CSE-E4810 Machine Learning and Neural Networks (5 cr)

Lecture 13: Deep Learning

Prof. Juha Karhunen

https://mycourses.aalto.fi/course/view.php?id=13086

Introduction

- In this lecture, we discuss deep learning and neural networks.
- Deep learning was introduced by Hinton and Salakhutdinov in 2006.
- Deep learning has become a hot topic in machine learning, leading to a renaissance of neural networks research.
- This is because deep networks have provided world-record results in many benchmark classification and regression problems.
- It is currently used extensively for example in image classification and speech recognition.
- There is almost no literal material on this lecture in addition to these slides.
- Except for a couple of pages explaining convolutional networks.

- In Du's and Swamy's book, there is only about 1.5 pages on deep learning in Section 19.13 with no mathematics.
- At the end of lecture there are references from which you can find more information.
- An especially useful is the first reference, Bengio's et al. almost complete book "Deep learning".
- It is mathematically not too difficult, and will probably become a widely used standard reference on deep learning.
- Our discussion is somewhat superficial because deep learning is a quite advanced topic.
- Its more thorough discussion is beyond the scope of this course.
- There are no exercises associated with this lecture.
- Instead, we present some experimental results.

Justifications for deep learning

- It would be desirable to have deep neural networks having several hidden layers.
- The layer closest to the data vectors learns simple features, while the higher layers could learn higher-level features.
- For example in digital images first low-level features such as edges and lines in different directions are learned in the first hidden layer.
- Followed by shapes, objects, etc. in higher level layers (see Fig. 1).
- Human brains, especially cortex, contain deep biological neural networks working in this way.

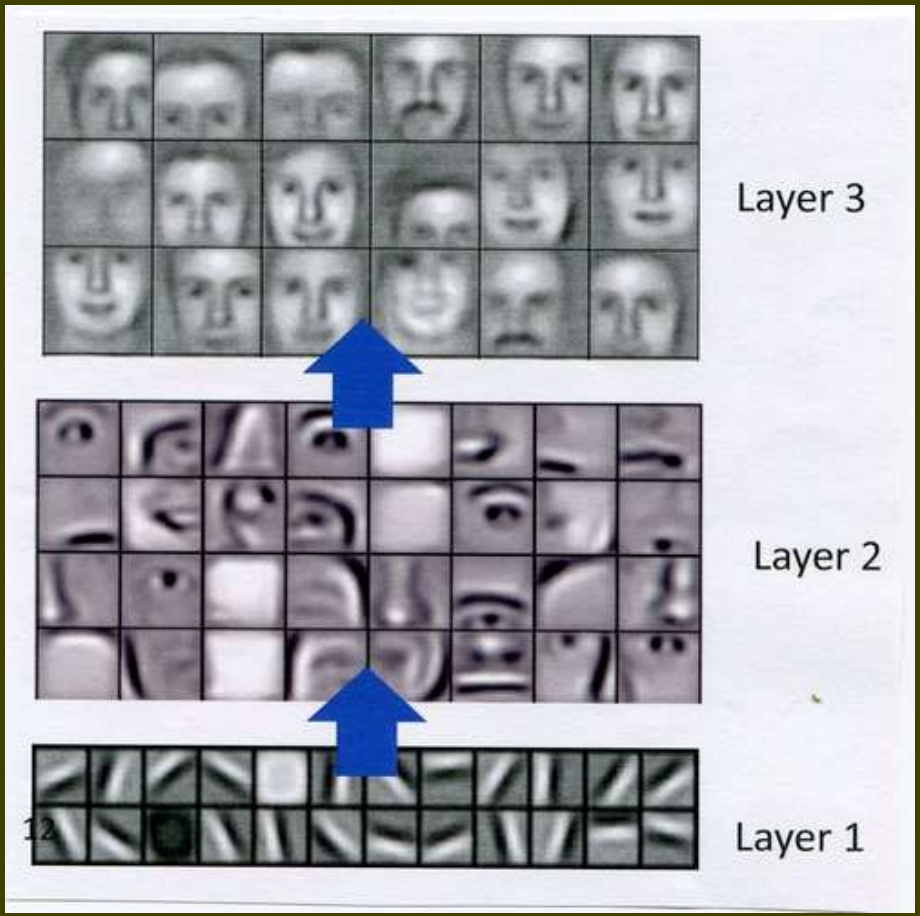

Figure 1: Higher layers extract more general features of face images. The input layer not shown consist of pixels.

- It has turned out that when trained properly, such deep networks can achieve better performance than neural networks with only one or two hidden layers.
- This is largely caused by unsupervised pre-training using restricted Boltzmann machines or autoencoders.
- The idea is not only to try to learn the nonlinear input-output mapping, but also the underlying structure of input vectors.

Multilayer perceptron (MLP) networks

- Multilayer perceptron (MLP) networks and their learning algorithms were discussed in lectures 4 and 5.
- A schematic diagram of an MLP network consisting of an input layer, two hidden layers, and output layer is shown in Figure 2.

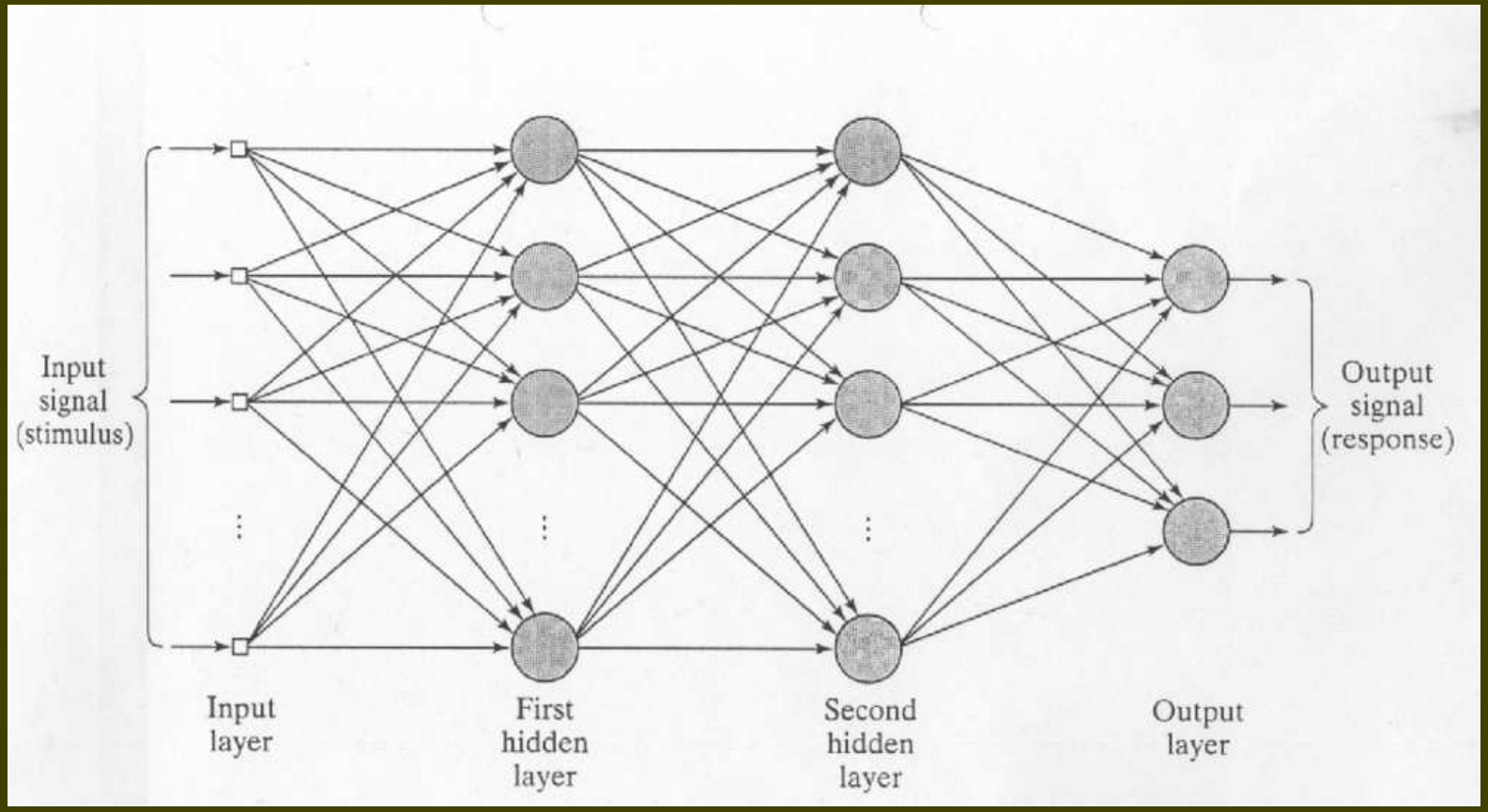

Figure 2: The architecture of a multilayer perceptron network with two hidden layers.

