Deep Learning

Standard Approaches

Tobias Croon, Heiko Guckes
Coach: Oliver Kroemer

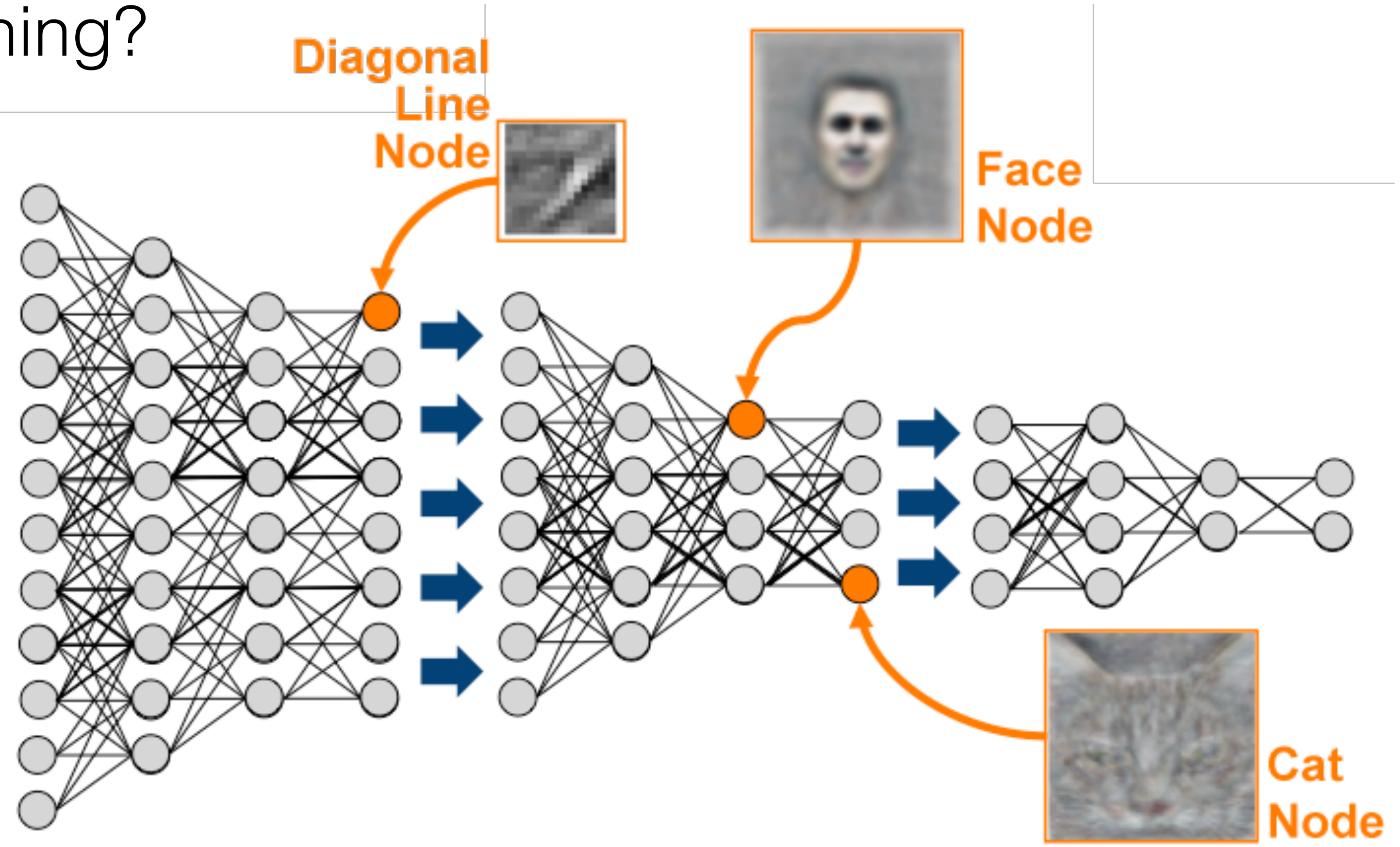
Motivation

- In machine learning, direct processing of raw input data is often not possible
- Features are needed for machine learning tasks like classification
- Engineering features is often hard
- Deep learning methods can automatically compute useful features of the input
- State-of-the-art in different areas like image classification or speech recognition

Introduction

- Shallow Neural Networks: Only one/few hidden layers
  Deep Learning: Multiple/many hidden layers

- Why **deep** learning?



http://theanalyticsstore.ie/deep-learning/

- Abstractions of abstractions

Overview

- Introduction
- Basic Components
- Topologies
- Recent Ideas
- Applications
- Summary

Overview

- Introduction
- Basic Components
  - Restricted Boltzmann Machines
  - Auto-Encoders
- Topologies
- Recent Ideas
- Applications
- Summary

Restricted Boltzmann Machines (RBMs)

- Special Case of **Markov Random Fields**:

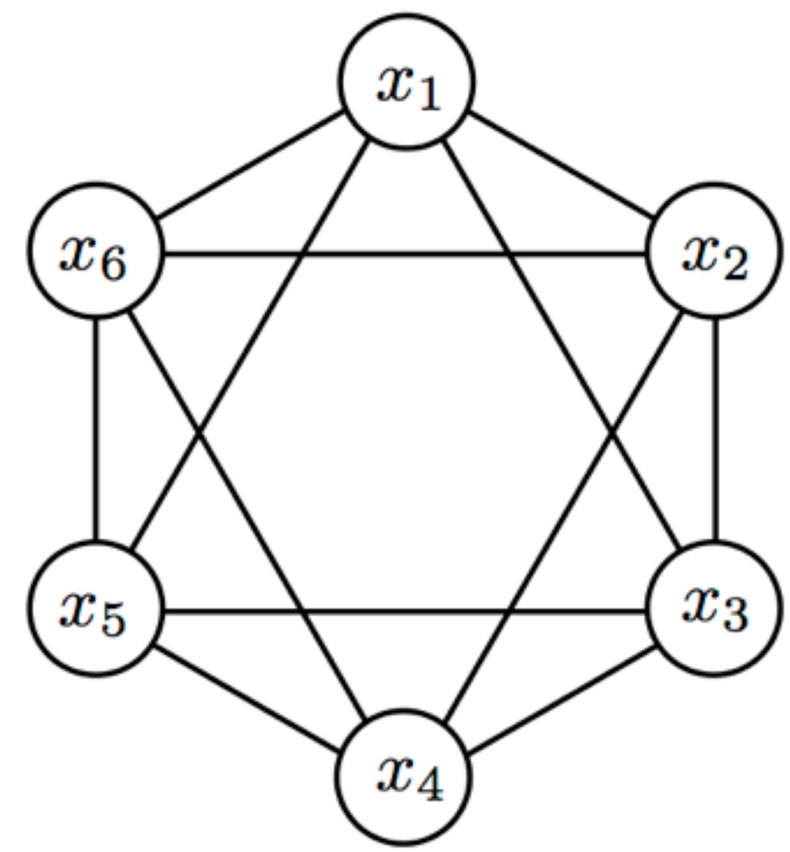
  - Energy Based Model (Energy depends on configuration of variables)

  - Joint probability:

$$p(x) = \frac{e^{-E_\theta(x)}}{Z_\theta}$$

with the normalizing partition function:

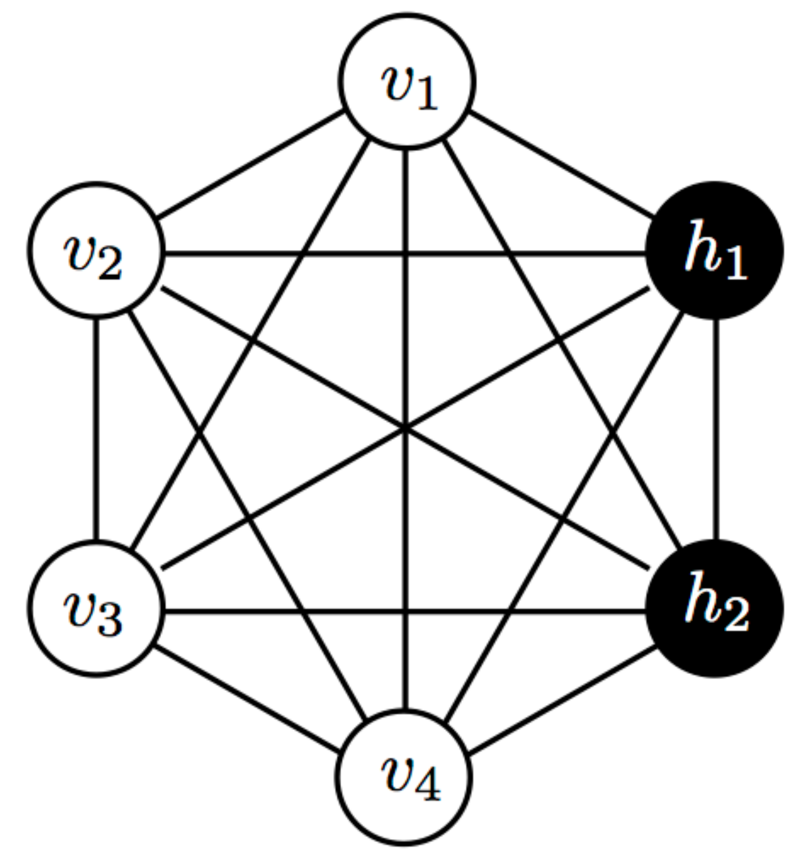
$$Z_\theta = \sum_x e^{-E_\theta(x)}$$



An Introduction to Restricted Boltzmann Machines; Fischer and Igel 2012
Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications

Restricted Boltzmann Machines (RBMs)

- Special Case of **Boltzmann Machines**:
  - Visible variables v_j (e.g. pixel image)
  - Hidden variables h_i (e.g. feature detectors)



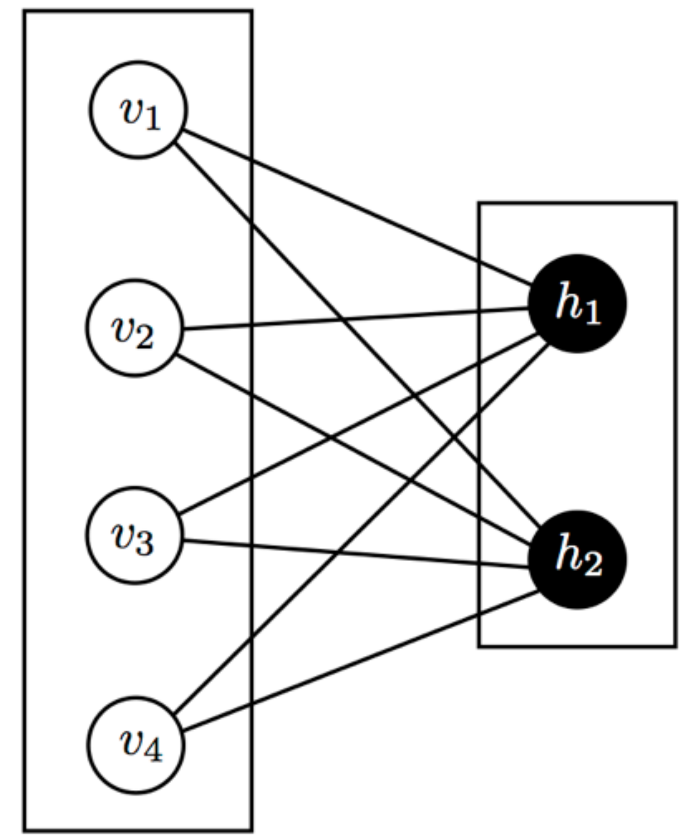
$$p(v) = \sum_h p(v,h) = \frac{1}{Z}\sum_h e^{-E(v,h)}$$

$$E(v,h) = -\sum_{i=1}^{n}\sum_{j=1}^{m} h_i w_{ij} v_j - \sum_{k=1}^{m}\sum_{l<k} v_k u_{kl} v_l - \sum_{k=1}^{n}\sum_{l<k} h_k y_{kl} h_l - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i$$

An Introduction to Restricted Boltzmann Machines; Fischer and Igel 2012
Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications

Restricted Boltzmann Machines (RBMs)

- Restriction:
  No connections between different visible or between different hidden variables



$$E(v,h) = -\sum_{i=1}^{n}\sum_{j=1}^{m} h_i w_{ij} v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i$$

- Bipartite structure (Layers)

- Independencies of conditional probabilities:

$$p(h|v) = \prod_{i=1}^{n} p(h_i|v) \qquad p(v|h) = \prod_{j=1}^{m} p(v_j|h)$$

An Introduction to Restricted Boltzmann Machines; Fischer and Igel 2012
Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications

Training RBMs

- Maximize Log-Likelihood of Parameters θ

$$\ln \mathcal{L}(\theta|v) = \ln p(v|\theta) = \ln \frac{1}{Z} \sum_{\boldsymbol{h}} e^{-E(v,h)}$$
$$= \ln \sum_{\boldsymbol{h}} e^{-E(v,h)} - \ln \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(v,h)}$$

- Gradient of Log-likelihood:

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{v})}{\partial \boldsymbol{\theta}} = \frac{\partial}{\partial \boldsymbol{\theta}} \left(\ln \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \right) - \frac{\partial}{\partial \boldsymbol{\theta}} \left(\ln \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \right)$$
$$= -\frac{1}{\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{\theta}} + \frac{1}{\sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}} \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{\theta}}$$
$$= -\sum_{\boldsymbol{h}} p(\boldsymbol{h} \mid \boldsymbol{v}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{\theta}}$$

An Introduction to Restricted Boltzmann Machines; Fischer and Igel 2012
Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications

Training RBMs

- Gradient of Log-likelihood:

$$\frac{\partial \ln \mathcal{L}(\theta|v)}{\partial \theta} = -\sum_h p(h|v) \frac{\partial E(v,h)}{\partial \theta} + \sum_{v,h} p(v,h) \frac{\partial E(v,h)}{\partial \theta}$$

- Gradient w.r.t. weights w_{ij}

$$\frac{\partial \ln \mathcal{L}(\theta|v)}{\partial w_{ij}} = p(h_i = 1|v)v_j - \sum_v p(v)p(h_i = 1|v)v_j$$

$$\frac{\partial \ln \mathcal{L}(\theta|v)}{\partial w_{ij}} = \langle h_i v_j \rangle_{data} - \langle h_i v_j \rangle_{model}$$

- Second part is difficult to obtain $\Rightarrow$ Approximation

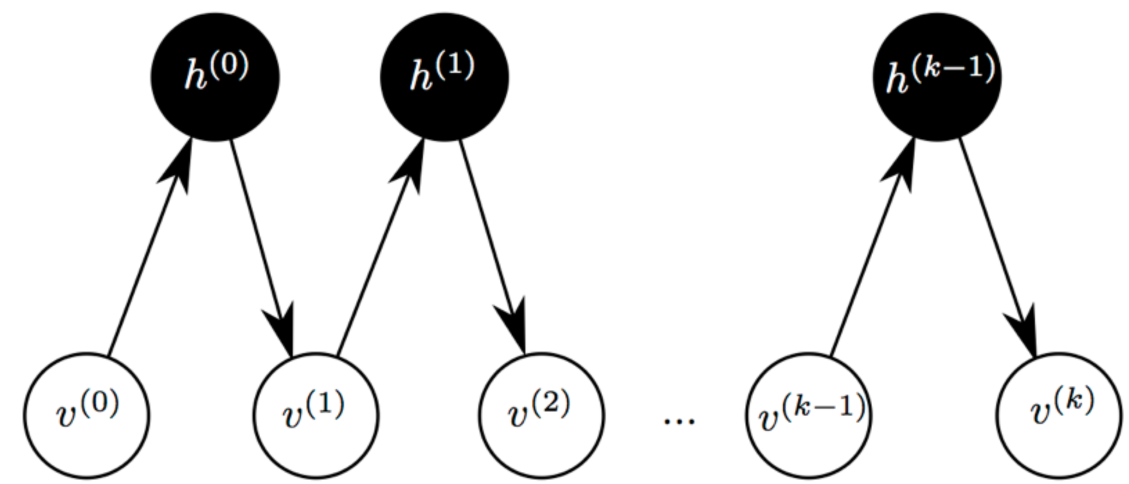
An Introduction to Restricted Boltzmann Machines; Fischer and Igel 2012
Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications

Training RBMs

- Approximation by Gibbs sampling

$$p(h_i = 1|v) = \sigma\left(\sum_{j=1}^{m} w_{ij} v_j + c_i\right)$$

$$p(v_j = 1|h) = \sigma\left(\sum_{i=1}^{n} w_{ij} h_i + b_j\right)$$



  - inefficient

- Contrastive Divergence (CD_k)
  - computes only k steps of the chain (starting from training sample as $v^{(0)}$
  - not really following a gradient (but works)
  - computes reconstruction error $\Rightarrow$ similar to Auto-Encoder (following)

An Introduction to Restricted Boltzmann Machines; Fischer and Igel 2012
Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications

Training RBMs

- Variations for Contrastive Divergence:
  - Persistent Contrastive Divergence (PCD):
    - no reinitialization of Gibbs chain
  - Fast Persistent Contrastive Divergence (FPCD):
    - like PCD but with additional fast parameters for sampling

$$\tilde{p}(h_i = 1|v) = \sigma\left(\sum_{j=1}^{m}(w_{ij} + w_{ij}^f)v_j + (c_i + c_i^f)\right)$$

$$\tilde{p}(v_j = 1|h) = \sigma\left(\sum_{i=1}^{n}(w_{ij} + w_{ij}^f)h_i + (b_j + b_j^f)\right)$$

