# 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 Distributions
- Representation & Appearance Based Methods
- Least Squares, Correlation, Gaussians, Bases
- Principal Components Analysis and its Shortcomings

# 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: Maximum Entropy, Exponential Families, Maximum Entropy Discrimination, Large Margin Probability Models

Week 4: Conditional Random Fields and Linear Models, Iterative Scaling and Majorization

Week 5: Graphical Models, Multi-Class Support Vector Machines, Structured Support Vector Machines, Cutting Plane Algorithms

Week 6: Kernels and Probabilistic Kernels

# Syllabus

Week 7: Feature Selection and Kernel Selection, Support Vector Machine Extensions

Week 8: Meta-Learning and Multi-Task Support Vector Machines

Week 9: Semi-Supervised Learning and Graph-Based Semi-Supervised Learning

Week 10: High-Tree Width Graphical Models, Approximate Inference, Graph Structure Learning

Week 11: Clustering, Spectral Clustering, Normalized Cuts.

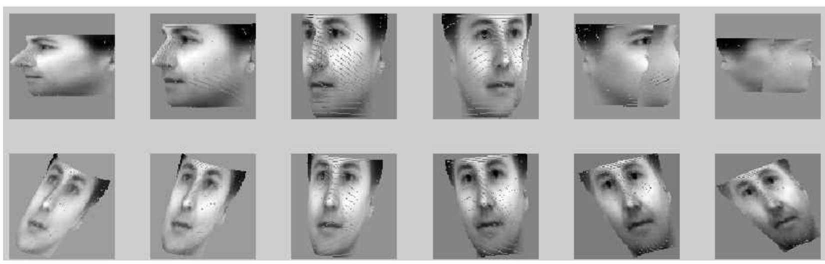
Week 12: Boosting, Mixtures of Experts, AdaBoost, Online Learning

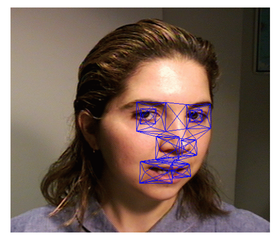
Week 13: Project Presentations

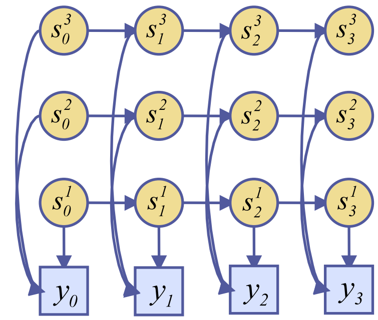
Week 14: Project Presentations

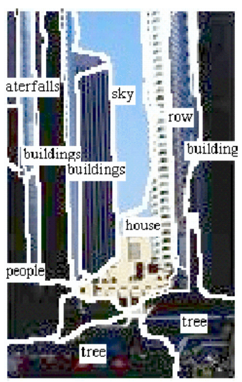
# Beyond Canned Learning

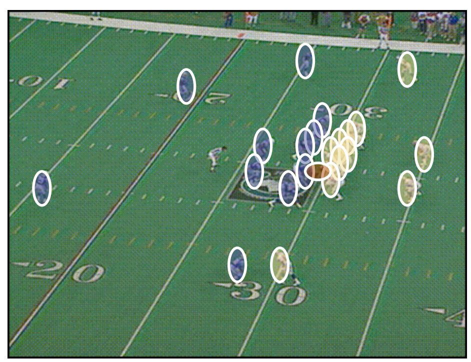
- Latest research methods, high dimensions, nonlinearities, dynamics, manifolds, invariance, unlabeled, feature selection
- Modeling Images / People / Activity / Time Series / MoCAP

