SF2525 HW4

Wille Viman Vestberg willev@kth.se

21 April 2023

1 Part 1.2

We consider the iterations

$$\theta_n = \theta_{n-1} - \Delta t \frac{df}{d\theta}(\theta_n, Y_n)$$

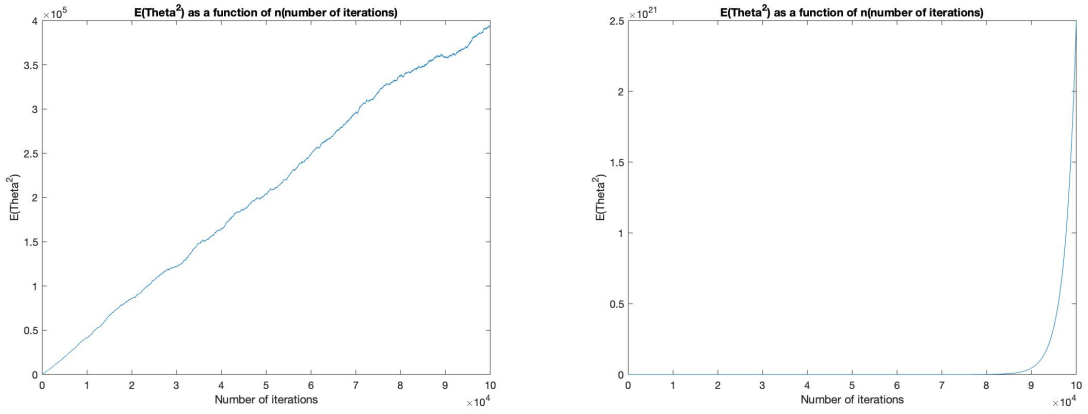
where

$$\begin{aligned} \theta_0 &= 1 \\ Y_n &\sim N(0,1) \\ f(\theta, Y) = |\theta - Y|^2 \rightarrow \frac{df}{d\theta}(\theta, Y) &= 2(\theta - Y) \end{aligned}$$

1.1 1a

For what range of Δt does $\mathbb{E}[|\theta_n|^2]$ remain bounded?

I get that this is true for all $\Delta t \leq 1$ to verify this i show a figure of the expected value for 1 bounded stepsize and one unbounded.



(a) Using Δ t = 1, we can still bound it

(b) Using Δ t = 1.0001, it blows up

Figure 1: Figure illustrating one Δt for which we can still bound $\mathbb{E}[|\theta_n|^2]$ and one for which we can not.

We could also show this analytically:

$$
\begin{aligned}
E|\theta_n^2| &= E((\theta_{n-1} - \Delta t \frac{df}{d\theta}(\theta_n, Y_n))^2) \\
&= E(\theta_{n-1}^2) - 2\Delta t E(\theta_{n-1} \frac{\partial f}{\partial \theta}) + \Delta t^2 E(\frac{\partial f}{\partial \theta}^2) \\
&= E(\theta_{n-1}^2) - 2\Delta t E(\theta_{n-1} 2(\theta_{n-1} - Y_{n-1})) + \Delta t^2 E(4(\theta_{n-1} - Y_{n-1})^2) \\
&= E(\theta_{n-1}^2) - 4\Delta t E(\theta_{n-1}^2) + 4\Delta t^2 E((\theta_{n-1} - Y_{n-1})^2) \\
&= E(\theta_{n-1}^2) - 4\Delta t E(\theta_{n-1}^2) + 4\Delta t^2 E(\theta_{n-1}^2 - 2\theta_{n-1} Y_{n-1} + Y_{n-1}^2) \\
&= E(\theta_{n-1}^2) - 4\Delta t E(\theta_{n-1}^2) + 4\Delta t^2 E((\theta_{n-1}^2) + 1) \\
&= (1 - 4\Delta t + 4\Delta t^2) E(\theta_{n-1}^2) + 4\Delta t^2 \\
&= (1 - 2\Delta t)^2 E(\theta_{n-1}^2) + 4\Delta t^2
\end{aligned}
$$

