UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

MASTER IN COMPUTER SCIENCE

Pattern Recognition

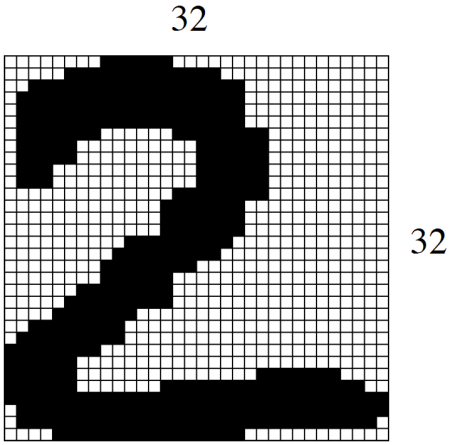
Lecture 6 : Feature Selection & Classifier Combination

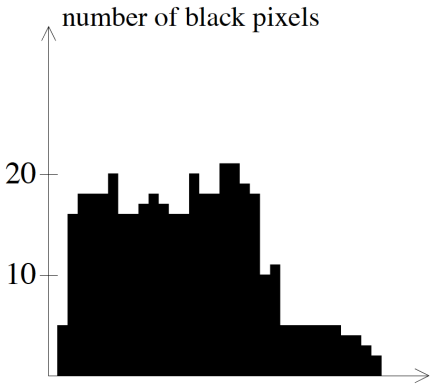
Dr. Andreas Fischer
andreas.fischer@unifr.ch

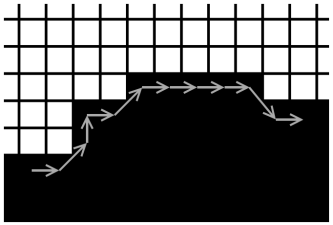
Feature Selection

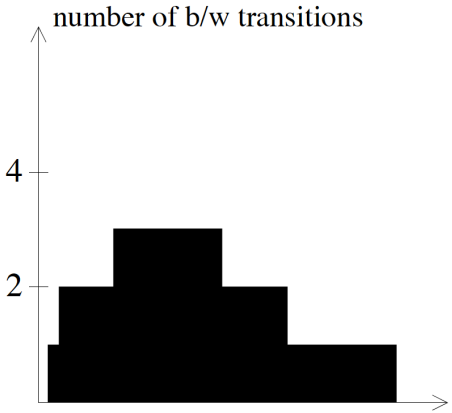
Example: Digit Recognition

- Preprocessing:
  - Binarization
  - Slant correction (shearing operation)
  - Thickness (morphological operations)
  - Size (scale to n x n pixels)
- Pixel features:
  - Raw 0/1 values
- Row/column/diagonal features:
  - Number of black pixels (projection profile)
  - Black/white transitions
  - Margin until first black pixel
- Grid features (k x k cells):
  - Number of black pixels
  - Histogram of contour orientations
- …









Feature Normalization

- Combining different feature domains problematic for distance-based classifiers such as KNN.
- Notation: x_{ij} is the feature j of the learning sample i.
- Linear normalization:
  - *min-max* (set n_{ij}=0 if n_{ij} < 0 and n_{ij} = 1 if n_{ij} > 1 for test samples):

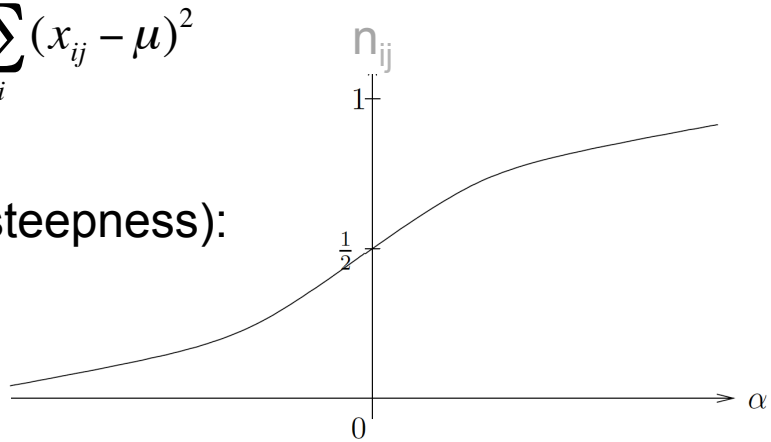
$$n_{ij} = \frac{x_{ij} - \min}{\max - \min}; \ \min = \min\{x_{ij} : 1 \le i \le N\}; \ \max = \max\{x_{ij} : 1 \le i \le N\}$$

  - *z-score*:

$$n_{ij} = \frac{x_{ij} - \mu}{\sigma}; \ \mu = \frac{1}{N}\sum_{i}^{N} x_{ij}; \ \sigma = \sqrt{\frac{1}{N-1}\sum_{i}^{N}(x_{ij} - \mu)^2}$$

- Non-linear normalization:
  - *sigmoid* (parameter c > 0 controls steepness):

$$n_{ij} = \frac{1}{1 + \exp(-c\alpha)}; \ \alpha = \frac{x_{ij} - \mu}{\sigma}$$



Feature Selection

- Goal: Find subset of features to reduce the dimensionality and to improve the classification performance.
- Classification performance expected to improve:
  - When removing irrelevant, noisy, and redundant features.
  - When focusing on independent features that capture complementary properties of the patterns.
- Quality J(F) of a feature set F drives the search procedure:
  - *Filter* method: J measures the class overlap in the feature space.
  - *Wrapper* method: J is the performance of a classifier.

