Кластерлеу Алгоритмдері, Методологиялары және Өнімділік Метрикалары

Бұл презентация IT-мамандары мен деректер ғалымдарына арналған. Біз кластерлеу алгоритмдерінің тереңдігіне, оларды бағдарламалық қамтамасыз етуде іске асыру әдістеріне (мысалы, `scikit-learn`, `pandas`, `numpy` кітапханаларын қолдану арқылы) және олардың өнімділігін бағалаудың техникалық метрикаларына тоқталамыз. Сондай-ақ, нақты әзірлеу кейстеріндегі практикалық қолданыстарды қарастырамыз.

Оқытушы: Сарсембаева Т.С.

Кіріспе: Кластерлеудің Техникалық Негіздері

Дәрістің Техникалық Мақсаты

Бұл модуль IT-мамандары мен деректер ғалымдарына кластерлеудің мәнін, оның ішкі алгоритмдерін (K-Means, DBSCAN, Hierarchical Clustering), scikit-learn кітапханасы арқылы жүзеге асыру ерекшеліктерін және кластерлеу модельдерінің өнімділігін бағалау метрикаларын (Silhouette Score, Davies-Bouldin Index, Adjusted Rand Index) терең түсіндіруге бағытталған. Сондай-ақ, pandas және numpy арқылы деректерді дайындау мен трансформациялау мәселелері қамтылады.

Кластерлеудің Программалық Концепциясы

Кластерлеу – бақылаусыз машиналық оқытудың (Unsupervised Machine Learning) іргелі әдісі, деректер жиынындағы құрылымдық ұқсастықтар негізінде объектілерді логикалық топтарға (кластерлерге) бөлуге арналған. Мақсат – бір кластердегі деректер нүктелерінің бір-біріне ұқсас, ал әртүрлі кластерлердегі нүктелердің айтарлықтай ерекшеленуін қамтамасыз ету. Бұл деректерді барлау, аномалияны анықтау және сегменттеу сияқты көптеген қолданбаларда маңызды.

Кластерлеудің Негізгі Принциптері: Техникалық Түсініктеме

Ішкі-кластерлік Когезияны Арттыру

Бұл принцип деректер нүктелерінің бір кластер ішінде бір-біріне тығыз орналасуын талап етеді. Математикалық тұрғыдан, әр кластердегі нүктелердің кластер центроидынан (немесе медианынан) орташа қашықтығын немесе **ішкі-кластерлік дисперсияны (intra-cluster variance)** минимизациялауға бағытталған. Мысалы, k-means алгоритмі Sum of Squared Errors (SSE) немесе inertia метрикасын минимизациялау арқылы осыған қол жеткізеді.

Кластераралық Айырмашылықты Максимизациялау

Бұл қағида әр түрлі кластерлердің бір-бірінен айтарлықтай алшақ орналасуын білдіреді. Басқаша айтқанда, **кластераралық қашықтықты (inter-cluster distance)** арттыру қажет. Бұл **төмен байланыс (low coupling)** идеалына сәйкес келеді, мұнда кластерлер арасындағы шекаралар анық болады. DBSCAN сияқты алгоритмдер тығыздықты пайдалана отырып, осыған қол жеткізеді, ал k-means центроидтар арасындағы қашықтықты жанама түрде арттыруға тырысады.

Кластерлеудің Қолданылуы және Түрлері (IT-контексте)

Клиент Сегменттеу

K-means немесе иерархиялық кластерлеу арқылы клиенттік деректерді талдап, әртүрлі тұтынушы топтарын анықтау, жекелендірілген өнім ұсыныстарын жасау.

Биоинформатика

Ген экспрессиясының деректерін, ақуыздардың құрылымдық ұқсастықтарын және жасуша типтерін жіктеу үшін кластерлеу алгоритмдерін қолдану.

Аномалияларды Анықтау

Қаржылық транзакциялардағы немесе желілік трафиктегі әдеттен тыс үлгілерді, яғни алаяқтық немесе кибершабуыл әрекеттерін DBSCAN немесе Isolation Forest көмегімен анықтау.

