UC San Diego

Topic 8: Applied Math (C) Optimization Methods

Group Number 9
Ryan Hammonds, Benjamin Pham, Gabriel Riegner
08 November 2022

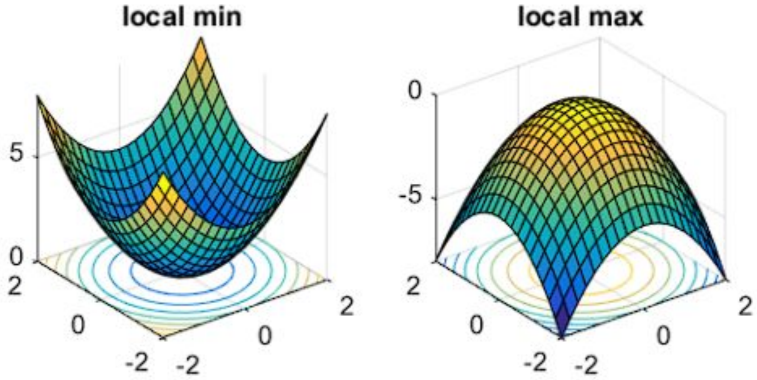
Sec 1: Introduction

Overview of Optimization

- Optimization involves minimizing (or maximizing) an objective or loss function.

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$$

- Solved using:
  - Linear algebra:
    - $\mathbf{A}\mathbf{x} = \mathbf{b}$
  - Iterative methods
    - Gradient Descent
    - (Quasi) Newton Method

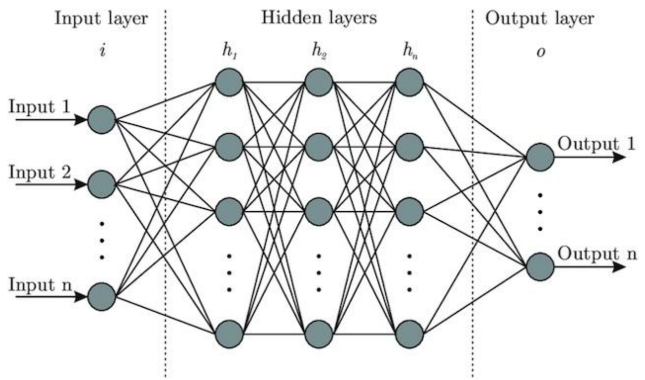


$$\nabla f(\mathbf{x}) = 0$$

Importance of the topic

- Optimization is required to solve regression and classification problems.
  - Parameter tuning
- Applies to simple and complex models.
  - Linear Regression
  - Neural networks
- Applications
  - Medicine
  - Economics / Finance
  - Computer vision
  - Speech recognition

$$
\begin{array}{rcl}
Y_1 & = & \beta_0 + \beta_1 X_1 + \epsilon_1 \\
Y_2 & = & \beta_0 + \beta_1 X_2 + \epsilon_2 \\
\vdots & \vdots & \vdots \\
Y_n & = & \beta_0 + \beta_1 X_n + \epsilon_n
\end{array}
$$



Sec 2: Problem Formulation

#1 Problem formulation

1. Let $\mathbf{x}$ be a vector of unknown parameters.
2. Let $\mathbf{y}$ be a vector of the known targets.
3. Given an arbitrary loss function (e.g. L0, L1, or L2), iteratively minimize:

$$f(\mathbf{x}) = loss(\hat{\mathbf{y}}, \mathbf{y})$$

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x})$$

4. At each iterative step k, update $\mathbf{x}$ by subtracting either:
   a. the gradient $\nabla \cdot$ scaled by step size η

   $$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta_k \nabla f(\mathbf{x}_k)$$

   b. the gradient $\nabla \cdot$ scaled by the inverse Hessian $\mathbf{H}^{-1}$

   $$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{H}(\mathbf{x}_k)^{-1} \nabla f(\mathbf{x}_k)$$

