StNor Data

Data Model Validation

in Machine Learning

Instructor: Dr. Andrei Borodich, “StNor Data”

Webinar – LearnQuest – 22 June 2020

OUTLINE

StNor Data

Intro notes

- ❖ Types of Machine Learning
- ❖ Model Learning and Model Validation
- ○ High-bias and High-variance Models
- ○ Validation Curves
- ○ Models' Dependence on the Training Dataset Size
- ○ Learning Curves
- ✓ Validation in Practice: Grid Search & Cross Validation
- ✓ Grid Search & Cross Validation for a Best Fit Model in Python

Closure notes

OBJECTIVES

1. *Learn the relationship between Model Learning and Model Validation*
2. *Understand Validation Curves' behavior depending on the Model Complexity*
3. *Understand Learning Curves' behavior depending on the Training Dataset Size*
4. *Learn how to perform GridSearch and CrossValidation for a Best Fit Model*

What Is a Machine Learning?

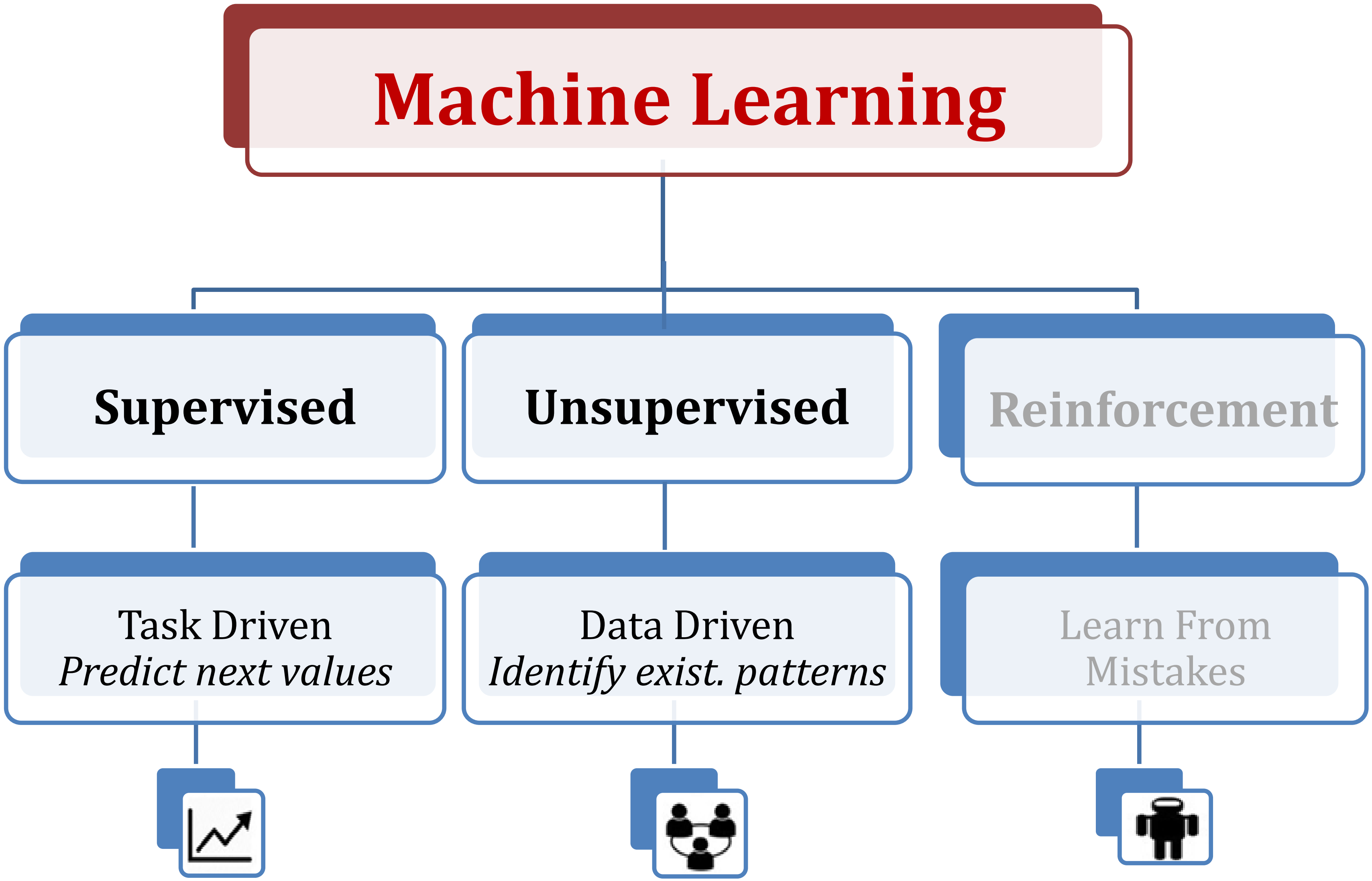
Fundamentally, **Machine Learning** involves **building mathematical models** to help understanding data.

"**Learning**" starts when we give the models **tunable parameters** that can be computationally adapted to the observed data; in this way a computer system does "**learning**" from the data.

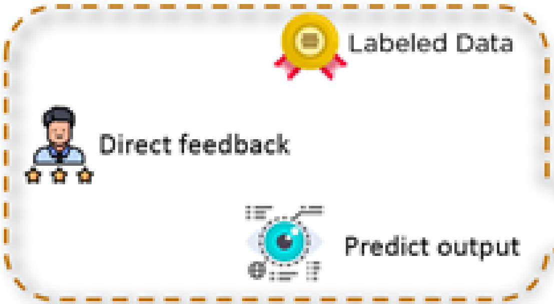
Once the models have been fit to observed data, we can use them to **predict** aspects of new data or explore structural **patterns** in the observed data.

