Lecture Notes for

Machine Learning in Python

⟦👨‍🏫, 👨‍💻, 🐍, 👨‍🔬⟧

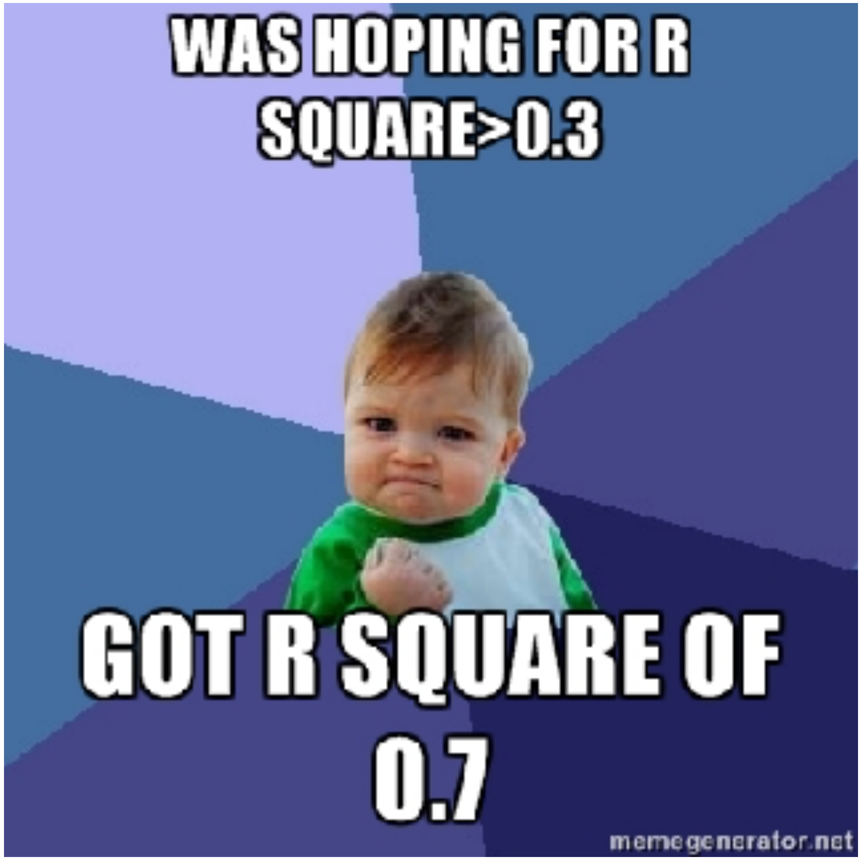
Professor Eric Larson

Visualization and Dimensionality Reduction

Class Logistics and Agenda

- Welcome back!!!!
- Dimensionality Reduction
  - PCA
  - Sampling
  - Kernel Methods

Dimensionality Reduction: PCA



Curse of Dimensionality

- When dimensionality increases, data becomes increasingly sparse in the space that it occupies

- Definitions of density and distance between points, which is critical for clustering and outlier detection, become less meaningful

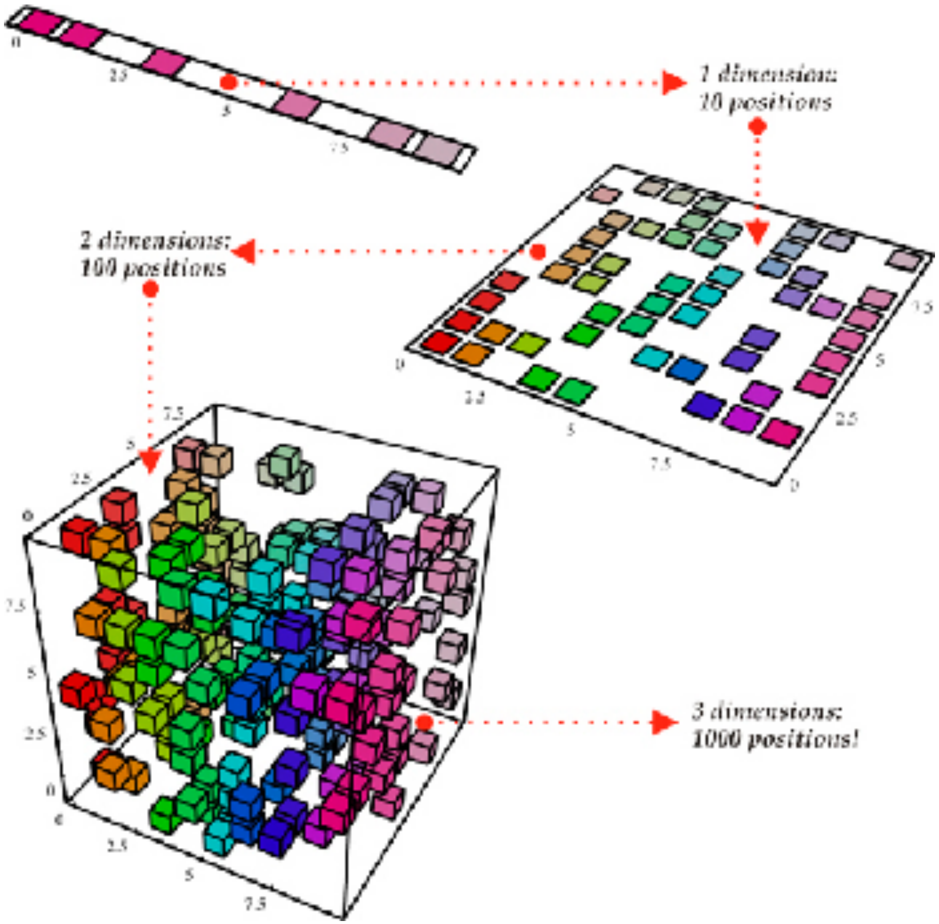


image source: http://www.iro.umontreal.ca/~bengioy/yoshua_en/research_files/CurseDimensionality.jpg

Dimensionality Reduction

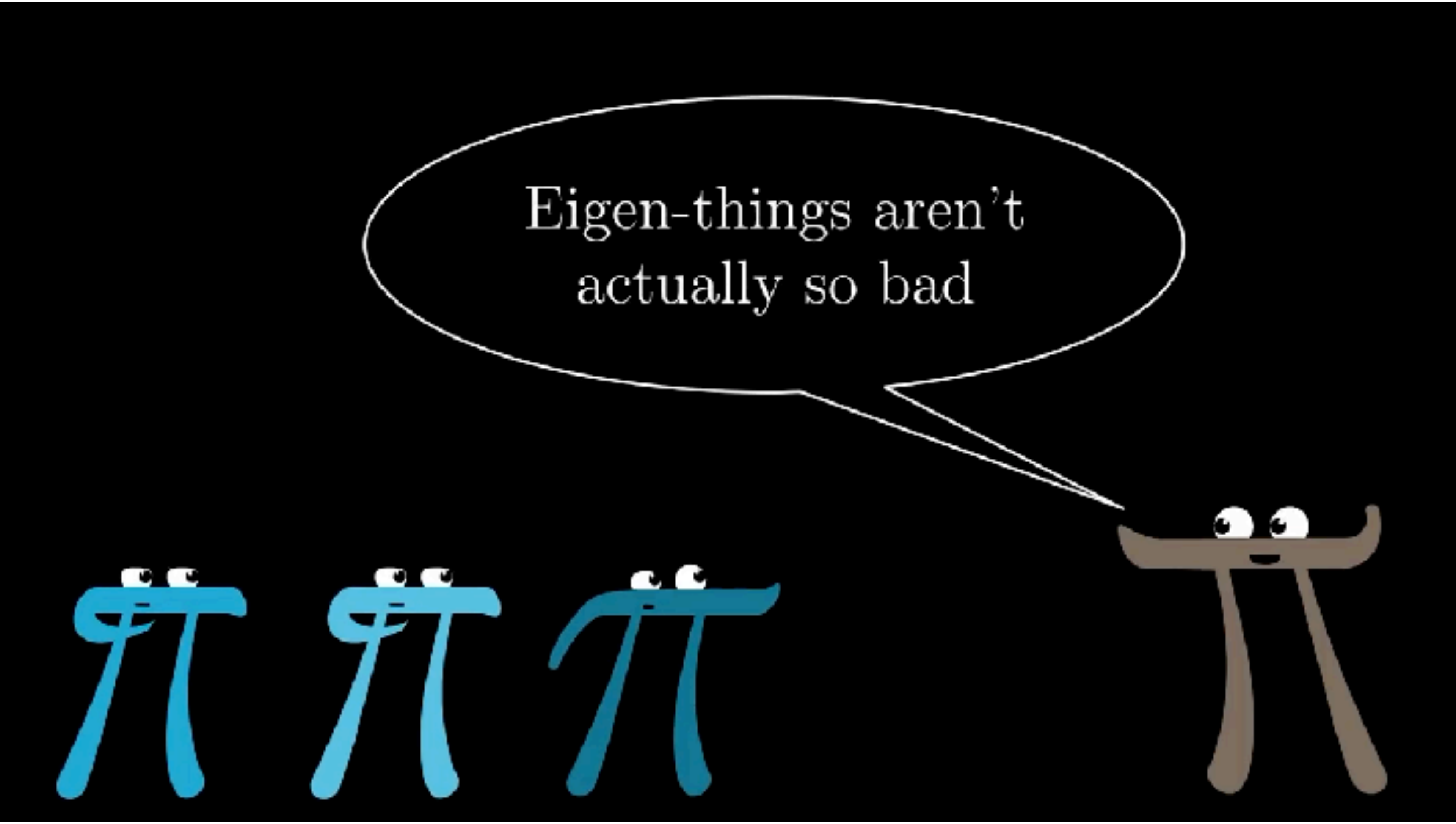
- Purpose:
  - Avoid curse of dimensionality
  - Select subsets of independent features
  - Reduce amount of time and memory required by data mining algorithms
  - Allow data to be more easily visualized
  - May help to eliminate irrelevant features or reduce noise

- Techniques
  - Principle Component Analysis
  - Non-linear PCA
  - Stochastic Neighbor Embedding

I invented PCA…
and *Social Darwinism*

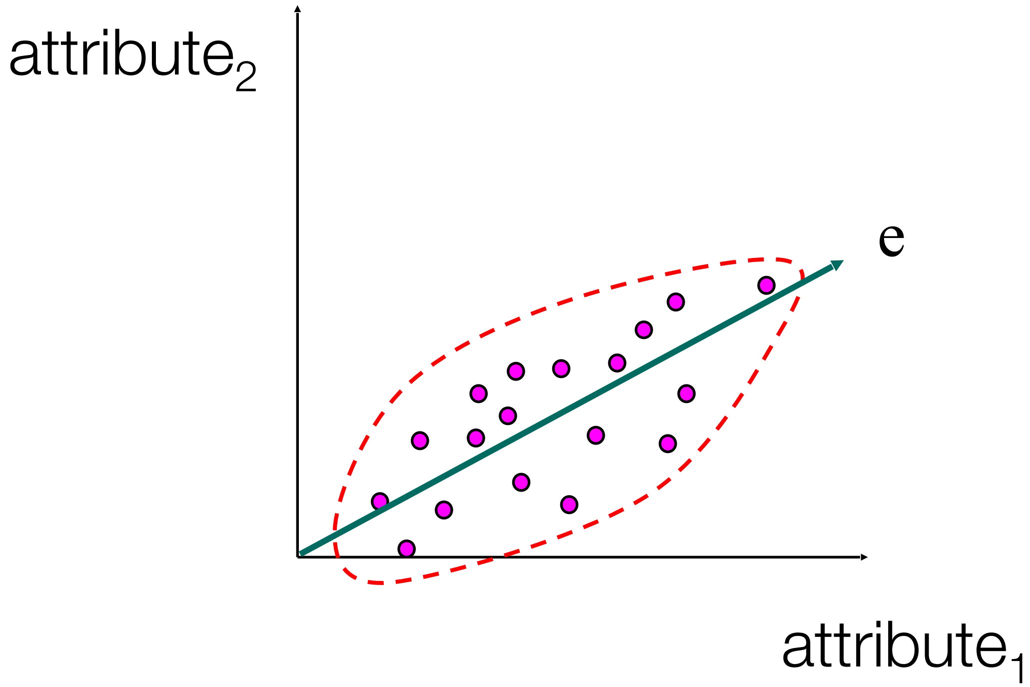
Aside: Eigen Vectors are your friend!

