1) Consider the (univariate) regression task defined by the data

x	-5	-4	-3	-2	-1	0	1	2	3	4	5
y	-284	-168	-88	-38	-12	-4	-8	-18	-28	-32	-24

Write down the dual formulation of SVR (the one allowing for the kernel trick). Then, write a `Matlab` code that implements the training of the SVR and succinctly describe it (the fundamental steps, leaving aside the unnecessary details). Train the SVR with Gaussian kernel using the code with hyperparameters $C = 100$, $\epsilon =$ `1e-6` and $\sigma = 0.5$. Knowing that the data is generated by the polynomial $p^3 - 6p^2 + p - 4$, run the trained SVR on the interval $[-5, 5]$ with small stepsize (say, `-5 : 0.01 : 5`) and compare graphically and/or algebraically the prediction with the underlying original data commenting on its quality.

SOLUTION

The dual formulation of SVR is

$$\begin{array}{lll} \max & \sum_{i \in I} y^i \alpha_i - \varepsilon \sum_{i \in I} v_i - \frac{1}{2} \sum_{i \in I} \sum_{j \in I} \alpha_i \kappa(x^i , x^j) \alpha_j & \\ & \sum_{i \in I} \alpha_i = 0 & \\ & v_i \geq \alpha_i \ , \ v_i \geq -\alpha_i & i \in I \\ & -C \leq \alpha_i \leq C & i \in I \end{array}$$

where $I = \{ 1, \dots 11 \}$ is the set of indices of data points, and $\kappa(\cdot , \cdot)$ is the kernel function. For the Gaussian kernel,

$$\kappa(x , z) = e^{- \| x - z \|^2 / (2\sigma^2)}$$

Solving the dual formulation requires either an ad-hoc library with MATLAB interface, like LIBSVM, or an optimization library capable of solving convex Quadratic Programs, possibly with a modelling system as the interface. Using the YALMIP system the above formulation can be easily implemented as

```
alpha = sdpvar( n , 1 );
v = sdpvar( n , 1 );

F = [ - C <= alpha <= C ];
F = F + [ v >= alpha , v >= - alpha ];
F = F + [ ( ones( 1 , n ) * alpha == 0 ):'simplex' ];

c = (1/2) * alpha' * H * alpha - y' * alpha + ( eps * ones( 1 , n ) ) * v;
```

where `n = 11` and `H` is the Gram matrix, computed (for the Gaussian kernel) by

```
H = zeros( n , n );
s2 = 2 * sigma * sigma;
for i = 1 : n
    H( i , i ) = 1;
    for j = 1 : i - 1
        H( i , j ) = exp( - norm( X( i ) - X( j ) )^2 / s2 );
        H( j , i ) = H( i , j );
    end
end
```

Then, the optimization can be obtained by just calling

```
 optimize( F , c , ops );
```

