CMPS 242: Homework #3

Due: *December 13, 2016*

S V N Vishwanathan

Kostas Zampetakis 1567380

Panos Karagiannis 1309484

Contents

Question 1

Consider the following table which details the use of contraception by age of currently married women in El Salvador in 1985.

Table 1: Contraceptive Method

Age	Ster.	Other	None
15-19	3	61	232
20-24	80	137	400
25-29	216	131	301
30-34	268	76	203
35-39	197	50	188
40-44	150	24	164
44-48	91	10	183

Let t_i denote an indicator variable which takes values $\{1, 2, 3\}$ where 1 is Sterilization, 2 is Other, 3 is None. Consider the activation function:

$$\alpha_{ij} = \alpha_j + \beta_j x_i + \gamma_j x_i^2$$

where x_i is the mid-point of the $i-th$ age group, and $j \in \{1, 2, 3\}$. Recall that the multinomial logit model is given by:

$$p(t_i = j \mid x_i) = \frac{exp(\alpha_{ij})}{\sum_{j'} exp(\alpha_{ij'})}$$

Implement multinomial logistic regression from scratch (for the underlying optimization you may use modules such as the ones available in scipy) and apply it to the given data. Comment on your observations.

Answer:

This question asks to implement a multiclass logistic classifier. In this case, our data points consist of the number of women, having as their only identification feature the age x, labeled with the method of contraception t they used. Although x is a real positive number, our data have been clustered in 7 different age groups. As the exercise suggests, we are going to use a quadratic transformation $\phi(x) = [1, x, x^2]$ of x, for our regression model such that the probability $\mathbb{P}(t = j \mid x = i)$ is given by

$$\mathbb{P}(t = j \mid x = i) = \frac{exp(a_{ij})}{\sum_{j'} exp(a_{ij'})}$$

where $i \in [7], j \in [3]$ and $a_{ij} = \alpha_j + \beta_j x + \gamma_j x^2$. Our goal is to learn the values of the coefficients $\alpha_j, \beta_j, \gamma_j$, based on our data. In fact in our implementation, instead of estimating all nine coefficients, we calculate only 6, namely:

$$\alpha_2 - \alpha_1, \alpha_3 - \alpha_1, \beta_2 - \beta_1, \beta_3 - \beta_1, \gamma_2 - \gamma_1, \gamma_3 - \gamma_1$$

as by using log-odds it suffices to estimate the above probabilities. To do this we use MLE approximation:

Question 1 continued on next page. . .

$$\max_{\alpha_j,\beta_j,\gamma_j} \prod_{n=1}^{N} \mathbb{P}(t = j \mid x_n)$$

or equivalently, using the above form of the probability and taking logs:

$$\max_{\alpha_j,\beta_j,\gamma_j} \sum_{n=1}^{N} a_{i_n j} - \sum_{n=1}^{N} \log \left(\sum_{j'} exp(a_{i_n j'}) \right)$$

where i_n denotes the age group of the i-th observation.
During our first attempts to calculate the coefficients we faced overflow issues in our code. Therefore, we used the so called "log sum exp trick" to avoid overflows due to big arguments of the exponentials. More precisely, we made use of the well-known formula

$$\log \sum_{n}^{N} e^{x_n} = c + \sum_{n}^{N} e^{x_n - c}$$

for $c = \max x_n$.
Hence our results are:

$$\begin{array}{lll} \alpha_2 - \alpha_1 = 7.895 & \beta_2 - \beta_1 = -0.421 & \gamma_2 - \gamma_1 = 0.006 \\ \alpha_3 - \alpha_1 = 12.370 & \beta_3 - \beta_1 = -0.649 & \gamma_3 - \gamma_1 = 0.008 \end{array}$$

