Homework 3

For the third homework, the maximum number of uploads is limited to 5. A detailed explanation of this step is on its own page [/b221/courses/b0b36jul/en/hw/start].

SVM classifier training

One of the most popular classifiers is SVM [https://en.wikipedia.org/wiki/Support-vector_machine] (support vector machine). Let us have n observations $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$. For simplicity, let's assume that this is a binary classification with labels $y_1, \dots, y_n \in \{-1, 1\}$, and that the bias is already embedded in the data. SVM searches for a linear separating hyperplane $\mathbf{w}^\top \mathbf{x}$ by introducing an auxiliary variable $\xi_i \geq 0$ for each observation i that measures the misclassification rate. It achieves this using a condition

$$\xi_i \geq 1 - y_i \mathbf{w}^\top \mathbf{x}_i.$$

Since we will want to minimize ξ_i, we will reach its minimum value for $y_i \mathbf{w}^\top \mathbf{x}_i \geq 1$. Thus, for the label $y_i = 1$ we want the condition $\mathbf{w}^\top \mathbf{x}_i \geq 1$ and similarly for the negative label. This is a stronger condition than the classical $\mathbf{w}^\top \mathbf{x}_i \geq 0$. In other words, SVM tries to achieve a certain distance of well-classified observations from the separating hyperplane, which adds robustness.

The entire optimization problem can then be written as

$$\begin{aligned} \underset{w,\xi}{\text{minimize}} \qquad & C \sum_{i=1}^n \xi_i + \frac{1}{2}||\mathbf{w}||^2 \\ \text{under conditions} \qquad & \xi_i \geq 1 - y_i \mathbf{w}^\top \mathbf{x}_i, \\ & \xi_i \geq 0. \end{aligned}$$

The regularization $\frac{1}{2}||\mathbf{w}||^2$ and the regularization constant $C > 0$ appeared in the objective function. Although this problem can be solved in this formulation, it is converted to a dual formulation by default

$$\begin{aligned} \underset{z}{\text{maximize}} \qquad & -\frac{1}{2}\mathbf{z}^\top Q \mathbf{z} + \sum_{i=1}^n z_i \\ \text{under conditions} \qquad & 0 \leq z_i \leq C, \end{aligned}$$

where the matrix Q has elements $q_{ij} = y_i y_j \mathbf{x}_i^\top \mathbf{x}_j$. There is a connection between the primal and dual problems. First, their optimal values coincide. Second, when we solve the dual problem, the solution of the primal problem is obtained by $\mathbf{w} = \sum_{i=1}^n y_i z_i \mathbf{x}_i$.

This problem can be solved using the method coordinate descent [https://en.wikipedia.org/wiki/Coordinate_descent]. This method updates only one component of the vector $\mathbf{z}$ in each iteration. Therefore, in each iteration, we fix some i and we replace the optimization over $\mathbf{z}$ by the optimization over $\mathbf{z} + d\mathbf{e}_i$, where $\mathbf{e}_i$ is the zero vector with one on component i. In each iteration we then solve

$$\begin{aligned} \underset{d}{\text{maximize}} \qquad & -\frac{1}{2}(\mathbf{z}+d\mathbf{e}_i)^\top Q(\mathbf{z}+d\mathbf{e}_i) + \sum_{i=1}^n z_i + d \\ \text{under conditions} \qquad & 0 \le z_i + d \le C. \end{aligned}$$

This optimization problem is simple since $d \in \mathbb{R}$ and there is a closed-form solution. This procedure is repeated for a given number of epochs, where one epoch performs the above-mentioned iterations for $i = 1$, $i = 2$ up to $i = n$.

Input

Implement the functions `Q = computeQ(X, y)` , `w = computeW(X, y, z)` , `z = solve_SVM_dual(Q, C; max_epoch)` and `w = solve_SVM (X, y, C; kwargs…)` . This diagram shows both input and output arguments. In more detail, these arguments are:

- `X` : matrix of $n \times d$ input data;
- `y` : vector of n input labels;
- `C` : positive regularization constant;
- `w` : solving the primal problem $\mathbf{w}$;
- `z` : solving the dual problem $\mathbf{z}$;
- `Q` : matrix of the dual problem Q.

The `computeQ` function computes the `Q` matrix and the `computeW` function computes the `w` from the dual solution. The `solve_SVM_dual` function takes as inputs the parameters of the dual problem and does `max_epoch` epochs (therefore `n*max_epoch` iterations) of the coordinate descent method. The solution should be initialized as $z = 0$. The `solve_SVM` function combines the previous functions into one.

Finally, write the function `w = iris(C; kwargs…)` which will load the iris dataset, as positive class it will consider `versicolor` , as negative class `virginica` , as features it will use `PetalLength` and `PetalWidth` , normalizes these inputs, adds a bias (as the last column of X), and finally uses the functions written above to calculate the optimal separating hyperplane `w` .

Recommended testing

Writing the following tests is not necessary. However, it can help you to debug your code. For example, you can:

- Verify the correctness of the dual solution by trying a large number of different values of d.

- Verify the equality of the optimal values of the primal and dual problem.
- Verify correct propagation of `kwargs` by replacing them with different values of `max_epoch` .
- Anything else.

courses/b0b36jul/en/hw/hw3.txt · Last modified: 2022/12/12 12:31 by adamluk3