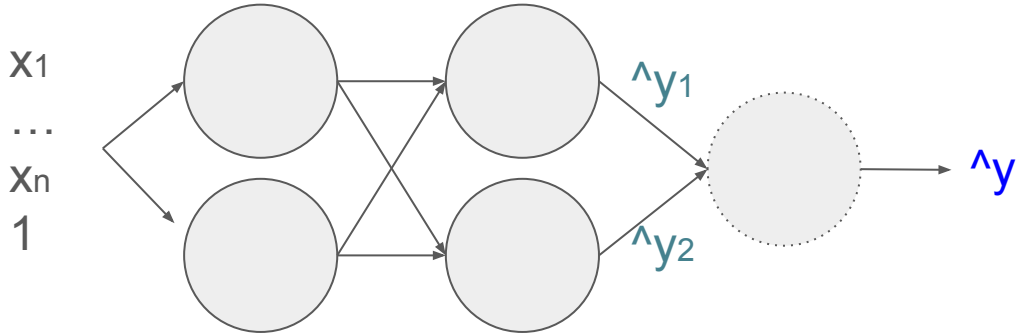
Regresión y Clasificación con RNAs

U-TAD

Clasificación con Redes Neuronales Artificiales

- Las Redes Neuronales Profundas pueden utilizarse para predecir un valor discreto
- Para predecir un único valor discreto, la *output layer* debe estar formado por una única neurona
- También pueden utilizarse RNAs para predecir varios valores discretos de manera simultánea
- Para predecir varios valores discretos se requiere una neurona por cada uno de los valores en la *output layer*



Perceptrón Multicapa: Clasificación

- Las RNAs utilizadas para tareas de clasificación binaria, en general, van a utilizar la función de activación sigmoide
- El resultado de la Red Neuronal es la probabilidad de que el ejemplo pertenezca a la clase positiva
- Pueden realizar clasificaciones binarias simultáneas, en este caso, se utilizarán dos neuronas en la output layer con funciones de activación sigmoide. Por ejemplo, quiero clasificar emails como spam o legítimos y, a la vez, como importantes o no. Por lo que *y = [0, 1] (vector formado por dos números)*
- Pueden realizar clasificaciones de más de los clases (multiclase), en este caso, se utilizarán tantas neuronas como clases se desean predecir y la **función de activación softmax.** Es decir, quiero clasificar el ejemplo en varias clases (no binario), por ejemplo, si aparece en una imagen una persona, un vehículo, …
- En la clasificación multiclase, la salida de las neuronas de la *output layer* debe sumar 1

Perceptrón Multicapa: Clasificación

Resumiendo, tendríamos:

- Clasificación binaria $y=0|1$

  a = sigmoid()

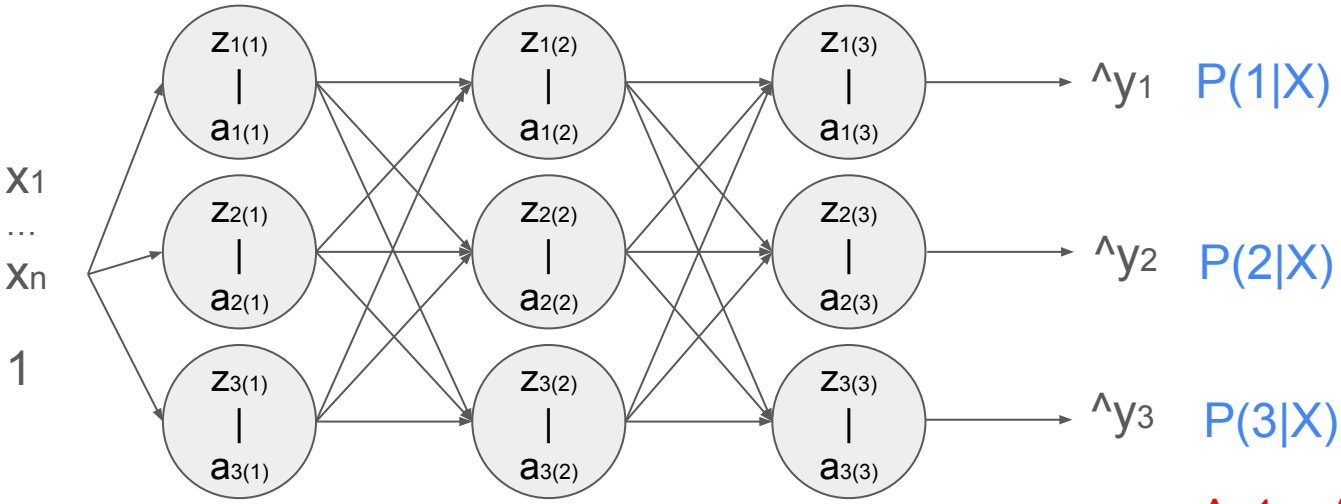
- Clasificación binaria múltiple (multilabel binary classification) $y_1=0|1$, $y_2=0|1$

  a = sigmoid()

- Clasificación multiclase (multiclass classification) $0<=\hat{y}_i>=1$, $y_1+y_2+...+y_n = 1$

  a = softmax()

Introducción a la Clasificación Multiclase con RNAs



^y1 + ^y2 + ^y3 = 1

Clasificar imágenes: coches (y=1), motos (y=2) o bicicletas (y=3)

Podríamos extraer los píxeles de cada imagen y entrenar la red neuronal, para poder realizar predicciones. Para ello, en las funciones de activación usaremos la función soft-max (al ser multiclase). Necesitaremos en la última capa (output layer), un número igual de neuronas al de las clases posibles (C = #clases= 3).

