Myths of Representation Learning

Rich Sutton
Reinforcement Learning & Artificial Intelligence Lab
University of Alberta, Canada

with thanks to
Rupam Mahmood, PhD student

Representation Learning: Learning Slow to Enable Learning Fast

Rich Sutton
Reinforcement Learning & Artificial Intelligence Lab
University of Alberta, Canada

with thanks to
Rupam Mahmood, PhD student

What is Representation Learning (RL)?

- A learning process, generally over a long period of time, that enables subsequent learning to be *fast*
- RL enables fast learning!
- That was the original idea, and for many it remains the strongest idea
- But most of what goes on in our field is something different

Representation learning (RL)
Four meanings

RL is a relatively slow (2nd-order) process that results in:

1. Faster learning

2. Greater expressive power
   and thus better approximation of complex functions

3. Better generalization

4. Representations pleasing to people

Outline

- Representation learning should enable fast learning, but it doesn't
- How can we make RL about fast learning? What is required?
  - Online, continual learning, thus nonstationary (or sequences of learning tasks)
  - A stronger methodology, allowing for more solid conclusions
- A proposal in the form of a synthetic challenge task
- Some results...almost on the challenge task

Online, continual learning

- How can RL, a slow, 2nd-order learning process, enable fast learning?
  - How can slow learning enable fast learning?
- You have to have the slow learning first, then the opportunity for fast learning
- Thus, learning must be online, continual
- It cannot be one batch of data, then no more learning
- It could be a sequence of tasks...
- But the most elegant way is a non-stationary task – non-stop learning, with temporal symmetry

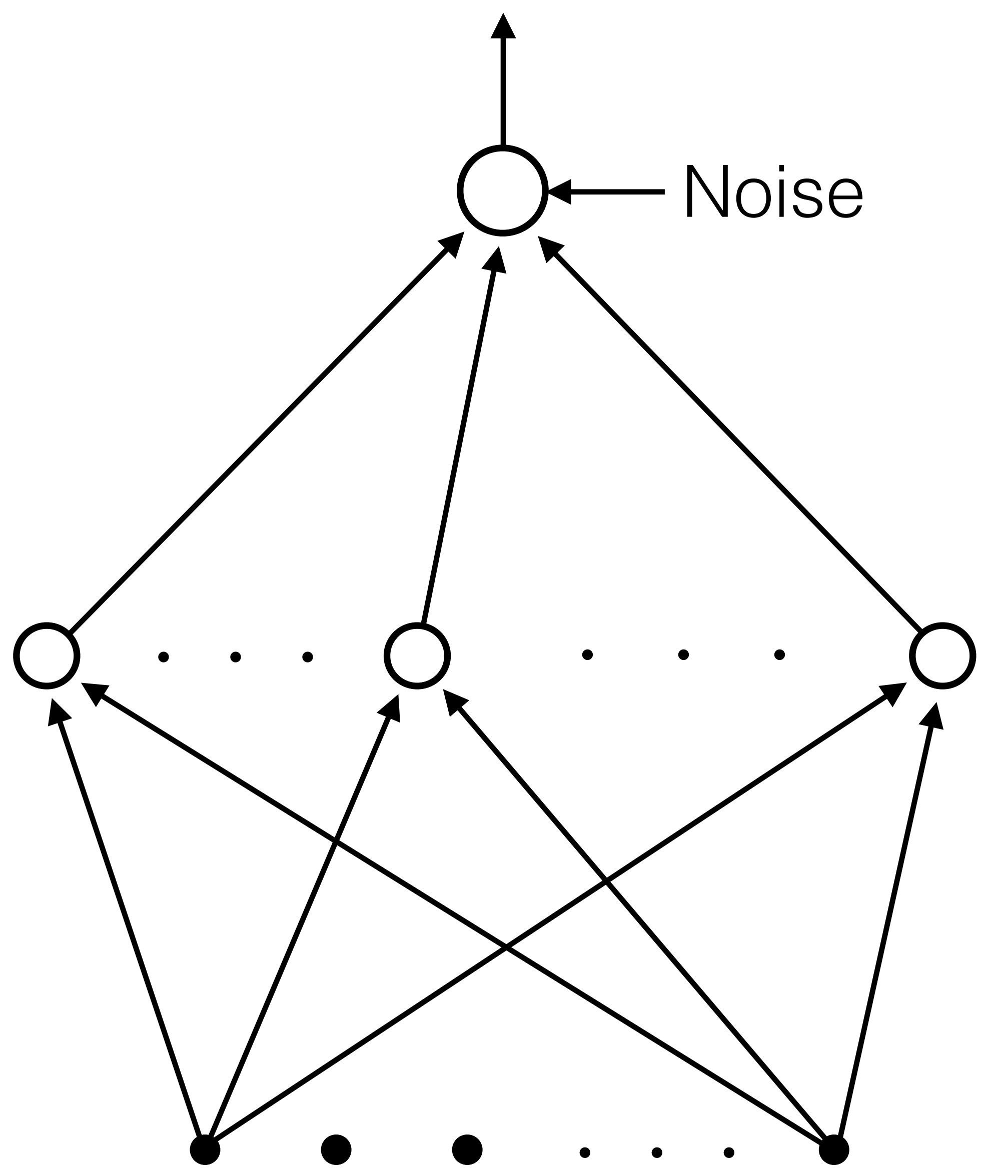
The GEOFF challenge
(GEneric Online Feature Finding)

