Optimization in Machine Learning

Lecture 1: Introduction and Course Overview

Ganesh Ramakrishnan

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

January, 2021

Outline

- Why take this course?
- Prerequisites
- Week by Week Course plan
- Course Logistics
- Continuous Optimization in Machine Learning
- Discrete Optimization in Machine Learning

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

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.

Why take this Course?

Machine Learning and AI are embedded in practically every spear of our life: Google/Bing Search & Ads, Amazon product search/recommendation, driverless cars, Google Maps, Uber/Ola matching, Google Photos, Youtube, Facebook, Twitter, Microsoft Office, ...

- With democratization of AI, software engineers can write ML applications with a few lines of code!

Why take this Course?

Machine Learning and AI are embedded in practically every spear of our life: Google/Bing Search & Ads, Amazon product search/recommendation, driverless cars, Google Maps, Uber/Ola matching, Google Photos, Youtube, Facebook, Twitter, Microsoft Office, ...

- With democratization of AI, software engineers can write ML applications with a few lines of code!
- Open-source libraries today offer capabilities to build products with practically zero knowledge of ML.

Why take this Course?

Machine Learning and AI are embedded in practically every spear of our life: Google/Bing Search & Ads, Amazon product search/recommendation, driverless cars, Google Maps, Uber/Ola matching, Google Photos, Youtube, Facebook, Twitter, Microsoft Office, ...

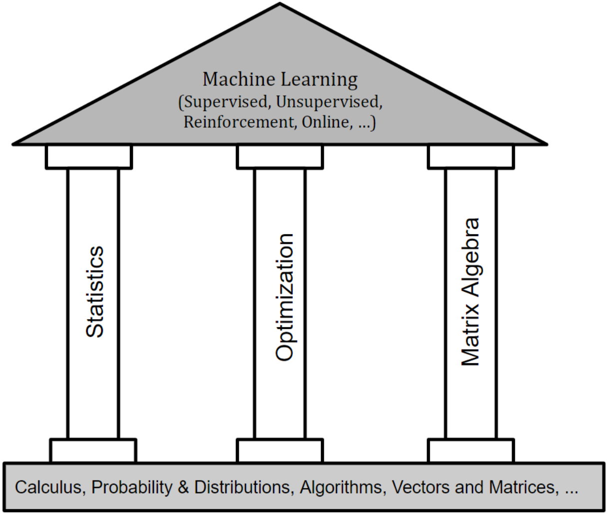
- With democratization of AI, software engineers can write ML applications with a few lines of code!
- Open-source libraries today offer capabilities to build products with practically zero knowledge of ML.
- However to push the boundaries of research and really solve problems, you need to gain hands on experience in ML!

Why take this Course?

Machine Learning and AI are embedded in practically every spear of our life: Google/Bing Search & Ads, Amazon product search/recommendation, driverless cars, Google Maps, Uber/Ola matching, Google Photos, Youtube, Facebook, Twitter, Microsoft Office, ...

- With democratization of AI, software engineers can write ML applications with a few lines of code!
- Open-source libraries today offer capabilities to build products with practically zero knowledge of ML.
- However to push the boundaries of research and really solve problems, you need to gain hands on experience in ML!
- Optimization is one of the important ackbones of machine learning.

Why take this Course?



Why take this Course?

Optimization one of the pillers of ML!

- 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**.

Why take this Course?

Optimization one of the pillers of ML!

- 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, Architechture search etc.

Why take this Course?

Optimization one of the pillers of ML!

- 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, Architechture search etc.
- Mixed Continuous and Discrete Optimization:
  - A growing number of problems including classical problems such as clustering, feature selection, structured sparsity occur as mixed discrete/continuous optimization problems.

Why take this Course?

- Countless number of ML libraries available which implement all kinds of optimization algorithms (Tensorflow, pytorch, scipy, sklearn, Vowpal Wabbit, ...)

Why take this Course?

- 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.

Why take this Course?

- 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.

Why take this Course?

- 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.
- Even if you don't implement new algorithms, you will have a better idea of which algorithm to use in which sceario.

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 implementational aspects and as a part of assignments, we will implement many ML loss functions and algorithms.

Why did you enroll for this Course?

Lets hear from a few of you why you took this course...

