



دانشگاه صنعتی اصفهان
دانشکده مهندسی برق و کامپیوتر
مهندسی کامپیوتر گرایش فناوری اطلاعات

گزارش کارآموزی

عنوان:

پیاده سازی مجموعه ای از الگوریتم های مورداستفاده در صنعت حمل و نقل هوشمند با استفاده از پردازش تصویر و بینایی کامپیوتر

نگارش:

ملوین مختاری (۹۸۳۱۱۴۳)

استاد کارآموزی:

دکتر نادر کریمی

سرپرست کارآموزی:

دکتر علی مهمان دوست

محل کارآموزی:

شرکت فنی مهندسی پارتاک سیستم ویرا

تابستان ۱۴۰۱

چکیده:

شرکت فنی مهندسی پارتاک سیستم ویرا با هدف ارائه‌ی راهکارهای جدید بر پایه یادگیری ماشین برای رفع مشکلات روزمره صنایع و سازمان‌ها، در کنار ارتقاء سطح کیفی زندگی شهر و ندان از طریق تصمیم‌ایمنی، بهبود سلامت، صرفه‌جویی در وقت و انرژی و رسیدن به آرامش خاطر بنا شد. این شرکت از ابتدا تلاش کرده تا راهکارهای جدیدی طراحی کرده و با منابعی که در اختیار داشته، کارهای مهم و معنادار انجام دهد. بدین منظور تیم شرکت دانش‌بنیان پارتاک سیستم ویرا متشکل از جمیع از زبده‌ترین متخصصین کشور در تولید سیستم‌های هوش مصنوعی مبتنی بر پردازش تصویر و تجربه تولید انواع محصولات مبتنی بر بینایی کامپیوتر است.

از مهم‌ترین فعالیت‌های انجام‌شده در این دوره کارآموزی می‌توانم به موارد ذیل اشاره کنم:

- آشنایی با توابع کاهش بعد PCA و پیاده‌سازی
- آشنایی با مدل‌های یادگیری ماشین SVM و پیاده‌سازی
- آشنایی با شبکه‌های عصبی و مباحث مربوط به Deep Learning و پیاده‌سازی
- آشنایی با Shape Context Descriptor و پیاده‌سازی

در ادامه این گزارش به تفصیل به هر کدام از موارد فوق پرداخته شده است.

فهرست مطالب

۱	آشنایی با شرکت پارتاک سیستم ویرا	۵
۲	آشنایی با توابع کاهش بعد PCA	۷
۱.۲	تئوری	۸
۲.۲	پیادهسازی عملی	۹
۳	آشنایی با مدل یادگیری SVM	۱۱
۱.۳	تئوری	۱۲
۲.۳	پیادهسازی عملی	۱۳
۴	به دست آوردن پارامترهای بهینه شبکه	۱۴
۱.۴	تئوری	۱۵
۲.۴	پیادهسازی عملی	۱۶
۵	آشنایی با شبکه های عصبی	۲۲
۱.۵	تئوری	۲۳
۲.۵	پیادهسازی عملی	۲۵
۶	تشخیص نوع پلاک	۲۷
۱.۶	آموزش مدل SVM برای تشخیص نوع پلاک	۲۸
۲.۶	آموزش شبکه عصبی برای تشخیص نوع پلاک	۳۰
۷	آشنایی با Shape Context Descriptor	۳۲
۱.۷	تئوری	۳۳
۲.۷	پیادهسازی عملی	۳۳
۸	جمع بندی، نتیجه گیری و پیشنهادها	۳۶

فهرست تصاویر

۹	کد قسمت PCA	۱.۲
۱۰	نمودار واریانس بر حسب تعداد بردار ویژه	۲.۲
۱۲	خروجی مدل با دو نوع داده	۱.۳
۱۳	دقت‌های تعریف شده برای هر مدل	۲.۳
۱۳	برای داده‌های آموزش و تست Precision و Recall	۳.۳
۱۵	داده‌های آموزش	۱.۴
۱۶	مقایسه‌ی نتایج آموزش شبکه با پارامترهای مختلف C	۲.۴
۱۶	داده‌های دیتابست Mnist	۳.۴
۱۷	هیستوگرام داده‌های موجود در دیتابست Mnist	۴.۴
۱۷	اعمال PCA روی داده‌های train و test	۵.۴
۱۸	پیدا کردن پارامتر با بیشترین دقต روی داده validation	۶.۴
۱۸	دقت شبکه‌ی نهایی روی داده test	۷.۴
۱۸	دقت نهایی شبکه با استفاده از کل داده‌های train	۸.۴
۱۹	کشیدن نمودار میله‌ای مربوط به احتمال تشخیص در عکس‌های غلط پیش‌بینی شده	۹.۴
۱۹	نمودار میله‌ای تشخیص برای داده‌های train و test که به اشتباه تشخیص داده شدند	۱۰.۴
۲۰	میانگین زمان لازم برای پیش‌بینی هر عکس	۱۱.۴
۲۰	جدا کردن عکس‌های اشتباه پیش‌بینی شده و تولید فایل txt	۱۲.۴
۲۱	نمونه‌ای از داده‌های اشتباه پیش‌بینی شده	۱۳.۴
۲۱	نمونه‌ای از فایل txt	۱۴.۴
۲۳	یک شبکه عصبی با یک لایه ورودی، سه لایه مخفی و یک لایه خروجی	۱.۵
۲۴	آنواع لایه‌ها در شبکه‌های عصبی	۲.۵
۲۵	معماری شبکه‌ی اول	۳.۵
۲۵	آموزش شبکه‌ی اول	۴.۵
۲۵	نتیجه شبکه‌ی اول	۵.۵
۲۶	معماری شبکه‌ی دوم	۶.۵
۲۶	آموزش شبکه‌ی دوم	۷.۵
۲۶	نتیجه شبکه‌ی دوم	۸.۵
۲۸	آنواع پلاک در ایران	۱.۶
۲۸	بارگزاری عکس پلاک‌های معمولی	۲.۶
۲۹	بارگزاری عکس پلاک‌های منطقه آزاد	۳.۶
۲۹	ساختن عکس‌های رندوم از اطراف خودرو	۴.۶
۲۹	نتیجه‌ی نهایی روی داده‌های train و test و validation	۵.۶
۳۰	معماری شبکه‌ی اول	۶.۶

۳۰	نتیجه شبکه‌ی اول	۷.۶
۳۱	معماری شبکه‌ی دوم	۸.۶
۳۱	نتیجه شبکه‌ی دوم	۹.۶
۳۳	Shape Context	۱.۷
۳۴	به دست آوردن میانگین نقاط بزرگترین کانتور	۲.۷
۳۴	ساخت هیستوگرام log-polar	۳.۷
۳۵	تابع به دست آوردن کانتور ۳۴ تایی	۴.۷
۳۵	تابع به دست آوردن فاصله، زاویه و هیستوگرام	۵.۷
۳۵	نتایج شبکه‌ی SVM روی داده‌های train و test	۶.۷

فصل ۱

آشنایی با شرکت پارتاک سیستم ویرا

نشانی شرکت: اصفهان، خیابان وحید، بن بست شقایق، پلاک ۳۶۶

شماره تماس: ۰۹۱۳۷۳۹۸۱۵۰ - ۰۳۱۳۶۲۶۱۷۹۵

ویسایت: psvira.com

ایمیل: info@psvira.com

شرکت فنی مهندسی پارتاک سیستم ویرا با هدف ارائه‌ی راهکارهای جدید بر پایه یادگیری ماشین برای رفع مشکلات مردم، صنایع و سازمان‌ها بنا شد. از ابتدا تأسیس این شرکت تلاش شده تا راهکارهای جدیدی طراحی شود و با کمک منابعی که در اختیار بود، کارهای مهم و معنادار انجام شود.

خوبی‌خانه باتکیه بر دانش و علم تیم فعال و پویایی از نخبگان این حوزه که در این مجموعه گرد هم آمدند، راهکارهای متعددی بر پایه یادگیری ماشین به مشتریان ارائه شده و به عنوان یک شرکت دانش‌بنیان سهم خود در تولید و اشتغال را به نحو احسن ادا کرده است.

شرکت پارتاک سیستم ویرا طراح سیستم‌های نظارت تصویری با افزونه‌های هوش مصنوعی است. در این شرکت تلاش می‌شود تا راهکارهایی مختلف بر پایه هوش مصنوعی مطالعه و پیاده‌سازی شوند. محصول اصلی شرکت سامانه نظارت تصویری است که جز محدود نرم‌افزارهای مورداستفاده بومی در این زمینه است و توانسته به عنوان یک سکوی مناسب برای ارائه راهکارهای مبتنی بر یادگیری ماشین به مشتریان خود عمل کند.

رسالت این شرکت به کارگیری یادگیری ماشین برای تحقیق، طراحی و توسعه راهکارهای نوآورانه نظارت تصویر است. به همین منظور گروه‌های متنوعی در این مجموعه مشغول به تحقیق، طراحی و توسعه هستند.

نیاز مشتریان نرم‌افزارهایی با رابطه‌های کار متفاوت است به همین منظور مهندسین گروه نرم‌افزار شرکت، همیشه در حال طراحی و پیاده‌سازی سامانه‌های مختلف نرم‌افزاری بر روی پلتفرم‌های مختلف هستند.

شاید به نظر نیاید؛ اما نمایش برخط و دائم تصاویر و ارائه دسترسی‌های مختلف به ویدئوها احتیاج به تحقیق و پیاده‌سازی‌های بسیار پیچیده‌ای دارد. افرادی در تیم نرم‌افزار شرکت فعالیت می‌کنند که آشنایی بالایی با انکودينگ‌های مختلف تصویر دارند. درنهایت برای ارائه راهکار، همواره در حال توسعه اپلیکیشن‌های مختلف، از وب و اندروید گرفته تا پیاده‌سازی بر روی سخت‌افزارهای تخصصی، هستند.

