Machine Learning hw4 Report

物理三 b03202017 李漪莛

1. Analyze the most common words in the clusters. Use TF-IDF to remove irrelevant words such as “the”.

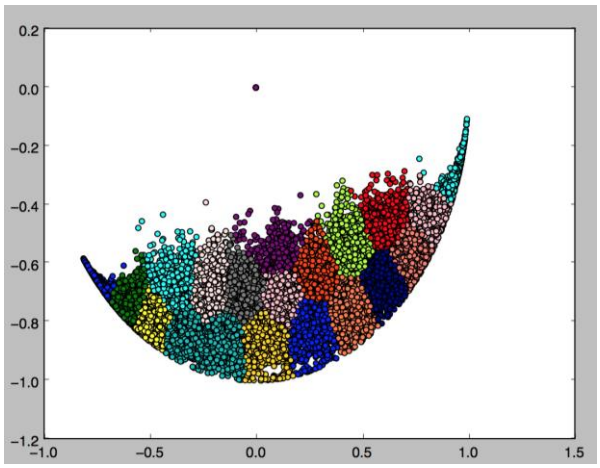
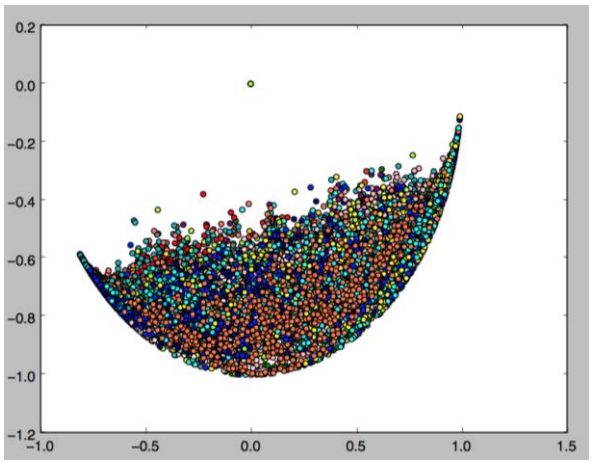
- 在此 idf 的算法為：idf(word) = $\log\left(\frac{1+n}{1+df(word)}\right)+1$，n 表示所有 title 的字數，df（word）表示有多少 title 包含這個 word，“加 1”可做 smoothing。
- 計算每個 label 的 tf 和 tf-idf 個別得到的分數，顯示分數前五高的字於表一。（由於版面關係，只顯示前 6 個 label 的前五名。）
- 其中可看出在 tf 時，第一名的字可能就是 label 本身，但接下來的多為介系詞等較無意義、不相關的字，若加上 idf 之後，可找到更能顯示出該類別特色的字。
- 從表中也可看出預處理可再做更好，如：post, posts 應有辦法被認定為同一字等。

label		1 (most common)	2	3	4	5
wordpress	tf	wordpress	to	in	a	how
	idf	wordpress	post	posts	plugin	category
oracle	tf	oracle	to	in	a	how
	idf	oracle	sql	table	database	query
svn	tf	svn	to	a	subversion	how
	idf	svn	subversion	repository	files	file
apache	tf	apache	to	how	a	in
	idf	apache	modrewrite	htaccess	rewrite	server
excel	tf	excel	to	in	a	how
	idf	excel	vba	file	data	cell
matlab	tf	matlab	in	to	a	how
	idf	matlab	matrix	function	array	plot

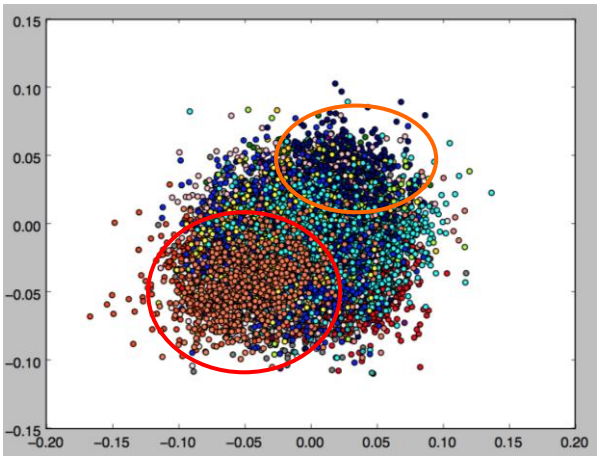
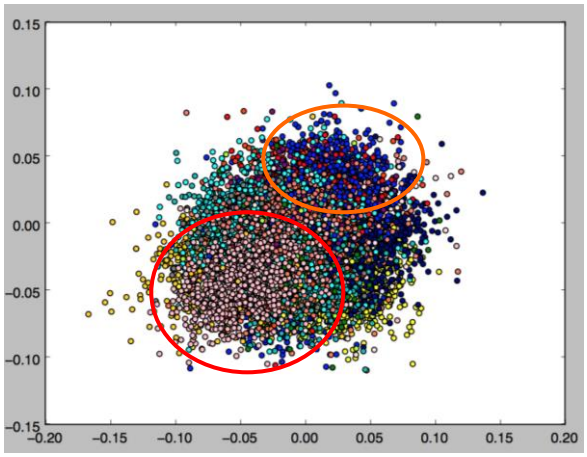
2. Visualize the data by projecting onto 2-D space. Plot the results and color the data points using your cluster predictions. Comment on your plot. Now plot the results and color the data points using the true labels. Comment on this plot.