- In the input layer a data vector is inputted to the MLP network.
- Computations take place in the hidden layers and output layer.
- With detailed notations MLP network becomes quite complicated.
- It is trained in supervised manner using N known training pairs $\{\mathbf{x}_i, \mathbf{d}_i\}$ where $\mathbf{x}_i$ is i:th input vector and $\mathbf{d}_i$ is the corresponding desired response.
- The vector $\mathbf{x}_i$ is inputted to the MLP network and the corresponding output $\mathbf{y}_i$ is computed.
- In principle, the mean-square error $E = \mathrm{E}\{\| \mathbf{d}_i - \mathbf{y}_i \|^2\}$ is used for learning the weights of the network.
- The steepest descent learning rule for a weight w_{ji} in any layer is given by

$$\Delta w_{ji} = -\mu \frac{\partial E}{\partial w_{ji}} \tag{1}$$

- In practice, steepest descent is replaced by instantaneous gradient or a mini batch over 100-1000 training pairs.
- The required gradients can be computed first for the neurons in the output layer using their local errors.
- These local errors are then propagated to the previous layer, and the weights of its neurons can be updated, and so on.
- The name backpropagation come from this.
- Usually numerous iterations and sweeps over the training data are required for convergence.
- Especially if instantaneous stochastic gradient is used.
- In practice, MLP networks have formerly had one or two hidden layers.

Problems with "deep" MLP networks

- Deep learning addresses problems encountered when applying backpropagation type algorithms to deep networks with many layers:
  - The training data set must be large.
  - Often there are lots of data but too little labeled data.
  - Learning (convergence) is slow.
  - Saddle points and plateaus in minimization of the error E cause more problems than local minima.
  - Training MLP networks having more than two hidden layers using backpropagation algorithms was previously unsuccessful.
  - The additional hidden layers do not learn useful features because the gradients in them decay exponentially.

Boltzmann Machines

- Boltzmann machines (Figure 3) are a class of neural networks introduced already in late 1980's.
- They are based on statistical physics and use stochastic neurons contrary to most other neural network methods.
- They can be used for constructing associative memories or unsupervised learning of features describing the input data.
- Learning rules for the weight matrices $\mathbf{L}$, $\mathbf{J}$, and $\mathbf{W}$ can be derived in a straightforward manner.
- However, Boltzmann machines are of little practical interest even though they perform well.
- Reason: they are limited to small-scale "toy" problems only because learning becomes intractable for problems of realistic scale.

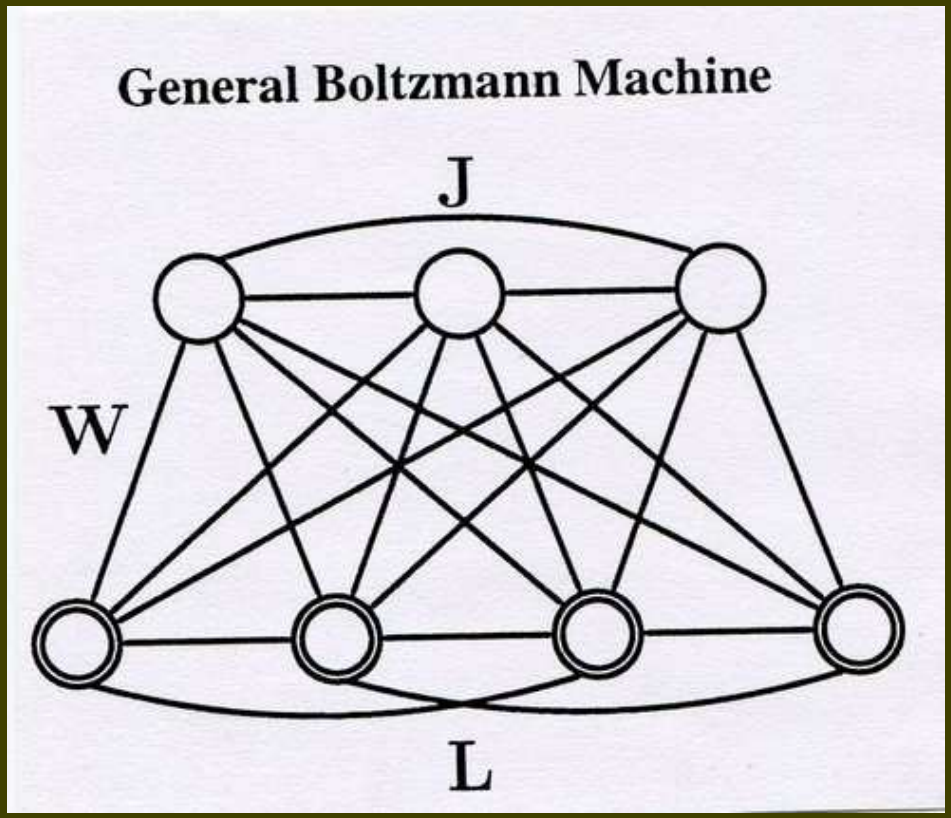

Figure 3: Boltzmann machine with connection weight matrices $\mathbf{L}$, $\mathbf{J}$, and $\mathbf{W}$ of the neurons in the visible layer, hidden layer, and hidden-to-visible layer, respectively.

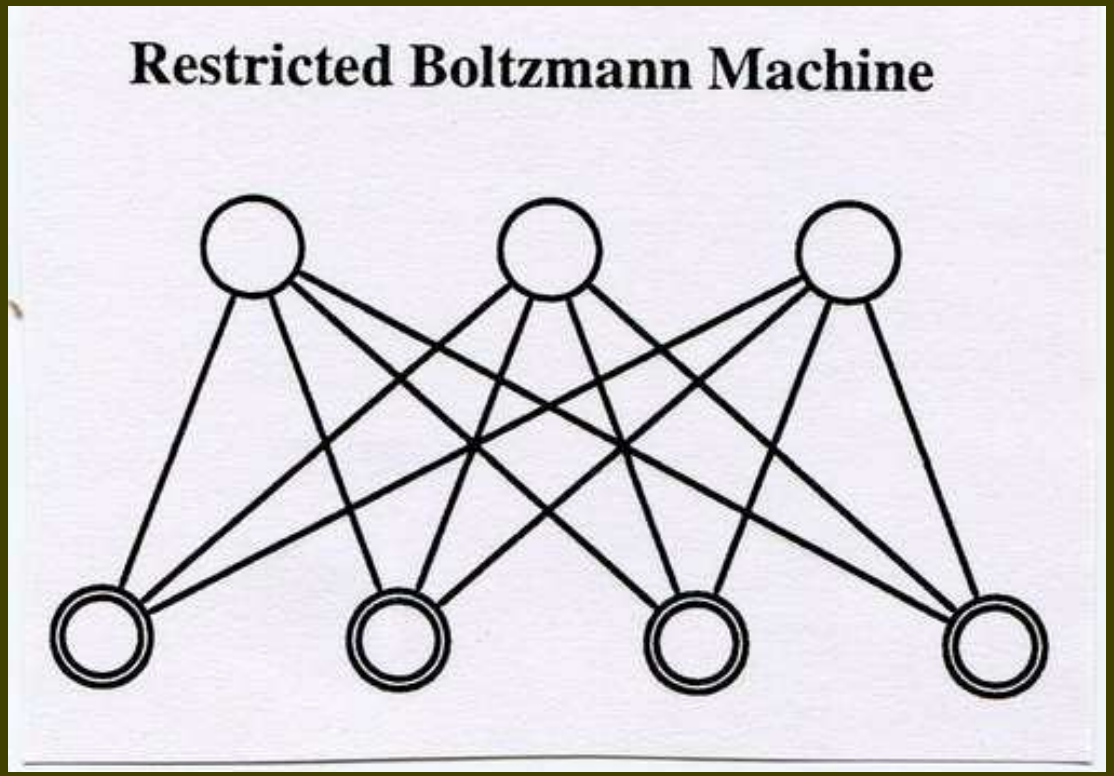

Figure 4: Restricted Boltzmann machine. Only the weights $\mathbf{W}$ between the neurons in the visible layer (bottom) and hidden layer (top) remain.

