NMK40403 ARTIFICIAL INTELLIGENCE

SUPPORT VECTOR MACHINES
Lec 08

Solving the Optimization Problem

Mohamed Elshaikh

Solving the Optimization Problem

Lagrange multipliers

The Italian-French mathematician **Giuseppe Lodovico Lagrangia**, also known as **Joseph-Louis Lagrange**, invented a strategy for finding the local maxima and minima of a function subject to equality constraint. It is called the method of Lagrange multipliers.

The method of Lagrange multipliers

Lagrange noticed that when we try to solve an optimization problem of the form:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g(\mathbf{x}) = 0 \end{aligned}$$

the minimum of f is found when its gradient point in the same direction as the gradient of g. In other words, when:

$$\nabla f(\mathbf{x}) = \alpha \nabla g(\mathbf{x})$$

So if we want to find the minimum of f under the constraint g, we just need to solve for:

$$\nabla f(\mathbf{x}) - \alpha \nabla g(\mathbf{x}) = 0$$

Solving the Optimization Problem

Here, the constant α is called a Lagrange multiplier.

To simplify the method, we observe that if we define a function $\mathcal{L}(\mathbf{x}, \alpha) = f(\mathbf{x}) - \alpha g(\mathbf{x})$, then its gradient is $\nabla\mathcal{L}(\mathbf{x}, \alpha) = \nabla f(\mathbf{x}) - \alpha\nabla g(\mathbf{x})$. As a result, solving for $\nabla\mathcal{L}(\mathbf{x}, \alpha) = 0$ allows us to find the minimum.

The Lagrange multiplier method can be summarized by these three steps:

1. Construct the Lagrangian function $\mathcal{L}$ by introducing one multiplier per constraint
2. Get the gradient $\nabla\mathcal{L}$ of the Lagrangian
3. Solve for $\nabla\mathcal{L}(\mathbf{x}, \alpha) = 0$

The SVM Lagrangian problem

We saw in the last chapter that the SVM optimization problem is:

$$\begin{aligned} &\underset{\mathbf{w},b}{\text{minimize}} && \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{subject to} && y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 \geq 0,\ i = 1,\ldots,m \end{aligned}$$

Solving the Optimization Problem

Let us return to this problem. We have one objective function to minimize:

$$f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$$

and m constraint functions:

$$g_i(\mathbf{w}, b) = y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1, \ i = 1, \ldots, m$$

We introduce the Lagrangian function:

$$\mathcal{L}(\mathbf{w}, b, \alpha) = f(\mathbf{w}) - \sum_{i=1}^{m} \alpha_i g_i(\mathbf{w}, b)$$

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i \left[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right]$$

Note that we introduced one **Lagrange multiplier** α_i for each constraint function.

Solving the Optimization Problem

We could try to solve for $\mathcal{L}(\mathbf{w}, b, \alpha) = 0$, but the problem can only be solved analytically when the number of examples is small (Tyson Smith, 2004). So we will once again rewrite the problem using the duality principle.

To get the solution of the primal problem, we need to solve the following **Lagrangian problem**:

$$\begin{aligned} &\min_{\mathbf{w},b} \quad \max_{\alpha} \quad \mathcal{L}(\mathbf{w}, b, \alpha) \\ &\text{subject to} \quad \alpha_i \geq 0, \quad i = 1, \ldots, m \end{aligned}$$

What is interesting here is that we need to minimize with respect to $\mathbf{w}$ and b, and to maximize with respect to α at the same time.

Tip: You may have noticed that the method of Lagrange multipliers is used for solving problems with equality constraints, and here we are using them with inequality constraints. This is because the method still works for inequality constraints, provided some additional conditions (the KKT conditions) are met. We will talk about these conditions later.

Solving the Optimization Problem

The Wolfe dual problem

The Lagrangian problem has m inequality constraints (where m is the number of training examples) and is typically solved using its dual form. The **duality principle** tells us that an optimization problem can be viewed from two perspectives. The first one is the primal problem, a minimization problem in our case, and the other one is the dual problem, which will be a maximization problem. What is interesting is that the maximum of the dual problem will always be less than or equal to the minimum of the primal problem (we say it provides a lower bound to the solution of the primal problem).

