Introduction to Machine Learning in Geosciences GEO371T/GEO398D.1

Performance Evaluation

Mrinal K. Sen
EPS
UT Austin

November 7, 2023

Content

- Bias and Variance
- Overfitting and Underfitting
- Holdout Method
- Resampling Methods
- Evaluation Metrics

https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/08_eval-intro_notes.pdf

Overfitting and Underfitting

Our overall goal is to derive a model or a hypothesis that generalizes well to new observations!

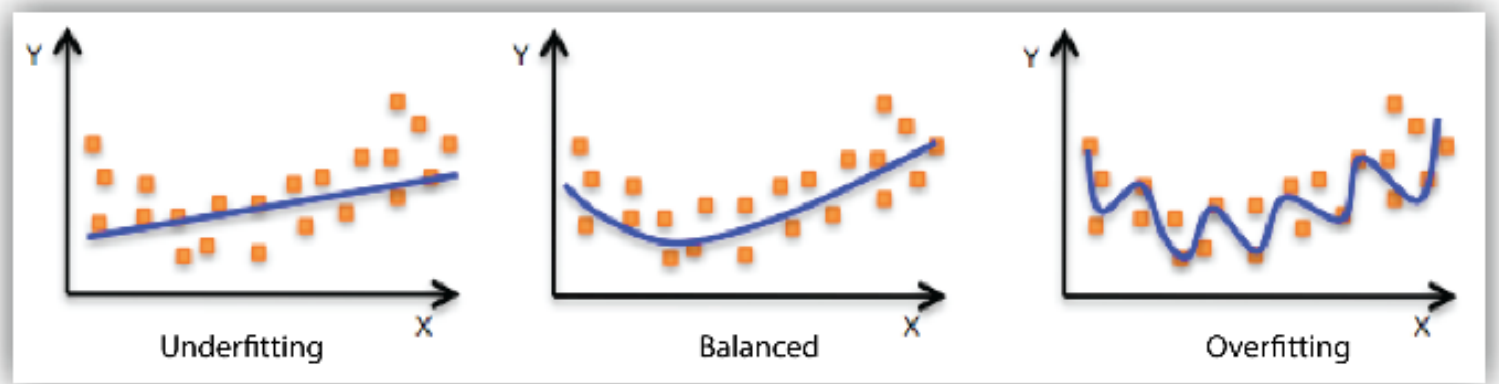
- Some of the evaluation metrics we can use to measure the performance on the test set are the **prediction accuracy** and **misclassification error** in the context of classification models – we say that a good model has a "high generalization accuracy" or "low generalization error" (or, simply "good generalization performance").
- The assumptions we generally make are the following: - i.i.d. assumption: inputs are independent, and training and test examples are identically distributed (drawn from the same probability distribution).
- For some random model that has not been fit to the training set, we expect both the training and test error to be equal.
- The training error or accuracy provides an (optimistically) biased estimate of the generalization performance.

Overfitting and Underfitting

Our overall goal is to derive a model or a hypothesis that generalizes well to new observations!

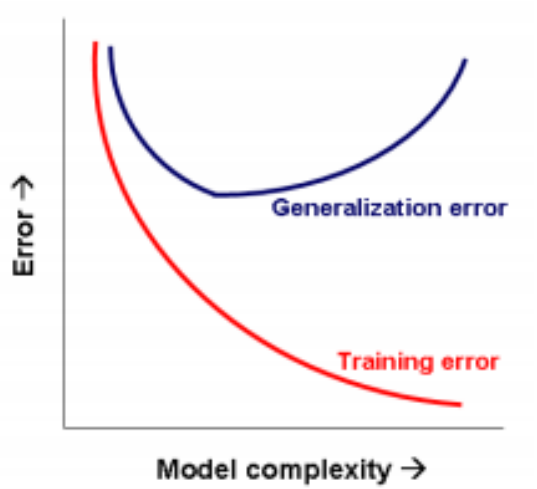
- In many learning algorithms, we have some flexibility in choosing a model complexity parameter:
  - the degree of a polynomial kernel
  - the number of hidden nodes in a neural network
  - the depth of a decision tree
  - the number of neighbors in nearest neighbor methods; and so on.
- We have seen that as the model complexity increases, the training error generally decreases, but the generalization (or test) error generally has a 'U' shape: it is high for models of low complexity, decreases until the model complexity matches the unknown data distribution, and then becomes high again for models of higher complexity:

Overfitting and Underfitting



https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html

Overfitting and Underfitting



https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html

Overfitting and Underfitting

- Overfitting and underfitting are two terms that we can use to diagnose a machine learning model based on the training and test set performance.
- A model that suffers from underfitting does perform well on the test AND training set.
- In contrast, a model that overfits (e.g., from fitting the noise in the training dataset) can be usually recognized by a high training set accuracy, but low test set accuracy.
- Intuitively, as a rule of thumb, the larger the hypothesis space a model has access to, the higher the risk of overfitting.

Bias and Variance

- Machine learning models are assessed based on how well the they predict response when provided with unseen input data, which is referred to as prediction accuracy, or alternatively, prediction error.
- Three sources contribute to prediction error:
  - Bias
  - Variance
  - Irreducible Model

- **Bias** stems from using an incorrect model.
- For example, a linear algorithm is used when a nonlinear one fits the data better for a classification problem.
- Bias is the difference between the expected value and predicted value.
- The linear model will have high bias since it is unable to learn the nonlinear boundary between the classes.
- High bias would produce consistently incorrect results.