Деректердің Кеңістіктік Талдауы

Геокеңістіктік деректердегі үлгілерді, мысалы, қалалық аудандарды ұқсастығы бойынша топтастыру, ұялы желілердің тиімді орналасуын жоспарлау.

Кластерлеу әдістерінің негізгі түрлері:

- **Бөлуші әдістер** (k-means, k-medoids): Деректерді белгілі бір кластер санына бөледі. Мысалы, scikit-learn кітапханасындағы KMeans модулі жиі қолданылады.
- **Тығыздыққа негізделген әдістер** (DBSCAN, OPTICS): Кластерлерді тығыздығы жоғары аймақтар ретінде анықтайды және шуды (аномалияларды) бөлектейді.
- **Иерархиялық әдістер** (Агломеративті, Дивизивті): Деректер арасындағы иерархиялық байланыстарды көрсететін кластерлер құрылымын қалыптастырады.

k-means (К-Орташа) Алгоритмі: Тереңдетілген Талдау

Алгоритмнің Негізгі Принциптері

k-means — бұл деректерді алдын ала белгіленген **K** кластерге бөлуге бағытталған, кеңінен қолданылатын бөлуші кластерлеу алгоритмі. Оның мақсаты — әр кластердегі деректер нүктелерінің өз центроидтарына дейінгі қашықтықтарының квадраттарының қосындысын (Within-Cluster Sum of Squares, WCSS) минималдау.

Алгоритм Қадамдары (Псевдокод)

1. K (кластер саны) мәнін таңдау.
2. Бастапқы K центроидты кездейсоқ инициализациялау (мысалы, деректер жинағынан кездейсоқ K нүкте таңдау).
3. Итерацияларды бастау:
   a. Тағайындау қадамы: Әрбір деректер нүктесін ең жақын центроидқа тағайындау (мысалы, Евклидтік қашықтық бойынша).
   b. Жаңарту қадамы: Әрбір кластердің жаңа центроидын сол кластерге тағайындалған барлық нүктелердің орташа мәні ретінде есептеу.
4. Егер центроидтардың позициясы өзгермесе немесе максималды итерация санына жетсе, алгоритмді тоқтату. Әйтпесе, 3-қадамға оралу.

Евклидтік Қашықтық Формуласы

Деректер нүктесі x_i мен центроид c_j арасындағы қашықтықты есептеу үшін:

$$d(x_i, c_j) = \sqrt{\sum_{l=1}^{m} (x_{il} - c_{jl})^2}$$

Мұндағы m — өлшемдер саны, x_il — i-ші нүктенің l-ші өлшемдегі мәні, c_jl — j-ші центроидтың l-ші өлшемдегі мәні.

k-means Алгоритмінің Кемшіліктері

Алдын ала K мәнін таңдау

K-means алгоритмі кластерлер саны K алдын ала белгілеуді талап етеді. Бұл мәнді таңдау көбінесе elbow method немесе silhouette score сияқты эвристикалық әдістер арқылы жүзеге асырылады, бірақ олар деректерге байланысты күрделілік тудыруы мүмкін. Дұрыс K мәнін анықтау алгоритмнің тиімділігіне тікелей әсер етеді.

Бастапқы Центроидтарға Сезімталдық

Алгоритм бастапқы центроидтарды кездейсоқ таңдауға негізделген. Бұл таңдаудың әртүрлі болуы кластерлеудің әртүрлі нәтижелеріне және локальды минимумдарға әкелуі мүмкін. scikit-learn кітапханасында бұл мәселені n_init параметрі арқылы бірнеше рет инициализациялау арқылы шешуге тырысады (kmeans++ әдісі).

Дөңгелек (Сфералық) Кластер Пішіндері

K-means алгоритмі тек дөңгелек немесе сфералық пішінді кластерлерді тиімді таба алады. Егер деректерде сопақша, күрделі немесе бір-бірімен қиылысқан пішінді кластерлер болса, k-means дұрыс бөлу нәтижесін бермейді. Мұндай жағдайларда DBSCAN немесе Gaussian Mixture Models сияқты алгоритмдер тиімдірек болуы мүмкін.

