Beginning of Sept. 22, 2022

Support Vector Machines, Separable Case

Why SVM?

- It is one of the most commonly used classification algorithms.
- It allows us to explore the concept of *margins* in classification.
- It works well with the kernel trick.
- It has strong theoretical guarantees.

We will again focus on binary classification here. The function class for SVMs is a function on a feature map φ applied to the data points, namely $\operatorname{sgn}(w^T\varphi(x) + b)$. The bias term b is taken separately for SVMs, the reason of which will be explained later.

Margins: Geometric Intuition

When data is linearly separable, there are infinitely many hyperplanes with zero training error. Back to the same old question — which one should we choose? We claim that the further away the separating hyperplane is from the data points, the better. To this end, we define the **margin** for linearly separable data to be the distance from the hyperplane the closest point.
Some math first: given x and a hyperplane $\{x : w^Tx + b = 0\}$, we compute the distance as follows:

- Orthogonally project x onto the hyperplane and obtain x'.
- Since w is orthogonal to the hyperplane, x' is of form $x' = x - \beta w/\|w\|_2$.
- Find β using $w^Tx' + b = 0$, namely
$$0 = w^T(x - \beta w/\|w\|_2) + b = w^Tx - b\|w\| + b \implies \beta = \frac{w^Tx + b}{\|w\|_2}.$$
- The distance is then $\|x - x'\|_2 = |\beta| = |w^Tx + b|/\|w\|_2$.
- More generally, if the hyperplane classifies (x, y), then $\operatorname{sgn}(w^Tx + b) = y$, so the distance becomes
$$\frac{y(w^Tx + b)}{\|w\|_2}.$$

Margins: Functional Motivation

Recall from previous lectures that we used the signoid function to show the probability of a data point getting 1 or 0 by
$$\mathbb{P}(y = 1 \mid x, w) = \sigma(y(w^Tx + b)) = \frac{1}{1 + \exp(-y(w^Tx + b))}.$$
Here, if $y = 1$, we want $w^Tx + b \gg 0$ and if $y = -1$, we want $w^Tx + b \ll 0$. Hence we want $y(w^Tx + b) \gg 0$. However, we can easily "cheat" by making w large. To offset this potential effect, we normalize the quantity and instead try to maie $(y(w^Tx + b))/\|w\|$ as large as possible.

Maximizing Margin

The formal definition of a margin distance from all trainig points is

$$\min_i \frac{y_i(w^T\varphi(x_i)+b)}{\|w\|_2} \qquad \text{for data points} \qquad (x_i, y_i).$$

Since we want to maximize the smallest distance among all data points, this translates to solving

$$\max_{w,b}\min_i \frac{y_i(w^T\varphi(x_i)+b)}{\|w\|_2} = \max_{w,b}\frac{1}{\|w\|_2}\min_i y_i(w^T\varphi(x_i)+b).$$

Note that if we rescale (w, b), multiplying both by some scalar, the hyperplane remains the same, i.e.,

$$\{x : w^T\varphi(x)+b=0\} = \{x : kw^T\varphi(x)+kb=0\}.$$

We can pick the appropriate quantity so that $\min_i y_i(w^T\varphi(x_i)+b) = 1$. More concretely, this scalar is $\frac{1}{\min_i y_i(w^T\varphi(x_i)+b)}$. After rescaling, the margin simply becomes $1/\|w\|_2$.

SVM for Separable Data: "Primal" Formulation

From above, on a separable training set, lour goal is to solve

$$\max_{w,b}\frac{1}{\|w\|_2} \qquad \text{subject to} \qquad \min_i y_i(w^T\varphi(x_i)+b) = 1.$$

This is equivalent to solving

$$\min_{w,b}\frac{1}{2}\|w\|_2^2 \qquad \text{subject to} \qquad y_i(w^T\varphi(x_i)+b) \geqslant 1 \text{ for all } i \in [n]. \tag{SVM1}$$

In doing so, we transformed a non-convex objective into a convex one! Because of the new formulation, SVM is also called the **max-margin** classifier. The constraints above are called **hard-margin constraints**.

