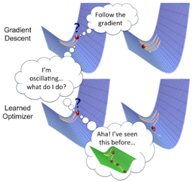
CS6301: Optimization in Machine Learning

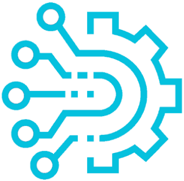
Lecture 2: Basics of Continuous Optimization

Rishabh Iyer

Department of Computer Science
University of Texas, Dallas
https://sites.google.com/view/cs-6301-optml/home

January 15, 2020

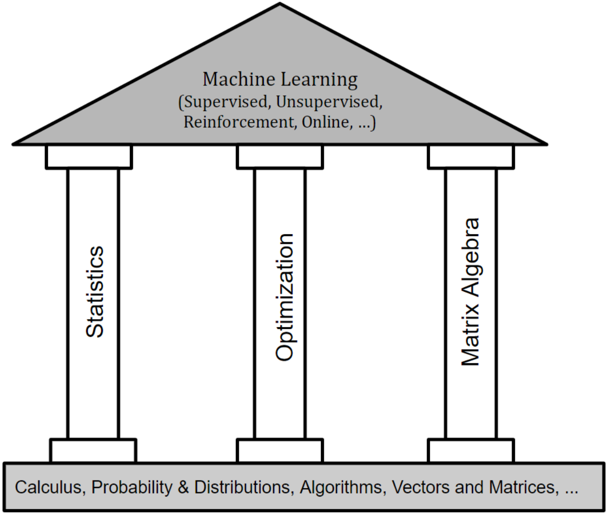




Outline

- Recap from Previous Lecture
- Brief Project Discussion
- Notation and Basics of Continuous optimization: Vectors, Matrices, Gradients, Hessians, ...
- Deriving Gradients and Hessian for various Loss Functions
- Practical Aspects: Implementing Loss Functions and Gradients in Python.

Why take this Course?



Why take this Course?

- Optimization everywhere in Machine Learning.
- Countless number of ML libraries available which implement all kinds of optimization algorithms (Tensorflow, pytorch, scipy, sklearn, Vowpal Wabbit, ...)
- This course will give you the expertise to look inside these algorithms, understand how they work, why they work and how fast they work.
- Invariably in Research, you will come up with new optimization problems which you might need to implement custom algorithms or atleast the loss functions.
- Over 70% of the projects I worked at Microsoft involved new optimization problems (and sometimes new algorithms)
- Even if you don't implement new algorithms, you will have a better idea of which algorithm to use in which sceario.

First Half of this Course: Continuous optimization

- Basics of Continuous Optimization
- Convexity
- Gradient Descent
- Projected/Proximal GD
- Subgradient Descent
- Accelerated Gradient Descent
- Newton & Quasi Newton
- Duality: Legrange, Fenchel
- Coordinate Descent
- Frank Wolfe
- Optimization in Practice

Second Half of this Course: Discrete optimization

- Linear Cost Problems
- Matroids, Spanning Trees
- s-t paths, s-t cuts
- Matchings
- Covers (Set Covers, Vertex Covers, Edge Covers)
- Optimal Transport (if time permits)

- Non-Linear Discrete Optimization
- Submodular Functions
- Submodularity and Convexity
- Submodular Minimization
- Submodular Maximization
- Optimization in Practice

Evaluation

- Assignments (50%):
  - Assignments will comprise of a mix of practical and theoretical.
  - Simple Theory questions, e.g. proving certain loss functions are convex/non-convex, computing gradients etc.
  - Practical $\Rightarrow$ implementing optimization algorithms and loss functions in python.
  - One assignment every week (or maybe every 2 weeks)

Evaluation

- Assignments (50%):
  - Assignments will comprise of a mix of practical and theoretical.
  - Simple Theory questions, e.g. proving certain loss functions are convex/non-convex, computing gradients etc.
  - Practical $\Rightarrow$ implementing optimization algorithms and loss functions in python.
  - One assignment every week (or maybe every 2 weeks)
- Project: (50%):
  - Examples: Picking a certain real world problem, modeling the problem and optimizing the loss function with different algorithms and drawing insights on its performance.
  - Alternatively,also be a survey on a particular class of optimization algorithms and theoretical results
  - Could also be creating a toolkit (python/c++) of the various optimization/learning algorithms.