- **Three Blue One Brown:** https://www.youtube.com/watch?v=PFDu9oVAE-g



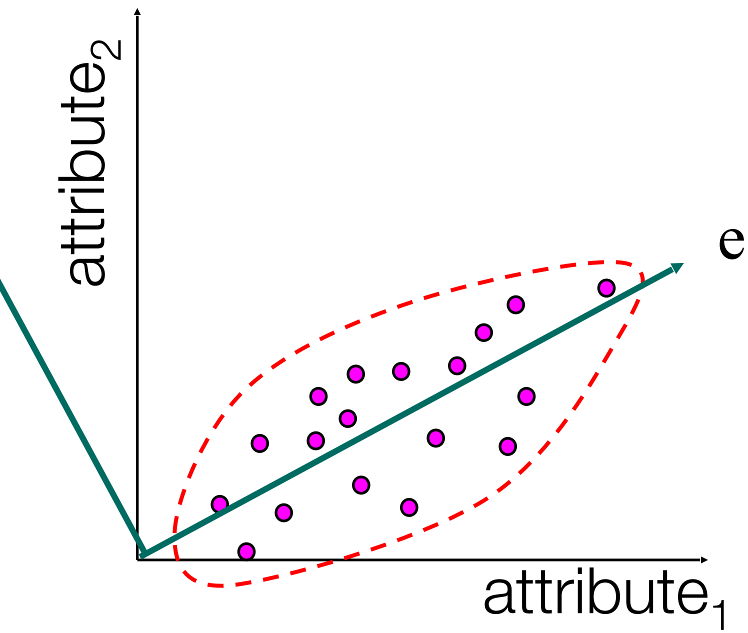
Dimensionality Reduction: PCA

- Goal is to find a projection that captures the largest amount of variation in data



Dimensionality Reduction: PCA

- Find the **eigenvectors** of the **covariance** matrix
- keep the "k" **largest** eigenvectors



E1	E2
0.85	0.85
0.52	-0.52

covariance

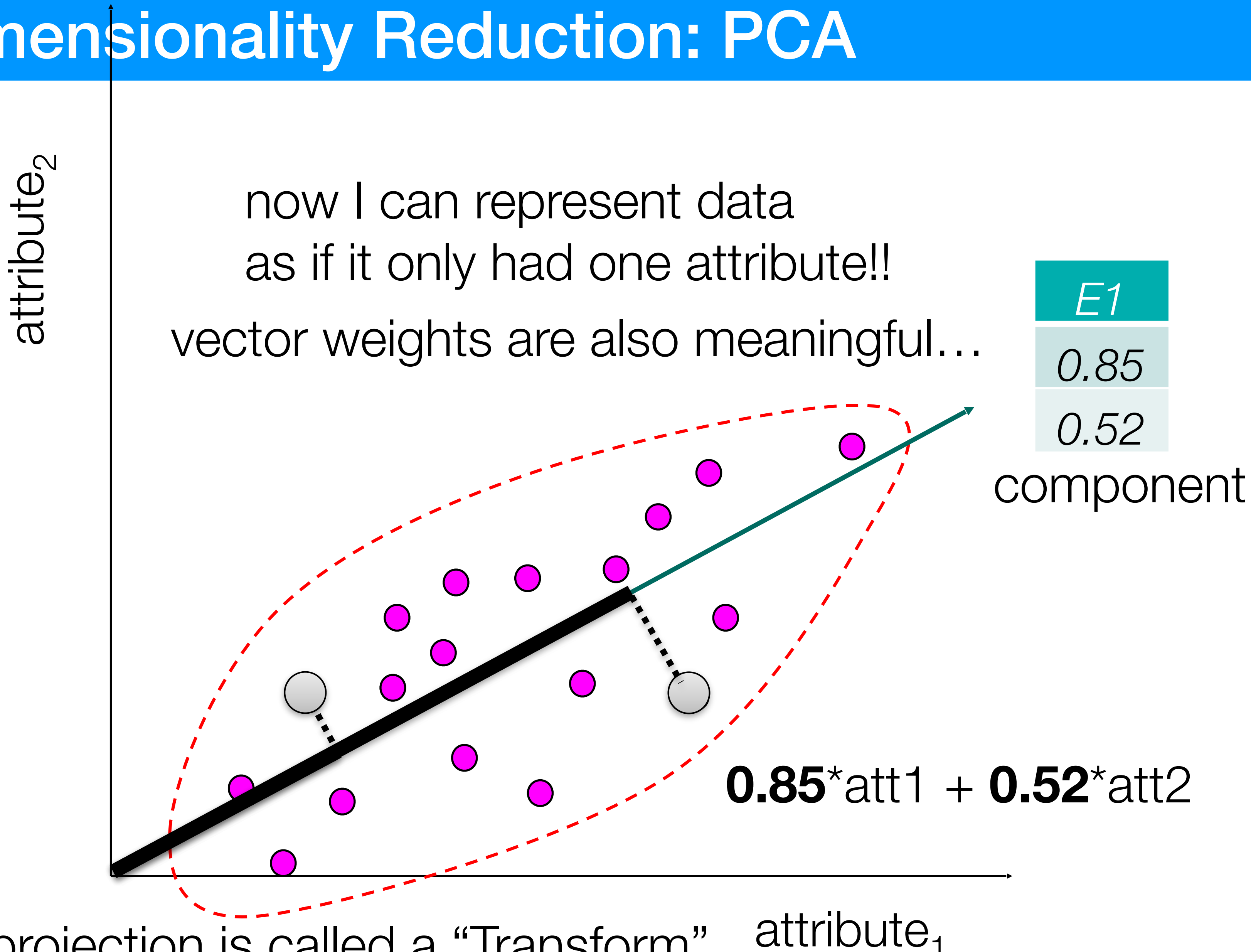
37.1	-6.7
-6.7	43.9

	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

Dimensionality Reduction: PCA

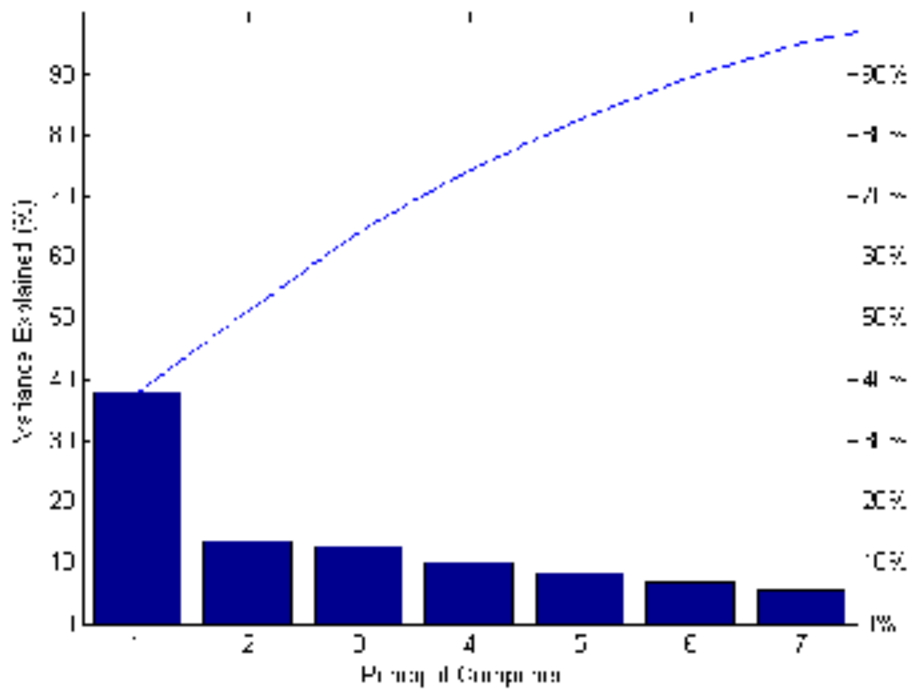


This projection is called a “Transform”
known as the **Karhunen-Loève Transform**

Explained Variance

- Each principle component explains a certain amount of variation in the data.
- This explained variation is embedded in the eigenvalues for each eigenvector

