The University of Texas at Austin
Department of Electrical and Computer Engineering

**EE380L: Data Mining — Spring 2019**

Problem Set Two

C. Caramanis Due: Monday, March 11th, 2019.

---

**Problem 1: Starting in Kaggle**.
On February 27th, we are opening a Kaggle competition made for this class. In that one, you will be participating on your own. This is an intro to get us started, and also an excuse to work with regularization and regression which we have been discussing. Thus, *even though this problem set is not due until March 11, it would be a great idea to get experience on Kaggle before we start the Kaggle competition on February 27th.*

1. Lets start with our first Kaggle submission in a playground regression competition. Make an account to Kaggle and find `https://www.kaggle.com/c/house-prices-advanced-regression-technique`

2. Follow the data preprocessing steps from `https://www.kaggle.com/apapiu/house-prices-advanced-regr` `regularized-linear-models`. Then run a ridge regression using $\alpha = 0.1$. Make a submission of this prediction, what is the RMSE you get?
(Hint: remember to exponentiate np.expm1(ypred) your predictions).

3. Compare a ridge regression and a lasso regression model. Optimize the alphas using cross validation. What is the best score you can get from a single ridge regression model and from a single lasso model?

4. Plot the $l_0$ norm (number of nonzeros) of the coefficients that lasso produces as you vary the strength of regularization parameter alpha.

5. Add the outputs of your models as features and train a ridge regression on all the features plus the model outputs. This is called Ensembling and Stacking, which we discussed in class. Be careful not to overfit. What score can you get?

6. Install XGBoost (Gradient Boosting) and train a gradient boosting regression. What score can you get just from a single XGB? (you will need to optimize over its parameters). We will discuss boosting and gradient boosting in more detail later. XGB is a great friend to all good Kagglers!

7. Do your best to get the more accurate model. Try feature engineering and stacking many models. You are allowed to use any public tool in python. No non-python tools allowed.

8. Read the Kaggle forums, tutorials and Kernels in open competitions. This is an excellent way to learn. Include in your report if you find something in the forums you like, or if you made your own post or code post, especially if other Kagglers liked or used it afterwards.

9. Be sure to read and learn the rules of Kaggle! No sharing of code or data outside the Kaggle forums. Every student should have their own individual Kaggle account. This is (even) more important for live competitions of course.

10. As in the real Kaggle competition (which will be next), you will be graded based on your public score (include that in your report) and also on the creativity of your solution. In your report (**that you will submit as a pdf file**), explain what worked and what did not work. Many creative things will not work, but you will get partial credit for developing them.

**Problem 2: Starting with Neural Nets and Fast.ai**.

The goal of the next part of this problem set is to get us up and running with some modern tools for playing with Neural Networks. Specifically, we will be exploring a very powerful library built on top of PyTorch, called **Fast.ai**. *This problem set and the next one will require access to a GPU.* While Fast.ai is all open source and in principle you can download and install anywhere, doing this successfully may require considerable effort. Therefore, we *highly recommend* that you use a cloud computing service that already has Fast.ai set up. There are many options. We recommend either **colab** (free), or **paperspace** (not free, but not too expensive – I've been using a machine that costs $0.78/hour). There are other options as well.

**Note: only parts 6 and 7 below have anything to hand in (parts 1, 2 and 3 are just describing the tools and setup, and parts 4 and 5 are optional).**

1. **Paperspace**. As an example... if you want to use Paperspace, then: create an account on paperspace or a cloud service of your choice. You should make sure that you are able to install all the libraries and tools needed for Fast.ai. If you choose to do this on `paperspace.com`, then:

   (a) Create an account on paperspace.

   (b) Log in.

   (c) Go to Gradient in the toolbar in the left.

   (d) Create Notebook.

   (e) Paperspace Fast.AI 1.0 (V3).

   (f) Choose the P5000 machine ($0.78/hour).

   (g) Launch, and go to: course-v3 $\longrightarrow$ nbs $\longrightarrow$ dl1 and open up the first lesson: lesson1-pets.ipynb.

2. **Fast.ai**

   Fast.ai has created a huge library of tools that make setting up and training a neural network very easy, especially through the use of *transfer learning*. The notebook you opened in the previous exercise corresponds to the first lesson under "Practical Deep Learning for Coders." Watch this lesson, and follow along on the notebook.

3. Make sure you understand the key elements of using Fast.ai, including understanding how to access the help documentation. Specifically:

   (a) Figure out: ImageDataBunch: this is the main data structure that is used.

   (b) Figure out: create_cnn. This is the way you will create a "learner" that you will then train for some number of epochs. The key here is how easy it is to download a pre-set architecture (e.g., the notebook starts with resnet34, but has various others as well).

4. (Optional) Explain what is a residual network, and the basic motivation for using it. Also explain what are the main elements of resnet34 and resnet50. How many layers, how many neurons total, how many weights; and then anything else you want to say.

5. (Optional) Transfer learning using Fast.ai and create_cnn: Please explain how *pretrained* resnet34 is modified to get the network that the notebook ultimately trains (i.e., explain what are the last layers that are added).

6. Download a NOT pre-trained resnet34 (read the Fast.ai documentation to see how to do this), and then by playing with the number of epochs and learning rates (possibly different learning rates across layers), see how low you can get the error. Can you get below 20%?

7. And for the main part of this exercise: download (and label) your own data set of your choie, create a classification problem, and then use the main tools/ideas of this notebook to build a classifier. It does not need to be a multi-label classifier.

   For getting data, you may want to refer to the discussion here, for various tools that could be useful: `https://forums.fast.ai/t/tips-for-building-large-image-datasets/26688/36`, or the beginning part of Lesson 2.