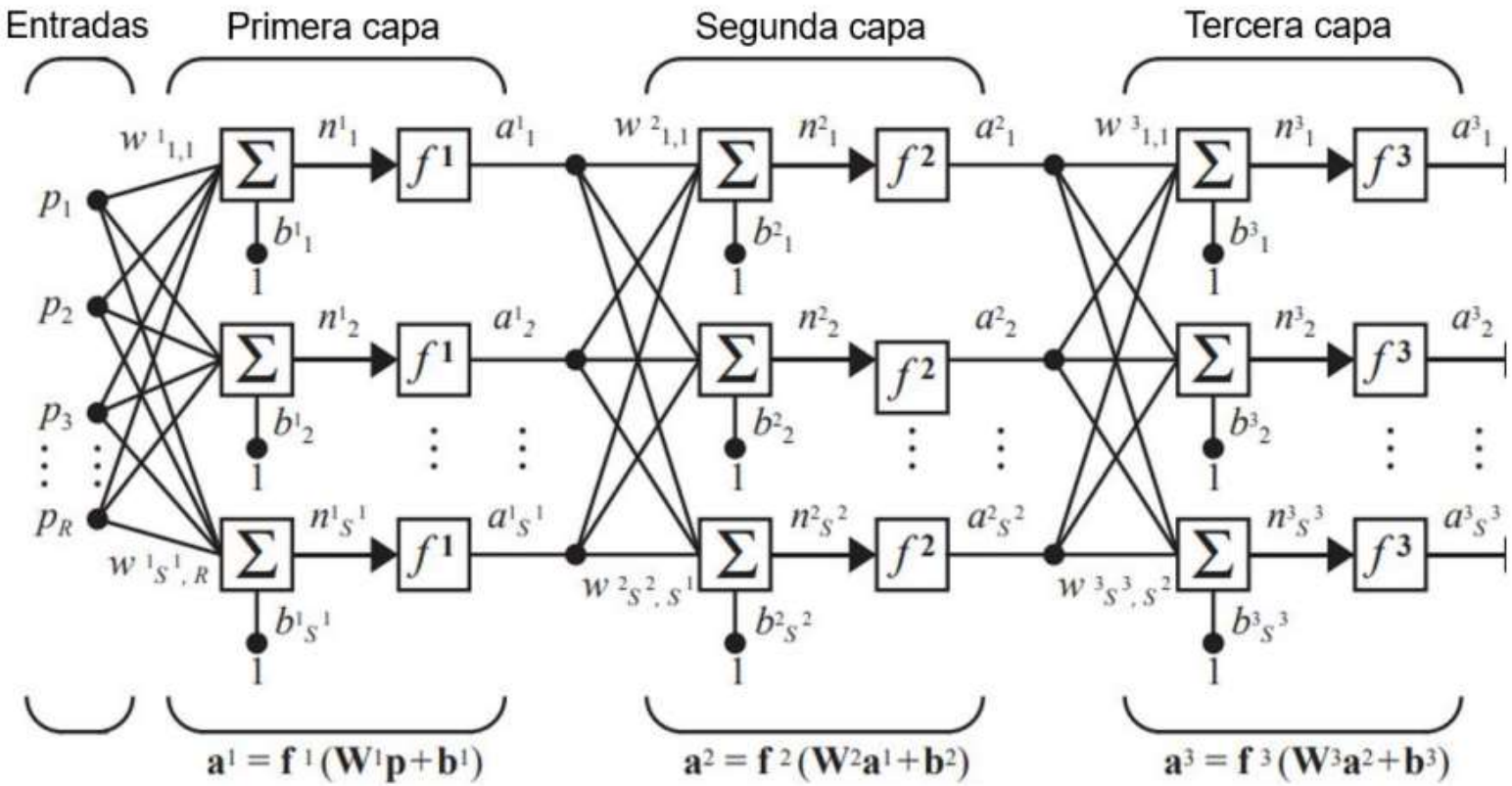
Explosión y desvanecimiento del gradiente

Al entrenar las redes neuronales artificiales mediante métodos de aprendizaje basados en descenso estocástico de gradiente y retropropagación, en cada iteración los pesos de las neuronas se actualizan mediante una proporción de la derivada parcial de la función de error respecto al peso actual.