Sequential Forward Search (SFS)

- Bottom-up method that iteratively adds the best remaining feature.

sequential forward search

input: $F = \{x_1, \ldots, x_n\}$; $n' < n$
output: $F' = \{x_{i1}, x_{i2}, \ldots, x_{in'}\} \subseteq F$
begin
    $F' = \emptyset$
    for $i = 1$ to n' **do**
        $j = argmax_{x_k \in F - F'} J(F' \cup \{x_k\})$
        $F' = F' \cup \{x_j\}$
    endfor
end

Dynamic SFS

- Feature selection can be stopped dynamically if the quality J(F') cannot be improved anymore.

sequential forward search with dynamic number of features

input: $F = \{x_1, \ldots, x_n\}$; $n' < n$
output: $F' = \{x_{i1}, x_{i2}, \ldots, x_{in'}\} \subseteq F$
begin
  $F' = \emptyset$
  while true **do**
    $j = argmax_{x_k \in F - F'} J(F' \cup \{x_k\})$
    if $J(F' \cup \{x_j\}) \leq J(F')$
      then output F';
        exit **while**-loop
    $F' = F' \cup \{x_j\}$
  endwhile
end

Sequential Backward Search (SBS)

- Top-down variant that iteratively removes a feature from F.

sequential backward search

input: $F = \{x, \ldots, x_n\}$; $n' < n$
output: $F' = \{x_{i1}, x_{i2}, \ldots, x_{in'}\} \subseteq F$
begin
    $F' = F$
    for $i = 1$ to $n - n'$ **do**
        $j = argmax_{x_k \in F'} J(F' - \{x_k\})$
        $F' = F' - \{x_j\}$
    endfor
end

Sequential Floating Forward Search (SFFS)

- Plus-l minus-r: Add l features, then remove r < l features.
- SFFS: Add one feature, then remove as many features as possible until the quality J(F') cannot be improved anymore.

plus-l minus-r forward search

input: $F = \{x, \ldots, x_n\}$; l; r; $r < l$
output: $F' = \{x_{i1}, x_{i2}, \ldots, x_{in'}\} \subseteq F$
begin
  $F' = \emptyset$
  for $i = 1$ to max **do**
    for $i' = 1$ to l **do**
      $j = argmax_{x_k \in F - F'} J(F' \cup \{x_k\})$
      $F' = F' \cup \{x_j\}$
    endfor
    for $i' = 1$ to r **do**
      $j = argmax_{x_k \in F'} J(F' - \{x_k\})$
      $F' = F' - \{x_j\}$
    endfor
  endfor
end

Genetic Algorithms

Genetic Algorithms (GA)

- Search procedure inspired by evolutionary selection mechanisms, *survival of the fittest*.
- *Chromosomes* $y \in Y$ consisting of *genes* y_i are states in the search space. For feature selection:

y_1	y_2	$\cdots$	y_i	$\cdots$	y_n

$$y_i = \begin{cases} 0 \Leftrightarrow \text{feature i is not used} \\ 1 \Leftrightarrow \text{feature i is used} \end{cases}$$

- The *fitness function* J(y) measures the quality of the chromosome y. For feature selection this could be the cross-validation performance of a classifier (wrapper method) based on the selected features.

Population

- A *population* $P \subseteq Y$ is a finite set of chromosomes, providing a set of solutions. Goal is to improve from one population to the next.
- Two methods to derive a population P(t+1) from a population P(t):
  - *Mutation*: Randomly change a gene. For feature selection, change 0 to 1 and 1 to 0.
  - *Cross-Over*: Combine two chromosomes to obtain two new chromosomes.
    - Single-point cross-over:

      x = (0,0,1,0,1,1,1,0) ⇒ u = (0,0,1,1,1,0,0,1)
      y = (1,0,1,1,1,0,0,1) ⇒ v = (1,0,1,0,1,1,1,0)

    - Double-point cross-over:

      x = (0,0,1,0,1,1,1,0) ⇒ u = (0,0,1,1,1,1,1,0)
      y = (1,0,1,1,1,0,0,1) ⇒ v = (1,0,1,0,1,0,0,1)

Algorithm

```
generate initial population P(0)
t = 0
while termination criterion not fullfilled do
    select parent subset R ⊆ P(t)
    generate new children population P(t+1) based on R
    t = t + 1
end while
```

- Initialization: typically random genes.
- Termination criterion: fixed number of iterations, small change of average or maximum fitness.
- Parent selection:
  - Deterministic: n-best chromosomes.
  - Stochastic: selection with probability $p(y) = \frac{J(y)}{\sum_{z \in P(t)} J(z)}$
  - Two-stage selection with competition: stage 1 randomly selects chromosomes, stage 2 selects the best among them. The two stages are repeated until R has the desired size.

New Population

```
P(t+1) = ∅
repeat
    randomly select two chromosomes x and y from R
    generate u and v from x and y by cross-over
    Generate w from u and z from v by mutation
    P(t+1) = P(t+1) ∪ {w, z}
until P(t+1) has the desired size
```

- Apply cross-over with probability p_c (typically a high probability). Chromosomes u and v are identical to x and y with probability $(1-p_c)$.
- Apply mutation with probability p_m (typically a low probability). Chromosomes w and z are identical to u and v with probability $(1-p_m)$.
- Elite principle: variant that ensures that the n-best chromosomes are added without change to P(t+1).
- Note that by means of random changes, it is possible to avoid local optima in the search space.

Parameters

- Size of the population.
- Probabilities p_c and p_m.
- Specific choice of methods for initialization, selection, and generation.
- Reasonable defaults that have proven successful for several applications:
  - $(|P(0)|, p_c, p_m) = (100, 0.6, 0.001)$
  - $(|P(0)|, p_c, p_m) = (30, 0.9, 0.01)$

Feature Transformation

Principal Component Analysis (PCA)

- Feature space transformation method that removes linear correlations among the features and reduces the number of features.
- Step 1: Center the features around the origin by subtracting the mean vector m.

