Neutron-capture prompt-gamma activation analysis (PGAA) using diverse machine learning methods

J. Mathew, S. Kanarachos, M.E. Fitzpatrick
Faculty of Engineering and Computing
Coventry University

Nuclear Security Detection Workshop
15-16 April 2019
University of Surrey

OUTLINE

1. Background
2. PGAA database
3. Machine learning (ML) methods
4. Phase 1- results
5. On-going work
6. Optimisation studies: An Illustration with ANN

BACKGROUND

- Prompt gamma ray neutron activation analysis (PGAA) is an efficient non-destructive multi-elemental detection technique
- PGAA can be performed in-situ, identify most elements from hydrogen to uranium.
- It can characterise wide array of trace elements that may exhibit varying degrees of complexity in chemistry and highly sensitivity technique for performing elemental analyses in highly heterogeneous materials.
- No sample preparation needed and can be performed on samples held in containers made of low cross-section.
- **All these factors make PGAA an attractive candidate for radioactive material detection (eg. Uranium) in nuclear forensics**.

PGAA DATABASE

- International Atomic Energy Agency (IAEA) coordinated development of a database for Prompt Gamma-ray Neutron Activation Analysis.
- Inaccurate and incomplete data have been a significant hindrance in the qualitative and quantitative analysis of complicated capture-gamma spectra by means of PGAA.
- The measurements performed at the Budapest Reactor Centre for all natural elements (from H to U) includes data of **energy of the gamma rays** (E_γ) and **effective capture cross section** (σ).

https://www-nds.iaea.org/pgaa/databases.htm

RESEARCH OBJECTIVES

- Develop machine learning algorithms using python scripting to analyse PGAA data to classify the elements such as Cobalt, Caesium, Iridium, Uranium and Thorium based on the PGAA energy spectra (E_{γ}) and effective capture cross section (σ)
- Evaluate model performance based on classification metrics namely accuracy, precision, recall and f1-score.
- Evaluate the multi-class classification performance of different machine learning algorithms using confusion matrix and Receiver Operating Characteristic (ROC) curve
- Optimisation studies and robust validation to identify ML algorithms that are best-suited for classifying elements using PGAA (low dimensional data)

MACHINE LEARNING (ML) METHODS

- K nearest neighbours
- Decision trees
- Random Forest
- Artificial neural networks
- Support vector machines
- K-means clustering

Supervised learning

Unsupervised learning

CLASSIFICATION METRICS

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where *TP* = True Positives, *TN* = True Negatives, *FP* = False Positives, and *FN* = False Negatives.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

F_1 score is the harmonic mean of precision and recall:

$$F_1 = \left(\frac{\text{recall}^{-1} + \text{precision}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

[1-Best, 0-Worst]

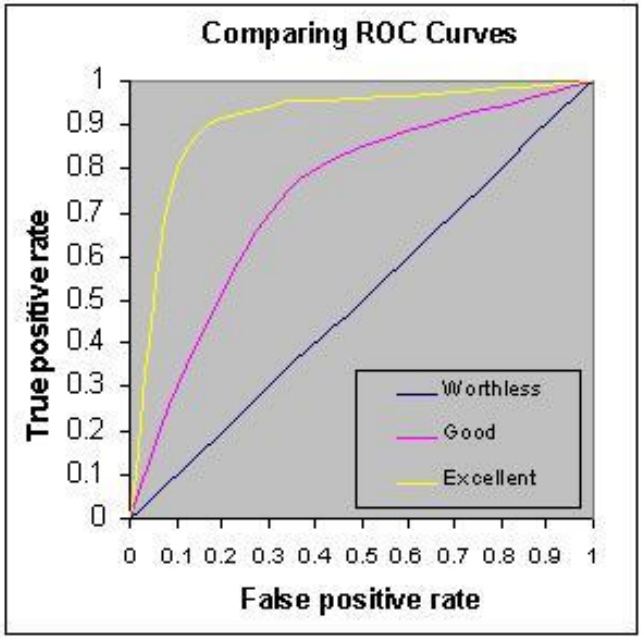
CLASSIFICATION METRICS

Confusion matrix: Model's ability to identify the existing classes in a given data set

		Predicted class: P	Predicted class: N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

For multi-class classification, matrix can be N by N where N is the number of classes in the data set.