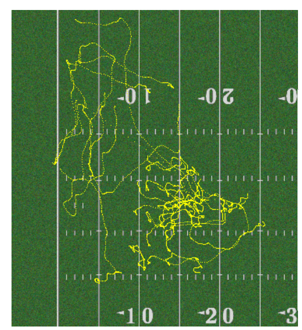


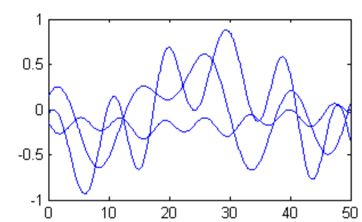




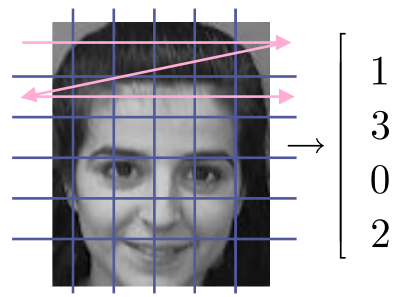







# Representation & Vectorization

- 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



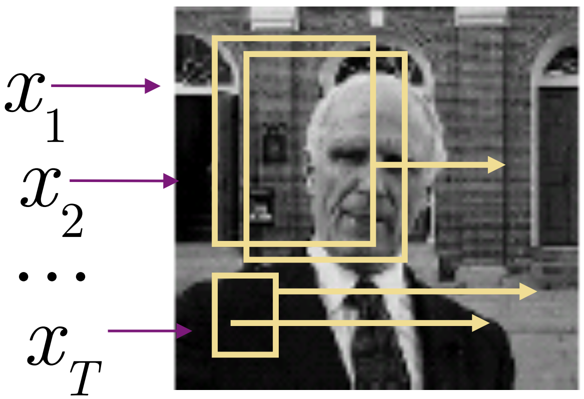

- Or, a gene sequence can be written
  as a binary vector

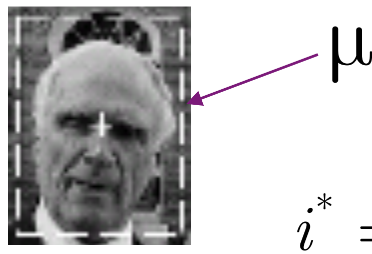
GATACAC = [0100 0001 1000 0001 0010 0001 0010]

- 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 as Gaussian Model

- 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: $\log p(x \mid \theta)$
- For a Gaussian probability or likelihood is:

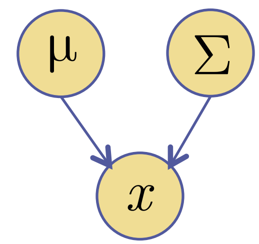
$$p(x_i \mid \theta) = \frac{1}{(2\pi)^{D/2}}\exp\left(-\tfrac{1}{2}\left\|\mu - x_i\right\|^2\right)$$

# Multivariate Gaussian

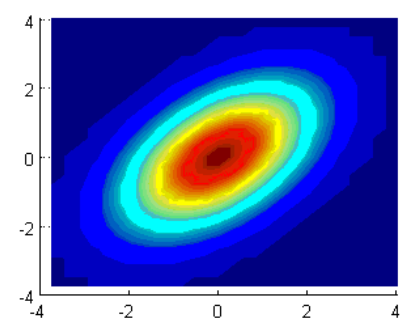
- Gaussian extend to D-dimensions and have 2 parameters:
    - Mean vector $\mu$ vector (translates)
    - Covariance matrix $\Sigma$ (stretches and rotates)

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

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

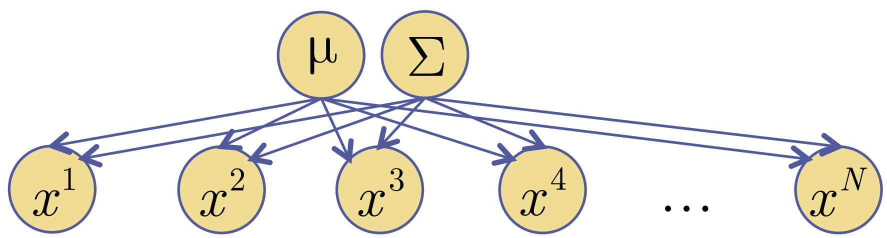



- Max and expectation = $\mu$
- Mean is any real vector, variance is now $\Sigma$ matrix
- Covariance matrix is positive semi-definite
- Covariance matrix is symmetric
- Need matrix inverse (inv)
- Need matrix determinant (det)
- Need matrix trace operator (trace)

