Computational Intelligence

Report: Final Project

–

Ali Izadi 9431001

اهداف پروژه:

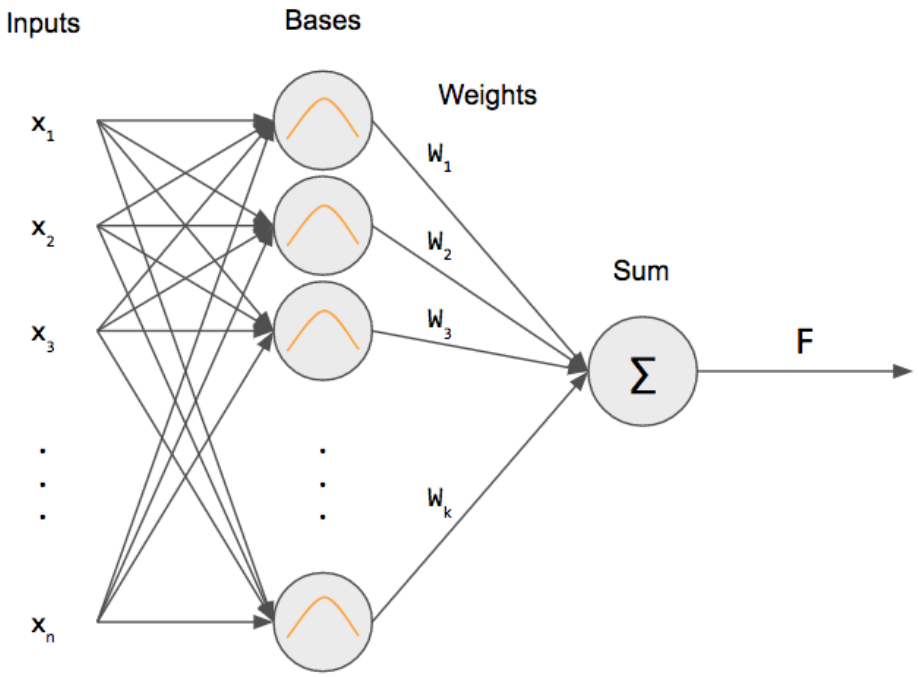
هدف از این پروژه آشنایی با پیاده سازی شبکه عصبی مصنوعی و آموزش آن با استفاده از الگوریتم تکاملی است.

توضیحات:

شبکه عصبی RBF یک شبکه عصبی سه لایه است که لایه میانی آن از تابع فعال سازی RBF استفاده می‌کند.

یک شبکه عصبی سه لایه با تابع فعال سازی خطی تنها می‌تواند کلاس‌هایی را که صورت خطی جداپذیر(linearly separable) هستند را از هم تشخیص دهد اما تابع فعال سازی RBF این امکان را میدهد تا کلاس‌هایی که جداپذیر خطی نیستند را نیز تشخیص داد.

معماری این شبکه عصبی مطابق زیر است:

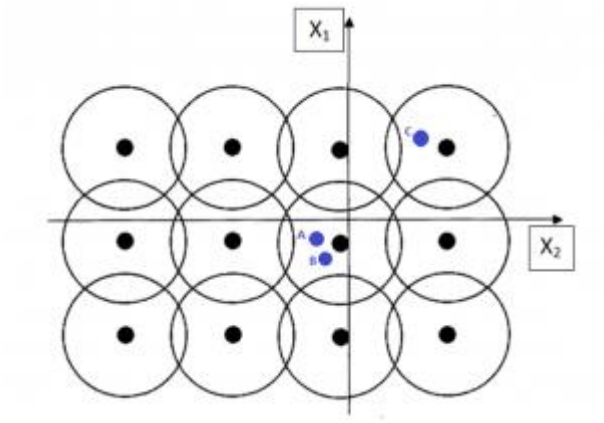


تابع RBF مطابق فرمول روبه‌رو تعریف می‌شود:

$$g_i = e^{-\gamma_i (x-v_i)^T (x-v_i)}$$

g_i مشخص‌کننده مقدار هر نورون در لایه دوم است. این تابع دو پارامتر V_i و سیگما دارد که برای محاسبه ورودی X_i استفاده میشود. با فرض دانستن این پارامترها این تابع میزان فاصله اقلیدسی ورودی را با مرکز خود می‌سنجد. اگر این فاصله زیاد باشد خروجی تابع یا فعال سازی آن کم است بنابراین هر نورون میتواند به عنوان معیاری برای نزدیکی ورودی استفاده شود با این شرط که مرکز آن نمایانگر داده‌های اطراف آن باشد.

به عنوان مثال فرض کنیم مراکز یادگرفته شده در مختصات زیر نقاط مشکی رنگ باشند. در نتیجه نقطه اول از سمت راست بالا هنگام ورودی نقطه c به شبکه مقدار فعال سازی بیشتری دارد چون به آن نزدیک تر است و این مرکز میتواند در زمان پیش بینی به عنوان نماینده ای برای نقاط همانند C در classification یا regression استفاده شود. به همین ترتیب نقطه مشکی رنگ ردیف دوم ستون دوم از راست به پایین نیز نمایانگر نقاط A و B خواهد بود.



در ادامه وزن های بین لایه دوم و سوم نیز بیانگر تاثیر هر یک از نقاط مرکزی در محاسبه خطای آموزش هستند. خطای شبکه مطابق با فرمول روبه‌رو محاسبه میشود.

$$L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^T (\hat{y} - y)$$

که با اعمال شرط صفر شدن و با روش pseudo inverse مقادیر وزن های W نیز به دست می‌آید که از آن میتوان برای محاسبه y پیش‌بینی شده مطابق با فرمول روبه‌رو استفاده کرد:

$$\hat{y} = GW$$

بنابراین تنها وزن‌های بین لایه اول نیاز به آموزش دارند. برای آموزش مراکز و پارامتر سیگما از الگوریتم های ژنتیک استفاده میکنیم.

استراتژی های تکامل نوعی black-box optimization هستند. در این الگوریتم فرض بر وجود تعدادی پارامتر است که شایستگی آن ها با استفاده از تابعی مشخص میشود. با استفاده از مقادیری رندوم برای این پارامترها شایستگی آن‌ها محاسبه میشود و سعی میشود به استفاده از توزیع گوسی مقادیر جدیدی تولید شود که پارامترها به سمت بهینگی بروند.

در این پروژه نیز پارامترهای بهینه همان مراکز و سیگما هستند که به الگوریتم Evolutionary Strategy داده میشود و شایستگی آن‌ها با استفاده از خطای شبکه عصبی برای پیشروی الگوریتم ES استفاده می‌شود.

به طور کلی شبه کد الگوریتم‌های ES مطابق زیر است:

```
solver = EvolutionStrategy()
while True:

  # ask the ES to give us a set of candidate solutions
  solutions = solver.ask()

  # create an array to hold the solutions.
  # solver.popsize = population size
  rewards = np.zeros(solver.popsize)

  # calculate the reward for each given solution
  # using your own evaluate() method
  for i in range(solver.popsize):
    rewards[i] = evaluate(solutions[i])

  # give rewards back to ES
  solver.tell(rewards)

  # get best parameter, reward from ES
  reward_vector = solver.result()
```

```
if reward_vector[1] > MY_REQUIRED_REWARD:
  break
```

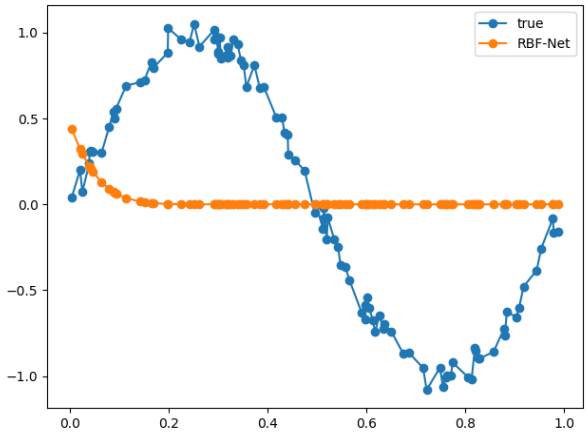
در ابتدا به بررسی نتایج پیاده سازی برای مسئله regression خواهیم پرداخت.

