10-725 Optimization, Spring 2010: Homework 2 Solutions

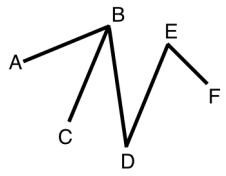
March 23, 2010

1 Vertex Cover [Sivaraman, 20 points]

The goal of this problem is to illustrate the use of LPs for approximating NP-hard optimization problems. We will obtain an approximation to the *vertex cover* problem.

Vertex Cover: Given an undirected graph $G = (V, E)$ and a cost function on vertices $c : V \longrightarrow Q^+$ (positive rationals), find a set of vertices $V' \subseteq V$ of minimum cost, such that every edge has at least one endpoint incident at V'.

For example, consider the following graph. Assume all the vertices have equal weight. Then the problem reduces to picking the minimum number of vertices such that each edge is covered. For this graph, the minimum cost vertex cover is {B,E}.



1. [1 pt] Associate a binary (0/1) variable x_i with each vertex i, which denotes whether the vertex has been picked in the vertex cover or not. Express the condition that each edge has at least one of its ends picked, as a linear constraint.

   ★ **Solution:** For every edge, we want at least one end point to be part of the vertex cover. Therefore, we get the constraint

   $$x_i + x_j \geq 1 \quad \forall (i,j) \in E$$

2. [1 pt] If all the vertices have equal weight, what is the objective function?

   ★ **Solution:** We are trying to find the minimum vertex cover. Thus, the objective fuction is

   $$\min_x \sum_{i \in V} x_i$$

3. [1 pt] If the vertices have different weights, what is the new objective function?

★ **Solution:**

$$\min_x \sum_{i \in V} c_i x_i$$

where c_i is the weight of the vertex v_i.

4. [2 pts] Write out an integer program (IP) that exactly solves the vertex cover problem.
Hint: An integer program is one in which the objective function and constraints are linear, but there are some variables that are constrained to take only integer values.

★ **Solution:**

$$\min \sum_{i \in V} c_i x_i$$

$$\begin{aligned} \text{subject to} \quad & x_i + x_j \geq 1 \quad \forall (i,j) \in E \\ & x_i \in \{0, 1\} \end{aligned}$$

The last constraint is what makes this optimization problem an integer program, instead of a linear program.

5. [2 pts] Integer Programming is an NP-hard problem. This means that there cannot exist any polynomial time algorithm for solving an integer program, unless $P = NP$. Therefore, we will try to get an approximate solution by obtaining an LP from the IP. This is known as an LP relaxation. To get an LP relaxation from an IP, we retain the objective function and constraints, except for the constraints that some variables have to take integer values. We replace these constraints by new constraints that say that the variables come from a continuous range. Obtain an LP relaxation of the above IP problem.

★ **Solution:** We relax the last (integrality) constraint in the previous formulation to obtain the following linear program

$$\min \sum_{i \in V} c_i x_i$$

$$\begin{aligned} \text{subject to} \quad x_i + x_j &\geq 1 \quad \forall (i,j) \in E \\ x_i &\geq 0 \\ x_1 &\leq 1 \end{aligned}$$

Since this is a minimization problem and all weights c_i are positive, the first and second constraint above make the last constraint redundant.

6. [6 pts] Prove half integrality of any extreme point solution for the above LP relaxation i.e. prove that in an optimal basic feasible solution, all variables take the value 0, 1/2 or 1.
Hint: Assume that a feasible solution x is not half integral. Then show that you can express this solution as a convex combination of two other feasible solutions y and z. The solutions y and z are obtained by slight perturbations of the variables in the solution x that are not half integral. Note that if the constraint in the LP corresponding to an edge is active, then even a slight perturbation of the variables might make the constraint unsatisfied. You must ensure that for any feasible solution, all the constraints are satisfied.

★ **Solution:** Suppose x is a feasible extreme point solution for the above LP, but it is not half integral. Thus, $\exists i \in V$ such that $x_i \notin \{0, \frac{1}{2}, 1\}$. Let us divide the vertices whose corresponding x-values are not half integral into 2 categories

$$V_- = \left\{ i \middle| 0 < x_i < \frac{1}{2} \right\}$$
$$V_+ = \left\{ i \middle| \frac{1}{2} < x_i < 1 \right\}$$

Since x is not half integral, $V_- \cup V_+$ is non-empty. Define ϵ, y and z as follows. (ϵ is half minimum distance to $\{0, \frac{1}{2}, 1\}$ for any non-half-integral vertex, and y and z are copies of x, perturbed by ϵ in opposite directions.)

$$\epsilon = \frac{1}{2} \min_i \min \begin{cases} x_i & \text{if } \ i \in V_- \\ \frac{1}{2} - x_i & \text{if } \ i \in V_- \\ x_i - \frac{1}{2} & \text{if } \ i \in V_+ \\ 1 - x_i & \text{if } \ i \in V_+ \end{cases}$$

$$y_i = \begin{cases} x_i + \epsilon & \text{if } \ i \in V_- \\ x_i - \epsilon & \text{if } \ i \in V_+ \\ x_i & \text{otherwise} \end{cases}$$

$$z_i = \begin{cases} x_i - \epsilon & \text{if } \ i \in V_- \\ x_i + \epsilon & \text{if } \ i \in V_+ \\ x_i & \text{otherwise} \end{cases}$$

We define ϵ above in such a way that even if we perturb the non half-integral values of x by epsilon, the constraints are still satisfied, as we will show below.

Notice that $x = \frac{y+z}{2}$. Since $V_- \cup V_+$ is non-empty, it means that x, y and z are distinct. Therefore, if we can show that y and z are feasible solutions, then it would imply that x is a convex combination of two other feasible solutions and hence not an extreme point solution.

Since we have chosen ϵ to be half of the minimum distance between a non half-integral value and $\{0, \frac{1}{2}, 1\}$, we are guaranteed that for $i \in V_- \cup V_+$

$$\begin{aligned} x_i - \epsilon &> 0 \\ x_i + \epsilon &> 0 \\ x_i - \epsilon &< 1 \\ x_i + \epsilon &< 1 \end{aligned}$$

Therefore, $y_i \geq 0$ and $z_i \geq 0 \ \forall \ i \in V$.

The other constraints are of the form $x_i + x_j \geq 1$. Since x is feasible, x satisfies all these constraints. Suppose this constraint is not tight i.e. $x_i + x_j > 1$. The lowest value that $y_i + y_j$ or $z_i + z_j$ can take is $x_i - \epsilon + x_j - \epsilon = x_i + x_j - 2\epsilon$.

Either both x_i and x_j are $\in [\frac{1}{2}, 1]$, or either x_i or x_j is $\in [0, \frac{1}{2}]$ and the other is in $\in [\frac{1}{2}, 1]$.

In the first case, neither x_i nor x_j could be reduced below $\frac{1}{2}$ (since, x_i and x_j are either $\frac{1}{2}$ or at least $\frac{1}{2} + 2\epsilon$ by the definition of ϵ. If x_i or x_j are $= \frac{1}{2}$ then they are untouched in y and z, otherwise they are each reduced by at most ϵ) and the constraint is still satisfied in y and z.

