Redes Neuronales

Procesamiento de datos

Facultad de Ingeniería. Universidad del Valle

Febrero de 2019

Contenido

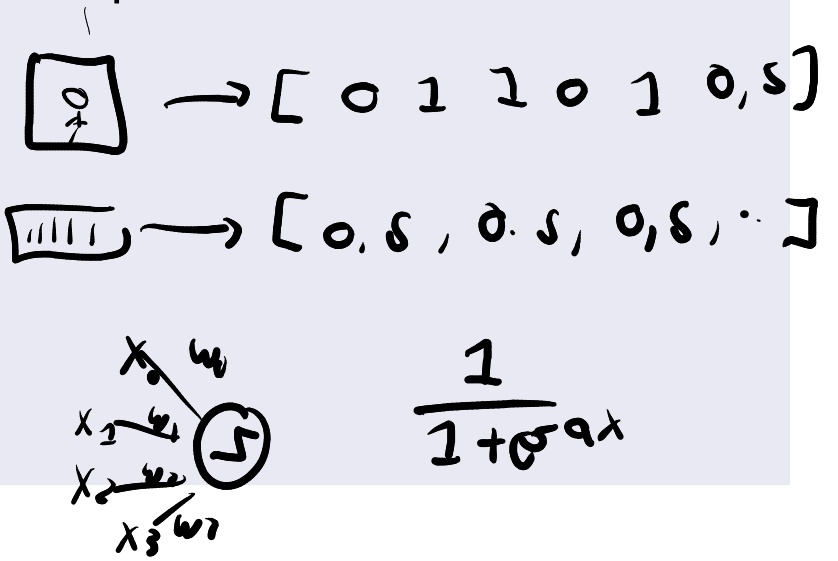
Contenido

Introducción

Aplicaciones de las redes Neuronales

Las redes Neuronales pueden aplicarse a problemas de:

- Reconocimiento de imágenes
- Reconocimiento de voz
- Análisis y filtrado de señales
- Clasificación
- Análisis de textos



