hw7

Tom LOU

July 2024

1 Q1

For Support Vector Regression (SVR),

$$w_0 = y_j - \epsilon - \sum_{i \in S} (\lambda_i^+ - \lambda_i^-) \kappa(x_i, x_j)$$

In SVR, the goal is to find a function

$$f(x) = w^\top x + w_0$$

that approximates the relationship between input x and output y. The optimization problem can be formulated as:

$$\min_{w, w_0, \xi, \xi^*} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} (\xi_i + \xi_i^*)$$

subject to

$$y_i - (w^\top x_i + w_0) \leq \epsilon + \xi_i$$
$$(w^\top x_i + w_0) - y_i \leq \epsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \geq 0$$

The dual formulation of this optimization problem involves Lagrange multipliers α_i and α_i^*. The Lagrangian for this problem is:

$$L = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} (\xi_i + \xi_i^*) - \sum_{i=1}^{n} \lambda_i (\xi_i) - \sum_{i=1}^{n} \lambda_i^* (\xi_i^*)$$

$$- \sum_{i=1}^{n} \alpha_i [y_i - (w^\top x_i + w_0) - \epsilon - \xi_i] - \sum_{i=1}^{n} \alpha_i^* [(w^\top x_i + w_0) - y_i - \epsilon - \xi_i^*]$$

Setting the derivatives of L with respect to w, w_0, ξ_i, and ξ_i^* to zero, we get:

$$w = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) x_i$$

$$\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0$$

The dual problem becomes:

$$\max_{\alpha_i, \alpha_i^*} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\kappa(x_i, x_j) + \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)y_i - \epsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*)$$

subject to

$$0 \leq \alpha_i \leq C, \quad 0 \leq \alpha_i^* \leq C$$

$$\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0$$

For support vectors x_j on the upper edge of the ϵ-tube, we have the condition:

$$y_j = f(x_j) + \epsilon = w^\top x_j + w_0 + \epsilon$$

Rewriting $f(x_j)$ using the dual representation:

$$f(x_j) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\kappa(x_i, x_j) + w_0$$

So, the equation becomes:

$$y_j = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\kappa(x_i, x_j) + w_0 + \epsilon$$

Solving for w_0:

$$w_0 = y_j - \epsilon - \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\kappa(x_i, x_j)$$

Since α_i and α_i^* are non-zero only for support vectors, we can restrict the sum to the set S of support vectors:

$$w_0 = y_j - \epsilon - \sum_{i \in S}(\alpha_i - \alpha_i^*)\kappa(x_i, x_j)$$

By defining $\lambda_i^+ = \alpha_i$ and $\lambda_i^- = \alpha_i^*$, we get:

$$w_0 = y_j - \epsilon - \sum_{i \in S}(\lambda_i^+ - \lambda_i^-)\kappa(x_i, x_j)$$

2 Q2

Part (a)

We start with an initial weight matrix $W(0)$ which is a 2×3 zero matrix:

$$W(0) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The learning rate $\eta(i)$ is 0.5. The one-hot encoded target vectors are:

$$y_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, \quad y_2 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}, \quad y_3 = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}$$

The input vectors are:

$$x_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad x_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad x_3 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

Let's go through the updates for the first few iterations.

First Iteration

For x_1 with target y_1:

$$W(1) = W(0) + \eta x_1 y_1^\top$$

Calculation:

$$W(1) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + 0.5 \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & -1 \end{bmatrix}$$

$$W(1) = \begin{bmatrix} 0.5 & -0.5 & -0.5 \\ 0.5 & -0.5 & -0.5 \end{bmatrix}$$

For x_2 with target y_2:

$$W(2) = W(1) + \eta x_2 y_2^\top$$

Calculation:

$$W(2) = \begin{bmatrix} 0.5 & -0.5 & -0.5 \\ 0.5 & -0.5 & -0.5 \end{bmatrix} + 0.5 \begin{bmatrix} 1 \\ -1 \end{bmatrix} \begin{bmatrix} -1 & 1 & -1 \end{bmatrix}$$

$$W(2) = \begin{bmatrix} 0.5 & -0.5 & -0.5 \\ 0.5 & -0.5 & -0.5 \end{bmatrix} + \begin{bmatrix} -0.5 & 0.5 & -0.5 \\ 0.5 & -0.5 & 0.5 \end{bmatrix}$$

$$W(2) = \begin{bmatrix} 0 & 0 & -1 \\ 1 & -1 & 0 \end{bmatrix}$$

For x_3 with target y_3:

$$W(3) = W(2) + \eta x_3 y_3^\top$$

Calculation:

$$W(3) = \begin{bmatrix} 0 & 0 & -1 \\ 1 & -1 & 0 \end{bmatrix} + 0.5 \begin{bmatrix} -1 \\ -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & 1 \end{bmatrix}$$

$$W(3) = \begin{bmatrix} 0 & 0 & -1 \\ 1 & -1 & 0 \end{bmatrix} + \begin{bmatrix} 0.5 & 0.5 & -0.5 \\ 0.5 & 0.5 & -0.5 \end{bmatrix}$$

$$W(3) = \begin{bmatrix} 0.5 & 0.5 & -1.5 \\ 0.5 & -1.5 & 0.5 \end{bmatrix}$$

Part (b)

We will update each column of W separately as individual perceptrons.

