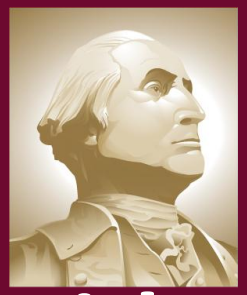


THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Application of a Multi-algorithm Decision Scheme for Improving the Robustness of Network Intrusion Detection

John M. Fossaceca

16th Annual

Systems Engineering Conference

October 28-31, 2013

NDIA
National Defense Industrial Association

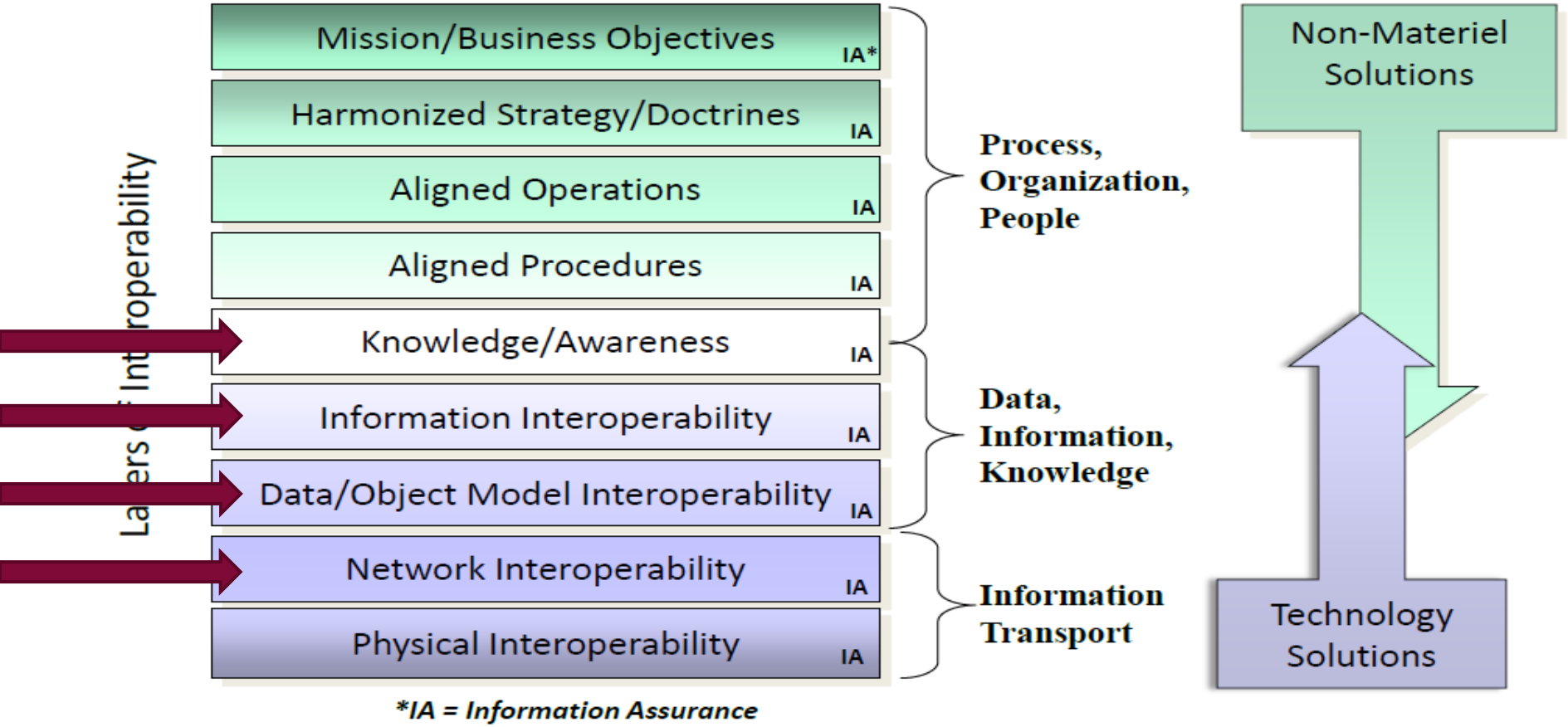
Co-Authors:

Dr. Thomas A. Mazzuchi, Professor and Department Chair

Dr. Shahram Sarkani, Professor and Director of EMSE Off-Campus Programs

What I'll Cover

- How this talk fits into Net Centric Operations/Interoperability
- Review of Network Intrusion Detection Systems Challenges
- Inspiration for my research
- Overview of Ensemble Decision Making
- Some Other Tools
  - Boosting
  - Support Vectors
  - Kernel Methods & Multiple Kernel Methods
  - Extreme Learning Machines
- My Approach to the Problem
- Preliminary Findings & Future Work
- Discussion & Questions



"Interoperability must be synergized with Information Assurance to assure obtaining the best of both" – Jack Zavin, Beyond Technical Interoperability, NDIA SE Conference, October 2012

Some Challenges in Network Intrusion Detection

- High False Alarm Rates
- High Dimensionality
- Data Imbalance
- Online Processing
- Zero Day & New Attacks (thwart Signature Based Systems)

For Anomaly Based "Learning" NIDS above are True Plus:

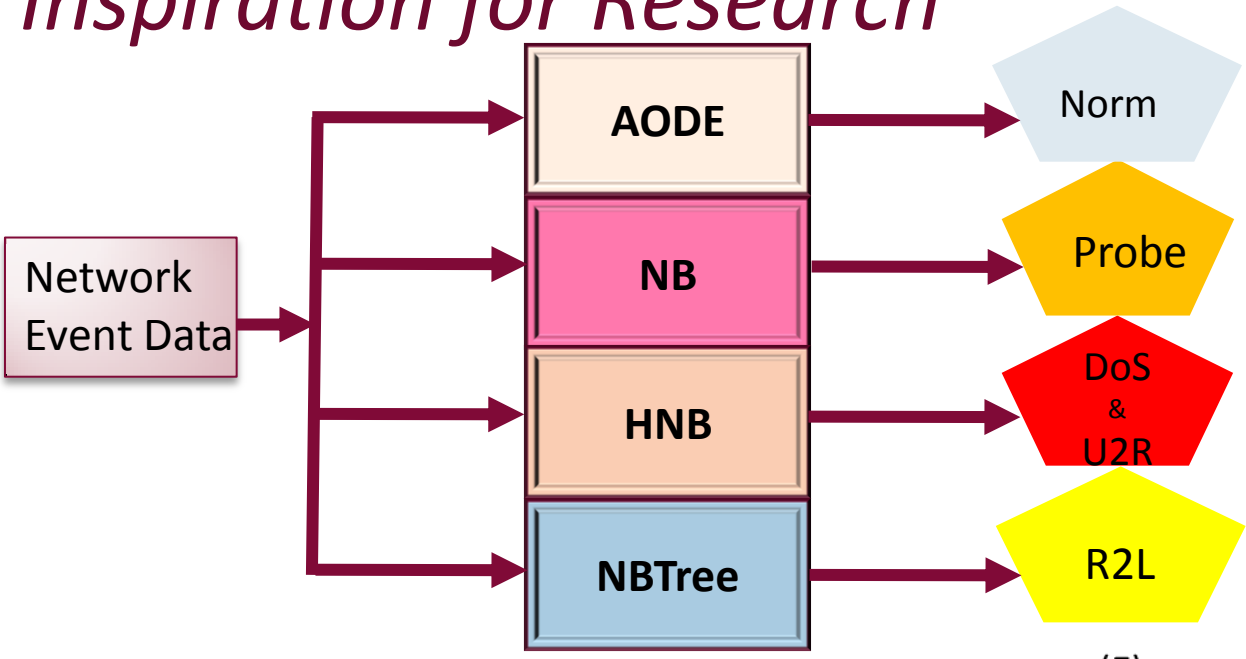
- Danger of Over-Fitting Data
- Opportunity for "Incremental Leaning" for "Concept Shifts"
- Opportunity to incorporate human expertise
- Opportunity to incorporate multiple algorithms

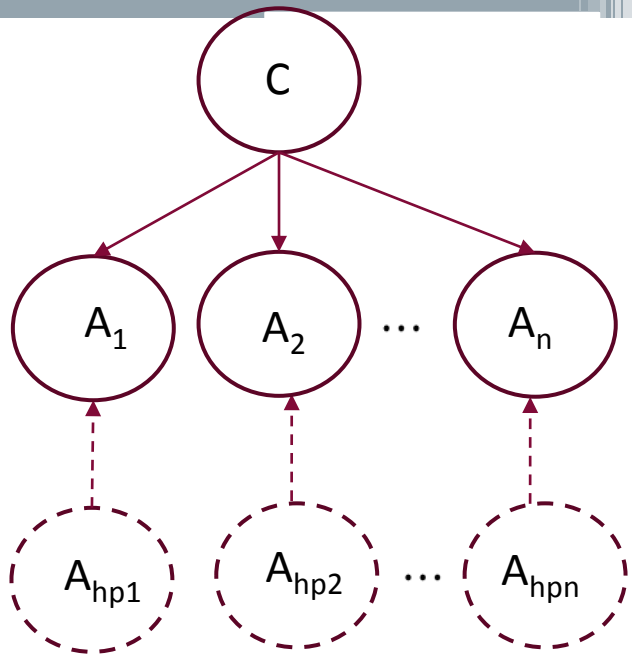
Net Centric View: Practical Considerations relative to Network Intrusion Detection

- Data from Disparate Sources
- Multiple Perspectives/Multiple Formats
- Real-time Data Fusion & Decision Optimization
- Dealing with Uncertainty
- Agility & Resilience
- Online Approach vs. Batch Processing
- How do we do all of this quickly?

