

تمرین سری سه
درس شبکه‌های عصبی

فرهاد دلیرانی

۹۶۱۳۱۱۲۵

dalirani@aut.ac.ir
dalirani.1373@gmail.com

فهرست

ابزارهای استفاده شده.....	۱
توضیح کد.....	۲
بررسی درستی کد SOM با استفاده از دیتاست IRIS و مقایسه‌ی عملکرد SOM با Kmeans.....	۹
عنوان: معماری های مختلف (یک بعدی، دو بعدی و سه بعدی) را بر اساس معیارهای ارزیابی مقایسه کرده و نتایج را تحلیل کنید.....	۱۰
بررسی شبکه‌های یک بعدی مختلف.....	۱۰
شرایط آموزش برای شبکه‌های یک بعدی.....	۱۰
نتیجه‌ی آموزش برای شبکه‌های یک بعدی.....	۱۴
نتیجه‌گیری برای شبکه‌های یک بعدی.....	۱۷
بررسی شبکه‌های دو بعدی مختلف.....	۱۸
شرایط آموزش برای شبکه‌های دو بعدی.....	۱۸
نتیجه‌ی آموزش برای شبکه‌های دو بعدی.....	۲۳
نتیجه‌گیری برای شبکه‌های دو بعدی.....	۲۶
بررسی شبکه‌های سه بعدی مختلف.....	۲۸
شرایط آموزش برای شبکه‌های سه بعدی.....	۲۸
نتیجه‌ی آموزش برای شبکه‌های دو بعدی.....	۳۳
نتیجه‌گیری برای شبکه‌های سه بعدی.....	۳۷
نتیجه‌گیری کلی.....	۳۷
عنوان: نتایج خوشه بندی حاصل از این شبکه و خوشه بندی kmeans را مقایسه نمایید.....	۳۹
شرایط آزمایش.....	۳۹
نتیجه‌ی انجام آزمایش.....	۳۹
نتیجه‌گیری.....	۴۰

ابزارهای استفاده شده


زبان برنامه نویسی: پایتون ۳,۶

محیط توسعه: PyCharm

سیستم عامل: ویندوز ۱۰

برای کاهش حجم فایل ارسالی دیتاست از فایل‌ها حذف شده است، که باید قبل از اجرا فایل در پوشه‌ی **dataset** قرار بگیرد.

Dropbox > codes > ANN > Assignment 3 > dataset

^	Name	Date modified	Type
	 20news-18828	5/1/2018 11:48 PM	File folder

توضیح کد

در فایل `create_clean_dataset.py` تابعی با همین نام قرار دارد.

```
def create_clean_dataset(max_features):  
    """  
    Create clean data set of 20 news data set, it uses stemming, eliminating stop words and ...  
    :param max_features:  
    :return:  
    """
```

در این تابع خبرهای مختلف دسته‌های متفاوت را از فایل می‌خوانم، سپس برای بهتر کردن دیتاست این کارها را انجام می‌دهم:

- کلماتی که شامل کدهای اسکی غیر از عددها و حرف‌ها هستند را حذف می‌کنم.
- تمام کلمات را به صورت حرف‌های کوچک در می‌آورم.
- Stop words ها را حذف می‌کنم.
- کلمات را به ریشه‌ی اصلیشان بر می‌گردانم، به طور مثال `opened` تبدیل به `open` می‌شود.

سپس با استفاده از تابع `TfidfVectorizer` کلماتی که در بیش از 30 درصد خبرها آمده است را حذف می‌کنم.

در نهایت به تعداد آرگمان `max features` از کلمات را نگه می‌دارم. البته در کد هنگام فراخوانی این تابع، این آرگمان را برابر با `None` قرار می‌دهم و تعداد فیچرها را کم نمی‌کنم. در انتها کل دیتاست را با استفاده از تابع `TfidfVectorizer` با دیکشنری که این تابع می‌سازد را از حالت کلمات به عددها در می‌آورم. هر خبر یک آرایه هم اندازه‌ی دیکشنری می‌شود که در صورتی که کلمه `i` ام از دیکشنری در آن خبر وجود داشته باشد خانه‌ی `i` ام برابر با عددی غیر از صفر می‌شود.

در فایل `read_dataset.py` تابعی با همین نام قرار دارد.

```
def read_dataset(train_ratio=0.80, valid_ratio=0.10, test_ratio=0.10, random_state=None, max_features=None):  
    """  
    Read 20 news category dataset. split it in train, validation and test  
    :param train_ratio: portion of dataset that should be assigned to training  
    :param valid_ratio: portion of dataset that should be assigned to validation  
    :param test_ratio: portion of dataset that should be assigned to testing  
    :param random_state: for controlling generating random number and making function reproducible  
    :return: trainX, validX, testX, trainY, validY, testY  
  
    X format : ndarray [[observation1],[observation2],...,[observationN]]  
    Y format : ndarray [[label observation1],[label observation2],...,[label observationN]]  
    """
```

این تابع از تابع `create_clean_dataset` استفاده می‌کند و دیتاست را از آن می‌گیرد. سپس در صورتی که تعداد کلمات دیکشنری بیش از ۸۰۰ تا باشد بر روی آن PCA انجام می‌دهد و تعداد فیچرها را به ۸۰۰ عدد می‌رساند. در انتها دیتاست را بر اساس آرگمان‌های ورودی به مجموعه‌ی آموزش، ارزیابی و تست تقسیم می‌کند.

فایل `measures.py` حاوی سه تابع زیر است:

```
def purity_measure(clusters, classes):  
    """  
    This function calculate the purity for given cluster and real classes value  
    :param clusters: the cluster assignments array  
    :param classes: the ground truth classes  
    :returns: the purity score  
    """
```

این تابع معیار `purity` را محاسبه می‌کند.

```
def rand_index(clusters, classes):  
    """  
    Rand Index Measure  
    :param clusters: the cluster assignments array  
    :param classes: the ground truth classes  
    :return: the RI score  
    """
```

این تابع معیار `rand index` را محاسبه می‌کند.

```
def f_measure(clusters, classes):  
    """  
    f Measure  
    :param clusters: the cluster assignments array  
    :param classes: the ground truth classes  
    :return: the f score  
    """
```

این تابع معیار `f` را محاسبه می‌کند.

در فایل `SOM.py` کلاس شبکه‌ی SOM قرار دارد. در زیر کلاس SOM و متدهایش را مشاهده می‌کنید:

```

class SOM:
    # constructor
    def __init__(self, shape, number_of_feature, samples_for_init,
                  distance_measure_str='cosine', topology='cubic',
                  init_learning_rate=0.01, max_epoch=200):...

    def topology_cubic(self, indeces, r):...

    def topology_hex(self, indeces, r):...

    def unravel_index(self, index):...

    def ravel_index(self, indices):...

    def learning_rate(self, learning_rate_0, previous_learning_rate, cur_iter, final_iter):...

    def neighbourhood_size(self, nb_0, previous_nb, cur_iter, final_iter):...

    def neighbour_strength(self, std, index_winner, index_neighbour):...

    def winner_neuron(self, x):...

    def fit(self, X):...

    def predict(self, X):...

    def write_in_file(self):...

    def load_from_file(self, file_name):...

