TEC | Tecnológico de Costa Rica

Escuela de Ingeniería en Electrónica

Licenciatura en Ingeniería Electrónica

EL5852 Aprendizaje Automático

Tarea 5

Clasificadores SVM, RDF, kNN y ANN

Diego Bogarín Picado

diegobp11@gmail.com

2018160264

Oscar Andrés Rojas Fonseca

osaf2412@hotmail.com

2018102187

Cartago, Costa Rica

2 de mayo, 2023

Índice

1. Conceptos

1.1. Matriz de confusión

La matriz de confusión es una herramienta para visualizar el rendimiento de un modelo de clasificación. Esta matriz muestra la cantidad de verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos que el modelo ha producido para cada clase. Dicha matriz puede ser utilizada para el cálculo de las métricas de precisión y exhaustividad [1]

1.2. Precisión y Exhaustividad

La precisión es la proporción de ejemplos que el modelo ha clasificado correctamente para una clase determinada. Es decir, es la relación entre el número de verdaderos positivos y la suma de verdaderos positivos y falsos positivos. [1]

La exhaustividad, también conocida como sensibilidad o recall, es la proporción de ejemplos que el modelo ha identificado correctamente para una clase determinada con relación al número total de ejemplos de esa clase. Es decir, es la relación entre el número de verdaderos positivos y la suma de verdaderos positivos y falsos negativos. [1]

1.3. Frente de Pareto

Se refiere a un sistema de graficación de de múltiples objetivos, donde el punto principal es optimizar el desempeño de esos objetivos de manera simultánea. Así, se generan las gráficas del frente de Pareto, donde los valores que presenten un desempeño mayor en ambos objetivos, en este caso precisión y exhaustividad, corresponden a los valores dominantes u optimos del desarrollo del sistema [2].

2. Análisis de resultados

Para la elaboración de la tarea se procedió a generar modelos entrenados de cada método clasificador; máquinas de soporte vectorial (SVM), bosques de decisión aleatorios (RDF), k vecinos más cercanos (kNN), y redes neuronales artificiales (ANN) usando los parámetros definidos por defecto en cada caso, al cual se le generó la matriz de confusión correspondiente.

Acto seguido se corrió la función *Mejores_parametros()* encargada de realizar barridos de series de parámetros con sus respectivos entrenamientos, a los cuales se les configuró una partición de los datos en cinco secciones ($cv = 5$).

El siguiente paso lo realizó la función *proceso_pareto()*, encargada de seleccionar los parámetros óptimos para cada método clasificador entre las opciones presentadas y gráficar los puntos de precisión contra

exhaustividad (Frente de Pareto). Así, se seleccionaron los mejores casos para ser configurados en un nuevo entrenamiento del modelo, al cual también se le graficó su matriz de confusión correspondiente.

2.1. Clasificador SVM

El clasificador SVM, con sus parámetros por defecto, entrenó en primera instancia para generar un primer modelo con la matriz de confusión presente en la Figura 1.

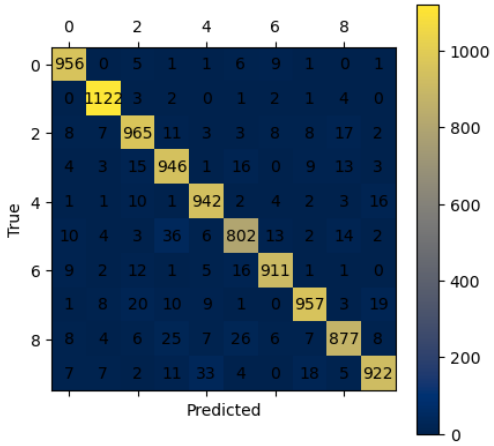

Figura 1: Matriz de confusión SVM sin optimizar

2.1.1. Elección de hiperparámetros

Para el svm se basó en el ejemplo dado por el profesor, sin embargo, se eliminaron unos parámetros de prueba por cuestiones de tiempo y recursos de procesamiento. Se vario el C el cual representa la medida de regularización en SVM, el gamma se refiere a la influencia de un solo ejemplo de entrenamiento y el kenel indica la forma en que se separaran los datos [3].A la hora de correr los entrenamientos para diferentes parámetros con la función *Mejores_parametros()*, se definió la siguiente lista de combinaciones:

$$params = [\,'kernel' : ['rbf'],\,'gamma' : [5e-3, 1e-3],\,'C' : [1, 10, 100],\,'kernel' : ['linear'],\,'C' : [1, 10, 100]]$$

Obteniendo así la gráfica de la Figura 2 con la combinación optima por el Frente de Pareto remarcada en color rojo.

