Lecture Notes for

Machine Learning in Python

⟦👨‍🏫, 👨‍💻, 🐍, 👨‍🔬⟧

Professor Eric Larson

Optimization Techniques for Logistic Regression Continued

Class Logistics and Agenda

- Agenda
  - Numerical Optimization Techniques
    - Types of Optimization
    - Programming the Optimization
- **Last Time:**
  - Logistic regression update equations
  - Line Searches
  - Stochastic small batches
  - Hessian-based methods

Class Overview, by topic

Table Data Visualization
Numpy, Pandas, Seaborn
Overviews with some in-depth discussion

Dimension Reduction and Image Processing
Scikit-learn, Scikit Image,
Intuition only, Some mathematics

Linear and Logistic Regression
Numpy, Recreate API for Scikit-learn
Detailed mathematics for simple optimization
intuition for advanced optimization

Neural Networks and Back Prop.
Numpy
Detailed mathematics for NN operations

Wide and Deep Networks | **Convolutional Networks** | **Recurrent Networks**
Keras, Tensorflow
Intuition, Detailed implement.

Ethics in Language Models
ConceptNet
Case studies

Demo Lecture, Continued

$$\mathbf{H}_{j,k}(\mathbf{w}) = \frac{\partial}{\partial w_k}\frac{\partial}{\partial w_j} l(\mathbf{w}) \qquad \frac{\partial}{\partial w_j} l(\mathbf{w}) = \sum_i \left(y^{(i)} - g(\mathbf{w}^T \cdot \mathbf{x}^{(i)})\right) x_j^{(i)}$$

$$\mathbf{H}_{j,k}(\mathbf{w}) = \frac{\partial}{\partial w_k} \sum_i \left(y^{(i)} - g(\mathbf{w}^T \cdot \mathbf{x}^{(i)})\right) x_j^{(i)}$$

$$= \cancel{\sum_i \frac{\partial}{\partial w_k} y^{(i)}} x_j^{(i)} - \sum_i \frac{\partial}{\partial w_k} g(\mathbf{w}^T \cdot \mathbf{x}^{(i)}) x_j^{(i)}$$

no dependence on w_k, zero

$$= - \sum_i x_j^{(i)} \boxed{\frac{\partial}{\partial w_k} g(\mathbf{w}^T \cdot \mathbf{x}^{(i)})}$$

already know this as $g(1-g)x_k$

$$\mathbf{H}_{j,k}(\mathbf{w}) = - \sum_{i=1}^{M} \left[g(\mathbf{w}^T \mathbf{x}^{(i)})[1 - g(\mathbf{w}^T \mathbf{x}^{(i)})] \right] \cdot x_k^{(i)} x_j^{(i)}$$