Introducción

Modelo de Red Neuronal

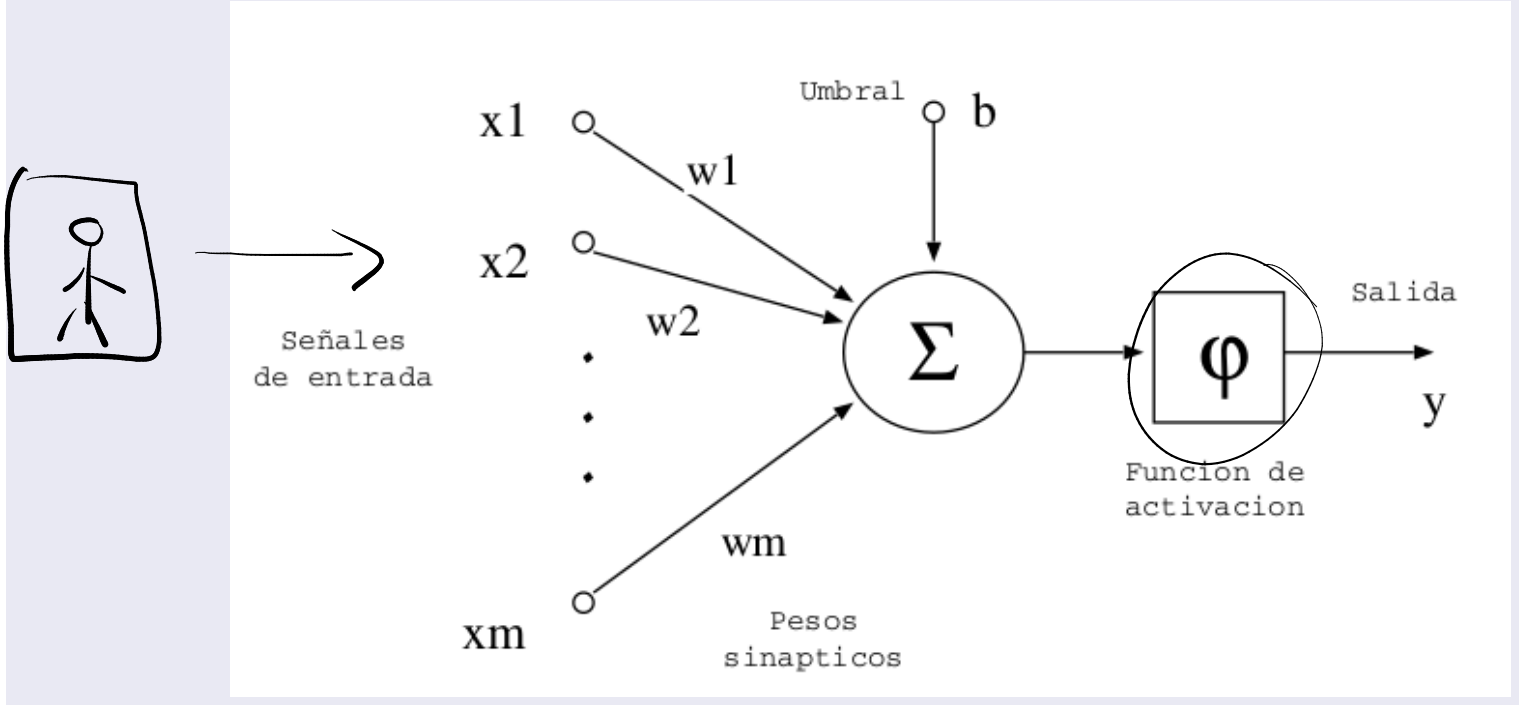


Figura 1: Modelo de red neuronal. Tomado de: [**?**]

Introducción

Modelo de Red Neuronal

La salida de la red neuronal z obedece a:

$$z = \varphi(\sum_{i=1} mw_i x_i + b)$$

En forma vectorial:

$$z = \varphi(wx^T + b)$$

Donde φ es una función conocida como función de activación:

¿Como podemos procesar texto o imágenes aquí? ¿Que debemos considerar?

Introducción

Funciones de activación

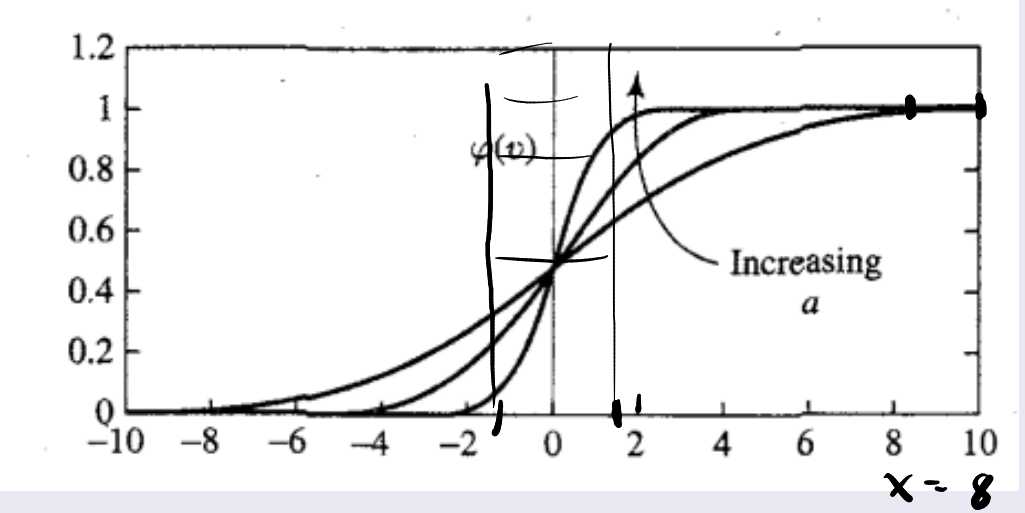


Figura 2: Función sigmoide. Tomado de: [?]

¿Como debemos ingresar los datos a la red Neuronal, para usar apropiadamente las funciones de activación?

Contenido

Consideraciones

Introducción al procesamiento de datos

1. **Reducir el tamaño del espacio de entrada:** Reducir el número de variables de entrada es un objetivo común en el preprocesamiento
2. **Normalización:** Para muchos problemas es necesario ajustar los valores de variables a:
   - Valores de operación de funciones de activación
   - Mejorar la distribución de los datos
3. **Importante:** Estas estrategias se deben aplicar a todos los datos (entrenamiento, prueba o futuros) que ingresen a la red Neuronal.

Consideraciones

Introducción al procesamiento de datos

3. **Mantener la relación:** La transformación de datos es problema de mapeo de datos. Se trata de conservar el significado de las entradas.
4. **Reducción del ruido:** Eliminar datos ruidosos o sin contextos en el problema, ayuda a mejorar la precisión de los algoritmos

Consideraciones

Introducción al procesamiento de datos

Los algoritmos de procesamiento de datos pueden verse de la siguiente forma:

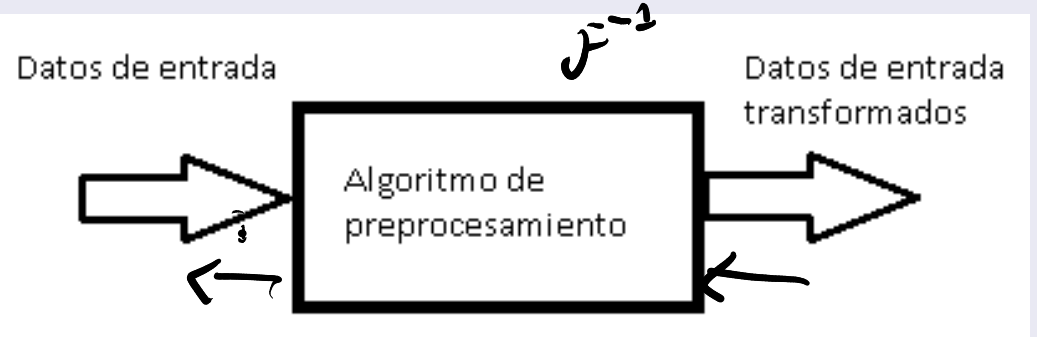


Figura 3: Estrategia de preprocesamiento

Contenido

Estrategias para las entradas

Estandarizar entradas

- Para que las redes neuronales puedan procesar correctamente los datos, debemos asegurarnos que sus entradas estén dentro de los rangos de las funciones de activación.
- Así mismo, se debe asegurar que estos se encuentran apropiadamente distribuidos.
- Se busca garantizar que los diferentes valores de entrada que se puedan presentar, sean reconocidos por la red Neuronal.
- Excluir los valores únicos, ya que estos no presentan patrones, por ejemplo ID.
- Para el manejo de datos, vamos a utilizar las librerias Sklearn y Pandas.

Estrategias para las entradas

Preprocesamiento

Inicialmente, se debe realizar una limpieza de datos no deseados:

- Se deben eliminar los duplicados, para esto contamos con **drop_duplicates** que nos ofrece Pandas.
- Eliminar valores nulos, con esto contamos con **dropna**
- Binarizar, categorizar y discretizar valores continuos. Este proceso depende del problema

Estrategias para las entradas

Eliminar nulos

- Si una columna tiene pocos nulos (menos del 10 %) se puede eliminar las filas sin afectar la calidad de los datos
- En caso contrario, se pueden reemplazar por: moda, promedio u otra estrategia.
- Para llenar con promedio es: **fillna(df['column'].mean(), inplace = True))** o con moda es **fillna(df['column'].mode(), inplace = True)**

Estrategias para las entradas

Outliers

Son datos que pueden alterar el rendimiento de los modelos de predicción, existen tres tipos

- Error: Por ejemplo una edad de 160 años o negativa
- Limites: Escapan del grupo medio perjudicando medidas estadisticas como el promedio
- Punto de interés: Son datos anómalos que queremos detectar como objetivo

Para identificar outliers podemos usar graficas de puntos o histogramas. Una grafica util es el boxplot.

Estrategias para las entradas

Outliers

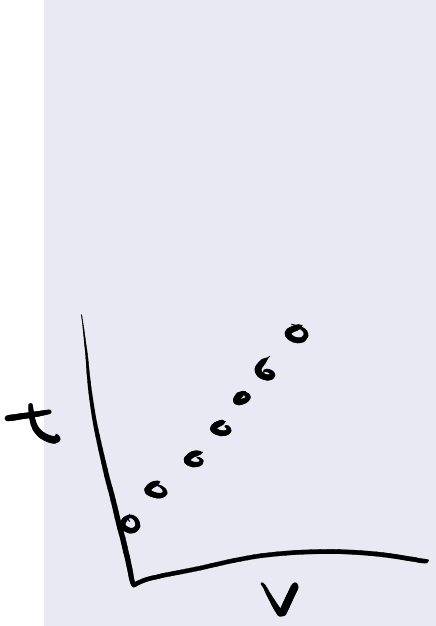
Son datos que pueden alterar el rendimiento de los modelos de predicción, existen tres tipos