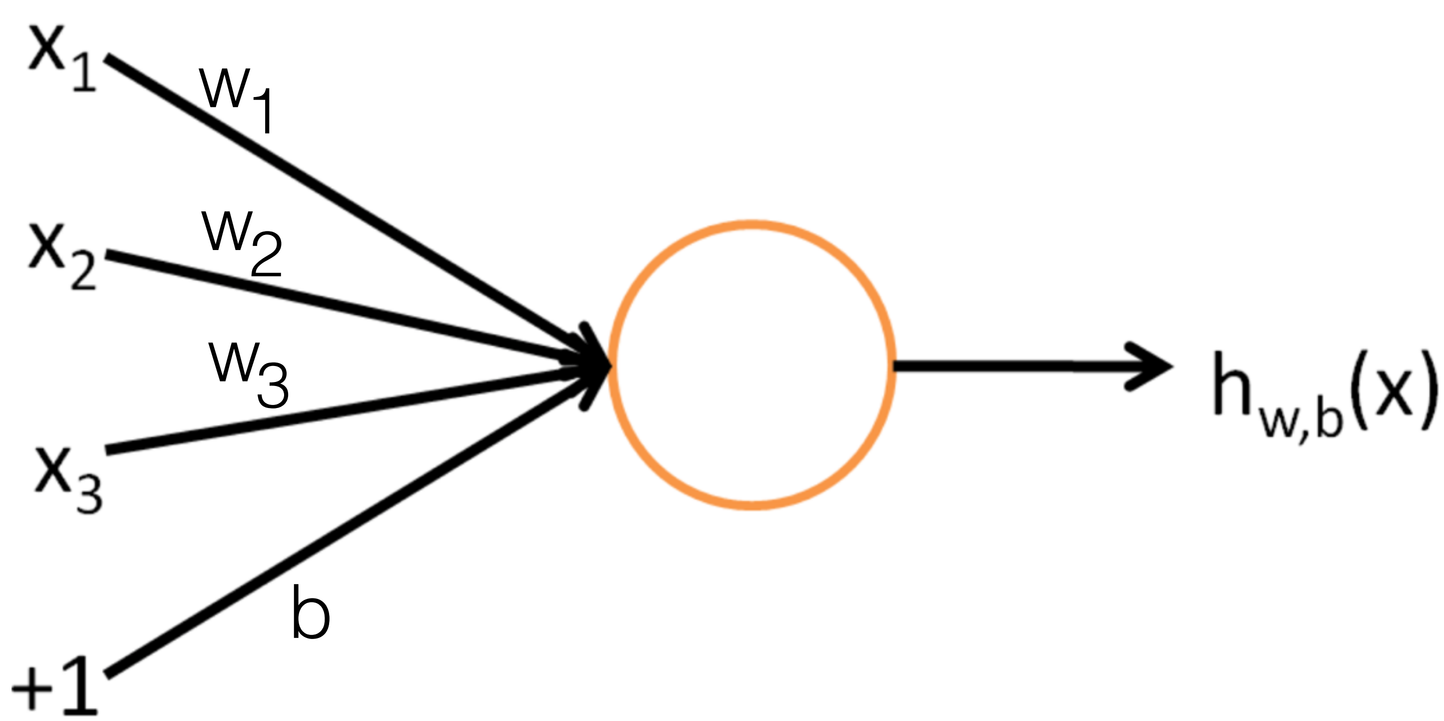
    - faster update of fast parameters with weight decay

An Introduction to Restricted Boltzmann Machines; Fischer and Igel 2012
Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications

From Neurons to Auto-Encoders

- Single Neuron
- Sigmoid Activation Function
- Multilayer Feedforward Neural Network
- Backpropagation Sketch
- Backpropagation Building Blocks (Equations)
- Finally: Auto-Encoder

Single Neuron



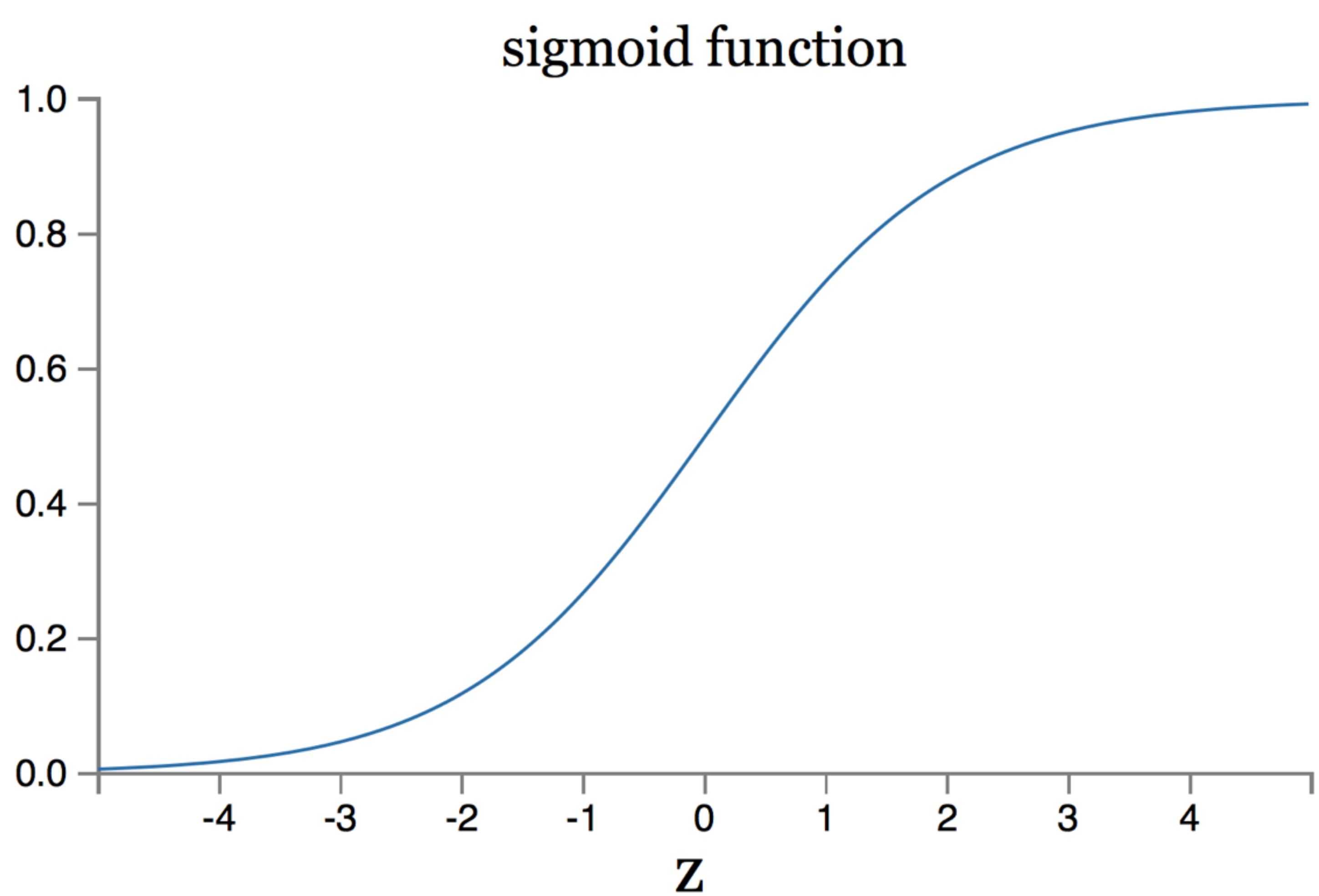
$$h_{W,b}(x) = f(W^T x) = f(\textstyle\sum_{i=1}^{3} W_i x_i + b)$$

$$\begin{aligned} z^{(l+1)} &= W^{(l)} a^{(l)} + b^{(l)} \\ a^{(l+1)} &= f(z^{(l+1)}) \end{aligned}$$

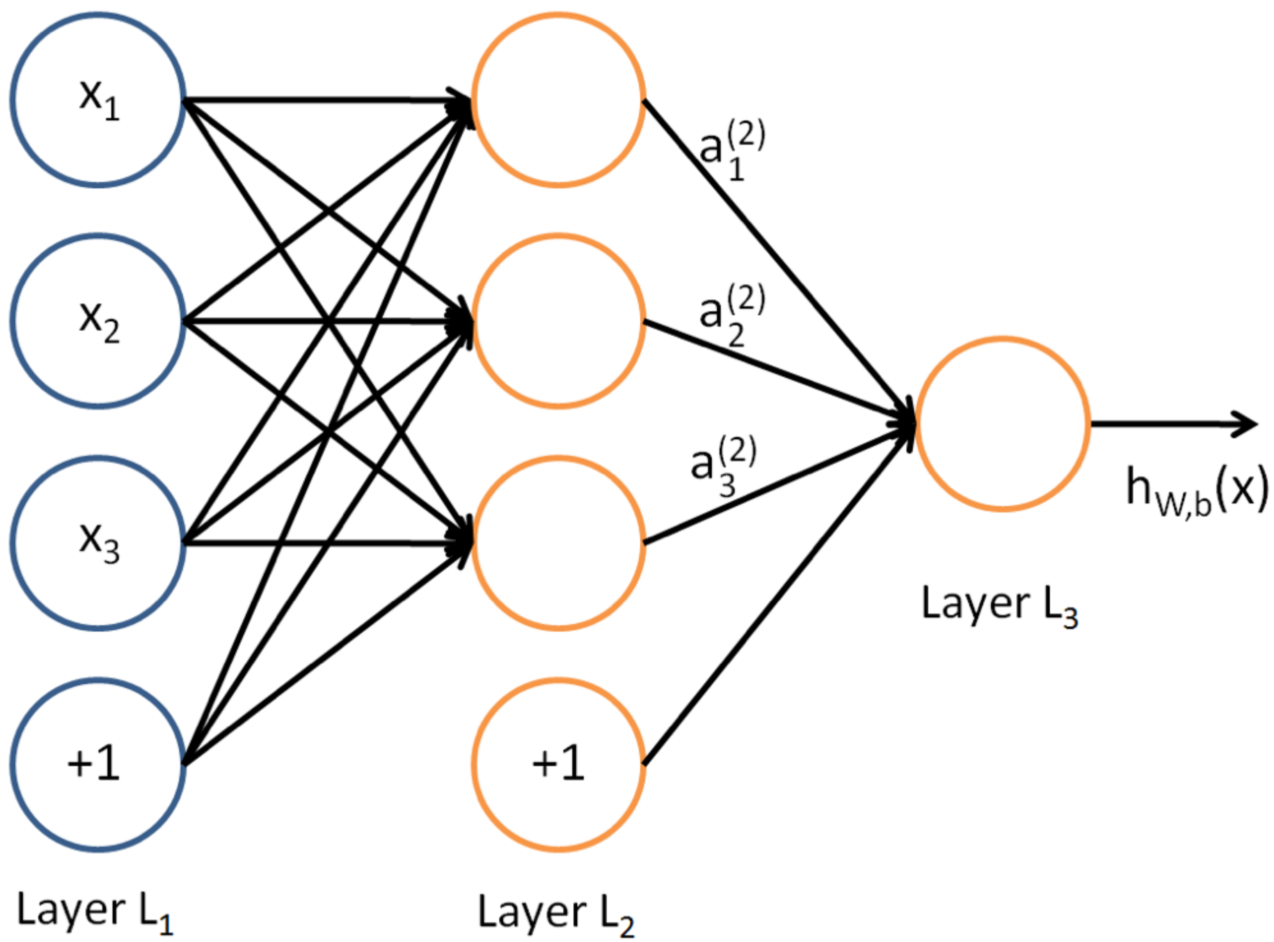
from: http://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf

Sigmoid Activation Function

- Squashes values into [0,1]
- Saturates for small and large values

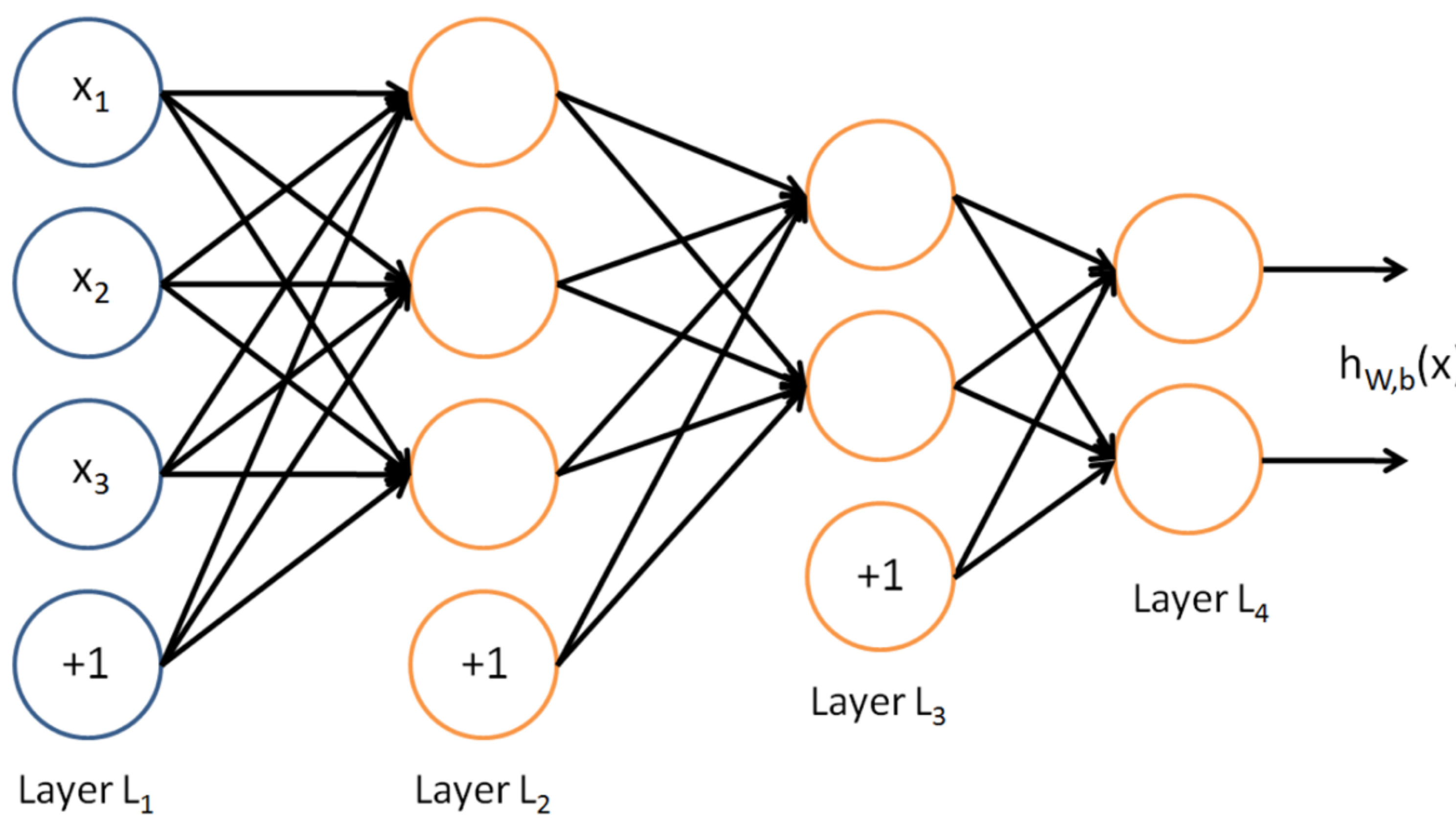


Multi Layer Feedforward Neural Network (NN)



How to train a Neural Network?

- How does changing a single weight influence the global cost function? $$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x, y)$$
- If we know this we can do Gradient Descent



Backpropagation Sketch

- Initialize all parameters randomly near zero
- Given an example
  - Compute all the activations in the network (forward pass)
  - Compute error responsibility for final layer
  - Backpropagate the error for hidden layers
  - Update all parameters

Backpropagation Building Blocks

- An equation for ...
  - ... measuring the total cost
  - ... the error in the output layer
  - ... the error of one layer in terms of the error in the next layer
  - ... the rate of change of the cost with respect to any bias in the network
  - ... the rate of change of the cost with respect to any weight in the network

Total Cost

- Goal: Minimize average error over all m training samples

$$
\begin{aligned}
J(W,b) &= \left[\frac{1}{m}\sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)})\right] \\
&= \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{W,b}(x^{(i)})-y^{(i)}\right\|^2\right)\right]
\end{aligned}
$$

Error = Average SSD + Weight Decay

Output Layer Error

- For each output unit in the last layer

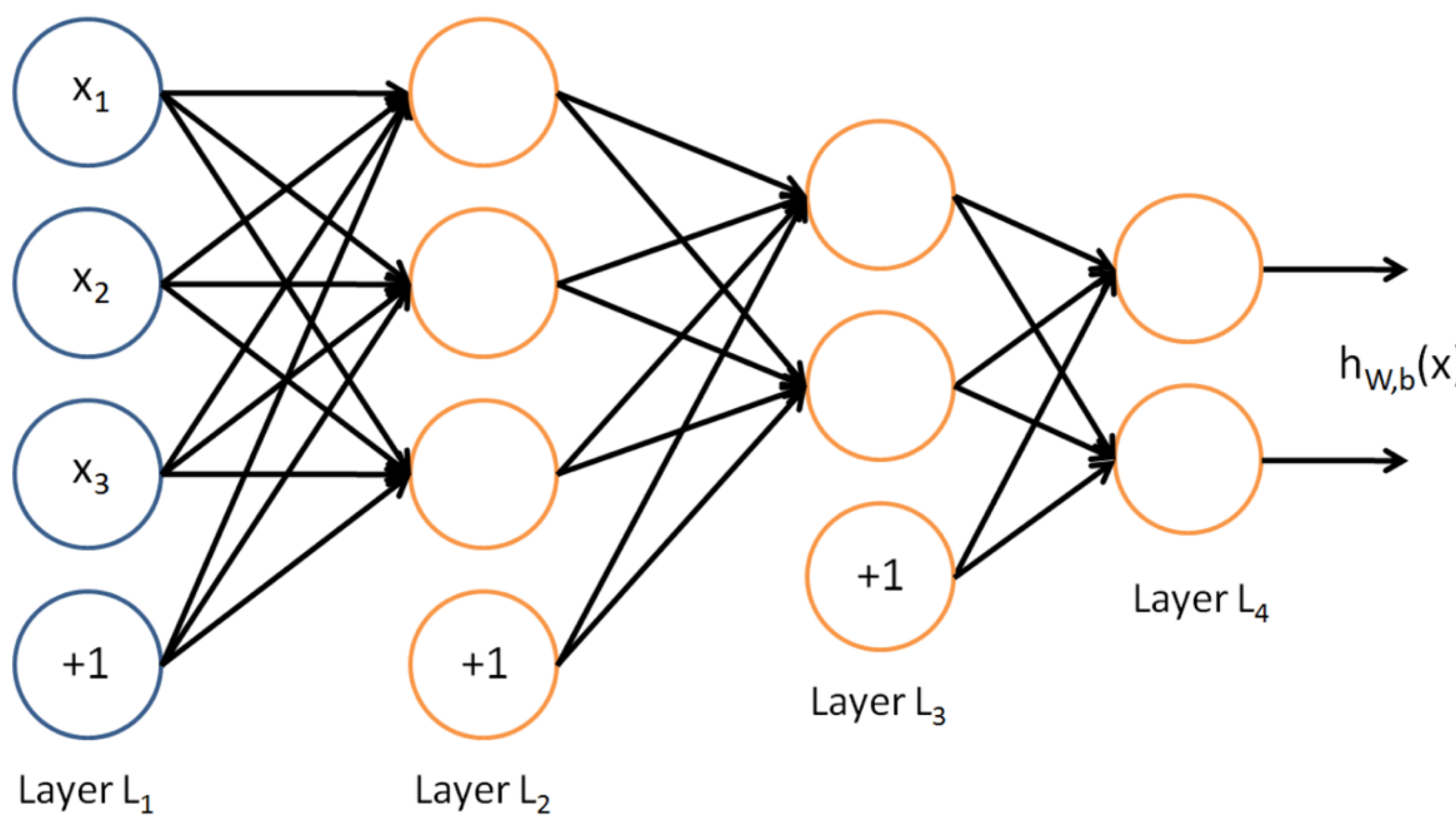
$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \;\; \frac{1}{2} \left\| y - h_{W,b}(x) \right\|^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)})$$

