Data Science Lab - HUST
Summer 2019

Học máy với Python

Session 2

THS. NGÔ VĂN LINH
NGUYỄN BÁ KHẢI – NGUYỄN ĐỨC TÙNG

Tổng quan

- Triển khai Kmeans
- Sử dụng Scikit-learn:
  - > Kmeans
  - > SVMs

Triển khai Kmeans

- Nội dung chính:
  1. Nhắc lại thuật toán
  2. Ý tưởng triển khai
  2. Triển khai

1. Nhắc lại thuật toán

- Input:

  > Tập dữ liệu $R = \{r_d : d \in D\}$ với $r_d \in \mathbb{R}^{|V|}$ là biểu diễn tf-idf của d

  > Số cụm K

- Output: $A = \{a_d : d \in D\}$ với $a_d \in \{1, 2, \ldots, K\}$ cho biết d được phân vào cụm nào.

1. Nhắc lại thuật toán

- Procedure:

> B1: Khởi tạo tâm cho K cụm:

$E = \{ e_k \}$ với e_k là tâm của cụm k ,

$k \in \{1, 2, \ldots, K\}$ và $|E| = K$,

E là 1 tập con gồm K phần tử được lấy[*] từ $R = \{r_d : d \in D\}$

[*] Lấy ngẫu nhiên, hoặc lấy theo chiến lược

1. Nhắc lại thuật toán

- Procedure:

> B1: Khởi tạo tâm cho K cụm: $E = \{ e_k \}, |E| = K$

> B2: Lặp cho tới khi hội tụ:

* Với mỗi $d \in D$:

+ Tính $similarity(r_d , e_k)$

+ Gán d vào cụm k* với $k^* = \underset{k}{\operatorname{argmax}}(similarity(r_d , e_k))$

* Cập nhật lại E

1. Nhắc lại thuật toán

- Lựa chọn điều kiện dừng:

  > Số bước lặp vượt quá 1 ngưỡng đặt trước: iteration > max_iters

  > $E = \{ e_k \}$ thay đổi không đáng kể:

  $|E_{new} \setminus E_{old}| < n_0$ với $n_0 << K$

  > Độ tương đồng trung bình không tăng hoặc tăng không đáng kể

  * Độ giảm lỗi phân cụm: $S_{new} - S_{old} < \varepsilon$

  * Lỗi phân cụm: $S = \frac{1}{|D|} \sum_{r_d \in D} \text{similarity}(r_d, e_{a_d})$

1. Nhắc lại thuật toán

- Đánh giá chất lượng phân cụm:

  > Purity:

$$\text{purity}(\Omega, \mathbb{C}) = \frac{1}{N} \sum_{k=1,K} \max_{j} |\omega_k \cap c_j|$$

với ω_k = {d: a_d = k, d ∈ D}

và c_j = {d: label(d) = j, d ∈ D}

$\omega_k \cap c_j$: tập hợp các văn bản trong cụm k có nhãn j

1. Nhắc lại thuật toán

- Đánh giá chất lượng phân cụm:

  > NMI (*normalized mutual information*):

$$\mathrm{NMI}(\Omega, \mathbb{C}) = \frac{\mathrm{I}(\Omega,\mathbb{C})}{[\mathrm{H}(\Omega)+\mathrm{H}(\mathbb{C})]/2}$$ với $\mathbb{C} = \{c_0, c_1, \ldots, c_{J-1}\}$, J là số lớp

với $$\mathrm{I}(\Omega, \mathbb{C}) = \sum_k \sum_j \frac{|\omega_k \cap c_j|}{N} \cdot \log_{10} \frac{N \cdot |\omega_k \cap c_j|}{|\omega_k| . |c_j|}$$

$$\mathrm{H}(\Omega) = -\sum_k \frac{|\omega_k|}{N} \cdot \log_{10} \frac{|\omega_k|}{N}$$

$$\mathrm{H}(\mathbb{C}) = -\sum_k \frac{|c_j|}{N} \cdot \log_{10} \frac{|c_j|}{N}$$