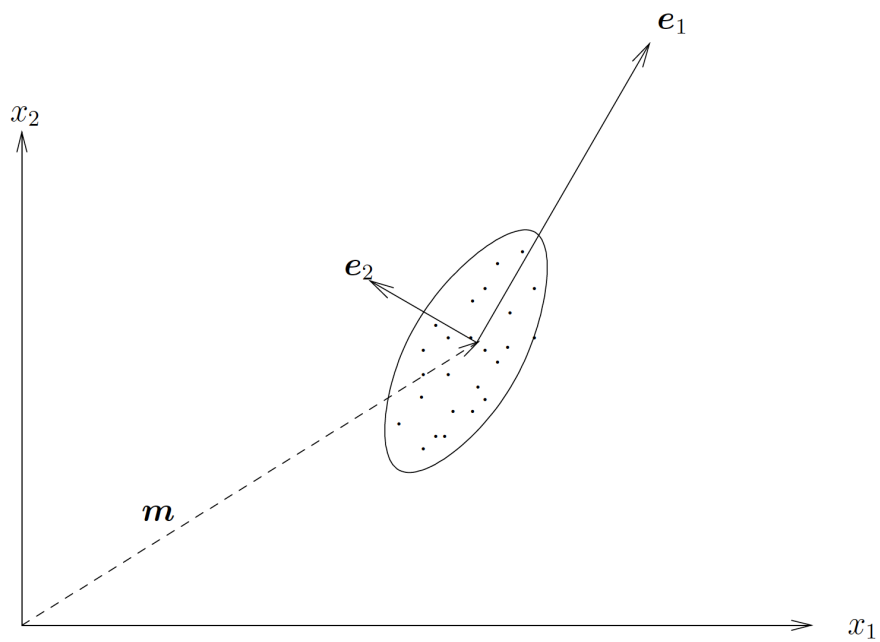
$$\hat{x} = x - m$$

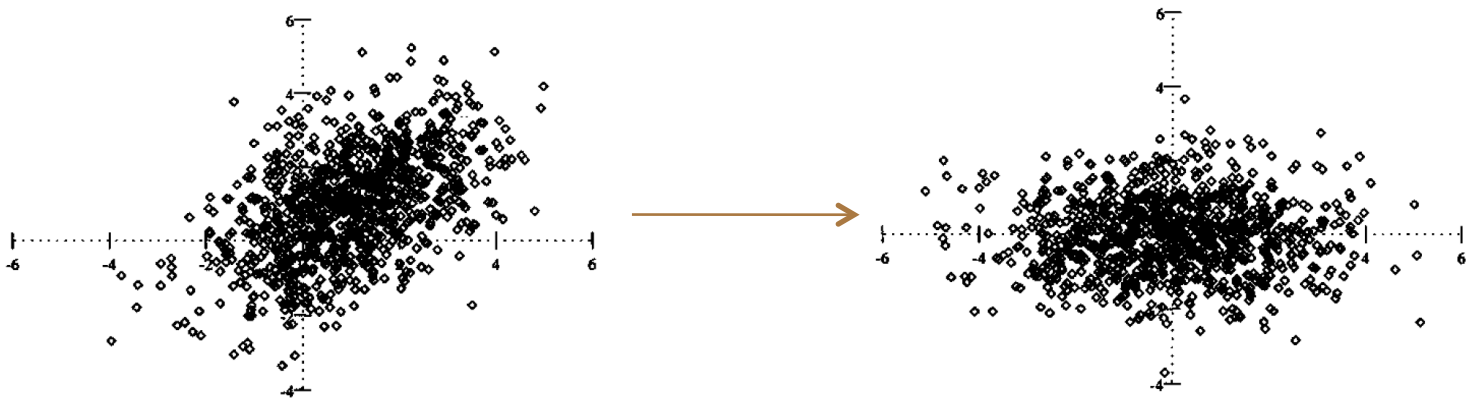
- Step 2: Compute the eigenvectors $e_1,\ldots,e_n$ of the covariance matrix (see lecture 2) and order them according to their eigenvalues $\lambda_1 \geq \ldots \geq \lambda_n$.
- Step 3: Apply principal axis transform to obtain a diagonal covariance matrix in the new feature space Y.

$$y = \begin{bmatrix} e_1' \\ \vdots \\ e_n' \end{bmatrix} \hat{x}$$

  - The transform is a rotation since the eigenvectors are orthogonal.
  - The new coordinate system is spanned by the eigenvectors.
  - First eigenvector points in the direction with maximum variance.
- Step 4: Select k first features to reduce dimensionality while keeping most of the variance.

