Homework #1

403410034 資工四 黃鈺程

Environment

只在 Fedora 26 下測試過，不過其他 Linux 環境也應該可行。

Requirement

使用 python 與其科學計算環境。

```
python3.6
numpy
scipy
scikit-learn
scikit-image
pillow
matplotlib
jupyter
tqdm
```

如果是使用 Anaconda/miniconda，可以直接執行

```
conda env create -f environment.yml
```

來創造與我相同的虛擬環境，具體細節請參官方文檔 (https://conda.io/docs/user-guide/tasks/manage-environments.html#creating-an-environment-from-an-environment-yml-file)

因為有使用 `tqdm_notebook` 所以記得[開啟]（http://ipywidgets.readthedocs.io/en/stable/user_install.html (http://ipywidgets.readthedocs.io/en/stable/user_install.html)) plugin

```
pip install ipywidgets
jupyter nbextension enable --py widgetsnbextension
```

Execution

使用 jupyter notebook 開啟 `main.ipynb` ，執行 “Cell/Run All” 即可。

Algorithms

1. histogram
2. color histogram
3. block histogram: 2x2 or 4x4
4. block statistics: likelihood (mean, variance)

這四種演算法搭配不同的參數

1. L1 distance: manhattan distance
2. L2 distance: euclidean distance
3. Intersection distance

與不同的 color space

1. RGB
2. HSV

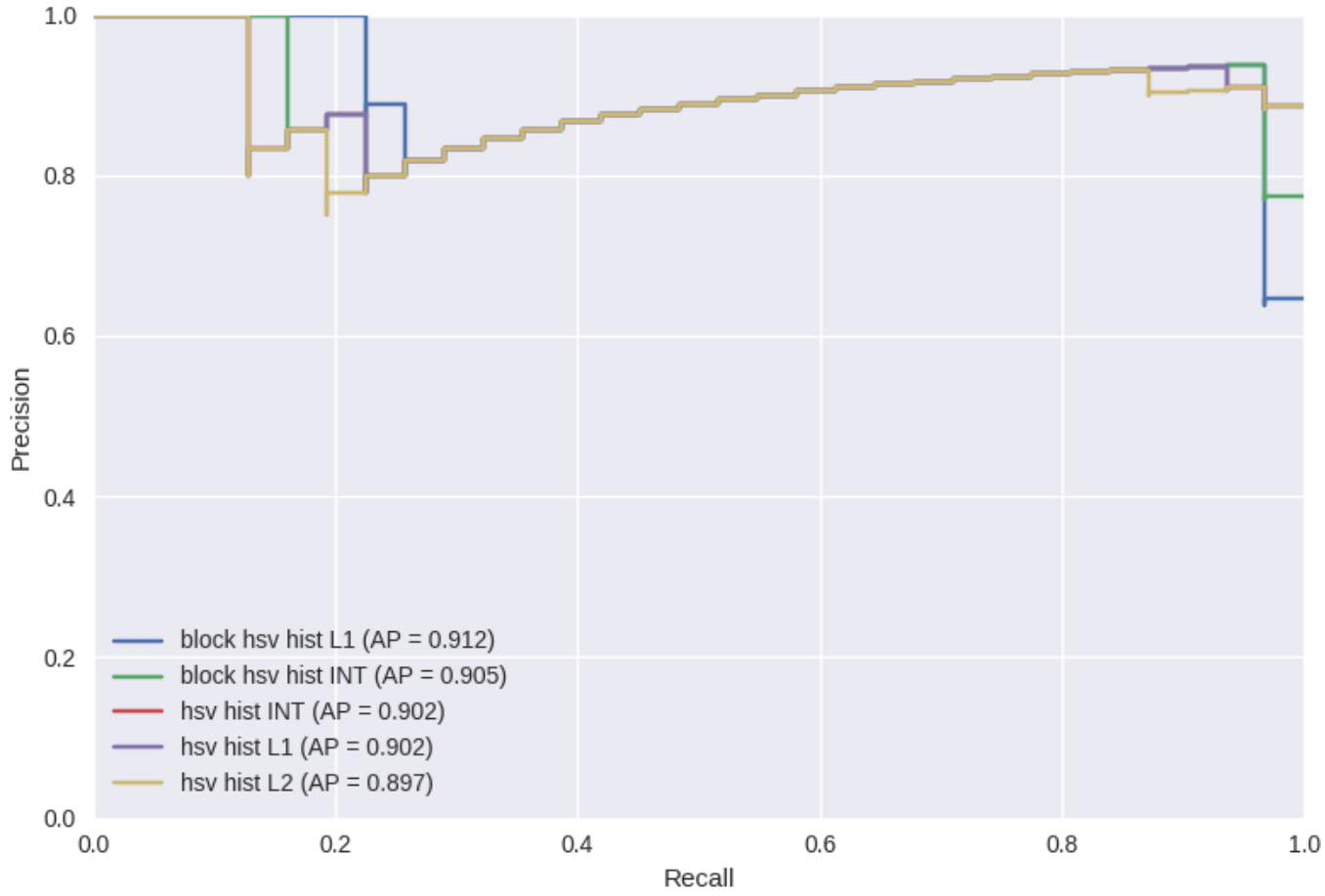
共得到 19 種演算法。
完整列表請見 `main.ipynb`

Result

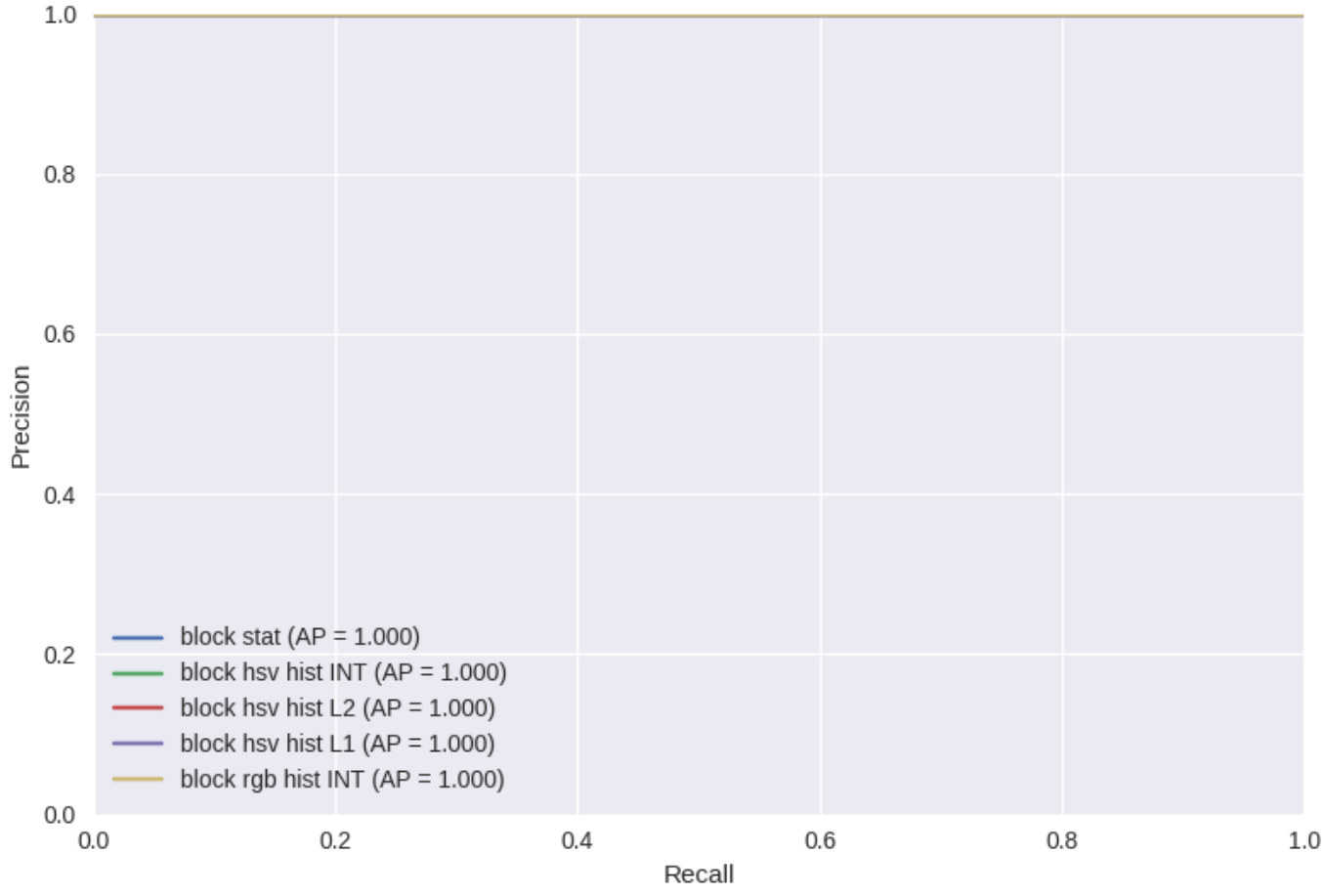
除了 PR Curve 以外，我同時計算 AP 是多少。

以下以圖表方式顯示出各影片最佳的 5 個演算法。

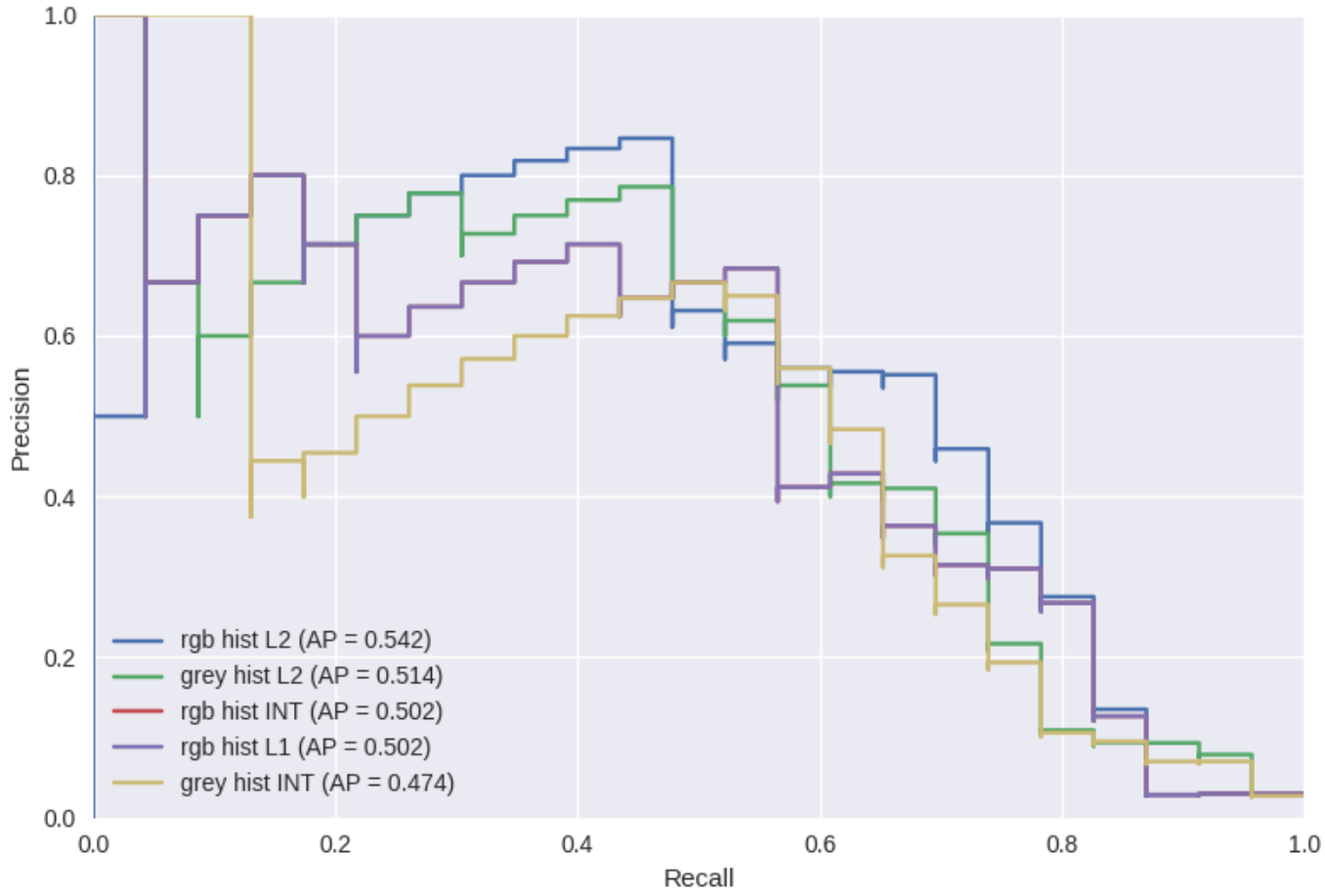
`friends.mpg`：



news.mpg：



soccer.mpg：



詳細結果為：

Friends:

name	AP
block hsv hist L1	0.912118333525
block hsv hist INT	0.904777063684
hsv hist INT	0.902055725882
hsv hist L1	0.902055725882
hsv hist L2	0.897005286525
block grey hist L1	0.884776324495
block grey hist INT	0.872007586385
rgb hist INT	0.870516637331
grey hist L2	0.864574673241
grey hist INT	0.860068061292
grey hist L1	0.849016205753
rgb hist L2	0.84719369999
rgb hist L1	0.825055388246
block rgb hist INT	0.811617326335
block stat	0.806451612903
block hsv hist L2	0.803443787582
block grey hist L2	0.754159250713
block rgb hist L1	0.438093500206
block rgb hist L2	0.310372617032