2. Ý tưởng triển khai

- 20newsgroups dataset đã tiền xử lý:

```
1  0<fff>49960<fff>9370:0.0967868358288 8553:0.229793603446 7347:0.132
2  0<fff>51060<fff>6637:0.0264613146274 9930:0.0291134757707 2501:0.03
3  0<fff>51119<fff>8648:0.16054261467 5695:0.211258519544 9671:0.62138
4  0<fff>51120<fff>10288:0.127635484843 9671:0.372830827488 10305:0.04
5  0<fff>51121<fff>7763:0.538997270778 10255:0.195617964114 9473:0.400
6  0<fff>51122<fff>10030:0.0719050870906 10174:0.0610246365229 3930:0.
7  0<fff>51123<fff>10164:0.7012549744 10288:0.170180646457 9410:0.5434
8  0<fff>51124<fff>8717:0.172847984547 5979:0.225946768213 8475:0.7192
9  0<fff>51125<fff>9370:0.137114684091 9004:0.150470242337 10022:0.205
10 0<fff>51126<fff>10243:0.205727965198 9511:0.395948539607 8294:0.506
11 0<fff>51127<fff>9385:0.546772136773 10235:0.285813329788 10261:0.74
12 0<fff>51128<fff>9868:0.341744976285 10054:0.297276273338 9333:0.417
13 0<fff>51130<fff>10160:0.211947851892 9241:0.341062262272 10255:0.31
14 0<fff>51131<fff>10288:0.145869125535 10058:0.169030324161 9977:0.18
15 0<fff>51132<fff>9370:0.274229368182 10288:0.0850903232285 10135:0.1
```

2. Ý tưởng triển khai

- Mỗi cụm ta lưu trữ các thông tin sau:
  - > centroid: tâm cụm
  - > members: danh sách các điểm dữ liệu trong cụm
- Mỗi điểm dữ liệu d ta sẽ lưu trữ thông tin sau:
  - > r_d: biểu diễn tf-idf r_d của văn bản d
  - > label: newsgroup của văn bản d
  - > doc_id: tên file chứa văn bản d

2. Ý tưởng triển khai

- Ta sẽ xây dựng 3 lớp:
  - > 2 lớp cho lưu trữ thông tin:
    - * class Cluster
    - * class Member
  - > 1 lớp Kmeans cho triển khai thuật toán
    - * class Kmeans

2. Ý tưởng triển khai

- class Member:

```
160     class Member:
161         def __init__(self, r_d, label=None, doc_id=None):
162             self._r_d = r_d
163             self._label = label
164             self._doc_id = doc_id
```

- class Cluster:

```
148     class Cluster:
149         def __init__(self):
150             self._centroid = None
151             self._members = []
152 
153         def reset_members(self):
154             self._members = []
155 
156         def add_member(self, member):
157             self._members.append(member)
```

2. Ý tưởng triển khai

- class Kmeans:

```
167  class Kmeans:
168      def __init__(self, num_clusters):...
174
175      def load_data(self, data_path):...
203
204      def random_init(self, seed_value):...
215      def compute_similarity(self, member, centroid):...
217      def select_cluster_for(self, member):...
228      def update_centroid_of(self, cluster):...
235      def stopping_condition(self, criterion, threshold):...
259      def run(self, seed_value, criterion, threshold):...
288
289      def compute_purity(self):...
296      def compute_NMI(self):...
```

Class Kmeans

- Hàm khởi tạo:

```
167  class Kmeans:
168      def __init__(self, num_clusters):
169          self._num_clusters = num_clusters
170          self._clusters = [Cluster() for _ in
171                            range(self._num_clusters)]
172          self._E = []  # list of centroids
173          self._S = 0  # overall similarity
```

Class Kmeans

- Đọc dữ liệu:

```
175     def load_data(self, data_path):
176         def sparse_to_dense(sparse_r_d, vocab_size):...
183
184         with open(data_path) as f:
185             d_lines = f.read().splitlines()
186         with open('../datasets/20news-bydate/words_idfs.txt') as f:
187             vocab_size = len(f.read().splitlines())
188
189         self._data = []
190         self._label_count = defaultdict(int)
191         for data_id, d in enumerate(d_lines):
192             features = d.split('<fff>')
193             label, doc_id = int(features[0]), int(features[1])
194             self._label_count[label] += 1
195             r_d = sparse_to_dense(sparse_r_d=features[2], vocab_size=vocab_size)
196
197             self._data.append(Member(r_d=r_d,label=label,doc_id=doc_id))
```

