Applied Natural Language Processing (BUAN 6342)

Lecture Model Generalization

Harpreet Singh

University of Texas at Dallas

Spring 2024

Model Generalization

What is Machine Learning (ML)?

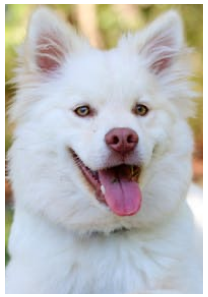
Arthur Samuel (1959). Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed

- The study of computer programs (algorithms) that can learn by example
- ML algorithms can generalize from existing examples of a task

Supervised Learning

Data

Dog



Dog

Not dog

Train a model using labels to learn rules to **predict labels** on unseen data.

Algorithm f(x) →  **Label?**

Training data **Validation data**

Test data

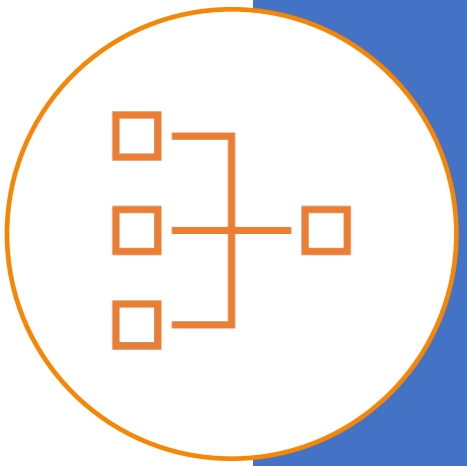
Model Generalization

- Model is trained on a training set
- $\hat{y}$ is the estimate of actual output y
- Error estimate J_{in} is obtained as sum of squared error or some other measure.
  - In-sample error

$$J_{in} = \frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y} - y^{(i)}\right)^2$$

Here m is number of observations in training set

- How well $\hat{y}$ estimates the actual output y on the data model is trained on?



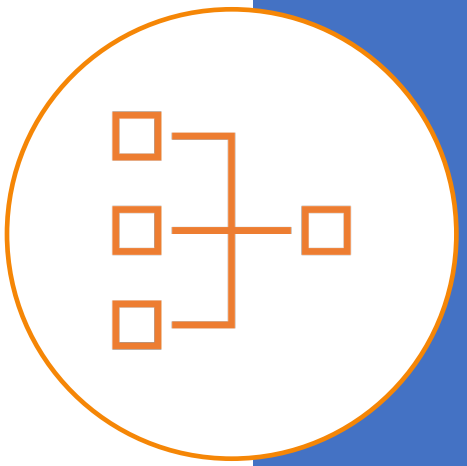
Credit: C. Bishop. Pattern Recognition and Mach. Learning.

Test Set

Model is tested on test set

- How well $\hat{y}$ estimates the actual output y for this previously unseen data?
- How does error measure J_{test} differ from the error J_{in} obtained using training set?
- Is the model able to generalize to data outside the training set?

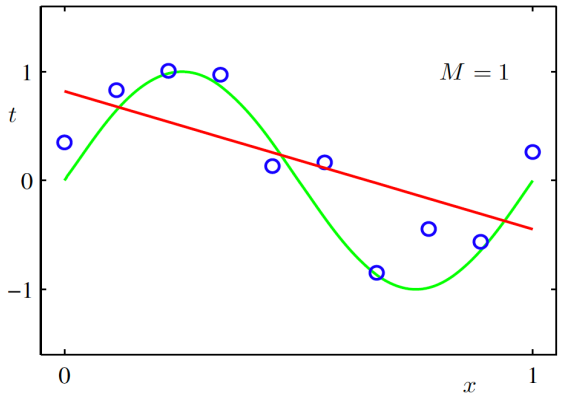
$$\mathrm{J_{test}} = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} \left(\hat{y} - y^{(i)}\right)^2$$



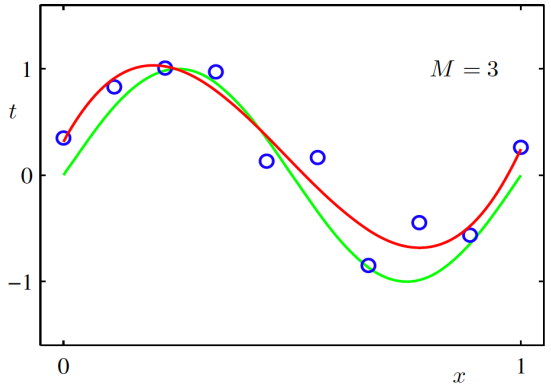
Credit: C. Bishop. Pattern Recognition and Mach. Learning.

