Lecture Notes for

Machine Learning in Python

⟦👨‍🏫👨‍💻🐍👨‍🔬⟧

Professor Eric Larson

Dimensionality Reduction and Images

Class Logistics and Agenda

- **Logistics**:
  - Lab due soon!
  - Next Time: Flipped Module
  - Turn in one per team (HTML), please include team member names from canvas
- **Agenda**
  - Common Feature Extraction Methods for Images
  - Begin Town Hall, if time

Last time…

E1
0.85
0.52

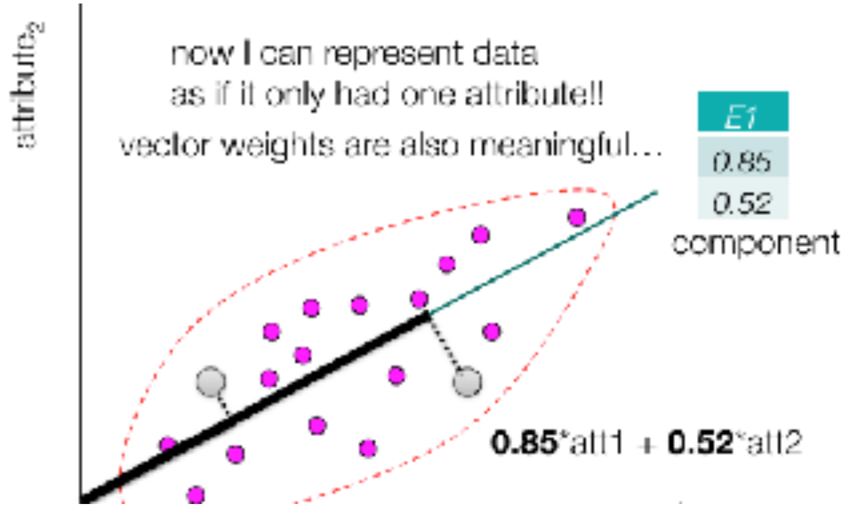
E2
0.85
-0.52

37.1	-6.7
-6.7	43.9

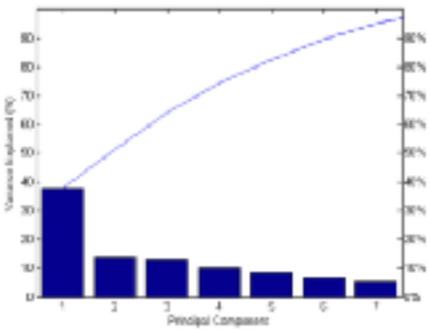
	A1	A2
1	66	33.6
2	54	26.6
3	69	23.3
4	73	28.1
5	61	43.1
6	62	25.6

	A1	A2
1	1.83	3.55
2	-10.1	-3.45
3	4.83	-6.75
4	8.83	-1.95
5	-3.17	13.05
6	-2.17	-4.45

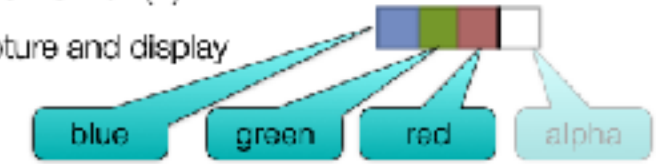
zero mean

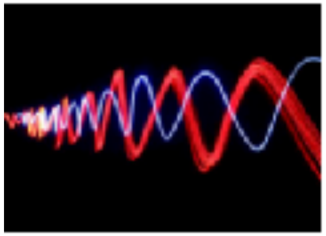


$$r_q = \frac{\sum_{j=1}^{q} \lambda_j}{\sum_{j=1}^{p} \lambda_j}$$



- an image can be represented in many ways
- most common format is a matrix of pixels
  - each "pixel" is BGR(A)
- used for capture and display





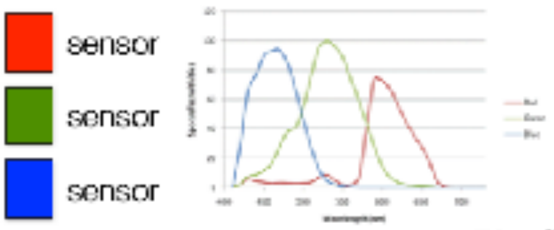


Image Representation, Features

Problem: need to represent image as table data

- need a compact representation

1	4	2	5	6	9
1	4	2	5	5	9
1	4	2	8	8	7
3	4	3	9	9	8
1	0	2	7	7	9
1	4	3	9	8	6
2	4	2	8	7	9

