Data Engineering

204426

Feature Extraction

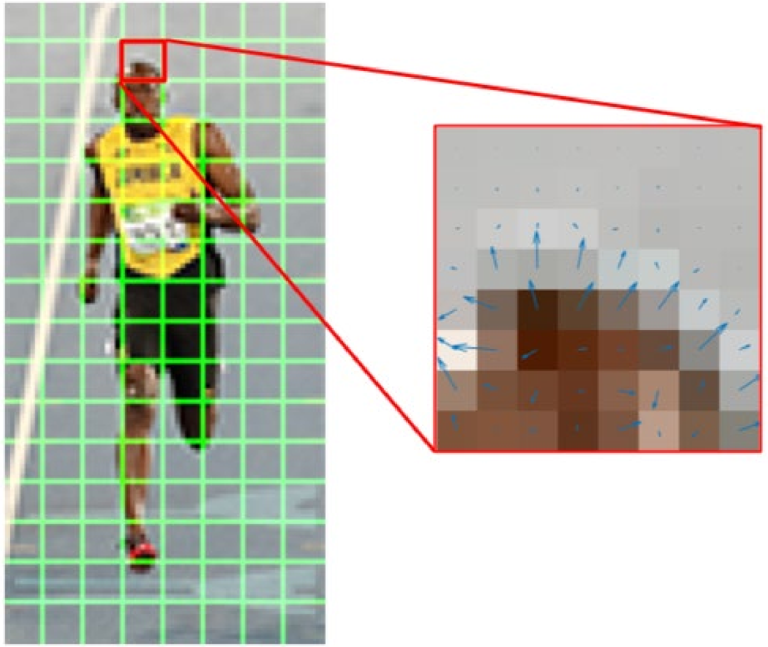
Feature Extraction

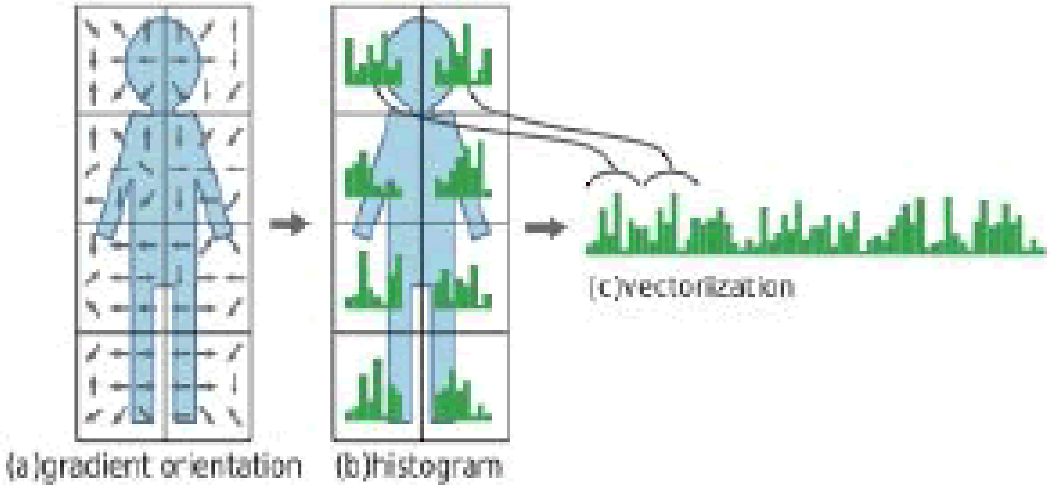
- **Feature extraction** involves reducing the number of resources required to describe a large set of data.
- **Feature extraction** is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing.
- **Feature extraction** is the name for methods that select and /or combine variables into features, effectively reducing the amount of data that must be processed, while still accurately and completely describing the original data set.

