MTAT.03.227 Machine Learning (Spring 2015)
Exercise session XIII: Support Vector Machines

Konstantin Tretyakov

May 12, 2015

The aim of this exercise session is to get acquainted with the inner workings of support vector machine classification and regression. As usual, for all exercises you need to write a brief explanation and, for most of them, also a short piece of code demonstrating the result. You can submit your whole solution as a single *decently commented* R or Rmd file, provided it is sequentially readable and executable.

We shall use a slightly modified version of the familiar base code from Exercise Session VI. Fetch it at `svm_base.R`.

Maximal margin

Exercise 1 (2pt). We begin by analyzing a trivialized dataset consisting of just four points. Use the functions `load.data` and `plot.data` to visualize it. Use manual examination of the data to derive the parameters $(\mathbf{w}, b)$ of the canonical[1] maximal-margin classifier.

Hints.

1. Use pen and paper to sketch the location of the maximal margin separating hyperplane for the given dataset and its two margin-defining isolines $f(\mathbf{x}) = \pm 1$.
2. By visual examination, guess the coordinates of the weight vector $\mathbf{w}$ (up to a constant). Use `plot.classifier(w)` and `plot.data(data, add=T)` to verify your guess.
3. Next, find the length of the normal, $|\mathbf{w}|$, such that would assure that the closest training points are located at the isolines ± 1. For this, note that the distance between those two isolines is equal to $d = 2/|\mathbf{w}|$.
4. Rescale the $\mathbf{w}$ you guessed in Step 2 to have length you computed in Step 3.
5. Finally, compute the value for the bias parameter b by using the fact that $f(\mathbf{x}_1) = -1$.

[1] A canonical maximal-margin classifier is the classifier for which the functional margin of the closest training example is equal to 1. It is what we were talking about on the lecture.

6. Visualize the computed $(\mathbf{w}, b)$ using `plot.classifier`.

Next, we shall use R's quadratic programming facilities to find a maximal margin classifier for the same dataset. There are several packages which we could use, but the simplest option seems to be the `quadprog` package and its `solve.QP` function. Load the package using the command `library(quadprog)`.

For the following you need to know that the invocation

```
> solve.QP(D, d, t(A), b)
```

will numerically solve the following quadratic programming problem:

$$\text{argmin}_{\mathbf{x}} \frac{1}{2}\mathbf{x}^T\mathbf{D}\mathbf{x} - \mathbf{d}^T\mathbf{x},$$

$$\text{s.t. } \mathbf{A}\mathbf{x} \geq \mathbf{b}.$$

Exercise 2 (2pt). Recall that the hard-margin SVM optimization problem is: *Primal form*

$$\text{argmin}_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2,$$

$$\text{s.t. } \forall i \quad (\mathbf{w}^T\mathbf{x}_i + b)y_i \geq 1.$$

Use `solve.QP` to compute a solution to this problem for our sample dataset.

Hints.

1. Determine what is $\mathbf{x}$ in this problem.
2. Rewrite the objective in the form $\frac{1}{2}\mathbf{x}^T\mathbf{D}\mathbf{x} - \mathbf{d}^T\mathbf{x}$. What are $\mathbf{D}$ and $\mathbf{d}$?
3. How many constraints are there? Rewrite them in the form $\mathbf{A}\mathbf{x} \geq \mathbf{b}$.
4. Finally, define the necessary variables in R and invoke `solve.QP` as shown above.
5. The quadratic programming solver will probably complain that the matrix $\mathbf{D}$ is not positive semidefinite. You can address by adding a tiny value on the diagonal of this matrix:

   ```
   D = D + 1e-10*diag(nrow(D))
   ```

6. Confirm that the solution matches the one you obtained manually.

Exercise 3 (2pt). Now let us solve the SVM in *dual* form. Some formalities aside, the dual form is obtained by representing the weight vector $\mathbf{w}$ as a linear combination of training points: *Dual form*

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i,$$

where α_i are the unknowns (the *dual variables*) we shall be seeking for instead of $\mathbf{w}$.

In the dual form the optimization problem turns into[2]:

$$\text{argmin}_{\boldsymbol{\alpha}} \left(\frac{1}{2} \boldsymbol{\alpha}^T (\mathbf{K} \circ \mathbf{Y}) \boldsymbol{\alpha} - \mathbf{1}^T \boldsymbol{\alpha} \right)$$

$$\text{s.t. } \boldsymbol{\alpha} \geq 0,$$
$$\mathbf{y}^T \boldsymbol{\alpha} = 0,$$

where

$\mathbf{K} = \mathbf{X}\mathbf{X}^T$ is the *kernel matrix*,
$\mathbf{Y} = \mathbf{y}\mathbf{y}^T$,
$\mathbf{1}$ is a vector of ones, and
$\circ$ denotes elementwise multiplication.

Recast this formulation in the format suitable for `solve.QP` and find both the dual solution $\boldsymbol{\alpha}$ and the corresponding bias term b. Finally, use the obtained $\boldsymbol{\alpha}$ values to compute $\mathbf{w}$ and compare your result to two previous attempts.

Hints.

1. Similarly to the previous exercise, you first need to recast the problem in terms of $\mathbf{D}$, $\mathbf{d}$, $\mathbf{A}$, $\mathbf{b}$. Note that there is one equality constraint now. Include its coefficients as the first row of the matrix $\mathbf{A}$, its right side as the first element of vector $\mathbf{b}$ and provide the parameter `meq=1` to `solve.QP`.
2. Similarly to the previous exercise, if the solver complains about the lack of positive semidefiniteness, add a small value to the diagonal of $\mathbf{D}$.
3. Compute $\mathbf{w}$ as
$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i.$$
Note that this can be done concisely matrix multiplication.

