Support Vector Machines (SVMs) and its solution by Stochastic Gradient Descent

Yossi Keshet
March 18, 2014

We consider an input instance $x \in \mathcal{X} = \mathbb{R}^d$ and a target label $y \in \mathcal{Y} = \{-1, +1\}$. They are assumed to be drawn from a fixed but unknown distribution $\rho$. We don’t have access to this distribution, but instead we have a training set of examples $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, which serves as a restricted window to the unknown distribution $\rho$.

We also consider a linear classifier parameterized by a weight vector $w \in \mathbb{R}^d$, such that the parameter setting $w$ determines a mapping of any input $x \in \mathcal{X}$ to an output $\hat{y}_w(x) \in \mathcal{Y}$, defined as follows

$$\hat{y}_w(x) = \text{sign}(w \cdot x).$$

(1)

The goal of the learning problem is to find a setting of the weight vector $w$ of the classifier so as to minimize the error

$$w^* = \arg \min_w \mathbb{P}[\hat{y}_w(x) \neq y].$$

(2)

This probability is with respect to the distribution $\rho$. We can replace probability by expectation over indicator function (Kronecker delta), hence

$$w^* = \arg \min_w \mathbb{E}_{(x,y) \sim \rho}[\mathbb{1}[\hat{y}_w(x) \neq y]],$$

(3)

where $\mathbb{1}[\pi]$ is the indicator function which equals to 1 when the predicate $\pi$ holds and 0 otherwise.

We replace the expectation with the mean over the training set and add regularization to avoid overfitting:

$$w^* = \arg \min_w \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}[\hat{y}_w(x_i) \neq y_i] + \frac{\lambda}{2} \|w\|^2.$$

Since the term $\mathbb{1}[\hat{y}_w(x_i) \neq y_i]$ is a combinatorial quantity and is thus difficult to minimize directly. Instead we use a convex upper bound to it:

$$\mathbb{1}[\hat{y}_w(x) \neq y] = \mathbb{1}[y_i; w \cdot x_i \leq 0] \leq \max\{0, 1 - y_i; w \cdot x_i\} = \ell(w; x_i, y_i).$$

(4)

The function $\ell(w; x_i, y_i)$ is called loss. See Figure 1 for graphical presentation of the loss as an upper bound to the 0-1 error.
We replace $\mathbf{1}[\hat{y}_w(x) \neq y]$ with the loss:

$$
\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i \mathbf{w} \cdot \mathbf{x}_i\} + \frac{\lambda}{2} \|\mathbf{w}\|^2.
$$

(7)

Note that the loss is a convex function of $\mathbf{w}$ since for any two vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ and a scalar $\alpha$ the following holds:

$$
\ell(\alpha \mathbf{w}_1 + (1 - \alpha) \mathbf{w}_2; \mathbf{x}, y) \leq \alpha \ell(\mathbf{w}_1; \mathbf{x}, y) + (1 - \alpha) \ell(\mathbf{w}_2; \mathbf{x}, y).
$$

The regularization $\frac{\lambda}{2} \|\mathbf{w}\|^2$ is also a convex function. This means that the optimization problem in (7) is convex and there exits a single optimum. How do we find this optimum?

**Stochastic Gradient Descent**

We are interested in finding $\mathbf{w}$ which minimizes the following objective

$$
\min_{\mathbf{w}} \sum_{i=1}^{m} f_i(\mathbf{w})
$$

where $f_i$ are functions from vectors $\mathbf{w}$ to scalars. The idea of stochastic gradient descent is to iterate over the functions $f_i$, and update the vector $\mathbf{w}$ after each iteration. The update at each iteration is

$$
\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} f_i(\mathbf{w}),
$$

(8)

where $\nabla_{\mathbf{w}} f_i(\mathbf{w})$ is the gradient of function $f_i(\mathbf{w})$.

Let’s apply that to the SVM optimization problem defined in (7). Here, the functions $f_i$ are of the form $f_i(\mathbf{w}) = \max\{0, 1 - y_i \mathbf{w} \cdot \mathbf{x}_i\}$. The gradient of those functions is not defined, but instead the sub-gradient of the functions is defined as follows:

$$
\nabla_{\mathbf{w}} \left[ \max\{0, 1 - y \mathbf{w} \cdot \mathbf{x}\} \right] = \begin{cases} 
-\mathbf{y} \mathbf{x} & \text{if } 1 - y \mathbf{w} \cdot \mathbf{x} \geq 0 \\
0 & \text{otherwise}
\end{cases}
$$
Hence the (sub-)gradient of (7) for example $i$ is:

$$
\begin{cases}
-yx + \lambda w & \text{if } 1 - y w \cdot x \geq 0 \\
0 + \lambda w & \text{otherwise}
\end{cases}
$$

Plugging this gradient into the update rule in (8), then the stochastic sub-gradient descent algorithm to (7) is as follows:

**Input:** training set $S = \{ (x_1, y_1), \ldots, (x_m, y_m) \}$, parameter $\lambda$, $\eta_0$

**Initialize:** $w_0 = 0$

**For:** $t = 1, \ldots, T$

- choose example: $(x_i, y_i)$ uniformly at random
- set $\eta_t = \eta_0 / \sqrt{t}$
- if $1 - y_i w \cdot x_i \geq 0$
  - $w_t = (1 - \eta_t \lambda) w_{t-1} + \eta_t y_i x_i$
- else
  - $w_t = (1 - \eta_t \lambda) w_{t-1}$

**Output:** $w = \sum_{t=1}^{T} w_t$

Figure 2: Stochastic sub-gradient descent algorithm for SVM

Compare this to the Perceptron algorithm:

**Input:** training set $S = \{ (x_1, y_1), \ldots, (x_m, y_m) \}$

**Initialize:** $w_0 = 0$

**For:** $t = 1, \ldots, T$

- choose example: $(x_i, y_i)$ uniformly at random
- if $y_i w \cdot x_i \leq 0$
  - $w_t = w_{t-1} + y_i x_i$
- else
  - $w_t = w_{t-1}$

**Output:** $w = \sum_{t=1}^{T} w_t$

Figure 3: Perceptron algorithm

The differences are:

1. In the Perceptron algorithm we have no margin, hence the if check for errors ($y_i w \cdot x_i \leq 0$) and not for margin errors ($1 - y_i w \cdot x_i \geq 0$)

2. The Perceptron has no regularization and the reduction of $w$ by $(1 - \eta_t \lambda)$ is not there. It is as if $\lambda = 0$.

3. The learning rate of the Perceptron algorithm is fixed to $1$, $\eta_t = 1$. 

3