Underfitting

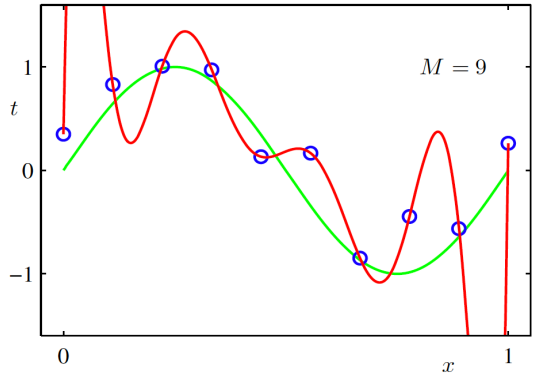
f is linear

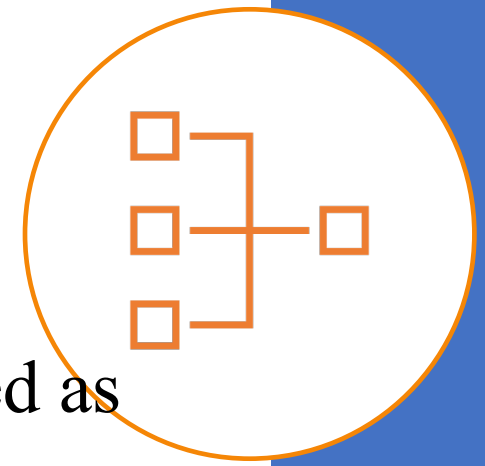


f is cubic



f is a polynomial of degree 9



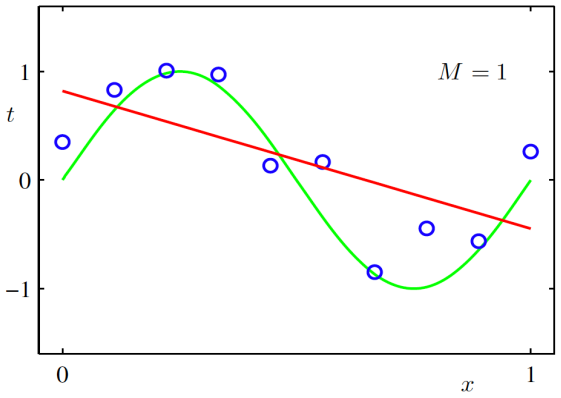


- The linear model suffers from “underfitting” (also called as high bias)
- The model is limited in terms of its free parameters and cannot adapt to a complex target function “f”
- Both in-sample (train) and test errors will be high

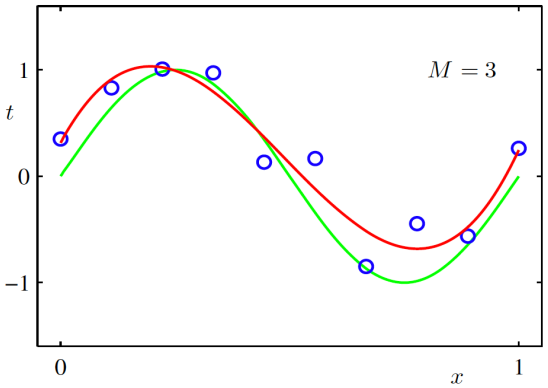
Credit: C. Bishop. Pattern Recognition and Mach. Learning.

Overfitting

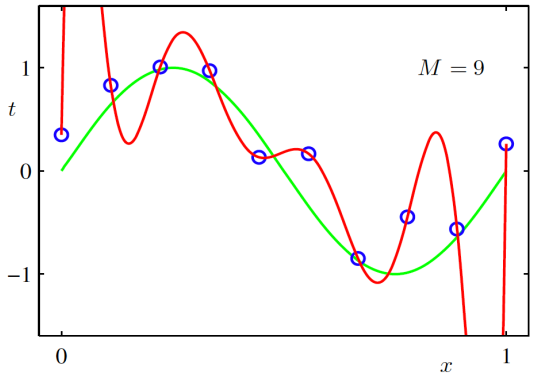
f is linear

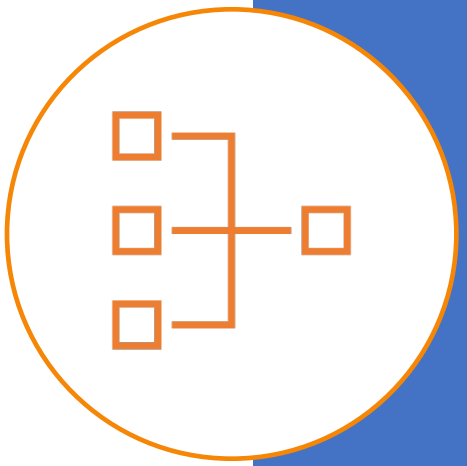


f is cubic



f is a polynomial of degree 9





- The ninth -order model (model-3) is performing what is known as "overfitting" (also known as high variance)
- The model does not do well as it has more free parameters and fits the data more than is warranted. Essentially, the model is fitting to the noise in the training set
- In-sample (train) error will be low, but test error will be high

Credit: C. Bishop. Pattern Recognition and Mach. Learning.

