Introduction to Numerical Optimization
Assignment 3

May 26, 2022

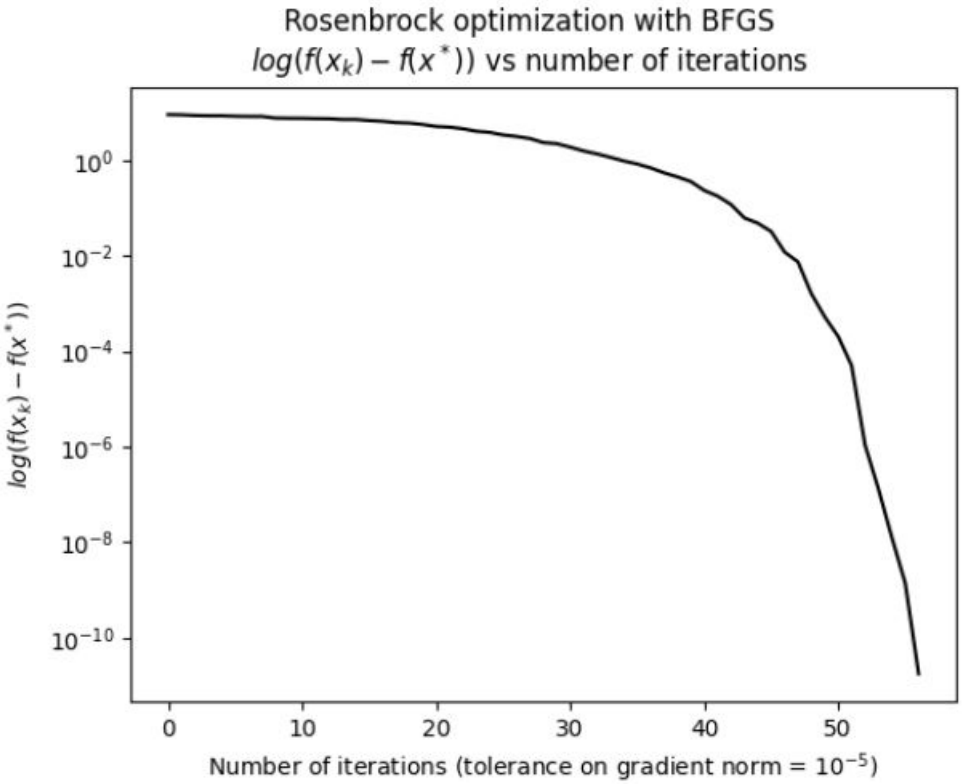
1 Quasi-Newton Methods

1.1 BFGS Method Implementation

Implementation in code

1.2 Find the Minimum of the Rosenbrock Function

Implementation in code



The plot seems similar to the newton method rosenbrock convergence and is much better than simple gradient decent rate of convergence.

1.3 Neural Networks

1.3.4 Explicit Expression of the Neural Network Model

- Hidden layer 1:

$$x^1 \in \mathbb{R}^2, W^1 \in \mathbb{R}^{2\times 4}, b^1 \in \mathbb{R}^4$$

- Hidden layer 2:

$$x^2 \in \mathbb{R}^4, W^2 \in \mathbb{R}^{4\times 3}, b^2 \in \mathbb{R}^3$$

- Output layer:

$$x^3 \in \mathbb{R}^3, W^3 \in \mathbb{R}^{3\times 1}, b^3 \in \mathbb{R}$$

Now we calculate the layers' outputs:

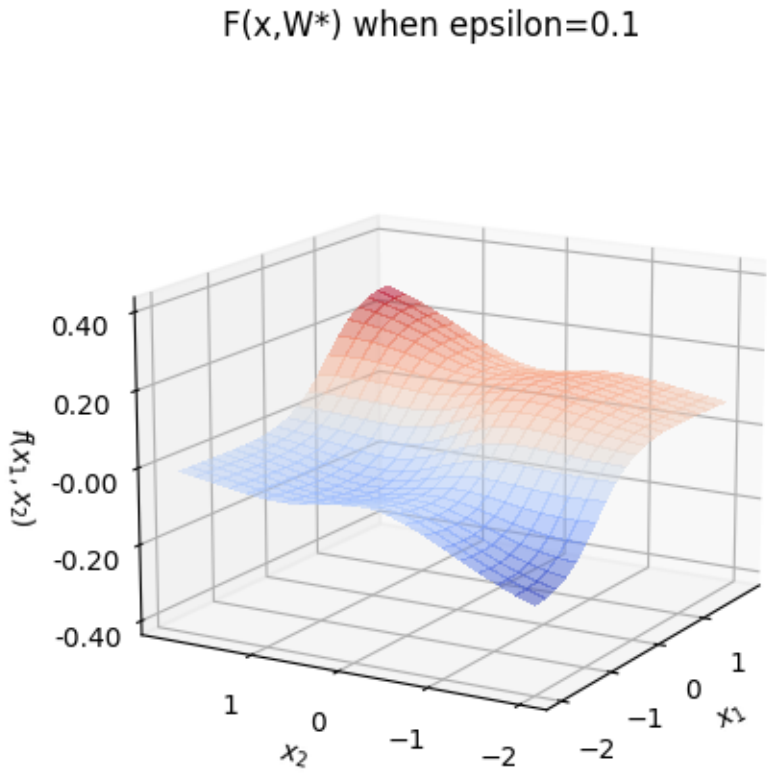
$$x^1 = x$$
$$x^2 = \phi(W^{1^T} x^1 + b^1)$$
$$x^3 = \phi(W^{2^T} \phi(W^{1^T} x^1 + b^1) + b^2)$$
$$F\left(x|\mathcal{W}\right) = \phi(W^{3^T} \phi(W^{2^T} \phi(W^{1^T} x^1 + b^1) + b^2) + b^3)$$

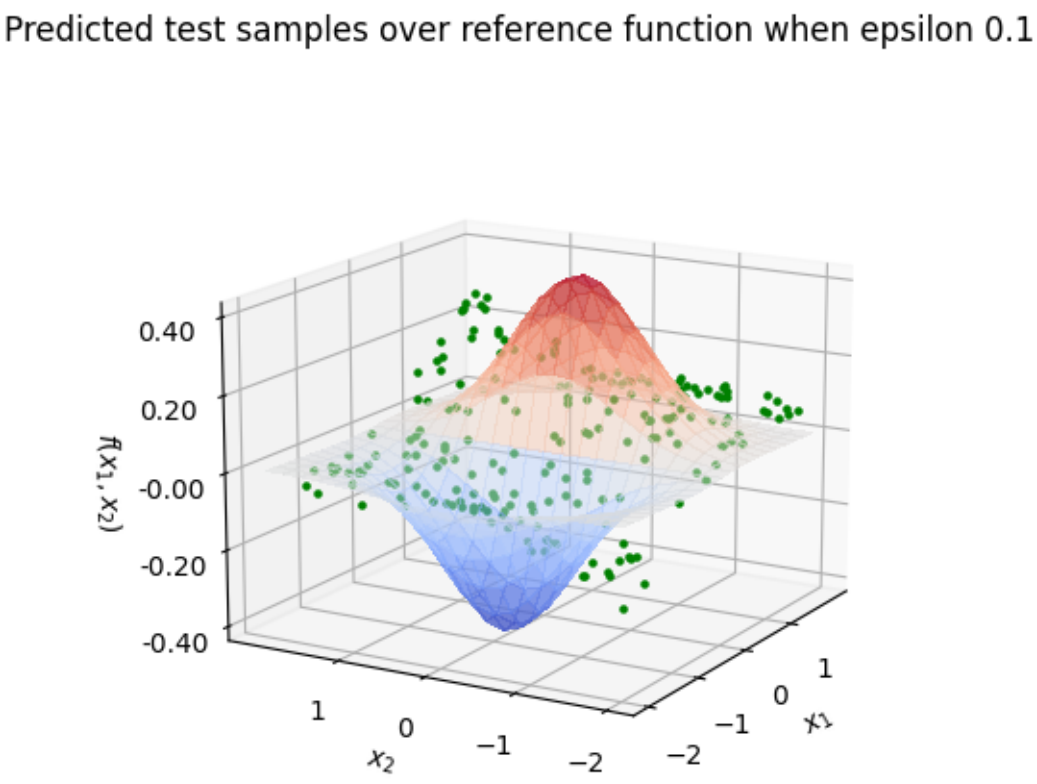
1.3.7 Evaluating the Loss Function's Derivative with Respect to Network's Output of Single Training Example