Machine Learning is the process of teaching a computer system to

- make accurate **predictions** for the new data or
- identify hidden **patterns** in the existing data.


Machine Learning
Supervised
Unsupervised
Reinforcement
Task Driven
Predict next values
Data Driven
Identify exist. patterns
Learn From
Mistakes

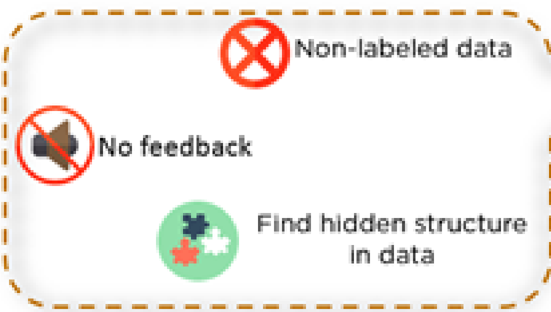
Types of Machine Learning



✓ Supervised Learning

Machine gets labeled inputs and their desired outputs.

The goal is to learn rules how to map inputs to the outputs.

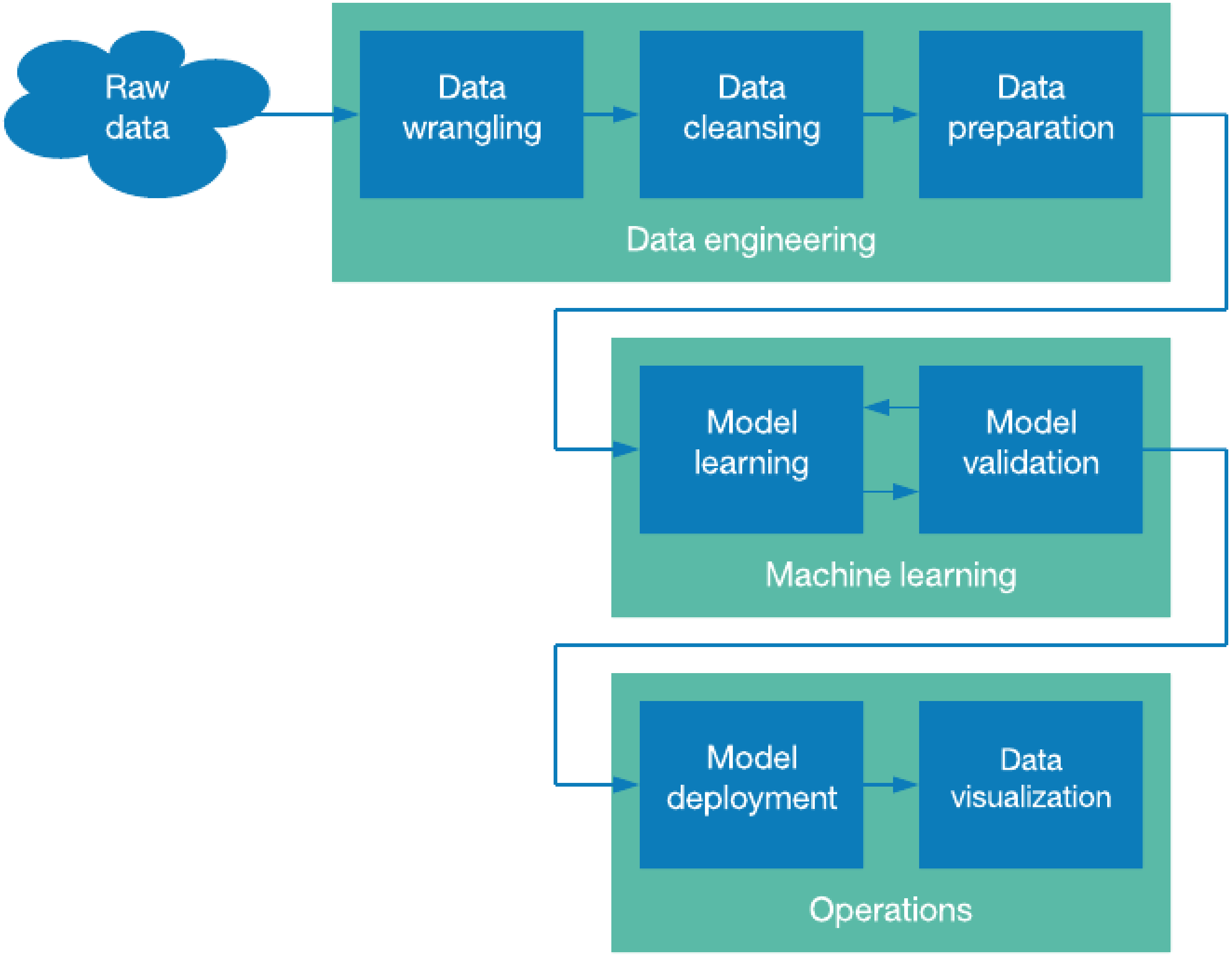


✓ Unsupervised Learning

Machine gets unlabeled inputs without desired outputs.

The goal is to find structures or patterns as useful info in inputs.