Feature Extraction

Feature Extraction for Images

Histogram of Oriented Gradients (HOG)





Source: https://www.learnopencv.com/histogram-of-oriented-gradients/
http://www.rroij.com/open-access/pedestrian-detectiona-comparative-studyusing-hog-and-cohog.php?aid=51543

Feature Extraction for Images

Histogram of Oriented Gradients (HOG)

1. Calculate the Gradient Images.
2. Calculate Histogram of Gradients
3. Block Normalization
4. Calculate the HOG feature vector

Feature Extraction for Images

Histogram of Oriented Gradients (HOG)

1. **Calculate the Gradient Images**

- Apply a convolution operation to obtain the gradient images:

$$G_x = I * H_x, \qquad G_y = I * H_y$$

- Compute the final gradient magnitude

$$|G| = \sqrt{G_x^2 + G_y^2}$$

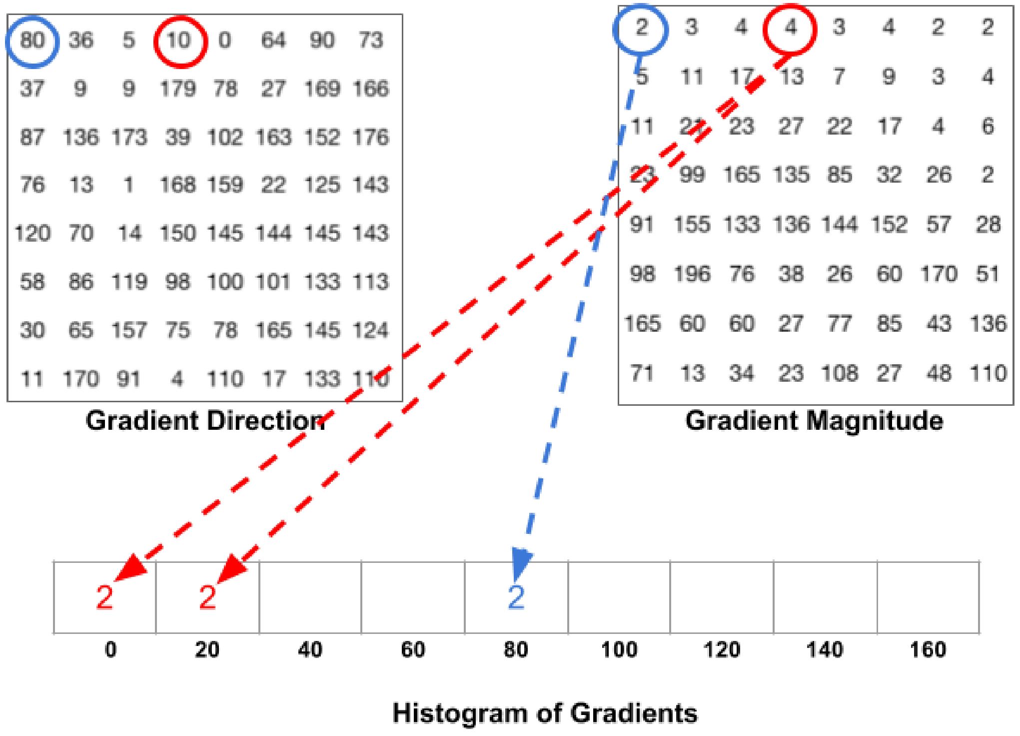
- Compute the orientation of the gradient

$$\theta = \arctan \frac{G_y}{G_x}$$

Feature Extraction for Images

Histogram of Oriented Gradients (HOG)

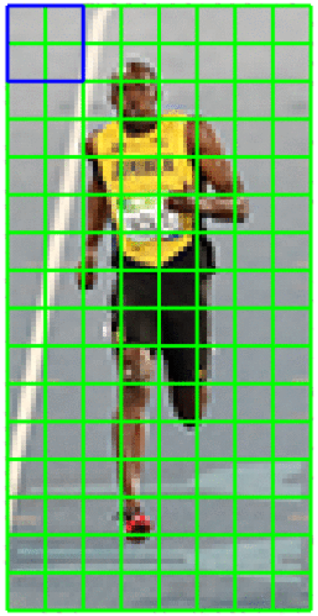
2. Calculate Histogram of Gradients



Feature Extraction for Images

Histogram of Oriented Gradients (HOG)

3. Block Normalization



For each of the cells in the current block

- Concatenate their corresponding gradient histograms
- Perform L1 or L2 normalization by dividing each element of the histogram by L1 or L2 norm.