$$\frac{\partial L}{\partial F\left(x^i, \mathcal{W}\right)} = \frac{\partial}{\partial F\left(x^i, \mathcal{W}\right)} \{\sum_{i=1}^{n} (F(x^i, \mathcal{W}) - y^i)^2\} = 2(F(x^i, \mathcal{W}) - y^i)$$

1.3.14 Combining it All Together

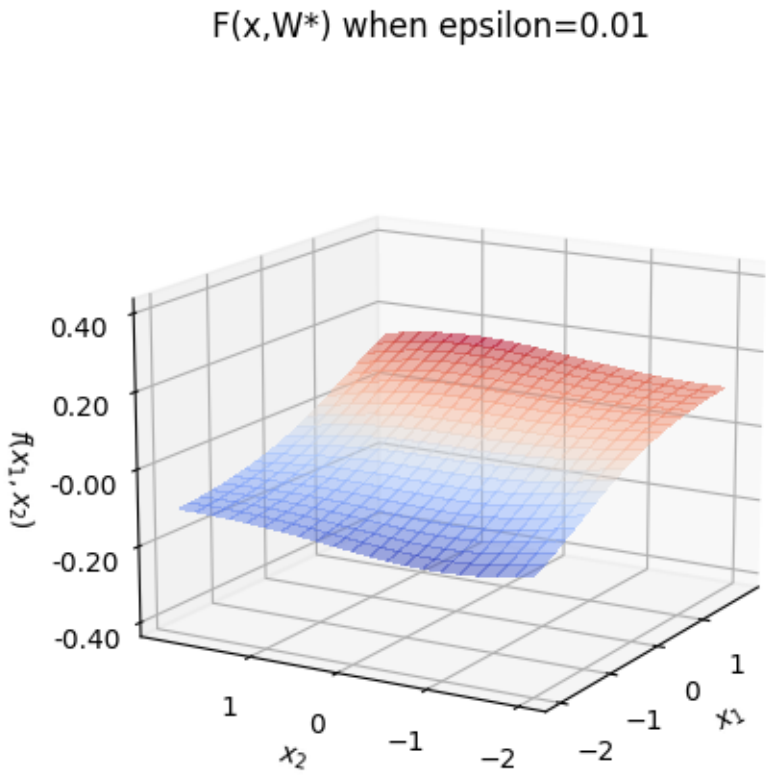
$\epsilon = 0.1$:

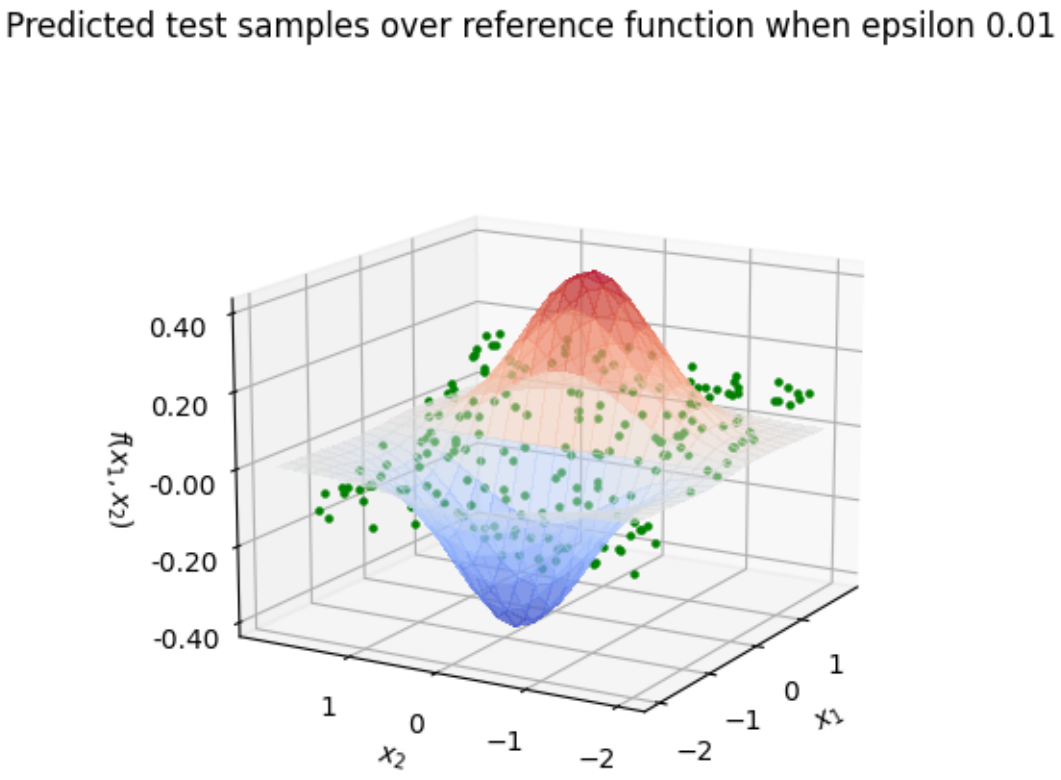




As seen on plots, with $\epsilon = 0.1$ the model approximation is still far and biased from the true function as we didn't train it enough to the needed accuracy (small enough loss or gradient norm).

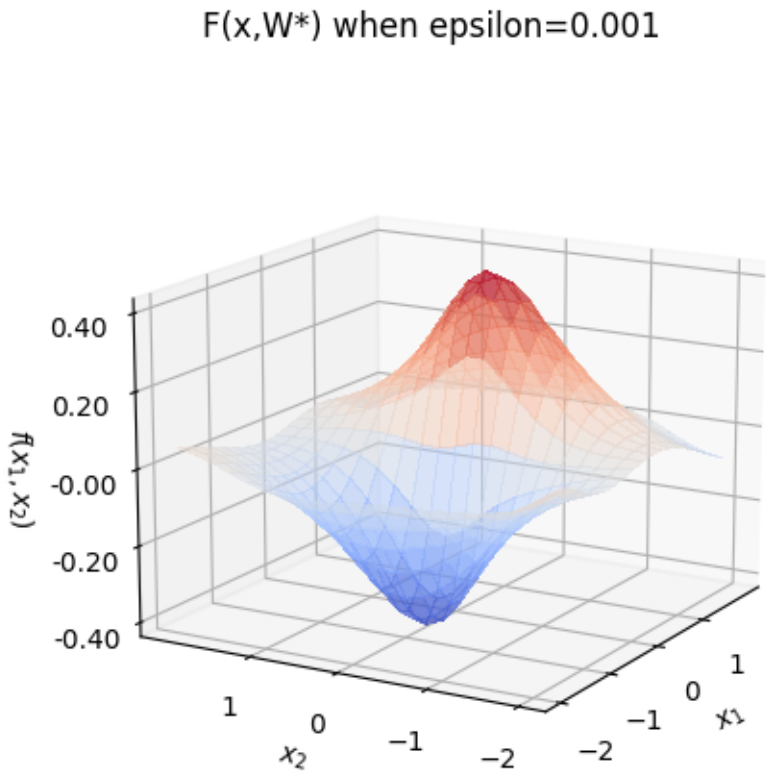
$\epsilon = 0.01$:

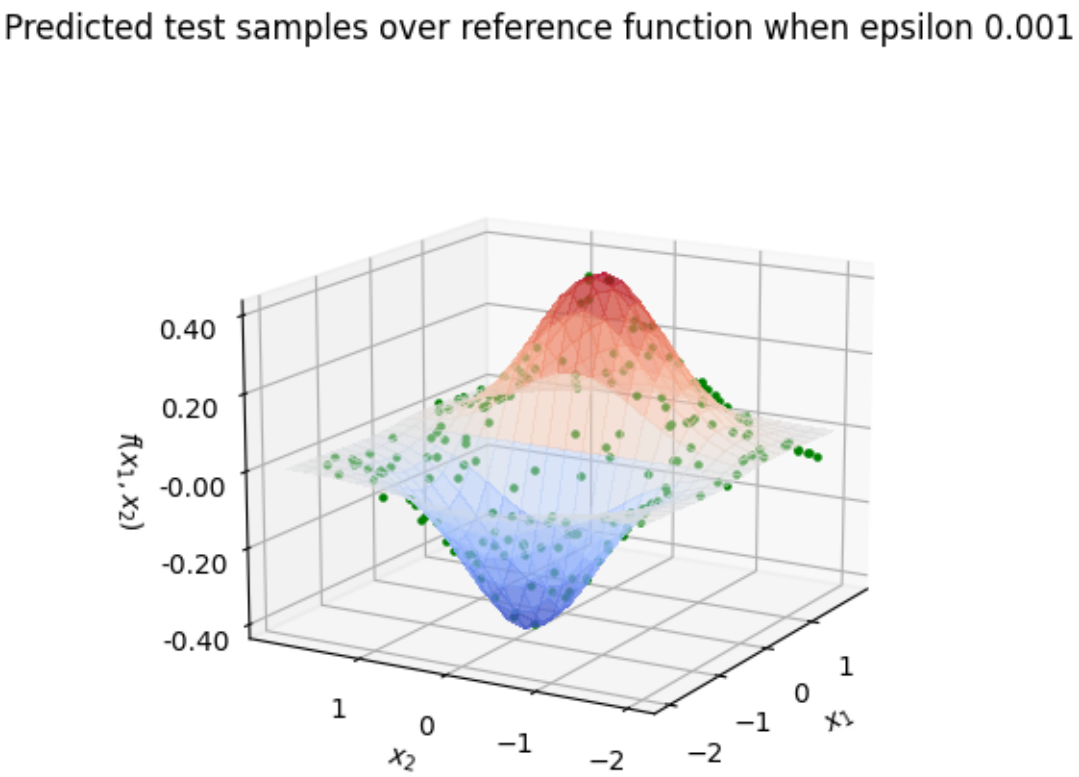




With $\epsilon = 0.01$, we get better approximation compared to the previous one, but still there is a lot of room for improvement as the bias is still big.

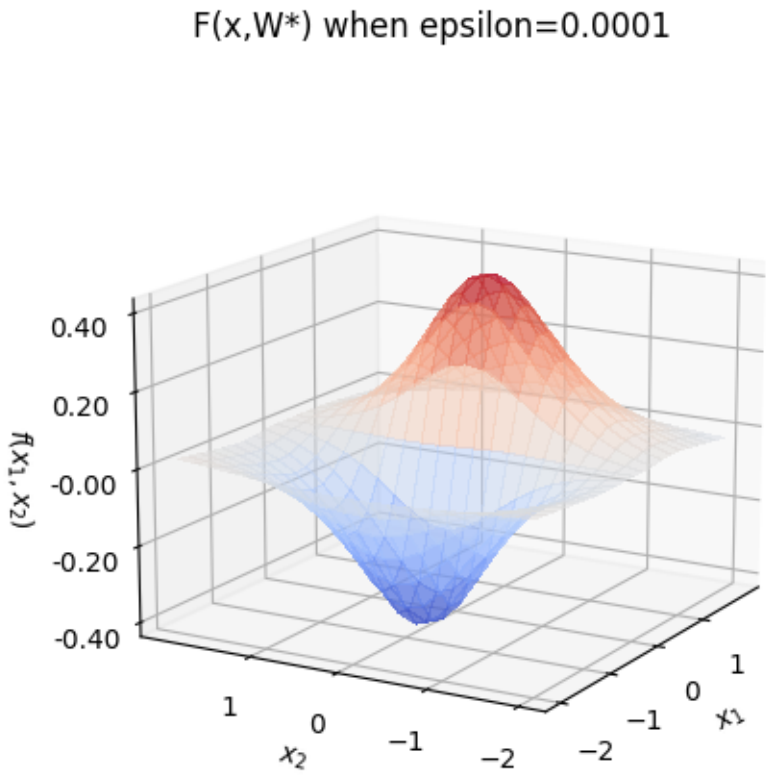
$\epsilon = 0.001$:

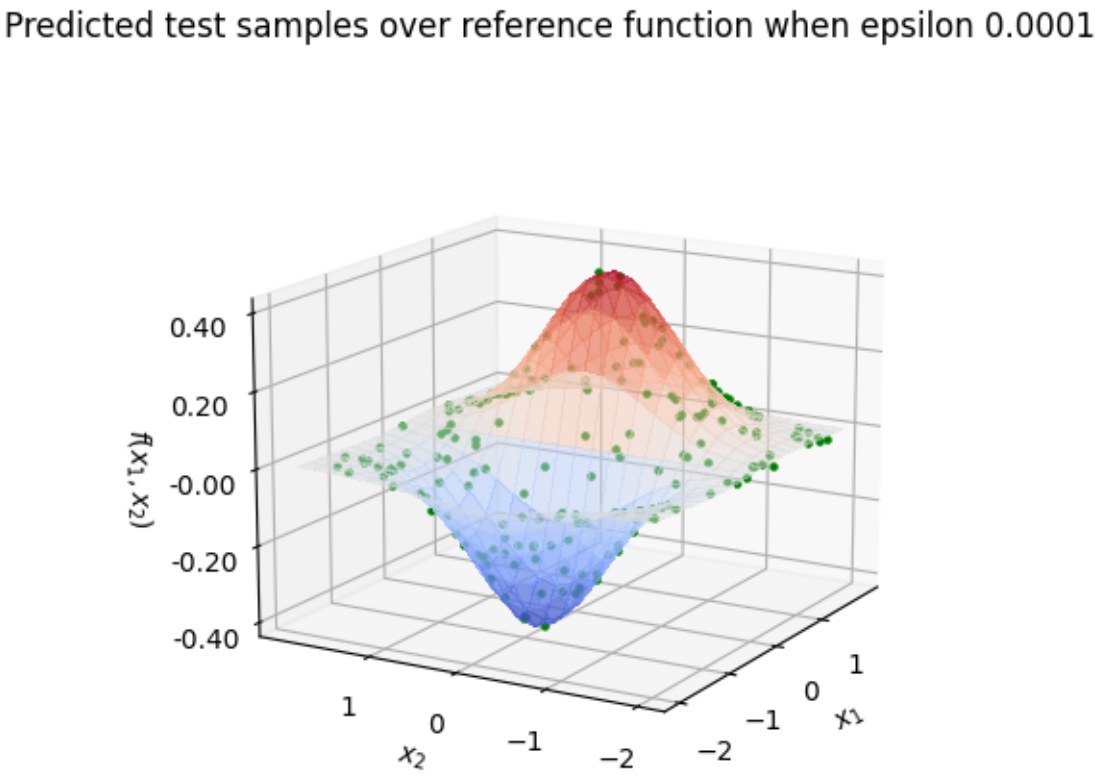




With $\epsilon = 0.001$, we are very close to the true function but still there are mismatches and gaps at the corners as these locations suffer from less data to train on.
Training more (with even the same data but with more epochs) gives even better results as we will see next with even smaller ϵ

$\epsilon = 0.0001$:





With $\epsilon = 0.0001$, we get smaller less (better minimum point from all others) and due to longer training we get very accurate approximation of the true function.
The danger with training on small amount of data is over fitting to the training data, but we can see we didn't get there yet. From plots and final loss it seems to predict almost perfectly the true function.