Generalization

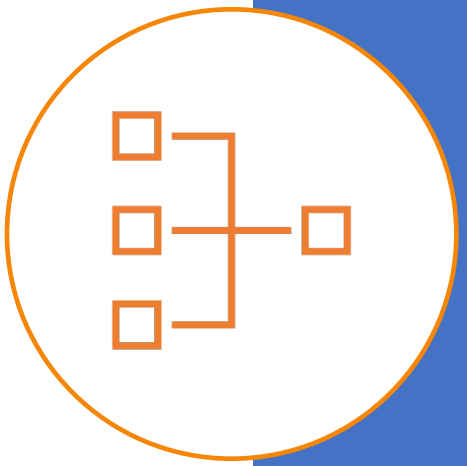
- Generalization ability refers to an algorithm's ability to give accurate predictions for new, previously unseen data.

- Assumptions:

  - Future unseen data (test set) will have the same properties as the current training sets.
  - Thus, models that are accurate on the training set are expected to be accurate on the test set.
  - But that may not happen if the trained model is tuned too specifically to the training set.

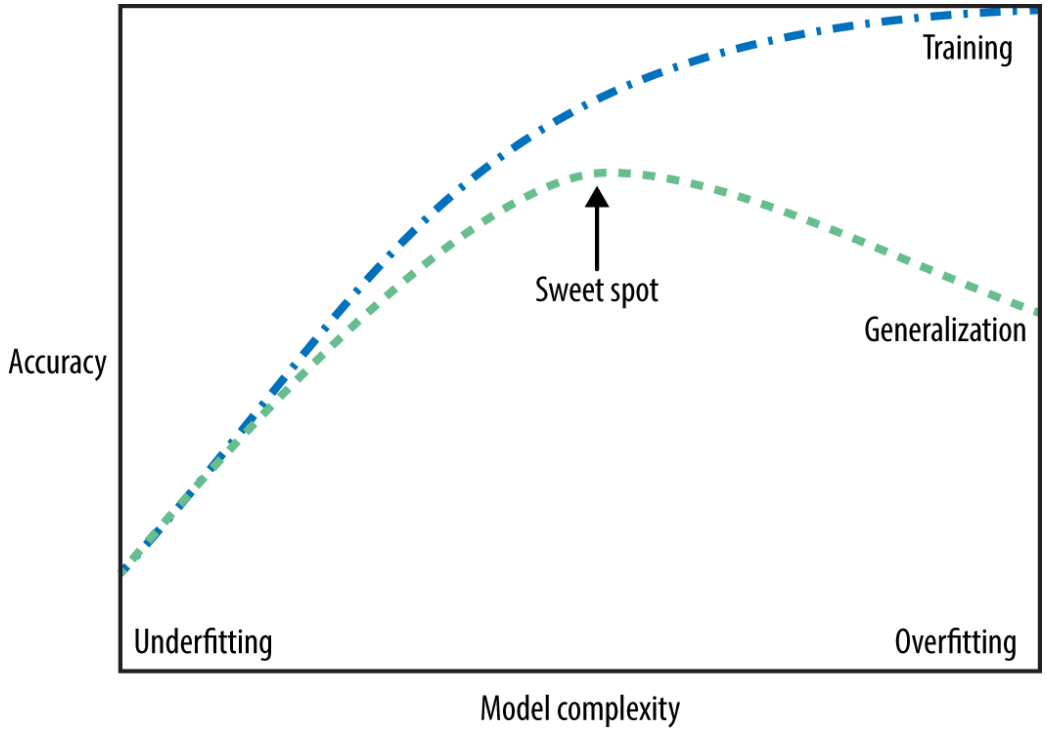
- Models that are too complex for the amount of training data available are said to overfit and are not likely to generalize well to new examples.

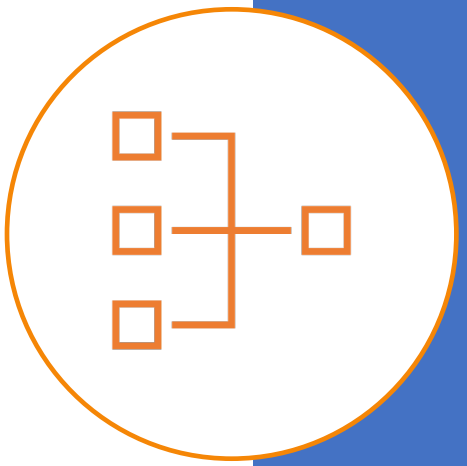
- Models that are too simple, that don't even do well on the training data, are said to underfit and also not likely to generalize well.



