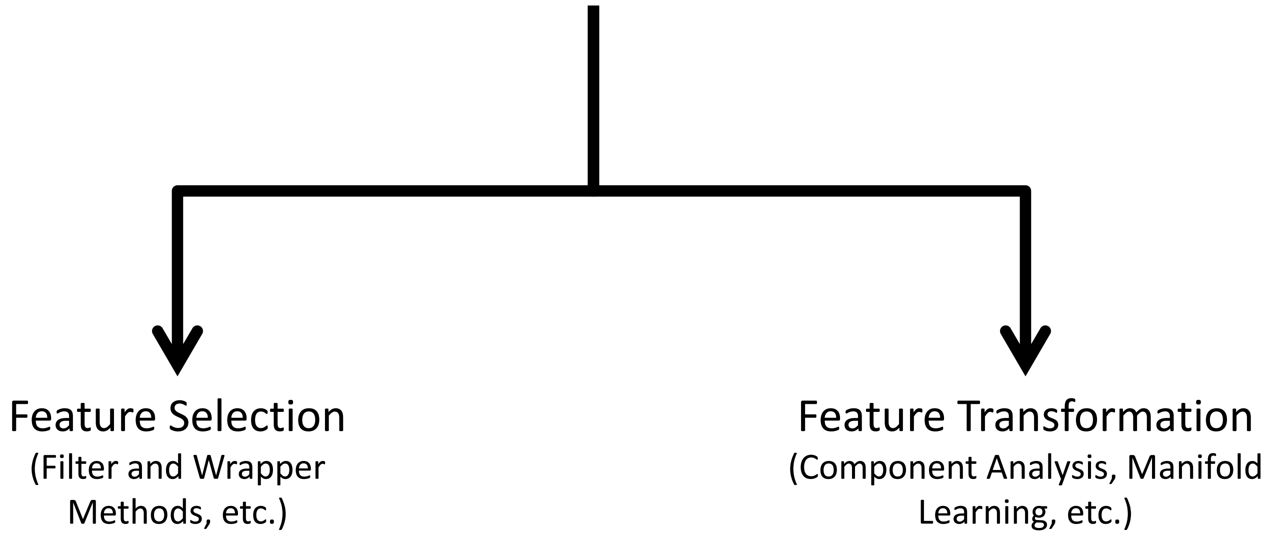
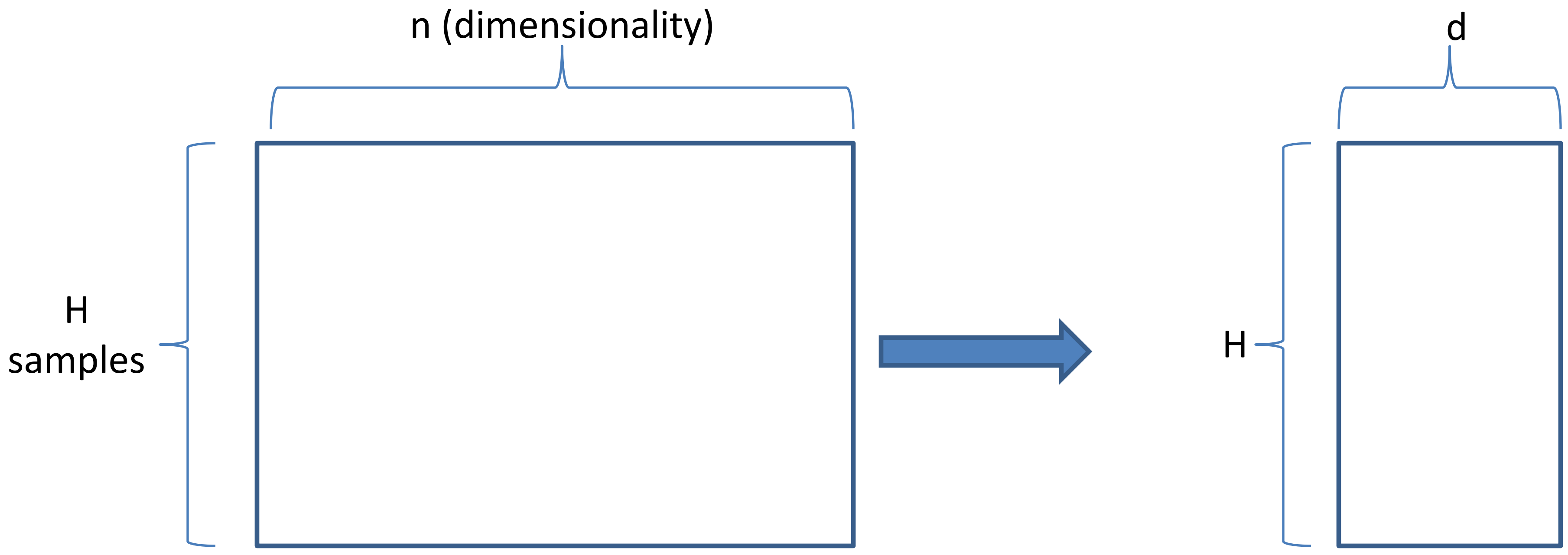
Dimensionality Reduction

Dimensionality Reduction


Feature Selection
(Filter and Wrapper Methods, etc.)
Feature Transformation
(Component Analysis, Manifold Learning, etc.)

Dimensionality Reduction

Summarization of data with many (n) variables by a smaller set of (d) derived (synthetic, composite) variables.



d << n

Feature Selection

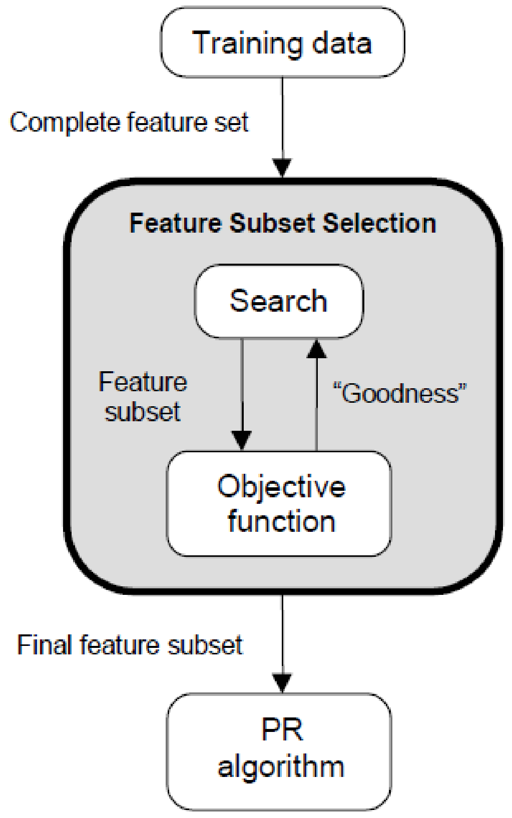
- Given a set of **n** features, the goal of **feature selection** is to select a subset of **d** features (**d** < **n**) in order to minimize the classification error.

