# 10-725 Optimization, Spring 2010: Homework 2

Due: Wednesday, February 17, beginning of class

**Instructions** There are 4 questions on this assignment. The last question involves coding. Do *not* attach your code to the writeup. Instead, copy your implementation to

`/afs/andrew.cmu.edu/course/10/725/Submit/your_andrew_id/HW2`

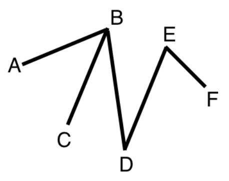
for andrew, or you can log into, for example, unix.andrew.cmu.edu. Please submit each problem *separately* with your name and userid on each problem. Refer to the webpage for policies regarding collaboration, due dates, and extensions.
*Last revised: Wednesday, February 3rd at 11:00PM*

## 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. [2 pts] 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.

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

3. [2 pts] If the vertices have different weights, what is the new objective function?

4. [3 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.*

5. [3 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.

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.*

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 ($V_{IP}$).

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?*

## 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.

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

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.*

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.

## 3 Degeneracy and Uniqueness [Sivaraman, 15 points]

Q1. [6 pts] Consider the following pair of problems that are duals of each other:

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

and

$$\begin{aligned} \text{maximize} \quad & p^T b \\ \text{subject to} \quad & p^T A \leq c^T \end{aligned}$$

Prove that if the primal problem has a nondegenerate and unique optimal solution, so does the dual.
*Hint 1: You can use the following fact. A solution is nondegenerate and uniquely optimal, if the simplex method terminates, and the reduced cost vector (denoted as $\bar{c}$ in the previous assignment) is *strictly* positive for all non-basic variables.*
*Hint 2: To show that the dual solution is unique use the complementary slackness conditions.*

Q2. [6 pts] 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), or explain why such a pair cannot exist.

Q3. [3 pts] 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.

## 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 $\xi_i$ 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 $\xi_i$ (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.

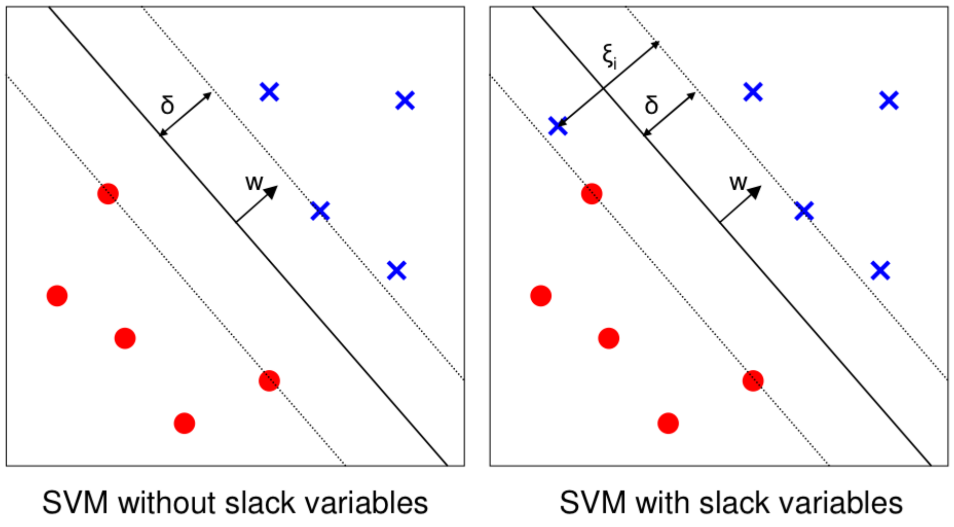


Figure 1: Binary classification with SVMs. The bold hyperplane in each figure represents the hyperplane chosen as the classifier. The dotted hyperplanes are the *margin hyperplanes*, i.e. the hyperplanes which are parallel to and at distance $\delta$ (the margin) from the classifying hyperplane.

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

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.

### 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.
2. [2 pts] Write down an equation for computing $b$ using the dual 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 $\xi_i$ and Lagrange multipliers?

### 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 duel", 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?
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).
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?

## 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).

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.

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.

## 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.

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.

## 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 $\phi(\mathbf{x})$, 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 $\phi_d(\mathbf{x}_i) \cdot \phi_d(\mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$.

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.

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?