CS 383 – Machine Learning

Dealing w/ Data

Part 2

Slides adapted from material created by E. Alpaydin
Prof. Mordohai, Prof. Greenstadt, Pattern Classification (2^{nd} Ed.),
Pattern Recognition and Machine Learning

Objectives

- Feature Projection (PCA, LDA)
- Springer Text: 6.1, 6.3.0-6.3.1, 6.4.0-6.4

Additional Resources

- http://people.cs.pitt.edu/~milos/courses/cs3750-Fall2011/lectures/class16.pdf

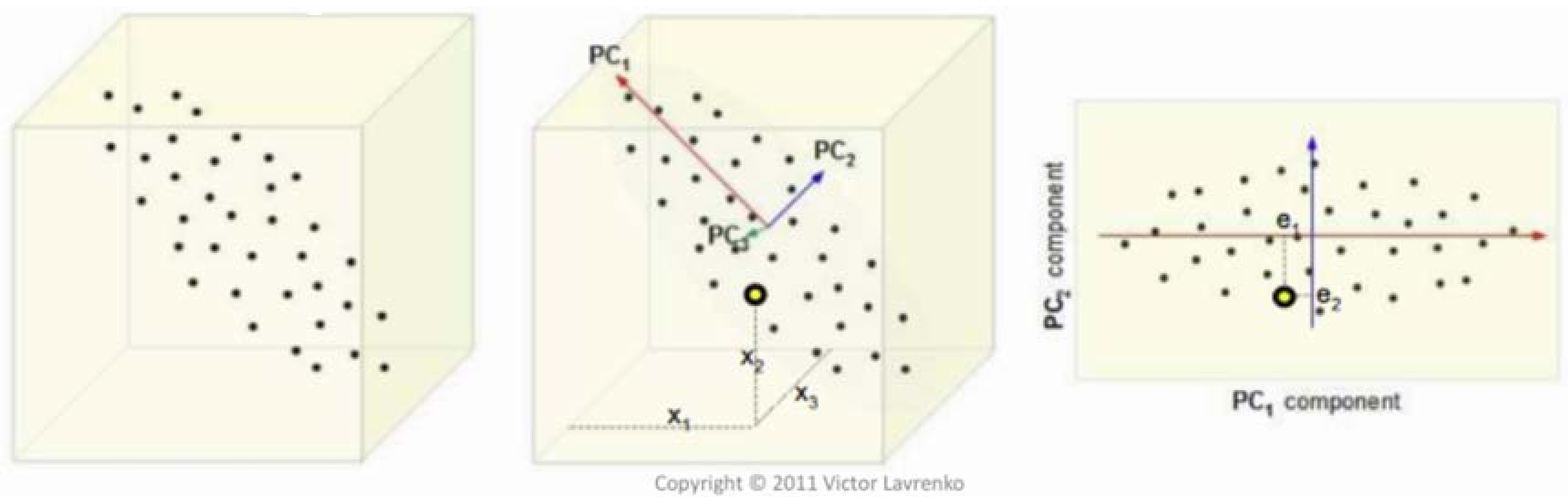
Feature Projection

Feature Construction

- We said another way to reduce the dimensionality of our data is to construct new features from the existing ones.
- Hopefully we can some “encode” information from higher dimensional data in these new features so that we don’t lose much information
- In particular we’ll look at the idea of *feature projection*