for each j,k pair

$$L_2 = C \sum_j w_j^2$$

`penalty = 'l2'`

$$L_1 = C \sum_j |w_j|$$

`penalty = 'l1'`

$$L_{12} = C_1 \sum_j |w_j| + C_2 \sum_j w_j^2$$

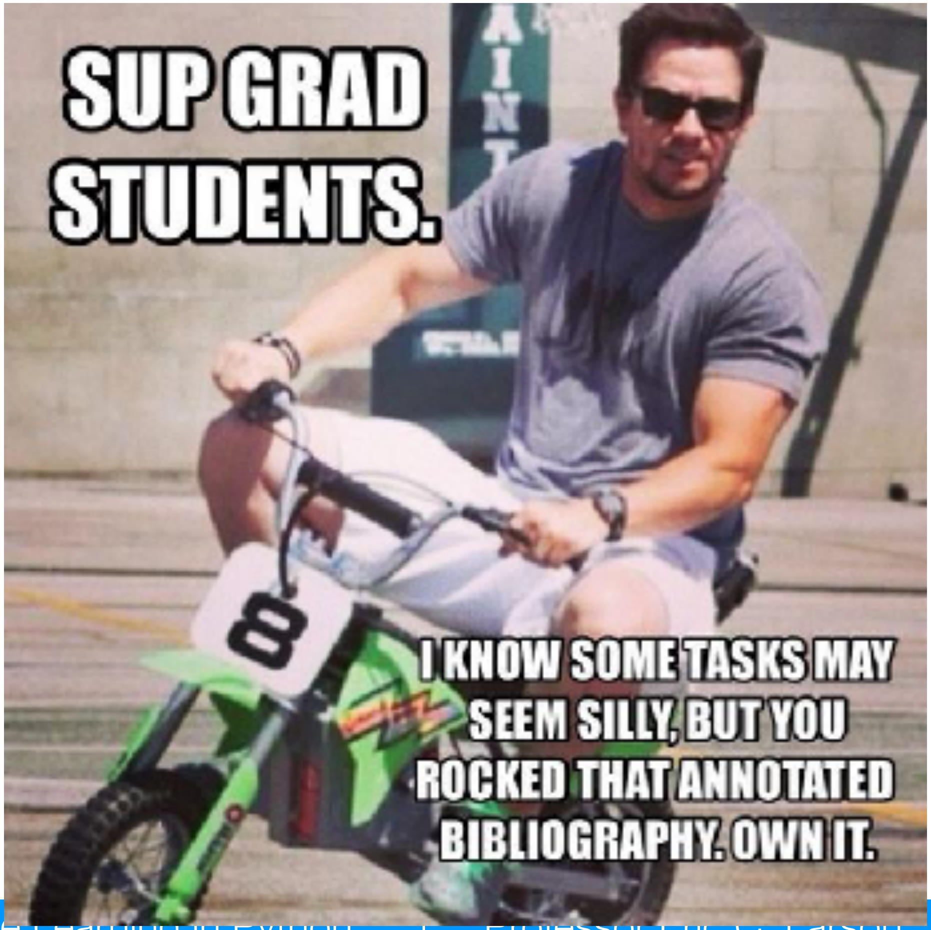
`penalty = 'elasticnet'`

Warning: The choice of the algorithm depends on the penalty chosen. Supported penalties by solver:

- 'lbfgs' - ['l2', None]
- 'liblinear' - ['l1', 'l2']
- 'newton-cg' - ['l2', None]
- 'newton-cholesky' - ['l2', None]
- 'sag' - ['l2', None]
- 'saga' - ['elasticnet', 'l1', 'l2', None]

Scratch Paper

Back Up Slides

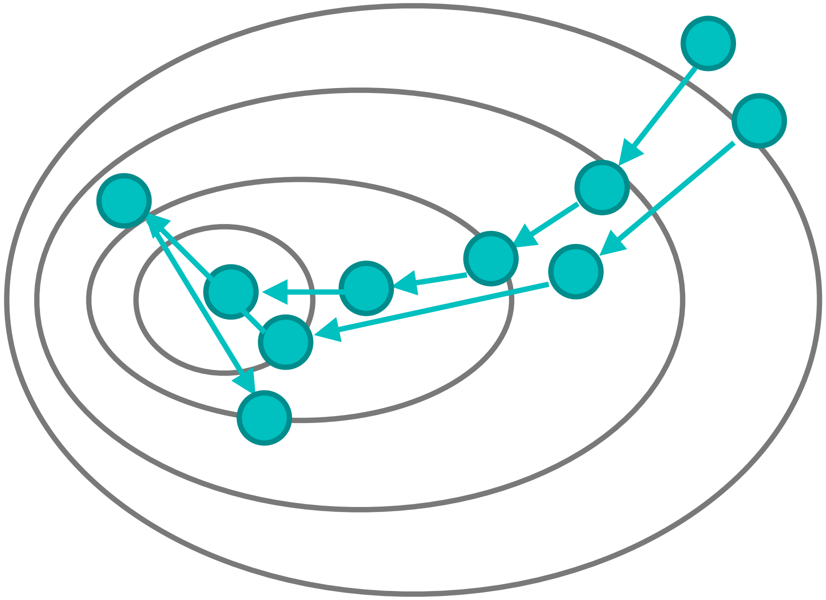

SUP GRAD STUDENTS.
I KNOW SOME TASKS MAY SEEM SILLY, BUT YOU ROCKED THAT ANNOTATED BIBLIOGRAPHY. OWN IT.

