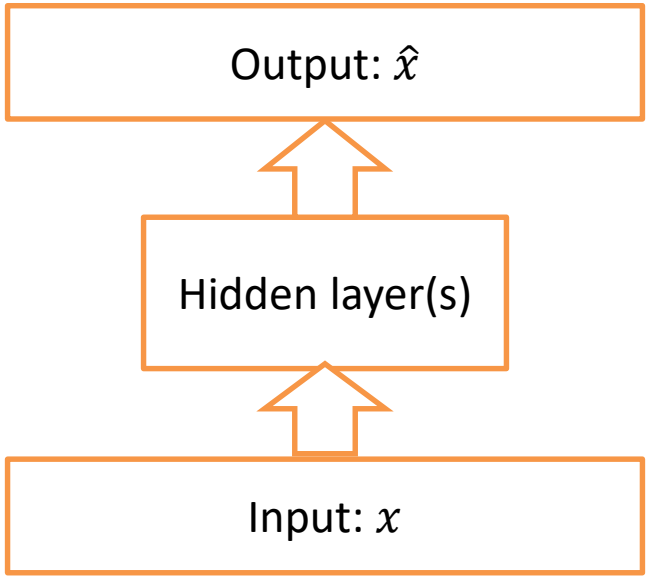
Lecture 8: Autoencoder

Topics of this lecture

- What is an autoencoder?
- Training of autoencoder
- Autoencoder for “internal representation”
- Training with l_2-norm regularization
- Training with sparsity regularization
- Training of deep network

What is an autoencoder? (1/2)

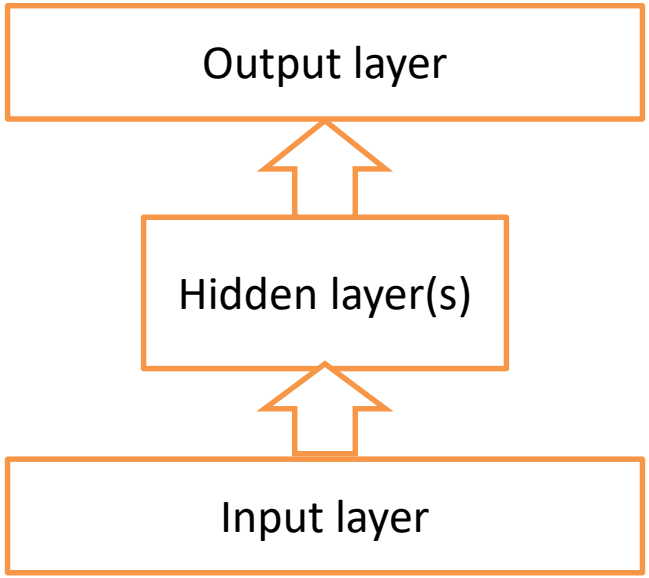
- autoencoder is a special MLP.
- There is one input layer, one output layer, and one or more hidden layers.
- The teacher signal equals to the input.



The main purpose of using an autoencoder is to find a new (internal or latent) representation for the given feature space, with the hope of obtaining the true factors that control the distribution of the input data.

What is an autoencoder? (2/2)

- Using principal component analysis (PCA) we can obtain a linear autoencoder.
- Each hidden unit corresponds to an eigenvector of the co-variance matrix, and each datum is a linear combination of these eigenvectors.



Using a non-linear activation function in the hidden layer, we can obtain a non-linear autoencoder. That is, each datum can be represented using a set of non-linear basis functions.

Training of autoencoder (1/4)

- Implementing an autoencoder using an MLP, we can find a more compact representation for the given problem space.
- Compared with classification MLP, an autoencoder can be trained with un-labelled data.
- However, training of an autoencoder is supervised learning because the input itself is the teacher signal.
- BP algorithm can also be used for training.

