I. INTRODUCTION

In many scenarios, the ability to quickly decide if a radiation source is a threat is crucial. While many methods exist to accurately identify the precise isotope or combination of isotopes from a spectrum, these algorithms are generally computationally expensive, making them unusable in scenarios with a large number of spectra, many of which may not even contain useful information.

To approach this problem, I used neural networks and a naive Bayesian classifier to classify spectra into “threat” and “non-threat” categories. The Bayesian classifier is the subject of my research at UIUC and is described with some detail in Section II-C. Neural networks have the advantage of being very fast, but as shown later they do not perform as well as desired. While this classification may seem excessively simple, this is an active research area in nuclear forensics and has applications in both nuclear emergency (e.g. post-detonation forensics) and non-emergency (e.g. checking shipping crates) scenarios.

To compare the different methods, by far the most important metrics are the correct identification rate vs false alarm rate (i.e. the ROC curve) and the computational complexity (analyzed by number of operations needed and also by average run-time). The loss function is also analyzed for the neural networks.

II. METHODOLOGY

A. Data

For training and test data, a small NaI scintillator was used to observe 30-second gamma-ray spectra in our lab. For non-threat spectra, some measurements on the natural background radiation of our lab were taken (natural background primarily includes gammas from the decay chain $^{232}$Th, found in dirt and bricks, from $^{40}$K, from cosmic background, and a few other trace sources). To simulate different background levels, a brick of natural thorium and a brick of $^{40}$K were placed in various configurations around the room, with different attenuating materials between the source and the detector.

For threat spectra, sources containing $^{152}$Eu, $^{60}$Co, $^{133}$Ba, and $^{151}$Eu were placed in different locations around the lab, varying location and attenuating materials. While there are hundreds of sources that must be identifiable, these four together produce gamma-rays over the most interesting detectable range (from approximately 300 keV to 1600 keV, with a max detectable energy of around 3000 keV). While there are a few isotopes with gamma emissions at a higher energy (including the decay chain of $^{232}$Th), gammas in the 300-1600 keV range are the most common. Sample spectra with good counting statistics are shown in Figure ??.

Each spectrum is a 1024x1 vector, where the $i^{th}$ element is the number of counts binned into the $i^{th}$ channel. Each channel corresponds to a small energy range (approximately 3 keV per channel).

To normalize the data, the neural networks are trained with $z_j = \log(x_j + 1)$, where $x_j$ is the $j^{th}$ spectrum. Putting the counts into a log-space is necessary because the low-energy channels ($\leq$ 300 keV) often have 2-4 orders of magnitude more counts than the high-energy channels.

B. Neural Networks

The neural networks are trained by gradient descent with random initializations and with simulated annealing. The gradient descent gets stuck in poor local optima very frequently, which is why I added the simulated annealing method to compare.

C. Bayesian Classifier

Peak information (centroid, area) is extracted from the spectrum using Origin. There are several “better” methods for peak extraction, but this step is not the focus of this project and I opted to keep it simple. Our research group is developing a wavelet-NNLS approach for automated peak extraction as well, but I was not able to use it due to time constraints.

The idea of a Bayesian classifier is simple: for a given data set $D$ (in this case a list of peak energies and areas from a spectrum), the probability that an isotope or set of isotopes $M_k$ is present is:

$$P(M_k|D) = \frac{P(D|M_k)P(M_k)}{\sum_k P(D|M_k)P(M_k)}$$

The prior used for this project is inversely proportional to the number of isotopes in the model, i.e. when using an isotope library with $n$ isotopes, $P(M_k) \propto 1/n$ if $M_k$ contains one isotope, $P(M_k) \propto 1/n^2$ if $M_k$ contains two isotopes, and so on. Combinations of up to three isotopes are considered for this project.

The likelihood $P(D|M_k)$ is approximated as the product of four terms:

$$P(D|M_k) = f_{LPI}f_{DPI}f_{FP}f_{AR}$$
where

- $f_{LP}$ scores how well the library isotope’s peaks (model $M_k$’s peaks) is represented by the data. Each library peak that we expect to see should ideally be present in the data. Missing peaks are penalized, with large area and high energy peaks weighted more than small area or low energy peaks (low area peaks may be missed by peak fitting algorithm, and low-energy peaks may be attenuated out by shielding).
- $f_{DP}$ scores how well the data peaks are represented by the model $M_k$, similarly to $f_{LP}$.
- $f_{PP}$ scores how closely the peaks in the data set match the library (model) peaks. With a perfectly calibrated detector and a perfect peak-fitting algorithm and perfect counting statistics, the data peak should have the exact same energy centroid as the library peak. However, several effects cause a shift in this energy, and the deviation between library peak and data peak centroids is approximately Gaussian.
- $f_{AR}$ scores how well the areas in the data compare to the expected areas. Since count time / source activity / distance to the source / attenuating material will all influence the number of counts (area) observed (linear increase in time and source activity, $1/r^2$ decrease in distance, etc), we compare the ratio of areas of neighboring peaks. Peaks that are close in energy are attenuated similarly, so ideally the area ratio should match the library ratio precisely. Attenuation will slightly decrease these ratios (shielding affects lower-energy peaks more), but no physical effect should increase these ratios.

A more thorough explanation of model terms can be found in [?].