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \\ x_N \end{bmatrix} \xrightarrow{\text{feature selection}} \begin{bmatrix} x_{i_1} \\ x_{i_2} \\ \\ x_{i_M} \end{bmatrix} \qquad \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \\ x_N \end{bmatrix} \xrightarrow{\text{dimensionality reduction}} \begin{bmatrix} y_1 \\ y_2 \\ \\ y_M \end{bmatrix} = f\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \\ x_N \end{bmatrix}\right)$$

- Fundamentally different from feature transformation(e.g., PCA or LDA) based on feature combinations (i.e., **feature extraction**).

Feature Selection Steps

- Feature selection is an **optimization** problem.
  - Step 1: Search the space of possible feature subsets.
  - Step 2: Pick the subset that is optimal or near-optimal with respect to some objective function.



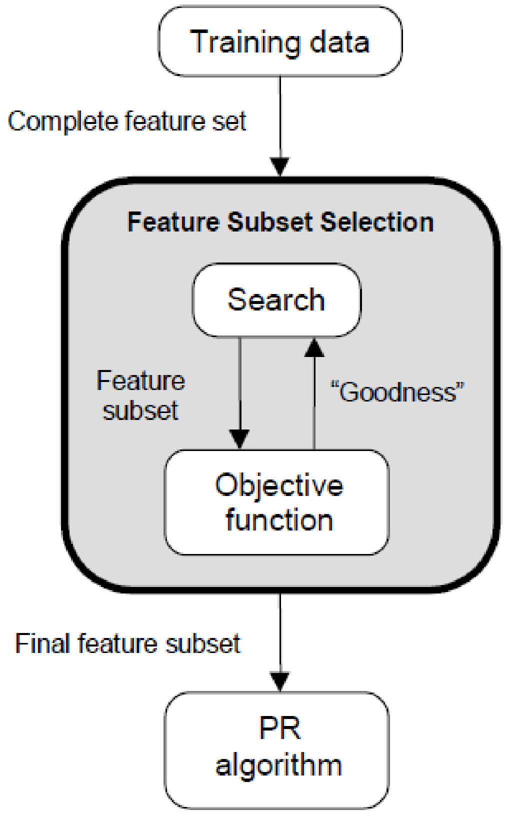
Feature Selection Steps (cont'd)

Search strategies

- Optimum
- Heuristic
- Randomized

Evaluation strategies

- Filter methods
- Wrapper methods

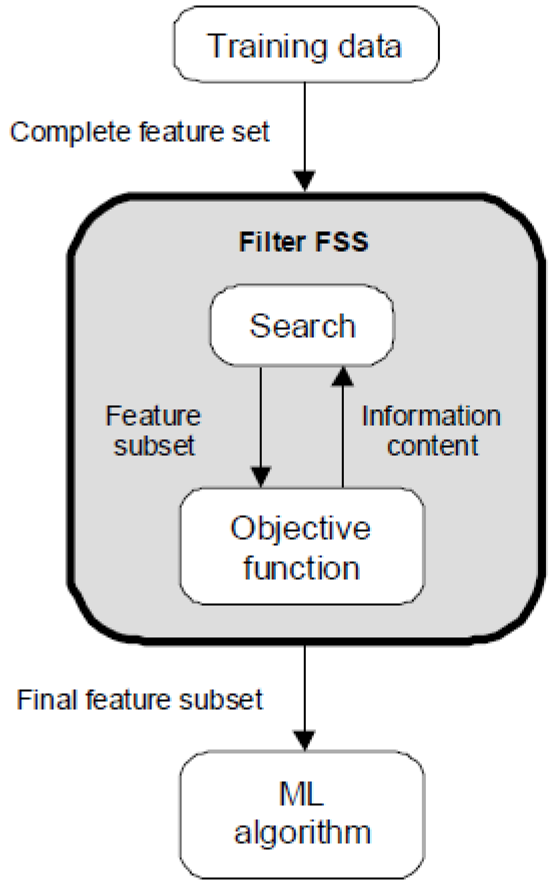


Search Strategies

- Assuming n features, an exhaustive search would require:
  - Examining all $\binom{n}{d}$ possible subsets of size d.
  - Selecting the subset that performs the best according to the criterion function.
- The number of subsets grows combinatorially, making exhaustive search impractical.
- In practice, heuristics are used to speed-up search but they **cannot** guarantee optimality.

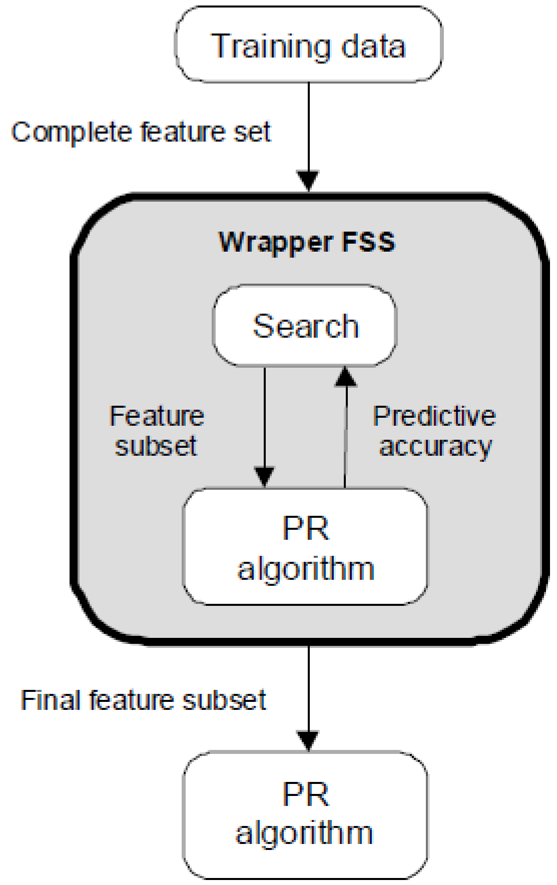
Evaluation Strategies

- Filter Methods
  - Evaluation is **independent** of the classification algorithm.
  - The objective function evaluates feature subsets by their information content, typically interclass distance, statistical dependence or information-theoretic measures.



Evaluation Strategies

- Wrapper Methods
  - Evaluation uses criteria **related** to the classification algorithm.
  - The objective function is a pattern classifier, which evaluates feature subsets by their predictive accuracy (recognition rate on test data) by statistical resampling or cross-validation.



