بسمه تعالی

دانشکده مهندسی کامپیوتر

آبان ۱۴۰۰

مبانی هوش محاسباتی

نام استاد: دکتر مزینی

تمرین دوم

آرمان حیدری

شماره دانشجویی: ۹۷۵۲۱۲۵۲

۱. پاسخ سوال اول

توضیحات دقیق مربوط به هر تابع در نوتبوک داده شده است. الگوریتم به این صورت است که ابتدا آرایه ای از مختصات رندومی به عنوان وزن های شبکه ایجاد میکنیم. تعداد این وزن ها عددی دلخواه است اما اینجا با حدس این که تعداد پیکسل های سیاه حدودا یک سوم سفید ها میباشد این تعداد را یک سوم ابعاد عکس اصلی در نظر گرفته ام.

بعد در هر epoch، یکی از پیکسل های عکس را به صورت رندوم انتخاب میکنیم. نزدیک ترین مختصات در آرایه وزن به آن را پیدا میکنیم و آن را نورون برنده می نامیم.

حال وزن هایی که در شعاع تعیین شده هستند را پیدا میکنیم و آن ها را بر اساس فرمول کوهونن آپدیت میکنیم.

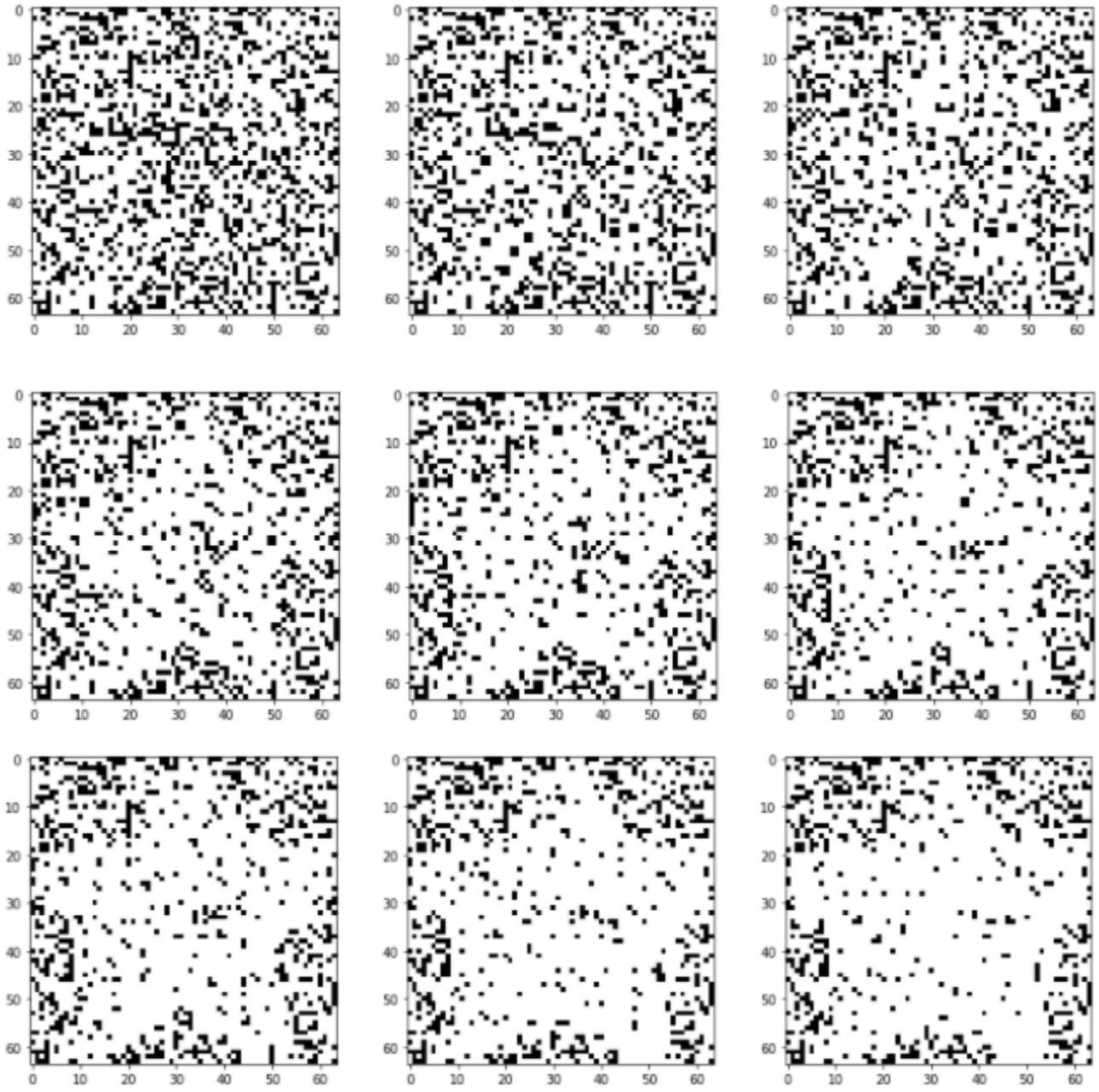
$$\Delta w_j = \eta h_{j,i(x)}(x - w_j)$$