Image Representation, Features

Problem: need to represent image as table data

- need a compact representation

Solution: row concatenation (also, vectorizing)

Row 1	1	4	2	5	6	9	1	4	2	5	5	9	1	4	2	8	8	7	3
Row 2	1	4	2	8	8	7	3	4	3	9	9	8	1	4	2	5	5	9	1
...																			
Row N	9	4	6	8	8	7	4	1	3	9	2	1	1	5	2	1	5	9	1

Dimension Reduction with Images

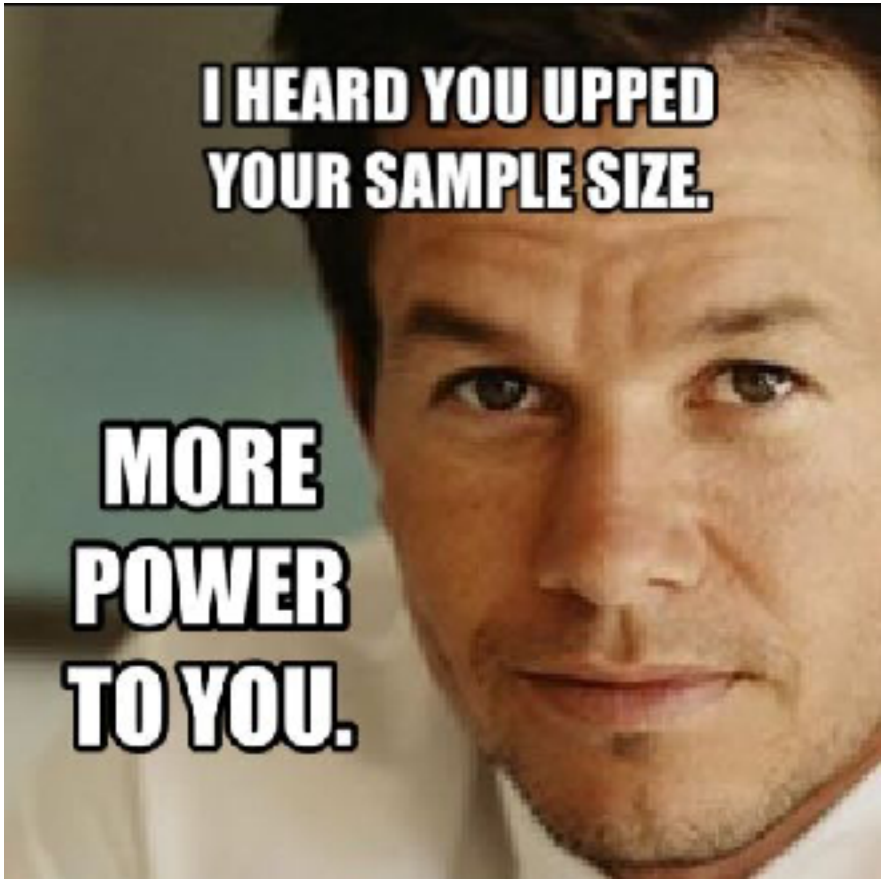
Demo

“Finish”

Images Representation in PCA and Randomized PCA

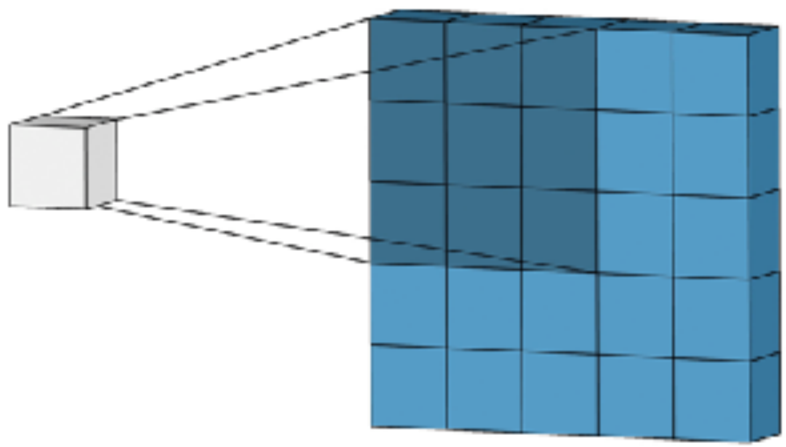
`04.Dimension Reduction and Images.ipynb`

Features of Images



Extracting Features: Convolution

- For images:
  - kernel (matrix of values)
  - slide kernel across image,pixel by pixel
  - multiply and accumulate



