Máster en Big Data Analytics

Curso 2017-2018

Introducción al Aprendizaje Automático

Práctica 5

Uso y estudio del k-Means

Jon Ander Gómez Adrián

jon@dsic.upv.es

Departament de Sistemes Informàtics i Computació

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Índice

1. Motivación

El k-Means o k-medias es uno de los algoritmos más importantes de clustering no paramético. Aunque pueden utilizarse otras métricas, habitual-

mente se basa en distancia euclidea entre cada par de puntos en un espacio d-dimensional $\mathbb{R}^d$.

Existen otras métricas que para según qué tipo de datos pueden resultar más adecuadas. Pero quedan fuera del alcance de este curso.

2. Objetivos

2.1. Conceptuales

- Entender el algoritmo k-Means y la función de distorsión

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2$$

  para utilizarla como criterio de parada.

- Combinar uno o más criterios de parada. Cuando se trabaja con *training sets* grandes es posible que algunas muestras estén oscilando entre dos o más clases y el número de muestras que cambian de *cluster* nunca llega a ser cero.

- Ver en ejemplos sencillos la importancia de la inicialización de las medias.

- Estudiar para un ejemplo sintético la evolución de la función de distorsión.

- Aplicar esta técnica de *clustering* para reducir el tamaño de imágenes con poca pérdida de la calidad en base a utilizar un *codebook* y pocos bits por cada píxel.

2.2. En el manejo de utilidades

- Crear una clase para implementar el algoritmo k-Means.

- Entender la implementación de la inicialización de *Katsavounidis* utilizando `numpy`.

- Comprobar si esta inicialización es la más idónea siempre.

3. Tareas

3.1. Tarea 1

Estudiad el código `MyKMeans.py` para ver la implementación de lo que se ha explicado en clase. Probad dicha clase ejecutando el programa Python `testing-k-means.py`. Variad el número de *clusters* para ver la evolución en este ejemplo sintético.

Probad el modo de inicialización de los *clusters* en base a *Katsavounidis* y observad las diferencias. ¿Qué tipo de inicialización es mejor?

3.2. Tarea 2

Utilizad la implementación de `MyKMeans.py` para discretizar imágenes. Para ello primero debéis descargar el ejemplo de cuantificar imágenes de la siguiente URL:

`http://scikit-learn.org/stable/auto_examples/cluster/plot_color_quantization.html`

En este programa hay dos ejemplos de discretización de imágenes con objeto de reducir su tamaño. Bajando el número de colores distintos se consigue disminuir la cantidad de memoria necesaria para almacenar una imagen. También se consigue mediante interpolación para reducir la cantidad de píxeles, pero ese no es nuestro objetivo aquí.

Se pide que añadáis un tercer ejemplo utilizando objetos de la clase `MyKMeans.py` y probad el efecto que tiene el tipo de inicialización `random` y la tipo `Katsavounidis`.

3.3. Tarea 3

La versión de `MyKMeans.py` no incorpora la posibilidad de realizar varias inicializaciones aleatorias, realizad el entrenamiento con cada una de las inicializaciones y, posteriormente, quedaros con la solución que minimiza la función de distorsión.

Esta tarea es para realizarla como trabajo fuera del aula y el alumno deberá entregar un report de no más de 6 páginas indicando cómo ha modificado el código, mostrando tablas y/o gráficas con resultados y explicando, a modo de conclusiones, si el hecho de probar con distintas inicializaciones aleatorias permite alcanzar mejores resultados (mayor precisión) para algunos *datasets*.

Se recomienda realizar pruebas con el *dataset* Iris que es pequeño, indicándole al *k-Means* que ajuste al mismo número de centroides que clases distintas hay en dicho *dataset*. El alumno deberá:

1. Identificar a qué clase corresponde cada centroide obtenido.

   Para esta parte se realiza una predicción y se asume que cada centroide se corresponde con la clase que más se repite entre las muestras que se le asignan. Esto no tiene porque ir bien cuando la dimensionalidad es alta, pero en el caso de este *dataset* las muestras pertenecen a $\mathbb{R}^4$.

2. Comprobar en qué grado la clasificación por distancia mínima a un centroide se ajusta a las verdaderas clases.

   Para esto, una vez emparejados los centroides que encuentra el *k-Means* y las labels o etiquetas de las muestras utilizadas para test, se obtiene el porcentaje de aciertos o de fallos, como se hace en algunos ejemplos de clasificación que se han visto en prácticas anteriores.

Utilizad ambas versiones del *k-Means* la que viene con *Scikit Learn* y la proporcionada para esta práctica.

Si da tiempo, se recomienda buscar otro *dataset* en `http://mldata.org` u otra fuente y realizar el mismo proceso.

4. Conclusiones

Con esta práctica se ha pretendido que los alumnos realicen cambios en código Python de mayor complejidad, en particular por lo que respecta a la tarea 3 que implica modificar bastante el código de la clase `MyKMeans.py`.

Por otra parte, se ha trabajado con un ejemplo artificial para ver la evolución del algoritmo en dos dimensiones y la relevancia de la inicialización.

También se ha presentado y trabajado en un ejemplo de uso habitual del *k*-Means en discretización de imágenes para reducir su tamaño en el caso de compresión con pérdida. Las técnicas de compresión sin pérdida son mucho más complejas y no están basadas en este algoritmo.

5. Bibliografía

[1] S. authors. Deep Learning. `http://www.deeplearning.net`, 2015. [Online: accessed June 2015].

[2] Y. Bengio, I. J. Goodfellow, and A. Courville. Deep learning. Book in preparation for MIT Press, 2015.

[3] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[4] C. M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006.

[5] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Addison-Wesley, second edition, 2000.

[6] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):2–42, 2006.

[7] G. Hinton. A practical guide to training restricted boltzmann machines, 2010.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, JMLR*, 12:2825–2830, 2011.

[9] Wikipedia. Machine Learning. `http://en.wikipedia.org/wiki/Machine_learning`, 2015. [Online: accessed April 2015].