Training of autoencoder (2/4)

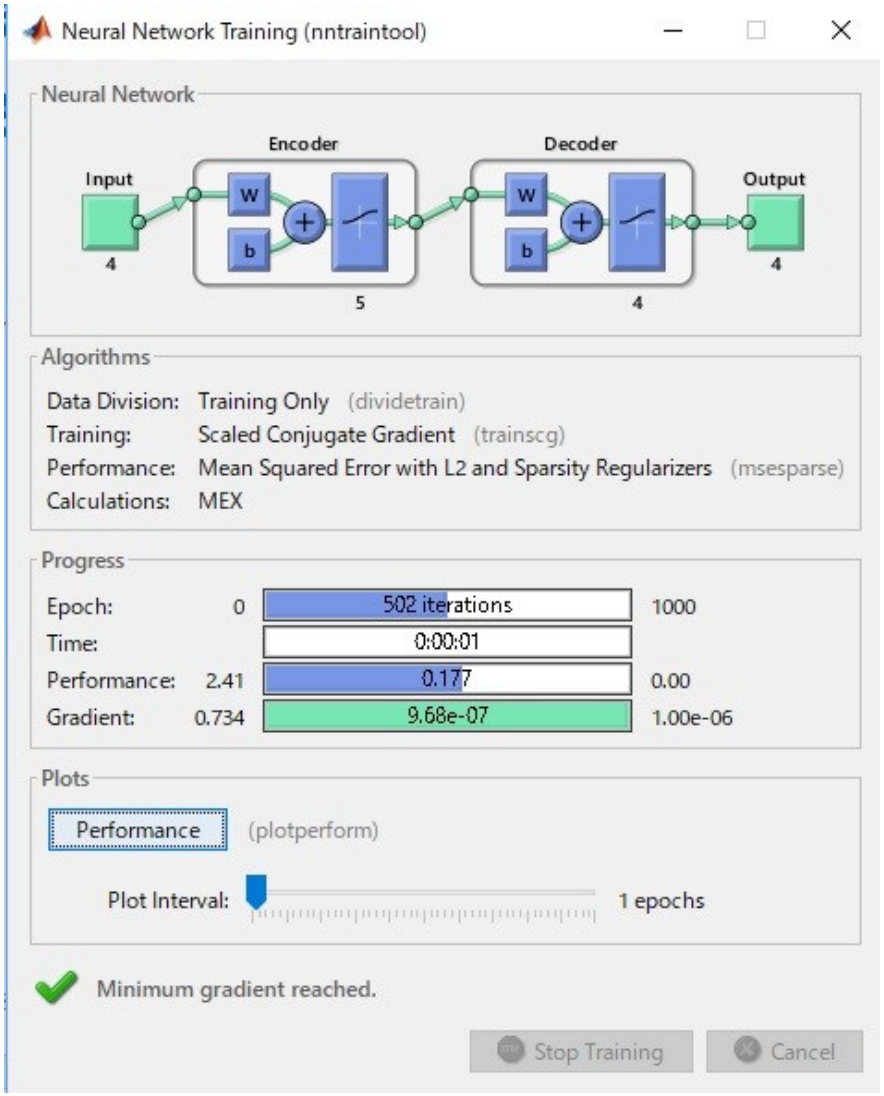
- The n-th input is denoted by x, and the corresponding output is denoted by $\widehat{x_n}$.
- Normally, the objective (loss) function used for training an autoencoder is defined as follows:

$$E(w) = \sum_{n=1}^{N} \|x_n - \widehat{x_n}\|^2 \quad (1)$$

- where w is a vector containing all weights and bias of the network.
- If the input data are binary, we can also define the objective function as the cross-entropy.
- For a single hidden layer MLP, the hidden layer is called the encoder, and the output layer is called the decoder.

