183.605 Machine Learning for Visual Computing Assignment 2

Michael Reiter

25. November 2019

Assignment via TUWEL. Please be aware of the deadlines in TUWEL.

- Upload a zip-file with the required programs. You can choose the programming language.
- Add a PDF document with answers to all of the questions of the assignment (particularly all required plots) and description and discussion of results.

1 Assignment 2

The aim of this second assignment is to deepen the understanding of the support vector machine by implementing it using a general quadratic optimizer. The tasks are described for implementation in Matlab, however you can use any other programming language of your choice. In Matlab you can use the function `quadprog`. If you are using Python, R or any other programming language please use a suitable function for quadratic optimization (available SVM functions are not allowed to be used). You can find the explanation of the relevant methods in the lecture slides (Part 4). Of course you can also refer to other sources.

1.1 The dual optimization problem

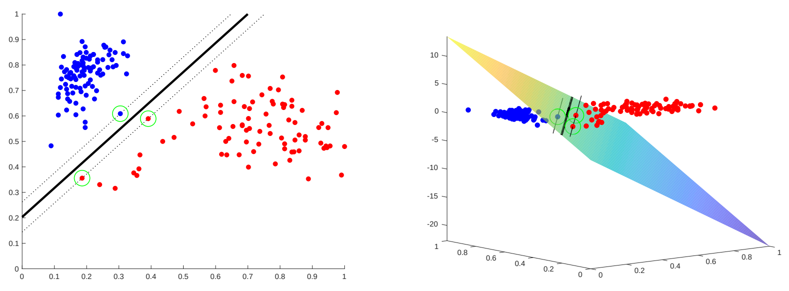
Tasks:

- Choose a suitable training set of linearly separable data $\mathbf{x}_i \in \mathbb{R}^2$ with targets t_i, where $i \in \{1, ..., N\}, N \geq 100$. For example, you can use a linearly separable subset of $\mathcal{T}$ of *assignment1*.
- Plot the input vectors in $\mathbb{R}^2$ and visualize corresponding target values (e.g. by using color).
- Implement the function `[alpha, w0] = trainSVM(X,t)` using the quadratic optimizer (e.g., `quadprog`) to solve the dual optimization problem. The vector `alpha` holds the optimal dual parameter values, i.e., the lagrange multipliers α_i for all N input vectors (see lecture slides). Alternatively, the function could also return only $\alpha_i > 0$ together with corresponding support vectors `Xs` and

target values `ts`. In this first step, do not employ a kernel function, directly use the inner products instead (i.e., use a "linear kernel") and do not use slack variables. `w0` is the offset of the decision plane, which can be computed using `alpha` and one support vector (see lecture slides).

- Write the function `[y] = discriminant(alpha,w0,X,t,x)` which implements the discriminant function $y = d(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$. Use the dual form parameterized by $\boldsymbol{\alpha}$ instead of $\mathbf{w}$ in the implementation. Use this function to predict the class label for a new input vector using the trained SVM. A probably more efficient[1] variant is `[y] = discriminant(alpha,w0,X,t,Xnew)` which returns a vector `y` containing the y-values for a batch of input vectors stored in columns of the matrix `Xnew`.

- Visualize the support vectors and plot the decision boundary and the margin curves $d(\mathbf{x}) = 1$ and $d(\mathbf{x}) = -1$ (see Figure 1(a)).



(a) (b)

Figure 1: (a) the resulting decision boundary and geometric margin of the SVM using no kernel (i.e., a *linear* kernel) and (b) the corresponding discriminate function function as a surface.

1.2 The kernel trick

Tasks:

- Enhance the above functions with RBF-Kernel functions: Either write the function `[k] = rbfkernel(x1,x2,sigma)` or use an anonymous function for the kernel evaluation of two input vectors and the RBF parameter σ. Use a function handle as an additional parameter of `trainSVM` and `discriminant` to pass the kernel to these functions[2].

- Try different values for `sigma`.

- Enhance `trainSVM` for the use of *slack variables*, i.e., introduce the regularization parameter C in the constraints of the dual optimization problem (see

[1]This variant allows to calculate the function evaluation for a batch of input data, reducing the number of function calls. In most cases this speeds up the calculation significantly.

[2]Use a suitable mechanism in other programming languages.

lecture slides). Hint: Note that with $C < \infty$ you have to take care about selecting a support vector $\mathbf{x}_s$ with margin $|d(\mathbf{x}_s)| = 1$ to calculate w_0.

