Abstract

This is the abstract

1 Ordinary Least Squares on the Franke Function

In this section, we will generate our target values by sampling the Franke Function on the unit square. This function, which is widely used when testing interpolation and fitting algorithms, is given by

$$\begin{aligned}
f(x,y) = &\frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4}-\frac{(9y-2)^2}{4}\right)\\
&+\frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49}-\frac{(9y+1)^2}{10}\right)\\
&+\frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4}-\frac{(9y-3)^2}{4}\right)\\
&-\frac{1}{5}\exp\left(-(9x-4)^2-(9y-7)^2\right)
\end{aligned}$$

It should be noted right away that even if the Franke function is exponential in x and y, Taylor's theorem should guarantee that we will be able to get quite close to the original Franke data by using fifth-degree polynomials. In addition to the "pure" Franke values given by the function, we will generate a couple of perturbed variation by adding normally distributed noise. This will give us the opportunity to study how it will be more difficult to create a good fit to the data as it grows more complex.

1.1 Generating and visualizing data

The Franke data are generated in the notebook `20190920-Generating-Franke-data.ipynb`, which relies on `src/data/generate_data.py`. By varying the noise term, we get three sets of target values, which are stored in separate files:

- `no_noise.csv` with no noise added to the Franke data
- `some_noise.csv`, where normally distributed noise with mean 0 and standard deviation 0.1 is added to the Franke data
- `noisy.csv`, where normally distributed noise with mean 0 and standard deviation 0.9 is added to the Franke data

The data are stored in `data/generated/`. Of course, the data could easily have been generated on the fly as needed, but by storing it at this point, we facilitate reproducibility, as we do not run the risk of generating lots of datasets that may look similar, but actually are different from each other. From this point on, we will work with the data read from the files, and nothing else.

In the same notebook (`20190920-Generating-Franke-data.ipynb`) we also generate a feature matrix X. In addition to the original grid points $\{(x = 0.05i, y = 0.05j) : i, j = 0, ..., 19\}$ we include features of the form $x^m y^n$, where m and n are non-negative integers and $m + n \leq 5$. This is in order to perform polynomial regression analysis, which is just another name for ordinary linear regression performed on a feature matrix augmented by polynomial combinations of the original features. The heavy lifting in constructing the polynomial features is performed by the class `PolynomialFeatures` in `src/features/polynomial.py`. Again, we choose to store the generated feature matrix for subsequent use (`data/generated/X.csv`).

We now have one common feature matrix and three different sets of target values. In the notebook `20190905-visualizing-franke.ipynb`, we perform some simple exploratory data analysis on those datasets.

2 Resampling techniques, adding more complexity

3 Bias-variance tradeoff

4 Ridge regression on the Franke function with resampling

5 Lasso regression on the Franke function with resampling

6 Introducing real data

7 OLS, Ridge and Lasso with resampling