This Example:
3x3 Kernel (dark)
Ignoring edges of input
Input Image is 5x5
Output is then 3x3

https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Convolution

$$\sum \left(\mathbf{I}\left[i \pm \frac{r}{2}, j \pm \frac{c}{2} \right] \odot \mathbf{k} \right) = \mathbf{O}[i,j]$$

output image at pixel $\mathbf{i,j}$

input image at r x c range of pixels centered in i,j

kernel of size, r x c usually $r{=}c$

0	0	0	0	0	0	0	0	0
0	1	2	3	4	12	9	8	0
0	5	2	3	4	12	9	8	0
0	5	2	1	4	10	9	8	0
0	7	2	1	4	12	7	8	0
0	7	2	1	4	14	9	8	0
0	5	2	3	4	12	7	8	0
0	5	2	1	4	12	9	8	0
0	0	0	0	0	0	0	0	0

input image, $\mathbf{I}$

1	2	1
2	4	2
1	2	1

kernel filter, $\mathbf{k}$ 3x3

20	21	36	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…

output image, $\mathbf{O}$

Convolution Examples

Blur

1	1	1
1	1	1
1	1	1

Vertical Edges

-1	0	1
-1	0	1
-1	0	1

Sharpen

0	-1	0
-1	5	-1
0	-1	0

Move Left by One Pixel

0	0	0
1	0	0
0	0	0

?

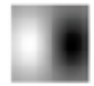
https://www.semanticscholar.org/paper/Automatic-Car-Damage-Recognition-using-Neural-Deijn/48f258246f7248a81458ddb14d9214e33759d7d7/figure/13

Common operations

- the gradient (2D derivative)

