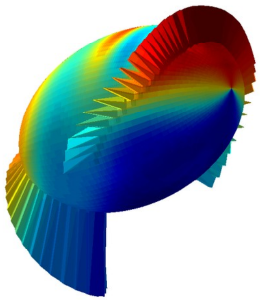
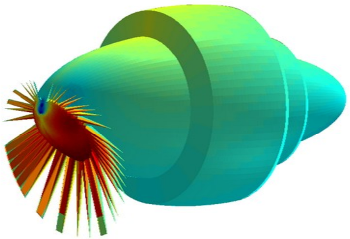
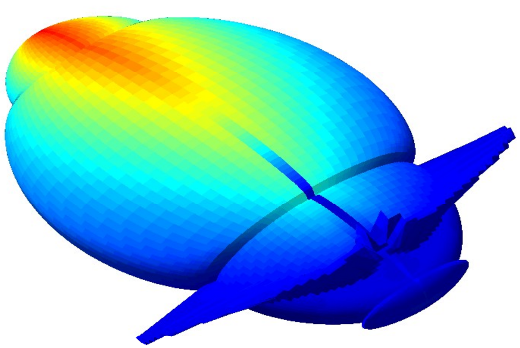
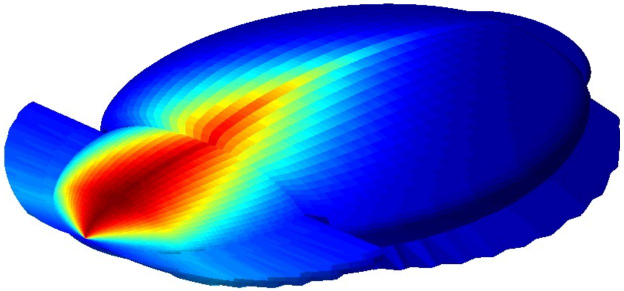




Lecture 7
Autoencoders

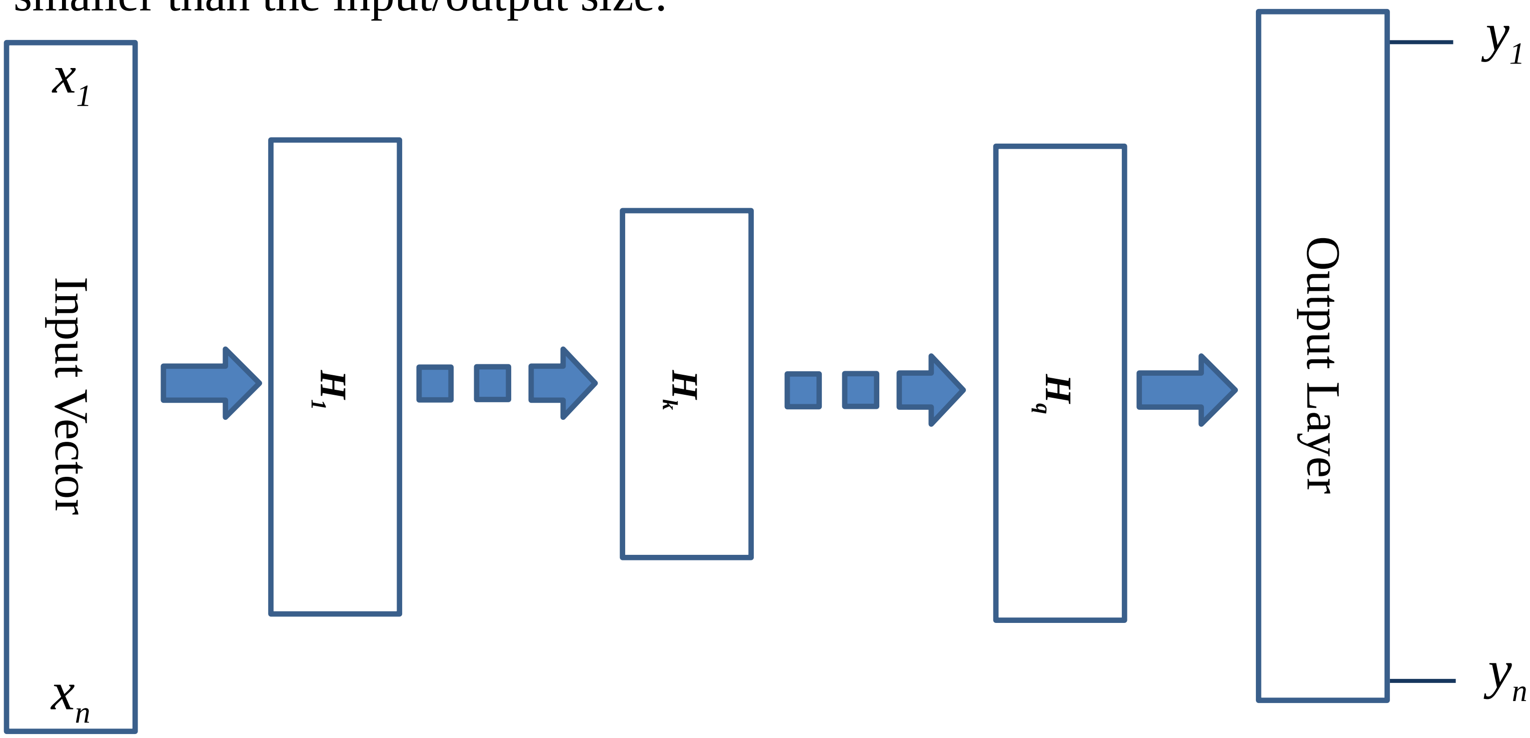




Autoencoder

A multi-layer feed-forward neural network that is trained to reproduce its input vector as output.

