GMDH and FAKE GAME: Evolving ensembles of inductive models

Pavel Kordík

CTU Prague, Faculty of Electrical Engineering, Department of Computer Science
Outline of the tutorial ...

- Introduction, FAKE GAME framework, architecture of GAME
- GAME – niching genetic algorithm
- GAME – ensemble methods
  - Trying to improve accuracy
- GAME – ensemble methods
  - Models’ quality
  - Credibility estimation
Theory in background

- Knowledge discovery
- Data preprocessing
- Data mining
- Neural networks
- Inductive models
- Continuous optimization
- Ensemble of models
- Information visualization
Problem statement

- Knowledge discovery (from databases) is time consuming and expert skills demanding task.
- Data preprocessing is extremely time consuming process.
- Data mining involves many experiments to find proper methods and to adjust their parameters.
- Black-box data mining methods such as neural networks are not considered credible.
- Knowledge extraction from data mining methods is often very complicated.
Solutions proposed – FAKE GAME concept

FAKE INTERFACE

INPUT DATA

AUTOMATED DATA PREPROCESSING

KNOWLEDGE EXTRACTION and INFORMATION VISUALIZATION

GAME ENGINE

AUTOMATED DATA MINING

KNOWLEDGE
Detailed view of proposed solutions

DATA WAREHOUSING
DATA INTEGRATION
DATA CLEANING
INPUT DATA
DATA INSPECTION
DATA COLLECTION
PROBLEM IDENTIFICATION

FAKE INTERFACE

Math equations
Feature ranking
Credibility estimation
Classification, Prediction, Identification and Regression

Game Engine

Automated Data Preprocessing

Classes boundaries, relationship of variables

Interesting behaviour
The GAME engine for automated data mining

PREPROCESSED DATA → GAME → REGRESSION OR CLASSIFICATION OR IDENTIFICATION OR PREDICTION

What is hidden inside?
Background (GMDH)

- **Group Method of Data Handling** (GMDH) – prof. Ivakhnenko in 1968

- The principle of **INDUCTION** employed:

  - **Inductive model** grows from data, it decomposes a multivariate problem into subproblems, solves it in the space of lower dimensionality (two dimensions for units with 2 inputs), combines partial solutions to get the global solution of the problem.

  - **MIA GMDH** – the most commonly used algorithm

\[
y = ax_1 + bx_2 + c
\]
Our improvements

MIA GMDH -> ModGMDH -> GAME

input variables

output variable

Unified units

Exhaustive search

Non-heuristic search

GA+DC
Recent improvements (GAME)

- Group of Adaptive Models Evolution (GAME)
  - Heterogeneous units
  - Several competing training methods
  - Interlayer connections, number of inputs allowed increases
  - Niching genetic algorithm (will be explained) employed in each layer to optimize the topology of GAME networks.
  - Ensemble of models generated (will be explained)
Heterogeneous units in GAME

**Linear (LinearNeuron):**
\[ y = \sum_{i=1}^{n} a_i x_i + a_{n+1} \]

**Sin (SinusNeuron):**
\[ y = a_{n+1} \sin \left( a_{n+2} \left( \sum_{i=1}^{n} a_i x_i + a_{n+3} \right) \right) + a_0 \]

**Gaussian (GaussianNeuron):**
\[ y = \left( 1 + a_{n+1} \right) e^{-\frac{\sum_{i=1}^{n} (x_i - a_i)^2}{(1+a_{n+2})^2}} + a_0 \]

**Polynomial (CombiNeuron):**
\[ y = \sum_{i=1}^{n} \left( a_i \prod_{j=1}^{m} x_j^r \right) + a_0 \]

**Logistic (SigmNeuron):**
\[ y = \frac{1}{1 + e^{-\sum_{i=1}^{n} a_i x_i}} + a_0 \]

**Rational (PolyFractNeuron):**
\[ y = \frac{a_{n^2+2}}{\sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n} a_{n+i} x_i x_j + a_{n^2+1}} + a_0 \]

**Exponential (ExpNeuron):**
\[ y = a_{n+2} e^{\sum_{i=1}^{n} a_i x_i} + a_0 \]

**Universal (BPNetwork):**
\[ y = \sum_{q=1}^{2n+1} \psi_q \left( \sum_{p=1}^{n} \phi_{pq} (x_p) \right) \]

11/67
Optimization of coefficients (learning)

