DS 595/ MA 590

Optimization for Deep learning & ML

Homework 2

1. In this problem you will prove a special case of the universal approximation theorem for 1-dimensional functions.

   Suppose that $f : R \to R$ is a continuous and differentiable function (If it makes it easier for you, you may also assume that f is second-order differentiable)).

   (a) Prove that there exists convex functions $g : R \to R$ and $h : R \to R$ such that $f(x) = g(x) - h(x)$ for all $x \in R$. (hint: If I give you a function f, then how would you construct g and h?)

   (b) Prove that for any continuous and differentiable function $f : [0, 1] \to R$ with continuous derivative, and any $\epsilon > 0$, there is a wide enough neural network $N : R \to R$ with just 2 hidden layers and RELU activation functions, and a choice of weights for this neural network, such that $|N(x) - f(x)| < \epsilon$ for all $x \in [0, 1]$. (hint: it might help to first prove that f and its first derivative are uniformly bounded above and below on $[0, 1]$. You can prove this fact by applying the extreme value theorem from high school calculus.)

2. In this problem you will use the code from https://nextjournal.com/gkoehler/pytorch-mnist to train a neural network to classify handwritten digits from the MNIST dataset. Try trainining the model using different optimizers (SGD, SGD with Momentum, and ADAM), and different hyper-parameters for the learning rate (and different hyperparameter for the momentum when you are doing momentum SGD).

(a) Plot the testing and training error for the "best" choice of parameters for ADAM, SGD, and SGD with momentum.

(b) Which method has the best training error? Which has the best testing error? What hyperparameter values work best for the different methods?

(c) For this neural network, you may get a lower testing error than training error. How is this possible?

3. Consider a second-order differentiable function $f : R^d \to R$. You have access to a "zeroth-order" oracle which lets you compute the value $f(x)$ of this function at any point $x \in R^d$, and also a "first-order" oracle which lets you compute the gradient $\nabla f(x)$ of this function at any point $x \in R^d$.

(a) Come up with a method to compute an approximation to the Hessian $\nabla^2 f(x)$ which uses $O(d^2)$ function evaluations (and no gradient evaluations). (For any $x \in R^d$, and any $\epsilon > 0$, you should be able to use your method to approximate the Hessian to within error at most ϵ.)

(b) Come up with a method to compute an approximation to the Hessian $\nabla^2 f(x)$ which uses at most $O(d)$ gradient evaluations (and no function evaluations). (For any $x \in R^d$, and any $\epsilon > 0$, you should be able to use your method to approximate the Hessian to within error at most ϵ.)

4. In this problem you will design a fast approximation to Newton's method.

(a) Design an *approximation* to Newton's method where each step requires only $O(1)$ gradient evaluations and storage for $O(d)$ numbers in memory to implement. (Hint: Try using an optimization algorithm, such as gradient descent, to solve the system of equations $\nabla f(x) = [\nabla^2 f(x)]z$ for the vector z. What minimization problem would give you the solution z to this equation? Each step of the optimization algorithm you use to solve this minimization problem should require you to only compute matrix-vector products and not the entire Hessian matrix $\nabla^2 f(x)$.)

(b) Your method will probably only be a good approximation to Newton's method for some classes of functions. For what classes of functions do you expect your method to work well?

5. We learned in class that convolution operations on one-dimensional "images" (e.g. sound files) can be represented as a circulant matrix. And we also learned in class that convolution operations on two-dimensional images can be represented by block-circulant matrices.

(a) What type of matrix corresponds to convolution operations on a three-dimensional image, like an MRI scan?

(b) Design a convolution kernel which detects edges in a three-dimensional image (for example, you could use this to detect the boundary of an organ in the MRI scan of a person) (Hint: take a look at the kernel examples in this wikipedia article `https://en.wikipedia.org/wiki/Kernel_(image_processing)` , which allow you to perform different operations like edge detection, sharpening, and smoothing, on two-dimensional images. Then try to use these ideas to make an edge detection kernel for a three dimension "image".)