RBM

What is the effect of different parameters on training the RBM, evaluate the performance visually by reconstructing unseen test images.

```
rbm1 = BernoulliRBM(n_components=10, learning_rate=0.01, random_state=0, n_iter=10, verbose=True)
rbm1.fit(X_train)
rbm8 = BernoulliRBM(n_components=20, learning_rate=0.06, random_state=0, n_iter=20, verbose=True)
rbm8.fit(X_train)
```

I chose these values for these **parameters**:
n_componenrs —-> 10,20;
learning_rate —-> 0.01,0.06;
n_iter —-> 10,20

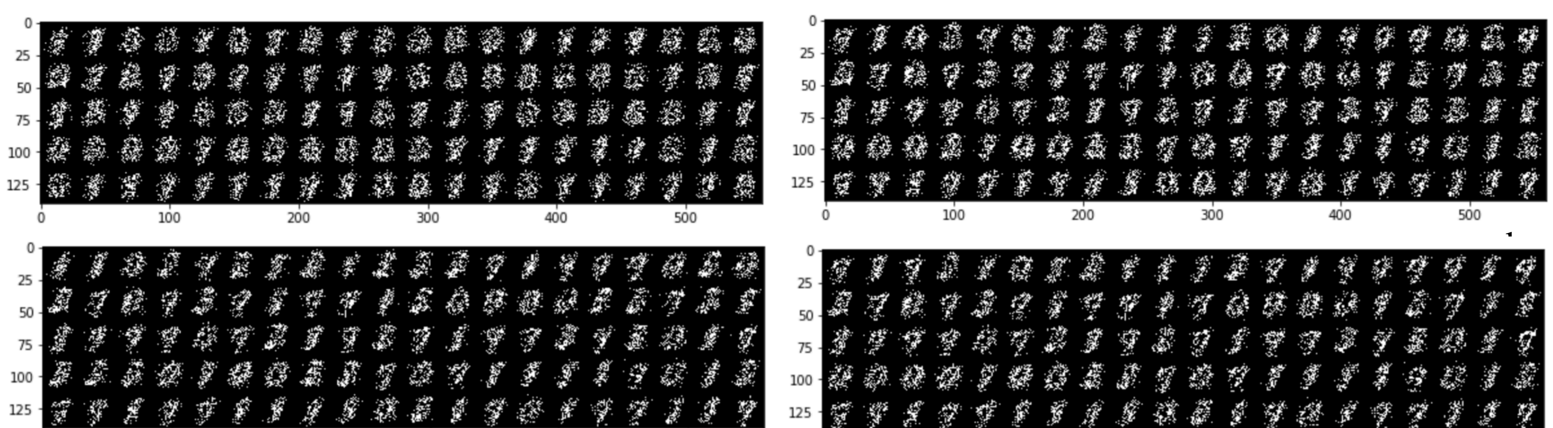
we evaluate probabilities with the **pseudo-likelihood** (since conditional probabilities can be computed without knowledge of the partition function)

the performance of the training, evaluated with the pseudo-likelihood metric, is negatively correlated to the **time** of the training.

