OSEMN methodology
Step 4: Model
Feature selection
Excerpts from Python Machine Learning
Second Edition
By Sebastian Raschka and Vahid Mirjalili[1]
and other sources

Stepan Oskin

September 18, 2019

Abstract

1 Selecting meaningful features

If we notice that a model performs much better on a training dataset than on the test dataset, this observation is a strong indicator of overfitting. Overfitting means the model fits the parameters too closely with regard to the particular observations in the training dataset, but does not generalize well to new data, and we say the model has a high variance. The reason for the overfitting is that our model is too complex for the given training data. Common solutions to reduce the generalization error are listed as follows:

- Collect more training data
- Introduce a penalty for complexity via regularization
- Choose a simpler model with fewer parameters
- Reduce the dimensionality of the data

Collecting more training data is often not applicable. In the following sections, we will look at common ways to reduce overfitting by regularization and dimensionality reduction via feature selection, which leads to simpler models by requiring fewer parameters to be fitted to the data.

2 L1 and L2 regularization as penalties against model complexity

L2 regularization is one approach to reduce the complexity of a model by penalizing large individual weights, we define the L2 norm of our weight vector $\boldsymbol{w}$ as follows:

$$L2 : \|\boldsymbol{w}\|_2^2 = \sum_{j=1}^{m} w_j^2 \tag{1}$$

Another approach to reduce the model complexity is the related L1 regularization:

$$L1 : \|\boldsymbol{w}\|_1 = \sum_{j=1}^{m} |w_j| \tag{2}$$

Here, we simply replaced the square of the weights by the sum of the absolute values of the weights. In contrast to L2 regularization, L1 regularization usually yields sparse feature vectors; most feature weights will be zero. Sparsity can be useful in practice if we have a high-dimensional dataset with many features that are irrelevant, especially cases where we have more irrelevant dimensions than samples. In this sense, L1 regularization can be understood as a technique for feature selection.

3 Sequential feature selection algorithms

An alternative way to reduce the complexity of the model and avoid overfitting is dimensionality reduction via feature selection, which is especially useful for unregularized models. There are two main categories of dimensionality reduction techniques: feature selection and feature extraction. Via feature selection, we select a subset of the original features, whereas in feature extraction, we derive information from the feature set to construct a new feature subspace. In this section, we will take a look at a classic family of feature selection algorithms.

Sequential feature selection algorithms are a family of greedy search algorithms that are used to reduce an initial d-dimensional feature space to a k-dimensional feature subspace where k¡d. Greedy algorithms make locally optimal choices at each stage of a combinatorial search problem and generally yield a suboptimal solution to the problem, in contrast to exhaustive search algorithms, which evaluate all possible combinations and are guaranteed to find the optimal solution. However, in practice, an exhaustive search is often computationally not feasible, whereas greedy algorithms allow for a less complex, computationally more efficient solution.

3.1 Sequential Backward Selection (SBS) algorithm

The motivation behind feature selection algorithms is to automatically select a subset of features that are most relevant to the problem, to improve computational efficiency or reduce the generalization error of the model by removing irrelevant features or noise, which can be useful for algorithms that don't support regularization. A classic sequential feature selection algorithm is Sequential Backward Selection (SBS), which aims to reduce the dimensionality of the initial feature subspace with a minimum decay in performance of the classifier to improve upon computational efficiency. In certain cases, SBS can even improve the predictive power of the model if a model suffers from overfitting.

The idea behind the SBS algorithm is quite simple: SBS sequentially removes features from the full feature subset until the new feature subspace contains the desired number of features. In order to determine which feature is to be removed at each stage, we need to define the criterion function J that we want to minimize. The criterion calculated by the criterion function can simply be the difference in performance of the classifier before and after the removal of a particular feature. Then, the feature to be removed at each stage can simply be defined as the feature that maximizes this criterion; or in more intuitive terms, at each stage we eliminate the feature that causes the least performance loss after removal. Based on the preceding definition of SBS, we can outline the algorithm in four simple steps:

1. Initialize the algorithm with $k = d$, where d is the dimensionality of the full feature space $\boldsymbol{X}_d$.

2. Determine the feature x^- that maximizes the criterion:
   $x^- = arg\ max\ J(\boldsymbol{X}_k - \boldsymbol{x})$, where $\boldsymbol{x} \in \boldsymbol{X}_k$.

3. Remove the feature x^- from the feature set:

$\boldsymbol{X}_{k-1} := \boldsymbol{X}_k - \boldsymbol{x}^-;\ k := k - 1.$

4. Terminate if k equals the number of desired features; otherwise, go to step 2.

You can find a detailed evaluation of several sequential feature algorithms in Comparative Study of Techniques for Large-Scale Feature Selection by F. Ferri et al.[2].

4 Other feature selection algorithms

4.1 Variance Threshold

`VarianceThreshold` in `scikit-learn` is a simple baseline approach to feature selection. It removes all features whose variance doesn't meet some threshold. By default, it removes all zero-variance features, i.e. features that have the same value in all samples.

As an example, suppose that we have a dataset with boolean features, and we want to remove all features that are either one or zero (on or off) in more than 80% of the samples. Boolean features are Bernoulli random variables, and the variance of such variables is given by:

$$Var[X] = p(1-p) \tag{3}$$

so we can select using the threshold .8 * (1 - .8):

```
from sklearn.feature_selection import VarianceThreshold
sel = VarianceThreshold(threshold=(.8 * (1 - .8)))
```

References

[1] S. Raschka and V. Mirjalili, *Python Machine Learning, 2nd Ed.* . Birmingham, UK: Packt Publishing, 2 ed., 2017.

[2] F. J. Ferri, P. Pudil, M. Hatef, and J. Kittler, "Comparative study of techniques for large-scale feature selection," *Machine Intelligence and Pattern Recognition*, vol. 16, no. C, pp. 403–413, 1994.