The Machine Learning Pipeline



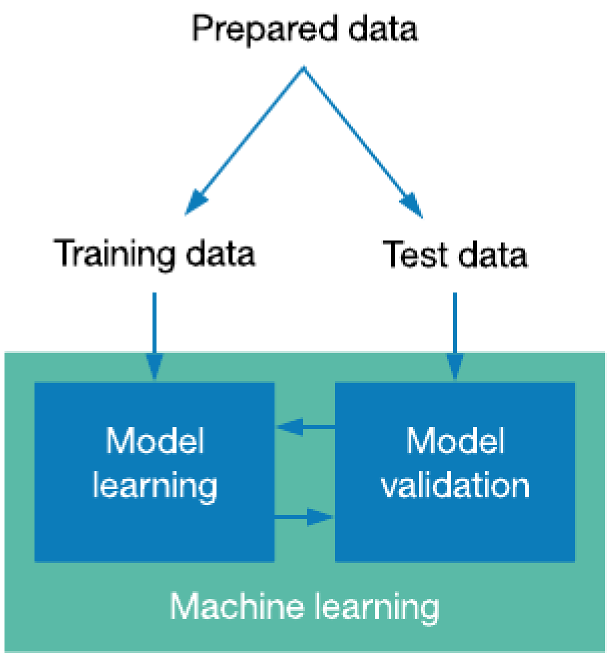
https://developer.ibm.com/

Model Learning as a Process

Model learning is referred to as the process where the model's parameters are calculating with a training data set.

Model learning begins as soon as model initialization phase has been completed.

Model learning is about a **model training** by fitting the model to a training data set.



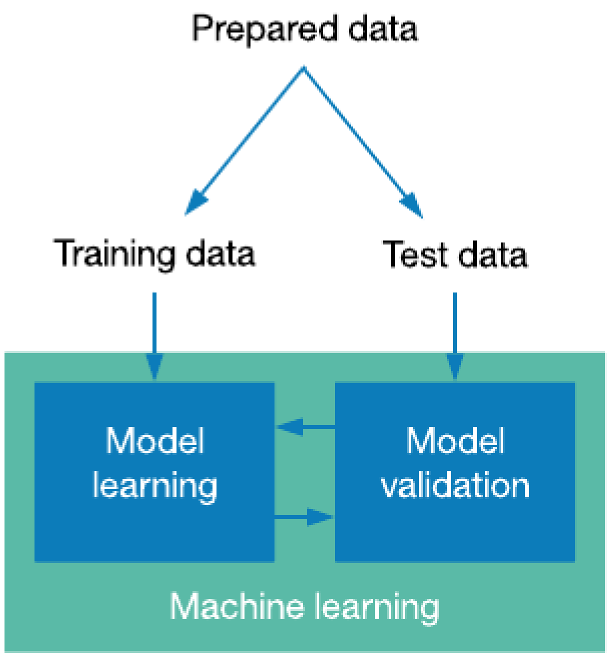
Model initialization
1. Choose a class of model.
2. Choose model hyperparameters.

Model Validation as a Process

Model validation is referred to as the process where a **trained model** is evaluated with a testing/validation data set.

Model validation begins after **model training** phase has been completed.

Model validation is about testing generalization ability of a **trained model**.



Data preparation
1. Training dataset
2. Testing data set

Model Validation as a Process

A typical ML task can be formalised as the following nested loop:

```
while (error in validation set > X) {
    tune hyper-parameters
    while (error in training set > Y) {
        tune parameters
    }
}
```

The **outer loop** is performed by human, on the validation/testing set.

The **inner loop** is done by machine, on the training set.

NOTE.
Often, in ML / AI researches, one can use 3 datasets (training, validation, and testing) instead of 2 datasets (training and testing/validating).
Then, testing data is unseen while model training & validation are going on.

Model Validation as a Process

Model parameters are those which a machine finds for the model. They are **learnt during training**.

For example, the polynomial coefficients in the Regression model.

Model hyperparameters are those which we supply to the model. They **cannot be learnt during training** but are set beforehand.

For example, the degree "n" of the polynomial in the Regression model.

Polynomial Regression model: $f(x) = \sum_{i=0}^{n} (a_i \, x^i)$

Selecting the Best Model

Consider: *two regression fits to the same dataset*

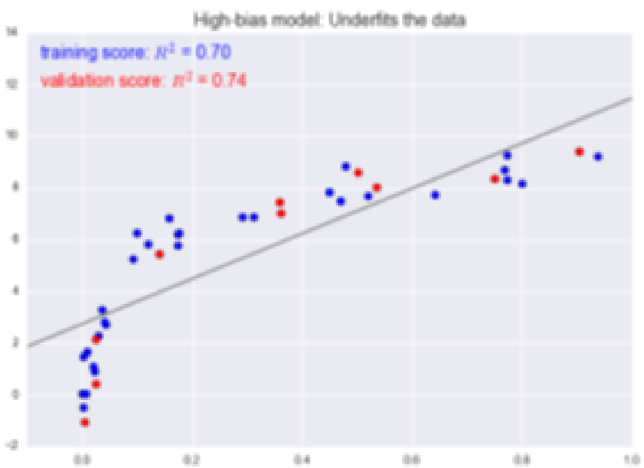
Compare: how two models represent existing data

Figure: Training and validation scores in two regression models.

R2-score (coeff. of determination) measures how well a model performs relative to a mean of the target values.

$$R^2 = 1 - \frac{SSR}{SST}$$

Selecting the Best Model



The model on the left attempts to fit **a linear function** through the data.

Because the data are intrinsically more complicated than a straight line, the straight-line model fit will never be able to describe this dataset well.

