CMPE 452 Assignment 2: Backpropagation

Design Decisions

Firstly, the number of input nodes depends on the number of inputs per data-point from the MNIST dataset. Since each input was a 28x28 image, the number of inputs equals the number of pixels in each image, 28*28 or 784. With this design, each image's 28x28 input matrix was flattened into a 784x1 vector, causing a loss of information but simplifying the overall network architecture. In this design, bias was absorbed, which means the input vector had an additional appended value of 1.

Next, the number of output nodes depends on the number of classes, in this case 10, where each class represents one digit in the range 0 to 9. Using one-hot encoded output vectors where 10 different one-hot encodings represent 1 of the 10 possible digits, we remove the possibility of crosstalk and simplify the network architecture.

Determining the number of hidden nodes in the hidden layer requires some empirical analysis (trial-and-error) or some research. Based on my research, a hidden layer with 32 nodes results in almost the same accuracy as a hidden layer with 64 nodes for the MINST dataset. Again, since this design absorbs bias, the output vector of the hidden layer includes an additional appended value of 1.

Since the training and test datasets are large, this design implemented per-batch training, where changes to weight values are summed, and once the program has iterated through a number of samples equal to the batch size, then those weight changes are applied to the weight matrix.

Initial weights for the weight matrix were assigned a random value from -1 to 1 where the mean of the distribution of weights is zero. Stochastic Gradient Descent depends on random values, so the decision to randomly initialize weights between -1 and 1 helps the learning algorithm find weights that more accurately classify data.

The learning rate is set to 0.5, since learning rates should generally fall in the range of 0.1 to 0.9. The network could be improved by adaptively changing the learning rate, but defining and maintaining a constant learning rate maintains the simplicity of the network design

As for the momentum factor, the chosen value was 0.5. The momentum factor must fall between 0 and 1, where a value close to 0 implies that past history does not impact weight change, while a number close to 1 implies the opposite. Since it was not clear, based on empirical analysis, whether or not a value closer to 0 or 1 would be better for network accuracy, the selected momentum factor was 0.5, where past history does impact weight change, but not a lot.

Finally, the design uses epochs and total error as termination criteria. The number of epochs for the first model was set to 3. Since training on 60,000 data points takes a long time, an epoch termination value in the tens or hundreds would result in training times that are too long. The maximum error was set to 1000. This value was determined empirically but is also a somewhat arbitrary decision. The Error value starts out very large and approaches 0. An Error of 1000 was determined to be an acceptable amount of error.

Model 1

Confusion Matrix Training Data

0	0	0	0	0	0	0	0	5923	0	0.0%
0	0	0	0	0	0	0	0	6742	0	0.0%
0	0	0	0	0	0	0	0	5958	0	0.0%
0	0	0	0	0	0	0	0	6131	0	0.0%
0	0	0	0	0	0	0	0	5842	0	0.0%
0	0	0	0	0	0	0	0	5421	0	0.0%
0	0	0	0	0	0	0	0	1918	0	0.0%
0	0	0	0	0	0	0	0	6265	0	0.0%
0	0	0	0	0	0	0	0	5851	0	100.0%
0	0	0	0	0	0	0	0	5949	0	0.0%
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	10.4%	0.0%	10.4%

Confusion Matrix Test Data

0	0	0	0	0	0	0	0	980	0	0.0%
0	0	0	0	0	0	0	0	1135	0	0.0%
0	0	0	0	0	0	0	0	1032	0	0.0%
0	0	0	0	0	0	0	0	1010	0	0.0%
0	0	0	0	0	0	0	0	982	0	0.0%
0	0	0	0	0	0	0	0	892	0	0.0%
0	0	0	0	0	0	0	0	958	0	0.0%
0	0	0	0	0	0	0	0	1028	0	0.0%
0	0	0	0	0	0	0	0	974	0	100.0%
0	0	0	0	0	0	0	0	1009	0	0.0%
0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	9.7%	0.0%	9.7%