Support Vector Machines: General Non-Separable Case

If the data are not linearly separable, the previous constraint

$$y_i(w^T\varphi(x_i)+b) \geqslant 1 \qquad \text{for all} \qquad 1 \leqslant i \leqslant n$$

is obviously not feasible.

In fact, even in the separable case, sometimes it is *not* the best idea to completely separate them. Consider, for example, a rectangle, in which all blue dots are in bottom right. Almost all red points are in top left, so an "ideal" classifier would separate the rectangle diagonally. But if we have an extra red point just on top of the blue ones, this forces a horizontal separating line, and clearly this is not what we want. (Though we successfully classify all training data.)

To deal with this issue, we relax the constraints to $\boldsymbol{\ell_1}$ **norm soft-margin** constraints:

$$y_i(w^T\varphi(x_i)+b) \geqslant 1-\xi_i \iff 1-y_i(w^T\varphi(x_i)+b) \leqslant \xi_i, \qquad i \in [n].$$

Here we introduce the **slack variables** $\xi_i \geqslant 0$. (This should ring a bell on hinge loss $\ell_{\text{hinge}}(z) = \max\{0, 1-z\}$ with $z = (w^T\varphi(x)+b)$.)

But Why ℓ_1 Penalization?

Functions like squared hinge loss really penalizes a misclassification as $\max(0, 1 - z)^2$ grows very quickly as z gets increasing negative. Hinge loss itself, however, is much more robust compared to squared loss when facing outliers in data.

Example: A one-dimensional example. Suppose we have $x_1, x_2, ..., x_n$. We know

$$w^*_{\ell_2} = \operatorname*{argmin}_w \sum_{i=1}^n (x_i - w)^2 = \frac{1}{n}\sum_{i=1}^n x_i.$$

On the other hand,

$$w^*_{\ell_1} = \operatorname*{argmin}_w \sum_{i=1}^n |x_i - w| = \text{median of } \{x_1, ..., x_n\}.$$

It is obvious that the median is much more robust to outliers — one significant outlier shifts the mean significantly but has little impact on the median.

Example: For one-dimensional regression. Suppose the data points are nice enough to satisfy $y_i = 10x_i + \epsilon_i$ (ϵ for noise). Consider

$$w^*_{\ell_2} = \operatorname*{argmin}_w \sum_{i=1}^n (y_i - wx_i)^2 \qquad \text{and} \qquad w^*_{\ell_1} = \operatorname*{argmin}_w \sum_{i=1}^n |y_i - wx_i|.$$

What if we add an outlier now? $w^*_{\ell_2}$ will shift much more than $w^*_{\ell_1}$ does.

SVM: General Primal Formulation

Ideally, we want ξ_i to be as small as possible. The objective therefore becomes

$$\min_{w,b,\{x_i\}} \frac{1}{2}\|w\|_2^2 + C\sum_i \xi_i \qquad \text{subject to} \qquad \begin{aligned} &y_i(w^T\varphi(x_i) + b) \geqslant 1 - \xi_i \\ &\xi_i \geqslant 0. \end{aligned} \tag{SVM2}$$

- When $\xi_i = 0$, the data is classified correctly.
- When $\xi_i < 1$, the data is classified correctly but does not satisfy the large margin constraint.
- When $\xi_i > 1$, the data is misclassified.

