中国科学院大学人工智能学院硕士课《模式识别》

特征提取与特征选择

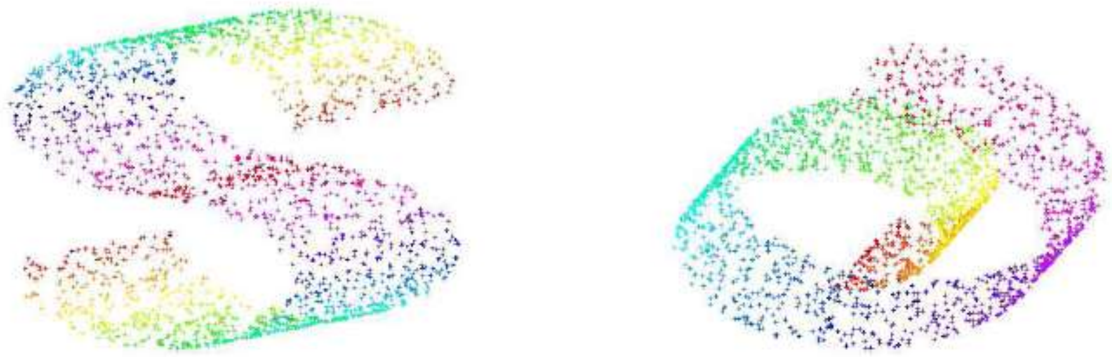
张煦尧(xyz@nlpr.ia.ac.cn)

2019年12月25日

Nonlinear Extensions

Nonlinear Extensions

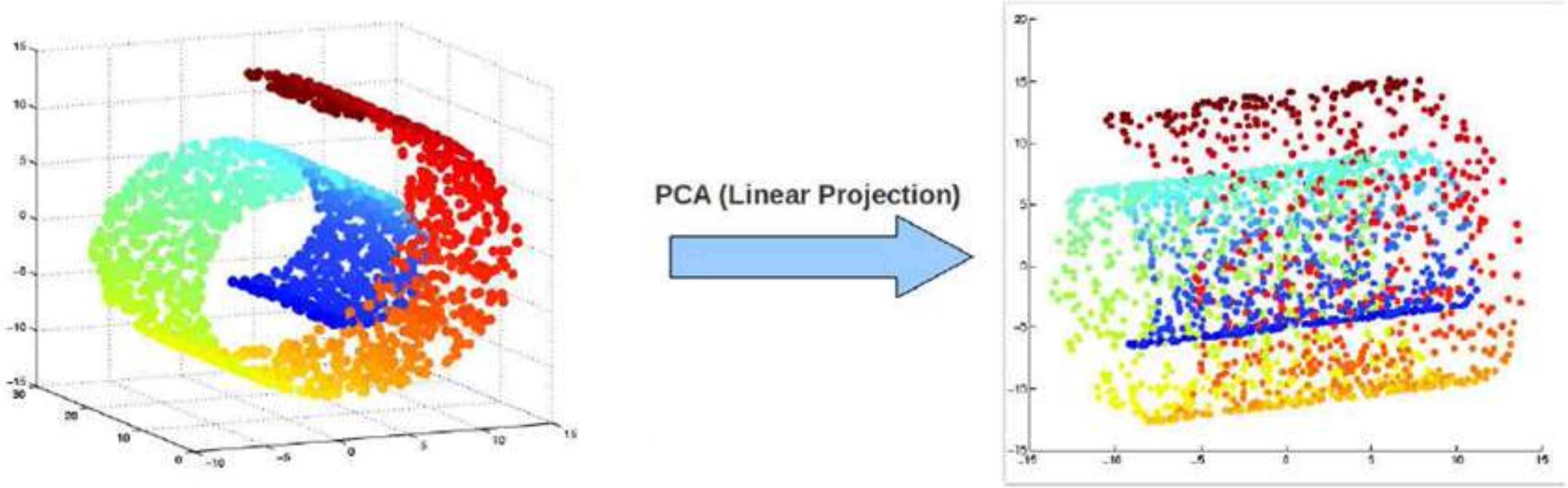
- Given: Low-dim. surface embedded **nonlinearly** in high-dim. space
  - Such a structure is called a **Manifold**



- Goal: Recover the low-dimensional surface

Nonlinear Extensions

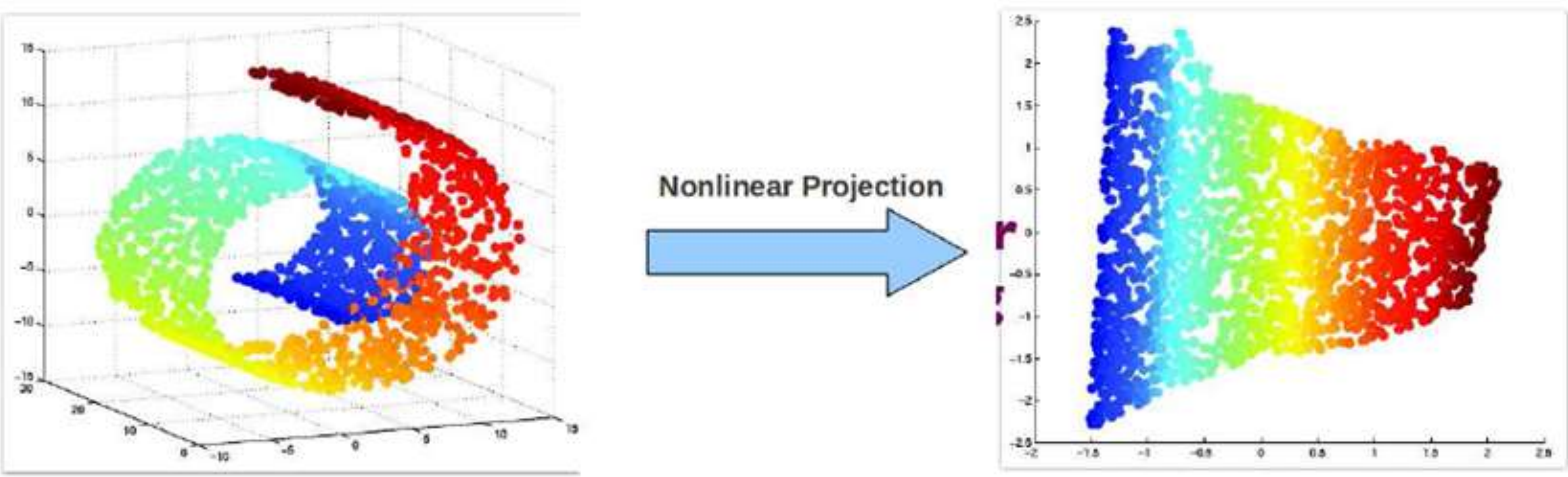
- Consider the swiss-roll dataset (points lying close to a manifold)



- Linear projection methods (e.g., PCA) can't capture intrinsic nonlinearities

Nonlinear Extensions

- We want to do nonlinear projections
- Different criteria could be used for such projections
- Most nonlinear methods try to preserve the neighborhood information
  - Locally linear structures (locally linear $\Rightarrow$ globally nonlinear)
  - Pairwise distances (along the nonlinear manifold)
- Roughly translates to "unrolling" the manifold



Nonlinear Extensions

Two ways of doing it:

- Nonlinearize a linear dimensionality reduction method. E.g.:
  - **Kernel PCA (nonlinear PCA)**
- Using manifold based methods. E.g.:
  - **Locally Linear Embedding (LLE)**
  - **Isomap**
  - Maximum Variance Unfolding
  - Laplacian Eigenmaps
  - And several others (Hessian LLE, Hessian Eigenmaps, etc.)

Kernel PCA

- Given N observations $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, $\forall \mathbf{x}_n \in \mathbb{R}^D$, define the $D \times D$ covariance matrix (assuming centered data $\sum_n \mathbf{x}_n = \mathbf{0}$)

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^\top$$

- Linear PCA: Compute eigenvectors $\mathbf{u}_i$ satisfying: $\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i \; \forall i = 1, \ldots, D$
- Consider a nonlinear transformation $\phi(\mathbf{x})$ of $\mathbf{x}$ into an M dimensional space
- $M \times M$ covariance matrix **in this space** (assume centered data $\sum_n \phi(\mathbf{x}_n) = \mathbf{0}$)

$$\mathbf{C} = \frac{1}{N} \sum_{n=1}^{N} \phi(\mathbf{x}_n) \phi(\mathbf{x}_n)^\top$$

- Kernel PCA: Compute eigenvectors $\mathbf{v}_i$ satisfying: $\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i \; \forall i = 1, \ldots, M$
- Ideally, we would like to do this without having to compute the $\phi(\mathbf{x}_n)$'s

Kernel PCA

- Kernel PCA: Compute eigenvectors $\mathbf{v}_i$ satisfying: $\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$
- Plugging in the expression for $\mathbf{C}$, we have the eigenvector equation:

$$\frac{1}{N}\sum_{n=1}^{N} \phi(\mathbf{x}_n)\{\phi(\mathbf{x}_n)^\top \mathbf{v}_i\} = \lambda_i \mathbf{v}_i$$

- Using the above, we can write $\mathbf{v}_i$ as: $\boxed{\mathbf{v}_i = \sum_{n=1}^{N} a_{in}\phi(\mathbf{x}_n)}$
- Plugging this back in the eigenvector equation:

$$\frac{1}{N}\sum_{n=1}^{N} \phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^\top \sum_{m=1}^{N} a_{im}\phi(\mathbf{x}_m) = \lambda_i \sum_{n=1}^{N} a_{in}\phi(\mathbf{x}_n)$$

- Pre-multiplying both sides by $\phi(\mathbf{x}_l)^\top$ and re-arranging

$$\frac{1}{N}\sum_{n=1}^{N} \phi(\mathbf{x}_l)^\top \phi(\mathbf{x}_n) \sum_{m=1}^{N} a_{im}\phi(\mathbf{x}_n)^\top \phi(\mathbf{x}_m) = \lambda_i \sum_{n=1}^{N} a_{in}\phi(\mathbf{x}_l)^\top \phi(\mathbf{x}_n)$$

Kernel PCA

- Using $\phi(\mathbf{x}_n)^\top \phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$, the eigenvector equation becomes:

$$\frac{1}{N}\sum_{n=1}^{N} k(\mathbf{x}_l, \mathbf{x}_n) \sum_{m=1}^{N} a_{im} k(\mathbf{x}_n, \mathbf{x}_m) = \lambda_i \sum_{n=1}^{N} a_{in} k(\mathbf{x}_l, \mathbf{x}_n)$$

- Define $\mathbf{K}$ as the $N \times N$ **kernel matrix** with $K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m)$
  - $\mathbf{K}$ is the similarity of two examples $\mathbf{x}_n$ and $\mathbf{x}_m$ in the ϕ space
  - ϕ is implicitly defined by kernel function k (which can be, e.g., RBF kernel)
- Define $\mathbf{a}_i$ as the $N \times 1$ vector with elements a_{in}
- Using $\mathbf{K}$ and $\mathbf{a}_i$, the eigenvector equation becomes:

$$\mathbf{K}^2\mathbf{a}_i = \lambda_i N \mathbf{K}\mathbf{a}_i \quad \Rightarrow \quad \boxed{\mathbf{K}\mathbf{a}_i = \lambda_i N \mathbf{a}_i}$$

- This corresponds to the original Kernel PCA eigenvalue problem $\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$
- For a projection to $K < D$ dimensions, top K eigenvectors of $\mathbf{K}$ are used

$$\boxed{\mathbf{v}_i = \sum_{n=1}^{N} a_{in}\phi(\mathbf{x}_n)}$$

Kernel PCA: Centering Data

- In PCA, we centered the data before computing the covariance matrix
- For kernel PCA, we need to do the same

$$\tilde{\phi}(\mathbf{x}_n) = \phi(\mathbf{x}_n) - \frac{1}{N}\sum_{l=1}^{N}\phi(\mathbf{x}_l)$$

- How does it affect the kernel matrix $\mathbf{K}$ which is eigen-decomposed?

$$\begin{aligned}
\tilde{K}_{nm} &= \tilde{\phi}(\mathbf{x}_n)^\top\tilde{\phi}(\mathbf{x}_m) \\
&= \phi(\mathbf{x}_n)^\top\phi(\mathbf{x}_m) - \frac{1}{N}\sum_{l=1}^{N}\phi(\mathbf{x}_n)^\top\phi(\mathbf{x}_l) - \frac{1}{N}\sum_{l=1}^{N}\phi(\mathbf{x}_l)^\top\phi(\mathbf{x}_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{l=1}^{N}\phi(\mathbf{x}_j)^\top\phi(\mathbf{x}_l) \\
&= k(\mathbf{x}_n,\mathbf{x}_m) - \frac{1}{N}\sum_{l=1}^{N}k(\mathbf{x}_n,\mathbf{x}_l) - \frac{1}{N}\sum_{l=1}^{N}k(\mathbf{x}_l,\mathbf{x}_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{l=1}^{N}k(\mathbf{x}_l,\mathbf{x}_l)
\end{aligned}$$

- In matrix notation, the centered $\boxed{\tilde{\mathbf{K}} = \mathbf{K} - \mathbf{1}_N\mathbf{K} - \mathbf{K}\mathbf{1}_N + \mathbf{1}_N\mathbf{K}\mathbf{1}_N}$
- $\mathbf{1}_N$ is the $N \times N$ matrix with every element $= 1/N$
- Eigen-decomposition is then done for the centered kernel matrix $\tilde{\mathbf{K}}$

Kernel PCA: The Projection

- Suppose $\{\mathbf{a}_1, \ldots, \mathbf{a}_K\}$ are the top K eigenvectors of kernel matrix $\tilde{\mathbf{K}}$
- The K-dimensional KPCA projection $\mathbf{z} = [z_1, \ldots, z_K]$ of a point $\mathbf{x}$:

$$z_i = \phi(\mathbf{x})^\top \mathbf{v}_i$$

- Recall the definition of $\mathbf{v}_i$

$$\boxed{\mathbf{v}_i = \sum_{n=1}^{N} a_{in} \phi(\mathbf{x}_n)}$$

- Thus

$$\boxed{z_i = \phi(\mathbf{x})^\top \mathbf{v}_i = \sum_{n=1}^{N} a_{in} k(\mathbf{x}, \mathbf{x}_n)}$$

Kernel Subspace Learning

KPCA plus LDA: a complete kernel Fisher discriminant framework for feature extraction and recognition
J Yang, AF Frangi, J Yang, D Zhang… - IEEE Transactions on …, 2005 - ieeexplore.ieee.org
This paper examines the theory of kernel Fisher discriminant analysis (KFD) in a Hilbert space and develops a two-phase KFD framework, ie, kernel principal component analysis (KPCA) plus Fisher linear discriminant analysis (LDA). This framework provides novel …
☆ 99 被引用次数：904 相关文章 所有 15 个版本

Kernel PCA

Fisher discriminant analysis with kernels
S Mika, G Ratsch, J Weston… - Neural networks for …, 1999 - ieeexplore.ieee.org
A non-linear classification technique based on Fisher's discriminant is proposed. The main ingredient is the kernel trick which allows the efficient computation of Fisher discriminant in feature space. The linear classification in feature space corresponds to a (powerful) non …
☆ 99 被引用次数：3050 相关文章 所有 13 个版本

Kernel LDA

Kernel and nonlinear canonical correlation analysis
PL Lai, C Fyfe - International Journal of Neural Systems, 2000 - World Scientific
… 2. We then use the kernel methods popularised by Support Vector Machines (SVMs) to create a Kernel CCA method … This is a somewhat more ad hoc proce- dure than that used to derive Kernel CCA. Thus we calculate y1 and y2 using y1 = ∑ j w1j tanh(v1jx1j) = w1f1 and …
☆ 99 被引用次数：360 相关文章 所有 11 个版本

Kernel CCA

Kernel ICA: An alternative formulation and its application to face recognition
J Yang, X Gao, D Zhang, J Yang - Pattern Recognition, 2005 - Elsevier
This paper formulates independent component analysis (ICA) in the kernel-inducing feature space and develops a two-phase kernel ICA algorithm: whitened kernel principal component analysis (KPCA) plus ICA. KPCA spheres data and makes the data structure …
☆ 99 被引用次数：123 相关文章 所有 8 个版本

Kernel ICA

Manifold Learning

- **Locally Linear Embedding (LLE)**
- **Isomap**
- Maximum Variance Unfolding
- Laplacian Eigenmaps
- And several others (Hessian LLE, Hessian Eigenmaps, etc.)