Restricted Boltzmann Machines

- Restricted Boltzmann Machines (RBMs) are simplified versions of Boltzmann machines; see Figure 4.
- The connections between the hidden neurons (top) and between the visible neurons (bottom) are removed.
- Only the connections between the neurons in the visible layer and the hidden layer remain.
- Their weights are collected to the matrix $\mathbf{W}$.
- In an RBM, the top layer represents a vector of stochastic binary features $\mathbf{h}$.
- That is, the value of the state of each neuron can be either $+1$ or -1 with a certain probability.
- The bottom layer contains stochastic binary "visible" variables $\mathbf{x}$.

- When modeling real-valued visible variables, the bottom layer is composed of linear units with Gaussian noise.
- The Restricted Bolzmann Machine was introduced already in 1980's.
- A few years ago, Hinton developed so-called contrastive divergence learning for RBMs.
- We do not discuss this learning method in detail here.
- Contrastive divergence learning is efficient enough to be practical.
- It has been applied to denoising of natural images, rapid document retrieval, recognition of handwritten digits, etc.
- But contrastive divergence learning fails for deep multilayer networks because learning is still far too slow.

Modeling binary data

- Binary data is modeled using Restricted Boltzmann Machine (Fig. 4).
- x_i is the i:th component of the "visible" input vector $\mathbf{x}$.
- And h_j is the j:th component of the "hidden" feature vector $\mathbf{h}$.
- Their joint Boltzmann distribution is

$$p(\mathbf{x}, \mathbf{h}) = \frac{1}{Z} \exp(-E(\mathbf{x}, \mathbf{h})) \tag{2}$$

where $E(\mathbf{x}, \mathbf{h})$ is an energy term given by

$$E(\mathbf{x}, \mathbf{h}) = -\sum_i b_i x_i - \sum_j b_j h_j - \sum_{i,j} x_i h_j W_{ij} \tag{3}$$

- W_{ij} is a symmetric interaction term between the input x_i and feature h_j, and b_i, b_j are bias terms.

- And the normalization constant

$$Z = \sum_x \sum_h \exp(-E(\mathbf{x}, \mathbf{h})) \tag{4}$$

- From these equations, one can derive the conditional Bernoulli distributions

$$p(h_j = 1 \mid \mathbf{x}) = \sigma(b_j + \sum_i W_{ij} x_i) \tag{5}$$

$$p(x_i = 1 \mid \mathbf{h}) = \sigma(b_i + \sum_i W_{ij} h_j) \tag{6}$$

- There $\sigma(z)$ is the logistic sigmoidal function

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{7}$$

- The marginal distribution over visible vector $\mathbf{x}$ is:

$$p(\mathbf{x}) = \sum_{\mathbf{h}} \frac{\exp(-E(\mathbf{x}, \mathbf{h}))}{\sum_{\mathbf{u},\mathbf{g}} \exp(-E(\mathbf{u}, \mathbf{g}))} \tag{8}$$

- The parameter update required to perform gradient ascent in the log-likelihood becomes ($< . >$ denotes expectation)

$$\Delta W_{ij} = \epsilon(< x_i h_j >_{data} - < x_i h_j >_{model}) \tag{9}$$

- In the data distribution $\mathbf{x}$ is taken from the data set and $\mathbf{h}$ from the conditional distribution $p(\mathbf{h} \mid \mathbf{x}, \boldsymbol{\theta})$ given by the model.
- In the model distribution both are taken from the joint distribution $p(\mathbf{x}, \mathbf{h})$ of the model.
- For the bias terms, one gets a similar but simpler equation.
- The expectations can be estimated using Gibbs sampling in which samples are generated from the respective probability distributions.

Modeling real-valued data

- Restricted Bolzmann machines can be generalized to exponential family distributions.
- For example, images with real-valued pixels can be modeled by visible units that have a Gaussian distribution.
- Its mean is determined by the hidden units:

$$p(x_i \mid \mathbf{h}) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp(-\frac{(x - b_i - \sigma_i \sum_j h_j w_{ij})^2}{2\sigma_i^2}) \tag{10}$$

$$p(h_j = 1 \mid \mathbf{x}) = \sigma(b_j + \sum_i W_{ij} \frac{x_i}{\sigma_i}) \tag{11}$$

- The marginal distribution over visible units $\mathbf{x}$ is given by Eq. (8)

with an energy term

$$E(\mathbf{x}, \mathbf{h}) = \sum_i \frac{(x_i - b_i)^2}{2\sigma_i^2} - \sum_j b_j h_j - \sum_{i,j} h_j w_{ij} \frac{x_i}{\sigma_i} \qquad (12)$$

- If the variances are set to $\sigma_i^2 = 1$ for all visible units i, the parameter updates are the same as defined in Eq. (9).
- For details and refinements of the learning procedure, see the paper (Salakhutdinov and Hinton, AISTATS 2007).

Deep belief networks

- Deep belief networks are generative models with many layers of hidden causal variables.
- Each layer of a deep belief network consists of a restricted Boltzmann machine.
- Recently, Hinton et al. (2006) derived a way to perform fast, greedy

learning of deep belief networks one layer at a time.

- When a RBM has learned, its feature activations are used as the "data" for training the next RBM in the deep belief networks.
- See Figure 5.
- Important aspect of this layer-wise learning procedure:
- Each extra layer increases a lower bound on the log probability of the data.
- Provided that the number of features per layer does not decrease.
- Layer-by-layer training can be repeated several times for learning a deep, hierarchical model of the data.
- Each layer of features captures strong high-order correlations between the activities of the features in the layer below.

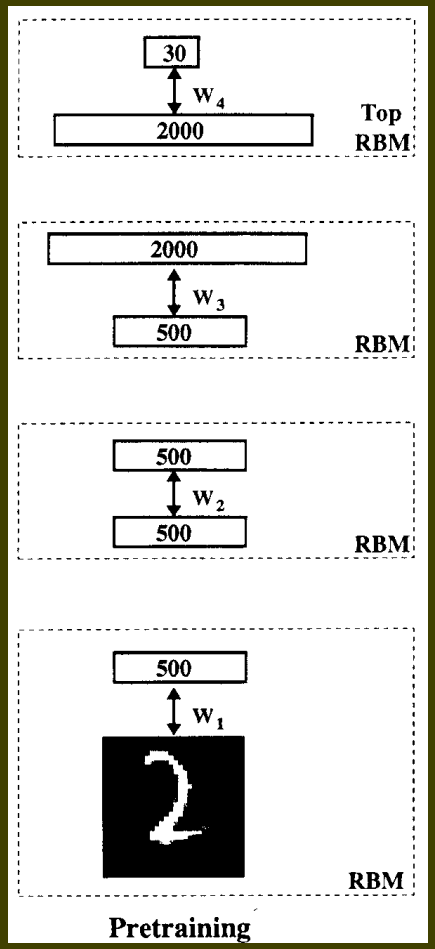

Figure 5: Pretraining of a stack of Restricted Boltzmann Machine.

- This representation is more efficient than using a single hidden layer with many units.
- Using the greedy algorithm, one can learn a relatively good hierarchical representation of the data.
- But it is not yet an optimal representation.
- This is because the weights of each layer are learned independently of the weights of the next layers.
- Therefore, the representation found by the greedy learning algorithm can be improved using a fine tuning algorithm for the weights.
- To this end, one can use a variant of the standard backpropagation algorithm which is good at fine tuning.

Recursive learning of deep generative model

1. Learn the parameter vector W^1 of the first layer of a binary Bernoulli or real-valued Gaussian model.
2. Freeze the parameters of the lower level model, and:
   - Use as the data for training the next layer of binary features:
   - The activation probabilities of the binary features, when they are driven by the training data.
3. Freeze the parameters W^2 that define the second layer of features, and:
   - Use the activation probabilities of those features as data for training the third layer of features.
4. Proceed recursively for as many layers as desired.

Experimental results with MNIST data

- Consider the widely used MNIST handwritten digit recognition task.
- There are 60,000 28×28 images of handwritten digits $0, 1, \ldots, 9$ used for training and 10,000 more for testing.
- Furthermore, 10,000 of the training images were used for validation.
- The next two figures show classification errors as a function of the number of layers in the neural network.
- Results obtained using standard supervised learning start to degrade after adding third layer.
- While with unsupervised pre-training they are better and improve up to adding fourth layer.

