Classification using Newton's Method

Numerical Methods for Deep Learning

Newton-like Methods

Goal: Solve $\min_{\mathbf{W}} E(\mathbf{W})$. Consider kth iteration. Assume E convex.

To find optimal step $\mathbf{D}$, use Taylor's theorem

$$E(\mathbf{W}_k+\mathbf{D}) = E(\mathbf{W}_k)+\nabla E(\mathbf{W}_k)^\top \mathbf{D}+\frac{1}{2}\mathbf{D}^\top \nabla^2 E(\mathbf{W}_k)\mathbf{D}+\mathcal{O}(\|\mathbf{D}\|^3)$$

and differentiate w.r.t $\mathbf{D}$ to obtain

$$\nabla^2 E(\mathbf{W}_k)\mathbf{D} = -\nabla E(\mathbf{W}_k).$$

Practical Newton methods (see, e.g., [1, Ch.7])

- do not compute $\mathbf{D}$ accurately (add line search for safety)
- use, e.g., Conjugate Gradient (CG) methods
- do not generate $\nabla^2 E$ since CG only needs mat-vecs
- give quadratic/superlinear/good linear convergence

Newton-like Methods for Softmax

Need to compute Hessian $\nabla^2 E$. Recall:

$$\begin{aligned}\nabla_{\mathbf{W}} E &= \frac{1}{n}\left(-\mathbf{C}_{\text{obs}} + \exp(\mathbf{S}) \odot \left(\mathbf{e}_{n_c}\left(\frac{1}{\mathbf{e}_{n_c}^\top \exp(\mathbf{S})}\right)\right)\right)\mathbf{Y}^\top \\ &= \nabla_{\mathbf{S}} E(\mathbf{S})\mathbf{Y}^\top,\end{aligned}$$

where $\mathbf{S} = \mathbf{W}\mathbf{Y}$. For Hessian we know

$$\nabla^2_{\mathbf{W}} E(\mathbf{W}) = \mathbf{Y}\nabla^2_S E(\mathbf{S})\mathbf{Y}^\top$$

Remarks:

- size of $\nabla^2_{\mathbf{S}} E$ is $n_c n \times n_c n$, typically sparse
- size of $\nabla^2_{\mathbf{W}} E$ is $n_c n_f \times n_c n_f$, typically dense
- building Hessian can be costly (when n is large)
- Hessian is spd since E is convex in $\mathbf{S}$

Hessian of Softmax Function - 1

Recall

$$\nabla_{\mathbf{S}} E = \frac{1}{n}\left(-\mathbf{C} + \exp(\mathbf{S}) \odot \frac{1}{\mathbf{e}_{n_c}\mathbf{e}_{n_c}^\top \exp(\mathbf{S})}\right)$$

Let's first vectorize this $\mathbf{s} = \mathrm{vec}(\mathbf{S})$ and $\mathbf{c} = \mathrm{vec}(\mathbf{C})$

$$\nabla_{\mathbf{s}} E = \frac{1}{n}\left(-\mathbf{c} + \exp(\mathbf{s}) \odot \frac{1}{(\mathbf{I} \otimes (\mathbf{e}_{n_c}\mathbf{e}_{n_c}^\top)) \exp(\mathbf{s})}\right)$$

Use product rule

$$\begin{aligned}
\nabla_{\mathbf{s}}^2 E &= \mathrm{diag}\left(\frac{1}{(\mathbf{I} \otimes (\mathbf{e}_{n_c}\mathbf{e}_{n_c}^\top)) \exp(\mathbf{s})}\right)\mathbf{J}_{\mathbf{s}} \exp(\mathbf{s}) + \\
&\quad \mathrm{diag}(\exp(\mathbf{s}))\mathbf{J}_{\mathbf{s}}\left(\frac{1}{(\mathbf{I} \otimes (\mathbf{e}_{n_c}\mathbf{e}_{n_c}^\top)) \exp(\mathbf{s})}\right) \\
&= \nabla_{\mathbf{s}}^2 E_1 + \nabla_{\mathbf{s}}^2 E_2
\end{aligned}$$

Hessian of Softmax Function - 2

First term is easy

$$\begin{aligned}\nabla^2_{\mathbf{s}} E_1 &= \operatorname{diag}\left(\frac{1}{(\mathbf{I}\otimes(\mathbf{e}_{n_c}\mathbf{e}_{n_c}^\top))\exp(\mathbf{s})}\right)\operatorname{diag}(\exp(\mathbf{s})) \\ &= \operatorname{diag}\left(\frac{\exp(\mathbf{s})}{(\mathbf{I}\otimes(\mathbf{e}_{n_c}\mathbf{e}_{n_c}^\top))\exp(\mathbf{s})}\right)\end{aligned}$$