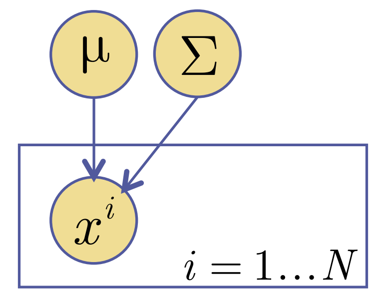


# Max Likelihood Gaussian

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

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

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

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

see Jordan Ch. 12, get sample mean... $$\mu=\tfrac{1}{N}\sum_{i=1}^{N}x^i$$

- For $\Sigma$ need Trace operator: $$tr\left(A\right)=tr\left(A^T\right)=\sum\nolimits_{d=1}^{D}A_{dd}$$

and several properties:

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

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

$$tr\left(xx^TA\right)=tr\left(x^TAx\right)=x^TAx$$

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

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

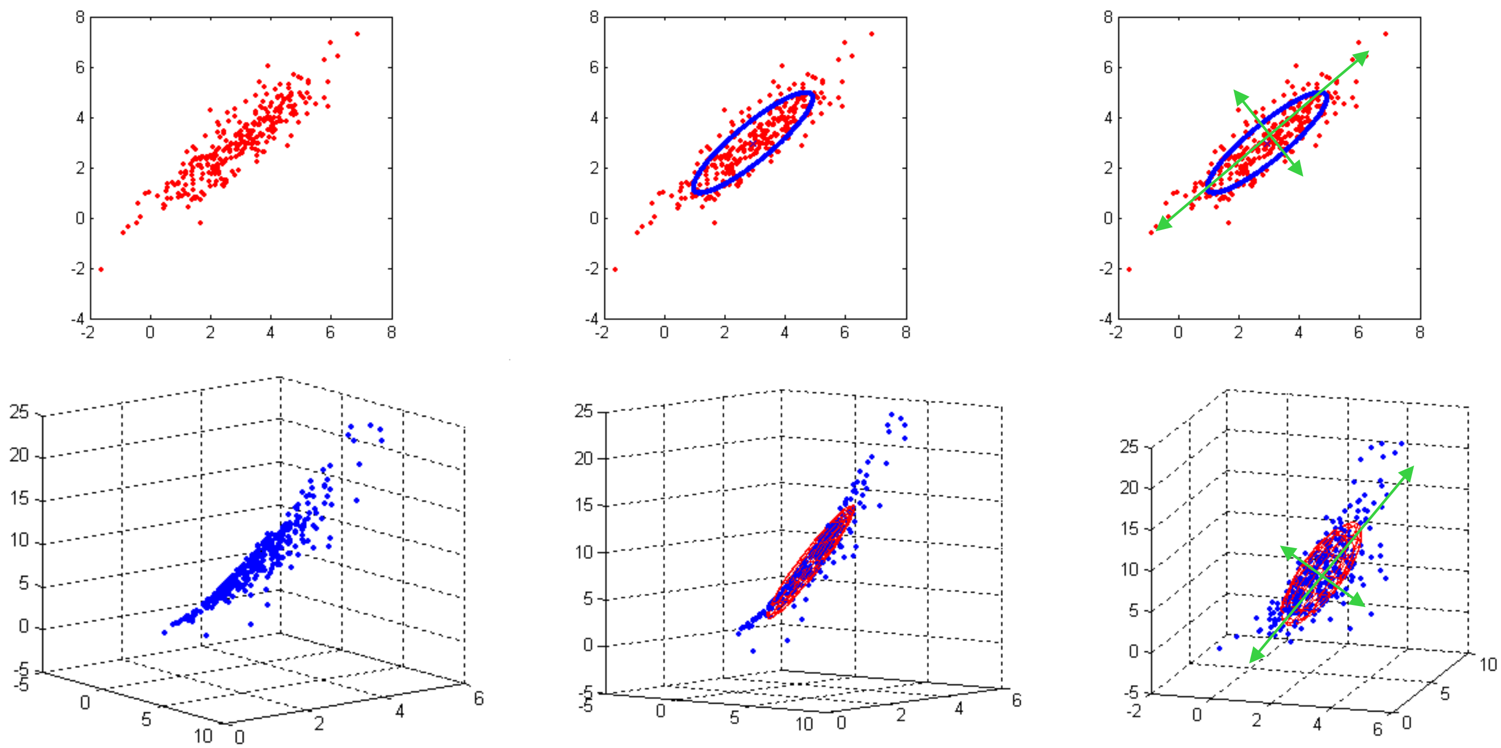
$$= \tfrac{N}{2}\Sigma - \tfrac{1}{2}\sum_{i=1}^{N}\left(x^i - \mu\right)\left(x^i - \mu\right)^T$$

