Lecture Notes for
Machine Learning in Python

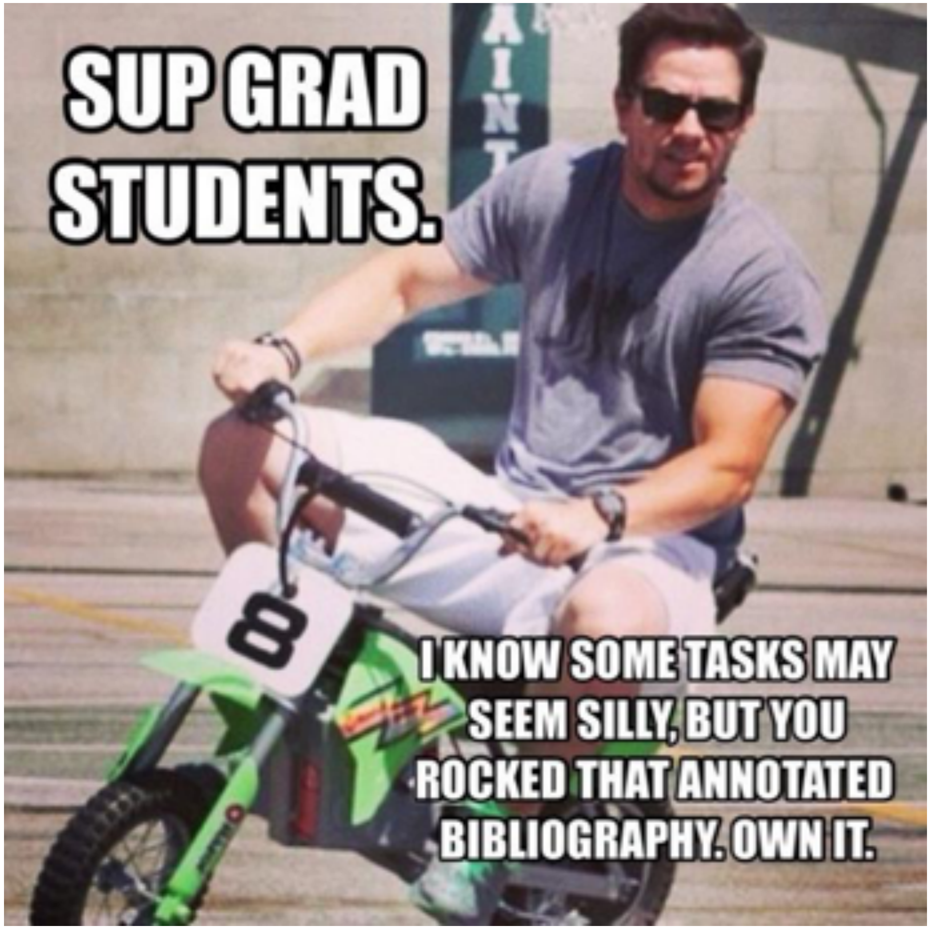
Professor Eric Larson

Week Five, Lecture A

Class Logistics and Agenda

- Grades are coming…
- Agenda
  - Numerical Optimization Techniques
    - Types of Optimization
    - Programming the Optimization
- **Whirlwind Lecture Alert**: entire classes cover these concepts
  - We only want an intuition!

Gradient Descent Techniques



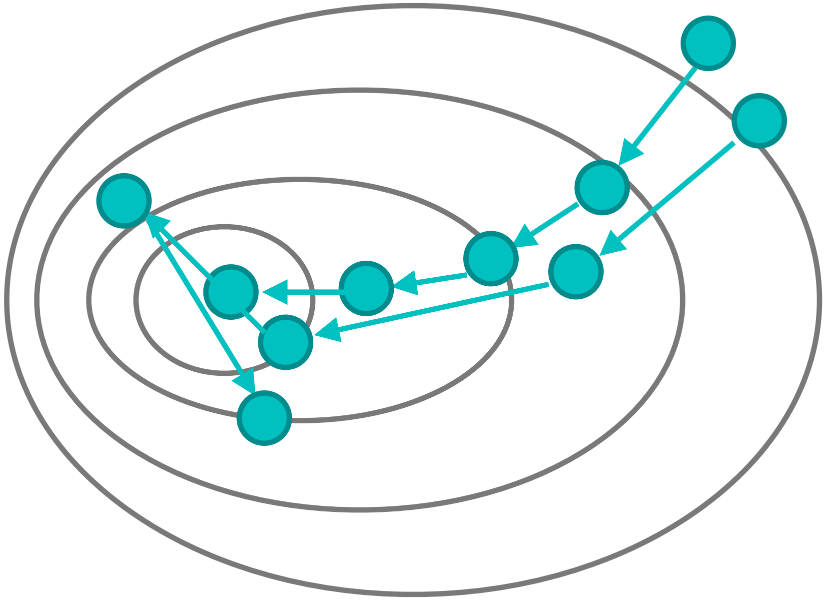
Optimization: gradient descent

- What we know thus far:

$$\underbrace{w_j}_{\text{new value}} \leftarrow \underbrace{w_j}_{\text{old value}} + \underbrace{\eta \sum_{i=1}^{M} (y^{(i)} - \hat{y}^{(i)}) x_j^{(i)}}_{\text{gradient}}$$

$$w \leftarrow w + \eta \sum_{i=1}^{M} (y^{(i)} - \hat{y}^{(i)}) x^{(i)}$$

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



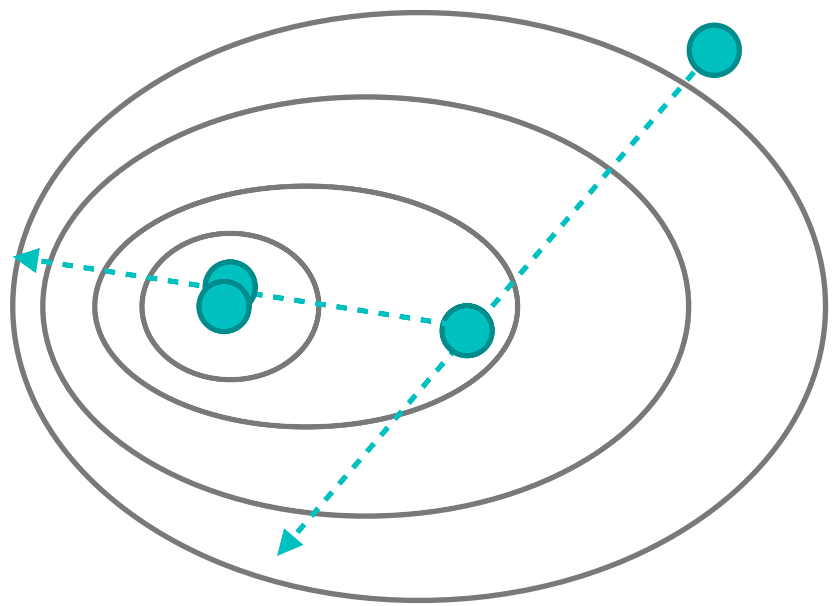
Line Search: a better method

- Line search in direction of gradient:

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

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

$$\eta \leftarrow \arg\min_{\eta} \sum_{i=1}^{M} (y^{(i)} - \hat{y}^{(i)})^2$$



Stochastic Methods

- How much computation is required for the gradient?

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

Per iteration:

M*N multiplies

2M add/subtract

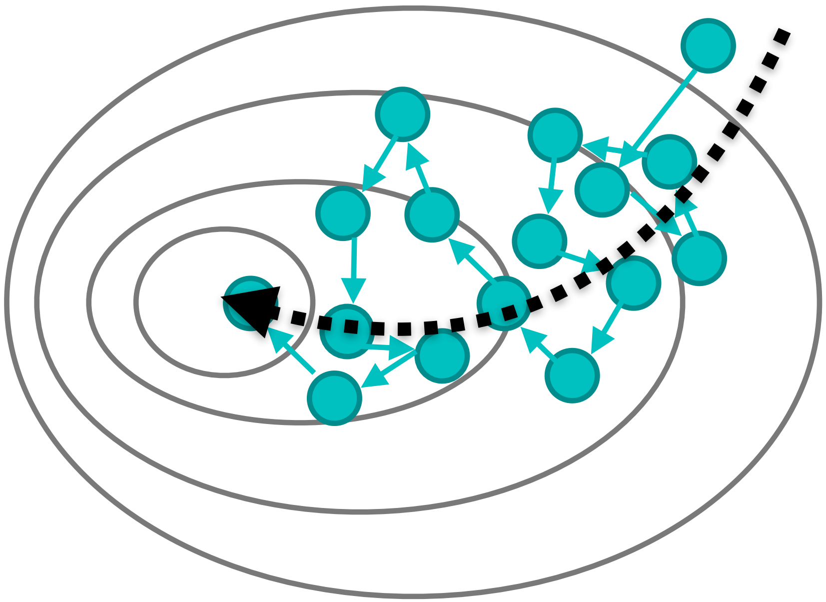
$$w \leftarrow w + \eta\underbrace{(y^{(i)} - \hat{y}^{(i)})x^{(i)}}_{\text{approx. gradient}}$$

i chosen at random

Per iteration:

N multiplies

1 add/subtract



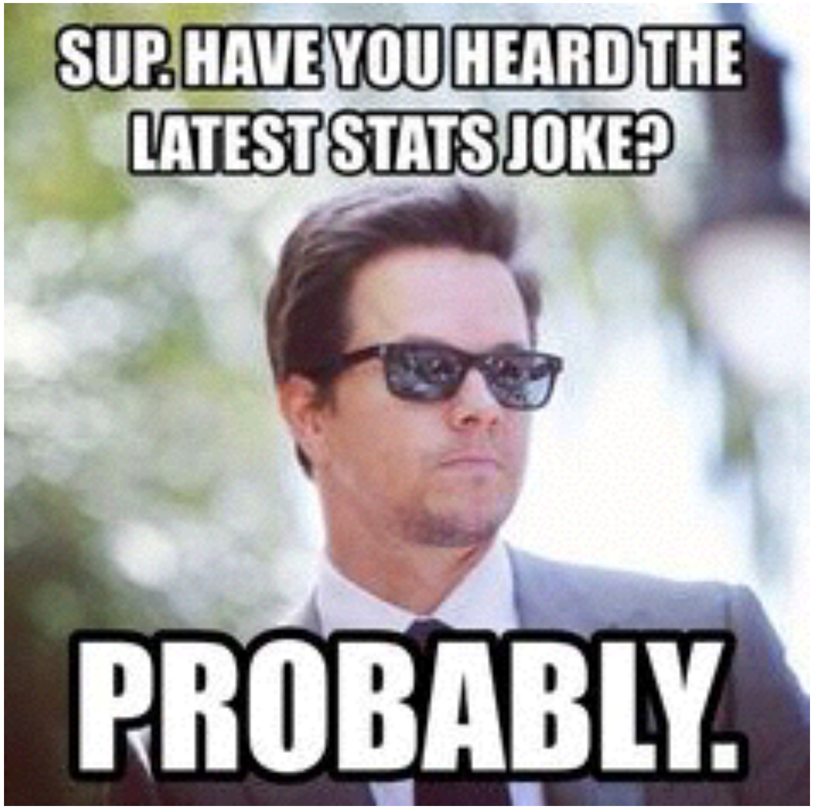
Demo

Numerical Optimization

Gradient Descent (with line search)

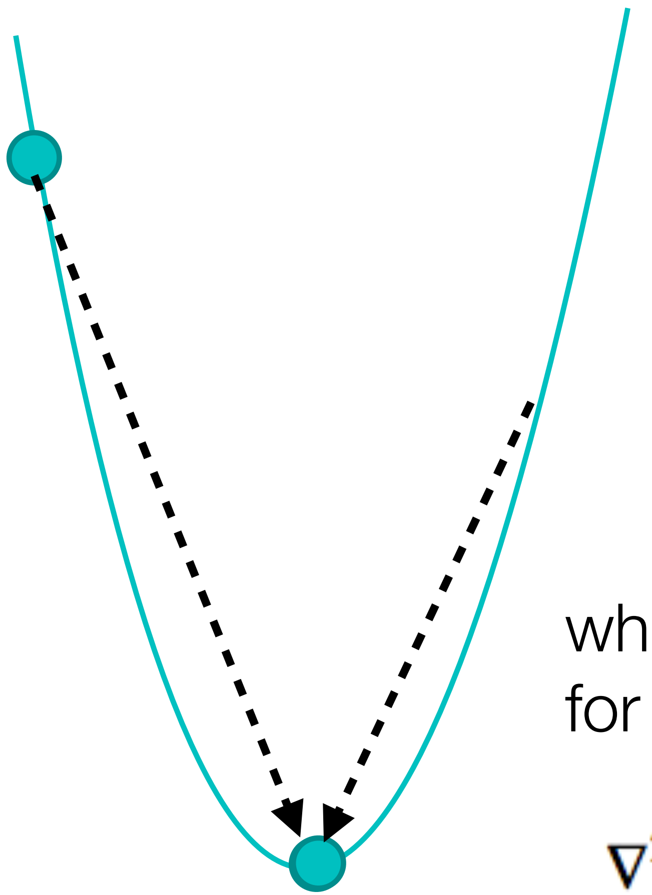
Stochastic Gradient Descent

Optimization Techniques with the Hessian



The Hessian

- Assume function is quadratic:



function of one variable:

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

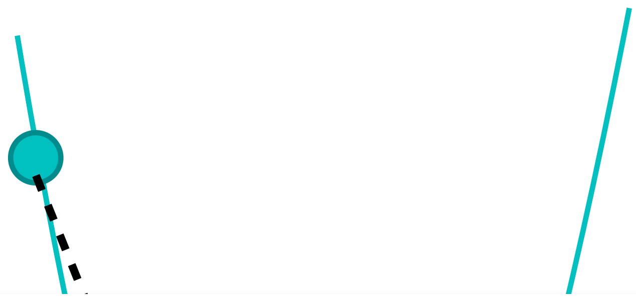
will solve in one step!