#2 Relation to Numerical Linear Algebra

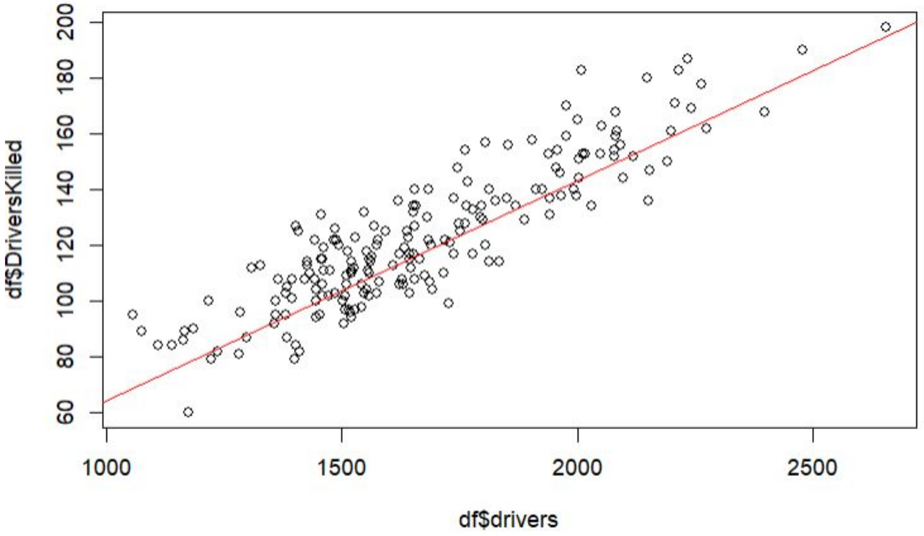
In general, an optimization problem involves solving a linear system with multiple parameters to minimize the loss function.

Solved via two main types of methods:

- Closed Form (Exact)
- Numerical (Estimation)

The solution of this linear system results in the set of the most optimal parameters

Simple Example



$$Y = \beta X + \epsilon$$

$$\underbrace{\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ 1 & x_{31} & x_{32} \\ \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} \end{bmatrix}}_{\text{Known}} \underbrace{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \epsilon_n \end{bmatrix}}_{\text{Unknown}}$$

Optimization Problem: Get β that creates the best fit line while minimizing the error term (distance between points and the best fit line)

Sec 3: State of the Art (SOTA)

Quasi Newton Methods: L-BFGS

- Computing the inverse Hessian is costly
  - Replace Hessian with an approximation
- Approximations must satisfy the secant equation:

$$\mathbf{B}_{k+1}\Delta\mathbf{x} = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$$

- L-BFGS
  - Store last *m* gradients and parameter estimates
  - At each step *k*:
    - Use last *m* estimates to approximate the Hessian

$$[\nabla f(x_{k-1}) \cdots \nabla f(x_{k-m})]$$

$$[x_{k-1} \cdots x_{k-m}]$$

Stochastic Gradient Descent

- Batch GD computes a gradient from the entire dataset, then steps.
    - batch_size = len(X)
- SGD computes a gradient from one instance, then steps.
    - batch_size = 1
- SOTA: First-Order Optimizers
    - Adam, AdaGrad, RMSProp
    - Adaptive learning rates (step sizes)

Normal Equation vs SGD vs Newton vs L-BFGS

Method	Normal Equation	SGD	Newton	L-BFGS
Speed	Depends on the size of X (slower as (X'X)^-1 becomes harder to compute)	Fast (approximation of Gradient)	Very Slow (computes the full Hessian at each update)	Medium (approximation of Hessian)
Convergence	N/A	Linear	Quadratic	Quadratic
Computational Cost	O(mn^2)	O(kmn)	O(m^3)	O(kn)
Type	Exact	Estimation	Estimation	Estimation

