به نام خدا

درس: مقدمه‌ای بر یادگیری ماشین

استاد: دکتر صالح کلیبر

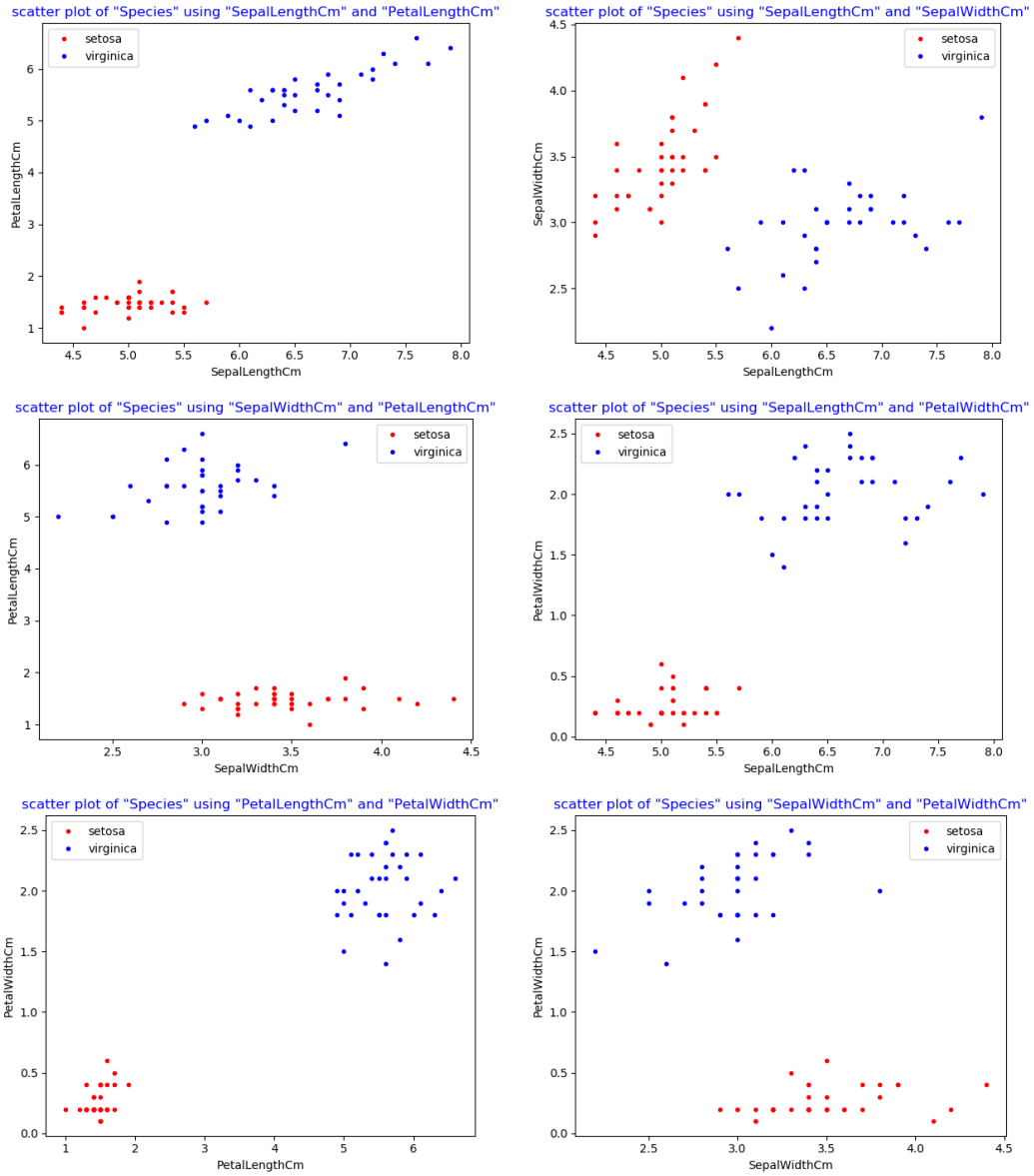
گزارش تمرین شماره ۱

سیّدمحمّدامین منصوری طهرانی

۹۴۱۰۵۱۷۴

بخش اول – رسم نمودار

در این مسأله مطابق دستور کار، داده‌ها را به دو بخش train و test تقسیم می‌کنیم. با توجه این‌که از وجود ترتیب خاصی در داده‌ها اطلاع نداریم، انتخاب ۷۰ داده اول به عنوان train و ۳۰ داده آخر به عنوان test بدون اشکال به نظر می‌رسد و با سایر انتخاب‌ها از لحاظ تصادفی بودن تفاوتی ندارد (چون از ترتیب احتمالی در داده‌ها اطلاعی نداریم.) برای بدست آوردن درک کلی از داده‌ها مطابق آن‌چه که خواسته شده نمودارهای زیر رسم می‌شوند. این تصاویر از اجرا کردن کد jupyter notebook بدست می‌آیند. هم‌چنین در کد plotter.py در فایل پیوست نیز آورده شده‌اند و با اجرای کد قابل مشاهده‌اند.



در تصاویر بالا مشخص است همان‌طور که در توصیف داده‌ها گفته شده به شدت linear separable هستند و انتظار می‌رود عملیات classify کردن به سادگی صورت پذیرد. (با دقت کامل)

❇توجه: لطفاً کدهای py. را در یک پوشه قرار دهید تا import ها دچار مشکل نشوند.

بخش دوم – یادگیری با مدل فاصله اقلیدسی

کد این بخش در قسمت اول فایل main.py که به پیوست تمرین قرار دارد نوشته شده‌است. برای این قسمت با توجه به این‌که مدل در دستور تمرین تعیین شده و کد نیز تماماً کامنت‌گذاری شده‌است، از آوردن توضیح اضافه در این‌جا خودداری می‌کنیم. لطفاً به کامنت‌ها مراجعه بفرمایید.

برای دقت مدل هم تعداد داده‌های تستی که نتیجه پیش‌بینی شده آن‌ها از مدل با برچسب آن‌ها یکی بوده را بر تعداد کل داده‌های تست تقسیم می‌کنیم.

نتیجه مطابق خواسته مسأله ۱۰۰ درصد درست پیش‌بینی شده‌است.

بخش سوم – یادگیری با مدل دلخواه

مدل انتخاب شده: Perceptron