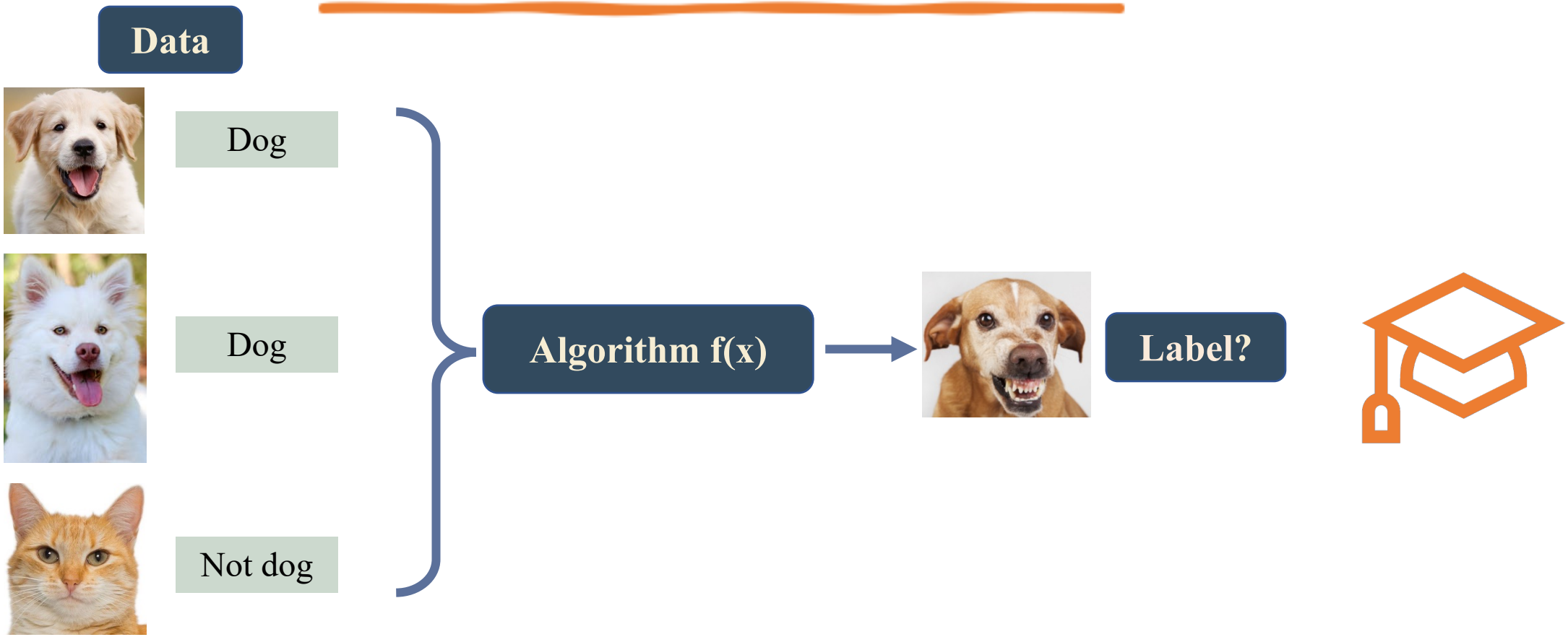
Generalization

- Overfitting
  - Low test score, high train score
  - Complex models
- Under fitting
  - Low test and train score





Need for Validation Set



Training data

Validation data

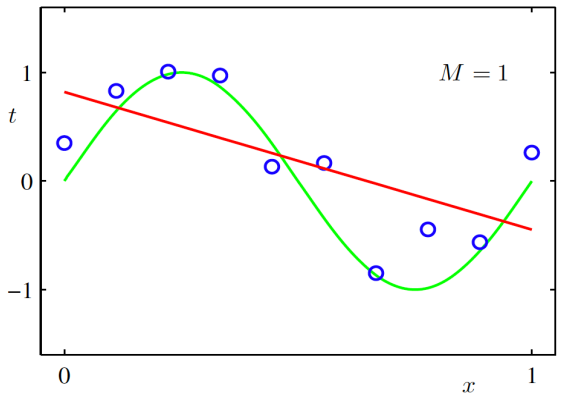
Test data

Need for Validation Set

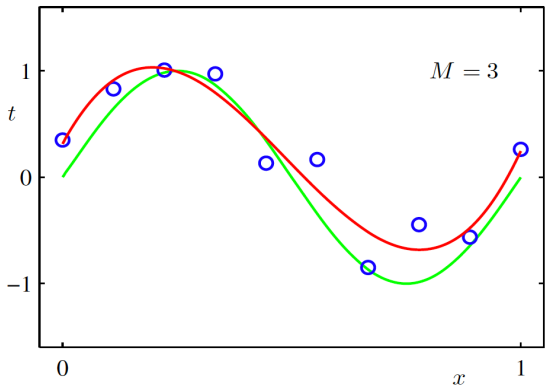
Model: $\hat{y} = \theta_n x^n + \cdots + \theta_1 x + \theta_0$

How do we select n ?

