Report

1) Formulation of the problem:

Develop class “EmojifierV1”, which using pretrained 50-dimensional GloVe embeddings of word. “EmojifierV1” must define what emoji can be used with input sentence. Train the “EmojifierV1”, evaluate the importance of learning rate, learning iterations and neural network architecture.

Develop methods to prepare database for “predict” and “fit” and “test” methods.

2) Preprocessing:

Example input data:
[I am always working, 3]

Lowercase, slit and one-hot representation to output:
[i, am, always, working] [0, 0, 0, 1, 0]

GloVe embeddings:
[1.1891e-01 1.5255e-01 ... -2.6671e-01 9.2121e-01]
[0.34664 0.39805 ... 0.34037 1.3588]
[1.5778e-01 2.6380e-01 ... -2.2232e-01 5.2731e-01]
[0.25792 -0.14413 ... -0.50055 0.54358]

Average of sentence:

$$Avg = \frac{word_i}{count(words)}$$

[0.2203125 0.1675675 -0.01825675 -0.5823375 0.5751325 0.46347975
-0.359695 0.36966125 -0.76170225 -0.05831811 0.00868975 0.31552
0.6267425 0.04421435 0.804925 0.398175 0.2166125 0.6795025
0.021271 -0.433325 -0.176527 0.857225 0.555932 0.6414575
1.0257525 -1.83725 -0.6941275 0.138315 0.683832 -0.31363238
3.1559 0.37279275 -0.50122075 -0.1706975 -0.2846885 -0.3585625
0.2655025 0.43667 0.11652 -0.15790025 0.03425143 -0.29827532
0.031 0.48315703 0.0288805 0.07005925 -0.3304275 -0.1512855
0.1623025 0.837725]

3) **Algorithm of "EmojifierV1":**

$$z = \begin{pmatrix} 0.56 \\ 0.17 \\ \dots \\ 0.03 \end{pmatrix} * \begin{pmatrix} w & w & w & \dots & w \\ w & w & w & \dots & w \\ w & w & w & \dots & w \\ w & w & w & \dots & w \\ w & w & w & \dots & w \end{pmatrix} + b$$

Avg [1, 50] W [50, 5]

$$a = \text{softmax}(z)$$

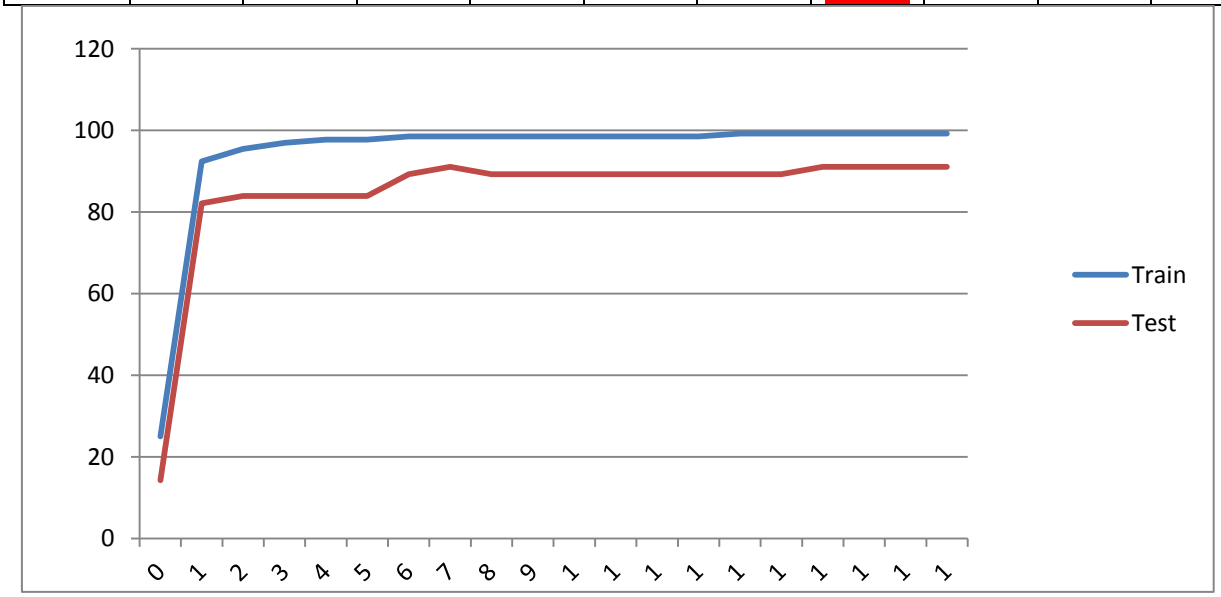
$$a_{max} = 1, a_{else} = 0$$

4) Experiments:

learning_rate = 0.01:

Iteration:	0	100	200	300	400	500	600	700	800
Acc (train):	25.0	92.42	95.45	96.96	97.72	97.72	98.48	98.48	98.48
Acc (test):	14.28	82.14	83.92	83.92	83.92	83.92	89.28	91.07	89.28