We want $E(\theta_{n-1}^2) < C$ hence

$$
\begin{aligned}
(1 - 2\Delta t)^2 E(\theta_{n-1}^2) + 4\Delta t^2 &< (1 - 2\Delta t)^2 C + 4\Delta t^2 \\
(1 - 2\Delta t)^2 C + 4\Delta t^2 < C \implies 4\Delta t^2 &\leq (4\Delta t - 4\Delta t^2) C
\end{aligned}
$$

$$
\begin{cases}
\frac{4\Delta t^2}{(4\Delta t - 4\Delta t^2)} = \frac{\Delta t}{(1 - \Delta t)} \leq C & if \Delta t \leq 1 \\
\frac{4\Delta t^2}{(4\Delta t - 4\Delta t^2)} = \frac{\Delta t}{(1 - \Delta t)} > C & if \Delta t > 1
\end{cases}
$$

But note that if $\frac{\Delta t}{(1-\Delta t)} > C$, C is a negative number which contradicts our assumption that $\theta_0 = 1$ Hence $E(\theta_n^2) < C$ will not hold for each n. Thus we want $\Delta t \leq 1$ which agrees with what we see in figure 1.

Now lets find θ_* which is the theta that minimizes the expression

$$
E(f(\theta, Y))
$$

start by expanding the absolute value

$$
\begin{aligned}
E(f(\theta, Y)) &= E(|\theta - Y|^2) = E((\theta - Y)^2) \\
&= E(\theta^2 - 2\theta Y + Y^2) = E(\theta^2) - 2E(\theta Y) + E(Y^2) \\
\theta^2 - 2\theta E(Y) + E(Y^2) &= \theta^2 - 0 + Var(Y) + E(Y)^2 \\
&= \theta^2 + 1
\end{aligned}
$$

Where I've used that $Y \sim N(0, 1), \rightarrow E(Y) = 0, Var(Y) = 1$

Hence the value of θ that minimizes the expression is:

$$
\frac{d}{d\theta}(\theta^2 + 1) = 0 \rightarrow \theta = 0
$$

To establish the convergence rate we note that $E(|\theta_n - \theta_*|^2) = E(|\theta_n|^2)$ since $\theta_* = 0$

$$E(|\theta_n|^2) = (1-2\Delta t)^2 E(\theta_{n-1}^2) + 4\Delta t^2$$
$$= (1-2\Delta t)^{2n} E(\theta_0^2) + (1-2\Delta t)^{2(n-1)} 4\Delta t^2 + ... + (1-2\Delta t)^{2(n-n)} 4\Delta t^2 + 4\Delta t^2$$
$$= (1-2\Delta t)^{2n} E(\theta_0^2) + 4\Delta t^2 \sum_{i=1}^{n-1} (1-2\Delta t)^{2i}$$
$$lim\ n \to \infty = (1-2\Delta t)^{2n} + 4\Delta t^2 \frac{1}{4\Delta t - 4\Delta t^2} = \frac{\Delta t}{1-\Delta t}$$