Bias and variance

The Mathematical View

- Assume a model

$$Y = f(X) + \epsilon$$

- where X us the dependent variable and Y is the target or dependent vaiable. ϵ is the is the random error term whose mean is zero.
- Say that $\hat{Y}$ is the predicted value pf Y and $\hat{f}$ represents the estimate for f. Thus. $\hat{Y} = \hat{f}(X)$. Therefore, expected value of the squared difference between the actual and predicted value is given by

$$E(Y - \hat{Y})^2 = E[f(X) + \epsilon - \hat{f}(X)]^2$$

$$= \underbrace{(E[\hat{f}(X)] - f(X))^2}_{\text{Bias}} + \underbrace{E[(\hat{f}(X) - E\hat{f}(X)])^2}_{\text{Variance}} + \underbrace{Var(\epsilon)}_{\text{Irreducible Error}}$$

https://towardsdatascience.com/bias-variance-trade-off-a-360-degree-view-86648f69f8

Bias and Variance

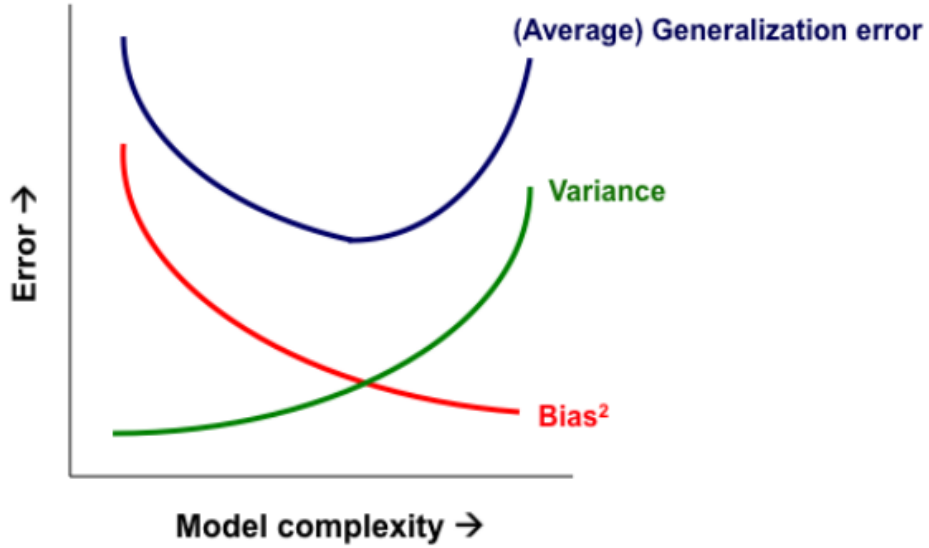


Figure 3: For a fixed sample size, as model complexity increases, the bias typically decreases variance typically increases. A high value of either contributes to a high (average) generalization bias is associated with underfitting; high variance is associated with overfitting.

http://www.shivani-agarwal.net/Teaching/CIS-520/Spring-2018/Lectures/Reading/error-bounds-decompositions.pdf

Bias and Variance

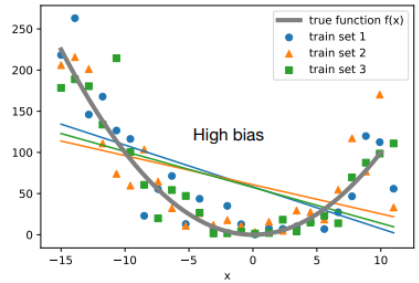


Figure 5: Suppose there is an unknown target function or "true function" to which we do want to approximate. No, suppose we have different training sets drawn from an unknown distribution defined as "true function + noise." This plot shows different linear regression models, each fit to a different training set. None of these hypotheses approximate the true function well, except at two points (around x=-10 and x=6). Here, we can say that the bias is large because the difference between the true value and the predicted value, on average (here, average means "expectation of the training sets" not "expectation over examples in the training set"), is large

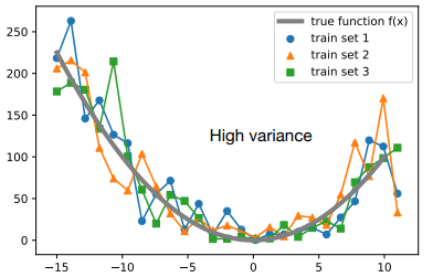


Figure 6: Suppose there is an unknown target function or "true function" to which we do want to approximate. No, suppose we have different training sets drawn from an unknown distribution defined as "true function + noise." This plot shows different unpruned decision tree models, each fit to a different training set. Note that these hypotheses fit the training data very closely. However, if we would consider the expectation over training sets, the average hypothesis would fit the true function perfectly (given that the noise is unbiased and has an expected value of 0). However, as we can see, the variance is very high, since on average, a prediction differs a lot from the expectation value of the prediction.

