UNIVERSITAT POLITÈCNICA DE CATALUNYA

FACULTAT D'INFORMÀTICA DE BARCELONA

Extension Report: Stop Condition

Analysis of the SVM-RFE algorithm for feature selection

Author:
Robert PLANAS

Director:
Luis A. BELANCHE

Bachelor Degree in Informatics Engineering
Specialization: Computing

May 12, 2021

Contents

1 Stop Condition **2**
1.1 Description and reasoning 2
1.2 Pseudocode formalization 3
1.3 Results 4

Chapter 1

Stop Condition

This modification intends to find the optimal number of features that still performs reasonably well in terms of accuracy (i.e. stop condition) in a non-expensive manner.

1.1 Description and reasoning

For SVM-RFE usually two stop conditions are considered, one is to calculate all feature subsets until a single feature remains. The second is to use a parameter to specify the number of desired features.

Using all feature subsets

Calculating all feature subsets and then trying to find the best one is expensive. However, this process allows us to define a trade-off and find all Pareto optimal solutions by the method of scalarization as discussed in section [m4.1.3]. This process also allows us to plot the hyperplane (in our case a red line) indicating all points that, if feasible, would also be Pareto optimal.

Using a feature subset size parameter

Using a desired feature subset size parameter is usually inconvenient. The best situation would be if that size is a requirement, but often we rather have a preference (e.g. prioritize accuracy). It is impossible to know which is the optimal size without running the algorithm, but we need to know that before running it. This is similar to the chicken and the egg problem[1].

This kind of problems can usually be solved by approximation, i.e. we first run a simplified, computationally inexpensive, version of SVM-RFE (e.g. resampling or increasing the step) and we deduce the optimal size form that. Once the optimal size is known (approximately), we can run the full SVM-RFE version.

Note that implementing this stop condition is trivial, we only need to modify the exit condition in line 4 of [Algorithm 1] such that we exit if $|\vec{s}| > p$ with p being the subset size parameter.

Alternative

Here we propose a more efficient alternative. As discussed in section [m4.4.1] the ranking criteria is an indicative of the performance of the classification when removing some feature f_i, in other words, the feature importance. Thus, we can determine

[1] If something is a chicken and egg problem, it is impossible to deal with a problem because the solution is also the cause of the problem.

a stop condition such that when important features are selected for being removed (their ranking criteria value is big) we stop iterating.

The actual limit can be indicated with a parameter, a threshold, this one being a subjective question similar to that of the trade-off between accuracy and feature subset size.

It would also be interesting to investigate whether we can actually estimate the final accuracy using the ranking criteria and from there perform linear scalarization in the same way that we did when using all feature subsets. We could then compare if both methods give similar results.

1.2 Pseudocode formalization

Definitions:

- $X_0 = [\vec{x_0}, \vec{x_1}, \ldots, \vec{x_k}]^T$ list of observations.
- $\vec{y} = [y_1, y_2, \ldots, y_k]^T$ list of labels.

Algorithm 1: SVM-RFE with Stop Condition

Input: t, t_0 // t = step, t_0 = threshold, $0 \leq t_0$
Output: $\vec{r}$
Data: $X_0, \vec{y}$
1 $\vec{s} = [1, 2, \ldots, n]$ // subset of surviving features
2 $\vec{r} = []$ // feature ranked list
3 $q = 0$ // stop condition
4 **while** $|\vec{s}| > 0 \wedge t_0 > q$ **do**
/* Restrict training examples to good feature indices */
5 $X = X_0(:, \vec{s})$
/* Train the classifier */
6 $\vec{\alpha} =$ SVM-train(X, y)
/* Compute the weight vector of dimension length $|\vec{s}|$ */
7 $\vec{w} = \sum_k \vec{\alpha_k} \vec{y_k} \vec{x_k}$
/* Compute the ranking criteria */
8 $\vec{c} = [(w_i)^2$ for all $i]$
/* Find the t features with the smallest ranking criterion */
9 $\vec{f} =$ argsort$(\vec{c})(:t)$
/* Iterate over the feature subset */
10 **for** $f_i \in \vec{f}$ **do**
/* Store last weight eliminated */
11 $q = \vec{c}(f_i)$
/* Update the feature ranking list */
12 $\vec{r} = [\vec{s}(f_i), ...\vec{r}]$
/* Eliminate the feature selected */
13 $\vec{s} = [...\vec{s}(1 : f_i - 1), ...\vec{s}(f_i + 1 : |\vec{s}|)]$
14 **end**
15 **end**

1.3 Results