Course Project Ideas

- Let's spend a few minutes discussing some ideas for course project(s)

Course Project Ideas

- Let's spend a few minutes discussing some ideas for course project(s)
- Why not we all jointly create a *OptML* python toolkit which implements several (discrete and continuous) loss functions, optimization algorithms along with wrappers to machine learning models (e.g. classification, recommender systems, regression etc.)

Course Project Ideas

- Let's spend a few minutes discussing some ideas for course project(s)
- Why not we all jointly create a *OptML* python toolkit which implements several (discrete and continuous) loss functions, optimization algorithms along with wrappers to machine learning models (e.g. classification, recommender systems, regression etc.)
- Why another toolkit when there are already so many out there?

Course Project Ideas

- Let's spend a few minutes discussing some ideas for course project(s)
- Why not we all jointly create a *OptML* python toolkit which implements several (discrete and continuous) loss functions, optimization algorithms along with wrappers to machine learning models (e.g. classification, recommender systems, regression etc.)
- Why another toolkit when there are already so many out there?
- A lot of the base for this toolkit will already be covered in this course

Course Project Ideas

- Let's spend a few minutes discussing some ideas for course project(s)
- Why not we all jointly create a *OptML* python toolkit which implements several (discrete and continuous) loss functions, optimization algorithms along with wrappers to machine learning models (e.g. classification, recommender systems, regression etc.)
- Why another toolkit when there are already so many out there?
- A lot of the base for this toolkit will already be covered in this course
- Each group can take on a particular component of the toolkit: with components as a) linear classification/regression, b) non-linear classification/regression, c) recommendation and matrix factorization, d) contextual bandits, e) submodular minimization, f) submodular maximization g) graph algorithms and so on...

Course Project Ideas

- Let's spend a few minutes discussing some ideas for course project(s)
- Why not we all jointly create a *OptML* python toolkit which implements several (discrete and continuous) loss functions, optimization algorithms along with wrappers to machine learning models (e.g. classification, recommender systems, regression etc.)
- Why another toolkit when there are already so many out there?
- A lot of the base for this toolkit will already be covered in this course
- Each group can take on a particular component of the toolkit: with components as a) linear classification/regression, b) non-linear classification/regression, c) recommendation and matrix factorization, d) contextual bandits, e) submodular minimization, f) submodular maximization g) graph algorithms and so on...
- We can discuss ideas on this as the class progresses.

Continuous Optimization in Machine Learning

- Supervised Learning: Logistic Regression, Least Squares, Support Vector Machines, Deep Models
- Unsupervised Learning: k-Means Clustering, Principal Component Analysis
- Contextual Bandits and Reinforcement Learning: Soft-Max Estimators, Policy Exponential Models
- Recommender Systems: Matrix Completion, Non-Negative Matrix Factorization, Collaborative Filtering

Supervised Learning: Optimization Problem

- "Loss plus Regularizer" Framework:

$$\min_{\theta} G(\theta) = \sum_{i=1}^{n} L(F_\theta(x_i), y_i) + \lambda\Omega(\theta)$$

- L: *Loss function*, Ω: *Regularizer*. Example $F_\theta(x) = \theta^T x$

Supervised Learning: Optimization Problem

- "Loss plus Regularizer" Framework:

$$\min_{\theta} G(\theta) = \sum_{i=1}^{n} L(F_\theta(x_i), y_i) + \lambda\Omega(\theta)$$

- L: *Loss function*, Ω: *Regularizer*. Example $F_\theta(x) = \theta^T x$

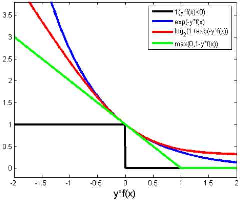
Supervised Learning: Optimization Problem