Adapted from and added to "Beyond Technical Interoperability" Context for the Net Centric Operations & Interoperability Track @ 2011 NDIA SE Conference 26 – 27 October 2012, Jack Zavin NCO/I Chair

Inspiration for Research





$$c(E) = \arg\max_{c \in C} P(c) \prod_{1=1}^{n} P(a_i | a_{hpi}, c)$$

$$P(a_i | a_{hpi}, c) = P(c) \sum_{j=1.j\neq i}^{n} w_{ij} * P(a_i | a_j, c)$$

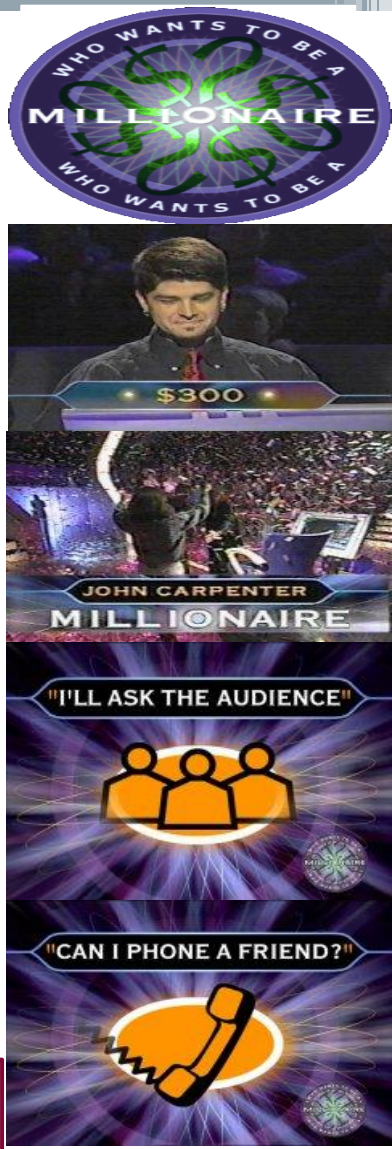
There is a "hidden parent" A_{hpi} for *each* attribute A_i accounting for the "weighted influences" from the rest of the attributes (dashed circles).

QUESTION: Is there a way to combine the use of Multiple Algorithms?

Koc, Levent, Application of a Hidden Naïve Bayes Multiclass Classifier in Network Intrusion Detection, Ph.D. Dissertation, George Washington University

Decision Making with Ensembles

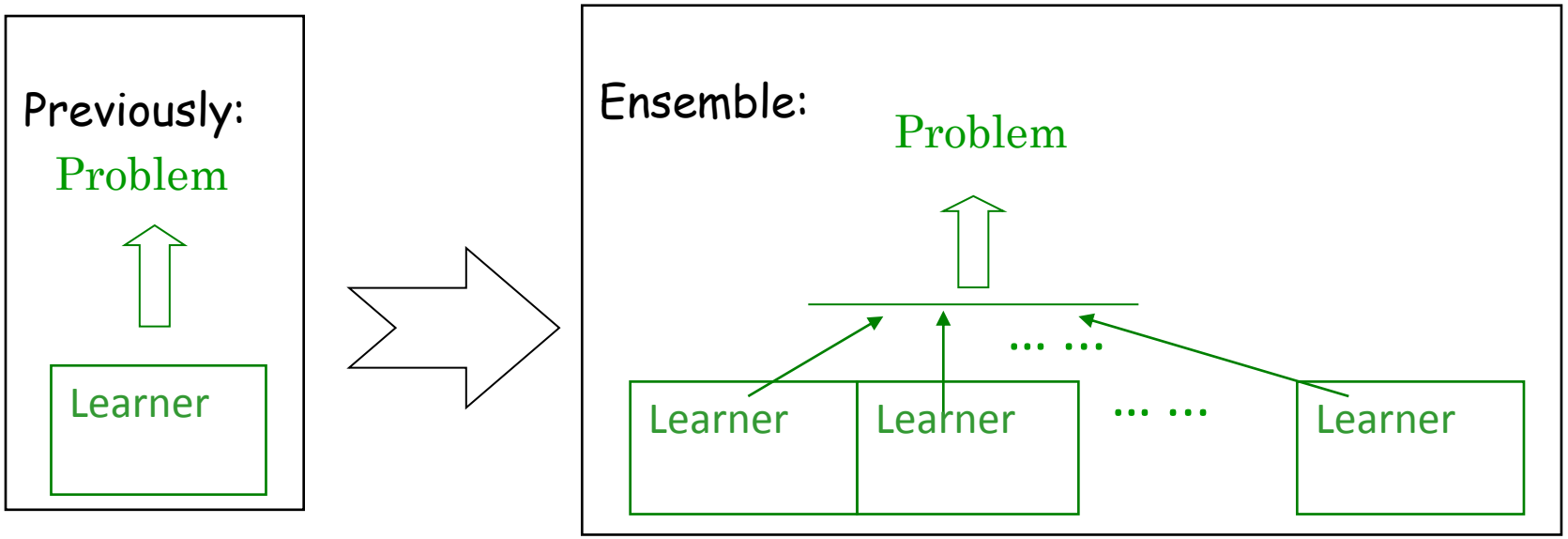
- **Intuitive Basis** for why Ensembles should provide a better decision than other approaches
- Good performance on training data does not predict good **generalization** performance (i.e. performance of the classifier on data not seen during training)
- **Too much data:** training different classifiers with different partitions of data, and combining their outputs using an intelligent combination rule is often a more efficient approach
- **Too little data:** resampling techniques resampling techniques can be used for drawing overlapping random subsets of the available data, each of which can be used to train a different classifier, creating the ensemble
- **Data Fusion:** data from different sources are combined to make a more informed decision



Polikar, Robi. "Ensemble based systems in decision making." Circuits and Systems Magazine, IEEE 6.3 (2006): 21-45 & pictures from: http://game-shows.chris-place.com/shows/millionaire/hot-seat.htm

Ensemble Learning:

A machine learning paradigm where multiple learners are used to solve the problem



- The generalization ability of the ensemble is usually significantly better than that of an individual learner
- Boosting is one of the most important families of ensemble methods

Rachlin, Glen. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting" – Rachlin's updated version presentation by Zhi-Hua Zhou, ICDM 2006 based on Freund & Schapire's original paper)

Boosting

- In 1990 it was proven that *a weak learner*, an algorithm that generates classifiers that can merely do better than random guessing, can be turned into a *strong learner* that generates a classifier that can correctly classify all but an arbitrarily small fraction of the instances

- Creates an ensemble of classifiers by resampling the data, which are then combined by majority voting.

- **Basic Algorithm:**
  - Classifier *C*1 is trained with a random subset of the available training data.
  - The training data subset for the second classifier *C*2 is chosen as the most informative subset, given *C*1. That is, *C*2 is trained on a training data only half of which is correctly classified by *C*1, and the other half is misclassified.
  - The third classifier *C*3 is trained with instances on which *C*1 and *C*2 disagree.
  - The three classifiers are combined through a three-way majority vote.

- **AdaBoost:** is a more general version of the original boosting algorithm (1997 – AT&T Labs Research)

Polikar, Robi. "Ensemble based systems in decision making." Circuits and Systems Magazine, IEEE 6.3 (2006): 21-45