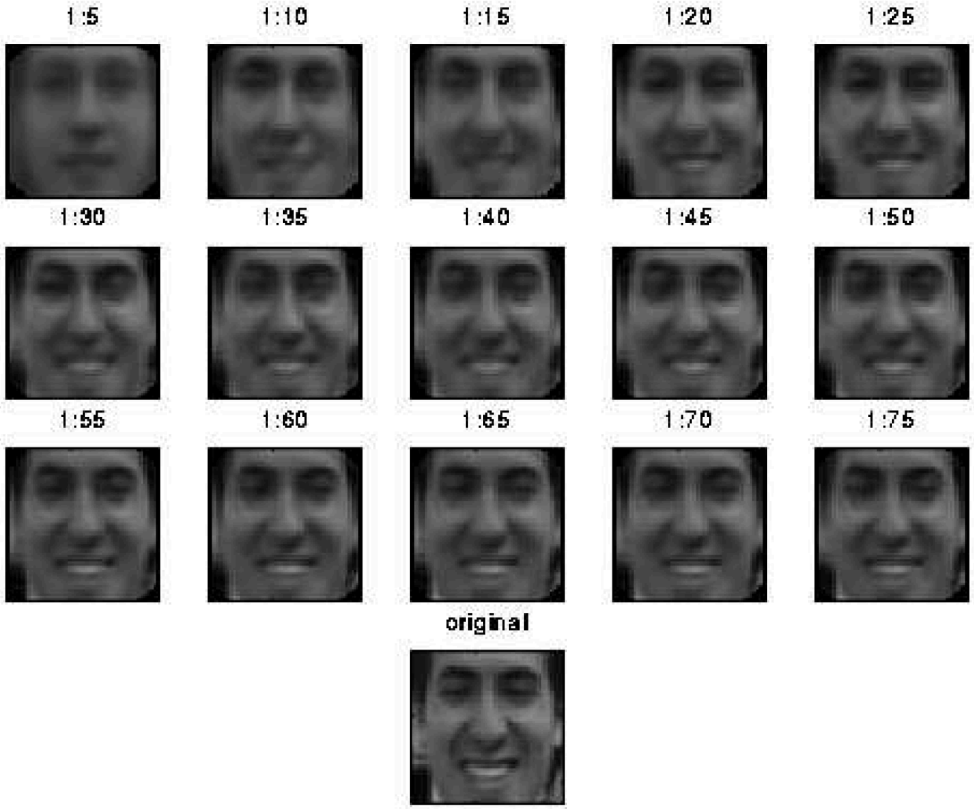
Example





Example: Face Recognition

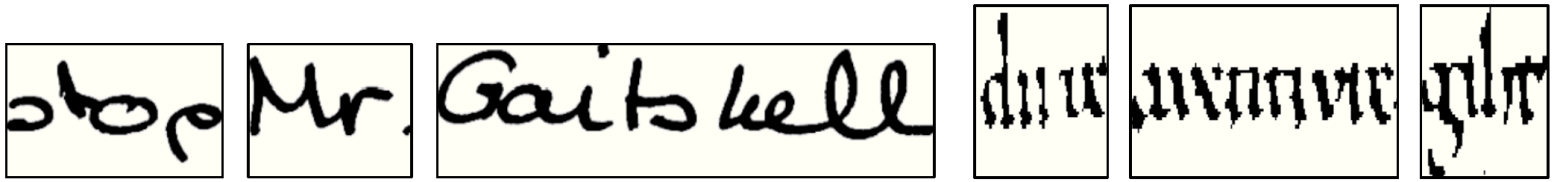
- Reconstruct faces by means of *eigenfaces*.



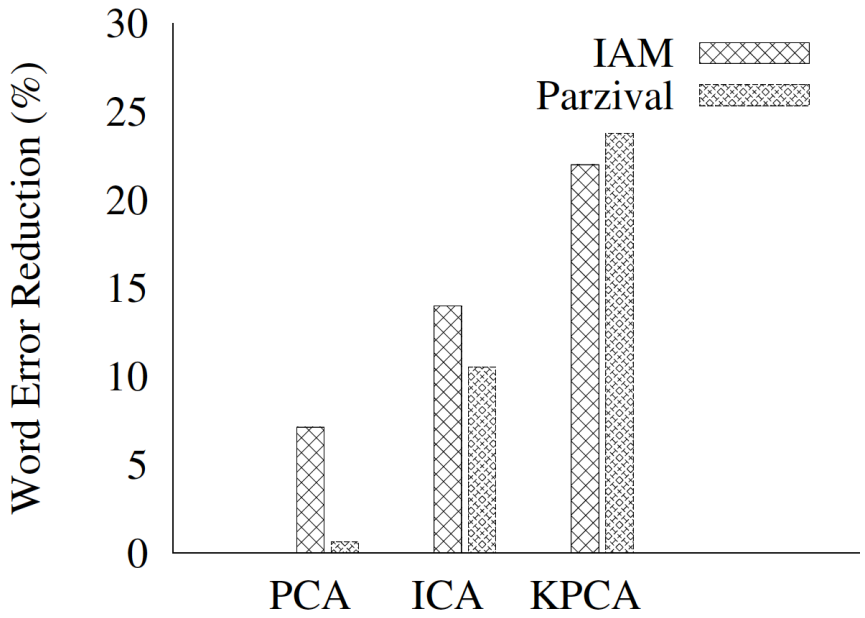
Other Feature Transformations

- Independent component analysis (ICA) aims to improve the statistical independence among the features.
  - E. Oja. Independent component analysis: algorithms and applications. Neural Networks 13: 411-430, 2000.
- PCA is kernelizable (see lecture 4), hence kernel PCA can be applied.
  - B. Schölkopf, A. Smola, and K.R. Müller. Kernel principal component analysis. In: Advances in Kernel Method – Support Vector Learning. MIT Press, pages 327-352, 1999.

Example: Handwriting Recognition



(a) IAM (b) Parzival



Example: Writer Identification

- 100 features extracted from text lines, including slant, ascender and descender heights, width, size and form of convex hulls, …

Experiment	N. of Features	Writer Id. Rate
Baseline	100	92.08%
SBS	42	94.26%
SFS	51	92.35%
SFBS	42	93.17%
SFFS	55	93.44%
GA	50	95.08%
PCA	59	92.35%

