HW 6

Swagat Adhikary

11/19/2024

What is the difference between gradient descent and *stochastic* gradient descent as discussed in class? (*You need not give full details of each algorithm. Instead you can describe what each does and provide the update step for each. Make sure that in providing the update step for each algorithm you emphasize what is different and why.*)

The key difference between **gradient descent** and **stochastic gradient descent** lies in how much data is used to compute the gradient during each update. Gradient descent calculates the gradient of the loss function using the **entire dataset**, resulting in more stable but computationally expensive updates. Its update step is:

$$\theta_{i+1} = \theta_i - \alpha \nabla f(\theta_i, X, Y),$$

where X, Y represent the entire dataset, and α is the learning rate.

In contrast, **stochastic gradient descent** computes the gradient using a **random subset** of the data (a single data point or small batch) at each step. This makes SGD faster and better suited for large datasets but introduces noise, which can cause more fluctuation in updates. The update step for SGD is:

$$\theta_{i+1} = \theta_i - \alpha \nabla f(\theta_i, X_i', Y_i'),$$

where X_i', Y_i' is the random subset. This random sampling makes SGD computationally efficient and capable of escaping local minima, but less stable compared to the smoother updates of standard gradient descent.

Consider the `FedAve` algorithm. In its most compact form we said the update step is $\omega_{t+1} = \omega_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} \nabla F_k(\omega_t)$. However, we also emphasized a more intuitive, yet equivalent, formulation given by $\omega_{t+1}^k = \omega_t - \eta \nabla F_k(\omega_t); w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$.

Prove that these two formulations are equivalent.
(*Hint: show that if you place ω_{t+1}^k from the first equation (of the second formulation) into the second equation (of the second formulation), this second formulation will reduce to exactly the first formulation.*)

We aim to show that the two formulations of the FedAve algorithm are equivalent: 1. Compact form: $\omega_{t+1} = \omega_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} \nabla F_k(\omega_t)$ 2. Intuitive form: - $\omega_{t+1}^k = \omega_t - \eta \nabla F_k(\omega_t)$ - $\omega_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \omega_{t+1}^k$

Substitute $\omega_{t+1}^k = \omega_t - \eta \nabla F_k(\omega_t)$ into $\omega_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \omega_{t+1}^k$:

$$\omega_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \left(\omega_t - \eta \nabla F_k(\omega_t)\right)$$

Distribute $\frac{n_k}{n}$:

$$\omega_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \omega_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} \nabla F_k(\omega_t)$$

Factor out ω_t in the first term:

$$\omega_{t+1} = \omega_t \sum_{k=1}^{K} \frac{n_k}{n} - \eta \sum_{k=1}^{K} \frac{n_k}{n} \nabla F_k(\omega_t)$$

Since $\sum_{k=1}^{K} \frac{n_k}{n} = 1$, this simplifies to:

$$\omega_{t+1} = \omega_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} \nabla F_k(\omega_t)$$

The compact form is recovered:

$$\omega_{t+1} = \omega_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} \nabla F_k(\omega_t)$$

Thus, the two formulations are equivalent.

Now give a brief explanation as to why the second formulation is more intuitive. That is, you should be able to explain broadly what this update is doing.

The second formulation of the FedAve algorithm is more intuitive because it breaks the global update into two clear and interpretable steps:

1. **Local Updates**: Each client computes its own updated parameters:

   $$\omega_{t+1}^k = \omega_t - \eta \nabla F_k(\omega_t)$$

   This step uses only the client's local data, representing how that client's data informs the parameter update.

2. **Global Aggregation**: The server aggregates these local updates into a weighted average:

   $$\omega_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \omega_{t+1}^k$$

   This step combines contributions from all clients, with each client's update weighted by the proportion of data it contributes ($\frac{n_k}{n}$).

This formulation intuitively mirrors collaborative learning: - Each client contributes its individual learning (local updates) based on its unique data. - The server aggregates these updates, ensuring clients with more data have a stronger influence on the global model. - By keeping data siloed, this approach protects privacy while still enabling effective model training.

