CS 4710: Artificial Intelligence (Fall'21)

P5: Machine Learning and Game Theory

Due Date: Tuesday 11/23 10:00 pm

Written Assignment (20 points)

As usual please go to the PDF in the written folder for the written instructions.

SVM (6 points)

1. Part A (2 points)
2. Part B (2 points)
3. Part C (2 points)

Linear Regression and Linear Program (4 points)

1. Part A (2 points)
2. Part B (2 points)

Nash Equilibrium (5 points)

1. Part A (2 points)
2. Part B (3 points)

Mechanism Design (5 points)

1. Part A (2 points)
2. Part B (3 points)

SVM (10 points)

In class you learned that learning SVM corresponds to solving a convex optimization problem. In this problem you will be implementing the constraints for this problem for 2-dimensional inputs as a quadratic program and running it through a quadratic program solver. You will need CVXOPT to run the solver. You can simply install it with

```
pip install cvxopt
```

In essence, you will be setting up matrices that represent the coefficients in the quadratic program. The user guide section on Quadratic Programming should prove useful. Additionally, the tutorial examples for creating matrices and solving a quadratic program will probably prove useful as well.

Question 1 (4 points): Hard Margin Classifier

First, you will implement the constraints for the hard margin problem. With a quadratic program, you are trying to minimize or maximize a quadratic function subject to a set of constraints. When looking at the optimization problem for an SVM, we see that after some manipulation, we are simply trying to minimize the norm of the line (weight vector) subject to a set of linear constraints. Thus, the objective function can be represented as the sum of squares of the weights. This can be represented as a set of matrices described in the quadratic programming user guide linked above. Note that the autograder uses the constraints shown on the slides. This means the bias should be added rather than subtracted (sometimes shown the other way). Note that the bias term is a variable in the optimization problem but doesn't contribute to the norm of the line. Additionally, you do not have to worry about being given an infeasible problem in this part and the next.

Question 2 (4 points): Soft Margin Classifier

Next, you will extend the constraints you set in the previous problem to include slack variables as well as a regularization term. You may ignore the square root in the calculation of the norm of the line when defining your objective function. This will give you a set of quadratic terms similar to what you had in the hard case, as well as a new set of linear terms that represent the slack variables. Note that each slack variable will be its own column/row in the objective function matrix. Feel free to consult other resources to try and understand the constraints, but make sure that the version you implement reflects the slides. For the regularization term, you should simply multiply it by your quadratic term matrix (P on the user guide).

Question 3 (2 points): Classify

For the last part, all you need to do is take the separator set by one of your two models and return the label for a given point. Note that your function should use the currently stored model whichcould have been set by either hard margin or soft margin. You can test your code by running autograder.py

Mechanism Design (10 points)

In this problem, you will design a mechanism for the situation where a seller sells a single item to one buyer. The buyer has a value over the item, which is drawn randomly from some discrete distribution. The buyer of course knows the realization of his value, but the seller only know the value distribution. We want to design a mechanism to allocate the item to this buyer and meanwhile ask for a payment from the buyer. Our goal is to maximize the expected payment from the buyer, which is also called the revenue of the mechanism. Detailed instructions for each function can be found in the code/MechanismDesign/mechanism.py file. Note you will need CVXOPT here as well. See more instructions about CVXOPT as described in the SVM problem.

Question 1 (4 points): Posted Price Mechanism

The first part of this problem asks you to implement the optimal Posted Price mechanism. Recall that in a posted price mechanism, the seller post some price x; the buyer will buy the item and pay x if his value is higher than or equal to x. Otherwise, the buyer will not buy the item and pay 0. In this problem, you task is to figure out the optimal posted price x to maximize the expected payment from the buyer.

Hint: Without loss of generality, your posted price can be one of the buyer's value because for any posted price x, satisfying $v_i < x < v_{i+1}$, resetting $x = v_{i+1}$ would only increase the revenue. Therefore, you can simply do an exhaustive search over the space of all possible buyer values to find out the optimal posted price.

Question 2 (6 points): Optimal Mechanism Design

In the second part, you will directly compute the optimal mechanism, among all possible mechanisms. This turns out to be a linear programming problem. In particular, any mechanism can be thought of as the following procedure. Seller first asks the buyer to report his value v, and then make the following two decisions: (1) an allocation decision variable $p(v)$ which is the probability of allocating the item to this buyer; (2) a payment decision variable $x(v)$ which is payment from the buyer. For example, the previous posted price mechanism, parameterized by price x, is a special case where $p(v) = 1$, $x(v) = x$ whenever $v \geq x$ and $p(v) = 0$, $x(v) = 0$ whenever $v < x$. More generally, you should have the freedom to pick any $p(v)$ and $x(v)$, but under two important constraints:

(1) the buyer should be incentivized to tell us his true value, i.e., $vp(v) - x(v) \geq v'p(v) - x(v)$ for any v' (called Incentive Compatibility, or IC); (2) the buyer is willing to participate the mechanism, i.e.,$v \cdot p(v) - x(v) \geq 0$ (called Individual Rationality or IR).

By solving the above optimization problem, which can be easily seen to be a linear program, you will be able to compute the optimal mechanism.

For both questions, we have multiple different test cases (i.e. buyer's valuation distributions), you will be tested based on whether you correctly computed the optimal mechanism for each test case. You can test your code by running autograder.py

Please submit only your svm.py and mechanism.py to Gradescope.