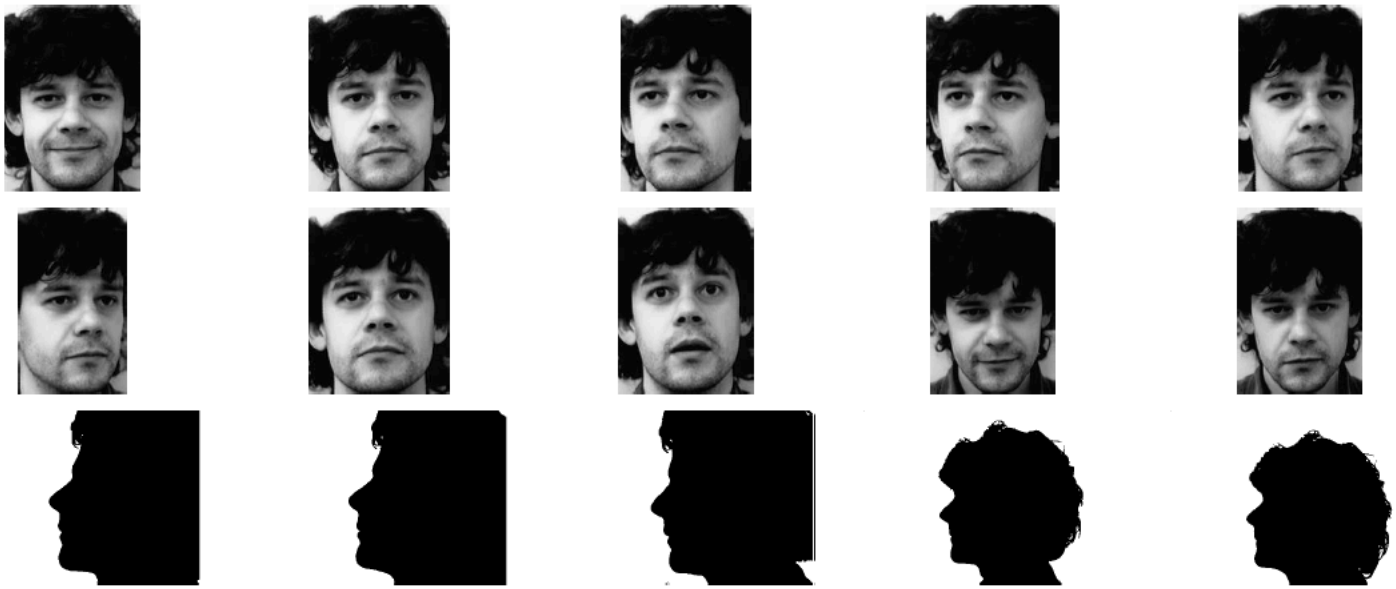
Classifier Combination

Multiple Classifier Systems

- Idea: consult several experts. Ideally, they do not commit the same mistakes and can help each other.
- Two levels of combination.
  - *Early fusion:*
    - Combine sensor data, e.g. color image and depth image.
    - Combine features, e.g. by concatenation $(x_1,\ldots x_n,x_{n+1},x_{n+m})$.
  - *Late fusion:*
    - Combine classifier output. Each classifier is trained independently with its own data and features.

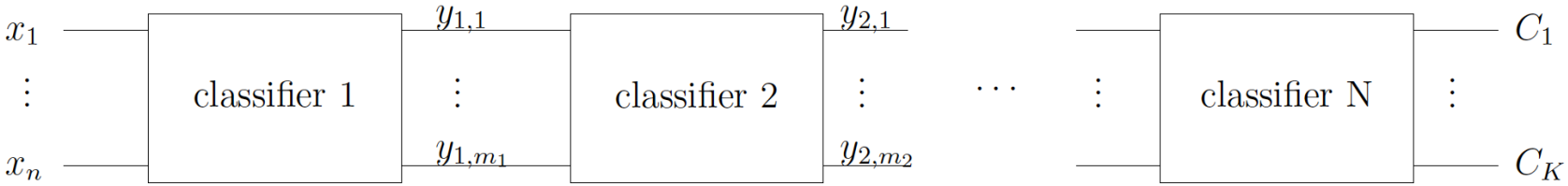
Example: Person Identification

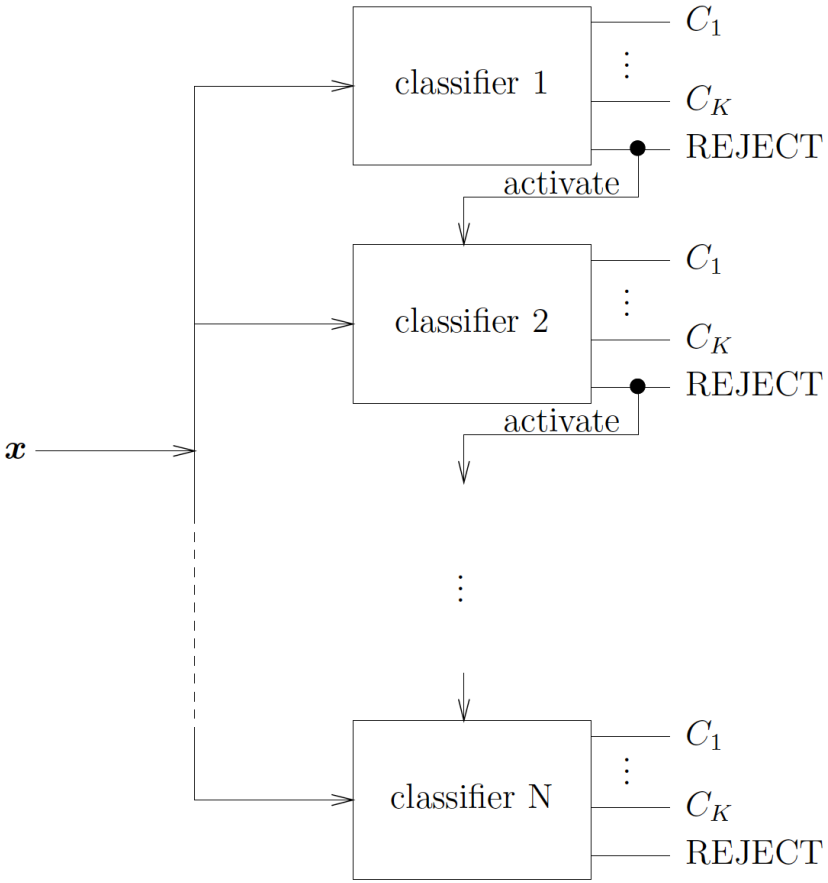
- Combining frontal and profile views.
- Reliability can be further improved by combining with other biometric traits such as voice.