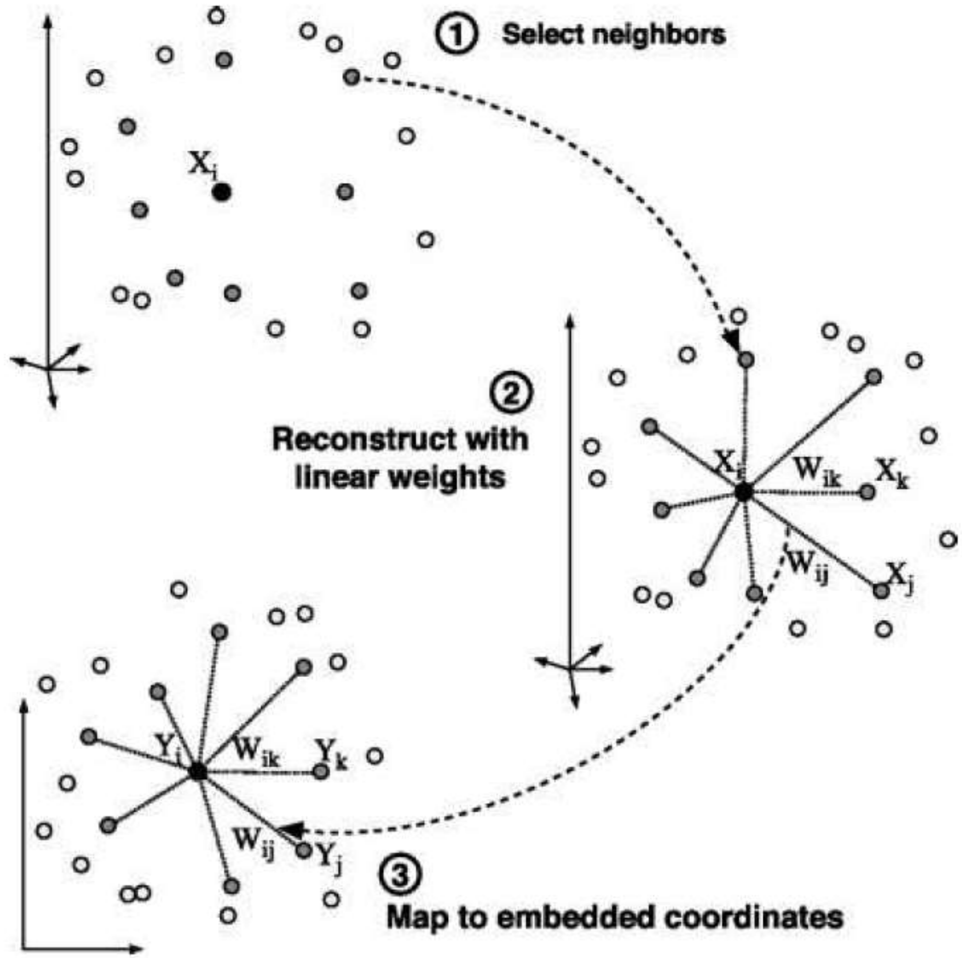
Locally Linear Embedding (LLE)

Nonlinear dimensionality reductio

ST Roweis, LK Saul - science, 2000 - scien

Many areas of science depend on explorato
analyze large amounts of multivariate data r
reduction: how to discover compact represe

☆ 99 被引用次数：12882 相关文章



- Based on a simple geometric intuitic
- Assume each example and its neighl patch of the manifold
- LLE assumption: Projection should
  - Projected point should have the s

Locally Linear Embedding (LLE)

- Given D dim. data $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, compute K dim. projections $\{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$
- For each example $\mathbf{x}_i$, find its L nearest neighbors
- Assume $\mathbf{x}_i$ to be a weighted linear combination of the L nearest neighbors

$$\mathbf{x}_i \approx \sum_{j \in \mathcal{N}} W_{ij} \mathbf{x}_j \qquad \text{(so the data is assumed locally linear)}$$

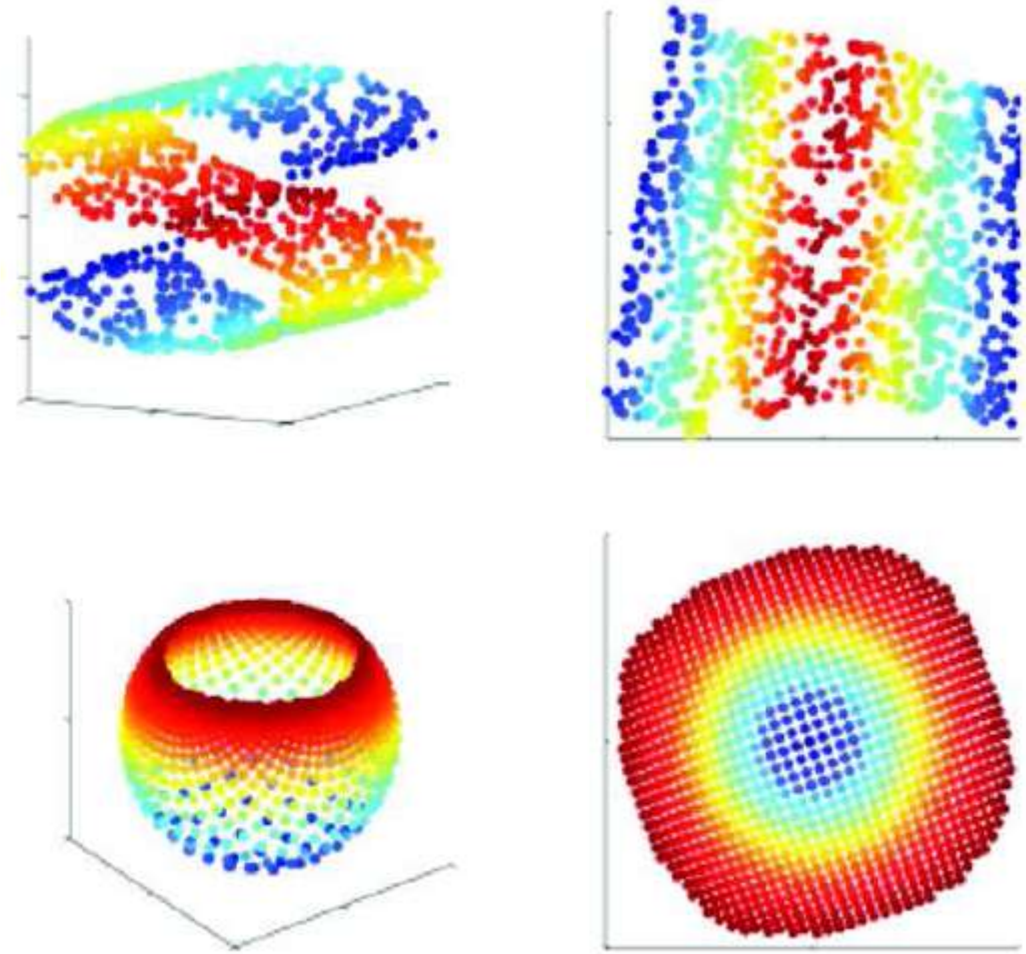
- Find the weights by solving the following least-squares problem:

$$\boxed{W = \arg\min_W \sum_{i=1}^{N} ||\mathbf{x}_i - \sum_{j \in \mathcal{N}_i} W_{ij} \mathbf{x}_j||^2} \qquad s.t. \forall i \quad \sum_j W_{ij} = 1$$

- $\mathcal{N}_i$ are the L nearest neighbors of $\mathbf{x}_i$ (note: should choose $L \geq K + 1$)
- Use W to compute low dim. projections $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$ by solving:

$$\boxed{\mathbf{Z} = \arg\min_{\mathbf{Z}} \sum_{i=1}^{N} ||\mathbf{z}_i - \sum_{j \in \mathcal{N}} W_{ij} \mathbf{z}_j||^2} \qquad s.t. \forall i \quad \sum_{i=1}^{N} \mathbf{z}_i = 0, \quad \frac{1}{N} \mathbf{Z}\mathbf{Z}^\top = \mathbf{I}$$

LLE: Examples

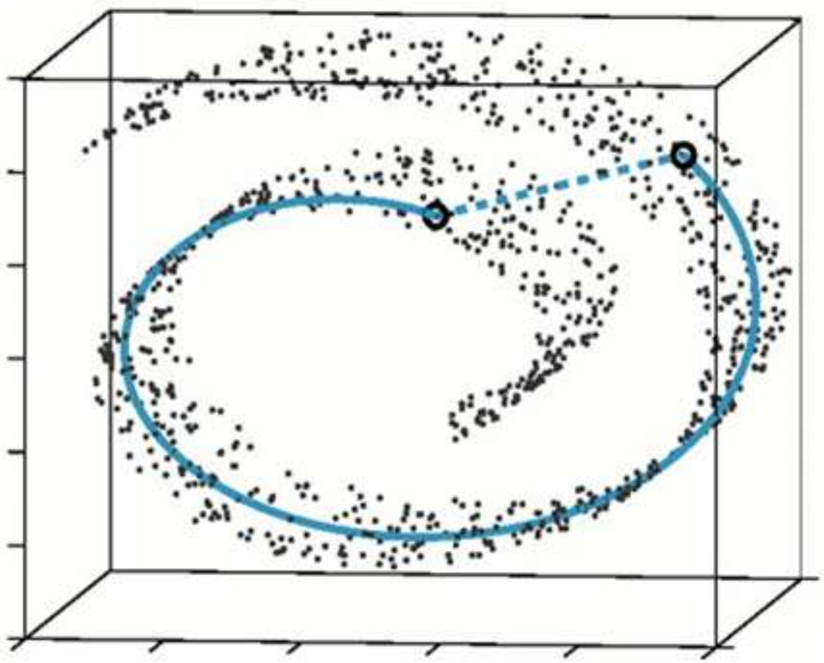


✓ ***Nonlinear dimension reduction***

✓ ***Out-of-sample problem***

Isometric Feature Mapping (ISOMAP)

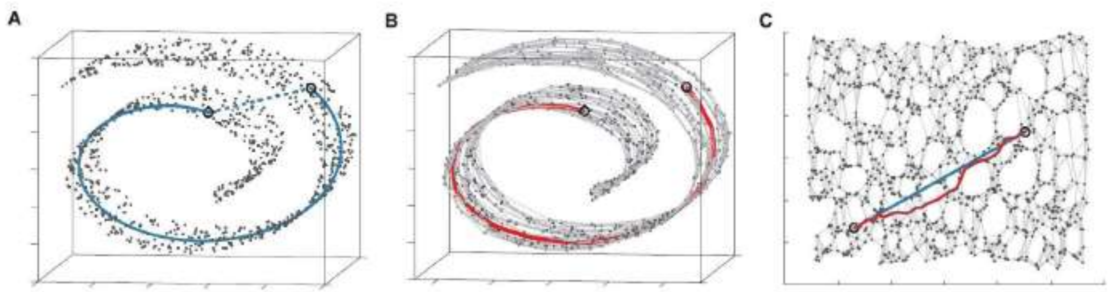
A global geometric framework for nonlinear dimensionality reduction
JB Tenenbaum, V De Silva, JC Langford - science, 2000 - science.sciencemag.org
… Here we describe an approach that combines the major algorithmic features of PCA and MDS—computational efficiency, global optimality, and … The complete isometric feature mapping, or Isomap, algorithm has three steps, which are detailed in Table 1. The first step determines …
☆ 99 被引用次数：11858 相关文章 所有 95 个版本



Isometric Feature Mapping (ISOMAP)

A graph based algorithm based on constructing a matrix of geodesic distances

- Identify the L nearest neighbors for each data point (just like LLE)
- Connect each point to all its neighbors (an edge for each neighbor)
- Assign weight to each edge based on the Euclidean distance
- Estimate the geodesic distance d_{ij} between any two data points i and j
  - Approximated by the sum of arc lengths along the shortest path between i and j in the graph (can be computed using Djikstras algorithm)
- Construct the $N \times N$ distance matrix $\mathbf{D} = \{d_{ij}^2\}$



Isometric Feature Mapping (ISOMAP)

- Use the distance matrix $\mathbf{D}$ to construct the Gram Matrix

$$\boxed{\mathbf{G} = -\frac{1}{2}\mathbf{HDH}}$$

where $\mathbf{G}$ is $N \times N$ and

$$\mathbf{H} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\top$$

$\mathbf{I}$ is $N \times N$ identity matrix, $\mathbf{1}$ is $N \times 1$ vector of 1s

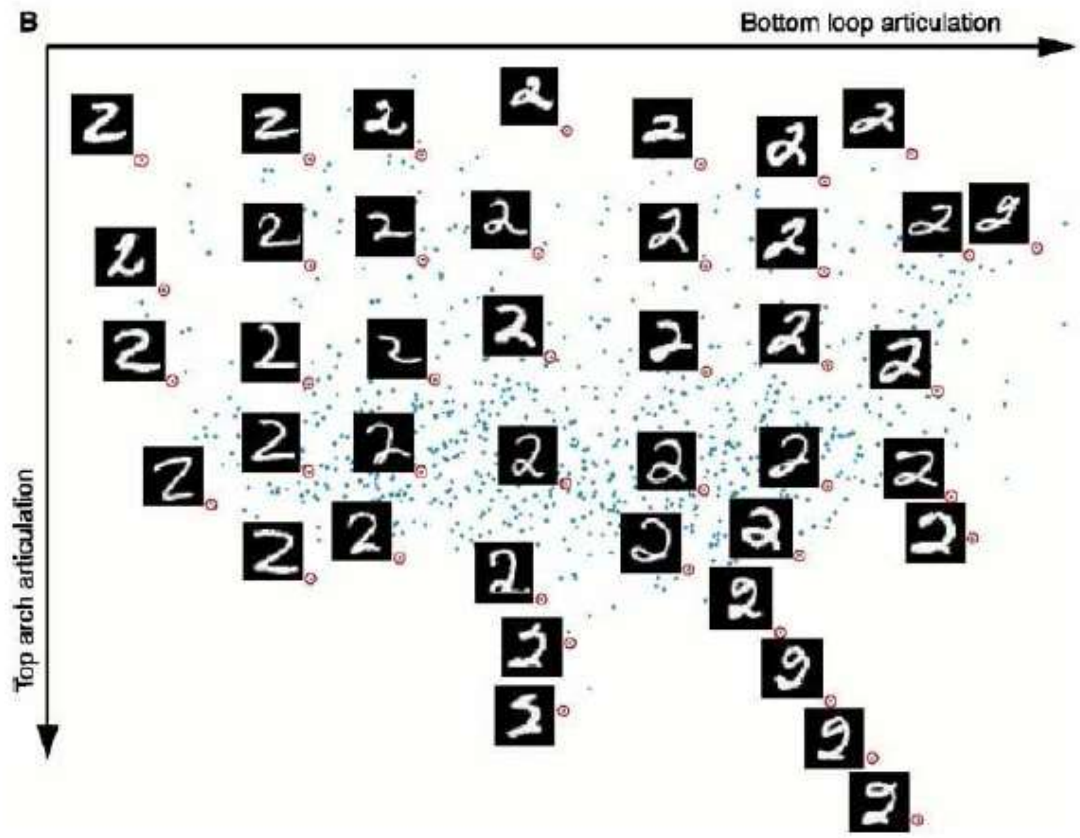
- Do an eigen decomposition of $\mathbf{G}$
- Let the eigenvectors be $\{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$ with eigenvalues $\{\lambda_1, \ldots, \lambda_N\}$
  - Each eigenvector $\mathbf{v}_i$ is N-dimensional: $\mathbf{v}_i = [v_{1i}, v_{2i}, \ldots, v_{Ni}]$
- Take the top K eigenvalue/eigenvectors
- The K dimensional embedding $\mathbf{z}_i = [z_{i1}, z_{i2}, \ldots, z_{iK}]$ of a point $\mathbf{x}_i$:

$$\boxed{z_{ik} = \sqrt{\lambda_k}\, v_{ki}}$$

Out-of-sample problem

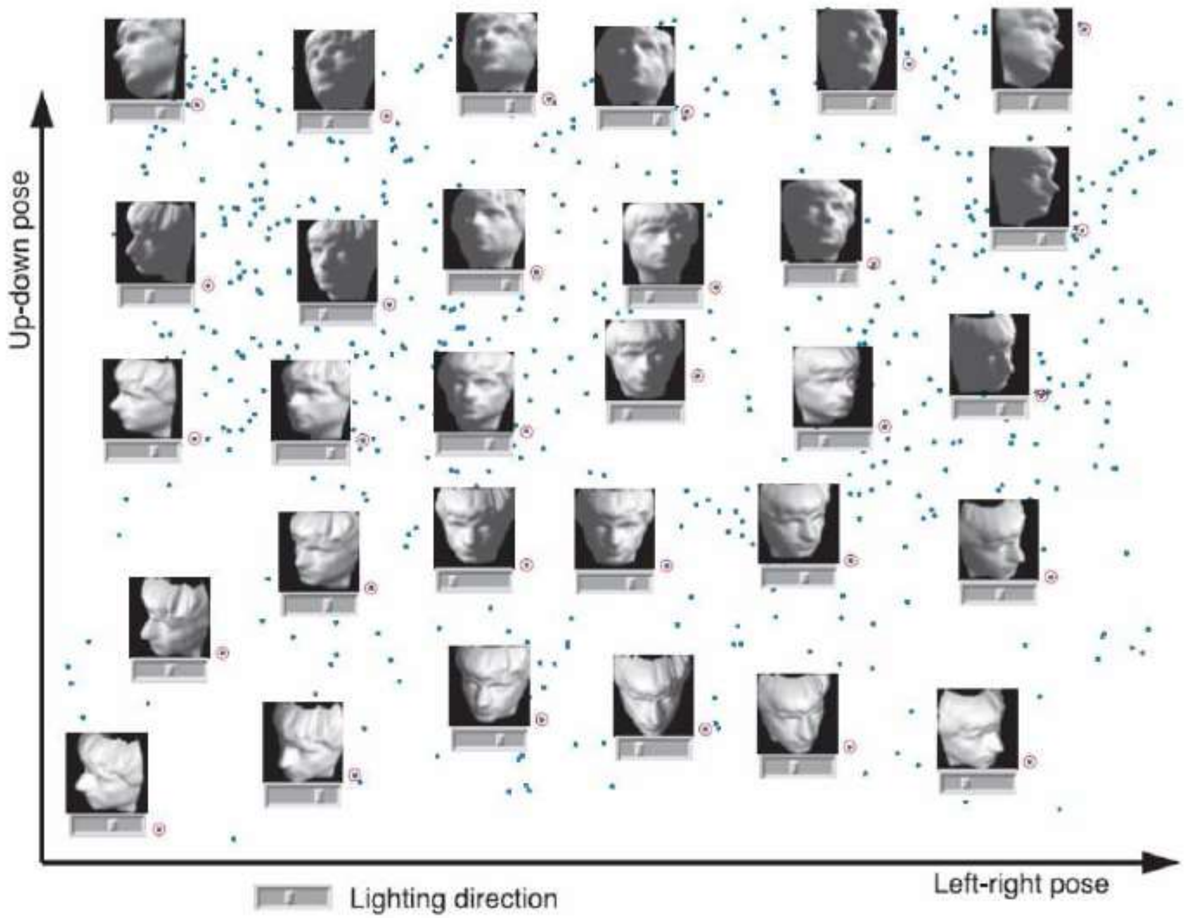
ISOMAP: Example

Digit images projected down to 2 dimensions



ISOMAP: Example

Face images with varying poses



LPP: Locality Preserving Projection

- **Out-of-sample problem:**
  - LLE and ISOMAP are computationally intensive
  - **The embedding is only defined on actual data points**.
- Solution:
  - LPP is a **linear method that approximates nonlinear methods** (specifically, the Laplacian Eigenmap.)
  - LPP is a linear approximation to nonlinear methods, which takes locality into account

$$\min \sum_{ij} (y_i - y_j)^2 S_{ij}$$

$$S_{ij} = \begin{cases} \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / t), & \|\mathbf{x}_i - \mathbf{x}_j\|^2 < \varepsilon \\ 0 & \text{otherwise} \end{cases}$$

$$S_{ij} = \begin{cases} \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / t), & \text{if } \mathbf{x}_i \text{ is among } k \text{ nearest neighbors of } \mathbf{x}_j \\ & \text{or } \mathbf{x}_j \text{ is among } k \text{ nearest neighbors of } \mathbf{x}_i \\ 0 & \text{otherwise,} \end{cases}$$

LPP: Locality Preserving Projection

$$
\begin{aligned}
&\tfrac{1}{2}\sum_{ij}(y_i - y_j)^2 S_{ij} \\
&= \tfrac{1}{2}\sum_{ij}\left(\boxed{\mathbf{w}^T \mathbf{x}_i} - \boxed{\mathbf{w}^T \mathbf{x}_j}\right)^2 S_{ij} \\
&= \sum_{ij} \mathbf{w}^T \mathbf{x}_i S_{ij} \mathbf{x}_i^T \mathbf{w} - \sum_{ij} \mathbf{w}^T \mathbf{x}_i S_{ij} \mathbf{x}_j^T \mathbf{w} \\
&= \sum_i \mathbf{w}^T \mathbf{x}_i D_{ii} \mathbf{x}_i^T \mathbf{w} - \mathbf{w}^T XSX^T \mathbf{w} \\
&= \mathbf{w}^T XDX^T \mathbf{w} - \mathbf{w}^T XSX^T \mathbf{w} \\
&= \mathbf{w}^T X(D-S)X^T \mathbf{w} \\
&= \boxed{\mathbf{w}^T XLX^T \mathbf{w}}
\end{aligned}
$$

The matrix D provides a natural measure on data points → a measure importance of the ith image

So a constraint can be imposed

$$
\mathbf{y}^T D\mathbf{y} = 1
$$

$$
\Rightarrow \quad \mathbf{w}^T XDX^T \mathbf{w} = 1
$$

Thus the optimization problem is:

$$
\underset{\substack{\mathbf{w} \\ \mathbf{w}^T XDX^T \mathbf{w}=1}}{\arg\min} \quad \mathbf{w}^T XLX^T \mathbf{w}
$$

- The solution is the Generalized Eigenvalue problem.
- The solution is also called Laplacianfaces.

Face Recognition

[PDF] Locality preserving projections
X He, P Niyogi - Advances in neural information processing systems, 2004 - papers.nips.cc
Many problems in information processing involve some form of dimensionality reduction. In this paper, we introduce Locality Preserving Projections (LPP). These are linear projective maps that arise by solving a variational problem that optimally preserves the neighborhood …
☆ 99 被引用次数：4137 相关文章 所有 17 个版本 »

Eigenface (PCA) – preserves global structure of image space (unsupervised)

Fischerface (LDA) – preserves discriminating information (supervised)

Laplacianface (LPP) – preserves local structure of image space (unsupervised)

TABLE 1
Performance Comparison on the Yale Database

Approach	Dims	Error Rate
Eigenfaces	33	25.3%
Fisherfaces	14	20.0%
Laplacianfaces	**28**	**11.3%**

TABLE 2
Performance Comparison on the PIE Database

Approach	Dims	Error Rate
Eigenfaces	150	20.6%
Fisherfaces	67	5.7%
Laplacianfaces	**110**	**4.6%**

特征选择
Feature Selection

降维 vs 特征选择

- Dimensionality reduction
  - All original features are used
  - The transformed features are linear combinations of the original features
- Feature selection
  - Only a subset of the original features are selected

- Continuous versus discrete

Feature Selection

- Definitions of subset optimality

- Perspectives of feature selection
  - Subset search and feature ranking
  - Feature/subset evaluation measures
  - Models: filter vs. wrapper
  - Results validation and evaluation

An Example for Optimal Subset

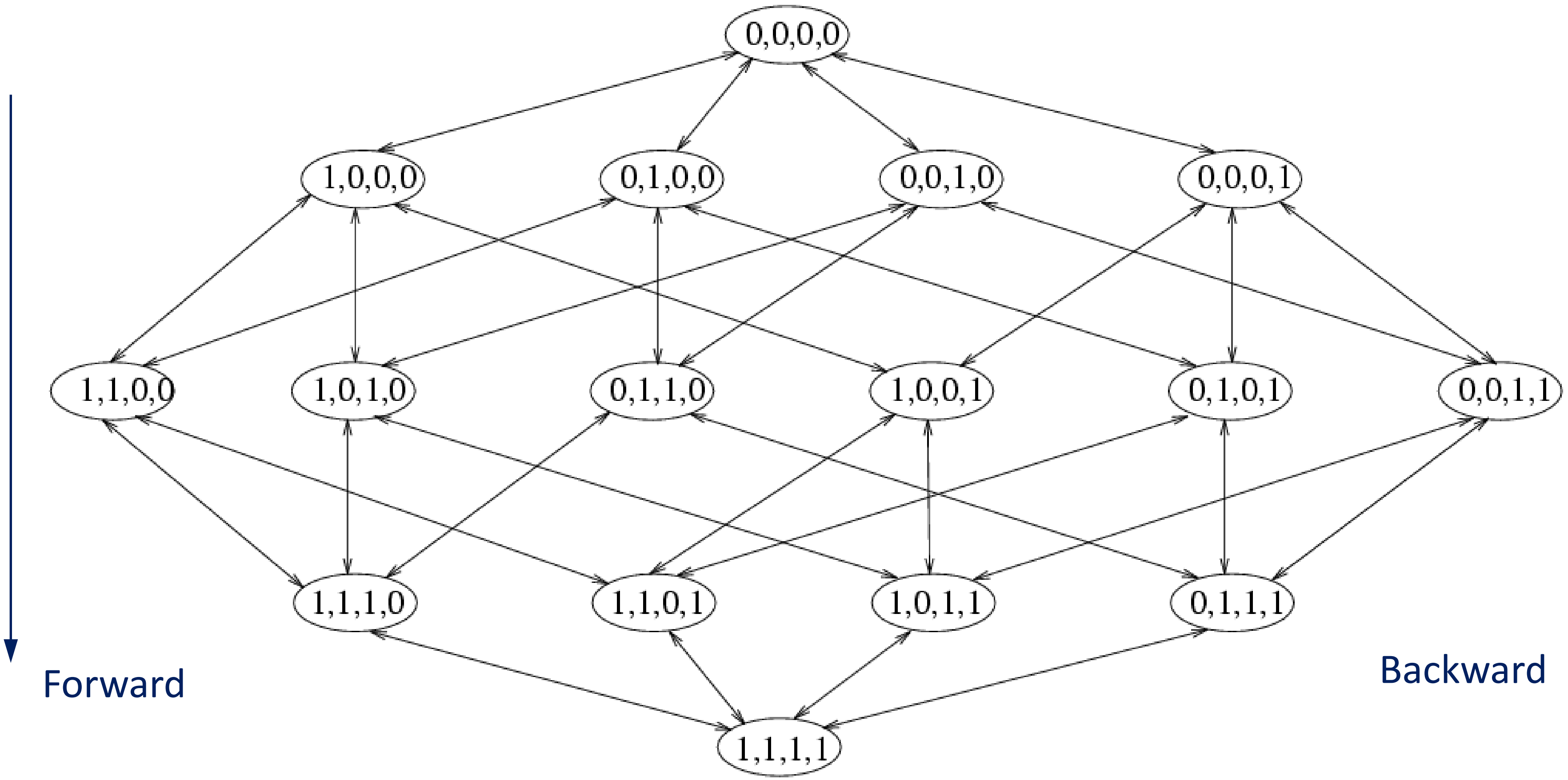
F_1	F_2	F_3	F_4	F_5	C
0	0	1	0	1	0
0	1	0	0	1	1
1	0	1	0	1	1
1	1	0	0	1	1
0	0	1	1	0	0
0	1	0	1	0	1
1	0	1	1	0	1
1	1	0	1	0	1

- Data set (whole set)
  - Five Boolean features
  - $C = F_1 \vee F_2$
  - $F_3 = \neg F_2$, $F_5 = \neg F_4$
  - Optimal subset:

    $\{F_1, F_2\}$ or $\{F_1, F_3\}$
- Combinatorial nature of searching for an optimal subset

Subset Search Problem

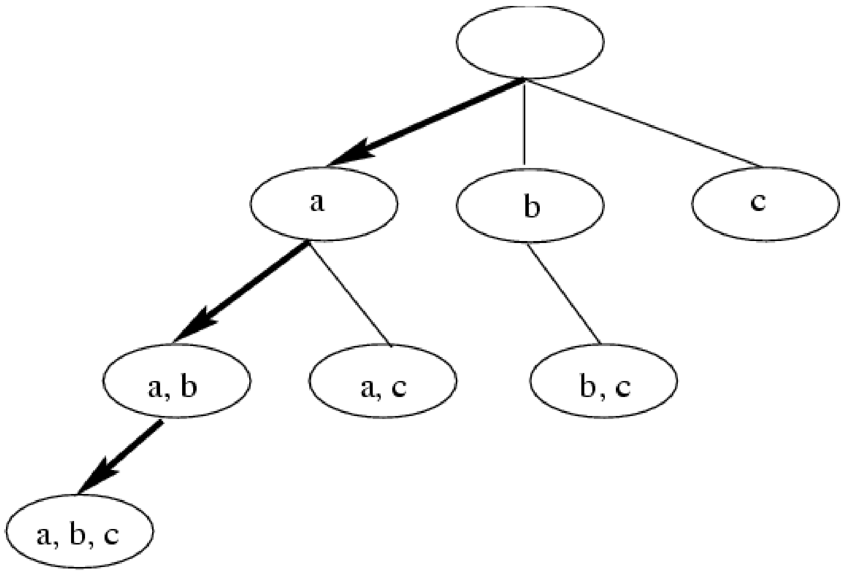
- An example of search space (*Kohavi & John 1997*)



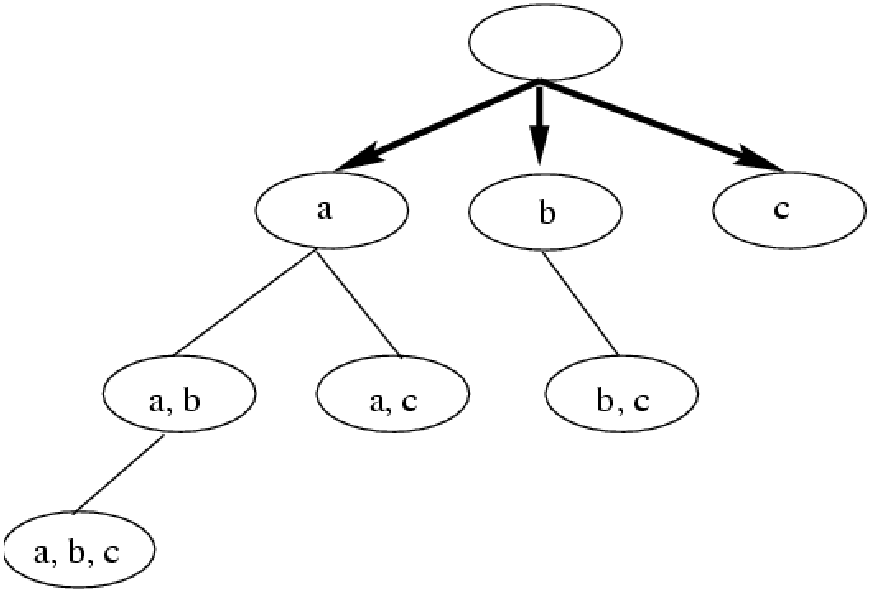
Different Aspects of Search

- Search starting points
  - Empty set
  - Full set
  - Random point
- Search directions
  - Sequential forward selection
  - Sequential backward elimination
  - Bidirectional generation
  - Random generation
- Search Strategies
  - Exhaustive/complete search
  - Heuristic search
  - Nondeterministic search

Illustration of Search Strategies



Depth-first search

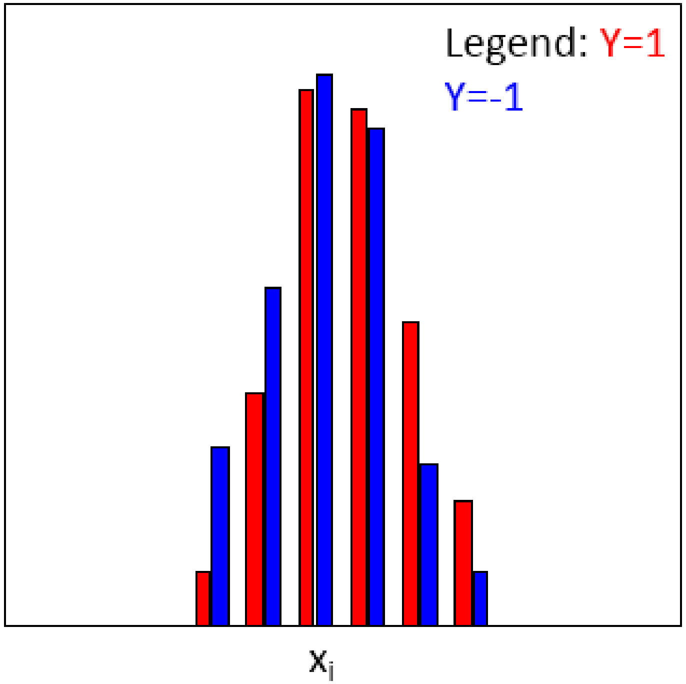


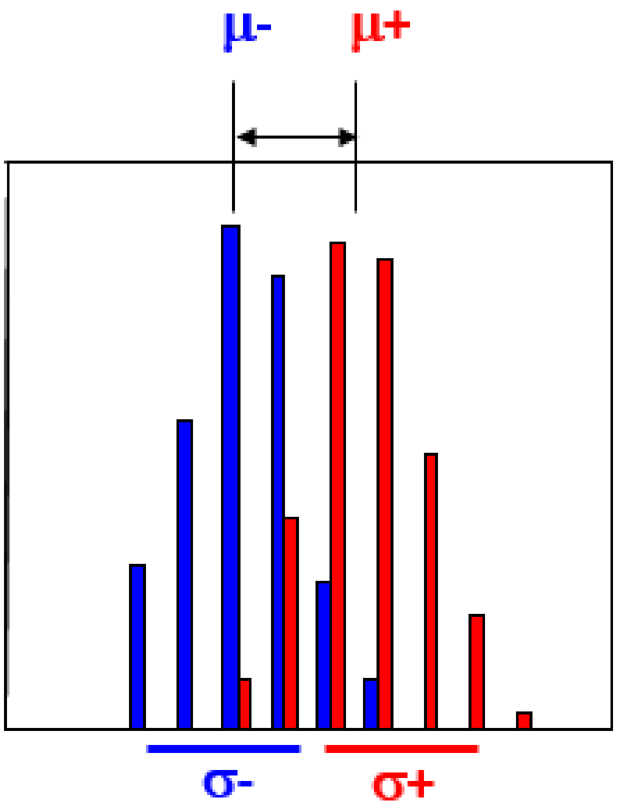
Breadth-first search

Feature Ranking

- Weighting and ranking individual features
- Selecting top-ranked ones for feature selection
- Advantages
  - Efficient: $O(N)$ in terms of dimensionality N
  - Easy to implement
- Disadvantages
  - Hard to determine the threshold
  - Unable to consider correlation between features

