Circuit-Based Intrinsic Methods to Detect Overfitting

Sat Chatterjee ⟨schatter@google.com⟩ and Alan Mishchenko ⟨alanmi@berkeley.edu⟩

Google AI   Berkeley UNIVERSITY OF CALIFORNIA

1. Intrinsic and Extrinsic Methods to Detect Overfitting

Intrinsic methods depend *only* on the model and training data.

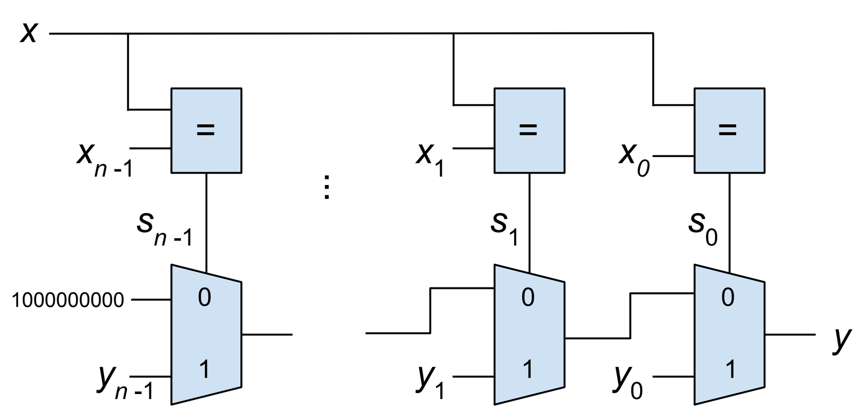
Extrinsic methods, in contrast, use additional knowledge such as

- 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., Akaike Information Criterion)

3. A Thought Experiment: The Adversarial Modeler

Arthur has a **public** dataset $\mathcal{S}$ (drawn from a distribution $\mathcal{D}$) 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 $\mathcal{M}$ but does not disclose any other details of his modeling process.

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



(Usually a model $\mathcal{M}$ 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 $\mathcal{S}$ 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

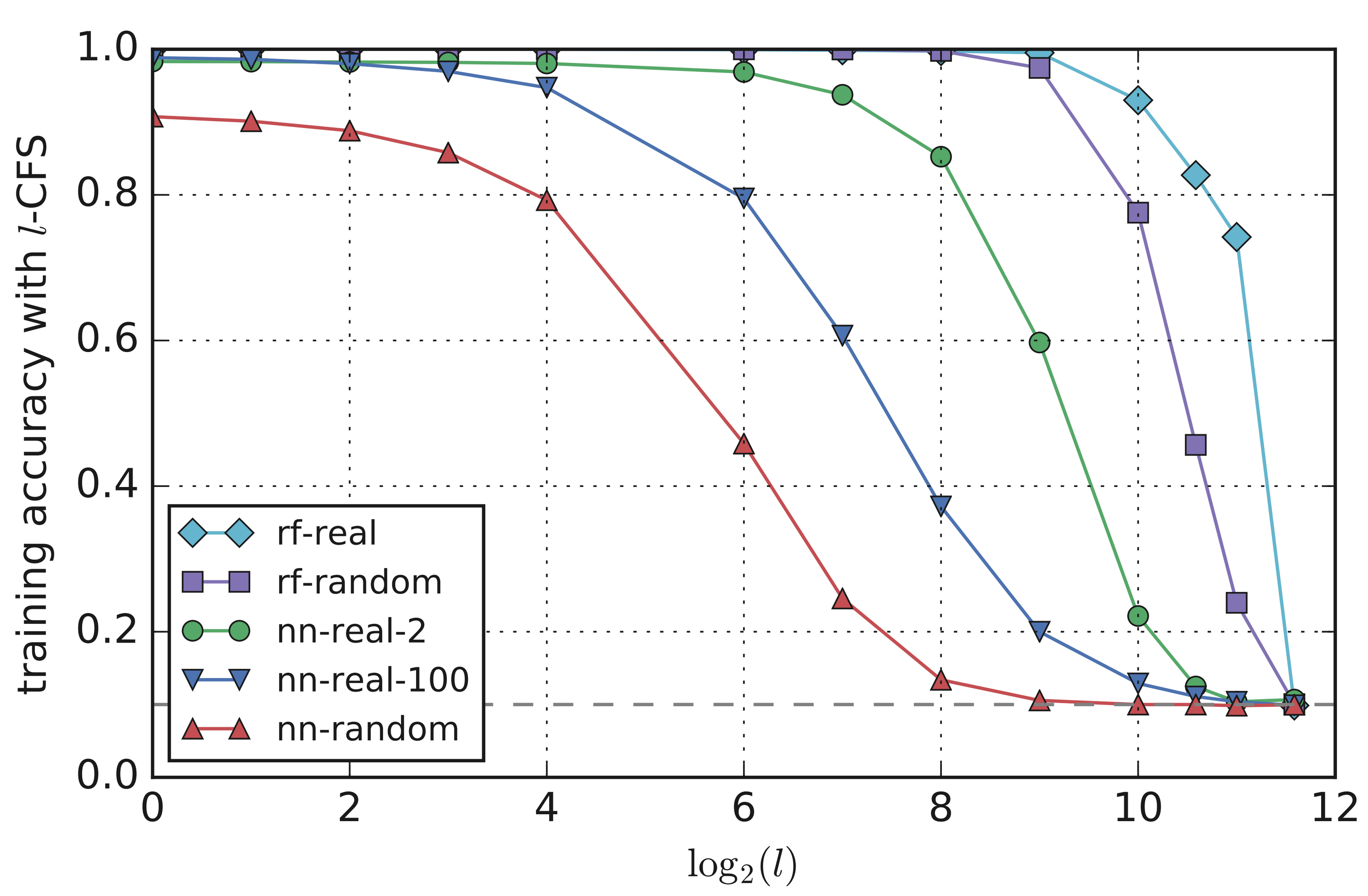
While l-CFS detects overfit for lookup tables, how about neural networks and random forests? We trained several models on MNIST, compiled them down to primitive logic gates, and applied l-CFS.

