N-tuple Classifier

Robert M. Haralick

Computer Science, Graduate Center
City University of New York
Solving Complex Computational Problems

- Break global problem into smaller subproblems
- Each of which can be solved independently
- Optimally solve the subproblems
- Combine the solutions to the subproblems to obtain the solution to the global problem
Decompositions

- Maximize the Dependencies within each of the smaller problems
- Maximize the Independence between each of the smaller problems
Decompositions

- Recursive Decomposition
- Data Decomposition
- Functional Decompositions
- Search Space Decompositions
Sometimes the Solution to the decomposed problem is optimal
Sometimes the Solution to the decomposed problem is sub-optimal
The Solution obtained by decomposition can be close to optimal
The Subspace Classifier

Definition

A Subspace Classifier is one that projects the measurement vector to one or more subspaces where the projected vector is processed and then the processed projected vectors are combined in a way to form an assigned classification.

It is typical for the projection operators to be orthogonal projection operators. It is not unusual for the projection operators to be axis aligned.
Developed For Printed Character Recognition

Each character is contained in an image of $I \times J$ pixels

Each pixel is a binary 1 or a binary 0

Designed for table lookup hardware

Bledsoe and Browning had an array of $10 \times 15$ pixels.

In general, $N$ Randomly Chosen Pixel Positions
A small number of pixel positions are randomly selected for each subspace

- Bledsoe and Browning selected 2 pixels at a time

Have multiple sets of such randomly selected pixel positions

- Bledsoe and Browning selected 75 sets of randomly selected mutually exclusive pixel pairs
- Each subspace was two dimensions, there were 4 possible values in each subspace dimension

Each of the pixel positions had been thresholded (quantized) and contained a binary 0 or a binary 1
N-Tuple Method

- Concatenate all the binary values to form a binary number as an address for the subspace.
- The memory required for each subspace for each class was 2 dimensions \( \times \) 4 possible values per subspace.
  - Number of classes: 26 letters, 10 digits make up 36 classes.
  - For each of 36 classes.
  - Bledsoe and Browning implementation needed 8 locations for each of the 75 two dimensional subspaces for each class.
  - The number of memory locations was 600 for each of 36 classes.
- Use the 4 bit numbers to access an address in memory.
- For each subspace.
- For each character class.
**N-Tuple Method**

- $M$ pattern sets of $N$ randomly selected pixel positions
- A printed character produces $M$ $N$-digit binary numbers $b_1, \ldots, b_M$
- $K$ character classes
- $T_{mk}$ lookup table for pattern set $m$ and class $k$
- $T_{mk}(b_m)$ holds the fraction of times a character in the training set of class $k$ has the binary number $b_m$ for the $m^{th}$ pattern set
- Compute
  - $f_k = \prod_{m=1}^{M} T_{mk}(b_m)$
  - $f_k = \sum_{m=1}^{M} T_{mk}(b_m)$
- Assign the character to unique class $k$, if there is one, for which $f_k > 0$ is highest
- Otherwise reserve decision
An Alternate N-Tuple Method

- $M$ pattern sets of $N$ randomly selected pixel positions
- A printed character produces $M$ binary numbers $b_1, \ldots, b_M$
- $K$ character classes
- $T_m$ lookup table for pattern set $m$
- $T_m(b_m)$ holds the subset of classes most associated with the binary number $b_m$ for the $m^{th}$ pattern set
- Compute
  - $f = \bigcap_{m=1}^{M} T_m(b_m)$
  - Assign the character to unique class $k$, if there is one, where $k \in f$ and $|f| = 1$
  - Otherwise reserve decision
The N-tuple Calculation for Class \( k \)