Training of autoencoder (3/4)

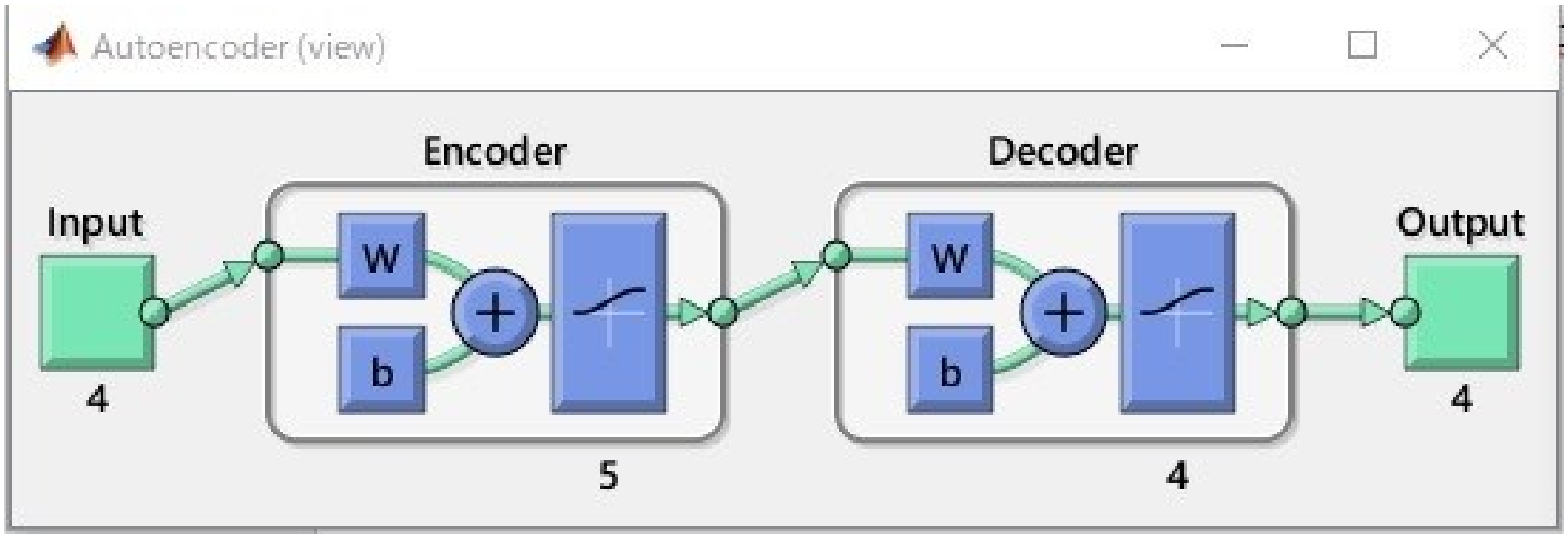
- Train an autoencoder for the well-known IRIS database using Matlab.
- The hidden layer size is 5.
- We can also specify other parameters. For detail, visit the web page given below.



[Matlab manual] https://www.mathworks.com/help/nnet/ref/trainautoencoder.html

Training of autoencoder (4/4)

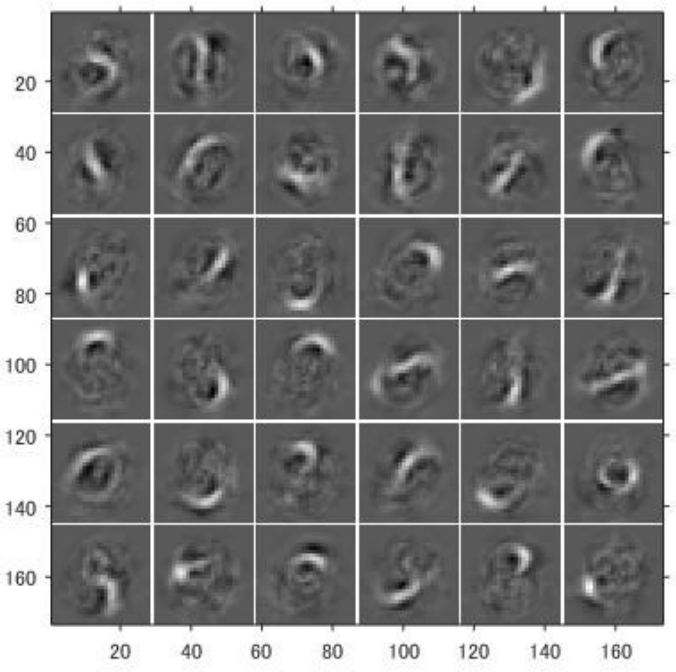
```
1. X=iris_dataset;
2. Nh=5;
3. enc=trainAutoencoder(X,Nh);
4. view(enc)
```



Internal representation of data (1/2)

```
1. X=digitTrainCellArrayData;
2. Nh=36;
3. enc=trainAutoencoder(X,Nh);
4. plotWeights(enc);
```

- An autoencoder is trained for the dataset containing 5,000 handwritten digits.
- We used 500 iterations for training, and the hidden layer size 36.
- The right figure shows the weights of the hidden layer.
- These are similar to the **Eigenfaces**.

