CMPUT 466/566 — Programming Exercise 4 (Grad)

Instructor: R Greiner
Due Date: 5pm, Wed, 20/Feb/2019
The following exercises are intended to further your understanding of SVMs and "Dual"ity.
Relevant reading: [HTF: Chapters 5.8 and 12 (esp 12.3)]
Only grad students are required to do this exercise; undergrads should instead solve the Coursera programming exercise; see the eClass page.

Recall that the primal problem for SVM is:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\mathbf{w}\|^2 \ + \ C_1 \sum_{i=1}^{m} \xi_i \tag{1}$$

subject to

$$y_i({\mathbf{x}_i}^T\mathbf{w} + b) \ \geq \ 1 - \xi_i, \quad (i = 1, \ldots, m) \tag{2}$$

$$\xi_i \ \geq \ 0, \quad (i = 1, \ldots, m) \tag{3}$$

while the Dual form is:

$$\max_{\boldsymbol{\alpha}} \quad C_2 \sum_{i=1}^{m} \alpha_i \quad - \quad \frac{1}{2} \sum_{i,j} \alpha_i \, \alpha_j \, y_i \, y_j \, {\mathbf{x}_i}^T \, \mathbf{x}_j \tag{4}$$

subject to

$$0 \leq \alpha_i \leq 1, \quad (i = 1, ..., m) \tag{5}$$

$$\sum_{i=1}^{m} \alpha_i y_i \quad = \quad 0 \tag{6}$$

In this assignment, you will implement both, the dual and primal versions of the SVM with a linear kernel, as well the SVM with RBF and Polynomial kernels (dual form). At the same time, you will develop the entire procedure for:

- Creating a classifier that can be used in an external dataset, with parameters selected using (internal) 5-fold cross validation.
- Estimate the performance of that classifier using (external) 5-fold cross validation.

We have provided some template functions that will guide you through this assignment. You can create more functions to organize your code if you want, but you cannot change the input/output arguments of the functions that we provide. You will need to complete the following functions:

- `trainSVM_model.m` : Given a labeled training set, a kernel, and a set of parameters for that kernel (*e.g.*, Linear Kernel, C = 1), it will estimate the weight vector (or support vectors) and the bias term needed by SVM. the `.m` file provides more information.

- `predictUsingSVM.m` : Given a weight vector (or support vector), a bias term, the parameters of the kernel, and a new set of instances, this subroutine will classify these new instances.
- `findBestParameters.m` : This subroutine takes, as input, a dataset *dataSamples*, its corresponding labels *dataLabels*, a kernel, and a vector *Folds* assigning each sample to a fold. Then it uses (internal) 5-fold cross validation to select the parameters that produces the highest cross-validation accuracy. Of course, it will use the two previous subroutines.
- `expectedAccuracy.m`: Given a labeled dataset (specified by a matrix *dataSamples* and a vector *dataLabels*), a kernel, and a vector/matrix for assigning each sample to a fold (for nested cross validation), this subroutine will estimate the expected accuracy of your learning algorithm using (nested) 5-fold cross validation.
- `generateModelsToBeGraded.m` : This subroutine will train a SVM with the different kernels and you will save a series of outputs that we will use to grade your assignment. The `.m` file provides more details.

1 Datasets

We will provide you with three different datasets:

- `Iris1_v24`: contains 120 2D instances (X) with labels (Y)
- `Chessboard`: contains 240 2D instances (X) with labels (Y)
- `Survival`: contains 120 2D instances (X) with labels (Y)

You will create 4 classifiers for each dataset – each using a different kernel: linear_primal, linear_dual, RBF, and Polynomial kernels.

Note: For `Chessboard` and `Survival`, we will also test your code on a separate test set.

2 Quadprog

You might use the function `quadprog` in MatLab for solving the optimization problems, but may NOT use SVM toolboxes. This section will give you some useful hints for using this `quadprog`.

2.1 Primal problem

The implementation of support vector machines (SVM) requires solving a quadratic optimization problem (check equations $1-3$). `quadprog` solves this kind of problems; however, it might not be obvious how to translate your problem to the format required by this function.

We will explain the notation used in Matlab; however, this description will also apply to Octave *mutatis mutandis* – in particular, the name of the matrices will be different.

The optimization objective of the function quadprog, as described on the official website `http://www.mathworks.com/help/optim/ug/quadprog.html` is:

$$\begin{aligned}
\min_{\mathbf{z}} \quad & \frac{1}{2}\,\mathbf{z}^T H \mathbf{z} \;+\; \mathbf{f}^T \mathbf{z} && (7)\\
\text{subject to} \quad & A\mathbf{z} \;\leq\; \mathbf{b} && (8)\\
& A_{eq}\mathbf{z} \;=\; \mathbf{b}_{eq} && (9)\\
& \mathbf{l} \;\leq\; \mathbf{z} \leq\; \mathbf{u} && (10)
\end{aligned}$$

We can map our optimization objective to `quadprog's` as follows: Define a vector $\mathbf{w} = [w_1, w_2, \ldots, w_n]^T$, where n is the number of features in the training set. Define also $\boldsymbol{\xi} = [\xi_1, \xi_2, \ldots, \xi_m]^T$, where m is the number of samples in the training set. We can then create a new vector $\mathbf{z} = [\mathbf{w}, b, \boldsymbol{\xi}]^T$; note $\mathbf{z} \in \Re^{n+1+m}$. Having defined $\mathbf{z}$, it is straightforward to determine the values that the other matrices and vectors have to take. If a particular entry in $\mathbf{z}$ does not have an upper ($\mathbf{u}$) or lower ($\mathbf{l}$) bound, you can always set it to `-inf`, or `inf`.

The procedure is similar for the dual problem, but note that the variables to optimize are different!

3 Deliverables

You should upload to eclass:

1. Your well-commented Matlab files (`*.m`) and the `*.mat` files created.
2. A report stating the best parameters that you found, and the expected accuracy of your learned classifier for every dataset – you need to use the function `expectedAccuracy` here. You will need to include these results for the linear (primal and dual), RBF, and polynomial kernel. This report can be very informal – *e.g.*, just including a simple table reporting the name of the dataset, the best parameters found, and the expected accuracy (5-fold cv).
3. Plot the data points and separators using the plotting code provided within the folder "SVCplot_demo" for the dual form. There is also an example Matlab script that calls the appropriate plotting function. For the primal form, you can use the plot function in MATLAB. Note that the data is 2D. Include a plot, in `.jpg` file format, for each of the following 4×3 cases:

   `Q2-primal-{Iris1_v24,chess,survival}.jpg`

   `Q2-dual-lin-{Iris1_v24,chess,survival}.jpg`

   `Q2-dual-rbf-{Iris1_v24,chess,survival}.jpg`

   `Q2-dual-poly-{Iris1_v24,chess,survival}.jpg`

To submit your solution: create a directory named "`PE_4_Grad`" that contains all of your files, the report, and the images.

Submit only ONE zip file that contains the folder.