In the second case, assume without loss of generality that $x_i \in [0, \frac{1}{2}]$ and $x_j \in [\frac{1}{2}, 1]$. Note that since the constraint is not tight, $x_i \geq 2\epsilon$, since if x_i was 0 the constraint would have to be tight in x (since, x_j is at most 1). Similarly x_j can't be $\frac{1}{2}$ since x_i is at most $\frac{1}{2}$ and in that case the constraint would be tight. We conclude that $x_i \in [2\epsilon, \frac{1}{2}]$, and $x_j \in [\frac{1}{2} + 2\epsilon, 1]$. Now, if x_j is 1 it is untouched (in y and z) and the constraint remains satisfied. If x_i is $\frac{1}{2}$, then it is clear that $x_i + x_j - 2\epsilon \geq 0$. In all other cases, we add ϵ to

one of these and subtract ϵ from the other in the two perturbed solutions (y and z), and the sum remains unchanged.

Thus, we have, $x_i + x_j - 2\epsilon \geq 1$. Thus, $y_i + y_j \geq 1$ and $z_i + z_j \geq 1$. Thus, y and z satisfy all the constraints where the inequality is not tight.

Now lets consider the case where the constraint is tight i.e. $x_i + x_j = 1$. Without loss of generality, we assume $x_i \leq x_j$. The following possiblities exist for this case

- x_i and x_j are both half-integral. Then $y_i = x_i = z_i$ and $y_j = x_j = z_j$. Therefore, $y_i + y_j = z_i + z_j = x_i + x_j = 1$.
- $x_i \in V_-$ i.e. $x_i < \frac{1}{2}$. Since the constraint is tight, this means that $x_j = 1 - x_i > 1 - \frac{1}{2} = \frac{1}{2}$. Therefore, $x_j \in V_+$. Thus, $y_i = x_i - \epsilon$ and $y_j = x_j + \epsilon$. Thus, $y_i + y_j = x_i - \epsilon + x_j + \epsilon = x_i + x_j = 1$. Similarly, $z_i = x_i + \epsilon$ and $z_j = x_j - \epsilon$. Thus, $z_i + z_j = x_i - \epsilon + x_j + \epsilon = z_i + z_j = 1$.

Thus, in case the constraint is tight for x_i and x_j, it is also tight for y_i and y_j, and for z_i and z_j.

Therefore, y and z are both feasible solutions for our optimization problem. Hence x, which is a convex combination of y and z, cannot be an extreme point solution. This is a contradiction. Therefore, any extreme point solution must be half-integral.

■ **Common mistake 1:** You must explicitly show that y and z are feasible.

7. [3 pts] Prove that the optimal value obtained from the LP relaxation (V_{LP}) will be smaller than or equal to the optimal value obtained from the IP (V_{IP}).

★ **Solution:** The only difference between the IP formulation and the LP relaxation is that we replace the IP constraint $x_i \in \{0, 1\}$ by the constraints $0 \leq x_i \leq 1$. Notice that for the LP relaxation, $x_i = 0$ and $x_i = 1$ are still feasible. In other words, the feasible region of the LP relaxation is larger than the feasible region of the IP formulation. So the optimal IP solution is a feasible solution for the LP relaxation, but it might not be the optimal LP solution. Thus, we conclude that $V_{LP} \leq V_{IP}$.

8. [4 pts] Determine the approximation ratio for the LP relaxation. The approximation ratio is the smallest number k such that for all problem instances, the optimal IP solution value is less than k times the value of the linear programming solution i.e. $V_{LP} \leq V_{IP} \leq k * V_{LP}$.
Hint: Consider a simple scheme for obtaining a feasible IP solution from an LP solution. Round all the variables $\geq$ 1/2 to 1, and all variables $<$ 1/2 to 0. What is the objective value of the rounded solution?

★ **Solution:** Let the optimal LP solution be x^*. We follow the following rounding scheme to obtain a rounded solution x^R

$$x_i^R = \begin{cases} 0 & \text{if } x_i^* \in \left[0, \frac{1}{2}\right) \\ 1 & \text{if } x_i^* \in \left[\frac{1}{2}, 1\right] \end{cases}$$

By construction, $x_i^R \in \{0, 1\}$. Consider the constraint $x_i^* + x_j^* \geq 1$. At least one of the variables x_i^* or x_j^* must be $\geq \frac{1}{2}$ in order to satisfy this constraint. This variable will be rounded to 1, and hence this constraint will still be satisfied by the rounded LP solution x_R. Therefore, we conclude that x_R is a feasible solution for the IP. Thus,

$$V_{LP} \leq V_{IP} \leq V_R$$

where V_R is the value of the objective function at x_R.

In our rounding scheme, we at most double the value of each variable. That is

$$\begin{aligned} & x_i^R \leq 2x_i^* \\ \implies & c_i x_i^R \leq 2c_i x_i^* \\ \implies & \sum_{i \in V} c_i x_i^R \leq 2 \sum_{i \in V} c_i x_i^* \\ \implies & V_R \leq 2V_{LP} \end{aligned}$$

$$\implies V_{LP} \leq V_{IP} \leq V_R \leq 2V_{LP}$$
$$\implies V_{LP} \leq V_{IP} \leq 2V_{LP}$$

Thus, the approximation ratio for the LP relaxation is at most 2.

To show that this approximation ratio is tight (i.e., that we are off by a factor of 2 in some graph), consider an n-clique (n nodes, each connected to every other). You can show that the LP optimal solution for this graph is $x_i = \frac{1}{2}$ for all i, since if you decrease any x_i by ϵ, you would have to increase all other x_j by ϵ to maintain feasibility. Now, twice V_{LP} would be n, and V_{IP} is $n-1$. For $n \geq 2$ nodes in the clique, this ratio ($\frac{n}{n-1}$) can be anywhere between 1 and 2: in particular, for $n = 2$, the ratio is 2, showing that we cannot claim a tighter approximation factor than 2 for this algorithm.

■ **Common mistake 1:** It is important to argue that the rounded solution is indeed a feasible solution for the IP formulation. Only then can we claim that $V_{IP} \leq V_R$.

■ **Common mistake 2:** Though we haven't taken off points for this it is important to argue that k can't be *less* than 2 as well.

2 Von-Neumann's Minimax Theorem [Sivaraman, 10 points]

In this problem you will derive Von Neumann's minimax theorem in game theory using LP duality.

A finite two-person zero-sum game is specified by an m x n matrix A with real entries. In each round, the row player, R, selects a row, say i; simultaneously, the column player, C, selects a column, say j. The *payoff* to R at the end of this round is a_{ij}. Thus, —a_{ij}— is the amount that C pays R if a_{ij} is positive (if a_{ij} is negative then R pays C); no money is exchanged if a_{ij} is 0. This type of game is called a zero-sum game, reflecting the fact that the total amount of money possessed by R and C together is conserved.

The *strategy* of each player is specified by a vector whose entries are non-negative and add up to one, giving the probabilities with which the player picks each row or column. Let R's strategy be given by the m-dimensional $\mathbf{x}$, and C's strategy be given by the n dimensional vector $\mathbf{y}$. The expected payoff to R in a round is $\mathbf{x}^T A\mathbf{y}$. The job of each player is to pick a strategy that guarentees maximum possible expected winnings (equivalently, minimum possible expected losses), regardless of the strategy chosen by the other player. Thus if R chooses strategy $\mathbf{x}$, he can be sure of winning only $\min_{\mathbf{y}} \mathbf{x}^T A\mathbf{y}$, where the minimum is taken over all possible strategies of the player C. Thus, R's best strategy is given by $\max_{\mathbf{x}} \min_{\mathbf{y}} \mathbf{x}^T A\mathbf{y}$. Similarly, C's best strategy is given by $\min_{\mathbf{y}} \max_{\mathbf{x}} \mathbf{x}^T A\mathbf{y}$. The minimax theorem, simply put, states that for any matrix A, $\max_{\mathbf{x}} \min_{\mathbf{y}} \mathbf{x}^T A\mathbf{y} = \min_{\mathbf{y}} \max_{\mathbf{x}} \mathbf{x}^T A\mathbf{y}$.

