Homework-3

April 24, 2020

1 Exercise 1

Gradient Derivation

For $d = 1, n = 1$:

$$\nabla F(\beta) = \frac{1}{1 + e^{-yx\beta}}(e^{-yx\beta})(-yx) + 2\lambda\beta \tag{1}$$

$$\nabla F(\beta) = \frac{-yxe^{-yx\beta}}{1 + e^{-yx\beta}} + 2\lambda\beta \tag{2}$$

When $d \geq 1, n \geq 1$, we generalize (by linearity):

For the $j = 1, ..., d$ features,

$$\frac{\partial F}{\partial \beta_j} = \frac{1}{n}\sum_{i=1}^{n} \frac{-y_i x_{ij}\, exp\left(-y_i x_i^T \beta\right)}{1 + exp\left(-y_i x_i^T \beta\right)} + 2\lambda\beta_j \tag{3}$$

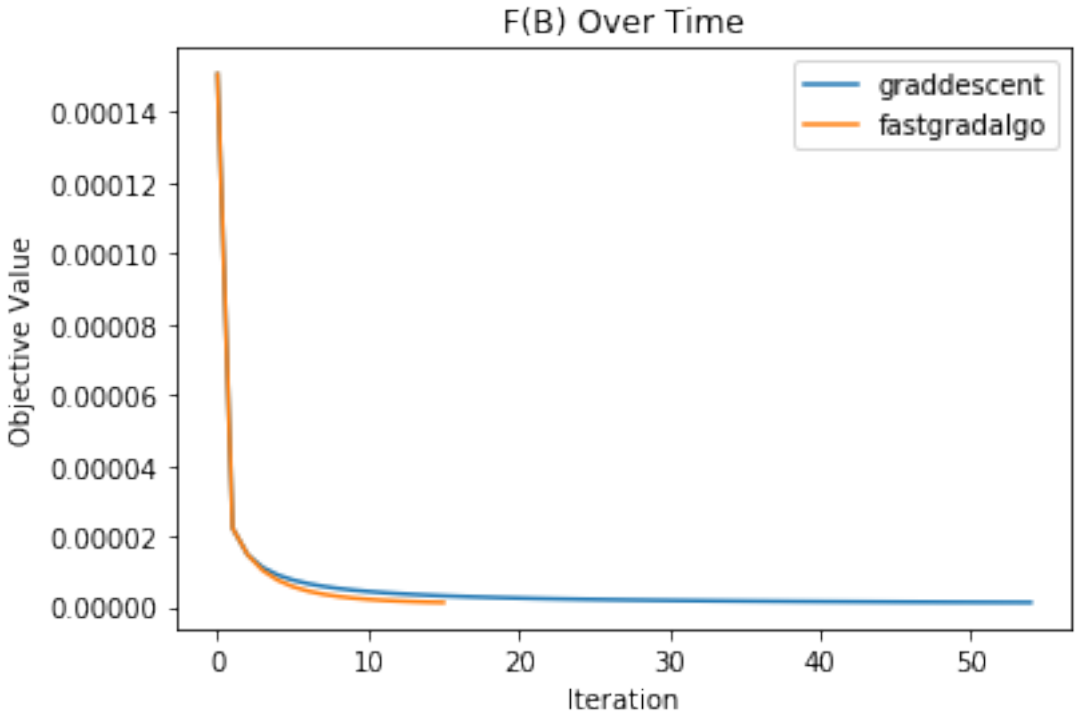
Then,

$$\nabla F(\beta) = \frac{-1}{n}\sum_{i=1}^{n} \frac{y_i x_i\, exp\left(-y_i x_i^T \beta\right)}{1 + exp\left(-y_i x_i^T \beta\right)} + 2\lambda\beta \tag{4}$$

Retrieving Spam Dataset *(see code in homework3.py)*

Function Definitions *(see code in homework3.py)*

Comparing Gradient Descent Algorithms



We observe a similar hyperbolic decline in objective values for both algorithms. In early iterations (t in $[0, 5]$), *fastgradalgo* begins to have objective values (very slightly) lower than *graddescent*. The steepness of the two curves is similar, but *fastgradalgo* approaches a marginally lower limit for $F(\beta)$. Also, *fastgradalgo* meets its stopping criterion between $t = 10$ and $t = 20$, notably earlier than *graddescent's* stopping after $t = 50$. So, we can summarize the lower objective threshold reached (and earlier termination) by arguing that *fastgradalgo* is more efficient than *graddescent*, while producing similarly-desirable loss.

