Advanced Machine Learning & Perception

Instructor: Tony Jebara

Topic 1

- Introduction, researchy course, latest papers
- Going beyond simple machine learning
- Perception, strange spaces, images, time, behavior
- Info, policies, texts, web page
- Syllabus, Overview, Review
- Gaussian → simplest distribution / signal handling…
- Representation & Appearance Based Methods
- Least Squares, Correlation, Gaussians, Bases
- Principal Components Analysis and shortcomings it entails

About me

- Tony Jebara, Associate Professor in Computer Science
- Started at Columbia in 2002
- PhD from MIT in Machine Learning
  - Thesis: *Discriminative, Generative and Imitative Learning (2001)*
- Research Areas: (Columbia Machine Learning Lab, CEPSR 6LE5)
  - www.cs.columbia.edu/learning
  - Machine Learning
  - Some Computer Vision

Course Web Page

http://www.cs.columbia.edu/~jebara/4772

http://www.cs.columbia.edu/~jebara/6772

Some material & announcements will be online

But, many things will be handed out in class such as photocopies of papers for readings, etc.

Check NEWS link to see deadlines, homework, etc.

Available online, see TA info, etc.

Follow the policies, homework, deadlines, readings closely please!

Syllabus

Week 1 Introduction, Review of Basic Concepts, Representation Issues, Vector and Appearance-Based Models, Correlation and Least Squared Error Methods, Bases, Eigenspace Recognition, Principal Components Analysis

Week 2 Nonlinear Dimensionality Reduction, Manifolds, kernel PCA, Locally Linear Embedding, Maximum Variance Unfolding, Minimum Volume Embedding

Week 3 Support Vector Machines and related machines, VC Dimension, Large Margin, Large Relative Margin

Week 4 Kernel Methods, Reproducing Kernel Hilbert Space, Probabilistic Kernel Approaches, Kernel Principal Components Analysis, Bag of Vectors/Pixel Kernels

Week 5 Maximum Entropy, Iterative Scaling, Maximum Entropy Discrimination, Large Margin Probability Models

Week 6 SVM Extensions, Multi-Class Classification, Error-Correcting Codes, Feature Selection, Kernel Selection, Meta-Learning, Transduction

Syllabus

Week 7 Bayesian Networks, Belief Propagation, Hidden Markov Models, Gesture Recognition

Week 8 Kalman Filtering, Structure from Motion, Parameter Estimation, Coupled and Linked Hidden Markov Models, Variational and Mean-Field Methods

Week 9 Factorial Hidden Markov Models, Switched Kalman Filters, Dynamical Bayesian Networks, Structured Mean-Field

Week 10 Graph Learning, b-Matching, Loopy Belief Propagation, Perfect Graphs

Week 11 Spectral Clustering, Random Walks, Ncuts Methods, Stability, Image Segmentation

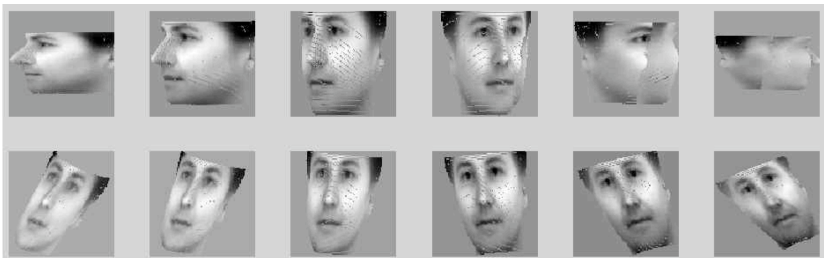
Week 12 Boosting, Mixtures of Experts, AdaBoost, Online Learning

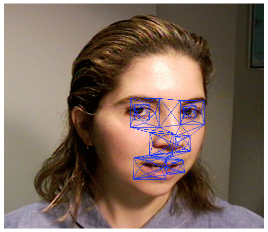
Week 13 TBD

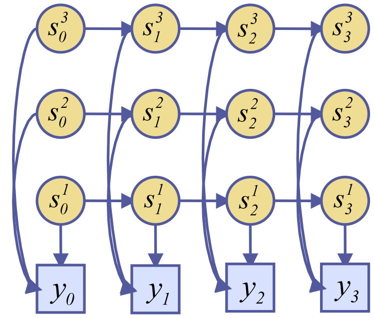
Week 14 Project Presentations

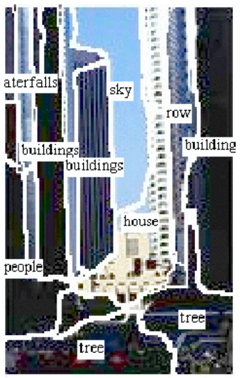
Beyond Canned Learning

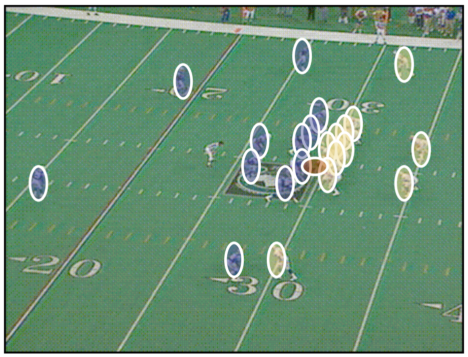
•Latest research methods, high dimensions, nonlinearities, dynamics, manifolds, invariance, unlabeled, feature selection
•Modeling Images / People / Activity / Time Series / MoCAP

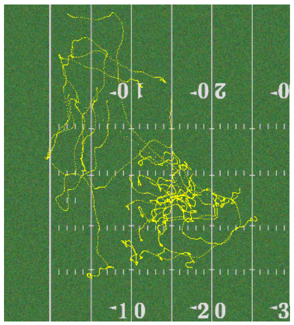


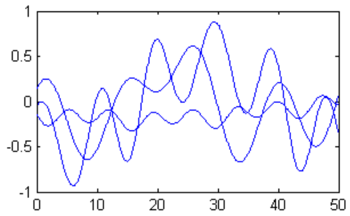






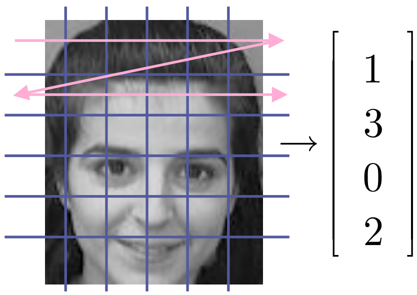






Representation & Vectorization

- How to represent our data? Images, time series, genes…
- Vectorization: the poor man's representation
- Almost anything can be written as a long vector
- E.g. image is read lexicographically
  RGB of eachpixel is added to a vector



