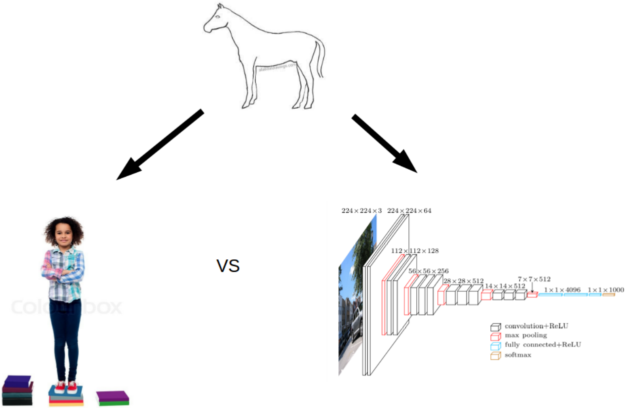
Inverse Classification using Generative Models

Marius Hobbhahn

November 19, 2018

Learning concepts more effectively



VS

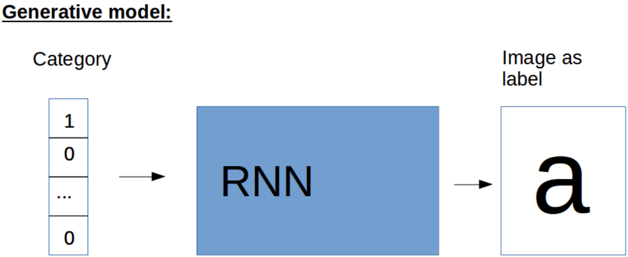
#training samples child << #training samples CNN

Conceptual learning child > Conceptual learning CNN

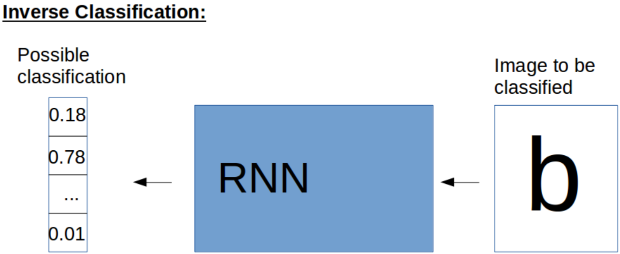
Sizes of popular datasets

- Imagenet: 14,197,122 images
- MNIST: 70,000 images
- COCO: 330,000 images
- Twitter sentiment analysis: 1,578,627 tweets
- $\vdots$

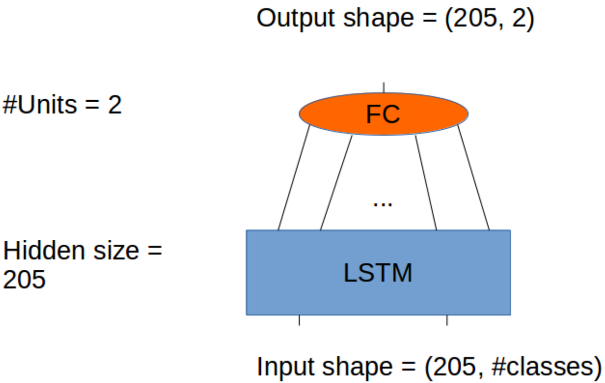
Generative Model



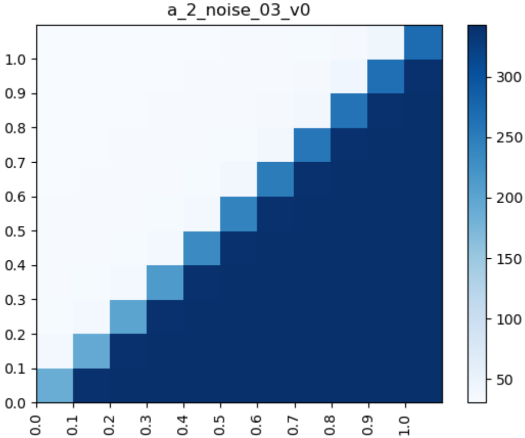
Inverse Classification



Structure


Output shape = (205, 2)
#Units = 2
FC
...
Hidden size = 205
LSTM
Input shape = (205, #classes)