Comparing Via Scikit-Learn

Note that in the scikit-learn package, the objective function is:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1) \tag{5}$$

Minimizing this function is equivalent to minimizing:

$$\min_{w,c} \frac{1}{2\,n\,C} w^T w + \frac{C}{n\,C} \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1) \tag{6}$$

We introduced the $\frac{1}{n\,C}$ factor to arrive at our objective function. This now becomes:

$$\min_{w,c} \frac{1}{2\,n\,C} w^T w + \frac{1}{n} \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1) \tag{7}$$

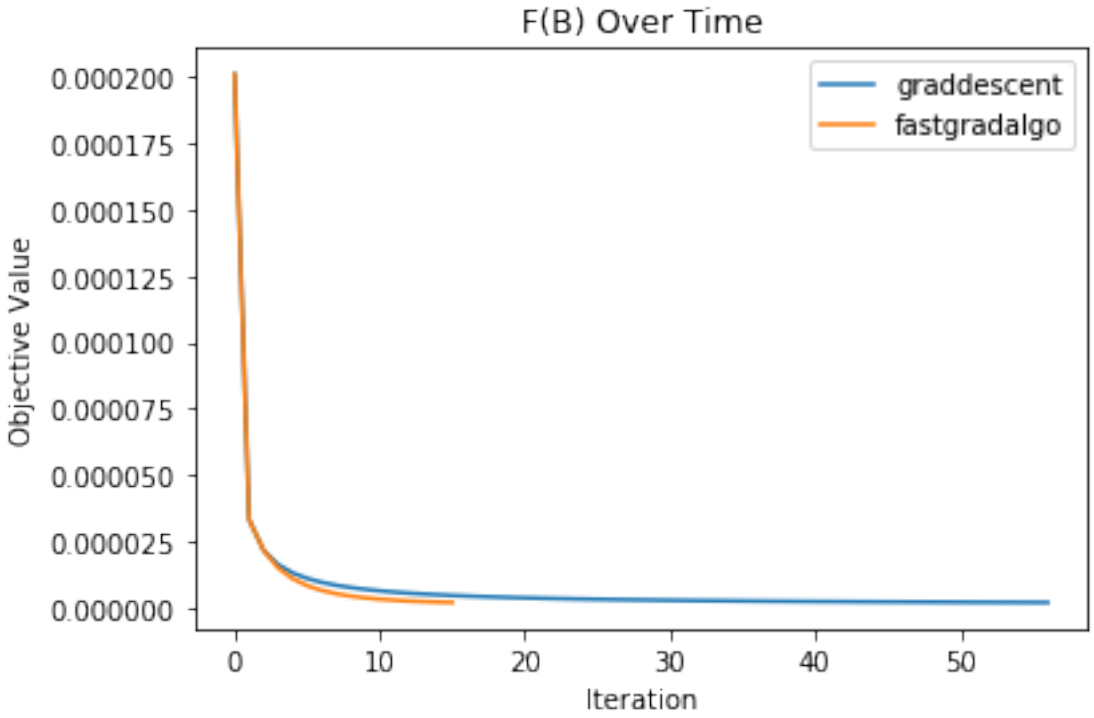
Which resembles our function. This means that $\frac{1}{2\,n\,C} = \lambda$, or that $C = \frac{1}{2\,n\,\lambda}$.

