HW 6

Hailey Flo

04/10/2024

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.*)

Both gradient descent and stochastic descent are used for calculating the gradient of the loss function. They iterate through training data to calculate the gradient, then adjust the weights of the parameters to minimize loss. These two types of gradient descent differ in the iteration process.

Normal gradient descent iterates through ALL the samples in the training data. It uses the entire training dataset to compute the gradient for each iteration, and the parameters are updated only after all the data has been iterated through. This requires a lot of computational power and is therefore better suited for small datasets. But because all of the data is included, the loss function will be calculated with greater accuracy. The update step for gradient descent is:

$$\theta_{i+1} = \theta_i - \alpha \cdot \nabla F_k(\theta)$$

Stochastic gradient descent does not iterate through every datapoint, rather only through samples taken from the training data. These samples are random subsets of the data. In stochastic, the pararmeter weights are updated after each sample, so less data needs to be computed per iteration. This makes it the preffered method for large datasets, as SGD will converge faster compared to GD. There are however, two main cons to using stochastic gradient descent. The first is that it can induce more noise in your loss function if you are not careful about setting your alpha, or learning rate. Second, SGD will not be as accurate in computing the overall loss function because the data has been subset. The update step for stochastic gradient descent is:

$$\theta_{i+1} = \theta_i - \alpha \cdot \nabla F_k(\theta; x^{(i)}, y^{(i)})$$

The resulting difference between the updates of GD and SGD is that GD is only able to find the local minimum of descent, whereas SGD is able to calculate this minimum globally. Because SGD pulls random samples from the training data, it considers multiple sections of the gradient to be able to view the bigger picture. Gradient descent will iterate consecutively through the model data and terminate the process once it recognizes a single minimum. Therefore if the function has multiple minimums, GD will not be able to recognize them all on a global level. Gradient descent gets bogged down in a local convex while stochastic gradient descent can be used to find the overall function minimum globally.

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.*)

If we substitute the first equation of ω_{t+1}^k into the second equation, it reduces as follows:

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

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

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

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

The reduced equation matches the formulated of the `FedAve` algorithm exactly, proving that the two formulations are equivalent.

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 updates each client (k) individually before including it in the collective update of the entire dataset, making it more intuitive. Because we can see the exact formulation of how each client is updated, it is easier to understand the changes being implemented. The original `FedAve` algorithm is less clear about local vs global parameter updates. The second formulation, however, distinguishes between ω_{t+1}^k and ω_{t+1}, emphasizing an update that occurs on both the local level then the global level. First the local model's parameters are updated by iterating through the local data, and second the updates from this iteration are averaged into the global data.

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 be 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 an individual's autonomy should extend up until its use results in the objective harm of another moral agent. This means that people do not have unlimited autonomy—they cannot do anything that infringes upon the rights of another person. Essentially, personal autonomy must be kept in check to ensure that all persons are given an equal amount of autonomy on the whole.

The harm principle is currently applicable to machine learning models because algorithmic results are causing harm to the individuals who's data is being used. One example of this is the COMPAS dataset, wherein people's personal data was being given to the ML model to determine whether or not to offer parole. In many instances, people were denied parole because of the results of the algorithm claiming them as likely re-offenders. These individuals were not asked to provide consent for using their personal data, nor can a case be argued that tacit consent was given due to the fact that they never willingly opted into joining the model. A large number of people were directly harmed by the ML model by being forced to remain in prison, illustrating that the COMPAS algorithm was indeed infringing on their personal autonomy. Furthermore, the use of this model in the court of law shows that it was given enough agency to sway the opinion of a judge and make serious decisions on people's life. Knowing that this ML model only has an accuracy rate of 61%, nearly 40% of the individuals in this scenario are being caused unjustifiable harm as a result of the algorithm to some extent. Not receiving consent from the model's users worsens the situation and supports that autonomy of ML algorithms is violating the harm principle. Machine learning models are plainly influencing important societal decisions, giving them enough agency to limit the autonomy of the users of said algorithms.

Since ML models have enough autonomy to make decisions about human lives, it can also be argued that humans are able to infringe upon the rights of these models. In principle, society has collectively decided to trust machine algorithms and give them decision making rights so therefore these models also have a right to protect themselves from harm. In practice, however, it is important to establish that these models are inhuman entities. ML models have personal autonomy, but not moral autonomy. They don't have the ability to think freely and make their own decisions; the decisions the algorithm provides are essentially just the results the developers have allowed it to give. Users cannot 'harm' these models because they do not feel or understand harm. Once these models evolve to a point that they are free-thinkers, understand societal consequences and the repercussions of their results, then the harm principle can be considered. Something like generative AI that uses its own developed models could potentially possess rights to its own data, seeing as no human took part in its formation, causing the harm principle to be applicable.