Filter vs Wrapper Approaches

- **Wrappers**
  - Advantages
    - **Accuracy**: wrappers generally achieve better recognition rates than filters since they are tuned to the specific interactions between the classifier and the dataset
    - **Ability to generalize**: wrappers have a mechanism to avoid overfitting, since they typically use cross-validation measures of predictive accuracy
  - Disadvantages
    - **Slow execution**: since the wrapper must train a classifier for each feature subset (or several classifiers if cross-validation is used), the method can become unfeasible for computationally intensive methods
    - **Lack of generality**: the solution lacks generality since it is tied to the bias of the classifier used in the evaluation function. The "optimal" feature subset will be specific to the classifier under consideration

Filter vs Wrapper Approaches (cont’d)

- Filters
  - Advantages
    - **Fast execution**: Filters generally involve a non-iterative computation on the dataset, which can execute much faster than a classifier training session
    - **Generality**: Since filters evaluate the intrinsic properties of the data, rather than their interactions with a particular classifier, their results exhibit more generality: the solution will be “good” for a larger family of classifiers
  - Disadvantages
    - **Tendency to select large subsets**: Since the filter objective functions are generally monotonic, the filter tends to select the full feature set as the optimal solution. This forces the user to select an arbitrary cutoff on the number of features to be selected

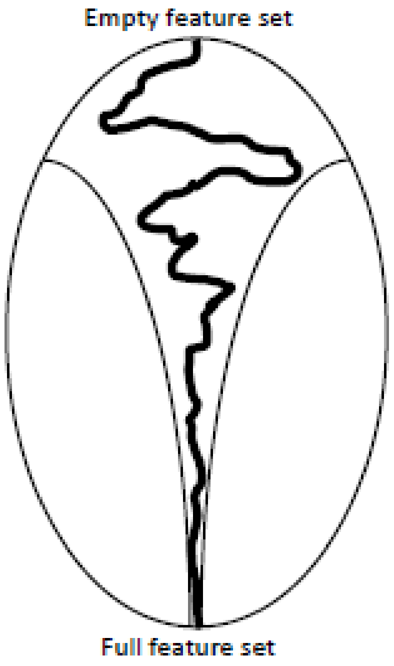
Searching for the Best Feature

- Sort the given n features in order of their probability of correct recognition.
- Select the top d features from this sorted list.
- Criterion
  - Correlation among features.
  - Correlation with the class.

Sequential forward selection (SFS)

(heuristic search)

- First, the best **single** feature is selected (i.e., using some criterion function).
- Then, **pairs** of features are formed using one of the remaining features and this best feature, and the best pair is selected.
- Next, **triplets** of features are formed using one of the remaining features and these two best features, and the best triplet is selected.
- This procedure continues until a predefined number of features are selected.

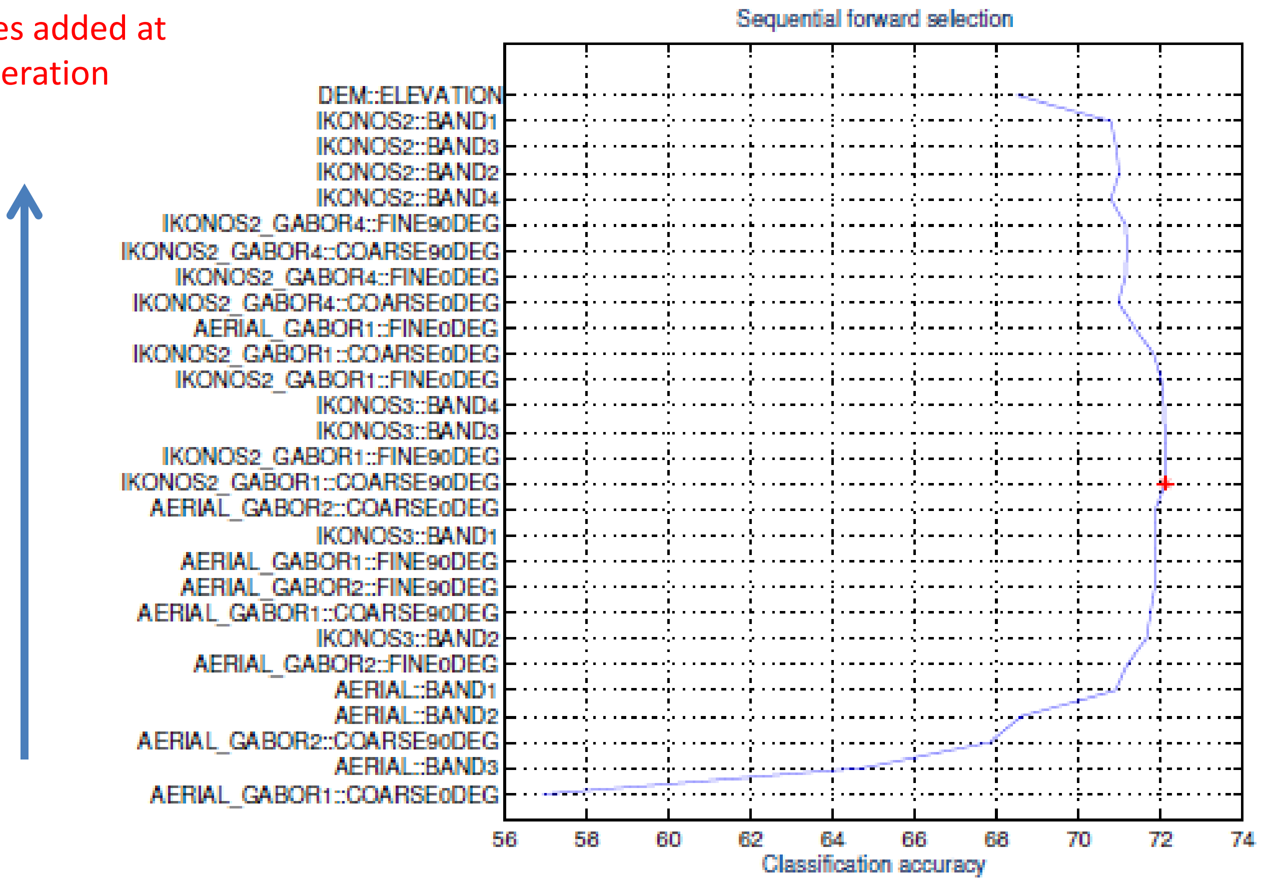


```
1. Start with the empty set $Y_0 = \{\emptyset\}$
2. Select the next best feature $x^+ = \arg\max_{x \notin Y_k} J(Y_k + x)$
3. Update $Y_{k+1} = Y_k + x^+;\ k = k + 1$
4. Go to 2
```

SFS performs best when the optimal subset is small.