Gaussian (GaussianNeuron)

We are looking for optimal values of coefficients $a_0$, $a_1$, $\ldots$, $a_{n+2}$

We have inputs $x_1$, $x_2$, $\ldots$, $x_n$ and target output $y$ in the training data set

The difference between unit output $y'$ and the target value $y$ should be minimal for all vectors from the training data set

$$E = \sum_{i=1}^{m} (y' - y)^2$$
How to derive analytic gradient?

\[ E = \sum_{j=0}^{m} (y_j - d_j)^2 , \]

Error of the unit (energy surface)

\[ \nabla E = \left( \frac{\partial E}{\partial a_1}, \frac{\partial E}{\partial a_2}, \ldots, \frac{\partial E}{\partial a_n} \right) , \]

Gradient of the error

\[ y_j = \left( 1 + a_{2n+1} \right) \ast e \left( \frac{\sum_{i=1}^{n} (a_i \ast x_{ij} - a_{n+i})^2}{(1 + a_{2n+2})^2} \right) + a_0 \]

Unit with gaussian transfer function

\[ \frac{\partial E}{\partial a_i} = \sum_{j=0}^{m} \left[ \frac{\partial E}{\partial y_j} \ast \frac{\partial y_j}{\partial \rho_j} \ast \frac{\partial \rho_j}{\partial a_i} \right] \]

Partial derivation of error in the direction of coefficient \( a_i \)
Partial derivatives of the Gauss unit

\[ y_j = \left( 1 + a_{2n+1} \right) e^{\rho_j} + a_0 \]

\[
\frac{\partial \vec{E}}{\partial a_{2n+1}} = 2 \sum_{j=0}^{m} \left[ (y_j - d_j) * e^{\rho_j} * \frac{\sum_{i=1}^{n} \left( a_i * x_{ij} - a_{n+i} \right)^2}{(1 + a_{2n+2})^2} \right]
\]

\[
\frac{\partial \vec{E}}{\partial a_{2n+2}} = 2 \sum_{j=0}^{m} \left[ (y_j - d_j) * (1 + a_{2n+1}) e^{\rho_j} * \frac{\sum_{i=1}^{n} \left( a_i * x_{ij} - a_{n+i} \right)^2}{(1 + a_{2n+2})^3} \right]
\]

\[
\frac{\partial \vec{E}}{\partial a_i} = 2 \sum_{j=0}^{m} \left[ (y_j - d_j) * (1 + a_{2n+1}) e^{\rho_j} * -2 \frac{\alpha_i^2 * x_{ij}^2 - a_{n+i} * x_{ij}}{(1 + a_{2n+2})^2} \right]
\]

\[
\frac{\partial \vec{E}}{\partial a_{n+i}} = 2 \sum_{j=0}^{m} \left[ (y_j - d_j) * (1 + a_{2n+1}) e^{\rho_j} * -2 \frac{\alpha_{n+i} - a_i * x_{ij}}{(1 + a_{2n+2})^2} \right].
\]
Optimization of their coefficients

a) Unit does not provide analytic gradient just error of the unit

b) Unit provides analytic gradient and the error of the unit
Very efficient gradient based training for hybrid networks developed!
Outline of the talk …

• Introduction, FAKE GAME framework, architecture of GAME

• **GAME – niching genetic algorithm**

• **GAME – ensemble methods**
  – Trying to improve accuracy

• **GAME – ensemble methods**
  – Models’ quality
  – Credibility estimation
Genetic algorithm

Computer-Automated Design by Artificial Evolution

- **Initial random designs**
  - \( f(P_1: 12090217) = 5\% \)
  - \( f(P_2: 40030161) = 60\% \)
  - \( f(P_3: 01641801) = 35\% \)

- **Evaluation**

- **Final optimised designs**

- **Selection**

- **Variation**
  - **Mutation**
    - \( P_1': 40030061 \)
    - \( P_2': 40130801 \)
    - \( P_3': 01641161 \)
  - **Crossover**
    - \( P_1: 40030161 \)
    - \( P_2: 40030161 \)
    - \( P_3: 01641801 \)

© Yun Li, University of Glasgow, 1995-97
Example of encoding into chromosome

Niching GA

1234567

Linear transfer unit (LinearNeuron)

1234567

0001000

not implemented

n.i.