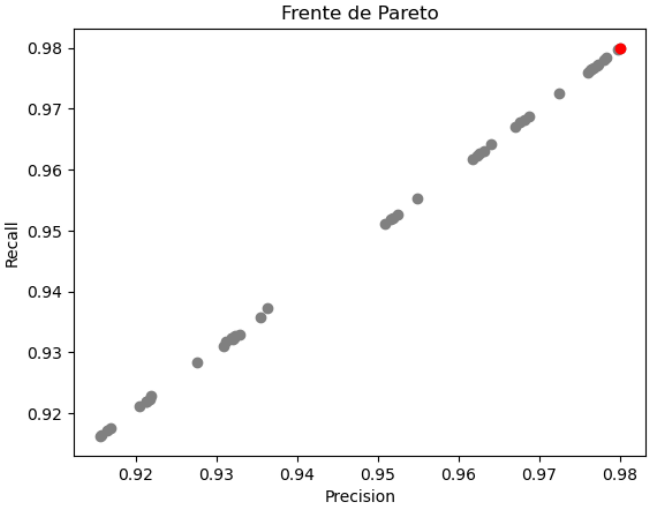

Figura 2: Frente de Pareto para hiperparámetros SVM.

Los parámetros óptimos seleccionados fueron: $\{'C' : 10, 'gamma' : 0{,}005, 'kernel' : 'rbf'\}$, referentes a una configuración del Kernel en RBF y los parámetros $C = 10$ y $Gamma = 0{,}005$. De manera que estos últimos se configuraron en el siguiente entrenamiento del modelo, el cual arrojó la matriz de confusión de la Figura 3.

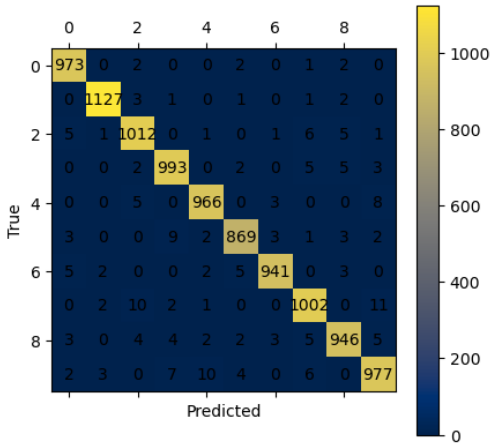

Figura 3: Matriz de confusión SVM optimizado

2.2. Clasificador RDF

Entrenando el modelo RDF con su configuración por defecto, el resultado fue la matriz de confusión en la Figura 4.

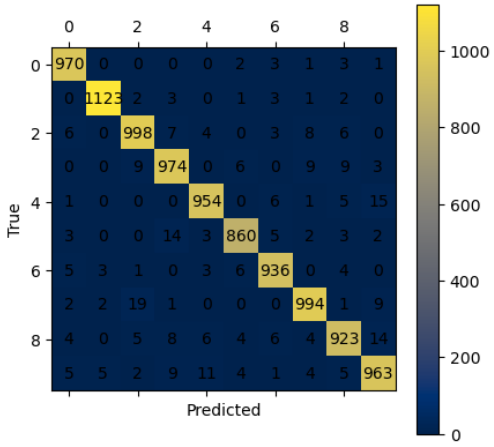

Figura 4: Matriz de confusión RDF sin optimizar

2.2.1. Elección de hiperparámetros

Para la elección de hyperparametros se cambió la cantidad de estimadores y el criterion la cual indica como mide la calidad de división en el árbol de decisión. [5]A la función *Mejores_parametros()* se le envió la siguiente lista de combinaciones:

$$params = ['criterion' : ['gini'],' n_estimators' : [50, 200, 500],'criterion' : ['entropy'],' n_estimators' : [50, 200, 500]]$$

Donde sus resultados de precisión contra exhaustividad fueron graficados en la Figura 5.

