机器学习与视频处理简介

·AIPKU·

人工智能引论实践课 计算机视觉小班

主讲人：胡越予

机器学习部分回顾

- 感知机模型
  - 数学形式（分类超平面）

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathbf{b}) \longrightarrow f(\mathbf{x}) = \text{sgn}(\tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}})$$

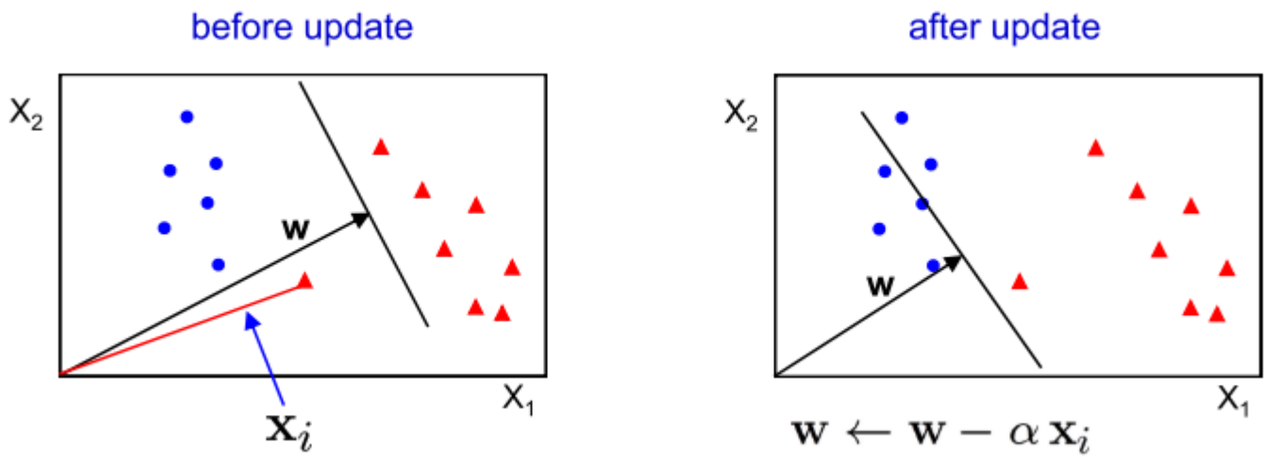
  - 训练算法（针对误分类点）

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha y_i \mathbf{x}_i$$

  - 收敛性

可以证明对于**线性可分**的数据该算法在有限步内结束

$\mathbf{w}$ 实际上直接由各个特征向量的加权和决定



机器学习部分回顾

- 感知机学习转化为最优化问题
- 在给定的数据集上，求参数 $\mathbf{w}$ 和 $\mathbf{b}$，使其为

$$\min_{\mathbf{w},\mathbf{b}} L(\mathbf{w},\mathbf{b}) = -\sum_{\mathbf{x}_i \in M} y_i(\mathbf{w}\cdot\mathbf{x}_i + \mathbf{b})$$

的解，其中 M 是误分类点的点集

- 训练方法：
  - 首先选取一个初始超平面 $(\mathbf{w}_0, \mathbf{b}_0)$
  - 用梯度下降法不断极小化目标函数
  - 损失函数的梯度如下给出

$$\nabla\mathbf{w} = -\sum_{\mathbf{x}_i \in M} y_i\mathbf{x}_i, \nabla\mathbf{b} = -\sum_{\mathbf{x}_i \in M} y_i$$

  - 随机选取误分类点进行更新 $\mathbf{w} \leftarrow \mathbf{w} + \alpha y_i \mathbf{x}_i$

机器学习部分回顾

- 线性可分支持向量机
  - 最大间隔分类超平面（为什么要这么做?
  - 如果所有的点都能分对:

$$y_i(\mathbf{w}\cdot\mathbf{x}_i+\mathbf{b})\geq 1$$

  - $\mathbf{w}$ 实际上由距离分类超平面最近的点决定
    - 这些点使得上式等号成立
    - 这些点被称为支持向量
    - 它们到分类超平面的距离 $r=\dfrac{|\mathbf{w}\cdot\mathbf{x}_i+\mathbf{b}|}{\|\mathbf{w}\|}$
    - 支持向量之间的间隔 $\gamma=2r=\dfrac{2}{\|\mathbf{w}\|}$

机器学习部分回顾

- 线性可分支持向量机最优化问题

Find $\mathbf{w}$ and b such that

$$\rho = \frac{2}{\|\mathbf{w}\|} \text{ is maximized}$$

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Find $\mathbf{w}$ and b such that

$\Phi(\mathbf{w}) = \|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w}$ is minimized

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

机器学习部分回顾

- 软间隔支持向量机
  - 允许部分点更靠近分类超平面甚至被错误分类
  - 参数 C 控制了惩罚力度
    - 更小的C：注重泛化性能
    - 更大的C：注重拟合性能

Find **w** and b such that

$\Phi(\mathbf{w}) = \mathbf{w}^T\mathbf{w} + C\sum\xi_i$ is minimized

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$, , $\xi_i \geq 0$

机器学习部分回顾

- Scikit-Learn 中的SVM

```
sklearn.svm.LinearSVC(
        penalty='l2', loss='squared_hinge', dual=True,
        tol=0.0001, C=1.0, multi_class='ovr',
        fit_intercept=True, intercept_scaling=1,
       class_weight=None, verbose=0, random_state=None,
        max_iter=1000
)
```

机器学习部分回顾

- Scikit-Learn 中的SVM

```
sklearn.svm.SVC(
        C=1.0, kernel='rbf', degree=3,
        gamma='auto_deprecated', coef0=0.0,
        shrinking=True, probability=False, tol=0.001,
        cache_size=200, class_weight=None,
        verbose=False, max_iter=-1,
        decision_function_shape='ovr',
        random_state=None
)
```