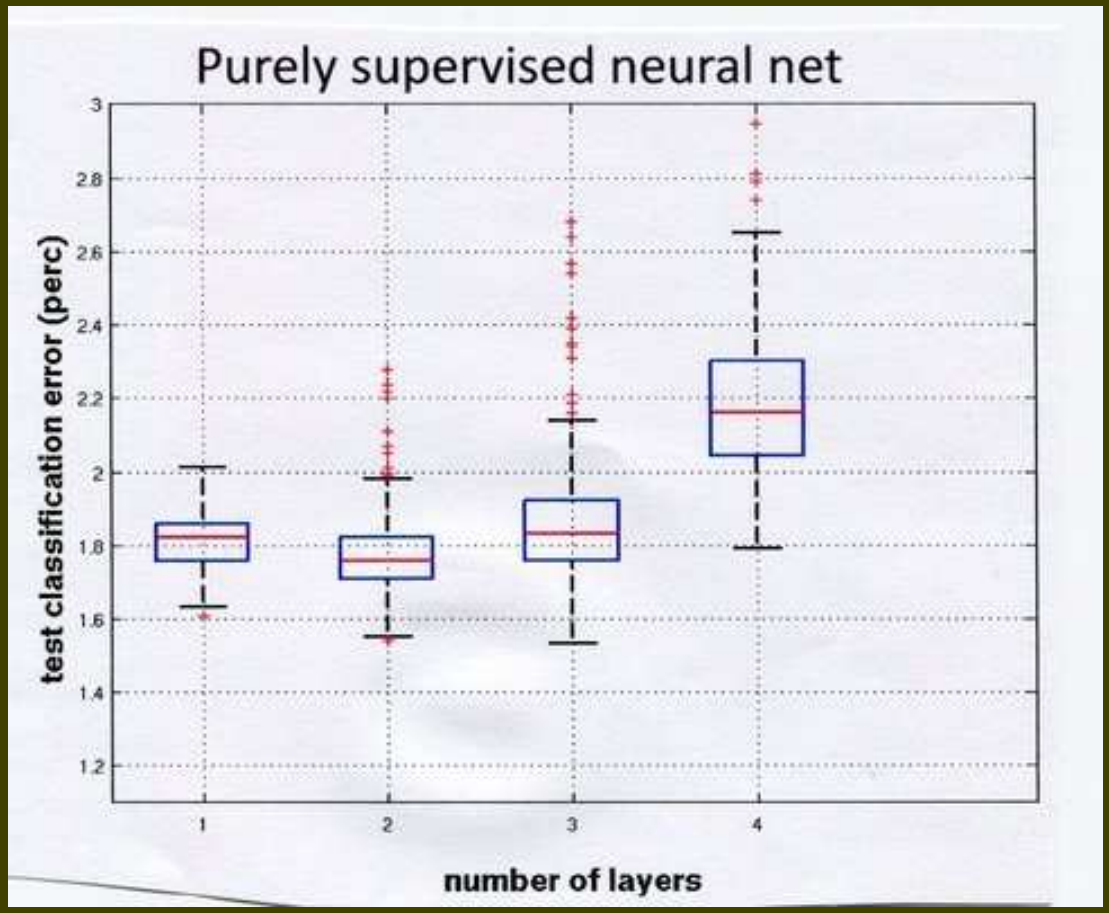

Figure 6: Handwritten digit recognition error rates using MLP networks trained in standard manner (MNIST data).

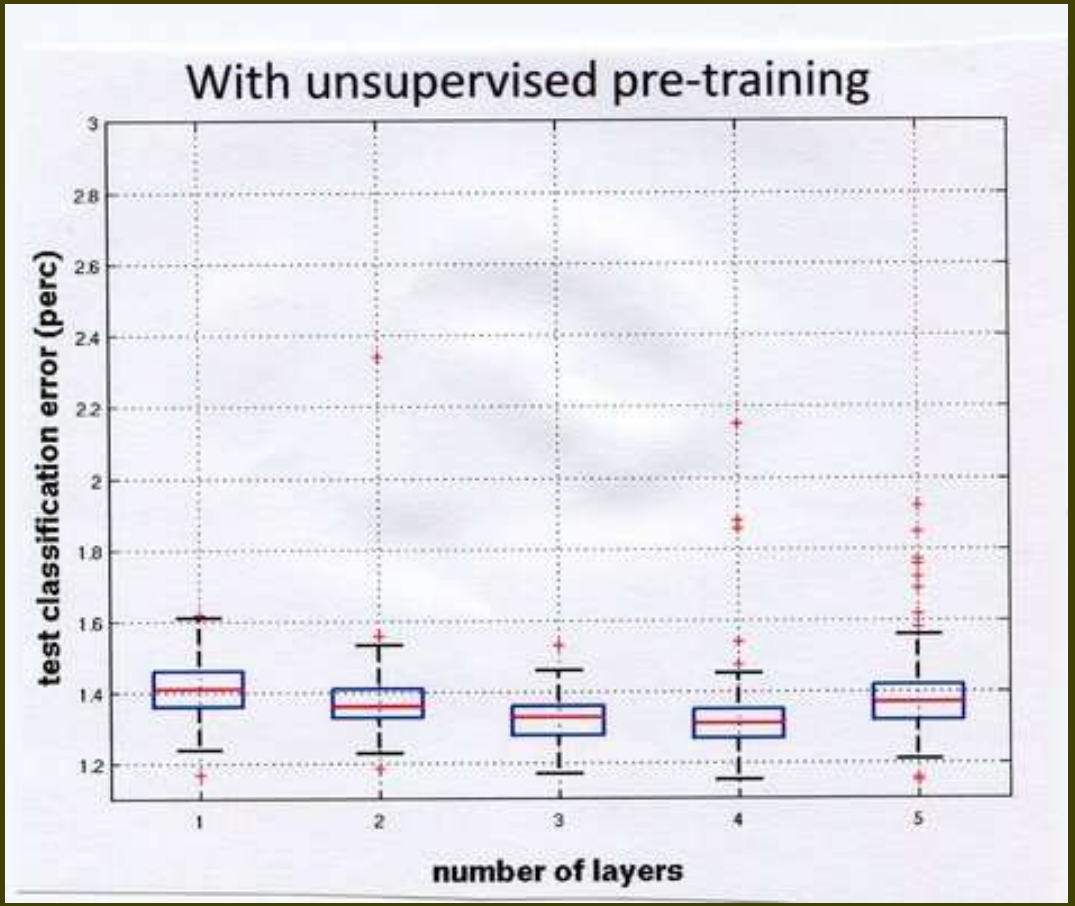

Figure 7: Handwritten digit recognition error rates using unsupervised RBMs and denoising auto-encoders in pre-training (MNIST data).

Nonlinear neighborhood component analysis

- K nearest neighbour (KNN) method is an old simple classification method which performs often well.
- A new data vector is assigned to the class in which it has most neighbours among its K nearest neighbours.
- Neighborhood Components Analysis (NCA) tries to maximize the performance of KNN classification algorithm.
- It tries to construct such a mapping that the class separation is as large as possible in the low-dimensional feature space.
- Developed by Goldberger et al. (NIPS 2004)
- Essentially the same method was introduced by Sami Kaski and Jaakko Peltonen one year earlier (ICML 2003).
- Under the name Informative Discriminant Analysis.

- A nonlinear version of neighborhood component analysis can be developed by appying it in one of the layers of a deep network.
- The next figures show mappings of the 784-dimensional MNIST data vectors to two dimensions provided by:
  - Nonlinear Neighbourhood Component Analysis (NCA).
  - Linear Neighbourhood Component Analysis (NCA).
  - Linear Discriminant Analysis (LDA).
  - Linear Principal Component Analysis (PCA).
- Nonlinear NCA with deep belief network performs clearly the best.
- When the MNIST data vectors are compressed to 30-dimensional, it can achieve 1.00% classification error.
- The respective best error rates are 1.6% for back-propagation and 1.4% for support vector machines.