\[
\begin{align*}
X_1 & \rightarrow Q_1 \rightarrow A_1 \rightarrow T_{1k} \\
X_2 & \rightarrow Q_2 \rightarrow A_2 \rightarrow T_{2k} \\
X_3 & \rightarrow Q_3 \rightarrow A_3 \rightarrow T_{3k} \\
& \quad \vdots \\
X_{V-2} & \rightarrow Q_{V-2} \rightarrow A_{M-2} \rightarrow T_{M-2k} \\
X_{V-1} & \rightarrow Q_{V-1} \rightarrow A_{M-1} \rightarrow T_{M-1k} \\
X_V & \rightarrow Q_V \rightarrow A_M \rightarrow T_{Mk} \\
\sum & \rightarrow Y_k
\end{align*}
\]
The N-tuple Class Index Generator

if $Y_k > Y_i, i \neq k$, $k = \text{Argmax}\{Y_1, Y_2, \ldots, Y_K\}$

else $k = \text{reserved decision}$

Class Scores

Class Index
Consider the five dimensional measurement vector \((a, b, c, d, e)\) where

- \(a\) is the value produced by feature \(f_1\)
- \(b\) is the value produced by feature \(f_2\)
- \(\vdots\)
- \(e\) is the value produced by feature \(f_5\)
The Need For the Indexed Tuple

- Project the measurement vector \((a, b, c, d, e)\) to the third and fifth feature
- The resulting tuple is \((c, e)\).
- But now we have lost from which features \(c\) and \(e\) came.

In the database world, every value comes from a field and the connection between field and value is never lost.
Index Sets serve as Field Names

The tuple \((a, b, c, d, e)\) is written as
\((\{1, 2, 3, 4, 5\}, (a, b, c, d, e))\)

\((c, d)\) is written as \((\{3, 4\}, (c, d))\)

\((a, b, e)\) is written as \((\{1, 2, 5\}, (a, b, e))\)

A tuple list \(R = \langle (a, b, e), (q, r, s), (t, x, z) \rangle\) is written as
\((\{1, 2, 5\}, R)\)

- First component is an index set for the features
- Second component is a set of tuples
- Each component of a tuple is the value for the corresponding indexed features
Suppose that $S$ is a tuple list with respect to the index set $I$ $(I, S)$

Let $J \subset I$.

The projection of $(I, S)$ from the space indexed by $I$ to the subspace indexed by $J$

$\pi_J(I, S) = (J, R)$
\[ \pi_{\{2\}}(\{1, 2\}, R) \]

\[ \pi_{\{1\}}(\{1, 2\}, R) \]
N-tuple Method Using Index Sets and Projections

- Index Set $I = \{1, \ldots, V\}$
- $X_1, \ldots, X_V$ are the $V$ quantized features
- $L_1, \ldots, L_V$ are the corresponding range sets
  - $X_v \in L_v$, $v = 1, \ldots, V$
  - Measurement Space $\mathcal{M} = \times_{i \in I} L_i$
- $\langle (I, x_1), \ldots, (I, x_Z) | x_z \in \mathcal{M} \rangle$ Measurement Sequence
- $\langle c_1, \ldots, c_z \rangle$ corresponding sequence of class tags
- $\{\langle (I, x_1), \ldots, (I, x_Z) | x_z \in \mathcal{M} \rangle, \langle c_1, \ldots, c_z \rangle\}$ Training Set
- $J_1, \ldots, J_M \subset I$ are the $M$ index sets specifying subspaces
- $\pi_{J_m}(I, x_z) = (J_m, u_z)$, $u_z \in \times_{j \in J_m} L_j$ Projection of $I$ onto $J_m$
N-tuple Method

Tables For Each Index Set and Class

- $Z_1 = |\{z \in [1, Z] | c_z = 1\}|$
- $Z_2 = |\{z \in [1, Z] | c_z = 2\}|$
- $T_{m1}(J_m, u) = |\{z \in [1, Z] | (J_m, u) = \pi_{J_m}(I, x_z), c_z = 1\}|/Z_1$
- $T_{m2}(J_m, u) = |\{z \in [1, Z] | (J_m, u) = \pi_{J_m}(I, x_z), c_z = 2\}|/Z_2$

Scores For Each Class

- $S_k(I, x) = \sum_{m=1}^{M} T_{mk}(\pi_{J_m}(I, x))$

Identification

- Assign class $k$ if $S_k(I, x) > S_j(I, x) + \epsilon, j \neq k$
- Otherwise Assign reserve decision
Scanning N-tuple Classifier

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

$J_0 = \{0, 1, 2\} \\
J_1 = \{1, 2, 3\} \\
\vdots \\
J_9 = \{7, 8, 9\}$
The N-tuple Subspace Classifier is a kind of universal approximator.

