COMPARACION DE MODELOS DE CLASIFICACION DE MACHINE LEARNING

Kevin Pedroza[1]

***Abstract*— Este articulo presenta la comparacion del desempeo 3 tecnicas de clasificación del Aprendizaje de Maquina aplicado sobre un problema de prediccion de resultados de unas medidas de seales obtenidas con un radar. Se explica como fue la metodologia aplicada y la implementacion realizada a traves de la libreria Scikit-learn.**

***Index Terms*— Clasificación, Regresión logística, Máquinas de Soporte Vectorial, Redes Neuronales, Scikit-learn**

I. INTRODUCCION

Dentro de las tareas que el Aprendizaje de Maquina desarrolla actualmente, la clasificación supervisada es una de las más frecuentemente desempeadas por los Sistemas Inteligentes[1]. En este sentido, una gran cantidad de técnicas han sido desarrolladas basadas en Inteligencia Artificial[1] para la solución de dichos problemas de aprendizaje. Un problema de este tipo involucra una serie de parámetros de entrada comunmente llamados ”features”los cuales describen caracteristicas o estados de una objeto, y todos esos elementos descriptivos se resumen en una salida discreta. En este paper, se tratará un problema de clasificación binaria, es decir, existen como salida únicamente 2 clases: la clase positiva y la clase negativa.

El problema que corresponde analizar aquí, como bien se describirá más adelante, son una serie de valores numericos que describen lecturas realizadas por un sensor a la Ionósfera[2] y cuya tarea de clasificación será evaluar el desempeo de un conjunto de descripciones como buena o mala. Como se mencionó anteriormente, existen varias técnicas para desempear esta tarea y en el presente articulo se aplicarán 3 y se medirá el desempeo de cada modelo para así determinar cual de todos es el que mejor se adapta a nuestro conjunto de datos y predice mejor las salidas.

II. CONTEXTO DEL CASO

II-A. Origen del dataset

Para la tarea de clasificación actual, se tomo un dataset que fue recolectado por el Grupo de Física Espacial del Laboratorio de Física Aplicada de la Jhons Hopkins University[2]. Como bien es descrito en el artículo, el sistema recolector consiste en un radar localizado en la Bahía Goose, Labrador en Canadá, el cual tiene una serie de antenas que emiten un patrón de pulsos hacia la Ionosfera1 que luego se reciben para determinar evidencias de algún tipo de estructura en la capa de la atmósfera estudiada.

II-B. Descripción del dataset

La información recolectada por el radar fueron procesadas usando cierta función relacionada con las seales electromagnéticas recibidas que procesaba la información de los pulsos. Cada uno de los 17 pulsos del sistema esta descrito por 2 atributos correspondientes a los valores obtenidos luego de pasar por la función. En este sentido, el dataset consta de 35 columnas, donde la última es la clase de salida que describe si dicha seal es buena (clase positiva) o mala (clase negativa) de acuerdo a los estudios que el Laboratorio quiere hacer.

Fig. 1. Representación de la Ionósfera

III. TÉCNICAS USADAS PARA ALCANZAR EL OBJETIVO

Para el problema de clasificación presente, se quiere determinar cuál modelo o algoritmo predice con mejores resultados los registros del dataset. Los algoritmos usados en el procediento se describen a continuación[3].

- Regresión Logística. La regresión logistica es un algoritmo iterativo el cual determina un modelo lineal de clasificación y que calcula las posibles salidas de un problema usando una función logística.
- Maquinas de Soporte Vectorial (SVM). Es un metodo de aprendizaje supervisado el cual determina la mejor frontera de decision a partir de sus vectores de soporte (puntos del dataset).
- Perceptrón Multicapa. Es un algoritmo de Redes Neuronales en el cual la función de aprendizaje pasa por varias capas con neuronas que se entrenan hasta obtener una hipotesis.

Usando los algoritmos anteriores y partiendo de las métricas obtenidas, se determinará entonces el modelo con los mejores resultados de predicción.

IV. METODOLOGíA E IMPLEMENTACION

Cada uno de los algoritmos descritos tiene cierta complejidad que hace que su implementación manual sea compleja, por esto, para realizar tarea de clasificación se hizo uso de una libreria de Python llamada Scikit-learn[3] cuyo desarrollo se hizo orientado a la aplicación de las técnicas de Aprendizaje de maquina. Scikit-learn integra todo un repertorio de métodos que resultan fáciles de implementar y prácticos a la hora de desempear tareas como clasificación, regresión, etc. con las técnicas usadas actualmente.

IV-A. Descripción del procedimiento

El procedimiento llevado a cabo fue el siguiente: En primer lugar, se lee el dataset y se prepara para cada algoritmo haciendo uso de la función $data\_scaler()$ la cual normaliza los datos, realiza las divisones en conjuntos de entrenamiento y conjuntos de pruebas (entrenamiento: 57 %, pruebas: 43 %) asegurandose de estratificar los datos. Luego se hace el llamado de una función para cada algoritmo el cual aplica la técnica y arroja las métricas que nos ayudarán a determinar el desempeo de cada método.

Para cada uno de los procesos se realizó la Regularización y la Validación cruzada. En primer lugar, en cada función se evaluan ciertos parámetros de penalización del algoritmo, y luego se busca valida el que tuvo mejor desempeo según los recursos que Scikit-learn proporciona. A continuación, se observan ciertas lineas del script implementado.