L1 Norm: $\|x\|_1 = \sum_{i=1}^{n} |x_i|$

L2 Norm: $\|x\|_1 = \sqrt{\sum_{i=1}^{n} x_i^2}$

Feature Extraction for Images

Histogram of Oriented Gradients (HOG)

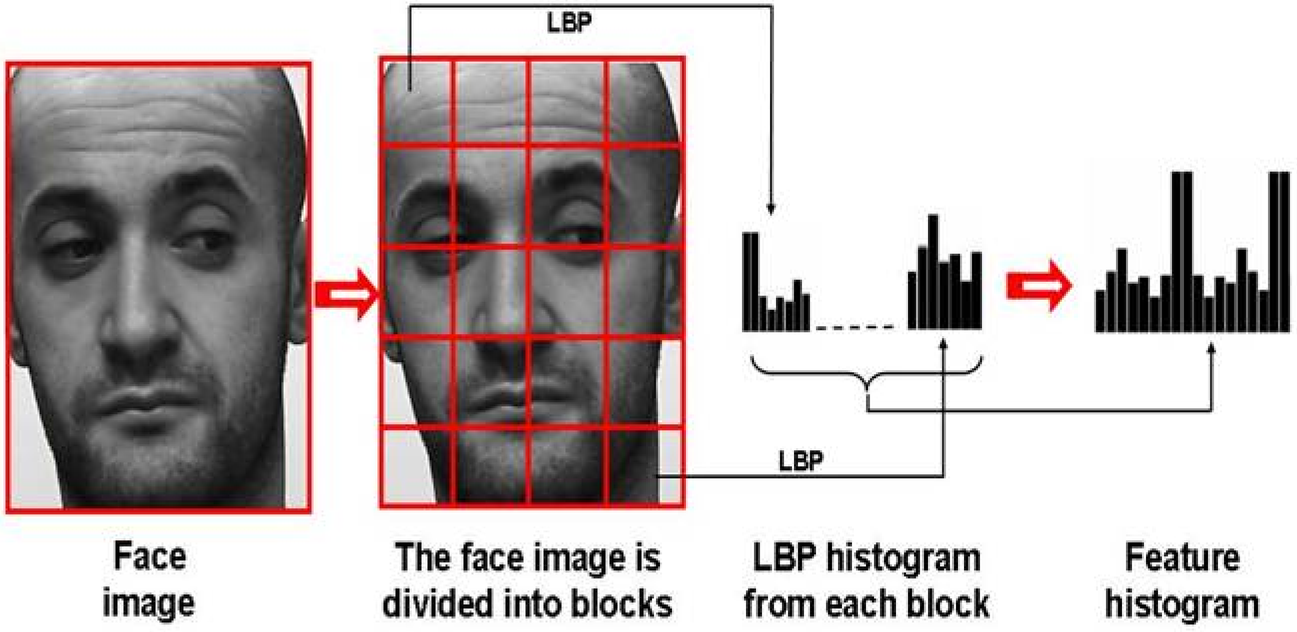
4. Calculate the HOG feature vector

After all blocks are normalized

- we take the resulting histograms
- concatenate them
- treat them as our final feature vector.

Feature Extraction for Images

Local Binary Patterns (LBP)



Source: http://www.scholarpedia.org/article/File:LBP-face.jpg

Feature Extraction for Images

Local Binary Patterns (LBP)

1. Convert the image to grayscale
2. For each pixel in the grayscale image
    1. select a neighborhood of size *r* surrounding the center pixel.
    2. calculate LBP value for this center pixel
3. Compute a histogram over the output LBP array

Feature Extraction for Images

Local Binary Patterns (LBP)