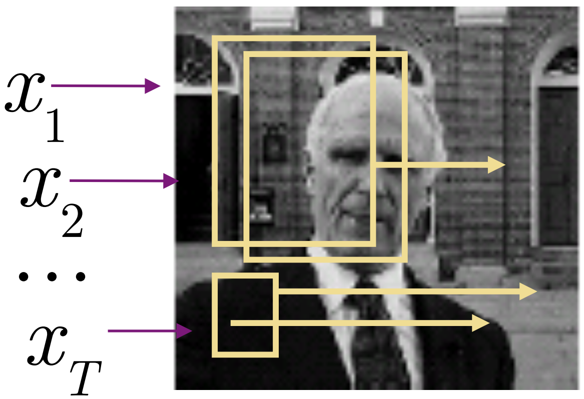
- Or, a gene sequence can be written
  as a binary vector

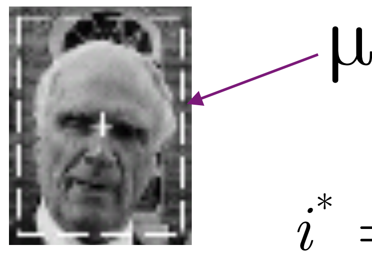
GATACAC = [0100 0001 1000 0001 0010 0001 0010]

- For images, this is called "Appearance Based" representation
- But, we lose many important properties this way
- We will fix this in later lectures
- For now, it is an easy way to proceed

Least Squares Detection

- How to find a face in an image given a template?
- Template Matching and Sum-Squares-Difference (SSD)
- Naïve: try all positions/scales, find least squares distance





$$i^* = \arg\min_{i \in [1,T]} \tfrac{1}{2} \left\| \mu - x_i \right\|^2$$

$$i^* = \arg\min_{i \in [1,T]} \tfrac{1}{2} \left(\mu - x_i \right)^T \left(\mu - x_i \right)$$

- Correlation and Normalized Correlation
  Could normalize length of all vectors (fixes lighting)

$$\hat{\mu} = \frac{\mu}{\left\| \mu \right\|}$$

$$\begin{aligned} i^* &= \arg\min_{i \in [1,T]} \tfrac{1}{2} \left(\hat{\mu}^T \hat{\mu} - 2 \hat{\mu}^T \hat{x}_i + \hat{x}_i^T \hat{x}_i \right) \\ &= \arg\max_{i \in [1,T]} \hat{\mu}^T \hat{x}_i \end{aligned}$$

Least Squares & Gaussians

- Minimum squared error is equivalent to maximum likelihood under a Gaussian

$$i^* = \arg\min_{i\in[1,T]} \tfrac{1}{2}\left\|\mu - x_i\right\|^2$$

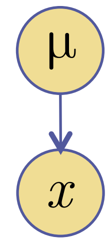
$$= \arg\max_{i\in[1,T]} \log\left(\frac{1}{(2\pi)^{D/2}}\exp\left(-\tfrac{1}{2}\left\|\mu - x_i\right\|^2\right)\right)$$

- Can now treat it as a probabilistic problem
- Trying to find the most likely position x_i (or sub-image) in search image given the Gaussian model $\theta = \mu$ of the template
- Define the log of the likelihood as: $\log p\left(x \mid \theta\right)$
- For a Gaussian probability or likelihood is:

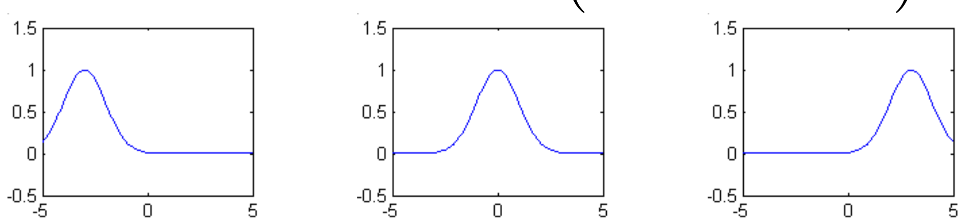
$$p\left(x_i \mid \theta\right) = \frac{1}{(2\pi)^{D/2}}\exp\left(-\tfrac{1}{2}\left\|\mu - x_i\right\|^2\right)$$

Gaussian Distribution

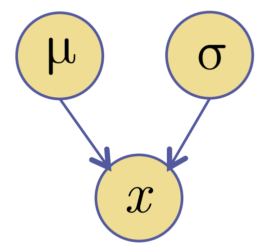
- Recall 1-dimensional Gaussian with mean parameter μ translates Gaussian left & right



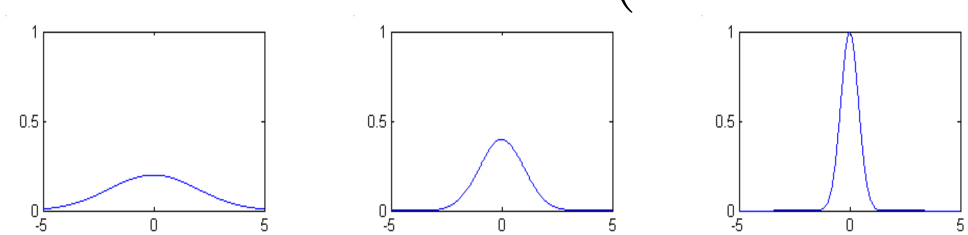
$$p\left(x \mid \mu\right) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(x-\mu\right)^2\right)$$



- Can also have variance parameter σ widens or narrows the Gaussian



$$p\left(x \mid \mu, \sigma\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}\left(x-\mu\right)^2\right)$$



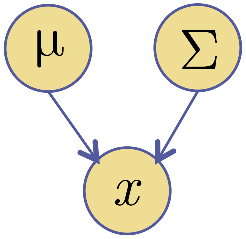
Note: $\int_{x=-\infty}^{\infty} p\left(x\right) dx = 1$

