Information Fusion for PHM Models (Anomaly Detection, Diagnostics, and Prognostics)

Piero P. Bonissone, Chief Scientist
Neil Eklund, Computer Scientist

GE Global Research
Niskayuna, NY 12309, USA

Email: bonissone@ge.com, eklund@ge.com

Outline

1. Introduction and Motivation
2. Example of Model Ensemble Fusion: Random Forest
3. PHM Models
4. Fusion for PHM models
   - 4.1 Case Study 1: Anomaly Detection (1-class classifier)
   - 4.2 Case Study 2: Diagnostics (Multi-class Classifier)
   - 4.3 Case Study 3: Prognostics (Prediction)
5. Summary and Conclusions
6. References and Resources

Acknowledgments

This presentation contains information from a variety of sources. The main sources are listed below.

1 Ensemble Learning:

- Robi Polikar (2009), Scholarpedia, 4(1):2776 - doi:10.4249/scholarpedia.2776
http://www.scholarpedia.org/article/Ensemble_learning
- Fabio Roli (2009), Mini Tutorial on Multiple Classifier Systems - School on the Analysis of Patterns
http://www.analysis-of-patterns.net/files/MCS-Part1.pdf - *[adapted with the author's permission]*

2 Random Forest:

- T. Hastie, R. Tibshirani, J. Friedman (2008). *The Elements of Statistical Learning, 2nd Ed.*, Chapter 15, Springer
http://www-stat.stanford.edu/~hastie/Papers/ESLII.pdf

3 PHM:

- P. Bonissone, N. Iyer (2007), Soft Computing Applications to Prognostics and Health Management (PHM): Leveraging field data and domain knowledge, *9th International Work-Conference on Artificial Neural Networks (IWANN 2007)*, pp. 928-939, San Sebastián (Spain), June 20-22, 2007 – [*GE GR Tech. Report*, 2007GRC799, Oct. 2007]

4.1 Anomaly Detection:

- P. Bonissone, X Hu, R. Subbu (2009), A Systematic PHM Approach for Anomaly Resolution: A Hybrid Neural Fuzzy System for Model Construction, Proc. PHM 2009, San Diego, CA, Sept 27-Oct 1, 2009. - [GE GR Tech. Report GRC839, Sept. 2009]

4.2 Diagnostics:

- W. Yan, F. Xue, Jet Engine Gas Path Fault Diagnosis Using Dynamic Fusion of Multiple Classifiers, IJCNN 2008, Hong Kong, pp-1585-1591 - [GE GR Tech. Report 2008GRC395, May 2008] - *[adapted with the author's permission]*

4.3 Prediction:

- P. Bonissone, F. Xue, and R. Subbu (2008), Fast Meta-models for Local Fusion of Multiple Predictive Models, *Applied Soft Computing Journal,* 2008, doi:10.1016/j.asoc.2008.03.006 - [GE GR Tech. Report 2007GRC832, Oct 2007].

1 Fusion of Ensemble of Models: Motivation &Introduction

1 Introduction and Motivation

- Definition
- Motivations
  - Ceiling of performance of individual classifiers
- Basic Design Criteria
  - Topology: Parallel, Serial, Hybrid
  - Ensemble Models
  - Fuser:
    - Type: Fusion, Selection
    - Fuser as a Meta-model
      - Static or Dynamic
      - Structure & parameters
- Fusion Type
  - Integration (Fusion) of Competing Models
  - Selection of Complementary Models
- Diversity

Definition: *Model Fusion or Ensemble Learning*

- **Ensemble Learning** is the process by which multiple models, such as classifiers or experts, are strategically generated and combined to solve a particular computational intelligence problem.
- Ensemble learning is primarily **used to improve the** (classification, prediction, function approximation, etc.) **performance of a model,** or reduce the likelihood of an unfortunate selection of a poor one.
- Other applications of ensemble learning include assigning a confidence to the decision made by the model, selecting optimal (or near optimal) features, data fusion, incremental learning, nonstationary learning and error-correcting

Source: Robi Polikar (2009), Scholarpedia, 4(1):2776 - doi:10.4249/scholarpedia.2776 http://www.scholarpedia.org/article/Ensemble_learning

Ensemble Learning Example

1. Definition

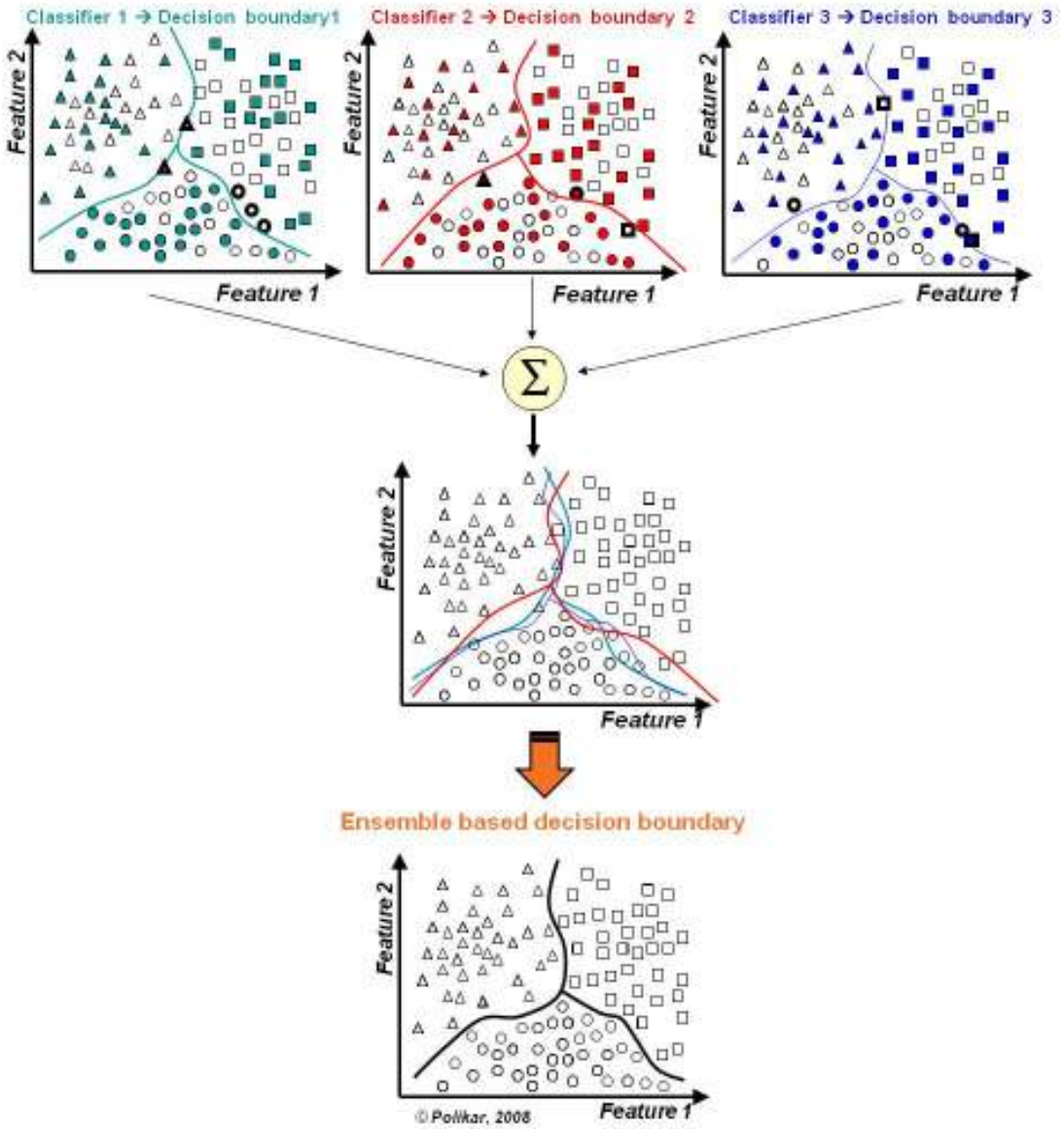


Figure 1: Combining an ensemble of classifiers for reducing classification error and/or model selection

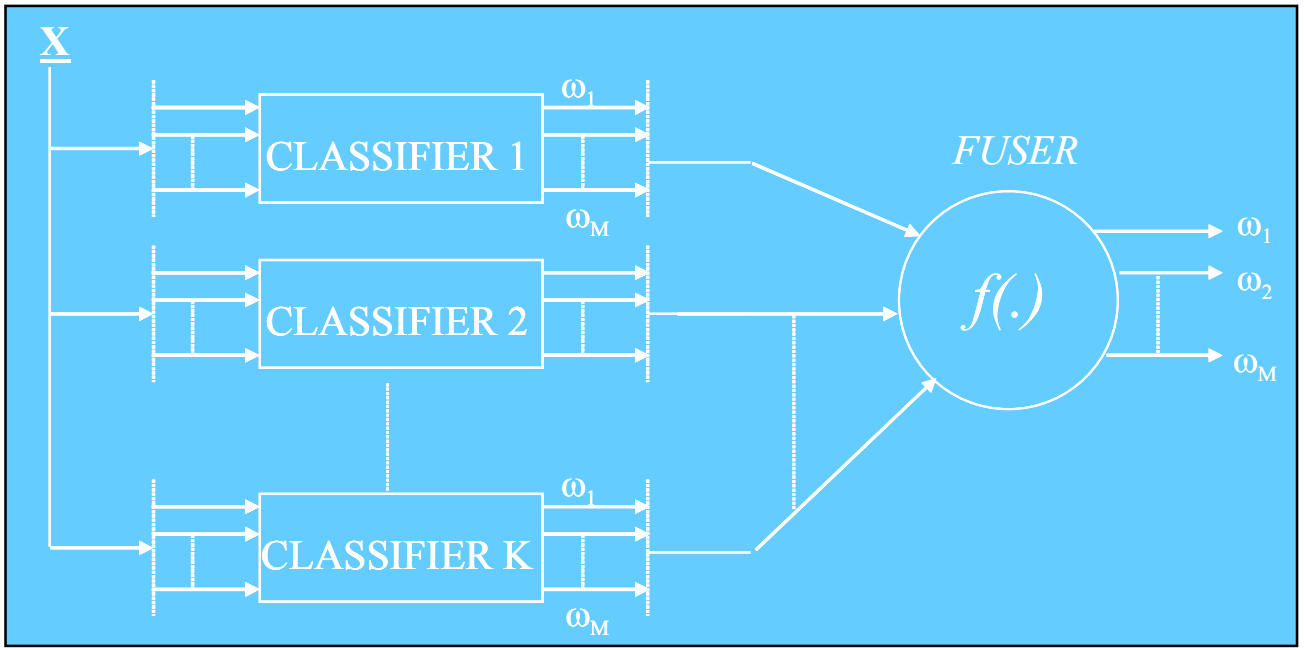
Source: Robi Polikar (2009), Scholarpedia, 4(1):2776 - doi:10.4249/scholarpedia.2776 http://www.scholarpedia.org/article/Ensemble_learning

Motivations

- **Worst Classifier Motivation:**
  - In 2005 Fumera and Roli confirmed theoretically the claim of Tom Dietterich (2000). They proved that ***averaging of classifiers outputs guarantees a better test set performance than the worst classifier of the ensemble*** *(IEEE-T on PAMI, June 2005)*
- **Worst Classifier Motivation:**
  - Beside avoiding the selection of the worst classifier, under particular hypotheses (linear combiners of individual classifiers with unbiased and uncorrelated errors), fusion of multiple classifiers ***can improve the performance of the best individual classifiers***. In some special cases (infinite number of classifiers) fusion can provide the optimal Bayes classifier (Tumer and J. Ghosh; 1996).
  - *This is possible if individual classifiers make "different" errors (diversity).*
- **Computational Motivation:**
  - Many learning algorithms suffer from the problem of local minima Neural Networks, Decision Trees (optimal training is NP-hard!). Finding the best classifier C can be difficult even with enough training data
  - Fusion of multiple classifiers constructed by running the training algorithm from different starting points can better approximate C

Adapted with permission from: Fabio Roli (2009). Mini Tutorial on Multiple Classifier Systems - School on the Analysis of Patterns 2009
http://www.analysis-of-patterns.net/files/MCS-Part1.pdf

Multi Classifiers: Basics



- **Architecture/Topology**
  - Parallel, Serial, Hybrid
- **Classifier Ensemble**
  - Type and number of base classifiers. The ensemble can be subdivided into subsets in the case of non parallel architectures
- **Fuser**
  - Integration (Fusion), Selection

Adapted with permission from: Fabio Roli (2009). Mini Tutorial on Multiple Classifier Systems - School on the Analysis of Patterns 2009
http://www.analysis-of-patterns.net/files/MCS-Part1.pdf

MCS Architectures/Topologies

• **Parallel** topology: multiple classifiers operate in parallel. A single combination function merges the outputs of the individual classifiers

• **Serial/Conditional** topology

-Classifiers are applied in succession, with each classifier producing a reduced set of possible classes

-A primary classifier can be used. When it rejects a pattern, a secondary classifier is used, and so on

• **Hybrid** topologies

Adapted with permission from: Fabio Roli (2009). Mini Tutorial on Multiple Classifier Systems - School on the Analysis of Patterns 2009
http://www.analysis-of-patterns.net/files/MCS-Part1.pdf

The Classifier Ensemble

- The most common type of MCS, widely used and investigated, includes an ensemble of classifiers, named "base" classifiers, and a function for parallel combination of classifier outputs
- The base classifiers are often algorithms of the same type (e.g., decision trees or neural networks), and statistical classifiers are the most common choice.
- The use of hybrid ensembles containing different types of algorithms has been investigated much less, as well as ensembles of structural, graph-based, classifiers have not attracted much attention, although they could be important for some real applications.

Adapted with permission from: Fabio Roli (2009). Mini Tutorial on Multiple Classifier Systems - School on the Analysis of Patterns 2009
http://www.analysis-of-patterns.net/files/MCS-Part1.pdf

Fuser ("Combination" Rule)

Two main categories of fuser:

- **Integration (fusion)** functions: for each pattern, all the classifiers contribute to the final decision. Integration assumes **competitive** classifiers

- **Selection*** functions: for each pattern, just one classifier, or a subset, is responsible for the final decision. Selection assumes **complementary** classifiers

Integration and Selection can be "merged" for designing hybrid fuser

---

Note by P. Bonissone: In the case of continuous outputs (model residuals, predictions, etc.) selection might be improved by interpolating rather than switching between models– see Case Study 1

Adapted with permission from: Fabio Roli (2009). Mini Tutorial on Multiple Classifier Systems - School on the Analysis of Patterns 2009
http://www.analysis-of-patterns.net/files/MCS-Part1.pdf

Fuser Type

Type

- Static (Algebraic functions)

$h_{final}(\mathbf{x}) = \arg\max_j \mu_j(\mathbf{x})$, where the final class supports are computed as follows:

- **Mean rule:** $\mu_j(\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} d_{t,j}(\mathbf{x})$
- **Sum rule:** $\mu_j(\mathbf{x}) = \sum_{t=1}^{T} d_{t,j}(\mathbf{x})$ (provides identical final decision as the mean rule)
- **Weighted sum rule:** $\mu_j(\mathbf{x}) = \sum_{t=1}^{T} w_t d_{t,j}(\mathbf{x})$ (where w_t is the weight assigned to the tth classifier h_t
- **Product rule:** $\mu_j(\mathbf{x}) = \prod_{t=1}^{T} d_{t,j}(\mathbf{x})$
- **Maximum rule** $\mu_j(\mathbf{x}) = \max_{t=1,\cdots,T}\{d_{t,j}(\mathbf{x})\}$
- **Minimum rule** $\mu_j(\mathbf{x}) = \min_{t=1,\cdots,T}\{d_{t,j}(\mathbf{x})\}$
- **Median rule** $\mu_j(\mathbf{x}) = \operatorname{med}_{t=1,\cdots,T}\{d_{t,j}(\mathbf{x})\}$
- **Generalized mean rule** $\mu_{j,\alpha}(\mathbf{x}) = \left(\frac{1}{T}\sum_{t=1}^{T} d_{t,j}(\mathbf{x})^{\alpha}\right)^{1/\alpha}$
  - $\alpha \to -\infty \Rightarrow$ Minimum rule
  - $\alpha \to \infty \Rightarrow$ maximum rule
  - $\alpha = 0 \Rightarrow$ Geometric mean rule
  - $\alpha = 1 \Rightarrow$ Mean rule

- Dynamic (with variable structures and/or parameters)

A meta-model that embodies selection or integration knowledge

Source: Robi Polikar (2009), Scholarpedia, 4(1):2776 - doi:10.4249/scholarpedia.2776 http://www.scholarpedia.org/article/Ensemble_learning

Classifiers "Diversity" vs. Fuser Complexity

Fusion is obviously useful only if the combined classifiers are not the same classifier…

Intuition: classifiers with high accuracy and high "diversity"

The required degree of error *diversity* depends on the fuser complexity

- Majority vote fuser: the majority should be always correct
- Ideal selector ("oracle"): only one classifier should be correct for each pattern

An example, four diversity levels (A. Sharkey, 1997)

Level 1: no more than one classifier is wrong for each pattern
Level 2: the majority is always correct
Level 3: at least one classifier is correct for each pattern
Level 4: all classifiers are wrong for some patterns

Adapted with permission from: Fabio Roli (2009). Mini Tutorial on Multiple Classifier Systems - School on the Analysis of Patterns 2009
http://www.analysis-of-patterns.net/files/MCS-Part1.pdf

Classifiers Diversity Measures: An Example

- Various measures (classifier outputs correlation, Partridge's diversity measures, Giacinto and Roli compound diversity, etc.) can be used to assess how similar two classifiers are

- For two classifier D_i and D_k, L. Kuncheva (2000) proposes the use of Yule's Q statistics:

$$Q_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}$$

where:

	D_k correct (1)	D_k wrong (0)
D_i correct (1)	N^{11}	N^{10}
D_i wrong (0)	N^{01}	N^{00}

- Q varies between –1 and 1. Classifiers that tend to classify the same patterns correctly will have values of Q close to 1, and those which commit errors on different patterns will render Q negative

Adapted with permission from: Fabio Roli (2009). Mini Tutorial on Multiple Classifier Systems - School on the Analysis of Patterns 2009
http://www.analysis-of-patterns.net/files/MCS-Part1.pdf

② Example of Model Ensemble Fusion: Random Forest

2 Example of Model Ensemble Fusion: Random Forest

- SC: Probabilistics and Statistics Systems
- Bias and Variance
- Ensembles of classifiers
- Bagging/boosting
- Classification trees
- Random forests (RF)
- RF Design
- Resources

Soft Computing: Probabilistic Systems

Approximate Reasoning

Functional Approximation/ Randomized Search

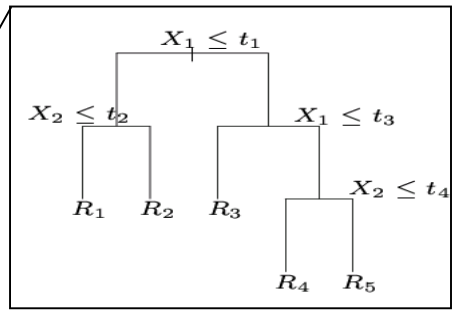
Probabilistic & Statistical Models

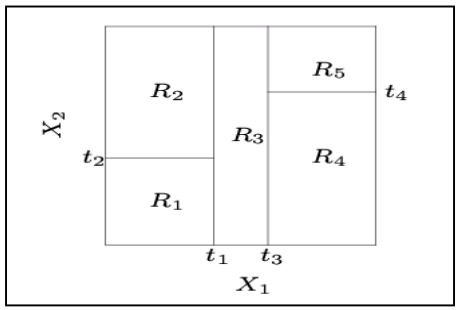
Multivalued & Fuzzy Logics

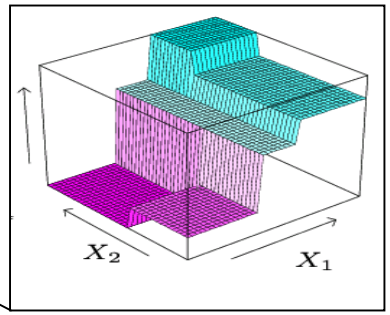
Neural Networks

Evolutionary Algorithms

- Bayesian Belief Nets
- Dempster-Shafer
- CART Trees
- Random Forest

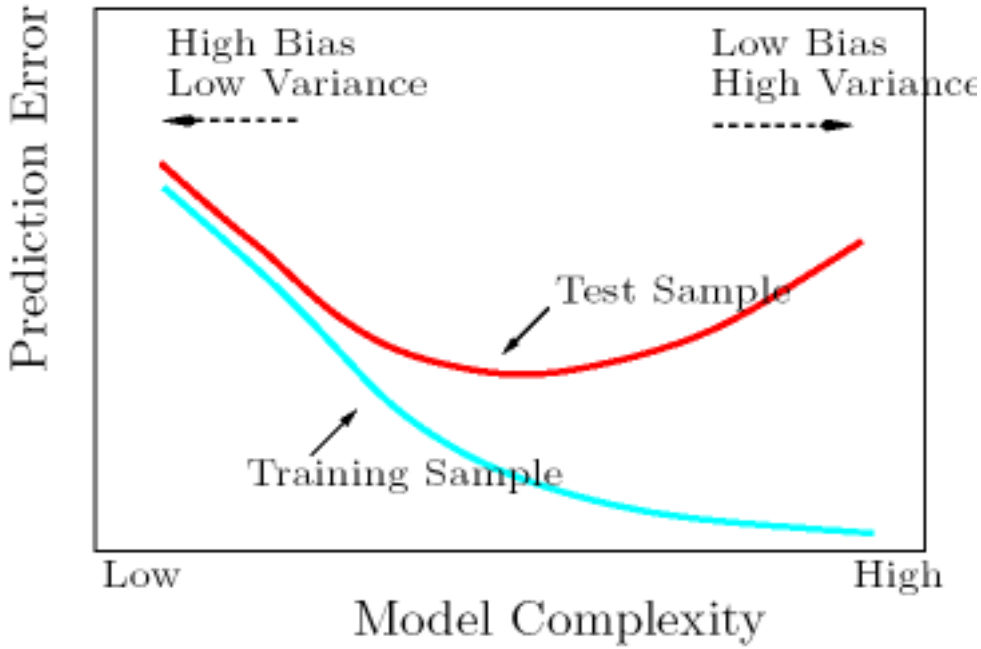






Bias and Variance

- Bias:
  - the classifier (regressor) cannot represent the true function
  - i.e., the classifier (regressor) **underfits** the data
- Variance:
  - variance arises when the classifier **overfits** the data
- There is often a tradeoff
- between bias and variance:



Source : Fig. 2.11, T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Chapter 2, Springer 2009.

Bias and Variance Example

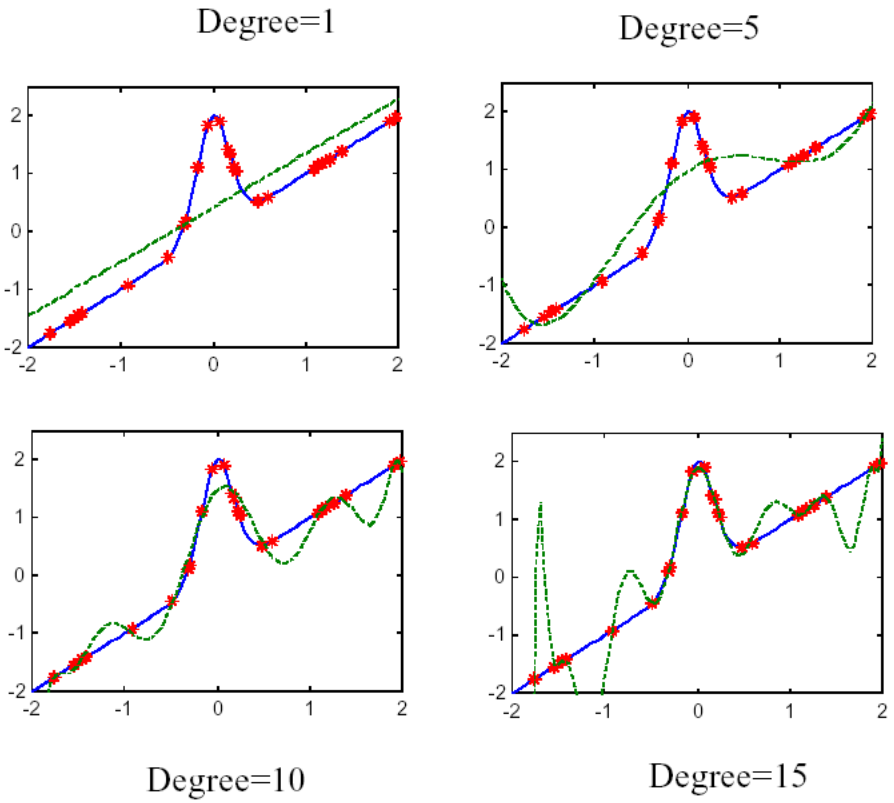
•**red** – experimental data

•**blue** – underlying function

•**green** - fit

• Large bias, small variance:

• Small bias, high variance:



Ensembles of Classifiers

- For any single classifier, there is typically a **tradeoff** between bias and variance.
- Might we achieve high accuracy by combining ensembles of high variance (i.e., uncorrelated), low bias classifiers?
  - variance is reduced by combining outputs
  - bias remains low
- **Basic idea:**

  Train a *set* of *diverse* classifiers (or regressors) and combine their output
- **Math:**
  - For independent identically distributed (iid) variables:

    The average of B iid variables, each with variance σ^2, has a $\boxed{\frac{1}{B}\sigma^2}$ variance
  - For identically distributed (id) variables:

    The average of B id variables with positive pairwise correlation ρ, each with variance σ^2, has a variance:

$$\boxed{\rho\sigma^2 + \frac{(1-\rho)}{B}\sigma^2}$$

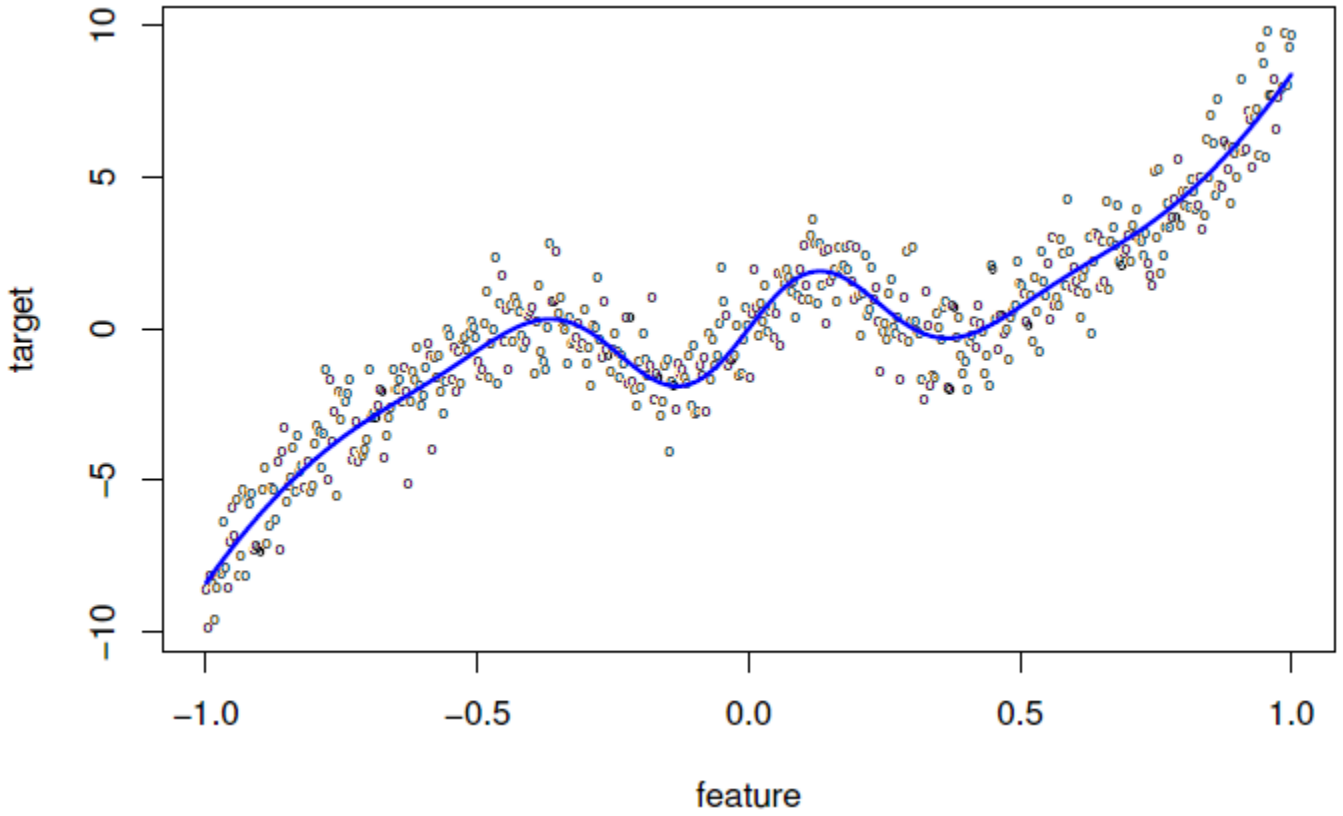
    when B is large, variance is:

$$\boxed{\approx \rho\sigma^2}$$

Source : T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Chapter 15 Springer 2009.

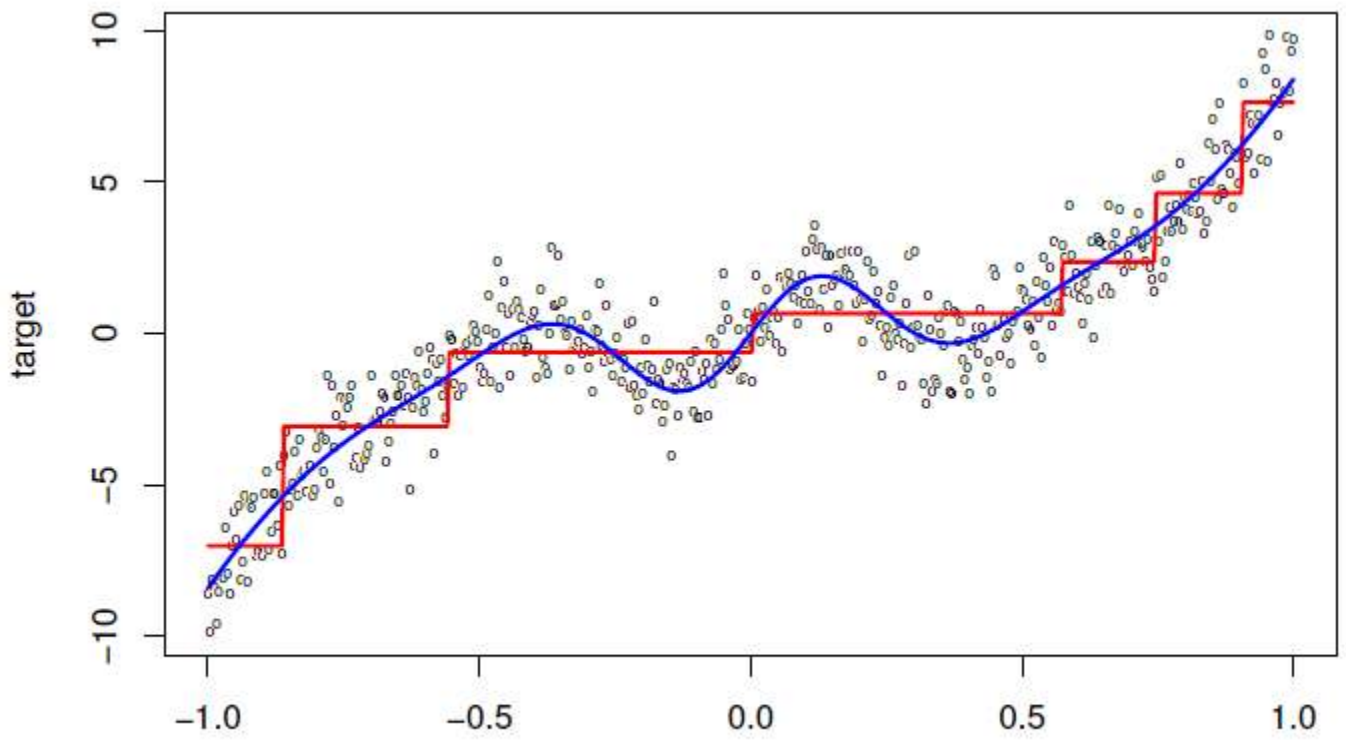
Example: Data Generation

- **blue** – underlying function
- **black** – data with noise



Case 1a: Large Bias…

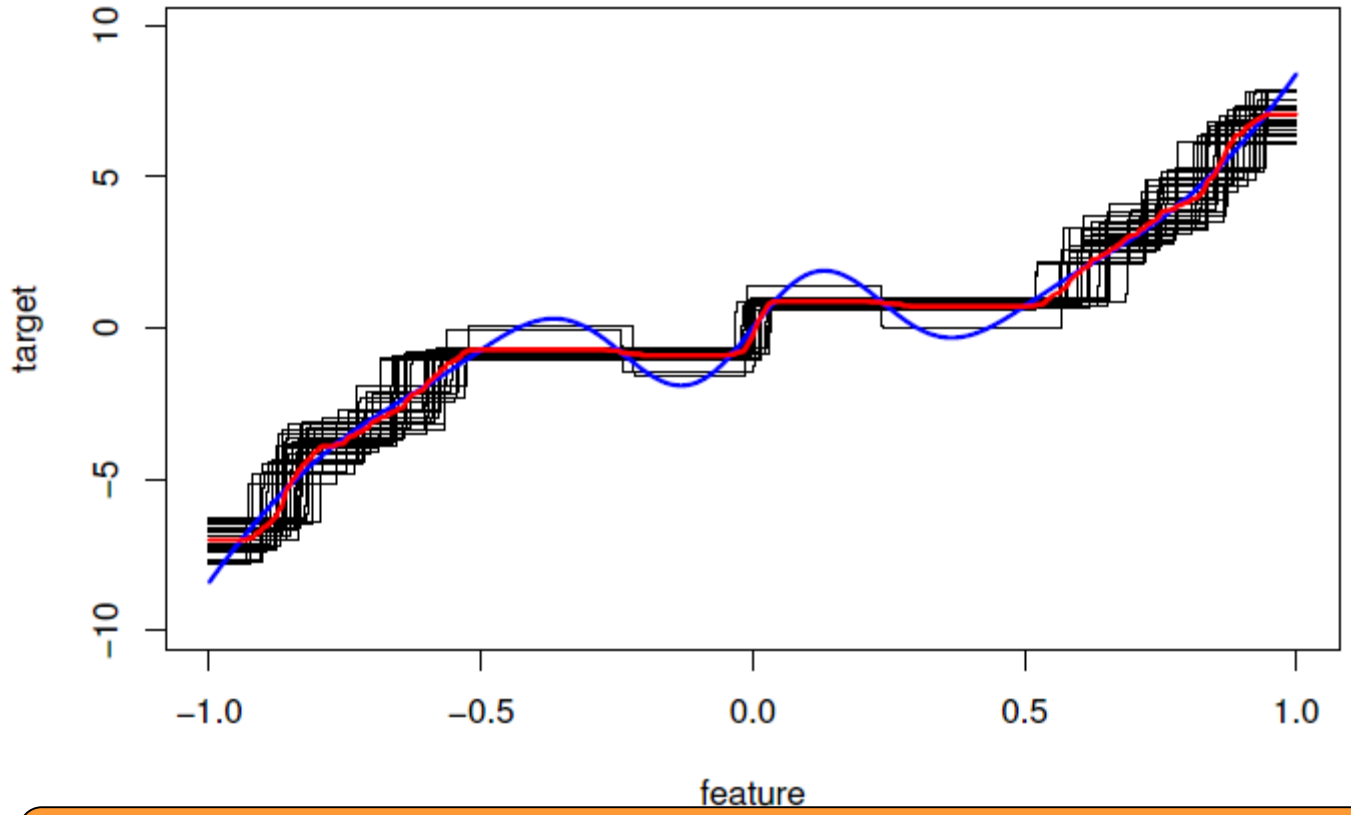
- **red** - fit



So, what happens if we run it multiple times?