Q1. [2 pts] A strategy is called *pure* if it picks a single row or column, i.e the vector corresponding to the strategy has only one 1 and the rest 0s. Argue that for any strategy $\mathbf{x}$ of R, the optimal strategy for C can always be expressed as a pure strategy.

★ **Solution:** For any strategy $\mathbf{x}$ of R, the optimal strategy for the column player is

$$\arg\min_y x^T A y$$

Now, any strategy of y gives a value by where $b = x^T A$. (The vector b has a value corresponding to each column of A.) We denote the minimum *value* by b_{min}. Consider the optimal strategy y^*. Now, there are some possibilities.

- The vector y^* is a pure strategy
- The vector y^* is not a pure strategy but the value of b for each of the selected columns is b_{min}. In this case the vector y^*, can be expressed as a pure strategy by setting y^* to be 1 for any of the selected columns and 0 everywhere else.
- The vector y^* is not a pure strategy, and the value of b is not the same for each of the selected columns. Since the value is not the same for all selected columns, we can consider any selected column that doesn't have a value b_{min} (let's say the j^{th} column) and change the column player's strategy by setting y^*_j to 0, and instead playing a column with value b_{min} (say the k^{th} column) with probability $y_k + y_j$. The new strategy attains a value $by^* - (b - b_{min})y_j < by^*$, contradicting the fact that y^* is the optimal strategy.

■ **Common mistake 1:** Although, we haven't penalized you for this it is important to understand the distinction between an optimal strategy, and the optimal response given the opponent's strategy

Q2. [3 pts] Using the observation from above, write the problem of computing R's optimal strategy as an LP.

★ **Solution:** Consider any strategy x for R. Since C's optimal replying strategy is pure, the value of x to R (under optimal play by C) is $\min_j \sum_{i=1}^m a_{ij}x_i$. So, R's optimal strategy is given by

$$\arg\max_x \min_j \sum_{i=1}^m a_{ij}x_i$$

Now, the problem of computing R's strategy is just the LP

$$\begin{array}{lll} \text{maximize} & z & \\ \text{subject to} & z - \sum_{i=1}^m a_{ij}x_i \leq 0, & j = 1, \ldots, n \\ & \sum_{i=1}^m x_i = 1 & \\ & x_i \geq 0 & i = 1, \ldots, m \end{array}$$

Q3. [3 pts] Find the dual of this LP and show that it computes C's optimal strategy.
Hint: Can you argue that for any strategy $\mathbf{y}$ *of C,* $\max_{\mathbf{x}} \mathbf{x}^T A \mathbf{y}$ *is attained for a pure strategy of R? Use this fact.*

★ **Solution:** We introduce the following Lagrange multipliers. A set of α_js for each of the first set of constraints, a β for the second constraint, and a set of γ_is for the third set of constraints.

We get the following dual,

$$\begin{array}{ll} \text{minimize} & \beta \\ \text{subject to} & \sum_{j=1}^{n} \alpha_j = 1 \\ & \beta \geq \sum_{j=1}^{n} \alpha_j a_{ij} \\ & \alpha_j \geq 0 \qquad j = 1, \ldots, n \end{array}$$

The rest of the argument proceeds exactly as before. We first show that, given any strategy for $\mathbf{y}$ of C, $\max_x x^T Ay$ is attained for a pure strategy for R, by considering three cases as before.

Given this fact, we can see that C's optimal strategy is given by

$$\arg\min_y \max_i \sum_{j=1}^{n} a_{ij} y_j$$

Now, the problem of computing C's optimal strategy is just given by the dual LP we formulated.

Q4. [2 pts] Verify that strong duality holds for the primal-dual pair you formulated. Argue that thus strong LP duality implies the Von Neumann minimax theorem.

★ **Solution:** Assuming that the elements of the matrix A are bounded (they are given and fixed), the primal LP is feasible (any strategy with $\sum_i x_i = 1$ and $x_i \geq 0$ is feasible), and the solution is bounded (trivially by $\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}$).

Since the primal LP is both feasible and bounded strong duality holds and the dual LP has the same optimal value. We have already shown that the primal LP computes the optimal strategy for R and the dual computes the optimal strategy for C. Since, these attain the same value we conclude $\max_x \min_y x^T Ay = \min_y \max_x x^T Ay$. This is Von-Neumann's minimax theorem.

3 Degeneracy and Uniqueness [Sivaraman, 15 points]

Q1. Consider the following pair of problems that are duals of each other:

$$\begin{array}{ll} \text{minimize}_x & c^T x \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array}$$

and

$$\begin{array}{ll} \text{maximize}_p & p^T b \\ \text{subject to} & p^T A \leq c^T \end{array}$$

Prove that if the primal has a nondegenerate and unique optimal solution, so does the dual.

★ **Solution:** Let us assume $A \in \mathbf{R}^{m \times n}$.

There is a unique, nondegenerate optimal solution to the primal problem. Let us apply the simplex method, to the problem. Simplex terminates when the reduced cost vector is nonnegative:

$$\bar{c} = c - c_B^T A_B^{-1} A \geq 0 \ .$$

Now, the primal is unique and nondegenerate if and only if $\bar{c_j} > 0$ for all non-basic c_j.

Let us consider a dual candidate solution $p^T = c_B^T A_B^{-1}$. Let us show that this is an optimal feasible solution.

First, $p^T A = c_B^T A_B^{-1} A \leq c$, therefore it is feasible. Also, $p^T b = c_B^T A_B^{-1} b = c_B^T x_B = c^T x$, since all non-basic x's are zero. Thus, since x and p are primal and dual feasible solutions, and $c^T x = p^T b$, we know that x and p are primal and dual optimal.

First, let us show that p is nondegenerate. To do this we will utilize the fact that the primal is *both* non-degenerate and unique. $p^T A_j = c_j$ iff the reduced cost $\bar{c}_j = 0$. The number of constraints that are tight at p is therefore $\leq m$ (strictly speaking, since the reduced cost for any variable in the basis is 0, the number of tight constraints is exactly m), and since $p \in \mathbf{R}^m$, the solution is nondegenerate.

Now, let us show that p is unique. Again, we use the fact that the optimal x is nondegenerate. Since the primal is in standard form this means that *exactly* m of the x_js are non-zero (i.e. the basis x_js have to be non-zero if the solution is non-degenerate since otherwise we would have m equality constraints and at least (n-m+1) inequality constraints in the primal being tight at the optimal which leads to degeneracy). Using complementary slackness (which says that $(c_j - p^T A_j)x_j = 0$), we have that $p^T A_j = c_j$ for at least these m indices. Since the basic columns A_j are independent (by the definition of a basis), this gives a system of m equations, which has a unique solution $p^T = c_B^T A_B^{-1}$.