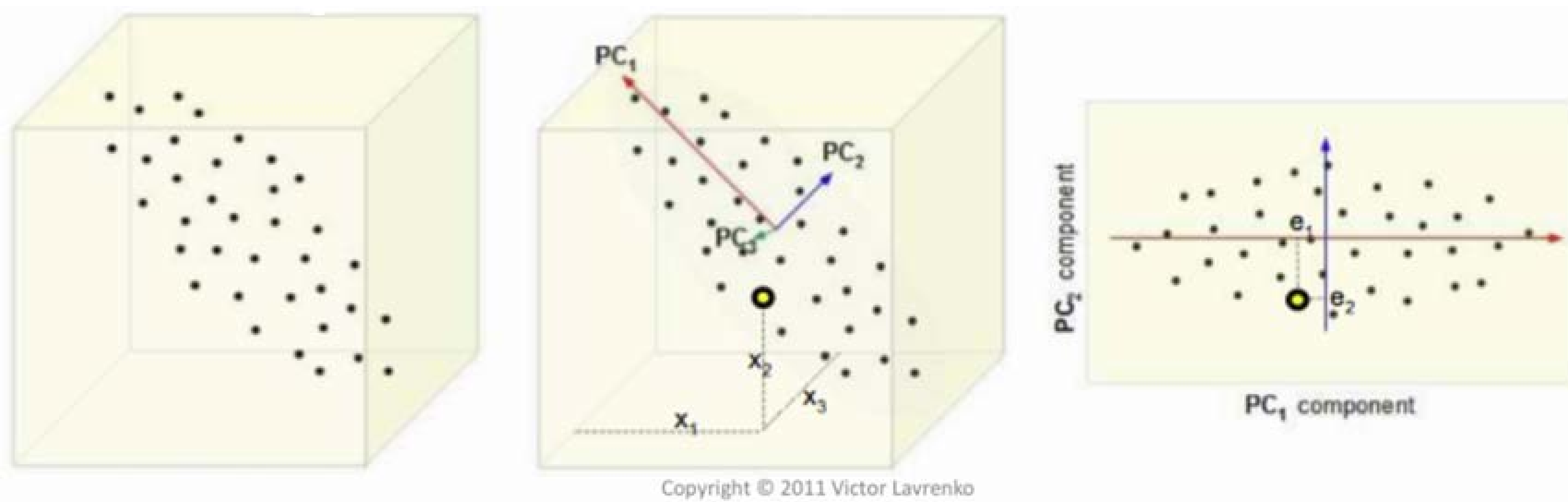
Feature Projection by PCA

- Principal component analysis (PCA) defines a set of principal components (basis)
  - 1st: direction of the greatest variability in the data
  - 2nd: perpendicular to 1st, greatest variability of what's left
  - Etc… until D, the original dimensionality



Feature Projection by PCA

- We can then choose the number of dimensions we want, $k < D$ and project the original data onto the principal components
- Each projection will result in a point on that axis, resulting in an new k-dimensional feature vector



PCA Derivation

- We want to find new points $Z = Xw$
- Start with the first new dimension:
  - Assume w is a $D \times 1$ column vector

$$Z = Xw$$

- We want to maximize the variance of Z

$$argmax_w\left(Var(Z)\right)$$
$$= arg\max_w\left(Var(Xw)\right)$$
$$= arg\max_w\left(w^{\mathrm{T}}\Sigma w\right)$$

- Subjecting this to the constraint that $w^T w = 1$ we get the Lagrange problem:

$$\mathrm{argmax}_w\left(\mathrm{w}^{\mathrm{T}}\Sigma w - \alpha(w^T w - 1)\right)$$

PCA Derivation

$$\text{argmax}_w \left(\mathrm{w}^{\mathrm{T}}\Sigma w - \alpha(w^T w - 1)\right)$$

- Taking the derivative and setting this equal to zero we get:
$$2\Sigma w - 2\alpha w = 0$$
- Which becomes $(\Sigma - \alpha I)w = 0$
- We could also write this as $\Sigma w = \alpha w$ and since Σ is a matrix, α is a scalar, and the thing that we're solving for, w, is a vector, this can be solved using eigen-decomposition!

Eigenvalues/Eigenvectors

- While you'll usually just use some package to get the eigenvalues and eigenvectors for you, let's make sure we can do it manually as well.
- The following is just a repeat of what in the Week 0 Linear Algebra slides.

Eigenvalues/Eigenvectors

- The basic equation in the eigenvalue problem is:
$$Ax = \lambda x$$
- Where A is a square matrix, x is a vector, and λ is a scalar
- The vector x is called an *eigenvector* and the scalar λ is called an *eigenvalue*
- Real eigenvalues only exist if A is a square matrix and the determinant of $A - \lambda \mathrm{I}$ is equal to zero

Finding Eigenvalues/Eigenvectors

- To find the eigen-values and vectors let's use that constraint on the determinant:

$$|A - \lambda \mathrm{I}| = 0$$

- Let's look at finding them for a 2x2 matrix
  - $|A - \lambda I| = \begin{vmatrix} a_{11} - \lambda & a_{12} \\ a_{21} & a_{22} - \lambda \end{vmatrix}$
  - $= (a_{11} - \lambda)(a_{22} - \lambda) - a_{12}a_{21} = 0$
  - $\lambda^2 - \lambda(a_{11} + a_{22}) + (a_{11}a_{22} - a_{12}a_{21}) = 0$
- We want to solve for λ
  - What type of equation is this?
  - How can we solve for it?
- Once we know the values of λ we can then solve for the eigenvectors x
  - For each value of λ take the original equation $Ax = \lambda x$ and solve the equation $(A - \lambda I)x = 0$

Eigenvalues/Vectors

- The general procedure (called eigen-decomposition) is:
    1. Compute $|A - \lambda I|$
    2. Find the roots of the polynomial given by $|A - \lambda I| = 0$
    3. Solve the system of equations $(A - \lambda I)x = 0$
- In MATLAB you can get these by:
    - `[V,D]=eig(A)`
    - The columns of V will be the eigen-vectors
    - You also may want to normalize the eigen-vectors to be of unit length: $V_{:,i}/|V_{:,i}|$

Choosing k

- Next we want to choose the k eigenvectors that correspond to the k highest eigenvalues
- How do we decide k?
  - User?
  - Fraction of variation explained by first k principle components:

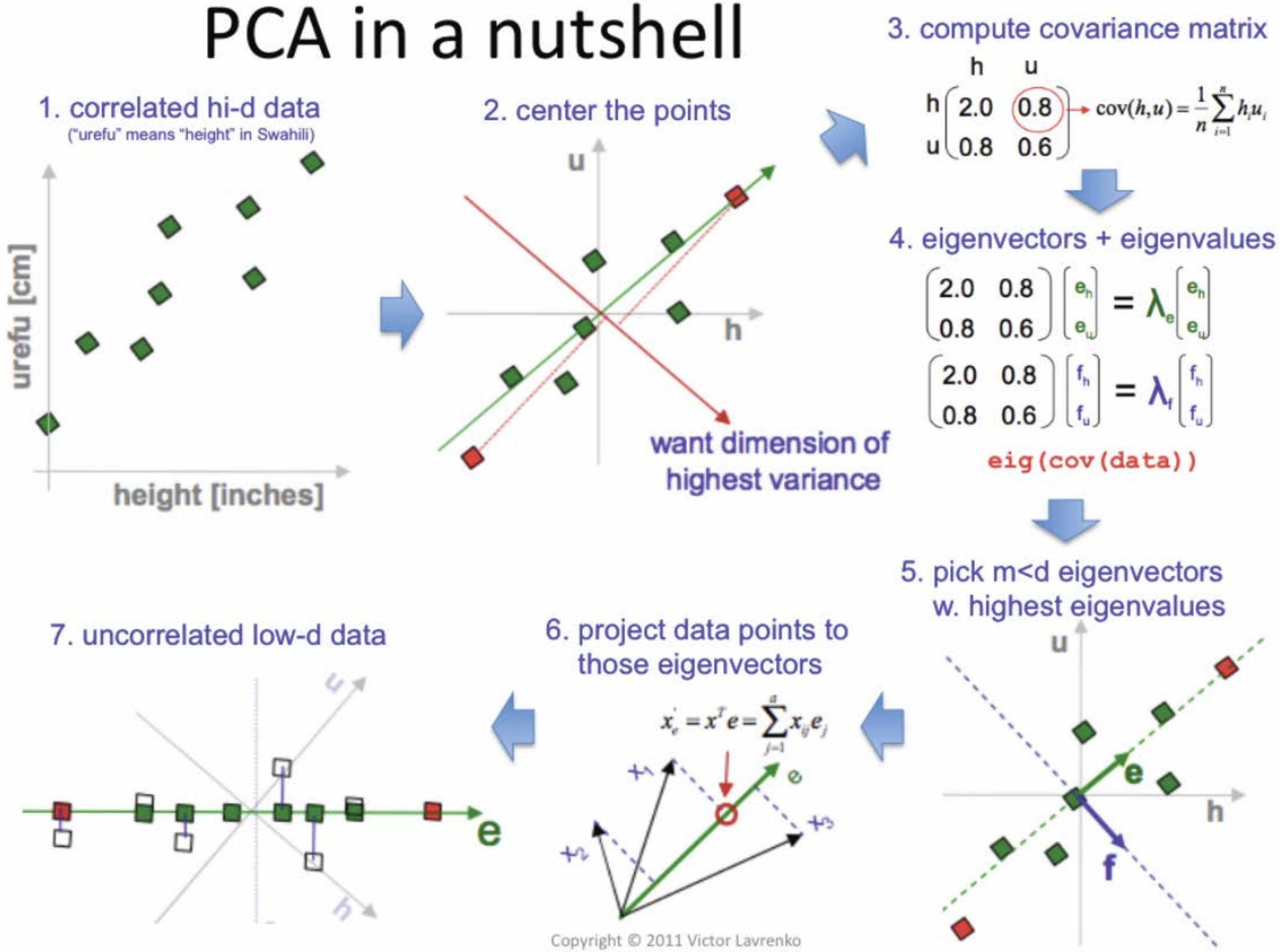
$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i} \geq \alpha$$

    - Typical threshold α values 0.9 or 0.95

Using PCA for Dimensionality Reduction

- Now we have a set of k principal components $e_1, \dots, e_k$
  - Orthogonal, unit length
- Concatenated, they form an $N \times k$ *projection matrix* $W = [e_1, \dots, e_k]$
- Now can *project* our D-dimensional data into k-dimensions
  - $Z = XW$

PCA in a nutshell



Example

- Assume data

  X={(4,1),(2,4),(2,3),(3,6),(4,4),(9,10),(6,8),(9,5),(8,7),(10,8)}
- What is the first principal component?
- What are the features projected onto that component?

Example

- Assume data

  X={(4,1),(2,4),(2,3),(3,6),(4,4),(9,10),(6,8),(9,5),(8,7),(10,8)}

1. First lets standardize our data
   - μ_1=(4+2+2+3+4+9+6+9+8+10)/10 = 5.7
   - μ_2= (1+4+3+6+4+10+8+5+7+8)/10 = 5.6
   - $\sigma_1 = 3.093, \sigma_2 = 2.7162$
   - So our new (standardized) data is
     $X = [(-0.55, -1.69), (-1.2, -0.59), (-1.2, -0.96)$
     $(-0.87, 0.147), (-0.55, -0.59), (1.067, 1.62), (0.097, 0.88)$
     $(1.067, -0.22), (0.74, 0.515), (1.39, 0.88)]$

