Optimization in Machine Learning

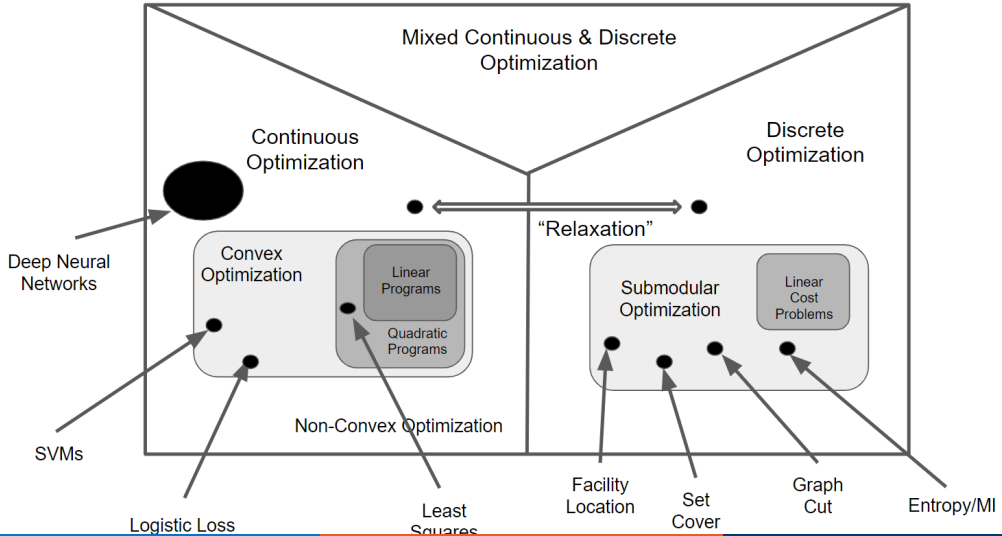
Lecture 3: Continuous Optimization Examples (Clustering, Contextual Bandits), Discrete Optimization Examples (MAP Inference, Feature Selection, Data Subset Selection, *etc.*) and Calculus Brushup for Optimization in Machine Learning

Ganesh Ramakrishnan

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

January, 2025

[Recap] Big Picture: Types of Optimization Problems



Recap Logistics, Grading etc: Mostly moodle, some at 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
  - 25% End semester exam
  - 35% 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.
  - 20% Reading and paper presentation.
- Lectures: In Slot 9, Lectures on MS Teams (Code: **cnsjv56**)
- All lecture recordings and slides will be organized on moodle (CS 769-2024-2— Course name Optimization in Machine Learning)
- TA(s): Prateek Chand, Priya Mishra, Suraj Racha, Vedant Goswami (and possibly more TAs might get added)

Seminar + Project: Next Steps

- To help us facilitate seminar and grading, we have compiled a list of potential seminar + project topics which you can find here: `https://bit.ly/cs769-course-paper-2025`
- Please review the project topics and submit your top three preferences through this form: https://forms.gle/AtVeRvFbqihpp2rW9.
- The deadline to submit your preferences is Tuesday, January 14th at 11:59 PM.
- If you have already connected with classmates and are interested in forming a team together, please indicate this within the form.
- Otherwise, we will use your project preferences to create well-matched teams.

What we have covered so far

- Why take this course? [Done]
- Prerequisites [Done]
- Course plan [Done]
- Course Logistics [Done]
- Continuous Optimization in Machine Learning: Applications include
  1. Supervised Learning [Done]
  2. Principal Component Analysis [Done]
  3. Low Rank and Non Negative Matrix Factorization [Done]
  4. Clustering [Started]
  5. Contextual Bandits and Learning from Logged Data [Today]
- Discrete Optimization in Machine Learning

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
  6. Contextual Bandits and Learning from Logged Data
- Discrete Optimization in Machine Learning
  1. MAP inference in Probabilistic Models: Ising Models, DPPs
  2. Feature Subset Selection
  3. Data Partitioning
  4. Data Subset Selection
  5. Data Summarization: Text, Images, Video Summarization
  6. Social networks, Influence Maximization

