Optimization in Machine Learning

Lecture 4: Concluding Discrete Optimization Examples (MAP Inference, Feature Selection, Data Subset Selection, *etc.*), Calculus Brushup for Optimization in Machine Learning, Convex Sets and Functions

Ganesh Ramakrishnan

Department of Computer Science
Dept of CSE, IIT Bombay
https://www.cse.iitb.ac.in/~ganesh

January, 2025

Outline of Content for Today

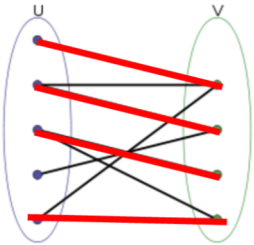
- Continuous Optimization in Machine Learning: Applications include
  1. Supervised Learning [Done]
  2. Principal Component Analysis [Done]
  3. Matrix Completion [Done]
  4. Low Rank and Non Negative Matrix Factorization [Done]
  5. Clustering [Done]
  6. Contextual Bandits and Learning from Logged Data [Done]
- Discrete Optimization in Machine Learning
  1. MAP inference in Probabilistic Models: Ising Models, DPPs [Done]
  2. Feature Subset Selection
  3. Data Partitioning
  4. Data Subset Selection
  5. Data Summarization: Text, Images, Video Summarization
  6. Social networks, Influence Maximization

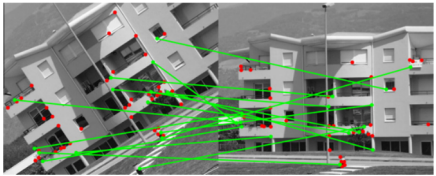
Discrete Optimization in Machine Learning

- MAP inference in Probabilistic Models: Ising Models, DPPs
- Feature Subset Selection
- Data Partitioning
- Data Subset Selection
- Data Summarization: Text, Images, Video Summarization
- Social networks, Influence Maximization
- Natural Language Processing: words, phrases n-grams, syntax trees, semantic structures
- Computer Vision: Image Segmentation, Imag Correspondence
- Genomics and Computational Biology: cell types or assay selection, selecting peptides ar proteins

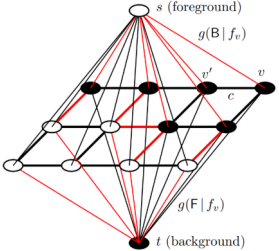
Application 1: Image Segmentation and Correspondence

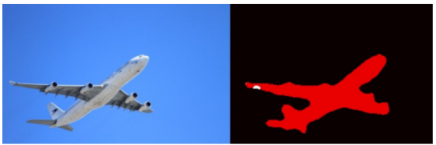
Bipartite Matchings





Minimum Cuts





Source: `https://www.softserveinc.com/en-us/blog/reduce-seismic-data-interpretation-aws-sagemaker`

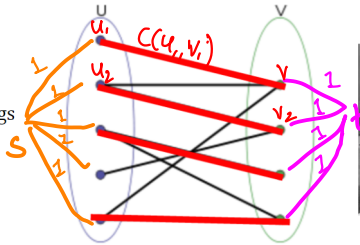
Application 1: Image Segmentation and Correspondence

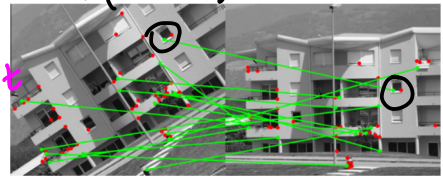
Bipartite Matchings

$$\max_{f \in \mathbb{R}} \sum_{u_i \in U} f_{su_i}$$

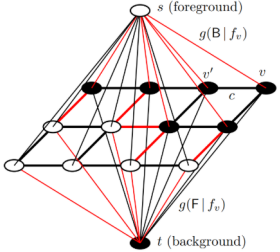
$$s.t \quad \forall p \quad \sum_{(q,p)\in E} f_{qp} = \sum_{(p,r)\in E} f_{pr}$$

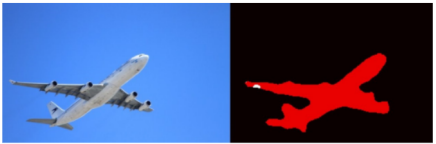
$$0 \le f_{p,q} \le c(p,q)$$





Minimum Cuts

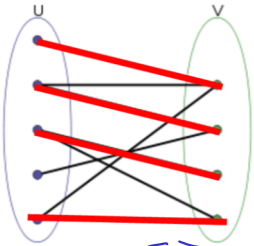


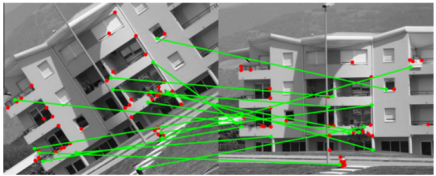


Source: https://www.softserveinc.com/en-us/blog/reduce-seismic-data-interpretation-aws-sagemaker

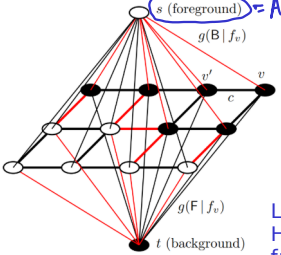
Application 1: Image Segmentation and Correspondence

Bipartite Matchings



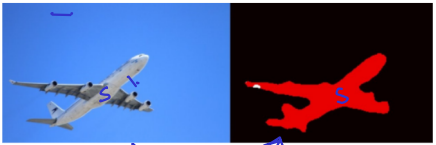


Minimum Cuts



= Aeroplane

c_{uv} = cost = say difference in grayscale/RBG values



Limitations
Homogenity of S can be further accounted for...

$\max_{S \subseteq V} \sum_{(u,v) \in (S,\bar{S})} c(u,v)$

Source: https://www.softserveinc.com/en-us/blog/reduce-seismic-data-interpretation-aws-sagemaker

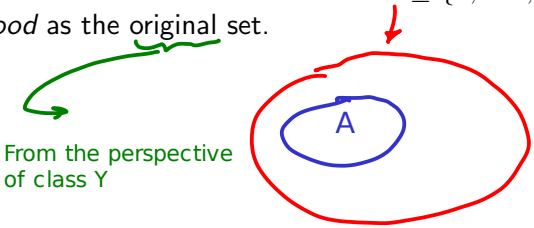
Application 2: Feature Selection

- Data: Given random variables $X_1, X_2, \cdots, X_n$ as features of a given ML task. Denote $I(X_1, X_2)$ as the Mutual Information between variables X_1 and X_2.