Multivariate Gaussian

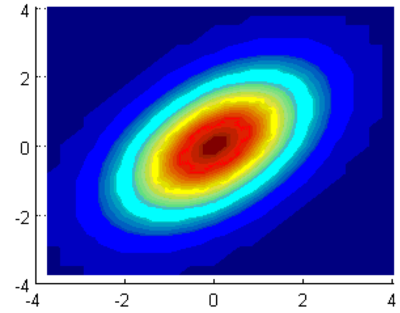
- Gaussian can extend to D-dimensions
- Gaussian mean parameter μ vector (translates)
- Covariance matrix Σ stretches and rotates bump

$$p\left(x \mid \mu, \Sigma\right) = \frac{1}{\left(2\pi\right)^{D/2}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}\left(x-\mu\right)^T \Sigma^{-1}\left(x-\mu\right)\right)$$

$$x, \mu \in \Re^D,\ \Sigma \in \Re^{D \times D}$$



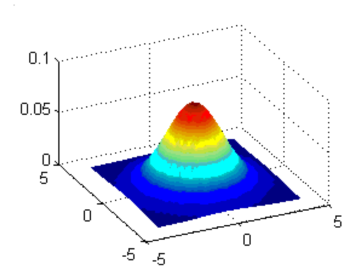
- Max and expectation = μ
- Mean is any real vector, variance is now Σ matrix
- Covariance matrix is positive semi-definite
- Covariance matrix is symmetric
- Need matrix inverse (inv)
- Need matrix determinant (det)
- Need matrix trace operator (trace)

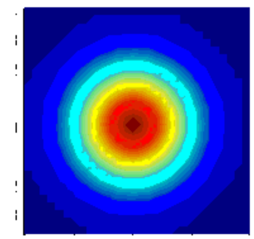


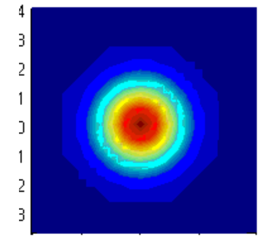
Multivariate Gaussian

- Spherical:

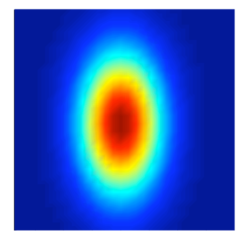
$$\Sigma = \sigma^2 I = \begin{bmatrix} \sigma^2 & 0 & 0 \\ 0 & \sigma^2 & 0 \\ 0 & 0 & \sigma^2 \end{bmatrix}$$

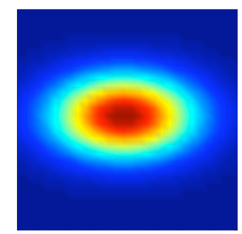


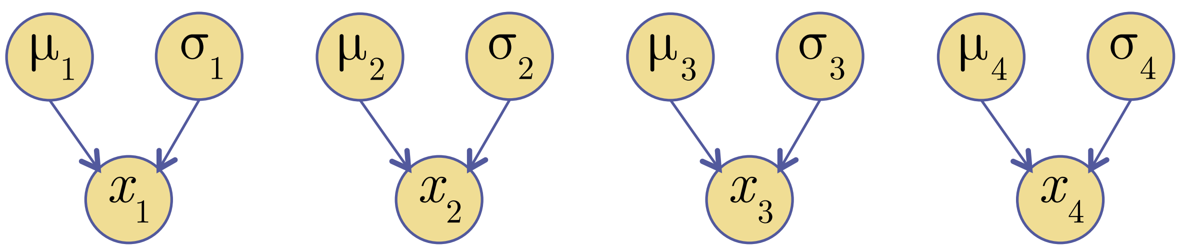




- Diagonal Covariance:
  dimensions of x are independent
  product of multiple 1d Gaussians





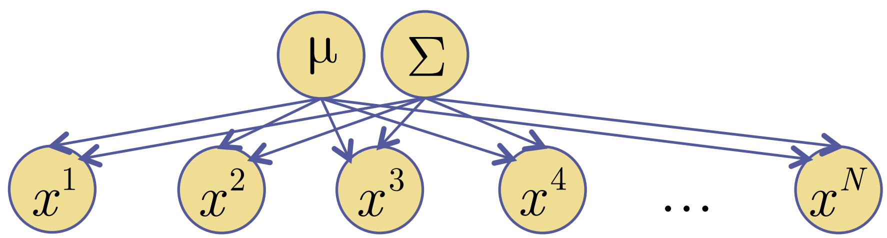


$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 & 0 & 0 \\ 0 & \sigma_2^2 & 0 & 0 \\ 0 & 0 & \sigma_3^2 & 0 \\ 0 & 0 & 0 & \sigma_4^2 \end{bmatrix}$$

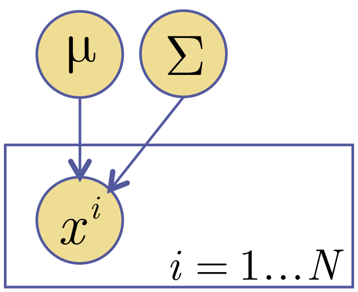
$$p\left(x \mid \mu, \Sigma\right) = \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi}\sigma_d} \exp\left(-\frac{\left(x_d - \mu_d\right)^2}{2\sigma_d^2}\right)$$

Max Likelihood Gaussian

- How to make face detector to work on all faces?
- Why use a template? How can we use many templates?
- Have IID samples of template vectors i=1..N: $x^1, x^2, x^3, \ldots, x^N$
- Represent IID samples with parameters as network:



More efficiently drawn using Replicator Plate



- Let us get a good Gaussian from these many templates.
- Standard approach: Max Likelihood

$$\sum_{i=1}^{N} \log p\left(x^i \mid \mu, \Sigma\right) = \sum_{i=1}^{N} \log \frac{1}{(2\pi)^{D/2}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}\left(x^i - \mu\right)^T \Sigma^{-1}\left(x^i - \mu\right)\right)$$

Max Likelihood Gaussian

- Max over μ

$$\frac{\partial}{\partial \mu}\sum_{i=1}^{N} \log \frac{1}{(2\pi)^{D/2}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}\left(x^i-\mu\right)^T \Sigma^{-1}\left(x^i-\mu\right)\right) = 0$$

$$\frac{\partial}{\partial \mu}\sum_{i=1}^{N} -\frac{D}{2}\log 2\pi - \frac{1}{2}\log\left|\Sigma\right| - \frac{1}{2}\left(x^i-\mu\right)^T \Sigma^{-1}\left(x^i-\mu\right) = 0$$

$$\boxed{\frac{\partial x^T x}{\partial x} = 2x^T}$$

$$\sum_{i=1}^{N}\left(x^i-\mu\right)^T \Sigma^{-1} = 0$$

see Jordan Ch. 12, get sample mean...

