# 图像压缩编码与DCT（C15）

胡俊峰 2022/04/27
北京大学计算机学院

# 关于超人组大作业：（邮件提交：hujf@pku.edu.cn 1份A就行）

- 生成一幅图片，对用户实现观影兴趣画像
- 要求：画面符合直觉，内容表达用户特点

单人组：
- 第一步：用户特点挖掘（单人组作业的内容）
- 第二步：结合电影海报信息，生成一幅图片，表达用户观影兴趣

双人组：
- 第一步：用户特点挖掘（单人组作业的内容）
- 第二步：结合电影海报信息，分析用户观影图像特征偏好（建议）
- 第三步：生成一幅图片，表达用户观影兴趣

三人组：
- 第一步：用户特点挖掘（单人组作业的内容）
- 第二步：结合电影海报信息，分析用户观影图像特征偏好，优化用户画像（双人组作业2）
- 第三步：生成一幅图片，表达用户观影兴趣

- 多人组作业提交：文件名：学号1-学号2_A.zip；学号1-学号2_B.zip…
- 学号1-学号2_A.zip中需要包含全部环境-代码-报告。除包含综合运行环境外，每人单独分出子目录。
- 代码实现、作业报告（含方案、效果及评测、结果分析）要独立。

# 电影聚类及用户画像（by 李一飞助教）

- 读取数据
- 数据预处理：
  - 筛选出观影数量大于100的用户信息，共2908个用户（这里可以用所有用户，画像>100）
  - 筛选出被这些用户观看过的电影，共3601部电影
  - 删除电影矩阵中未出现在ratings矩阵里的电影id对应的行
  - 构建用户-电影评分矩阵

```
Movies table:
   movie_id                        genres                               title  \
0         1   Animation|Children's|Comedy                    Toy Story (1995)
1         2  Adventure|Children's|Fantasy                      Jumanji (1995)
2         3                Comedy|Romance             Grumpier Old Men (1995)
3         4                  Comedy|Drama            Waiting to Exhale (1995)
4         5                        Comedy  Father of the Bride Part II (1995)

                                               intro        directors  \
0  A cowboy doll is profoundly threatened and jea...    John Lasseter
1  When two kids find and play a magical board ga...     Joe Johnston
2  John and Max resolve to save their beloved bai...    Howard Deutch
3  Based on Terry McMillan's novel, this film fol...  Forest Whitaker
4  George Banks must deal not only with the pregn...    Charles Shyer

                                          stars
0                Tom Hanks|Tim Allen|Don Rickles
1       Robin Williams|Kirsten Dunst|Bonnie Hunt
2          Walter Matthau|Jack Lemmon|Ann-Margret
3  Whitney Houston|Angela Bassett|Loretta Devine
4          Steve Martin|Diane Keaton|Martin Short
Movies shape: (3601, 6)

User-Movie Matrix shape: (2908, 3601)
```

# 特征提取：

- User-movie矩阵降维及隐含话题语义空间挖掘
  - svds(user_movie_matrix, k=50)  # k为提取的隐含语义向量的数量
- TF-IDF特征：
  - # 初始化TF-IDF向量器
  - vectorizer = TfidfVectorizer(max_features=100, stop_words="english")
  - # 计算TF-IDF值
  - tfidf_matrix = vectorizer.fit_transform(movies["intro"].values.astype('U'))

# TF-IDF空间（100维）

```python

# 结果转换为DataFrame
text_semantic_vectors = pd.DataFrame(tfidf_matrix.toarray(), index=movies["movie_id"])
text_semantic_vectors
```

| movie_id | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.775515 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 |
| **2** | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 |
| **3** | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 |
| **4** | 0.0 | 0.523075 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.000000 | 0.0 | 0.620963 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 |
| **5** | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.671198 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **3948** | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 |
| **3949** | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 |
| **3950** | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.368253 |
| **3951** | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.577998 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 |
| **3952** | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 |

# 已标注特征处理方式：（one-hot）

```
# 使用pandas的get_dummies方法进行独热编码
occupation_dummies = pd.get_dummies(users["occ_desc"], prefix="occupation")

# 将独热编码的特征添加到用户数据表中，并删除原始职业特征
users_temp = pd.concat([users.drop(columns=["occ_desc"]), occupation_dummies], axis=1)

```

```
users_temp
```

| | user_id | gender | zipcode | age_desc | movie_count | occupation_K-12 student | occupation_academic/educator | occupation_artist | occupation_clerical/admin | oc |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 3 | 1 | 55117 | 25-34 | 129.0 | 0 | 0 | 0 | 0 | |
| **1** | 6 | 0 | 55117 | 50-55 | 198.0 | 0 | 0 | 0 | 0 | |
| **2** | 9 | 1 | 61614 | 25-34 | 139.0 | 0 | 0 | 0 | 0 | |
| **3** | 10 | 0 | 95370 | 35-44 | 106.0 | 0 | 1 | 0 | 0 | |
| **4** | 11 | 0 | 04093 | 25-34 | 401.0 | 0 | 1 | 0 | 0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **2903** | 6033 | 1 | 78232 | 50-55 | 104.0 | 0 | 0 | 0 | 0 | |

# 特征的中心化-标准化处理

```python
# 对连续型特征进行标准化
from sklearn.preprocessing import MinMaxScaler, StandardScaler
scaler = MinMaxScaler()
audience_features_scaled = scaler.fit_transform(audience_features_occu)

# 转换为DataFrame
audience_feature_vectors = pd.DataFrame(audience_features_scaled, index=audience_features_occu.index)
audience_feature_vectors
```

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **movie_id** | | | | | | | | | | | | | |
| **1** | 0.052361 | 0.067762 | 0.039014 | 0.031828 | 0.132444 | 0.032854 | 0.034908 | 0.104723 | 0.002053 | 0.046201 | ... | 0.129363 | 0.069815 |
| **2** | 0.047059 | 0.079412 | 0.050000 | 0.032353 | 0.129412 | 0.047059 | 0.041176 | 0.111765 | 0.008824 | 0.022059 | ... | 0.147059 | 0.041176 |
| **3** | 0.025316 | 0.059072 | 0.037975 | 0.037975 | 0.160338 | 0.033755 | 0.050633 | 0.101266 | 0.004219 | 0.042194 | ... | 0.147679 | 0.054852 |
| **4** | 0.022472 | 0.089888 | 0.033708 | 0.044944 | 0.213483 | 0.022472 | 0.067416 | 0.134831 | 0.000000 | 0.000000 | ... | 0.146067 | 0.011236 |
| **5** | 0.060150 | 0.060150 | 0.030075 | 0.052632 | 0.165414 | 0.060150 | 0.030075 | 0.097744 | 0.007519 | 0.056391 | ... | 0.157895 | 0.030075 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

# 对电影进行聚类

```
movies
```

| | movie_id | genres | title | intro | directors | stars | cluster |
|---|---|---|---|---|---|---|---|
| **0** | 1 | Animation\|Children's\|Comedy | Toy Story (1995) | A cowboy doll is profoundly threatened and jea... | John Lasseter | Tom Hanks\|Tim Allen\|Don Rickles | 2 |
| **1** | 2 | Adventure\|Children's\|Fantasy | Jumanji (1995) | When two kids find and play a magical board ga... | Joe Johnston | Robin Williams\|Kirsten Dunst\|Bonnie Hunt | 2 |
| **2** | 3 | Comedy\|Romance | Grumpier Old Men (1995) | John and Max resolve to save their beloved bai... | Howard Deutch | Walter Matthau\|Jack Lemmon\|Ann-Margret | 2 |
| **3** | 4 | Comedy\|Drama | Waiting to Exhale (1995) | Based on Terry McMillan's novel, this film fol... | Forest Whitaker | Whitney Houston\|Angela Bassett\|Loretta Devine | 0 |
| **4** | 5 | Comedy | Father of the Bride Part II (1995) | George Banks must deal not only with the pregn... | Charles Shyer | Steve Martin\|Diane Keaton\|Martin Short | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... |

# 对每个类进行一下数据分析

- 对应传统的类标签情况？
- 类型观众分布？
- 进行一些定性的理解与描述？

```
]: 1 cluster_audience_features
```

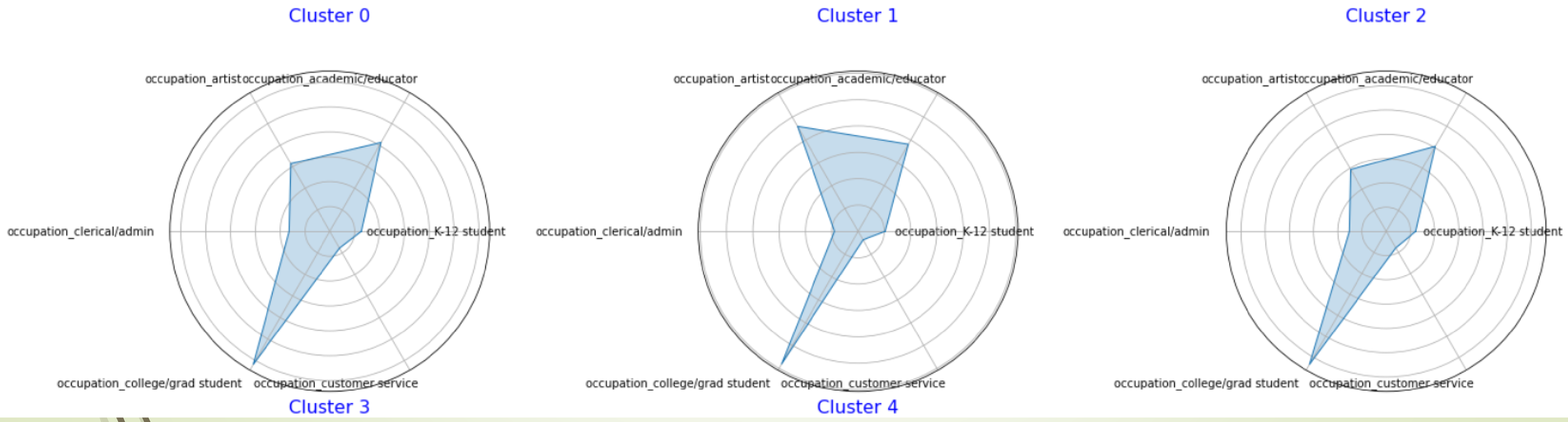
]:

| cluster | occupation_K-12 student | occupation_academic/educator | occupation_artist | occupation_clerical/admin | occupation_college/grad student | occupation_customer service | occupation_dc |
|---|---|---|---|---|---|---|---|
| **0.0** | 0.025324 | 0.082655 | 0.062899 | 0.032796 | 0.123468 | 0.015577 | |
| **1.0** | 0.020665 | 0.076382 | 0.092002 | 0.017906 | 0.116145 | 0.007500 | |
| **2.0** | 0.024454 | 0.081038 | 0.058835 | 0.030517 | 0.126563 | 0.015723 | |
| **3.0** | 0.023709 | 0.084548 | 0.054021 | 0.030483 | 0.122956 | 0.013055 | |
| **4.0** | 0.027239 | 0.072425 | 0.049660 | 0.031073 | 0.125689 | 0.019764 | |