Example

2. Compute covariance matrix

$$\Sigma(X) = \sigma(X, X)$$

- This is actually quite easy if we have already centered the data!
  - $\Sigma(X) = \frac{X^T X}{N-1} = \begin{bmatrix} 1.0 & 0.6851 \\ 0.6851 & 1.0 \end{bmatrix}$

Example

$$\Sigma(X) = \frac{X^T X}{N-1} = \begin{bmatrix} 1.0 & 0.6851 \\ 0.6851 & 1.0 \end{bmatrix}$$

3. Compute the Eigenvalues/vectors of the covariance matrix
   - Eigenvalues = [0.3149, 1.6851]
   - Eigenvectors
     - $[-0.7071, 0.7071]^T$
     - $[0.7071, 0.7071]^T$

Example

$$\chi = [(-0.55, -1.69), (-1.2, -0.59), (-1.2, -0.96) \\ (-0.87, 0.147), (-0.55, -0.59), (1.067, 1.62), \\ (0.097, 0.88), (1.067, -0.22), (0.74, 0.515), (1.39, 0.88)]$$

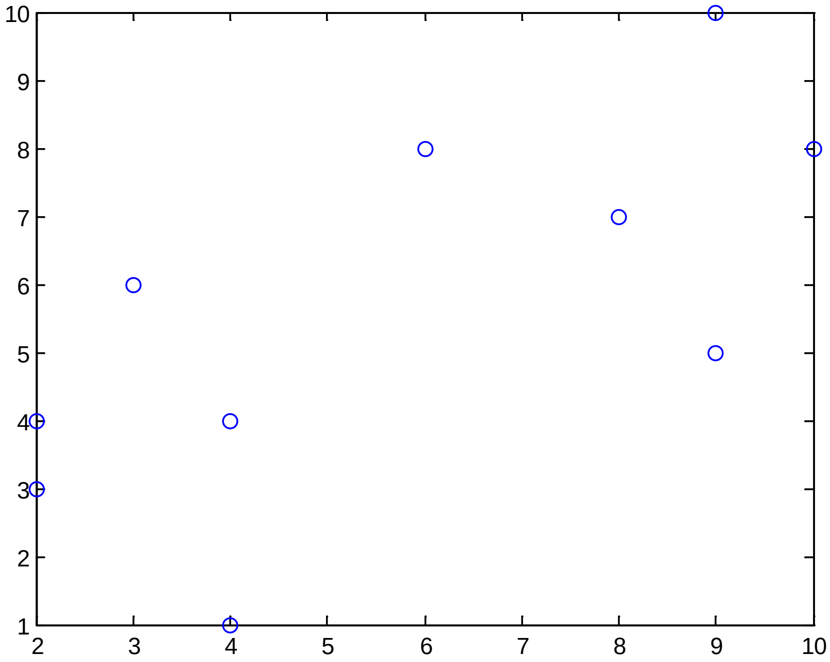
Eigenvalues = [0.3149, 1.6851]

Eigenvectors

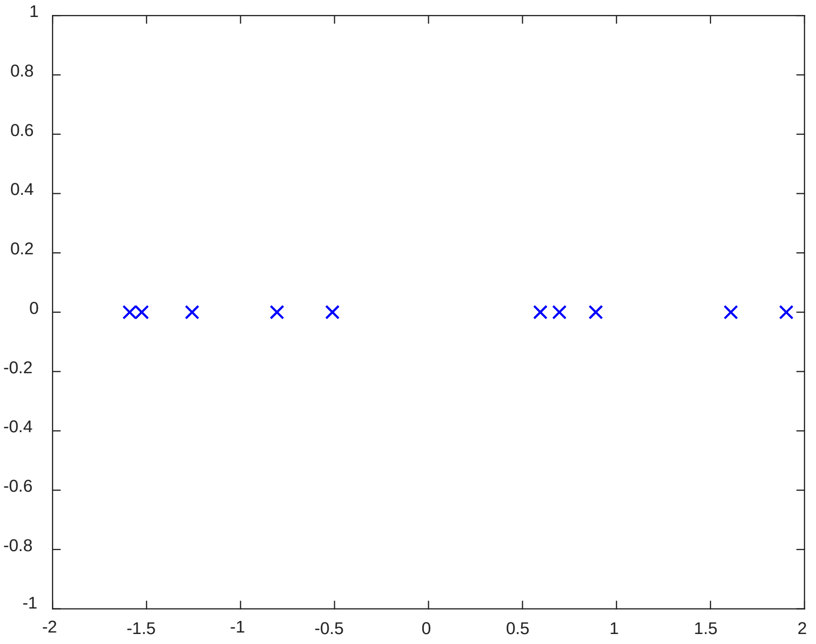
- [-0.7071, 0.7071]^T
- [0.7071, 0.7071]^T => 45 degrees

4. Finally let's project the points onto the single best vector (i.e the one with the highest eigenvalue)
   - $W = [0.7071, 0.7071]^T$
   - $Z_{1,1} = X_1 W$= [-0.55,-1.69][0.7071,0.7071]^T = -1.5862
   - Etc…