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x^i$$

- For Σ need Trace operator:

$$tr\left(A\right) = tr\left(A^T\right) = \sum_{d=1}^{D} A_{dd}$$

and several properties:

$$tr\left(AB\right) = tr\left(BA\right)$$

$$tr\left(BAB^{-1}\right) = tr\left(A\right)$$

$$tr\left(xx^T A\right) = tr\left(x^T A x\right) = x^T A x$$

Max Likelihood Gaussian

- Likelihood rewritten in trace notation:

$$l = \sum_{i=1}^{N} -\tfrac{D}{2}\log 2\pi - \tfrac{1}{2}\log\left|\Sigma\right| - \tfrac{1}{2}\left(x^i-\mu\right)^T \Sigma^{-1}\left(x^i-\mu\right)$$

$$= -\tfrac{ND}{2}\log 2\pi + \tfrac{N}{2}\log\left|\Sigma^{-1}\right| - \tfrac{1}{2}\sum_{i=1}^{N} tr\left[\left(x^i-\mu\right)^T \Sigma^{-1}\left(x^i-\mu\right)\right]$$

$$= -\tfrac{ND}{2}\log 2\pi + \tfrac{N}{2}\log\left|\Sigma^{-1}\right| - \tfrac{1}{2}\sum_{i=1}^{N} tr\left[\left(x^i-\mu\right)\left(x^i-\mu\right)^T \Sigma^{-1}\right]$$

$$= -\tfrac{ND}{2}\log 2\pi + \tfrac{N}{2}\log\left|A\right| - \tfrac{1}{2}\sum_{i=1}^{N} tr\left[\left(x^i-\mu\right)\left(x^i-\mu\right)^T A\right]$$

- Max over $A=\Sigma^{-1}$ use properties:

$$\frac{\partial \log\left|A\right|}{\partial A} = \left(A^{-1}\right)^T \qquad \frac{\partial tr\left[BA\right]}{\partial A} = B^T$$

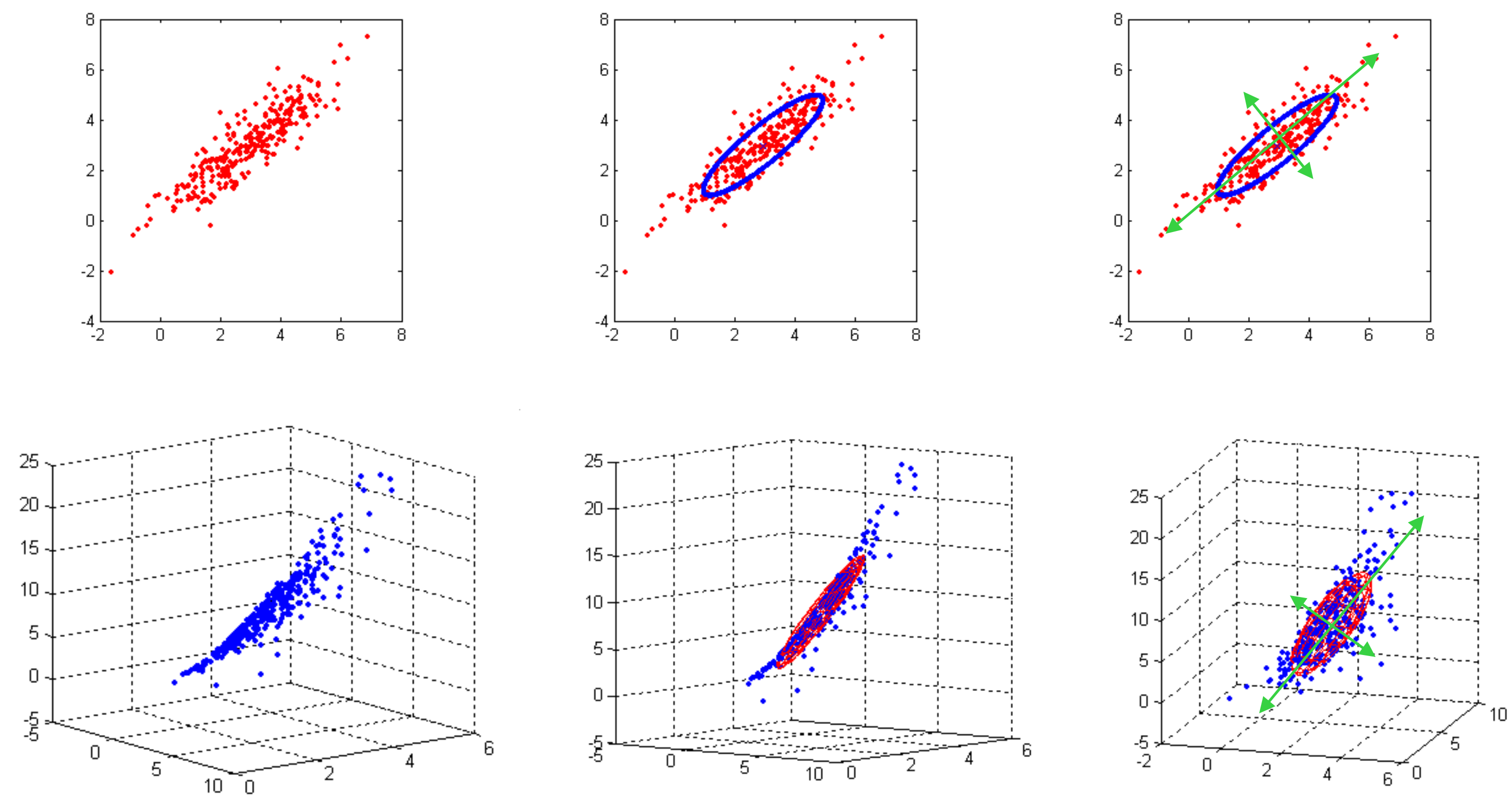
$$\frac{\partial l}{\partial A} = -0 + \tfrac{N}{2}\left(A^{-1}\right)^T - \tfrac{1}{2}\sum_{i=1}^{N}\left[\left(x^i-\mu\right)\left(x^i-\mu\right)^T\right]^T$$

$$= \tfrac{N}{2}\Sigma - \tfrac{1}{2}\sum_{i=1}^{N}\left(x^i-\mu\right)\left(x^i-\mu\right)^T$$

