Discussion April 20: Homework 1 Tutorial

1. Obtaining and Reading the Data

1.1. Obtaining the Data

- **Problem 1 (1).** The data is called *cpusmall*. The data is presented on an html page. You can copy and paste the data into a blank text document and save it as *cpusmall.txt*.

- **Problem 1 (3).** The data are *E2006.train* and *E2006.test* respectively. Start by downloading the .bz2 files. The .bz2 file extension is a compression format mainly used in Linux systems. Go here for a list of all programs in Windows and Mac that open this file extension: https://fileinfo.com/extension/bz2. After extracting the file, there is no need to rename them (they can end in the extension .train and .test).

- **Problem 2 (2).** The data is *news20*. The steps are similar to Problem 1 (3). There is also no need to rename this file (it will end in the extension .binary).

1.2. Reading the Data into Python

All of the data in this homework are in a specific format called *svmlib*. The easiest way to read the data into Python is using a function in the datasets package from *scikit-learn*. If you have not done so, you will need to install scikit-learn first in order to do this. The following example will read in *E2006.train*.

```
from sklearn import datasets

filename = "C:/Users/jstwa/Desktop/STA 141C/E2006.train"
X,y = datasets.load_svmlight_file(filename)
```

The feature matrix X is in a weird format called CSR (Compressed Sparse Row). In order to make it easier to work with, we will convert it into a regular numpy 2D array. This process involves using scipy:

```
from scipy import sparse

X_array = sparse.csr_matrix.todense(X)
```

2. Machine Learning Concepts Review

2.1. Gradient Descent

This is an optimization algorithm used to obtain a local minimum of a given function. It is popular in machine learning and is usually used when an analytical solution is not possible. Here is the algorithm in pseudocode:

init $w_i^{(0)}$ = random value for each i = 1,...p

$r_0 = \|\nabla f(\mathbf{w^{(0)}})\|$, $\quad \eta$ = fixed step size chosen in the interval $(0, 1)$

for t in range(0,200) or until $\|\nabla f(\mathbf{w^{(t)}})\| < \epsilon \cdot r_0$ **do:**

$\quad \mathbf{w^{(t+1)}} \leftarrow \mathbf{w^{(t)}} - \eta \cdot \nabla f(\mathbf{w^{(t)}})$

Gradient descent is very easy to implement. The main thing you will have to do is to (1) Find an expression for $\nabla f(w_t)$, and (2) Implement this expression in Python. I recommend doing (2) as its own function so the code will be cleaner.

2.2. Cross-Validation

5-fold Cross-validation is a way to validate your model; i.e. check its prediction power. Here is a rough pseudocode for Problem 1 (2):

```
Divide your data as evenly as possible into 5 pieces: $D_0, ..., D_4$.

for i in range(0,4) do:
     test.data = D_i
     train.data = Everything else except D_i
     w.train = ridge.regression(train.X)
     mse_i = mse(x = test.X, y=test.y, w = w.train)

mse_cv = average(mse_i, i = 0,...,4)
```

3. Deriving the Gradient for Ridge Regression

3.1. Reminder: What is a gradient? Suppose that f is a function that takes a p x 1 vector as an input and returns a single number (scalar). The gradient of f, denoted by ∇f, is given by:

$$\nabla f = \nabla f(w_1, ..., w_p) = \left[\frac{\delta f}{\delta w_1}, ..., \frac{\delta f}{\delta w_p}\right]$$

In other words, it is a vector containing all the partial derivatives for each w_i in f.

3.2. Objective function for ridge regression. For ridge regression, letting $\lambda = 1$, we wish to minimize the following function:

$$f(w_1, ..., w_p) = \frac{1}{2}\sum_{i=1}^{n}\left(x_{i1}w_1 + ... + x_{ip}w_p - y_i\right)^2 + \frac{1}{2}\left(w_1^2 + ... + w_p^2\right)$$

Let's examine the partial derivative of the above with respect to an arbitrary w_j:

$$\frac{\delta f}{\delta w_j} = \sum_{i=1}^{n}\left(x_{i1}w_1 + ... + x_{ip}w_p - y_i\right)\left(x_{ij}\right) + w_j$$

So overall, the gradient looks like this:

$$\nabla f = \left[\sum_{i=1}^{n}\left(\mathbf{x_i^T w} - y_i\right)\left(x_{i1}\right) + w_1, ..., \sum_{i=1}^{n}\left(\mathbf{x_i^T w} - y_i\right)\left(x_{ip}\right) + w_p\right]$$

$$= \left[\sum_{i=1}^{n}\mathbf{x_i^T w}\cdot x_{i1} - \sum_{i=1}^{n} y_i x_{i1} + w_1, ..., \sum_{i=1}^{n}\mathbf{x_i^T w}\cdot x_{ip} - \sum_{i=1}^{n} y_i x_{ip} + w_p\right]$$

$$= \left[\sum_{i=1}^{n}\mathbf{x_i^T w}\cdot x_{i1}, ..., \sum_{i=1}^{n}\mathbf{x_i^T w}\cdot x_{i1}\right] - \left[\sum_{i=1}^{n} y_i x_{i1}, ..., \sum_{i=1}^{n} y_i x_{ip}\right] + [w_1, ..., w_p]$$

$$= \mathbf{X^T X w} - \mathbf{X^T y} + \mathbf{w}$$

$$= \mathbf{X^T}(\mathbf{Xw} - \mathbf{y}) + \mathbf{w}$$