Case 1b: Large Bias… Small Variance

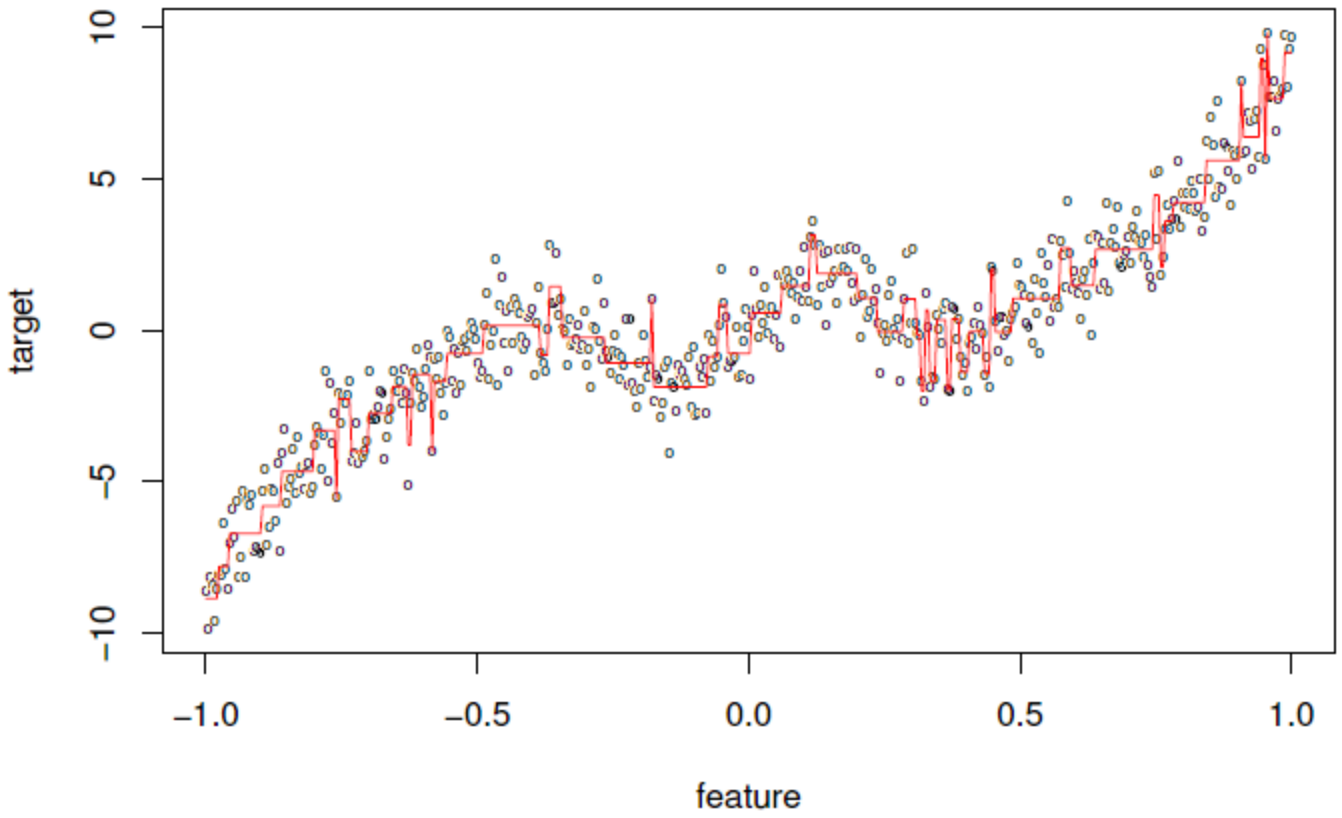
- **black** – multiple fits
- **red** – average of fits



poor ensemble fit

Case 2a: Small Bias…

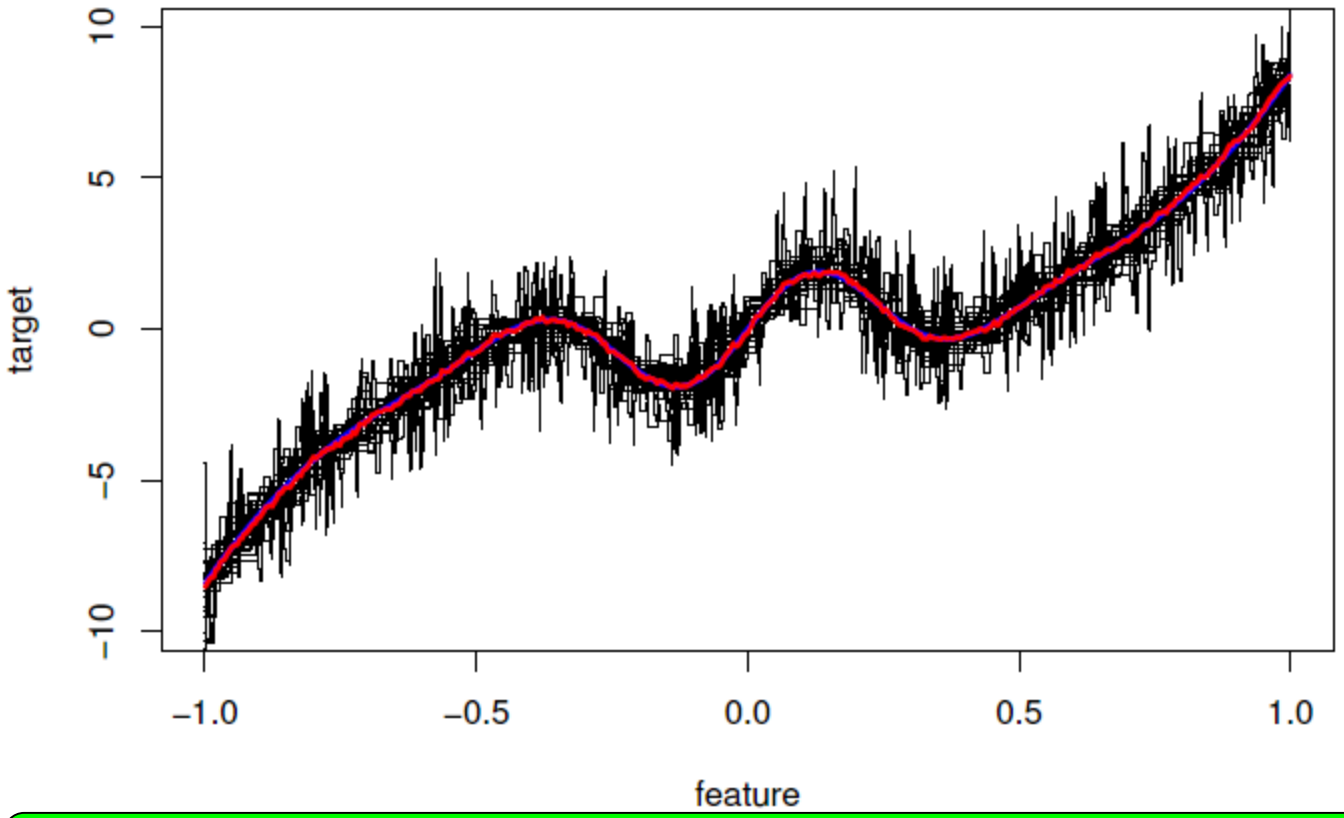
- **red** - fit



So, what happens if we run it multiple times?

Case 2b: Small Bias… …High Variance

- **black** – multiple fits
- **red** – average of fits



Fantastic ensemble fit

Bagging (Parallel Topology)

- **Bootstrap AGGregation**
  - create ***multiple*** bootstrap samples (samples of the same size, ***with replacement***)
  - train a classifier on each sample
  - combine output of classifiers by voting

- Good for unstable (with large variance) classifiers – otherwise different classifiers aren't very diverse.
  - e.g., good with decision trees
  - e.g., bad with naive Bayes

Key idea: reduce the variance by averaging many noisy but approximately unbiased models

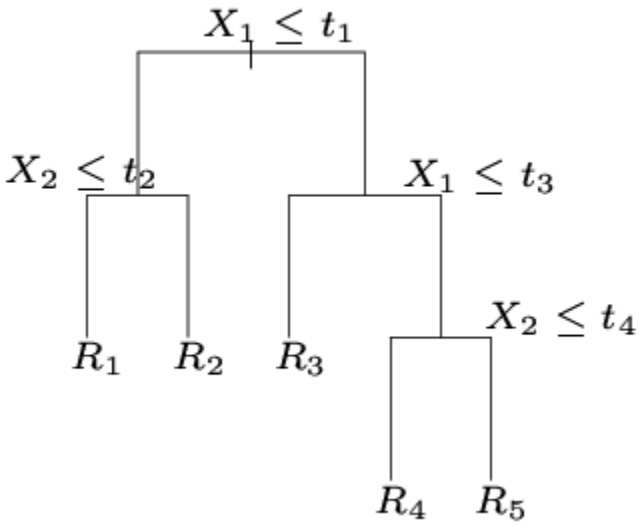
Boosting

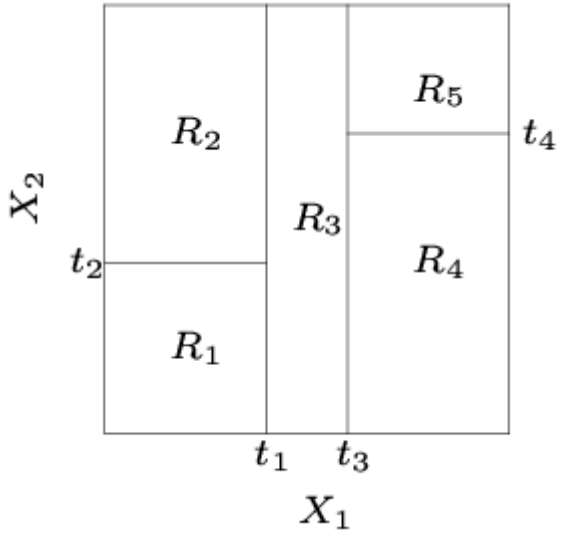
- Family of methods (several different approaches):
  - ***sequential production*** of classifiers
  - each classifier is dependent on the previous one
  - examples that are incorrectly predicted in previous classifiers are chosen more often or weighted more heavily
- Description of Boosting from: Robi Polikar (2009), Scholarpedia, 4(1):2776

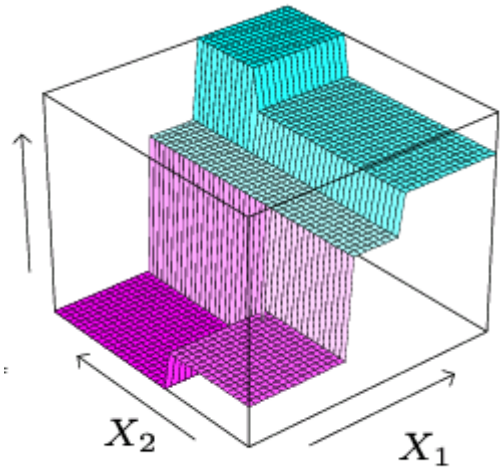
  "*... in boosting, resampling is strategically geared to provide the most informative training data for each consecutive classifier. In essence, each iteration of boosting creates three weak classifiers: the first classifier* ***C1 is trained with a random subset of the available training data****. The training data subset for the second classifier C2 is chosen as the most informative subset, given C1. Specifically,* ***C2 is trained on a training data only half of which is correctly classified by C1, and the other half is misclassified****. The third classifier* ***C3 is trained with instances on which C1 and C2 disagree****. The three classifiers are combined through a three-way majority vote.*"
- Good for relatively stable (with low variance) classifiers.
  - e.g., good with naive Bayes
  - e.g., bad with decision trees

Classification (Regression) Trees
(CART, C4.5,etc.)

- Binary Recursive Partitioning
  - binary: split parent node into two child nodes
    - *look at all features at each split, and choose best one*
  - recursive: each child node can be treated as parent node
  - partitioning: data set is partitioned into mutually exclusive subsets in each split
  - *prune tree to get good generalization*







CART Description

- Classification and Regression Tree (CART)
  - Algorithm defined by Breiman et al in 1984
  - Creates a binary decision tree to classify the data into one of 2^n linear regression models to minimize the Gini index for the current node c:

$$\mathbf{Gini(c) = 1 - \Sigma P_j^2 = \Sigma P_i P_j} \qquad \text{(for different i, j)}$$

where:

- n is the depth of the tree
- p_j is the probability of class j in node c
- Gini(c) measure the amount of "impurity" (incorrect classification) in node c
- For binary outcomes, Gini(c) has minima at 0 and maxima at 0.5

For regressions, CART minimizes sum of variance over all leaves

Classification (Regression) Trees (CART)

Performance Function

In a node m, representing region R_m with N_m observations.

$\hat{p}_{mk(m)}$ is the proportion of class k observations in node m =

$$\hat{p}_{mk(m)} = \frac{1}{N_m} \sum_{i \in R_m} I\left(y_i = k(m)\right)$$

Functions commonly used for classification:

Misclassification error $\frac{1}{N_m} \sum_{i \in R_m} I\left(y_i \neq k(m)\right) = 1 - \hat{p}_{mk(m)}$

Gini Index $\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$ measures the amount of "impurity

Cross-Entropy $-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$

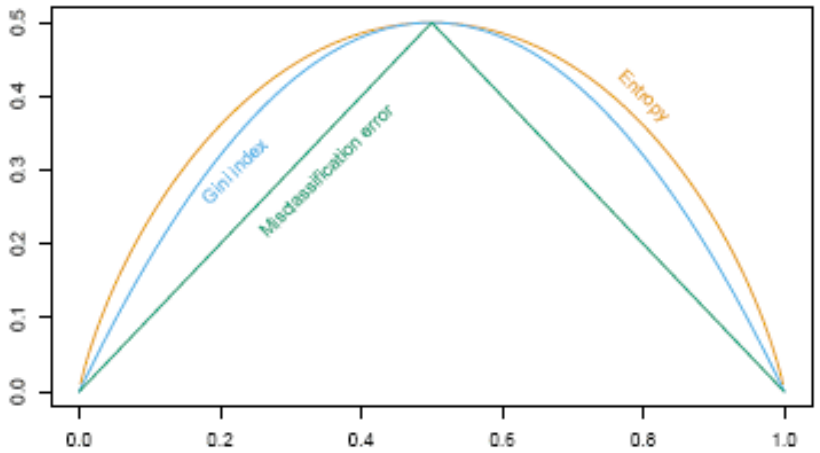
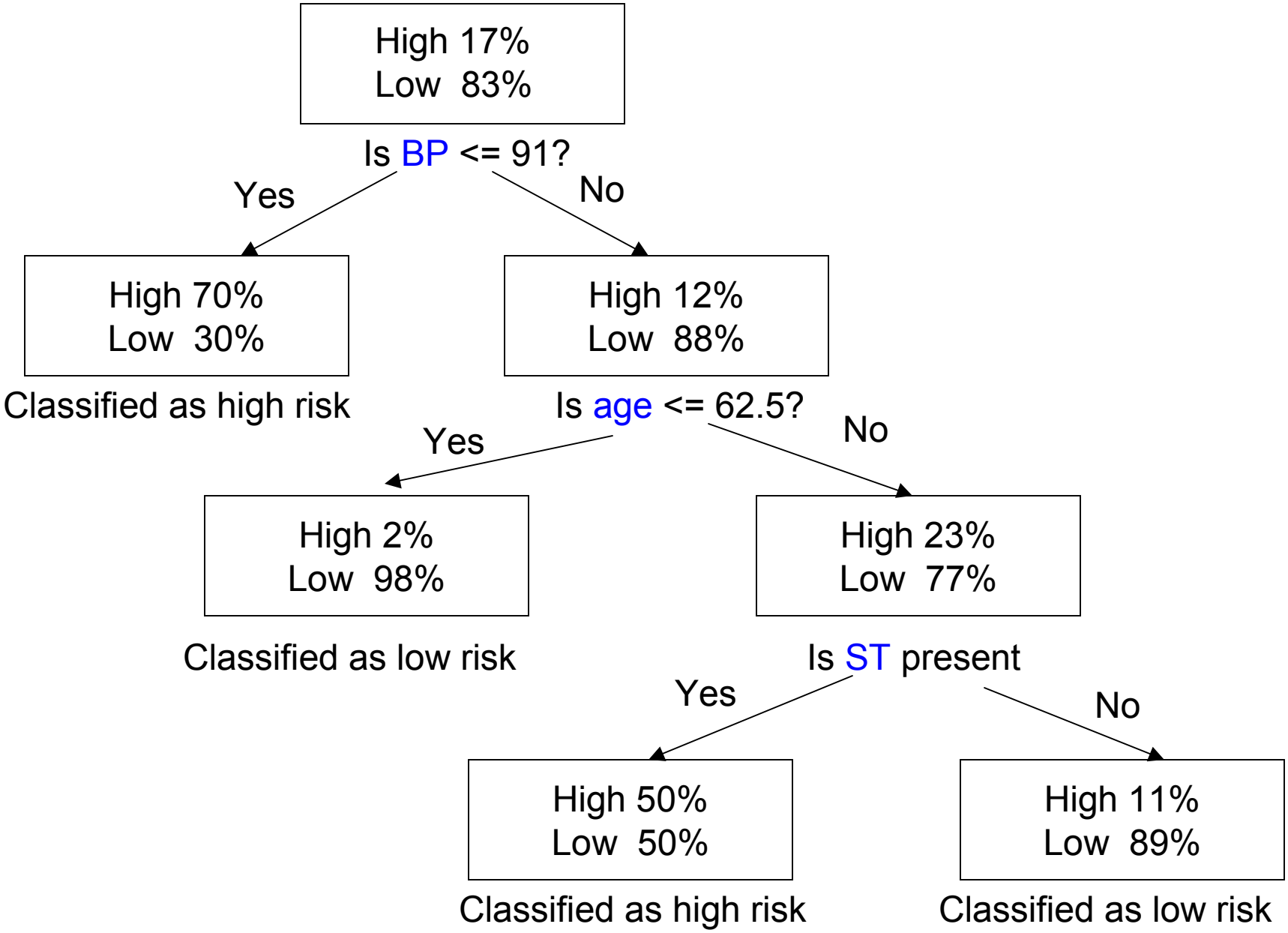


FIGURE 9.3. *Node impurity measures for two-class classification, as a function of the proportion p in class 2. Cross-entropy has been scaled to pass through* $(0.5, 0.5)$.

Source: **The Elements of Statistical Learning -** Data Mining, Inference, and Prediction (**Ch 9**)Trevor Hastie, Robert Tibshirani, Jerome Friedman, Springer

CART Construction



Random Forests

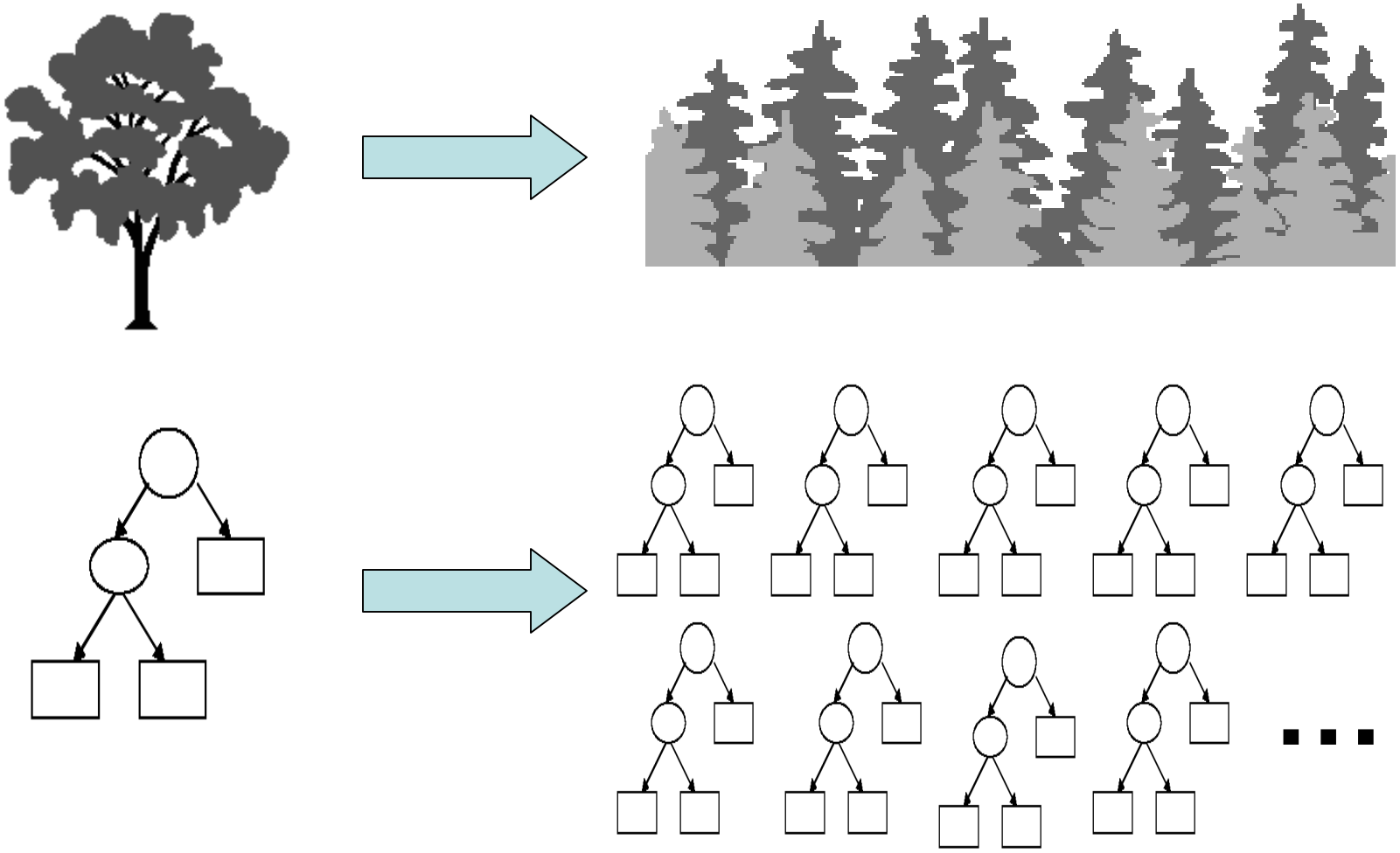
2. Random Forest (RF)

- **Bagging decision trees with "randomization injection".**
  - create multiple bootstrap samples (samples of the same size, with replacement)
  - train a decision tree on each sample
    - *at each node, select a random subset of* ***m*** *variables to split on*
    - *grow trees to maximum depth (i.e., no pruning)*
  - combine resulting trees by voting

- Properties of Random Forests (RF):
  - test set error rates (modulo a little noise) are monotonically decreasing and converge to a limit
    - i.e., there is no overfitting as the number of trees increases

- The key to accuracy is **low correlation** (high variance across trees, achieved by small values of ***m***) and low bias:
  - to maximize variance, randomness in variable selection is introduced
  - to minimize bias, trees are grown to maximum depth.

RF Construction



Growing Each Tree

Each tree is grown as follows:

1. If the number of cases in the training set is N, sample N cases at random - ***with replacement*** - from the original data. This sample will be the training set for growing the tree

2. If there are ***p*** input variables, a number ***m<<p*** is specified such that at each node, ***m*** variables are selected at random out of the M
   - The best split on these ***m*** is used to split the node.
   - The value of ***m*** is held constant during the forest growing

3. Each tree is grown to the largest extent possible. There is **no** pruning

Prediction by plurality voting

The forest consists of B trees.

- To classify a new object from an input vector, we put the input vector down each of the trees in the forest.
- Each tree gives a classification, and we say the tree "votes" for that class
- The forest chooses the classification having the most votes (over all the trees in the forest).
  - **Class prediction:** Each tree votes for a class; the predicted class C for an observation is the plurality,

    $\max_C \Sigma_k [f_k(\mathbf{x},\mathbf{T}) == C]$

  - **Regression random forest:** predicted value is the average prediction

Random Forest Algorithm

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Source: T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning*, **Chapter 15**, Springer

Random Forest: Classification and Regressions

RF for Classification and Regressions

When used for **classification**, a random forest obtains a class vote from each tree, and then classifies using majority vote.

When used for **regression**, the predictions from each tree at a target point *x* are simply averaged.

Recommendations default values of *m and MinNodeSize*

Classification: $m = \lfloor \sqrt{p} \rfloor$

minimum node size : 1

Regression : $m = \lfloor p/3 \rfloor$

minimum node size : 5

Out-of-bag (oob) error estimate

- In RF, there is ***no need for cross-validation*** or a separate test set to get an unbiased estimate of the test set error. It is estimated internally, during the run, as follows:
  - Each tree is constructed using a **different bootstrap sample** from the original data. About **one-third** of the cases **are left out** of the bootstrap sample and not used in the construction of the kth tree.
  - Put each case left out in the construction of the k^{th} tree down the k^{th} tree to get a classification. In this way, a test set classification is obtained for each case in about one-third of the trees.
  - At the end of the run, take **j** to be the class that received most of the votes **every time case n** was oob.
  - The proportion of times that j is not equal to the true class of n **averaged over all cases** is the oob error estimate. This has proven to be unbiased in many tests.

Source: www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

Variable Importance

Random forests also use the oob samples to construct a variable importance measure, apparently to measure the **prediction strength of each variable**.

(1) When the b^{th} tree is grown, the **oob samples** are passed down the tree, and the prediction accuracy is recorded.

(2) Then the values for the j^{th} variable are randomly permuted in the oob samples, and the accuracy is again computed.

(3) The **decrease in accuracy** as a result of this permuting is averaged over all trees, and is used as a **measure of the importance of variable j** in the random forest.

These are expressed as a percent of the maximum.

The randomization effectively voids the effect of a variable, much like setting a coefficient to zero in a linear model.

Source: T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning*, **Chapter 15**, Springer

Variable Importance

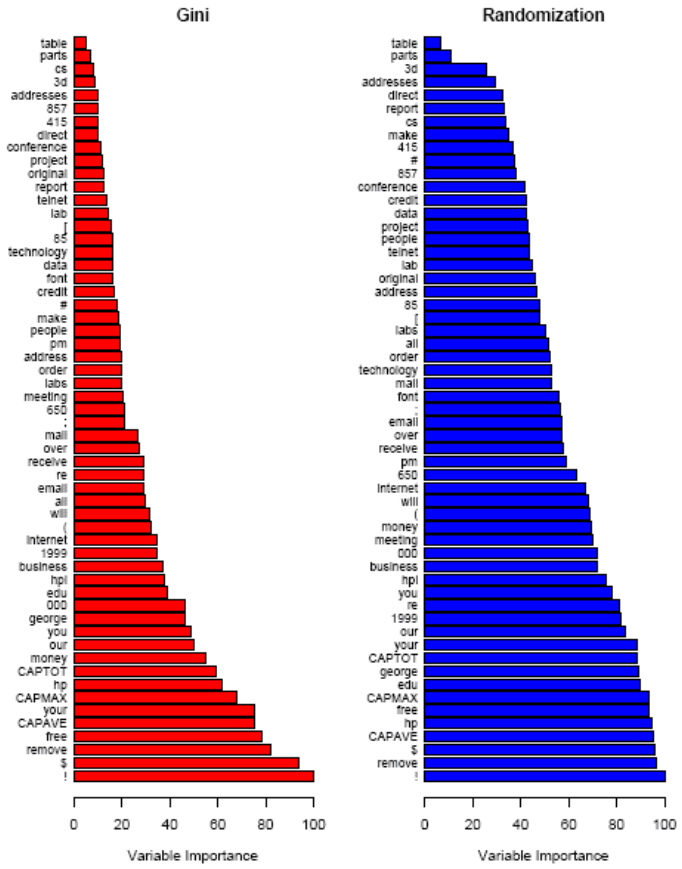


FIGURE 15.5. *Variable importance plots for a classification random forest grown on the* **spam** *data. The left plot bases the importance on the Gini splitting index, as in gradient boosting. The rankings compare well with the rankings produced by gradient boosting (Figure 10.6 on page 354). The right plot uses* OOB *randomization to compute variable importances, and tends to spread the importances more uniformly.*

Source: T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning*, **Chapter 15**, Springer

Forest Error Rate

The forest error rate depends on two things:

- The ***correlation*** between any two trees in the forest. Increasing the correlation increases the forest error rate.
- The ***strength*** of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.

Reducing ***m*** reduces both correlation and strength.

Increasing it increases both.

Somewhere in between is an "optimal" range of ***m*** - usually quite wide.

Notes on parameter ***m***

- ***m*** is the only adjustable parameter to which random forests is somewhat sensitive.
- Using the out of bag (OOB) error rate a value of m in the range can quickly be found.
- Typically this range is quite broad. Good default values for ***m***

Some Properties of RF

- Bias:
  - The bias of RF is the same of any of the individual Sampled trees
  - Prediction improvements in RF are solely a result of **variance reduction**
- RF accuracy is as good as Adaboost (and sometimes better.)
- It's relatively robust to outliers and noise.
- It's faster than bagging or boosting.
- It gives useful internal estimates of error, strength, correlation and variable importance.
- It's simple and easily parallelized.

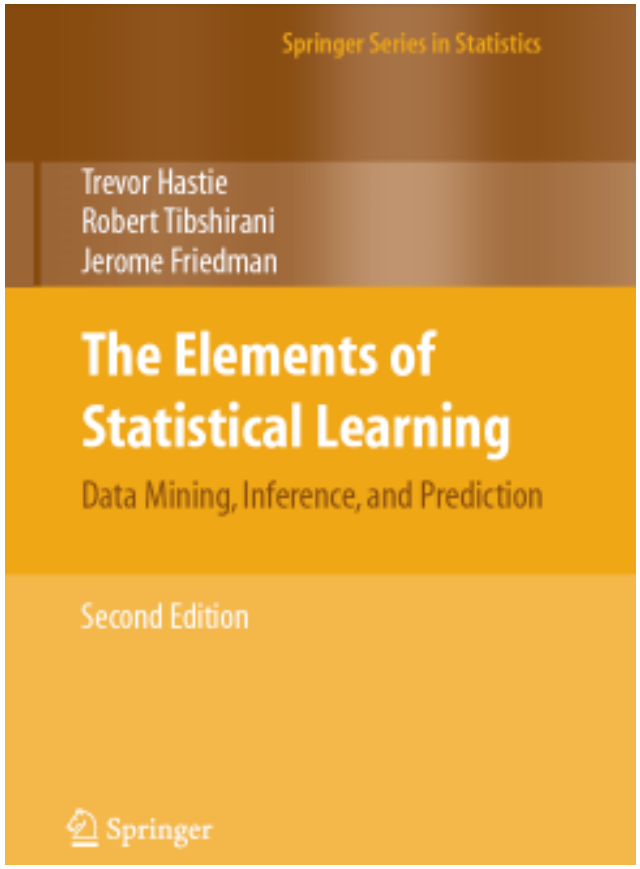
RF = Bagging + Random Subspace Method (Ho 1998) at every node

Random Forest Sources

- T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning*, **Chapter 15,** Springer, 2009

  http://www-stat.stanford.edu/~tibs/ElemStatLearn/index.html
  http://www-stat.stanford.edu/~hastie/Papers/ESLII.pdf

- L. Breiman (2001). Random forests. *Machine Learning* 45(1), 5-32.



Random Forest Sources (cont.)

Free software implementations of random forests.

Google Code implementations (most current)

- http://code.google.com/p/randomforest-matlab/
- http://code.google.com/p/fast-random-forest/

randomForest package **in R**, maintained by Andy Liaw, available from the CRAN website (cran-r.project.org)

This allows both split-variable selection, as well as sub-sampling.

Adele Cutler maintains a random forest website **http://www.math.usu.edu/ ~adele/forests/**

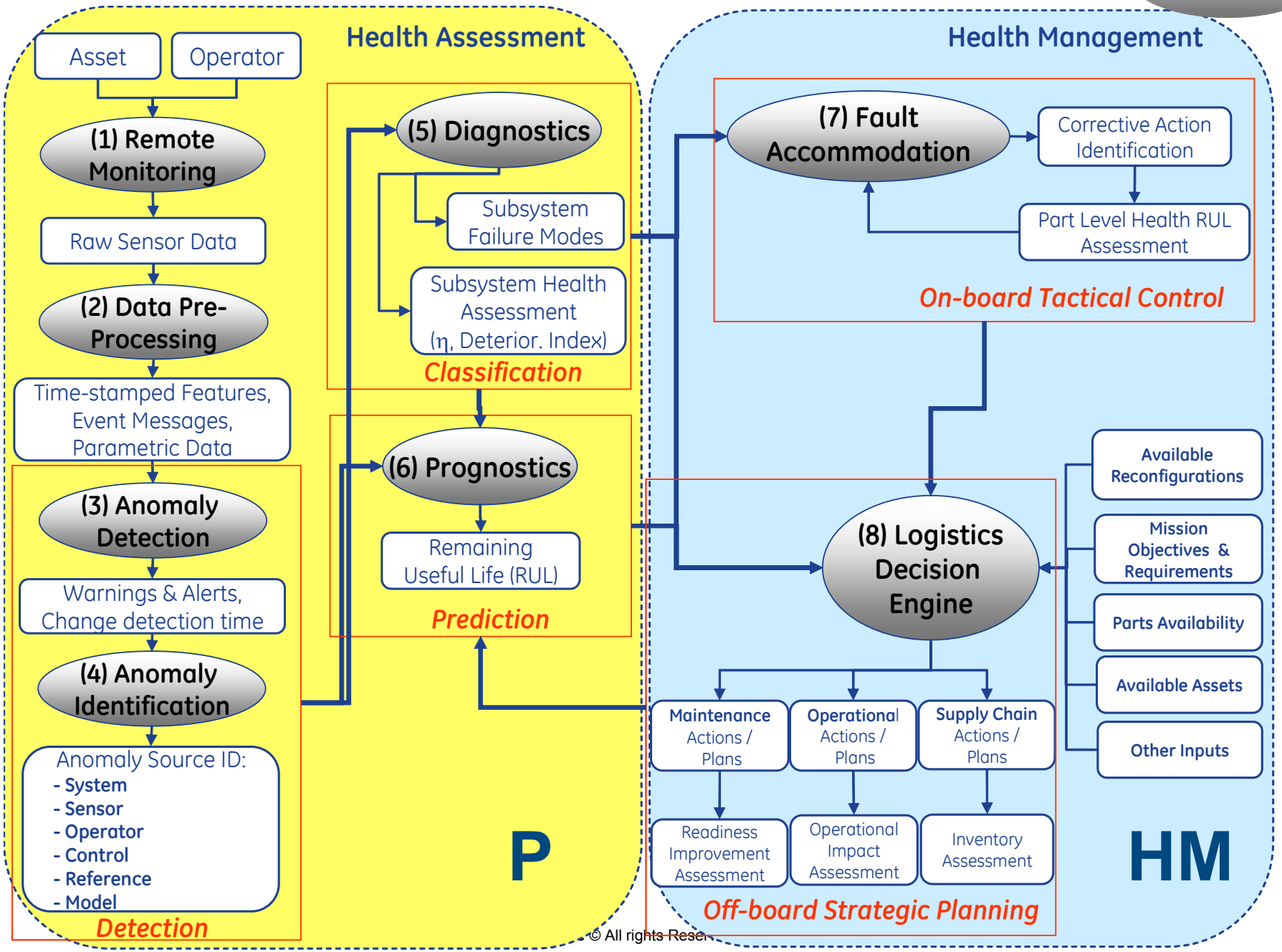
where (as of August 2008) the software written by Leo Breiman and Adele Cutler is freely available

The Weka machine learning archive **http://www.cs.waikato.ac.nz/ml/weka/**

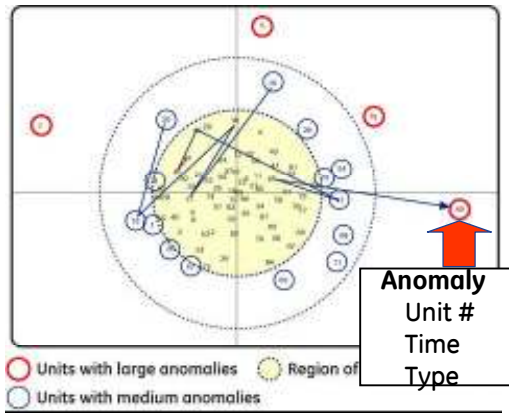
at Waikato University, New Zealand, offers a free java implementation of random forests.

