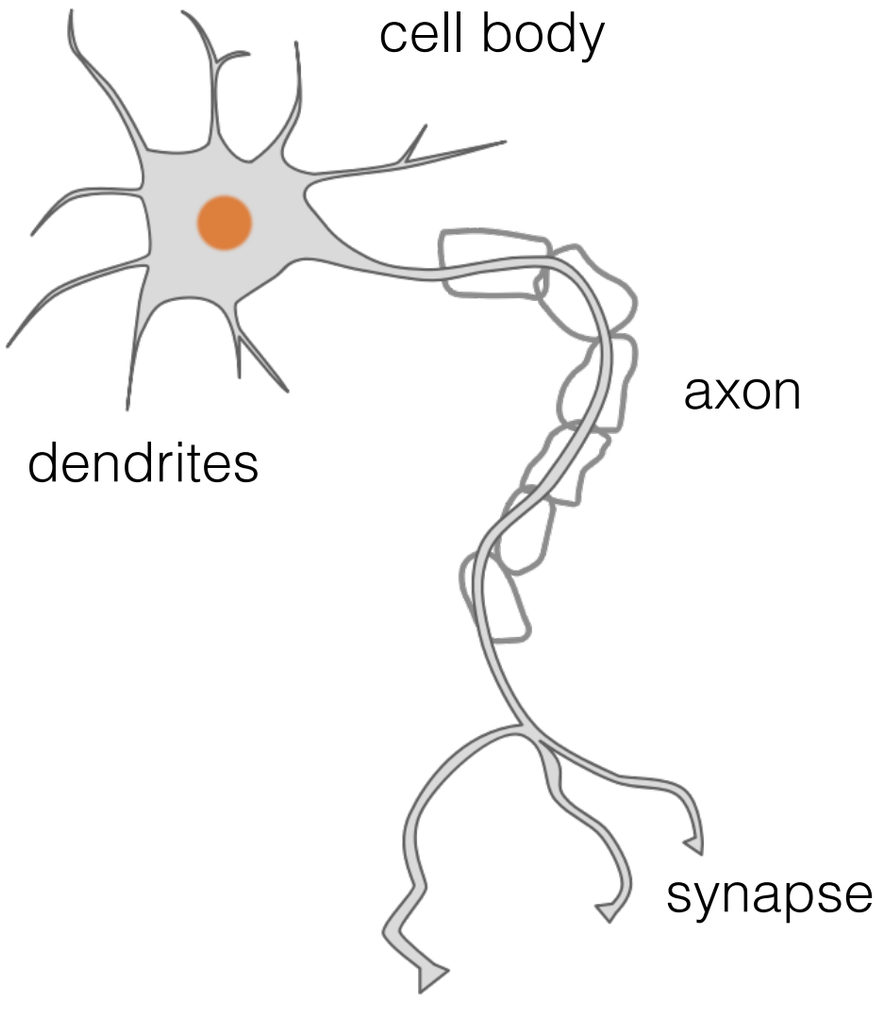
Ch11 Neural networks

MATH 6312
Department of mathematics, UTA

ESL 11.3-5, 11.7

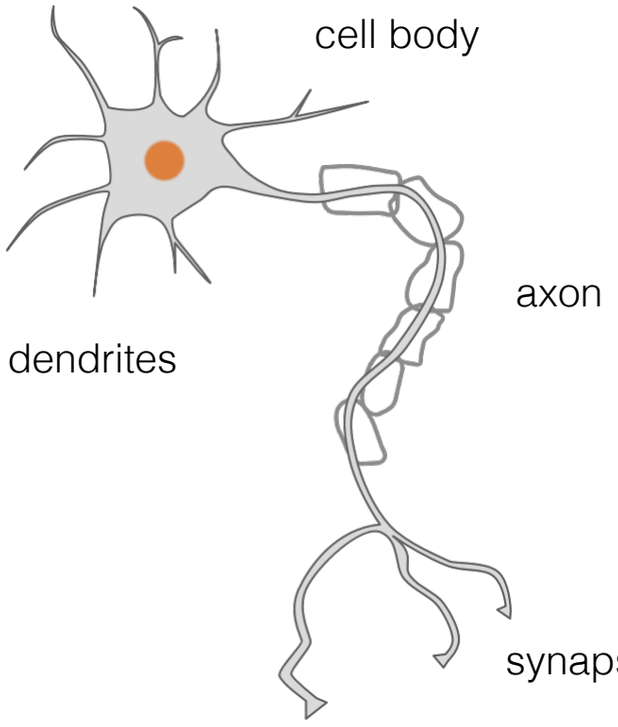
Optional reading: ESL 11.1-2

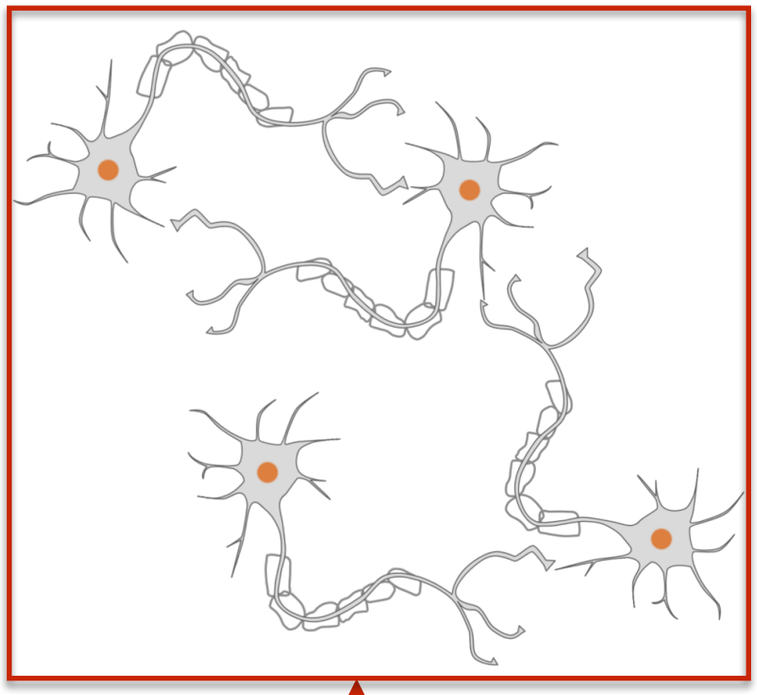
Neuron

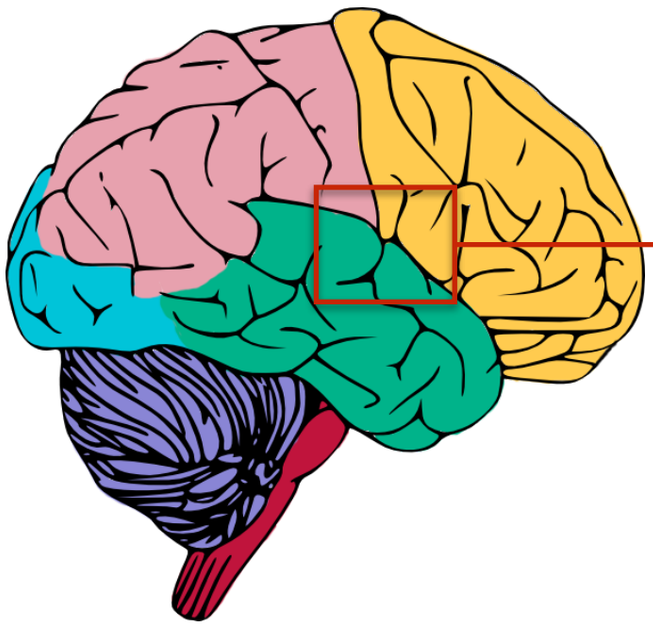


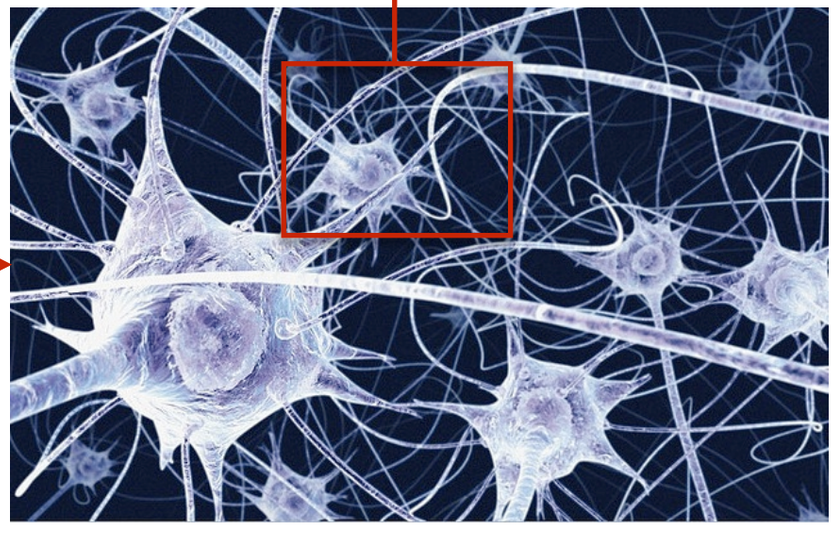
- Neurons are specialized cells of the nervous system that transmit signals throughout the body.
- Dendrites bring information to the cell body and axons take information away from the cell body.
- Information from one neuron flows to another neuron across a synapse.

Anatomy of a neuron





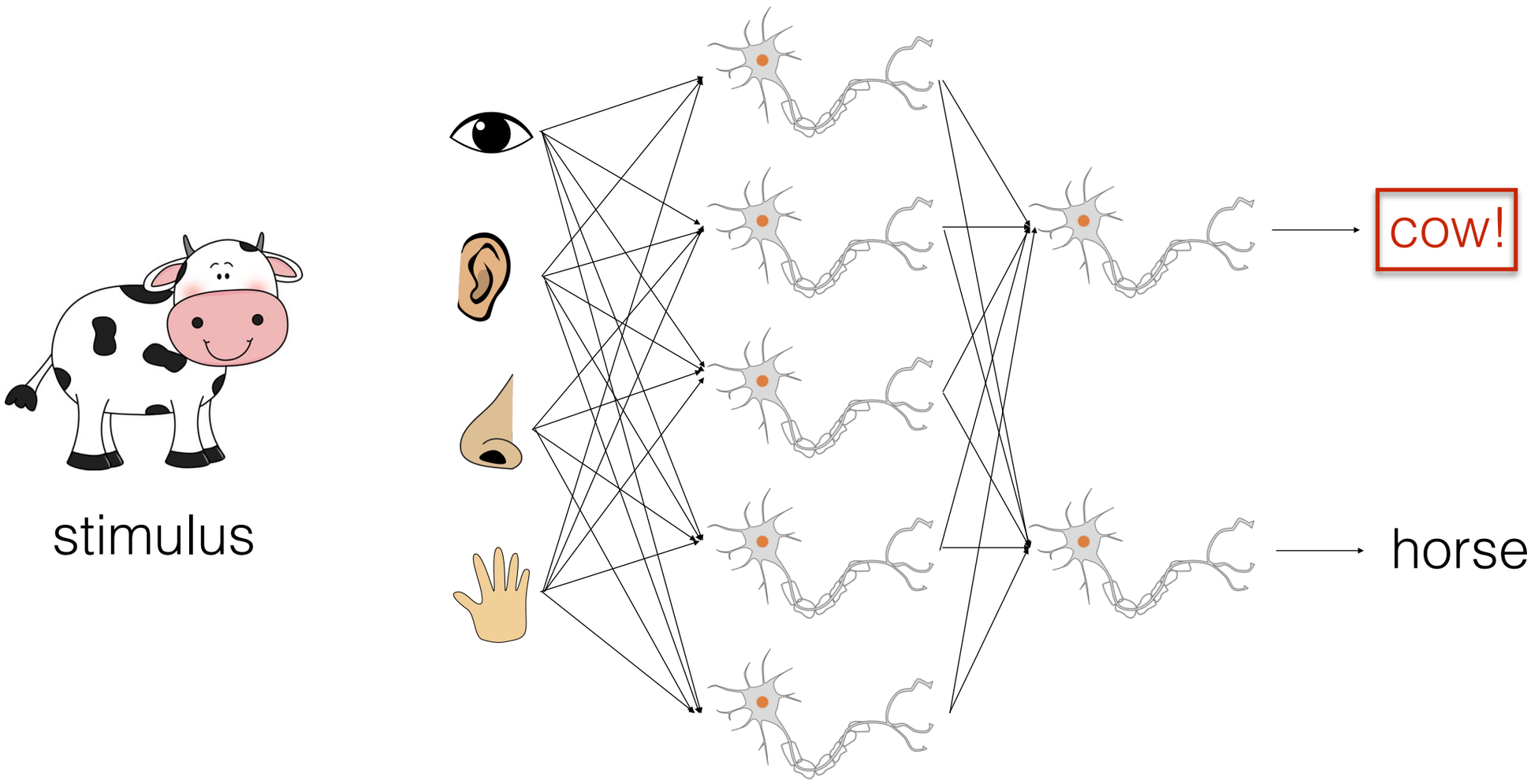




Biological neural networks

Biological neural networks

Neural Network is the representation of brain's learning approach



Some neurons are activated and pass the signal forward.

Artificial neuron

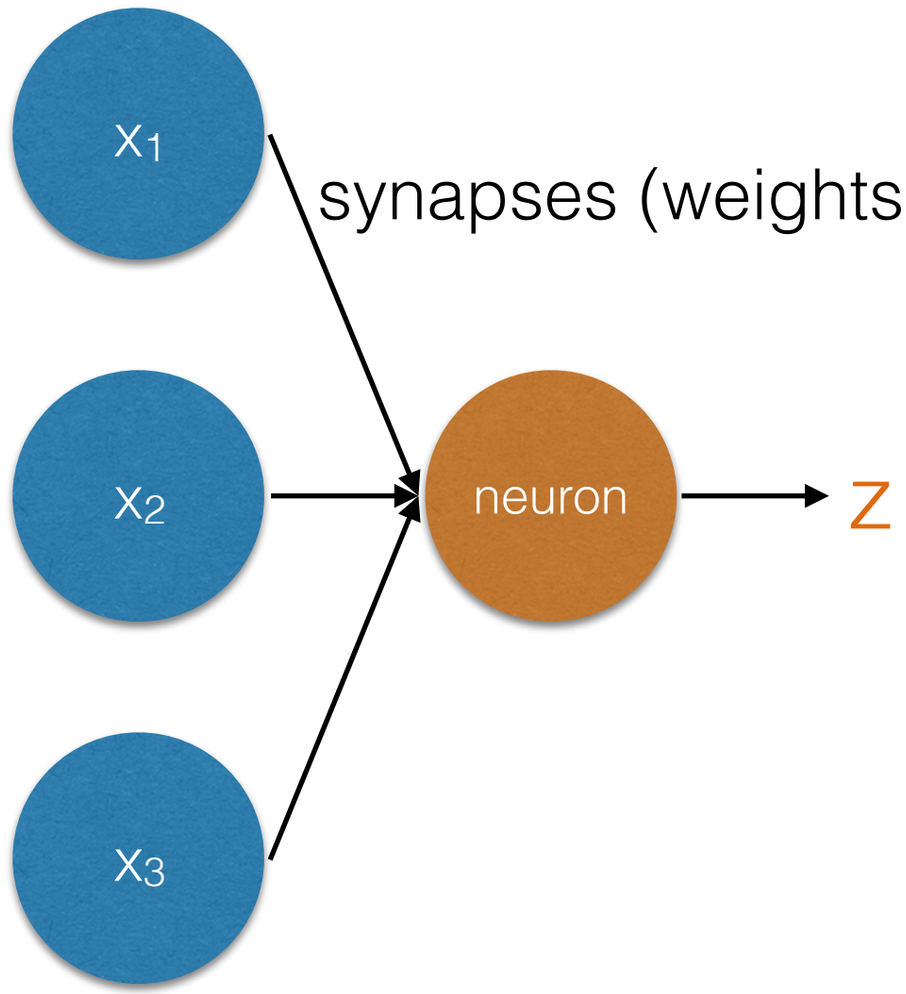
- Perceptron (artificial neuron) is a mathematical model of a biological neuron

- Activation of neuron: value computed by a neuron model

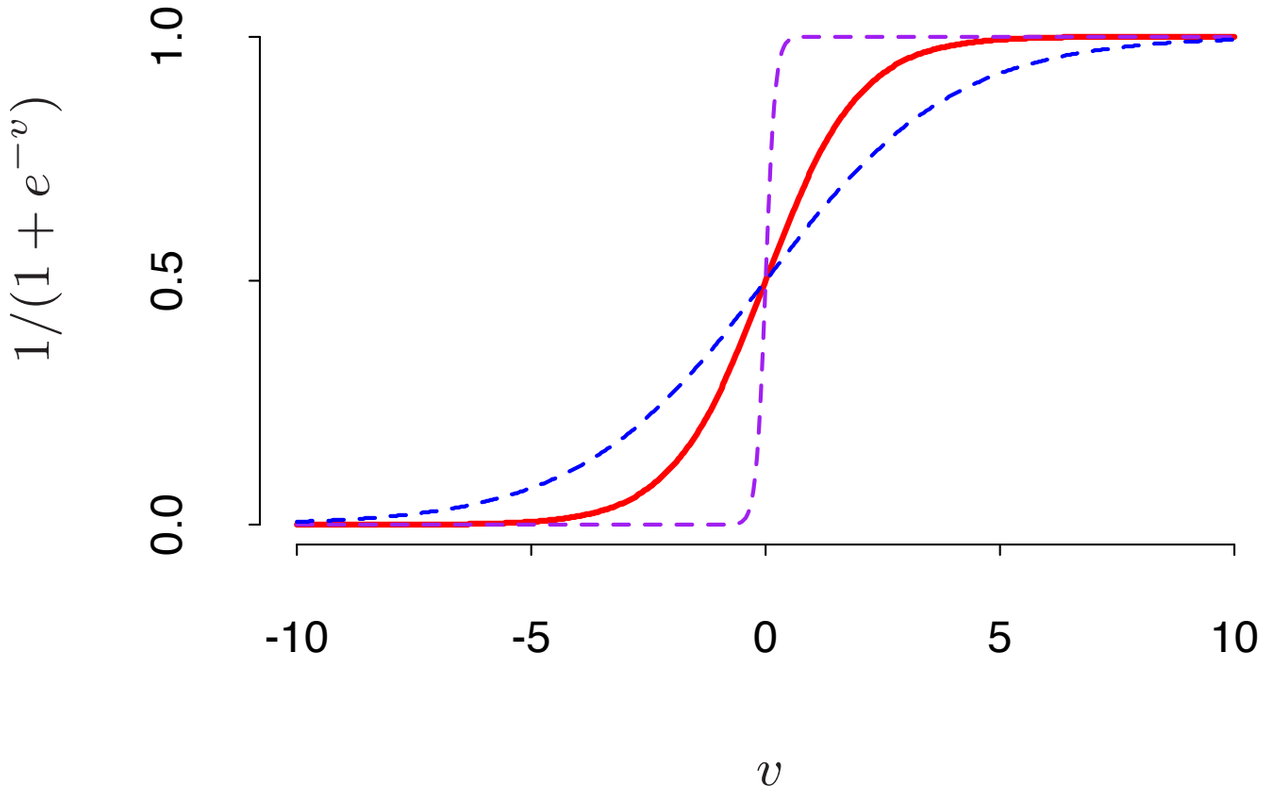
$$z = \sigma(\alpha_0 + \alpha' x)$$

- input vector: $x = (x_1, x_2, x_3)$, weights $\alpha = (\alpha_1, \alpha_2, \alpha_3)$ and α_0 (bias parameter).