assuming $(1-2\Delta t) < 1$ Hence it converges as: $(1-2\Delta t)^{2n}$

1b

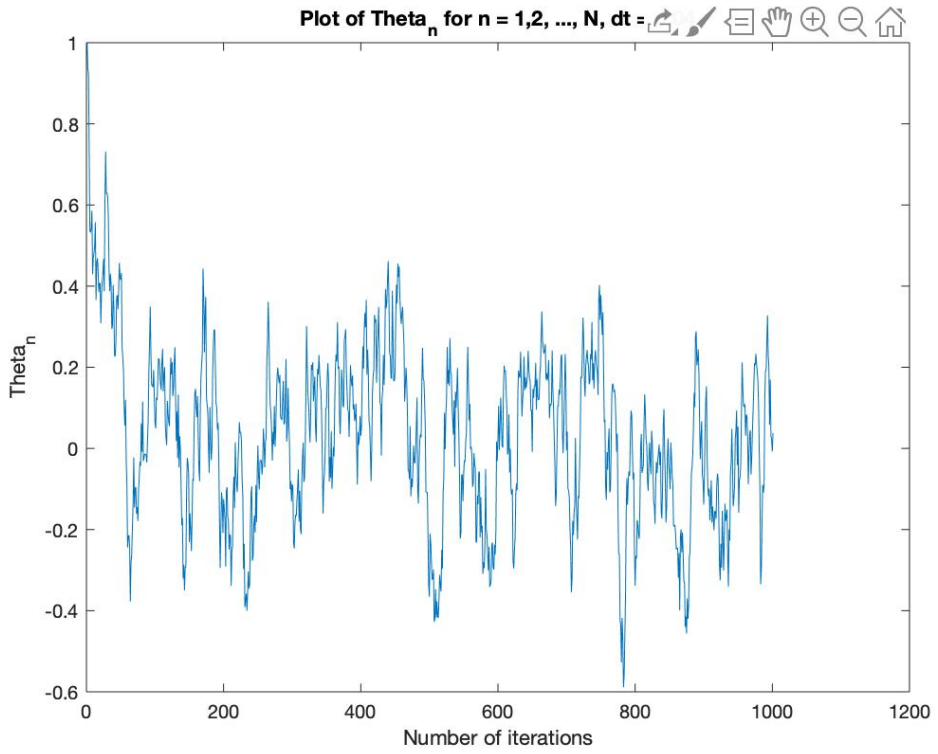


Figure 2: $\Delta t = 0.04$

1c

Show that the stochastic gradient descent iterations are in fact for- ward Euler steps for an Ornstein-Uhlenbeck process.

$$\theta_{n+1} = \theta_n - \Delta t \frac{\partial f}{\partial \theta}(\theta_n, Y_n)$$
$$= \theta_n - 2\Delta t(\theta_n - Y_n) = \theta_n - 2\Delta t\theta_n + 2\Delta t Y_n$$

Note that

$$\theta_{n+1} - \theta_n = -2\theta_n \Delta t + 2\Delta t Y_n$$

And note that $Y_n \sim N(0,1)$ which implies that$\Delta t Y_n \sim N(0, \sqrt{\Delta t})$ Which is exactly the same properties, normally distributed with mean 0 and variance Δt, as ΔW_n has in a wiener process. Hence

$$\theta_{n+1} - \theta_n = -2\theta_n \Delta t + 2\Delta W_n$$

Is the forward Euler Approximation of the Stochastic differential equation.

$$d\theta_t = \lambda(\mu - \theta_t)dt + \sigma dW_t$$

with the parameter values $\lambda = 2, \mu = 0, \sigma = 2$ Which is a Ornstein-Uhlenbeck process.

Formulate also the gradient descent method, based on computing the expected value exactly, to solve the minimization problem $argmin(E(f(\theta, Y))$

With the gradient decent method we start with some initial guess for θ_0 and then we search for new values in the direction of the steepest descent

hence we would get an update scheme for θ_n as

$$\begin{aligned} \theta_{n+1} = \theta_n - \alpha \nabla_\theta E(f(\theta, Y)) = \theta_n - \alpha \nabla_\theta E(|\theta_n - Y_n|^2) \\ = \theta_n - \alpha \nabla_\theta E((\theta_n^2 - 2\theta_n Y_n + Y_n^2) = \theta_n - \alpha \nabla_\theta E((\theta_n^2) + 1) \\ = \theta_n - 2\alpha E((\theta_n) = \theta_n - 2\alpha\theta_n = (1 - 2\alpha)\theta_n \end{aligned}$$

Convergence:

$$\begin{aligned} \theta_n^2 = (1 - 2\alpha)^2 \theta_{n-1}^2 \\ = (1 - 2\alpha)^{2n} \theta_0^2 \end{aligned}$$

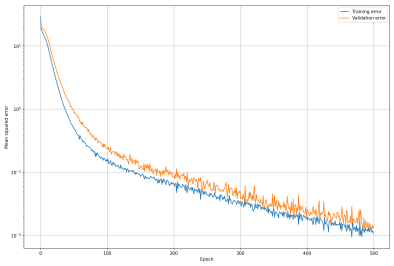
Where α is the step-size. Which will converge to zero as long as $(1 - 2\alpha) < 1$ The convergence is as fast as in part a) but it converges to zero and not to $\frac{\Delta t}{1 - \Delta t}$