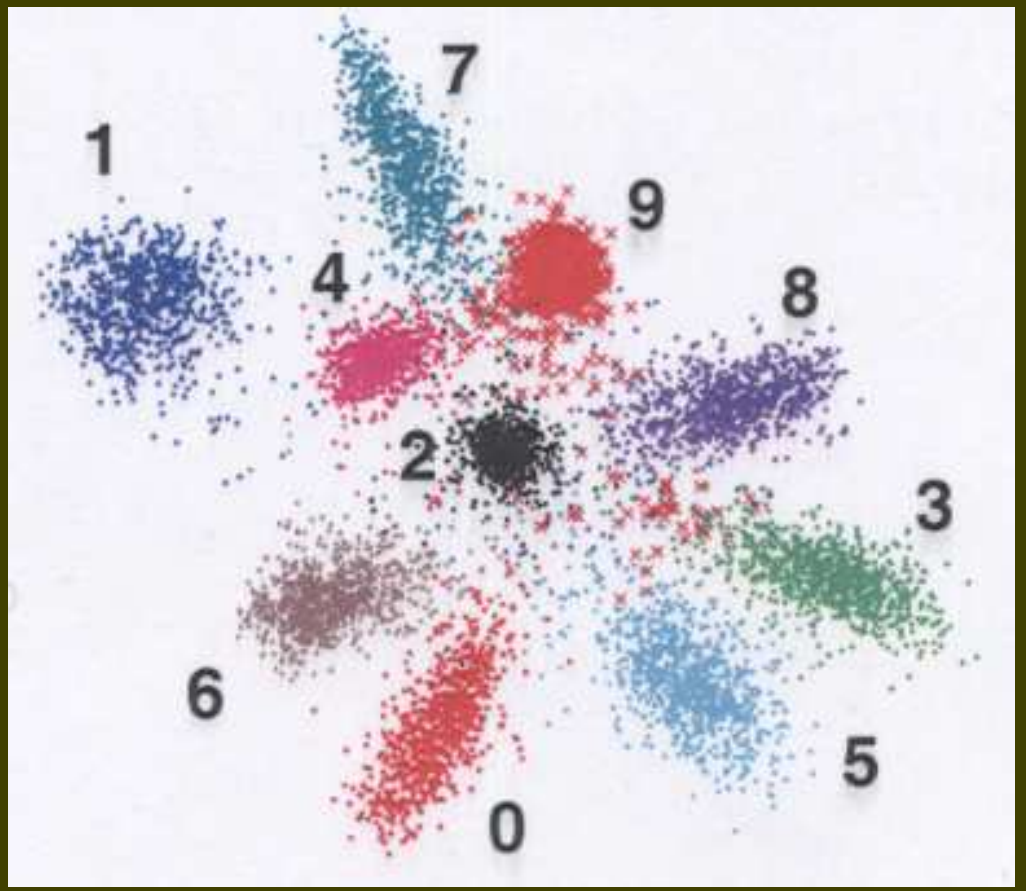

Figure 8: Mapping of MNIST data using nonlinear neighborhood component analysis (NCA) and a 784-500-500-2000-2 encoder.

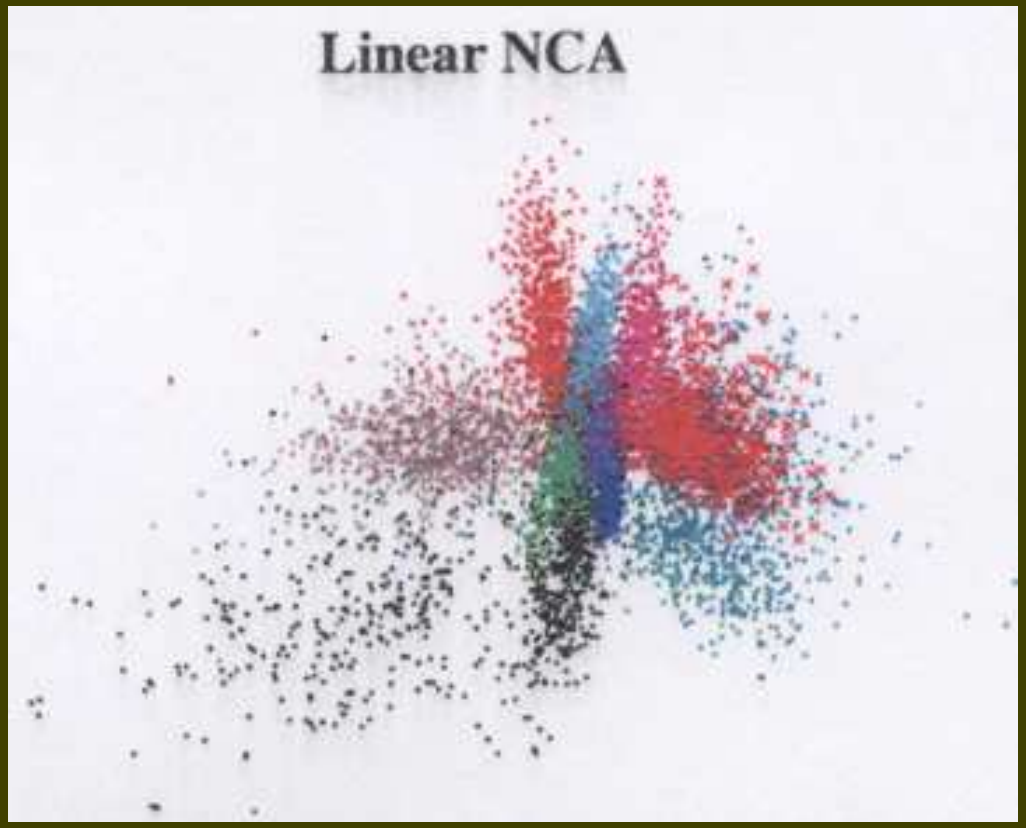

Figure 9: Mapping of MNIST data using linear neighborhood component analysis (NCA).

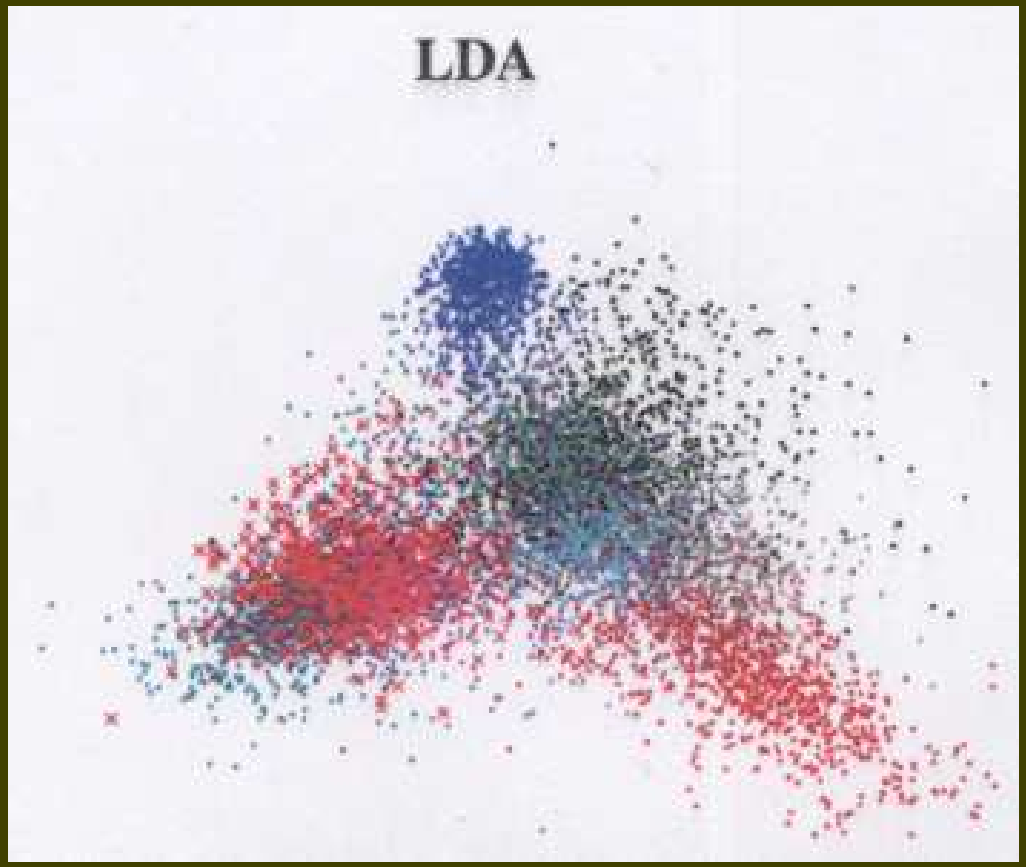

Figure 10: Mapping of MNIST data using linear discriminant analysis (LDA).