از توضیح مدل پرسپترون با توجه به این‌که در کلاس درس تدریس شده‌است، صرف‌نظر می‌کنیم. توضیح کد:

برای استفاده از این روش یک class به نام Perceptron در فایل perceptron.py تعریف شده‌است که پس از پایان این بخش به توضیح آن می‌پردازیم.

پس از پایان کد قسمت قبل، دو آرایه label و training_inputs تعریف می‌شوند که برای آموزش طبقه‌بند استفاده می‌شوند. می‌خواهیم پرسپترون خطی استفاده کنیم که داده‌هایی با ۴ ویژگی را از هم جدا کند.(۴ ویژگی که در داده‌ها وجود دارد، طول و عرض گل‌برگ و کاس‌برگ) در حلقه اول برای داده‌های آموزش، مقادیر این ویژگی‌ها را با توجه به Perceptron ای که تعریف کرده‌ایم در متغیر training_inputs ذخیره می‌کنیم. سپس از آن‌جایی که پرسپترون ما به عنوان برچسب مقادیر ۰ و ۱ می‌پذیرد، گونه setosa را به عدد ۱ و گونه virginica را در همین داده‌های آموزش به عدد ۰ می‌نگاریم. سپس یک object از نوع Perceptron تعریف می‌کنیم با تعداد ویژگی ۴.(تنها آرگومانی که مشاهده می‌شود همین را نشان می‌دهد.) سپس با صفتی که در کلاس برای آن تعریف کرده‌ایم (train) و ورودی داده‌های آموزش و برچسب صفر و یکی آن‌ها، آن را آموزش می‌دهیم. سپس داده‌های تست به فرمت قابل استفاده برای کلاس و صفت predict آن در می‌آیند و آرایه result در حلقه بعدی آن هر بار یک مقدار جدید به عنوان پیش‌بینی برچسب داده‌های تست به خود می‌پذیرد. (با صفت

perceptron.predict) در انتها نیز شمارش‌گری مانند کد قسمت قبل برای بررسی دقت مدل انتخاب می‌شود که اگر برچسب پیش‌بینی ۱ باشد و برچسب داده نیز setosa باشد یا اگر برچسب پیش‌بینی ۰ باشد و برچسب داده virginica باشد به معنی پیش‌بینی درست خواهد بود و این شمارش‌گر زیاد می‌شود. در غیر این‌صورت زیاد نمی‌شود. در نهایت نیز تعداد پیش‌بینی‌های درست به کل داده‌های تست تقسیم می‌شود و دقت چاپ می‌شود.