An **ROC curve** (**receiver operating characteristic curve**) is a graph showing the performance of a classification model at all classification thresholds. This curve plots True Positive Rate and False Positive Rate.



RESULTS – PHASE 1

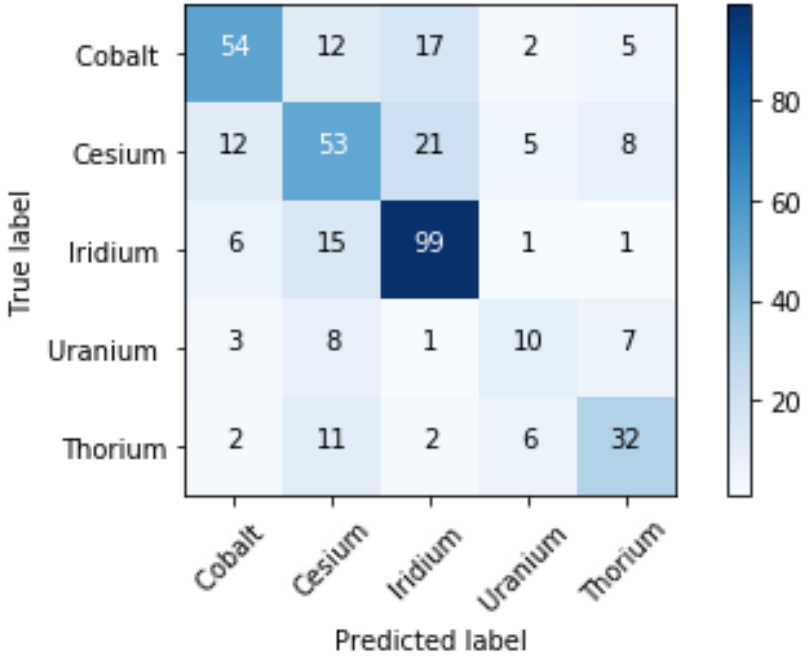
Results using test data for Decision Trees

Output classes

0	Cobalt
1	Cesium
2	Iridium
3	Uranium
4	Thorium

	precision	recall	f1-score	support
0	0.70	0.60	0.65	90
1	0.54	0.54	0.54	99
2	0.71	0.81	0.76	122
3	0.42	0.34	0.38	29
4	0.60	0.60	0.60	53
avg / total	0.63	0.63	0.63	393