■ **Common mistake 1:** While it is true that primal (dual) non-degeneracy implies dual (primal) uniqueness, it is *not* true that primal (dual) uniqueness implies dual (primal) non-degeneracy. You need both non-degeneracy and uniqueness to conclude that the dual problem is unique.

Q2. Give an example in which the primal problem has a degenerate optimal basic feasible solution, but the dual has a unique optimal solution. (The example need not be in standard form.)

★ **Solution:** Consider the following example in standard form

$$\begin{aligned} \text{minimize}_x \quad & \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}^T x \\ \text{subject to} \quad & \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 5 & 1 & -1 \end{pmatrix} x = \begin{pmatrix} 2 \\ 0 \end{pmatrix} \\ & x \geq 0 \end{aligned}$$

and its dual

$$\begin{aligned} \text{maximize}_p \quad & p^T \begin{pmatrix} 2 \\ 0 \end{pmatrix} \\ \text{subject to} \quad & p^T \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 5 & 1 & -1 \end{pmatrix} \leq \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}^T \end{aligned}$$

One of the primal optima (there are several but this doesn't matter) is [2 0 0 0]', which is degenerate because it can be expressed in terms of several bases ({1,2}, {1,3} etc.).

The dual optimum on the other hand is [1 0]' and is unique.

Q3. Consider the problem

$$\begin{aligned} \text{minimize} \quad & x_2 \\ \text{subject to} \quad & x_2 = 1 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned}$$

Write down its dual. For both the primal and the dual problem determine whether they have unique optimal solutions and whether they have nondegenerate optimal solutions. Is this example in agreement with the statement that nondegeneracy of an optimal basic feasible solution in one problem implies uniqueness of optimal solutions for the other? Explain.

★ **Solution:** The primal does not have a unique optimal solution, but it is nondegenerate. The dual is

$$\begin{aligned} \text{maximize} \quad & \alpha \\ \text{subject to} \quad & \alpha + \gamma = 1 \\ & \beta = 0 \\ & \gamma \geq 0 \\ & \beta \geq 0 \end{aligned}$$

where α, β, γ are the Lagrange multipliers for each of the three constraints in order. The dual has a unique solution [1 0 0]', but it is degenerate. The bases {1,2,3} and {1,3,4} lead to the same dual optimal. Thus, the problem given doesn't contradict the statement.

■ **Common mistake 1:** If you take the dual using the cheat sheet you must argue one of two things, either you mustn't drop the $0p \leq 0$ (here p is the dual variable and you will have only one dual variable for this problem) constraint and therefore conclude that the dual was in fact degenerate, or you must take the dual of the dual (after dropping this constraint) and show that it isn't the original primal problem anymore. Both of these were given full credit, but you must conclude that the given statement was in fact true. Note the cheat sheet dual variable p is the same as α above (the cheat sheet eliminates the other two dual variables by treating them as slack variables).

4 SVMs: Duality and Kernel Tricks [Yi, 25 points]

In this problem, we will consider training Support Vector Machines (SVMs) as an optimization problem. We will derive the dual form of the problem, use the dual solution to get the primal solution, and discuss kernel tricks under the dual form.

4.1 Deriving the dual

In class, we saw SVMs phrased as the problem of finding a maximum-margin hyperplane that separates positive and negative examples (Figure 1, left):

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2} \parallel \mathbf{w} \parallel^2 \\ \text{s.t.} \quad & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \ \forall i = 1, 2, \ldots, m \end{aligned}$$

Here each training example $\mathbf{x}_i \in \mathbf{R}^n$ is an $n \times 1$ vector, w is also an $n \times 1$ vector containing n coefficients, b is a scalar, and m is the number of training examples. Note that $(\mathbf{w}, b)$ together defines a separating hyperplane $\{\mathbf{x} \mid \mathbf{w} \cdot \mathbf{x} + b = 0\}$ in $\mathbf{R}^n$. We want all positive examples (with $y = 1$) to satisfy $\mathbf{w} \cdot \mathbf{x} + b \geq 1$ and negative examples (with $y = -1$) to satisfy $\mathbf{w} \cdot \mathbf{x} + b \leq -1$, which is exactly the constraints in the above formula.

In class, we then derived the dual problem and saw how we could interpret the Lagrange multipliers. If, however, the examples are not linearly separable, this optimization problem may be changed to allow some examples to be misclassified by the hyperplane. We do this by introducing a slack variable ξ_i for each example i, which represents the distance we would need to move example i for it to be on the correct side of the margin area. (Figure 1, right). We can penalize examples for ξ_i (degree of margin violation), but we have to balance this with our previous goal of maximizing the margin. This gives a new optimization problem:

$$\begin{aligned} \min_{\mathbf{w},b,\xi_i} \quad & \frac{1}{2} \parallel \mathbf{w} \parallel^2 + C \sum_{i=1}^{m} \xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \ \xi_i \geq 0, \ \forall i = 1, 2, \ldots, m \end{aligned} \tag{1}$$

where C is a constant chosen to trade off the simultaneous goals of maximizing the margin and minimizing the margin violation. We call this problem as the **primal form** of SVMs with slack variables.

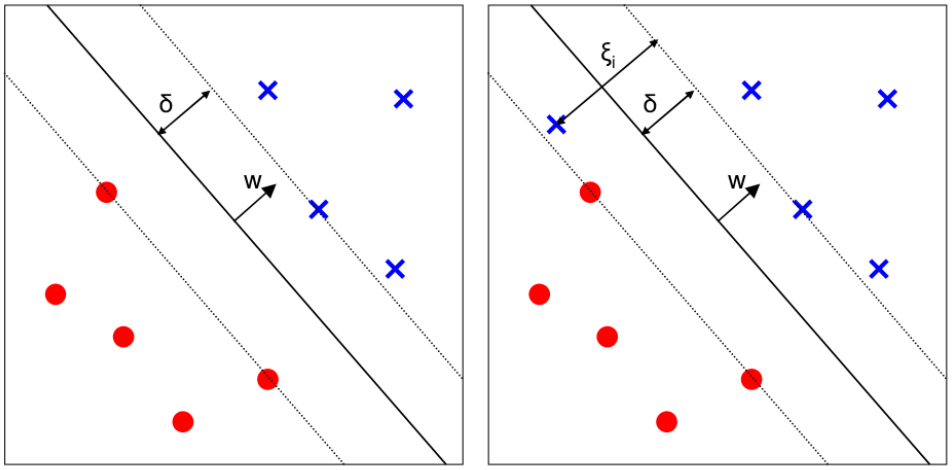


Figure 1: Binary classification with SVMs. The bold hyperplane in each figure represents the hyperplane chosen as the classifier. The dotted hyperplanes are the *margin hyperplanes*, i.e. the hyperplanes which are parallel to and at distance δ (the margin) from the classifying hyperplane.

1. [2 pts] How can we set C to make this problem essentially equivalent to the original problem without slack variables?

   ★ **Solution:** Set C to be very large. As $C \longrightarrow \infty$, maximizing the margin becomes much less important than ensuring that the chosen hyperplane correctly classifies all examples, so the optimal solution will be an exactly separating hyperplane (if one exists) rather than a large-margin hyperplane which misclassifies a few examples.

