Report
HW#4

Shiva vafadar
810899074

بخش عملی :

۱.

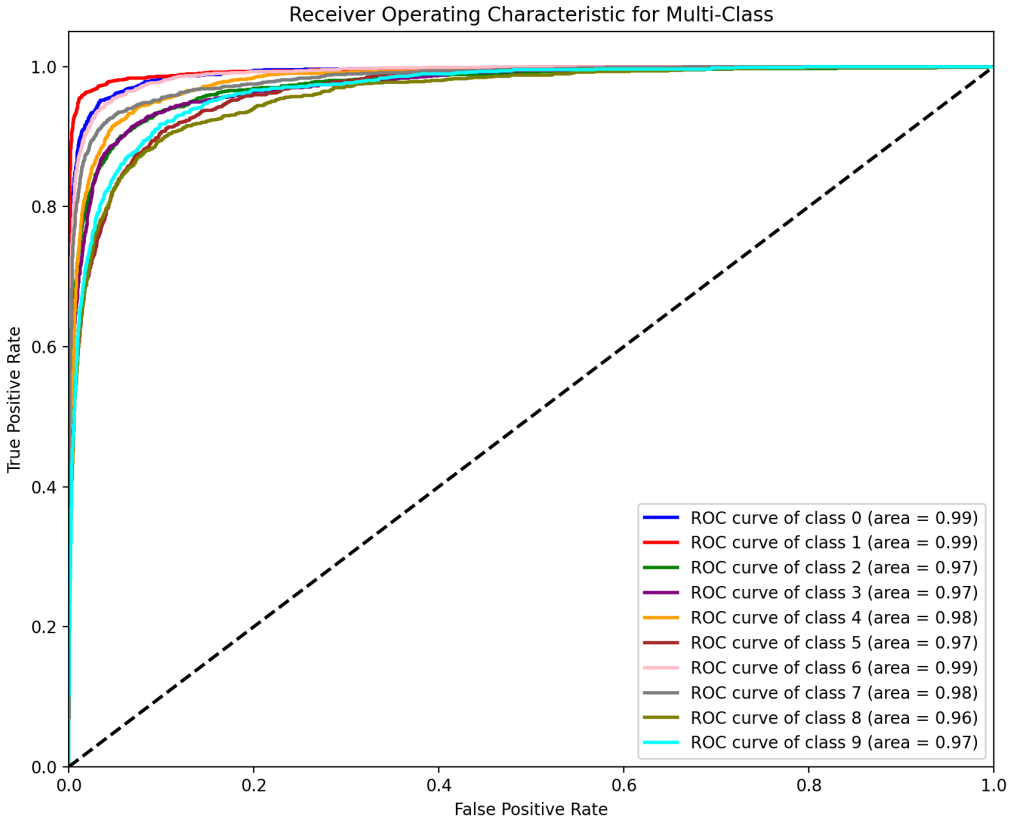
الف)

```
Class Accuracies:
Class 0: 0.0853
Class 1: 0.1077
Class 2: 0.0796
Class 3: 0.0846
Class 4: 0.0761
Class 5: 0.0638
Class 6: 0.0881
Class 7: 0.0921
Class 8: 0.0705
Class 9: 0.0781
Overall Accuracy: 0.8259
```

ب)

```
Class 0 - Precision: 0.8844, Recall: 0.8891, F1-Score: 0.8867
Class 1 - Precision: 0.9297, Recall: 0.9425, F1-Score: 0.9361
Class 2 - Precision: 0.8114, Recall: 0.8072, F1-Score: 0.8093
Class 3 - Precision: 0.7781, Recall: 0.8269, F1-Score: 0.8018
Class 4 - Precision: 0.8080, Recall: 0.8224, F1-Score: 0.8152
Class 5 - Precision: 0.7600, Recall: 0.7015, F1-Score: 0.7296
Class 6 - Precision: 0.8720, Recall: 0.8832, F1-Score: 0.8776
Class 7 - Precision: 0.8646, Recall: 0.8583, F1-Score: 0.8614
Class 8 - Precision: 0.7581, Recall: 0.7273, F1-Score: 0.7424
Class 9 - Precision: 0.7638, Recall: 0.7697, F1-Score: 0.7667
```

ج)



د)
نمره جکارد یا شاخص جکارد یک معیار عملکرد برای ارزیابی شباهت و تنوع مجموعه‌های نمونه است. این نمره شباهت بین برچسب‌های صحیح و پیش‌بینی‌ها را اندازه‌گیری می‌کند که تعریف می‌شود به عنوان اندازه تقاطع تقسیم بر اندازه اتحاد مجموعه‌های برچسب. به طور اساسی، این نمره بازتابدهنده تعداد برچسب‌های پیش‌بینی شده‌ای است که با برچسب‌های درست در یک نمونه مطابقت دارند.
برای محاسبه نمره جکارد در زمینه یادگیری ماشین، به خصوص در وظایف طبقه‌بندی، می‌توانید از تابع jaccard_score در ماژول sklearn.metrics استفاده کنید. این تابع قابل تنظیم است و می‌تواند سناریوهای طبقه‌بندی دودویی، چندکلاسی و چندبرچسبی را مدیریت کند. بسته به تنظیمات، ممکن است پارامترهایی مانند میانگین (گزینه‌ها شامل 'weighted' ،'samples' ،'macro' ،'micro'، یا 'binary') را برای محاسبه نمره بر اساس کاربرد خاص تنظیم کنید.