Example

features added at each iteration

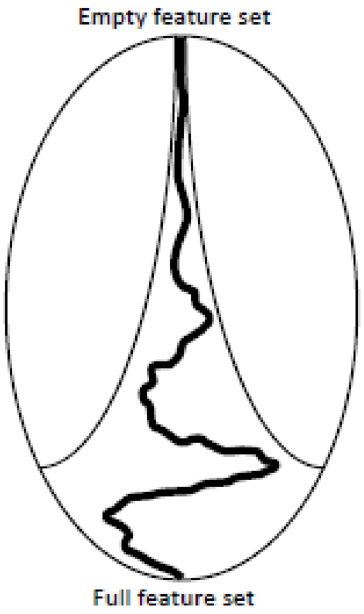


Results of sequential forward feature selection for classification of a satellite image using 28 features. x-axis shows the classification accuracy (%) and y-axis shows the features added at each iteration (the first iteration is at the bottom). The highest accuracy value is shown with a star.

Sequential backward selection (SBS)

(heuristic search)

- First, the criterion function is computed for all **n** features.
- Then, each feature is deleted one at a time, the criterion function is computed for all subsets with **n-1** features, and the worst feature is discarded.
- Next, each feature among the remaining **n-1** is deleted one at a time, and the worst feature is discarded to form a subset with **n-2** features.
- This procedure continues until a predefined number of features are left.

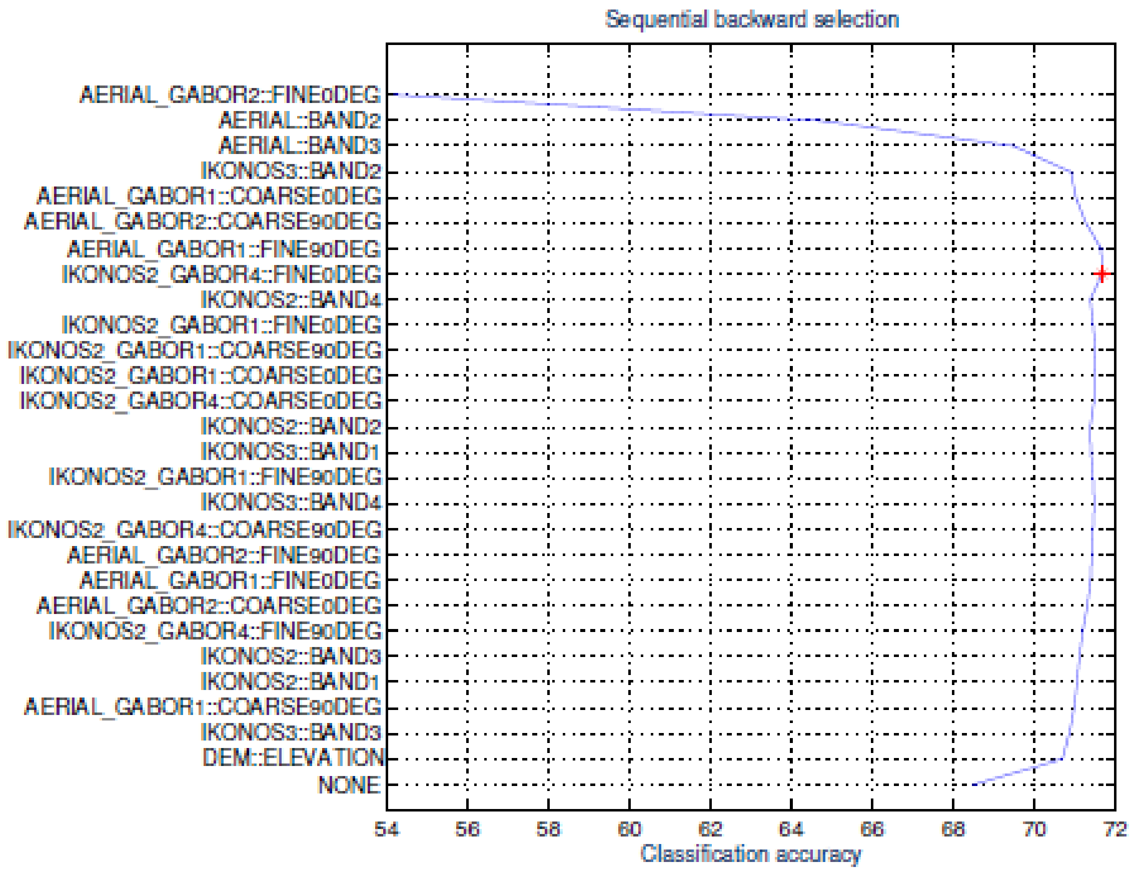


1. Start with the full set $Y_0 = X$
2. Remove the worst feature $x^- = \underset{x \in Y_k}{\arg\max}\, J(Y_k - x)$
3. Update $Y_{k+1} = Y_k - x^-;\ k = k + 1$
4. Go to 2

SBS performs best when the optimal subset is large.

Example

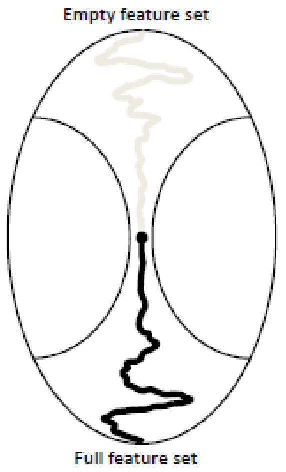
features removed at each iteration

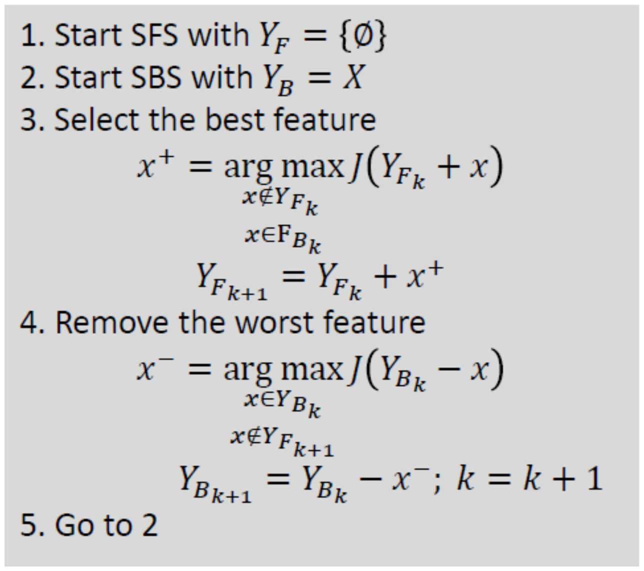


Results of sequential backward feature selection for classification of a satellite image using 28 features. x-axis shows the classification accuracy (%) and y-axis shows the features removed at each iteration (the first iteration is at the top). The highest accuracy value is shown with a star.