Optimization: gradient descent

- What we know thus far:

$$\underbrace{w_j}_{\text{new value}} \leftarrow \underbrace{w_j}_{\text{old value}} + \eta \underbrace{\left[\left(\sum_{i=1}^{M} (y^{(i)} - g(x^{(i)}))x_j^{(i)} \right) - C \cdot 2w_j \right]}_{\nabla l(w)}$$

$$w \leftarrow w + \eta \nabla l(w)$$

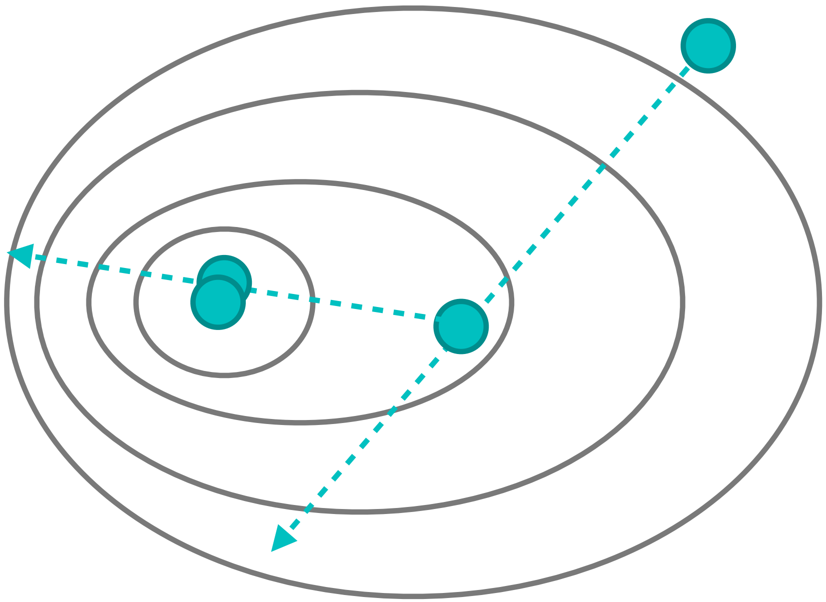


Line Search: a better method

- Line search in direction of gradient:

$$w \leftarrow w + \eta \nabla l(w)$$

$$w \leftarrow w + \underbrace{\eta}_{\text{best step?}} \nabla l(w)$$



Revisiting the Gradient

- How much computation is required (for gradient)?

$$\sum_{i=1}^{M}(y^{(i)} - \hat{y}^{(i)})x^{(i)} - 2C \cdot w$$

M = number of instances
N = number of features

Self Test: How many multiplies per gradient calculation?

A. M*N+1 multiplications
B. (M+1)*N multiplications
C. 2N multiplications
D. 2N-M multiplications

Stochastic Methods

- How much computation is required (for gradient)?

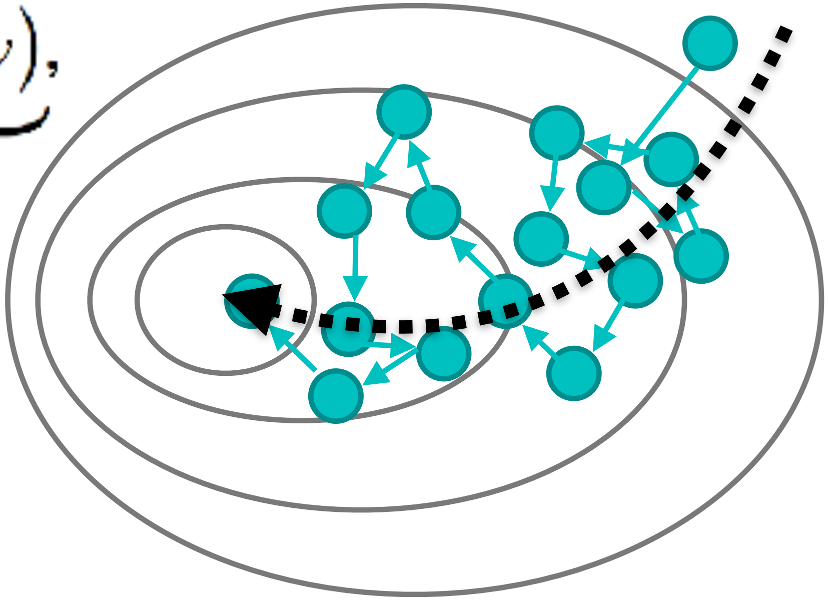
$$\sum_{i=1}^{M}(y^{(i)} - \hat{y}^{(i)})x^{(i)} - 2C \cdot w$$

Per iteration:

(M+1)*N multiplications

2M add/subtract

$$w \leftarrow w + \eta\underbrace{\left((y^{(i)} - \hat{y}^{(i)})x^{(i)} - 2C \cdot w\right)}_{\text{approx. gradient}},$$



i chosen at random

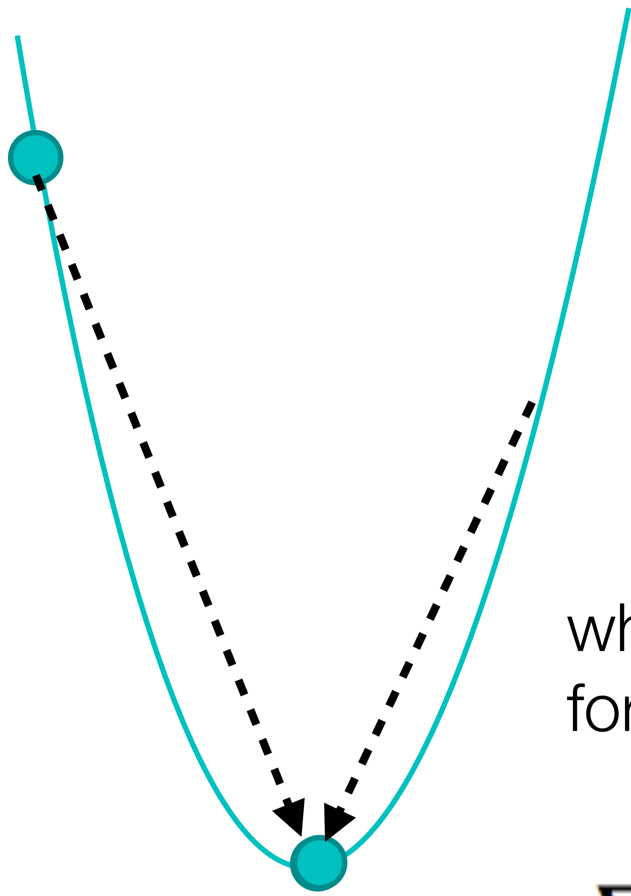
Per iteration:

N+1 multiplications

1 add/subtract

Can we do better than the gradient?

- Assume function is quadratic:



function of one variable:

$$w \leftarrow w - \underbrace{[\frac{\partial^2}{\partial w} l(w)]^{-1}}_{\text{inverse 2nd deriv}} \underbrace{\frac{\partial}{\partial w} l(w)}_{\text{derivative}}$$

will solve in one step!

what is the second order derivative for a multivariate function?

$$\nabla^2 l(w) = \mathbf{H}[l(w)]$$

The Hessian

- Assume function is quadratic:

function of one variable:

$$\mathbf{H}[l(w)] = \begin{bmatrix} \frac{\partial^2}{\partial w_1} l(w) & \frac{\partial}{\partial w_1}\frac{\partial}{\partial w_2} l(w) & \dots & \frac{\partial}{\partial w_1}\frac{\partial}{\partial w_N} l(w) \\ \frac{\partial}{\partial w_2}\frac{\partial}{\partial w_1} l(w) & \frac{\partial^2}{\partial w_2} l(w) & \dots & \frac{\partial}{\partial w_2}\frac{\partial}{\partial w_N} l(w) \\ & \vdots & & \\ \frac{\partial}{\partial w_N}\frac{\partial}{\partial w_1} l(w) & \frac{\partial}{\partial w_N}\frac{\partial}{\partial w_2} l(w) & \dots & \frac{\partial^2}{\partial w_N} l(w) \end{bmatrix}$$

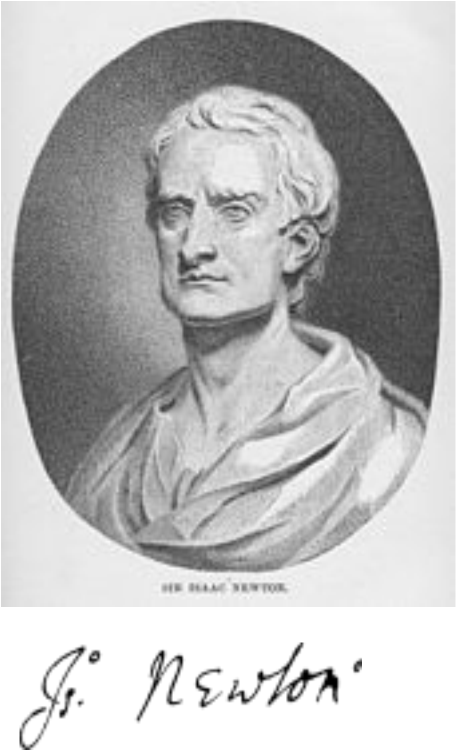
$$\nabla^2 l(w) = \mathbf{H}[l(w)]$$

The Newton Update Method

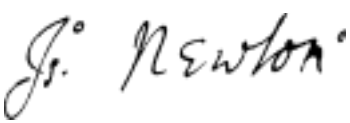
- Assume function is quadratic (in high dimensions):