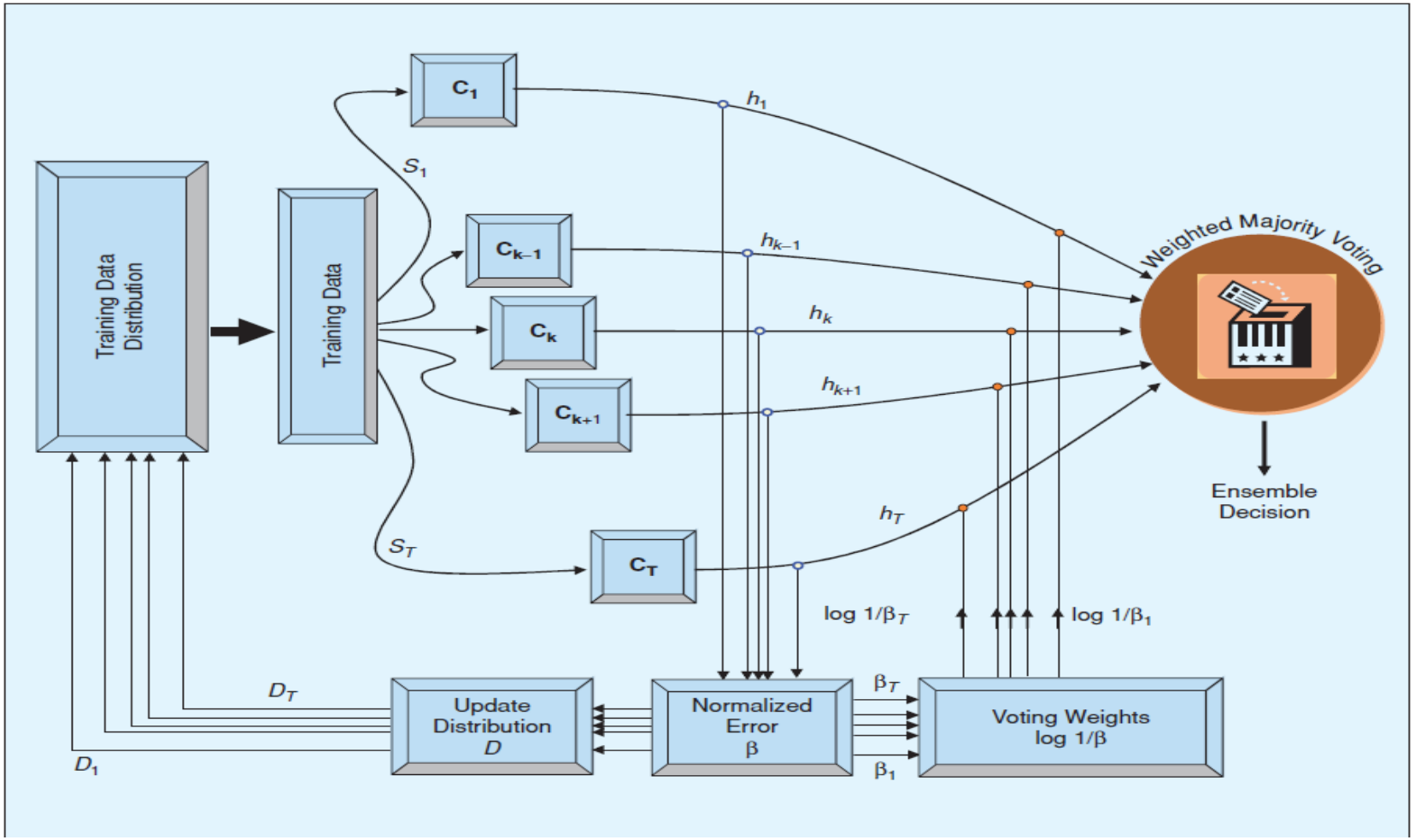
High Level Description of Adaboost

Weighted combinations of classifiers

- "Committee" decisions
  - Trivial example
  - Equal weights (majority vote)
  - Might want to weight unevenly – up-weight good experts
- Boosting
  - Focus new experts on examples that others get wrong
  - Train experts sequentially
  - Errors of early experts indicate the "hard" examples
  - Focus later classifiers on getting these examples right
  - Combine the whole set in the end
  - Convert many "weak" learners into a complex classifier

Professor Alexander Ihler, Lecture at UC Irvine:*http://sli.ics.uci.edu/Classes/2012F-273a?action=download&upname=10-ensembles.pdf*

Adaboost – Block Diagram



Polikar, Robi. "Ensemble based systems in decision making." Circuits and Systems Magazine, IEEE 6.3 (2006): 21-45

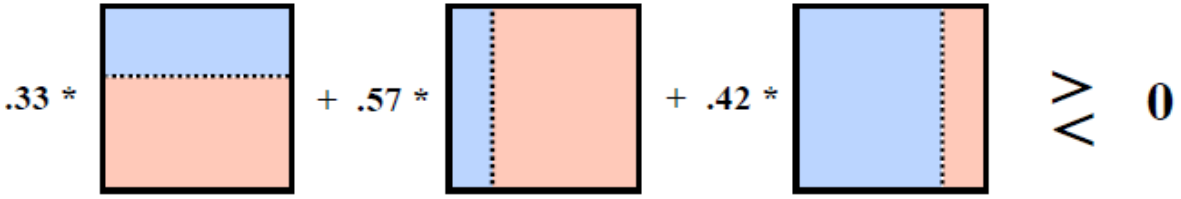
The AdaBoost Algorithm:

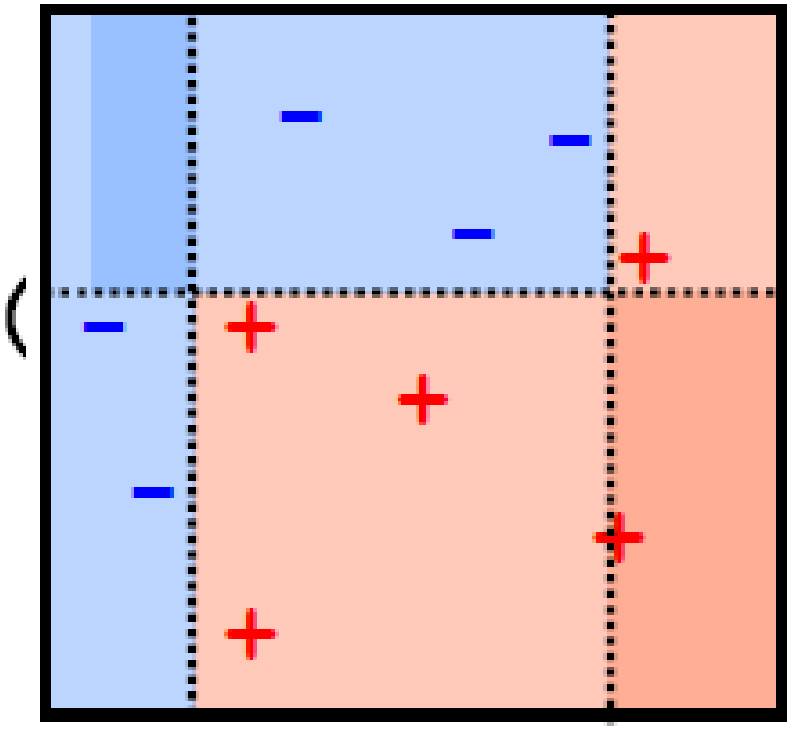
TrainAdaBoost(D, BaseLearn)
For each example d_i in D let its weight $w_i=1/|D|$
Let H be an empty set of hypotheses
For t from 1 to T do:
Learn a hypothesis, h_t, from the weighted examples: h_t=BaseLearn(D)
Add h_t to H
Calculate the error, ε_t, of the hypothesis h_t as the total sum weight of the examples that it classifies incorrectly.
If $\varepsilon_t > 0.5$ then exit loop, else continue.
Let $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$
Multiply the weights of the examples that h_t classifies correctly by β_t
Rescale the weights of all of the examples so the total sum weight remains 1.
Return H

TestAdaBoost(ex, H)
Let each hypothesis, h_t, in H vote for ex's classification with weight $\log(1/\beta_t)$
Return the class with the highest weighted vote total.

Professor Ray Mooney – Course Slides for: http://www.cs.utexas.edu/~mooney/cs391L/slides/ensembles.pdf

Weight each classifier and combine them:

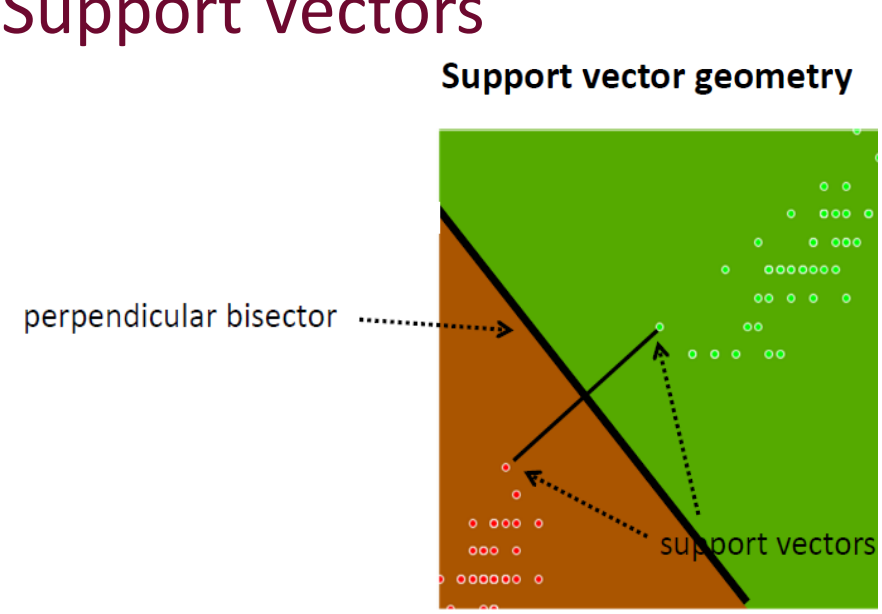


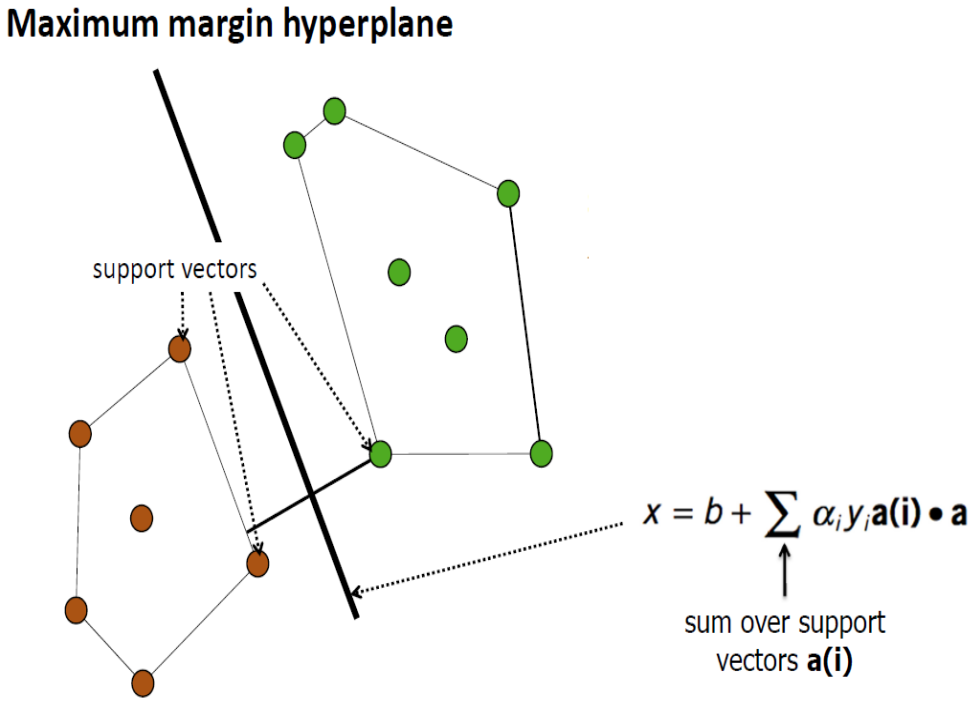


Combined classifier

Professor Alexander Ihler, Lecture at UC Irvine: http://www.youtube.com/watch?v=ix6IvwbVpw0 & http://sli.ics.uci.edu/Classes/2012F-273a?action=download&upname=10-ensembles.pdf

Support Vectors





Strengths of SVM:

- Training is relatively easy.
- No issues with local optima, unlike in neural networks.
- Scales relatively well to high dimensional data

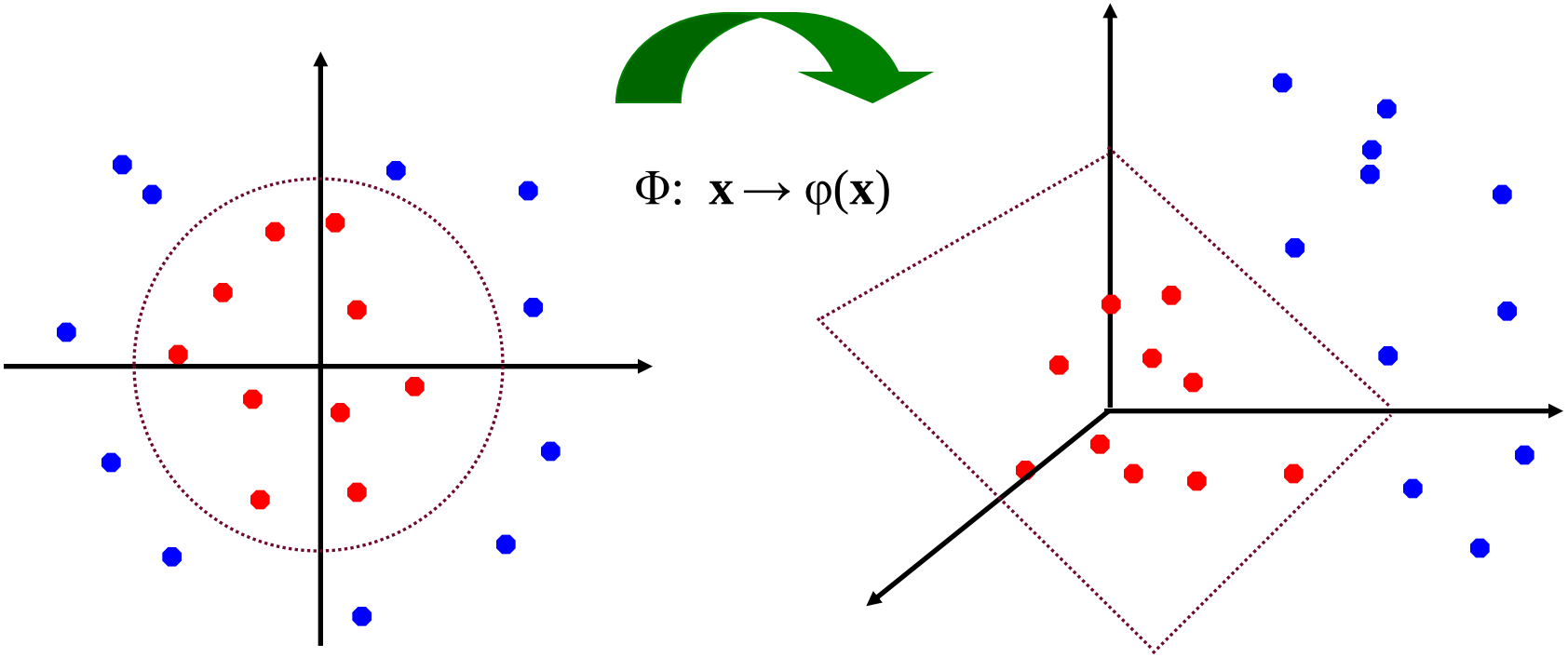
Weaknesses:

- Requires selection of good kernel function
- CPU Intensive

Tutorial on Support Vector Machine (SVM) Vikramaditya Jakkula, School of EECS, Washington State University, Pullman 99164 - *eecs.wsu.edu/~vjakkula/**SVM**Tutorial.doc & Ian Witten: https://weka.waikato.ac.nz/dataminingwithweka/unit?unit=4&lesson=5*

Kernels

General idea: the original input space can be mapped to some higher-dimensional feature space where the training set is separable:



Professor Jinwei Gu: *www1.cs.columbia.edu/~belhumeur/courses/biometrics/2010/svm.**ppt***

Non-Linear SVMs: Kernel Trick

- With this mapping, our discriminant function is now:

$$g(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b = \sum_{i \in \mathrm{SV}} \alpha_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + b$$

- No need to know this mapping explicitly, because we only use the dot product of feature vectors in both the training and test.

- A *kernel function* is defined as a function that corresponds to a dot product of two feature vectors in some expanded feature space:

$$K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

Professor Jinwei Gu: *www1.cs.columbia.edu/~belhumeur/courses/biometrics/2010/svm.**ppt***

Online Multiple Kernel Classification

- Online learning is designed to sequentially learn a prediction model based on iterative feedback and from prior runs
- Kernel learning aims to learn an effective kernel function for a given learning task from training data
- Kernel Learning (MKL) finds the optimal combination of multiple kernels to optimize learning performance
- **Online Multiple Kernel Classification** (OMKC) aims to learn multiple kernel classifiers and their linear combination simultaneously
- Due to online operations, both the optimal kernel classifiers and their linear combinations need to be learned **simultaneously**
- Solutions to kernel classifiers and their linear combinations are strongly correlated, making this a significantly more challenging problem than a typical online learning problem

Hoi, Steven CH, Rong Jin, Peilin Zhao, and Tianbao Yang. "Online multiple kernel classification." *Machine Learning* 90, no. 2 (2013): 289-316

Online Multiple Kernel Classification - Approach

Deterministic Algorithm for OMKC ($\mathbf{OMKC}_{(D,D)}$)

1: INPUT:
   - Kernels: $k_i(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, i = 1, \ldots, m$
   - Weights $w_i(1) = 1, i = 1, \ldots, m$
   - Discount weight $\beta \in (0, 1)$.

2: **Initialization**: $\mathbf{f}^1 = 0, \mathbf{w}^1 = 1, \theta^1 = \frac{1}{m}1$

3: **for** $t = 1, 2, \ldots$ **do**

4: Receive an instance: x_t

5: Predict $\hat{y}_t = \text{sign}\Big(\sum_{i=1}^m \theta_i \text{sign}\big(f_i^t(x_t)\big)\Big)$

6: Receive the class label: y_t

7: **for** $i = 1, 2, \ldots, m$ **do**

8: Set $z_i^t = I(y_t f_i^t(x_t) \leq 0)$

9: Update $w_i^{t+1} = w_i(t)\beta^{z_i^t}$

10: Update $f_i^{t+1}(x) = f_i^t(x) + z_i^t y_t \kappa_i(x_t, x)$

11: **end for**

12: $\theta_i^{t+1} = \frac{w_i^t}{W_t}, i = 1, \ldots, m$, where $W_t = \sum_{i=1}^m w_i^t$

