Assignment 5

Machine Learning, Summer term 2014, Ulrike von Luxburg

To be discussed in exercise groups on May 19-21

Exercise 1 (Primal hard margin SVM problem, 1+3 points) Given training data $(X_i, Y_i)_{i=1,\dots,n}$ with $X_i \in \mathbb{R}^d$ and $Y_i \in \{-1, +1\}$ the primal hard margin SVM problem is given as

$$\begin{aligned} &\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{2}\|w\|^2 \\ &Y_i(w^T X_i + b) \geq 1, \quad i = 1, \dots, n \end{aligned} \tag{1}$$

(a) Recall the meaning of a hyperplane in canonical representation. Show that any solution of (1) gives rise to a hyperplane in canonical representation.

(b) Assume the data is linearly seperable, that is there exists a solution of (1). Show that this solution is unique.

Linear Programming (LP): A linear program is a special case of a convex optimization problem. We want to optimize a linear objective function, subject to linear constraints. For example, consider the following linear program:

$$\begin{aligned} \text{minimize} \quad & 4x_1 + 3x_2 - x_3 \\ \text{subject to} \quad & -x_1 + x_2 \leq 1 \\ & 4x_1 - 2x_2 + 3x_3 \leq -2 \\ & -2x_2 - 3x_3 + 4 \leq 0 \\ \text{and} \quad & x_i \leq 0 \; ; \; i = 1, 2, 3 \end{aligned}$$

We can rewrite this linear program in a standard form

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ \text{and} \quad & x \leq 0 \end{aligned} \tag{2}$$

where $x = (x_1, x_2, x_3)^T \in \mathbb{R}^3$, $A = \begin{bmatrix} -1 & 1 & 0 \\ 4 & -2 & 3 \\ 0 & -2 & -3 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ -2 \\ -4 \end{bmatrix}$ and $c = \begin{bmatrix} 4 \\ 3 \\ -1 \end{bmatrix}$.

Exercise 2 (LP in standard form, 2 points) Make a transformation of the variables such that you can write the following linear program in the standard form (2). Determine the corresponding matrix A and the vectors c and b.

$$\begin{aligned} \text{minimize} \quad & x_1 - 2x_2 + 4x_3 \\ \text{subject to} \quad & -x_1 + x_2 \geq 1 \\ & 3x_1 + 2x_3 \leq -1 \\ & -2x_1 - 5x_3 + 4 \leq 0 \\ & x_1 + x_2 + 8x_3 \leq 10 \\ \text{and} \quad & x_1, x_2 \leq 0 \\ & x_3 \geq 0 \end{aligned}$$

Web page: `http://www.informatik.uni-hamburg.de/ML/contents/people/luxburg/teaching/2014-ss-vorlesung-ml/`
Login: "machine", Password: "learning"

Exercise 3 (LP and its dual, 2+1 points) We want to derive the Lagrangian dual problem for the linear program (2). We assume $x, c \in \mathbb{R}^d$, $A \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$. First form the Lagrangian

$$L(x, \lambda_1, \lambda_2) = c^T x + \lambda_1^T (Ax - b) + \lambda_2^T x$$

where $\lambda_1 \in \mathbb{R}^n$ and $\lambda_2 \in \mathbb{R}^d$ are vectors of Lagrange multipliers.

(a) For any pair $(\lambda_1, \lambda_2) \in \mathbb{R}^n \times \mathbb{R}^d$ determine

$$g(\lambda_1, \lambda_2) = \inf_{x \in \mathbb{R}^d} L(x, \lambda_1, \lambda_2).$$

g is called the Lagrange dual function. *Hint: In particular, this requires to determine the pairs* (λ_1, λ_2) *for which* $\inf_{x \in \mathbb{R}^d} L(x, \lambda_1, \lambda_2) = -\infty$.

The Lagrangian dual problem is given by

$$\begin{array}{ll} \text{maximize} & g(\lambda_1, \lambda_2) \\ \text{subject to} & \lambda_1, \lambda_2 \geq 0 \end{array}$$

(b) Show that in our case the dual problem can be written as a linear program. (You do not have to rewrite it in standard form (2)).

Optimization in MATLAB: For the following, we highly recommend to use the CVX optimization package (read `prepare05.pdf` for an introduction to this package - available on the course webpage). However, in principle you could also use the MATLAB functions `linprog` and `quadprog`.