Calculating LBP value

5	8	1
5	4	1
3	7	2

0	0	1
0		1
1	0	1

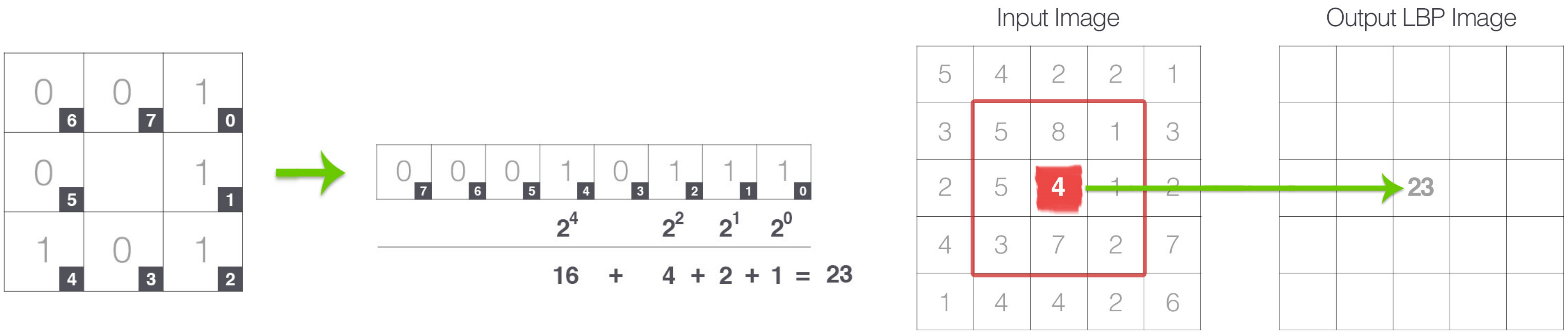
If the intensity of the center pixel is greater than or equal to its neighbor, then we set the value to *1*; otherwise, we set it to *0*.

Threshold the center pixel against its neighbor pixels

Feature Extraction for Images

Local Binary Patterns (LBP)

Calculating LBP value



- Start at the top-right point and work our way ***clockwise*** accumulating the binary string as we go along.
- Convert this binary string to decimal.
- Store in an output array with the same width and height as the original image.

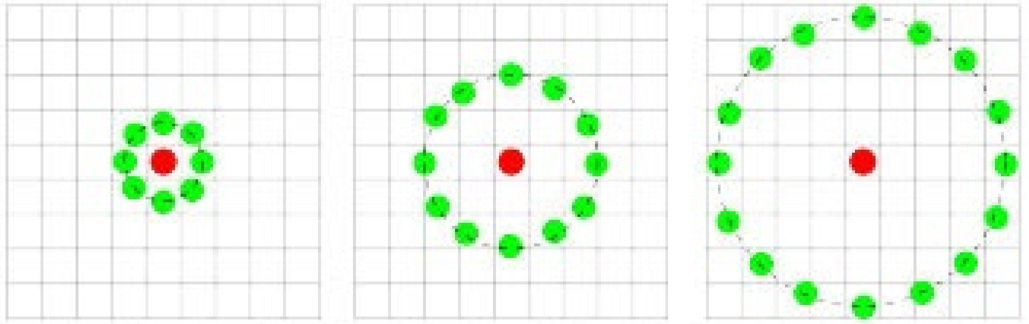
Feature Extraction for Images

Local Binary Patterns (LBP)

Neighborhood Sizes

To account for variable neighborhood sizes, two parameters were introduced:

- *p*: the number of points in a circularly symmetric neighborhood to consider.
- *r*: the radius of the circle , which allows us to account for different scales.



Feature Extraction for Images

Local Binary Patterns (LBP)

Neighborhood Sizes

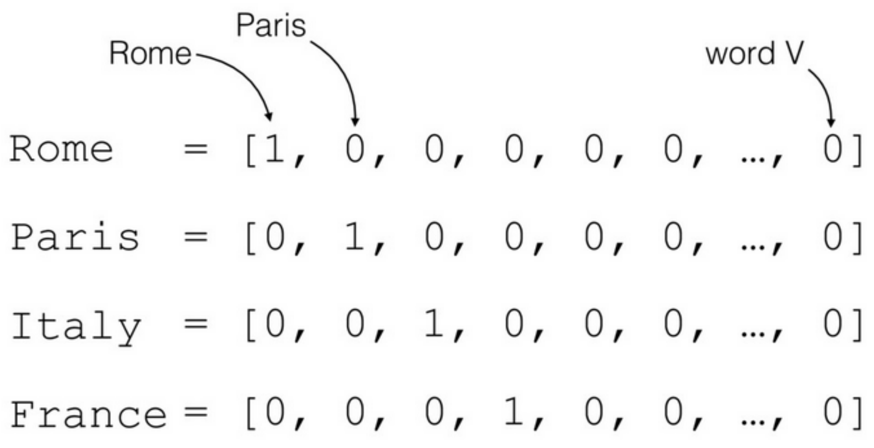
The Concept of LBP Uniformity

- A LBP is considered to be uniform if it has ***at most*** two *0-1* or *1-0* transitions.
  - 00001000 : 2 transitions -> **uniform pattern**
  - 10000000 : 1 transitions -> **uniform pattern**
  - 01010010 : 6 transitions -> **non-uniform pattern**
- Uniform LBP patterns add an extra level of *rotation and grayscale invariance.*

Feature Extraction for Texts

One-hot Encoding

- A representation of categorical variables as binary vectors.
- Each word is represented as a binary vector that is:
  - All zero values
  - Except the index of the word, which is marked with a 1.



