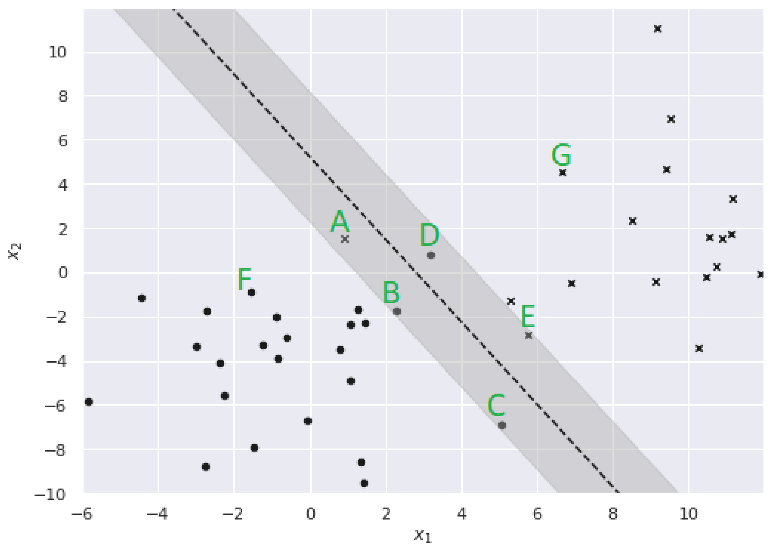
AML 5203 | Machine Learning Principles & Applications | Sessional-1 Solutions

1. [10 points] [L5, CO 1] Answer the questions based on the SVM linear decision boundary shown below:



(a) Justify briefly if the data is linearly separable or not.

Solution: The data is not linearly separable because no hyperplane (a line here) can separate the two classes as shown.

(b) How many support vectors are there?

Solution: The 5 samples A, B, C, D, and E are the support vectors.

(c) For each sample $A - G$, choose one of the following for the slack ξ with a brief justification as to why:

$$\xi = 0 \text{ or } 0 < \xi < 1 \text{ or } \xi \geq 1.$$

Solution:

B, C, E: $\xi = 0$ because these are support vectors lying on the margin boundary on the correct side.

G, F: $\xi = 0$ because they are on the correct side and outside the margin.

A, D: $\xi \geq 1$ because they are on the wrong side of the hyperplane.

(d) Calculate the full-margin width if the equation of the separating line (hyperplane) is $x_2 = -1.8x_1 + 5$.

Solution: Rewriting the equation of the separating line as $1.8x_1 + x_2 - 5 = 0$ and comparing with the equation of a hyperplane $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0$, we identify $\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 1.8 \\ 1 \end{bmatrix}$ and $b = -5$. The full-margin width is $2/\|\mathbf{w}\| = 2/\sqrt{w_1^2 + w_2^2} = 2/\sqrt{1.8^2 + 1^2} \approx 1$.

2. [10 points] [L5, CO 2] Consider solving the MNIST classification problem (labels 0 through 9) using the hinge loss function-based formulation of SVM.

(a) Fill in the missing entries below in the ith sample's loss:

$$L_i = \sum_{\substack{j=? \\ j \neq y^{(i)}}}^{\boxed{?}} \max\left(0, z_j^{(?)} - z_{\boxed{?}}^{(i)} + \boxed{?}\right)$$

Solution:

$$L_i = \sum_{\substack{j=1 \\ j \neq y^{(i)}}}^{\boxed{2}} \max\left(0, z_j^{(i)} - z_{\boxed{y^{(i)}}}^{(i)} + \boxed{1}\right).$$

(b) Rewrite the ith sample's loss in terms of the weight vectors assuming that the bias trick is done.

Solution:

$$L_i = \sum_{\substack{j=1 \\ j \neq y^{(i)}}}^{2} \max\left(0, w_j^{\mathrm{T}} x^{(i)} - w_{y^{(i)}}^{\mathrm{T}} x^{(i)} + 1\right).$$

(c) What is the computational advantage of using a loss function-based formulation of SVM over the optimization-based formulation of SVM (such as SVC or LinearSVC)?

Solution: The loss function-based formulation of SVM allows for batch processing which can be used to overcome the limitation of SVM that it is computationally expensive when the number of samples is large.

(d) The SVM algorithm presented here results in a linear decision boundary similar to the hyperplane generated by the optimization-based formulation of linear SVM. How can such a linear model over-fit? How can you avoid over-fitting? Justify using one or two lines for both questions.

Solution: A linear model can over-fit when the number of features is large. Overfitting can be avoided by using regularization.

