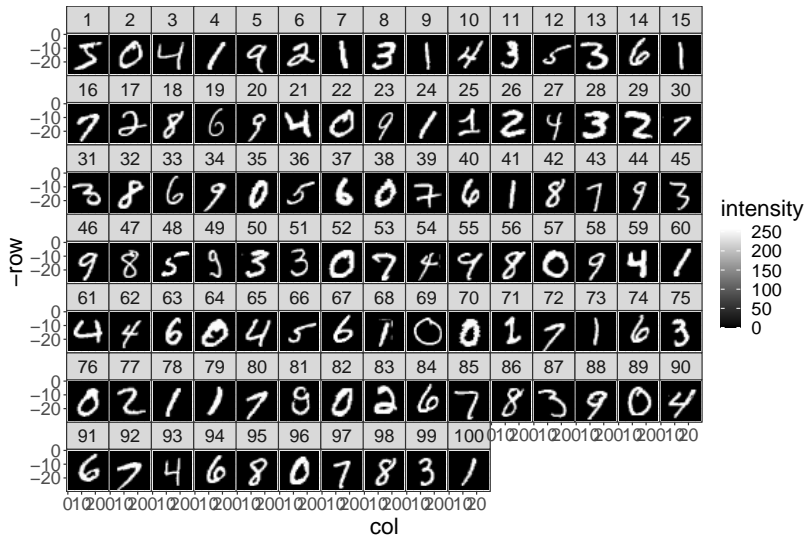


Auto-encoders

Toby Dylan Hocking

Motivation: MNIST digits data



Set of digits is represented as a matrix

- ▶ Each digit image in MNIST data set is a matrix of 28×28 pixel intensity values, $x_i \in \{0, \dots, 255\}^{784}$.
- ▶ Each of the images is a row in the data matrix.
- ▶ Each of the columns is a pixel.
- ▶ All images on last slide represented by a data matrix with $n = 100$ rows/images and $p = 784$ columns/pixels.

Background/motivation: non-linear dimensionality reduction

- ▶ High dimensional data are difficult to visualize.
- ▶ For example each observation/example in the MNIST data is of dimension $28 \times 28 = 784$ pixels.
- ▶ We would like to map each observation into a lower-dimensional space for visualization / understanding patterns in the data.
- ▶ Principal Components Analysis (PCA) is a linear dimensionality reduction algorithm.
- ▶ Auto-encoders are non-linear, which means they can be more accurate than PCA, in terms of reconstruction error.

Auto-encoders are a type of neural network

- ▶ A neural network with L layers is a function $f(x) = f_{L-1}[\dots f_1(x)]$.
- ▶ Each function $f_l(z) = \sigma_l(W_l z)$ consists of multiplication by a matrix W_l followed by an activation function σ_l .
- ▶ The number of layers L , the sizes of the weight matrices W_l , and the activation functions σ_l are all hyper-parameters that must be chosen prior to learning.
- ▶ Number of units/features in each layer determines weight matrix sizes.
- ▶ Auto-encoders have a middle “code” layer which is the low dimensional embedding.

Auto-encoder learning algorithm

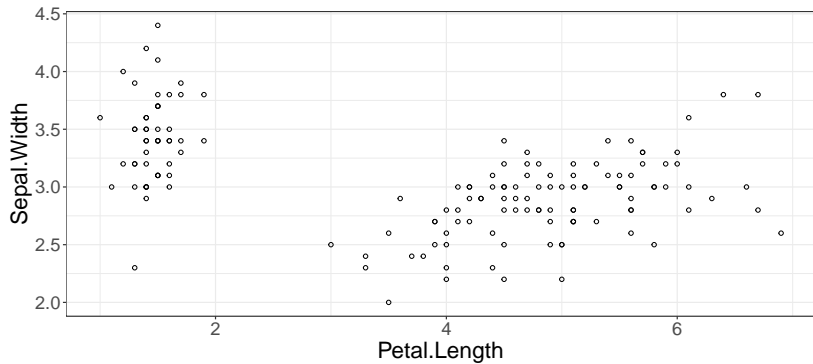
- ▶ The goal of learning is to find a low dimensional mapping of the data which is able to reconstruct the original data.
- ▶ The values in the weight matrices W_l are the model parameters which are learned using gradient descent.
- ▶ The gradient descent learning algorithm starts with arbitrary/random weight matrices W_l close to zero.
- ▶ Each iteration of gradient descent updates the weight matrices W_l in order to get better predictions (lower mean squared reconstruction error).
- ▶ The batch size is the number of observations for which the gradient is computed and summed during each iteration.
- ▶ An “epoch” involves one or more gradient descent iterations (computes gradient with respect to each observation once).

