Universidad de la República

Facultad de Ingeniería

Aprendizaje Automático para Datos en Grafos

Laboratorio 4 |

Autores:
Federico Bello
Gonzalo Chiarlone

28 de septiembre de 2025

UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

1.

Para implementar la función alcanza con sortear puntos al azar en el plano, calcular para cada par de vértices distintos el peso dado por $W_{ij} = e^{-\frac{d(i,j)}{2\sigma^2}}$. Si este peso supera una tolerancia r se agrega una arista al grafo de dicho peso. Por lo tanto:

```
def generate_random_graph(N, sigma, r):
    points = np.random.rand(N,2) # devuelve en  [0,1) en vez de [0,1]
    weights = np.zeros((N,N))
    for i in range(N):
        for j in range(N):
            peso = np.exp(-np.linalg.norm(points[i] - points[j])/ (2 * sigma**2))
            if (peso > r) and (i!=j):
                weights[i,j] = peso
    return points, weights
```

De esta forma, se obtuvo el grafo de la figura 1

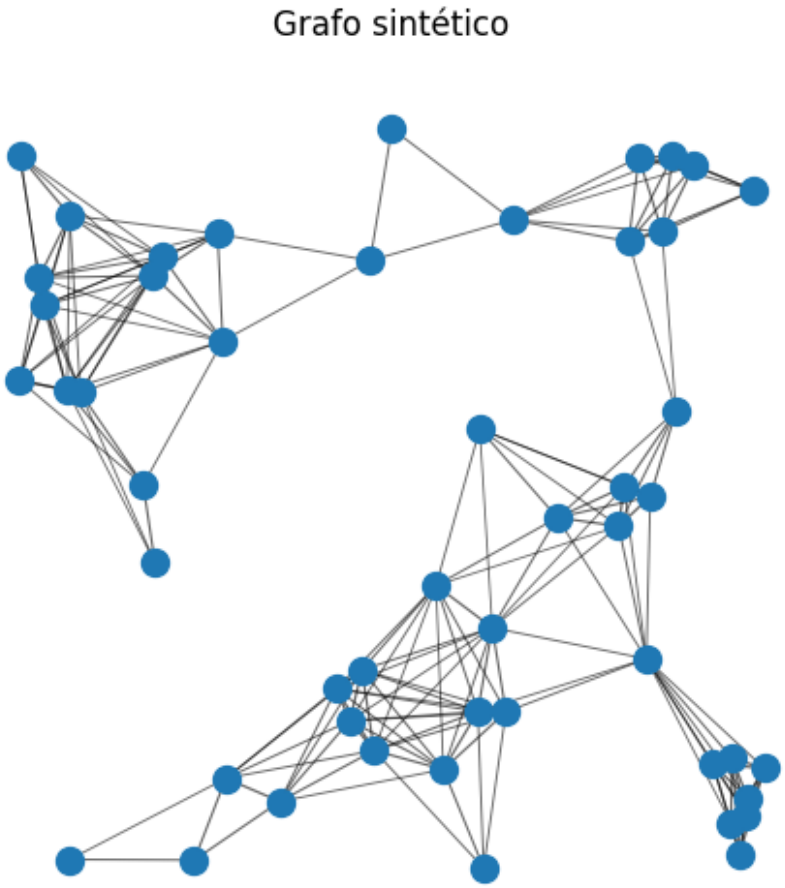
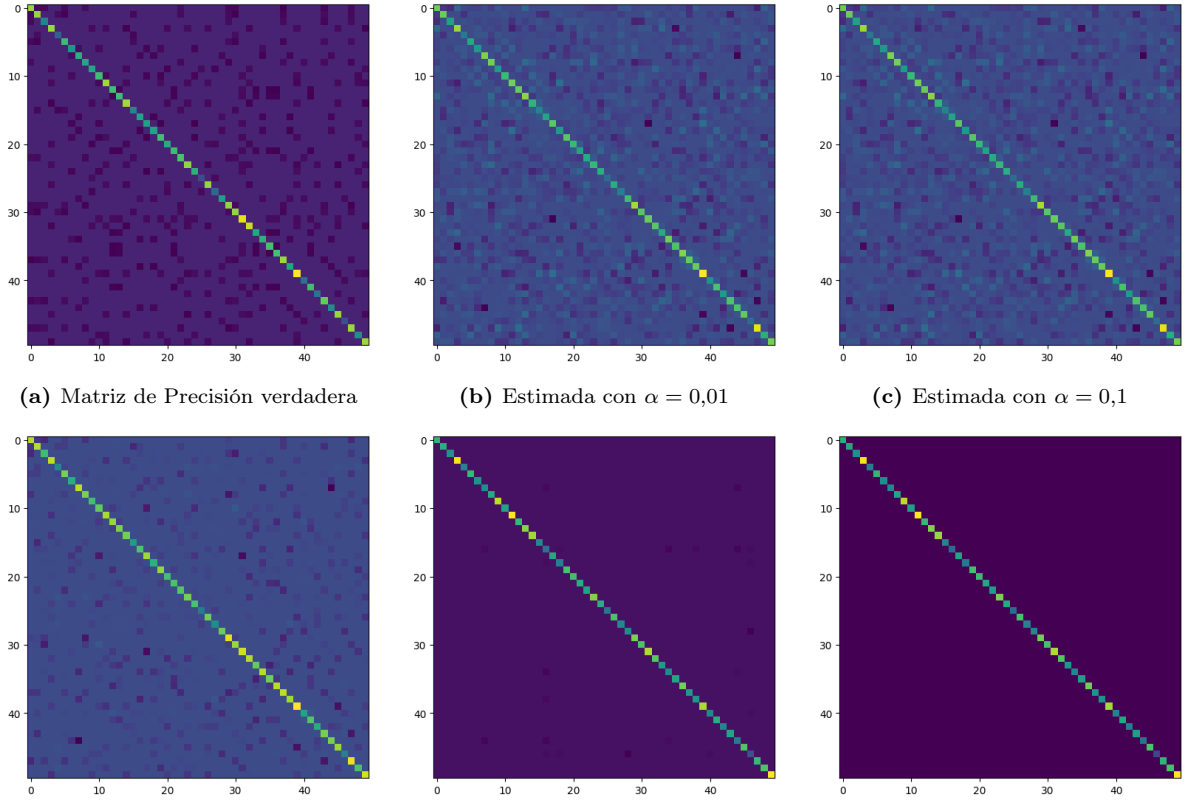