2. [8 pts] Write the Lagrangian of this optimization problem for SVMs with slack variables. Use $\{\alpha_i\}_{i=1}^m$ to denote Lagrangian multipliers and derive the dual. Explicitly state how you derive the dual.

★ **Solution:** Adding Lagrangian multipliers (dual variables) α, μ, we can write down the Lagrangian:

$$\min_{\mathbf{w},b,\xi} \max_{\alpha,\mu} \mathscr{L}(\mathbf{w}, b, \xi, \alpha, \mu) = \frac{1}{2} \parallel \mathbf{w} \parallel^2 + C \sum_i \xi_i - \sum_i \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - (1 - \xi_i)] - \sum_i \mu_i \xi_i,$$
$$\text{s.t. } \alpha_i \geq 0,\ \mu_i \geq 0,\ \forall i$$

The derivatives of this Lagrangian give:

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$
$$\frac{\partial L}{\partial b} = 0 \quad \Rightarrow \quad \sum_i \alpha_i y_i = 0$$
$$\frac{\partial L}{\partial \xi_i} = 0 \quad \Rightarrow \quad \alpha_i = C - \mu_i$$

We can use these equalities to eliminate the primal variables $\mathbf{w}, b, \xi$ from the Lagrangian to get the dual problem:

$$\max_{\alpha,\mu} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$
$$\text{s.t. } 0 \leq \alpha_i \leq C,\ \sum_i \alpha_i y_i = 0,\ \forall i$$

Here, we have eliminated μ_i by combining $\alpha_i = C - \mu_i$ and $\mu_i \geq 0$ to get $\alpha_i \leq C$.

4.2 Interpreting the Dual & Complementary Slackness

Given the solution of the dual problem, we can use it to compute the solution to the primal problem (i.e. $\mathbf{w}$ and b).

1. [1 pts] Write down an equation for computing $\mathbf{w}$ using the dual solution.

   ★ **Solution:**

   $$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \qquad \text{(from above)}$$

2. [2 pts] Write down an equation for computing b using the dual solution.

   ★ **Solution:**

   $b = y_s - \mathbf{w} \cdot \mathbf{x}_s$ where s is the index for any support vector with $\xi_s = 0$, or, equivalently, $0 < \alpha_s < C$

   To compute b, we know that if vector i lies on the margin hyperplane, then $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i$ holds with equality, and $\xi_i = 0$, so $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1$. Multiplying by y_i and using $y_i^2 = 1$ gives the above equation. Note: Only support vectors which lie on the margin hyperplanes are used to compute b in this equation. Note: Averaging over all such support vectors instead of a single one s gives a more numerically stable solution.

3. [3 pts] As you just showed, we can compute the optimal separating hyperplane using the solution to the dual problem. Only a subset of the examples $\mathbf{x}_i$ are actually used to compute $\mathbf{w}$ and b. Where do these examples lie with respect to the hyperplane and two margin hyperplanes (Figure 1), and what can you state about the values of their corresponding slack variables ξ_i and Lagrange multipliers?

★ **Solution:** Since $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$, you can see that the only $y_i, \mathbf{x}_i$ which contribute to this sum are the ones for which $\alpha_i > 0$. These are called *support vectors* (SVs). We can use complementary slackness to describe these vectors as follows: If $\alpha_i > 0$, then the constraint $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i$ in the primal problem must be tight. In other words, vector i is exactly ξ_i away from the margin hyperplane for y_i (i.e. the margin hyperplane on the correct side of the separating hyperplane). If $\xi_i = 0$, then vector i is directly on the (correct) margin hyperplane, and $\mu_i > 0$. If $1 > \xi_i > 0$, then vector i is correctly classified but lies between the separating and margin hyperplanes. If $\xi_i = 1$, then vector i lies directly on the separating hyperplane. If $\xi_i > 1$, then vector i lies on the wrong side of the separating hyperplane. (Note the margin hyperplanes are at distance $\delta = 1/ \parallel \mathbf{w} \parallel$ from the separating hyperplane.) Finally, we know $\mu_i < C$ since $\alpha_i > 0$ and $\alpha_i = C - \mu_i$. Combining these, we get:

- SVs on correct margin hyperplane:
  $0 < \alpha_i < C$, $\xi_i = 0$, $0 < \mu_i < C$, correctly classified
- SVs strictly between correct margin hyperplane and separating hyperplane:
  $\alpha_i = C$, $0 < \xi_i < 1$, $\mu_i = 0$, correctly classified
- SVs on the separating hyperplane:
  $\alpha_i = C$, $\xi_i = 1$, $\mu_i = 0$
- SVs on the wrong side of the separating hyperplane:
  $\alpha_i = C$, $\xi_i > 1$, $\mu_i = 0$, incorrectly classified

4.3 Kernel Tricks

Now we are not satisfied by fitting a linear hyperplane on the original input space. Thus, for each example $\mathbf{x}_i \in \mathbf{R}^n$, we use a more informative representation $\phi(\mathbf{x}_i)$. If this mapping $\phi()$ is nonlinear, a "linear" hyperplane on $\phi(\mathbf{x})$ is actually nonlinear on $\mathbf{x}$. This is how we generalize SVMs into nonlinear classifiers. Now the primal form of our favorite SVMs is as follows:

$$\begin{aligned} \min_{\mathbf{w},b,\xi_i} \quad & \frac{1}{2} \parallel \mathbf{w} \parallel^2 + C \sum_{i=1}^{m} \xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w} \cdot \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i,\ \xi_i \geq 0,\ \forall i = 1, 2, \ldots, m \end{aligned} \tag{2}$$

Then, the dual form will become (note: feel free to use this as a hint for your previous question of "derive the dual," but in that question we focus more on the derivation):

$$\begin{aligned} \max_{\{\alpha_i\}_{i=1}^m} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C,\ \forall i = 1, 2, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y_i = 0 \end{aligned} \tag{3}$$

Consider, however, if we can *not* directly access the training examples in the new feature space $\{\phi(\mathbf{x}_i)\}_{i=1}^m$. Instead, we have some lucky way (e.g., a "kernel" function $k()$) to compute the dot product of any two examples: $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$. In the following question, you can use kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ on any two examples $\mathbf{x}_i$ and $\mathbf{x}_j$ since you have access to $\mathbf{x}$.

1. [2 pts] Without using $\{\phi(\mathbf{x}_i)\}_{i=1}^m$, can we solve the dual form? Why?

   ★ **Solution:** Yes. In the dual form, $\phi(\mathbf{x}_i)$ only appears in inner products, which can be computed by kernel. In other words, we just use $k(\mathbf{x}_i, \mathbf{x}_j)$ to replace $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$ in the dual form.

2. [4 pts] Suppose we solve the dual form and obtain the dual solution, can we write out our solution to the hyperplane ($\mathbf{w}$ and b) using the dual solution? Why? (Hint: you may want to discuss $\mathbf{w}$ and b separately).

★ **Solution:** For $\mathbf{w}$, no, since $\mathbf{w} = \sum_i \alpha_i y_i \phi(\mathbf{x}_i)$ (as shown in previous questions) and we do not have access to $\phi(\mathbf{x}_i)$. For b, yes, since $b = y_s - \mathbf{w} \cdot \phi(\mathbf{x}_s)$ (as shown in previous questions where s is the index for any example with $0 < \alpha_s < C$) and we can compute $\mathbf{w} \cdot \phi(\mathbf{x}_s) = \sum_i \alpha_i y_i \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_s) = \sum_i \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}_s)$, even without an explicit form of $\mathbf{w}$.