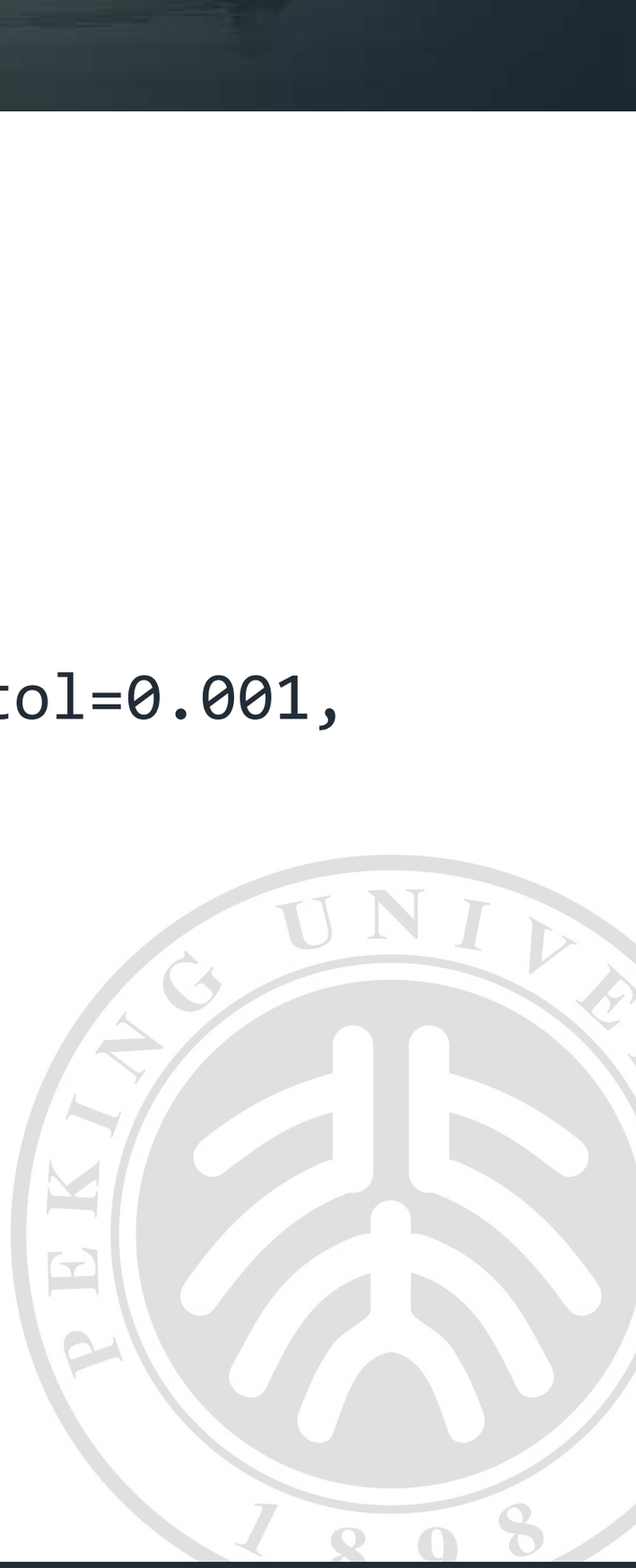
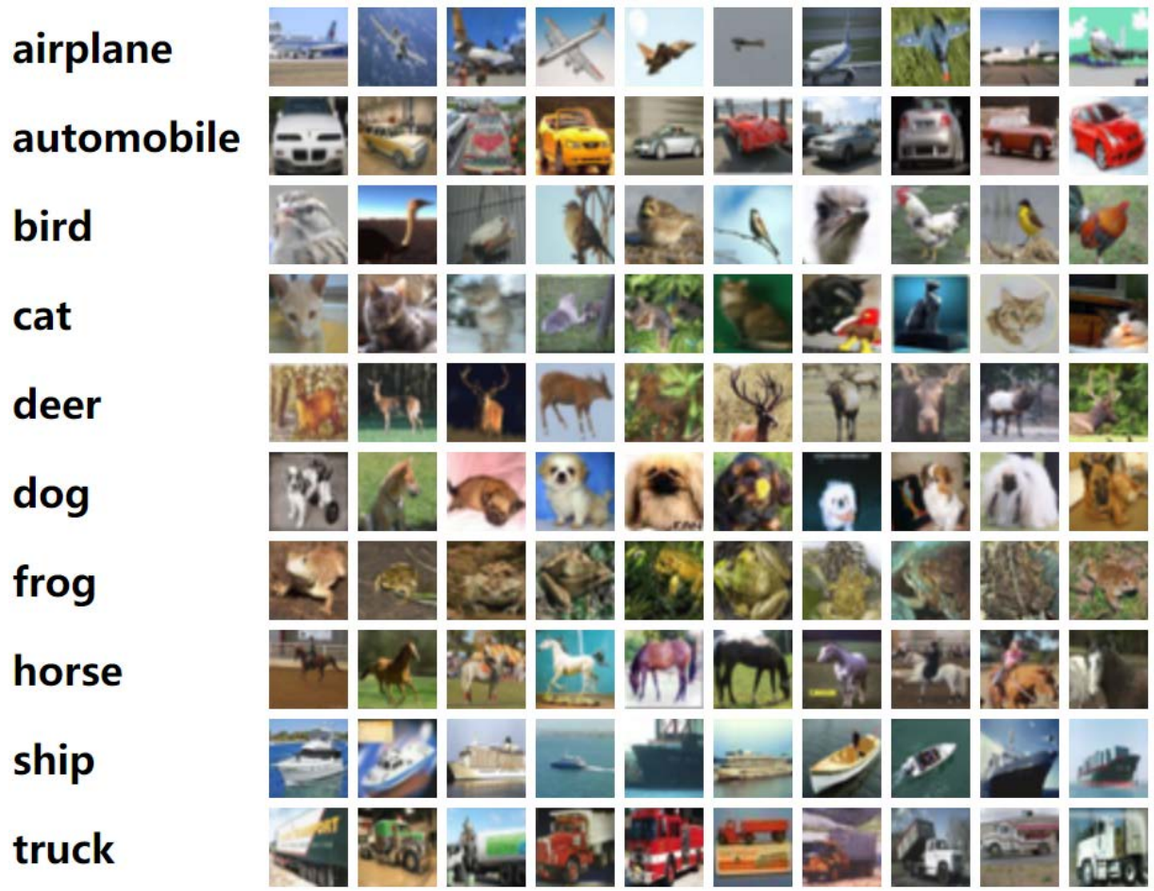


Image Classification: Example

- Cifar 10
  - Tiny image recognition datasets
  - 60,000 32x32 color images in 10 different classes

Version	Size
CIFAR-10 python version	163 MB
CIFAR-10 Matlab version	175 MB
CIFAR-10 binary version (suitable for C programs)	162 MB



"The first 1024 entries contain the red channel values, the next 1024 the green, and the final 1024 the blue. The image is stored in row-major order, so that the first 32 entries of the array are the red channel values of the first row of the image."