Prerequisites

- Basic Linear Algebra: Matrices, Vectors
- Basics of Machine Learning (Ideally you should have taken either an undergraduate or graduate ML course)
- Algorithms course (either in undergraduate or graduate version)

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: Lagrange, 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

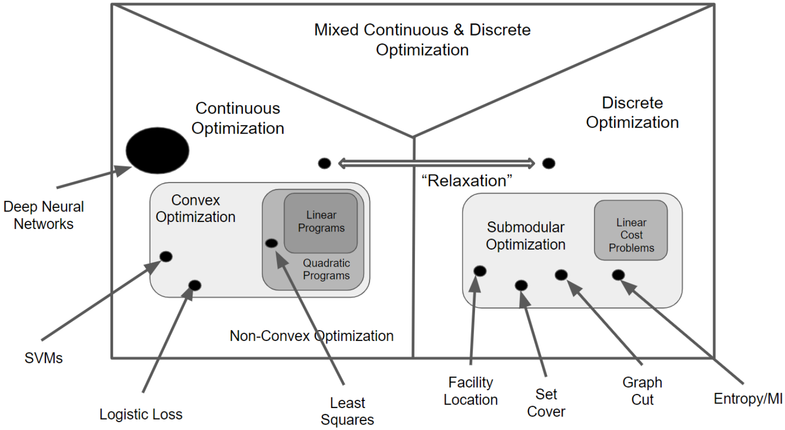
Relevant Books for this Course

- 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.

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

Big Picture: Types of Optimization Problems



Logistics, Grading etc: https://www.cse.iitb.ac.in/~ganesh/cs769/

- Credit/Audit Requirements Anyone who does an exceptional course project that has the potential to be a publishable paper is eligible for a straight AA grade. Otherwise the grading breakup would be:
  - 20% Mid-semester exam
  - 30% End semester exam
  - 20% Project: A basic project will take any of the algorithms we study or any related papers, implement the algorithms in the paper, do a basic performance study and diagnose the performance. However, I would expect most projects to suggest ideas for improvement (atleast in specific settings such as multi core or multiple nodes or reasonable assumptions on matrices etc in the problem for which greater speedup is possible). A more advanced project would take a problem specification for which no solution is publicly available, figure out how to solve it, and implement the solution.
  - 10% Reading and paper presentation.
  - 20% 2 Programming Assignments
- Lectures: In Slot 2, Lectures on MS Teams (Code: 6yxnonu) and will be recorded. All lecture recordings and slides will be organized on moodle.

Course Project Ideas

- Let's spend a few minutes discussing some ideas for course project(s)
- Why not we all jointly contribute to our **DECILE** `https://decile.org/` python toolkit and add to it a component **OptML** 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.

Adding **OptML** to DECILE

The **DECILE** (`https://decile.org/`) python toolkit is being built toward principled human-machine interaction for machine learning, has 4 important components and we can have **OptML** as the fifth!

1. Submodlib (in C++ with Python wrappers) `https://github.com/vishkaush/submodlib/`: Submodlib is an efficient and scalable library for submodular optimization which finds its application in summarization, data subset selection, hyper parameter tuning etc.
2. DISTIL (`https://github.com/decile-team/distil`): This is a library in python for Deep dIverSified inTeractIve Learning
3. CORDS (`https://github.com/decile-team/cords`): This is a library in python for COResets and Data Subset selection
4. DOMAIN (`https://github.com/oishik75/CAGE`): This is a library (work-in-progress) in python for Data prOgraMming viA rule induction through human INteraction
5. **OptML** can implement several continuous loss functions optimization algorithms along with wrappers to machine learning models (*e.g.*, classification, recommender systems, regression *etc.*).

Application 1: Supervised Learning

- Data: Given training examples $\{(x_1, y_1), \cdots, x_n, y_n\}$ where $x_i \in \mathbf{R}^m$ 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 $x_i \in \mathbf{R}^m$ is the feature vectors and y_i is the label.

Supervised Learning: Modeling

- Data: Given training examples $\{(x_1, y_1), \cdots, (x_n, y_n)\}$ where $x_i \in \mathbf{R}^m$ 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:

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

Supervised Learning: Modeling

- Data: Given training examples $\{(x_1, y_1), \cdots, (x_n, y_n)\}$ where $x_i \in \mathbf{R}^m$ 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:

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

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 $F_\theta(x) = \theta^T x$

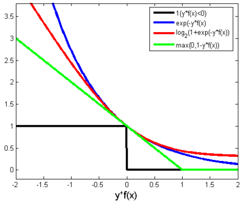
Supervised Learning: Optimization Problem

- "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: $(F_\theta(x_i) - y_i)^2$



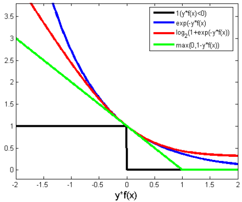
Supervised Learning: Optimization Problem

- "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: $\sum_{i=1}^{m} |\theta[i]|$
  - L2 Regularizer: $\sum_{i=1}^{m} \theta[i]^2$

Some Concrete Supervised Learning Instances

- L1 Regularized Logistic Regression:
  $\min_\theta \sum_{i=1}^{n} \log(1 + \exp(-y_i F_\theta(x_i))) + \lambda \sum_{i=1}^{m} |\theta[i]|$

Some Concrete Supervised Learning Instances

- L1 Regularized Logistic Regression:
  $\min_\theta \sum_{i=1}^{n} \log(1 + \exp(-y_i F_\theta(x_i))) + \lambda \sum_{i=1}^{m} |\theta[i]|$
- L2 Regularized Logistic Regression:
  $\min_\theta \sum_{i=1}^{n} \log(1 + \exp(-y_i F_\theta(x_i))) + \lambda \sum_{i=1}^{m} \theta[i]^2$