**Conjecture**

Let $M = \times_{d=1}^{D} L_d$ be the D-dimensional measurement space. Let $f : M \rightarrow \{0, 1\}$ be a given function associating every measurement tuple with a 0 or a 1. Let $P$ be a probability distribution on $M$. Let $T$ be the tables and $T$ be the function that the n-tuple method produces to assign a class. If $f$ is ‘zzz’ simple, then for every $\epsilon > 0$, there exists $K << D$ and $M < \binom{D}{K}$ and a two class N-tuple subspace classifier $C = (M, \mathcal{J}, \mathcal{T}, K, M)$ such that

$$P(\{x \in M | f(x) \neq T(x)\} < \epsilon$$
Where Did the Olives Come From?

- Classes
  - Northern Italy
  - Southern Italy
  - Sardinia

- Fatty Acid Measurements
  - Eicosenoic: \( x_1 \)
  - Linoleic: \( x_2 \)
Olives

Linoleic

Eicosenoic
Olives

Linoleic

Eicosenoic
Decision Tree

![Decision Tree Diagram]

- $x_1 \geq 0.07$
  - $x_2 \geq 10.5$
    - $Y$: S. Italy
    - $N$: N. Italy
  - $N$: Sardinia

29 / 53
Olives
The N-tuple Calculation for Class $k$

$X_1 \rightarrow Q_1 \rightarrow A_1 \rightarrow T_{1k}$

$X_2 \rightarrow Q_2 \rightarrow A_2 \rightarrow T_{2k}$

$X_3 \rightarrow Q_3 \rightarrow A_3 \rightarrow T_{3k}$

\[ \ldots \]

$X_{V-2} \rightarrow Q_{V-2} \rightarrow A_{M-2} \rightarrow T_{M-2k}$

$X_{V-1} \rightarrow Q_{V-1} \rightarrow A_{M-1} \rightarrow T_{M-1k}$

$X_V \rightarrow Q_V \rightarrow A_M \rightarrow T_{Mk}$

\[ \sum \rightarrow Y_k \]
Non-uniform Quantization

$Q_1$  $Q_2$
The N-tuple Calculation for Class $k$

$X_1 \rightarrow Q_1 \rightarrow A_1 \rightarrow T_{1k}$

$X_2 \rightarrow Q_2 \rightarrow A_2 \rightarrow T_{2k}$

$X_3 \rightarrow Q_3 \rightarrow A_3 \rightarrow T_{3k}$

$\vdots$

$X_{V-2} \rightarrow Q_{V-2} \rightarrow A_{M-2} \rightarrow T_{M-2k}$

$X_{V-1} \rightarrow Q_{V-1} \rightarrow A_{M-1} \rightarrow T_{M-1k}$

$X_V \rightarrow Q_V \rightarrow A_M \rightarrow T_{Mk}$

$\sum \rightarrow Y_k$

Projections

Measurement
Quantizers
Tuple

Address
Generators

Tables
Class $k$

Combiner
The N-tuple Class Index Generator

if $Y_k > Y_i$, $i \neq K$, $k = \text{Argmax}\{Y_1, Y_2, \ldots, Y_K\}$
else reserved decision

