EECS 545: Machine Learning

Lecture 18. Unsupervised Learning: PCA

Honglak Lee

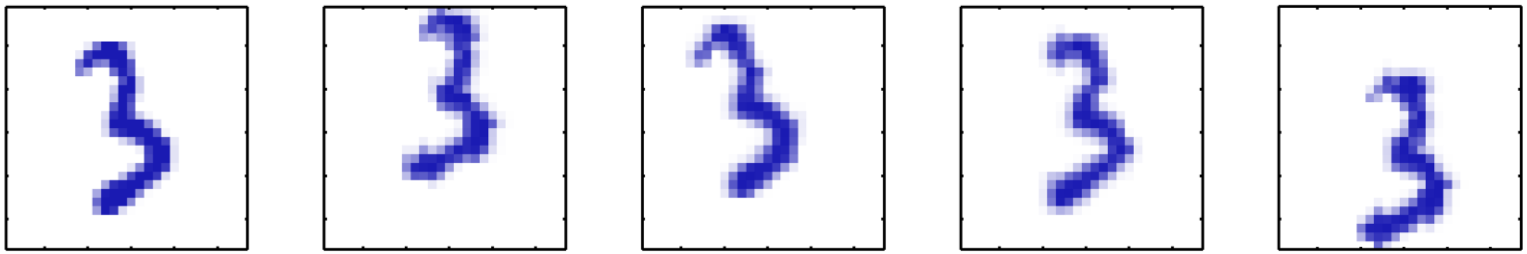
3/16/2011

Outline

- Principal Component Analysis

High-Dimensional Data

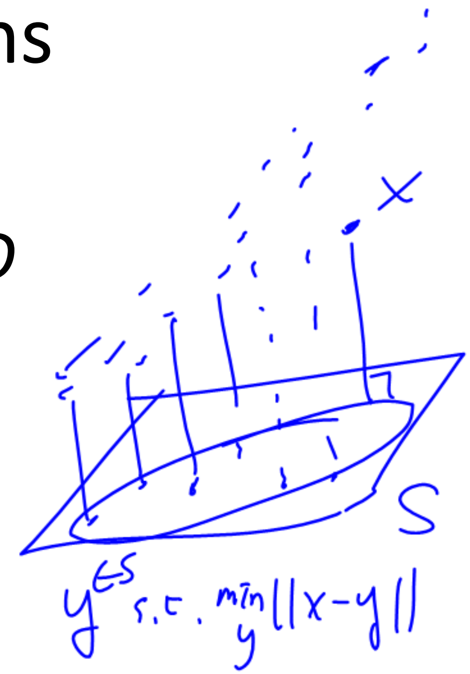
- . . . may have low-dimensional structure.



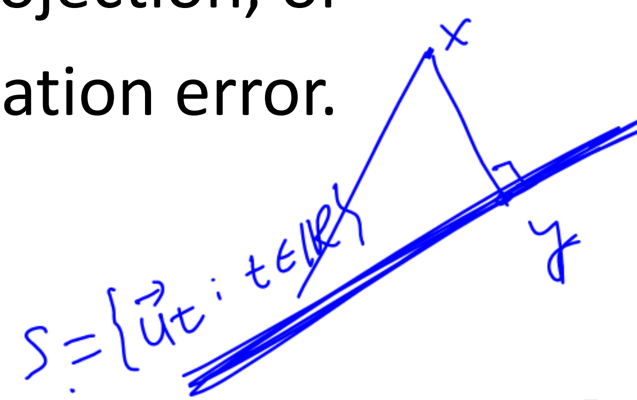
- The data is 100x100-dimensional.
- But there are only three degrees of freedom, so it lies on a 3-dimensional subspace.
  - (on a non-linear manifold, in this case)

Principal Component Analysis

- Given a set $X = \{x_n\}$ of observations
  – in a space of dimension D,
  – find a subspace of dimension $M < D$
  – that captures most of its variability.