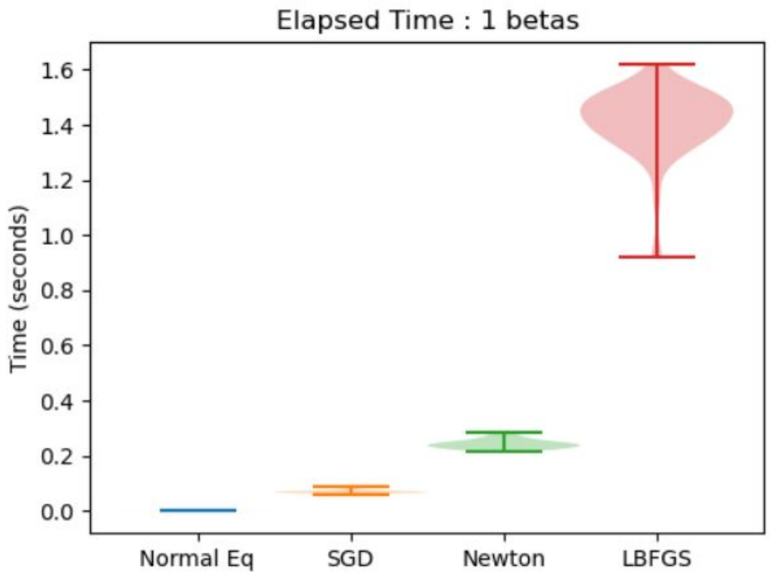
Scheme

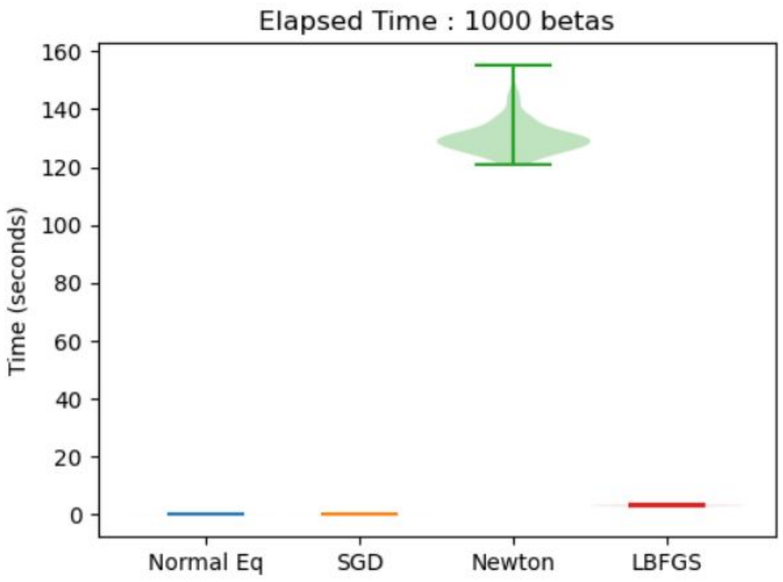
Simulations:

- Simulate Regression coefficients (1,10,100,1000 betas), Compare iterative methods
- 100 Simulations with observations (100,5000), L-BFGS and SGD; lr : 0.01
- Criterions: Accuracy (MSE), Computation Time (seconds), Effectiveness (Iteration to convergence)
- Goal: Identify the best iterative method for a general dataset type

