CS577 - Machine Learning – Assignment 6

Alexandros Angelakis csdp1363, angelakis@csd.uoc.gr

Exercise 1

In Table 1 we can see the training data of a 1-norm soft margin SVM as well as the (fictional) Lagrange multipliers α that stem from training the model with cost $C = 10$. The kernel employed is the full polynomial quadratic of degree 2: $K(x, z) = (x \cdot z + 1)^2$

Sample	α	**y**	X_1	X_2
x_1	1	1	1	0
x_2	10	-1	0	-1
x_3	10	-1	1	1
x_4	0	1	-1	0
x_5	0	-1	0	-1

Table 1: Training data

1. Explain why the Lagrange multipliers cannot really be the solution to an 1-norm, soft-margin SVM problem

In a 1-norm soft-margin SVM, the Lagrange multipliers α_i should satisfy the constraints:

$$0 \leq \alpha_i \leq C$$

where $C = 10$.
A key problem is that the sum of weighted Lagrange multipliers should satisfy the Karush-Kuhn-Tucket (KKT) conditions, particularly:

$$\sum_i \alpha_i y_i = 0$$

For our given data:

$$\sum_i \alpha_i y_i = 1(1) + 10(-1) + 10(-1) + 0(1) + 0(-1) = 1 - 10 - 10 = -19$$

Thus, the multipliers violate the KKT conditions, indicating that they cannot be the actual solution of an optimal 1-norm soft-margin SVM.

2. How are the features in feature space related to the input variables X_i

In order to answer that question, we first need to find the feature space Φ. Our data are 2D, so we can find Φ by using the kernel $K(x, z)$:

$$\begin{aligned} K(x,z) &= (x \cdot z + 1)^2 = (X_1 Z_1 + X_2 Z_2 + 1)^2 = (X_1 Z_1 + X_2 Z_2 + 1)(X_1 Z_1 + X_2 Z_2 + 1) = \\ &= X_1^2 Z_1^2 + X_1 Z_1 X_2 Z_2 + X_1 Z_1 + X_2 Z_2 X_1 Z_1 + X_2^2 Z_2^2 + X_2 Z_2 + X_1 Z_1 + X_2 Z_2 + 1 = \\ &= X_1^2 Z_1^2 + X_2^2 Z_2^2 + 2 X_1 Z_1 X_2 Z_2 + 2 X_1 Z_1 + 2 X_2 Z_2 + 1 \end{aligned}$$

We can see that if we take the dot product of the vector:

$$\vec{v} = (1, \sqrt{2} X_1, \sqrt{2} X_2, \sqrt{2} X_1 X_2, X_1^2, X_2^2)$$

with the vector:

$$\vec{u} = (1, \sqrt{2} Z_1, \sqrt{2} Z_2, \sqrt{2} Z_1 Z_2, Z_1^2, Z_2^2)$$

we get the kernel $K(x,z) = u \cdot v$ So the feature space Φ is:

$$\Phi(x) = (1, \sqrt{2}X_1, \sqrt{2}X_2, \sqrt{2}X_1X_2, X_1^2, X_2^2)$$

Thus, each input vector is mapped into a six-dimensional feature space consisting of quadratic and linear terms of the input variables, as well as some interaction and bias terms.

3. What is the weight vector w and the intercept term b that defines the decision surface f($x_{\textbf{test}}$) = $sign(w \cdot x_{\textbf{test}} + b)$.

We know from the dual of the SVM Formulation that

$$w = \sum_i a_i y_i \Phi(x_i)$$

As we can see from Table 1, both sample x_4 and x_5 have $\alpha = 0$, which means these two samples will not contribute in calculating the weight vector w. So we can write w now as:

$$w = \alpha_1 y_1 \Phi(x_1) + \alpha_2 y_2 \Phi(x_2) + \alpha_3 y_3 \Phi(x_3)$$

We need to calculate the vectors $\Phi(x_1)$, $\Phi(x_2)$ and $\Phi(x_3)$. This step is straightforward, we just substitute the values of X_1 and X_2 into the feature space Φ for each sample. Thus, we get the following results:

$$\Phi(x_1) = (1, \sqrt{2}, 0, 0, 1, 0)$$

$$\Phi(x_2) = (1, 0, -\sqrt{2}, 0, 0, 1)$$

$$\Phi(x_3) = (1, \sqrt{2}, \sqrt{2}, \sqrt{2}, 1, 1)$$

After multiplying each vector with its corresponding class and Lagrangian multiplier, we get our final weight vector w:

$$\boxed{\vec{w} = (-19, -9\sqrt{2}, 0, -10\sqrt{2}, -9, -20)}$$

Calculating the bias term b is a more complicated process, we need to solve the equation:

$$\alpha_j (y_j \sum_i \alpha_i y_i K(x_i, x_j) + b - 1) = 0, \quad \text{for any} \quad j \quad \text{with} \quad 0 < \alpha_j < C$$