Recap: Continuous Optimization in Machine Learning

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

- Supervised Learning: Fine tuning of Deep Models, Fine tuning of LLMs
  - Canonical examples of Logistic Regression, Least Squares, Support Vector Machines [Done]
- Unsupervised Learning: Pre-training of Deep Models, Pre-training of LLMs,
  - Principal Component Analysis, k-Means Clustering
- Reinforcement Leaning from Human Feedback (RLHF), Agentic Systems
  - Canonical examples of Contextual Bandits and Reinforcement Learning: Soft-Max Estimators, Policy Exponential Models
  - Canonical examples of Recommender Systems: Matrix Completion, Non-Negative Matrix Factorization, Collaborative Filtering

Application 5: Clustering

- This is an instance of unsupervised learning.

Application 5: 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 5: Clustering

Continuous relaxation.

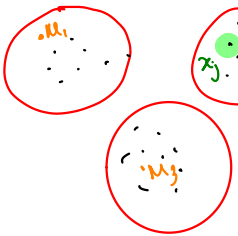
- This is an instance of unsupervised learning.
- Data: Given unsupervised data $\{x_1, x_2, \cdots, x_n\}$ where $x_i \in \mathbf{R}^m$ is the feature vectors.

Simplicity for clustering?

$$\min_{P_{ij} \in \{0,1\}} \|\mu_i - P_{ij} x_j\|^2$$

Suggestion: $\sum_i P_{ij} - 1 = 0$, $P_{ij} \in [0,1]$?

The idea behind soft-k-means clustering



$$P_{ij}^t \propto \left[d(x_j, \mu_i^{t-1})\right]^{-1}$$

$$\mu_i^t = \sum_j P_{ij}^t x_j$$

Application 5: 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 5: 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 $\cup_{i=1}^{k} C_i = V$.

$C_m \cap C_n = \emptyset \quad \forall\ m \neq n$

What if we want membership P such that the data is partitioned

$P_{ij} \in \{0, 1\}$

Will be discrete (+ continuous) optimization problems

Application 5: 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:

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

Application 5: 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:

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

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

Application 6: RLHF, Contextual Bandits and Learning from Logged Data

- Introduction to Contextual Bandits.
- Applications in Generative AI, especially RLHF for LLM.
- Motivation and connection to Large Language Models (LLMs).
- Simpson's Paradox and its relevance to RLHF.
- Mathematical optimization objectives.

Contextual Bandits: A Primer

- A "Contextual Bandit" receives a *context* x_t (features describing the current situation), chooses an *action* a_t, and receives a *reward* $r_t(x_t, a_t)$.
- The objective is to select actions that maximize the expected cumulative reward.
- Example: Personalized content recommendation.
  - **Context:** User demographics, browsing history.
  - **Action:** Displaying a specific article.
  - **Reward:** A click on the article.

Contextual Bandits in Reinforcement Learning with Human Feedback (RLHF)

- **Context in RLHF:** The prompt and previously generated tokens.
- **Action:** Generating a token or sequence.
- **Reward:** Feedback from human annotators or an automated reward model.
- Connection to Contextual Bandits:
  - RLHF can be viewed as a contextual bandit problem where the agent learns to predict helpful completions.
  - Example: For the query *"Where is the Virupaksha temple?"*, multiple completions may include:
    - *"in Hampi"* (correct).
    - *"in Srirangam"* (incorrect).

    The model receives a higher reward for *"in Hampi"* and updates its policy accordingly.