[1] https://github.com/decile-team/trust

Application 2: Feature Selection

- Data: Given random variables $X_1, X_2, \cdots, X_n$ as features of a given ML task. Denote $I(X_1, X_2)$ as the Mutual Information between variables X_1 and X_2.
- Goal: Select a subset of features $A \subseteq \{1, \cdots, n\}$ such that the subset of features are *as good* as the original set.



[1] https://github.com/decile-team/trust

Application 2: Feature Selection

- Data: Given random variables $X_1, X_2, \cdots, X_n$ as features of a given ML task. Denote $I(X_1, X_2)$ as the Mutual Information between variables X_1 and X_2.
- Goal: Select a subset of features $A \subseteq \{1, \cdots, n\}$ such that the subset of features are *as good* as the original set.
- Optimization Problem: Maximize the Mutual Information between the set of features and the label Y:

$$\max_{A:|A|\leq k} I(X_A; Y)$$

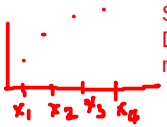
[1] https://github.com/decile-team/trust

Application 2: Feature Selection

- Data: Given random variables $X_1, X_2, \cdots, X_n$ as features of a given ML task. Denote $I(X_1, X_2)$ as the Mutual Information between variables X_1 and X_2.
- Goal: Select a subset of features $A \subseteq \{1, \cdots, n\}$ such that the subset of features are *as good* as the original set.
- Optimization Problem: Maximize the Mutual Information between the set of features and the label Y:

$$\max_{A:|A|\leq k} I(X_A; Y)$$

Concave functions also exhibit diminishing returns



Submodular functions are Discrete functions with diminishing returns

[1] https://github.com/decile-team/trust

Application 2: Feature Selection

- Data: Given random variables $X_1, X_2, \cdots, X_n$ as features of a given ML task. Denote $I(X_1, X_2)$ as the Mutual Information between variables X_1 and X_2.
- Goal: Select a subset of features $A \subseteq \{1, \cdots, n\}$ such that the subset of features are *as good* as the original set.
- Optimization Problem: Maximize the Mutual Information between the set of features and the label Y:

$$\max_{A:|A|\leq k} I(X_A; Y)$$

- We will see in the second part of this course that this is related to the concept of *submodularity*. See TRUST[1], *etc* and our associated publications at AAAI '22, ACL '22, ACL '23, ICDAR '23, ICLR '24, ICDAR '24, ACL '24 *etc*.

[1] https://github.com/decile-team/trust

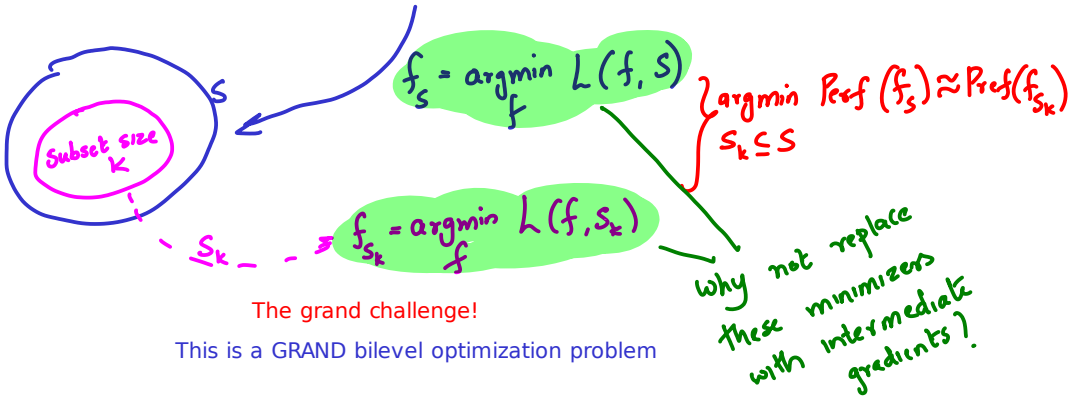
Application 3: Training Data Subset Selection

- Data: Given a training dataset $\{(x_1, y_1), \cdots, (x_n, y_n)\}$ and a budget k.

[2] http://proceedings.mlr.press/v139/killamsetty21a.html
[3] http://proceedings.mlr.press/v139/s21a.html
[4] https://ojs.aaai.org/index.php/AAAI/article/view/16988
[5] https://arxiv.org/abs/2103.00128

Application 3: Training Data Subset Selection

- Data: Given a training dataset $\{(x_1, y_1), \cdots, (x_n, y_n)\}$ and a budget k.

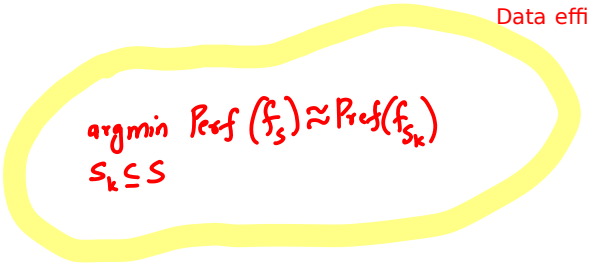


[2] http://proceedings.mlr.press/v139/killamsetty21a.html
[3] http://proceedings.mlr.press/v139/s21a.html
[4] https://ojs.aaai.org/index.php/AAAI/article/view/16988
[5] https://arxiv.org/abs/2103.00128

Application 3: Training Data Subset Selection

- Data: Given a training dataset $\{(x_1, y_1), \cdots, (x_n, y_n)\}$ and a budget k.
- Goal: Select a subset of data-points $A \subseteq \{1, \cdots, n\}$ such that the model trained on the subset of data is as good as the entire dataset. This has become critical for for self-supervised learning such as while pre-training LLMs.

Data efficiency (for effecient training)



A measure of goodness! (See previous slide)

[2] http://proceedings.mlr.press/v139/killamsetty21a.html
[3] http://proceedings.mlr.press/v139/s21a.html
[4] https://ojs.aaai.org/index.php/AAAI/article/view/16988
[5] https://arxiv.org/abs/2103.00128

Application 3: Training Data Subset Selection

- Data: Given a training dataset $\{(x_1, y_1), \cdots, (x_n, y_n)\}$ and a budget k.
- Goal: Select a subset of data-points $A \subseteq \{1, \cdots, n\}$ such that the model trained on the subset of data is as good as the entire dataset. This has become critical for for self-supervised learning such as while pre-training LLMs.
- This setting makes even more sense when the labels are missing on some or all of the given data-points.

