Hw3

Friday, March 5, 2021 8:11 PM

1 The ℓ_2 Support Vector Machine [20pts]

In class, we discussed that if our data is not linearly separable, then we need to modify our optimization problem to include slack variables. The formulation that was used is known as the ℓ_1-norm soft margin SVM. Now consider the formulation of the ℓ_2-norm soft margin SVM, which squares the slack variables within the sum. Notice that non-negativity of the slack variables has been removed.

$$\arg\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2$$
$$\text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \qquad \forall i \in [n]$$

Derive the dual form expression along with any constraints. Work must be shown. *Hints*: Refer to the methodology that was used in class to derive the dual form. The solution is given by:

$$\arg\max_{\alpha} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j - \frac{1}{2C}\sum_{i=1}^{n}\alpha_i^2$$
$$\text{s.t. } \alpha_i \geq 0 \quad \forall i \in [n] \quad \text{and} \quad \sum_{i=1}^{n}\alpha_i y_i = 0$$

solution:

- form the lagrangian function:

$\|w\| = \sqrt{w^T w}$

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 - \sum_{i=1}^{n}\alpha_i\left[y_i(w^T x_i + b) - 1 + \xi_i\right]$$

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n}\alpha_i y_i x_i = 0$$

$$\Rightarrow w = \sum_{i=1}^{n}\alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n}\alpha_i y_i = 0$$

$$w^T w = \sum_{i=1}^{n}\alpha_i y_i^T x_i^T$$
$$\sum_{j=1}^{n}\alpha_j y_j \alpha_j$$

$$\frac{\partial L}{\partial \xi_i} = C\xi_i - \alpha_i = 0$$

$$\Rightarrow \xi_i = \frac{\alpha_i}{C}$$

$$L(w, b, \xi) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j + \frac{C}{2}\sum_{i=1}^{n}\frac{\alpha_i^2}{C^2}$$
$$- \sum_{i=1}^{n}\left[\alpha_i y_i w^T x_i + \alpha_i y_i b - \alpha_i + \alpha_i\xi_i\right]$$

$$= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j + \frac{1}{2C}\sum_{i=1}^{n}\alpha_i^2$$
$$- \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_i + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\frac{2\alpha_i^2}{2C}$$

$$-\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_i + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\frac{2\alpha_i^2}{2C}$$

$$L(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_i - \frac{1}{2C}\sum_{i=1}^{n}\alpha_i^2$$

$$\text{s.t.} \quad \alpha_i \geq 0 \;, \quad \sum_{i=1}^{n}\alpha_i y_i = 0$$

$$\alpha_i = C\,\xi_i$$

2 Domain Adaptation Support Vector Machines [20pts]

We now look at a different type of SVM that is designed for domain adaptation and optimizes the hyperplanes given by $\mathbf{w}_S$ (source hyperplane) before optimizing $\mathbf{w}_T$ (target hyperplane). The process begins by training a support vector machine on source data then once data from the target are available, train a new SVM using the hyperplane from the first SVM and the data from the target to solve for a new "domain adaptation" SVM.

The primal optimization problem is given by

$$\begin{aligned}
\arg\min_{\mathbf{w}_T,\xi} \quad & \frac{1}{2}\|\mathbf{w}_T\|^2 + C\sum_{i=1}^{n}\xi_i - B\mathbf{w}_T^T\mathbf{w}_S \\
\text{s.t.} \quad & y_i(\mathbf{w}_T^T\mathbf{x}_i + b) \geq 1 - \xi_i && \forall i \in \{1,\ldots,n\} \\
& \xi_i \geq 0 && \forall i \in \{1,\ldots,n\}
\end{aligned}$$

where $\mathbf{w}_S$ is hyperplane trained on the source data (*assumed to be known*), $\mathbf{w}_T$ is hyperplane for the target, $y_i \in \{\pm 1\}$ is the label for instance $\mathbf{x}_i$, C & B are regularization parameters defined by the user and ξ_i is a slack variable for instance $\mathbf{x}_i$. The problem becomes finding a hyperplane, $\mathbf{w}_T$, that minimizes the above objective function subject to the constraints. Solve/derive the dual optimization problem.