- "Loss plus Regularizer" Framework:

$$\min_{\theta} G(\theta) = \sum_{i=1}^{n} L(F_\theta(x_i), y_i) + \lambda \Omega(\theta)$$

- L: *Loss function*, Ω: *Regularizer*. Example $F_\theta(x) = \theta^T x$

- Examples of L:
  - Logistic Loss: $\log(1 + \exp(-y_i F_\theta(x_i)))$
  - Hinge Loss: $\max\{0, 1 - y_i F_\theta(x_i)\}$
  - Softmax Loss: $-F_{\theta_{y_i}}(x_i) + \log(\sum_{c=1}^{k} \exp(F_{\theta_c}(x_i)))$
  - Absolute Error: $|F_\theta(x_i) - y_i|$
  - Least Squares: $(F_\theta(x_i) - y_i)^2$



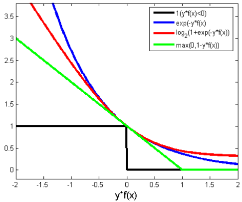
Supervised Learning: Optimization Problem

- "Loss plus Regularizer" Framework:

$$\min_{\theta} G(\theta) = \sum_{i=1}^{n} L(F_\theta(x_i), y_i) + \lambda\Omega(\theta)$$

- L: *Loss function*, Ω: *Regularizer*. Example $F_\theta(x) = \theta^T x$

- Examples of L:
  - Logistic Loss: $\log(1 + \exp(-y_i F_\theta(x_i)))$
  - Hinge Loss: $\max\{0, 1 - y_i F_\theta(x_i)\}$
  - Softmax Loss: $-F_{\theta_{y_i}}(x_i) + \log(\sum_{c=1}^{k} \exp(F_{\theta_c}(x_i)))$
  - Absolute Error: $|F_\theta(x_i) - y_i|$
  - Least Squares: $(F_\theta(x_i) - y_i)^2$



- Examples of Ω:
  - L1 Regularizer: $||\theta||_1 = \sum_{i=1}^{m} |\theta[i]|$
  - L2 Regularizer: $||\theta||_2^2 = \sum_{i=1}^{m} \theta[i]^2$

Derivatives

- Given a function $y = f(x)$, we denote the derivative $\frac{dy}{dx} = \frac{df(x)}{dx}$.

Derivatives

- Given a function $y = f(x)$, we denote the derivative $\frac{dy}{dx} = \frac{df(x)}{dx}$.
- Recall some simple Derivative Rules:

Derivatives

- Given a function $y = f(x)$, we denote the derivative $\frac{dy}{dx} = \frac{df(x)}{dx}$.
- Recall some simple Derivative Rules:
- Constant: $y = c, \frac{dy}{dx} = 0$

Derivatives

- Given a function $y = f(x)$, we denote the derivative $\frac{dy}{dx} = \frac{df(x)}{dx}$.
- Recall some simple Derivative Rules:
- Constant: $y = c, \frac{dy}{dx} = 0$
- Multiplication by a constant: $y = cf(x), \frac{dy}{dx} = c\frac{df(x)}{dx}$.

Derivatives

- Given a function $y = f(x)$, we denote the derivative $\frac{dy}{dx} = \frac{df(x)}{dx}$.
- Recall some simple Derivative Rules:
- Constant: $y = c, \frac{dy}{dx} = 0$
- Multiplication by a constant: $y = cf(x), \frac{dy}{dx} = c\frac{df(x)}{dx}$.
- Power: $y = x^a, \frac{dy}{dx} = ax^{a-1}$.

Derivatives

- Given a function $y = f(x)$, we denote the derivative $\frac{dy}{dx} = \frac{df(x)}{dx}$.
- Recall some simple Derivative Rules:
- Constant: $y = c, \frac{dy}{dx} = 0$
- Multiplication by a constant: $y = cf(x), \frac{dy}{dx} = c\frac{df(x)}{dx}$.
- Power: $y = x^a, \frac{dy}{dx} = ax^{a-1}$.
- Sum rule: $y = f(x) + g(x), \frac{dy}{dx} = \frac{df(x)}{dx} + \frac{dg(x)}{dx}$.