3 PHM Models

PHM: A Comprehensive View



Focus on the P of PHM: From Anomaly Detection to Diagnostics and Prognostics



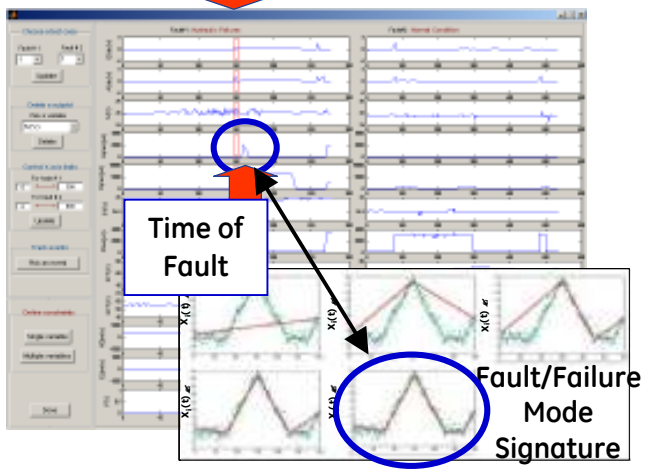
Diagnostics

Subsystem Failure Mode(s)

Subsystem Health Assessment (η, Deterior. Index)

Failure Mode Diagnostics

Time of Anomaly



Anomaly Detection

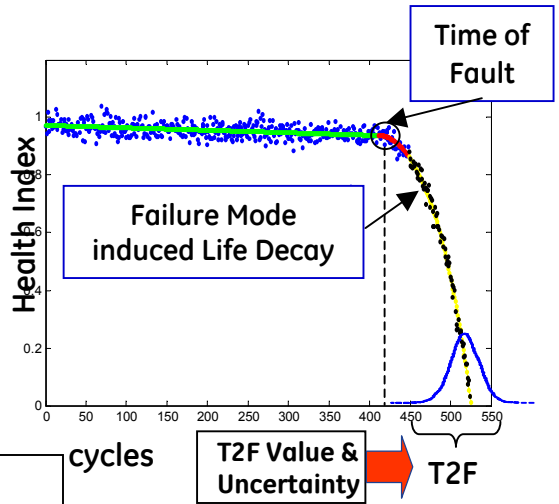
Time and Type of Anomaly

Time of Early Warning

Prognostics

Time to Failure Threshold (T2F)

T2F Prediction



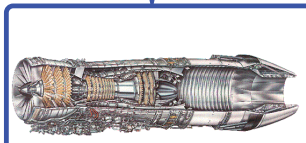
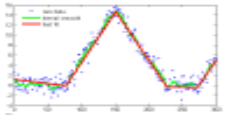
PHM Models

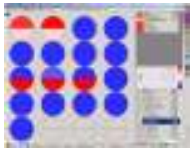
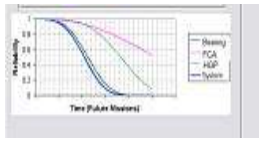
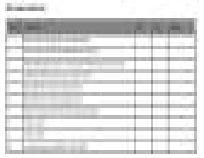
- **Anomaly Detection: 1-class classification**
- **Diagnostics: Multi-class classification**
- **Prognostics: Prediction**

PHM Capabilities and Enabling Technologies

Stage 1: AD + RM&D | ***Stage 2: Prognostics*** | ***Stage 3: Control & Optimization***

Data Acquisition → **Anomaly Detection** → **Diagnostics** → **Prognostics** → **Control / Advisory Generation** → **Optimization** → **Feedback & Learning**


Functions/Enabling Technologies

On-platform sensors
Data recorders
Off-platform sensors
Inspection / NDE
Data repository
Data de-noising
Data smoothing
Data compression
Data transmission

Features
Generation/Selection
Manual/Automated
Anomaly Detection (AD)
Normal / Abnormal structures (trajectory, clusters, states, patterns)
Anomaly Resolution / Id.
Resolving sensors faults, op. transients, model Faults, system faults

State Interpretation
Manual/Automated
Classification
Fault mode Id.
Health Assessment

RUL Prediction
Automated forecast of:
- Deterioration level
- RUL

Health Assessment
Projection
Lifing
Fault propagation

On-board Fault Accommodation
Off-board DSS

Maintenance Optimization
Workscope (build level)
Shop scheduling
Logistics Optimization
Supply Chain Mgmt
Operational Optimization
Asset/mission allocation

Performance metrics dashboard
Ground truth acquisition
Structured feedback

Uncertainty Management

Sensor Fusion
Extended KF

Feature Fusion
Hierarchy of local AD models
Fusion of global (or hierarchies of) AD models

Classifier Fusion

Predictors Fusion
Reduce bias & variance

Uncertainty in DSS

Robust optimization
Functional approximations error bounds; extrapolations
MCDM: Tolerance/fuzziness in dominance

Evaluation Uncertainty
Stochastic simulations (confidence interval)
Noisy ground truth (consistency checks)

Model Lifecycle Management

On-platform sensors
lifecycle/maintenance
Off-platform sensors
lifecycle/maintenance
Obsolescence challenges

Anomaly Detectors update
Update training sets
Recompile detectors
Re-assess meta-level (thresholds & costs) to re-tune fusion

Diagnostics Models updates
Update training sets
Recompile classifiers
Re-assess meta-level (misclassification costs) to re-tune fusion

Prognostics Models updates
Update training sets
Recompile predictors
Re-assess meta-level (misclassification costs) to re-tune fusion

DSS Models updates
Update DSS models

Optimization Models updates
Update training sets
Re-assess fitness (cost) function parameters
Evolve/Compute new optimization models

(Semi-) Automate new requirements acquisition
Learning vs adaptation
-Testing adaptivity
-Integr. adaptations in new releases & sharing
-Tracking region of competenc
Automated spec gen.
-Autonomous V&V

4 Fusion for PHM Models

4.1. Case Study 1:

Interpolation Among ~~Selection of~~ ^ Complementary AD Models

Reference: "P. Bonissone, X Hu, R. Subbu (2009) A Systematic PHM Approach for Anomaly Resolution: A Hybrid Neural Fuzzy System for Model Construction, Proc. PHM 2009, San Diego, CA, Sept 27-Oct 1, 2009. - [GE GR Technical Report, 2009, GRC839, Sept. 2009]

4.1 Anomaly Detection (AD) - 1class classification

- SC Technologies for AD
- Example of AD for Aircraft Engines
- Design:
  - Offline MH (EA)
- Run-time:
  - Online MH (Fuzzy Sup.)
  - Object-level (AANN)
- Evolutionary Search for Designing a F-IBM
- Experiments

Soft Computing Technologies for AD: Fuzzy Systems

Approximate Reasoning

Functional Approximation/ Randomized Search

Probabilistic Models

Multivalued & Fuzzy Logics

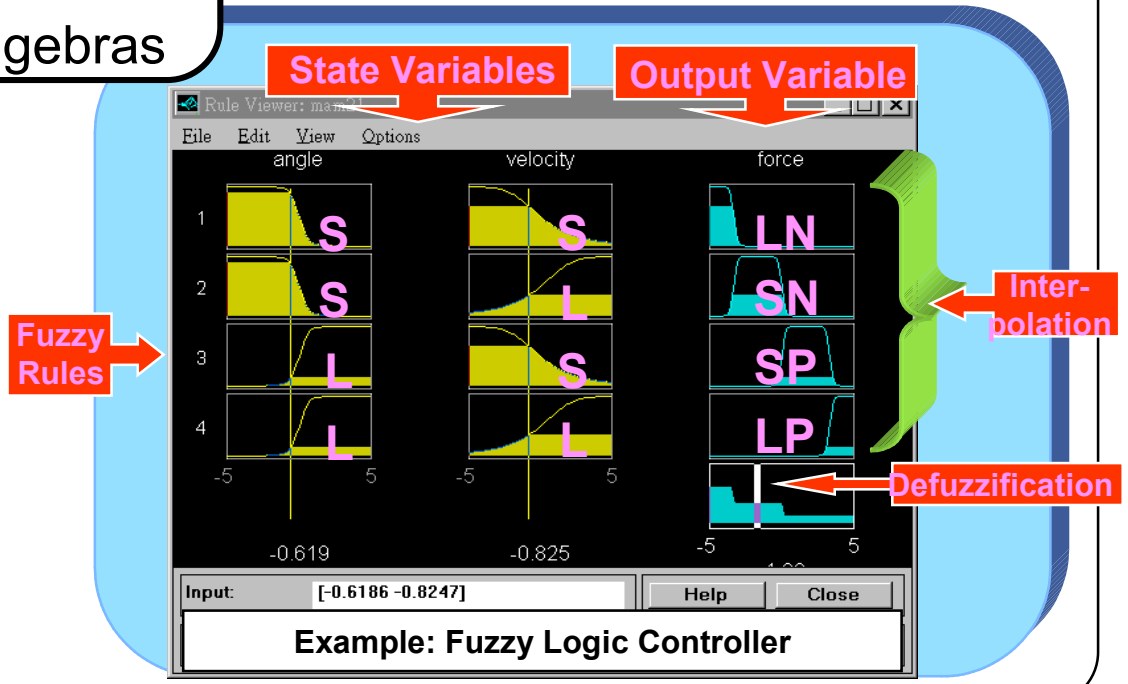
Neural Networks

Evolutionary Algorithms

Fuzzy Systems

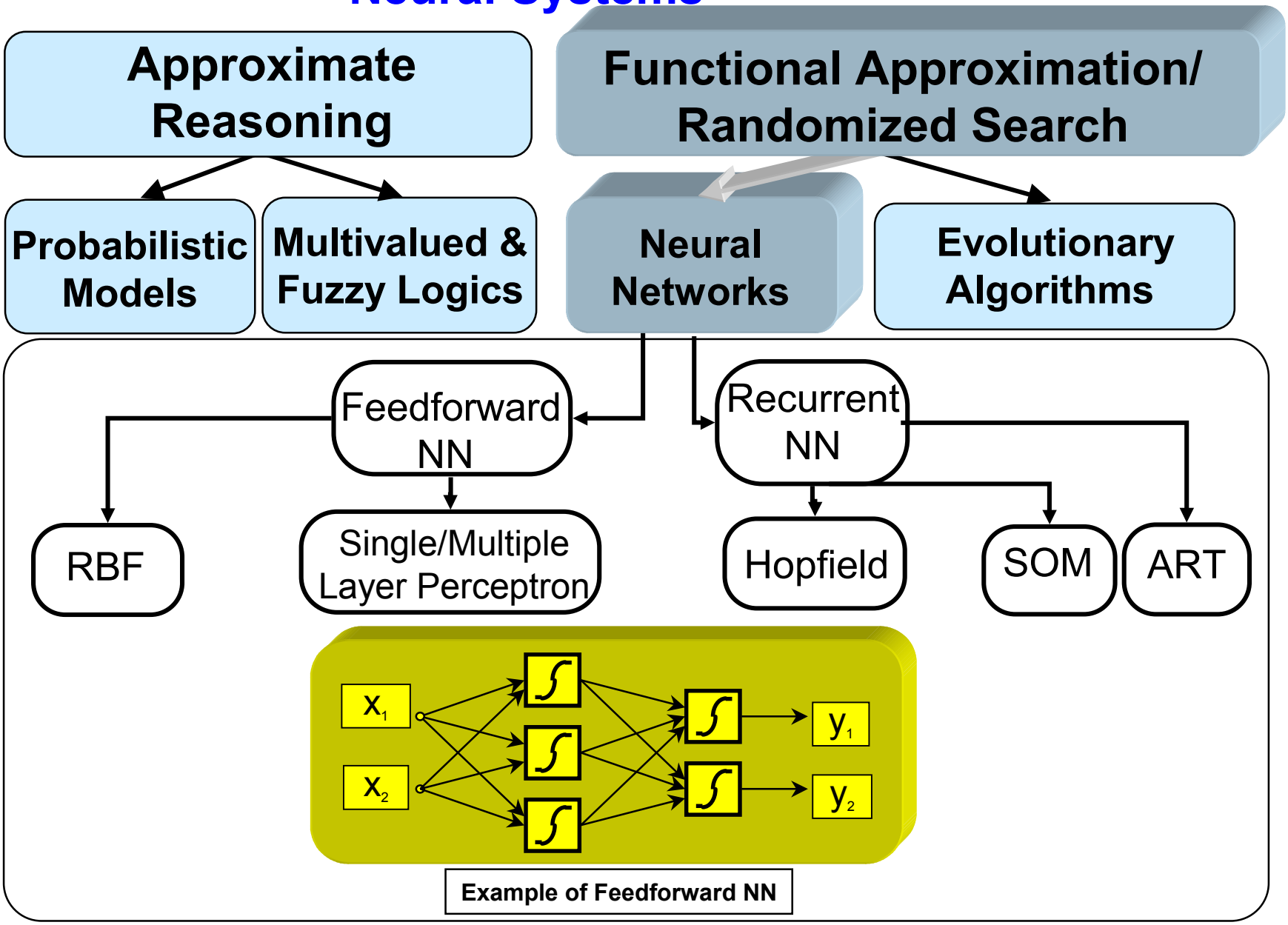
Multivalued Algebras

Fuzzy Logic Controllers



Example: Fuzzy Logic Controller

Soft Computing Technologies for AD
Neural Systems



Example of Feedforward NN

Soft Computing Technologies for AD
Evolutionary Systems

Approximate Reasoning

Functional Approximation/ Randomized Search

Probabilistic Models

Multivalued & Fuzzy Logics

Neural Networks

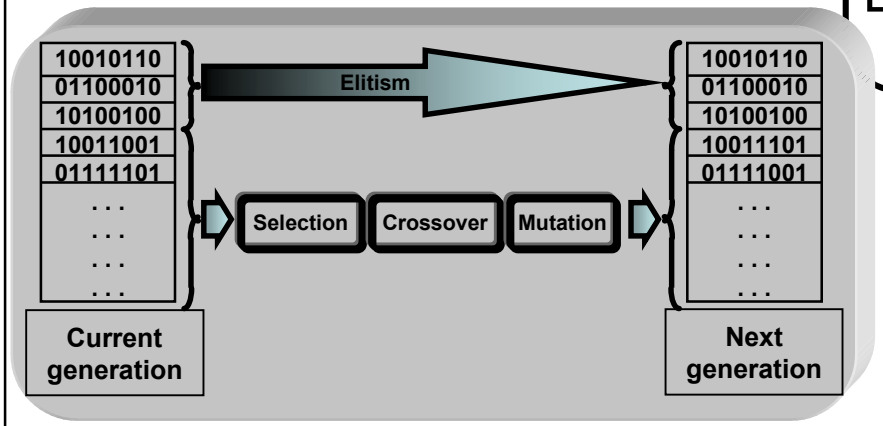
Evolutionary Algorithms

Evolution Strategies

Genetic Algorithms

Evolutionary Programs

Genetic Progr.



Example of Binary- Encoded GA

Soft Computing Technologies for AD
Hybrid Systems

Approximate Reasoning

Functional Approximation/ Randomized Search

Probabilistic Models

Multivalued & Fuzzy Logics

Neural Networks

Evolutionary Algorithms

Fuzzy Controller

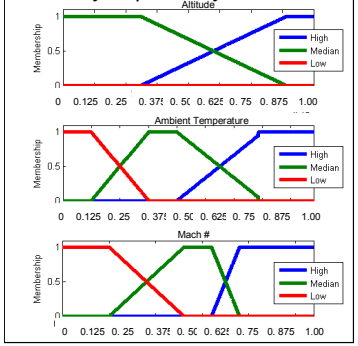
Auto-Associative NN

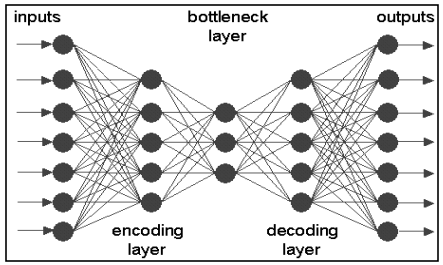
Genetic Algorithm

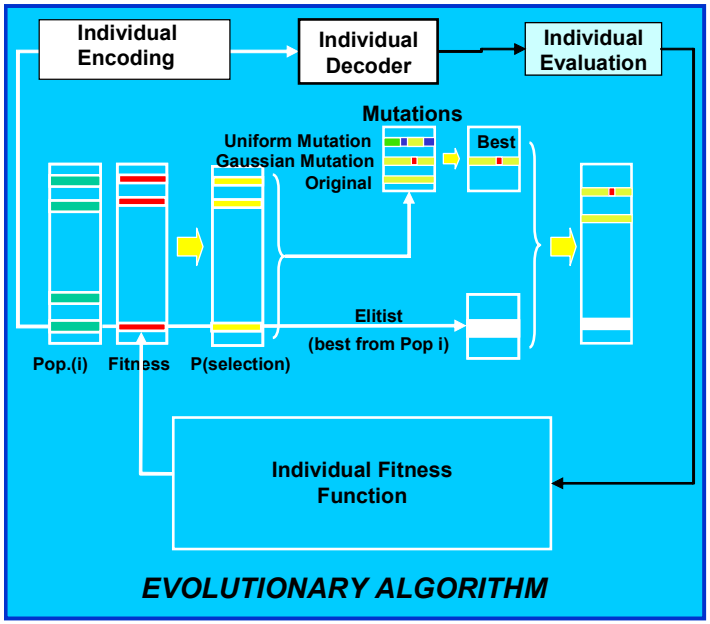
Fuzzy Supervisory Rule Set

State Variables		Altitude	Amb. Temp.	Mach #	Model #
RULES	R1	High $V_{1,1}(Input_1)$	High	High	AANN-1
	R2	High $V_{2,1}(Input_1)$	Medium	Medium	AANN-2
	R3	Medium $V_{3,1}(Input_1)$	Low	Low $V_{3,3}(Input_3)$	AANN-3

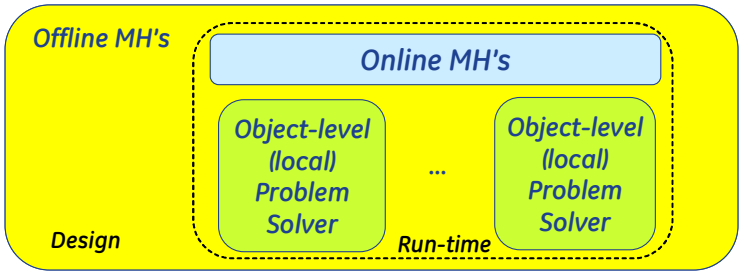
Fuzzy Supervisory Term Set







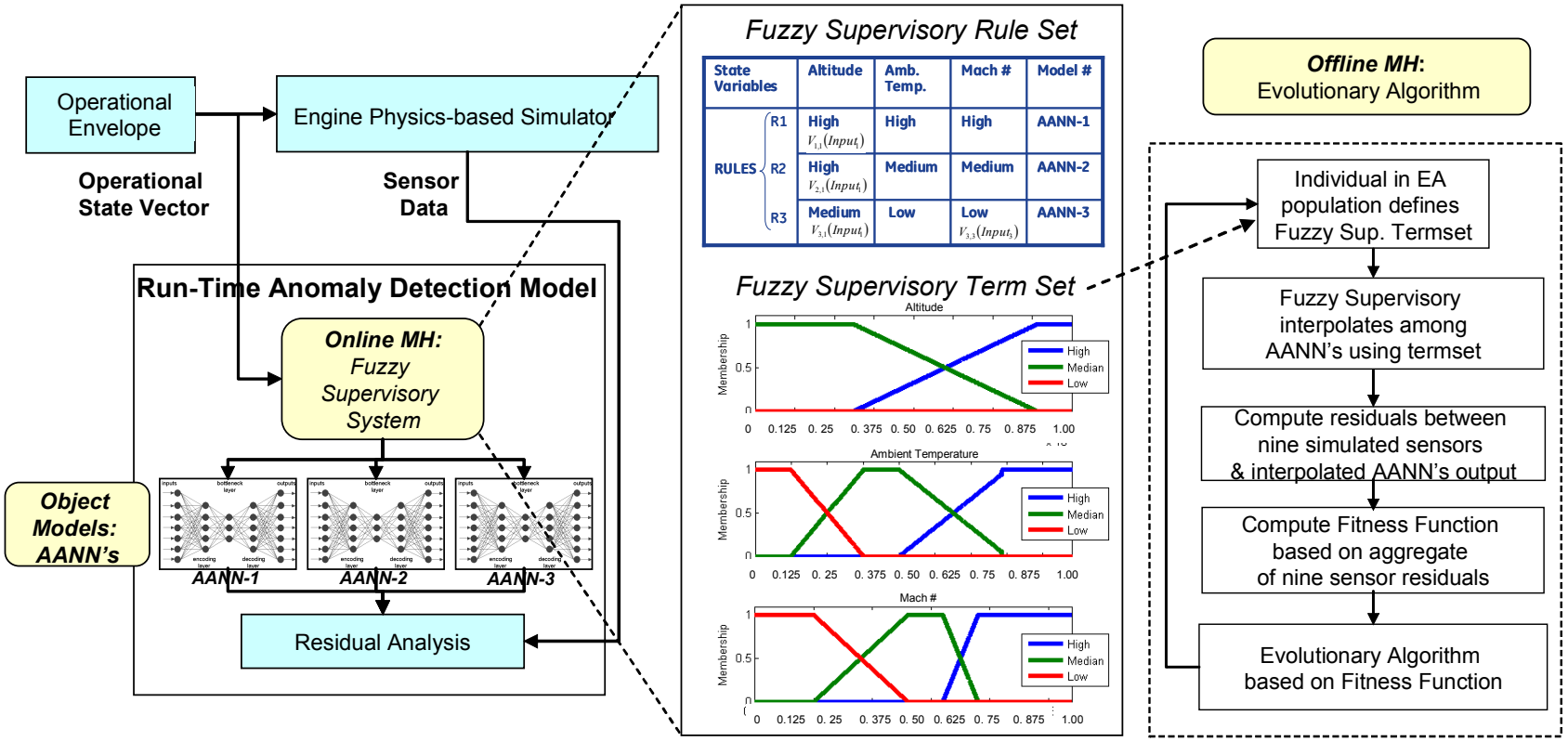
SC Techniques for *Offline MH's, Online MH's, & Object-level Models*



Problem Instance	**Problem Type**	**Model Design (*Offline MH's*)**	**Model Controller (*Online MH's*)**	**Object-level models**	**References** *[As listed in Bonissone 2010]*
Anomaly Detection (System)	Classification	Model T-norm tuning	Fuzzy Aggregation	*Multiple Models:* SVM, NN, Case-Based, MARS	[24]
Anomaly Detection (System)	Classification	Manual design	Fusion	*Multiple Models:* Kolmogorov Complexity, SOM. Random Forest, Hotteling T2, AANN	[25, 26]
Anomaly Detection (Model)	Classification & Prediction	EA tuning of fuzzy supervisory termset	Fuzzy Supervisory	*Multiple Models:* Ensemble of AANN's	[27, 28]
Insurance Underwriting: Risk management	Classification	EA	Fusion	*Multiple Models:* NN, Fuzzy, MARS,	[29, 30]
Load, HR, NOx forecast	Prediction	Multiple CART trees	Fusion	*Multiple Models:* Ensemble of NN's	[31, 34]
Aircraft engine fault recovery	Control/Fault Accommodation	EA tuning of linear control gains	Crisp supervisory	*Multiple Models (Loop):* SVM + linear control	[14]
Power plant optimization	Optimization	Manual design	Fusion	*Multiple Models (Loop):* MOEA + NN's	[32, 33, 34]
Flexible mfg. optimization	Optimization	Manual design	Fuzzy supervisory	EA	[10, 35]

Anomaly Detection

4.1 AD Model Design



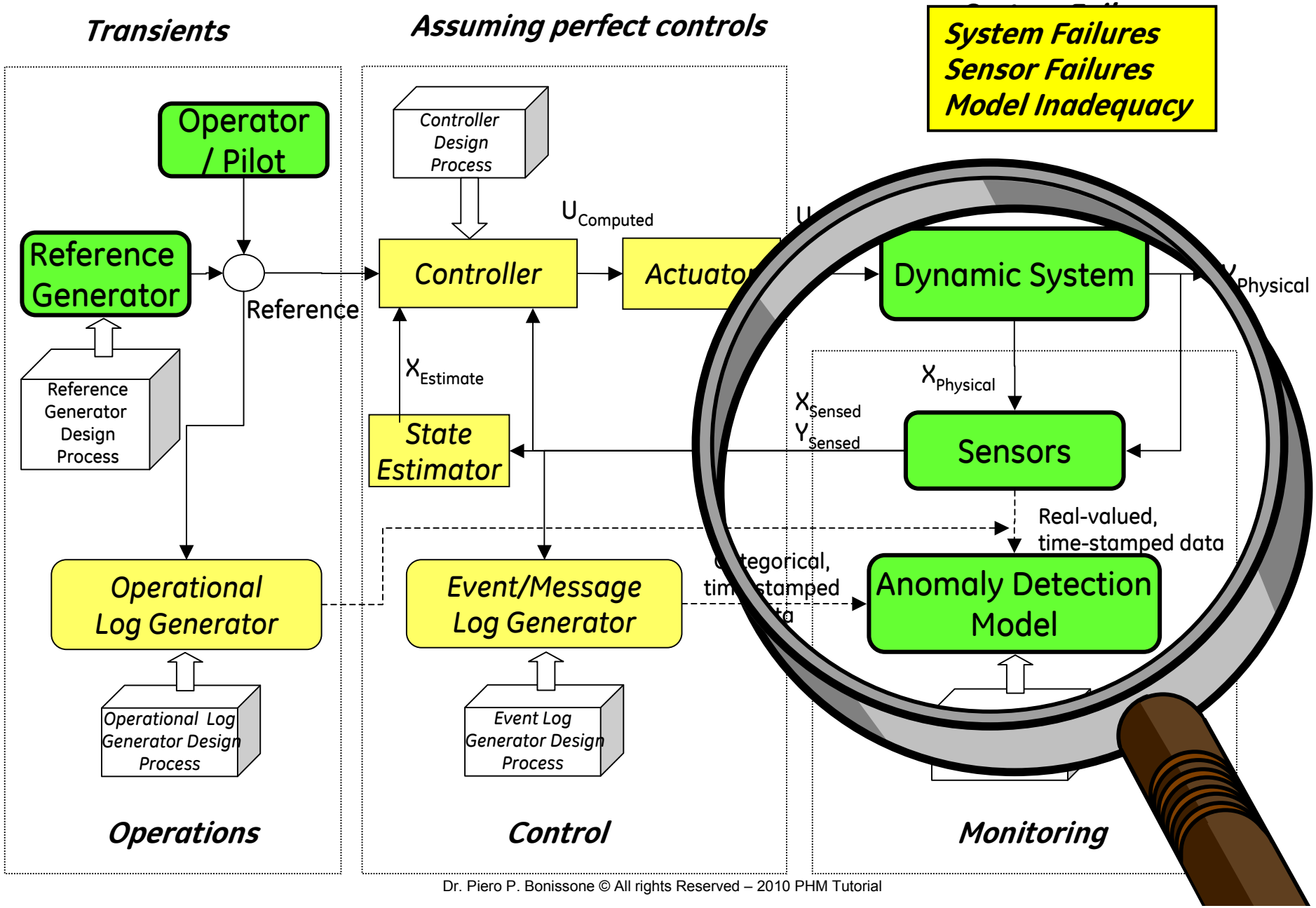
Problem Instance	Problem Type	Model Design (Offline MH's)	Model Controller (*Online MH's*)	Object-level models
Anomaly Detection	1-class Classification	EA tuning of fuzzy supervisory termset	Fuzzy Supervisory	*Multiple Models:* Ensemble of AANN's

Reference: "A Systematic PHM Approach for Anomaly Resolution: A Hybrid Neural Fuzzy System for Model Construction", Proc. PHM 2009, San Diego, CA, Sept 27-Oct 1, 2009. - [GE GR Technical Report, 2009, GRC839, Sept. 2009]

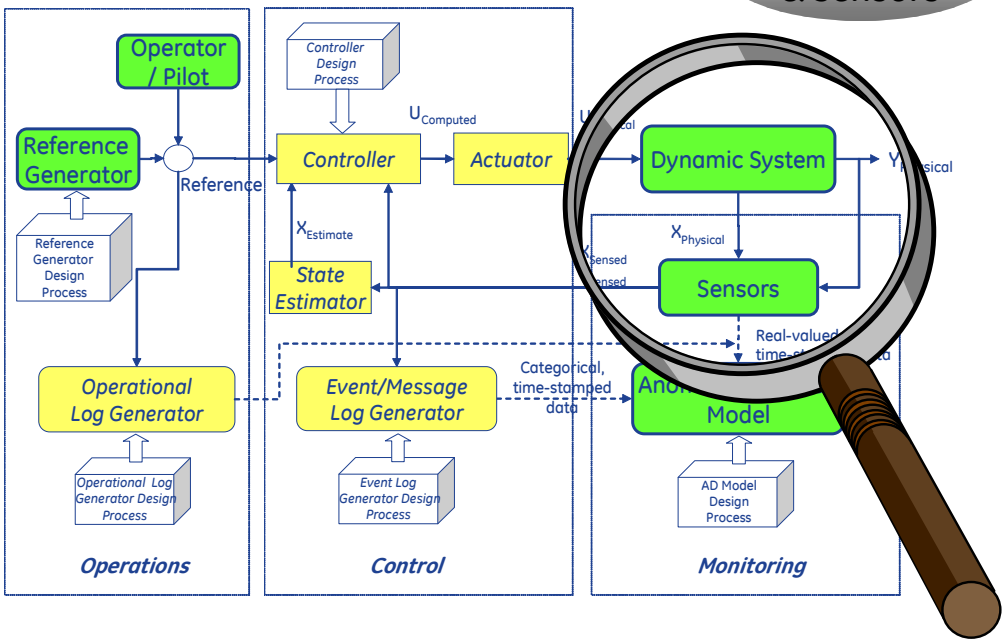
Ensemble of Complementary AD models with Simulated GE90 Aircraft Engines (Detailed Description)

- Potential Sources of Anomalies
- Dynamic System and Sensors Simulation
- AD Model (AANN)
- Experiment Setup
- Segmentation of the Operating Space
- Experiments
  - 1st - 3 local models
  - 2nd - 1 Global Model
  - 3rd - 3 local Models + Supervisory Model

Potential Sources of Anomalies



Dynamic System: Simulated Aircraft Engine [GE 90]



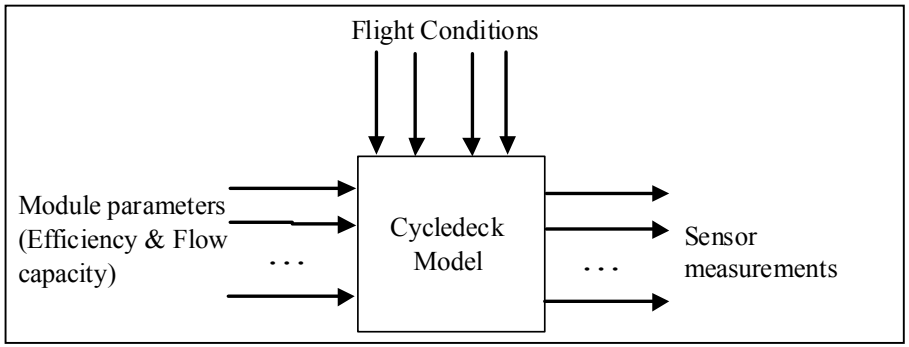
Physics-Based Simulation

–**CLM:** Component Level Model is a physics-based thermodynamic model widely used to simulate the performance of a commercial aircraft engines.

–**Flight Regime:** Flight conditions, such as altitude, Mach number, ambient temperature, and engine fan speed, and a large variety of model parameters, such as module efficiency and flow capacity are inputs to the CLM

–**Outputs:** CLM's outputs are the values for **pressures, core speed and temperatures** at various locations of engine, which simulate sensor measurements.

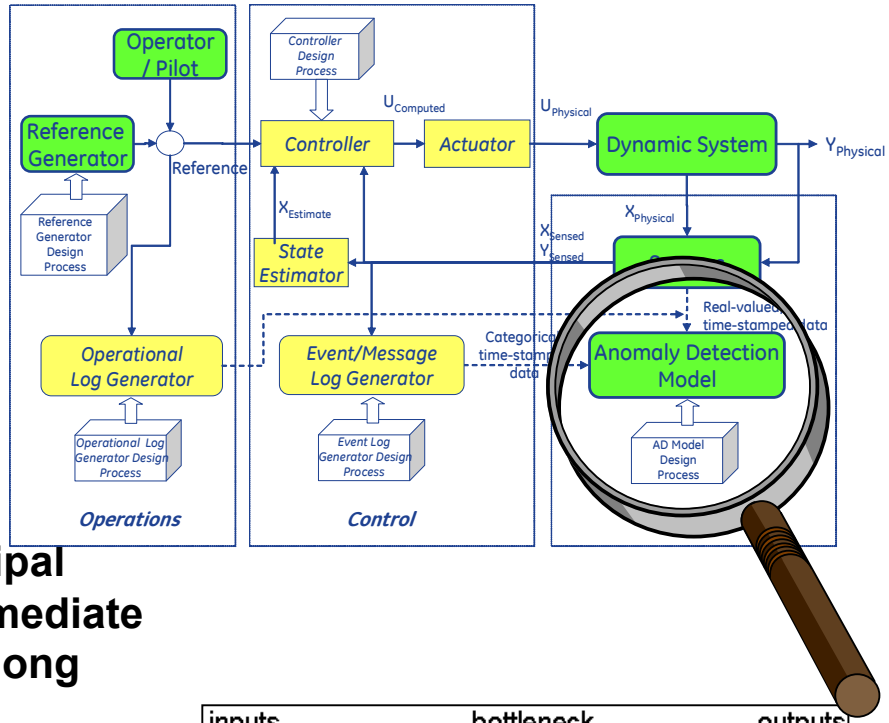
–**Noise**: Realistic values of sensor noise can be added after the CLM calculation.



Basic AD Model: Auto-Associative Neural Network

Rationale

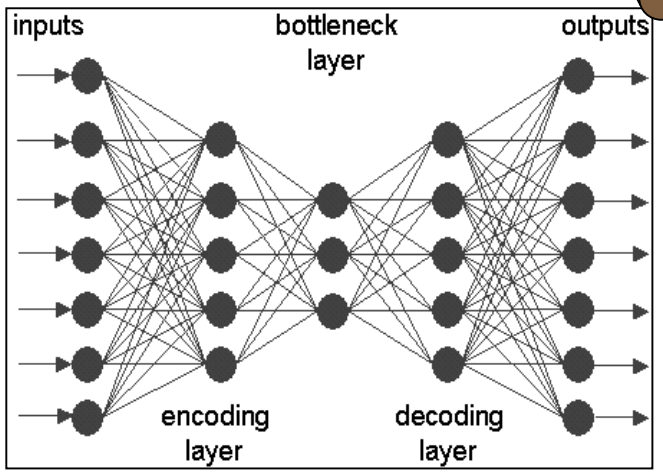
The Auto-Associative Neural Network (AANN) leverages covariance information like other approaches (SRC and T2). The AANN also produces sensor estimated values to replace the ones generated by faulty sensors. This approach provides a better discrimination between sensor faults and system component faults.



Definition/Properties

AANN computes the largest Non-Linear Principal components (NLPCA) - the nodes in the intermediate layer – to identify and remove correlations among variables.

NLPCA uncover both linear and nonlinear correlations, without restriction on the type of the nonlinearities present in the data.



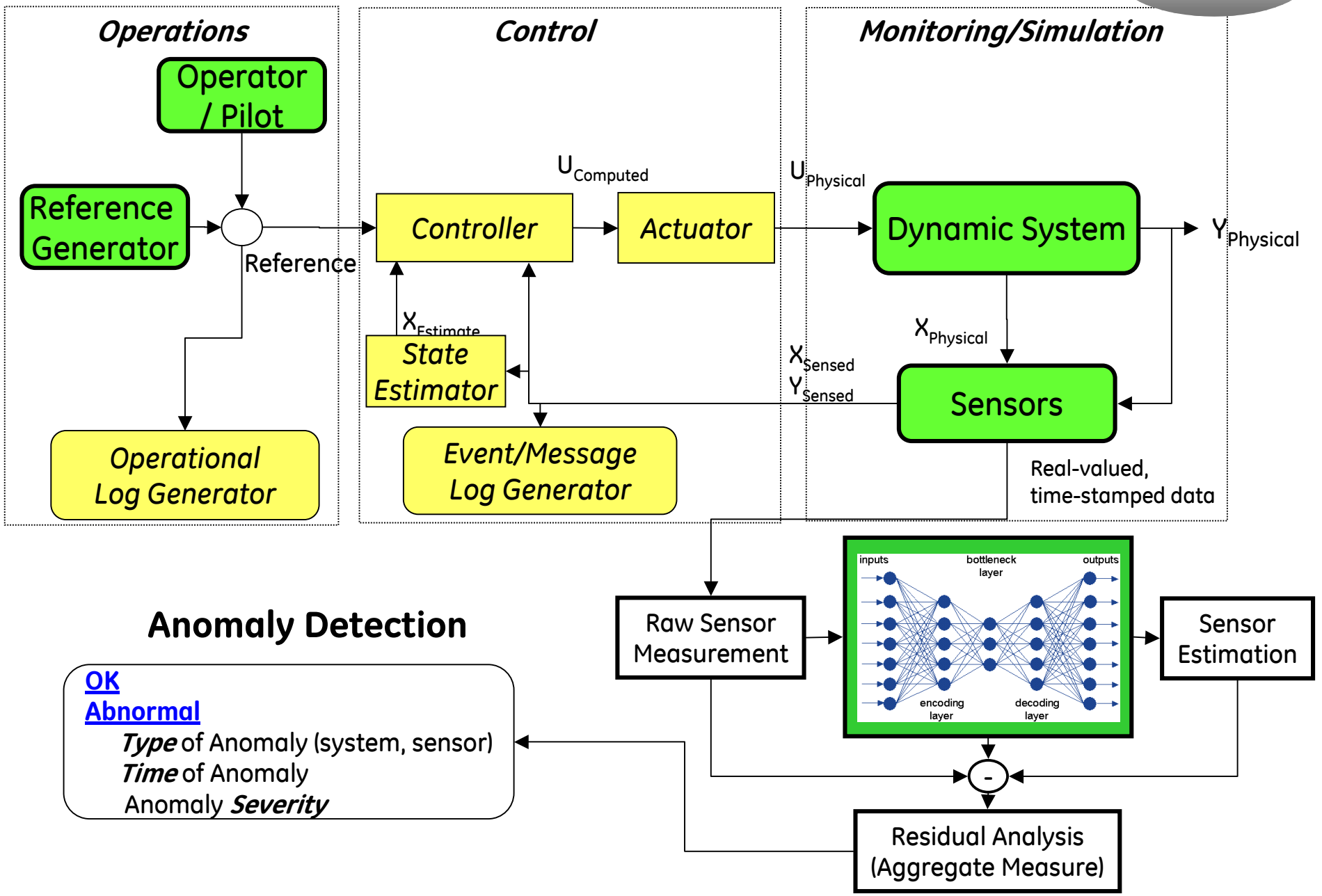
Computation

Traditional NN training with back-propagation

Variable Contribution

Residuals magnitude/distribution

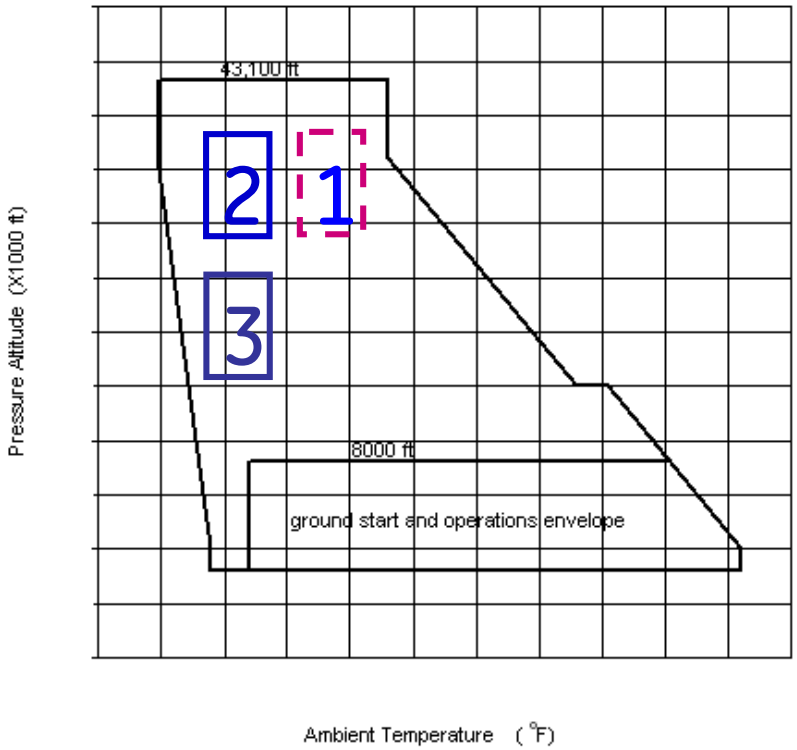
Experiment Setup

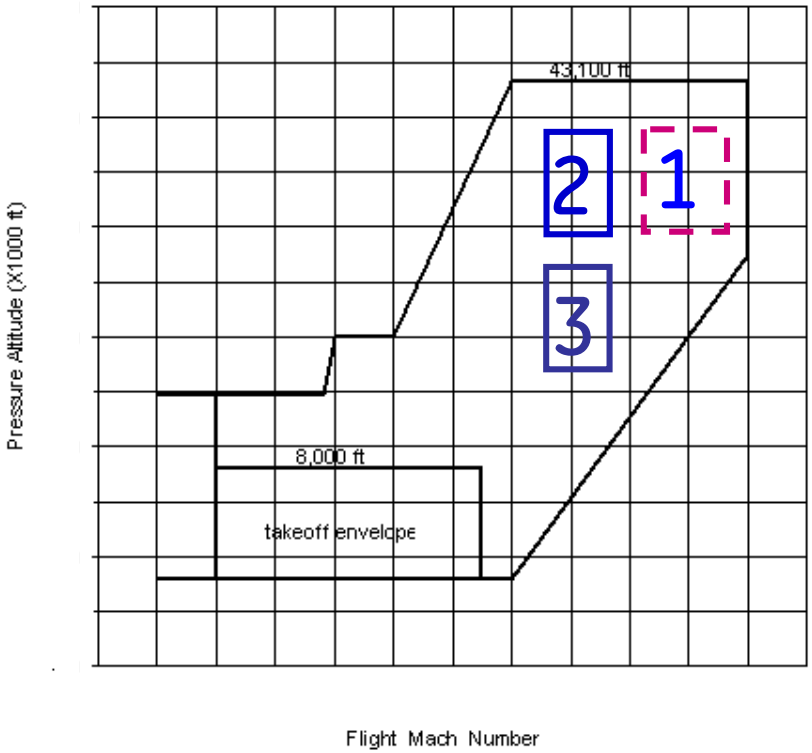


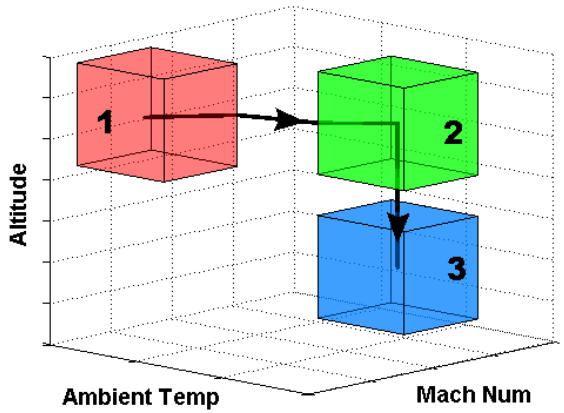
Anomaly Detection

OK
Abnormal
Type of Anomaly (system, sensor)
Time of Anomaly
Anomaly *Severity*

Segmentation of the Operating Space
Three regions in the Flight Envelops







Experiments

Experiments Settings

– We used a **steady state CLM** model for a commercial, **high-bypass, twin-spool, turbofan engine**.

