بینایی کامپیوتری

نیم‌سال دوم ۱۳۹۹-۱۴۰۰

مدرس: دکتر مصطفی کمالی تبریزی

دانشکده علوم ریاضی

تمرین سری چهارم

نام و نام‌خانوادگی: امین کشیری

شماره دانشجویی: ۹۷۱۰۱۰۲۶

توضیحات کلی

- عکس‌های ورودی باید در پوشه‌ی inputs/images قرار بگیرند.
- خروجی تمام کدها، با همان اسم گفته شده در صورت سوال، در پوشه‌ی outputs/images قرار خواهد گرفت (اگر عکس دیگری نیز در این پوشه باشد، در توضیحات سوال مربوطه نوشته‌ام).
- به جز دو پوشه‌ی بالا، یک پوشه‌ی دیگر نیز نام temps وجود دارد. این پوشه خود به خود در حین اجرا ساخته می‌شود و برای نگه‌داری از داده‌های میانی است.
- بعضی از خط‌ها در کد، دارای comment هستند، که با uncomment کردن آن‌ها معمولاً می‌توانید عکس را در مراحل میانی ببینید (توضیحات بیشتر را در صورت لزوم، در توضیحات هر سوال داده‌ام).
- همراه فایل‌های یک فایل requirements.txt قرار دارد که محیط اجرای کدهای من است (در صورت نیاز).

سوال ۱

آماده‌سازی

ابتدا باید دو پوشه‌ی inputs/images و outputs/images ساخته‌شده باشند. سپس باید دو دیتاست lfw و kaggle را دانلود کرده و در پوشه‌ی inputs/datasets قرار دهید (من فرض کرده‌ام دیتابیس lfw در پوشه‌ای با همین اسم، و دیتابیس kaggle نیز در پوشه‌ی ای با نام archive قرار گرفته‌اند، زیرا این دو همان اسم‌هایی هستند که بعد از extract کردن در آن قرار می‌گیرند، و نیازی به تغییر این دو اسم ندارید). دیتابیس kaggle را می‌توانید از https://www.kaggle.com/prasunroy/natural-images دانلود کنید.

توضیحات کلی کد

کد من از تعدادی تابع تشکیل شده‌است که هرکدام را در مکان خود توضیح می‌دهم. پس از تعاریف توابع، تعدادی از ثوابت وجود دارند که مقدار دهی می‌کنم. سپس روند اصلی کد اجرا می‌شود.

- VERBOSE : این متغییر در صورتی که فعال باشد، توضیحات بیشتری حین اجرای کد چاپ می‌شوند.
- FIND_BEST_PARAMETERS : در صورتی که این متغییر فعال شود، کد وارد حالت پیدا کردن بهترین پارامترها می‌شود و در نهایت قبل از ادامه دادن بهترین پارامترها را خروجی می‌دهد. در ادامه بیشتر توضیح خواهم داد.
- RUN_ON_TEST : در صورتی که این متغییر فعال باشد و متغییر قبلی غیر فعال باشد، یک بارمدل را روی test اجرا می‌کند و دقت را چاپ می‌کند. در صورتی که متغییر قبلی فعال باشد، این اتفاق به صورت خودکار رخ می‌دهد (با بهترین پارامتر) و نیازی به فعال کردن این متغییر نیست.
- N_BEST_PARAMETERS : اگر کد در حالت پیدا کردن پارامترها باشد، به این اندازه از بهترین نتیجه‌ها را نگه می‌دارد (مثلا اگر این متغییر برابر ۵ باشد، ۵ تا بهترین نتیجه را نگه می‌دارد)
- NMS_THRESHOLD : حد IoU ی که در آن دو bounding_box را یکی در نظر می‌گیریم و یکی را حذف می‌کنیم (در الگوریتم Non Maximum Suppression در انتهای کار و پیدا کردن صورت‌ها).