Bias and Variance

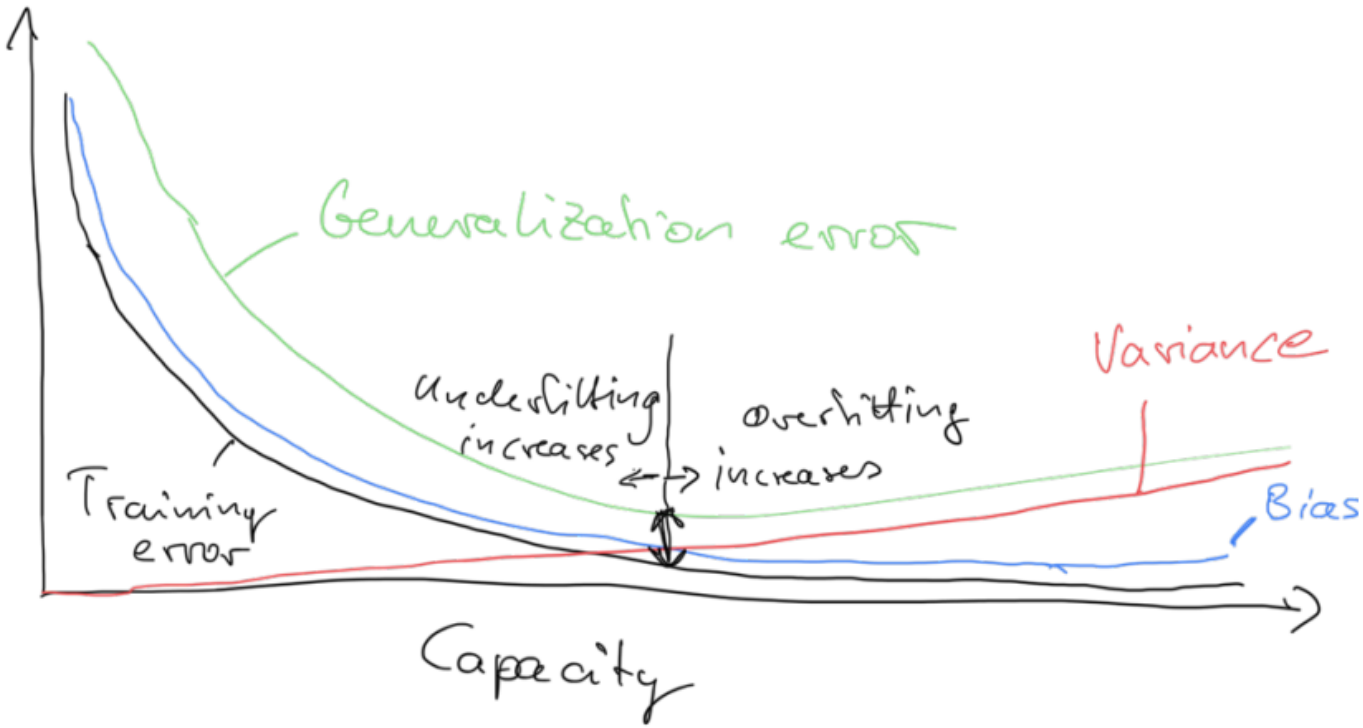


Figure 7: A sketch of variance and bias in relation to the training error and generalization error – how high variance related to overfitting, and how large bias relates to underfitting.

https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/08_eval-intro_notes.pdf

Performance Estimation

- **How do we estimate the performance of a machine learning model?**
  - we feed the training data to our learning algorithm to learn a model.
  - we predict the labels of our test set.
  - we count the number of wrong predictions on the test dataset to compute the model's prediction accuracy."
- Why do we care about performance estimates at all?
  - Ideally, the estimated performance of a model tells how well it performs on unseen data
  - making predictions on future data is often the main problem we want to solve in applications of machine learning or the development of new algorithms.
- **We are usually not only experimenting with the one single algorithm that we think would be the "best solution" under the given circumstances.**
- **More often than not, we want to compare different algorithms to each other, oftentimes in terms of predictive and computational performance.**

Performance Estimation

Resubstitution Validation and the Holdout Method

- The holdout method is inarguably the simplest model evaluation technique
  - We take a labeled dataset and split it into two parts: A training and a test set.
  - We fit a model to the training data and predict the labels of the test set.
  - The fraction of correct predictions, which can be computed by comparing the predicted labels to the ground truth labels of the test set, constitutes our estimate of the model's prediction accuracy
- Typically, the splitting of a dataset into training and test sets is a simple process of random subsampling.
- We assume that all data points have been drawn from the same probability distribution (with respect to each class). And we randomly choose 2/3 of these samples for the training set and 1/3 of the samples for the test set. Note that there are two problems with this approach, which we will discuss next.

https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/09_eval-ci_notes.pdf

Performance Estimation

Stratification

- We have to keep in mind that a dataset represents a random sample drawn from a probability distribution, and we typically assume that this sample is representative of the true population – more or less.
- Now, further subsampling without replacement alters the statistic (mean, proportion, and variance) of the sample.
- The degree to which subsampling without replacement affects the statistic of a sample is inversely proportional to the size of the sample.
- Let us take a look at an example using the Iris dataset, which we randomly divide into 2/3 training data and 1/3 test data.
- When we randomly divide a labeled dataset into training and test sets, we violate the assumption of statistical independence.
- The Iris datasets consists of 50 Setosa, 50 Versicolor, and 50 Virginica flowers; the flower species are distributed uniformly: 33.3% Setosa, 33.3% Versicolor, 33.3% Virginica.
- If a random function assigns 2/3 of the flowers (100) to the training set and 1/3 of the flowers (50) to the test set, it may yield the following :
  - training set 38 Setosa, 28 Versicolor, 34 Virginica
  - test set 12 Setosa, 22 Versicolor, 16 Virginica

`https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/09_eval-ci_notes.pdf`

