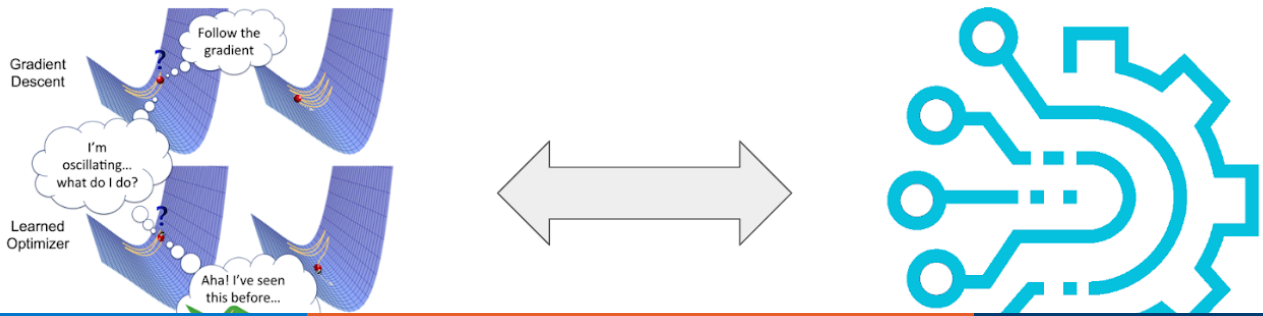
CS7301: Advanced Topics in Optimization for Machine Learning

Lecture 1.1: Introduction and Course Overview

Rishabh Iyer

Department of Computer Science
University of Texas, Dallas
https://github.com/rishabhk108/AdvancedOptML

January 22, 2020



Course Outline

- Why take this course?
- Prerequisites
- Week by Week Course plan
- Course Logistics
- Basics of Linear Algebra
- Applications of Optimization in Machine Learning
- Basics of Calculus: Derivatives, Gradients

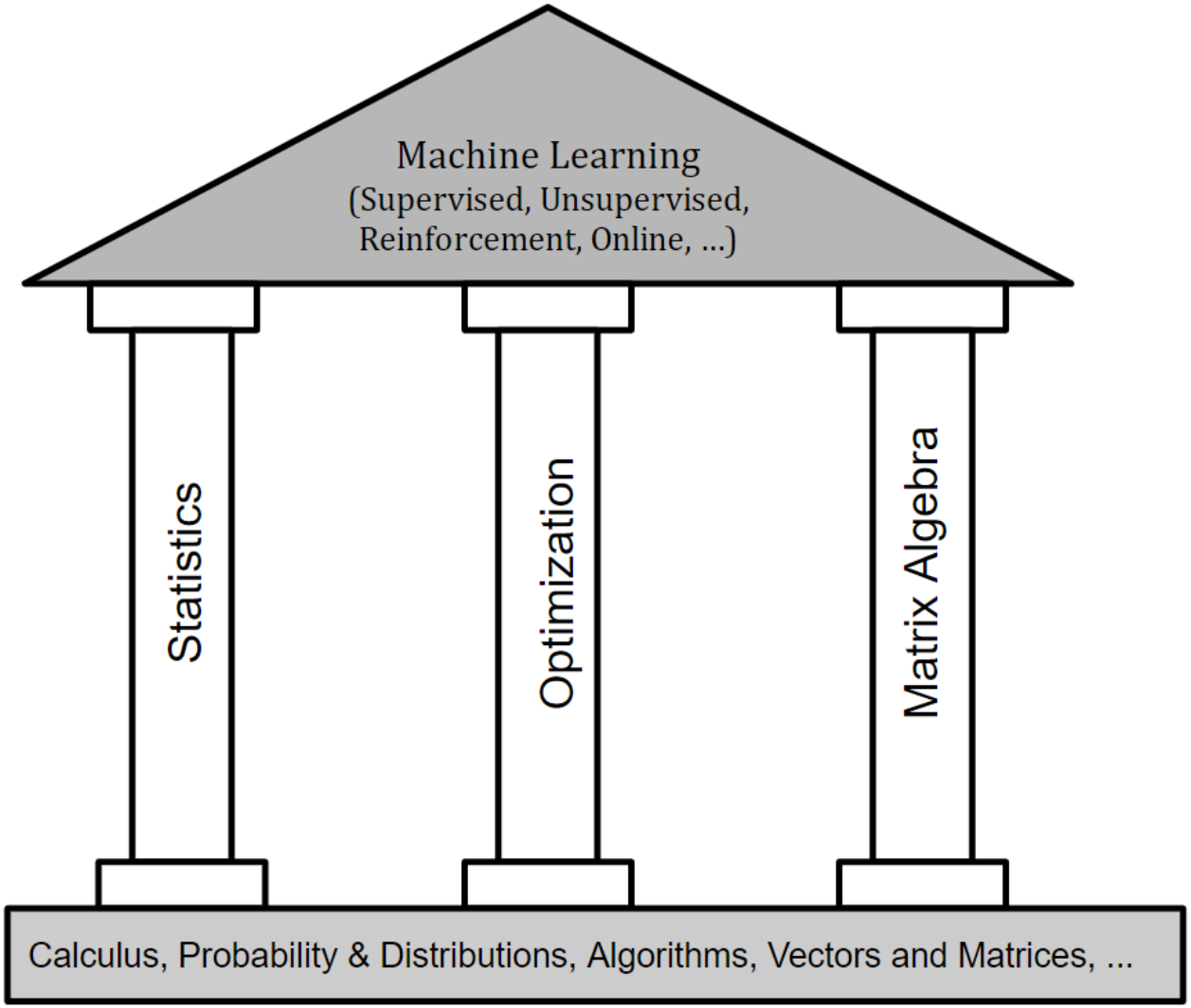
Why take this Course?