- Activation function σ is typically chosen to be the sigmoid (step and radial-basis function are commonly used).



Sigmoid function

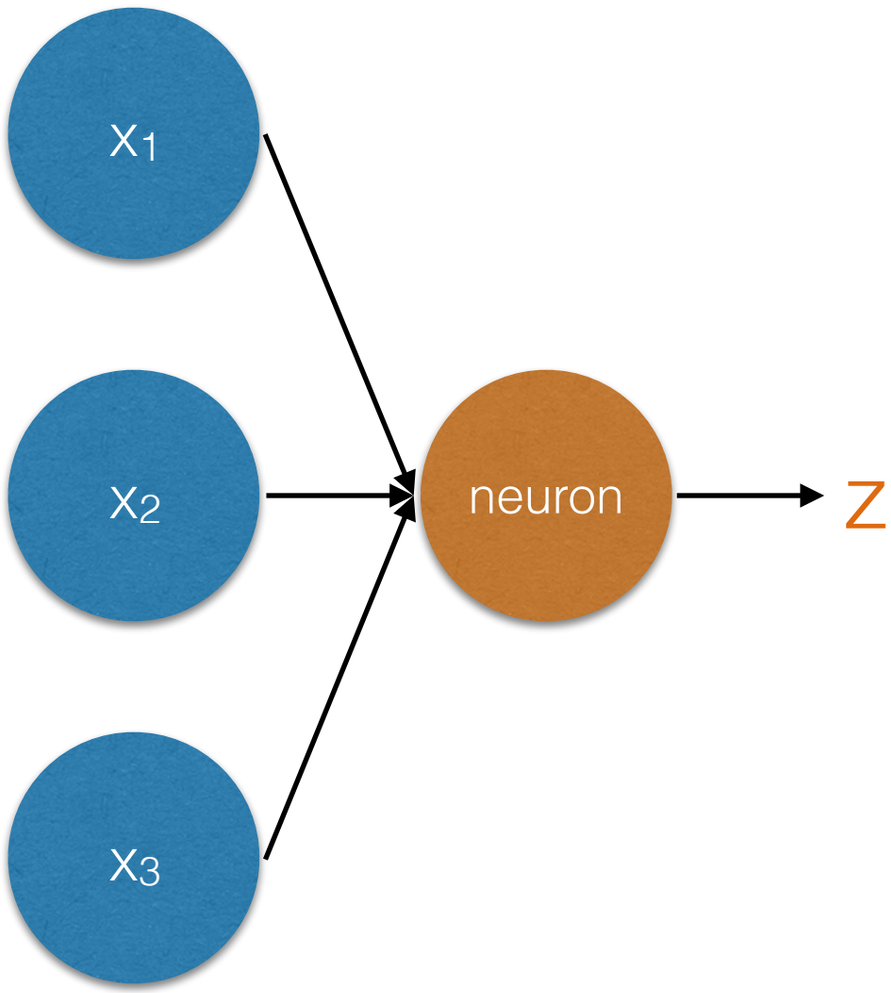


- ESL 11.3: Plot of sigmoid function σ(s·v)=1/[1+exp(-s·v)]
- s=1 (red curve), s=1/2 (blue curve) and s=10 (purple curve).
- Scale parameter s controls the activation rate: large s amounts to a hard activation at v = 0.

Sigmoidal perceptrons

- With the sigmoid activation function, the signal neuron model can be written as

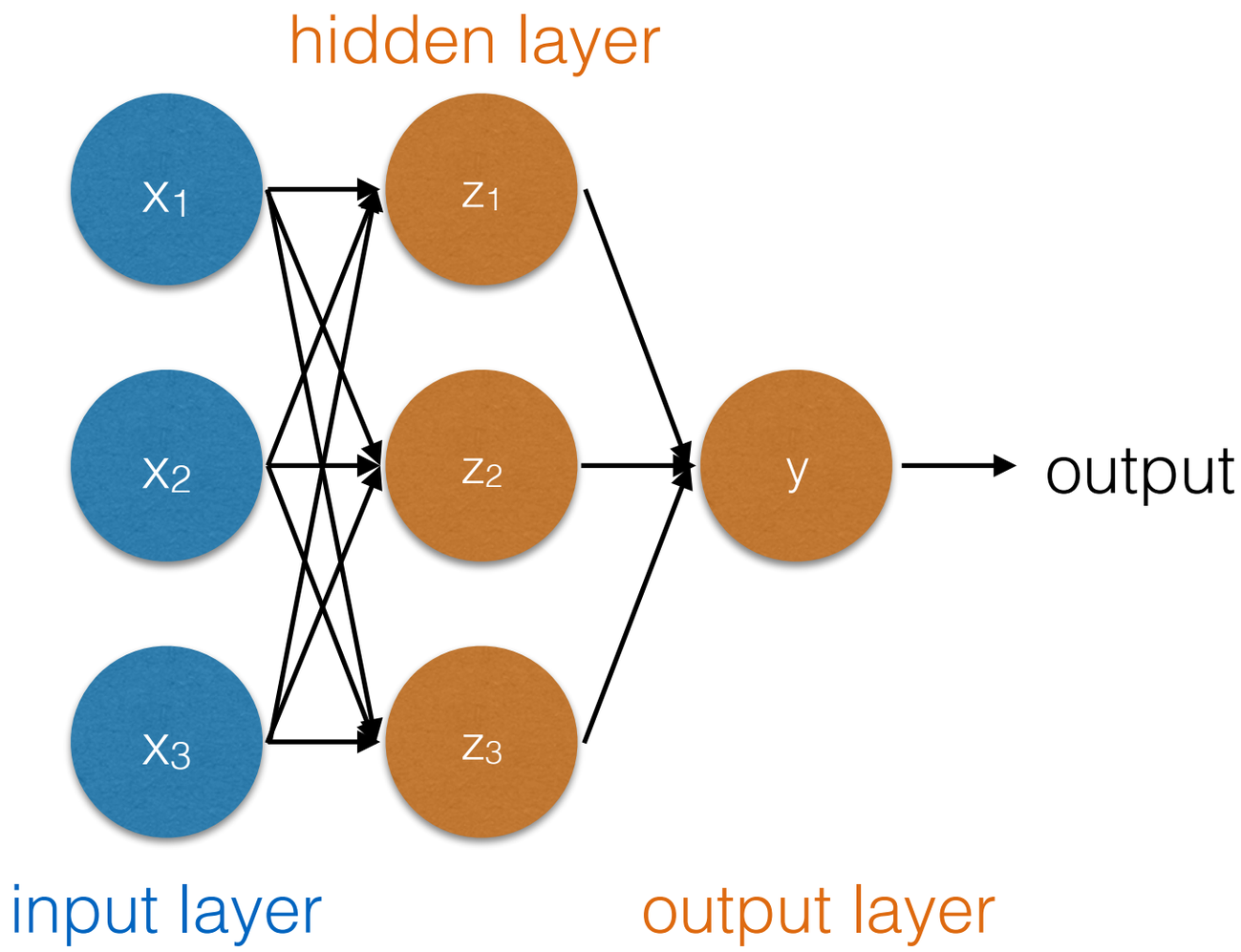
$$z = \frac{\exp(\alpha_0 + \alpha' x)}{1+\exp(\alpha_0 + \alpha' x)} = \frac{\exp(\alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3)}{1+\exp(\alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3)}$$



- Rate of activation of the sigmoid depends on the norm of α
  - if ||α|| is very small, the unit (model) will operate in the linear part of its activation function.
- Q. does this model look familiar?

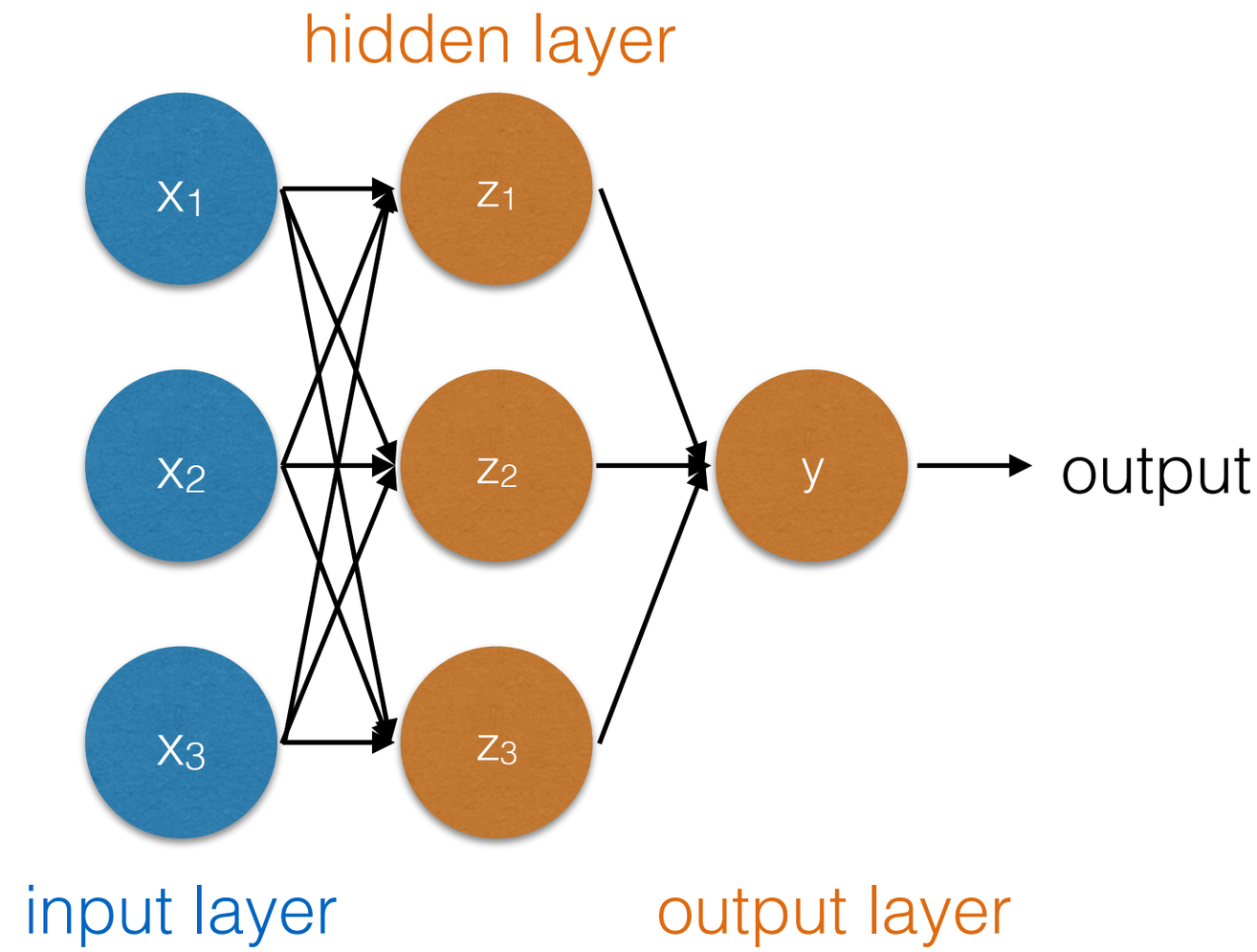
Artificial neural networks

- Neural networks is a combination of many signal neuron models.
- z_1, z_2, z_3: hidden unit (not directly observed).
- Connections represent weights (synapses).
- The model with one hidden layer is called vanilla neural net, single hidden layer back-propagation network, or single layer perceptron.
- Neural networks can have more than one hidden layer.



Artificial neural networks

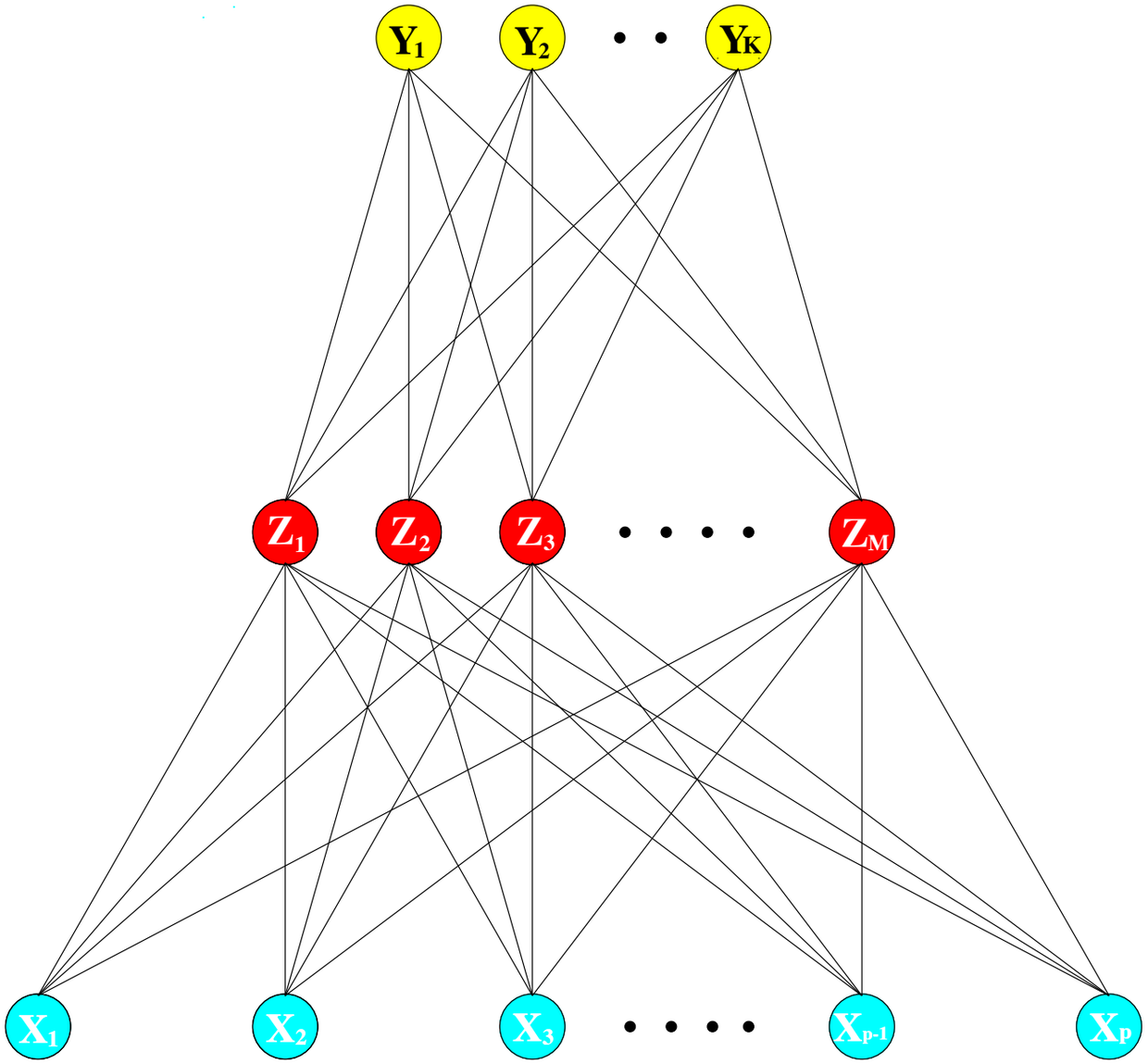
- Activation of hidden unit is computed based on the linear combination of inputs.
- Value in the output layer is computed by the linear combination of hidden units using some activation function.
- Feed-forward neural network: connections between the units do not form a cycle.



- If we choose the identity activation function, then the entire model collapses to a linear model in the inputs

- A neural network can be thought of as a nonlinear generalization of the linear model, both for regression and classification.

- By introducing the nonlinear activation function, it greatly enlarges the class of linear models.



$$Z = (Z_1, Z_2, \ldots, Z_M), \text{ and } T = (T_1, T_2, \ldots, T_K)$$

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X),\ m = 1, \ldots, M,$$
$$T_k = \beta_{0k} + \beta_k^T Z,\ k = 1, \ldots, K,$$
$$f_k(X) = g_k(T),\ k = 1, \ldots, K,$$

output function:
softmax activation function

$$g_k(T) = \frac{e^{T_k}}{\sum_{\ell=1}^{K} e^{T_\ell}}$$

for regression, $G_k(T) = T_k$
(identity activation function)

- ESL 11.2: Schematic of a single hidden layer, feed-forward neural network for K-class classification

- K units at the top, with the k*th* unit modeling the probability of class k.

Fitting neural network

- Unknown parameter (θ) in neural networks called weights. We seek weights that make the model fit the training data well:

$\{\alpha_{0m}, \alpha_m;\ m = 1, 2, \ldots, M\}$ $M(p+1)$ weights, $\quad Z_m = \sigma(\alpha_{0m} + \alpha_m^T X),\ m = 1, \ldots, M,$

$\{\beta_{0k}, \beta_k;\ k = 1, 2, \ldots, K\}$ $K(M+1)$ weights. $\quad T_k = \beta_{0k} + \beta_k^T Z,\ k = 1, \ldots, K,$

- Error measures
  - regression: squared errors $\quad R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2.$
  - classification: squared errors or cross-entropy (deviance)

$$R(\theta) = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log f_k(x_i),$$

Gradient descent

- Gradient descent is a generic method to minimizing the error function R(θ).

