CSC2541 Lecture 5
Natural Gradient

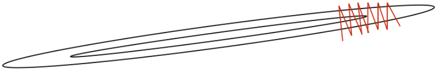
Roger Grosse

Motivation

- Two classes of optimization procedures used throughout ML
  - (stochastic) gradient descent, with momentum, and maybe coordinate-wise rescaling (e.g. Adam)
    - Can take many iterations to converge, especially if the problem is ill-conditioned
  - coordinate descent (e.g. EM)
    - Requires full-batch updates, which are expensive for large datasets
- Natural gradient is an elegant solution to both problems.
- How it fits in with this course:
  - This lecture: it's an elegant and efficient way of doing variational inference
  - Later: using probabilistic modeling to make natural gradient practical for neural nets
    - Bonus groundbreaking result: natural gradient can be interpreted as variational inference!

Motivation

- SGD bounces around in high curvature directions and makes slow progress in low curvature directions. (Note: this cartoon *understates* the problem by orders of magnitude!)



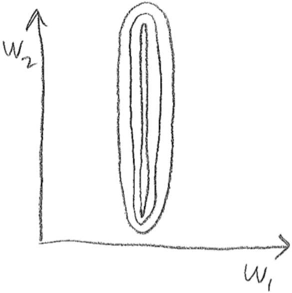
- This happens because when we train a neural net (or some other ML model), we are optimizing over a complicated manifold of functions. Mapping a manifold to a flat coordinate system distorts distances.

- Natural gradient: compute the gradient on the globe, not on the map.

Motivation: Invariances

- Suppose we have the following dataset for linear regression.

x_1	x_2	t
114.8	0.00323	5.1
338.1	0.00183	3.2
98.8	0.00279	4.1
⋮	⋮	⋮

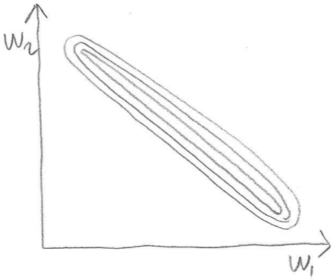


- This can happen since the inputs have arbitrary units.
- Which weight, w_1 or w_2, will receive a larger gradient descent update?
- Which one do you want to receive a larger update?
- Note: the figure vastly *understates* the narrowness of the ravine!

Motivation: Invariances

- Or maybe x_1 and x_2 correspond to years:

x_1	x_2	t
2003	2005	3.3
2001	2008	4.8
1998	2003	2.9
$\vdots$	$\vdots$	$\vdots$



Motivation: Invariances

- Consider minimizing a function $h(x)$, where x is measured in feet.
- Gradient descent update:

$$x \leftarrow x - \alpha \frac{\mathrm{d}h}{\mathrm{d}x}$$

- But $\mathrm{d}h/\mathrm{d}x$ has units 1/feet. So we're adding feet and 1/feet, which is nonsense. This is why gradient descent has problems with badly scaled data.
- Natural gradient is a dimensionally correct optimization algorithm. In fact, the updates are equivalent (to first order) in any coordinate system!

Steepest Descent

- (Rosenbrock example)
- Gradient defines a linear approximation to a function:

$$h(\mathbf{x} + \Delta\mathbf{x}) \approx h(\mathbf{x}) + \nabla h(\mathbf{x})^\top \Delta\mathbf{x}$$

- We don't trust this approximation globally. Steepest descent tries to prevent the update from moving too far, in terms of some dissimilarity measure D:

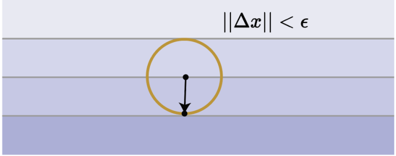
$$\mathbf{x}^{k+1} \leftarrow \arg\min_{\mathbf{x}} \left\{ \nabla h(\mathbf{x}^k)^\top (\mathbf{x} - \mathbf{x}^k) + \lambda D(\mathbf{x}, \mathbf{x}^k) \right\}$$

- Gradient descent can be seen as steepest descent with $D(\mathbf{x}, \mathbf{x}') = \frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|^2$.
  - Not a very interesting D, since it depends on the coordinate system.