Example: 2d iris data

- ▶ Simple example: iris.
- ▶ One row for each flower (only 6 of 150 shown below).
- ▶ One column for each measurement/dimension.

##	Sepal.Width	Petal.Length
## 1	3.5	1.4
## 2	3.0	1.4
## 3	3.2	1.3
## 4	3.1	1.5
## 5	3.6	1.4
## 6	3.9	1.7

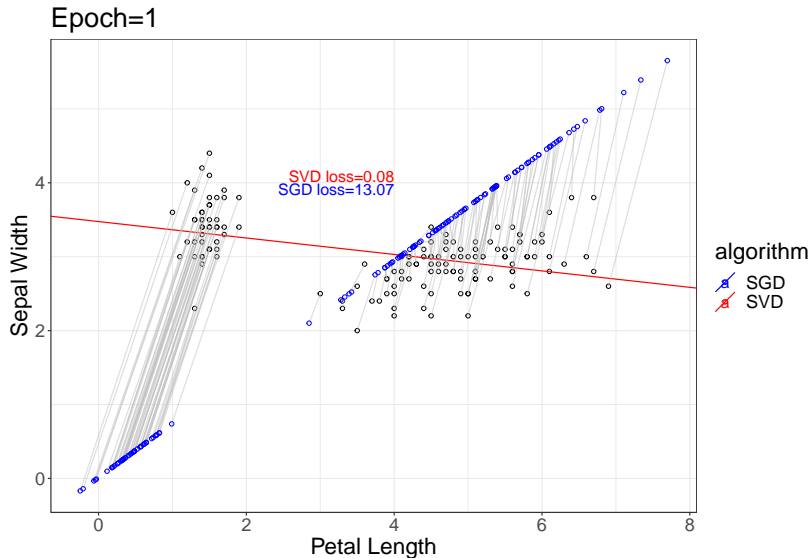
Example: 2d iris data



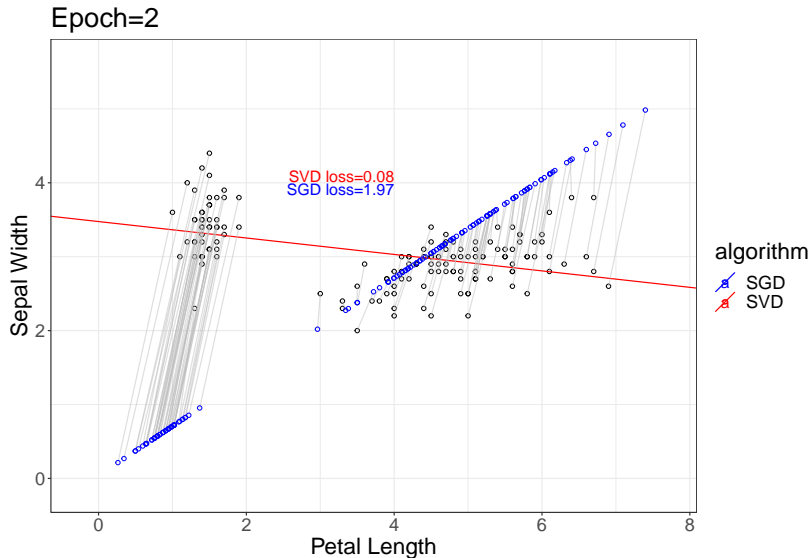
Auto-encoder neural network architecture

- ▶ In this example the number of units in each layer is (2, 1, 2).
- ▶ Input/output layers have two units.
- ▶ Code layer has one unit.
- ▶ Linear activation function, so same model as PCA:
low-dimensional embedding is a linear combination of input features.
- ▶ Learning algorithm iteratively searches for best linear model.