5 rows × 21 columns

```python
for i, row in enumerate(cluster_audience_features.iterrows()):
    plot_radar_chart(axes[i], row[1][feature_labels].values[:6], f"Cluster {i}")

fig.tight_layout(pad=1.0)  # 增加子图之间的间距
plt.show()
```

```
['occupation_K-12 student', 'occupation_academic/educator', 'occupation_artist', 'occupation_clerical/admin', 'occupation_college/grad student', 'occupation_customer service']
6
[0.0, 1.0471975511965976, 2.0943951023931953, 3.141592653589793, 4.1887902047863905, 5.235987755982989, 0.0]
7
```



# 用户分析：

## 3. 用户画像部分

### 统计用户特征

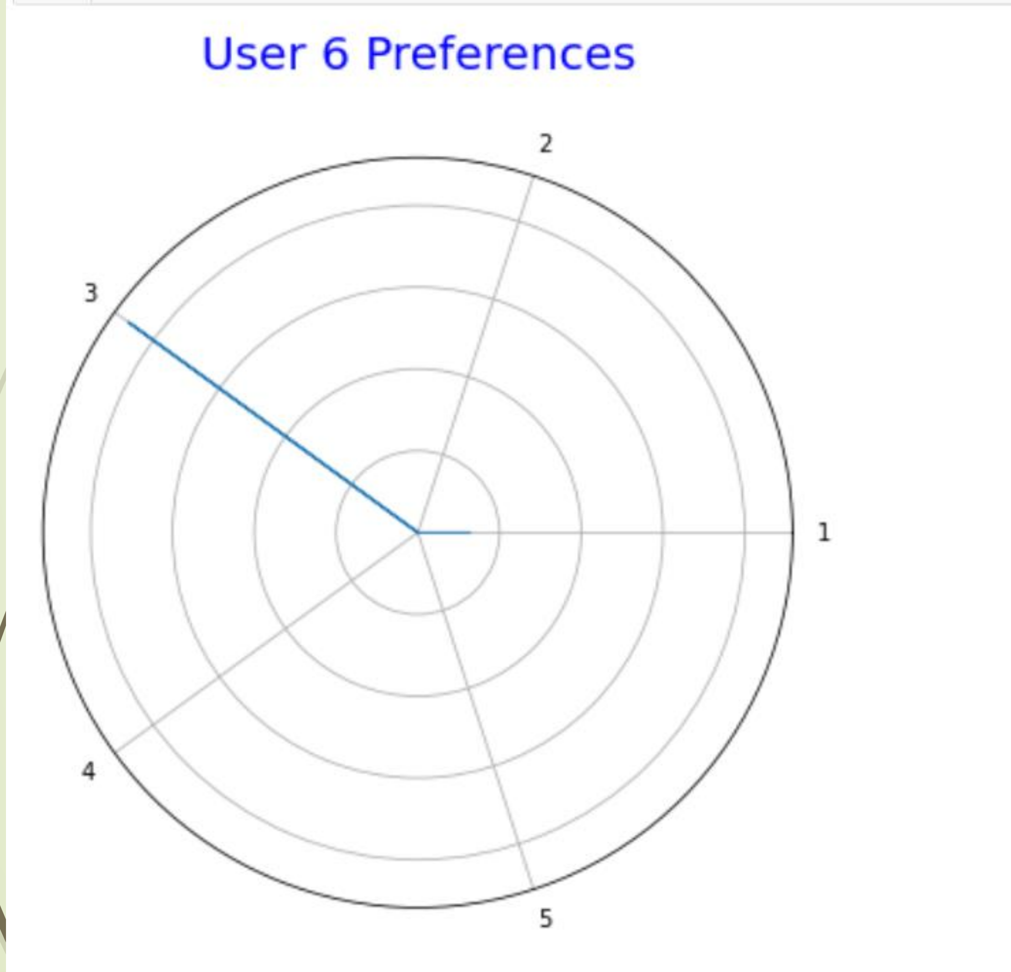
In [22]:

```
# 获取每个用户观看过的电影的聚类标签
user_cluster_counts = ratings.merge(movies[['movie_id', 'cluster']], on='movie_id').groupby(['user_i

# 找到每个用户最喜欢和最不喜欢的电影聚类
user_fav_cluster = user_cluster_counts.loc[user_cluster_counts.groupby('user_id')['count'].idxmax()]
user_least_fav_cluster = user_cluster_counts.loc[user_cluster_counts.groupby('user_id')['count'].id

users['fav_cluster'] = user_fav_cluster
users['least_fav_cluster'] = user_least_fav_cluster

```

In [23]:

```
user_cluster_distribution = user_cluster_counts.pivot(index='user_id', columns='cluster', values='c
user_cluster_distribution_normalized = user_cluster_distribution.div(user_cluster_distribution.sum(a
```

# 生成用户观影偏好雷达图、词云




# 该项目评分：

- 代码风格清晰（有关键注释）
- 报告内容完整（包含对数据方案，流程，结果的阐述分析）
- 没有其他明显的缺陷
- 90+

- 提取新的概念并通过数据分析观察进行论证：
- 95+

# 图像向量空间特征聚类（by朱成轩助教）

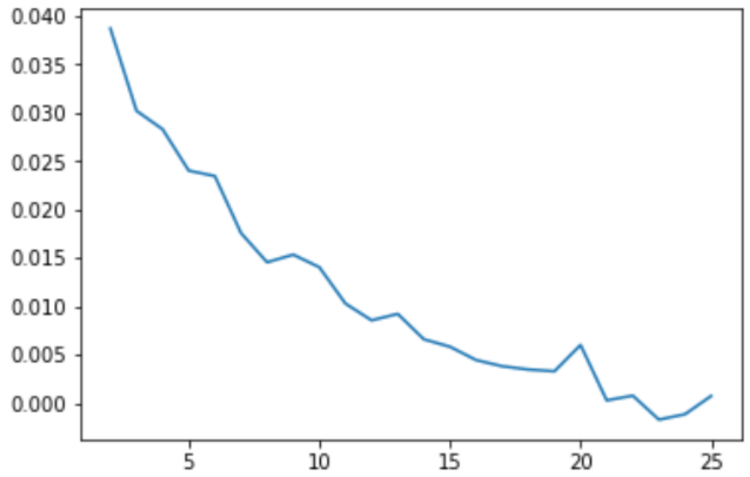
- 可以直接使用：

```python
# from img2vec_pytorch import Img2Vec
# img2vec = Img2Vec(cuda=False)  # 装入已经预训练好的模型
# img_embed = img2vec.get_vec(imgs).squeeze()  # 得到图片的嵌入向量 1min

#img_embed.shape
#np.save("data/poster_embed",img_embed ) #保存数组

img_embed = np.load('data/poster_embed.npy')
img_embed.shape
```

```
(2938, 512)
```

- 多次尝试找到相对好的聚类结果
- 每次随机种子要设定为不一样

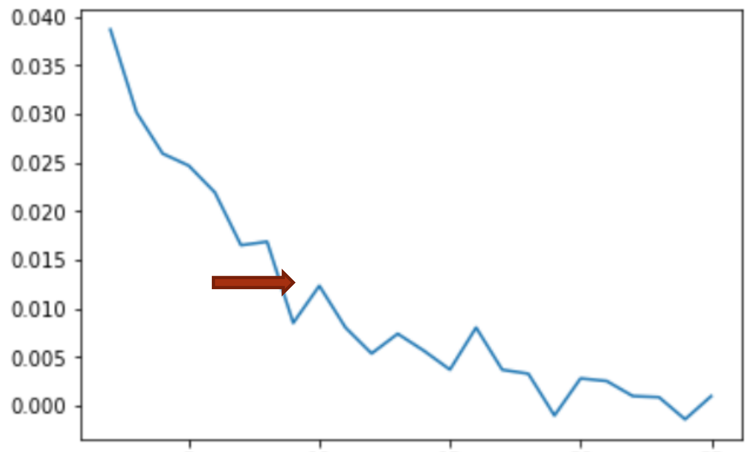
Out[5]: [<matplotlib.lines.Line2D at 0x220f42c95b0>]



In [6]:

```python
sc = []
for N_CL in range(2, 26):  # 尝试2-26个类质心,
    clf = KMeans(n_clusters = N_CL, random_state=2).fit(img_embed)
    labels = clf.labels_
    eval_score = metrics.silhouette_score(img_embed, labels, metric='eucli
    sc.append(eval_score)

plt.plot(np.arange(2,26),np.array(sc))  # 画出类属相关的loss值
```

Out[6]: [<matplotlib.lines.Line2D at 0x220f435ff70>]