[2] http://proceedings.mlr.press/v139/killamsetty21a.html
[3] http://proceedings.mlr.press/v139/s21a.html
[4] https://ojs.aaai.org/index.php/AAAI/article/view/16988
[5] https://arxiv.org/abs/2103.00128

Application 3: Training Data Subset Selection

- Data: Given a training dataset $\{(x_1, y_1), \cdots, (x_n, y_n)\}$ and a budget k.
- Goal: Select a subset of data-points $A \subseteq \{1, \cdots, n\}$ such that the model trained on the subset of data is as good as the entire dataset. This has become critical for for self-supervised learning such as while pre-training LLMs.
- This setting makes even more sense when the labels are missing on some or all of the given data-points. Self-supervised learning and Label efficiency

[2] http://proceedings.mlr.press/v139/killamsetty21a.html
[3] http://proceedings.mlr.press/v139/s21a.html
[4] https://ojs.aaai.org/index.php/AAAI/article/view/16988
[5] https://arxiv.org/abs/2103.00128

Application 3: Training Data Subset Selection

- Data: Given a training dataset $\{(x_1, y_1), \cdots, (x_n, y_n)\}$ and a budget k.
- Goal: Select a subset of data-points $A \subseteq \{1, \cdots, n\}$ such that the model trained on the subset of data is as good as the entire dataset. This has become critical for for self-supervised learning such as while pre-training LLMs.
- This setting makes even more sense when the labels are missing on some or all of the given data-points.
- Optimization Problem: If $D(.)$ denotes the distance (such as KL divergence D_{KL}) between the two dataset (*e.g.* distributions $P(.)$ or gradients). The optimization problem can be cast as:

$$\max_{A:|A|\leq k} -D(X_A, X_V)$$

or max Similarity(X_A,X_V)

[2] http://proceedings.mlr.press/v139/killamsetty21a.html
[3] http://proceedings.mlr.press/v139/s21a.html
[4] https://ojs.aaai.org/index.php/AAAI/article/view/16988
[5] https://arxiv.org/abs/2103.00128

Application 3: Training Data Subset Selection

- Data: Given a training dataset $\{(x_1, y_1), \cdots, (x_n, y_n)\}$ and a budget k.
- Goal: Select a subset of data-points $A \subseteq \{1, \cdots, n\}$ such that the model trained on the subset of data is as good as the entire dataset. This has become critical for for self-supervised learning such as while pre-training LLMs.
- This setting makes even more sense when the labels are missing on some or all of the given data-points.
- Optimization Problem: If $D(.)$ denotes the distance (such as KL divergence D_{KL}) between the two dataset (*e.g.* distributions $P(.)$ or gradients). The optimization problem can be cast as:

$$\max_{A:|A|\leq k} -D(X_A, X_V)$$

- We will see in the second part that this is also related to the concept of *submodularity*. See our papers Grad-Match[2], SELCON[3] and GLISTER[4] and SPEAR[5]

[2] http://proceedings.mlr.press/v139/killamsetty21a.html
[3] http://proceedings.mlr.press/v139/s21a.html
[4] https://ojs.aaai.org/index.php/AAAI/article/view/16988
[5] https://arxiv.org/abs/2103.00128

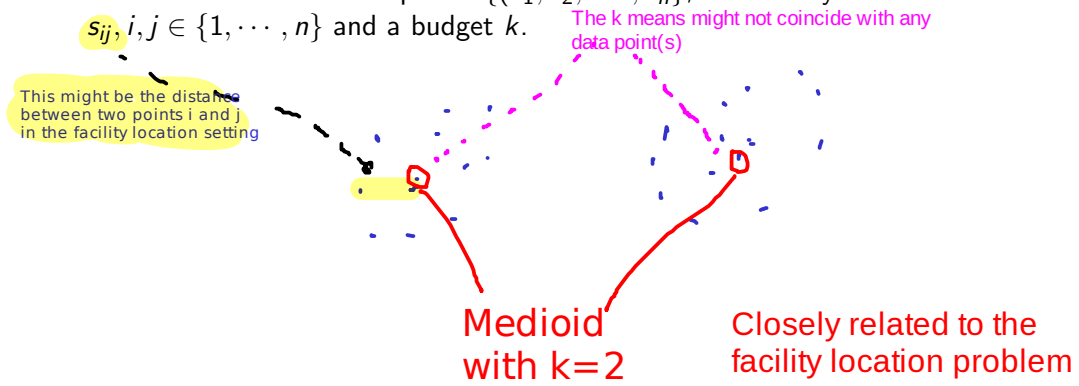
Application 4: k-Mediods Clustering

- Data: Given a set of datapoints $\{(x_1, x_2, \cdots, x_n\}$, a similarity function $s_{ij}, i, j \in \{1, \cdots, n\}$ and a budget k.

[6] https://github.com/decile-team/submodlib

Application 4: k-Mediods Clustering

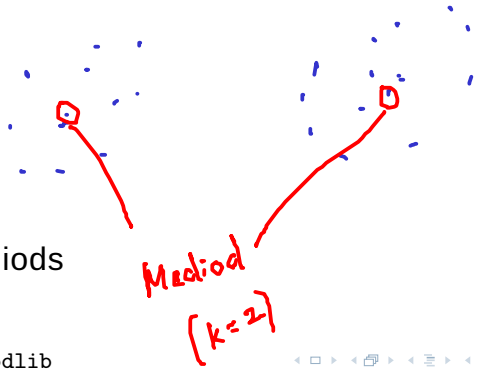
- Data: Given a set of datapoints $\{(x_1, x_2, \cdots, x_n\}$, a similarity function $s_{ij}, i, j \in \{1, \cdots, n\}$ and a budget k.



[6] https://github.com/decile-team/submodlib

Application 4: k-Mediods Clustering

- Data: Given a set of datapoints $\{(x_1, x_2, \cdots, x_n\}$, a similarity function $s_{ij}, i, j \in \{1, \cdots, n\}$ and a budget k.
- Goal: Select a subset of data-points $A \subseteq \{1, \cdots, n\}$ which can act as k-mediods (similar to k-means except that the means are *a part* of the original set of points.