900	1000	1100	1200	1300	1400	1500	1600	1700	1800	1900
98.48	98.48	98.48	98.48	98.48	99.24	99.24	99.24	99.24	99.24	99.24
89.28	89.28	89.28	89.28	89.28	89.28	89.28	91.07	91.07	91.07	91.07

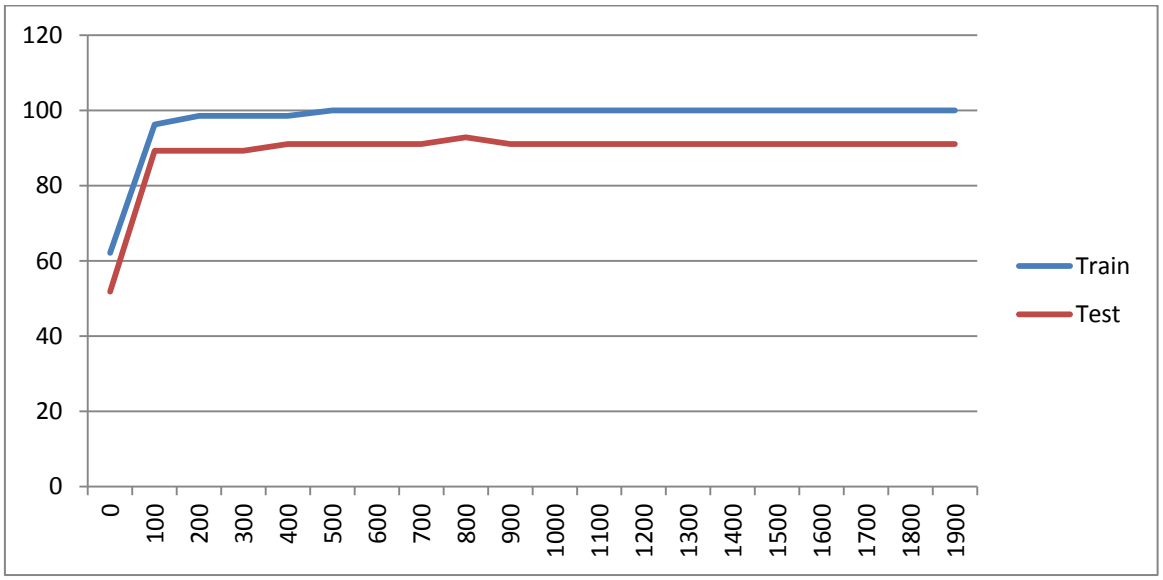


Inference: best results beginning with 700 iteration.
(Why accuracy on test data go down?)

learning_rate = 0.1:

Iteration:	0	100	200	300	400	500	600	700	800
Acc (train):	62.12	96.21	98.48	98.48	98.48	100	100	100	100
Acc (test):	51.78	89.28	89.28	89.28	91.07	91.07	91.07	91.07	92.85

900	1000	1100	1200	1300	1400	1500	1600	1700	1800	1900
100	100	100	100	100	100	100	100	100	100	100
91.07	91.07	91.07	91.07	91.07	91.07	91.07	91.07	91.07	91.07	91.07

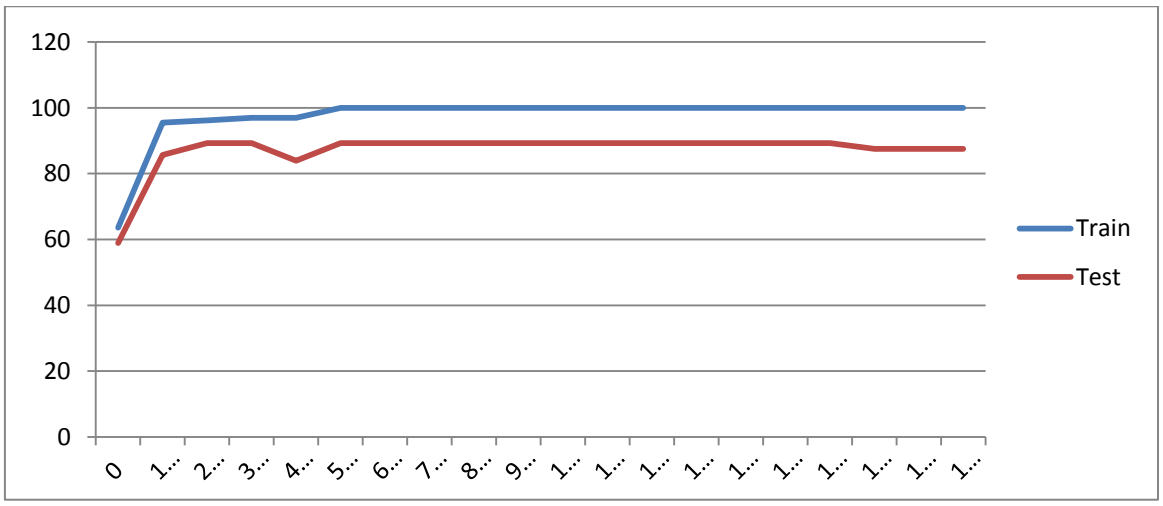


Inference: best results beginning with 800 iteration. It`s best result of all experiments. If model can has accuracy on train data 100%, then we must achieve this result in all next experiments.
(Why accuracy on test data go down?)