In our case, we are trying to solve a convex optimization problem, and **Slater's condition** holds for affine constraints (Gretton, 2016), so **Slater's theorem** tells us that **strong duality** holds. This means that the maximum of the dual problem is equal to the minimum of the primal problem. Solving the dual is the same thing as solving the primal, except it is easier.

Recall that the Lagrangian function is:

$$\begin{aligned}\mathcal{L}(\mathbf{w}, b, \alpha) &= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] \\ &= \frac{1}{2}\mathbf{w} \cdot \mathbf{w} - \sum_{i=1}^{m} \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1]\end{aligned}$$

Solving the Optimization Problem

The Lagrangian primal problem is:

$$\begin{aligned} &\min_{\mathbf{w},b} \quad \max_{\alpha} \quad \mathcal{L}(\mathbf{w}, b, \alpha) \\ &\text{subject to} \quad \alpha_i \geq 0, \;\; i = 1, \ldots, m \end{aligned}$$

Solving the minimization problem involves taking the partial derivatives of $\mathcal{L}$ with respect to $\mathbf{w}$ and b.

$$\nabla_{\mathbf{w}} \mathcal{L} = \mathbf{w} - \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i = \mathbf{0}$$

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

From the first equation, we find that:

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$

Solving the Optimization Problem

Let us substitute $\mathbf{w}$ by this value into $\mathcal{L}$:

$$\begin{aligned}
W(\alpha, b) &= \frac{1}{2}\Big(\sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i\Big) \cdot \Big(\sum_{j=1}^{m} \alpha_j y_j \mathbf{x}_j\Big) - \sum_{i=1}^{m} \alpha_i \Big[y_i \Big(\big(\sum_{j=1}^{m} \alpha_j y_j \mathbf{x}_j\big) \cdot \mathbf{x}_i + b \Big) - 1 \Big] \\
&= \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - \sum_{i=1}^{m} \alpha_i y_i \Big(\big(\sum_{j=1}^{m} \alpha_j y_j \mathbf{x}_j\big) \cdot \mathbf{x}_i + b \Big) + \sum_{i=1}^{m} \alpha_i \\
&= \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - b \sum_{i=1}^{m} \alpha_i y_i + \sum_{i=1}^{m} \alpha_i \\
&= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - b \sum_{i=1}^{m} \alpha_i y_i
\end{aligned}$$

Solving the Optimization Problem

So we successfully removed $\mathbf{w}$, but b is still used in the last term of the function:

$$W(\alpha, b) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - b\sum_{i=1}^{m} \alpha_i y_i$$

We note that $\frac{\partial \mathcal{L}}{\partial b} = 0$ implies that $\sum_{i=1}^{m} \alpha_i y_i = 0$. As a result, the last term is equal to zero, and we can write:

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

This is the **Wolfe dual Lagrangian function**.

The optimization problem is now called the **Wolfe dual problem**:

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ \text{subject to} \quad & \alpha_i \geq 0, \text{ for any } i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y_i = 0 \end{aligned}$$

Solving the Optimization Problem

Traditionally the Wolfe dual Lagrangian problem is constrained by the gradients being equal to zero. In theory, we should add the constraints $\nabla_{\mathbf{w}}\mathcal{L} = 0$ and $\frac{\partial \mathcal{L}}{\partial b} = 0$. However, we only added the latter. Indeed, we added $\sum_{i=1}^{m} \alpha_i y_i = 0$ because it is necessary for removing b from the function. However, we can solve the problem without the constraint $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$.

The main advantage of the Wolfe dual problem over the Lagrangian problem is that the objective function W now depends only on the Lagrange multipliers. Moreover, this formulation will help us solve the problem in Python in the next section and will be very helpful when we define kernels later.

Solving the Optimization Problem

Karush-Kuhn-Tucker conditions

Because we are dealing with inequality constraints, there is an additional requirement: the solution must also satisfy the Karush-Kuhn-Tucker (KKT) conditions.

The KKT conditions are first-order **necessary** conditions for a solution of an optimization problem to be optimal. Moreover, the problem should satisfy some regularity conditions. Luckily for us, one of the regularity conditions is **Slater's condition**, and we just saw that it holds for SVMs. Because the primal problem we are trying to solve is a convex problem, the KKT conditions are also **sufficient** for the point to be primal and dual optimal, and there is zero duality gap.