Шу мен Аномалияларға Сезімталдық

K-means шулы деректерге (noise) және аномалияларға (outliers) өте сезімтал. Себебі, орташа мән центроидтары аномалияларға қарай ығысып, кластерлердің шекарасын бұрмалауы мүмкін. Бұл деректерді алдын ала тазалауды немесе шуға төзімдірек кластерлеу алгоритмдерін қолдануды қажет етеді.

Алдын ала Деректерді Өңдеу

Алгоритм евклидтік қашықтыққа негізделгендіктен, ерекшеліктердің (features) масштабының әртүрлілігі нәтижеге қатты әсер етеді. Сондықтан кластерлеу алдында деректерді стандарттау (StandardScaler) немесе қалыпқа келтіру (MinMaxScaler) міндетті болып табылады. Бұл қадамсыз, үлкен мәнді ерекшеліктер доминантты болып, кластерлеудің дұрыстығын төмендетеді.

DBSCAN Алгоритмінің Қағидасы

DBSCAN – Тығыздыққа Негізделген Кластерлеу

Тығыздыққа негізделген кеңістіктік кластерлеу алгоритмі (**Density-Based Spatial Clustering of Applications with Noise**). Ол деректер жиынындағы тығыздығы жоғары аймақтарды кластерлер ретінде анықтайды, ал тығыздығы төмен аймақтарды шу (**noise**) ретінде жіктейді. Бұл алгоритм күрделі пішінді кластерлерді (**non-convex shapes**) тиімді анықтай алады.

Негізгі Параметрлер

- Epsilon (\epsilon **немесе** eps)**:** Берілген нүктенің көршілес аймағын анықтайтын максималды радиус. Бұл параметрдің шамасы кластерлердің тығыздығына тікелей әсер етеді.
- Minimum Points (MinPts)**:** \epsilon радиусы ішінде болуы тиіс нүктелердің минималды саны. Бұл сан кластердің «тығыздығын» анықтайды және шуды бөлуге көмектеседі.

Нүктелердің Типтері

- **Ядролық нүкте (**Core Point**):** Егер нүктеге \epsilon радиусы ішінде MinPts-тен кем емес басқа нүктелер (өзін қоса алғанда) кіретін болса.
- **Шекаралық нүкте (**Border Point**):** Ядролық нүкте емес, бірақ қандай да бір ядролық нүктенің \epsilon радиусы ішінде орналасқан нүкте.
- **Шу нүктесі (**Noise Point**):** Ядролық та, шекаралық та емес нүкте. Ол ешбір кластерге жатпайды (аномалия немесе **outlie**).

Кластер Құру Механизмі

DBSCAN алгоритмі кездейсоқ таңдалған ядролық нүктеден басталып, "тығыздық бойынша қолжетімділік" (**density-reachability**) принципі арқылы кластерді кеңейтеді. Егер B нүктесі A ядролық нүктесінен тығыздық бойынша тікелей қолжетімді болса (\epsilon радиусында және MinPts шартына сай), онда олар бір кластерге жатады. Бұл процесс барлық тығыздық бойынша қолжетімді нүктелер қамтылғанша жалғасады.

Scikit-learn арқылы іске асыру

```
from sklearn.cluster import DBSCAN
import numpy as np
import pandas as pd
# Деректерді генерациялау (мысал үшін)
X = np.array([[1, 2], [1.5, 1.8], [5, 8],
              [8, 8], [1, 0.6], [9, 11],
              [8, 2], [10, 2], [9, 3]])
# DBSCAN моделін инициализациялау және қолдану
dbscan = DBSCAN(eps=2, min_samples=2)
clusters = dbscan.fit_predict(X)
print(f"Кластер белгілері: {clusters}")
# Нәтижеде -1 шу нүктелерін білдіреді
# (мысалы, [0, 0, 1, 1, 0, 2, 1, 1, 1] сияқты)
# Кластерленген деректерді DataFrame-ге біріктіру
df = pd.DataFrame(X, columns=['Feature1', 'Feature2'])
df['Cluster'] = clusters    print(df)
```