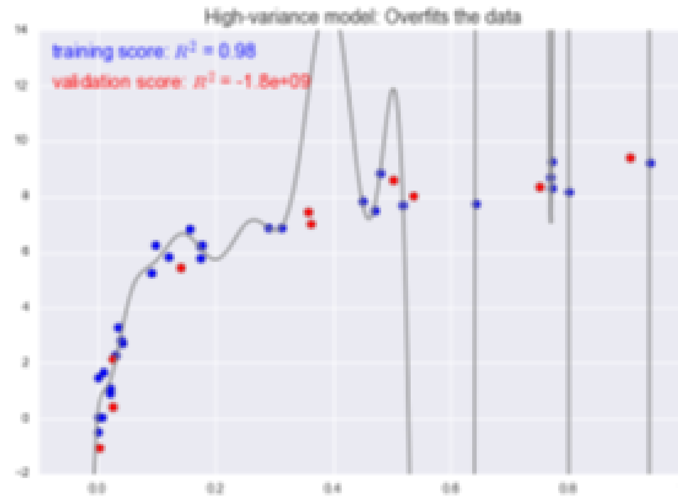
Such a model is said to **underfit** the data;

it does not have enough model flexibility to suitably account for all the features in the data.

The model has **high bias**.

Selecting the Best Model

The model on the right attempts to fit **a high-order polynomial** through the data.



The model fit has enough flexibility to well account for the fine features in the data. However, it is extremely reflective of the particular noise properties of the data.

The model is said to **overfit** the data;

it has so much model flexibility that ends up accounting for random errors together with the underlying data distribution.

The model has **high variance**.

Selecting the Best Model

We can summarize:

- For **high-bias** models, the performance of the model on the validation set is similar to the performance on the training set.
- For **high-variance** models, the performance of the model on the validation set is far worse than the performance on the training set.

We can differentiate the models using the **model complexity**:

- a **high-bias** model is a very low model complexity
- a **high-variance** model is a very high model complexity

Fundamentally, the question of "*the best model*" is about finding a sweet spot in the trade-off between **bias** and **variance**.

Selecting the Best Model

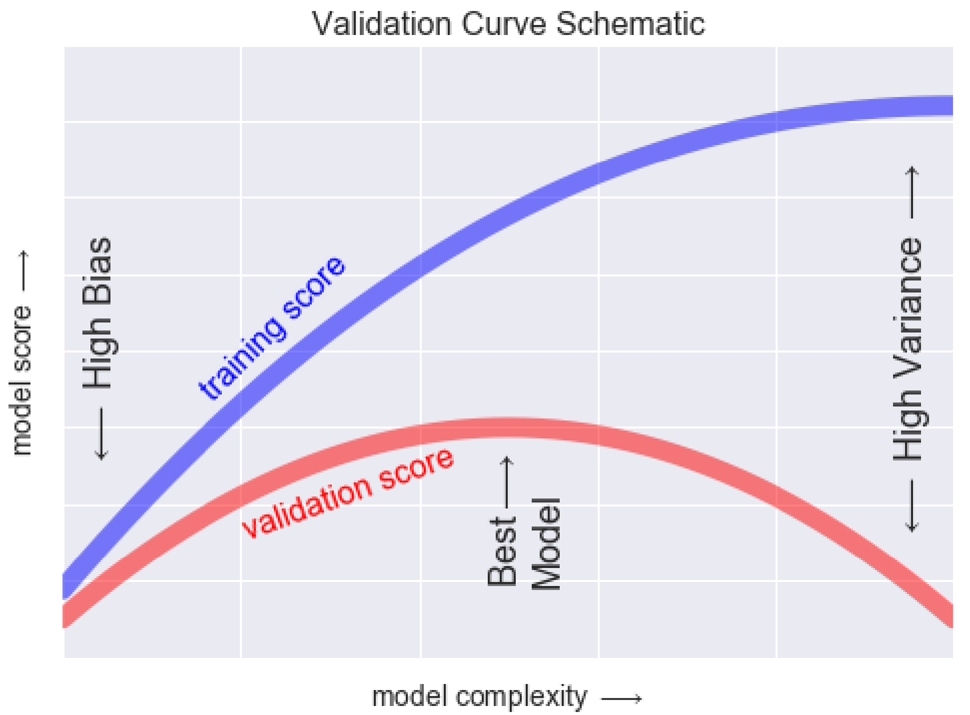
In statistics and machine learning,
the **bias–variance tradeoff** is the property of a set of predictive models:
models with a lower **bias** in parameter estimation
have a higher **variance** of the parameter estimates across samples,
and vice versa.

- The **bias** is an error from erroneous assumptions in the learning algorithm.
  - **High bias** can cause an algorithm to miss the relevant relations between features and target outputs (**underfitting**).

- The **variance** is an error from sensitivity to small fluctuations in the training set.
  - **High variance** can cause an algorithm to model the random noise in the training data, rather than the intended outputs (**overfitting**).

Validation Curve

A **plot** of the training/validation score with respect to the model complexity (polynomial degree) is known as a **validation curve**.



Model complexity, training score and validation score (a schematic relationship)

Validation Curve

The general behavior of a **Validation curve**:

The training score is everywhere higher than the validation score.
The model will be a better fit to data it has seen than to data it has not seen.

- ***For very low model complexity, the training data is underfit***.

The model is a poor predictor both for the training data and for any previously unseen data.

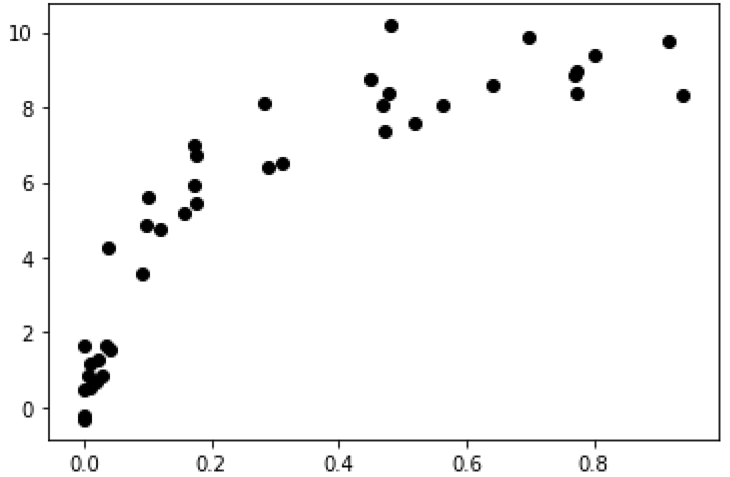
- ***For very high model complexity, the training data is overfit***.

