Project 5: Support Vector Classification

CS 425

Samuel Steinberg

November 25th, 2019

Description:

This paper analyzes support vector classification (SVC) in machine learning, which is part of the bigger support vector machine (SVM) family of classification, regression, and outlier detection supervised learning methods. These have several advantages, such as: They are effective in high dimensional space, effective when number of dimensions is greater than the number of samples, is efficient (support vectors used in its decision function), and can use different kernel functions for the decision function. An SVM can be used for classification by constructing a hyperplane or set of hyperplanes in high dimensional space. Quality separation is by the hyperplane that has the largest distance to the nearest training data for any class. When this functional margin is larger, the lower the generalization error of the classifier. There were multiple datasets used in this project. The first was *ionosphere.data* which consists of radar data collected in Goose Bay, Labrador targeting free electrons in the ionosphere and was accompanied by *ionosphere.names* which describes the dataset. Next, there was the *vowel-context.data* set which contains speaking/vocal data to investigate context-sensitive learning and had a description file names *vowel-context.names*. The last two datasets were *sat.trn* and *sat.tst*. These sets contained data on mulit-spectral pixel values in a 3x3 neighborhood in a satellite image. It was given with *sat.name,* which contains information on the source and purpose of the collected data.

Data Pre-Processing:

The *ionosphere.data* dataset consisted of 351 instances and 35 attributes. The attributes consisted of:

- 34 continuous predictors
- 1 labeled “good” or “bad”, this is the binary classification task.

Since there were no missing values in the dataset, the only processing that was necessary was replacing the string binary classification task and representing it as an integer. “Good” classification was converted to 1, and “bad” classifications was converted to 0. This was necessary to use the float-type predictors to actually attempt to predict the classification. Data was separated into training, test, and validation sets, where training consisted of 70%, testing 20%, and validation 10% of the data instances.

Next, the *vowel-context.data* dataset consisted of 990 instances and 14 attributes. The attributes consisted of:

- Train or Test

- Speaker Number
- Sex
- 10 continuous features
- 1 vowel label to classify, this is a multiclass problem as classifications range from [0, 10].

Here, there were a few lines where rows of data were read in as one single string. Hence, the data needed to be whitespace delimited. Next, the first three columns (Train or Test, Speaker Number, and Sex) were irrelevant and were removed from the dataset. Besides this, the dataset did not contain any missing values and was complete. Data was separated into training, test, and validation sets, where training consisted of 60%, testing 30%, and validation 20% of the data instances.

The last two datasets used for this project were *sat.trn* and *sat.tst.* They hold information from the Landsat Satellite, with *sat.trn* representing the training data and *sat.tst* representing the testing (validation) data. The training data consisted of 4435 instances, while the testing data contained 2000 instances. This was another multiclass problem, since classifiers ranged from [1, 7] with classifier 6 being removed by the creators of the set due to validity concerns. The sets consisted of 37 attributes:

- 36 numerical ranging from 0-255 (4 spectral bands x 9 pixel neighborhood)
- 1 classification label

Similar to *vowel-context.data*, these datasets were complete but needed to be whitespace delimited. The data was delivered already split between training and testing/validation, so there was minimal pre-processing work for this dataset. It is of note that in *sat.name* it explicitly decrees that cross-validation should not be used on these datasets, and to only train and test with the given datasets.

Solution Description:

The first step after pre-processing is to standardize the data, which is a common requirement for most machine learning estimators. The formula used in this project was the z-score measurement represented in Figure 1:

$$z = \frac{(x - u)}{s}$$

Figure 1: Standard scaler used for this project: z-score

The z-score is a numerical measurement of a values relationship to its mean, measured in standard deviations from the mean. In the equation, *x* represents the sample for which the standard score *z* is being calculated, and where *u* is the mean of the population of samples and *s* is the standard deviation of the training samples.