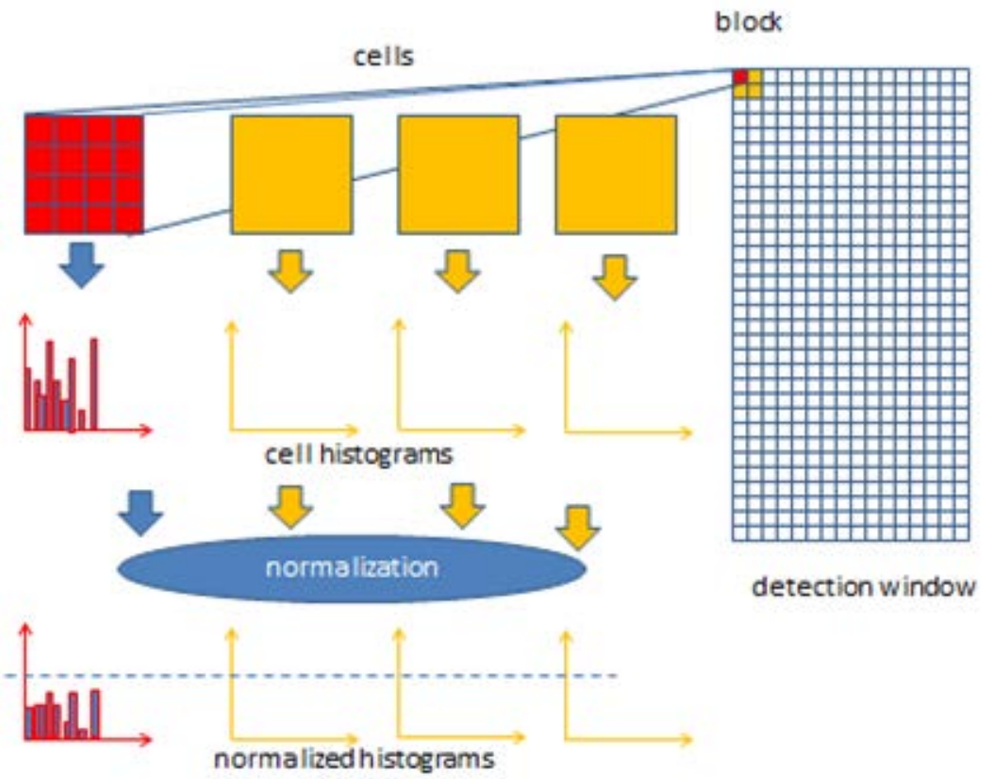
Step 1: Prepare your data

- Load data from binary files (pickle)

```
 8  dicts = []
 9  batch_files = sorted(glob.glob('./data_batch*'))
10  for fn in batch_files:
11      with open(fn, 'rb') as f:
12          d = pickle.load(f, encoding='bytes')
13          dicts.append(d)
14  train_x = []
15  train_y = []
16  for d in dicts:
17      train_x.append(d[b'data'])
18      train_y.append(d[b'labels'])
19  train_x = np.array(train_x)
20  train_x = np.reshape(train_x, (-1, 3, 32, 32)) # why?
21  train_x = np.transpose(train_x, (0,2,3,1))
22  X = []
```

Step 2: Extract features

- Extract features using HOG
  - Choose Scikit-Image Package Largely written in Python
    - Core algorithms written in Cython to achieve performance
  - OpenCV is basically a C++ Library
    - Written in C++, exported to Python
    - Some functions are not Python-Style



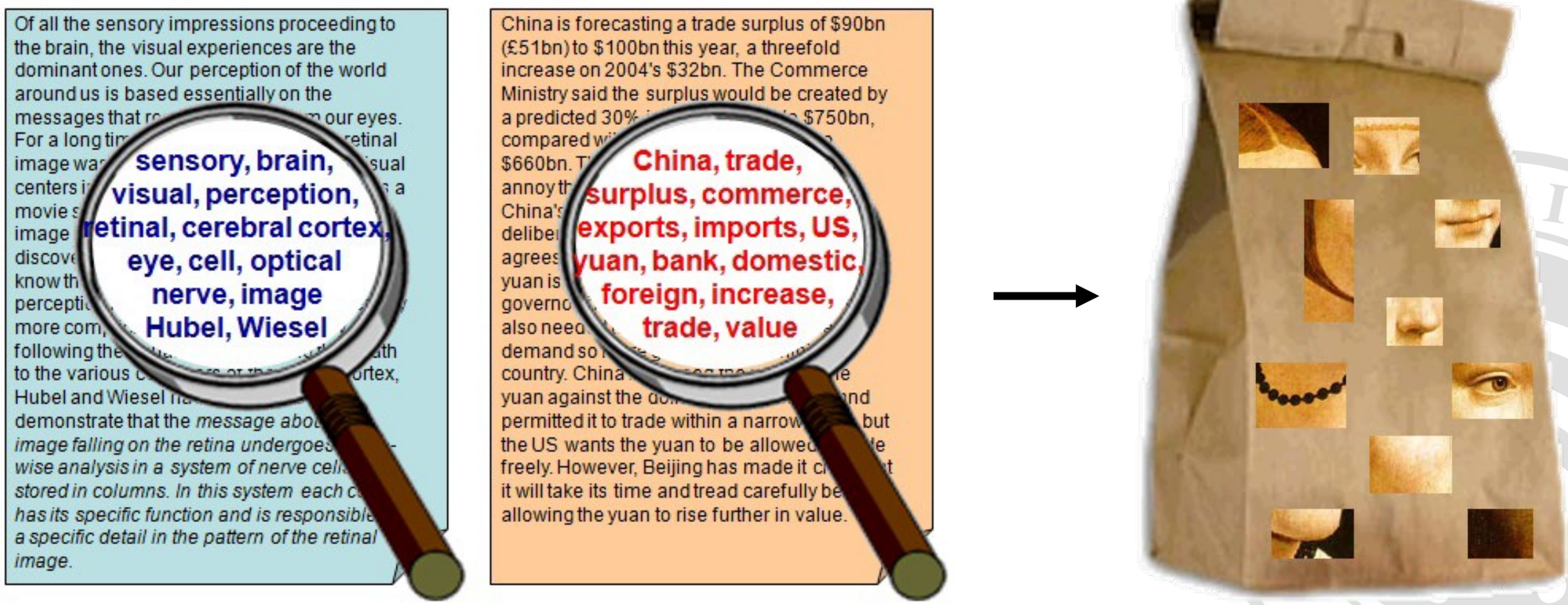

```python
from skimage.feature import hog
X = []
for img in train_x:
    fd = hog(img, pixels_per_cell=(4, 4),
             cells_per_block=(2, 2), multichannel=True, feature_vector=True)
    X.append(fd)
X = np.array(X)
```