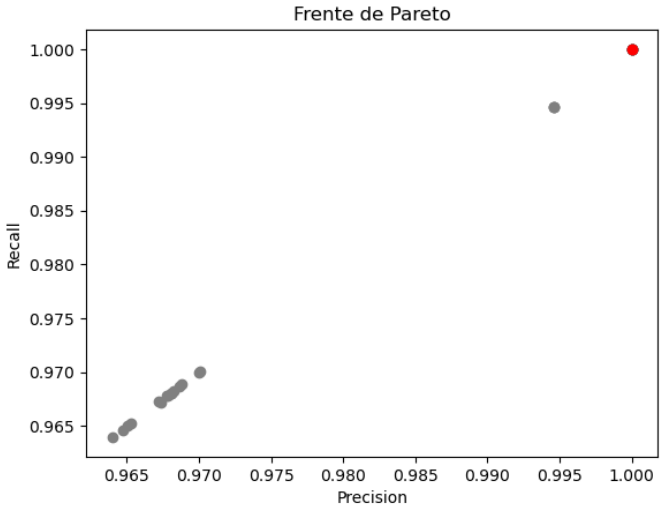

Figura 5: Frente de Pareto para hiperparámetros RDF.

Los parámetros óptimos obtenidos fueron: $\{'criterion' : 'entropy', 'n_estimators' : 50\}$, lo cual quiere decir que se utilizó el criterio de entropía para medir la calidad de las divisiones en los árboles de decisión en un total de 50 árboles. De manera que se entrenó nuevamente al modelo y su matriz de confusión se presenta en la Figura 6

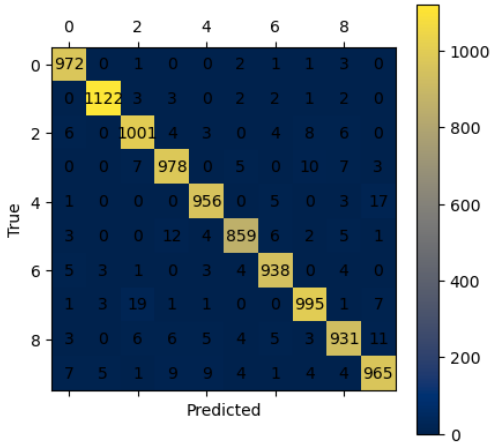

Figura 6: Matriz de confusión RDF optimizado

2.3. Clasificador KNN

Para el clasificador KNN el n_neighbors especifica el número de vecinos más cercanos que se considerarán, el parámetro weights especifica la función de peso que se utiliza para la predicción y el parámetro que indica la distancia utilizada para el cálculo de los vecinos más cercanos. Entrenando el modelo kNN con su configuración por defecto, el resultado fue la matriz de confusión en la Figura 7.

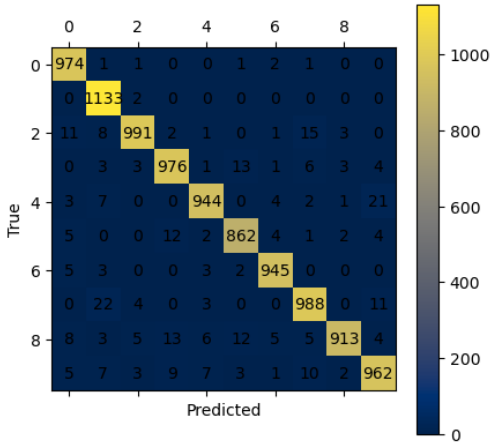

Figura 7: Matriz de confusión kNN sin optimizar

2.3.1. Elección de hiperparámetros

A la función *Mejores_parametros()* se le envió la siguiente lista de combinaciones:

$$params = \{'n\_neighbors' : [3, 5, 10], 'weights' : ['uniform', 'distance'], 'p' : [1, 2]\}$$

Donde sus resultados de precisión contra exhaustividad fueron graficados en la Figura 8.

