Combining Implicit & Explicit Regularization for Efficient Learning in Deep Networks

Dan Zhao

Background

- Why can deep, over-parameterized neural networks trained with gradient descent-like optimizers generalize so well?

- One explanation: implicit regularization
  - Gradient descent implicitly regularizes towards “good” solutions
  - Depth acts as an accelerative pre-conditioning during optimization

- Previous works[1] have shown how in linear networks, gradient descent implicitly regularizes towards low-rank solutions in matrix completion, whose effect becomes stronger with depth (i.e., deeper networks)

1. *“Implicit Regularization in Deep Matrix Factorization” Arora, Cohen et al. (2019)*

Key Questions

- Can we mimic the effects of implicit regularization with help from an explicit penalty (i.e., explicit regularization?)

- Do the interactions between the implicit bias of an optimizer and an explicit penalty matter?
  - Previous works focus largely on gradient descent, but it may be natural to expect that different optimizers have different inductive biases
  - Given this, different optimizers can interact differently with explicit penalties

- We try to shed light on the questions above by considering the following explicit regularizer on matrix completion tasks: $||W||_* / ||W||_F$

Key Findings

- Our proposed penalty allows a depth 1 linear network to generalize as well if not better than deeper linear networks

- However, this only takes effect when training with Adam (not gradient descent!)

- At higher depths (depth > 1), networks trained with Adam and the proposed penalty show a degree of depth invariance: all depths are now able to achieve low generalization error and recover rank perfectly

Setup

Matrix Completion

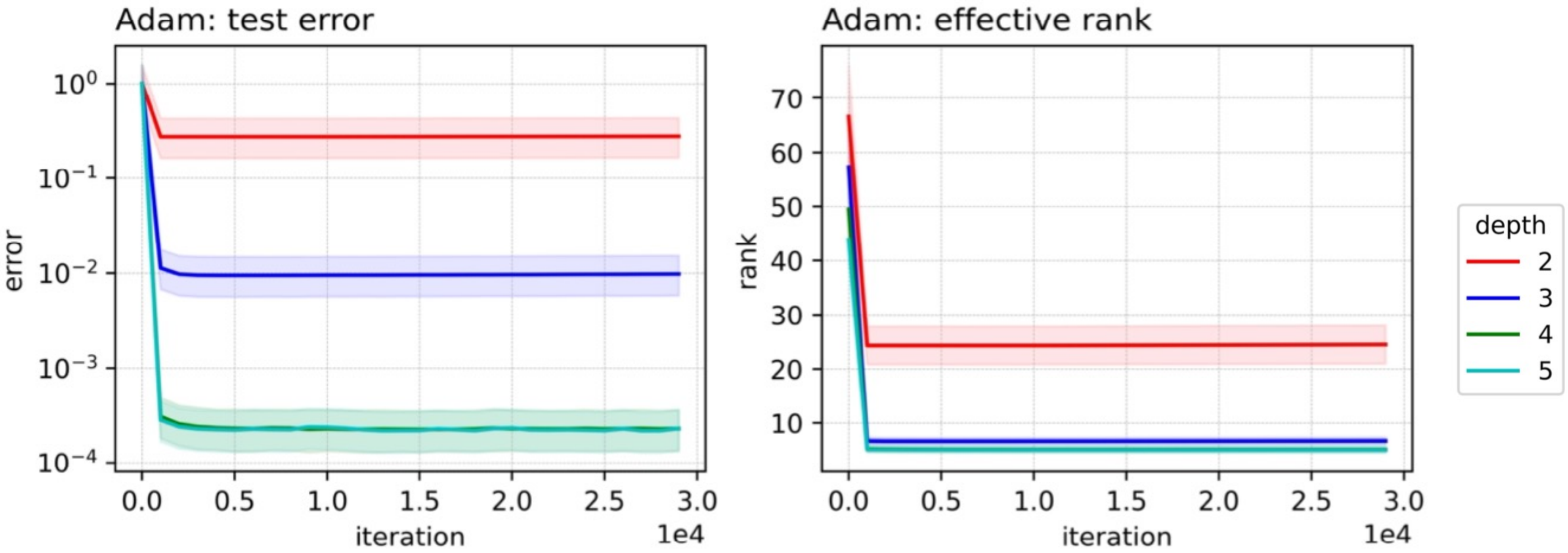
- Having observed some portion of a matrix W^* (typically low-rank), the goal is to recover the remaining entries (i.e., low test error) and/or the rank of the original matrix

Loss function: $\min_W L(W) \triangleq \min_W \|W - W^*\|^2 + \lambda\, R(W)$

- W^* is the ground-truth matrix
- $W = W_N \ldots W_1$ is the linear neural network of depth $N \geq 1$
  - $N = 1$ corresponds to a convex problem (i.e., depth 1 or no depth)
  - $N = 2$ corresponds to a shallow linear network (i.e., depth 2)
  - $N \geq 3$ corresponds to *deep* matrix factorization or a *deep* linear network (depth > 2)