Prove that randomized-response differential privacy is ϵ-differentially private.

To prove that the randomized-response mechanism is ϵ-differentially private, we follow these steps.

1. Each individual flips a fair coin (heads or tails).

2. If the coin lands heads, they respond truthfully to the sensitive question.
3. If the coin lands tails, they flip the coin again:
    - Heads: Respond "Yes" (regardless of the truth).
    - Tails: Respond "No" (regardless of the truth).

The mechanism satisfies ϵ-differential privacy if, for any two datasets D_1 and D_2 differing in exactly one individual, and for any subset of possible outputs S:

$$\frac{P[A(D_1) \in S]}{P[A(D_2) \in S]} \leq e^{\epsilon}$$

Here, A is the randomized-response mechanism, and $P[A(D) \in S]$ represents the probability of a specific output given the dataset D.

Let p represent the probability of a "Yes" response: - If the individual is truthful and the answer is "Yes," the probability is:

$$P[\text{Output} = \text{Yes} \mid \text{True} = \text{Yes}] = \frac{1}{2} + \frac{1}{4} = \frac{3}{4}$$

- If the individual is truthful and the answer is "No," the probability is:

$$P[\text{Output} = \text{Yes} \mid \text{True} = \text{No}] = \frac{1}{4}$$

To satisfy ϵ-differential privacy, the ratio of probabilities for any pair of outputs (e.g., "Yes") must satisfy:

$$\frac{P[A(D_1) = \text{Yes}]}{P[A(D_2) = \text{Yes}]} \leq e^{\epsilon}$$

Using the probabilities above:

$$\frac{P[\text{Output} = \text{Yes} \mid \text{True} = \text{Yes}]}{P[\text{Output} = \text{Yes} \mid \text{True} = \text{No}]} = \frac{\frac{3}{4}}{\frac{1}{4}} = 3$$

Thus, the mechanism is ln(3)-differentially private, as:

$$e^{\ln(3)} = 3$$

The randomized-response mechanism satisfies ln(3)-differential privacy because the ratio of probabilities for any two datasets is bounded by $e^{\ln(3)} = 3$, as shown above.

Define the harm principle. Then, discuss whether the harm principle is *currently* applicable to machine learning models. (*Hint: recall our discussions in the moral philosophy primer as to what grounds agency. You should in effect be arguing whether ML models have achieved agency enough to limit the autonomy of the users of said algorithms.*)

The harm principle, as articulated by John Stuart Mill, states that the only purpose for which power can be rightfully exercised over a person is to prevent harm to others.

In essence, individuals should be free to act as they choose, provided their actions do not harm others. This principle places limits on the restriction of autonomy, with harm prevention being the sole justification.

The applicability of the harm principle to machine learning (ML) models hinges on whether ML models possess sufficient **agency** to restrict the autonomy of their users. Agency is typically grounded in moral and rational capabilities—attributes that ML models, despite their increasing complexity, currently lack. Key considerations include:

1. **Sentience**: ML models are tools created and operated under human direction. They do not possess an independent will or moral reasoning, which are prerequisites for agency as discussed in the moral philosophy primer.
2. **Algorithmic Influence**: While ML models can influence user behavior (e.g., recommendation systems shaping preferences), this influence is indirect and mediated by the users' choices. Users retain the ultimate decision-making authority, distinguishing these models from entities capable of directly restricting autonomy.
3. **Potential for Harm**: Although ML models do not independently exert agency, their use can lead to harm, such as discrimination in automated decision-making or surveillance concerns. These harms, however, are a function of their design and application by human agents, not the models themselves.

Currently, ML models do not satisfy the criteria for agency sufficient to invoke the harm principle. They lack moral and rational capacities to autonomously restrict user autonomy. Instead, the harm principle should be applied to the humans designing and deploying these models, ensuring their actions prevent harm while respecting user autonomy.