Optimization is everywhere: Big Data and Machine Learning, Scheduling and Planning, Operations Research, control theory, data analysis, simulations, almost all technology we use, search engines, computers/laptops, smart-phones, hardware/software of all kinds, ...

- Mathematical Modeling:
  - defining and modeling the problem
- Computational Optimization:
  - Algorithms to solve these optimization problems optimally or near optimally.

UT DALLAS

Why take this Course?



Logistics

- Class is every Friday, 12pm - 2:45pm
- Venue: Online on MS Teams
- Office Hours: Fridays before class (11pm - 12pm).
- Additional office hours by appointment.
- TA: TBD

Parts I and II of this course

Applications of Optimization in Machine Learning

- Continuous Optimization
- Discrete Optimization

Part III of this course

Applications of Optimization in Machine Learning

- Continuous Optimization:
  - Continuous Optimization often appears as *relaxations* of risk/error minimization problem. The *Learning* problem in many parametrized models (whether supervised, semi-supervised, unsupervised, or reinforcement learning) involves **Continuous Optimization**.
- Discrete Optimization:
  - Discrete Optimization occurs in Inference problems in structured spaces, certain learning problems and auxilliary problems such as Feature Selection, Data Subset Selection, Data Summarization, Neural Architechture Search etc.

UT DALLAS

Part I: Continuous Optimization

- Week 1: Logistics, Outline of this Course, Convex Optimization: Basics & Definitions, Continuous Optimization in ML
- Week 2: Gradient Descent and Family (Grad Descent & Line Search, Accelerated Gradient Descent, Projected GD, Proximal GD)
- Week 3: Second Order Methods, Stochastic Gradient and Family
- Week 4: Non-Convex Optimization, Duality, Survey of recent optimization algorithms.

UT DALLAS

Part II: Discrete Optimization

- Week 5: Submodular Optimization: Basics, Definitions, Properties, and Examples. Submodular Information Measures, Probabilistic Submodular Functions, and Determinantal Point Processes
- Week 6: Submodular Maximization Variants, Submodular Set Cover, Approximate submodularity. Algorithms under different constraints and monotone/non-monotone settings. Also, distributed and streaming algorithms.
- Week 7: Submodular Minimization, Continuous Extensions of Submodular Functions, Submodular Polyhedra
- Week 8: Other Optimization Problems: DS Optimization, Submodular Constraints, Continuous Submodularity, Multisubmodularity