Results

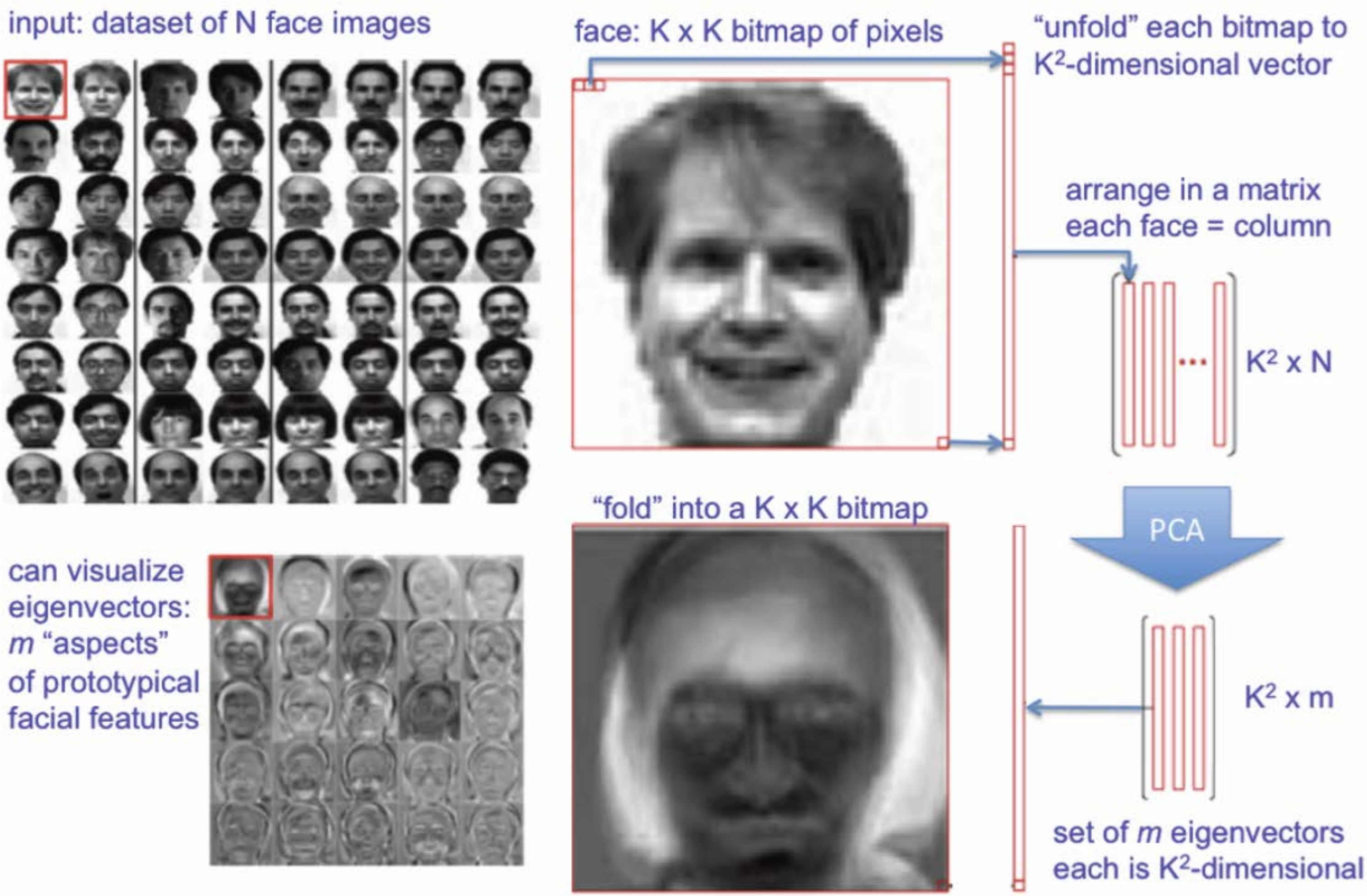


10
9
8
7
6
5
4
3
2
1
2
3
4
5
6
7
8
9
10



1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

PCA example: Eigen Faces



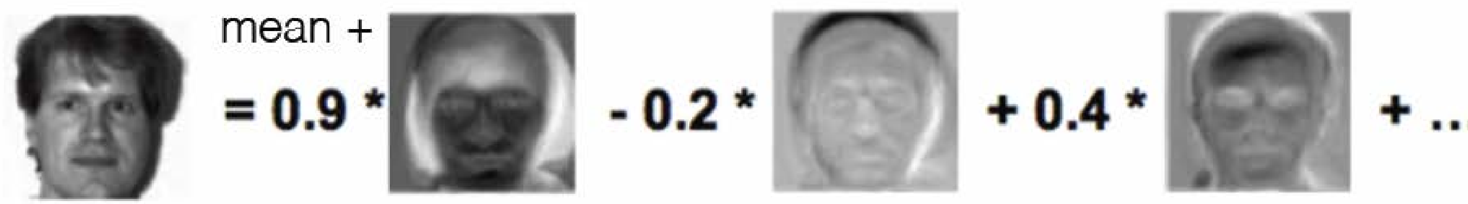
Eigen Faces: Projection

- We can taken our chosen k eigenvectors and form our $N^2 \times k$ projection matrix, W
- Then we can project each image onto this space
  - Let W be the projection matrix, X_i be the original $N^2 \times 1$ face, and $\bar{x}$ be the $N^2 \times 1$ average face.
  - Then we can represent this face in lower dimensional space as