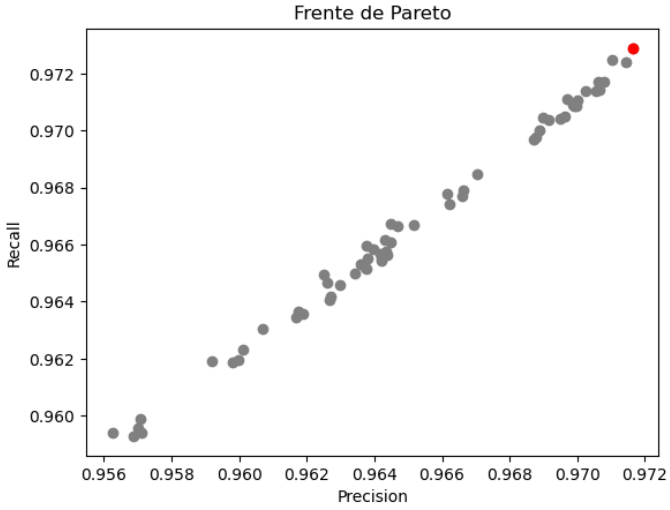

Figura 8: Frente de Pareto para hiperparámetros kNN.

Los parámetros óptimos obtenidos fueron: $\{'n\_neighbors' : 3, 'p' : 2, 'weights' : 'distance'\}$, correspondientes a un número de 3 vecinos más cercanos que se consideran para la predicción, la distancia de Minkowski con Un valor de 2 en la distancia euclidiana y la función de peso utilizada en la predicción utilizando distance, lo que significa que los vecinos más cercanos tienen un peso mayor en la predicción que los vecinos más lejanos.

Al configurar los parámetros anteriores, se volvió a entrenar un modelo de clasificación kNN y se obtuvo su matriz de confusión con la forma mostrada en la Figura 9

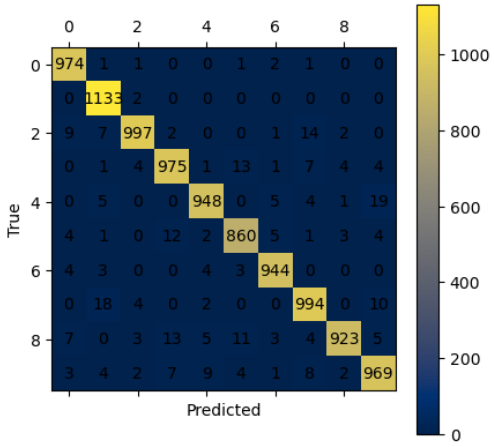

Figura 9: Matriz de confusión kNN optimizado

2.4. Clasificador ANN

Entrenando el modelo ANN con su configuración por defecto, el resultado fue la matriz de confusión en la Figura 10.

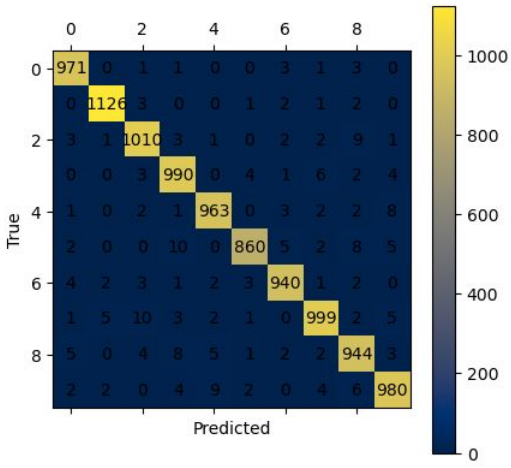

Figura 10: Matriz de confusión ANN sin optimizar

2.4.1. Elección de hiperparámetros

En el ANN se pueden escoger los números de las capas ocultas, las funciones de activación a utilizar el tipo de learning rate, las iteraciones máximas, el tipo de optimización entre otras. Cabe destacar que se podrían haber hecho mas pruebas sin embargo este es el método que mas tarda por lo que no se logró hacer muchas.A la función *Mejores_parametros()* se le envió la siguiente lista de combinaciones:

$$params = \{'hidden\_layer\_sizes' : [(50,), (100,)], 'activation' : ['logistic', 'tanh',],$$
$$'solver' : ['sgd'], 'learning\_rate' : ['adaptive'], 'max\_iter' : [100, 1000]\}$$

Donde sus resultados de precisión contra exhaustividad fueron graficados en la Figura 11.