Heat Maps


a_2_noise_03_v0
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
300
250
200
150
100
50

Self-organizing feature Maps (SOMs)

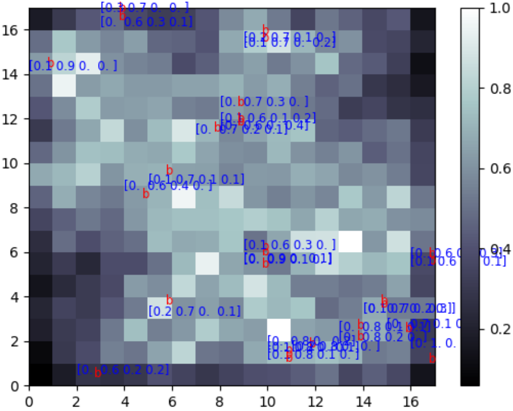


Figure 1: character 'b' and corresponding input vector; color of background denotes average distance to neighboring nodes

Accuracy and RMSE

Table 1: Results for the 2D models without additions. The first five rows represent the accuracy and RMSE for each instance of the model while the last two show mean and standard deviation for each respective category.

model	accuracy	RMSE
2_naked_v0	0.7	0.36
2_naked_v1	0.57	0.44
2_naked_v2	0.81	0.32
2_naked_v3	0.84	0.17
2_naked_v4	0.38	0.6
mean	0.66	0.38
standard deviation	0.17	0.14

Model without additions

- Train a simple generative model
- Investigate latent space
- Test Inverse Classification
- (Additional experiment: add a second stack of LSTMs)

Addition of Noise

- Add a Gaussian noise layer and a clip layer to the model
- Train with standard deviation of 0.1, 0.2, 0.3 and 0.4
- proceed as before
- example: [0.93, 0.2, 0.13, 0.05]

Repeller vector

- Add a new class of uniformly distributed vectors with zero-sequences as targets of size $\frac{1}{\#\text{classes}}$
- proceed as before
- example: [0.25, 0.25, 0.25, 0.25]

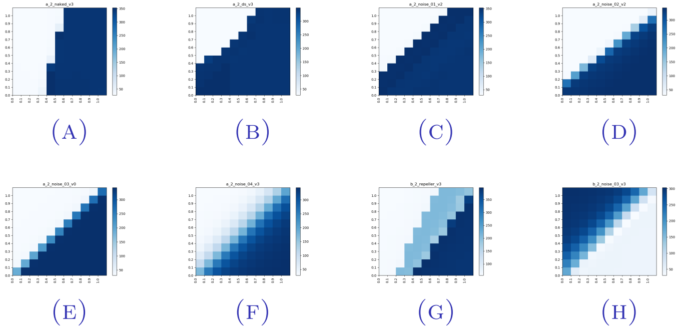
Addition of Types

- Determine type of sequence through k-means clustering
- Give every input its type. It now becomes a 2D one-hot vector
- Test inverse classification and reduce type dimension with sum to become 1D one-hot vectors
- (Additional experiment: test with addition of noise)
- example: [[1,0,0,0], [0,0,0,0], [0,0,0,0], [0,0,0,0]]

Sparse data

- Take only the center of type clusters and train on them
- proceed as before
- (Additional experiment: test with addition of noise)

2D Latent Space (selected)



(A) (B) (C) (D)

(E) (F) (G) (H)

Figure 2: (A) Naked; (B) double-stacked; (C) Noise_01; (D) Noise_02; (E) Noise_03; (F) Noise_04; (G) Repeller; (H) Noise_03_B

2D

model	accuracy
naked_2	0.66
ds_2	0.54
noise_01_2	0.67
noise_02_2	0.98
noise_03_2	1.0
noise_04_2	1.0
repeller_2	0.79
types_2	0.79
types_noise_04_2	0.85
sparse_noise_04_2	0.97
comparison_mse_2	1.0
comparison_sparse_2	0.98
random_values	0.5