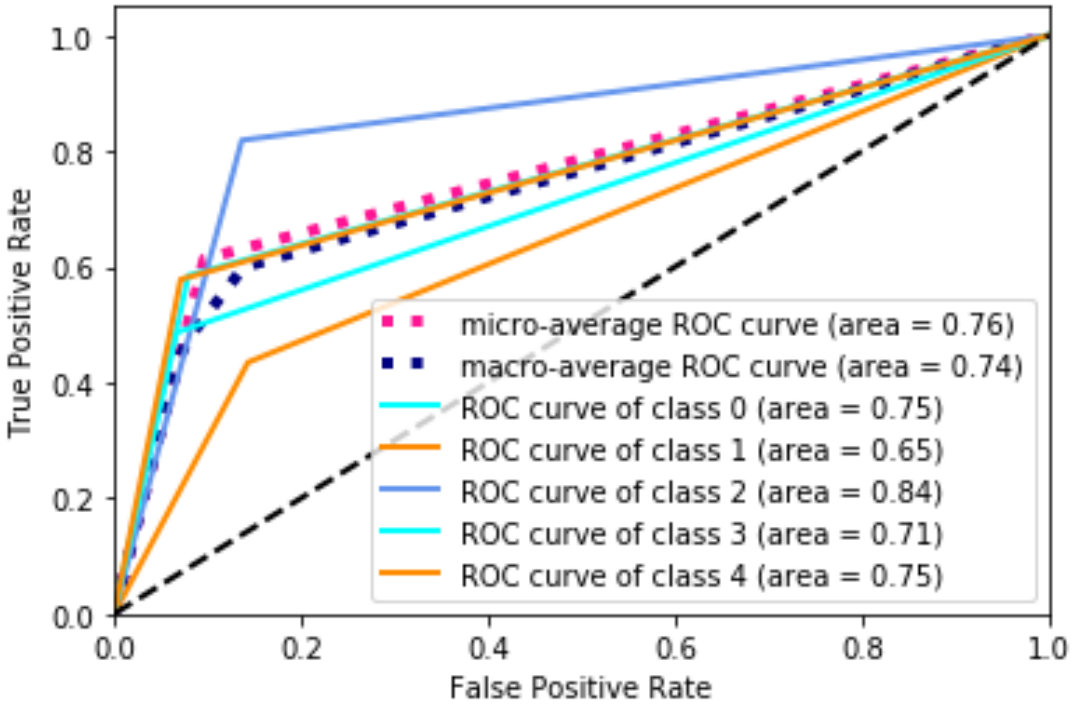
(a) Classification report



(b) Confusion matrix

RESULTS – PHASE 1

Results using test data for Decision Trees



(a) Receiver Operator Characteristic (ROC) curve

RESULTS – PHASE 1

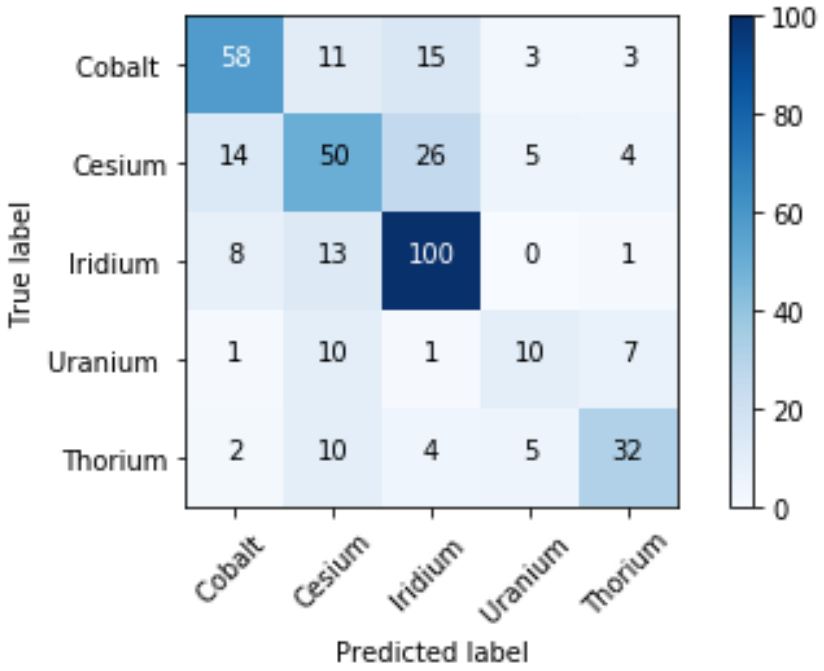
Results using test data for Random Forest

Output classes

0	Cobalt
1	Cesium
2	Iridium
3	Uranium
4	Thorium

	precision	recall	f1-score	support
0	0.70	0.64	0.67	90
1	0.53	0.51	0.52	99
2	0.68	0.82	0.75	122
3	0.43	0.34	0.38	29
4	0.68	0.60	0.64	53
avg / total	0.63	0.64	0.63	393