Note: "[...]Solving the SVM problem is equivalent to finding a solution to the KKT conditions." (Burges, 1988)

Solving the Optimization Problem

- **To sum up, if a solution satisfies the KKT conditions, we are guaranteed that it is the optimal solution.**

The Karush-Kuhn-Tucker conditions are:

- Stationarity condition:

$$\nabla_{\mathbf{w}} \mathcal{L} = \mathbf{w} - \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i = \mathbf{0}$$

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

- Primal feasibility condition:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0 \qquad \text{for all } i = 1, \ldots, m$$

- Dual feasibility condition:

$$\alpha_i \geq 0 \qquad \text{for all } i = 1, \ldots, m$$

- Complementary slackness condition:

$$\alpha_i \big[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \big] = 0 \qquad \text{for all } i = 1, \ldots, m$$

Solving the Optimization Problem

Stationarity condition

The stationarity condition tells us that the selected point must be a **stationary point**. It is a point where the function stops increasing or decreasing. When there is no constraint, the stationarity condition is just the point where the gradient of the objective function is zero. When we have constraints, we use the gradient of the Lagrangian.

Primal feasibility condition

Looking at this condition, you should recognize the constraints of the primal problem. It makes sense that they must be enforced to find the minimum of the function under constraints.

Dual feasibility condition

Similarly, this condition represents the constraints that must be respected for the dual problem.

Complementary slackness condition

From the complementary slackness condition, we see that either $\alpha_i = 0$ or $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 = 0$.

Support vectors are examples having a positive Lagrange multiplier. They are the ones for which the constraint $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0$ is **active**. (We say the constraint is active when $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 = 0$).

Solving the Optimization Problem
What to do once we have the multipliers?

When we solve the Wolfe dual problem, we get a vector α containing all Lagrange multipliers. However, when we first stated the primal problem, our goal was to find $\mathbf{w}$ and b. Let us see how we can retrieve these values from the Lagrange multipliers.

Compute w

Computing $\mathbf{w}$ is pretty simple since we derived the formula: $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$ from the gradient $\nabla_{\mathbf{w}} \mathcal{L}$.

Compute b

Once we have $\mathbf{w}$, we can use one of the constraints of the primal problem to compute b:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0$$

Indeed, this constraint is still true because we transformed the original problem in such a way that the new formulations are equivalent. What it says is that the closest points to the hyperplane will have a functional margin of 1 (the value 1 is the value we chose when we decided how to scale $\mathbf{w}$):

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1$$

Solving the Optimization Problem

From there, as we know all other variables, it is easy to come up with the value of b. We multiply both sides of the equation by y_i , and because $y_i^2 = 1$, it gives us:

$$\mathbf{w} \cdot \mathbf{x}_i + b = y_i$$

$$b = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

However, as indicated in *Pattern Recognition and Machine Learning* (Bishop, 2006), instead of taking a random support vector $\mathbf{x}_i$, taking the average provides us with a numerically more stable solution:

$$b = \frac{1}{S} \sum_{i=1}^{S} (y_i - \mathbf{w} \cdot \mathbf{x}_i)$$

where S is the number of support vectors.

Other authors, such as (Cristianini & Shawe-Taylor, 2000) and (Ng), use another formula:

$$b = -\frac{max_{y_i=-1}(\mathbf{w} \cdot \mathbf{x}_i) + min_{y_i=1}(\mathbf{w} \cdot \mathbf{x}_i)}{2}$$

They basically take the average of the nearest positive support vector and the nearest negative support vector. This latest formula is the one originally used by *Statistical Learning Theory* (Vapnik V. N., 1998) when defining the optimal hyperplane.

Hypothesis function

The SVMs use the same hypothesis function as the Perceptron. The class of an example $\mathbf{x}_i$ is given by:

$$h(\mathbf{x}_i) = \text{sign}(\mathbf{w} \cdot \mathbf{x}_i + b)$$

When using the dual formulation, it is computed using only the support vectors:

$$h(\mathbf{x}_i) = \text{sign}\left(\sum_{j=1}^{S} \alpha_j y_j (\mathbf{x}_j \cdot \mathbf{x}_i) + b \right)$$

Solving the Optimization Problem

Solving SVMs with a QP solver

A QP solver is a program used to solve quadratic programming problems. In the following example, we will use the Python package called CVXOPT.