- Get sample covariance: $\frac{\partial l}{\partial A} = 0 \rightarrow \Sigma = \tfrac{1}{N}\sum_{i=1}^{N}\left(x^i-\mu\right)\left(x^i-\mu\right)^T$

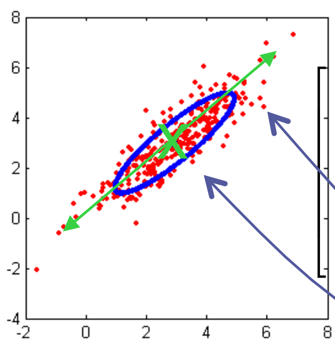
Principal Components Analysis

- Gaussians: for Classification, Regression… & Compression!
- Data can be constant in some directions, changes in others
- Use Gaussian to find directions of high/low variance

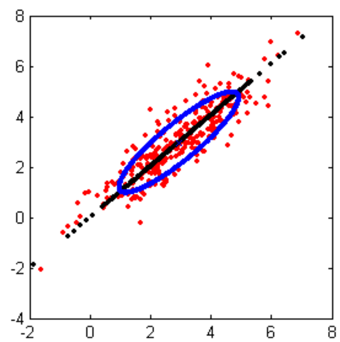


Principal Components Analysis

- Idea: instead of writing data in all its dimensions, only write it as mean + steps along one direction

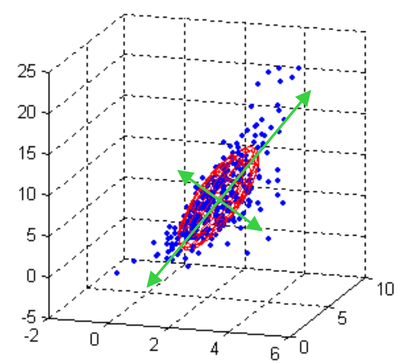


$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} \approx \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix} + c_i \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$



- More generally, keep a subset of dimensions C from D (I.e. 2 of 3)

$$\vec{x}_i \approx \vec{\mu} + \sum_{j=1}^{C} c_{ij} \vec{v}_j$$



- Compression method: $\vec{x}_i \gg \vec{c}_i$
- Optimal directions: along eigenvectors of covariance
- Which directions to keep: highest eigenvalues (variances)

Principal Components Analysis

- If we have eigenvectors, mean and coefficients:

$$\vec{x}_i \approx \vec{\mu} + \sum_{j=1}^{C} c_{ij} \vec{v}_j$$

- Getting eigenvectors (I.e. approximating the covariance):

$$\Sigma = V \Lambda V^T$$

$$\begin{bmatrix} \Sigma_{11} & \Sigma_{12} & \Sigma_{13} \\ \Sigma_{12} & \Sigma_{22} & \Sigma_{23} \\ \Sigma_{13} & \Sigma_{23} & \Sigma_{33} \end{bmatrix} = \begin{bmatrix} \left[\vec{v}_1\right] & \left[\vec{v}_2\right] & \left[\vec{v}_3\right] \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} \left[\vec{v}_1\right] & \left[\vec{v}_2\right] & \left[\vec{v}_3\right] \end{bmatrix}^T$$

- Eigenvectors are orthonormal: $\vec{v}_i^T \vec{v}_j = \delta_{ij}$
- In coordinates of v, Gaussian is diagonal, cov = Λ
- All eigenvalues are non-negative $\lambda_i \geq 0$
- Higher eigenvalues are higher variance, use those first

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \lambda_4 \geq \ldots$$

- To compute the coefficients: $c_{ij} = \left(\vec{x}_i - \vec{\mu}\right)^T \vec{v}_j$

Snapshot Method

- Assume 2000 images each containing D=10,000 pixels
- How big is the covariance matrix?
- It is DxD pixels
- Finding the eigenvectors or inverting DxD, requires O(D^3)
- First compute mean of all data (easy) and subtract it from each point

Instead of: $\Sigma = \frac{1}{N}\sum_{i=1}^{N} x_i x_i^T$ compute $\Phi \; where \; \Phi_{i,j} = x_i^T x_j$

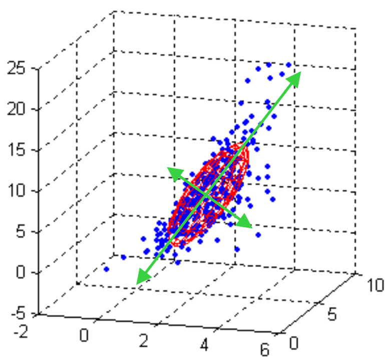
Then find eigenvectors/eigenvalues of $\Phi = \tilde{V}\Lambda\tilde{V}^T$

Eigenvectors of Σ are then: $v_i \propto \sum_{j=1}^{N} x_j \tilde{v}_i\left(j\right)$