The model predicts the training data very well, but fails for any previously unseen data.

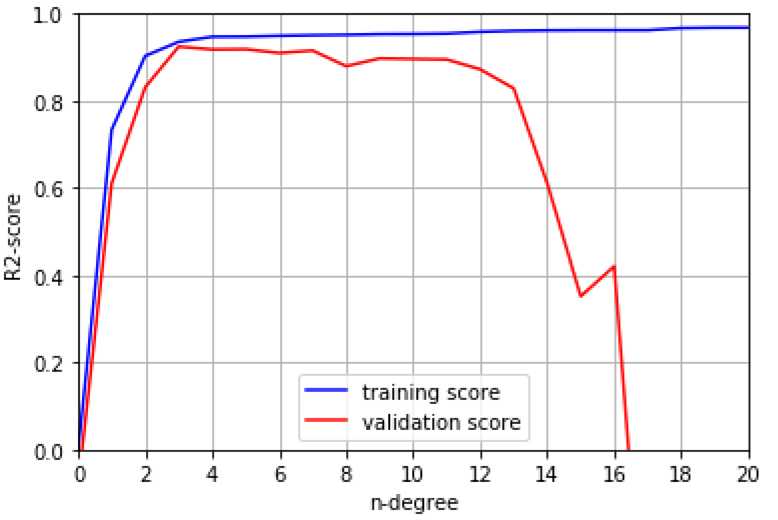
The Validation Curve has a maximum at some intermediate value.
This level of complexity indicates a suitable trade-off between bias and variance. – **This is a notable feature!**

Validation Curve

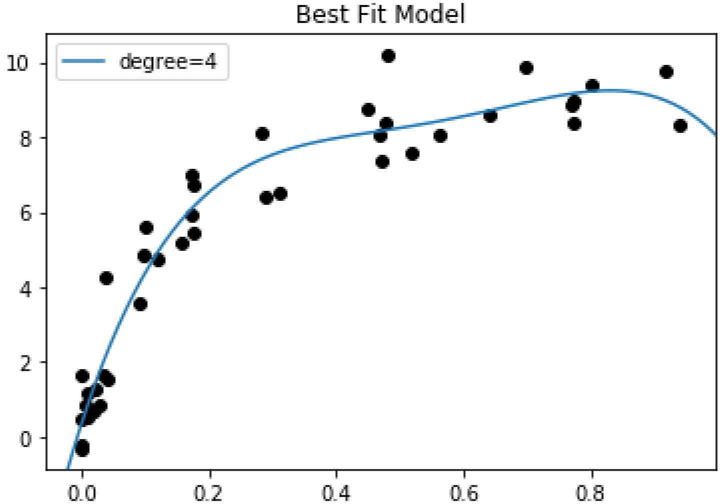
How does the optimal model generally depend on the polynomial degree?



A generated data set of 40 pairs (*X*,*y*)



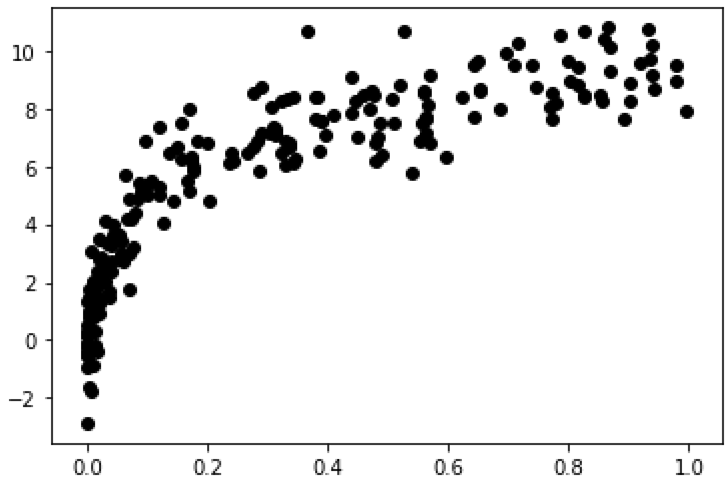
A **Validation Curve**
for the particular data (40) and
polynomial regression models



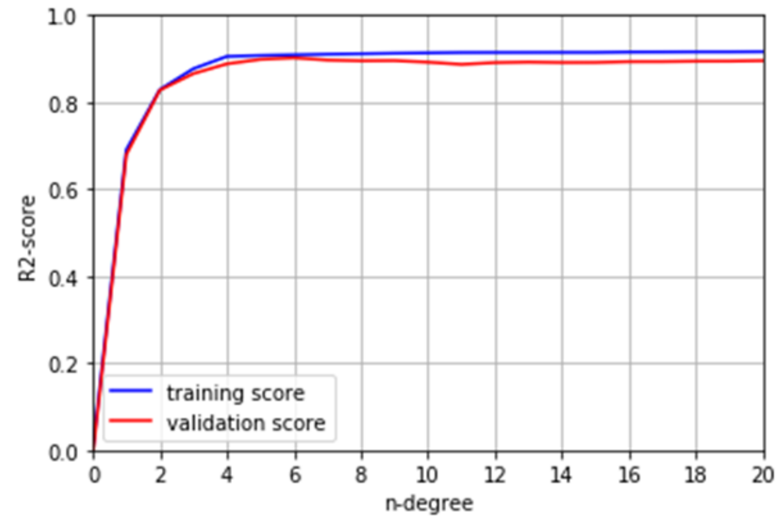
A **Best Fit Model**
(in a class of polynomial regression)
for the generated data set (40).