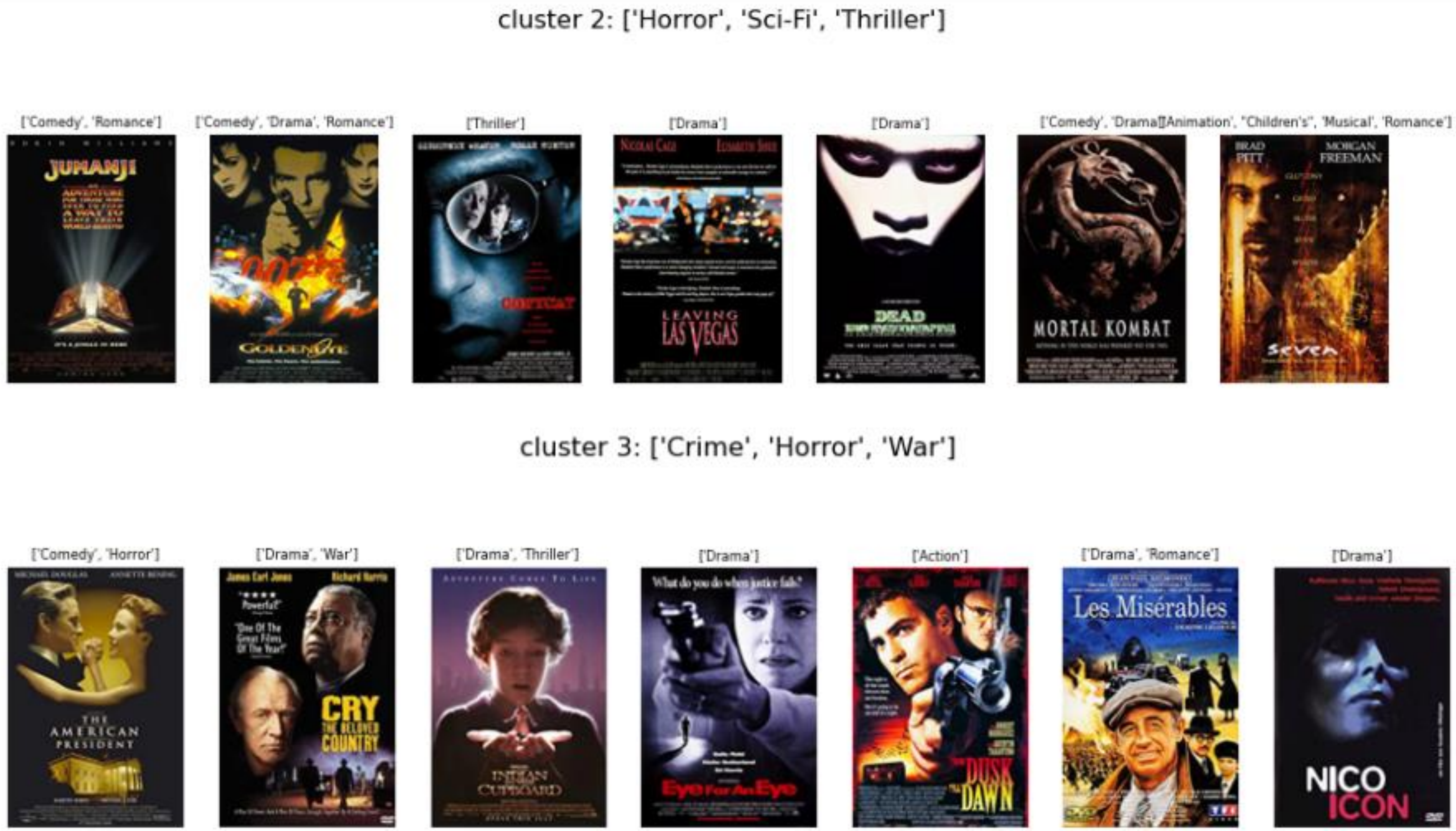


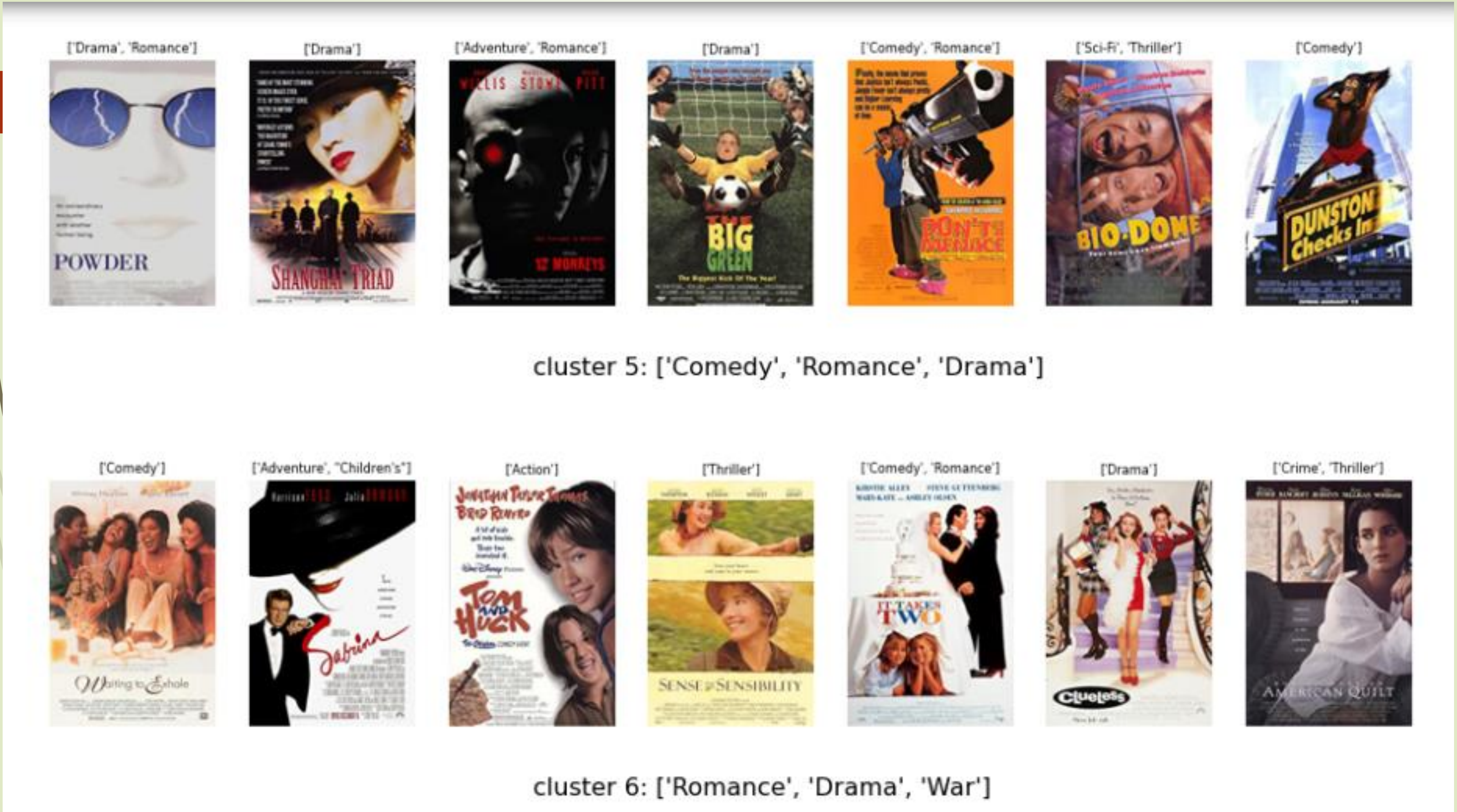
# 对不同类的海报观察分析：

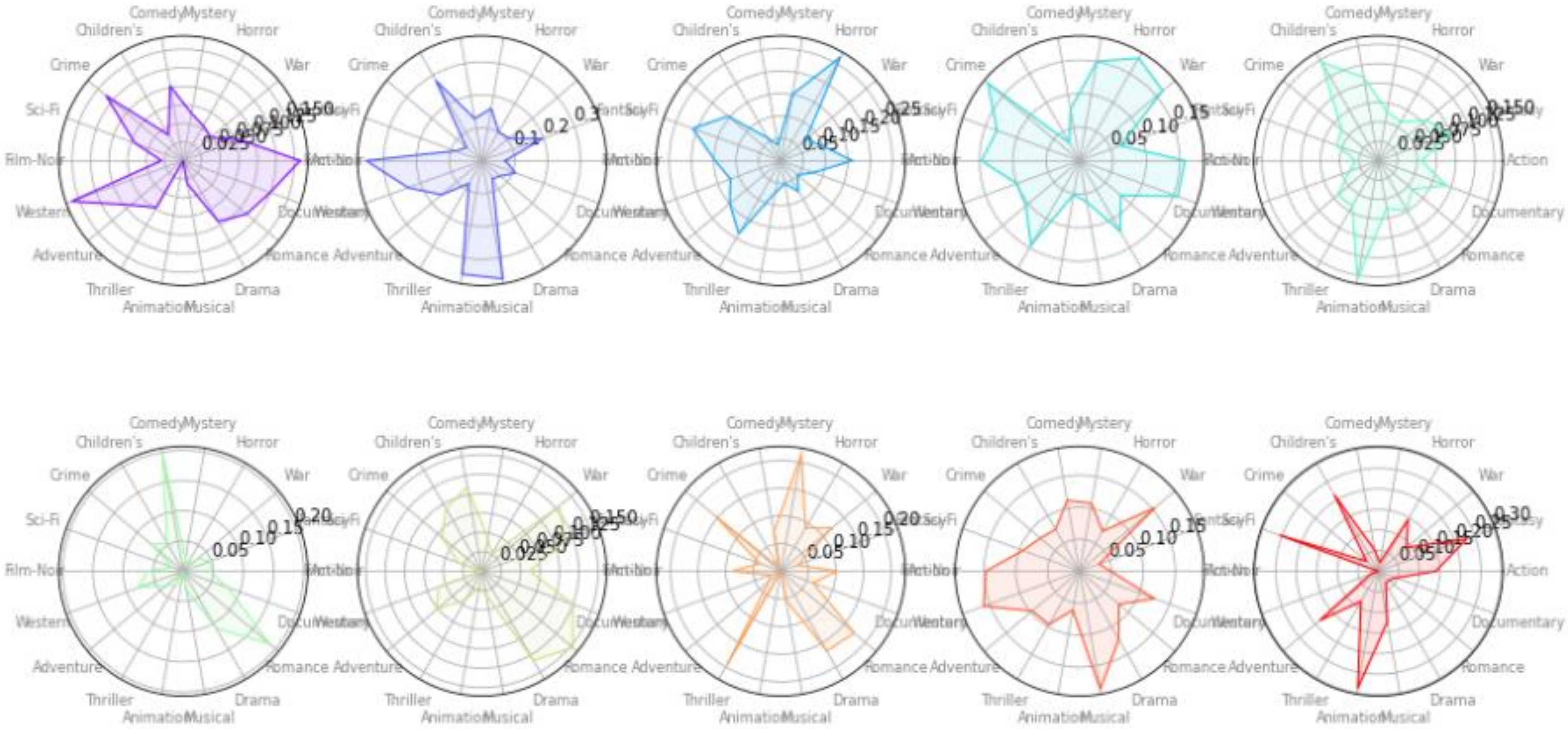
- 年龄偏好？
- 职业偏好？
- 与标注类型对比分析？

```
cluster_genres_distrib = []  # 各个聚类的电影类型分布?

for INDEX in range(10):
    ind = clf.predict(img_embed) == INDEX  # 返回第i个类为True的布尔下标

    imgs = np.array(imgs,dtype=object)
    example_images = imgs[ind] # 只有numpy数组才能正常用布尔下标
    img_id = np.array(img_ids)[ind]
    genres_cls = np.array(gen_labels)[ind]
    NUMS = 7
    fig = plt.figure(figsize=(16,7))
    for i in range(NUMS):
        plt.subplot(1,NUMS,i+1);plt.axis('off')
        plt.imshow(example_images[i])
        plt.title( data[img_id[i]][-1], size = 8) # 显示电影类型

    perc = np.sum(genres_cls,axis=0)/np.sum(gen_labels,axis=0)  # 得到每
    cluster_genres_distrib.append(perc)

    fig.text(0.4,0.8,f'cluster {INDEX}: {[genres[k] for k in np.argsort(-
    plt.show()
```