I

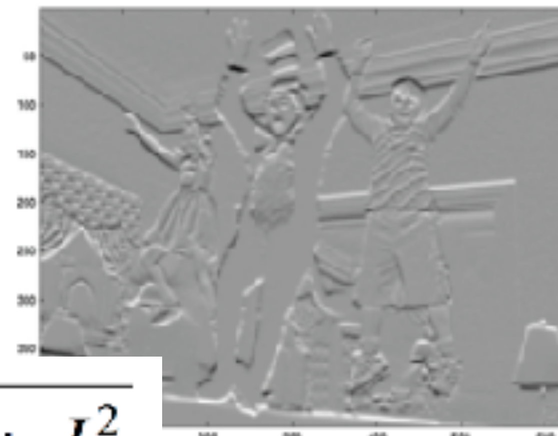
filter



filter

J_x

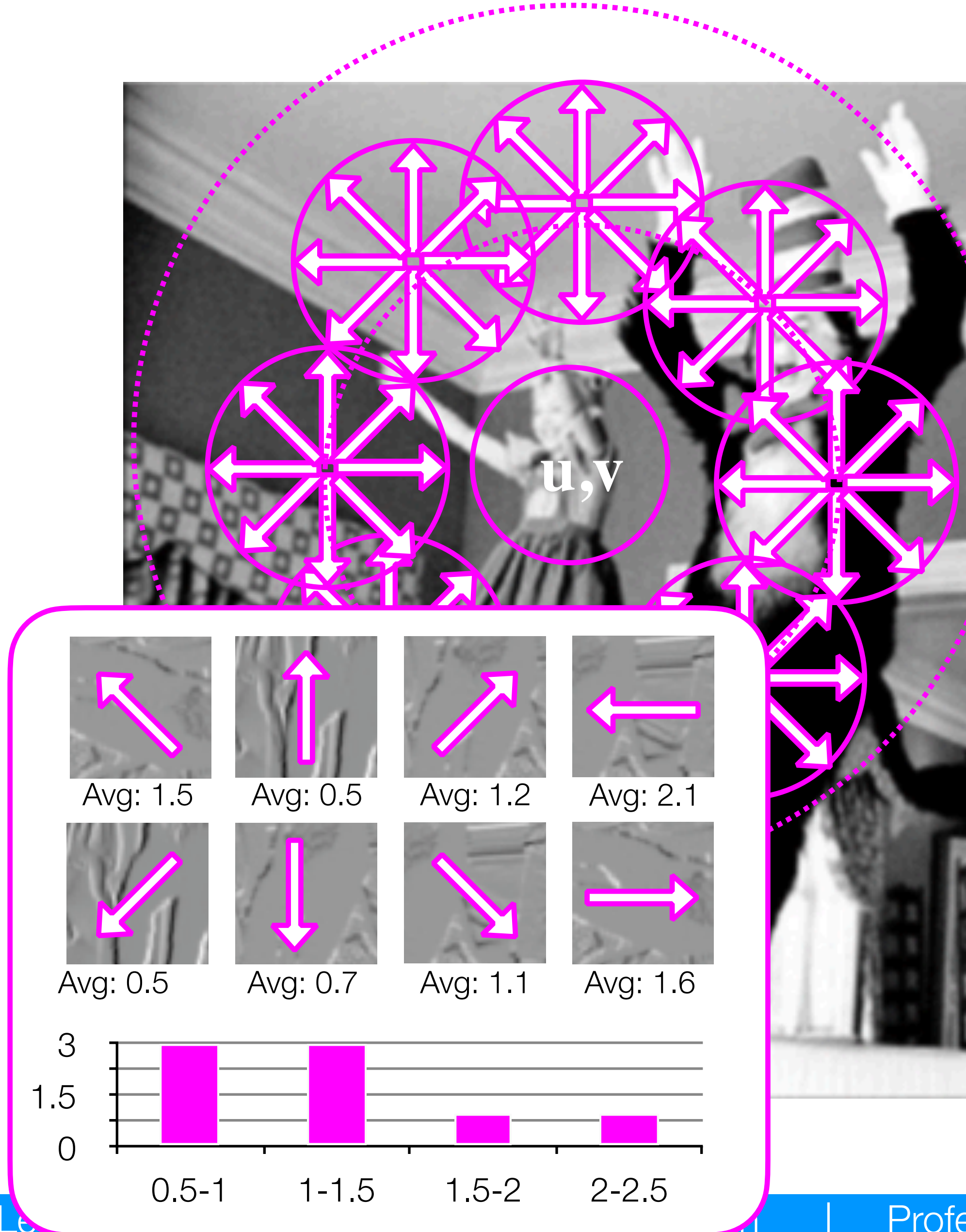
J_y



$$\|\nabla J\| = \sqrt{J_x^2 + J_y^2}$$

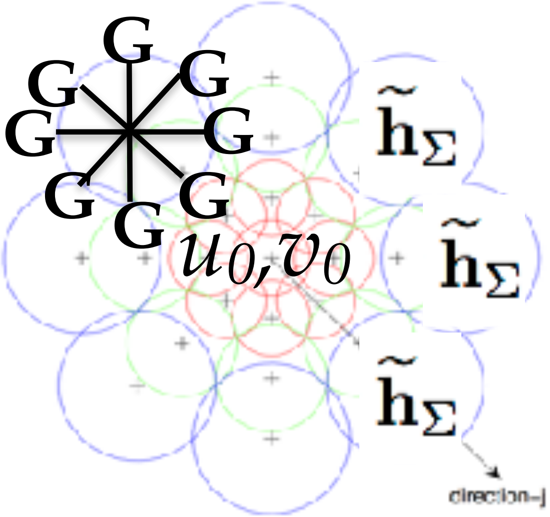
$$O=\tan^{-1}(J_x / J_y)$$

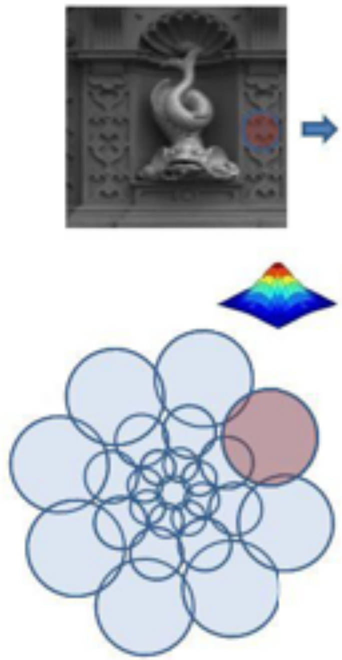
DAISY: same features, regardless of orientation



1. Select u,v pixel location in image
2. Take histogram of average gradient magnitudes in circle for each orientation
3. Select more circles in a ring
4. Go to next ring
5. For each circle on the ring, take another histogram
6. Repeat for more rings
7. Concat all histograms
8. Values become “feature” vector at that pixel location

Summary DAISY





Concatenate Histograms

$$
\begin{aligned}
\mathcal{D}(u_0, v_0) = \\
\Big[\ & \widetilde{\mathbf{h}}^{\top}_{\Sigma_1}(u_0, v_0), \\
& \widetilde{\mathbf{h}}^{\top}_{\Sigma_1}(\mathbf{l}_1(u_0, v_0, R_1)), \cdots, \widetilde{\mathbf{h}}^{\top}_{\Sigma_1}(\mathbf{l}_T(u_0, v_0, R_1)), \\
& \widetilde{\mathbf{h}}^{\top}_{\Sigma_2}(\mathbf{l}_1(u_0, v_0, R_2)), \cdots, \widetilde{\mathbf{h}}^{\top}_{\Sigma_2}(\mathbf{l}_T(u_0, v_0, R_2)),
\end{aligned}
$$

