

Instituto Tecnológico de Costa Rica

Tarea 4

Introducción al reconocimiento de patrones

Profesor: José Pablo Alvarado

Integrantes: Sebastián Vargas Zúñiga

Alonso Vega Badilla

I Semestre

Entrega: 04/06/2020

1. Scikit Learn y SVM

Para esta primera parte se utilizan diferentes tipos de SVC variando el kernel utilizado y modificando los parámetros y los C's que estos SVC utilizan, se busca obtener las diferentes matrices de confusión con el fin de visualizar los diferentes parámetros derivados de esta matriz con el fin de ver que tan bien trabajan.

1.1 Kernel lineal

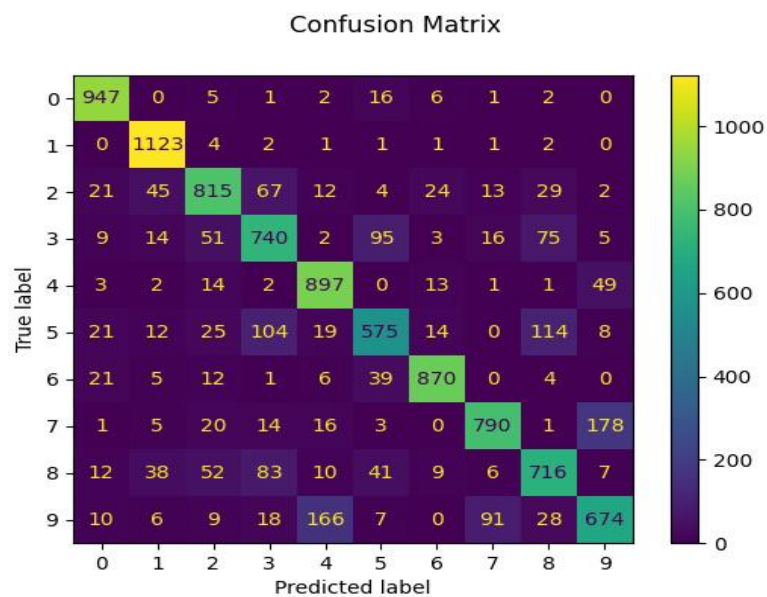


Fig 1. Matriz de confusión para Kernel lineal con C=1.

```
Classification report for classifier SVC(C=1, kernel='linear', max_iter=500):
```

	precision	recall	f1-score	support
0	0.91	0.97	0.94	980
1	0.90	0.99	0.94	1135
2	0.81	0.79	0.80	1032
3	0.72	0.73	0.72	1010
4	0.79	0.91	0.85	982
5	0.74	0.64	0.69	892
6	0.93	0.91	0.92	958
7	0.86	0.77	0.81	1028
8	0.74	0.74	0.74	974
9	0.73	0.67	0.70	1009
accuracy			0.81	10000
macro avg	0.81	0.81	0.81	10000
weighted avg	0.81	0.81	0.81	10000

Fig 2. Parámetros de evaluación para Kernel lineal con C=1.