learning_rate = 1:

Iteration:	0	100	200	300	400	500	600	700	800
Acc (train):	63.63	95.45	96.21	96.96	96.96	100	100	100	100
Acc (test):	58.92	85.71	89.28	89.28	83.92	89.28	89.28	89.28	89.28

900	1000	1100	1200	1300	1400	1500	1600	1700	1800	1900
100	100	100	100	100	100	100	100	100	100	100
89.28	89.28	89.28	89.28	89.28	89.28	89.28	89.28	87.5	87.5	87.5



Inference: best results beginning with 300 iteration. With big learning rate learning is really fast on train data, but on test data leaves much to be desired.

(Why accuracy on test data go down?)

learning_rate = 0.001:

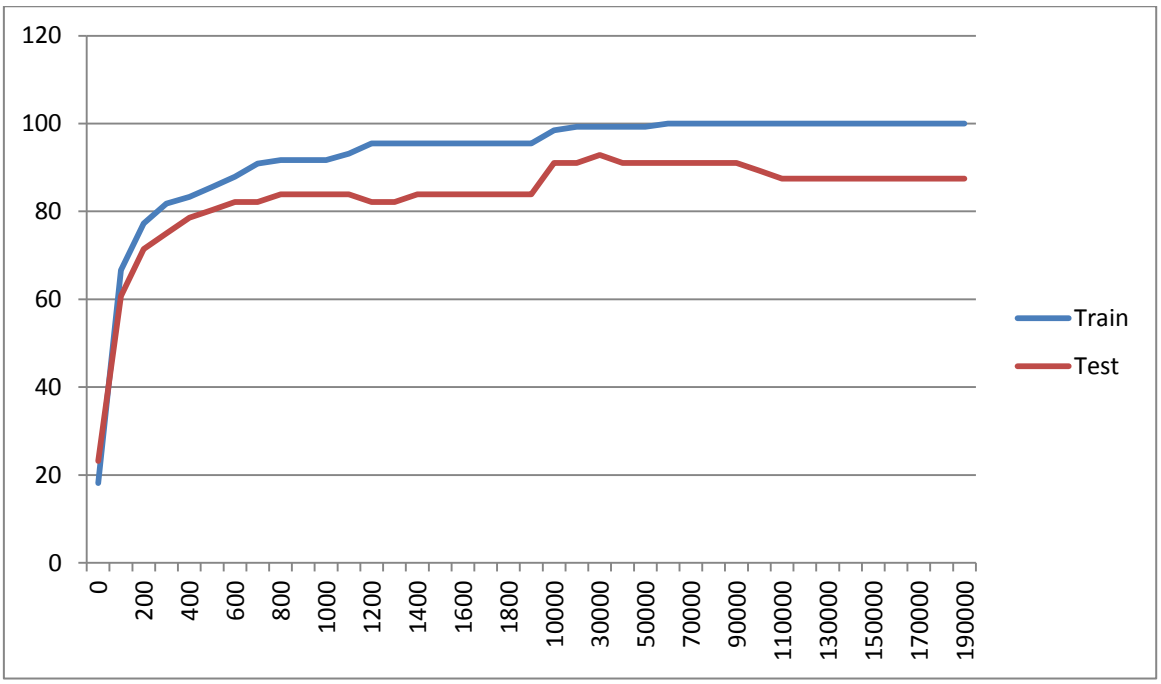
Iteration:	0	100	200	300	400	500	600	700	800
Acc (train):	18.18	66.66	77.27	81.81	83.33	85.6	87.87	90.9	91.66
Acc (test):	23.21	60.71	71.42	75.0	78.57	80.35	82.14	82.14	83.92

900	1000	1100	1200	1300	1400	1500	1600	1700	1800	1900
91.66	91.66	93.18	93.18	95.45	95.45	95.45	95.45	95.45	95.45	95.45
83.92	83.92	83.92	82.14	82.14	83.92	83.92	83.92	83.92	83.92	83.92

10000	20000	30000
98.48	99.24	99.24
91.07	91.07	92.85

40000	50000	60000	70000	80000	90000	100000	110000
99.24	99.24	100	100	100	100	100	100
91.07	91.07	91.07	91.07	91.07	91.07	89.28	87.5

120000	130000	140000	150000	160000	170000	180000	190000
100	100	100	100	100	100	100	100
87.5	87.5	87.5	87.5	87.5	87.5	87.5	87.5



Inference: with learning rate = 0.001 accuracy on train dataset slowly and confidently go up and achieve 100% in 60000 iteration.

Accuracy on test dataset go up and down for unexplained reasons. Best value is 92.85% in 30000 iteration.

5) **Inference:** Best result on train dataset – 100%
(learning_rate = 1, learning_rate = 0.1, learning_rate = 0.001).
Best result on test dataset – 92.85%
(learning_rate = 0.1, learning_rate = 0.001)
Is`t good results on train data only 132 sets.