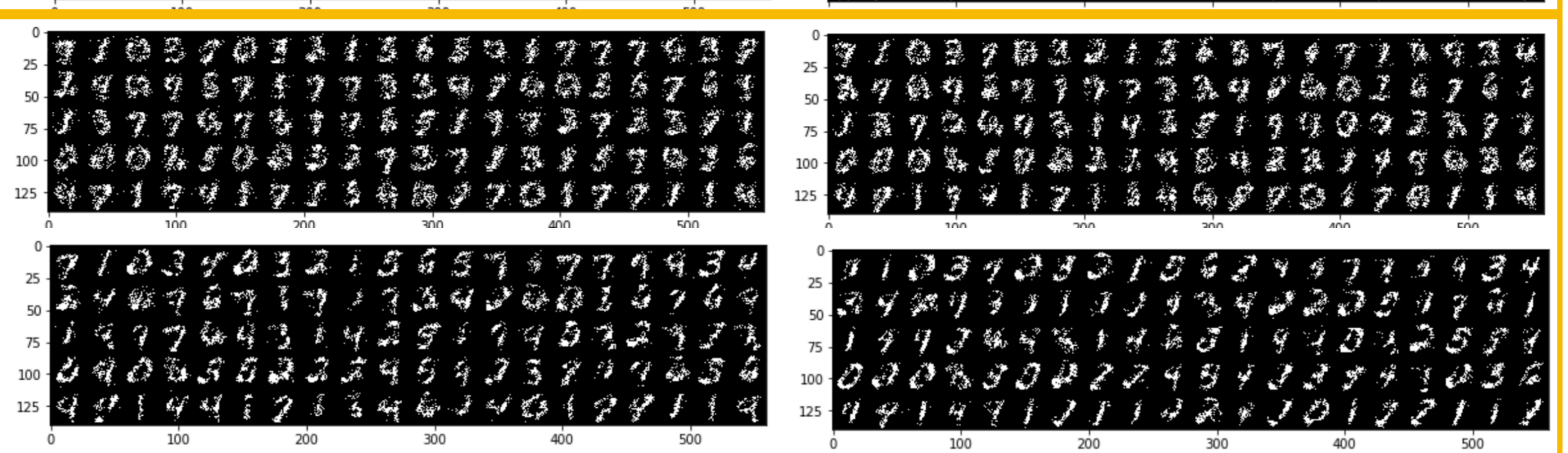
the **best training model** (which may overfits) is, in descending order:
rbm pseudolikelihood:
7, n_components: 20; learning_rate: 0.06; n_iter: 10; pseudo-likelihood = -128.93, time = 3.85s
8, n_components: 20; learning_rate: 0.06; n_iter: 20; pseudo-likelihood = -130.42, time = 3.74s
6, n_components: 20; learning_rate: 0.01; n_iter: 20; pseudo-likelihood = -137.32, time = 3.74s
5, n_components: 20; learning_rate: 0.01; n_iter: 10; pseudo-likelihood = -146.39, time = 3.75s
2, n_components: 10; learning_rate: 0.01; n_iter: 20; pseudo-likelihood = -178.26, time = 3.40s
4, n_components: 10; learning_rate: 0.06; n_iter: 20; pseudo-likelihood = -186.34, time = 3.27s
1, n_components: 10; learning_rate: 0.01; n_iter: 10; pseudo-likelihood = -186.68, time = 3.29s
3, n_components: 10; learning_rate: 0.06; n_iter: 10; pseudo-likelihood = -187.26, time = 3.26s

n_components: higher values leads to more defined results
learning_rate —-> higher values leads to more defined results
n_iter —-> higher values leads to more defined results

rbm1 10, 0.01,10;
rbm2 10, 0.01,20;
rbm3 10, 0.06,10;
rbm4 10, 0.06,20;

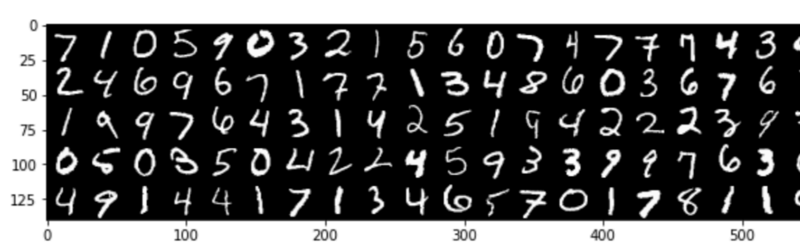


rbm5 20, 0.01,10;
rbm6 20, 0.01,20;
rbm7 20, 0.06,10;
rbm8 20, 0.06,20;



X_train

X_test



RBM

Change the number of Gibbs sampling steps, can you explain the result?

I tried values 10, 100, 1000 for the Gibbs steps,
using my best and worst scoring model in the training step, ie respectively:
rbm 7, n_components: 20; learning_rate: 0.06; n_iter: 10;
rbm 3, n_components: 10; learning_rate: 0.06; n_iter: 10;

Notes: both for best and worst training models, a lower number of Gibbs sampling steps leads to better results.