Figura 1: Grafo real sintético para experimentar

Sobre este grafo se toma una señal aleatoria como un vector aleatorio dado por $\mathbf{x} \sim \mathcal{N}(0, L^\dagger)$, donde $L^\dagger$ es la pseudoinversa del laplaciano L del grafo. El objetivo de las siguientes secciones es inferir la topología del grafo a partir de esta señal.

2.

Primero se realizaran experimentos utilizando el método de *Graphical Lasso*. El mismo utiliza el estimador de máxima verosimilitud con un parámetro, en este caso α el cual promueve la esparsidad. Es decir, a mayor valor de α menos aristas tendrá el grafo. Este comportamiento se puede observar en la figura 2, donde se ve claramente como a medida que el valor de α aumenta la matriz estimada se parece cada vez mas a una diagonal, es decir, un grafo sin aristas. En particular, la figura 4c es correspondiente a utilizar el α obtenido mediante *cross-validation*, mientras que 4d es utilizando $\alpha = \sqrt{\frac{\log N}{P}}$, correspondiente a la garantía de *performance* [2].



(a) Matriz de Precisión verdadera **(b)** Estimada con $\alpha = 0{,}01$ **(c)** Estimada con $\alpha = 0{,}1$

(d) Estimada con $\alpha = 0{,}17$ **(e)** Estimada con $\alpha = 0{,}31$ **(f)** Estimada con $\alpha = 3$

Figura 2: Matriz de precisión verdadera y estimadas para distintos calores de α

a

3.

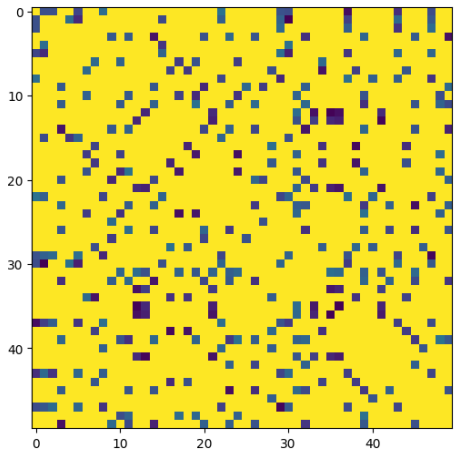
Luego, se experimentara con el método de Meinshausen y Bühlmann, el cual consiste en realizar una regresión lineal entre la señal de un determinado nodo con las señales en el resto. De esta forma, si la señal en dos nodos es similar el coeficiente en la regresión sera alto, mientras que si no tiene correlación tendera a ser bajo. La implementación del método consiste en, para cada nodo, calcular una regresión lineal respecto al resto, teniendo cuidado de no considerar el mismo nodo en la misma:

```
def meinshausen(X,alpha,max_iter=1000):
    k,p = X.shape
    Xn = normalize(X,axis=0)
    B = np.zeros((p,p))
    for j in range(p):
        y = Xn[:, j]
        X_temp = np.delete(Xn, j, axis=1)
        reg = linear_model.Lasso(alpha=alpha, max_iter=max_iter)
        reg.fit(X_temp, y)
        B[:, j] = np.insert(reg.coef_, j, 0)
    epsilon = 1e-2
    B[np.abs(B) < epsilon] = 0
    return B
```

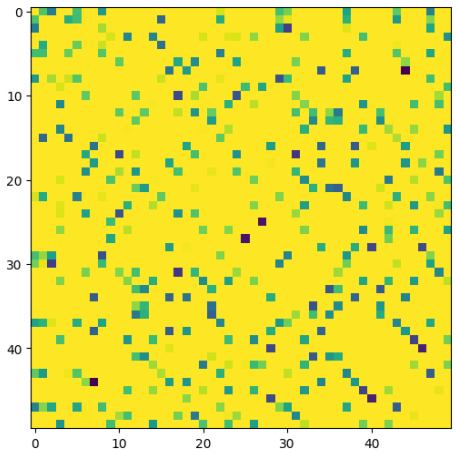
4.