Application 6: RLHF, Contextual Bandits and Learning from Logged Data

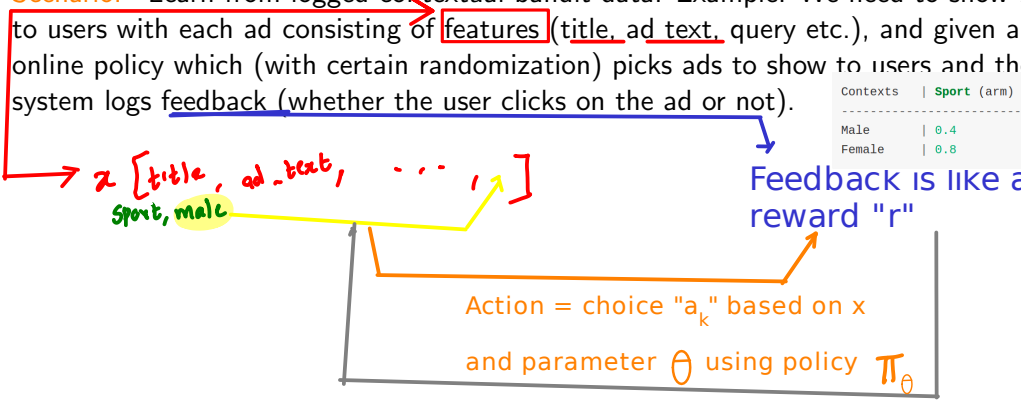
- Scenario: Learn from logged contextual bandit data. Example: We need to show K ads to users with each ad consisting of features (title, ad text, query etc.), and given an online policy which (with certain randomization) picks ads to show to users and the system logs feedback (whether the user clicks on the ad or not).

https://www.usu.edu/math/schneit/ctis/sp/
Try out the above to appreciate the Simpson's paradox. How can you pose an optimization problem in the context of this Paradox in order to learn the right recommendation POLICY
based on featurization of the visitors as well as featurization of the objects being recommended

Application 6: RLHF, 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 consisting 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).

Contexts	Sport (arm)	Movie (arm)
Male	0.4	0.3
Female	0.8	0.7



Application 6: RLHF, 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 consisting 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 $\{(\mathbf{x}_1, a_1, r_1, p_1), \cdots, (\mathbf{x}_n, a_n, r_n, p_n)\}$. Notation:: **(usual jargon)**

Application 6: RLHF, 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 consisting 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 $\{(\mathbf{x}_1, a_1, r_1, p_1), \cdots, (\mathbf{x}_n, a_n, r_n, p_n)\}$. Notation:: **(usual jargon)**

The system's reference model had a propensity of recommending movies to 75% Females and sports to 75% Males

Results in a bias

Contexts	Sport (arm)	Movie (arm)
Male	0.4 x 0.75	0.3 x 0.25
Female	0.8 x 0.25	0.7 x 0.75
CTR total	0.5	0.6

Application 6: RLHF, 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 consisting 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 $\{(\mathbf{x}_1, a_1, r_1, p_1), \cdots, (\mathbf{x}_n, a_n, r_n, p_n)\}$. Notation:: **(usual jargon)**
  - Assume we can take fixed number of actions **(arms)** $1 : K$

Application 6: RLHF, 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 consisting 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 $\{(\mathbf{x}_1, a_1, r_1, p_1), \cdots, (\mathbf{x}_n, a_n, r_n, p_n)\}$. Notation:: **(usual jargon)**
  - Assume we can take fixed number of actions **(arms)** $1 : K$

Eg: a_1 = showing sports ad

a_2 = showing movies ad

Application 6: RLHF, 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 consisting 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 $\{(\mathbf{x}_1, a_1, r_1, p_1), \cdots, (\mathbf{x}_n, a_n, r_n, p_n)\}$. Notation:: **(usual jargon)**
  - Assume we can take fixed number of actions **(arms)** $1 : K$
  - Let $\mathbf{x}_i = \{x_i^1, \cdots, x_i^K\}$ be feature vector associated with the K actions

Application 6: RLHF, 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 consisting 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 $\{(\mathbf{x}_1, a_1, r_1, p_1), \cdots, (\mathbf{x}_n, a_n, r_n, p_n)\}$. Notation:: **(usual jargon)**
  - Assume we can take fixed number of actions **(arms)** $1 : K$
  - Let $\mathbf{x}_i = \{x_i^1, \cdots, x_i^K\}$ be feature vector associated with the K actions
  - Denote a_i as the chosen action by the current online policy

Goal: Have the model predicted argmax action for each input i match the action for which the user assigns the highest reward

