Activity 3

Deep Learning Lab

October 4, 2019

1 Assignment 1

Consider the polynomial p given by

$$p(x) = x^3 + 2x^2 - 4x - 8 = \sum_{i=1}^{4} w_i^* x^{i-1},$$

where $\mathbf{w}^* = [-8, -4, 2, 1]^T$.

Consider also an iid dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, where $y_i = p(x_i) + \epsilon_i$, and each ϵ_i is drawn from a normal distribution with mean zero and standard deviation $\sigma = 1/2$.

If the vector $\mathbf{w}^*$ were unknown, linear regression could estimate it given the dataset $\mathcal{D}$. This would require applying a feature map to transform the original dataset $\mathcal{D}$ into an expanded dataset $\mathcal{D}' = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}_i = [1, x_i, x_i^2, x_i^3]^T$.

Such data generation and expansion is partially illustrated in the code presented below.

Listing 1: Polynomial regression dataset generation (incomplete).

```
def create_dataset(w_star, x_range, sample_size, sigma, seed=None):
    random_state = np.random.RandomState(seed)

    x = random_state.uniform(x_range[0], x_range[1], (sample_size))
    X = np.zeros((sample_size, w_star.shape[0]))
    for i in range(sample_size):
        X[i, 0] = 1.
        for j in range(1, w_star.shape[0]):
            X[i, j] = ? # Incomplete

    y = X.dot(w_star)
    if sigma > 0:
        y += random_state.normal(0.0, sigma, sample_size)

    return X, y
```

1. Adapt the snippet presented in Slides 37-38 to perform polynomial regression using a dataset $\mathcal{D}'$ created using the code presented above (after

it is properly completed). More specifically, find an estimate of $\mathbf{w}^* = [-8, -4, 2, 1]^T$ supposing that such vector is unknown. Each x_i should be in the interval $[-3, 2]$. Use a sample of size 100 created with a seed of 0 for training, and a sample of size 100 created with a seed of 1 for validation. Let $\sigma = 1/2$.

2. Find a suitable learning rate and number of iterations for gradient descent.

3. Plot the training and validation losses as a function of the gradient descent iterations. You can use tensorboard (see Slides 28-29) or store this information in a list after each iteration.

4. Plot the polynomial defined by $\mathbf{w}^*$ and the polynomial defined by your estimate $\hat{\mathbf{w}}$. Plot the training dataset.

5. Document what happens when the training dataset is reduced to 50, 10, and 5 observations.

6. Document what happens when σ is increased to 2, 4, and 8.

7. **Bonus**: Reduce your training dataset to 10 observations, and compare fitting a polynomial of degree three (as before) with fitting a polynomial of degree four (which does not match the underlying polynomial p). Plot the resulting polynomials and document the validation loss.

You should deliver the following by the deadline stipulated on iCorsi3:

- Report: a single *pdf* file that clearly and concisely provides evidence that you have accomplished each of the tasks listed above. The report should not contain source code (not even snippets). Instead, if absolutely necessary, briefly mention which functions were used to accomplish a task.

- Source code: a single Python script that could be easily adapted to accomplish each of the tasks listed above. The source code will be read superficially and checked for plagiarism. Unless this reveals that your code is suspicious, your grade will be based entirely on the report. Therefore, if a task is accomplished but not documented in the report, it will be considered missing. Note: Jupyter notebook files are not acceptable.