Machine learning in NLP
The averaged perceptron

UNIVERSITY OF GOTHENBURG

Språk-
BANKEN

Richard Johansson

September 29, 2014

your project

- please select a project within the next couple of weeks
- see web page for ideas

today

- a simple modification of the perceptron algorithm
- often gives quite nice improvements in practice
- implementing it is an optional task in assignment 3

multiclass/structured perceptron pseudocode

$\boldsymbol{w} = (0, \ldots, 0)$
repeat N times
  for (x_i, y_i) in $\mathcal{T}$
    $g = \arg\max_y \boldsymbol{w} \cdot \boldsymbol{f}(x_i, y)$
    if g is not equal to y_i
      $\boldsymbol{w} = \boldsymbol{w} + \boldsymbol{f}(x_i, y_i) - \boldsymbol{f}(x_i, g)$
return $\boldsymbol{w}$

a problem with the perceptron?

- we return the most recent version of the weight vector
- intuitively, this version is over-adapted to the last few instances, and may work less well for other instances

intuition: combining classifiers by voting or averaging

- let's assume we have a lot of classifiers
- each of them has its own strengths and weaknesses
- could they somehow work together?
  - **voting**: return the output favored by most of the classifiers
  - **averaging**: compute the prediction scores for all classifiers; return the output selected by considering the average of all the scores

using averaging to handle the overfitting problem

- in the perceptron, each version of the weight vector can be seen as a separate classifier
  - so we have $N \cdot |\mathcal{T}|$ classifiers
- each of them is over-adapted to the last examples it saw
- but if we compute their average, then maybe we get something that works better overall?
- **averaged perceptron**: return the average of all versions of the weight vector

averaged perceptron pseudocode (naive)

$\boldsymbol{w}_0 = (0, \ldots, 0)$
$t = 0$
repeat N times
  for (x_i, y_i) in $\mathcal{T}$
    $g = \arg\max_y \boldsymbol{w}_t \cdot \boldsymbol{f}(x_i, y)$
    if g is not equal to y_i
      $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \boldsymbol{f}(x_i, y_i) - \boldsymbol{f}(x_i, g)$
    else
      $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t$
    $t = t + 1$

return $\frac{\boldsymbol{w}_1 + \ldots + \boldsymbol{w}_{N \cdot |\mathcal{T}|}}{N \cdot |\mathcal{T}|}$

this is too impractical!

- it's a waste of memory to remember all the versions of $\boldsymbol{w}$ that we have used during training
- can we do something smarter?

an observation

- the weight vector $\mathbf{w}_3$ is the sum of all updates so far:

$$\begin{aligned}
\mathbf{w}_0 &= (0, \ldots, 0) \\
\mathbf{w}_1 &= \mathbf{w}_0 + \Delta_1 = \Delta_1 \\
\mathbf{w}_2 &= \mathbf{w}_1 + \Delta_2 = \Delta_1 + \Delta_2 \\
\mathbf{w}_3 &= \mathbf{w}_2 + \Delta_3 = \Delta_1 + \Delta_2 + \Delta_3
\end{aligned}$$

- the average of three vectors can be written:

$$\begin{aligned}
\frac{\mathbf{w}_1 + \mathbf{w}_2 + \mathbf{w}_3}{3} &= \frac{\Delta_1}{3} + \frac{\Delta_1 + \Delta_2}{3} + \frac{\Delta_1 + \Delta_2 + \Delta_3}{3} \\
&= \frac{3}{3}\Delta_1 + \frac{2}{3}\Delta_2 + \frac{1}{3}\Delta_3
\end{aligned}$$

better averaged perceptron

$\boldsymbol{w} = (0, \ldots, 0)$
$\boldsymbol{a} = (0, \ldots, 0)$
$step = N \cdot |\mathcal{T}|$
repeat N times
  for (x_i, y_i) in $\mathcal{T}$
    $g = \arg\max_y \boldsymbol{w} \cdot \boldsymbol{f}(x_i, y)$
    if g is not equal to y_i
      $\boldsymbol{w} = \boldsymbol{w} + \boldsymbol{f}(x_i, y_i) - \boldsymbol{f}(x_i, g)$
      $\boldsymbol{a} = \boldsymbol{a} + \frac{step}{N \cdot |\mathcal{T}|}(\boldsymbol{f}(x_i, y_i) - \boldsymbol{f}(x_i, g))$
    $step = step - 1$
return $\boldsymbol{a}$