IE521 - Assignment 3

Abhishek Avinash Narwekar
an41@illinois.edu

March 13, 2017

Problem 1: Support Vector Machine

Exercise 1.1: Lagrange Duality

We write the two constraints in the form of $\leq$ constraints:

$$\begin{aligned}\text{s.t}\quad & 1 - y_i(w^T x_i + b) - \epsilon_i \leq 0 \quad i = 1...m\\ & -\epsilon_i \leq 0 \quad i = 1...m\end{aligned}$$

The Lagrangian function can then be written as:

$$\begin{aligned}L(w, b, \epsilon, \alpha, \beta) &= \frac{1}{2}||w||_2^2 + C\sum_{i=1}^{m}\epsilon_i + \sum_{i=1}^{m}\alpha_i\left(1 - \epsilon_i - y_i(w^T x_i + b)\right) - \sum_{i=1}^{m}\beta_i\epsilon_i\\ &= \frac{1}{2}||w||_2^2 + \sum_{i=1}^{m}(C - \alpha_i - \beta_i)\epsilon_i + \sum_{i=1}^{m}\alpha_i - \sum_{i=1}^{m}\alpha_i y_i(w^T x_i) - \sum_{i=1}^{m}\alpha_i y_i b\end{aligned}$$

We need to minimize $L(w, b, \epsilon, \alpha, \beta)$ over w, b and ϵ. Consider the linear terms - i.e., the terms in b and ϵ. Since there is no restriction on b, we note that if the coefficient of b is non-zero, then we can simply set b to $+\infty$ or $-\infty$ to achieve the minimum. Therefore, the coefficient of b must be zero.

$$\implies \sum_{i=1}^{m}\alpha_i y_i = 0$$

For ϵ, however, we note that since $\epsilon_i \geq 0$ for $i = 1...m$, the coefficients of ϵ must all be non-negative; else, we can achieve the minimum by setting that ϵ_i whose coefficient is negative to $+\infty$. Thus:

$$\begin{aligned}& C - \alpha_i - \beta_i = 0 \quad i = 1...m\\ \implies & 0 \leq \alpha_i \leq C\\ & 0 \leq \beta_i \leq C\end{aligned}$$

The Lagrangian thus simplifies to:

$$L(w, b, \epsilon, \alpha, \beta) = \frac{1}{2}||w||_2^2 + \sum_{i=1}^{m} \alpha_i - \sum_{i=1}^{m} \alpha_i y_i (w^T x_i)$$

The Lagrangian is thus quadratic in w, and its minimum is achieved by setting the derivative with respect to w to zero:

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

Substituting back into the Lagrangian, we get:

$$\underline{L}(\alpha) = \min_{w,b,\epsilon} L(w, b, \epsilon, \alpha, \beta) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

The Lagrangian dual of the problem is therefore:

$$\begin{aligned} \max_{\alpha} \underline{L}(\alpha) = \max_{\alpha} & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \text{s.t} & \sum_{i=1}^{m} \alpha_i y_i = 0 \\ & 0 \le \alpha_i \le C \end{aligned}$$

Note that the constraint $0 \le \beta_i \le C$ is irrelevant in the dual function, since β the dual isn't a function of β.
We now investigate the consequences of α_i taking various points in its domain. We recall that:

$$\begin{aligned} 0 \le \alpha_i, \beta_i \le C \\ C = \alpha_i + \beta_i \end{aligned}$$

The complementary slackness condition for KKT optimality states that:

$$\begin{aligned} \alpha_i \left(1 - \epsilon_i - y_i(w^T x_i + b)\right) = 0 \\ -\beta_i \epsilon_i = 0 \end{aligned}$$

We know, that either the primal constraint is satisfied **or** its Lagrangian multiplier is zero. With this in mind, we consider the following three cases:

- Case 1: $\alpha_i = 0$

$$\begin{aligned} &\implies \beta_i = C \implies \epsilon_i = 0 \quad \text{(primal constraint 2)} \\ &\implies 1 - y_i(w^T x_i + b) \le 0 \quad \text{(substituting in primal constraint 1, which is satisfied)} \end{aligned}$$

- Case 2: $0 < \alpha_i < C$

$$\begin{aligned} &\implies \beta_i = C - \alpha > 0 \implies \epsilon_i = 0 \quad \text{(primal constraint 2)} \\ &\implies 1 - y_i(w^T x_i + b) = 0 \quad \text{(substituting in primal constraint 1, which is 0)} \end{aligned}$$