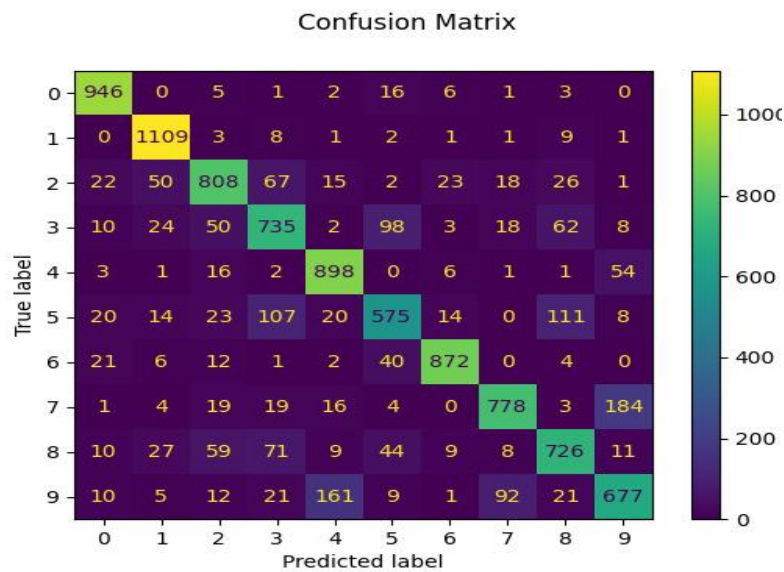


Fig 3. Matriz de confusión para Kernel lineal con C=10.

Classification report for classifier SVC(C=10, kernel='linear', max_iter=500):

	precision	recall	f1-score	support
0	0.91	0.97	0.94	980
1	0.89	0.98	0.93	1135
2	0.80	0.78	0.79	1032
3	0.71	0.73	0.72	1010
4	0.80	0.91	0.85	982
5	0.73	0.64	0.68	892
6	0.93	0.91	0.92	958
7	0.85	0.76	0.80	1028
8	0.75	0.75	0.75	974
9	0.72	0.67	0.69	1009
accuracy			0.81	10000
macro avg	0.81	0.81	0.81	10000
weighted avg	0.81	0.81	0.81	10000

Fig 4. Parámetros de evaluación para Kernel lineal con C=10.

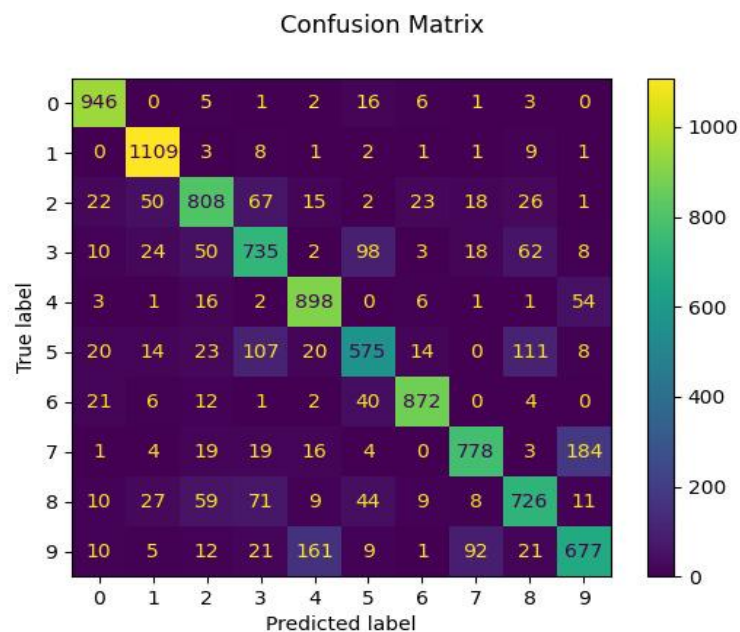


Fig 5. Matriz de confusión para Kernel lineal con C=20.

```

classification report for classifier SVC(C=20, kernel='linear', max_iter=500):

```

	precision	recall	f1-score	support
0	0.91	0.97	0.94	980
1	0.89	0.98	0.93	1135
2	0.80	0.78	0.79	1032
3	0.71	0.73	0.72	1010
4	0.80	0.91	0.85	982
5	0.73	0.64	0.68	892
6	0.93	0.91	0.92	958
7	0.85	0.76	0.80	1028
8	0.75	0.75	0.75	974
9	0.72	0.67	0.69	1009
accuracy			0.81	10000
macro avg	0.81	0.81	0.81	10000
weighted avg	0.81	0.81	0.81	10000

Fig 6. Parámetros de evaluación para Kernel lineal con C=20.