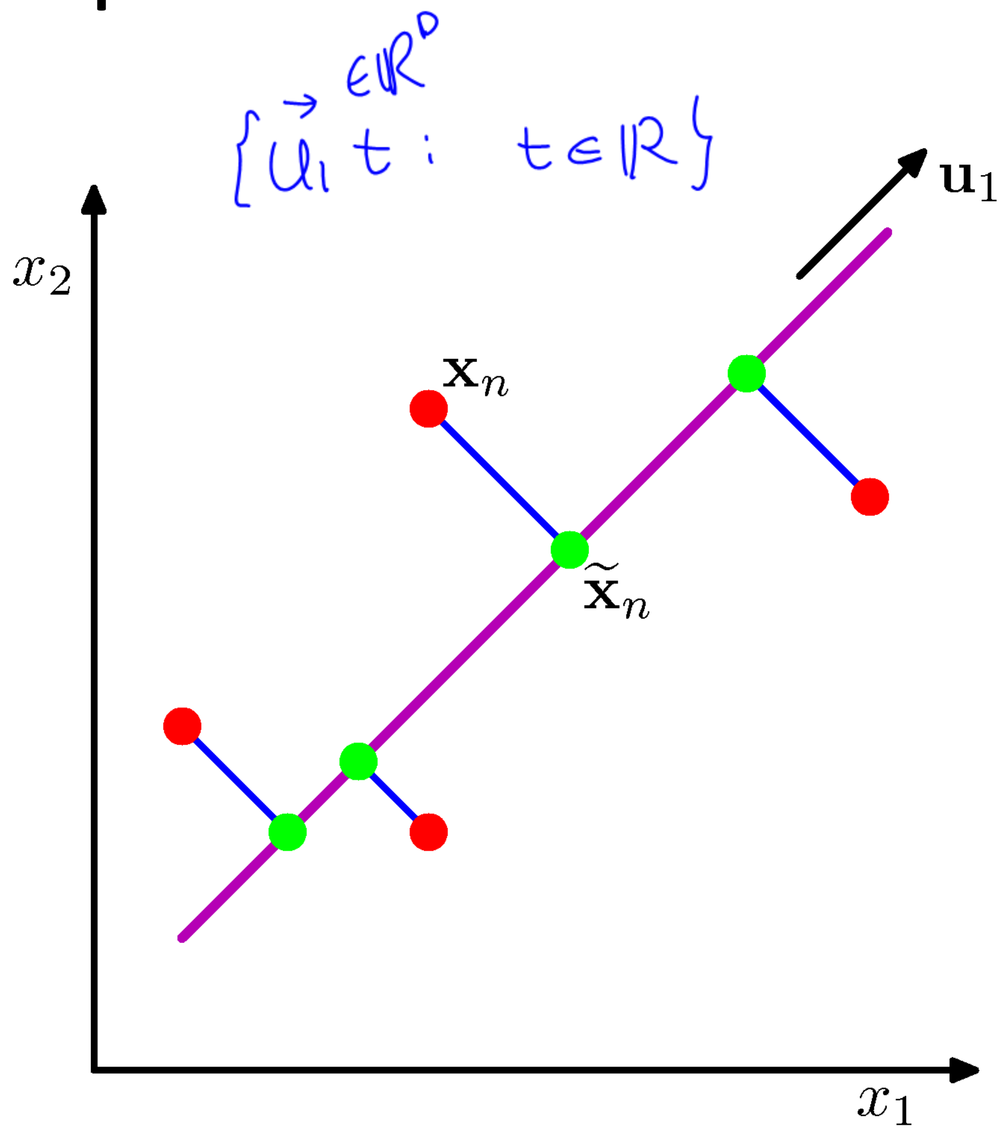
- PCA can be described as either:
  – maximizing the variance of the projection, or
  – minimizing the squared approximation error.



Two Descriptions of PCA

Approximate with the projection:

- Maximize variance, or
- Minimize squared error

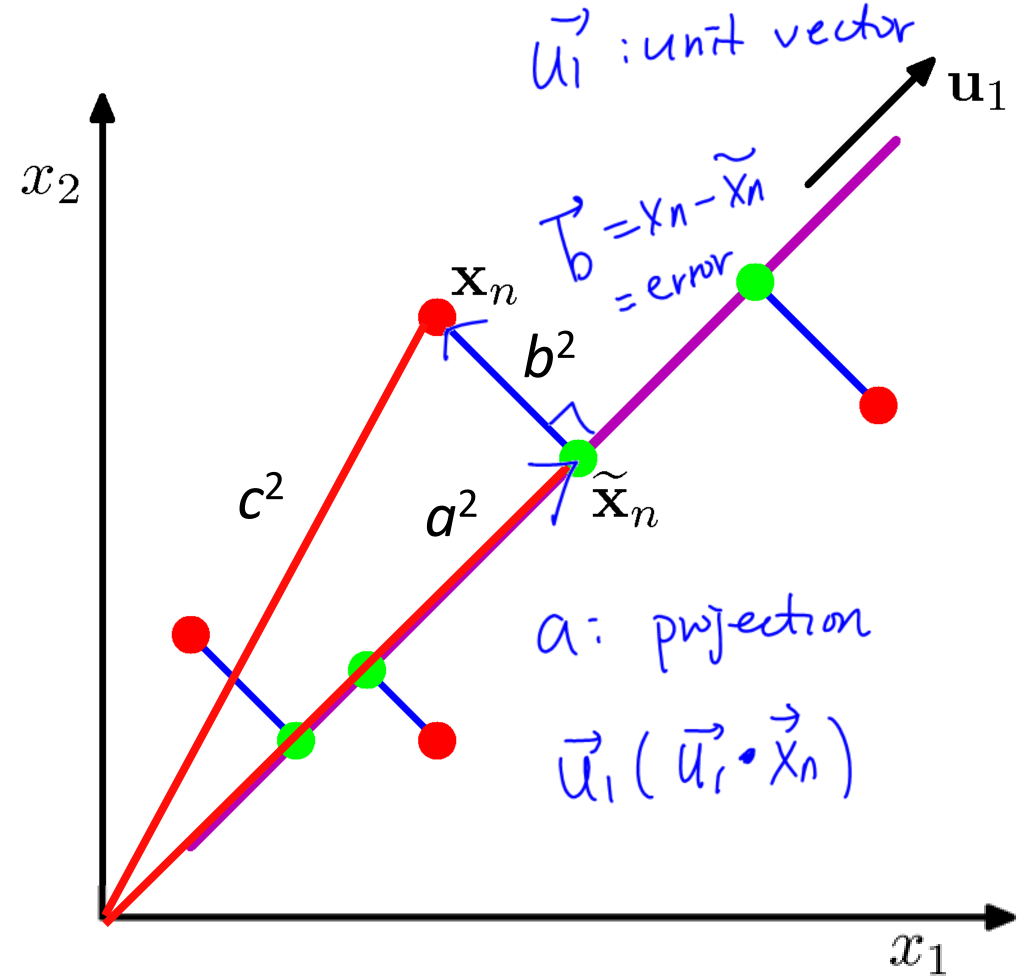


Equivalent Descriptions

- With mean at the origin

$$\sum_i c_i^2 = \sum_i a_i^2 + \sum_i b_i^2$$

- With constant $\sum_i c_i^2$
  - Minimizing $\sum_i b_i^2$
  - Maximizes $\sum_i a_i^2$
  - And vice versa



First Principal Component

- Given data points $\{x_n\}$ in D-dim space.
  - Mean $\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$

    $\mathbb{E}_{emp}\left[(x_n - \bar{x})(x_n - \bar{x})^T\right]$
  - Data covariance $\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T$
    - D x D matrix
- Let u_1 be the principal component we want.
  - Length 1: $\mathbf{u}_1^T\mathbf{u}_1 = 1$
  - Projection of $\mathbf{x}_n$: $\mathbf{u}_1^T\mathbf{x}_n$

    $\vec{u}_1 (\vec{u}_1^T x_n)$

First Principal Component

$\mathbb{E}_{emp}[(s-\bar{s})^2]$ $s_n = u_1^T x_n$

- Maximize the projection variance:

$$\frac{1}{N}\sum_{n=1}^{N}\{\mathbf{u}_1^T\mathbf{x}_n - \mathbf{u}_1^T\bar{\mathbf{x}}\}^2 = \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1$$

- Use a Lagrange multiplier to enforce $\mathbf{u}_1^T\mathbf{u}_1 = 1$
- Maximize: $\mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^T\mathbf{u}_1)$
- Derivative is zero when $\mathbf{S}\mathbf{u}_1 = \lambda_1\mathbf{u}_1$
  - That is $\mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 = \lambda_1$
- So u_1 is eigenvector with largest eigenvalue.

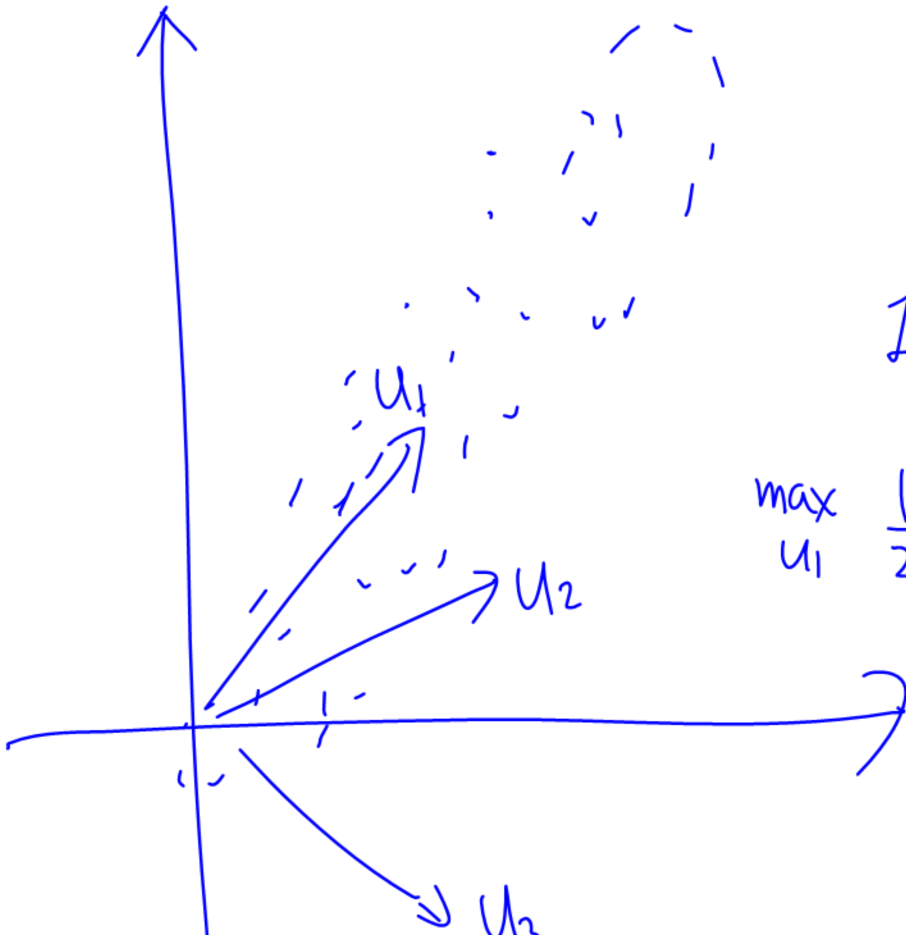
$$\frac{1}{N}\sum_{n=1}^{N}\left(u_1^T x_n - u_1^T \bar{x}\right)^2$$

$$= \frac{1}{N}\sum_n \left[\underbrace{u_1^T (x_n - \bar{x})}_{\text{Scalar}}\right]^2$$

$$= \frac{1}{N}\sum_n \left(u_1^T (x_n - \bar{x})\right)\left(u_1^T (x_n - \bar{x})\right)^T$$

$$= \frac{1}{N}\sum_n u_1^T (x_n - \bar{x})(x_n - \bar{x})^T u_1$$

$$= u_1^T \underbrace{\left[\frac{1}{N}\sum_n (x_n - \bar{x})(x_n - \bar{x})^T\right]}_{S} u_1$$



