ALPhA Week 14 Presentation

PHY 496

BRADEN KRONHEIM

APRIL 26, 2019

Summary

- Implemented Parallel Tempering
- Investigated using Riemann manifold geometry
- Fixed log probability calculation
  - Allows better choice of priors
  - Allows training from scratch
- Started writing code to implement a better step size and leapfrog step optimizer

Summary

Network	Inside 1 SD	Inside 2 SDs	Inside 3 SD3	Percent Error
25,000 Not Tempered 5,000 burnin	33.48	58.51	74.08	11.88
5,000 Tempered 3,000 burnin	32.95	58.16	74.08	12.88
Combined 50,000 and 900	67.83	88.21	94.90	7.68
Flipout Batched PRELU	25.59	48.12	65.86	11.40

Riemann manifold algorithm

- Create a metric tensor which is the sum of the expected Fisher information matrix plus the negative Hessian of the log-prior
- This metric tensor becomes the mass matrix in the Hamiltonian
- The new Hamiltonian is not separable, so the normal leapfrog method doesn't work
- Instead, the generalized leapfrog method must be used
  - This method is implicit, meaning you must use fixed point iteration to actually solve it

$$\mathbf{p}\left(\tau+\frac{\varepsilon}{2}\right)=\mathbf{p}(\tau)-\frac{\varepsilon}{2}\nabla_{\boldsymbol{\theta}}H\left\{\boldsymbol{\theta}(\tau),\mathbf{p}\left(\tau+\frac{\varepsilon}{2}\right)\right\}, \tag{16}$$

$$\boldsymbol{\theta}(\tau+\varepsilon)=\boldsymbol{\theta}(\tau)+\frac{\varepsilon}{2}\left[\nabla_{\mathbf{p}}H\left\{\boldsymbol{\theta}(\tau),\mathbf{p}\left(\tau+\frac{\varepsilon}{2}\right)\right\}+\nabla_{\mathbf{p}}H\left\{\boldsymbol{\theta}(\tau+\varepsilon),\mathbf{p}\left(\tau+\frac{\varepsilon}{2}\right)\right\}\right], \tag{17}$$

$$\mathbf{p}(\tau+\varepsilon)=\mathbf{p}\left(\tau+\frac{\varepsilon}{2}\right)-\frac{\varepsilon}{2}\nabla_{\boldsymbol{\theta}}H\left\{\boldsymbol{\theta}(\tau+\varepsilon),\mathbf{p}\left(\tau+\frac{\varepsilon}{2}\right)\right\}. \tag{18}$$

Step size/leapfrog adaptation

- Assumes some max/min values
- Run m steps of HMC, calculate average value of expected square jumping distance divided by the square root of number of leapfrog steps
- Calculate covariance matrix of all step size/leapfrog combinations
- Calculate scale factor of a constant alpha divided by max ESJD value
- Randomly decide whether to update step size/leapfrog
  - This probability goes down over time
- Use Bayes rule to calculate a posterior for the full loss functions using the covariance matrix and the calculated ESJDs
- Calculate max of this distribution, this corresponds to new step size/leapfrog

Goals for next week

- Implement step size/leapfrog adaptation algorithm
- Try adding something to show whether the parallel tempering algorithm is switching states as it should
- Run more tests training from random initialization