Comparing our results with the relative frequencies we observe that our model is quite accurate. As an example, doing our calculations for the group of ages 15-19 we get:

$$\mathbb{P}(t = 1 \mid x) = 0.02,\ \mathbb{P}(t = 2 \mid x) = 0.23,\ \mathbb{P}(t = 3 \mid x) = 0.74$$

where the corresponding relative frequencies for the same group are:

$$f_1 = 0.01,\ f_2 = 0.2,\ f_3 = 0.78.$$

Question 2

Download the Enron-spam dataset from `http://www.aueb.gr/users/ion/data/enron-spam/` . Use the pre-processed datasets `Enron1...Enron5` as training data and `Enron6` as test data. Implement the Naive Bayes algorithm that we discussed in the class (a good in-depth reference is `http://nlp.stanford.edu/IR-book/pdf/13bayes.pdf`). Remember to perform all calculations on the log scale to prevent underflow.

- Report the accuracy on the test set
- How do you account for different prior probabilities for spam and ham?
- Does the performance of the classifier change when you do add one Laplace smoothing ?
- What are the most discriminative words as per the naive Baye's classifier?

Answer:

Question 2 continued on next page...

(a) In order to implement Naive Baye's classifier we considered many approaches to the problem. In all of the approaches considered we first filtered our data in order to remove words and punctuation marks that would not help in our classification. That way we managed to reduce our data set by approximately 20%.

First we created a dictionary of words comprising the union of words we found in our training spam and ham emails. The value corresponding to each key in the dictionary was the relative frequency of the word and was obtained by the formula $r(w) = \frac{N(w,c)}{N(c)}$ where $N(w,c)$ is the the number of times we encountered the word w in class c and in distinct emails, while $N(c)$ is the total number of words in the class. Therefore after training, our union dictionary was of size $N = 73532$ while the number of misclassified emails was $M = 1408$. The total number of emails in our testing data set was 1500 ham and 4500 spam giving a total number of $T = 7000$ emails for testing. As a result the misclassification error rate using this approach was:

$$E = \frac{M}{T} = \frac{1408}{7000} \approx 20\%$$

Next we followed another approach in implementing Naive Baye's classifier that yielded better results. In creating our dictionary of words while training, we considered only the common words in spam and ham. Similarly to the previous case, the value corresponding to each key in the dictionary was the relative frequency of the word and was obtained by the formula $r(w) = \frac{N(w,c)}{N(c)}$ where $N(w,c)$ is the the number of times we encountered the word w in class c, while N is the total number of words in the class. Nevertheless, in this case $N(w,c)$ included the multiplicity of word w in each email. This approach resulted in a total dictionary of size $N = 14325$ while the number of misclassified emails was $M = 150$. Our testing data remained unchanged and the misclassification error rate obtained was merely:

$$E = \frac{M}{T} = \frac{150}{7000} \approx 2.1\%$$

Finally, we considered applying Laplace smoothing in our first approach of the problem. Laplace smoothing yielded a far better result since now the union dictionary that resulted from training had no words that corresponded to 0 relative freqeucny, as it was previously the case with words not appearing in both the ham and spam datasets. The way we added Laplace smoothing was by using the formula:

$$r(w) = \frac{N(w,c) + 1}{N(c) + N}$$

Where $N(w,c)$ is the the number of times we encountered the word w in class c, $N(c)$ is the total number of words in the class c including multiplicities and N is the total number of distinct words in the vocabulary. Since we followed the same procedure in order to produce the dictionary our size did not change compared to case 1, nevertheless upon testing, the misclassified emails were only $M = 85$ resulting in a misclassification rate of:

$$E = \frac{M}{T} = \frac{85}{7000} \approx 1.2\%$$

Question 2 [(a)] continued on next page. . .