Derivatives

- Given a function $y = f(x)$, we denote the derivative $\frac{dy}{dx} = \frac{df(x)}{dx}$.
- Recall some simple Derivative Rules:
- Constant: $y = c, \frac{dy}{dx} = 0$
- Multiplication by a constant: $y = cf(x), \frac{dy}{dx} = c\frac{df(x)}{dx}$.
- Power: $y = x^a, \frac{dy}{dx} = ax^{a-1}$.
- Sum rule: $y = f(x) + g(x), \frac{dy}{dx} = \frac{df(x)}{dx} + \frac{dg(x)}{dx}$.
- Difference rule: $y = f(x) - g(x), \frac{dy}{dx} = \frac{df(x)}{dx} - \frac{dg(x)}{dx}$.

Derivatives

- Given a function $y = f(x)$, we denote the derivative $\frac{dy}{dx} = \frac{df(x)}{dx}$.
- Recall some simple Derivative Rules:
- Constant: $y = c, \frac{dy}{dx} = 0$
- Multiplication by a constant: $y = cf(x), \frac{dy}{dx} = c\frac{df(x)}{dx}$.
- Power: $y = x^a, \frac{dy}{dx} = ax^{a-1}$.
- Sum rule: $y = f(x) + g(x), \frac{dy}{dx} = \frac{df(x)}{dx} + \frac{dg(x)}{dx}$.
- Difference rule: $y = f(x) - g(x), \frac{dy}{dx} = \frac{df(x)}{dx} - \frac{dg(x)}{dx}$.
- Product rule: $y = f(x)g(x), \frac{dy}{dx} = f(x)\frac{dg(x)}{dx} + g(x)\frac{df(x)}{dx}$.

Derivatives

- Given a function $y = f(x)$, we denote the derivative $\frac{dy}{dx} = \frac{df(x)}{dx}$.
- Recall some simple Derivative Rules:
- Constant: $y = c$, $\frac{dy}{dx} = 0$
- Multiplication by a constant: $y = cf(x)$, $\frac{dy}{dx} = c\frac{df(x)}{dx}$.
- Power: $y = x^a$, $\frac{dy}{dx} = ax^{a-1}$.
- Sum rule: $y = f(x) + g(x)$, $\frac{dy}{dx} = \frac{df(x)}{dx} + \frac{dg(x)}{dx}$.
- Difference rule: $y = f(x) - g(x)$, $\frac{dy}{dx} = \frac{df(x)}{dx} - \frac{dg(x)}{dx}$.
- Product rule: $y = f(x)g(x)$, $\frac{dy}{dx} = f(x)\frac{dg(x)}{dx} + g(x)\frac{df(x)}{dx}$.
- Chain rule: $y = f(g(x))$, $\frac{dy}{dx} = \frac{df(g(x))}{dg(x)}\frac{dg(x)}{dx}$.

Derivatives

- Given a function $y = f(x)$, we denote the derivative $\frac{dy}{dx} = \frac{df(x)}{dx}$.
- Recall some simple Derivative Rules:
- Constant: $y = c, \frac{dy}{dx} = 0$
- Multiplication by a constant: $y = cf(x), \frac{dy}{dx} = c\frac{df(x)}{dx}$.
- Power: $y = x^a, \frac{dy}{dx} = ax^{a-1}$.
- Sum rule: $y = f(x) + g(x), \frac{dy}{dx} = \frac{df(x)}{dx} + \frac{dg(x)}{dx}$.
- Difference rule: $y = f(x) - g(x), \frac{dy}{dx} = \frac{df(x)}{dx} - \frac{dg(x)}{dx}$.
- Product rule: $y = f(x)g(x), \frac{dy}{dx} = f(x)\frac{dg(x)}{dx} + g(x)\frac{df(x)}{dx}$.
- Chain rule: $y = f(g(x)), \frac{dy}{dx} = \frac{df(g(x))}{dg(x)}\frac{dg(x)}{dx}$.
- If $y = \frac{f(x)}{g(x)}$, can you compute $\frac{dy}{dx}$?