Individual Feature Measures





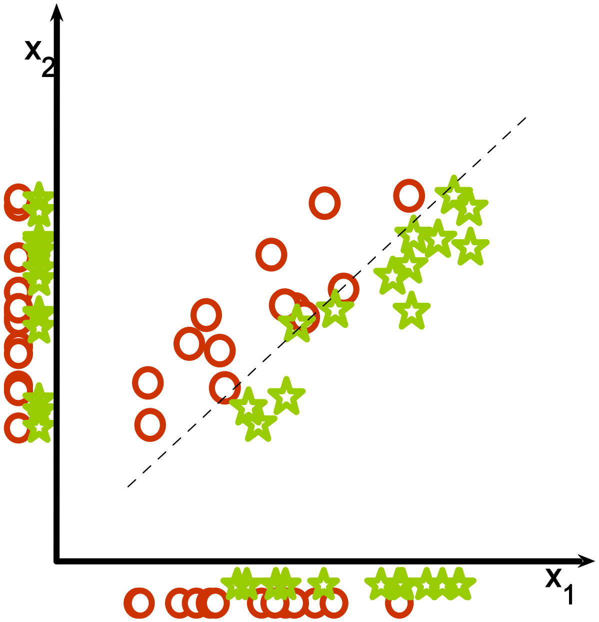
$P(X_i, Y) = P(X_i)\, P(Y)$

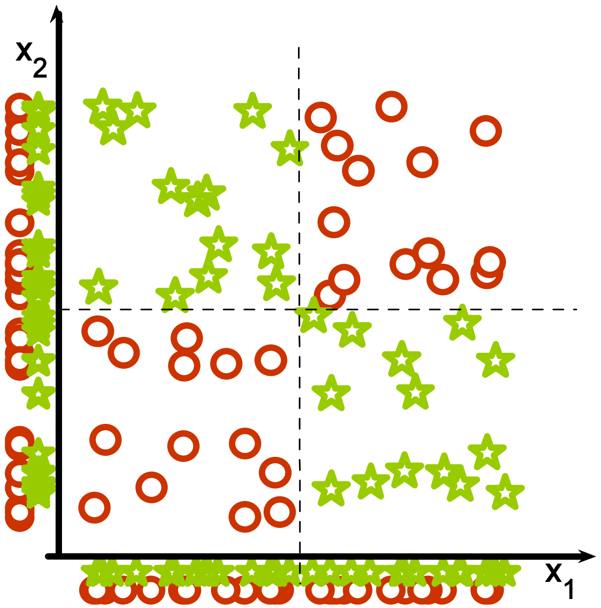
$P(X_i \mid Y) = P(X_i)$

$P(X_i \mid Y=1) = P(X_i \mid Y=-1)$

$$FDR = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}$$

Univariate Selection May Fail





Evaluation Measures

- The goodness of a feature/feature subset is dependent on measures
- Various measures
  - Information measures (Yu & Liu 2004, Jebara & Jaakkola 2000)
  - Distance measures (Robnik & Kononenko 03, Pudil & Novovicov 98)
  - Dependence measures (Hall 2000, Modrzejewski 1993)
  - Consistency measures (Almuallim & Dietterich 94, Dash & Liu 03)
  - Accuracy measures (Dash & Liu 2000, Kohavi&John 1997)

Illustrative Data set

	Hair	Height	Weight	Lotion	Result
i_1	1	2	1	0	1
i_2	1	3	2	1	0
i_3	2	1	2	1	0
i_4	1	1	2	0	1
i_5	3	2	3	0	1
i_6	2	3	3	0	0
i_7	2	2	3	0	0
i_8	1	1	1	1	0

Sunburn data

	Result (Sunburn)	
	No	Yes
P(Result)	5/8	3/8
P(Hair=1|Result)	2/5	2/3
P(Hair=2|Result)	3/5	0
P(Hair=3|Result)	0	1/3
P(Height=1|Result)	2/5	1/3
P(Height=2|Result)	1/5	2/3
P(Height=3|Result)	2/5	0
P(Weight=1|Result)	1/5	1/3
P(Weight=2|Result)	2/5	1/3
P(Weight=3|Result)	2/5	1/3
P(Lotion=0|Result)	2/5	3/3
P(Lotion=1|Result)	3/5	0

Priors and class conditional probabilities

Information Measures

- Entropy of variable *X*

$$H(X) = -\sum_i P(x_i) \log_2(P(x_i))$$

- Entropy of *X* after observing *Y*

$$H(X|Y) = -\sum_j P(y_j) \sum_i P(x_i|y_j) \log_2(P(x_i|y_j))$$

- Information Gain

$$IG(X|Y) = H(X) - H(X|Y)$$

Distance Measures

– Distance Measures.
  - Measures of separability, discrimination or divergence measures. The most typical is derived from distance between the class conditional density functions.

	Mathematical form
Euclidean distance	$D_e = \left\{\sum_{i=1}^{m} (x_i - y_i)^2\right\}^{\frac{1}{2}}$
City-block distance	$D_{cb} = \sum_{i=1}^{m} \lvert x_i - y_i \rvert$
Cebyshev distance	$D_{ch} = \max_i \lvert x_i - y_i \rvert$
Minkowski distance of order m	$D_M = \left\{\sum_{i=1}^{m} (x_i - y_i)^m\right\}^{\frac{1}{m}}$
Quadratic distance Q, positive definite	$D_q = \sum_{i=1}^{m} \sum_{j=1}^{m} (x_i - y_i) Q_{ij} (x_j - y_j)$
Canberra distance	$D_{ca} = \sum_{i=1}^{m} \frac{\lvert x_i - y_i \rvert}{x_i + y_i}$
Angular separation	$D_{as} = \frac{\sum_{i=1}^{m} x_i \cdot y_i}{\left[\sum_{i=1}^{m} x_i^2 \sum_{i=1}^{m} y_i^2\right]^{\frac{1}{2}}}$

Consistency Measures

- Consistency measures
  - Trying to find a minimum number of features that separate classes as consistently as the full set can
  - They aim to achieve **P(C|FullSet) = P(C|SubSet)**.
  - An inconsistency is defined as two instances having the same feature values but different classes
    - E.g., one inconsistency is found between instances i4 and i8 if we just look at the first two columns of the data table

	Hair	Height	Weight	Lotion	Result
i_1	1	2	1	0	1
i_2	1	3	2	1	0
i_3	2	1	2	1	0
i_4	1	1	2	0	1
i_5	3	2	3	0	1
i_6	2	3	3	0	0
i_7	2	2	3	0	0
i_8	1	1	1	1	0

Dependence Measures

- Dependence Measures.
  - known as measures of association or correlation.
  - Its main goal is to quantify how strongly two variables are correlated or present some association with each other, in such way that knowing the value of one of them, we can derive the value for the other.
  - *Pearson correlation* coefficient:

$$\rho(X,Y) = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\left[\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2\right]^{\frac{1}{2}}}$$

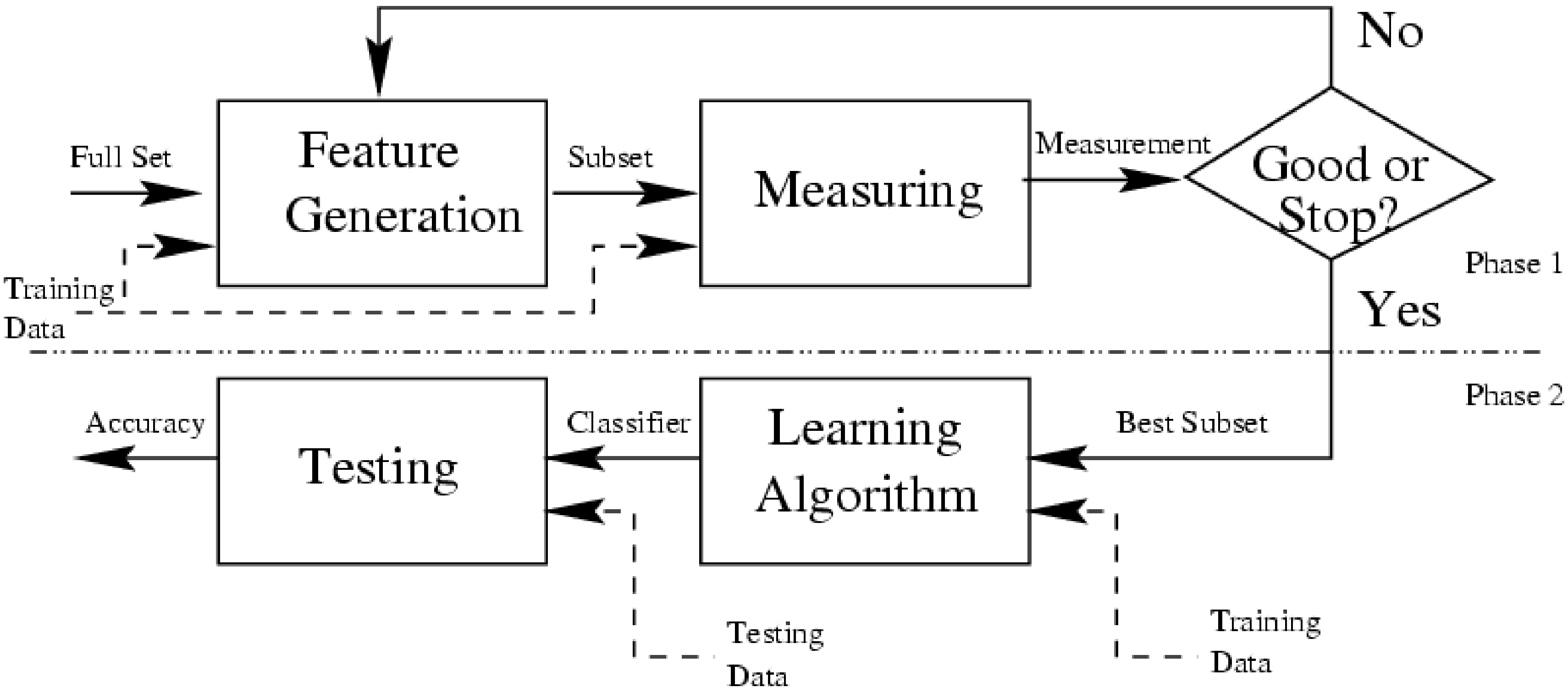
Accuracy Measures

- Using classification accuracy of a classifier as an evaluation measure
- Factors constraining the choice of measures
  - Classifier being used
  - The speed of building the classifier
- Compared with previous measures
  - Directly aimed to improve accuracy
  - Biased toward the classifier being used
  - More time consuming

Models of Feature Selection

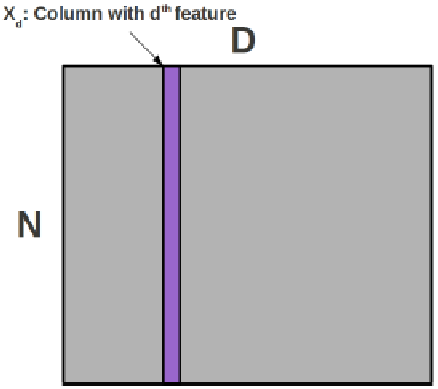
- Filter model
  - Separating feature selection from classifier learning
  - Relying on general characteristics of data (*information, distance*, *dependence*, *consistency*)
  - No bias toward any learning algorithm, fast
- Wrapper model
  - Relying on a predetermined classification algorithm
  - Using predictive accuracy as goodness measure
  - High accuracy, computationally expensive
- Embedded model
  - Feature selected during learning process

Filter Model



Filter Feature Selection

- Uses heuristics but is much faster than wrapper methods



- **Correlation Critera:** Rank features in order of their correlation with the labels

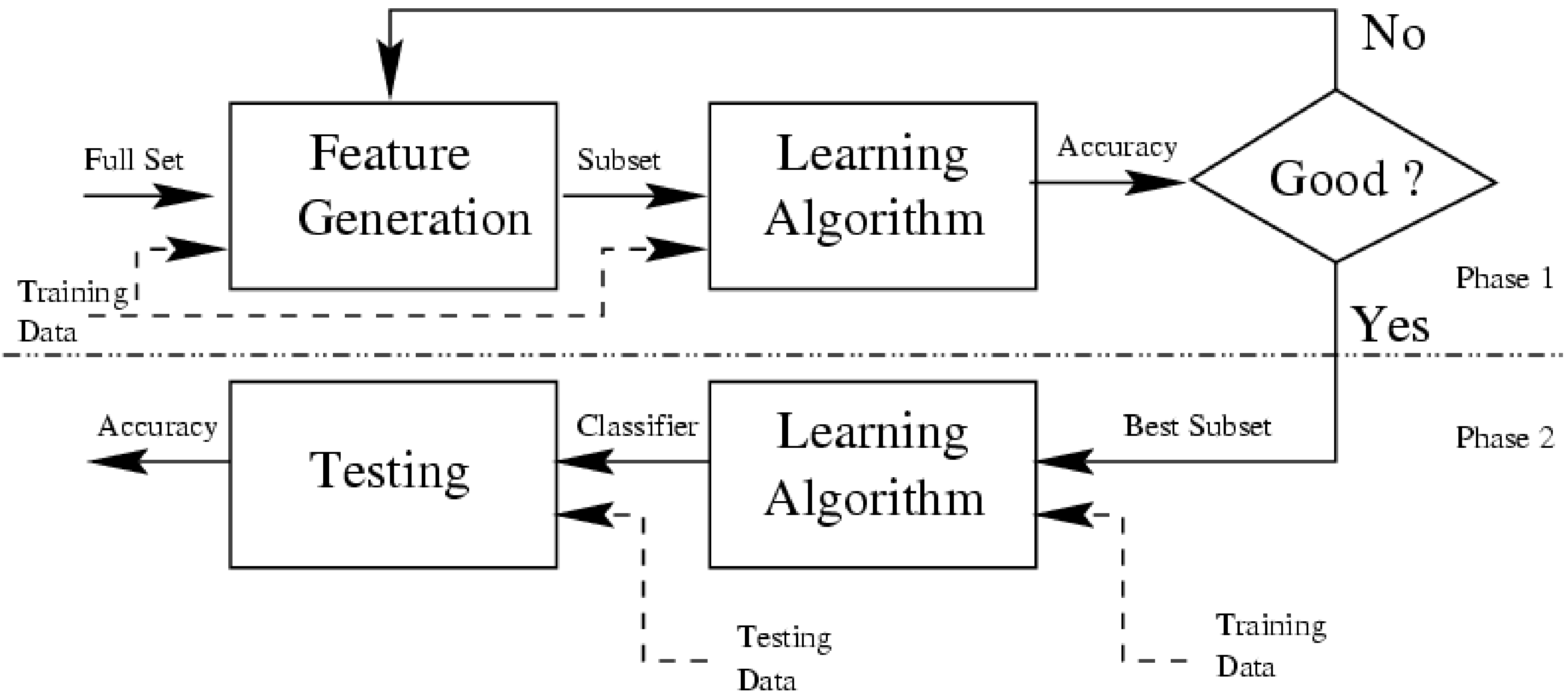
$$R(X_d, Y) = \frac{cov(X_d, Y)}{\sqrt{var(X_d)var(Y)}}$$

- **Mutual Information Criteria:**

$$MI(X_d, Y) = \sum_{X_d \in \{0,1\}} \sum_{Y \in \{-1,+1\}} P(X_d, Y) \frac{\log P(X_d, Y)}{P(X_d)P(Y)}$$

  - High mutual information mean high relevance of that feature

Wrapper Model



Wrapper Feature Selection

- **Forward Search**
  - Let $\mathcal{F} = \{\}$
  - While not selected desired number of features
  - For each unused feature f:
    - Estimate model's error on feature set $\mathcal{F} \bigcup f$ (using cross-validation)
  - Add f with lowest error to $\mathcal{F}$

- **Backward Search**
  - Let $\mathcal{F} = \{\text{all features}\}$
  - While not reduced to desired number of features
  - For each feature $f \in \mathcal{F}$:
    - Estimate model's error on feature set $\mathcal{F} \backslash f$ (using cross-validation)
  - Remove f with lowest error from $\mathcal{F}$

How to Validate Selection Results

- Direct evaluation (if we know *a priori* …)
  - Often suitable for artificial data sets
  - Based on prior knowledge about data

- Indirect evaluation (if we don't know …)
  - Often suitable for real-world data sets
  - Based on
    - number of features selected
    - performance on selected features (e.g., predictive accuracy, goodness of resulting clusters)
    - interpretability, speed