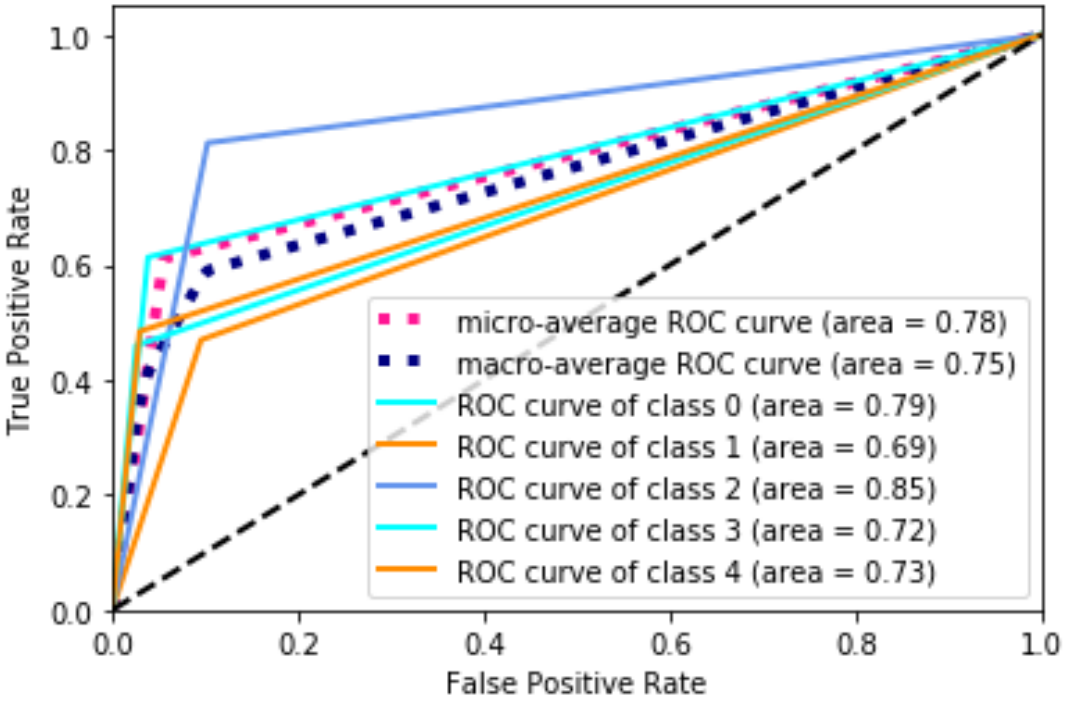
(a) Classification report



(b) Confusion matrix

RESULTS – PHASE 1

Results using test data for Random Forest



(a) Receiver Operator Characteristic (ROC) curve

ON-GOING WORK

Optimisation studies

- Rigorous optimisation studies should be performed to draw a fair comparison with the different ML algorithms.
- Methods based on ANN and SVM have more free parameters that need to be iteratively optimised

Robust Validation

- Machine learning algorithms are currently validated using independent test datasets, however an effective cross-validation mechanism should be included to address key validation issues
- Implement leave-p-out or K-fold cross validation methods

How ANN works?