Examples Derivatives

- Examples of Derivatives of some important functions:

Examples Derivatives

- Examples of Derivatives of some important functions:
- Power Function: $y = x^a$, $\frac{dy}{dx} = ax^{a-1}$.

Examples Derivatives

- Examples of Derivatives of some important functions:
- Power Function: $y = x^a, \frac{dy}{dx} = ax^{a-1}$.
- Exponential: $y = e^x, \frac{dy}{dx} = e^x$.

Examples Derivatives

- Examples of Derivatives of some important functions:
- Power Function: $y = x^a, \frac{dy}{dx} = ax^{a-1}$.
- Exponential: $y = e^x, \frac{dy}{dx} = e^x$.
- Logarithmic: $y = \log x, \frac{dy}{dx} = \frac{1}{x}$.

Examples Derivatives

- Examples of Derivatives of some important functions:
- Power Function: $y = x^a$, $\frac{dy}{dx} = ax^{a-1}$.
- Exponential: $y = e^x$, $\frac{dy}{dx} = e^x$.
- Logarithmic: $y = \log x$, $\frac{dy}{dx} = \frac{1}{x}$.
- Max: $y = \max\{x, 0\}$, $\frac{dy}{dx} = 1$, if $x \geq 0$ and 0 else ($x = 0$ is a point of non-differentiability)

Examples Derivatives

- Examples of Derivatives of some important functions:
- Power Function: $y = x^a$, $\frac{dy}{dx} = ax^{a-1}$.
- Exponential: $y = e^x$, $\frac{dy}{dx} = e^x$.
- Logarithmic: $y = \log x$, $\frac{dy}{dx} = \frac{1}{x}$.
- Max: $y = \max\{x, 0\}$, $\frac{dy}{dx} = 1$, if $x \geq 0$ and 0 else ($x = 0$ is a point of non-differentiability)
- Sin and Cos Functions: $y = \sin(x)$, $\frac{dy}{dx} = \cos(x)$. Similarly $y = \cos(x)$, $\frac{dy}{dx} = \sin(x)$

Examples Derivatives

- Examples of Derivatives of some important functions:
- Power Function: $y = x^a, \frac{dy}{dx} = ax^{a-1}$.
- Exponential: $y = e^x, \frac{dy}{dx} = e^x$.
- Logarithmic: $y = \log x, \frac{dy}{dx} = \frac{1}{x}$.
- Max: $y = \max\{x, 0\}, \frac{dy}{dx} = 1$, if $x \geq 0$ and 0 else ($x = 0$ is a point of non-differentiability)
- Sin and Cos Functions: $y = \sin(x), \frac{dy}{dx} = \cos(x)$. Similarly $y = \cos(x), \frac{dy}{dx} = \sin(x)$
- Derivatives for most loss functions can be obtained with the basic derivatives above and with the right choices of product/division/addition and chain rules (basic principal of autograd).

Partial Derivatives

- Computing the partial derivative of simple functions is easy: simply treat every other variable in the equation as a constant and find the usual scalar derivative.

Partial Derivatives

- Computing the partial derivative of simple functions is easy: simply treat every other variable in the equation as a constant and find the usual scalar derivative.
- Example: Consider the function $f(x_1, x_2) = 3x_1^2 x_2$.

Partial Derivatives

- Computing the partial derivative of simple functions is easy: simply treat every other variable in the equation as a constant and find the usual scalar derivative.
- Example: Consider the function $f(x_1, x_2) = 3x_1^2 x_2$.
- We denote the partial derivative as $\frac{\partial f(x_1, x_2)}{\partial x_1}$.

