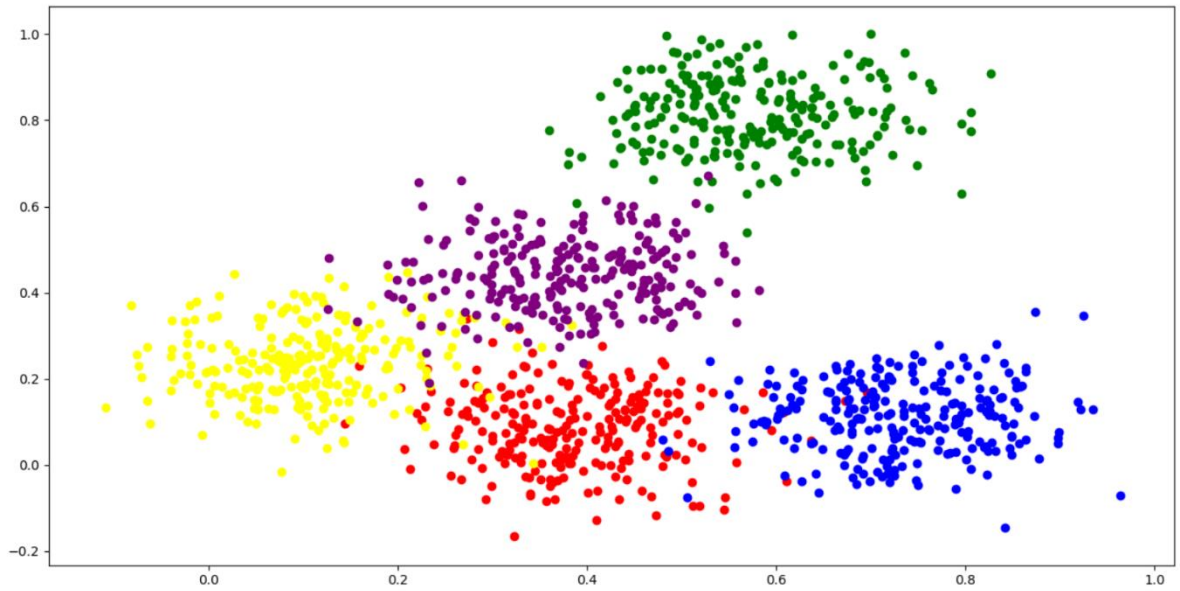
بسم الله الرحمن الرحیم

گزارش پروژه پایانی هوش محاسباتی – دکتر عبادزاده

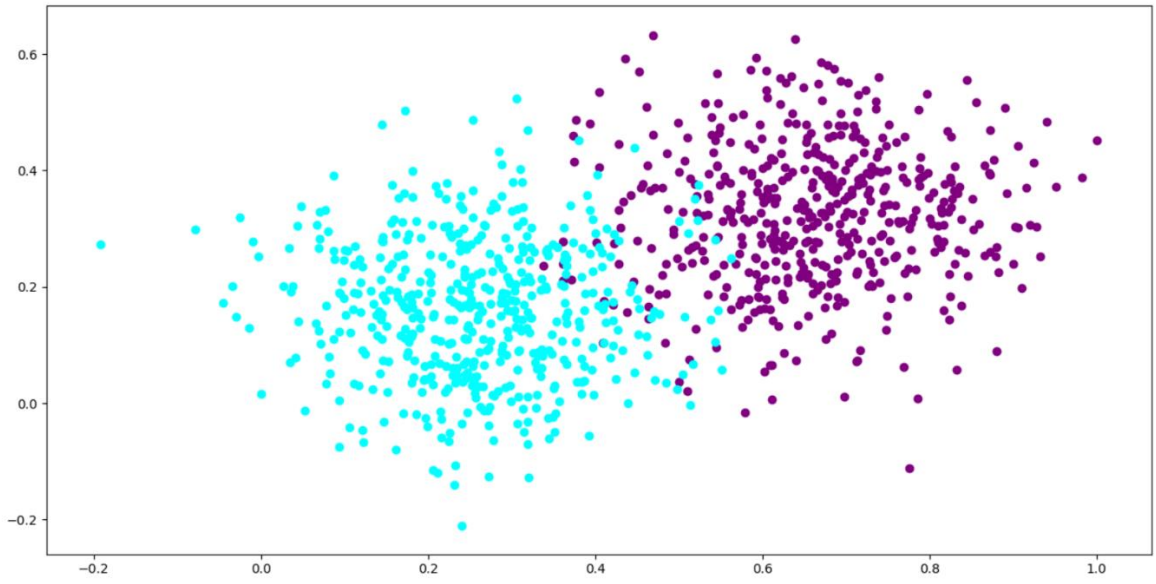
پرهام رحیمی – ۹۵۳۱۰۳۱

ورودی:

با استفاده از data_generator.py می توان داده هایی به تعداد و ابعاد و تعداد کلاس های دلخواه و با توزیع نرمال و به مرکز های رندوم و به شعاع دلخواه تولید کرد. خروجی آن در دو فایل آموزش و تست با فرمت csv، با نام دلخواه ذخیره می گردد. فایل تست به صورت پیش فرض با همان مشخصات فایل آموزش منتهی با ۱۰ برابر تعداد نقاط آن تولید می گردد. به عنوان مثال شکل زیر نمودار داده های تولید شده توسط این برنامه با سایز خوشه‌ی ۵۰ و شعاع نرمال ۱ و تعداد خوشه ۵ و به صورت دو بعدی است:



و شکل زیر نمودار داده های تولید شده توسط این برنامه با سایز خوشه‌ی ۱۰۰ و شعاع نرمال ۱,۵ و تعداد خوشه ۲ و به صورت دو بعدی است:

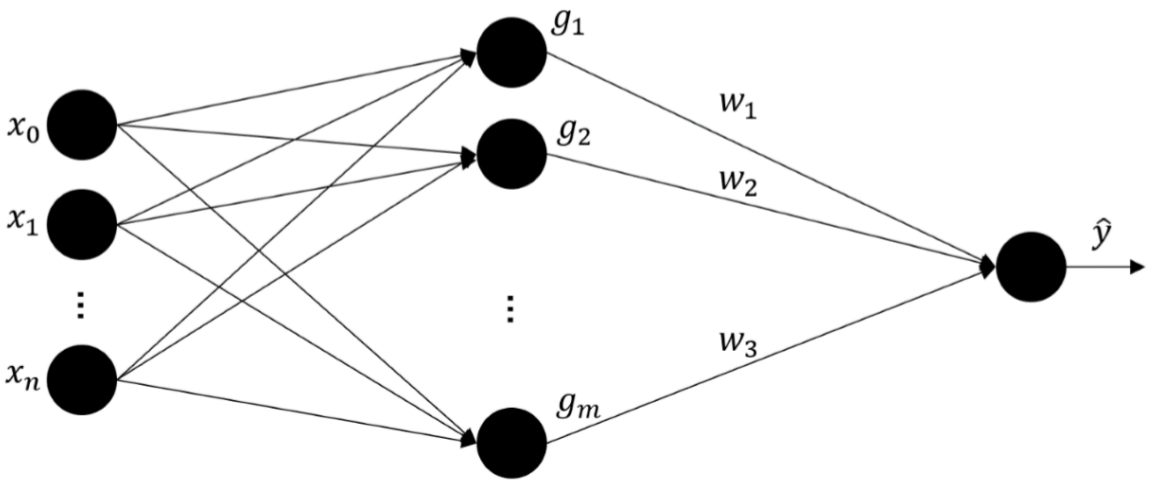


البته برنامه می تواند با سایر داده ها که به فرمت CSV هستند نیز کار کند.

الگوریتم:

در این پروژه شبکه عصبی RBF پیاده سازی شده که به وسیله ی الگوریتم تکاملی (ES) آموزش می یابد.

پیاده سازی:



شبکه عصبی به صورت فوق فرض می شود. هر کدام از ورودی ها (x_i ها) برداریست به طول بعد ورودی. و این n ورودی بر روی m مرکز مپ می شوند. و در نهایت خروجی با توجه به آن ها به دست می آید.