cluster 0: ['Western', 'Action', 'Crime']
['Animation', "Children's"]
LESLIE NIELSEN
DRACULA DEAD and LOVING IT
['Drama', 'Thriller']
['Comedy']
FOUR ROOMS
['Action', 'Comedy', 'Drama']
MONEY TRAIN
['Crime', 'Drama', 'Thriller']
GET SHORTY
['Adventure', 'Sci-Fi']
PERSUASION
['Drama', 'Sci-Fi']
cluster 1: ['Musical', 'Film-Noir', 'Animation']
['Adventure', "Children's", 'Fantasy']
TOY STORY
['Crime', 'Drama', 'Romance']
CAMERA D'OR
The White Balloon
['Comedy']
['Documentary']
['Drama']
UMBRELLAS of CHERBOURG
['Drama', 'Musical']
GOOFY MOVIE
['Adventure', "Children's", 'Comedy', 'Fantasy', 'Romance']
JERKY BOYS
cluster 2: ['Horror', 'Sci-Fi', 'Thriller']


cluster 2: ['Horror', 'Sci-Fi', 'Thriller']
['Comedy', 'Romance']
['Comedy', 'Drama', 'Romance']
['Thriller']
['Drama']
['Drama']
['Comedy', 'Drama'][Animation', "Children's", 'Musical', 'Romance']
JUMANJI
GOLDENEYE
LEAVING LAS VEGAS
MORTAL KOMBAT
BRAD PITT
MORGAN FREEMAN
SEVEN
cluster 3: ['Crime', 'Horror', 'War']
['Comedy', 'Horror']
['Drama', 'War']
['Drama', 'Thriller']
['Drama']
['Action']
['Drama', 'Romance']
['Drama']
THE AMERICAN PRESIDENT
CRY THE BELOVED COUNTRY
INDIAN CUPBOARD
What do you do when justice fails?
Eye For An Eye
DUSK DAWN
Les Misérables
NICO ICON


['Drama', 'Romance']
['Drama']
['Adventure', 'Romance']
['Drama']
['Comedy', 'Romance']
['Sci-Fi', 'Thriller']
['Comedy']
cluster 5: ['Comedy', 'Romance', 'Drama']
['Comedy']
['Adventure', "Children's"]
['Action']
['Thriller']
['Comedy', 'Romance']
['Drama']
['Crime', 'Thriller']
cluster 6: ['Romance', 'Drama', 'War']



Comedy
Mystery
Children's
Horror
Crime
War
Sci-Fi
Fantasy
Film-Noir
Action
Western
Documentary
Adventure
Romance
Thriller
Drama
Animation
Musical

# 图像色彩聚类（by 陈福康助教）

```python
def image_colorfulness(img):                #假设输入是RGB图像
    R,G,B = cv2.split(img.astype('float'))

    rg = np.absolute(R - G)
    yb = np.absolute(0.5 * (R + G) - B)

    (rgMean, rgStd) = (np.mean(rg),np.std(rg))
    (ybMean, ybStd) = (np.mean(yb),np.std(yb))

    stdRoot = np.sqrt(rgStd ** 2 + ybStd ** 2)
    meanRoot = np.sqrt(rgMean ** 2 + ybMean ** 2)

    return stdRoot + 0.3 * meanRoot
```

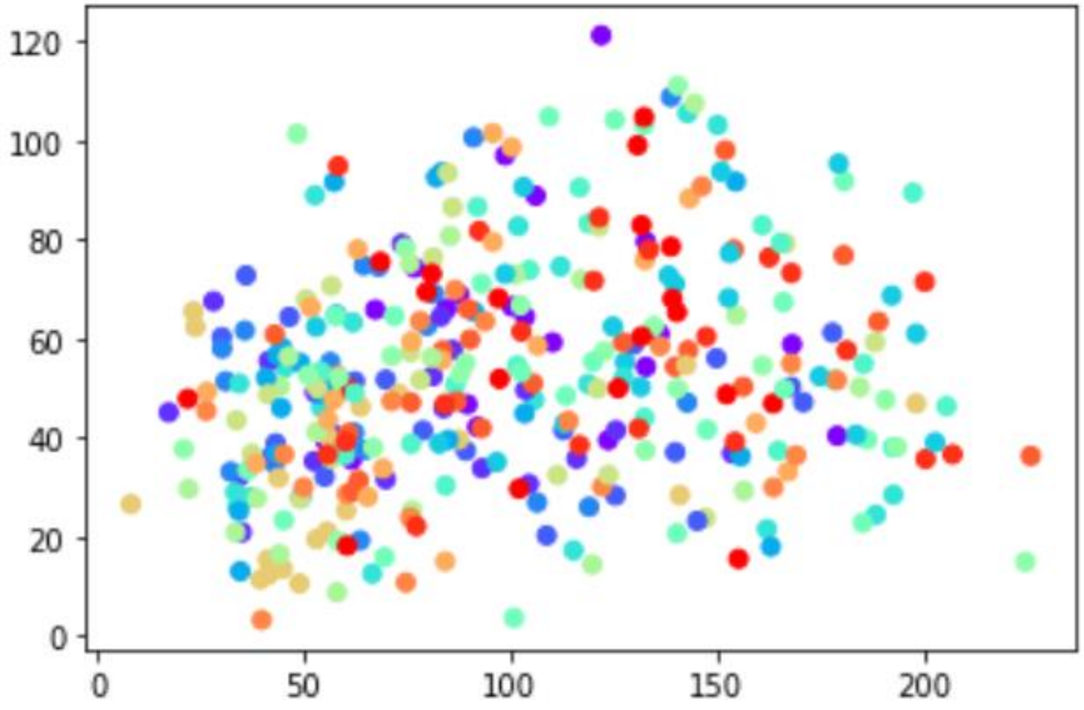
$$\begin{aligned} rg &= R - G \\ yb &= \frac{1}{2}(R+G) - B \end{aligned}$$

$$\begin{aligned} \hat{M}^{(3)} &= \sigma_{rgyb} + 0.3 \cdot \mu_{rgyb}, \\ \sigma_{rgyb} &:= \sqrt{\sigma_{rg}^2 + \sigma_{yb}^2}, \\ \mu_{rgyb} &:= \sqrt{\mu_{rg}^2 + \mu_{yb}^2}, \end{aligned}$$

# 直接观察分布

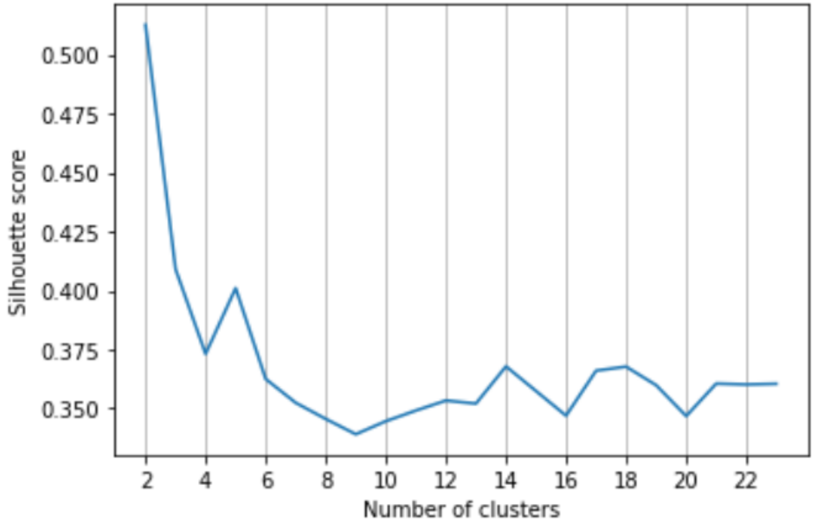
```
plt.scatter(features[:,0],features[:,1],c = colors[labels]) # 显示类
```

<matplotlib.collections.PathCollection at 0x22095be6b20>



# 聚类及优化分析：

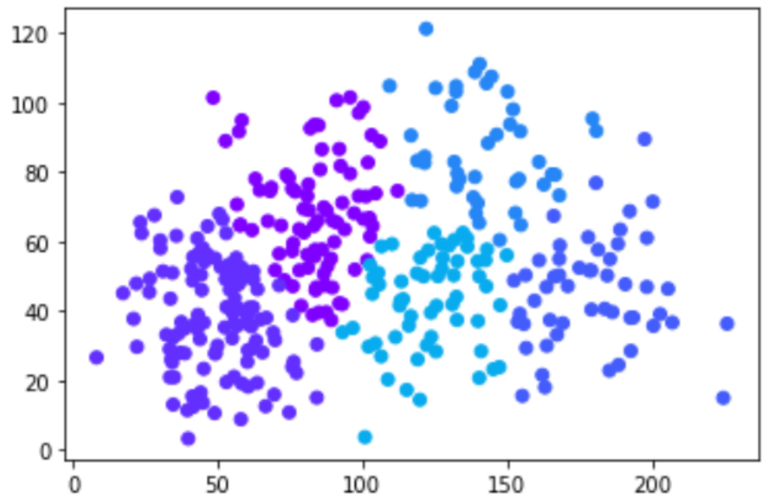
Out[90]: [<matplotlib.lines.Line2D at 0x22095e723a0>]




In [91]:

```
clf = KMeans(n_clusters=5,random_state=42)
pred = clf.fit_predict(features)

plt.scatter(features[:,0],features[:,1],c = colors[pred])
```