Model 2A

Confusion Matrix Training Data

4179	16	0	62	607	67	23	759	112	98	70.6%
137	732	0	8	651	899	494	1634	22	165	15.4%
1589	102	2	53	228	148	235	1987	1166	448	0.0%
2200	58	2	266	595	303	176	799	888	844	4.3%
706	136	0	11	828	214	312	1175	1549	911	14.2%
1446	133	0	184	1327	259	239	925	502	406	4.8%
1878	205	2	4	286	56	818	562	1261	846	13.8%
288	21	0	61	433	226	34	4282	412	508	68.3%
750	102	0	64	615	224	249	1117	2154	576	36.8%
379	51	0	8	867	263	289	1598	1260	1234	20.7%
30.8%	47.0%	33.3%	36.9%	12.9%	9.7%	28.5%	28.9%	23.1%	20.4%	25.4%

Confusion Matrix Test Data

718	2	0	9	89	7	1	124	21	9	73.3%
18	123	0	3	120	153	83	245	358	32	10.8%
274	10	2	18	54	25	43	316	217	73	0.2%
331	11	0	45	92	52	29	151	139	160	4.5%

101	29	0	2	143	26	47	204	267	163	14.6%
278	17	0	26	216	48	35	147	73	52	5.4%
326	21	1	2	42	10	113	99	204	140	11.8%
32	1	0	10	62	34	4	706	75	103	68.7%
128	16	0	13	82	45	40	151	383	116	39.3%
64	8	0	1	141	39	37	249	233	237	23.5%
31.6%	51.7%	66.7%	34.9%	13.7%	10.9%	26.2%	29.5%	19.4%	21.8%	25.2%

Model2B

Confusion Matrix Training Data

5677	0	38	31	41	23	57	14	33	9	95.8%
2	6454	69	24	13	14	22	30	98	16	95.7%
88	74	5015	138	120	4	204	116	179	20	84.2%
33	35	270	5217	34	107	68	156	150	61	85.1%
45	10	33	26	5039	18	92	15	74	490	86.3%
232	33	146	658	274	2959	214	129	629	147	54.6%
147	13	48	4	31	48	5575	0	47	5	94.2%
30	79	70	64	103	5	7	5682	25	200	90.7%
80	133	73	219	181	94	85	52	4804	130	82.1%
66	17	16	105	411	32	33	211	68	4990	83.9%
88.7%	94.2%	86.8%	80.4%	80.7%	89.6%	87.7%	88.7%	78.7%	82.2%	85.7%

Confusion Matrix Training Data

947	0	1	6	2	2	10	2	9	1	96.6%
0	1093	4	6	1	1	7	3	20	0	96.3%
17	12	866	27	14	0	26	24	40	6	83.9%
4	1	36	871	4	17	9	35	25	8	86.2%
9	4	6	6	855	2	19	1	12	68	87.1%
24	1	26	106	44	501	36	32	99	23	56.0%
25	2	12	4	8	5	891	1	9	1	93.0%
5	22	20	17	15	0	3	911	6	29	88.6%
9	8	9	33	24	22	18	18	819	15	85.5%
17	2	1	14	70	9	4	25	15	852	84.4%
89.6%	95.5%	88.3%	80.0%	82.4%	89.6%	87.1%	86.6%	77.8%	85.0%	86.0%

Discussion and Conclusions

Model 1 training results in a weight matrix that always predicts the output of any given input to be the digit 8. This is a result of not properly adjusting the weight matrix values caused by an error in the code.

After comparing the results of models 2A and 2B, the decision to use per-batch training where mini-batch sizes were equal to 250, in combination with a small number of maximum epochs, 3, resulted in very low accuracy. Model 2B uses mini batches of size 32 and 20 epochs, resulting in an accuracy of ~86% in comparison to ~25% of Model 2A. More epochs and smaller batch sizes result in higher accuracy of classification because of a well-trained weight matrix.