417T Homework 2

Matthew DeSantis

2023-10-06

1

(a)

iters	E_in	binary_error_train	binary_error_test	time_to_finish
10^4	0.5448753	0.2631579	0.2689655	0.8
10^5	0.5146686	0.2565789	0.2000000	6.6
10^6	0.4654394	0.1842105	0.1586207	69.8

As can be seen by the good test binary error, the model generalizes well. Increasing the max iterations made the model better and better.

(b)

learning_rate	E_in	binary_error_train	binary_error_test	iterations_to_finish	time_to_finish
0.01	0.4074311	0.1710526	0.1172414	1e+06	67.3
0.10	0.4073821	0.1710526	0.1103448	1e+06	63.9
1.00	0.4073814	0.1710526	0.1103448	1e+06	64.7
4.00	0.4073814	0.1710526	0.1103448	1e+06	66.0
7.00	0.4073814	0.1710526	0.1103448	1e+06	66.7
7.50	0.4073814	0.1710526	0.1103448	1e+06	66.5
7.60	0.4073814	0.1710526	0.1103448	1e+06	63.4
7.70	0.4073814	0.1710526	0.1103448	1e+06	62.7

Note that there were small changes in E_in, just not significant enough to display on the graph. Values were normalized using the means and stds from X_test. It appears that normalizing helped our model slightly, although since we also changed the learning rate it's not certain. Increasing the learning rate very slightly helped improve the model. It's worth noting that at no point did the norm of the gradient go below 10^{-6}, as all models took the same number of steps to finish, but it's likely that the models with the higher learning rate reached the point of diminishing returns more quickly.

2: LFD Problem 2.22

$\overline{g}(x)$ is the expected function, same as $E_D[(g^D(x)]$.

$E_D[E_{out}(g^D)] = E_D[E_{x,y}[(g^D(x) - y(x))^2]]$
$= E_D[E_{x,y}[(g^D(x) - (f(x) + \epsilon))^2]]$
$= E_{x,y}[E_D[(g^D(x) - (f(x) + \epsilon))^2]]$
$= E_{x,y}[E_D[g^D(x)^2 - 2(g^D(x)(f(x) + \epsilon) + (f(x) + \epsilon)^2]]$
$= E_{x,y}[E_D[g^D(x)^2] - 2E_D[(g^D(x)](f(x) + \epsilon) + (f(x) + \epsilon)^2]]$
$= E_{x,y}[E_D[g^D(x)^2] - 2\overline{g}(x)(f(x) + \epsilon) + (f(x) + \epsilon)^2]]$
$= E_{x,y}[E_D[g^D(x)^2] - 2\overline{g}(x)f(x) - 2\overline{g}(x)\epsilon + (f(x) + \epsilon)^2]]$
$= E_{x,y}[E_D[g^D(x)^2] - 2\overline{g}(x)f(x) - 2\overline{g}(x)\epsilon + f(x)^2 + 2f(x)\epsilon + \epsilon^2]]$
$= E_{x,y}[E_D[g^D(x)^2] - 2\overline{g}(x)f(x) + \overline{g}(x)^2 - \overline{g}(x)^2 - 2\overline{g}(x)\epsilon + f(x)^2 + 2f(x)\epsilon + \epsilon^2]]$
$= E_{x,y}[E_D[g^D(x)^2] + (\overline{g}(x) - f(x))^2 - \overline{g}(x)^2 - 2\overline{g}(x)\epsilon + 2f(x)\epsilon + \epsilon^2]]$
$= E_{x,y}[E_D[g^D(x)^2] - \overline{g}(x)^2 + (\overline{g}(x) - f(x))^2 - 2\overline{g}(x)\epsilon + 2f(x)\epsilon + \epsilon^2]]$
$= variance + E_{x,y}[(\overline{g}(x) - f(x))^2 - 2\overline{g}(x)\epsilon + 2f(x)\epsilon + \epsilon^2]]$
$= variance + bias + E_{x,y}[-2\overline{g}(x)\epsilon + 2f(x)\epsilon + \epsilon^2]]$
$= variance + bias + E_{x,y}[\epsilon^2]]$
$= variance + bias + \sigma^2$

3: LFD Problem 2.24

(a)

The algorithm will minimize E_{in}. Let's do this analytically. $E_{in} = \sum_{i=1}^{2}[f(x_i) - h(x_i)]^2$
$\sum_{i=1}^{2}[x_i^2 - (ax_i + b)]^2$
Take partial derivatives with respect to both a and b and set to zero to get: $-2\sum_{i=1}^{2} x_i(x_i^2 - ax_i - b) = 0$ and $-2\sum_{i=1}^{2}(x_i^2 - ax_i - b) = 0$
We can multiply the second equation by x_2 to get the second item of the sum in the first Subtracting it from the first we get $x_1^2 - ax_1 - b = 0$. Do the same with x_1 to get $x_2^2 - ax_2 - b = 0$
Solving, we get $a = x_1 + x_2$ and $b = -x_1x_2$, thus the final hypothesis is $(x_1 + x_2)x - x_1x_2$. Since each of x_1 and x_2 have an expected value of 0, the average function is also just 0.

(b)

First, fix an x. Then we sample two datapoints from [-1,1], compute the value of g(x) for this dataset. Repeat this a large number of times to find the value of the average function at this x. To compute E_out, we generate a large number of average function values for different x's. With this group of average function outputs, we can calculate variance, average squared distance from average function to f() (bias), and average squared distance from $g^D()$ to f() (E_out).

