ECE 417: Multimedia Signal Processing
Fall 2018

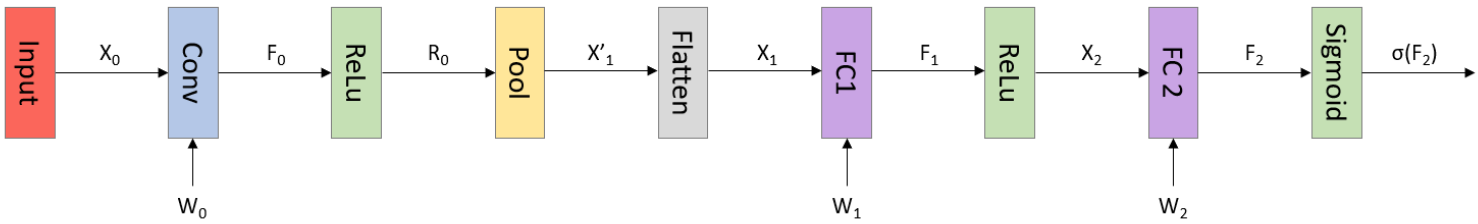
Machine Problem #3

Due: Tuesday, November 13, 2018

---

1 Overview

The goal of this machine problem is to develop and train a convolutional neural network (CNN) using mainly the NumPy library in Python. You will use the CNN to solve a binary image classification problem: distinguishing between elephant and lionfish. The images that you will be using come from the ImageNet database which includes tens of thousands of classes called synsets. The architecture used for this MP is shown below.



The sigmoid or logistic activation function is defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

The variable names above are used to denote the parameters/weights and outputs of each block for this MP. They are also defined this way in the provided skeleton code that you must complete. For example, X_0 is the batch of RGB images that will go through the convolutional layer and will output array F_0. W_0, W_1, and W_2 are the weights that will be trained using backpropagation.

Do not use a bias term for any of your weights. For a more specific configuration of the model, follow the guidelines below:

- **Convolutional Layer:** For this layer, you will use "valid" convolution to generate 3 feature maps. Here, valid convolution restricts the output values to be valid where the filter completely overlaps the input image. Filter sizes are 5×5 for each input channel and have a stride of 1. Note that the number of feature maps is independent from the number input channels.
- **Pooling Layer:** This is going to be a max pooling layer and you will pool along valid regions. The receptive field has a size of 12×12 and a stride of 12. Pool each feature map independently.
- **FC Layers:** There will be 2 nodes for the first fully connected layer and 1 node for the second layer.

2 Procedure

1. **Extracting image features:** the dataset provided contains images with dimensions 100×100×3 (RGB). Normalize the pixel value domain of each image from 0-255 to 0-1. This is called feature scaling. In your report, please address why this is beneficial/necessary.

2. **Training and testing data:** The data has already been split into a training and testing set. The training set contains 2000 images and the testing set contains 800, each set divided equally among the two classes. The function for loading data is provided to you as `load_images`.

   For training your CNN, you will be using mini-batch gradient descent. This technique is widely used in machine learning since it tends to give a more stable convergence. The idea is as follows:

   (a) Shuffle the samples in your training set uniformly at random.

   (b) Separate your samples into equally sized batches. For this MP, the batch size is 16.

   (c) Perform backpropagation on each batch and update your parameters accordingly.

   (d) Go through all the batches in your training set. This will complete one epoch.

   You will fill out the code for this process in the function `batchify`.

3. **Implement forward pass of CNN:** You will write the code for the CNN described above. The function skeletons are given to help you start. To test the correctness of the forward pass, pre-trained weights are provided in `weights.npz`. Applying them to your network should give a loss of 0.232 and an accuracy of 91.75% on the test set.

4. **Implement backward pass of CNN:** In this section you will implement the code for mini-batch gradient descent. Since this is a classification problem, the error criterion/loss function used is cross-entropy. In the binary case, it is defined as follows:

$$L = \frac{1}{N}\sum_{i=0}^{N} -y_i \log(\sigma(f_{2i})) - (1-y_i)\log(1-\sigma(f_{2i}))$$

   where N is the number of samples, y_i is the class label of the i^{th} sample (0 or 1), and $\sigma(f_{2i})$ is the output of your network for the i^{th} sample.