13: **end for**

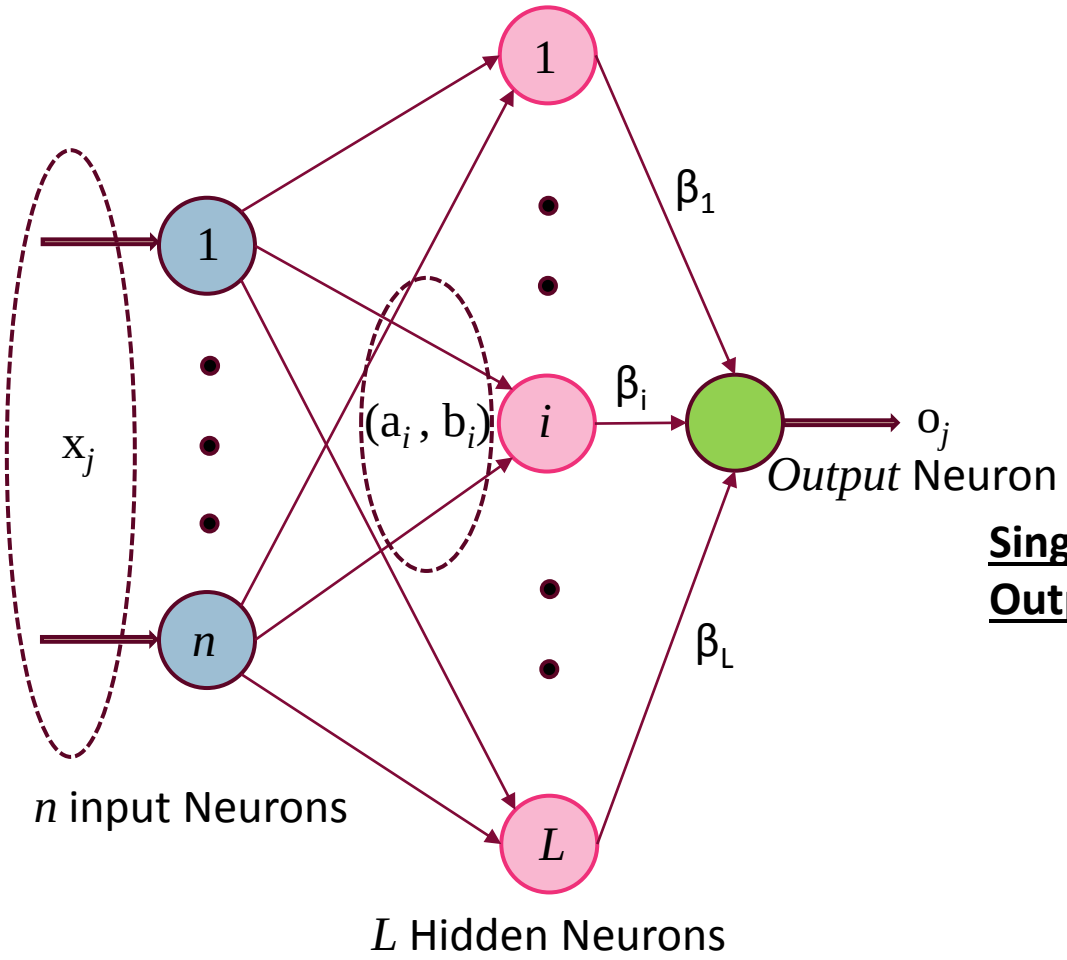
Hoi, Steven CH, Rong Jin, Peilin Zhao, and Tianbao Yang. "Online multiple kernel classification." *Machine Learning* 90, no. 2 (2013): 289-316

Performance Results - OMKC

Algorithm	Perceptron	Perceptron (u)	Perceptron (*)	OM-2	OMKC$_{(D,D)}$
Ionosphere	n= 351 d= 34	m= 16	best kernel expert: gaussian kernel of $\sigma = 1$		
Mistake (%)	26.82 ± 1.63	18.73 ± 1.23	22.07 ± 6.77	17.41 ± 1.20	**16.07 ± 1.42**
SV (#)	94.2 ± 5.7	65.8 ± 4.3	77.5 ± 23.7	128.5 ± 4.4	1347.3 ± 53.0
Time (s)	0.004 ± 0.000	0.003 ± 0.000	0.042 ± 0.002	0.113 ± 0.001	0.384 ± 0.024
votes84	n= 435 d= 16	m= 16	best kernel expert: polynomial kernel of $p = 1$		
Mistake (%)	8.17 ± 0.73	8.68 ± 0.62	9.45 ± 1.94	**7.21 ± 0.68**	**7.38 ± 0.69**
SV (#)	35.5 ± 3.2	37.8 ± 2.7	41.1 ± 8.4	36.5 ± 2.3	939.6 ± 40.2
Time (s)	0.004 ± 0.000	0.004 ± 0.000	0.045 ± 0.002	0.124 ± 0.001	0.406 ± 0.017
wdbc	n= 569 d= 30	m= 16	best kernel expert: gaussian kernel of $\sigma = 64$		
Mistake (%)	34.51 ± 1.82	41.52 ± 3.70	12.29 ± 1.01	41.70 ± 4.06	**11.70 ± 1.01**
SV (#)	196.3 ± 10.3	236.3 ± 21.0	70.0 ± 5.7	237.3 ± 23.1	3032.2 ± 37.3
Time (s)	0.007 ± 0.000	0.008 ± 0.000	0.065 ± 0.001	0.214 ± 0.007	0.597 ± 0.013
breast	n= 683 d= 9	m= 16	best kernel expert: gaussian kernel of $\sigma = 8$		
Mistake (%)	26.73 ± 1.19	41.90 ± 3.40	6.12 ± 0.79	44.33 ± 3.88	**4.86 ± 0.51**
SV (#)	182.6 ± 8.1	286.1 ± 23.2	41.8 ± 5.4	303.4 ± 26.4	1606.6 ± 49.9
Time (s)	0.008 ± 0.000	0.009 ± 0.001	0.068 ± 0.001	0.276 ± 0.015	0.641 ± 0.017
australian	n= 690 d= 14	m= 16	best kernel expert: gaussian kernel of $\sigma = 2$		
Mistake (%)	39.54 ± 1.51	39.50 ± 2.70	**38.04 ± 2.38**	39.62 ± 2.88	**37.67 ± 1.20**
SV (#)	272.9 ± 10.4	272.6 ± 18.6	262.4 ± 16.4	273.4 ± 19.9	4743.8 ± 70.0
Time (s)	0.010 ± 0.001	0.010 ± 0.001	0.091 ± 0.003	0.266 ± 0.006	0.779 ± 0.017
diabetes	n= 768 d= 8	m= 16	best kernel expert: gaussian kernel of $\sigma = 32$		
Mistake (%)	44.14 ± 1.86	45.18 ± 2.19	35.55 ± 2.07	45.35 ± 2.18	**33.69 ± 1.29**
SV (#)	339.0 ± 14.3	347.0 ± 16.8	273.0 ± 15.9	348.3 ± 16.7	4614.6 ± 63.8
Time (s)	0.012 ± 0.001	0.012 ± 0.001	0.099 ± 0.006	0.321 ± 0.014	0.886 ± 0.021
fourclass	n= 862 d= 2	m= 16	best kernel expert: gaussian kernel of $\sigma = 8$		
Mistake (%)	36.29 ± 1.09	35.82 ± 1.56	3.78 ± 0.76	35.92 ± 1.65	**3.19 ± 0.38**
SV (#)	312.8 ± 9.4	308.8 ± 13.4	32.6 ± 6.6	309.6 ± 14.2	3151.0 ± 33.5
Time (s)	0.013 ± 0.001	0.012 ± 0.001	0.092 ± 0.001	0.348 ± 0.005	0.862 ± 0.010

Hoi, Steven CH, Rong Jin, Peilin Zhao, and Tianbao Yang. "Online multiple kernel classification." *Machine Learning* 90, no. 2 (2013): 289-316

Brief Introduction to Extreme Learning Machines



Hidden Node Output

$$G(a_i, b_i, \boldsymbol{x}) = g(a_i \cdot \boldsymbol{x} + b_i)$$

a_i : the weight vector connecting the ith hidden node and the input nodes.

b_i : the threshold of the ith hidden node.