X همان مختصات پیکسل انتخاب شده از عکس است و h هم که با رابطه زیر به دست می آید:

$$h_{j,i(x)} = e^{\frac{-d_{j,i}^2}{2\sigma^2}}$$

که d همان فاصله اقلیدسی بین وزنی که میخواهد آپدیت شود و نورون برنده است. اما اتا یه همان نرخ یادگیری محل ایده ما برای حل این سوال است. چون در این سوال هدف نزدیک شدن وزن ها به مقادیر پیکسل های سیاه است و نه سفید، آرایه میسازیم که مقادیر آن برای پیکسل های کاملا سیاه (که در عکس grayscale با ۰ مقداردهی شده اند) ۱ باشد و برای کاملا سفید ها هم صفر باشد. این آرایه را با رابطه 1-input/255 میسازیم که input همان عکس ورودی است. پس از مقدار سیاهی پیکسلی که به صورت رندوم انتخاب کرده بودیم به عنوان نرخ یادگیری یا اتا استفاده میکنیم و به این صورت وزن ها که نقاطی سیاه را نمایش میدهند به نقاط سیاه عکس اصلی نزدیک و نزدیک تر میشوند.

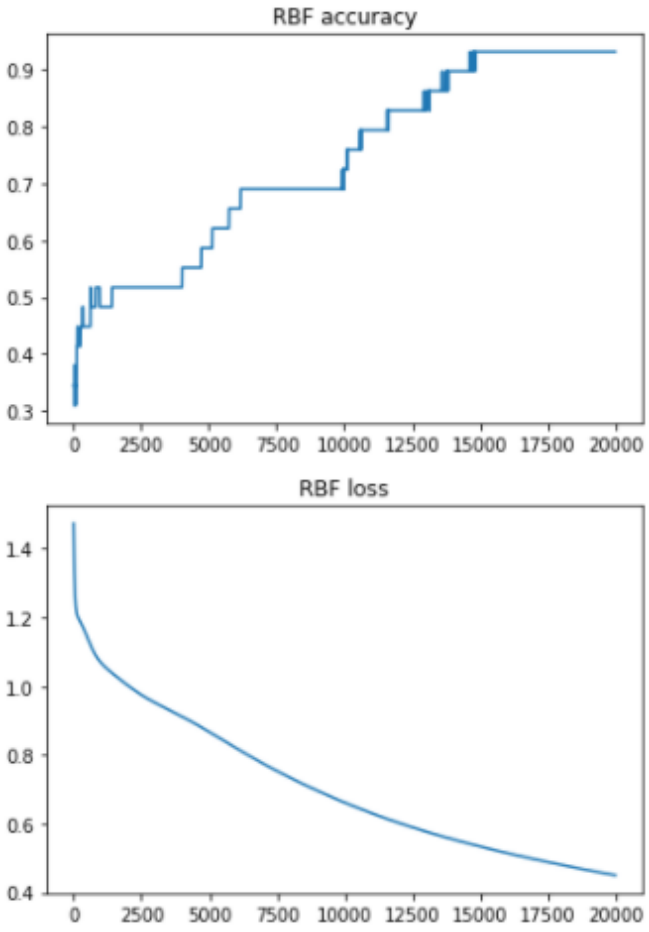
چون در هر بار تکرار این کار را فقط برای یک پیکسل انجام میدهیم نیاز به تکرار بسیار زیادی برای نتیجه خوب وجود دارد. من این شبکه را با epoch=20000 و شعاع ۲.۵ اجرا کردم و خروجی ها که به ترتیب از بالا سمت چپ مربوط به شروع یادگیری و پایین سمت راست مربوط به پایان آن است به این شکل به دست آمد:



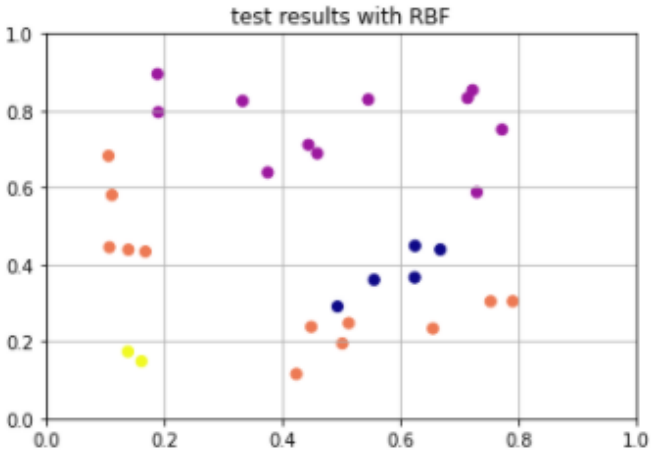
روند یادگیری الگوریتم مشخص است و احتمالا با اجرای بیشتر به عکس اصلی میرسد.

۲. پاسخ سوال دوم

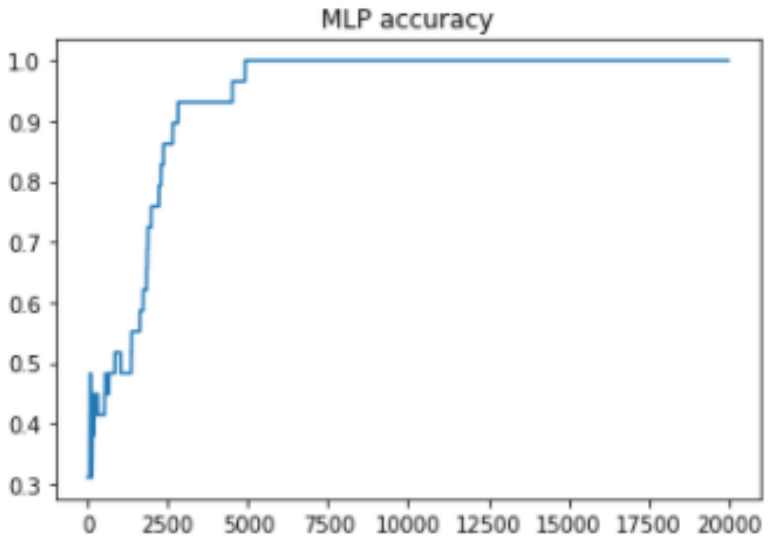
۲.۱) توضیحات مربوط به مراحل کد و انتخاب پارامترها در نوتبوک داده شده است. نتایج مقدار ضرر و دقت شبکه به این صورت بود:

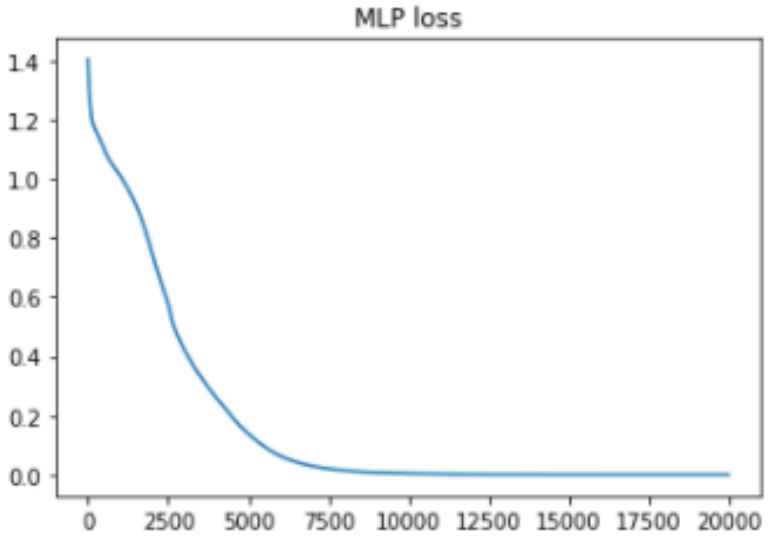


سرعت همگرایی شبکه بسیار پایین بوده است. و حتی بعد از ۲۰۰۰۰ اجرا هم نمیتوان با قطعیت گفت که همگرا شده است. اما رسیدن به دقت ۹۳ درصد اتفاق خوبی است. حالت پله ای نمودار دقت به علت کم بودن دیتا و در نتیجه تغییر ناگهانی دقت با اضافه شدن یک یا چند حدس صحیح است. داده تست که متشکل از ۳۰ نقطه دلخواه در بازه گفته شده بود هم به این شکل با این شبکه کلاسبندی شده است:

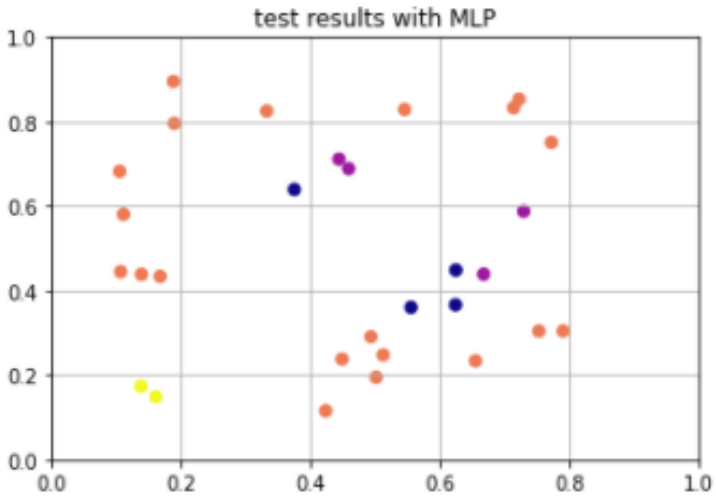


۲.۲) هایپرپارامتر های شبکه را دقیقا مانند قبل تنظیم کرده و فقط به جای لایه rbf از لایه معمولی با همان تعداد نورون استفاده میکنیم. نتایج مقدار ضرر و دقت شبکه به این صورت بود:



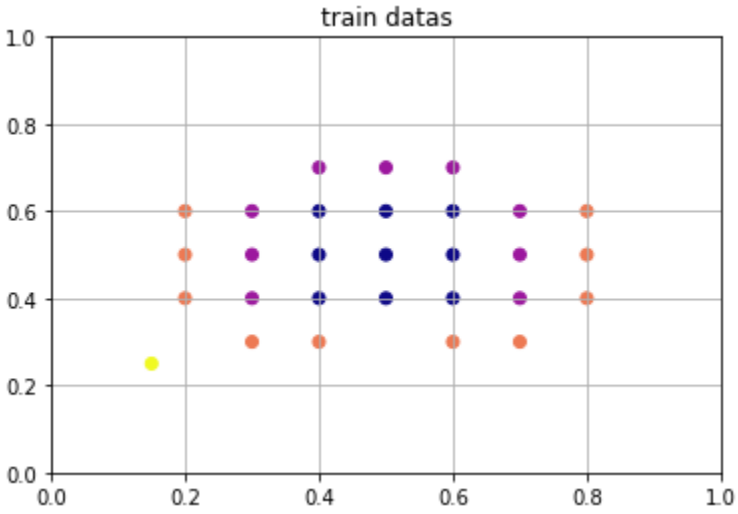


میبینیم حدودا از epoch=6000 شبکه به دقت ۱۰۰ درصد روی داده آموزشی رسیده است و از نظر آموزش ایرادی بر آن وارد نیست. حالت پله ای نمودار دقت در ابتدا به علت کم بودن دیتا و در نتیجه تغییر ناگهانی دقت با اضافه شدن یک حدس صحیح است. حالا خروجی شبکه برای داده تست هم میبینیم:



۲.۳) از نظر داده آموزشی چون شبکه MLP قابلیت تطبیق پذیری بیشتری دارد و ضرایبش خیلی زود میتوانند طوری تعیین کند که با داده فیت شوند زود به دقت ۱۰۰ درصد میرسیم و به نوعی این شبکه داده ها را که کلا ۳۰ عدد هستند حفظ میکند. چون این تعداد داده تمرینی برای یک شبکه MLP با دو لایه بسیار کم است. البته دقت ۹۳ درصد شبکه RBF هم مطلوب است هرچند خیلی دیر به این دقت رسیده و پیدا شدن پارامتر های مناسب برای این شبکه بسیار زمانبر تر است.

وقتی شکل خروجی روی داده تست این دو شبکه را با شکل ورودی اصلی مقایسه میکنیم :



کلاس unknown را هر دو شبکه به خوبی جدا کرده اند. اما شبکه RBF در زمینه تشخیص داده های آبی پررنگ (کلاس ۰) به نظر بهتر از MLP بوده که تعداد کمی را ازین کلاس حدس زده و عمده نقاط را کلاس ۲ (کرمی رنگ) در نظر گرفته است. مثلا در y های زیاد ما داده های آموزشیمان از رنگ بنفش بوده اند و نه کرمی که این موضوع در شبکه RBF بهتر تفکیک شده است. در کل شبکه RBF به دلیل ویژگی های تطبیق پذیری خوب با داده های نویزی و جدید، generalization قوی در این مسئله بهتر عمل کرده اند.

