گزارش فاز سوم پروژه هوش محاسباتی

یوسف رضازاده
علیرضا امینی

۱ تقسیم داده آموزشی به ۵۰ کلاستر

ابتدا داده آموزشی را با استفاده از الگوریتم kmeans به ۵۰ خوشه تقسیم می کنیم.

```
kmeans = KMeans(n_clusters=50).fit(x_train)
```

۲ انتخاب n تا مرکز نزدیک تر به داده تست

در این تابع با داشتن خروجی خوشه بندی اولیه، مقدار k و داده تست، k تا مرکز نزدیکتر به داده تست را شناسایی و خوشه های منتسب به این مراکز را بازمی گرداند.
ابتدا مراکز خروجی خوشه بندی اولیه را به دست می آوریم. سپس فاصله اقلیدوسی این مراکز با داده تست را در ۵۱۲ بعد محاسبه می کنیم. k تا مرکز با کمترین فاصله را انتخاب می کنیم و آن ها را به عنوان ورودی به تابع $clustering.predict$ می دهیم تا خوشه منتسب به این مرکز را به عنوان خروجی بازگرداند.

```
def cluster_select(datapoint, clustering, k):
    cluster_centers = clustering.cluster_centers_
    distances = np.sum((datapoint - cluster_centers)**2, axis=1)
    distances_index = np.argsort(distances)
    k_nearest_clusters = clustering.predict(cluster_centers[distances_index[:k]])
    return k_nearest_clusters
```

۳ اجرای الگوریتم KNN

در این تابع با داشتن داده تست، داده های موجود در خوشه های انتخاب شده و لیبل آن ها و مقدار k ، الگوریتم KNN را اجرا می کنیم.
ابتدا فاصله اقلیدوسی داده های آموزشی انتخاب شده با داده تست را در ۵۱۲ بعد محاسبه می کنیم. سپس k تا داده با کمترین فاصله از داده تست را انتخاب و لیبل آن ها را می یابیم. درنهایت فراوانی لیبل های k تا داده انتخاب شده را به دست می آوریم. لیبل با بیشترین فراوانی به عنوان لیبل آن داده تست انتخاب می شود.

```python
def knn_test(test_datapoint, compare_datapoints, compare_labels, k):
    distances = np.sum((test_datapoint - compare_datapoints)**2, axis=1)
    distances_index = np.argsort(distances)
    k_nearest_neighbor = distances_index[:k]
    k_nearest_neighbor_labels = compare_labels[k_nearest_neighbor]
    counts = np.bincount(k_nearest_neighbor_labels)
    return np.argmax(counts)
```

بنابراین به ازای هر داده تست با استفاده از تابع $inference$ ابتدا n تا کلاستر از داده های آموزشی با کمترین فاصله از آن داده تست را به کمک تابع $cluster\_select$ به دست می آوریم. سپس با استفاده از کلاسترهای به دست آمده و به کمک تابع knn_test ، کلاسترهای انتخاب شده را به عنوان داده آموزشی در نظر می گیریم تا در نهایت لیبل پیش بینی شده را به ما خروجی بدهد.

```python
def inference(datapoint, train_datapoints, train_labels, clustering, k1, k2):
    cluster_labels = clustering.labels_
    cluster_index = cluster_select(datapoint, clustering, k1)
    cluster_mask = np.isin(cluster_labels, cluster_index)
    selected_data = train_datapoints[cluster_mask]
    selected_labels = train_labels[cluster_mask]
    label = knn_test(datapoint, selected_data, selected_labels, k2)

    return label

y_pred = []

for test_datapoint in x_test:
    y_pred.append(inference(test_datapoint, x_train, y_train, kmeans, 50, 19))

print(accuracy_score(y_test, np.array(y_pred)))
```

۴ پیدا کردن بهترین امتیاز

برای انتخاب مقدار n معیار ما انتخاب مقداری است که accuracy مدل بیشینه و مقدار پنالتی کمینه شود. بنابراین برای یافتن مقدار n ای که بیشترین مقدار امتیاز را به ما خروجی بدهد، مقادیر متفاوت $k1(n)$ از ۱ تا ۵۰ را با مقادیر مختلف $k2$ (اعداد فرد از ۱ تا ۱۵) تست کردیم. خروجی تمامی حالت ها در فایل $iml - phase - 3.log$ دخیره شده است.

```
clsuters_range = list(range(1,50))
knn_range = list(range(1,15,2))
scores = np.zeros((len(clsuters_range), len(knn_range)))

for i in range(len(clsuters_range)):
    for j in range(len(knn_range)):
        y_pred = []

        for test_datapoint in x_test:
            y_pred.append(inference(test_datapoint, x_train, y_train, kmeans, clsuters_range[i], knn_range[j]))

        scores[i][j] = accuracy_score(y_test, np.array(y_pred))*100 - clsuters_range[i]*0.2
```