El ultimo método explorado fue el de Kalofolias. El mismo cuenta con dos parámetros: un parámetro α que escala la solución y otro β el cual controla la esparsidad. A diferencia de los métodos anteriores, la esparsidad se controla con la norma de Frobenius en lugar de la norma l_1. Esto lleva a que se obtengan grafos mas densos para mayores valores de β, a diferencia de lo que pasaba con los métodos anteriores.

Si bien tener dos parámetros podría llegar a complejizar la búsqueda, este no es el caso. La proposición 2 de [1] afirma que, dada la solución del modelo $F(Z, \alpha, \beta)$, donde Z es la matriz de distancias se cumple que: $F(Z, \alpha, \beta) = \alpha F(Z, 1, \alpha\beta)$. Esto quiere decir que en lugar de realizar la búsqueda en ambos parámetros, es posible fijar α en un valor cualquiera, por ejemplo 1, buscar en β hasta conseguir una densidad de ejes deseada y luego la escalar la norma del resultado W a la que se desee. Este procedimiento fue el que se utilizo en este caso, donde se decidió fijar $\alpha = 1$, realizando una *grid search* para estimar el β óptimo, consiguiendo así un valor de $\beta = 0{,}7$. Luego, si hubiera una forma de estimar la norma de la matriz de precisión verdadera, podría llevarse el resultado W a tener una norma similar. Mediante este procedimiento y dichos valores de parámetros se llega al resultado de la figura 3b, cuya estructura de aristas tiene un gran parecido a la matriz de precisión real (3a).