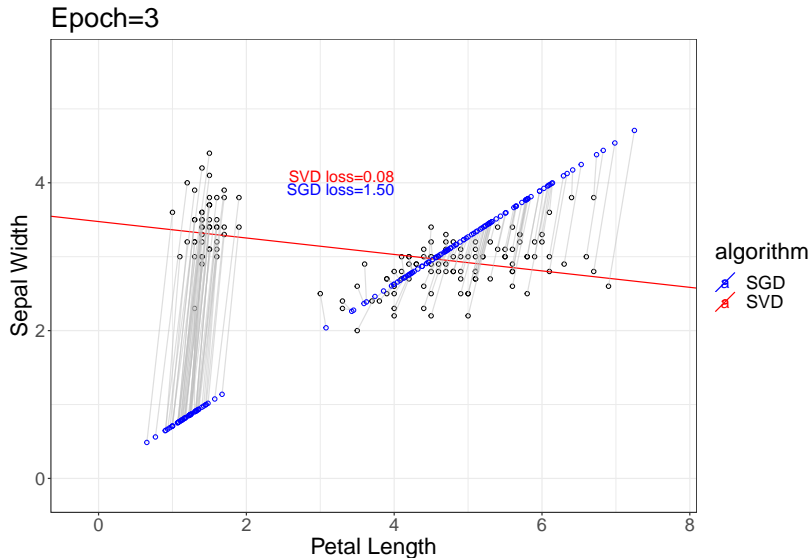
Visualization of predicted values



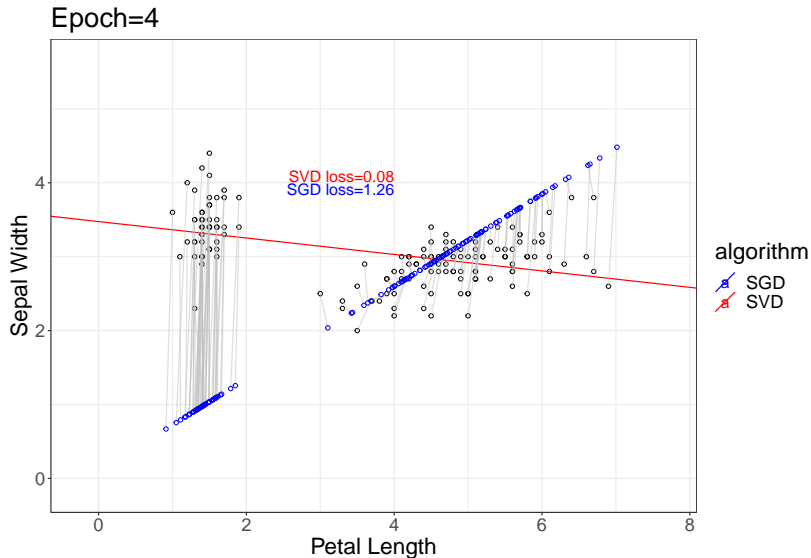
Visualization of predicted values



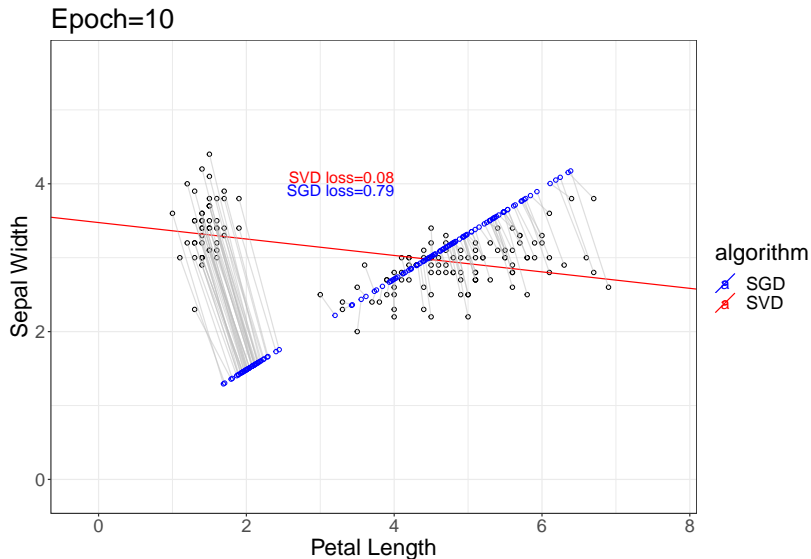
Visualization of predicted values



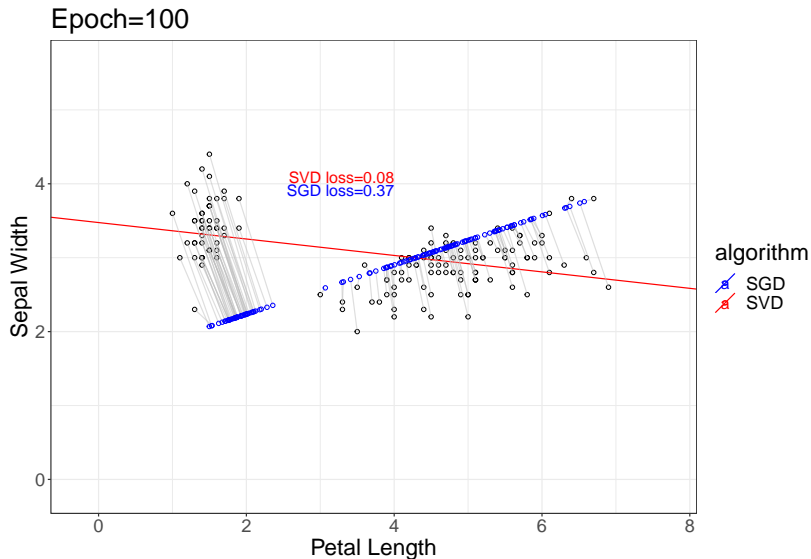
Visualization of predicted values



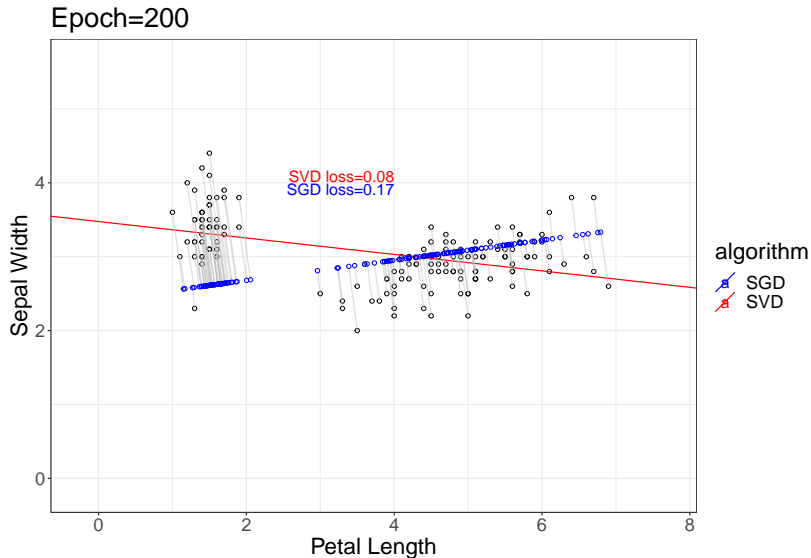
Visualization of predicted values