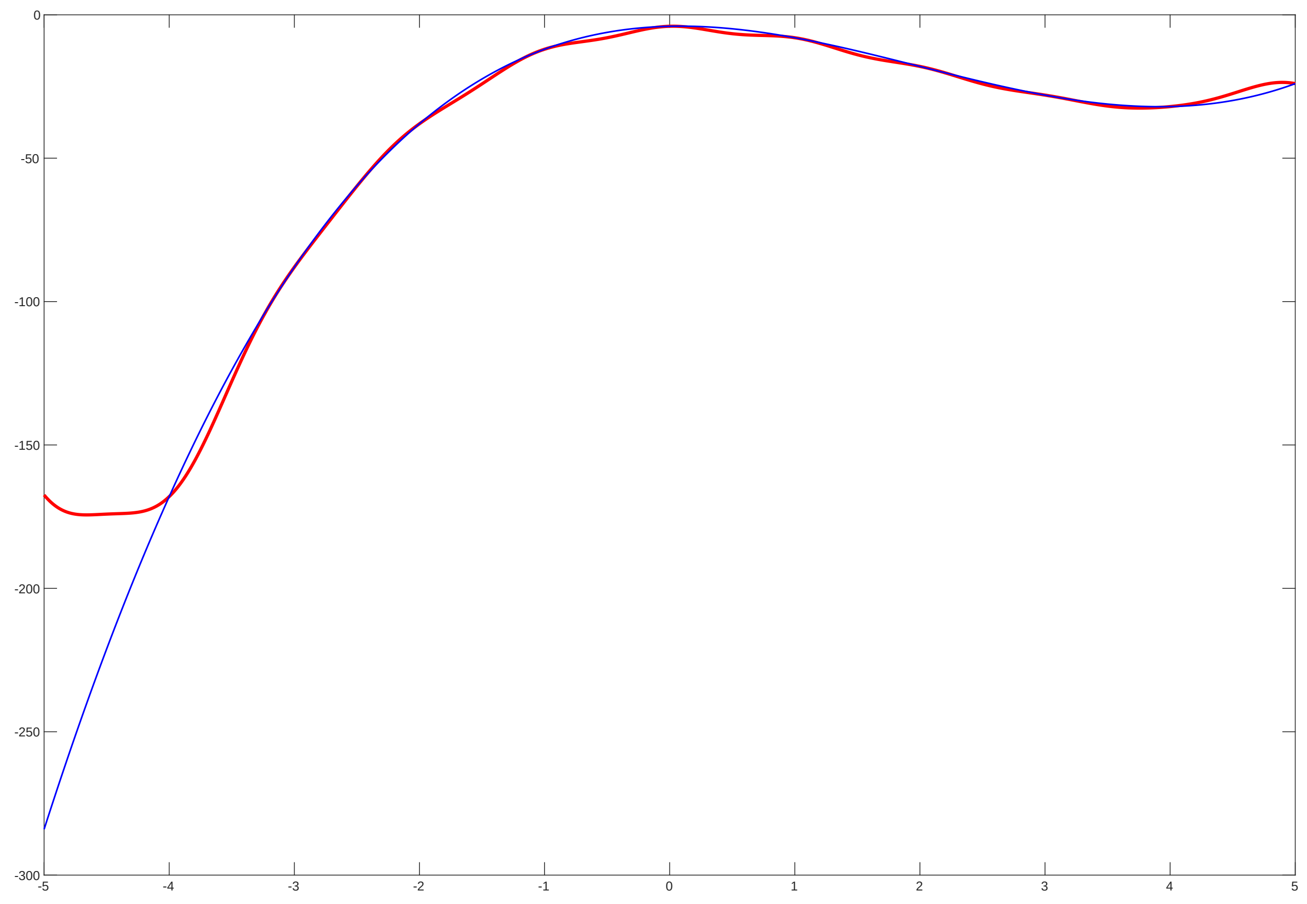
(having specified in `ops` a proper solver, e.g., `QUADPROG`), and the optimal solution can be retrieved by

```
 alpha = value( alpha );
 b = dual( F( 'simplex' ) );
```

Once this is done, the prediction `v` at a point `x` is computed by

```
v = b;
for i = 1 : n
    v = v + alpha( i ) *  exp( - norm( x - X( i ) )^2 / s2 );
end
```

This could in principle be made a bit more efficient by only using the support vectors, i.e., the indices such that `alpha( i ) > 0`, but in this case all original points should turn out to be support vectors. Doing so for all the points in `-5 : 0.01 : 5` and plotting the results against the original polynomial should produce a picture like the one below:



The SVR thereby produces a good interpolation of the real polynomial, except towards the left endpoint of the interval. By changing the hyperparameters better (or much worse) results may be obtained.

2) Solve the box-constrained quadratic optimization problem

$$\min\left\{ x^TQx/2 + qx \,:\, 0 \le x \le u \right\} \quad \text{with data} \quad Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1001 & -999 \\ 0 & -999 & 1001 \end{bmatrix} \quad , \quad q = \begin{bmatrix} -4 \\ -1 \\ -1 \end{bmatrix} \quad , \quad u = \begin{bmatrix} 3 \\ 4 \\ 3 \end{bmatrix}$$

using the projected gradient method (the "standard form projecting over the tangent cone of the active constraints) starting from the point "in the middle of the box" $x^0 = u/2$. Write a `Matlab` code that implements the algorithm and succinctly describe it (the fundamental steps, leaving aside the unnecessary details), then run it on the instance above and report the obtained (approximate) solution (detailing the stopping condition and what tolerances have been used in it). Optionally comment on the number of iterations and the convergence rate of the approach.

SOLUTION

Starting from the initial iterate `x = u / 2`, the projected gradient algorithm initially computes the descent direction as the anti-gradient, after which it is then projected over the (tangent cone of the) active constraint. This is especially simple in the box-constrained case in that it boils down to remove any negative component corresponding to an active lower bound constraint and any positive component corresponding to an active upper bound constraint:

```
d = - ( Q * x + q );
d( u - x <= 1e-12 & d > 0 ) = 0;
d( x <= 1e-12 & d < 0 ) = 0;
```

The norm of the projected (anti-)gradient being zero is the stopping condition; in practice one should always use some tolerance, which is here taken as `1e-6`. One can then compute the unconstrained minimum along it with the usual formula

```
alpha = ( - g' * d ) / d' * Q * d;
```