   We have provided the function skeletons used to calculate the gradients of the trainable parameters `dL_dW2`, `dL_dW1`, `dL_dW0`. These functions take in an argument called `cache`. This argument is a dictionary defined for you in `cnn_fwd` and saves all calculations in your forward pass that are relevant for calculating the gradients. You will then write a function that trains your network for a certain number of epochs. Use a learning rate of 0.1 for training.

   To test the correctness of your gradient functions, do the following check:

   - Initialize your weight matrices to all ones.
   - pass the first and last 8 samples of your training data into your training function for one epoch (no mini-batching) and calculate the gradients for each weight matrix using your 3 functions.

- Update your weights and then sum across all axes for each of them. You should get the following for each weight matrix: $W_2 = -834.63$, $W_1 = -452.63$, $W_0 = -611.63$.

We have provided you with a code skeleton for testing this part. Note that if you get the above values, there may still be a chance that your implementation is incorrect. If this is the case, the cause is most likely an incorrect backward pass through a ReLu block.

3 Experiments

1. Once you are confident in the implementation of your model, begin the training process. Set the learning rate to 0.1 and initialize your parameters randomly using a normal distribution $\mathcal{N}(0, 0.05^2)$. You can use `np.random.randn` for this. After 10 epochs, expect to see a test accuracy of 82-92%. It took me around 30 seconds per epoch to train. Record the test accuracy at the end of each epoch and graph your results. What do you notice? **Note:** If your accuracy is stuck at exactly 0.5 for the first couple epochs and your forward/backward passes are correct, try restarting the training process. This should not happen too often, but given the small amount of data and parameters, the model can be unstable at times.

2. Apply your newly trained filters (W_0) on input images using convolution. Display your results in your report. What do you notice? What features were learned?

3. Graph the confusion matrix of your test set. Which class did your model get confused by the most? Observe the images that the model predicted incorrectly. What kind of features do you think confused the model?

4. EXTRA CREDIT: [Up to 20%] There are several different techniques that are used on neural networks to increase performance. Change up the architecture of your model by adding more advanced ML techniques. Choose up to 2 additions for up to 10% each, giving a max of 20%. Extra credit will be awarded for the functionality of your additions and the difficulty of implementation. It is okay if it does not increase your test accuracy. Just be sure to prove in your report that your additions are functional by supplying evidence. You can choose techniques from the following list:

    - L1/L2 regularization
    - Data augmentation
    - Dropout
    - Adding momentum/decay to gradient descent.
    - Other optimizers: AdaGrad, RMSprop, Adam
    - You can suggest other techniques by making a post on Piazza

4 Notes

- You can use `scipy.signal.convolve2d` for convolution in the forward and backward pass. **Hint:** Backprop through a convolutional layer is also convolution.

- Save a binary mask that represents the max indices during pooling. This will be used during backpropagation. For any ties during pooling, choose the first occurence. This can be done easily using `np.argmax`.

5 Submission

You will submit (1) a report in PDF format, and (2) a zip file containing your code along with a Readme file to Compass. You must name your report as `<Lastname>_<Firstname>_report.pdf` and your zip file as `<Lastname>_<Firstname>_code.zip`.

Please use the following rubric when formatting your report:

- **Introduction**: Spend a few sentences describing the data and learning algorithm used in this MP.
- **Methods:** Choose at least two equations used in your model and spend a few sentences describing them in more detail.
- **Results:** Display all your figures from your experiments in this section (accuracy vs. epoch, convolution on sample images, confusion matrix).
- **Discussion:** Provide insight on your results and address any questions asked in the assignment.
- **Extra Credit:** All extra credit work will go in this section.

If you are working as a team, only one person should upload the report. Please make sure t hat the title page includes the names of all the team members.