Class Scores

Class Index
Optimizing the N-tuple Classifier

- **Quantization**
  - Optimize the number of quantized levels for each feature
  - Find the Optimal Quantizer boundaries
- **Projections**
  - Find the Optimal Index Sets
- **Tables**
  - Find the Optimal Values for all Table Entries
- **Combiner**
  - Find the Optimal way to Combine Scores
- **Class Index**
  - Optimize the way the Class Index is Determined
N-tuple Method Using Measurement Conditional Probabilities

\[
\begin{align*}
X_1 & \quad Q_1 \quad A_1 \quad T_{1k} \\
X_2 & \quad Q_2 \quad A_2 \quad T_{2k} \\
X_3 & \quad Q_3 \quad A_3 \quad T_{3k} \\
& \quad \quad \quad \quad \vdots \\
X_{V-2} & \quad Q_{V-2} \quad A_{M-2} \quad T_{M-2k} \\
X_{V-1} & \quad Q_{V-1} \quad A_{M-1} \quad T_{M-1k} \\
X_V & \quad Q_V \quad A_M \quad T_{Mk} \\
\end{align*}
\]

\[
\sum \quad Y_k
\]
Conditional Probability of Class Given Projected Tuple

\[ T_{mk}(J_m, u) = \hat{\text{Prob}}((J_m, u) \mid k) \]

\[ \hat{\text{Prob}}(J_m, u) = \sum_{k'=1}^{K} \hat{\text{Prob}}((J_m, u) \mid k') P(k') \]

\[ \hat{\text{Prob}}(k \mid (J_m, u)) = \frac{\hat{\text{Prob}}((J_m, u) \mid k) P(k)}{\sum_{k'=1}^{K} \hat{\text{Prob}}((J_m, u) \mid k') P(k')} \]

\[ T_{mk}(\pi J_m(I, x)) = \hat{\text{Prob}}(\pi J_m(I, x) \mid k) \]

\[ \hat{\text{Prob}}(k \mid \pi J_m(I, x)) = \frac{\hat{\text{Prob}}(\pi J_m(I, x) \mid k) P(k)}{\sum_{k'=1}^{K} \hat{\text{Prob}}(\pi J_m(I, x) \mid k') P(k')} \]
$T_{mk} (\pi_{J_m}(l, x)) = \hat{Prob}(k | \pi_{J_m}(l, x)) = \frac{\hat{Prob}(\pi_{J_m}(l,x) | k)P(k)}{\sum_{k'=1}^{K} \hat{Prob}(\pi_{J_m}(l,x) | k')P(k')}$
Score Generator

\[ S_k = \sum_{m=1}^{M} \hat{P}(k | \pi_{J_m}(l, x)) \]

Conditional Probabilities

Class Given Projected Measurement
A Bayes rule can always be implemented as a deterministic decision rule

\[
\begin{array}{|c|c|c|c|}
\hline
 & 1 & 2 & K \\
\hline
T & P_T(1, d) & e(1, 1) & e(1, 2) & e(1, K) \\
R & P_T(2, d) & e(2, 1) & e(2, 2) & e(2, K) \\
U & & & \cdots & \\
E & & & \vdots & \\
K & P_T(K, d) & e(K, 1) & e(K, 2) & e(K, K) \\
\hline
\end{array}
\]

\[\sum_{j=1}^{K} e(j, k) P_T(j, d)\]

\(P_T(j, d)\) is the fraction of instances that a \(d\) from the training set has true class \(j\)

Assign any class \(k\) to \(d\) such that \(\sum_{j=1}^{K} e(j, k) P_T(j, d)\) is maximal
Maximizing Expected Economic Gain: Subspace Case

- Training Set: \( \{ \langle x_1, \ldots, x_Z \rangle, \langle c_1, \ldots, c_Z \rangle \} \)
- Tuple \( x_Z \) produces \( K \) scores \( S_1(x_Z), \ldots, S_K(x_Z) \)
- The scores are quantized
  - \( q_k : R \to L_k = \{0, 1, \ldots, P_k\}, \ k = 1, \ldots, K \)
  - \( q_1(S_1(x_Z)), \ldots, q_K(S_K(x_Z)) \)
- The quantized score produces an address
  - \( a(q_1(S_1(x_Z)), \ldots, q_K(S_K(x_Z))) \)
- The address enables us to define the table \( T \)
  - \( T(k, b) = \frac{|\{z \in [1,Z] \mid b = a(q_1(S_1(x_z)), \ldots, q_K(S_K(x_z))), c_z = k\}|}{Z} \)
  - \( T(k, b) \) is the fraction of instances that an \( x_z \) from the training sequence has address \( b \) and class \( k \)
  - \( (T(1, b), T(2, b) \ldots T(K, b)) \) are the probabilities that an \( x \) that produces address \( b \) will have true classes \( (1, 2, \ldots, K) \)
- Assign any class \( k \) to an \( x \) that produces address \( b \) such that \( \sum_{j=1}^{K} e(j, k) T(j, b) \) is maximal
Tuple $x$ produces address $b$