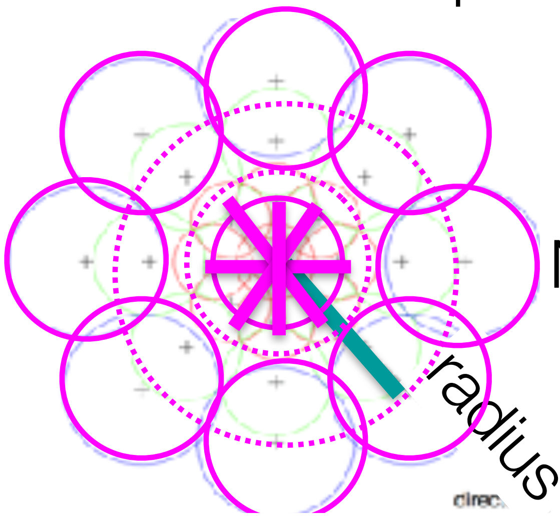
take normalized histogram of magnitudes

$$\widetilde{\mathbf{h}}_\Sigma(u, v) = \left\| \left[\mathbf{G}_1^\Sigma(u, v), \ldots, \mathbf{G}_H^\Sigma(u, v)\right]^\top \right\|$$

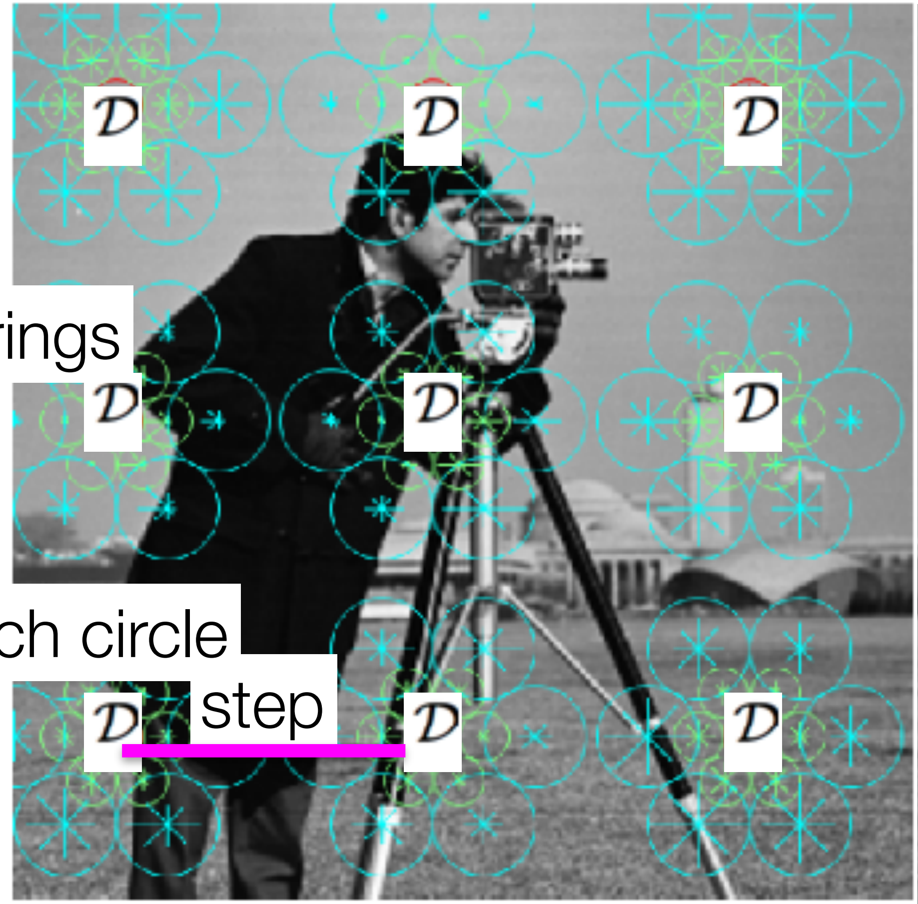
Tola et al. "Daisy: An efficient dense descriptor applied to wide-baseline stereo." Pattern Analysis and Machine Intelligence, IEEE

Free Parameters in DAISY

Num circles/per ring



Num rings



Num orientations within each circle

Num of bins in histogram

Params:
step, radius, num rings,
num histograms per ring,
orientations, bins per histogram