همان طور که مشاهده می شود. بهترین امتیاز 84.1328، و با accuracy 84.9328 محاسبه شده است که پارامتر های آن $k1 = 4$ و $k2 = 5$ بوده اند.

همان طور که مشاهده می شود با افزایش مقدار n $(k1)$ در مقادیر n ابتدایی (مقدار n کوچک) دقت الگوریتم KNN بالاتر می رود و آن هم به این علت است که با توجه به اینکه دیتاست شامل ۱۰۰ کلاس است و ما دیتاست را به ۵۰ خوشه تقسیم بندی می کنیم، طبیعتا در هر کلاستر، داده های کلاس های مختلفی قرار می گیرد و بنابراین افزایش مقدار n، باعث می شود داده های کلاس درست در کنار یکدیگر تجمیع شوند و دقت مدل بالاتر برود. همچنین احتمال دارد به علت خوب و متمایزکننده نبودن ویژگی های استخراج شده از داده ها برای الگوریتم طبقه بند، داده های یک کلاس در کلاسترهای متفاوتی توزیع شده باشند و بنابراین هر چه تعداد کلاسترهای انتخاب شده بیشتر باشد، بیشتر می توان احتمال داد که از میان کلاسترهای انتخاب شده، داده های متعلق به کلاس تست به اندازه کافی وجود داشته باشند تا پیش بینی لیبل آن داده درست باشد.
از طرفی از یک جایی به بعد (مقدار n بزرگ) مقدار accuracy ثابت می ماند و دیگر روند صعودی ندارد و نشان دهنده این است که با افزایش تعداد کلاسترها نمی توان به داده بیشتری از کلاس هر داده تست رسید و داده هایی از کلاس های دیگر اضافه می شوند که تاثیری در افزایش دقت مدل نخواهند داشت. همچنین چون در این جا با افزایش مقدار n ، پنالتی منفی دریافت شده نیز افزایش یافته و از امتیاز کاهش می یابد و ملاک ما برای انتخاب پارامتر مناسب n ، امتیاز به دست آمده می باشد، با افزایش مقدار n نیز از یک جایی به بعد امتیاز کسب شده روند نزولی پیدا می کند. بنابراین باید مقدار بهینه ای برای این پارامتر پیدا کرد تا از کم شدن امتیاز جلوگیری کند.

بهترین مقدار انتخاب شده برای پارامتر n برابر با ۴ می باشد که نشان دهنده این است بردار ویژگی های استخراج شده برای هر عکس دارای ویژگی های متمایزکننده خوبی بوده است که با انتخاب تعداد کمی کلاستر از ۵۰تا کلاستر ابتدایی توانستیم دیتاهای آموزشی متعلق به کلاس داده تست را تا حد کافی انتخاب کنیم که لیبل درستی برای داده تست، پیش بینی شود.