Methods for Result Evaluation

- Learning curves
  - For results in the form of a ranked list of features
- Before-and-after comparison
  - For results in the form of a minimum subset
- Comparison using different classifiers
  - To avoid learning bias of a particular classifier
- Repeating experimental results
  - For non-deterministic results

Representative Algorithms

- Filter algorithms
  - Feature ranking algorithms
    - Example: Relief (*Kira & Rendell 1992*)
  - Subset search algorithms
    - Example: consistency-based algorithms
      - Focus (*Almuallim & Dietterich, 1994*)

- Wrapper algorithms
  - Feature ranking algorithms
    - Example: SVM
  - Subset search algorithms
    - Example: RFE

Relief Algorithm

Relief

Input: $\mathbf{x}$ - features

m - number of instances sampled

τ - adjustable relevance threshold

initialize: $\mathbf{w} = 0$

for $i = 1$ to m

begin

randomly select an instance I

find nearest-hit H and nearest-miss J

for $j = 1$ to N

$\mathbf{w}(j) = \mathbf{w}(j) - \mathtt{diff}(j, I, H)^2/m + \mathtt{diff}(j, I, J)^2/m$

end

Output: $\mathbf{w}$ greater than τ

Focus Algorithm

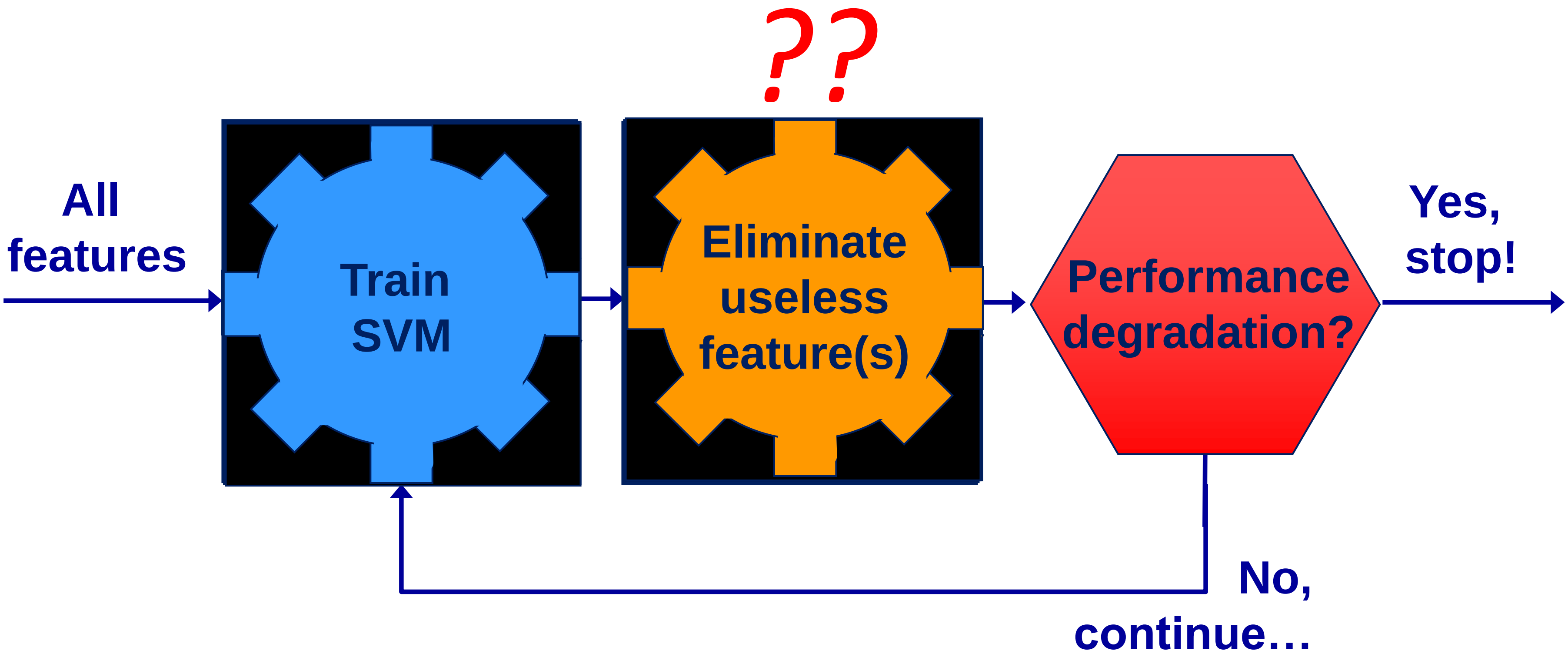
Focus

Input: F - all features x in data D
U - inconsistency rate as evaluation measure

initialize: $S = \{\}$

for $i = 1$ to N
  for each subset S of size i
    if $\mathrm{Cal}U(S, D) = 0$ /* $CalU(S, D)$ *returns inconsistency**/
      return S
Output: S - a minimum subset that satisfies U

Embedded Methods (RFE)



Recursive Feature Elimination (RFE) SVM. *Guyon-Weston, 2000. US patent 7,117,188*

Feature Selection via Regularization (Sparse)

- Data: $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$, $i = 1, \ldots, n$
- Minimize with respect to function $f : \mathcal{X} \rightarrow \mathcal{Y}$:

$$\sum_{i=1}^{n} \ell(y_i, f(x_i)) \quad + \quad \frac{\lambda}{2}\|f\|^2$$

Error on data + Regularization

Loss & function space ? Norm ?

- Two theoretical/algorithmic issues:

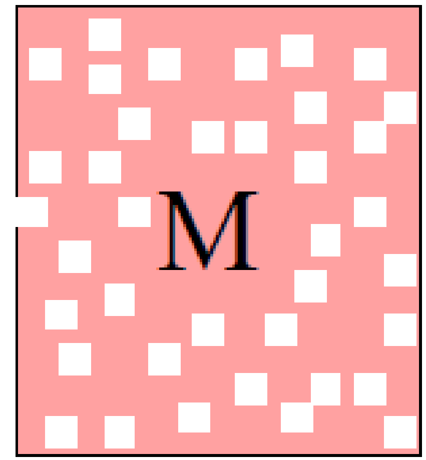
1. Loss
2. **Function space / norm**

Ridge Regression and LASSO

- Compared methods to reach the least-square solution
  - Ridge regression: $\min_{w \in \mathbb{R}^p} \frac{1}{2}\|y - Xw\|_2^2 + \frac{\lambda}{2}\|w\|_2^2$
  - Lasso: $\min_{w \in \mathbb{R}^p} \frac{1}{2}\|y - Xw\|_2^2 + \lambda\|w\|_1$
  - Forward greedy:
    - Initialization with empty set
    - Sequentially add the variable that best reduces the square loss
- Each method builds a path of solutions from 0 to ordinary least-squares solution

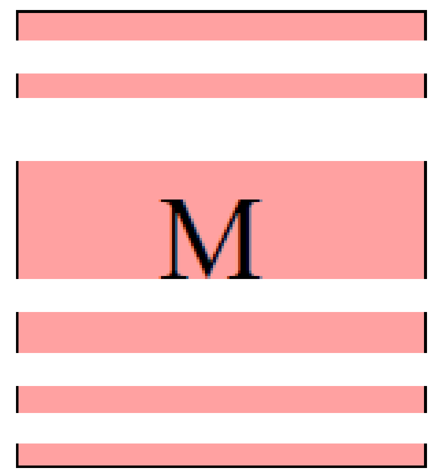
Group Sparsity (Multi-Class)

$$\min_{\mathbf{W},\mathbf{b}} \sum_{i=1}^{n} \left\| \mathbf{W}^T \mathbf{x}_i + \mathbf{b} - \mathbf{y}_i \right\|_2^2 + \lambda \|\mathbf{W}\|_F^2$$



$$\|\mathbf{W}\|_{2,1} = \|\bar{\mathbf{w}}\|_1 = \sum_{i=1}^{m} \sqrt{\sum_{j=1}^{c} W_{ij}^2}.$$

$$\min_{\mathbf{W},\mathbf{t},\mathbf{M}} \ ||\mathbf{X}\mathbf{W} + \mathbf{e}_n \mathbf{t}^T - \mathbf{Y} - \mathbf{B} \odot \mathbf{M}||_{2,1} + \lambda ||\mathbf{W}||_{2,1}$$
$$\text{s.t.} \quad \mathbf{M} \geq \mathbf{0}$$



S. Xiang et al, ***Discriminative Least Squares Regression*** for Multiclass Classification and Feature Selection, IEEE Trans. NNLS, 2012.

Dimension Reduction + Feature Selection

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 39, NO. 12, DECEMBER 2017

Forward Selection Component Analysis: Algorithms and Applications

Luca Puggini, *Student Member, IEEE*, and Seán McLoone

Abstract—Principal Component Analysis (PCA) is a powerful and widely used tool for dimensionality reduction. However, the principal components generated are linear combinations of all the original variables and this often makes interpreting results and root-cause analysis difficult. Forward Selection Component Analysis (FSCA) is a recent technique that overcomes this difficulty by performing variable selection and dimensionality reduction at the same time. This paper provides, for the first time, a detailed presentation of the FSCA algorithm, and introduces a number of new variants of FSCA that incorporate a refinement step to improve performance. We then show different applications of FSCA and compare the performance of the different variants with PCA and Sparse PCA. The results demonstrate the efficacy of FSCA as a low information loss dimensionality reduction and variable selection technique and the improved performance achievable through the inclusion of a refinement step.

应用案例
Special Case on Face Detection

Viola-Jones face detector

ACCEPTED CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION 2001

Rapid Object Detection using a Boosted Cascade of Simple Features

Paul Viola
viola@merl.com
Mitsubishi Electric Research Labs
201 Broadway, 8th FL
Cambridge, MA 02139

Michael Jones
mjones@crl.dec.com
Compaq CRL
One Cambridge Center
Cambridge, MA 02142

P. Viola and M. J. Jones. Robust Real-Time Face Detection. *IJCV* 2004.

Viola-Jones Face Detector: Results

经典物体检测算法框架

- 在图像中通过滑动窗口（多尺度）生成候选
- 对每一个候选，用分类器判别是否待检物体
  - 人工特征提取
  - 分类器学习
- 对候选对象根据分类器score进行筛选

Step-1 两类分类器训练

Given the representation, train a binary classifier

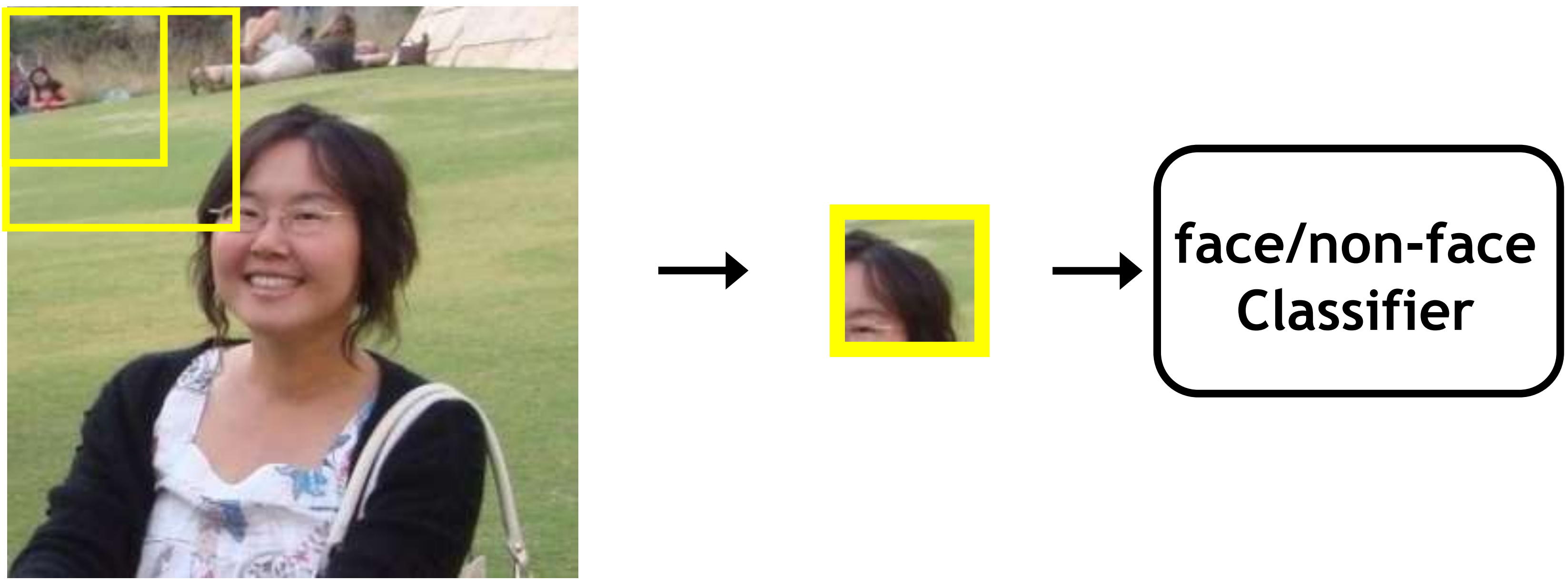
Face/non-face
Classifier

No, not face.

Yes, face.

Step-2 滑窗生成候选

- Scans the detector at multiple **locations** and **scales**



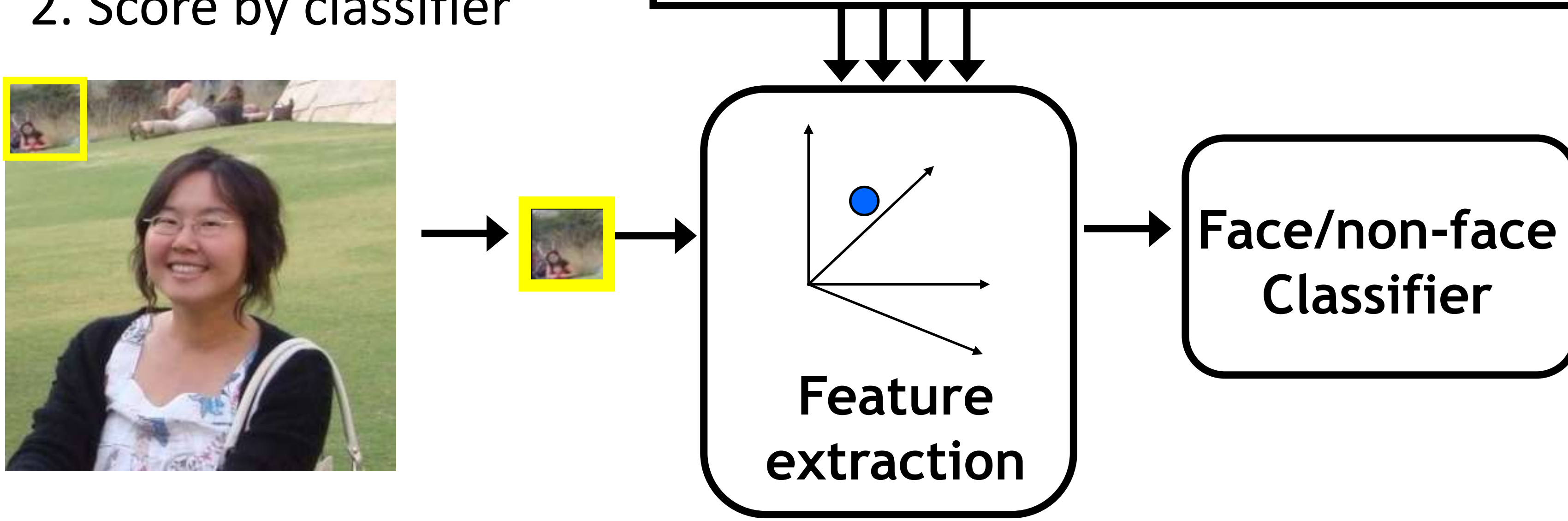
经典物体检测算法

Training:

1. Obtain training data
2. Define features
3. Define classifier

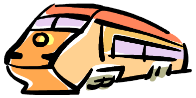
Given new image:

1. Slide window
2. Score by classifier

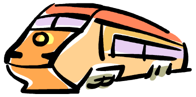


Viola-Jones detection approach

- Viola and Jones’ face detection algorithm
  - The first object detection framework to provide competitive object detection rates ***in real-time***
  - Implemented in OpenCV
- Components
  - Features
    - Haar-features
    - Integral image