- Case 3: $\alpha_i = C$

$$\begin{aligned} &\implies \beta_i = 0 \implies \epsilon_i \geq 0 \quad \text{(primal constraint 2 is satisfied)} \\ &\implies 1 - \epsilon_i - y_i(w^T x_i + b) = 0 \\ &\implies 1 - y_i(w^T x_i + b) = \epsilon_i \geq 0 \quad \text{(substituting into primal constraint 1, which is 0)} \end{aligned}$$

Exercise 1.2: Reformulation

The original convex optimization problem is:

$$\begin{aligned} \min_{w,b,\epsilon} \quad & \frac{1}{2}||w||_2^2 + C\sum_{i=1}^{m} \epsilon_i \\ \text{s.t} \quad & y_i(w^T x_i + b) \geq 1 - \epsilon_i \quad i = 1...m \\ & \epsilon_i \geq 0 \quad i = 1...m \end{aligned}$$

From the two constraints, we observe: $\epsilon_i \geq 1 - y_i(w^T x_i + b)$ and $\epsilon \geq 0$ for $i = 1...m$. Therefore, we conclude: $\epsilon_i \geq \max(1 - y_i(w^T x_i + b), 0)$. Further, we can rewrite the objective function as:

$$\begin{aligned} f(w, b, \epsilon) = \frac{1}{2}||w||_2^2 + C\sum_{i=1}^{m} \epsilon_i &= mC(\frac{1}{m}\sum_{i=1}^{m} \epsilon_i + \frac{1}{2mC}||w||_2^2) \\ &= mC(\frac{1}{m}\sum_{i=1}^{m} \epsilon_i + \lambda||w||_2^2) \end{aligned}$$

Since $mC > 0$, we can reformulate the minimization problem as:

$$\min_{w,b,\epsilon} \quad \frac{1}{m}\sum_{i=1}^{m} \epsilon_i + \lambda||w||_2^2$$

Now we note that $f(w, b, \epsilon) = \frac{1}{m}\sum_{i=1}^{m} \epsilon_i + \lambda||w||_2^2 \geq \frac{1}{m}\sum_{i=1}^{m} \max(1 - y_i(w^T x_i + b), 0) + \lambda||w||_2^2$. Given that the problem is one of minimization, and that ϵ_i is has no other constraints or dependencies except that $\epsilon_i \geq \max(1 - y_i(w^T x_i + b), 0)$, the argument that minimizes the objective function will contain:

$$\epsilon_i = \max(1 - y_i(w^T x_i + b), 0)$$

Since the only constraints were on ϵ, and since we eliminated the dependency of the objective function on ϵ, the original problem can be reformulated in to the following **unconstrained** optimization problem:

$$\min_{w,b} \quad \frac{1}{m}\sum_{i=1}^{m} \max(1 - y_i(w^T x_i + b), 0) + \lambda||w||_2^2 \quad \text{where } \lambda > 0$$

Exercise 1.3: Programming

Since this problem is unconstrained, we do not need to implement the separation oracle. The update step is governed by the subgradient alone. To compute the subgradient, we compute an intermediate vector g at each time step, computed as follows:

$$g[i] = \begin{cases} 1 & \text{if } 1 - y_i(w^T x) + b > 0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

This definition tells us which region of the search space we are in for the purpose of computing the subgradient. The subgradient with respect to b is:

$$\frac{\partial f}{\partial b} = -\frac{1}{m} y \odot g$$

The subgradient with respect to w is:

$$\begin{aligned} \frac{\partial f}{\partial w} &= \frac{1}{m}\left(-\sum_{i=1}^{m} (X_i y_i) \odot g + 2\lambda w \right) \\ &= \frac{1}{m}\left(-X^T y \odot g + 2\lambda w \right) \end{aligned}$$

Since $||w^*||_2 \leq 1/\sqrt{\lambda}$, we set $R = 5$ as the initial radius of the ball. The update equations were implemented as discussed in class.

