

# CS4104 Applied Machine Learning

## Dimensionality Reduction

# Dimensionality Reduction

- The input space of many learning problems is of high dimensionality
- This has computational implications and, it makes finding the intrinsic information content difficult
- Context: unsupervised learning
- given a collection of data points in an  $n$ -dimensional space a good representation of the data in  $r$ -dimensional

# Dimensionality Reduction

- m examples; n dimensional space

- $$X = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{nm} \end{bmatrix}$$

- The dimensionality reduction is to reduce dimensions n to r

- $$X = \begin{bmatrix} x_{11} & \cdots & x_{1r} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{nr} \end{bmatrix}$$

# Dimensionality Reduction Techniques

- Visualize, categorize, or simplify large datasets
- **Principal Component Analysis (PCA)**: Finds the dimensions that capture the most variance
- **Multidimensional Scaling (MDS)**: Finds data points in lower dimensional space that best preserves the inter-point distance.
- **Isomap**: Estimates the distance between two points on a manifold by following a chain of points with shorter distances between them. (More accurate in representing global distances than LLE; slower than LLE)
- **Local Linear Embedding (LLE)**: Only worries about representing the distances between local points. Faster than Isomap (no worry about global distances)
- **Decision Tree based Selection**

# Feature Selection

- In many applications, we often encounter a very large number of **potential features** that can be used
- Which **subset of features** should be used for the best classification?
- Need for a small number of discriminative features
  - To avoid “curse of dimensionality”
  - To reduce feature measurement cost
  - To reduce computational burden
- Given an  $n \times d$  pattern matrix ( $n$  patterns in  $d$ -dimensional feature space), generate an  $n \times m$  pattern matrix, where  $m \ll d$

# Feature Selection vs. Extraction

- Both are collectively known as dimensionality reduction
- **Selection**: choose a **best** subset of size  $m$  from the available  $d$  features
- **Extraction**: given  $d$  features (set  $Y$ ), **extract**  $m$  new features (set  $X$ ) by **linear or non-linear combination** of all the  $d$  features
  - Linear feature extraction:  $X = TY$ , where  $T$  is a  $m \times d$  matrix
  - Non-linear feature extraction:  $X = f(Y)$
- New features by extraction may not have physical interpretation/meaning
- Examples of linear feature extraction
  - Unsupervised: PCA; Supervised: LDA/MDA
- Criteria for selection/extraction: either improve or maintain the classification accuracy, simplify classifier complexity

# Feature Selection

- How to find the **best** subset of size  $m$ ?
- Recall, **best** means classifier based on these  $m$  features has the lowest probability of error of all such classifiers
- Simplest approach is to do an **exhaustive search**; computationally prohibitive
  - For  $d=24$  and  $m=12$ , there are about 2.7 million possible feature subsets! Cover & Van Campenhout (IEEE SMC, 1977) showed that to guarantee the best subset of size  $m$  from the available set of size  $d$ , one must examine all possible subsets of size  $m$
- Heuristics have been used to avoid exhaustive search
- How to evaluate the subsets?
  - Error rate; but then **which classifier** should be used?
  - Distance measure; Mahalanobis, divergence,...
- Feature selection is an optimization problem

# Feature Selection: Evaluation, Application, and Small Sample Performance (Jain & Zongker, IEEE Trans. PAMI, Feb 1997)

- Value of feature selection in combining features from different data models
- Potential difficulties feature selection faces in small sample size situation
- Let  $Y$  be the original set of features and  $X$  is the selected subset
- Feature selection criterion function for the set  $X$  is  $J(X)$ ; large values of  $J$  indicates better feature subset; problem is to find subset  $X$  such that  $J(X) = \max_{Z \in Y, |Z|=d} J(Z)$