Serial Combination

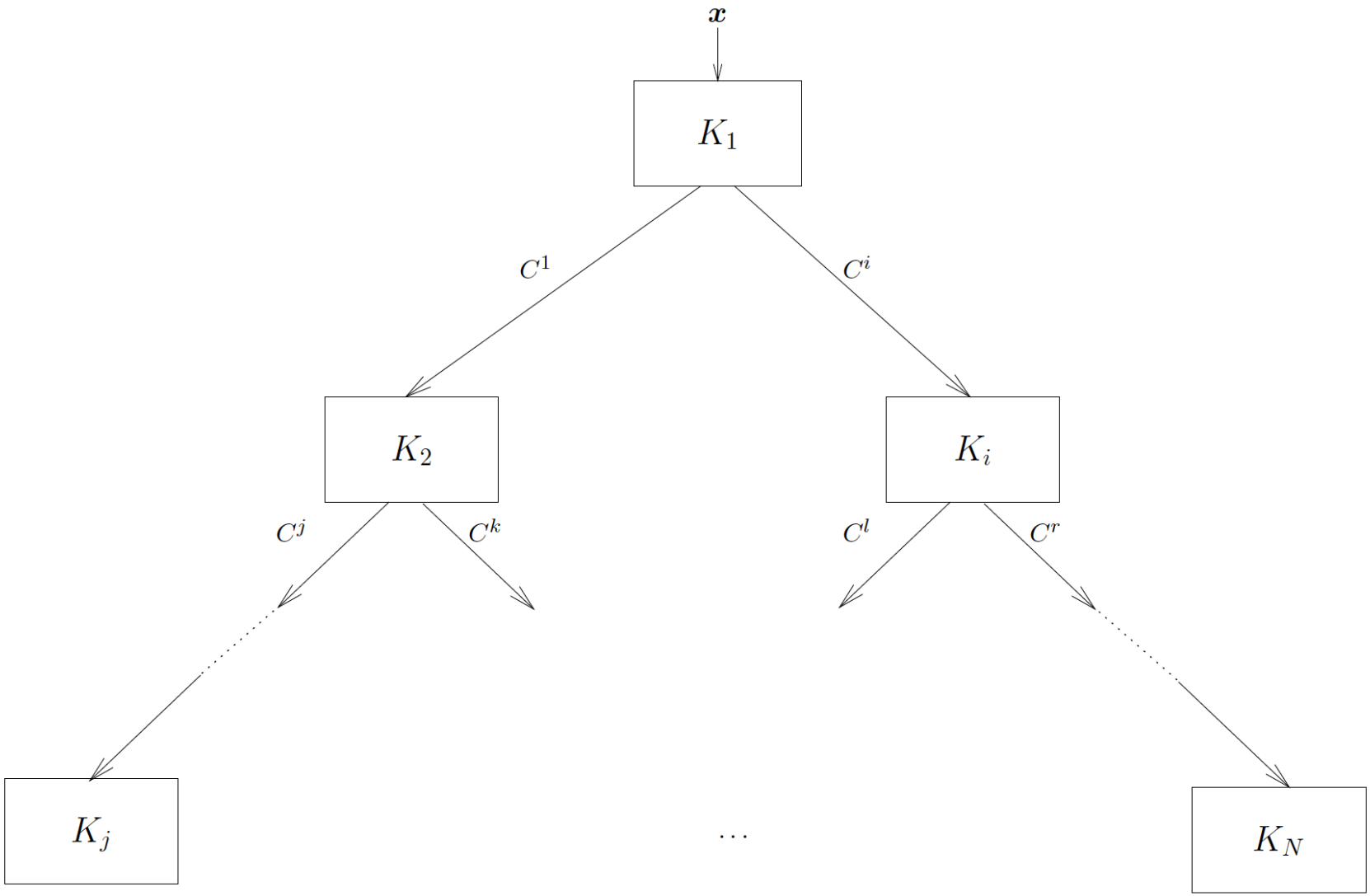
- Classifiers 1,…,N-1 can be interpreted as feature transformations.