To conclude, in order to speed up our program we considered shortening the dictionary that resulted from training, by only keeping the most discriminative words. The resulting dictionary had size $N = 3633$ and allowed the program to run approximately 20 times faster that the previous dictionary. Nevertheless, the number of misclassified emails increased to $M = 103$ resulting in a misclassification rate of:

$$E = \frac{M}{T} = \frac{103}{7000} \approx 1.4\%$$

(b) The way we account for the different probabilities of spam and ham is by taking the relative frequency of the spam and ham emails in our training data set. More precisely after training on Enron1 through Enron5 we got that:

$$f_{ham} = \frac{3672 + 4361 + 4012 + 1500 + 1500}{27716} = \frac{15045}{27716} \approx 54\%$$

$$f_{spam} = \frac{1500 + 1496 + 1500 + 4500 + 3675}{27716} = \frac{12671}{27716} \approx 46\%$$

(c) Some of the most discriminative words according to our classifier in the ham class are:
`'yesterday','2001','meeting','interview','utilities','weather','wednesday','edu','analys`
Some of the most discriminative words according to our classifier in the spam class are:
`'cheap','men','body','weight','!','prize','winning','instant','proven'`

Question 3

Work on the same dataset as the above problem.

- Implement the binary logistic regression algorithm that we discussed in the class.
  - Report accuracy on test set.
  - How do you account for different prior probabilities for spam and ham?
  - What are the most discriminative words as per the logistic regression classifier?
- Implement the Bayesian logistic regression algorithm that we dis- cussed in the class.
  - Report accuracy on test set.
  - How does the accuracy change when the strength of the prior changes?

You may use any optimizer of your choice including optimization modules from scipy.

Answer:

Question 3 continued on next page. . .

(a) In order to implement logistic regression we need to assume that:

$$P(t \mid \mathbf{w}^\top \phi(\mathbf{x})) = \sigma(\mathbf{w}^\top \phi(\mathbf{x}))$$

Now assuming i.i.d. on the observed values T $= \{t_1 \ldots t_N\}$ and corresponding values X $= \{\mathbf{x}_1 \ldots \mathbf{x}_N\}$ we get that the likelihood is:

$$\mathcal{L}(\mathbf{w}) = \prod_{n=0}^{N} \sigma(\mathbf{w}^\top \phi(\mathbf{x}_n))^{t_n} (1 - \sigma(\mathbf{w}^\top \phi(\mathbf{x}_n)))^{1-t_n}$$

with the variable t_n being 1 if $C = 1$ and 0 otherwise.
Carrying out simple calculations, we get that the log likelihood which we need to maximize is simply:

$$\log(\mathcal{L}(\mathbf{w})) = \prod_{n=0}^{N} t_n \log(\sigma(\mathbf{w}^\top \phi(\mathbf{x}_n)))(1 - t_n) \log((1 - \sigma(\mathbf{w}^\top \phi(\mathbf{x}_n))))$$

In this expression we need to maximize $\log(\mathcal{L}(\mathbf{w}))$, nevertheless in our approach we minimize $-\log(\mathcal{L}(\mathbf{w}))$.Here we need to mention that for our implementation of logistic regression we considered a population consisting only of the most discriminative words thus reducing significantly the size of our dictionary down to $D = 3633$ words. Moreover we considered each email $\mathbf{x}$ to be a binary vector thus each entry x_i signifying whether word i was included in mail $\mathbf{x}$. Moreover, it is worthwhile to mention that in order to avoid overflow errors we implemented a more stable version of the simgoid function:
For x negative:

$$\sigma(x) = \frac{e^x}{1 + e^x}$$

Else for x positive:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Finally using the $BFGS$ solver in python and supplying the gradient, we get that the number of misclassified emails is $M = 572$ resulting in a misclassification error rate:

$$E = \frac{M}{T} = \frac{572}{7000} \approx 7\%$$

The most discriminative words for ham according to our classifier are:
`'2001','mike',tuesday','yesterday','load','edu','url','best','controlled'`
Discriminative words for spam according to our classifier are:
`'!','wish',attack','immediatly','prize','fortune','cheap','commission','body'`

As far as the prior probabilities are concerned we can see that the ratio of the priors determines the x-independent coordinate of $\phi(x)$.

