The objective of this lab is for you explore the behavior of several types of support vector machines (SVMs) for binary classification, and apply them to some datasets. The TA will first demonstrate the results of the algorithms on a toy dataset, and then you will replicate those results, and further explore the datasets with the algorithms.

We provide you with the following:

- The script `lab10.m` sets up the problem (toy dataset) and plots various figures.
- The function `linsvmtrain.m` trains a linear SVM (taking input vectors in D dimensions); it handles both the optimal separating hyperplane case and the soft margin hyperplane case. See also the function `linsvm.m`.
- The function `svm2train.m` trains a nonlinear (kernel) SVM (taking input vectors in D dimensions). See also the function `svm2.m`.

I Datasets

Construct your own toy datasets in 2D to visualize the result easily and be able to understand the algorithm, such as Gaussian classes with no or some overlap, or classes with curved shapes as in the 2moons dataset.

II Using SVMs

Review Most types of SVMs define a constrained optimization problem where the objective function is convex quadratic and the constraints (equalities or inequalities) are linear. Such problems are called convex quadratic programs (QPs). They have a unique solution, which can be found by solving either the original, primal QP, or the dual QP.

To solve a QP, we use Matlab's Optimization Toolbox, specifically the following two functions:

- `quadprog`: this solves any kind of QP. All we have to do is put the QP for the SVM (primal or dual) in the form required by `quadprog` (see `help quadprog`). As output arguments, it returns everything we need to construct the SVM discriminant function $g(\mathbf{x})$ (the optimal solution and/or its Lagrange multipliers).
- `optimoptions`: this is not strictly necessary, but we can use it to select which QP solver to use and various other options or parameters (what to display, the maximum number of iterations, whether we want to provide an initialization, etc.). A good choice is the active-set algorithm because (with small problems) it reliably identifies the support vectors:
```
options = optimoptions('quadprog','Algorithm','active-set','MaxIter',1000);
... = quadprog(...,options);
```
Annoyingly, Matlab has discontinued the active-set algorithm from version 2016a onwards. So we just use Matlab's default algorithm; however, it often misidentifies some support vectors.

Then, we implement the following types of SVM in `linsvmtrain.m` and `svm2train.m`, respectively:

Linear SVM It has two types:

- *For data that is linearly separable: the optimal separating hyperplane.* We can solve either the primal problem and get $(\mathbf{w}, w_0)$, and hence the discriminant $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$, or the dual problem and get the Lagrange multiplier for each data point, α_n, and from there obtain the support vectors (SVs), which have $\alpha_n > 0$, and construct $(\mathbf{w}, w_0)$.
- *For data that is not linearly separable: the soft margin hyperplane.* Now the QP uses a slack variable $\xi_n \geq 0$ to account for possible constraint violations (points correctly classified but within the margin, or points misclassified), and a hyperparameter $C > 0$ that controls the tradeoff between minimizing the total violations $\sum_{n=1}^{N} \xi_n$ and maximizing the margin $\frac{1}{\|\mathbf{w}\|}$ (i.e., minimizing $\frac{1}{2}\|\mathbf{w}\|^2$). Again, we can solve either the primal or the dual problem.

Nonlinear (kernel) SVM You must choose what kernel $K(\mathbf{x}, \mathbf{y})$ to use (polynomial, Gaussian, etc.). In addition to C, the kernel may have its own hyperparameters (e.g. the degree q for the polynomial kernel or the width σ

for the Gaussian kernel). With nonlinear SVMs we can only solve the dual QP, obtain the Lagrange multipliers $\alpha_1, \ldots, \alpha_N \geq 0$, and from there construct the nonlinear discriminant:

$$g(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}) = \sum_{n \in \text{SVs}} \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}).$$

In either case (linear or nonlinear SVM), the discriminant function $g(\mathbf{x}) \in \mathbb{R}$ classifies an instance into the $+1$ or -1 class according to the sign of $g(\mathbf{x})$.

The hyperparameters C and (for kernel SVMs) q or σ are set by cross-validation over a suitable range of values; for example, try $C \in \{10^{-2}, 10^{-1}, 10^{0}, 10^{1}\}$ and $q \in \{1, 2, 3, 5, 10\}$. Plot the resulting SVM classifier for different hyperparameter values and observe how it looks like.

Exploration: toy problem with a linear SVM Start with the simplest case: the optimal separating hyperplane. Find the SVM optimal parameters given the training set of instances $\{\mathbf{x}_n\}_{n=1}^N \subset \mathbb{R}^2$ and their class labels $\{y_n\}_{n=1}^N \subset \{-1, +1\}$, by running `linsvmtrain.m` with appropriate arguments. Inspect the output arguments it produces (SVM parameters $(\mathbf{w}, w_0)$, support vectors, Lagrange multipliers $\alpha_1, \ldots, \alpha_N \geq 0$, etc.). We visualize the results with the following plots:

- The dataset in 2D, i.e., each training point $\mathbf{x}_n$ colored according to its class label $y_n \in \{-1, +1\}$.
- The SVM discriminant function $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$, using `contour`. The discrimination boundary corresponds to the contour $g(\mathbf{x}) = 0$. (We can also plot the line $\mathbf{w}^T\mathbf{x} + w_0 = 0$ directly, but the other contours are informative with nonlinear SVMs.)
- The margin, indicated by the SVs on each side of the separating hyperplane.

Consider the following questions:

- Verify that solving the primal QP and the dual QP give the same result (same $(\mathbf{w}, w_0)$, Lagrange multipliers, SVs, margin).
- Verify that the margin (i.e., the distance from the hyperplane to its closest instance) is $\frac{1}{\|\mathbf{w}\|}$.
- When training the optimal separating hyperplane, what happens if the data is not linearly separable? *Hint*: look at the error code returned by `quadprog`.

Once you understand the case of the optimal separating hyperplane well, proceed with the soft margin hyperplane:

- What is the effect on the discrimination boundary and the SVs of varying the value of $C \in (0, \infty)$?
- If the data is linearly separable, does it give the same result as the optimal separating hyperplane case? Why?

Exploration: toy problem with a nonlinear SVM Proceed as with the linear SVM, but now running `svm2train.m` with appropriate arguments. What is the effect on the discrimination boundary and the SVs of varying the value of C? How about the value of the kernel hyperparameter (q, σ)? Determine the best hyperparameter values (C, q) or (C, σ) by using cross-validation.