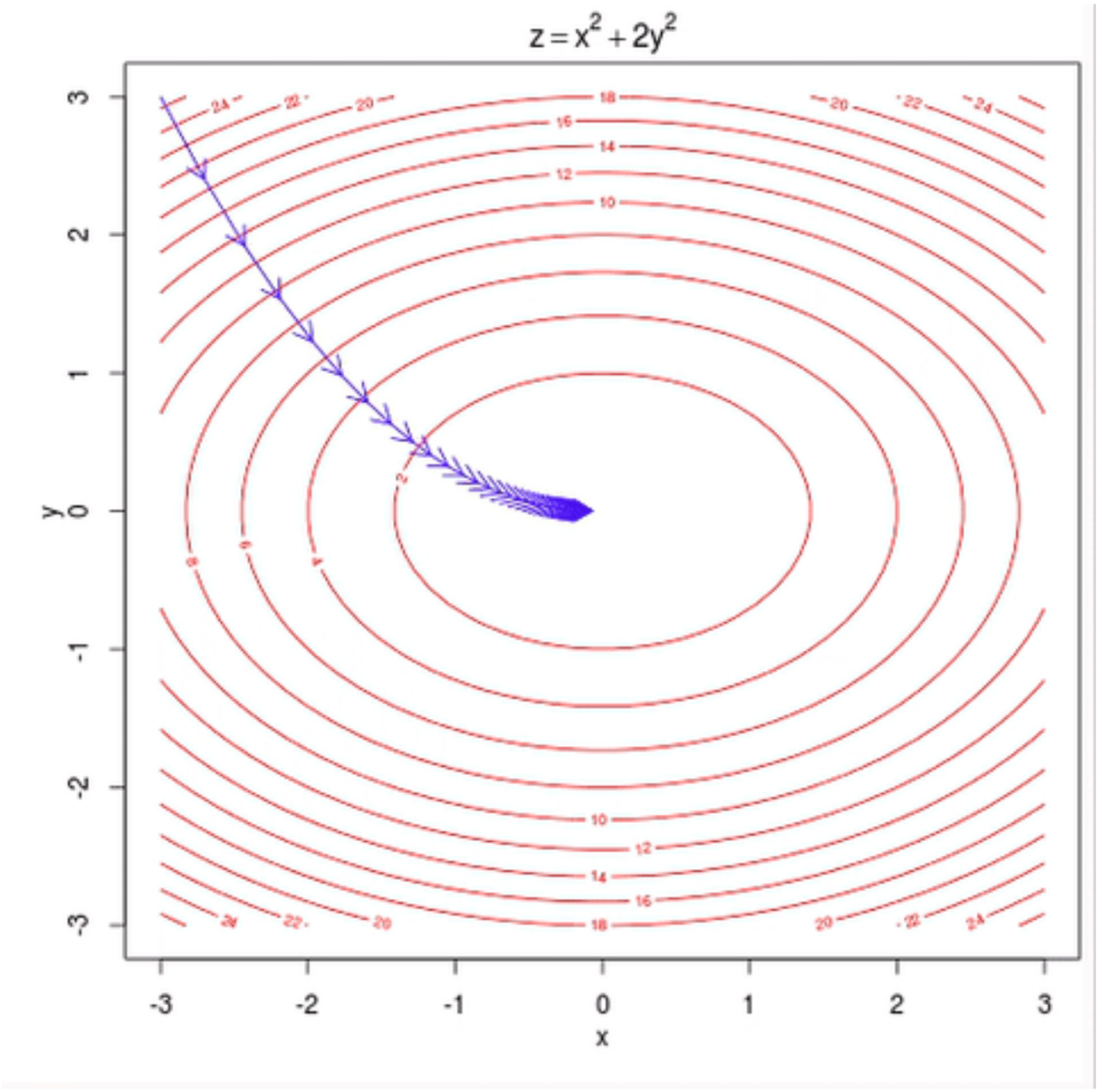
- It updates the value of θ at each iteration by (γ: learning rate)

$$\theta^{new} = \theta^{old} - \gamma \cdot \nabla R(\theta^{old})$$

- In the neural network, gradient can be easily derived using the chain rule for differentiation.

- The Newton-Raphson method is not attractive, because the Hessian matrix can be very large.

Toy example: gradient descent in 2D



Images Credit: http://vis.supstat.com/2013/03/gradient-descent-algorithm-with-r/

Schematic of gradient descent for neural network (back-propagation)

error function

$$R(\theta) \equiv \sum_{i=1}^{N} R_i$$
$$= \sum_{i=1}^{N}\sum_{k=1}^{K}(y_{ik} - f_k(x_i))^2,$$

initial guess

gradients

$$\frac{\partial R_i}{\partial \beta_{km}} = -2(y_{ik} - f_k(x_i))g_k'(\beta_k^T z_i)z_{mi},$$
$$\frac{\partial R_i}{\partial \alpha_{m\ell}} = -\sum_{k=1}^{K} 2(y_{ik} - f_k(x_i))g_k'(\beta_k^T z_i)\beta_{km}\sigma'(\alpha_m^T x_i)x_{i\ell}.$$

gradient descent update

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}},$$
$$\alpha_{m\ell}^{(r+1)} = \alpha_{m\ell}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{m\ell}^{(r)}},$$

γ_r: learning rate

- Gradients can be rewritten as

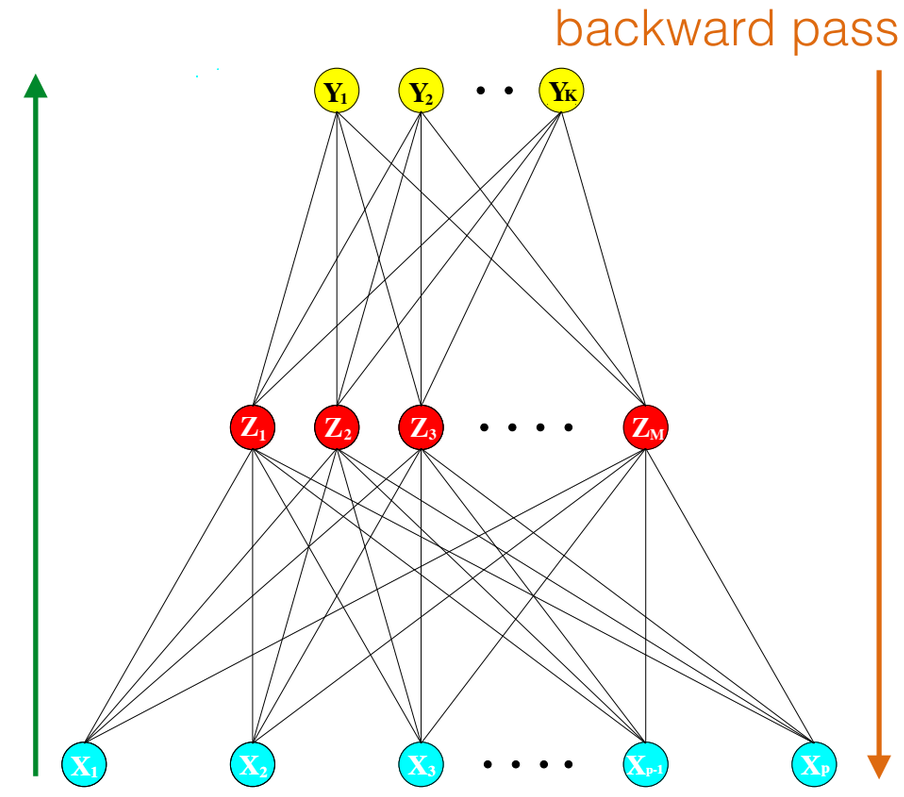
$$\frac{\partial R_i}{\partial \beta_{km}} = \delta_{ki} z_{mi},$$
$$\frac{\partial R_i}{\partial \alpha_{m\ell}} = s_{mi} x_{i\ell}.$$

Back-propagation

- δ_{ki} and s_{mi} are errors from the current model at output and hidden units, respectively.
- In the forward pass, predicted values are computed under the current model.
- In the backward pass, the errors δ_{ki} are computed (0-1 error for softmax), and then the errors s_{mi} are computed via the back-propagation equations:

$$s_{mi} = \sigma'(\alpha_m^T x_i) \sum_{k=1}^{K} \beta_{km} \delta_{ki},$$

- Both sets of errors are then used to compute the gradients for the updates



one training epoch =
one forward pass + one backward pass

- Back-propagation technique provide a computationally efficient method for evaluating gradients.

- In the back propagation algorithm, each hidden unit passes and receives information only to and from units that share a connection.

  - Hence it can be implemented efficiently on a parallel architecture computer.

- Back-propagation can also be carried out online—processing each observation one at a time, updating the gradient after each training case, and cycling through the training cases many times.

- Online training allows the network to handle very large training sets, and also to update the weights as new observations come in.

- One training epoch ('iːpɒk; US also ‘ɛpək) refers to one sweep through the entire training set.

Stochastic gradient descent

- Stochastic gradient descent (SGD) simply approximates the gradient using only a single or a few training examples.

$$\theta^{new} = \theta^{old} - \gamma \cdot \nabla R(\theta^{old})$$

- In SGD, the learning rate γ is typically much smaller than the learning rate in gradient descent because there is much more variance in the update.

- One benefit of SGD is that it's computationally faster as each update requires a few training samples to approximate the gradient, which may be computationally tractable for a very large data set.

- Usually, this computational advantage is leveraged by performing many more iterations of SGD, making more iterations than the gradient descent.

Issues in training neural networks

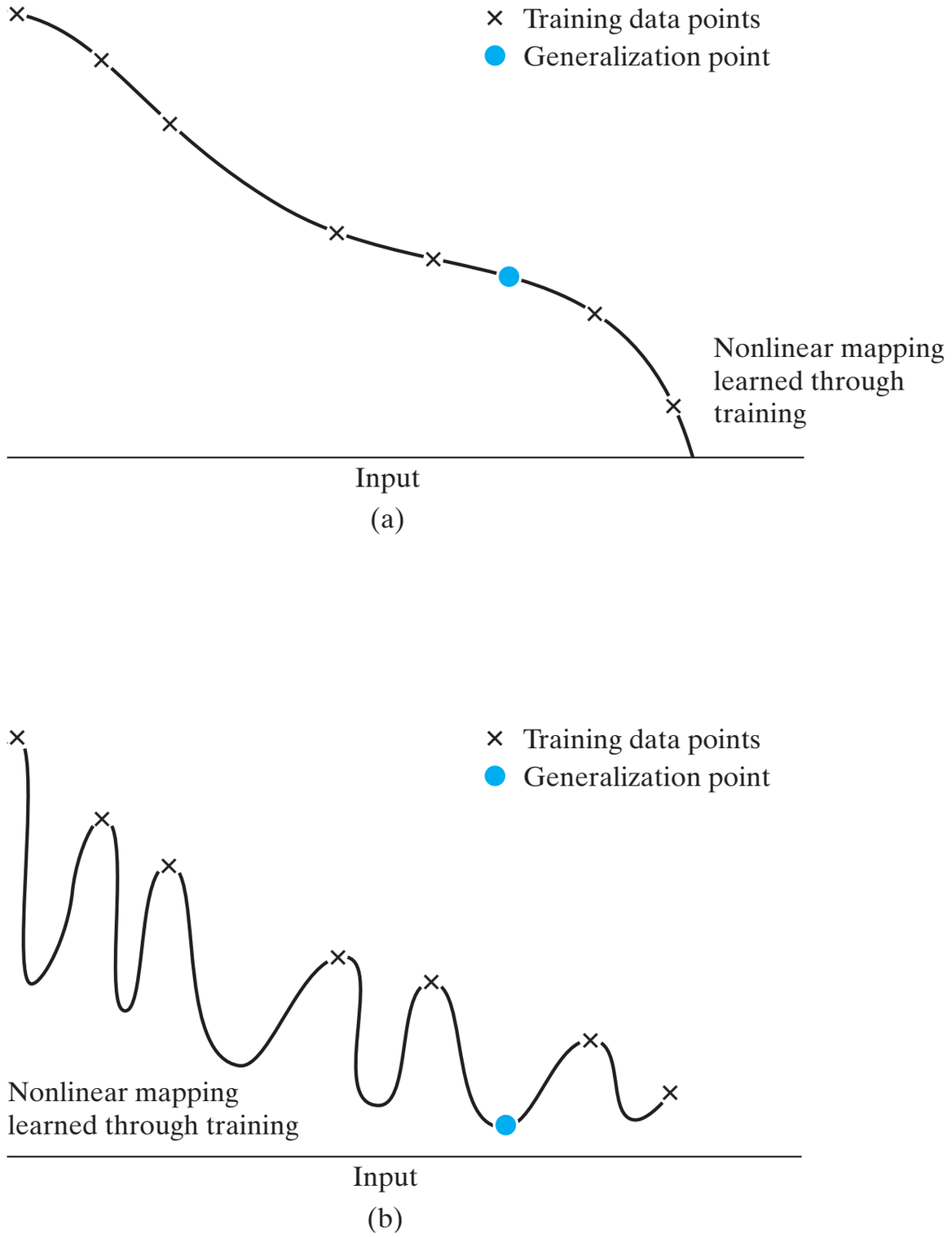
- Usually starting values for weights are chosen to be random values near zero, so the model starts out nearly linear, and becomes nonlinear as the weights increase.
- Training neural networks is a non-convex minimization problem:
  - There are multiple local minimums.
  - One must at least try a number of random starting configurations, and choose the solution giving lowest (penalized) error

Overfitting

- Q. what is overfitting?
- Q. what problem would be caused by overfitting the data?

- With many weights, the global minimizer of $R(\theta)$ is likely to be an overfit solution.

- Instead some regularization is needed: this is achieved directly through a penalty term, or indirectly by early stopping.



- NNLM 4.16: (a) Properly fitted nonlinear mapping with good generalization. (b) Overfitted nonlinear mapping with poor generalization.

- Early stopping: weights start at a highly regularized (linear) solution, this has the effect of shrinking the final model toward a linear model.

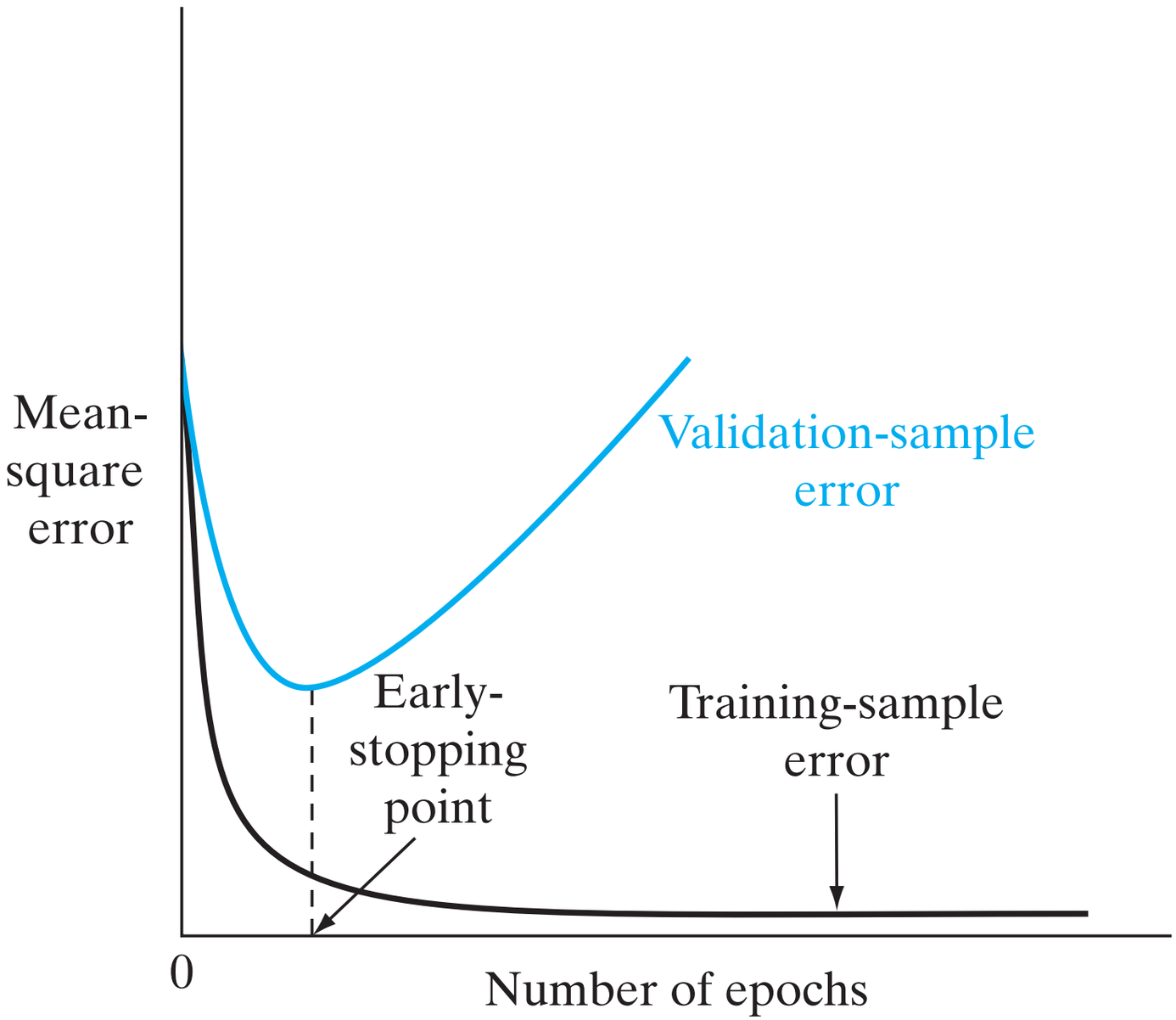
- Weight decay: we consider the penalized error R(θ) + λJ(θ) with

- the ridge penalty $J(\theta) = \sum_{km} \beta_{km}^2 + \sum_{m\ell} \alpha_{m\ell}^2$

- or weight elimination penalty (shrink smaller weights more than the ridge)

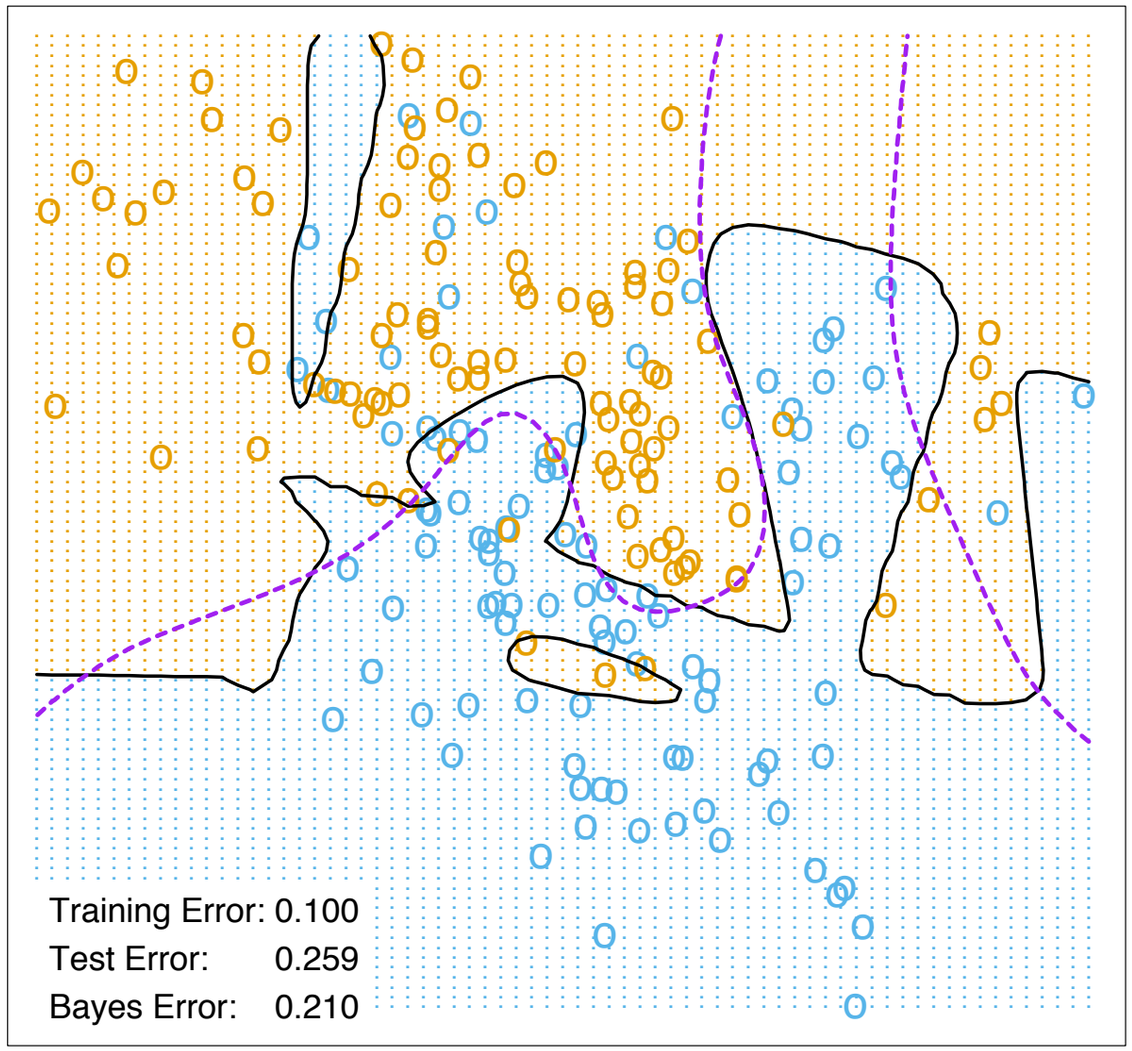
- $$J(\theta) = \sum_{km} \frac{\beta_{km}^2}{1+\beta_{km}^2} + \sum_{m\ell} \frac{\alpha_{m\ell}^2}{1+\alpha_{m\ell}^2}$$