^y1 se corresponde a la probabilidad de que la imagen pertenezca a la clase 1 (coche), y así sucesivamente.

La suma de probabilidades P(1|X)+P(2|X)+P(3|X)=1, por ejemplo, 0.75 + 0.15 + 0.10, es decir, un 75% a que sea coche

Función de activación Softmax

$\hat{y}_1 \rightarrow z_{1(3)} = a_{1(2)}w_{11(3)} + a_{2(2)}w_{21(3)} + a_{3(2)}w_{3(1)} + b_{1(3)}$

$a_{1(3)} = \text{softmax}(z_{1(3)}) =$ #neuronas de la última capa = 3

$= e^{z_{1(3)}} / \sum_{j=1}^{3} e^{z_{j(3)}} = e^{z_{1(3)}} / (e^{z_{1(3)}} + e^{z_{2(3)}} + e^{z_{3(3)}}) = \hat{y}_1$

$\hat{y}_2 = e^{z_{2(3)}} / (e^{z_{1(3)}} + e^{z_{2(3)}} + e^{z_{3(3)}})$

$\hat{y}_3 = e^{z_{3(3)}} / (e^{z_{1(3)}} + e^{z_{2(3)}} + e^{z_{3(3)}})$

núm.real -> Sigmoid -> núm.real

vector -> SoftMax -> vector

$[z_{1(3)}\ z_{2(3)}\ z_{3(3)}]$ $\quad t = [e^{z_{1(3)}}\ e^{z_{2(3)}}\ e^{z_{3(3)}}]$ $\quad \hat{y} = t / e^{z_{1(3)}} + e^{z_{2(3)}} + e^{z_{3(3)}}$ $\quad \hat{y} = [\hat{y}_1\ \hat{y}_2\ \hat{y}_3]$

Función de error para Clasificación Multiclase

$a^{(L)} = \hat{y} = [\ 0.7\ 0.2\ 0.1\ ]$ L=#capas

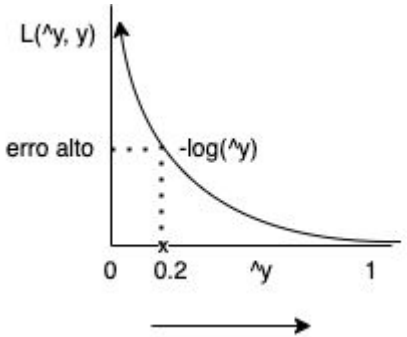
$y = 1 = [\ y_1\ y_2\ y_3\ ] = [\ 1\ 0\ 0\ ]$; $y = 2 = [\ y_1\ y_2\ y_3\ ] = [\ 0\ 1\ 0\ ]$; $y = 3 = [\ y_1\ y_2\ y_3\ ] = [\ 0\ 0\ 1\ ]$

Loss function $L(\hat{y}, y)$

$$L(\hat{y}, y) = -\sum_{j=1}^{3} y_j \log(\hat{y}_j) = -(y_1 \log(\hat{y}_1) + y_2 \log(\hat{y}_2) + y_3 \log(\hat{y}_3))$$

ejemplo para y = 2 e $\hat{y}2 = 0.2$:

$$L(\hat{y}, y) = -\ y_2 \log(\hat{y}_2) = -\log(\hat{y}_2)$$



Entrenamiento de un Perceptrón para clasificador multiclase

En forward propagation: es como hemos visto hasta ahora, con la salvedad que en la última capa usaremos la función softmax en vez de la función sigmoide como función a.

En back propagation (regla de la cadena por la que íbamos multiplicando derivadas para obtener finalmente una ecuación que modifica ese valor de los parámetros en la dirección adecuada): es como hemos visto hasta ahora, y lo único que va a cambiar van a ser las primeras derivadas, ya que hemos modificado tanto la función de error como la función de activación. Como derivamos la función de error respecto a esos parámetros del modelo, pues esa función de error ha cambiado y el resultado de esa derivada pues cambia también.

Es decir, lo que va a cambiar fundamentalmente va a ser esa derivada (parcial):

$dL / da(L) * da(L) / dz(L)$ … (el resto es igual)

Regresión con Redes Neuronales Artificiales

Perceptrón Multicapa: Regresión

- Las Redes Neuronales Profundas pueden utilizarse para predecir un valor continuo
- Para predecir un único valor continuo la output layer debe estar formada por una única neurona
- También pueden utilizarse RNAs para predecir varios valores continuos de manera simultánea
- Para predecir varios valores continuos se requiere una neurona por cada uno de los valores en la output layer

Regresión con Redes Neuronales Artificiales

Perceptrón Multicapa: Regresión

- Las RNAs utilizadas para regresión, en general, **no van a utilizar función de activación en las neuronas de la output layer**, esto permite que salida se encuentre dentro de un rango amplio de valores.
- En determinados casos de uso, si se requiere que los valores se encuentren dentro de un rango acotado, pueden utilizarse algunas funciones de activación (por ejemplo, veremos *relu*)
- La función de error que suele utilizarse es la del **error cuadrático medio**

$$ECM = 1/m \sum (\hat{y}_i - y_i)^2$$