- $R(W)$ is the explicit penalty or regularizer, $\lambda \geq 0$ is the regularization strength
  - In our work, our proposed penalty is a ratio of the nuclear to the Frobenius norm: $\|W\|_* / \|W\|_F$

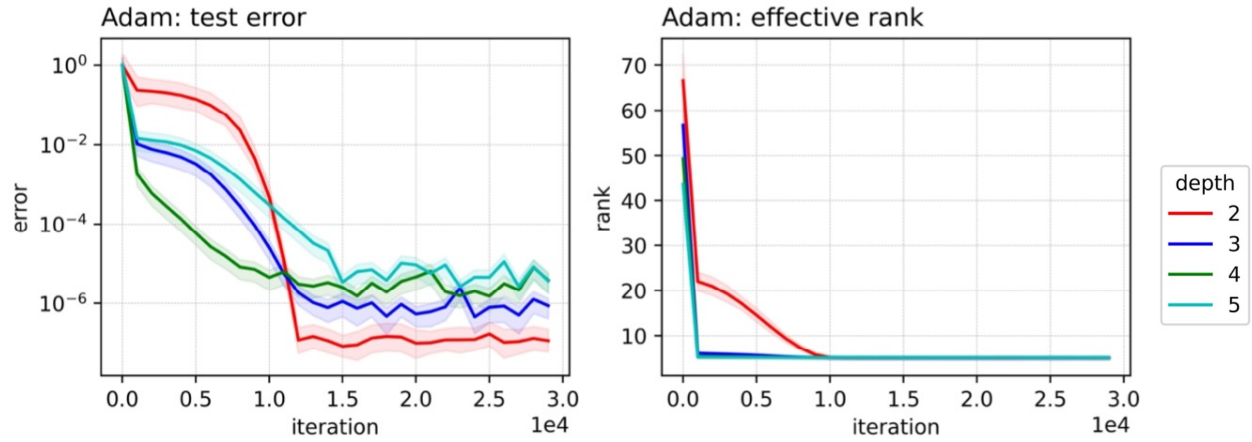
Adam



- During training, Adam requires a sufficiently deep network (above depth 3) in order to generalize well and reduce rank down to the rank of the ground-truth matrix (i.e., perfect rank recovery)

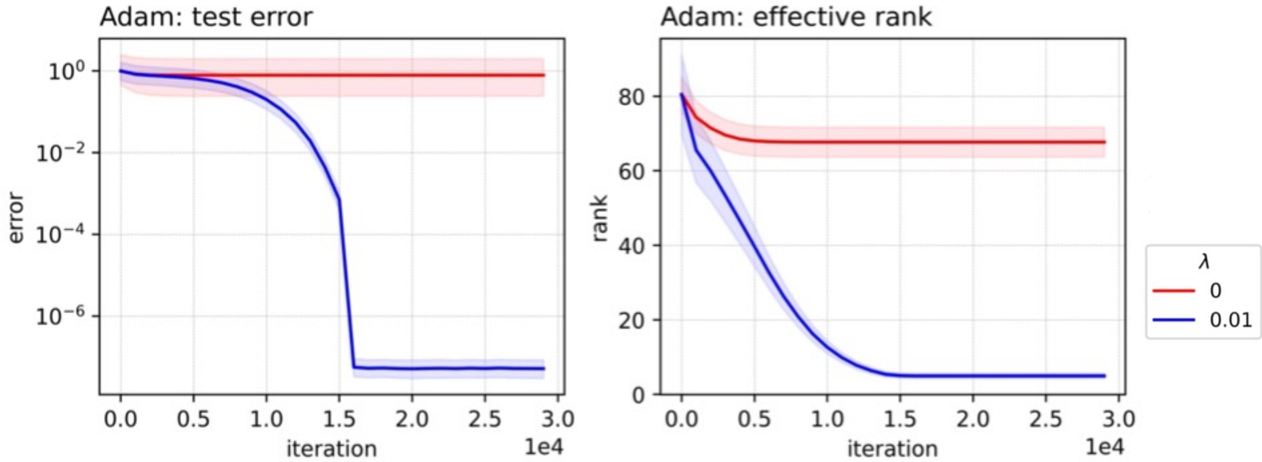
Adam + penalty

Deep Linear Network: Depths 2/3/4/5



- However, combined with our proposed penalty, Adam shows a degree of *depth invariance*: generalizing well and recovering rank at all depths…

Degenerate Network: Depth 1



- Even at depth 1!

