Exercise 3: Linear Support Vector Machines

I. Goal of the exercise

The main goal is to understand the SVM classifier and the particularities of the solution achieved. For that purpose we will code the linear SVM from scratch using a QP solver at the optimization step.

II. Deliverables

As you progress in this exercise, you will find several questions. You are expected to answer them properly with adequate figures when required and deliver a document with all these evidences in due time. A file or files with the working code used for generating and discussing the results must be also delivered.

III. Code design

All classifiers must be coded from scratch, no toolkits are allowed to be used except for CVX (see details in the next section). Each classifier must have two basic interfaces: one for training and another for testing. The format of the interfaces is:

```
function [model, other_values]= train_<name_of_classifier> (labels, data, params)
function [class_label, other_values]= test_<name_of_classifier> (data, model)
```

The model can be built as you like and stores all the necessary information for it to be applied, e.g. in an linear SVM, the model stores λ and the support vectors.

IV. Introduction to the tools

The support vector machine defines a quadratic programming problem. Solving efficiently the problem is out of the scope of this course. For this reason, you are welcomed to use CVX convex optimization toolbox (http://cvxr.com/cvx/). This toolbox allows to solve many convex

optimization problems of medium size, i.e. linear, quadratic, cone, geometric and semi-definite programming problems. The toolbox allows to use user friendly notation for writing the convex optimization problem. E.g. given the following problem

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

the next CVX code corresponds to the optimization of the former problem:

```
cvx_begin
    variable x(n)
    minimize( norm( A * x - b, 2 ) )
    subject to
        C * x == d
        norm( x, Inf ) <= e
cvx_end
```

Please, refer to the reference documentation for additional details.

V. Toy problems creation

In order to create your own toy problems, you can use the following code:

```
figure;axis([-1 1 -1 1]);hold on;
i=1;
button=1;
while button==1
[x(i,1),x(i,2),button]=ginput(1);
plot(x(i,1),x(i,2),'rx');
i=i+1;
end

i=1;
```

```
button=1;
while button==1
[y(i,1),y(i,2),button]=ginput(1);
plot(y(i,1),y(i,2),'b.');
i=i+1;
end

data=[x', y'];
labels=[-ones(1,size(x,1)), ones(1,size(y,1))]';

save('toy_dataset.mat','data','labels');
```

VI. Understanding Support Vector Machines

A. Understanding the primal

Code the primal problem for hard margin linear SVM (i.e. without slack variables) using CVX. It is recommended to clear your workspace before running the CVX code.

Question block 1:

1) Load the dataset `'example_dataset_1.mat'`.
2) Run your training algorithm on that dataset.
3) Plot the dataset and the separating hyperplane.
4) Identify the support vectors and explain how you know they are support vectors.

Code the primal problem for the soft-margin linear SVM using CVX.

Question block 2:

1) Load the dataset `'example_dataset_1.mat'`.
2) Consider the soft-margin formulation for $\lambda = 0$. Is it reasonable to think that the resulting hyperplane should be very similar to the case of hard-margin SVM in a separable problem? Why?
3) Run the training algorithm for non-separable data sets with $\lambda = 0$. Plot the dataset and the separating hyperplane. What do you observe? Hypothesize a reasonable explanation.
4) Plot the dataset and the separating hyperplane when training with $\lambda = 1e-2$, $\lambda = 1$ and $\lambda = 1e2$.
5) Which is the expected value of u_i for the support vectors with margin equals 1?
6) Observe the values of u_i for $\lambda = 1e2$. Can you identify the SVs by observing those values? Describe the rule you use for identifying them.

Question block 3:

1) Load the dataset `'example_dataset_2.mat'`.
2) Run your training algorithm on that dataset for $\lambda = 1e-2$, $\lambda = 1$ and $\lambda = 1e2$.
3) Plot the dataset and the separating hyperplane for each value of lambda.
4) Observe the values of u_i for $\lambda = 10$. Can you identify the SVs simply by observing those values? Describe the rule you use for identifying them.

B. Understanding the dual

Code the dual problem for linear SVM using CVX. Be careful when analyzing the values of ν and use an appropriate tolerance.

Question block 4:

1) Load the dataset `'example_dataset_2.mat'`.
2) Run your training algorithm on that dataset for $\lambda = 1e-2$, $\lambda = 1$ and $\lambda = 1e2$.
3) Plot the dataset and the separating hyperplane for each value of lambda.
4) Which are the expected values of ν_i for the SVs?
5) Observe the values of ν_i for $\lambda = 10$. Can you identify the SVs simply by observing those values? Describe the rule you use for identifying them.
6) Is it easier to identify the SVs in the primal or in the dual? Why?

C. Unbalanced data

In this section we will vary the formulation of SVM to handle unbalanced data. Suppose that in this problem data with label -1 is critical, e.g. examples from a rare disease, machine failure, etc.

Question block 5:

1) Load the dataset `'example_dataset_3.mat'`.
2) Check how many examples we have for each class. Is the problem unbalanced? Why?
3) Search for the optimum value of λ for this problem. Plot the separating hyperplane and justify your choice of λ.
4) Is the result satisfying? Why?
5) Compute and report the training error rate.

In general, the cost of an error in a critical class is set by the user/client according to the application. However if we don't have *a priori* knowledge it is sensible to use a balancing weight so that an error on the majority class has less importance. A possible balancing weight is to use the quotient between the cardinality of the minority class set over the cardinality of the majority class set.

Question block 6:

1) Modify your code to take into account the balancing weight. Hint: instead of using the compact constraint $y_i(a^T x_i + b) \geq 1 - u_i$, use separate constraints for the class $+1$ and -1. Look at the first slides regarding *A convex optimization view to linear classification*.
2) Search for the optimum value of λ for this problem. Plot the separating hyperplane and justify your choice of λ.
3) Is the result satisfying? Why?
4) Compute and report the error rate. Is this error rate smaller than the one obtained in block 5? Why?
5) Use the balancing weight to define a weighted error rate and compute the weighed training error rates for the models in block 5 an 6. Is this error rate smaller than the weighed error for the model in block 5? Why?