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 300 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.

2. Why Study Intrinsic Methods?

They can be practically useful:

- Complexity based bounds are vacuous for deep learning (and even random forests)
- Keeping a hold out means less data to train with
- Often hard to ensure pristine hold out in long running research
- No hold out when the data is public (and modelers are adversarial)

They can also provide interesting theoretical perspectives:

- Can Supervised Learning be solved with infinite compute?
- Can we have a certificate of generalization? Is Supervised Learning in NP?
- Explain why nets generalize (e.g., normalized margin, curvature, etc.)?

4. Counterfactual Simulation (CFS) to Detect Overfit

l-CFS is a method to detect overfit based on the above observation. It uses two ideas:

Idea 1: Identify l-rare patterns in the circuit.

- If a signal s takes on the value v at most l times on $\mathcal{S}$, then v is a l-rare pattern for s.
- The presence of l-rare patterns (especially many l-rare patterns for small l) suggests overfitting.

Idea 2: Resimulate the circuit while perturbing l-rare patterns and measure accuracy.

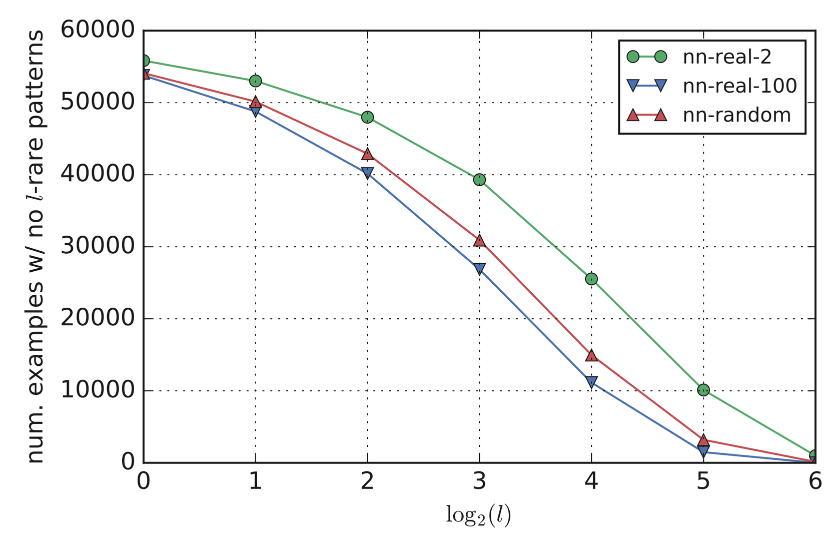
- This rules out observability don't-cares (ODC), i.e., patterns with no effect on the circuit outputs.
- A simple perturbation is flipping the value (in the lookup table example, s_0 becomes constant 0).

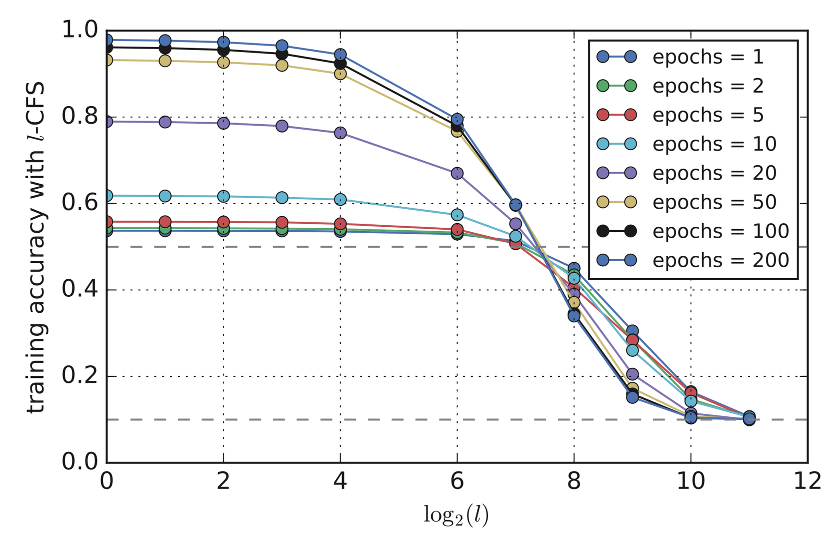
The degradation in accuracy due to l-CFS indicates the extent of overfit. For example, in the lookup table, 1-CFS leads to an extreme degradation with an accuracy no better than chance.