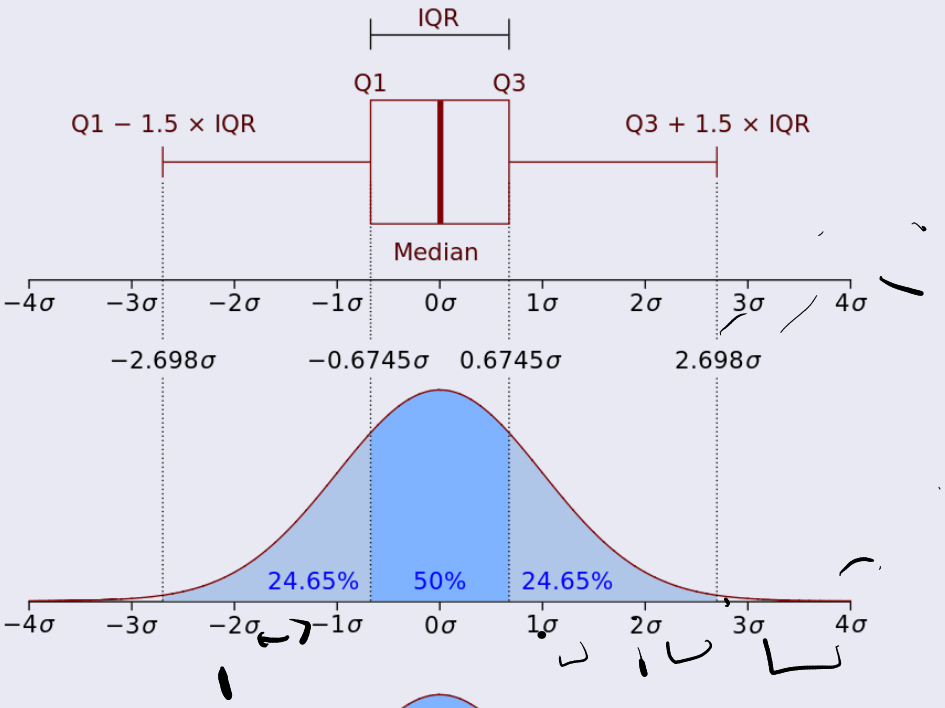
- Error: Por ejemplo una edad de 160 años o negativa
- Limites: Escapan del grupo medio perjudicando medidas estadisticas como el promedio
- Punto de interés: Son datos anómalos que queremos detectar como objetivo

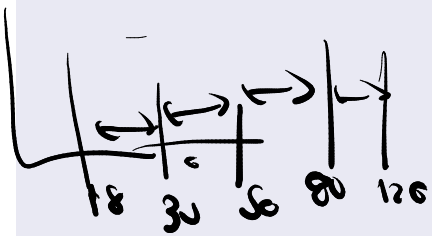
Para identificar outliers podemos usar graficas de puntos o histogramas. Una grafica util es el boxplot.

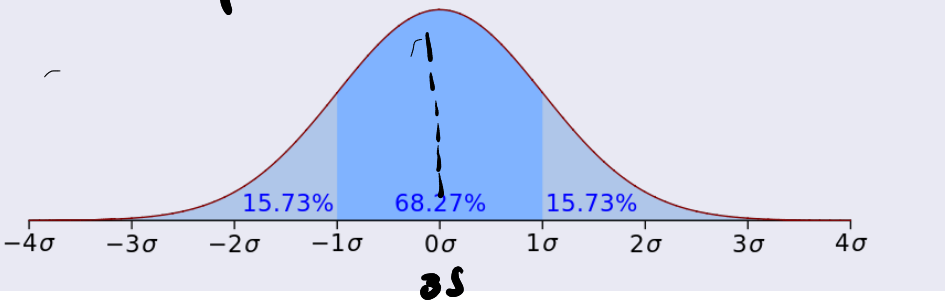
Estrategias para las entradas

Outliers









Estrategias para las entradas

Outliers

Podemos eliminar los datos que están $> 3\,\%$ y mayor > 0.97 en los cuartiles, para esto contamos, aunque esto depende del problema. Para esto contamos con:

- qlow = df.quartile(0.97)
- qhi = df.quartile(0.03)
- df[(df[col] < qhi) & (df[col] > qlow)]

Estrategias para las entradas

Outliers

Se pueden eliminar los datos que están más alla de 3 desviaciones estándar

```
mean = df[col].mean()
sd = df[col].std()
df = df[(df[col] <= mean+(n_std*sd))]
```

Estrategias para las entradas

Categorizar

```
>>edades = [20, 22, 25, 27, 21, 23, 37, 31, 61, 45, 41, 32]
>>rangos = [18, 25, 35, 60, 100]
>>catEdades = pd.cut(edades, rangos)
>>##Tambien se puede pd.cut(edades, num) num = numero
    Natural
>>catEdades
Categorical:
array([(18, 25], (18, 25], (18, 25], (25, 35], (18, 25],
    (18, 25],
(35, 60], (25, 35], (60, 100], (35, 60], (35, 60], (25,
    35]], dtype=object)
```

Estrategias para las entradas

Binarizar

Para esto contamos con **sklearn.preprocessing.Binarizer**

- Se selecciona un valor conocido como limite
- Los valores mayores o iguales se transforman en 1 y los menores en 0
- Debe usarse con cuidado, ya que representa pérdida de información

