CSC 411
Machine Learning & Data Mining
ASSIGNMENT # 1
Out: 21st September
Due: 10pm, 5th October

In this assignment we will be working with the Boston Houses dataset. This dataset contains 506 entries. Each entry consists of a house price and 13 features for houses within the Boston area. We suggest working in python and using the scikit-learn package to load the data.

Starter code written in python is provided for each question.

1 Learning basics of regression in Python (3%)

This question will take you step-by-step through performing basic linear regression on the Boston Houses dataset. You need to submit modular code for this question. Your code needs to be functional once downloaded. Non-functional code will result in losing all marks for this question. If your code is non-modular or otherwise difficult to understand, you risk losing a significant portion of the marks, even if the code is functional.

Environment setup: For this question you are strongly encouraged to use the following python packages:

- sklearn
- matplotlib
- numpy

It is strongly recommended that you download and install Anaconda 3.4 to manage the above installations. This is a Data Science package that can be downloaded from `https://www.anaconda.com/download/`.

You will submit a complete regression analysis for the Boston Housing data. To do that, here are the necessary steps:

- Load the Boston housing data from the sklearn datasets module
- Describe and summarize the data in terms of number of data points, dimensions, target, etc
- Visualization: present a single grid containing plots for each feature against the target. Choose the appropriate axis for dependent vs. independent variables. **Hint:** *use pyplot.tight_layout function to make your grid readable*
- Divide your data into training and test sets, where the training set consists of 80% of the data points (chosen at random). **Hint:** *You may find numpy.random.choice useful*

- Write code to perform linear regression to predict the targets using the training data. Remember to add a bias term to your model.
- Tabulate each feature along with its associated weight and present them in a table. Explain what the sign of the weight means in the third column ('INDUS') of this table. Does the sign match what you expected? Why?
- Test the fitted model on your test set and calculate the Mean Square Error of the result.
- Suggest and calculate two more error measurement metrics; justify your choice.
- Feature Selection: Based on your results, what are the most significant features that best predict the price? Justify your answer.

2 Locally reweighted regression (6%)

1. Given $\{(\mathbf{x}^{(1)}, y^{(1)}), .., (\mathbf{x}^{(N)}, y^{(N)})\}$ and positive weights $a^{(1)}, ..., a^{(N)}$ show that the solution to the *weighted* least square problem

$$\mathbf{w}^* = \arg\min \frac{1}{2} \sum_{i=1}^{N} a^{(i)} (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})^2 + \frac{\lambda}{2} ||\mathbf{w}||^2 \tag{1}$$

   is given by the formula

$$\mathbf{w}^* = \left(\mathbf{X}^T \mathbf{A} \mathbf{X} + \lambda \mathbf{I}\right)^{-1} \mathbf{X}^T \mathbf{A} \mathbf{y} \tag{2}$$

   where $\mathbf{X}$ is the design matrix (defined in class) and $\mathbf{A}$ is a diagonal matrix where $\mathbf{A}_{ii} = a^{(i)}$

2. Locally reweighted least squares combines ideas from k-NN and linear regression. For each new test example $\mathbf{x}$ we compute distance-based weights for each training example $a^{(i)} = \frac{\exp(-||\mathbf{x}-\mathbf{x}^{(i)}||^2/2\tau^2)}{\sum_j \exp(-||\mathbf{x}-\mathbf{x}^{(j)}||^2/2\tau^2)}$, computes $\mathbf{w}^* = \arg\min \frac{1}{2} \sum_{i=1}^{N} a^{(i)} (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})^2 + \frac{\lambda}{2} ||\mathbf{w}||^2$ and predicts $\hat{y} = \mathbf{x}^T \mathbf{w}^*$. Complete the implementation of locally reweighted least squares by providing the missing parts for q2.py.