$u_1 \in \mathbb{R}^D$

$$\max_{u_1} \quad u_1^T S u_1$$

$$\text{s.t.} \quad u_1^T u_1 - 1 = 0$$

Lagrange multiplier.

$$\max_{u_1} \frac{1}{2}\Big(\underbrace{u_1^T S u_1 - \lambda\,(u_1^T u_1 - 1)}_{= \mathcal{L}(u_1, \lambda)}\Big)$$

$$\Rightarrow \nabla_{u_1} \mathcal{L} = S u_1 - \lambda u_1 = 0$$

$$\Rightarrow S u_1 = \lambda u_1$$

$$\text{s.t.} \quad u_1^T u_1 = 1$$

PCA by Maximizing Variance

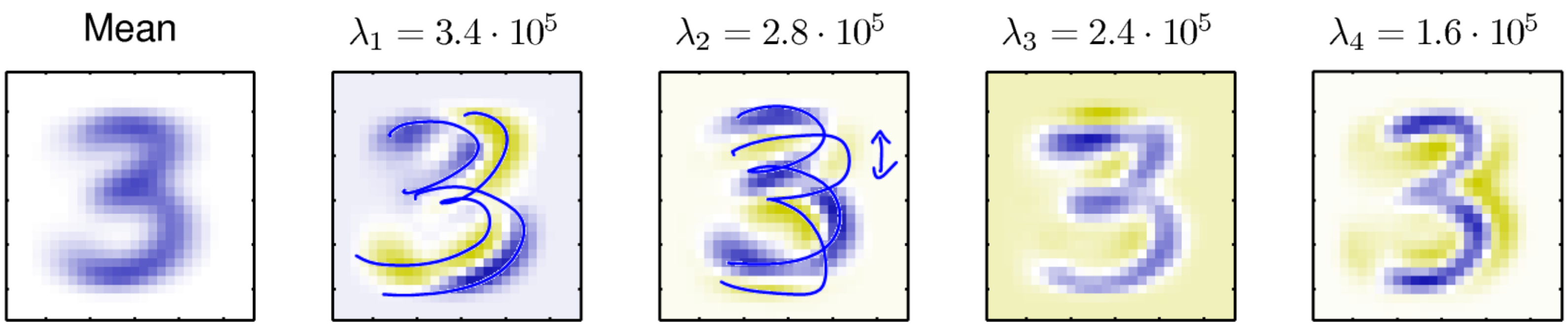
- Repeat to find the M eigenvectors of the data covariance matrix S corresponding to the M largest eigenvalues.

- We can do the same thing by minimizing the squared error of the projection.
  – Homework

Digit Image Example

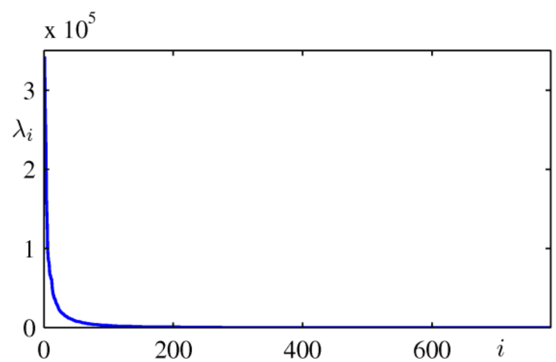
10,000

- The mean and first four PCA eigenvectors.



- The eigenvalue spectrum:

$$X = \begin{array}{c} \begin{bmatrix} \leftarrow \vec{x}_1 \rightarrow \\ \leftarrow \vec{x}_2 \rightarrow \\ \vdots \\ \leftarrow \vec{x}_N \rightarrow \end{bmatrix} N \\ \leftarrow D \rightarrow \end{array}$$



(a)