Simple LR Simulation Results

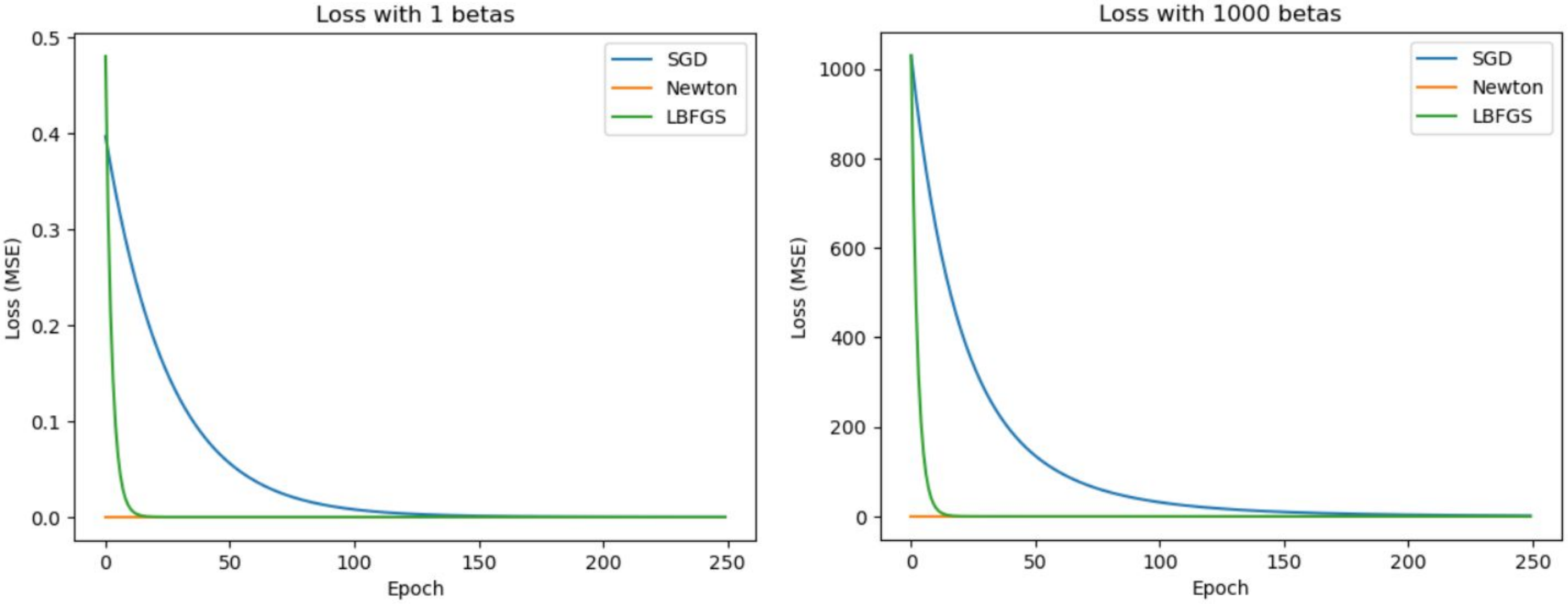
Simulation: n = 5000; 100 replicates

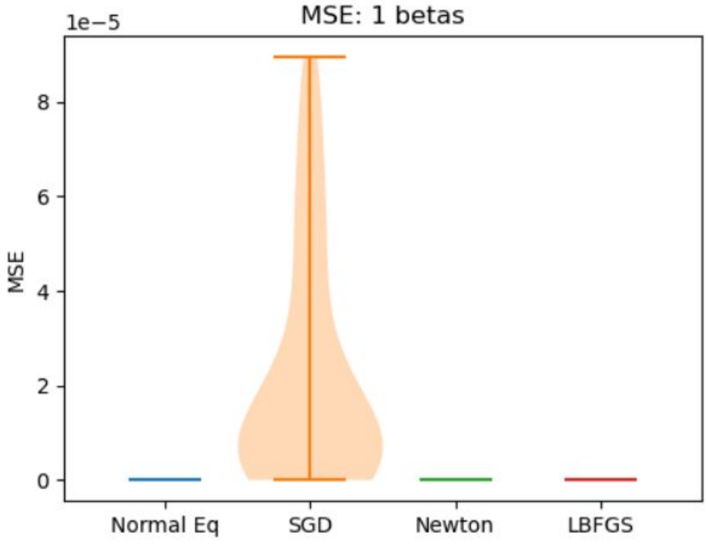


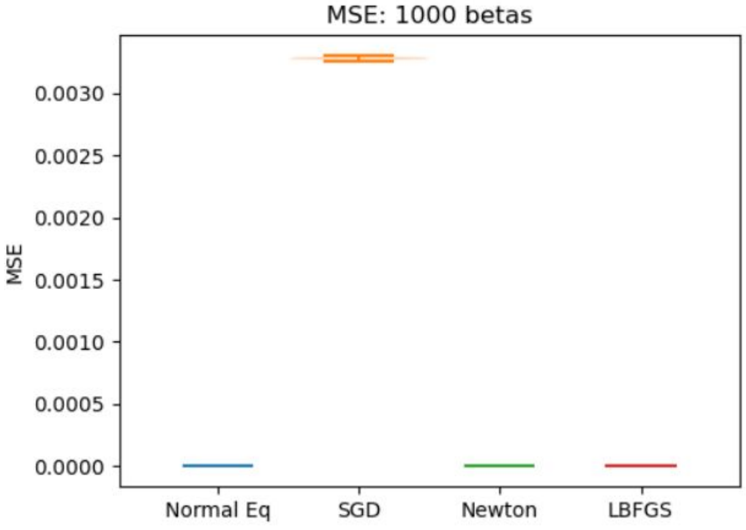


Second-Order Methods Converge Fast

Simulation: n = 5000; 100 replicates