```

در ادامه متدهای این کلاس را توضیح می‌دهم.

```

# constructor
def __init__(self, shape, number_of_feature, samples_for_init,
              distance_measure_str='cosine', topology='cubic',
              init_learning_rate=0.01, max_epoch=200):
    """
    This is constructor of self-organized map.
    :param shape: shape of network, should be in form (a, b, c)
    :param number_of_feature: number of feature of input vector
    :param distance_measure_str: kind of distance measure, euclidean, manhattan,...
    :param topology: different topology for neighbours, line, square, Hexadecimal, circle
    :param init_learning_rate: initial learning rate
    :param max_epoch: maximum number of epochs that allowed to train network
    :param samples_for_init: it contains n=shape[0]*shape[1]*shape[2] different samples
                           for initializing SOM
    :return:
    """

```

این تابع، سازنده‌ی کلاس SOM است. این تابع ساختار شبکه، تعداد فیچرهای هر خبر، نمونه‌های تصادفی انتخاب شده به عنوان وزن اولیه‌ی نرون‌ها، نوع فاصله، نوع همسایگی، ضریب یادگیری اولیه و حداکثر اپاک‌ها را می‌گیرد. و پارامترهای مختلف شبکه را بر اساس این آرگمان‌ها تنظیم می‌کند.

```
def topology_cubic(self, indeces, r):
    """
    return indeces of neuron that are neighbour of a neuron that is located
    in networks_structure[indeces]. neighbourhood is a cube with radius r.
    in 1-D network it is line
    in 2-D network it is square
    in 3-d network it is cube
    :param indeces: in form of (a, b, c)
    :return: return indeces of neighbours in form
    """
    return [[nb1-a, nb1-b, nb1-c],
            [nb2-a, nb2-b, nb2-c],
            ...,
            [nbN-a, nbN-b, nbN-c]]
```

این تابع یک همسایگی از نوع خط/مربع/مکعب به شعاع همسایگی ۲ را برمی‌گرداند. در صورتی که شبکه یک بعدی باشد این همسایگی یک خط است، در صورتی که ساختار شبکه دو بعدی باشد این همسایگی یک مربع است و در صورتی که ساختار شبکه ۳ بعدی باشد، این همسایگی از نوع مکعب است.

```
def unravel_index(self, index):
    """
    This function gets an index in one dimensional array and
    returns equivalent index in networks shape. network is
    1d, 2d or 3d matrix.
    :param index: index in one dimensional array
    :return: index in matrix is size= (network shape)
    """
```

این تابع اندیس در مختصات یک بعدی را می‌گیرد و اندیس نرون در ساختار شبکه را برمی‌گرداند.

```
def ravel_index(self, indices):
    """
    This function get an index in form (a, b, c) then it return:
    equivalent position in 1d
    :param indices:
    :return:
    """
```

این تابع اندیس نرون در ساختار شبکه را می‌گیرد و یک اندیس در مختصات یک بعدی بر می‌گرداند.

```
def learning_rate(self, learning_rate_0, previous_learning_rate, cur_iter, final_iter):
    """
    exponential learning rate decay
    :param learning_rate_0: initial learning rate
    :param previous_learning_rate: learning rate of previous iteration
    :param cur_iter: current iteration
    :param final_iter: final iteration
    :return:
    """
```

این تابع به صورت نمایی ضریب یادگیری را کم می‌کند. این تابع کمترین ضریب یادگیری که بر می‌گرداند برابر با ۰,۰۱ است.

$$\beta_t = \beta_0 \text{Exp}[-t/T]$$

```
def neighbourhood_size(self, nb_0, previous_nb, cur_iter, final_iter):
    """
    neighbourhood decay
    :param nb_0: initial radius of learning rate
    :param previous_nb: neighbourhood size of previous iteration
    :param cur_iter: current iteration
    :param final_iter: final iteration
    :return:
    """
```

این تابع به صورت نمایی اندازه‌ی همسایگی را کم می‌کند.

$$\sigma_t = \sigma_0 \text{Exp}[-t/T]$$

```
def neighbour_strength(self, std, index_winner, index_neighbour):
    """
    gaussian strength of neurons in neighbourhood of winner neuron
    :param std: standard deviation-equal size of neighbourhood
    :param index_winner: indexes of winner neuron in network
    :param index_neighbour: indexes of a neighbour to winner neuron in network
    :return: return strength, a scalar
    """
```


این تابع مشخص می‌کند نرون‌ها در یک همسایگی به چه میزان تاثیر باید آپدیت شوند. نرون برنده بیشترین قدرت را دارد و هر چه از آن دور می‌شویم به تاثیر کمتری نرون‌ها به روزرسانی می‌شوند. برای این کار از توزیع گوسی استفاده کرده‌ام:

$$NS(d, t) = \text{Exp}[-d_{i,j}^2/2\sigma_t^2]$$

```
def winner_neuron(self, x):  
    """  
    It computes distance of input sample to all neurons and returns index of  
    winner neuron  
    :param x: input sample  
    :return: indexes of winner  
    """
```

این تابع فاصله یک نمونه با تمام نرون‌ها را پیدا می‌کند، سپس نرونی را پیدا می‌کند که کم‌ترین فاصله را با نمونه دارد و اندیس آن نرون را به عنوان نرون برنده باز می‌گرداند.

```
def fit(self, X):  
    """  
    compute weights of cluster for each neuron  
    :param X: numpy.ndarray, each row is an observation  
    :return:  
    """
```

این تابع تا زمان رسیدن به شرط پایان، در هر ایپاک تمام نمونه‌ها را به شبکه می‌دهد. یک نمونه را به شبکه می‌دهد، نرون برنده را مشخص می‌کند و بر اساس سائز همسایگی و ضریب یادگیری و قدرت همسایگی در ایپاکی که در آن قرار دارد، وزن نرون برنده و همسایگی‌اش را به روز رسانی می‌کند.

```
def predict(self, X):  
    """  
    compute index of winner neuron for each observation  
    :param X:  
    :return:  
    """
```

این تابع یک یا بیش از یک نمونه می‌گیرد و نرون برنده را برای هر کدام از آن‌ها پیدا می‌کند و اندیس نرون برنده برای هر کدام را باز می‌گرداند.

```
def write_in_file(self):...
```

```
def load_from_file(self, file_name):...
```

این دو تابع شبکه را در فایل ذخیره و یا از فایل بازیابی می‌کنند.

در فایل SOM_run.py یک رابط کاربری فراهم شده است. که با توجه به انتخاب کاربر پارامترهای آموزش شبکه از فایل خوانده می‌شود و یا شبکه آموزش دیده شده از فایل بارگذاری می‌شود. بعد از آن شبکه‌ی آموزش دیده شده با سه معیار f , $purity$ و RI سنجیده می‌شود و نتایج بر روی صفحه نمایش داده می‌شود.

بررسی درستی کد SOM با استفاده از دیتاست IRIS و مقایسه‌ی عملکرد SOM با Kmeans

قبل از اینکه کد را با دیتاست 20 news مورد بررسی قرار دهیم و عملکرد آن را بررسی کنیم در فایل correctness.py برای اینکه نشان دهیم کد نوشته شده درست است آن را با دیتاست IRIS مورد سنجش قرار می‌دهیم و عملکرد SOM و Kmeans را بر روی دیتاست Iris را نمایش می‌دهیم.