برای الگوریتم تکاملی از ابزار Deap[1] استفاده شده است. هر individual متشکل از {تعداد مرکزها × تعداد بعد های ورودی + تعداد مرکزها} ژن است. که در واقع {تعداد مرکزها × تعداد بعد های ورودی} ژن اول

[1] https://deap.readthedocs.io/en/master/examples/es_fctmin.html

نشان دهنده مختصات مرکز خوشه ها است و {تعداد مرکزها} ژن بعدی نشان دهنده طول شعاع هر کدام از خوشه ها به ترتیب است. برای آنکه ابزار Deap استفاده شده بتواند به درستی و مطابق با خواست ما کار کند تابع error() مطابق با زیر نوشته شد و به جای تابع evaluate این ابزار به آن ورودی داده شد:

```
def error(ind):
    if classification:
        G = []
        W = []
        V = np.array(ind[0:-number_of_groups]).reshape([number_of_groups, dimension])
        for point in train:
            g_row = []
            for group_num in range(number_of_groups):
                difference = np.array(point)[:-1] - V[group_num]
                pow = -ind[number_of_groups * dimension + group_num] * (np.matmul(np.transpose(difference), difference))
                element = np.math.e ** pow
                g_row.append(element)
            G.append(g_row)
        G = np.array(G)
        W = np.array(W)
        Gt = np.transpose(G)
        y = np.array(train)[:, -1]

        a = list(map(int, y))
        a = np.array(a)
        wy = np.zeros((len(train), number_of_groups))
        wy[np.arange(len(train)), a] = 1

        W = np.matmul(np.matmul(np.linalg.pinv(np.matmul(Gt, G)), Gt), wy)

        y_hat = np.matmul(G, W)
        sub = np.subtract(np.argmax(y_hat, axis=1), y)
        return 0.5 * np.matmul(np.transpose(sub), sub),
    else:
        G = []
        W = []
        V = np.array(ind[0:-number_of_groups]).reshape([number_of_groups, dimension])
        for point in train:
            g_row = []
            for group_num in range(number_of_groups):
                difference = np.array(point)[:-1] - V[group_num]
                pow = -ind[number_of_groups * dimension + group_num] * (np.matmul(np.transpose(difference), difference))
                element = np.math.e ** pow
                g_row.append(element)
            G.append(g_row)
        G = np.array(G)
        W = np.array(W)
        Gt = np.transpose(G)
        y = np.array(train)[:, -1]
        try:
            W = np.matmul(np.matmul(np.linalg.pinv(np.matmul(Gt, G)), Gt), y)
        except:
            W = np.random.ranf((number_of_groups, number_of_groups))
        y_hat = np.matmul(G, W)
        sub = np.subtract(y_hat, y)
        return 0.5 * np.matmul(np.transpose(sub), sub),
```

این تابع بسته به آنکه برنامه برای classification صدا زده شود یا regression خروجی یک تاپل می دهد که الگوریتم تکاملی سعی در کمینه کردن آن می کند. به علت اینکه ممکن است G معکوس پذیر نباشد در صورت رخ دادن این اتفاق W ای رندوم به جا W غیر مجاز در نظر میگیریم. در طی نسل ها الگوریتم یاد میگیرد که کمتر این خطا را ایجاد کند.

در نهایت نیز برای آنکه بتوانیم مدل تربیت شده خود را بسنجیم آن را با داده های تست امتحان می کنیم و درصد شباهت خروجی آن را با واقعیت به وسیله تابع accuracy می سنجیم:

```
def accuracy(ind):
    if classification:
        G = []
        W = []
        V = np.array(ind[0:-number_of_groups]).reshape([number_of_groups, dimension])
        for point in test:
            g_row = []
            for group_num in range(number_of_groups):
                difference = np.array(point)[:-1] - V[group_num]
                pow = -ind[number_of_groups * dimension + group_num] * (np.matmul(np.transpose(difference), difference))
                element = np.math.e ** pow
                g_row.append(element)
            G.append(g_row)
        G = np.array(G)
        W = np.array(W)
        Gt = np.transpose(G)
        a = np.array(test)[:, -1]
        a = list(map(int, a))
```

```
        a = np.array(a)
        y = np.zeros((len(test), number_of_groups))
        y[np.arange(len(test)), a] = 1
        try:
            W = np.matmul(np.matmul(np.linalg.pinv(np.matmul(Gt, G)), Gt), y)
        except:
            W = np.random.ranf((number_of_groups, number_of_groups))
        y_hat = np.matmul(G, W)
        a = np.zeros((len(test), dimension + 1))
        for i in range(len(test)):
            if test[i][-1] != np.argmax(y_hat, axis=1)[i]:
                a[i] = test[i]
        green = np.full(len(test), 'green')
        red = np.full(len(test), 'red')
        test_all = test[:, -1] == np.argmax(y_hat, axis=1)
        colors = np.where(test_all, green, red)
        plot_data(test, a, V, colors, ind[-number_of_groups:])
        return np.sum(test_all)
    else:
        G = []
        W = []
        V = np.array(ind[0:-number_of_groups]).reshape([number_of_groups, dimension])
        for point in test:
            g_row = []
            for group_num in range(number_of_groups):
                difference = np.array(point)[:-1] - V[group_num]
                pow = -ind[number_of_groups * dimension + group_num] * (np.matmul(np.transpose(difference), difference))
                element = np.math.e ** pow
                g_row.append(element)
            G.append(g_row)
        G = np.array(G)
        W = np.array(W)
        Gt = np.transpose(G)
        y = np.array(test)[:, -1]
        W = np.matmul(np.matmul(np.linalg.pinv(np.matmul(Gt, G)), Gt), y)
        y_hat = np.matmul(G, W)
        a = np.zeros((len(test), 2))
        for i in range(len(test)):
            a[i][0] = i
            a[i][1] = y_hat[i]
        b = np.zeros((len(test), 2))
        for i in range(len(test)):
            b[i][0] = i
            b[i][1] = test[i][-1]
        scatter_data(b, a)
        return y_hat
```