DBSCAN Артықшылықтары: IT-практикадағы маңызы

Кез келген пішіндегі кластерлерді анықтау

DBSCAN тығыздыққа негізделген алгоритм ретінде, сфералық емес немесе күрделі геометриялық пішіндегі кластерлерді (мысалы, жарты ай, концентрлі шеңберлер) тиімді анықтай алады. Бұл K-Means сияқты центроидқа негізделген алгоритмдердің шектеулерін айналып өтеді, олар тек дөңгелек кластерлерді жақсы табады.

```
# K-Means-тің шектеулілігіне мысал
from sklearn.datasets import make_moons
X, y = make_moons(n_samples=200, noise=0.05, random_state=0)
# DBSCAN мұндай деректерде K-Means-ке қарағанда әлдеқайда жақсы нәтиже береді.
```

Шуды (аномалияларды) тиімді өңдеу

Деректердегі шу нүктелерін (аутлайерлерді) автоматты түрде анықтайды және оларды ешбір кластерге жатқызбайды. Бұл деректерді алдын ала тазалау қажеттілігін азайтады және алгоритмнің беріктігін арттырады. Әсіресе желілік шабуылдарды, сенсорлық қателерді немесе транзакциялық аномалияларды анықтау сияқты мәселелерде пайдалы.

```
# DBSCAN шуды қалай өңдейді
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5)
clusters = dbscan.fit_predict(data)
noise_points = data[clusters == -1] # Шу нүктелері -1 ретінде белгіленеді
```

Кластер санын (K) алдын ала көрсету қажет емес

K-Means немесе Gaussian Mixture Models сияқты алгоритмдерден айырмашылығы, DBSCAN кластерлердің санын алдын ала беруді талап етпейді. Оның орнына, ол деректердің тығыздығына сүйеніп кластерлерді динамикалық түрде табады. Бұл кластерлердің нақты санын білмейтін деректер жиындары үшін өте қолайлы.

```
# scikit-learn-дегі DBSCAN API
# Кластер санын көрсетудің қажеті жоқ
from sklearn.cluster import DBSCAN
model = DBSCAN(eps=0.3, min_samples=10)
# model.fit(X) немесе model.fit_predict(X)
```

Жоғары қолданылу аймақтары мен масштабталуы

Географиялық ақпараттық жүйелерде (GIS) ыстық нүктелерді (мысалы, қылмыс немесе трафик тығыздығы), киберқауіпсіздікте аномалияларды анықтауда (мысалы, DDoS шабуылдары), өндірістегі ақауларды табуда және тұтынушы сегментациясында кеңінен қолданылады. Үлкен деректер жиындарымен де жақсы жұмыс істейді.

```
# Pandas және NumPy-мен деректерді өңдеу
import pandas as pd
import numpy as np
data = pd.read_csv('geo_data.csv')
dbscan = DBSCAN(eps=0.01, min_samples=5)
data['cluster'] = dbscan.fit_predict(data[['latitude', 'longitude']])
```

Кластерлеу Сапасын Бағалау: Ішкі Метрикалар

Бағалаудың күрделілігі

Бақылаусыз оқыту (**Unsupervised Learning**) моделі ретінде кластерлеу алгоритмдерінің сапасын бағалау, белгіленген деректерді (**labeled data**) қолданатын классификацияға қарағанда күрделірек. Себебі "шындық" белгілері (**ground truth labels**) болмайды.

Ішкі метрикалар (Intrinsic Metrics)

Ішкі метрикалар кластерлеу нәтижесінің сапасын бағалау үшін тек деректердің ішкі құрылымына (ішкі тығыздық және кластерлер арасындағы алшақтық) сүйенеді. Олар кластерлердің **тығыздығын (compactness)** және **бөлінуін (separation)** сандық түрде өлшейді.

