| | |
|---|---|
| **Coursework** | **Fall 2016** |
| **Statistical Pattern Recognition (COMP136)** | **Roni Khardon** |
| **Department of Computer Science, Tufts University** | **Rishit Sheth** |

# Programming Project 3

This programming project is due by Wednesday, November 16 in hardcopy (in class) and electronically (via provide).

## Experiments with Classification Algorithms

In this programming project, we will be training linear classification models and assessing their performance on artificial and real datasets. Your goals in this assignment are to (1) compare the performance of the generative and discriminative models, and (2) compare Newton's method to gradient ascent.

## Task 1: Generative vs. Discriminative

In this portion, you will implement and evaluate two algorithms for the binary classification problem. In particular, we will look at the following algorithms discussed in class:

- 2 class generative model with shared covariance matrix as developed in Section 4.2 of the textbook.
- Bayesian logistic regression where we use the simple prior $w \sim \mathcal{N}(0, \frac{1}{\alpha}I)$. In this case the update formula for $w$ is: $w_{n+1} \leftarrow w_n - (\alpha I + \Phi^T R \Phi)^{-1}[\Phi^T(y - t) + \alpha w_n]$ where $R$ is as in Eq (4.98) in the textbook. The predictive distribution is described in Section 4.5.2 of the textbook.

  Note that unlike the other algorithms, logistic regression relies on a free parameter ($w_0$) to capture an appropriate separating hyperplane. Therefore, you will need to add a feature fixed at one to the data for this algorithm. For the implementation, initialize $w = 0$ and set $\alpha = 0.1$ (unlike in previous projects, we won't be performing model selection here). Use the following test as a stopping criterion: $\frac{\|w_{n+1} - w_n\|_2}{\|w_n\|_2} < 10^{-3}$ or $n \geq 100$.

We will use the following datasets, available through the course web page, to evaluate the algorithms. Each dataset is given in two files with the data in one and the labels in the other file.

- The first dataset (marked as A) has uniformly distributed data with an arbitrary weight vector separating positive from negative examples. Therefore data does not conform to the Gaussian generative model but the data is linearly separable.

- In the second dataset (marked as B) each class is generated from multiple Gaussians with differing covariance structure. This diverges even further from the generative model yet we designed it to be somewhat linearly separable.
- The examples of the third dataset, USPS[1], represent $16 \times 16$ bitmaps of the characters 3 and 5. We use this dataset for the binary classification problem of distinguishing between these two characters.

Your task is to implement the 2 algorithms, and evaluate them as follows. For each combination of dataset and algorithm: Step 1) Set aside 1/3 of the total data (randomly selected) to use as a test set. Step 2) Record performance as a function of increasing training set size (with each training set randomly selected from the other 2/3 of the total data). Repeat Steps 1 & 2 a total of 30 times to generate learning curves with error bars (i.e., $\pm 1\sigma$).

In your submission plot these results and discuss them. What can you observe on the performance of the algorithms? and what can you conclude from these observations?

## Task 2: Newton's method vs. gradient ascent

We have introduced Newton's method in order to improve convergence of gradient based optimization. However, the updates in Newton's method can be expensive when the computation of the inverse Hessian is demanding. In general gradient ascent can perform many updates for the same time cost of one update of Newton's method. In this task we compare the two methods for the quality of the weight vector they produce as a function of time.

Recall that the update of gradient ascent is given by $w_{n+1} \leftarrow w_n - \eta[\Phi^T(y-t) + \alpha w_n]$. Gradient ascent is sensitive to the choice of learning rate $\eta$. For this part you should use $\eta = 10^{-3}$ (which we have verified to work reasonably well).

Our goal is to compare the two methods for the quality of the weight vector they produce as a function of time. To achieve this we need intermediate values of $w$ as well as time stamps for when that value is produced. In your implementation you should store the value of $w$ as well as the wall clock time after each update (or every few updates for gradient ascent). Once training is done you can evaluate the test set error of each of the $w$ vectors produced. Finally you can plot the performance of the algorithm as a function of run time.

Please perform this evaluation on datasets A and USPS. To make sure we have stable results across submissions - please use the first 1/3 of the dataset for testing and the rest for training the algorithms. Since estimates of run time from your computing environment can be noisy please repeat the above 3 times (on the same 1/3, 2/3 partition of the data) and average the times across these runs. Note that the $w$ vectors will be identical because the data is the same and their evaluation does not need to be repeated.

The above descsription does not require a stopping criterion. However, to control run time in your experiments run them with the same stopping criterion for $w$, that is, $\frac{\|w_{n+1}-w_n\|_2}{\|w_n\|_2} < 10^{-3}$ and with a bound on the number of iteration stopping if $n \geq 100$ iterations for Newton's method and $n \geq 6000$ for gradient ascent.

[1] http://www.gaussianprocess.org/gpml/data/

In your submission plot these results and discuss them. What can you observe on the performance of the algorithms? and what can you conclude from these observations?

## Extra Credit

In task 2 we fixed $\eta$ to a value that was chosen carefully (and with hindsight of the result). In general one would need to choose the learning rate carefully, or better change it dynamically in every step so as to guarantee fast convergence. For this extra credit portion you should explore this idea. First run the algorithm with multiple fixed $\eta$ values to see the sensitivity. Then do some independent study to learn about the notion of *line search*, implement such a scheme and evaluate it along the fixed $\eta$ values. Write a report explaining your method and your findings.

## Additional Notes

- Equation 4.143 of Bishop (pg. 218) is incorrect as documented in the textbook errata. The LHS should be $S_N^{-1}$, **not** $S_N$.
- To help you test your BLR implementation, we provide an additional dataset, *irlstest*, and solution weight vector in *irlsw*. The first entry in *irlsw* corresponds to $w_0$.
- If you have problems with singular matrices, you can simply "regularize" them by adding a small constant to the diagonal, e.g., $+10^{-9}I$.

## Submission

- You should **submit** the following items **both electronically and in hardcopy**:
  (1) All your source code for the assignment. Please write clear code with documentation as needed. The source code should (i) run on *homework.eecs.tufts.edu*, (ii) run from the command line *without editing* with a single command (if there is more than one execution command required, include those commands in a Bash script which we can run), and (iii) output the requested results.
  You can assume the data files will be available in the same directory as where the code is executed. Please use filenames as provided for the data. Please include a short README file with the code execution command.
  (2) A PDF report on the experiments, their results, and your conclusions as requested above.
- For electronic submission, put all the files into a zip or tar archive, for example `myfile.zip` (you do not need to submit the data we give you). Please do not use another compression format such as RAR. Then submit using `provide comp136 pp3 myfile.zip`.
- Your assignment will be graded based on the clarity and correctness of the code, and presentation and discussion of the results.