Settings

Inputs

Transfer function

1234567

0000110

y = a_1x_1 + a_2x_2 + a_0

Polynomial transfer unit (CombiNeuron)

1234567

1234567

1234567

0000110

2115130

1203211

Settings

Inputs

Transfer function

y = a_1x_1^3 + a_2x_1^2x_2 + a_0
The process of GAME models evolution – Genetic Algorithm

Encoding the GAME unit to genotype (first layer):

Genetic algorithm (no niching)

Generate initial population of $n$ individuals

repeat for $m$ generations

repeat $n/2$ times

Select two fit individuals $p_1, p_2$

Cross them, yielding $c_1, c_2$

Apply mutation, yielding $c'_1, c'_2$

if $f(c'_1) > f(p_1)$ then replace $p_1$

with $c'_1$

if $f(c'_2) > f(p_2)$ then replace $p_2$

with $c'_2$

end

end

Result: REDUNDANT information extracted

All individuals are connected to the most significant feature after standard GA.
Better way is to use worse, but non-correlated units

By using less significant features we get more additional information than by using several best individuals connected to the most significant feature!

\[
\begin{align*}
f(A) &= 8 \\
f(B) &= 7.99 \\
f(C) &= 8 \\
f(X) &= 8 \\
f(Y) &= 5 \\
f(Z) &= 9
\end{align*}
\]

The outputs of units A and B are correlated combining them does not bring improvement in performance.

\[f(C) < f(Z)\]

How to do it?

By using less significant features we get more additional information than by using several best individuals connected to the most significant feature!
Niching Genetic Algorithms

• Niching method - extends genetic algorithm to be able to locate multiple solutions.
• Promote the formation and maintenance of stable subpopulations in genetic algorithm.

Mnoho metod pro zachování různorodosti jedinců (niching methods):

Overspecification – dobré u dynamických systémů
Ecological genetic algorithms – více prostředí
Heterozygote advantage – diploidní genom, fitness zvýšit o dist. rodičů
Crowding (restricted replacement) – nahrazení podobných jedinců v subpop.
Restricted competition – turnaj kompatibilních jedinců (threshold)
Fitness sharing – sharing methods – přelidnino -> hledám jinde
Immune system models - antigeny
Niching GA – Deterministic Crowding

- Subpopulations (niches) are based on the **distance** of individuals.

For units in the GAME network:

*distance of units in one population is based on the phenotypic difference of units.*

Distance = Hamming dist. of chromosomes + Hamming dis. of inputs processed

\[
\begin{array}{c|c|c}
\text{Unit} & \text{Chromosome} & \text{Inputs\_processed} \\
\hline
A & 1000 & 1000 \\
B & 0010 & 0010 \\
C & 010010 & 1100 \\
D & 000011 & 1010 \\
\end{array}
\]

\[
\begin{align*}
d(A, B) &= H(1000, 0010) + H(1000, 0010) = 4 \\
d(C, D) &= H(010010, 000011) + H(1100, 1010) = 4
\end{align*}
\]
Distance of GAME units

Distance\((P_1, P_2)\) = genotypic distance + correlation of errors

- Normalized distance of **Inputs** → Hamming(100010,101100) + features used
- Normalized distance of **Transfer functions** → Euclid distance of coefficients
- Normalized distance of **Other attributes** → Distance of configuration variables

Encoding units to chromosomes:

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Transfer function</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_1</td>
<td>123456</td>
<td></td>
</tr>
<tr>
<td>100010</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Transfer function</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_2</td>
<td>123456</td>
<td></td>
</tr>
<tr>
<td>101100</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Computed from units deviations on training & validation set
The process of GAME models evolution – GA with DC

Encoding the GAME unit to genotype (first layer):

Niching genetic algorithm (Deterministic Crowding)

Generate initial population of $n$ individuals
repeat for $m$ generations
repeat $n/2$ times
Select two individuals $p_1, p_2$ randomly
Cross them, yielding $c_1, c_2$
Apply mutation, yielding $c'_1, c'_2$
if $[d(p_1, c'_1) + d(p_2, c'_2)] \leq [d(p_1, c'_2) + d(p_2, c'_1)]$
  if $f(c'_1) > f(p_1)$ then replace $p_1$ with $c'_1$
  if $f(c'_2) > f(p_2)$ then replace $p_2$ with $c'_2$
else
  if $f(c'_2) > f(p_1)$ then replace $p_1$ with $c'_2$
  if $f(c'_1) > f(p_2)$ then replace $p_2$ with $c'_1$
end
end