The SVC equation used in this project is displayed in Figure 2. Variables in the equation are listed below it. Given training vectors $x_i \in R^p$, i = 1, …, n in two classes and $y \in \{1, -1\}^n$:

$$\min_{\alpha} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha$$
$$\text{subject to } y^T\alpha = 0$$
$$0 \leq \alpha_i \leq C, i = 1, \ldots, n$$

Figure 2: SVC Formula

- *Q* represents an *n x n* positive semidefinite matrix
  - $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, where $K(x_i, x_j) = \emptyset(x_i)^T \emptyset(x_j)$ is the kernel.
- *e* represents a vector of all 1's
- $C > 0$ is the upper-bound
- α represents the dual coefficients used

Using the same variables as above, its decision function is illustrated in Figure 3:

$$\text{sgn}(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + \rho)$$

Figure 3: SVC Decision Function

Instances should be on the right side of the hyperplane to be classified correctly but should also be some distance away. The goal is to maximize the margin, which is the distance from the hyperplane to the instances closest to it on either side. See Figure 4:

$$\frac{r^t(\boldsymbol{w}^T\boldsymbol{x}^t + w_0)}{\|\boldsymbol{w}\|} \geq \rho, \forall t$$

Figure 4: Optimal Separating Hyperplane

Here, *w* and w_0 are found such that:

$w^T x^t + w_0 \geq 1$ for $r^t = 1$

$w^T x^t + w_0 \leq -1$ for $r^t = -1$

Also, x^t is found as the member of a sample $X = \{x^t, r^t\}$. All of this we would like to be at least some value ρ.

Analysis and Discussion:

The goal of this project was to find the most optimal hyperparameters for each dataset. The first step was to standardize each set, fit the data with SVC, and then run a course grid search with this data to find the best range of hyperparameters, and then a fine grid search for finding the optimal hyperparameters. There were different parameters used in the SVC: 'kernel', 'C', and 'gamma'.

The 'kernel' parameter specifies the kernel type to be used, which for this project was either linear, polynomial, Radial Basis Function (RBF), or sigmoid. These functions are used to perform the *kernel trick*. The kernel trick avoids explicit mapping needed to get linear algorithms to learn a nonlinear function or boundary. For the entire input space, functions can be expressed as an inner product in another space. The 'C' parameter represents the penalty of the error term. And the 'gamma' parameter controls the scaling of the kernel function. The following set was used for this project in order to get a wide range of values, and to get the most optimal range:

Kernel Function (kernel): Linear, Polynomial, RBF, Sigmoid

Error Cost (C): 1, 2, 3, 4, 5, 10, 15, 20, 25, 50

Kernel Scaling (gamma): .0001, .001, .01, 1, 10, 100

For each of these hyperparameters they were tested for recall and precision. Recall is a measure of how many actual relevant results are returned, while precision measures result relevancy.

The first dataset tested was the *ionosphere.data* set. After running the course grid search to find the best range of hyperparameters, and the fine grid search was run to find the optimal hyperparameters, which came out to the following:

Kernel Function: RBF

Error Cost (C): 2

Kernel Scaling (gamma): 0.01

When running the training data these resulted in the best accuracy metrics, giving almost 95% accuracy:

Accuracy: 94.9%

Standard Deviation: (+/-0.050)

Below are the total metrics for the coarse search:

Table 1: Ionosphere Course Search Results

Classification	Precision	Recall	F1 Score	Support
0	0.94	0.81	0.87	21
1	0.92	0.98	0.95	48
Totals	0.93	0.89	0.91	69

Because the range was now narrowed down to the best hyperparameters, this data was used on the testing and validation sets. This resulted in superb performance metrics:

Table 2: Ionosphere Fine Search Best Results

Classification	Precision	Recall	F1 Score	Support
0	0.96	1.00	0.98	21
1	1.00	0.90	0.95	48
Totals	0.98	0.95	0.965	69

After narrowing down the best hyperparameters the model improved by 5% accuracy, 5.5% in F1 Score, and 6% in recall. These are all very significant improvements. The RBF kernel is a spherical kernel that does well with smaller datasets but does not scale very well, so it is not a shock that it did well on a dataset with only 351 instances. Additionally, the relatively small scaling probably helped here too. Figure 5 represents an illustration of just how accurate the model was on the validation set. Overall, it seemed that a lower error cost and kernel scaling fit this set best, as shown in the results. Besides the great accuracy performance, F1 Score was also very high. This is important because it is a measure of a test's accuracy using the harmonic mean of precision and recall. This helps to balance precision and recall, which at times can offer a better idea of model performance than accuracy.

