گزراش پروژه نهایی

هوش محاسباتی

SUM FOR KIDS

هلیا گهرباونگ

ملیکا شکرریز

رومینا روشنی

امین رزاقی

SUM FOR KIDS

مفهوم و هدف نهایی این پروژه به این صورت است که عبارت ریاضی به صورت ساده نوشته شود و به کاربر نشان داده شود سپس کاربر حدس نهایی خود را روی کاغذ و یا صفحه نمایشگری بنویسد و روبروی وب کم لپتاپ اصلی بگیرد ، کار اصلی که باید در این پروژه انجام شود ، پردازش تصویر و تشخیص عدد یا اعداد نوشته شده است .

پس از تشخیص درست عدد نوشته شده باید پاسخ کاربر مورد ارزیابی قرار گیرد .

مراحل انجام پروژه

1.Database

اولین قدم در راخ حل ما ساختن و یا پیدا گردن دیتابیس برای رقم ها بود ، دیتابیسی که باید شامل رقم های دست نوشته باشد ، به دلیل اینکه ساختن چنین دیتابیسی زمان زیادی می برد و اینکه پیدا کردن دیتا بیس هدف اصلی ما نبودُ ، ما راه انتخاب دیتا بیس را انتخاب نکردیم ، خبر خوب این بود که ما از دیتابیس مشهور MNIST که شمال ۷۰۰۰۰ زفم دست نوشته بودُ که هر کدام از آنها یک عکس سیاه سفید در سایز ۲۸*۲۸ بود ما از کتابخانه sklearn.datasets برای دانبود این کتابخانه از سایت mldata.org استفاده کردیم . سایز این دیتا بیس حدود ۵۴ مگا بیت بود ، و تنها نیاز بود یک بار دانلود شود و بعد از آن در کش سیستم اصلی ذخیره می شد .

شکل ۱ – نمونه ای از دیتا ست

2. Train

مرحله بعدی ، اصلی ترین مرحله این پروژه و هدف اصلی این درس بود ، حال باید شبکه عصبی می ساختیم که بر اساس این دیتابیس موجود الگوریتمی را پیدا کند و یاد بگیرد تا اگر داده ای به صورت داده از طریق وب کم به آن نشان داده شود بتواند ، رقم را تشخیص دهد .

اولین کار برای این امر ، پیدا کردن HOG features های این عکس های دیتا بیس بود .

HOG features

در حقیقت HOG features که مخفف Histogram of Oriented Gradients است یک feature descriptor می باشد که به طور کلی هر feature descriptor اطلاعات اصلی و مفید عکس را می گیرد و اطلاعات غیز صروری آن را پاک می کند ، اما منظور از اطلاعات مفید چیست ، تصور کنید که میخواهید عکسی را پردازش کنید و به طور خصوص ،شی خاصی را در آن عکس تشخیص دهید object detector ، مشخص است که در این مسیله خاص که تفریبا همان مشکل ما هم هست ، اطلاعاتی مثل رنگ عکس خیلی اهمیت ندارد و گوشه ها و لبه ها برای ما اهمیت بیشتری دارند ، همان مرزهایی که شکل ها و اجسام را از هم جدا می کند ، پس می توانیم از اطلاعاتی مثل رنگ صرف نظر کنیم .

حال HIOG نیز همین کار را تقریبا می کند ، این توضیف کننده ویژگی ، ۲تا پارامتر دارد ، یکی گرادینت در جهت x و دیگری در جهت y ، پس می شود به طور کلی گفت که مفهموم گرادینت را تداعی می کند مفهومی مانند مفهوم مشتق و تغییرات .

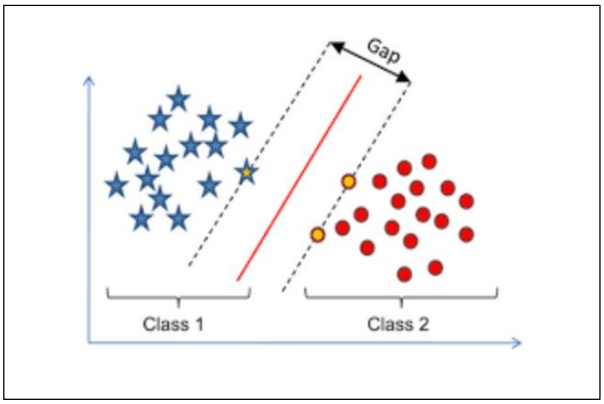
هر پیکسل عکس ، گرادیانت دارد که به طور خلاصه تر ، گرادینت در هر پیکسل هم شامل اندازه و هم شامل جهت می باشد و در رابطه با عمس های رنگی عنصر رنگ را نیز دخیل می کند ،و اندازه ماکزیمم که د ر االمه توضیح می دهیم از اهیمیت ویژه ای برای ما برخوردار است ، ماکزیمم آنها است .