.5
1.5

.25
2.25

0.0625
5.0625



$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

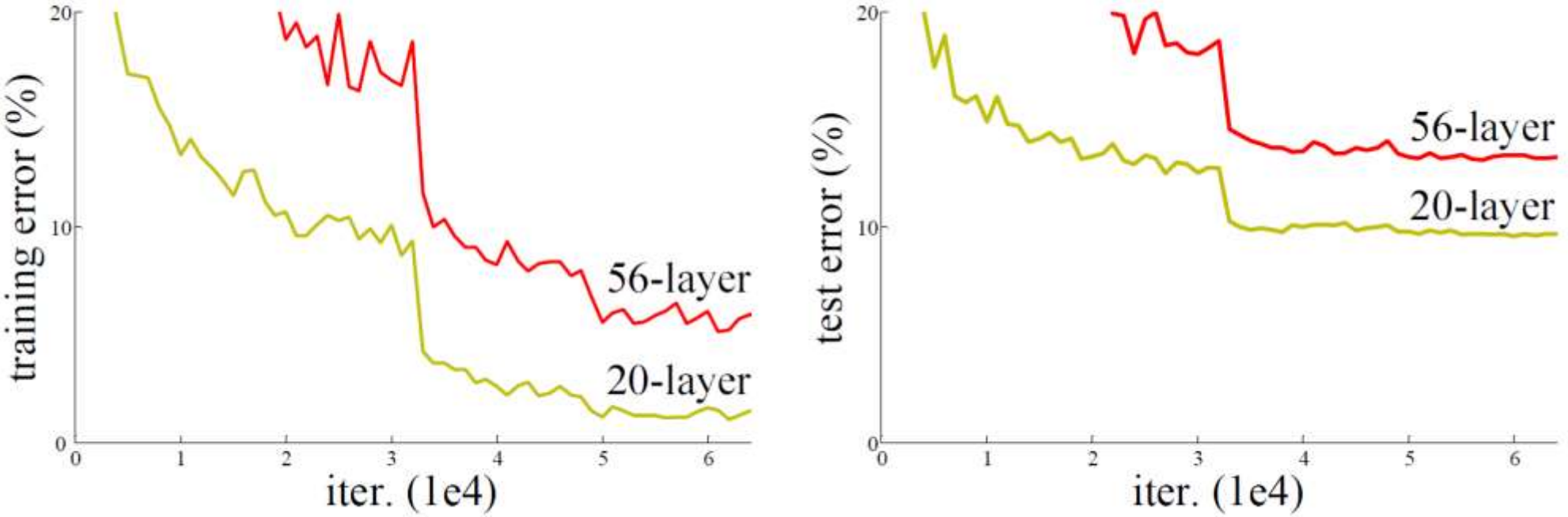
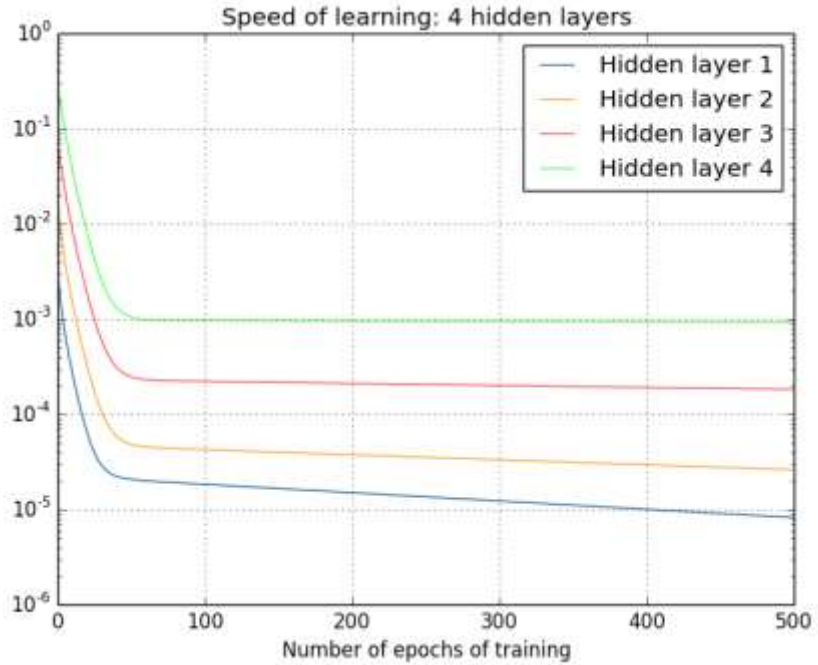


Imagen tomada de He et al. *Deep Residual Learning for Image Recognition*, 2015

Explosión y desvanecimiento del gradiente: 4 capas ocultas

- Gradientes de primeras capas se vuelven muy pequeños si la red es muy profunda
- Muy lento actualizar pesos de estas capas



