ANLY-590 Assignment 1

September 29th 2018

1 Feedforward: Building a ReLu 2 Layer neural network

Previously we built a network where the hidden layer included a logistic transform. Recall that logistic units have fallen from favor in deep networks because they saturate easily and are not zero-centered. Rather consider the rectified linear activation function : $h_j = \max(0, a_j)$.

1. Plot (draw) a network with:
   - 2 inputs,
   - 2 hidden layers (where the first layer contains 3 hidden units and the second contains 2 hidden units) and a
   - 3-class output (use a `softmax` function)
2. Write out the mathematical equation for this network
3. Write out the function in python, call it `ff_nn_2_ReLu(...)`
4. Suppose you have the following set of weight matrices:

$$W^{(1)} = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 0.5 \end{bmatrix} \qquad b^{(1)} = [0, 0, 1]^T$$

$$W^{(2)} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & -1 & 0 \end{bmatrix} \qquad b^{(2)} = [1, -1]^T$$

$$V = \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ -1 & -1 \end{bmatrix} \qquad c = [1, 0, 0]^T$$

   and inputs:

$$X = \begin{bmatrix} 1 & 0 & 0 \\ -1 & -1 & 1 \end{bmatrix}$$

   what are the class probabilities associated with the forward pass of each sample?

2 Gradient Descent

The Rosenbrock function is a famous non-convex function that is used to explore optimization algorithms. This simple 2-D function has some very tricky structure.

$$f(x, y) = (1 - x)^2 + 100 * (y - x^2)^2$$

1. What are the partial derivatives of f with respect to x and to y?

2. Create a visualization of the contours of the Rosenbrock function.

3. Write a Gradient Descent algorithm for finding the minimum of the function. Visualize your results with a few different learning rates.

4. Write a Gradient Descent With Momentum algorithm for finding the minimum. Visualize your results with a few different settings of the algorithm's hyperparameters.

3 Backprop

1. For the same network as in Number 1, derive expressions of the gradient of the Loss function with respect to each of the model parameters.

2. Write a function `grad_f(...)` that takes in a weights vector and returns the gradient of the Loss at that location.

3. Generate a synthetic dataset of 3 equally sampled bivariate Gaussian distributions with parameters

$$\mu_1 = (0, 2), \mu_2 = (2, -2), \mu_3 = (-2, -2) \quad ; \quad \Sigma_i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; i = 1, 2, 3$$

   that you'll use for fitting your network. Plot your sample dataset, coloring data points by their respective class.

4. Fit your network using Gradient Descent. Keep track of the total Loss at each iteration and plot the result.

5. Repeat the exercise above using Momentum. Comment on whether your algorithm seems to converge more efficiently.