HOG هر عکس را که یک ماتریس است به یک آرایه و یا یک وکتور تیدیل می کند ، اما قبل از آن هر عکس را به پارت های کوچکتر تقسیم می کند و آن قسمت های کوچکتر را بررسی می کند ، همانطور که گفتیم HOG به دنبال تغییرات است بنابراین در جاهایی که زنگ پس زمینه ثابت است و تغییرات خاصی ندارد ، اندازه خیلی کوچکتری دارد ولی در گوشه ها و لبه ها ، اندازه گرادینت بسیار زیاد است و در قسما های قبل توضیح دادیم که چفدر بله ها برای ما از اهمیت ویژه ای برخوردار هستند در تشخیص اجسام .

پس تا این جای کار ما بله های عکس را پیدا کردیم پ این کار را دز کد خود با دستور آماده که در بخش کد توضیح داده می شود انجام دادیم .

SVM

Svm یا support vector machine در واقع یک الگوریتمی است که داده ها را دسته بندی می کند در یک نگاه ساده تر بین داده های دو دسته مختلف ، بهترین و بهینه ترین حالت را انتخاب می کند به صورتی که خطی را برای جدا کردن انتخاب می کند که از دادها های هر دسته بیشترین فاصله را داشته باشد .



شکل ۲ - svm

اما در حالت و یا حالت هایی که داده ی ما با یک خط جدا نمی شوند آنهارا به یک فضای دیگر می برد و در یک نگاه ساده ، یک بعد به آنها اضافه می کند و به داده های هر دسته یک مقدار می دهد و انجا کار دسته بندی را انجام می دهد و این کار را براساس یک kernel خاص انحام می دهد که به طور مصال می تواند RBF باشد .

ما در اینجای پروژه ، با کمک feature HOG توانستیم گوشه های عکس و آنجاهایی که تغییرات داریم را پیدا کنیم ، حال با استفاده ازsvm باید داده ها را دسته بندی کنیم

توجه شود که داده های ما در این مرحله به صورت یک آرایه می باشند و یا یک وکتور .

در انتهای گزارش توضیح کد مرحله train آمده است .

3.Test

پس از ران کردن کد train یک فایلی در دایرکتوری ما ساخته می شود که همان مغز prossec است ، حال باید کد قسمت تست را نیز بنویسیم .

اولین قدم در این مرحله ، capture کردن عکس از طریق وب کم می باشد که در بخش کد توضیح داده می شود .
قدم بعدی پردازش بر روی این عکس و مقایسه آن با دیتاست می باشد .

کد در فایل ارسالی پیوست شده است

توضیح بخش اول کد:

در ابتدا يك ديتا بيس از اعداد دست نويس نياز داريم
كه براي هر عدد بايد ويژي هاي hog را بدست بياوريم و توسط linear SVM جداسازي و classify كنيم
و در آخر از مدل classify شده اعداد را تخمين بزنيم.
ما از ديتا بيس Mnist استفاده كرديم كه شامل ۷۰۰۰۰ نمونه از اعداد ۰-۹ با دست خط هاي متفاوت است.
اين نمونه ها در ابعاد ۲۸X۲۸ و سياه سفيد ميباشند.
براي دانلود ديتا بيس Mnist از sklearn.datasets package اسفاده ميكنيم.
براي هر عدد حدود ۷۰۰۰ نمونه وجود دارد.
كد به دو بخش train و test تقسيم ميشود.
براي train كردن سه مرحله داريم:
۱- ويژگي هاي hog را براي هر نمونه در ديتا بيس محاسبه ميكنيم
۲- با توجه به اين ويژگي ها و الگوريتم svm , داده ها را train ميكنيم.
۳-فايل classifier را ذخيره ميكنيم.
در اين بخش ابتدا كتابخانه هاي مورد نياز را وارد ميكنيم.
skilearn.externals.joblib: براي ذخيره كردن فايل classify شده استفاده ميشود و اين امكان را ميدهد
كه هر بار نياري به train كردن دوباره وجود ندارد.
Sklearn.datasets: براي دانلود ديتا بيس mnist
Skimage.feature.hog: براي محاسبه ويژگي هاي HOG
Sklearn.svm.LinearSv: كلاس براي اينكه تخمين حاصل از train را نمايش ميدهد.
sklearn.datasets.fetch_mldata: براي دانلود ديتا ست