Results (synthetic data)

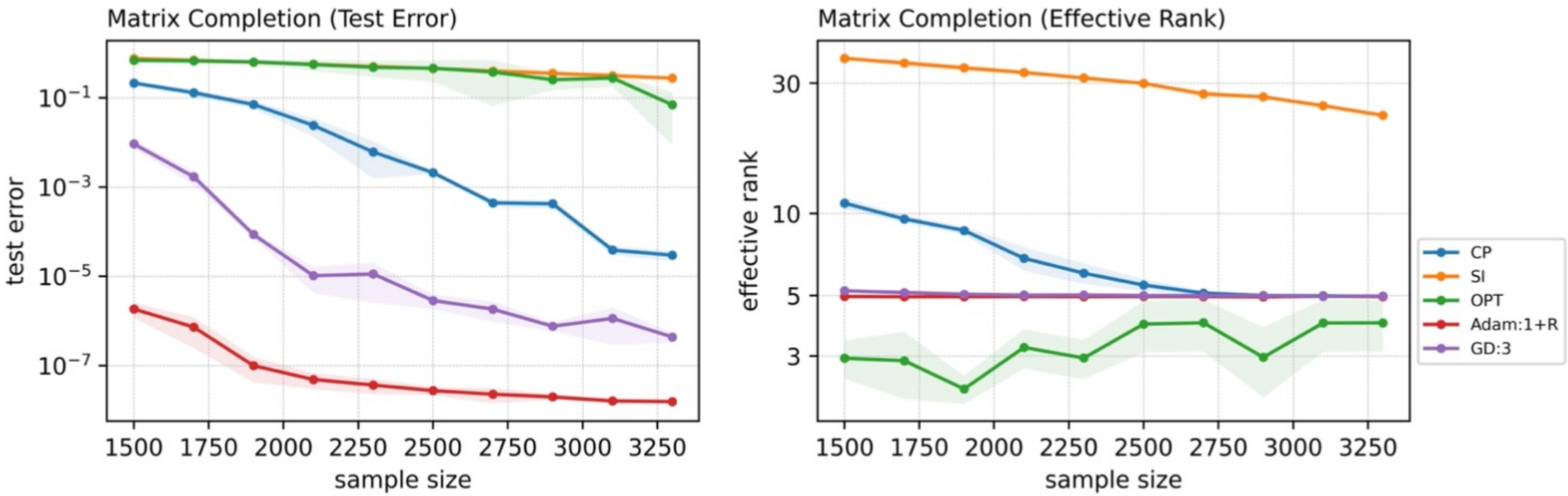


Figure 4: Comparative performance in generalization error and rank minimization for rank-5 matrix completion (100×100). x-axis stands for the number of observed entries (out of 10^5 entries) and shaded regions indicate error bands. **Adam:1+R** refers to a depth 1 network trained with Adam and our penalty, **CP** is the minimum nuclear norm solution, **GD:3** is a depth 3 network trained with gradient descent, **OPT** is `OptSpace` [38], and **SI** is `SoftImpute` [47]. To reduce clutter, we omit results with similar performance (e.g. GD:4, GD:5 etc.).

- A depth 1 network trained with Adam + penalty can outperform a variety of other methods in both generalization error and rank reduction/recovery---and also do so with less training data

Results (real-world data)

Model	Uses side info, add. features, or other info, etc?	90% RMSE
Depth 1 LNN	No	
w. GD		2.814
w. GD+penalty		2.808
w. Adam		1.844
w. Adam+penalty		**0.915**
User-Item Embedding	No	
w. GD		2.453
w. GD+penalty		2.535
w. Adam		1.282
w. Adam+penalty		**0.906**
NMF [48]	No	0.958
PMF [48]	No	0.952
SVD++ [41]	Yes	0.913
NFM [30]	No	0.910
FM [55]	No	0.909
GraphRec [53]	No	0.898
AutoSVD++ [59]	Yes	0.904
GraphRec+sidefeat.[53]	Yes	0.899
GraphRec+graph/side feat.[53]	Yes	0.883

(a) Performance on 90:10 (90%) train-test split

Model	Uses side info, add. features, or other info, etc?	80% RMSE
Depth 1 LNN	No	
w. GD		2.797
w. GD+penalty		2.821
w. Adam		1.822
w. Adam+penalty		**0.921**
User-Item Embedding	No	
w. GD		2.532
w. GD+penalty		2.519
w. Adam		1.348
w. Adam+penalty		**0.919**
IMC [33, 66]	Yes	1.653
GMC [36]	Yes	0.996
MC [18]	Yes	0.973
GRALS [52]	Yes	0.945
sRGCNN (sRMGCNN) [49]	Yes	0.929
GC-MC [16]	Yes	0.910
GC-MC+side feat. [16]	Yes	0.905

(b) Performance on 80:20 (80%) train-test split

- On MovieLens100K, a depth 1 linear network trained with Adam + penalty (in **bold**) can improve performance considerably over gradient descent alone
- Surprisingly, a depth 1 linear network with Adam + penalty can come close to or even outperform other more complex methods---without any non-linearities, side information, extra features, deep networks, etc.

Conclusion

- Takeaway: Combining Adam's own implicit bias with our proposed penalty can enable more efficient learning

- What's next?
  - Extensions/applicability to non-linear networks for other tasks?
  - Convergence rates
  - How does this fit in with other stylized facts and works?

Thank you!