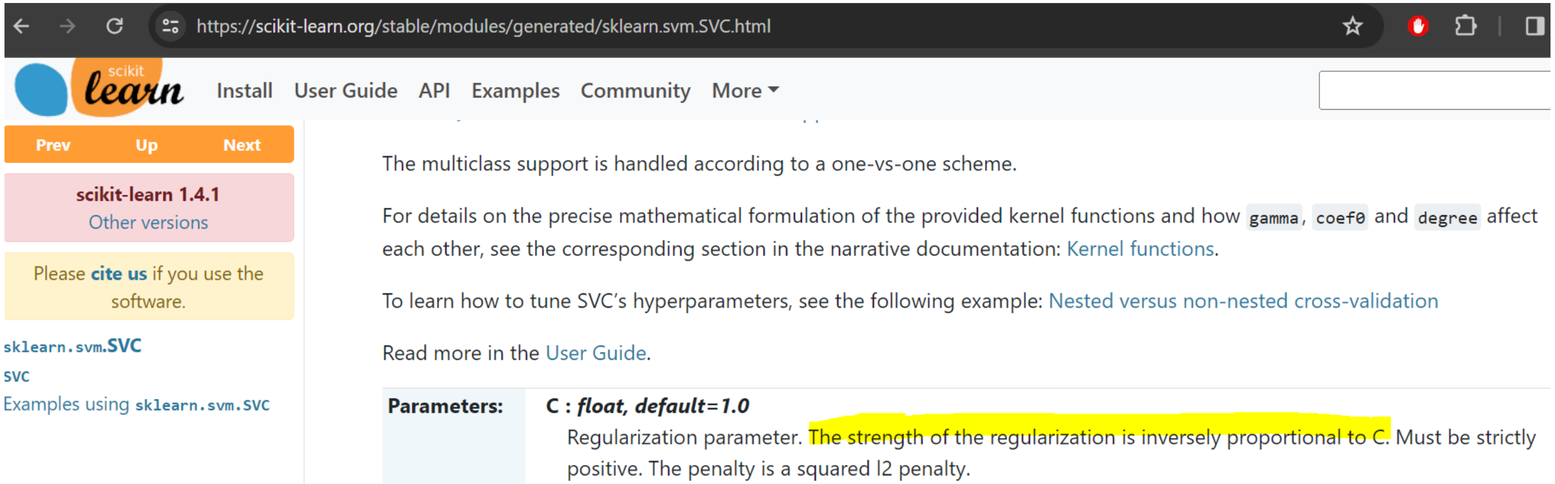
3. [10 points] [L5, CO 2] Select the correct option in each of the following: a large value of the SVC hyperparameter C results in

- more/less misclassifications;
- more/less regularization;
- more/less overfitting;
- more/less number of support vectors;
- a decision boundary that is close to linear/nonlinear.

Solution: Note that the effect of the hyperparameter C is different in the pen-and-paper soft-margin SVM formualtion

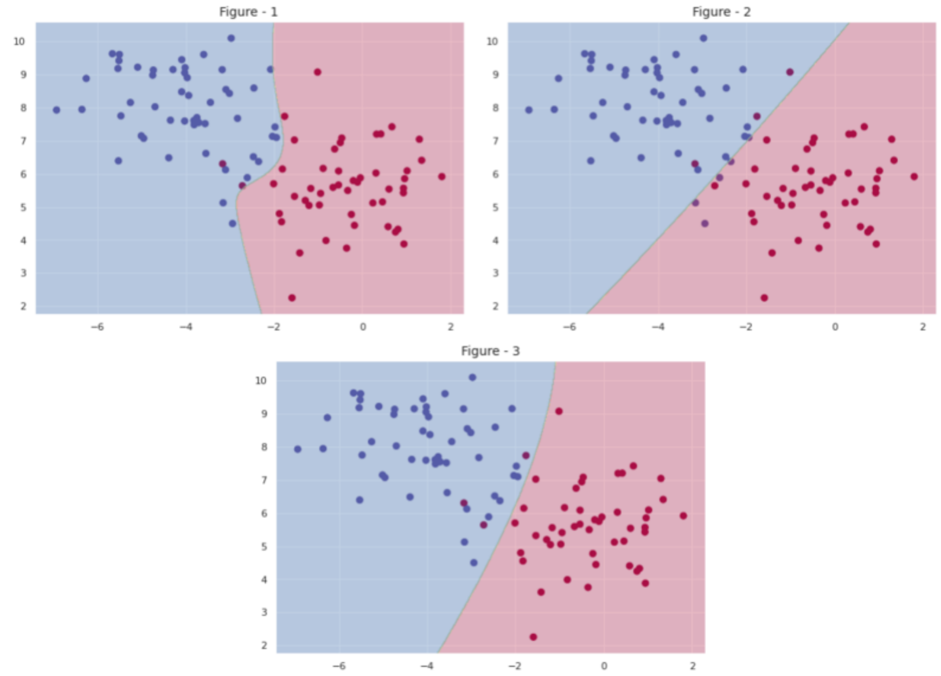
$$\min \frac{\|\mathbf{w}^2\|}{2} + C\sum_{i=1}^{n} \xi^{(i)}$$