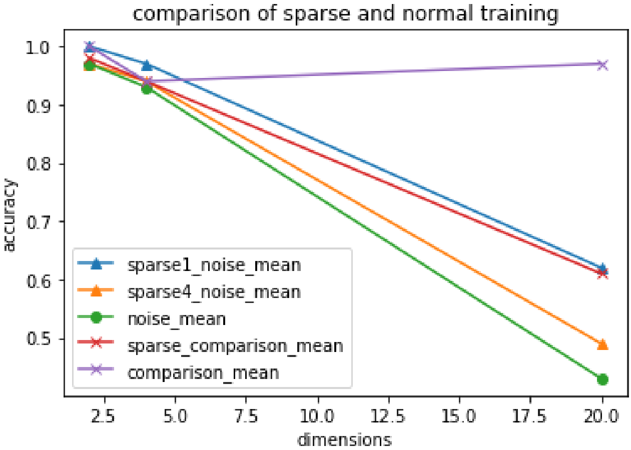
4D

model	accuracy
naked_4	0.49
ds_4	0.65
noise_01_4	0.88
noise_02_4	0.91
noise_03_4	0.93
noise_04_4	0.93
repeller_4	0.58
types_4	0.27
types_noise_04_4	0.79
sparse_noise_04_4	0.97
comparison_mse_4	0.94
comparison_sparse_4	0.94
random_values	0.25

20D

model	accuracy
naked_20	0.35
ds_20	0.32
noise_01_20	0.35
noise_02_20	0.38
noise_03_20	0.43
noise_04_20	0.2
repeller_20	0.29
types_20	0.08
types_noise_03_20	0.09
sparse_noise_03_20	0.49
comparison_mse_20	0.97
comparison_sparse_20	0.61
random_values	0.05

Results as plot



Conclusions

Conclusion 1

The Inverse Classification is as good as the conceptual representation of the latent space.

Conclusion 2

The Inverse Classification is (at least for this task) invariant to the number of training data.

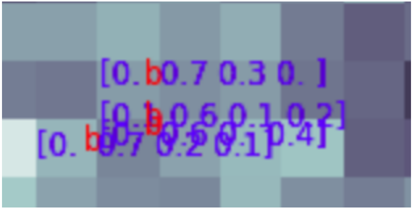
Sparse_1_noise_04

Table 2: Sparse_1_noise_04 average accuracy and RMSE

model	accuracy	RMSE
2_sparse_1_noise_04	1.0	0.13
4_sparse_1_noise_04	0.97	0.18
20_sparse_1_noise_03_v2	0.62	0.18

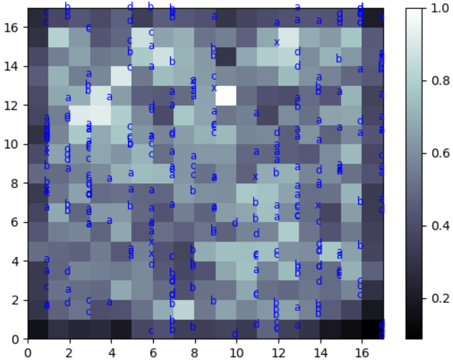
SOM trends 1

- Vectors with similar properties tend to cluster at neurons with small distances.



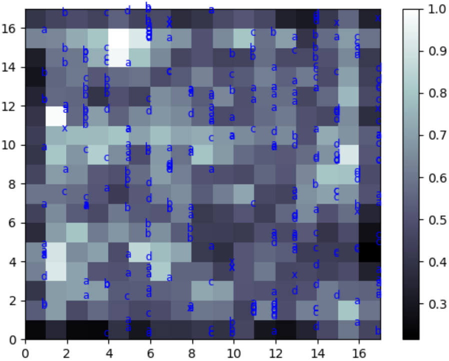
SOM trends 2

- Borders between classes are visible



SOM trends 3

- Addition of noise: borders become more fuzzy but do not overlap


0
2
4
6
8
10
12
14
16
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3