**MACHINE LEARNING ASSIGNMENT\_20**

**1.What is the underlying concept of Support Vector Machines?**

The underlying concept of Support Vector Machines (SVMs) is to find the optimal hyperplane that maximally separates the classes in a high-dimensional space. SVMs use a kernel function to transform the input data into a higher-dimensional feature space, where it is easier to separate the classes with a linear boundary. The hyperplane is then chosen as the one that has the largest margin, i.e., the maximum distance to the closest points of each class. SVMs can be used for classification and regression tasks and are known for their ability to handle high-dimensional data and non-linear decision boundaries.

**2. What is the concept of a support vector?**

In machine learning, a support vector is a data point in a training dataset that is closest to the decision boundary or hyperplane that separates different classes or categories. Support vectors are used in support vector machines (SVMs), which are a type of supervised learning algorithm used for classification and regression tasks. The position and number of support vectors determine the location and orientation of the decision boundary, and thus, have a significant impact on the performance and accuracy of an SVM.

**3. When using SVMs, why is it necessary to scale the inputs?**

Scaling the inputs is necessary when using SVMs because SVMs are sensitive to the scale of the features. When the features are on different scales, the SVM may assign disproportionate importance to the features with larger scales, and may not perform well on the features with smaller scales. Scaling the inputs ensures that all the features have similar scales, and prevents the SVM from being biased towards any particular feature. This can improve the accuracy and performance of the SVM.

**4. When an SVM classifier classifies a case, can it output a confidence score? What about a percentage chance?**

Yes, SVM classifiers can output a confidence score that represents the distance between the input data point and the decision boundary. This confidence score is often used as a measure of how confident the classifier is in its prediction. However, SVMs do not directly output a percentage chance or probability estimate of the input data belonging to a particular class. In order to obtain a probability estimate, additional techniques such as Platt scaling or isotonic regression can be applied to the confidence scores.

**5. Should you train a model on a training set with millions of instances and hundreds of features using the primal or dual form of the SVM problem?**

It depends on the specifics of the dataset and the computational resources available.

In general, if the number of instances is much larger than the number of features, using the primal form of the SVM problem may be more efficient. On the other hand, if the number of features is much larger than the number of instances, using the dual form of the SVM problem may be more efficient.

However, when dealing with datasets that have millions of instances and hundreds of features, it is often a good idea to use methods that can handle such large-scale data, such as stochastic gradient descent (SGD) or mini-batch SGD. These methods can be more computationally efficient and are often better suited for online learning scenarios.

**6. Let’s say you’ve used an RBF kernel to train an SVM classifier, but it appears to underfit the training collection. Is it better to raise or lower (gamma)? What about the letter C?**

If the SVM classifier with an RBF kernel underfits the training data, then you should try increasing gamma. Gamma controls the kernel's influence, and increasing it makes the decision boundary more complex, which can help the SVM fit the data better.

As for the letter C, it is the regularization parameter that controls the trade-off between achieving a low training error and a low testing error. If the model is underfitting, it is possible that the value of C is too low, which means the model is too constrained. You can try increasing C to allow the model to fit the training data better, but be careful not to overfit the data by making C too large.

**7. To solve the soft margin linear SVM classifier problem with an off-the-shelf QP solver, how should the QP parameters (H, f, A, and b) be set?**

To solve the soft margin linear SVM classifier problem with an off-the-shelf QP solver, the QP parameters should be set as follows:

H: A positive definite matrix defined as XXT where X is the matrix of training examples and X^T is its transpose, multiplied by a regularization parameter C (or the inverse of C, depending on the convention used by the solver). That is, H = C \* XXT.

f: A vector of length n, where n is the number of training examples, containing only the value -1. This vector is multiplied by a regularization parameter C. That is, f = -C \* ones(n,1).

A: A matrix of size m x n, where m is the number of constraints (equal to 2n in the soft margin case), and n is the number of features in each training example. The rows of A correspond to the constraints, which are of the form y(i)(w^Tx(i) + b) >= 1 - xi, where xi is the slack variable associated with the ith training example. The elements of A are derived from the training data and depend on the label vector y and the feature matrix X.

b: A vector of length m, containing the values 1 or 0 depending on the type of constraint. Specifically, the first n elements of b are set to 1 (corresponding to the constraints y(i)(w^Tx(i) + b) >= 1 - xi), and the remaining n elements of b are set to 0 (corresponding to the constraints xi >= 0).

Once these parameters are set, the QP solver can be used to obtain the optimal values of the primal variables w and b, which define the decision boundary of the linear SVM classifier.

**8. On a linearly separable dataset, train a LinearSVC. Then, using the same dataset, train an SVC and an SGDClassifier. See if you can get them to make a model that is similar to yours.**

To train a LinearSVC on a linearly separable dataset, we can use the following code in scikit-learn:

from sklearn.svm import LinearSVC

from sklearn.datasets import make\_classification

from sklearn.model\_selection import train\_test\_split

X, y = make\_classification(n\_samples=100, n\_features=2, n\_informative=2, n\_redundant=0, random\_state=42)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

clf = LinearSVC()

clf.fit(X\_train, y\_train)

To train an SVC and an SGDClassifier on the same dataset, we can use the following code:

from sklearn.svm import SVC

from sklearn.linear\_model import SGDClassifier

clf\_svc = SVC(kernel='linear')

clf\_svc.fit(X\_train, y\_train)

clf\_sgd = SGDClassifier(loss='hinge')

clf\_sgd.fit(X\_train, y\_train)

To see if the models are similar, we can compare their accuracy and decision boundaries. We can use the test set to evaluate the accuracy of the models and plot their decision boundaries using matplotlib.

from matplotlib import pyplot as plt

fig, ax = plt.subplots(ncols=3, figsize=(12, 4))

titles = ['LinearSVC', 'SVC', 'SGDClassifier']

clfs = [clf, clf\_svc, clf\_sgd]

for i in range(3):

ax[i].scatter(X[:, 0], X[:, 1], c=y, cmap='bwr', alpha=0.5)

ax[i].set\_title(titles[i])

ax[i].set\_xlabel('Feature 1')

ax[i].set\_ylabel('Feature 2')

xlim = ax[i].get\_xlim()

ylim = ax[i].get\_ylim()

xx, yy = np.meshgrid(np.linspace(xlim[0], xlim[1], 100),

np.linspace(ylim[0], ylim[1], 100))

Z = clfs[i].decision\_function(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

ax[i].contour(xx, yy, Z, colors='k', levels=[-1, 0, 1], alpha=0.5,

linestyles=['--', '-', '--'])

acc = clfs[i].score(X\_test, y\_test)

ax[i].text(xlim[0] + 0.1, ylim[1] - 0.1, f'Accuracy: {acc:.2f}')

This will create a plot with three subplots, one for each model, showing the decision boundary and the accuracy. We can visually inspect the decision boundaries and compare the accuracy scores to determine if the models are similar.

**9. On the MNIST dataset, train an SVM classifier. You’ll need to use one-versus-the-rest to assign all**

**10 digits because SVM classifiers are binary classifiers. To accelerate up the process, you might want to tune the hyperparameters using small validation sets. What level of precision can you achieve?**

**10. On the California housing dataset, train an SVM regressor.**