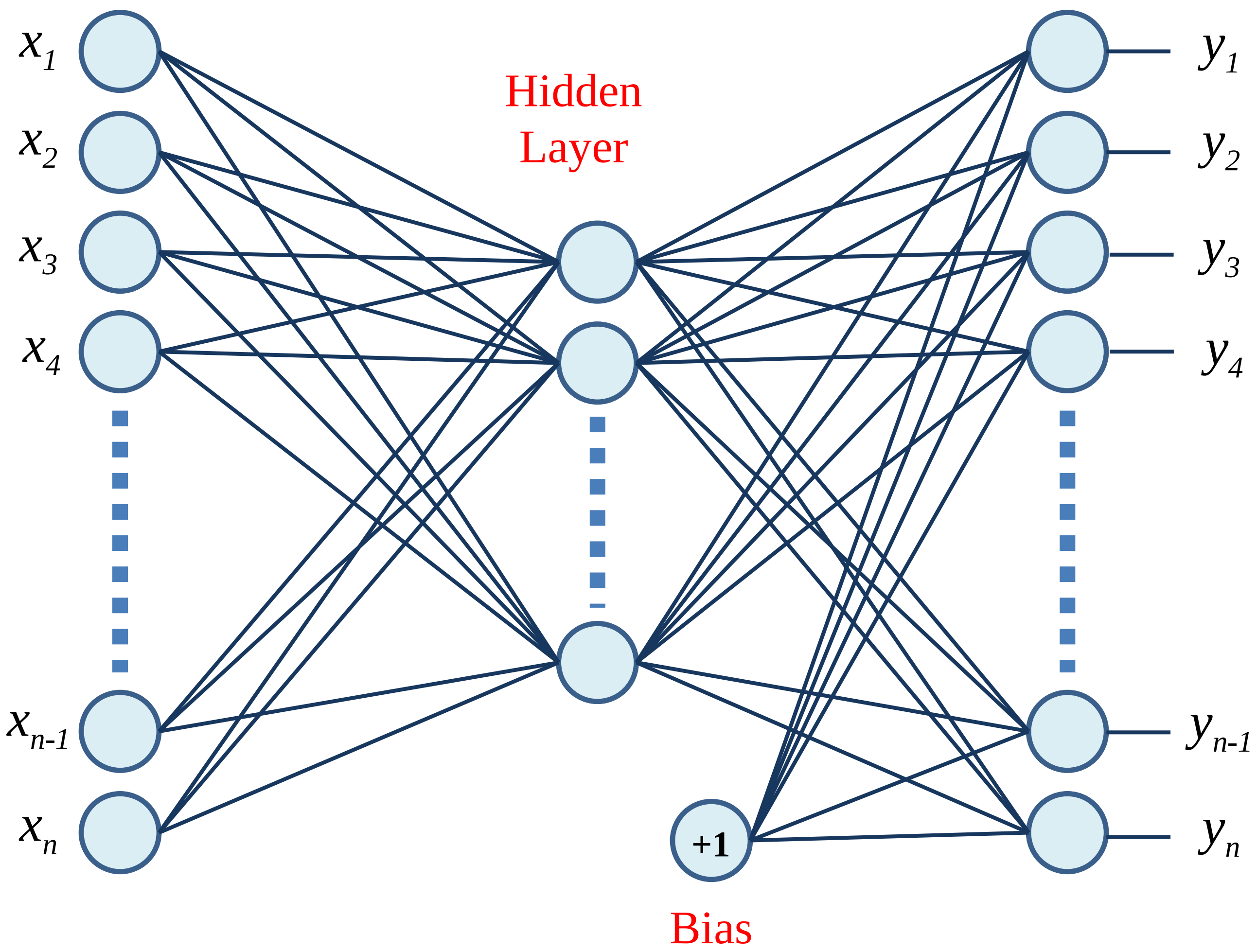
Autoencoders typically – but not always – use hidden layers that are smaller than the input/output size.