Primal Formulation: Another View

In a nutshell: SVM can be thought of as a linear model with ℓ_2 regularized hinge loss. We claim (SVM2) is equivalent to finding

$$\min_{w,b,\{\xi_i\}} \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^n \xi_i \qquad \text{subject to} \qquad \max\{0, 1 - y_i(w^T\varphi(x_i) + b)\} = \xi_i \text{ for } i \in [n]. \tag{SVM3}$$

In order to minimize $\sum \xi_i$, we should set ξ_i to be as small as possible. When if $\xi_i = 0$ we want $y_i(w^T\varphi(x_i) + b) \geqslant 1$; when $\xi_i > 0$ we want it to be precisely $1 - y_i(w^T\varphi(x_i) - b))$. But then we can rewrite (SVM3) as

$$\min_{w,b} C\sum_{i=1}^n \max\{0, 1 - y_i(w^T\varphi(x_i) + b)\} + \frac{1}{2}\|w\|_2^2$$

which, after setting $\lambda = 1/C$, gives the form identical to minimizing ℓ_2 regularized hinge loss:

$$\min_{w,b} \sum_{i=1}^{n} \max\{0, 1 - y_i(w^T\varphi(x_i) + b)\} + \frac{\lambda}{2}\|w\|_2^2. \tag{SVM4}$$

Optimizing / Kernelizing SVM

We now go back to (SVM2), the convex objective. We can apply any convex optimization algorithms (e.g. SGD), but usually we apply kernel trick, which requires solving the **dual problem**. It suffices to show that w^*, the solution, is a linear combination of the feature vectors $\varphi(x_i)$.

Proposition

We claim that, for SVM, w^* *is* a linear combination of $\varphi(x_i)$, i.e., $w^* = \sum_{i=1}^{n} \alpha_i y_i \varphi(x_i)$ for some α_i's.

Informal proof. We first formulate the SVM question as a linear model $F(w)$ with ℓ_2 regularized hinge loss as in (SVM4). Since the function is convex, GD is guaranteed to find a minimizer with any initialization (with appropriate learning rate).

Taking derivatives,

$$\frac{\partial F(w)}{\partial w} = \sum_{i=1}^{n} \left(\left.\frac{\partial \ell_{\text{hinge}}(z)}{\partial z}\right|_{z=y_i(w^T\varphi(x_i)+b)} \cdot \overbrace{(-y_i\varphi(x_i))}^{\text{chain rule}} \right) + \lambda w. \tag{Δ}$$

Therefore, we can initialize $w^{(0)}$ as 0 and iteratively define

$$w^{(t+1)} \leftarrow w^{(t)} - \eta(\Delta).$$

Observe that $w^{(t)}$ is always in the span of $y_i\varphi(x_i)$: every time when we update, we are adding a linear combination of the $y_i\varphi(x_i)$'s. Taking the limit we see w^* also lies in the span. $\square$

Having shown $w^* = \sum_i \alpha_i y_i \varphi(x_i)$, in the separable case, minimizing $\|w\|_2^2/2$ becomes

$$\min_{\alpha,t} \frac{1}{2}\Big\| \sum_{i=1}^{n} \alpha_i y_i \varphi(x_i) \Big\|_2^2 \qquad \text{subject to} \qquad y_i(\sum_{i=1}^{n} \alpha_i y_i \varphi(x_i)^T \varphi(x_i) + b) \geqslant 1.$$

This is equivalent to

$$\min_{\alpha_i,b} \frac{1}{2} \sum_{i,j} \alpha_i\alpha_j y_i y_j \varphi(x_i)^T\varphi(x_j) \qquad \text{subject to} \qquad y_i(\sum_{i=1}^{n} \alpha_i y_i \varphi(x_i)^T \varphi(x_j) + b) \geqslant 1.$$

Using **Lagrange duality** (not covered), for the separable case, the objective above is equivalent to

$$\max_{\alpha_i} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i\alpha_j y_i y_j \varphi(x_i)^T\varphi(x_j) \qquad \text{subject to} \qquad \sum_{i=1}^{n} \alpha_i y_i = 0 \text{ and } \alpha_i \geqslant 0.$$

In particular, we no longer need to compute $\varphi(x)$, and the objective is quadratic.