Bidirectional Search (BDS)

- BDS applies SFS and SBS simultaneously:
  - SFS is performed from the empty set.
  - SBS is performed from the full set.
- To guarantee that SFS and SBS converge to the same solution:
  - Features already selected by SFS are not removed by SBS.
  - Features already removed by SBS are not added by SFS.





```
1. Start SFS with $Y_F = \{\emptyset\}$
2. Start SBS with $Y_B = X$
3. Select the best feature
       $x^+ = \underset{\substack{x \notin Y_{F_k} \\ x \in F_{B_k}}}{\arg\max} J(Y_{F_k} + x)$
       $Y_{F_{k+1}} = Y_{F_k} + x^+$
4. Remove the worst feature
       $x^- = \underset{\substack{x \in Y_{B_k} \\ x \notin Y_{F_{k+1}}}}{\arg\max} J(Y_{B_k} - x)$
       $Y_{B_{k+1}} = Y_{B_k} - x^-;\ k = k + 1$
5. Go to 2
```

Limitations of SFS and SBS

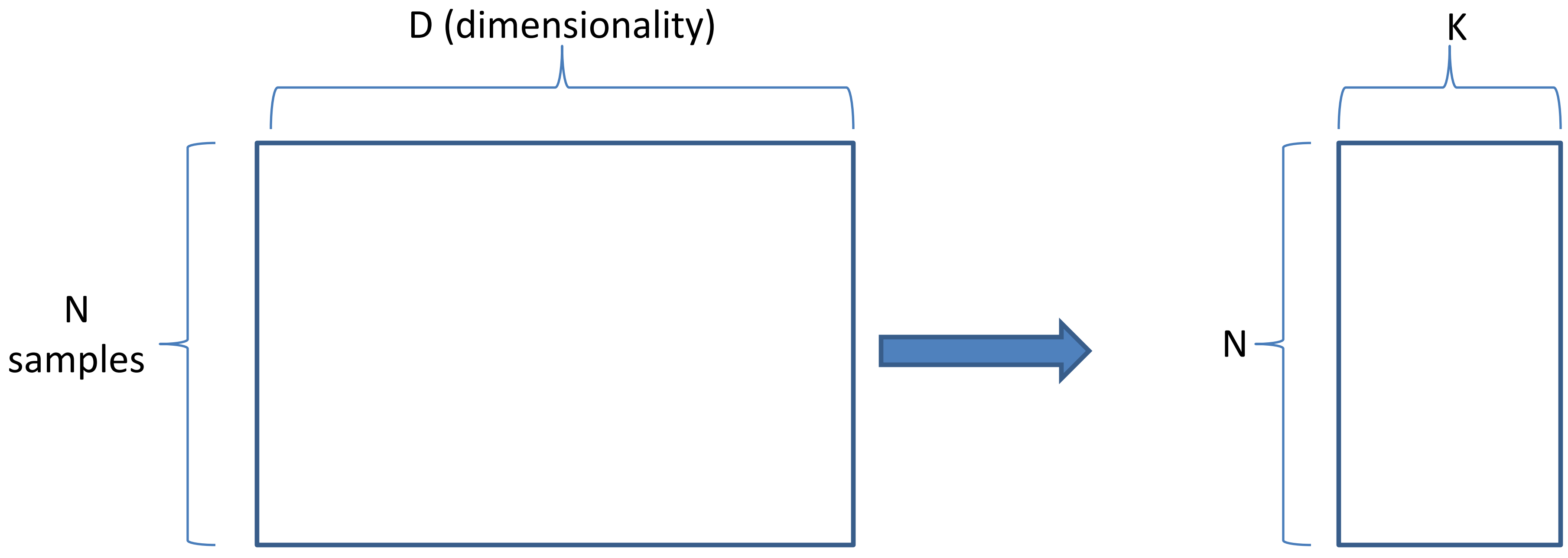
- The main limitation of SFS is that it is unable to remove features that become non useful after the addition of other features.

- The main limitation of SBS is its inability to reevaluate the usefulness of a feature after it has been discarded.

Dimensionality Reduction by Component Analysis Techniques

Dimensionality Reduction

Summarization of data with many (p) variables by a smaller set of (k) derived (synthetic, composite) variables.



K << D

Principal Component Analysis

(on the board)

Steps for PCA

- Compute the mean of the data

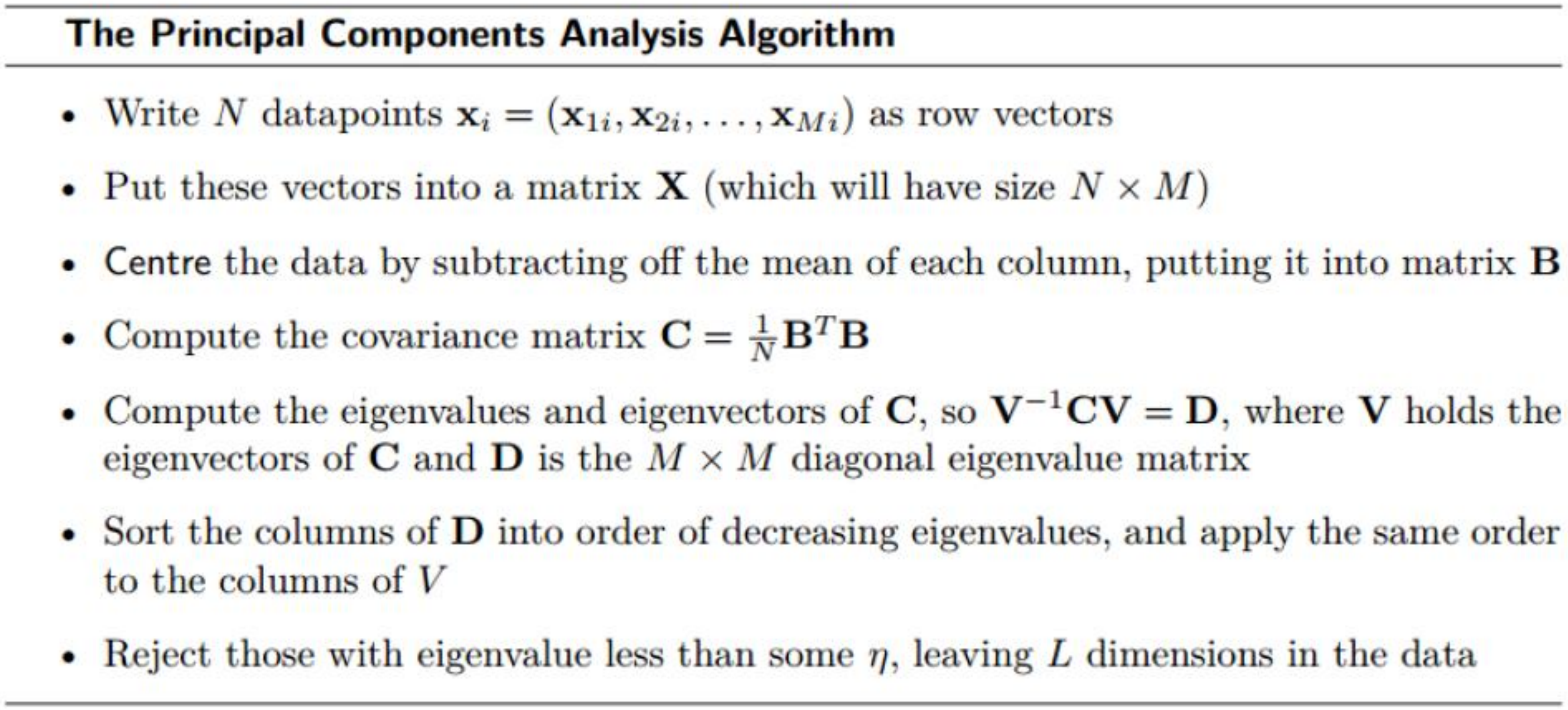
$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