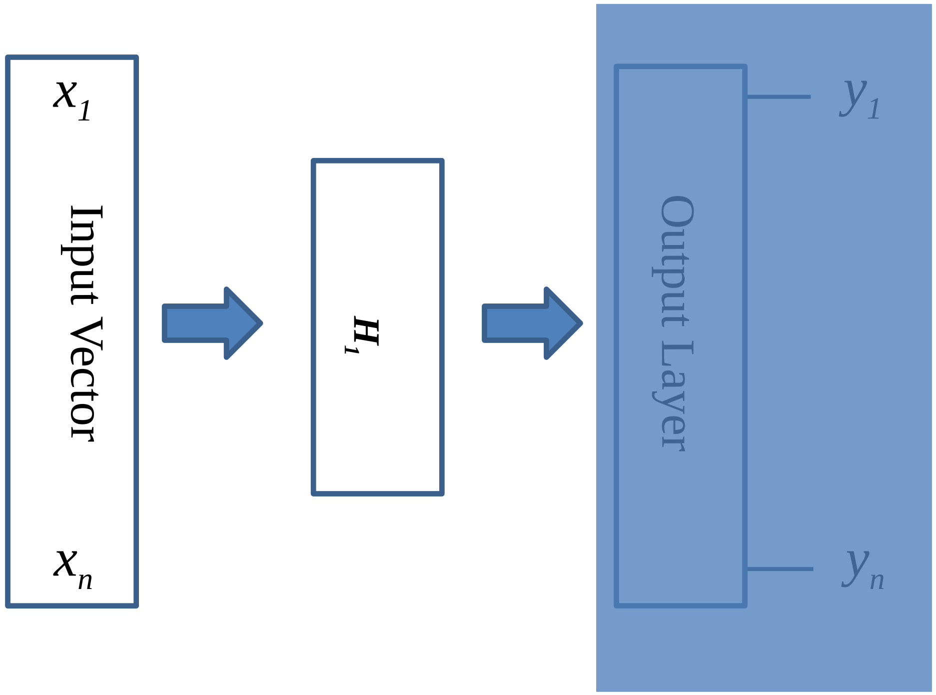
For a nice introduction, see:

http://www.subsubroutine.com/sub-subroutine/2015/7/15/how-neural-net-autoencoders-can-automatically-abstract-visual-features-from-handwriting

Shallow Autoencoder

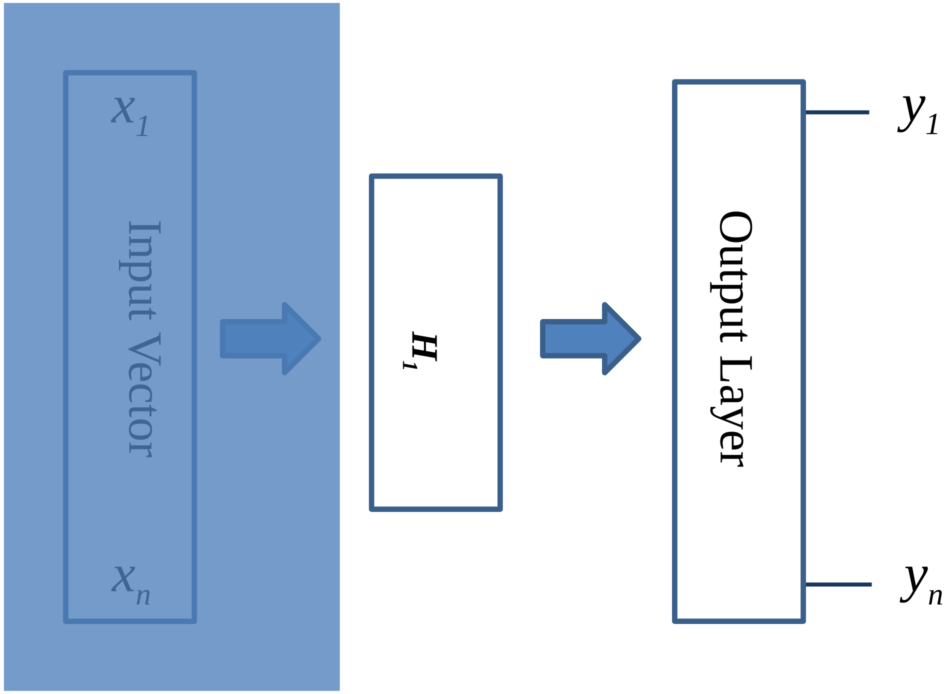


Basic Idea



Encoder Stage:
The hidden layer encodes the input into a new, more useful representation.

For example, the new representation could be:
- Lower-dimensional (compressed).
- More orthogonal (having less correlated features).
- More sparsely coded.



Decoder Stage:

The encoded representation in the hidden layer is decoded back to the original.

This requires that the encoded representation must retain all the information necessary to reconstruct the original

The hidden layer neurons define a new feature space for the original data.

If the hidden layer is narrower than the input/output layers, the learning process is forced to:

Compress the original data into fewer dimensions

↳ Squeeze out useless and redundant information

↳ Find a smaller set of features that are more significant and informative.

If the hidden layer activity is forced to be sparse, the learning process is forced to:

Find features that are more dissimilar from each other

↳ Make the features more specific to particular aspects of the input data

↳ Create a representation with more distinct and meaningful dimensions.

↳ Create more distinct representations for sufficiently different data points.

Latent Variables

Latent (hidden) variables or factors are a set of informative but unobservable variables that can explain the behavior of a set of observable variables.

Latent variables can serve many functions:

Acting as Basis Variables: Observables are represented as mixtures (e.g., linear combinations) of the latent variables,
e.g. principal components, eigenvectors, eigenfunctions, etc.

Explaining Complex Behavior: Complex, high-dimensional observable behavior can be explained in terms of structured interactions between a few latent variables, e.g., explaining texts in terms of topic mixtures, or human behavior in terms of personality types.