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



Dimensionality Reduction: PCA

- Genetic profiles distilled to 2 components

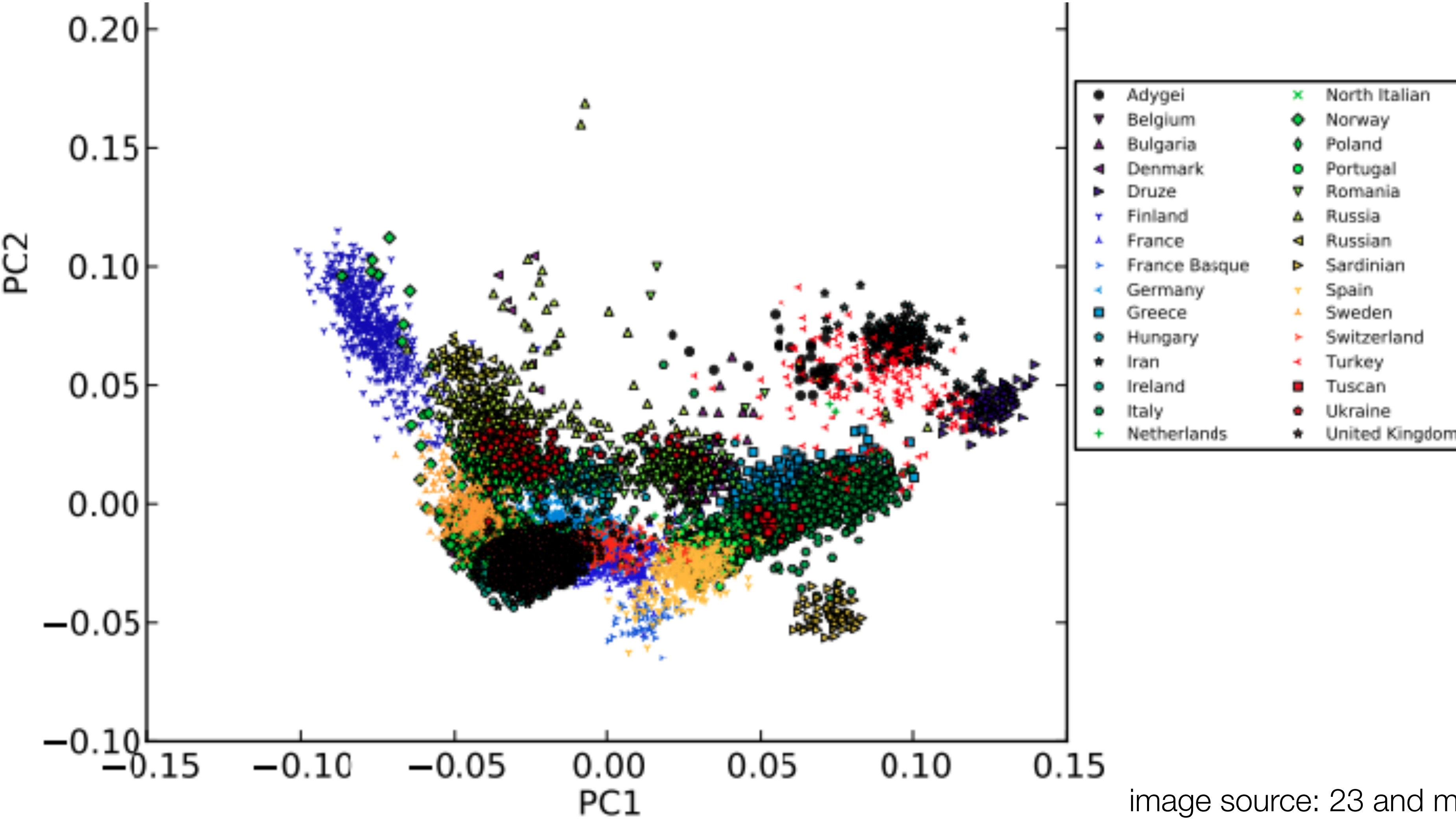


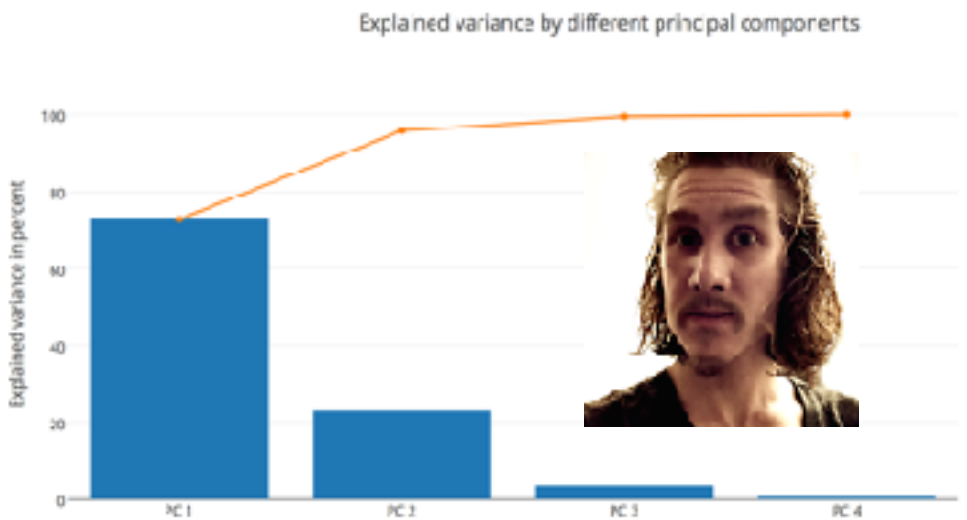
image source: 23 and me

Dimensionality Reduction: PCA

- 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

Or check out PCA for dummies:
https://georgemdallas.wordpress.com/2013/10/30/principal-component-analysis-4-dummies-eigenvectors-eigenvalues-and-dimension-reduction/



Dimension Reduction

Demo

`04.Dimension Reduction and Images.ipynb`

PCA

biplots

Other Tutorials:

http://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_vs_lda.html#example-decomposition-plot-pca-vs-lda-py

http://nbviewer.ipython.org/github/ogrisel/notebooks/blob/master/Labeled%20Faces%20in%20the%20Wild%20recognition.ipynb

Self Test ML2b.1

Principal Components Analysis works well for categorical data by design.

A. True
B. False
C. It doesn’t but people do it anyway

Dimensionality Reduction: Randomized PCA

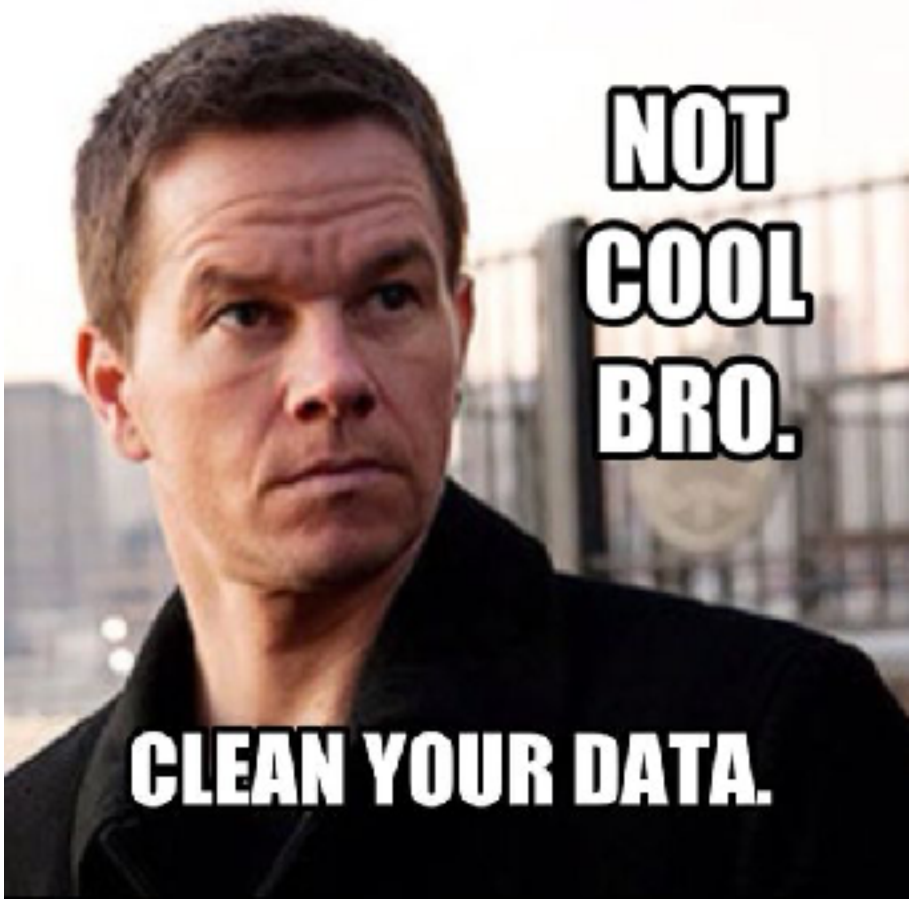
- **Problem**: PCA on all that data can take a while to compute
  - What if the number of instances is gigantic?
  - What if the number of dimensions is gigantic?

- What if we partially construct the covariance matrix with a lower rank matrix?
  - By **randomly sampling from the dataset and projecting**, we can get something representative of covariance matrix, but with **lower rank**
  - Gives a matrix with typically good enough precision of actual eigenvectors

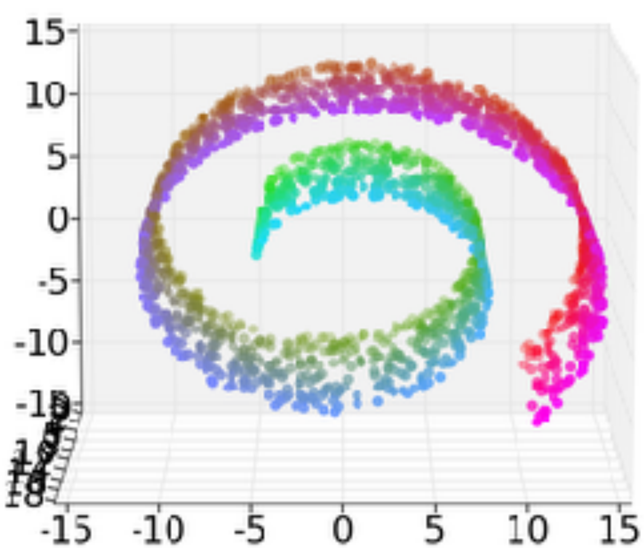
$$\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\| \leq \left[1 + 11\sqrt{k+p} \cdot \sqrt{\min\{m,n\}}\right] \sigma_{k+1}$$

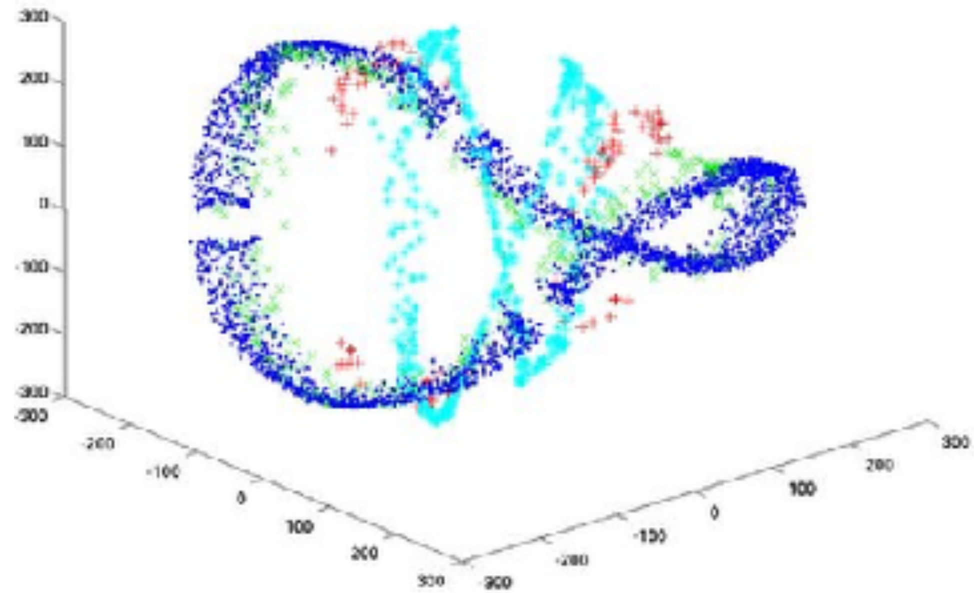
source: Halko, et al., (2009) Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions. https://arxiv.org/pdf/0909.4061.pdf

Non-linear Dimensionality Reduction



Dimensionality Reduction: non-linear

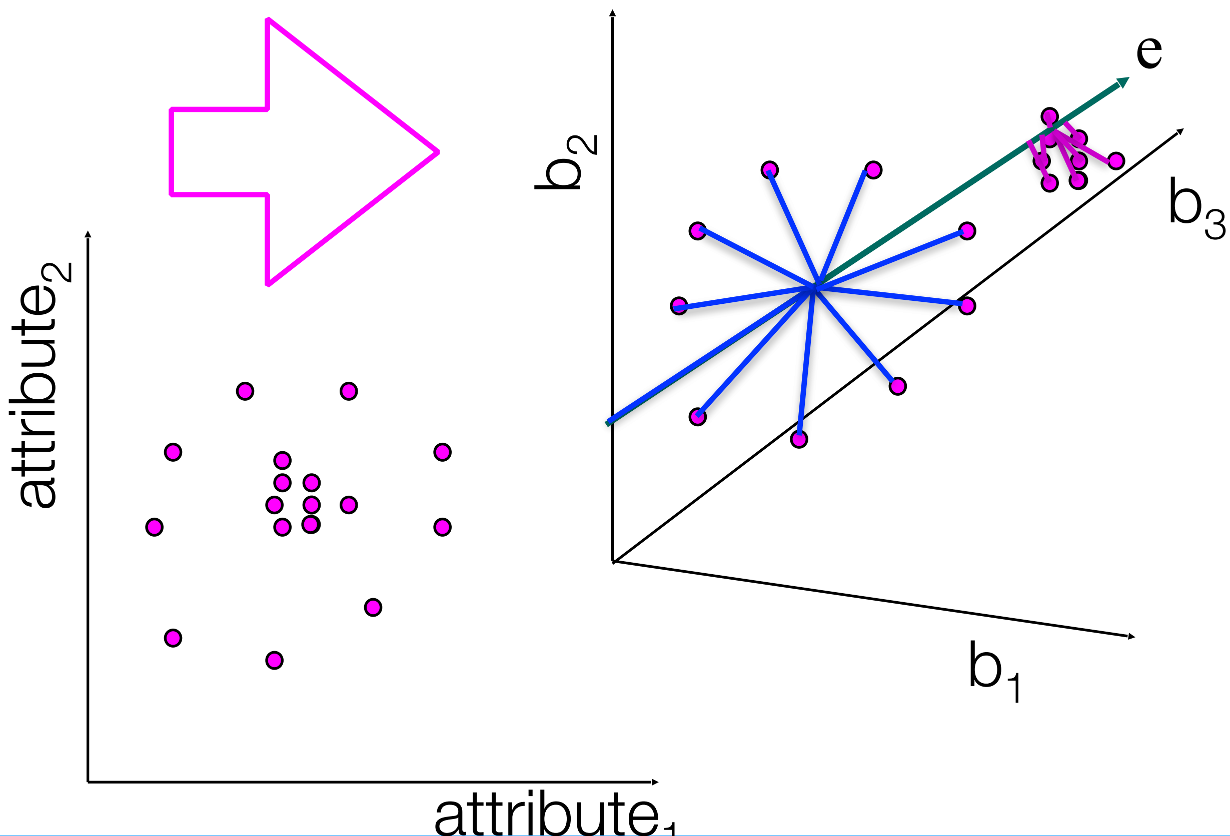




- Sometimes a **linear transform** is not enough
- A powerful non-linear transform has seen a resurgence in past decade: **kernel PCA**