Estrategias para las entradas

Binarizar

```
>>edades = [20, 22, 25, 27, 21, 23, 37, 31, 61, 45, 41, 32]

>>transformer = Binarizer(threshold=25).fit(edades) #
    Construye la transformación
>>catEdades = transformer.transform(edades) #Aplicar la
    transformación
>>print(catEdades)
 [[0  0  0  1  0  0  1  1  1  1  1  1]]
```

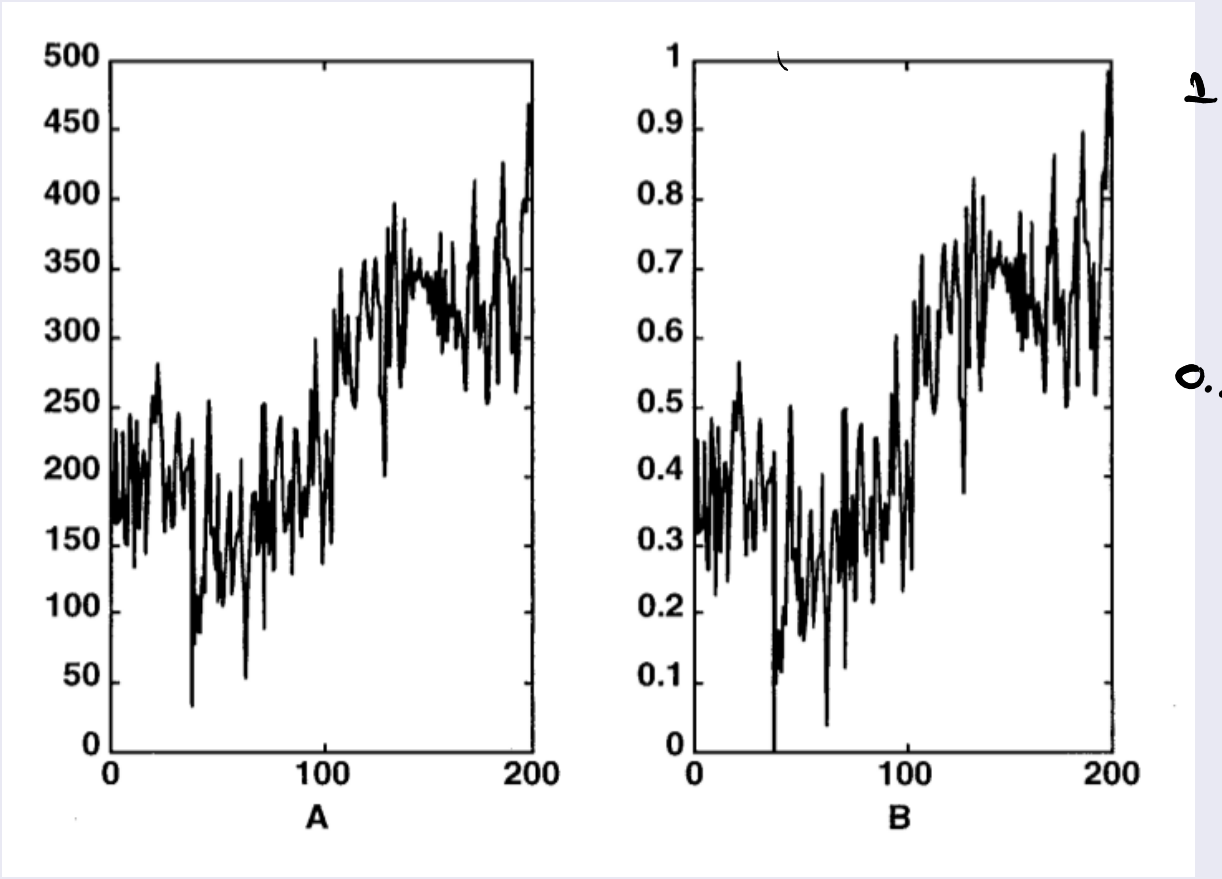
Estrategias para las entradas

Normalización Min-Max

- La normalización es un algoritmo de escalamiento lineal
- Se busca que los datos se escalen y queden en un rango 0-1 o -1 - 1.
- En Sklearn contamos con la clase **sklearn.preprocessing.MinMaxScaler()** para realizar este proceso.