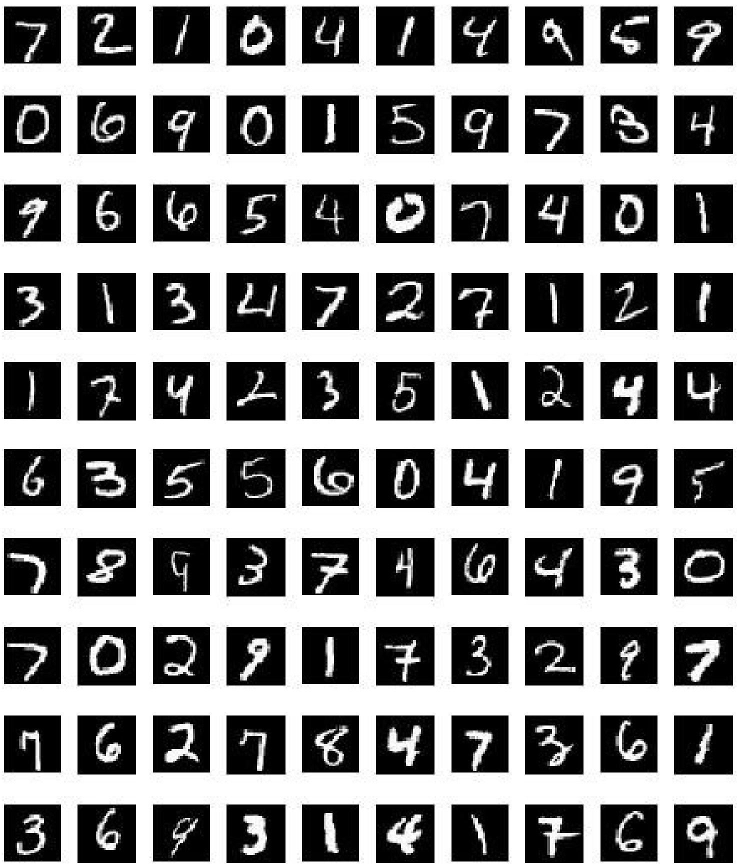
Explaining Statistical Dependencies: The statistical dependence between observables can explained by latent variables, e.g.

$$P(x,y|h) = P(x|h)P(y|h)$$

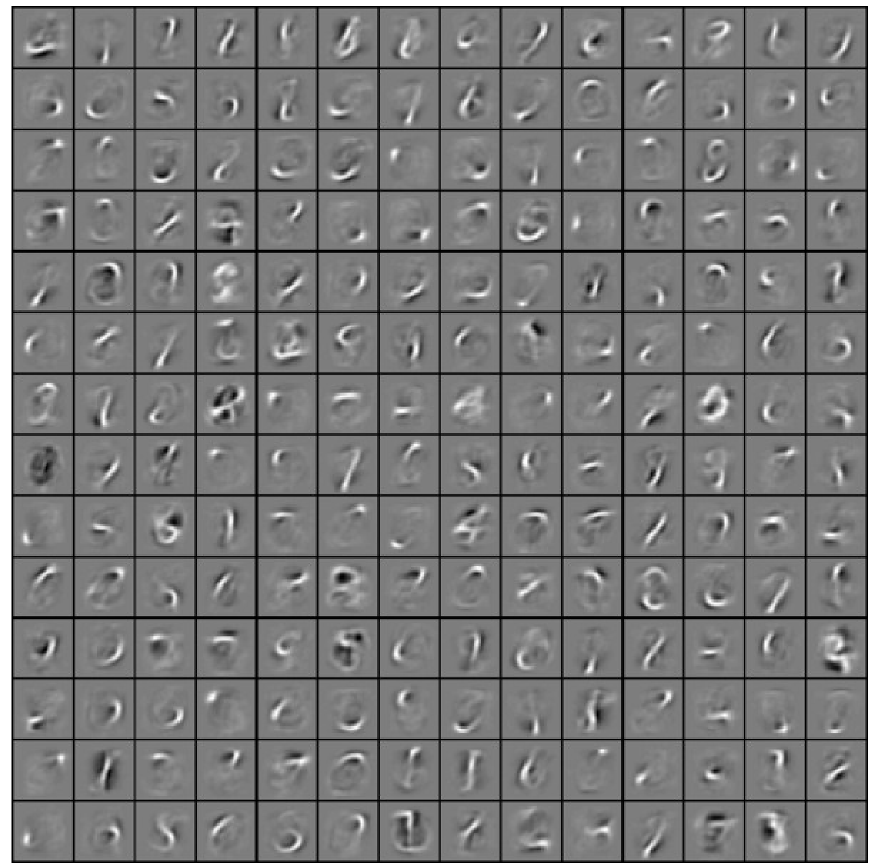
Autoencoders can discover useful ***latent variables***

Discovering Features with Autoencoders

MNIST Dataset (partial)



Features found by a Sparse Autoencoder



http://ufldl.stanford.edu/wiki/index.php/Exercise:Vectorization

Regularization

Weight Penalty: Forces better generalization by keeping the weights smaller.

$$J^q = \frac{1}{2}\sum_{i\in Outputs}\left(y_i^q - \hat{y}_i^q\right)^2 + \boxed{\frac{\lambda}{2}\sum_{L=1}^{N-1}\sum_{j\in Layer\ L}\sum_{i\in Layer\ L+1}\left(w_{ij}\right)^2}$$