Part 2

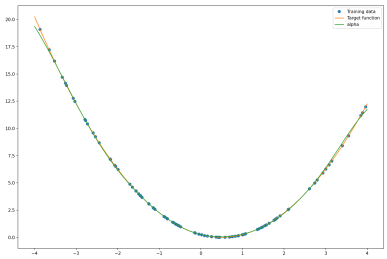
1. Gradually first increase and then gradually decrease dt. Explain what you see. A suitable range to test is 0.05 > dt, cf. Part 1.

If i increase dt I see faster convergence but more noise in both the training and the validation error.

While if I decrease it I see slower convergence but less noise. However, in both when increasing and decreasing dt the learned function looks worse which we see in figure 4. This is also reflected by the test error which is one order larger (0.3170 and 0.2057 versus 0.0207 initially).

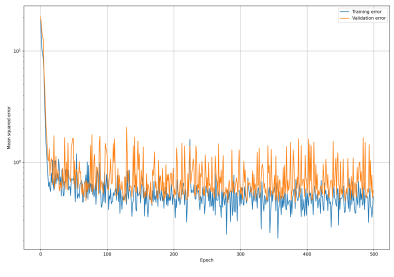


(a) error

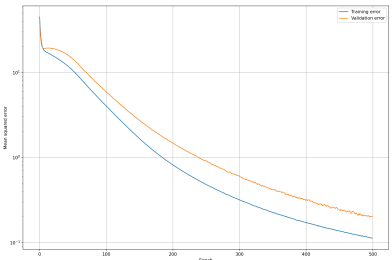


(b) learned function

Figure 3: Error and learned function for the initial parameter settings (dt = 0.005, K = 10, N = 100.