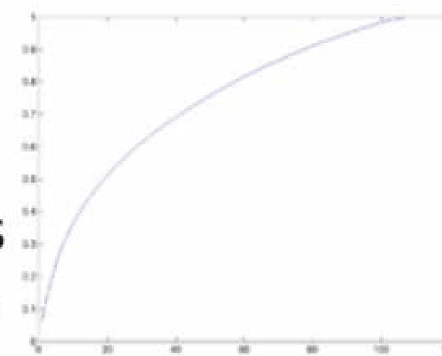
$$Z_i = (X_i - \bar{x})W$$

- Now we can use this low-dimensional feature for things like classification.

Eigen Faces: Projection



- Project new face to space of eigen-faces
- Represent vector as a linear combination of principal components
- How many do we need?

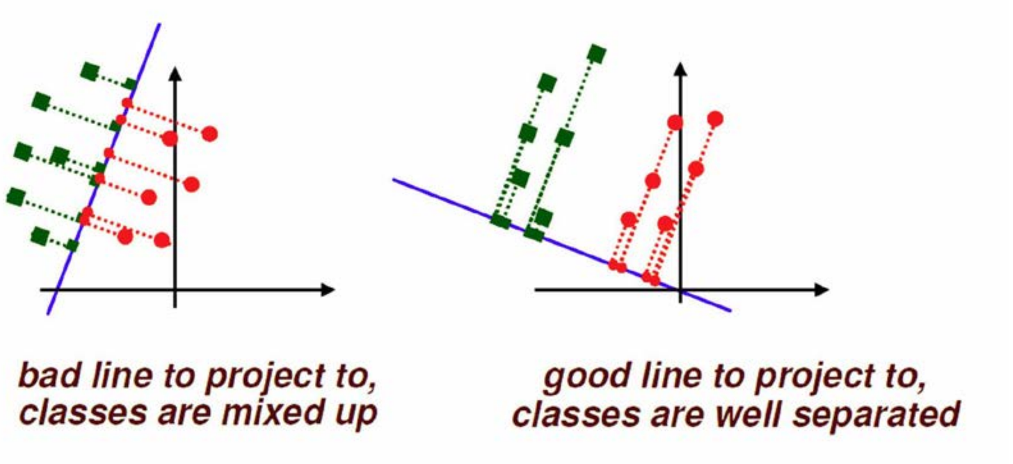


PCA Issues

- PCA finds directions to project the data so that variance is maximized
- PCA **does not consider class labels**
- Variance maximization is not necessarily beneficial for classification
  - Where we want to maximize class separation

Feature Projection by LDA (6.8)

- Linear Discriminant Analysis (LDA) projects to a line which preserves direction useful for data classification
  - **Limitation**: If we have C classes, then LDA returns a $(C-1)$ dimensional feature
    - So for binary classification, we'll just get one feature per observation.
- Main idea: find projection to a line such that samples from different classes are well separated



Feature Projection by LDA

- Suppose we have 2 classes and D-dimensional data
  - Let C_1 be the set of samples from class 1
  - Let C_2 are the samples from class 2
- We want to find some projection matrix W
- Let $\mu_1$and μ_2 be the mean of classes 1 and 2, respectively, **before** projection.
- Let σ_1 and σ_2 the standard deviation of classes 1 and 2, respectively, **before** projection.

Feature Projection by LDA

- Idea: Find a projection that maximizes the difference in the means **after** projection:
$$J(W) = argmax_W(|\mu_1 W - \mu_2 W|)$$
- Issue:
  - $|\mu_1 - \mu_2|$ is non differentiable!
  - So let's maximize $(\mu_1 W - \mu_2 W)^2$
$$J(W) = (\mu_1 W - \mu_2 W)^T(\mu_1 W - \mu_2 W)$$
- Issue:
  - This doesn't take variance of the classes into account ☹
  - We also would like to minimize the sum of the variances:
$$J(W) = \frac{(\mu_1 W - \mu_2 W)^2}{(\sigma_1 W)^2 + (\sigma_2 W)^2}$$
$$J(W) = \frac{(\mu_1 W - \mu_2 W)^T(\mu_1 W - \mu_2 W)}{(\sigma_1 W)^T(\sigma_1 W) + (\sigma_2 W)^T(\sigma_2 W)}$$

Feature Projection by LDA

- Ok so we want to maximize the function

$$J(W) = \frac{(\mu_1 W - \mu_2 W)^T(\mu_1 W - \mu_2 W)}{(\sigma_1 W)^T(\sigma_1 W) + (\sigma_2 W)^T(\sigma_2 W)}$$

  - This is called the *Fisher Discriminant Function*
- How can we find the value of W to maximize this?
  - Calculus?
- But this is a bit ugly (and pain when taking the derivative), so let’s do some substitutions to clean it up!