- A generic, synthetic, feature-finding testbed – infinitely many task instances
- Each task has different ideal features (randomly chosen)
- *Online regression* (i.i.d., squared-error loss, no test set)
- Target function is a *two-layer network* with *random weights*
    - the hidden units are the ideal 'target' features
    - the output layer is a single linear unit with *non-stationary weights*

The GEOFF 'target' network

that generates the training data for learning

Output unit
(Real-valued)



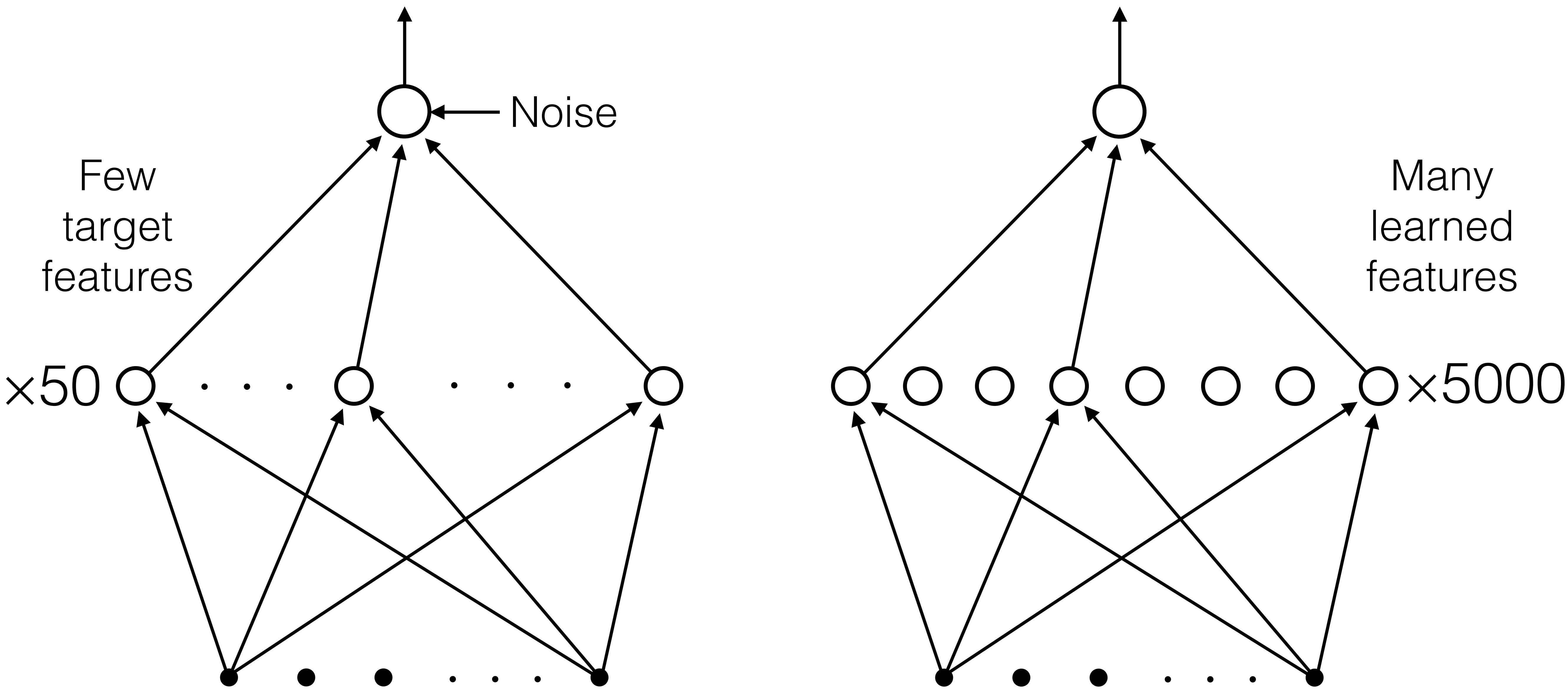
Slowly changing output weights
$\in \{+1, 0, -1\}$

Feature layer
$50 \times \{0,1\}$
(linear threshold units)

Fixed random input weights
$\in \{+1, -1\}$

Input layer
$20 \times \{0,1\}$
(random input bits)

Target network   Solution network



Both networks have the same structure, one is learned.
Tests our algorithms' ability to find good features efficiently.

Benefits of the GEOFF problem

- Direct measure of “RL enabling fast learning” (as asymptotic error)