Tomado de http://neuralnetworksanddeeplearning.com/chap5.html

Capas de normalización

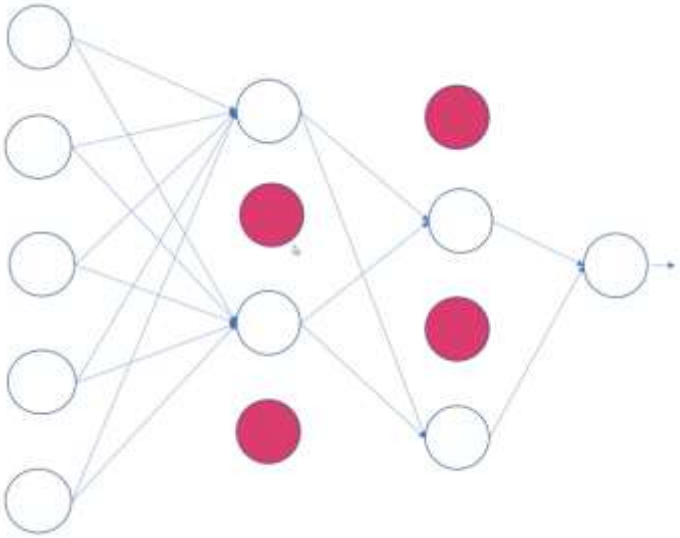
Entrenar redes neuronales profundas con decenas de capas es un desafío, ya que pueden ser sensibles a los pesos aleatorios iniciales y la configuración del algoritmo de aprendizaje.

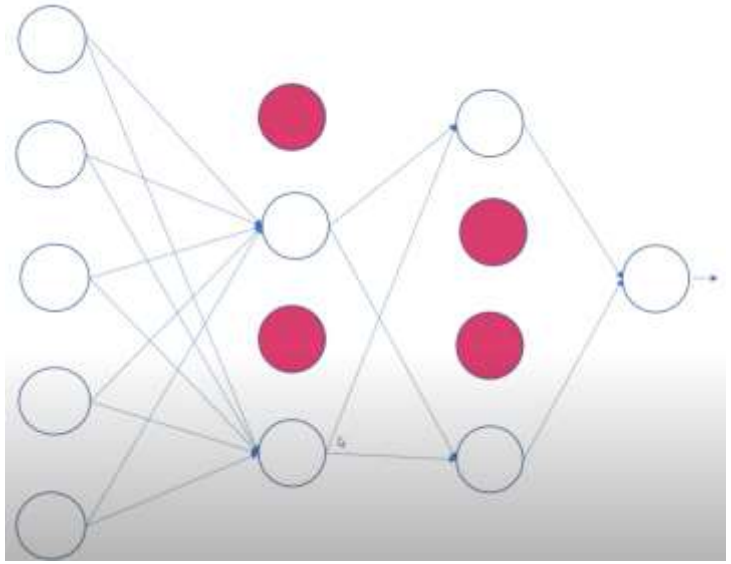
Hay dos métodos muy importantes que ayudan a las redes neuronales en el proceso de entrenamiento: Dropout y Batch

Dropout

Es un método que desactiva un numero de neuronas de una red neuronal de forma aleatoria. En cada iteración de la red neuronal dropout desactivara diferentes neuronas, las neuronas desactivadas no se toman en cuenta para el **forwardpropagation** ni para el **backwardpropagation** lo que obliga a las neuronas cercanas a no depender tanto de las neuronas desactivadas. Este método ayuda a reducir el **overfitting** ya que las neuronas cercanas suelen aprender patrones que se relacionan y estas relaciones pueden llegar a formar un patrón muy especifico con los datos de entrenamiento.

Activación y desactivación de neuronas en las capas





https://colab.research.google.com/drive/134Ib-CgC_-5XJpqEHHwkZEksUYIP_P8x#scrollTo=UoU2OSraDSsg

Normalización Batch

La normalización por lotes es una técnica para entrenar redes neuronales muy profundas que estandariza las entradas a una capa para cada mini-lote. Esto tiene el efecto de estabilizar el proceso de aprendizaje y reducir drásticamente la cantidad de épocas de entrenamiento necesarias para entrenar redes profundas.

https://colab.research.google.com/drive/1D-95a72ZYa9XTaVZCU1c7i6dAFKTI2TM#scrollTo=EcseZRE2ddoe

Referencias

Libros

- Deep Learning, 2016. https://amzn.to/2NJW3gE.
- Pattern Recognition and Machine Learning, 2006.https://amzn.to/2Q2rEeP
- Neural Smithing: Supervised Learning in Feedforward Articial Neural Networks, 1999. https://amzn.to/2poqOxc

Artículos

- Practical Recommendations for Gradient-Based Training of Deep Architectures, 2012. https://arxiv.org/abs/1206.5533
- Neural Network FAQ. ftp://ftp.sas.com/pub/neural/FAQ.html

Contacto

Edgar Morales Palafox

Doctor en ciencias de la computación

Edgar_morales_p@yahoo.com.mx

Tels: 55 3104 1600