3. [3 pts] Given a new example $\mathbf{x}_{new}$, how can we compute $\mathbf{w} \cdot \phi(\mathbf{x}_{new}) + b$ in order to classify it?

★ **Solution:** $\mathbf{w} \cdot \phi(\mathbf{x}_{new}) + b = \sum_i \alpha_i y_i \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_{new}) + b = \sum_i \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}_{new}) + b$.

5 SVMs: Programming [Yi, 30 points]

In this problem, we try to implement our solvers for the SVM problem (2), as QPs in both the primal form and the dual form. All datasets in this problem are given in text files with one example $\mathbf{x}_i$ per line. Each line gives a whitespace-delimited list of the example's attribute values, followed by the +/-1 value of the label, which can be directly loaded into Matlab using "load".

Hint 1: each experiment should finish within a few minutes (actually less than a minute). If any experiment seems to run forever, check you code :)

Hint 2: It is very common to misspecify your QP problem when you call a QP solver. I did that several times. If you encounter infeasibility or unboundedness issue when you call the QP solver, the first thought is to check the code for calling the QP solver.

5.1 Solving the Dual Form

We first solve an SVM using the dual form you just derived. In this part, we use $\phi(\mathbf{x}_i) = \mathbf{x}_i$ for each example i, i.e., the original representation as given in the text files. Thus the kernel function is $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$.

1. [4 pts] Using a quadratic program (QP) solver (I recommend Geoff's iqph.m, as posted with HW2 on the class website), implement your own solver for SVM problem in the dual form. In principle, your program should take three inputs: 1) an $m \times m$ kernel matrix K, where each element $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, and m is the number of training examples. 2) An $m \times 1$ vector y of all labels of training examples. 3) A scalar C as given in the primal form (2).

★ **Solution:** See the solution code on the course website.

2. [4 pts] Download the two-gaussians dataset from the homework webpage, and use your SVM solver to compute the solution to the dual form for $C = 1$, 10, 100, respectively. Then, compute $\mathbf{w}$ and b of the optimal separating hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$ for each choice of C. For each hyperplane you compute, plot the dataset using different colors for the 2 classes, and superimpose the hyperplane, labeled with the value of C. Indicate the support vectors in your plots. NOTE: use the command scatterplot() to plot examples and line() to plot the hyperplane.

★ **Solution:** See figure 2.

3. [4 pts] Download the OCR dataset from the homework webpage (two files: ocr-train and ocr-test). This dataset is from the UCI ML Repository (Optical Recognition of Handwritten Digits Data Set, archive.ics.uci.edu/ml/datasets/ Optical+Recognition+of+Handwritten+Digits). The original dataset contains images of handwritten digits (0-9), but this version has been modified so class -1 is digits (1,4) and class +1 is digits (5,7). The training set contains 724 images from 13 people, and the test set contains 1539 images from 30 different people. For this experiment, set $C = 1$. Train your SVM solver

using the first 7, 20, 55, 148, 403 and 724 training examples. Plot the number of training examples used vs. the accuracy of the computed hyperplane on the test data. Also, plot the number of training examples vs. the number of support vectors.

★ **Solution:** See figure 3.

5.2 Solving the Primal Form

We implement a solver for the primal form SVM as eq. (2).

1. [4 pts] Using a quadratic program solver (e.g., Geoff's iqph.m), implement your own solver for SVM primal form as eq. (2). In principle, your program should take three inputs: 1) an $m \times n$ data matrix $\mathbf{X}$, where each row is one of the m training examples, and n is the number of features. 2) An $m \times 1$ vector y of all labels of training examples. 3) A scalar C as given in the primal form. NOTE: when your primal form solver tries to call the QP solver (e.g., iqph.m), you will find the some of the input arguments you send to the QP solver are actually sparse matrices. Don't forget to explicitly declare these matrices as sparse matrices before you call the QP solver, e.g., H = sparse(H) if most elements in H are zeros. This makes the solver running quickly.

   ★ **Solution:** See the solution code on the course website.

2. [3 pts] Repeat your experiments on the "two-gaussians" dataset in part 2 of 5.1. Again, plot the dataset, the hyperplane and the support vectors for each C. The basic idea is to check if your primal form solver is working properly. NOTE: given the primal form solution, you need a different way to find out support vectors (because you no longer have the dual solution). Also you may finally have slightly different set of support vectors due to numerical issue of the solver. That's fine, but make sure the hyperplane for each C looks the same as the hyperplane from the dual solver.

   ★ **Solution:** See figure 4.

5.3 Primal VS. Dual

In this part, we compare the difference of the above two ways to optimize SVMs (i.e., as QPs in primal form and as QPs in dual form). In particular, we will see how the number of features will change the performance of two QP solvers. This part is based on the third "two-gaussians-large" data set, which contains 1000 examples in two dimensional space. The third column in the file contains labels.

We will use some nonlinear feature mapping $\phi_d(\mathbf{x})$, where the dot product in the mapped space is defined as $\phi_d(\mathbf{x}_i) \cdot \phi_d(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$, i.e., the polynomial kernel of degree d. We will change d in our experiments. Recall that the SVM solver in the dual form takes the $m \times m$ kernel matrix K as input. The size of K does not depend on the dimensionality of $\phi_d(\mathbf{x})$, and each element K_{ij} of K is efficiently computed by the kernel function $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$. Therefore, increasing the degree d of polynomial should not change too much the optimization time in the dual form. On the other hand, the primal form SVM solver operates on the actual feature vectors $\{\phi_d(\mathbf{x})_i\}_{i=1}^m$, whose dimensionality can be very large even we choose only a moderately large d. The goal of this experiment is to explore what effects this different has on the running time of solvers.

1. [2 pts] Derive the explicit feature mapping $\phi_d(\mathbf{x})$, for which the dot product is defined as $\phi_d(\mathbf{x}_i) \cdot \phi_d(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$. Hint: I actually provide the code for generating the kernel matrix K and the feature mapping $\phi_d(\mathbf{x})$, see generateFeatures.m and generateKernelMtx.m posted with HW2. You can use this code as a hint for answering your question. Also, the answer to this question is not unique. The only requirement is $\phi_d(\mathbf{x}_i) \cdot \phi_d(\mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$.

★ **Solution:** For an example $\mathbf{x} = (x_1, x_2, ..., x_p)^T$ with p variables, the kernel feature vector $\phi_d(\mathbf{x})$, as shown by the hint code, can be generated by two steps: 1) add an intercept term so that we have $(1, x_1, x_2, ..., x_p)^T$; 2) $\phi_d(\mathbf{x})$ has a dimensionality of $(p+1)^d$, where each term is the product of any d elements from $(1, x_1, x_2, ..., x_p)$ (selecting *with* replacement). Intuitively, $\phi_d(\mathbf{x})$ contains all terms with a degree d or smaller. If we don't add the intercept term $x_0 = 1$ to $\mathbf{x}$ in step 1), step 2) will generate p^d terms of exactly degree d. After step 1), step 2) generates $(p+1)^d$ terms of degree d or smaller.

