Support Vector Machines

Support vector machines (SVMs) are one of the central concepts in all of machine learning. They are simply a combination of two ideas: linear classification via maximum (or optimal soft) margin hyperplanes, and kernels.

Conceptually, give training data $(\boldsymbol{x}_i, y_i),\ i = 1, \ldots, n$, with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$, we are going to

1. Put the feature vectors through a nonlinear map: $\boldsymbol{x}_i \to \boldsymbol{\Phi}(\boldsymbol{x}_i)$. This takes them from $\mathbb{R}^d$ to some abstract Hilbert space $\mathcal{H}$.
2. Fit a linear classifier in $\mathcal{H}$. Since $\mathcal{H}$ has a valid inner product, there is a notion of inner product (hence projections onto subspaces) and distance in $\mathcal{H}$, making this problem well-defined.
3. Translate the rule back to a function (classifier) in $\mathbb{R}^d$ using

$$h(\boldsymbol{x}) = \text{sign}\left(\langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x}) \rangle + b\right)$$

The "trick" is that step 1 is actually implicit. By using a kernel, we avoid having to actually compute the map. This is also true for the inner product $\langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x}) \rangle$; we will see that this can also be computed using a kernel.

To see how this works, we turn to our recently acquired knowledge of constrained optimization and duality.

Georgia Tech ECE 6254 Spring 2024; Notes by M. Bloch, M. A. Davenport, and J. Romberg. Last updated 14:08, February 22, 2024

Dual of optimal soft margin

The optimal soft-margin classifier is found by solving

$$\begin{aligned} \text{(SM)} \qquad & \underset{b,\boldsymbol{w},\boldsymbol{\xi}}{\text{minimize}} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \frac{C}{n}\sum_{i=1}^n \xi_i \\ & \text{subject to} \qquad y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, n. \\ & \qquad\qquad\qquad\qquad \xi_i \geq 0, \quad i = 1, \ldots, n. \end{aligned}$$

If we try to solve this program after we map $\boldsymbol{x}_i \to \boldsymbol{\Phi}(\boldsymbol{x}_i)$, then the optimization program is looking for a $\boldsymbol{w}$ in the same space as $\boldsymbol{\Phi}(\boldsymbol{x}_i)$. Since this space will in general be much higher dimensional than $\mathbb{R}^d$ — and indeed could very well be infinite dimensional — this is problematic.

Fortunately, duality and the KKT conditions will come to our rescue.

Here are two facts, which we verify in the later sections of these notes.

1. The **dual** of program (SM) is

$$\begin{aligned} \text{(DSM)} \qquad & \underset{\boldsymbol{\alpha}\in\mathbb{R}^n}{\text{maximize}} \ -\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j + \sum_{i=1}^n \alpha_i \\ & \text{subject to} \qquad \sum_{i=1}^n \alpha_i y_i = 0 \\ & \qquad\qquad\qquad\qquad 0 \leq \alpha_i \leq C/n, \quad i = 1, \ldots, n. \end{aligned}$$

   Note that this is an optimization program in n variables. In fact, just like (SM), it is a (concave) quadratic program with linear equality and inequality constraints.

2. Given the solution $\boldsymbol{\alpha}^*$ to the dual, we can easily compute the primal solution $\boldsymbol{w}^*, b^*$. The weights $\boldsymbol{w}^*$ are computed as

$$\boldsymbol{w}^* = \sum_{n=1}^{n} \alpha_i^* y_i \boldsymbol{x}_i.$$

Recall that the weights are used to define a linear functional which the classifier evaluates, and note that the expression above implies

$$\boldsymbol{w}^{*\mathrm{T}} \boldsymbol{x} = \sum_{i=1}^{n} \alpha_i^* y_i \, \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}.$$

The offset b^* is simply[1]

$$b^* = y_{i_0} - \boldsymbol{w}^{*\mathrm{T}} \boldsymbol{x}_{i_0} = y_{i_0} - \sum_{i=1}^{n} \alpha_i^* y_i \, \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_{i_0}, \tag{1}$$

where i_0 is any index where the inequality constraint for α_{i_0} is slack: $0 < \alpha_{i_0} < C/n$.