- Compute the sample covariance matrix (using the mean subtracted data)

$$\boldsymbol{S} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})(x_n - \bar{x})^\top$$

- Do the eigenvalue decomposition of the $D \times D$ matrix $\boldsymbol{S}$
- Take the top K eigenvectors (corresponding to the top K eigenvalues)
- Call these $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K$ (s.t. $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_{K-1} \geq \lambda_K$)
- $\mathbf{W} = [\boldsymbol{w}_1 \ \boldsymbol{w}_2 \ \ldots \ \boldsymbol{w}_K]$ is the projection matrix of size $D \times K$
- Projection of each example $\boldsymbol{x}_n$ is computed as $\boldsymbol{z}_n = \mathbf{W}^\top \boldsymbol{x}_n$
  - $\boldsymbol{z}_n$ is a $K \times 1$ vector (also called the embedding of $\boldsymbol{x}_n$)

The PCA Algorithm



The Principal Components Analysis Algorithm

- Write N datapoints $\mathbf{x}_i = (\mathbf{x}_{1i}, \mathbf{x}_{2i}, \ldots, \mathbf{x}_{Mi})$ as row vectors
- Put these vectors into a matrix $\mathbf{X}$ (which will have size $N \times M$)
- Centre the data by subtracting off the mean of each column, putting it into matrix $\mathbf{B}$
- Compute the covariance matrix $\mathbf{C} = \frac{1}{N}\mathbf{B}^T\mathbf{B}$
- Compute the eigenvalues and eigenvectors of $\mathbf{C}$, so $\mathbf{V}^{-1}\mathbf{C}\mathbf{V} = \mathbf{D}$, where $\mathbf{V}$ holds the eigenvectors of $\mathbf{C}$ and $\mathbf{D}$ is the $M \times M$ diagonal eigenvalue matrix
- Sort the columns of $\mathbf{D}$ into order of decreasing eigenvalues, and apply the same order to the columns of V
- Reject those with eigenvalue less than some η, leaving L dimensions in the data

Geometric Interpretation

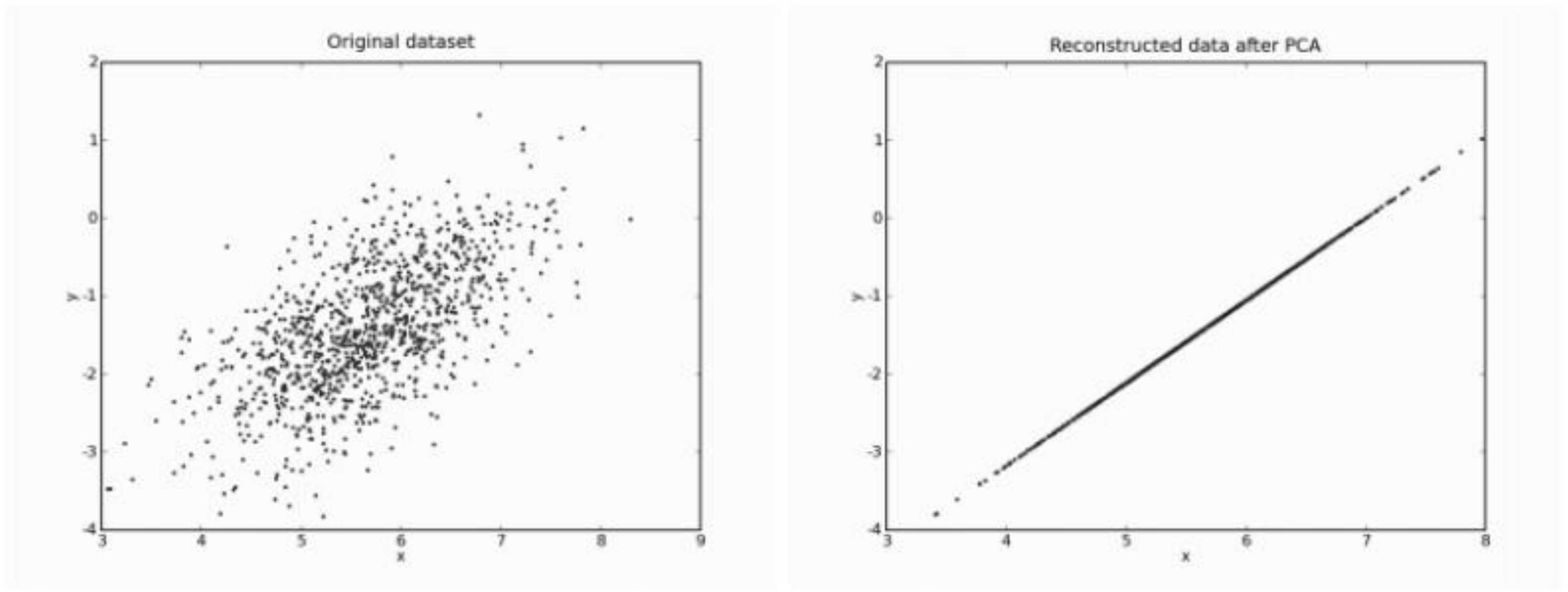
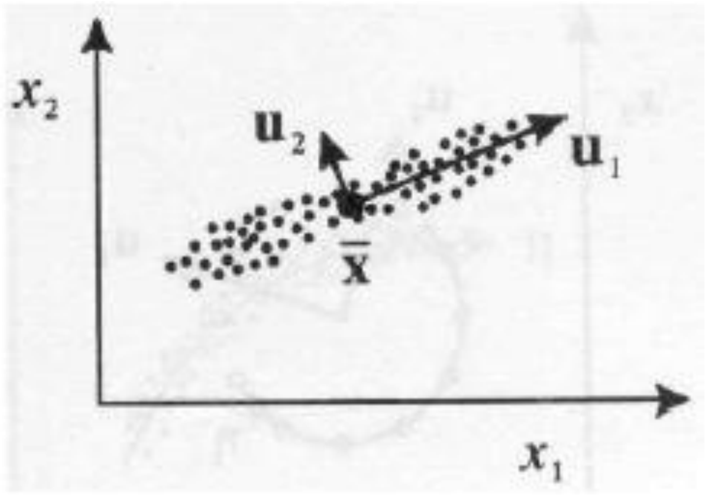


FIGURE 6.7 Computing the principal components of the 2D dataset on the left and using only the first one to reconstruct it produces the line of data shown on the right, which is along the principal axis of the ellipse that the data was sampled from.