(b) In implementing Bayesian logistic regression we considered the prior distribution over $\mathbf{w}$ to be:

$$p(\mathbf{w}) = N(\mathbf{m}_0, \mathbf{S}_0) = N(0, \alpha\mathbf{I})$$

Then the posterior is given by:

$$p(\mathbf{w} \mid t) \propto p(t \mid \mathbf{w})p(\mathbf{w})$$

This is an intractable form hence we make a Gaussian approximation $p(\mathbf{w}|t) = N(\mathbf{w}_{MAP}, \mathbf{S}_N)$. To find $\mathbf{w}_{MAP}$ we simply need to maximize the posterior in terms of $\mathbf{w}$:

$$log(p(\mathbf{w} \mid t)) \propto \prod_{n=0}^{N} t_n \log(\sigma(\mathbf{w}^\top \mathbf{x}_n))(1-t_n)\log((1-\sigma(\mathbf{w}^\top \mathbf{x}_n))) - \frac{1}{2}(\mathbf{w}-0)^\top \frac{1}{\alpha}\mathbf{I}(\mathbf{w}-0) =$$

$$\prod_{n=0}^{N} t_n \log(\sigma(\mathbf{w}^\top \mathbf{x}_n))(1-t_n)\log((1-\sigma(\mathbf{w}^\top \mathbf{x}_n))) - \frac{\alpha}{2}\mathbf{w}^\top \mathbf{w}$$

Notice that this form is identical to finding $\mathbf{w}_{MLE}$ in the context of Logistic regression when we consider a regularizer. Therefore, it is logical to expect that our estimates for $\mathbf{w}_{MAP}$ are less likely to overfit our data yielding a smaller misclassification rate. In order to find $\mathbf{w}_{MAP}$ we used Python's *BFGS* solver and we supplied the gradient. In order to calculate $\mathbf{S}_N$ we used the following formula:

$$\mathbf{S}_N^{-1} = \frac{1}{\alpha}\mathbf{I} + \sum_{n=0}^{N}(\sigma(\mathbf{w}_{MAP}^\top \mathbf{x}_n))(1-\sigma(\mathbf{w}_{MAP}^\top \mathbf{x}_n)\mathbf{x}_n\mathbf{x}_n^\top$$

After training, and obtaining values for $\mathbf{w}_{MAP}$ and $\mathbf{S}_N$ it is time to predict. Let $\mathbf{x}_i$ be the email which we are trying to predict then also let:

$$\mu_a = \mathbf{w}_{MAP}^\top \mathbf{x}_i$$
$$\sigma_a = \mathbf{x}_i^\top \mathbf{S}_N \mathbf{x}_i$$
$$\Phi(x) = \int_{-\infty}^{x} N(0,1)$$

Then a good approximation to $P(t \mid \mathbf{x})$, which we do not prove for this exercise, is given by:

$$P(t \mid \mathbf{x}) = \sigma\left(\left(1 + \pi\frac{\sigma_a^2}{8}\right)^{-1/2} \mu_a\right) = \frac{1}{1 + exp\left\{\left(1+\pi\frac{\sigma_a^2}{8}\right)^{-1/2} \mu_a\right\}}$$

After testing on 7000 mails contained in Enron6 we counted $M = 234$ misclassifications thus resulting in an error rate of:

$$E = \frac{M}{T} = \frac{234}{7000} \approx 3.3\%$$

Changing the strength of the prior by shifting $\mathbf{m}_0$ away from 0, we notice that the rate of misclassification increases. Also, we noticed that as we decrease the hyperparameter α our prediction relies more on our prior, as expected.
In conclusion, the results we get for each of the different methods of classifying emails are highly dependent on the quality of the optimization obtained and on the extent that overflows and underflows are minimized. From our attempts, we consider Naive Bayes as the most successful method. This is because in general Naive Bayes has low complexity and gives accurate results. Even though, it can be shown that Naive Bayes will perform worse than Regression in many cases, from our experience, optimizing non-linear equations has many difficulties ranging from overflows to lack of memory and large computation times. Even if the loss function is convex and a unique minimum exists there are still many factors one has to consider in implementing such an algorithm. These are some of the reasons that prevented us from fully exploiting the potential of Regression methods. Let us also note that in a future implementation we consider fully vectorizing our code in order to achieve smaller complexity.