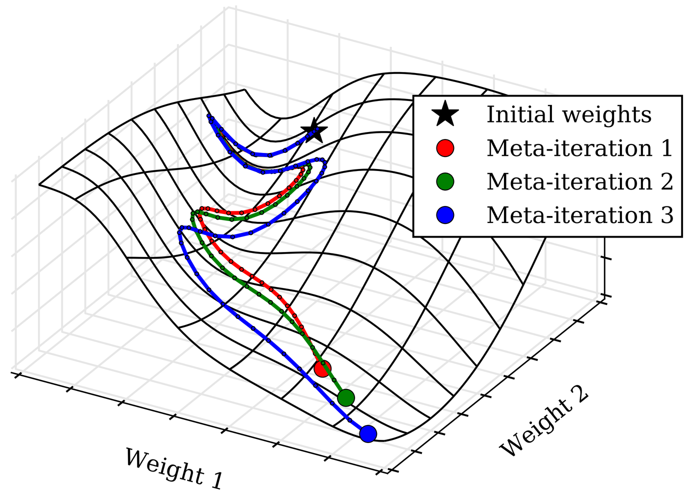
Gradient-based Hyperparameter Optimization

Dougal Maclaurin, David Duvenaud, Ryan P. Adams

Abstract

- Hyperparameters are everywhere
- Gradient-free optimization fails in high dimensions
- Validation loss is a just a function
- Why not take gradients?

Stochastic gradient descent is a function



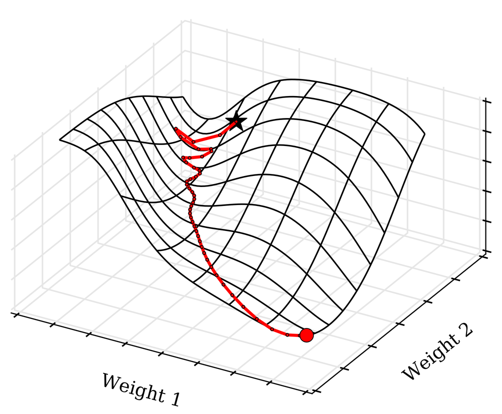
SGD (with momentum) is Reversible

Forward update rule:

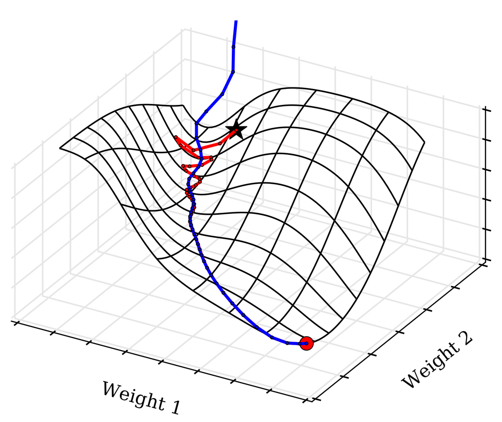
$$\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \alpha \mathbf{v}_t$$
$$\mathbf{v}_{t+1} \leftarrow \beta \mathbf{v}_t - \nabla L(\mathbf{x}_{t+1})$$

Reverse update rule:

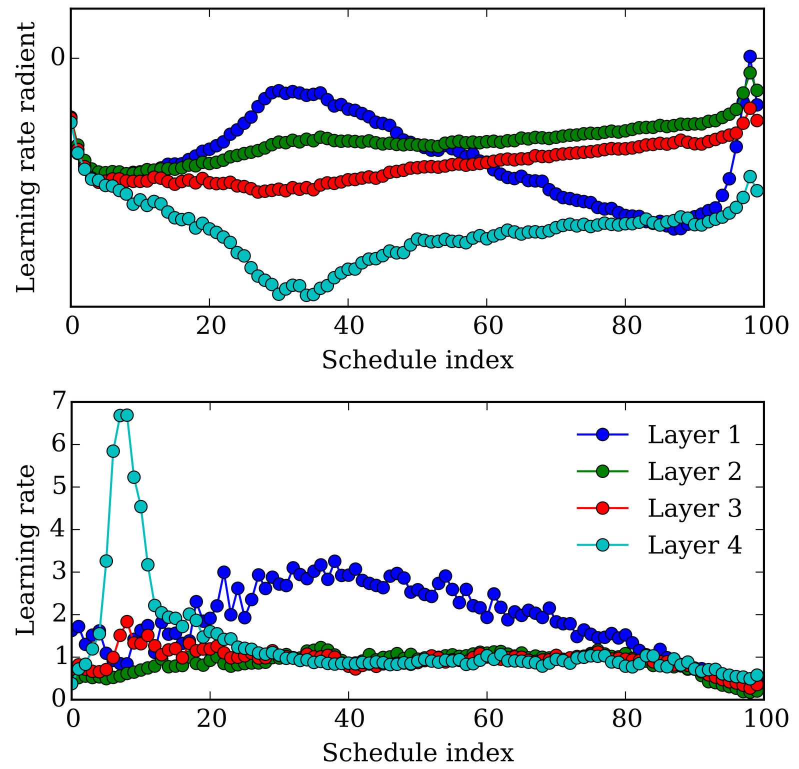
$$\mathbf{v}_t \leftarrow (\mathbf{v}_{t+1} + \nabla L(\mathbf{x}_{t+1}))/\beta$$
$$\mathbf{x}_t \leftarrow \mathbf{x}_{t+1} - \alpha \mathbf{v}_t$$