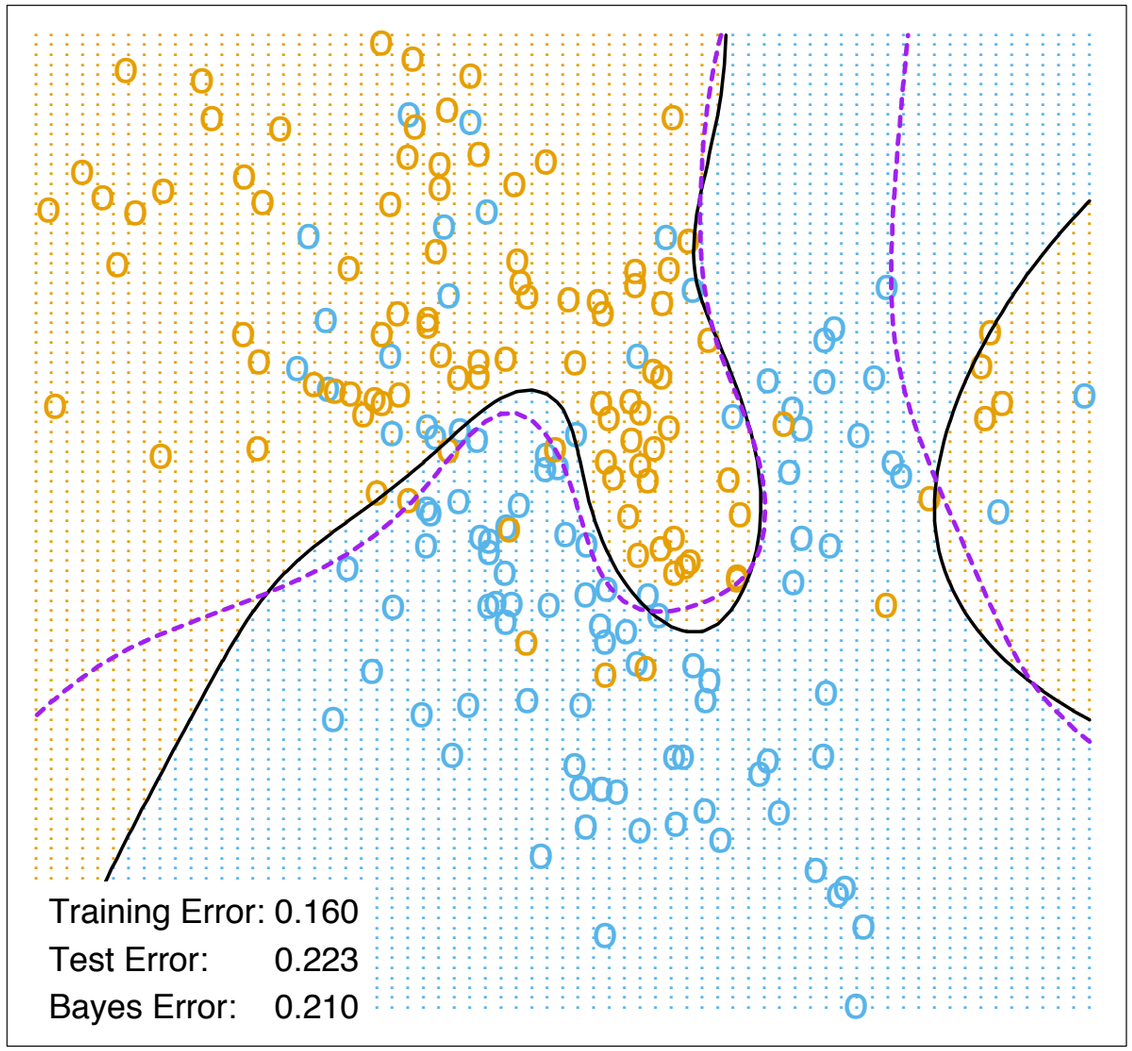
Early stopping



NNLM (Neural Networks and Learning Machines (2009) by Haykin, S., 3rd edition).

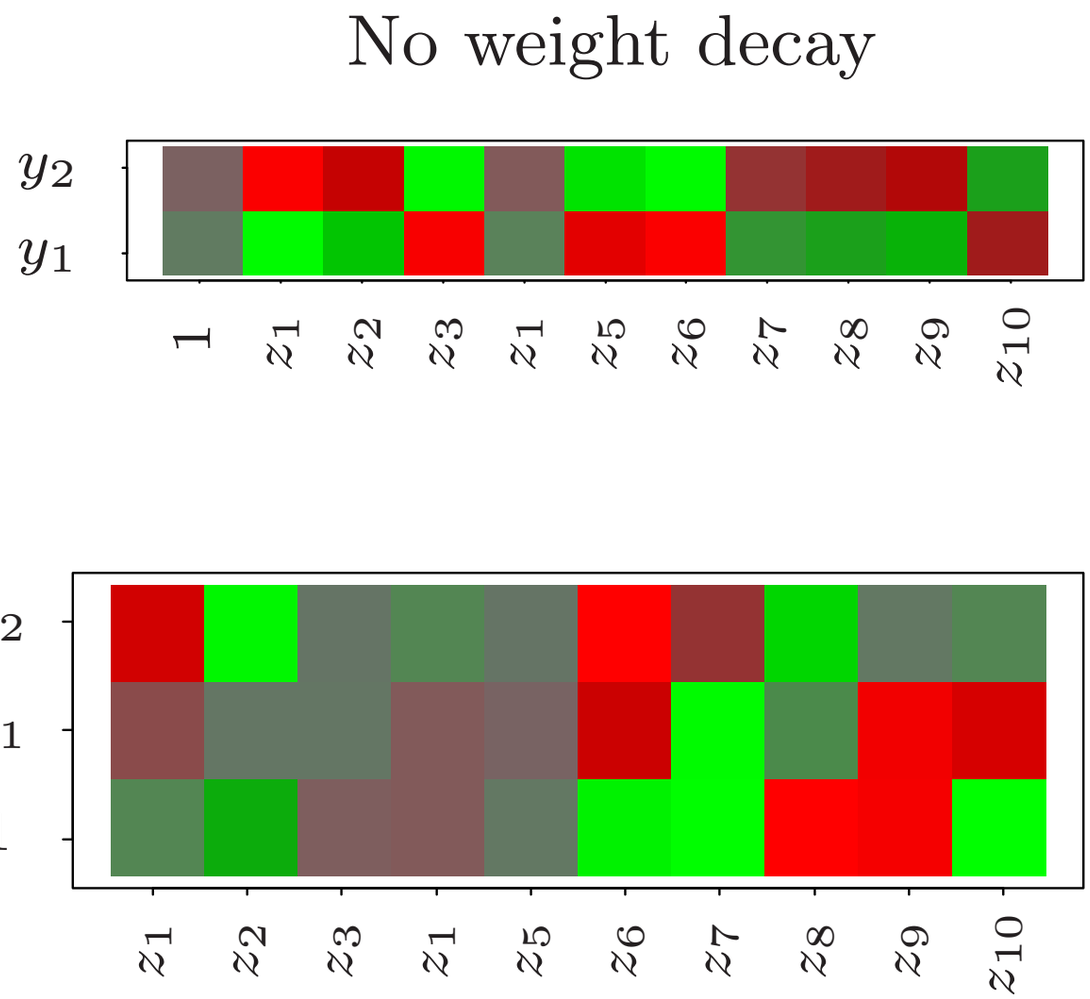
NNLM 4.17: Illustration of the early-stopping rule based on cross validation.

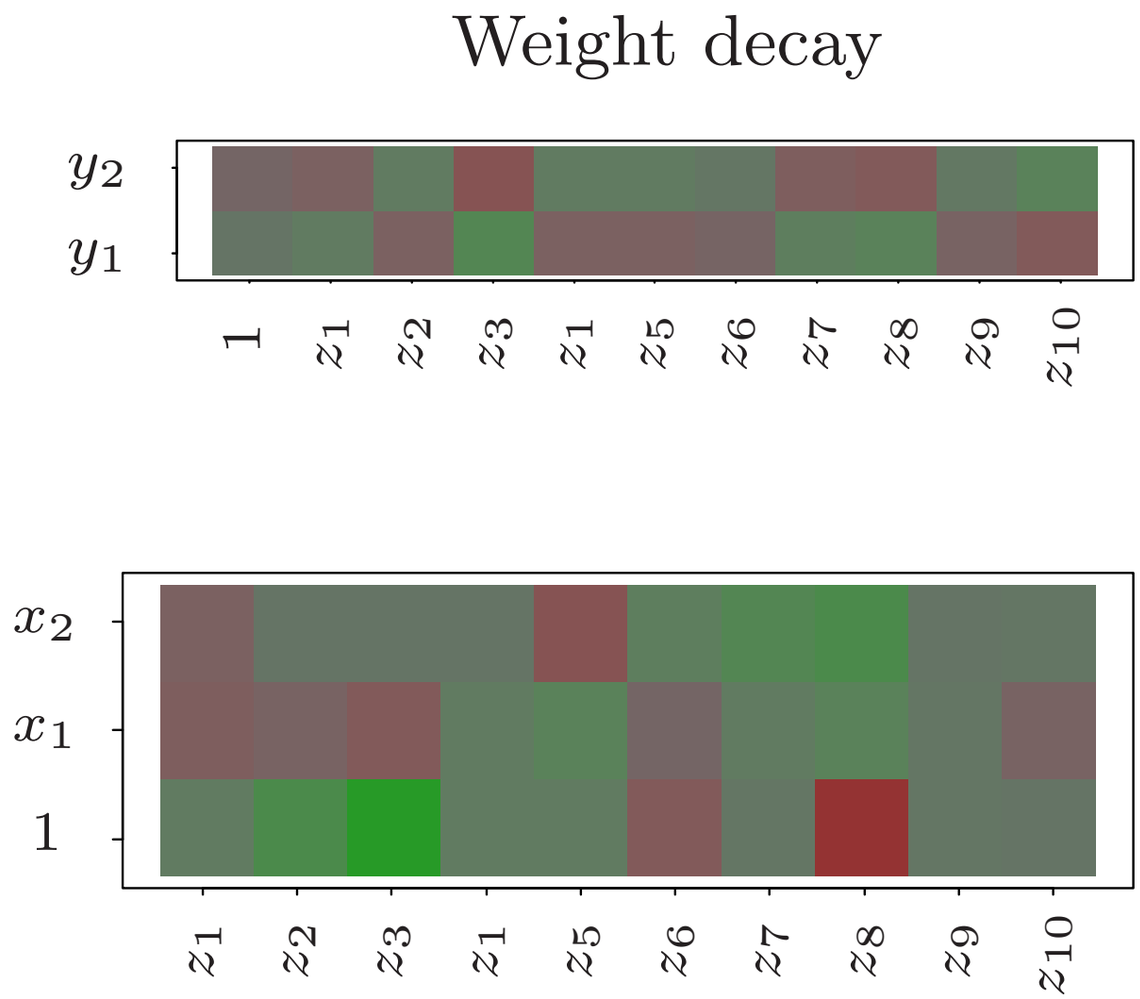




- ESL 11.4: The left panel uses no weight decay, and overfits the training data. The right panel uses weight decay, and achieves close to the Bayes error rate (broken purple boundary).

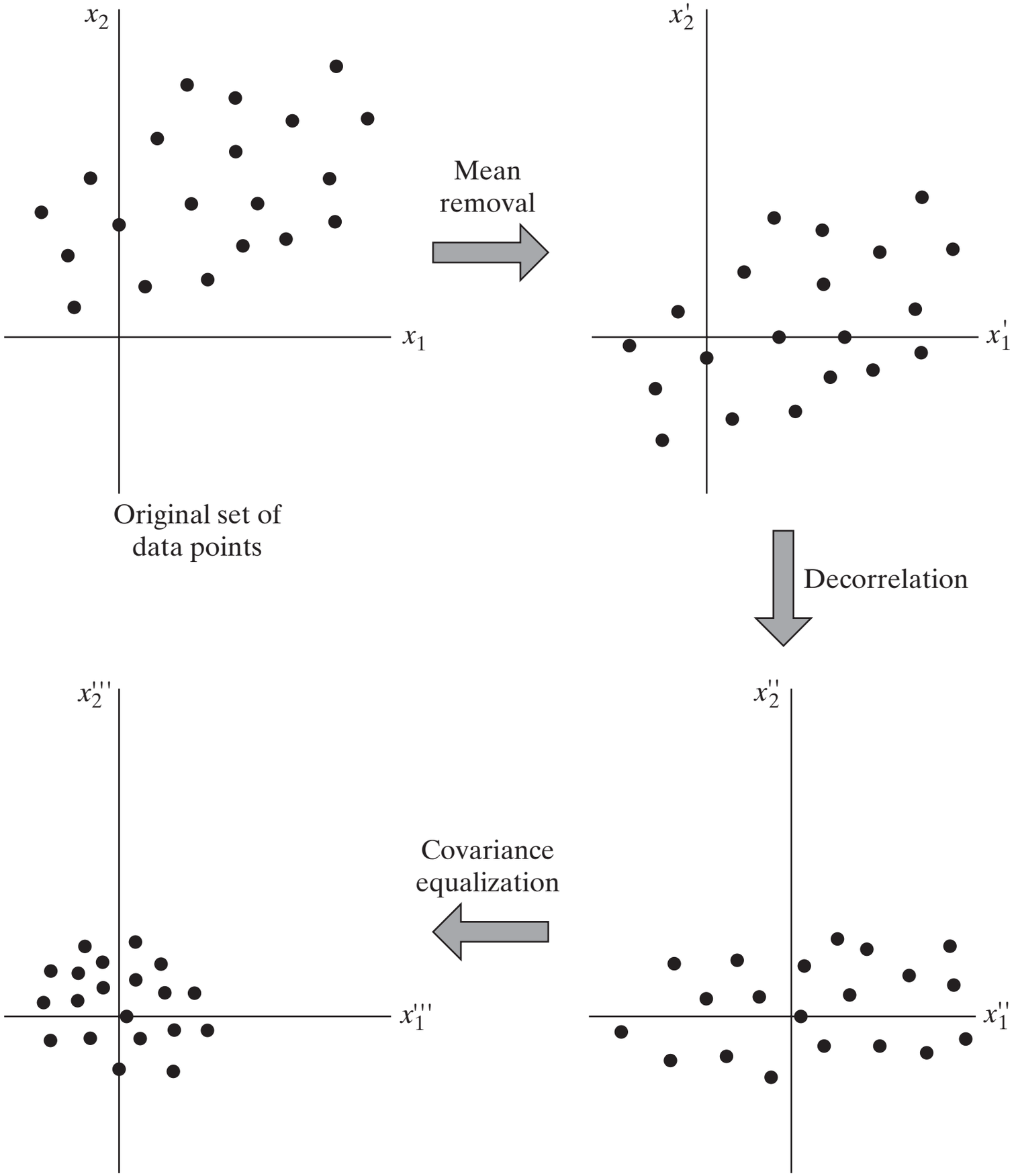
- Neural network can provide non-linear decision boundaries.





- ESL 11.5: Heat maps of the estimated weights from the training of neural networks from ELS Figure 11.4. The display ranges from bright green (negative) to bright red (positive).

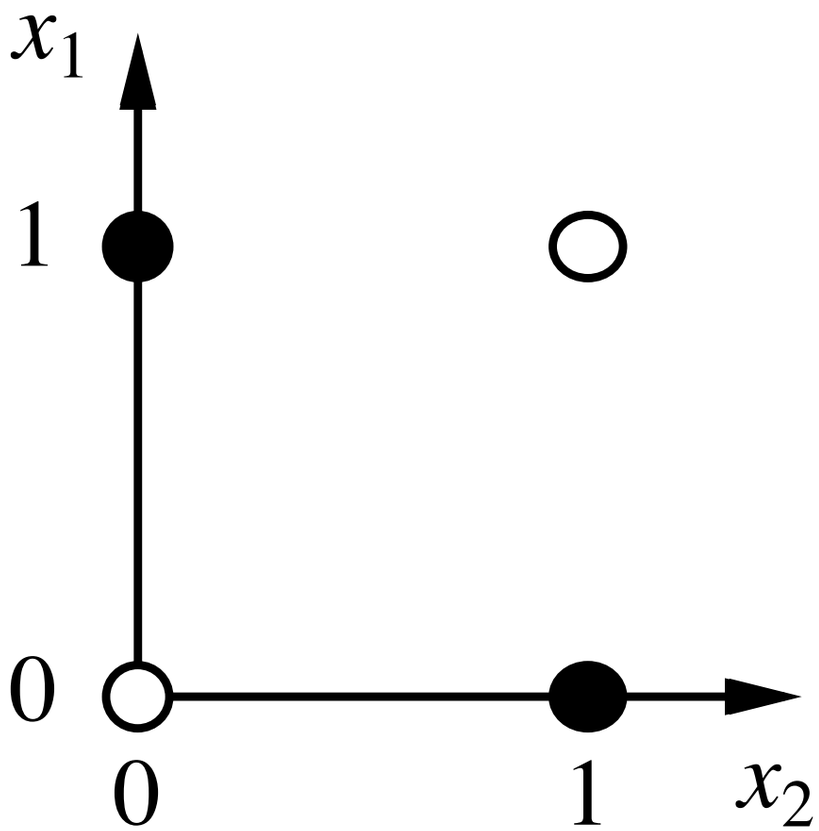
- Standardized inputs (all inputs to have mean zero and unit variance)
  - This ensures all inputs are treated equally in the regularization process, and allows one to choose a meaningful range for the random starting weights from [–0.7, +0.7].
- Number of hidden units and layers
  - With too few hidden units, the model might not have enough flexibility to capture the nonlinearities in the data.
  - With too many hidden units, the extra weights can be shrunk toward zero if appropriate regularization is used.