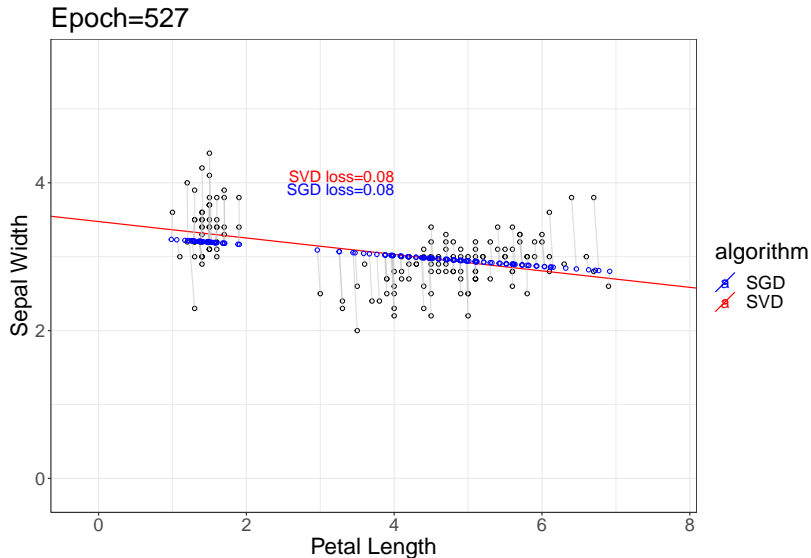
Visualization of predicted values



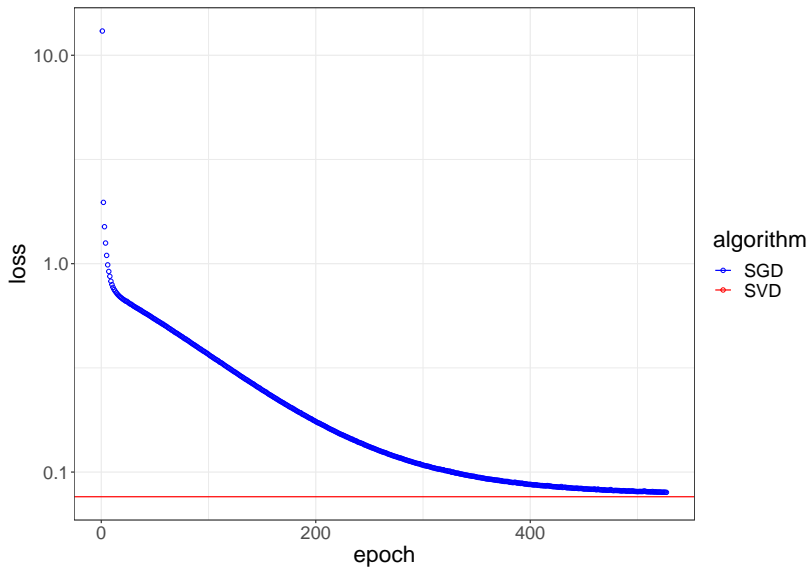
Visualization of predicted values



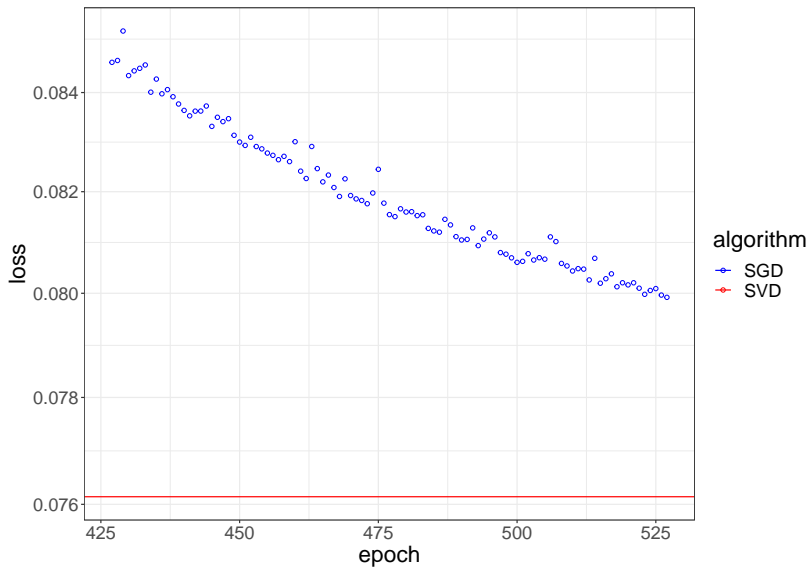
Visualization of predicted values



Loss decreases with number of epochs

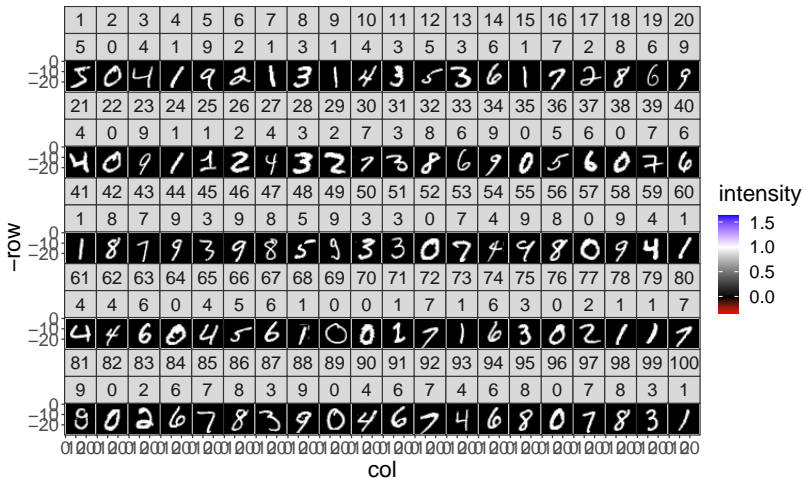


Zoom to last 100 epochs



Actual image data

Scaled images



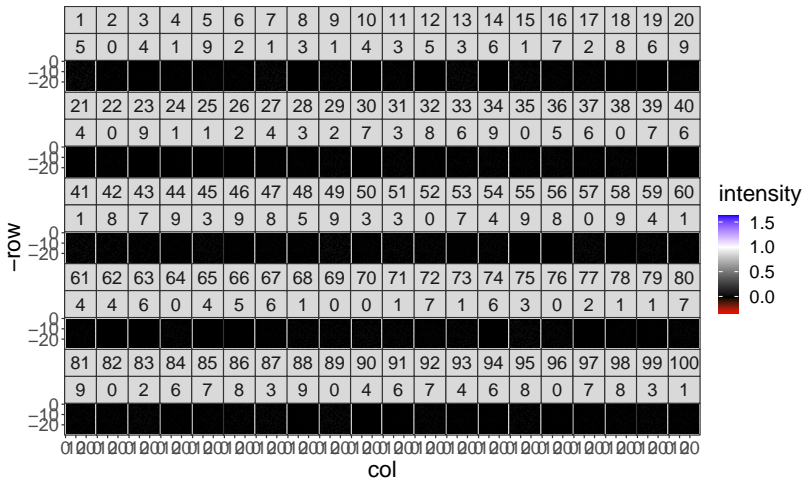