در مشکل چندکلاسی، جایی که هر نمونه به یکی از چندین کلاس اختصاص داده شده است، نمره جکارد می‌تواند برای هر کلاس محاسبه شود یا به چندین روش میانگین گیری شود:

- 'binary': تنها نتایج برای کلاس مشخص شده توسط pos_label گزارش می‌شود. این تنها زمانی قابل اجرا است که هدف‌ها دودویی باشند.
- 'micro': معیارها را به صورت جهانی با شمارش کل مثبت‌های واقعی، منفی‌های کاذب و مثبت‌های کاذب محاسبه می‌کند.
- 'macro': معیارها را برای هر برچسب محاسبه می‌کند و میانگین آنها را بدون وزن‌دهی پیدا می‌کند. این موضوع تعادل برچسب‌ها را در نظر نمی‌گیرد.
- 'weighted': معیارها را برای هر برچسب محاسبه می‌کند و میانگین آنها را، وزن‌دهی شده توسط تعداد موارد واقعی برای هر برچسب، پیدا می‌کند.
- 'samples': معیارها را برای هر نمونه محاسبه می‌کند.

```
Jaccard Score (Macro Average): 0.7038
```

۲.
مراحل اجرای کد:

1. **بارگذاری داده‌ها:** داده‌های MNIST که شامل تصاویر دست‌نویس ارقام هستند، بارگذاری می‌شوند. هر تصویر به صورت یک بردار 784 بُعدی (28x28 پیکسل) است.
2. **پیش‌پردازش داده‌ها:** داده‌ها به صورت نرمال شده (تقسیم بر 255) و برچسب‌ها به صورت یک‌هات انکود می‌شوند.
3. **کاهش بُعد با PCA:** با استفاده از PCA، بُعد داده‌ها کاهش می‌یابد. در این کد، تعداد ویژگی‌ها به 100 کاهش یافته‌است.
4. **تعریف شبکه عصبی:** یک شبکه عصبی با یک لایه پنهان تعریف شده است. تعداد نورون‌ها در لایه ورودی به 100 (به دلیل کاهش بُعد توسط PCA)، لایه پنهان 50 و لایه خروجی 10 (برای 10 کلاس ارقام) تنظیم شده‌اند.
5. **آموزش شبکه عصبی:** شبکه با داده‌های آموزشی که بُعد آن‌ها کاهش یافته آموزش داده می‌شود. همچنین، دقت شبکه در هر ایپوک روی داده‌های آموزشی و تست محاسبه و چاپ می‌شود.

نتایج:
این کد دقت شبکه عصبی را در پیش‌بینی ارقام دست‌نویس از داده‌های MNIST بعد از اجرای PCA اندازه‌گیری می‌کند. دقت بر روی داده‌های آموزشی و تست در هر دور از آموزش (ایپوک) چاپ می‌شود که امکان مقایسه عملکرد شبکه در طول زمان را فراهم می‌آورد. این اطلاعات می‌توانند در تحلیل بهینه‌سازی و تنظیم پارامترهای شبکه به منظور بهبود عملکرد مفید باشند.

-بله، می‌توانیم با استفاده از تکنیک تحلیل مؤلفه‌های اصلی یا PCA (Principal Component Analysis)، بُعد داده‌ها را کاهش دهیم و در عین حفظ 95 درصد از اطلاعات، تعدادی از ویژگی‌ها را حذف کنیم. این کار می‌تواند به چند دلیل مفید باشد:

چرا استفاده از PCA مفید است؟

1. کاهش پیچیدگی محاسباتی: وقتی تعداد ویژگی‌ها کاهش می‌یابد، مدل‌های یادگیری ماشین می‌توانند سریع‌تر آموزش ببینند، زیرا حجم داده‌هایی که باید پردازش شوند کمتر است.

2. کاهش ابعاد: با کاهش ابعاد، مشکلات ناشی از "نفرین ابعاد" (Curse of Dimensionality) که در آنالیز داده‌های با ابعاد بالا اتفاق می‌افتد، کاهش می‌یابد. این مشکلات شامل داده‌های پراکنده و نیاز به حجم نمونه بسیار بزرگتر برای تخمین مدل‌ها است.

3. بهبود عملکرد مدل: در برخی موارد، حذف ویژگی‌هایی که اطلاعات کمی دارند یا نویز هستند، می‌تواند به کاهش اثر overfitting کمک کرده و عملکرد مدل بر روی داده‌های تست بهبود یابد.

چگونگی اجرای PCA برای حفظ 95% اطلاعات

از آنجا که خواستار حفظ 95% از اطلاعات هستید، می‌توانیم تعداد مؤلفه‌ها را به گونه‌ای انتخاب کنیم که این میزان از واریانس داده‌ها را توضیح دهند. این کار را می‌توان با استفاده از PCA در `sklearn` انجام داد:

```python
from sklearn.decomposition import PCA

تنظیم PCA برای حفظ 95% از واریانس
pca = PCA(n_components=0.95)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)

print(f"Number of components retained to preserve 95% of variance: {pca.n_components_}")
```

در این کد، `PCA(n_components=0.95)` خودکار تعداد مؤلفه‌ها را به گونه‌ای انتخاب می‌کند که دست‌کم 95% از کل واریانس داده‌ها حفظ شود. این به ما اجازه می‌دهد که داده‌ها را با حداقل از دست دادن اطلاعات مهم، به فضایی با بعد پایین‌تر منتقل کنیم.