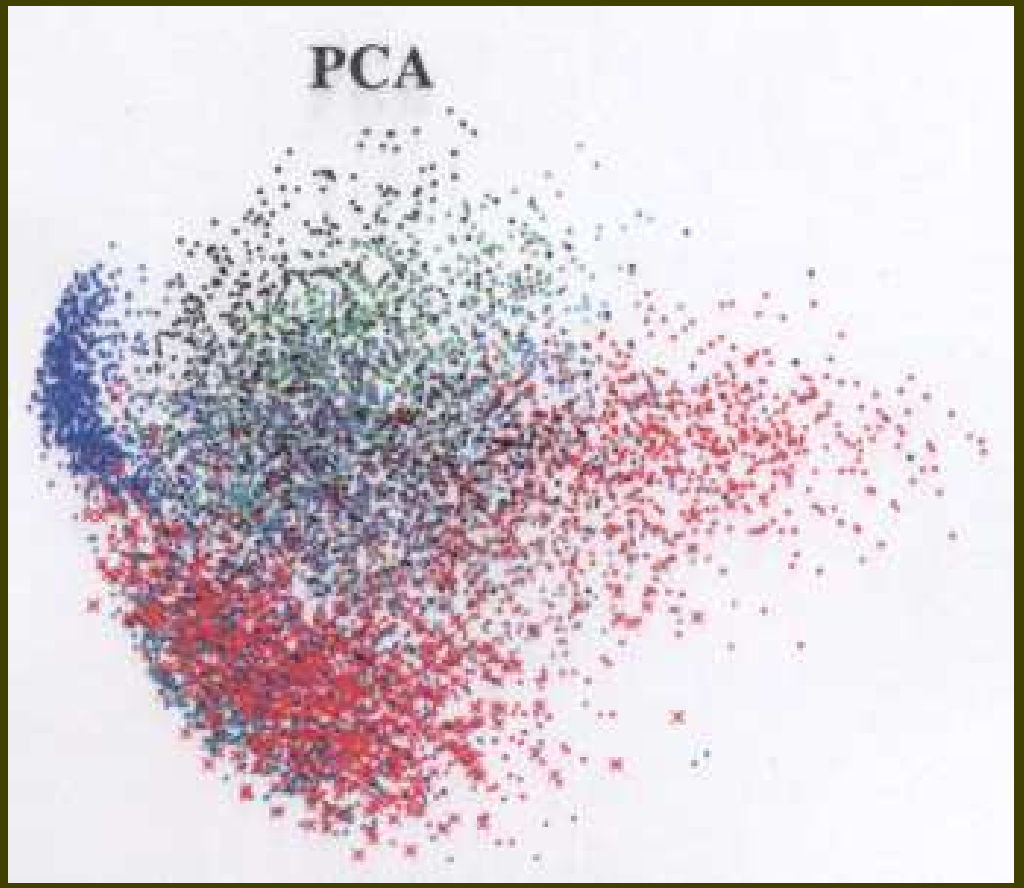

Figure 11: Mapping of MNIST data using principal component analysis (PCA).

Deep Boltzmann Machines

- Salakhutdinov and Hinton introduced Deep Boltzmann Machines (DBM) in 2009.
- Contrary to Deep Belief Networks (DBN), in DBM the connections between the layers are undirected ; see Figure 12.
- This allows better flow of information between neurons.
- Note that in Figure 12 the activities of the visible units (data vector) are denoted by $\mathbf{v}$.
- The learning algorithms for Deep Boltzmann Machines are quite advanced and beyond the scope of our course.
- Unsupervised pretraining using restricted Boltzmann machines or autoencoders is needed in Deep Boltzmann machines.
- Starting from random weights, DBM does not learn anything useful.

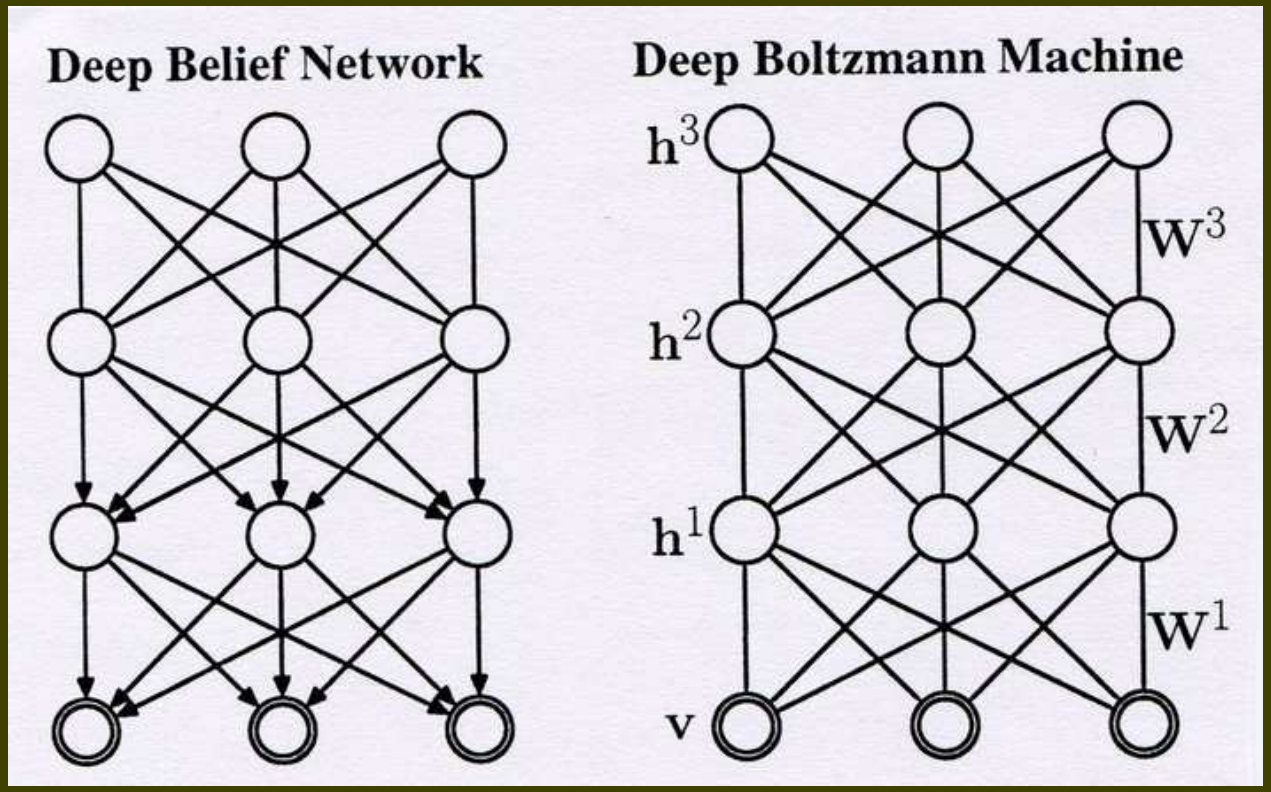

Figure 12: Deep Boltzmann machine has undirected connections.

Experimental results

- When trained well, Deep Boltzmann Machines (DBMs) can yield somewhat better results than Deep Belief Networks (DBNs)
- For the MNIST data, DBM can achieve 0.95% classification error compared with 1.2% error using DBN.
- The NORB dataset is considerably more difficult than MNIST.
- It contains images of 50 different toy objects belonging to 5 classes: cars, trucks, planes, animals, and humans; see Fig. 13.
- These 96×96 digital images have been taken from different viewpoints under different lightning conditions.
- In classifying NORB data, DBM achieved somewhat better results than SVM's and much better results than K-nearest neighbours.
- DBM can be used for generate new samples resembling NORB data.