– We can manipulate flight conditions to simulate different operation regimes (i.e. flight envelops of aircraft) and generate data corresponding to them

1st Experiment

Three AANN's: **One for each region in the flight envelop (region)**

Vary ALT, Mach and Tamb ->1000 normal operating pts for each region

Run each operation point through CLM to generate a 9x1 sensor vector

900 points for training (200 for validation); 100 points reserved for test

Results: Each local model performs very well (better than global model) in region of competence, and performs poorly (outside its limited scope)

2nd Experiment

One Global AANN

Train on same 2700 training data points from experiment 1

Run each operation point through CLM to generate a 9x1 sensor vector

Test on the left 300 points

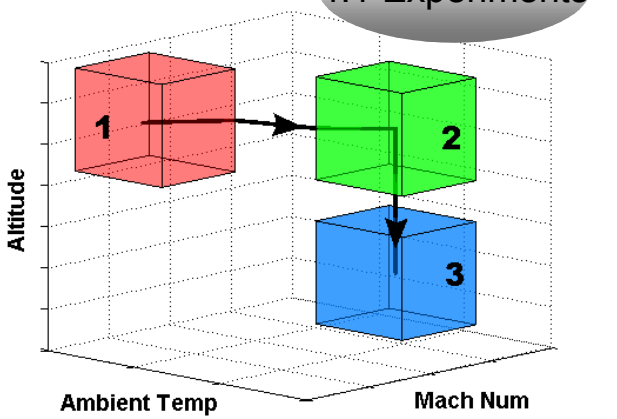
Results: Global model performs fairly across all three regions - shows higher variance than each local AANN operating within its scope

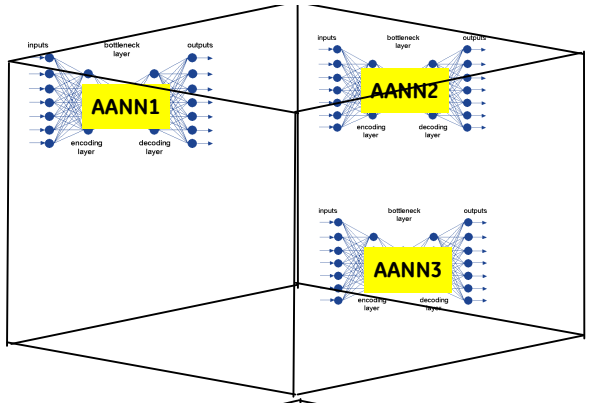
3rd Experiment

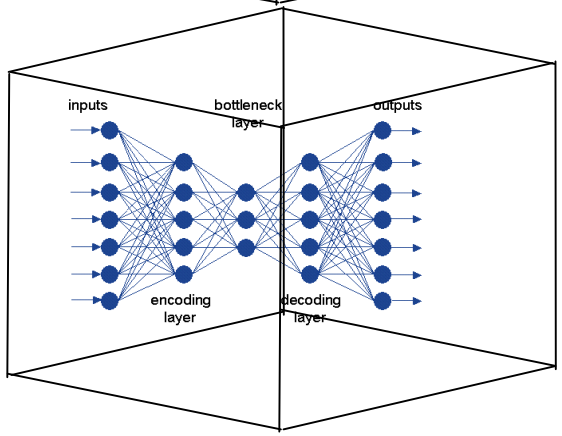
Three AANN's: One for each region in the flight envelop

A Fuzzy Supervisory Model (FSM) to interpolate among local AANN's

Results: Hierarchical structure performs very well across all regions – including transitions





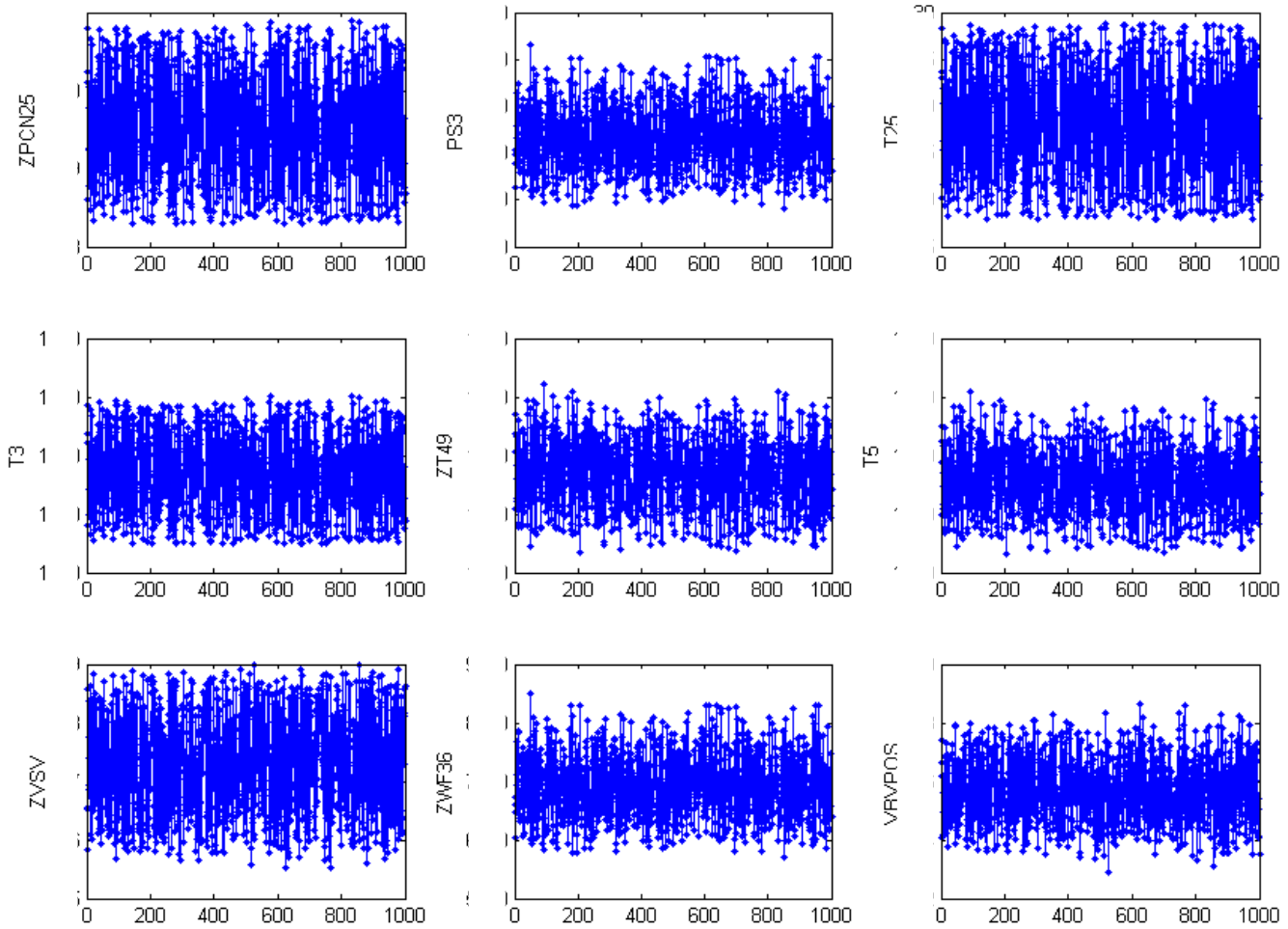


Experiment 1

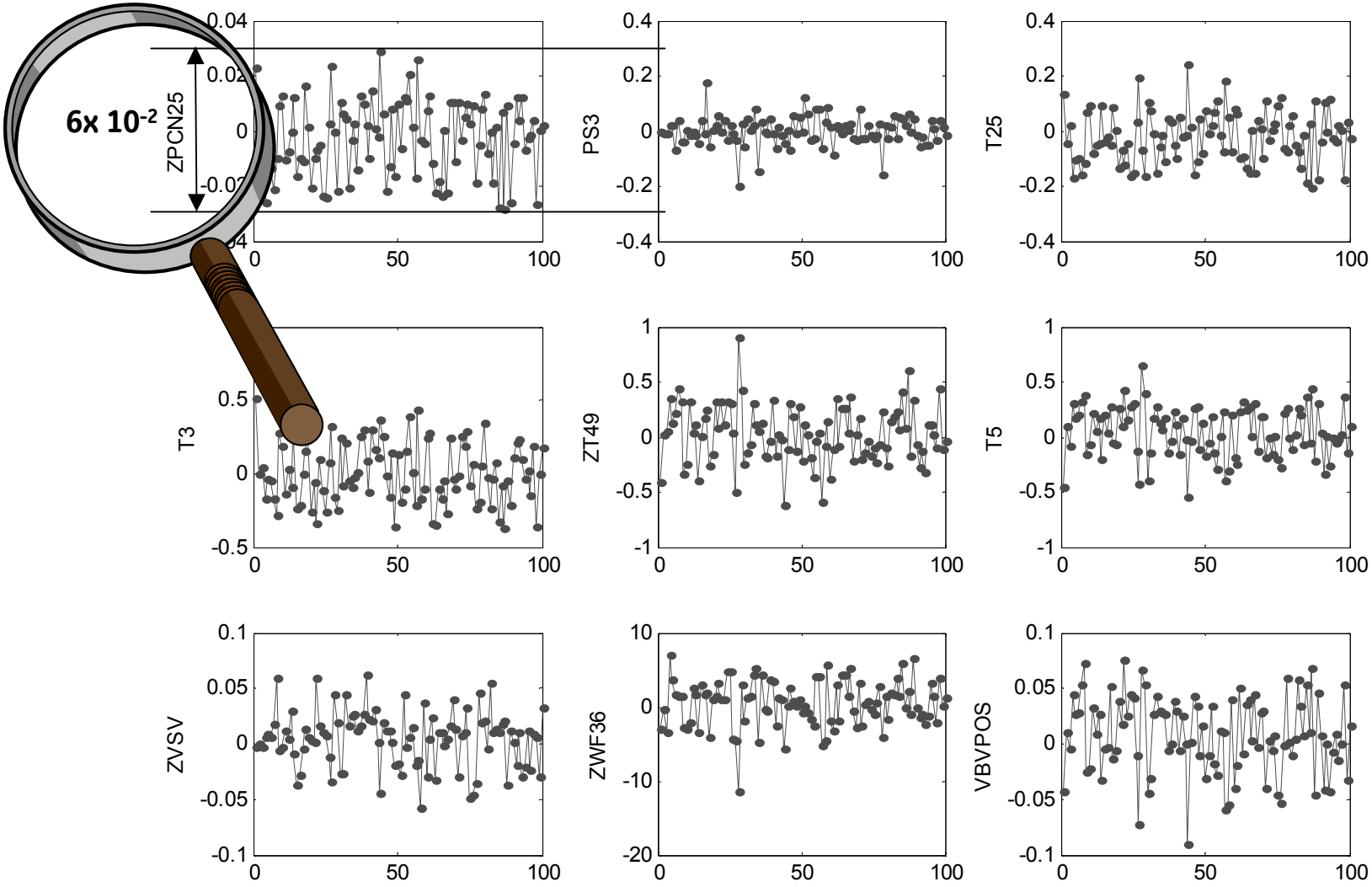
- Vary ALT, Mach and Tamb ->1000 normal operating pts for each flight envelop
- Run each operation point thru CLM to generate a sensor vector (9x1)
- Three AANN's: One for **each region** in the flight envelop
- 900 points for training (200 for validation); 100 points reserved for test

Goal: Create **three local models**
Results: High performance when in scope
inadequate performance when out of scope

Raw Data from Flight Env 1

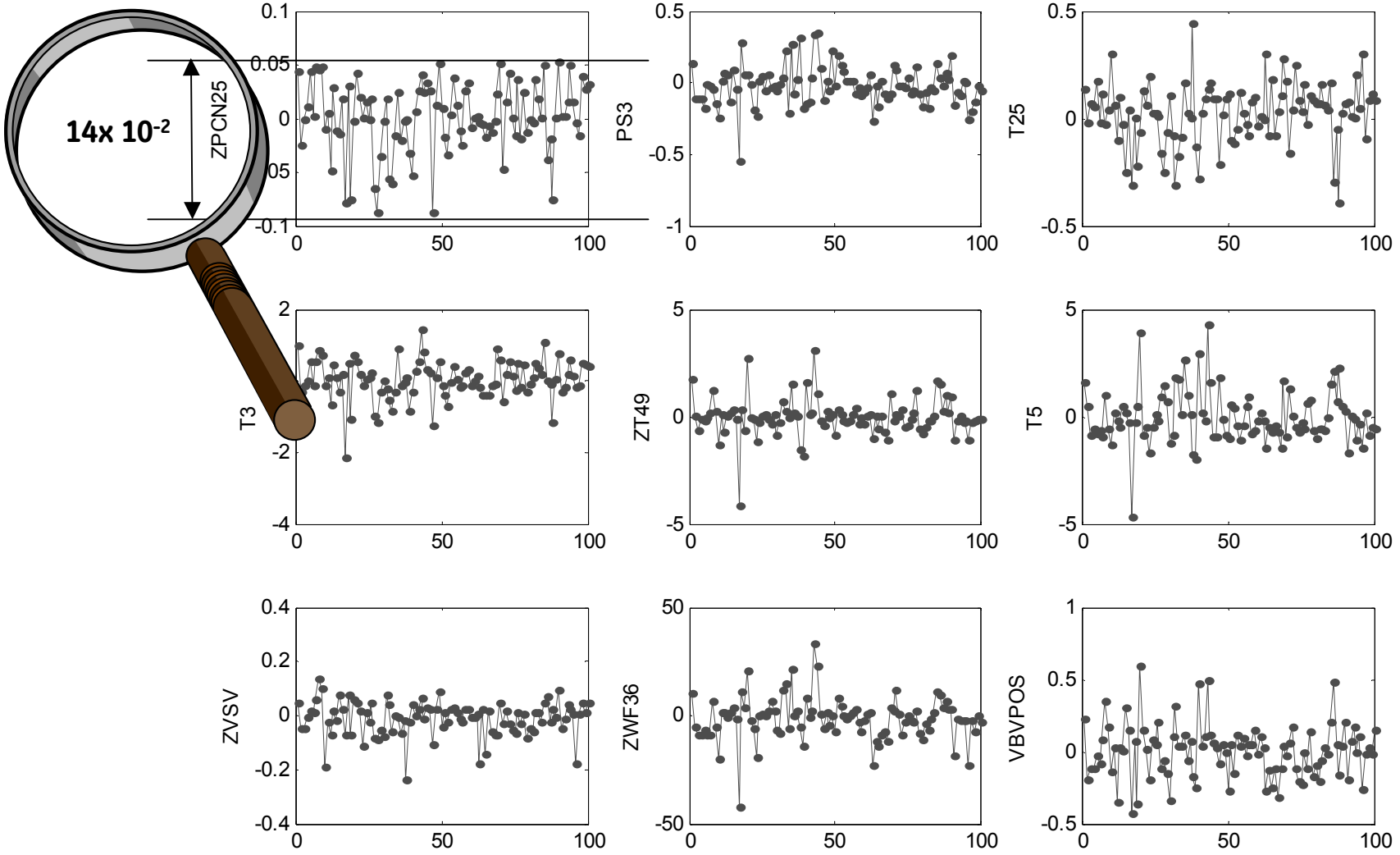


Residuals: test set from FE1 on AANN1



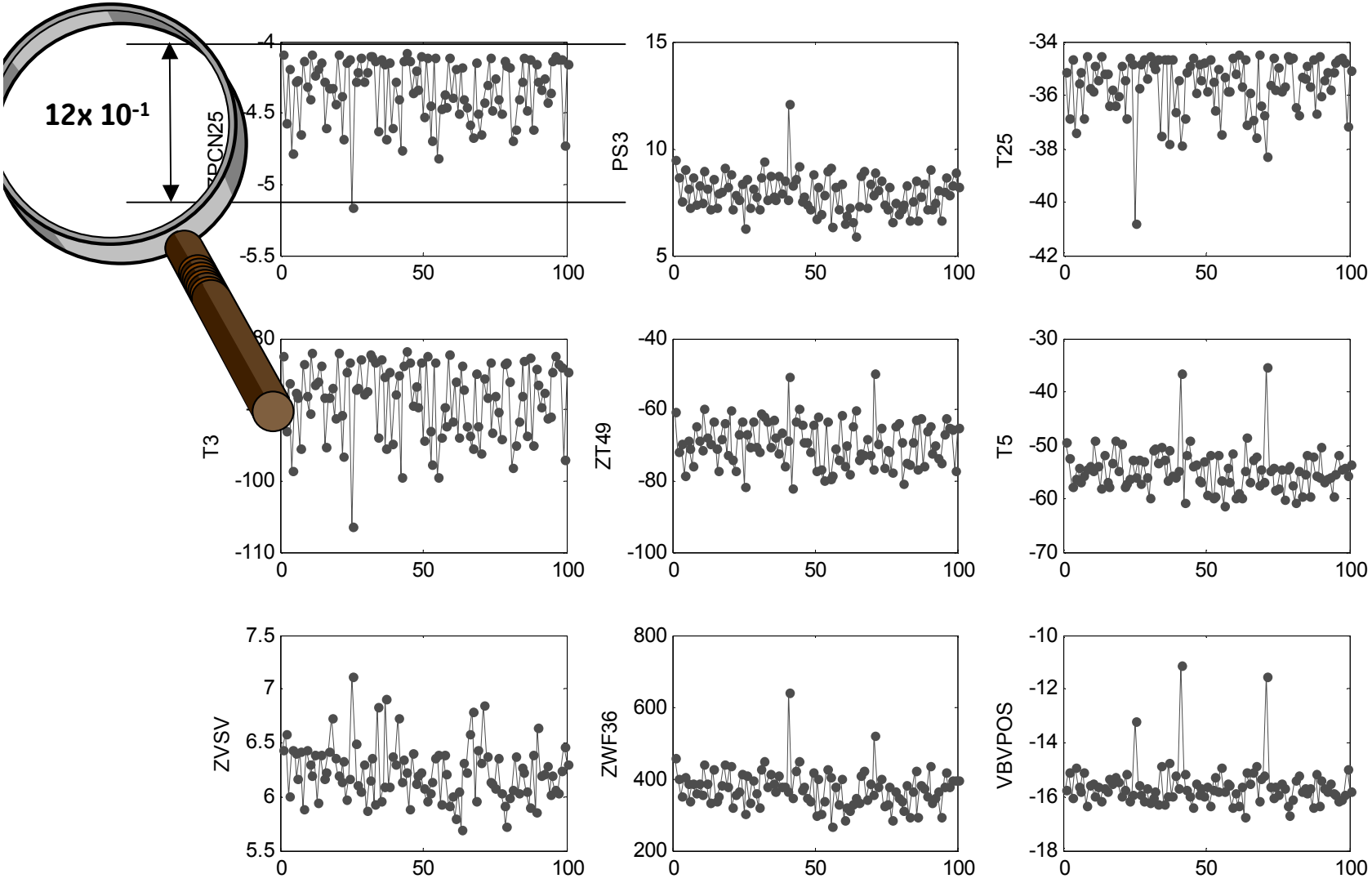
Correct Scope of local model: Small Residuals!

Residuals: test set from FE3 on AANN3



Correct Scope of local model: Small Residuals!

Residuals: test set from FE2 on AANN1



Incorrect Scope of local model: Larger Residuals (10x)

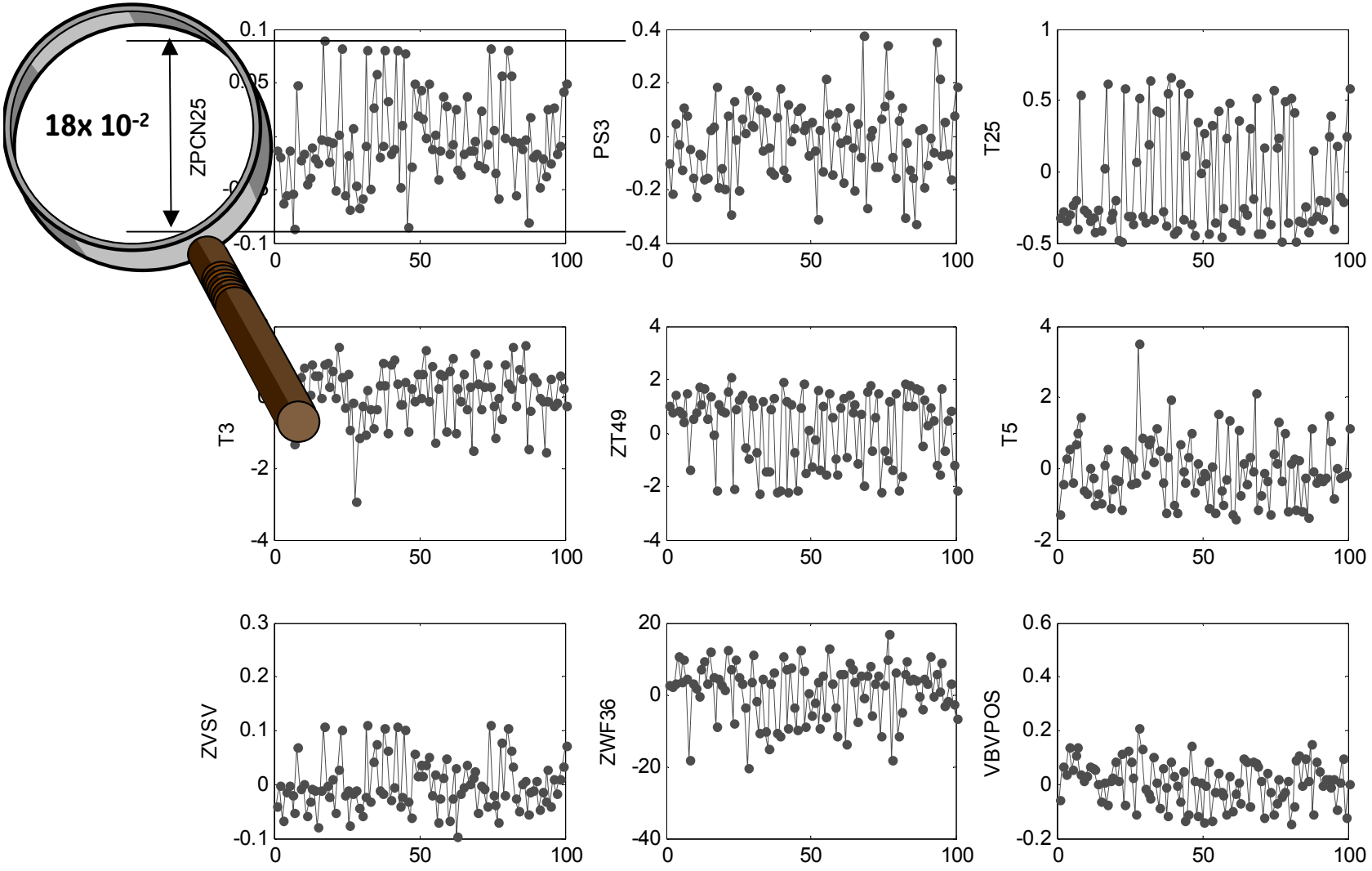
Experiment 2

- One global AANN
- Train on the 2700 training data points from experiment 1
- Test on the left 300 points

Goal: Create **one Global model**
Results: Mediocre performance across entire space
– better than worse performance of local models

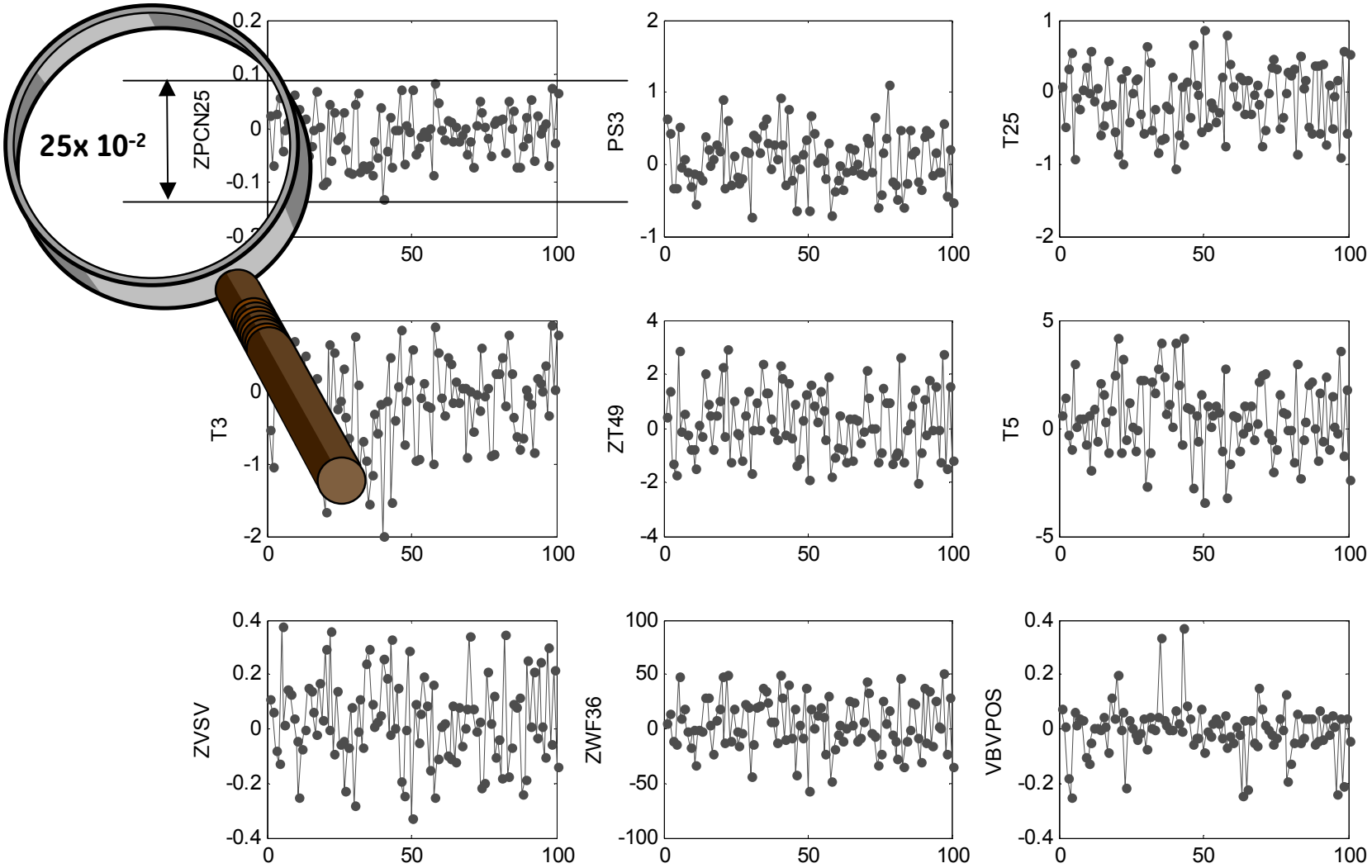
Test data from FE1

4.1 Experiment 2



Increased variance (3x) compared to experiment

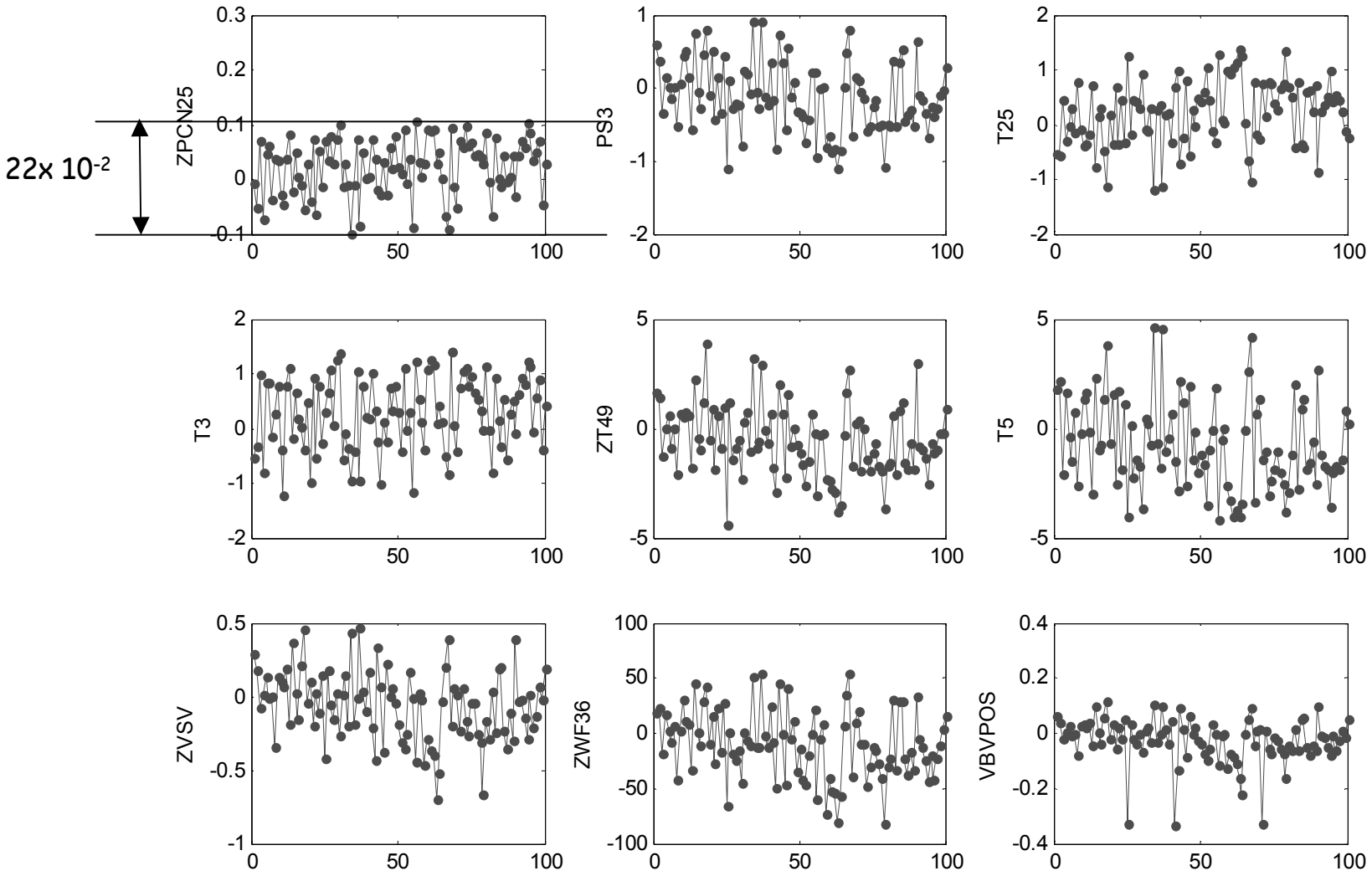
Test data from FE3



Increased variance (2x) compared to experiment 1

Test data from FE2

4.1 Experiment 2



Smaller residuals (20%) compared to FE2 on NN1

Experiments (con't)

3rd Experiment

- **Three AANN's:** One for each region in the flight envelop
- **Fuzzy Supervisory Model (FSM)** to interpolate among local AANN's
- Simulate the change of flight conditions

  FE1: 200 pts

  FE1 → FE2: 200 pts

  FE2: 200 pts

  FE2 → FE3: 200 pts

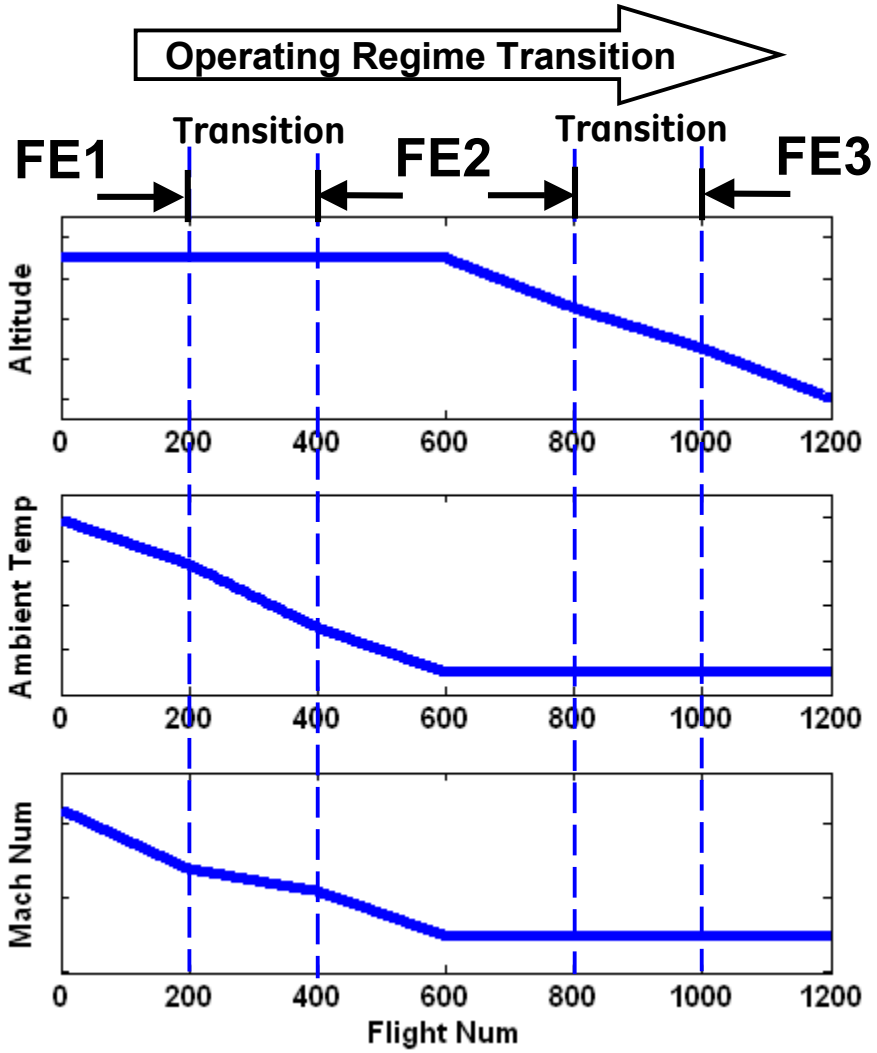
  FE3: 200 pts

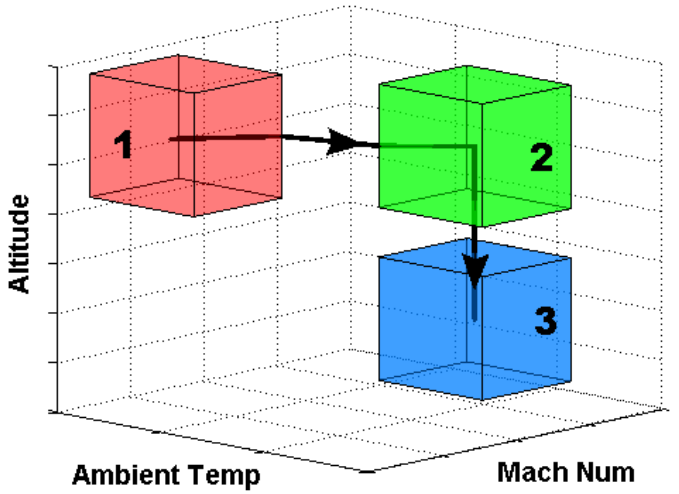
- Test simulated data on Fuzzy Supervisory Model with AANN1, AANN2, AANN3
- Intentionally make transitions in state space not covered by any pre-trained flight envelop