Kernel PCA

- Estimate Covariance in higher dimensional space
- Get eigen vectors from nonlinear dot product
- Projecting onto these can be understood as principle components from a higher dimensional space



$\phi(A1)\cdot\phi(A1)$	$\phi(A2)\cdot\phi(A1)$
$\phi(A2)\cdot\phi(A1)$	$\phi(A2)\cdot\phi(A2)$

	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*

Kernel PCA

- **Key insight**: don’t need to know the actual principle components, just the projections

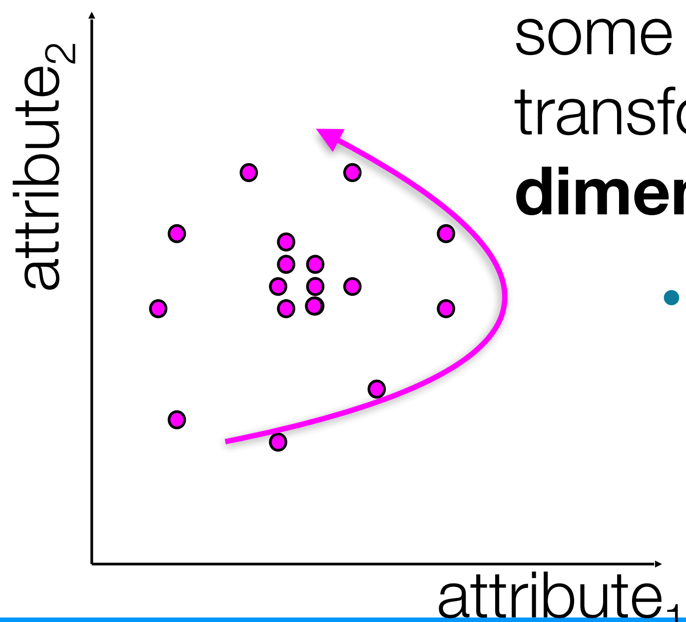
$\phi(A1)\cdot\phi(A1)$	$\phi(A2)\cdot\phi(A1)$
$\phi(A2)\cdot\phi(A1)$	$\phi(A2)\cdot\phi(A2)$

$$\boldsymbol{\kappa}(A1,A2) = \phi(A1)\cdot\phi(A2)$$

kernel: defines what the dot product is in higher dimensional space

	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

some kernels have corresponding transformations with **infinite dimensions**!!



- **Never need** eigen vectors of **full** covariance matrix, just how much the vectors co-vary in higher space!

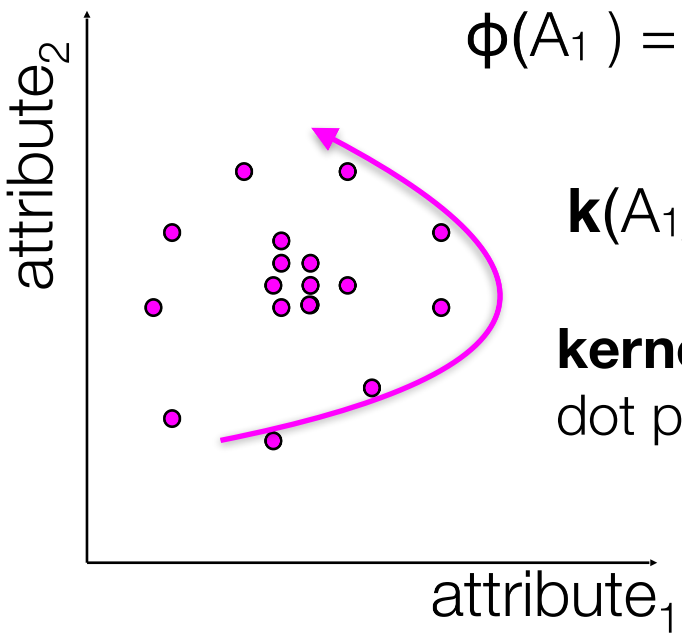
Kernel PCA

- **Key insight**: don't need to know the actual principle components, just the projections

$\phi(A1)\cdot\phi(A1)$	$\phi(A2)\cdot\phi(A1)$
$\phi(A2)\cdot\phi(A1)$	$\phi(A2)\cdot\phi(A2)$

$$A_1 = [a_1 \ a_2]^T$$

$$\phi(A_1) = [a_1 \quad a_2 \quad a_1 * a_2 \quad a_1^2 \quad a_1 * a_2^3 \cdots]^T$$



$$\mathbf{k}(A_1, A_2) = \exp(-\gamma \, || A_1 - A_2 ||^2)$$

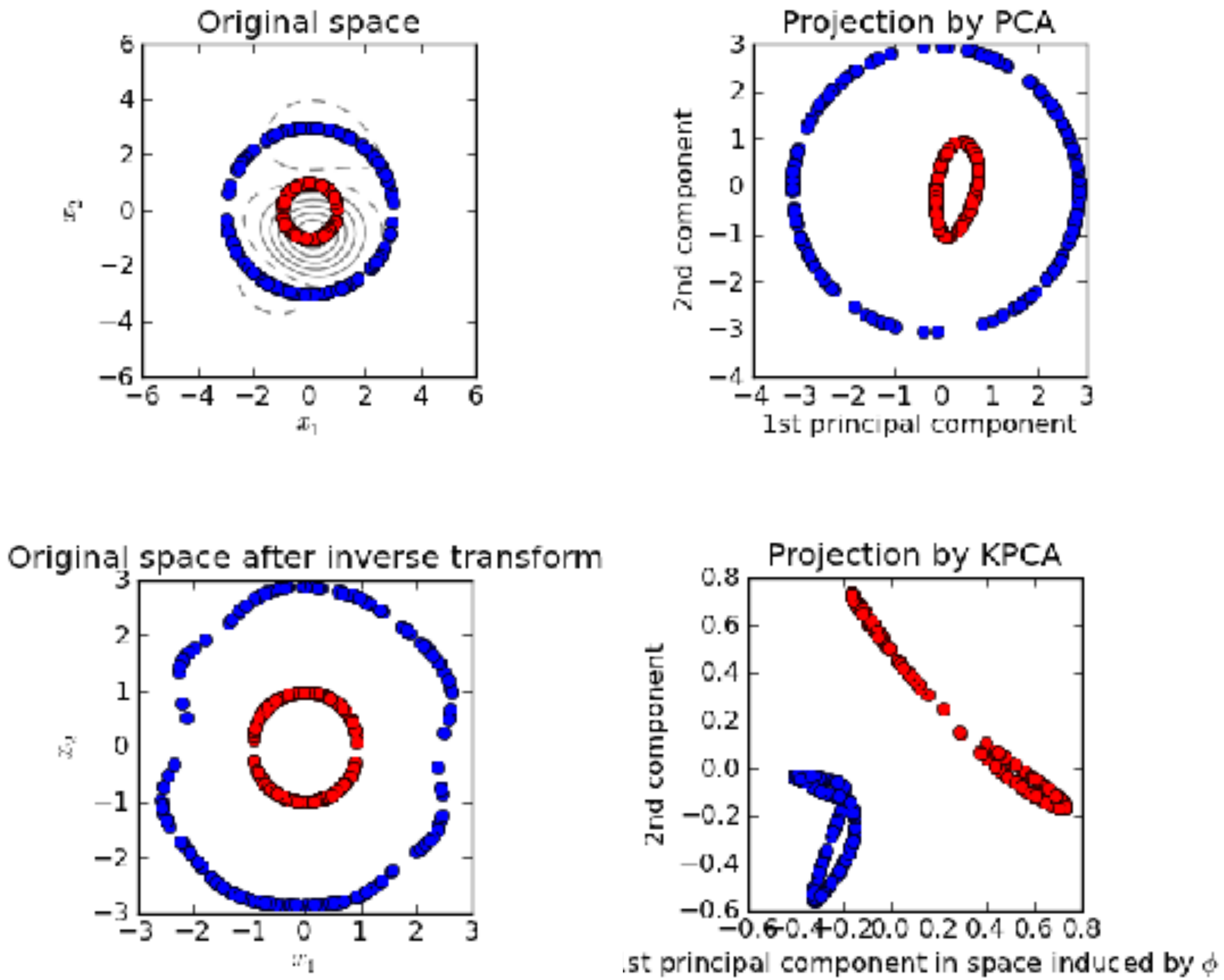
kernel: radial basis function (rbf)
dot product in higher dimensional space

	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

http://sebastianraschka.com/Articles/2014_kernel_pca.html#kernel-functions-and-the-kernel-trick

Kernel PCA

images: sklearn documentation



For Next Lecture

- Next Lecture:
  - Kernel Methods
  - Dimension Reduction Demo
  - Crash-course Image Feature Extraction

Lecture Notes for

Machine Learning in Python

⟦👨‍🏫👨‍💻🐍👨‍🔬⟧

Professor Eric Larson

Dimensionality Reduction and Images

Class Logistics and Agenda

- **Logistics**:
  - Next lab due soon!
- **Agenda**
  - Common Feature Extraction Methods for Images
  - Begin Town Hall

Last time…

E1
0.85
0.52

E2
0.85
-0.52

37.1	-6.7
-6.7	43.9

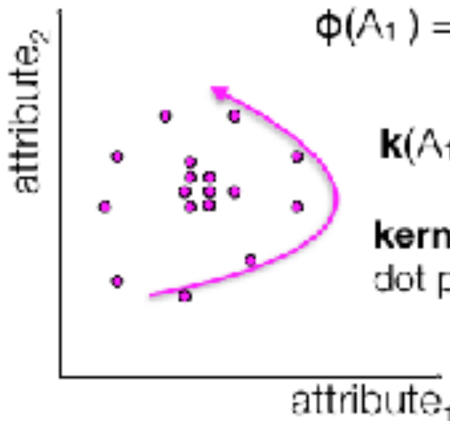
	A1	A2
1	66	33.6
2	54	26.6
3	69	23.3
4	73	28.1
5	61	43.1
6	62	25.6