1.2 Kernel radial

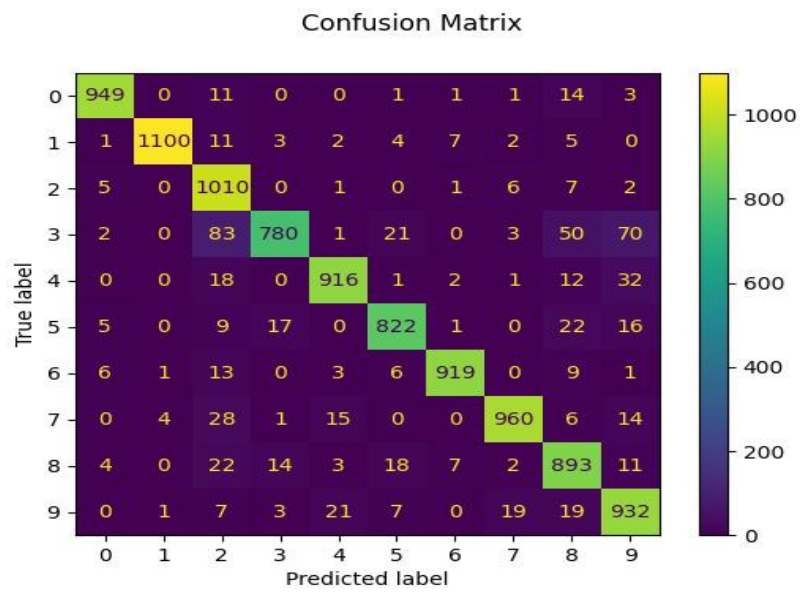


Fig 7. Matriz de confusión para Kernel lineal con C=1 y gamma=0.7.

```

Classification report for classifier SVC(C=1, gamma=0.7, max_iter=500):

```

	precision	recall	f1-score	support
0	0.98	0.97	0.97	980
1	0.99	0.97	0.98	1135
2	0.83	0.98	0.90	1032
3	0.95	0.77	0.85	1010
4	0.95	0.93	0.94	982
5	0.93	0.92	0.93	892
6	0.98	0.96	0.97	958
7	0.97	0.93	0.95	1028
8	0.86	0.92	0.89	974
9	0.86	0.92	0.89	1009
accuracy			0.93	10000
macro avg	0.93	0.93	0.93	10000
weighted avg	0.93	0.93	0.93	10000

Fig 8. Parámetros de evaluación para Kernel radial con C=1 y gamma=0.7.

1.3 Kernel polinomial

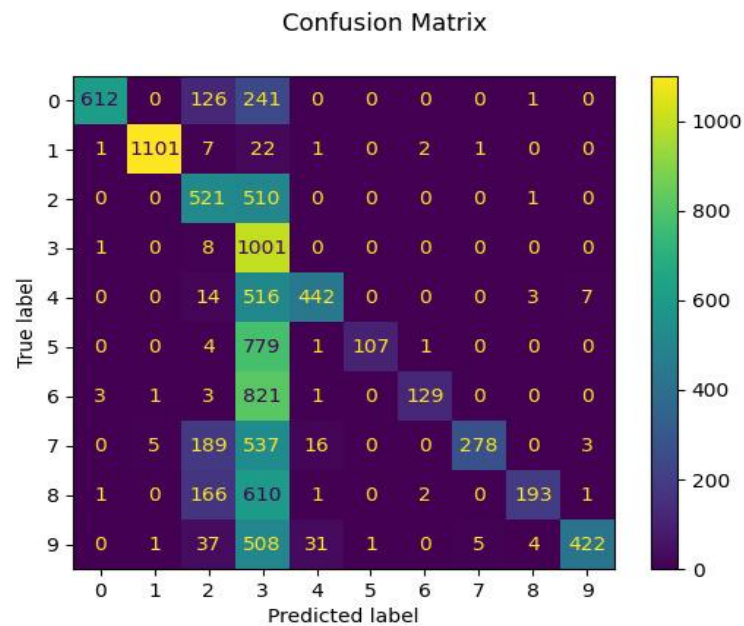


Fig 7. Matriz de confusión para Kernel polinomial con C=10 y gamma=0.7.