Results: Hierarchical structure performs very well across all regions – including transitions

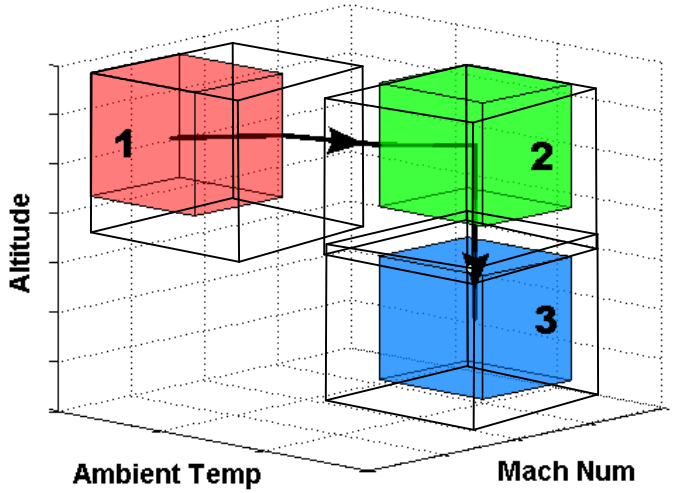
Goal: Create a **Fuzzy Supervisory Model for three local AANN models**
Results: Higher performance across all regions

Flight Envelop Transitions





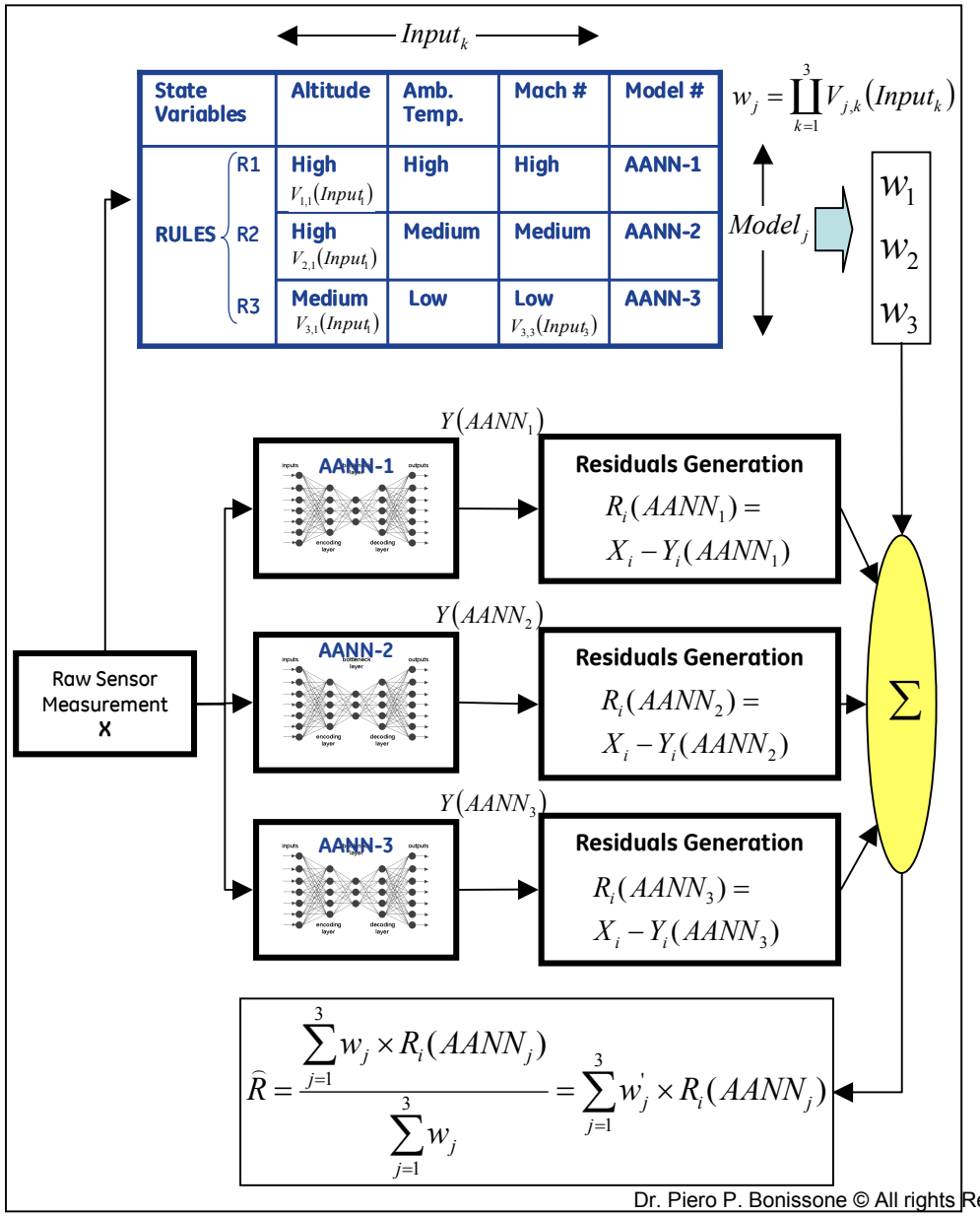
Crisp Model-Transition



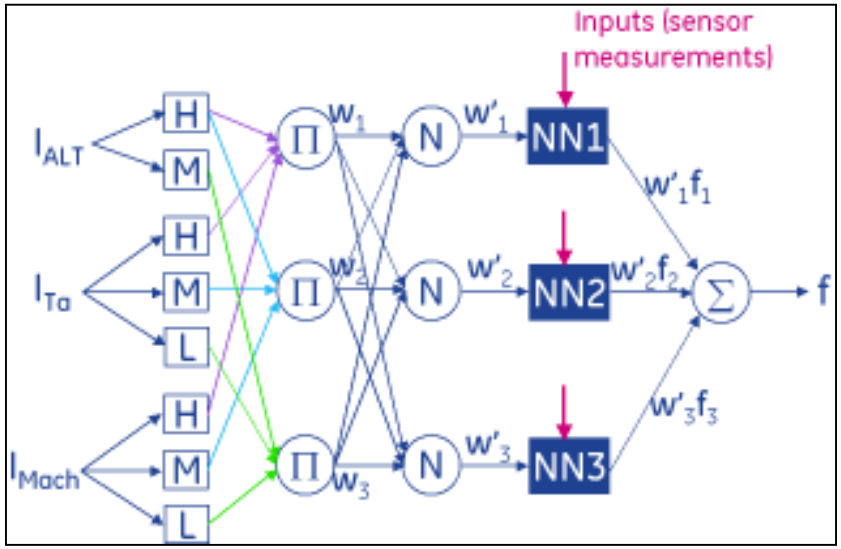
Fuzzy Model-Transition

Transition Management Using Fuzzy Supervisory Model

AANN Interpolation by Fuzzy Supervisory



Network Implementation



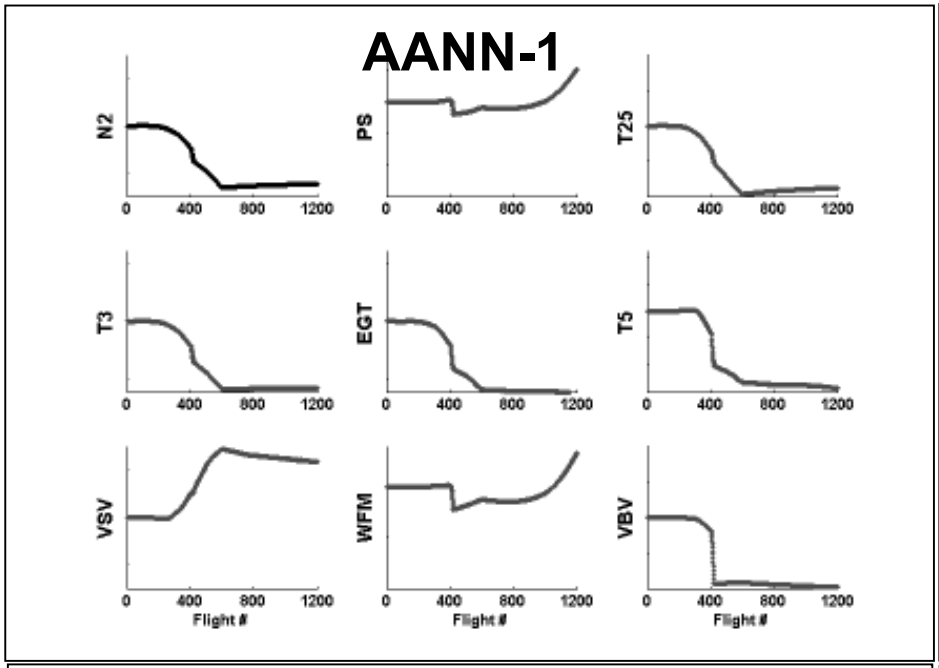
$w_i = \prod_{j=1..3} X_{ij}(I_i)$ $\quad w'_i = w_i/(w_1+w_2+w_3)$

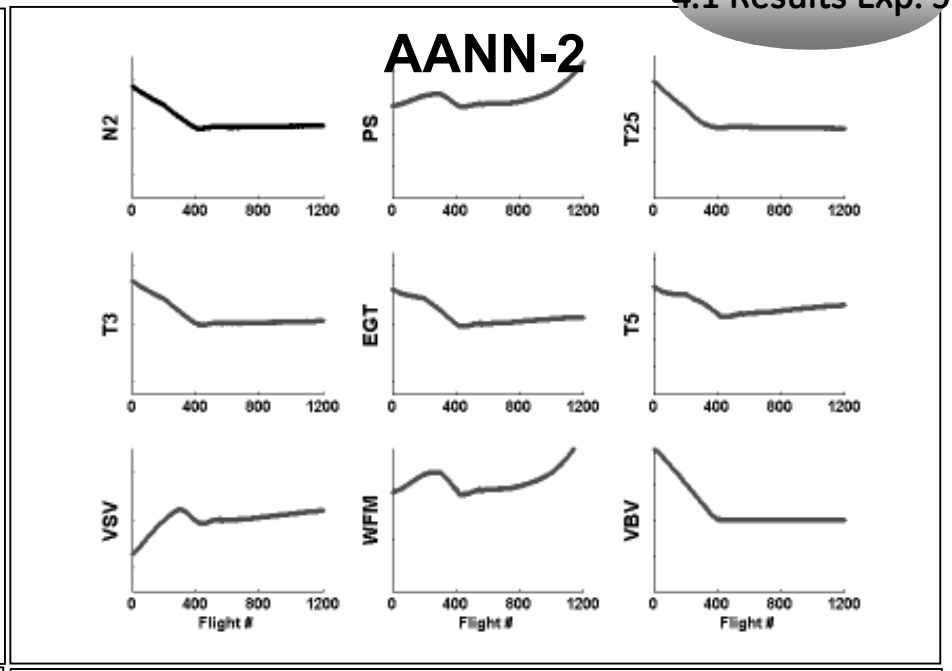
Figure Of Merit (FOM)

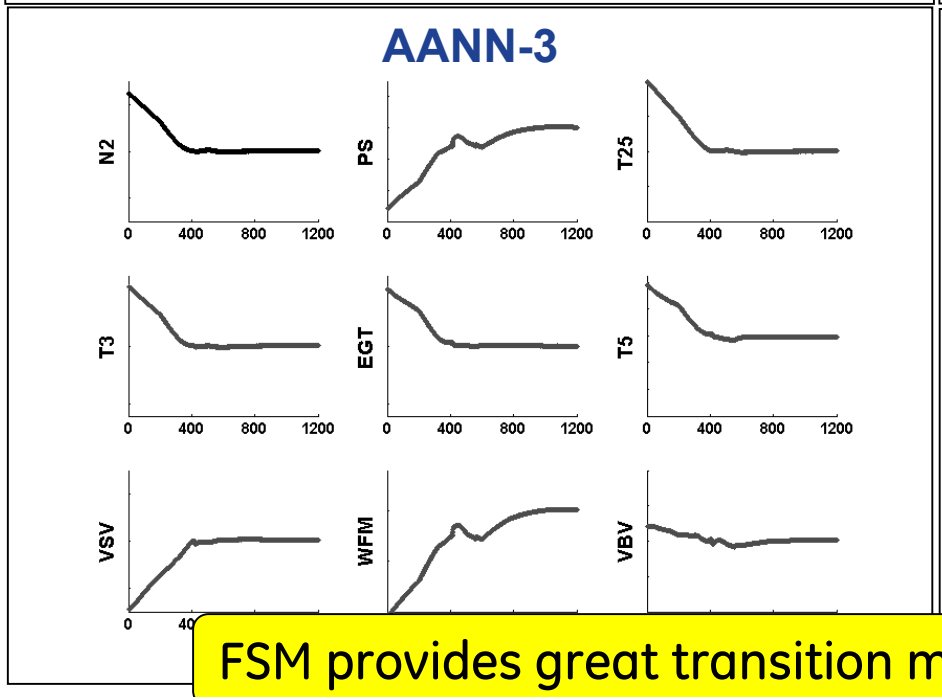
$$FOM = \sqrt{\frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(\frac{R_{ij}}{\overline{X}_i}\right)^2}$$

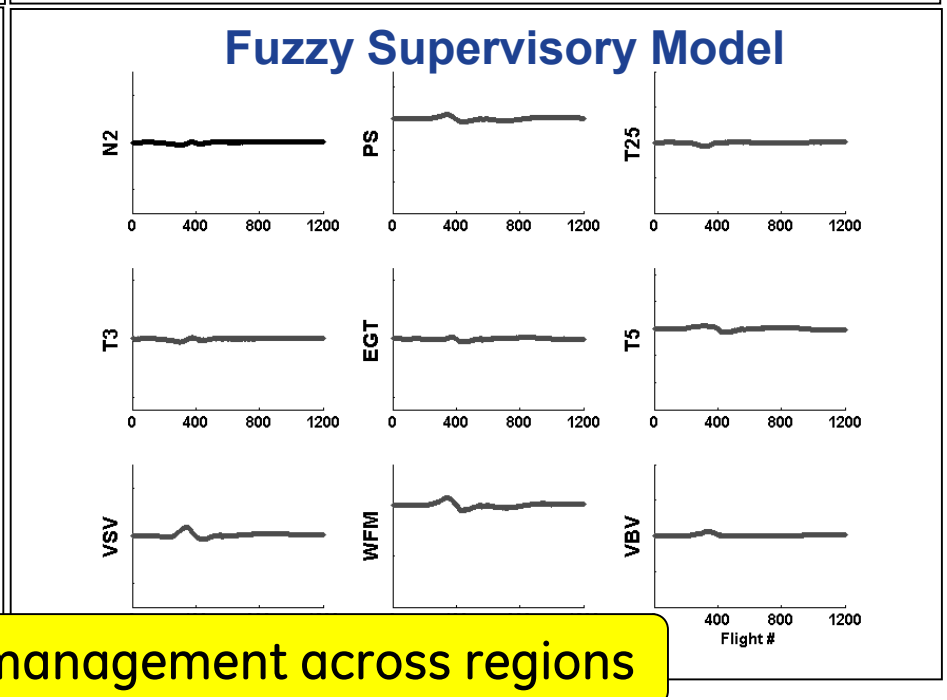
- n is the number of the variables (sensors)
- m is the number of data points (measurement)
- R_{ij} is the residual between true measurement and AANN estimation,
- $\overline{X}_i$ is the mean of the true measurement

Residuals for each AANN and for hierarchical system (with FSM)









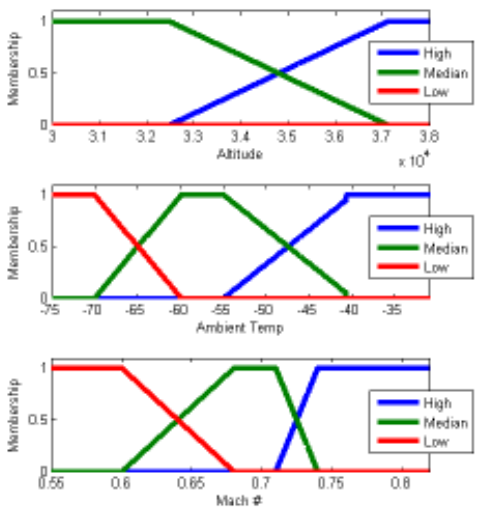
FSM provides great transition management across regions

Design Tuning

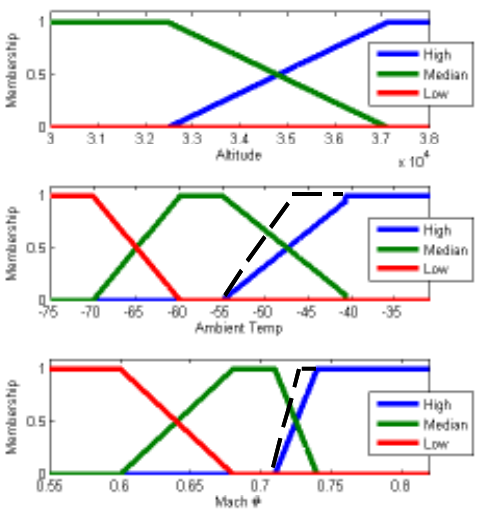
- Design Choices in the Fuzzy Supervisory Model (FSM)

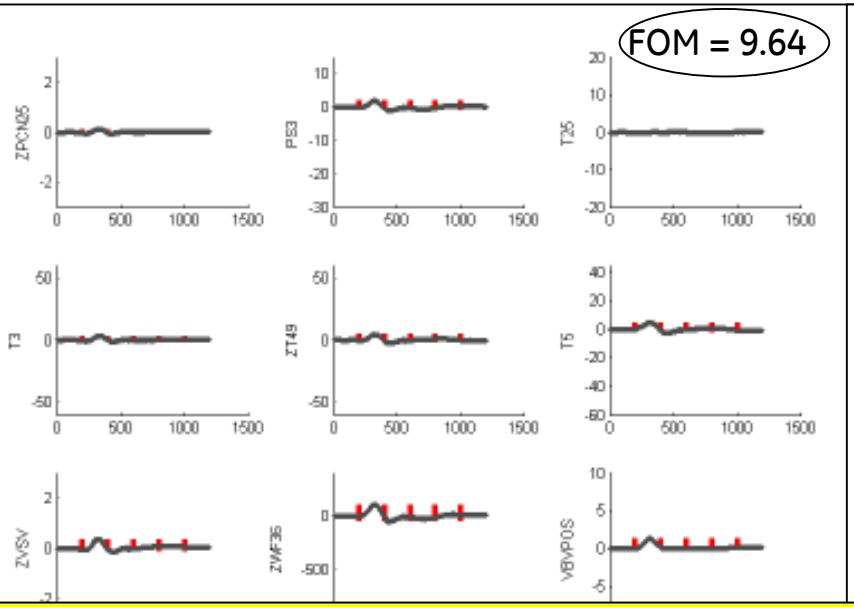
- Tuning the Fuzzy Supervisory Model
  - Manual tuning of FSM State Partitions
  - Automated tuning of FSM State Partitions

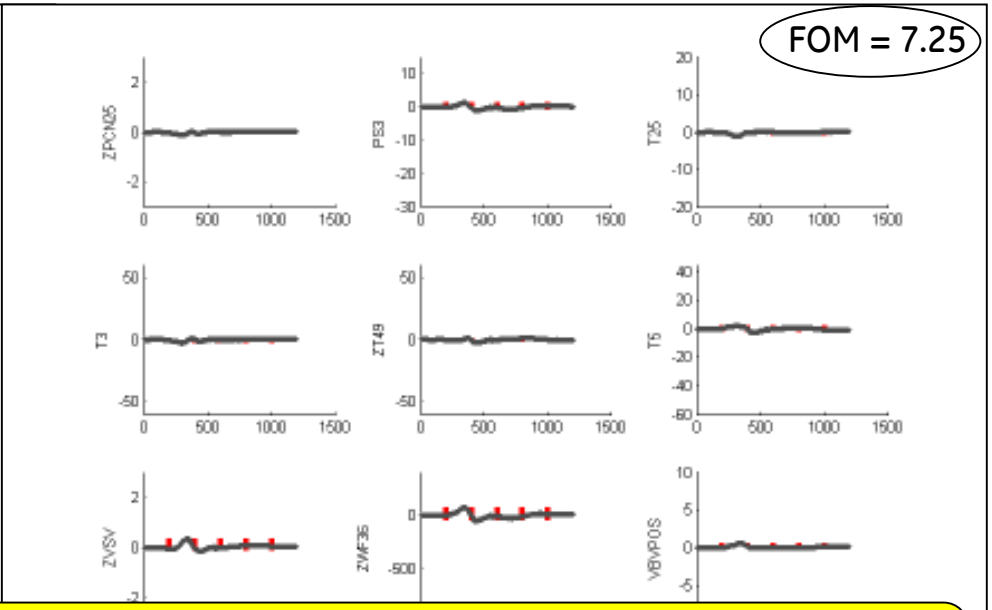
Manual FLS Tuning: Membership function parameters



Manual Tuning (extending AANN1 scope)

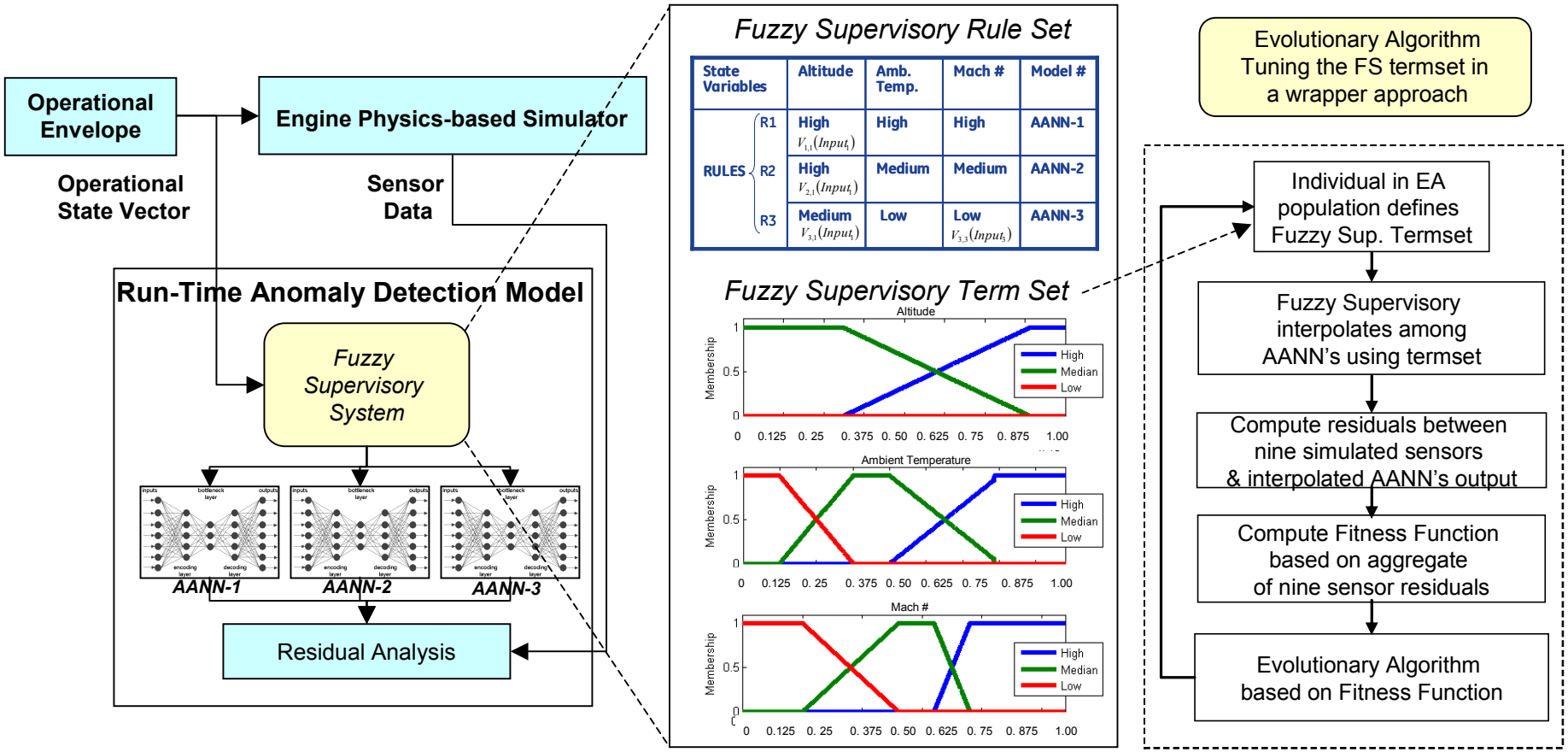




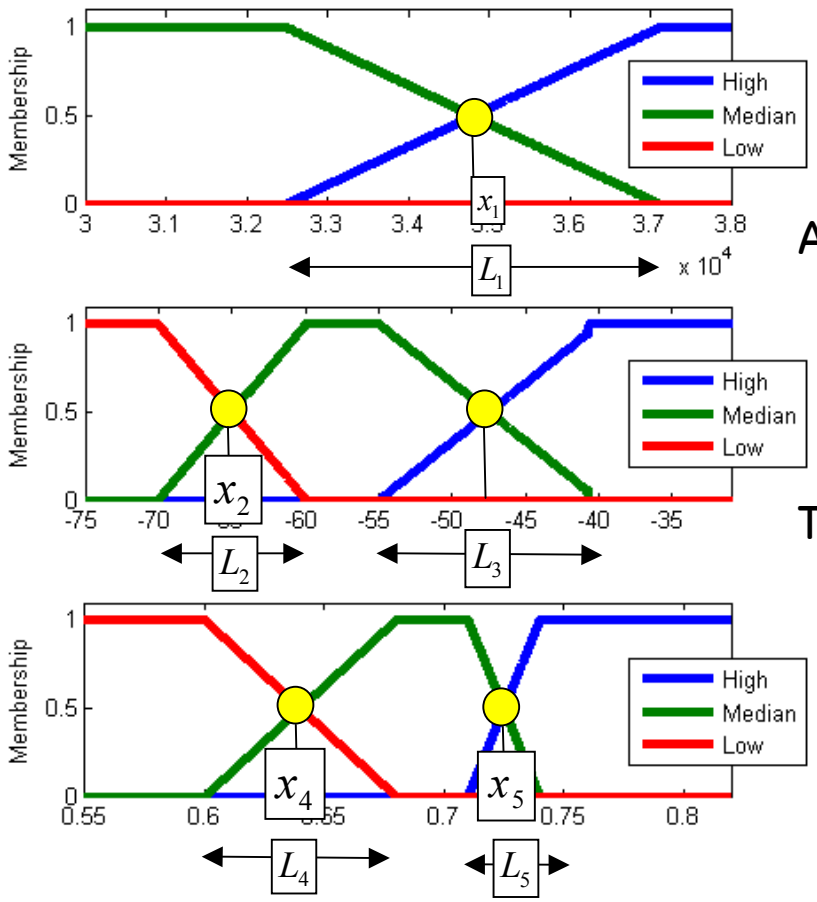


Manual tuning, extending AANN1's scope, lead to a 25% FOM improvement
We could use FOM for gradient or evolutionary parametric tuning

Automated FLS Tuning with an EA using a Wrapper Approach



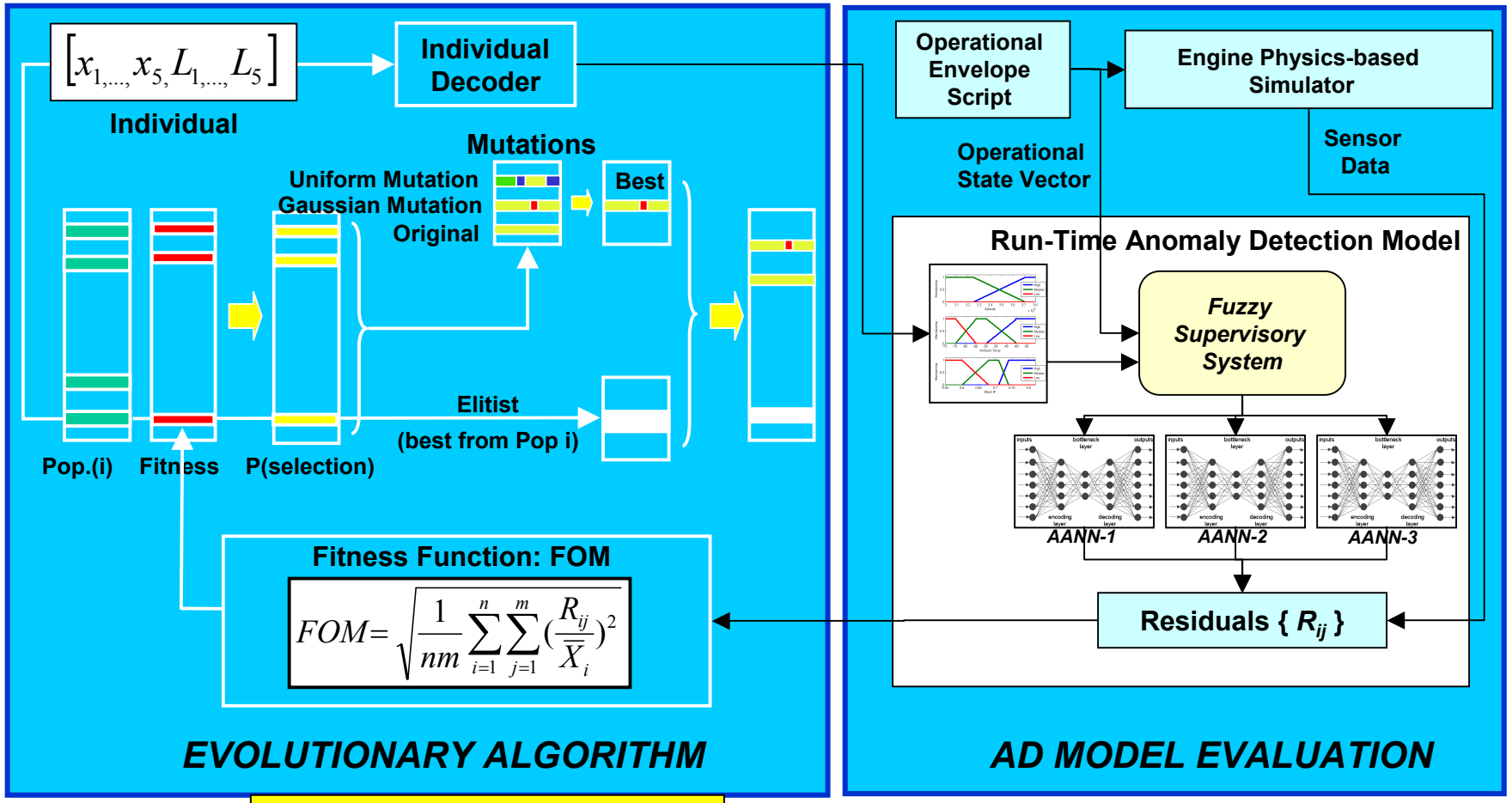
Automated FLS Tuning: Encoding Trapezoidal Membership functions



$$\left[x_{1,\ldots,}x_5,L_{1,\ldots,}L_5\right]$$

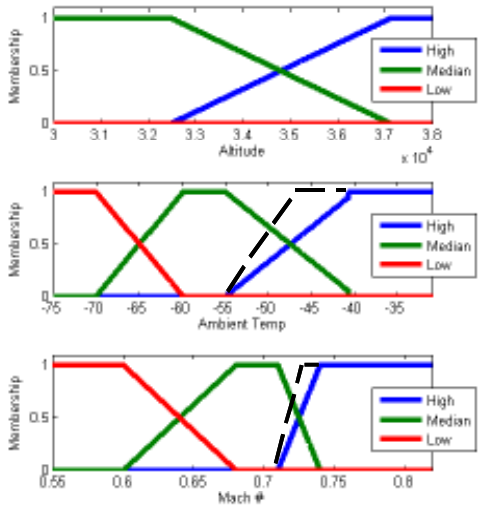
Encoding the abscissa of the slope intersections (x_i) and the lengths of the bases of each triangle (L_i) as an individual in the Evolutionary Algorithm population

Evolutionary Search for Tuning a Fuzzy Supervisory System using a Wrapper Approach



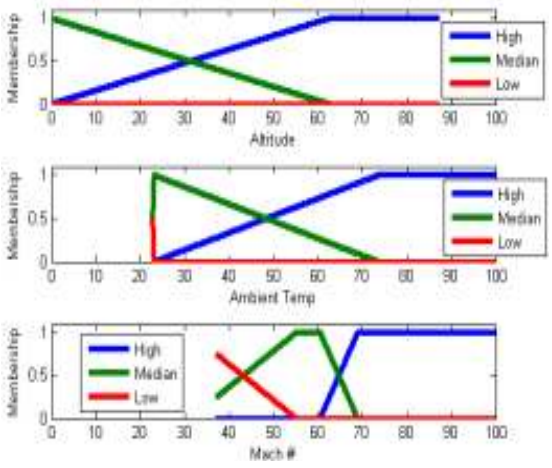
Pop Size = 500 individuals
GenMax = 1,000 generations

Anomaly Detection - Results
Automated FLS Tuning: Membership function parameters



Meta-Heuristic Tuning

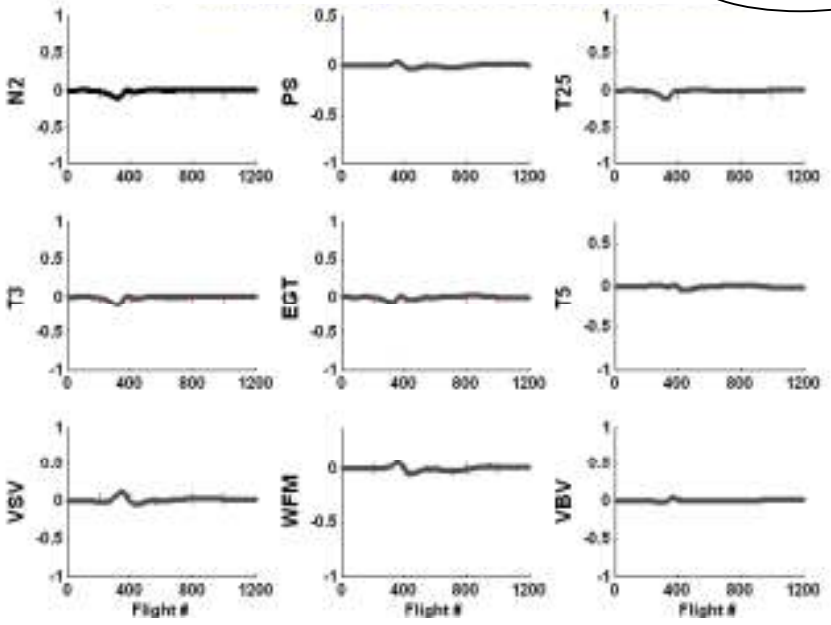
Use **Global Tuning**
(based on FOM fitness function and Genetic Algorithm)
to further improve results



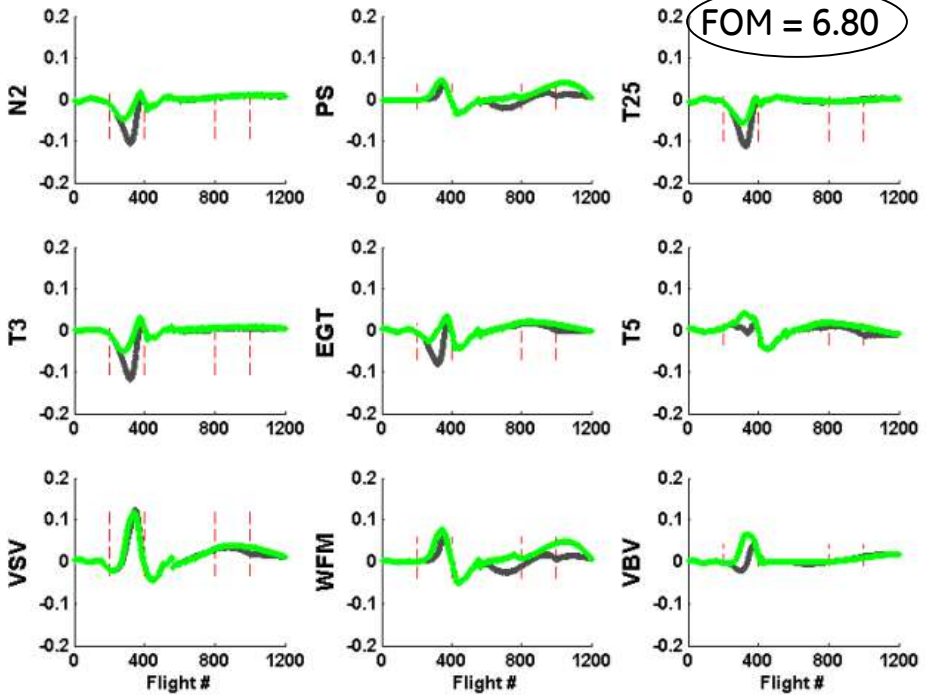
Note: Magnified scale to enhance comparison