Estrategias para las entradas

Normalización Min-Max



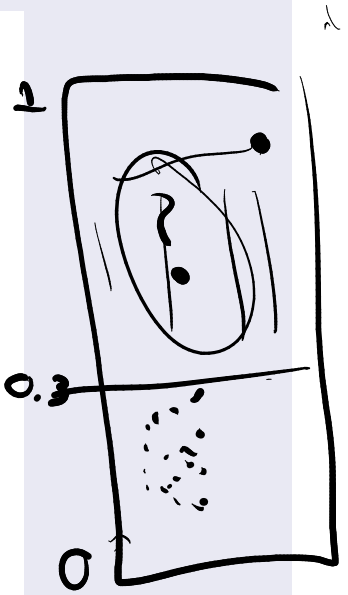


Figura 4: Normalización Min-Max [Li, 2000]

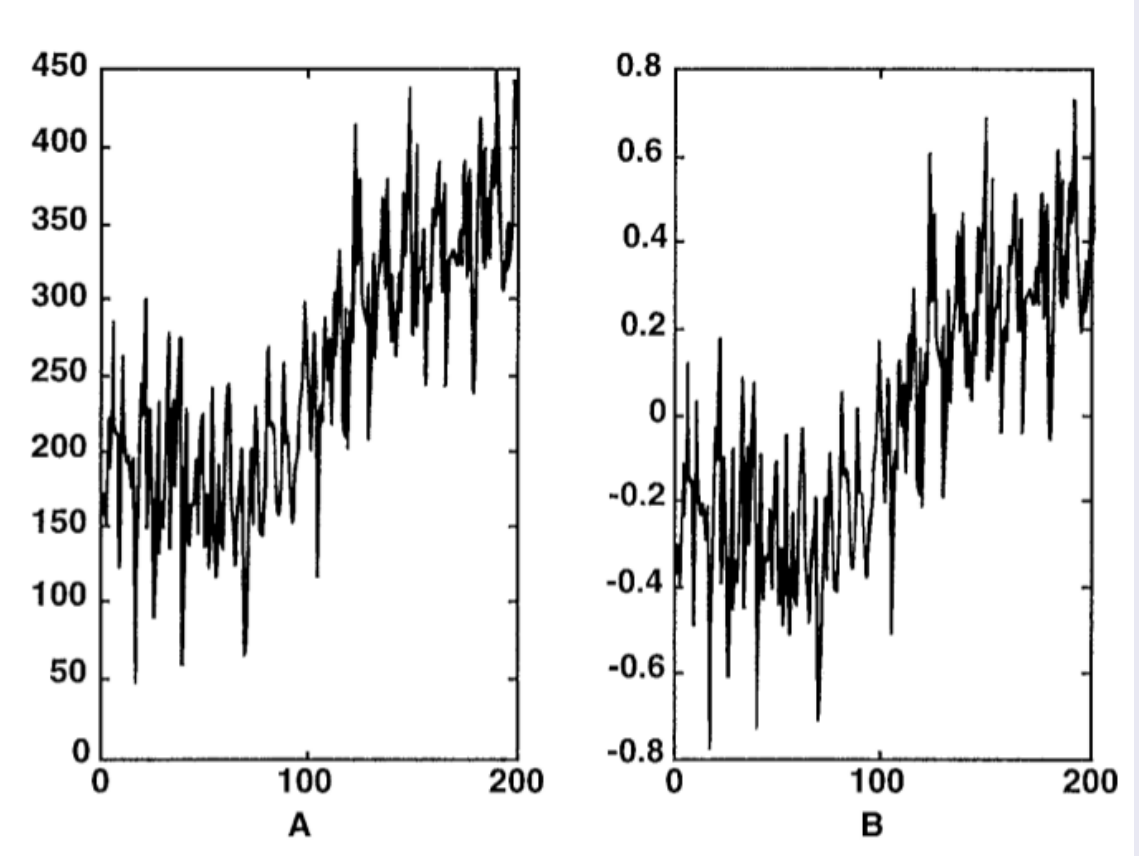
Estrategias para las entradas

Normalización Zscore

- Se busca que los datos correspondan a una distribución normal con media cero y varianza 1.
- Este algoritmo está basado en la desviación estándar de los datos.
- En Sklearn contamos con la clase **sklearn.preprocessing.StandardScaler()** para realizar este proceso.

Estrategias para las entradas

Normalización Zscore



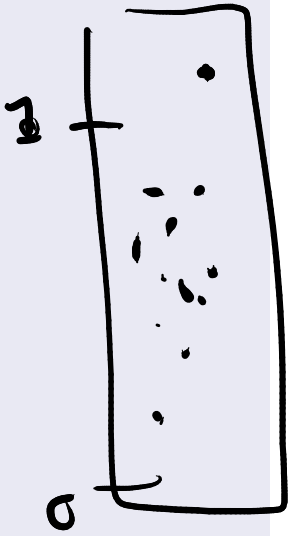


Figura 5: Normalización ZScore [Li, 2000]

Estrategias para las entradas

Reducir el tamaño del espacio de entrada:

- Si el número de datos no es tan grande comparado con el número de variables de entrada, podemos reducir su espacio
- Existen muchas estrategias para el procesamiento de datos, una de ellas es el análisis de componentes principales (PCA)[Li, 2000].