Step 3: Classifier: Train and Evaluate

```
23  from skimage.feature import hog
24  X = []
25  for img in train_x:
26      fd = hog(img, pixels_per_cell=(4, 4),
27                    cells_per_block=(2, 2), multichannel=True, feature_vector=True)
28      X.append(fd)
29
30  X = np.array(X)
31  Y = np.reshape(np.array(train_y), (-1, ))
32  clf = svm.LinearSVC(C=0.1)
33  clf.fit(X, Y)
34
35  for img in test_x:
36      fd = hog(img, pixels_per_cell=(4, 4),
37                    cells_per_block=(2, 2), multichannel=True, feature_vector=True)
38      testX.append(fd)
39  testX = np.array(testX)
40  testY = np.reshape(np.array(test_y), (-1, ))
41
42  print('HOG: ', clf.score(testX, testY)) # ~ 55.6%
```

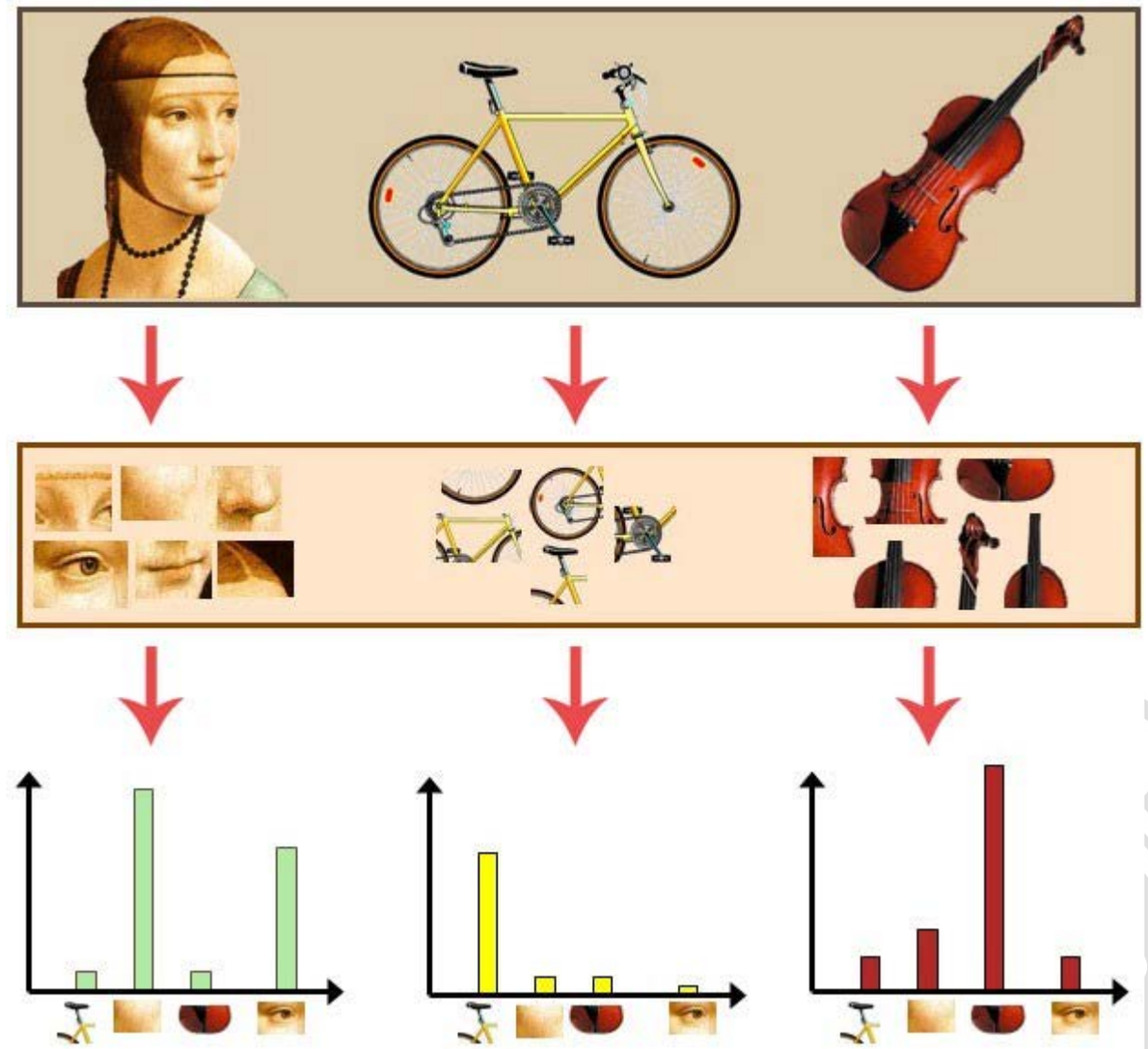
Bag of Feature (BOF)

- Bag of Feature (BoF), aka Bag of Visual Words
  - From Bag of Words in NLP
  - Documents represented by bags of key words
  - Images represented by bags of features