Инерция (Inertia - k-means үшін)

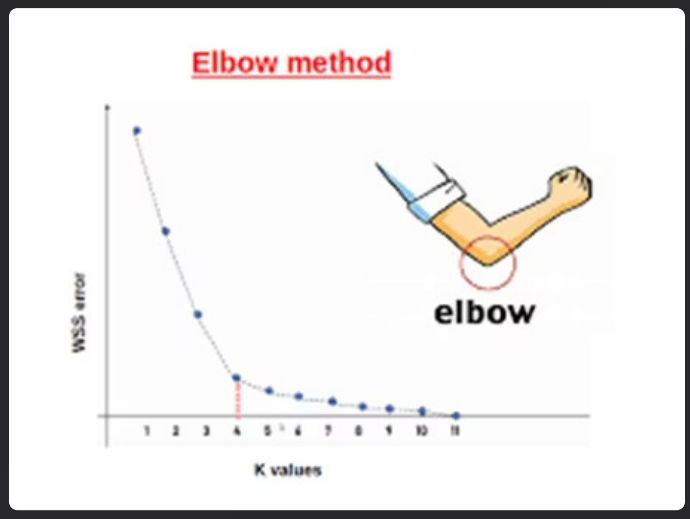
K-means алгоритмінде **инерция** кластердегі әрбір нүктенің өз центроидына (орталық нүктесіне) дейінгі қашықтықтарының квадраттарының қосындысын білдіреді. Алгоритм осы мәнді минималдандыруға тырысады. Төмен инерция мәні кластерлердің тығыздығын көрсетеді.

```
# Python scikit-learn мысалы
from sklearn.cluster import KMeans
import numpy as np

X = np.random.rand(100, 2) # Мысал деректер
kmeans = KMeans(n_clusters=3, random_state=0, n_init=10)
kmeans.fit(X)
inertia_value = kmeans.inertia_
print(f"Инерция мәні: {inertia_value}")
```

Шынтақ Әдісі (Elbow Method)

Оңтайлы K кластер санын анықтау үшін жиі қолданылатын әдіс. Әртүрлі K мәндері үшін инерцияны есептеп, график тұрғызады. График "шынтаққа" ұқсас ең үлкен бұрылыс нүктесін таңдау арқылы оңтайлы K мәнін көрсетеді, бұл инерцияның төмендеуінің айтарлықтай бәсеңдеуін білдіреді.



7.5. Силуэт Коэффициенті (Silhouette Coefficient)

Силуэт коэффициенті (немесе Силуэт көрсеткіші) кластер ішіндегі ұқсастықты (**cohesion**) және кластерлер арасындағы айырмашылықты (**separation**) біріктіреді. Ол әрбір деректер нүктесі үшін есептеліп, барлық нүктелер бойынша орташа мән алынады. Мән [-1, 1] аралығында болады:

- 1-ге жақын: Нүкте өз кластерімен өте тығыз және басқа кластерлерден жақсы бөлінген. Бұл жақсы кластерлеуді көрсетеді.
- 0-ге жақын: Нүкте екі кластер арасындағы шекарада орналасқан, бұл кластерлердің нашар бөлінгенін білдіреді.
- -1-ге жақын: Нүкте дұрыс емес кластерге тағайындалған.

```
# Python scikit-learn мысалы
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans

# X - деректер жиыны, labels - кластер белгілері
# k_means_model = KMeans(...)
# k_means_model.fit(X)
# labels = k_means_model.labels_

# silhouette_score = silhouette_score(X, labels)
# print(f"Силуэт көрсеткіші: {silhouette_score}")
```

8. Кластерлеуді Бағалау: Сыртқы Метрикалар және Алгоритмдерді Салыстыру

Сыртқы Бағалау Метрикалары

Бұл метрикалар кластерлеу нәтижелерін деректерде бар "шындық" сыныптық жапсырмалармен (Ground Truth) салыстыру үшін қолданылады. Олардың қолданылуы scikit-learn кітапханасында стандартталған.