Single Layer Feed Forward Network (SLFN) Output

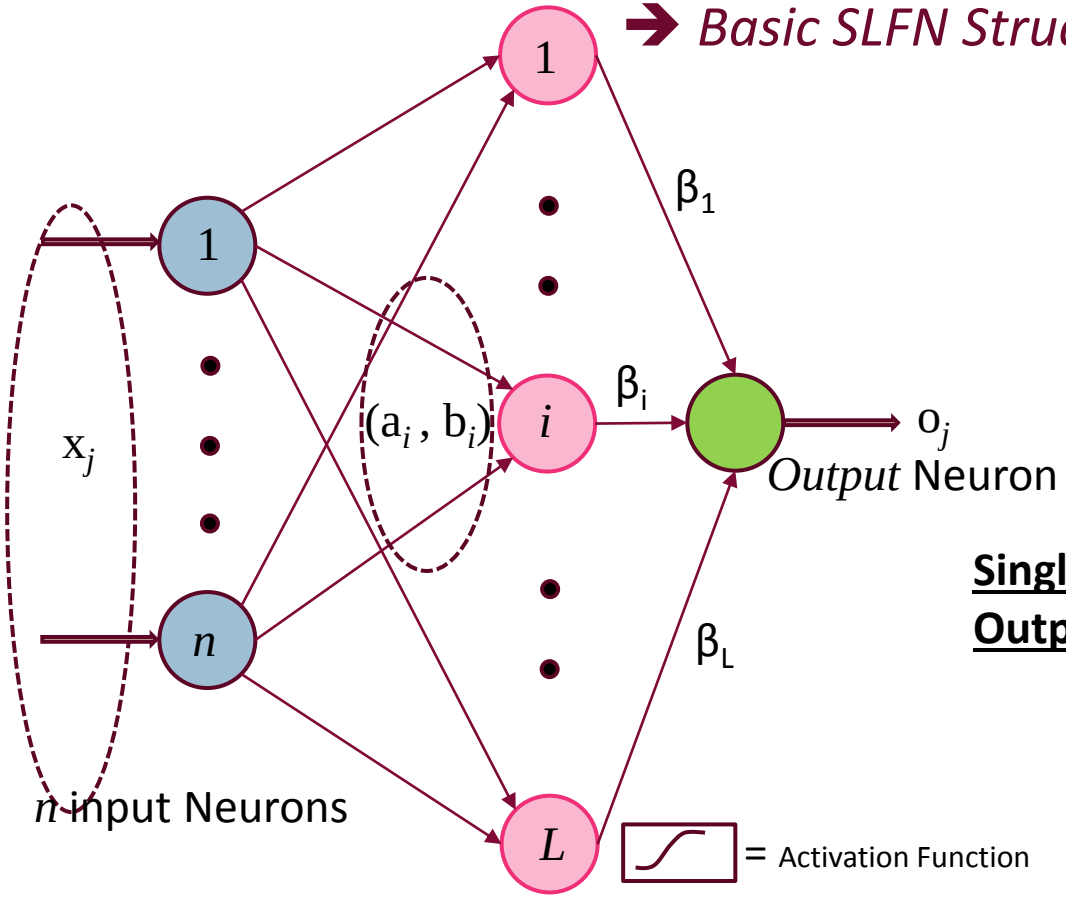
$$f_L(x) = \sum_{1=1}^{L} \beta_i G(a_i, b_i, x)$$

β_i : the weight vector connecting the Ith hidden node and the output nodes.

Huang, Guang-Bin, Hongming Zhou, Xiaojian Ding, and Rui Zhang. "Extreme learning machine for regression and multiclass classification." *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 42, no. 2 (2012): 513-529

Brief Introduction to Extreme Learning Machines:

➔ *Basic SLFN Structure*



Hidden Node Output

$$G(a_i, b_i, \boldsymbol{x}) = g(a_i \cdot \boldsymbol{x} + b_i)$$

a_i : the weight vector connecting the ith hidden node and the input nodes.

b_i : the threshold of the ith hidden node.

Single Layer Feed Forward Network (SLFN) Output

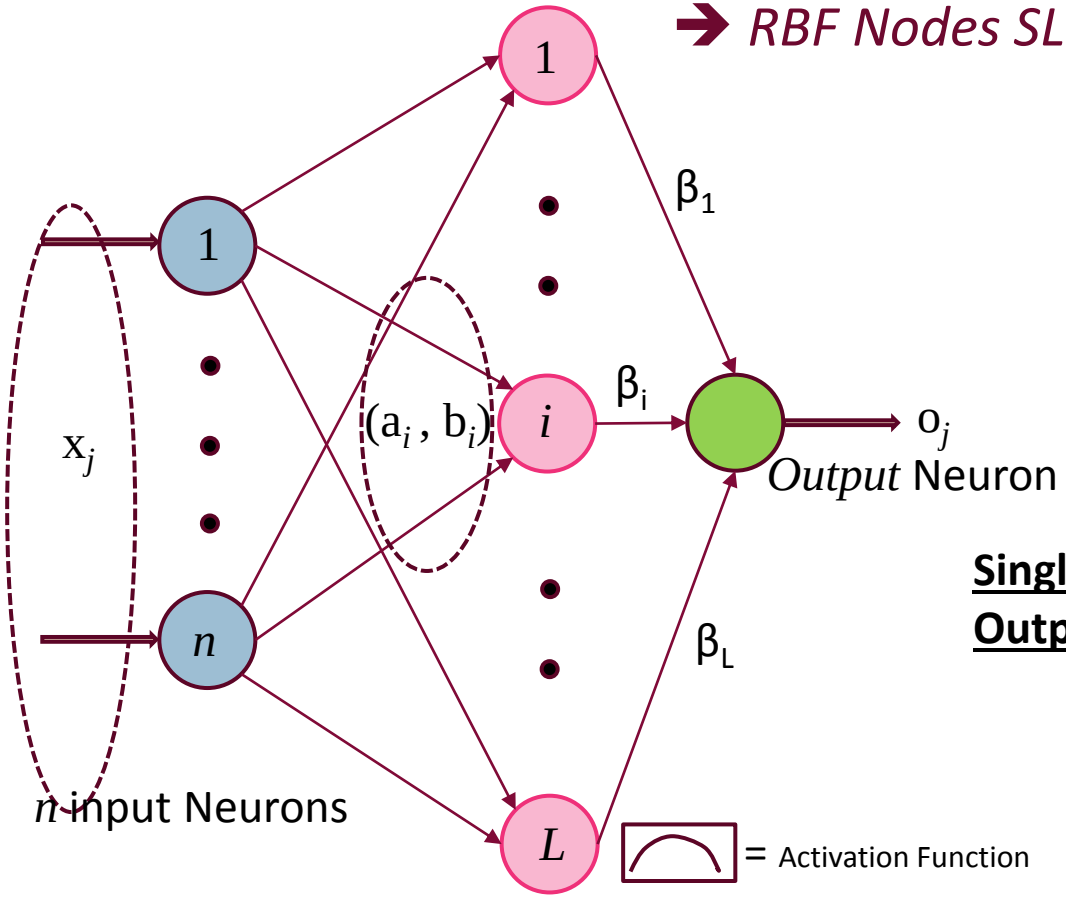
$$f_L(x) = \sum_{1=1}^{L} \beta_i G(a_i, b_i, x)$$

β_i : the weight vector connecting the ith hidden node and the output nodes.

Huang, Guang-Bin, Hongming Zhou, Xiaojian Ding, and Rui Zhang. "Extreme learning machine for regression and multiclass classification." *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 42, no. 2 (2012): 513-529

Brief Introduction to Extreme Learning Machines:

➔ *RBF Nodes SLFN*



Hidden Node Output

$$G(a_i, b_i, \boldsymbol{x}) = g(b_i \|\boldsymbol{x} - a_i\|)$$

a_i : the center of the ith hidden node.

b_i : the impact factor of the ith hidden node.

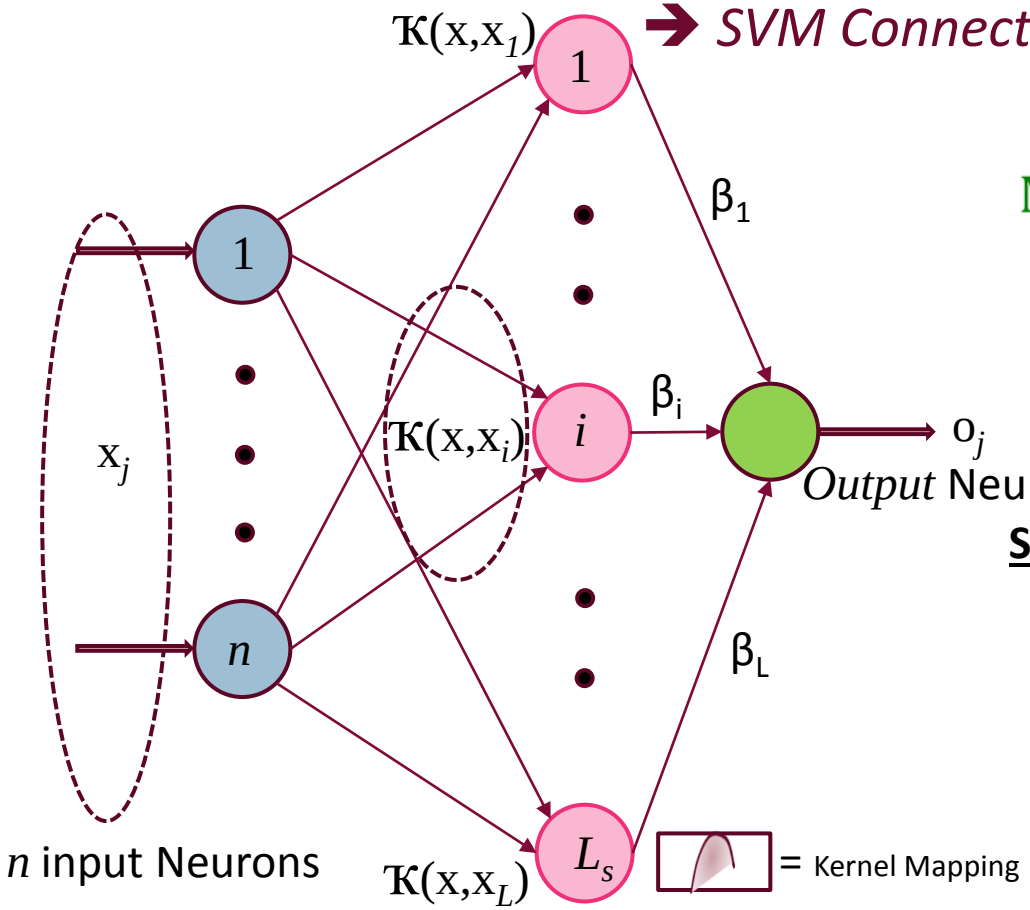
Single Layer Feed Forward Network (SLFN) Output

$$f_L(x) = \sum_{1=1}^{L} \beta_i G(a_i, b_i, x)$$

β_i : the weight vector connecting the ith hidden node and the output nodes.