Auto-encoder for image data

- ▶ Each image is represented by a vector of 784 pixel intensity values, so this is the number of units in the first/last layer.
- ▶ The code layer will have 2 units for visualization purposes (two axes on a scatterplot).
- ▶ There is a choice of the number of intermediate layers; here we choose one layer with 100 units (on each side of the code layer).
- ▶ Overall model architecture, in terms of number of units/features per layer, is (784,100,2,100,784).
- ▶ To compute low-dimensional embedding for an image x we compute:

$$f_2[f_1(x)] = \sigma_2[W_2\sigma_1(W_1x)].$$

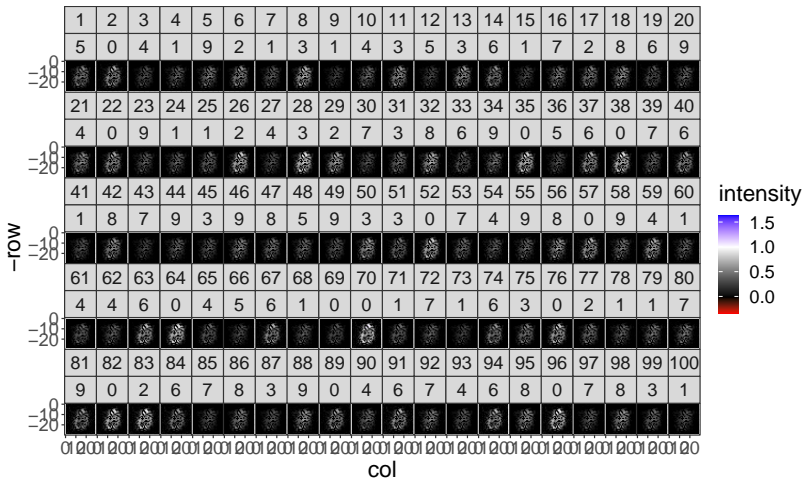
Reconstruction improves with epochs of learning

epoch=1 MSE=0.1072



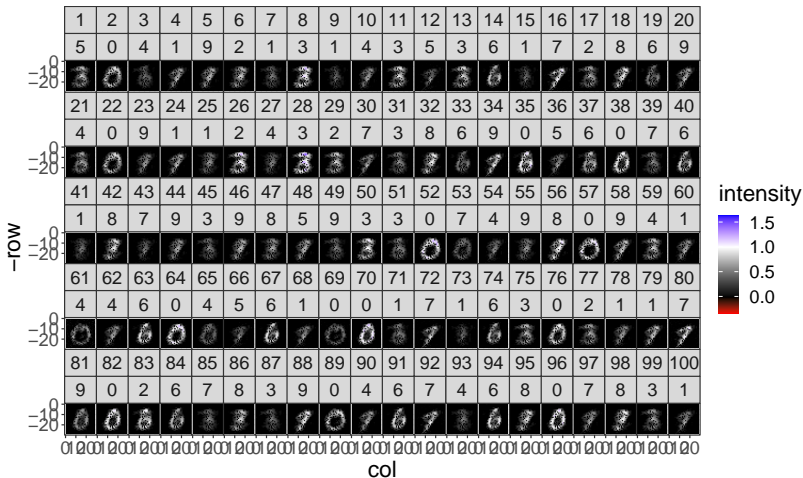
Reconstruction improves with epochs of learning