تنظیمات:

```
number_of_groups = 10
dimension = 2
classification = True

IND_SIZE = number_of_groups * dimension + number_of_groups
MIN_VALUE = -1
MAX_VALUE = 1
MIN_STRATEGY = 0.01
MAX_STRATEGY = 0.5
```

مقدار number_of_groups که در واقع نشان دهنده تعداد مرکز هاست و dimension که نشان دهنده ابعداد ورودی هاست، بسته به ورودی تعیین می شوند. بولین classification نیز بسته به اینکه سوال از جنس classification است یا regression، به ترتیب مقدار True و False می پذیرد. Individual size با توجه به number of groups و dimension به شکلی که گفته شد تعیین می شود. با توجه به اینکه داده های گرفته شده در ابتدا نرمال می شوند (به بازه -۱ تا ۱ مپ می شوند) MIN_VALUE و MAX_VALUE، به ترتیب -۱ و ۱ قرار داده می شود. و به همین نسبت MIN_STRATEGY و MAX_STRATEGY، ۰,۰۱ و ۰,۵ قرار داده می شوند که با توجه به چند آزمون و خطا خروجی مناسب تری داشتند. در انتها ngen که تعداد نسل ها است نیز با توجه به سختی مسئله و تعداد نسل های لازم برای همگرا شدن تعیین می شود.

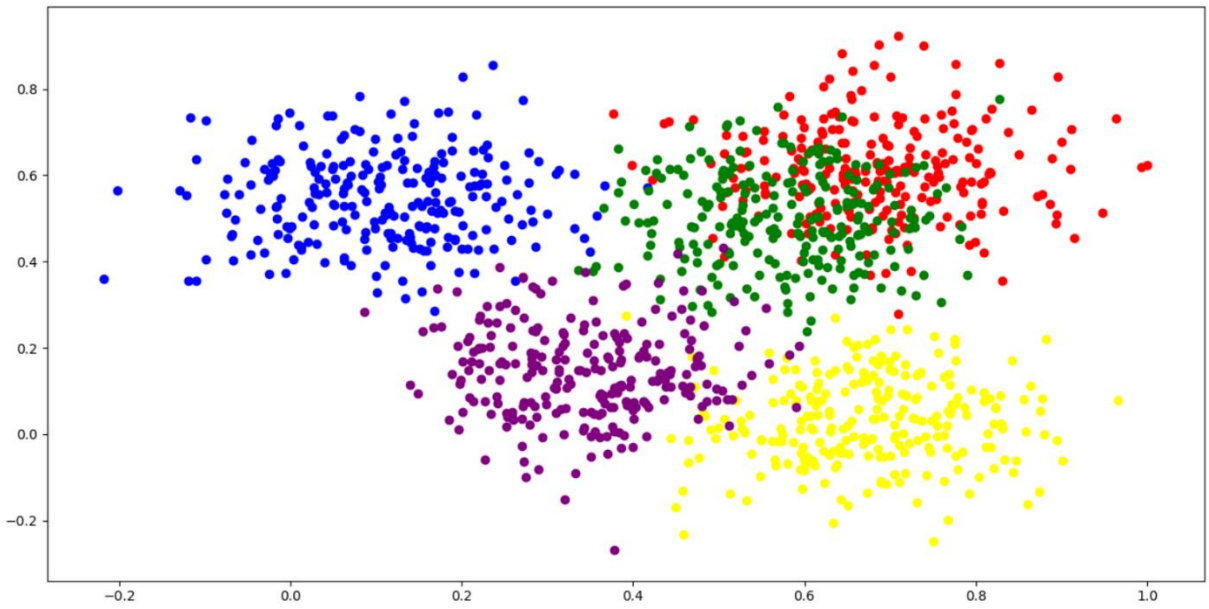
خروجی:

در انتها نیز خروجی به شکل نمودار داده های تست با رنگ های سبز (برای نقاطی که شبکه ما به درستی آن ها را تشخیص داده) و قرمز (برای نقاطی که شبکه ما به اشتباه آن ها را تشخیص داده) به همراه درصد دقت نشان داده می شود:

مثال یک:

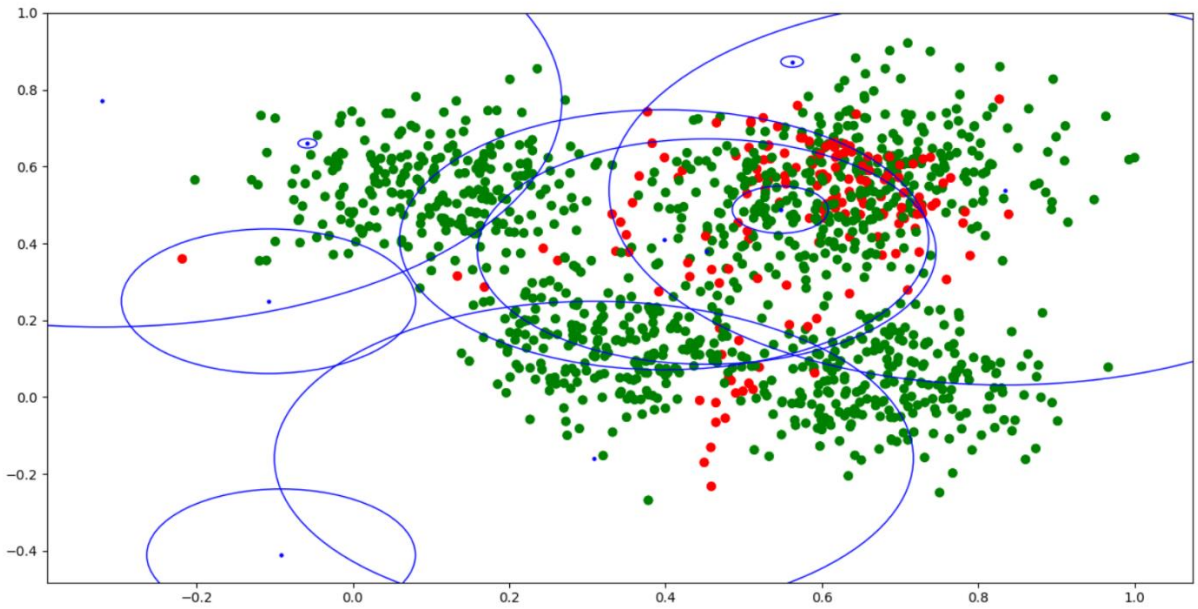
داده های ورودی (train.csv و test.csv):

تعداد کلاس ها: ۵



تشخیص شبکه عصبی ما:

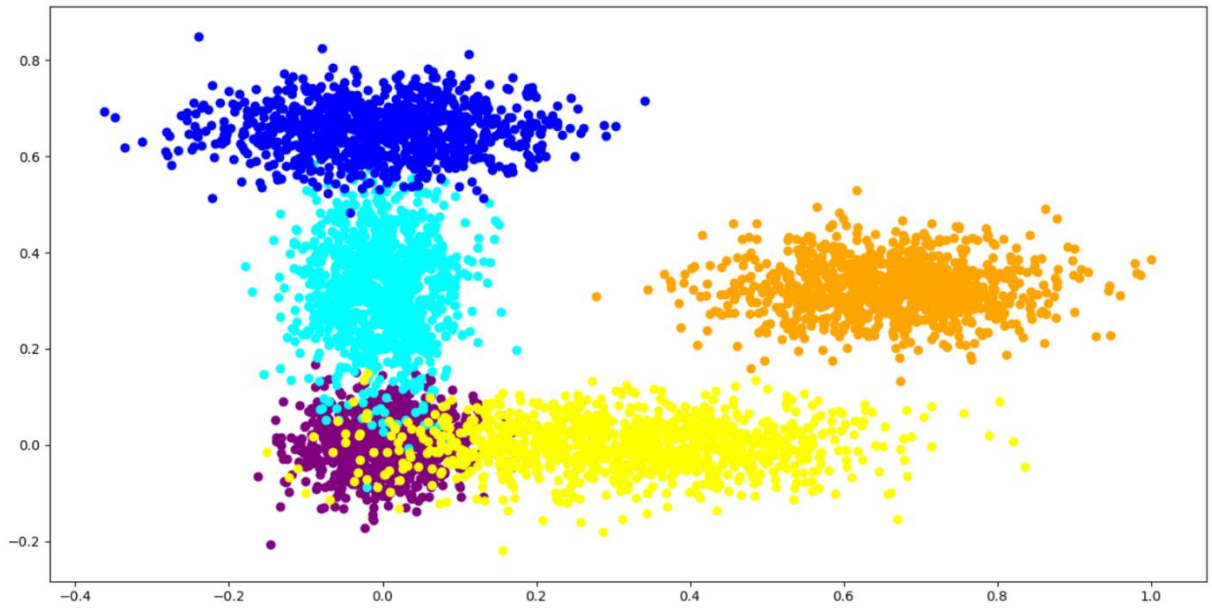
تعداد نسل ها: ۱۵ | تعداد مرکز ها: ۱۰ | جمعیت هر نسل: ۱۰ | تعداد فرزندان هر نسل: ۱۰۰ | دقت: ۹۷,۱۹٪



مثال دو:

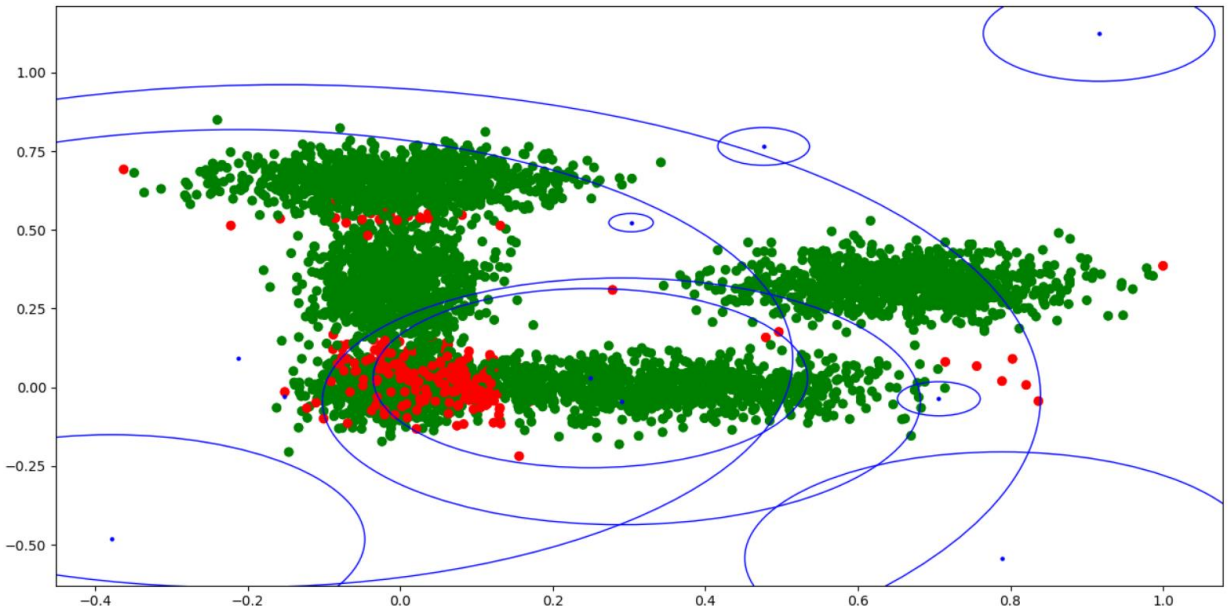
داده های ورودی (5clstrain1500.csv و 5clstest5000.csv):