توضیح بخش دوم کد:

ایجاد دو عدد رندم بین 0-9

```
num1 = randrange(10)
num2 = randrange(10)
```

لود کردن classifier

```
clf = joblib.load("digits_cls.pkl")
```

دریافت تصویر از web cam

با روشن شدن وب کم و با فشار دادن کلید s **از صفحه عکس گرفته میشود و عکس سیو میشود.**

```
key = cv2.waitKey(1)
webcam = cv2.VideoCapture(cv2.CAP_DSHOW)
while True:
  try:
    check, frame = webcam.read()
    cv2.imshow("Capturing", frame)
    key = cv2.waitKey(1)
    if key == ord('s'):
      cv2.imwrite(filename='Picture.jpg', img=frame)
      webcam.release()
       cv2.waitKey(1650)
       cv2.destroyAllWindows()
```

```
        print("Image saved!")
        break
```

با فشردن کلیدِ D پنجره وب کم بسته میشد.

```
elif key == ord('q'):
        webcam.release()
        print("Camera off.")
        print("Program ended.")
        cv2.destroyAllWindows()
        break
```

خواندن تصویر از ورودی:

```
im = cv2.imread(os.getcwd() + '\Picture.jpg')
```

برای سهولت کار و همچنین بهینه تر کردن کد از نظر حافظه ای که تصویر اشغال میکند؛ تصویر را سیاه و سفید میکنیم

```
im_gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
```

برای برطرف کردن نویز های تصویر از یک فیلتر گوسی استفاده میکنیم

```
im_gray = cv2.GaussianBlur(im_gray, (5, 5), 0)
```

پس از سیاه و سفید کردن رنگ هر پیکسل عددی بین 0 و 255 میشود.برای اینکه تصویر را به حالت باینری تبدیل کنیم، حدی برای رنگ در نظر میگیریم که از آن حد بیشتر را سفید و از کمتر را سیاه درنظر میگیریم

```
ret, im_th = cv2.threshold(im_gray, 90, 255, cv2.THRESH_BINARY_INV)
```

پیدا کردن کانتور ها که مرز بین نقاط سیاه و سفید هستند با دستورات زیر صورت میگیرد

```
,_ctrs, _= cv2.findContours(im_th.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

مستطیل های زردی که هر عدد را احاطه میکنند از کد زیر بدست می آیند

```
leng = int(rect[3] * 1.5)
pt1 = int(rect[1] + rect[3] // 2 - leng // 2)
pt2 = int(rect[0] + rect[2] // 2 - leng // 2)
roi = im_th[pt1:pt1+leng, pt2:pt2+leng]
```

با توجه به اینکه دیتا های دیتا بیس به ابعاد 28*28 هستند، دیتا های ورودی نیز باید با این فرمت تطبیق داشته باشند تا قابل مقایسه باشند.در کد زیر این عمل صورت میگیرد

```
roi = cv2.resize(roi, (28, 28), interpolation=cv2.INTER_AREA)
   roi = cv2.dilate(roi, (3, 3))
```

برای هر مستطیل HOG features را بدست می آوریم و با linear SVM رقم را پیش بینی میکنیم

```
roi_hog_fd = hog(roi, orientations=9, pixels_per_cell=(14, 14), cells_per_block=(1, 1), visualize=False)
nbr = clf.predict(np.array([roi_hog_fd], 'float64'))
```

در این بخش بررسی میکینیم که اگر بیشتر از یک رقم در عکس داشتیم ، هر دو بخش را درنظر بگیریم

متغیر cnt در لوپ قرار دارد و در صورتی که بیشتر از یک عدد دریافت شود، مقدار آن بزرگتر از صفر میشود

```
 print('number is: ', nbr[0])
   if cnt == 1:
      num = num + nbr[0]
   else:
      num = num + 10*nbr[0]
   cv2.putText(im, str(int(nbr[0])), (rect[0], rect[1]),cv2.FONT_HERSHEY_DUPLEX, 2, (0, 255, 255), 3)
   cnt = cnt + 1
```

و در پایان هم جواب داده شده را با جمع دو عددی که از ابتدا به کاربر دادیم را بررسی میکنیم و صحیح یا غلط بودن خروجی را مشخص میکنیم

```
   if cnt == 1:
```

```
        num = num + nbr[0]
    else:
        num = num + 10*nbr[0]
    cv2.putText(im, str(int(nbr[0])), (rect[0], rect[1]),cv2.FONT_HERSHEY_DUPLEX, 2, (0, 255,
255), 3)
    cnt = cnt + 1
```