مزیت اصلی شرکت نسبت به رقبا استفاده از یادگیری ماشین است و اعضای مجموعه در این زمینه همواره در حال تحقیق و توسعه هستند، بنابراین اعضای گروه هوش مصنوعی همواره در حال مطالعه‌ی راهکارهای جدید، جمع‌آوری داده و آموزش و ارزیابی راهکارهای مختلف هستند.

در برخی پروژه‌ها نیاز است دوربین خاصی طراحی شود و یا حتی دوربین به یک ربات وصل شده تا همواره بر روی یک شیء تسلط داشته باشد و یا نیاز است که سامانه بر روی بستر خاصی پیاده‌سازی و راهاندازی شود. بدین منظور شرکت همواره تلاش کرده تا در رفع نیازهای سخت‌افزاری گروه، سخاوتمندانه عمل کند.

یکی از مطالبی که از ابتدای توسعه نرم‌افزارها در شرکت به آن اهمیت داده می‌شود، امنیت آن‌ها است. تیم امنیت شرکت، همواره دوش به دوش تیم توسعه در حال ارزیابی و بررسی راهکارها ارائه شده و نرم‌افزارهای تهیه شده است. در این زمینه، شرکت علاوه بر وجود آزمایشگاه مستقر دائمی، با آزمایشگاه‌های افتتا تفاهم‌نامه و همکاری داشته است.

درنهایت اکثر پروژه‌های نظارت تصویری بر روی شبکه اجرا خواهد شد، به همین منظور شرکت همیشه آمادگی دارد تا با کمترین تغییرات و مطابق با نیاز مشتریان شبکه‌های آن‌ها را با راهکارهای ارائه شده تطبیق دهد. تیم شبکه شرکت نیز با ارائه راهکارهای پسیو و اکتیو شبکه در زمان استقرار سامانه‌ها، سایر تیم‌ها را همراهی می‌کند.

فصل ۲

آشنایی با توابع کاهش بعد PCA

۱۰.۲ تئوری

PCA یک روش کاهش بعد است که اغلب برای کاهش بعد داده‌های بزرگ استفاده می‌شود. این تبدیل مجموعه‌های بزرگ را به مجموعه‌های کوچک‌تر به نحوی تبدیل می‌کند که اکثر اطلاعات مجموعه‌ی اصلی باقی بماند.

برای انجام PCA به روش زیر عمل می‌کنیم:

۱) نرمال کردن داده:

هدف از این مرحله استانداردسازی محدوده متغیرهای اولیه است تا هر یک از آن‌ها به یک اندازه در تجزیه و تحلیل مشارکت داشته باشد.

$$\text{Normalized Data} = \frac{\text{Data} - \text{Mean}}{\text{Standard Deviation}} \quad (1.2)$$

۲) به دست آوردن ماتریس کوواریانس:

هدف از این مرحله درک و بررسی رابطه بین متغیرهای مجموعه داده‌های ورودی است. زیرا گاهی اوقات متغیرها به شدت باهم ارتباط دارند به گونه‌ای که حاوی اطلاعات اضافی هستند. بنابراین، برای شناسایی این همبستگی‌ها، ماتریس کوواریانس را محاسبه می‌کنیم.

ماتریس کوواریانس یک ماتریس متقاضن $p \times p$ است که تعداد بعد داده‌های ورودی است.

۳) به دست آوردن مقدار ویژه و بردار ویژه‌های ماتریس کوواریانس:

بردارهای ویژه و مقادیر ویژه مفاهیم جبری خطی هستند که برای تعیین اجزای اصلی داده‌ها باید از ماتریس کوواریانس محاسبه کنیم. اجزای اصلی متغیرهای جدیدی هستند که به صورت ترکیب خطی یا مخلوطی از متغیرهای اولیه ساخته می‌شوند. این ترکیبات به گونه‌ای انجام می‌شوند که متغیرهای جدید ارتباطی ندارند و بیشتر اطلاعات درون متغیرهای اولیه فشرده می‌شوند تا به اولین اجزاء تبدیل شوند. اجزای اصلی جهت‌های داده را نشان می‌دهند که حداقل مقدار واریانس را دارند، یعنی خطوطی که بیشتر اطلاعات داده‌ها را دارند. هرچه واریانس یک خط بزرگ‌تر باشد، پراکندگی نقاط داده در امتداد آن بزرگ‌تر است پس اطلاعات بیشتری نیز دارد.

بردارهای ویژه ماتریس کوواریانس درواقع جهت محورهایی هستند که بیشترین واریانس (بیشترین اطلاعات) در آن‌ها وجود دارد و ما آن‌ها را اجزای اصلی (Principal Components) می‌نامیم و مقادیر ویژه مقدار واریانس موجود در هر جزء اصلی را نشان می‌دهند. با رتبه‌بندی بردارهای ویژه خود به ترتیب ارزش ویژه خود، از بالاترین تا کمترین، اجزای اصلی را به ترتیب اهمیت به دست می‌آوریم.

۴) بردار ویژگی:

با محاسبه بردارهای ویژه و ترتیب آن‌ها بر اساس مقدار ویژه خود به ترتیب نزولی، به ما اجازه می‌دهد تا اجزای اصلی را به ترتیب اهمیت پیدا کنیم. در این مرحله، آنچه ما انجام می‌دهیم این است که انتخاب کنیم آیا همه این اجزا را حفظ کنیم یا آن‌هایی که اهمیت کمتری دارند را کنار بگذاریم و با مابقی اجزای اصلی، ماتریسی از بردارها را که بردار ویژگی (Feature Vector) می‌نامیم، تشکیل دهیم.

۵) تصویر کردن داده‌های اولیه در جهت بردار ویژگی:

هدف این است که داده را در جهت بردار ویژگی به دست آمده در قسمت قبل تصویر کنیم. این را می‌توان با ضرب ماتریسی به شکل زیر انجام داد:

$$\text{Final Data} = \text{Feature Vector}^T * \text{Normalized Data}^T \quad (2.2)$$

۲.۲ پیاده‌سازی عملی

برای پیاده‌سازی الگوریتم PCA از دیتاست ارقام Sklearn استفاده کردم. این دیتاست شامل ارقام در سایز 8^*8 است. هدف کد زیر پیدا کردن تعداد بردارهای ویژگی لازم برای حفظ حداقل ۹۹ درصد واریانس است.

```

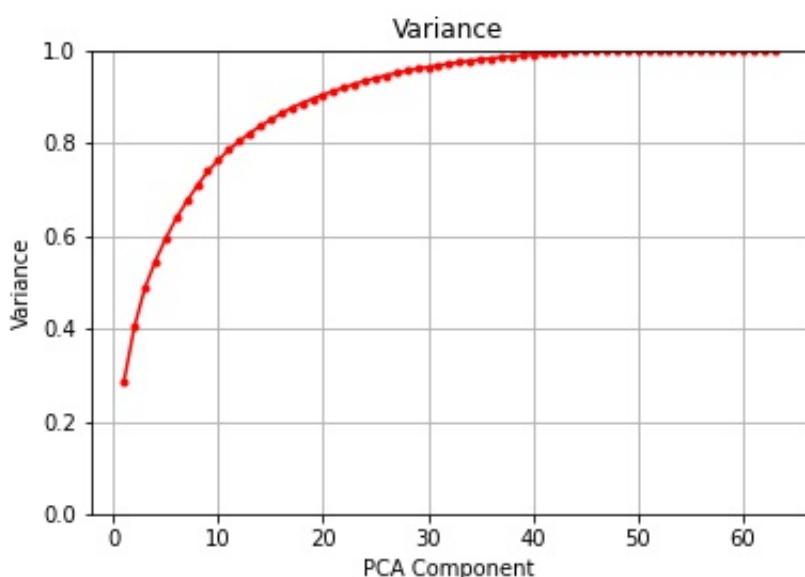
from sklearn import datasets
import matplotlib.pyplot as plt
import cv2
import numpy as np
from sklearn.decomposition import PCA
def plot_func(y,plt_file,plt_title,dim,jump,start):
    x = range(start,dim+1,jump)
    plt.plot(x, y, color='red', marker='o', markerfacecolor='red', markersize=3)
    plt.xlabel('PCA Component')
    plt.ylabel(f'{plt_title}')
    plt.title(f'{plt_title}')
    plt.ylim(0, 1)
    plt.grid()
    plt.savefig(plt_file)
    plt.close()
    return
# loading data set
digits = datasets.load_digits()
# reshaping data
n_samples = len(digits.images)
image = digits.images
data = digits.images.reshape((n_samples, -1))
# fitting pca to the data
dim = data.shape[1]
variance = np.zeros((dim-1))
pca = PCA(n_components=dim)
pca.fit(data)
var = pca.explained_variance_ratio_
# finding the number of component to get 0.99 of variance
variance[0] = var[0] + var[1]
for i in range(1,dim-1):
    variance[i] = variance[i-1] + var[i+1]
n = np.where(variance>0.99)[0][0]
print(f'The image size was {image[0].shape}')
print(f'For having at least 0.99 variance, We need n_component of PCA to be {n}.')

```

☞ The image size was (8, 8)
For having at least 0.99 variance, We need n_component of PCA to be 39.

شکل ۱.۲: کد قسمت PCA

همان طور که مشخص است برای هر عکس 8^*8 که ۶۴ المان دارد توانستیم با یک بردار ۳۹ تابی به ۹۹ درصد واریانس داده اصلی برسیم. نمودار واریانس به ازای تعداد بردار ویژه را در تصویر صفحه بعد میتوانید مشاهده کنید.