تعداد کلاس ها: ۵



تشخیص شبکه عصبی ما:

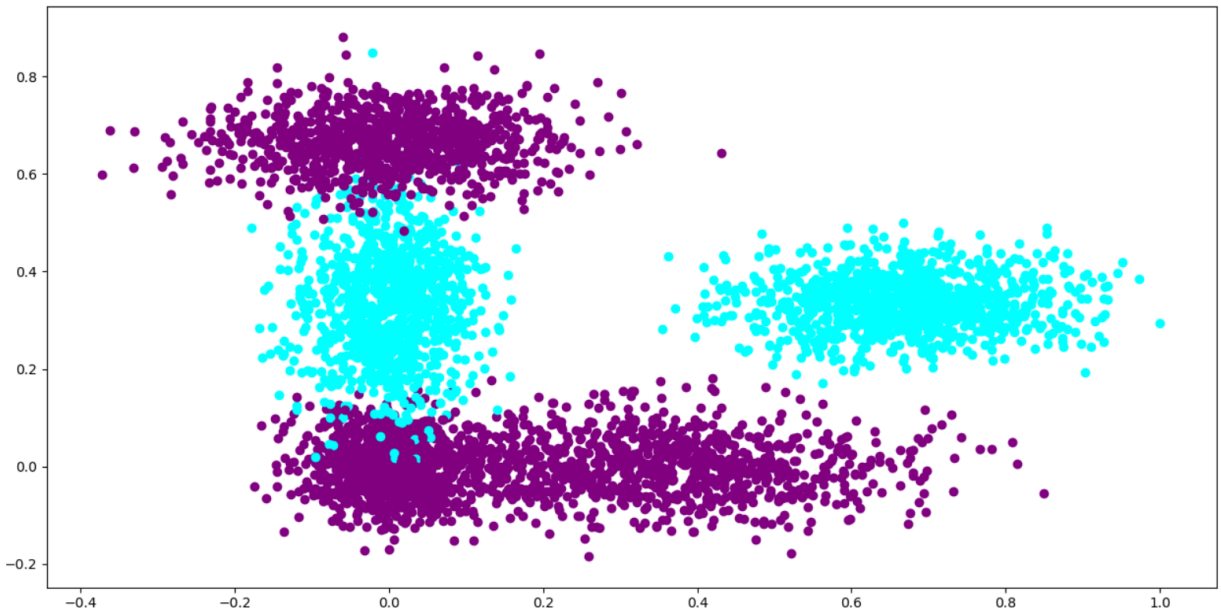
تعداد نسل ها: ۲۰ | تعداد مرکز ها: ۱۰ | جمعیت هر نسل: ۱۰ | تعداد فرزندان هر نسل: ۱۰۰ | دقت: ۹۵,۳۷٪



مثال سه:

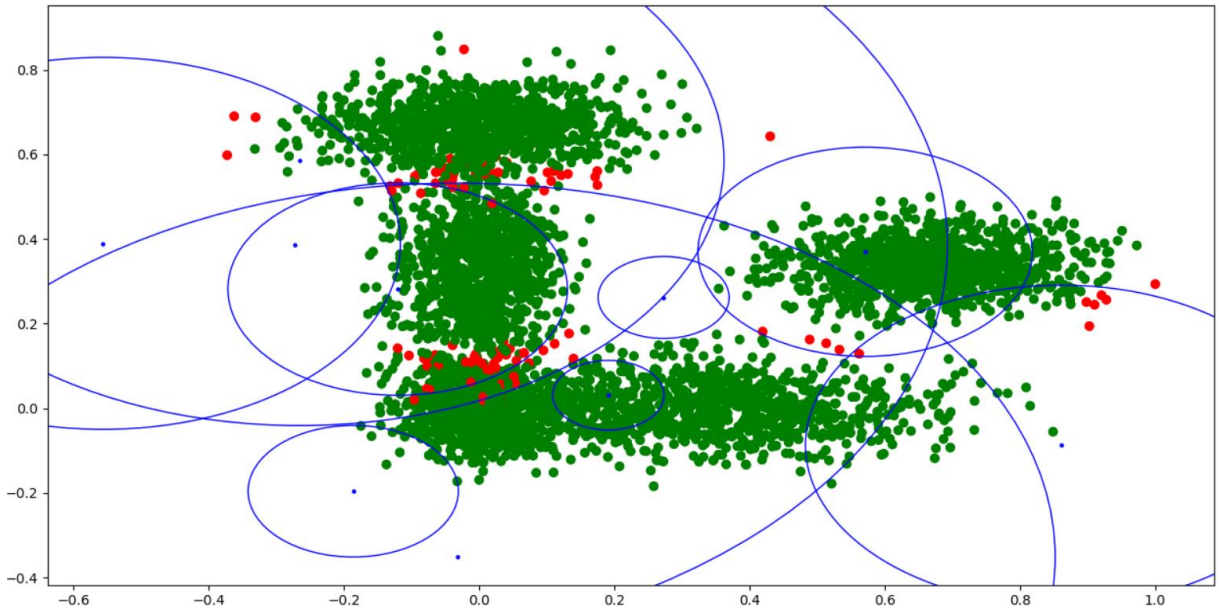
داده های ورودی (2clstrain1500.csv و 2clstest5000.csv):