Classification report for classifier SVC(C=10, gamma=0.7, max_iter=500):

	precision	recall	f1-score	support
0	0.99	0.62	0.77	980
1	0.99	0.97	0.98	1135
2	0.48	0.50	0.49	1032
3	0.18	0.99	0.31	1010
4	0.90	0.45	0.60	982
5	0.99	0.12	0.21	892
6	0.96	0.13	0.24	958
7	0.98	0.27	0.42	1028
8	0.96	0.20	0.33	974
9	0.97	0.42	0.59	1009
accuracy			0.48	10000
macro avg	0.84	0.47	0.49	10000
weighted avg	0.84	0.48	0.50	10000

Fig 8. Parámetros de evaluación para Kernel polinomial con C=10 y gamma=0.7.

2. Keras y Deep Learning

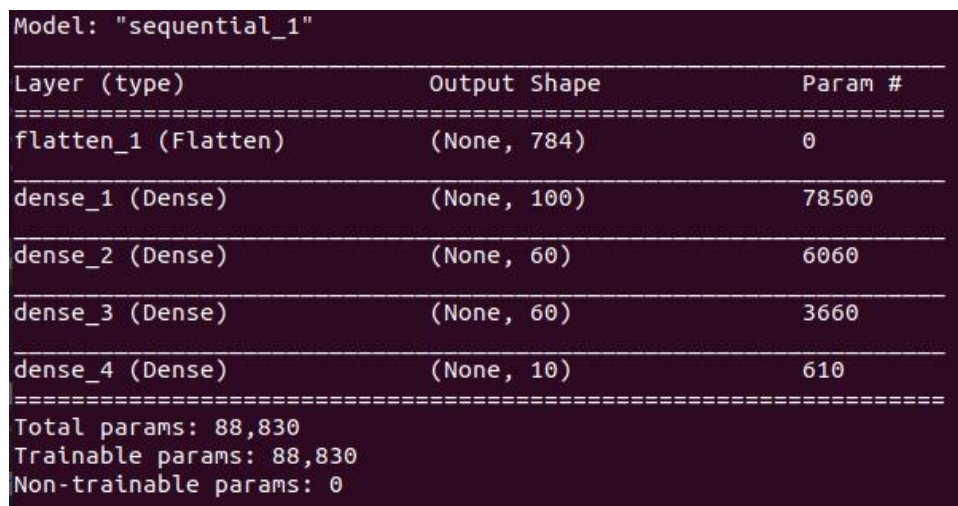
El modelo secuencial de red neuronal consiste en varias capas, donde la última capa tiene una salida de clasificación softmax. Este modelo se implementó con el siguiente código:

```
model = Sequential()
model.add(Flatten(input_shape=(28,28)))
model.add(Dense(100, activation='relu'))
model.add(Dense(60, activation='relu'))
model.add(Dense(60, activation='relu'))
```



```
model.add(Dense(10, activation='softmax'))  
model.summary()
```

Se escogió este modelo de red neuronal debido a que se probaron varias arquitecturas y esta dio buenos resultados a la hora establecer los parámetros óptimos con los datos de entrada propuestos.



Layer (type)	Output Shape	Param #
flatten_1 (Flatten)	(None, 784)	0
dense_1 (Dense)	(None, 100)	78500
dense_2 (Dense)	(None, 60)	6060
dense_3 (Dense)	(None, 60)	3660
dense_4 (Dense)	(None, 10)	610
Total params: 88,830		
Trainable params: 88,830		
Non-trainable params: 0		

Fig 10. Arquitectura de red neuronal con keras.

