HW 6

Collin Register

1/21/2024

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.*)

Gradient descent uses the entire training sample in each iteration of the algorithm whereas stochastic gradient descent uses part of the training sample. Stochastic gradient descent has more frequent updates than gradient descent because the weights are updated after each sample. The update rule for SGD is theta i plus 1 equals theta i minus alpah Df(theta,Xi,Yi). For gradient descent the update step is theta i +1 equals theta minus alpha times gradient of the loss function. The stochastic gradient updates are more complicated and could take longer since the have to be done more frequently.

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.*)

Plugging in ω_{t+1}^k into $w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$ we get : $w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} (\omega_t - \eta \nabla F_k(\omega_t))$ then: $w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} (\omega_t) - \sum_{k=1}^{K} \frac{n_k}{n} (\eta \nabla F_k(\omega_t))$ next: take a partial derivative with respect to $\nabla F_k(\omega_t)$ resulting in $w_{t+1} = (\omega_t) - \sum_{k=1}^{K} \frac{n_k}{n} (\eta \nabla F_k(\omega_t))$ which is the original formulation.

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 is more intuitive because it allows the update to be made based on each individual k and weighted based on the size of the gradient. Looking at it on the individual level gives us an algorithm that is easier to understand because we can see each step and how each individual effects the algorithm.

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 states that autonomy should extend exactly up until the point where it would result in the harm of another agent. The example we used in class was "my right to swing my fist ends at your face". This principle limits personal autonomy because it prevents people from having complete free will and limits them from harming others with their own free will. I think that Machine learning models have not achieved enough agency to be limited in this way because while they make complex decisions, they cannot make those decisions with moral considerations. ML models also cannot be limited by the harm principle because there is not a way to hold the algorithm accountable but rather those in charge of the algorithm would need to hold this responsibility.