Before GA — After GA

FOM = 7.25

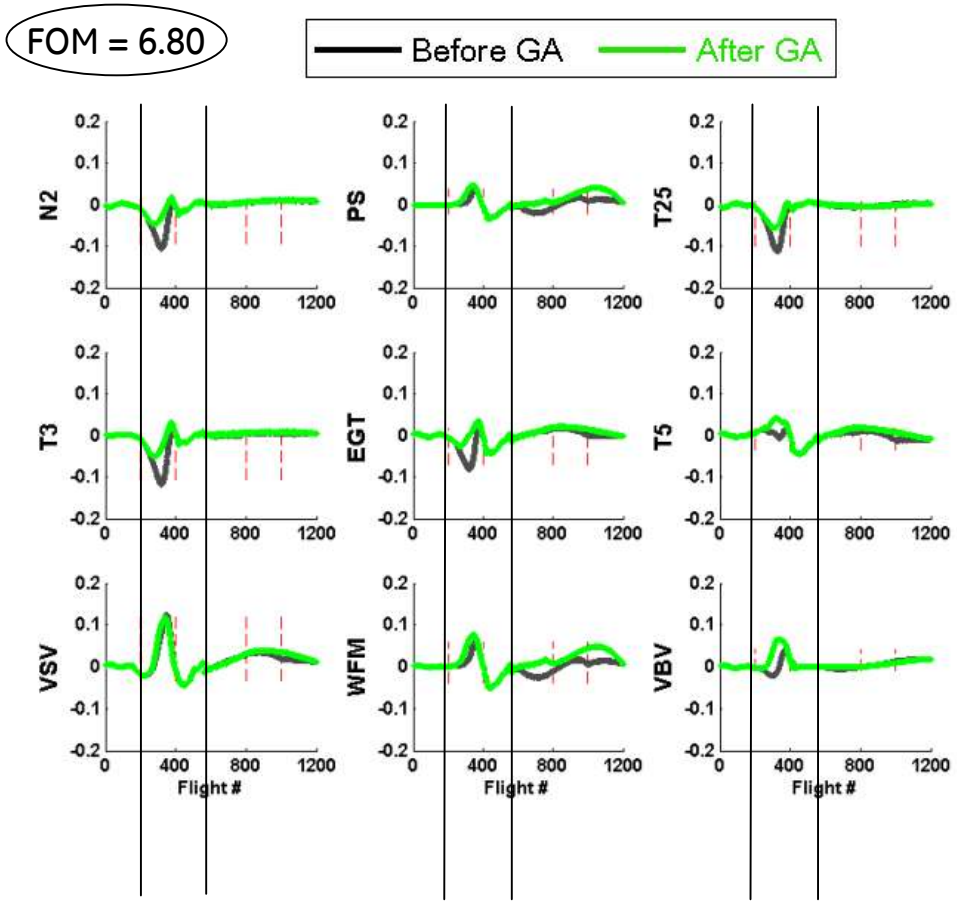


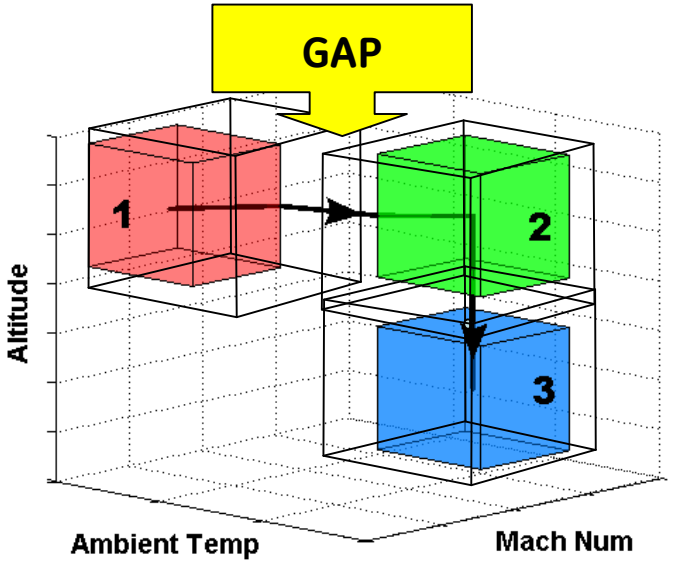
FOM = 6.80

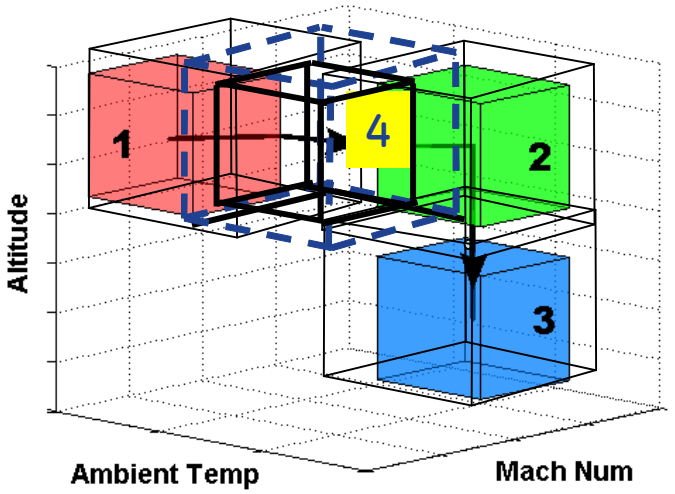


Improving AD Design : Add AANN-4 & retune FLS

4.1 Design

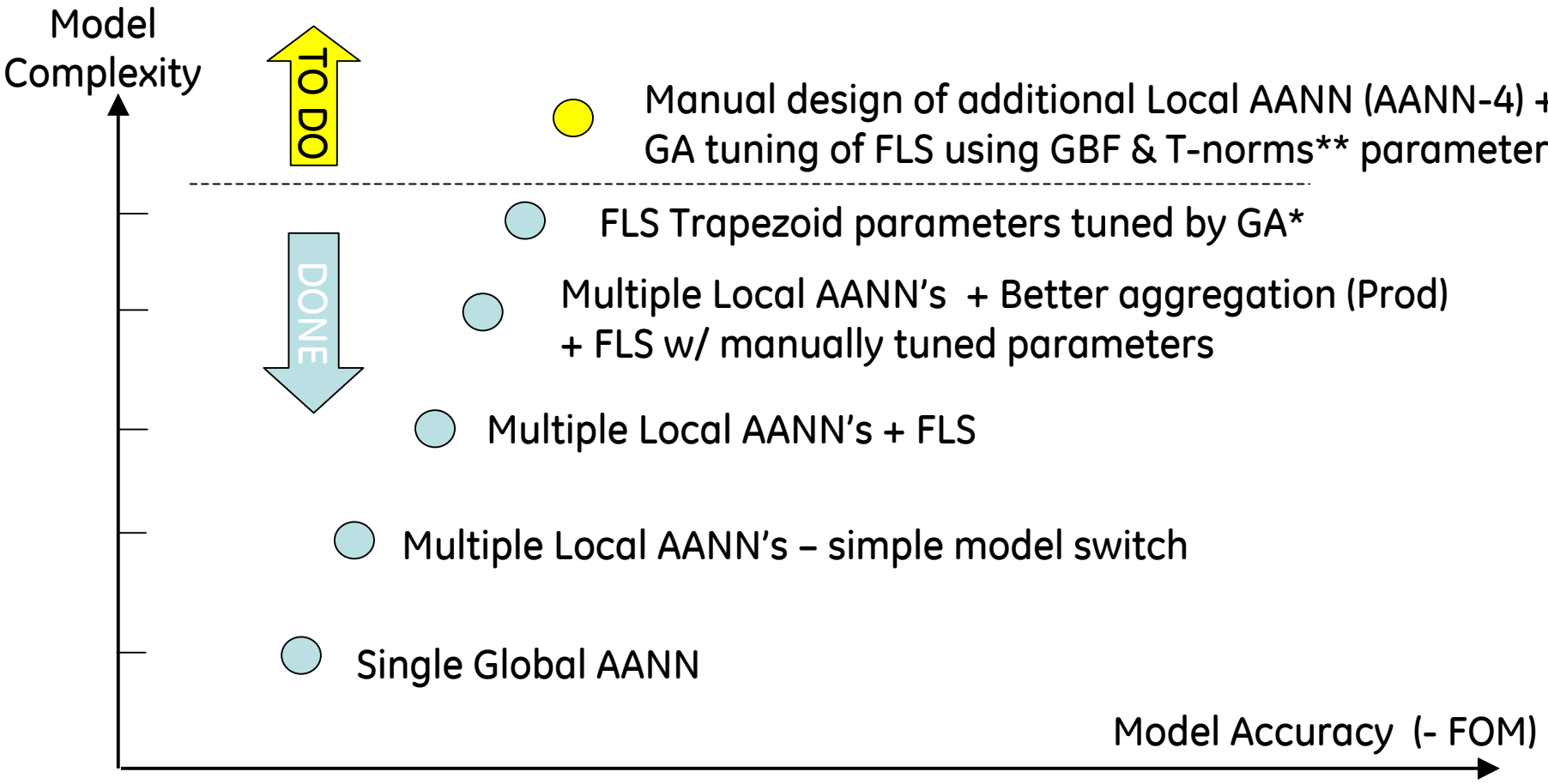






Most residual errors occur in the [200, 600] interval, indicating a performance limit that cannot be addressed only by tuning the FLS. Rather it suggests the need for an additional AANN-4 to provide better coverage in that region

Design Tradeoffs



* Chromosome: $[x_1,\ldots,x_5,L_1,\ldots,,L_5]$

** Chromosome: $[(a_{11},b_{11},c_{11}),\ldots,(a_{13},b_{13},c_{11}),\ldots,(a_{n3},b_{n3},c_{n3}),p]$

Future Work

- **Hierarchical Design (to Improve Accuracy and Extend Region of competence)**
  - **+ Used Offline Metaheuristics (EA) and Online Metaheuristics (FLS) with AANN model**
  - **- Use a more complex encoding for the EA individual to evolve BOTH structure and parameters:**

    # AANN Models
    Scope of AANN Models
    Evolve membership Functions (GBF) in FLS
    Evolve Aggregation operators (parameterized T-norms)

- **Model Lifecycle (to maintain model Vitality)**
  - **- Use Offline Metaheuristics (EA) to create/retune hierarchical design with updated data sets (e.g. reflecting more recent engine degradation)**

4.2 Case Study 2:

Fusion of (Competing) Classifiers for Diagnostics (multi-class classification)

Reference: W. Yan and F. Xue (2008). Jet Engine Gas Path Fault Diagnosis Using Dynamic Fusion of Multiple Classifiers, IJCNN 2008, Hong Kong, pp-1585-1591– [GE GR Technical Report, 2008GRC395, May 22, 2008].

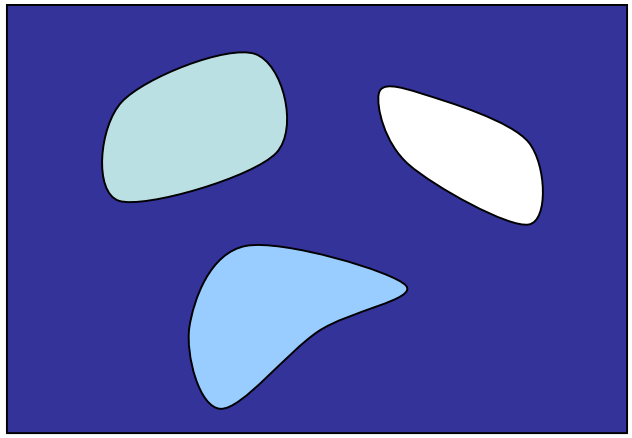
4.2 Fusion of (Competing) Classifiers

- MCS Objectives
- Static Selection/Fusion
- Dynamic Selection/Fusion
- Performance Evaluation
- Application Example: Jet engine fault diagnosis
- Results
- Conclusions & Future Work

Multiple Classifier Systems (MCS)

Objectives

- Introduce a new classifier fusion scheme: dynamic classifier fusion
- Apply it to a real-world classification problem - Jet engine fault diagnosis



Global methods

Local methods

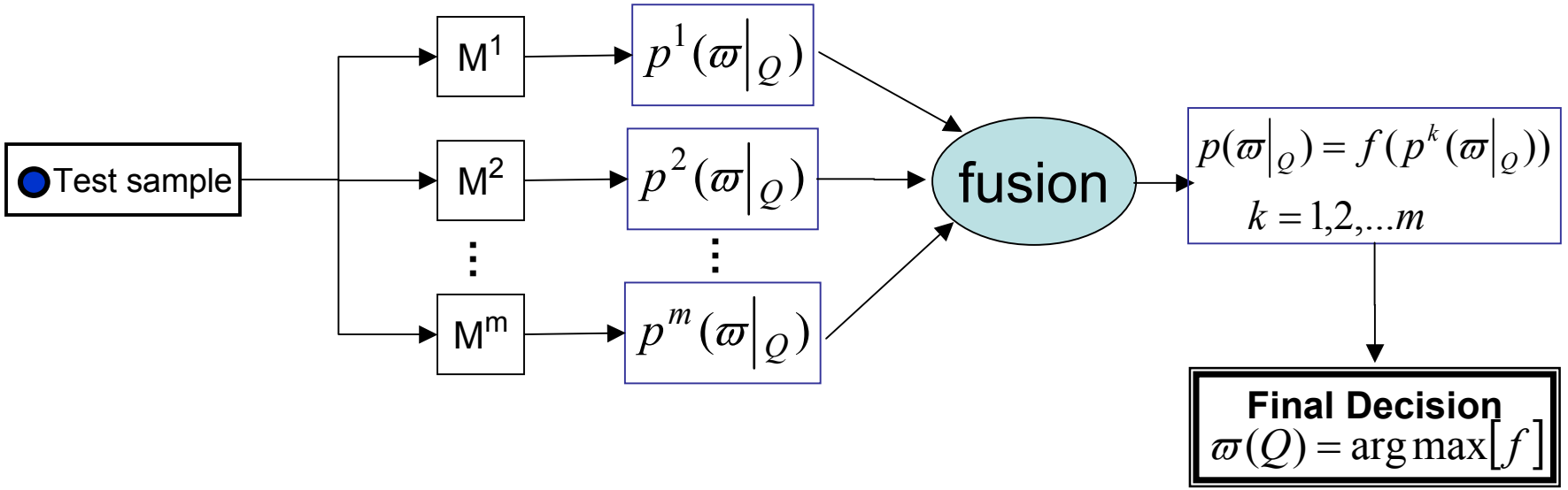
- static
  - selection
  - fusion
- dynamic
  - selection
  - **fusion**

Dynamic fusion of multiple classifiers

Adapted with permission from: W. Yan and F. Xue (2008). Jet Engine Gas Path Fault Diagnosis Using Dynamic Fusion of Multiple Classifiers, *IJCNN 2008*, Hong Kong, pp-1585-1591

Static Fusion

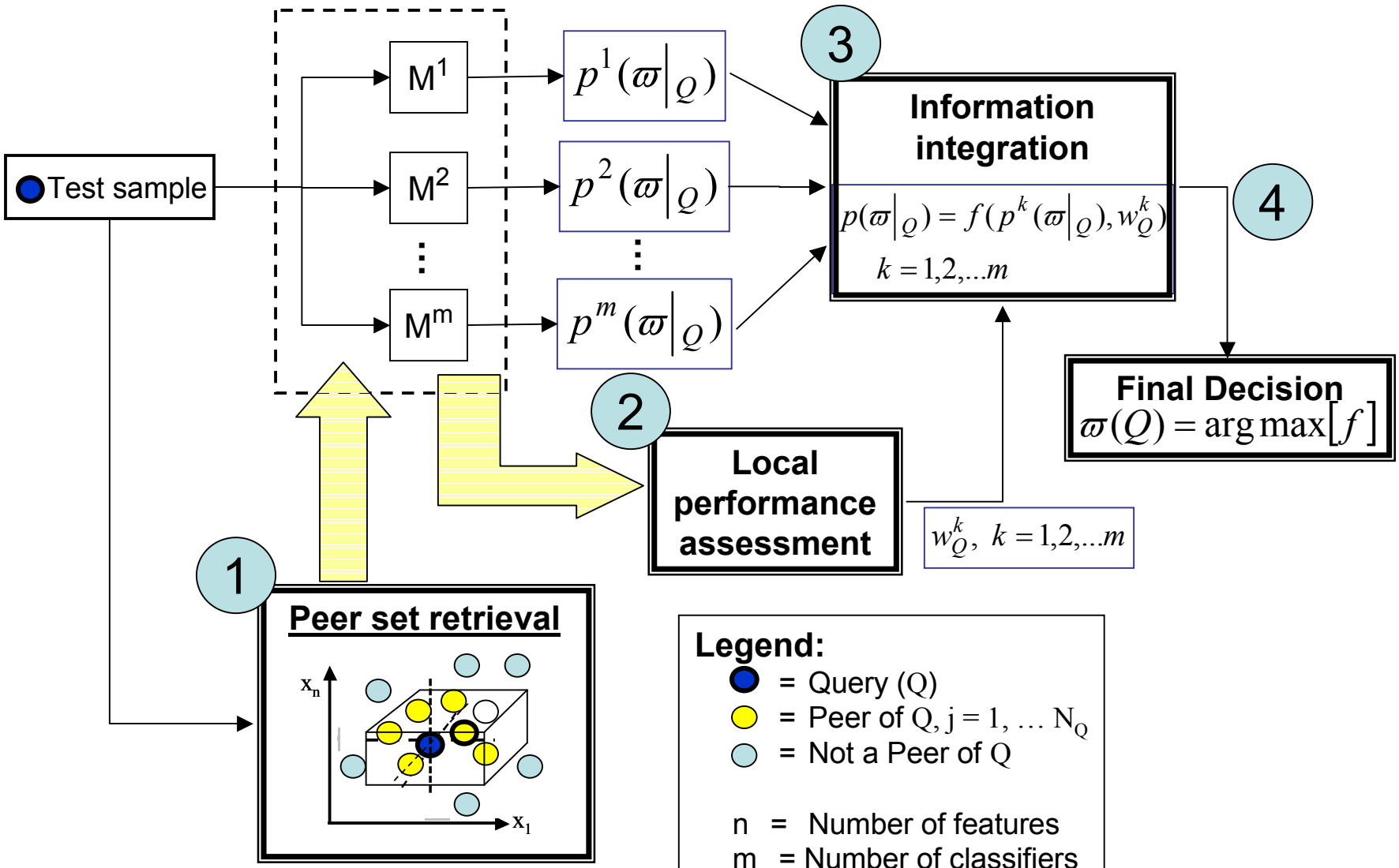
Conventional static fusion



M_i = i^{th} Classifier, i=1, …m,…

Adapted with permission from: W. Yan and F. Xue (2008). Jet Engine Gas Path Fault Diagnosis Using Dynamic Fusion of Multiple Classifiers, *IJCNN 2008*, Hong Kong, pp-1585-1591

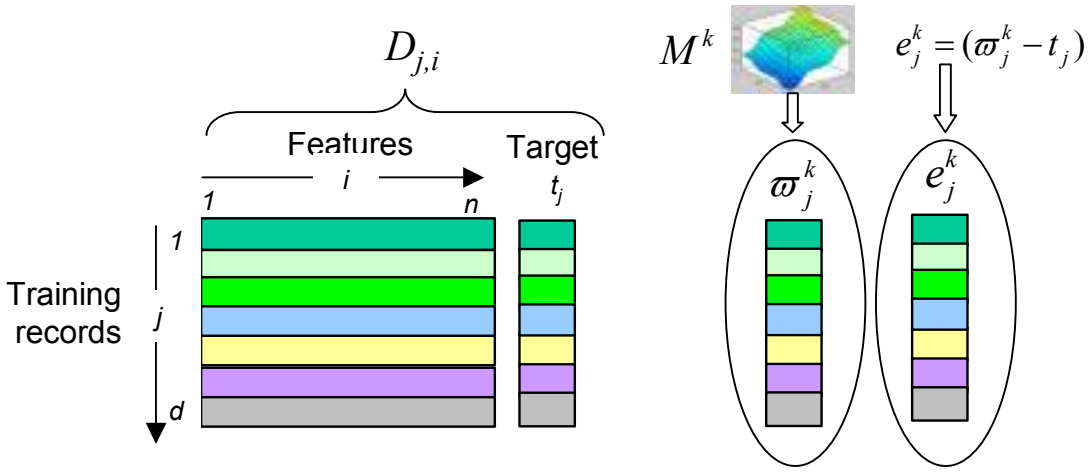
Dynamic fusion



Adapted with permission from: W. Yan and F. Xue (2008). Jet Engine Gas Path Fault Diagnosis Using Dynamic Fusion of Multiple Classifiers, *IJCNN 2008*, Hong Kong, pp-1585-1591

Dynamic fusion – cont'd

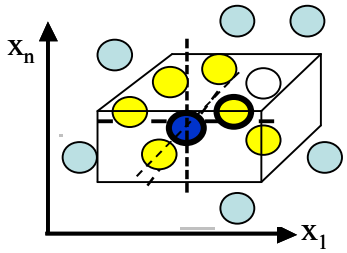
0. Training Set



Example for # classes = $\left|\varpi_j^k\right| = C = 7$

$$\varpi_j^k = [0.2, 0, 0.1, 0.7, 0, 0, 0]$$
$$t_j = [0, 0, 0, 0, 1, 0, 0, 0]$$
$$e_j^k = [0.2, 0, 0.1, -0.3, 0, 0, 0]$$

1. Peer set retrieval



$$N(Q) = \left\{ u_j \,\middle|\, \left\| x_{i,Q} - x_{i,j} \right\| < R_i \quad i = 1,\ldots,n \right\}$$

Design Parameter R_i

$\left|N(Q)\right| = N_Q$ is a function of the point density and the value of hyper-edge R_i

R_i should be tuned for each problem using local or global search techniques

In our example $R_i = 5\%\ range\ X_i$

Adapted with permission from: W. Yan and F. Xue (2008). Jet Engine Gas Path Fault Diagnosis Using Dynamic Fusion of Multiple Classifiers, *IJCNN 2008*, Hong Kong, pp-1585-1591

Dynamic fusion – cont'd

2. Local performance assessment

Local performance for k^{th} classifier:

(a) Compute error for each record of the k^{th} classifier training set

$$e_j^k = (\varpi_j^k - t_j)$$

(b) Compute me_Q^k the mean error of the k^{th} classifier over all the Peers of Q

$$me_Q^k = \frac{1}{N_Q}\sum_{j=1}^{N_Q} e_j^k$$

me_Q^k represents the bias the k^{th} classifier in the neighborhood of Q

3. Information integration

Bias adjusted output of k^{th} classifier:

$$p^k(\varpi|_Q) - me_Q^k$$

4. Final Decision

Baselines

(a) Final decision (by simple averaging):

$$\varpi(Q) = \arg\max\left[\frac{1}{m}\sum_{k=1}^{m}(p^k(\varpi|_Q))\right]$$

(b) Final decision (by dynamic selection) [Woods 1997]: k^{th} classifier Local Accuracy for Q

$$LAC^k(Q) = \sum_{i=1}^{C} CM_Q^k(i,i) \Big/ \sum_{i,j=1}^{C} CM_Q^k(i,j)$$

$$\varpi(Q) = \arg\max\left[p^{\arg\max[LAC_Q^k]}(\varpi|_Q)\right]$$

Select k^{th} classifier with highest Local Accuracy for Q

Proposed Decision Method

(c) Final decision (by dynamic fusion):

$$\varpi(Q) = \arg\max\left[\frac{1}{m}\sum_{k=1}^{m}(p^k(\varpi|_Q) - me_Q^k)\right]$$

MEAN rule

Adapted with permission from: W. Yan and F. Xue (2008). Jet Engine Gas Path Fault Diagnosis Using Dynamic Fusion of Multiple Classifiers, *IJCNN 2008*, Hong Kong, pp-1585-1591

Performance evaluation

Performance indices

Overall accuracy: $$OAC = \sum_{i=1}^{C} CM(i,i) \Big/ \sum_{i,j=1}^{C} CM(i,j)$$

False positive rate: $$FPR = \sum_{j=2}^{C} CM(1,j) \Big/ \sum_{j=1}^{C} CM(1,j)$$

False negative rate: $$FNR = \sum_{i=2}^{C} CM(i,1) \Big/ \sum_{i=2,j=1}^{C} CM(i,j)$$

Example for C=7

i^{th} Classifier		Predicted Classes							pcAC
		NF	F1	F2	F3	F4	F5	F6	
True Classes	NF	2139	7	106	150	197	70	136	76.26
	F1	0	2786	13	0	2	0	4	99.32
	F2	118	7	2454	62	94	57	13	87.49
	F3	188	2	76	2210	291	16	22	78.79
	F4	288	2	81	317	1893	177	47	67.49
	F5	202	1	86	7	190	2292	27	81.71
	F6	77	1	4	8	37	28	2650	94.47
Classifier Performance Indices									
OAC=83.65%			TPR =23.74%				FNR =5.19%		

Performance evaluation

Stratified 5-fold cross-validation

Adapted with permission from: W. Yan and F. Xue (2008). Jet Engine Gas Path Fault Diagnosis Using Dynamic Fusion of Multiple Classifiers, *IJCNN 2008*, Hong Kong, pp-1585-1591

Application Example: Jet engine fault diagnosis

4.2 Example

What?

Why?

- Increases flight safety
- Prevents costly component damage and/or catastrophic failure
- Reduces turnaround time
- Reduces delays and cancellations
- Increases engine on-wing time

Characteristics

- Engine initial quality varies
- Engine quality deteriorates over time
- Engines are operated at different points in flight regime

Constraints

- Complexity of engine as a system
- Limited number of sensors allowed
- Noisy environment and noisy sensed data

More accurate and reliable fault diagnostic systems are the key

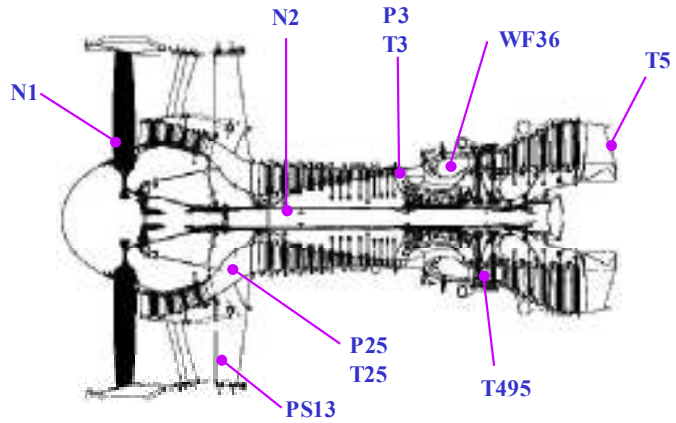
Adapted with permission from: W. Yan and F. Xue (2008). Jet Engine Gas Path Fault Diagnosis Using Dynamic Fusion of Multiple Classifiers, *IJCNN 2008*, Hong Kong, pp-1585-1591

Application Example: Jet engine fault diagnosis

4.2 Example

General information

- GEAE CFM56-7B engine for commercial aircrafts
- Simulated data
- 6 engine gas path faults
- 9 standard sensor parameters
- 5 levels of engine deterioration



6 engine gas path faults

- Fan fault (FAN)
- Compressor fault
- High Pressure Turbine (HPT) fault
- Low Pressure Turbine (LPT) fault
- Customer Discharge Pressure (CDP) fault
- Variable Bleed Valve (VBV) fault

12 parametric inputs

1.	fuel flow rate	7.	comp. inlet temp.
2.	fan speed	8.	comp. exit temp.
3.	core speed	9.	HP turbine exit temp.
4.	comp. inlet pressure	10.	LP turbine exit temp.
5.	comp. exit pressure	11.	5/4
6.	fan tip exit pressure	12.	1/5

Jet engine fault diagnosis is a 7-class classification problem

Adapted with permission from: W. Yan and F. Xue (2008). Jet Engine Gas Path Fault Diagnosis Using Dynamic Fusion of Multiple Classifiers, *IJCNN 2008*, Hong Kong, pp-1585-1591

Results

NN classifier

True Classes \ Predicted Classes	NF	FAN	CMP	HPT	LPT	CDP	VBV	pcAC
NF	2139	7	106	150	197	70	136	76.26
FAN	0	2786	13	0	2	0	4	99.32
CMP	118	7	2454	62	94	57	13	87.49
HPT	188	2	76	2210	291	16	22	78.79
LPT	288	2	81	317	1893	177	47	67.49
CDP	202	1	86	7	190	2292	27	81.71
VBV	77	1	4	8	37	28	2650	94.47

Classifier Performance Indices

OAC=83.65% **TPR =23.74%** **FNR =5.19%**

SVM classifier

True Classes \ Predicted Classes	NF	FAN	CMP	HPT	LPT	CDP	VBV	pcAC
NF	1912	0	181	253	237	75	147	68.16
FAN	0	2803	2	0	0	0	0	99.93
CMP	98	1	2512	60	77	52	5	89.55
HPT	183	0	107	2221	249	18	27	79.18
LPT	302	0	115	547	1563	229	49	55.72
CDP	197	0	105	4	184	2288	27	81.57
VBV	61	0	2	4	16	23	2699	96.22

Classifier Performance Indices

OAC=81.48% **TPR =31.84%** **FNR =5.00%**

DT classifier

True Classes \ Predicted Classes	NF	FAN	CMP	HPT	LPT	CDP	VBV	pcAC
NF	1773	3	160	320	316	84	149	63.21
FAN	2	2797	5	1	0	0	0	99.71
CMP	148	2	2250	117	169	111	8	80.21
HPT	319	0	181	1835	414	32	24	65.42
LPT	366	0	182	449	1436	304	68	51.19
CDP	187	0	106	35	302	2105	70	75.04
VBV	151	0	8	36	112	60	2438	86.92

Classifier Performance Indices

OAC=74.53% **TPR =36.79%** **FNR =6.97%**

Adapted with permission from: W. Yan and F. Xue (2008). Jet Engine Gas Path Fault Diagnosis Using Dynamic Fusion of Multiple Classifiers, *IJCNN 2008*, Hong Kong, pp-1585-1591

Results – cont'd

Overall accuracy: $$OAC = \sum_{i=1}^{C} CM(i,i) \Big/ \sum_{i,j=1}^{C} CM(i,j)$$

False positive rate: $$FPR = \sum_{j=2}^{C} CM(1,j) \Big/ \sum_{j=1}^{C} CM(1,j)$$

False negative rate: $$FNR = \sum_{i=2}^{C} CM(i,1) \Big/ \sum_{i=2,j=1}^{C} CM(i,j)$$

		Overall Accuracy (OAC)	False Positive Rate (FPR)	False Negative Rate (FNR)
Individual Classifiers	NN	83.65	23.74	5.19
	SVM	81.48	31.84	5.00
	CART	74.53	36.79	6.97
Fusion Methods	Fusion by Averaging	85.51	21.21	4.63
	Dynamic Selection	85.12	21.96	4.64
	Dynamic Fusion	86.04	18.72	4.92

Dynamic Fusion outperforms both baselines

Adapted with permission from: W. Yan and F. Xue (2008). Jet Engine Gas Path Fault Diagnosis Using Dynamic Fusion of Multiple Classifiers, *IJCNN 2008*, Hong Kong, pp-1585-1591

Summary

Dynamic fusion of multiple classifiers proposed in this study can be effective in improving performance of jet engine fault diagnostic systems and general classification problems as well.

Future work

Improve the dynamic fusion scheme by exploring different approaches for each of the three key components of the DF.

Apply DF to other real-world data.

Adapted with permission from: W. Yan and F. Xue (2008). Jet Engine Gas Path Fault Diagnosis Using Dynamic Fusion of Multiple Classifiers, *IJCNN 2008*, Hong Kong, pp-1585-1591

4.3. Case Study 3:

Fusion of (Competing) Predictive Models

Reference: "Fast Meta-models for Local Fusion of Multiple Predictive Models" Applied Soft Computing Journal, 2008, doi:10.1016/j.asoc.2008.03.006 – [GE GR Technical Report, 2007GRC832, Oct 12, 2007].

4.3 Fusion of (Competing) Predictive Models

- Prognostics Issues
- Prediction Problem: Motivation & Description
- Technical Challenge: Prediction Uncertainty
- Multiple Predictive Model Fusion
- Model Generation
- Spectrum of Weighting Schemes
  - Global
  - Local: Run-time Peers (Lazy Learning), Compiled (CART)
- Evaluation and Results
- Conclusions & Future Work

Prognostics (RUL Prediction)

- Prognostics is considered as Remaining Useful Life (RUL) Prediction
- Typically run-to-failure data is not widely available (due to its operational cost)
- As a result RUL prediction tends to have high variance
- Fusion of multiple predictive models is one way to decrease such variance
- Usually multiple prediction models include combination of physics-based models and data-driven models

For a variety of reasons, such as background complexity, proprietary information, and time constraints, in this tutorial ***we will focus on a prediction problem*** *for optimization,* ***as a surrogate for the prognostics problem.***
The problem will be solved using a dynamic fusion of predictive models

Prediction Problem: Motivation

Primary Market Opportunity

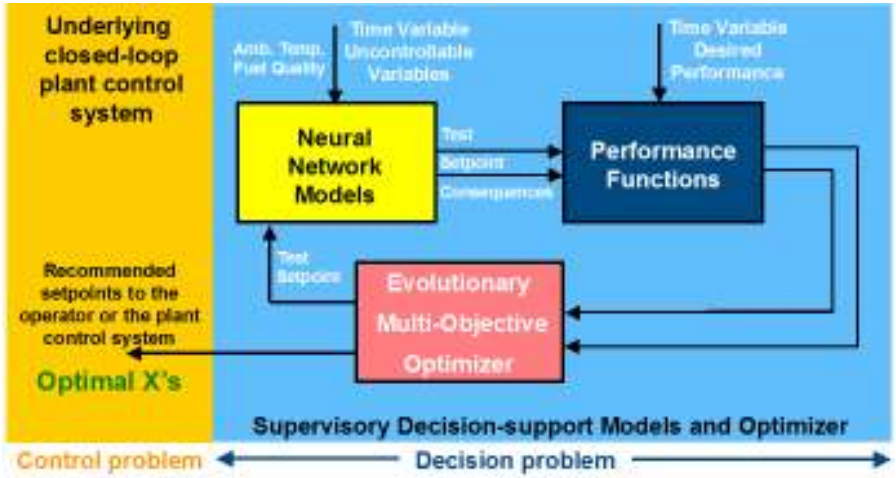
- 700+ coal-fired boilers > 100 MW in U.S.A.
- 2000+ coal-fired boilers > 100 MW outside U.S.A.

Secondary Market Opportunity

- Industrial boilers worldwide
- Coal, gas, oil, combined cycle

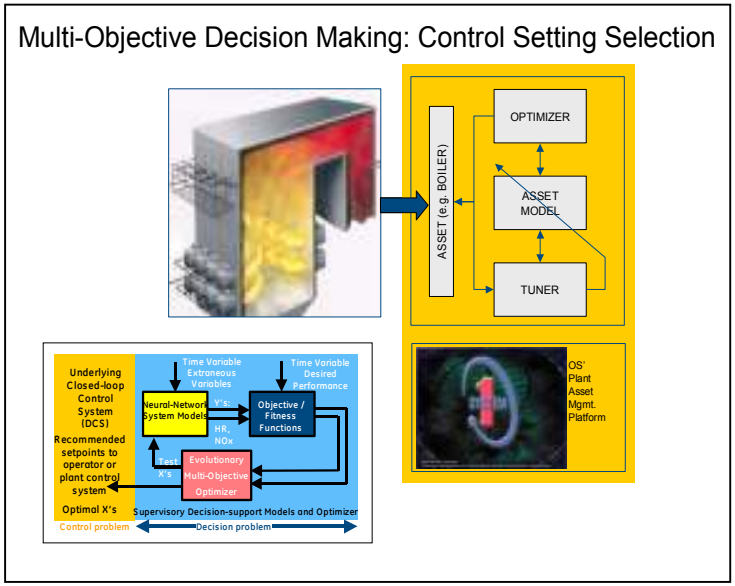
Emphasis on Environmental Issues

- Clean Air Act – U.S.A.
- Kyoto Protocol



Significant market need for model-predictive optimization of coal-fired boilers

Prediction Problem: Motivation (cont.)



