UNIVERSITAT POLITÈCNICA DE CATALUNYA

FACULTAT D'INFORMÀTICA DE BARCELONA

Extension Report: Dynamic Step

Analysis of the SVM-RFE algorithm for feature selection

Author:
Robert PLANAS

Director:
Luis A. BELANCHE

Bachelor Degree in Informatics Engineering
Specialization: Computing

April 12, 2021

Contents

Chapter 1

Dynamic Step

This modification is based on the constant step variant of the SVM-RFE algorithm, however, instead of using some constant number as the step in each iteration we calculate that number dynamically. The most straightforward way to do this is by using a percentage. Another possibility is to let the model score influence this percentage.

1.1 Description and reasoning

The percentage is a hyperparameter. It is used within every iteration to select a number of the first ranked features. A constant step has already been used in practice, but it is expected that this method will be significantly faster without effecting the accuracy performance, or even improving it.

The reasoning behind this is that when you have many features for which you only want to select a very small subset, the required amount of iterations will be of lineal complexity on the amount of features, however, by using a percentage, we get a logarithmic complexity.

Other similar modifications are also found in the literature, including using the square root of the remaining features `SQRT-RFE`, an entropy based function `E-RFE`, or `RFE-Annealing` which sets the step at $|\vec{s}|\frac{1}{i+1}$, thus, changing the percentage each iteration (Ding and Wilkins, 2006).

We assume that, the bigger the step in each iteration, the worse the performance of the final selection. However, since we're selecting the worst variables first, selecting more of them at once shouldn't affect performance because they would have been selected anyway with height probability in the following iterations. The fewer the iterations remaining, the riskier it becomes selecting multiple variables at once, and thus a smaller step is beneficial.

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 DynamicStep

Input: p // p = percentage, $0 \leq p \leq 1$
Output: $\vec{r}$
Data: $X_0, \vec{y}$

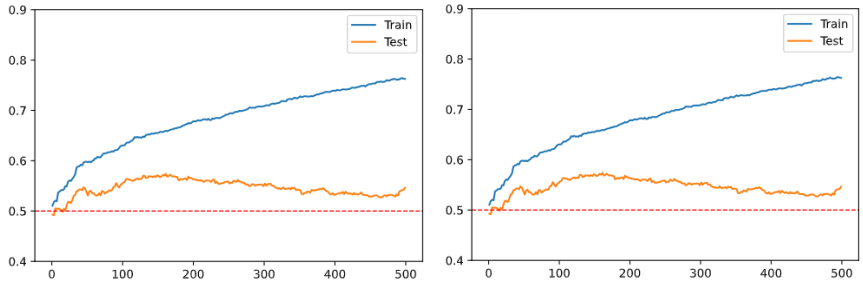
$\vec{s} = [1, 2, \ldots, n]$ // subset of surviving features
$\vec{r} = []$ // feature ranked list
while $|\vec{s}| > 0$ **do**
  /* Restrict training examples to good feature indices */
  $X = X_0(:, \vec{s})$
  /* Train the classifier */
  $\vec{\alpha} = \texttt{SVM-train}(X, y)$
  /* Compute the weight vector of dimension length $|\vec{s}|$ */
  $\vec{w} = \sum_k \vec{\alpha}_k \vec{y}_k \vec{x}_k$
  /* Compute the ranking criteria */
  $\vec{c} = [(w_i)^2 \text{ for all } i]$
  /* Compute t based on the percentage */
  $t = p|\vec{s}|$
  /* Find the t features with the smallest ranking criterion */
  $\vec{f} = \texttt{argsort}(\vec{c})(:t)$
  /* Update the feature ranking list */
  $\vec{r} = [\vec{s}(\vec{f}), ...\vec{r}]$
  /* Eliminate the features with the t smallest ranking criterion */
  $\vec{s} = [[...\vec{s}(1 : f_i - 1), ...\vec{s}(f_i + 1 : |\vec{s}|)] \text{ for all } i]$
end

1.3 Results

1.3.1 Initial Analysis with Madelon

We've initially run some tests to find how hard it is for the SVM-RFE algorithm to work with it.