The kernelized dual

The dual program and the computation of the affine functional $\boldsymbol{w}^{*\mathrm{T}} \boldsymbol{x} + b^*$ rely on computations of inner products between feature vectors. We can kernelize the procedure above by replacing the inner products with a kernel evaluation,

$$\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j \to k(\boldsymbol{x}_i, \boldsymbol{x}_j),$$

[1] In practice, you might average the associated values over all such indexes that have this property.

where $k(\cdot,\cdot)$ is a symmetric positive semidefinite kernel. As we have seen, $k(\boldsymbol{x}_i,\boldsymbol{x}_j) = \langle \boldsymbol{\Phi}(\boldsymbol{x}_i), \boldsymbol{\Phi}(\boldsymbol{x}_j)\rangle$ for some associated nonlinear mapping $\boldsymbol{\Phi}(\cdot)$ and inner product $\langle\cdot,\cdot\rangle$ in an abstract Hilbert space.

We compute $\boldsymbol{\alpha}^*$ by solving

$$\begin{aligned} \text{(KDSM)} \qquad \underset{\boldsymbol{\alpha}\in\mathbb{R}^N}{\text{maximize}} \quad & -\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y_i y_j\, k(\boldsymbol{x}_i,\boldsymbol{x}_j) + \sum_{i=1}^n \alpha_i \\ \text{subject to} \quad & \sum_{i=1}^n \alpha_i y_i = 0 \\ & 0 \le \alpha_i \le C/n, \quad i=1,\dots,n. \end{aligned}$$

Notice that no matter what $k(\cdot,\cdot)$ is, this is an optimization program in $\mathbb{R}^n$. Given a solution $\boldsymbol{\alpha}^*$ to the above, we define our classifier as

$$h^*(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^n \alpha_i^* y_i\, k(\boldsymbol{x}_i,\boldsymbol{x}) + b^*\right), \tag{2}$$

where b^* is defined as in (1). Notice that back in the feature space $\mathbb{R}^d$, this classifier is in general *nonlinear*, as the set

$$\left\{\boldsymbol{x}\in\mathbb{R}^d \;:\; \sum_{i=1}^n \alpha_i^* y_i\, k(\boldsymbol{x}_i,\boldsymbol{x}) = b^*\right\},$$

is no longer a hyperplane. It is some smooth surface that depends on the properties of $k(\cdot,\cdot)$ (and of course the data).

The term *support vector machine* means solving the optimization program (KDSM) to implement the classifier (2). As we dig a little deeper into how the dual is derived, we will see where this term "support vector" comes from.

General approach to "kernelization"

Optimal soft margin is not the only linear machine learning algorithm that can be kernelized. It can be applied to other classification and regression algorithms as well (we will see more on the regression side of things a little later in the course).

There are three main steps to kernelizing an algorithm:

1. Show that the training process depends only on inner products between input feature vectors, $\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j$.
2. Show that the applying the decision rule to a general $\boldsymbol{x}$ only involves computing inner products (i.e. $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$). (This means that your classification rule is linear.)
3. Replace all inner products with evaluations of the kernel function $k(\cdot, \cdot)$.

In short, this idea of kernleization has applications well beyond SVMs.

Technical details: Primals and duals in convex programming

For a general convex program[2] with inequality constraints,

$$\begin{aligned} \text{(P)} \qquad & \underset{\boldsymbol{x}}{\text{minimize}} \;\; f(\boldsymbol{x}) \\ & \text{subject to} \qquad g_m(\boldsymbol{x}) \leq 0, \quad m = 1, \ldots, M \end{aligned}$$

the Lagragian is

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{m=1}^{M} \lambda_m \, g_m(\boldsymbol{x}).$$

Recall that if $\boldsymbol{x}_0$ is feasible, meaning

$$g_m(\boldsymbol{x}_0) \leq 0, \quad \text{for all } m = 1, \ldots, M,$$

then clearly

$$L(\boldsymbol{x}_0, \lambda) \leq f(\boldsymbol{x}_0) \quad \text{for all } \boldsymbol{\lambda} \geq \mathbf{0}.$$

And in fact,

$$\max_{\boldsymbol{\lambda} \geq \mathbf{0}} L(\boldsymbol{x}_0, \boldsymbol{\lambda}) = f(\boldsymbol{x}_0).$$

Note that if $\boldsymbol{x}'$ is not feasible ($g_m(\boldsymbol{x}') > 0$ for some m), then

$$\max_{\boldsymbol{\lambda} \geq \mathbf{0}} L(\boldsymbol{x}', \boldsymbol{\lambda}) = +\infty.$$

Thus solving (P) is the same as solving

$$\underset{\boldsymbol{x}}{\text{minimize}} \; \underset{\boldsymbol{\lambda} \geq \mathbf{0}}{\text{maximize}} \, L(\boldsymbol{x}, \boldsymbol{\lambda}).$$

[2] This means that $f(\cdot)$ and the $g_m(\cdot)$ are convex functions.