More Image Processing

Demo

Gradients

DAISY

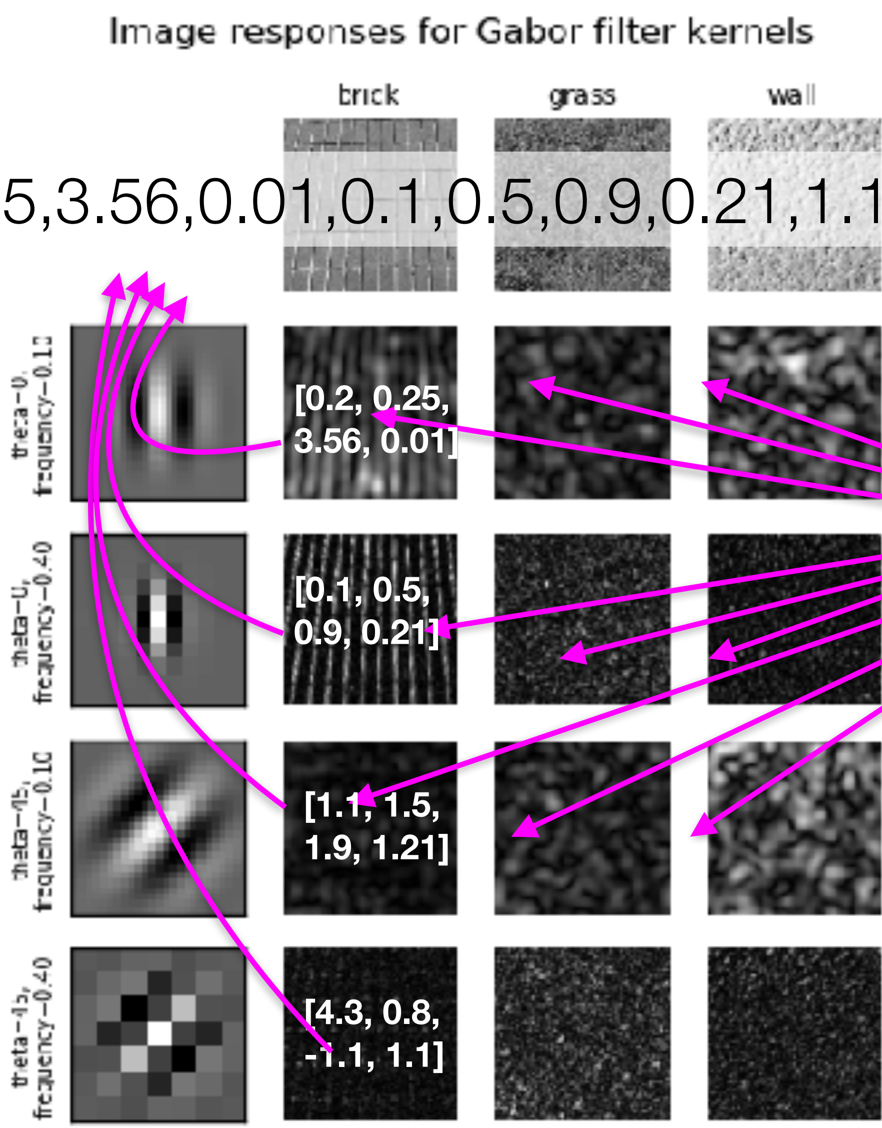
(if time)Gabor Filter Banks

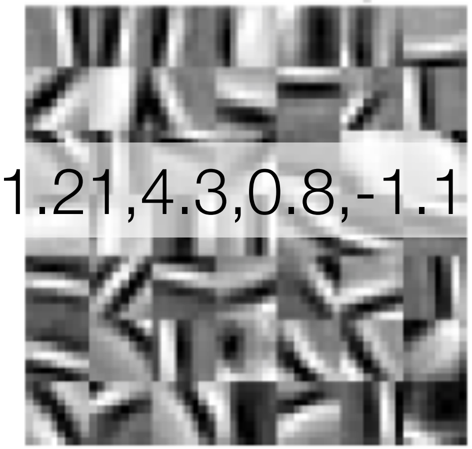
Other Tutorials:

http://scikit-image.org/docs/dev/auto_examples/

Common operations: Gabor filter Banks (if time)