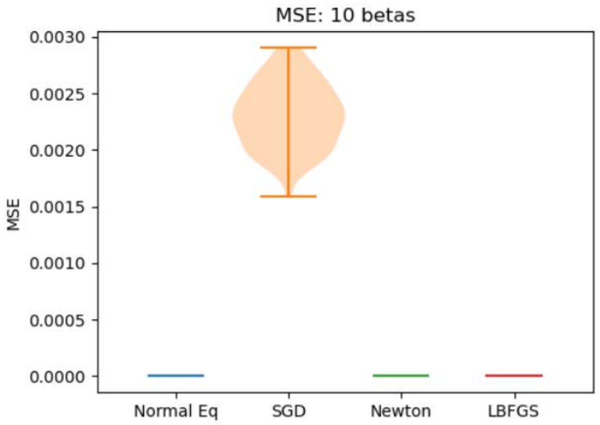


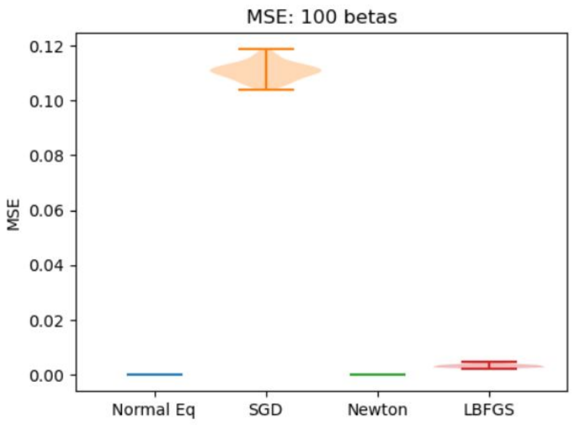

MSE: 1 betas
1e−5
MSE
0
2
4
6
8
Normal Eq
SGD
Newton
LBFGS

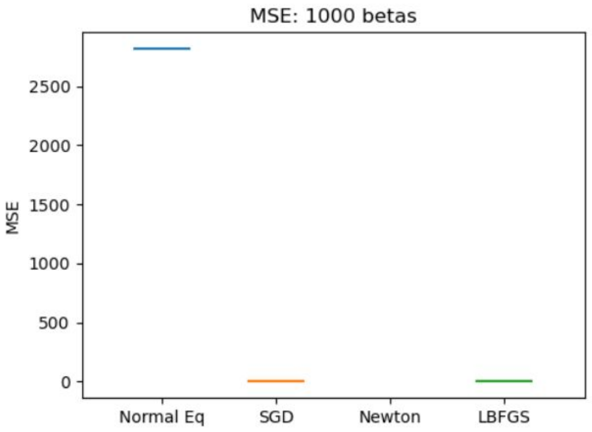

MSE: 1000 betas
MSE
0.0000
0.0005
0.0010
0.0015
0.0020
0.0025
0.0030
Normal Eq
SGD
Newton
LBFGS

