Tarea Práctica 2
Aprendizaje de Máquinas - MA5204

Daniel Carmona G. [*1] **Matías Jara** [*2] **Juan Pablo Cabeza** [*3]

Abstract

En este informe de la Tarea Práctica 2, estudiaremos un algoritmo para el aprendizaje supervisado de clasificadores binarios, llamado perceptron. El objetivo de este problema es extender el algoritmo para clasificación binaria, a uno de K-clases utilizando el enfoque One versus All, utilizando el dataset *sklearn handwritten digist* y comparar con Support Vector Classifier (SVC) implementado en sklearn.svm.

Por otra parte, considerando la base de datos de *Breast Cancer*, estudiaremos el comportamiento de los SVC en distintas situaciones. Para ello, utilizaremos los algoritmos de Regresión Logit, SVC y Linear Discriminant Analysis (LDA).

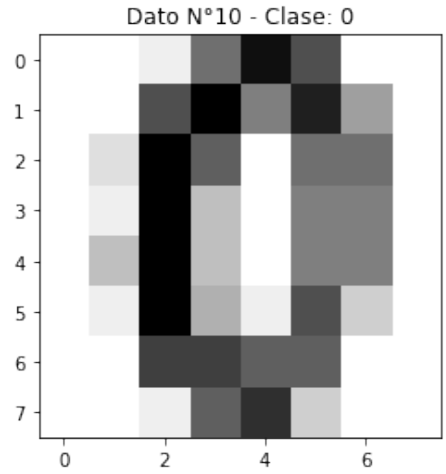

Figure 1. Representación Gráfica Elemento N°10.

1. Handwritten Digits Classification

1.1. Exploración de Datos

Se importa el dataset *sklearn handwritten digits* en las variables X,y las cuales contienen las características y respectivas clases del dataset. El dataset cuenta con 1796 elementos (o dígitos) y 10 clases, las cuales corresponden a los números del 0 al 9.

Una vez conocidas las dimensiones del dataset y que describe la información en cada variable (X,y), se implementa una función que recibe el elemento que se desea representar, y lo imprime mediante un mapa de grises junto a la clase a la cual corresponde, tal como se muestra en la Figura 1.

Por ultimo, se dividen los conjuntos X, y en conjuntos de entrenamiento y prueba, con las proporciones de 80% y 20% respectivamente.

1.2. Clase Perceptron

Para la implementación del algoritmo Perceptron de K-Clases con enfoque *One versus All* se implementó primero una clase llamada Perceptron, la cual recibe como parámetro el dígito en específico que se desea clasificar, el número de iteraciones, tasa de aprendizaje y por ultimo la tolerancia que se desea utilizar en el entrenamiento.

Para esto el algoritmo, al utilizar la función *fit* asigna pesos aleatorios a las señales de entrada y las combina como una entrada neta. Las entradas ponderadas o netas representan el hecho de que una señal de entrada diferente puede tener una fuerza diferente. Luego se pasa por una función de activación, la cual procesa el peso neto y lo emite como señal binaria (0 o 1). Esta señal se compara con la clase actual de la entrada, y se actualizan los pesos de acuerdo a la función $w_i \leftarrow w_i + \cup w_i$, la cual depende de la tasa de aprendizaje.

Para extender el algoritmo para una clasificación binario, se crearon 10 instancias de la clase perceptron creada anteriormente, donde a cada uno se le entregó un dígito (o clase) en especifico. Luego, cada clase fue entrenada utilizando el conjunto de entrenamiento, pero dependiendo de la clase asociada al perceptron, se reemplazaron los valores del conjunto *y_train* por 1 y -1. Por ejemplo, para entrenar un perceptron que diferencie la clase 0 por sobre las demás, al conjunto de clases de entrenamiento, se

reemplazaron los 0's por 1's, y el resto de clases (1-9) se reemplazaron por -1. De esta manera se obtienen 10 perceptrones entrenados para predecir una clase por sobre el resto.