The *dual* optimization program switches the order of the min and max, solving

$$\begin{aligned} \text{(D)} \qquad & \underset{\boldsymbol{\lambda}}{\text{maximize}} \;\; d(\boldsymbol{\lambda}) \\ & \text{subject to} \quad \boldsymbol{\lambda} \geq \mathbf{0}, \end{aligned}$$

where

$$d(\boldsymbol{\lambda}) = \min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}).$$

If p^* is the optimal value of the primal program (P) and d^* is the optimal value of the dual (D), then it is always true that $d^* \leq p^*$. Under certain conditions on the constraint functions $g_m(\boldsymbol{x})$, we have strong duality, meaning $d^* = p^*$. One example of these conditions is that the $g_m(\boldsymbol{x})$ are affine[3], which is the case for optimal soft margin.

In addition, when the $g_m(\boldsymbol{x})$ are affine, the KKT conditions are necessary and sufficient. That is, any solution $\boldsymbol{x}^*$ to the primal (P) and any solution $\boldsymbol{\lambda}^*$ to the dual (D) will obey:

$$\begin{aligned} g_m(\boldsymbol{x}^*) &\leq 0, \quad m = 1, \ldots, M, \\ \lambda_m &\geq 0, \quad m = 1, \ldots, M, \\ \lambda_m g_m(\boldsymbol{x}^*) &= 0, \quad m = 1, \ldots, M, \\ \nabla f(\boldsymbol{x}^*) + \sum_{m=1}^{M} \lambda_m^* \nabla g_m(\boldsymbol{x}^*) &= \mathbf{0}. \end{aligned}$$

[3]This means they can be written as $g_m(\boldsymbol{x}) = \boldsymbol{a}_m^{\mathrm{T}} \boldsymbol{x} + b_m$ for some vector $\boldsymbol{a}_m$ and scalar b_m.

Example: The dual of a QP

We know that optimal soft margin is a quadratic program with linear inequality constraints (QP). So for a quick example of how to derive a dual function, let's look at the general QP:

$$\text{(QP)} \qquad \begin{aligned} &\underset{\boldsymbol{z}\in\mathbb{R}^N}{\text{minimize}} && \frac{1}{2}\boldsymbol{z}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{z} + \boldsymbol{c}^{\mathrm{T}}\boldsymbol{z} \\ &\text{subject to} && \boldsymbol{A}\boldsymbol{z} \leq \boldsymbol{q}. \end{aligned}$$

Above, $\boldsymbol{A}$ is an $M \times N$ matrix, and $\boldsymbol{q} \in \mathbb{R}^M$. The expression $\boldsymbol{A}\boldsymbol{z} \leq \boldsymbol{q}$ is a compact way to write the M affine inequality constraints

$$g_m(\boldsymbol{z}) = \boldsymbol{a}_m^{\mathrm{T}}\boldsymbol{z} - q_m \leq 0,$$

where $\boldsymbol{a}_m^{\mathrm{T}}$ is mth row of $\boldsymbol{A}$, and q_m is the mth entry in $\boldsymbol{q}$. For (QP) to be convex, the $N \times N$ matrix $\boldsymbol{P}$ needs to be symmetric postive semidefinite (i.e. all of its eigenvalues are ≥ 0). In this example, we will also assume $\boldsymbol{P}$ is invertible — the discussion below can be extended to the general case, but this assumption just makes the derivation cleaner.

