Assignment 2

CS6360: Advanced Topics in Machine Learning
IIT-Hyderabad
Jan-Apr 2020

Max Marks: 35
Due: 29th Mar 2020 11:59 pm

This homework is intended to cover the following topics:

- Deeper understanding of backpropagation/implementation of neural networks
- Familiarity with Colab, iPython Notebooks, Convolutional Neural Networks
- Topics in basics of image processing/computer vision

Instructions

- Please use Google Classroom to upload your submission by the deadline mentioned above. Your submission should comprise of a single file (ZIP), named `<Your_Roll_No>_Assign2`, with all your solutions.
- For late submissions, 10% is deducted for each day (including weekend) late after an assignment is due. Note that each student begins the course with 12 grace days for late submission of assignments (with a max of 7 per submission). Late submissions will automatically use your grace days balance, if you have any left. You can see your balance on the CS6360 Marks and Grace Days document under the course Google drive (soon to be shared).
- Please use PYTHON for the programming assignments.
- Please read the department plagiarism policy. Do not engage in any form of cheating or plagiarism - if we find such behavior in your submission, both receiver and giver will be imposed with severe penalties. Please talk to instructor or TAs if you have concerns.

1 Programming Questions

1. ***(4 + 2 + 3 + 2 + 3 = 14 marks)*** In the last assignment, you tweaked given code to understand optimization hyperparameters that work for a given problem. In this assignment, we will get our hands dirtier, and write our own code for backpropagation to get a better understanding.

   We will consider the Occupancy dataset from the UCI Machine Learning Repository, which consists of experimental data used for binary classification of room occupancy (i.e., room

is occupied versus empty) based on temperature, humidity, light, and CO2 sensors. The training and test data sets are each collected over a week period. Information about the data set and how it has been used in academic publications can be found here.

The data set includes time stamps with date and hour/minute/second within the day. You are not to use time stamp features for predicting occupancy. Since this is a commercial office building, the time stamp is a strong predictor of occupancy. Rather, the goal is to determine whether occupancy can be sensed from: (1) temperature, expressed in degrees Celsius, (2) relative humidity, expressed as a %, (3) light, in lux, (4) CO2, in ppm, and (5) the humidity ratio, which is derive from the temperature and the relative humidity.

The training and test data are provided with this assignment. There are 8144 training examples and 9753 test examples.

(a) Write your own Python code (no PyTorch/TensorFlow) to implement a feedforward neural network with a single hidden layer. You should have 5 input units, H hidden units, and 1 output unit. Write your own code to train the network with backpropagation. Write your code so that it loops over training epochs, and within a training epoch, it chooses mini-batches of size N, where N can range from 1 to the total number of examples in the training set. As a way of testing your code, set the learning rate very small and set N to be the training set size (i.e., you're doing batch training). In this situation, you should be guaranteed that the error monotonically decreases over epochs of training. Decide what error function you wish to use. Two obvious candidates are squared error and cross entropy. Report which you have picked.

(b) As a measure of baseline performance, use the training set to determine the constant output level of the network, call it C, that will minimize your error measure. That is, assume your net doesn't learn to respond to the inputs, but rather gives its best guess of the output without regard to the input. Then for any example where the target is 0 the network will output C and for any example where the target is 1 the network will output C. Using your error function, solve for C and compute the baseline error. Report the baseline error.

(c) Using a network with H=5 hidden units, and mini-batches of size N=100, select a learning rate (or a learning rate schedule) that results in fairly consistent drops in error from one epoch to the next, make a plot of the training error as a function of epochs. On this graph, show a constant horizontal line for the baseline error. If your network doesn't drop below this baseline, there's something going awry. For now, train your net until you're pretty sure the training error isn't dropping further (i.e., a local optimum has been reached). Report the learning rate (or learning rate schedule) you used to produce the plot. Report training and test set performance in terms of % examples classified correctly.