- División de los features y salidas en conjuntos de entrenamiento y prueba.

```
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.43,
random_state=0, stratify=y)
scaler = StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

- Regularización de cada método. Para el método de Regresión Lógística y SVM, el parametro de regularización es 'C', mientras que para el Perceptrón Multicapa es el parámetro 'alpha'.

```
clf_list = [LogisticRegression(C=i,
solver="liblinear",
max_iter=1000).fit(X_train, y_train)
for i in c]
clf_list = [SVC(C=i, kernel='linear',
random_state=0).fit(X_train, y_train)
for i in param]
clf_list = [MLPClassifier(hidden_layer_sizes=layers,
alpha=i, activation='relu',
solver='lbfgs', random_state=0).fit(X_train,
y_train) for i in param]
```

- Validación del mejor score. Usando la funcion $cross\_val\_score()$, Scikit arroja una medida de desempeo que permite evaluar a los modelos probados.

```
score = [cross_val_score(i, X_train,
y_train, cv=10).mean() for i in
clf_list]
```

- Cálculo de las métricas. Scikit provee de funciones que arrojan las métricas de cada modelo. Aquí se reportan 5: F-1 score, Recall, Accuracy, Precision y Specificity.

```
f1 = m.f1_score(y_test, model)
recall = m.recall_score(y_test, model)
accuracy = m.accuracy_score(y_test,
model)
precision = m.precision_score(y_test,
model)
tn, fp, fn, tp =
m.confusion_matrix(y_test,
model).ravel()
specificity = tn/float(tn+fp)
```

MATRIZ DE CONFUSIÓN PARA EL MODELO SVM CON KERNEL RBF (DATOS NORMALIZADOS)			
		Negativo	Positivo
Valores Actuales	Negativo	50	4
	Positivo	10	87

Fig. 2. Matriz de Confusió del modelo SVM kernel RBF datos normalizados

MATRIZ DE CONFUSIÓN PARA EL MODELO SVM CON KERNEL RBF			
		Negativo	Positivo
Valores Actuales	Negativo	45	9
	Positivo	4	93

Fig. 3. Matriz de Confusió del modelo SVM kernel RBF

V. RESULTADOS DEL PROCEDIMIENTO Y ANÁLISIS

Una vez hecho el proceso, se obtuvieron las métricas de todos los modelos, se muestra a continuación el modelo con el mejor desempeo y algunos otros cercanos.

El modelo que tuvo las mejores métricas fue el SVM con kernel Gaussiano o RBF con datos normalizados y con un parametro de regularizacion de 0.1. A continuación se muestran las métricas de dicho modelo:

1. F1-score: 0.9255319148936171
2. Recall: 0.8969072164948454
3. Accuracy: 0.9072847682119205
4. Precision: 0.9560439560439561
5. Specificity: 0.9259259259259259

El modelo presenta unas metricas aceptables porque todas estan cercanas al 90 % de desempeo. Si observamos las métricas del mismo modelo pero sin datos normalizados, se observa que tiene un desempeo similar.

Y cuyas metricas son:

1. F1-score: 0.9346733668341709

MATRIZ DE CONFUSIÓN PARA EL MODELO SVM CON KERNEL POLY			
		Negativo	Positivo
Valores Actuales	Negativo	28	26
	Positivo	0	97

Fig. 4. Matriz de Confusió del modelo SVM kernel POLY

2. Recall: 0.9587628865979382
3. Accuracy: 0.9139072847682119
4. Precision: 0.9117647058823529
5. Specificity: 0.8333333333333334

Observamos que ambos modelos se acercan considerablemente a un buen desempeo de predicción de los valores de prueba.

Por otro lado, el modelo que peor se comportó fue el SVM con kernel Poly de grado 2. Se observan a continuacion sus métricas y matriz de confusion.

1. F1-score: 0.8818181818181818
2. Recall: 1.0
3. Accuracy: 0.8278145695364238
4. Precision: 0.7886178861788617
5. Specificity: 0.5185185185185185

VI. CONCLUSIONES

Al haber realizado el proceso de prueba de 3 de las técnicas de Aprendizaje de maquina para problemas de clasificación sobre el conjunto de datos del artículo y luego de analizar los resultados obtenidos dados por las métricas de todos lo modelos, se obtuvo que aquel que mejor se acercaba a las predicciones y que en general tuvo mejor desempeo fue el modelo de Máquinas de Soporte Vectorial con el kernel RBF y cuyos datos fueron escalados. Se observó además, a comparar con algunos otros modelos, que aún con las métricas obtenidas, se varios falsos positivos y falsos negativos del modelo, lo que le aade cierto error no tan ajustable.

REFERENCES

[1] Maglogiannis, I.G., Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies. Frontiers in artificial intelligence and applications. 2007. Available on: https://books.google.com.co/books?id=vLiTXDHr_sYC

[2] V. G. Sigillito, S. P. Wing, L. V. Hutton aand K. B. Baker. Classification of radar returns from the ionosphere using neural networks. Vol. 10. 1989.

[3] Pedregosa, F. and Varoquaux, G. and others. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. Vol. 12. 2011.