$$w \leftarrow w - \underbrace{[\frac{\partial^2}{\partial w} l(w)]^{-1}}_{\text{inverse 2nd deriv}} \underbrace{\frac{\partial}{\partial w} l(w)}_{\text{derivative}}$$

$$w \leftarrow w + \eta \cdot \underbrace{\mathbf{H}[l(w)]^{-1}}_{\text{inverse Hessian}} \cdot \underbrace{\nabla l(w)}_{\text{gradient}}$$



I do not know what I may appear to the world, but to myself I seem to have been only like a boy playing on the sea-shore, and diverting myself in now and then finding a smoother pebble or a prettier shell than ordinary, whilst the great ocean of truth lay all undiscovered before me.



I do not know what I may appear to the world, but to myself I seem to have been only like a boy playing on the sea-shore, and diverting myself in now and then finding a smoother pebble or a prettier shell than ordinary, whilst the great ocean of truth lay all undiscovered before me.

$$\frac{\partial}{\partial w_j} l(w) = \sum_i \left(y^{(i)} - g(x^{(i)})\right) x_j^{(i)}$$

PLUG IN

$$H[k,j] = \frac{\partial}{\partial w_k} \frac{\partial}{\partial w_j} l = \frac{\partial}{\partial w_k} \left(\sum_i \left(y^{(i)} - g(x^{(i)})\right) x_j^{(i)} \right)$$

$$= \sum_i \frac{\partial}{\partial w_k} \overset{\nearrow 0}{y^{(i)}} x_j^{(i)} - \frac{\partial}{\partial w_k} g(x^{(i)}) x_j^{(i)}$$

LEFT OVER TERM

$$= - \sum_i \frac{\partial}{\partial w_k} g(x^{(i)}) x_j^{(i)}$$

Already know $\frac{\partial}{\partial w_k} g(w^T x^{(i)})$ side calculation

$$= g(x^{(i)})\left(1 - g(x^{(i)})\right) \frac{\partial}{\partial w_k}\left(w^T x^{(i)}\right)$$

$$= g(x^{(i)})\left(1 - g(x^{(i)})\right) x_k^{(i)}$$

PLUG IN

$$\therefore = - \sum_i g(x^{(i)})\left(1 - g(x^{(i)})\right) x_k^{(i)} x_j^{(i)}$$

THAT'S THE HESSIAN!

$H(k,j) =$ This is a valid equation for the Hessian, but we want to represent it using linear algebra

$$\boxed{= - \sum_i \underset{\substack{\uparrow \\ \text{SIGMOID}}}{g}(x^{(i)})\left(1 - g(x^{(i)})\right) x_k^{(i)} x_j^{(i)}}$$

... LINEAR ALG...

The Hessian for Logistic Regression

- The hessian is easy to calculate from the gradient for logistic regression

$$w \leftarrow w + \eta \cdot \underbrace{\mathbf{H}[l(w)]^{-1}}_{\text{inverse Hessian}} \cdot \underbrace{\nabla l(w)}_{\text{gradient}}$$

$$\mathbf{H}_{j,k}[l(w)] = -\sum_{i=1}^{M} g(x^{(i)})(1 - g(x^{(i)})x_k{}^{(i)}x_j{}^{(i)}$$

$$\underbrace{\sum_{i=1}^{M}(y^{(i)} - \hat{y}^{(i)})x_j^{(i)}}_{\text{gradient}}$$

$$\mathbf{H}[l(w)] = X^T \cdot \text{diag}[g(x^{(i)})(1 - g(x^{(i)}))] \cdot X$$

$$X * y_{diff}$$

$$w \leftarrow w + \eta[X^T \cdot \text{diag}[g(x^{(i)})(1 - g(x^{(i)}))] \cdot X]^{-1} \cdot X * y_{diff}$$

Newton’s method

Demo

BFGS (if time)
parallelization