Supervised classification exercise

I. Goal of the exercise

In this exercise you will practice the complete pipeline of the supervised classification task. The main goal is to code some classifiers, understand the different formulation choices and practice the parameter tuning to ensure generalization. The exercise is divided in two different parts. First, SVM is to be coded in both its linear and kernelized versions. Then, the team must select one additional classifier from the ones proposed and compare it with SVM.

The classifiers proposed are:

- [**Optional:** Code and evaluate Adaboost.] {diff: medium}
- [**Optional:** Code and evaluate a random forest.] {diff: medium to hard}
- [**Optional:** Code and evaluate a multi-class SVM.] {diff: easy}
- [**Optional:** Code your own QP solver for SVM (stochastic gradient descent is recommended).] {diff: hard}

II. 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 SVM with RBF kernel, the model stores λ, σ and the support vectors.

III. 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 welcome 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.

```
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 details.

IV. Toy problems creation

In order to create 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');
```

V. Work packages

The following work packages are a guide for scheduling your work:

1) **Week 1:** Familiarize with CVX and code linear SVM. Create a couple of toy problems in order to assert that your code behaves as expected.
2) **Week 2:** Code the primal or dual SVM with RBF kernel. Create a couple of toy problems in order to assert that your code behaves as expected.
3) **Week 3:** Evaluate your codes in three binary datasets. Use the parsers coded for your last exercise on CBR. Report the E_{out} estimation using 10-fold cross-validation. Attention: there are two parameters to tune using additional validation strategies.
4) **Week 4:** Code and evaluate your optional code.

VI. Deliverables

- A report containing the following Sections: Title, Abstract, Introduction, Specification of the problem and organization of the software, Experiments, Conclusions and Future Work and Bibliography. Optional: The report can be shaped as a journal paper in LaTeX following the IEEE template.
- Your code.

VII. What results should we report?

The experiments sections should contain at least:

- Table comparing performances. This includes, for instance, error rate and standard deviation.
- Graphical presentation of the information in the previous table.
- Graphical representation of the error surfaces when changing the parameters of your model.