UT DALLAS

Part III: Optimization in ML

- Week 9, 10: Applications of Continuous Optimization: Non-convex Optimization, Supervised and Deep Learning, Meta-Learning, Semi-supervised Learning, Bi-Level Optimization and Data Reweighting
- Week 11, 12: Applications of Discrete Optimization: Feature Selection, Data Subset Selection and Core-Sets, Data Partitioning, Active Learning, Subset Selection, Data Summarization

UT DALLAS

Philosophy of this Course

- The spirit of this course is best summarized by the quote of Thomas Cover: *Theory is only the first term of the Taylor's series of Practice*
- This course will focus on mainly the algorithmic aspects of optimization (both continuous and discrete optimization) and not so much on the modeling.
- Give a flavor of the proofs and proof techniques but will try to not make this course heavily theoretical.
- Focus extensively on implementation aspects and as a part of assignments, we will implement many ML loss functions and algorithms.

Prerequisites

- Calculus Courses
- Basic Linear Algebra: Matrices, Vectors & Matrix Algebra
- Machine Learning (either an undergraduate or graduate ML course)
- Algorithms course (either in undergraduate or graduate version)
- Bonus: Linear Programming or Numerical Optimization

UT DALLAS

How is this course different from Machine Learning or Convex Optimization

- Many of you have taken the Machine learning course at UTD.
- Similarly, many Universities offer a Convex Optimization course
- Convex Optimization courses are mostly theoretical and focus heavily on the algorithmic/theoretical aspects
- This course will focus:
  - More on the *optimization algorithms* and not so much on the modeling and statistical aspects
  - Machine learning as an application domain and also emphasize the practical/implementation of algorithms,
  - A flavor of the proof techniques without getting into too many details
- One of the first courses to provide an overview of the entire spectrum of optimization (discrete & continuous) for ML.

Relevant Books for this Course

- Convex Optimization: Algorithms and Complexity, by Sébastien Bubeck
- Convex Optimization, Stephen Boyd and Lieven Vandenberghe
- Introductory Lectures on Convex Optimization, Yurii Nesterov
- A Course in Combinatorial Optimization, Alexander Schrijver
- Learning with Submodular Functions: A Convex Optimization Perspective, Francis Bach
- Zhang, Lipton, Li and Smola, Dive into Deep Learning (`http://d2l.ai/`)
- Schrijver, Alexander, Combinatorial optimization: polyhedra and efficiency, Vol. 24. Springer Science & Business Media, 2003.
- Fujishige, Satoru. Submodular functions and optimization. Vol. 58. Elsevier, 2005.
- A few recent papers which we will discuss towards the end of this course.

Evaluation

- Class Participation (10%):
  - Interaction, asking questions, answering questions
- Paper Presentation: (30%):
  - Everyone will pick 2 papers based on the topics discussed in this course (continuous optimization, discrete optimization, applications of optimization) and present a careful review of it (pros of the papers, cons, open questions, and bonus: any thoughts on improving the algorithms/theory).
  - Please choose topics/papers which I have not already covered in this course but are relevant.
- Project: (50%): Take up a project related to this course. It can be either implementation based or theoretical. As a part of the project, there should be something new (either an implementation from scratch or a new theorem).

Continuous Optimization in Machine Learning

- Continuous Optimization often appears as *relaxations* of empirical risk minimization problems.
- 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

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 $x_i \in \mathrm{R}^m$ as a m-dimensional vector (feature vector).
- We shall denote the jth feature of x_i as $x_i[j]$.
- y_i is the Label as the ith instance.
- Given two vectors $w, x \in \mathrm{R}^m$, define the inner product $\langle w, x \rangle = \sum_{j=1}^{m} x[j]w[j]$.
- The L1 Norm $\|w\|_1 = \sum_{i=1}^{m} |w[i]|$
- The Squared L2 Norm $\|w\|_2^2 = \sum_{i=1}^{m} |w[i]|^2$
- Sometimes we shall refer to L2 as the default norm: $\|w\| = \|w\|_2$.