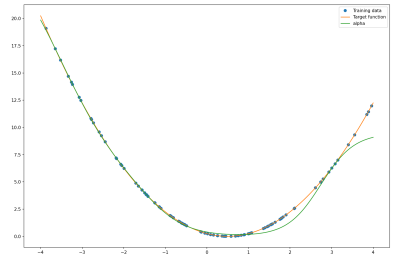


(a) dt = 0.03

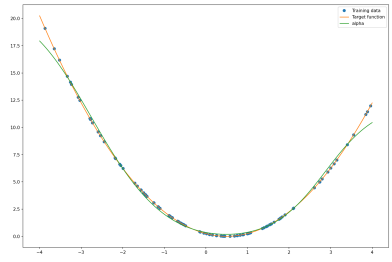


(b) dt = 0.0001

Figure 4: Error for larger and smaller dt



(a) dt = 0.03



(b) dt = 0.0001

Figure 5: Learned function for larger and smaller dt

2. Plot the training error

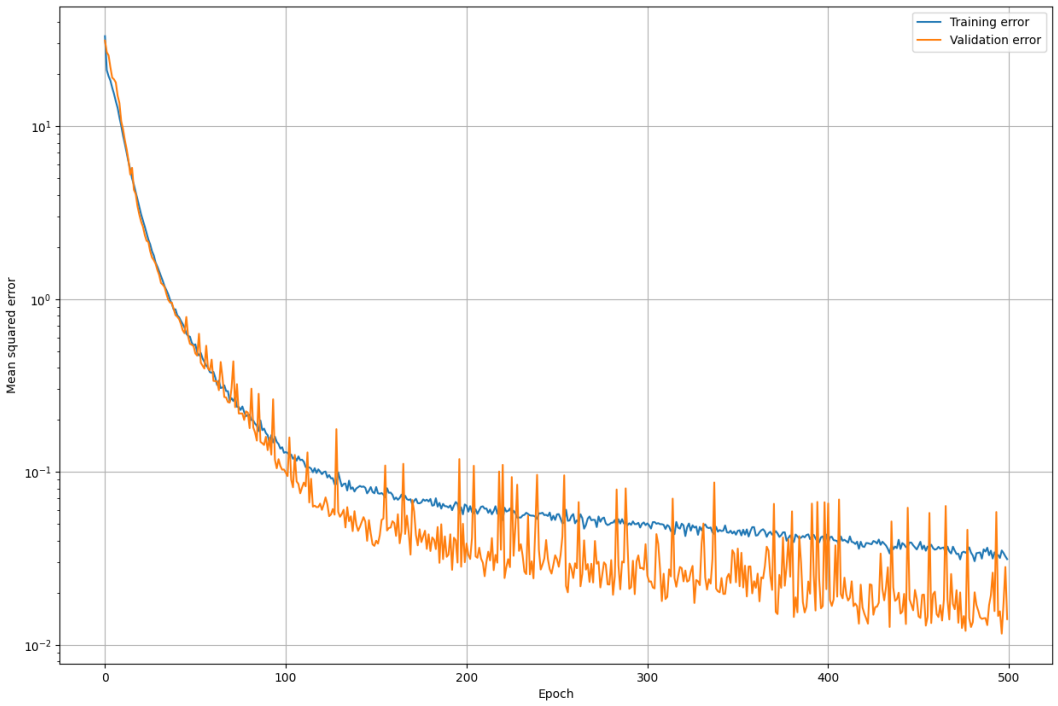
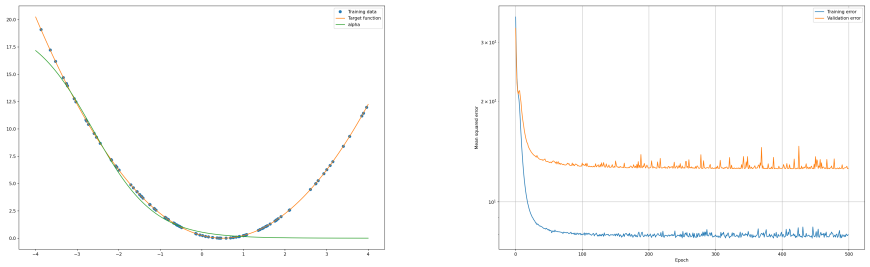


Figure 6: Caption

3. Set K = 1 what do you see?



(a) Learned function

(b) error

Figure 7: setting K = 1

Wee see that the learned function is a sigmoid function since we no longer superimpose several sigmoids in our learned function. The sigmoidfunction is the best fit and wee see that it captures the behaviour on the left good but fails to get the rise since its a sigmoid.

Wee see that the training error never converges, this validates it further.

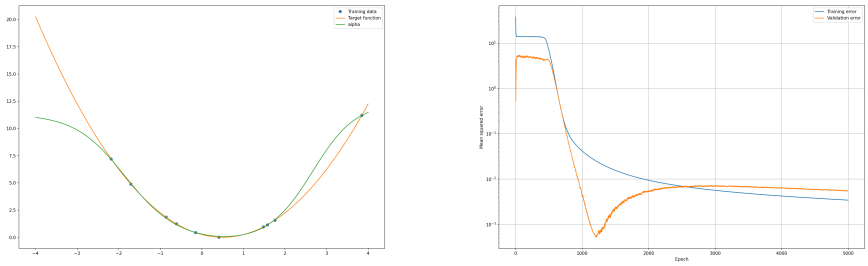
4 For K = 10. What is a suitable choice of N? Motivate your answer.

For K = 10 we start with fewer training points and see that the learned function achieves good performance on the training data but if generalizes poorly. Since the true function is poorly represented by those samples. see figure 7.

If wee increase N to 50 wee see a lot better generalization. ((test loss 0.052 compared to 2.52 with N = 10).