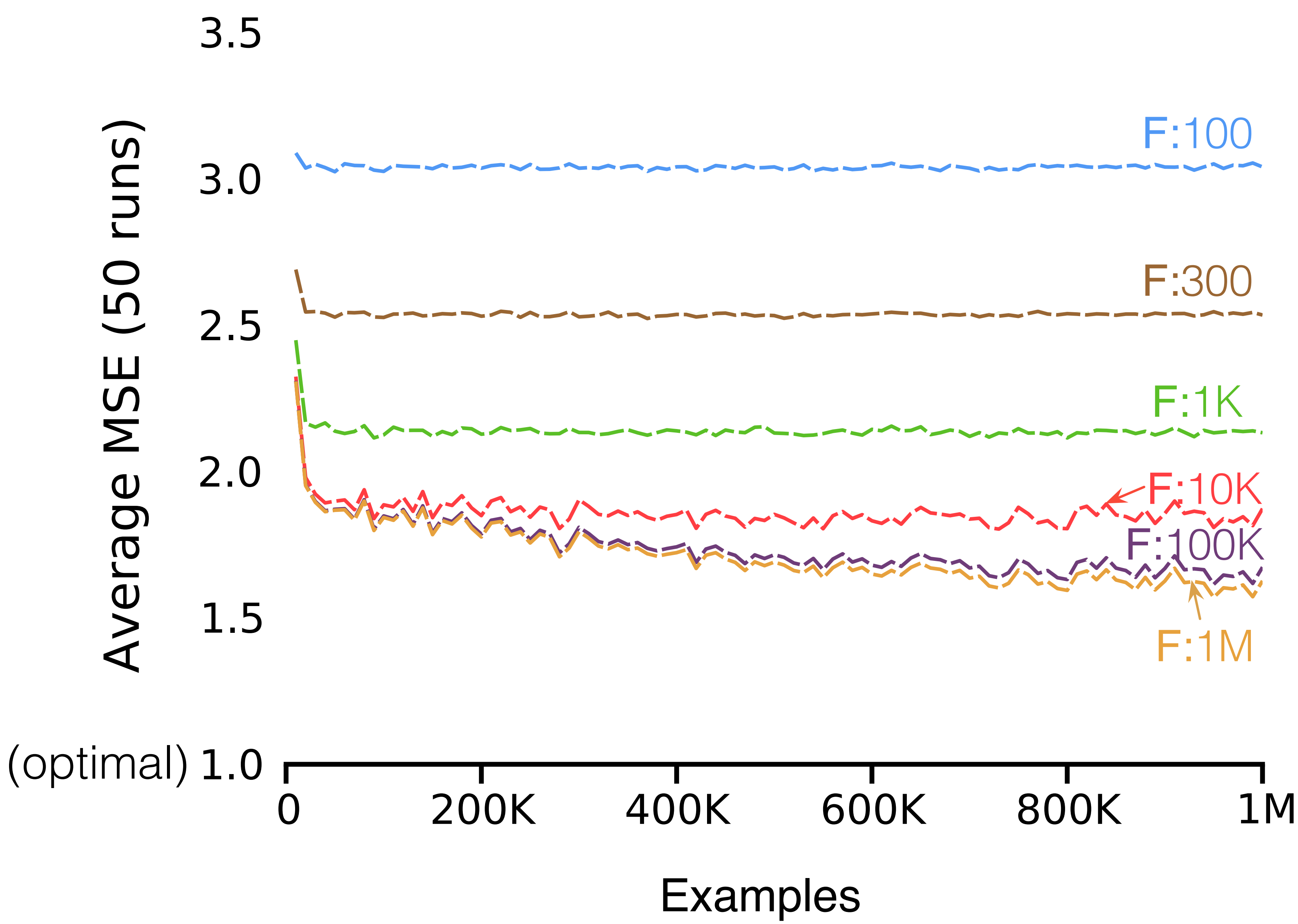
because it’s nonstationary

- Direct, sensitive measure of feature-finding ability (as rate of reduction of error)

because it’s synthetic

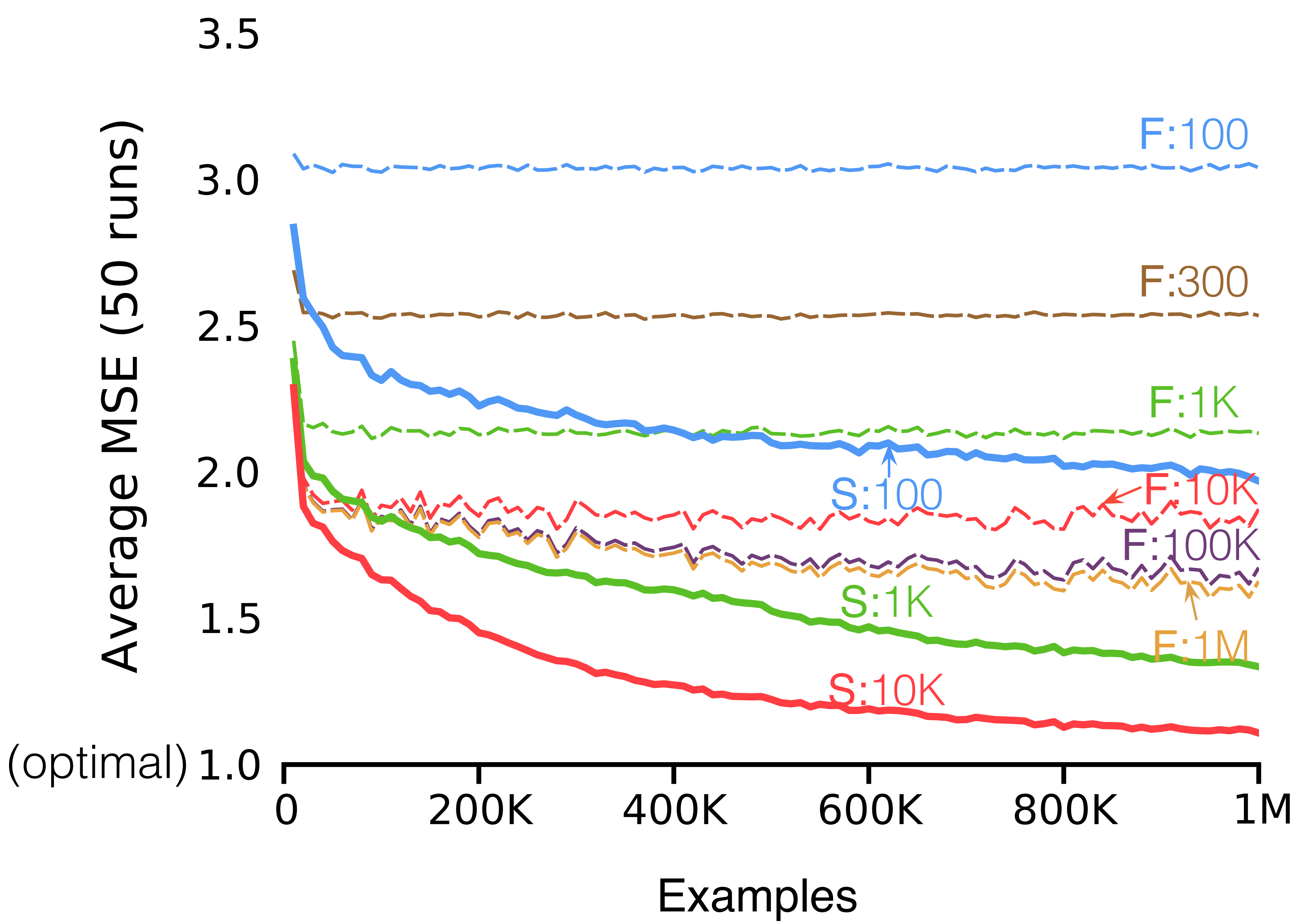
- Little domain knowledge; all of it explicit
- No possibility of test-set leakage
- No role for positive proxies (still a role for negative proxies)
- Objective; no reliance on human assessment of rep’n
- Small, easy to implement