Partial Derivatives

- Computing the partial derivative of simple functions is easy: simply treat every other variable in the equation as a constant and find the usual scalar derivative.
- Example: Consider the function $f(x_1, x_2) = 3x_1^2 x_2$.
- We denote the partial derivative as $\frac{\partial f(x_1,x_2)}{\partial x_1}$.
- Then $\frac{\partial f(x_1,x_2)}{\partial x_1} = 3x_2 \frac{\partial x_1^2}{\partial x_1} = 6x_1 x_2$.

Partial Derivatives

- Computing the partial derivative of simple functions is easy: simply treat every other variable in the equation as a constant and find the usual scalar derivative.
- Example: Consider the function $f(x_1, x_2) = 3x_1^2 x_2$.
- We denote the partial derivative as $\frac{\partial f(x_1,x_2)}{\partial x_1}$.
- Then $\frac{\partial f(x_1,x_2)}{\partial x_1} = 3x_2 \frac{\partial x_1^2}{\partial x_1} = 6x_1 x_2$.
- Similarly $\frac{\partial f(x_1,x_2)}{\partial x_2} = 3x_1^2 \frac{\partial x_2}{\partial x_2} = 3x_1^2$.

Gradient

- For ease of notation, define the vector $w = [w_1 \ \cdots \ w_m]^T$.

Gradient

- For ease of notation, define the vector $w = [w_1 \ \cdots \ w_m]^T$.
- We can write the Loss function $L(w) = L(w_1, \cdots, w_m)$.

Gradient

- For ease of notation, define the vector $w = [w_1 \ \cdots \ w_m]^T$.
- We can write the Loss function $L(w) = L(w_1, \cdots, w_m)$.
- The gradient $\nabla L(w)$ is defined as:

$$\nabla L(w) = [\frac{\partial L(w_1, \cdots, w_m)}{\partial w_1} \ \ \ldots \ \ \frac{\partial L(w_1, \cdots, w_m)}{\partial w_m}]^T$$

Gradient

- For ease of notation, define the vector $w = [w_1 \ \cdots \ w_m]^T$.
- We can write the Loss function $L(w) = L(w_1, \cdots, w_m)$.
- The gradient $\nabla L(w)$ is defined as:

$$\nabla L(w) = [\frac{\partial L(w_1, \cdots, w_m)}{\partial w_1} \ \cdots \ \frac{\partial L(w_1, \cdots, w_m)}{\partial w_m}]^T$$

- For simplicity of notation, we can write this as:

$$\nabla L(w) = \left[\frac{\partial L}{\partial w_1} \ \frac{\partial L}{\partial w_2} \ \cdots \ \frac{\partial L}{\partial w_i} \ \cdots \ \frac{\partial L}{\partial w_m}\right]$$

Gradient

- For ease of notation, define the vector $w = [w_1 \ \cdots \ w_m]^T$.
- We can write the Loss function $L(w) = L(w_1, \cdots, w_m)$.
- The gradient $\nabla L(w)$ is defined as:

$$\nabla L(w) = [\frac{\partial L(w_1, \cdots, w_m)}{\partial w_1} \ \cdots \ \frac{\partial L(w_1, \cdots, w_m)}{\partial w_m}]^T$$

- For simplicity of notation, we can write this as:

$$\nabla L(w) = \left[\frac{\partial L}{\partial w_1} \ \frac{\partial L}{\partial w_2} \ \cdots \ \frac{\partial L}{\partial w_i} \ \cdots \ \frac{\partial L}{\partial w_m}\right]$$

- Compute the gradient with $L(w_1, w_2) = w_1^2/w_2$:

$$\nabla L(w_1, w_2) = [2w_1/w_2 \quad -w_1^2/w_2^2]$$

Hessian

- We can similarly compute the Hessian of a Function:

$$\nabla^2 f(w) = \begin{pmatrix} \frac{\partial^2 f}{\partial w_1^2} & \frac{\partial^2 f}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_1 \partial w_n} \\ \frac{\partial^2 f}{\partial w_2 \partial w_1} & \frac{\partial^2 f}{\partial w_2^2} & \cdots & \frac{\partial^2 f}{\partial w_2 \partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial w_n \partial w_1} & \frac{\partial^2 f}{\partial w_n \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_n^2} \end{pmatrix}$$

