Report for Sheet 04

Lab Course Machine Learning and Data Analysis

Mario Tambos

July 13, 2017

Implementation comments

The code was structured mostly in functions following an imperative paradigm, except for the `svm_smo` and `svm_qp` classes, which were required by the assignment's statement, and the ancillary `SupervisedMethod` class.

One function was declared for each assignment in Part 2.

Beyond `matplotlib`, `numpy` and `scipy`, `seaborn` was used to improve the plots aesthetics in Part 2, `scikit-learn` for hyperparameter selection and `joblib` to parallellize some tasks.

All tasks were completed, and all tests passed.

For some, as of the time of this writing, undiscovered reason, the `cv` function written (included in the code for reference) did not work as expected. Therefore, `scikit-learn`'s `GridSearchCV` was used for the experiments in Part 2.

Moreover, using the written `svm_smo` class in Assignment 7 caused an extremely long run time (unfinished after more than 10 hours, see `assignment7_own` function). Consequently, `scikit-learn`'s `SVC` class was used instead for this experiment.

In all experiments, the `zero-one-loss` functions was chosen as the objective to minimize.

Assignment 3

If we set the variables as follows:

$$
\begin{aligned}
P_{i,j} &= y_i * y_j * k(x_i, x_j); \vec{P} \in \mathbb{R}^{n \times n} \\
\vec{q} &= -\mathbb{1}; \quad \vec{q} \in \mathbb{R}^n \\
G_{i,j} &= -\mathbb{I}; \quad \vec{G} \in \mathbb{R}^{n \times n} \\
\vec{h} &= \vec{0}; \quad \vec{h} \in \mathbb{R}^n \\
A_{1,i} &= y_i; \quad \vec{A} \in \mathbb{R}^{1 \times n} \\
b &= 0; \quad b \in \mathbb{R}
\end{aligned}
$$

and considering that $\max_x f(x) = \min_x -f(x)$, then the `cvxopt` quadratic optimization problem is equivalent to the dual of the SVM problem. In the `cvxopt` formulation, the variable $\vec{x}$ corresponds to the $\vec{\alpha}$ of the SVM dual.

Assignment 4

Point 1.

Figure 1 shows the decision boundary of the model trained with the best C and σ parameters, found using cross validated grid search.

Point 2.

Figure 2 shows the decision boundaries of under- and over-fitting SVM models. For the under-fitting model $C = 100$ and $\sigma = 100$ were used. For the over-fitting model $C = 1$ and $\sigma = 0.001$ were used.

Point 3.

Figure 3 shows the ROC curve resulting from varying the b parameter in the model from Point 1. The model's parameters are:

- $C \approx 1.5199$
- $\sigma \approx 0.0123$

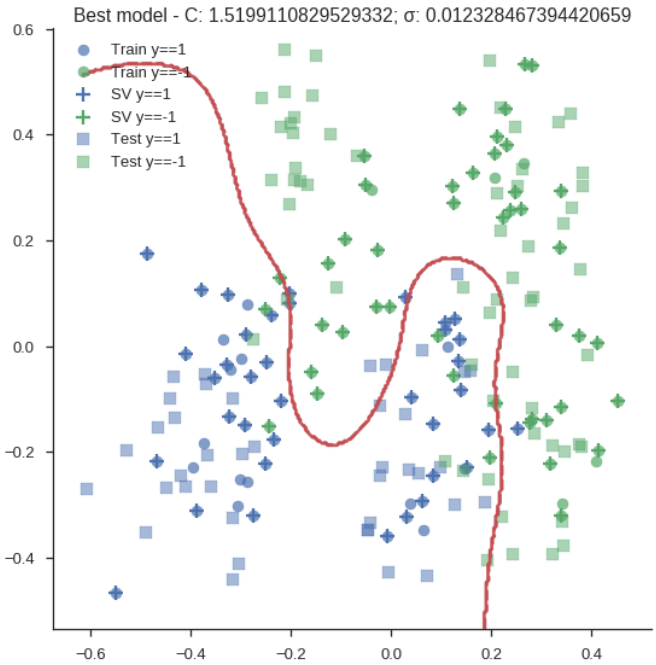

Figure 1: Scatter plot of the `easy_2d` dataset, and decision boundary for the SVM's optimal C and σ parameters.

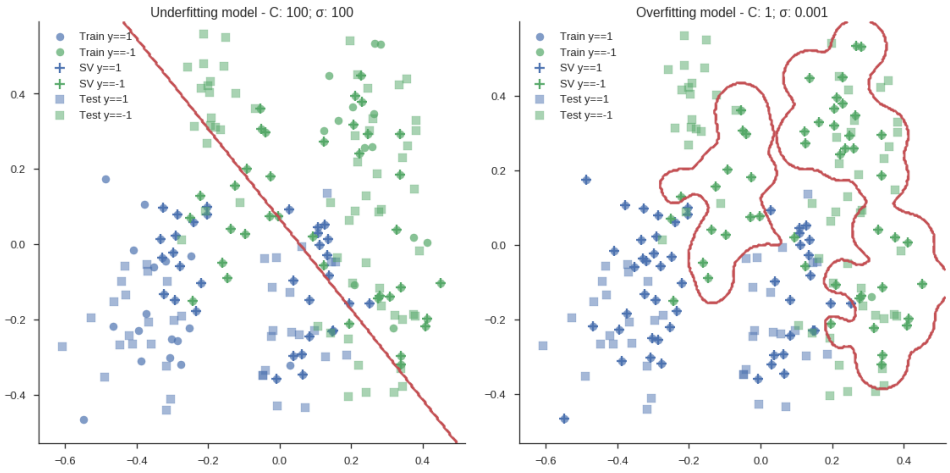

Figure 2: Scatter plots of the `easy_2d` dataset, and decision boundaries for under- (left) and over-fitting (right) SVMs.

