Notes on efficient online SVM implementation

J. A. Bagnell

2010-10-06

1 Introduction

The linear online support vector machine is one of the fastest implementation of the algorithm available for large data-sets. It also tends to perform very well in practice. This version was first developed in [2] and related work. It relies directly upon the version of gradient descent due to [1] which changes the learning rate of Zinkevich's [4, 3] online projected gradient algorithm to handle strongly convex objective functions.

2 Algorithm

In short, the online SVM minimizes a sum of hinge loss and an l_2 norm on the weight vector:

$$\text{loss}(w) = \frac{\lambda w^T w}{2} + \frac{1}{T}\sum_{1\ldots T} \textsc{Hinge}(w, x_t, y_t) \tag{1}$$

Computing the gradient of the hinge loss is exactly like that in class. The gradient of the norm on the vector adds a "shrinking" factor at each step. Finally, we can explicitly constrain the solution to lie inside the prior ball using projected gradient descent.

Algorithm 1 Online linear support vector machine.

procedure OnlineSvm(x, y, λ)
  $w \leftarrow 0$
  $t \leftarrow 1$
  while not done **do**
    $\alpha_t = 1/(\lambda t)$
    if $y_t w^T x < 1$ **then** ▷ If Margin Violation
      $w \leftarrow (1 - \alpha_t \lambda) w + \alpha_t y_t x_t$ ▷ Apply Gradient
    else $w \leftarrow (1 - \alpha_t \lambda) w$ ▷ Apply Gradient (just weight shrinkage)

    end if
    if $w^T w > \frac{1}{\lambda}$ **then** ▷ Optionally Project to ball of size $\sqrt{\lambda}$
      $w = \frac{w}{\sqrt{w^T w}\sqrt{\lambda}}$
    end if
    $t \leftarrow t + 1$
  end while
  return w ▷ Optionally return the *average* of all w's computed during process
end procedure

In this version of the algorithm, there is only one parameter: the strength of the prior. We've directly set the margin to 1, as well as the scale of the learning rate. You should normalize the input data to have unit norm on the input as well.

Additional tricks that also help are to make small batches of, say 100 data points, before doing an update and using the averaged gradient. Be sure data-points are randomized if you want to ensure good off-line (batch) performance.

References

[1] Elad Hazan, Adam Kalai, Satyen Kale, and Amit Agarwal. Logarithmic regret algorithms for online convex optimization, 2006. To appear in COLT 2006.

[2] N. D. Ratliff, J. A. Bagnell, and M. Zinkevich. Approximate subgradient methods for online structured prediction. In *AISTAT*, 2007.

[3] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of ICML*, 2004.

[4] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.