Project 3: Optimal Scoring and Lasso

Isaiah Chen

April 15, 2019

Question 1

Given a training set (x_1, y_1),...,(x_N, y_N) with $x_k \in \mathbb{R}^d$ and $y_k \in \mathcal{G} = \{g_1, ..., g_q\}$, optimal scoring is formulated in terms of the minimization of

$$F_0(\theta, b) = -2\text{trace}(b^T M^T C\theta) + \text{trace}(b^T \Sigma_{XX} b)$$

with respect to θ and b with the constraints $\theta^T C\theta = \text{Id}_{\mathbb{R}^r}$ and $\theta^T C\mathbf{1}_d = 0$. Here, b is a $d \times r$ matrix;

$$\theta = \begin{pmatrix} \theta_{g_1}^T \\ \vdots \\ \theta_{g_q}^T \end{pmatrix}$$

is a $q \times r$ matrix (with r chosen between 1 and $q-1$);

$$M = \begin{pmatrix} (\mu_{g_1} - \mu)^T \\ \vdots \\ (\mu_{g_q} - \mu)^T \end{pmatrix}$$

is a $q \times d$ matrix, where μ_g is the average over x_k such that $y_k = g$ and μ is the global average over all x_k's; C is the diagonal matrix with diagonal coefficients N_{g_1}/N,...,N_{g_q}/N, N_g being the size of class g and $\mathbf{1}_d$ is the d-dimensional vector with all coefficients equal to one; finally, $\Sigma_{XX} = \mathcal{X}_c^T \mathcal{X}_c / N$, where $\mathcal{X}_c$ is the $N \times d$ matrix with kth row given by $(x_k - \mu)^T$.

We want to minimize F_0 with respect to θ for fixed b. We choose to define $\tilde{\theta} = U^T C^{1/2} \theta V$, where UDV^T is the SVD of $C^{1/2}Mb$. $C^{1/2}Mb$ will be a $q \times r$ matrix. Consquently, U, D, and V^T will have dimensions $q \times q, q \times r$, and $r \times r$, respectively. In order for there to be an optimal solution with the variables we have defined, $\text{rank}(Mb) \geq r$. If $\text{rank}(Mb) < r$, the number of independent equations that are being minimized in the optimization problem

will be too low for there to be a feasible solution. This also explains why r must be chosen appropriately between 1 and $q-1$.

First, we can solve for θ:

$$\tilde{\theta} = U^T C^{1/2} \theta V$$
$$\tilde{\theta} V^{-1} = U^T C^{1/2} \theta$$
$$\theta = C^{-1/2} U \tilde{\theta} V^{-1}$$

We can plug this expression for θ into the original function F_0 and assume that b is fixed. F_0 can now be written in terms of $\tilde{\theta}$:

$$F_0(\tilde{\theta}) = -2\text{trace}(b^T M^T C^{1/2} U \tilde{\theta} V^{-1}) + \text{trace}(b^T \Sigma_{XX} b)$$

The two constraints can now be written as $V\tilde{\theta}^T\tilde{\theta}V^{-1} - \text{Id}_{\mathbb{R}^r} = 0$ and $V\tilde{\theta}^T U^T C^{1/2}\mathbf{1}_d = 0$. We can use the method of Lagrange multipliers to minimize F_0. We choose the Lagrange multipliers α and β to satisfy

$$\frac{\partial F_0}{\partial \tilde{\theta}} + \alpha \frac{\partial G_1}{\partial \tilde{\theta}} + \beta \frac{\partial G_2}{\partial \tilde{\theta}} = 0$$

where G_1 and G_2 correspond to the two constraints, respectively. After taking derivatives of both the function to be minimized and the two constraints, we can plug them into the expression above:

$$-2U^T C^{1/2} M^T b^T V + \alpha(V\tilde{\theta}^T V^T + V\tilde{\theta} V^T) + \beta V \tilde{\theta}^T U^{-1} C^{1/2} \mathbf{1}_d = 0$$

This equation, along with the two constraints, are a system of equations that can be solved for $\tilde{\theta}, \alpha$, and β. While the solutions of α and β are not necessarily relevant for finding $\tilde{\theta}$ for this specific case, we can find that

$$\tilde{\theta} = \begin{pmatrix} \text{Id}_{\mathbb{R}^r} \\ 0 \end{pmatrix} = \begin{pmatrix} V^T \\ 0 \end{pmatrix} V$$