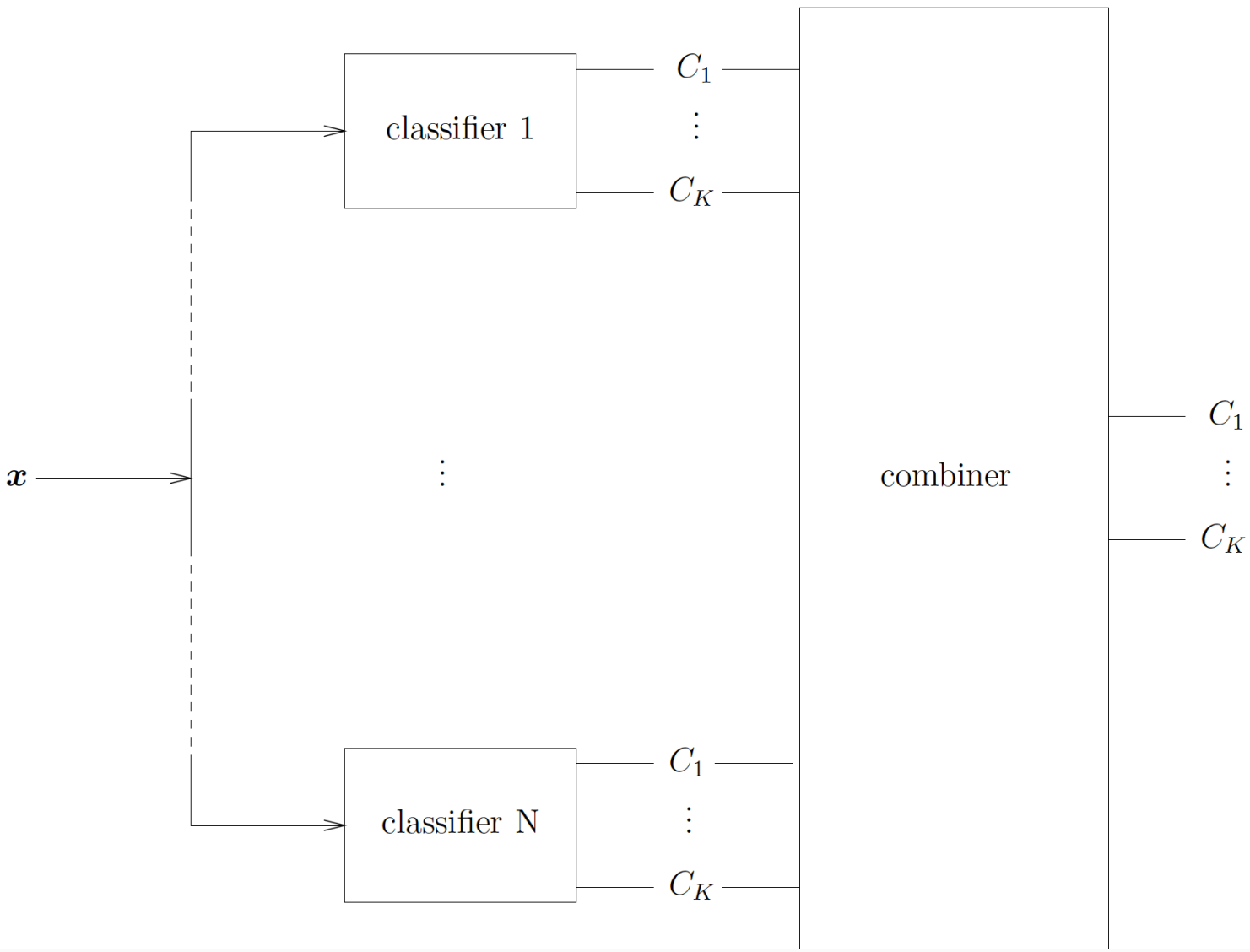


Hierarchical Combination

- Often used for large number of classes $C=\{C_1,\ldots,C_K\}$, $C^i \subset C$.



Parallel Combination



Classifier Output

- Possible combiners for parallel combination depend on the output of the individual classifiers.
  - *Type-1* Classifiers: output class label (e.g. 1NN).
  - *Type-2* Classifiers: output ranked list of classes.
  - *Type-3* Classifiers: output plausibility value $p(C_i)$ for each class, for example posterior probability but also MLP output.
- Clearly, type-3 output can also be reduced to type-2 (ranking by plausibility), and type-2 output can be reduced to type-1 (top rank).

Example

- A possibility to transform type-1 output into type-3 is to compute a confusion matrix on the training set (class i classified as j).

	C_1	C_2	C_3
C_1	8	1	1
C_2	3	7	0
C_3	1	3	6

- C_1: $p(C_1|\boldsymbol{x}) = 8/12$; $p(C_2|\boldsymbol{x}) = 3/12$; $p(C_3|\boldsymbol{x}) = 1/12$;
- C_2: $p(C_1|\boldsymbol{x}) = 1/11$; $p(C_2|\boldsymbol{x}) = 7/11$; $p(C_3|\boldsymbol{x}) = 3/11$;
- C_3: $p(C_1|\boldsymbol{x}) = 1/7$; $p(C_2|\boldsymbol{x}) = 0$; $p(C_3|\boldsymbol{x}) = 6/7$;

Type-1 Combination

- *Consensus voting:* Choose class with most votes.
  - Tie resolution needed.
- *Majority voting:* Choose class with more than N/2 votes, otherwise reject.
  - Note that if classifiers are independent, have an error rate smaller than pure chance (K-1)/K, and $N \rightarrow \infty$, it can be shown that the error rate of the multiple classifier system approaches 0.
- *Weighted voting:* Each vote is weighted, for example with the accuracy of the classifier achieved on the training set. Choose class with maximum sum of weights.
  - Weights can also be learned, for example with an MLP.

K_1 (w_1=0.9): C_1
K_2 (w_2=0.3): C_2
K_3 (w_3=0.8): C_3
K_4 (w_4=0.4): C_2

$\Rightarrow$

C_1: 0.9
C_2: 0.7
C_3: 0.8

Type-2 Combination

- *Maximum rank:* for each class, compute the best rank among all classifier results. Choose class with the best maximum rank.
  - Tie resolution needed.

K_1: (C_3,C_1,C_4,C_2)
K_2: (C_1,C_4,C_3,C_2)
K_3: (C_3,C_4,C_2,C_1)
K_4: (C_4,C_3,C_2,C_1)

$\Rightarrow$

C_1: 1
C_2: 3
C_3: 1
C_4: 1

- *Borda count:* for each class C_i and classifier K_j, compute the number of classes with lower rank $B_j(C_i)$. Choose class with the maximum sum:

$$B(C_i) = \sum_{j=1}^{N} B_j(C_i)$$

Equivalently, sum up ranks over all classifiers. Choose class with the best rank sum.

Borda count:
C_1: 2+3+0+0=5
C_2: 0+0+1+1=2
C_3: 3+1+3+2=9
C_4: 1+2+2+3=8

Rank sum:
C_1: 2+1+4+4=11
C_2: 4+4+3+3=14
C_3: 1+3+1+2=7
C_4: 3+2+2+1=8

Type-3 Combination

- Combined plausibility $p(C_i)=f(p_1(C_i),\ldots,p_N(C_i))$:
  - Maximum (f=max): optimistic estimate of $p(C_i)$.
  - Minimum (f=min): conservative estimate of $p(C_i)$.
  - Mean:
    $$p(C_i) = \frac{1}{N}\sum_{j=1}^{N} p_j(C_i)$$
  - Weighted sum:
    $$p(C_i) = \sum_{j=1}^{N} w_j p_j(C_i)$$
  - Product: assumes probability output and statistical independence, also known as *Bayes' combination rule*:
    $$p(C_i) = \prod_{j=1}^{N} p_j(C_i)$$
- Plausibility values need to be normalized when combining different classifiers, e.g. Bayes posteriors and MLP outputs.
- Function f can also be replaced by a classifier, which receives the plausibility values as input features.