The first test consists on simply doing a random feature selection and plotting the accuracy for the training and test splits (Figure 1.1). This random selection algorithm will allow us to make comparisons with other feature selection algorithms to determine whether they work or not. Because we're using the average of many random selection experiments, the curve it produces should be always worse than that of any other working selection algorithm. It is expected that it will produce a somewhat linear function with accuracy dropping consistently when the amount of features selected is reduced.



$C = 1$ $C = 10^3$

FIGURE 1.1: Accuracy of an SVM classifier with a random feature selection with the Madelon dataset.

The results of this test are a bit unexpected. Although the training accuracy behaves as expected, the test accuracy is way too low for even the case where all features are used (0.55). Reducing the amount of features, even randomly, produces a slight increase in the test accuracy. This is a fine example of the effects of the curse of dimensionality, on how even a random selection, which should produce consistently worse models the less amount of features it has, ends up producing better ones simply because there are fewer features with independence on how the feature selection is made.

We've performed a second test to see if the regularization parameter has any effect and can be of any help in trying to reduce overfitting (Figure 1.2).

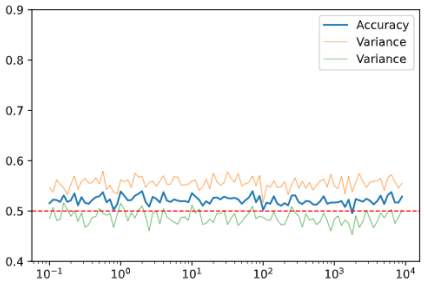


FIGURE 1.2: Test accuracy mean of SVM classifier with 10-fold cross-validation using all features for multiple values of C.

As the plot indicates, no value of the regularization parameter helps improve the performance. These poor results suggest that the dataset is not linearly separable and thus is producing huge amounts of overfitting in all cases, but, how does this affect SVM-RFE?

To see how SVM-RFE performs under these conditions we've designed an experiment very similar to that used in random selection, although instead of using 10 random selections we've used cross-validation to take the mean. In this experiment we also use a constant step of 10 for performance reasons, see Figure 1.3.

Comparing this plot with that of the random selection we can see that the results are quite disappointing. Although SVM-RFE does improve the performance on the training data, the test performance remains the same. With this we can conclude that SVM-RFE using a lineal kernel does not work well on this specific dataset, and thus, no matter the step strategy used, the results will all be quite bad.

We have also checked how does SVM-RFE perform if we use a dynamic step strategy (Figure 1.4), but no significant difference is appreciated.

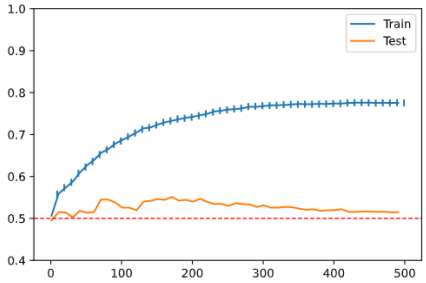
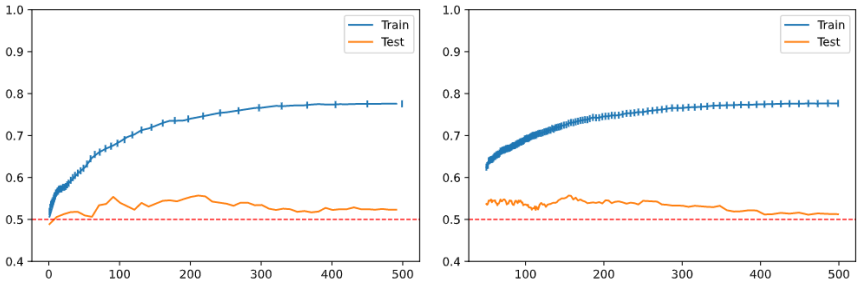


FIGURE 1.3: Mean accuracy using 20-fold CV of an SVM classifier with SVM-RFE selection of the Madelon dataset.