The Lagrangian is

$$L(\boldsymbol{z}, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{z}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{z} + \boldsymbol{c}^{\mathrm{T}}\boldsymbol{z} + \sum_{m=1}^{M} \lambda_m(\boldsymbol{a}_m^{\mathrm{T}}\boldsymbol{z} - q_m).$$

Notice that we can write the last term above more compactly as

$$\sum_{m=1}^{M} \lambda_m(\boldsymbol{a}_m^{\mathrm{T}}\boldsymbol{z} - q_m) = \boldsymbol{\lambda}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{z} - \boldsymbol{q}).$$

This is a convex quadratic function, so finding its unconstrained minimum amounts to finding a point where the gradient is equal to zero.

Since

$$\nabla_z L(\boldsymbol{z}, \boldsymbol{\lambda}) = \boldsymbol{P}\boldsymbol{z} + \boldsymbol{c} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda},$$

we know that $\min_{\boldsymbol{z}} L(\boldsymbol{z}, \boldsymbol{\lambda})$ will be achieved when

$$\boldsymbol{P}\boldsymbol{z} + \boldsymbol{c} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda} = \boldsymbol{0} \quad \Rightarrow \quad \boldsymbol{z} = -\boldsymbol{P}^{-1}(\boldsymbol{c} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda}).$$

Thus the dual function is

$$\begin{aligned} d(\boldsymbol{\lambda}) &= \frac{1}{2}(\boldsymbol{c} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda})^{\mathrm{T}}\boldsymbol{P}^{-1}(\boldsymbol{c} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda}) - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{P}^{-1}(\boldsymbol{c} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda}) \\ &\quad + \boldsymbol{\lambda}^{\mathrm{T}}(-\boldsymbol{A}\boldsymbol{P}^{-1}(\boldsymbol{c} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda}) - \boldsymbol{q}) \\ &= -\frac{1}{2}(\boldsymbol{c} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda})^{\mathrm{T}}\boldsymbol{P}^{-1}(\boldsymbol{c} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda}) - \boldsymbol{q}^{\mathrm{T}}\boldsymbol{\lambda}, \end{aligned}$$

and the dual optimization program is

$$\begin{aligned} \text{(DQP)} \qquad & \underset{\boldsymbol{\lambda} \in \mathbb{R}^M}{\text{maximize}} \; -\frac{1}{2}(\boldsymbol{c} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda})^{\mathrm{T}}\boldsymbol{P}^{-1}(\boldsymbol{c} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda}) - \boldsymbol{q}^{\mathrm{T}}\boldsymbol{\lambda} \\ & \text{subject to} \qquad \boldsymbol{\lambda} \geq \boldsymbol{0}. \end{aligned}$$

This is also a (concave) quadratic program with linear inequality constraints.

The dual of optimal soft margin

Now let's derive the dual of the optimal soft margin program

$$\begin{aligned} \text{(SM)} \qquad & \underset{b,\boldsymbol{w},\boldsymbol{\xi}}{\text{minimize}} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \frac{C}{n}\sum_{i=1}^n \xi_i \\ & \text{subject to} \qquad y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, n. \\ & \qquad\qquad\qquad\qquad \xi_i \geq 0, \quad i = 1, \ldots, n. \end{aligned}$$

We could put this in the form of a QP and use our result above, but it is just as easy (and more enlightening) to compute the dual directly. We will use two sets of Lagrange multiplies, $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^n$ for the two sets of linear inequality constraints above. The Lagrangian is

$$L = \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \frac{C}{n}\sum_{i=1}^n \xi_i + \sum_{i=1}^n \alpha_i(1 - \xi_i - y_i\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i - y_i b) - \sum_{i=1}^n \beta_i\xi_i.$$

This function is smooth in all of its variables $(b, \boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$; it will be minimized in the primal variables $b, \boldsymbol{w}, \boldsymbol{\xi}$ when[4]

$$\begin{aligned} \nabla_b L &= -\sum_{i=1}^n \alpha_i y_i = 0, \\ \nabla_{\boldsymbol{w}} L &= \boldsymbol{w} - \sum_{i=1}^n \alpha_i y_i \boldsymbol{x}_i = \mathbf{0}, \\ \nabla_{\boldsymbol{\xi}} L &= \frac{C}{n}\mathbf{1} - \boldsymbol{\alpha} - \boldsymbol{\beta} = \mathbf{0}. \end{aligned}$$

The last relation above means we can eliminate the second set of lagrange multipliers by substituting $\beta_i = C/n - \alpha_i$. Plugging these

[4] In this expression, $\mathbf{1}$ is a vector with every entry equal to 1.

relations in the Lagrangian gives us the dual function

$$\begin{aligned} d(\boldsymbol{\alpha}) &= \frac{1}{2}\left\|\sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i\right\|_2^2 + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\alpha_i y_i \left(\sum_{j=1}^{n}\alpha_j y_j \boldsymbol{x}_j\right)^{\mathrm{T}} \boldsymbol{x}_i \\ &= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j + \sum_{i=1}^{n}\alpha_i, \end{aligned}$$

where this simplification implicitly incorporates the constraint that $-\sum_{i=1}^{n}\alpha_i y_i = 0$.