Additional note: the kernel feature vector $\phi_d(\mathbf{x})$ generated this way is redundant, e.g., we will have such 6 terms: $x_1x_3x_4$, $x_1x_4x_3$, $x_3x_1x_4$, $x_3x_4x_1$, $x_4x_1x_3$, $x_4x_3x_1$. These 6 degree-3 terms can be replaced by only one term: $\sqrt{6}x_1x_3x_4$, while keeping the inner product $\phi(\mathbf{x}) \cdot \phi(\mathbf{x}')$ for any $\mathbf{x}$ and $\mathbf{x}'$ unchanged. Generally, any degree-k term will actually have $k!$ copies, which can be replaced by one term with a coefficient $\sqrt{k!}$

2. [5 pts] Use the first $m = 600$ examples $\{\mathbf{x}_i\}_{i=1}^m$ in the data set, change the degree of polynomial kernel $d = 1, 2, 3, 4, 5, 6$. For each choose degree d, generate the kernel matrix K (for your dual solver) and the mapped feature vectors $\{\phi_d(\mathbf{x})_i\}_{i=1}^m$ (for your primal solver), using the code I provided (or your own code). Solve the SVM using both your dual form solver and primal form solver, respectively. Plot the optimization time of QP solver vs. the polynomial degree d, in primal form and dual form, respectively. NOTE: the matlab command "tic" and "toc" are used to record the time for certain operations. Please only record the running time for the *QP solver* (e.g., Geoff's iqph.m): call "tic" exactly before you call the QP solver, and call, e.g., "runtime = toc" right after the QP solver finishes.

★ **Solution:** See figure 5.

3. [4 pts] Briefly describe what you observe. Which solver is faster when $d = 1$, and which solver is faster when $d = 6$? How did the running time change for the two solvers?

★ **Solution:** As we can see from figure 5, computational complexity of the primal solver is slightly lower than the dual form solver for low-degree polynomial kernels, but is exponentially higher as the polynomial degree increases. Computational complexity for the dual form solver is basically remaining constant as the polynomial degree changes. This is because the size of the dual form QP depends on the number of examples, while the size of the primal form QP depends on the number of features.

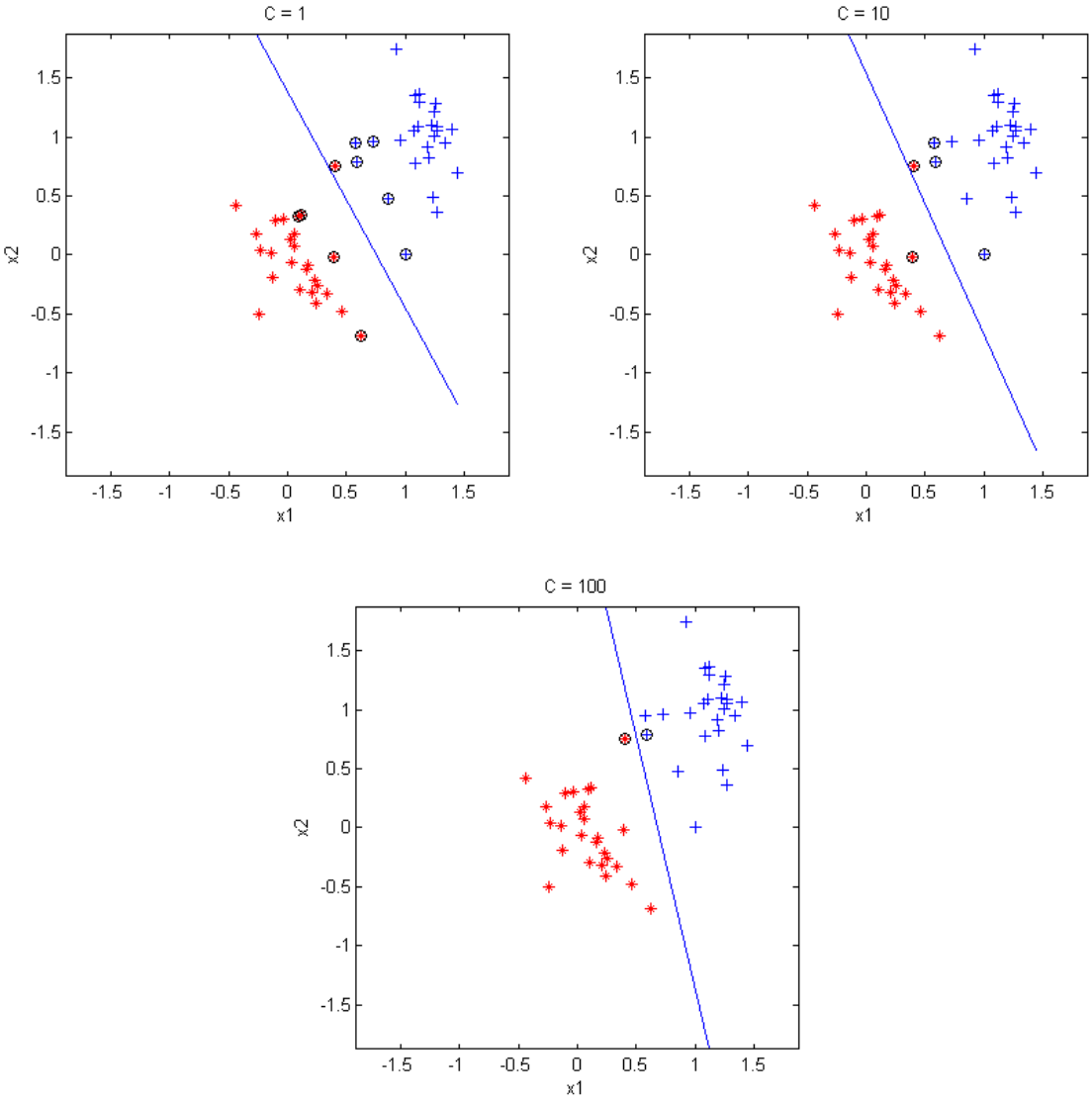


Figure 2: The SVM dual solver for two-Guassian data set. Support vectors are circled.