General Idea of BoF

- Key points are stand-out points of an image
  - Scale / Rotate invariant
- Descriptors: descriptions of key points
- Vocabularies: Key points + Descriptors
- Frequency histogram as the representation of an image



K-Means

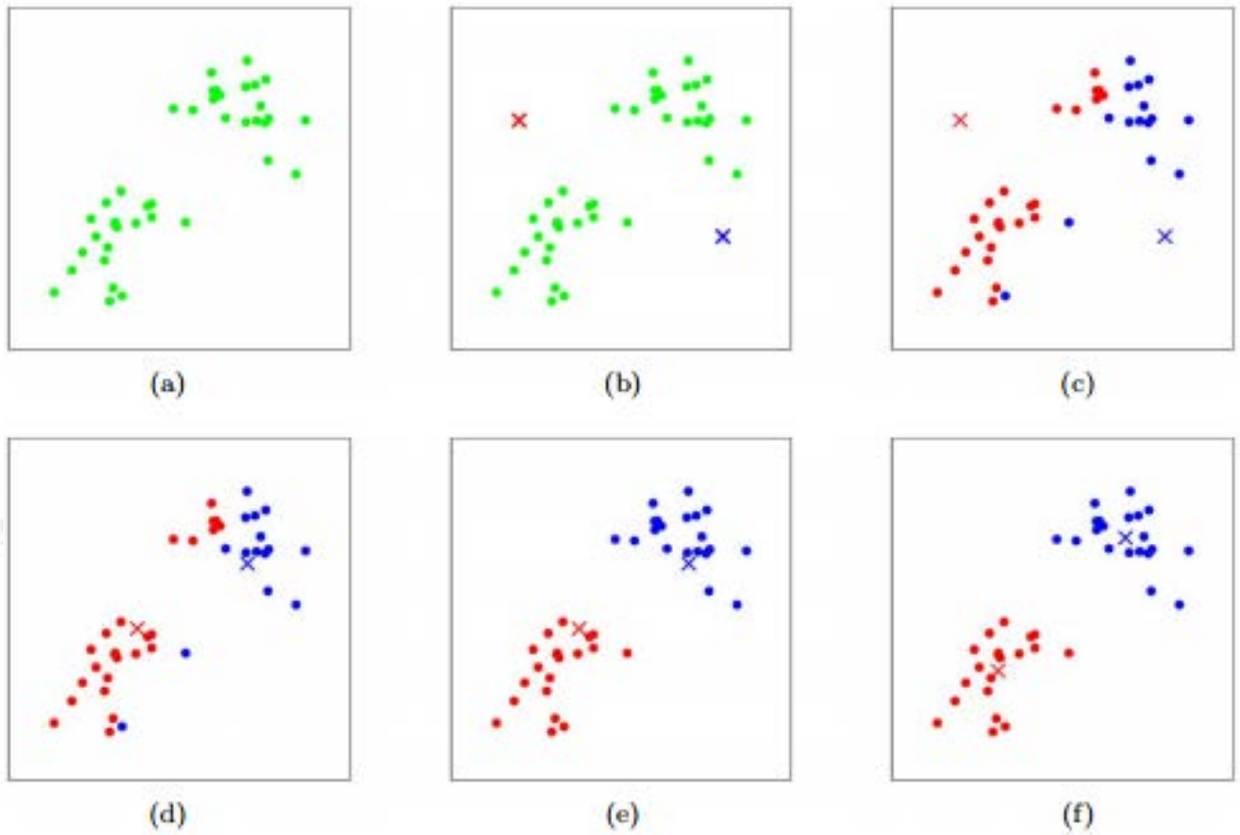
- Clustering Algorithm
  - Find possible clusters in data points
  - Stochastic Initialization
- `sklearn.cluster.KMeans`

1. Initialize **cluster centroids** $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$ randomly.
2. Repeat until convergence: {

   For every i, set
   $$c^{(i)} := \arg\min_j \|x^{(i)} - \mu_j\|^2.$$

   For each j, set
   $$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$$

   }



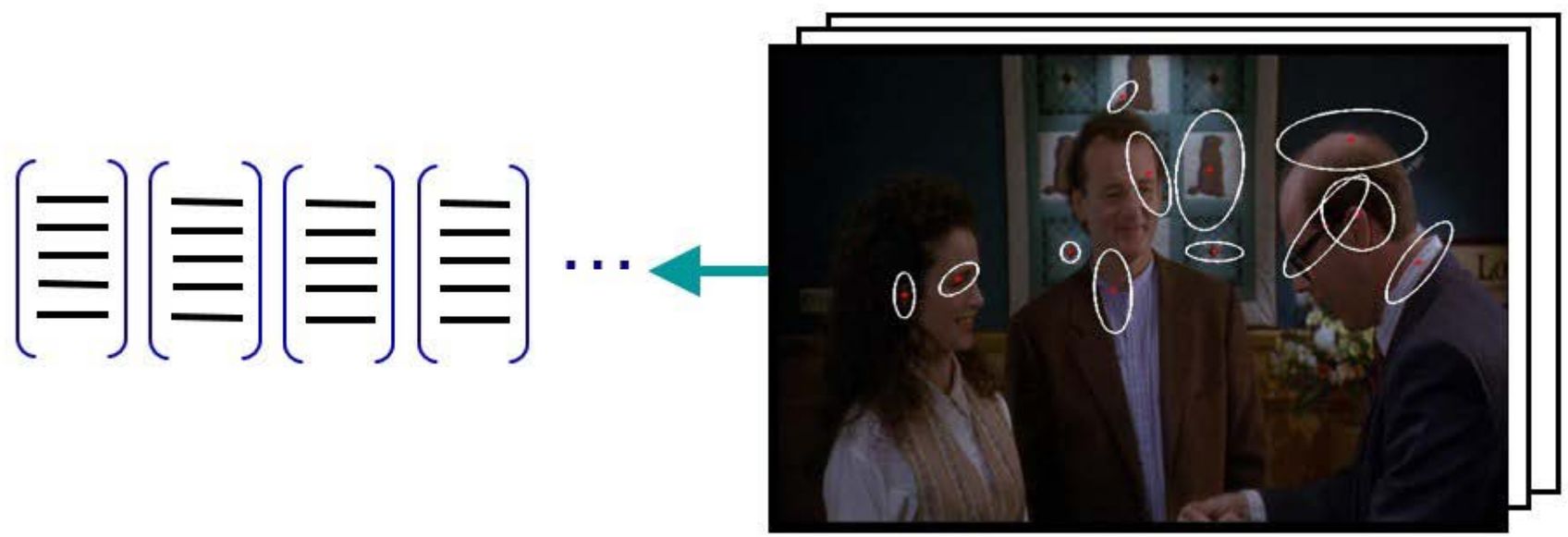
Implement BoF

- 1. Detect key points and extract feature

```python
import cv2

# defining feature extractor that we want to use
extractor = cv2.xfeatures2d.SIFT_create()

def features(image, extractor):
    keypoints, descriptors = extractor.detectAndCompute(image, None)
    return keypoints, descriptors
```