- THRESHOLD : حدی که در آن صورت تشخیص می‌دهم. هرچه کمتر باشد صورت‌های بیشتری تشخیص می‌دهیم (اما ممکن است اشتباه هم تشخیص دهیم) و برعکس. میزان آن را در کد برابر 0.995 گذاشتم که به صورت تجربی بهینه بود.
- تمامی ۱۰ متغییر بعدی آدرس‌های پوشه‌های مخلف کد من اند که نیاز به توضیحات ندارند.
- ۷ لیست بعدی، تمام پارامترهایی هستند که قرار است روی آن‌ها پارامترهایم را بهینه کنم. این ۷ لیست به ترتیب: BLOCK_SIZES ، CELL_SIZES ، IMAGE_SIZES ، BLOCK_NORMS ، ORIENTATIONS ، KERNELS و DECISIONS هستند. به ازای تمام حالت‌های ممکن در این لیست‌ها، مدل را آموزش می‌دهیم و روی دیتای validation تست می‌کنیم و بهترین‌ها را انتخاب می‌کنیم. دقت کنید که این لیست‌ها تنها وقتی کاربرد دارند که در حالت پیدا کردن بهترین پارامترها باشیم.
- ۷ مقدار بعدی، تمام پارامترهایی هستند که قرار است با آن‌ها مراحل بعدی را اجرا کنیم. به ترتیب: ORIENTATION ، BLOCK_NORM ، IMAGE_SIZE ، CELL_SIZE ، BLOCK_SIZE ، KERNEL و DECISION هستند. اگر دقت کنید لیست‌های قبلی، مقادیر مختلف برای این پارامترها را نگه می‌داشتند. از روی اسم آن‌ها، مشخص است که هرکدام چه پارامتری اند.

  IMAGE_SIZE سایزی است که تمام عکس‌ها را به آن ریسایز کرده‌ام تا مدل را آموزش دهم. DECISION نیز پارامتر SVM است که یا در حالت one vs one است یا one vs rest . در گذارش ذکر شده بود که بهترین مقادیری که پیدا کرده‌اید را در گذارش بنویسیم، که این مقدایر به صورت زیر بودند:

IMAGE_SIZE: (100,100)

ORIENTATION: 16

CELL_SIZE: (8,8)

BLOCK_SIZE: (2,2)

BLOCK_NORM: L2

KERNEL: poly

DECISION: ovo

به کمک مقادیر بالا، من توانستم به دقت 0.998 روی داده‌های اعتبارسنجی و دقت 0.9975 روی داده‌های تست برسم.

روند اجرای کد

تابع create_dir در تمرین قبلی نیز وجود داشت.

تابع Print همان پرینت کردن را انجام می‌دهد، منتهی فقط در حالتی که VERBOSE فعال باشد.

تابع get_feature_vectors_size به کمک پارامترهای حال حاضر برای HOG ، اندازه‌ی feature vector های روش HOG را محاسبه می‌کند و برمی‌گرداند.

در ابتدای کد، تابع unpack_dataset صدا زده می‌شود. این تابع تمام عکس‌ها را در یک پوشه می‌ریزد، عکس‌های منفی (بدون چهره) را نیز به کمک تکنیک flip کردن augment می‌کند تا تعداد آن‌ها افزایش یابد.

حال دو حالت داریم. اگر کد در حالت پیدا کردن بهترین پارامترها باشد، ابتدا بهترین پارامترها پیدا می‌شوند و بعد روی داده‌های تست، با بهترین پارامتر پیدا شده مدل را تست می‌کنیم. در غیر این صورت، با داده‌هایی که در قسمت قبل set کرده بودم، مدل را تست می‌کنیم. اما در هر دو حالت، یک مدل روی داده‌های تست آزمایش می‌شود، نمودارهای ROC و Precision-Recall کشیده می‌شوند و Average Precision نیز چاپ می‌شود.