- common used for texture classification





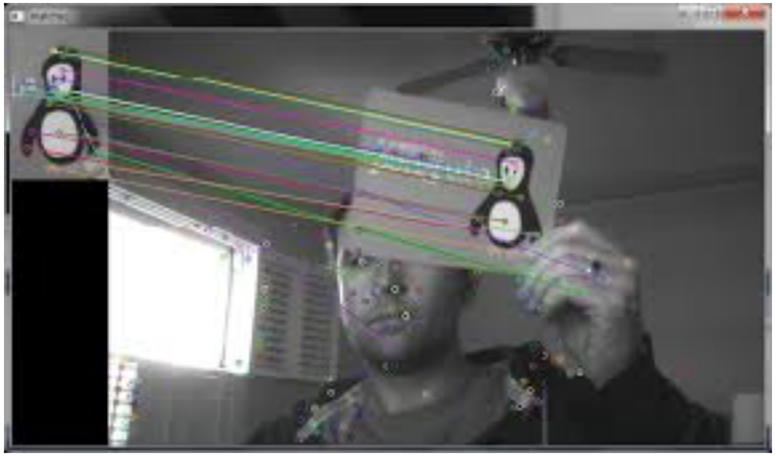
[0.2,0.25,3.56,0.01,0.1,0.5,0.9,0.21,1.1,1.5,1.9,1.21,4.3,0.8,-1.1,1.1]

mean
std
skew
kurtosis

Concatenate statistics of different filters into one feature vector

Matching versus Bag of Features

- Not a difference of vectors, but a percentage of matching points



- SURF, ORB, SIFT, DAISY

Feature Matching

Matching test image to source dataset

1. Choose src image from dataset
2. Take keypoints of src image
3. Take keypoints of test image
4. For each kp in src:
    1. Match with closest kp in test
    2. *How to define match?*
5. Count number of matches between images
6. Determine if src and test are similar based on number of matches
7. Repeat for new src image in dataset
8. Once all images measured, choose best match as the target for the test image

Scikit-image Implementation

match_descriptors

`skimage.feature.match_descriptors(descriptors1, descriptors2, metric=None, p=2, max_distance=inf, cross_check=True, max_ratio=1.0)` [source]

Brute-force matching of descriptors.

For each descriptor in the first set this matcher finds the closest descriptor in the second set (and vice-versa in the case of enabled cross-checking).

Town Hall for Lab 2, Images

- **Quiz is live**: Image Processing!
- **Next Time**: Logistic Regression

Supplemental Slides

Peruse these at your own leisure!

These slides might assist you as additional visual aides

Slides courtesy of Tan, Steinbach, Kumar

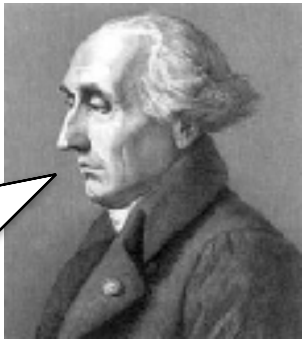
Introduction to Data Mining

Dimensionality Reduction: LDA

- PCA tell us variance explained by the data in different directions, but it ignores class labels
- Is there a way to find “components” that will help with **discriminate** between the classes?

$$\arg\max_{comp.} \frac{\sum \text{differences between classes}}{\sum \text{variance within classes}}$$

- called Fisher’s discriminant
- …but we need to solve this using using *Lagrange multipliers* and gradient-based optimization
- which we haven’t covered yet



I invented Lagrange multipliers… and …*nothing impresses me…*

Dimensionality Reduction: LDA versus QDA

$$\underset{comp.}{\arg\max} \frac{\sum \text{differences between classes}}{\sum \text{variance within classes}}$$

