50.021 -AI

Alex

Week 03: Better gradients

[The following notes are compiled from various sources such as textbooks, lecture materials, Web resources and are shared for academic purposes only, intended for use by students registered for a specific course. In the interest of brevity, every source is not cited. The compiler of these notes gratefully acknowledges all such sources.]

in class Problem 1 - Relu

Work in the file `modules_mnistsimple.py`.

1. Complete the class definitions in `modules_mnistsimple.py` that will allow us to use ReLU layers to our networks.
2. Download the `data.zip` file and place the four files in there in the same folder as the `main_mnist.py` file.
3. Use `main_mnist.py` to test your code. You can just run: python `main_mnist.py` or evaluate `tester()` in a Python shell. This should print an accuracy in the shell and also plot pictures of the highest ranked digits in each class.
4. If your code generates an error accessing the data files, then open `main_mnist.py` and in the tester function, set `mnistpath` to be the fully specified path to the folder where your data is stored.
5. print a few digits from the code
6. Once your code is working, write down the training and test accuracy on the MNIST data using this training regime, namely "classic" gradient descent with a fixed step size. We will want to compare this to some of the more sophisticated methods we will implement later.

in class Problem 2 - Adam

Work in the file `modules_mnistsimple.py`.

1. Implement the Adam method. compare the training performance on mnist to the classic gradient descent. – see the $ADAM = False$ flag

in class Problem 3 - Bonus

Port the other methods from `quadfrom.py` that you implemented, try out which works best.