L05_capacity_under_overfitting

Model capacity and under/overfitting

Qa Explain the polynomial fitting via code review

Review the code below, write a **short** code review summary, and explain how the polynomial fitting is implemented?

Do not dig into the plotting details, but explain the outcome of the plots.

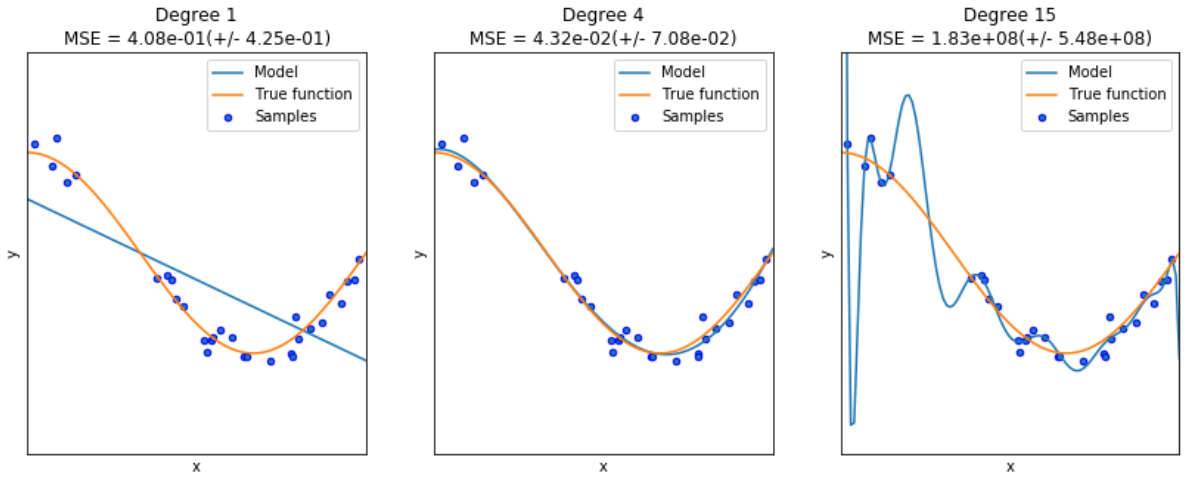
Qa implementation

In [1]:

```
# Assignment Qa:

# NOTE: code from https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html

%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score

def true_fun(X):
    return np.cos(1.5 * np.pi * X)

def GenerateData(n_samples = 30):
    X = np.sort(np.random.rand(n_samples))
    y = true_fun(X) + np.random.randn(n_samples) * 0.1
    return X, y

np.random.seed(0)

# 1) Generate data, and set up degrees array
X, y = GenerateData()
degrees = [1, 4, 15]

# 2) Loop that iterates the different degrees
print("Iterating...degrees=",degrees)
plt.figure(figsize=(14, 5))
for i in range(len(degrees)):
    ax = plt.subplot(1, len(degrees), i + 1)
    plt.setp(ax, xticks=(), yticks=())

    polynomial_features = PolynomialFeatures(degree=degrees[i], include_bias=False)

    # 3) Pipeline to assemble steps to be cross-validated while setting different parameters.
    linear_regression = LinearRegression()
    pipeline = Pipeline([
            ("polynomial_features", polynomial_features),
            ("linear_regression", linear_regression)
        ])

    # 4) Fit the polynomials with the help of the pipeline
    pipeline.fit(X[:, np.newaxis], y)

    # 5) Evaluate the models using crossvalidation
    scores = cross_val_score(pipeline, X[:, np.newaxis], y, scoring="neg_mean_squared_error", cv=10)

    # 6) Calculate mean score and print results for the iterated degrees.
    score_mean = -scores.mean()
    print(f"  degree={degrees[i]:4d}, score_mean={score_mean:4.2f},  {polynomial_features}")
```

```
    X_test = np.linspace(0, 1, 100)
    y_pred = pipeline.predict(X_test[:, np.newaxis])

    # 7) Plotting details
    plt.plot(X_test, y_pred           , label="Model")
    plt.plot(X_test, true_fun(X_test), label="True function")
    plt.scatter(X, y, edgecolor='b', s=20, label="Samples")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.xlim((0, 1))
    plt.ylim((-2, 2))
    plt.legend(loc="best")
    plt.title("Degree {}\nMSE = {:.2e}(+/- {:.2e})".format(degrees[i], -scores
.mean(), scores.std()))

plt.show()
print('OK')
```

```
Iterating...degrees= [1, 4, 15]
  degree=   1, score_mean=0.41,  PolynomialFeatures(degree=1, include_bias=Fa
lse, interaction_only=False)
  degree=   4, score_mean=0.04,  PolynomialFeatures(degree=4, include_bias=Fa
lse, interaction_only=False)
  degree=  15, score_mean=182815433.48,  PolynomialFeatures(degree=15, includ
e_bias=False, interaction_only=False)
```




```
OK
```

Qb results

7 main steps were written as comments in the code.
The code starts off by generating the X and y data and then moves into a for-loop that iterates the degrees. The for-loop creates a pipeline from scikit-learn library and uses it to assemble the steps to be cross-validated which then will be fit with the use of the pipeline.
Then the code evaluates the model with cross-validationand prints the results for the iterated degrees and plots.

Qb Explain the capacity and under/overfitting concept

Write a textual description of the capacity and under/overfitting concept using the plots in the code above.

What happens when the polynomial degree is low/medium/high with respect to under/overfitting concepts? Explain in details.

Qb Results

The capacity of the plots goes hand-in-hand with the amount of polynomial degrees used.
It can be seen on the first plot that the capacity is too low and thus the plot is underfitting.
The second plot shows some noise between the model and true function, yet it is still underfitting but the capacity seems to be at the right amount, because increasing the capacity would lead to issues similar to the last plot.
The last plot shows that the capacity of the model is too high and thus it generates a lot of noise when comparing the model with the true function. The model is trying to create a perfect fit for the samples and ends up making it worse because it cannot handle the capacity.

Thus to summarize, polynomial degrees that are too low would result in an underfitting solution. Degrees of medium would result in a best possible fit. Degrees of high would lead to an overfitting solution.

Conclusion

We have learned the concept of polynomial fitting by reviewing and explaining the steps within the given code. Furthermore the concepts of capacity and under/overfitting has been explained and how polynomial degrees influence them.