The Bayesian classifier uses a spectral library (set of all possible models) that contains all of the peak energies and areas (convolved with the detector efficiency). Tables of all of the known energies and branching ratios for isotopes are publicly available. Since some isotopes have many peaks that cannot be resolved with a NaI scintillator (too small or overlapping with others), peaks are reduced using the peak reduction algorithm of a past master’s thesis in our group [1]. For this problem we are only interested in threat vs non-threat discrimination, we can sum all of the threat-models to get the posterior probability that a threat source is present.

For $i = 1, 2, \ldots$:

- Load model $M_i$
- Calculate $P(M_i|D) = P(D|M_i)P(M_i)$
- Normalize $P(M_i|D)$
- Sum $P(M_i|D)$ over all threat sources

While this Bayesian classifier approach is more computationally intensive than desired, my goal is to eventually turn it into a suitable replacement for the identification algorithms onboard handheld radio-isotope identifiers (RIDs), where a few seconds is an acceptable computation time. RIDs are widely used, but their performance is inadequate even in the settings that they’re currently used.

### III. Performance

#### A. Correct Identification vs False Alarm Rate

To gauge performance, the correct identification rate is compared to the false alarm rate. While there has been a large push to improve these classification algorithms in the past decade, since this project is relatively simple mathematically I feel that it’s reasonable to compare performance here to the performance of circa-2003 detectors. One published evaluation found that across 7 different detectors (each with their own proprietary algorithms) and several hundred measurements, these detectors had an average correct or partially correct identification rate of approximately 70% with a false alarm rate of approximately 25% [2]. While these detectors were making identifications (rather than a threat/non-threat classification), it is still a useful starting point.

The performance of the gradient descent-trained 2-layer neural networks are shown in Figure 1. The best networks found here have approximately an 80% correct identification rate with a 25% false alarm rate, comparable to the old detectors. It is possible that with more computation time, some improvement could be found with more random initializations since gradient descent quickly gets stuck in local optima. However, each line on the ROC curve was generated with approximately 10,000 different initializations, which has a very long run-time (hours per line).

To try to find better optima, after the oral presentation I implemented simulated annealing. The results are shown in Figure 2. Only 100 different initializations were trained for this figure; simulated annealing with a long temperature distribution is much less likely to land in a local optima instead of the global optima than the previous gradient descent method is. A 16-node network was found that achieves much better identification vs false alarm rate (85% correct vs 2% false alarm) than any other solution found.

The Bayesian classifier currently has a 94% correct classification rate vs a 9% false alarm rate. Results have been improved on since the class presentation by better fitting of the peaks using Origin. Nearly all of the missed identifications are due to very poor counting statistics; in some of the spectra, peaks are not visible by eye and were missed by the peak fitting routine. For the false alarm instances, Origin found “false peaks”, which are merely noise peaks due to poor counting statistics. These could be removed from the data by filtering based on peak width information (a short-term goal of the Bayesian identifier project is to include peak-width information in the classification process).

#### B. Run-time

Besides having a high probability of correct identification and a low probability of false alarm, the other truly important factor is the run-time needed to make a decision. The time it takes to train a network is not important for this purpose; only the final classification time would impact the decision of whether or not to use an algorithm.
Fig. 1. ROC for 2-layer neural networks trained with gradient descent

Fig. 2. ROC for 2-layer neural networks trained with simulated annealing

The decision time for each of the methods is listed in Table III-B. For the networks, the decision time was averaged over the full test data set. Since the test time is identical for both of the neural network training methods (clearly, the test step is the same), it is not listed twice. For the Bayesian classifier, the run-time does not include the peak-extraction time in Origin (approximately 0.5 seconds per spectrum, lots of unnecessary computational overhead).

While the Bayesian classifier is well-suited for a small number of spectra, where 10-20 seconds of run-time would be acceptable, it is not an option in some of the time-sensitive scenarios mentioned earlier, where 50000 spectra would result in 20 hours of computation time, compared to 0.5 seconds for the neural network approach.

### C. Loss vs Iteration

Earlier, it was demonstrated that simulated annealing produced more accurate networks than the gradient descent networks. Given enough run time and random initializations, this wouldn’t be the case, but the simulated annealing finds good optima relatively quickly.

In Figure 3, we see an admittedly terrible run of gradient descent. Only one in a few thousand random initializations produces a useful network (one with less than 50% training error).

In Figure 4, a much more reasonable loss vs iteration is presented. Simulated annealing reaches a steady-state solution around 1500 iterations regularly, and generally converges to a useful solution.

As expected, the simulated annealing vastly out-performs the gradient descent method, so for all future experimentation with neural networks I will continue to use simulated annealing (or a similar method).

### IV. Conclusions and Future Work

To my surprise, the neural network approach seems to be a suitable approach to this problem. I assumed that if it worked well that it would have been the accepted solution several years ago (or at least more widely used). For these results to be truly useful, a much larger training and test set need to be observed, with different radiation sources and different conditions (temperature, shielding, distance, etc). This will result in more regions of a spectrum to be correlated with threat sources, which may degrade performance.

In the near future, I have a few experiments to try with the neural networks. In particular, I’m interested in training networks on wavelet-transformed spectra and on using feature extraction methods (such as peak fitting) and training networks with extended data sets (i.e. give the network a spectrum and a list of peak energies, areas, and uncertainties to make decisions). As suggested by Dr. Hasegawa-Johnson, it would also be interesting to use the likelihood model terms in the network data as well. It’s very likely that our group will pursue some of the methods discussed in this class more seriously to work on this problem; another student in our research group is working on this problem for his thesis, and I believe this will be the route he goes eventually with it.

### REFERENCES


Fig. 3. Training error for gradient descent

Fig. 4. Training error for simulated annealing