Validation Curve

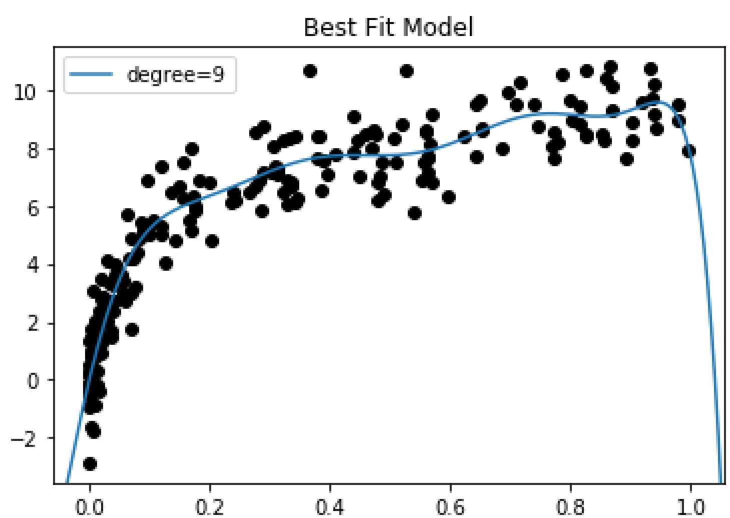
How does the optimal model generally depend on the size of the training data?



A generated data set of 200 pairs (*X2*,*y2*)

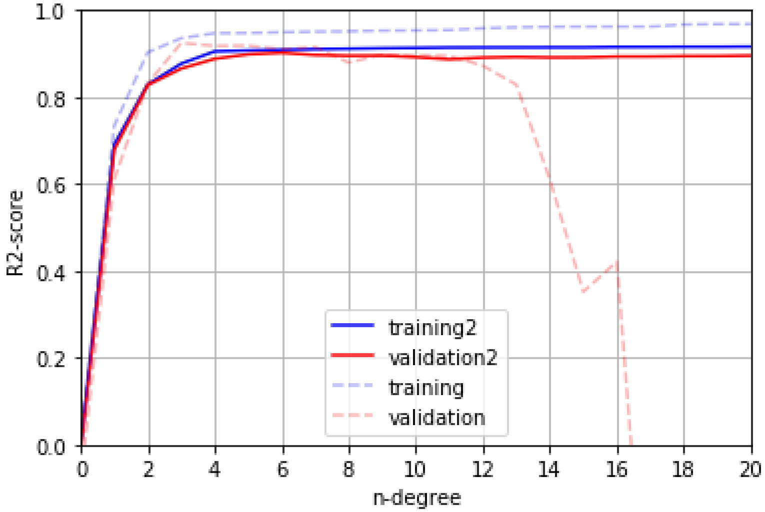


A **Validation Curve**
for the particular data (200) and
polynomial regression models



A **Best Fit Model**
(in a class of polynomial regression)
for the generated data set (200).

Validation Curve



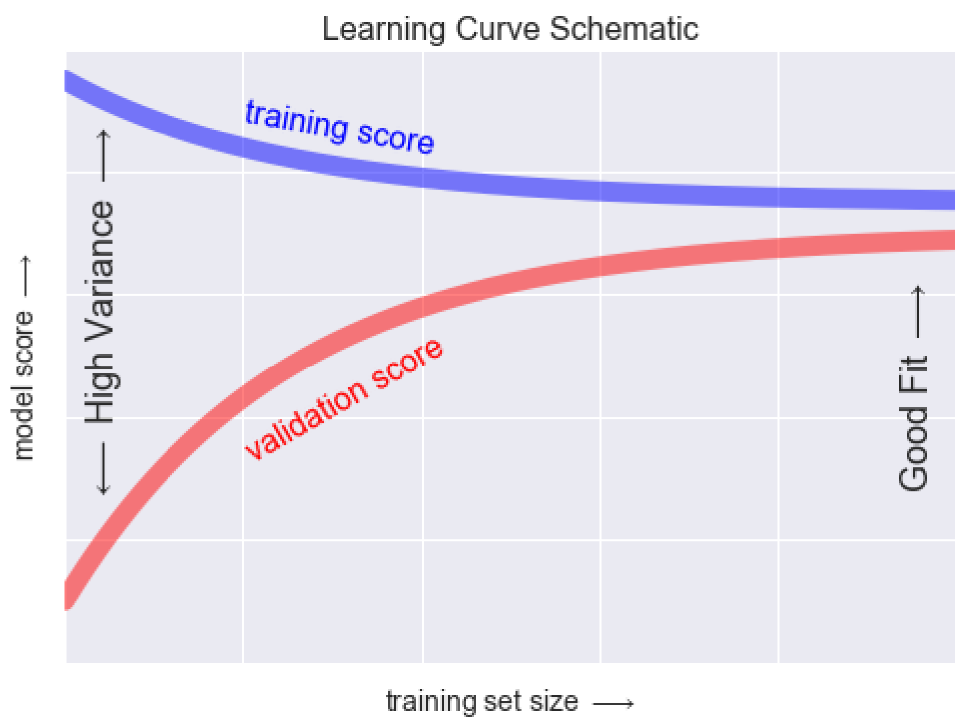
Validation Curves
for 2 particular datasets
(the 1st is a subset of the 2nd)

The larger dataset can support a much more complicated model:
the peak here is around n=4, but even an n=20 model is not seriously overfitting the data as the validation and training scores remain very close.