(d) Now train nets with varying size, H, in 1, 2, 5, 10 20. You may have to adjust your learning rates based on H, or use one of the heuristics in the text for setting learning rates to be independent of H. Decide when to stop training the net based on training set performance. Make a plot, as a function of H, of the training and test set performance in terms of % examples classified correctly.

(e) See how much adding information about time of day helps the network. Add a new set of inputs that represent the time of day. (Don't add information about day of week or absolute date.) Determine an appropriate representation for the time of day.

Describe the representation you used. For example, you might add one unit with a value ranging from 0 to 1 for times ranging from 00:00 to 23:59. Report the representation you selected. Train your net with H=5 hidden and compare training and test set performance to the net you built in part (d).

2. ***(5 + 2 + 2 = 9 marks)*** In this problem, we will train a convolutional neural network for a task known as image colorization. That is, given a greyscale image, we wish to predict the colour at each pixel. This is a difficult problem for many reasons, one of which being that it is ill-posed: for a single greyscale image, there can be multiple, equally valid colourings.

   We recommend you to use Colab (https://colab.research.google.com/) for this assignment. From the assignment zip file, you will find two python notebook files: `colour_regression.ipynb`, `colourization.ipynb`. To setup the Colab environment, you will need to upload the two notebook files using the upload tab at https://colab.research.google.com/.

   We will use the CIFAR-10 data set, which consists of images of size 32×32 pixels. For most of the questions, we will use a subset of the dataset. The data loading script is included with the notebooks, and should download automatically the first time it is loaded. If you have trouble downloading the file, you can also do so manually from the provided `cifar-10-python.tar.gz`. To make the problem easier, we will only use the "Horse" category from this data set.

   (a) **Colorization as Regression:** Image colorization can be posed as a regression problem, where we build a model to predict the RGB intensities at each pixel given the greyscale input. In this case, the outputs are continuous, and so mean-squared error can be used to train the model. A set of weights for such a model is included with the assignment. In this question, you will get familar with training neural networks using cloud GPUs. Read the code in `colour_regression.py`, and answer the following questions.

      i. *(1 mark)* Describe the model `RegressionCNN`. How many convolution layers does it have? What are the filter sizes and number of filters at each layer? Construct a table or draw a diagram.

      ii. *(2 marks)* Run all the notebook cells in `colour_regression.ipynb` on Colab (No coding involved). You will train a CNN, and generate some images showing validation outputs. Now, retrain a couple of new models (adding new layers, removing some, etc - any changes of your choice) using different numbers of training epochs. You may train each new models in a new code cell by copying and modifying the code from the last notebook cell. Comment on how the results (output images, training loss) change as we increase or decrease the number of epochs, and compare with the model already provided to you (without your changes).

      iii. *(1 mark)* A color space[1] is a choice of mapping of colors into three-dimensional coordinates. Some colors could be close together in one color space, but further apart in others. The RGB color space is probably the most familiar to you, but most state-of-the-art colorization models do not use RGB color space. The model used in `colour_regression.ipynb` computes squared error in RGB color space. Why could using the RGB color space be problematic?

[1] https://en.wikipedia.org/wiki/colour_space

iv. *(1 mark)* Most state-of-the-art colorization models frame colorization as a classification problem instead of a regression problem. Why? (Hint: what does minimizing squared error encourage?)

(b) **Colorization as Classification:** We will select a subset of 24 colors and frame colorization as a pixel-wise classification problem, where we label each pixel with one of 24 colors. The 24 colors are selected using k-means clustering over colors, and selecting cluster centers. This has already been done for you, and cluster centers are provided in `colour/colour_kmeans*.npy` files. For simplicity, we still measure distance in RGB space. This is not ideal but reduces the dependencies for this assignment. Open the notebook `colourization.ipynb` and answer the following questions.

i. *(1 mark)* Complete the model `CNN` on `colourization.ipynb`. This model should have the same layers and convolutional filters as the original model on `RegressionCNN`, with the exception of the output layer. Continue to use PyTorch layers like `nn.ReLU`, `nn.BatchNorm2d` and `nn.MaxPool2d`, however we will not use `nn.Conv2d`. We will use our own convolution layer `MyConv2d` included in the file to better understand its internals (which we will explore in a later assignment).

ii. *(1 mark)* Run main training loop of CNN in `colourization.ipynb` on Colab. This will train a CNN for a few epochs using the cross-entropy objective. It will generate some images showing the trained result at the end. How do the results compare to the previous regression model?