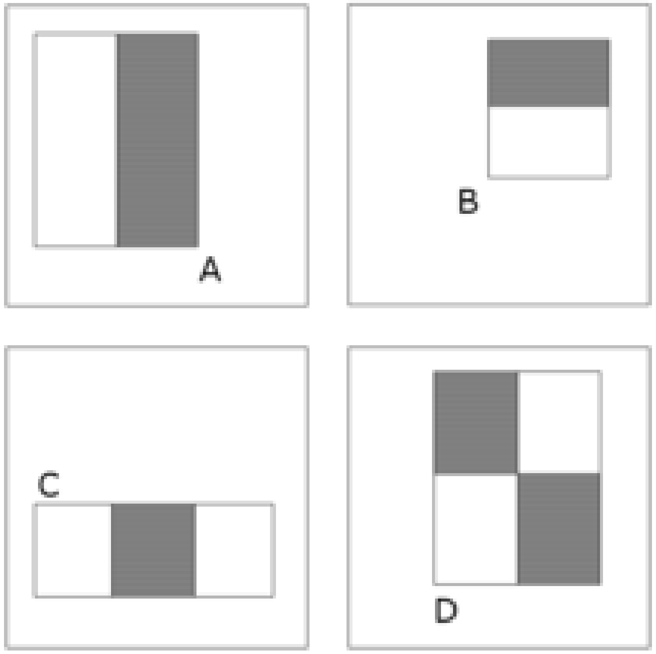


  - Learning
    - Boosting algorithm
  - Cascade method



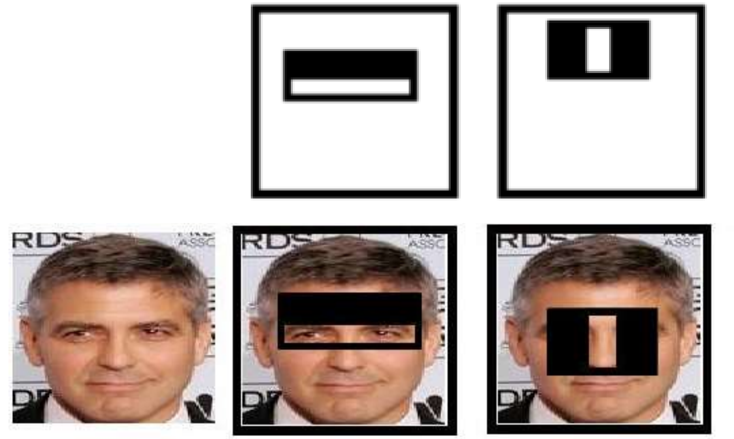
Haar-features

- The difference between pixels' sum of the white and black areas



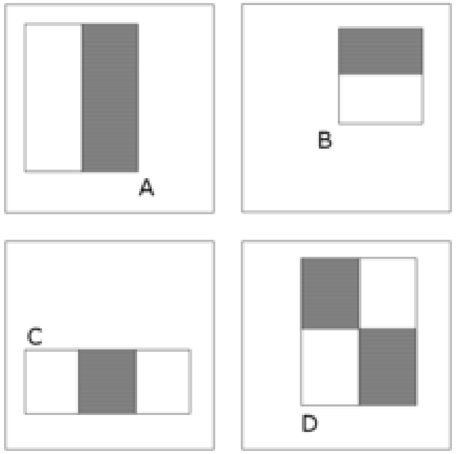
Haar-features

- Capture the face symmetry





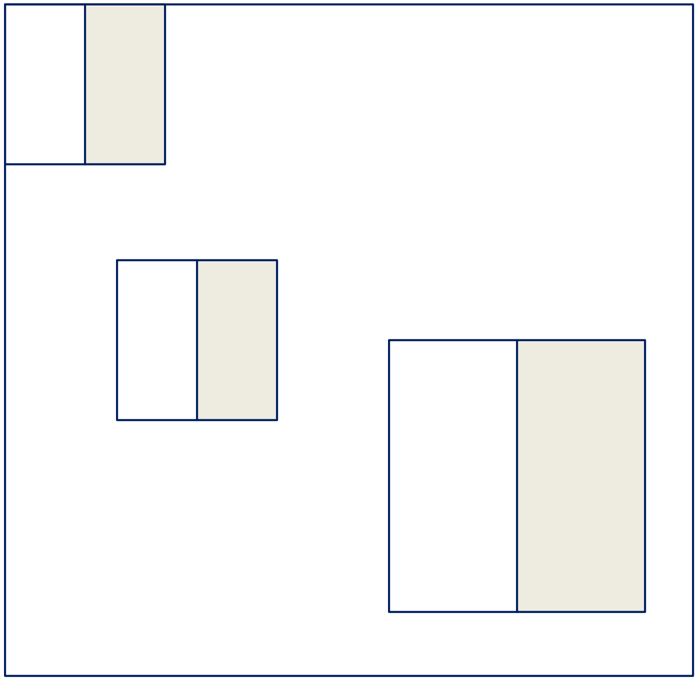
Haar-features



Four types of haar features

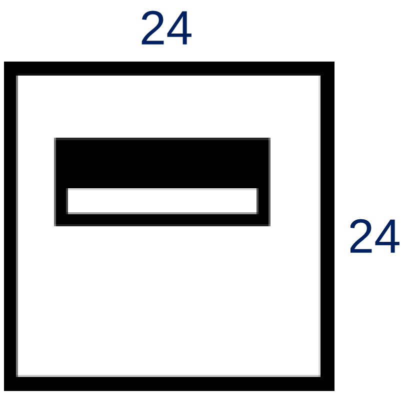
Can be extracted at any location with any scale!

Type A



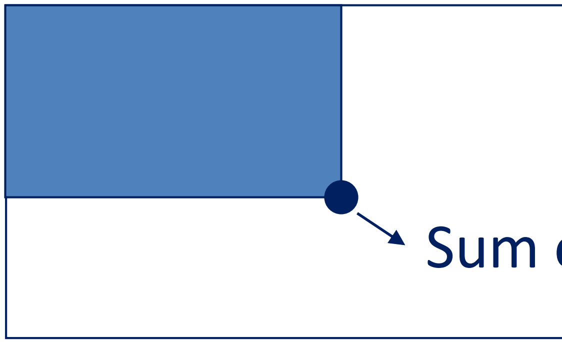
A 24x24 detection window

Haar-features



- Too many features!
  - location, scale, type
  - 180,000+ possible features associated with each 24 x 24 window
- Not all of them are useful!
- Speed-up strategy
  - Fast calculation of haar-features
  - Selection of good features

Integral image



Sum of pixel values in the blue area

Example:

Time complexity?

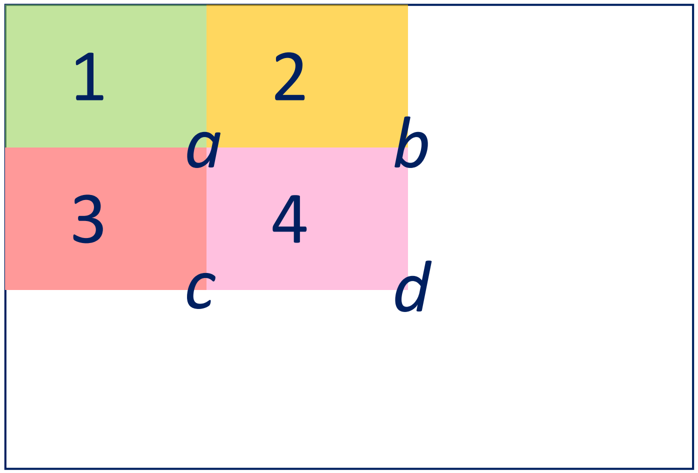
2	1	2	3	4	3
3	2	1	2	2	3
4	2	1	1	1	2

Image

2	3	5	8	12	15
5	8	11	16	22	28
9	14	18	24	31	39

Integral image

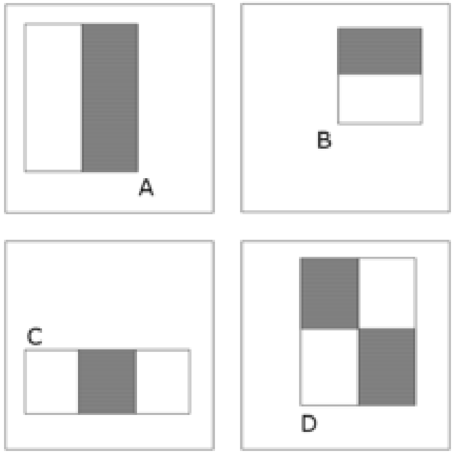
Integral image



a = sum(1)
b = sum(1+2)
c = sum(1+3)
d = sum(1+2+3+4)

Sum(4) = *d* + *a* – *b* – *c*

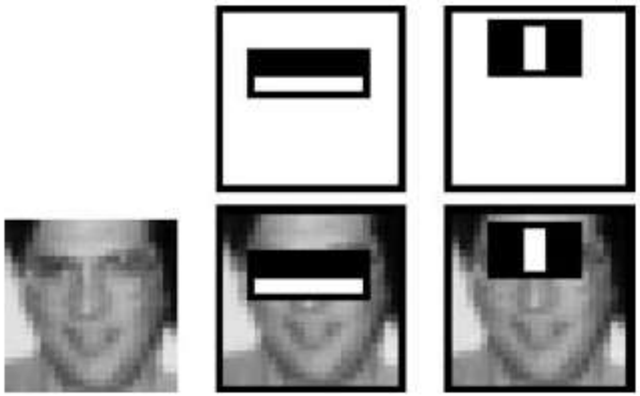
Four-point calculation!



A, B: 2 rectangles => **6-point**
C: 3 rectangles => **8-point**
D: 4 rectangles => **9-point**

特征选择

- A very ***small*** number of features can be combined to from an effective classifier!
- Example: The 1st and 2nd features selected by ***AdaBoost***



特征选择

- A weak classifier *h*

f_1

f_2

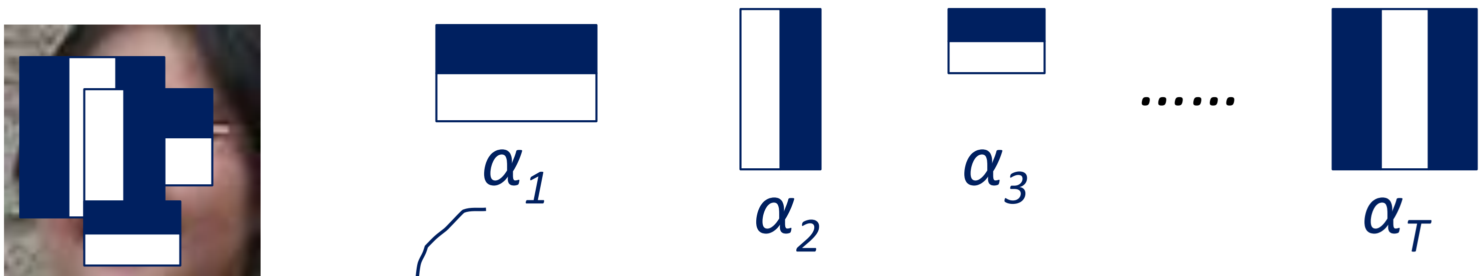
$f_1 > \theta$ (a threshold) => Face!

$f_2 \leq \theta$ (a threshold) => Not a Face!

$$h = \begin{cases} 1 & \text{if } f_i > \theta \\ 0 & \text{otherwise} \end{cases}$$

特征选择

- Idea: Combining several ***weak classifiers*** to generate a ***strong classifier***



$$\alpha_1 h_1 + \alpha_2 h_2 + \alpha_3 h_3 + \ldots + \alpha_T h_T \gtrless T_{thresold}$$

~performance of the weak classifier on the ***training set***

weak classifier (feature, threshold)

$h_1 = 1$ or 0

特征选择

- Training Dataset
  - 4916 face images
  - non-face images cropped from 9500 images

positive samples

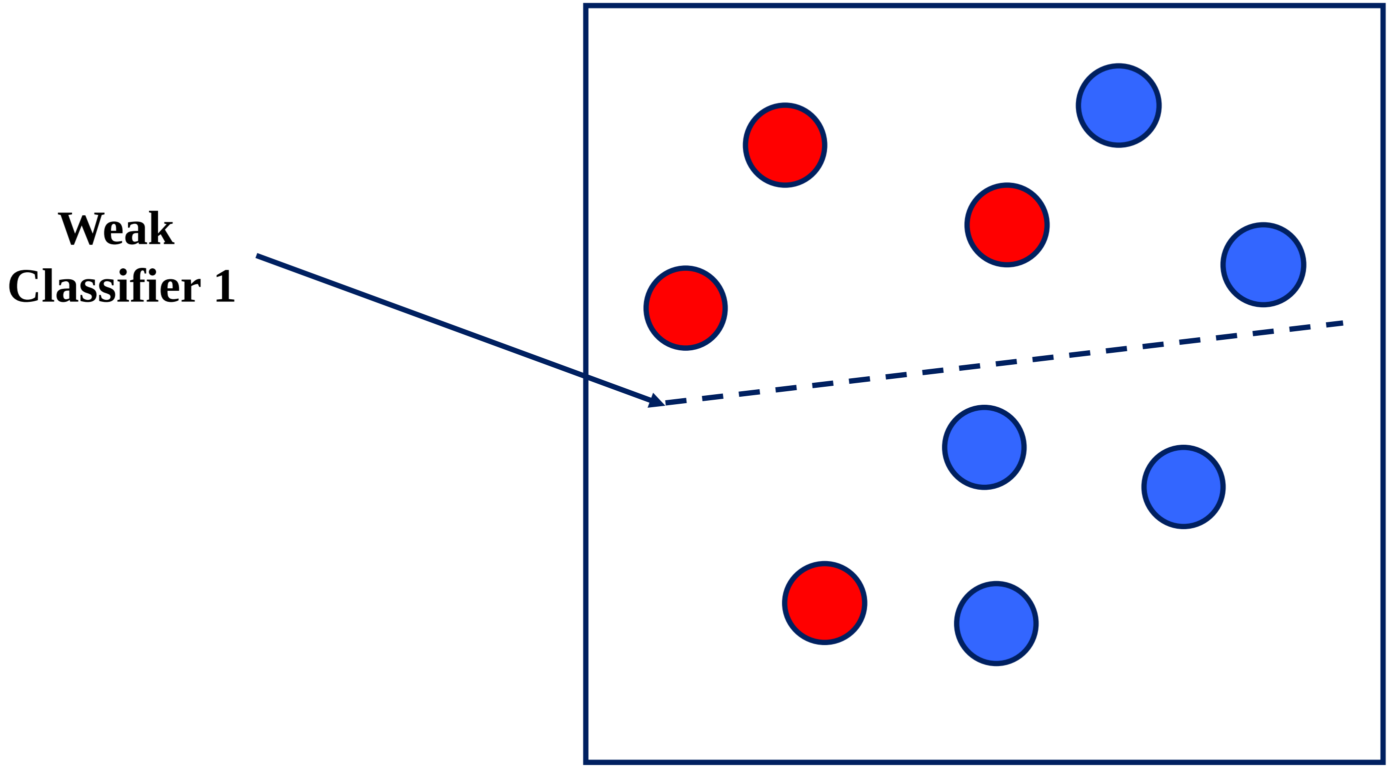
non-face images

negative samples

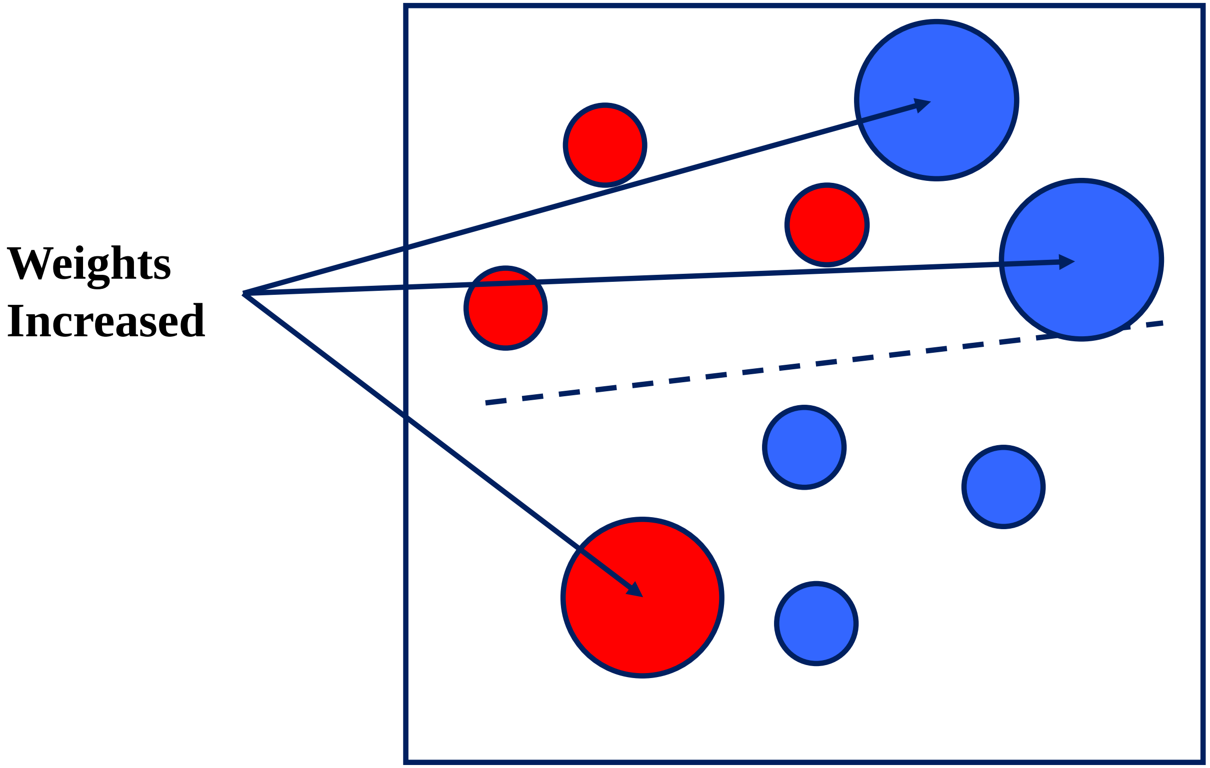
AdaBoost

- Each training sample may have different importance!
- Focuses more on previously ***misclassified*** samples
  - Initially, all samples are assigned ***equal weights***
  - Weights may change at each boosting round
    - **misclassified samples → increase their weights**
    - **correctly classified samples → decrease their weights**