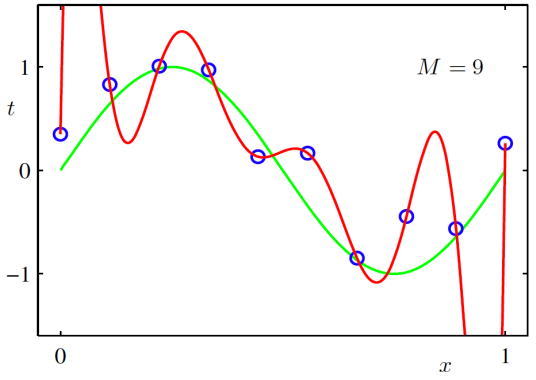
f is linear

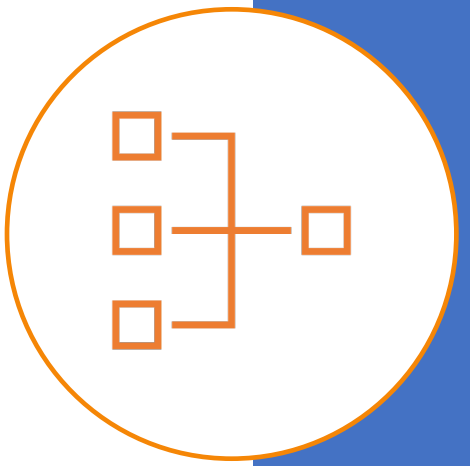


f is cubic



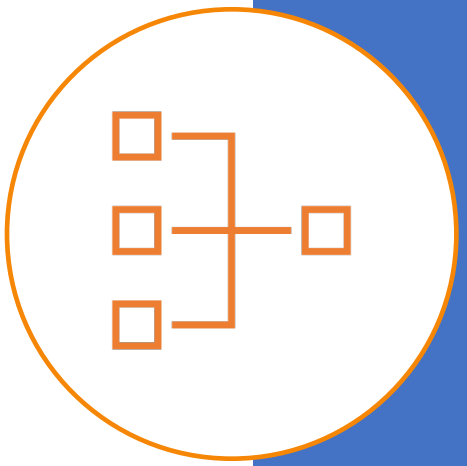
f is a polynomial of degree 9





Need for Validation Set

- We cannot use test set – test error now is optimistically biased as the learning algorithm has been tuned for the best performance using test set.
- Need a separate set if we need to do any form of model selection –in ML literature, we refer to this set as a "validation" set (or cross-validation set)
- We try to minimize J_{val} for the purposes of model selection



Data Division- Summary

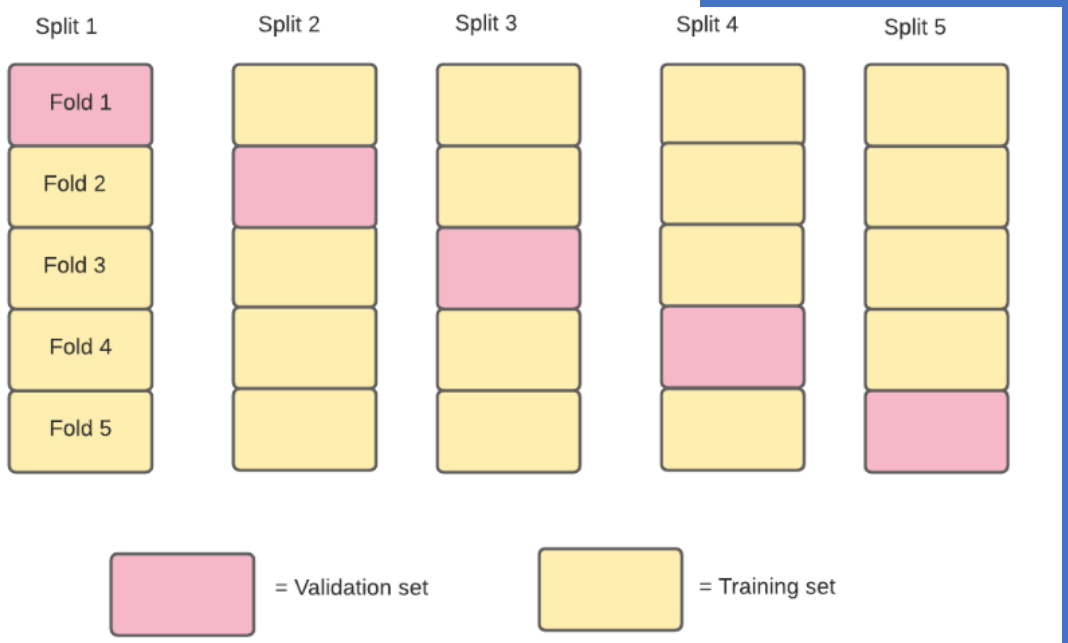
Training dataset: the sample of data used to fit the model

Validation Dataset: the sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters.