This package provides a method that is able to solve quadratic problems of the form:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} \quad & Gx \preceq h \\ & Ax = b \end{aligned}$$

It does not look like our optimization problem, so we will need to rewrite it so that we can solve it with this package.

First, we note that in the case of the Wolfe dual optimization problem, what we are trying to minimize is α, so we can rewrite the quadratic problem with α instead of x to better see how the two problems relate:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{2}\alpha^T P \alpha + q^T \alpha \\ \text{subject to} \quad & G\alpha \preceq h \\ & A\alpha = b \end{aligned}$$

Here the $\preceq$ symbol represents component-wise vector inequalities. It means that each row of the matrix $G\lambda$ represents an inequality that must be satisfied.

We will change the Wolfe dual problem. First, we transform the maximization problem:

$$\begin{aligned}
&\underset{\alpha}{\text{maximize}} \quad -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j + \sum_{i=1}^{m}\alpha_i \\
&\text{subject to} \quad \alpha_i \geq 0, \text{ for any } i = 1,\dots,m \\
&\qquad\qquad\quad \sum_{i=1}^{m}\alpha_i y_i = 0
\end{aligned}$$

into a minimization problem by multiplying by -1.

$$\begin{aligned}
&\underset{\alpha}{\text{minimize}} \quad \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - \sum_{i=1}^{m}\alpha_i \\
&\text{subject to} \quad -\alpha_i \leq 0, \text{ for any } i = 1,\dots,m \\
&\qquad\qquad\quad \sum_{i=1}^{m}\alpha_i y_i = 0
\end{aligned}$$

Then we introduce vectors $\alpha = (\alpha_1, \ldots, \alpha_m)^T$ and $\mathbf{y} = (y_1, \cdots, y_m)^T$ and the Gram matrix K of all possible dot products of vectors $\mathbf{x}_i$:

$$K(\mathbf{x}_1, \ldots, \mathbf{x}_m) = \begin{pmatrix} \mathbf{x}_1 \cdot \mathbf{x}_1 & \mathbf{x}_1 \cdot \mathbf{x}_2 & \ldots & \mathbf{x}_1 \cdot \mathbf{x}_m \\ \mathbf{x}_2 \cdot \mathbf{x}_1 & \mathbf{x}_2 \cdot \mathbf{x}_2 & \ldots & \mathbf{x}_2 \cdot \mathbf{x}_m \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_m \cdot \mathbf{x}_1 & \mathbf{x}_m \cdot \mathbf{x}_2 & \ldots & \mathbf{x}_m \cdot \mathbf{x}_m \end{pmatrix}$$

We use them to construct a vectorized version of the Wolfe dual problem where $\mathbf{y}\mathbf{y}^T$ denotes the outer product of $\mathbf{y}$.

$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & \frac{1}{2}\alpha^T(\mathbf{y}\mathbf{y}^T K)\alpha - \alpha \\ \text{subject to} \quad & -\alpha \leq 0, \\ & \mathbf{y} \cdot \alpha = 0 \end{aligned}$$

We are now able to find out the value for each of the parameters P, q, G, h, A, and b required by the CVXOPT **qp** function.

Solving the Optimization Problem

When we plot the result in Figure 32, we see that the hyperplane is the optimal hyperplane. Contrary to the Perceptron, the SVM will always return the same result.

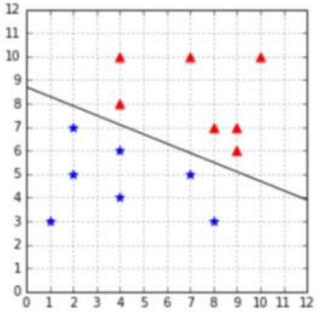

Figure 32: The hyperplane found with CVXOPT

This formulation of the SVM is called the **hard margin SVM**. It cannot work when the data is not linearly separable. There are several Support Vector Machines formulations. In the next chapter, we will consider another formulation called the **soft margin SVM**, which will be able to work when data is non-linearly separable because of outliers.

Summary

Minimizing the norm of **w** is a **convex optimization problem**, which can be solved using the Lagrange multipliers method. When there are more than a few examples, we prefer using convex optimization packages, which will do all the hard work for us.

We saw that the original optimization problem can be rewritten using a Lagrangian function. Then, thanks to duality theory, we transformed the Lagrangian problem into the Wolfe dual problem. We eventually used the package CVXOPT to solve the Wolfe dual.

END