Some Concrete Supervised Learning Instances

- L1 Regularized Logistic Regression: $\min_\theta \sum_{i=1}^{n} \log(1+\exp(-y_i F_\theta(x_i))) + \lambda \sum_{i=1}^{m} |\theta[i]|$
- L2 Regularized Logistic Regression: $\min_\theta \sum_{i=1}^{n} \log(1+\exp(-y_i F_\theta(x_i))) + \lambda \sum_{i=1}^{m} \theta[i]^2$
- L2 Regularized SVMs: $\min_\theta \sum_{i=1}^{n} \max\{0, 1 - y_i F_\theta(x_i)\} + \lambda \sum_{i=1}^{m} \theta[i]^2$

Some Concrete Supervised Learning Instances

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

Some Concrete Supervised Learning Instances

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

Some Concrete Supervised Learning Instances

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

Application 2: Clustering

- This is an instance of unsupervised learning.

Application 2: Clustering

- This is an instance of unsupervised learning.
- Data: Given unsupervised data $\{x_1, x_2, \cdots, x_n\}$ where $x_i \in \mathbf{R}^m$ is the feature vectors.

Application 2: Clustering

- This is an instance of unsupervised learning.
- Data: Given unsupervised data $\{x_1, x_2, \cdots, x_n\}$ where $x_i \in \mathbf{R}^m$ 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 $\cup_{i=1}^{k} C_i = V$.

Application 2: Clustering

- This is an instance of unsupervised learning.
- Data: Given unsupervised data $\{x_1, x_2, \cdots, x_n\}$ where $x_i \in \mathbf{R}^m$ 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:

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

Application 2: Clustering

- This is an instance of unsupervised learning.
- Data: Given unsupervised data $\{x_1, x_2, \cdots, x_n\}$ where $x_i \in \mathbf{R}^m$ 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.

Application 3: Principal Component Analysis

- This is an instance of unsupervised learning.
- Data: Given unsupervised data $\{x_1, x_2, \cdots, x_n\}$ where $x_i \in \mathbf{R}^m$ is the feature vectors.

Application 3: Principal Component Analysis

- This is an instance of unsupervised learning.
- Data: Given unsupervised data $\{x_1, x_2, \cdots, x_n\}$ where $x_i \in \mathbf{R}^m$ is the feature vectors.
- Goal: Find compressors $U \in \mathbb{R}^{k \times m}$ (and correspondingly decompresses $V \in \mathbb{R}^{m \times k}$) such that x_i is *close* to UVx_i.

Application 3: Principal Component Analysis

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

Application 3: Principal Component Analysis

- This is an instance of unsupervised learning.
- Data: Given unsupervised data $\{x_1, x_2, \cdots, x_n\}$ where $x_i \in \mathbf{R}^m$ is the feature vectors.
- Goal: Find compressors $U \in \mathbb{R}^{k \times m}$ (and correspondingly decompresses $V \in \mathbb{R}^{m \times k}$) 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.

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 $A_j(X) \approx y_j, \forall j \in 1, \cdots, n$

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:

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

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

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:

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

(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 $A_j(LR) \approx y_j, \forall j \in 1, \cdots, n$

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.

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: $\min_{L,R:L \geq 0, R \geq 0} \sum_{i=1}^{n} ||y_i - A_j(LR)||_2^2$

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: $\min_{L,R:L\geq 0,R\geq 0} \sum_{i=1}^{n} ||y_i - A_j(LR)||_2^2$
- 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).

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::

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 $1 : k$

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

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

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)

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

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):

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

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):

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

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

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

Discrete Optimization in Machine Learning

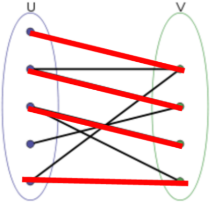
- MAP inference in Probabilistic Models: Ising Models, DPPs
- Feature Subset Selection
- Data Partitioning
- Data Subset Selection
- Data Summarization: Text, Images, Video Summarization
- Social networks, Influence Maximization

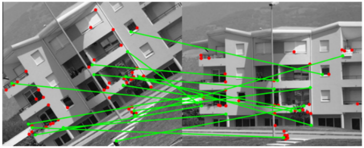
Discrete Optimization in Machine Learning

- 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

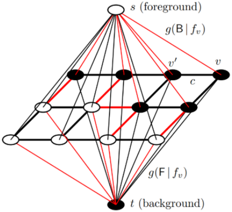
Application 1: Image Segmentation and Correspondence

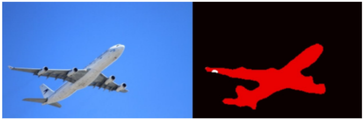
Bipartite Matchings





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.

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.

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:

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

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.

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.

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.

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:

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

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.

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.

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:

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

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:

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

- 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