شکل ۲.۲: نمودار واریانس بر حسب تعداد بردار ویژه

فصل ۳

آشنایی با مدل یادگیری SVM

۱۰.۳ تئوری

Support Vector Machines (SVM) مجموعه‌ای از روش‌های یادگیری supervised برای طبقه‌بندی و رگرسیون استفاده می‌شوند. در این روش‌ها تعدادی داده به همراه label خود را به شبکه می‌دهیم و شبکه سعی می‌کند یکتابع هزینه‌ای را مینیمم کند. این تابع هزینه شامل پارامترها مختلف مثل دقت شبکه، اندازه‌ی وزن‌های مورداستفاده در شبکه و میزان اهمیت margin است.

Objective Cost :

$$\min (||W||^2 + C \sum_i \zeta_i) \quad (1.3)$$

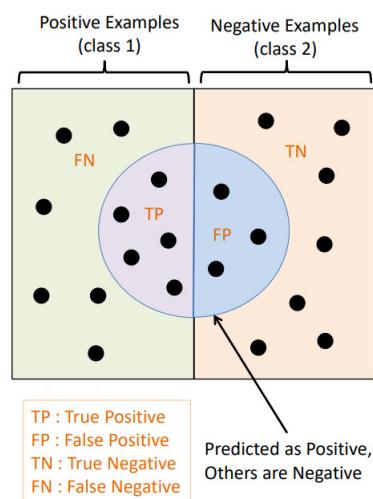
Subject to :

$$y_i(W^T x_i + b) \geq 1 - \zeta_i \quad (2.3)$$

مزایای SVM عبارتند از:

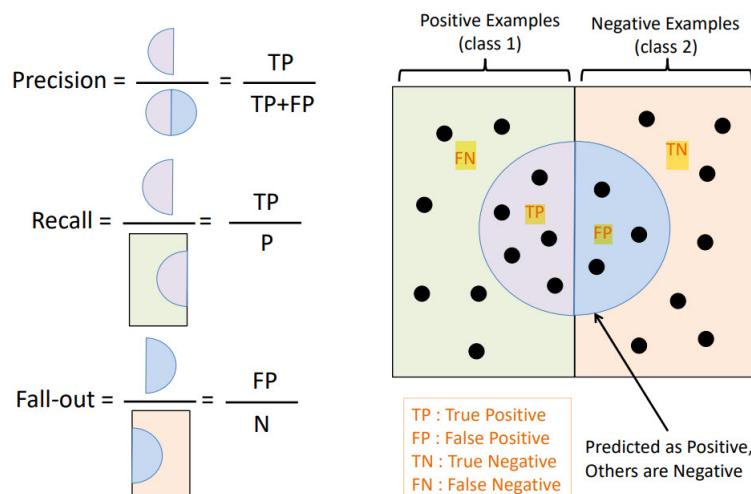
- موثر در ابعاد بالا
- موثر در مسائلی که تعداد بعد از تعداد نمونه‌ها بیشتر است.
- استفاده بهنیه از حافظه

بعد از آموزش شبکه به وسیله داده‌های آموزش، مدل خود را می‌آزماییم. برای این کار از داده‌های تست استفاده می‌کنیم. برای هر داده مدل ما یک خروجی می‌دهد که نشان‌دهنده‌ی label پیش‌بینی شده توسط مدل است. فرض کنید دو نوع داده داریم.



شکل ۱.۳: خروجی مدل با دو نوع داده

دقت مدل ما بسته به موردی که قرار است استفاده شود می‌تواند تعریف شود؛ مثلاً در موارد قضایی، دقتش به این معناست که اگر کسی بی‌گناه باشد خروجی مدل حتماً منفی باشد. بر عکس در برخی کاربردهای پزشکی دقتش به این معناست که موارد بیمار خروجی حتماً مثبت باشد. بنابراین موارد زیر را برای هر مدل به دست می‌آوریم و بر اساس کاربرد موردنظرمان دقتش را به دست می‌آوریم.



شکل ۲.۳: دقت‌های تعریف شده برای هر مدل

۲.۳ پیاده‌سازی عملی

برای پیاده‌سازی SVM از کتابخانه‌ی Sklearn استفاده کردم. داده‌ها را از خروجی PCA گرفتم و برای داده‌های آموزش و تست به دست آوردم.

```
pca = PCA(n_components=n)
pca.fit(data)
pca_data = pca.transform(data)
x_train, x_test, y_train, y_test = train_test_split(pca_data, digits.target, test_size=0.3, shuffle=True)

clf = svm.SVC(kernel = 'rbf', gamma=0.001, C = 10)
clf.fit(x_train, y_train)
Predicted_Train = clf.predict(x_train)
Predicted_Test = clf.predict(x_test)
Precision_Score_Train = metrics.precision_score(y_train,Predicted_Train,average='macro')
Precision_Score_Test = metrics.precision_score(y_test,Predicted_Test,average='macro')
Recall_Score_Train = metrics.recall_score(y_train,Predicted_Train,average='macro')
Recall_Score_Test = metrics.recall_score(y_test,Predicted_Test,average='macro')
print('Precision: ')
print(f'Precision score for train data is {Precision_Score_Train}.')
print(f'Precision score for test data is {Precision_Score_Test}.')
print('recall: ')
print(f'recall score for train data is {Recall_Score_Train}.')
print(f'recall score for test data is {Recall_Score_Test}.')
```

For having at least 0.99 variance, We need n_component of PCA to be 39.
 Precision:
 Precision score for train data is 1.0.
 Precision score for test data is 0.9901818181818183.
 recall:
 recall score for train data is 1.0.
 recall score for test data is 0.9906847011453417.

شکل ۳.۳: Precision و Recall برای داده‌های آموزش و تست

فصل ۴

به دست آوردن پارامترهای بهینه شبکه

۱.۴ تئوری

الگوریتم‌های SVM از مجموعه‌ای از توابع ریاضی استفاده می‌کنند که به kernel تعریف می‌شوند. وظیفه kernel این است که داده‌ها را به عنوان ورودی گرفته و به شکل موردنیاز تبدیل کند. الگوریتم‌های مختلف SVM از انواع مختلف توابع kernel استفاده می‌کنند. این توابع می‌توانند انواع مختلفی داشته باشند. به عنوان مثال:

Linear •

Nonlinear •

Polynomial •

Sigmoid •

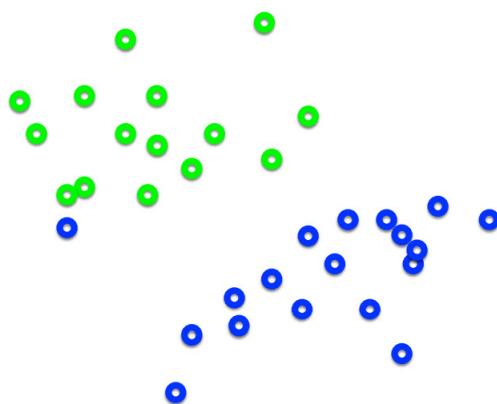
Radial Basis Function •

پرکاربردترین نوع عملکرد kernel kernel، آخر یا به اختصار RBF است. برای استفاده از این کرنل باید دو پارامتر C و gamma را مقداردهی کنیم. کرنل RBF فرمولی به این شکل دارد:

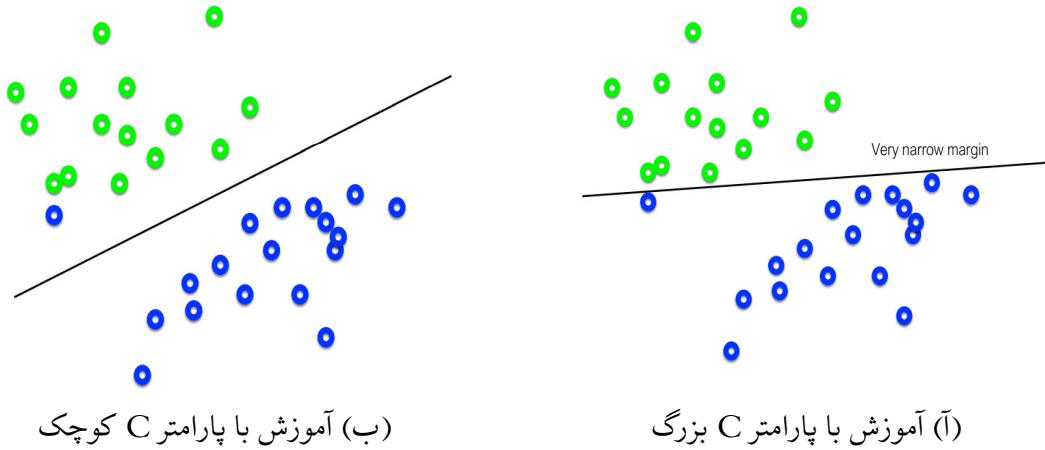
$$k(x_i, x_j) = \exp(\gamma \|x_i - x_j\|^2) \quad (1.4)$$

پارامتر gamma تعیین می‌کند که تأثیر یک نمونه آموزشی تا چه حد می‌رسد. پارامتر gamma را می‌توان معکوس شعاع تأثیرگذاری نمونه‌هایی دانست که توسط مدل به عنوان بردارهای پشتیبانی (Support Vector) انتخاب شده‌اند.

پارامتر C یک بدهبستان (Trade off) بین درصد تشخیص صحیح داده‌های آموزش در مقابل حداکثرسازی حاشیه تابع تصمیم‌گیری برقرار می‌کند. برای مقادیر بزرگتر C، تابع تصمیم‌گیری در نقاط آموزشی بهتر عمل می‌کند و حاشیه کوچک‌تر می‌شود. برای مقادیر کوچک‌تر C، دقت آموزش کمتر می‌شود ولی حاشیه بزرگ‌تر است؛ بنابراین تابع تصمیم‌گیری ساده‌تر را تشویق می‌کند.