$X = X - \text{mean}(X)$

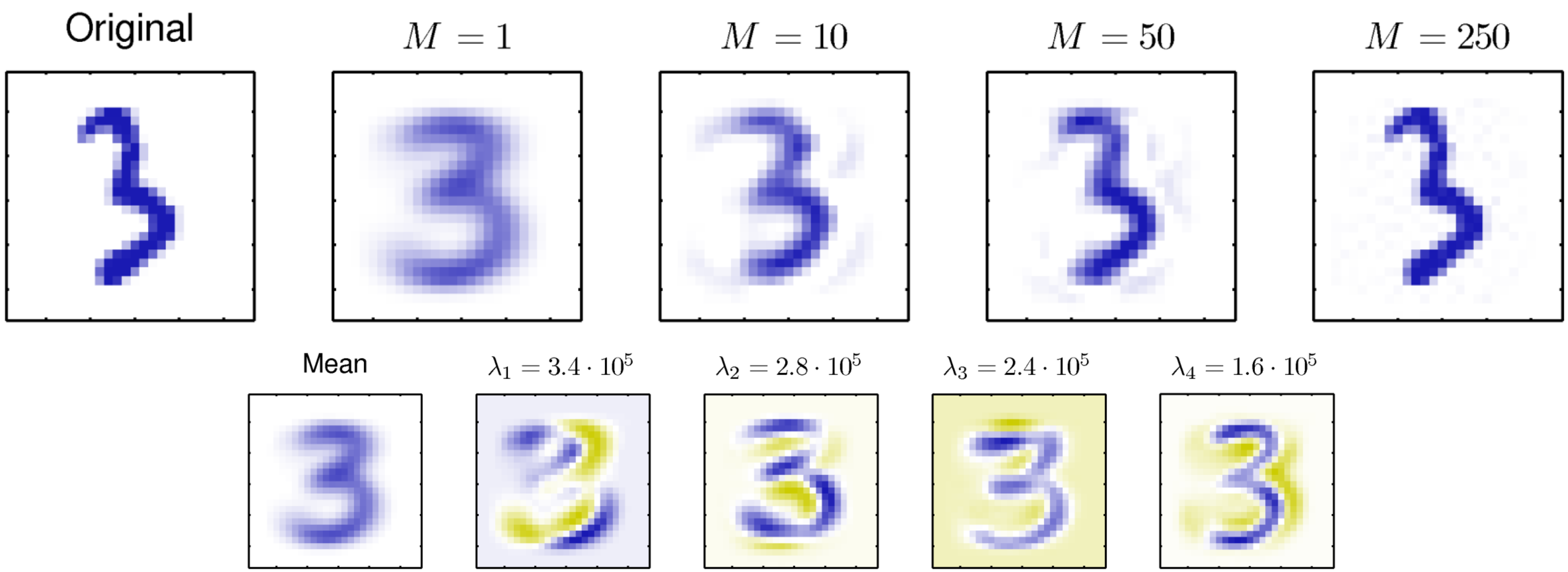
$C = \frac{1}{N} X^T X$

$[V\ D] = \text{eig}(C)$

$\Rightarrow \quad V = \begin{bmatrix} | & | & & | \\ v_1 & v_2 & \cdots & v_M \\ | & | & & | \end{bmatrix}$

Reconstructing the Image

- Compress the image representation by using only first M eigenvectors, and discarding the less important information.



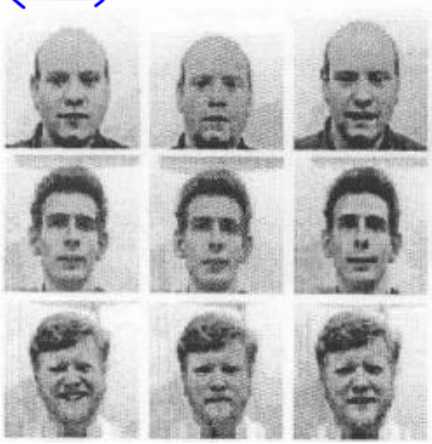
Learning features via PCA

- Example: Eigenfaces

Training face images

32

32

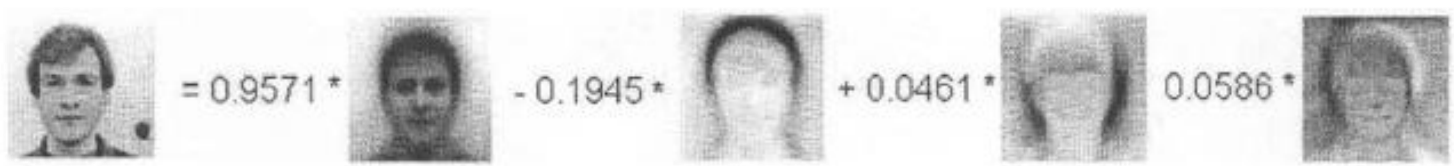


32^2 pixels $= 1024$ pixels

Learned PCA bases

$x \approx \overset{*}{a}_1 + \overset{*}{a}_2 + \overset{*}{a}_3 + \overset{*}{a}_4$

Test example



Images from www.cse.unr.edu/~bebis/CS485/Lectures/Eigenfaces.ppt

PCA

$$Pc_i = u_i^T x$$

PCA whitening

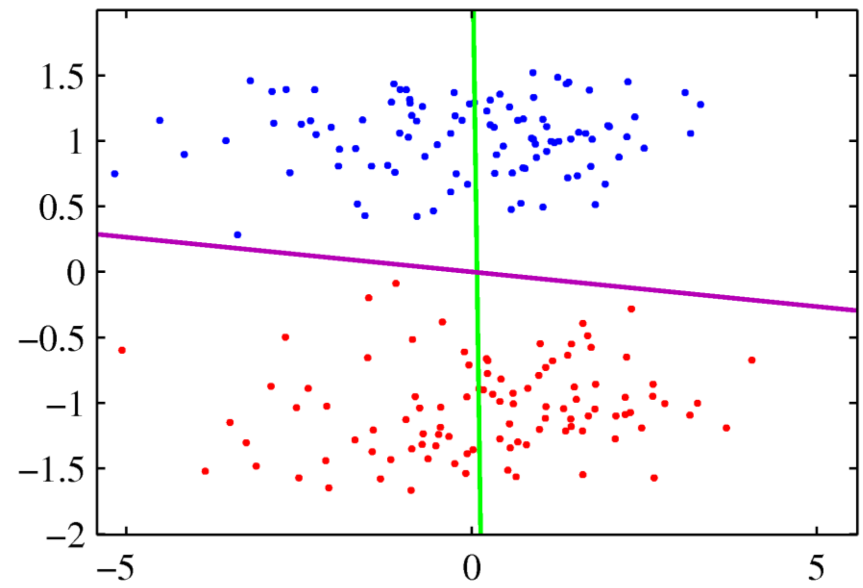
$$PC\_whiten_i = \frac{1}{\sqrt{\lambda_i}} u_i^T x \qquad \lambda_i : i\text{th eigenvalue of } C.$$