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

Tony Jebara, Columbia University

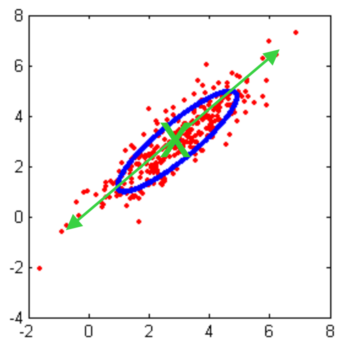
# Principal Components Analysis

- Problem: for high dimensional data, D is large
- Storing $\Sigma$, inverting $\Sigma^{-1}$ and determining $|\Sigma|$ are expensive!
- Idea: limit Gaussian model to directions of high variance
- Use Principal Components Analysis to mimic $\Sigma$

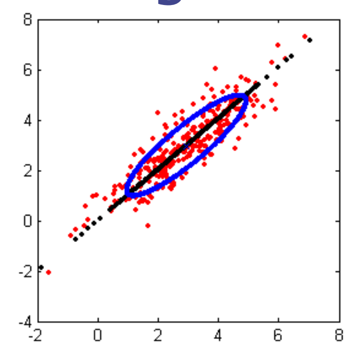


# Principal Components Analysis

- PCA approximates each datapoint as a mean vector plus steps along eigenvector directions. E.g. $c_i$ steps along v

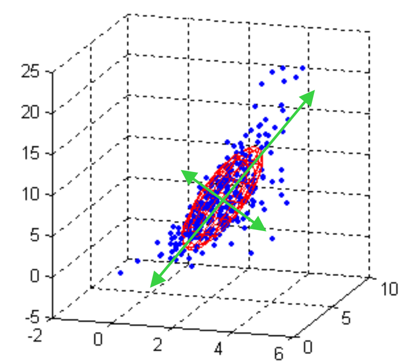



$$\begin{bmatrix} \vec{x}_i(1) \\ \vec{x}_i(2) \end{bmatrix} \approx \begin{bmatrix} \vec{\mu}(1) \\ \vec{\mu}(2) \end{bmatrix} + c_i \begin{bmatrix} \vec{v}(1) \\ \vec{v}(2) \end{bmatrix}$$




- More generally, PCA uses a set of eigenvectors M (where M<<D)

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




- PCA selects $\left\{\vec{\mu}, c_{ij}, \vec{v}_j\right\}$ to minimize $\sum_{i=1}^{N} \left\| \vec{x}_i - \hat{x}_i \right\|^2$
- The optimal directions are eigenvectors of covariance
- Which directions to keep: highest eigenvalues (variances)

# Principal Components Analysis

- Use eigenvectors, mean & coefficients to approximate data

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

- PCA finds eigenvectors by decomposing covariance matrix:

$$\Sigma = V \Lambda V^T = \sum_{i=1}^{D} \lambda_i \vec{v}_i \vec{v}_i^T$$

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

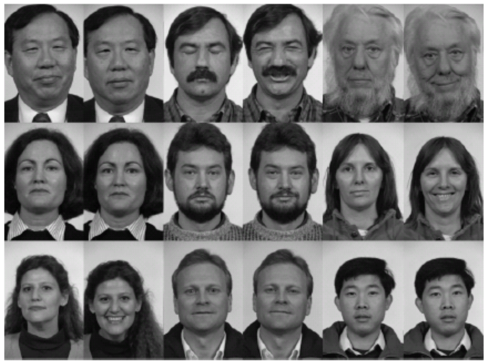
- Eigenvectors are orthonormal: $\vec{v}_i^T \vec{v}_j = \delta_{ij}$
- Eigenvalues are non-negative and non-increasing

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D \geq 0$$

- PCA selects the M eigenvectors with largest eigenvalues
- Truncating gives an approximate covariance: $\hat{\Sigma} = \sum_{i=1}^{M} \lambda_i \vec{v}_i \vec{v}_i^T$
- PCA finds coefficients by: $c_{ij} = \left(\vec{x}_i - \vec{\mu}\right)^T \vec{v}_j$