EA (EMOO) + NN + Fuzzy + Fusion

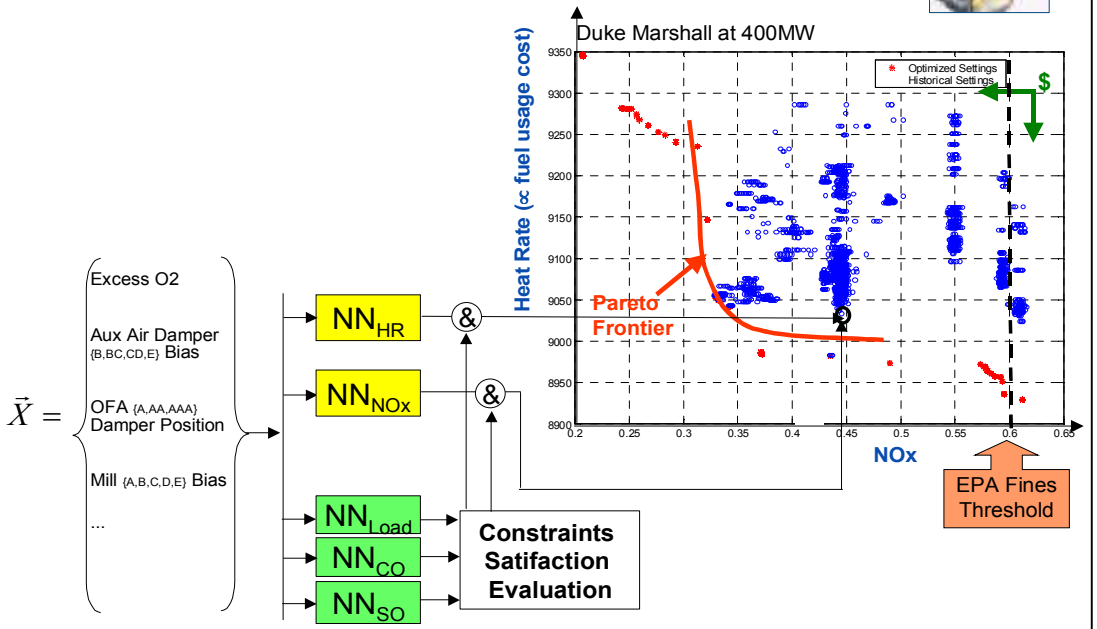
- ***Key Goal:*** *Generate hierarchical control settings for Coal-Fired boiler to*
  - ***Reduce emissions*** *(NOx) and* ***Decrease fuel cost*** *(Heat Rate),*
  - *while satisfying all operational constraints (load, CO, SO, etc.)*
- ***Key Challenge:***
  - ***Prediction Uncertainty*** *[due to model extrapolation from training set image]*

- ***Solution:***

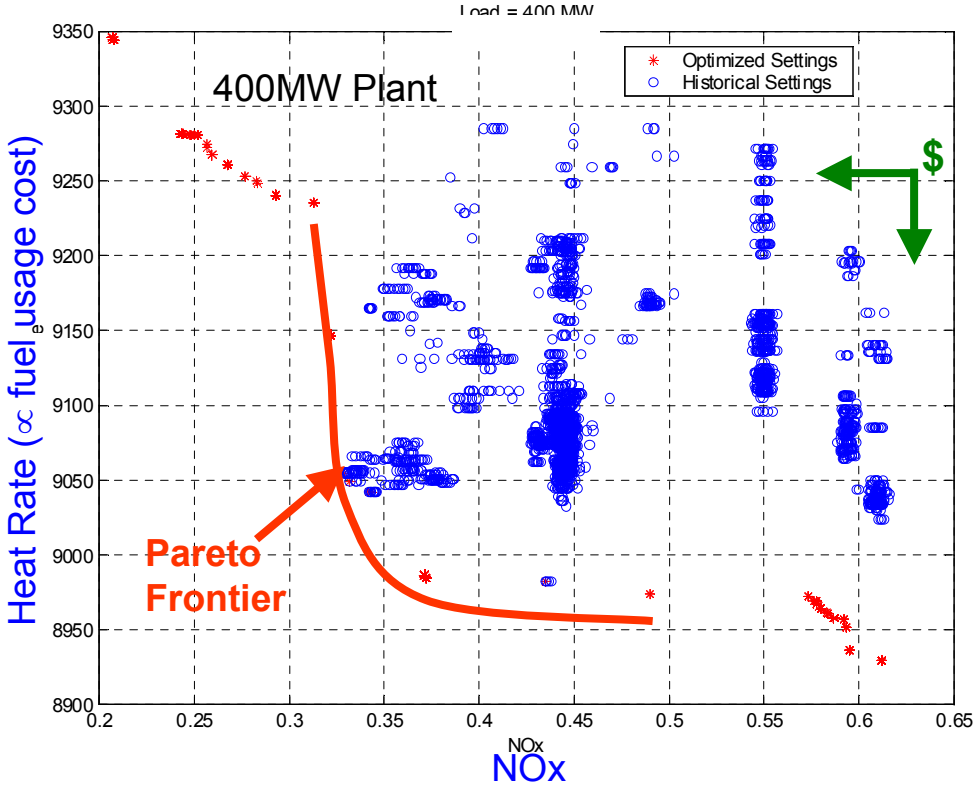
⇨ *Use* ***fusion of predictive models*** *(hybrid first-principle and data-driven NN's) to determine (HR , NOx) coordinates of given setting.*

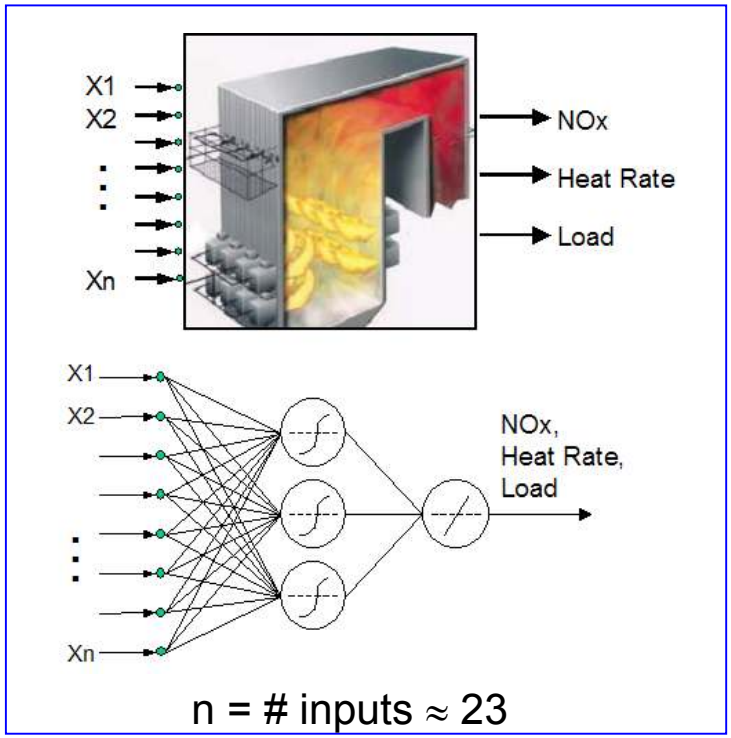
- Use of ***Evolutionary Multi-Objective*** *Search to generate Pareto Surface in (HR , NOx)*

Pareto Front for Heat Rate and NOx



Prediction Problem: Description





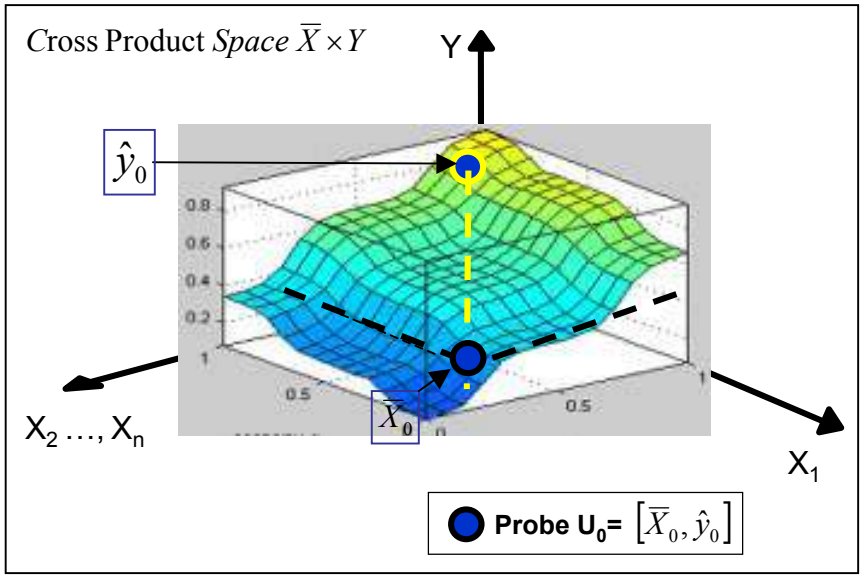
- **Pareto Frontier ➔ best achievable operation**
- **Goal: Optimize NOx**
  Optimize Heat Rate (∝ fuel usage cost)
  Operate the boiler at Pareto frontier

- **Model-predictive multi-objective optimization**
- **Neural networks, evolutionary algorithms**
- **Needed: High fidelity predictive models**
- **Modeling: NOx, Heat Rate, Load**

Technical Challenge: Prediction Uncertainty

$$t(\vec{x}) = f(\vec{x}) + \varepsilon(\vec{x})$$

Function approximation model



Data noise $\varepsilon(\vec{x})$

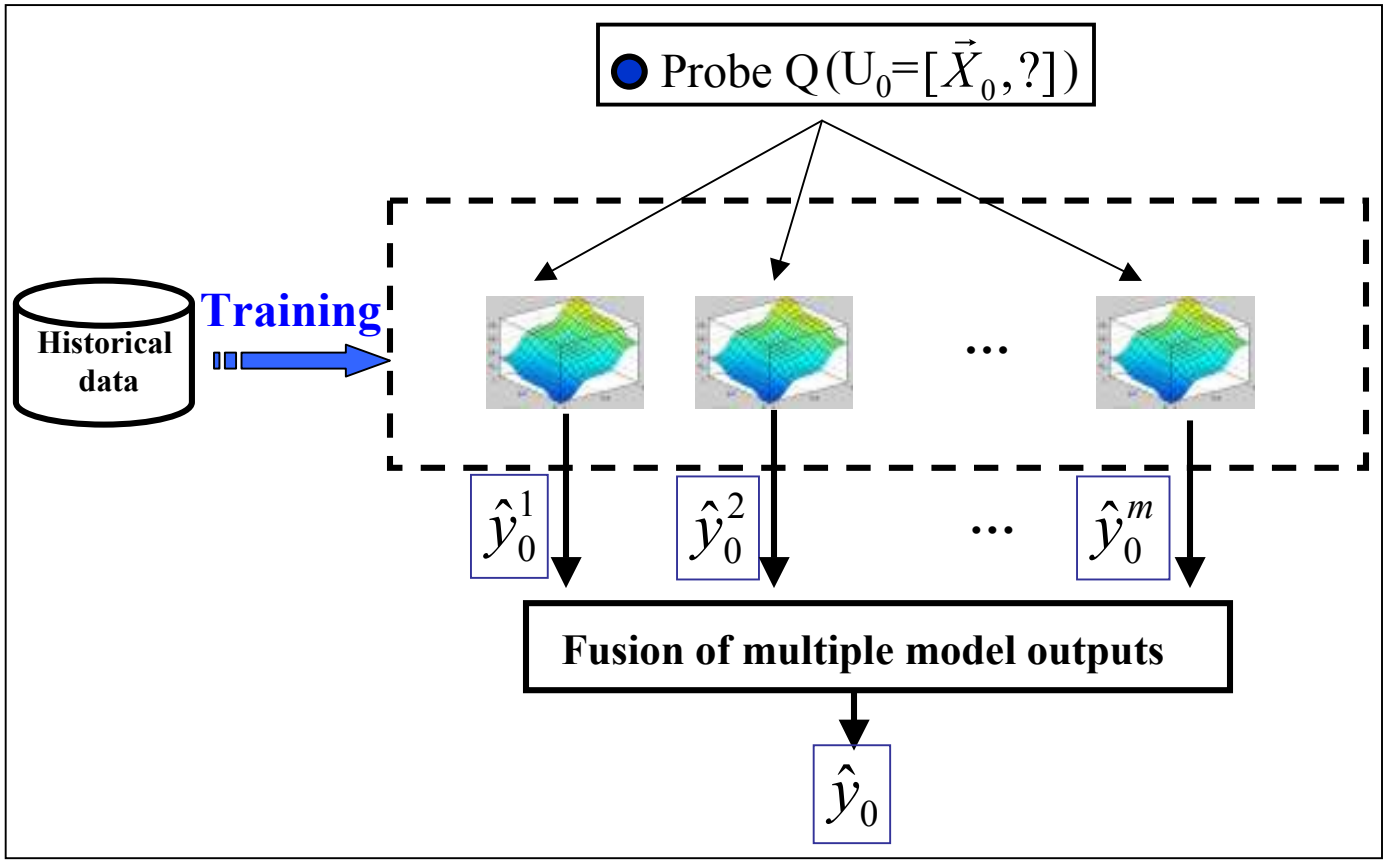
Model parameter misspecification

- Variation due to randomly sampling of training dataset
- Non-deterministic training results
- Varying initial conditions

Model structure misspecification

- e.g. not enough neurons
- Misspecification of regression models

Multiple Predictive Model Fusion



Effective fusion of multiple diverse models can boost accuracy

1. **Active literature in multiple classifier fusion (voting, weighting, fuzzy, Dempster-Shafer)**
2. **Ensemble of neural networks (Stacked generalization, decorrelated neural networks)**

Need for Locally Weighted Fusion

System model $f(\vec{x})$ obtained from a given data set *D*: $f(\vec{x};D)$

Mean square error over all possible dataset *D*:

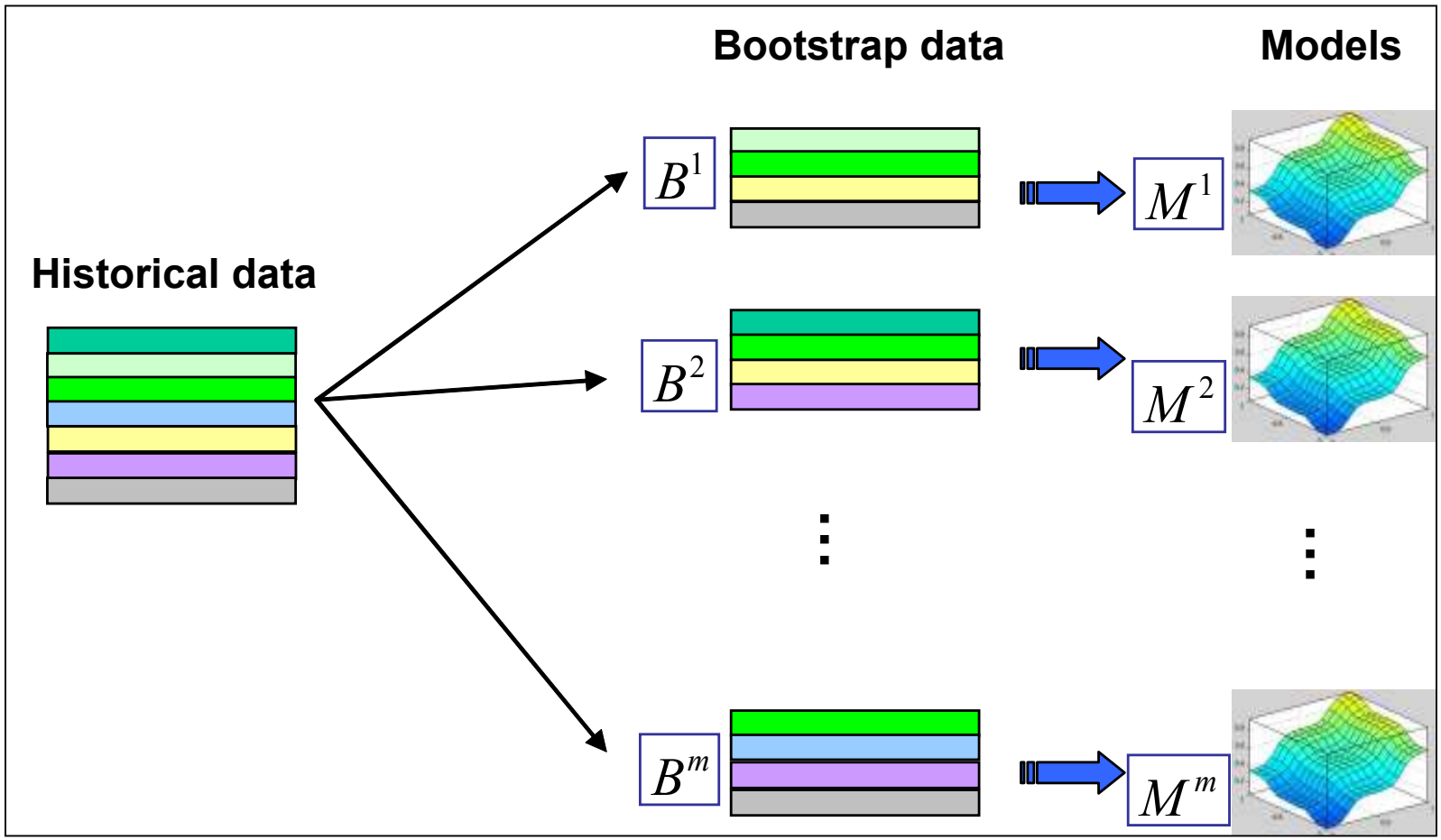
$$E_D[(f(\vec{x};D)-t(\vec{x}))^2] = (E_D[f(\vec{x};D)]-t(\vec{x}))^2 + E_D[(f(\vec{x};D)-E_D[f(\vec{x};D)])^2]$$

Bias

Variance

1. **Mean square error depends on the position of query points in the inputs space**
2. **Averaging will not eliminate bias error**

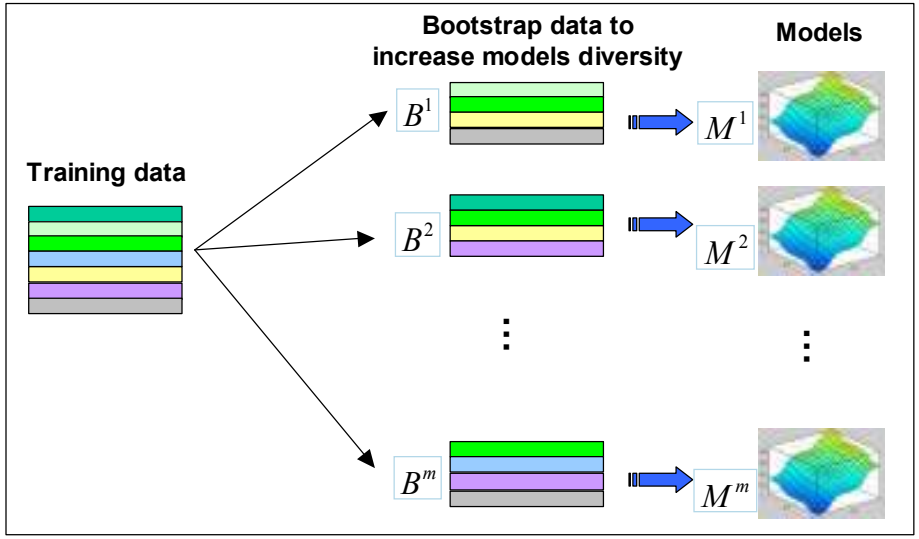
Weighted Fusion – Model Generation



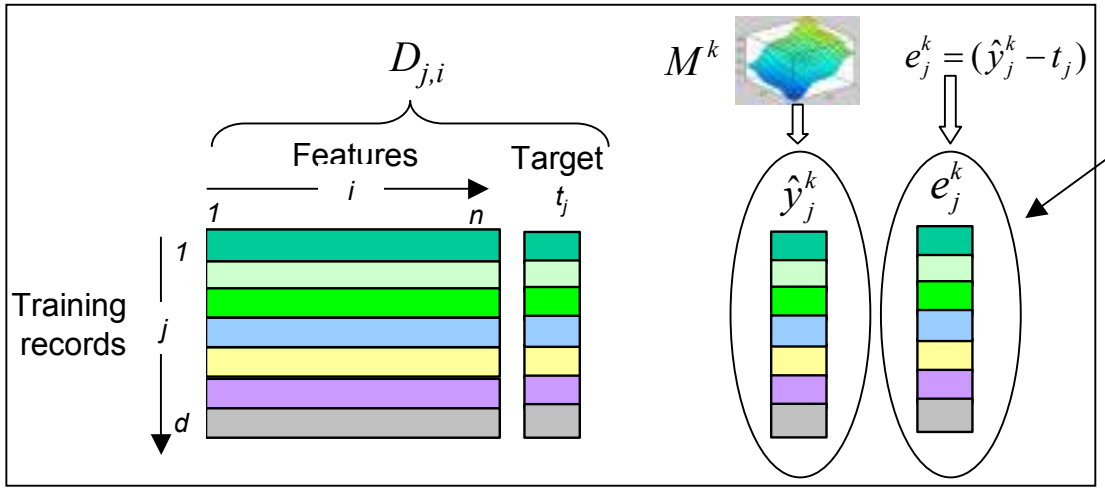
Train *m* models from same historical dataset using Bootstrap

Weighted Fusion – Model Generation (cont.)

Models: M^k, $k = 1, \ldots m$



Training Data: $D_{j,i}$ $j = 1, \ldots d$; $i=1, \ldots, n$



Since each model has been trained in the **best** possible way, the quality of the fusion will depend on selecting the **correct fusion meta-model *structure* & *parameters*.**

Let's start with on the parameters (***weights***)

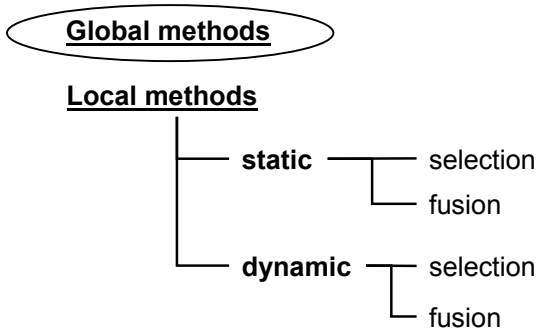
Each weight reflects the **degree to which we want each model to contribute to the fusion**. We want the most reliable models to have the highest weights, and vice versa.

Thus the computation of the weights will be based on the ***historical errors*** that each model has shown during training.

The training data should then be update by actual runs, models errors should be updated accordingly, once ground truth is available, and weights should be recomputed to avoid obsolescence.

By features, we mean the inputs used to train the individual models (e.g., altitude, speed, pressure, temperature,, derivatives of above vars., etc).

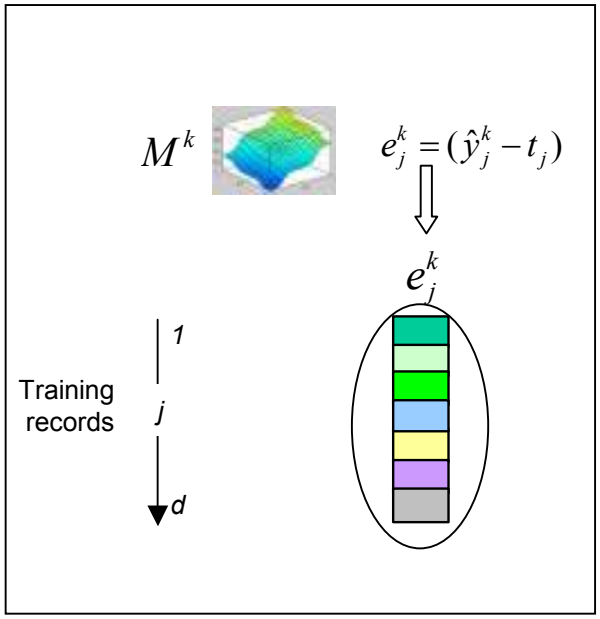
Spectrum of Weighting Schemes (Global)



The most important aspect in selecting the weights is to determine which training records to use

For offline fusion model generation we have many choices, depending on the tradeoff between accuracy and complexity:

(1) Large granularity → Global Weight (for each model)
Use all *d* training records



$$Weight\left(\hat{y}_j^k\right) = K^k = 1/mae^k$$

where

$$mae^k = \frac{\sum_{j=1}^{d} |e_j^k|}{d}$$

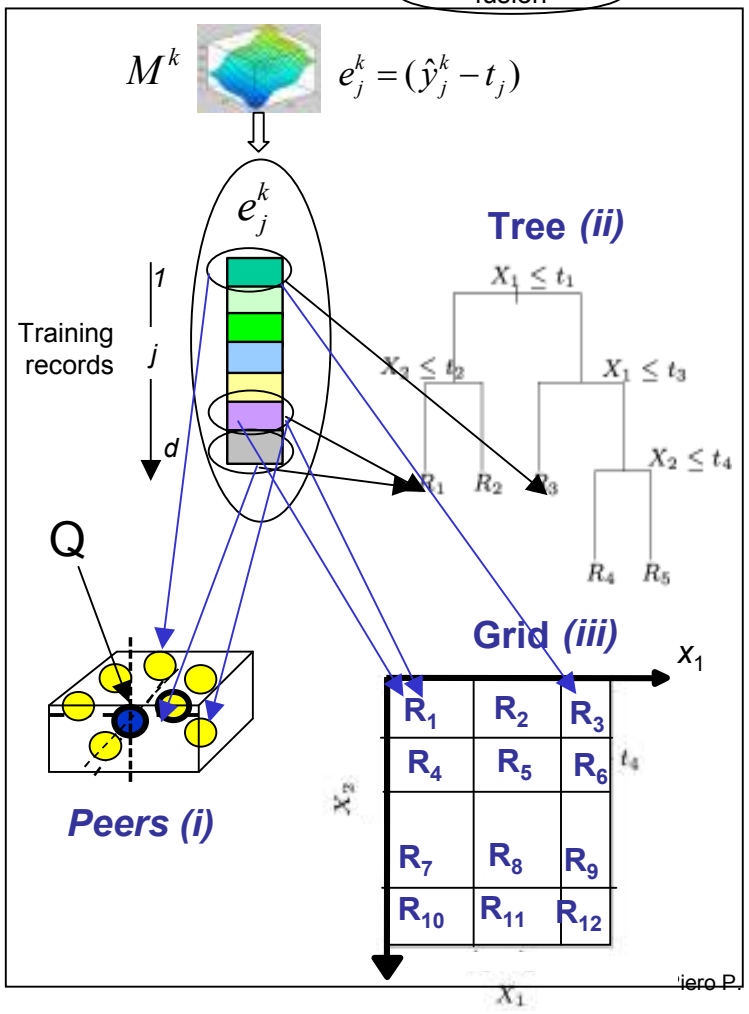
Mean Absolute Error for model k
$k=1, \ldots, m$

Spectrum of Weighting Schemes (Local -cont.)

Global methods

Local methods

- **static**
  - selection
  - fusion
- **dynamic**
  - selection
  - fusion



(2) Small granularity – Automatically select a subset of training records – *With Learning* (dynamic)

Quantitative Description: Use domain knowledge or expert knowledge to determine [using mathematical expressions] regions of the feature space in each model should have a different weight:

Region R_i should be assigned weight K_i

Run-time region generation

(i) Run-time Peers – Lazy Learning

(using hyper-rectangles or hyper-spheres around Q)

Compiled-time region generation

(ii) Trees *(using inequalities)*

(iii) Grids (using intervals) -> exp. complexity

Qualitative Description: [using linguistic expressions]

(iv) Fuzzy Partitions (using fuzzy rules)

-> exp. complexity

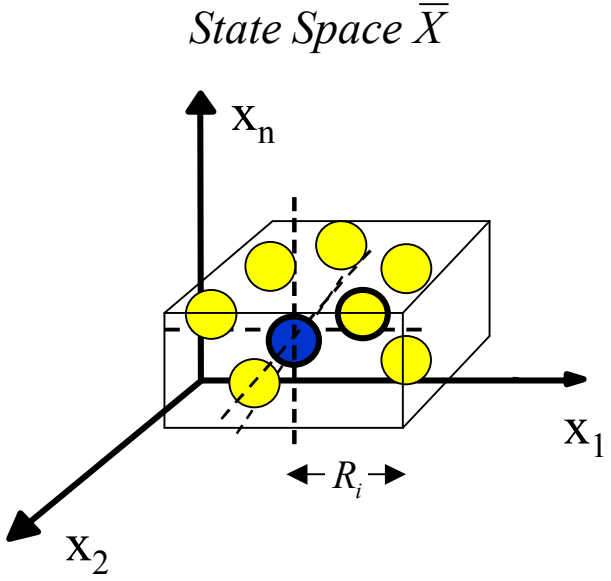
Not covered

Run-time: Lazy Learning → Predictive model weights (2 i)

Manage complexity by leveraging contextual knowledge
Problem decomposition using
- Lazy Learning (Run-tim Peers)

The model is in the data
+ no model to train
- need to compute region for every query Q
- curse of dimensionality

Lazy Learning Fusion – Step 1 of 3



Probe Q

Peer u_j $u_j \in P(Q)$

Peers (Q) $= P(Q)$

$P(Q) = \{u_j, j = 1,...,m \mid u_j \in N(Q)\}$

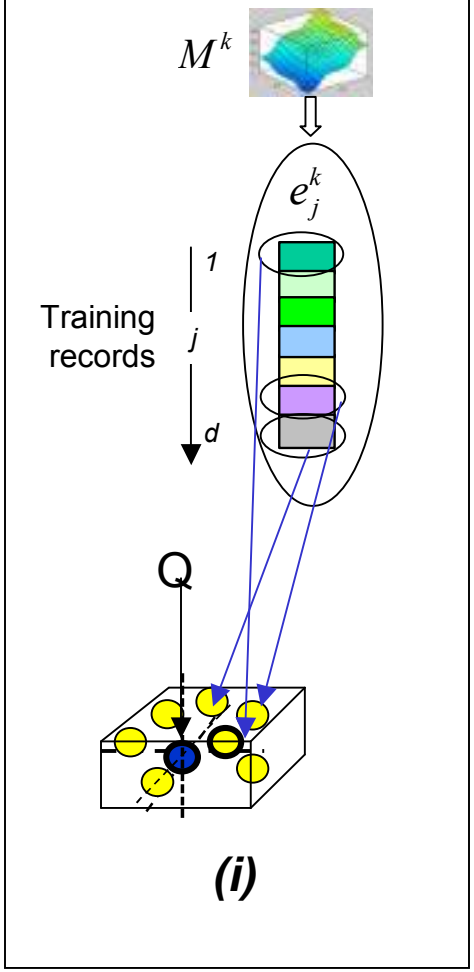
# of peers of (Q) $= N_Q$

$N_Q = |P(Q)|$

Neighborhood (Q) = N(Q)

$N(Q) = \|x_{i,Q} - x_{i,j}\| < R_i \quad i = 1,...,n$

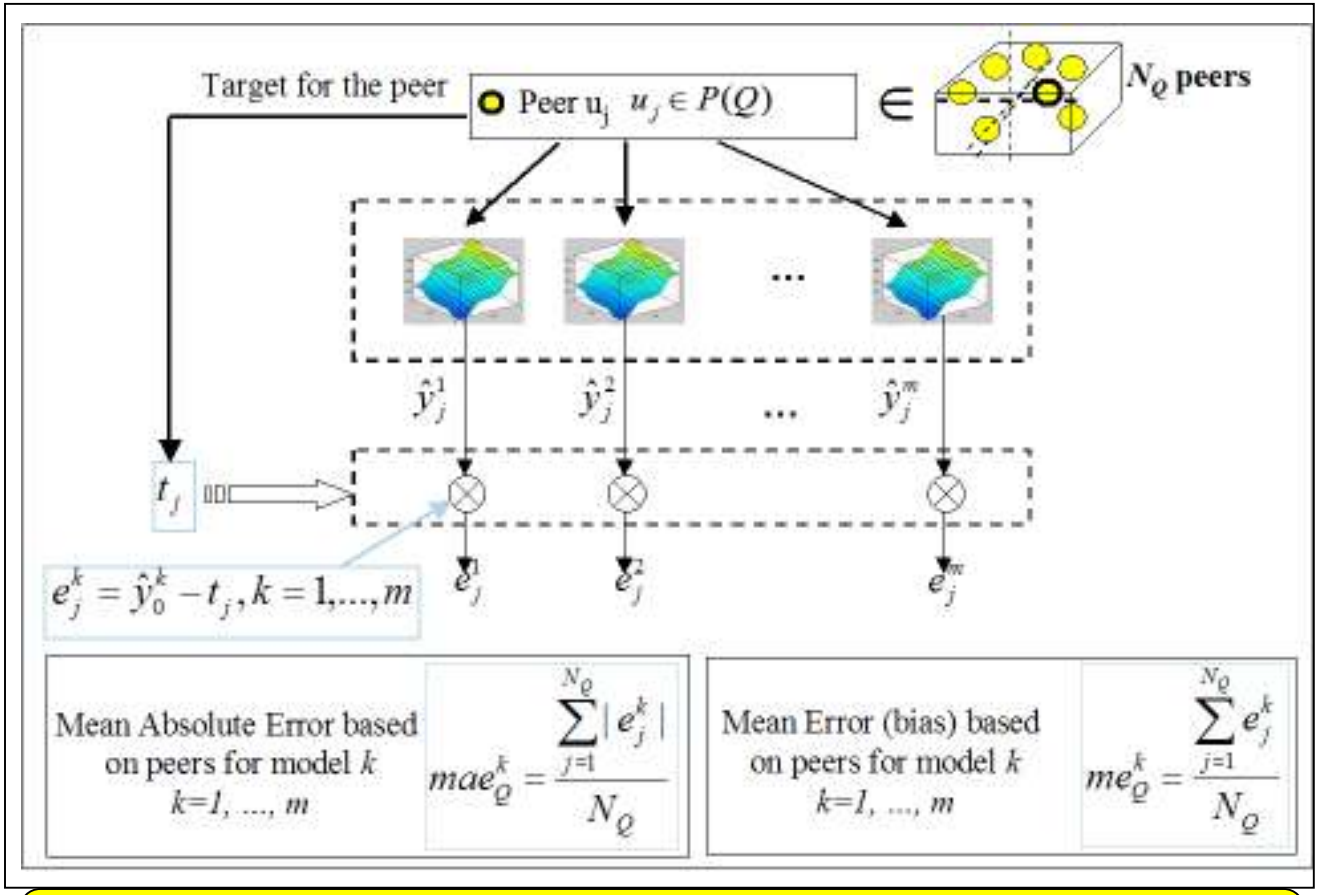
where $R_i = 5\% X_i$



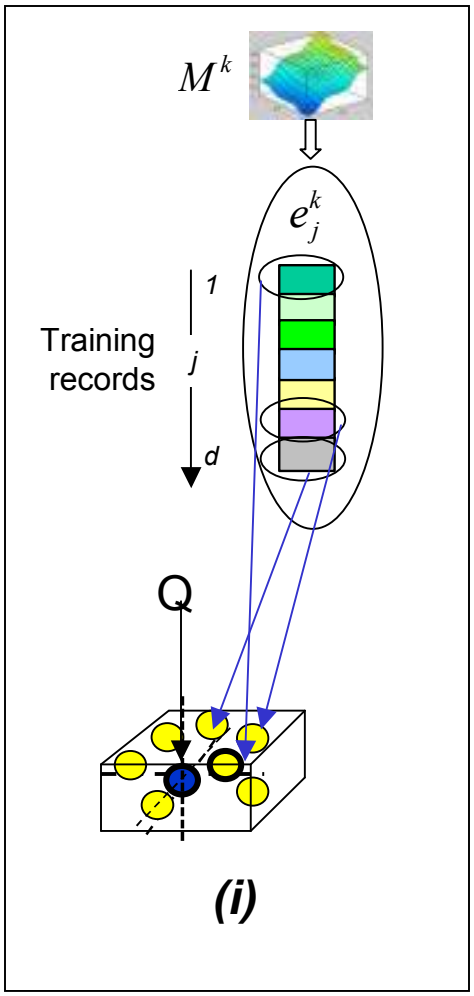
Retrieve historically similar neighbors/peers of probe Q

Run-time: Lazy Learning → Predictive model weights (2 i)

Lazy Learning Fusion – Step 2 of 3

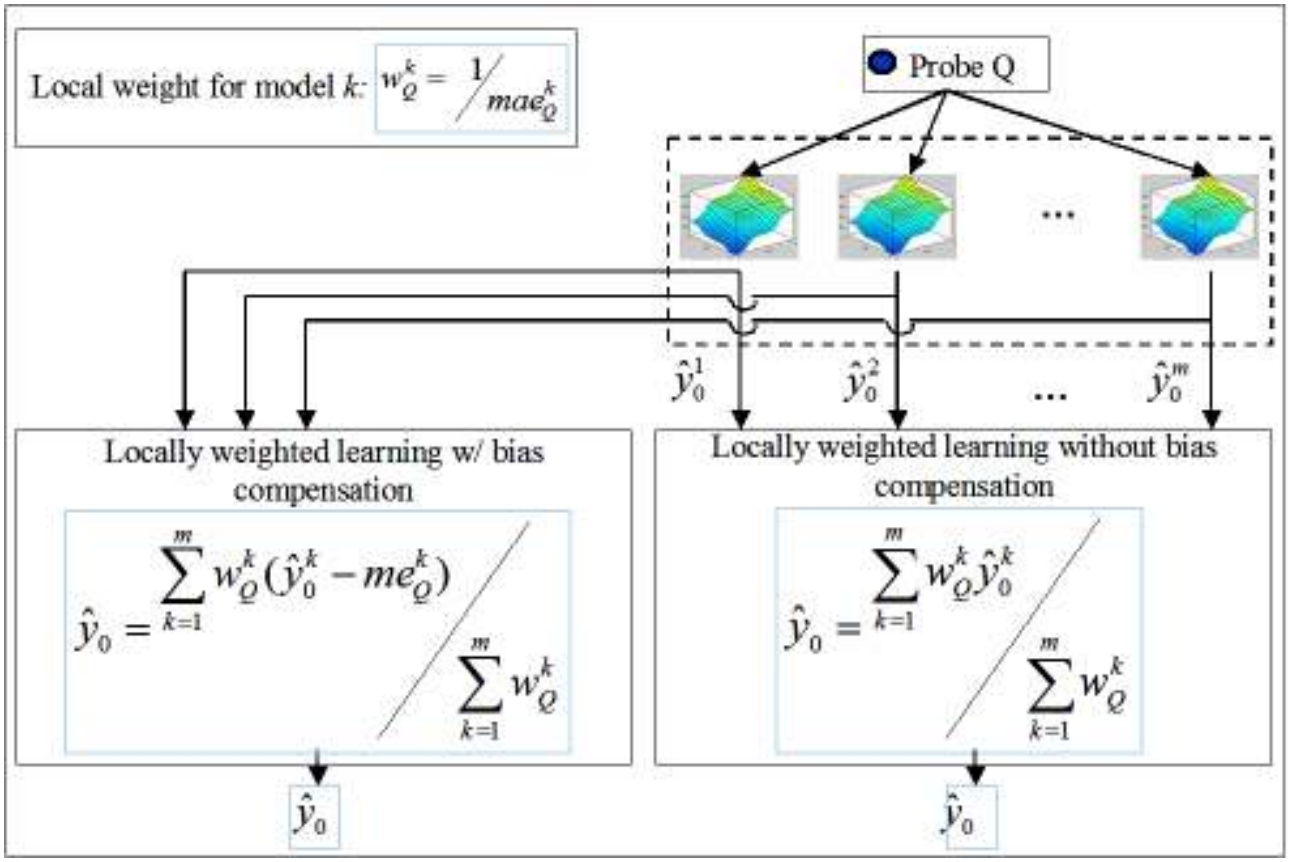


Evaluate local performance of each of the *m* predictive models using their training sets

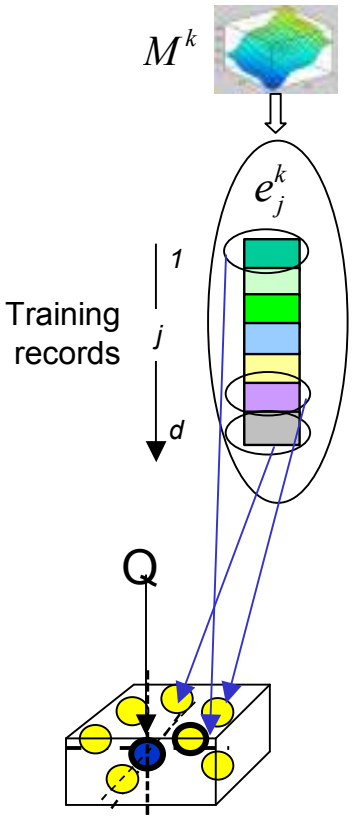


Run-time: Lazy Learning → Predictive model weights (2 i)