Goal: Identify the k mediods

[6] https://github.com/decile-team/submodlib

Application 4: k-Mediods Clustering

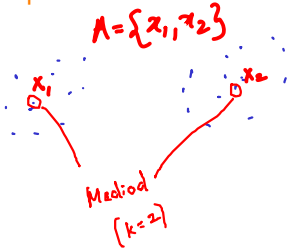
- Data: Given a set of datapoints $\{(x_1, x_2, \cdots, x_n\}$, a similarity function $s_{ij}, i, j \in \{1, \cdots, n\}$ and a budget k.
- Goal: Select a subset of data-points $A \subseteq \{1, \cdots, n\}$ which can act as k-mediods (similar to k-means except that the means are *a part* of the original set of points.
- Optimization Problem: The optimization problem can be cast as:

$$\max_{A:|A|\leq k} \sum_{i=1}^{n} \max_{j \in A} s_{ij}$$

[6] https://github.com/decile-team/submodlib

Application 4: k-Mediods Clustering

- Data: Given a set of datapoints $\{(x_1, x_2, \cdots, x_n\}$, a similarity function $s_{ij}, i, j \in \{1, \cdots, n\}$ and a budget k.
- Goal: Select a subset of data-points $A \subseteq \{1, \cdots, n\}$ which can act as k-mediods (similar to k-means except that the means are *a part* of the original set of points.
- Optimization Problem: The optimization problem can be cast as:



$$\max_{A:|A|\leq k} \sum_{i=1}^{n} \max_{j \in A} s_{ij}$$

2^n

Any point i is assigned to the medoid j which is most similar to it

Search space is the power set

Similarities s_ij can be chosen to be rich, including gradients of loss functions

[6] https://github.com/decile-team/submodlib

Application 4: k-Mediods Clustering

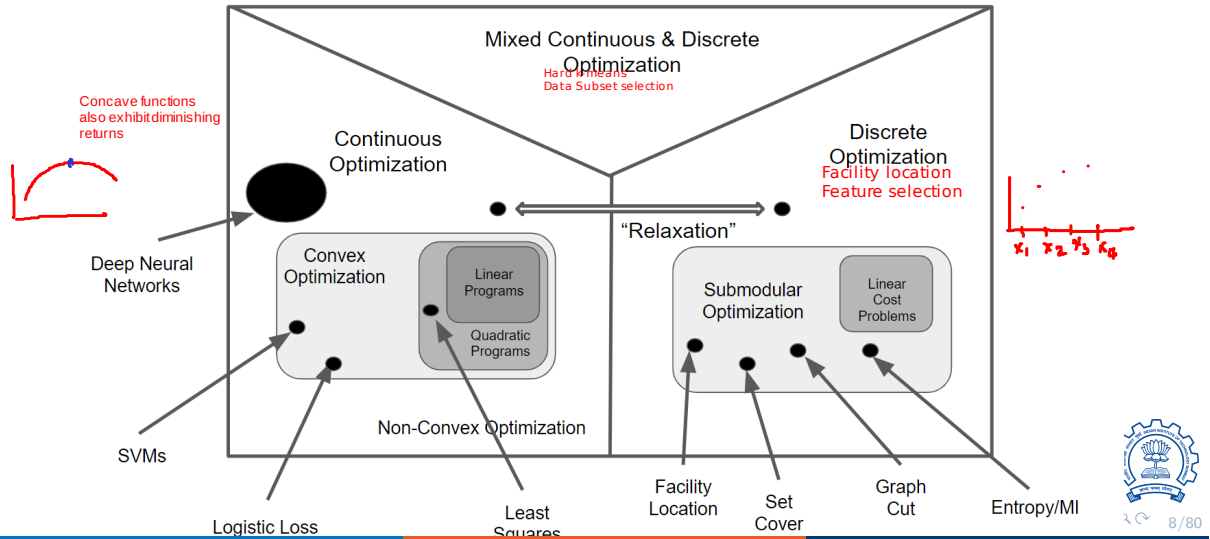
- Data: Given a set of datapoints $\{(x_1, x_2, \cdots, x_n\}$, a similarity function $s_{ij}, i, j \in \{1, \cdots, n\}$ and a budget k.
- Goal: Select a subset of data-points $A \subseteq \{1, \cdots, n\}$ which can act as k-mediods (similar to k-means except that the means are *a part* of the original set of points.
- Optimization Problem: The optimization problem can be cast as:

$$\max_{A:|A| \leq k} \sum_{i=1}^{n} \max_{j \in A} s_{ij}$$

- We will see in the second part of this course that this problem is called *Facility Location* also related to the concept of *submodularity*. See Submodlib[6], and several of our earlier papers.

[6] https://github.com/decile-team/submodlib

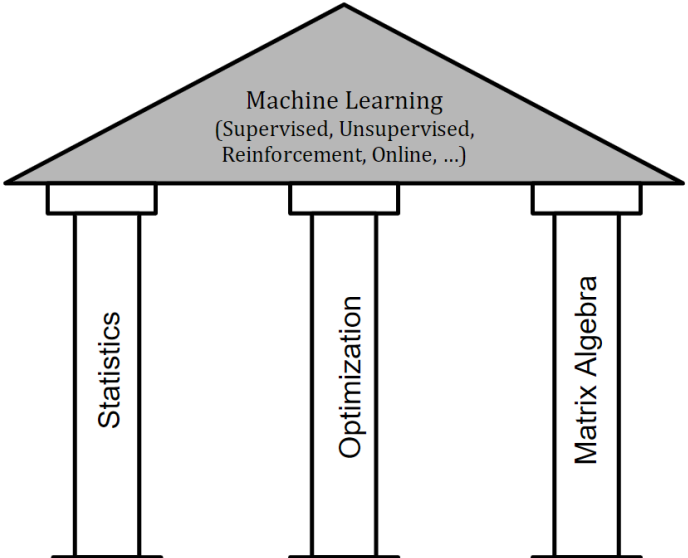
Big Picture: Types of Optimization Problems



Outline

- Summing up so far
- Notation and Basics of Continuous optimization: Vectors, Matrices, Gradients, Hessians, ...
- Deriving Gradients and Hessians for various Loss Functions
- Practical Aspects: Implementing Loss Functions and Gradients in Python.

Summing up: The Road Ahead



Summing up: The Road Ahead

- 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 on in operational settings, 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 scneario.