If we take a look at our data table, we can see that 4 of our 5 samples have α that do not satisfy the constrain. So our only x_j is sample x_1. For x_1 we know that $\alpha = 1$ and $y = 1$. So the equation can be simplified:

$$\sum_i \alpha_i y_i K(x_i, x_j) + b = 1 \rightarrow b = 1 - \sum_i \alpha_i y_i K(x_i, x_j)$$

We need to calculate $K(x_i, x_j)$, meaning the calculating the kernel between each sample and x_1:

$$K(x_1, x_1) = ((1 * 1 + 0 * 0) + 1)^2 = (1 + 1)^2 = 4$$

$$K(x_2, x_1) = 1$$

$$K(x_3, x_1) = 4$$

$$K(x_4, x_1) = 0$$

$$K(x_5, x_1) = 0$$

Thus,

$$\sum_i \alpha_i y_i K(x_i, x_j) = -46$$

and

$$\boxed{b = 47}$$

4. Write the same classi cation function f without using w but only using the kernel

We can rewrite the classification function f using only the kernel as:

$$f(x_{\text{test}}) = sign(\sum_i \alpha_i y_i K(x_i, x_{\text{test}}) + b)$$

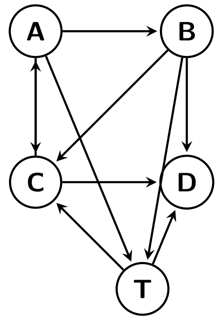
where $K(x_i, x_{\text{test}}) = (x_i \cdot x_{\text{test}} + 1)^2$

5. Describe as clearly as possible all the regions in feature space in comparison to the SVM margin where the input vectors x_i fall for $i = 1, \ldots, 5$.

For samples x_1, x_2 and x_3, their $\alpha > 0$, which means that they are support vectors. But, since for samples x_2 and x_3 their $\alpha = C = 10$, they lie inside the margin or they are misclassified (depending on the bias term b) and the sample x_1 is likely on the margin. Samples x_4 and x_5 lie outside the margin, meaning they are correctly classified with high confidence, since their $\alpha = 0$.

Exercise 2

Below we can see a Bayesian Network with 5 nodes and 10 edges, where each node has at least 2 neighbours and there are at least 3 colliders in the graph.



In this network:
- T has parents A, B.
- T has children C, D.
- T has spouses A, B.
- C, D and T are colliders.
- Each node has at least 2 neighbors.

1. Which variables will be returned by the feature selection algorithm that selects all variables dependent with T.

All the variables are dependent with T, because they are all connected to T through direct paths, making them unconditionally independent with T. Thus, the feature selection algorithm will return the variables {A, B, C, D}.

2. Show apossible trace of the Forward-Backward Feature Selection algorithm. In the trace show the conditional independence tests that the algorithm performs at each step, and the selections that it makes. Use the d-separation criterion to explain why the algorithm could have made the selections you indicate. (hint: create the network in such a way that the algorithm nishes in the minimal number of steps, or you'll never finish the computations).

Forward Phase:

1. **Initialization**:
   - Start with an empty set $S = \varnothing$.
   - Initialize the remaining variables: $R = \{A, B, C, D\}$.
2. **Step 1: Test** A:
   - Compute Pvalue$(T; A|\varnothing)$.
   - Since A is unconditionally dependent on T (A direct causes of T), Pvalue$(T; A|\varnothing) \leq \alpha$.
   - Add A to S: $S = \{A\}$.
   - Update R: $R = \{B, C, D\}$.
3. **Step 2: Test** B:
   - Compute Pvalue$(T; B|A)$.
   - Since B is dependent on T given A (B direct causes of T), Pvalue$(T; B|A) \leq \alpha$.
   - Add B to S: $S = \{A, B\}$.
   - Update R: $R = \{C, D\}$.
4. **Step 3: Test** C:

- Compute Pvalue$(T;C|A,B)$.
- Since C is dependent on T given A and B (T direct causes of C), Pvalue$(T;C|A,B) \leq \alpha$.
- Add C to S: $S = \{A,B,C\}$.
- Update R: $R = \{D\}$.

5. **Step 4: Test** D:
   - Compute Pvalue$(T;D|A,B,C)$.
   - Since D is dependent on T given A, B, and C (T direct causes of D), Pvalue$(T;D|A,B,C) \leq \alpha$.
   - Add D to S: $S = \{A,B,C,D\}$.
   - Update R: $R = \varnothing$.