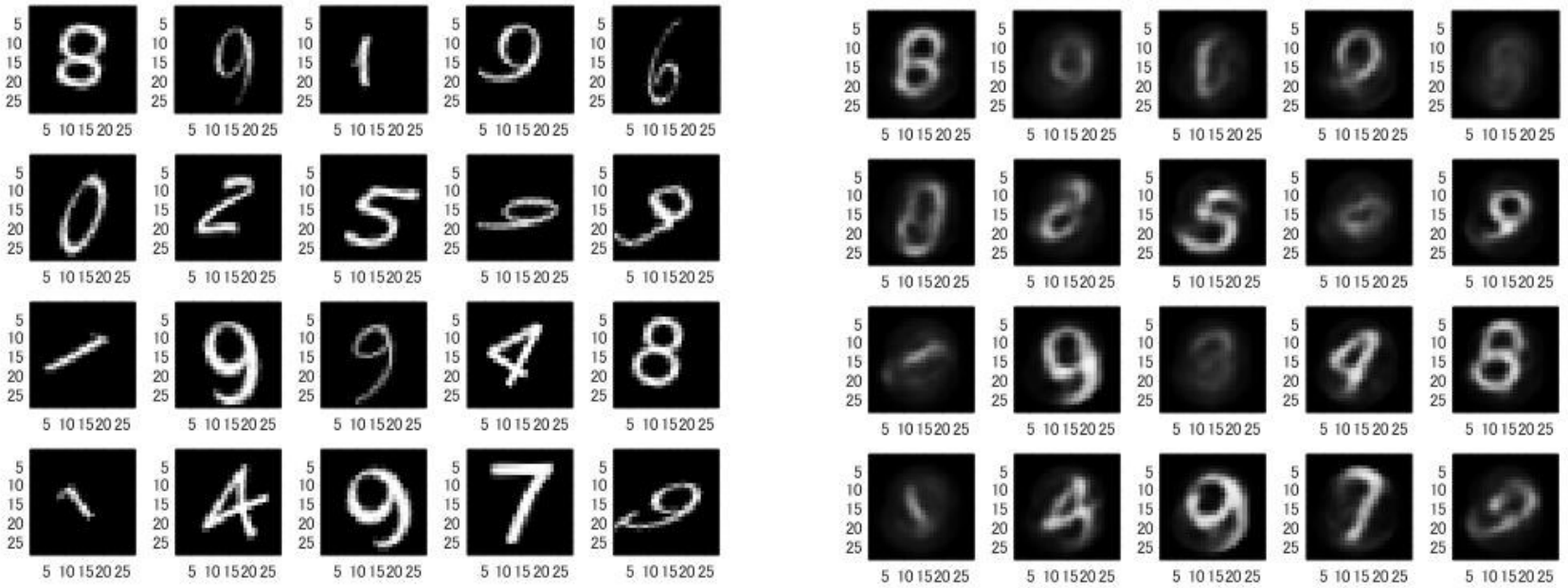


The “weight image” for each hidden neuron serves as a basis image. Each datum is mapped to these basis images, and the results are then combined to reconstruct the original image.