Result: MAXIMUM information extracted

Individuals connected to less significant features survived. We gained more info.
Experimental Results: Standard GA versus GA+Deterministic Crowding

- **Units connected to the same input belongs to one niche.**

- **Single uniform population**
  - Genetic Algorithm

- **Three subpopul.(niches)**
  - GA with Deterministic Crowding

<table>
<thead>
<tr>
<th>Time</th>
<th>Day</th>
<th>Rs</th>
<th>Rn</th>
<th>PAR</th>
<th>Tair</th>
<th>RH</th>
<th>uSatVapPr</th>
<th>VapPress</th>
<th>Battery</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100</td>
<td>200</td>
<td>150</td>
<td>100</td>
<td>50</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>100</td>
</tr>
</tbody>
</table>
Experimental Results: Standard GA versus GA+Deterministic Crowding

<table>
<thead>
<tr>
<th></th>
<th>RMS cold water consumption</th>
<th>RMS energy consumption</th>
<th>RMS hot water consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GA</strong></td>
<td>8.80E-03</td>
<td>5.30E-02</td>
<td>4.10E-02</td>
</tr>
<tr>
<td><strong>GA+DC</strong></td>
<td>8.70E-03</td>
<td>5.25E-02</td>
<td>4.05E-02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.20E-02</td>
<td>4.00E-02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.15E-02</td>
<td>3.95E-02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.10E-02</td>
<td>3.90E-02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.05E-02</td>
<td>3.85E-02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.00E-02</td>
<td>3.80E-02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.95E-02</td>
<td>3.75E-02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.90E-02</td>
<td>3.70E-02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.85E-02</td>
<td>3.65E-02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.80E-02</td>
<td>3.60E-02</td>
</tr>
</tbody>
</table>

Results:

The statistical test proved that on the level of significance 95%, the **GA+DC** performs better than simple GA for the **energy** and **hot water consumption**.
Experimental Results: Standard GA versus GA+Deterministic Crowding

Except the models of the *fire radius* attribute, the performance of all other models is significantly better with Deterministic Crowding enabled.
Results: the best way is to combine genotypic and phenotypic distance
Distance of units can be monitored during the training phase.

Start of the niching Genetic Algorithm, units are randomly initialized, trained and their error is computed,

after 30 epochs the niching Genetic Algorithm terminates,

finally units are sorted according to their RMSE, chromosome difference and the correlation.
Another visualization of diversity

Clusters are not apparent in the matrix
Units connected to the same input
Some of them are well trained

Each unit has different genotype (units are connected to diverse inputs)
Half of units are better trained than the others
GA or niching GA?

- Genetic search is more effective than exhaustive search
- The use of niching genetic algorithm (e.g. with Deterministic Crowding) is beneficial:
  - More accurate models are generated
  - Feature ranking can be derived
  - Non-correlated units (active neurons) can be evolved
- The crossover of active neurons in transfer function brings further improvements
Remember? Units with various transfer functions compete in the niching GA

Linear (LinearNeuron)

$y = \sum_{i=1}^{n} a_i x_i + a_{n+1}$

Sin (SinusNeuron)

$y = a_{n+1} \sin \left( a_{n+2} \left( \sum_{i=1}^{n} a_i x_i + a_{n+3} \right) \right) + a_0$

Gaussian (GaussianNeuron)

$y = (1 + a_{n+1}) e^{\frac{-\sum (x_i - a_i)^2}{(1 + a_{n+2})^2}} + a_0$

Polynomial (CombiNeuron)

$y = \sum_{i=1}^{n} a_i \prod_{j=1}^{m} x_j^r + a_0$

Logistic (SigmNeuron)

$y = \frac{1}{1 + e^{-\sum a_i x_i}} + a_0$

Rational (PolyFractNeuron)

$y = \frac{a_{n+2}}{\sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} a_{n+i+j} x_i x_j + a_{n+1}} + a_0$

Exponential (ExpNeuron)

$y = a_{n+2} e^{a_{n+1} \sum a_i x_i} + a_0$

Universal (BPNetwork)

$y = \sum_{q=1}^{2n+1} \psi_q \left( \sum_{p=1}^{n} \phi_{pq} (x_p) \right)$
Natural selection of units really works!