1.3. Otra Forma de Implementación Perceptron Multiclase

Otra forma que se pudo haber abordado para la implementación de un clasificador multiclase mediante el perceptron de clasificación binaria, podría haber sido utilizando una extensión de *One vs One*, donde hubiese sido necesario crear distintos subconjuntos de entrenamiento solamente con 2 clases, para todas las combinaciones posibles entre las 10 clases del modelo, por lo que se tendrían 45 clasificadores binarios $(K(K-1)/2)$, donde K representa las clases, lo cual es bastante más complejo.

1.4. Predicción de las Clases para los Datos de Prueba

Luego, se definió una función que realiza la predicción del conjunto de prueba con cada perceptron entrenado anteriormente, y se calculó el $\theta^T \phi(x)$ de cada elemento para cada perceptron, donde el mayor de estos valores nos indica a cual clase corresponde el elemento. De esta manera, se obtuvo la matriz de confusión de la Figura 2 donde se obtuvo un accuracy del 76.39% para un número de 100 iteraciones y una tasa de aprendizaje de 0.01.

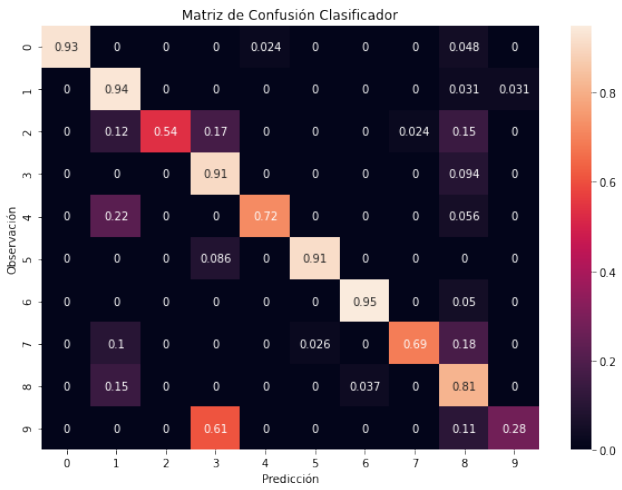

Figure 2. Matriz de Confusión Perceptron One vs All.

De la matriz de confusión del Perceptron podemos ver que predice correctamente para la mayoría de los dígitos. Sin embargo, para la clase 9, acierta el 28% de los casos y confunde con el número 3 el 61%, y con el 8 el 11% restante, debido a la similitud de estos números en varios casos. Esto se podría solucionar variando la cantidad iteraciones y encontrando una tolerancia que permita disminuir el *Loss* en el entrenamiento del perceptron.

1.5. Comparación Perceptron - SVC

Para la implementación del clasificador SVC, se utilizó la función *gridsearch* de la librería *scikit learn*, para encontrar los mejores hiperparámetros para el conjunto de datos utilizado. A partir de esta grilla se obtuvo que la mejor opción es utilizar un clasificador de kernel *RBF*, del cual se obtuvo un accuracy del 100%, es decir que este clasificador es capaz de predecir correctamente todas los dígitos del conjunto de prueba. Adicionalmente, se implementó un clasificador SVC de kernel lineal, donde se obtuvo un accuracy del 98.61%, valor que se encuentra muy por sobre el valor obtenido por el perceptron.

La gran diferencia de precisión entre ambos tipos de algoritmos, es que el SVC busca encontrar hiperplanos del espacio de caraterísticas que separe a las clases. Además de separar las clases, el hiperplano debe maximizar el margen geometrico (maximal margin hyperplane), el cual corresponde a encontrar la mínima distancia entre los puntos del conjunto y el hiperplano.

Table 1. Comparación de Accuracy por Algoritmo.

ALGORITMO	ACCURACY
PERCEPTRON	76.39%
SVC - LINEAL	98.61%
SVC - RBF	100%

2. Breast Cancer Prediction

El objetivo de este problema es estudiar el comportamiento de los SVC en distintas situaciones. Para esto, se considera la base de datos de Breast Cancer de la librería sklearn.datasets para implementar 3 algoritmos distintos.

2.1. Exploración de Datos

Primero se crea un dataframe de pandas a partir del dataset anteriormente mencionado. Luego, se utilizó la función size para obtener las dimensiones del dataset, donde se obtuvo que este cuenta con 31 columnas y 17639 elementos.

Para obtener la cantidad de tumores malignos y benignos se utilizó la función *value_counts()*, obtieniendo las proporciones de la Tabla 2.