epoch=101 MSE=0.0776



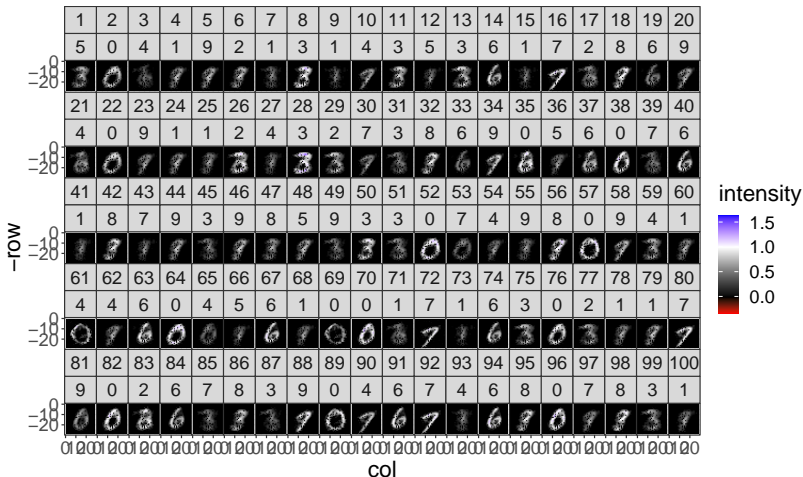
Reconstruction improves with epochs of learning

epoch=1001 MSE=0.0584



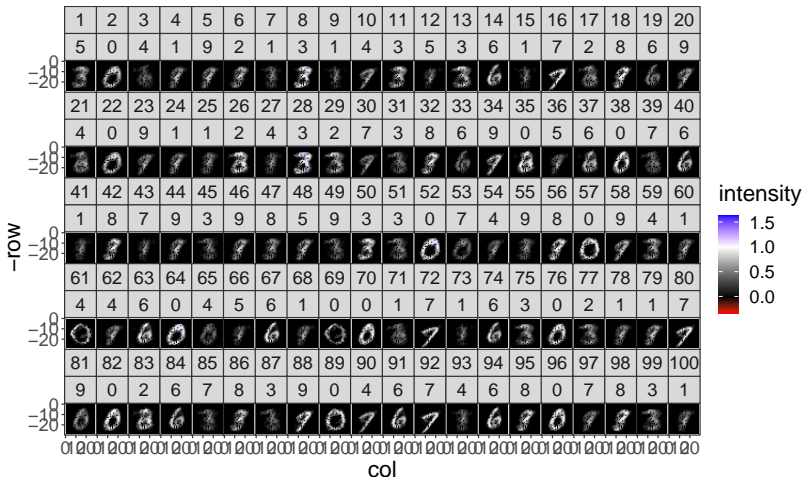
Reconstruction improves with epochs of learning

epoch=2001 MSE=0.0502



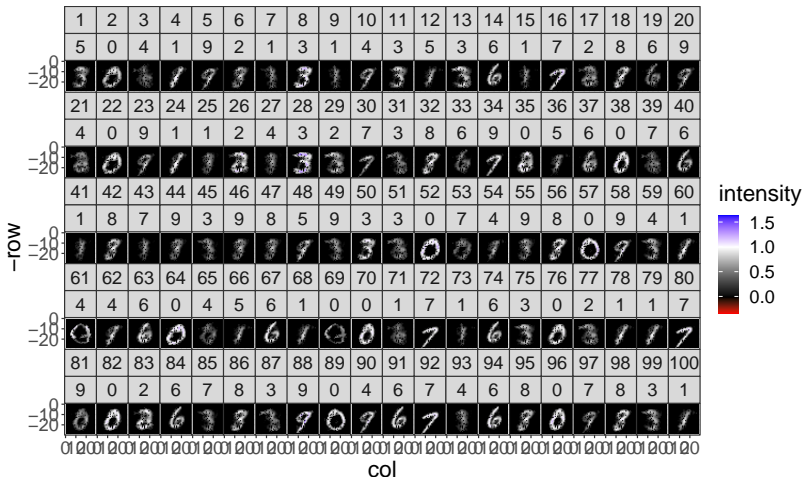
Reconstruction improves with epochs of learning