The best performance is achieved when units of \textbf{all} types are competing in the construction process.
Optimization methods available in GAME

- Quasi Newton Method [QuasiNewtonTrainer]
- SADE - genetics method [SADETrainer]
- Particle Swarm Optimization [PSOTrainer]
- Hybrid of the GA and the PSO [HGAPSOTrainer]
- PAL: Differential Evolution [PALDifferentialEvolutionTrainer]
- Differential Evolution [DifferentialEvolutionTrainer]
- Stochastic Orthogonal Search [StochasticOSearchTrainer]
- Orthogonal Search [OrthogonalSearchTrainer]
- WEKA : Quasi-Newton [WEKADifferentialEvolutionTrainer]
- PAL : Conjugate Gradient [PALConjugateGradientTrainer]
- ACO*: Ant Colony Optimization [ACOTrainer]
- CACO: Continuous Ant Colony Optimization [CACOTrainer]
- AACA: Adaptive Ant Colony Algorithm [AACATrainer]
- API [APITrainer]
- DACO: Direct Ant Colony Algorithm [DACOTrainer]
Experimental results of competing training methods on Building data set

RMS error on testing data sets (Building data) averaged over 5 runs
RMS error on the Boston data set

[Box plot diagram showing the distribution of RMS error for different algorithms, including 'All', 'CACO08', 'ACO15std', 'DACO02', 'GAPSO', 'SADE', 'Quasinewton', 'AACAD04', 'PSO', 'API01'.]
Classification accuracy [%] on the Spiral data set

- Random01
- PSO
- Quasinewton
- GAPSO
- API01
- SADE
- ACO15std
- CACO08
- AACA04
- DACO02
- All

38/67
Evaluation on diverse data sets

What is it ALL?
Remember the Genetic algorithm optimizing the structure of GAME?

Linear transfer unit

\[
y = a_1 x_1 + a_2 x_2 + a_0
\]

Polynomial transfer unit

\[
y = a_1 x_1^3 + a_2 x_1^2 x_2 + a_0
\]
Outline of the talk …

• Introduction, FAKE GAME framework, architecture of GAME
• GAME – niching genetic algorithm
  • GAME – ensemble methods
    – Trying to improve accuracy
  • GAME – ensemble methods
    – Models’ quality
    – Credibility estimation
Ensemble models – why to use it?

• Hot topic in machine learning communities

• Successfully applied in several areas such as:
  – Face recognition [Gutta96, Huang00]
  – Optical character recognition [Drucker93, Mao98]
  – Scientific image analysis [Cherkauer96]
  – Medical diagnosis [Cunningham00, Zhou02]
  – Seismic signal classification [Shimshoni98]
  – Drug discovery [Langdon02]
  – Feature extraction [Brown02]
Ensemble models – what is it?

- The collection of models (e.g. neural networks) is trained for the same task.
- Outputs of models are then combined.
Bagging

- Sample with replacement several training sets of size $n$ (instead of having just one training set of size $n$).
- Build model/classifier for each training set.
- Combine classifier’s predictions.
We employed *Bagging* in GAME

- **G**roup of **A**daptive **M**odels **E**volution is a method we are working on in our department (see next slide).

```
Sampling with replacement

Training data

Sample 1 | GAME | GAME model 1
---------|------|-------------
Sample 2 | GAME | GAME model 2
...      | ...  | ...         
Sample M | GAME | GAME model M

Averaging or voting

GAME ensemble
```

45/67
Results of Bagging GAME models

The weighted ensemble has apparent tendency to overfitt the data.

While its performance is superior on the training and validation data, on the testing data there are several individual models performing better.

\[
\Phi_m(x) = \sum_{n=1}^{m} w_n f_n(x), \quad w_i = \frac{\exp(-\alpha e_i)}{\sum_j \exp(-\alpha e_j)}
\]
GAME ensembles are not statistically significantly better than the best individual GAME model!

Why?