عملکرد Kmeans، ۳ خوشه، پکیج Sklearn	عملکرد SOM یک بعدی با ۳ نرون، همسایگی خطی، فاصله کسینوسی
PU Train-Kmean: 0.8833333333333333	PU-Train-SOM: 0.9666666666666667
PU Test-Kmean: 0.9	PU-Test-SOM: 0.9666666666666667
PU Test-Kmean: 0.9	RI-Train-SOM: 0.9568627450980393
RI Train-Kmean: 0.8686274509803922	RI-Test-SOM: 0.9586206896551724
RI Test-Kmean: 0.8896551724137931	F Measure-SOM: 0.9351578947368421
F Measure Train-Kmean: 0.8010182435299108	F Measure-SOM: 0.9379310344827587
F Measure Test-Kmean: 0.8527607361963191	

همین طور که در سه معیار مختلف، در جدول بالا مشاهده می‌شود شبکه‌ی SOM که پیاده سازی کرده‌ام بسیار بهتر از Kmeans عمل می‌کند و در هر سه نوع معیار ارزیابی، بر روی مجموعه‌ی تست و آموزش بهتر عمل کرده است.

عنوان: معماری های مختلف (یک بعدی، دو بعدی و سه بعدی) را بر اساس معیارهای ارزیابی مقایسه کرده و نتایج را تحلیل کنید.

بررسی شبکه های یک بعدی مختلف

در این قسمت تعداد مختلفی از شبکه های یک بعدی با تعداد نورن های مختلف را بررسی می کنیم.

شرایط آموزش برای شبکه های یک بعدی

شبکه ی اول:

```
input-1.json
1  {
2      "ratio_of_train_set":0.80,
3      "ratio_of_valid_set":0.10,
4      "ratio_of_test_set":0.10,
5      "initial_learning_rate":0.1,
6      "max_epochs":30,
7      "cut_cost": -1,
8      "random_state":0,
9      "shape_som":[3, 1, 1],
10     "type_distance":"cosine",
11     "type_of_neighbourhood": "cubic"
12 }
```

پارامترهای این شبکه در فایل input-1.json قرار دارد. شبکه ی اول یک شبکه ی یک بعد با ۳ نرون است، فاصله از نوع کسینوسی است، همسایگی یک خط (مکعب در فضای یک بعدی یک خط است) است، حداکثر تعداد ایپاک برابر با ۳۰ است، ضریب یادگیری اولیه برابر با ۰٫۱ است، ۸۰ درصد دیتاست برای آموزش است، ۱۰ درصد برای ارزیابی و ۱۰ درصد برای تست.

شبکه ی دوم:

input-2.json

```
1  {
2      "ratio_of_train_set":0.80,
3      "ratio_of_valid_set":0.10,
4      "ratio_of_test_set":0.10,
5      "initial_learning_rate":0.1,
6      "max_epochs":30,
7      "cut_cost": -1,
8      "random_state":0,
9      "shape_som": [4, 1, 1],
10     "type_distance": "cosine",
11     "type_of_neighbourhood": "cubic"
12 }
```

پارامترهای این شبکه در فایل input-2.json قرار دارد. شبکه‌ی دوم یک شبکه‌ی یک بعدی با ۴ نرون است، فاصله از نوع کسینوسی است، همسایگی یک خط (مکعب در فضای یک بعدی یک خط است) است، حداکثر تعداد ایپاک برابر با ۳۰ است، ضریب یادگیری اولیه برابر با ۰٫۱ است، ۸۰ درصد دیتاست برای آموزش است، ۱۰ درصد برای ارزیابی و ۱۰ درصد برای تست.

شبکه‌ی سوم:

```

input-3.json
1  {
2      "ratio_of_train_set":0.80,
3      "ratio_of_valid_set":0.10,
4      "ratio_of_test_set":0.10,
5      "initial_learning_rate":0.1,
6      "max_epochs":30,
7      "cut_cost": -1,
8      "random_state":0,
9      "shape_som": [9, 1, 1],
10     "type_distance": "cosine",
11     "type_of_neighbourhood": "cubic"
12 }

```

پارامترهای این شبکه در فایل input-3.json قرار دارد. شبکه‌ی سوم یک شبکه‌ی یک بعد با ۹ نرون است، فاصله از نوع کسینوسی است، همسایگی یک خط (مکعب در فضای یک بعدی یک خط است) است، حداکثر تعداد ایپاک برابر با ۳۰ است، ضریب یادگیری اولیه برابر با ۰٫۱ است، ۸۰ درصد دیتاست برای آموزش است، ۱۰ درصد برای ارزیابی و ۱۰ درصد برای تست.

شبکه‌ی چهارم:

input-4.json

```
1  {  
2    "ratio_of_train_set":0.80,  
3    "ratio_of_valid_set":0.10,  
4    "ratio_of_test_set":0.10,  
5    "initial_learning_rate":0.1,  
6    "max_epochs":30,  
7    "cut_cost": -1,  
8    "random_state":0,  
9    "shape_som":[16, 1, 1],  
10   "type_distance":"cosine",  
11   "type_of_neighbourhood": "cubic"  
12 }
```

پارامترهای این شبکه در فایل input-4.json قرار دارد. شبکه‌ی اول یک شبکه‌ی یک بعد با ۱۶ نرون است، فاصله از نوع کسینوسی است، همسایگی یک خط (مکعب در فضای یک بعدی یک خط است) است، حداکثر تعداد ایپاک برابر با ۳۰ است، ضریب یادگیری اولیه برابر با ۰,۱ است، ۸۰ درصد دیتاست برای آموزش است، ۱۰ درصد برای ارزیابی و ۱۰ درصد برای تست.

شبکه‌ی پنجم:

```

input-5.json
1  {
2      "ratio_of_train_set":0.80,
3      "ratio_of_valid_set":0.10,
4      "ratio_of_test_set":0.10,
5      "initial_learning_rate":0.1,
6      "max_epochs":30,
7      "cut_cost": -1,
8      "random_state":0,
9      "shape_som": [20, 1, 1],
10     "type_distance": "cosine",
11     "type_of_neighbourhood": "cubic"
12 }

```

پارامترهای این شبکه در فایل input-5.json قرار دارد. شبکه‌ی اول یک شبکه‌ی یک بعد با ۲۰ نرون است، فاصله از نوع کسینوسی است، همسایگی یک خط (مکعب در فضای یک بعدی یک خط است) است، حداکثر تعداد ایپاک برابر با ۳۰ است، ضریب یادگیری اولیه برابر با ۰٫۱ است، ۸۰ درصد دیتاست برای آموزش است، ۱۰ درصد برای ارزیابی و ۱۰ درصد برای تست.

نتیجه‌ی آموزش برای شبکه‌های یک بعدی

در زیر معیارهای ارزیابی برای شبکه‌های بالا بر روی دیتاست 20new مشاهده می‌کنید.