برای تست شبکه عصبی تابع sin را با استفاده از داده‌های تولید شده همراه با نویز ایجاد کردیم و این داده‌ها را به عنوان داده‌های آموزش به شبکه دادیم تا تابع را با استفاده از مختصات آن مدل کنیم. در واقع مختصات y هر داده x را پیش بینی کنیم.

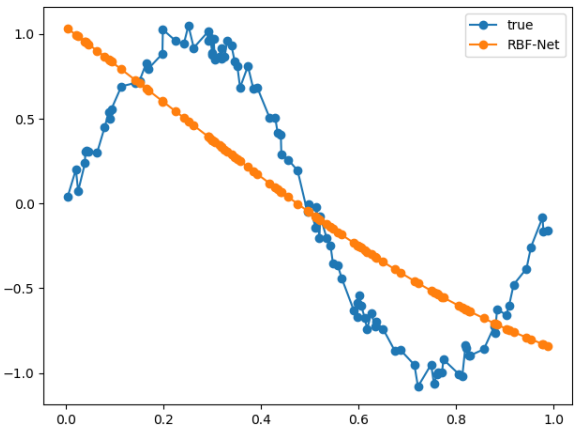
با استفاده از ۲۰ iteration برای الگوریتم ژنتیک fitness آن در مرحله آخر مطابق با زیر به دست آمد:

best fitness at iteration: 20 -0.0002832897522148558

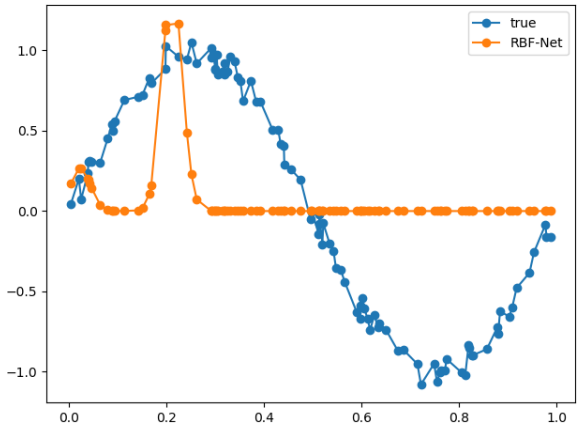
نکته ای که در آموزش و تخمین تابع وجود داشت تعداد نورون های لایه دوم یا تعداد مراکزی بود که صفحه مختصات را مدل میکردند. در نمودارهای زیر این نکته نشان داده شده است. به صورتی که تعداد نورون های از ۱ تا ۵ به ترتیب افزایش داده شده اند تا در نهایت تعداد نورون‌های ۵ توانسته اند تابع sin را مدل کنند. نقاط آبی رنگ در هر شکل داده‌های آموزش و نقاط نارنجی نقاط پیش بینی شده هستند.