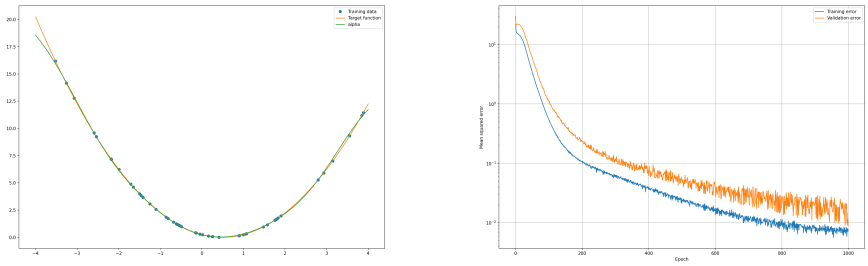
Interestingly choosing a significantly larger N = 500, we get bigger test loss than with N = 100 (test loss 0.021, as with the original configuration). This is because with a larger training set size N we see that the function approximates better around the endpoints but with a little gap in the minimum. Because the MSE is rather small for this little gap here, so the error at the endpoints have larger weight. Instead using fewer points N we get a bigger majority of points around the minimum and hence this part will be better approximated, leading to a overall lower MSE.

To conclude a good choice is N = 100.



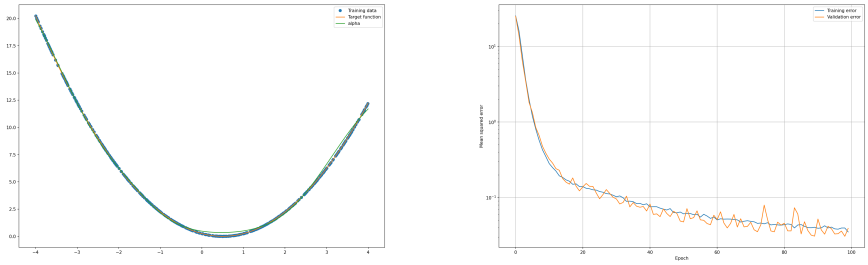
(a) Learned function (b) error

Figure 8: Error and learned function with number of training samples: N = 10 with dt = 0.005 and K = 10.



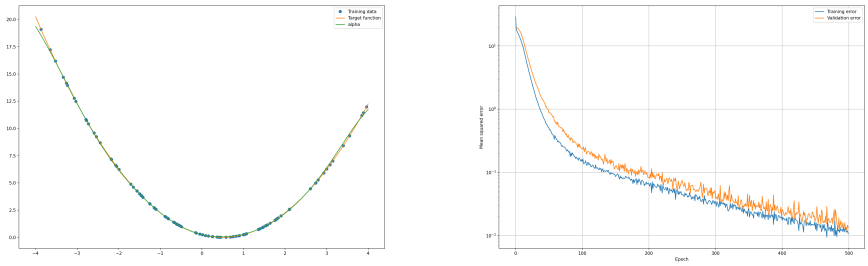
(a) Learned function (b) error

Figure 9: Error and learned function with number of training samples: N = 50 with dt = 0.005 and K = 10.



(a) Learned function (b) error

Figure 10: Error and learned function with number of training samples: N = 500 with dt = 0.005 and K = 10.



(a) Learned function (b) error

Figure 11: Error and learned function with number of training samples: N = 100 with dt = 0.005 and K = 10.

Looking at the images above a good trade-of between sufficient representation of the underlying function and fast convergence seems to be fulfilled best with N = 100.

5

My example of parameter settings that gives a small value of the training error but a large value of the generalization error is N = 10, K=10, dt = 0.005.

Both the training error and the validadtion error are $< 10^{-2}$ while the test loss is 2.52.

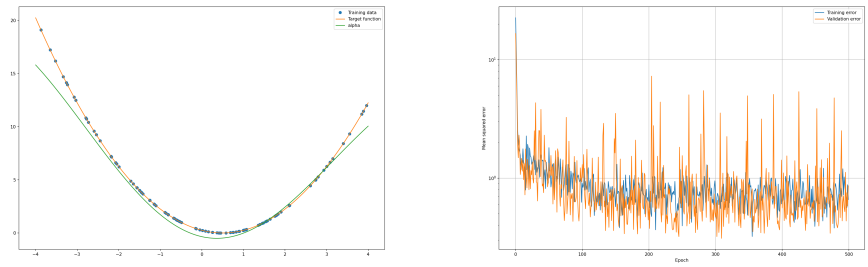
6

First we change the initialization of the bias to large values. Then we use N = 100, K = 10 and dt = 0.0005.