شبکه‌ی اول:

معیارهای Purity, F و RI بعد از خوشه‌بندی دیتاست 20News با شبکه‌ی اول:

PU-Train-SOM: 0.1498472978356128
PU-Valid-SOM: 0.15082315454062667
PU-Test-SOM: 0.14869888475836432
RI-Train-SOM: 0.6845199018965858
RI-Valid-SOM: 0.6801348606554648
RI-Test-SOM: 0.6870020537241598
F-Measure-Train-SOM: 0.18151420957034772
F-Measure-Valid-SOM: 0.1798304005556842
F-Measure-Test-SOM: 0.18428406697494912

این شبکه در فایل som_net-1.json ذخیره شده است.

شبکه‌ی دوم:

معیارهای Purity, F و RI بعد از خوشه‌بندی دیتاست 20News با شبکه‌ی دوم:

PU-Train-SOM: 0.16325853140353208
PU-Valid-SOM: 0.15878916622411046
PU-Test-SOM: 0.16250663834306958
RI-Train-SOM: 0.7167381396828483
RI-Valid-SOM: 0.7074044120925356
RI-Test-SOM: 0.7186132649473476
F-Measure-Train-SOM: 0.17376151550071325
F-Measure-Valid-SOM: 0.17089626828078647
F-Measure-Test-SOM: 0.17384964623618498

این شبکه در فایل som_net-2.json ذخیره شده است.

شبکه‌ی سوم:

معیارهای Purity, F و RI بعد از خوشه‌بندی دیتاست 20News با شبکه‌ی سوم:

PU-Train-SOM: 0.2811711592086044
PU-Valid-SOM: 0.27562400424853956
PU-Test-SOM: 0.29314922995220394
RI-Train-SOM: 0.8721209356107541
RI-Valid-SOM: 0.8672760303470337
RI-Test-SOM: 0.8739242498037421
F-Measure-Train-SOM: 0.2461778408708507
F-Measure-Valid-SOM: 0.23568870371212955
F-Measure-Test-SOM: 0.2524078201447034

این شبکه در فایل som_net-3.json ذخیره شده است.

شبکه‌ی چهارم:

معیارهای F, Purity, RI بعد از خوشه‌بندی دیتاست 20News با شبکه‌ی چهارم:

PU-Train-SOM: 0.22686230248307
PU-Valid-SOM: 0.23632501327668615
PU-Test-SOM: 0.23366967604885822
RI-Train-SOM: 0.8585909974160673
RI-Valid-SOM: 0.8558431245954209
RI-Test-SOM: 0.8554407323651464
F-Measure-Train-SOM: 0.2092568521133822
F-Measure-Valid-SOM: 0.20641745787145202
F-Measure-Test-SOM: 0.20555240230879693

این شبکه در فایل som_net-4.json ذخیره شده است.

شبکه‌ی پنجم:

معیارهای F, Purity, RI بعد از خوشه‌بندی دیتاست 20News با شبکه‌ی پنجم:

PU-Train-SOM: 0.22467135838534058
 PU-Valid-SOM: 0.23473181093998938
 PU-Test-SOM: 0.22092405735528411
 RI-Train-SOM: 0.8697136756061578
 RI-Valid-SOM: 0.8669989271421743
 RI-Test-SOM: 0.8674504191256519
 F-Measure-Train-SOM: 0.20505284954047376
 F-Measure-Valid-SOM: 0.1944095960510978
 F-Measure-Test-SOM: 0.2047936184404319

این شبکه در فایل som_net-5json ذخیره شده است.

نتیجه‌گیری برای شبکه‌های یک بعدی

شبکه‌ی یک نرون ۳	شبکه‌ی دو نرون ۴	شبکه‌ی سوم نرون ۹	شبکه‌ی چهارم نرون ۱۶	شبکه‌ی پنجم نرون ۲۰	
0.1498	0.1632	0.2811	0.2268	0.2246	PU Train
0.1508	0.1587	0.2756	0.2363	0.2347	PU Valid
0.1486	0.1625	0.2931	0.2336	0.2209	PU Test
0.6845	0.7167	0.8721	0.8585	0.8697	RI Train
0.6801	0.7074	0.8672	0.8558	0.8669	RI Valid
0.6870	0.7186	0.8739	0.8554	0.8674	RI Test
0.1815	0.1737	0.2461	0.2092	0.2050	F Train
0.1798	0.1708	0.2356	0.2064	0.1944	F Valid
0.1842	0.1738	0.2524	0.2055	0.2047	F Test

همین طور که مشاهده می‌شود، با زیاد کردن تعداد نرون‌ها در ابتدا دقت افزایش یافته است ولی در ادامه معیارهای ارزیابی کاهش پیدا کرده‌اند. در شبکه‌های یک بعدی ایجاد شده شبکه یک بعدی با ۹ نرون بهترین عملکرد را در تمام معیارهای ارزیابی دارد.

بررسی شبکه‌های دو بعدی مختلف

در این قسمت تعداد مختلفی از شبکه‌های دو بعدی با تعداد نرون‌های مختلف را بررسی می‌کنیم.

شرایط آموزش برای شبکه‌های دو بعدی

شبکه‌ی ششم:

```
input-6.json
1  {
2      "ratio_of_train_set":0.80,
3      "ratio_of_valid_set":0.10,
4      "ratio_of_test_set":0.10,
5      "initial_learning_rate":0.1,
6      "max_epochs":30,
7      "cut_cost": -1,
8      "random_state":0,
9      "shape_som": [2,2, 1],
10     "type_distance": "cosine",
11     "type_of_neighbourhood": "cubic"
12 }
```

پارامترهای این شبکه در فایل input-6.json قرار دارد. شبکه‌ی اول یک شبکه‌ی دو بعدی با ۴ نرون به صورت 2×2 است، فاصله از نوع کسینوسی است، همسایگی یک مربع (مکعب در فضای دو بعدی یک مربع است) است، حداکثر تعداد اپاک برابر با ۳۰ است، ضریب یادگیری اولیه برابر با ۰٫۱ است، ۸۰ درصد دیتاست برای آموزش است، ۱۰ درصد برای ارزیابی و ۱۰ درصد برای تست.

شبکه‌ی هفتم:

input-7.json

```
1  {  
2      "ratio_of_train_set":0.80,  
3      "ratio_of_valid_set":0.10,  
4      "ratio_of_test_set":0.10,  
5      "initial_learning_rate":0.1,  
6      "max_epochs":30,  
7      "cut_cost": -1,  
8      "random_state":0,  
9      "shape_som":[3, 3, 1],  
10     "type_distance":"cosine",  
11     "type_of_neighbourhood": "cubic"  
12 }
```

پارامترهای این شبکه در فایل input-7.json قرار دارد. شبکه‌ی هفتم یک شبکه‌ی دو بعدی با ۹ نرون به صورت 3×3 است، فاصله از نوع کسینوسی است، همسایگی یک مربع (مکعب در فضای دو بعدی یک مربع است) است، حداکثر تعداد ایپاک برابر با ۳۰ است، ضریب یادگیری اولیه برابر با ۰٫۱ است، ۸۰ درصد دیتاست برای آموزش است، ۱۰ درصد برای ارزیابی و ۱۰ درصد برای تست.