Note: I will give the class the solution to this problem prior to the due date because Problem #3 requires that you implement this algorithm in code.

$$L(w_T, b, \xi_i) = \frac{1}{2}\|w_T\|^2 + C\sum_{i=1}^{n}\xi_i - B w_T^T w_S - \sum_{i=1}^{n}\alpha_i\left[y_i(w_T^T x_i + b) - 1 + \xi_i\right] - \sum_{i=1}^{n}\mu_i\xi_i$$

$$\frac{\partial L}{\partial w_T} = w_T - B w_S - \sum_{i=1}^{n}\alpha_i y_i x_i = 0$$

$$\Rightarrow \quad w_T = B w_S + \sum_{i=1}^{n}\alpha_i y_i \alpha_i$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n}\alpha_i y_i = 0$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \mu_i = 0$$

$$\Rightarrow C = \alpha_i + \mu_i \quad \therefore \quad 0 \leq \alpha_i \leq C$$

$$L = \frac{1}{2}\|W_T\|^2 + C\sum_{i=1}^{n}\xi_i - BW_T^T W_S - \sum_{i=1}^{n}\alpha_i\left[y_i(W_T^T x_i + b) - 1 + \xi_i\right]$$

$$- \sum_{i=1}^{n}\mu_i\xi_i$$

$$W_T^T W_T$$

$$= \frac{1}{2}\left[BW_S^T + \sum_{i=1}^{n}\alpha_i y_i x_i^T\right]\cdot\left[BW_S + \sum_{i=1}^{n}\alpha_i y_i x_i\right]$$

$$= \frac{1}{2}\Bigg[B^2 W_S^T W_S + BW_S^T\sum_{i=1}^{n}\alpha_i y_i x_i + \left(\sum_{i=1}^{n}\alpha_i y_i x_i^T\right)BW_S + \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j\Bigg]$$

$$+ C\sum_{i=1}^{n}\xi_i - B\left[BW_S^T + \sum_{i=1}^{n}\alpha_i y_i x_i^T\right]W_S$$

$$-\sum_{i=1}^{n}\alpha_i y_i BW_S^T - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\alpha_i\xi_i - \sum_{i=1}^{n}\mu_i\xi_i$$

$$= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j - \frac{1}{2}B^2 W_S^T W_S$$

$$+ \frac{1}{2}BW_S^T\sum_{i=1}^{n}\alpha_i y_i x_i - \frac{1}{2}\left(\sum_{i=1}^{n}\alpha_i y_i x_i^T\right)BW_S$$

$$\text{S.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0$$

$$C > \alpha_i \geq 0$$

Since $\frac{1}{2}B^2 W_S^T W_S$ is constant w.r.t. the optimization the equation reduces to

$$L = \frac{1}{2}\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^{n}B\alpha_i y_i W_S^T x_i$$

$$\text{S.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0$$

$$C > \alpha_i \geq 0$$

3 Density Estimation (Code) [20pts]

Implement the domain adaptation SVM from Problem #2. A data setfor the source and target domains (both training and testing) have been uploaded to D2L. There are several ways to implement this algorithm. If I were doing this for an assignment, I would implement the SVM (both the domain adaptation SVM and normal SVM) directly using quadratic programming. You do not need to build the classifier (i.e., solve for the bias term); however, you will need to find $\mathbf{w}_T$ and $\mathbf{w}_S$. To find the weight vectors, you will need to solve a quadratic programming problem and look through the documentation to learn how to solve this optiization task. The following Python packages are recommended:

- CVXOPT(https://cvxopt.org/)
- PyCVX (https://www.cvxpy.org/install/)

Note: Your solution can use any of the packages above.