Out[91]: <matplotlib.collections.PathCollection at 0x22095ed0820>



# Image and Parameter Spaces

Equation of Line: $y = mx + c$

Find: $(m, c)$

Consider point: $(x_i, y_i)$

$$y_i = mx_i + c \quad or \quad c = -x_i m + y_i$$

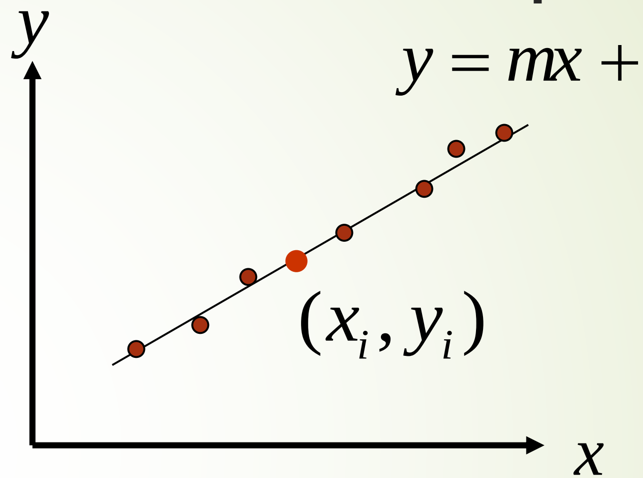
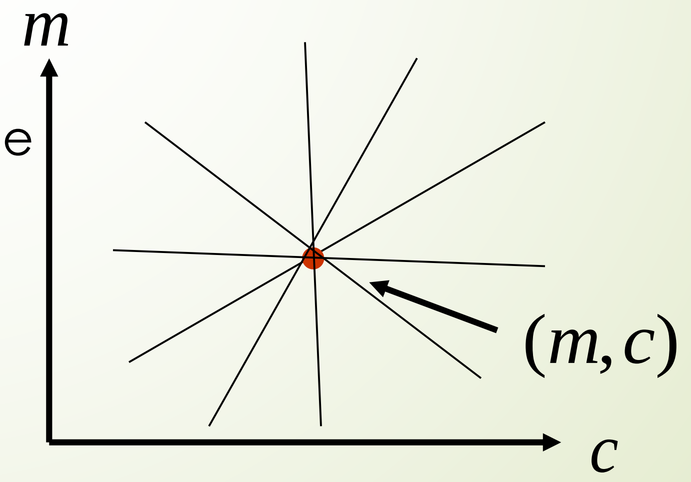



Image Space

Parameter space also called Hough Space




Parameter Space

以下英文内容ppt来自： S. Narasimhan

# 图像的DCT变换与压缩编码

- 频域变换
- 降低高频分量的分辨度
- 降阶编码-压缩

# Quantization

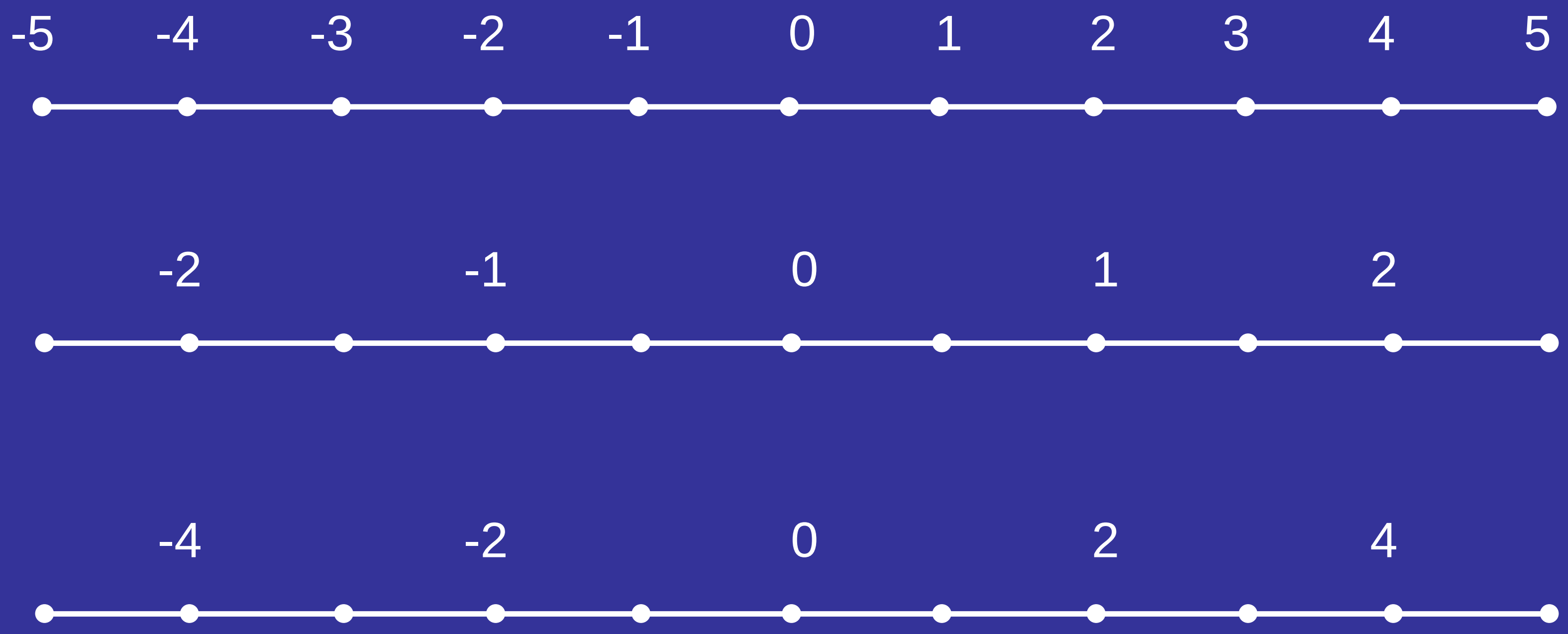
- Quantization is the process of approximating a continuous (or range of values) by a (much) smaller range of values

$$Q(x,\Delta) = \text{Round}\left(\frac{x+0.5}{\Delta}\right)$$

- Where Round(y) rounds y to the nearest integer
- $\Delta$ is the quantization stepsize

# Quantization

- Example: Δ=2



# Quantization

- Quantization plays an important role in lossy compression
  - This is where the loss happens

# Fundamentals of images

- Here is an image represented with 8-bits per pixel

# Fundamentals of images

- Here is the same image at 7-bits per pixel

# Fundamentals of images

- And at 6-bits per pixel

# Fundamentals of images

- And at 5-bits per pixel

# Fundamentals of images

- And at 4-bits per pixel

# Fundamentals of images

- Do we need all these bits?
  - No!
- The previous example illustrated the eye's sensitivity to luminance
- We can build a perceptual model
  - Only code what is important to the human visual system (HVS)
    - Usually a function of spatial frequency

# Fundamentals of Images

- Just as audio has temporal frequencies
- Images have spatial frequencies
- Transforms
  - Fourier transform
  - Discrete cosine transform
  - Wavelet transform
  - Hadamard transform

# Discrete cosine transform

- Forward DCT

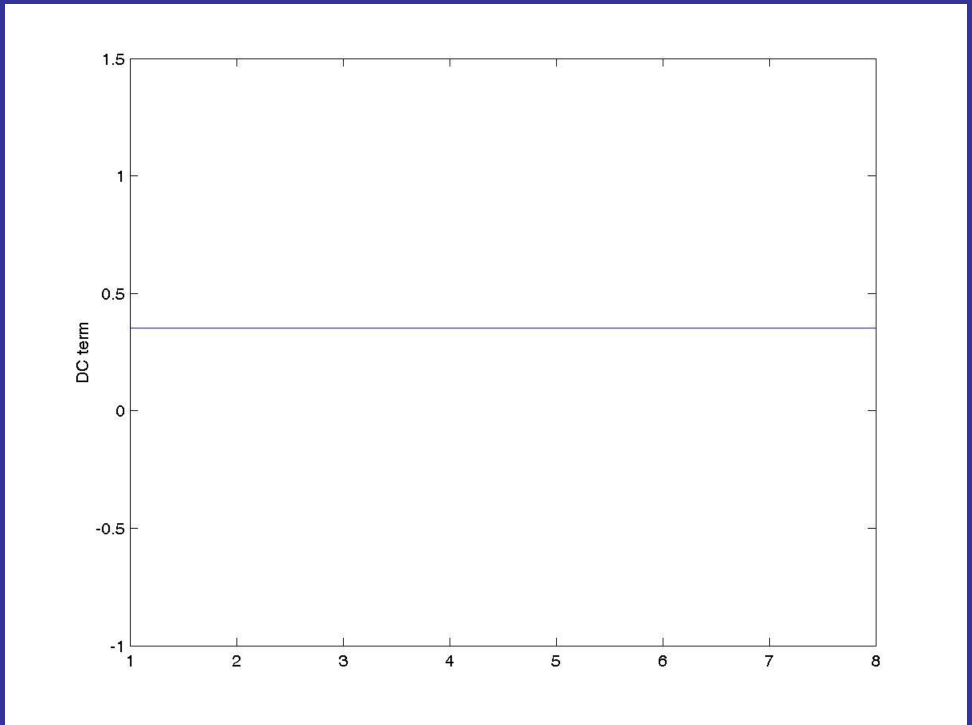
$$S(u) = \frac{C(u)}{2} \sum_{n=0}^{N-1} s(n) \cos\left( \frac{u\pi}{8} (n + 0.5) \right)$$

- Inverse DCT

$$s(n) = \frac{C(u)}{2} \sum_{u=0}^{N-1} S(u) \cos\left( \frac{u\pi}{8} (n + 0.5) \right)$$