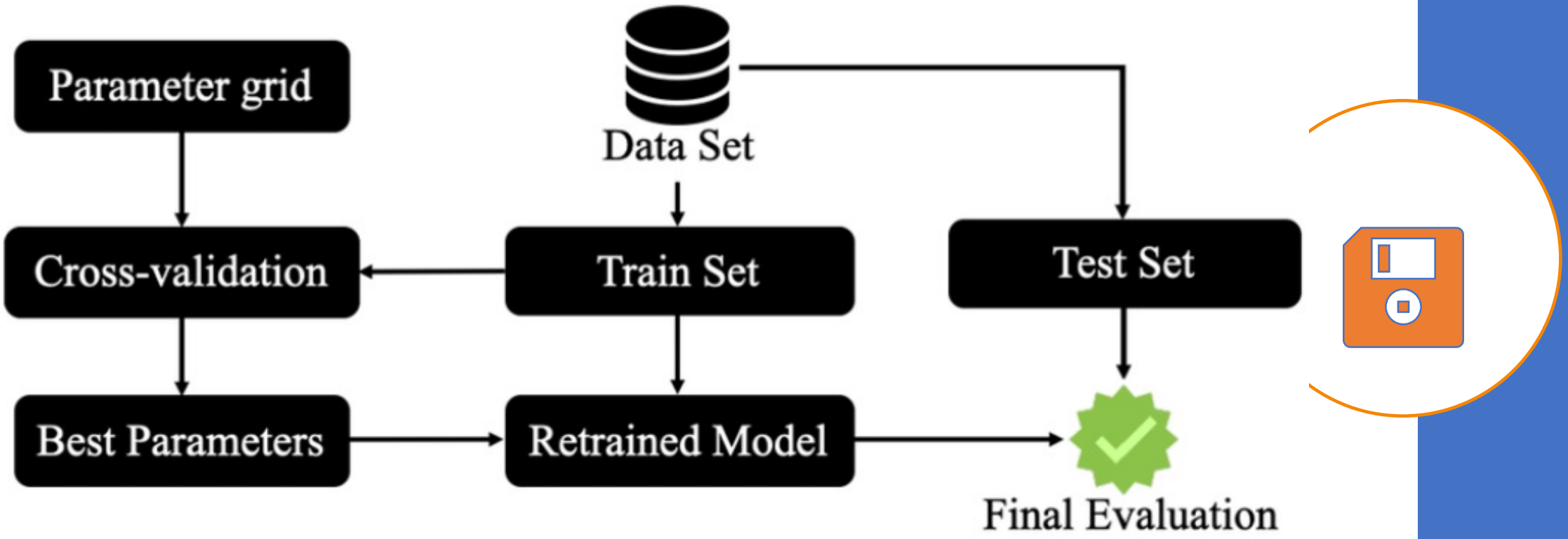
Test Dataset: a set of examples used only to assess the performance (i.e. generalization) of a fully specified classifier

Cross Validation

- In the split 1, We start by using fold1 as the validation data, and the remaining ones as the training data. You build your model on the training data, and evaluate it on the test fold. In each split, a different fold is used as validation data.
- For each of the splits of the data, we get a model evaluation and a score. In the end, we can aggregate the scores, for example by taking the mean.
- This usually takes a longer time but gives better results.,



GridSearchCV



Data Leakage

In statistics and machine learning, leakage (also known as data leakage or target leakage) is the use of information in the model training process which would not be expected to be available at prediction time, causing the predictive scores (metrics) to overestimate the model's utility when run in a production environment

Source: [Wikipedia](https://en.wikipedia.org/wiki/Leakage_(machine_learning)

Data Leakage

```
# Creating 2000 normally distributed data with 2 class labels and 5000 features.
np.random.seed(123)
X = np.random.standard_normal((2000, 5000))
y = np.random.choice(2, 2000)
```

```
# select top 10 features
X_selected = SelectKBest(k=20).fit_transform(X, y)
```

```
# split the data into train/test split
X_train, X_test, y_train, y_test = train_test_split( X_selected, y, random_state=200, train_size = 0.3)
```

```
# fit KNNClassifier on train data
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
```

```
[ ]  knn.score(X_train, y_train)
```

```
0.695
```

```
[ ]  knn.score(X_test, y_test)
```

```
0.5685714285714286
```

What is wrong here?

- the accuracy of model on test data set should not be more than 50%.
- By performing, preprocessing on complete dataset, we exposed the information from test dataset to the model as well.