Geometric interpretation

- PCA projects the data along the directions where the data varies most.
- These directions are determined by the eigenvectors of the covariance matrix corresponding to the largest eigenvalues.
- The magnitude of the eigenvalues corresponds to the variance of the data along the eigenvector directions.



How to choose K?

- Choose *K* using the following criterion:

$$\frac{\sum_{i=1}^{K} \lambda_i}{\sum_{i=1}^{N} \lambda_i} > Threshold \quad \text{(e.g., 0.9 or 0.95)}$$

- In this case, we say that we “preserve” 90% or 95% of the information (variance) in the data. [Eigenenergy]

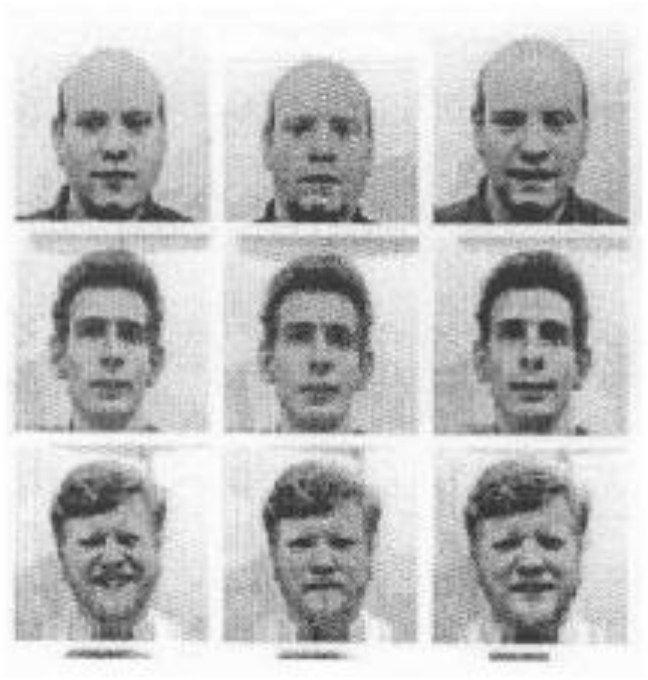
- If K=N, then we “preserve” 100% of the information in the data.

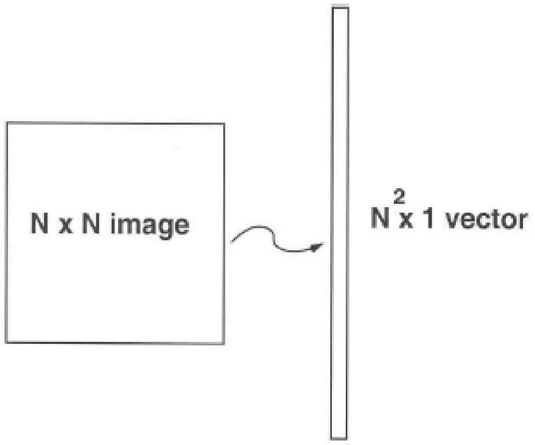
Application to Faces

- Computation of low-dimensional basis (i.e.,eigenfaces):

<u>Step 1:</u> obtain face images $I_1, I_2, ..., I_M$ (training faces)

(**very important:** the face images must be *centered* and of the same *size*)





<u>Step 2:</u> represent every image I_i as a vector Γ_i

Application to Faces (cont'd)

- Computation of the eigenfaces – cont.

Step 3: compute the average face vector Ψ:

$$\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i$$

Step 4: subtract the mean face:

$$\Phi_i = \Gamma_i - \Psi$$

Step 5: compute the covariance matrix C:

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T = \frac{1}{M}AA^T \quad (N^2 \text{x} N^2 \text{ matrix})$$

$$\text{where } A = [\Phi_1\ \Phi_2 \cdots \Phi_M] \quad (N^2 \text{x} M \text{ matrix})$$

Application to Faces (cont'd)

- Computation of the eigenfaces – cont.

Note 1: AA^T can have up to N^2 eigenvalues and eigenvectors.

Note 2: $A^T A$ can have up to M eigenvalues and eigenvectors.

Note 3: The M eigenvalues of $A^T A$ (along with their corresponding eigenvectors) correspond to the M *largest* eigenvalues of AA^T (along with their corresponding eigenvectors).

Step 6.3: compute the M best eigenvectors of AA^T: $u_i = Av_i$ (i.e., using A^TA)

(**important:** normalize u_i such that $\|u_i\| = 1$)

Step 7: keep only K eigenvectors (corresponding to the K largest eigenvalues)

each face Φ_i can be represented as follows:

$$\hat{\Phi}_i - mean = \sum_{j=1}^{K} w_j u_j, \quad (w_j = u_j^T \Phi_i) \Rightarrow \Omega = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_K \end{bmatrix}$$