(c)

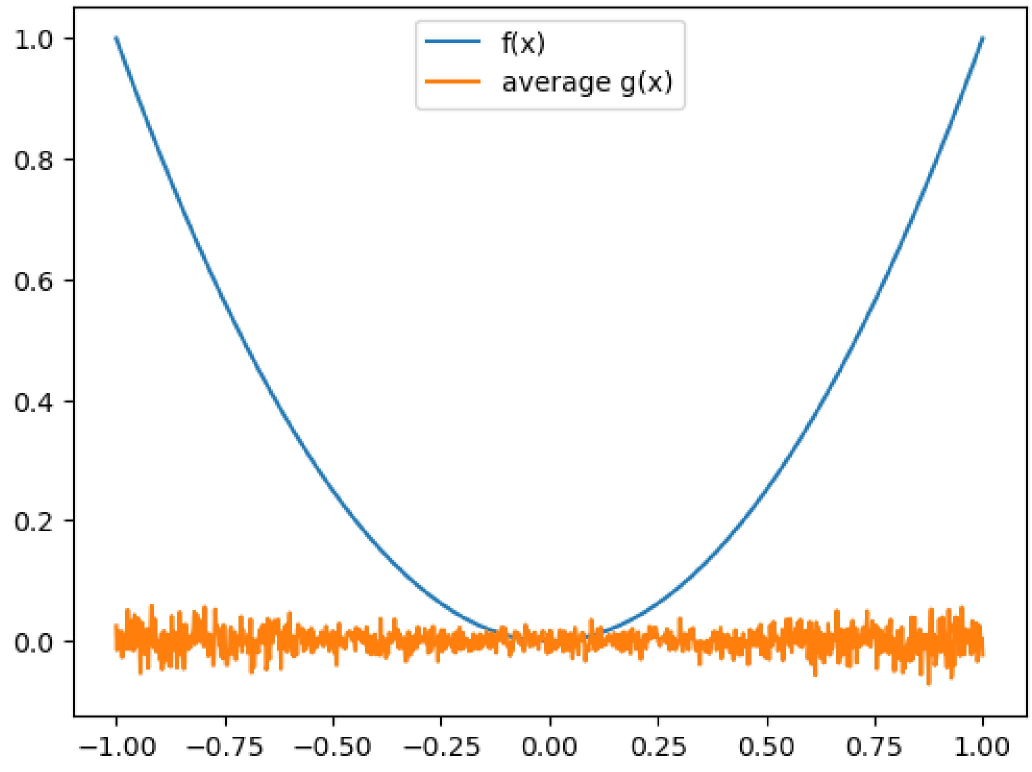

Figure 1: plot

$E_{out} = 0.506116614706004$
$var = 0.321961885870224$
$bias = 0.18370167813452587$

(d)

Since x is uniform on [-1,1], the following is true:
$E[x] = 0$
$E[x^2] = 1/3$
$E[x^3] = 0$
$E[x^4] = 1/5$

We know E_{out} is just variance + bias, so let's calculate those first.
$variance = E_x[E_D[(g^D(x) - \overline{g}(x))^2]]$
$variance = E_x[E_D[((x_1 + x_2)x - x_1x_2 - 0)^2]]$
$variance = E_x[E_D[((x_1 + x_2)^2x^2 + x_1^2x_2^2 - 2x_1x_2(x_1 + x_2)x]]$
$variance = E_x[x^2E_D(x_1^2 + x_2^2 + 2x_1x_2) + E_D(x_1^2x_2^2) - 2xE_D(x_1^2x_2 + x_1x_2^2)]$
plug in the previously listed values to get
$variance = E_x[\frac{2}{3}x^2 + \frac{1}{9}]$
$variance = \frac{2}{3}\frac{1}{3} + \frac{1}{9}$ $variance = \frac{1}{3}$

$bias = E_x[(\overline{g}(x) - f(x))^2]$
$bias = E_x[(0 - x^2)^2]$
$bias = E_x[x^4]$
$bias = \frac{1}{5}$

Finally, we add the two to get $\frac{8}{15}$ for the out of sample error term. These values are pretty close to what we got in our experiment. Hooray!

4: LFD Problem 3.4

(a)

If $y_n w^T x_n > 1$, then $E_n(w) = 0$, which is continuous and differentiable.
If $y_n w^T x_n < 1$, then $E_n(w) = 1 - y_n w^T x_n$, which is a polynomial function which is continuous and differentiable.
If $y_n w^T x_n = 1$, then $E_n(w) = 0$. $E_n(w)$ is therefore continuous.

The gradient is the derivative of $(1 - y_n w^T x_n)^2$ w.r.t. w, which is $-2y_n x_n(1 - y_n w^T x_n)$. That is equal to 0 when $y_n = w^T x_n$, which is when $y_n w^T x_n = 1$. Therefore it is differentiable everywhere.

(b)

If $sign(w^T x_n) \neq y_n$, then $y_n w^T x_n$ is negative and thus $E_n(w)$ returns something greater than 1. If the signs are the same, then $E_n(w)$ returns at least 0, therefore it upper bounds it. Hence, $\frac{1}{N}\sum_{n=1}^{N} E_n(w)$ is an upper bound for $E_{in}(w)$.

(c)

Adaline performs gradient descent because it updates the weight vector based on the gradient of its loss function, which is $E_n(w)$, as shown above. This is an improvement on PLA because it allows for degrees of wrongness and correctness in the weight vector.

5: LFD Problem 3.19

(a)

If I'm understanding the notation correctly, this is binary encoding all variables. This would be an issue because it would balloon the number of variables, each binary entry essential becomes its own variable with its own entry in w. This makes the hypothesis set more complicated and thus the generalization worse.

(b)

This has the same issue as (a), where it expands the number of variables and thus hurts the generalization ability of the model.

(c)

Once again, this massively increases the number of variables by turning every x into a 101x101 grid. This balloons the amount of data needed to achieve good generalization.