[2]Note that in the lecture there was also the constraint $\boldsymbol{\alpha} \leq C$. This constraint is not present in the hard-margin case, however.

4. To find the bias term b find any *support vector* and use the fact that for a support vector $\mathbf{x}_i$, in the case of hard-margin classification, it must hold that
$$y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1.$$

Examine the values α that you obtained. Try the following visualization:

```
plot.classifier(w, b)
plot(data, add=T)
text(data$X[,1], data$X[,2], alphas)
```

Exercise 4* (3pt). In the case of hard-margin classification you may always find b by relying on the fact that all support vectors are always located exactly on the margin (i.e. they satisfy $f(\mathbf{x}_i) = y_i$). In the case of soft margin classification this idea can also sometimes be exploited. Namely, all support vectors which have $0 < \alpha < C$ are also necessarily on the margin. However, this need not always be the case – it may turn out so that *all* support vectors of a soft-margin SVM are violating the margin.

Finding the bias term in dual

How should you compute b in this case? Provide a proof of your claim.

Exercise 5 (1pt). Finally, let us use a third-party tool to fit an SVM model. The de-facto standard implementation in R is provided by the package `e1071` and its `svm` function[3]. Invoke it as follows:

e1071

```
m = svm(data$X, data$y, scale=FALSE,
              kernel="linear", type="C-classification", cost=9e9)
```

Examine the output. Find the $\boldsymbol{\alpha}$ vector and the bias term. Do they match your previous results? Compute the $\mathbf{w}$ vector.

Exercise 6 (1pt). So far we have only examined hard-margin classification (note that the `svm` function does not have a specific hard-margin mode, but specifying $C = 9 \cdot 10^9$, as we did in the previous exercise, is more-or-less as good as prohibiting any margin violations). Now let us study the effect of "relaxing" the margin.

Soft-margin

First, add a new datapoint $\mathbf{x}_5 = (0, 5)^T$ of class $y_5 = -1$ to the dataset:

```
data = load.data()
data$X = rbind(data$X, c(0, 5))
data$y = c(data$y, -1)
```

Now run `svm` with `cost = 9e9` and plot the resulting classifier using `plot.classifier` and `plot.data`. Try to guess what happens to the separating line once you reduce the cost to 1 and then further to 0.1. Verify your guesses.

Exercise 7 (2pt). A *kernel* (to be covered in the upcoming lecture) is a generalization of an *inner product*. It turns out that everywhere where we use an inner product $\mathbf{x}_i^T \mathbf{x}_j$, we may instead use a nonlinear function $K(\mathbf{x}_i, \mathbf{x}_j)$, which will serve the same purpose but will make our model non-linear. *RBF kernel*

One of the most popular kernels is the *RBF* kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right).$$

Study its properties on our toy example and answer the following questions.

1. The linear SVM in dual form, as you should know, corresponds to the following functional:
$$f(\mathbf{x}) = \sum_i \alpha_i y_i \mathbf{x}_i \mathbf{x} + b.$$
Is this functional linear in $\mathbf{x}$? Prove it.
2. What is the corresponding functional of an SVM with an RBF kernel? Is it linear in $\mathbf{x}$?
3. What is the value of $K(\mathbf{x}, \mathbf{z})$ for any point $\mathbf{z}$ that is geometrically very close to $\mathbf{x}$? What is the value of $K(\mathbf{x}, \mathbf{z})$ for points $\mathbf{z}$ that are far from $\mathbf{x}$?
4. Use the `svm` function to train an SVM classifier with an RBF kernel for the data you used in the previous exercise. Let $\gamma = 1$ and $C = 9 \cdot 10^9$.
5. Use the function `plot.svm.functional`, provided in the base code, to visualize the resulting functional[4]. Locate the decision boundary on the resulting plot.
6. Try different values of $\gamma \in \{0.001, 0.01, 0.1, 1, 10\}$ and visualize the results. What do you observe?
7. For $\gamma = 0.1$ try different values of $C \in \{0.1, 1, 10\}$. What do you observe?
8. What, do you think, is a good way for picking suitable values of γ and C in real-life applications?

Exercise 8* (1pt). In the lecture we did not get to cover the concept of *Support vector regression (SVR)*. Read about this approach on your own. As the answer to this exercise, provide the formulation of the SVR optimization problem in primal form, explaining the meaning of the parameters. *Support vector regression*

Exercise 9* (2pt). Finally, let us consider a simple case study. We shall use the `spam` dataset from the package `ElemStatLearn`: *Case study*

[3] This is actually a packaged version of the LibSVM C library.

[4] Note that there is a function `plot.svm` in the `e1071` package, but it does not exactly do what we need here.

```
> install.packages("ElemStatLearn")
> library(ElemStatLearn)
> data(spam)
```

Each row in the `spam` data frame corresponds to an email. The emails have been preprocessed and frequencies of certain words have been extracted – those frequencies form the input features. You need to train a classifier to predict the value of the output feature (the `spam` attribute).

Leave 25% of the training examples for final testing, and use the 75% for training. Try SVM and at least one other classifier. For SVM, try at least the linear and the RBF kernel and do not forget to tune the the `gamma` and `cost` parameters. Explain how you proceeded. Once you settle on the best model, evaluate it once on your held-out 25% and report the result.