	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

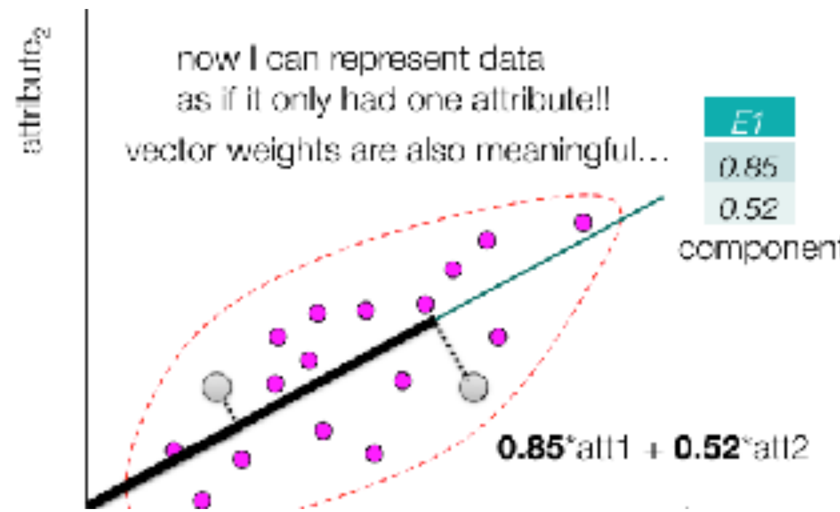
$A_1 = [a_1\ a_2]^T$

$\phi(A_1) = [a_1 \quad a_2 \quad a_1 a_2 \quad a_1^2 \quad a_1 a_2^3 \cdots]^T$



$\mathbf{k}(A_1, A_2) = \exp(-\gamma \| A_1 - A_2 \|^2)$

kernel: radial basis function (rbf)
dot product in higher dimensional space



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

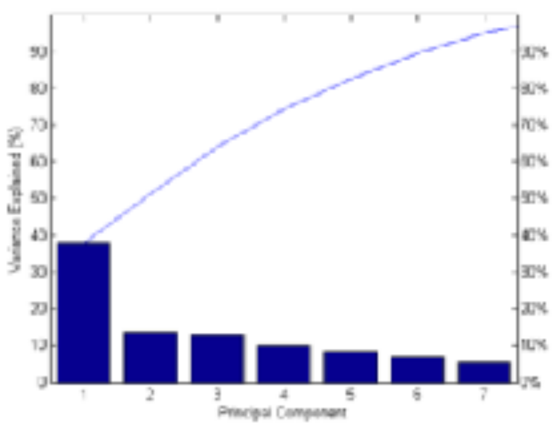
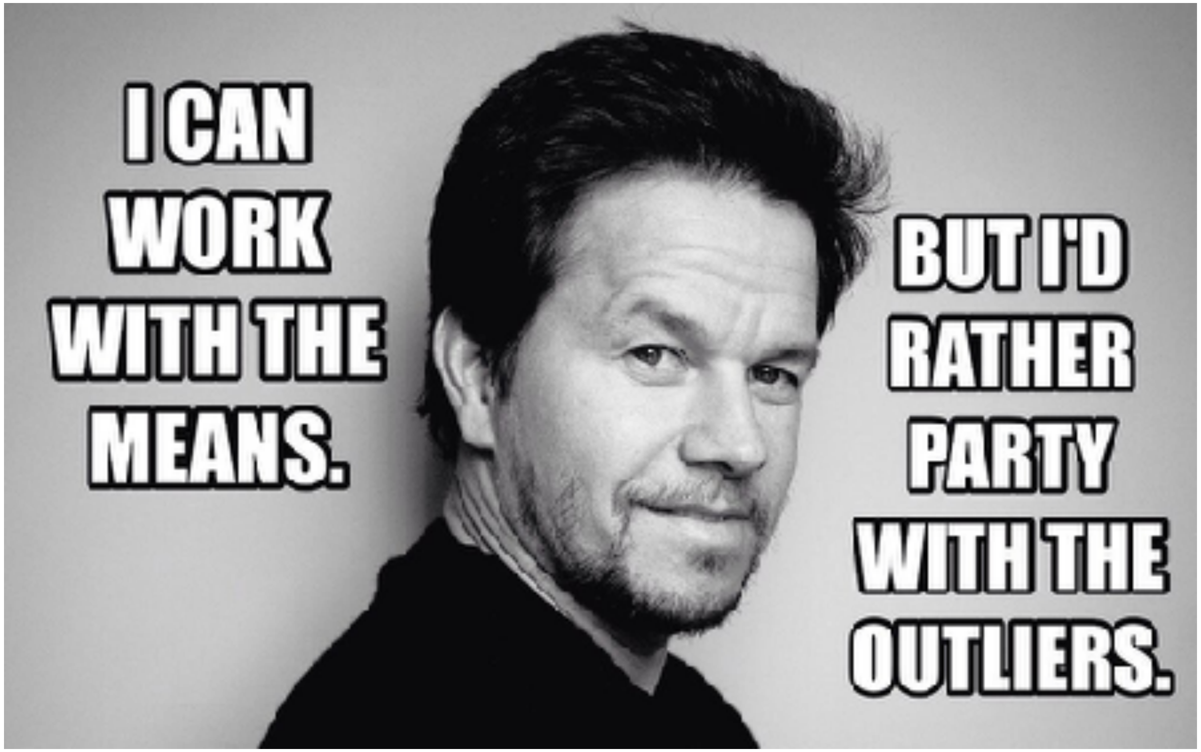
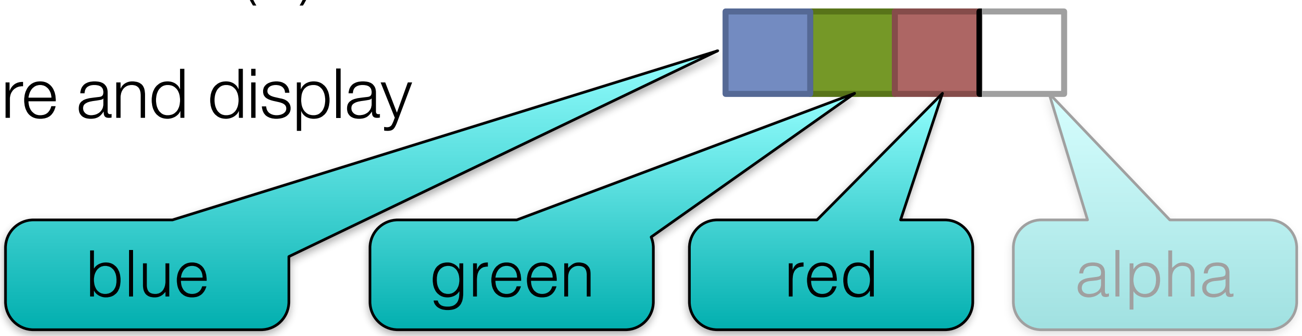


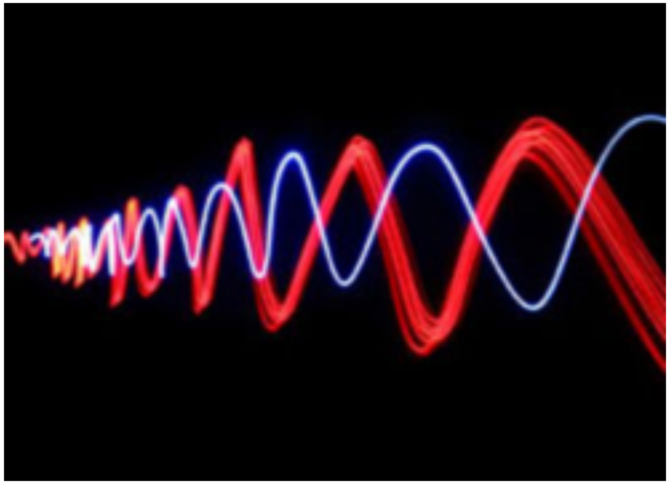
Image Processing and Representation

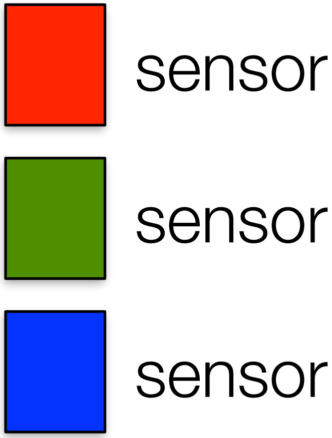


Images as data

- 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







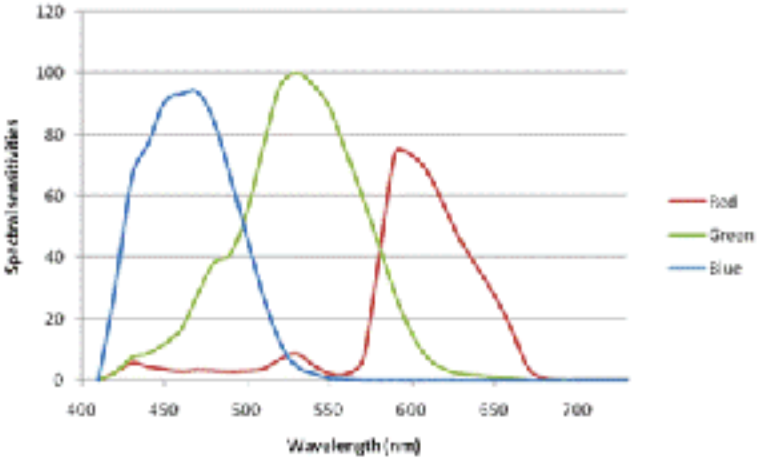


Image Representation

- need a compact representation

- **grayscale**

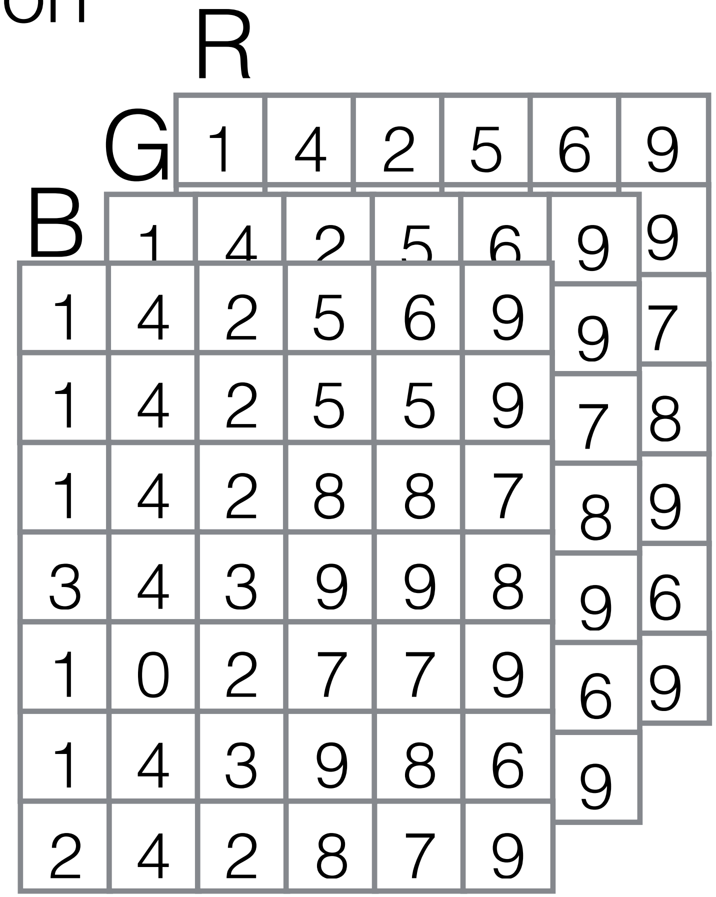
  $0.3*R+0.59*G+0.11*B$,

  “luminance”

gray

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

Numpy Matrix
image[rows, cols]



Numpy Matrix
image[rows, cols, channels]

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

Self test: 3a-1

- When vectorizing images into table data, each "feature column" corresponds to:
  - a. the value (color) of pixel
  - b. the spatial location of a pixel in the image
  - c. the size of the image
  - d. the spatial location and color channel of a pixel in an image

