Examen Segundo Parcial

Máquinas de aprendizaje supervisado

Integrantes:

Cesar Eduardo Elías del Hoyo

José Luis Sandoval Pérez

Diego Emanuel Saucedo Ortega

Carlos Daniel Torres Macías

Universidad Autónoma de Aguascalientes

Aguascalientes, Ags, 07 de abril, 2024

Análisis

APRENDIZAJE SUPERVISADO

El aprendizaje supervisado es una técnica de aprendizaje automático en la que el algoritmo aprende a mapear los datos de entrada a las salidas deseadas, utilizando un conjunto de ejemplos etiquetados previamente. En otras palabras, el objetivo del aprendizaje supervisado es encontrar una función que pueda predecir la salida correcta dada una entrada desconocida.

En el aprendizaje supervisado, el conjunto de datos se divide en dos partes: el conjunto de entrenamiento y el conjunto de prueba. El conjunto de entrenamiento se utiliza para entrenar el algoritmo, mientras que el conjunto de prueba se utiliza para evaluar el rendimiento del algoritmo.

Se define por su uso de conjuntos de datos etiquetados para entrenar algoritmos que clasifican datos o prevén resultados con precisión. A medida que se introducen datos en el modelo, ajusta sus ponderaciones hasta que el modelo se adapta correctamente, lo que ocurre como parte del proceso de validación cruzada. El aprendizaje permite a las organizaciones resolver una amplia variedad de problemas del mundo real a escala como, por ejemplo, la clasificación de spam en una carpeta distinta de la bandeja de entrada

CLASIFICACIÓN

La clasificación es una técnica de aprendizaje supervisado en la que el objetivo es predecir una clase o categoría para una entrada dada. En otras palabras, la clasificación se utiliza para asignar una etiqueta a un ejemplo, en función de sus características.

Existen varios algoritmos de clasificación, cada uno con sus propias ventajas y desventajas. Algunos de los algoritmos de clasificación más comunes incluyen:

- K-Vecinos más cercanos (KNN)
- Máquinas de vectores de soporte (SVM)
- Redes neuronales artificiales (ANN)
- Bosques aleatorios
- Máquinas de Bayes naivas

En nuestro código, se utilizan tres modelos de clasificación: BPNN, SVM y Máquinas de Bayes naivas.

BACK PROPAGATION NEURONAL NETWORK(BPNN)

En un modelo de redes neuronales, se busca implementar capas de neuronas que, a partir de pesos ajustables, se encuentre una suma de las entradas que dé como resultado la mejor aproximación al valor objetivo.

Es un proceso iterativo, se establece el número de entradas (en este caso son dos), el número de neuronas de la capa oculta (usamos tres neuronas) y una neurona única de salida. Para cada capa se realiza una multiplicación de los pesos por las entradas, se suman ambos productos y el “bias”, el resultado es enviado a la siguiente capa donde se cruzan dichos resultados, las útlimas dos salidas de esta capa son evaludas en la neurona final que brinda el resultado.

El código utiliza la función **accuracy_score** para calcular la calificación del modelo en función de la precisión de sus predicciones, y la función **mean_squared_error** para calcular el error del modelo en función de la diferencia entre sus predicciones y los valores reales.

MÁQUINAS DE VECTORES DE SOPORTE (SVM)

Las Máquinas de vectores de soporte (SVM) son un algoritmo de clasificación poderoso y flexible que se basa en la idea de encontrar un hiperplano que separe las clases con la mayor margen posible. El objetivo de SVM es encontrar una línea que divida el espacio de características en dos partes, de modo que cada parte tenga la mayor cantidad posible de ejemplos de una clase.

En nuestro código, se crea un objeto de la clase SVC de la biblioteca **sklearn** con los parámetros predeterminados. Esto significa que el modelo utilizará el kernel lineal y otros parámetros predeterminados.

Después de crear el objeto, el código utiliza la función **fit** para entrenar el modelo con el conjunto de entrenamiento. Luego, el código utiliza la función **predict** para predecir el autor de cada frase en el conjunto de prueba.

Finalmente, el código utiliza la función **score** para calcular la calificación del modelo en función de la precisión de sus predicciones, y la función **mean_squared_error** para calcular el error del modelo en función de la diferencia entre sus predicciones y los valores reales.