شکل ۱.۴: داده‌های آموزش



همان‌طور که مشخص است وقتی پارامتر C زیاد شده شبکه همه داده‌های آموزش را به درستی تشخیص می‌دهد. هدف ما به دست آوردن مدلی است که روی داده‌هایی که مشاهده نشده هم درست جواب بدهد و به اصلاح generalization بالایی داشته باشد. از آنجایی که خطا و نویز جز غیرقابل جداسازی هر آموزشی است به نظر می‌رسد مدل سمت راست مدل خوبی نیست چون نتوانسته داده‌های آبی و سبز را به خوبی از هم جدا کند. اگر مقدار C را کمتر کنیم باعث می‌شود در مینیمم کردن تابع هزینه به سمتی حرکت کنیم که خط جداکننده می‌تواند ممکن فاصله‌ی زیادی از داده‌ها داشته باشد هرچند این کار باعث می‌شود چند داده به اشتباہ پیش‌بینی شوند.

۲.۴ پیاده‌سازی عملی

برای پیاده‌سازی SVM از کتابخانه libsvm استفاده کردم. از دیتاست ارقام Mnist استفاده کردم. سایز عکس‌های این دیتاست 28×28 است. روی این دیتاست PCA اعمال کردم. داده‌های آموزش را به دو قسمت validation و train تقسیم کردم. با قسمت train شبکه را با پارامترهای C و γ مختلف آموزش دادم و دقت شبکه را روی قسمت validation به دست آوردم. پارامترهایی با بیشترین دقت را پیدا کردم. سپس شبکه‌ای را با این پارامترها آموزش دادم و روی داده‌ی test نتیجه نهایی را به دست آوردم.

```

(TrainX, TrainY), (TestX, TestY) = mnist.load_data()

train_histogram = Create_Histogram(TrainY)
plot_func(range(0,10),train_histogram,'mnist_train_histogram.jpg','train_histogram')
test_histogram = Create_Histogram(TestY)
plot_func(range(0,10),test_histogram,'mnist_test_histogram.jpg','test_histogram')
train_samples = len(TrainX)
trainX = TrainX.reshape((train_samples,-1)).astype(np.float64)
test_samples = len(TestX)
testX = TestX.reshape((test_samples,-1)).astype(np.float64)
dim = (trainX.shape)[1]

```

convert to zero mean and one variance

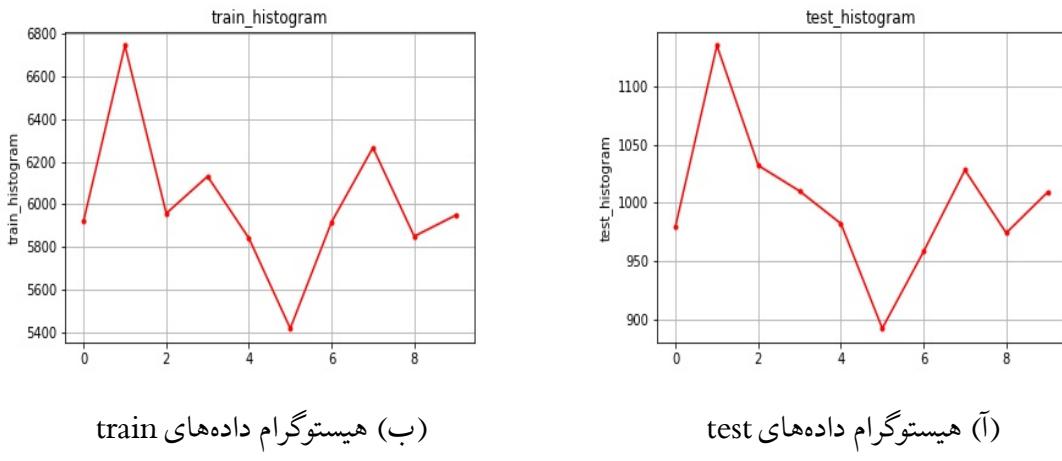
```

[5] for i in range(train_samples):
    trainX[i] = (trainX[i]-np.mean(trainX[i]))/np.std(trainX[i])
for i in range(test_samples):
    testX[i] = (testX[i]-np.mean(testX[i]))/np.std(testX[i])

```

شکل ۳.۴: داده‌های دیتاست Mnist

هر عکس 28^*28 را به یک بردار 784×1 تبدیل کرد. سپس هر بردار را نرمال کردم یعنی میانگین بردار صفر و انحراف از معیار آن یک شد.



شکل ۴.۴: هیستوگرام داده‌های موجود در دیتاست Mnist

```

Find n_component of PCA for variance = 0.99

[?] variance = np.zeros((dim-1))
pca = PCA(n_components=dim)
pca.fit(trainX)
var = pca.explained_variance_ratio_
variance[0] = var[0] + var[1]
for i in range(1,dim-1):
    variance[i] = variance[i-1] + var[i+1]

plot_func(range(2,dim+1),variance,'variance.jpg','variance')
n = np.where(variance>0.99)[0][0]
print(f'For having at least 0.99 variance, We need n_component of PCA to be {n}.')
print(f'For having at least 0.99 variance, We need n_component of PCA to be 331.')

Set the n_component

[?] print("If you want to set a new n_component, press n.")
print("Otherwise, press any key.")
print("Key: ")
word = input()
if word == 'n' or word == 'N':
    print("Enter the new n_component.")
    print("n_component: ")
    n = int(input())

Apply the PCA

[8] pca = PCA(n_components=n)
pca.fit(trainX)
pca_trainX = pca.transform(trainX)
pca_testX = pca.transform(testX)

```

شکل ۵.۴: اعمال PCA روی داده‌های train و test

همان‌طور که مشخص است برای رسیدن به ۹۹ درصد واریانس به ۳۳۱ بعد از ۷۸۴ بعد نیاز داریم. این PCA را روی داده‌ی train فیت می‌کنیم و روی داده‌ی test و train اعمال می‌کنیم.

```
Create samples for tuning the SVM
[1] train_X,train_Y = Create_data_label(pca_trainX,TrainY,n,0,1000)
validation_X,validation_Y = Create_data_label(pca_trainX,TrainY,n,1000,1500)
test_X,test_Y = Create_data_label(pca_testX,TestY,n,0,500)

sample_train_histogram = Create_Histogram(train_Y)
plot_func(range(0,10),sample_train_histogram,'train_histogram.jpg','train_histogram')
sample_validation_histogram = Create_Histogram(validation_Y)
plot_func(range(0,10),sample_validation_histogram,'validation_histogram.jpg','validation_histogram')
sample_test_histogram = Create_Histogram(test_Y)
plot_func(range(0,10),sample_test_histogram,'test_histogram.jpg','test_histogram')

Tuning parameter
[2] prob = svm_problem(train_Y, train_X)
Best_C = 0
Best_gamma = 0
Best_p = 0
for C in range(1,10):
    for gamma in range(1,10):
        param = svm_parameter(f'-s 0 -t 2 -g {gamma/1000} -c {C}')
        m = svm_train(prob, param)
        print(f'gamma is {gamma/1000} and C is {C}')
        p_label, p_acc, p_val = svm_predict(validation_Y, validation_X, m)
        p = p_acc[0]
        if p > Best_p:
            Best_p = p
            Best_C = C
            Best_gamma = gamma/1000
print('Highest precision is (Best_p).')
print('Best C is {Best_C}.')
print('Best gamma is {Best_gamma}.')
```

شکل ۴.۶: پیدا کردن پارامتر با بیشترین دقت روی داده‌ی validation

بیشترین دقت به ازای $C = 2$ و $gamma = 0.001$ به دست آمد. حال با این دو پارامتر دقت را روی داده‌های test به دست می‌آورم.

```
Final SVM
[3] prob = svm_problem(train_Y, train_X)
param = svm_parameter(f'-s 0 -t 2 -g {0} -c {2} -b 1')
m = svm_train(prob, param)
p_label, p_acc, p_val = svm_predict(validation_Y, validation_X, m, options='-b 1')
p_label, p_acc, p_val = svm_predict(test_Y, test_X, m, options='-b 1')

Accuracy = 96.46% (4823/5000) (classification)
Accuracy = 95.06% (4753/5000) (classification)
```

شکل ۷.۴: دقت شبکه‌ی نهایی روی داده‌ی test

همان‌طور که مشخص است به دقت خوبی روی داده‌هایی که شبکه تابه‌حال ندیده بود رسیدیم. حال که از درستی ساخت شبکه مطمئن شدیم برای ساخت شبکه از همه‌ی داده‌های train استفاده می‌کنیم.

```
[13] train_X,train_Y = Create_data_label(pca_trainX,TrainY,n,0,int(np.min(train_histogram)))
test_X,test_Y = pca_testX,TestY
prob = svm_problem(train_Y, train_X)
param = svm_parameter(f'-s 0 -t 2 -g {0} -c {2} -b 1')
m = svm_train(prob, param)

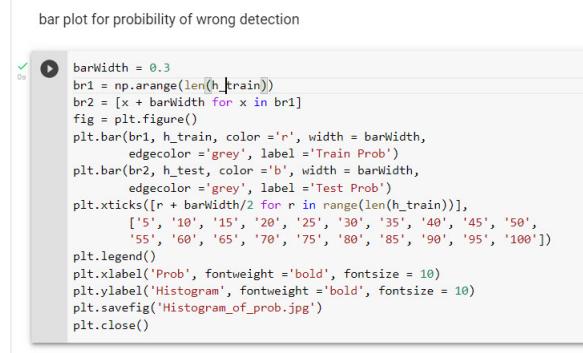
Find Accuracy
[15] p_label, p_acc, p_val = svm_predict(train_Y, train_X, m, options='-b 1')
h_train = Create_Histogram_Probability(p_val,p_label,train_Y)
Accuracy = 99.3765% (53872/54210) (classification)

[16] p_label, p_acc, p_val = svm_predict(test_Y, test_X, m, options='-b 1')
h_test = Create_Histogram_Probability(p_val,p_label,test_Y)
Accuracy = 98.34% (9834/10000) (classification)
```

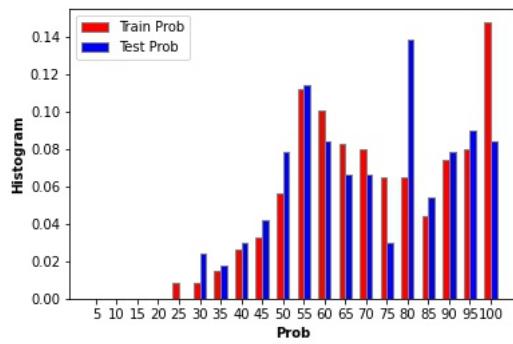
شکل ۸.۴: دقت نهایی شبکه با استفاده از کل داده‌های train

و درنهایت به دقت ۹۸ درصد رسیدیم. برای داده‌هایی که توسط شبکه به اشتباه تشخیص داده می‌شدند احتمال این تشخیص را پیدا کردیم و در یک جدول میله‌ای این احتمالات را به صورت نرمال شده

نشان دادیم. همچنین این عکس‌ها را جدا کردیم. در بسیاری از موارد تشخیص عدد برای چشم انسان هم سخت بود.