نتیجه مطابق خواسته مسأله ۱۰۰ درصد درست پیش‌بینی شده‌است.

توضیح کد perceptron.py:

برای این کلاس چند متغیر ورودی در نظر گرفته‌ام. تعداد وزن‌هایی که لازم است یاد بگیریم، تعداد دفعات تکرار و آپدیت کردن وزن‌ها، نرخ یادگیری(پارامتر آپدیت کردن وزن‌ها). به این مقادیر اجازخ دسترسی در زمان runtime نیز می‌دهیم تا در صورت لزوم بتوانیم آن‌ها را تغییر دهیم. هم‌چنین دقت می‌کنیم برای بدست آوردن شکل ماتریسی صحیح بدون حضور جمله بایاس، یک وزن دیگر هم اضافه شده و ورودی این بعد ۱ می‌باشد.

پس از آن متد پیش‌بینی را برای این کلاس تعریف می‌کنیم. مقدار متغیر summation همان‌طور که واضح است، حاصل ضرب داخلی ورودی در بردار وزن‌ها(بدون وزن مربوط به جمله بایاس) به اضافه جمله بایاس می‌باشد. با توجه به این‌که تابع فعالیت را پله در نظر می‌گیریم، اگر این مجموع(که همان پیش‌بینی با این وزن‌های فعلی می‌باشد) از صفر بیشتر بود ۱ و در غیر این‌صورت مقدار ۰ را باز می‌گردانیم.

پس از آن نیز متد یادگیری را برای این کلاس تعریف می‌کنم. این متد دو متغیر داده‌های آموزش و برچسب باینری آن‌ها را برای آموزش دیدن می‌خواهد.(که نحوه تبدیل برچسب‌ها به باینری که در بالا توضیح داده شد این‌جا به کار می‌آید.) در ادامه این متد در صورتی که تعداد دفعات تکرار داده شده باشد از آن استفاده می‌کنیم و در غیر این‌صورت از مقدار default استفاده می‌شود.(آن را ۱۰۰ اختیار کردم.) در ادامه یک شئ که بتوان روی آن لوپ زد ایجاد می‌کنیم. این کار با زیپ کردن داده‌های آموزش و برچسب‌ها انجام می‌شود. در هر حلقه مقدار training_inputs در inputs و مقدار labels در label ذخیره می‌شود. سپس در ادامه همین حلقه، مقدار پیش‌بینی شده مدل با وزن‌های فعلی در prediction ذخیره می‌شود. پس از آن وزن‌ها همان‌طور که بلد هستیم تغییر می‌کنند و این حلقه به تعداد threshold تکرار می‌شود تا وزن‌ها بدست آیند.(البته وزن‌ها ممکن است قبل از رسیدن به تعداد تکرار تعیین شده نیز به مقدار صحیح رسیده باشند.)

در واقع خطای فعلی در نرخ یادگیری و سپس در ورودی‌ها ضرب شده و با وزن‌های فعلی(بدون وزن ناشی از بایاس) جمع می‌شود. وزن بایاس هم که ناشی از جمع: خطا ضرب در نرخ یادگیری ضرب در ورودی ۱ است- با مقدار قبلی خود جمع می‌شود و آپدیت می‌شود و به این ترتیب همه وزن‌ها آپدیت می‌شوند و در تکرار بعدی همین حلقه دوباره پیش‌بینی و آپدیت داریم و این چرخه تا تعداد threshold تکرار می‌شود.(که امیدواریم تا این

تکرار وزن‌ها به مقدار صحیح رسیده باشند که در این مسأله به خاطر خوبی داده‌ها و تعداد نسبتاً کم آن‌ها این وضعیت مطلوب پیش می‌آید.)

در کد اصلی main با اجرا کردن کد، داده‌های تست تک تک توسط predict پیش‌بینی شده و دقت آن‌ها همان‌طور که گفته شد محاسبه و چاپ می‌شود.

خروجی کد پس از پایان اجرای کد main.py که خود از perceptron.py برای قسمت آخر استفاده می‌کند. دقت هر دو روش ۱۰۰ درصد است. با توجه به معروف بودن داده‌ها این انتظار می‌رفت. داده‌های این مشاهده به شدت linear separable هستند و این دقت را مدیون این ویژگی آن‌ها هستیم.