Hessian

- We can similarly compute the Hessian of a Function:

$$\nabla^2 f(w) = \begin{pmatrix} \frac{\partial^2 f}{\partial w_1^2} & \frac{\partial^2 f}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_1 \partial w_n} \\ \frac{\partial^2 f}{\partial w_2 \partial w_1} & \frac{\partial^2 f}{\partial w_2^2} & \cdots & \frac{\partial^2 f}{\partial w_2 \partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial w_n \partial w_1} & \frac{\partial^2 f}{\partial w_n \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_n^2} \end{pmatrix}$$

- The reason $\nabla^2 f(w) = \nabla(\nabla f(w))$ is a Matrix is since its a gradient of vector function $\nabla f(w)$.

Hessian

- We can similarly compute the Hessian of a Function:

$$\nabla^2 f(w) = \begin{pmatrix} \frac{\partial^2 f}{\partial w_1^2} & \frac{\partial^2 f}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_1 \partial w_n} \\ \frac{\partial^2 f}{\partial w_2 \partial w_1} & \frac{\partial^2 f}{\partial w_2^2} & \cdots & \frac{\partial^2 f}{\partial w_2 \partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial w_n \partial w_1} & \frac{\partial^2 f}{\partial w_n \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_n^2} \end{pmatrix}$$

- The reason $\nabla^2 f(w) = \nabla(\nabla f(w))$ is a Matrix is since its a gradient of vector function $\nabla f(w)$.
- Notice that the Hessian is:

$$\nabla^2 f(w) = \begin{bmatrix} \nabla \frac{\partial f}{\partial w_1} & \nabla \frac{\partial f}{\partial w_2} & \cdots & \nabla \frac{\partial f}{\partial w_n} \end{bmatrix}$$

Hessian

$$\nabla^2 f(w) = \begin{pmatrix} \frac{\partial^2 f}{\partial w_1^2} & \frac{\partial^2 f}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_1 \partial w_n} \\ \frac{\partial^2 f}{\partial w_2 \partial w_1} & \frac{\partial^2 f}{\partial w_2^2} & \cdots & \frac{\partial^2 f}{\partial w_2 \partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial w_n \partial w_1} & \frac{\partial^2 f}{\partial w_n \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_n^2} \end{pmatrix}$$

Compute the Hessian for the Function $L(w_1, w_2) = w_1^2 / w_2$.

Hessian

$$\nabla^2 f(w) = \begin{pmatrix} \frac{\partial^2 f}{\partial w_1^2} & \frac{\partial^2 f}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_1 \partial w_n} \\ \frac{\partial^2 f}{\partial w_2 \partial w_1} & \frac{\partial^2 f}{\partial w_2^2} & \cdots & \frac{\partial^2 f}{\partial w_2 \partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial w_n \partial w_1} & \frac{\partial^2 f}{\partial w_n \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_n^2} \end{pmatrix}$$

Compute the Hessian for the Function $L(w_1, w_2) = w_1^2/w_2$.The Hessian is:

$$\nabla^2 L(w) = \begin{bmatrix} \frac{2}{w_2} & -\frac{2w_1}{w_2^2} \\ -\frac{2w_1}{w_2^2} & \frac{2w_1^2}{w_2^3} \end{bmatrix}$$

Examples: Regularized Logistic Regression

- Lets start with Regularized Logistic Regression. Assume the Labels $y_i \in \{-1, +1\}$.

Examples: Regularized Logistic Regression

- Lets start with Regularized Logistic Regression. Assume the Labels $y_i \in \{-1, +1\}$.
- The objective of Reg Logistic Loss is:

$$L(w) = \lambda/2\|w\|^2 + \sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i)) \tag{1}$$

Examples: Regularized Logistic Regression

- Lets start with Regularized Logistic Regression. Assume the Labels $y_i \in \{-1, +1\}$.
- The objective of Reg Logistic Loss is:

$$L(w) = \lambda/2\|w\|^2 + \sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i)) \tag{1}$$

- Compute the gradient of this Loss?