News:

name	AP
block stat	1.0
block hsv hist INT	1.0
block hsv hist L2	1.0
block hsv hist L1	1.0
block rgb hist INT	1.0
block rgb hist L1	1.0
block grey hist INT	1.0
block grey hist L2	1.0
block grey hist L1	1.0
block rgb hist L2	0.982142857143
hsv hist INT	0.968253968254
hsv hist L1	0.968253968254
rgb hist INT	0.961734693878
rgb hist L2	0.961734693878
rgb hist L1	0.961734693878
grey hist INT	0.961734693878
grey hist L1	0.961734693878
grey hist L2	0.947845804989
hsv hist L2	0.93253968254

Soccer:

name	ap
rgb hist L2	0.542220031294
grey hist L2	0.514471195978
rgb hist INT	0.502207001528
rgb hist L1	0.502207001528
grey hist INT	0.473762250746
grey hist L1	0.472756966371
hsv hist L2	0.466977489492
hsv hist INT	0.331401751637
hsv hist L1	0.331401751637
block stat	0.219441985515
block grey hist INT	0.111647295396
block rgb hist INT	0.104557584849
block grey hist L1	0.0958278980323
block grey hist L2	0.0678073896719
block rgb hist L1	0.0671979325344
block hsv hist INT	0.0563070142977
block hsv hist L1	0.0498553384751
block rgb hist L2	0.0482722030928
block hsv hist L2	0.0402684364415

從上述圖表可以發現，各個演算法有各自擅長的部份。在 Friends 中，HSV 相關的演算法有著最好的結果；在 News 中，大部份演算法都不錯；在 Soccer 中，反而是最簡單的 histogram 或 color histogram 有著較好的 AP。整體而言，似乎真的沒有哪一個演算法特別好。這應該是因為測試的資料量太小造成的。

Difficulties

Color Histogram 實作

我原先以為是三個 color channel 各自算 histogram，然後加總起來。不過在同學的告示下才發現這是錯的。
我最後使用 numpy 下的 histogramdd 來實作。

PR Curve 不好畫

研究了半天，發現自己實在畫不出 PR Curve，最後使用 scikit-learn 中的函式來畫。
一使用才發現，我原來的方法了，PR Curve 並不需要每個 threshold 時都要重跑一次演算法來得到 precision, recall，只需要跑一次，記錄下 score，在這個 score 上使用不同 threshold 來決定 TP, FP, etc 的數量，進而得到所有的 precision, recall，這樣就能畫出 PR Curve 了。想想也對，例如那些計算量大的演算法，例如深度學習的模型，不太可能每試一個 threshold 就要重新訓練一次模型，這實在太花時間了。

顏色不好調

matplotlib 預設的顏色太醜啦，我以前都透過 `import seaborn` 來使用 seaborn 的配色來解決。
不過這次仔細翻了一下文檔，發現 matplotlib 已經內建許多配色了，包含 seaborn 的配色。
例如，可以使用

```
import matplotlib.pyplot as plt
plt.style.use('seaborn')
```

來使用 seaborn 的配色。

ffmpeg

我本來想使用我之前寫的，將影片轉成 frame 的程式。但發現轉出來的 frame 的數量竟然跟老師提供的不一樣，多了兩張圖片。後來研究了一下才發現是因為老師固定使用 fps = 30，而我的程式是使用比較精確的值，fps 精度到達小數點第一位。所以為了一致，我最後還是使用老師提供的 frame。

Intersection Distance

我 intersection distance 的實現為：

```
INT = lambda h1, h2: np.sum(np.minimum(h1, h2)) / np.sum(h2)
```

這是對的，但你拿這個去跑演算法會錯，因為我的演算法中要的是 dissimilarity 而不是 similarity。
所以正確的寫法是：

```
INT = lambda h1, h2: 1 - np.sum(np.minimum(h1, h2)) / np.sum(h2)
```

找這個 bug 花去我不少的時間。

另外，跟 intersection distance 比較相關的是 Block + Color Space 那幾個演算法，那幾個演算法一定得用 intersection distance 的樣子，用 L1, L2 都跑出極差的結果。

感想

不錯的一個作業，可惜我估錯我的實作能力，比預期花了更多的時間才做出這些結果。我本來還想來試試 hog, sift, lbp 這三個最近在 CV 學到的 local feature 表現如何，會不會比目前多媒體學到的這些 global feature 好。