Summing up: Next Part - 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

Some might be optional reading

Summing up: Subsequent Part - 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

Summing up: Continuous Optimization in Machine Learning

Another classification/bucketing of the optimmization problems we have looked at so far

- 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

Outline

- Doubts on Basics of Continuous Optimization?
- Basics of Convexity: Convex Sets and Convex Functions
- Properties and Examples of Convex functions
- Basic Subgradient Calculus: Subgradients for non-differentiable convex functions
- Understanding the Convexity of Machine Learning Loss Functions
- Convex Optimization Problems

WHAT FOLLOWS IS MATERIAL FOR BRUSHING UP INDIVIDUALLY BY THE STUDENT. WITH RESPECT TO THE FOLLOWING SLIDES WE WILL ONLY DISCUSS DOUBTS/QUESTIONS IN THE CLASS ON THURSDAY 16th January AND THEN GO ONTO TO DISCUSS CONVEXITY AND CONVEX OPTIMIZATION

Notation: Vectors and Matrices

- For the most part through this course, we will use n as the number of training instances and m as the number of features.
- Denote $\mathbf{x}_i \in \mathbf{R}^m$ as a m-dimensional vector (feature vector).
- We shall denote the jth feature of $\mathbf{x}_i$ as $\mathbf{x}_i[j]$.
- y_i is the Label of the ith instance.
- Given two vectors $\mathbf{w}, \mathbf{x} \in \Re^m$, define the inner product $\langle \mathbf{w}, \mathbf{x} \rangle = \sum_{j=1}^{m} \mathbf{x}[j] w[j]$.
- The L1 Norm $\|\mathbf{w}\|_1 = \sum_{i=1}^{m} |\mathbf{w}[i]|$
- The Squared L2 Norm $\|\mathbf{w}\|_2^2 = \sum_{i=1}^{m} |\mathbf{w}[i]|^2$

Matrix-Vector Product

- If $A \in \mathbb{R}^{m \times n}$ and $\mathbf{x} \in \mathbb{R}^n$, we can define $\mathbf{y} = A\mathbf{x}$ where $y \in \mathbb{R}^m$ is a m dimensional vector.
- Matrix vector product is defined as below:

Matrix-Vector Product

- If $A \in \mathbb{R}^{m \times n}$ and $\mathbf{x} \in \mathbb{R}^n$, we can define $\mathbf{y} = A\mathbf{x}$ where $y \in \mathbb{R}^m$ is a m dimensional vector.
- Matrix vector product is defined as below:

$$A\mathbf{x} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix}$$

Matrix-Vector Product Example

For example, if

$$A = \begin{bmatrix} 1 & -1 & 2 \\ 0 & -3 & 1 \end{bmatrix}$$

and $\mathbf{x} = (2, 1, 0)$, then

$$\begin{aligned} A\mathbf{x} &= \begin{bmatrix} 1 & -1 & 2 \\ 0 & -3 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 2 \cdot 1 - 1 \cdot 1 + 0 \cdot 2 \\ 2 \cdot 0 - 1 \cdot 3 + 0 \cdot 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 \\ -3 \end{bmatrix}. \end{aligned}$$

Matrix-Matrix Product

- Matrix product between Matrices A and B is defined only when the number of columns in A equals number of rows in B.
- If A is a $m \times n$ Matrix, B is a $n \times p$ Matrix, the product Matrix $C = AB$ is a $m \times p$ Matrix.
- We can view the Matrix-Matrix product as stacked Matrix-vector products by viewing B as a set of p vectors, each of dimension, $n \times 1$ stacked up.

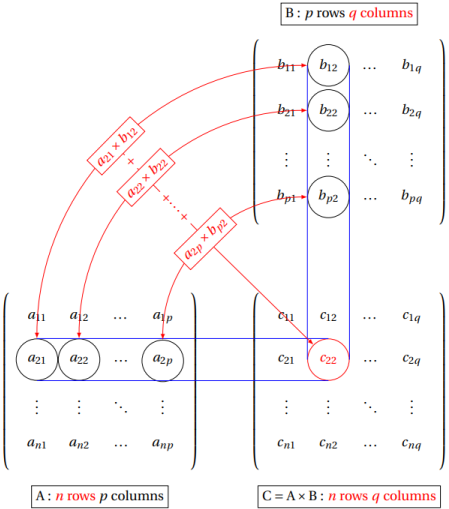
$$\begin{bmatrix} b_{11} & b_{12} & \dots & b_{1p} \\ b_{21} & b_{22} & \dots & b_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{np} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} b_{11} \\ b_{21} \\ \vdots \\ b_{n1} \end{bmatrix} \begin{bmatrix} b_{12} \\ b_{22} \\ \vdots \\ b_{n2} \end{bmatrix} \dots \begin{bmatrix} b_{1p} \\ b_{2p} \\ \vdots \\ b_{np} \end{bmatrix} \end{bmatrix}$$

Matrix-Matrix Product

$$\begin{bmatrix} b_{11} & b_{12} & \dots & b_{1p} \\ b_{21} & b_{22} & \dots & b_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{np} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} b_{11} \\ b_{21} \\ \vdots \\ b_{n1} \end{bmatrix} \begin{bmatrix} b_{12} \\ b_{22} \\ \vdots \\ b_{n2} \end{bmatrix} \cdots \begin{bmatrix} b_{1p} \\ b_{2p} \\ \vdots \\ b_{np} \end{bmatrix} \end{bmatrix}$$

- We can view the Matrix-Matrix product as stacked Matrix-vector products by viewing B as a set of p vectors (each of dimension $n \times 1$) lined up.
- Let $b_1, \cdots, b_p$ be these p column vectors.
- Note that $c_i = Ab_i$ is a $m \times 1$ column vector.
- Hence $AB = [Ab_1 \ Ab_2 \ \cdots \ Ab_p] = [c_1 \ \cdots \ c_p] = C$ is a $m \times p$ Matrix

Matrix-Matrix Product Illustration



Matrix-Matrix Product Example

Compute BC, where

$$B = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad \text{and} \quad C = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}.$$

$$\begin{aligned} BC &= \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \\ &= \begin{bmatrix} 1 \cdot 1 + 2 \cdot 3 + 3 \cdot 5 & 1 \cdot 2 + 2 \cdot 4 + 3 \cdot 6 \\ 4 \cdot 1 + 5 \cdot 3 + 6 \cdot 5 & 4 \cdot 2 + 5 \cdot 4 + 6 \cdot 6 \end{bmatrix} \\ &= \begin{bmatrix} 22 & 28 \\ 49 & 64 \end{bmatrix} \end{aligned}$$

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:
- 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 $x > 0$ 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.
- 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 $w = [w_1 \ \cdots \ w_m]^T$.
- 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:

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

- 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 $L(w_1, w_2) = w_1^2/w_2$.
- 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

- Let us consider 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.
- Recall the Gradient:

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

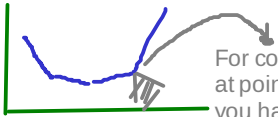
- You can derive the Hessian as:

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

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

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

Theoretical Exercise



Derive the Gradient and Hessian for the following Loss Functions:

- (Only Gradient) Hinge Loss/SVMs: $L(w) = \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\}$. Here $y_i \in \{-1, +1\}$.
- (Gradient and Hessian) Least Squares Loss: $L(w) = \sum_{i=1}^{n} (y_i - w^T x_i)^2$. Here $y_i \in \mathbb{R}$.
- (Only Gradient) Simple 2 Layer Function: $L(w) = \sum_{i=1}^{n} (y_i - \max(0, w^T x_i + b))^2$. Here $y_i \in \mathbb{R}$.
- (Gradient and Hessian): SoftMax Function: $L(w) = \sum_{i=1}^{n} r_i / p_i \frac{\exp(w^T x_i^{a_i})}{\sum_{j=1}^{k} \exp(w^T x_i^j)}$.

Outline

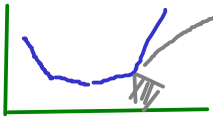
- Homework Exercises Discussion
- Basics of Continuous Optimization
- Basics of Convexity: Convex Sets and Convex Functions
- Properties and Examples of Convex functions
- Basic Subgradient Calculus: Subgradients for non-differentiable convex functions
- Understanding the Convexity of Machine Learning Loss Functions
- Convex Optimization Problems

Homework: Exercises

Derive the Gradient and Hessian for the following Loss Functions:

- (Only Gradient) Hinge Loss/SVMs: $L(w) = \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\}$. Here $y_i \in \{-1, +1\}$.
- (Gradient and Hessian) Least Squares Loss: $L(w) = \sum_{i=1}^{n} (y_i - w^T x_i)^2$. Here $y_i \in \mathbb{R}$.
- (Only Gradient) Simple 2 Layer Function: $L(w) = \sum_{i=1}^{n} (y_i - \max(0, w^T x_i + b))^2$. Here $y_i \in \mathbb{R}$.
- (Gradient and Hessian): SoftMax Function: $L(w) = \sum_{i=1}^{n} r_i / p_i \frac{\exp(w^T x_i^{a_i})}{\sum_{j=1}^{k} \exp(w^T x_i^j)}$.

Homework: Exercises



For convex functions,
at points of non-differentiability
you have sudifferentials

Derive the Gradient and Hessian for the following Loss Functions:

- (Only Gradient) Hinge Loss/SVMs: $L(w) = \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\}$. Here $y_i \in \{-1, +1\}$.
- (Gradient and Hessian) Least Squares Loss: $L(w) = \sum_{i=1}^{n} (y_i - w^T x_i)^2$. Here $y_i \in \mathbb{R}$.
- (Only Gradient) Simple 2 Layer Function: $L(w) = \sum_{i=1}^{n} (y_i - \max(0, w^T x_i + b))^2$. Here $y_i \in \mathbb{R}$.
- (Gradient and Hessian): SoftMax Function: $L(w) = \sum_{i=1}^{n} r_i / p_i \frac{\exp(w^T x_i^{a_i})}{\sum_{j=1}^{k} \exp(w^T x_i^j)}$.

Homework: Exercises

Derive the Gradient and Hessian for the following Loss Functions:

- (Only Gradient) Hinge Loss/SVMs: $L(w) = \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\}$. Here $y_i \in \{-1, +1\}$.
- (Gradient and Hessian) Least Squares Loss: $L(w) = \sum_{i=1}^{n} (y_i - w^T x_i)^2$. Here $y_i \in \mathbb{R}$.
- (Only Gradient) Simple 2 Layer Function: $L(w) = \sum_{i=1}^{n} (y_i - \max(0, w^T x_i + b))^2$. Here $y_i \in \mathbb{R}$.
- (Gradient and Hessian): SoftMax Function: $L(w) = \sum_{i=1}^{n} r_i / p_i \frac{\exp(w^T x_i^{a_i})}{\sum_{j=1}^{k} \exp(w^T x_i^j)}$.

Example: Regularized Logistic Regression

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

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

- Compute the gradient of this Loss?
- Gradient:

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

Example: Regularized Logistic Regression

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

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

- Compute the gradient of this Loss?

Example: Regularized Logistic Regression

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

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

- Compute the gradient of this Loss?

$$\frac{\partial\left(-\sum_j (w_j x_{ij}) y_i\right)}{\partial w_j} = -y_i x_{ij}$$

$$\left[\frac{\partial L}{\partial w_j} = \sum_i \frac{-y_i x_{ij} \exp(\cdots)}{1+\exp(\cdots)}\right]$$

Example: Regularized Logistic Regression

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

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

- Compute the gradient of this Loss?
- Gradient:

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

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

Example: Regularized Logistic Regression

- Let us 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:

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

Example: Regularized Logistic Regression

- Let us next compute the Hessian.
- Recall the Gradient:

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

- You can derive the Hessian as:

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

$\frac{\partial \text{comp}_i}{\partial w_j}$

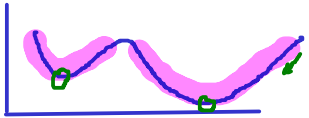
- Define $\sigma(z) = 1/(1 + \exp(-z))$. Then its easy to see that:

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

Next: Convexity and Motivation

- Why Convexity: Every function has convex regions
- Even if a function is non-convex, convergence of algorithms is generally in terms of convergence in convex regions of the function

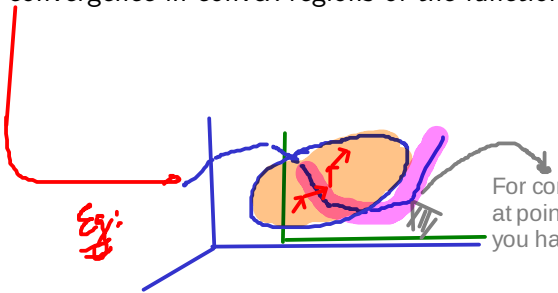
Next: Convexity and Motivation



Most deep learning models (compositions of activation units such as sigmoid) exhibit such losses

- Why Convexity: Every function has convex regions
- Even if a function is non-convex, convergence of algorithms is generally in terms of convergence in convex regions of the function

Especially important for projection/Frank Wolfe/interior point/Lagrange multiplier based methods