What is a Norm

A function $f : \mathbb{R}^n \Rightarrow \mathbb{R}_+$ is a norm if:

- $f(x) \geq 0, \forall x$
- $f(x+y) \leq f(x) + f(y)$ [Triangle Inequality]
- $f(\alpha x) = \alpha f(x)$
- $f(x) = 0 \Rightarrow x = 0$

Can you make sure the L1 and L2 norms defined above satisfy all four conditions?

Matrix-Vector Product

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

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

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

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

DALLAS

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 $n \times 1$ vectors lined up.

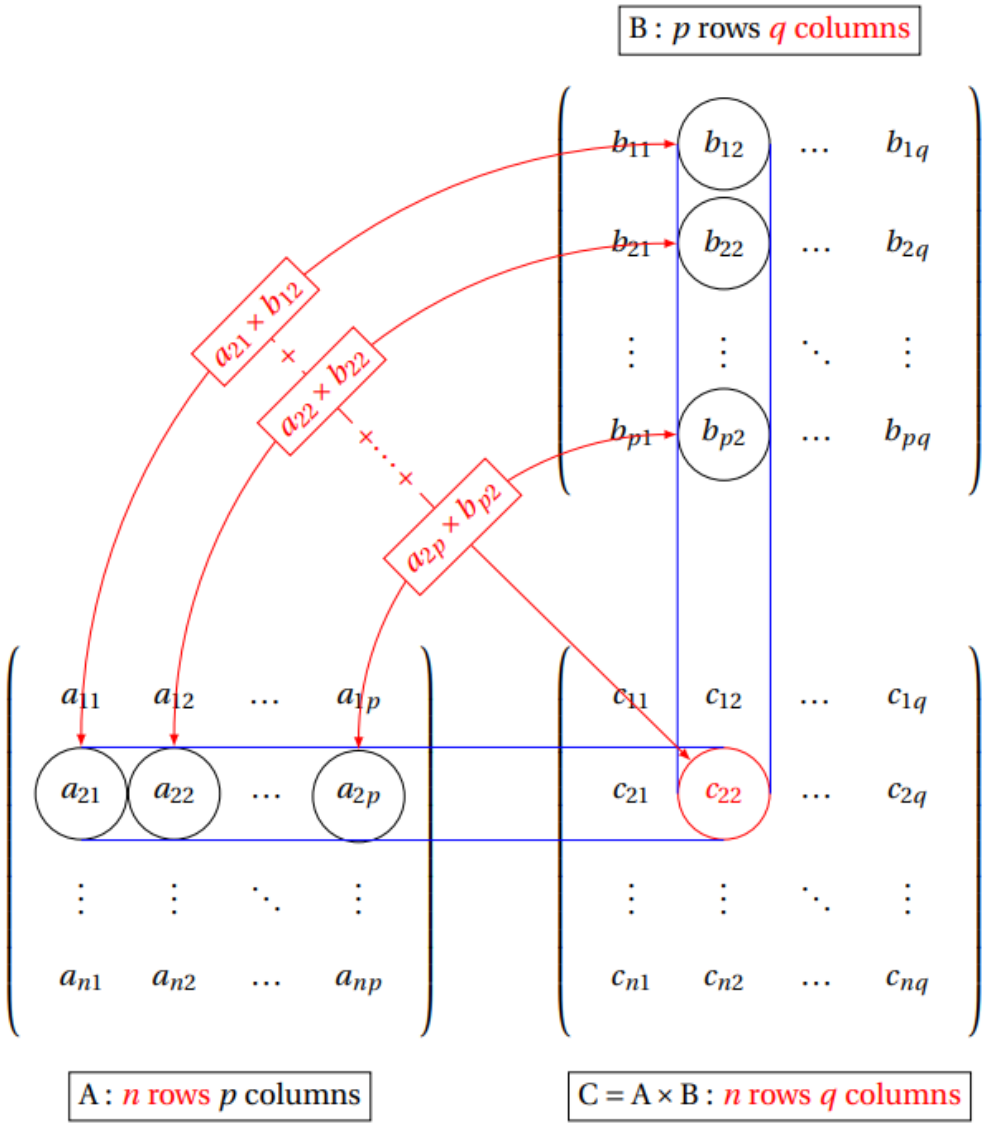
$$\begin{bmatrix} b_{11} & b_{12} & \dots & b_{1p} \\ b_{21} & b_{22} & \dots & b_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{np} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} b_{11} \\ b_{21} \\ \vdots \\ b_{n1} \end{bmatrix} \begin{bmatrix} b_{12} \\ b_{22} \\ \vdots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