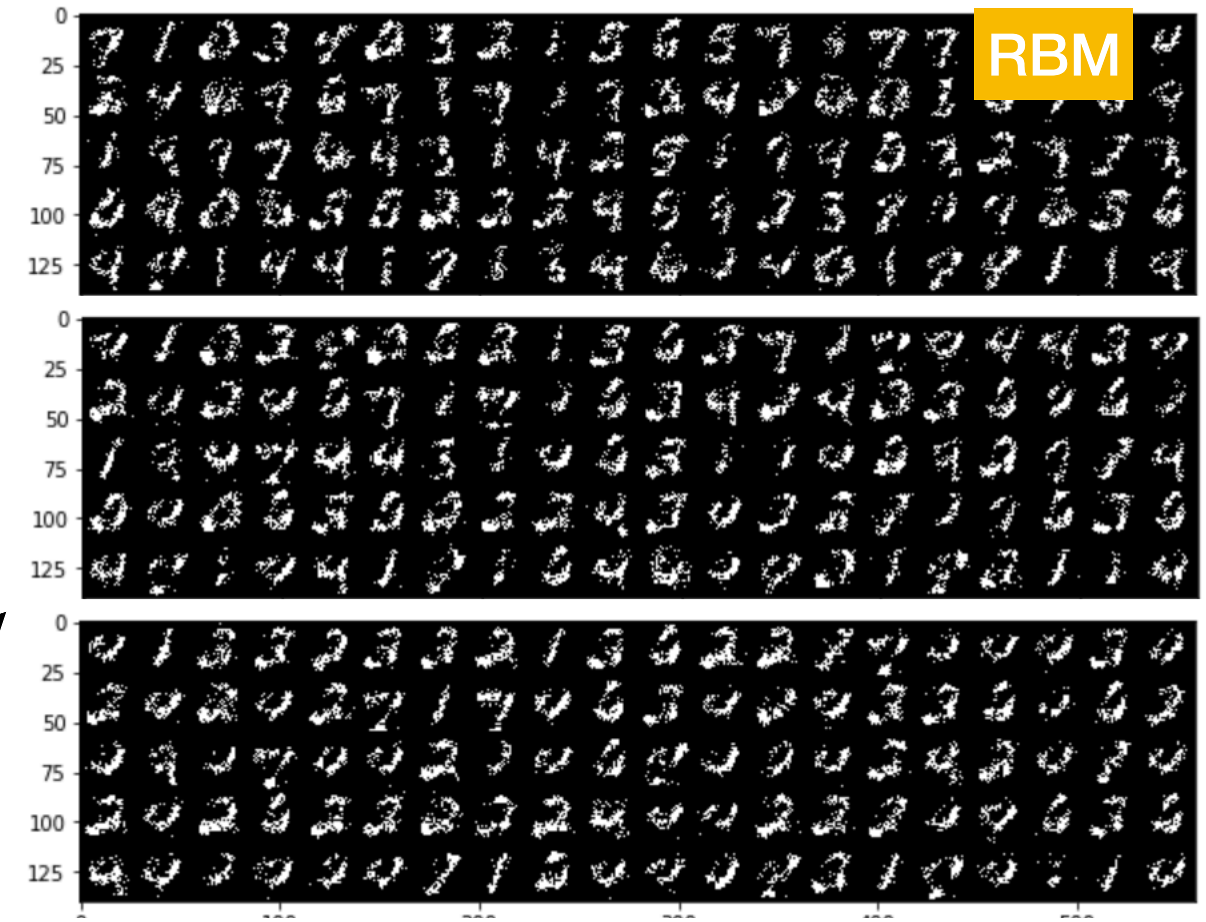


Fig 1: rbm7, Gibbs 10,100,1000 starting from the top.