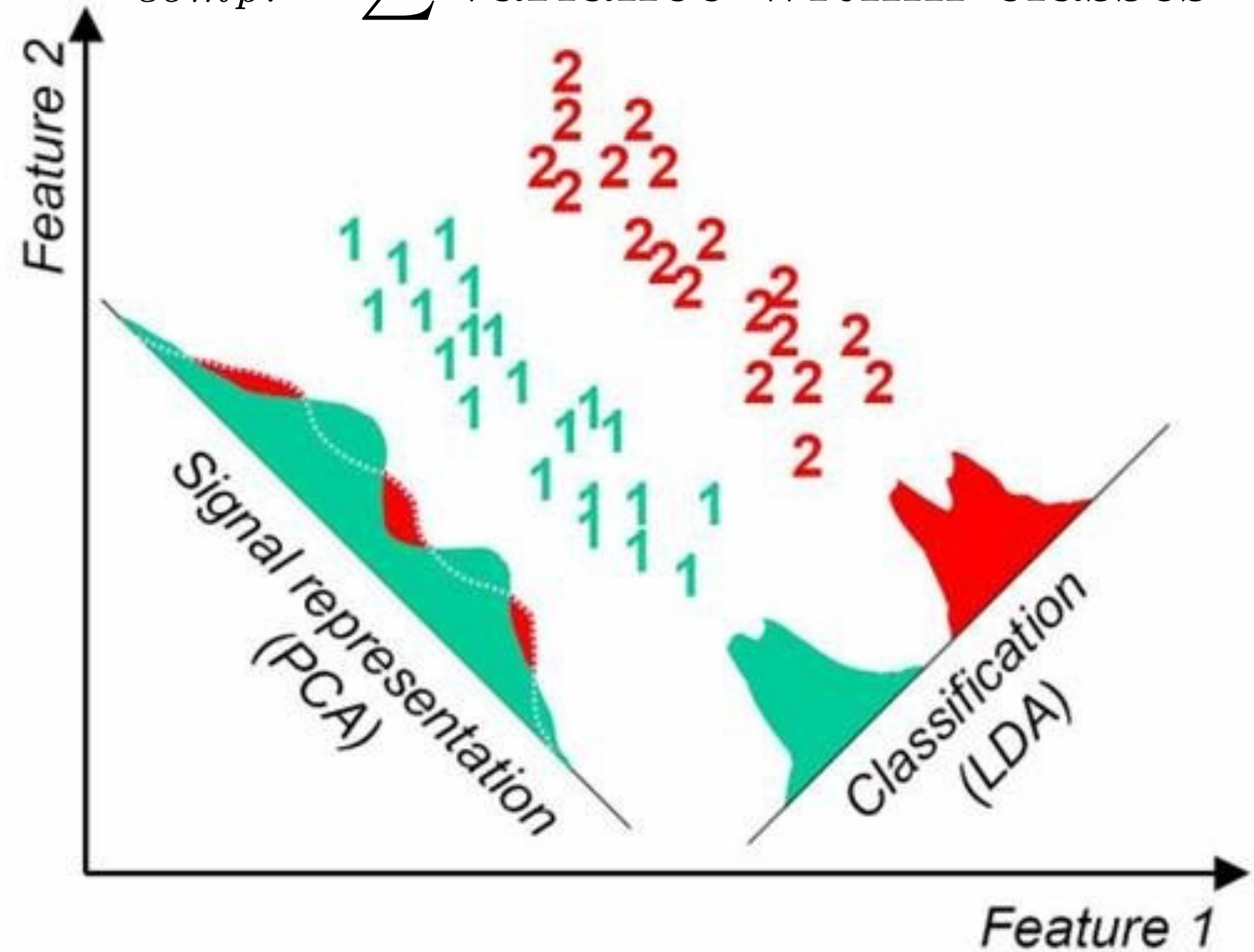
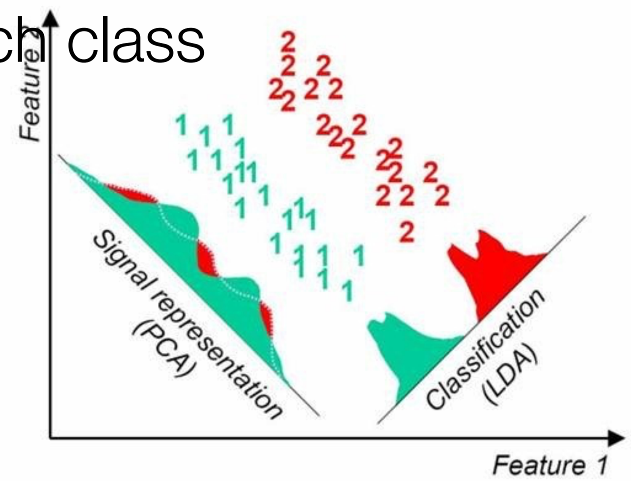


image source: 不想飞翔，不是因为没有翅膀，而是失去了梦想

Dimensionality Reduction: LDA versus QDA

$$\underset{comp.}{\arg\max} \frac{\sum \text{differences between classes}}{\sum \text{variance within classes}}$$

- "*differences between classes*" is calculated by trying to separate the **mean value** of each **feature** in each **class**
- Linear discriminant analysis:
  - assume the covariance in each class is the same
- Quadrature discriminant analysis:
  - estimate the covariance for each class



Self Test ML2b.2

LDA only allows as many components as there are unique classes in a dataset.

A. True
B. False

- Need more help with the PCA algorithm (and python)?
  - check out Sebastian Raschka's notebooks:

http://nbviewer.ipython.org/github/rasbt/pattern_classification/blob/master/dimensionality_reduction/projection/principal_component_analysis.ipynb