$$\frac{1}{\sqrt{\lambda_i + \epsilon}}$$

$$\epsilon = 0.01$$

Limits to PCA

- Maximizing variance is not always the best way to make the structure visible.
- PCA vs Fisher's linear discriminant



Probabilistic PCA

- We can view PCA as solving a probabilistic latent variable problem.
- Describe a distribution $p(\mathrm{x})$ in D-dimensional space, in terms of a latent variable z in M-dimensional space.

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \mu + \epsilon \qquad\qquad p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$$

- W is a DxM linear transformation from z to x

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \mu, \sigma^2\mathbf{I})$$

Probabilistic PCA

- Given the generative model

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \mu + \epsilon$$

- we can infer

$$E[\mathbf{x}] \quad = \quad E[(\mathbf{W}\mathbf{z} + \mu + \epsilon)] = \mu$$

$$\begin{aligned} cov[\mathbf{x}] &= E[(\mathbf{W}\mathbf{z} + \epsilon)(\mathbf{W}\mathbf{z} + \epsilon)^T] \\ &= E[\mathbf{W}\mathbf{z}\mathbf{z}^T\mathbf{W}^T] + E[\epsilon\epsilon^T] = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I} \end{aligned}$$

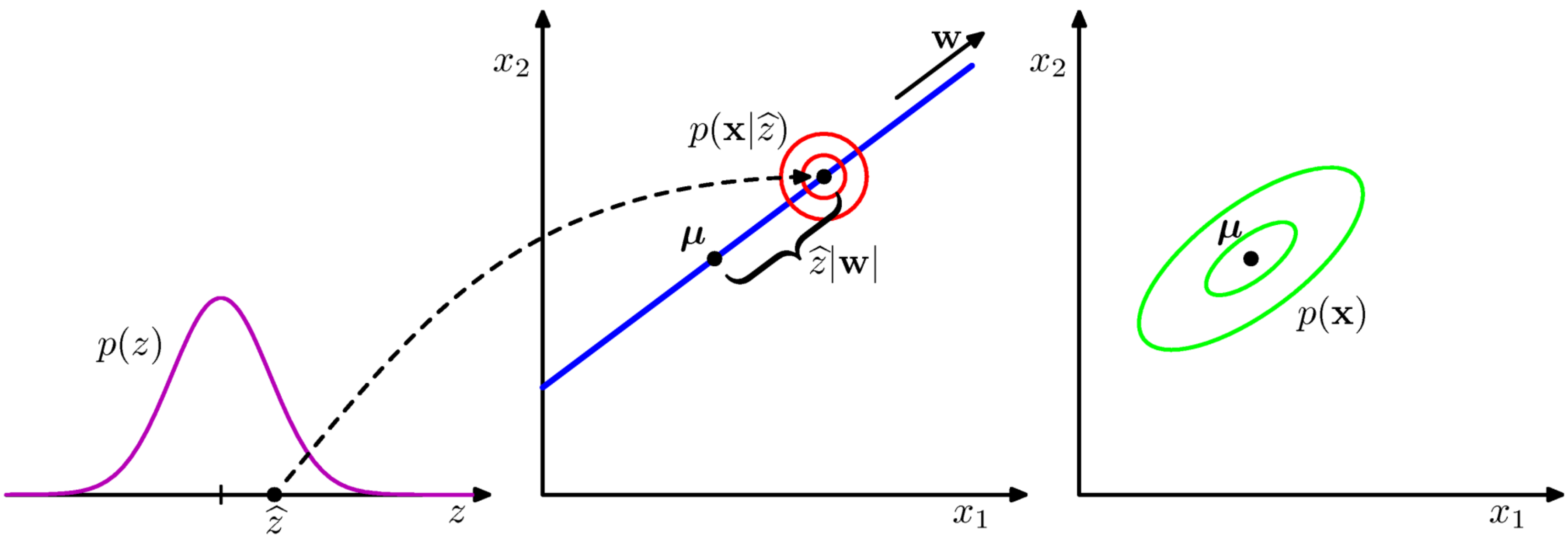
Q. Verify this

Probabilistic PCA

- The generative model

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \mu + \epsilon$$

- can be illustrated



Likelihood of Probabilistic PCA

- (Marginal) likelihood

$$\ln p(X|\mu, W, \sigma^2) = \sum_n p(x_n|W, \mu, \sigma^2)$$

$$= -\frac{ND}{2}\ln 2\pi - \frac{N}{2}\ln|C| - \frac{1}{2}\sum_n (x_n - \mu)^T \ C^{-1}(x_n - \mu)$$

$$where\, C = WW^T + \sigma^2 I$$

- We can simply maximize this likelihood function with respect to μ, W, σ.

