COMP9417 Machine Learning and Data Mining

Note: Perceptron Learning with the Pocket Algorithm

Last revision: Thu May 3 2018

Introduction

The "Pocket" Learning Algorithm [Gallant, 1990] can be implemented for different linear classifier-type learning algorithms as a simple extension that may improve performance, e.g., when there is misclassification noise in the data. For Perceptrons, the algorithm can be implemented as a variant of the Perceptron training algorithm. Here the idea is that the set of weights that are the *best so far* are maintained "in the pocket" during Perceptron training, and are returned at the end of learning. The Perceptron training algorithm is on slide 130 of the lecture notes on "Supervised Learning - Classification".

Pocket Algorithm for Perceptron Training

POCKET ALGORITHM

Input: Data D, Learning rate η, Max. iterations m
Output: Vector $\vec{w}_{pocket}$ of "pocket" weights
Initialize: $\vec{w}_{percept} = \vec{w}_{pocket}$ = vector of zeros, or small random values;
$i = r_{percept} = r_{pocket} = p_{percept} = p_{pocket} = 0$.
While $i < m$ Do
  Randomly select a training example $(\vec{x_i}, y_i) \in D$
  If $y_i \; \vec{w}_{percept} \cdot \vec{x_i} > 0$ Then // example i correctly classified
    $r_{percept} = r_{percept} + 1$
    If $r_{percept} > r_{pocket}$ Then // perceptron correct for longer than pocket
      Let $p_{percept}$ be the number of examples in D that are
        correctly classified by the perceptron using weights $\vec{w}_{percept}$
      If $p_{percept} > p_{pocket}$ Then // perceptron more accurate than pocket
        $\vec{w}_{pocket} = \vec{w}_{percept}$
        $r_{pocket} = r_{percept}$
        $p_{pocket} = p_{percept}$
        If $p_{pocket} = |D|$ Then // all examples correctly classified
          Exit
  Else // example i *incorrectly* classified
    $\vec{w}_{percept} = \vec{w}_{percept} + \eta \; y_i \; \vec{x_i}$
    $r_{percept} = 0$

Notes

We assume for all examples $(\vec{x}_d, y_d) \in D$ that $\vec{x}_d$ is a vector $< x_{d,0}, x_{d,1}, \ldots, x_{d,n} >$ where the

value of $x_{d,0} = 1$. Both the Perceptron $\vec{w}_{percept}$ and "pocket" $\vec{w}_{pocket}$ weights are $n + 1$-dimensional vectors, the first component of which is the "bias" weight.

The variables $r_{percept}$ and r_{pocket} count the number of consecutive correct classifications for the perceptron's and the pocket's current weights, respectively, whereas $p_{percept}$ and p_{pocket} measure the accuracy of each set of weights by the number of examples in the training set that are classified correctly. These variables can be used to track the algorithm's performance during learning.

The above algorithm gives a general version of the approach, but the implementation details of this algorithm could be changed, for example, because you do not want to check the current weights on all examples with the pocket after each update – this is a design decision that should be guided by experimentation.

Another design decision is how to set m, the maximum number of iterations. Clearly this should be high enough to let the perceptron learn if the dataset is linearly separable, but in general this will not be known.

Acknowledgement: Material based on [Gallant, 1990].

References

[Gallant, 1990] Gallant, S. (1990). Perceptron-Based Learning Algorithms. *IEEE Transactions on Neural Networks*, 1(2):179–191.