# PCA via the Snapshot Method

- Careful… how big is the covariance matrix?
- Assume 1000 images each containing D=20,000 pixels
- It is DxD pixels, that's unstoreable!
- Also, 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 $\Phi$ $where\ \Phi_{i,j} = x_i^T x_j$

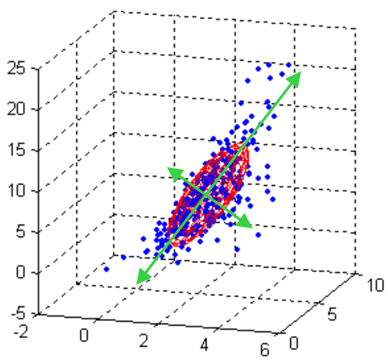
Then find eigendecomposition of Gram matrix $\Phi = \tilde{V}\Lambda\tilde{V}^T$

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

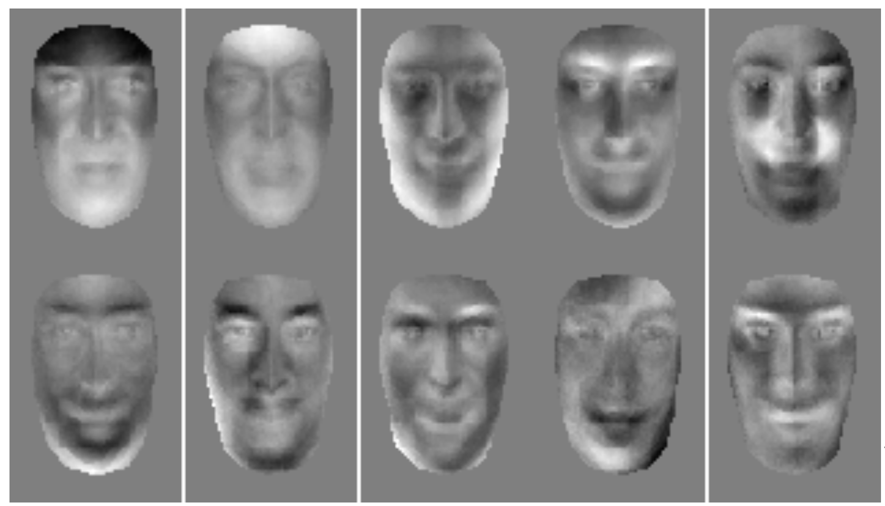
# PCA via the Snapshot Method

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

$\vec{\mu}$




$\vec{v}_1$



$\vec{v}_{10}$

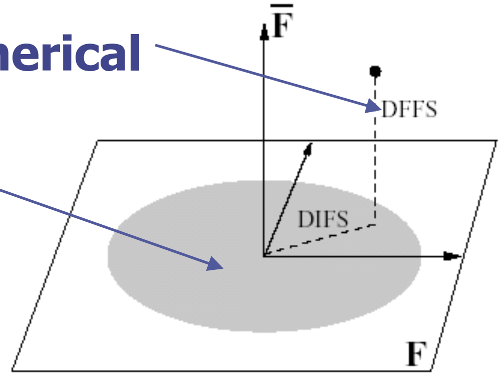
$$\left\{ \mu + \sum c_{1j} \vec{v}_j, \ldots \right\} =$$

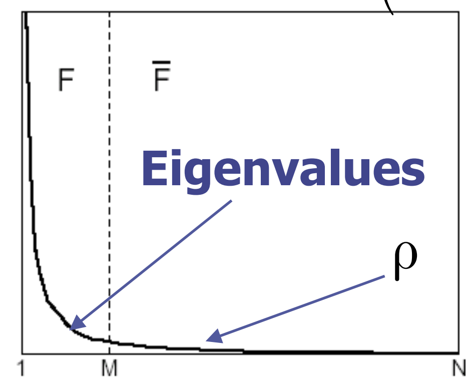
# Truncated Gaussian Detection

- Approximate $\Sigma$ with PCA plus spherical term via $p\left(x \mid \mu, \hat{\Sigma}\right)$