Problem: Stationary GEOFF
Solution #1: *Many static features*



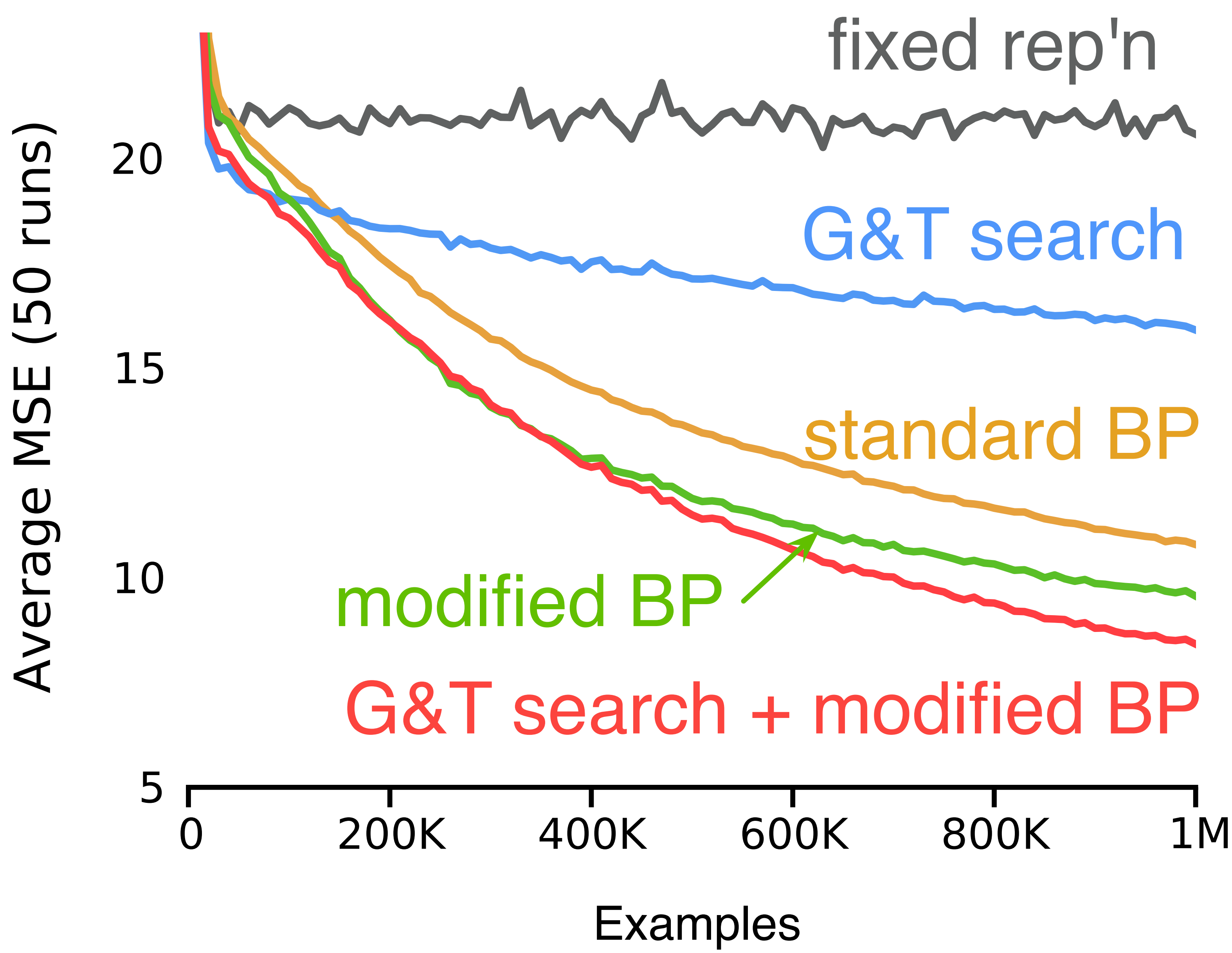
- Solution network:
  - input weights random and static
  - output weights learned by gradient descent
  - vary numbers of features
- 20 target features in the target network
- Apparently, the more features the better, up to a point