Matrix-Matrix Product

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

- We can view the Matrix-Matrix product as stacked Matrix-vector products by viewing B as a set of p $n \times 1$ vectors 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

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

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

Application 1: Supervised Learning

- Data: Given training examples $\{(x_1, y_1), \cdots, x_n, y_n\}$ where $x_i \in \mathrm{R}^m$ is the feature vectors and y_i is the label.
- Applications: Several different models depending on the applications:
  - Email Spam Filtering: Features are words, phrases, regexps in the email, Label is "+1" for Spam, "0" for Not Spam.
  - Handwritten Digit Recognition: Features are Images of Images, Label is the Digit (say between "0" to "9").
  - Housing price Prediction: Features are House properties (square footage, # Bedrooms/Bathrooms, Location, ...) and Label is the Cost (continuous variable).

Supervised Learning: Modeling

- Data: Given training examples $\{(x_1, y_1), \cdots, (x_n, y_n)\}$ where $x_i \in \mathrm{R}^m$ is the feature vectors and y_i is the label.
- Model: Denote the Model by $F_\theta(x)$ with θ being the parameters of the model. Model examples: $F_\theta(x) = \theta^T x$ as a simple linear model. Deep Models are recursive functions:

$$F_{\theta_1,\theta_2,\cdots,\theta_l}(x) = f_1(\theta_1^T f_2(\cdots \theta_{l-1}^T f_l(\theta_l^T x)))$$

- Loss Functions: The Loss Function L tries to measure the *distance* between $F_\theta(x_i)$ and y_i.

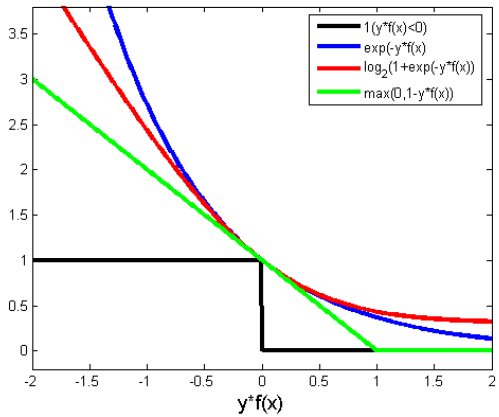
Supervised Learning: Optimization Problem

- "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: $||\theta||_2^2 = \sum_{i=1}^{m} \theta[i]^2$

Some Concrete Supervised Learning Instances

- L1 Regularized Logistic Regression:
  $\min_\theta \sum_{i=1}^n \log(1 + \exp(-y_i F_\theta(x_i))) + \lambda||\theta||_1$
- L2 Regularized Logistic Regression:
  $\min_\theta \sum_{i=1}^n \log(1 + \exp(-y_i F_\theta(x_i))) + \lambda||\theta||_2^2$
- L2 Regularized SVMs: $\min_\theta \sum_{i=1}^n \max\{0, 1 - y_i F_\theta(x_i)\} + \lambda||\theta||_2^2$
- L2 Regularized Multi-class Logistic Regression:
  $\min_{\theta_1,\cdots,\theta_k} \sum_{i=1}^n \{-F_{\theta_{y_i}}(x_i) + \log(\sum_{c=1}^k \exp(F_{\theta_c}(x_i)))\} + \sum_{i=1}^c \lambda||\theta_i||_2^2$
- L1 Regularized Least Squares (Lasso):
  $\min_\theta \sum_{i=1}^n (F_\theta(x_i) - y_i)^2 + \lambda||\theta||_1$
- L2 Regularized Least Squares (Ridge):
  $\min_\theta \sum_{i=1}^n (F_\theta(x_i) - y_i)^2 + \lambda||\theta||_2^2$

Application 2: Clustering

- This is an instance of unsupervised learning.
- Data: Given unsupervised data $\{x_1, x_2, \cdots, x_n\}$ where $x_i \in \mathrm{R}^m$ is the feature vectors.
- Goal: Find clusters (sets) $C_1, C_2, \cdots, C_k$ with each cluster consisting of *similar* instances. Denote $V = \{1, \cdots, n\}$. Then $\cup_{i=1}^{k} C_i = V$.
- Optimization Problem: The k-means optimization problem is:

$$\min_{C_1, C_2, , C_k} \sum_{i=1}^{k} \sum_{x \in C_i} |x - 1/|C_i| \sum_{x_j \in C_i} x_j|_2^2$$

- This problem can actually be viewed as a joint discrete and continuous problem.