Notese que se cuenta con un total de 88,830 parámetros distribuidos entre las diferentes capas de la red neural, esta cantidad es bastante grande sin embargo fueron los parámetros necesarios para lograr una adecuada estimación.

Se procedió a implementar la red neuronal con un set de entrenamiento y otro de validación, como consecuencia se obtuvo entonces un error alrededor del 0.02 para el set de entrenamiento mientras que para la validación fue alrededor de 0.09, así mismo se obtuvo la precisión donde se obtuvo un 99% de precisión para el set de entrenamiento y un 97.5% para el de validación, con lo cual se verifica el buen funcionamiento de la red neuronal para reconocer los dígitos nuevos ingresados a la red. Este comportamiento se puede observar en la siguiente imagen donde se muestra la evolución de la precesión y del error para los dos sets en función de las épocas.

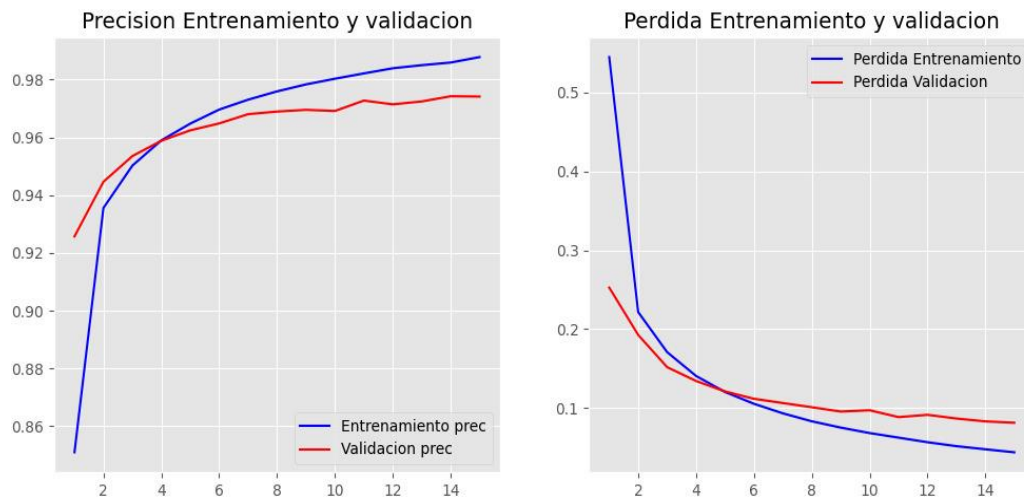


Fig 11. Gráficas de error y precisión para los sets de entrenamiento y validación.

Para evaluar el adecuado desempeño de una red neuronal es necesario implementar una matriz de confusión de la cual se derivan varios parámetros de interés que nos cuantifican que tan bien es esta red a la hora de reconocer lo solicitado, parámetros como precisión y exhaustividad (entre otros). En la siguiente imagen se muestra la matriz de confusión para la red neuronal implementada con keras, donde las filas representan los valores que deberían de ser (observaciones) y las columnas representan los valores predichos por la red.

