HW2 Solution sketch

Jonatan Kilhamn

February 2017

1 Theoretical problems

1.1 SVM by inspection

The decision boundary should pass between the two classes of points, and the distance to the closest point on each side should be the same. We see that this line should pass through e.g. $(1, 2)$ and $(3, 0)$. The weights and bias fall out as

$$\begin{aligned} \mathbf{w} &= (1, 1)^\top \\ b &= -3 \end{aligned} \tag{1}$$

(Although $\mathbf{w} = (-1, -1)^\top, b = 3$ is equivalent.) The distance γ from this line to the closest points is $1/\sqrt{2}$, so the margin is $2\gamma = \sqrt{2}$.

1.2 SVM by optimisation

The primal formulation of the problem is

$$\begin{aligned} \arg\min_{\mathbf{w}} \ & \frac{1}{2}\mathbf{w}^\top \mathbf{w} \\ & \text{subject to} \\ 2w1 + 2w2 + b \geq & 1 \\ 4w1 + 4w2 + b \geq & 1 \\ 4w1 + b \geq & 1 \\ -b \geq & 1 \\ -2w1 - b \geq & 1 \\ -2w2 - b \geq & 1 \end{aligned} \tag{2}$$

In order to feed this into Matlab's `quadprog` function, we include b by optimising over the three-dimensional vector $\mathbf{x} = (w_1, w_2, b)^\top$ instead of just $\mathbf{w}$ (It's also possible to find b analytically from the constraints, and then re-write the problem to solve for only $\mathbf{w}$ with b known).

From `help quadprog` (but using `c` instead of `b` to avoid confusion):

```
min 0.5*x'*H*x + f'*x subject to:  A*x <= c
```
(3)

The input matrices `H` and `A` and vectors `f` and `c` are constructed by adapting our problem to this form. Notably, `H` has no entries in row or column 3, and `f` is all-zero.

The optimal solution is the same as the one found by inspection; see Eq. 1.

The dual problem

The dual problem, in general, is

$$\begin{aligned} &\arg\max_{\boldsymbol{\alpha}} \sum_{n=1}^{N} -\frac{1}{2} \sum_{m,n=1}^{N} \alpha_n \alpha_m t_n t_m \mathbf{x}_n^\top \mathbf{x}_m \\ &\text{subject to} \\ &\sum_{n=1}^{N} \alpha_n t_n = 0 \\ &\alpha_n \geq 0 \end{aligned} \tag{4}$$

The particulars of the problem appears in the data points $\mathbf{x}_i$, and we can re-write the sum they appear in on the form $\boldsymbol{\alpha}^\top \mathbf{H} \boldsymbol{\alpha}$, where $\mathbf{H}$ is a matrix found from the data points and their labels t_i.

The problem can be formulated on the standard form and fed into `quadprog`, which yields a solution where α_2 and α_4 are both 0. This means these two values are not support vectors, but the other four (i.e. the ones with non-zero α values) are.

Matlab and code

Despite being listed in the "Theoretical" section of the homework, this task required the use of Matlab (or possibly some other tool for solving quadratic programming problems).

The difference between this and the practical problems is that (a) you don't need to hand in your code, and (b) a solution which sets up the problem very clearly, but then fails to deliver the last piece that is the result from Matlab, would get more points here than a similar solution to a "practical" problem.

2 Practical problems

This solution sketch does not include a solution for the practical coding part. However, I do include some indication for the expected results. Since several things can be done differently—for example using autoscaling or not, using `svmtrain` or `fitcsvm`, choice of C—many groups get results other than these but still get full marks.

2.1 SVMs

With `svmtrain` using $C = 1$, and autoscaling turned off, the expected result is a bias of $b = -2.4529$ and a margin of $2\gamma = 2 * \frac{1}{||\mathbf{w}||} = 2 * 0.4994 = 0.9986$.

With higher C, the margin becomes smaller as wel as "harder" i.e. there are few points inside the margin, and fewer misclassifications. For very high values, the number of support vectors go down to the minimum of 3, each lying exactly on the margin. For very low values of C, conversely all data points in the entire set become support vectors.

2.2 Kernels

An SVM using either the RBF and Quadratic kernels should be able to classify 100% of the points correctly. The linear kernel achieves close to 50% correctness, since there is no way to draw a straight line to separate the inner and outer circles that the points essentially lie on.

The SMO optimisation method should be noticeably faster than the QP one for the non-linear kernels. With either opimisation method, the RBF kernel is faster than the quadratic one.

Comments

This is not a complete solution; only a sketch. If students were to hand in a report with as many steps omitted as this one, they would not get full marks.

The purpose of this document is to show students whether their solution was along the right lines. Firstly, this allows them to improve their understanding, if they had not solved the problem correctly; secondly, it gives a hint as to whether they can expect to score points for each task. (Of course, eventually each submission will be graded, and the actual number of points reported to the student.)

2.3 Jonatan's grading policy

If a student submission (a) explains its reasoning and (b) is correct, it will receive full marks.

A correct answer presented poorly can be grounds for deductions, but never for more than half of the point total on the task.

If the answer is incorrect, at least 1 point will be deducted. I will try to follow the reasoning and find where the mistake occurred. Depending on how severe the mistake was, and how good the reasoning was before and after the mistake(s), the deduction might be only the 1 point, or anything up to all the points for that task.

For tasks with more than one sub-problem (a, b, c...) my policy is that a correct answer to one sub-problem guarantees at least 1 point, but an incorrect answer to one of them rules out getting the maximum points for the task. In between those two extremes, each sub-problem does not correspond to an exact number of points; rather, the solution is considered as a whole.

2.3.1 Code

I do not concern myself very much with coding style. If (1) the answers in the report are correct, (2) the code works when I run it, and (3) I don't have reason to suspect that code has been copied, then the quality of the code won't further affect the grade.

If there is a mistake or incorrect answer, clear code might make it easy for me to realise that it was only a typo, and deduct fewer points.

If the answers are correct but I can't see how the code provides them (e.g. it won't run), I will deduct up to half of the task's points.