Application 6: RLHF, 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 consisting 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 $\{(\mathbf{x}_1, a_1, r_1, p_1), \cdots, (\mathbf{x}_n, a_n, r_n, p_n)\}$. Notation:: **(usual jargon)**
  - Assume we can take fixed number of actions **(arms)** $1 : K$
  - Let $\mathbf{x}_i = \{x_i^1, \cdots, x_i^K\}$ be feature vector associated with 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 reference policy π_{ref} **(Propensity Score)**

Goal: Have the model predicted argmax action for each input i match the action for which the user assigns the highest reward

That is, Learn the model from Data and Regularize based on propensity

Application 6: RLHF, 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 consisting 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 $\{(\mathbf{x}_1, a_1, r_1, p_1), \cdots, (\mathbf{x}_n, a_n, r_n, p_n)\}$. Notation:: **(usual jargon)**
  - Assume we can take fixed number of actions **(arms)** $1 : K$
  - Let $\mathbf{x}_i = \{x_i^1, \cdots, x_i^K\}$ be feature vector associated with 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 reference policy π_{ref} **(Propensity Score)**
  - Denote r_i as the Reward obtained by choosing action a_i

Application 6: RLHF, 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 consisting 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 $\{(\mathbf{x}_1, a_1, r_1, p_1), \cdots, (\mathbf{x}_n, a_n, r_n, p_n)\}$. Notation:: **(usual jargon)**
  - Assume we can take fixed number of actions **(arms)** $1 : K$
  - Let $\mathbf{x}_i = \{x_i^1, \cdots, x_i^K\}$ be feature vector associated with 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 reference policy π_{ref} **(Propensity Score)**
  - Denote r_i as the Reward obtained by choosing action a_i

Index i is for input

Contexts	Sport (arm)	Movie (arm)
Male	0.4	0.3
Female	0.8	0.7

Application 6: RLHF, 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 consisting 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 $\{(\mathbf{x}_1, a_1, r_1, p_1), \cdots, (\mathbf{x}_n, a_n, r_n, p_n)\}$. Notation:: **(usual jargon)**
  - Assume we can take fixed number of actions **(arms)** $1 : K$
  - Let $\mathbf{x}_i = \{x_i^1, \cdots, x_i^K\}$ be feature vector associated with 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 reference policy π_{ref} **(Propensity Score)**
  - Denote r_i as the Reward obtained by choosing action a_i
  - Define our policy as $\pi_\theta(x) = \text{argmax}_{j=1:K} F_\theta(x^j)$. Again the simplest example of $F_\theta(\mathbf{x}) = \theta^T \mathbf{x}$ **(F is the Link Function)**

Application 6: RLHF, 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):

$$\max_{\theta} \text{IPS}(\theta) = \max_{\theta} \sum_{i=1}^{n} \frac{r_i}{p_i} I(\pi_\theta(\mathbf{x}_i) == a_i)$$

Application 6: RLHF, 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):

$$\max_{\theta} \text{IPS}(\theta) = \max_{\theta} \sum_{i=1}^{n} \frac{r_i}{p_i} I(\pi_\theta(\mathbf{x}_i) == a_i)$$

Issues/limitation
1) rewards are assumed to be non-zero only for the selected action. What if there are non-zero rewards also for other actions that were not selected?
2) no explicit index into the type of the input (the input type was to be handled by the black box featurizer)
3) The entire log is consired to be one monolithic set. No temporal index

Application 6: RLHF, 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):

$$\max_{\theta} \text{IPS}(\theta) = \max_{\theta} \sum_{i=1}^{n} \frac{r_i}{p_i} I(\pi_\theta(\mathbf{x}_i) == a_i)$$

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

$$\max_{\theta} \text{SM}(\theta) = \max_{\theta} \sum_{i=1}^{n} \frac{r_i}{p_i} \frac{\exp(F_\theta(x_i^{a_i}))}{\sum_{j=1}^{k} \exp(F_\theta(x_i^j))} = \max_{\theta} \mathbb{E}_{y \sim \pi_\theta(\cdot|x)} \left[\frac{r(x,y)}{\pi_{\text{ref}}(y|x)} \pi_\theta(y|x) \right]$$

Even more relevant in the context of RLHF (Reinforcement Learning from Human Feedback) for Gen AI and LLMs.