if $T(k, b) > T(i, b), i \neq k$ assign class $k$ to $x$
else assign reserved decision
Dehn in 1911 articulated the following fundamental group
Decision Problems

- Word Problem
- Conjugacy Problem
- Isomorphism Problem

Haralick et. al. describes a successful machine learning approach related to Whitehead minimal words in free groups.

Conjugacy Problem

**Definition**

Given a group $G$ and elements $u$ and $v \in G$, determine if there is an element $a \in G$ such that

$$u = ava^{-1}$$
Non-Abelian Infinite Groups Tested

- Three Non-virtually Nilpotent Polycyclic Groups
  - $O \rtimes U_{14}$ determined by $x^9 - 7x^3 - 1$
  - $O \rtimes U_{16}$ determined by $x^{11} - x^3 - 1$
  - $O \rtimes U_{34}$ determined by $x^{23} - x^3 - 1$

- Two Non-polycyclic Metabelian Groups
  - Baumslag-Solitar Group BS(1,2)
  - Generalized Metabelian Baumslag-Solitar group GMBS(2,3)

- A Non-Solvable Linear Group
  - $\text{SL}(2, \mathbb{Z})$

In these infinite groups the conjugacy problem is undecidable.

Definition

A family of problems with Yes/No answers is Undecidable if and only if there is no algorithm that terminates with the correct answer for every problem in the family.

Definition

The Halting Problem is to determine whether there exists an algorithm that takes a computer program and the computer program’s input and decides whether it eventually halts instead of entering an infinite loop.

The halting problem is undecidable.
Protocols

- Machine Learning Techniques
  - Decision Trees: Scikit-learn DecisionTreeClassifier
  - Random Forests: Scikit-learn Random ForestClassifier
  - N-Tuple Neural Network: Our own Python Implementation

- Data Generation
  - Three Independent Data Sets For Each Group
    - Training
    - Optimization
    - Verification
  - 20,000 Geodesic word pairs
    - 10,000 conjugate pairs
    - 10,000 non conjugate pairs
## Best Performing Decision Tree Classifiers

<table>
<thead>
<tr>
<th>Group</th>
<th>Split Criterion</th>
<th>Depth</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>BS(1,2)</td>
<td>Entropy</td>
<td>Depth Limit</td>
<td>92.00%</td>
</tr>
<tr>
<td>$O \times U_{14}$</td>
<td>Entropy</td>
<td>Depth Limit</td>
<td>98.49%</td>
</tr>
<tr>
<td>$O \times U_{16}$</td>
<td>Entropy</td>
<td>No Depth Limit</td>
<td>97.23%</td>
</tr>
<tr>
<td>$O \times U_{34}$</td>
<td>Entropy</td>
<td>Depth Limit</td>
<td>98.47%</td>
</tr>
<tr>
<td>GMBS(2,3)</td>
<td>Gini Impurity</td>
<td>Depth Limit</td>
<td>95.43%</td>
</tr>
<tr>
<td>SL(2, $\mathbb{Z}$)</td>
<td>Entropy</td>
<td>No Depth Limit</td>
<td>96.26%</td>
</tr>
</tbody>
</table>
## Best Performing Random Forest Classifiers for All Groups