- Use the complete 2D data set $\mathcal{T}$ of *assignment1*, plot the data, visualize the support vectors and plot the decision boundary and margin curves (see Figure 2(a)).

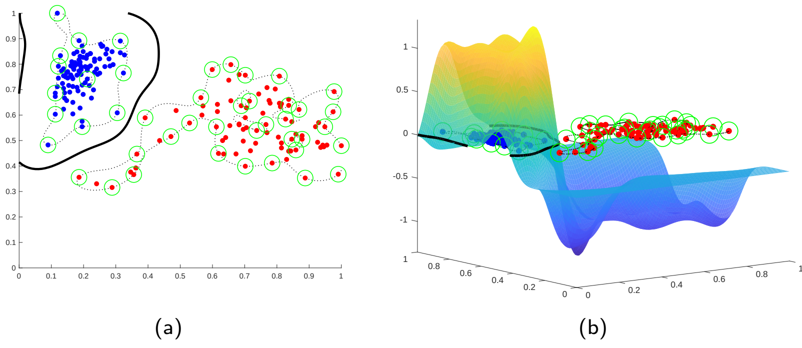


Figure 2: (a) the resulting decision boundary and geometric margin of the SVM using an RBF kernel and (b) the corresponding discriminate function as a surface.

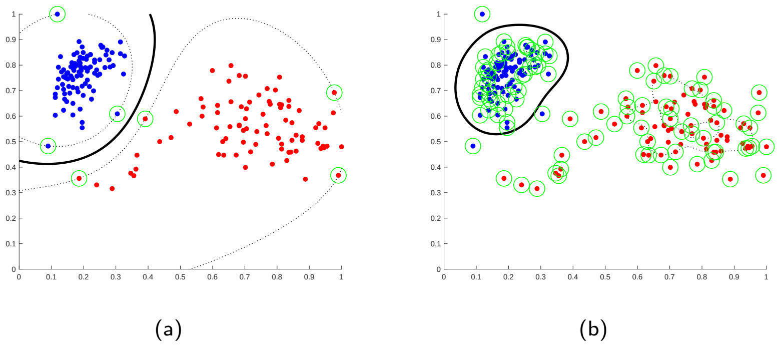


Figure 3: (a) effect of increasing the RBF-kernel parameter σ and (b) decreasing the regularization parameter C (SVM with *slack variables*) in comparison to Figure 2.

1.3 Model-complexity and model-selection

Once your SVM implementation has been testet on the 2D data set, test the classification performance on image data, i.e. the input vectors of dimensionality $d = 28 \times 28 = 784$ are the MNIST images. Start with a comparison of the linear SVM with the perceptron:

Tasks:

- Generate at least $M = 150$ disjoint training sets $\mathcal{T}_k, k \in \{1, \ldots, M\}$ by selecting images from the MNIST training set, where each $\mathcal{T}_k$ consists of not more than $N = 70$ images.
- For all $k \in \{1, \ldots, M\}$ train the SVM and the perceptron (see *assignment1*) using $\mathcal{T}_k$ and calculate the test error rate R_k on the *MNIST test set* for both models (using all available images of the two respective classes in the test set).
- Compare the average error $R_{avg} = \frac{1}{M} \sum_{k=1}^{M} R_k$ of the perceptron with that of the SVM using the *linear* kernel and $C = \infty$ (no slack variables).

The last experiment is about tuning of meta-parameters C and σ when using an RBF-kernel:

Tasks:

- Analyse the effect of changing C or σ on the average test error rate R_{avg} of the SVM.
- Analyse the effect of changing C or σ on the average training error (proportion of false classifications in the training set $\mathcal{T}_k$ after training with $\mathcal{T}_k$).
- Analyse the relation of the average number of support vectors and the average test error rate R_{avg} of the SVM.
- Use the RBF-kernel and $C < \infty$: find the optimal C and σ by M-fold cross validation on the union of all training sets: For all $k \in \{1, \ldots, M\}$ train the SVM using $\mathcal{T}_k$ and compute the error rate on the union of $\mathcal{T}_j, j \in \{1, \ldots, M\}, j \neq k$. Choose C and σ such that this error is smallest on average.
- Does the RBF-kernel SVM with optimal C and σ outperform the linear SVM on the MNIST test set?