- Propensity could be the probability of a fine-tined model
- Reward is the the human feedback

Mathematical Formulation: Contextual Bandits generalized to RLHF

- Given context x_t at time t, the agent selects action a_t and receives reward $r_t(x_t, a_t)$.
- **Simple Propensity-free Objective:** Maximize the expected cumulative reward:

$$\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[r(x,a)\right]$$

- In RLHF for LLMs:
  - **Context:** User query and conversation history.
  - **Action:** Generated response.
  - **Reward:** Annotator-provided preference score.
- Regularization term to penalize large deviations from the reference policy (PPO objective):

$$\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[r(x,y)] - \beta\mathsf{KL}(\pi_\theta||\pi_{\mathsf{ref}}).$$

Connecting PPO Regularization and Inverse Propensity Score (IPS) for Rewards

- **PPO Regularization:**
  - Ensures that the updated policy π_θ does not deviate significantly from the reference policy π_{ref}.
  - The objective includes a KL-divergence regularization term:

$$\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[r(x,y)] - \beta \text{KL}(\pi_\theta||\pi_{\text{ref}}).$$

- **Inverse Propensity Score (IPS) for Rewards:**
  - IPS is used to reweight observed samples to correct for biased sampling.
  - The reward is adjusted using importance weights:

$$\text{Adjusted reward} = \frac{r(x,y)}{\pi_{\text{ref}}(y|x)}\pi_\theta(y|x).$$

  - Corrects the reward estimate based on how probable an action was under the reference policy.

Connecting PPO Regularization and Inverse Propensity Score (IPS) for Rewards

- **Key Connection:**
  - PPO uses importance sampling to maintain sample efficiency and stable policy updates.
  - The clipping mechanism in PPO ensures that the importance weights do not grow too large:

$$1 - \epsilon \leq \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} \leq 1 + \epsilon.$$

  - This prevents the inverse propensity weights from introducing high variance, which can destabilize learning.
- **Practical Implication:** IPS-based weighting in conjunction with PPO clipping balances exploration and prevents reward hacking by controlling large deviations from the reference policy.

Understanding Simpson's Paradox

- **Simpson's Paradox:** A trend that appears in different groups may reverse when the groups are combined.
- Example:
  - A model may perform well across individual tasks (summarization, question answering).
  - However, when aggregated across tasks, the performance may appear worse due to sample distribution differences.
- **Relevance to RLHF:**
  - Annotator bias may vary by task or user demographic.
  - Aggregated feedback can introduce misleading trends if feedback is not stratified.

Handling Simpson's Paradox in RLHF

- **Stratification:** Maintain separate feedback distributions for different tasks and contexts.
- **Normalization:** Normalize feedback to ensure fair comparisons across groups.
- **Example:** In summarization and Q&A, normalize rewards so that extreme preferences in one task do not dominate.
- **Result:** The model adapts more robustly to variations across tasks without amplifying feedback biases.

Optimization Challenges and Practical Considerations

- **Challenge 1: High variance in feedback rewards.** [We will deal with this later]
  - Solution: Use the advantage function $A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$ to reduce variance.
- **Challenge 2: Sampling efficiency.**
  - Solution: Use importance weights and Proximal Policy Optimization (PPO) clipping.
  - Clip ratios to ensure stability:

$$1 - \epsilon \leq \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} \leq 1 + \epsilon.$$

- **Challenge 3: Reward hacking.**
  - Solution: Introduce diversity constraints and cross-task regularization.

Summarily...

- Contextual Bandits provide a structured approach to feedback-driven decision-making.
- In RLHF, this framework enhances the fine-tuning of LLMs based on human feedback.
- Awareness of Simpson's Paradox helps mitigate biases introduced during aggregated evaluations.
- Advanced (project) directions: Improved contextual reward modeling and better reward alignment strategies.

Outline

- Continuous Optimization in Machine Learning: Applications include
  1. Supervised Learning [Done]
  2. Principal Component Analysis [Done]
  3. Low Rank and Non Negative Matrix Factorization [Done]
  4. Clustering [Done]
  5. Contextual Bandits and Learning from Logged Data [Done]
- Discrete Optimization in Machine Learning
  1. MAP inference in Probabilistic Models: Ising Models, DPPs
  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

Optimization is used ubiquoutously in all inference problems in ML all the time!

