Mindren Lu 1/7/19 – 2/1/19

Argonne Externship 2019: Documentation and Writeup

Code:

* Find\_neighbors.py
  + Given a POSCAR/CONTCAR and a central atom, returns the coordinates of all neighboring atoms within a specified radius
* Parse\_POSCAR.py
  + Given a POSCAR/CONTCAR, performs calculations to return relevant data and information on the atoms in the structure
* extract\_training\_data.py
  + Given POSCARs/CONTCARs and data files, generates the spectra and labels used in the testing of all models
  + There was originally an atoms\_dist.py, but it was very short (a one-line function to calculate the distance between two atoms), so I just moved the code from the file to the top of this one
* ml\_helpers\_new.py
  + A collection of helper code to simplify the implementation of the neural networks
  + Initialization, forward propagation, cost functions, mini batch generation, loading data
* xray\_ml\_new.py
  + Code used to train networks for the classification task (Fe coordination numbers)
* xray\_ml\_regression.py
  + Code used to train networks for the regression task (on averaged spectra)
* xray\_ml\_multi.py
  + Code used to train networks for multi-task learning (on Fe coordination #s and charges)
* predict.py
  + Used to evaluate the accuracy of trained models on final test data

Folders:

* argonne: main folder
* charge\_reg\_models: saved Tensorflow models for the charge regression task
* coord\_models: saved Tensorflow models for the coordination number classification task
* graphs: visualizations, many of which are used in the slideshow presentation
* multi\_models: saved Tensorflow models for multi-task learning
* new\_data: all original data used (spectra, POSCARs/CONTCARs, charge files)
* parsed\_data: conversion of the original data into saved NumPy arrays which are then directly used in training and testing, saving time as we no longer need to recompute data each time
* reg\_models: saved Tensorflow models for the average coordination number regression task
* slides: various PowerPoints
* training\_outputs: the printed raw outputs during training to log cost functions, accuracies, MAPEs, etc. for future reference

Timeline:

* Prior to arrival
  + Received Python code written by Liang and familiarized myself with it
    - A neural network for classification, identifying whether the Fe coordination number of an oxygen atom in an Li3FeO3.5 structure was 0, 1, 2, 3, or 4
  + Learned about neural networks from the first two classes in Andrew Ng’s Coursera deep learning specialization
* Week of 1/7/19
  + Employee orientation on the first two days
  + Implemented mini-batch gradient descent to speed up training with larger data sets
  + Wrote helper code allowing for more efficient testing of hyperparameters
    - Allows the user to feed in lists of different hyperparameter settings and evaluate them in a grid search
    - Gives many hyperparameters the flexibility to take on different values, rather than having fixed values in the code itself
  + Changed the code to incorporate a cross-validation set, as the original code only split into a training and test set
  + Learned from the third class in Andrew Ng’s specialization, on structuring machine learning projects
  + Implemented data normalization for the input data (normalized each value in the spectra to have a mean of 0 and a variance of 1 in the training set) and batch normalization (used with a similar purpose after each hidden layer)
  + Attempted data augmentation
    - Created artificial data by averaging spectra together that had the same label and using them as additional training data
    - The network would achieve 99.5% accuracy on the test set, which was unreasonably high – suggesting that this model was fitting too well to the artificial data and would not generalize to real data
    - This augmentation was removed once Liang had completed more calculations and generated more training data
  + Set up the Mac in my cubicle to run Python/Tensorflow properly and allow for me to train networks overnight or over a weekend
  + Updated data augmentation
    - This time, this was written for a new task – one of regression, predicting the average value of the Fe coordination number among a set of 49 averaged spectra
    - This new code could randomly choose 49 spectra to average together and output as a data point, generating as much data as desired
      * Eventually would settle on using 10,000 data points to train and test the regression model
* Week of 1/14/19
  + Testing numerous combinations of hyperparameters to find the best possible model for the classification task
  + Began implementing the regression model
    - Writing the necessary helper code, as the structure is now different from the classification model
    - Switched to only using one hidden layer instead of two
    - New cost function: mean squared error, rather than cross-entropy loss
  + Fixed mistakes found in earlier code (data/batch normalization, cost function)
  + Began work on the multi-task learning model
    - With one spectrum for one oxygen atom, trying to use one network to predict both the Fe coordination number and its charge
    - The structure of this model is very different from the previous two, so it required many new functions and implementations
  + Reading papers on Google Scholar to look at work others have done for similar applications
  + Implemented recall as another measure of model performance for the classification model
    - For this application, it is important that the model correctly identify all those atoms with a label of 0 (i.e. have a low rate of false negatives)
  + Configured settings for using PuTTY between my Windows laptop and Mac at work so I would be able to train models on the Mac from home
  + Realized that I had been inefficiently computing data and models repeatedly, where I should be saving as much as possible to local files instead, both to increase efficiency and allow for ease of future reference
    - Learned how to send the output from Terminal to text files
      * At this point I had only been printing training results from the models to the console; by saving the outputs, I could go back and compare both past and present results
    - Learned how to save models in Tensorflow to .ckpt files
      * Rather than repeatedly retraining models, saves the trained parameters so they can be instantly used to evaluate any new data
  + Once I received the charge data, I first tried tuning my existing regression model to make predictions on this data instead to ensure that there was enough information in a spectrum that a network could make meaningful predictions from it
    - Worked quite well, with about a 2% mean average percent error
* Week of 1/21/19
  + Learned how to save NumPy arrays to .npy files
    - The previous code would need to recompute the training data every time through extract\_training\_data(), which could take many minutes; now, models can begin training as soon as the code is run
  + Created many graphs to help visualize my data
    - Histograms showing the distribution of training/cross-validation/test data
    - Computing means/standard deviations to better understand the prediction strength of the regression model
      * Mean average percent error should be compared to the standard deviation
  + Resolved issue with training the regression model
    - For a few days, I couldn’t figure out why it wasn’t working
    - Eventually realized it wasn’t because of the code, it was because when I was trying to save the output to a text file in Terminal, I needed to use a ‘-u’ in my command; otherwise the program would stall
  + Wrote code allowing for testing trained models on test data
    - Required learning about how to properly save and load models in Tensorflow
    - For a while, I couldn’t figure out how to make prediction work at test-time with batch normalization involved
      * During training, batch normalization continuously updates a moving mean and average; this is what is used at test time; however, I didn’t know how to properly use these values at test time
      * It was because I was trying to use a higher-level function, but I needed to use a lower-level one to be able to use the saved values for the moving mean/average properly
  + Generated more graphs to depict the results of the classification/regression models, and screenshotted the appropriate outputs from Terminal to use in my presentation
  + Wrote code to create graphs showing the effect of changing certain hyperparameters on the cross-validation set accuracy in order to demonstrate the effectiveness of my final chosen hyperparameters
  + Cleaned up code: removing redundancies, adding necessary comments, improving ease of use and readability
  + Repeated many of the tasks I had done for the classification model for the regression model: creating graphs, allowing for test data to be evaluated using trained models
  + Wrote code to extract ‘real’ averaged spectra data (i.e. averaging the 49 spectra together that belong to a single POSCAR/CONTCAR)
    - These are the only data points we have that are not entirely artificial (i.e. created by averaging any random 49 spectra together), so it’s important to test the models on this data because it’s the best indicator of if the models will generalize to real-world examples
* Week of 1/28/19
  + Went through the same process as with the averaged spectra regression, but this time to finalize my results with charge data
    - Wrote functions to accommodate the charge data with the existing code
    - Trained many models to find the best setting of hyperparameters
    - Identified outliers to understand a common connection between them
    - Produced outputs and visualizations to document my results
  + Returned to the multi-task model, going through a similar process as before
    - Structure: one shared hidden layer, then one task-specific hidden layer per task
    - Adjusted many settings of hyperparameters to optimize the model
    - Documented my results
  + Finalized all results and visualizations for my previous models
  + Organized my existing folders and files across both the Mac at work and my laptop
  + Completed a slideshow presentation showcasing my findings, and typed out the details of my progress and implementations