Huang, Guang-Bin, Hongming Zhou, Xiaojian Ding, and Rui Zhang. "Extreme learning machine for regression and multiclass classification." *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 42, no. 2 (2012): 513-529

Brief Introduction to Extreme Learning Machines:



➔ *SVM Connection to SLFN*

SVM Optimization Formula

$$\text{Minimize: } Q_p = \frac{1}{2}\|W\|^2 + C\sum_{1=1}^{L}\epsilon_i$$

$$\text{s.t.: } t_i(w \cdot \emptyset(x_i) + b) \geq 1 - \epsilon_i, \forall i$$

$$\epsilon_i \geq 0, \forall i$$

Support Vector Machine Decision Function:

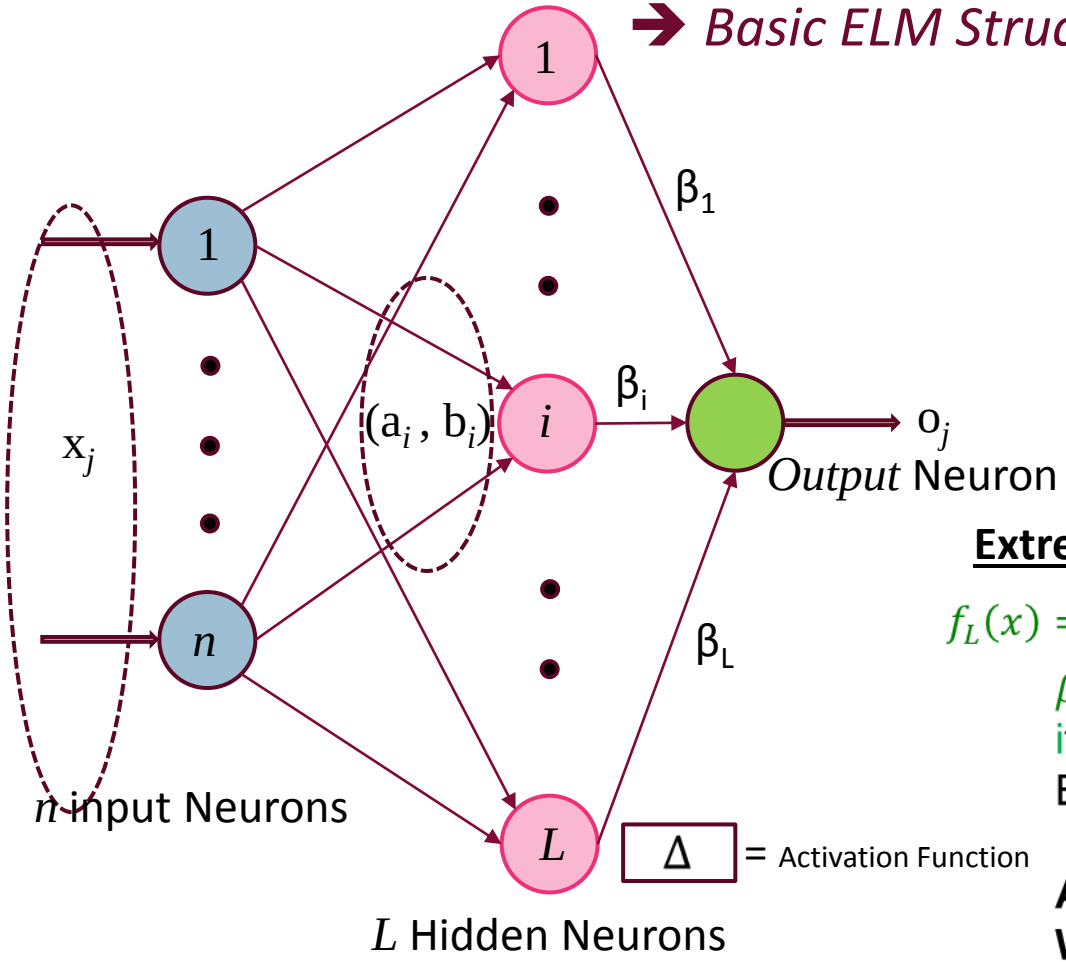
$$f_L(x) = sign\left(\sum_{1=1}^{L_s} \alpha_s t_s K(x, x_i) + b_i\right)$$

β_i : the weight vector connecting the ith hidden node and the output nodes.

Huang, Guang-Bin, Xiaojian Ding, and Hongming Zhou. "Optimization method based extreme learning machine for classification." *Neurocomputing* 74, no. 1 (2010): 155-163

Brief Introduction to Extreme Learning Machines:

➔ *Basic ELM Structure*



Hidden Node Output

$$G(a_i, b_i, \boldsymbol{x}) = g(a_i \cdot \boldsymbol{x} + b_i)$$

a_i : the weight vector connecting the ith hidden node and the input nodes.

b_i : the threshold of the ith hidden node.

Extreme Learning Machine (ELM) Output

$$f_L(x) = \sum_{1=1}^{L} \beta_i G(a_i, b_i, x) = t_j \; for \; j = 1, .., N$$

β_i : the weight vector connecting the ith hidden node and the output nodes.

Equivalent to:

$$\mathbf{H}\beta = \mathbf{T}$$

A Matrix Equation that can be solved With a simple Pseudo Inverse!

Huang, Guang-Bin, Hongming Zhou, Xiaojian Ding, and Rui Zhang. "Extreme learning machine for regression and multiclass classification." *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 42, no. 2 (2012): 513-529

ELM Operation

Given a training set $\aleph = \{(x_i, t_i) | x_i \in R^n, t_i \in R^m, i = 1, \ldots, N\}$ with hidden node output function G(**a**, b, **x**), and the number of hidden nodes L:

1 - Assign randomly hidden node parameters (a_i , b_i), i = 1, · · · , L.

2 - Calculate the hidden layer output matrix **H**.

3 - Calculate the output weight : β: β = $\mathbf{H}^{\dagger}\mathbf{T}$.

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of hidden layer output matrix **H**.

NOTE:

- The hidden node parameters a_i and b_i are independent of the training data and each other.
- Unlike conventional learning methods which MUST see the training data before generating the hidden node parameters, ELM generates the hidden node parameters prior to seeing the training data.
- Unlike traditional gradient-based learning algorithms which only work for differentiable activation functions, ELM works for all bounded non-constant piecewise continuous activation functions.

Huang, Guang-Bin, Qin-Yu Zhu, and Chee-Kheong Siew. "Extreme learning machine: a new learning scheme of feedforward neural networks." In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 2, pp. 985-990. IEEE, 2004.

Advantages of Extreme Learning Machine Approach versus SLFN & SVM

- Need to choose different algorithms & architectures in SLFN (i.e. number of hidden layers)
- Manual Parameter Tuning is Required
- Danger of Over-fitting the Data
- Possibility of Local minima
- CPU Intensive/Time consuming

The ELM learning algorithm is simpler than many other popular learning algorithms: neural networks and support vector machines.

Huang, Guang-Bin, Qin-Yu Zhu, and Chee-Kheong Siew. "Extreme learning machine: a new learning scheme of feedforward neural networks." In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 2, pp. 985-990. IEEE, 2004.

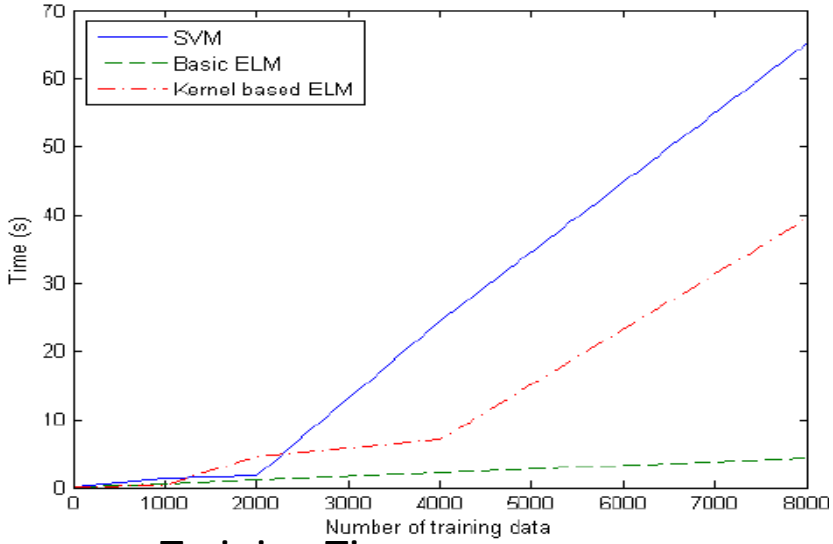
Application of ELM to Network Intrusion Detection – Cheng et. al.