- For the sigmoid activation function

$$f'(z_i^{(l)}) = a_i^{(l)}(1 - a_i^{(l)})$$

Backpropagate

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)} \right) f'(z_i^{(l)})$$



Desired partial derivatives

- How the weights/biases affect the cost function

$$
\begin{aligned}
\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x,y) &= a_j^{(l)} \delta_i^{(l+1)} \\
\frac{\partial}{\partial b_{i}^{(l)}} J(W,b;x,y) &= \delta_i^{(l+1)}.
\end{aligned}
$$

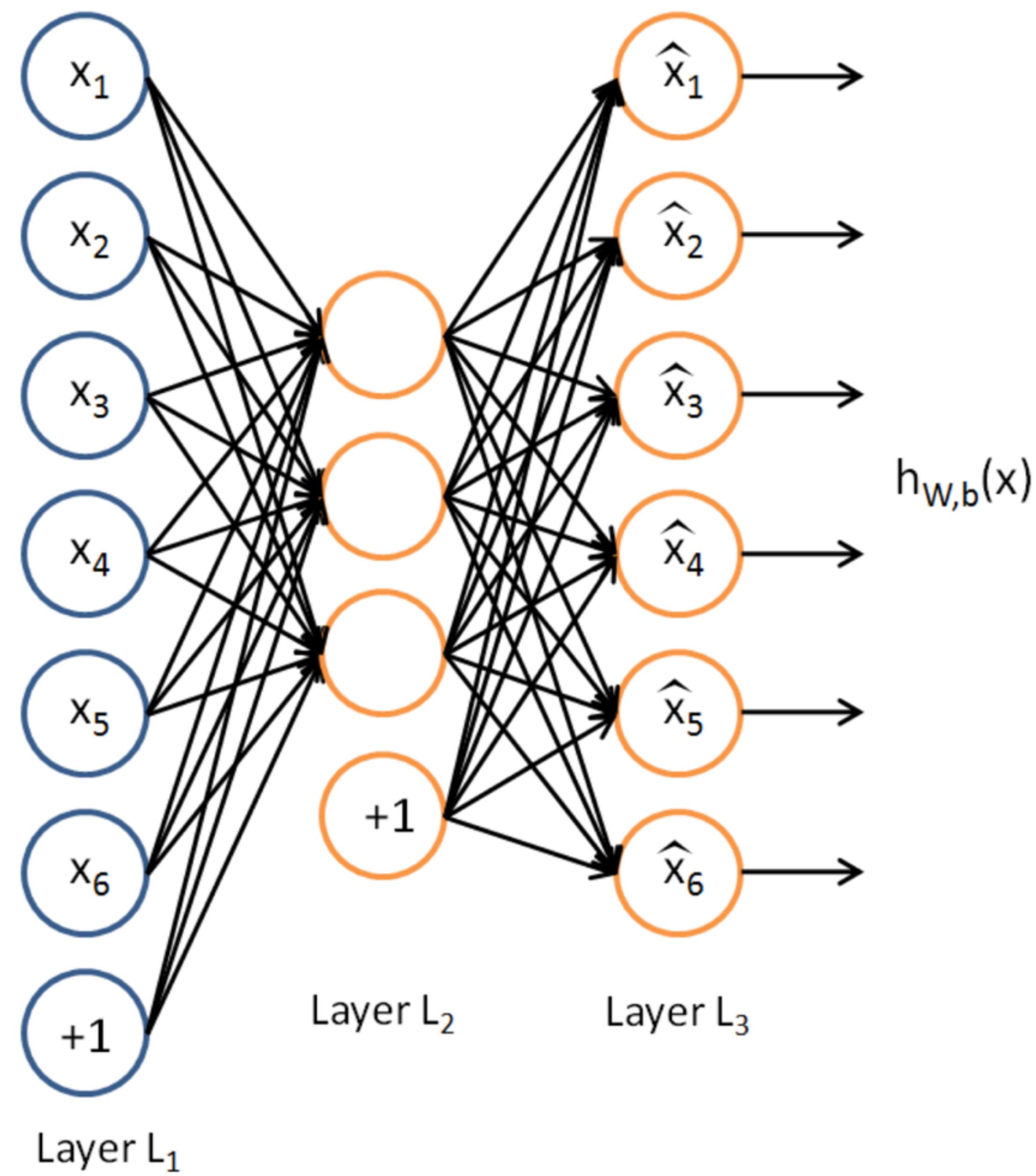
Weight Update

- One iteration of gradient descent

$$
\begin{aligned}
W_{ij}^{(l)} &:= W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W,b) \\
b_{i}^{(l)} &:= b_{i}^{(l)} - \alpha \frac{\partial}{\partial b_{i}^{(l)}} J(W,b)
\end{aligned}
$$

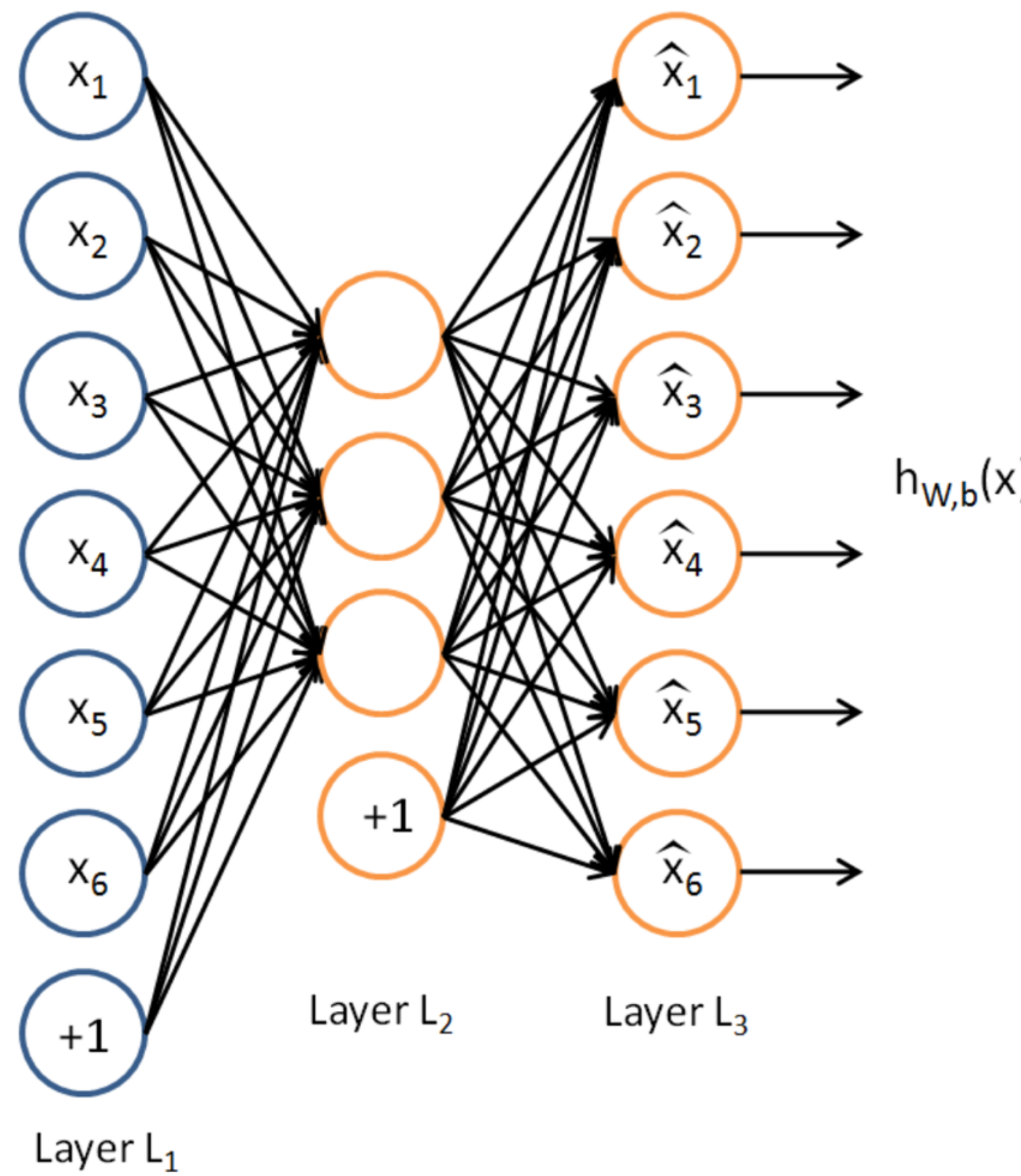
Auto-Encoder

- Unsupervised
- Output similar to Input
  - $h_{W,b}(x) \approx x$
- Dimensionality reduction
- Tied weights



Auto-Encoder Variants

- Sparse
  - Apply regularization to hidden unit activation
  - Can use more hidden units (overcomplete)
  - More robust to small changes and noise
- Denoising
  - Add noise to input data
  - Learns to remove additional noise



