MLCC Laboratory 2: Regularization networks

This lab is about Kernel Regularized Least Squares (KRLS). Follow the instructions below. Think hard before you call the instructors or you look at the solution file!

1 Kernel Regularized Least Squares (KRLS)

- **1.A** Generate a 2-class training set by using the following code:

```
[Xtr, Ytr] = MixGauss([[0;0],[1;1]], [0.5,0.3], 100);
[Xts, Yts] = MixGauss([[0;0],[1;1]], [0.5,0.3], 100);
Ytr(Ytr==2) = -1; Yts(Yts==2) = -1;
```

- **1.B** Have a look at the code of functions `regularizedKernLSTrain` and `regularizedKernLSTest`. Use the former to generate a decision function:

```
c = regularizedKernLSTrain(Xtr, Ytr, 'gaussian', sigma, lambda);
```

- **1.C** Check how the separating function changes with respect to `lambda` and `sigma`. Use the Gaussian kernel (`kernel='gaussian'`) and try some regularization parameters, with `sigma` in `[0.1, 10]` and `lambda` in `[1e-10, 10]`. To visualize the separating function (and thus get a more general view of what areas are associated with each class) you may use the routine `separatingFKernRLS` (type "`help separatingFKernRLS`" in the Matlab shell, if you still have doubts on how to use it, have a look at the code).

- **1.D** Perform the same experiment using flipped labels (`Ytr_noisy = flipLabels(Ytr, p)`) with `p` equal to 5% and 10%. Check how the separating function changes with respect to `lambda`.

- **1.E** Load the Two moons dataset by using the command `[Xtr, Ytr, Xts, Yts] = two_moons(npoints, pflipped)` where `npoints` is the number of points in the dataset (between `1` and `100`) and `pflipped` is the percentage of flipped labels. Then visualize the training and the test set by the following lines:

```
figure; scatter(Xtr(:,1), Xtr(:,2), 50, Ytr, 'filled');
figure; scatter(Xts(:,1), Xts(:,2), 50, Yts, 'filled');
```

- **1.F** Perform the exercises 1.C and 1.D on this dataset.

2 Parameter selection

- **2.A** By using the dataset in 1.E with `100` points and 5% flipped labels, select the suitable `lambda`, by using `holdoutCVKernRLS` (see `help holdoutCVKernRLS` for more information), and the sequence:

```
intKerPar = 0.5;
intLambda = [5, 2, 1, 0.7, 0.5, 0.3, 0.2, 0.1, 0.05, 0.02, 0.01, 0.005,
    0.002, 0.001, 0.0005, 0.0002, 0.0001,0.00001,0.000001];
nrip = 51;
perc = 0.5;
```

  Then plot the validation error and the training error with respect to the choice of `lambda` by the following code (the x-axis has a logarithmic scale):

```
semilogx(intLambda, Tm, 'r');
hold on; semilogx(intLambda, Vm, 'b');
hold off; legend('Training','Validation');
```

- **2.B** Perform the same experiment for different fraction of flipped labels `(0.0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5)`. Check how the training and validation error change with different `p`.

- **2.C** Now select the suitable `sigma`, by using `holdoutCVKernRLS`, on the following collection, as in exercise 2.A. Then plot the validation and test error:

```
intKerPar = [10, 7, 5, 4, 3, 2.5, 2.0, 1.5, 1.0, 0.7, 0.5, 0.3, 0.2, 0.1,
     0.05, 0.03, 0.02, 0.01];
intLambda = 0.00001;
nrip = 51;
perc = 0.5;
```

- **2.D** Perform the same experiment for different fraction of flipped labels `{0.0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5}`. Check how the training and validation error change with different `p`.

- **2.E** Now select the best `lambda` and `sigma`, by using `holdoutCVKernRLS`, on the following collection, as in exercise 2.A:

```
intKerPar = [10, 7, 5, 4, 3, 2.5, 2.0, 1.5, 1.0, 0.7, 0.5, 0.3, 0.2, 0.1,
     0.05, 0.03, 0.02, 0.01];
intLambda = [5, 2, 1, 0.7, 0.5, 0.3, 0.2, 0.1, 0.05, 0.02, 0.01, 0.005,
    0.002, 0.001, 0.0005, 0.0002, 0.0001,0.00001,0.000001];

nrip = 7;
perc = 0.5;
```

  Then plot the separating function computed with the best `lambda` and `sigma` you have found (use `separatingFKernRLS`).

- **2.F** Compute the best `lambda` and `sigma`, and plot the related separating functions with 0%, 5%, 10%, 20%, 30%, 40% of flipped labels. How do the parameters differ, and the curves?

3 If you have time - more experiments

- **3.A** Repeat the experiment in part 2, with less points `(20, 30, 50, 70)` and 5% of flipped labels. How do the parameters vary with respect to the number of points?

- **3.B** Repeat the experiment in part 1 with the polynomial kernel (`kernel = 'polynomial'`) and with parameters `lambda` in the interval `[0, 10]` and `t`, the exponent of the polynomial kernel, in `{1,2,...,10}`.

- **3.C** Perform the 2.F with the polynomial kernel and the following range of parameters:

```
IntKerPar = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
intLambda = [5, 2, 1, 0.7, 0.5, 0.3, 0.2, 0.1, 0.05, 0.02, 0.01, 0.005,
    0.002, 0.001, 0.0005, 0.0002, 0.0001,0.00001,0.000001];
```

  What is the best exponent for the polynomial kernel on this problem? Why?

- **3.D** Analyze the eigenvalues of the Gram matrix for the polynomial kernel with different values of `t` (plot them by using `semilogy`). What happens with different `t`? Why?

- **3.E** At the heart of the KRLS (see `regularizedKernLSTrain`) is the following least squares problem:

$$\mathbf{c} = (\mathbf{K} + \lambda \mathbf{I}_{N \times N}) \backslash \mathbf{y}. \tag{1}$$

  Solving the problem above for a single value of `lambda` takes $O(N^3)$ operations, if $\mathbf{K}$ is of size $N \times N$. In this work, we solve the problem above for multiple values of `lambda`. Do we have to pay $O(N^3)$ for each `lambda`? Given multiple values for `lambda`, is there a faster way to solve the problem above once we have solved it for the first value of `lambda`? **Hint:** Any symmetric matrix $\mathbf{S}$ has an eigenvalue decomposition as $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$.