- SVM and ELM are compared (SVM is the Benchmark).
- A Data processing script is used to convert the raw TCP/IP dump data into machine readable form.
- Training Phase: SVM and ELM are trained on normal data and different types of attacks.
- For the binary classification case, the data has 41 features and falls into 2 classes: **normal and attack**; for the multi-class.
- For the multi-classification case, the data has 41 features and falls into 23 classes: normal and **22 types of attacks**. 4 main categories (DoS, U2R, R2L, Probe)
- Testing phase: SVM and ELM are used to predict the type of each data point in the testing dataset, and their individual performances were compared.

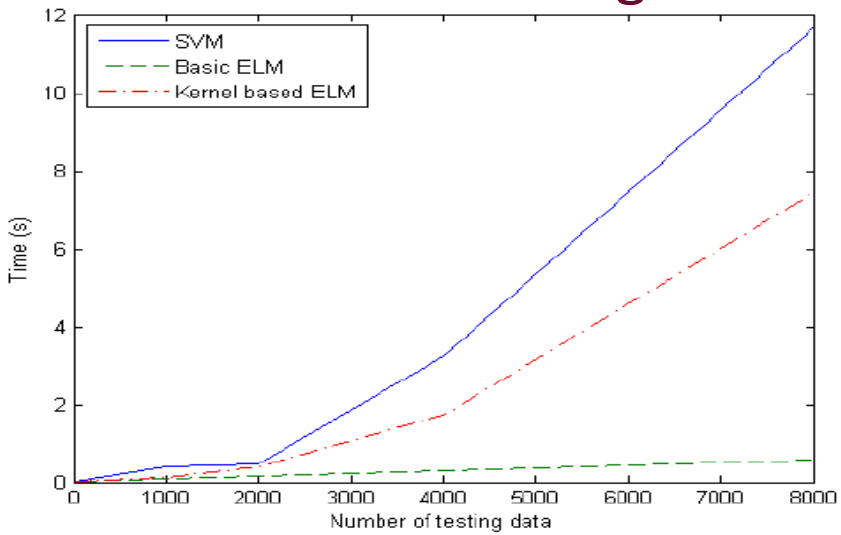
NOTE: Both SVM and ELM **cannot process symbolic data**, so a method was employed to convert symbolic data into continuous data.

Cheng, Chi, Wee Peng Tay, and Guang-Bin Huang. "Extreme learning machines for intrusion detection." In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pp. 1-8. IEEE, 2012.

Application of ELM to Network Intrusion Detection – Cheng et. al.



Training Time



Testing Time

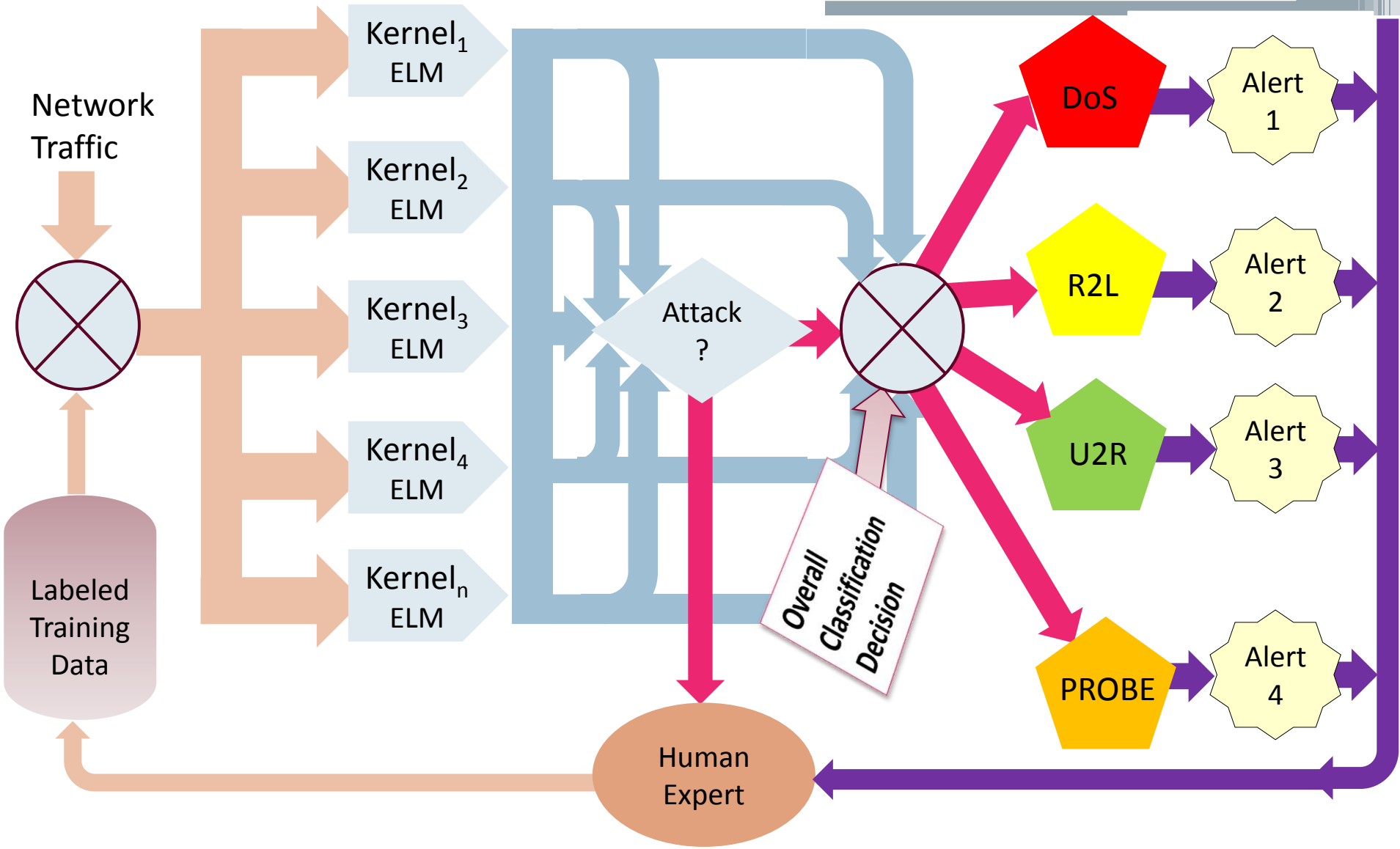
Dataset Size	SVM		ELM		Kernel ELM	
Training/Testing	Accuracy (%)	95% Confidence Interval (%)	Accuracy (%)	95% Confidence Interval (%)	Accuracy (%)	95% Confidence Interval
1000/1000	97.58	97.52 - 97.64	96.83	96.40 - 97.23	97.78	97.72 - 97.83
2000/2000	98.31	98.27 - 98.34	97.07	96.77 - 97.37	98.81	98.76 - 98.86
4000/4000	98.69	98.66 - 98.72	97.00	96.68 - 97.32	98.74	98.70 - 98.78

Multi-Class Classification Performance Results

Cheng, Chi, Wee Peng Tay, and Guang-Bin Huang. "Extreme learning machines for intrusion detection." In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pp. 1-8. IEEE, 2012.

My Approach to the Problem

- Using a Systems Engineering approach to combine a number of successful techniques in order to optimize the overall decision process
- Basic Requirements:
  - Multiple Algorithm Combination (i.e. “Multiple Perspectives”, “Best of All Worlds”)
  - Online Processing (Suitability for Net Centric Operation)
  - Provision for Data Imbalance
  - Reduce False Alarms
  - Allow for Human Input (i.e. Tuning)


Network Traffic
Kernel1 ELM
Kernel2 ELM
Kernel3 ELM
Kernel4 ELM
Kerneln ELM
Attack ?
Overall Classification Decision
DoS
R2L
U2R
PROBE
Alert 1
Alert 2
Alert 3
Alert 4
Labeled Training Data
Human Expert

Very Preliminary Findings on UCI Machine Learning Data Set

Ionosphere	$OMKC_{D,D}$	Batch ELM Voting	Online ELM Process
Mistake %	16.07	1.89	5.89
Range	$\pm$ 1.42	$\pm$ 1.28	$\pm$ 2.74

Breast Cancer	$OMKC_{D,D}$	Batch ELM Voting	Online ELM Process
Mistake %	4.86	2.45	2.91
Std Dev	$\pm$ 0.51	$\pm$ 0.63	$\pm$ 1.03

Note: 100 Node ELM with Radial Basis Activation Function

Planned Future Work

- Pre-processing of KDD Cup '99 Data Set
- Analysis and Pre-processing of a Secondary Intrusion Data Set
- Incorporation of scheme for addressing Imbalanced Data issues (e.g. W-ELM, SMOTE)
- Optimal Feature Selection
- Incorporation of Human Expert in the Loop

Discussion/Q & A

John Fossaceca Contact Information
Ph.D. Student, George Washington University
VP, Technology & Product Management
Ultra Electronics, 3eTI

- jfossaceca@gwu.edu
- phone: +1 301.944.1359
- Company website: www.ultra-3eTI.com

Thank You!