Problem: Stationary GEOFF
Solution #2: *Generate & test search*



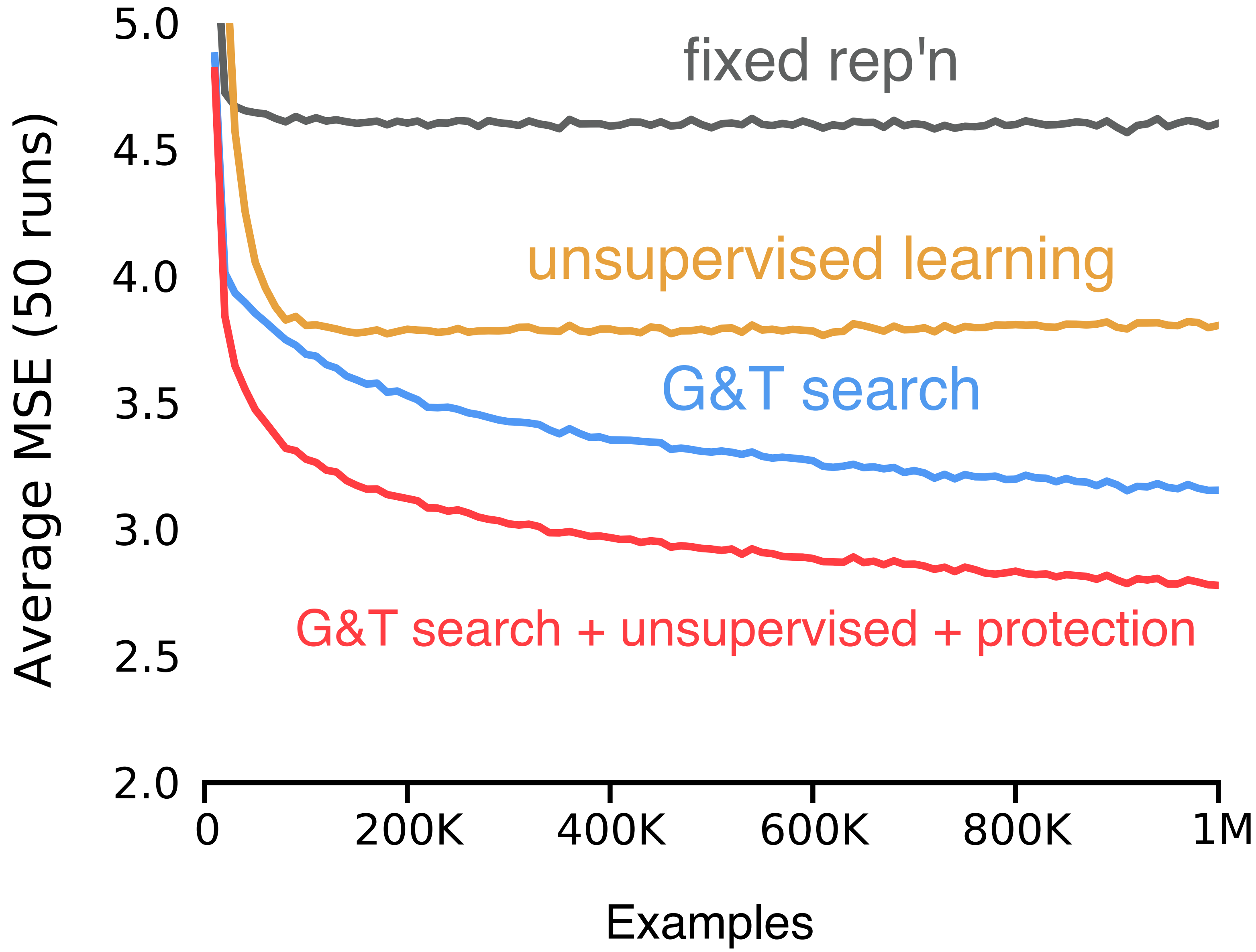
- Generate & test search is static features plus:
    - Rank utility of features
    - Slowly replace the least useful features with newly minted ones
- Apparently, G&T search enables better performance with fewer features

Problem: Stationary GEOFF
Solution #3: *Add backpropagation*



- Now 500 target features and 1000 solution features
- Backpropagation (BP) is gradient descent throughout the solution network
    - features are now tanh units rather than threshold units
- Modified BP removes the effect of the magnitude of the output weight
- Apparently, both gradient descent and G&T search contribute to efficient feature finding