RBM vs. Auto-Encoder

	RBM	AE
Generative	√	Denoising*
Non Linear	√	√
Dimensionality Reduction	√	√
Remove Noise	√	√
Trained with Backprop		√
Use as feedforward NN	√	√
Undirected Connections	√	

*Learning Deep Architectures for AI; Yoshua Bengio; 2009

Overview

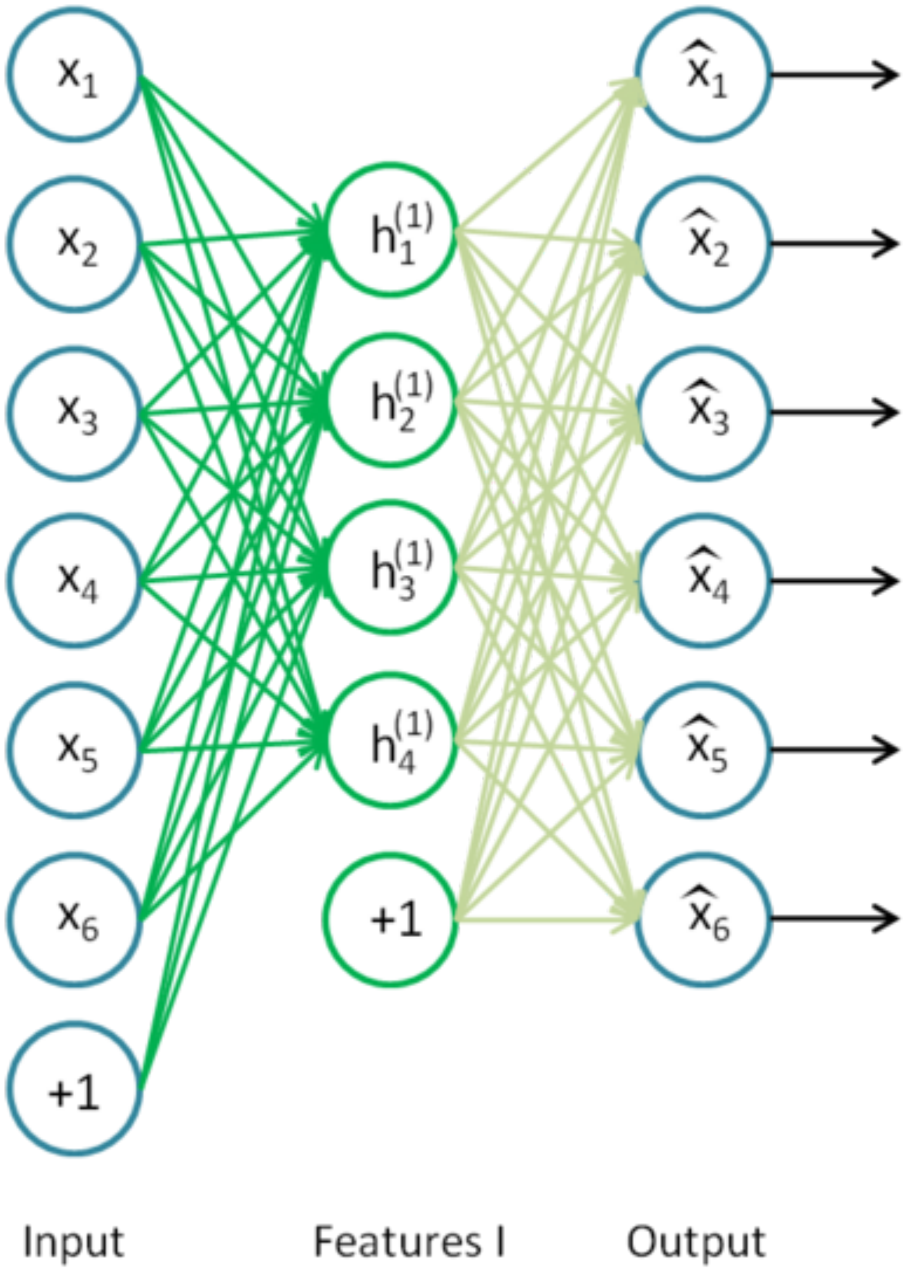
- Introduction
- Basic Components
- Topologies
- Recent Ideas
- Applications
- Summary

Topologies

- Composed of fully connected layers:
  - Stacked Auto-Encoders
  - Deep Belief Networks
- Composed of partially/locally connected layers:
  - Convolutional Neural Networks

Stacked Auto-Encoder

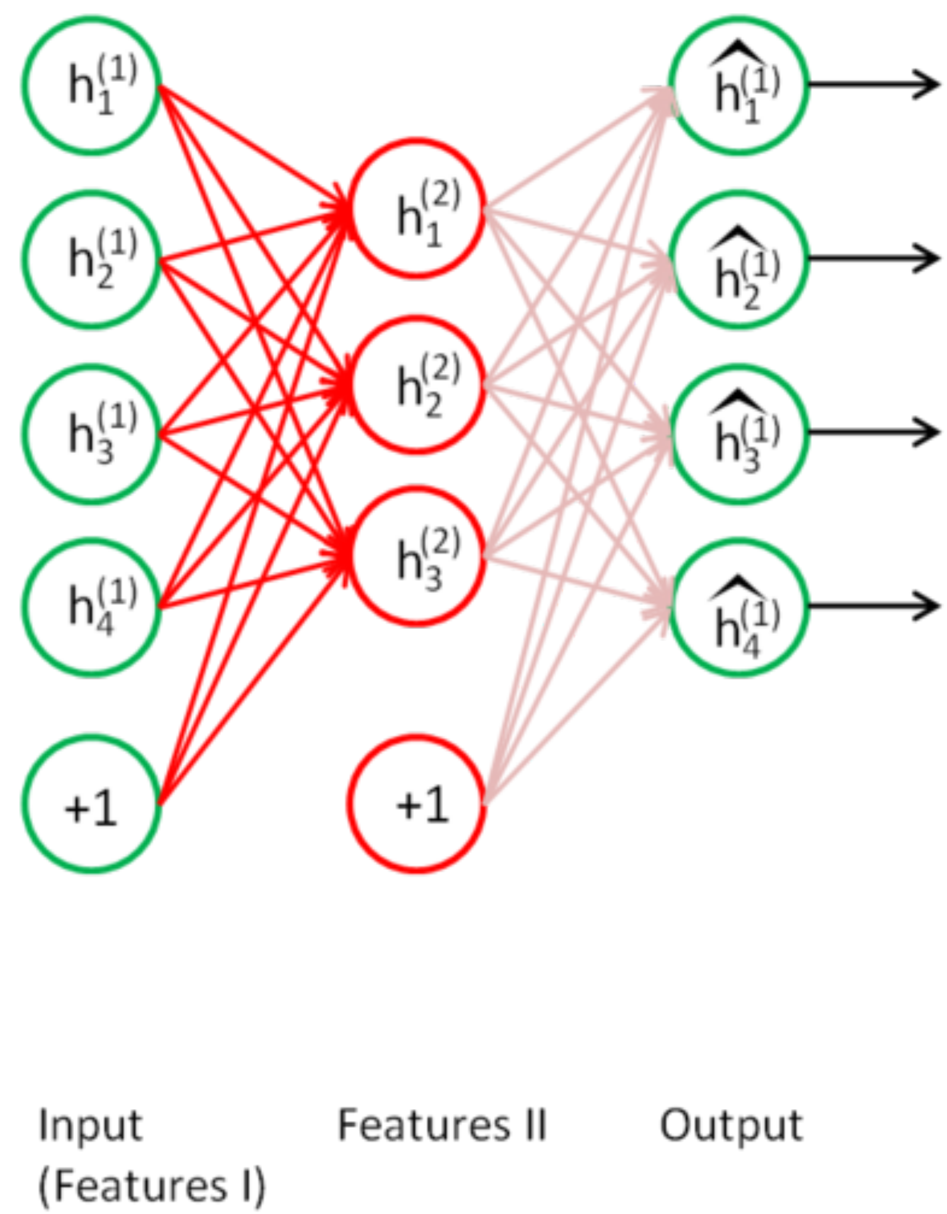
- Goal: Classify handwritten digits
- Learn primary features on raw input