Application 3: Principal Component Analysis

- This is an instance of unsupervised learning.
- Data: Given unsupervised data $\{x_1, x_2, \cdots, x_n\}$ where $x_i \in \mathrm{R}^m$ is the feature vectors.
- Goal: Find compressors $V \in \mathbb{R}^{k \times m}$ (and correspondingly decompresses $U \in \mathbb{R}^{m \times k}$) such that x_i is *close* to UVx_i.
- It turns out the compressor must satisfy $V = U^T$ such that $U^T U = I$.
- Optimization Problem: The PCA optimization problem is:

$$\min_{U: U^T U = I} \sum_{i=1}^{n} ||x_i - UU^T x_i||_2^2$$

Application 4: Matrix Completion

- Data: Given observations $y_1, \cdots, y_n$, such that each $y_j = A_j(X)$ where A_j could be a single element or a combination of elements in $X \in \mathbb{R}^{m \times n}$. Consider for example X being product recommendation matrix.
- Goal: Find the *simplest* matrix X s.t $A_j(X) \approx y_j, \forall j \in 1, \cdots, n$
- Optimization Problem: Matrix completion optimization problem is:

$$\min_X \sum_{j=1}^{n} ||y_j - A_j(X)||_2^2 + ||X||_*$$

(The nuclear norm tries to ensure the Matrix X is low-rank)

- Another way is to explicitly model this is by assuming $X = LR$ where $L \in \mathbb{R}^{m \times k}$ and $R \in \mathbb{R}^{k \times n}$ (and hence X is rank r), and optimize for L and R.

Application 5: Low Rank and Non Negative Matrix Factorization

- Goal: Find low rank matrices L, R with $L \in \mathbb{R}^{m \times k}$ and $R \in \mathbb{R}^{k \times n}$ s.t $A_j(LR) \approx y_j, \forall j \in 1, \cdots, n$
- Optimization Problem: Matrix Factorization optimization problem is:

$$\min_{L,R} \sum_{i=1}^{n} ||y_i - A_j(LR)||_2^2$$

No need of matrix regularization.

- We can also add non-negativity constraints and this becomes non-negative Matrix Factorization: $\min_{L,R:L \geq 0, R \geq 0} \sum_{i=1}^{n} ||y_i - A_i(LR)||_2^2$
- Sometimes Y is fully observed and we want a non-negative low rank factorization of $Y \approx LR$. The optimization problem is: $\min_{L,R:L \geq 0, R \geq 0} \sum_{i=1}^{n} ||Y - LR||_2^2$.

Application 6: Contextual Bandits and Learning from Logged Data

- Scenario: Learn from logged contextual bandit data. Example: We need to show k ads to users with each ad comprising of features (title, ad text, query etc.), and given an online policy which (with certain randomization) picks ads to show to users and the system logs feedback (whether the user clicks on the ad or not).
- Data: We are given bandit logged data in the form of $\{(x_1, a_1, r_1, p_1), \cdots, (x_n, a_n, r_n, p_n)\}$. Notation::
  - Assume we can take fixed number of actions $1 : k$
  - Denote $x_i = \{x_1^1, \cdots, x_1^k\}$ as the feature vectors of the k actions
  - Denote a_i as the chosen action by the current online policy
  - Denote p_i as the probability of the logged action (by the current online policy)
  - Denote r_i as the Reward obtained by choosing action a_i
  - Define our policy as $\pi_\theta(x) = \text{argmax}_{i=1:k} F_\theta(x^i)$. Again the simplest example of $F_\theta(x) = \theta^T x$.