Exercise 4 (Solving a linear program, 3 points) Solve in MATLAB the linear program

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \leq b \end{array} \tag{3}$$

where $A = \begin{bmatrix} -1 & -1 \\ -0.5 & -1 \\ -2 & -1 \end{bmatrix}$, $b = \begin{bmatrix} -4 \\ -2 \\ -4 \end{bmatrix}$ and $c = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Then solve the program (3) with A and b replaced by

$$\widetilde{A} = \begin{bmatrix} -1 & -1 \\ -1 & -1 \\ -0.5 & -1 \\ -2 & -1 \end{bmatrix}, \qquad \widetilde{b} = \begin{bmatrix} -2 \\ -4 \\ -2 \\ -4 \end{bmatrix}.$$

Do you get the same solution? What would you expect? Try to solve the system by hand and explain.

Exercise 5 (SVM cancer detection, 4 points) In this exercise you should learn a (soft margin) SVM that classifies cancers as either benign (-1) or malignant (+1) depending on the characteristics of sample biopsies. Load the patients data from `cancer_data2014.mat` (available on the course webpage). For every patient, 9 attributes are measured:

1- Clump thickness 2- Uniformity of cell size 3- Uniformity of cell shape 4- Marginal Adhesion 5- Single epithelial cell size 6- Bare nuclei 7- Bland chomatin 8- Normal nucleoli 9- Mitoses.

For $C \in \{0.01, 0.1, 0.5, 1, 5, 10, 50\}$ train a SVM classifier on the training data and evaluate it on the test data. Plot the train and the test error (with respect to the 0-1-loss) as a function of C. What is the effect of choosing a large C on the training error? Does this effect coincide with what you are expecting?

MACHINE LEARNING- ASSIGNMENT 5

VICTOR BERNAL ARZOLA

Exercise 2

Consider the optimization problem given by

$$\begin{array}{ll} min & x_1 - 2x_2 + 4x_3 \\ subject & -x_1 + x_2 \geq 1 \\ & 3x_1 + 2x_3 \leq -1 \\ & -2x_1 - 5x_3 + 4 \leq 0 \\ & x_1 + x_2 + 8x_3 \leq 10 \\ x_3 \geq 0 & x_1, x_2 \leq 0 \end{array}$$

it can be written in the standar form, calling $\tilde{x_3} := -x_3$

$$\begin{array}{ll} min & x_1 - 2x_2 - 4\tilde{x_3} \\ subject & x_1 - x_2 \leq -1 \\ & 3x_1 - 2\tilde{x_3} \leq -1 \\ & -2x_1 - 5x_3 \leq -4 \\ & x_1 + x_2 - 8\tilde{x_3} \leq 10 \\ & x_1, x_2, \tilde{x_3} \leq 0 \end{array}$$

Then $c = \begin{pmatrix} 1 & -2 & -4 \end{pmatrix}^T$, $b = \begin{pmatrix} -1 & -1 & -4 & 10 \end{pmatrix}^T$ and

$$A = \begin{pmatrix} 1 & -1 & 0 \\ 3 & 0 & -2 \\ -2 & 0 & 5 \\ 1 & 1 & -8 \end{pmatrix}$$

attaining the standar form

$$\begin{array}{ll} min & c^T x \\ subject & Ax \leq b \\ & x_i \leq 0 \end{array}$$

Exercise 3

In mathematical optimization theory, duality means that optimization problems may be viewed from either of two perspectives, the primal problem or the dual problem (the duality principle). The solution to the dual problem provides a lower bound to the solution of the primal (minimization) problem. However in general the optimal values of the primal and dual problems need not be equal. Their difference is called the duality gap. Consider

$$\begin{aligned} L(x, \lambda_1, \lambda_2) &= c^T x + \lambda_1^T (Ax - b) + \lambda_2^T x \\ &= \left(c^T + \lambda_1^T A + \lambda_2^T\right) x - \lambda_1^T b \end{aligned}$$

this is a linear funtion on x. We have that

$$\inf_x L(x, \lambda_1, \lambda_2) = \begin{cases} -\lambda_1^T b & \text{if } \left(c^T + \lambda_1^T A + \lambda_2^T\right) = 0 \\ -\infty & \text{otherwise} \end{cases}$$

the dual problem would be

$$\begin{array}{ll} max & -\lambda_1^T b \\ subject & \lambda_1, \lambda_2 \geq 0 \\ & \left(c^T + \lambda_1^T A + \lambda_2^T\right) = 0 \end{array}$$