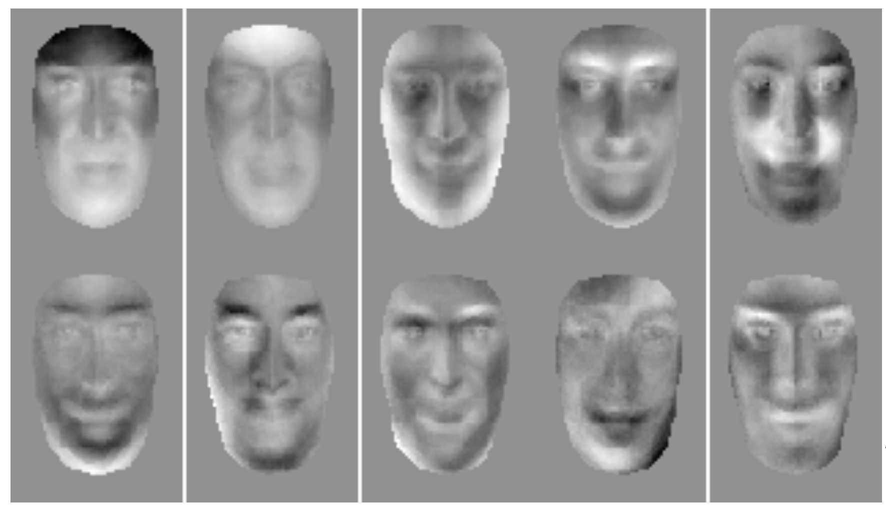
Eigenfaces

$$\left\{x_1,\ldots,x_N\right\} =$$

$\vec{\mu}$



$\vec{v}_1$



$\vec{v}_{10}$

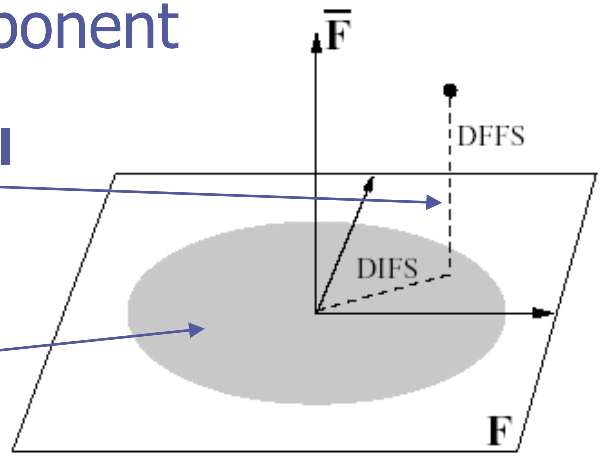
$$\left\{\mu + \sum c_{1j}\vec{v}_j,\ldots\right\} =$$

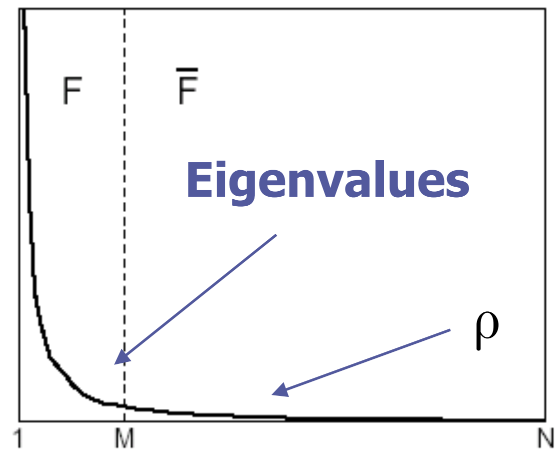
Eigenface Detection

- Use a Gaussian model but only approximate the huge covariance matrix with its eigenvectors and a constant spherical component

Spherical here

Use variable Eigenvalues Eigenvectors here





$$N\left(x \mid \mu, \Sigma\right) = \frac{1}{\left(2\pi\right)^{D/2}\sqrt{\left|\Sigma\right|}} \exp\left(-\frac{1}{2}\left(x-\mu\right)^T \Sigma^{-1}\left(x-\mu\right)\right)$$

$$\Sigma \approx \sum_{k=1}^{M} \lambda_k v_k v_k^T + \rho \sum_{k=M+1}^{D} \delta_k \delta_k^T$$

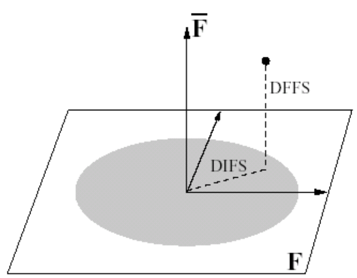
$$\Sigma^{-1} \approx \sum_{k=1}^{M} \frac{1}{\lambda_k} v_k v_k^T + \frac{1}{\rho} \sum_{k=M+1}^{D} \delta_k \delta_k^T$$

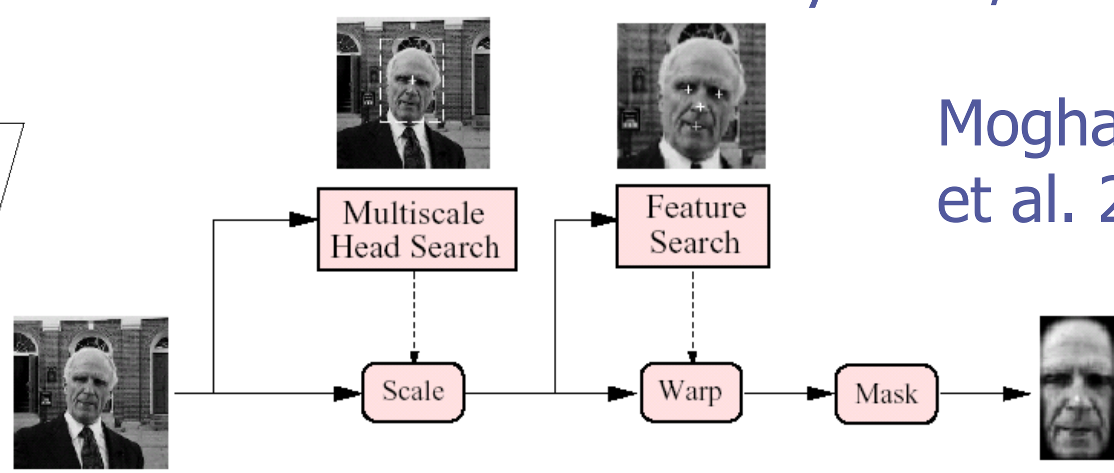
Gaussian Face Finding

- Instead of minimizing squared error, use Gaussian model

$$p\left(x \mid \mu, \Sigma\right) = \frac{1}{\left(2\pi\right)^{D/2}\sqrt{\left|\Sigma\right|}} \exp\left(-\frac{1}{2}\left(x-\mu\right)^T \Sigma^{-1}\left(x-\mu\right)\right)$$

- Euclidean distance: $\frac{1}{2}\left(x-\mu\right)^T\left(x-\mu\right)$ (assumed Σ=I)
- Mahalanobis distance: $\frac{1}{2}\left(x-\mu\right)^T \Sigma^{-1}\left(x-\mu\right)$
  (shrink distances in directions with high variance)
- State of the art face finder as evaluated by NIST/DARPA





Moghaddam et al. 2000

- Top performer after 2000 is Viola-Jones (boosted cascade)