Lazy Learning Fusion – Step 3 of 3



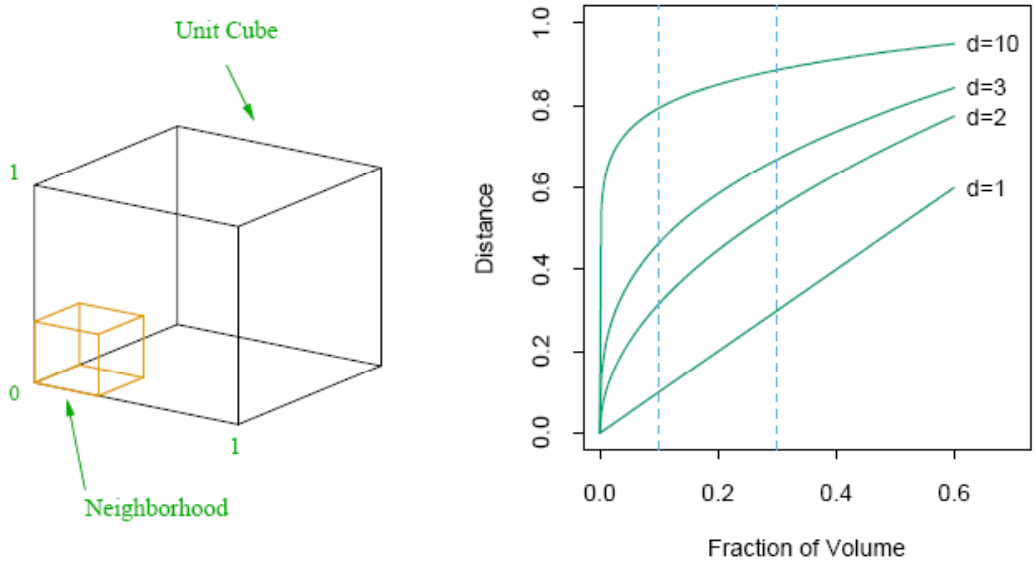
Aggregation of multiple predictions based on local performance



(i)

Curse of dimensionality

(for Lazy Learning type of models – e.g. kernel based models)



*The figure on the right shows the side-length of the sub-cube needed to capture a fraction **r** of the volume of the data, for different dimensions **n**.*

…In ten dimensions we need to cover 80% *of the range of each coordinate to capture* 10% *of the data.*

***Source**:* **The Elements of Statistical Learning -** Data Mining, Inference, and Prediction, Chapter 2, Figure 2.6
Trevor Hastie, Robert Tibshirani, Jerome Friedman, Springer

Local Methods need to reduce their dimensionality to avoid this "curse":
*→ dimensionality reduction via **transformation** (e.g. PCA)*
*→ dimensionality reduction via **subset selection** (e.g. CART)*

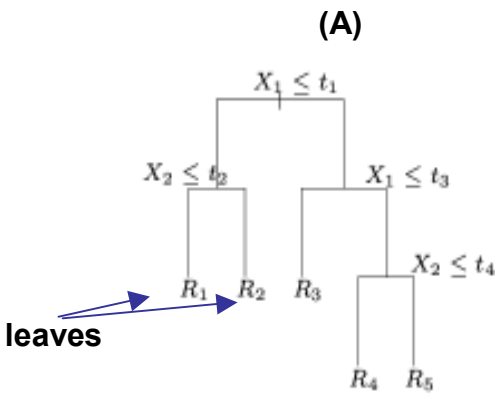
Compile-time: CART Tree → Predictive model weights (2 ii)

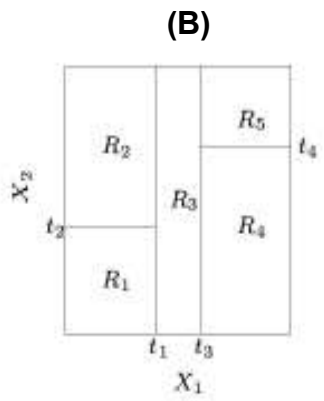
Manage complexity by leveraging contextual knowledge
Problem decomposition using:
- CART (Classification Analysis and Regression Tree)

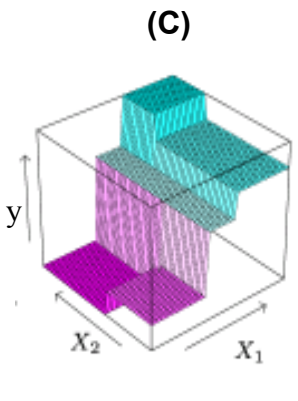
(A), (B), (C), and (D) are equivalent representations

(C) and (D) assume that each output Y in a region is constant, i.e., $Weight(R_i) = K_i$

CART (Classification Analysis and Regression Tree)





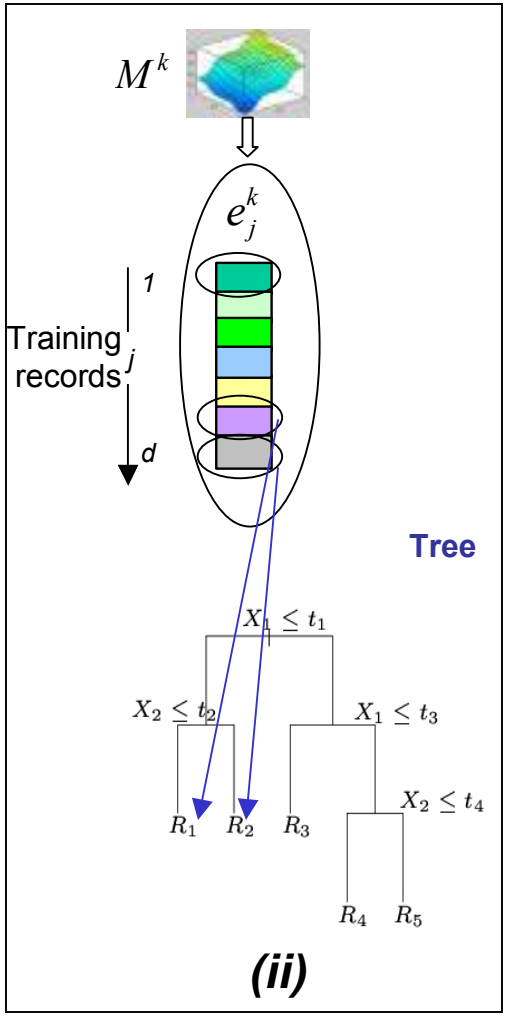


(D)

x_1	x_2	y
$\le t_1$	$\le t_2$	K_1
$\le t_1$	$> t_2$	K_2
$(t_1, t_3]$	don't care	K_3
$> t_3$	$\le t_4$	K_4
$> t_3$	$> t_4$	K_5

Context

Local Models = Local Precomputed Weights



*Dimensionality reduction via **subset selection** (using CART)*

Compile-time: CART Tree → Predictive model weights (2 ii)

Each Model will have a TREE $\mathbf{T_k}$

CART **minimizes the sum of the variances of the leaves**, so there is no need to compute weights as the inverse of mean absolute errors (or the inverse of the variance) as in case *3i*

So the constants K_i assigned to each region (a leaf in the tree) are just the **biases** to be used for that model, when the input Q falls in region R_i

1) Pre-compile the bias of each region: For each model k compute the **bias** as the Mean Error of all the points (residuals) in region $R_{i,.}$

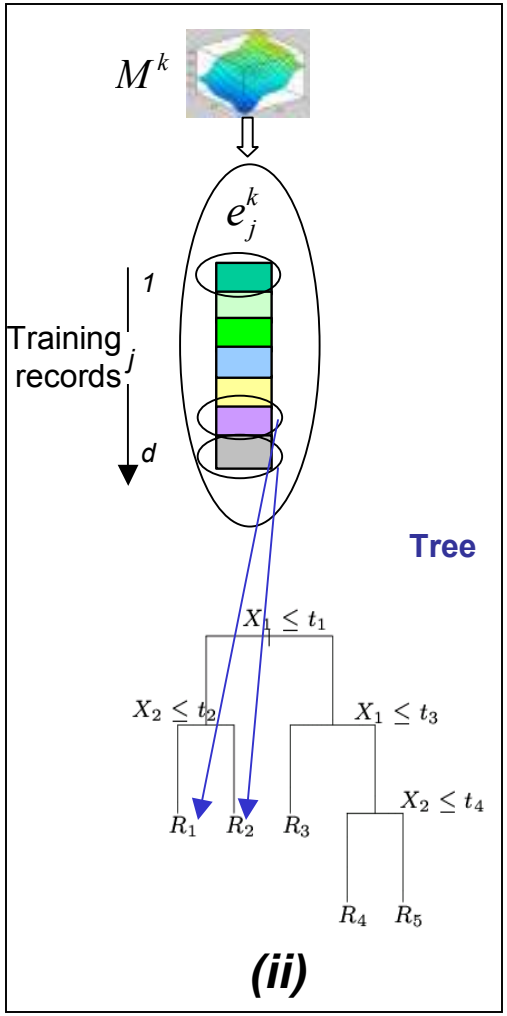
$$b_{R_i}^k = \frac{\sum_{j=1}^{|R_i|} e_j^k}{|R_i|}$$

2) Determine which bias to use: For each model k, for a given query Q, find the region R_Q (leaf node in tree $\mathbf{T_k}$) to which Q belongs:

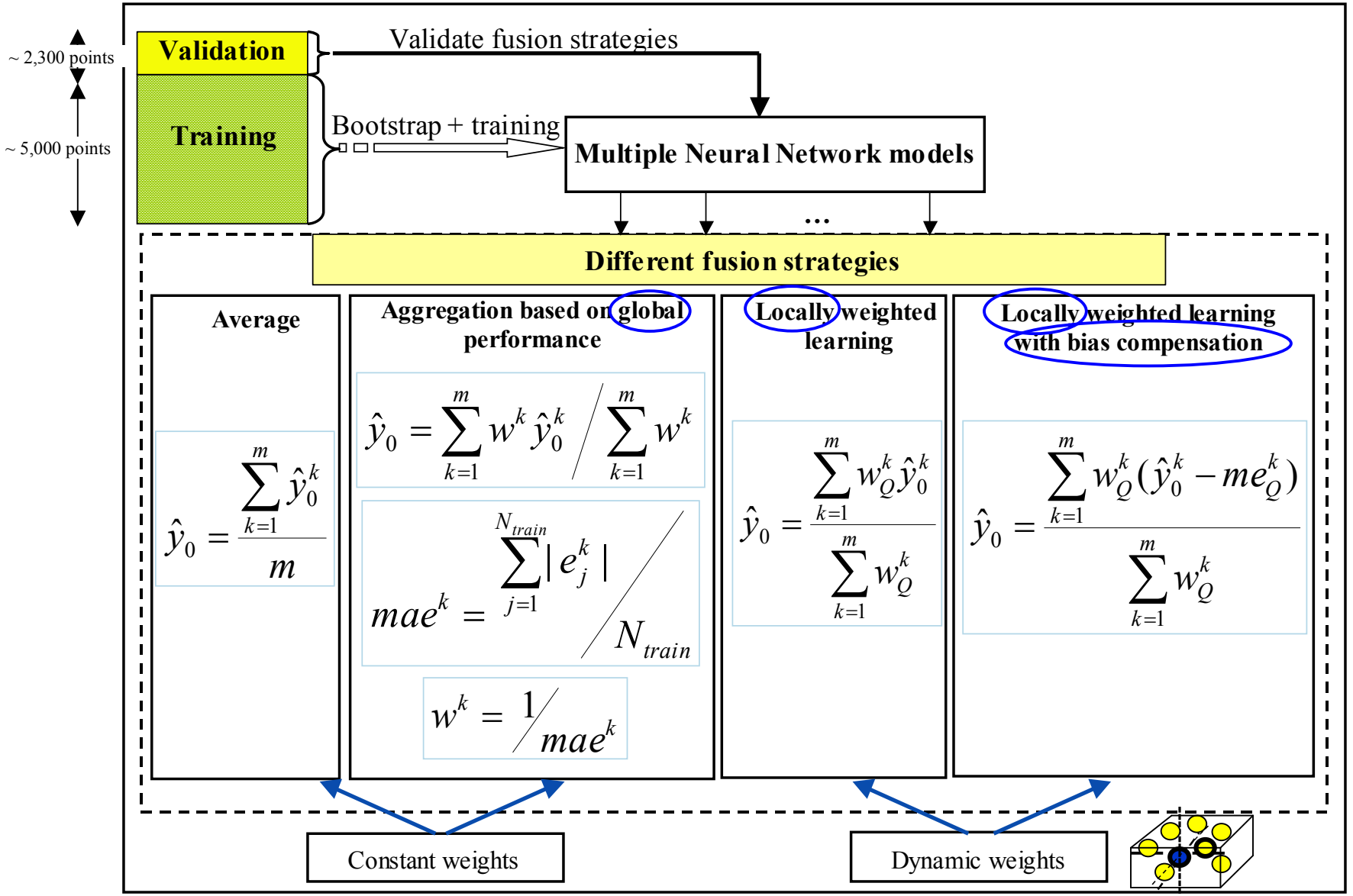
$$b_Q^k = b_{R_i}^k \text{ such that } Q \in R_i$$

3) Apply bias removal to output of model k, and average over all the models *(considering all weights* w_Q^k *equal to 1)*

$$\hat{y}_0 = \frac{\sum_{k=1}^{m} w_Q^k \left(\hat{y}_0^k - b_Q^k\right)}{\sum_{k=1}^{m} w_Q^k} \approx \frac{1}{m} \sum_{k=1}^{m} \left(\hat{y}_0^k - b_Q^k\right)$$



Fusion Performance Evaluation



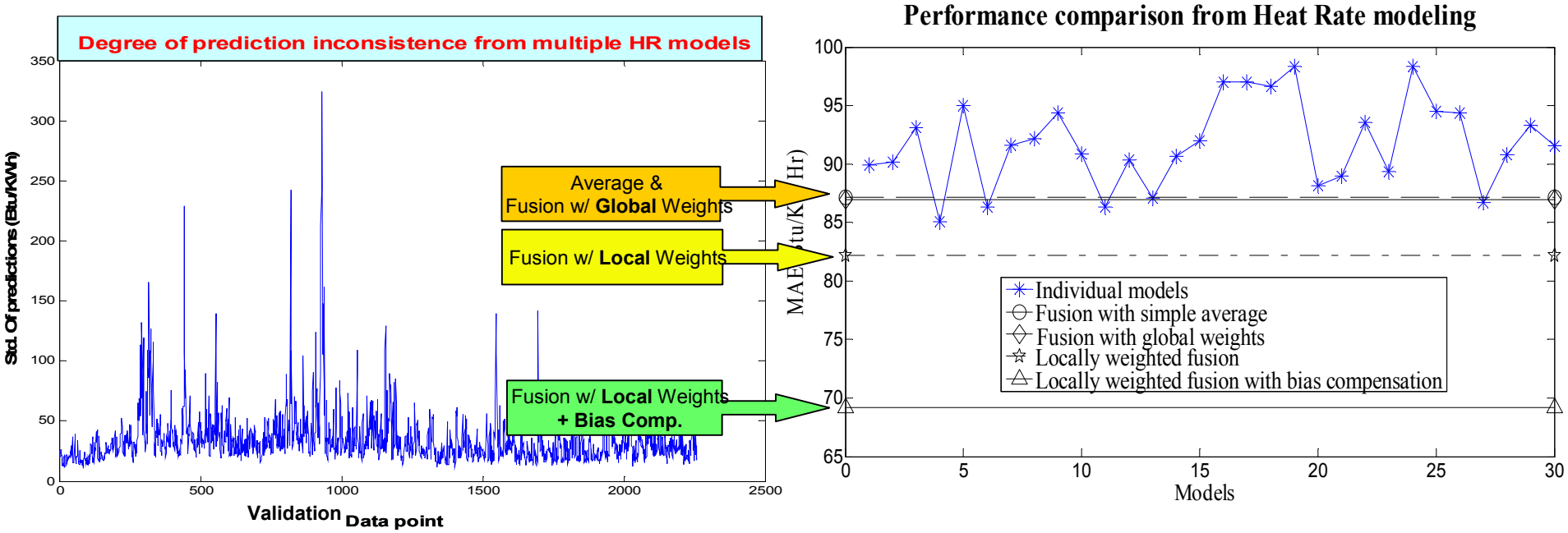
Results – Fusion Perf. Evaluation (with Hyper-rectangles)

Settings: 30 NN models trained, performance based on **validation dataset**

Performance metric: MAE (mean absolute error) over validation dataset

Baseline: average MAE of 30 MAEs from singleton NN model

Fusion performance gain: $$pg_{fs} = \frac{MAE_{fs} - baseline}{baseline}$$



Large model to model variation for each individual query point
Locally weighted fusion takes advantage of such discrepancy

Summary of Results Fusion Perf. Evaluation
(with Hyper-rectangles)

4.3 Experiment Results

			Heat Rate		NOx *(lb/MBtu)*		Load *(MW)*	
	Fusion Strategy		*MAE*	*pg*	*MAE*	*pg*	*MAE*	*pg*
		Baseline	91.79	0.00%	0.0228	0.00%	1.050	0.00%
	fusion with global information	*Simple Average*	87.15	5.00%	0.0214	6.00%	1.042	0.76%
(1)		*GWF*	86.91	5.30%	0.0214	6.00%	1.040	1.80%
	hyper-rectangle weights w bias	*LWF*	82.19	10.46%	0.0202	11.40%	1.024	2.48%
(3i)		*LWF+bias*	69.20	24.61%	0.0140	38.60%	0.855	18.57%

MAE: mean absolute error

pg: performance gain → $pg_{fs} = \frac{MAE_{fs} - baseline}{baseline}$

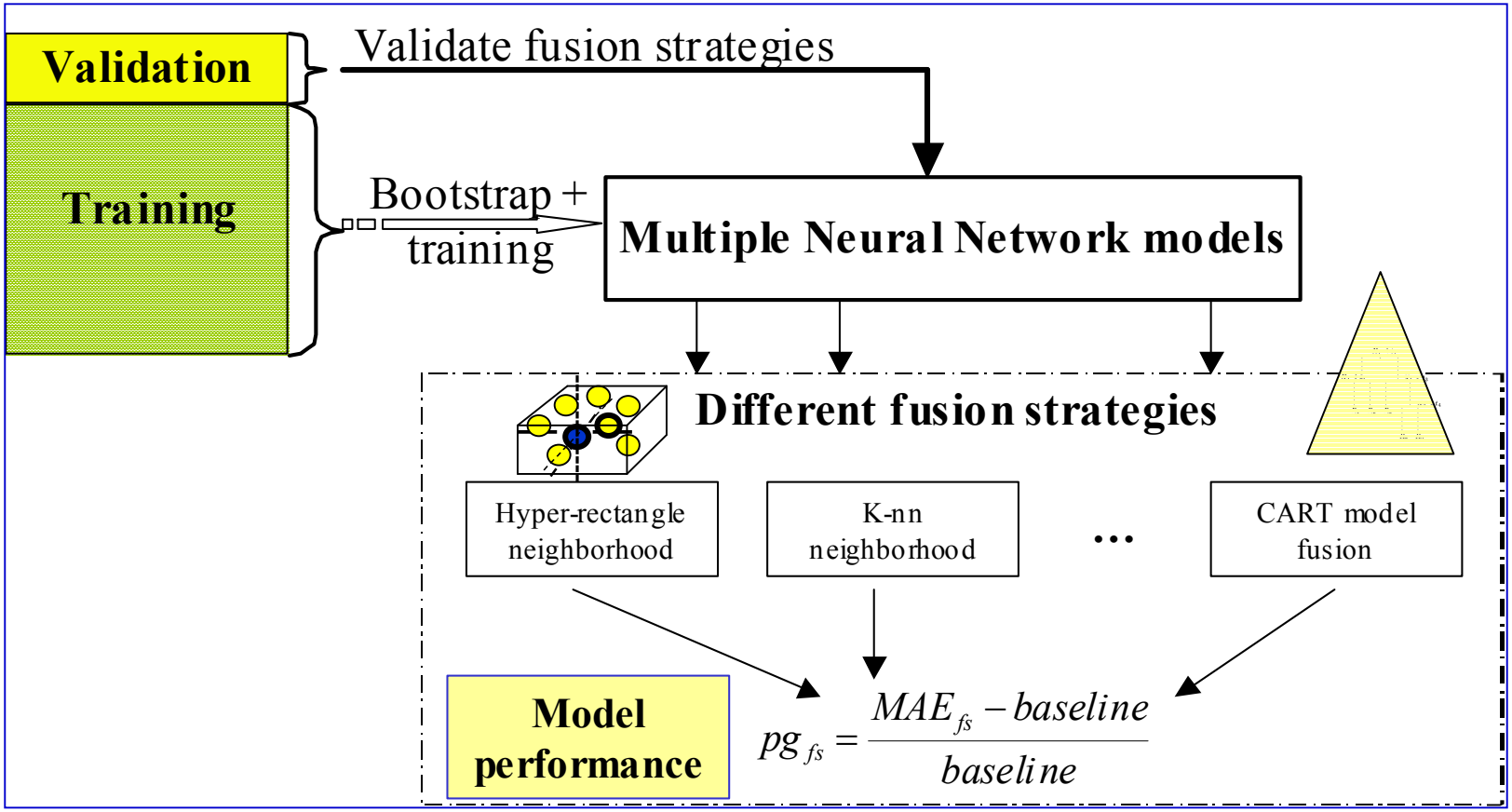
GWF: globally weighted fusion

LWF: locally weighted fusion

LWF+bias: locally weighted fusion with bias compensation

Locally weighted fusion (hyper-rectangle) with bias compensation boosts performance 18~38% over baseline

Additional Experiments Training & Validation



Predictive Model Fusion - Results

4.3 Experiment Results

Baseline

Best Predictor

(1) Global Weights

(3i) Lazy Learning (Hyper-Rectangles)

(3ii) CART

Fusion Strategy		Heat Rate *(Btu/KWHr)*		NOx *(lb/MBtu)*		Load *(MW)*	
		MAE	*pg*	*MAE*	*pg*	*MAE*	*pg*
	Baseline 1 (average of 30 predictors)	91.79	0.00%	0.0228	0.00%	1.050	0.00%
	Baseline 2 (best of 30 predictors)	85.10	7.29%	0.0213	6.58%	0.987	6.00%
Global neighborhood	*Average*	87.15	5.06%	0.0214	6.14%	1.042	0.78%
	GWF	86.91	5.32%	0.0214	6.14%	1.040	0.95%
	Least square weight	83.05	9.52%	0.0200	12.28%	0.984	6.29%
Hyper-rectangle neighborhood + Weight	*LWF*	82.19	10.46%	0.0202	11.40%	1.024	2.48%
	LWF+bias	69.20	24.61%	0.0140	38.60%	0.855	18.57%
Hyper-rectagle neighborhood	*Average*	87.15	5.06%	0.0214	6.14%	1.042	0.78%
	Average + bias	69.23	24.58%	0.0140	38.60%	0.855	18.56%
Hyper-rectangle neighborhood + Weight w/o bias	*LWF*	83.93	8.56%	0.0208	8.77%	1.030	1.93%
	LWF+bias	69.16	24.66%	0.0140	38.60%	0.854	18.63%
1nn neighborhood + Weight	*LWF*	81.19	11.55%	0.0214	6.14%	1.008	4.02%
	LWF+bias	72.99	20.48%	0.0143	37.28%	0.861	17.98%
5nn neighborhood + Weight	*LWF*	84.31	8.15%	0.0206	9.65%	1.029	1.97%
	LWF+bias	76.34	16.83%	0.0169	25.88%	0.903	14.04%
CART model	*LWF*	82.12	10.53%	0.0201	11.84%	1.022	2.67%
	LWF+bias	68.56	25.31%	0.0148	35.09%	0.817	22.19%
CART model	*Average*	87.15	5.06%	0.0214	6.14%	1.042	0.78%
	Average+bias	60.45	34.14%	0.0117	48.68%	0.720	31.38%

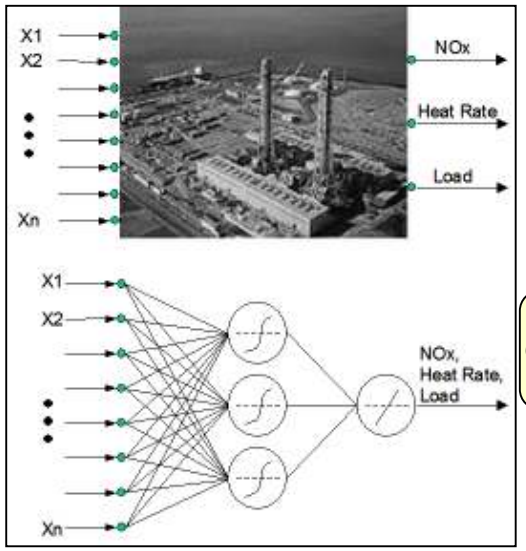
MAE: mean absolute error
pg: performance gain
GWF: globally weighted fusion
LWF: locally weighted fusion
LWF+bias: locally weighted fusion with bias compensation

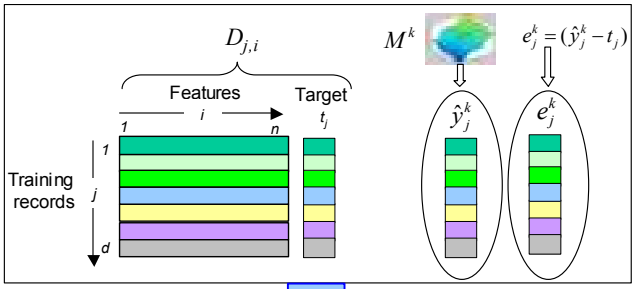
$$pg_{fs} = \frac{MAE_{fs} - baseline}{baseline}$$

CART- segmented fusion with bias compensation boosts performance 31~48% over baseline

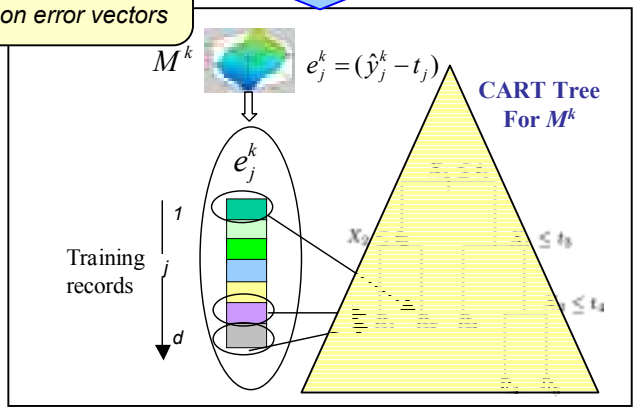
Reference: "Fast Meta-models for Local Fusion of Multiple Predictive Models" Applied Soft Computing Journal, 2008, doi:10.1016/j.asoc.2008.03.006 – [GE GR Technical Report, 2007GRC832, Oct 12, 2007].

Predictive Model Fusion: Summary

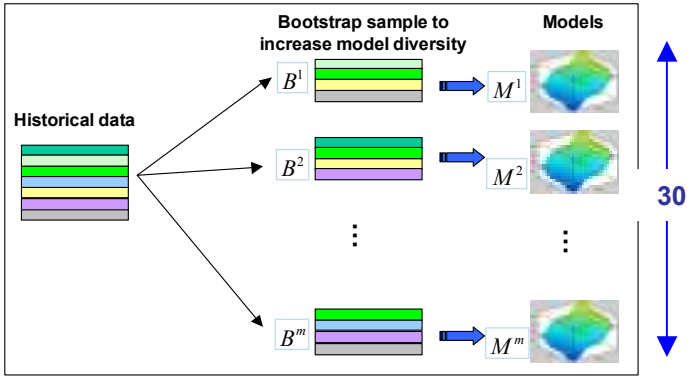


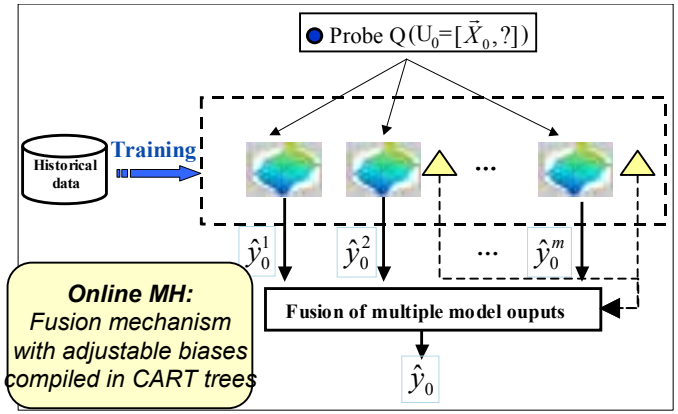


Offline MH: *CART Trees trained on error vectors*



Object Models: *NN's*





Online MH: *Fusion mechanism with adjustable biases compiled in CART trees*

Problem Instance	Problem Type	Model Design (Offline MH's)	Model Controller (*Online MH's*)	Object-level models
Load, HR, NOx Forecast	Prediction	Multiple CART trees (one for each model)	Fusion	Multiple Models: Ensemble of 30 NN's

Reference: "Fast Meta-models for Local Fusion of Multiple Predictive Models" Applied Soft Computing Journal, 2008, doi:10.1016/j.asoc.2008.03.006 – [GE GR Technical Report, 2007GRC832, Oct 12, 2007].

Predictive Model Fusion: Conclusions

Locally Weighted Fusion (off-line): CART-based Precompiled Segmentation

Adaptive fusion based on localized model-specific characteristics

- Takes non-uniform prediction uncertainties into account
- Model prediction quality is input dependent
- Uses CART Segmentation to compute local bias
- 31-48% performance boost observed over baseline fusion techniques

Advantage:

- Can be precomputed: Faster than on-line locally weighted fusion

Future Work

- k-nearest neighbors retrieval: tested but did not improve performance
- Kernel-based functions for peer/neighborhood distance computations
- Winner-take-all with bias compensation

5 Summary and Conclusions

Summary and Conclusions

- **Fusion Benefits:**
  - Overcoming the ceiling of performance of single models
  - Increase accuracy and robustness
  - Reduce uncertainty to make information actionable
- **Fusion Requirements**
  - Diversity in model ensemble(via boosting or multi technologies)
  - Aggregation function (meta model capturing meta-knowledge)
- **Tradeoff Performance versus Complexity**
  - Including models lifecycle in complexity (e.g., cost of ownership)

6 References and Resources

References and Resources

FUSION TUTORIALS

- Robi Polikar (2009), Ensemble learning. Scholarpedia, 4(1):2776
  http://www.scholarpedia.org/article/Ensemble_learning
- Fabio Roli (2009), Mini Tutorial on Multiple Classifier Systems - School on the Analysis of Patterns 2009
  http://www.analysis-of-patterns.net/files/MCS-Part1.pdf
  http://www.analysis-of-patterns.net/files/MCS-Part2.pdf
  http://www.analysis-of-patterns.net/files/MCS-Part3.pdf
- Fabio Roli (2003), Fusion of Multiple Pattern Classifiers – AI*IA 2003
  http://homepages.inf.ed.ac.uk/rbf/IAPR/researchers/MLPAGES/TUTORIALS/ROLI1/Part1.pdf
  http://homepages.inf.ed.ac.uk/rbf/IAPR/researchers/MLPAGES/TUTORIALS/ROLI1/Part2.pdf
  http://homepages.inf.ed.ac.uk/rbf/IAPR/researchers/MLPAGES/TUTORIALS/ROLI1/Part3.pdf
  http://homepages.inf.ed.ac.uk/rbf/IAPR/researchers/MLPAGES/TUTORIALS/ROLI1/Part4.pdf

FUSION INTRODUCTION & BACKGROUND

- Giorgio Fumera, Fabio Roli (2005), A Theoretical and Experimental Analysis of Linear Combiners for Multiple Classifier Systems, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):942-956, June 2005, doi:10.1109/TPAMI.2005.109
- T.G. Dietterich (2000), Ensemble Methods in Machine Learning. J. Kittler and F. Roli (Ed.) *First International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science (pp. 1-15).* New York: Springer Verlag
- K. Tumer and J. Ghosh (1996), Error Correlation and Error Reduction in Ensemble Classifiers, *Connection Science*, 8, (3 & 4): 385 – 404
- L.I. Kuncheva (2005), *Combining Pattern Classifiers: Methods and Algorithms. Hoboken*, N.J.: Wiley,.
- L.I. Kuncheva, C.J. Whitaker (2000), Ten measures of diversity in classifier ensembles: limits for two classifiers, *DERA/IEE Workshop on Intelligent Sensor Processing* (Ref. No. 2001/050),
- A Sharkey, N Sharkey (1997), Combining diverse neural nets, *The Knowledge Engineering Review* 12(3):231-247

RANDOM FOREST

- T. K. Ho (1997), Random subspace method for constructing decision forests, IEEE Transactions on PAMI, 20(8):832-844.
- L. Breiman (2001). Random forests. *Machine Learning* 45(1):5-32
  http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm
- T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning, 2nd Ed.*, **Chapter 15,** Springer, 2008
  http://www-stat.stanford.edu/~hastie/Papers/ESLII.pdf
- Free software:
  http://www.math.usu.edu/~adele/forests

PHM

- P. Bonissone, N. Iyer (2007), Soft Computing Applications to Prognostics and Health Management (PHM): Leveraging field data and domain knowledge, *9th International Work-Conference on Artificial Neural Networks (IWANN 2007)*, pp. 928-939, San Sebastián (Spain), June 20-22, 2007 – [*GE GR Tech. Report*, 2007GRC799, Oct. 2007]

References and Resources (cont.)

ONE-CLASS CLASSIFICATION (Ensemble Learning for Anomaly Detection)

- **Case Study 4.1:** P. Bonissone, X Hu, R. Subbu (2009), A Systematic PHM Approach for Anomaly Resolution: A Hybrid Neural Fuzzy System for Model Construction, Proc. PHM 2009, San Diego, CA, Sept 27-Oct 1, 2009 - [GE GR Tech. Report, GRC839, Sept. 2009]

ONE-CLASS CLASSIFICATION (Other related Ensemble Learning Applications)

- A. Varma, P. Bonissone, W. Yan, N. Eklund, K. Goebel, N. Iyer, S. Bonissone (2007), Anomaly Detection using Non-Parametric information, *Proc. ASME Turbo Expo 2007: Power for Land, Sea and Air*, Montreal, Canada, May 14-17, 2007 – [GE GR Tech. Report, 2007GRC362, Apr. 2007]
- P. Evangelista, M. Embrechts, P. Bonissone, B. Szymanski (2005), Fuzzy ROC Curves for Unsupervised Nonparametric Ensemble Techniques, *IJCNN 2005*, Montreal, Canada - [GE GR Tech. Report, 2005GRC254, Aug. 2005]
- P. Evangelista, P. Bonissone, M. Embrechts, B. K. Szymanski (2005), Unsupervised Fuzzy Ensembles Applied to Intrusion Detection, *Proc. 13th European Symposium on Artificial Neural Networks 2005*, pp. 345-350, Bruges, Belgium, April 27-29, 2005.

MULTI-CLASS CLASSIFICATION (Ensemble Learning for Diagnostics)

- **Case Study 4.2:** W. Yan, F. Xue, Jet Engine Gas Path Fault Diagnosis Using Dynamic Fusion of Multiple Classifiers, *IJCNN 2008*, Hong Kong, pp-1585-1591 - [GE GR Tech. Report 2008GRC395, May 2008]

MULTI-CLASS CLASSIFICATION (Other related Ensemble Learning Applications)

- P. Bonissone, J. M. Cadenas, M.C. Garrido, R.A. Diaz (2010), A Fuzzy Random Forest, *The International Journal of Approximate Reasoning*, to appear, 2010, *doi:10.1016/j.ijar.2010.02.003*
- P. Bonissone, N. Eklund, K. Goebel (2005), Using an Ensemble of Classifiers to Audit a Production Classifier, *6th Int.'l Workshop on Multiple Classifier Systems* (*MCS 2005*), pp. 376-386, Monterey, CA, June 13 -1, 2005
- P. Bonissone, K. Goebel, and W. Yan (2004), Classifier Fusion using Triangular Norms, *Multiple Classifier Systems (MCS) 2004*, pp. 154-163, Cagliari, Italy, June 2004 - [GE GR Tech. Report, 2006GRC143, Feb 21, 2006]
- K. Woods, WP. Kegelmeyer, and K. Bowyer (1997), Combination of multiple classifiers using local accuracy estimates, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):405-410

PREDICTION (Ensemble Learning for Prediction)

- **Case Study 4.3:** P. Bonissone, F. Xue, and R. Subbu (2008), Fast Meta-models for Local Fusion of Multiple Predictive Models, *Applied Soft Computing Journal,* 2008, doi:10.1016/j.asoc.2008.03.006 - [GE GR Tech. Report 2007GRC832, Oct 2007]
- F. Xue, R. Subbu, P. Bonissone (2006), Locally Weighted Fusion of Multiple Predictive Models, *IEEE International Joint Conference on Neural Networks (IJCNN'06),* pp. 2137-2143, Vancouver, BC, Canada, July 16 –21, 2006 – [GE GR Technical Report, 2006GRC454, Nov. 2006]
- F. Xue, P. Bonissone, A. Varma, W. Yan, N. Eklund, K. Goebel (2008), An Instance-Based Method for Remaining Useful Life Estimation for Aircraft Engines, *Journal of Failure Analysis and Prevention*, (8)-2:199-206, April 2008, http://www.springerlink.com/content/4t7656400t571727/) – [GE GR Tech. Report 2007GRC276, Apr. 2007].
- P. Bonissone, A. Varma, K. Aggour, and F. Xue (2006), Design of local fuzzy models using evolutionary algorithms, *Computational Statistics and Data Analysis*, 51:398-416, 2007 – [GE GR Tech. Report 2006GRC594, Oct. 2007]