۳. پاسخ سوال سوم

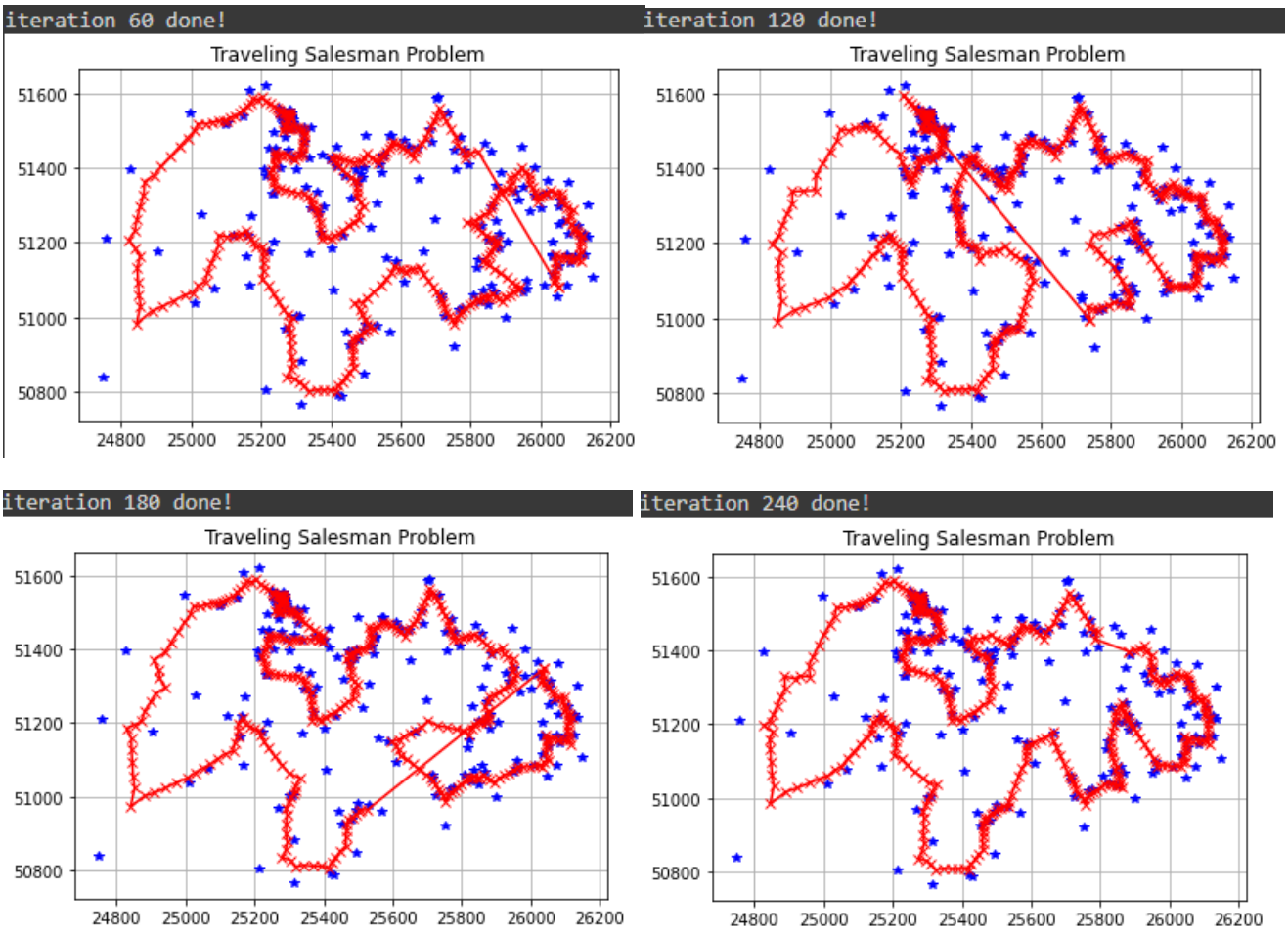
- RBF : بله می‌توانیم جداسازی را انجام دهیم. اتفاقاً بهترین روش برای تفکیک این داده ها احتمالاً RBF است.
چون این سؤال به سؤال دوم شبیه است. جایی که با mapping درب این داده ها وانتخاب تابع φ مناسب می‌توان آن ها را به فضایی برد که براحتی تفکیک پذیر باشند. زیرا می‌بینیم که داده های با کلاس یکسان در این فضا نزدیک به هم هستند و صرفاً برای جدا کردن نشان کافیست طوری جابجا شوند که از کلاس های دیگر تمایز بیشتری پیدا کنند. استفاده از (مختصات دوتا لایه ورودی 2 واحد دارد و یک لایه مخفی RBF با حدود 10 تا نورون که به لایه خروجی Dense با 4 نورون و تابع فعالسازی softmax متصل است، انتخاب مناسبی است. البته رابطه تابع φ نباید خطی باشد چون این فضا به صورت خطی تفکیک پذیر نمی‌باشد. در واقع quadrically seperable هستند. اگر t_i مرکز و σ_i عرض پارامترها باشد، برای هر نورون i (گاوسی) $\varphi_i(x) = e^{\frac{-\|x-t_i\|^2}{\sigma_i^2}}$ تعریف می‌کنیم که نگاشت را به فضای جدیدی ببرد. که مقادیر t_i و σ_i را می‌توانیم آموزش دهیم یا خودمان با توجه به مختصات تعریف کنیم