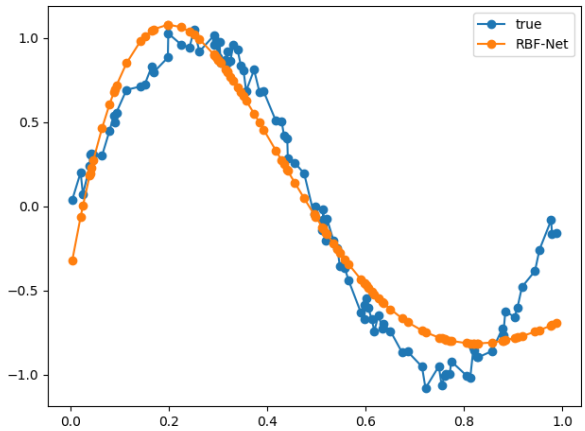
1 neuron in hidden layer



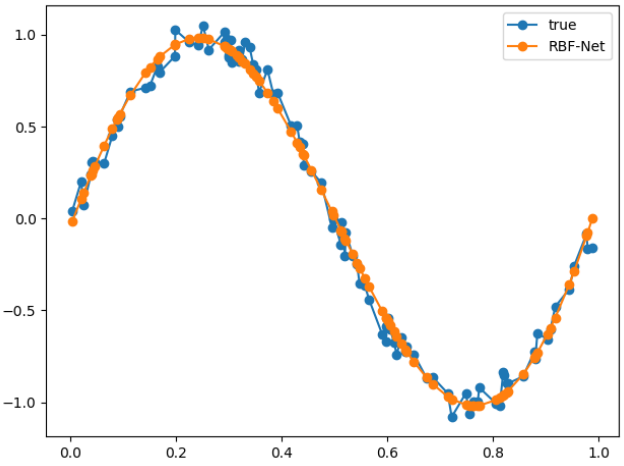
2 neurons in hidden layer



3 neurons in hidden layer

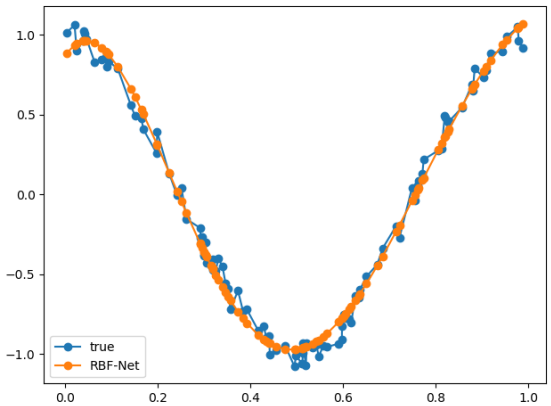


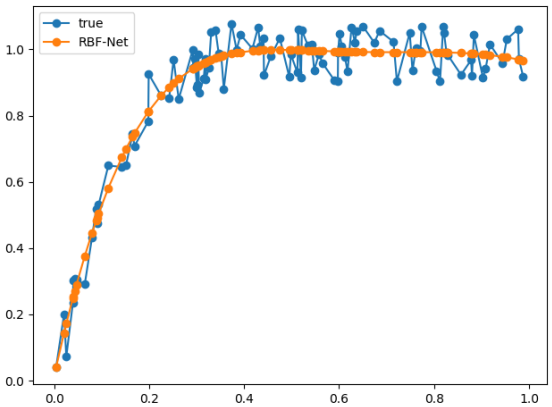
4 neurons in hidden layer



5 neurons in hidden layer

شبکه عصبی آموزش داده شده مطابق بالا نیز برای توابع cos و tan نیز تست شد که نتایج آن مطابق شکل‌های زیر است. برای مدل کردن tan نیز تنها دو neuron در لایه میانی کافی است.





شبکه عصبی توضیح داده شده برای مسئله regression استفاده میشود. برای این که شبکه بتواند برای multi label classification استفاده شود. نیاز است تا لایه آخر به جای یک نورون به تعداد کلاس ها نورون استفاده کند. و خروجی هر نورون در لایه آخر با استفاده از تابع فعال سازی softmax محاسبه شود تا مقدار آن به عنوان احتمال تعلق هر نورون در لایه آخر به آن کلاس استفاده شود.