First Epoch

For x_1 with target y_1:

$$e_1 = y_1 - \text{sign}(W^\top x_1)$$

Calculation:

$$W(0) = \begin{bmatrix} w_1(0) & w_2(0) & w_3(0) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$w_1(1) = w_1(0) + \eta e_1 x_1$$
$$w_2(1) = w_2(0) + \eta e_2 x_1$$
$$w_3(1) = w_3(0) + \eta e_3 x_1$$

The errors e_i are computed based on the current weights and input:

$$e_1 = 1 - \text{sign}(0) = 1$$
$$e_2 = -1 - \text{sign}(0) = -1$$
$$e_3 = -1 - \text{sign}(0) = -1$$

Updating the weights:

$$w_1(1) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 0.5 \times 1 \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

$$w_2(1) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 0.5 \times (-1) \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}$$

$$w_3(1) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 0.5 \times (-1) \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}$$

For x_2 with target y_2:
Calculation of errors e_i:

$$e_1 = -1 - \text{sign}(w_1(1)^\top x_2) = -1 - \text{sign}(0.5 \times 1 + 0.5 \times (-1)) = -1 - 0 = -1$$

$$e_2 = 1 - \text{sign}(w_2(1)^\top x_2) = 1 - \text{sign}(-0.5 \times 1 + (-0.5) \times (-1)) = 1 - 0 = 1$$

$$e_3 = -1 - \text{sign}(w_3(1)^\top x_2) = -1 - \text{sign}(-0.5 \times 1 + (-0.5) \times (-1)) = -1 - 0 = -1$$

Updating the weights:

$$w_1(2) = w_1(1) + 0.5 \times (-1) \times \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$w_2(2) = w_2(1) + 0.5 \times 1 \times \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix} + \begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

$$w_3(2) = w_3(1) + 0.5 \times (-1) \times \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix} - \begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

For x_3 with target y_3:
Calculation of errors e_i:

$$e_1 = -1 - \text{sign}(w_1(2)^\top x_3) = -1 - \text{sign}(0 \times (-1) + 1 \times (-1)) = -1 - (-1) = 0$$

$$e_2 = -1 - \text{sign}(w_2(2)^\top x_3) = -1 - \text{sign}(0 \times (-1) + (-1) \times (-1)) = -1 - 1 = -2$$

$$e_3 = 1 - \text{sign}(w_3(2)^\top x_3) = 1 - \text{sign}(-1 \times (-1) + 0 \times (-1)) = 1 - 1 = 0$$

Updating the weights:

$$w_1(3) = w_1(2) + 0.5 \times 0 \times \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$w_2(3) = w_2(2) + 0.5 \times (-2) \times \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} - \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$$

$$w_3(3) = w_3(2) + 0.5 \times 0 \times \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

The classification stabilizes and matches the one-hot encoded targets, we have convergence.

3 q3

Input Layer

The input layer consists of 4 neurons, representing the four binary inputs (x_1, x_2, x_3, x_4).

Hidden Layer

The hidden layer will consist of neurons that perform the AND operations. These neurons can be modeled with weighted sums followed by an ReLU activation function .

Output Layer

The output layer will consist of a neuron that performs the OR operation on the results of the hidden layer. This is weighted sum followed by an activation function.

Network Structure

- Input Layer: 4 neurons
- Hidden Layer 1: 2 neurons performing AND operations
- Output Layer: 1 neuron performing OR operation

The weights and biases for the connections between the layers. We'll use w_{ij} to denote the weight from neuron j in the previous layer to neuron i in the current layer, and b_i to denote the bias for neuron i.

Step-by-Step Network Design

- Input Layer: 4 input neurons (x_1, x_2, x_3, x_4)
- Hidden Layer:
  - Neuron h_1: performs AND(x_1, x_2)
    - Weights: $w_{11} = 1$, $w_{12} = 1$
    - Bias: $b_1 = -1.5$ (to implement AND)
  - Neuron h_2: performs AND(x_3, x_4)
    - Weights: $w_{21} = 1$, $w_{22} = 1$
    - Bias: $b_2 = -1.5$ (to implement AND)
- Output Layer:
  - Neuron o_1: performs OR(h_1, h_2)
    - Weights: $w_{31} = 1$, $w_{32} = 1$
    - Bias: $b_3 = -0.5$ (to implement OR)

For the hidden layer neurons:

$$h_1 = f(w_{11}x_1 + w_{12}x_2 + b_1)$$
$$h_2 = f(w_{21}x_3 + w_{22}x_4 + b_2)$$

For the output layer neuron:

$$o_1 = f(w_{31}h_1 + w_{32}h_2 + b_3)$$

Here, f is an activation function of ReLU or sigmoid.
The weight matrices and bias vectors can be represented as:
Weights:

$$W_1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

Biases:

$$b_1 = \begin{bmatrix} -1.5 \\ -1.5 \end{bmatrix}$$

$$b_2 = \begin{bmatrix} -0.5 \end{bmatrix}$$

Activation Function

For simplicity, let's use a step function $f(x)$ such that:

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

Final Neural Network Design

- Input Layer: 4 neurons
- Hidden Layer: 2 neurons with weights W_1 and biases b_1
- Output Layer: 1 neuron with weights W_2 and biases b_2

This network structure can be used to distinguish between the given classes by performing logical operations. The AND operations in the hidden layer and the OR operation in the output layer enable the network to learn the patterns and classify the input accordingly.