Implement BoF

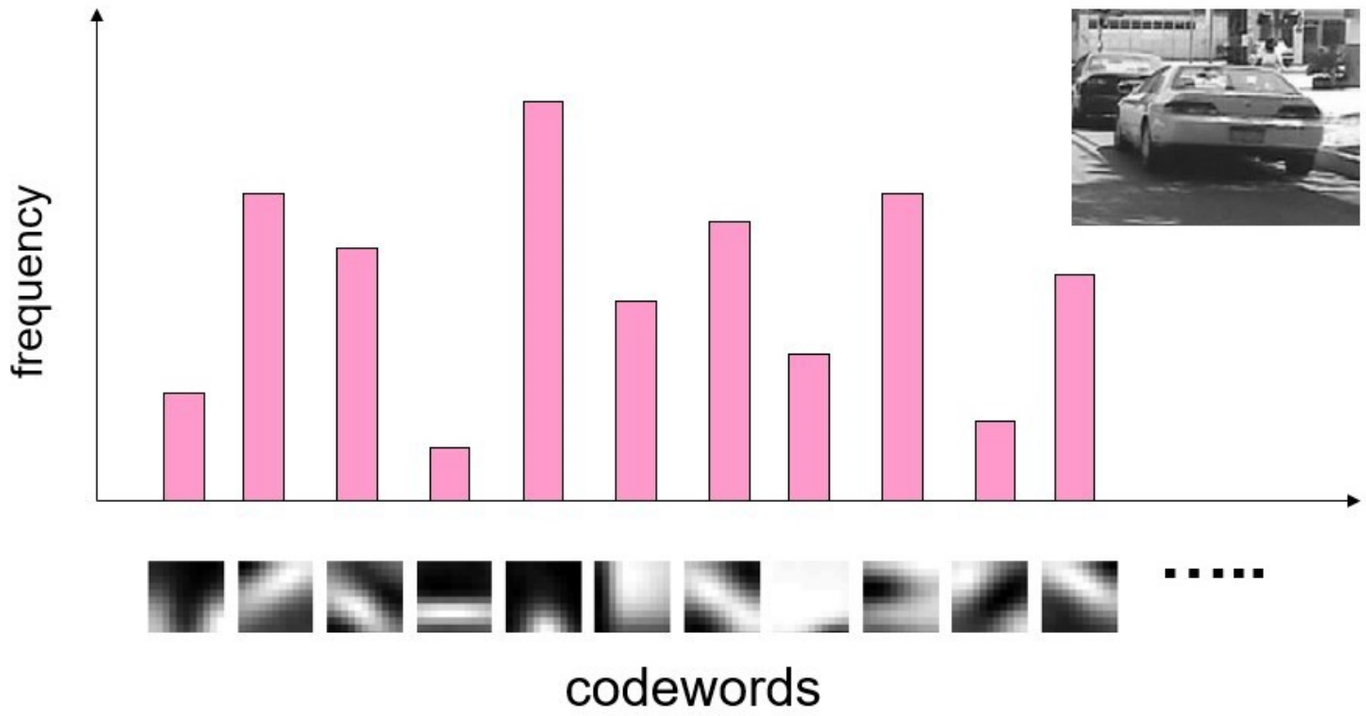
- 2. Clustering to find vocabularies

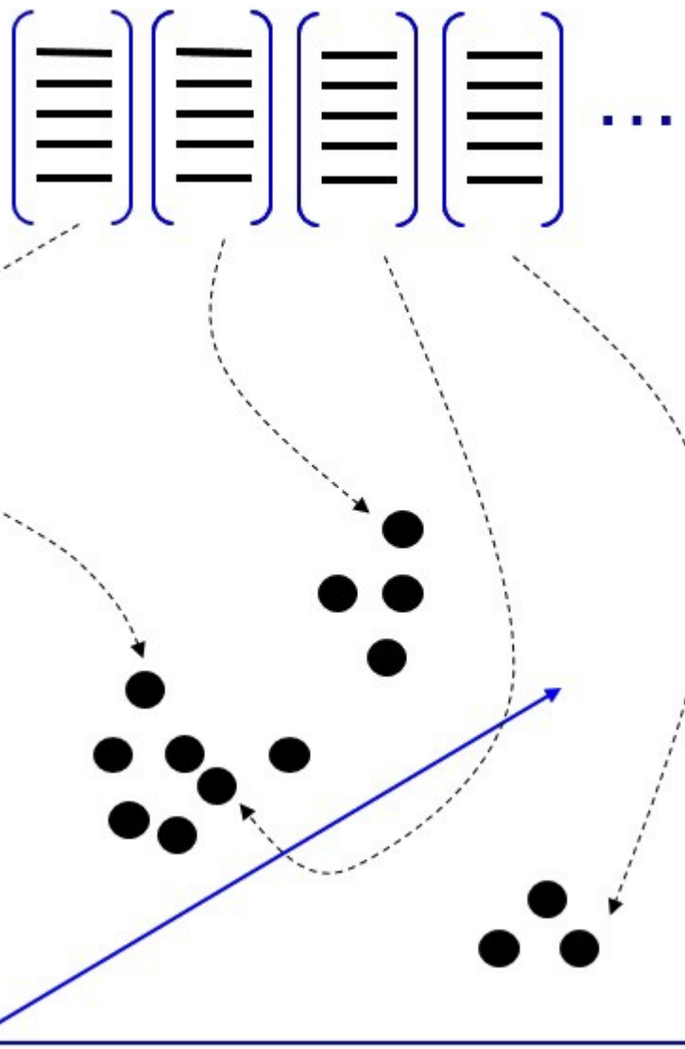
```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters = 800)

kmeans.fit(descriptor_list)
```

- 3. Build histogram of features





补充：Naïve Bayes Classifier

- 小明在下雨天有20%的可能会跑步，在晴天有70%的可能会跑步，已知下雨和晴天的可能性分别是40%和60%，现在发现小明跑步了，求今天下雨的可能性?