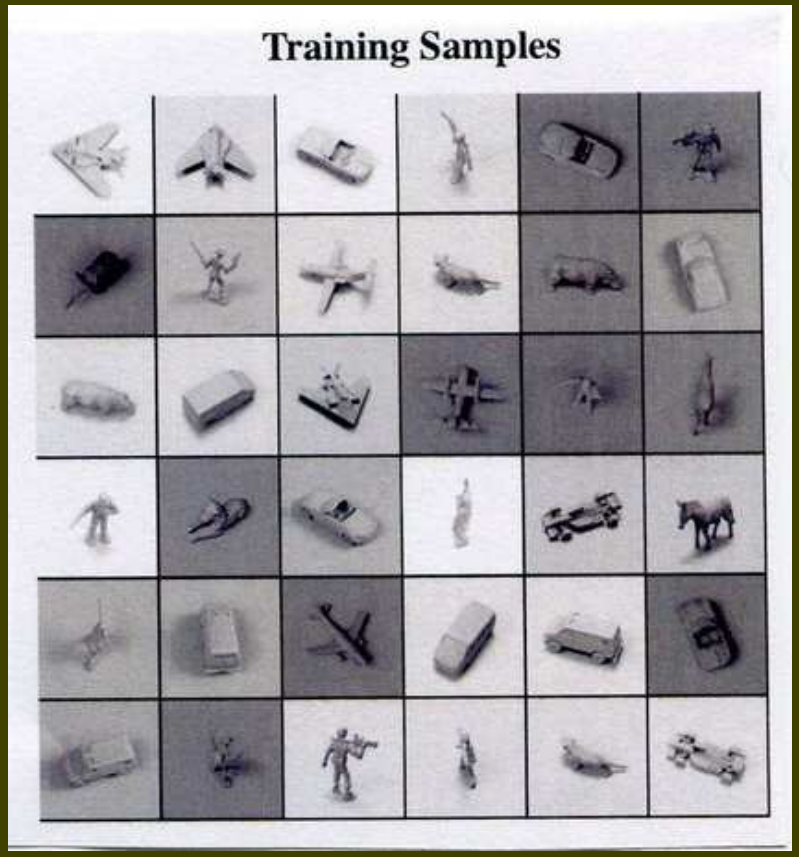

Figure 13: Random samples from the NORB training set.

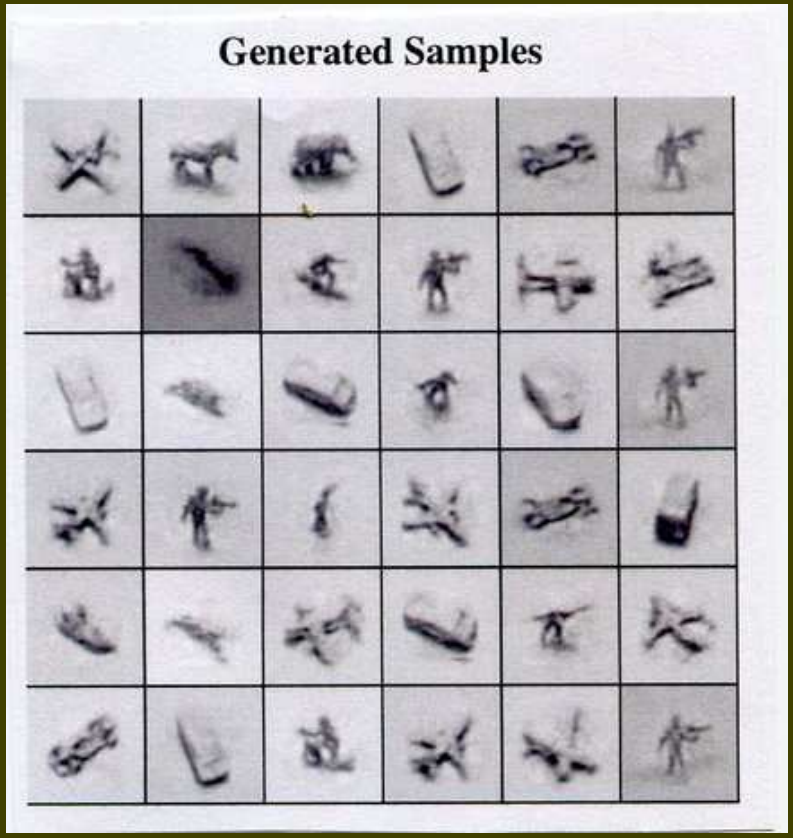

Figure 14: Samples resembling NORB data generated from a 3-hidden layer DBM by running the Gibbs sampler for 10,000 steps.

Inference and sampling in Deep Belief Networks and Deep Boltzmann Machines

- Thus far we have discussed only learning, but not using these networks for inference and generation of new samples.
- Actually Deep Belief Network (DBN) provides two networks that have common weights.
- In the recognition network, data vector is inputted to the visible layer, and information then proceeds upwards.
- A binary classifier constructed from the the activities of the neurons in the uppermost hidden layer is used for recognition.
- New data vectors resembling training vectors can be generated by sampling the uppermost hidden layer with MCMC (Markov chain Monte Carlo) methods.

- After this, the information proceeds downwards to the visible layer, generating a new data vector that should resemble some existing ones.
- Every layer in DBN tries to represent all the dependencies in the layer below it.
- Deep Boltzmann machines have only one undirected network.
- Both recognition and generation of new samples require inference over the entire network.
- For recognition mean field methods are typically used, and MCMC for generation.
- These Bayesian learning methods are beyond the scope of our course.
- The upper layers represent only such information that lower layers have not managed to represent.

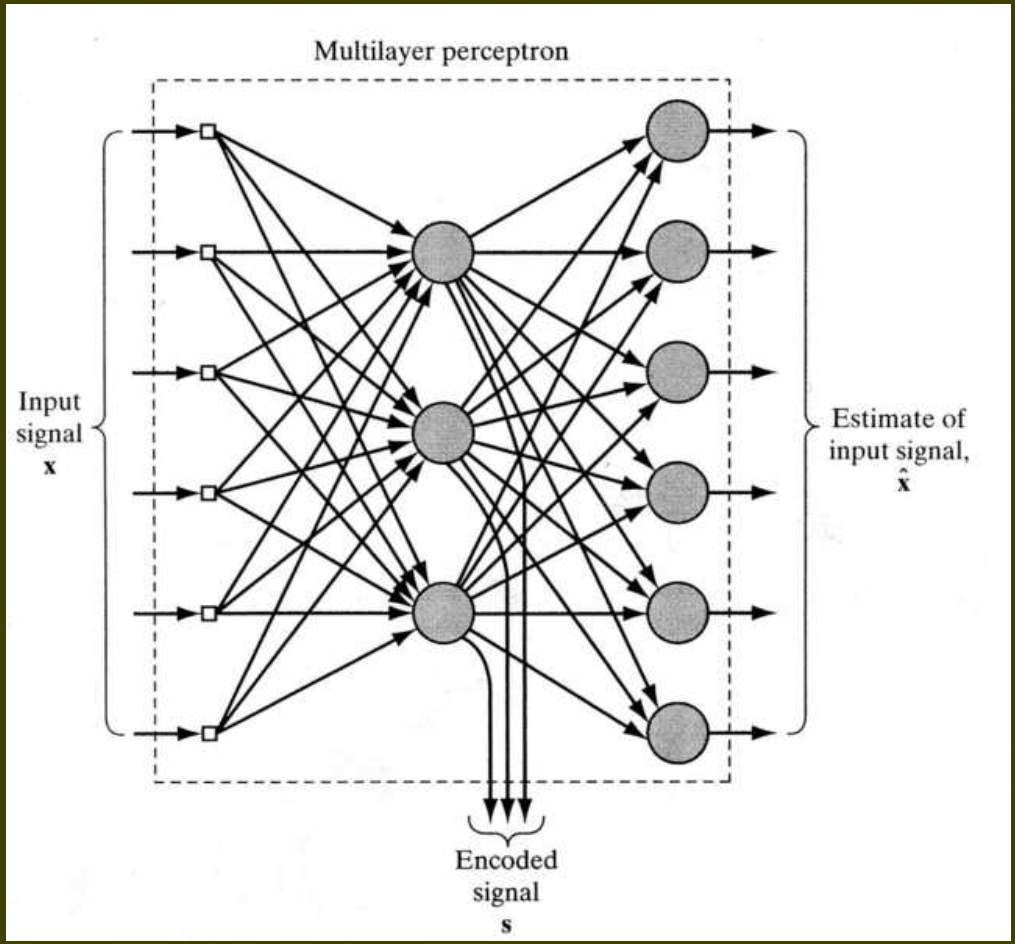

Figure 15: An MLP network acting as an autoencoder.

Nonlinear autoencoders

- Deep belief networks can be used for training nonlinear autoencoders.
- Autoencoder is a neural network (or mapping method) where the desired output is the input (data) vector itself.
- This is meaningful because in the middle of autoencoder, there is a data compressing layer having fewer neurons than in the input and output layers.
- Therefore, the output vector of an autoencoder network is usually an approximation of the input vector only.
- Comparing it with the input vector provides the error vector needed in training the autoencoder network.
- Figure 15 shows a simple example of an autoencoder.