Dimension Reduction with Images

Demo

Images Representation
Randomized PCA
Kernel PCA

`04.Dimension Reduction and Images.ipynb`

Features of Images

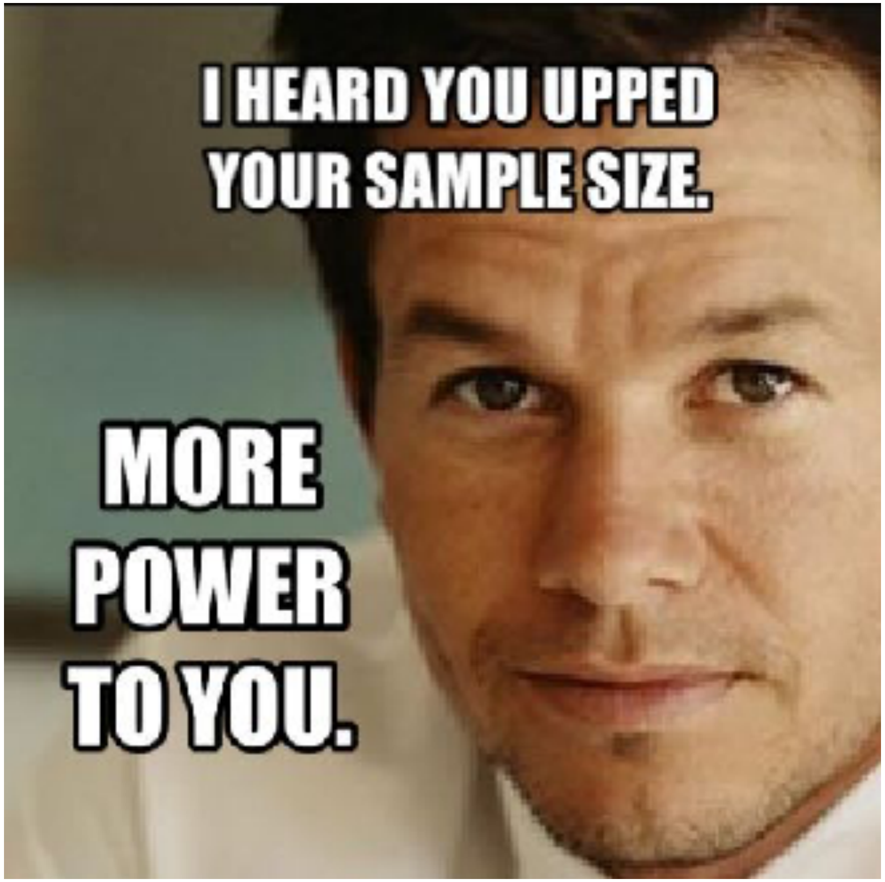
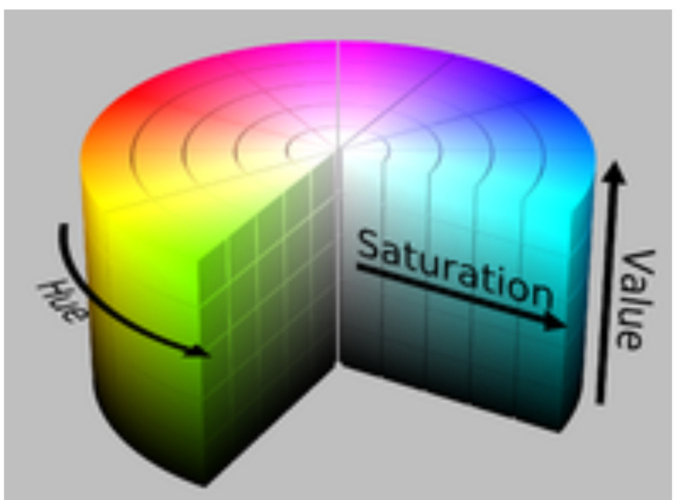
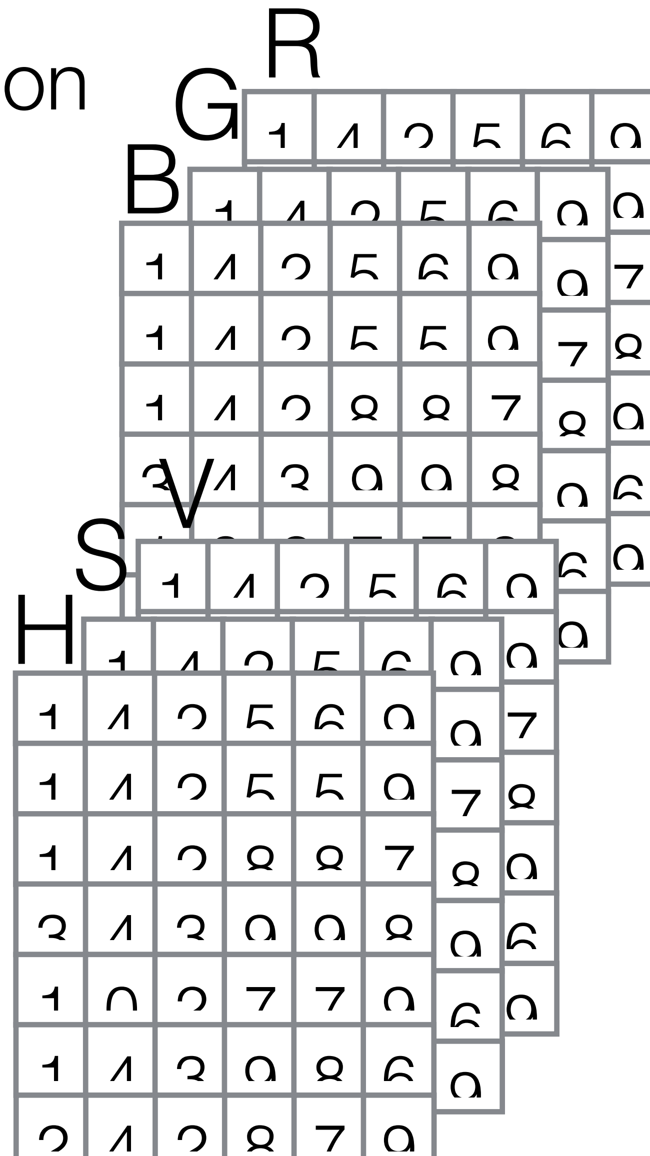


Image Representation

- need a compact representation

- **hsv**
  - what we perceive as color (ish)
    - hue: the color value
    - saturation: the richness of the color relative to brightness
    - value: the gray level intensity





Convolution

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

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

input image

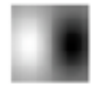
output image

Common operations

- the gradient (2D derivative)

I

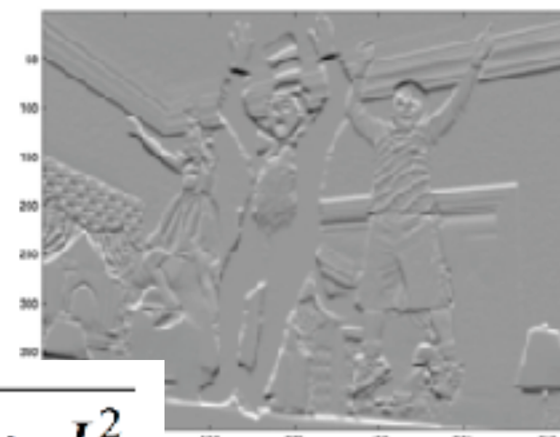
filter



filter

J_x

J_y

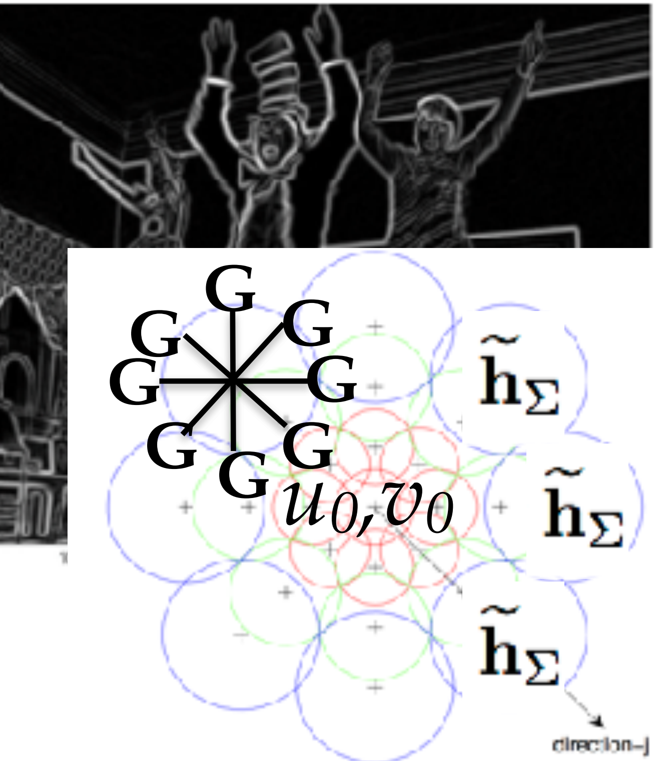


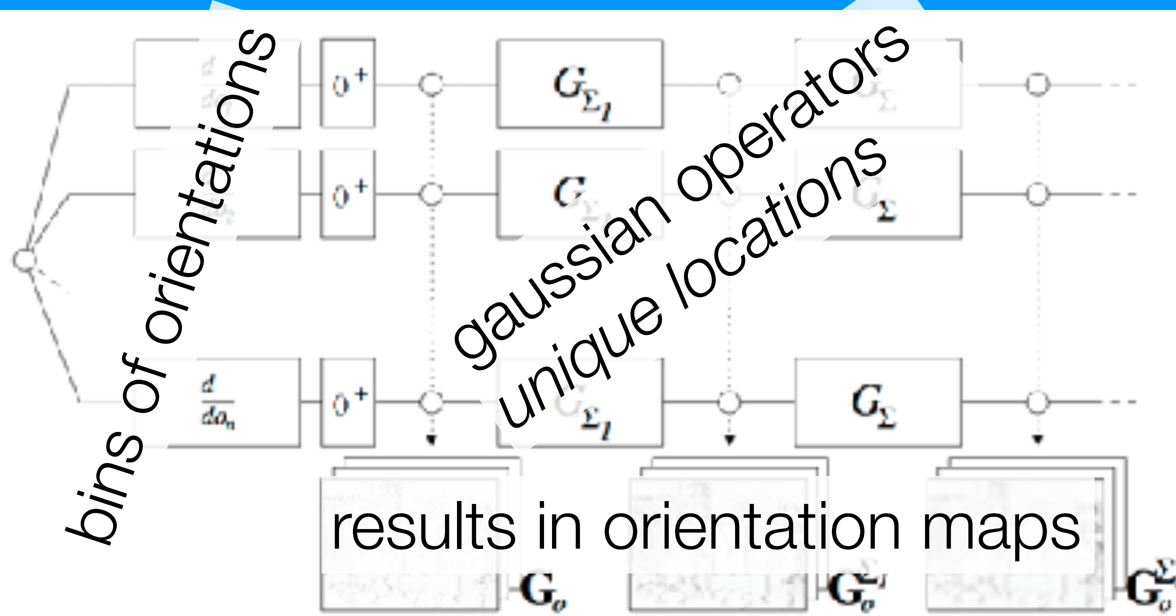
$$\|\nabla J\| = \sqrt{J_x^2 + J_y^2}$$

$$O = \tan^{-1}(Jx / Jy)$$

images: Jianbo Shi, Upenn

Common operations: DAISY





take normalized histogram at point u,v

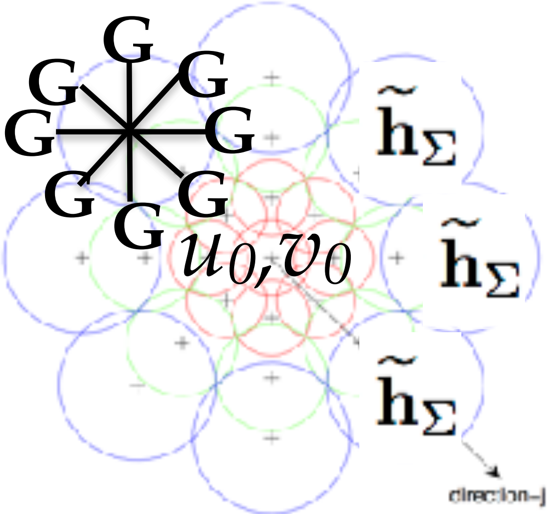
$$\tilde{\mathbf{h}}_\Sigma(u,v) = \left\|[\mathbf{G}_1^\Sigma(u,v),\ldots,\mathbf{G}_H^\Sigma(u,v)]^\top\right\|$$