NNLM 4.14: Illustrating the operation of mean removal, decorrelation, and covariance equalization

XOR (exclusive OR)

- XOR
  - one or the other is true, but not both
  - separates two classes below:



$[(x_1=1)\ \text{XOR}\ (x_2=1)]$

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

- Nonlinearly separable patterns commonly occur.

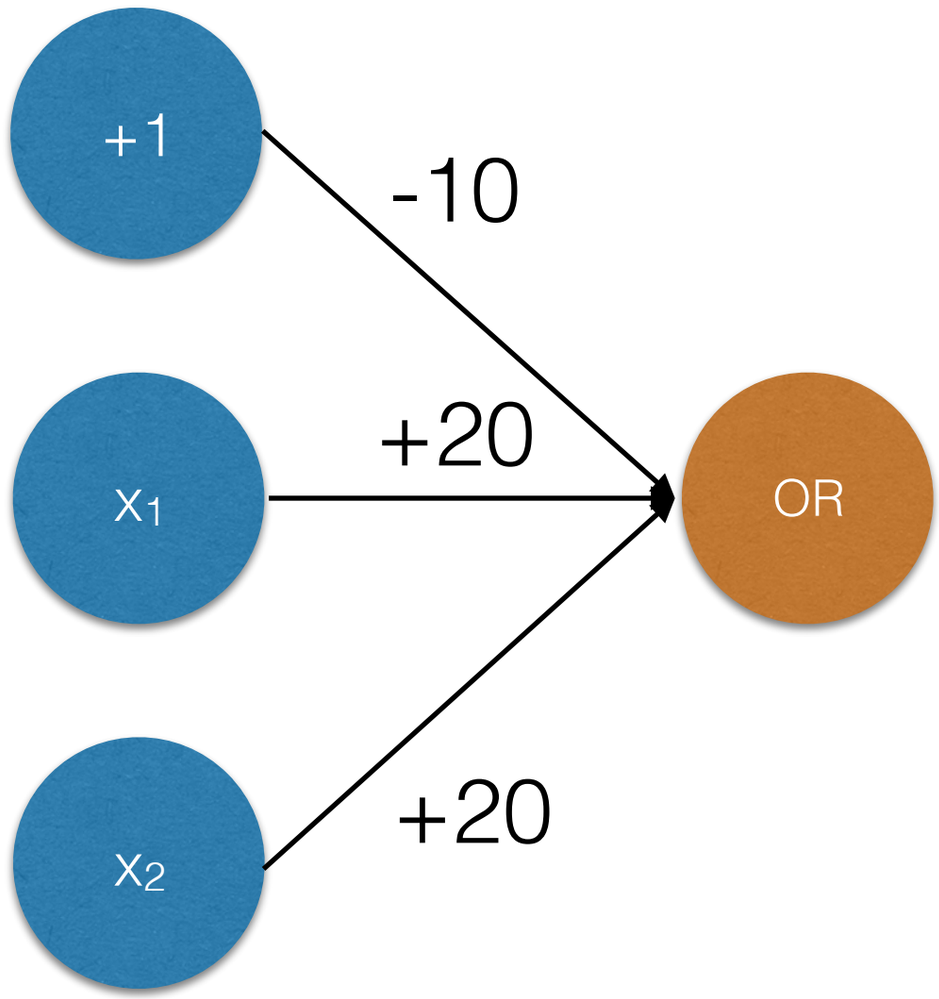
OR gate

$$z = \sigma(-10 + 20x_1 + 20x_2)$$

$$= [1 + \exp(10 - 20x_1 - 20x_2)]^{-1}$$

$(x_1=1)$ OR $(x_2=1)$

x_1	x_2	z
0	0	0
1	0	1
0	1	1
1	1	1



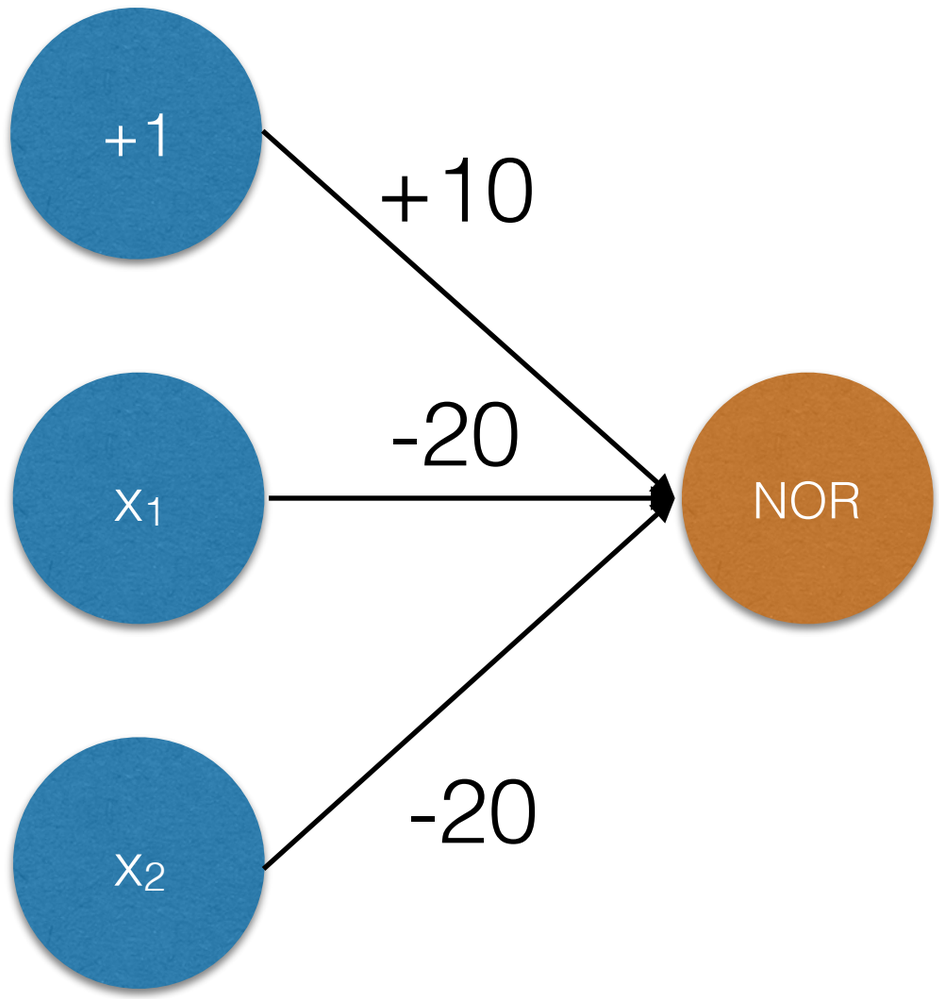
NOR gate

$$z = \sigma(10 - 20x_1 - 20x_2)$$

$$= [1 + \exp(-10 + 20x_1 + 20x_2)]^{-1}$$

NOT [(x_1=1) OR (x_2=1)]

x_1	x_2	z
0	0	1
1	0	0
0	1	0
1	1	0



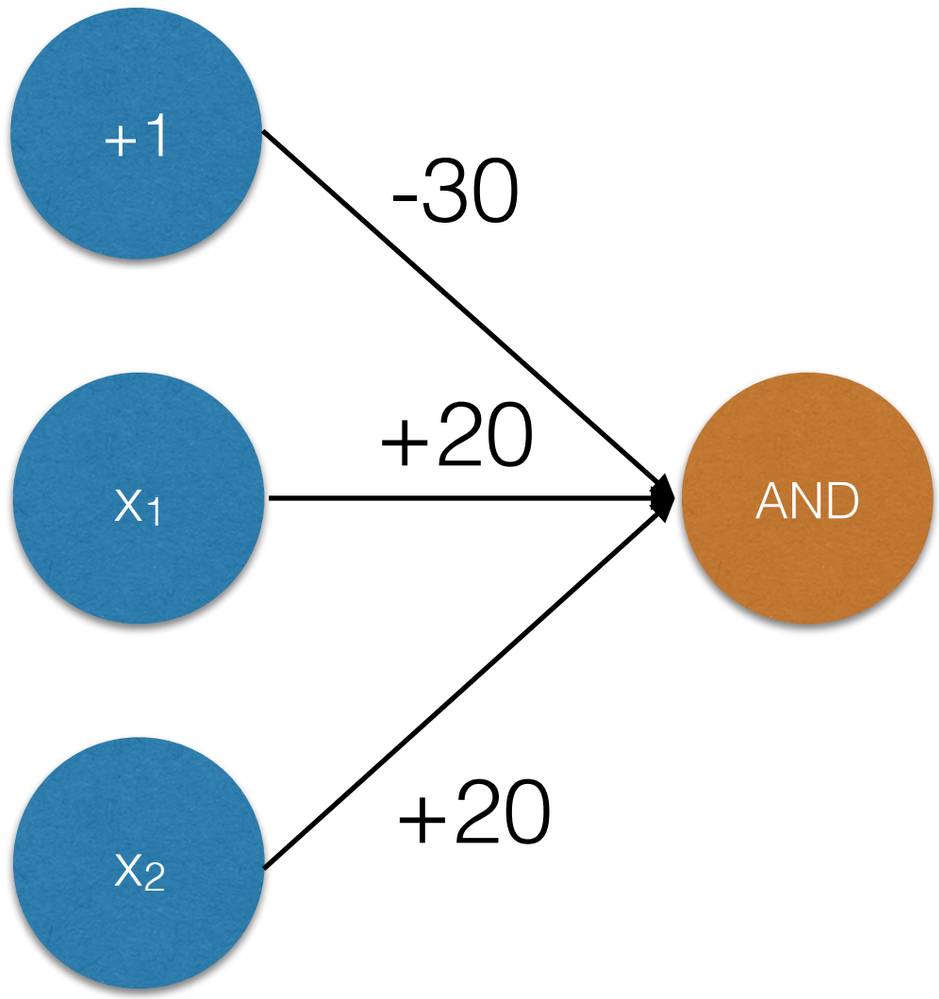
AND gate

$$z = \sigma(-30 + 20x_1 + 20x_2)$$

$$= [1 + \exp(30 - 20x_1 - 20x_2)]^{-1}$$

$(x_1=1)$ AND $(x_2=1)$

x_1	x_2	z
0	0	0
1	0	0
0	1	0
1	1	1



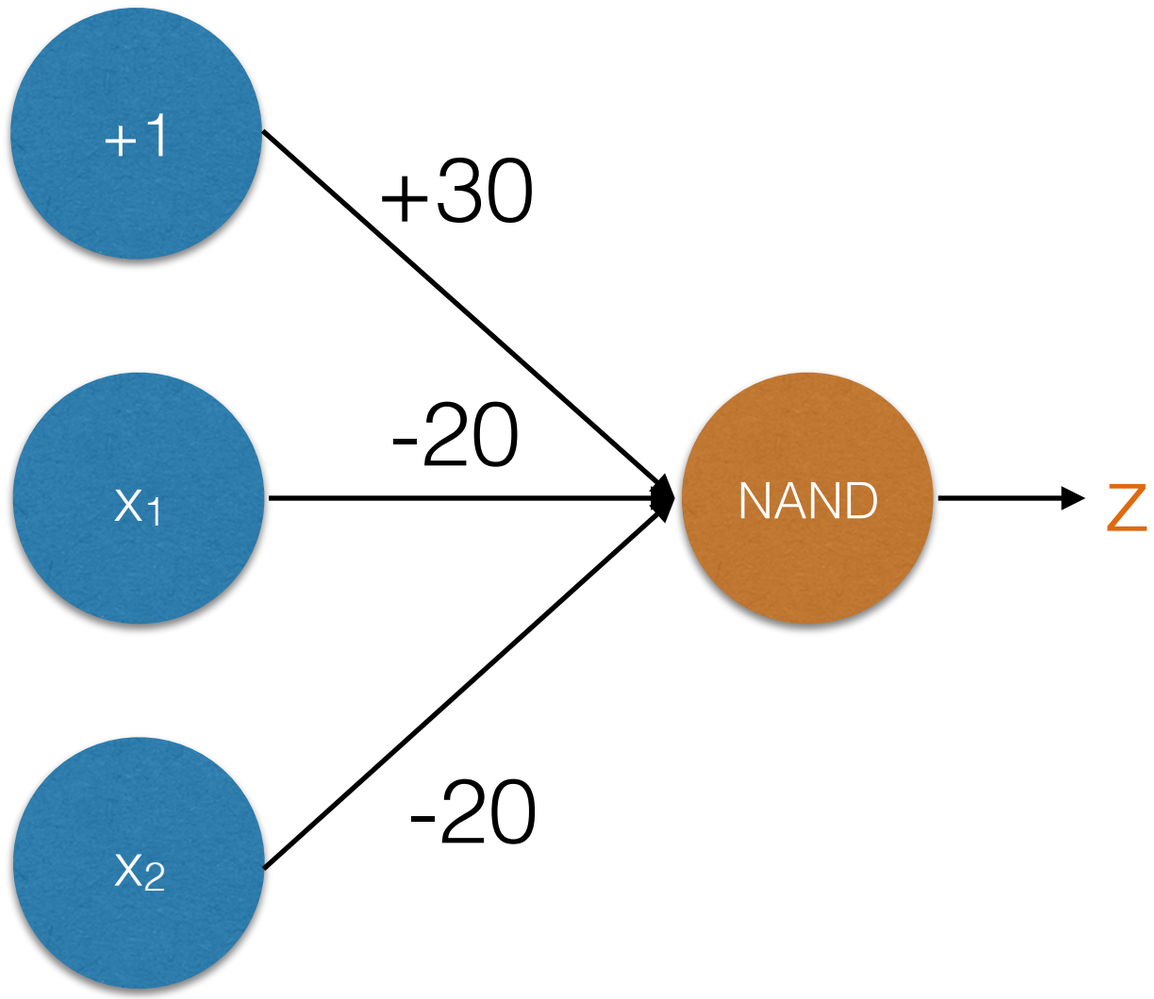
NAND gate

$$z = \sigma(30 - 20x_1 - 20x_2)$$

$$= [1 + \exp(-30 + 20x_1 + 20x_2)]^{-1}$$

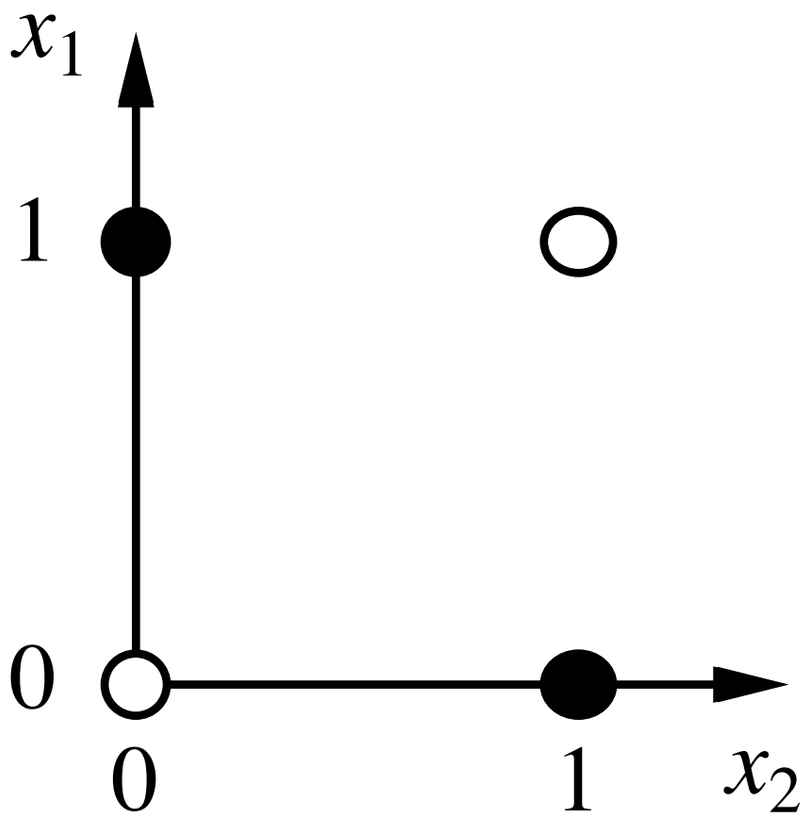
NOT [(x_1=1) AND (x_2=1)]