Looking at the results as shown in the figure below, we see that the training and validation loss both do not show any signs of convergence.

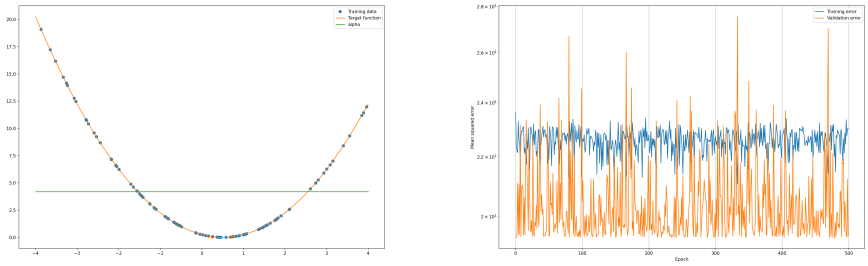
This is confirmed by the fact that the learned function is constant. This seems right since the large biases usually lead to a large input to the sigmoid, and large inputs to the sigmoid outputs 1.

We can get by this problem by using a sinus activation function instead. Now the nodes in the hidden layer outputs different values. The learned function looks more like a sine wave and generally is much closer to the true function (test loss 1.25). And we have convergence now.



(a) Learned function (b) error

Figure 12: sinus activation



(a) Learned function (b) error

Figure 13: sigmoid activation

7

Reformulate the minimization problem to d dimensions.

$$min_{\theta \in R^{(d+2)xK}} E[|\alpha_{\theta(x)-f(x)}|^2]$$

$$f(x) = ||x - [\frac{1}{2}...\frac{1}{2}]^T||, x \in R^d$$

$$\alpha_\theta(x) = \sum_{k=1}^{K} \theta_k^1 \sigma(\theta_k^{2T} x + \theta_k^3), x \in R^d, \theta \in R^{(d+2)xK}$$

where $|| * ||$ is the euclidean norm, $\sigma(x)$ is the sigmoid function.

Below you see how I changed the code to work in d dimensions.

```

N = 100
K = 10
M = 40000
dt = 0.005
T = M*dt
d = 10 #dimension
Nbatch = 1
validation_split_ratio = 0.2
EPOCHS = np.int64(M/(N*(1-validation_split_ratio)/Nbatch))
x_training = np.random.uniform(-4,4,size =(N,d))
x_test = np.random.uniform(-4,4,size =(100,d))

def f(x):
  d = len(x)
  ones = 1/2*np.ones((d,))
  norm = np.linalg.norm(x - ones)
  return norm
y_training = np.zeros(N)
y_test = np.zeros(N)
for i in range(N):
  y_training[i] = f(x_training[i])
  y_test[i] = f(x_test[i])
Input_layer = tf.keras.Input(shape =(d,))
Hidden_layer = keras.layers.Dense(units=K,
                                  activation ="sigmoid",
                                  use_bias =True,
                                  kernel_initializer ='random_normal',
                                  bias_initializer ='zeros',
                                  name= "hidden_layer_1")

Output_layer = keras.layers.Dense(units=1,
                                  use_bias =False,
                                 kernel_initializer='random_normal',
                                  name="output_layer")
```

```
36
37 model = keras.Sequential([Input_layer, Hidden_layer, Output_layer])
38
39 optimizer = tf.keras.optimizers.SGD(learning_rate=dt / d)
40
41 model.compile(optimizer = optimizer,loss='mean_squared_error')
42
43 history = model.fit(x=x_training,
44                     y=y_training,
45                     batch_size=Nbatch,
46                     epochs = EPOCHS,
47                     validation_split = validation_split_ratio,
48                     verbose =1)
49
50 model.evaluate (x=x_test, y=y_test, verbose=1)
51
52 pts = np.meshgrid(*[np.linspace(-4, 4, 300) for _ in range(d)])
53 pts = np.column_stack([pts[i].ravel() for i in range(d)])
54
55 target_fcn_vals = f(pts)
56 alpha_vals = model(pts)
```

Listing 1: Program adapted to d dimensions