$$J(X) = \max_{Z \subseteq Y, |Z|=d} J(Z)$$



# Taxonomy of Feature Selection Algorithms

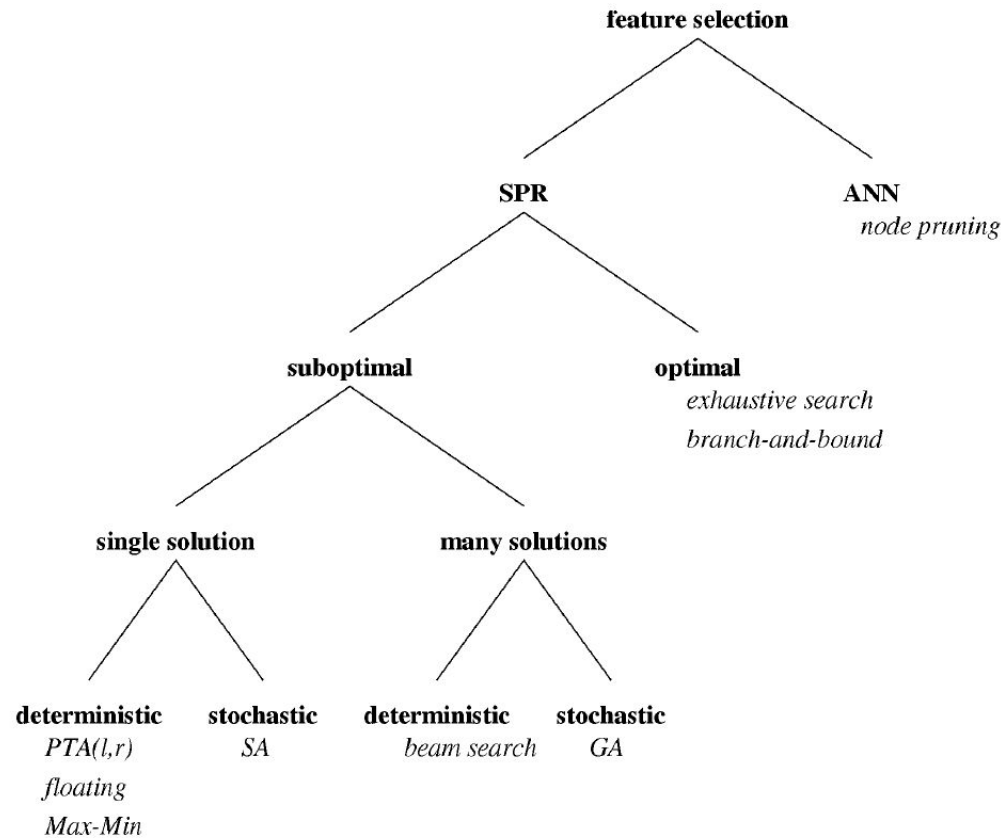


Fig. 1. A taxonomy of feature selection algorithms.

## Deterministic Single-Solution Methods

- Begin with a single solution (feature subset) & iteratively add or remove features until some termination criterion is met
- Also known as **sequential methods; most popular**
  - **Bottom up/forward methods:** begin with an empty set & add features
  - Top-down/backward methods: begin with a full set & delete features
- Since they do not examine all possible subsets, no guarantee of finding the optimal subset
- Pudil introduced two floating selection methods: **SFFS, SFBS**
- 15 feature selection methods listed in Table 1 were evaluated

# Sequential Forward Selection (SFS)

- Start with empty set,  $X=0$
- Repeatedly add most significant feature with respect to  $X$
- *Disadvantage: Once a feature is retained, it cannot be discarded; nesting problem*

# Sequential Backward Selection (SBS)

- Start with full set,  $X=Y$
- Repeatedly delete least significant feature in  $X$
- *Disadvantage: SBS requires more computation than SFS; Nesting problem*

# CS4104 Applied Machine Learning

Principal Component Analysis (PCA)

# PCA

- PCA was invented in 1901 by Karl Pearson
- statistical procedure that uses an orthogonal transformation ( $T: V \rightarrow V : \forall (u, v) \in V \text{ then } (u, v) = (Tu, Tv)$ ) to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.
- Example: marks of quiz and midterm can be correlated in a dataset and PCA will convert it into a non-correlated set of variables/components.

# PCA for Features Reduction

- Computation of Covariance Matrix (M)
- Compute the Eigen values
- Sort the Eigen values
- For first k (all non zero) Eigen values
  - Compute the Eigen vectors
- Use the Eigen Vectors as features

# Covariance Matrix

- The relationship between attributes/features
- Provided the dataset of  $n$  elements as  $Data = \{p_1, p_2, p_3, \dots, p_n\}$  where each data point is of  $d$  dimensions  $p = \{x_1, x_2, x_3, \dots, x_d\}$ . The dataset can be represented as a matrix of  $n$  rows and  $d$  columns as

$$M = \begin{bmatrix} p_1, x_1 & \dots & p_1, x_d \\ \vdots & \ddots & \vdots \\ p_n, x_1 & \dots & p_n, x_d \end{bmatrix}$$

•  $M^T M$  is the covariance matrix



# Example

- $M = \begin{bmatrix} -4 & -20 & -11 \\ -2 & -30 & -7 \\ 0 & -10 & -3 \\ 1 & 60 & 6 \\ 5 & 0 & 15 \end{bmatrix}$

- $M^T M = \begin{bmatrix} -4 & -2 & 0 & 1 & 5 \\ -20 & -30 & -10 & 60 & 0 \\ -11 & -7 & -3 & 6 & 15 \end{bmatrix} * \begin{bmatrix} -4 & -20 & -11 \\ -2 & -30 & -7 \\ 0 & -10 & -3 \\ 1 & 60 & 6 \\ 5 & 0 & 15 \end{bmatrix}$

- $A = M^T M = \begin{bmatrix} 46 & 440 & 139 \\ 440 & 5000 & 1360 \\ 139 & 1360 & 440 \end{bmatrix}$

# Eigen values

- $AV = \lambda V$ 
  - $A$  is an  $n$ -by- $n$  matrix
  - $V$  is a non-zero  $n$ -by-1 vector
  - $\lambda$  is a scalar/eigenvalue