Consider the sentence

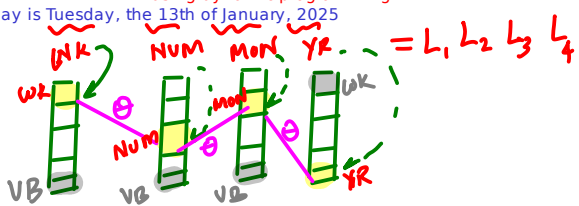
- MAP inference in Probabilistic Models: Today is Tuesday, the 13th of January, 2025
  Ising Models, DPPs
- Feature Subset Selection
- Data Partitioning
- Data Subset Selection
- Data Summarization: Text, Images, Video
  Summarization
- Social networks, Influence Maximization

Discrete Optimization in Machine Learning

$$L = \underset{L}{\operatorname{argmax}}\ \psi_\theta(L_1, L_2 \ldots L_4 \ldots)$$

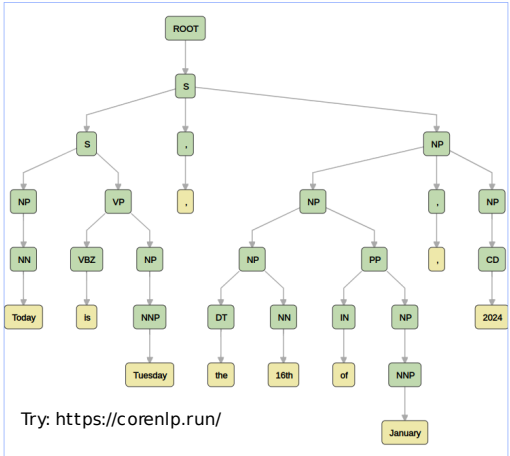
For such Chain (and in general also tree structures) the optimization algorithm can be solved efficiently using dynamic programming

- 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



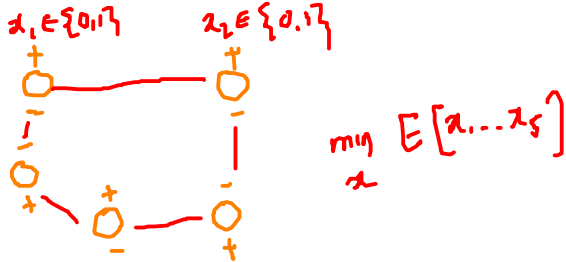
For effective date extraction (or annotation) we would like the configuration highlighted in yellow to have lower energy (or higher likelihood) than any other configuration

BestParse = argmax (all parse trees) Pr (parse | word sequence)



Today is Monday, the 13th of January, 2025

ISING MODEL

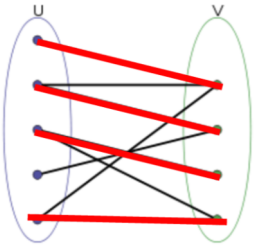


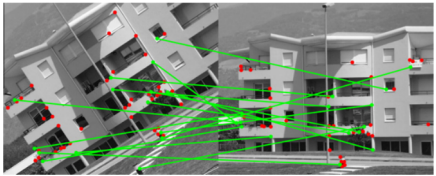
Discrete Optimization in Machine Learning

- 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

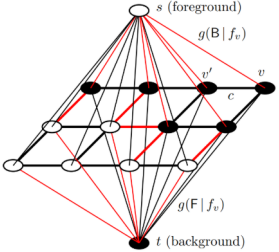
Application 1: Image Segmentation and Correspondence