10%, 52 iterations, stop at 0 3%, 136 iterations, stop at 50

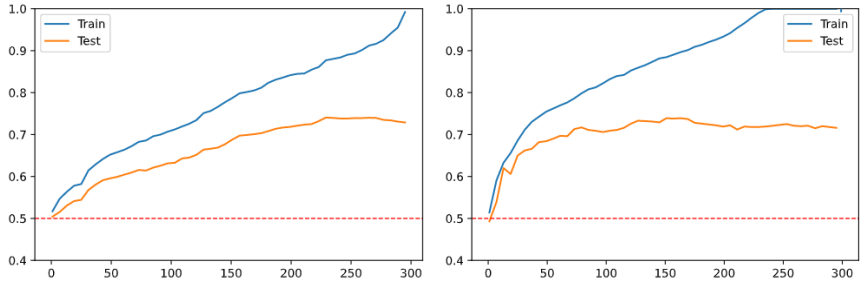
FIGURE 1.4: Mean accuracy using 20-fold CV of an SVM classifier with SVM-RFE selection using dynamic step of the Madelon dataset.

This is a problem. We've initially wanted to use the Madelon dataset to do our experiments, but these results show that the dataset in not adequate for a linear classifier. Thus, we've decided to use simpler artificially generated datasets for this experiment and come back to the Madelon dataset later on with other extensions of the SVM-RFE algorithm.

1.3.2 Initial Analysis with artificially generated data

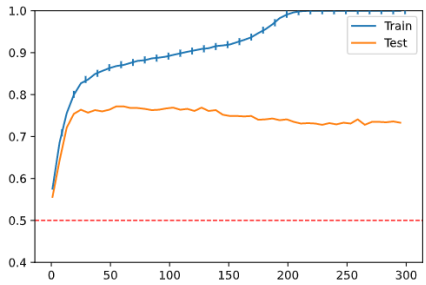
To generate the dataset we've used the *make_classification* function of the `Sklearn` library. We've created a dataset with 300 features, 50 of which are informative. Ideally, a perfect feature selection algorithm should be able to produce a peak test accuracy at the exact amount of informative features or less.

Similar to the experiments descried before, we've performed random selection and SVM-RFE. Here tough, we also make a comparison with different values of the step to see how it effects the actual performance of the selection. In the particular case where the step is equal to the amount of features this is no longer SVM-RFE but a filter method based on SVM (one single iteration).

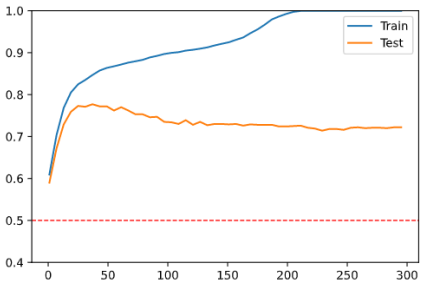


A. Random selection

B. SVM-based filter method, step = 300



C. SVM-RFE, step = 10



D. SVM-RFE, step = 1

FIGURE 1.5: Mean accuracy using 20-fold CV of an SVM classifier with different selection methods.

From this analysis we conclude that SVM-RFE works well on this dataset. We also verify, by comparing plots **B** and **C**, our previous assumption that the more iterations (smaller step) the better the accuracy. This relationship, however, doesn't seem to be lineal, as the same effect can not be appreciated when comparing plots **C** and **D**.

Notice that when evaluating the performance of a feature selection algorithm two variables must be considered, one is accuracy performance and the other is amount of features. A selection with a great accuracy but a lot of features may not be as good as a section with a decent accuracy and only a handful of features. That is, we want to maximize accuracy while minimizing amount of features. These two variables do not operate on the same range, and it can even be the case that the importance of one over the other is problem-dependent. This makes comparing multiple instances hard, this is why are comparing pairs of plots instead of simply a pair of values.

1.3.3 Dynamic Step vs Constant Step

Based on our assumptions, dynamic step can never beat in accuracy performance a constant step of one. Beating constant steps bigger than 1 should theoretically be possible when the amount of features is huge, since at some point the specific step used will be smaller than the constant, and there an improvement may be found. For our specific dataset of 300 features however, no such improvement was found, likely because the amount of features was insufficient, but also because of the no linearity of the stated property.

Rather than for improving accuracy, dynamic step is more useful for improving computational cost. Because each iteration reduces the amount of features, training the SVM will be faster with dynamic step even if the same amount of iterations are used, since there will be more iterations using fewer features in the dynamic step case.