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:

$$w \leftarrow w + [\frac{\partial^2}{\partial w} l(w)]^{-1} \frac{\partial}{\partial w} l(w)$$

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

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

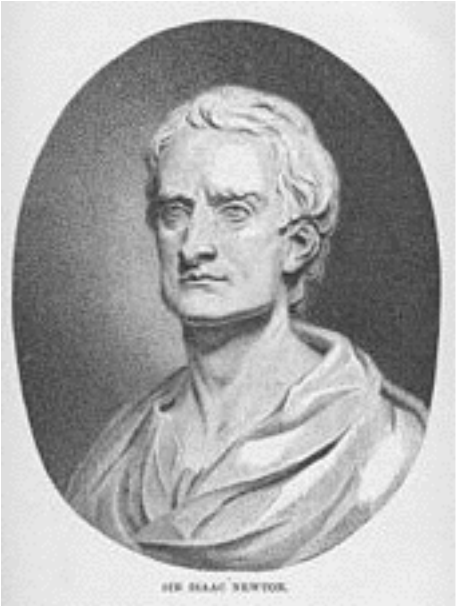
The Newton Update Method

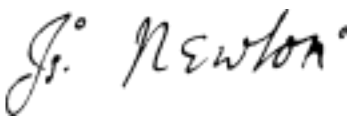
- Assume function is quadratic (in high dimensions):

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

$$w \leftarrow w + \eta \cdot \mathbf{H}[l(w)]^{-1} \cdot \nabla l(w)$$

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





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.

The Hessian for Logistic Regression

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

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

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

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

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

$$X * y_{diff}$$

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

Numerical Optimization

Newton’s method

Problems with only using Newton's Method

- Quadratic isn't always a great assumption:
  - highly dependent on starting point
    - jumps can get REALLY random!
  - near saddle points, inverse hessian unstable
  - hessian not always invertible…
    - or invertible with correct numerical precision

The solution: quasi Newton methods

- Typically built as follows:
  - approximate the Hessian with something numerically sound and readily invertible
  - back off to gradient descent when the approximate hessian is not stable
  - use momentum to update approximate hessian
- **A popular approach**: use Broyden-Fletcher-Goldfarb-Shanno (BFGS)
  - which you can look up if you are interested …

https://en.wikipedia.org/wiki/Broyden–Fletcher–Goldfarb–Shanno_algorithm

BFGS (if time)

1. $B_0 = I,$

2. $B_k \mathbf{p}_k = -\nabla f(\mathbf{x}_k)$

update direction

3. $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$

update equation

4. $\mathbf{s}_k = \alpha_k \mathbf{p}_k$

5. $\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$ intermediate constants

6. $B_{k+1} = B_k + \frac{\mathbf{y}_k \mathbf{y}_k^{\mathrm{T}}}{\mathbf{y}_k^{\mathrm{T}} \mathbf{s}_k} - \frac{B_k \mathbf{s}_k \mathbf{s}_k^{\mathrm{T}} B_k}{\mathbf{s}_k^{\mathrm{T}} B_k \mathbf{s}_k}$

redefine Hessian approx

$$B_{k+1}^{-1} = B_k^{-1} + \frac{(\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k + \mathbf{y}_k^{\mathrm{T}} B_k^{-1} \mathbf{y}_k)(\mathbf{s}_k \mathbf{s}_k^{\mathrm{T}})}{(\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k)^2} - \frac{B_k^{-1} \mathbf{y}_k \mathbf{s}_k^{\mathrm{T}} + \mathbf{s}_k \mathbf{y}_k^{\mathrm{T}} B_k^{-1}}{\mathbf{s}_k^{\mathrm{T}} \mathbf{y}_k}$$

invertibility of B well defined and only matrix operations

images: wikipedia

Numerical Optimization

BFGS (if time)
parallelization

For Next Lecture

- **Next time**: SVMs via in class assignment
- **Next Next time**: Neural Networks