سپس با استفاده از تابع FaceDetector صورت‌های عکس‌های ورودی را تشخیص می‌دهیم. دقت کنید که مدلی که در این مرحله استفاده می‌شود، اگر در حالت پیدا کردن پارامترها باشیم بهترین پارامتر پیدا شده، و در غیر این صورت با پارامترهای set شده است (در ۷ متغییری که اول توضیحات گفتم).

حال، توضیحات توابع گفته شده را به صورت دقیق تر بیان می‌کنم:

- unpack_dataset

در این تابع، یک تابع به نام unpack_dataset_util داریم، که یک آدرس ورودی و خروجی می‌گیرد، به همراه تعداد کل عکس‌هایی که قرار است در آدرس ورودی وجود داشته باشد. این تابع سپس روی تک تک پوشه‌هایی که داخل آدرس ورودی قرار دارند حرکت می‌کند و ساختار پوشه بندی آن را از بین می‌برد. در واقع تمامی عکس‌ها را از آنجا کپی کرده (فقط عکس‌ها و نه همراه با پوشه‌ی مادر آن‌ها) و در آدرس خروجی می‌ریزد. در آدرس خروجی، اسم هرکدام از عکس‌ها را به صورت تصادفی، عددی بین ۰ تا تعداد کل عکس‌های ورودی قرار می‌دهد. این کار باعث می‌شود که در حین unpack کردن (از بین بردن ساختار پوشه‌بندی دیتاست ورودی) تمام عکس‌ها shuffle شوند و ترتیب آن‌ها تصادفی باشد. این تابع در ابتدای اجرا، اگر در آدرس خروجی تمامی عکس‌ها وجود داشته باشند، دوباره این مراحل را طی نمی‌کند. یعنی تنها بار اول اجرا کافی است، و از دفعه‌های بعدی این کد کاری انجام نمی‌دهد (مگر این که مثلا پوشه‌ی خروجی را پاک کرده باشید).

در تابع اصلی، ما یک بار برای دیتاست صورت‌ها و یک بار برای دیتاست غیرصورت، تابع util را صدا می‌زنیم. آدرس ورودی آن‌ها که آدرسی است که در ابتدای این داکیومنت خواسته‌ام دیتاست‌ها را در آن‌ها قرار دهید. آدرس خروجی آن‌ها نیز به ترتیب inputs/datasets/face_dataset و inputs/datasets/negative_dataset است.

تابع دیگری که در این تابع استفاده شده است نیز تابع double_negative_dataset که همان مراحل دوبرابر کردن با کمک flip و ... را انجام می‌دهد.

- compute_feature_vectors

این تابع با توجه به پارامترهای موجود، feature vector ها را برای تمام داده‌های ما می‌سازد و ذخیره می‌کند. مانند تابع قبلی، این تابع نیز یک تابع util درون خودش دارد. که به این صورت کار می‌کند که آدرس پوشه‌ای که عکس‌ها در آن قرار گرفته‌اند را می‌گیرد (عکس‌ها باید همه در یک پوشه باشند، که تابع قبلی ما این کار را می‌کند) و بازه‌ای از اعداد و یک اسم می‌گیرد. سپس عکس‌های آن بازه از اعداد را می‌گیرد، روی تک تک آن‌ها به کمک تابع hog و پارامترهایی که set کرده‌ایم بردارهای ویژگی را به دست می‌آورد، و سپس تمام بردارهای ویژگی را روی دیسک ذخیره می‌کند. دقت کنید که بازه‌های مختلف از عکس‌های من، به ترتیب عکس‌های train و validation و test من شده‌اند. زیرا در مرحله‌ی قبلی shuffle کرده بودم و الان این خاصیت را دارند که به صورت تصادفی پشت هم قرار گرفته‌اند. همچین اسمی که بردارهای ویژگی را با آن نام ذخیره می‌کند، هم شامل تمامی پارامترها است (به جز پارامترهای مربوط به SVM) و هم شامل این که داده‌ی train است یا test و .... . همچنین در این تابع اول چک می‌شود که آیا بردارهای ویژگی قبلا روی دیسک ذخیره شده‌اند یا نه و در این صورت دوباره از اول محاسبه نمی‌کند.