شبکه‌ی هشتم:

```
input-8.json
1  {
2      "ratio_of_train_set":0.80,
3      "ratio_of_valid_set":0.10,
4      "ratio_of_test_set":0.10,
5      "initial_learning_rate":0.1,
6      "max_epochs":30,
7      "cut_cost": -1,
8      "random_state":0,
9      "shape_som":[4, 4, 1],
10     "type_distance":"cosine",
11     "type_of_neighbourhood": "cubic"
12 }
```

پارامترهای این شبکه در فایل input-8.json قرار دارد. شبکه‌ی هشتم یک شبکه‌ی دو بعدی با ۹ نرون به صورت 3×3 است، فاصله از نوع کسینوسی است، همسایگی یک مربع (مکعب در فضای دو بعدی یک مربع است) است، حداکثر تعداد ایپاک برابر با ۳۰ است، ضریب یادگیری اولیه برابر با ۰٫۱ است، ۸۰ درصد دیتاست برای آموزش است، ۱۰ درصد برای ارزیابی و ۱۰ درصد برای تست.

شبکه‌ی نهم:

input-9.json

```
1  {  
2    "ratio_of_train_set":0.80,  
3    "ratio_of_valid_set":0.10,  
4    "ratio_of_test_set":0.10,  
5    "initial_learning_rate":0.1,  
6    "max_epochs":30,  
7    "cut_cost": -1,  
8    "random_state":0,  
9    "shape_som": [4, 5, 1],  
10   "type_distance": "cosine",  
11   "type_of_neighbourhood": "cubic"  
12 }
```

پارامترهای این شبکه در فایل input-6.json قرار دارد. شبکه‌ی نهم یک شبکه‌ی دو بعد با ۲۰ نرون به صورت ۵*۴ است، فاصله از نوع کسینوسی است، همسایگی یک مربع (مکعب در فضای دو بعدی یک مربع است) است، حداکثر تعداد ایپاک برابر با ۳۰ است، ضریب یادگیری اولیه برابر با ۰,۱ است، ۸۰ درصد دیتاست برای آموزش است، ۱۰ درصد برای ارزیابی و ۱۰ درصد برای تست.

شبکه‌ی دهم:

input-10.json

```
1  {  
2    "ratio_of_train_set":0.80,  
3    "ratio_of_valid_set":0.10,  
4    "ratio_of_test_set":0.10,  
5    "initial_learning_rate":0.1,  
6    "max_epochs":30,  
7    "cut_cost": -1,  
8    "random_state":0,  
9    "shape_som": [6, 6, 1],  
10   "type_distance": "cosine",  
11   "type_of_neighbourhood": "cubic"  
12 }
```

پارامترهای این شبکه در فایل input-10.json قرار دارد. شبکه‌ی دهم یک شبکه‌ی دو بعد با ۳۶ نرون به صورت ۶*۶ است، فاصله از نوع کسینوسی است، همسایگی یک مربع (مکعب در فضای دو بعدی یک مربع است) است، حداکثر تعداد ایپاک برابر با ۳۰ است، ضریب یادگیری اولیه برابر با ۰٫۱ است، ۸۰ درصد دیتاست برای آموزش است، ۱۰ درصد برای ارزیابی و ۱۰ درصد برای تست.

شبکه‌ی یازدهم:


```

input-11.json
1  {
2      "ratio_of_train_set":0.80,
3      "ratio_of_valid_set":0.10,
4      "ratio_of_test_set":0.10,
5      "initial_learning_rate":0.1,
6      "max_epochs":30,
7      "cut_cost":-1,
8      "random_state":0,
9      "shape_som":[8, 8, 1],
10     "type_distance":"cosine",
11     "type_of_neighbourhood": "cubic"
12 }

```

پارامترهای این شبکه در فایل input-10.json قرار دارد. شبکه‌ی دهم یک شبکه‌ی دو بعد با ۶۴ نرون به صورت ۸*۸ است، فاصله از نوع کسینوسی است، همسایگی یک مربع (مکعب در فضای دو بعدی یک مربع است) است، حداکثر تعداد ایپاک برابر با ۳۰ است، ضریب یادگیری اولیه برابر با ۰,۱ است، ۸۰ درصد دیتاست برای آموزش است، ۱۰ درصد برای ارزیابی و ۱۰ درصد برای تست.

نتیجه‌ی آموزش برای شبکه‌های دو بعدی

در زیر معیارهای ارزیابی برای شبکه‌های بالا بر روی دیتاست 20new مشاهده می‌کنید.

شبکه‌ی ششم:

معیارهای Purity, F و RI بعد از خوشه‌بندی دیتاست 20News با شبکه‌ی ششم:

PU-Train-SOM: 0.19047935201168503
PU-Valid-SOM: 0.18852894317578334
PU-Test-SOM: 0.1938396176314392
RI-Train-SOM: 0.7547254717021139
RI-Valid-SOM: 0.7470973298199732
RI-Test-SOM: 0.7540147513718302
F-Measure-Train-SOM: 0.21102848339990313
F-Measure-Valid-SOM: 0.2073838763139695
F-Measure-Test-SOM: 0.2121533100037597

این شبکه در فایل som_net-6.json ذخیره شده است.

شبکه‌ی هفتم:

معیارهای Purity, F و RI بعد از خوشه‌بندی دیتاست 20News با شبکه‌ی هفتم:

PU-Train-SOM: 0.3786349754348692
PU-Valid-SOM: 0.379182156133829
PU-Test-SOM: 0.37652681890600104
RI-Train-SOM: 0.891501193953953
RI-Valid-SOM: 0.8897287266853773
RI-Test-SOM: 0.8922333784637195
F-Measure-Train-SOM: 0.349299600408427
F-Measure-Valid-SOM: 0.3466703224684687
F-Measure-Test-SOM: 0.3515004143289908

این شبکه در فایل som_net-7.json ذخیره شده است.

شبکه‌ی هشتم:

معیارهای Purity, F و RI بعد از خوشه‌بندی دیتاست 20News با شبکه‌ی هشتم:

PU-Train-SOM: 0.3687425308723941
PU-Valid-SOM: 0.3643122676579926
PU-Test-SOM: 0.37599575146043546
RI-Train-SOM: 0.9097285344913159
RI-Valid-SOM: 0.9064316726141329
RI-Test-SOM: 0.9101276988638769
F-Measure-Train-SOM: 0.298049082471633
F-Measure-Valid-SOM: 0.2824137393743183
F-Measure-Test-SOM: 0.2952295388864056

این شبکه در فایل som_net-8.json ذخیره شده است.

شبکه‌ی نهم:

معیارهای Purity, F و RI بعد از خوشه‌بندی دیتاست 20News با شبکه‌ی نهم:

PU-Train-SOM: 0.3651573496215642
PU-Valid-SOM: 0.3744025491237387
PU-Test-SOM: 0.372809346787042
RI-Train-SOM: 0.9156137677653478
RI-Valid-SOM: 0.914217651869205
RI-Test-SOM: 0.9162414646851436
F-Measure-Train-SOM: 0.307825536973665
F-Measure-Valid-SOM: 0.3028254288597377
F-Measure-Test-SOM: 0.2982353274952242

این شبکه در فایل som_net-9.json ذخیره شده است.

