so we obtain

$$b = w^\top (u_{i_0} + v_{j_0})/2.$$

For improved numerical stability, we can average over the sets of indices defined as $I_{\lambda>0} = \{i \in \{1, \ldots, p\} \mid \lambda_i > 0\}$ and $I_{\mu>0} = \{j \in \{1, \ldots, q\} \mid \mu_j > 0\}$. We obtain

$$b = w^\top \left(\Big(\sum_{i \in I_{\lambda>0}} u_i \Big) / |I_{\lambda>0}| + \Big(\sum_{j \in I_{\mu>0}} v_j \Big) / |I_{\mu>0}| \right) /2.$$

The `Matlab` programs implementing the above method are given in Appendix B, Section B.1. This should convince the reader that there is very little gap between theory and practice, although it is quite consuming to tune the tolerance parameters needed to deal with floating-point arithmetric.

Figure 52.5 shows the result of running the `Matlab` program implementing the above method using ADMM on two sets of points of 50 points each generated at random using the following `Matlab` code.

```
u14 = 10.1*randn(2,50)+18;
v14 = -10.1*randn(2,50)-18;
```

The function `SVMhard2` is called with $\rho = 10$ as follows

```
[lamb,mu,w] = SVMhard2(10,u14,v14)
```

and produces the output shown in Figure 52.5. Observe that there is one blue support vector and two red support vectors.

52.8 Applications of ADMM to ℓ^1-Norm Problems

Another important application of ADMM is to ℓ^1-norm minimization problems, especially lasso minimization, discussed below and in Section 55.4. This involves the special case of ADMM where $f(x) = \tau \|x\|_1$ and $A = I$. In particular, in the one-dimensional case, we need to solve the minimization problem: find

$$x^* = \arg\min_x \left(\tau|x| + (\rho/2)(x - v)^2 \right),$$

with $x, v \in \mathbb{R}$, and $\rho, \tau > 0$. Let $c = \tau/\rho$ and write

$$f(x) = \frac{\tau}{2c} \left(2c|x| + (x - v)^2 \right).$$

Minimizing f over x is equivalent to minimizing

$$g(x) = 2c|x| + (x - v)^2 = 2c|x| + x^2 - 2xv + v^2,$$