<table>
<thead>
<tr>
<th>Group</th>
<th>Split Criterion</th>
<th>Depth</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>BS(1,2)</td>
<td>Entropy</td>
<td>No Depth Limit</td>
<td>93.64%</td>
</tr>
<tr>
<td>(O \rtimes U_{14})</td>
<td>Entropy</td>
<td>No Depth Limit</td>
<td>98.69%</td>
</tr>
<tr>
<td>(O \rtimes U_{16})</td>
<td>Entropy</td>
<td>Depth Limit</td>
<td>98.19%</td>
</tr>
<tr>
<td>(O \rtimes U_{34})</td>
<td>Entropy</td>
<td>No Depth Limit</td>
<td>98.89%</td>
</tr>
<tr>
<td>GMBS(2,3)</td>
<td>Entropy</td>
<td>No Depth Limit</td>
<td>96.49%</td>
</tr>
<tr>
<td>SL(2, \mathbb{Z})</td>
<td>Entropy</td>
<td>No Depth Limit</td>
<td>97.47%</td>
</tr>
<tr>
<td>Group</td>
<td>Number of Subspaces</td>
<td>Size of Subspaces</td>
<td>Accuracy</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>BS(1,2)</td>
<td>30</td>
<td>4</td>
<td>92.41% (log)</td>
</tr>
<tr>
<td>$O \rtimes U_{14}$</td>
<td>20</td>
<td>3</td>
<td>98.77% (log)</td>
</tr>
<tr>
<td>$O \rtimes U_{16}$</td>
<td>20</td>
<td>5</td>
<td>98.46% ($\Sigma$)</td>
</tr>
<tr>
<td>$O \rtimes U_{34}$</td>
<td>100</td>
<td>3</td>
<td>99.50% (log)</td>
</tr>
<tr>
<td>GMBS(2,3)</td>
<td>30</td>
<td>4</td>
<td>96.13% ($\Sigma$)</td>
</tr>
<tr>
<td>SL(2, $\mathbb{Z}$)</td>
<td>50</td>
<td>4</td>
<td>99.81% (log)</td>
</tr>
<tr>
<td>Group</td>
<td>Decision Tree</td>
<td>Random Forest</td>
<td>N-tuple</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------</td>
<td>---------------</td>
<td>-----------</td>
</tr>
<tr>
<td>BS(1,2)</td>
<td>92.00%</td>
<td>93.64%</td>
<td>92.41%</td>
</tr>
<tr>
<td>$O \times U_{14}$</td>
<td>98.49%</td>
<td>98.69%</td>
<td>98.77%</td>
</tr>
<tr>
<td>$O \times U_{16}$</td>
<td>97.23%</td>
<td>98.19%</td>
<td>98.46%</td>
</tr>
<tr>
<td>$O \times U_{34}$</td>
<td>98.47%</td>
<td>98.89%</td>
<td>99.50%</td>
</tr>
<tr>
<td>GMBS(2,3)</td>
<td>95.43%</td>
<td>96.49%</td>
<td>96.13%</td>
</tr>
<tr>
<td>SL(2, $\mathbb{Z}$)</td>
<td>96.26%</td>
<td>97.47%</td>
<td>99.81%</td>
</tr>
</tbody>
</table>
# N-Tuple: Accuracy by Class for Tested Groups

<table>
<thead>
<tr>
<th>Group</th>
<th>Conjugate</th>
<th>Non-Conjugate</th>
</tr>
</thead>
<tbody>
<tr>
<td>BS(1,2)</td>
<td>88.17%</td>
<td>96.64%</td>
</tr>
<tr>
<td>$O \rtimes U_{14}$</td>
<td>99.95%</td>
<td>97.58%</td>
</tr>
<tr>
<td>$O \rtimes U_{16}$</td>
<td>99.50%</td>
<td>97.41%</td>
</tr>
<tr>
<td>$O \rtimes U_{34}$</td>
<td>99.14%</td>
<td>99.86%</td>
</tr>
<tr>
<td>GMBS(2,3)</td>
<td>97.37%</td>
<td>94.88%</td>
</tr>
<tr>
<td>SL(2, $\mathbb{Z}$)</td>
<td>99.87%</td>
<td>99.75%</td>
</tr>
</tbody>
</table>