UNIVERSITAT POLITÈCNICA DE CATALUNYA

FACULTAT D'INFORMÀTICA DE BARCELONA

Extension Report: Kernels

Analysis of the SVM-RFE algorithm for feature selection

Author:
Robert PLANAS

Director:
Luis A. BELANCHE

Bachelor Degree in Informatics Engineering
Specialization: Computing

May 18, 2021

Contents

Chapter 1

Non-linear Kernels

This modification intends to apply the required modifications in the calculation of the ranking criteria so that non-linear kernels can be used in the SVM.

1.1 Description and reasoning

When a problem is not linearly separable, we know that a hard-margin SVM will not be able to correctly place a decision boundary. In this case a soft-margin SVM may be used, but it only works to some extent and if the underlying distribution is near linearly separable. If it is not the case, much better results can be achieved by using non-linear kernels.

To use this method within SVM-RFE we must first be able to compute the ranking coefficient from a non-linear kernel. In contrast with the linear kernel case, where the ranking coefficient can be simplified to $(w_i)^2$, for non-linear kernels, however, since it is a more general case, no simplification can be performed. Instead, we use the general ranking coefficient for SVM (Equation 1), which we restate here:

$$DJ(i) = (1/2)(\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{H}\boldsymbol{\alpha} - \boldsymbol{\alpha}^{\mathrm{T}}\mathbf{H}(-i)\boldsymbol{\alpha})$$

Note that the *hessian* matrix $\mathbf{H_{i,j}} = y_i y_j k(\mathbf{x_i}, \mathbf{x_j})$ needs be computed each iteration (since the dimension of $\mathbf{x_i}$ and $\mathbf{x_j}$ will change), and also for each feature removed in each iteration. This is slow. However, various optimizations may exist, as discussed in Section [ref.].

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, k // t = step, k = kernel function
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 */
  $X = X_0(:, \vec{s})$
  /* Precompute hessian matrix */
  $\mathbf{H_{i,j}} = y_i y_j k(\mathbf{x_i}, \mathbf{x_j}) \quad \text{for all} \quad \mathbf{x_i}, \mathbf{x_j} \in X$
  /* Train the classifier */
  $\vec{\alpha} = \texttt{SVM-train}(X, y, k)$
  /* Compute the ranking criteria */
  $\vec{c} = [c_1, c_2, \ldots, c_{|\vec{s}|}]$
  for $c_l \in \vec{c}$ **do**
    /* Compute new hessian with the feature l removed */
    $\mathbf{H_{i,j}}(-l) = y_i y_j k(\mathbf{x_i}, \mathbf{x_j}) \quad \text{for all} \quad \mathbf{x_i}, \mathbf{x_j} \in X(-l)$
    /* Calculate ranking coefficient */
    $c_l = (1/2)(\boldsymbol{\alpha}^{\mathrm{T}} \mathbf{H} \boldsymbol{\alpha} - \boldsymbol{\alpha}^{\mathrm{T}} \mathbf{H}(-l) \boldsymbol{\alpha})$
  end
  /* Find the t features with the smallest ranking criterion */
  $\vec{f} = \texttt{argsort}(\vec{c})(: t)$
  /* Iterate over the feature subset */
  for $f_i \in \vec{f}$ **do**
    /* Update the feature ranking list */
    $\vec{r} = [\vec{s}(f_i), \ldots\vec{r}]$
    /* Eliminate the feature selected */
    $\vec{s} = [\ldots\vec{s}(1 : f_i - 1), \ldots\vec{s}(f_i + 1 : |\vec{s}|)]$
  end
end

1.3 Results