Анықталған Реттелген Индекс (Adjusted Rand Index - ARI**)**

Кластерлеу бөлінуі мен шынайы сыныптардың бөлінуі арасындағы ұқсастықты өлшейді, кездейсоқ тағайындаулар үшін түзетілген. Мән [-1, 1] аралығында болады, мұнда 1 толық сәйкестікті білдіреді.

```
from sklearn.metrics import adjusted_rand_score
y_true = [0, 0, 0, 1, 1, 1]
y_pred = [0, 0, 1, 1, 2, 2] # Мысал кластерлер
ari_score = adjusted_rand_score(y_true, y_pred)
print(f"ARI: {ari_score:.2f}")
```

Бір-біріне ұқсас балл (Homogeneity, Completeness, V-measure**)**

- **Homogeneity:** Әр кластердің негізінен бір ғана шынайы сыныптың мүшелерінен тұру дәрежесін өлшейді.
- **Completeness:** Бір шынайы сыныптың барлық мүшелерінің бір кластерге жиналу дәрежесін өлшейді.
- V-measure бұл екеуінің гармониялық орташа мәні болып табылады.

```
from sklearn.metrics import homogeneity_score, completeness_score,
v_measure_score
y_true = [0, 0, 0, 1, 1, 1]
y_pred = [0, 0, 1, 1, 2, 2]
homogeneity = homogeneity_score(y_true, y_pred)
completeness = completeness_score(y_true, y_pred)
v_measure = v_measure_score(y_true, y_pred)
print(f"Homogeneity: {homogeneity:.2f}")
print(f"Completeness: {completeness:.2f}")
print(f"V-measure: {v_measure:.2f}")
```

Кластерлеу Алгоритмдерін Салыстыру: Техникалық Нюанстар

- k-means**:** Жылдам және интуитивті, үлкен деректер жиынтығы үшін тиімді. Бірақ кластерлердің K санына өте сезімтал және дөңгелек пішінді кластерлерді жақсы табады. Бастапқы центроидтарды таңдауға (мысалы, k-means++ арқылы) және орташа квадраттық қашықтықтар сомасын минимизациялауға бағытталған.
- DBSCAN**:** Тығыздыққа негізделген алгоритм, кез келген пішінді кластерлерді таба алады және шуды өңдей алады. Алайда, екі негізгі параметрі – eps (көршілес радиусы) және MinPts (минималды нүктелер саны) таңдау өте күрделі және деректердің тығыздығына байланысты.
- **Бағалау Метрикалары:** Silhouette Score (ішкі), Adjusted Rand Index, Homogeneity, Completeness (сыртқы) сияқты метрикалар кластерлеу моделінің тиімділігін объективті түрде бағалауға мүмкіндік береді. Оларды әртүрлі параметрлермен эксперимент жасау кезінде ең жақсы модельді таңдау үшін қолданады.

```
# k-means және DBSCAN параметрлерін таңдау
from sklearn.cluster import KMeans, DBSCAN
from sklearn.datasets import make_blobs

# Жасанды деректер жиынтығын жасау
X, y_true = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)

# k-means мысалы
kmeans = KMeans(n_clusters=4, random_state=0, n_init=10)
kmeans_labels = kmeans.fit_predict(X)

# DBSCAN мысалы
dbscan = DBSCAN(eps=0.5, min_samples=5) # eps және min_samples таңдау маңызды
dbscan_labels = dbscan.fit_predict(X)

print(f"k-means ARI: {adjusted_rand_score(y_true, kmeans_labels):.2f}")
print(f"DBSCAN ARI: {adjusted_rand_score(y_true, dbscan_labels):.2f}")
```

> ### Бақылау Сұрақтары
>
> - k-means пен DBSCAN алгоритмдерінің архитектуралық және қолдану аймақтарындағы негізгі айырмашылықтарын сипаттаңыз.
> - scikit-learn-да Homogeneity және Completeness метрикаларын қалай интерпретациялауға болады және олардың V-measure-мен байланысы қандай?
> - DBSCAN алгоритміндегі eps және MinPts параметрлерін таңдау деректердің кластерлеу нәтижелеріне қалай әсер етеді?