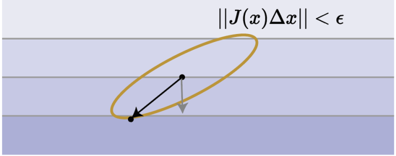
Steepest Descent

- A more interesting class of dissimilarity measures is Mahalanobis metrics:

$$D(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^\top \mathbf{A}(\mathbf{x} - \mathbf{x}')$$



Under normal "L2" distance, equidistant points form a circle and the gradient is the steepest direction.



Alternate notions of distance can make equidistant points form an elllipse and shift the steepest direction.

- Steepest descent update:

$$\mathbf{x} \leftarrow \mathbf{x} - \lambda^{-1}\mathbf{A}^{-1}\nabla h(\mathbf{x})$$

Steepest Descent

- It's hard to compute the steepest descent update for an arbitrary D. But we can approximate it with a Mahalanobis metric by taking the second-order Taylor approximation.

$$D(\mathbf{x}, \mathbf{x}') \approx \frac{1}{2}(\mathbf{x} - \mathbf{x}') \frac{\partial^2 D}{\partial \mathbf{x}^2} (\mathbf{x} - \mathbf{x}')$$

- One interesting example: simulating gradient descent on a different space.
- (Rosenbrock example)
- Later in this course, we'll use this insight to train neural nets much faster.

Fisher Metric

- If we're fitting a probabilistic model, the optimization variables parameterize a probability distribution.
- The obvious dissimilarity measure is KL divergence:

$$D(\boldsymbol{\theta}, \boldsymbol{\theta}') = \mathrm{D}_{\mathrm{KL}}(p_{\boldsymbol{\theta}} \| p_{\boldsymbol{\theta}'})$$

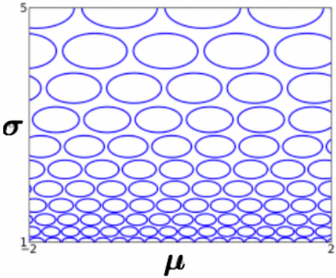
- The second-order Taylor approximation to KL divergence is the Fisher information matrix:

$$\frac{\partial^2 \mathrm{D}_{\mathrm{KL}}}{\partial \boldsymbol{\theta}^2} = \mathbf{F} = \underset{x \sim p_{\boldsymbol{\theta}}}{\mathrm{Cov}} (\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(x))$$

Fisher Metric

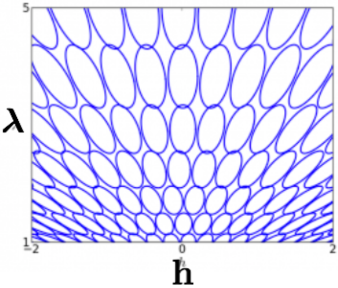
- Fisher metric for two different parameterizations of a Gaussian:

mean and std dev



$$p(x) \propto \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

information form



$$p(x) \propto \exp\left(hx - \frac{\lambda}{2}x^2\right)$$

Fisher Metric

- KL divergence is an intrinsic dissimilarity measure on distributions: it doesn't care how the distributions are parameterized.
- Therefore, steepest descent in the Fisher metric (which approximates KL divergence) is invariant to parameterization, to the first order.
  - This is why it's called natural gradient.
- Update rule:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \mathbf{F}^{-1} \nabla_{\boldsymbol{\theta}} h$$

- This can converge much faster than ordinary gradient descent.
- (example)
- Hoffman et al. found that if you're doing variational inference on conjugate exponential families, the variational inference updates are surprisingly elegant — even nicer than ordinary gradient descent!