Table 2. Cantidad de Tumores Malignos y Benignos.

TUMOR	CANTIDAD
MALIGNO	357
BENIGNO	212

2.2. Logit, SVC y LDA.

Primero se creo una instancia del modelo SVC utilizando un kernel lineal. Luego, se entrenó el modelo mediante la función *fit()* del modelo, para luego predecir las clases de los elementos del conjunto de prueba (función *predict(x_test)*). Para obtener el accuracy normalizado correspondiente al clasificador, se utilizó la función *accuracy_score*, la cual compara las predicciones obtenidas por el clasificador con las clases reales.

Por ultimo, los coeficientes asociados a las regresiones encontrados por el algoritmo SVC se obtienen mediante la función *coef_* del modelo.

El procedimiento para los modelos Logit y LDA es análogo, con la diferencia de que los clasificadores se crean instanciando sus respectivas regresiones (*LogisticRegression()* y *LinearDiscriminantAnalysis()*).

Los Accuracies, coeficientes y respectivos de intersección de cada modelo se encuentran resumidos en la Tabla 3.

Table 3. Comparación Algoritmos SVC.

REGRESIÓN	ACCURACY	INTERSECCIÓN	N° COEF
SVC	95.61%	7.77	30
LOGIT	96.49%	0.20	30
LDA	95.61%	49.30	30

Observemos que SVC con un kernel lineal, funciona bien con datos no estructurados (como texto e imágenes, en nuestro caso), mientras que la regresión logística funciona con variables independientes ya identificadas. Por otra parte, si bien SVC se centra en los puntos "difíciles" de clasificar (vectores de soporte), LDA se centra en todos los datos, ya que esta es una transformación lineal que maximiza la separabilidad.

2.3. Selección de Características

Para realizar la selección de características, primero se graficó la matriz de correlación de las características del dataset de la Figura 3, donde se encuentra que las correlaciones más altas (valores cercanos a 1) se encuentran para las variables relacionadas al radio, perímetro y área del tumor. Es por esto, que se decidió conservar todas las variables asociadas al radio, es decir *mean radius*, *radius error* y *worst radius*, ya que es posible calcular variables como el perímetro y área a partir de esta variable.

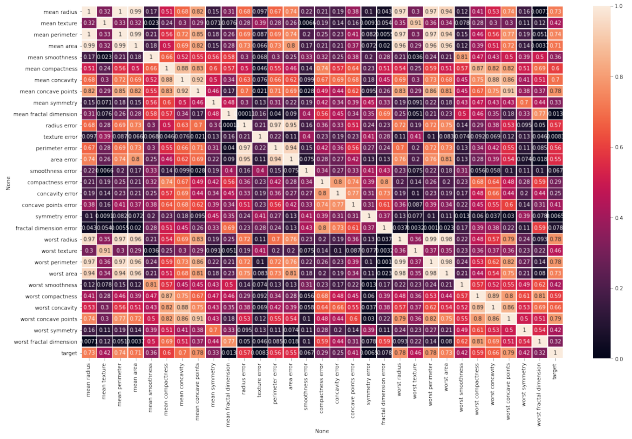

Figure 3. Matriz de Correlación.

Una vez que se quitaron las variables asociadas al perímetro y el área del tumor, se aplico un StandarScaler para estandarizar los valores de las características del dataset. Luego, se repitió el procedimiento utilizado anteriormente pero con los datos normalizados, obteniendo los accuracies, coeficientes y punto intersección de la Tabla 4, donde se logra observar que al normalizar los datos la precisión del algoritmo *SVC* se mantiene constante, la del algoritmo *Logit* incremente alrededor de un 1% y la del *LDA* decrece aproximadamente un 2%.

Table 4. Comparación Algoritmos SVC Normalizados.

REGRESIÓN	ACCURACY	INTERSECCIÓN	N° COEF
SVC	95.61%	0.16	24
LOGIT	97.37%	0.75	24
LDA	93.86%	2.19	24

De las Tablas 3 y 4 podemos ver que al normalizar los datos los coeficientes asociados a las características de cada modelo también decrecen, ya que se encuentra todos los datos en las mismas proporciones, por lo que la intersección de las regresiones también decrece.