**Generalization Error**

Welcome to chapter 2! In this video, you'll understand what is the generalization error of a supervised machine learning model.

**2. Supervised Learning - Under the Hood**

In supervised learning, you make the assumption that there's a mapping f between features and labels. You can express this as y=f(x). f which is shown in red here is an unknown function that you want to determine. In reality, data generation is always accompanied with randomness or noise like the blue points shown here.

**3. Goals of Supervised Learning**

Your goal is to find a model fhat that best approximates f. When training fhat, you want to make sure that noise is discarded as much as possible. At the end, fhat should achieve a low predictive error on unseen datasets.

**4. Difficulties in Approximating $f$**

You may encounter two difficulties when approximating f. The first is overfitting, it's when fhat fits the noise in the training set. The second is underfitting, it's when fhat is not flexible enough to approximate f.

**5. Overfitting**

When a model overfits the training set, its predictive power on unseen datasets is pretty low. This is illustrated by the predictions of the decision tree regressor shown here in red. The model clearly memorized the noise present in the training set. Such model achieves a low training set error and a high test set error.

**6. Underfitting**

When a model underfits the data, like the regression tree whose predictions are shown here in red, the training set error is roughly equal to the test set error. However, both errors are relatively high. Now the trained model isn't flexible enough to capture the complex dependency between features and labels. In analogy, it's like teaching calculus to a 3-year old. The child does not have the required mental abstraction level that enables him to understand calculus.

**7. Generalization Error**

The generalization error of a model tells you how much it generalizes on unseen data. It can be decomposed into 3 terms: bias, variance and irreducible error where the irreducible error is the error contribution of noise.

**8. Bias**

The bias term tells you, on average, how much fhat and f are different. To illustrate this consider the high bias model shown here in black; this model is not flexible enough to approximate the true function f shown in red. High bias models lead to underfitting.

**9. Variance**

The variance term tells you how much fhat is inconsistent over different training sets. Consider the high variance model shown here in black; in this case, fhat follows the training data points so closely that it misses the true function f shown in red. High variance models lead to overfitting.

**10. Model Complexity**

The complexity of a model sets its flexibility to approximate the true function f. For example: increasing the maximum-tree-depth increases the complexity of a decision tree.

**11. Bias-Variance Tradeoff**

The diagram here shows how the best model complexity corresponds to the lowest generalization error. When the model complexity increases, the variance increases while the bias decreases. Conversely, when model complexity decreases, variance decreases and bias increases. Your goal is to find the model complexity that achieves the lowest generalization error. Since this error is the sum of three terms with the irreducible error being constant, you need to find a balance between bias and variance because as one increases the other decreases. This is known as the bias-variance trade-off.

**12. Bias-Variance Tradeoff: A Visual Explanation**

Visually, you can imagine approximating fhat as aiming at the center of a shooting-target where the center is the true function f. If fhat is low bias and low variance, your shots will be closely clustered around the center. If fhat is high variance and high bias, not only will your shots miss the target but they would also be spread all around the shooting target.

**Diagnosing Bias and Variance Problems**

In this video, you'll learn how to diagnose bias and variance problems.

**Estimating the Generalization Error**

Given that you've trained a supervised machine learning model labeled fhat, how do you estimate the fhat's generalization error? This cannot be done directly because: - f is unknown, - usually you only have one dataset, - you don't have access to the error term due to noise.

**3. Estimating the Generalization Error**

A solution to this is to first split the data into a training and test set. The model fhat can then be fit to the training set and its error can be evaluated on the test set. The generalization error of fhat is roughly approximated by fhat's error on the test set.

**4. Better Model Evaluation with Cross-Validation**

Usually, the test set should be kept untouched until one is confident about fhat's performance. It should only be used to evaluate fhat's final performance or error. Now, evaluating fhat's performance on the training set may produce an optimistic estimation of the error because fhat was already exposed to the training set when it was fit. To obtain a reliable estimate of fhat's performance, you should use a technique called cross-validation or CV. CV can be performed using K-Fold-CV or hold-out-CV . In this lesson, we'll only be explaining K-fold-CV.

**5. K-Fold CV**

The diagram here illustrates this technique for K=10: - First, the training set (T) is split randomly into 10 partitions or folds, - The error of fhat is evaluated 10 times on the 10 folds, - Each time, one fold is picked for evaluation after training fhat on the other 9 folds. - At the end, you'll obtain a list of 10 errors.

**6. K-Fold CV**

Finally, as shown in this formula, the CV-error is computed as the mean of the 10 obtained errors.

**7. Diagnose Variance Problems**

Once you have computed fhat's cross-validation-error, you can check if it is greater than fhat's training set error. If it is greater, fhat is said to suffer from high variance. In such case, fhat has overfit the training set. To remedy this try decreasing fhat's complexity. For example, in a decision tree you can reduce the maximum-tree-depth or increase the maximum-samples-per-leaf. In addition, you may also gather more data to train fhat.

**8. Diagnose Bias Problems**

On the other hand, fhat is said to suffer from high bias if its cross-validation-error is roughly equal to the training error but much greater than the desired error. In such case fhat underfits the training set. To remedy this try increasing the model's complexity or gather more relevant features for the problem.

