VANDERBILT UNIVERSITY

Homework 1

CS 8395 - Special Topics in Deep Learning

Due Friday, Feb 09, Midnight

1 Gradient Descent and Convergence Rate [40]

In class we looked at the convergence rate of the Gradient Descent (GD) algorithm. Here, we would like to numerically validate the results we derived in the class for a nonlinear regression problem. Assume that we have data that was generated via:

$$y = f^*(x) + \epsilon \tag{1}$$

for some unknown f^* and where ϵ denotes noise. Given a set of data points $\{(x_n, y_n)\}_{n=1}^N$, we would like to approximate f^* via a parametric function as follows:

$$f(x) = \sum_{m=0}^{M-1} w_m \phi_m(x) = w^T \phi(x) \tag{2}$$

to find the optimal parameters we use the Mean Squared Error (MSE) loss function:

$$\begin{aligned} loss =& \frac{1}{2}\sum_{n=1}^{N}(f(x_n) - y_n)^2 = \frac{1}{2}\sum_{n=1}^{N}(w^T\phi(x_n) - y_n)^2 \\ =& \frac{1}{2}\|\Phi^T w - y\|^2 \end{aligned} \tag{3}$$

For this problem, we use Gaussian feature maps, $\phi_m = \mathcal{N}(\mu_m, 0.25)$, where:

$$\mu_m = 1.25 * m - 3.75, \quad \text{for } m \in \{0, 1, ..., 6\}.$$

Figure 1 shows the data and the basis functions used.

1. Write down the gradient descent updates for this problem, with learning rate ϵ.
2. Derive the theoretical upper bound for ϵ that ensures convergence.
3. Use the data, calculate the eigenvalues of the Hessian matrix and provide the numerical upper bound of ϵ for this problem.

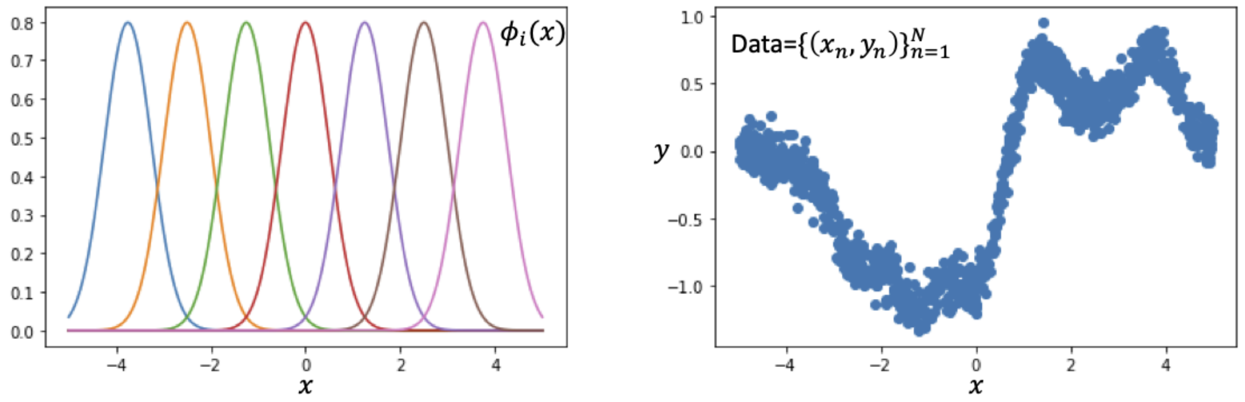

Figure 1: Data and the nonlinear feature maps used in this problem.

4. Set $\epsilon = \frac{1}{\lambda_{max}}$, where λ_{max} is the maximum eigenvalue of the Hessian matrix. Run the gradient descent updates for 1000 steps, and log the **loss value** as well as the **norm-squared of the gradient** as a function of number of steps. Provide the following two plots as part of your write-up:
   (a) Loss as a function of number of iterations
   (b) The logarithm of norm-squared of the loss compared with the logarithm of the convergence upper bound, i.e.,
   $$log(\frac{2\lambda_{max}(loss(w^0) - loss(w^*))}{t+1})$$
   where t denotes the iterations starting from $t = 0$.
5. Plot the data, as well as your optimized regression curve.

The data can be found in 'hw1_p1.npy.'

2 Multi-Layer Perceptrons (MLP) [30pt]

We have seen that multilayer perceptrons are universal function approximators. Here, our goal is to understand the behavior of a MLP in a very simple example. We would like to solve the classification problem shown in Figure 2. For this classification/regression problem provide your answer to the following questions:

1. What is the smallest MLP that can solve this problem? Provide your architecture (draw your network), the weights, biases, and activations you used.