- For an input d and a class c

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

Naïve Bayes Classifier (I)

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(c \mid d)$$

MAP is "maximum a posteriori" = most likely class

$$= \underset{c \in C}{\operatorname{argmax}} \frac{P(d \mid c)P(c)}{P(d)}$$

Bayes Rule

$$= \underset{c \in C}{\operatorname{argmax}} P(d \mid c)P(c)$$

Dropping the denominator

Naïve Bayes Classifier (II)

- 小明在下雨天有20%的可能会跑步，在晴天有70%的可能会跑步。小红在下雨天有90%的可能性会把花盆搬到室外，在晴天则有50%的可能性把花盆搬到室外。
- 已知下雨和晴天的可能性分别是40%和60%。现在看到小明跑步了，小红把花盆搬到了室外。

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(d \mid c) P(c)$$

$$= \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

Document d represented as features x1..xn

Naïve Bayes Classifier (IV)

- 现在有很多很多的观测数据：
- （跑步，搬花盆，下雨），（跑步，搬花盆，天晴），（不跑步，搬花盆，下雨），（跑步，不搬花盆，下雨），（跑步，搬花盆，下雨），（跑步，搬花盆，下雨），

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}}\, P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

O(|X|^n•|C|) parameters

How often does this class occur?

Could only be estimated if a very, very large number of training examples was available.

We can just count the relative frequencies in a corpus

Independence Assumptions

- **Bag of Words assumption**: Assume position doesn't matter $P(x_1, x_2, \ldots, x_n \mid c)$
- **Conditional Independence**: Assume the feature probabilities $P(x_i \mid c_j)$ are independent given the class c.

$$P(x_1, \ldots, x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet P(x_3 \mid c) \bullet \ldots \bullet P(x_n \mid c)$$

Multinomial Naïve Bayes Classifier

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}}\, P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}}\, P(c_j) \prod_{x \in X} P(x \mid c)$$

Multinomial Naive Bayes Classifier

- 现在有很多很多的观测数据：
- （跑步，搬花盆，下雨），（跑步，搬花盆，天晴），（跑步，搬花盆，下雨），（跑步，搬花盆，下雨），（跑步，搬花盆，下雨），（跑步，搬花盆，下雨），（跑步，搬花盆，下雨），
- 假设小明和小红的行动是独立的。

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

Learning the Model

Sec.13.3

- Maximum likelihood estimates
  - simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{doccount(C = c_j)}{N}$$

$$\hat{P}(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

Problem with Maximum Likelihood

Sec.13.3

- What if we have seen no training sample for a pair of observation and class?

$$\hat{P}(\text{run} \mid \text{raining}) = \frac{count(\text{run}, \text{raining})}{\sum_{w \in V} count(w, \text{raining})} = 0$$

- Zero probabilities cannot be conditioned away, no matter the other evidence!

$$c_{MAP} = \text{argmax}_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$

Smoothing for Naïve Bayes

$$\hat{P}(w_i \mid c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} \left(count(w, c) + 1 \right)}$$

$$= \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c) \right) + |V|}$$

Summary

- 分类问题的算法流程
  - 提取特征 (HoG, SIFT,…)
  - 特征编码 (BoF, Fisher Vector, …)
  - 分类器训练 (Perceptron, SVM, Naïve Bayes,…)
- 学习算法
  - 感知机：分类超平面，随机梯度下降
  - SVM：线性可分，软间隔，核方法（非线性SVM）

工具：视频处理简介

- 视频是什么?
  - 电影，电视，流媒体，直播，…
  - 容器 + 视频流 + 音频流 [+ 字幕流]
  - 视频流：二进制码流
  - 解码：从二进制码流生成图像序列
  - 视频：按顺序排列的图像序列
- 视频从哪里来?
  - 摄像机每秒采集大量图像（分辨率?)
  - 编码器：对图像序列进行压缩编码，形成码流（码率？）
  - 混流：将视频码流，音频码流打包在容器中（文件格式？）

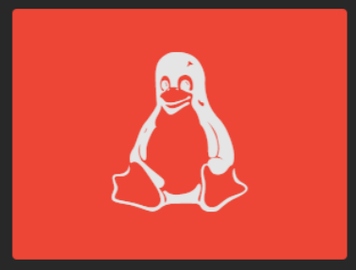
视频处理工具 FFmpeg

- What is FFmpeg
- “A complete cross-platform solution to record, convert, and stream audio and video…the leading multimedia framework able to decode, encode, transcode, mux, demux, stream, filter, and play pretty much anything that humans and machines have created.”

- Some things you can do…
  - Re-wrap/encode files，Edit files，Parse streams from files，Add audio/video effects and filters，…

FFmpeg 安装

Get the packages



Linux Packages

Debian – Official packages for Stable-Backports, Testing, Unstable

Debian – deb-multimedia packages for Oldstable, Stable, Testing, Unstable

Ubuntu – Official packages for Vivid, Wily, Xenial

Ubuntu – Ubuntu Multimedia for Trusty PPA. Provides static binaries from most recent release branch.

Fedora and **Red Hat Enterprise Linux** packages

Linux Static Builds

32-bit and 64-bit for kernel 2.6.32 and above

Get the Sources

Download Snapshot

You can retrieve the source code through Git by using the command:

```
git clone https://git.ffmpeg.org/ffmpeg.git ffmpeg
```

Cannot access Git or wish to speed up the cloning and reduce the bandwidth usage? git Snapshot Browse

> FFmpeg has always been a very experimental and developer-driven project. It is a key component in many multimedia projects and has new features added constantly. Development branch snapshots work really well 99% of the time so people are not afraid to use them.