Estrategias para las entradas

Matriz de correlación

- Permite encontrar si las las variables del dataset están realizadas
- La matriz tiene un tamaño *nxn* con valores entre 0 y 1
- Si la posición i, j tiene un valor cercano a 1 las variables i y j se relacionan directamente, en caso contrario se encontrará unvalor de 0
- Eso se realiza con el método corr() en pandas.

Estrategias para las entradas

Análisis de componentes principales

- Es un utilizado para determinar si un espacio m-dimensiones es significante a partir de un espacio n-dimensional.
- En términos prácticos consiste en reducir el número de entradas de m a n.
- Se utiliza una transformación que toma m características de entrada y la transforma en n.
- Estas nuevas variables no están correlacionadas

Estrategias para las entradas

Análisis de componentes principales

- Dado un conjunto de entrenamiento con N características $x = \{x_1, x_2, ..., x_N\}$ y n patrones
- Se puede plantear una transformación $y = \widehat{W}x$ el cual transforma las N características en m minimizando el error cuadrático de reconstrucción. En otras palabras, se minimiza la pérdida de información.
- En Sklearn contamos con la clase **sklearn.decomposition.PCA** para realizar este proceso.

Contenido

Estrategias para las salidas

Introducción

- Hasta el momento hemos manejado clasificadores binarios, por ejemplo, hombre o mujer, si o no.
- Para los clasificadores binarios, basta con tener una neurona de salida, cuya función de salida es un escalón.
- Sin embargo, para muchos problemas tenemos más de dos clases o salidas enteras.

Estrategias para las salidas

Estrategias de procesamiento

- Hasta el momento hemos manejado clasificadores binarios, por ejemplo, hombre o mujer, si o no.
- Para los clasificadores binarios, basta con tener una neurona de salida, cuya función de salida es un escalón.
- Sin embargo, para muchos problemas tenemos más de dos clases o salidas reales.

Estrategias para las salidas

Binarización salidas

- Podemos aplicarla si existe un número finito de clases.
- Consiste en convertir cada clase en una salida binaria, 0 si no se cumple y 1 si se cumple.
- Esto implica un aumento en el número de salidas.
- Ver archivo **4-BinarizarSalidas.py**.

Estrategias para las salidas

Datos continuos

- Se deben categorizar por rangos. Por lo que, debemos conocer los valores mínimo y máximo.
- Una vez esté realizado este paso, se deben binarizar.

Categorías de Texto

- Se deben categorizar al igual que las entradas. Tener presente que debe existir un número finito de clases.
- Posteriormente binarizarlos.

Contenido

Análisis de rendimiento

Introducción

- La precisión no es siempre la mejor medida para calificar que tan buena es una red Neuronal
- Puede suceder que la red sea muy buena prediciendo una clase y otras no. Si hay un gran número de datos de la clase que se predice correctamente, puede ocultarnos la realidad: **La red no está dando buenos resultados para un problema dado**
- Una herramienta que nos permite ver mejor el desempeño es la matriz de confusión

Análisis de rendimiento

Matriz de confusión

- Nos permite estudiar el desempeño de cada una de las clases individualmente
- Permite analizar las situaciones:
  1. **Falsos positivos:** La red indica que es de la clase y realmente no lo es
  2. **Verdaderos positivos:** Es de la clase y realmente lo es
  3. **Verdaderos negativos:** No es de la clase y esto es cierto
  4. **Falsos negativos:** No es de clase, pero realmente si es de la clase

Análisis de rendimiento

Matriz de confusión

Un ejemplo es la clasificación de animales

		Valor predecido			
		Lobo	Zorro	Caballo	Elefante
Valor Real	Lobo	2	1	0	0
	Zorro	3	3	4	2
	Caballo	1	1	1	6
	Elefante	2	0	7	4

Figura 6: Ejemplo matriz de confusión

Análisis de rendimiento

Matriz de confusión

- En Sklearn contamos con la función **sklearn.metrics.confusion_matrix** para generar la matriz de confusión.
- Se puede elegir estudiar con todos los datos disponibles, los de entrenamiento o los de prueba.
- Revisar: **6-MatrizConfusion.py**

Análisis de rendimiento

Curva ROC

- La Curva ROC nos permite comparar el desempeño de dos o mas algoritmos de predicción
- Utiliza los elementos calculados en la matriz de confusión para generar una gráfica
- El desempeño de un algoritmo es el área por debajo de esta cuerva.