Maximum Likelihood Parameters

- Mean: $\mu = \bar{\mathbf{x}}$
- Noise: $\sigma^2_{ML} = \frac{1}{D-M} \sum_{i=M+1}^{D} \lambda_i$
- W: $\mathbf{W}_{ML} = \mathbf{U}_M(\mathbf{L}_M - \sigma^2\mathbf{I})^{1/2}\mathbf{R}$
  - where $\mathbf{L}_M$ is diag with the *M* largest eigenvalues
  - and $\mathbf{U}_M$ is the M corresponding eigenvectors
  - And **R** is an arbitrary *M*x*M* rotation

Maximum likelihood by EM

- Latent variable model

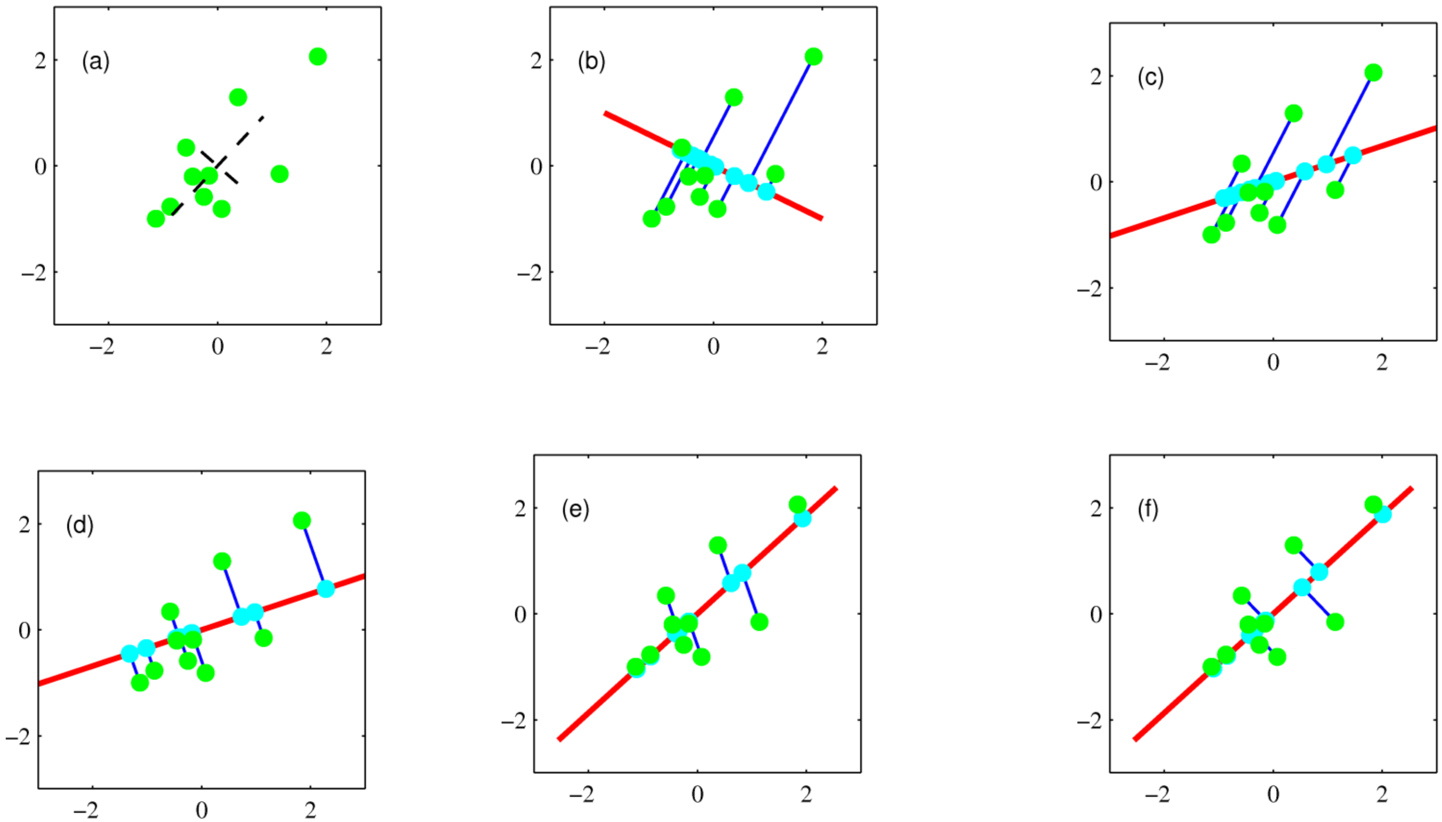
$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$$

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \mu, \sigma^2\mathbf{I})$$

- E-step: Estimate the posterior Q(z)=P(z|x)
  - Use linear Gaussian
- M-step: Maximize the data-completion likelihood given Q(z):

$$maximize_{\theta=\{\mu,W,\sigma\}} \sum_i \sum_{z^{(i)}} Q\left(z^{(i)}\right) \log P_\theta\left(x^{(i)}, z^{(i)}\right)$$

Finding PCA params by EM



- Illustrating EM on simulated data

Bayesian PCA (sketch)

- Note that the maximum likelihood for probabilistic PCA is still a point estimate on W.
- Main idea of Bayesian PCA: Put a prior on W