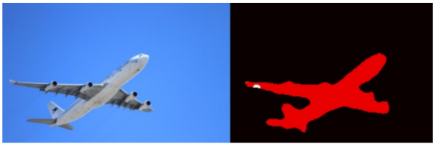
Bipartite Matchings





Minimum Cuts





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

Application 1: Image Segmentation and Correspondence

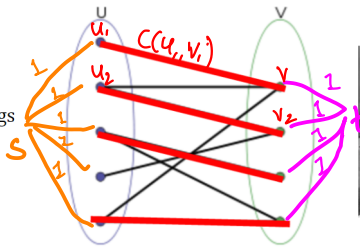
Bipartite Matchings

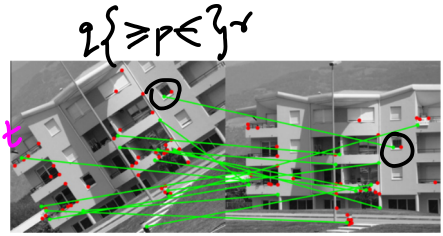
$$\max_{f \in \mathbb{R}} \sum_{u_i \in U} f_{su_i}$$

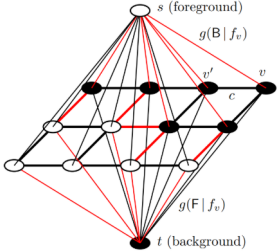
$$s.t \ \forall p \ \sum_{(q,p)\in E} f_{qp} = \sum_{(p,r)\in E} f_{pr}$$

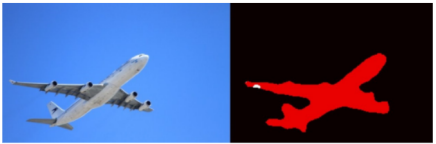
$$0 \le f_{p,q} \le c(p,q)$$

Minimum Cuts





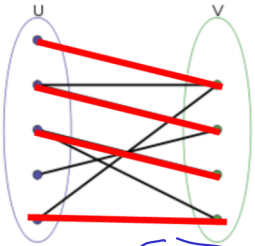


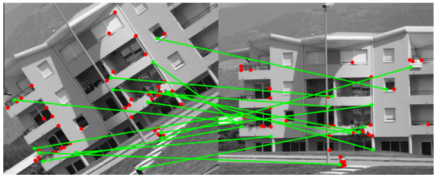


Source: https://www.softserveinc.com/en-us/blog/reduce-seismic-data-interpretation-aws-sagemaker

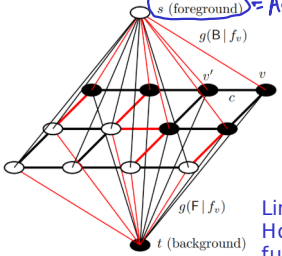
Application 1: Image Segmentation and Correspondence

Bipartite Matchings



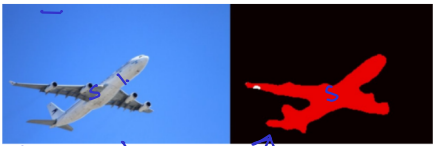


Minimum Cuts



= 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

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:

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

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.

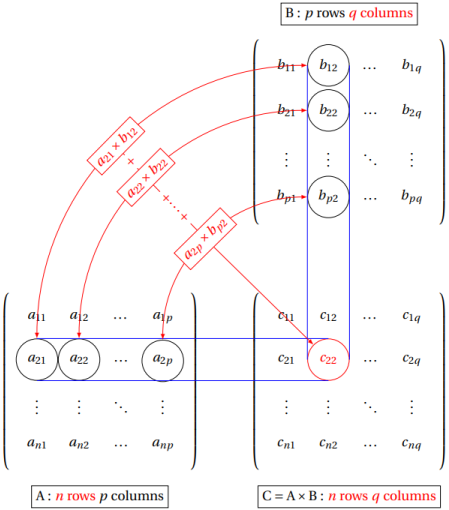
$$\begin{bmatrix} b_{11} & b_{12} & \dots & b_{1p} \\ b_{21} & b_{22} & \dots & b_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{np} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} b_{11} \\ b_{21} \\ \vdots \\ b_{n1} \end{bmatrix} \begin{bmatrix} b_{12} \\ b_{22} \\ \vdots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

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

- 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:

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

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: $L(w) = \sum_{i=1}^{n} r_i / p_i \frac{\exp(w^T x_i^{a_i})}{\sum_{j=1}^{k} \exp(w^T x_i^j)}$.