下圖都是由第三題的 D 方法得到的向量，左圖為 kmeans cluster，右圖為實際的 label：

- 若 word embedding 投影到二維：兩張圖相差很大，並且向量只集中在下半平面。

- 若 word embedding 投影到 300 維的平面，並且只顯示其中兩維：可看出對應的分群，如圈起來的部分（左圖的橘色點對應右圖的粉色點等）。

3. Compare different feature extraction methods.

- 試過四種方法：

  (A)tf-idf 後直接 PCA 降維，再 cluster。

  (B)tf-idf 後由於有 12000 維左右，不容易用 autoencoder train，因此中間先用 PCA 降到 200~1000 維，再接 autoencoder 和 cluster。

  (C)用 tf-idf 後直接接 LSA，再 cluster。

  (D)用 word embedding (word2vec)，投影成向量，再 cluster。

- cluster 的方法則試過(1)kmeans、agglomerative、(2)自行計算出一個 threshold，(3)500 萬組測資中，兩兩比較，相似度小於 threshold 的視為同一類。由於 word2vec 中向量中心點不一定在原點，不適合 agglomerative，也不能對向量平移，會破壞 word2vec 的資料特性（角度相近表示較相似）；kmeans 和計算相似度，在不同情況下各有優劣。
- 下表顯示 public score，四種方法中 D 可得到最佳的準確度。

A-1	tf-idf + PCA (to dim = 100) + kmeans	0.17306
A-2	tf-idf + PCA (to dim = 19) + kmeans	0.25672
A-3	tf-idf + PCA (to dim = 10) + kmeans	0.21308
B-1	tf-idf + PCA (to dim = 256) + auto (to dim 128) + kmeans	0.21435
B-2	tf-idf + PCA (to dim = 1024) + auto (to dim 256) + kmeans	0.29019
C-1	tf-idf + LSA (to dim = 100)	0.45639
D-1	word2vec on “title” (to dim = 100) + kmeans	0.31503
D-2	word2vec on “title + doc” (to dim = 300) +斷詞+ kmeans	0.51612
D-3	word2vec on “title + doc” (to dim = 300) +斷詞+ calculate similarity	0.57016
D-4	word2vec on “3*title + doc” (to dim = 300) +斷詞+ calculate similarity	0.60660
D-5	word2vec on “6*title + doc” (to dim = 300) +斷詞+ calculate similarity	0.60828
D-6	word2vec on “6*title + doc” (to dim = 300) +斷詞+ kmeans	0.66610
所有 tf-idf 都已包含 remove stopwords。 所有方法都經過 preprocess，包含：轉成小寫、在標點符號斷開、去掉網址和 stopwords，stopwords 包含 python 內建和自行上網下載。 經過斷詞系統後，可使 title 無法找到任何有效字的筆數從 17 筆降到 4 筆。		

- A-1&A-2：由於 kmeans 計算每個維度中的歐幾里德距離，PCA 越後面的維度，variance 越小，直接用 kmeans 去分群反而不準確，
- B-1&B-2：PCA 後若經過 autoencoder，則不會有在不同維度中 variance 不同的問題，因此可在更高維的空間中作 kmeans，得到的效果較好。
- D：資料中包含“重複 6 份 title”和 doc 可得到更高的準確度（有較準確的 domain）。

4. Try different cluster numbers and compare them. You can compare the scores and also visualize the data.

# cluster	15	19	20	21	25	28
score	0.457	0.652	0.666	0.652	0.652	0.664

大致上還是分成 20 群效果較佳，並且「增加維度」和「減少維度」相比，準確度不會差那麼多。