TDA: Proyecto final

Guillermo Santiago Novoa Pérez
125089

18 de mayo de 2016.

En este trabajo se utilizaron técnicas de análisis topológico para intentar encontrar relaciones no lineales en nuestra base de datos.

Base de datos: dígitos

Esta base de datos se encuentra en la libreria 'ElemStatLearn' de r y puede ser llamada con 'data(zip.train)', es muy conocida ya que ofrece un buen ejemplo para diferentes problemas de clasificación y reducción de dimensionalidad. La base de datos trae un total de 7291 números del 0 al 9 escritos a mano, parametrizados por 258 cuadrículas (o variables). En particular, este ejemplo se usa en el reconocimiento de dígitos escritos a mano.

El problema ha capturado la atención de la comunidad de aprendizaje de máquina por muchos años y sigue siendo un problema de referencia para cualquier científico de datos.

Algoritmo

Nuestro algoritmo de clasificación tiene las siguientes alternativas para hacer la gráfica final:

- Elección del filtro
  - Cualquiera de las 258 variables originales.
  - El primer eigenvector de los datos transformados con el algoritmo de Laplacian Eigenmaps (LE).
  - Crear un grid con los dos primeros eigenvectores de los datos transformados con LE.[1]
- Clustering
  - El clustering se hace con el algoritmo de BFS.

[1]El algoritmo de Laplacian Eigenmaps a su vez tiene la opción de elegir los pesos de las distancias entre puntos con un enfoque sencillo (ceros y unos) o usando el "heat kernel" como medida de distancia. Para encontrar qué puntos estaban conectados se utilizó k-vecinos más cercanos.

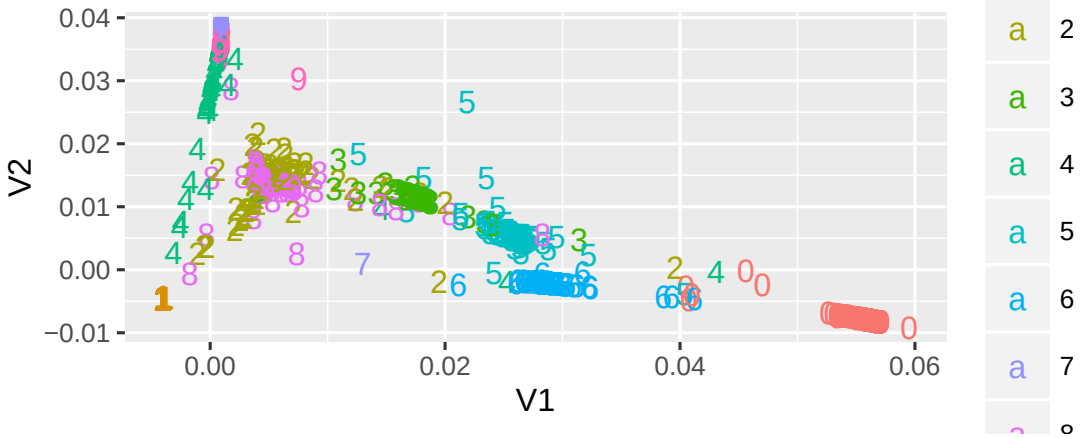
Implementación

Se implementó nuestro algoritmo utilizando como filtro los dos primeros valores propios del algoritmo de Laplacian Eigenmaps, para los pesos se usó el heat kernel.

Los parámetros utilizados fueron los siguientes:

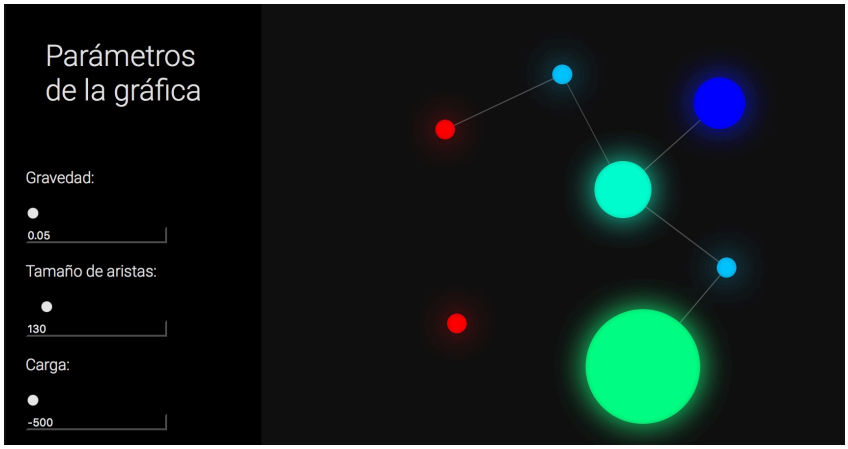
- Filtro (LE)
  - $n = 5$ *número de vecinos más cercanos*
  - $t = 10$ *parámetro del kernel del calor*
- Intervalos
  - $n_{int1} = 3$ *número de intervalos en los que se va a separar el primer eigenvalor de LE*
  - $n_{int2} = 2$ *número de intervalos en los que se va a separar el segundo eigenvalor de LE*
- Clustering
  - $n_2 = 5$ *número de vecinos que se utilizarán para checar el árbol*

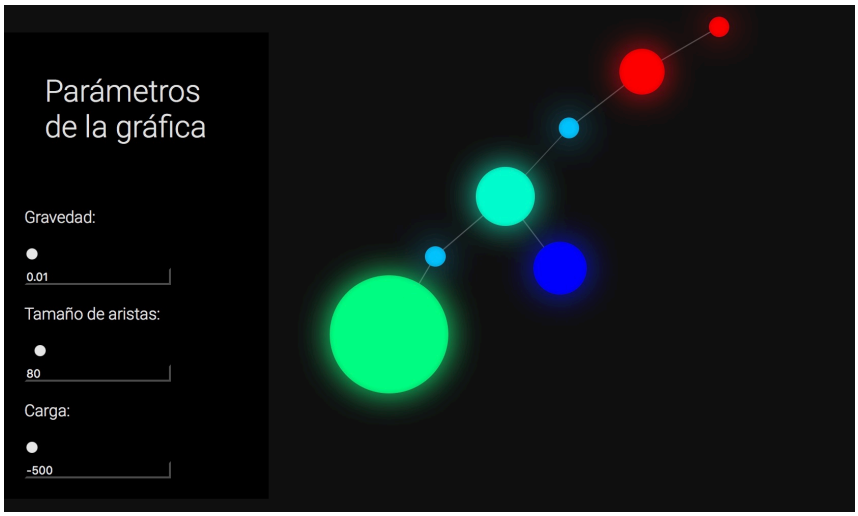
Primero veamos cómo quedan nuestros datos graficados en los dos filtros que estamos usando para pensar qué nos debería quedar en el resultado y si hace algún sentido.



Si nuestro algoritmo funcionara podríamos ver ciertas relaciones entre varios números, seises y cincos, treses y cincos, ochos y doses, y hasta unos y cuatros; además, sólo fijándonos en la primer componente (primer eigenvector del LE), los ceros y los unos toman papeles opuestos.

Resultados





En las dos gráficas existe un nodo que se aleja de los demás, en este nodo se encuentran todos los 0's (en una de las gráficas es una isla, en otra está al extremo). Contrario a este nodo siempre se encuentran aglomerados los 1's (que están cercanos a los 2's). Los nodos del centro contienen a los 5's y 6's, además de los 3's en mucho menor escala.

En si podemos decir que para valores en la izquierda del primer eigenvector se encuentran la mayoría de los números ya que el tamaño de la primer bola es mucho mayor a las demás (1's, 4's, 9's, 7's, 8's y 2's), y conforme se va yendo a la derecha nacen otros grupos (que contienen curvas o loops) terminando con el cero siendo el más redondo". La segunda componente principal (o el segundo eigenvalor del algoritmo LE) no aparece para valores cercanos al cero (parece que sólo divide a números que no son redondos) y puede hablar del número de cortes que tiene cada número, por ejemplo, los 5's tienen valores altos y bajos de esa componente, mientras que los 7's y los 4's son los que mayores valores alcanzan.

Viendo sólo nuestras dos gráficas finales, lo que más sobresale es la diferencia entre unos (palos) y ceros (bolas), pasando por todos los números en medio que parecen clasificarse por dos cosas, número de cortes (por eso la separación entre los cincos y seises) y redondez del número.