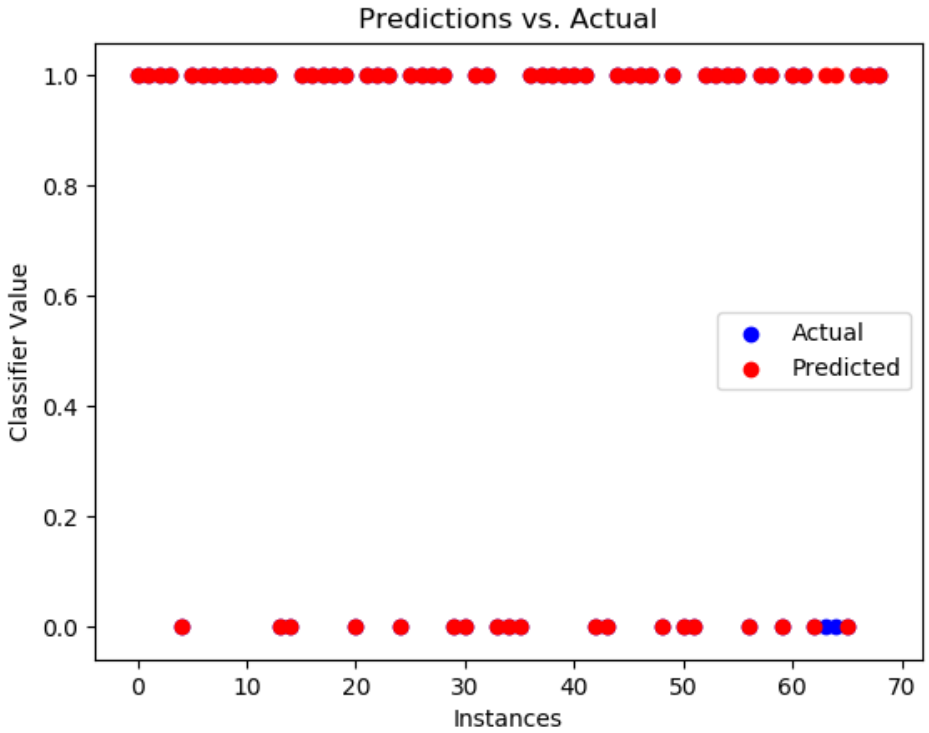

Figure 5: Ionosphere Dataset Validation Set Classifications

The next dataset tested was the *vowel-context.data* set. After running the course grid search to find the best range of hyperparameters, and the fine grid search was run to find the optimal hyperparameters, which came out to the following:

Kernel Function: RBF

Error Cost (C): 10

Kernel Scaling (gamma): 0.1

The performance here was even better than for the ionosphere set, which could be explained by having a smaller number of features and more data, which is a major advantage of SVC. The most optimal range of hyperparameters that achieved near perfect performance was chosen on the training set scores.

Below are the coarse search results:

Table 3: Vowel Course Search Best Results

Classification	Precision	Recall	F1 Score	Support
0	1.00	1.00	1.00	32
1	1.00	1.00	1.00	28
2	1.00	1.00	1.00	29
3	0.96	1.00	0.98	17
4	1.00	1.00	1.00	26
5	0.94	0.98	0.96	25
6	1.00	1.00	1.00	36
7	1.00	1.00	1.00	27
8	1.00	1.00	1.00	26
9	1.00	1.00	1.00	28
10	1.00	1.00	1.00	25
Totals	0.99	1.00	0.99	299

Unsurprisingly, utilizing the best range of hyperparameters resulted in perfect performance.