(a) Matriz de precisión verdadera

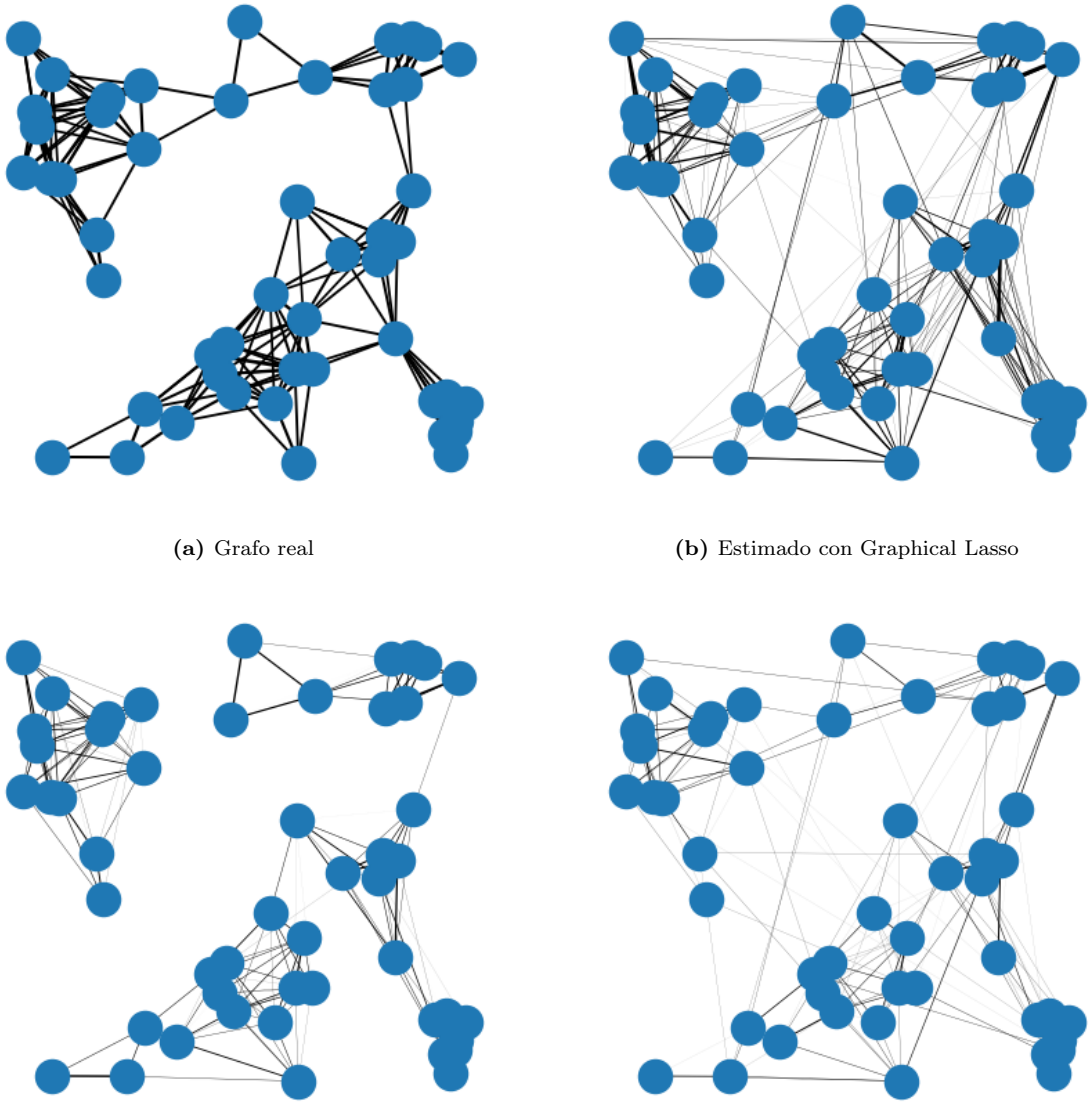


(b) Estructura estimada con Kalofolias

Figura 3: Matriz de precisión verdadera y estimada

5.

Luego de hacer la búsqueda de parámetros en cada método para obtener el mejor resultado posible en cada uno, los grafos obtenidos son los de la figura 10. Se ve como todos los métodos tienen un resultado razonablemente bueno. Sin embargo, al utilizar el α conseguido por *cross-validation* para Graphical Lasso se obtiene un grafo mas denso que el buscado, indicando que incrementar un poco el valor podría surgir en una mejor inferencia. Por otro lado, los métodos de Kalofolias y Meinshausen y Bühlmann obtienen resultados similares, aunque Kalofolias obtiene un grafo final un poco mas parecido al original.



(a) Grafo real

(b) Estimado con Graphical Lasso

(c) Estimado con Kalofolias

(d) Estimado con Meinshausen y Bühlmann

Figura 4: Grafo real y estimado mediante distintos métodos

6.

En la figura 5 se puede observar el grafo generado al utilizar las imágenes de *MNIST flateneadas* como señales de los nodos. Se observan vecindades densas entre los nodos correspondientes a un mismo dígito,

con algunas conexiones entre nodos de correspondientes a dígitos diferentes. Es interesante notar que hay ciertos *clusters* que están mas conectados entre si: 9 y 4, 9 y 7, 8 y 3, 5 y 3, etc. Todas estas conexiones son esperables, ya que son dígitos parecidos entre si, como se observa por ejemplo en la figura 6.