   Important things to notice while implementing: First, do not invert any matrix, use a linear solver (numpy.linalg.solve is one example). Second, notice that $\frac{\exp(A_i)}{\sum_j \exp(A_j)} = \frac{\exp(A_i - B)}{\sum_j \exp(A_j - B)}$ but if we use $B = \max_j A_j$ it is much more numerically stable as $\frac{\exp(A_i)}{\sum_j \exp(A_j)}$ overflows/underflows easily. *This is handled automatically in the scipy package with the scipy.misc.logsumexp function.*

3. Use k-fold cross-validation to compute the average loss for different values of τ in the range [10,1000] when performing regression on the Boston Houses dataset. Plot these loss values for each choice of τ.

4. How does this algorithm behave when $\tau \to \infty$? When $\tau \to 0$?

3 Mini-batch SGD Gradient Estimator (6%)

Consider a dataset $\mathcal{D}$ of size n consisting of $(\mathbf{x}, y)$ pairs. Consider also a model $\mathcal{M}$ with parameters θ to be optimized with respect to a loss function $L(\mathbf{x}, y, \theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{x}^{(i)}, y^{(i)}, \theta)$.

We will aim to optimize L using mini-batches drawn randomly from $\mathcal{D}$ of size m. The indices of these points are contained in the set $\mathcal{I} = \{i_1, \ldots, i_m\}$, where each index is distinct and drawn uniformly without replacement from $\{1, \ldots, n\}$. We define the loss function for a single mini-batch as,

$$L_{\mathcal{I}}(\mathbf{x}, y, \theta) = \frac{1}{m}\sum_{i \in \mathcal{I}} \ell(\mathbf{x}^{(i)}, y^{(i)}, \theta) \tag{3}$$

1. Given a set $\{a_1, \ldots, a_n\}$ and random mini-batches $\mathcal{I}$ of size m, show that
$$\mathbb{E}_{\mathcal{I}}\left[\frac{1}{m}\sum_{i \in \mathcal{I}} a_i\right] = \frac{1}{n}\sum_{i=1}^{n} a_i$$
2. Show that $\mathbb{E}_{\mathcal{I}}\left[\nabla L_{\mathcal{I}}(\mathbf{x}, y, \theta)\right] = \nabla L(\mathbf{x}, y, \theta)$
3. Write, in a sentence, the importance of this result.
4. (a) Write down the gradient, ∇L above, for a linear regression model with cost function $\ell(\mathbf{x}, y, \theta) = (y - w^T\mathbf{x})^2$.
   (b) Write code to compute this gradient.
5. Using your code from the previous section, for $m = 50$ and $K = 500$ compute
$$\frac{1}{K}\sum_{k=1}^{K} \nabla L_{\mathcal{I}_k}(\mathbf{x}, y, \theta)$$
, where $\mathcal{I}_k$ is the mini-batch sampled for the kth time.

   Randomly initialize the weight parameters for your model from a $\mathcal{N}(0, I)$ distribution. Compare the value you have computed to the true gradient, ∇L, using both the squared distance metric and cosine similarity. Which is a more meaningful measure in this case and why?

   [Note: Cosine similarity between two vectors $\mathbf{a}$ and $\mathbf{b}$ is given by $\cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}||_2||\mathbf{b}||_2}$.]
6. For a single parameter, w_j, compare the sample variance, $\tilde{\sigma}_j$, of the mini-batch gradient estimate for values of m in the range [1,400] (using $K = 500$ again). Plot $\log \tilde{\sigma}_j$ against $\log m$.

Submission

Assignments must be submitted by 10:00 pm on the day it is due; a late penalty of 10% per day will be assessed thereafter (up to 3 days, then submission is blocked). We highly recommend using Python 3.6 embedded in Anaconda 3.4 to solve the programming questions. However, you are

welcome to use any programming language you like, given that your code solves the problem, and is functional and modular.

What to submit

- All work to be submitted via Markus (details to follow soon).
- All mathematical work has to be clearly marked by the question it belongs to, we highly recommend using a math editor such as MathType, Word Equations or LaTex. However, if you prefer to use paper and pen(cil), you are welcome to do so as long as your hand writing is readable. Unreadable work will result in losing the full mark of the question.
- Provide a separate file containing your code for each question. Each file name has to be indicative to the question it belongs to.
- Your code must be clearly structured with an obvious entry point.
- All graphs and plots have to be clearly marked by its question. Use readable grids whenever applicable.