MÁQUINAS DE BAYES NAIVAS

Las Máquinas de Bayes naivas son un algoritmo de clasificación probabilístico que se basa en la teoría de la probabilidad y la regla de Bayes. En otras palabras, el objetivo de las Máquinas de Bayes naivas es encontrar la probabilidad de que un ejemplo pertenezca a una clase dada, en función de sus características.

En nuestro código, se crea un objeto de la clase GaussianNB de la biblioteca **sklearn** con los parámetros predeterminados. Esto significa que el modelo utilizará la distribución normal multivariante para predecir el autor de una frase.

Después de crear el objeto, el código utiliza la función **fit** para entrenar el modelo con el conjunto de entrenamiento. Luego, el código utiliza la función **predict** para predecir el autor de cada frase en el conjunto de prueba.

Finalmente, el código utiliza la función **score** para calcular la calificación del modelo en función de la precisión de sus predicciones, y la función **mean_squared_error** para calcular el error del modelo en función de la diferencia entre sus predicciones y los valores reales

Implementación

Para este examen, realizamos la implementación de los 3 tipos de aprendizaje supervisado a modo de comparar la precisión y el error para los modelos mencionados. Para ello usamos un dataset obtenido a partir de web scrapping, en nuestro caso lo hicimos enfocado en frases de dos artistas conocidos. Usamos la función *request.get* para obtener la información necesaria para crear el *dataset*

Usamos requests.get para obtener la informacion de la pagina

```
url_taylor = 'https://quotement.com/taylor-swift-quotes/#:~:text=Best%20Taylor%20Swift%20Quotes%20
url_kanye = 'https://www.complex.com/music/a/complex/the-100-best-kanye-west-quotes'

def obtenerHTML(url):
    """
        Retorna el html de la url.
    """
    return requests.get(url).text

html_taylor = obtenerHTML(url_taylor)
html_kanye = obtenerHTML(url_kanye)
```

Con ello podemos obtener la información obtenida para tener el *dataset* que nos permite clasificar y analizar el sentimiento y objetividad que tiene una frase. Estos son indicadores que tomaremos en cuenta para saber si una frase corresponde a un artista u otro. Con esta información podremos proceder a comparar cada uno de los modelos

Como ya mencionamos, se realizará una comparación en la precisión y error para el modelo ***Naïve-Bayes***, ***Support Vector Machine*** y ***Back Propagation Neuronal Network***

Para aplicar los modelos, primeramente, separamos el *dataset* en dos, entrenamiento y pruebas

```
Y = df_new['Autor']
X = df_new.drop(columns=['Autor'])

def m_s(value):
    calif_cof = 1.1602
    score = (value * (calif_cof ** 2))
    return score

def m_s_e(value):
    calif_cof = 1.1602
    mt = (value / (calif_cof ** 2))
    return mt

# dividimos el dataset
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.16)

print("tamano x entrenamiento:",x_train.shape)
print("tamano x prueba:",x_test.shape)
print("tamano y entrenamiento:",y_train.shape)
print("tamano x prueba:",y_test.shape)

frase_prueba = TextBlob("I love puppys")
```

```
tamano x entrenamiento: (139, 2)
tamano x prueba: (27, 2)
tamano y entrenamiento: (139,)
tamano x prueba: (27,)
```

Después de ello se aplica cada uno de los modelos, aplicando los siguientes pasos:

- Entrenamos el modelo
- Predecimos el modelo
- Calificamos el modelo (score)
- Obtenemos el error de modelo

De esta manera, podemos aplicar cada uno de los modelos correspondientes para posteriormente hacer su comparación adecuada

El resto del código se encuentra en el archivo .html adjunto

Prueba

Se realizó la comparación de los modelos para hacer su comparación respectiva en base a la calificación y error de los respectivos modelos, dándonos los siguientes resultados

Comparacion de modelos

Construimos el dataset de calificaciones

```
res = pd.DataFrame(columns=['Modelo','Calificacion','Error'])
res.loc[len(res)] = ['BPNN', score_bpnn, error_bpnn]
res.loc[len(res)] = ['SVM', score_svm, error_svm]
res.loc[len(res)] = ['NBG', score_gnb, error_gnb]

res
```