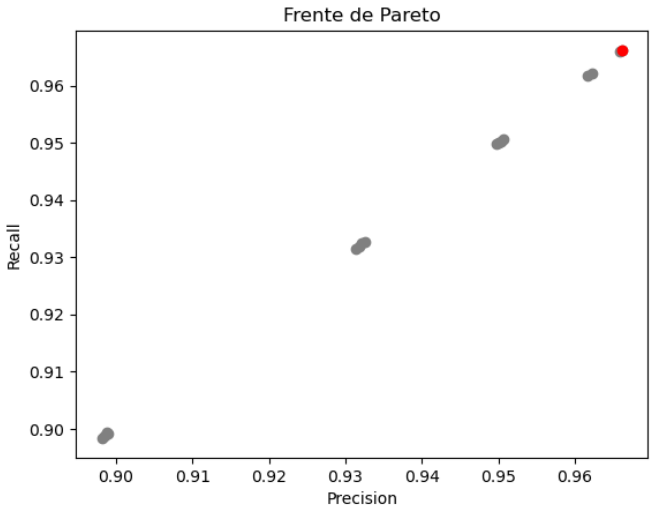

Figura 11: Frente de Pareto para hiperparámetros ANN.

Los parámetros óptimos obtenidos fueron: $\{'activation' :' tanh', 'hidden\_layer\_sizes' : (100,), 'learning\_rate' :' adaptive', 'max\_iter' : 1000, 'solver' :' sgd'\}$.
Para estos parámetros es posible apreciar que en la red neuronal en cuestión, la función de activación utilizada es 'tanh', que significa tangente hiperbólica. Se tiene una tupla que indica la cantidad de neuronas en cada capa oculta, en este caso con una sola capa oculta con 100 neuronas. La tasa de aprendizaje utilizada en el algoritmo de optimización es 'adaptive', lo que significa que se adaptará automáticamente a la situación de los datos. Un número máximo de iteraciones permitidas para el algoritmo de optimización de 1000. Y por último, el algoritmo de optimización utilizado para entrenar fue 'sgd'.

Con la configuración establecida, se generó un nuevo modelo y su matriz de confusión correspondiente mostrada en la Figura 12.

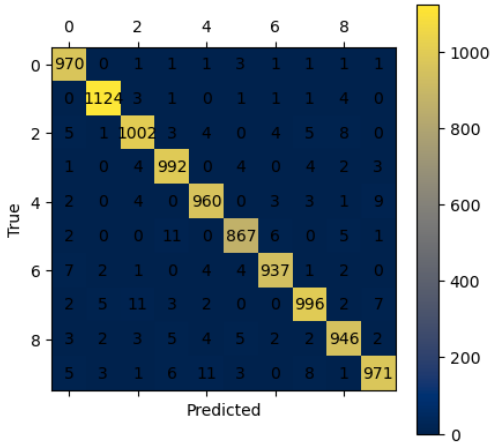

Figura 12: Matriz de confusión ANN optimizado

2.5. Predicción del digito dibujado

Para la prueba de predicción de los digitos dibujados en la ventana proporcionada por el cuaderno *predict_digit.ipynb*, se realizaron secuencias de dibujos del 0 al 9 en cuatro ocaciones, de manera que la suma total de aciertos corresponda y determine el modelo más eficiente. Los resultados se muestran en la tabla 1.

Tabla 1: Predicciones por método para cada secuencia realizada.

Método	**Exitos en la predicción**				
	Secuencia 1	Secuencia 2	Secuencia 3	Secuencia 4	Total
SVM	5	5	5	9	24
RDF	8	3	6	8	25
kNN	4	5	4	6	19
ANN	8	8	9	10	35

Se puede observar que el que tuvo un mejor desempeño fue el ann según los resultados obtenidos, pero también cabe destacar que este fue el método que más tardaba en su entrenamiento por lo que fue al que se le pudo hacer menos pruebas de hyperparametros por lo que podría ser optimizado aún más.
Estos son ejemplos de los numeros dibujados para las pruebas.