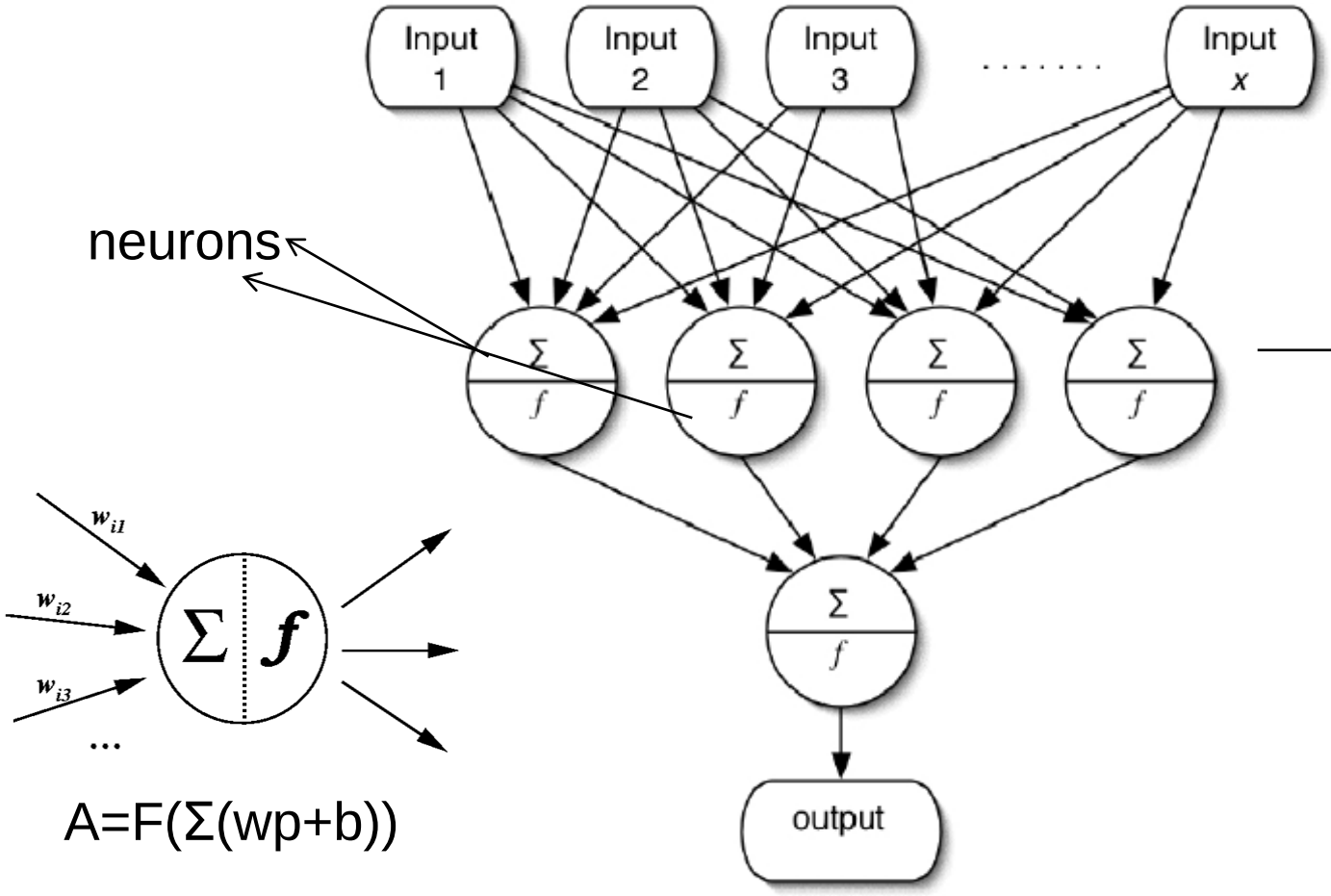


Fig. Structure of a 2 layer neural network

Bayesian framework for regression

$$E(w) = \beta Es + \alpha E_R$$

$$E_S = \frac{1}{2}\sum_{i=1}^{M}\{t - y(p, w)\}^2$$

$$E_R = \frac{1}{2}\sum_{i=1}^{R}\left|w_i\right|^2$$

w = weight and bias matrix in the network, *α* = hyper parameter controlling weight decay or regulariser, *β* = hyper parameter controlling the variance in noise, *y* = simulated output for Input I, *t* = target value, *p* = input vector, *M* = no: of patterns

Hyper-parameters control other parameters in the neural networks (weights and bias)

Methodology

$$p(w/D) = \frac{p(D/w)p(w)}{p(D)}$$

$$Posterior = \frac{Likelihood \times \Pr ior}{Evidence}$$

Evaluation of hessian matrix based on outer product approximation

$$\frac{\partial^2 E}{\partial w_{ji} w_{kj}} = \sum_n \frac{\partial y^n}{\partial w_{ji}} \frac{\partial y^n}{\partial w_{kj}} \quad when\ y^n - t^n\ is\ small$$

Methodology (contd.)

The evidence approximations is performed in two steps: (1) computing the w_{MP} in maximizing penalized likelihood and (2) periodically re- estimating the hyperparameters.

Find H^{-1}, Eigen values, determinant of H

Find gamma $\gamma = \sum_{i=1}^{R} \frac{\psi}{\psi + \alpha}$

$$2\beta E_S = N - \sum_{i=1}^{R} \frac{\psi}{\psi + \alpha} = N - \gamma$$

$$2\alpha E_R = R - \sum_{i=1}^{R} \frac{\psi}{\psi + \alpha} = \gamma$$

Evaluate new alpha and beta, reiterate using $\alpha_{new} = \frac{\gamma}{2E_R}$ $\beta_{new} = \frac{N - \gamma}{2E_s}$

Methodology (contd.)

Steps

1. Choose initial values for hyper-parameters α and β. Initialize the weights in the network using values from the prior distribution

2. Train the network to minimize the error function $E(w)$

3. Periodically re-estimate values of α and β (using Hessian matrix, Eigen value spectrum and parameter γ)

4. Repeat steps 1-3 for different choices of network weights in order to evaluate the local minima

5. Repeat steps 1-4 for different network models and compare their evidence

THANK YOU !

Acknowledgement: This work is funded by the PDRA grant awarded by the Nuclear Security Science Network (NuSec) in the United Kingdom