Análisis de rendimiento

Curva ROC

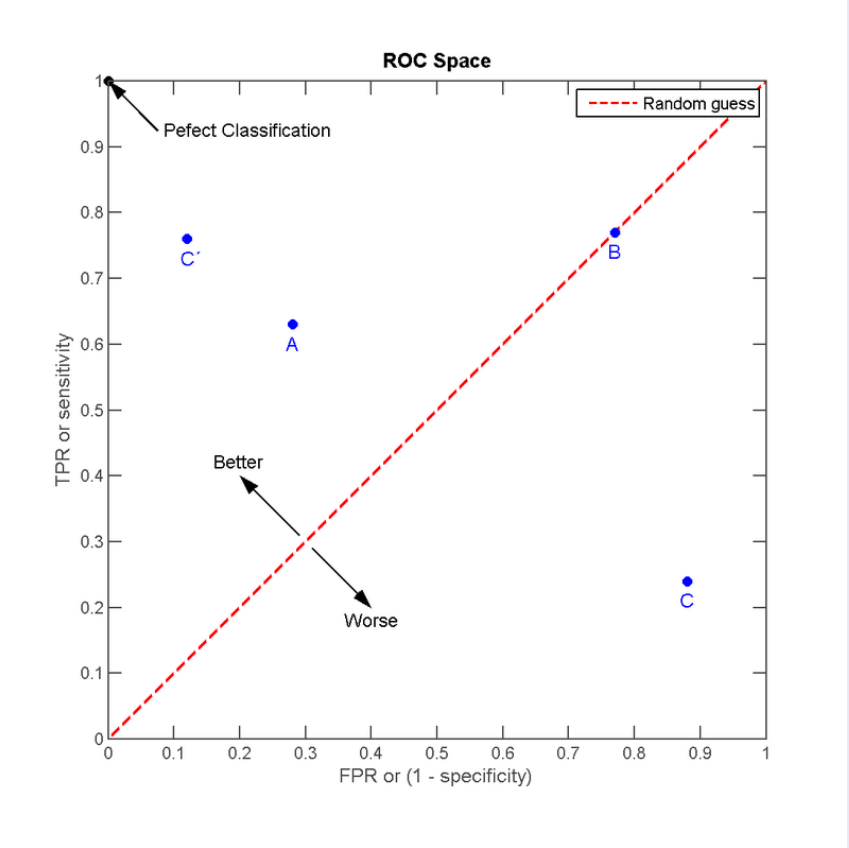


Figura 7: Ejemplo curva ROC: Tomado de Wikipedia

Análisis de rendimiento

Curva ROC

Tomemos un ejemplo sencillo con un clasificador binario para una población $P = 99$:

VP= 40	FP= 15
FN= 18	VN= 26

- $TPR = \frac{VP}{VP+FN} = 0{,}588$ Tasa positiva
- $FPR = \frac{FP}{FP+VN} = 0{,}366$ Tasa negativa
- $ACC = \frac{TP+VN}{P} = 0{,}666$ Precisión

El punto en la curva ROC corresponde a $(TRP, 1 - FPR)$ osea $(0{,}588, 0{,}634)$. La curva se genera a partir de la distribución logística.

Análisis de rendimiento

Curva ROC

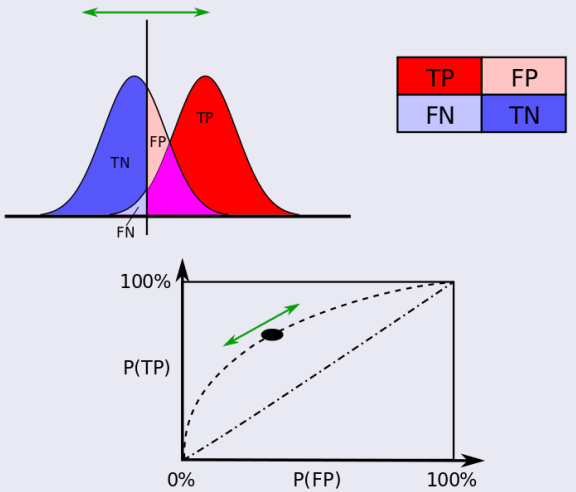


Figura 8: Generación de curva ROC: Tomado de Wikipedia

Referencias I


Curso de modelos computacionales.
`http://www.lcc.uma.es/~munozp/`.
Accessed: Ocubre-2017.


Du, K.-L. and Swamy, M. N. S. (2010).
Neural Networks in a Softcomputing Framework.
Springer Publishing Company, Incorporated, 1st edition.


Heaton, J. (2008).
Introduction to Neural Networks with Java.
Heaton Research.


Li, H. (2000).
Fuzzy Neural Intelligent Systems: Mathematical Foundation and the Applications in Engineering.
CRC Press.

¿Preguntas?

Próximo tema: Redes auto-organizadas.