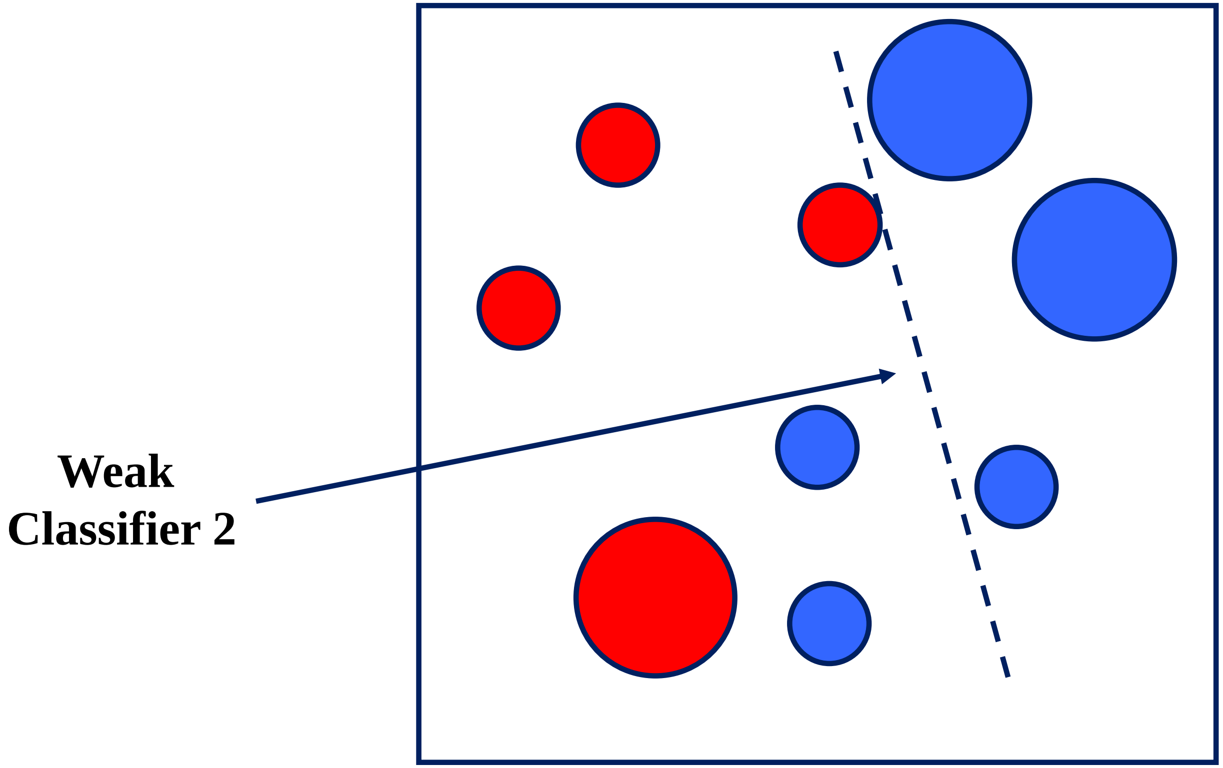
Boosting illustration



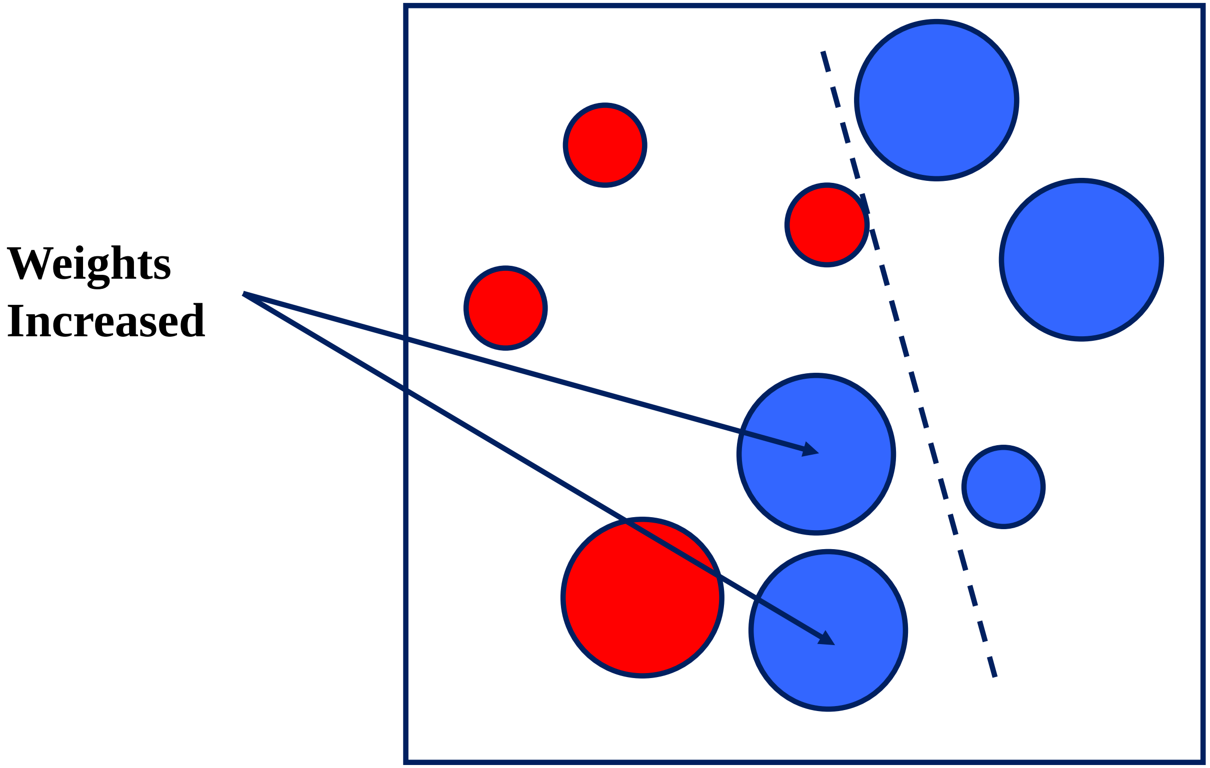
Boosting illustration



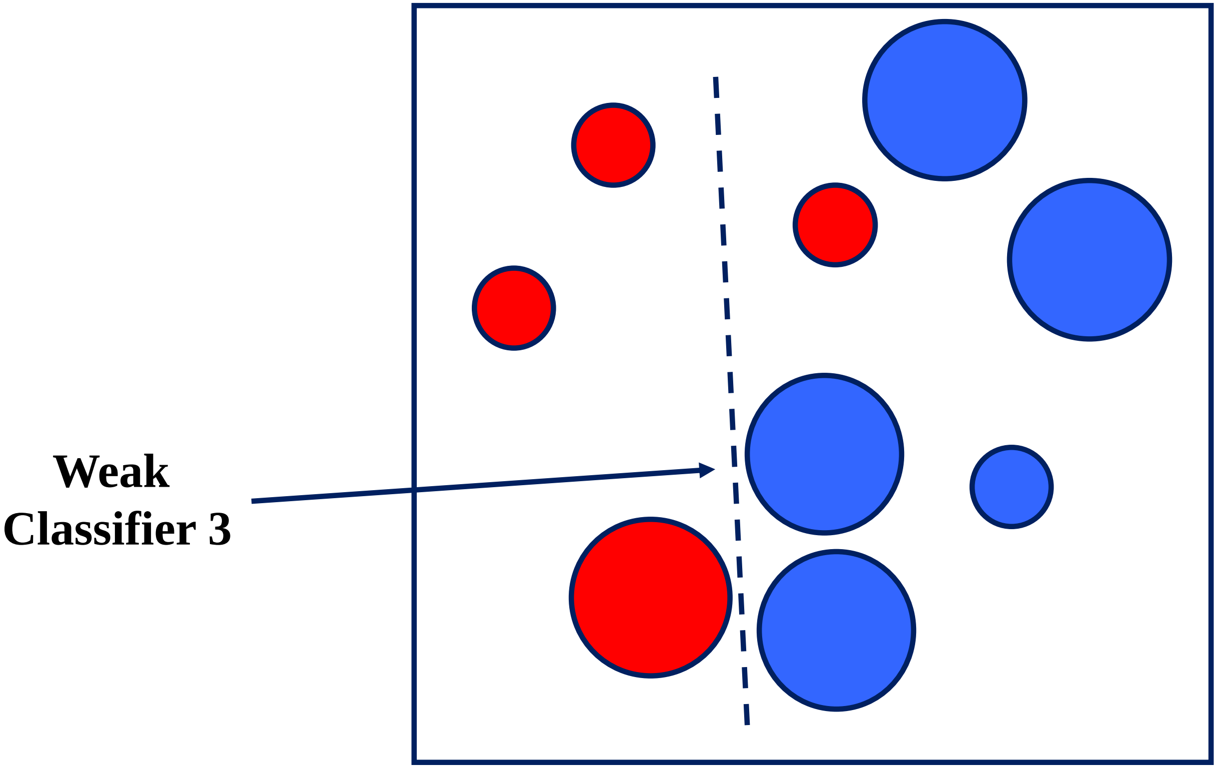
Boosting illustration



Boosting illustration

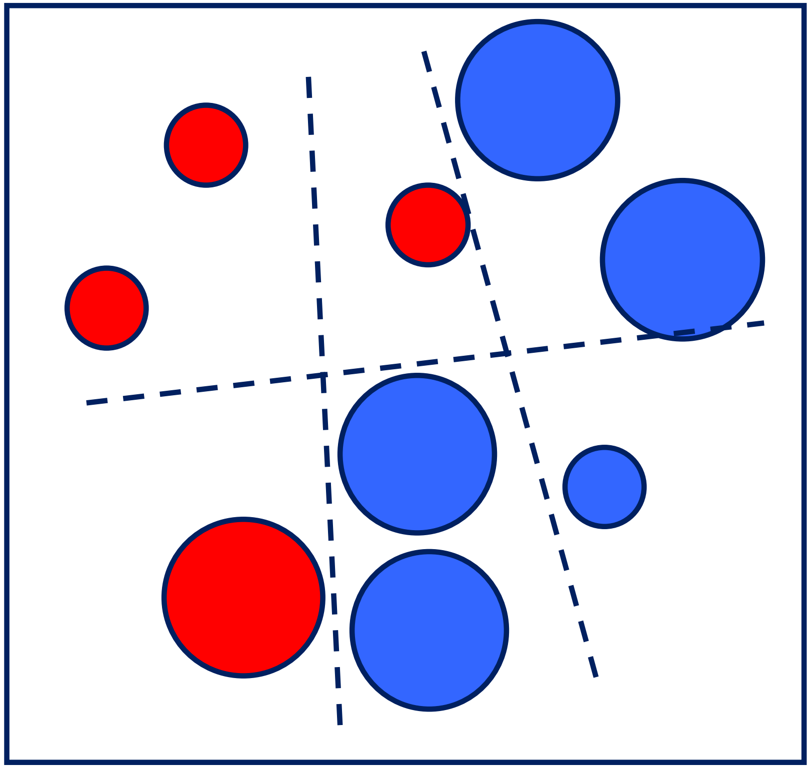


Boosting illustration



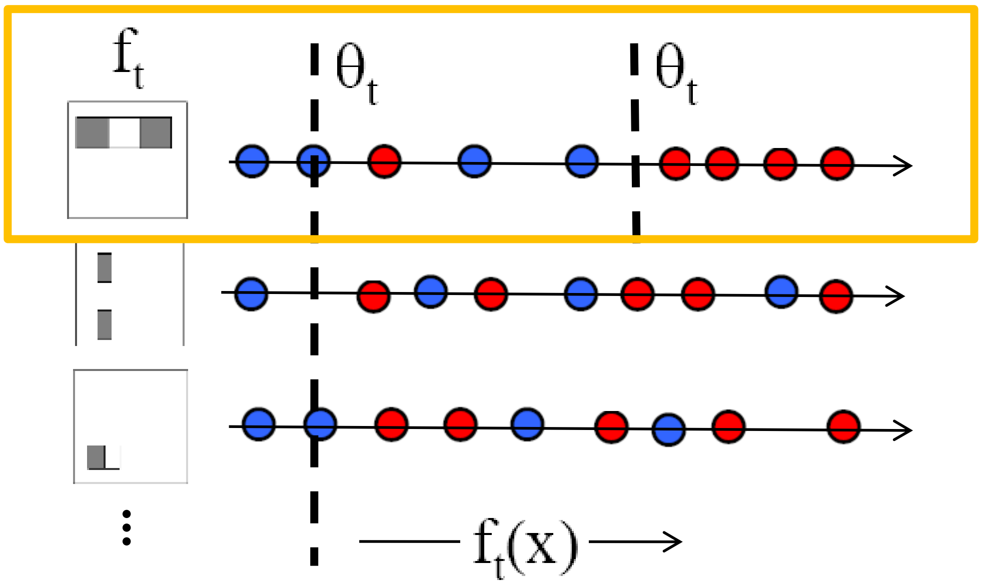
Boosting illustration

Final classifier is a combination of weak classifiers



Viola-Jones detector: AdaBoost

- **Want to select the single rectangle feature and threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of *weighted* error.**



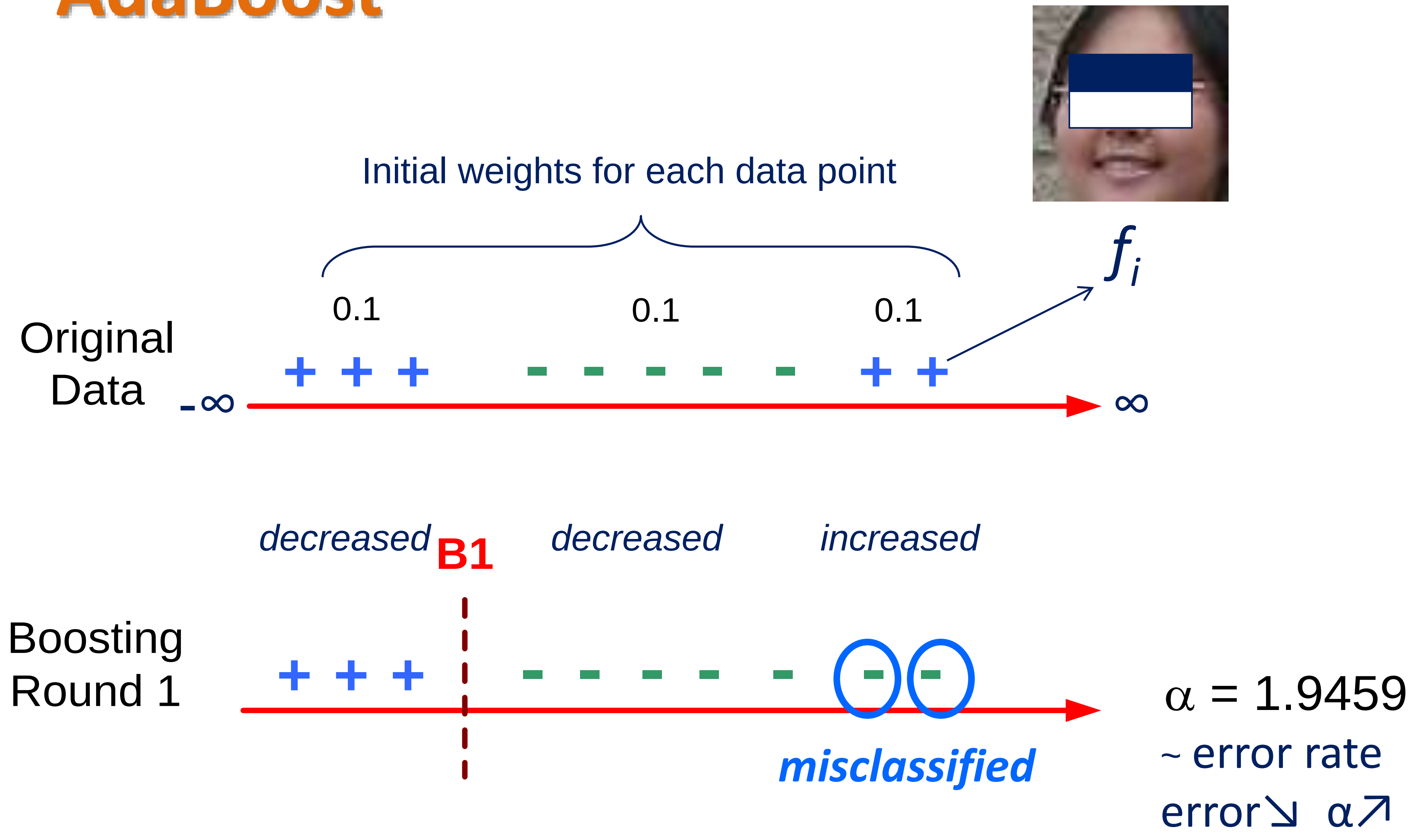
Outputs of a possible rectangle feature on faces and non-faces.