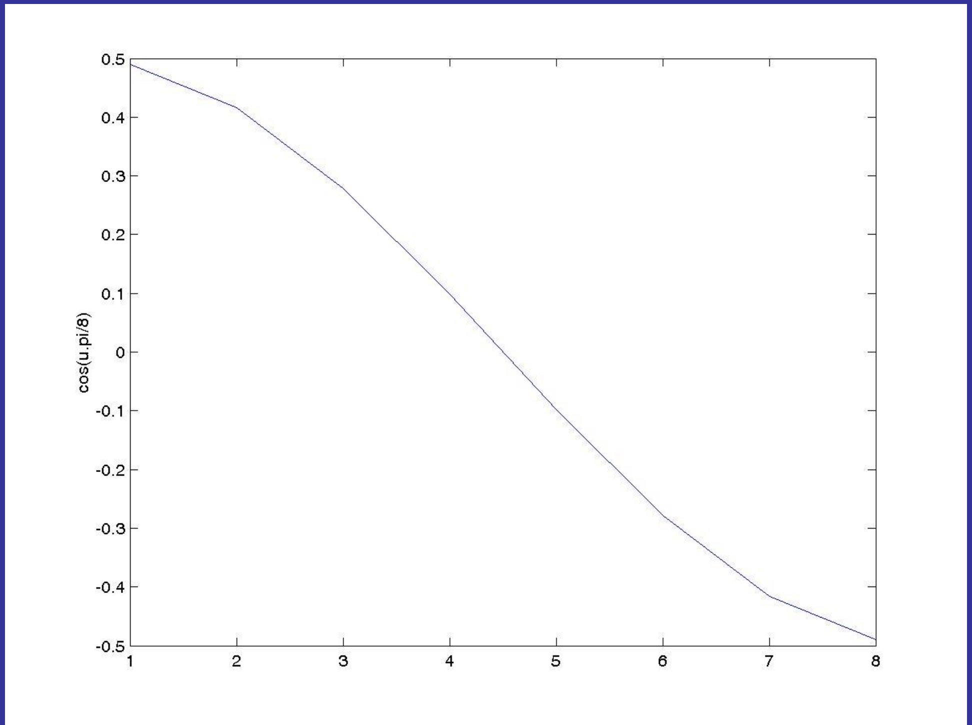
# Basis functions

- DC term



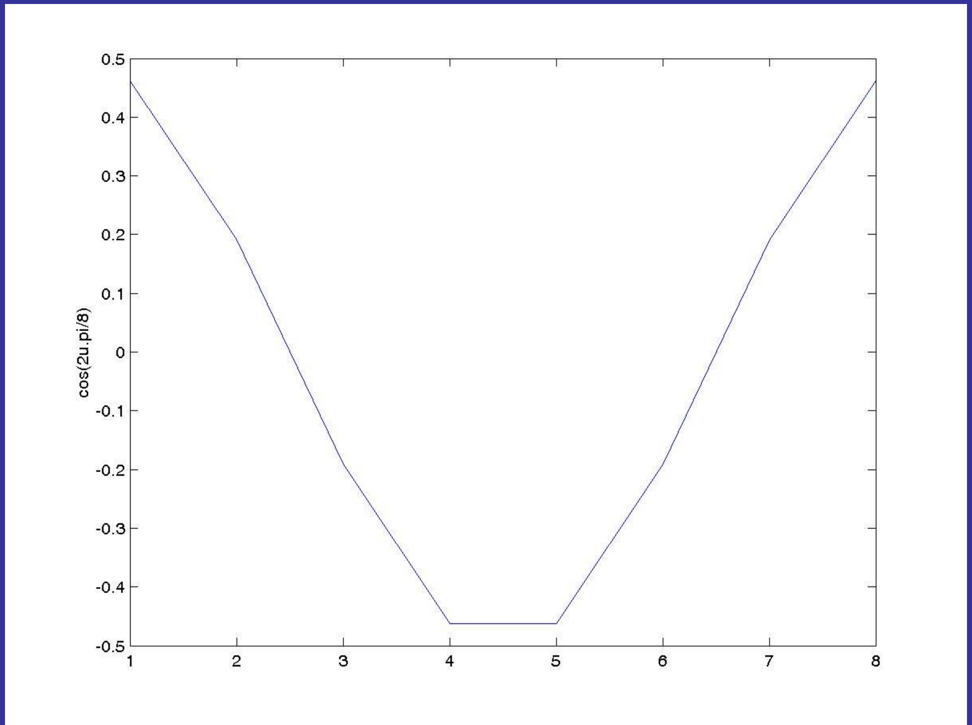
# Basis functions

- First term



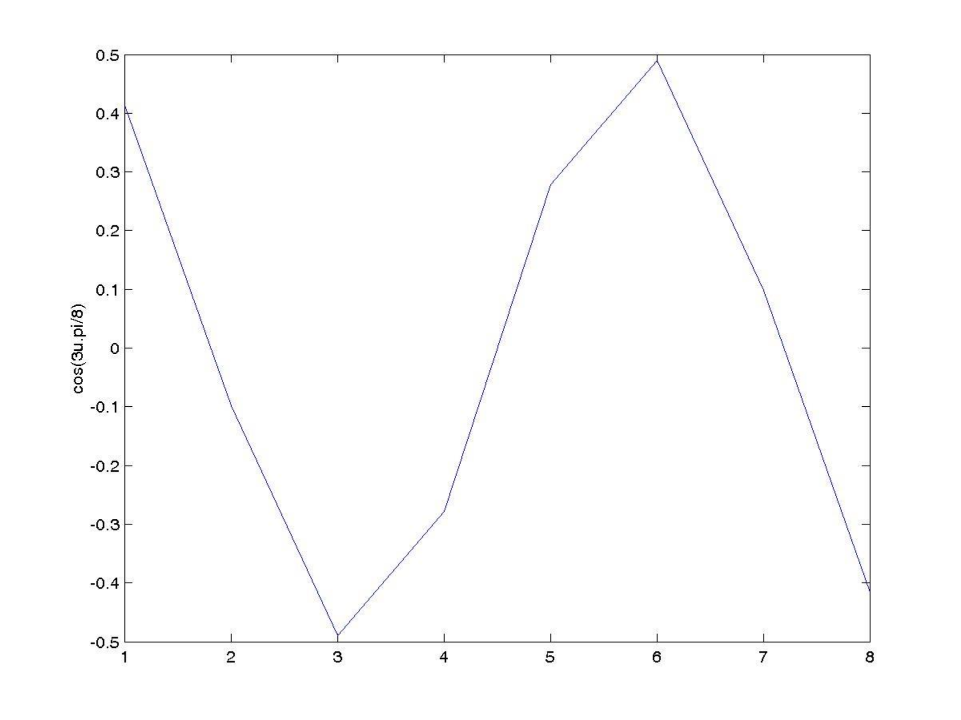
# Basis functions

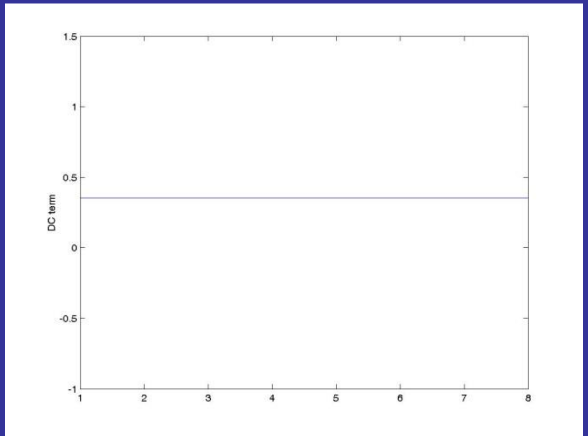
- Second term

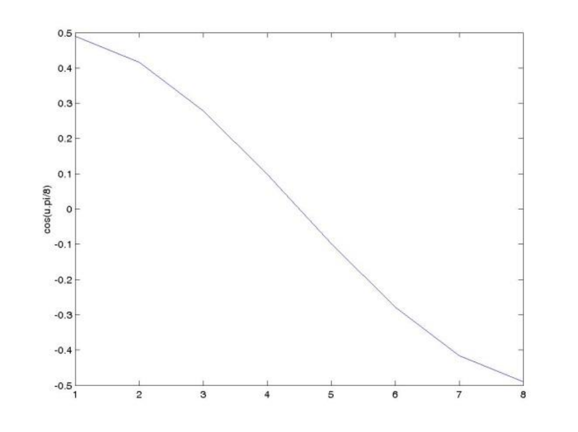


# Basis functions

- Third term

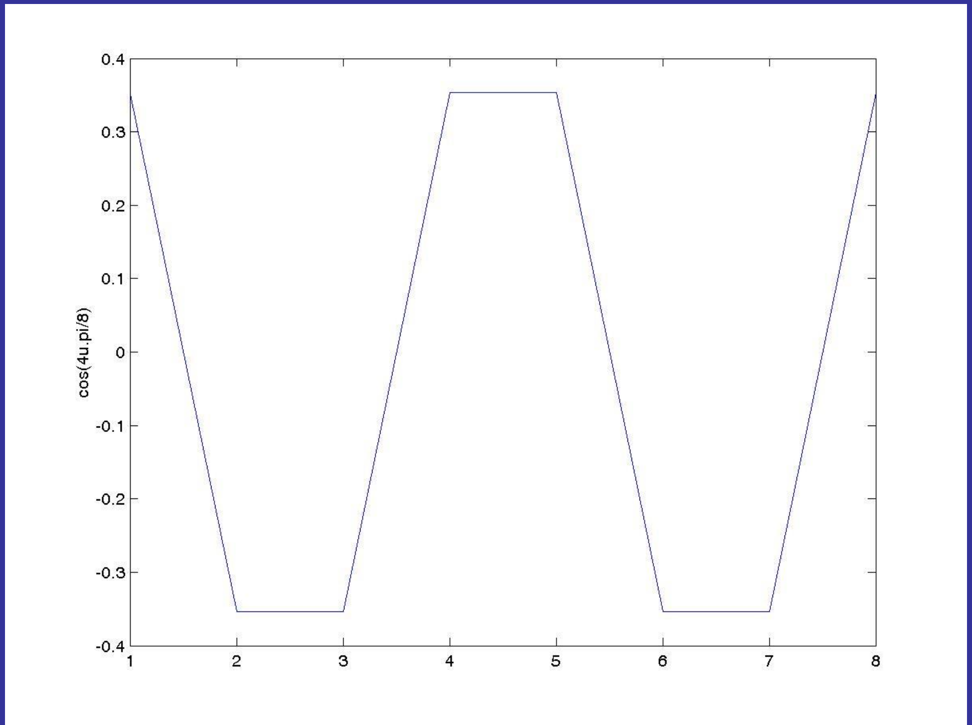


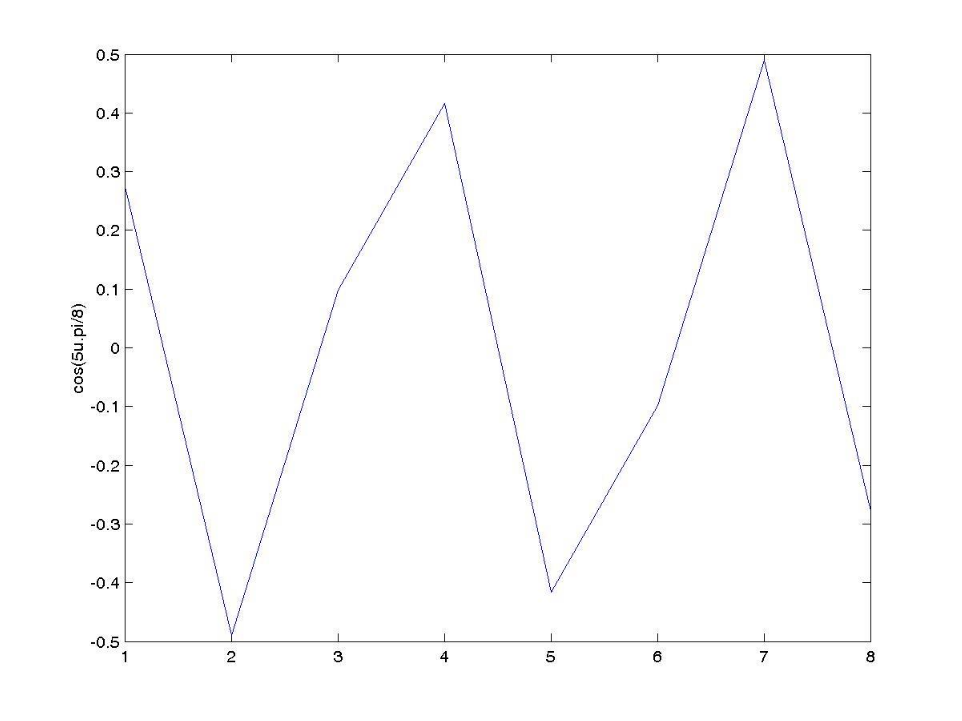
# Basis functions

- Fourth term



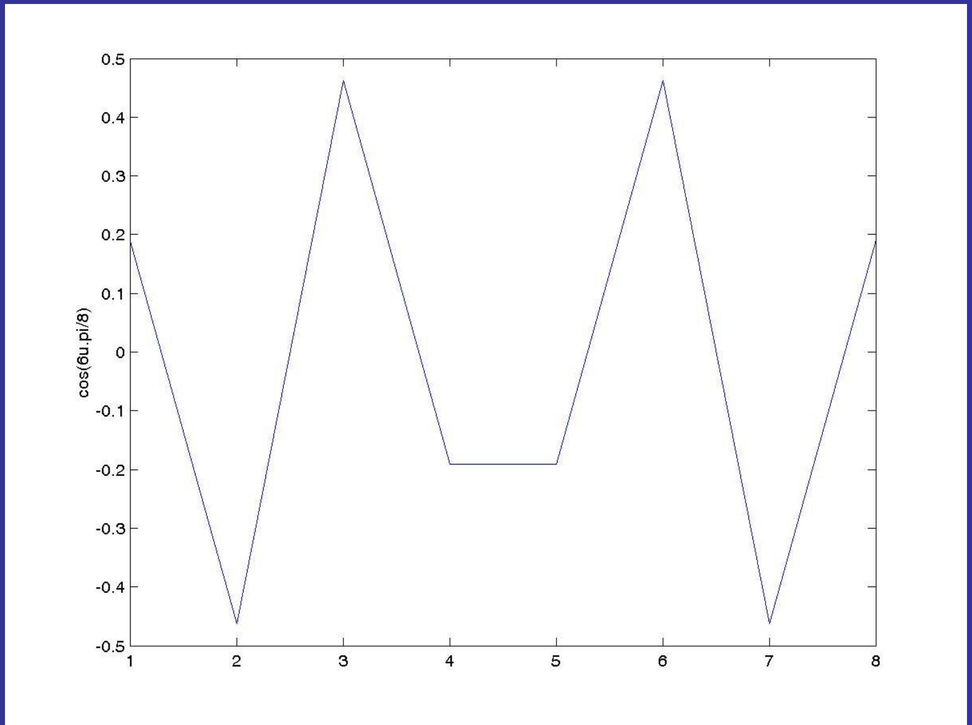
# Basis functions

- Fifth term



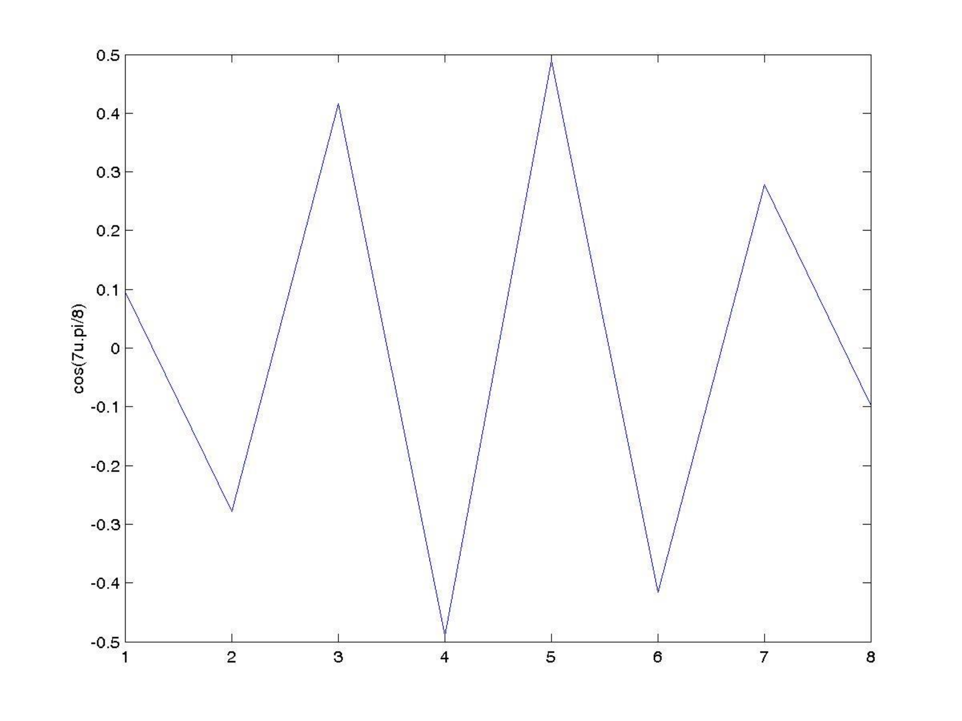