Gaussian Face Recognition

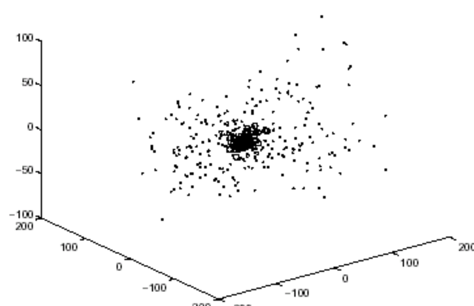
- Instead of modeling face images with Gaussian model the difference of two face images with a Gaussian
- Each difference of all pairs of images in our data is represented as a D-dimensional vector x
- Also have a binary label y, y=1 same person same, y=0 not

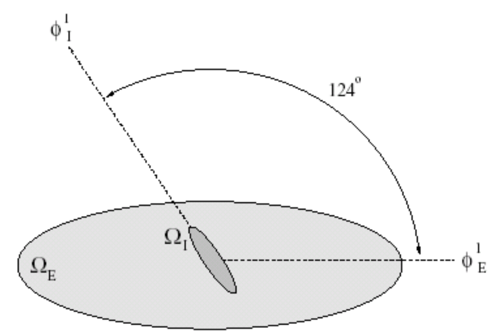
$$x \in R^D \quad y \in \{0,1\}$$

$y_i = 1, x_i =$  - 

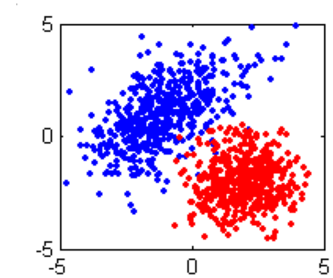
$y_j = 0, x_j =$  - 

- One Gaussian for same-face deltas another for different people deltas



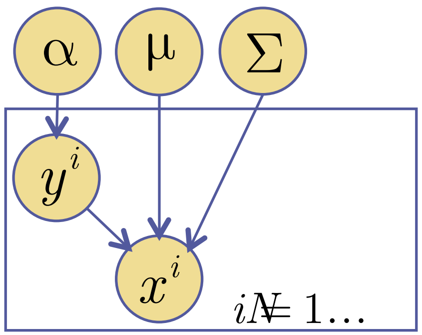


Gaussian Classification



- Have two classes, each with their own Gaussian:

$$\left\{\left(x_1, y_1\right), \ldots, \left(x_N, y_N\right)\right\} \quad x \in R^D \;\; y \in \left\{0,1\right\}$$



$$p(\alpha)\, p(\mu)\, p(\Sigma)\, p(y \mid \alpha)\, p(x \mid y, \mu, \Sigma)$$

$$p(y \mid \alpha) = \alpha^y (1-\alpha)^{1-y}$$

$$p(x \mid \mu, \Sigma, y) = N\left(x \mid \mu_y, \Sigma_y\right)$$

- Generation: 1) flip a coin, get y
  2) pick Gaussian y, sample x from it

- Maximum Likelihood:

$$\begin{aligned} l &= \sum_{i=1}^N \log p\left(x_i, y_i \mid \alpha, \mu, \Sigma\right) \\ &= \sum_{i=1}^N \log p\left(y_i \mid \alpha\right) + \sum_{i=1}^N \log p\left(x_i \mid y_i, \mu, \Sigma\right) \\ &= \sum_{i=1}^N \log p\left(y_i \mid \alpha\right) + \sum_{y_i \in 0} \log p\left(x_i \mid \mu_0, \Sigma_0\right) + \sum_{y_i \in 1} \log p\left(x_i \mid \mu_1, \Sigma_1\right) \end{aligned}$$

Gaussian Classification

- Max Likelihood can be done separately for the 3 terms

$$l = \sum_{i=1}^{N} \log p\left(y_i \mid \alpha\right) + \sum_{y_i \in 0} \log p\left(x_i \mid \mu_0, \Sigma_0\right) + \sum_{y_i \in 1} \log p\left(x_i \mid \mu_1, \Sigma_1\right)$$

- Count # of pos & neg examples (class prior): $\alpha = \frac{N_1}{N_0 + N_1}$
- Get mean & cov of negatives and mean & cov of positives:

$$\mu_0 = \frac{1}{N_0} \sum_{y_{i \in 0}} x_i \qquad \Sigma_0 = \frac{1}{N_0} \sum_{y_{i \in 0}} \left(x_i - \mu_0\right)\left(x_i - \mu_0\right)^T$$

$$\mu_1 = \frac{1}{N_1} \sum_{y_{i \in 1}} x_i \qquad \Sigma_1 = \frac{1}{N_1} \sum_{y_{i \in 1}} \left(x_i - \mu_1\right)\left(x_i - \mu_1\right)^T$$

- Given (x,y) pair, can now compute likelihood $p\left(x, y\right)$
- To make classification, a bit of Decision Theory
- Without x, can compute prior guess for y $p\left(y\right)$
- Give me x, want y, I need posterior $p\left(y \mid x\right)$
- Bayes Optimal Decision: $\hat{y} = \arg\max_{y = \{0,1\}} p\left(y \mid x\right)$
- Optimal iff we have true probability

Gaussian Classification

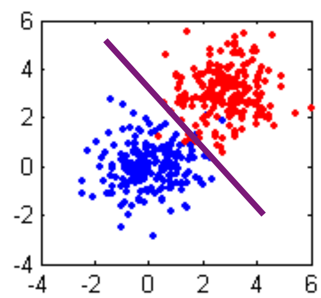
- Example cases, plotting decision boundary when = 0.5

$$p\left(y=1\,|\,x\right)=\frac{p\left(x,y=1\right)}{p\left(x,y=0\right)+p\left(x,y=1\right)}$$

$$=\frac{\alpha N\left(x\,|\,\mu_1,\Sigma_1\right)}{\left(1-\alpha\right)N\left(x\,|\,\mu_0,\Sigma_0\right)+\alpha N\left(x\,|\,\mu_1,\Sigma_1\right)}$$