Performance Estimation

Stratification

- The class ratio that the learning algorithm uses to learn the model is "38% / 28% / 34%."
- The test dataset that is used for evaluating the model is imbalanced as well, and even worse, it is balanced in the "opposite" direction: "24% / 44% / 32%."
- Unless the learning algorithm is completely insensitive to these perturbations, this is certainly not ideal.
- The problem becomes even worse if a dataset has a high class imbalance upfront, prior to the random subsampling. In the worst case scenario, the test set may not contain any instance of a minority class at all.
- Thus, the recommended practice is to divide the dataset in a stratified fashion. Here, stratification simply means that we randomly split a dataset such that each class is correctly represented in the resulting subsets (the training and the test set)
- **stratification is an approach to maintain the original class proportion in resulting subsets.**
- Note that random subsampling in non-stratified fashion is usually not a big concern when working with relatively large and balanced datasets.
- However, stratified resampling is usually beneficial in machine learning applications.

`https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/09_eval-ci_notes.pdf`

Performance Estimation

Stratification

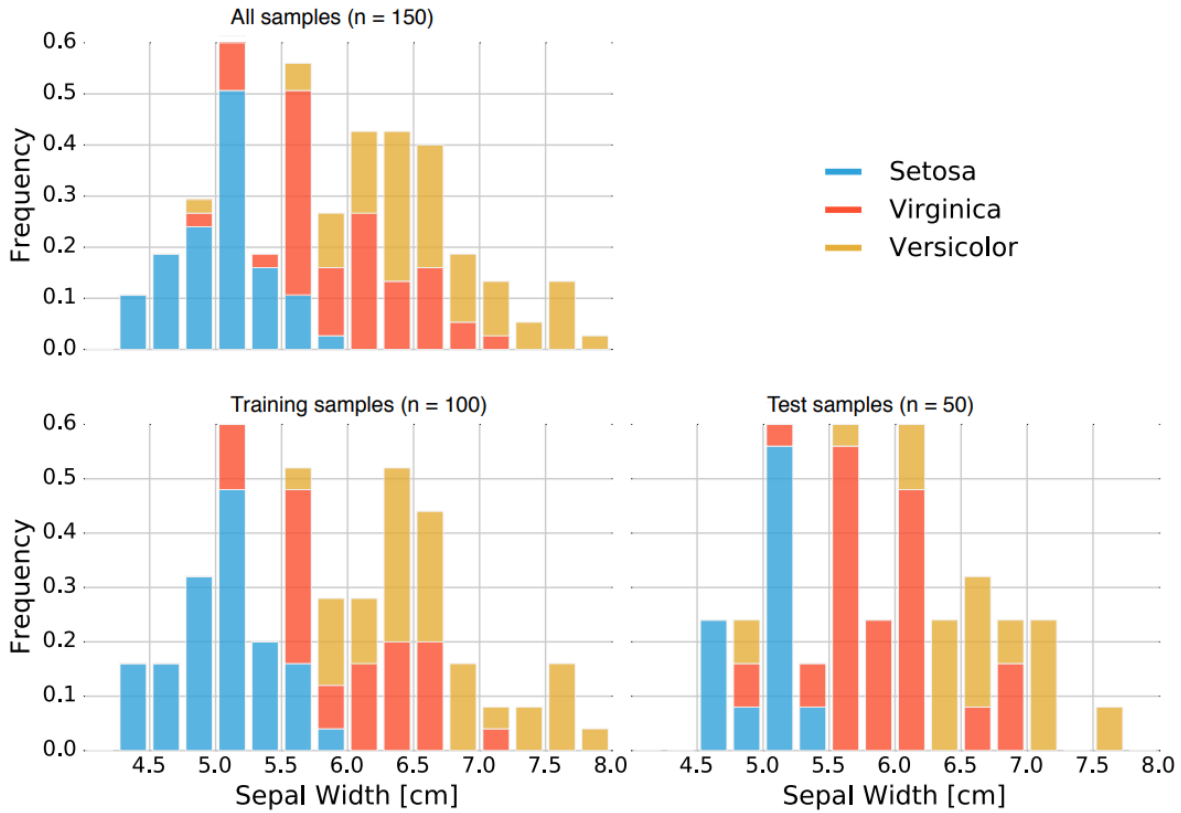


Figure 1: Distribution of *Iris* flower classes upon random subsampling into training and test sets.

`https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/09_eval-ci_notes.pdf`