Softmax:

$$y_i(z_i) = \frac{e^{z_i}}{\sum_{k=1}^{k} e^{z_k}}$$

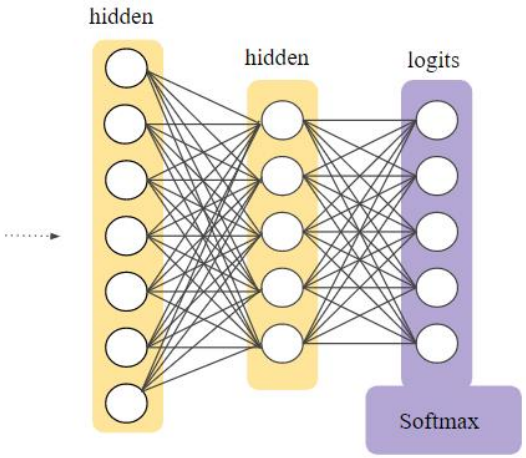
خطا با استفاده از فرمول روبه‌رو محاسبه شود.

$$H(y, \hat{y}) = -\sum_i y_i \log \hat{y}_i$$

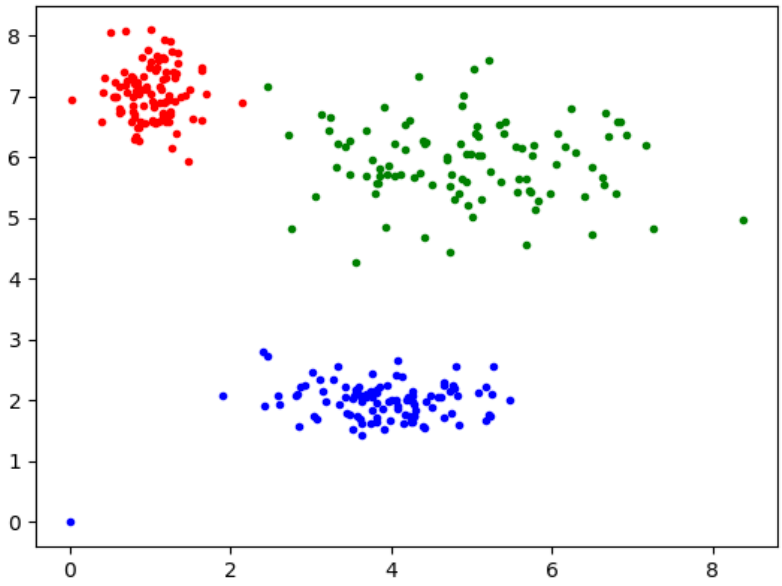
مقادیرtarget داده‌های آموزش به one hot encoding تبدیل شده‌اند تا بتوان در فرمول بالا برای محاسبه خطا از آن استفاده کرد.

در نهایت خروجی نهایی برای شبکه شماره نورونی است(n) که بیشترین احتمال بین ۰ و ۱ به آن تعلق گرفته است که نشان دهنده پیش بینی شبکه برای کلاس n است.

معماری شبکه عصبی چند کلاسه مطابق با زیر است:

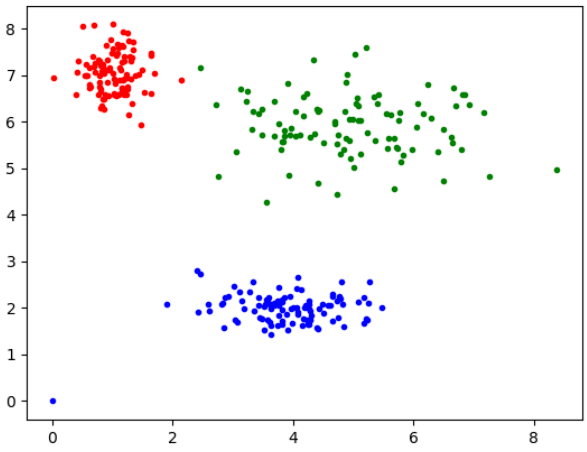


برای تست این شبکه داده های زیر در مختصات دو بعدی تولید شدند که داده سه کلاس را نشان می‌دهد که جداناپذیر خطی هستند.