N = number of layers

Weight Penalty

Sparse Coding: Forces hidden layer features to be more diverse by allowing each hidden neuron to be active only for a small fraction of data points.

$$\rho_j = \frac{1}{N}\sum_{q\in Training\ Set} h_j^q$$

Mean activity of hidden neuron j over all N data points

$$KL\left(\bar{\rho}\,\|\,\rho_j\right) = \bar{\rho}\log\frac{\bar{\rho}}{\rho_j} + (1-\bar{\rho})\log\frac{1-\bar{\rho}}{1-\rho_j}$$

Divergence of activity from desired $\bar{\rho}$

$$J_s^q = J^q + \beta\sum_{j\in Hidden\ Layer} KL\left(\bar{\rho}\,\|\,\rho_j\right)$$

Cost function with sparseness constraint

Dropout: Forces hidden layer features to be more diverse by disrupting co-adaptation

Weight Penalty: Forces better generalization by keeping the weights smaller.

$$J^q = \frac{1}{2}\sum_{i\in Outputs}\left(y_i^q - \hat{y}_i^q\right)^2 + \boxed{\frac{\lambda}{2}\sum_{L=1}^{D-1}\sum_{j\in Layer\ L}\sum_{i\in Layer\ L+1}\left(w_{ij}\right)^2}$$

D = number of layers

Weight Penalty

This leads to a small change in the calculation of the gradient function:

$$\frac{\partial J_{wd}^q}{\partial w_{ij}} = \frac{\partial J^q}{\partial w_{ij}} + \lambda w_{ij}$$

Typically, the *Lagrange multiplier* λ is set to a very small value between 0.0001 and 0.001

Appendix

Sparse Coding: Forces hidden layer features to be more diverse by allowing each hidden neuron to have only limited average activity.

$$\rho_j = \frac{1}{N} \sum_{q \in Training\ Set} h_j^q$$

Mean activity of hidden neuron j over all N data points

$$KL\left(\bar{\rho} \,||\, \rho_j\right) = \bar{\rho} \log \frac{\bar{\rho}}{\rho_j} + (1 - \bar{\rho}) \log \frac{1 - \bar{\rho}}{1 - \rho_j}$$

Divergence of activity from desired $\bar{\rho}$

$$J_s^q = J^q + \beta \sum_{j \in Hidden\ Layer} KL\left(\bar{\rho} \,||\, \rho_j\right)$$

Cost function with sparseness constraint

To see how this affects the gradient calculation, note that:

$$\frac{\partial \beta \sum_{l \in Hidden\ Layer} KL\left(\bar{\rho} \,||\, \rho_j\right)}{\partial w_{jk}} = \beta \sum_{l \in Hidden\ Layer} \frac{\partial KL\left(\bar{\rho} \,||\, \rho_l\right)}{\partial w_{jk}} = \beta \frac{\partial KL\left(\bar{\rho} \,||\, \rho_j\right)}{\partial w_{jk}}$$

Since only the $l = j$ term is affected by w_{jk}

$$\frac{\partial KL\left(\bar{\rho} \,||\, \rho_j\right)}{\partial w_{jk}} = \frac{\partial KL\left(\bar{\rho} \,||\, \rho_j\right)}{\partial \rho_j} \frac{\partial \rho_j}{\partial s_j^q} \frac{\partial s_j^q}{\partial w_{jk}}$$

Appendix

(a) $$\frac{\partial KL(\bar{\rho}\,\|\,\rho_j)}{\partial \rho_j} = \frac{\partial\left(\bar{\rho}\log\frac{\bar{\rho}}{\rho_j} + (1-\bar{\rho})\log\frac{1-\bar{\rho}}{1-\rho_j}\right)}{\partial \rho_j}$$

$$= \bar{\rho}\frac{\partial}{\partial \rho_j}\log\frac{\bar{\rho}}{\rho_j} + (1-\bar{\rho})\frac{\partial}{\partial \rho_j}\log\frac{1-\bar{\rho}}{1-\rho_j}$$

$$= \bar{\rho}\frac{1}{\bar{\rho}/\rho_j}\left(-\bar{\rho}\rho_j^{-2}\right) + (1-\bar{\rho})\frac{1}{(1-\bar{\rho})/(1-\rho_j)}\left(-(1-\bar{\rho})(1-\rho_j)^{-2}(-1)\right)$$

$$= -\frac{\bar{\rho}}{\rho_j} + \frac{1-\bar{\rho}}{1-\rho_j} \qquad (1)$$

(b) $$\frac{\partial \rho_j}{\partial s_j^q} = \frac{\partial}{\partial s_j^q}\frac{1}{N}\sum_{u\in Training\ Set} h_j^u$$

$$= \frac{1}{N}\sum_{u\in Training\ Set}\frac{\partial h_j^u}{\partial s_j^q} = \frac{1}{N}\frac{\partial h_j^q}{\partial s_j^q} = \frac{1}{N}f'\left(s_j^q\right) \qquad (2)$$