from http://deeplearning.stanford.edu/wiki/index.php/Stacked_Autoencoders

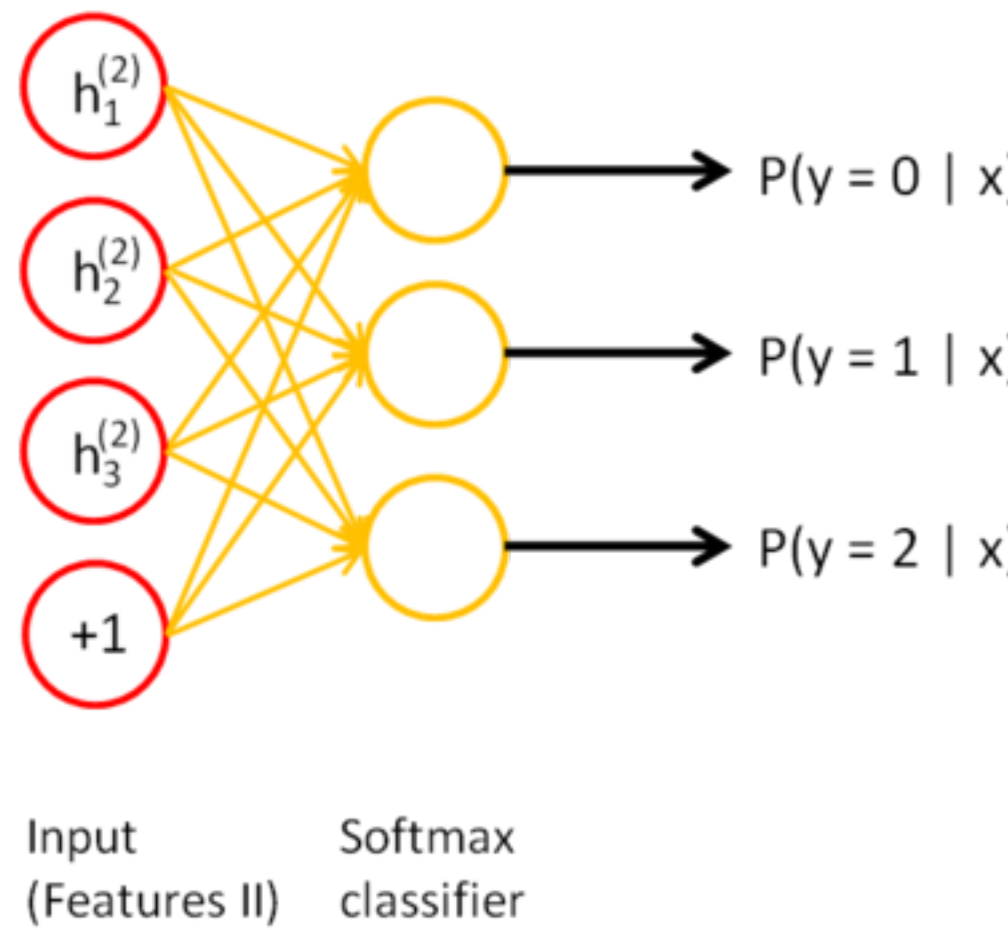
Stacked Auto-Encoder

- Learn secondary features
- Use primary features as input

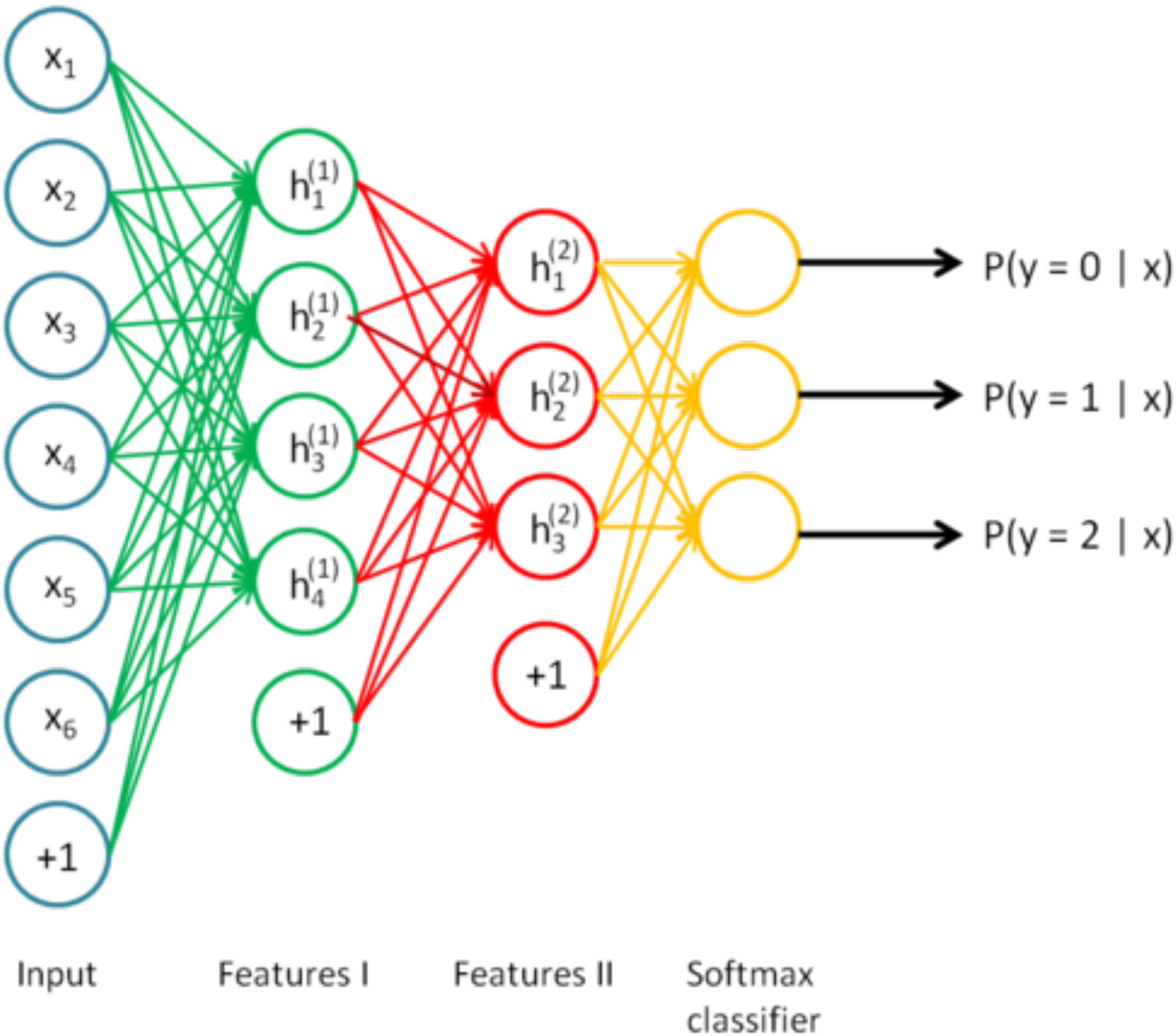


Stacked Auto-Encoder

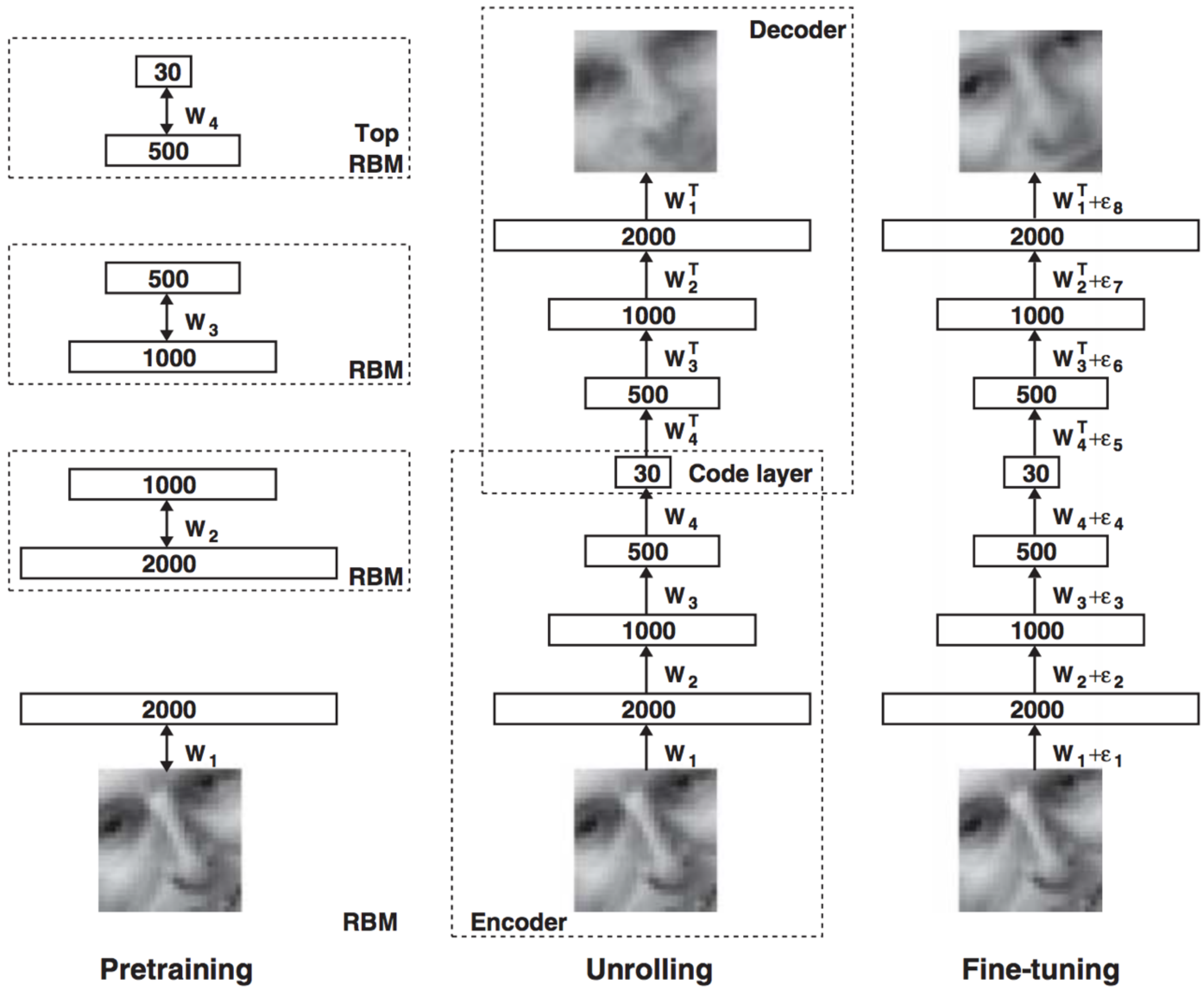
- Use secondary features as input to softmax classifier
- Map secondary features to digit labels