۵ ارتباط بین لیبل های KNN و لیبل های کلاسترها

در این قسمت، می خواهیم ببینیم هر کلاستر انتخاب شده چه میزان در پیش بینی الگوریتم KNN کمک کننده بوده است. معیار ما در این حالت محاسبه خلوص هر کلاستر انتخاب شده و در نهایت کلاستر ترکیب شده، برای هر داده تست به ازای لیبل واقعی و لیبل پیش بینی شده آن می باشد.
بنابراین در تابع $calculate\_purity$ با دریافت لیبل واقعی و پیش بینی شده داده تست، کلاسترهای انتخاب شده برای آن داده و تمامی لیبل ها، ابتدا مقدار خلوص داده هایی با لیبل واقعی و لیبل پیش بینی شده را به ازای تمام ۴تا کلاستر انتخاب شده را محاسبه می کنیم. سپس این مقادیر را برای کلاستر ایجاد شده از ترکیب این ۴ کلاستر را نیز محاسبه می کنیم. برای استفاده از تابع $calculate\_purity$ ، تابع $inference$ را طوری تغییر می دهیم تا لیبل واقعی هر داده تست را نیز دریافت کند تا بتوانیم پس از دریافت خروجی کلاسترهای انتخاب شده در این تابع، با فراخوانی تابع $calculate\_purity$ ، خلوص لیبل واقعی و لیبل پیش بینی شده در کلاسترها را برای هر داده تست به دست آوریم.

```python
def calculate_purity(cluster_index, label, clustering, train_labels, true_label):
  cluster_labels = clustering.labels_
  for index in cluster_index:
    cluster_mask = np.isin(cluster_labels, index)
    cluster =  train_labels[cluster_mask]
    pred = train_labels[cluster_mask & (train_labels == label)]
    true = train_labels[cluster_mask & (train_labels == true_label)]
    pred_purity = len(pred) / len(cluster) * 100
    true_purity = len(true) / len(cluster) * 100
    print("Cluster {} : purity of the predicted label: {}".format(index, pred_purity))
    print("Cluster {} : purity of the true label: {}".format(index, true_purity))

  cluster_mask = np.isin(cluster_labels, cluster_index)
  cluster =  train_labels[cluster_mask]
  pred = train_labels[cluster_mask & (train_labels == label)]
  true = train_labels[cluster_mask & (train_labels == true_label)]
  pred_purity = len(pred) / len(cluster) * 100
  true_purity = len(true) / len(cluster) * 100
  print("Integrated Cluster : purity of the predicted label : {}".format(pred_purity))
  print("Integrated Cluster : purity of the true label: {}".format(true_purity))
```

برای بررسی میزان تاثیر کلاسترها روی لیبل های انتخاب شده توسط KNN، از هر دو گروه داده که مدل لیبل آن ها را درست و اشتباه پیش بینی کرده است، این تابع را فراخوانی می کنیم. خروجی ۲ مورد از این حالت ها بعه صورت زیر می باشد:

- تشخیص درست

```
Correctly Classified Data:
Cluster 24 : purity of the predicted label: 80.3921568627451
Cluster 24 : purity of the true label: 80.3921568627451
Cluster 49 : purity of the predicted label: 3.4482758620689653
Cluster 49 : purity of the true label: 3.4482758620689653
Cluster 17 : purity of the predicted label: 0.0
Cluster 17 : purity of the true label: 0.0
Cluster 22 : purity of the predicted label: 2.4096385542168677
Cluster 22 : purity of the true label: 2.4096385542168677
Integrated Cluster : purity of the predicted label : 16.888045540796963
Integrated Cluster : purity of the true label: 16.888045540796963
```

همان طور که مشاهده می شود، ۸۰ درصد کلاستر شماره ۲۴ شامل دیتاهایی با لیبل داده تست می باشد. بنابراین این کلاستر بهترین کلاستر انتخاب شده برای تشخیص لیبل این دیتا می باشد زیرا با انتخاب آن در الگوریتم KNN تعداد همسایه های بیشتری با لیبل مشابه با داده تست انتخاب می شود و لیبل آن درست پیش بینی می شود. کلاستر های شماره ۴۹ و ۲۲ نیز به ترتیب 3.44 و 2.40 درصد شامل داده هایی با لیبل داده تست می شوند که به نسبت کلاستر ۲۴، تاثیر بسیار کمتری در بهبود عملکرد الگوریتم KNN دارند و کلاستر شماره ۱۷ نیز شامل هیچ داده ای از کلاس داده تست نمی باشد. همچنین هنگام با ترکیب تمامی ۴ کلاستر انتخاب شده، درصد خلوص لیبل واقعی و پیش بینی شده این داده تست به مقدار 16.88 می رسد که آن نیز به این علت می باشد که چون به جز در کلاستر ۲۴، ۳ کلاستر دیگر شامل مقدار کمی از داده کلاس تست می شدند، با ترکیب این ۴ کلاستر درصد خلوص داده هایی با این لیبل کاهش می یابد ولی در مجموع چون شامل مقدار قابل قبولی از این داده ها می باشد و داده های این کلاس شامل بردارهای ویژگی های متمایز کننده خوبی در مقایسه با دیگر کلاس های موجود در این کلاستر می شوند، لیبل داده تست درست تشخیص داده شده است.