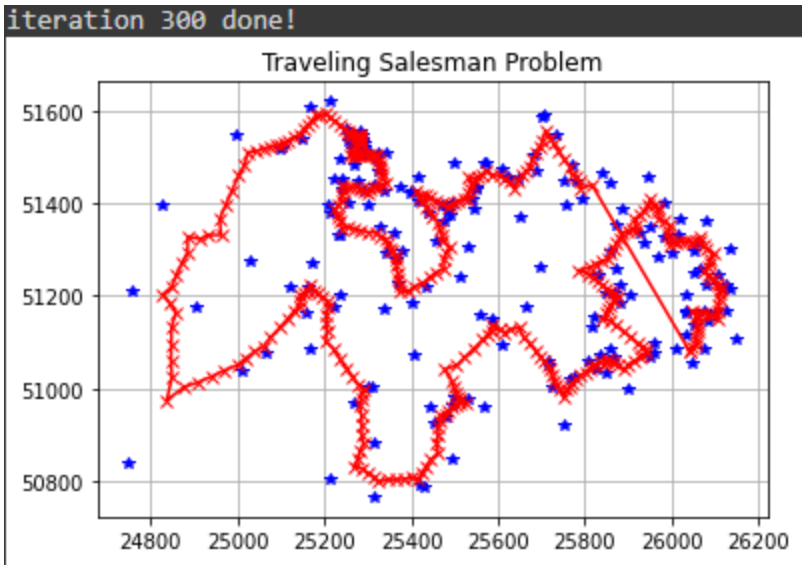
- SOFM : این سؤال را نمی‌توان با kohonen حل کرد. چون برای حل مسائل کلاس بندی و به خصوص چندین کلاسه این شبکه مناسب نیست. از این نوع شبکه ها استفاده های دیگری مثلاً در مسائل مسیریابی یا بازسازی یک تصویر یا شاید صوت استفاده می‌شود.

- MLP : با این نوع شبکه تقریباً هر نوع سؤال را می‌توان حل کرد و این سؤال هم استثنا نیست. لایه ورودی و خروجی مانند RBF خواهند بود. یعنی لایه ورودی دو نورون که مختصات هر نقطه است، ولایه خروجی 4 نورون با تابع فعالسازی softmax. لایه میانی می‌تواند یک یا دو تا یا نهایت سه تا باشد. تعداد نورون های این لایه عددی بین 10 تا 20 معقول خواهد بود.
  - مقادیر w و b برای هر نورون باید طی آموزش به دست بیایند. استفاده از تابع ضرر categorical crossentropy هم چون چندین کلاس داریم مناسب است.

