Advanced Machine Learning
Online learning

Master MLDM

Assignment: Due date December 9 2019, 22:00 on claroline
The work can be done by groups of 2 students

Note: all the material is available on the Claroline page!

Objectives of this practical session:

- Program an online learning method and compare it to the result of libSVM on UCI datasets.
- Study an active learning approach.

You should write a report explaining your work and your results with relevant discussions. For the parts that need an implementation, you should provide the code.

1 Online Passive-Aggressive Algorithms

Passive Agressive Online algorithm is a special case of online algorithms for binary classification that can be seen as a kind of extension of SVM to the context of online learning. The principle followed here is the following: For each iteration update the classifier in order to remain as close as possible to the current one while achieving at least a unit margin on the most recent example. However, forcing a unit margin might turn out to be too aggressive in the presence of noise. In this last context, 2 variations can be considered casting a tradeoff between the desired margin and the proximity to the current classifiers. The pseudo-code is provided in Algorithm 1.

begin
  Initialize $\boldsymbol{w}_1 \leftarrow (0, \ldots, 0)$;
  for $t = 1, 2, \ldots$ **do**
    receive instance: $\boldsymbol{x}_t \in \mathbb{R}^n$;
    predict $\hat{y}_t = \text{sign}(\boldsymbol{w}_t \cdot \boldsymbol{x}_t)$;
    receive correct label: $y_t \in \{-1, 1\}$;
    compute loss $l_t = \max\{0, 1 - y_t(\boldsymbol{w}_t \cdot \boldsymbol{x}_t)\}$;
    compute τ_t (cf text in the document);
    compute update: $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \tau_t y_t \boldsymbol{x}_t$;

Algorithm 1: Passive Aggressive Online algorithm. the 3 possible updates for τ are described in the text of the document

Three possible updates can be considered for the τ value considered in Algorithm 1:

1. $\tau_t = \frac{l_t}{\|x_t\|^2}$ - the classic update,
2. $\tau_t = \min\{C, \frac{l_t}{\|x_t\|^2}\}$ - a first relaxation,
3. $\tau_t = \frac{l_t}{\|x_t\|^2 + \frac{1}{2C}}$ - a second relaxation.

Work to do:

1. Implement the 3 versions of the algorithm in the context of linear binary classification.

2. Apply this algorithm on binary classification datasets available from the LibSVM software page and to a SVM implementation (from LibSVM or Scikit-learn).
   For the comparison, you should consider a learning set that are used by both algorithms for learning and test set for evaluation (think of a relevant setup). You can consider both running time and accuracy as comparison measures.
   http://www.csie.ntu.edu.tw/~cjlin/libsvm/

3. Try to introduce some noise (by flipping randomly some labels) and see the influence of the different update strategies.

4. Present your results in a report where you should first define clearly the experimental setup and discuss the results obtained.

5. Active Learning: We consider now an active learning version of the algorithm presented in Algorithm 2. It represents a situation where acquiring labels are costly and the objective is to minimize the number of labels to ask, thanks to a probabilistic criterion. The probabilistic criterion used here corresponds to a random draw over a Bernoulli variable taking into account the confidence of the classification of an example in its distribution parameter. Try to study the behavior of the algorithm with respect to the number of labels acquired. You can use different values of the δ parameter.

begin
  Initialize $\boldsymbol{w}_1 \leftarrow (0, \ldots, 0)$;
  for $t = 1, 2, \ldots$ **do**
    receive instance: $\boldsymbol{x}_t \in \mathbb{R}^n$;
    let $\hat{p}_t = (\boldsymbol{w}_t \cdot \boldsymbol{x}_t)$;
    predict $\hat{y}_t = \text{sign}(\hat{p}_t)$;
    Draw a Bernoulli random variable $Z_t \in \{0, 1\}$ of parameter $\delta/(\delta + |\hat{p}_t|)$;
    if $Z_t = 1$ **then**
      query label $y_t \in \{-1, 1\}$;
      compute loss $l_t = \max\{0, 1 - y_t(\boldsymbol{w}_t \cdot \boldsymbol{x}_t)\}$;
      compute τ_t (cf text in the document);
      compute update: $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \tau_t y_t \boldsymbol{x}_t$;
    else
      $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t$;

Algorithm 2: Online Passive Aggressive Active Learning algorithm. δ is a smoothing parameter greater or equal than 1.

Bonus question - if you have time Non-linearity: Try to implement a kernel-based version of the Passive-Aggressive Online Learning methods by considering that any classifier can be defined as a weighted sum of seen examples. For this last case, you can take a kernel and consider a fixed set of learning examples that will correspond to the considered landmarks on which the decision is taken (*i.e.* they are kind of support vectors. An update of this landmark set (*i.e.* removing or addingg examples) is out of the scope of this work but you could try to think about it.