Feature Projection by LDA

$$J(W) = \frac{(\mu_1 W - \mu_2 W)^T (\mu_1 W - \mu_2 W)}{(\sigma_1 W)^T (\sigma_1 W) + (\sigma_2 W)^T (\sigma_2 W)}$$

- If we applied transposes and distribution we'd arrive at:

$$J(W) = \frac{W^T (\mu_1 - \mu_2)^T (\mu_1 - \mu_2) W}{W^T (\sigma_1^T \sigma_1 + \sigma_2^T \sigma_2) W}$$

- $\sigma_1^2 = \sigma_1^T \sigma_1$ is called the **scatter matrix** for class 1 is defined as

$$\sigma_1^2 = \sum_{x_i \in C_1} (x_i - \mu_1)^T (x_i - \mu_1) = (|C_1| - 1) cov(C_1)$$

- The **within class scatter matrix** is defined as

$$S_W = \sigma_1^2 + \sigma_2^2$$

- The **between class scatter matrix** is defined as

$$S_B = (\mu_1 - \mu_2)^T (\mu_1 - \mu_2)$$

- We can now write $J(W)$ as

$$J(W) = \frac{W^T S_B W}{W^T S_W W}$$

Feature Projection by LDA

$$J(W) = \frac{W^T S_B W}{W^T S_W W}$$

- Taking the derivative with respect to W and setting it equal to zero we get:

$$S_B W - \frac{W^T S_B W S_w W}{W^T S_W W} = 0$$

- Now we need to solve for W ☹

Feature Extraction by LDA

$$S_B W - \frac{W^T S_B W S_w W}{W^T S_W W} = 0$$

- Let $\lambda = \frac{W^T S_B W}{W^T S_W W}$, which will in fact be a scalar.
- Then we have

$$S_B W = \lambda S_W W$$

- We can re-write this as

$$S_W^{-1} S_B W = \lambda W$$

- Which is our normal/standard eigenvalue problem!
  - $Ax = \lambda x$

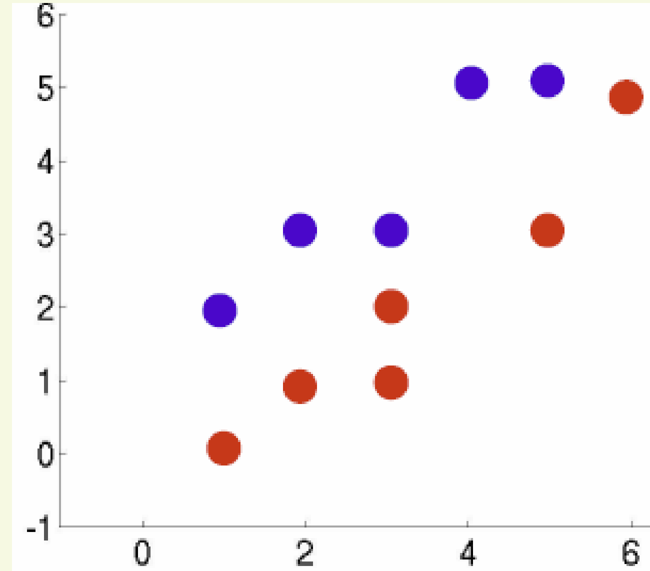
Feature Extraction by LDA

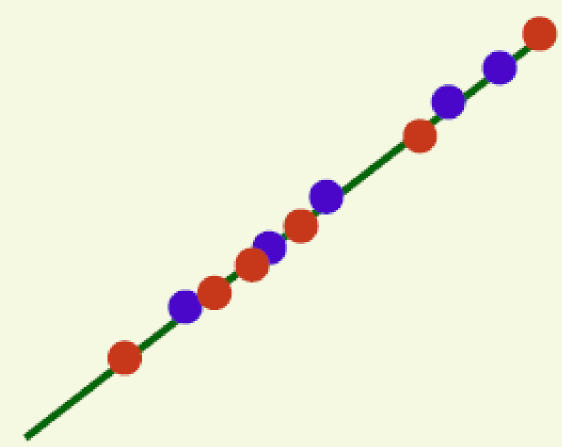
$$S_W^{-1} S_B W = \lambda W$$

- $S_W^{-1} S_B$ has only one non-zero eigenvalue (and therefore one eigenvector)
- Use that eigenvector as your projection matrix

Example: LDA

- Data:
  - Class 1 has samples $C_1=\{(1,2),(2,3),(3,3),(4,5),(5,5)\}$
  - Class 2 has samples $C_2=\{(1,0),(2,1),(3,1),(3,2),(5,3),(6,5)\}$
- If we did PCA and projected the points onto the "best line" we would get poor separation