The behavior of the validation curve has not one, but two, important inputs: the **model complexity** and the **number of training points**.

Learning Curve

A plot of the training/validation score with respect to the size of the training set is known as a **Learning Curve**.



Dataset size, training score and validation score (a schematic relationship)

Learning Curve

The general behavior of a **Learning Curve**:

A model will never, except by chance, give a better score to the validation set than the training set: this means the curves keep getting closer together but never cross.

- A model of a given complexity will **overfit a small dataset**:

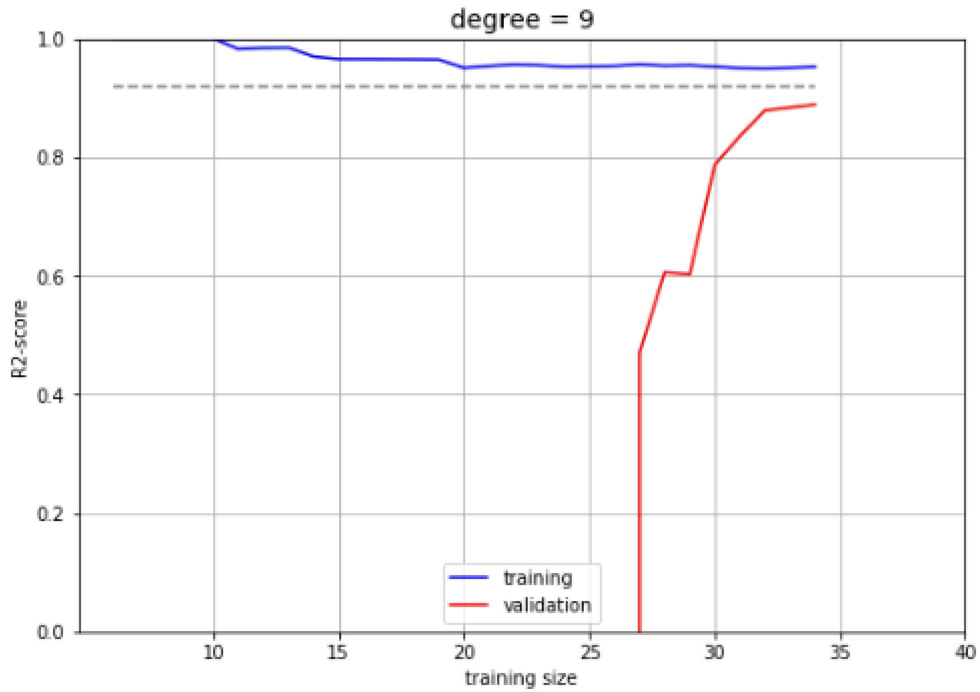
the training score will be relatively high, while the validation score will be relatively low.

- A model of a given complexity will **underfit a large dataset**:

the training score will decrease, but the validation score will increase.

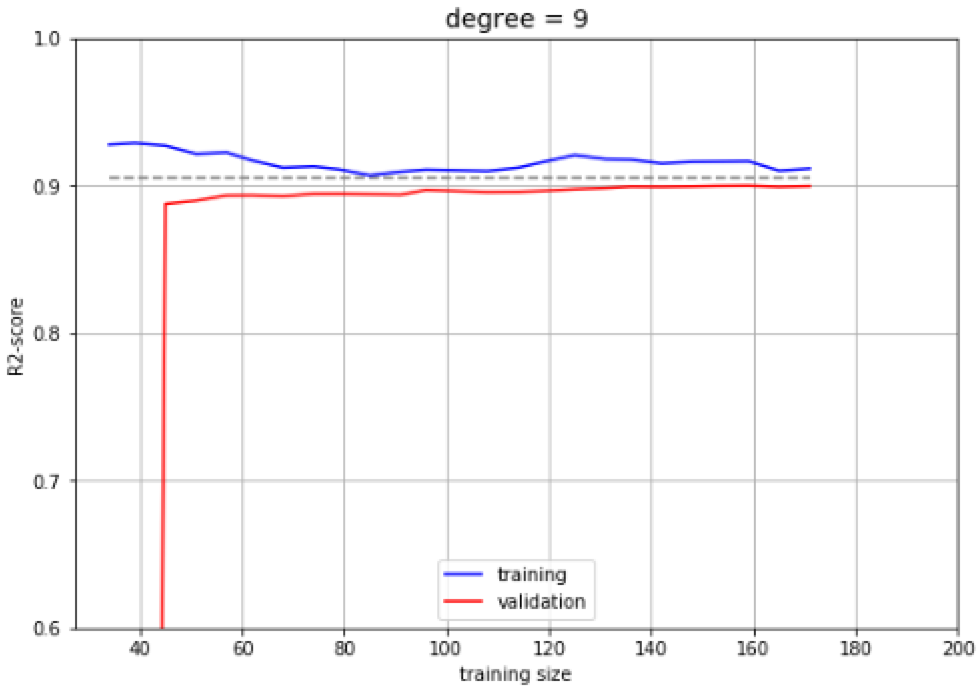
The Learning Curve is converged to a particular score
as the number of training samples grows. – **This is a notable feature!**

Learning Curve



Learning Curve of the Model for smaller training set (40)

Learning Curve



Learning Curve of the Model for larger training set (200)

Validation in Practice: Grid Search & CV

Practical methodology:

✓ Select a class of models using a **grid of parameter values** to find the particular model that maximizes the validation score.