Problem: Stationary GEOFF
Solution #4: *Add unsupervised learning*



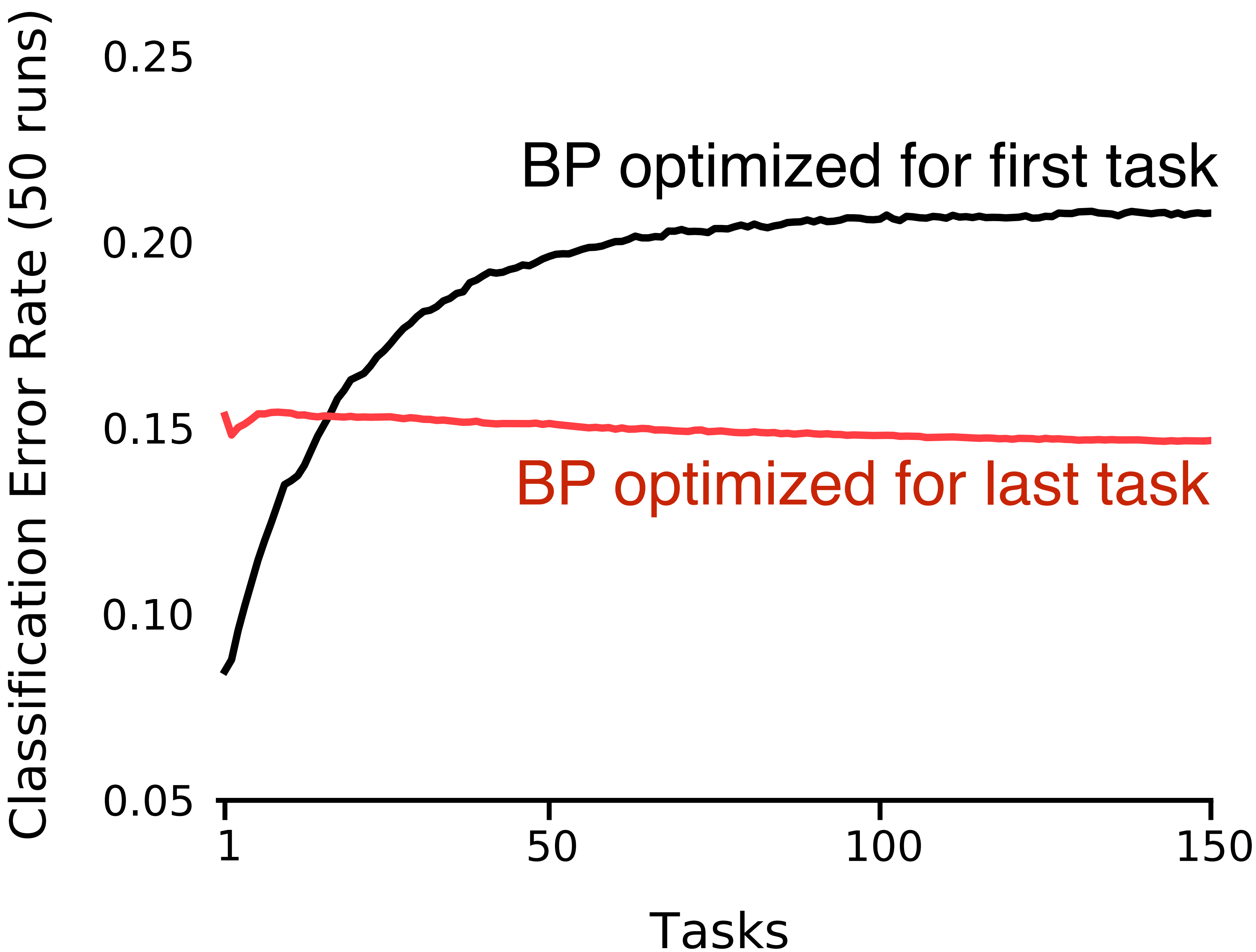
- Now 100 target features and 200 solution features
- Now input distribution is not uniform
- Unsupervised learning adjusts the solution features
    - so that each is active on ~20% of the examples
    - so that each example has ~20% active features
- Protection means the top half of features are not adjusted
- Apparently, this negative proxy can significantly improve G&T search

But what about fast learning?

- And what about the non-stationarity needed to measure it?
- There is some evidence that backprop performs poorly on non-stationary tasks

Mahmood, 2012

Problem: Non-stationary MNIST
Solution: *Backpropagation*



- MNIST modified to be a sequence of tasks, each with the same features, but different output labels
- Each task is full MNIST with 60,000 examples
- The mapping from number labels to the 10 output nodes is shifted by one in each successive task
- Backprop does not improve significantly on later tasks
- In fact, it tends to perform worse

But what about fast learning?

- And what about the non-stationarity needed to measure it?
- There is some evidence that backprop performs poorly on non-stationary tasks
    - it tends toward catastrophic interference
    - seems to be a need to protect useful features from being "taken over" for the new learning
- Step-size adaptation is part of the answer, and has been studied in a non-stationary setting

Sutton 1992; Mahmood 2010

Non-stationary step-size problem

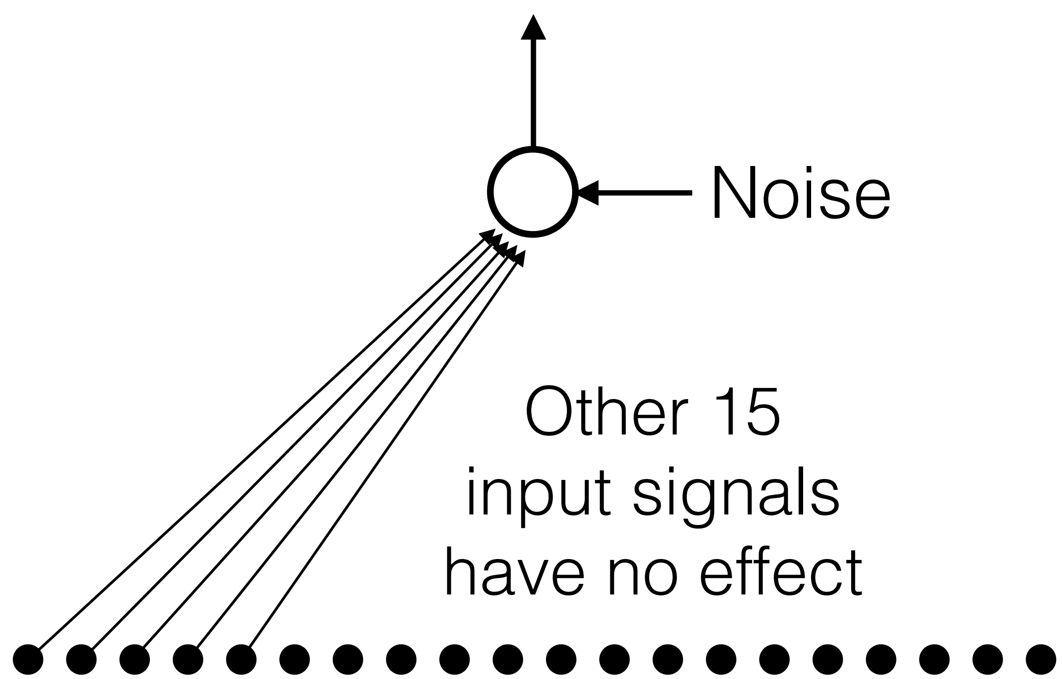
- Online linear regression (iid, squared error loss)
- 20 input signals, all standard normal N(0,1)
- Think of them as static features with output weights
- The target function is a weighted sum of the first five signals, where all the (target) weights are either +1 or -1
- The learned function is a weighted sum of all 20 input signals, with the learned weights adapted by gradient descent
- Step-size parameters, one per feature, are adapted by meta-gradient descent (the Incremental Delta-Bar-Delta algorithm, Sutton 1992)
- The step sizes shape the representation and generalization; learning them is RL