**9. K-Fold CV in sklearn on the Auto Dataset**

Let's now see how we can perform K-fold-cross-validation using scikit-learn on the auto-dataset which is already loaded. In addition to the usual imports, you should also import the function cross\_val\_score() from sklearn-dot-model\_selection. First, split the dataset into 70%-train and 30%-test using train\_test\_split(). Then, instantiate a DecisionTreeRegressor() dt with the parameters max\_depth set to 4 and min\_samples\_leaf to 0-dot-14.

**10. K-Fold CV in sklearn on the Auto Dataset**

Next, call cross\_val\_score() by passing dt, X\_train, y\_train; set the parameters cv to 10 for 10-fold-cross-validation and scoring to neg\_mean\_squared\_error to compute the negative-mean-squared-errors. The scoring parameter was set so because cross\_val\_score() does not allow computing the mean-squared-errors directly. Finally, set n\_jobs to -1 to exploit all available CPUs in computation. The result is a numpy-array of the 10 negative mean-squared-errors achieved on the 10-folds. You can multiply the result by minus-one to obtain an array of CV-MSE. After that, fit dt to the training set and evaluate the labels of the training and test sets.

**11. K-Fold CV in sklearn on the Auto Dataset**

The CV-mean-squared-error can be determined as the mean of MSE\_CV. Finally, you can use the function MSE to evaluate the train and test set mean-squared-errors. Given that the training set error is smaller than the CV-error, we can deduce that dt overfits the training set and that it suffers from high variance. Notice how the CV and test set errors are roughly equal.

**Ensemble Learning**

In this lesson, you will learn about a supervised learning technique known as ensemble learning.

**2. Advantages of CARTs**

00:07 - 00:42

Let's first recap what we learned from the previous chapter about CARTs. CARTs present many advantages. For example they are easy to understand and their output is easy to interpret. In addition, CARTs are easy to use and their flexibility gives them an ability to describe nonlinear dependencies between features and labels. Moreover, you don't need a lot of feature preprocessing to train a CART. In contrast to other models, you don't have to standardize or normalize features before feeding them to a CART.

**3. Limitations of CARTs**

00:42 - 01:19

CARTs also have limitations. A classification tree for example, is only able to produce orthogonal decision boundaries. CARTs are also very sensitive to small variations in the training set. Sometimes, when a single point is removed from the training set, a CART's learned parameters may changed drastically. CARTs also suffer from high variance when they are trained without constraints. In such case, they may overfit the training set. A solution that takes advantage of the flexibility of CARTs while reducing their tendency to memorize noise is ensemble learning.

**4. Ensemble Learning**

01:19 - 01:58

Ensemble learning can be summarized as follows: -As a first step, different models are trained on the same dataset. -Each model makes its own predictions. -A meta-model then aggregates the predictions of individual models and outputs a final prediction. -The final prediction is more robust and less prone to errors than each individual model. -The best results are obtained when the models are skillful but in different ways meaning that if some models make predictions that are way off, the other models should compensate these errors. In such case, the meta-model's predictions are more robust.

**5. Ensemble Learning: A Visual Explanation**

01:58 - 02:20

Let's take a look at the diagram here to visually understand how ensemble learning works for a classification problem. First, the training set is fed to different classifiers. Each classifier learns its parameters and makes predictions. Then these predictions are fed to a meta model which aggregates them and outputs a final prediction.

**6. Ensemble Learning in Practice: Voting Classifier**

02:20 - 02:44

Let's now take a look at an ensemble technique known as the voting classifier. More concretely, we'll consider a binary classification task. The ensemble here consists of N classifiers making the predictions P0,P1,to,PN with P=0-or-1. The meta model outputs the final prediction by hard voting.

**7. Hard Voting**

02:44 - 03:08

To understand hard voting, consider a voting classifier that consists of 3 trained classifiers as shown in the diagram here. While classifiers 1 and 3 predict the label of 1 for a new data-point, classifier 2 predicts the label 0. In this case, 1 has 2 votes while 0 has 1 vote. As a result, the voting classifier predicts 1.

**8. Voting Classifier in sklearn (Breast-Cancer dataset)**

03:08 - 03:36

Now that you know what a voting classifier is, let's train one on the breast cancer dataset using scikit-learn. You'll do so using all the features in the dataset to predict whether a cell is malignant or not. In addition to the usual imports, import LogisticRegression, DecisionTreeClassifier and KNeighborsClassifier. You also need to import VotingClassifier from sklearn-dot-ensemble.

**9. Voting Classifier in sklearn (Breast-Cancer dataset)**

03:36 - 03:55

Then, split the data into 70%-train and 30%-test and instantiate the different models as shown here. After that, define a list named classifiers that contains tuples corresponding the the name of the models and the models themselves.

**10. Voting Classifier in sklearn (Breast-Cancer dataset)**

03:55 - 04:15

You can now write a for loop to iterate over the list classifiers; fit each classifier to the training set, evaluate its accuracy on the test set and print the result. The output shows that the best classifier LogisticRegression achieves an accuracy of 94-dot-7%.

**11. Voting Classifier in sklearn (Breast-Cancer dataset)**

04:15 - 04:35

Finally, you can instantiate a voting classifier vc by setting the estimators parameter to classifiers. Fitting vc to the training set yields a test set accuracy of 95-dot-3%. This accuracy is higher than that achieved by any of the individual models in the ensemble.