شبکه‌ی دهم:

معیارهای Purity, F و RI بعد از خوشه‌بندی دیتاست 20News با شبکه‌ی دهم:

PU-Train-SOM: 0.545013942371531
PU-Valid-SOM: 0.541157727031333
PU-Test-SOM: 0.5554965480616039
RI-Train-SOM: 0.9444807246088718
RI-Valid-SOM: 0.9427101822165209
RI-Test-SOM: 0.944990216733083
F-Measure-Train-SOM: 0.38584207819140753
F-Measure-Valid-SOM: 0.36084421553689033
F-Measure-Test-SOM: 0.38501918029477084

این شبکه در فایل som_net-10.json ذخیره شده است.

شبکه‌ی یازدهم:

معیارهای F, Purity, RI و بعد از خوشه‌بندی دیتاست 20News با شبکه‌ی یازدهم:

PU-Train-SOM: 0.6191740804674014
PU-Valid-SOM: 0.6229421136484333
PU-Test-SOM: 0.6282527881040892
RI-Train-SOM: 0.9508721717536046
RI-Valid-SOM: 0.9501090070957608
RI-Test-SOM: 0.9520374422301898
F-Measure-Train-SOM: 0.3469512365198604
F-Measure-Valid-SOM: 0.34531585573576246
F-Measure-Test-SOM: 0.3575030429490524

این شبکه در فایل som_net-11.json ذخیره شده است.

نتیجه‌گیری برای شبکه‌های دو بعدی

شبکه‌ی ششم ۲*۲ نرون	شبکه‌ی هفتم ۳*۳ نرون	شبکه‌ی هشتم ۴*۴ نرون	شبکه‌ی نهم ۵*۴ نرون	شبکه‌ی دهم ۶*۶ نرون	شبکه‌ی یازدهم ۸*۸ نرون	
0.1904	0.3786	0.3687	0.3651	0.5450	0.6191	PU Train
0.1885	0.3791	0.3643	0.3744	0.5441	0.6229	PU Valid
0.1938	0.3765	0.3759	0.3728	0.5554	0.6282	PU Test
0.7547	0.8915	0.9097	0.9156	0.9444	0.9508	RI Train
0.7470	0.8897	0.9064	0.9142	0.9427	0.9501	RI Valid
0.7540	0.8922	0.9101	0.9162	0.9449	0.9520	RI Test
0.2110	0.3492	0.2980	0.3078	0.3858	0.3469	F Train
0.2073	0.3466	0.2824	0.3028	0.3608	0.3453	F Valid
0.2121	0.3515	0.2952	0.2980	0.3850	0.3575	F Test

با زیاد کردن تعداد نرون‌ها در دو بعد، دو معیار Purity و RI افزایش داشته‌اند ولی معیار F-Measure در ابتدا با زیاد کردن تعداد نرون‌ها افزایش داشته است و سپس مقدار آن کاهش یافته است. بیشترین معیار F را شبکه‌ی ۶*۶ دارد.

بررسی شبکه‌های سه بعدی مختلف

در این قسمت تعداد مختلفی از شبکه‌های سه بعدی با تعداد نورن‌های مختلف را بررسی می‌کنیم.

شرایط آموزش برای شبکه‌های سه بعدی

شبکه‌ی دوازده:

```
input-12.json
1  {
2      "ratio_of_train_set":0.80,
3      "ratio_of_valid_set":0.10,
4      "ratio_of_test_set":0.10,
5      "initial_learning_rate":0.1,
6      "max_epochs":30,
7      "cut_cost": -1,
8      "random_state":0,
9      "shape_som":[2, 2, 2],
10     "type_distance":"cosine",
11     "type_of_neighbourhood": "cubic"
12 }
```

پارامترهای این شبکه در فایل input-12.json قرار دارد. شبکه‌ی دوازده یک شبکه‌ی سه بعدی با ۸ نرون به صورت $2 \times 2 \times 2$ است، فاصله از نوع کسینوسی است، همسایگی یک مکعب است، حداکثر تعداد ایپاک برابر با ۳۰ است، ضریب یادگیری اولیه برابر با ۰٫۱ است، ۸۰ درصد دیتاست برای آموزش است، ۱۰ درصد برای ارزیابی و ۱۰ درصد برای تست.

شبکه‌ی سیزده:

```

input-13.json
1  {
2      "ratio_of_train_set":0.80,
3      "ratio_of_valid_set":0.10,
4      "ratio_of_test_set":0.10,
5      "initial_learning_rate":0.1,
6      "max_epochs":30,
7      "cut_cost": -1,
8      "random_state":0,
9      "shape_som": [2, 2, 4],
10     "type_distance": "cosine",
11     "type_of_neighbourhood": "cubic"
12 }

```

پارامترهای این شبکه در فایل input-13.json قرار دارد. شبکه‌ی سیزده یک شبکه‌ی سه بعد با 16 نرون به صورت $4 \times 2 \times 2$ است، فاصله از نوع کسینوسی است، همسایگی یک مکعب است، حداکثر تعداد اپیاک برابر با 30 است، ضریب یادگیری اولیه برابر با 0.1 است، 80 درصد دیتاست برای آموزش است، 10 درصد برای ارزیابی و 10 درصد برای تست.

شبکه‌ی چهارده:

```

input-14.json
1  {
2      "ratio_of_train_set":0.80,
3      "ratio_of_valid_set":0.10,
4      "ratio_of_test_set":0.10,
5      "initial_learning_rate":0.1,
6      "max_epochs":30,
7      "cut_cost": -1,
8      "random_state":0,
9      "shape_som":[2, 3, 4],
10     "type_distance":"cosine",
11     "type_of_neighbourhood": "cubic"
12 }

```

پارامترهای این شبکه در فایل input-14.json قرار دارد. شبکه‌ی چهارده یک شبکه‌ی سه بعد با ۲۴ نرون به صورت $4 \times 3 \times 2$ است، فاصله از نوع کسینوسی است، همسایگی یک مکعب است، حداکثر تعداد ایپاک برابر با ۳۰ است، ضریب یادگیری اولیه برابر با ۰٫۱ است، ۸۰ درصد دیتاست برای آموزش است، ۱۰ درصد برای ارزیابی و ۱۰ درصد برای تست.

شبکه‌ی پانزده:


```

input-15.json
1  {
2      "ratio_of_train_set":0.80,
3      "ratio_of_valid_set":0.10,
4      "ratio_of_test_set":0.10,
5      "initial_learning_rate":0.1,
6      "max_epochs":30,
7      "cut_cost": -1,
8      "random_state":0,
9      "shape_som":[3, 3, 3],
10     "type_distance":"cosine",
11     "type_of_neighbourhood": "cubic"
12 }

```

پارامترهای این شبکه در فایل input-15.json قرار دارد. شبکه‌ی پانزدهم یک شبکه‌ی سه بعدی با ۲۷ نرون به صورت $3 \times 3 \times 3$ است، فاصله از نوع کسینوسی است، همسایگی یک مکعب است، حداکثر تعداد ایپاک برابر با ۳۰ است، ضریب یادگیری اولیه برابر با ۰٫۱ است، ۸۰ درصد دیتاست برای آموزش است، ۱۰ درصد برای ارزیابی و ۱۰ درصد برای تست.