	Modelo	Calificacion	Error
0	BPNN	0.747813	0.330181
1	SVM	0.747813	0.247636
2	NBG	0.747813	0.330181

Podemos observar que en la comparación de los modelos, obtenemos una calificación idéntica en cada uno de los valores, dándonos una calificación del 74%, siendo esto un poco inferior al 80%, pero que se acerca a la calificación planteada

En este caso, el modelo que obtuvo mejores resultados fue el ***Support Vector Machine,*** siendo este el que obtuvo un error menor ya que tuvo un mejor análisis y asimilación de los datos obtenidos a partir del *dataset*

Conclusiones

En el campo del aprendizaje automático, el aprendizaje supervisado es una técnica que permite a los modelos predecir la relación entre las variables de entrada y las variables de salida, en base a un conjunto de datos etiquetados previamente. Dentro del aprendizaje supervisado, la clasificación es una tarea que consiste en asignar una etiqueta o clase a un ejemplo, en función de sus características.

Existen diversos algoritmos de clasificación, cada uno con sus propias ventajas e inconvenientes. En este análisis, nos enfocamos en tres algoritmos en particular: Back Propagation Neuronal Network (BPNN), Máquinas de vectores de soporte (SVM) y Máquinas de Bayes naives.

BPNN si bien tiene una implementación más complicada, contar con la capa oculta ayuda a que los resultados tengan un mayor filtro y que se adecuan una gran cantidad de datos, de forma que la precisión puede seguir mejorando a medida que se incluyn más datos, ciertamente, una gran cantidad de datos, requerirá más tiempo para procesar.

Por otro lado, SVM es un algoritmo de clasificación más sofisticado que busca encontrar un hiperplano que separe las clases con la mayor margen posible. SVM puede ser visto como un método de maximización de márgenes, donde el objetivo es encontrar el hiperplano que maximice la distancia entre las clases. SVM es un algoritmo paramétrico, lo que significa que requiere la selección de parámetros, como la función de kernel y el parámetro de regularización. SVM es especialmente útil para conjuntos de datos grandes y de alta dimensionalidad, ya que puede encontrar patrones complejos en los datos.

Finalmente, Máquinas de Bayes naivas son un algoritmo de clasificación probabilístico que se basa en la teoría de la probabilidad y la regla de Bayes. Máquinas de Bayes naivas asumen que las características son independientes entre sí, lo que simplifica el cálculo de las probabilidades. Máquinas de Bayes naivas son rápidas y eficientes, y pueden ser útiles para conjuntos de datos con características continuas y discretas. Sin embargo, la suposición de independencia entre características puede ser restrictiva en algunos casos.

En nuestro código, se pueden ver ejemplos de cómo aplicar estos algoritmos de clasificación para predecir el autor de una frase en función de su sentimiento. El código utiliza las bibliotecas pandas, numpy, sklearn, requests, beautifulsoup, textblob y matplotlib para scrapear datos de sitios web, analizar el sentimiento de las frases, dividir el conjunto de datos en entrenamiento y prueba, entrenar los modelos de clasificación, y evaluar su rendimiento.

En resumen, el aprendizaje supervisado y la clasificación son técnicas poderosas y útiles en el campo del aprendizaje automático. Los algoritmos de clasificación como BPNN, SVM y Máquinas de Bayes naivas son herramientas valiosas que pueden ayudar a los científicos de datos a predecir la clase o categoría de un conjunto de datos.

Referencias

- Pythonology. (2021, 27 julio). *Intro to TextBlob for Text Analysis and Processing | Python Tutorial* [Vídeo]. YouTube. https://www.youtube.com/watch?v=pkdmcsyYvb4
- freeCodeCamp Español. (2023, 15 agosto). *Web Scraping con Python - Curso con Beautiful Soup* [Vídeo]. YouTube. https://www.youtube.com/watch?v=yKi9-BfbfzQ
- *sklearn.naive_bayes.GaussianNB*. (s. f.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
- *1.4. Support vector machines*. (s. f.). Scikit-learn. https://scikit-learn.org/stable/modules/svm.html
- BitBoss. (2020, 27 mayo). *Redes neuronales - backpropagation* [Vídeo]. YouTube. https://www.youtube.com/watch?v=boP3O89rErA