- Traditional autoencoders have five layers: a hidden layer between the input layer and data compressing middle bottleneck layer.
- And similar extra hidden layer with many neurons between the middle bottleneck layer and output layer.
- They were trained using the backpropagation algorithm by minimizing the mean-square error.
- But this is difficult for multiple hidden layers with millions of parameters.
- Output vector of the middle bottleneck layer in autoencoders can be used for nonlinear data compression.
- The greedy learning algorithm for restricted Boltzmann machines can be used to pre-train autoencoders also for large problems.
- It performs a global search for a good, sensible region in the parameter space.

- The fine-tuning of model parameters is carried out using a variant of standard backpropagation (wake-sleep algorithm).
- Backpropagation is better at local fine-tuning of the model parameters than global search.
- So further training of the entire autoencoder using backpropagation will result in a good local optimum.
- Nonlinear autoencoders trained in this way perform considerably better than linear data compression methods such as PCA.
- Autoencoders must be regularized for preventing them to learn identity mapping.
- Instead of middle bottleneck layer, one can add noise to input vectors or put some of their components zero.
- Or one can impose sparsity by penalizing hidden unit activations so at or near 0.

- Or force encoder to have small derivatives with respect to inputs $\mathbf{x}$ (contractive constraint).
- Discrete inputs can be handled by using a cross-entropy or log-likelihood reconstruction criterion.
- Instead of a stack of Restricted Boltzmann Machines (RBMs), one can use a stack of autoencoders.
- The "Deep Learning and Bayesian Modeling" research group led by Assistant Prof. Tapani Raiko in our department uses autoencoders.
- Because they are easier to train than RBMs.
- See their homepage http://research.ics.aalto.fi/bayes/ for more information.

Convolutional networks

- Probably the most important single development that has increased the popularity of deep learning is convolutional networks.
- They allow processing of large digital images which is impossible using fully connected multilayer perceptron (MLP) networks.
- Because they would have millions of parameters to be learned.
- Our brief discussion of convolutional networks is based on Section 4.17 Convolutional Networks in the book S. Haykin, "Neural Networks and Learning Systems, 3rd ed.", Pearson, 2009.
- You can find this section on the "Materials" subpage of the home pages of our course.
- Convolutional networks were introduced by LeCun and Bengio already 2003.

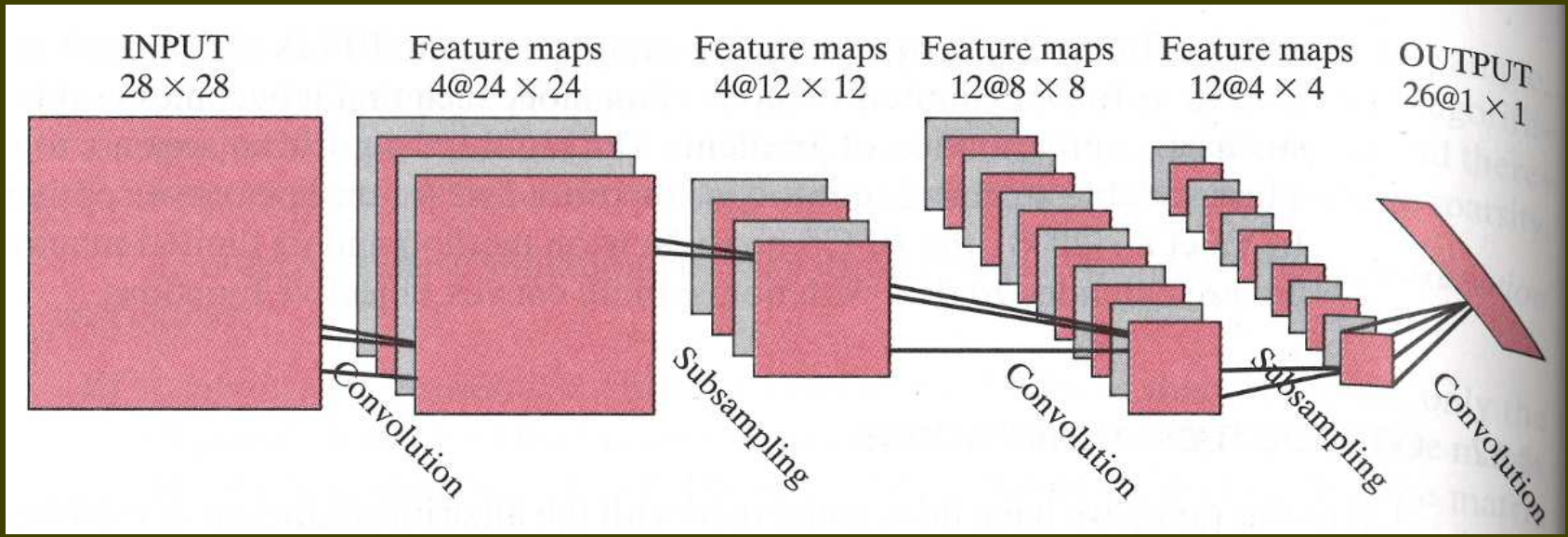

Figure 16: A convolutional network for classification of 28×28 digital images of handwritten numerals from the MNIST database.

- A convolutional network is a multilayer perceptron network designed to recognize two-dimensional shapes.
- It has a high degree of invariance to translation, scaling, skewing, and other forms of distortion.
- Obviously Figure 16 represents the convolutional network that LeCun and Bengio used in 2003.
- For classification of 28×28 digital images of handwritten numerals from the MNIST database.

Constraints in convolutive networks

1. Feature extraction. Each neuron takes its inputs from a local receptive field (local area) in the previous layer.
2. Feature mapping. Each computational layer of the network is composed of multiple feature maps.

- Individual neurons in the feature map are constrained to share the same set of synaptic weights.
- This structural constraint provides the beneficial effects of shift invariance and reduction in the number of free parameters.

3. Subsampling. Each convolutional layer is followed by a computational layer that performs local averaging and subsampling.

- All weights in all layers of a convolutional network are learned through training.
- Figure 16 shows the architecture of a convolutional network made up of an input layer, four hidden layers, and an output layer.
- The operation of these layers is described in more detail in Section 4.17 of Haykin's book (2009).

- The computational layers alternate between convolution and subsampling.
- At each layer the number of feature maps is increased while the spatial resolution is reduced compared with the previous layer.
- The MLP network shown in Figure 16 contains approximately 100,000 synaptic connections but only about 2,600 free parameters.
- This dramatic reduction is achieved through the use of weight sharing.
- The free parameters of the networks are learned using the stochastic mode of the backpropagation algorithm.

Conclusions

- Deep learning has become a hot topic in machine learning.
- Because can provide world record results in different classification and regression problems and datasets.
- Many corporations including Google, Microsoft, Nokia, etc. study it actively.
- Understanding deep learning well requires mathematical maturity and good knowledge of probabilistic modeling.
- Learning algorithms are complicated and good initialization is important.
- The field is developing quite rapidly with new structures and learning methods introduced every year.

References

1. I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning", MIT Press, 2016. Available at http://www.deeplearningbook.org/ .
2. R. Salakhutdinov, "Learning Deep Generative Models", doctoral thesis. http://www.mit.edu/∼rsalakhu/papers/Russ_thesis.pdf.
3. Web site on deep learning: http://deeplearning.net/.
4. G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets". Neural Computation, vol. 18, pp. 1527-1554, 2006.
5. R. Salakhutdinov and G. Hinton, "Learning a nonlinear embedding by preserving class neighbourhood structure". In Proc. of the 11th Int. Conf. on Artificial Intelligence and Statistics (AISTATS 2007), San Juan, Puerto Rico, March 2007, pp. 409–416.

6. R. Salakhutdinov and G. Hinton, "Deep Boltzmann Machines". In Proc. of the 12th Int. Conf. on Artificial Intelligence and Statistics (AISTATS 2009), Clearwater Beach, Florida, USA.

7. J. Goldberger et al., "Neighbourhood components analysis". In Advances in Neural Inf. Processing Systems 17 (Proc. of NIPS 2004), MIT Press, 2005.

8. S. Kaski and J. Peltonen, "Informative discriminant analysis". In T. Fawcett and N. Mishna (Eds.), Proc. of the 20th Int. Conf. on Machine Learning (ICML 2003), pp. 329–336, AAAI Press, Menlo Park, CA, 2003.

9. K. Cho, "Foundations and advances in deep learning". Doctoral thesis, Aalto University, March 2014. Available at http://publicationslist.org/kyunghyun.cho