شکل ۹.۴: کشیدن نمودار میله‌ای مربوط به احتمال تشخیص در عکس‌های غلط پیش‌بینی شده



شکل ۱۰.۴: نمودار میله‌ای تشخیص برای داده‌های train و test که به اشتباه تشخیص داده شدند

برای پیدا کردن میانگین زمان برای مشخص کردن label هر عکس روی کل داده‌ها به تعداد ۱۰ بار تشخیص را انجام دادیم و به زمان ۰.۰۹۸۲۲۳۰۶۵۱۳۷۸۶۳۱۷ ثانیه برای هر تشخیص رسیدیم.

Find average time for 10 itation to predict test images

```

import time
sum = 0
iter = 10
for i in range(iter):
    start = time.time()
    p_label, p_acc, p_val = svm_predict(test_Y, test_X, m, options="-b 1")
    end = time.time()
    print(end-start)
    sum += (end-start)

print(f'The average time for predict label of test sample is {sum/iter/test_Y.size}.')

```

Accuracy = 98.34% (9834/10000) (classification)
98.38333296775818
Accuracy = 98.34% (9834/10000) (classification)
98.4785053730011
Accuracy = 98.34% (9834/10000) (classification)
98.4918041229248
Accuracy = 98.34% (9834/10000) (classification)
98.08489418029785
Accuracy = 98.34% (9834/10000) (classification)
97.92722201347351
Accuracy = 98.34% (9834/10000) (classification)
98.051198720932
Accuracy = 98.34% (9834/10000) (classification)
98.18302750587463
Accuracy = 98.34% (9834/10000) (classification)
97.99168539047241
Accuracy = 98.34% (9834/10000) (classification)
98.09775590896606
Accuracy = 98.34% (9834/10000) (classification)
98.54122519493103
The average time for predict label of test sample is 0.009822306513786317.

شکل ۱۱.۴ : میانگین زمان لازم برای پیش‌بینی هر عکس

سپس همهی عکس‌هایی که با اشتباه پیش‌بینی شده بودند را ذخیره کردم و یک فایل txt شامل label درست و سه label پیش‌بینی شده به ترتیب احتمالاتی درست کردم.

```

[19] try:
    !rm -rf img
    !mkdir img
except:
    !mkdir img

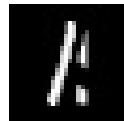
False_Pred_Index = np.where(p_label!=test_Y)[0]
f = open("Info.txt","w+")
f.write('{:15} '.format("Image name"))
f.write('{:15} '.format("Image label"))
f.write('{:15} '.format("First Guess"))
f.write('{:15} '.format("First Prob"))
f.write('{:15} '.format("Second Guess"))
f.write('{:15} '.format("Second Prob"))
f.write('{:15} '.format("Third Guess"))
f.write('{:15} \n'.format("Third Prob"))
for i in range(np.size(False_Pred_Index)):
    img = TestX[False_Pred_Index[i]]
    p = np.array(p_val[False_Pred_Index[i]])
    p1 = np.max(p)
    x1 = np.where(p==p1)[0][0]
    p[x1] = 0
    p2 = np.max(p)
    x2 = np.where(p==p2)[0][0]
    p[x2] = 0
    p3 = np.max(p)
    x3 = np.where(p==p3)[0][0]
    f.write('{:15} '.format(f'img_{i}'))
    f.write('{:15} '.format(test_Y[False_Pred_Index[i]]))
    f.write('{:15} '.format(x1))
    f.write('{:15} '.format(p1))
    f.write('{:15} '.format(x2))
    f.write('{:15} '.format(p2))
    f.write('{:15} '.format(x3))
    f.write('{:15} \n'.format(p3))
    file = f'img/img_{i}.jpg'
    cv2.imwrite(file,img)
f.close()

```

شکل ۱۲.۴ : جدا کردن عکس‌های اشتباه پیش‌بینی شده و تولید فایل txt



(ب) داده‌ی اشتباه پیش‌بینی شده به عدد ۷



(آ) داده‌ی اشتباه پیش‌بینی شده به عدد ۸

شکل ۱۳.۴: نمونه‌ای از داده‌های اشتباه پیش‌بینی شده

Image name	Image label	First Guess	First Prob	Second Guess	Second Prob	Third Guess	Third Prob
img_0	4	2	0.791448	6	0.159005	4	0.036934
img_1	2	7	0.900786	2	0.089741	8	0.005041
img_2	5	3	0.903545	5	0.087911	1	0.007134
img_3	6	0	0.909226	6	0.081630	5	0.007393
img_4	3	5	0.566680	3	0.433128	2	0.000071
img_5	8	2	0.667169	8	0.161655	0	0.105234
img_6	8	2	0.794132	8	0.098109	1	0.056788
img_7	2	1	0.254184	8	0.252430	7	0.157966
img_8	7	3	0.895223	7	0.059672	2	0.026104
img_9	8	4	0.738619	8	0.174119	9	0.042060
img_10	4	9	0.608003	4	0.386576	7	0.002543
img_11	5	8	0.786798	5	0.188866	3	0.013749
img_12	4	9	0.977834	4	0.019689	7	0.001934
img_13	8	7	0.412495	8	0.396467	3	0.078878
img_14	9	7	0.582327	9	0.360923	4	0.039055
img_15	8	9	0.616406	8	0.379115	4	0.003184
img_16	6	0	0.535885	6	0.274906	5	0.116020
img_17	6	5	0.987918	2	0.004357	0	0.002913
img_18	6	8	0.771657	6	0.178293	2	0.024977
img_19	4	6	0.973922	4	0.022518	0	0.003106
img_20	9	4	0.665813	9	0.295215	7	0.013986
img_21	7	9	0.676426	7	0.263939	8	0.030553
img_22	7	2	0.981024	7	0.008206	3	0.004896
img_23	9	4	0.830743	9	0.112615	8	0.028242
img_24	4	9	0.878939	4	0.109337	7	0.005279
img_25	9	5	0.627365	3	0.241438	0	0.111910
img_26	7	1	0.557202	7	0.360769	2	0.065026
img_27	8	3	0.923430	5	0.036750	0	0.016418
img_28	7	9	0.538845	7	0.249528	8	0.113386
img_29	7	9	0.384327	7	0.305387	0	0.112774
img_30	7	1	0.773222	3	0.084472	8	0.047538
img_31	7	9	0.772865	7	0.184204	4	0.014362
img_32	8	7	0.991890	3	0.004626	8	0.003169
img_33	4	2	0.496930	6	0.355277	4	0.126273
img_34	9	3	0.655168	9	0.126511	8	0.109584
img_35	7	9	0.583758	7	0.166098	2	0.108542
img_36	2	6	0.862634	3	0.054609	5	0.026435
img_37	3	7	0.889124	3	0.096892	9	0.013011
img_38	8	0	0.545502	8	0.394098	5	0.039615
img_39	7	2	0.473614	7	0.280935	3	0.179205
img_40	2	7	0.384722	8	0.306908	9	0.233941

شکل ۱۴.۴: نمونه‌ای از فایل txt

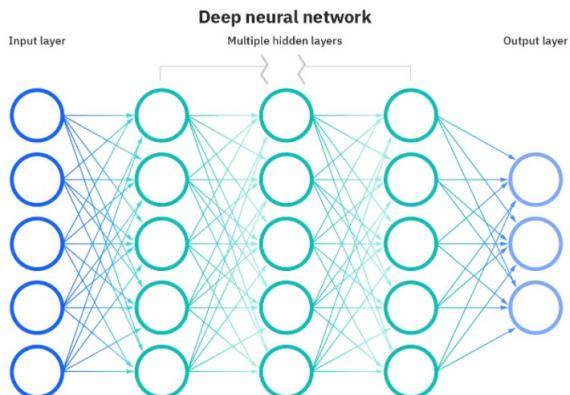
فصل ۵

آشنایی با شبکه‌های عصبی

۱.۵ تئوری

شبکه‌های عصبی، زیرمجموعه‌ای از علوم یادگیری ماشین هستند و در مرکز الگوریتم‌های یادگیری عمیق قرار دارند. نام و ساختار آنها از ساختار مغز انسان الهام گرفته شده است و از راهی که نورون‌های بیولوژیکی به یکدیگر نشان می‌دهند تقلید می‌کند.

شبکه‌های عصبی مصنوعی از لایه‌هایی از گره تشکیل شده‌اند که شامل یک لایه ورودی، یک یا چند لایه مخفی و یک لایه خروجی است. هر گره یا نورون مصنوعی به نورون‌های دیگر متصل می‌شود و وزن و آستانه مخصوص دارد. اگر خروجی هر گره از مقدار آستانه‌اش بیشتر شود، آن گره فعال می‌شود و داده‌ها را به لایه بعدی شبکه ارسال می‌کند. در غیر این صورت، هیچ داده‌ای به لایه‌ی بعدی شبکه منتقل نمی‌شود.



