HW 6

John Yuan

11/17/2024

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.*)

Student Input Gradient Descent and Stochastic Gradient Descent are both optimization methods used to minimize loss functions in machine learning models. Gradient Descent computes the gradient of the loss function using the entire training dataset and updates the model parameters in one precise step, leading to stable and consistent convergence but can be slow with large datasets. In contrast, Stochastic Gradient Descent updates the model parameters more frequently by using the gradient from a single randomly selected training example at each step, resulting in faster and more scalable updates that introduce some randomness, which can help the algorithm "escape local minima and converge to a globally optimal solution" faster.

# 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.)

In the second equation, each client k updates its local model parameters. When we aggregate these updates by taking a weighted sum based on the proportion of data each client has, we get:

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

Distributing the weights inside the sum, this becomes:

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

Since the weights sum to 1 $\sum_{k=1}^{K} \frac{n_k}{n} = 1$ the equation simplifies to:

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

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 equation for FedAve is more intuitive because it has an update for each of the K clients. In FedAve, each client independently adjusts the global model based on its local data by performing a local gradient descent step and evaluating the gradient of its loss function. After these individual updates, the central server collects all the modified parameters from the clients and combines them by taking a weighted average, where each client's contribution is proportional to the size of its dataset. This step-by-step process makes it easier to understand how federated learning uses decentralized data sources to combine and improve a global model without directly sharing sensitive local data.

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

Randomized response achieves ϵ-differential privacy by ensuring that the probability of any specific output does not change significantly whether an individual's true response is "Yes" or "No." According to the definition of e-differential privacy, a randomized algorithm is ϵ-differentially private if, for any two neighboring datasets differing by one individual's data and for any possible output, the ratio of the probabilities of that output under the two datasets is bounded by e^{ϵ}. To prove, recall the class example, an individual first flips a fair coin: if it lands heads, they answer truthfully; if tails, they flip again and respond "Yes" with a probability of 1/2 or "No" with a probability of 1/2, regardless of their true answer. When comparing two neighboring datasets, one where the individual's true answer is "Yes" and the other "No", the probability of responding "Yes" changes from 3/4 to 1/4, and "No" changes from 1/4 to 3/4. The maximum ratio of these probabilities is 3, which corresponds to setting ϵ to the ln of 3

$\epsilon = ln(3)$.

This ensures that the presence or absence of any single individual's true response has only a limited and controlled impact on the overall output, thereby satisfying the requirements of ϵ-differential privacy.

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 states that individuals should be free to act as they wish unless their actions cause harm to others. This principle attempts to uphold autonomy while ensuring that one person's freedoms do not infringe upon the rights or well-being of others, balancing individual freedom with societal protection.

Currently, the harm principle is not directly applicable to machine learning models themselves because these models lack independence in that they do not possess consciousness, intent, or the ability to make decisions by themselves. However, the principle remains relevant in how individuals and organizations develop and use these models. The decisions made by developers and users can lead to significant harms such as discrimination of protected groups and privacy violations. Applying the harm principle involves regulating the responsible use of ML technologies to prevent such harms, ensuring that their deployment respects individuals' rights and societal well-being. To sum, while ML models do not have agency, the harm principle guides the ethical governance of their human developers to prevent harm to others.