There are two relevant functions in the code: `readWDBC()` and `ellipsoid()`. The first one reads the database into numpy matrices, and the second one implements the ellipsoid method by taking the following inputs: the input matrices `X` and `y`, the maximum number of time steps `nmax` and the value of λ (`lam`). It then plots the value of the objective function and the cumulative minimum of the objective function over the time steps. The code can be viewed in the file `ellipsoid.py`, and has also been printed below for reference.

```python
# ellpisoid.py
import matplotlib.pyplot as plt
import numpy as np

def readWDBC():
    f = open('wdbc.data','r')
    X = []
    Y = []
    labelMap = {'B':1.0,'M':-1.0}
    for line in f.readlines():
        lsplit = line.strip().split(',')
        Y.append(labelMap[lsplit[1]])
        toAdd = []
        for x in lsplit[2:]:
            toAdd.append(float(x))
        X.append(toAdd)
    return np.array(X), np.reshape(np.array(Y),(len(Y),1))

def ellipsoid(X,Y, lam=1, nmax=100, plotFlag=True):
    # Initialize the following: parameters - w,b; subgradient - omega;
    # incorrect classifications - wrong; ellipse - Q, mat
    n = X.shape[1]
    m = X.shape[0]
    c = np.zeros((n+1,1))
    omega = np.zeros(c.shape) # Sub-gradient

    lossPlot = []
    minlossPlot = []

    R = 5
    Q = np.identity(n+1)*(R*R) # Ellipsoid
    for i in range(nmax):
        classifications = 1 - Y*(X.dot(c[0:n])+c[n]) # 1 - y.*(w^T x + b)
        wrong = classifications > 0
        loss = 1.0/m*sum(wrong*classifications) + lam*c[:n].T.dot(c[:n])
        lossPlot.append(float(loss))
        minlossPlot.append(min(lossPlot))

        # if i%10 == 0:
        #     print i, loss

        sum_xy = np.zeros((n,1))
        for k in range(m):
            sum_xy += np.reshape(Y[k]*X[k,:]*wrong[k],(n,1))
        omega[:n] = (2*lam*c[:n] - sum_xy )/ m
        omega[n] = -np.sum(Y*wrong)/m

        # Update c, Q
        c = c - 1.0 / ( (n+1) * np.sqrt( omega.T.dot(Q).dot(omega) +1e-10) ) * Q.dot(omega)
        Q = float(n*n)/(n*n-1) * (Q - 2.0/(n+1) * Q.dot(omega).dot(omega.T).dot(Q) / (
            omega.T.dot(Q).dot(omega) + 1e-10 ))

    if plotFlag:
        fig1 = plt.figure()
        ax1 = fig1.add_subplot(111)

        ax1.plot(range(0,len(lossPlot)),lossPlot, label='Objective Function')
        ax1.plot(range(0,len(lossPlot)),minlossPlot, label='Cumulative Minimum of Objective
            Function')

        plt.xlabel('Timestep')
        plt.ylabel('Objective Function')
        plt.title('Variation of Objective Function Over Time')
        plt.legend()
        plt.show()

    print 'Misclassification Error:', float(sum(wrong)) / m

if __name__ == '__main__':
    (X,Y) = readWDBC()
    ellipsoid(X,Y)
```

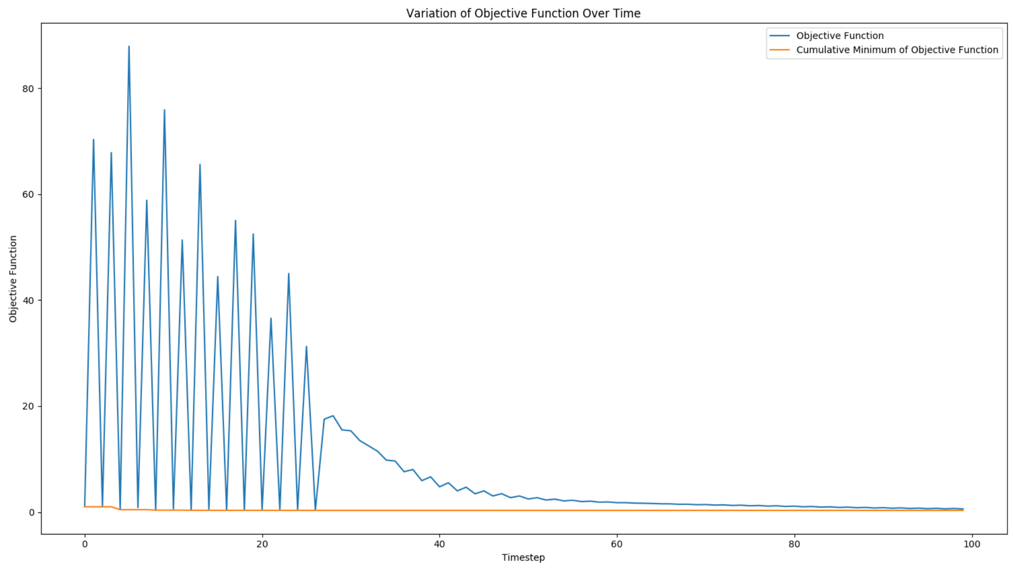

Figure 1: Plot of objective function over time steps

Exercise 1.4: Test on Real Dataset

For $\lambda = 1$ and $T = 100$, the plot for the objective function is shown in figure 1. The minimum value of the objective function, $\min_t f(w_t) = 0.3173$. After 100 iterations, the ratio of misclassified points to all points is **0.1388**.