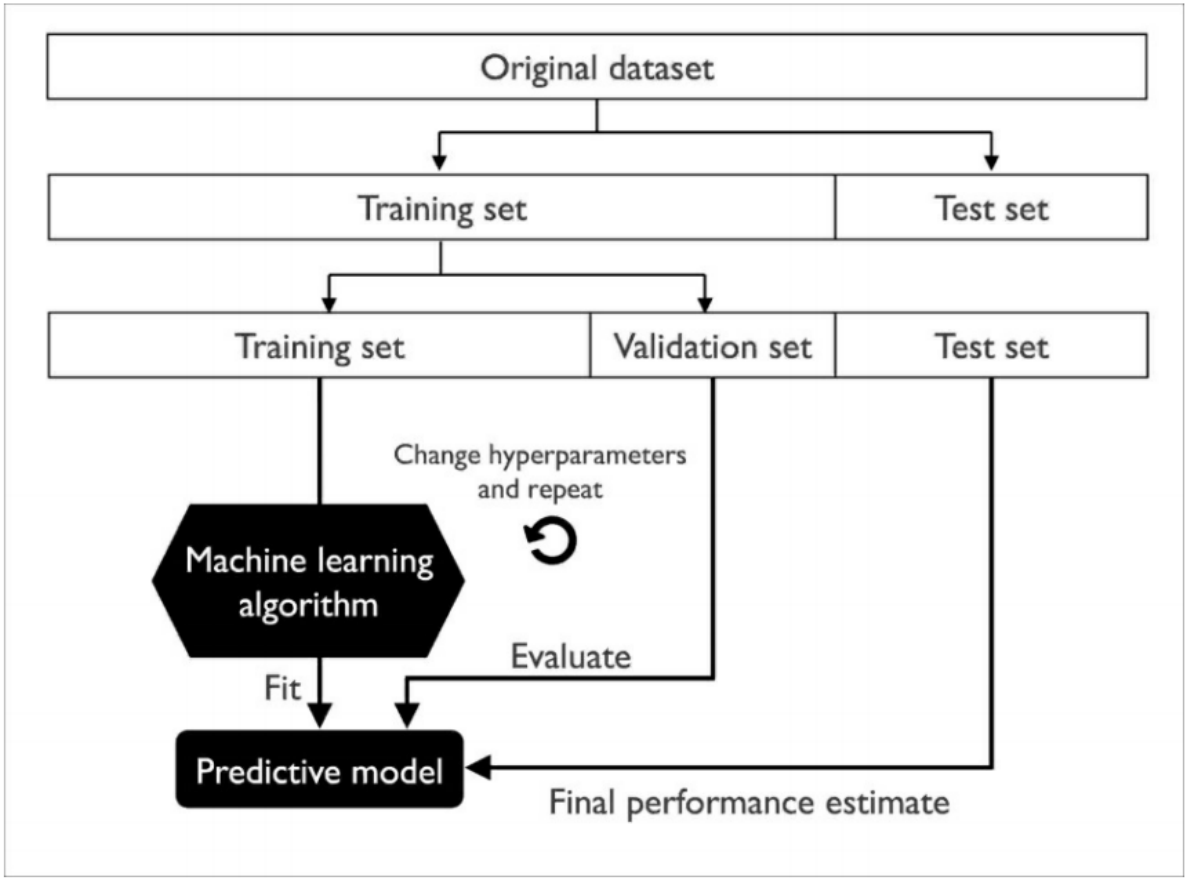
Performance Estimation

The Holdout Method

- **cross-validation. Using the holdout method**: split the initial dataset into a separate training and test dataset — the former is used for model training, and the latter is used to estimate its generalization performance.
- we are also interested in tuning and comparing different parameter settings to further improve the performance for making predictions on unseen data - select the optimal values of tuning parameters hyperparameters).
- Better Approach : separate the data into three parts: a training set, a validation set, and a test set.
- The training set is used to fit the different models, and the performance on the validation set is then used for the model selection.
- The advantage of having a test set that the model hasn't seen before during the training and model selection steps is that we can obtain a less biased estimate of its ability to generalize to new data. The following figure illustrates the concept of holdout cross-validation,

Performance Estimation

The Holdout Method



http://diggerdnepr.ddns.net/wp-content/uploads/2019/02/python-machine-learning-2nd.pdf