**Specific Eigenvalues & Eigenvectors**

**Spherical**




$$\Sigma = \sum_{k=1}^{D} \lambda_k v_k v_k^T \approx \hat{\Sigma}$$

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

$$\hat{\Sigma}^{-1} = \sum_{k=1}^{M} \tfrac{1}{\lambda_k} v_k v_k^T + \sum_{k=M+1}^{D} \tfrac{1}{\rho} v_k v_k^T$$

$$\hat{\Sigma}^{-1} = \sum_{k=1}^{M} \left(\tfrac{1}{\lambda_k} - \tfrac{1}{\rho}\right) v_k v_k^T + \tfrac{1}{\rho} I$$

$$\left|\hat{\Sigma}\right| = \prod_{k=1}^{M} \lambda_k \prod_{k=M+1}^{D} \rho$$

$$\begin{aligned}
\rho &= \tfrac{1}{D-M} \sum_{k=M+1}^{D} \lambda_k \\
&= \tfrac{1}{D-M} \left( \sum_{k=1}^{D} \lambda_k - \sum_{k=1}^{M} \lambda_k \right) \\
&= \tfrac{1}{D-M} \left( \sum_{k=1}^{D} \Sigma_{kk} - \sum_{k=1}^{M} \lambda_k \right)
\end{aligned}$$

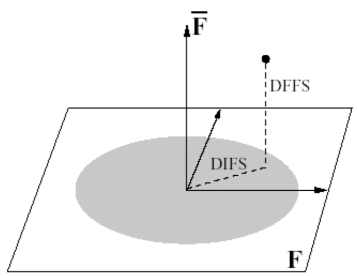
$$\Sigma_{kk} = \tfrac{1}{N} \sum_{i=1}^{N} \left( x_i(k) - \mu(k) \right)^2$$

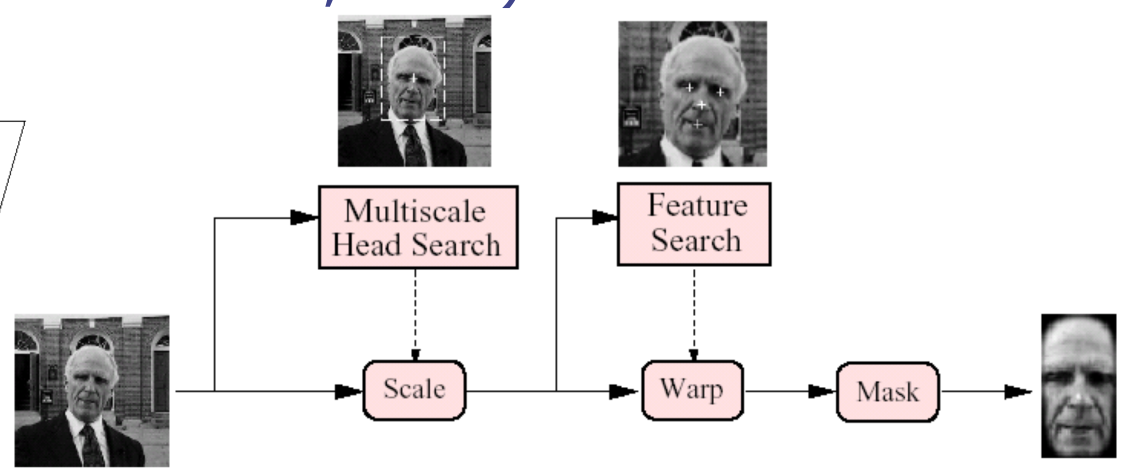
# Truncated Gaussian Detection

- Instead of minimizing squared error, use Gaussian model

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

- Use Snapshot PCA to efficiently store the big covariance
- This maximum likelihood Gaussian model achieved state of the art face finding as evaluated by NIST/DARPA (Moghaddam et al., 2000)




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

# Gaussian Face Recognition

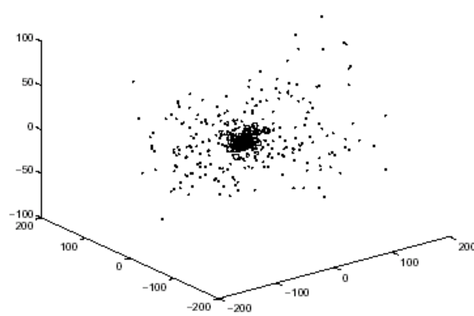
- Instead of modeling face images with Gaussian model the difference of two face images with a Gaussian
- Each difference of all pairs of images in our data is represented as a D-dimensional vector x
- Also have a binary label y, y=1 same person same, y=0 not