Finally, this gives us the dual program

$$\begin{aligned} \text{(DSM)} \qquad & \underset{\boldsymbol{\alpha}\in\mathbb{R}^n}{\text{maximize}} \quad -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j + \sum_{i=1}^{n}\alpha_i \\ & \text{subject to} \quad \sum_{i=1}^{n}\alpha_i y_i = 0 \\ & \qquad\qquad 0 \le \alpha_i \le C/n, \quad i = 1,\ldots,n. \end{aligned}$$

Support vectors

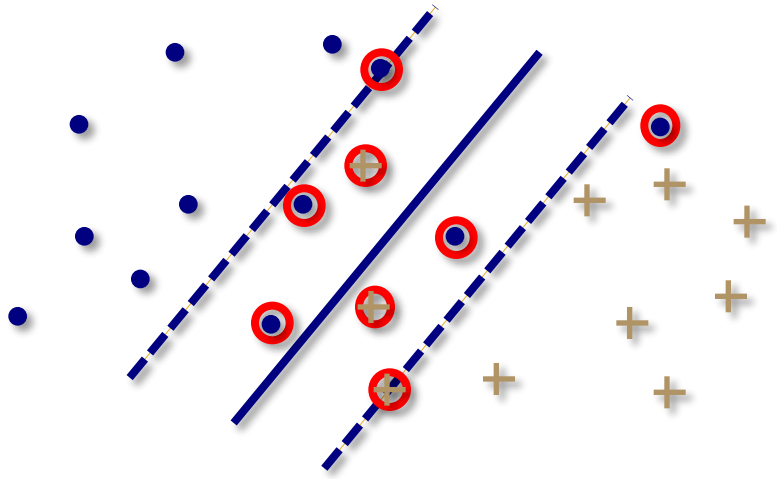
From the complementary slackness conditions ($\lambda_m g_m(\boldsymbol{x}^*) = 0$), we know that

$$\alpha_i^* \left(1 - \xi_i^* - y_i(\boldsymbol{w}^{*\mathrm{T}}\boldsymbol{x}_i + b^*)\right) = 0, \quad i = 1, \ldots, n.$$

Thus[5]

$$\alpha_i^* > 0 \quad \Leftrightarrow \quad \boldsymbol{w}^{*\mathrm{T}}\boldsymbol{x}_i + b^* = y_i(1 - \xi_i^*).$$

This means the i for which $\alpha_i > 0$ correspond to data point $\boldsymbol{x}_i$ that are on or inside the margin of separation.



These are called the **support vectors**, and in practice, there tend to be a small number of them. One consequence for this is that the weights can be calculated from just this small portion of the data. Recall that we derived the following expression:

$$\boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \boldsymbol{x}_i = \sum_{i \in \mathrm{SV}} \alpha_i^* y_i \boldsymbol{x}_i, \quad \text{where SV} = \text{set of support vectors}$$

[5]Note that since $y_i \in \{-1, +1\}$, $y_i^{-1} = y_i$.

If the set of support vectors is small, then computing $\boldsymbol{w}^*$ is can be very inexpensive.

The other thing that the complementary slackness conditions tell us is that

$$\left(\frac{C}{n} - \alpha_i^*\right) \xi_i^* = 0 \quad \text{for all } i = 1, \ldots, n.$$

This means that $\alpha_i^* < C/n \Rightarrow \xi_i^* = 0$. Combining this with the condition above means that

$$0 < \alpha_i^* < C/n \quad \Rightarrow \quad b^* = y_i - \boldsymbol{w}^{*\mathrm{T}} \boldsymbol{x}_i.$$

This gives us the relation that allows us to recover the affine offset from the dual solution.