- run_test

این تابع، با استفاده از پارامترهای حال حاضر، یک مدل آموزش می‌دهد، که با استفاده از تابع get_SVM_model انجام می‌شود. این تابع بردارهای ویژگی با پارامترهای الان را از روی دیسک می‌خواند (فرضش این است که وجود دارند، زیرا در مرحله‌ی قبلی تابع قبلی حتما آن‌ها را ساخته است). سپس با کمک sklearn مدل را با این بردارهای ویژگی و پارامترهای من آموزش می‌دهد، سپس خود مدل را نیز روی دیسک ذخیره می‌کند. زیرا آموزش دادن مدل نیز کار هزینه بری است. در صورتی که از قبل مدلی با این مشخصات روی دیسک وجود داشته باشد، از اول آموزش نمی‌دهد و همان را می‌خواند. سپس مدل را (classifier) برمی‌گرداند.

حال که مدل را داریم، بردارهای ویژگی یا اعتبار سنجی (بسته به این که متغییر ورودی تابع True بوده است یا False) را از روی دیسک می‌خوانیم، و مدل خود را تست می‌کنیم و دقت آن را چاپ می‌کنیم.

حال دو اتفاق ممکن است رخ دهد. اگر واقعا روی داده‌ی test ، تست کرده باشیم، یعنی این تست اصلی است. و در این حالت منحنی‌های خواسته شده در صورت سوال را می‌سازیم و ذخیره می‌کنیم. میزان Average Precision را نیز به دست می‌آوریم که در بهترین حالت برابر بود با:

$$\text{Average Precision} = 0.9999919959599176$$

اما اگر روی داده‌های validation کار کرده باشیم، یعنی در حالت پیدا کردن بهترین پارامترها هستیم. در این صورت تابع keep_records صدا زده می‌شود، که اطلاعات تمام پارامترها و نتیجه‌ی آن‌ها را ذخیره می‌کند (در مورد آن توضیح خواهم داد).

نحوه‌ی پیدا کردن بهترین پارامتر

اگر به کد دقت کنید، در حالتی که می‌خواهیم بهترین پارامترها را پیدا کنیم، روی تمام پارامترهای ممکن (که در لیست‌های اولیه گذاشته‌ام) حرکت می‌کنیم وتمام مراحل آموزش و تست را روی آن‌ها انجام می‌دهیم. در کد من یک لاگر وجود دارد که نتایج این تست‌ها را یادداشت می‌کند. آماده‌سازی اولیه این لاگر در تابع init_logger انجام می‌شود. دقت کنید که هر دسته از پارامترها یک لاگ جدید (با اسم جدید زیرا پارامترهایی که روی آن‌ها تست می‌کنیم تغییر کرده است) ایجاد می‌کند. تمام لاگ‌ها در پوشه‌ی temps_logs قرار می‌گیرند و در صورت علاقه می‌توانید آن‌ها را نگاه کنید. دلیل لاگ کردن این بود که اگر به هر دلیلی اجرا با خطا متوقف شد راهی برای بازیابی نتایج داشته باشم. برای هر سایزی که روی آن آموزش انجام می‌دهم نیز یک فایل لاگ جدا ساخته می‌شود. یکی از فایل‌های لاگ را برای نمونه در کنار بقیه فایل‌ها در فایل zip با نام example.log قرار دادم.