Table 4: Vowel Fine Search Best Results

Classification	Precision	Recall	F1 Score	Support
0	1.00	1.00	1.00	32
1	1.00	1.00	1.00	28
2	1.00	1.00	1.00	29
3	1.00	1.00	1.00	17
4	1.00	1.00	1.00	26
5	1.00	1.00	1.00	25
6	1.00	1.00	1.00	36
7	1.00	1.00	1.00	27
8	1.00	1.00	1.00	26
9	1.00	1.00	1.00	28

10	1.00	1.00	1.00	25
Totals	1.00	1.00	1.00	299

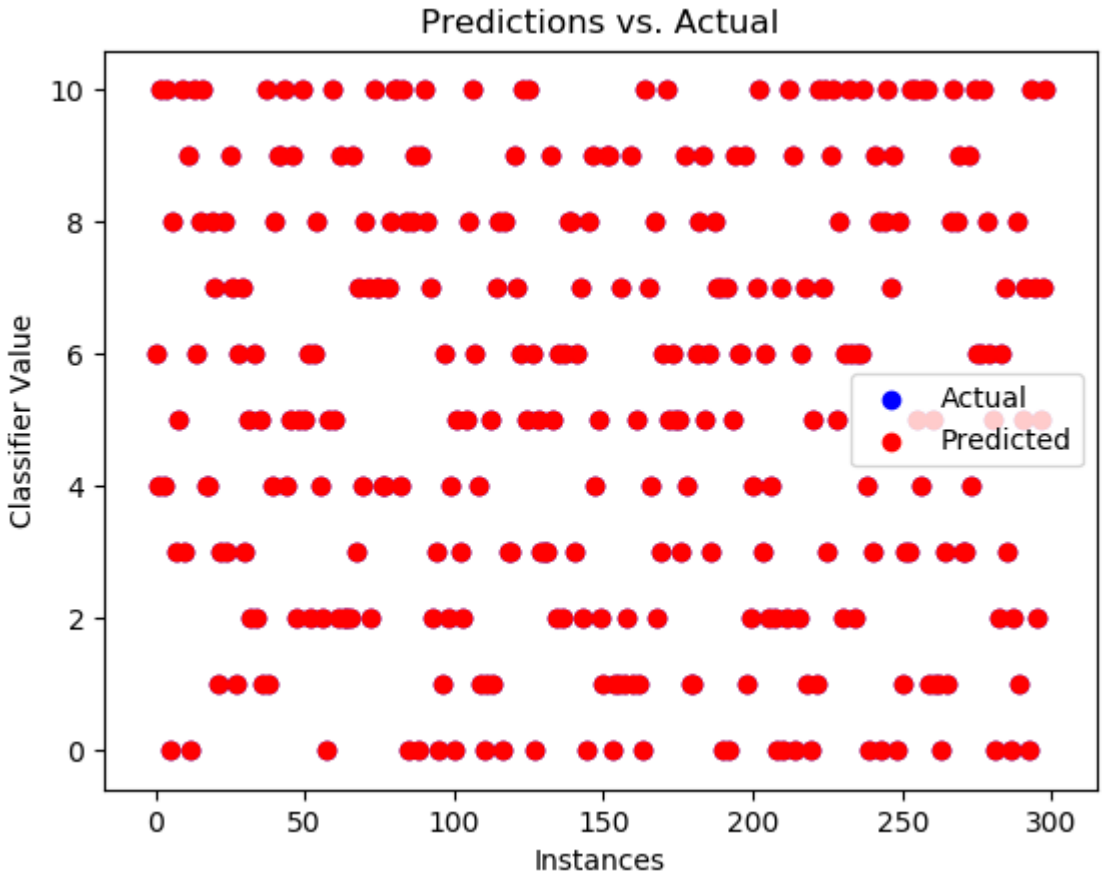

Figure 6: Vowel Dataset Validation Classifications

With both the fine grid search and coarse grid search resulted in perfect or near-perfect classifications, it could be argued that a dataset around 1000 instances with a high dimension could be what SVC works best on. Performance was spectacular. A surprise was that the RBF spherical kernel worked so well here. It could have been aided by a larger scaling factor at 0.1 than the ionosphere's set, or the Error Cost (C) which trades off correct classification against the maximization of the decision function's margin since it was so high.