داده‌های اصلی

با آموزش الگوریتم داده‌های آموزش با دقت ۱۰۰ درصد مطابق شکل زیر پیش بینی شدند و خطای آموزش صفر است.

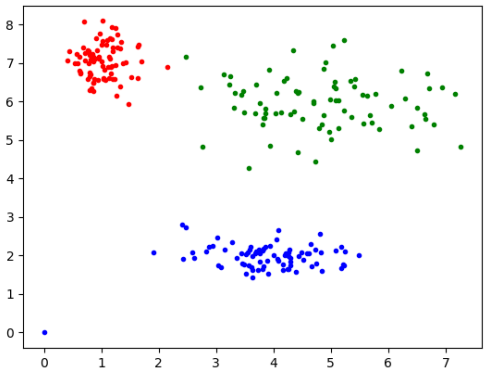


پیش بینی داده‌های آموزش

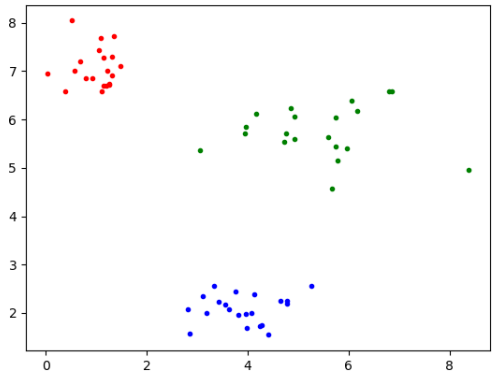
در ادامه داده‌های موجود با نسبت ۲۰ درصد به داده‌های آموزش و تست تقسیم شدند تا دقت الگوریتم برای داده‌های جدید تست شود.

همان طور که در شکل زیر دیده می‌شود داده های شکل سمت چپ به عنوان داده‌های آموزش به شبکه داده شدند و داده‌های سمت راست نیز با دقت ۱۰۰ درصد به عنوان داده‌های تست پیش بینی شدند.

برای آموزش این شبکه از ۴ نورون برای لایه میانی استفاده شده است.

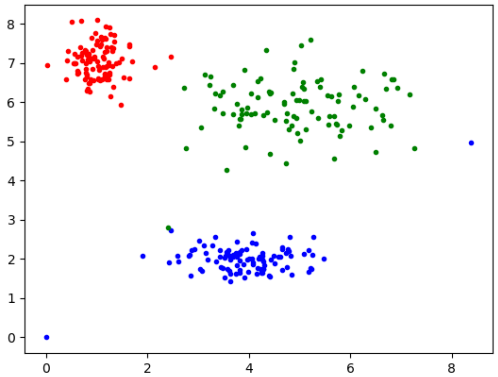


Train data (%80)

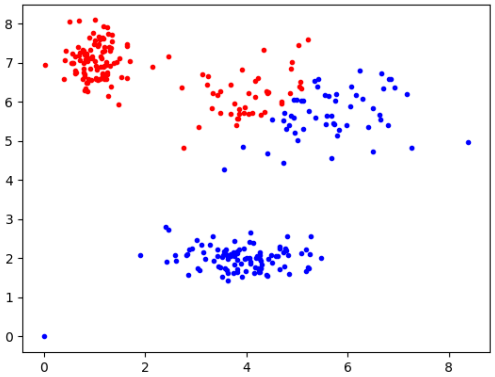


Test data (%20)

در ابتدا شبکه با ۲ و ۳ نورون میانی آموزش داده شد که به ترتیب دقت پیش‌بینی داده‌های آموزش برابر با ۶۷ و ۹۹ درصد به دست آمد. در شکل های زیر نتیجه دسته بندی با استفاده از ۲ و ۳ نورون مشخص شده است.



2 neurons in hidden layer



3 neurons in hidden layer