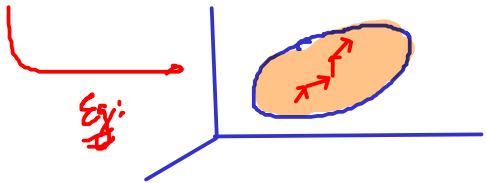
For convex regions of functions, at points of non-differentiability you have sudifferentials/subgradients

Next: Convexity and Motivation

- Why Convexity: Every function has convex regions
- Even if a function is non-convex, convergence of algorithms is generally in terms of convergence in convex regions of the function
- Further, the domain/constraints of optimization functions we discuss will be typically convex sets so that algorithm based updates are guaranteed to lie within the set

Next: Convexity and Motivation

- Why Convexity: Every function has convex regions
- Even if a function is non-convex, convergence of algorithms is generally in terms of convergence in convex regions of the function
- Further, the domain/constraints of optimization functions we discuss will be typically convex sets so that algorithm based updates are guaranteed to lie within the set



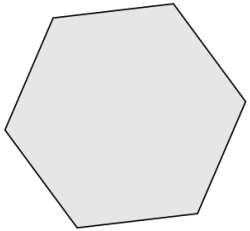
Especially important for projection/Frank Wolfe/interior point/Lagrange multiplier based methods

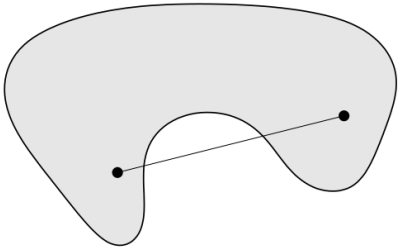
Next: Convexity and Motivation

- Why Convexity: Every function has convex regions
- Even if a function is non-convex, convergence of algorithms is generally in terms of convergence in convex regions of the function
- Further, the domain/constraints of optimization functions we discuss will be typically convex sets so that algorithm based updates are guaranteed to lie within the set
- We will begin with Convex Sets and quickly move on to Convex functions

Convex Sets

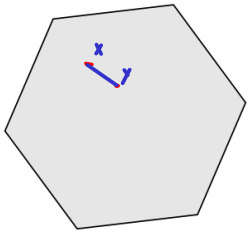
A set C is a **convex set** if the line segment between any two points of C lies in C, i.e. if for any $x, y \in C$ and for any $0 < \lambda < 1$, we have that $\lambda x + (1 - \lambda)y \in C$.

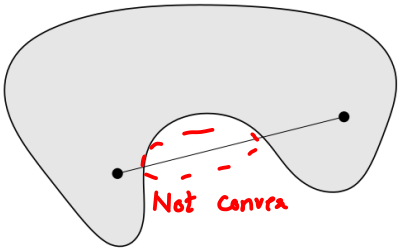




Convex Sets

A set C is a **convex set** if the line segment between any two points of C lies in C, i.e. if for any $x, y \in C$ and for any $0 < \lambda < 1$, we have that $\lambda x + (1 - \lambda)y \in C$.





Properties of Convex Sets

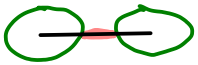
- Intersections of Convex Sets are Convex. Let $C_1, \cdots, C_k$ be convex sets, then $\cap_{i=1}^{k} C_i$ is convex.
- Is the union of convex sets convex?
- Projections onto convex sets are unique (and often efficient to compute).

$$P_C(x) = \text{argmin}_{\mathbf{y} \in C} \|\mathbf{y} - \mathbf{x}\|$$

- Examples of Convex Sets:
  - $C = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| \leq k\}$
  - $C = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{w}^T\mathbf{x} \leq k\}$
  - Given a convex function f, the associated set $C_f = \{\mathbf{x} \in \mathbb{R}^n : f(\mathbf{x}) \leq k\}$ is convex.

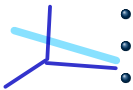
Properties of Convex Sets

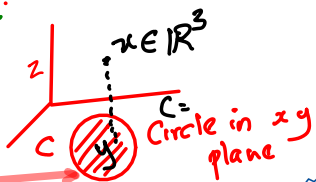
- Intersections of Convex Sets are Convex. Let $C_1, \cdots, C_k$ be convex sets, then $\cap_{i=1}^{k} C_i$ is convex.
- Is the union of convex sets convex?



- Projections onto convex sets are unique (and often efficient to compute).

$$P_C(x) = \text{argmin}_{\mathbf{y} \in C} \|\mathbf{y} - \mathbf{x}\|$$





- Examples of Convex Sets:
  - $C = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| \leq k\}$
  - $C = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{w}^T\mathbf{x} \leq k\}$
  - Given a convex function f, the associated set $C_f = \{\mathbf{x} \in \mathbb{R}^n : f(\mathbf{x}) \leq k\}$ is convex.

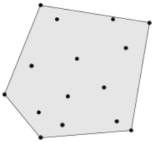
Convex combination and convex hull

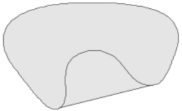
- **Convex combination** of points $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k$ is any point $\mathbf{x}$ of the form

$$\mathbf{x} = \theta_1 \mathbf{x}_1 + \theta_2 \mathbf{x}_2 + ... + \theta_k \mathbf{x}_k = conv(\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k\})$$

$$\text{with} \quad \theta_1 + \theta_2 + ... + \theta_k = 1, \theta_i \geq 0.$$

- **Convex hull or conv(S)** is the set of all convex combinations of point in the set S.





- Should S be always convex?
- What about the convexity of conv(S)?

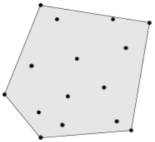
Convex combination and convex hull

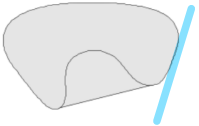
- **Convex combination** of points $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k$ is any point $\mathbf{x}$ of the form

$$\mathbf{x} = \theta_1\mathbf{x}_1 + \theta_2\mathbf{x}_2 + ... + \theta_k\mathbf{x}_k = conv(\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k\})$$

$$\text{with} \quad \theta_1 + \theta_2 + ... + \theta_k = 1, \theta_i \geq 0.$$

- **Convex hull or conv(S)** is the set of all convex combinations of point in the set S.





- Should S be always convex?
- What about the convexity of conv(S)?

H/W: Is there a notion of "Supporting hyperplane" that are characteristic to convex sets and not found in non-convex sets?

