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 $\mathbf{D}$

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 $\mathbf{D}$ and obtain

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

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 $\mathbf{D}$ and obtain

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

Practical Newton methods

- 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:

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

where $\mathbf{S} = \mathbf{YW}$.

Newton-like Methods for Softmax

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

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

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

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

Newton-like Methods for Softmax

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

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

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

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

Remarks:

- size of $\nabla^2_{\mathbf{S}} E$ is $n \times n$, 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

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

As before, let's first vectorize this $\mathbf{s} = \mathrm{vec}(\mathbf{S})$ and $\mathbf{c} = \mathrm{vec}(\mathbf{C})$

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

Hessian of Softmax Function - 1

Recall

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

As before, let's first vectorize this $\mathbf{s} = \mathrm{vec}(\mathbf{S})$ and $\mathbf{c} = \mathrm{vec}(\mathbf{C})$

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

Use product rule

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

Hessian of Softmax Function - 2

First term easy

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

Need only mat-vec

$$\mathbf{HV} \approx \mathbf{Y}^\top \left(\left(\frac{\exp(\mathbf{S})}{\exp(\mathbf{S})\mathbf{e}}\right) \odot (\mathbf{YV})\right)$$

Newton for softMax function

Second term mat-vec

$$\nabla^2 E_2 = -(\mathbf{Y}^\top(\exp(\mathbf{S}) \odot \left(\frac{1}{(\exp(\mathbf{S})\mathbf{e})^2}\right) \odot (\exp(\mathbf{S}) \odot ((\mathbf{YV})\mathbf{e}))$$

A little bit longer do derive

May not want to use the second term in Newton

Newton for softMax function

Use the mat-vec in Newton-CG algorithm

Newton for softMax function

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 second derivatives of the cross entropy function code it and and check your code.
2. Compute the second derivatives of the cross entropy function times a vector, code it and and check your code.

Newton-like Methods

Class problem

- Code Newton's method
- Test it on a simple problem

Coding: Newton for Classification

Outline

- data: take 7 and 1 in MNIST
- logregression
- Hessian as function
- CG