(c) $$\frac{\partial s_j^q}{\partial w_{jk}} = x_k^q \qquad (3)$$

Appendix

(1), (2) and (3) give: $$\frac{\partial KL(\bar{\rho}\,\|\,\rho_j)}{\partial w_{jk}} = \frac{\partial KL(\bar{\rho}\,\|\,\rho_j)}{\partial \rho_j}\frac{\partial \rho_j}{\partial s_j^q}\frac{\partial s_j^q}{\partial w_{jk}}$$

$$= \left(-\frac{\bar{\rho}}{\rho_j} + \frac{1-\bar{\rho}}{1-\rho_j}\right)\frac{1}{N} f'(s_j^q)\, x_k^q$$

Recall that in basic backpropagation, we had: $$\frac{\partial J^q}{\partial w_{jk}} = -\underbrace{\left[f_j'(s_j^q) \sum_i w_{ij}\delta_i^q \right]}_{\delta_j^q} x_k^q$$

Now we have: $$\frac{\partial J_s^q}{\partial w_{jk}} = -\left[f_j'(s_j^q) \sum_i w_{ij}\delta_i^q \right] x_k^q + \frac{\beta}{N}\left(-\frac{\bar{\rho}}{\rho_j} + \frac{1-\bar{\rho}}{1-\rho_j}\right) f'(s_j^q)\, x_k^q$$

$$= -\underbrace{\left[\sum_i w_{ij}\delta_i^q - \gamma\left(\frac{1-\bar{\rho}}{1-\rho_j} - \frac{\bar{\rho}}{\rho_j}\right)\right] f_j'(s_j^q)}_{\delta_j^q}\, x_k^q$$

$$\gamma \equiv \frac{\beta}{N}$$

Appendix

Then, as before:

$$\frac{\partial J_s^q}{\partial w_{jk}} = -\delta_j^q x_k^q$$

$$\boxed{\Delta w_{jk} = \eta \delta_j^q x_k^q = \eta \left[\sum_i w_{ij} \delta_i^q - \gamma \left(\frac{1-\bar{\rho}}{1-\rho_j} - \frac{\bar{\rho}}{\rho_j} \right) \right] f_j'(s_j^q)\, x_k^q}$$

Note that

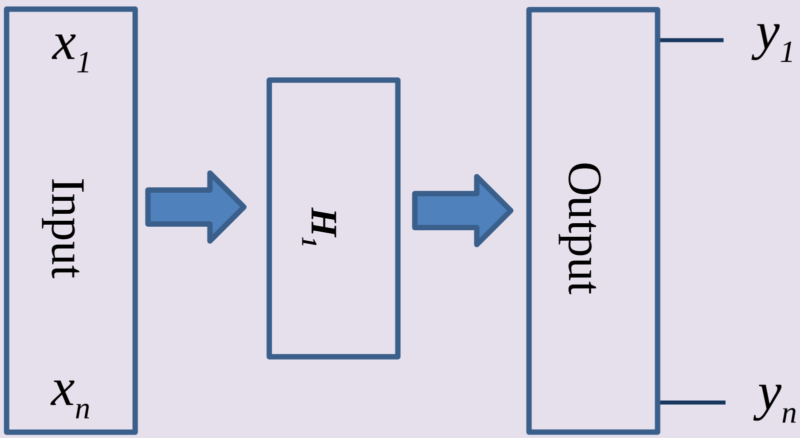
$$\rho_j > \bar{\rho} \Rightarrow -\gamma \left(\frac{1-\bar{\rho}}{1-\rho_j} - \frac{\bar{\rho}}{\rho_j} \right) < 0 \longrightarrow w_{jk} \text{ decreases}$$

$$\rho_j < \bar{\rho} \Rightarrow -\gamma \left(\frac{1-\bar{\rho}}{1-\rho_j} - \frac{\bar{\rho}}{\rho_j} \right) > 0 \longrightarrow w_{jk} \text{ increases}$$

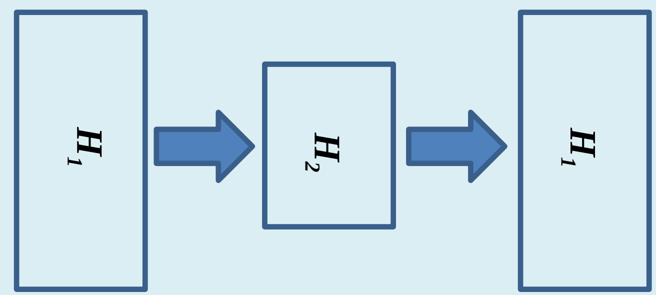
Typically, γ is set to a value in the range of 1-5, and $\bar{\rho}$ in the range of 0.01-0.05.

