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

2. What is the concept of a support vector?

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

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

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?

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?

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?

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.

Answer:

1. The underlying concept of Support Vector Machines (SVMs) is to find the hyperplane that best separates two classes of data points by maximizing the margin between the two classes. SVMs aim to find a decision boundary that can generalize well to new data points by maximizing the distance between the boundary and the closest data points.
2. In SVMs, a support vector is a data point that is closest to the hyperplane or the boundary separating the two classes. The support vectors are used to define the hyperplane, and their distances to the hyperplane are used to compute the margin.
3. It is necessary to scale the inputs when using SVMs because SVMs are sensitive to the scale of the input features. If the input features have different scales, then the SVM will give more weight to the features with larger scales, and the smaller scale features may be ignored. Therefore, it is important to scale the inputs so that each feature has a similar scale and contributes equally to the model.
4. Yes, an SVM classifier can output a confidence score that represents how confident the model is in its prediction. However, SVMs do not output a percentage chance or probability like some other classification models.
5. When training a model on a training set with millions of instances and hundreds of features, it is generally better to use the dual form of the SVM problem. The primal form is computationally expensive and can be slow to converge, whereas the dual form is more efficient and can be solved using off-the-shelf quadratic programming solvers.
6. If an SVM classifier trained using an RBF kernel appears to underfit the training data, it is generally better to increase the value of gamma to increase the model's flexibility. However, increasing C may also be helpful as it increases the penalty for misclassified points and can make the model more accurate.
7. To solve the soft margin linear SVM classifier problem with an off-the-shelf quadratic programming (QP) solver, the QP parameters (H, f, A, and b) should be set as follows:

* H is the matrix that defines the quadratic term of the optimization problem and is set to the identity matrix multiplied by a scalar value of C.
* f is the vector that defines the linear term of the optimization problem and is set to 1 for all data points in the training set.
* A is the matrix that defines the constraint that all training examples lie on the correct side of the hyperplane and is set to the transpose of the training set.
* b is the vector that defines the inequality constraints and is set to -1 for all data points in the training set.

1. Training a LinearSVC, an SVC, and an SGDClassifier on a linearly separable dataset can result in similar models. However, the specific parameters used for each model can impact the resulting decision boundary. It is important to experiment with different parameters and hyperparameters to find the best model for the specific dataset.
2. Training an SVM classifier on the MNIST dataset using one-versus-the-rest can achieve high levels of precision. The exact level of precision will depend on the specific hyperparameters used, such as the kernel type and kernel parameters, regularization parameter, and others. Using small validation sets to tune the hyperparameters can help achieve higher precision.
3. Training an SVM regressor on the California housing dataset can produce a model that can accurately predict housing prices. The exact level of accuracy will depend on the specific hyperparameters used, such as the kernel type, kernel parameters, regularization parameter, and others. It is important to experiment with different hyperparameters to find the best model for the specific dataset.