• Are GAME models diverse?
  – Vary input data ✓ (bagging)
  – Vary input features ✓ (using subset of features)
  – Vary initial parameters ✓ (random initialization of weights)
  – Vary model architecture ✓ (heterogeneous units used)
  – Vary training algorithm ✓ (heterogeneous units used)
  – Use a stochastic method when building model ✓ (niching GA)

Yes, they use several methods to promote diversity!
Ensemble models – why it works?

**Bias-variance decomposition**
( theoretical tool to study how a training data affects the performance of models/classifiers)

Total expected error of model = **variance** + **bias**

\[
E_T((f - d)^2) = E_T((f - E_T(f))^2) + (E_T(f) - d)^2
\]

\[
MSE(f) = \text{var}(f) + \text{bias}(f)^2
\]

**variance**: the part of the expected error due to the nature of the training set

**bias**: the part of the expected error caused by the fact the model is not perfect

Ensembling reduces **variance**

Ensembling reduces **bias**
GAME ensembles are not statistically significantly better than the best individual GAME model!

Why?

• GAME models grows until optimal complexity
  (grows from the minimal form up to the required complexity)
• Therefore bias cannot be further reduced
• Bagging slightly reduces variance, but not significantly (for datasets we have been experimented with)
Results of Bagging GAME models

Simple ensemble of GAME models

RMS – cold water consumption

RMS – age estimation

Is the ensemble is significantly better than any of individual models?

Suboptimal GAME models   YES

Optimal GAME models       NO
Outline of the talk …

- Introduction, FAKE GAME framework, architecture of GAME
- GAME – niching genetic algorithm
- GAME – ensemble methods
  - Trying to improve accuracy

- GAME – ensemble methods
  - Models’ quality
  - Credibility estimation
Ensembles can estimate credibility

• We found out that **ensembling GAME models does not further improve accuracy** of modelling.
• Why to use ensemble then?

• There is one big advantage that ensemble models can provide: **Using ensemble of models we can estimate credibility** for any input vector.
• Ensembles are starting to be used in this manner in several real world applications:
Visualization of model’s behavior

\[
\begin{align*}
\text{constant} & \quad x_1 \\
\text{moving} & \quad x_2 \\
\text{constant} & \quad x_3
\end{align*}
\]

GAME

\[
\begin{align*}
\text{moving} & \quad x_1 \\
\text{moving} & \quad x_2 \\
\text{constant} & \quad x_3
\end{align*}
\]

GAME

\[
\begin{align*}
\min & \quad x_2 \quad \max \\
\end{align*}
\]

\[
\begin{align*}
y_k & = x_1 - x_3 = \text{const.} \\
y_m & \quad \text{GAME}
\end{align*}
\]
Estimating credibility using ensemble of GAME models

We need to know when the response of a model is based on training data and when it is just random guessing – then we can estimate the credibility of the model.

In regions, where we have enough data, all models from ensemble have compromise response – they are credible.

When responses of models differ then models are not credible (dark background).

While \(x_2\) is varying from min to max and other inputs stay constant, the output of the model shows sensitivity of the model to variable \(x_2\) in the configuration \(x_1, x_3 = \text{const.}\).
Artificial data set – partial definition

Advantages:

- No need of the training data set,
- Modeling method success considered,
- Inputs importance considered.

Credibility: the criterion is a dispersion of models’ responses.
Practical example – Nuclear data

A.G. Ivakhnenko – experiments with GMDH

Extremely sparse data set – 6 dimensions, 10 data vectors
Typical application of softcomputing methods in industry: Building data set

Looking for the optimal configuration of the GAME simulator – Pavel Staněk

GAME ensemble

Black box

Model of energy consumption

Model of cold water consumption

Model of hot water consumption
Building data set

The data set is problem A of the "Great energy predictor shootout" contest. "The Great Energy Predictor Shootout" - The First Building Data Analysis And Prediction Competition; ASHRAE Meeting; Denver, Colorado; June, 1993;

\[ \text{(Temperature outside the building)} \]
\[ \text{(Cold water consumption)} \]

\[ \begin{array}{cccc}
\text{min} & \text{max} \\
y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_9 & \text{max} \\
2 & 3 & 4 & 5 & 6 & 7 & 8 & 1 \\
\end{array} \]

\[ \text{normalized humidity} = 0.272 \]
\[ \text{normalized solar radiation} = 0.7585 \]
\[ \text{normalized wind strength} = 0.248 \]
Credibility of models in classification