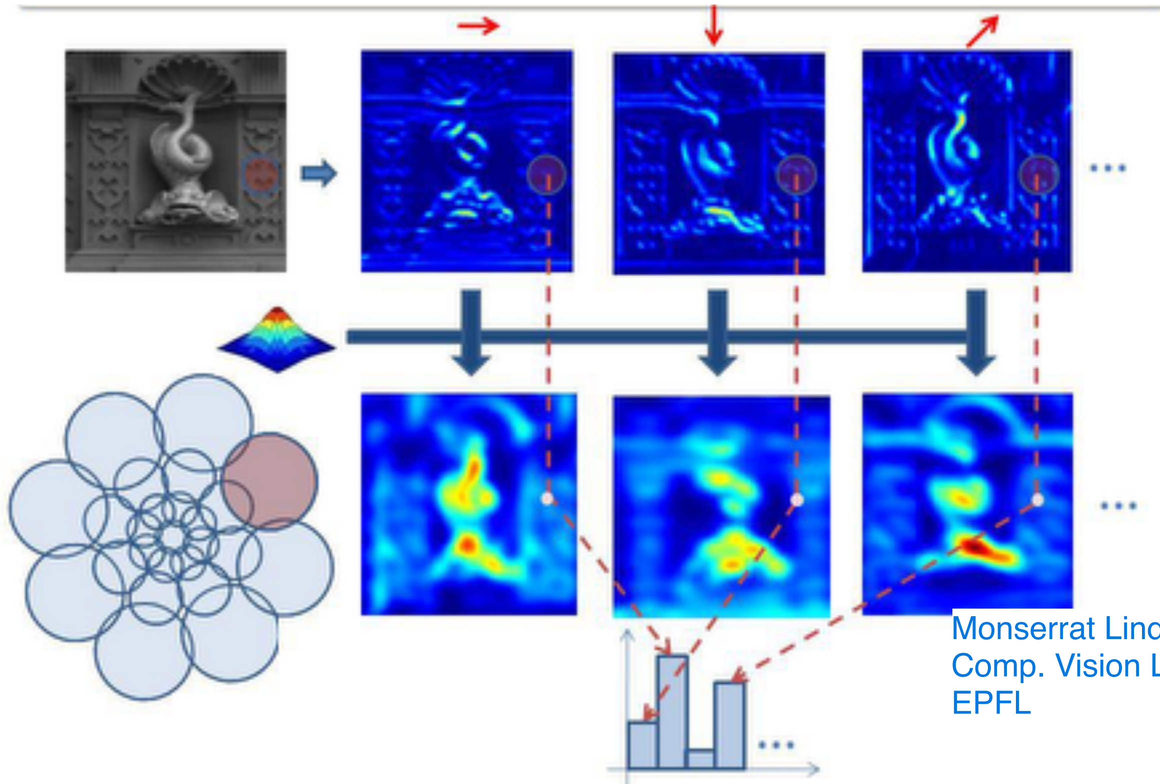
$$\mathcal{D}(u_0,v_0) =$$

$$\begin{bmatrix} \tilde{\mathbf{h}}_{\Sigma_1}^\top(u_0,v_0), \\ \tilde{\mathbf{h}}_{\Sigma_1}^\top(\mathbf{l}_1(u_0,v_0,R_1)),\cdots,\tilde{\mathbf{h}}_{\Sigma_1}^\top(\mathbf{l}_T(u_0,v_0,R_1)), \\ \tilde{\mathbf{h}}_{\Sigma_2}^\top(\mathbf{l}_1(u_0,v_0,R_2)),\cdots,\tilde{\mathbf{h}}_{\Sigma_2}^\top(\mathbf{l}_T(u_0,v_0,R_2)), \end{bmatrix}$$

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

Common operations: DAISY





Monserrat Lindley
Comp. Vision Lab
EPFL

$$\mathcal{D}(u_0, v_0) =$$

take normalized histogram at point u,v

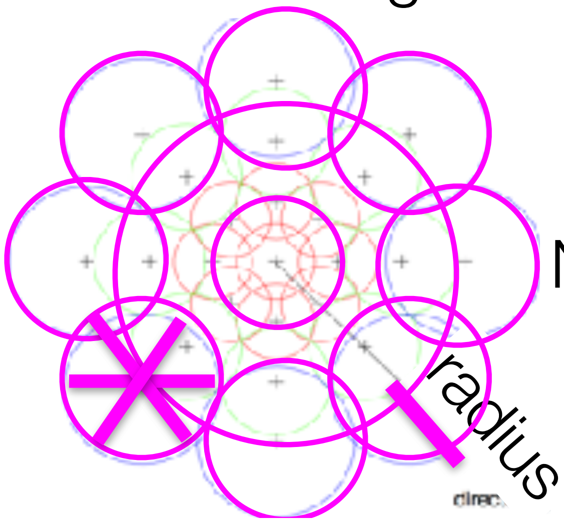
$$\tilde{\mathbf{h}}_\Sigma(u,v) = \left\| \left[\mathbf{G}_1^\Sigma(u,v), \ldots, \mathbf{G}_H^\Sigma(u,v)\right]^\top \right\|$$

$$\begin{aligned}[\; & \tilde{\mathbf{h}}^\top_{\Sigma_1}(u_0, v_0), \\ & \tilde{\mathbf{h}}^\top_{\Sigma_1}(\mathbf{l}_1(u_0, v_0, R_1)), \cdots, \tilde{\mathbf{h}}^\top_{\Sigma_1}(\mathbf{l}_T(u_0, v_0, R_1)), \\ & \tilde{\mathbf{h}}^\top_{\Sigma_2}(\mathbf{l}_1(u_0, v_0, R_2)), \cdots, \tilde{\mathbf{h}}^\top_{\Sigma_2}(\mathbf{l}_T(u_0, v_0, R_2)),\end{aligned}$$

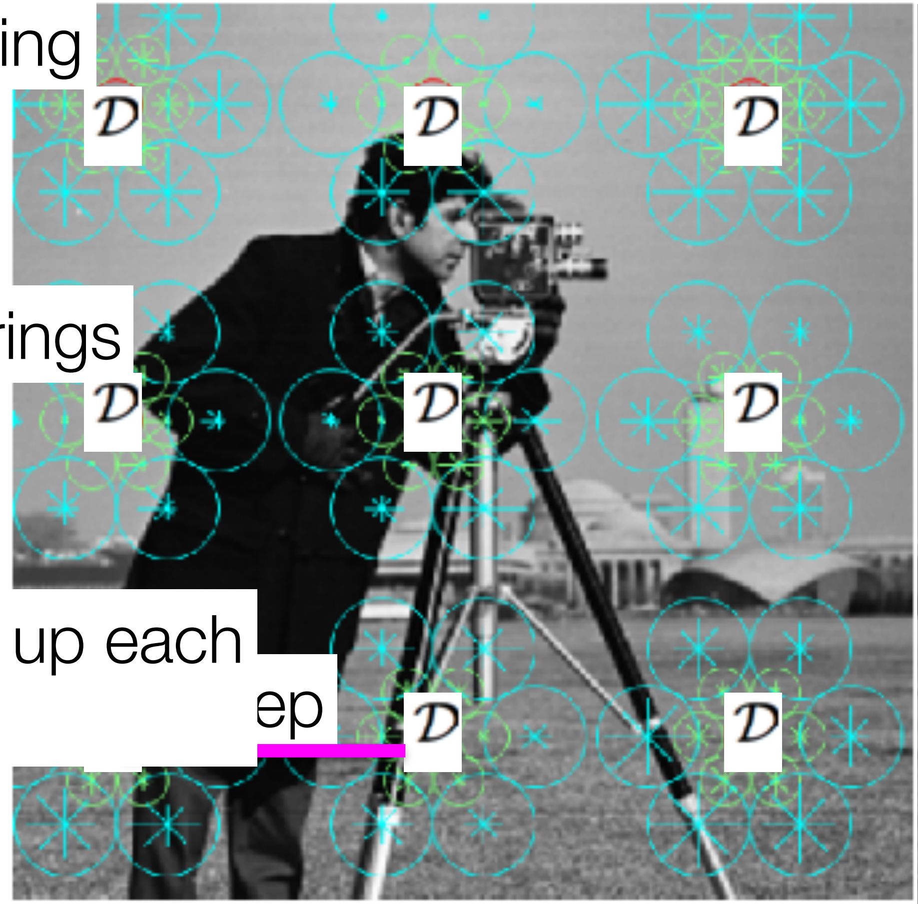
Tola et al. “Daisy: An efficient dense descriptor applied to wide-baseline stereo.” Pattern Analysis and Machine Intelligence, IEEE Transactions

