Support Vector Machine

Jack David Carson - February 7, 2025

01 Support Vector Machine

We have seen previously the perceptron algorithm and the passive-aggressive algorithm, two **on-line algorithms** (defined by processing each example individually) that fulfill the goal of minimize errors (perceptron) and minimize loss (passive-aggressive). However, the support vector machine attempts to solve the miminal loss solution with all of the data at once. Hence, it is called **off-line algorithm**. Now we will attempt to minimize θ, θ_0 with respect to all the data in S_n at once by evaluating

$$\arg\min_{\theta,\theta_0} \sum_{i=1}^{n} \left[\frac{\lambda}{2} \|x\|^2 + \mathrm{Loss}_h \left(y^{(i)} \left(\theta^\top x^{(i)} + \theta_0 \right) \right) \right]$$

How would someone go about computing this? It turns out that this SVM **objective function** can be reformulated as a quadratic program.

$$\frac{1}{n} \sum_{i=1}^{n} \left[\zeta_i + \frac{\lambda}{2} \|\theta\|^2 \right] \quad \text{subject to} \quad \begin{cases} y^{(i)} \left(\theta^\top x^{(i)} + \theta_{0a} \right) \geq 1 - \zeta_i \\ \zeta_i \geq 1 \end{cases}$$

In practice, this algorithm does not scale well to computation, with a complexity of $\mathcal{O}(n^3)$. In practice we can use a simple **stochastic gradient descent** algorithm, by taking exactly one term i randomly, and applying its gradient onto θ. The **Pegasos algorithm** relies on

$$\begin{aligned} \theta &\leftarrow \theta - \eta \boldsymbol{\nabla}_\theta \left[\mathrm{Loss}_h \left(y^{(i)} \left(\theta^\top x^{(i)} + \theta_0 \right) \right) + \frac{\lambda}{2} \|x\|^2 \right] \\ &= (1 - \lambda \eta) \theta + \eta \begin{cases} y^{(i)} x^{(i)} & \text{for} \quad y^{(i)} \left(\theta^\top x^{(i)} \right) \leq 1 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

The derivation of this is quite simple, and its implementation in a gradient descent loop is *much* more efficient.