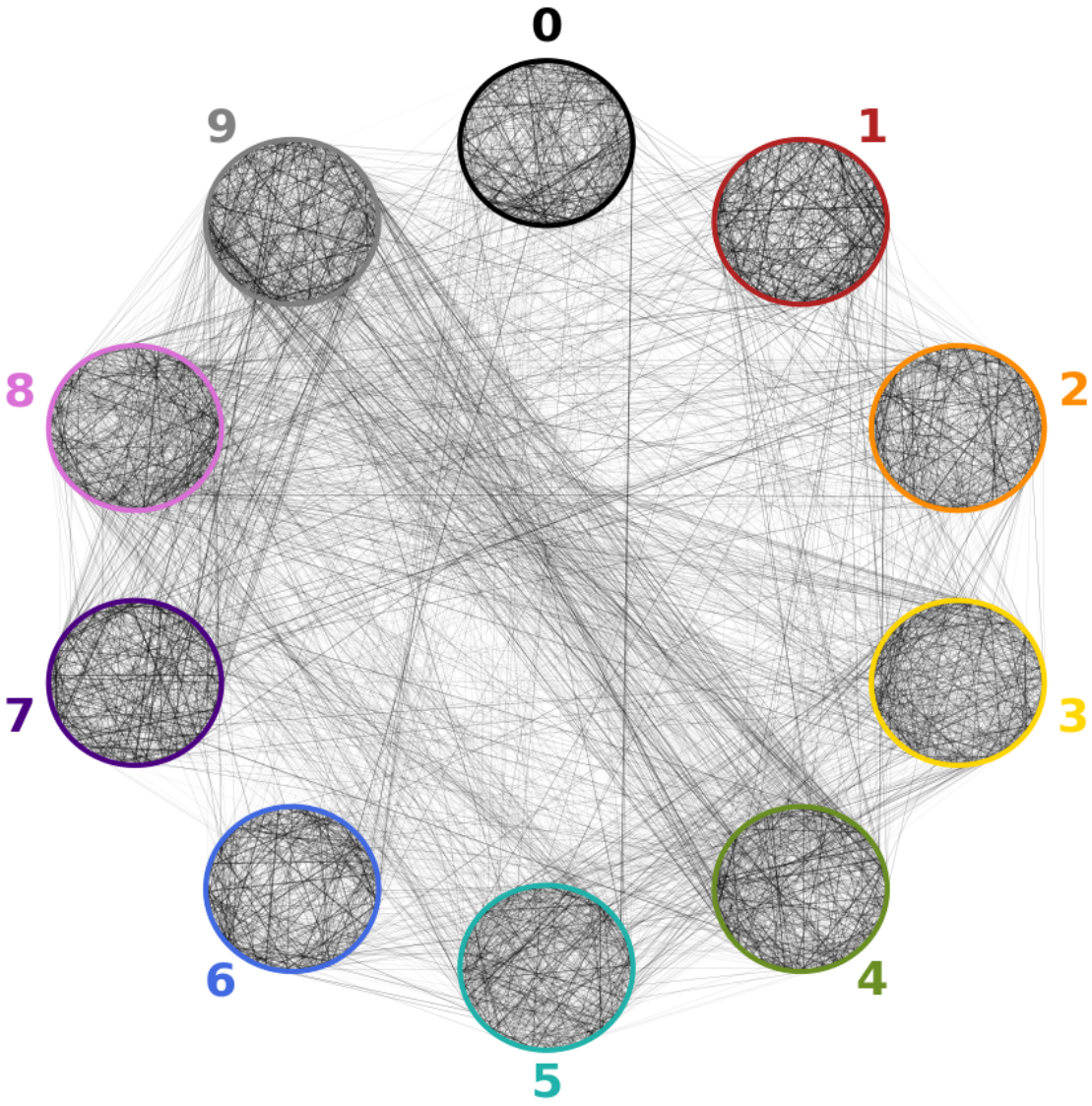


Figura 5: Grafo de Meinshausen

Uno de los problemas de este método es que el hecho de *flatenear* las imágenes para obtener las señales, ya que al hacer esto se pierde el orden de los píxeles. Una forma de corregir esto podría ser, por ejemplo, utilizando *positional encoding*.

Teniendo en cuenta este y otros problemas que presenta este método por lo básico que es, el resultado es bastante bueno ya que se pueden distinguir claramente los *clusters* en el grafo generado.

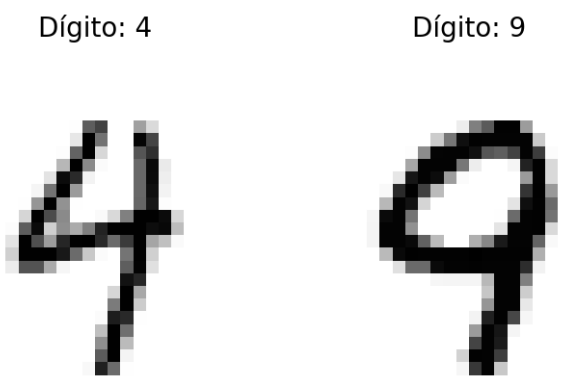


Figura 6: Ejemplos de dígitos parecidos.

7.

En la figura 7 se pueden observar los distintos etiquetados obtenidos al utilizar los distintos métodos. Se utilizo la representación 2D obtenida a partir de *TSNE* para mostrar los resultados. Es interesante notar que ninguno de los resultados esta demasiado alejado del etiquetado real.

8.

Para comparar el etiquetado de cada método resulta conveniente mirar las matrices de confusión (figura 8) en lugar de las figuras obtenidas en la parte anterior, ya que los aciertos y errores se vuelven mas explícitos. En esta la matriz correspondiente al metro de *Kalofolias* se pueden observar muy buenas predicciones para los digitos 0, 2, 6, 7, 8, y predicciones no tan buenas para los digitos 1, 3, 4, 5 y 9. En el caso de los digitos con malas predicciones, resulta interesante analizar las confusiones:

- A las imagenes del digito 1 se le asigna el valor 0.
- En el caso del 3, se les asigna la etiqueta 8 .
- En el caso del 4, se le asigna el 7 en la mitad de los casos.
- Para el 5, se le asigna el numero 8.
- Por ultimo, al 9 se le asigna o el valor 4 o el valor 7.

Notar que para casi todas los casos (todos excepto el 1 y el 9), se pueden ver muchas artistas entre los nodos de ambos *clusters* (ver figura 5), por lo que este comportamiento resulta esperado. El único caso que llama la atención es la poca confusión entre el 5 y el 3, ya que hay una gran cantidad de aristas entre ellos. Además, notar que, como ya se menciono, todas las confusiones cometidas por el método son entre dígitos que se pueden considerar parecidos.

Para comparar el resultado obtenido con *Kalofolias* al obtenido con *TSNE*, es necesario comparar ambas matrices de la figura 8. En concreto, en la figura 9 se puede observar la resta de ambas matrices de confusión, donde los valores positivos corresponden con predicciones hechas por la matriz de *Kalofolias* y no por la de *TSNE*, y viceversa. En este caso se observan las siguientes diferencias:

- Ambos metodos fallan para en el caso de los digitos 1 y 4, pero cometen diferentes errores.
- Para los dígitos **2**, **6** y **8**, el método de ***Kalofolias*** hace una mejor predicción.

