یادگیری ماشین

رویا شاهرودی 40240112093 تمرین سری 1–گزارش

الف) دیتاست **MNIST (Modified National Institute of Standards and Technology)** یکی از معروف ترین مجموعه داده‌ها در حوزه یادگیری ماشین و پردازش تصویر است. این دیتاست شامل 70,000 تصویر دست‌نویس از اعداد 0 تا 9 است که هر تصویر دارای ابعاد 28x28 پیکسل است. این مجموعه به‌عنوان یک استاندارد برای ارزیابی و مقایسه الگوریتم‌های مختلف یادگیری ماشین مورد استفاده قرار می‌گیرد. اهمیت این دیتاست به دلیل سادگی، گستردگی و قابلیت فهم آسان آن است که باعث شده به یک نقطه شروع مناسب برای آزمایش الگوریتم‌ها در پردازش تصویر و یادگیری ماشین تبدیل شود.

ب) برای استفاده آسانتر از این مجموعه داده از طریق keras.datasets از آن استفاده میکنیم. این مجموعه بصورت پیش فرض به دو بخش train و test با اندازه های 60000 و 10000 تقسیم شده است که ما نیز به همین صورت از این تقسیم بندی استفاده میکنیم.

برای مثال نمونه با index=5 موجود در مجموعه train را به همراه عکس و لیبل با شماره 2 را نشان میدهیم.

پ) با استفاده از دستور reshape در کتابخانه numpy ابعاد نمونه های train را از (60000, 28, 28) به (60000, 784) و همچنین ابعاد نمونه های test را نیز از (10000, 28, 28) به (10000, 784) تبدیل میکنیم.

در این قسمت داده‌های تصویری که به صورت ماتریس 28x28 هستند، به بردارهایی با طول 784 تبدیل می‌شوند تا به راحتی بتوان از آن‌ها در مدل‌های یادگیری ماشین استفاده کرد. هر تصویر به یک بردار تبدیل می‌شود که تمام پیکسل‌های تصویر را شامل می‌شود.

ت) در همان ابتدا در هنگام لود مجموعه داده، داده‌های train_X و train_y به عنوان مجموعه آموزشی و test_X و test_y به عنوان مجموعه تست تخصیص داده می‌شوند. داده‌ها شامل تصاویر (X) و برچسب‌های مربوط به آن‌ها (y) هستند.

برای جداسازی داده های validation از تابع train_test_split در کتابخانه sklearn استفاده میکنیم که ابتدا داده ها و سپس درصدی که میخواهیم جدا کنیم را بعنوان ورودی میگیرد و در نتیجه آن 10 درصد از داده های train که برابر 6000 سطر میشود در متغیر validation قرار میگیرد.

ث) ابتدا تابع هایی مخصوص محاسبه فاصله های منهتن و اقلیدسی میسازیم که با گرفتن بردار دو نقطه فاصله بین آنها را محاسبه میکند.

تابع اصلی knn بعنوان ورودی مقادیر x که همان نقطه مورد نظر است که میخواهیم کلاس آن را پیدا کنیم، پارامتر k که تعداد همسایه هاست، data که مجموعه داده ایست که میخواهیم از آن استفاده کنیم(داده های train)، label که آرایه ای از کلاس های مجموعه داده ورودیست و distance که تابع مورد استفاده برای اندازه گیری فاصله بین نقطه هاست.

به دلیل زمان اجرای بالا این قطعه کد با استفاده از GPT توانستم حلقه های for استفاده شده را حذف کرده و از تابع های numpy استفاده بیشتری ببرم و با اینکار زمان اجرا کاهش یافت و خروجی با استفاده از این قطعه کد نمایش داده شده است.

تابع apply_along_axis به‌جای استفاده از حلقه for برای محاسبه فاصله‌ها به کار گرفته شده تا عملیات محاسباتی را بهینه‌تر و سریع‌تر کند.

برای مثال نیز یک نمونه از داده test را به تابع knn با k=5 میدهیم تا کلاس آن را تشخیص دهد و همانطور که میبینیم مقدار پیش بینی شده با مقدار واقعی برار است.

۴) برای یافتن k مناسب اعداد فرد 1 تا 9 را به تابع knn میدهیم و با استفاده از داده validation بررسی میکنیم که چه تعداد از لیبل های داده های validation به درستی پیش بینی میشوند و در آخر این تعداد را تقسیم بر تعداد کل داده های ورودی میکنیم تا مقدار accuracy را بدست آوریم.

هر کدام از k های مختلف که دقت بالاتری را بعنوان خروجی دهد بعنوان k مناسب انتخاب میشود. برای پیدا کردن عدد k مربوط به بیشتری دقت با استفاده از تابع argmax(acc) ایندکس مربوط به بیشتری دقت پیدا میشود و با ضرب در 2 و بعلاوه 1 کردن ایندکس، مقدار اصلی k را میتوان بدست آورد.

۵) با استفاده از تابع concatenate از کتابخانه numpy داده های train و validation را ادغام کرده و به عنوان ورودی به تابع محاسبه accuracy میدهیم.

یکبار در پارامتر آخر تابع accuracy تابع فاصله اقلیدسی را قرار میدهیم و بار دیگر تابع فاصله منهتن را قرار میدهیم و خروجی را گزارش میکنیم.

۶) در این قسمت با استفاده از تابع KNeighborsClassifier از کتابخانه sklearn الگوریتم knn را با k=3 که در قسمت های قبلی بعنوان پارامتر مناسب شناخته شده بود اجرا میکنیم و با

استفاده از تابع fit عملیات آموزش انجام میشود. این ابزار آماده، روش‌های بهینه‌تری برای محاسبه دارد.

سپس مدل آموزش داده شده، روی داده‌های تست پیش‌بینی می‌کند و سپس دقت مدل با استفاده از accuracy_score محاسبه میشود.