Sutton 1992; Mahmood 2010

Non-stationary step-size problem

Target network

Output unit
(Real-valued)



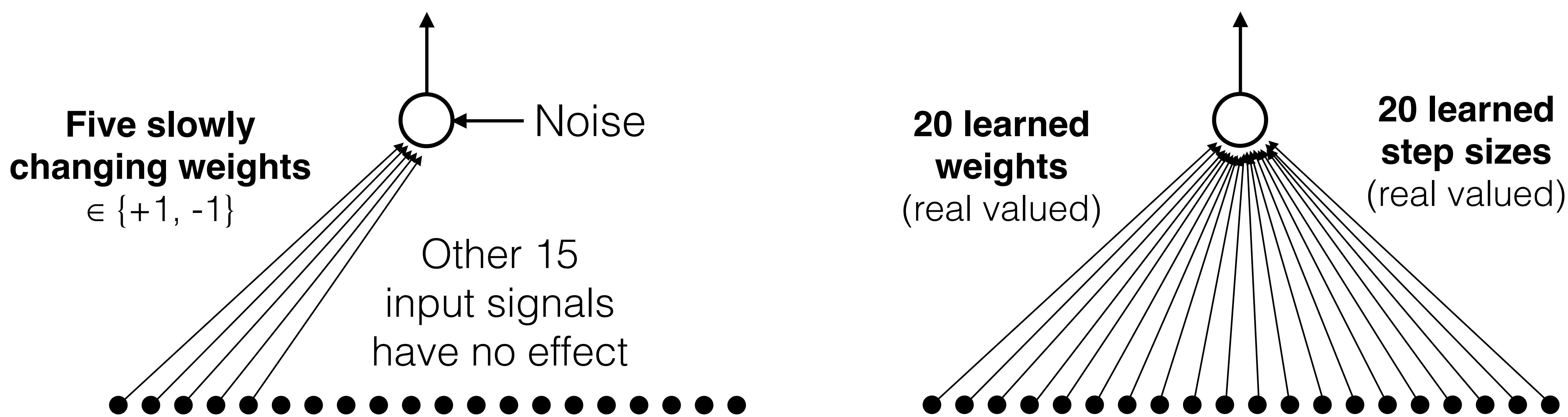
Five slowly changing weights
$\in \{+1, -1\}$

Input/feature layer
$20 \times N(0,1)$
(random-normal reals)