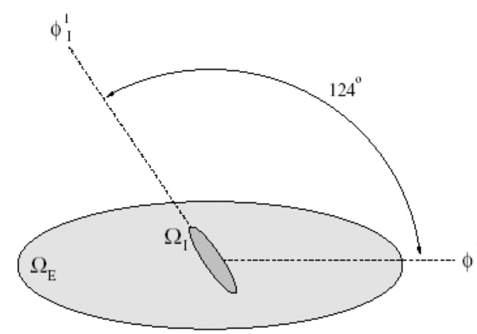
$$x \in R^D \quad y \in \{0,1\}$$

$y_i = 1, x_i =$  - 

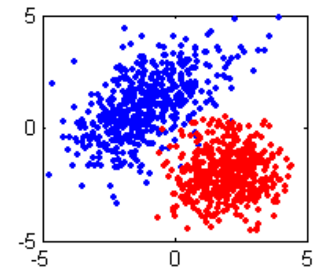
$y_j = 0, x_j =$  - 

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



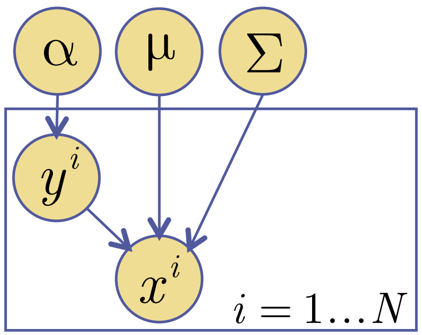



# Gaussian Classification




- Have two classes, each with their own Gaussian:

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




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

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

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

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

- Maximum Likelihood:

$$l=\sum_{i=1}^N \log p\left(x_i,y_i\mid\alpha,\mu,\Sigma\right)$$

$$=\sum_{i=1}^N \log p\left(y_i\mid\alpha\right)+\sum_{i=1}^N \log p\left(x_i\mid y_i,\mu,\Sigma\right)$$

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

# 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

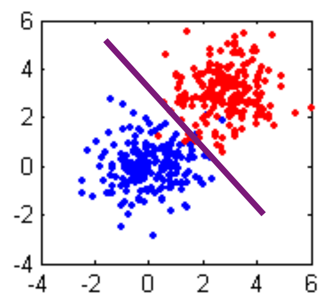
- Example cases, plotting decision boundary when = 0.5

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

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

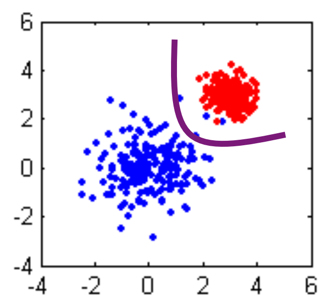
- If covariances are equal:

  linear decision



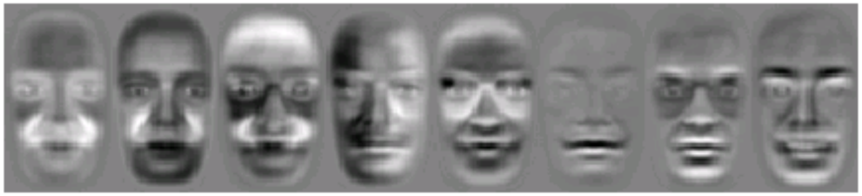

- If covariances are different:

  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:

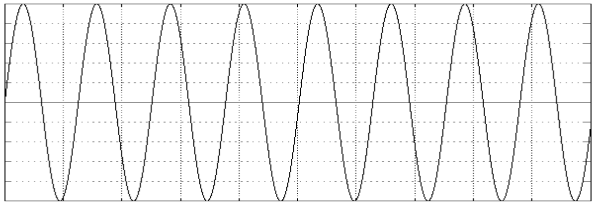
$$p\left(y=1 \mid x\right)=\frac{\alpha N\left(x \mid \mu_1, \Sigma_1\right)}{\left(1-\alpha\right)N\left(x \mid \mu_0, \Sigma_0\right)+\alpha N\left(x \mid \mu_1, \Sigma_1\right)}$$