Reshaped back, a matrix-vector-product with $\mathbf{V} \in \mathbb{R}^{n_c \times n_f}$ is

$$\mathbf{H}_1\mathbf{V} = \left(\left(\frac{\exp(\mathbf{S})}{\mathbf{e}_{n_c}\mathbf{e}_{n_c}^\top \exp(\mathbf{S})}\right)\odot(\mathbf{VY})\right)\mathbf{Y}^\top$$

Hessian of Softmax Function - 3

$$E_2 = \mathrm{diag}(\exp(\mathbf{s}))\, \underbrace{\mathbf{J}_\mathbf{s}\left(\frac{1}{(\mathbf{I}\otimes(\mathbf{e}_{n_c}\mathbf{e}_{n_c}^\top))\exp(\mathbf{s})}\right)}_{=:\mathbf{T}}.$$

Using chain rule, we get

$$\mathbf{T} = -\mathrm{diag}\left(\frac{1}{((\mathbf{I}\otimes(\mathbf{e}_{n_c}\mathbf{e}_{n_c}^\top))\exp(\mathbf{s}))^2}\right)(\mathbf{I}\otimes(\mathbf{e}_{n_c}\mathbf{e}_{n_c}^\top))\mathrm{diag}(\exp(s))$$

After reshape the matrix-vector-product with $\mathbf{V} \in \mathbb{R}^{n_f \times n_c}$ is

$$\mathbf{H}_2\mathbf{V} = -\left(\frac{(\exp(\mathbf{S}))}{\mathbf{e}_{n_c}(\mathbf{e}_{n_c}^\top \exp(\mathbf{S}))^2}\right)\odot(\mathbf{e}_{n_c}\mathbf{e}_{n_c}^\top(\exp(\mathbf{S})\odot(\mathbf{VY})))\mathbf{Y}^\top$$

Newton-CG for Softmax function

Mat-vecs with Hessian can be computed as

$$\begin{aligned}\nabla^2_{\mathbf{W}} E(\mathbf{W})\mathbf{V} = & \frac{1}{n}\left(\left(\frac{\exp(\mathbf{S})}{\mathbf{e}_{n_c}\mathbf{e}_{n_c}^\top \exp(\mathbf{S})}\right) \odot (\mathbf{VY})\right)\mathbf{Y}^\top \\ & - \frac{1}{n}\left(\frac{(\exp(\mathbf{S}))}{\mathbf{e}_{n_c}(\mathbf{e}_{n_c}^\top \exp(\mathbf{S}))^2}\right) \odot (\mathbf{e}_{n_c}\mathbf{e}_{n_c}^\top(\exp(\mathbf{S}) \odot (\mathbf{VY})))\mathbf{Y}^\top\end{aligned}$$

(possible to further simplify this to reduce operations)

Now, ready to use matrix-free Newton method with Armijo linesearch and CG solver that computes

$$\nabla^2_{\mathbf{W}} E(\mathbf{W})\mathbf{D} \approx -\nabla_{\mathbf{W}} E(\mathbf{W}).$$

Remarks:

- how well to solve? use large tolerance on relative residual
- can accelerate CG with preconditioning $\rightsquigarrow$ PCG
- possible to omit second term in Hessian?

Coding: Hessian of Softmax Function

Extend your softmax function, so that it returns a function handle computing mat-vecs with Hessian if needed.

```
function[E,dE,d2Emv] = softmaxFun(W,Y,C)

% Your code from before
if nargout > 1
% Your code from before
end

if nargout > 2

% Your new code here
d2Emv = @(V) ...;
end
end
```

Don't forget to check your derivatives!

Newton-like Methods - Derivatives

Consider the softmax function

$$E(\mathbf{W}) = -\sum \mathbf{Y} \odot (\mathbf{XW}) + \sum \log\left(\sum \exp(\mathbf{XW})\right)$$

Class problems:

1. Compute the Hessian of the cross entropy function
2. Write code that constructs the matrix it and do a derivative check at a random point $\mathbf{W}_0$.
3. Write a code that performs matrix vector products with the Hessian (without constructing it). Test by comparing results with the matrix-based code for a random vector.

References

[1] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer Science & Business Media, New York, Dec. 2006.