شکل ۱.۵: یک شبکه عصبی با یک لایه‌ی ورودی، سه لایه‌ی مخفی و یک لایه‌ی خروجی

هر گره به عنوان یک مدل رگرسیون خطی، متشکل از داده‌های ورودی، وزن‌ها، بایاس و خروجی است. فرمول چیزی شبیه به این خواهد بود:

$$\text{output} = \begin{cases} 1 & \sum_{i=1}^m w_i \cdot x_i + \text{bias} \geq 0 \\ 0 & \sum_{i=1}^m w_i \cdot x_i + \text{bias} < 0 \end{cases} \quad (1.5)$$

بنا بر قضیه‌ی تقریب جهانی (Universal Approximation Theorem): فرض کنید $f: \mathcal{R}^n \rightarrow \mathcal{R}^m$ یک تابع پیوسته باشد و $K \subseteq \mathcal{R}^n$ یک مجموعه فشرده باشد که دامنه تابع f باشد، در این صورت:

- می‌توان با یک شبکه عصبی با یک لایه پنهان که تعداد نورون‌های آن به اندازه کافی زیاد باشد تقریبی از تابع f به دست آورد.
- می‌توان با یک شبکه عصبی با تعداد نورون‌های مشخص و ثابت در هر لایه که تعداد لایه‌های آن به اندازه کافی زیاد باشد تقریبی از تابع f به دست آورد.

اگر همه‌ی لایه‌ها به صورت ۱.۵ باشند تعداد لایه اثری نخواهد داشت زیر درنهایت می‌توان با یک مدل خطی کل شبکه را مدل‌سازی کنیم. به همین دلیل Activation Function غیرخطی استفاده کنیم. از مشهورترین Activation Function:

Sigmoid •

$$S(x) = \frac{1}{1 + e^{-x}} \quad (2.5)$$

tanh •

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (3.5)$$

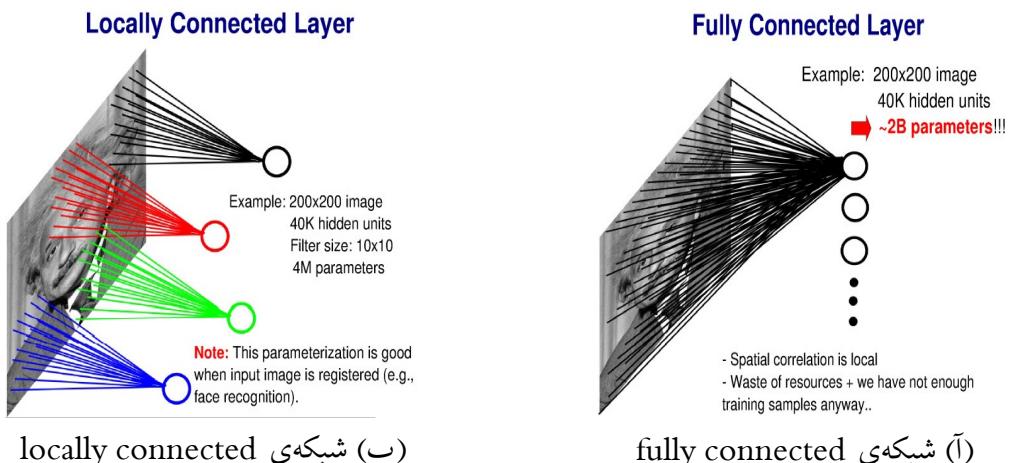
ReLU •

$$ReLU(x) = \max(x, 0) \quad (4.5)$$

Leaky ReLU •

$$\text{Leaky ReLu}(x) = \begin{cases} x & x \geq 0 \\ 0.01x & x < 0 \end{cases} \quad (5.5)$$

گره‌های لایه‌های مخفی و لایه‌ی خروجی به گره‌های لایه‌ی قبلی‌شان می‌توانند وصل بشوند. اگر معماری شبکه به نحوی باشد که هر گره به همه‌ی گره‌های لایه‌ی قبل وصل باشد یک لایه‌ی fully connected خواهیم داشت. اما نوع دیگری از لایه‌ها هم به نام locally connected وجود دارد که در این لایه‌ها هر گره به تعداد مشخصی از گره‌های لایه قبل متصل است.



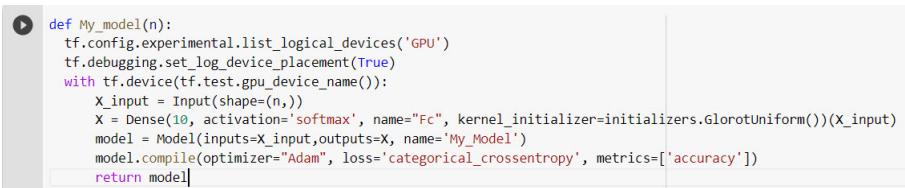
شکل ۲.۵: انواع لایه‌ها در شبکه‌های عصبی

۲.۵ پیاده‌سازی عملی

برای پیاده‌سازی شبکه عصبی از کتابخانه tensorflow استفاده کرد. از دیتاست ارقام Mnist استفاده کرد.

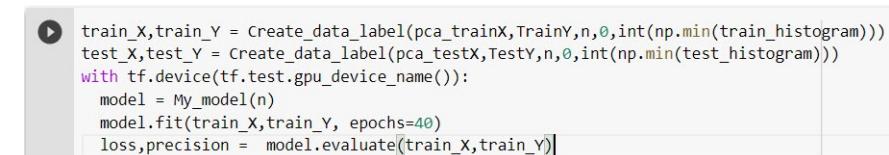
دو نوع شبکه عصبی طراحی کرد.

نوع اول: یک شبکه یک لایه ورودی و یک لایه خروجی به صورت fully connected
 نوع دوم: یک شبکه یک لایه ورودی و یک لایه مخفی به صورت locally connected و یک لایه خروجی به صورت fully connected



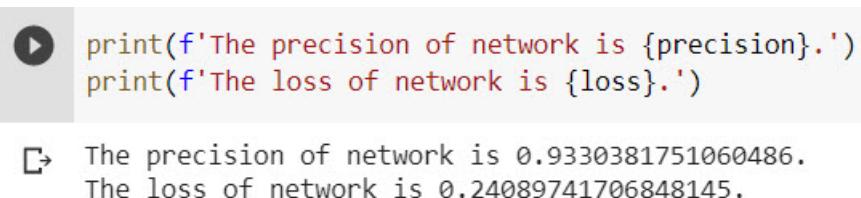
```
def My_model(n):
    tf.config.experimental.list_logical_devices('GPU')
    tf.debugging.set_log_device_placement(True)
    with tf.device(tf.test.gpu_device_name()):
        X_input = Input(shape=(n,))
        X = Dense(10, activation='softmax', name="Fc", kernel_initializer=initializers.GlorotUniform()(X_input))
    model = Model(inputs=X_input,outputs=X, name='My_Model')
    model.compile(optimizer="Adam", loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

شکل ۳.۵: معماری شبکه‌ی اول



```
train_X,train_Y = Create_data_label(pca_trainX,TrainY,n,0,int(np.min(train_histogram)))
test_X,test_Y = Create_data_label(pca_testX,TestY,n,0,int(np.min(test_histogram)))
with tf.device(tf.test.gpu_device_name()):
    model = My_model(n)
    model.fit(train_X,train_Y, epochs=40)
    loss,precision = model.evaluate([train_X,train_Y])
```

شکل ۴.۵: آموزش شبکه‌ی اول



```
print(f'The precision of network is {precision}.')
print(f'The loss of network is {loss}.')
```

⇨ The precision of network is 0.9330381751060486.
 The loss of network is 0.24089741706848145.

شکل ۵.۵: نتیجه شبکه‌ی اول

ورودی شبکه اول خروجی PCA دیتاهای Mnist بود.

```

def My_model(n):
    tf.config.experimental.list_logical_devices('GPU')
    tf.debugging.set_log_device_placement(True)
    with tf.device(tf.test.gpu_device_name()):
        X_input = Input(shape=(n,n,1))
        X = Conv2D(32,(3,3), name="Conv", kernel_initializer=initializers.GlorotUniform())(X_input)
        X = Activation('relu')(X)
        X = MaxPooling2D((2,2), name='Max')(X)
        X = Flatten()(X)
        X = Dense(10, activation='softmax', name="Fc", kernel_initializer=initializers.GlorotUniform())(X)
    model = Model(inputs=X_input,outputs=X, name='My_Model')
    model.compile(optimizer="Adam", loss='categorical_crossentropy', metrics=['accuracy'])
    return model

```

شکل ۶.۵: معماری شبکه دوم

```

n = 28
train_X,train_Y = Create_data_label(TrainX,TrainY,n,0,int(np.min(train_histogram)))
train_X = train_X.reshape(-1,28,28,1)
test_X,test_Y = Create_data_label(TestX,TestY,n,0,int(np.min(test_histogram)))

with tf.device(tf.test.gpu_device_name()):
    model = My_model(n)
    model.fit(train_X,train_Y, epochs=40)
    loss,precision = model.evaluate(train_X,train_Y)

```

شکل ۷.۵: آموزش شبکه دوم