Solution

- Always do preprocessing after train/test split.
- Use fit_transform on Train.
- Use only transform on Test
- If we use fit_transform on Test data then we will never know how the model perfromed on unseen data

Data Leakage in Cross Validation

```
[ ]  1  # Geneate Data
     2  np.random.seed(123)
     3  X = np.random.standard_normal((2000, 5000))
     4  y = np.random.choice(2, 2000)
     5  # split the data into train/test split
     6  X_train, X_test, y_train, y_test = train_test_split( X, y, random_state=200, train_size = 0.3)
```

```
[ ]  1  preprocessor = SelectKBest(k=20)
     2  X_train_selected = preprocessor.fit_transform(X_train, y_train)
     3  X_test_selected = preprocessor.transform(X_test)
     4
     5  # Cross Validation
     6  knn = KNeighborsClassifier()
     7  kfolds = KFold(n_splits = 5, random_state=0, shuffle = True)
     8  scores = cross_val_score(knn, X_train_selected, y_train, cv=kfolds)
     9
    10  scores.mean()
```

```
0.5883333333333334
```

```
[ ]  1  knn.fit(X_train_selected, y_train)
     2  knn.score(X_test_selected, y_test)
```

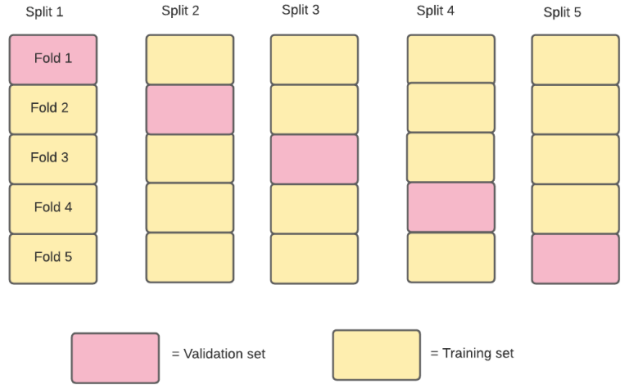
```
0.49142857142857144
```

What is wrong here?

- The cross validation score is over-optimistic. The cross validation score should also be close to 50%.
- This happened because of data leakage in cross validation step.

Create pipelines to avoid Data leakage

```
[ ]  1  np.random.seed(123)
     2  X = np.random.standard_normal((2000, 5000))
     3  y = np.random.choice(2, 2000)
```




```
[ ]  1  # split the data into train/test split
     2  X_train, X_test, y_train, y_test = train_test_split( X, y, random_state=200, train_size = 0.3)
```

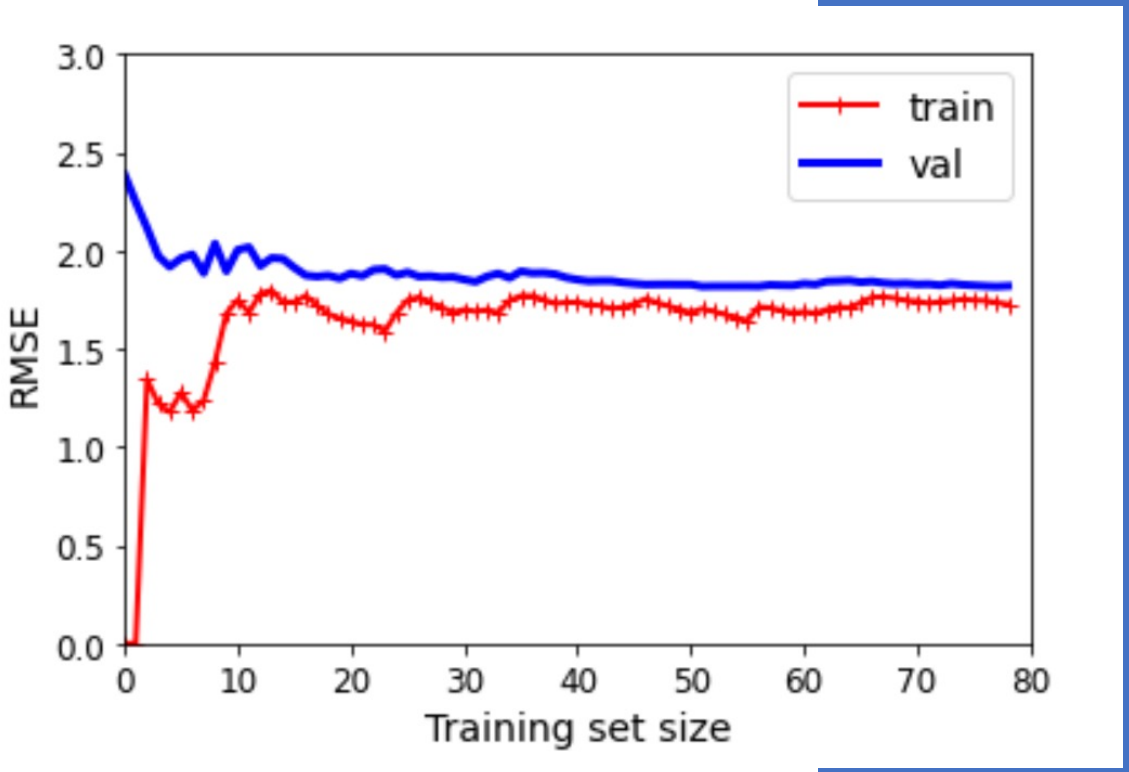
```
[ ]  1  # make a pipeline to combine SelectKBest and KNeighborsClassifier() in to one step
     2  model = make_pipeline(SelectKBest(k=20), KNeighborsClassifier())
```

```
[ ]  1  kfolds = KFold(n_splits = 5, random_state=0, shuffle = True)
     2  scores = cross_val_score(model, X_train, y_train, cv=kfolds)
     3  scores.mean()
```

```
0.49000000000000005
```