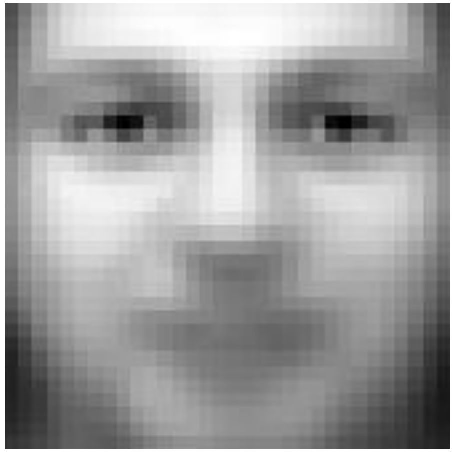
Example

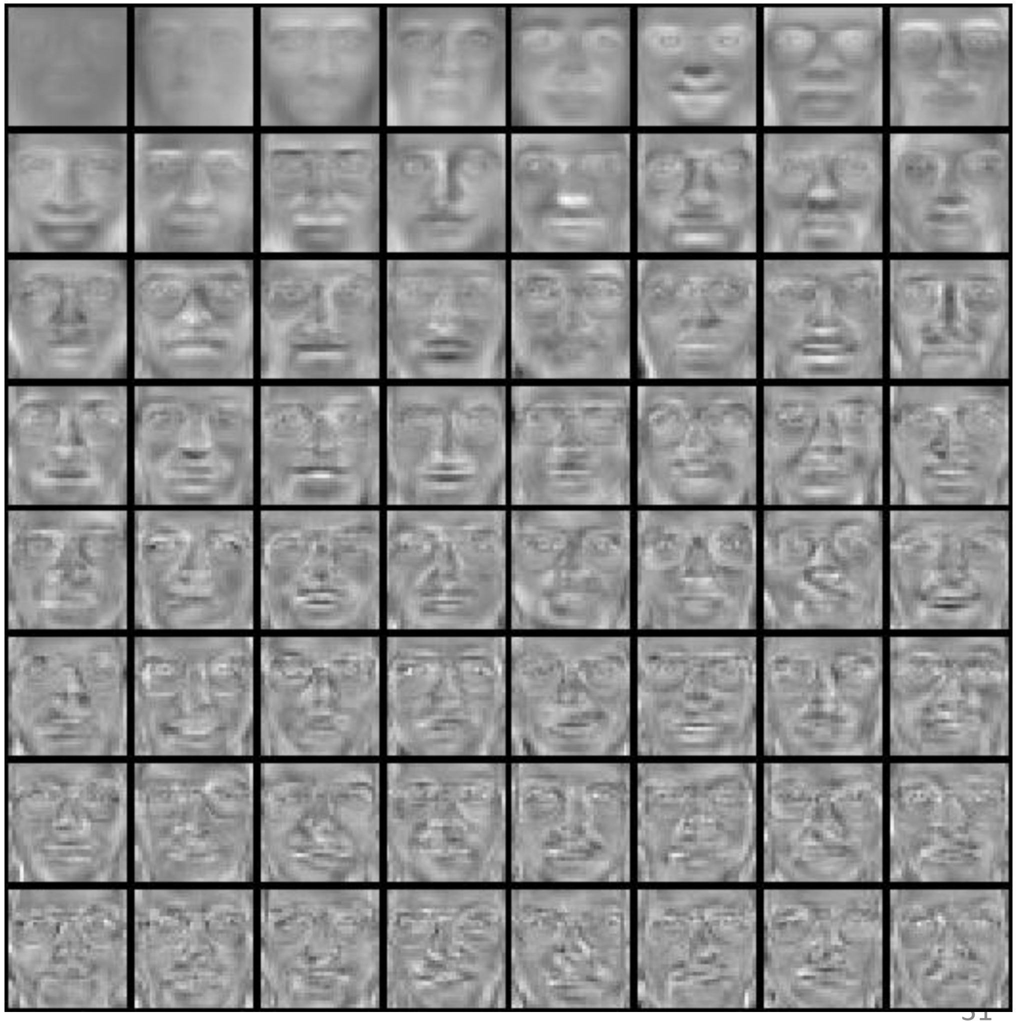
Normalized face images

Example (cont'd)

Top eigenvectors: $u_1, \ldots u_k$

Mean: μ





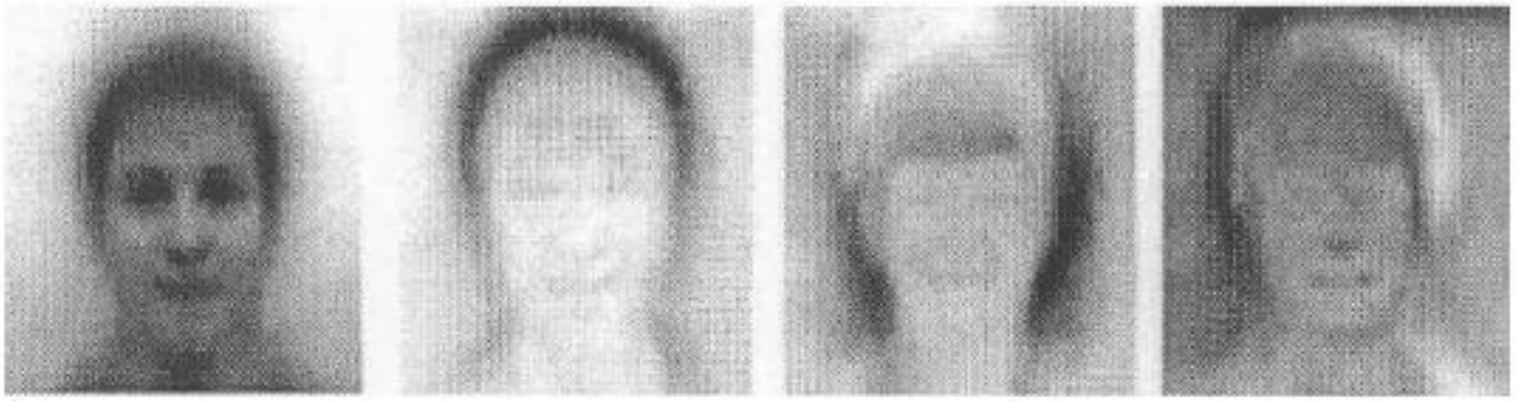
Application to Faces (cont'd)

- Representing faces onto this basis

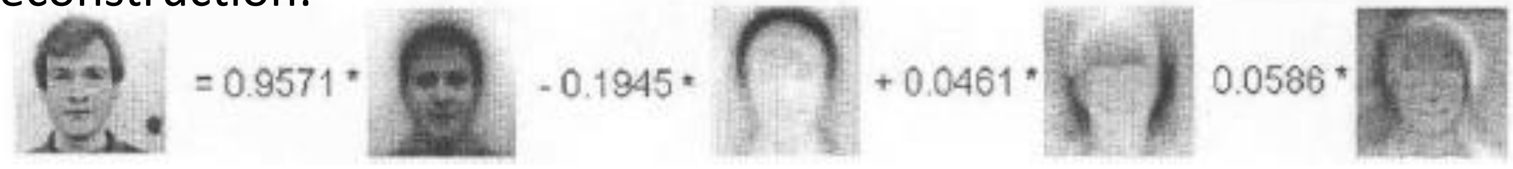
- Each face (minus the mean) Φ_i in the training set can be represented as a linear combination of the best K eigenvectors:

$$\hat{\Phi}_i - mean = \sum_{j=1}^{K} w_j u_j, \quad (w_j = u_j^T \Phi_i) \qquad (where \| u_j \| = 1)$$

(we call the u_j's *eigenfaces*)



Face reconstruction:



Questions ?