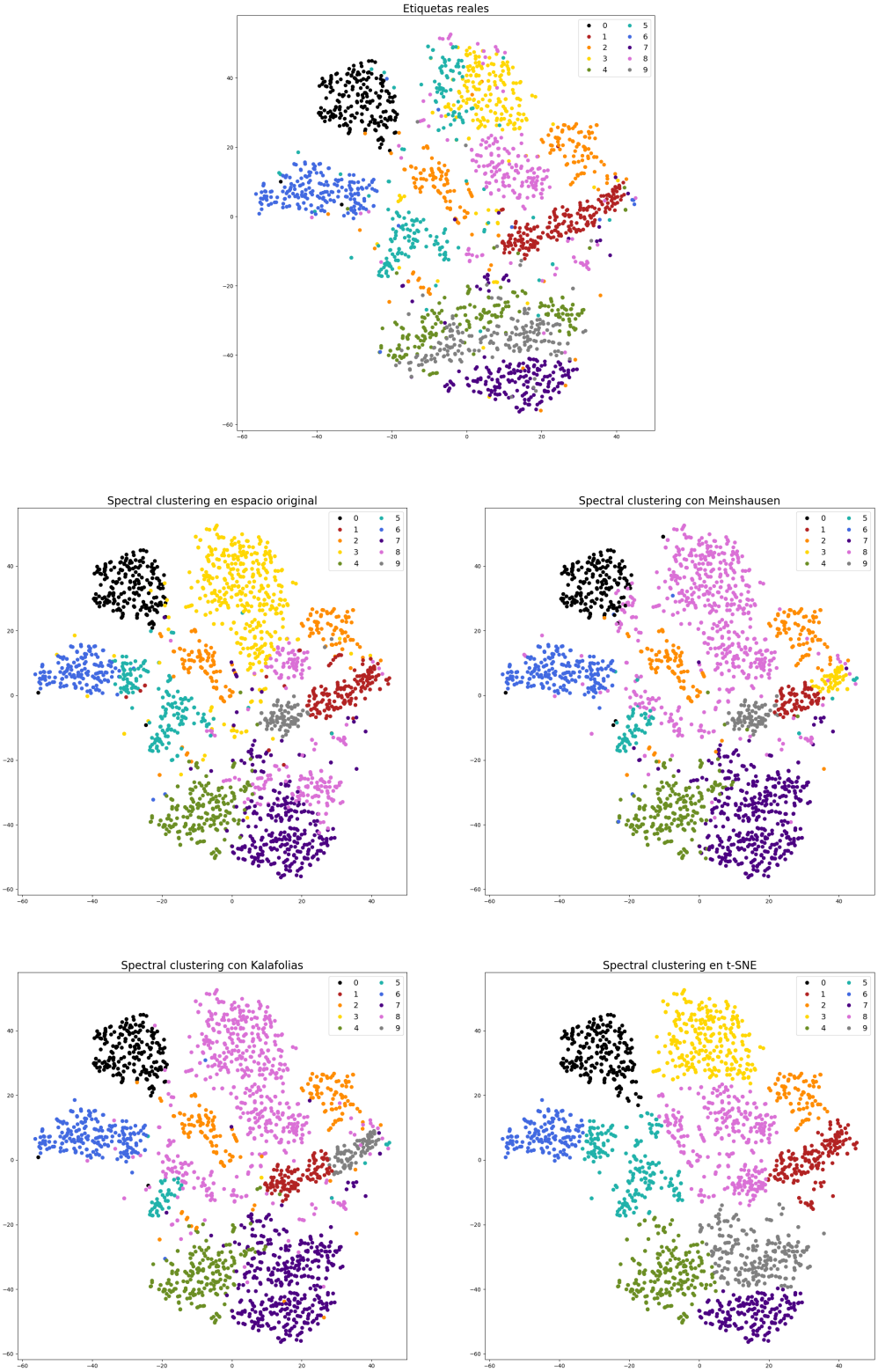


Figura 7: Etiquetados generados a partir de los distintos métodos, representados sobre el espacio 2D generado con *TSNE*.

- Para los dígitos **3**, **5** y **9**, el método de ***TSNE*** es el que hace la mejor predicción.
- Para el resto de los dígitos, ambas predicciones son parecidas.

En el caso del 3, 5 y 9 era esperable tener una gran cantidad de errores ya que el gafo generado por *Kalofolias* tiene muchas conexiones entre *clusters*. Por otro lado, en el caso del 2 y el 6 no se observan muchas conexiones con otros *clusters*, por lo que era esperable que el método lograra una buena predicción. El caso del 8 la predicción de *Kalofolias* es muy buena a pesar de tener tantas conexiones con el 3, ya que tanto a los 3 como a los 8 se les asigna el valor 8, es decir, se toman como un solo *cluster*.