Class Kmeans

- Đọc dữ liệu: Hàm sparse_to_dense

```
176         def sparse_to_dense(sparse_r_d, vocab_size):
177             r_d = [0.0 for _ in range(vocab_size)]
178             indices_tfidfs = sparse_r_d.split()
179             for index_tfidf in indices_tfidfs:
180                 index, = int(index_tfidf.split(':')[0])
181                 tfidf = float(index_tfidf.split(':')[1])
182                 r_d[index] = tfidf
183             return np.array(r_d)
```

Class Kmeans

- Chạy thuật toán: Hàm run

```
262     def run(self, seed_value, criterion, threshold):
263         self.random_init(seed_value)
264 
265         # continually update clusters until convergence
266         self._iteration = 0
267         while True:
268             # reset clusters, retain only centroids
269             for cluster in self._clusters:
270                 cluster.reset_members()
271             self._new_S = 0
272             for member in self._data:
273                 max_s = self.select_cluster_for(member)
274                 self._new_S += max_s
275             for cluster in self._clusters:
276                 self.update_centroid_of(cluster)
277 
278             self._iteration += 1
279             if self.stopping_condition(criterion, threshold):
280                 break
```

Class Kmeans

- Chạy thuật toán: xác định cụm cho từng điểm dữ liệu

```
217     def select_cluster_for(self, member):
218         best_fit_cluster = None
219         max_similarity = -1
220         for cluster in self._clusters:
221             similarity = self.compute_similarity(member, cluster._centroid)
222             if similarity > max_similarity:
223                 best_fit_cluster = cluster
224                 max_similarity = similarity
225 
226         best_fit_cluster.add_member(member)
227         return max_similarity
```

Class Kmeans

- Chạy thuật toán: cập nhật lại tâm cụm

```
229     def update_centroid_of(self, cluster):
230         member_r_ds = [member._r_d for member in cluster._members]
231         aver_r_d = np.mean(member_r_ds, axis=0)
232         sqrt_sum_sqr = np.sqrt(np.sum(aver_r_d ** 2))
233         new_centroid = np.array([value / sqrt_sum_sqr for value in aver_r_d])
234 
235         cluster._centroid = new_centroid
```

Class Kmeans

- Chạy thuật toán: Kiểm tra điều kiện dừng – max_iters

```
237     def stopping_condition(self, criterion, threshold):
238         criteria = ['centroid', 'similarity', 'max_iters']
239         assert criterion in criteria
240         if criterion == 'max_iters':
241             if self._iteration >= threshold:
242                 return True
243             else:
244                 return False
```

Class Kmeans

- Chạy thuật toán: Kiểm tra điều kiện dừng – centroid

```
245         elif criterion == 'centroid':
246             E_new = [list(cluster._centroid) for cluster in self._clusters]
247             E_new_minus_E = [centroid for centroid in E_new
248                              if centroid not in self._E]
249             self._E = E_new
250             if len(E_new_minus_E) <= threshold:
251                 return True
252             else:
253                 return False
```

Class Kmeans

- Chạy thuật toán: Kiểm tra điều kiện dừng – similarity

```
254         else:
255             new_S_minus_S = self._new_S - self._S
256             self._S = self._new_S
257             if new_S_minus_S <= threshold:
258                 return True
259             else:
260                 return False
```

```
271             self._new_S = 0
272             for member in self._data:
273                 max_s = self.select_cluster_for(member)
274                 self._new_S += max_s
```

Class Kmeans

- Đánh giá chất lượng phân cụm: Tính purity

```
282     def compute_purity(self):
283         majority_sum = 0
284         for cluster in self._clusters:
285             member_labels = [member._label for member in cluster._members]
286             max_count = max([member_labels.count(label) for label in range(20)])
287             majority_sum += max_count
288         return majority_sum * 1. / len(self._data)
```

$$\text{purity}(\Omega, \mathbb{C}) = \frac{1}{N} \sum_{k=1,K} \max_{j} |\omega_k \cap c_j|$$