Sutton 1992; Mahmood 2010

Non-stationary step-size problem

Target network   Solution network

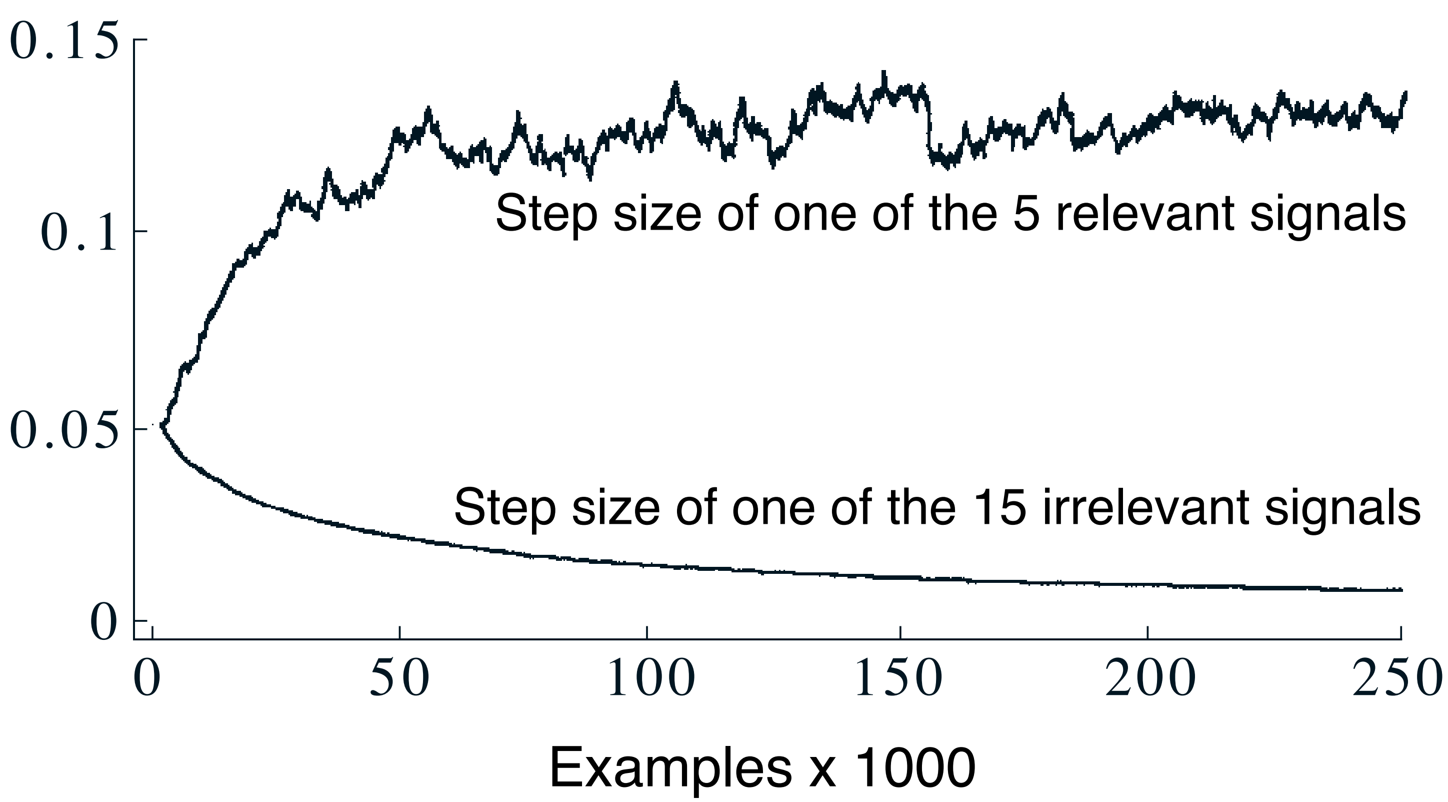


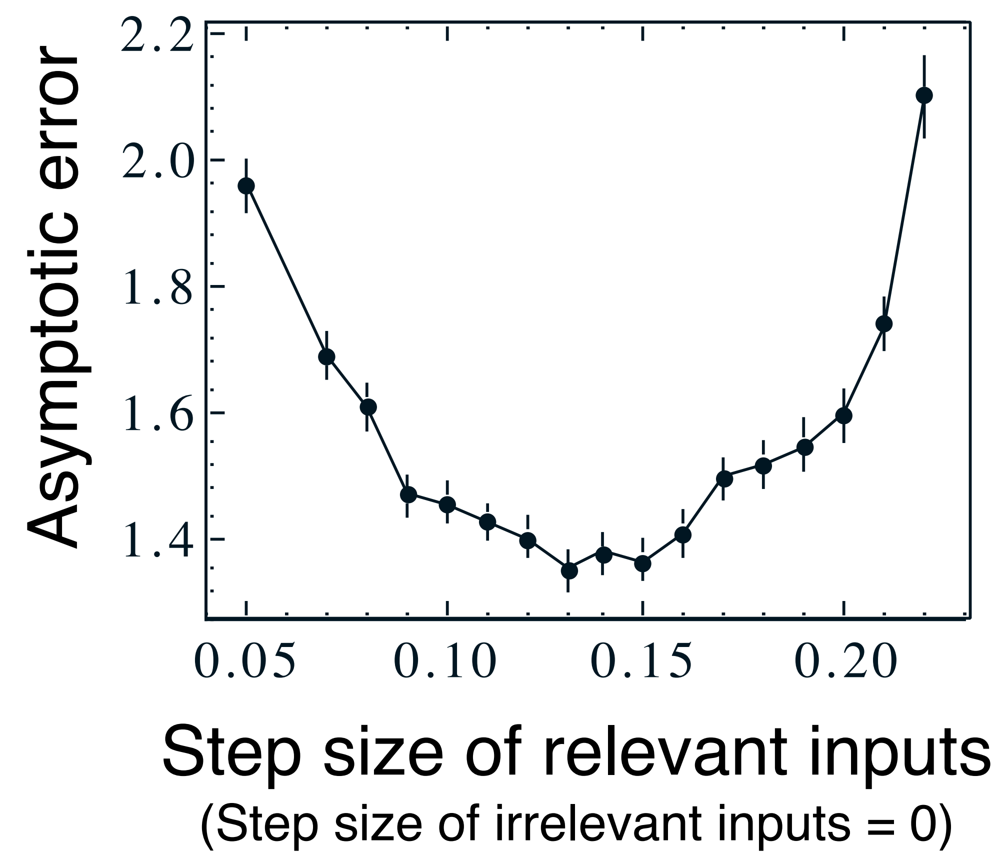
Can we find the relevant features and track their weights?
The step sizes determine the rep'n and generalization.

The step-size learning algorithm

- Incremental Delta-Bar-Delta (Sutton 1992)
  - vector step size (one for each weight)
  - meta-gradient descent:
    - $\Delta \text{Step-size}_t \propto \nabla_{\text{Step-size}} \text{Error}_t^2$
- Extended to Backprop networks by Schraudolph 1999

Problem: Non-stationary Step-size
Solution: *IDBD*





IDBD sends step sizes of irrelevant signals to ~0, and those of relevant signals to ~.13

These step-size values are near the empirically determined optimum

- IDBD slowly learns the step sizes that enable fast subsequent learning
- IDBD is true RL!

Summary

- RL should enable fast learning!
  - That was the original idea, but the field has strayed far from this goal
- Pursuing it requires online, continual learning
- The GEOFF challenge problem is generic, synthetic, online, non-stationary feature finding
  - it focuses on feature finding as an enabler of fast learning
  - and avoids many of the methodological problems
- I have presented results related to parts of this problem
- But so far the GEOFF challenge has not been squarely met