Notar que todas estas confusiones son observables también en la figura 7

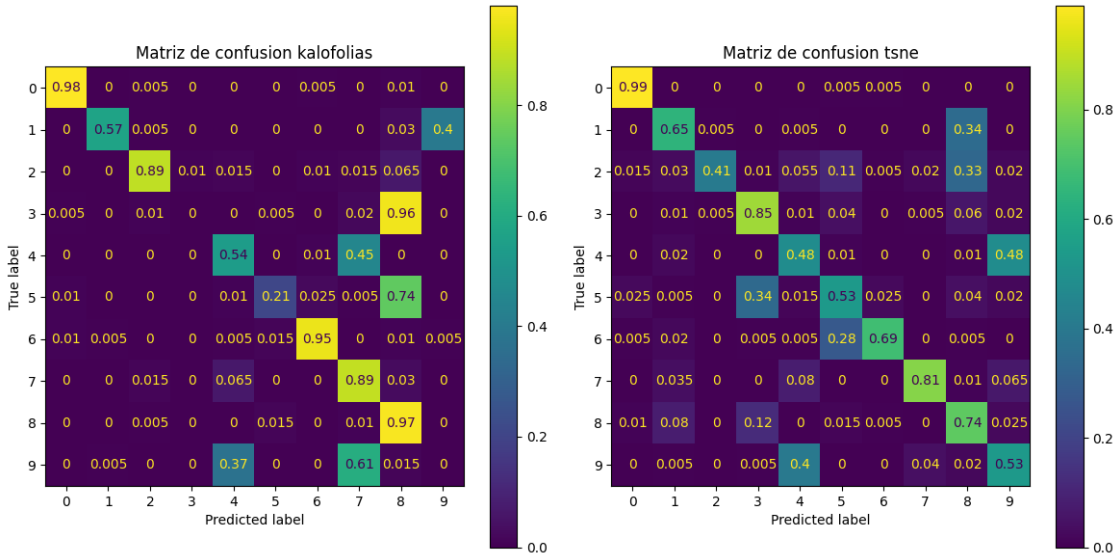


Figura 8: Matrices de confusión obtenidas a partir del etiquetado de ambos métodos. Ambas matrices se encuentran normalizadas.

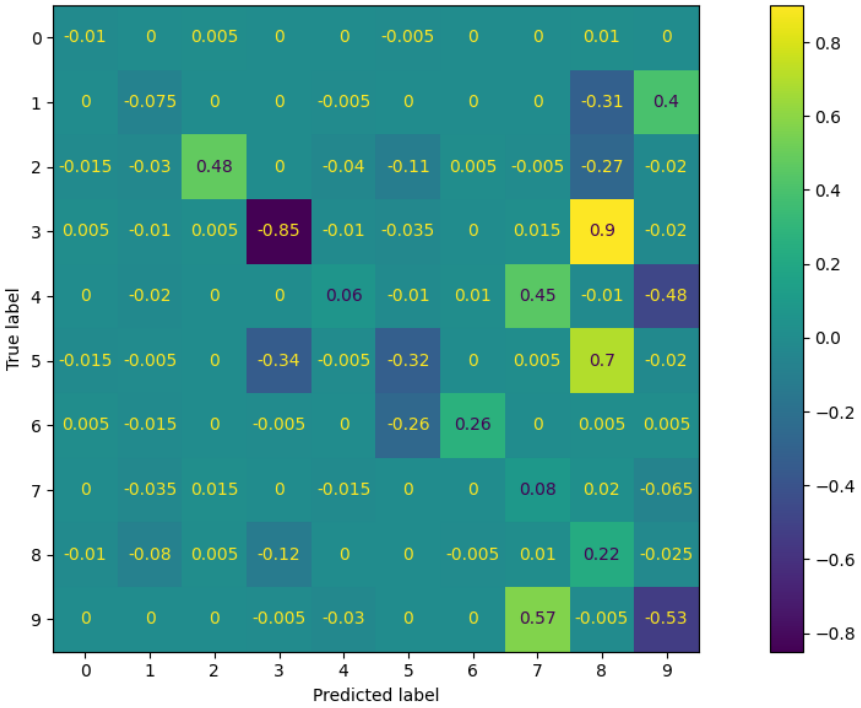


Figura 9: Diferencia entre ambas matrices normalizadas, restando la matriz de *TSNE* a la matriz de *Kalofolias*.

Finalmente, la tabla 1 resume el desempeño de cada método para algunas métricas distintas. Se ve