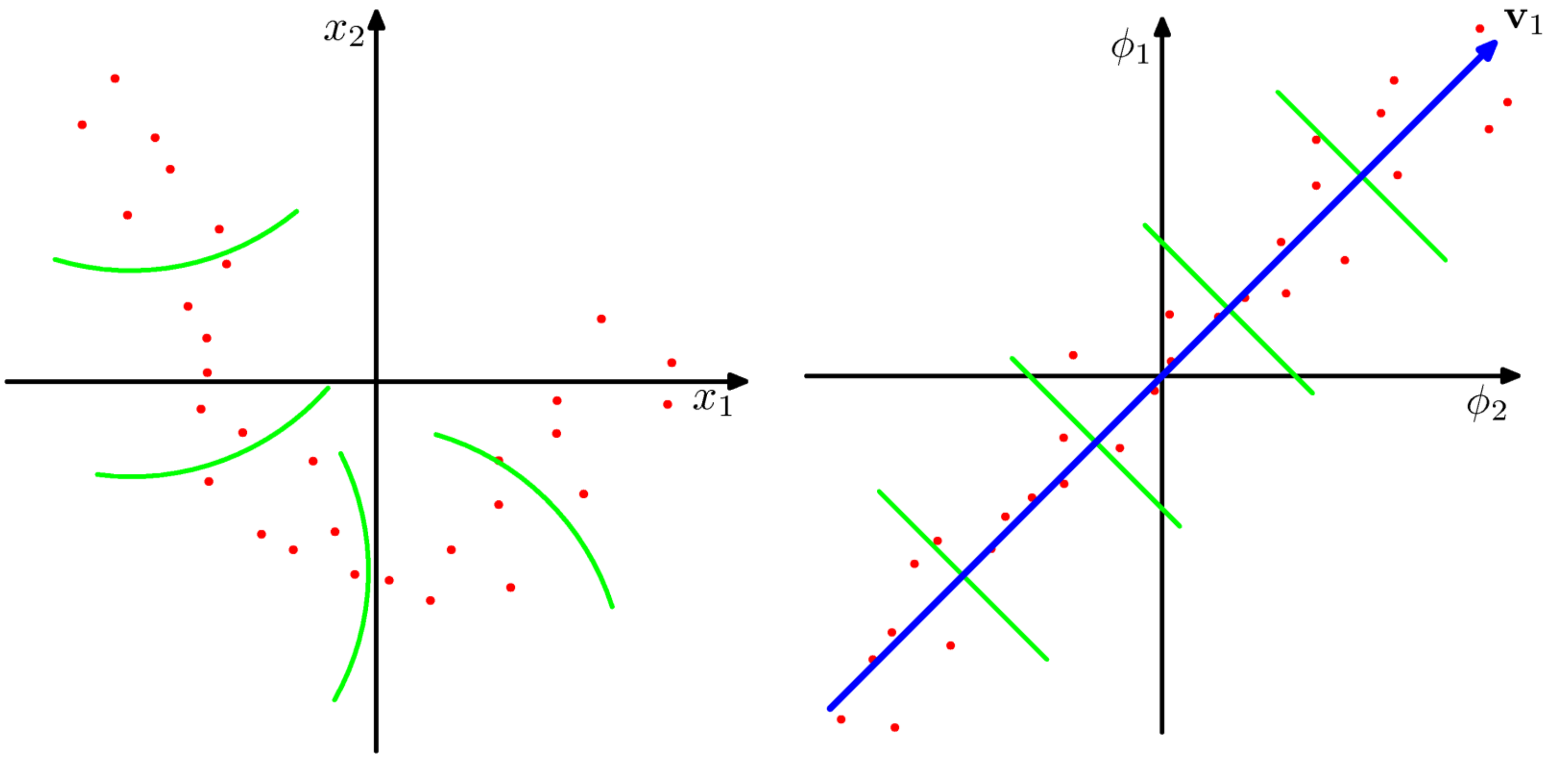
$$p(W|\alpha) = \prod_i \left(\frac{\alpha_i}{2\pi}\right)^{\frac{D}{2}} \exp(-\frac{1}{2}\alpha_i w_i^T w_i)$$

- Maximize the marginal likelihood (i.e., marginalize W)

$$p(X|\alpha, \mu, \sigma^2) = \int \; p(X|W, \mu, \sigma^2) p(W|\alpha) d\alpha$$

Kernel PCA

- Suppose the regularity that allows dimensionality reduction is non-linear.



Kernel PCA

- As with regression and classification, we can transform the raw input data $\{x_n\}$ to a set of feature values

$$\{\mathbf{x}_n\} \longrightarrow \{\phi(\mathbf{x}_n)\}$$

- Linear PCA gives us a linear subspace in the feature value space, corresponding to nonlinear structure in the data space.

Kernel PCA

- Define a kernel, to avoid having to evaluate the feature vectors explicitly.

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$$

- Define the Gram matrix K of pairwise similarities among the data points:

$$K_{nm} = \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$$

- Express PCA in terms of the kernel,
  – Some care is required to centralize the data.

Next

- Next, from Bishop:
  - Non-linear dimensionality reduction
  - Hidden Markov Models, Dynamical Systems
- Then, Reinforcement Learning
  - From the Sutton & Barto book