We can multiply both sides of this expression by the desired variables $C^{-1/2}$ and V^{-1} to return to our original definition of θ:

$$C^{-1/2}\tilde{\theta}V^{-1} = C^{-1/2} U \begin{pmatrix} V^T \\ 0 \end{pmatrix}$$

The left side of this equation is equal to original defintion of θ. Therefore, if $\text{rank}(Mb) \geq r$, the optimal θ is given by

$$\theta = C^{-1/2} U \begin{pmatrix} V^T \\ 0 \end{pmatrix}$$

Question 2

We now consider a penalized version of the previous problem minimizing

$$F_\lambda(\theta, b) = -2\text{trace}(b^T M^T C\theta) + \text{trace}(b^T \Sigma_{XX} b) + \lambda \sum_{i=1}^{d} \sum_{j=1}^{r} |b(i,j)|$$

with respect to θ and b with the constraints $\theta^T C\theta = \text{Id}_{\mathbb{R}^r}$ and $\theta^T C\mathbf{1} = 0$. We now want to assume that θ is fixed and minimize with respect to b.

(1) The optimal b can be obtained using the following version of the alternating direction method of multipliers (ADMM) algorithm, where we iterate

$$\begin{cases} b \leftarrow (\Sigma_{XX} + \frac{\text{Id}}{2\rho})^{-1}(M^T C\theta + \frac{\gamma - \tau}{2\rho}) \\ \gamma \leftarrow S_{\lambda p}(b + \tau) \\ \tau \leftarrow \tau + b - \gamma \end{cases}$$

τ and γ are both $d \times r$ matrices, ρ is a small positive number, and $S_{\lambda\rho}$ is the shrinking operator, which applies the function $t \mapsto (|t| - \lambda\rho)^+$ to every element of a matrix.

In order to prove that this algorithm can be used to find the optimal b, we need to make some assumptions. We assume that all functions involved are closed, proper, and convex as to ensure that it is always possible to update the relevant parameters in the algorithm. We also assume that the associated Lagrangian has a saddle point representing the minimum of the function. Therefore, we must have residual convergence of the primal problem, objective convergence to the optimal b, and convergence of the dual problem to 0.

Also, we can rewrite the minimization of b as r independent problems (each one corresponding to each column of b) that are similar to the lasso problem. The ADMM algorithm can be applied to each independent problem and can be proven to work provided that ρ is small enough.

If we simplify each term in F_λ, we can see that $b^T M^T C\theta$ and $b^T \Sigma_{XX} b$ will both be $r \times r$ matrices. Taking the trace of these expressions will result in scalar values that can be easily added to each other to calculate the value of F_λ. Looking at the individual iterations in the algorithm, we can simplify both terms in the expression for b and we observe that we are essentially multiplying a $d \times d$ matrix by a $d \times r$ matrix. This is representative of solving a system of linear equations with r independent equations. If the algorithm can be applied to each independent equation, then it must also follow that the original problem can be minimized using the same ADMM algorithm.

(2) We can describe a minimization algorithm for F_λ that is initialized with

$$\theta = C^{-1/2} \begin{pmatrix} \text{Id}_{\mathbb{R}^r} \\ 0 \end{pmatrix}$$

and alternates a minimization step with fixed θ and a minimization step with fixed b until convergence is reached. For all odd-numbered steps of the algorithm, we will update b by minimizing

$$-2\text{trace}(b^T M^T C\theta) + \text{trace}(b^T \Sigma_{XX} b) + \lambda \sum_{i=1}^{d} \sum_{j=1}^{r} |b(i,j)|$$

with fixed θ. This minimization is performed using the *Broyden-Fletcher-Goldfarb-Shanno (BFGS)* algorithm, which is suitable because there are no constraints on b that need to be considered. Once b is updated, we will also check that $\text{rank}(Mb) \geq r$ using the updated value of b. For all even-numbered steps of the algorithm, we will update θ by minimizing

$$-2\text{trace}(b^T M^T C\theta) + \text{trace}(b^T \Sigma_{XX} b) + \lambda \sum_{i=1}^{d} \sum_{j=1}^{r} |b(i,j)|$$

with fixed b and subject to the constraints $\theta^T C\theta = \text{Id}_{\mathbb{R}^r}$ and $\theta^T C\mathbf{1} = 0$. This minimization is performed using the *Sequential Least Squares Programming (SLSQP)* algorithm, which behaves similarly to any sequential quadratic programming method that is used to optimize a nonlinear function that is subject to constraints.