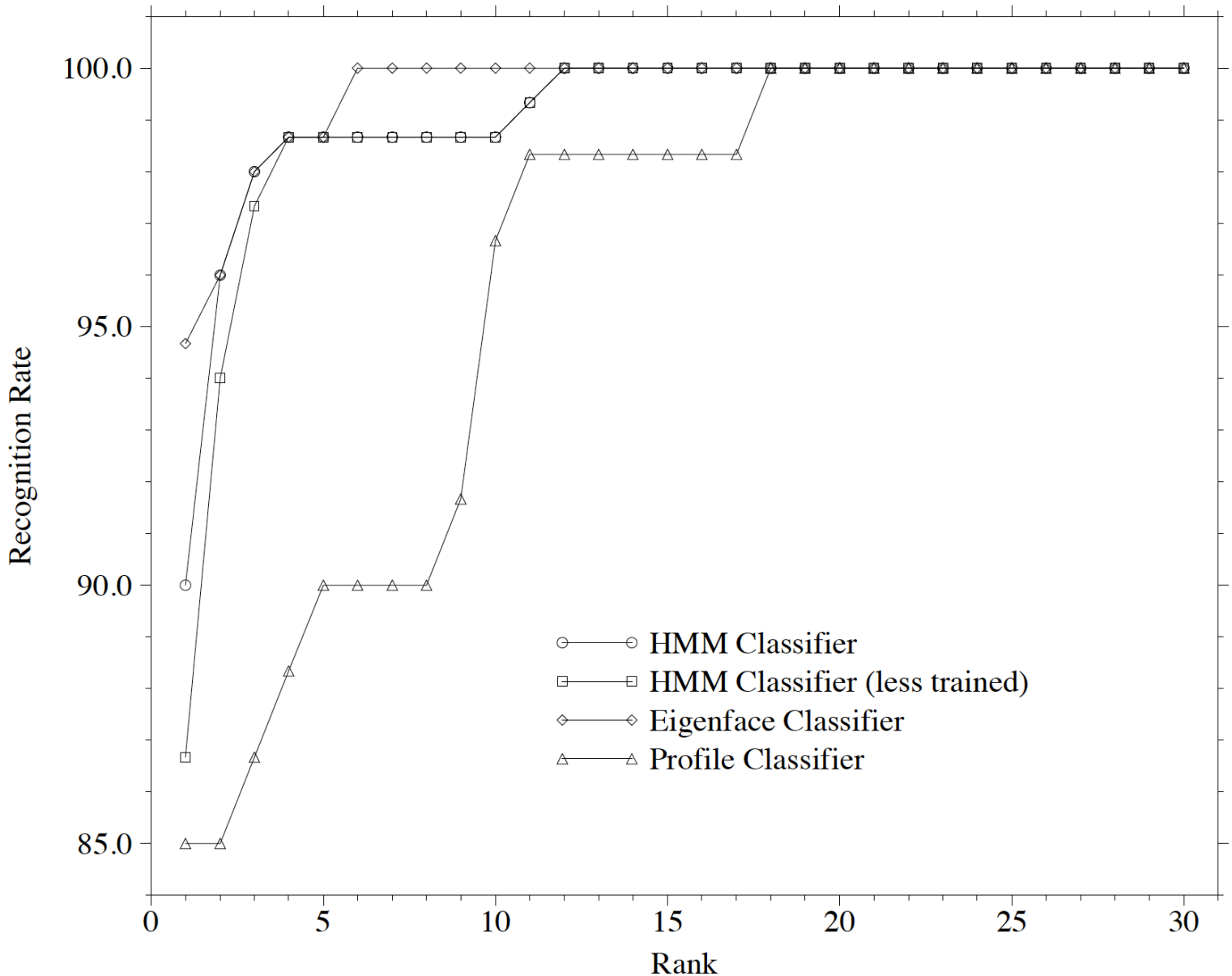
Ensemble Diversity

- Goal is to generate a diverse classifier ensembles.
- Consider two classifiers K_1 and K_2.
  - N_{00}: number of samples misclassified by K_1 and K_2
  - N_{10}: correctly classified by K_1, misclassified by by K_2
  - N_{01}: misclassified by by K_1, correctly classified by K_2
  - N_{11}: correctly classified by K_1 and K_2
- Common pairwise diversity measures:
  - Disagreement (should be high): $$\frac{N_{10} + N_{01}}{N_{00} + N_{10} + N_{01} + N_{11}}$$
  - Double-fault (should be low): $$\frac{N_{00}}{N_{00} + N_{10} + N_{01} + N_{11}}$$
- Empirical results indicate that diversity measures correlate with gain in accuracy.

Ensemble Generation

- Variation of classifier architecture and parameters. For example, MLP:
  - Number of hidden layers, number of neurons per layer.
  - Initial weights.
  - Number of training iterations, order of learning samples, learning rate, parameter of sigmoid threshold function, etc.
  - Focus on a subset of features.
- *Bagging* (bootstrap aggregating):
  - Create K times a new training set with |S| elements by choosing |S| times a training sample.
  - Each training sample is chosen with equal probability 1 / |S|. Multiple selections are kept in the new training set.
  - Train K classifiers on the K different training sets.

Example: Person Identification (Individual)



Example: Person Identification (Combined)