Examples: Regularized Logistic Regression

- Lets start with Regularized Logistic Regression. Assume the Labels $y_i \in \{-1, +1\}$.
- The objective of Reg Logistic Loss is:

$$L(w) = \lambda/2\|w\|^2 + \sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i)) \tag{1}$$

- Compute the gradient of this Loss?
- Gradient:

$$\begin{aligned}\nabla L(w) =& \lambda w + \sum_{i=1}^{n} \frac{-y_i \exp(-y_i(w^T x_i))}{1 + \exp(-y_i w^T x_i)} x_i \\ =& \lambda w + \sum_{i=1}^{n} \frac{-y_i}{1 + \exp(y_i w^T x_i)} x_i\end{aligned}$$

Examples: Regularized Logistic Regression

- Lets next compute the Hessian.

Examples: Regularized Logistic Regression

- Lets next compute the Hessian.
- Recall the Gradient:

$$\nabla L(w) = \lambda w + \sum_{i=1}^{n} \frac{-y_i}{1 + \exp(y_i w^T x_i)} x_i$$

Examples: Regularized Logistic Regression

- Lets next compute the Hessian.
- Recall the Gradient:

$$\nabla L(w) = \lambda w + \sum_{i=1}^{n} \frac{-y_i}{1 + \exp(y_i w^T x_i)} x_i$$

- You can derive the Hessian as:

$$\nabla^2 L(w) = \lambda I + \sum_{i=1}^{n} \frac{\exp(y_i w^T x_i)}{(1 + \exp(y_i w^T x_i))^2} x_i x_i^T$$

Examples: Regularized Logistic Regression

- Lets next compute the Hessian.
- Recall the Gradient:

$$\nabla L(w) = \lambda w + \sum_{i=1}^{n} \frac{-y_i}{1 + \exp(y_i w^T x_i)} x_i$$

- You can derive the Hessian as:

$$\nabla^2 L(w) = \lambda I + \sum_{i=1}^{n} \frac{\exp(y_i w^T x_i)}{(1 + \exp(y_i w^T x_i))^2} x_i x_i^T$$

- Define $\sigma(z) = 1/(1 + \exp(-z))$. Then its easy to see that:

$$\nabla^2 L(w) = \sigma(y_i w^T x_i)(1 - \sigma(y_i w^T x_i)) x_i x_i^T + \lambda I$$

Numerical Issues

- Lets start with Logistic Regression:
  $L(w) = \lambda/2\|w\|^2 + \sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i))$.

Numerical Issues

- Lets start with Logistic Regression: $L(w) = \lambda/2\|w\|^2 + \sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i))$.
- How would you implement this? Consider this code?

```
import numpy as np
# assume that a vector x and w are defined.
# y is a scalar (either 1 or −1)
L = np.log(1 + np.exp(−y∗x.dot(w))
print(L)
```

Numerical Issues

- Lets start with Logistic Regression:
  $L(w) = \lambda/2\|w\|^2 + \sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i))$.
- How would you implement this? Consider this code?

```
import numpy as np
# assume that a vector x and w are defined.
# y is a scalar (either 1 or -1)
L = np.log(1 + np.exp(-y*x.dot(w))
print(L)
```

- Is there any problem with this? Can there be numerical issues?

Numerical Issues

- Lets start with Logistic Regression: $L(w) = \lambda/2\|w\|^2 + \sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i))$.
- How would you implement this? Consider this code?

```
import numpy as np
# assume that a vector x and w are defined.
# y is a scalar (either 1 or -1)
L = np.log(1 + np.exp(-y*x.dot(w))
print(L)
```

- Is there any problem with this? Can there be numerical issues?
- Consider the following simpler code:

```
import numpy as np
x = -1000
L = np.log(1 + np.exp(-x))
print(L)
```