Machine Learning #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 can separate two classes of data points with the largest possible margin. In other words, SVMs aim to find a decision boundary that maximizes the distance between the closest points of the two classes, which are called support vectors. SVMs can be used for both classification and regression tasks and are particularly effective in high-dimensional spaces. SVMs achieve this by transforming the input data into a higher-dimensional space and then finding the optimal hyperplane in that space. The decision boundary is determined by the parameters of the hyperplane, which are optimized by minimizing a loss function that penalizes misclassification of data points.

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

In a support vector machine (SVM) algorithm, ***support vectors are data points that are closest to the decision boundary (also called the hyperplane) that separates different classes in the dataset.*** The decision boundary is determined by maximizing the margin between the classes, which is the distance between the decision boundary and the nearest data points from each class.

Support vectors are critical for SVMs because ***they define the decision boundary and determine the classification accuracy of the model.*** In essence, the support vectors are the "support" for the decision boundary.

During the SVM training process, the algorithm identifies the support vectors that define the decision boundary and uses them to create the final model. The other data points that are further away from the decision boundary do not affect the model's construction.

In summary, support vectors are the data points that are closest to the decision boundary in an SVM algorithm, and they play a crucial role in defining the decision boundary and determining the accuracy of the model.

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

When using Support Vector Machines (SVMs), it is often necessary to scale the inputs to ensure optimal performance. This is because SVMs try to maximize the margin between the support vectors (data points closest to the decision boundary) and the decision boundary itself. The margin is defined by the distance between the support vectors and the decision boundary, and is sensitive to the scale of the input features.

If some features have a larger range of values than others, they will dominate the optimization process and contribute more to the decision boundary. As a result, the decision boundary will be biased towards the features with the larger values, leading to poor performance on other features.

By scaling the inputs to a common range, we can avoid this problem and ensure that all features contribute equally to the optimization process. This can result in better performance and more stable models. Common scaling methods include standardization (scaling to zero mean and unit variance) and normalization (scaling to a range of 0 to 1 or -1 to 1).

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

Yes, an SVM classifier can output a confidence score, which indicates how confidently it has classified a case. However, it does not provide a percentage chance or probability for the classification, unlike some other classification algorithms such as logistic regression or naive Bayes.

The confidence score for an SVM classifier is based on the distance between the decision boundary and the test instance. The larger the distance, the higher the confidence of the classifier in its prediction. The distance is also known as the margin, and SVM tries to maximize the margin while minimizing the classification error.

It is important to note that the confidence score of an SVM does not provide any information about the probability or likelihood of a class, but rather only indicates the confidence of the classification. If probabilities are needed, an SVM can be trained with a probabilistic output, but this requires modification to the standard SVM algorithm.

▼ 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?

When dealing with a large dataset with millions of instances and hundreds of features, it is generally recommended to use the dual form of the SVM problem rather than the primal form. This is because the dual form is more computationally efficient in this scenario, as it involves computing only the inner products between pairs of instances rather than the explicit computation of the feature vectors themselves, which can be memory-intensive and computationally expensive. Additionally, the dual form allows for the use of kernel functions, which can be used to transform the input data into a higher-dimensional space, potentially improving the separability of the data. Overall, the dual form of the SVM problem is typically the preferred approach for large-scale machine learning tasks involving SVMs.

▼ 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 RBF kernel underfits the training set, it means that the decision boundary is too simple and does not capture the complexity of the data. In this case, we should adjust the parameters of the SVM to increase the model's flexibility.

To do this, we can increase the value of gamma, which controls the width of the Gaussian kernel. A smaller gamma results in a wider kernel and a smoother decision boundary, while a larger gamma makes the kernel narrower and the boundary more complex.

Similarly, we can also increase the value of the hyperparameter C, which controls the trade-off between maximizing the margin and minimizing the classification error. A larger C allows more misclassifications in the training set but results in a smaller margin, while a smaller C leads to a larger margin but may allow more misclassifications.

In summary, to address underfitting in an SVM classifier with an RBF kernel, we can increase gamma and/or C to increase the model's flexibility and complexity.

▼ 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 using a QP solver, we need to set the QP parameters as follows:

1. H: The H matrix should be set to the identity matrix multiplied by a small constant (C) and then concatenated with zeros for the slack variables. That is, H should be of size (n_features + n_samples) x (n_features + n_samples), where the top left n_features x n_features block is the identity matrix multiplied by C, and the bottom right n_samples x n_samples block is the zero matrix. The other blocks are also zero.
2. f: The f vector should be set to all zeros of size (n_features + n_samples) x 1.
3. A: The A matrix should be set to the concatenation of the transpose of the feature matrix X and an identity matrix multiplied by -1, both multiplied by a constant (C). That is, A should be of size n_samples x (n_features + n_samples), where the left n_features columns are the transpose of X multiplied by -C, and the right n_samples columns are the identity matrix multiplied by -C.
4. b: The b vector should be set to all -1s of size n_samples x 1.

These QP parameters can be passed to an off-the-shelf QP solver to obtain the solution to the soft margin linear SVM problem.

▼ 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.

▼ 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.