Appendix

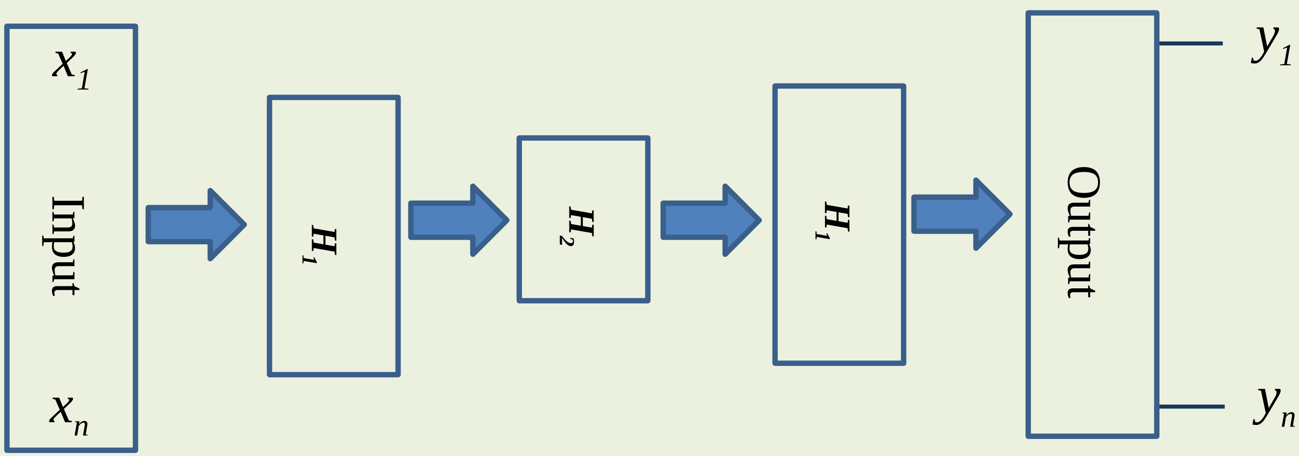
Building Multi-Stage Autoencoders



Stage 1

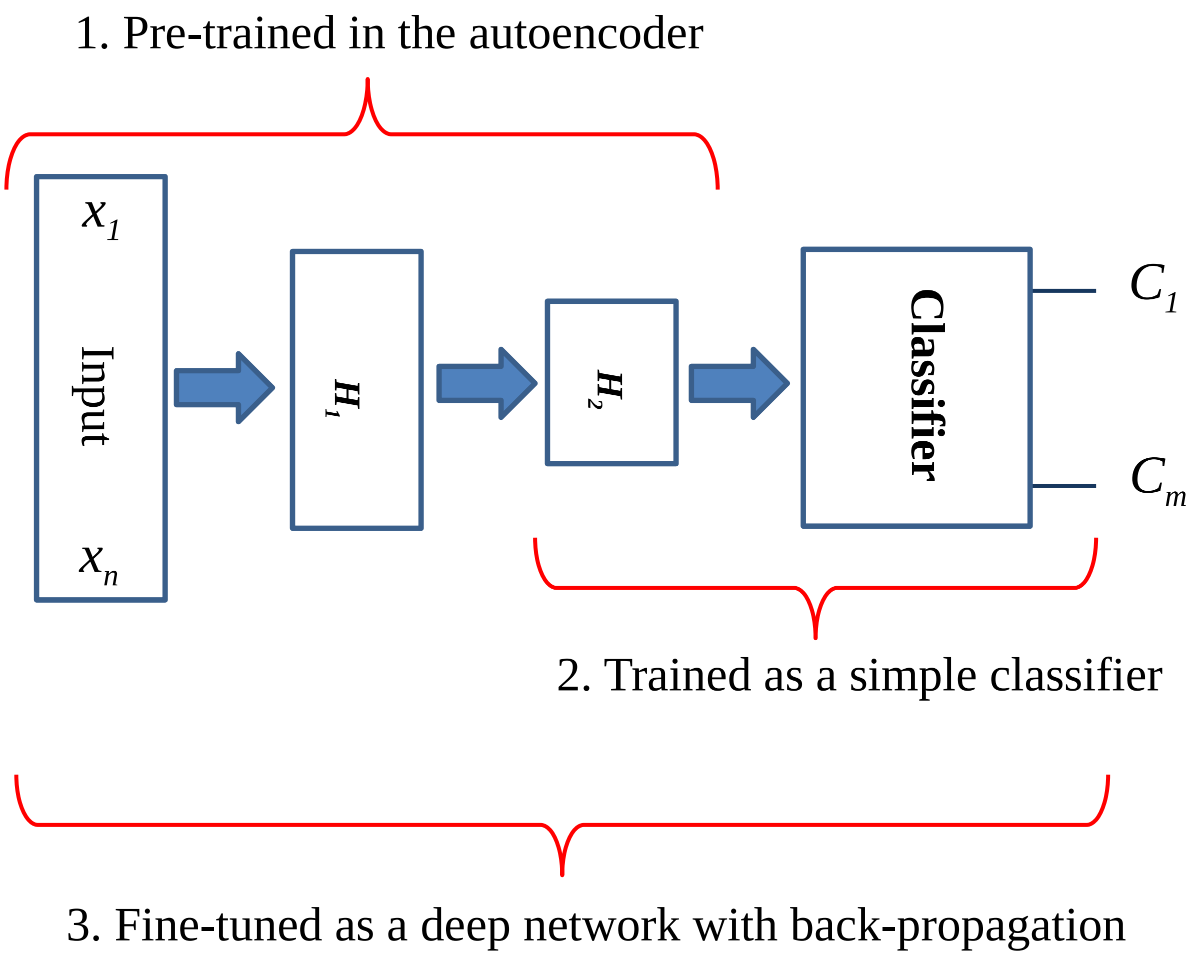


Stage 2



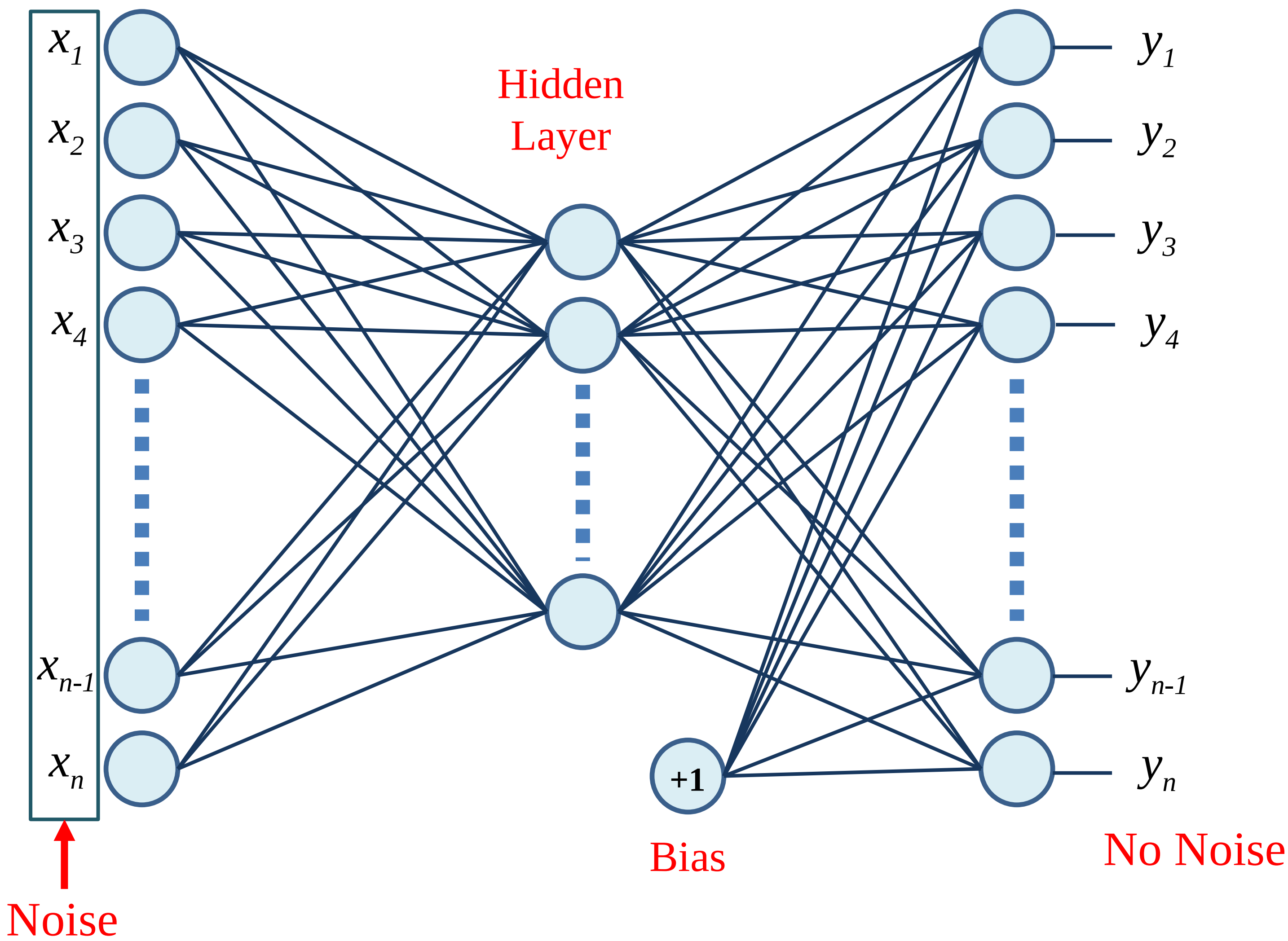
Final Network – possibly with some further end-to-end BP training

Autoencoder-Based Deep Classifier



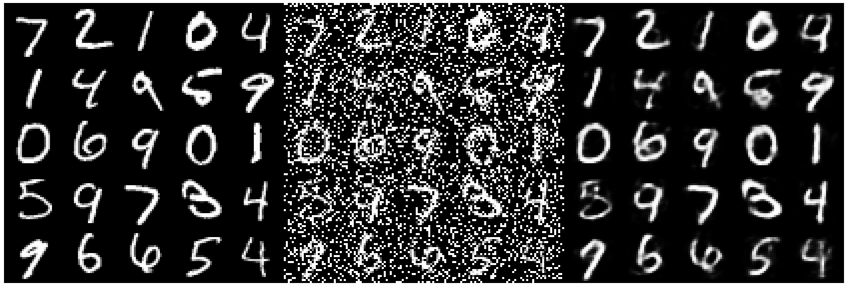
The classifier could be a layer of neurons (LMS), a multi-layer network, a softmax classifier, etc.

Denoising Autoencoder



After Learning:

- Can be used to clean up new noisy inputs.



From: https://towardsdatascience.com/denoising-autoencoders-explained-dbb82467fc2

- The hidden layer neurons can be used as robust features for classification, etc.

Other Autoencoder Models

Convolutional autoencoders use convolutional networks to encode images and deconvolution to decode them.

Variational autoencoders learn a ***stochastic latent space*** through the encoder and sample from it to ***generate*** reconstructions.

We may look at these later in the semester if there is time.