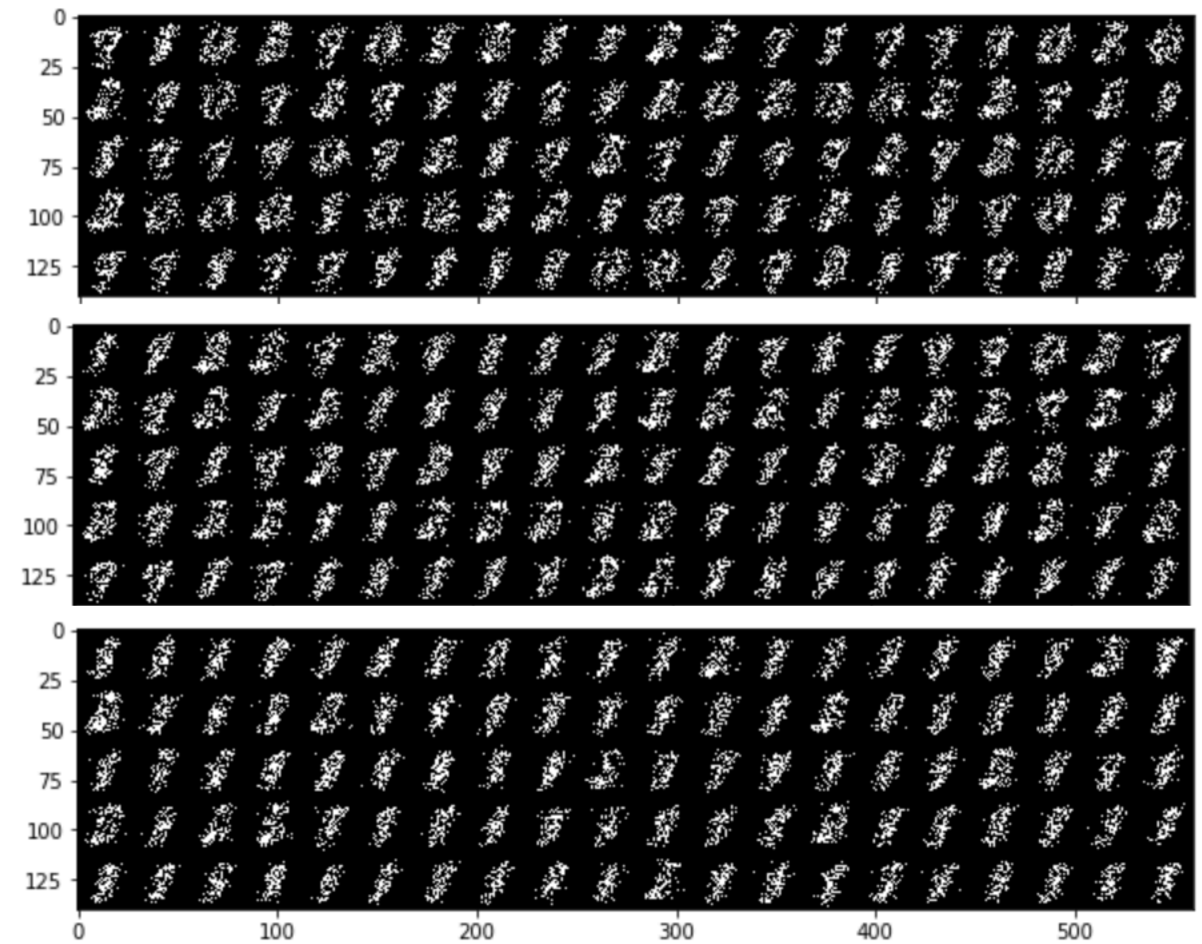


Fig 2: rbm3, Gibbs 10,100,1000 starting from the top.

Reconstruct 10 images from the test set with different numbers of Gibbs steps

RBM

Use the RBM to reconstruct missing parts of images[1].
1) What is the role of the number of hidden units, learning rate and number of iterations on the performance.?
2) How many rows can you remove such that reconstruction is still possible?

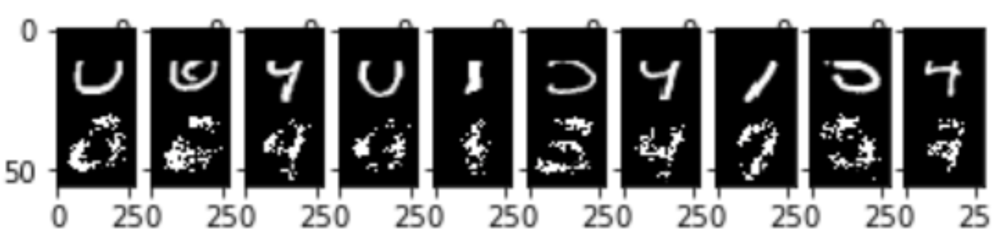
X_test

1) the learning rate didn't affect much my training, while a higher number of iterations and hidden units led to better performances without overfitting.

2) It's better to use a lower number of gibbs steps to reconstruct the image (here no more than 100), both for the best and worst trained RBM models;
results are pretty much equal

reconstruction_gibbs_steps = 10
start_test_index = 10
nr = 10
start_row_to_remove = 2
end_row_to_remove = 12

7, n_components: 20; learning_rate: 0.06; n_iter: 10;



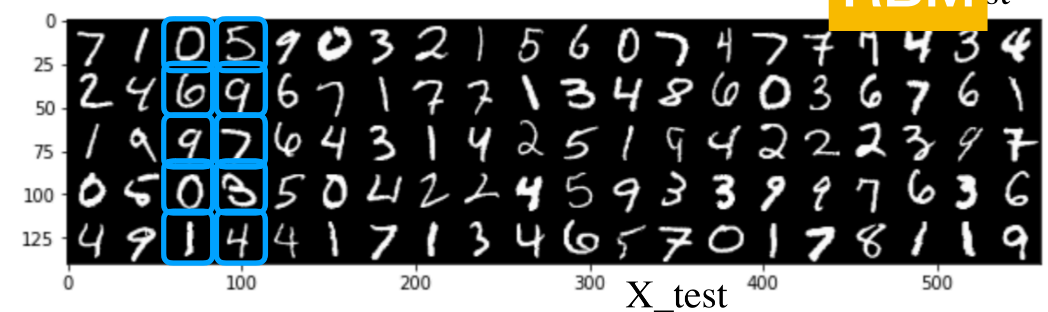
3, n_components: 10; learning_rate: 0.06; n_iter: 10;

reconstruction_gibbs_steps = 100
start_test_index = 10
nr = 10
start_row_to_remove = 2
end_row_to_remove = 12

Reconstruct 10 images from the test set with best numbers of Gibbs steps and different numbers of missing pixels.