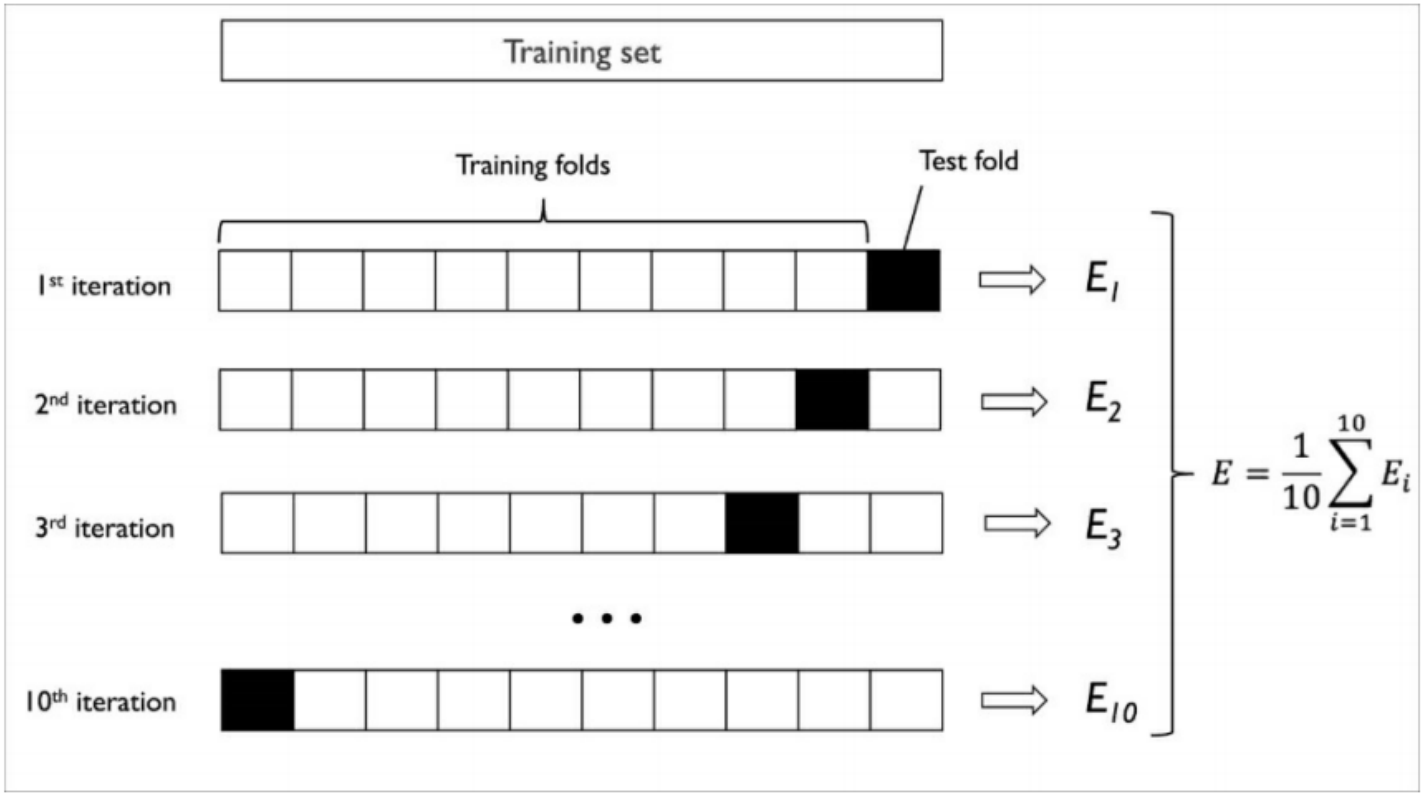
Performance Estimation

K-fold Cross-validation

- Randomly split the training dataset into k folds without replacement, where k — 1 folds are used for the model training, and one fold is used for performance evaluation. Repat k times so that we obtain k models and performance estimates.
- Calculate the average performance of the models based on the different, independent folds to obtain a performance estimate that is less sensitive to the subpartitioning of the training data compared to the holdout method. Typically, we use k-fold cross-validation for model tuning, that is, finding the optimal hyperparameter values that yields a satisfying generalization performance.
- Once we have found satisfactory hyperparameter values, we can retrain the model on the complete training set and obtain a final performance estimate using the independent test set.
- The rationale behind fitting a model to the whole training dataset after k-fold cross-validation is that providing more training samples to a learning algorithm usually results in a more accurate and robust

Performance Estimation

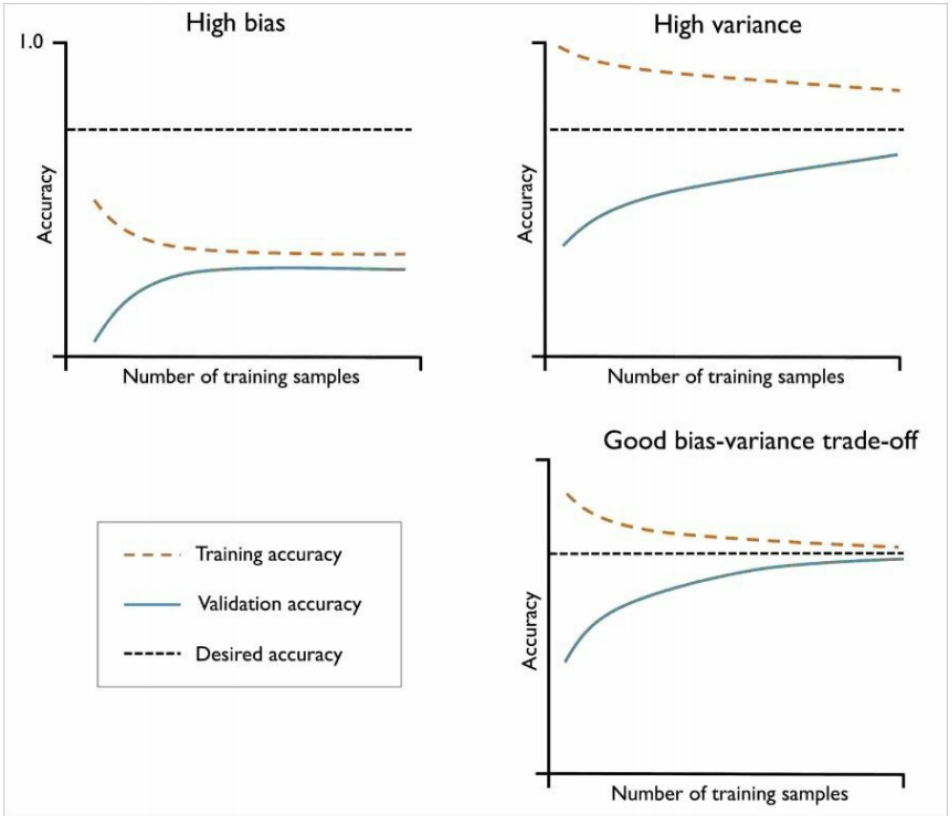
K-fold Cross-validation



http://diggerdnepr.ddns.net/wp-content/uploads/2019/02/python-machine-learning-2nd.pdf