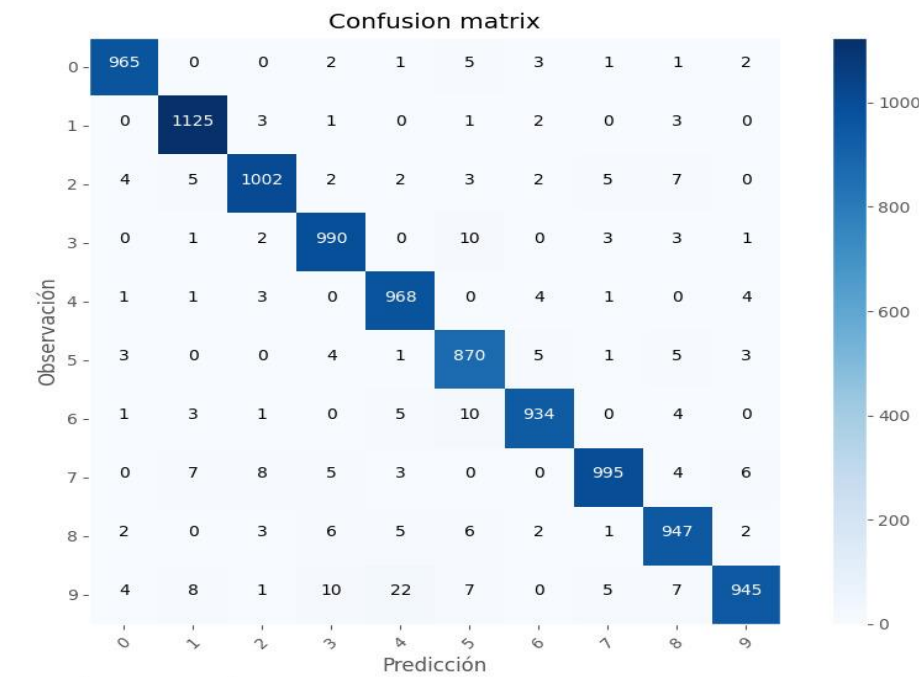


Fig 12. Matriz de confusión para la red neuronal con keras.

Nótese que los datos en la diagonal corresponden al número de aciertos para cada clase mientras que el resto corresponde a los fallos con otras clases.

3.Análisis y resultados finales

Para utilizar las diferentes técnicas para predecir los números se muestra en la siguiente imagen la interfaz gráfica utilizada para escribir el número a predecir, donde poly es polinomial.

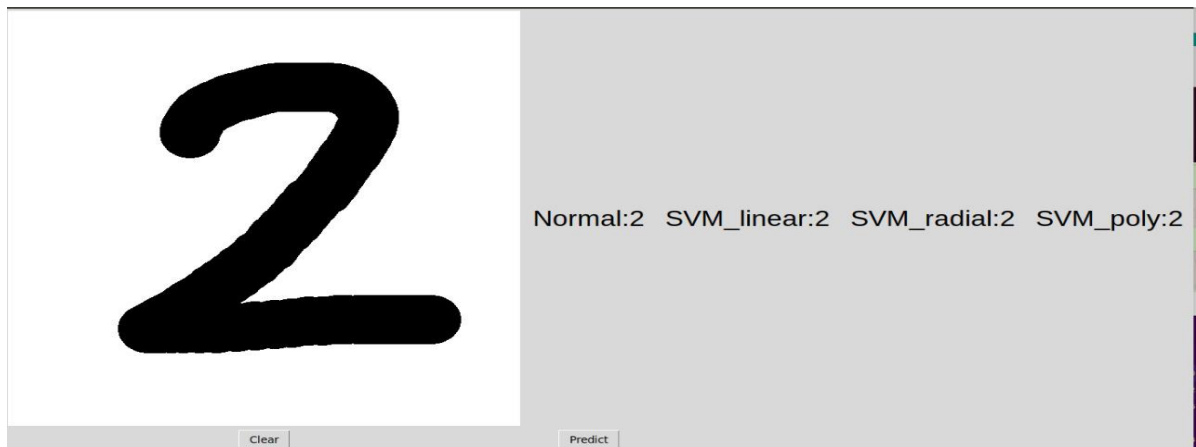


Fig 13. Interfaz para escribir el dato a predecir.

4.Conclusiones

- Usar SVM requiere de mucho tiempo de entrenamiento.
- El reconocimiento con scikit learn depende mucho de los parámetros utilizados, pero mayormente del costo que se le da para penalizar el error.
- De los SVC utilizados el que mejor predice es el radial.
- El reconocimiento de dígitos con keras depende de los parámetros aprendidos por la red neuronal por ende si no se entrena lo suficiente la red la predicción se vuelve imprecisa.