Git Repositories

Our 1st command – Simple Re-Wrap

Re-Wrap = to change a file's container (i.e. file extension)

```
ffmpeg -i new_file.mkv -c copy -map 0 new_file.avi
```

[Command Prompt] [Input File] [Flags/Actions] [Output File]

ffmpeg -i new_file.mkv -c copy -map 0 new_file.avi

ffmpeg starts the command

-i input_file.ext path and name of the input file

-c copy copy the streams directly, without re-encoding.

-map 0 map all streams of the input to the output. By default, FFmpeg will only map one stream of each type (video, audio, subtitles) to the output file. However, files may have multiple streams of a given type - for example, a video may have several audio tracks for different languages. Therefore, if you want to preserve all the streams in the original, it's necessary to use this option.

output_file.ext path and name of the output file. The new container you are rewrapping to is defined by the filename extension used here, e.g. .mkv, .mp4, .mov.

2nd command – Decode

Decode a video, extract frames as images.

[Command Prompt] [Input File] [Flags/Actions] [Output File]

```
ffmpeg -i example.mp4 %05d.png
```

[Example]

```
ffmpeg –i example.mp4 -ss 0:0:35 -t 25 -qscale:v 2 Frame%05d.jpg
```

输入example.mp4，从00:00:35开始，持续25秒，将这段视频解码存成JPEG图像 Frame00000.jpg，Frame00001.jpg ...

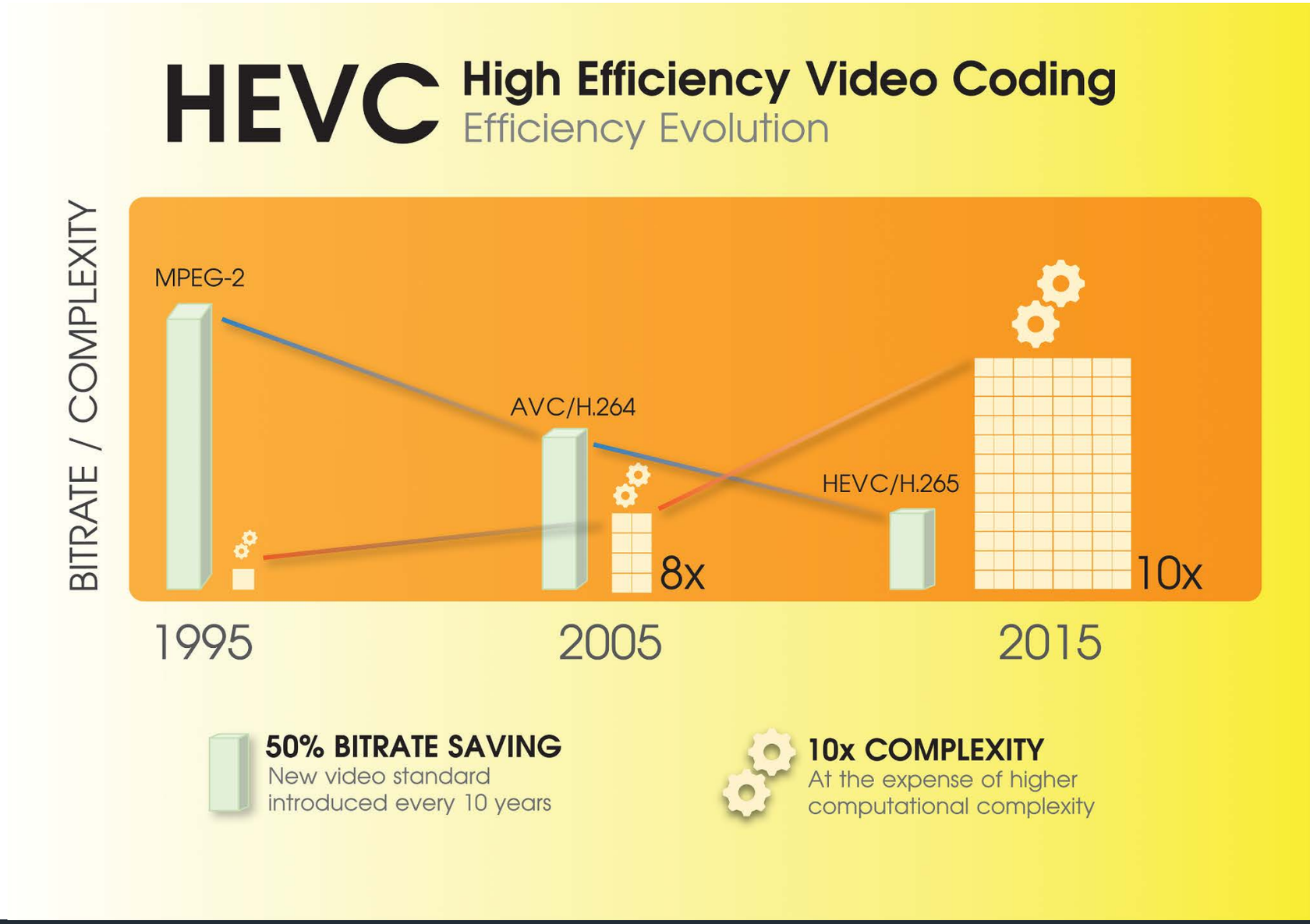
3rd command – Re-Encode

```
[Command Prompt] [Input File] [Flags/Actions] [Output File]

ffmpeg -i example.mov -c:v libx264 -crf 18 example.mp4
```

使用libx264将视频重新编码，使用H.264编码标准， crf控制质量/码率

- 编码标准:
  - AVC/H.264
  - HEVC/H.265
  - AV1 (Google)
  - AVS2



4th command: Encode Video from Images

For this to work properly, image files should be numbered and named consistently and sequentially (i.e. input_name_001, input_name_002, etc.)

[Command Prompt] [Input File] [Flags/Actions] [Output File]

```
ffmpeg -framerate 4 -pattern_type glob -i "input_name_*.jpg"
outputfile.gif
```

```
ffmpeg -i %05d.png -c:v libx264 -crf 18 -pix_fmt yuv420p result.mp4
# -pix_fmt yuv420p 使得产生的视频能够被大多数硬件解码器解码
```