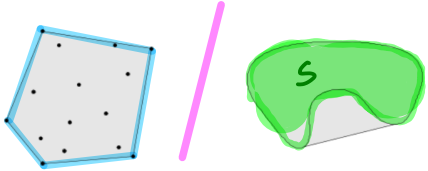
Convex combination and convex hull

- **Convex combination** of points $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k$ is any point $\mathbf{x}$ of the form

$$\mathbf{x} = \theta_1\mathbf{x}_1 + \theta_2\mathbf{x}_2 + ... + \theta_k\mathbf{x}_k = conv(\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k\})$$

$$\text{with} \quad \theta_1 + \theta_2 + ... + \theta_k = 1, \theta_i \geq 0.$$

- **Convex hull or conv(S)** is the set of all convex combinations of point in the set S.



- Should S be always convex? Homework: S is given to you! The S above is connected but not convex. Hence no need for S to be convex
- What about the convexity of conv(S)? YES

MANDATORY HOMEWORK: Please verify! $\lambda_1 x_1 + (1-\lambda_1) x_2$

Verify that by substituting for x_1 and x_2 in terms of y_1, y_2, z_1 and z_2 the new combination is still convex in y_1, y_2, z_1 and z_2

Assume: $y_1, y_2, z_1, z_2 \in S$ & $x_1, x_2 \in$ conv(S)

$\theta_1 y_1 + (1-\theta_1) y_2 \quad \theta_2 z_1 + (1-\theta_2) z_2$

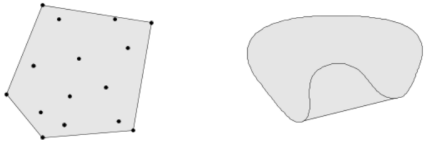
Convex combination and convex hull

- **Convex combination** of points $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k$ is any point $\mathbf{x}$ of the form

$$\mathbf{x} = \theta_1 \mathbf{x}_1 + \theta_2 \mathbf{x}_2 + ... + \theta_k \mathbf{x}_k = conv(\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k\})$$

$$\text{with} \quad \theta_1 + \theta_2 + ... + \theta_k = 1, \theta_i \geq 0.$$

- **Convex hull or conv(S)** is the set of all convex combinations of point in the set S.



- Should S be always convex? No.
- What about the convexity of conv(S)? It's always convex.

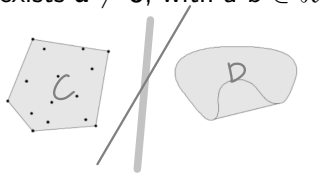
SHT: Separating hyperplane theorem (a fundamental theorem) [Homework solution]

If $\mathcal{C}$ and $\mathcal{D}$ are disjoint convex sets, *i.e.*, $\mathcal{C} \cap \mathcal{D} = \phi$, then there exists $\mathbf{a} \neq \mathbf{0}$, with a $b \in \Re$ such that
$\mathbf{a}^T\mathbf{x} \leq \mathbf{b}$ for $\mathbf{x} \in \mathcal{C}$,
$\mathbf{a}^T\mathbf{x} \geq \mathbf{b}$ for $\mathbf{x} \in \mathcal{D}$.
That is, the hyperplane $\left\{\mathbf{x}|\mathbf{a}^T\mathbf{x} = \mathbf{b}\right\}$ separates $\mathcal{C}$ and $\mathcal{D}$.

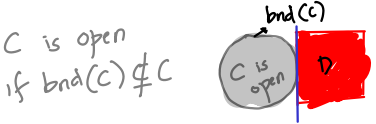
- The seperating hyperplane need not be unique though.
- Strict separation requires additional assumptions (e.g., C is closed, D is a singleton).

SHT: Separating hyperplane theorem (a fundamental theorem) [Homework solution]

If $\mathcal{C}$ and $\mathcal{D}$ are disjoint convex sets, *i.e.*, $\mathcal{C} \cap \mathcal{D} = \phi$, then there exists $\mathbf{a} \neq \mathbf{0}$, with a $b \in \Re$ such that
$\mathbf{a}^T\mathbf{x} \leq \mathbf{b}$ for $\mathbf{x} \in \mathcal{C}$,
$\mathbf{a}^T\mathbf{x} \geq \mathbf{b}$ for $\mathbf{x} \in \mathcal{D}$.
That is, the hyperplane $\{\mathbf{x}|\mathbf{a}^T\mathbf{x} = \mathbf{b}\}$ separates $\mathcal{C}$ and $\mathcal{D}$.



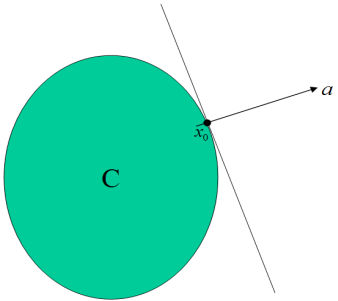
- The seperating hyperplane need not be unique though.
- Strict separation requires additional assumptions (e.g., C is closed, D is a singleton).



Supporting hyperplane theorem (consequence of separating hyperplane theorem) [Homework solution]

Supporting hyperplane to set $\mathcal{C}$ at boundary point $\mathbf{x}_o$:

- $\{\mathbf{x} | \mathbf{a}^T\mathbf{x} = \mathbf{a}^T\mathbf{x}_o\}$
- where $\mathbf{a} \neq 0$ and $\mathbf{a}^T\mathbf{x} \leq \mathbf{a}^T\mathbf{x}_o$ for all $\mathbf{x} \in \mathcal{C}$

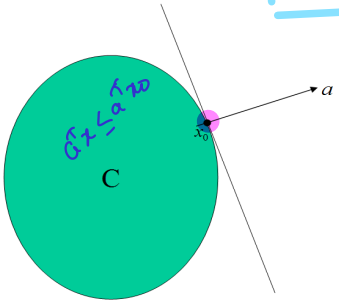


Supporting hyperplane theorem: if $\mathcal{C}$ is convex, then there exists a supporting hyperplane at every boundary point of C.

Supporting hyperplane theorem (consequence of separating hyperplane theorem) [Homework solution]

Supporting hyperplane to set $\mathcal{C}$ at boundary point $\mathbf{x}_o$:

- $\{\mathbf{x} | \mathbf{a}^T \mathbf{x} = \mathbf{a}^T \mathbf{x}_o\}$
- where $\mathbf{a} \neq 0$ and $\mathbf{a}^T \mathbf{x} \leq \mathbf{a}^T \mathbf{x}_o$ for all $\mathbf{x} \in \mathcal{C}$



Supporting hyperplane theorem: if $\mathcal{C}$ is convex, then there exists a supporting hyperplane at every boundary point of C.