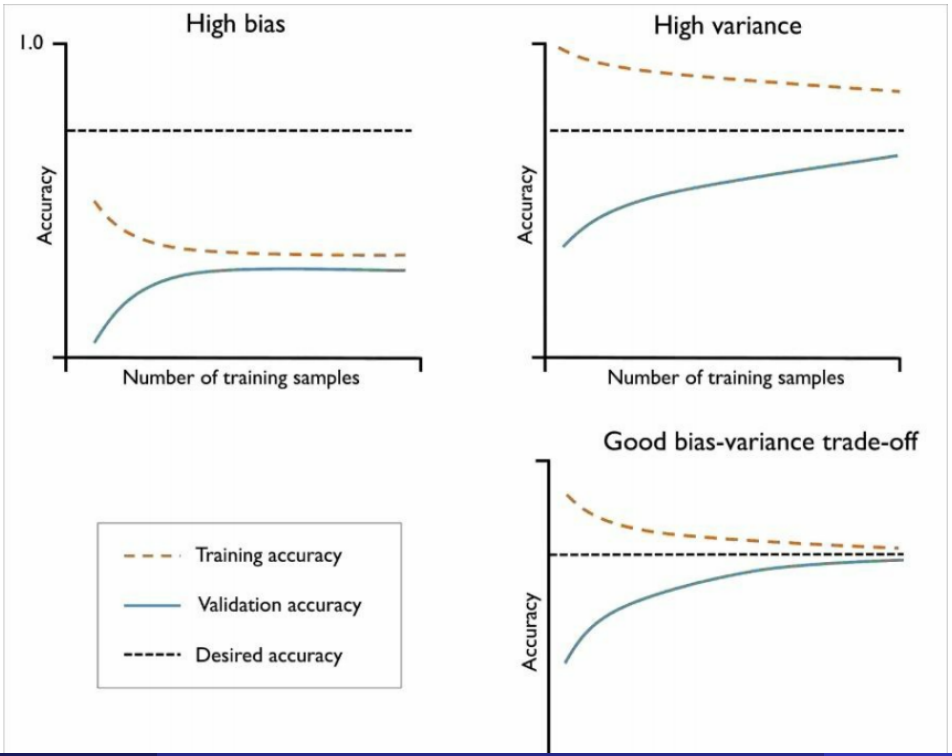
Diagnosing bias and variance problems with learning curves

- By plotting the model **training and validation accuracies** as functions of the training set size, we can easily detect whether the model suffers from high variance or high bias, and whether the collection of more data could help address this problem.



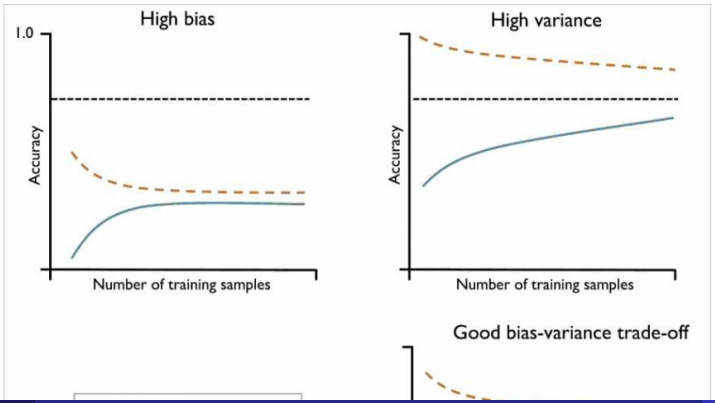
Diagnosing bias and variance problems with learning curves

- upper-left - model with high bias.
  - low training and cross-validation accuracy, which indicates that it underfits the training data.
  - Common ways to address this issue are to increase the number of parameters of the model, for example, by collecting or constructing additional features, or by decreasing the degree of regularization, for example, in SVM or logistic regression classifiers.



Diagnosing bias and variance problems with learning curves

- upper-right - suffers from high variance, which is indicated by the large gap between the training and cross-validation accuracy.
- we can collect more training data, reduce the complexity of the model, or increase the regularization parameter, for example. For unregularized models, it can also help decrease the number of features via feature selection or feature extraction to decrease the degree of overfitting.
- While collecting more training data usually tends to decrease the chance of overfitting, it may not always help, for example, if the training data is extremely noisy or the model is already very close to optimal.