شبکه‌ی شانزدهم:

```

input-16.json
1  {
2      "ratio_of_train_set":0.80,
3      "ratio_of_valid_set":0.10,
4      "ratio_of_test_set":0.10,
5      "initial_learning_rate":0.1,
6      "max_epochs":30,
7      "cut_cost": -1,
8      "random_state":0,
9      "shape_som":[3,3, 4],
10     "type_distance":"cosine",
11     "type_of_neighbourhood": "cubic"
12 }

```

پارامترهای این شبکه در فایل input-16.json قرار دارد. شبکه‌ی شانزدهم یک شبکه‌ی سه بعد با ۳۶ نرون به صورت $4 \times 3 \times 3$ است، فاصله از نوع کسینوسی است، همسایگی یک مکعب است، حداکثر تعداد اپیاک برابر با ۳۰ است، ضریب یادگیری اولیه برابر با ۰٫۱ است، ۸۰ درصد دیتاست برای آموزش است، ۱۰ درصد برای ارزیابی و ۱۰ درصد برای تست.

شبکه‌ی هفدهم:

```

input-17.json
1  {
2      "ratio_of_train_set":0.80,
3      "ratio_of_valid_set":0.10,
4      "ratio_of_test_set":0.10,
5      "initial_learning_rate":0.1,
6      "max_epochs":30,
7      "cut_cost": -1,
8      "random_state":0,
9      "shape_som":[4,4, 4],
10     "type_distance":"cosine",
11     "type_of_neighbourhood": "cubic"
12 }

```

پارامترهای این شبکه در فایل input-17.json قرار دارد. شبکه‌ی هفدهم یک شبکه‌ی سه بعد با ۶۴ نرون به صورت $4 \times 4 \times 4$ است، فاصله از نوع کسینوسی است، همسایگی یک مکعب است، حداکثر تعداد ایپاک برابر با ۳۰ است، ضریب یادگیری اولیه برابر با ۰٫۱ است، ۸۰ درصد دیتاست برای آموزش است، ۱۰ درصد برای ارزیابی و ۱۰ درصد برای تست.

نتیجه‌ی آموزش برای شبکه‌های دو بعدی

در زیر معیارهای ارزیابی برای شبکه‌های بالا بر روی دیتاست 20new مشاهده می‌کنید.

شبکه‌ی دوازده:

معیارهای Purity, F و RI بعد از خوشه‌بندی دیتاست 20News با شبکه‌ی دوازده:

PU-Train-SOM: 0.34696587438587173
PU-Valid-SOM: 0.34466277217206587
PU-Test-SOM: 0.34891131173659057
RI-Train-SOM: 0.8799898956477535
RI-Valid-SOM: 0.8786857971344932
RI-Test-SOM: 0.8810995861511607
F-Measure-Train-SOM: 0.3328688958652257
F-Measure-Valid-SOM: 0.32827406900474054
F-Measure-Test-SOM: 0.33356530541233037

این شبکه در فایل som_net-12.json ذخیره شده است.

شبکه‌ی سیزده:

معیارهای Purity, F و RI بعد از خوشه‌بندی دیتاست 20News با شبکه‌ی سیزده:

PU-Train-SOM: 0.3103173549329438
PU-Valid-SOM: 0.3138608603292618
PU-Test-SOM: 0.3090812533191715
RI-Train-SOM: 0.8985691005385252
RI-Valid-SOM: 0.8962014286335087
RI-Test-SOM: 0.8995864897796324
F-Measure-Train-SOM: 0.2511239905240455
F-Measure-Valid-SOM: 0.24102769375518404
F-Measure-Test-SOM: 0.2514084240376646

این شبکه در فایل som_net-13.json ذخیره شده است.

شبکه‌ی چهارده:

معیارهای Purity, F و RI بعد از خوشه‌بندی دیتاست 20News با شبکه‌ی چهارده:

PU-Train-SOM: 0.4041959899083787
PU-Valid-SOM: 0.4115772703133298
PU-Test-SOM: 0.4243228890069039
RI-Train-SOM: 0.9217000689031691
RI-Valid-SOM: 0.9207016411169234
RI-Test-SOM: 0.9234602571359719
F-Measure-Train-SOM: 0.31630411754634685
F-Measure-Valid-SOM: 0.30695320630755496
F-Measure-Test-SOM: 0.3136276449838302

این شبکه در فایل som_net-14.json ذخیره شده است.

شبکه‌ی پانزده:

معیارهای Purity, F و RI بعد از خوشه‌بندی دیتاست 20News با شبکه‌ی پانزده:

PU-Train-SOM: 0.6625946089496747
PU-Valid-SOM: 0.6617100371747212
PU-Test-SOM: 0.6686139139670738
RI-Train-SOM: 0.9523789816953921
RI-Valid-SOM: 0.9503200795980367
RI-Test-SOM: 0.9533117783535555
F-Measure-Train-SOM: 0.48346236290757166
F-Measure-Valid-SOM: 0.46279841820045886
F-Measure-Test-SOM: 0.4892228471934948

این شبکه در فایل som_net-15.json ذخیره شده است.

شبکه‌ی شانزده:

معیارهای Purity, F و RI بعد از خوشه‌بندی دیتاست 20News با شبکه‌ی شانزده:

PU-Train-SOM: 0.44024697915283495
PU-Valid-SOM: 0.4684014869888476
PU-Test-SOM: 0.46521508231545405
RI-Train-SOM: 0.9322205926589459
RI-Valid-SOM: 0.9333078616605988
RI-Test-SOM: 0.9342243903870584
F-Measure-Train-SOM: 0.34689550498665306
F-Measure-Valid-SOM: 0.35418784361303296
F-Measure-Test-SOM: 0.35872435953869175

این شبکه در فایل som_net-16.json ذخیره شده است.

شبکه‌ی هفدهم:

معیارهای F, Purity, RI بعد از خوشه‌بندی دیتاست 20News با شبکه‌ی هفدهم:

PU-Train-SOM: 0.5353870667905989
PU-Valid-SOM: 0.5523101433882103
PU-Test-SOM: 0.5581518852894317
RI-Train-SOM: 0.9471930424559212
RI-Valid-SOM: 0.946468288613993
RI-Test-SOM: 0.9483216632061687
F-Measure-Train-SOM: 0.32183028017540305
F-Measure-Valid-SOM: 0.3127739579635278
F-Measure-Test-SOM: 0.3318033552492356

این شبکه در فایل som_net-17.json ذخیره شده است.