For every step of the algorithm, once we have updated either b or θ, we will then evaluate F_λ using the updated values and compare it to the evaluation of this function from the previous step. If the absolute value of the difference between the two evaluations is within a desired tolerance, then we have converged at the minimum of F_λ and we have determined the optimal values for b and θ.

Question 3

(1) A training function was used to implement the algorithm described above in the previous question. The function returns optimal values of b and θ such that F_λ is minimized. These parameters were then used to predict the classes for both the training data and the testing data provided. After classification, the error was calculated as the fraction of incorrectly classified samples in the dataset. For $\lambda = 5/\sqrt{N}$, the training and testing errors were calculated to be 0.230 and 0.202, respectively. From observing the optimal value of b, there is only one non-zero coefficient at the (3, 1) index.

(2) The algorithm was also run on larger training and testing datasets for various values of λ. The total number of iterations required (i_{total}), the number of non-zero coefficients in each column of b (bnz_1, bnz_2), and the rates of correct classification on both the training set (RCC_{train}) and the testing set (RCC_{test}) for each case are all shown below in Table 1. A plot of the cost function after each minimization step for specific values of lambda is shown below in Figure 1. The total number of iterations that is listed contains all the iterations for the BFGS and SLSQP algorithms within the alternating minimization steps.

$\sqrt{N}\lambda$	i_{total}	bnz_1	bnz_2	RCC_{train}	RCC_{test}
0.1	32	29	27	0.847	0.791
0.5	165	19	21	0.848	0.801
1	147	12	16	0.845	0.807
2.5	188	3	4	0.832	0.813
5	222	3	3	0.843	0.825
7.5	284	2	3	0.852	0.835
10	226	2	2	0.853	0.834
20	319	2	1	0.848	0.834
30	238	0	0	0.790	0.770

Table 1: Results from the minimization algorithm for various values of λ

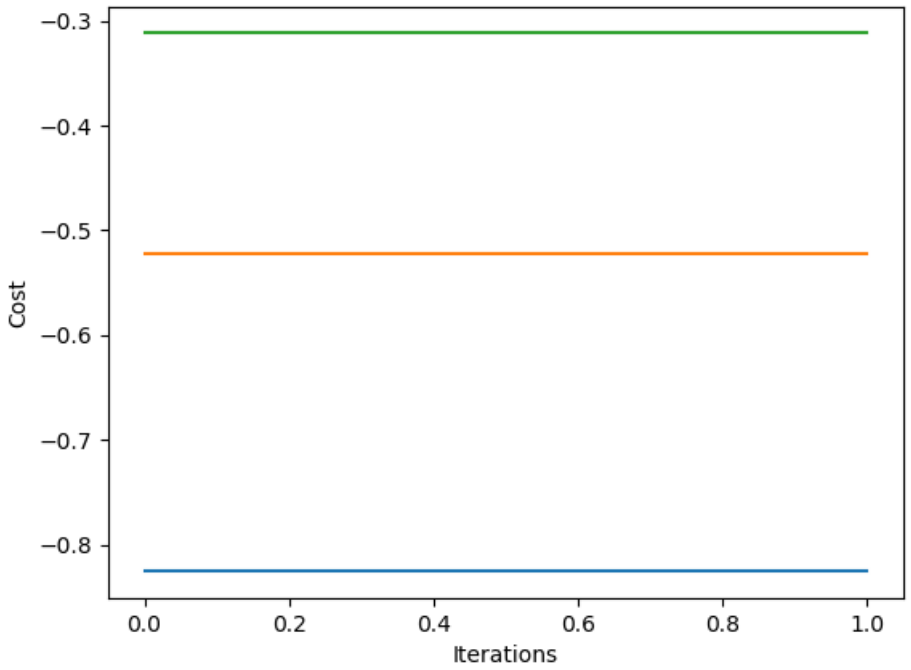

Figure 1: Cost function after each minimization step for different values of λ. The blue, orange, and green lines represent $\sqrt{N}\lambda = 0.1$, 5, and 10, respectively.

It is important to note that the program may have errors. The cost function does not appear to have been minimizing and as a result, the overall algorithm for most of the cases only ran once; this is observed in Figure 1. However, the optimal values for b still appear to be unique and can effectively classify most of the test data. The program that was used to generate the parameters is attached with this assignment.