compared to how it is implemented in scikit-learn SVC. In the pen-and-paper form, increasing $C = 0$ means we allow no slack and as C increase we allow for more slack (that is, regularization). Where as in scikit-learn SVC, regularization is inversely proportional to C as indicated in the website snapshot below:



- more/less misclassifications;
- more/less regularization;
- more/less overfitting;
- more/less number of support vectors;
- a decision boundary that is close to linear/nonlinear.

4. [10 points] [L2, CO 3]] Identify and match the Kernel-SVC hyperparameter values $C = 10^5, 10^1$, and 10^{-1} with the corresponding figures below with a brief justification:



Solution: A large value of C in scikit-learn SVC indicates less regularization and a more non-linear decision boundary where as a small value of C indicates more regularization and a close-to-linear decision boundary. Figure-1 has $C = 10^5$, Figure-2 has $C = 10^{-1}$, and Figure-3 has $C = 10^1$.

5. [10 points] [L5, CO 3] Consider a dataset where a generic sample $\mathbf{x}$ has 2 features and can correspond to 2 output labels. Your friend tries to fit a linear SVM model to the dataset and concludes that introducing new features as follows would be needed for a more accurate classification model:

$$\mathbf{x}_{\text{old}} = \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{\text{old features}} \mapsto \mathbf{x}_{\text{new}} = \underbrace{\begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}}_{\text{new features}}.$$

(a) Argue briefly as to why your friend is introducing these new features. What is a potential disadvantage in your friend's approach?

Solution: Your friend is introducing the new features because they think that the data is not linearly separable and will need more features to build the non-linear decision boundary.

(b) Which form of SVM – the primal or dual – is your friend using. Explain briefly.

Solution: The dual form because in the dual form, the similarity between the samples in the higher dimensional space can be calculated as dot products involving the low-dimensional features using the kernel trick.

(c) Compute the dot product between samples i and j in the the new feature space: $\mathbf{x}_{\text{new}}^{(i)} \cdot \mathbf{x}_{\text{new}}^{(j)}$.

Solution:

$$\mathbf{x}_{\text{new}}^{(i)} \cdot \mathbf{x}_{\text{new}}^{(j)} = \begin{bmatrix} \left(x_1^{(i)}\right)^2 \\ \sqrt{2}x_1^{(i)}x_2^{(i)} \\ \left(x_2^{(i)}\right)^2 \end{bmatrix} \cdot \begin{bmatrix} \left(x_1^{(j)}\right)^2 \\ \sqrt{2}x_1^{(j)}x_2^{(j)} \\ \left(x_2^{(j)}\right)^2 \end{bmatrix} = \left(x_1^{(i)}x_1^{(j)}\right)^2 + 2x_1^{(i)}x_2^{(i)}x_1^{(j)}x_2^{(j)} + \left(x_2^{(i)}x_2^{(j)}\right)^2.$$

(d) Show that the dot product result you derived in the previous part can also be efficiently computed by first computing $\mathbf{x}_{\text{old}}^{(i)} \cdot \mathbf{x}_{\text{old}}^{(j)}$ in the old feature space and making use of that result. This is the kernel trick.

Solution:

$$\underbrace{\mathbf{x}_{\text{new}}^{(i)} \cdot \mathbf{x}_{\text{new}}^{(j)}}_{=\left(x_1^{(i)}x_1^{(j)}\right)^2 + 2x_1^{(i)}x_2^{(i)}x_1^{(j)}x_2^{(j)} + \left(x_2^{(i)}x_2^{(j)}\right)^2} = \underbrace{\left(\mathbf{x}_{\text{old}}^{(i)} \cdot \mathbf{x}_{\text{old}}^{(j)}\right)^2}_{=\left(\begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \end{bmatrix} \cdot \begin{bmatrix} x_1^{(j)} \\ x_2^{(j)} \end{bmatrix}\right)^2 = \left(x_1^{(i)}x_1^{(j)} + x_2^{(i)}x_2^{(j)}\right)^2}.$$