Example: LDA

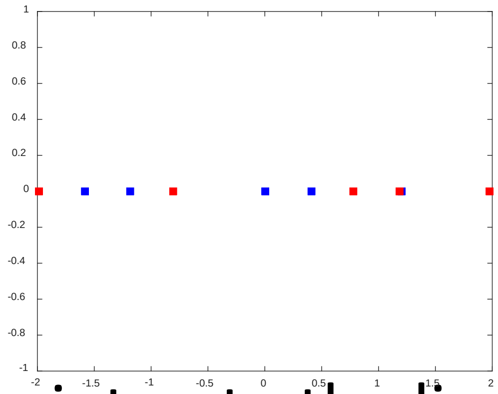
1. First standardize all the data
2. Next compute the means for each class
   - $\mu_1 = [-0.1094, 0.5023]$, $\mu_2 = [0.0911, -0.4186]$
3. Then compute the scatter matrices for each class
   - $\sigma_1^2 = (5-1) * cov(C_1) = \begin{bmatrix} 3.62 & 2.77 \\ 2.77 & 2.39 \end{bmatrix}$
   - $\sigma_2^2 = (6-1) * cov(C_2) = \begin{bmatrix} 6.27 & 5.54 \\ 5.54 & 5.30 \end{bmatrix}$

Example: LDA

4. Followed by the within class scatter matrix
   - $S_W = \sigma_1^2 + \sigma_2^2 = \begin{bmatrix} 9.89 & 8.31 \\ 8.31 & 7.69 \end{bmatrix}$
   - $S_W^{-1} = \begin{bmatrix} 1.10 & -1.19 \\ -1.19 & 1.42 \end{bmatrix}$ Remember how to do this?
     - Matlab also has `inv(sw)`
5. Perform eigen-decomposition on $S_w^{-1} S_B$
6. The eigenvector pertaining to the only non-zero eigenvalue is our projection matrix:
7. There is only one non-zero eigen-value so its corresponding eigen-vector becomes our direction of projection:

$$W = [-0.6484, 0.7613]^T$$

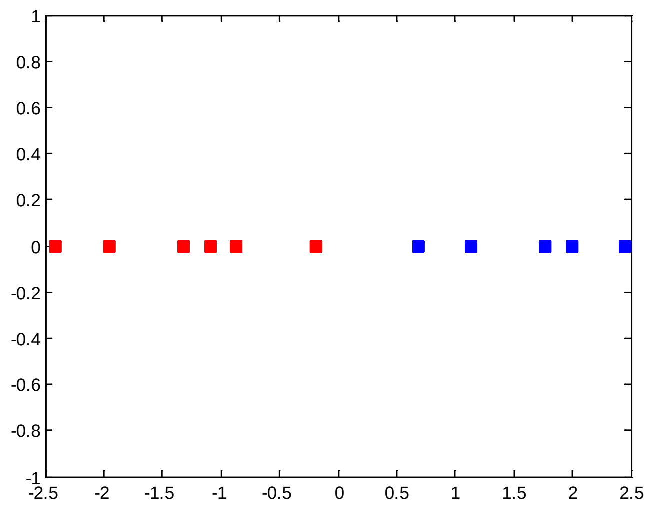
Example LDA

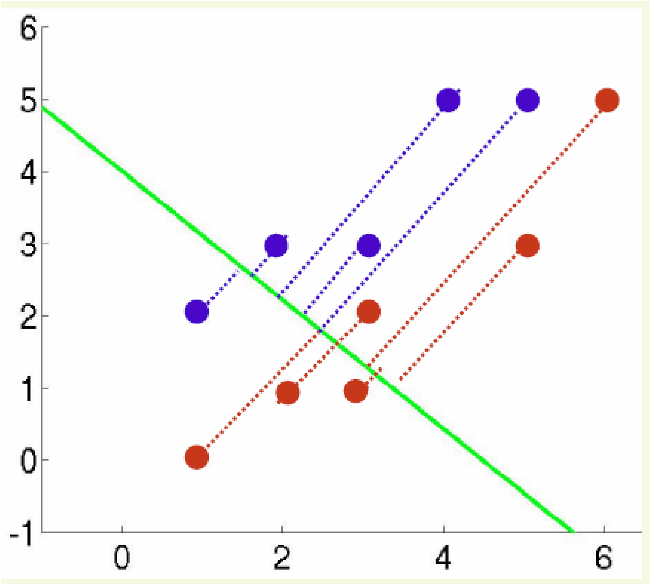


- Now we just project each point onto the line(s)

$$Z = XW$$

- Binary Classification reduces feature space to 1-D
  - Is that good enough?





Multi-Class LDA

- To do multi-class LDA ($C > 2$) we just need to make a few changes:
  - $S_W = \sum_{i=1}^{C}(|C_i| - 1)cov(C_i)$
  - $S_B = \sum_{i=1}^{C}|C_i|(\mu_i - \mu)^T(\mu_i - \mu)$
    - Where μ is the mean of **al**l data
  - Solve the eigenvalue problem and choose which eigenvectors to use based on the largest eigenvalues
    - There will be at most $C - 1$ non-zero ones.