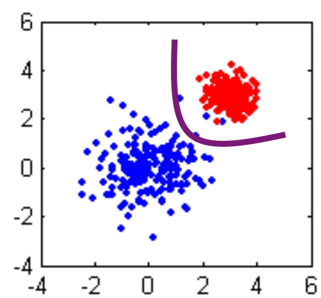
- If covariances are equal:

  linear decision



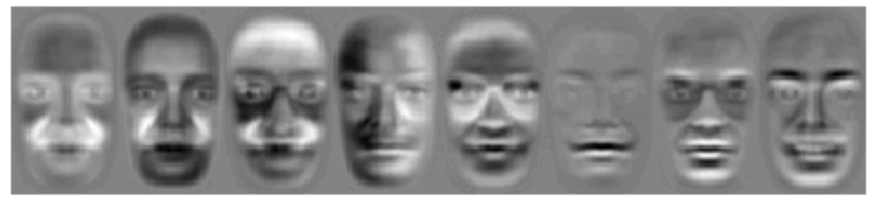
- If covariances are different:

  quadratic decision



Intra-Extra Personal Gaussians

- Intrapersonal Gaussian model $N\left(x \mid \mu_1, \Sigma_1\right)$
- Covariance is approximated by these eigenvectors:



- Extrapersonal Gaussian model $N\left(x \mid \mu_0, \Sigma_0\right)$
- Covariances is approximated by these eigenvectors:

- Question: what are the Gaussian means?
- Probability a pair is the same person:

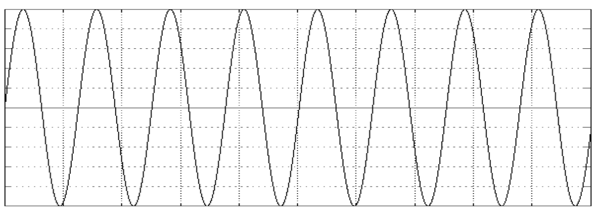
$$p\left(y = 1 \mid x\right) = \frac{\alpha N\left(x \mid \mu_1, \Sigma_1\right)}{\left(1-\alpha\right) N\left(x \mid \mu_0, \Sigma_0\right) + \alpha N\left(x \mid \mu_1, \Sigma_1\right)}$$

Other Standard Bases

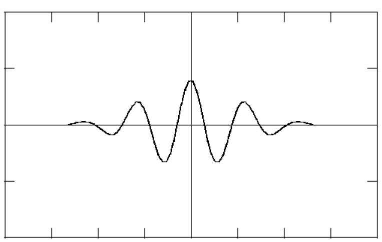
- There are other choices for the eigenvectors, not just PCA
- Could pick eigenvectors without looking at the data
- Just for their interesting properties

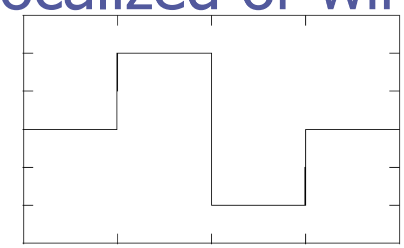
$$\vec{x}_i \approx \vec{\mu} + \sum_{j=1}^{C} c_{ij} \vec{v}_j$$

- Fourier basis: denoises, only keeps smooth parts of image



- Wavelet basis: localized or windowed Fourier





- PCA: optimal least squares linear dataset reconstruction

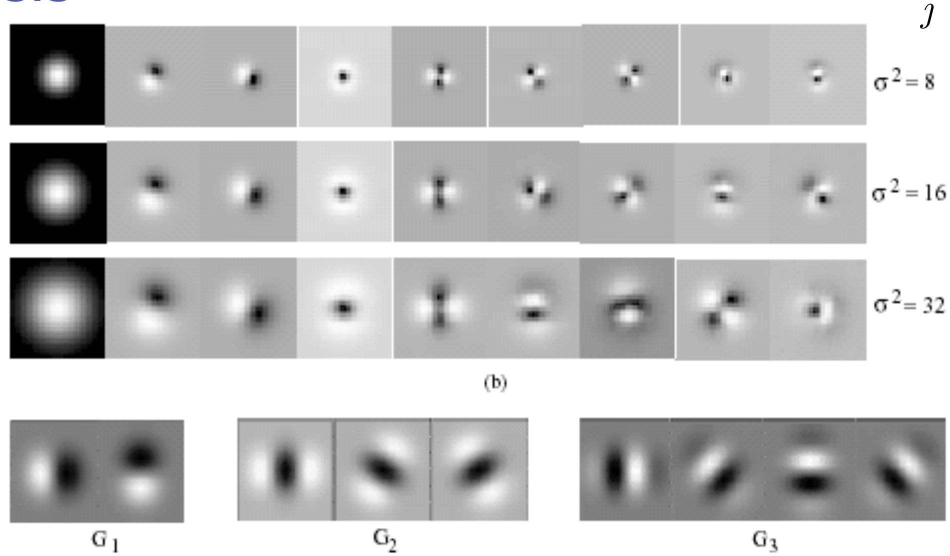
Basis for Natural Images

$\vec{x}_i$

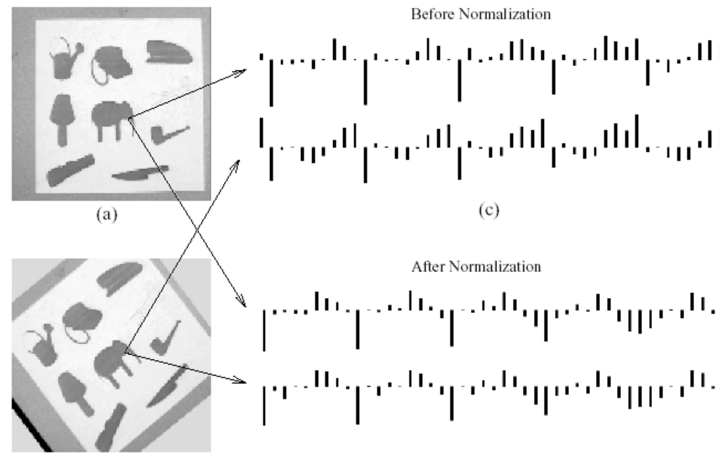
- What happens if we do PCA on all natural images instead of just faces?

- Get difference of Gaussian bases
- Like Gabor or Wavelet basis
- Not specific like faces
- Multi-scale & orientation
- Also called steerable filters
- Similar to visual cortex

$\vec{v}_j$



Problems with Linear Bases



- Coefficient representation changes wildly if image rotates and so does p(x)
- The eigenspace is sensitive to rotations, translations and transformations of the image
- Simple linear/Gaussian/PCA models are not enough
- What worked for aligned faces breaks for general image datasets
- Most of the PCA eigenvectors and spectrum energy is wasted due to NONLINEAR EFFECTS…

$$c_{ij} = \left(\vec{x}_i - \vec{\mu}\right)^T \vec{v}_j$$