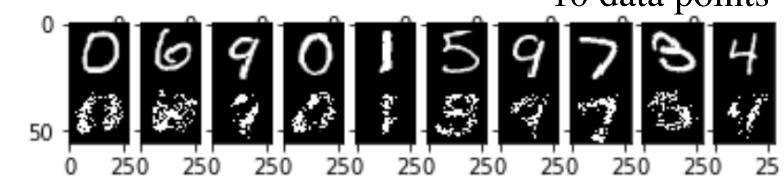
Notes:
rbm3 generalizes better when many test rows are missing, rbm7 overfits therefore only gets to reconstruct unseen images if the number of pixel rows removed is very little.

Using rbm3, I get to reconstruct removing 20 rows, with rbm7 only 1 becase then it starts making wrong guesses and predicts the wrong number.



X_test

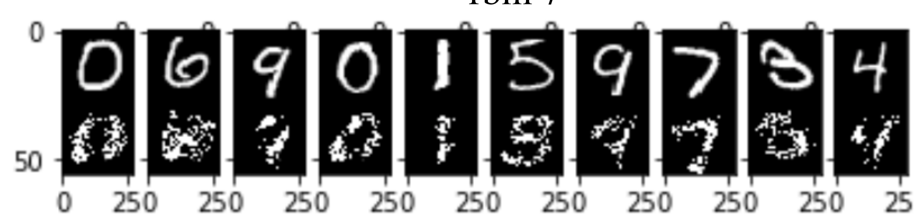
10 data points



rbm 7

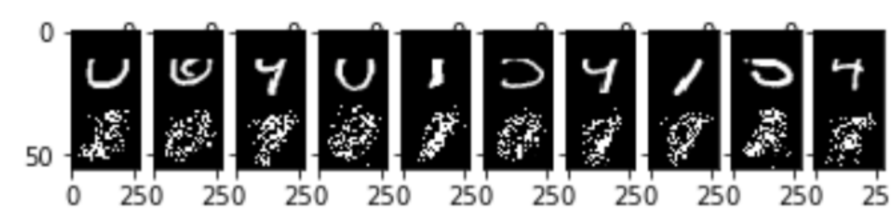
rbm 3

reconstruction_gibbs_steps = 10
start_test_index = 10
nr = 10
start_row_to_remove = 0
end_row_to_remove = 0

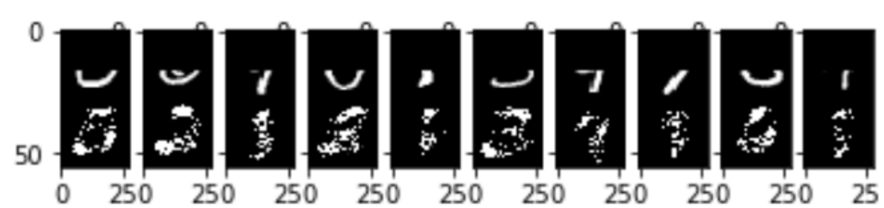


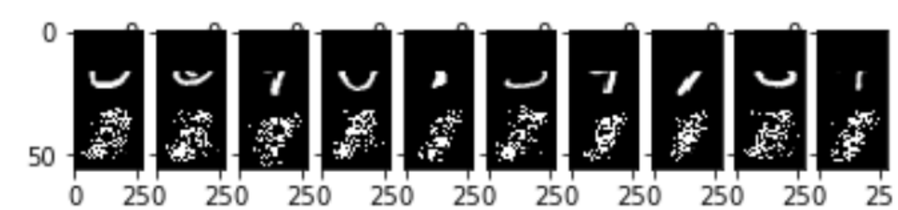
reconstruction_gibbs_steps = 10
start_test_index = 10
nr = 10
start_row_to_remove = 2
end_row_to_remove = 3

reconstruction_gibbs_steps = 10
start_test_index = 10
nr = 10
start_row_to_remove = 2
end_row_to_remove = 12



reconstruction_gibbs_steps = 10
start_test_index = 10
nr = 10
start_row_to_remove = 2
end_row_to_remove = 17





reconstruction_gibbs_steps = 10
start_test_index = 10
nr = 10
start_row_to_remove = 2
end_row_to_remove = 22