اما تابعی که در این قسمت آن را توضیح ندادم، تابع keep_records بود. این تابع، از اشیایی از کلاس Record استفاده می‌کند. این کلاس، تمامی اطلاعات مورد نیاز من در یک دور از اجرا را ذخیره می‌کند (تمام پارامترهای مورد نیاز). همچنین این کلاس طوری ساخته شده است که عملگر بزرگتر (<) بر اساس دقت به دست آمده از داده‌های validation کار کند. یک داده‌ساختار min heap نیز دارم، که در آن همواره N نتیجه‌ی بهتر را ذخیره می‌کنم (این داده‌ساختار به این دلیل درست کار می‌کند که عملگر بزرگتر را در کلاس Record تعریف کرده‌ام). این تابع پس از هر بار اجرا (با یک سری پارامتر خاص)، Record به دست آمده را لاگ می‌کند، سپس آن را به مجموعه‌ی تمام Record های قبلی اضافه می‌کند، و سپس کوچک ترین آن‌ها (کمترین accuracy) را حذف می‌کند. به صورت خلاصه، به کمک این تابع، N تا بهترین نتیجه‌ام را نگه می‌دارم (در داده‌ساختار heap).

پس از این که بر روی همه‌ی پارامترها حرکت کردم، از این داده ساختار، بهترین Record را برمی‌دارم، پارامترهای آن را به عنوان پارامترهای نهایی در نظر می‌گیرم و مدل با این پارامترها را روی دیتاست test ، آزمایش می‌کنم.

نحوه‌ی کار FaceDetector

زمانی که نوبت به اجراهای این تابع می‌رسد، ما بهترین مدل را پیدا کرده‌ایم. در ابتدای این تابع به کمک تابع get_SVM_model مدل برای بهترین پارامترها را می‌گیریم. دقت کنید که این مدل کمی با مدل‌های قبلی فرق دارد و حالت probability دارد. یعنی ۰ و ۱ی جواب نمی‌دهد بلکه به صورت احتمالاتی پاسخ می‌دهد که آیا صورت است یا خیر.

حال، از بزرگترین سایز عکس شروع می‌کنیم. یک sliding window روی آن شروع به حرکت می‌کند، و سپس descriptor آن پیدا می‌شود و تصمیم گرفته می‌شود با چه احتمالی صورت است. اطلاعات تمام window هایی که احتمال صورت بودن آن‌ها بیشتر از THRESHOLD است را در دو آرایه‌ی bounding_boxes و scores ذخیره می‌کند. این اطلاعت شامل نقطه‌ی شروع bounding_box و طول و عرض آن و احتمال صورت بودن آن‌ها است.

در هر مرحله‌ی، scale عکس را 0.05 کوچک می‌کنم و دوباره مراحل بالا را تکرار می‌کنم. این کار را آنقدر انجام می‌دهم که دیگر window داخل عکس جا نگیرد (تا آخرین جای ممکن). پس از اتمام این مرحله، با کمک تابع NMSBoxes در کتابخانه‌ی openCV ، bounding_box هایی که با هم اشتراک دارند را حذف می‌کنیم.

پس از این مرحله، bounding_box های باقی مانده را روی شکل می‌کشیم و شکل نهایی را خروجی می‌دهیم.

نکته: sliding_window من، گام‌هایی به طول step دارد که در کد آن را برابر ۷ گذاشته‌ام. زیرا اگر گام‌ها یکی یکی بود، کد بسیار کند اجرا می‌شود و شاید نزدیک به ۱ ساعت طول می‌کشید. همچنین، می‌شد گام را بیشتر از ۷ نیز گذاشت. مثلا با گام‌های برابر ۱۰ کد بسیار سریع تر هم می‌شد. ولی من با گام برابر با ۷ نتیجه‌های بهتری گرفتم و ترجیح دادم کمی زمان را از دست بدهم اما تشخیص هایم بهتر باشد.

درمورد threshold مثلا در عکس استقلال، می‌شد آن را کمی بیشتر گذاشت تا آن دو نفر پشت تصویر را تشخیص ندهد. البته به نظر من این تشخیص، تشخیص خوبی بود و به همین دلیل این کار را نکردم (منظور این است که از نظر دیداری این نتیجه را ترجیح دادم تا حالتی که افراد خیلی دور را اصلا نگیرد).