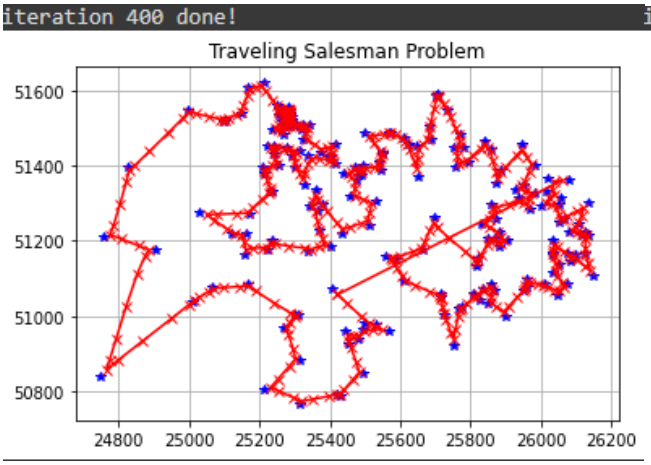
۴. پاسخ سوال چهارم

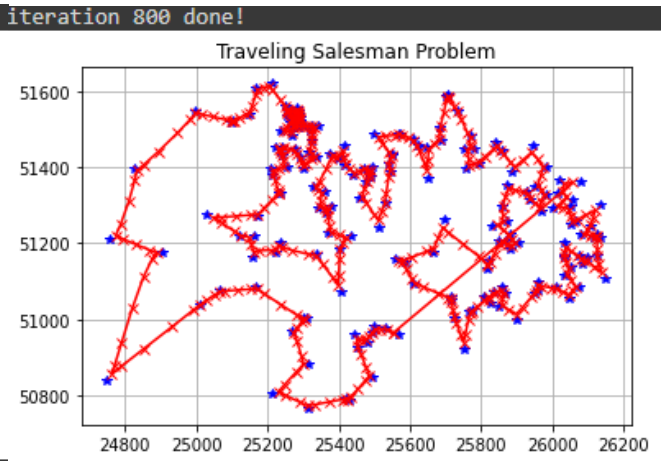
در نوتبوک توضیحات مربوط به هر کلاس و تابع و عملکردش آورده شده است. با نقشه ی کوهونن وزن هایی را مقدار دهی اولیه میکنیم، سپس با روابطی که از کوهونن میدانیم آن ها را بر اساس نزدیکی به شهر هایی که مختصاتشان را میدانیم آپدیت میکنیم. با تکرار این کار مسیری که مینیمم مسافت باید طی شود تا از همه شهر ها بگذریم مشخص میشود. برای مثل الگوریتم را با ۳۰۰ تکرار (Nk=300) روی داده هایی که در فایل data.csv داده بودید اجرا کردم و تصاویر اجرا به ترتیب پس از بار ۶۰، ۱۲۰، ۱۸۰، ۲۴۰، ۳۰۰ به صورت زیر شد که مشخص است map در حال یادگیری و بهتر شدن است:

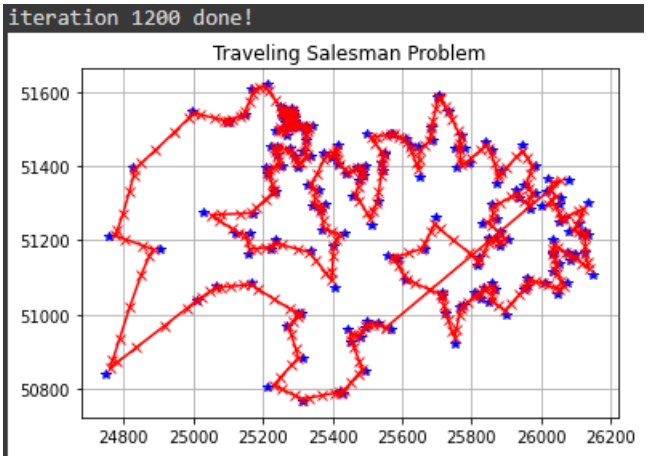


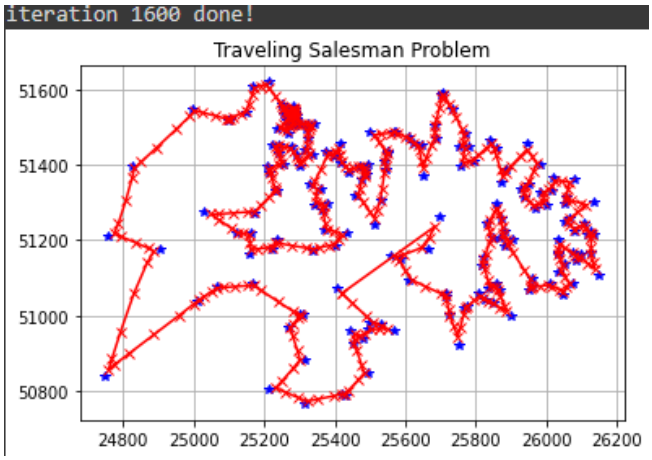


برای رسیدن به بهترین نتیجه نیاز به آموزش بیش از این‌ها داریم. با وجود زمانبر بودن این اجرا، برای اطمینان از درستی الگوریتم اجرا کردم و چنین نتایجی را در پی داشت که مطلوب هستند:









```
iteration 2000 done!
```