Backward Phase:

1. **Initialization**:
   - Start with the full set of selected variables: $S = \{A,B,C,D\}$.
2. **Step 1: Test** A:
   - Compute Pvalue$(T;A|B,C,D)$.
   - Since A is dependent on T given B, C, and D, Pvalue$(T;A|B,C,D) \leq \alpha$.
   - **Result**: Keep A in S.
3. **Step 2: Test** B:
   - Compute Pvalue$(T;B|A,C,D)$.
   - Since B is dependent on T given A, C, and D, Pvalue$(T;B|A,C,D) \leq \alpha$.
   - **Result**: Keep B in S.
4. **Step 3: Test** C:
   - Compute Pvalue$(T;C|A,B,D)$.
   - Since C is dependent on T given A, B, and D, Pvalue$(T;C|A,B,D) \leq \alpha$.
   - **Result**: Keep C in S.
5. **Step 4: Test** D:
   - Compute Pvalue$(T;D|A,B,C)$.
   - Since D is dependent on T given A, B, and C, Pvalue$(T;D|A,B,C) \leq \alpha$.
   - **Result**: Keep D in S.

Final Selected Set: $S = \{A,B,C,D\}$.

3. Do the same for the Forward-Backward with Early Dropping algorithm with one run.

Forward Phase with Early Dropping:

1. **Initialization**:
   - Start with an empty set $S = \varnothing$.
   - Initialize the remaining variables: $R = \{A,B,C,D\}$.
2. **Step 1: Test** A:
   - Compute Pvalue$(T;A|\varnothing)$.
   - Since A is unconditionally dependent on T, Pvalue$(T;A|\varnothing) \leq \alpha$.
   - Add A to S: $S = \{A\}$.

- Update R: $R = \{B, C, D\}$.
- **Early Dropping**:
  - Check if any variable in R is independent of T given $S = \{A\}$:
    * Pvalue$(T; B|A) \leq \alpha$ (dependent).
    * Pvalue$(T; C|A) \leq \alpha$ (dependent).
    * Pvalue$(T; D|A) \leq \alpha$ (dependent).
  - **Result**: No variables are dropped.

3. **Step 2: Test** B:
   - Compute Pvalue$(T; B|A)$.
   - Since B is dependent on T given A, Pvalue$(T; B|A) \leq \alpha$.
   - Add B to S: $S = \{A, B\}$.
   - Update R: $R = \{C, D\}$.
   - **Early Dropping**:
     - Check if any variable in R is independent of T given $S = \{A, B\}$:
       * Pvalue$(T; C|A, B) \leq \alpha$ (dependent).
       * Pvalue$(T; D|A, B) \leq \alpha$ (dependent).
     - **Result**: No variables are dropped.

4. **Step 3: Test** C:
   - Compute Pvalue$(T; C|A, B)$.
   - Since C is dependent on T given A and B, Pvalue$(T; C|A, B) \leq \alpha$.
   - Add C to S: $S = \{A, B, C\}$.
   - Update R: $R = \{D\}$.
   - **Early Dropping**:
     - Check if any variable in R is independent of T given $S = \{A, B, C\}$:
       * Pvalue$(T; D|A, B, C) \leq \alpha$ (dependent).
     - **Result**: No variables are dropped.

5. **Step 4: Test** D:
   - Compute Pvalue$(T; D|A, B, C)$.
   - Since D is dependent on T given A, B, and C, Pvalue$(T; D|A, B, C) \leq \alpha$.
   - Add D to S: $S = \{A, B, C, D\}$.
   - Update R: $R = \varnothing$.
   - **Early Dropping**:
     - No variables remain in R.

Selected Set S: $S = \{A, B, C, D\}$.
Backward Phase:

1. **Initialization**:
   - Start with the full set of selected variables: $S = \{A, B, C, D\}$.

2. **Step 1: Test** A:
   - Compute Pvalue$(T; A|B, C, D)$.
   - Since A is dependent on T given B, C, and D, Pvalue$(T; A|B, C, D) \leq \alpha$.
   - **Result**: Keep A in S.

3. **Step 2: Test** B:

- Compute Pvalue$(T; B|A, C, D)$.
- Since B is dependent on T given A, C, and D, Pvalue$(T; B|A, C, D) \leq \alpha$.
- **Result**: Keep B in S.

4. **Step 3: Test** C:
   - Compute Pvalue$(T; C|A, B, D)$.
   - Since C is dependent on T given A, B, and D, Pvalue$(T; C|A, B, D) \leq \alpha$.
   - **Result**: Keep C in S.
5. **Step 4: Test** D:
   - Compute Pvalue$(T; D|A, B, C)$.
   - Since D is dependent on T given A, B, and C, Pvalue$(T; D|A, B, C) \leq \alpha$.
   - **Result**: Keep D in S.

Final Set: $S = \{A, B, C, D\}$.

4. Compare the execution of the two algorithms on the data from your network: which algorithm is computationally faster and which has better quality of results.

Computational Speed: - The Forward-Backward with Early Dropping algorithm is computationally faster because it avoids unnecessary conditional independence tests by dropping irrelevant variables early.
Quality of Results: - Both algorithms provide the same quality of results, selecting the same set of variables $\{A, B, C, D\}$. However, the Forward-Backward with Early Dropping algorithm achieves this more efficiently.