(where in general one should account for the possibility of the denumerator is zero, but again this is not happening in our instance). However, such a step may violate some box constraint, hence one must first compute the maximum step that retains feasibility, i.e., the minimum among the steps that make each constraint active. Again, due to the trivial form of the constraints one should only consider the upper bound constraint $x_i \le u_i$ if $d_i > 0$ and the lower bound one $x_i \ge 0$ if $d_i < 0$, i.e.,

```
ind = d > 0;
maxt = min( ( u( ind ) - x( ind ) ) ./ d( ind ) );
ind = d < 0;
maxt = min( [ maxt min( - x( ind ) ./ d( ind ) ) ] );
```

Then, the next iterate

```
x = x + min( [ alpha maxt ] * d;
```

is guaranteed to be feasible and the process can be repeated, eventually leading to convergence. With a tolerance of `1e-6` the process should take around 200 iterations and return a solution close to

```
x = [ 3.0000 ; 0.5000 ; 0.5000 ] , v = -8.0000
```

Indeed, this point has zero projected gradient, in that

```
Q * x + q = [ -1.0000 ; 0.0000 ; 0.0000 ]
```

but the constraint $x_1 \le 3$ is active, which means that the first component of the anti-gradient (1) is zeroed as well from the projection.

For the optional part one can notice that Q is separable: the first variable has no cross-product terms with the second and third. Thus, the optimization is basically separable in the two groups of variables. The eigenvalues of the 2×2 block are 2 and 2000, which gives the convergence rate in the unconstrained case

$$[(2000 - 2)/(2000 + 2)]^2 \approx 0.996$$

Now, the second and third components of the optimal point are in the interior of the feasible region; basically, one should expect the algorithm to run as an unconstrained one for those. Since the initial function value is 122.75 and the final is 8, $a^0 = f(\,x^1\,) - f_* = 114.75$. The expected number of iterations is

$$k \ge [\,1\,/\,\log(\,1\,/\,r\,)\,]\log(\,a^0\,/\,\epsilon\,)$$

where ϵ is the desired final error. Using this formula with $\epsilon =$ `1e-6` one obtains an estimate of roughly 4500 iterations, while in practice the algorithm takes much less than that, partly due to a number of approximations done during the analysis: the value of a^0 also takes into account the first component, the stopping condition with tolerance `1e-6` does not directly measure the gap, the algorithm is not exactly the same as what it would be if it ran on the last two components alone, and so on. Besides, the estimate in the formula is worst-case, and faster convergence is always possible.

3) Consider the following unconstrained multiobjective optimization problem:

$$\begin{cases} \min\ (x_1^4 + 2x_2^2,\ -x_1 + x_2^2) \\ (x_1, x_2) \in \mathbb{R}^2 \end{cases}$$

(a) Prove that the problem admits a Pareto minimum point.

(b) Find the set of all weak Pareto minima.

(c) Find a suitable subset of Pareto minima.

SOLUTION

(a) It is enough to notice that the objective function f_1 admits a unique global minimum point, which, therefore, is a (Pareto) minimum point for the given problem.

(b) - (c) We preliminarly observe that the problem is convex, since the objective and the constraint functions are convex. Therefore the set of weak minima coincides with the set of solutions of the scalarized problems (P_{α_1}), where $0 \le \alpha_1 \le 1$, i.e.

$$\begin{cases} \min\ \alpha_1(x_1^4 + 2x_2^2) + (1-\alpha_1)(-x_1 + x_2^2) =: \psi_{\alpha_1}(x) \\ (x_1, x_2) \in \mathbb{R}^2 \end{cases}$$

We note that (P_{α_1}) is convex and differentiable; then the following system provides a necessary and sufficient condition for an optimal solution of (P_{α_1}):

$$\begin{cases} 4\alpha_1 x_1^3 - 1 + \alpha_1 = 0 \\ 2\alpha_1 x_2 + 2x_2 = 0 \\ 0 \le \alpha_1 \le 1, \end{cases}$$

We obtain:

$$\begin{cases} x_1 = \sqrt[3]{\dfrac{1-\alpha_1}{4\alpha_1}},\ \alpha_1 \neq 0 \\ x_2 = 0 \\ 0 \le \alpha_1 \le 1, \end{cases}$$

Notice that for $\alpha_1 = 0$ the previous system is impossible.