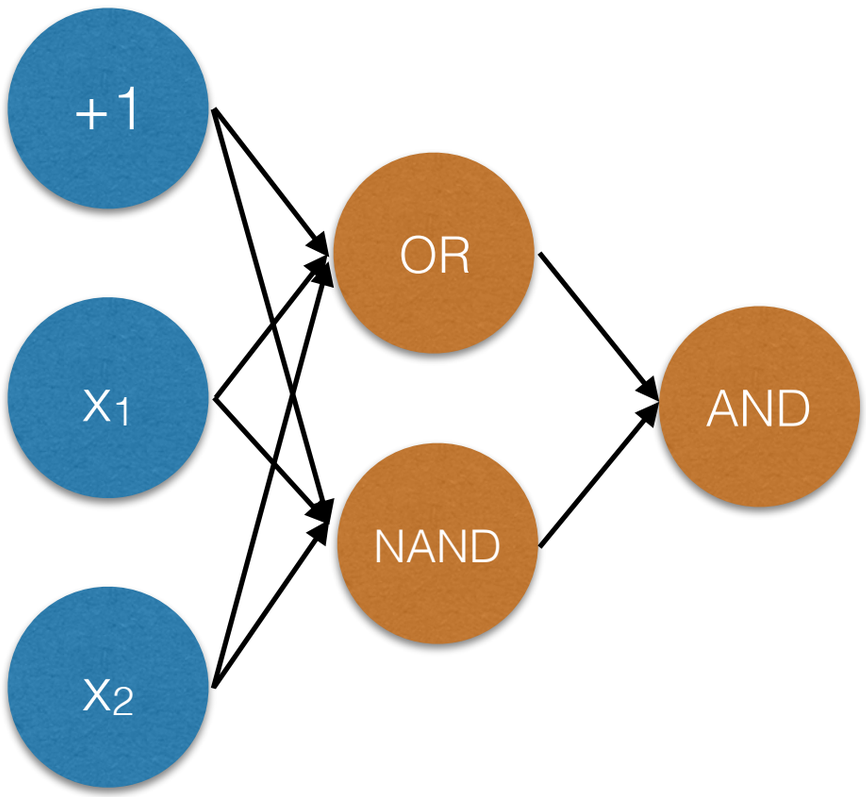
x_1	x_2	z
0	0	1
1	0	1
0	1	1
1	1	0



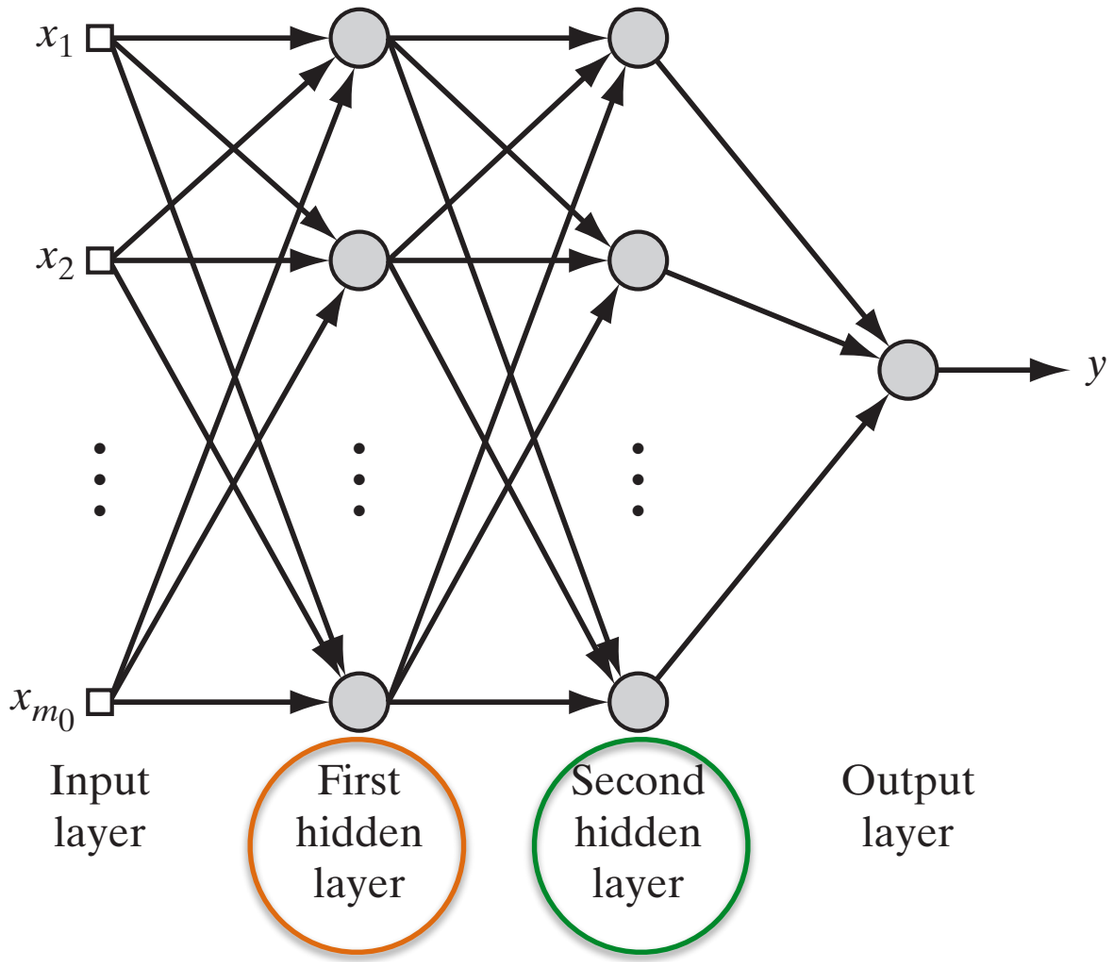
XOR gate

x_1	x_2	OR	NAND	AND
0	0	0	1	0
1	0	1	1	1
0	1	1	1	1
1	1	1	0	0





NNLM 4.14: Multilayer perceptron with two hidden layers



Simple functions of inputs

Complex functions of inputs

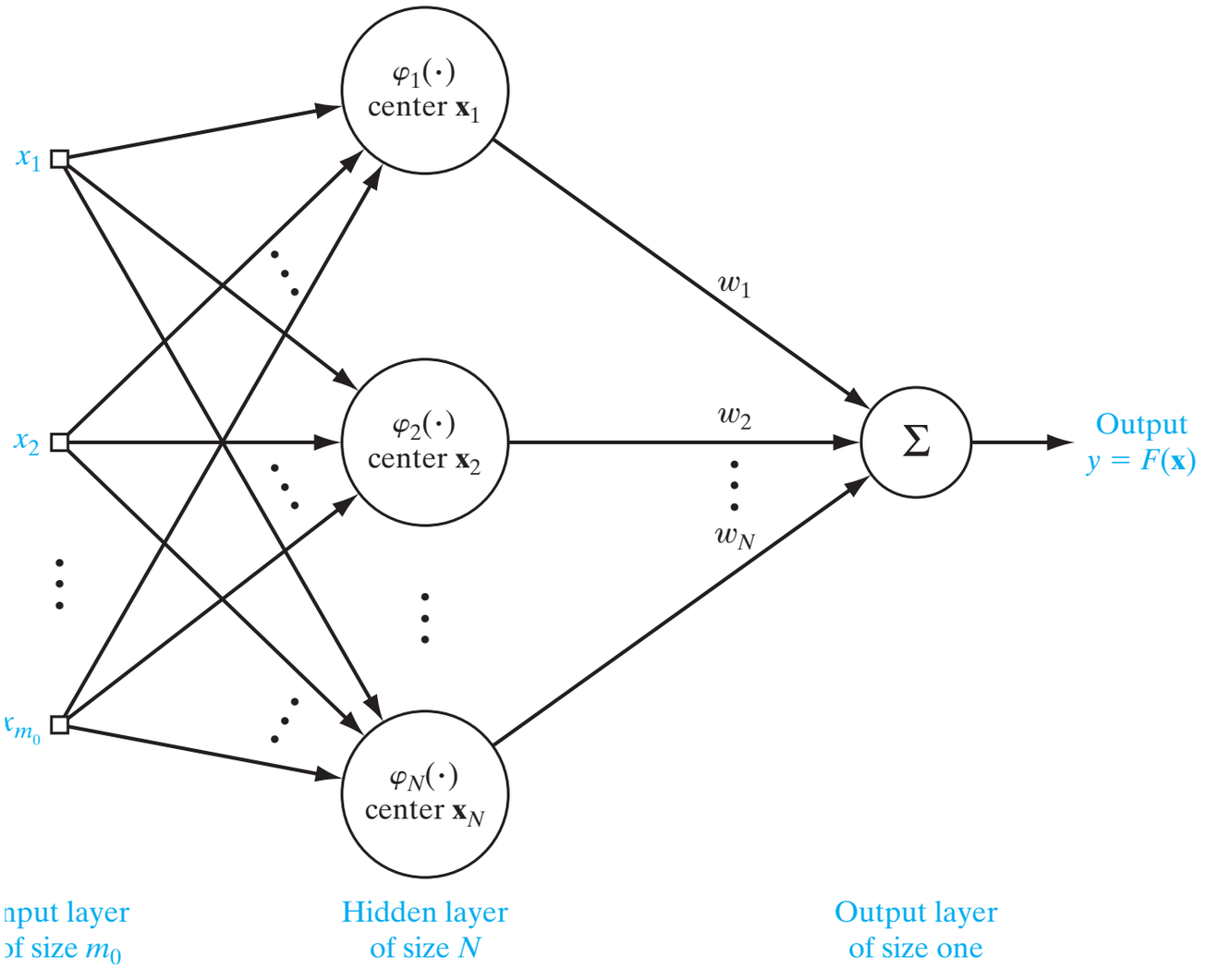
- Neural networks with multiple hidden layers can compute complex functions of inputs.
  - As number of hidden layers increases, the final output becomes a more complex function of features (higher level of abstraction)
  - Hidden units perform the feature extraction.

Structure of RBF neural networks

- NNLM 5.3: Radial-basis function (RBF) neural networks

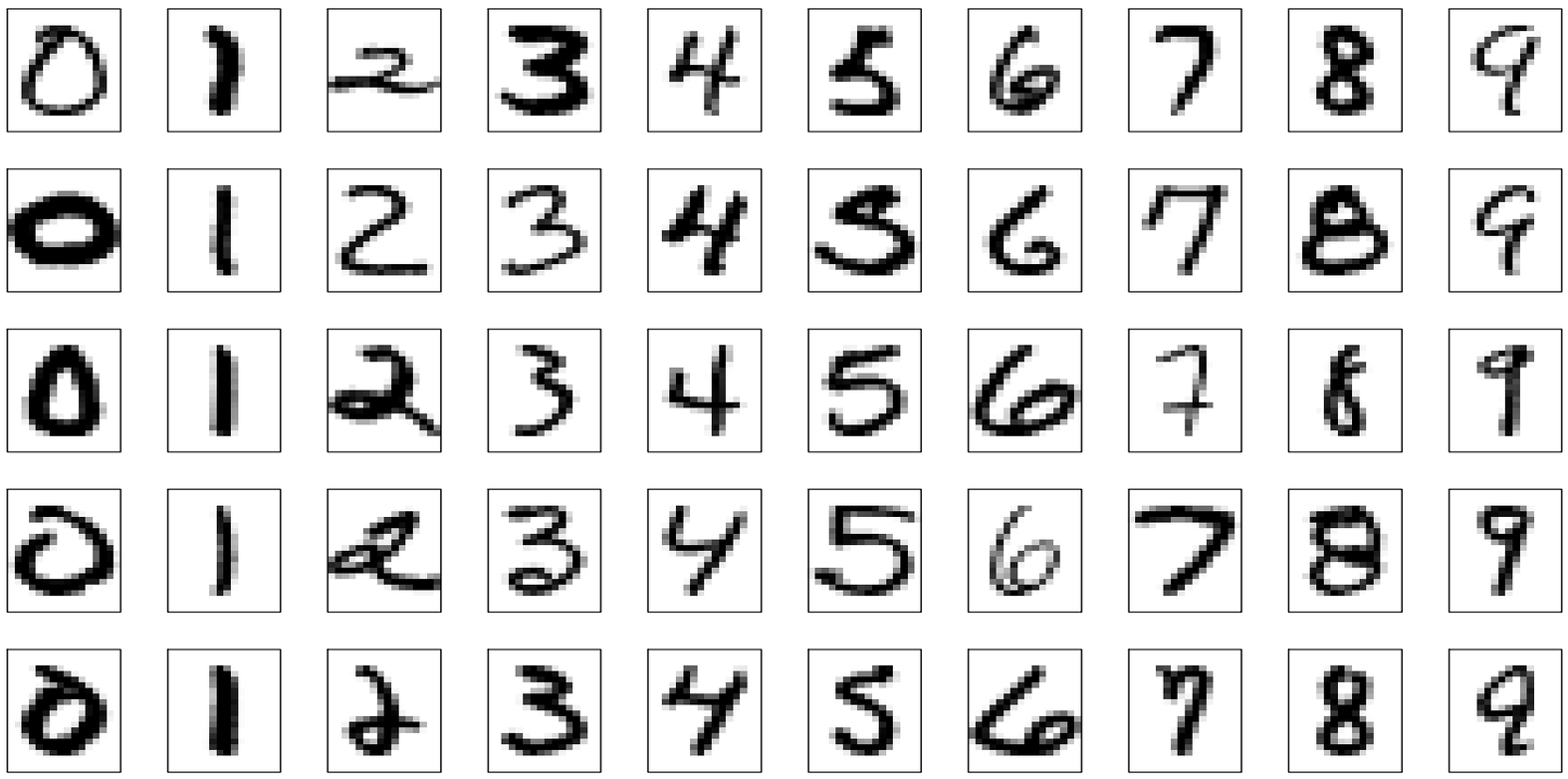
$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i \varphi(x - x_i),\ i = 1,\ldots,N$$

$$\varphi(x - x_i) = \exp(-\tfrac{1}{2\sigma_i^2} \| x - x_i \|^2)$$

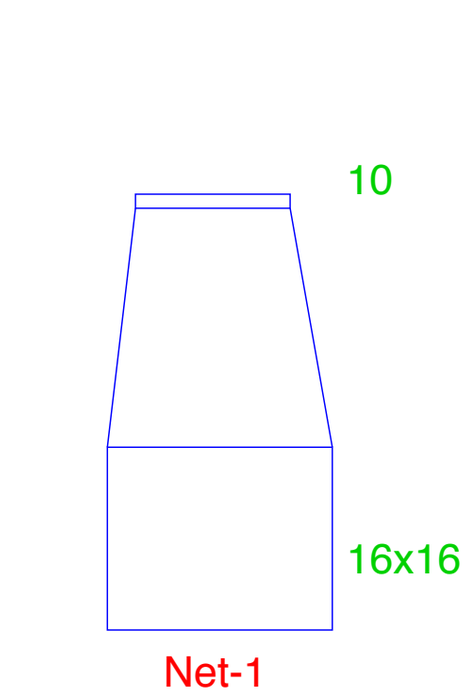


- Radial-basis activation function is used. Commonly, the *Gaussian hidden units* are assigned a common width σ.

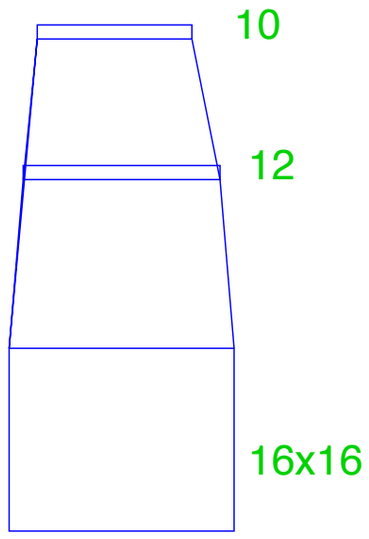
- RBF neural networks are conceptually similar to nearest
- neighbor models.

- ESL 11.9: Examples of normalized handwritten digits scanned from envelopes by the USPS.
  - The original scanned digits are binary and of different sizes and orientations.
  - Images have been de-slanted and size normalized, resulting in 16×16 grayscale images (Le Cun et al., 1990).
  - 

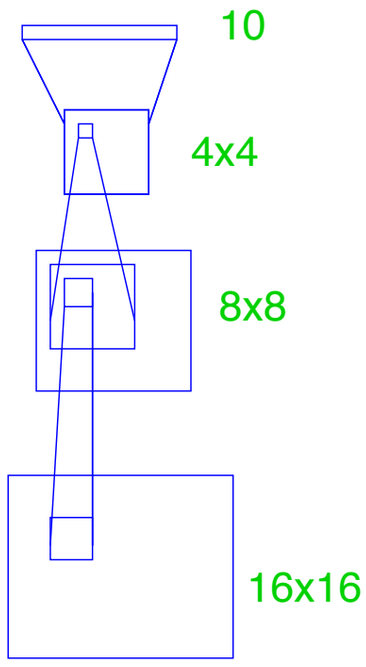
- Net-1: No hidden layer, equivalent to multinomial logistic regression.
- Net-2: One hidden layer, 12 hidden units fully connected.
- Net-3: Two hidden layers locally connected (3x3 local patch in inputs, 5x5 local patch in first hidden layers).
- Net-4: Two hidden layers, locally connected with weight sharing but own bias parameters (3x3 local patch in inputs).
- Net-5: Two hidden layers, locally connected, two levels of weight sharing.



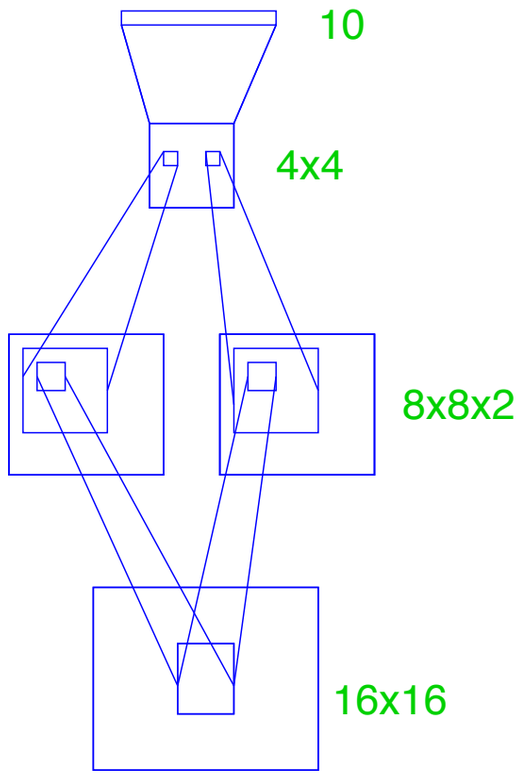
Net-1



Net-2

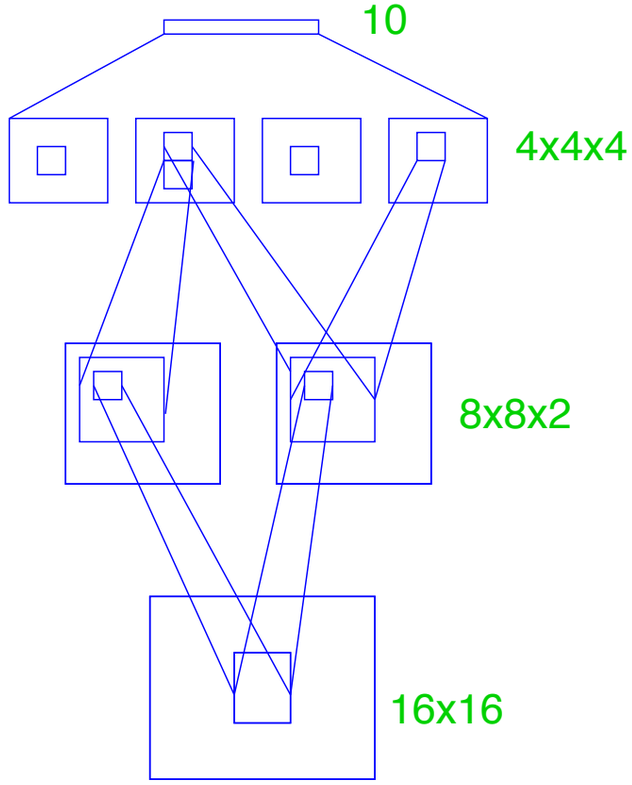


Net-3
Local Connectivity



Net-4

Shared Weights



Net-5

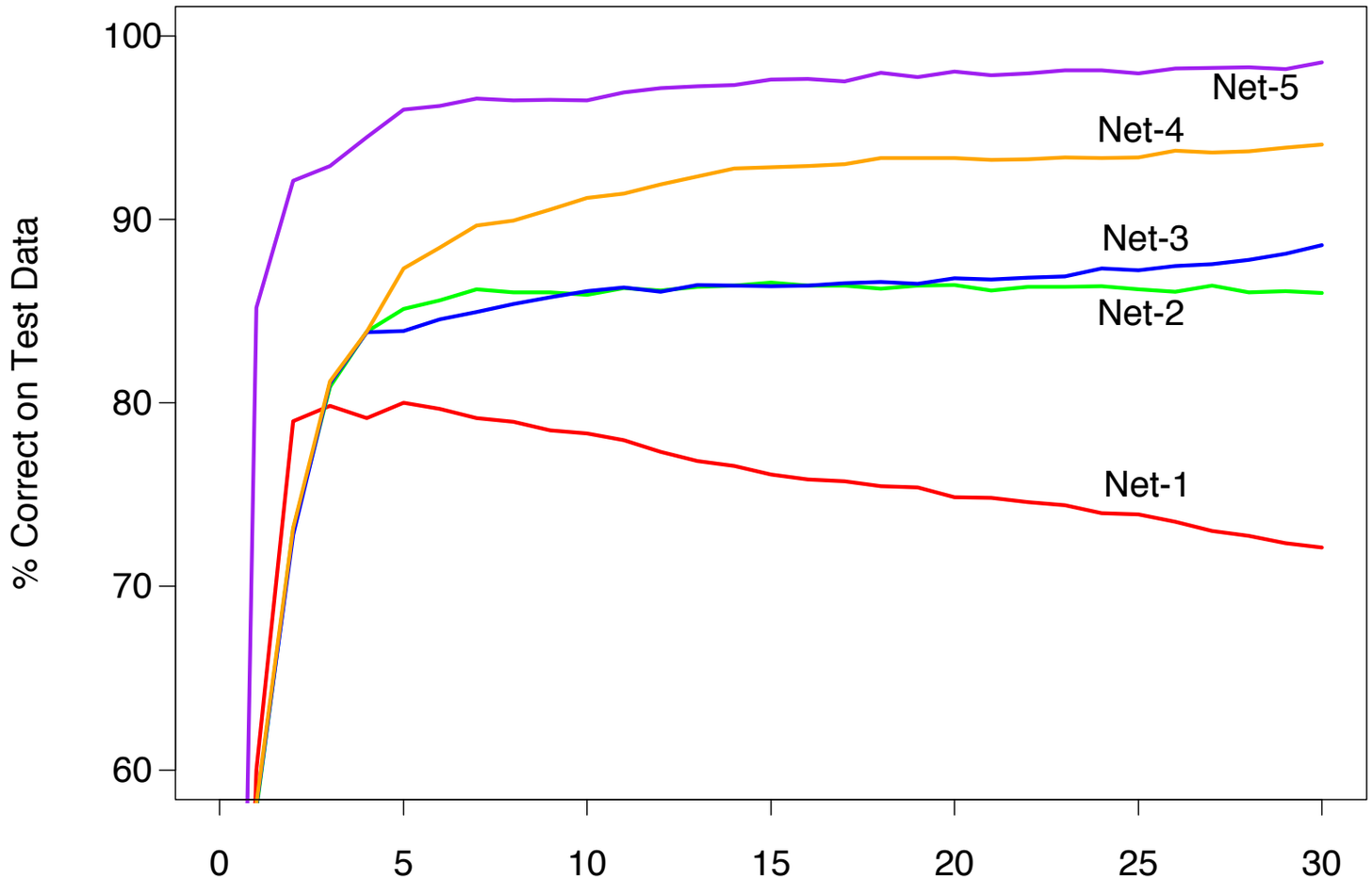


TABLE 11.1. *Test set performance of five different neural networks on a handwritten digit classification example (Le Cun, 1989).*