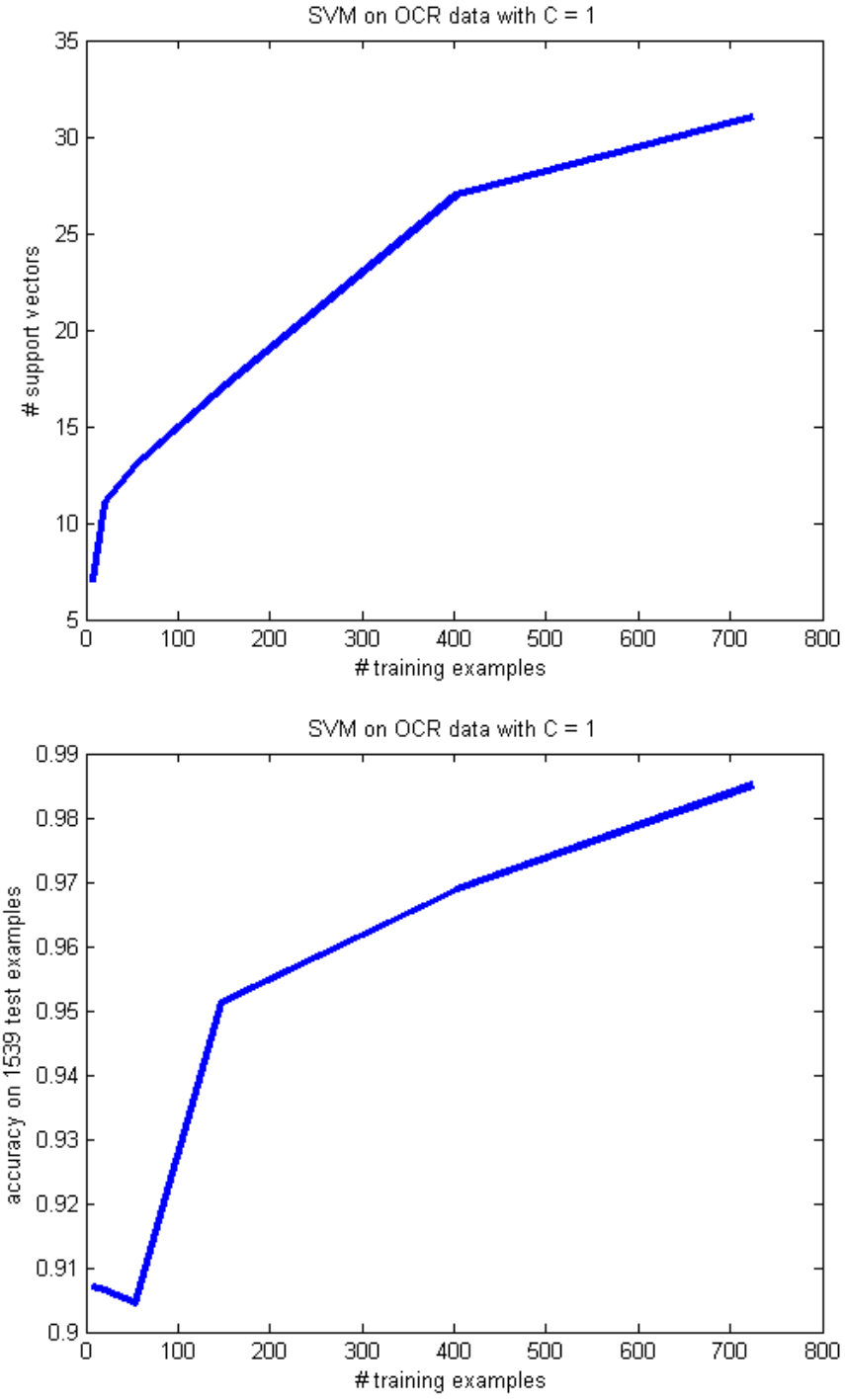


Figure 3: The SVM Dual solver for OCR data set, using $C = 1$.

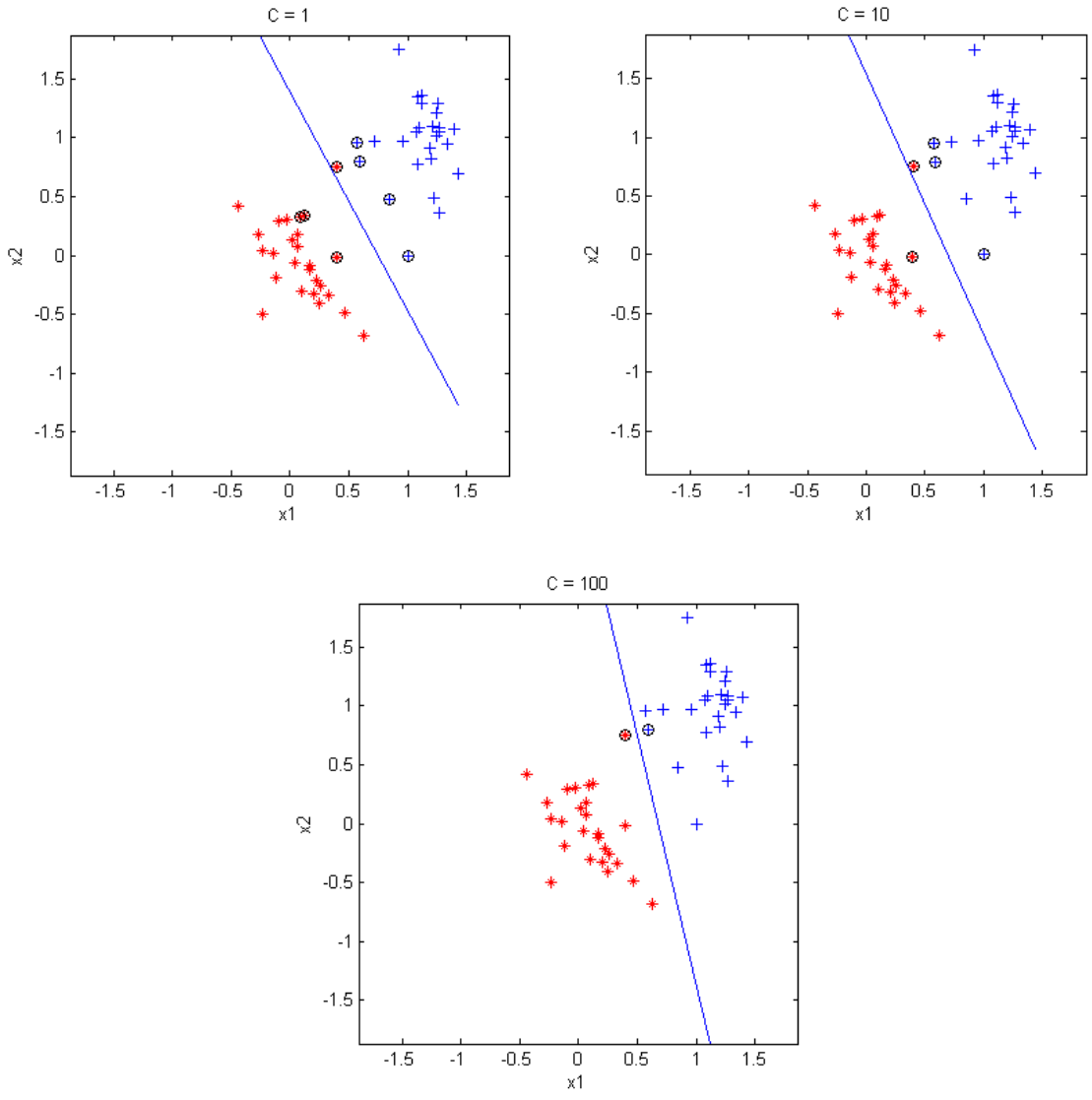


Figure 4: The SVM primal solver for two-Guassian data set. Support vectors are circled.

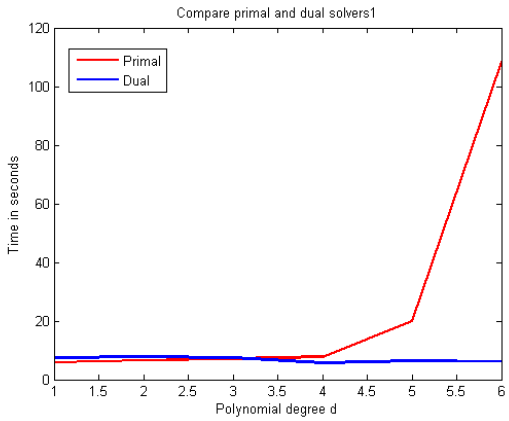


Figure 5: Compare primal and dual solvers with polynomial kernels.