Internal representation of data (2/2)

```
XTrain=digitTrainCellArrayData;
Nh=36;    % We can get better results using a larger Nh
enc=trainAutoencoder(XTrain,Nh);
XTest=digitTestCellArrayData;
xReconstructed=predict(enc,XTest);
```

$\left\|X-\hat{X}\right\|$=0.0184



Training with l_2-norm regularization (1/2)

- The main purpose of autoencoder is to reconstruct the input space using a smaller number of basis functions.
- If we use the objective function given by (1), the results may not generalize well for test data.
- To improve the generalization ability, a common practice is to introduce a penalty in the objective function as follows:

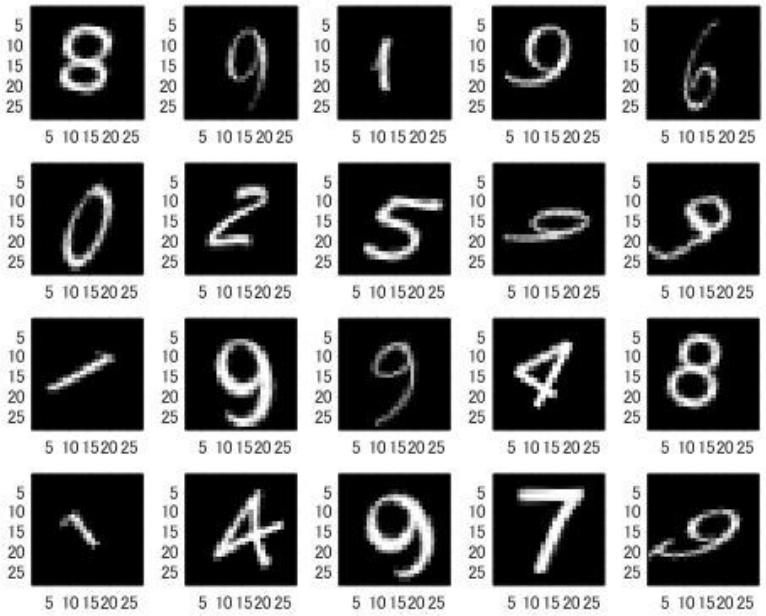
$$E(w) = \sum_{n=1}^{N} \|x_n - \widehat{x_n}\|^2 + \lambda \|w\|^2 \quad (2)$$

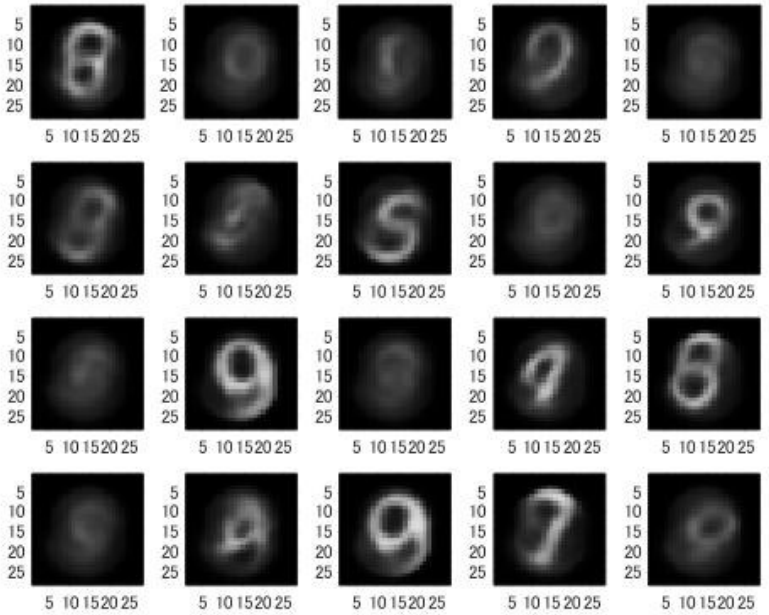
$$\|w\|^2 = \sum_{k=1}^{N} \sum_{j=1}^{L} \sum_{i=1}^{N_k} \left\|w_{ji}^{k}\right\|^2 \quad (3)$$

- The effect of introducing this l_2-norm is to make the solution more "smooth".

Training with l_2-norm regularization (2/2)

$\|X - \hat{X}\|$=0.0261





For this example, we cannot see the positive effect clearly. Generally speaking, however, if the inputs are noisy, regularization can obtain better results.