Resulting weak classifier:

$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ -1 & \text{otherwise} \end{cases}$$

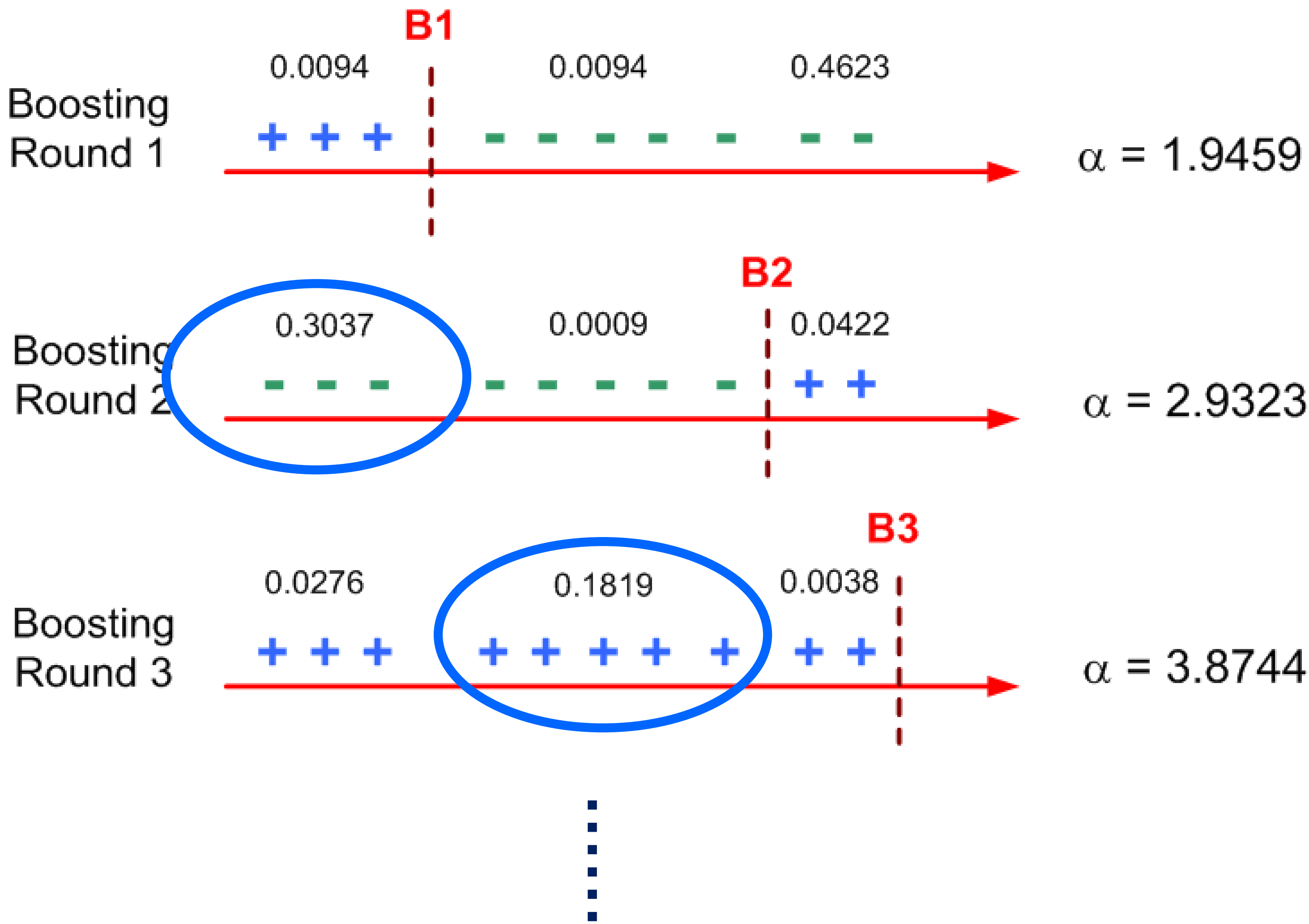
For next round, reweight the examples according to errors, choose another filter/threshold combo.

AdaBoost



Initial weights for each data point
0.1
0.1
0.1
Original Data
-∞
+ + + - - - - - + +
∞
f_i
decreased
B1
decreased
increased
Boosting Round 1
+ + + - - - - - - -
misclassified
α = 1.9459
~ error rate
error↘ α↗

AdaBoost



B1
0.0094
0.0094
0.4623
Boosting Round 1
α = 1.9459
B2
0.3037
0.0009
0.0422
Boosting Round 2
α = 2.9323
B3
0.0276
0.1819
0.0038
Boosting Round 3
α = 3.8744

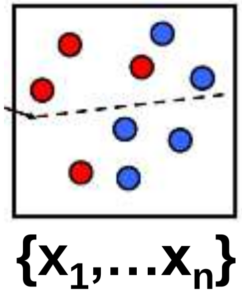
分类器学习

- Initialize equal weights to training samples
- For *T* rounds
  - normalize the weights
  - select the best weak classifier in terms of the weighted error
  - update the weights (raise weights to misclassified samples)
- Linearly combine these *T* weak classifiers to form a strong classifier

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

AdaBoost Algorithm

Start with uniform weights on training examples



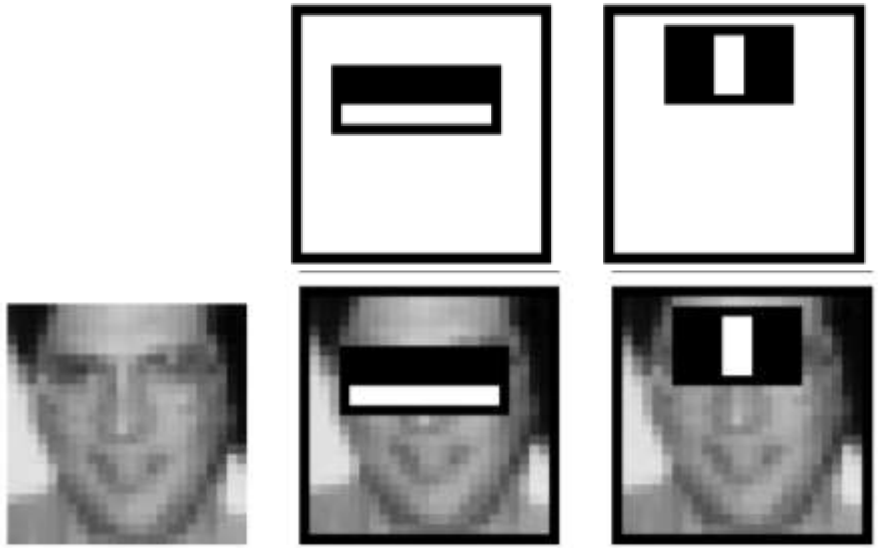
For T rounds

Evaluate *weighted* error for each feature, pick best.

Re-weight the examples:
Incorrectly classified -> more weight
Correctly classified -> less weight

Final classifier is combination of the weak ones, weighted according to error they had.

Feature Selection: Results



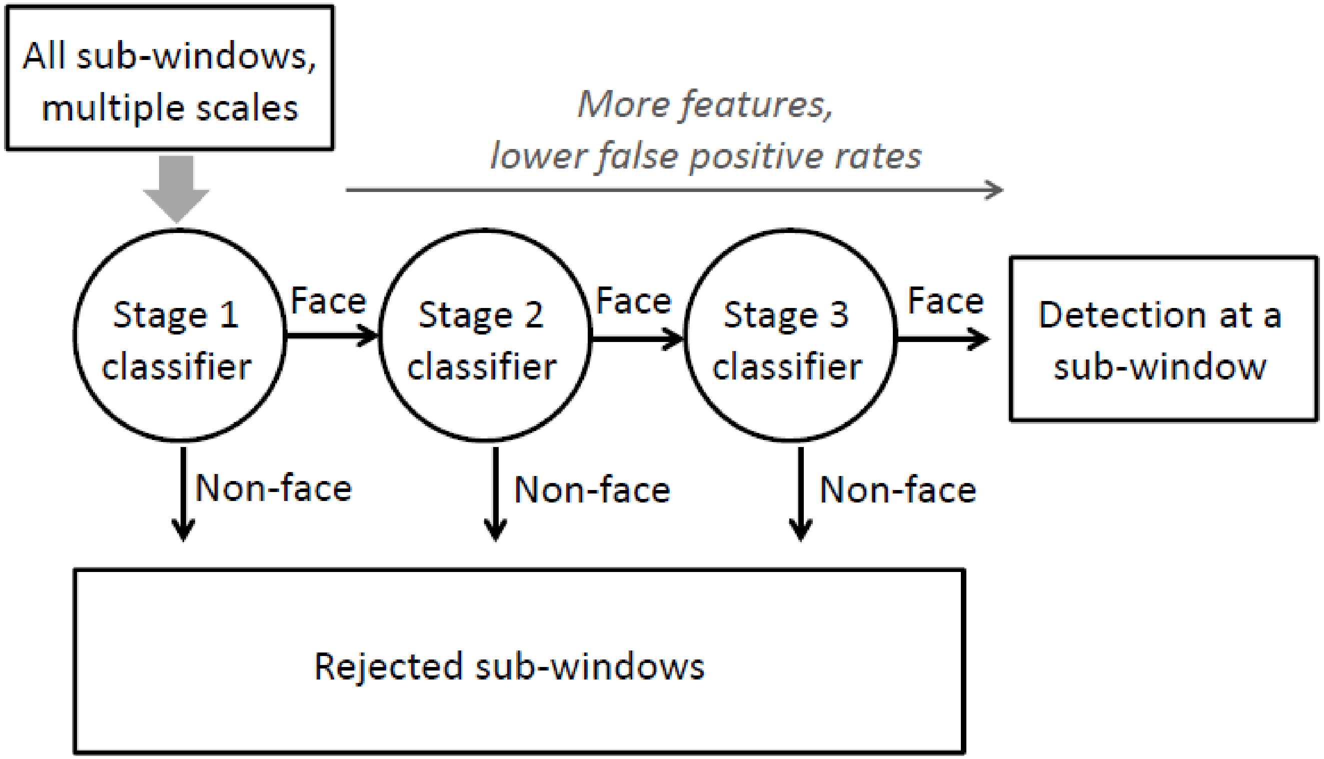
First two features selected

Viola-Jones detection approach

- Viola and Jones’ face detection algorithm
  - The first object detection framework to provide competitive object detection rates ***in real-time***
  - Implemented in OpenCV
- Components
  - Features
    - Haar-features
    - Integral image
  - Learning
    - Boosting algorithm
  - Cascade method
    - Even if the filters are fast to compute, each new image has a lot of possible windows to search.
    - How to make the detection more efficient?

Cascade method

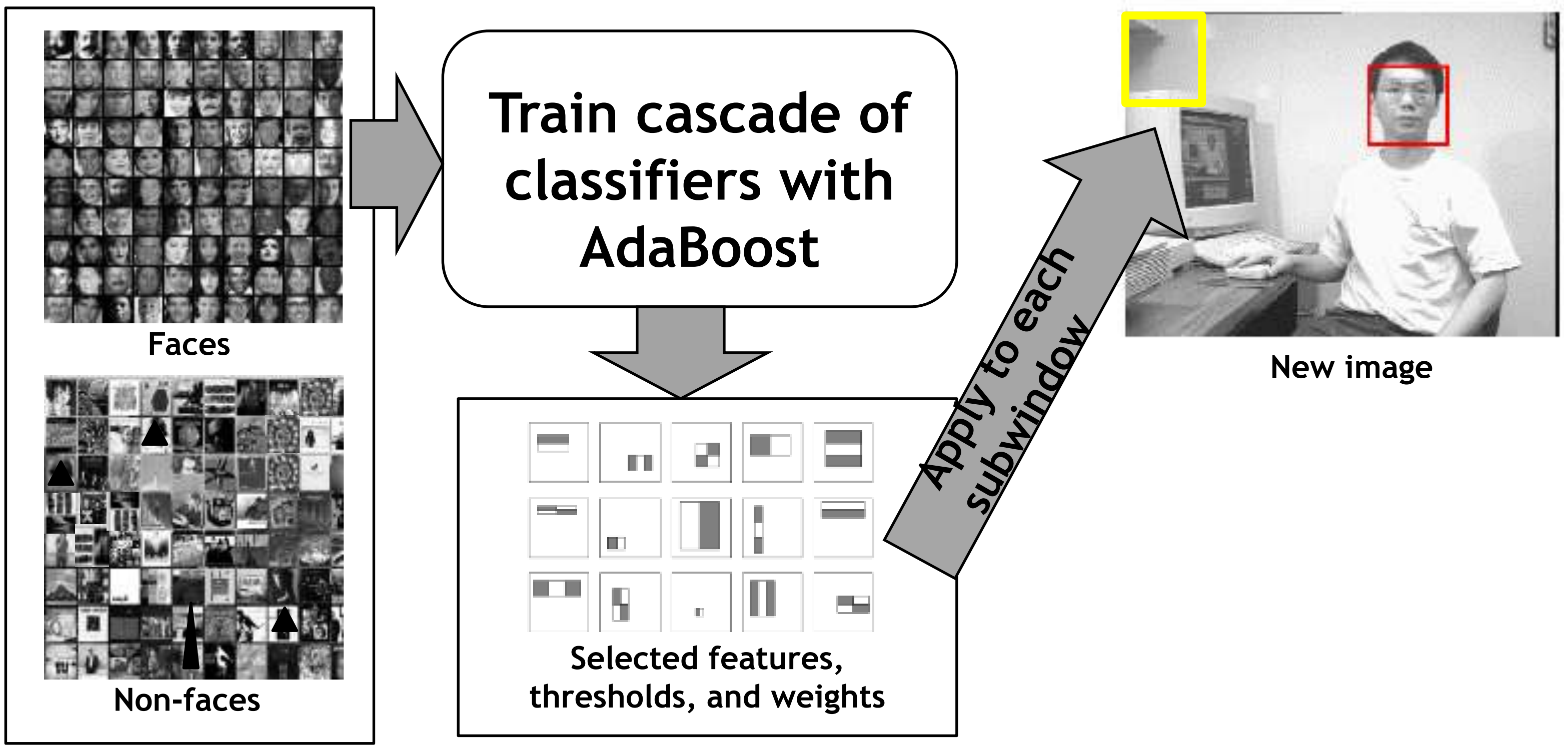
$$\text{Strong Classifier} = \underbrace{(\alpha_1 h_1 + \alpha_2 h_2)}_{1} + \underbrace{(\ldots)}_{2} + \underbrace{(\ldots + \alpha_T h_T)}_{3} \gtrless T_{thresold}$$



Most windows contain no face!

Rejects negative windows in an early stage!

Viola-Jones detector: summary

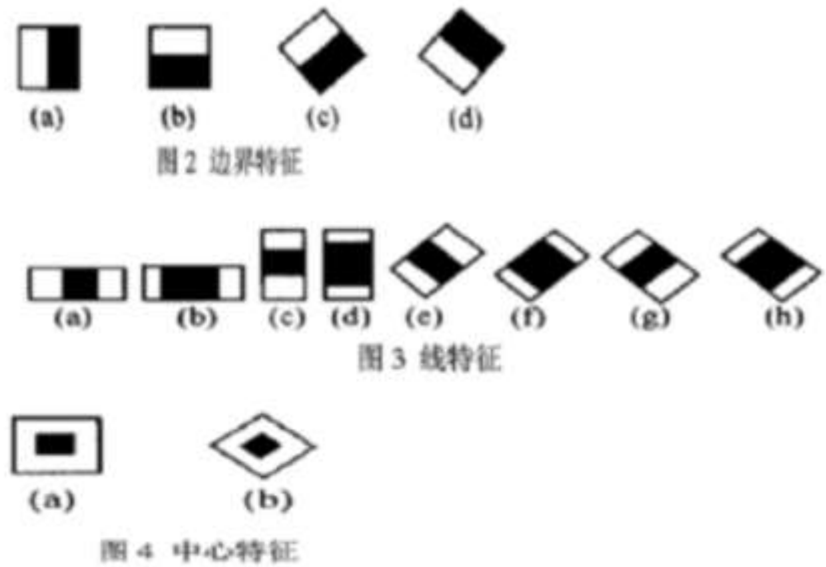


Train with 5K positives, 350M negatives
Real-time detector using 38 layer cascade
6061 features in all layers

Implementation available in OpenCV

经典物体检测算法改进

- Haar特征明暗对比
- Boosting改进
  - vector boosting
  - output code to boost multiclass
  - multiple instance boosting
  - multi-class boosting
  - floatboost learning
  - multi-class adaboost
  - textonboost
- SVM+HOG行人检测



Thank You!
Q&A