CS-6350: Homework 5

James Brissette

December 6, 2018

1 Logisitc Regression

1 We know the derivative of $log(x)$ is $\frac{1}{x}$, and the derivative of e^x is e^x, so applying the chain rule to te formula we get the following (going from the outermost to inner most terms):

$$\begin{aligned}\nabla g(w) &= \frac{\partial}{\partial w} log(1 + exp(-y_i w^T x_i)) \\ &= \frac{1}{1 + exp(-y_i w^T x_i)} * exp(-y_i w^T x_i) * (-y_i x_i)\end{aligned}$$

2 Since we're taking the gradient using a single example, we can eliminate the summation and our equation becomes:

$$\begin{aligned}\nabla J^t &= \frac{\partial}{\partial w} log(1 + exp(-y_i w^T x_i)) + \frac{1}{\sigma^2} w^T w \\ &= \frac{-y_i x_i exp(-y_i w^T x_i)}{1 + exp(y_i w^T x_i)} + \frac{2w}{\sigma^2}\end{aligned}$$

3 The sudo code for this would be:

```
1. Initialize $w^0 = 0 \in R^n$
2. For epoch 1...T:
    (a) Pick a random example $(x_i, y_i)$ from the training set S
    (b) Treat $(x_i, y_i)$ as a full dataset and compute the gradient found above
    (c) Update $w^t \leftarrow \gamma_t \frac{-y_i x_i exp(-y_i w^{t-1} x_i)}{1+exp(y_i w^{t-1} x_i)} + \frac{w^{t-1}}{\sigma^2}$
3. Return w
```

2 Experiments

For each algorithm, I implemented my algorithm to convert the liblinear data into sparse vectors using dictionaries. this allowed me to store the dataset in an array of dictionaries which made indexing simple. I opted to use python as that has been my goto language this semester. When running my cross validations using the hyper parameter choices provided, I found my results weren't as high as when I attempted other choices. My final decisions for each hyperparameter were the results of trial and error at different combinations during my cross validation and may be different based on the initial random seed I chose to allow for repeatable results.

1. Nothing special to report in terms of implementation outside of the above. I refactored my existing perceptron code to create my SVM.

	Best hyper-params	Avg Cross-val P/R/F1	Test P/R/F1
SVM	rate=1; c=100	P=1.0; R=0.7796; F1=0.8998	P=1.0; R=0.7753; F1=0.8734
Logistic regression	rate=1; tradeoff=1000	P=0.2203; R=1.0; F1=0.8761	P=1.0; R=0.7753; F1=0.8791
Naive Bayes	lambda=0	P=0.1800; R=0.1743; F1=0.8837	P=0.9027; R=0.8750; F1=0.8886
SVM over trees			

Table 1: Results table

3 Naive Bayes and Linear Classifiers

1. Based on the notes for the naive Bayes classifier and a little bit of intuition, if we want to find a classifier equivalent to:

$$\frac{Pr(x|y=1)Pr(y=1)}{Pr(x|y=0)Pr(y=0)} \geq 1$$

we can start by taking the log and simplifying:

$$log\Big(\frac{Pr(x|y=1)Pr(y=1)}{Pr(x|y=0)Pr(y=0)}\Big) \geq log(1)$$

$$log\big(Pr(x|y=1)Pr(y=1)\big) - log\big(Pr(x|y=0)Pr(y=0)\big) \geq log(1)$$

$$log\big[Pr(x|y=1)\big] + log\big[Pr(y=1)\big] - log\big[Pr(x|y=0)\big] - log\big[Pr(y=0)\big] \geq log(1)$$

From here we can plug in our probabilities by looking at the Gaussian/Normal probability density function in the problem:

$$f(x_j|\mu_{j,y},\sigma) = \frac{1}{\sqrt{2\sigma^2\pi}}e^{-\frac{(x_j-\mu_{j,y})^2}{2\sigma^2}}$$

and we get for our first probability:

$$Pr(x|y=1) = \frac{1}{\sqrt{2\sigma^2\pi}}e^{-\frac{(x_j-\mu_1)^2}{2\sigma^2}}$$

and for our second probability:

$$Pr(x|y=0) = \frac{1}{\sqrt{2\sigma^2\pi}}e^{-\frac{(x_j-\mu_0)^2}{2\sigma^2}}$$

Noticing here we are without a value for σ we can choose any arbitrary value as we will see our choice here does not alter the outcome in terms of our classifier having a linear decision boundary. Plugging these probabilities back into our logrithmic inequality we find the rest boils down to some algebra and take advantage of like terms cancelling out. Any term that is not dependent on x is technically constant and can be separated out into it's own place. Here we will throw all of those terms into a constant J:

$$log\big[\frac{1}{\sqrt{2\sigma^2\pi}}e^{-\frac{(x_j-\mu_1)^2}{2\sigma^2}}\big]+log\big[Pr(y=1)\big]-log\big[\frac{1}{\sqrt{2\sigma^2\pi}}e^{-\frac{(x_j-\mu_0)^2}{2\sigma^2}}\big]-log\big[Pr(y=0)\big] \geq log(1)$$

Here we see that $Pr(y=1)$ is independent of our x variable and so will become part of J, as will $Pr(y=0)$

$$log\big[\frac{1}{\sqrt{2\sigma^2\pi}}e^{-\frac{(x_j-\mu_1)^2}{2\sigma^2}}\big] - log\big[\frac{1}{\sqrt{2\sigma^2\pi}}e^{-\frac{(x_j-\mu_0)^2}{2\sigma^2}}\big] + J \geq log(1)$$

$$log\big[\frac{1}{\sqrt{2\sigma^2\pi}}\big] + log\big[e^{-\frac{(x_j-\mu_1)^2}{2\sigma^2}}\big] - log\big[\frac{1}{\sqrt{2\sigma^2\pi}}\big] - log\big[e^{-\frac{(x_j-\mu_0)^2}{2\sigma^2}}\big] + J \geq log(1)$$

Equivalently, we will ignore our constants and include them in J:

$$log\big[e^{-\frac{(x_j-\mu_1)^2}{2\sigma^2}}\big] - log\big[e^{-\frac{(x_j-\mu_0)^2}{2\sigma^2}}\big] + J \geq log(1)$$

$$-\frac{(x_j-\mu_1)^2}{2\sigma^2} - \frac{-(x_j-\mu_0)^2}{2\sigma^2} + J \geq log(1)$$

Since $log(1) = 0$ we can get rid of our $2\sigma^2$ on the bottom by multiplying it on both sides

$$-(x_j-\mu_1)^2 + (x_j-\mu_0)^2 + J \geq 0$$

Simplifying we get

$$-(x_j^2 - 2x_j\mu_1 + \mu_1^2) + (x_j^2 - 2x_j\mu_0 + \mu_0^2) + J \geq 0$$

and we see the x_j^2 terms cancel, and the μ_1^2 and μ_0^2 terms are constant. Cleaning this up we get:

$$-x_j^2 + 2x_j\mu_1 - \mu_1^2 + x_j^2 - 2x_j\mu_0 + \mu_0^2 + J \geq 0$$

$$2x_j\mu_1 - \mu_1^2 - 2x_j\mu_0 + \mu_0^2 + J \geq 0$$

$$2x_j\mu_1 - 2x_j\mu_0 + J \geq 0$$

$$-2x_j(-\mu_1 + \mu_0) + J \geq 0$$

$$x_j(\mu_0 - \mu_1) + J \geq 0$$

And we see this is linear in x