```

print(f'The precision of network is {precision}.')
print(f'The loss of network is {loss}.')

```

☞ The precision of network is 0.9984320402145386.
The loss of network is 0.013051732443273067.

شکل ۸.۵: نتیجه شبکه دوم

ورودی شبکه دوم، عکس‌های دیتاباگ Mnist بود.

فصل ۶

تشخیص نوع پلاک

۱.۶ آموزش مدل SVM برای تشخیص نوع پلاک

این قسمت از پروژه به تشخیص نوع پلاک اختصاص یافت.
در ایران دو نوع پلاک وجود دارد:

- پلاک معمولی
- پلاک منطقه آزاد



(ب) پلاک منطقه آزاد



(آ) پلاک معمولی

شکل ۱.۶: انواع پلاک در ایران

هدف این قسمت تشخیص این دو پلاک از همدیگر است. علاوه بر این دو دسته یک دسته از عکس‌های اطراف ماشین هم به عنوان دسته سوم اضافه می‌کنیم تا اگر دوربین عکسی به جز پلاک گرفت تشخیص بدهیم.

برای ساختن دیتاست از این لینک کمک گرفتم و آن را به صورت دستی به دو قسمت پلاک معمولی و پلاک منطقه آزاد تقسیم بندی کردم.
سپس از این لینک عکس‌هایی از خودرو پیدا کردم. برای دسته سوم به صورت رندوم از این عکس‌ها پنجره انتخاب می‌کرم و به دسته‌های سوم اضافه می‌کرم.

سپس همه‌ی عکس‌ها را به سایز ۳۰*۶۰ برمد. حال از هر دسته پلاک معمولی، پلاک منطقه آزاد و عکس غیر پلاک ۸۰۰ عکس به عنوان داده‌ی آموزش، ۲۰۰ عکس به عنوان داده validation و ۳۰۰ عکس به عنوان داده test قراردادم و بهترین پارامتر را با آموزش شبکه با پارامترهای C و gamma و به دست آوردن دقت روی داده validation به دست آوردم و سپس نتیجه‌ی بهترین شبکه را روی داده‌ی test هم به دست آوردم.

Loading and resizing tall plates to size 30*60

```
# file at https://drive.google.com/drive/folders/  
filenames2 = glob.glob('/content/drive/MyDrive/Intern/plates/tall/*')  
for i in range(len(filenames2)):  
    name2 = filenames2[i]  
    img2 = cv2.imread(name2)  
    resize_im2 = cv2.resize(img2,(60,30),interpolation = cv2.INTER_AREA)  
    new_name2 = f'/content/tall/{i}.jpg'  
    cv2.imwrite(new_name2,resize_im2)
```

شکل ۲.۶: بارگزاری عکس پلاک‌های معمولی

Loading and resizing fat plates to size 30*60

```
# file at https://drive.google.com/drive/folders
filenames1 = glob.glob('/content/drive/MyDrive/Intern/plates/fat/*')
for i in range(len(filenames1)):
    name1 = filenames1[i]
    img1 = cv2.imread(name1)
    resize_im1 = cv2.resize(img1,(60,30),interpolation = cv2.INTER_AREA)
    new_name1 = f'/content/fat/{i}.jpg'
    cv2.imwrite(new_name1,resize_im1)
```

شکل ۳.۶: بارگزاری عکس پلاک‌های منطقه آزاد

Creating random images from car images

```
# file at https://drive.google.com/drive/folders
filename = glob.glob('/content/drive/MyDrive/Intern/car_img/*')
k = 1
for i in range(len(filename)):
    name = filename[i]
    img = cv2.imread(name)
    row,col = img.shape[0],img.shape[1]
    for j in range(4):
        random_row = random.randint(100, row-100)
        random_col = random.randint(200, col-200)
        newimg = img[random_row-100:random_row+100,random_col-100:random_col+100]
        newimg = cv2.resize(newimg,(60,30),interpolation = cv2.INTER_AREA)
        new_name = f'car/{k}.jpg'
        cv2.imwrite(new_name,newimg)
    k = k + 1
```

شکل ۴.۶: ساختن عکس‌های رندوم از اطراف خودرو

Final Result

```
[14] prob = svm_problem(train_Y, train_X)
param = svm_parameter(f'-s 0 -t 2 -g {g} -c {c} -b 1')
m = svm_train(prob, param)

p_label, p_acc, p_val = svm_predict(train_Y, train_X, m, options='-b 1')
p_label, p_acc, p_val = svm_predict(validation_Y, validation_X, m, options='-b 1')
p_label, p_acc, p_val = svm_predict(test_Y, test_X, m, options='-b 1')

Accuracy = 100% (2400/2400) (classification)
Accuracy = 98.5% (591/600) (classification)
Accuracy = 97.5556% (878/900) (classification)
```

شکل ۵.۶: نتیجه‌ی نهایی روی داده‌های train و validation و test

۲.۶ آموزش شبکه‌ی عصبی برای تشخیص نوع پلاک

با دو معماری که در فصل ۵ دیدیم شبکه‌ی عصبی را با سه نوع داده آموزش می‌دهیم و دقت را گزارش می‌کنیم.

```
❶ def My_model(n):
    tf.config.experimental.list_logical_devices('GPU')
    tf.debugging.set_log_device_placement(True)
    with tf.device(tf.test.gpu_device_name()):
        X_input = Input(shape=(n,))
        X = Dense(3, activation='softmax', name="Fc", kernel_initializer=initializers.GlorotUniform())(X_input)
        model = Model(inputs=X_input,outputs=X, name='My_Model')
        model.compile(optimizer="Adam", loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

شکل ۶.۶: معماری شبکه‌ی اول

Fitting and testing the model

```
❷ with tf.device(tf.test.gpu_device_name()):
    model = My_model(dim)
    model.fit(train_X,train_Y, epochs=40)
    loss,precision = model.evaluate(train_X,train_Y)
```

Final result

```
[ ] print(f'The precision of network is {precision}.')
      print(f'The loss of network is {loss}.')
```

The precision of network is 1.0.
The loss of network is 0.023799067363142967.

شکل ۷.۶: نتیجه شبکه‌ی اول

Creating model

```

4s  def My_model(n,m):
      tf.config.experimental.list_logical_devices('GPU')
      tf.debugging.set_log_device_placement(True)
      with tf.device(tf.test.gpu_device_name()):
          X_input = Input(shape=(n,m,1))
          X = Conv2D(32,(3,3), name="Conv", kernel_initializer=initializers.GlorotUniform())(X_input)
          X = Activation('relu')(X)
          X = MaxPooling2D((2,2), name='Max')(X)
          X = Flatten()(X)
          X = Dense(3, activation='softmax', name="Fc", kernel_initializer=initializers.GlorotUniform()(X))
          model = Model(inputs=X_input,outputs=X, name='My_Model')
          model.compile(optimizer="Adam", loss='categorical_crossentropy', metrics=['accuracy'])
      return model

```

شکل ۸.۶: معماری شبکه‌ی دوم

Fitting and testing the model

```

[13] with tf.device(tf.test.gpu_device_name()):
      model = My_model(30,60)
      model.fit(train_X,train_Y, epochs=40)
      loss,precision = model.evaluate(train_X,train_Y)

```

Final result

```

[14] print(f'The precision of network is {precision}.')
      print(f'The loss of network is {loss}.')

```

The precision of network is 1.0.
The loss of network is 2.015161726376391e-06.

شکل ۹.۶: نتیجه شبکه‌ی دوم

فصل ۷

آشنایی با Shape Context Descriptor

۱.۷ تئوری

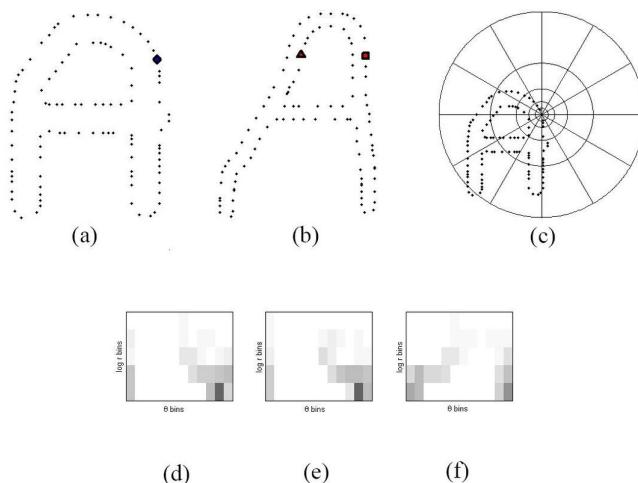
shape context یک روش برای توصیف اشکال است که امکان اندازه‌گیری میزان شباهت اشکال و پیدا کردن تطابق را فراهم می‌کند.

ایده اصلی این است که n نقطه را روی خطوط یک شکل انتخاب کنید. برای هر نقطه مثل p_i روی شکل، $1 - n$ بردار که از اتصال p_i به همه نقاط دیگر به دست می‌آید، در نظر بگیرید. مجموعه همه این بردارها توصیف غنی و دقیق از شکل موضعی در آن نقطه است. ایده اصلی این است که توزیع در موقعیت‌های نسبی توصیف‌کننده‌ای قوی است.

برای هر نقطه مثل p_i فاصله و زاویه‌ی خط واصل این نقطه با نقاط دیگر را به دست می‌آوریم و این عملیات را برای هر نقطه مانند p_i به دست می‌آوریم.

سپس یک هیستوگرام log-polar تشکیل می‌دهیم که به تعداد تقسیم‌بندی‌های طولی ستون و به تعداد تقسیم‌بندی‌های عرضی سطر دارد. به ازای هر دو نقطه‌ی p_i و p_j در صورتی که طول و زاویه خط واصل این دو نقطه در محدوده‌ی متناظر با خانه‌ی $(j|i)$ باشد مقدار آن خانه یکی زیاد می‌شود. در نهایت به یک ماتریس هیستوگرام میرسیم که میتوانیم به عنوان بردار ویژگی برای آموزش شبکه از آن استفاده کنیم.

برای مقاوم بودن در برابر scale طول را بر میانگین طول تقسیم می‌کنیم. معمولاً تعداد تقسیم‌بندی‌های طولی ۵ و زاویه‌ای ۱۲ تا است.



شکل ۱.۷ : Shape Context

۲.۷ پیاده‌سازی عملی

الگوریتم Shape Context Descriptor را به صورت دستی خودم نوشتم. همچنین از دیتابست ارقام Mnist استفاده کردم. اول از همه بزرگ‌ترین کانتور داخل هر عکس را به دست آوردم. سپس تعداد نقاطی که این کانتور دارد را حساب کردم. قصد دارم این نقاط را به عنوان p_i ها انتخاب کنم. از آنجایی که باید تعداد این نقاط در همه عکس‌ها برابر شود یک میانگین گیری بین نقاط بزرگ‌ترین کانتور عکس‌ها انجام دادم و به عدد ۳۴ رسیدم.