Training with sparsity regularization (1/6)

- In *nearest neighbor-based* approximation, each datum is approximated by one of the already observed data (i.e. the nearest one).
- In PCA, each datum is approximated by a point in a *linear space spanned by the basis vectors* (eigenvectors).
- Using autoencoder, each datum is approximate by a *linear combination* of the hidden neuron outputs.
- Usually, the basis functions are global in the sense that ANY given datum can be approximated well by using the same set of basis functions.
- Usually, the number N_b of basis functions equals to the rank r of the linear space. For PCA, $N_b \ll r$ because we use the "principal basis functions".

[Lv and ZHAO, 2007] https://www.emeraldinsight.com/doi/abs/10.1108/17427370710847327

Training with sparsity regularization (2/6)

- Using an autoencoder, however, we can make the number N_h of hidden neurons larger than the rank r.
- Instead, we can approximate each datum using a much smaller number of hidden neurons.
- This way, we can encode each data point using a small number of parameters as follows:

$$x = \sum_{n=1}^{m} w_{k_n} y_{k_n} \qquad (4)$$

- where m is the number of hidden neurons for approximating x, y_{k_n} is the output of the k_n-th hidden neuron, $k_1 < k_2 < \cdots < k_m \epsilon [1, N_h]$.

Training with sparsity regularization (3/6)

- For sparse representation, we introduce another penalty in the objective function as follows:

$$E(w) = \sum_{n=1}^{N} \|x_n - \widehat{x_n}\|^2 + \lambda \|w\|^2 + \beta \cdot F_{sparsity} \quad (5)$$

- To define the sparsity term $F_{sparsity}$, we need the *average output activation value* of a neuron given by

$$\widehat{\rho_j} = \frac{1}{N} \sum_{i=1}^{N} g(u_j(x_i)) \quad (6)$$

- where N is the number of training data, and $u_j(x_i)$ is the *effective input* of the j-th hidden neuron for the i-th datum.
- The neuron is very "active" if $\widehat{\rho_j}$ is high. To obtain a sparse neural network, it is necessary to make the neurons less active. This way, we can reconstruct any given datum using less number of hidden neurons (basis functions).

Training with sparsity regularization (4/6)

- Based on the average output activation value, we can define the sparsity term using the Kullback–Leibler divergence as follows:

$$F_{sparsity} = \sum_{j=1}^{N_h} KL(\rho \parallel \widehat{\rho_j})$$

$$= \sum_{j=1}^{N_h} [\rho \log\left(\frac{\rho}{\widehat{\rho_j}}\right) + (1-\rho)\log\left(\frac{1-\rho}{1-\widehat{\rho_j}}\right)] \quad (7)$$

- where ρ is a sparsity parameter to be specified by the user. The smaller, the sparser.

Training with sparsity regularization (5/6)