تعداد کلاس ها: ۲



تشخیص شبکه عصبی ما:

تعداد نسل ها: ۱۰ | تعداد مرکز ها: ۱۰ | جمعیت هر نسل: ۱۰ | تعداد فرزندان هر نسل: ۱۰۰ | دقت: ۹۸,۶۱٪



مثال چهار:

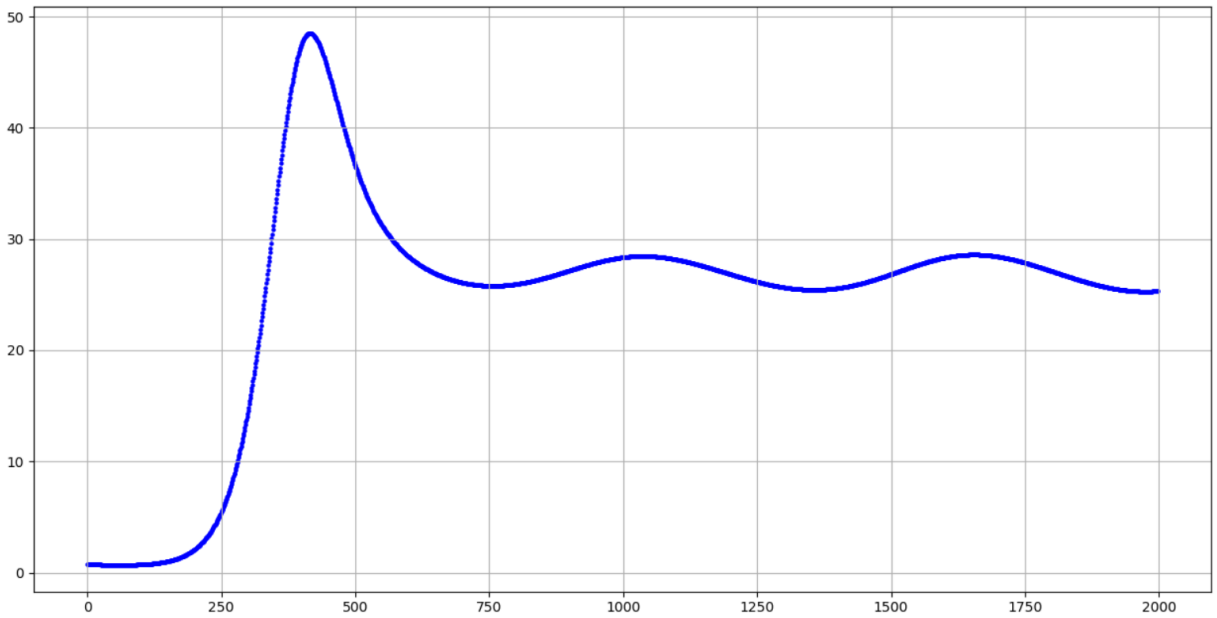
داده های ورودی (reg.csv):

تشخیص شبکه عصبی ما:

تعداد نسل ها: ۲۰ | تعداد مرکز ها: ۱۰ | جمعیت هر نسل: ۱۰ | تعداد فرزندان هر نسل: ۱۰۰

مثال پنج:

داده های ورودی (**regdata2000.csv**):



تشخیص شبکه عصبی ما:

تعداد نسل ها: ۲۰ | تعداد مرکز ها: ۱۰ | جمعیت هر نسل: ۱۰ | تعداد فرزندان هر نسل: ۱۰۰