Common operations: DAISY

Num histograms/per ring



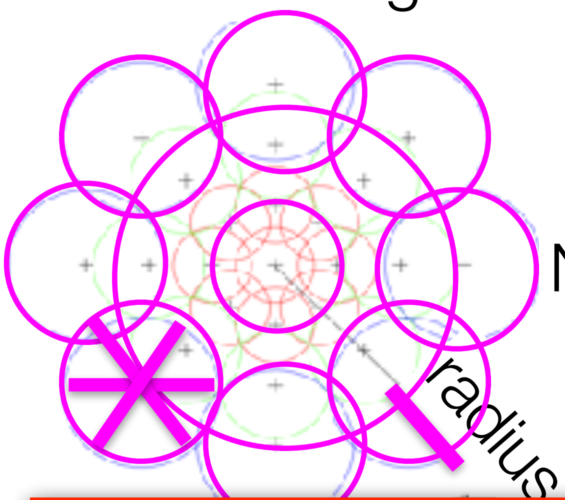
Num rings

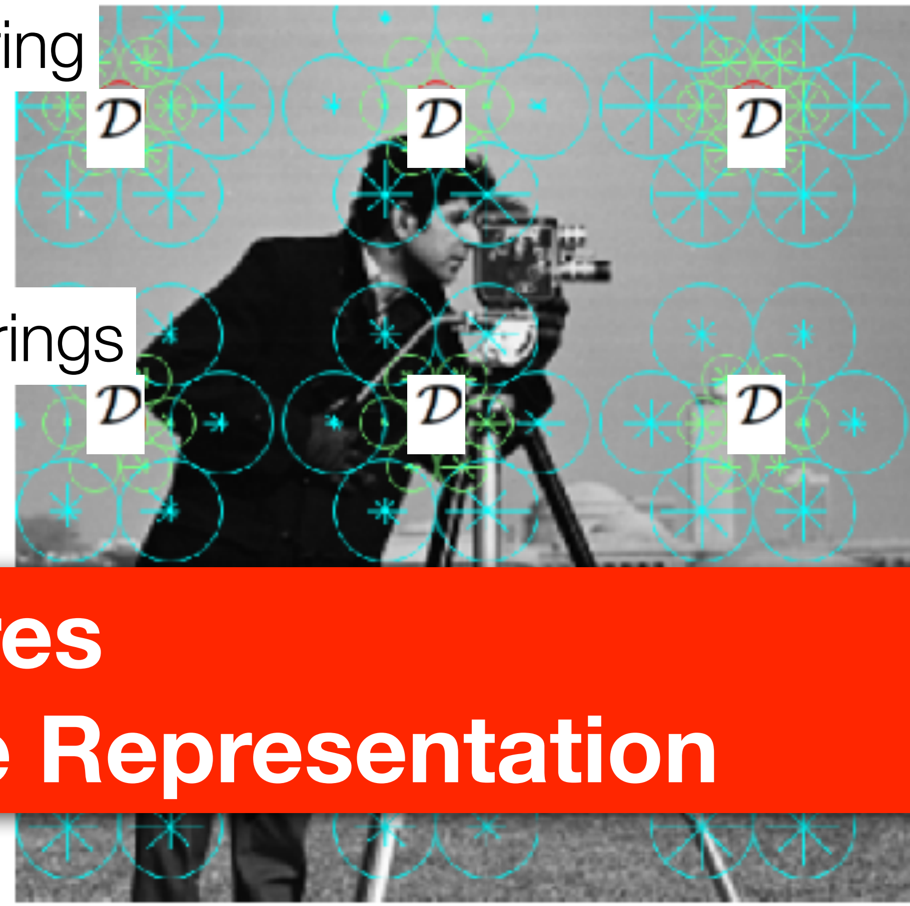


num orientations making up each histogram

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

Common operations: DAISY





Bag of Features

Image Representation

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

More Image Processing

Demo

Gradients

DAISY

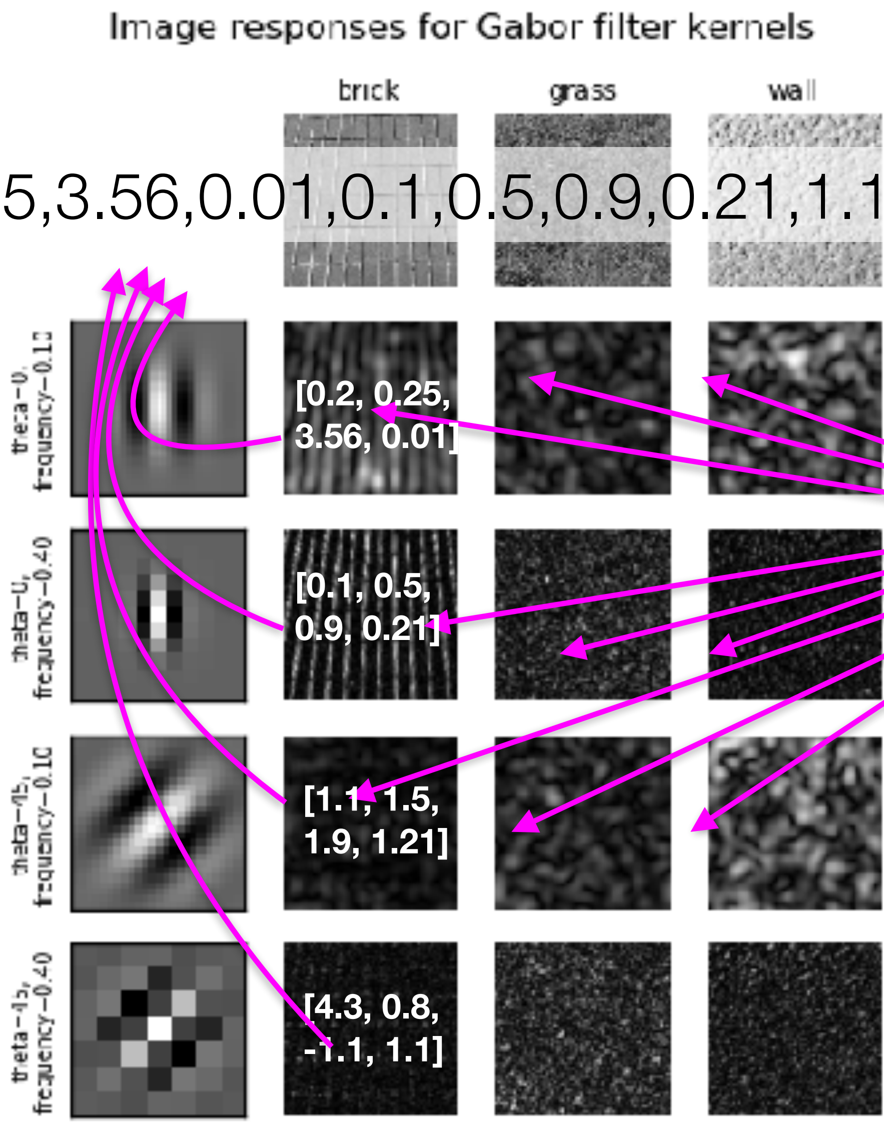
(if time)Gabor Filter Banks

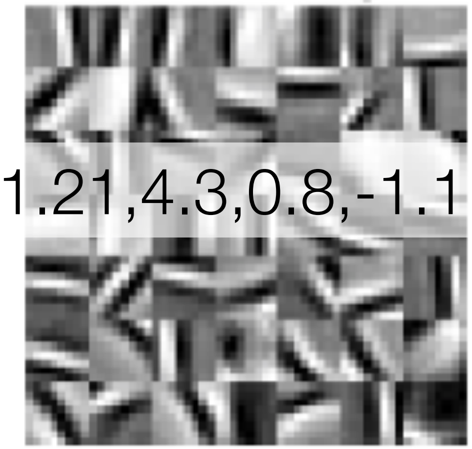
Other Tutorials:

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

Common operations: Gabor filter Banks (if time)

- common used for texture classification





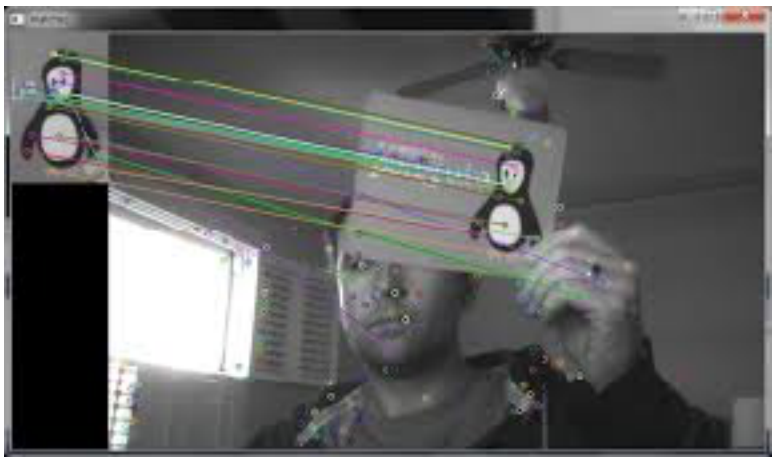
[0.2,0.25,3.56,0.01,0.1,0.5,0.9,0.21,1.1,1.5,1.9,1.21,4.3,0.8,-1.1,1.1]

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

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**

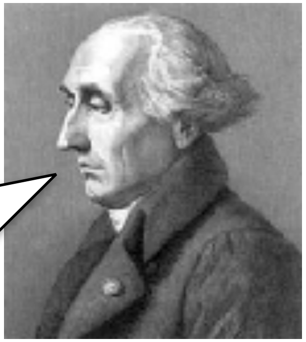
Dimensionality Reduction: LDA

- 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?

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

- 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

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

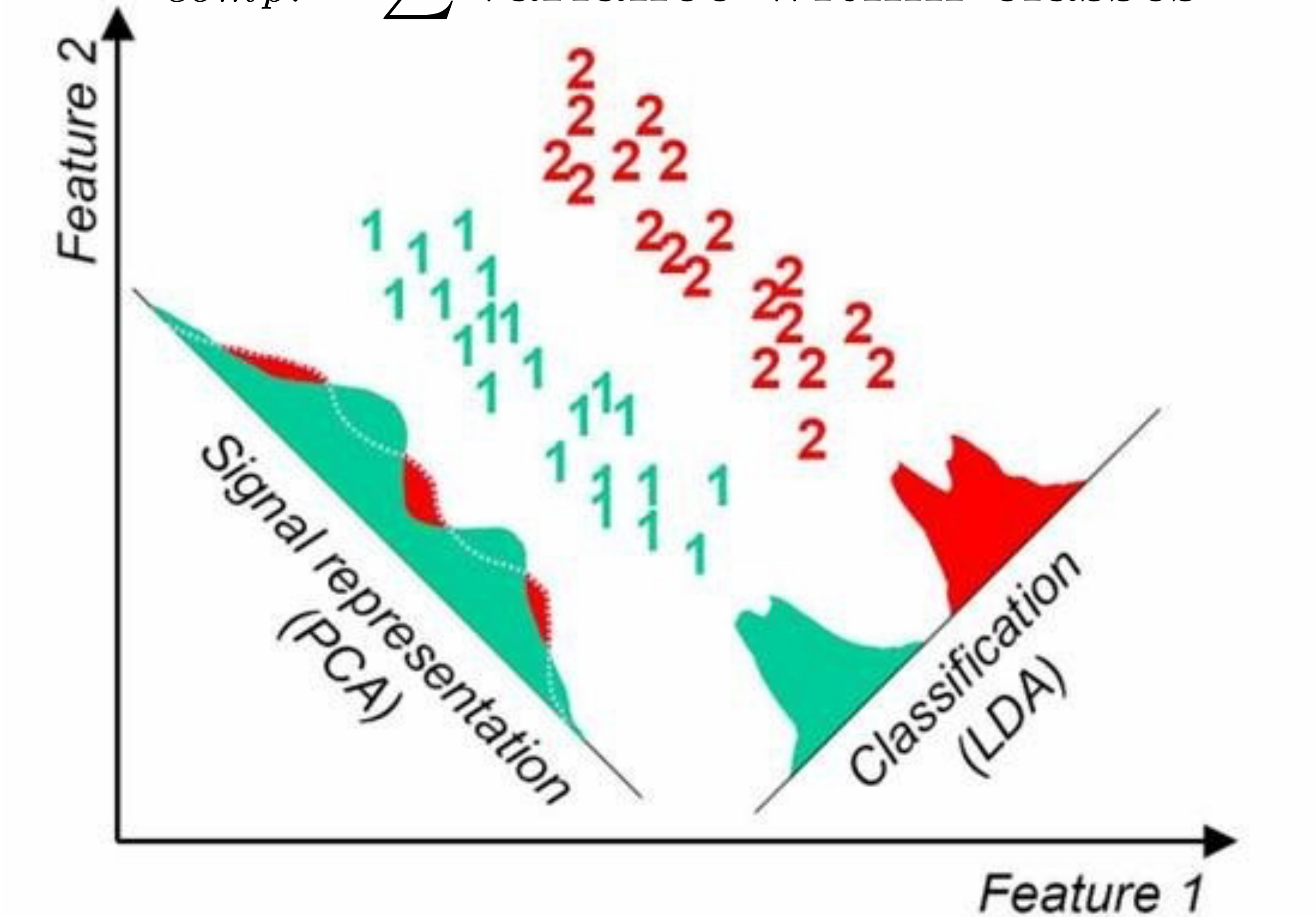
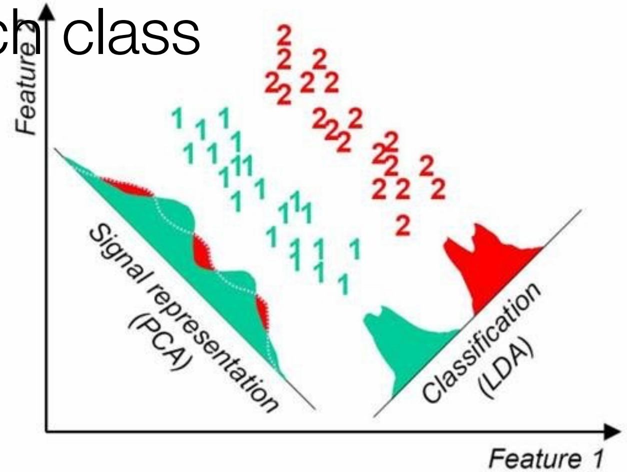


image source: 不想飞翔，不是因为没有翅膀，而是失去了梦想

Dimensionality Reduction: LDA versus QDA

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

- "*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