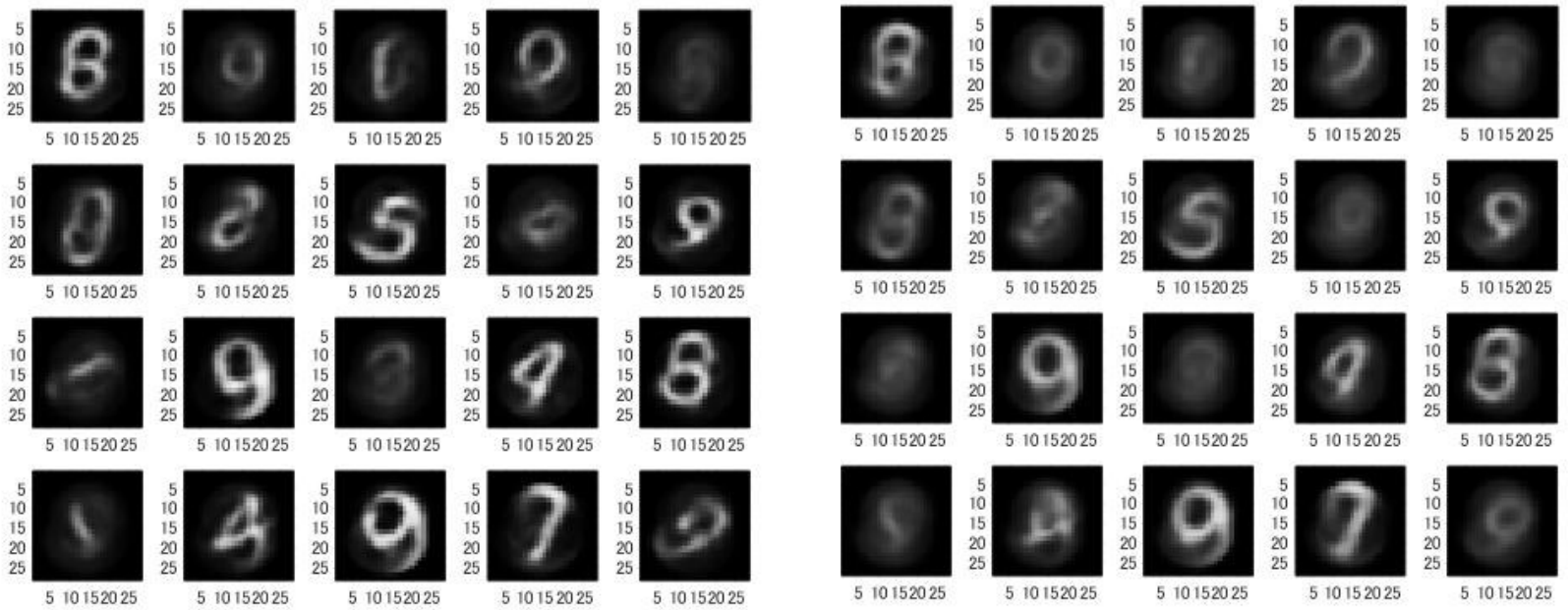
```
enc = trainAutoencoder(XTrain,Nh,...
        'L2WeightRegularization',0.004,...
        'SparsityRegularization',4,...
        'SparsityProportion',0.10);
```

$$E(w) = \sum_{n=1}^{N} \|x_n - \widehat{x_n}\|^2 + \lambda \|w\|^2 + \beta \cdot F_{sparsity} \quad (5')$$

- By specifying a small sparsity proportion ρ, we can get an autoencoder with less active hidden neurons.
- If we use a proper norm of w, we can also reduced the number of non-zero weights of the hidden neurons, and make the network more sparse.

Training with sparsity regularization (6/6)

$\|X - \hat{X}\|$=0.0268



In this example, the reconstructed images are not better. However, with less active hidden neurons, it is possible to “interpret” the neural network more conveniently (see reference below).

[Furukawa and ZHAO, 2017] *https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8328367*

Training of deep network (1/5)

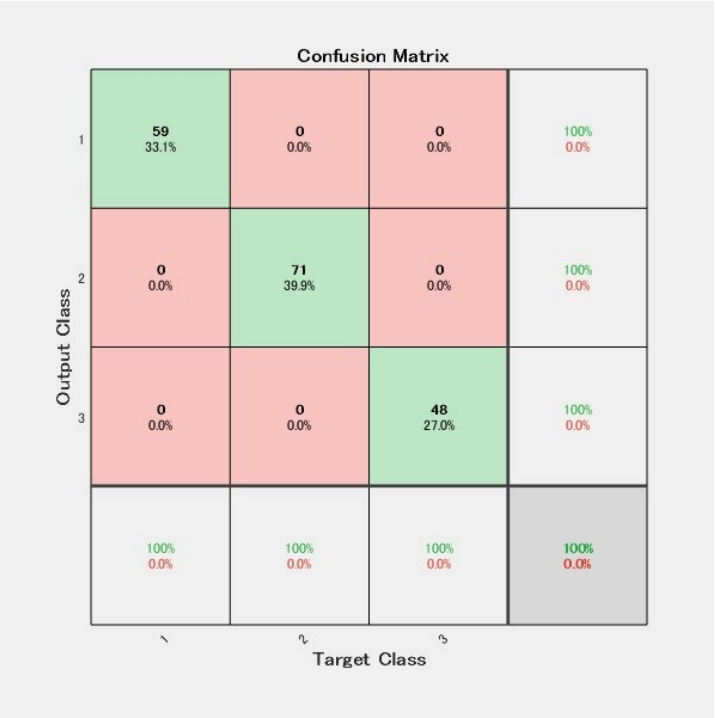
- Using autoencoder, we can extract useful features for representing the input data without using “labels”.
- The extracted features can be used by another MLP for making the final decision.
- Shown in the next page is a Matlab example. In this example,
  - The first two hidden layers are found by using autoencoder training;
  - The last layer is a soft-max layer.
  - The three layers are “stacked” to form a deep network, which can be re-trained using the given data using the BP algorithm.