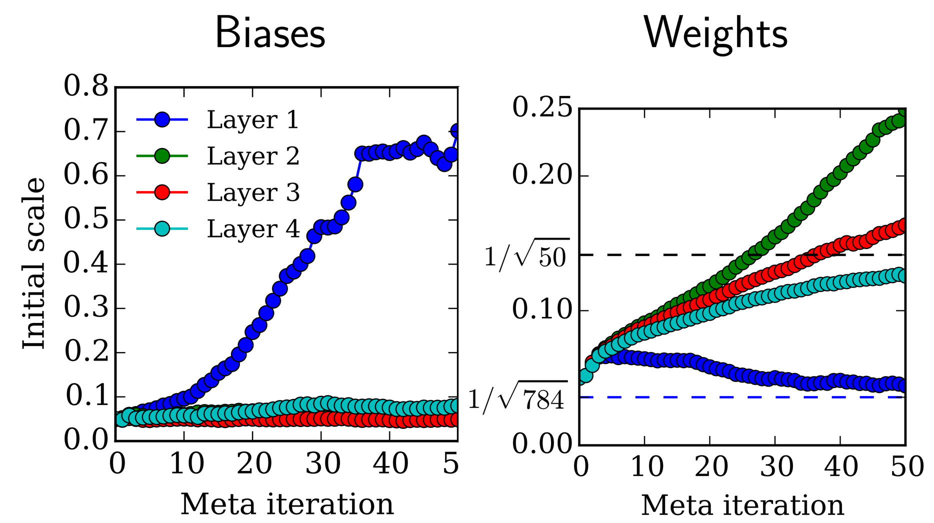
;



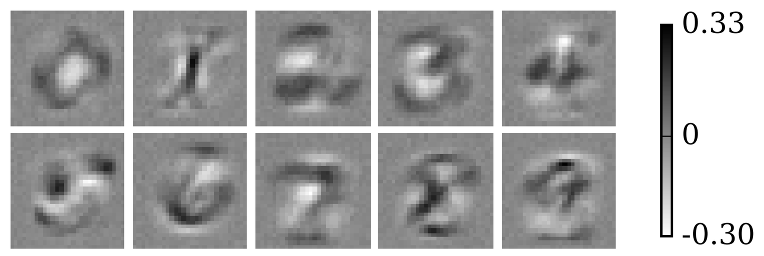
Example uses of hyper-gradients: learning rates



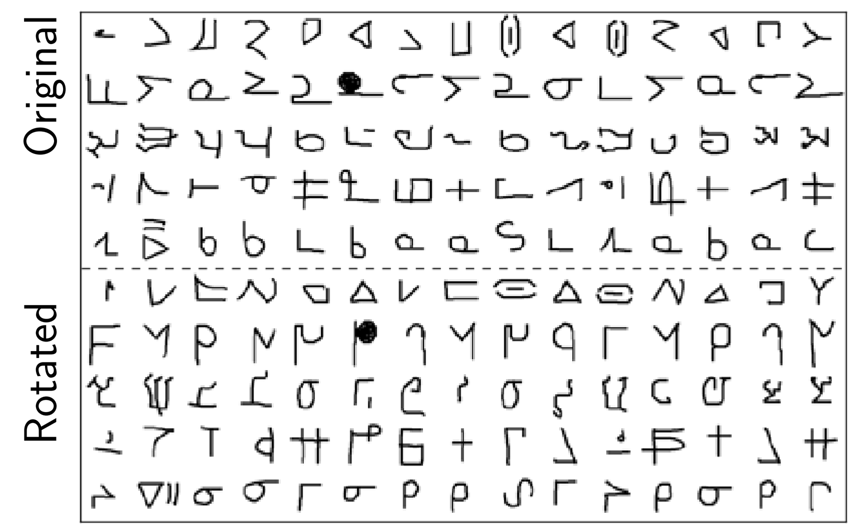
Optimizing initialization distributions



0.1 Optimizing training data

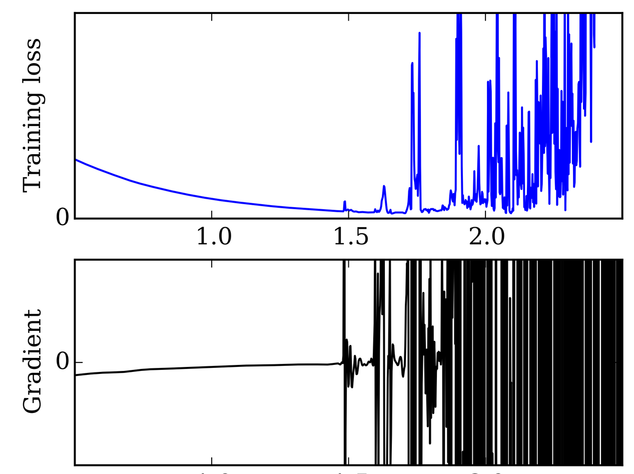


Optimizing regularization: Omniglot



	Input weights	Middle weights	Output weights	Train error	Test error
Separate networks				0.61	1.34
Tied weights				0.90	1.25
Learned sharing				0.60	**1.13**

Chaotic Learning Dynamics



Learning rate

Automatic differentiation of Numpy code

Conclusion

- We can compute gradients of learning procedures
- Can optimize thousands of hyperparameters