3.2 SVM is a Convex OP

Machine Learning 1: Foundations

Marius Kloft *(TUK)*

Kaiserslautern, 5–12 May 2020

Making the SVM Convex

Theorem

The linear hard-margin SVM from last week, that is,

$$\max_{\gamma, b \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d} \gamma \quad \text{s.t.} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq \|\mathbf{w}\| \gamma$$

can be equivalently rewritten in convex form as given below:

Linear hard-margin SVM in convex form

$$\begin{aligned} \min_{b \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d} \quad & \frac{1}{2} \|\mathbf{w}\|^2 \\ \text{s.t.} \quad & 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) \leq 0 \quad \forall i = 1, \ldots, n \end{aligned}$$

Core Idea of the Proof

The SVM results in a linear classifier $f(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$, the parameters $(\mathbf{w}, b)$ of which are not unique:

$$\forall \lambda > 0 : \quad \text{sign}(\mathbf{w}^\top \mathbf{x} + b) = \text{sign}\big(\underbrace{\lambda \mathbf{w}}_{=:\mathbf{w}_\lambda}{}^\top \mathbf{x} + \underbrace{\lambda b}_{=:b_\lambda}\big). \quad (\star)$$

Idea: if it helps, we could restrict in the SVM OP our search for $\mathbf{w}$ to $\mathbf{w}$s that have some nice norm, and we would—by $(\star)$—still search the space of all linear classifiers.

For reasons that will become clear below, we choose the restriction $\|\mathbf{w}\| = 1/\gamma$, by setting $\lambda := \frac{1}{\|\mathbf{w}\|\gamma}$.

Proof

$$
\begin{aligned}
& \max_{\gamma, b\in\mathbb{R}, \mathbf{w}\in\mathbb{R}^d} && \gamma && \text{s.t.} && y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq \|\mathbf{w}\|\,\gamma \\
\overset{(a),(b)}{\iff} & \min_{\gamma>0, b\in\mathbb{R}, \mathbf{w}\in\mathbb{R}^d} && \frac{1}{2\gamma^2} && \text{s.t.} && y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq \|\mathbf{w}\|\,\gamma \quad \Big|\cdot \boldsymbol{\lambda} \\
\overset{\forall\lambda>0}{\iff} & \min_{\gamma>0, b\in\mathbb{R}, \mathbf{w}\in\mathbb{R}^d} && \frac{1}{2\gamma^2} && \text{s.t.} && y_i(\boldsymbol{\lambda}\mathbf{w}^\top \mathbf{x}_i + \lambda b) \geq \boldsymbol{\lambda}\|\mathbf{w}\|\,\gamma \\
\overset{(c)}{\iff} & \min_{\gamma>0, b\in\mathbb{R}, \mathbf{w}\in\mathbb{R}^d} && \frac{1}{2\gamma^2} && \text{s.t.} && y_i(\lambda\mathbf{w}^\top \mathbf{x}_i + \boldsymbol{b}) \geq \lambda\|\mathbf{w}\|\,\gamma \\
\overset{\lambda:=\frac{1}{\|\mathbf{w}\|\gamma}}{\iff} & \min_{\gamma>0, b\in\mathbb{R}, \mathbf{w}\in\mathbb{R}^d} && \frac{1}{2\gamma^2} && \text{s.t.} && y_i\left(\frac{\mathbf{w}^\top \mathbf{x}_i}{\|\mathbf{w}\|\,\gamma} + b\right) \geq 1 \\
\overset{(d)}{\iff} & \min_{b\in\mathbb{R}, \tilde{\mathbf{w}}\in\mathbb{R}^d} && \frac{1}{2}\|\tilde{\mathbf{w}}\|^2 && \text{s.t.} && y_i(\tilde{\mathbf{w}}^\top \mathbf{x}_i + b) \geq 1 \\
\iff & \min_{b\in\mathbb{R}, \mathbf{w}\in\mathbb{R}^d} && \frac{1}{2}\|\mathbf{w}\|^2 && \text{s.t.} && 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) \leq 0
\end{aligned}
$$

Proof

In the former derivation we use the following arguments:

(a) Maximizing γ gives the same solution as minimizing $1/2\gamma^2$.

(b) $(\mathbf{w}, b, \gamma) = \mathbf{0}$ satisfies the constraints and has objective 0.

(c) $b \in \mathbb{R}$ is an unconstrained variable.

(d) We substitute $\tilde{\mathbf{w}} := \frac{\mathbf{w}}{\|\mathbf{w}\|\gamma}$, from which it follows:
$\|\tilde{w}\| = \left\|\frac{\mathbf{w}}{\|\mathbf{w}\|\gamma}\right\| = \frac{\|\mathbf{w}\|}{\|\mathbf{w}\|\gamma} = 1/\gamma$.

$\square$

Soft-margin SVM

Similar, we can formulate a soft-margin SVM as convex optimization problem:

Linear soft-margin SVM in convex form

$$\min_{b\in\mathbb{R},\mathbf{w}\in\mathbb{R}^d,\boldsymbol{\xi}\in\mathbb{R}^n} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \quad 1-\xi_i - y_i(\mathbf{w}^\top \mathbf{x}_i + b) \le 0, \quad -\xi_i \le 0 \quad \forall i = 1,\ldots n$$

How to solve?

How to Solve SVM?

We could just put our (convex) SVM OP into one of the many solvers out there for convex optimization problems.

- In Python: library CVXOPT is a standard

Problem: CVXOPT is very powerful, but rather slow
$\Rightarrow$ For big data we need a **faster** solution