The last datasets tested were the Satellite ones (test and training datasets pre-separated). Since these datasets were much larger than the others, they took much longer to run. This suggests that having a large number of hyperparameters with a large dataset can lead to a much higher computational cost in the context of both time and space. After running the same coarse search as previous datasets, the best range of hyperparameters came out to be:

Kernel Function: RBF

Error Cost (C): 2

Kernel Scaling (gamma): 0.1

The performance for this set ended up less than previous sets. The most optimal range of hyperparameters produced the following results:

Table 5: Satellite Fine Search Best Results

Classification	Precision	Recall	F1 Score	Support
1	0.99	1.00	0.99	462
2	0.97	0.97	0.97	224
3	0.87	0.97	0.92	396
4	0.79	0.64	0.71	211
5	0.95	0.91	0.93	237
7	0.89	0.89	0.89	470
Totals	0.91	0.90	0.90	2000

Here, accuracy was lower than in previous sets, dipping to 91%. There were certain classifications that scored very well in this set, such as "1" being nearly perfect, while others such as "4" were lacking. Overall, however, the results were still pretty good. The large size of the datasets might have somewhat affected the results.

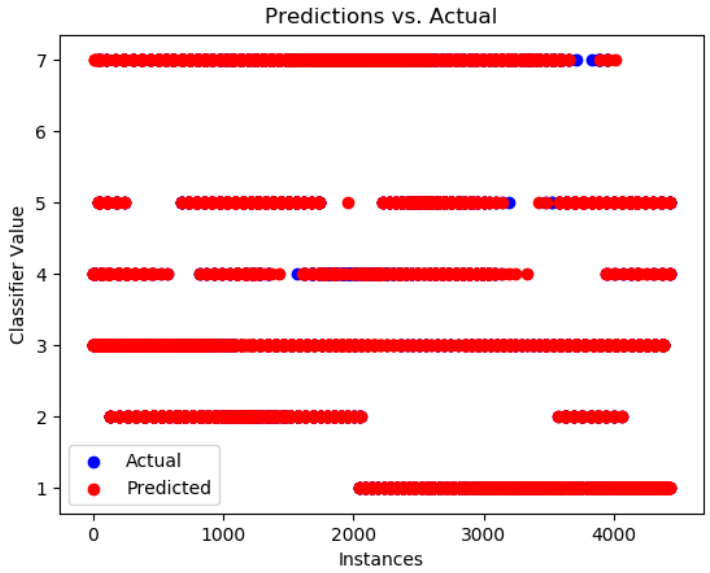

Figure 7: Satellite Dataset Testing/Validation Classification

Overall, SVM's and their SVC classifiers have advantages with high dimensional data and are generally efficient. Their flexibility in using different kernel methods and decision functions is also an advantage. A downside, however, is that a large number of hyperparameters can lead to a large computational time and space expense. This was seen in the much larger Satellite dataset but can be partially remedied by first doing a coarse grid search to narrow down a range of most optimal hyperparameters, followed by a fine grid search to find the actual most optimal parameters. The algorithms ended up working very well for all the datasets, with high performance metrics being reached. The best hyperparameters seemed to be a higher error cost (around 1 for the most part), the RBF spherical kernel, and a gamma around 0.01-0.1. This project also brings up the question of how performance might fluctuate with more data instances, despite good high dimensionality performance. Accuracies throughout the project for the most optimal range of hyperparameters went from ~98% for 351 instances, to 100% for 990 instances, to ~91% for 6435 instances (Satellite database combined). Additionally, runtimes were similar between the lower instance ionosphere and vowels datasets, however when the Satellite one was run the time expense was exponential. Additionally, the coarse grid search being run before the fine grid search turned out to be quite useful. Once determining the best range of hyperparameters, the fine grid search could narrow in on those and find what the most optimal hyperparameters actually are.