STOR 390: HOMEWORK 6

Riley Harper

April 9, 2024

Abstract: This homework explores optimization methods in machine learning, specifically Gradient Descent (GD) and Stochastic Gradient Descent (SGD), and their fundamental differences, particularly in the context of the update step. Additionally, it delves into the FedAve algorithm, presenting a proof of equivalence between two of its formulations and offering an intuitive explanation of its federated learning approach. Lastly, it examines the harm principle's relevance to machine learning, questioning the agency of ML models and their potential to limit user autonomy through the unintended consequences of their application.

Question 1

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.)*

Gradient Descent (GD) and Stochastic Gradient Descent (SGD) are methods for optimizing cost functions in machine learning and deep learning algorithms. They primarily differ in their approach to computing and applying gradients for parameter updates. **Gradient Descent (GD)** seeks to minimize the cost function by calculating the gradient based on the entire training dataset and updating the model parameters accordingly. The update rule provided in our lecture was,

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

In this formula, θ_i represents the parameters of the model at iteration i, α is the learning rate, and $\nabla f(\theta_i, x, y)$ denotes the gradient of the cost function with respect to the parameters. This approach uses the entire dataset to compute the gradient, ensuring a comprehensive update at each step but requiring significant computational resources. Conversely, **Stochastic Gradient Descent (SGD)** updates the model parameters using the gradient of the cost function derived from a single, randomly selected data point at each iteration, though in lecture this was taught to be a collection of data points as is commonly used in practice. This can be represented by the following formula,

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

Here, (x', y') is a randomly chosen sample from the dataset, and $\nabla f(\theta_i, x', y')$ represents the gradient of the cost function with respect to θ for that sample. The main difference in SGD from GD lies in the granularity of data used to compute the gradient, making SGD faster and more memory-efficient, particularly for large datasets since each step uses a lesser amount of data than would be used in GD. Both methods aim to reach the cost function's minimum, with GD providing a more stable but computationally intensive path, and SGD offering a faster, albeit potentially erratic, approach. The choice between them depends on the specific requirements and constraints of the machine learning problem at hand.

Question 2

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.)

Proof that the two formulations are equivalent:

Start with the local update step from the second formulation below,

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

Plug ω_{t+1}^k into the global aggregation step below,

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

Substitute ω_{t+1}^k from the local update into the aggregation step such that,

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

Distribute the summation over the terms inside the parentheses such that,

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

Since $\sum_{k=1}^{K} \frac{n_k}{n} = 1$, the first term simplifies to ω_t, leading to:

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

This final expression is identical to the compact form of the FedAve update rule given at the beginning, thus proving that the two formulations are equivalent.

Question 3

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 offers a more intuitive perspective by breaking down the update process into two steps. Firstly, it separates the **local update step** on each node (K total), where the model parameters ω_t are individually updated based on the local data and objectives of each node, represented by $\nabla F_k(\omega_t)$. This mirrors the real-world scenario where data remains on a client's device for privacy reasons, and the computation is decentralized. Each client computes the gradient of their own loss function, reflecting how their local data suggests adjusting the model for better performance. Secondly, the **global aggregation step** combines these local updates into a unified model update. This is achieved by weighting each local update ω_{t+1}^k by the proportion of data n_k/n that the client contributes to the total dataset, ensuring that clients with more data have a proportionally greater influence on the model update. This aggregation mimics the federated learning goal of learning a global model that benefits from all available data without actually sharing the data itself.

This formulation highlights the federated learning's primary objective-local computation followed by a global aggregation to preserve data privacy while benefiting from decentralized data sources. The intuitive formulation clearly outlines the distributed nature of learning and the collective effort in updating the model.

Question 4

Explain how the harm principle places a constraint on personal autonomy. 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, articulated by John Stuart Mill in his seminal work "On Liberty," suggests that the only reason for which power can be rightfully exercised over any member of a civilized community, against their will, is to prevent harm to others. This principle places a constraint on personal autonomy by asserting that individuals are free to act as they wish unless their actions harm others (ie your right to swing your fist ends where my nose begins). In essence, it balances individual freedom with the welfare of others, establishing a boundary where personal freedom ends when it infringes on the rights and well-being of someone else.

Applying the harm principle to machine learning models would involve assessing whether these models can exercise autonomy in a way that could harm individuals, and therefore, whether their operation should be limited based on the harm they could potentially cause. This raises the question of agency in machine learning models. Agency, in moral philosophy, refers to the capacity of an entity to act independently and make free choices. Traditional discussions of agency focus on sentient beings, particularly humans, because of their ability to deliberate, make decisions, and understand the consequences of their actions. Machine learning models, despite their complexity

and ability to make decisions or predictions based on data, do not possess consciousness, self-awareness, or the ability to deliberate in the moral or philosophical sense. They operate within the framework of their programming and the data provided to them, without an understanding of the moral weight of their ”decisions.”

However, the harm principle can still be relevant to the use of machine learning models insofar as it applies to the designers, operators, and users of these models. While the models themselves do not have agency yet, their applications can indeed cause harm—for instance, through biased decision-making processes (eg COMPAS), invasions of privacy (eg Data Extraction Attacks), or other negative impacts on individuals or groups. In this sense, the autonomy of those who deploy machine learning models is constrained by the harm principle such that they are morally and, increasingly, legally obligated to ensure that their use of these technologies does not harm others. This perspective suggests that while machine learning models do not possess agency in the sense required to apply the harm principle directly to them, the principle is highly relevant to their deployment and use. The harm principle should serve as an ethical guideline for developers, users, and regulators to prevent harm that the use of such models might cause, thus indirectly constraining personal autonomy in the development and application of AI technologies.