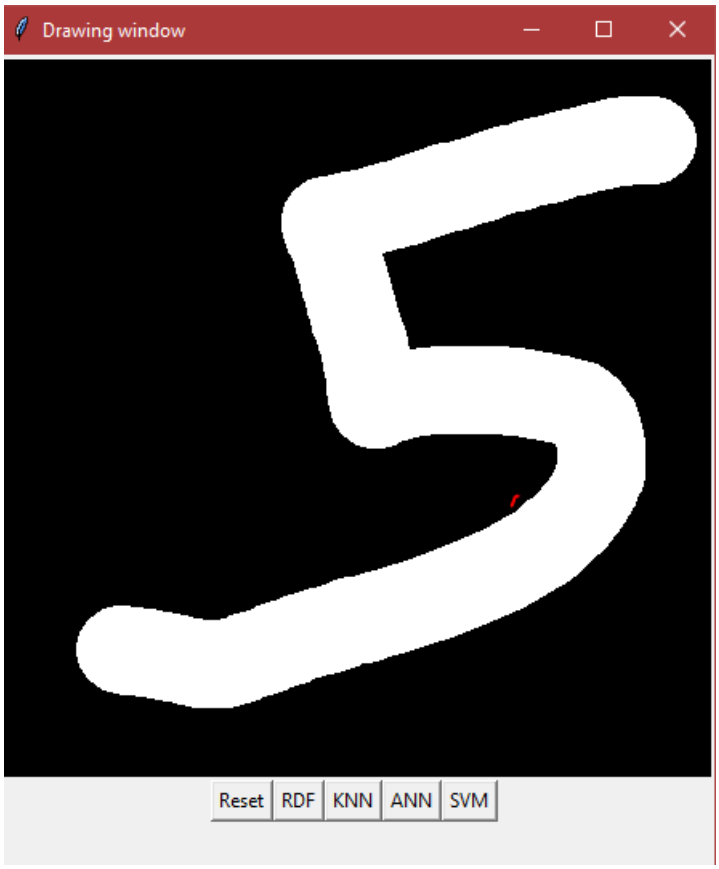

Figura 13: Ejemplo de número 5

Figura 14: Ejemplo de número 7

A la hora de realizar las pruebas se pudo observar que dependiendo del numero estos tienen su nivel de complejidad, por ejemplo, los números 7, 5 y 9 cuestan mas que los reconozcan, esto es lo que mas diferencia que tan bueno es un modelo de otro, por ejemplo, en las figuras 13 y 14 solo lo pudo clasificar correctamente el método ann. Ya que los otros necesitan que se dibujen de manera mas parecida a los datos de entrenamiento mientras que el ann es un poco más flexible.

Cabe destacar que para la prueba con dígitos se uso para el SVM el clasificador sin optimizar, esto debido a que a pesar de que mejoraba la en cuando a precisión y exhaustividad, a la hora de hacer las pruebas con los dígitos dibujados tiraba como predicción únicamente 2, esto puede haberse dado a que el método se hizo muy preciso, pero poco exacto por lo que se decidió correr las pruebas con el clasificador no optimizado.

Bibliografía

[1] A. Géron. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow,". O'Reilly Media. 2019.

[2] P. Alvarado-Moya. "Tarea 5: Clasificadores SVM, RDF, kNN, ANN". Aprendizaje Automático IS2023 ITCR. 2023.

[3] "Tuning the hyper-parameters of an estimator", scikit-learn. [En línea]. Disponible en: https://scikit-learn.org/stable/modules/grid_search.html. [Consultado: 04-may-2023].

[4] Machinelearningmastery.com. [En línea]. Disponible en: https://machinelearningmastery.com/hyperparameters-for-classification-machine-learning-algorithms/#:~':text=alpha'%3A%201.0%7D-,K%2DNearest%20Neighbors%20(KNN),perhaps%20just%20the%20odd%20numbers.text=It%20may%20also%20be [Consultado: 04-may-2023].

[5] W. Koehrsen, "Hyperparameter tuning the random forest in python", Towards Data Science, 10-ene-2018. [En línea]. Disponible en: https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74. [Consultado: 04-may-2023].