At a High-Dimensional Case, the NLA approach starts to break…

Simulations: n = 100; 100 replicates

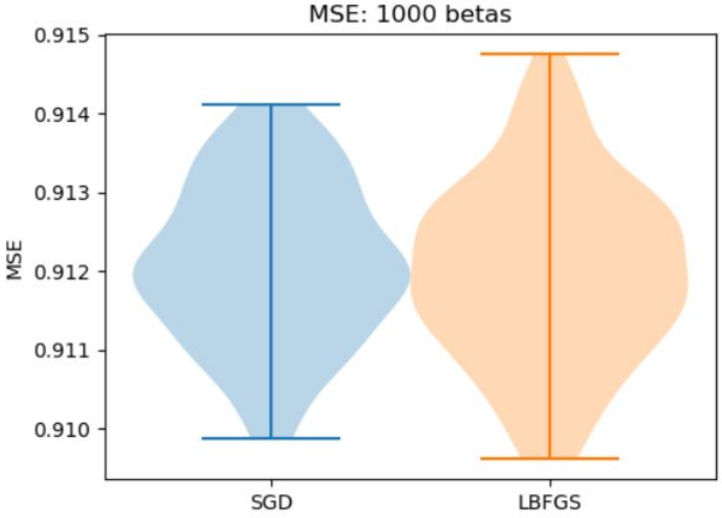


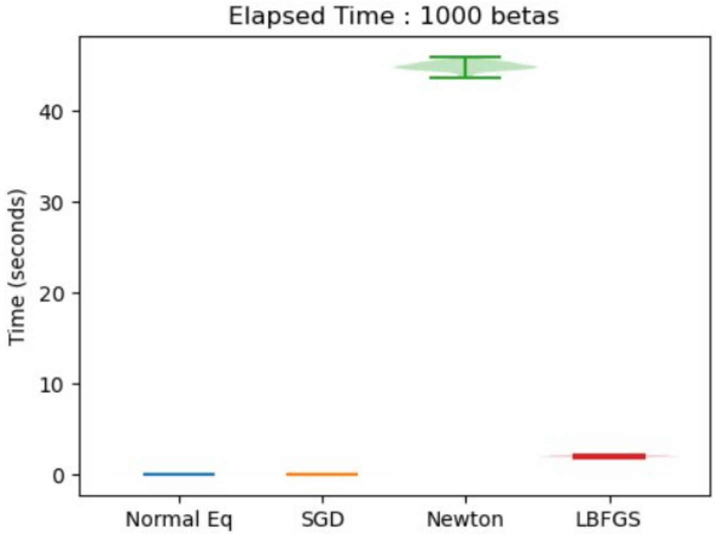




Computation Speed and MSE of SOTA Iterative Methods in the High-Dimensional case are still quite good

Simulations: n = 100; 100 replicates





Simple LR Simulation Summary

- SOTA methods were designed with large datasets in mind

- In general, SGD is very fast but estimates may not be as good as those found with L-BFGS

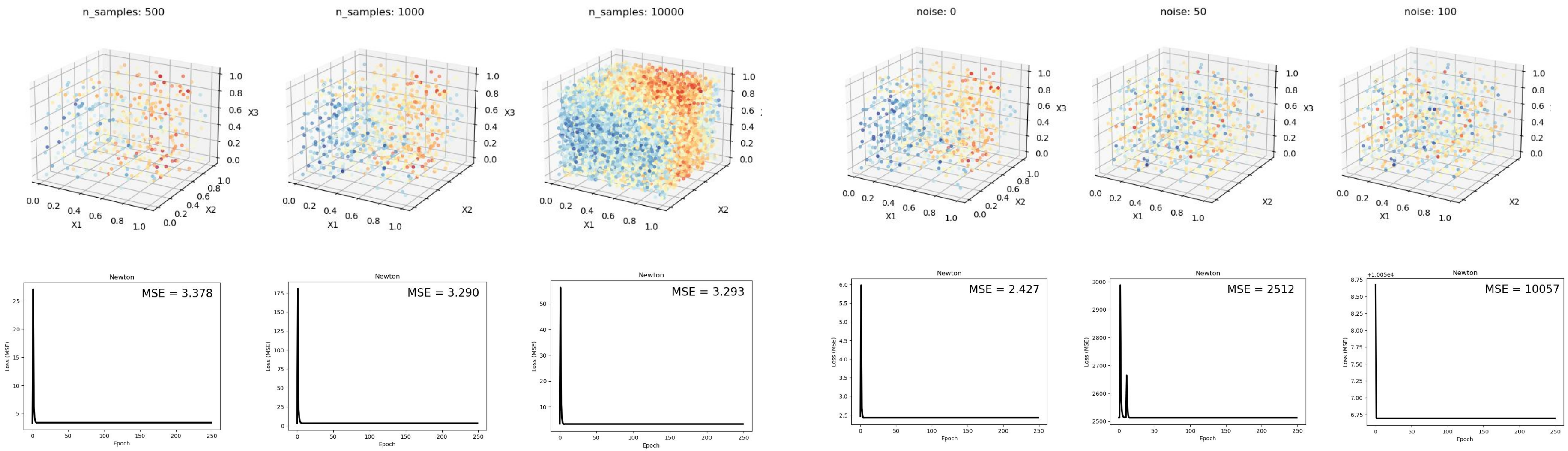
- For extreme cases, SOTA methods perform better than typical NLA approaches but the performance is not generalizable

Non-linear least squares

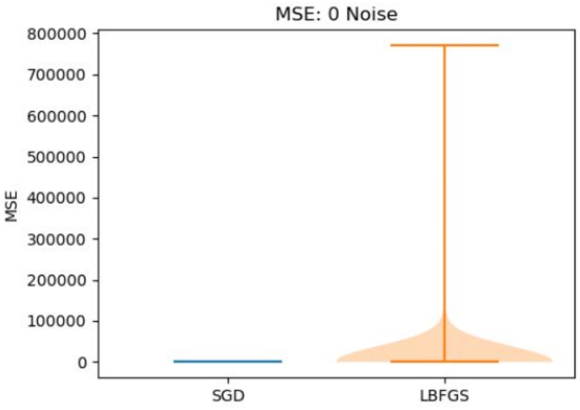
$$f(x) = 10sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \epsilon$$ **(known weights)**