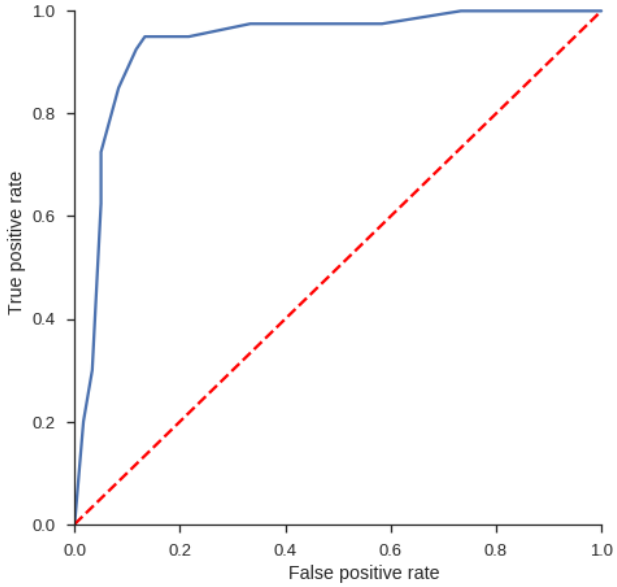

Figure 3: ROC curve resulting from varying the b parameter in the model from Point 1.

Assignment 5

The biggest point of improvement would be the iteration over the samples, in line 5. If the block of lines 5-15 could be expressed as matrix operations, one could take advantage of the optimizations of linear algebra libraries.

Assignment 6

For this assignment, three rounds of cross validated grid search were performed, one for each pair of classes. In all cases only the linear kernel was used, the C parameter was explored in the logarithmic range $[1E^0, 1E^4]$ and the kernel's r parameter [1] was explored in the linear range $[-10, 10]$.

The best parameters found, together with their minimum losses, are listed in table 1:

[1] Linear kernel's definition: $k(x, y) = \langle x, y \rangle + r$

Classes	Test Loss	C	r
1 vs 2	$= 0$	1	≈ -7.1429
1 vs 3	$= 0$	1	≈ -7.1429
2 vs 3	≈ 0.3333	≈ 3.7275	≈ 1.4286

Table 1: Iris dataset experiment's results

kernel	definition
linear	$k(x, y) = \langle x, y \rangle$
polynomial	$k(x, y) = (\gamma \langle x, y \rangle + r)^d$
Gaussian	$k(x, y) = \exp(-\gamma \|x, y\|_2^2)$

Table 2: Kernel's definitions

Assignment 7

Point 1.

For this assignment, nine rounds of cross validated grid search were performed. In each round one digit was chosen as the positive class, and the rest were set as the negative class.

In all cases linear, polynomial and Gaussian kernels were tried. The definition of each kernel is shown in table 2

The C parameter was explored in the logarithmic range $[1E^0, 1E^4]$. This was the only hyperparameter explored for linear kernels.

For polynomial kernels, the *degree*, γ and r hyperparameters were also explored in the values $[1, 2, 3]$, logarithmic range $[-1, 2]$ and values $[-2, -1, 0, 1, 2]$, respectively.

For Gaussian kernels, the γ hyperparameter was also explored in the logarithmic range $[-1, 2]$.

The best parameters found, together with their minimum losses, are listed in table 3:

Point 2.

Figure 4 shows, for each digit, five randomly selected support vectors, to-

Digit	Kernel	Test Loss	C	γ	r	*degree*
0	polynomial	≈ 0.002	1	0.1	1	2
1	polynomial	≈ 0.002	10	0.1	2	2
2	polynomial	≈ 0.0261	1	1	2	3
3	polynomial	≈ 0.0221	1	0.1	2	3
4	polynomial	≈ 0.0201	1	0.1	1	2
5	polynomial	≈ 0.0261	1	10	2	2
6	polynomial	≈ 0.004	1	0.1	1	3
7	polynomial	≈ 0.0141	1	1	1	3
8	polynomial	≈ 0.0141	1	10	2	2
9	Gaussian	≈ 0.0141	10	0.1	NA	NA

Table 3: USPS experiment's 1-vs-rest results

gether with five randomly selected images from the training set, with the same label.

Point 3.

A problem could arise from class imbalance. Given that there are much more negative examples in the one-vs-rest setting, misclassified positive examples end up contributing little to the overall loss. One possible solution could be to weight an individual sample's error proportionally to the inverse of the sample's class frequency.

Figure 4: USPS experiment - sample support vectors (right) and sample images(left). Each row indicates which digit was used as the positive class.