Digit Recognizer

By Philip Hasse

Purpose

The purpose of this machine learning algorithm is to be able to determine what number is being depicted from a picture of a handwritten digit.

Data Layout

- Each picture is divided into 784 pixels. Each pixel is given a value from 0-255 to depict how full each pixel is. If a pixel is completely black, it will be given the value of 255 and if it's completely white (empty), it will be given the value of 0.

	XZ	YA	YB	YC	YD	YE	YF	YG	YH	YI	YJ	YK	YL	YM	YN	YO	YP	YQ	YR
1	pixel648	pixel649	pixel650	pixel651	pixel652	pixel653	pixel654	pixel655	pixel656	pixel657	pixel658	pixel659	pixel660	pixel661	pixel662	pixel663	pixel664	pixel665	pixel666
2	0	0	0	0	214	218	95	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	8	76	146	254	255	254	255	146	19	15	0	0	0	0
4	0	0	0	0	0	0	0	0	63	254	254	62	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	186	159	0
6	0	0	0	1	18	129	208	253	253	253	253	159	129	90	4	0	0	0	0
7	0	0	0	0	0	23	64	158	200	174	61	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	152	253	82	0	0	0	0	0	0	0
9	0	0	94	248	209	73	12	0	0	0	0	0	0	42	147	252	136	9	0
10	0	0	0	0	79	243	234	254	253	253	216	117	0	0	0	0	0	0	0
11	0	0	67	50	176	148	78	16	0	12	12	0	0	0	0	0	0	0	0
12	0	7	186	252	227	184	191	252	252	252	252	253	240	50	0	0	0	0	0
13	0	0	0	0	33	218	252	252	192	141	14	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	9	254	87	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	255	255	255	255	255	255	255	255	191	0	0	0
16	0	100	221	252	252	253	127	112	112	112	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	45	253	66	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	33	179	241	253	253	253	253	253	253	246	179	44	0	0
19	0	0	45	233	253	253	232	120	114	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	1	191	226	27	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	25	91	165	252	252	63	7	28	3	0	0
22	0	0	0	207	169	83	174	242	230	80	4	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	62	252	210	128	252	62	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	11	23	22	107	137	137	23	22	22	14	0	0	0	0
26	0	0	0	0	0	0	94	238	238	238	240	112	106	17	0	0	0	0	0
27	145	248	253	212	229	253	214	199	106	42	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

First Hypothesis and Problems

- Use a Multi-layer Perceptron with one hidden layer to iterate through the data using an eta between .1 and 1 and neurons between 1 and 150 which were compared through 10-fold cross-validation.
- The training set supplied by Kaggle was too large so I shank it to 1000 entries
- The algorithm took approximately 9 hours to complete

Conclusions from First Experiment

- The best solution the Multi-Layer Perceptron found was with 122 neurons and an eta of 0.1.
- The cross-validation score came to about -6357.587 and the CV E_{out} came to about .28486
- Although the CV E_{out} is better than randomly guessing, it is still a bad method of guessing the digits
- Poor computation time and poor results lead me to use another algorithm

Second Hypothesis and Problems

- The next algorithm I tried to implement was a K-Nearest-Neighbors algorithm that just used all of the default parameters (n_neighbors = 5).
- The computation time using the whole dataset was a little more than an hour
- To test I shrunk the dataset to only 5000 entries

Conclusions from Second Experiment

- The solution using KNN was much better than the solution given by the Multi-Layer Perceptron, giving me a CV E_{out} of .928305
- This is a great score, however computation time for the entire dataset is a little long

Overall Conclusions and Difficulties

- The KNN neighbors approach yielded a much better solution than the Multi-Layer Perceptron
- One possible explanation for this could have incorrect implementation of the Multi-Layer Perceptron
- Possibly looking into other algorithms to compare against the KNN algorithm is a way to move forward to try and maximize the CV E_{out} and improve computation time