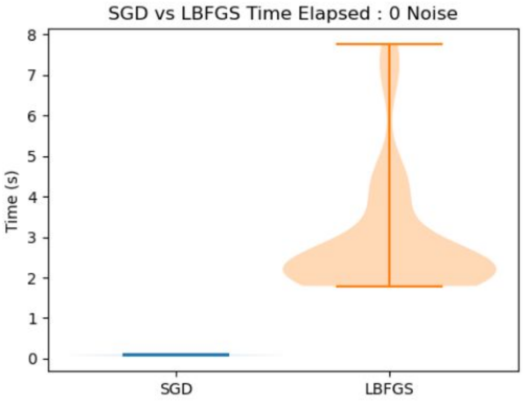
Friedman regression problem

$$f(x) = \beta_1 sin(\pi x_1 x_2) + \beta_2(x_3 - \beta_3)^2 + \beta_4 x_4 + \beta_5 x_5 + \epsilon \qquad \epsilon \sim \mathcal{N}(0,1)$$



LBFGS performs better than SGD with 0 noise but is sometimes unstable

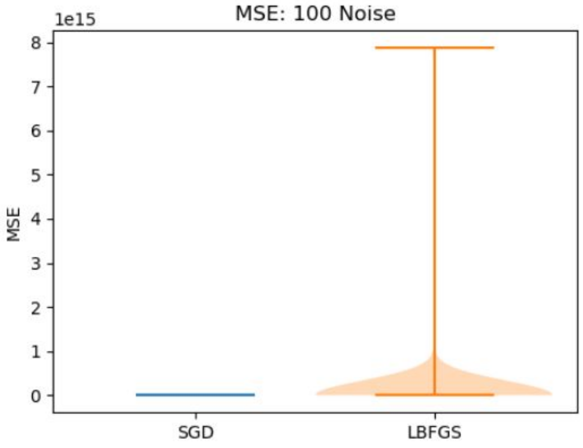


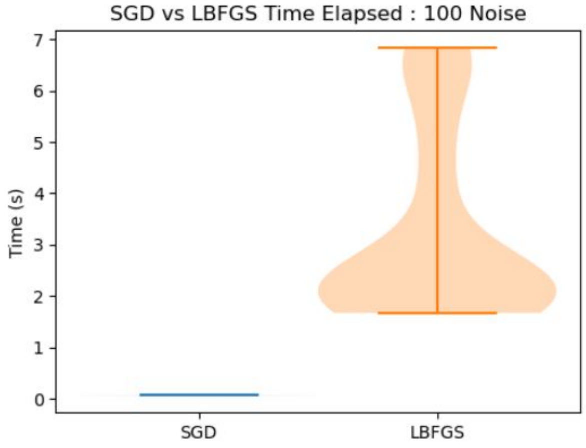


Descriptive summary of **MSE, Friedman, n = 100, noise = 0, 100 sims**

	SGD	LBFGS
Count (non-NA sims)	**100**	93
mean	82.61681	3.1E+11
std	4.025522	2.93E+12
min	72.7221	**5.57E-09**
25%	81.55313	**3.56E-08**
50%	83.72334	**4.04E-08**
75%	85.12831	**7.53E-08**
max	**88.90165**	2.82E+13

With Higher Noise, SGD and LBFGS solutions become more unstable





Descriptive summary of **MSE Friedman, n = 100, noise = 100, 100 sims**

	SGD	LBFGS
Count (non-NA sims)	71	**92**
mean	167.8209	8.55E+13
std	77.80087	8.2E+14
min	**61.92406**	139.574
25%	**124.2768**	599.0084
50%	**146.8394**	1256.123
75%	**201.3837**	3135.622
max	**437.6005**	7.87E+15

Scheme

Simulations:

- Simulate Regression coefficients (1,10,100,1000 betas), Compare iterative methods
- 100 replicates for each, L-BFGS and SGD, lr : 0.01
- Criterions: Accuracy (MSE), Computation Time (seconds), Effectiveness (Iteration to convergence)
- Goal: Identify the best iterative method for a general dataset type

Actual Data:

- Datasets: Diabetes and California Housing (Scikit-Learn)
- Criterion: Accuracy (MSE) in out-of-sample prediction and computation time
- We expect second order methods will outperform first order methods in accuracy, but not in computation time