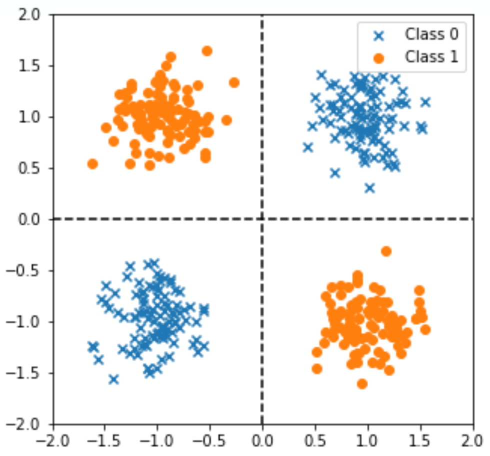

Figure 2: Dataset.

2. Code your model, and train it on the provided data (using GD). You can use the MLP code in Lecture 3. Repeat your training 10 times (with different network initializations), and report a histogram of the network's training accuracy on these ten runs. Was your network able to find the solution in every single run? If not, what is preventing your network from finding the right solution?

3. Repeat the above experiment, however, this time increase the length of your hidden layer (e.g., to 10-100). Report your training accuracy histogram of the ten runs. Compare your results for the larger and smaller networks. Interpret your results.

The data can be found in 'hw1_p2.npy.'

3 Bias, Variance, and the Role of Over-Param. [30pt]

In this problem, we are going to study the effect of overparameterization with and without regularization. We assume that we have observed 10 sets of data generated from the same model:

$$y = f^*(x) + \epsilon \tag{4}$$

See figure 3 for the 10 sets of observations. You are required to solve the MSE problem per observation (i.e., the 10 observations), using:

$$f(x) = \sum_{m=0}^{M-1} w_m \phi_m(x) = w^T \phi(x) \tag{5}$$

For $\phi_m = \mathcal{N}(\mu_m, s)$, where μ_m and s are generated via:

```
mu=torch.linspace(-5,5,M+2)[1:-1]
s=2/(M+1)
```

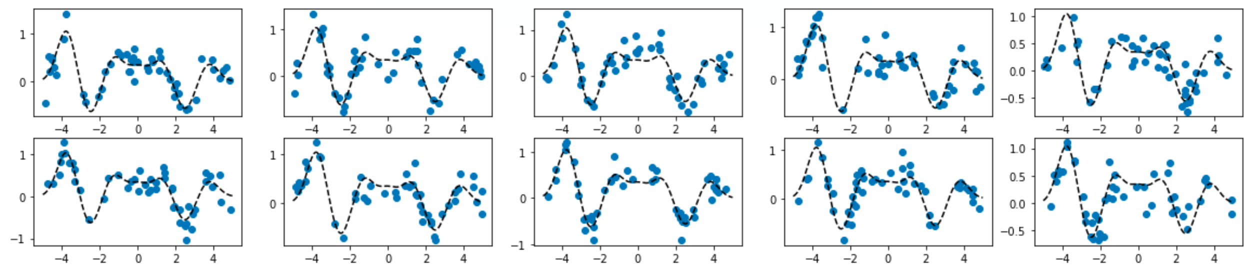

Figure 3: Observed data.

The optimization problem you need to solve is:

$$\text{argmin}_w \frac{1}{2}\|\Phi^T w - y\|^2 \tag{6}$$

Use the closed form solver to find the solution.

1. Solve the problems with $M = 7$. Let f_i denote your calculated regression per problem, i.e., $i \in [0, ..., 9]$.
   - Plot f_i vs f^* (in 'hw1_p3_gt.npy') for $\forall i$, in a 2×5 plot similar to Figure 3
   - Calculate $\bar{f} = \frac{1}{10}\sum_{i=1}^{10} f_i$ and plot it against f^*.
   - Calculate the bias and variance of your hypothesis class.
2. Repeat the above experiment with $M = 21$. Regenerate the plots and calculate the bias and covariance. Compare your solution of $M = 7$ with $M = 21$.
3. Write down the closed for solution for ridge regression, i.e.,

   $$\text{argmin}_w \frac{1}{2}\|\Phi^T w - y\|^2 + \lambda\|w\|^2. \tag{7}$$

   Solve the regression problems with $M = 21$ and $\lambda = 0.1$. Regenerate the previous plots and calculate the bias and variance. What happens to the bias and variance?

The data can be found in 'hw1_p3.npy' and the ground truth in 'hw1_p3_gt.npy.'