epoch=3001 MSE=0.0496



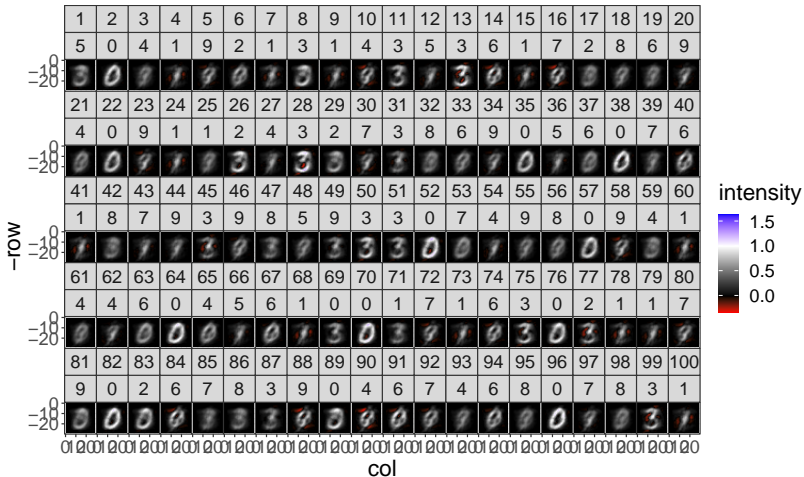
Reconstruction improves with epochs of learning

epoch=4001 MSE=0.0450

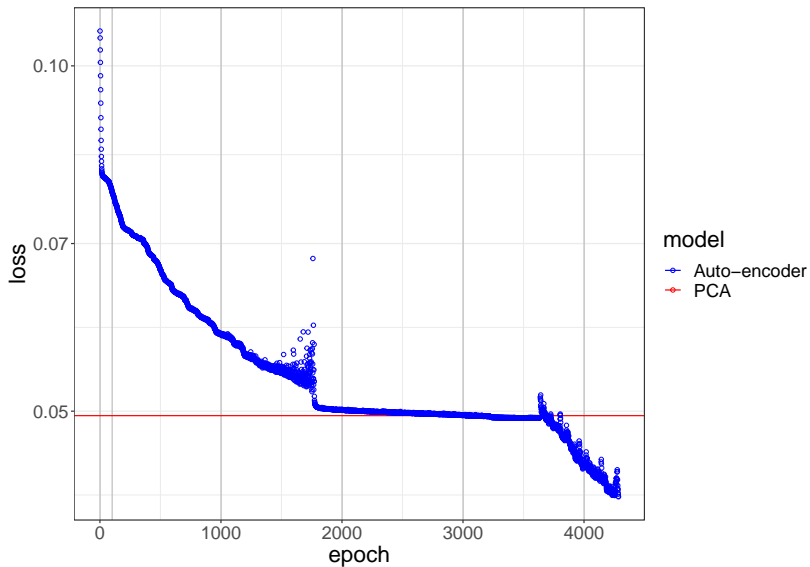


Reconstruction of PCA

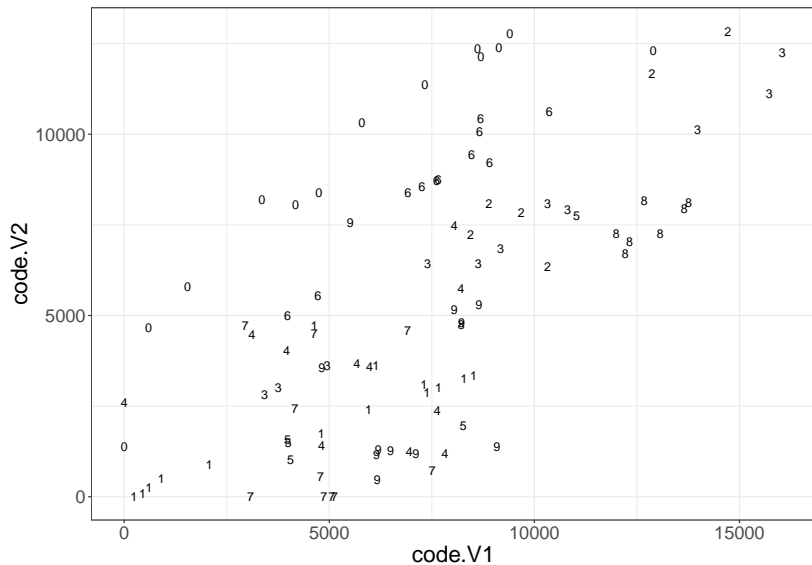
PCA MSE=0.0495510975282269



Loss versus number of epochs



Plot code layer variables instead of PCs



Possible exam questions

- ▶ What choices do you need to make in the auto-encoder in order to have the result be the same as PCA?