Application 6: Contextual Bandits and Learning from Logged Data

- Optimization Problem: The Inverse Propensity Estimate of the Reward (which is an unbiased estimate of the Reward function is):

$$\max_{\theta} \text{IPS}(\theta) = \max_{\theta} \sum_{i=1}^{n} r_i / p_i I(\pi_\theta(x_i) == a_i)$$

- SoftMax Relaxation: The IPS objective above is not continuous and non differentiable. We can define a softmax relaxation as:

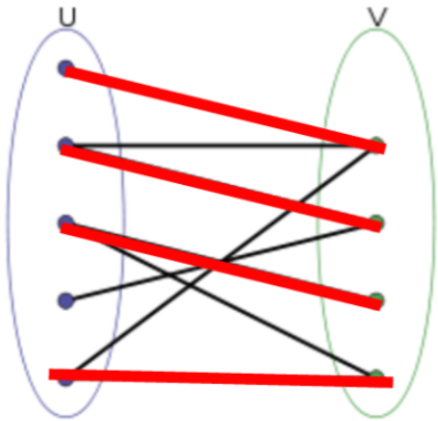
$$\max_{\theta} \text{SM}(\theta) = \max_{\theta} \sum_{i=1}^{n} r_i / p_i \frac{\exp(F_\theta(x_i^{a_i}))}{\sum_{j=1}^{k} \exp(F_\theta(x_i^{j}))}$$

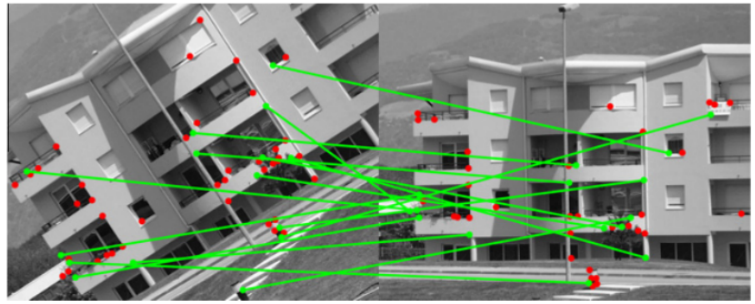
Discrete Optimization in Machine Learning

- 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, Image Correspondence
- Genomics and Computational Biology: cell types or assay selection, selecting peptides and proteins

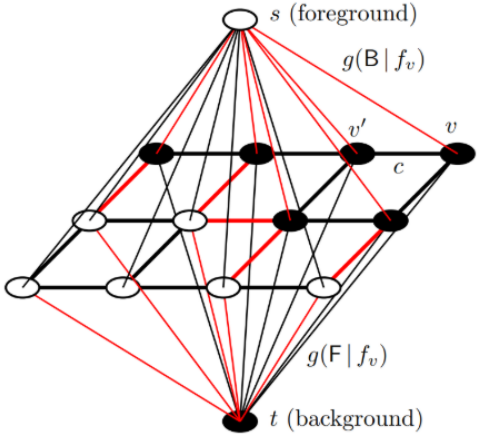
Application 1: Image Segmentation and Correspondence

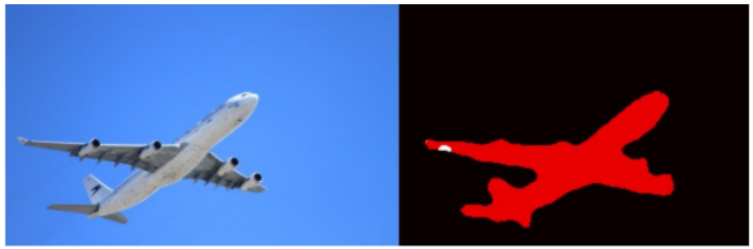
Bipartite Matchings





Minimum Cuts





Application 2: Feature Selection

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

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 setting makes even more sense when the labels are missing on some or all of the given data-points.
- Optimization Problem: Let $D_{KL}(.)$ denote the KL divergence between two distributions. The optimization problem can be cast as:

$$\max_{A:|A|\leq k} D_{KL}(p(X_A)|p(X_V))$$

- We will see in the second part of this course that this is also related to the concept of *submodularity*.