شبکه‌ی دوازده ۲*۲*۲ نرون	شبکه‌ی سیزده ۲*۲*۴ نرون	شبکه‌ی چهارده ۲*۳*۴ نرون	شبکه‌ی پانزده ۳*۳*۳ نرون	شبکه‌ی شانزده ۴*۳*۳ نرون	شبکه‌ی هفدهم ۴*۴*۴ نرون	
0.3469	0.3103	0.4041	0.6625	0.4402	0.5353	PU Train
0.3446	0.3138	0.4115	0.6617	0.4684	0.5523	PU Valid
0.3489	0.3090	0.4243	0.6686	0.4652	0.5581	PU Test
0.8799	0.8985	0.9217	0.9523	0.9322	0.9471	RI Train
0.8786	0.8962	0.9207	0.9503	0.9333	0.9464	RI Valid
0.8810	0.8995	0.9234	0.9533	0.9342	0.9483	RI Test
0.3328	0.2511	0.3163	0.4834	0.3468	0.3218	F Train
0.3282	0.2410	0.3069	0.4627	0.3541	0.3127	F Valid
0.3335	0.2514	0.3136	0.4892	0.3587	0.3318	F Test

با زیاد کردم تعداد نرون‌ها در سه بعد، دو معیار Purity و RI افزایش داشته‌اند ولی معیار F-Measure در ابتدا با زیاد کردن تعداد نرون‌ها افزایش داشته است و سپس مقدار آن کاهش یافته است. بیشترین معیار F-Measure را در بین شبکه‌های ۳ بعدی، شبکه‌ی ۳*۳*۳ داشته است.

نتیجه‌گیری کلی

در بخش‌های قبل شبکه‌های یک بعدی، دو بعدی، سه بعدی مختلفی را تست کردیم. **برای هر کدام یک نتیجه‌گیری ارائه دادیم** که در بالاتر آن را مشاهده می‌کنید.

در شبکه‌های یک، دو و سه بعدی با زیاد کردن تعداد نرون‌ها، دو معیار Purity و RI افزایش می‌یافتند ولی معیار F Measure ابتدا افزایش می‌یابد و در ادامه کاهش می‌یابد.

در زیر بهترین شبکه با بالاترین F-Measure هر کدام از شبکه‌های یک بعدی، دو بعدی و سه بعدی را مشاهده می‌کنید:

بهترین شبکه‌ی سه بعدی شبکه‌ی پانزده ۳*۳*۳ نرون	بهترین شبکه‌ی دو بعدی شبکه‌ی دهم ۶*۶ نرون	بهترین شبکه‌ی یک بعدی شبکه‌ی سوم ۹ نرون	
0.6625	0.5450	0.2811	PU Train
0.6617	0.5441	0.2756	PU Valid
0.6686	0.5554	0.2931	PU Test
0.9523	0.9444	0.8721	RI Train
0.9503	0.9427	0.8672	RI Valid
0.9533	0.9449	0.8739	RI Test
0.4834	0.3858	0.2461	F Train
0.4627	0.3608	0.2356	F Valid
0.4892	0.3850	0.2524	F Test

همین طور مشاهده می‌شود، بهترین شبکه‌ی دو بعدی از بهترین شبکه‌ی یک بعدی بهتر بوده است. همین‌طور بهترین شبکه‌ی ۳ بعدی از بهترین شبکه‌های یک بعدی و دو بعدی بهتر بوده است.

شبکه‌ی ۳ بعدی ۳*۳*۳ که بهترین عملکرد را در بین شبکه‌ها داشته است دارای ۲۷ نرون است و از بهترین شبکه‌ی دو بعدی که ۶*۶ است نرون‌ها کمتری دارد. در نتیجه عملکرد F-Measure فقط به تعداد نرون‌ها نیست، بلکه ساختار شبکه نیز مربوط است.

با توجه به شبکه‌هایی که ایجاد کردیم. به این نتیجه می‌رسیم برای رسیدن به یک شبکه‌ی SOM خوب باید از معیارهای مختلفی استفاده کنیم تا توسط یک معیار دچار اشتباه نشویم. همین‌طور باید ساختارهای مختلف با تعداد نرون‌های مختلف را تست کنیم. زیاد کردن نرون‌ها در ابتدا موجب بهبود می‌شود ولی بعد از اضافه کردن نرون از یک حدی به بعد کاهش F Measure را به دنبال دارد.

عنوان: نتایج خوشه بندی حاصل از این شبکه و خوشه بندی kmeans را مقایسه نمایید

در این قسمت از تابع Kmeans موجود در پکیج sklearn استفاده می‌کنم. کد این قسمت در kmeans_run.py قرار دارد.

شرایط آزمایش

در سوال قبل شبکه‌های SOM مختلفی با تعداد نرون‌ها و بعدهای مختلفی را تست کردیم. که هر کدام داده‌ها را به توجه به تعداد نرون‌های موجود در شبکه به خوشه‌هایی تقسیم می‌کرد. به ازای تعداد خوشه‌های مختلف در سوال قبل، Kmeans را اجرا می‌کنم.

`k in [2, 3, 4, 9, 12, 16, 20, 24, 27, 36, 64]`

نتیجه‌ی انجام آزمایش

در زیر دقت kmeans را به ازای K های مختلف مشاهده می‌کنید.

PU-Train	PU-Valid	PU-Test	RI-Train	RI-Valid	RI-Test	F Train	F Valid	F Test	K
0.0967	0.1024	0.1024	0.3939	0.4034	0.4226	0.1261	0.1285	0.1288	2
0.1360	0.1354	0.1439	0.5551	0.5722	0.5593	0.1397	0.1407	0.1407	3
0.1758	0.1704	0.1816	0.6145	0.6187	0.6168	0.1502	0.1481	0.1526	4
0.2756	0.2639	0.2968	0.6970	0.6992	0.7132	0.1658	0.1598	0.1724	9
0.2996	0.2936	0.3074	0.7930	0.7906	0.7922	0.2179	0.2089	0.2123	12
0.3706	0.3680	0.3860	0.7685	0.7692	0.7704	0.1857	0.1831	0.1900	16
0.3878	0.3878	0.3967	0.8251	0.8302	0.8234	0.2138	0.2091	0.2171	20
0.4395	0.4445	0.4503	0.8292	0.8330	0.8282	0.2062	0.2067	0.2086	24
0.4269	0.4259	0.4237	0.8549	0.8539	0.8524	0.1964	0.1907	0.1922	27
0.4796	0.4583	0.4875	0.8457	0.8397	0.8478	0.2021	0.1884	0.2050	36
0.5031	0.5082	0.5082	0.9022	0.9037	0.9007	0.1973	0.1945	0.1998	64

نتیجه‌گیری

همین طور که در جدول بالا دیده می‌شود، در ابتدا با افزایش K (تعداد خوشه‌ها) معیار F افزایش یافته است ولی در ادامه کاهش می‌یابد. دو معیار ارزیابی دیگر با افزایش K افزایش می‌یابند.

بهترین عملکرد از نظر معیار F ، مربوط به تعداد ۲۰ خوشه بوده اس که برابر با ۰,۲۱۷۱ است، در حالی که در شبکه‌های SOM بارها به معیار F بالاتر از ۰,۲۱ رسیدیم و بهترین عملکرد از نظر معیار F در شبکه‌های SOM برابر با ۰,۴۸۹۲ شد که بسیار بیشتر از عملکرد بهترین Kmeans به دست آمده است. شبکه‌های SOM نسبت خوشه‌بندهای Kmeans بسیار بهتر عمل کرده اند و در هر سه معیار عملکرد بهتری داشته اند.