✓ Partition the prepared data set into K-folds and use a **cross-validation** to test effectiveness of each calculated model.

5-fold CV

DATASET

Estimation 1	Test	Train	Train	Train	Train
Estimation 2	Train	Test	Train	Train	Train
Estimation 3	Train	Train	Test	Train	Train
Estimation 4	Train	Train	Train	Test	Train
Estimation 5	Train	Train	Train	Train	Test

Validation in Practice: Grid Search & CV

Scikit-Learn: https://scikit-learn.org/

-- Python library / Python package
-- provides implementations of main machine learning algorithms
-- known for a complete online documentation

GridSearchCV : https://scikit-learn.org/stable/modules/grid_search.html

-- Scikit-Learn module / Scikit-Learn class
-- generates candidates from a **grid of parameter values**
-- has automated tools to do **cross-validation**

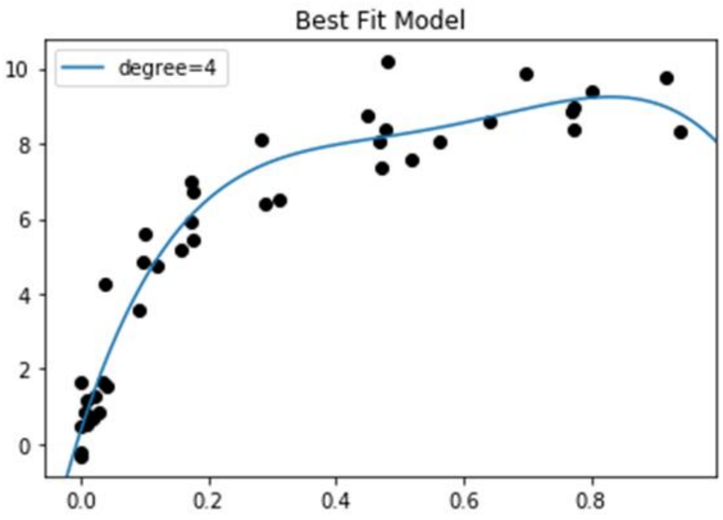
Validation in Practice: Grid Search & CV

```
from sklearn.model_selection import GridSearchCV

param_grid = {}

# GridSearchCV() estimator: it sets up the procedure; no dataset required
grid = GridSearchCV(PolynomialRegression(), param_grid, cv=7)
```

```
# fit the model at each grid point
grid.fit(X, y)

# ask for the best parameters
grid.best_params_
print(grid.best_params_)

# ask for the best model
model = grid.best_estimator_
```

{'linearregression__fit_intercept': True,
'linearregression__normalize': True,
'polynomialfeatures__degree': 4}

```
# use the best model and show the fit to our data
X_test = np.linspace(-0.1, 1.1, 40)[:, None]
y_test = model.fit(X, y).predict(X_test)
plot()
```



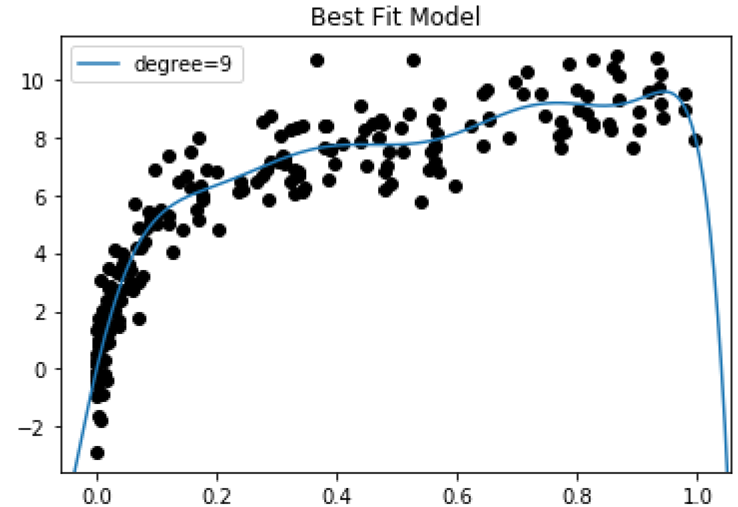
A **Best Fit Model**
(in a class of polynomial regression)
for the generated data set (40).

Validation in Practice: Grid Search & CV

```
from sklearn.model_selection import GridSearchCV

param_grid = {}

# GridSearchCV() estimator: it sets up the procedure; no dataset required
grid = GridSearchCV(PolynomialRegression(), param_grid, cv=7)
```

```
# fit the model at each grid point
grid.fit(X2, y2)

# ask for the best parameters
grid.best_params_
print(grid.best_params_)

# ask for the best model
model = grid.best_estimator_
```

{'linearregression__fit_intercept': True,
'linearregression__normalize': True,
'polynomialfeatures__degree': 9}




```
# use the best model and show the fit to our data
X_test = np.linspace(-0.1, 1.1, 200)[:, None]
y_test = model.fit(X2, y2).predict(X_test)
plot()
```

A **Best Fit Model**
(in a class of polynomial regression)
for the generated data set (200).

Conclusions

1. Together, **model training** and **model validation** aim to find an optimal data model with the best performance. — This is an ultimate goal of machine learning.

2. The technique as a **Validation Curve** can be used to select the trained model of the optimal complexity to fit well our data.

3. The technique as a **Learning Curve** can be used to quantify the correspondence of our data set size to the trained model.

4. Practically, **model validation** is performed with an advanced computational module from the ML library. This module should incorporate both **a grid** (of parameter values) **search** and **cross-validation** algorithms to find a best fit model to our data.

The END

Thank you for your attention!