Please note: From the *sklearn* documentation, it could not be determined what the little c in the exponential term within the summation is referring to. It is separate from the big C mentioned here, and assumed to have negligible effect when equating the minimization of the *sklearn* objective function, and ours.

```
Our final iterate, B_T:

[ 8.04949087e-05 -1.92765312e-05  1.25638488e-04  3.65913439e-05
  1.54296314e-04  1.48354502e-04  2.11823744e-04  1.31901983e-04
  1.47682985e-04  8.86297249e-05  1.49582244e-04  4.93702493e-06
  8.47803683e-05  3.82852881e-05  1.24946149e-04  1.67877746e-04
  1.67870911e-04  1.30243494e-04  1.74534178e-04  1.21029236e-04
  2.44425672e-04  5.85881639e-05  2.13526424e-04  1.37835101e-04
 -1.63737302e-04 -1.48586269e-04 -1.16974664e-04 -1.01282444e-04
 -8.51607904e-05 -1.09123745e-04 -8.09442902e-05 -7.28454545e-05
 -7.64917315e-05 -7.19143671e-05 -9.51755079e-05 -8.68260912e-05
 -1.13556996e-04 -1.97942381e-05 -7.83411163e-05 -4.13300618e-05
 -6.21054500e-05 -8.71329783e-05 -8.65263748e-05 -6.03318683e-05
 -8.95522045e-05 -9.32066081e-05 -2.84958876e-05 -5.35876415e-05
 -3.80316681e-05 -5.71925891e-05 -4.12714700e-05  1.54275859e-04
  2.06409729e-04  4.14994139e-05  7.01572074e-05  1.37825988e-04
  1.58916321e-04]

Estimate from sklearn:

[ 0.02257694 -0.0116148   0.04902917  0.01954015  0.07279777  0.06278226
  0.10452786  0.0598945   0.05852104  0.03417032  0.05694155 -0.00842633
  0.03034325  0.01340622  0.04841639  0.08190959  0.07155303  0.05675207
  0.06828139  0.05361687  0.10377371  0.03404875  0.09569375  0.0619975
 -0.06362613 -0.05404018 -0.05268105 -0.02768621 -0.02605095 -0.03407375
 -0.01951476 -0.01262928 -0.03494865 -0.01225642 -0.02868438 -0.01902747
 -0.0441835  -0.0107055  -0.03079572 -0.00068256 -0.02513218 -0.03994499
 -0.03267805 -0.02790379 -0.04298378 -0.04429519 -0.01547072 -0.02493272
 -0.02046344 -0.01871143 -0.01557885  0.0733483   0.0945139   0.02122305
  0.02792329  0.05714016  0.06920232]

Objective value for our final iterate, B_T:

1.31E-06

Objective value for sklearn estimate:

6.89E-02
```

We observe that the two estimates themselves are dissimilar, and hence the objective values for the two estimates are dissimilar. In fact, the objective value for our final iterate β_T is notably smaller than that for *sklearn's* iterate $\beta_\star$. It is unclear why this may be; theorized reasons include dissimilar data-standardizing techniques, and computation error in *our* derivations or algorithms.

Cross-Validation & Misclassification Error



Firstly we observe that the plot of objective values matches our earlier plots of the same. A quick check confirms that this is indeed because the optimal λ from *sklearn* turned out to be:

```
0.5
```

So, *sklearn* arrived at the same λ that had been used throughout the exercise, hence the plots of objective values were identical.

Finally, note that at this point, a working misclassification-error curve for *our* algorithms was unable to be devised.

2 Exercise 2

(a) As we increase λ, the misclassification error on the training set will (iii), steadily increase. This is because the term penalizes the complexity, or variance, of a model. Here, it penalizes coefficients that fit to the noise in the training data (which leads to overfitting). Since we

exclusively see the training set, it will seem that the error is only rising, because larger terms will increasingly restrict the coefficients, and hinder the model's learning of the training data.

(b) For a large, previously unseen dataset (from the same distribution as the training data), the error will (ii), decrease initially and then increase in a U-shaped pattern. This is because the new set helps validate the model. If λ is increasing, then even though the training set may see worse performance, a larger regularization term lowers model variance. So, for this new data, we should see an inital lowering of the error, since λ is helping us generalize the model. However, after a certain point, increasing λ has too harsh and simplistic an effect on the coefficients. This reductive effect will increase the bias of the model, which will later result in a sharp uptick in the error.

Please note: while the document says 'parts (a) through (e)', the provided questions only contain parts (a) and (b).