GAME model 1    GAME model 2    GAME model 3    GAME models (1*2*3)

Iris Setosa

Iris Virginica

Iris Versicolor
Credibility of models in classification

Before

After
Other visualization techniques
Automated retrieval of 3D plots showing interesting behaviour

Genetic algorithm with special fitness function is used to adjust all other inputs (dimensions)

$$fitness = y_{size} \cdot \frac{1}{p}$$

$$p = \sum_{j=x_{start}}^{x_{start}+x_{size}} \left( \arg\max_{0<i\leq m} y_i(j) - \arg\min_{0<i\leq m} y_i(j) \right)$$

$$y_{size} = \arg\max_{t \in (x_{start}, x_{size})} (\bar{y}(t)) - \arg\min_{t \in (x_{start}, x_{size})} (\bar{y}(t))$$
Conclusion

New quality in modeling:

- GAME models adapt to the data set complexity.
- Hybrid models more accurate than models with single type of unit.
- Compromise response of the ensemble indicates valid system states.
- Black-box model gives estimation of its accuracy $(2.3 \pm 0.1)$.
- Automated retrieval of “interesting plots” of system behavior.
- Java application developed – real time simulation of models.
Benchmarks of GAME

Classification of very complex data set

Benchmarking Spiral data classification problem solved by GAME

Note: inner crossvalidation mechanism to prevent overfitting was disabled for this problem
### Internet advertisements database

<table>
<thead>
<tr>
<th>JavaNNS</th>
<th>RMSE</th>
<th>Yes [%]</th>
<th>No [%]</th>
<th>Correct [%]</th>
<th>Winning MLP topology</th>
</tr>
</thead>
<tbody>
<tr>
<td>ica2.dat</td>
<td>0.169746</td>
<td>68.14</td>
<td>57.30</td>
<td>61.51</td>
<td>2+4+2+1</td>
</tr>
<tr>
<td>ica5.dat</td>
<td>0.079646</td>
<td>78.33</td>
<td>91.81</td>
<td>85.74</td>
<td>5+10+4+1</td>
</tr>
<tr>
<td>ica5.dat</td>
<td>0.078200</td>
<td>80.83</td>
<td>92.98</td>
<td>87.97</td>
<td>5+3+2+1</td>
</tr>
<tr>
<td>ica7.dat</td>
<td>0.071500</td>
<td>79.58</td>
<td>75.17</td>
<td>77.32</td>
<td>7+4+2+1</td>
</tr>
<tr>
<td>ica7.dat</td>
<td>0.066800</td>
<td>77.46</td>
<td>79.87</td>
<td>78.69</td>
<td>7+15+5+1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GAME config.</th>
<th>&quot;Yes&quot; model</th>
<th>&quot;No&quot; model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td>Accuracy</td>
</tr>
<tr>
<td>All-2</td>
<td>0.014689</td>
<td>92.44%</td>
</tr>
<tr>
<td>All-fast-5</td>
<td>0.014559</td>
<td>92.44%</td>
</tr>
<tr>
<td>Standard</td>
<td>0.015827</td>
<td>91.41%</td>
</tr>
<tr>
<td>Combi-5</td>
<td>0.016153</td>
<td>90.72%</td>
</tr>
</tbody>
</table>

BackProp best result: 87.97%
GAME best result: 93.13%
Other benchmarking problems

Zoo database:

Accuracy of classification into genotypes:

- Backpropagation: 80.77%; GAME: **93.5%**

Stock value prediction (S&P DEP RECEIPTS):

Change in direction prediction ratio:

- Backpropagation with momentum: 82.4%; GAME: **91.3%**

Pima indians data set: 10fold crossvalidation results:

<table>
<thead>
<tr>
<th>Pima data set</th>
<th>Bayes</th>
<th>MLP-BP</th>
<th>Lazy training</th>
<th>Soft propagation</th>
<th>GAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy[%)</td>
<td>72.2</td>
<td>77.47</td>
<td>76.69</td>
<td>76.69</td>
<td>79.35</td>
</tr>
<tr>
<td>Std.dev.[%)</td>
<td>6.9</td>
<td>3.75</td>
<td>5.22</td>
<td>2.37</td>
<td>3.89</td>
</tr>
</tbody>
</table>
Thank you!