Linear Classifier

Jack David Carson - February 5, 2025

(Switching to the notation from the notes because the bold vectors are exhausting to type out)

01 Loss

In the perceptron algorithm notes, we saw a classifer that was reinforced based on a loss function only sensitive to "true" and "false", or 0 and 1. In reality, we will have uncertainty near the decision boundary and want to penalize terms far from it more than terms close to it. We can formalize the old method as

$$\text{Loss}_{0,1}(y(\theta^\mathsf{T}x + \theta_0)) = [[y(\theta^\mathsf{T}x + \theta_0 \leq 0]]$$

We can now introduce a more sensitive kind of loss called the **hinge loss** that increases the penalty for greater violations

$$\text{Loss}_h(y(\theta^\mathsf{T}x + \theta_0)) = \max\{0, 1 - y(\theta^\mathsf{T}x + \theta_0)\}$$

If the prediction is correct, then $1 - y(\theta^\mathsf{T}x + \theta_0) \leq 0$ and the loss is 0. Otherwise, it is a function of the distance to the hyperplane decision boundary.

From the last bit, we reference the norm of orthogonal component of the projection of each point onto the hyperplane as the margin $\gamma = \frac{y_i(\theta^\mathsf{T}x + \theta_0)}{\|\theta\|}$. If we wish to set this margin to $\gamma = 1$, we can say

$$\min_i \frac{y_i(\theta^\mathsf{T}x + \theta_0)}{\|\theta\|} = 1$$

which implies $\min_i y_i(\theta^\mathsf{T}x + \theta_0) = \|\theta\|$ and that **the margin is inversely proportional to $\|\theta\|$**. This intuition leads to the Passive Aggressive (PA) algorithm.

02 Passive-Aggressive Algorithm

We can use this hinge loss to update our parameters based on

$$\theta^{(k+1)} = \underset{\theta}{\text{argmin}} \frac{\lambda}{2}\|\theta - \theta^{(k)}\|^2 + \max\{0, 1 - y(\theta^\mathsf{T}x + \theta_0)\}$$

- $\text{argmin}_\theta \|\theta - \theta^{(k)}\|^2$ penalizes the model for deviating too strongly from the current parameters.
- $\max\{0, 1 - y(\theta^\mathsf{T}x + \theta_0)\}$ is the hinge loss that highly incorrect predictions have a strong signal.
- λ is the **regularization parameter** that determines how much weight is given to preserving the old parameters versus aggressively correcting the mistake.

If we define a **learning rate** parameter

$$\eta := \min\left\{ \frac{\text{Loss}_h(y\theta^\mathsf{T}x)}{\|x\|^2}, \frac{1}{\lambda} \right\}$$

we can express the PA Algorithm in a form which strongly resembles a **perceptron update**

$$\theta^{(k+1)} = \theta^{(k)} + \eta y x$$

Proof: First lets validate that this perceptron update even exists. In an update case we should be able to express it as something along the lines of $\theta^{(k+1)} \leftarrow \theta^{(k)} - \eta \nabla_\theta \mathcal{L}\left(\theta^{(k)}, x\right)$ for some function $\mathcal{L}$ and observation x.

$$-\nabla_\theta \,\text{Loss}_h(y\theta^\mathsf{T}x) = -\nabla_\theta \begin{cases} 1 - y\theta^\mathsf{T}x \\ 0 \end{cases} = \begin{cases} yx & \text{false pred.} \\ 0 & \text{true pred.} \end{cases}$$

which can always be formulated as an update. Now we can plug our update case into the loss function to solve for eta. Starting with the penalty term

$$\frac{\lambda}{2}\|\theta + \eta y x - \theta\|^2 = \frac{\lambda}{2}\|\eta y x\|^2 = \frac{\lambda}{2}\eta^2\|x\|^2$$

since $y = \{-1, 1\}$, $y^2 = 1$, and the term disappears. Next, for the nonzero case of our loss,

$$1 - y(\theta + \eta y x)^\mathsf{T}x = 1 - (y\theta^\mathsf{T}x + \eta\|x\|^2)$$

Before the update then our loss was $l_0 := 1 - y\theta^T x$. Therefore we can say that the loss after update is $l_0 - \eta\|x\|^2$ and *evaluate this on the original criteria.* Defining a piecewise objective function $J(\eta)$

$$J(\eta) = \frac{\lambda}{2}\eta^2\|x\|^2 + \max\{0, l_0 - \eta\|x\|^2\}$$

and evaluate the piecewise condition $l - \eta\|x\|^2 > 0$

$$\eta < \frac{l}{\|x\|^2} \implies J(\eta) = \frac{\lambda}{2}\eta^2\|x\|^2 + l - \eta\|x\|^2 \implies \frac{\mathrm{d}J}{\mathrm{d}\eta} = \lambda\eta\|x\| - \|x\|$$

which, set to 0, implies $\eta = \frac{1}{\lambda}$. Next, the case $l - \eta\|x\|^2 \leq 0$

$$\eta \geq \frac{l}{\|x\|^2} = \frac{1 - y\theta^\mathsf{T}x}{\|x\|^2} = \frac{\text{Loss}_h(y\theta^\mathsf{T}x)}{\|x\|^2}$$

□ **Q.E.D.**

function PASSIVE-AGGRESSIVE($\{(x^{(i)}, y^{(i)}), i = 1, ..., n\}$, λ, T)
    ▷ Initialize the search range
    $\theta \leftarrow \mathbf{0}$ (vector)
    for $t = 1, ..., T$ **do**
        for $i = 1, ..., n$ **do**
            $\eta \leftarrow \min\left\{\frac{\text{Loss}_h(y\theta^\top x)}{\|x\|^2}, \frac{1}{\lambda}\right\}$
            $\theta \leftarrow \theta + \eta y^{(i)} x^{(i)}$
    return θ

03 Support Vector Machine

So far we have 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

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

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