(c) **Some Conceptual Questions:**

i. *(1 mark)* In the `RegressionCNN` model, `nn.MaxPool2d` layers are applied after `nn.ReLU` activations. Comment on how the output of CNN changes if we switch the order of the max-pooling and ReLU.

ii. *(1 mark)* The loss functions and the evaluation metrics in this assignment are defined at pixel-level. In general, these pixel-level measures correlate poorly with human assessment of visual quality. How can we improve the evaluation to match with human assessment better? (*Hint:* You may find this paper useful for answering this question.)

2 Theory Questions

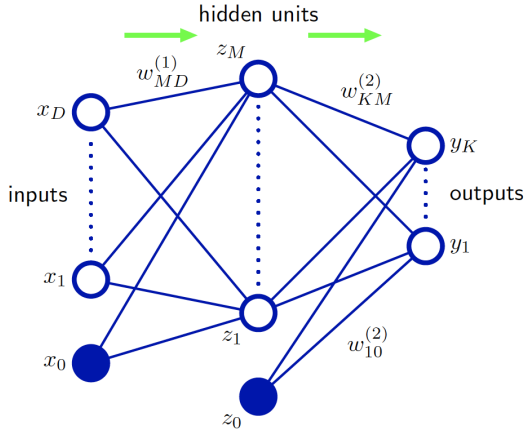
You can type this in or handwrite and submit your scanned solutions in PDF format.

1. ***(2 marks)*** Consider a two-layer network of the form shown the figure below. Let us add extra parameters corresponding to skip-layer connections that go directly from the inputs to the outputs. Write down the equations for the derivatives of the error function with respect to these additional parameters.

2. ***(4 marks)*** Consider an error function $E(\mathbf{w})$ of a neural network, and the Taylor series approximation of $E(\mathbf{w})$ around a particular weight configuration $\widetilde{\mathbf{w}}$ as below (ignoring higher order terms):

$$E(\mathbf{w}) \approx E(\widetilde{\mathbf{w}}) + (\mathbf{w} - \widetilde{\mathbf{w}})^T \nabla E|_{w=\widetilde{w}} + \frac{1}{2}(\mathbf{w} - \widetilde{\mathbf{w}})^T \mathbf{H} (\mathbf{w} - \widetilde{\mathbf{w}})$$

where $\mathbf{H}$ is the Hessian. Show that as a consequence of the symmetry of the Hessian matrix $\mathbf{H}$, the number of independent elements in the quadratic error function $E(\mathbf{w})$ is given by $W(W+3)/2$.



3. ***(1 + 1 + 1 + 1 + 2 = 6 marks)*** **Linear Filters:** In class, we introduced 2D discrete space convolution. Consider an input image $I[i, j]$ and a filter $F[i, j]$. The 2D convolution $F * I$ is defined as

$$(F * I)[i, j] = \sum_{k,l} I[i - k, j - l]F[k, l] \tag{1}$$

   (a) Convolve the following I and F (using pen and paper). Assume we use zero-padding where necessary.

$$I = \begin{bmatrix} 2 & 0 & 1 \\ 1 & -1 & 2 \end{bmatrix} F = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} \tag{2}$$

   Please DO NOT write programs. It will also be helpful for answering question (d).

   (b) Note that the F given in Equation 2 is separable, that is, it can be written as a product of two filters: $F = F_1F_2$. Find F_1 and F_2. Then, compute $(F_1 * I)$ and $F_2 * (F_1 * I)$, i.e., first perform 1D convolution on each column, followed by another 1D convolution on each row. (Please DO NOT write programs. Do it by hand.)