- $A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$

- $V = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}$

# Eigen values

- $AV = \lambda V$
- $AV - \lambda V = 0$
- $AV - \lambda . I . V = 0$
- $(A - \lambda I) . V = 0$
- for  $V \neq 0$
- $(A - \lambda I) = 0$

# Example

- $A - \lambda I = \begin{bmatrix} 6 & 1 \\ 4 & 5 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
- $A - \lambda I = \begin{bmatrix} 6 & 1 \\ 4 & 5 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$
- $A - \lambda I = \begin{bmatrix} 6 - \lambda & 1 \\ 4 & 5 - \lambda \end{bmatrix}$
- $\begin{vmatrix} 6 - \lambda & 1 \\ 4 & 5 - \lambda \end{vmatrix} = 0$
- $((6 - \lambda) * (5 - \lambda) - (4)) = 0$
- $30 - 11\lambda - \lambda^2 - 4 = 0$
- $\lambda^2 + 11\lambda - 26 = 0$
- $\lambda^2 + 13\lambda - 2\lambda - 26 = 0$
- $\lambda(\lambda + 13) - 2(\lambda + 13) = 0$
- $\lambda = 2, \lambda = -13$

# Example: 2

$$\bullet A - \lambda I = \begin{bmatrix} 46 & 440 & 139 \\ 440 & 5000 & 1360 \\ 139 & 1360 & 440 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\bullet A - \lambda I = \begin{bmatrix} 46 & 440 & 139 \\ 440 & 5000 & 1360 \\ 139 & 1360 & 440 \end{bmatrix} - \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix}$$

$$\bullet A - \lambda I = \begin{bmatrix} 46 - \lambda & 440 & 139 \\ 440 & 5000 - \lambda & 1360 \\ 139 & 1360 & 440 - \lambda \end{bmatrix}$$

$$\bullet A - \lambda I = \begin{vmatrix} 46 - \lambda & 440 & 139 \\ 440 & 5000 - \lambda & 1360 \\ 139 & 1360 & 440 - \lambda \end{vmatrix} = 0$$

# Example: 2

$$\cdot (46 - \lambda) \begin{vmatrix} 5000 - \lambda & 1360 \\ 1360 & 440 - \lambda \end{vmatrix}$$

$$\cdot -(440) \begin{vmatrix} 440 & 1360 \\ 139 & 440 - \lambda \end{vmatrix}$$

$$\cdot +(139) \begin{vmatrix} 440 & 5000 - \lambda \\ 139 & 1360 \end{vmatrix}$$

• ...

# Eigen Vector

- $AV = \lambda V$

- For  $\lambda = 2$

- $\begin{bmatrix} 6 & 1 \\ 4 & 5 \end{bmatrix} * \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 2 \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$

- $6v_1 + 1v_2 = 2v_1, 4v_1 + 5v_2 = 2v_2$

- $4v_1 + v_2 = 0, 4v_1 + 3v_2 = 0$

- For  $v_1 = 1$

- $v_2 = -4$

- ∴ Eigen Vector  $e_1 = \begin{bmatrix} 1 \\ -4 \end{bmatrix}$

- The unit vector  $e_1 = \frac{1}{\sqrt{17}} \begin{bmatrix} 1 \\ -4 \end{bmatrix}$

- $e_1 = \begin{bmatrix} \frac{1}{\sqrt{(17)}} \\ -\frac{4}{\sqrt{(17)}} \end{bmatrix}$

# Eigen Vector

- $AV = \lambda V$
- For  $\lambda = -13$
- $\begin{bmatrix} 6 & 1 \\ 4 & 5 \end{bmatrix} * \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = -13 \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$
- $6v_1 + 1v_2 = -13v_1, 4v_1 + 5v_2 = -13v_2$
- $19v_1 + v_2 = 0, 4v_1 + 18v_2 = 0$
- For  $v_1 = 1$
- $v_2 = -19$

∴ Eigen Vector  $e_2 = \begin{bmatrix} 1 \\ -19 \end{bmatrix}$

• The unit vector  $e_2 = \frac{1}{\sqrt{362}} \begin{bmatrix} 1 \\ -19 \end{bmatrix}$

•  $e_2 = \begin{bmatrix} \frac{1}{\sqrt{362}} \\ -\frac{19}{\sqrt{362}} \end{bmatrix}$



# PCA: Feature Vector

- Eigen Vector  $e_1 = \begin{bmatrix} \frac{1}{\sqrt{(17)}} \\ 4 \\ -\frac{4}{\sqrt{(17)}} \end{bmatrix}$

- Eigen Vector  $e_2 = \begin{bmatrix} \frac{1}{\sqrt{362}} \\ 19 \\ -\frac{19}{\sqrt{362}} \end{bmatrix}$

- Feature Vector  $E = [e_1 \quad e_2] = \begin{bmatrix} \frac{1}{\sqrt{(17)}} & \frac{1}{\sqrt{362}} \\ -4 & 19 \\ \frac{4}{\sqrt{(17)}} & -\frac{19}{\sqrt{362}} \end{bmatrix}$