This same mechanic does not apply only to using percentages, the other methods stated at section 1.1 follow the same logic. In Figure 1.6 you can see a comparison of dynamic step methods. Although all of them draw some kind of curve, they don't use the same shape, and thus can produce slightly different performance results using the same amount of iterations. The time cost of the methods can be estimated as the area under the curve.

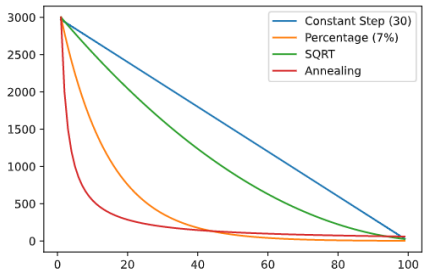


FIGURE 1.6: Amount of features remaining (vertical axis) at each iteration (horizontal axis).

Experimentally we confirm our results and see that using a percentage is faster compared to a constant step, without reducing its accuracy. The following table summarizes the time cost for various experiments. The SVM-RFE time shown is the mean of the times of every fold used during cross-validation.

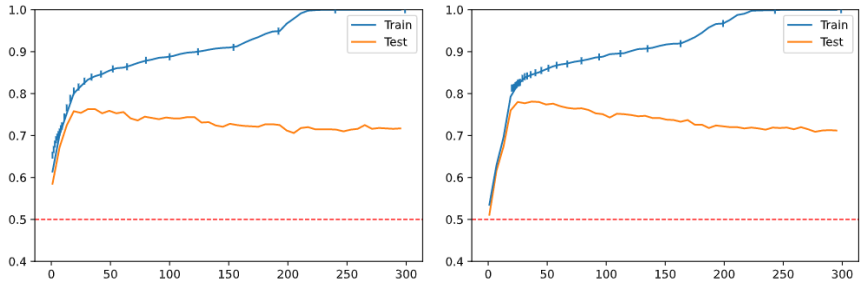
Strategy	**Iterations**	**Time**	**Total Time**
Constant	60	29.16s	00:02:36
Constant	30	15.90s	00:02:00
Constant	6	4.43s	00:01:29
Constant	3	2.74s	00:01:22
Percentage	47	7.03s	00:01:29
Percentage	26	3.83s	00:01:17
Percentage	10	2.59s	00:01:13

TABLE 1.1: Time costs for various experiments.

Notice that although one could think that the smaller the area under the curve the better, and it is true when it comes to time costs, a very pronounced curve would imply a very big initial step, this can result in a drop in accuracy performance in the very first iterations that can not be restored later on even with a step of 1. This is specially important for the annealing method, since it chooses a 50% percentage in the very first iteration. How much acute the curve can be without producing a drop in accuracy depends on the dataset, specifically on the amount of informative features. Thus, introducing a hyperparameter to specify the sharpness of the curve may be useful. The percentage approx already has this hyperparameter by default, but introducing such parameter in the other methods is also trivial. Another option would be to further customize the curve by introducing minimum and maximum steps.

Finally, one last parameter that may be of relevance is the stop value. If the amount of informative variables is known, it doesn't make sense to perform many iterations on selections so small that the performance has already dropped, instead

it may make more sense to increase the amount of iterations the closer you get to the amount of informative variables. This effect can be seen in Figure 1.7.



2%, 26 iterations, stop at 0 2%, 25 iterations, stop at 20

FIGURE 1.7: Comparison of the dynamic step behavior with and without using a stop parameter.

1.3.4 Annotations

- The Cross Validation procedure has been implemented by hand and parallelized.
- How do I know which method is better?
- How do I measure how good some method is (accuracy vs selection size)? Mathematically.
- Should I do hyperparameter search (of C) also here?
- Explain why plotting takes so much time.
- Should the percentage really be a hyperparameter? Annealing and Square Root do not, and it doesn't make much of a difference anyways?
- The complexity of SVM seems to be $O(\max(n,d)\min(n,d)^2)$.
  If dimensions > nº observations: $O(dn^2)$.
  Otherwise: $O(nd^2)$.

Bibliography

Ding, Yuanyuan and Dawn Wilkins (Sept. 2006). "Improving the Performance of SVM-RFE to Select Genes in Microarray Data". en. In: *BMC Bioinformatics* 7.2, S12. ISSN: 1471-2105. URL: https://doi.org/10.1186/1471-2105-7-S2-S12 (visited on 03/28/2021).