# Other Standard Bases

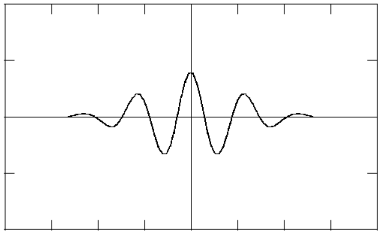
- There are other choices for the eigenvectors, not just PCA
- Could pick eigenvectors without looking at the data
- Just for their interesting properties

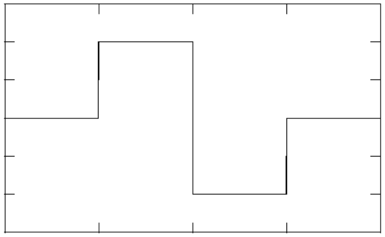
$$\vec{x}_i \approx \vec{\mu} + \sum_{j=1}^{C} c_{ij} \vec{v}_j$$

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



- Wavelet basis: localized or windowed Fourier





- PCA: optimal least squares linear dataset reconstruction

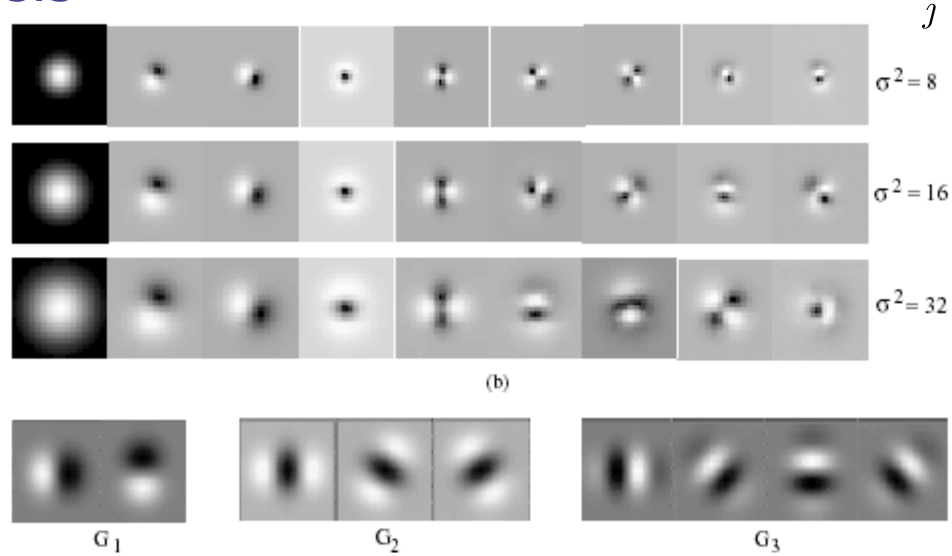
# Basis for Natural Images

$\vec{x}_i$

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

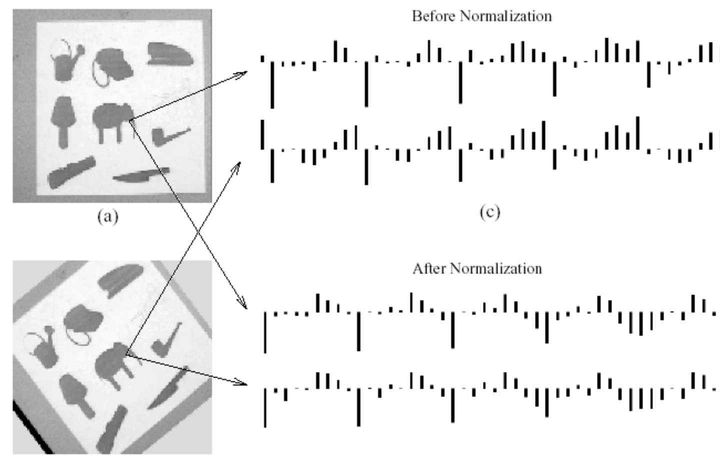
- Get difference of Gaussian bases
- Like Gabor or Wavelet basis
- Not specific like faces
- Multi-scale & orientation
- Also called steerable filters
- Similar to visual cortex

$\vec{v}_j$



# 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…




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