	Network Architecture	Links	Weights	% Correct
Net-1:	Single layer network	2570	2570	80.0%
Net-2:	Two layer network	3214	3214	87.0%
Net-3:	Locally connected	1226	1226	88.5%
Net-4:	Constrained network 1	2266	1132	94.0%
Net-5:	Constrained network 2	5194	1060	98.4%

Deep learning

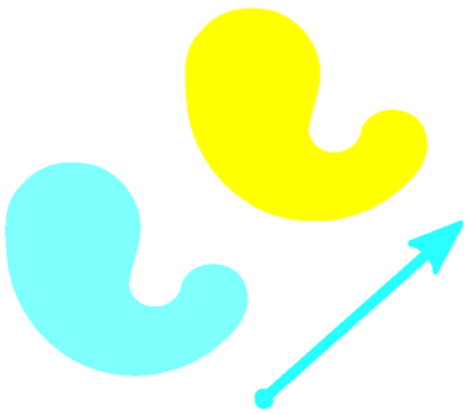
- From Wiki: Deep learning is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using multiple processing layers, with complex structures or otherwise, composed of multiple non-linear transformations.

- Various deep learning architectures such as deep neural networks, convolutional deep neural networks, deep belief networks and recurrent neural networks (restricted Boltzmann machine).

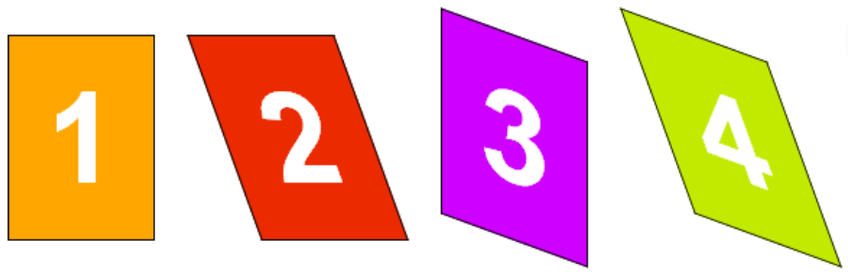
- Deep neural networks are neural networks with many hidden layers. Convolutional neural networks are Net-4 and 5 in the previous slides.

Convolutional neural networks

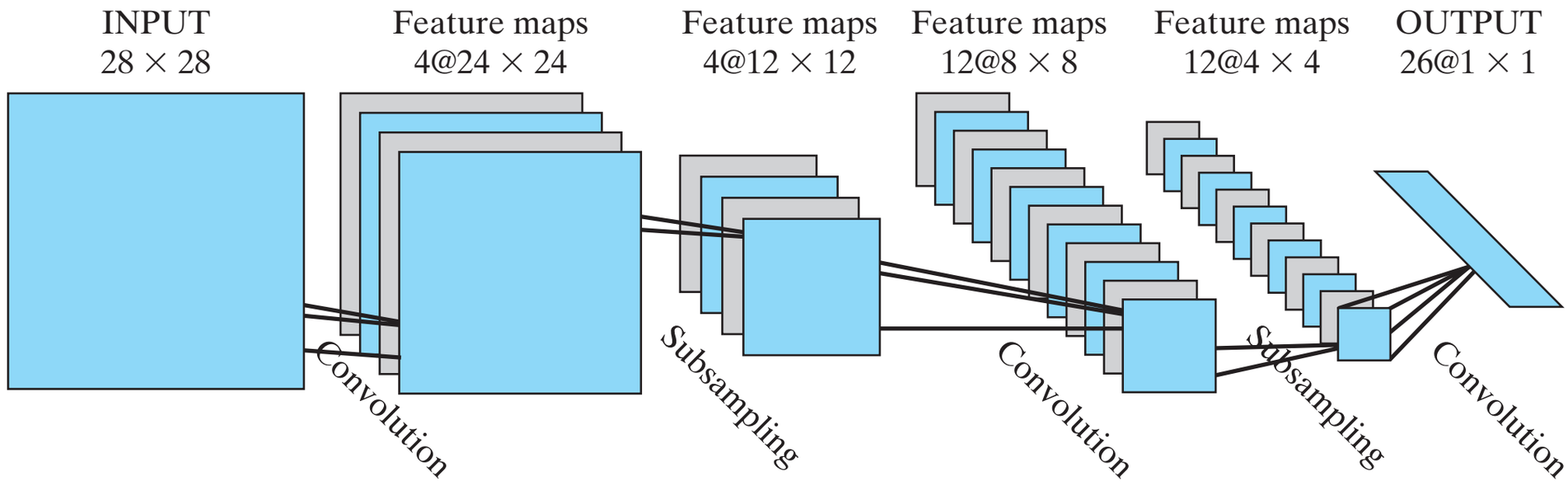
- A convolutional neural network is a multilayer perceptron designed specifically to recognize two-dimensional shapes with a high degree of invariance to translation, scaling, skewing, and other forms of distortion.



translation



skew



NNLM 4.23: Convolutional network for image processing such as handwriting recognition.

- *Feature extraction.* Each neuron takes its synaptic inputs from a local *receptive field* in the previous layer, thereby forcing it to extract local features.

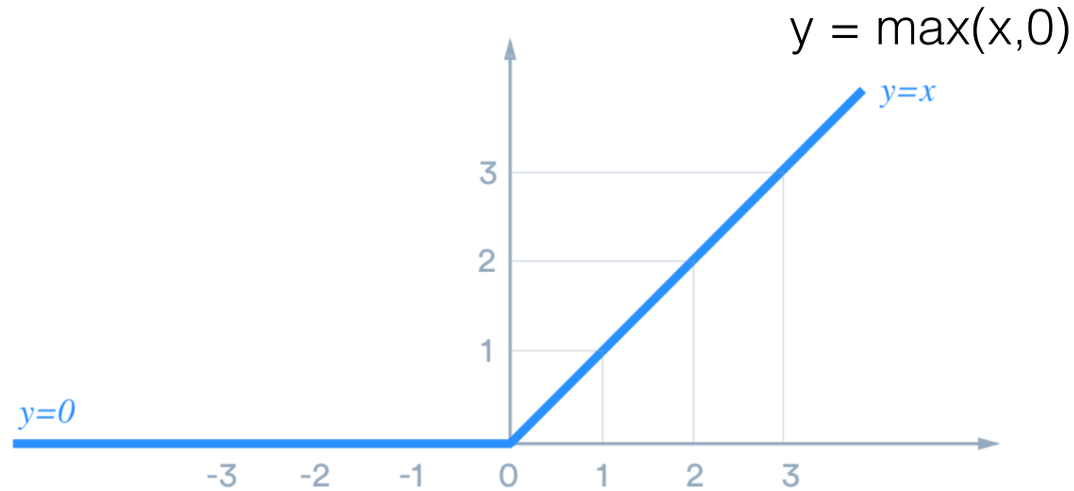
- *Feature mapping.* Each layer of the network is composed of multiple *feature maps*, with each feature map being in the form of a plane within which the individual neurons are *constrained* to share the same set of synaptic weights.

  - *shift invariance*, forced into the operation of a feature map through the use of *convolution* with a kernel of small size, followed by a sigmoid function;

  - reduction in the number of free parameters, accomplished through the use of weight sharing.

- *Subsampling.* Each convolutional layer is followed by a computational layer that performs *local averaging* and *subsampling*, whereby the resolution of the feature map is reduced.

Rectified linear unit (ReLU) activation function

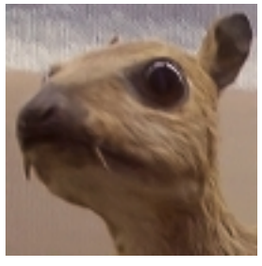


- Sparsity of the activation
  - A few neurons won't be activate, which makes activations sparse and efficient.
- commonly used activation function in neural networks, especially in CNNs.
  - easier to compute than RBF, tanh and sigmoid.
  - faster convergence

What does the convolution layer do?

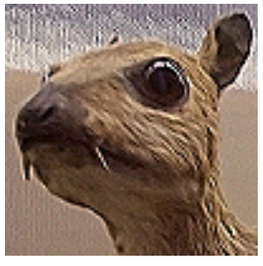
- In image processing, convolution means to apply a filter over an image at all possible offsets.
- Applying convolution of the input image with a linear filter, adding a bias term, and then applying a non-linear function.
- To form a richer representation of the data, each convolution layer is composed of multiple feature maps.

Example of linear filters

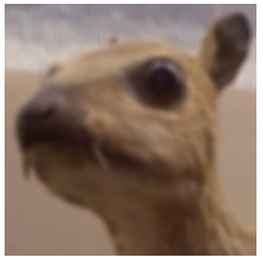


original

edge



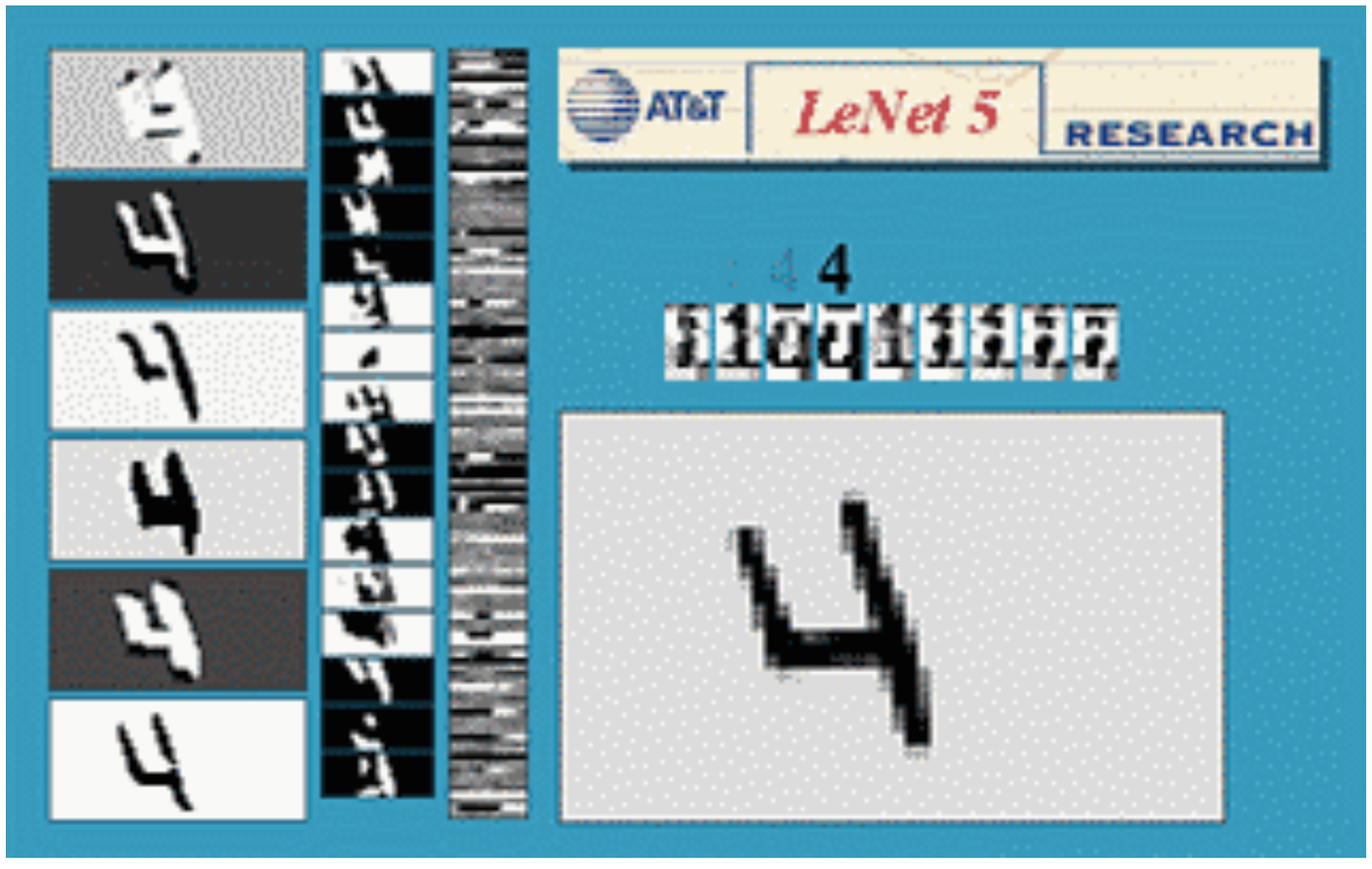
sharpen



blur

LeNet-5 Demo

- Demo credit: http://yann.lecun.com/exdb/lenet/rotation.html



Deep learning before 2006

- Before 2006, attempts at training deep architectures failed
  - Deep networks trained with back-propagation (with random initialization) perform worse than shallow networks.
  - Neural networks have limited to one or two layers.
- In usual settings, we can use only labeled data
  - Almost all data is unlabeled!
  - The brain can learn from unlabeled data

2006 breakthrough in deep learning

- Hinton, G. E., Osindero, S. and Teh, Y., A fast learning algorithm for deep belief nets Neural Computation 18:1527-1554, 2006

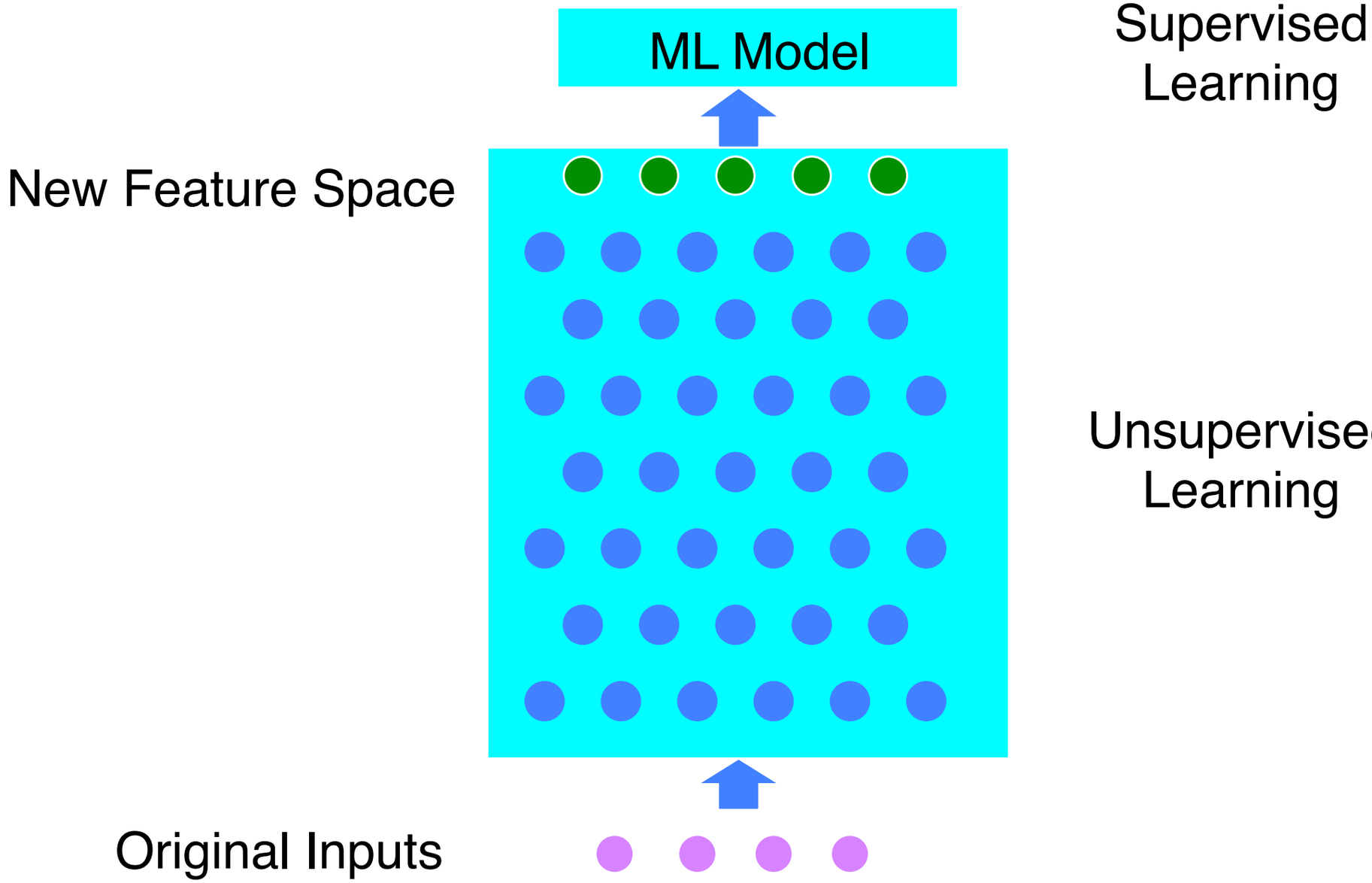
- Yoshua Bengio, Pascal Lamblin, Dan Popovici and Hugo Larochelle, Greedy Layer-Wise Training of Deep Networks, in J. Platt et al. (Eds), Advances in Neural Information Processing Systems 19 (NIPS 2006), pp. 153-160, MIT Press, 2007

- Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra and Yann LeCun, Efficient Learning of Sparse Representations with an Energy-Based Model, in J. Platt et al. (Eds), Advances in Neural Information Processing Systems (NIPS 2006), MIT Press, 2007

2006 breakthrough in deep learning

- Use unsupervised learning (greedy layer-wise training)
  - Allows abstraction to develop naturally from one layer to another
  - Help the network initialize with good parameters
- Perform supervised top-down training as final step
  - Refine the features (intermediate layers) so that they become more relevant for the task

Deep network with greedy layer-wise training



Greedy layer-wise training

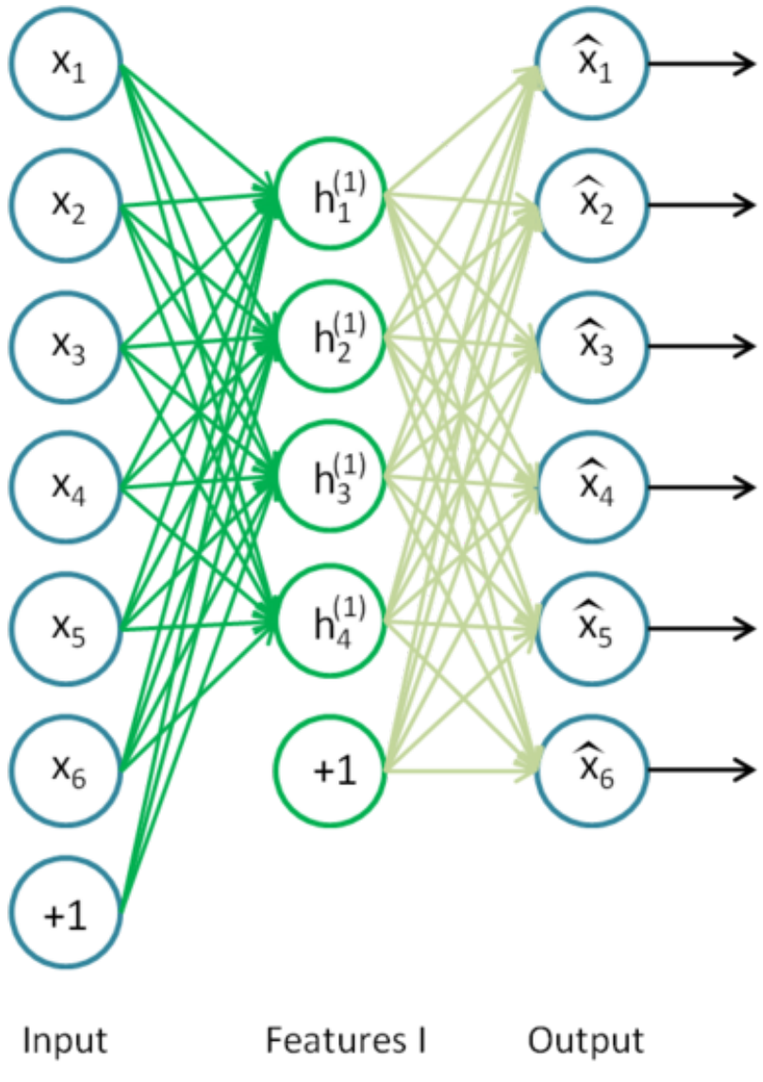
- Greedy layer-wise training avoids many of the problems of trying to train a deep net in a supervised fashion
  - Each layer gets full learning focus in its turn since it is the only current top layer
  - Can take advantage of unlabeled data
- When you finally tune the entire network with supervised training the network weights have already been adjusted so that you are in a good error basin and just need fine tuning. This helps with problems of
  - Ineffective early layer learning
  - Deep network local minima
- We will discuss one of most common approaches: sparse auto-encoders

Train sparse auto-encoders

- A type of unsupervised learning which tries to discover generic features of the data
  - learning important sub-features
  - compression
- Given unlabeled training set $x_1,\ldots,x_6$

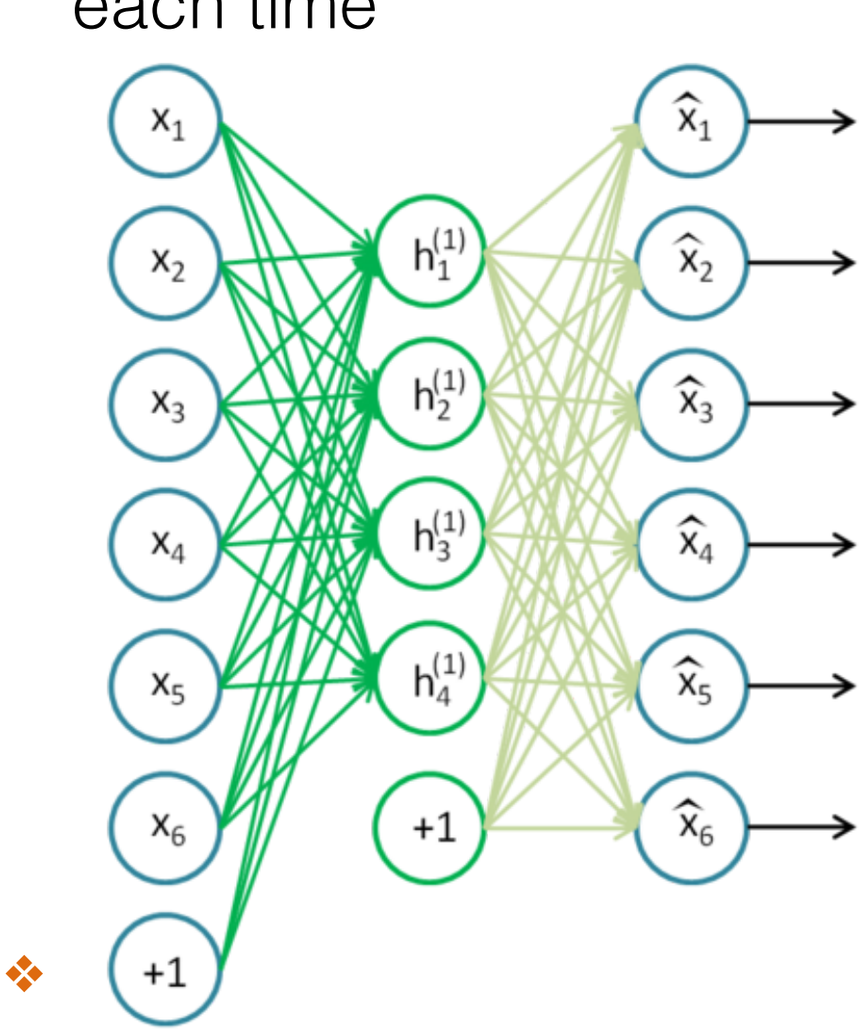
$$\min_{h^{(1)}} \| x_i - \hat{x}_i \|^2 + \lambda \sum_i | h_i^{(1)} |$$

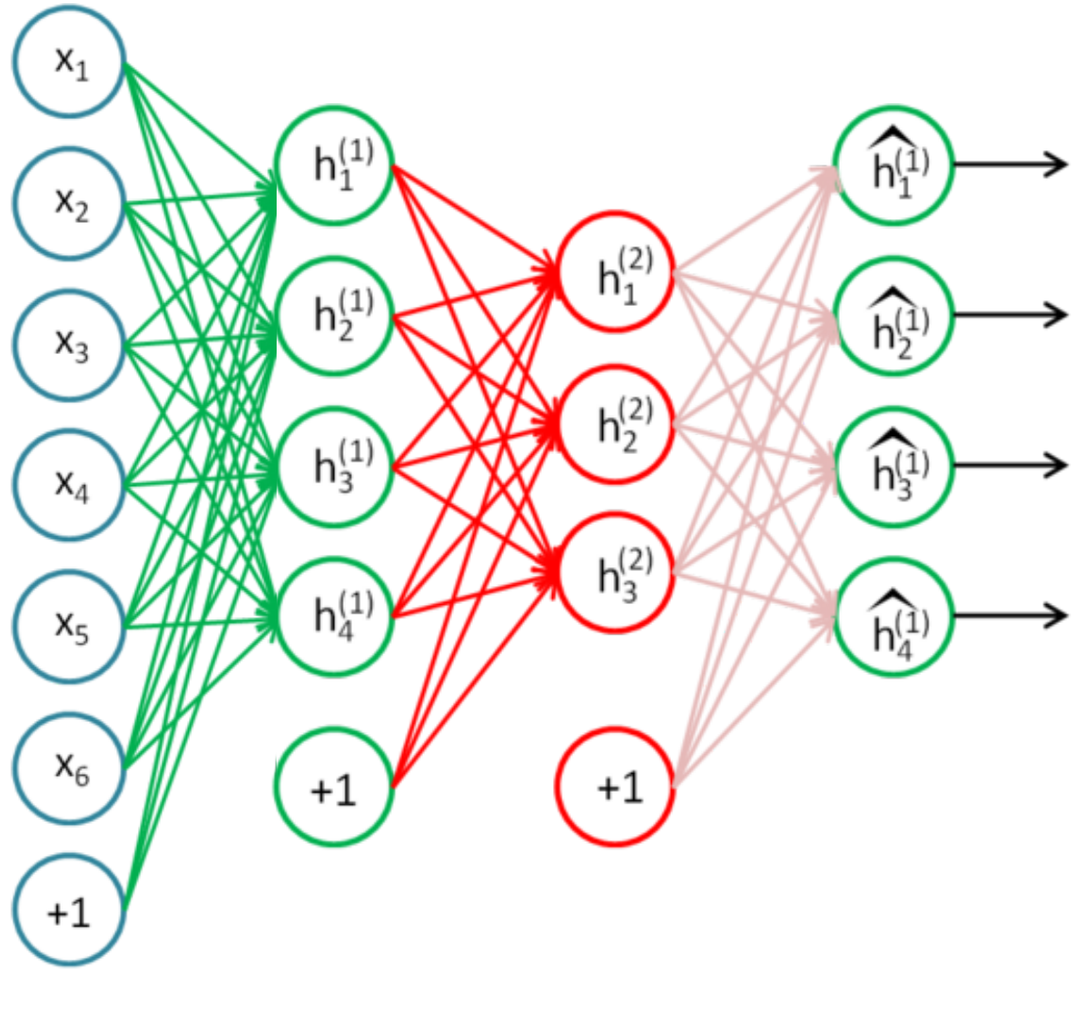
- reconstruction error L_1 penalty



Train sparse auto-encoders

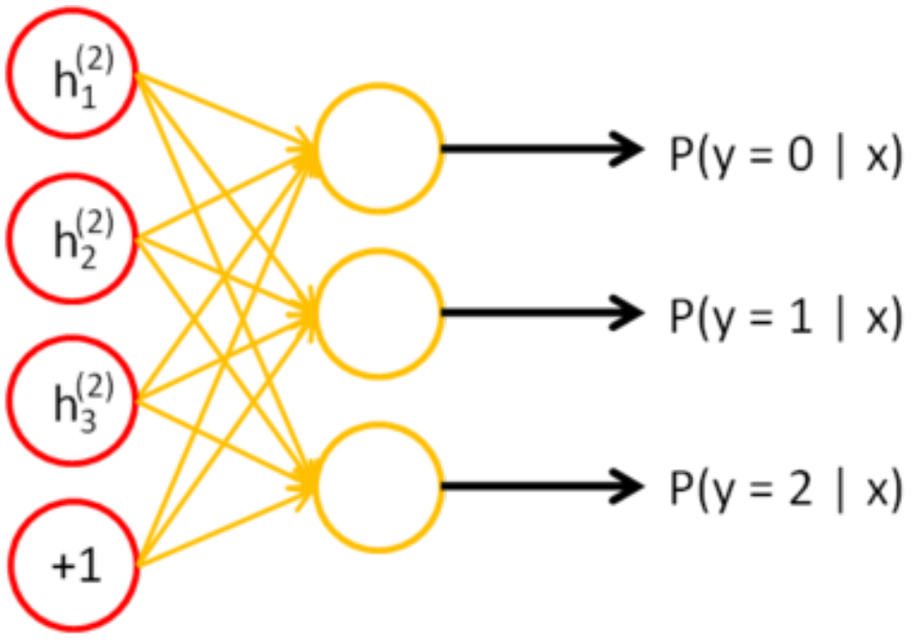
- Stack many (sparse) auto-encoders in succession and train them using greedy layer-wise training; Drop the decode output layer each time

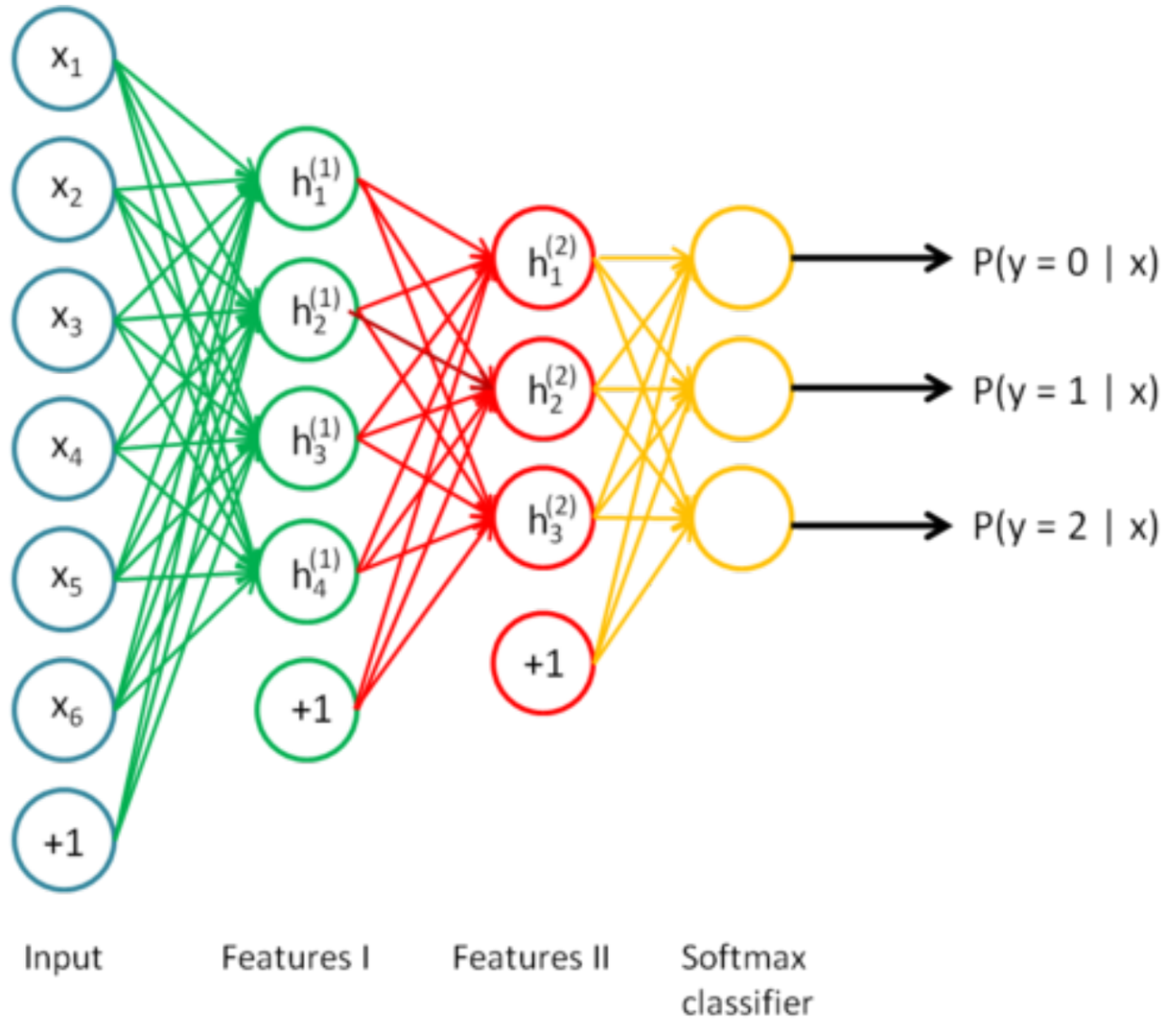




- Do supervised training on the last layer using final features. Then do supervised training on the entire network to fine-tune all weights

-





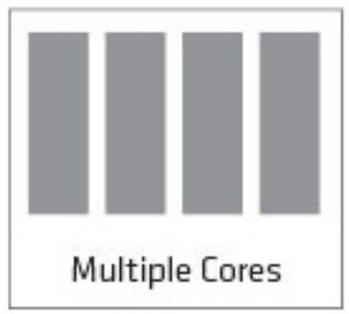
Deep learning since 2006

- Since 2006, the popularity of deep learning increases due to lots of developments:
  - New activation functions
  - Regularization methods
  - Initialization methods
  - Optimization techniques
  - Availability of big data
  - GPGPU (General-purpose computing on graphics processing units) to accelerate deep learning

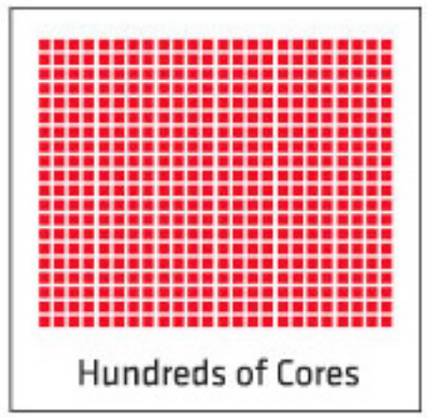
High performance computing

- The GPU acts as a co-processor and can accelerate applications due to its massively parallel processing power compared to the multicore design of CPUs.

- CUDA
  - Nvidia GeForce (3D gaming)
  - Nvidia Quadro (video rendering)
  - Nvidia Tesla (scientific computing)
- OpenCL
  - AMD FirePro
  - Intel MIC (Many Integrated Core Architecture)



CPU



GPU

Deep learning will change the internet

- Deep learning's particular strength is dealing with unsupervised data.

- Deep learning models are winning many prediction competitions and are state-of-the-art in image several recognition tasks and speech recognition.

- Deep learning will change the internet.

  - We'll be able to complete searches by voice, more efficiently search for images by the people or objects they contain, or use video analysis tools to see videos or ads that are more relevant to the content we're viewing.

Some neural networks architects

- Some networks have more than 100 millions weights and take a few months for training.

- https://www.jeremyjordan.me/convnet-architectures/

- https://adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html