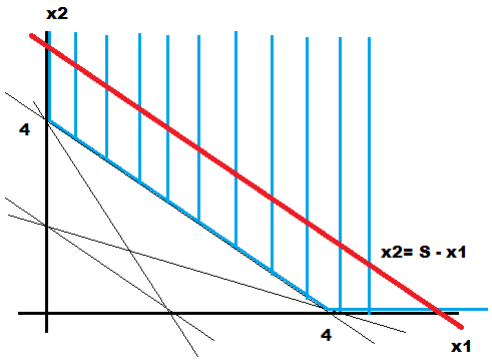

FIGURE 0.1. Feasible Space

EXERCISE 4

The solution for the problem involving $c = \begin{pmatrix} 1 & 1 \end{pmatrix}^T$, $\tilde{b} = \begin{pmatrix} -2 & -4 & -2 & -4 \end{pmatrix}^T$ has the same solution because the new constraint (first row of A) is redundant.

$$x_2 \geq -x_1 + 2, \;\; x_2 \geq -x_1 + 4, \;\; x_2 \geq -0.5x_1 + 2, \;\; x_2 \geq -2x_1 + 4$$

As the objective function

$$s = x_1 + x_2$$

has the same slope as the border of the feasible region it is straightforward to note that the minimum corresponds to

$$4 = x_1 + x_2$$

which is non unique. If the problem is well posed the minimum of a linear programming problem is attained in one of the vertex of the polygon given by the constraints (the feasible area).

Preparation for assignment 5

Machine Learning, Summer term 2014, Ulrike von Luxburg

CVX optimization package for MATLAB (unfortunately NOT compatible with Octave)

CVX is a Matlab-based modeling system for convex optimization. CVX turns Matlab into a modeling language, allowing constraints and objectives to be specified using standard Matlab expression syntax.
You can download CVX from `http://cvxr.com/cvx/download/`. Installation is relatively simple: unpack the distribution to an empty directory, and then run `cvx_setup` in this directory from the MATLAB command line. For more information, see `http://cvxr.com/cvx/doc/install.html`.
After installation, you can use CVX commands in your matlab files. To use CVX effectively, you need to know at least a bit about convex optimization. To use CVX for solving your optimization problem, you need to

- check if your problem indeed is a convex optimization problem. CVX is not meant to be a tool for checking if your problem is convex
- declare optimization variables, describe the objective function and describe the constraints according to the follwing scheme

```
cvx_begin
    variable x(n)
    minimize ( f(x) )
    subject to
        g(x) <= 0
        h(x) == 0
cvx_end
```

Example 1: Consider the following convex optimization problem

$$\begin{array}{rc} \text{minimize} & \|Ax - b\|_2 \\ \text{subject to} & Cx = d \\ & \|x\|_\infty \leq e \end{array}$$

The following code segment generates and solves a random instance of this model:

```
m = 20; n = 10; p = 4; A = randn(m,n); b = randn(m,1);
C = randn(p,n); d = randn(p,1); e = rand;
cvx_begin
    variable x(n)
    minimize( norm( A * x - b, 2 ) )
    subject to
        C * x == d
        norm( x, Inf ) <= e
cvx_end
```

Example 2: You can use CVX to solve soft margin SVM classification problems

$$\begin{array}{ll} \min\limits_{\mathbf{w},\xi,b} & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{i=1}^m \xi_i \\ & y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x_i} + b) \geq 1 - \xi_i \quad i = 1, \ldots, m \\ & \xi_i \geq 0 \qquad\qquad\quad i = 1, \ldots, m \end{array}$$

To solve this problem in matlab using CVX, first you need to initialize C, n, m, vector y and matrix X. Then

```
cvx_begin
    variables w(n) b xi(m)
    minimize 1/2*sum(w.*w) + C*sum(xi)/m
    y.*(X*w + b) >= 1 - xi;
    xi >= 0;
cvx_end
```

- You can keep CVX quiet during optimization by `cvx_quiet(true);`
- Documentation: `http://cvxr.com/cvx/doc/CVX.pdf`