Class Kmeans

- Đánh giá chất lượng phân cụm: Tính NMI

```
290  def compute_NMI(self):
291      I_value, H_omega, H_C, N = 0., 0., 0., len(self._data)
292      for cluster in self._clusters:
293          wk = len(cluster._members) * 1.
294          H_omega += - wk / N * np.log10(wk / N)
295          member_labels = [member._label
296                           for member in cluster._members]
297          for label in range(20):
298              wk_cj = member_labels.count(label) * 1.
299              cj = self._label_count[label]
300              I_value += wk_cj / N * \
301                         np.log10(N * wk_cj / (wk * cj) + 1e-12)
302      for label in range(20):
303          cj = self._label_count[label] * 1.
304          H_C += - cj / N * np.log10(cj / N)
305      return I_value * 2. / (H_omega + H_C)
```

$$NMI(\Omega,\mathbb{C}) = \frac{I(\Omega,\mathbb{C})}{[H(\Omega)+H(\mathbb{C})]/2}$$

$$I(\Omega,\mathbb{C}) = \sum_k \sum_j \frac{|\omega_k \cap c_j|}{N} \cdot \log_{10} \frac{N \cdot |\omega_k \cap c_j|}{|\omega_k| . |c_j|}$$

$$H(\Omega) = -\sum_k \frac{|\omega_k|}{N} \cdot \log_{10} \frac{|\omega_k|}{N}$$

$$H(\mathbb{C}) = -\sum_k \frac{|c_j|}{N} \cdot \log_{10} \frac{|c_j|}{N}$$

Class Kmeans

- Vấn đề khởi tạo tâm cụm:

> Kết quả của Kmeans phụ thuộc vào việc khởi tạo tâm cụm

=> Làm vài lần và chọn lấy lần tốt nhất

hoặc Khởi tạo theo chiến lược:

* Dùng Kmeans++

* Cluster center initialization algorithm for K-means clustering[*]

[*] https://www.researchgate.net/publication/223315329_Cluster_center_initialization_algorithm_for_K-means_clustering

Sử dụng Scikit-learn

- Nội dung chính:
  - > Kmeans
  - > SVMs:
    - * Linear SVMs
    - * kernel SVMs

Sử dụng Scikit-learn

- Kmeans:

```
63  def clustering_with_KMeans():
64      data, labels = load_data(data_path='../datasets/20news-bydate/20news-full-tfidf.txt')
65      # use csr_matrix to create a sparse matrix with efficient row slicing
66      from sklearn.cluster import KMeans
67      from scipy.sparse import csr_matrix
68      X = csr_matrix(data)
69      print '=========='
70      kmeans = KMeans(
71          n_clusters=20,
72          init='random',
73          n_init=5,  # number of time that kmeans runs with differently initialized centroids
74          tol=1e-3,  # threshold for acceptable minimum error decrease
75          random_state=2018  # set to get deterministic results
76      ).fit(X)
77      labels = kmeans.labels_
```

- Xem thêm: http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Sử dụng Scikit-learn

- SVMs: Linear SVMs

```
34  def classifying_with_linear_SVMs():
35      train_X, train_y = load_data(data_path='../datasets/20news-bydate/20news-train-tfidf.txt')
36      from sklearn.svm import LinearSVC
37      classifier = LinearSVC(
38          C=10.0,  # penalty coeff
39          tol=0.001,  # tolerance for stopping criteria
40          verbose=True # whether prints out logs or not
41      )
42      classifier.fit(train_X, train_y)
43
44      test_X, test_y = load_data(data_path='../datasets/20news-bydate/20news-test-tfidf.txt')
45      predicted_y = classifier.predict(test_X)
46      accuracy = compute_accuracy(predicted_y=predicted_y, expected_y=test_y)
47      print 'Accuracy:', accuracy
```

- Xem thêm: http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

Sử dụng Scikit-learn

- SVMs: Linear SVMs : Hàm compute_accuracy

```
28      def compute_accuracy(predicted_y, expected_y):
29          matches = np.equal(predicted_y, expected_y)
30          accuracy = np.sum(matches.astype(float)) / expected_y.size
31          return accuracy
```

Sử dụng Scikit-learn

- SVMs: Kernel SVMs:

```
60      classifier = SVC(
61          C=50.0,
62          kernel='rbf',  # 'linear', 'poly', 'rbf', 'sigmoid', precomputed'
63          gamma=0.1,
64          tol=0.001,
65          verbose=True
66      )
```

radial basis function (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \gamma > 0$

Chuẩn bị cho buổi tới

- Cài đặt tensorflow-GPU:
  https://www.youtube.com/watch?v=6iyweMKcX3w

Thank you