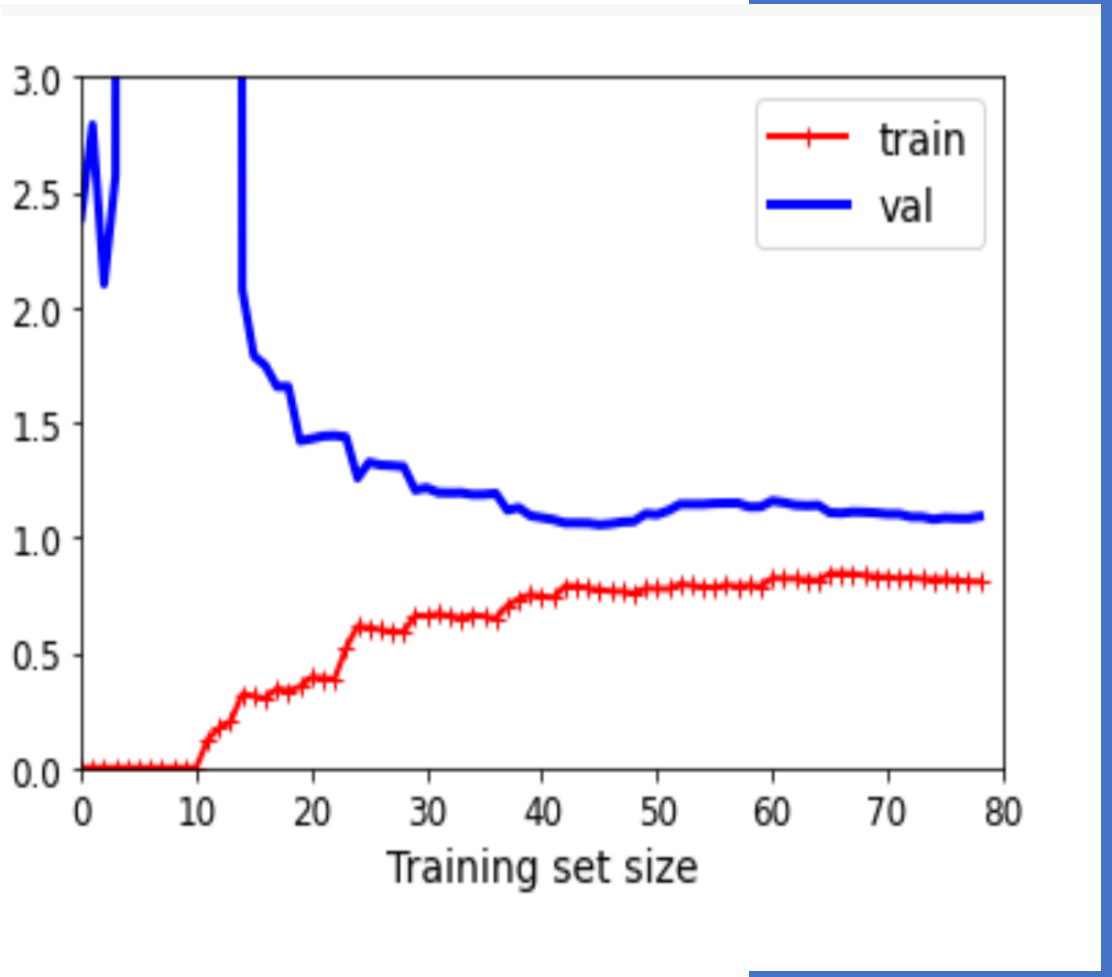
Learning Curves- Underfitting

- Training error is small at first and grows
- Training error becomes close to cross validation
- So the performance of the cross validation and training sets end up being similar (but very poor)
- **If a learning algorithm is suffering from underfitting, more data will not help**



Learning Curves-Overfitting

- When training set is small, training error is small too
- • As training set size increases, training error is still small but slowly increases
- Validation Error remains high, even when you have a moderate number of examples
- An indicative diagnostic that you overfitting is that there's a big gap between training error and cross validation error
- **If a learning algorithm is suffering from overfitting, more data is probably going to help.**



Key Takeaways

Model Generalization

- Train a model that so that it can make good predictions on unseen data.
- Overfitting:
  - Training error is small, validation error is large
  - Get more data so that model can learn underlying patterns instead of remembering the data
  - Reduce model complexity
  - Regularization – Add noise to inputs so that model becomes robust to small variations
- Underfitting:
  - Training error is large, validation error is large
  - Model does not have capacity, adding more data will not help
  - Increase model complexity
  - Try to fit model without any regularization