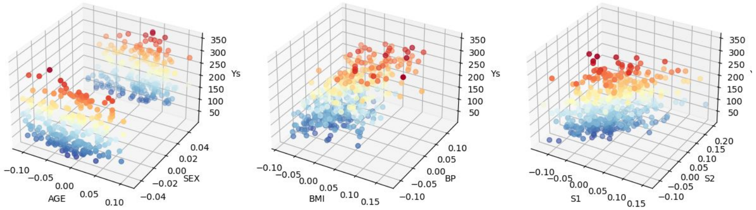
Datasets

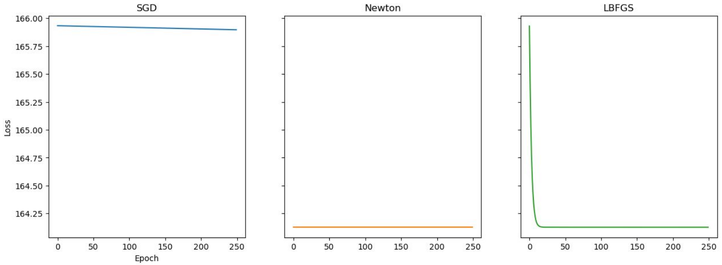
out-of-sample
prediction accuracy

Diabetes dataset > (420 samples, 6 features)

X : age, sex, body mass index, blood pressure, cholesterol levels (s1-2)

y : quantitative measure of disease progression one year after baseline



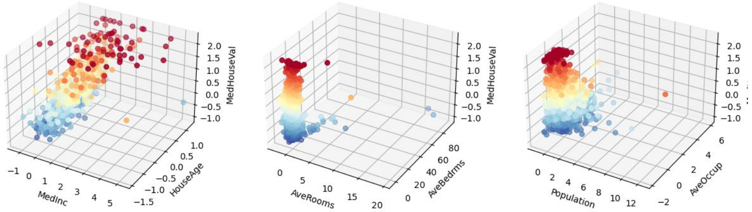


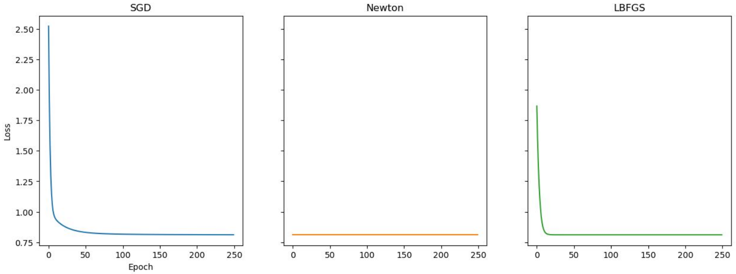

```
>>> lstsq
MSE = 166.001

>>> sgd
MSE = 168.945 (elapsed time = 0.095 s)

>>> newton
MSE = 178.506 (elapsed time = 0.709 s)

>>> lbfgs
MSE = 178.502 (elapsed time = 1.619 s)
```

California housing dataset > (20640 samples, 6 features)

X : medium income, house age, average rooms, average bedrooms, population, household members

y : medium house value for California districts






```
>>> lstsq
MSE = 0.051

>>> sgd
MSE = 0.159 (elapsed time = 69.567 s)

>>> newton
MSE = 0.156 (elapsed time = 560.798 s)

>>> lbfgs
MSE = 0.156 (elapsed time = 318.592 s)
```

Sec 4: Concluding remarks

Conclusions

- We explored different optimization methods in the context of both least-squares linear regression and non-linear regression
- We confirmed our expectations with rigorous theoretical simulations
- We applied these methods to real datasets
- We utilized numpy, pytorch, and pandas to carry out these experiments
- We conclude that the best method for these problems is dependent on the data and available computational resources

References

https://course.ece.cmu.edu/~ece739/lectures/18739-2020-spring-lecture-08-second-order.pdf

https://www.psychologie.uni-heidelberg.de/ae/meth/team/mertens/blog/hessian.nb.html

https://www.inf.ed.ac.uk/teaching/courses/irds/miniproject-datasets.html

Hardt, M., & Recht, B. (2022). Patterns, predictions, and actions: A story about machine learning. Princeton University Press.

Boyd, S., & Vandenberghe, L. (2004). Convex optimization. Cambridge university press.

J. Friedman, "Multivariate adaptive regression splines", The Annals of Statistics 19 (1), pages 1-67, 1991.