برای این کار اول هر عکس را به یک عکس با یزدی تبدیل کردم؛ یعنی خانه‌های کمتر از ۱۲۷ را ۰ کردم و خانه‌های بیشتر مساوی ۱۲۷ را به ۲۵۵ بردم. سپس کانتورهای این عکس با یزدی را به دست

Finding average contour points

```
avg_cont_num = 0
for i in range(len(TrainX)):
    img = TrainX[i]
    ret, thresh = cv2.threshold(img, 127, 255, 0)
    contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    sorted_contours= sorted(contours, key=cv2.contourArea, reverse= True)[0]
    avg_cont_num += len(sorted_contours)
avg_cont_num /= len(TrainX)
avg_cont_num = int(avg_cont_num)
avg_cont_num
```

۳۴

شکل ۲.۷: به دست آوردن میانگین نقاط بزرگ‌ترین کانتور

آوردم. سپس کانتورها را به ترتیب مساحت مرتب کردم و بزرگ‌ترین کانتور را انتخاب کردم و تعداد نقاط این کانتور را در میانگین‌گیری لحاظ کردم.
 پس برای هر عکس باید ۳۴ نقطه را انتخاب می‌کردم. برای شکل‌هایی که بزرگ‌ترین کانتورشان بیشتر از ۳۴ نقطه بود، ۳۴ نقطه اول و آن‌هایی که بزرگ‌ترین کانتورشان کمتر از ۳۴ نقطه بود، با تکرار نقاط به ۳۴ نقطه می‌رساندم.
 پس برای هر عکس ۳۴ نقطه داشتم. بین هر کدام فاصله نرمال شده و زاویه را حساب کردم و با کمک این دو یک هیستوگرام log-polar به دست می‌آورم.

```
def Find_norm(points):
    norm = np.zeros((len(points),avg_cont_num,avg_cont_num))
    for i in range(len(points)):
        for j in range(avg_cont_num):
            norm[i,j] = np.linalg.norm(points[i]-points[j],axis=1)
        norm[i] = norm[i] / norm[i].mean()
    return norm

def Find_angle(points):
    angle = np.zeros((len(points),avg_cont_num,avg_cont_num))
    for i in range(len(points)):
        for j in range(avg_cont_num):
            angle[i,j] = np.arctan2(points[i,:,1]-points[i,j,1],points[i,:,0]-points[i,j,0])/math.pi*180
    return angle

def Create_Histogram(num,norm,angle):
    r_bins = 5
    angle_bins = 12
    r_array = np.array([0, 0.125, 0.25, 0.5, 1, 2])
    angle_array = np.zeros(angle_bins+1)
    for i in range(angle_bins+1):
        angle_array[i] = -180 + i * 30
    Histogram = np.zeros((num,r_bins,angle_bins))
    for k in range(num):
        for i in range(r_bins):
            for j in range(angle_bins):
                Histogram[k,i,j] = len(np.where(((norm[k]>=r_array[i])*1) * ((norm[k]<r_array[i+1])*1) * ((angle[k]>=angle_array[j])*1) * ((angle[k]<angle_array[j+1])*1))[0]
    return Histogram
```

شکل ۳.۷: ساخت هیستوگرام log-polar

```

def Find_contour_points(data):
    contour_points = np.zeros((len(data),avg_cont_num,2))
    for i in range(len(data)):
        img = data[i]
        ret, thresh = cv2.threshold(img, 127, 255, 0)
        contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
        sorted_contours= sorted(contours, key=cv2.contourArea, reverse= True)[0]
        if len(sorted_contours)>=avg_cont_num:
            contour_points[i] = sorted_contours[:avg_cont_num].reshape((avg_cont_num,2))
        else:
            q,r = divmod(avg_cont_num, len(sorted_contours))
            sorted_contours = sorted_contours.reshape((-1,2))
            repeated_contours = sorted_contours.copy()
            for i in range(1,q):
                repeated_contours = cv2.vconcat([repeated_contours, sorted_contours])
            repeated_contours = cv2.vconcat([repeated_contours, sorted_contours[:r]])
            contour_points[i] = repeated_contours
    return contour_points

```

شکل ۴.۷: تابع به دست آوردن کانتور ۳۴ تایی

```

def Find_norm(points):
    norm = np.zeros((len(points),avg_cont_num,avg_cont_num))
    for i in range(len(points)):
        for j in range(avg_cont_num):
            norm[i,j] = np.linalg.norm(points[i]-points[i,j],axis=1)
        norm[i] = norm[i] / norm[i].mean()
    return norm

def Find_angle(points):
    angle = np.zeros((len(points),avg_cont_num,avg_cont_num))
    for i in range(len(points)):
        for j in range(avg_cont_num):
            angle[i,j] = np.arctan2(points[i,:,1]-points[i,j,1],points[i,:,0]-points[i,j,0])/math.pi*180
    return angle

def Create_Histogram(num,norm,angle):
    r_bins = 5
    angle_bins = 12
    r_array = np.array([0, 0.125, 0.25, 0.5, 1, 2])
    angle_array = np.zeros(angle_bins+1)
    for i in range(angle_bins+1):
        angle_array[i] = -180 + i * 30
    Histogram = np.zeros((num,r_bins,angle_bins))
    for k in range(num):
        for i in range(r_bins):
            for j in range(angle_bins):
                Histogram[k,i,j] = len(np.where(((norm[k]>=r_array[i])*1) * ((norm[k]<r_array[i+1])*1) * ((angle[k]>=angle_array[j])*1) * ((angle[k]<angle_array[j+1])*1))[0])
    return Histogram

```

شکل ۵.۷: تابع به دست آوردن فاصله، زاویه و هیستوگرام

سپس هر هیستوگرام به یک بردار تبدیل می‌شود که این بردار برای آموزش مدل SVM استفاده می‌شود. سپس بهترین پارامترها را برای به دست آوردن بیشترین دقیقت به دست می‌آوریم و مدل نهایی را روی داده‌ی test امتحان می‌کنیم.

```

Finalize the model with best parameters

[15]: train_X,train_Y = Create_data_label(Histogram_TrainX_reshaped,TrainY,n,0,int(np.min(train_histogram)))
test_X,test_Y = Histogram_TestX_reshaped,TestY
prob = svm_problem(train_Y, train_X)
param = svm_parameter('f'-s 0 -t 2 -g {g} -c {c} -b 1')
m = svm_train(prob, param)

Precision on train data

[18]: p_label, p_acc, p_val = svm_predict(train_Y, train_X, m, options='-b 1')
Accuracy = 92.1989% (49981/54210) (classification)

Precision on test data

[19]: p_label, p_acc, p_val = svm_predict(test_Y, test_X, m, options='-b 1')
Accuracy = 66.68% (6668/10000) (classification)

```

شکل ۶.۷: نتایج شبکه‌ی SVM روی داده‌های train و test

فصل ۸

جمع‌بندی، نتیجه‌گیری و پیشنهادها

جمع‌بندی:

در این گزارش توضیحات در مورد شرکت فنی مهندسی پارتاک سیستم ویرا، حیطه کاری اصلی شرکت، توضیحات تئوری و پیاده‌سازی‌های عملی PCA ، learning Deep SVM ، Context Descriptor پرداخته شد و کدهای مربوط به بخش‌های پیاده‌سازی توضیح داده شد.

نتیجه‌گیری:

دوره کارآموزی به جهت بهبود دانش و مهارت خود و به کارگیری مفاهیم و راهبردهای تدریس شده در دانشگاه در یک محیط واقعی بسیار مفید است. همچنین به‌واسطه این دوره، تجربیات کاری گران‌بهایی برای متمایز بودن از سایرین در صنعت کسب کردہ‌ایم که بسیار ارزشمند است. به علاوه کارآموزی فرصت مناسبی برای ایجاد شبکه‌ای متشکل از همکاران در حوزه صنعتی رشته تحصیلی و کاری مورد علاقه‌مان است. می‌توان گفت که در فضای رقابتی بازار کار امروزی، دوره کارآموزی به عنوان نقطه قوتی در رزومه کاری افراد و همچنین امتیاز مثبتی برای آنان محسوب می‌شود. درواقع افراد با گذراندن دوره کارآموزی، می‌توانند ویژگی‌های کاری منحصر به‌فرد خودشان را بهبود بیخشدند و به راحتی در بازار کار موردنظرشان از سایر همتایان خود متمایز شوند.

کارآموزی به عقیده من جزو بهترین تجربه‌هایی است که هر دانشجو تجربه می‌کند. آشنا شدن با فضای کار و صنعت تجربه ارزشمندی برای من بود. تا جایی که ممکن بود من سعی کردم خودم را مشتاق مطالب جدید نشان دهم و سرپرست کارآموزی هم تا جایی که توانست به من کمک کرد و ارتباط دوستانه‌ای بین دو طرف بود. در این کارآموزی من با دانش‌های جدید و قادرمندی آشنا شدم و علاوه بر بعد تئوری توانستم در بعد عملی هم شبیه‌سازی انجام دهم که لذت کار را دوچندان می‌کرد.

پیشنهادها:

به عنوان عضوی موقت در این شرکت برخی نکات را دیده‌ام که اکنون بر خود لازم می‌دانم حداقل از بیانش صرف نظر نکنم. در درجه اول به نظر من، اگر کارآموز با بخش‌های تجاری شرکت و نحوه فروش و تعامل با مشتریان هم وارد شود و اطلاعات کسب کند بسیار خوب است. همچنین احساس می‌کنم اگر میزان کار جمیعی بین کارآموزان نیز بیشتر شود، بسیار بازده کار بهتر خواهد بود.