Lab 6 Report - Dominic Sagers

Tuesday, December 6, 2022 6:06 PM

• Description of the network (e.g. number of layers, number of neurons per layer, etc.)
  - The composition of our neural network is quite basic as we have one hidden layer as well as an input and output layer. The input is comprised of images of 20x20 pixels in size such that there are 400 possible input features representing a greyscale value of each pixel and as such there 400 input nodes. There is also included a bias node which contains a value of 1 appended to the input matrix.
• Impact of specific parameters such as λ, number of iterations, weight initialization, etc.
  - For the lambda parameter, we use this to tune how much the model is allowed to change, when λ = 0, the model parameter has no effect but as λ is increased we see that estimated parameters are more discriminated against.
  - Regarding the number of iterations, we see that with repeated use of back-propogation we create a far more accurate classifier, but risk overfitting to our training set, so we implement tactics such as an early stop and regularization across the data.
  - We choose to randomly initialize the weights between a uniform distribution across [-.12,.12], this allows all parameters to be explored in a somewhat equal matter in the beginning of training. Over iterations of our training the weights return to more correct values.
• How does the regularization affect the training of your ANN?
  - Regularization allows us to prevent overfitting to our training data, as with this method we directly penalize any features with extreme weights that we would consider as noise by shrinking their value so that they do not affect our prediction as much. The λ value is used to tune our regularization and as such our handling of noise.
• Imagine that you want to use a similar solution to classify 50x50 pixel grayscale images containing letters (consider an alphabet with 26 letters). Which changes would you need in the current code in order to implement this classification task?
  - We would need 2500 input layer nodes, as we are dealing with 50x50 images and our number of labels is now 26 as we have 26 classifications.

• Change the value of the variable show_examples (in the python version, run the relevant block in the Jupyter one) in ex_nn, which information is provided? Did you get the expected information? Is anything unexpected there?
  - After running show_examples we see our predict function and the corresponding input image that was used in the prediction, the prediction is correct as we successfully guessed the correct output, although the image is incorrect, we display always +1 of the number which we predict and this is because of matlab indexing.
• How does your sigmoidGradient function work? Which is the return value for different values of z. How does it work with the input is a vector and with it is a matrix?
  - My sigmoid gradient function is simply $g(x)' = sigmoid(z) * \big(1 - sigmoid(z)\big)$ as explained in our handout ("sigmoid" is the method from sigmoid.py.
  - Return values for different values of z:

    Evaluating sigmoid gradient... Sigmoid gradient evaluated at [-15. -1. -0.5 0. 0.5 1. 15.] : [3.05902133e-07 1.96611933e-01 2.35003712e-01 2.50000000e-01 2.35003712e-01 1.96611933e-01 3.05902133e-07]