# Basis functions

- Sixth term

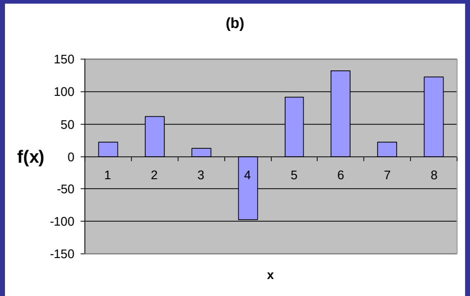


# Basis functions

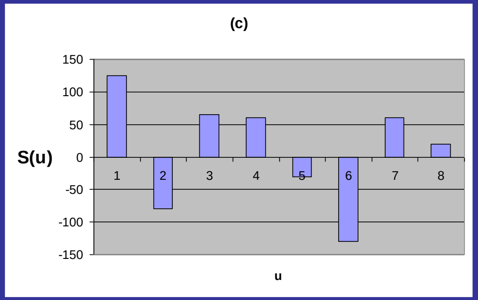
- Seventh term



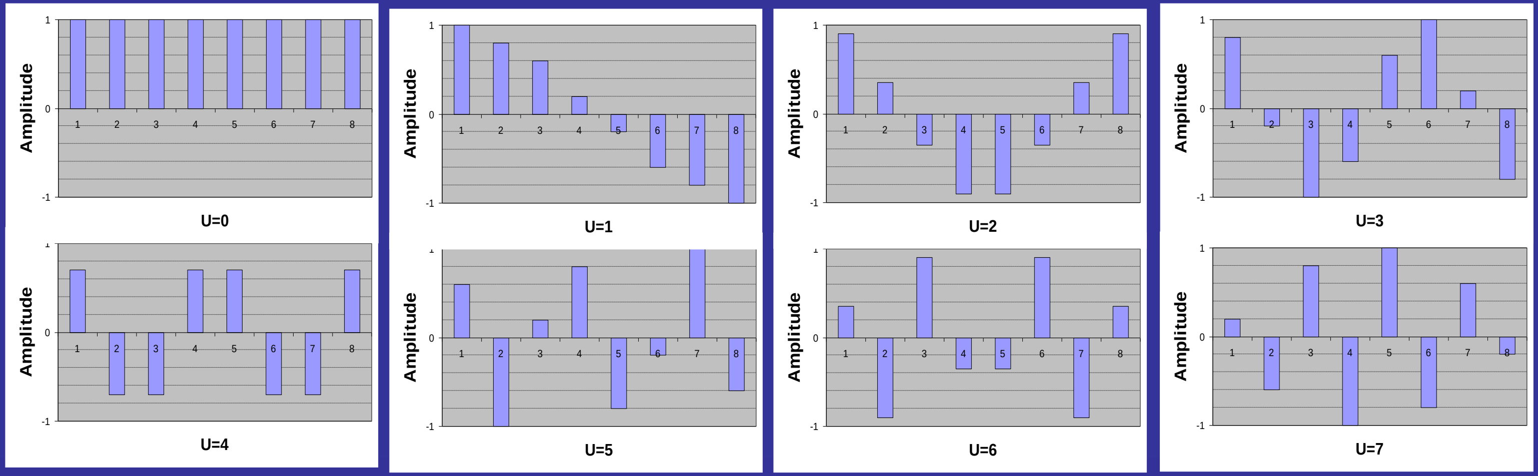
# Another example of 1-D DCT decomposition




Before DCT (image data)




After DCT (coefficients)




The 8 basic functions for 1-D DCT

# 2-D DCT Transform

- Let i(x,y) represent an image with N rows and M columns
- Its DCT I(u,v) is given by

$$I(u,v)=\frac{1}{4}C(u)C(v)\left[\sum_{x=1}^{M}\sum_{y=1}^{N} i(x,y)\cos\left(\frac{(2x+1)u\pi}{16}\right)\cos\left(\frac{(2y+1)v\pi}{16}\right)\right]$$