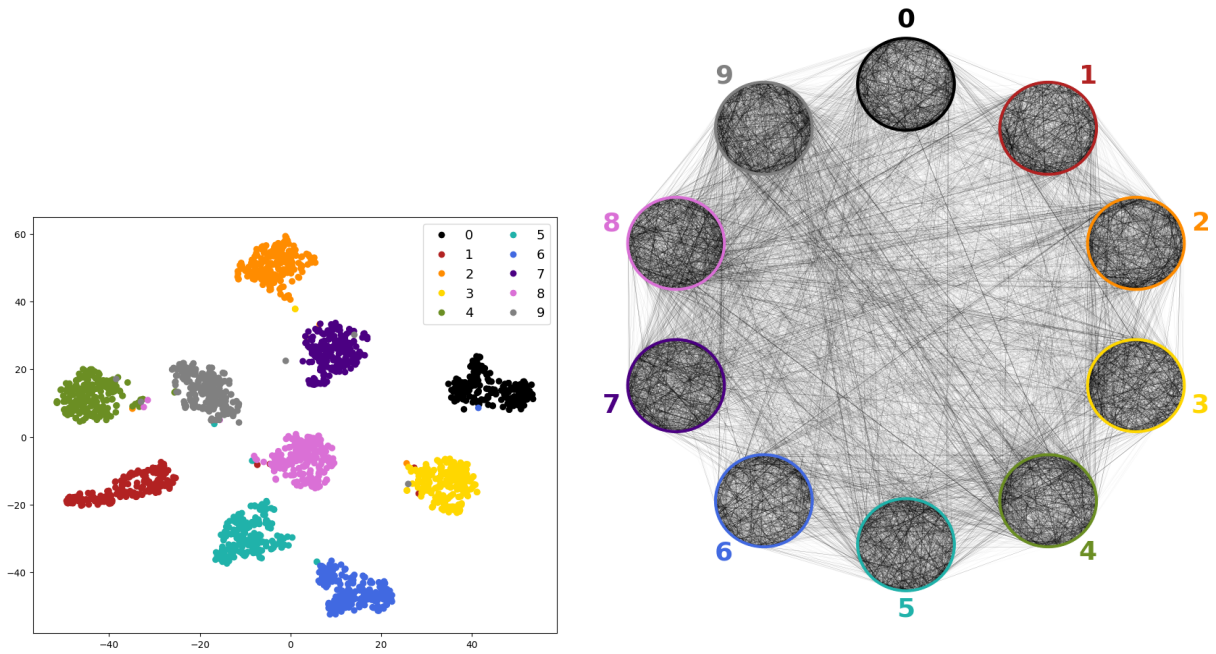
entonces como el Kalofolias es el que obtiene mejor desempeño en todas las métricas, aunque todos los métodos obtienen un desempeño razonable dado lo básico de la solución.

Metrica / Metodo	V-Measure	Rand	Fowlkes-Mallows
Clustering en espacio original	0.62	0.44	0.50
Meinshausen	0.67	0.45	0.53
Kalofolias	0.70	0.50	0.58
Clustering con t-SNE	0.69	0.54	0.60

Cuadro 1: Desempeño de distintos modelos

9. Extra extra

Parece claro que el procedimiento anterior de pasar de la imagen de los dígitos a un vector solamente *aplanando* la matriz puede ser fácilmente mejorable. En particular, al hacer esto se pierden propiedades importantes de la imagen, como puede ser la importancia de la ubicación de cada componente. Una forma de mejorar esto es realizar un *embedding* con una red convolucional, las cuales son ampliamente utilizadas para el tratamiento de imágenes dado que tienen la ventaja de aprender bien las estructuras de las mismas, no perdiendo así la información espacial. De esta forma, se obtienen *embeddings* de una dimensión (posiblemente menor) los cuales, si bien entrenada la red, son mucho mas representativos de cada figura.



(a) *Clusters* **(b)** Grafo con Meinshausen

Figura 10: *Clusters* y grafo de Meinshausen con los nuevos *embeddings*

Con esto en mente, se repitió el procedimiento descripto en las secciones anteriores solo que en este caso para este *embedding* mas representativo, con el objetivo de ver si la intuición de que debería mejorar el resultado efectivamente se cumple. Al repetir el grafo de los dígitos 10b se ve como la cantidad de aristas

entre números distintos disminuye, mientras que parece aumentar entre los números de la misma clase en comparación con la figura 7. Además, la diferencia es mas significativa al bajar de dimensión con t-SNE y visualizar los clusters. Como se ve en la figura 10a los clusters quedan casi perfectos, separados entre si formando las comunidades cuando son *casi* todos de la misma clase. Esto es un fuerte indicativo de que debería ser posible llegar a mejores resultados, dado que solamente con un algoritmo de clusterizacion basado en distancia euclídea como *K-Means* parece que se llegaría a un buen resultado en este espacio.

No solo la figura del grafo y los clusters parecen mejorar, sino que además las métricas indican que el resultado mejoro ampliamente, así como se ve en la tabla 2, viendo las ventajas de mantener la información espacial al momento de trabajar con imágenes. Además, se mantiene el patrón de que el método de Kalofolias es el que obtiene el mejor desempeño.

Metrica / Metodo	V-Measure	Rand	Fowlkes-Mallows
Clustering en espacio original	0.96	0.96	0.97
Meinshausen	0.95	0.94	0.95
Kalofolias	0.97	0.98	0.98
Clustering con t-SNE	0.72	0.44	0.55

Cuadro 2: Desempeño de distintos modelos

Referencias

[1] Vassilis Kalofolias. How to learn a graph from smooth signals, 2016.

[2] Adam J. Rothman, Peter J. Bickel, Elizaveta Levina, and Ji Zhu. Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics*, 2(none), January 2008.