Note that CFS is naturally free of hyper-parameters.

6. What can CFS tell us about Stochastic Gradient Descent (SGD)?

Why do neural nets trained with SGD generalize when they have sufficient (effective) capacity to memorize random data? Our experiments based on CFS provide insights into this open question.

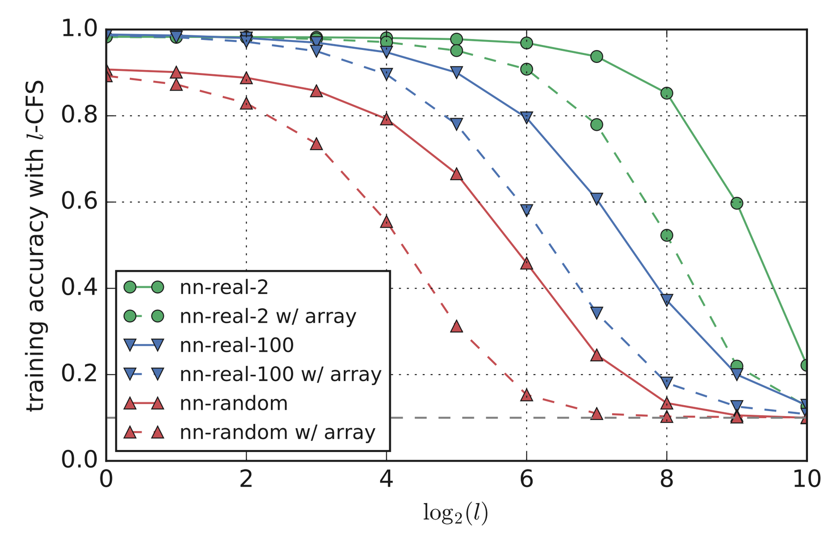


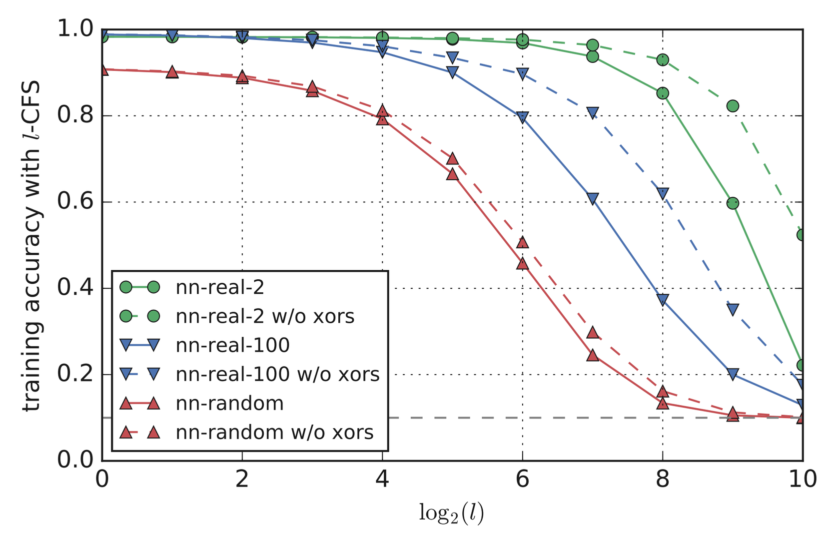


- Note that this phenomenon is not unique to neural networks and is also true of random forests. Random forests generalize when the algorithm is able to find commonality between examples.
- Analysis of rare patterns (left) shows that SGD **also** finds common patterns. It is direct evidence that, even for networks trained on random data, SGD does not "brute force" memorize.
- Experiments with a fraction of the labels randomized (right) provides evidence that SGD learns simpler examples first, before learning more complex examples (consistent with previous work).
- Simpler examples can be characterized as those having more in common with other examples, since they are more resilient to CFS perturbation (right).

7. Limitations of CFS and Future Work

There are two main limitations of the CFS procedure outlined here.





1. The results of CFS are sensitive to the choices made during compilation, such as the type of multipliers (left) and even the choice of primitives logic gates (right).
2. Even extremely overfit random forests are more robust to CFS than well trained neural networks. Thus, a tree construction approach can game CFS to prevent it from detecting overfit.

We have explored other variants of CFS. They have similar limitations to varying extents. Thus, CFS is not useful in the adversarial setting, which motivates looking for a more robust variant.