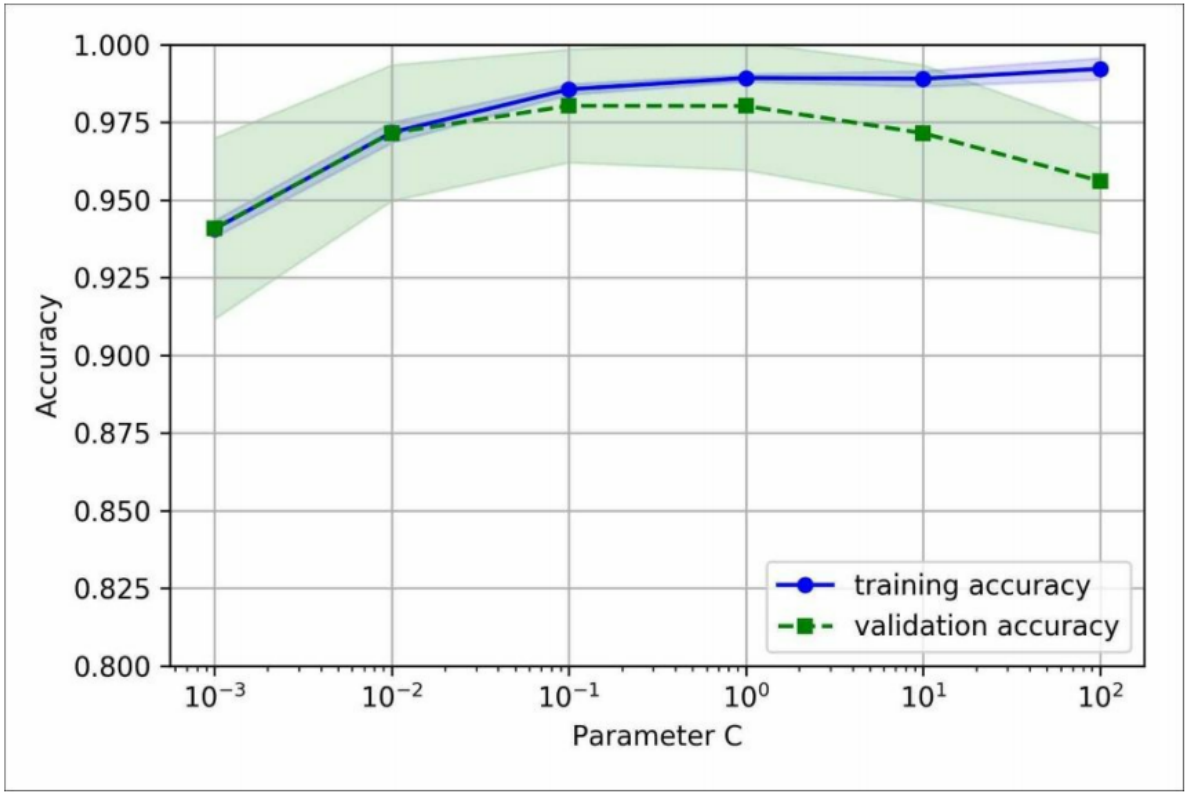


Addressing over- and underfitting with validation curves

- Validation curves are a useful tool for improving the performance of a model by addressing issues such as overfitting or underfitting.
- Validation curves are related to learning curves, but instead of plotting the training and test accuracies as functions of the sample size, we vary the values of the model parameters, for example, the inverse regularization parameter C in logistic regression.
- Although the differences in the accuracy for varying values of C are subtle, we can see that the model slightly underfits the data when we increase the regularization strength (small values of C). However, for large values of C, it means lowering the strength of regularization, so the model tends to slightly overfit the data. In this case, the sweet spot appears to be between 0.01 and 0.1 of the C value!

Addressing over- and underfitting with validation curves

Using the preceding code, we obtained the validation curve plot for the parameter c:



http://diggerdnepr.ddns.net/wp-content/uploads/2019/02/python-machine-learning-2nd.pdf

Confusion Matrix

- A summary of prediction results on a classification problem.
- The number of correct and incorrect predictions are summarized with count values and broken down by each class.
- This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions.
- It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.

	Class 1 Predicted	*Class 2 Predicted*
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

https://www.geeksforgeeks.org/confusion-matrix-machine-learning/: :text=A

Classification Rate/Accuracy

- Classification Rate or Accuracy is given by the relation:

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}}.$$

- However, there are problems with accuracy. It assumes equal costs for both kinds of errors. A 99 percent accuracy can be excellent, good, mediocre, poor or terrible depending upon the problem.

https://www.geeksforgeeks.org/confusion-matrix-machine-learning/: :text=A

Precision

- There are many cases in which classification accuracy is not a good indicator of your model performance.
- One of these scenarios is when your class distribution is imbalanced (one class is more frequent than others).
- In this case, even if you predict all samples as the most frequent class you would get a high accuracy rate, which does not make sense at all (because your model is not learning anything, and is just predicting everything as the top class).
- Therefore we need to look at class specific performance metrics too. Precision is one of such metrics, which is defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}.$$

https://www.geeksforgeeks.org/confusion-matrix-machine-learning/: :text=A

Recall

- TRecall is another important metric, which is defined as the fraction of samples from a class which are correctly predicted by the model. More formally:

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FN}}.$$

https://www.geeksforgeeks.org/confusion-matrix-machine-learning/: :text=A

F1 Score

- Depending on application, you may want to give higher priority to recall or precision.
- But there are many applications in which both recall and precision are important.
- Therefore, it is natural to think of a way to combine these two into a single metric. One popular metric which combines precision and recall is called F1-score, which is the harmonic mean of precision and recall defined as:

$$\text{PF1-score} = 2\text{*Precision*Recall}/(\text{Precision}+\text{Recall}).$$

- The generalized version of F-score is defined as below. As we can see F1-score is special case of F_β when $\beta = 1$.

$$\text{P}_\beta = (1+\beta^2)\frac{\text{precision.recall}}{\beta^2.\text{precision}+\text{recall}}.$$

- Note that there is always a trade-off between precision and recall of a

Sensitivity and Specificity

- Sensitivity and specificity are two other popular metrics mostly used in medical and biology related fields, and are defined as:
- Sensitivity= Recall= TP/(TP+FN)
- Specificity= True Negative Rate= TN/(TN+FP).

https://www.geeksforgeeks.org/confusion-matrix-machine-learning/: :text=A

Evaluation Metrics

Regression Related Metrics

- **Mean squared error** is perhaps the most popular metric used for regression problems. It essentially finds the average squared error between the predicted and actual values.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2.$$

- Sometimes people use RMSE to have a metric with scale as the target values, which is essentially the square root of MSE.
- **Mean absolute error (or mean absolute deviation)** is another metric which finds the average absolute distance between the predicted and target values. MAE is define as below:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|.$$

- MAE is known to be more robust to the outliers than MSE.
- There is also another metric for evaluating regression models, called **inlier ratio**, which is essentially the percentage of data points which are predicted with an error less than a margin.