- تشخیص اشتباه

```
Misclassified Data:
Cluster 31 : purity of the predicted label: 1.4925373134328357
Cluster 31 : purity of the true label: 4.477611940298507
Cluster 22 : purity of the predicted label: 78.78787878787878
Cluster 22 : purity of the true label: 0.0
Cluster 21 : purity of the predicted label: 0.0
Cluster 21 : purity of the true label: 3.389830508474576
Cluster 48 : purity of the predicted label: 0.0
Cluster 48 : purity of the true label: 15.789473684210526
Integrated Cluster : purity of the predicted label : 11.48936170212766
Integrated Cluster : purity of the true label: 7.234042553191489
```

همان طور که مشاهده می شود، در کلاستر شماره ۳۱ درصد خلوص لیبل پیش بینی شده برابر 1.49 و درصد خلوص لیبل واقعی4.47 می باشد که از هر دو کلاس درصد کمی وجود دارد. در دو کلاستر شماره۲۱ و ۴۸، درصد خلوص کلاس لیبل واقعی به ترتیب 3.38 و 15.78 می باشد و از کلاس لیبل پیش بینی شده هیچ داده ای وجود ندارد. اما 78.78 کلاستر شماره ۲۲ شامل داده های کلاس لیبل پیش بینی شده می باشد و هیچ داده ای از کلاس متعلق به داده تست وجود ندارد. در نهایت با ترکیب این ۴ کلاستر درصد خلوص لیب پیش بینی شده برابر 11.49 و لیبل واقعی برابر 7.23 می باشد. مهم ترین دلیل تشخیص اشتباه الگوریتم KNN ، انتخاب کلاستر شماره ۲۲ است که شامل تعداد زیادی از داده های کلاس اشتباه هستند و همچنین در کلاسترهای دیگر نیز به نظر می آید مقدار کافی از داده های کلاس درست وجود ندارد. علت آن نیز این می تواند باشد ویژگی های استخراج شده برای داده های این کلاس متمایزکننده و منحصر به آن کلاس نیستند و با داده های کلاس لیبل پیش بینی شده اشتباه تشخیص داده می شوند و به همین علت مرکز کلاستر ۲۲ به عنوان مرکز نزدیک به این داده تست انتخاب شده است. همچنین با وجود اینکه از هر دو کلاس نسبت های تقریبا نزدیک به همی در کلاستر نهایی وجود دارد اما به علت نزدیکی فیچرهای کلاس درست به فیچرهای کلاس پیش بینی شده، لیبل داده تست اشباه تشخیص داده شده است.

۶ ارزیابی K-means

در نهایت دو معیار purity و rand index برای ارزیابی خوشه بندی اولیه داده های اعمال شد که برای معیار rand index از تابع $rand\_score$ از تابخانه sklearn استفاده شد تا با دریافت لیبل های داده های آموزشی و لیبل های خوشه های پیش بینی شده بتواند این مقدار این معیار را خروجی بدهد. همچنین برای معیار purity از تابع $purity\_score$ استفاده شد که با دریافت لیبل های داده های آموزشی و لیبل های خوشه ها، پرتکرار ترین لیبل ها در هر خوشه را شناسایی می کنیم. در نهایت برای هر داده، بررسی می کنیم که آیا لیبل واقعی آن با لیبل غالب خوشه‌ای که به آن اختصاص داده شده مطابقت دارد یا خیر. مجموع تطابق ها را محاسبه و بر تعداد کل داده های آموزشی تقسیم می کنیم تا مقدار purity به دست آید.

```python
def purity_score(y_true, y_pred):
    cluster_labels = np.unique(y_pred)
    max_labels = np.zeros_like(cluster_labels)
    for i in range(len(cluster_labels)):
        mask = (y_pred == cluster_labels[i])
        max_labels[i] = np.argmax(np.bincount(y_true[mask]))
    purity = np.sum(y_true == max_labels[y_pred]) / len(y_true)
    return purity
```

خروجی ارزیابی خوشه بندی صورت گرفته با معیارهای rand index و purity به صورت زیر می باشد:

```python
# rand index
y_pred = kmeans.labels_
rand_index = rand_score(y_train, y_pred)
print("Rand Index: {}".format(rand_index))

# purity
purity = purity_score(y_train, y_pred)
print("Purity: {}".format(purity))
```

```
Rand Index: 0.9801593889790748
Purity: 0.5288226673180264
```