Then, $Weak\ Min(P) = \{(x_1, x_2) : x_1 = \sqrt[3]{\dfrac{1-\alpha_1}{4\alpha_1}},,\ x_2 = 0, 0 < \alpha_1 \le 1\}$, i.e.,

$$Weak\ Min(P) = \{(x_1, x_2) : x_1 \ge 0,\ x_2 = 0\}.$$

Moreover,

$$Min(P) \supseteq \{(x_1, x_2) : x_1 = \sqrt[3]{\frac{1-\alpha_1}{4\alpha_1}},\ x_2 = 0, 0 < \alpha_1 < 1\}.$$

Since, for $\alpha_1 = 1$, we obtain the point $(0, 0)$ which is a minimum point, as we have already observed, in conclusion:

$$Weak\ Min(P) = Min(P) = \{(x_1, x_2) : x_1 \ge 0,\ x_2 = 0\}.$$

4) Consider the following bimatrix game:

$$C_1 = \begin{pmatrix} 4 & 3 & 2 \\ 2 & 1 & 5 \end{pmatrix} \qquad C_2 = \begin{pmatrix} 5 & 4 & 3 \\ 7 & 2 & 6 \end{pmatrix}$$

(a) Find the set of pure strategies Nash equilibria, if any. Alternatively, show that no pure strategies Nash equilibrium exists.

(b) Find a mixed strategies Nash equilibrium.

SOLUTION

(a) Strategy 1 of Player 2 is dominated by Strategy 3, so that column 1 in the two matrices can be deleted. The reduced game is given by the matrices

$$C_1^R = \begin{pmatrix} 3 & 2 \\ 1 & 5 \end{pmatrix} \qquad C_2^R = \begin{pmatrix} 4 & 3 \\ 2 & 6 \end{pmatrix}$$

Now, it is simple to show that (2,2) and (1,3) are pure strategies Nash equilibria. Indeed, the minima on the columns of C_1^R, (i.e., 1 and 2), are obtained in correspondence of the minima on the rows of C_2^R, (i.e., 2 and 3) and are related to the components $(C_1)_{22}$ $(C_1)_{13}$ of the given matrices C_1 and C_2.

This will also be shown in part (b) in the wider context of mixed strategies Nash equilibria.

(b) Consider the reduced game obtained in (a). The optimization problem associated with Player 1 is

$$\begin{cases} \min\ x^T C_1^R y = (3x_1 + x_2)y_2 + (2x_1 + 5x_2)y_3 \\ x_1 + x_2 = 1 \\ x_1, x_2 \geq 0 \end{cases} \equiv \begin{cases} \min\ (5y_2 - 3)x_1 - 4y_2 + 5 \\ 0 \leq x_1 \leq 1 \end{cases} \qquad (P_1(y_2))$$

Then, the best response mapping associated with $P_1(y_2)$ is:

$$B_1(y_2) = \begin{cases} 0 & \text{if } y_2 \in (3/5, 1] \\ [0,1] & \text{if } y_2 = 3/5 \\ 1 & \text{if } y_2 \in [0, 3/5) \end{cases}$$

Similarly, the optimization problem associated with Player 2 is

$$\begin{cases} \min\ x^T C_2^R y = x_1(4y_2 + 3y_3) + x_2(2y_2 + 6y_3) \\ y_1 + y_2 = 1 \\ y_1, y_2 \geq 0 \end{cases} \equiv \begin{cases} \min\ (5x_1 - 4)y_2 - 3x_1 + 6 \\ 0 \leq y_2 \leq 1 \end{cases} \qquad (P_2(x_1))$$

Then, the best response mapping associated with $P_2(x_1)$ is:

$$B_2(x_1) = \begin{cases} 0 & \text{if } x_1 \in (4/5, 1] \\ [0,1] & \text{if } x_1 = 4/5 \\ 1 & \text{if } x_1 \in [0, 4/5) \end{cases}$$

The couples (x_1, y_2) such that $x_1 \in B_1(y_2)$ and $y_2 \in B_2(x_1)$ are

1. $x_1 = 0, y_2 = 1$,
2. $x_1 = \frac{4}{5}, y_2 = \frac{3}{5}$,
3. $x_1 = 1, y_2 = 0$,

so that, recalling that $y_1 = 0$,

- $(x_1, x_2) = (0, 1)$, $(y_1, y_2, y_3) = (0, 1, 0)$, is a pure strategies Nash equilibrium,
- $(x_1, x_2) = (\frac{4}{5}, \frac{1}{5})$, $(y_1, y_2, y_3) = (0, \frac{3}{5}, \frac{2}{5})$, is a mixed strategies Nash equilibrium,
- $(x_1, x_2) = (1, 0)$, $(y_1, y_2, y_3) = (0, 0, 1)$, is a pure strategies Nash equilibrium.