   (c) Prove that for any separable filter $F = F_1F_2$
   $F * I = F_2 * (F_1 * I)$
   Hint: Expand Equation 1 directly.

   (d) Carefully count the exact number of multiplications (multiplications only, including those multiplications due to zero-padding) involved in part (a) and part (b). which one of these requires fewer operations? You may find the computation steps you wrote down for (a) and (b) helpful here.

   (e) Consider a more general case: I is an $M_1 \times N_1$ image , and F is an $M_2 \times N_2$ separable filter.

      i. How many multiplications do you need to do a direct 2D convolution?

      ii. How many multiplications do you need to do 1D convolution on rows and columns?
      Hint: For (i) and (ii), we are asking for two functions of M_1, N_1, M_2 and N_2 here, no approximations.

      iii. Use Big-O notation to argue which one is more efficient in general: direct 2D convolution or two successive 1D convolutions?

3 Practice Exercises

NO SUBMISSION REQUIRED; PLEASE USE THIS FOR PRACTICE AND LEARNING.

1. **Basics of Python-OpenCV-I:** Suppose you are given a 100×100 matrix A representing a grayscale image. Write a few lines of code to do each of the following. Try to avoid using loops.

   (a) Plot all the intensities in A, sorted in decreasing value. (Note, in this case, we don't care about the 2-D structure of A, we only want to sort all the intensities in one list.)

   (b) Display a histogram of A's intensities with 20 bins.

   (c) Create and display a new color image the same size as A, but with 3 channels to represent R,G and B values. Set the values to be bright red (i.e., $R = 255$) wherever the intensity in A is greater than a threshold t, and black everywhere else.

   (d) Generate a new image, which is the same as A, but with A's mean intensity value subtracted from each pixel (without loops).

2. **Basics of Python-OpenCV-II:** Write functions to do each of the following to an input grayscale image, and then write a script that loads an image, applies each of your functions to the input image, and displays the output results. Label each subplot with title. (Sample images have been supplied with the assignment. Please submit answers using at least one of those images.)

   (a) Map a grayscale image to its "negative image", in which the lightest values appear dark and vice versa.

   (b) Map the image to its "mirror image", i.e., flipping it left to right.

   (c) Swap the red and green color channels of the input color image.

   (d) Add or subtract a random value between $[0, 255]$ to every pixel in a grayscale image, then clip the resulting image to have a minimum value of 0 and a maximum value of 255.

3. **Canny Edge Detector:** Suppose the Canny edge detector successfully detects an edge. The detected edge (shown as the red horizontal line in Figure 1) is then rotated by θ, where the relationship between a point on the original edge (x, y) and a point on the rotated edge (x', y') is defined as

$$x' = x\cos\theta \ ; \ y' = y\sin\theta$$

   (a) Will the rotated edge be detected using the same Canny edge detector? Provide either a mathematical proof or a counter example. *Hint:* The detection of an edge by the Canny edge detector depends only on the magnitude of its derivative. The derivative at point (x,y) is determined by its components along the x and y directions. Think about how these magnitudes have changed because of the rotation.

   (b) After running the Canny edge detector on an image, you notice that long edges are broken into short segments separated by gaps. In addition, some spurious edges appear. For each of the two thresholds (low and high) used in hysteresis thresholding, state how you would adjust the threshold (up and down) to address both problems. Assume that a setting exists for the two thresholds that produce the desired result.

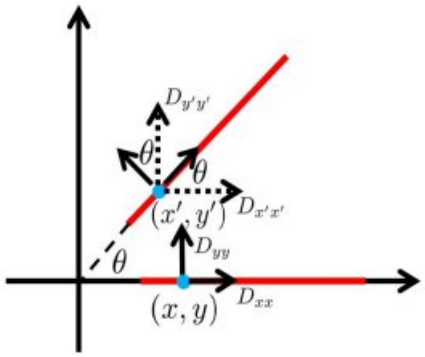


Figure 1: Canny Edge Detector