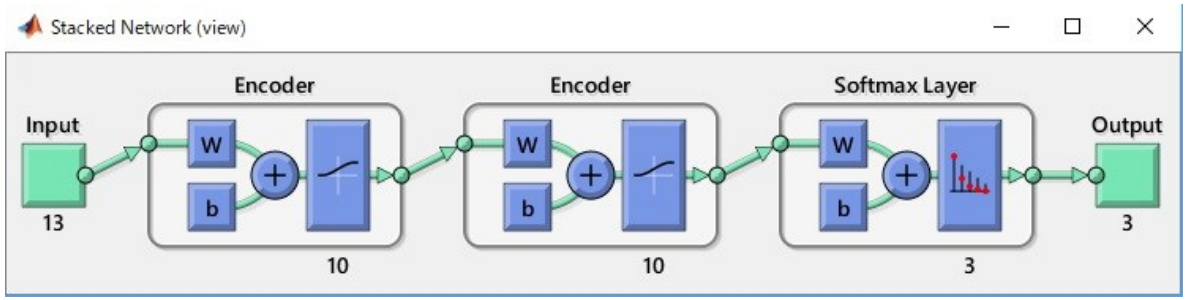
Training of deep network (2/5)

- autoenc1 = trainAutoencoder(X,Nh1,'DecoderTransferFunction','purelin');
- features1 = encode(autoenc1,X);
- autoenc2 = trainAutoencoder(features1,Nh2'DecoderTransferFunction',… 'purelin','ScaleData',false);
- features2 = encode(autoenc2,features1);
- softnet = trainSoftmaxLayer(features2,T,'LossFunction','crossentropy');
- deepnet = stack(autoenc1,autoenc2,softnet);
- deepnet = train(deepnet,X,T);

Training of deep network (3/5)

- Shown here is an example using the wine dataset.
- The right figure is the confusion matrix of the deep network, and
- the figure in the bottom is the structure of the deep network.





Training of deep network (4/5)

- To summarize, we can design a deep neural network (with K layers, not include the input layer) as follows:
  - Step 1: i=1; X(i)=X(0); % X(0) is the given data
  - Step 2: Train an autoencoder A(i) based on X(i);
  - Step 3: X(i+1)=encoder(X(i));
  - Step 4: If i<K, return to Step 3;
  - Step 5: Train a regression layer R using BP
    - Training data: X(K-1)
    - Teacher signal: Provided in the training set
  - Step 6: Stack [A(1), A(2),…,A(K-1),R] to form a deep MLP.

Training of deep network (5/5)

- We can also train a deep autoencoder by modifying the algorithm slightly as follows:
  - Step 1: i=1; X(i)=X(0); % X(0) is the given data
  - Step 2: Train an autoencoder A(i) based on X(i);
  - Step 3: X(i+1)=encoder(X(i));
  - Step 4: If i<K, return to Step 3;
    - K is the specified number of layers
  - Step 5: Train a regression layer R using BP
    - Training data: X(K-1)
    - Teacher signal: X(0)
  - Step 6: Stack [A(1), A(2),…,A(K-1),R] to form a deep autoencoder.

Homework

- Try the Matlab program for digit reconstruction given in the following page:

  https://www.mathworks.com/help/nnet/ref/trainautoencoder.html
- See what happen if we change the parameter for l2-norm regularization; and
- See what happen if we change the parameter for sparsity regularization.
- You may plot
  - The weights of the hidden neurons (as images);
  - The outputs of the hidden neurons; or
  - The reconstructed data.