For the general case, the dual is identical except the extra requirement that $0 \leqslant \alpha_i \leqslant C$.

Prediction Using SVM

How do we predict, given the solution $\{\alpha_i^*\}$ to the optimization problem?
Remember

$$w^* = \sum_{i=1}^{n} \alpha_i^* y_i \varphi(x_i) = \sum_{i:\alpha_i^*>0} \alpha_i^* y_i \varphi(x_i).$$

That is, we ignore indices with $\alpha_i = 0$. A point $\varphi(x_i)$ with $\alpha_i^* > 0$ is called a **support vector** hence the name SVM.
To make a prediction on any data point x:

$$\begin{aligned} \operatorname{sgn}(w^{*T}\varphi(x) - b^*) &= \operatorname{sgn}\left(\sum_{\alpha_i^*>0} \alpha_i^* y_i \varphi(x_i)^T \varphi(x) - b^*\right) \\ &= \operatorname{sgn}\left(\sum_{\alpha_i^*>0} \alpha_i^* y_i k(x_i, x) - b^*\right) \end{aligned}$$

with the help of kernels.

Bias Term b^*

It can be shown that, in the separable case, the support vectors lie on the margin.
In this case, for any i with $\alpha_i^* > 0$,

$$y_i^2(w^{*T}\varphi(x_i) + b^*) = y_i \implies w^{*T}\varphi(x_i) + b^* = y_i \implies b^* = y_i - w^{*T}\varphi(x_i).$$

In the general case, for any support vector $\varphi(x_i)$ with $0 < \alpha_i^* < C$, it can be shown that $y_i(w^{*T}\varphi(x_i) + b^*) = 1$ so

$$b^* = y_i - w^{*T}\varphi(x_i) = y_i - \sum_{j=1}^{n} \alpha_j^* y_j k(x_i, x_j).$$

With α^* and b^* known, we can make a prediction on any data point as stated above.

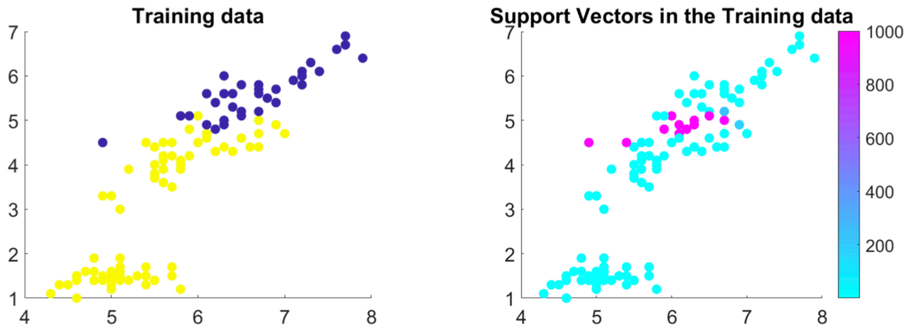
Understanding SVM

Straight from definition, support vectors are $\varphi(x_i)$'s with $\alpha_i^* > 0$. We can show that they are precisely the points satisfying one of the following:

- They lie on the large margin boundary. Consequently $\xi_i^* = 0$, so $y_i(w^{*T}\varphi(x_i)+b^*) = 1$ and the point is precisely $1/\|w^*\|_2$ away from the hyperplane.
- They do not satisfy the large margin constraint. This is when $\xi_i^* < 1$ (still classified correctly).
- They are misclassified, corresponding to $\xi_i^* > 1$,

All other points (i.e., those classified correctly and not on the margin boundary) have $\alpha_i^* = 0$ and are therefore not support vectors.

A potential drawback of the kernel method is that it's non-parametric and needs to sometimes keep track of all training data. SVM, however, usually have $|\{i : \alpha_i^* > 0\}| \ll n$, thereby avoiding this issue.


Training data
Support Vectors in the Training data
1000
800
600
400
200