1. Intrinsic and Extrinisc Methods to Detect Overfitting

Intrinsic methods depend only on the model and training data.

Extrinsic methods, in contrast, use additional knowledge such as:
- performance on a hold out set
- process used to find the model (e.g., multiple hypothesis testing with registration)
- complexity of the function family of the model (e.g., VC dimension, Rademacher Complexity)
- knowledge of size of parameter space of the model (e.g., Akaika Information Criterion)

Arthur has a public dataset \( S \) (drawn from a distribution \( D \)) for which he wants to build a model. Arthur outsources the model creation to Merlin who is not entirely trustworthy. Merlin comes back with a model \( M \) but does not disclose any other details of his modeling process.

How can Arthur convince himself that \( M \) is not horribly overfit, i.e., \( M \) is not just a lookup table? Recall that Arthur has no private hold out. If Arthur only can access \( M \) as a black box, there is not much he can do. But what if Arthur has access to the internal signals of \( M \)?

(Usually a model \( M \) can be described at different levels of abstraction but, in this work, we work at the lowest level of abstraction which is that of primitive logic gates. Thus Merlin does not even tell Arthur what sort of model it is but just presents a circuit.)

Key Observation. As we simulate the training examples in \( S \) through the lookup table we find that there are signals that identify specific training examples. For example, the signal \( s_0 \) is 1 only once (when \( x = x_0 \)) and is 0 everywhere else. We say that 1 is a rare pattern for signal \( s_0 \).

5. A Benchmark Problem

While \( l \)-CFS detects overfit for lookup tables, how about neural networks and random forests? We trained several models on MNIST, compiled them down to primitive logic gates, and applied \( l \)-CFS.

Each neural network has 784 inputs and 3 hidden layers with 256 nodes each and a final softmax with 10 outputs and is fixed-point quantized to 8 bit weights, 16 bit activations and 24 bit accumulation. Compiled down to gates this leads to circuits with 35M to 52M primitive gates.

- nn-real-2: Trained for 2 epochs (97% training and validation accuracies).
- nn-real-100: Trained for 100 epochs (99.90% training, 98.24% validation).
- nn-random: Trained for 350 epochs on randomized labels (91.27% training, 9.73% validation).

We also trained 2 random forests with 10 trees each with the default settings of Scikit-learn, except for bootstrapping. Quantizing them to 8 bits leads to circuits with 700K and 3M primitive gates.

- rf-real: Trained on MNIST (100% training and 95.58% validation).
- rf-random: Trained on MNIST with randomized labels (100% training and 10% validation).

Key Result. Within each model family, CFS distinguishes the degree of overfit, even from this low level of abstraction where most aspects of the high level structure are lost: distinction between weights and activations, and even distinction between a net or a forest.