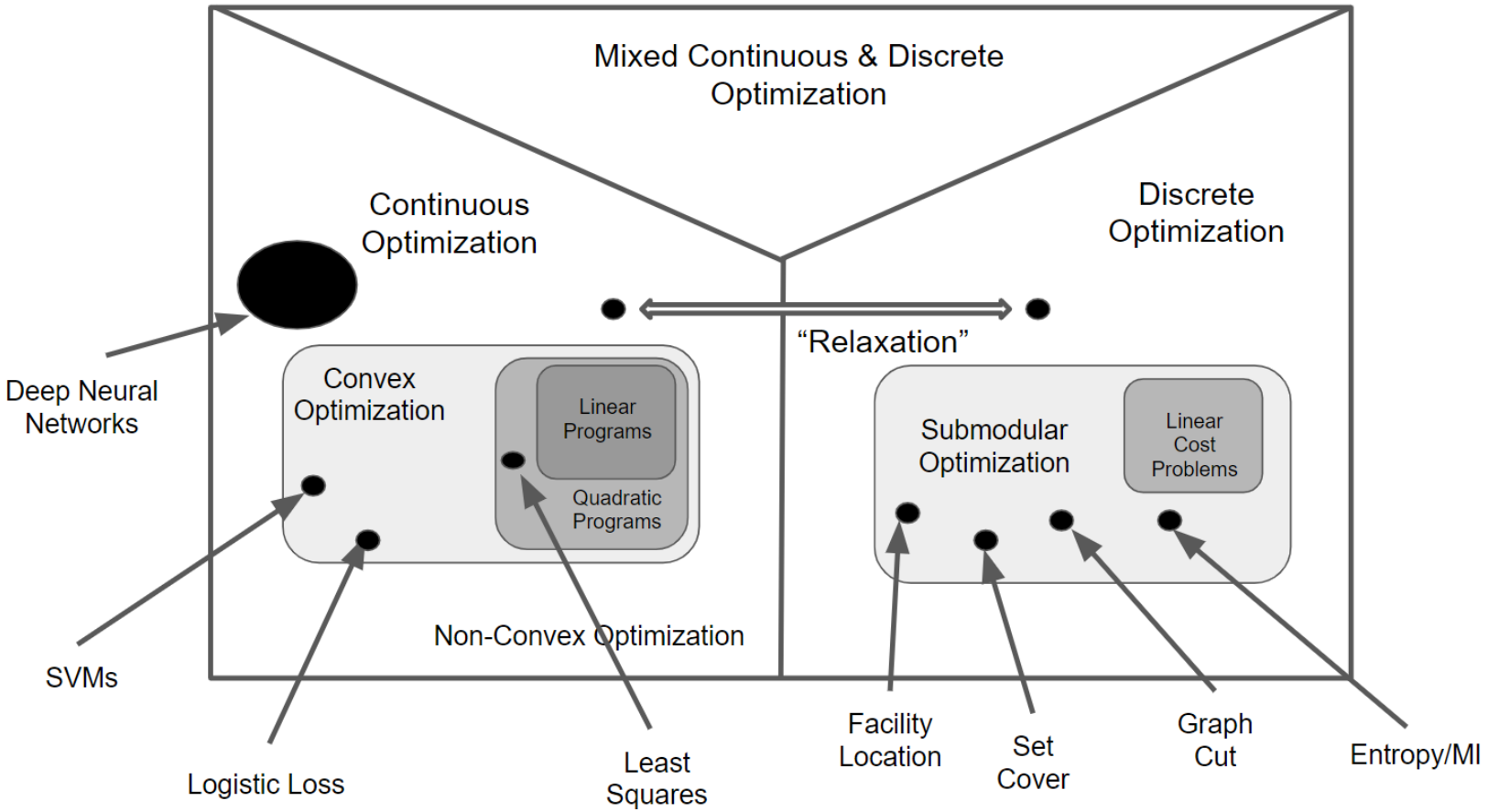
Application 4: k-Mediods Clustering

- 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*.

Big Picture: Types of Optimization Problems



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 $x \geq 0$ 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.
- 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

- 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.
- 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:

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