- where

$$C(0)=\frac{1}{\sqrt{2}} \qquad C(u)=1$$

# DCT Coefficients

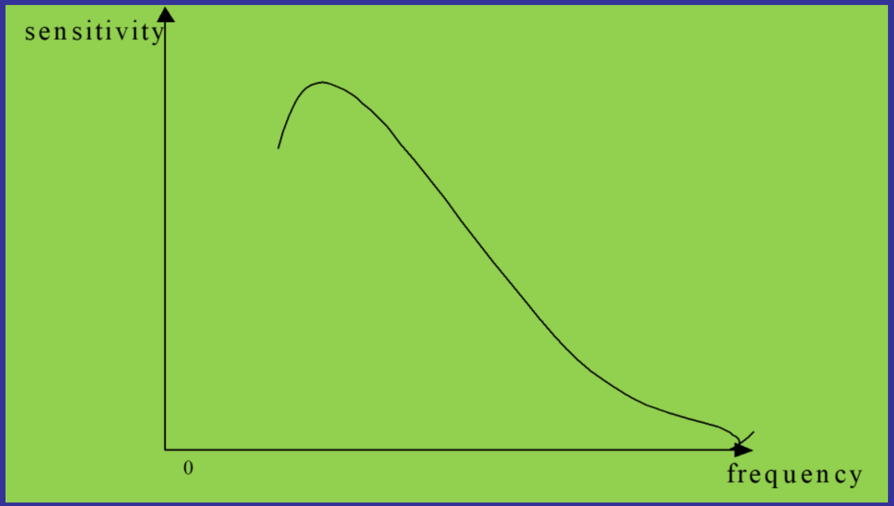
- The DCT coefficient values can be regarded as the relative amounts of the 2-D spatial frequencies contained in the 8×8 block
- The upper-left corner coefficient is called the **DC coefficient**, which is a measure of the average of the energy of the block
- Other coefficients are called **AC coefficients**, coefficients correspond to high frequencies tend to be zero or near zero for most natural images

# Fundamentals of images

- Discrete cosine transform
  - Coefficients are approximately uncorrelated
    - Except DC term
    - C.f. original 8×8 pixel block
  - Concentrates more power in the low frequency coefficients
  - Computationally efficient
- Block-based DCT
  - Compute DCT on 8×8 blocks of pixels

# Why does it work?

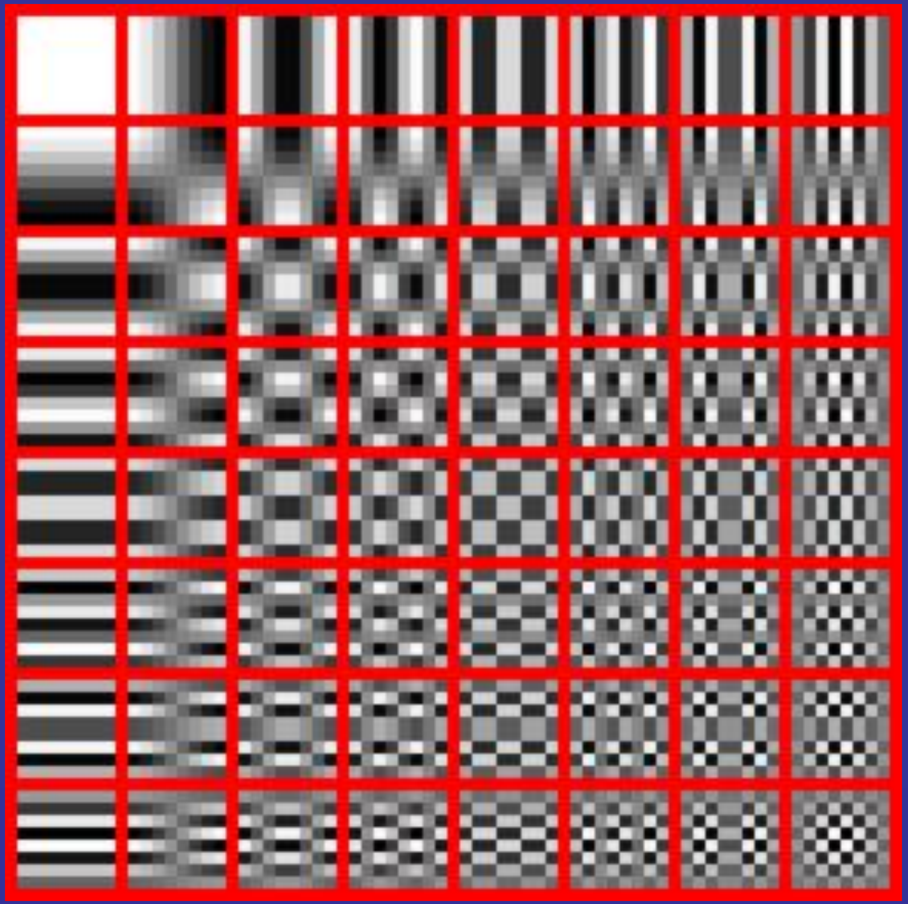
- Lossy encoding
- HVS is generally more sensitive to low frequencies
- Natural images



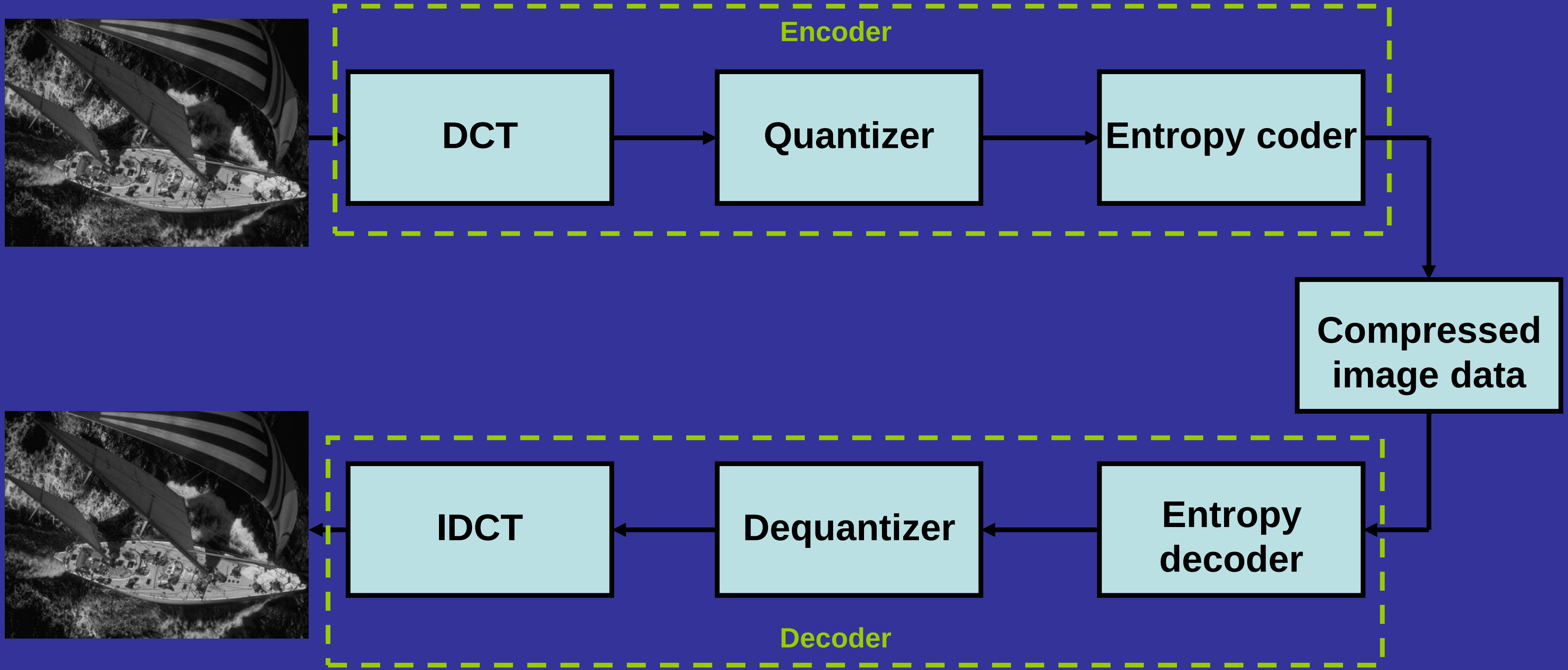

**Frequency sensitivity of Human Visual System**

# Fundamentals of images

- Basis functions for the 8×8 DCT (courtesy Wikipedia)



# Fundamentals of JPEG



# Fundamentals of JPEG

- JPEG works on 8×8 blocks
- Extract 8×8 block of pixels
- Convert to DCT domain
- Quantize each coefficient
  - Different stepsize for each coefficient
    - Based on sensitivity of human visual system
- Order coefficients in zig-zag order
- Entropy code the quantized values

# Fundamentals of JPEG

- A common quantization table is

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 16 | 11 | 10 | 16 | 24 | 40 | 51 | 61 |
| 12 | 12 | 14 | 19 | 26 | 58 | 60 | 55 |
| 14 | 13 | 16 | 24 | 40 | 57 | 69 | 56 |
| 14 | 17 | 22 | 29 | 51 | 87 | 80 | 62 |
| 18 | 22 | 37 | 56 | 68 | 109 | 103 | 77 |
| 24 | 35 | 55 | 64 | 81 | 104 | 113 | 92 |
| 49 | 64 | 78 | 87 | 103 | 121 | 120 | 101 |
| 72 | 92 | 95 | 98 | 112 | 100 | 103 | 99 |

# Fundamentals of JPEG

- A simple example

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -10 | -10 | -10 | 10 | 10 | -10 | -10 | -10 |
| -10 | 10 | 10 | 10 | 10 | 10 | 10 | -10 |
| 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| -10 | 10 | 10 | 10 | 10 | 10 | 10 | -10 |
| -10 | -10 | -10 | 10 | 10 | -10 | -10 | -10 |

*Digitized Image*

→

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 40 | 0 | -26 | 0 | 0 | 0 | -11 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -45 | 0 | -24 | 0 | 8 | 0 | -10 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -20 | 0 | 0 | 0 | 20 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -3 | 0 | 10 | 0 | 18 | 0 | 4 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*After FDCT*

# Fundamentals of JPEG

- A simple example – Cont.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 40 | 0 | -26 | 0 | 0 | 0 | -11 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -45 | 0 | -24 | 0 | 8 | 0 | -10 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -20 | 0 | 0 | 0 | 20 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -3 | 0 | 10 | 0 | 18 | 0 | 4 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*DCT coefficients*

→

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 3 | 0 | -3 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -3 | 0 | -2 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Quantized coefficients*

# Scalar Quantizer vs. Vector Quantizer

- Scalar Quantizer
  - Treats each pixel independently
  - Does not use correlation between neighboring pixels
- Vector Quantizer
  - Image (data) divided into vectors (blocks)
  - Correlation among pixels in vectors is exploited
  - Block size should be appropriate:
    - Too large block : correlation is lost
    - Too small block : More code vectors
  - If no inter-pixel correlation, then no gain

# CV部分内容总结

- 特征分析
- 图像变换