Source: https://medium.com/@athif.shaffy/one-hot-encoding-of-text-b69124bef0a7

Feature Extraction for Texts

Bag of Words

- What about full texts instead of single words?
- The vector representation of a text is simply the vector sum of all the words it contains:

All possible words

Word

$$
\begin{array}{rl}
 & \quad . \quad \text{It} \quad \text{a} \quad \text{cat} \quad \text{is} \\
\text{It} = & [0., 1., 0., 0., 0.], \\
\text{is} = & [0., 0., 0., 0., 1.], \\
\text{a} = & [0., 0., 1., 0., 0.], \\
\text{cat} = & [0., 0., 0., 1., 0.], \\
. = & [1., 0., 0., 0., 0.] \\
 & {\color{red}[1., 1., 1., 1., 1.]} \leftarrow
\end{array}
$$

Vector sum of all words represents the text "It is a cat."

Feature Extraction for Texts

Bag of Words

- In practice it’s much more convenient to use a dictionary instead of an actual vector
- This is known as a bag-of-words, and word order is discarded.

Dictionary

Word		cat	dog	bird	panda
	They =	[0.,	0.,	0.,	0.],
	are =	[0.,	0.,	0.,	0.],
	cat =	[1.,	0.,	0.,	0.],
	and =	[0.,	0.,	0.,	0.],
	dog =	[0.,	1.,	0.,	0.]
	They are cat and dog =	[1.,	1.,	0.,	0.]

Bag of words represents the text “They are cat and dog”

Feature Extraction for Texts

TF-IDF

Term Frequency (TF)

- The number of times that a word appears in a document is known as the “term frequency” (TF)
- An idea behind TF : “how popular a specific term is within a document”
- Possible definitions of TF:

$$\mathrm{tf}(t,d) = N_{t,d}$$

จำนวน คำ **t** ทั้งหมดที่ปรากฏในเอกสาร/ข้อความ เดียวกัน

$$\mathrm{tf}(t,d) = \frac{N_{t,d}}{\sum_{t'} N_{t',d}}$$

จำนวนคำอื่นที่ไม่ใช่คำ **t** ทั้งหมดที่ปรากฏในเอกสาร/ข้อความเดียวกัน

$$\mathrm{tf}(t,d) = \log(1 + N_{t,d})$$

Feature Extraction for Texts

TF-IDF

Inverse Document Frequency (IDF)

- How much information the word provides.
- An idea behind IDF : “words that appear in more documents are less meaningful”
- Possible definitions of IDF:

$$\mathrm{idf}(t, D) = \log\left(\frac{N}{N_t}\right)$$

จำนวน เอกสาร/ข้อความ ทั้งหมดในคลังข้อมูล

จำนวน เอกสาร/ข้อความ ทั้งหมดที่มี คำ **t** ปรากฏอยู่ในคลังข้อมูล

$$\mathrm{idf}(t, D) = \log\left(1 + \frac{N}{N_t}\right)$$

Feature Extraction for Texts

TF-IDF

Inverse Document Frequency (IDF)

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \times \mathrm{idf}(t, D)$$

```
Document 1: 'All my cat, cat and cat in a row',
Document 2: 'When my cat sits down, she looks like a Furby toy! ',
Document 3: 'The cat from outer space',
Document 4: 'Sunshine loves to sit like this for some reason. ']
```

$$\mathrm{tf}("cat", d_1) = \frac{3}{6}$$

$$\mathrm{idf}("cat", D) = \log\left(\frac{4}{3}\right)$$

$$\mathrm{tfidf}("cat", d_1, D) = \frac{3}{6}\log\left(\frac{4}{3}\right)$$