Stacked Auto-Encoder

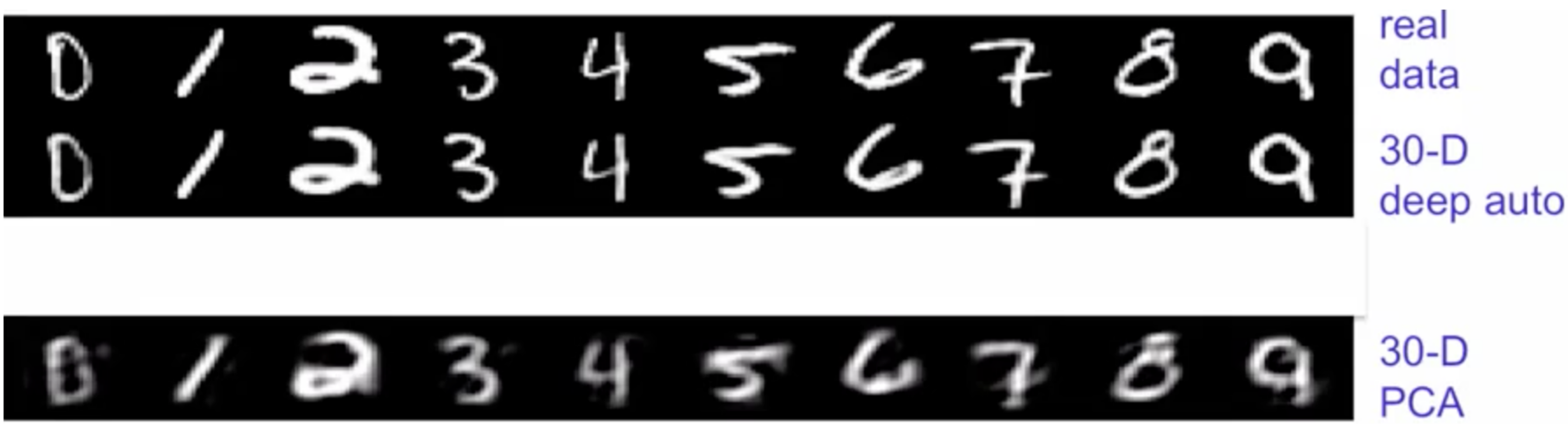


Stacked Auto-Encoder



G. E. Hinton and R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” Science, vol. 313, no. 5786, pp. 504–507, 2006.

3 Layer Auto-Encoder vs. PCA

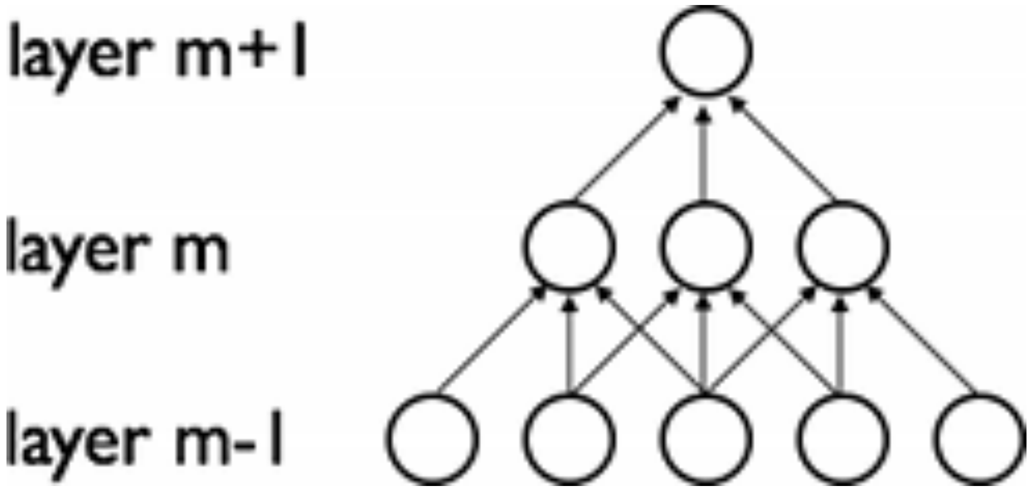


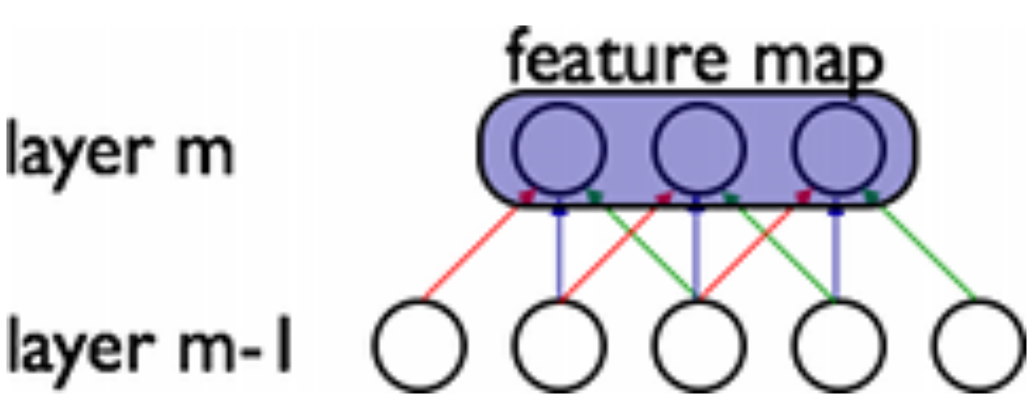
Why pre-training helps?

- Learning layer by layer scales very well
- Useful weights initially for Backpropagation
- Backpropagation only requires local search
- Labeled data is only used for fine tuning
- Can be seen as regularizer or prior

Convolutional Neural Networks (CNNs)

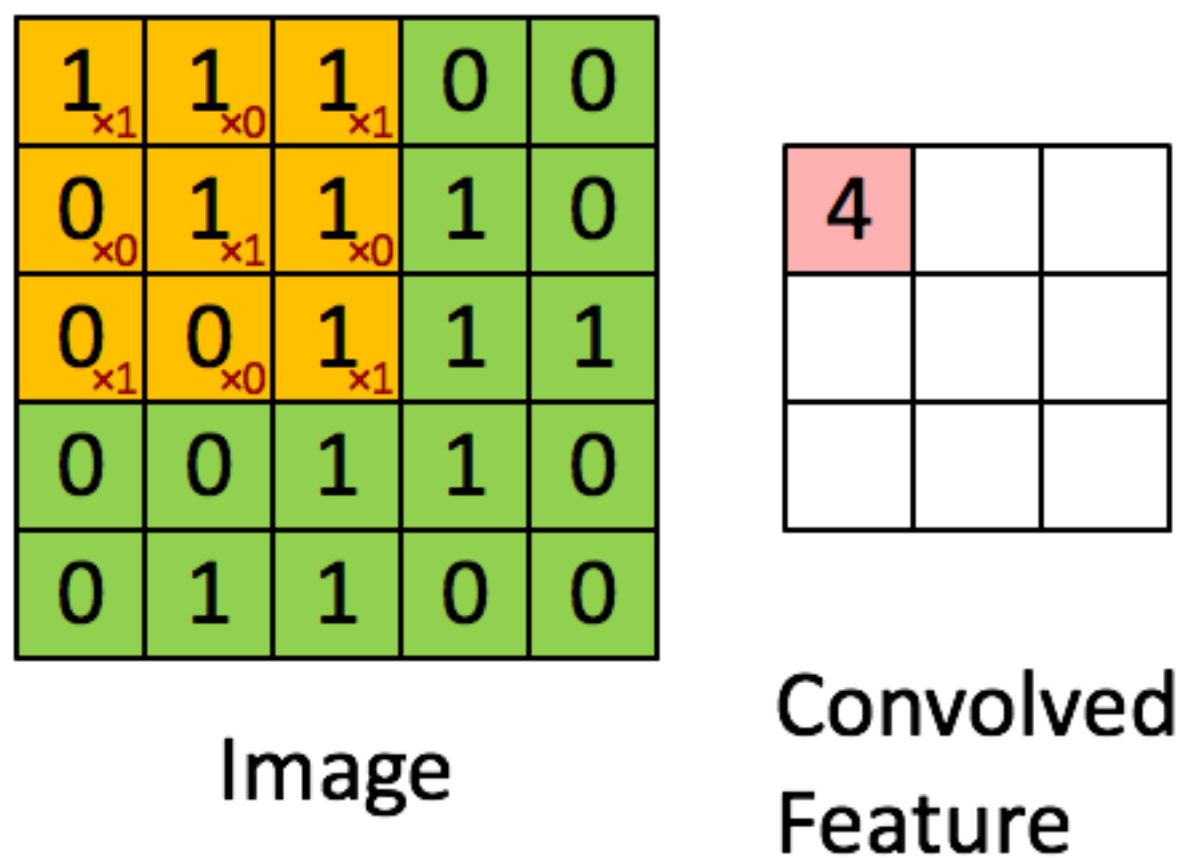
- Biologically inspired
- Sparse connectivity
  - Only local connectivity
  - One neuron spans whole area
- Shared weights
  - Images have stationary property, so same statistics everywhere
  - Enforce same weights





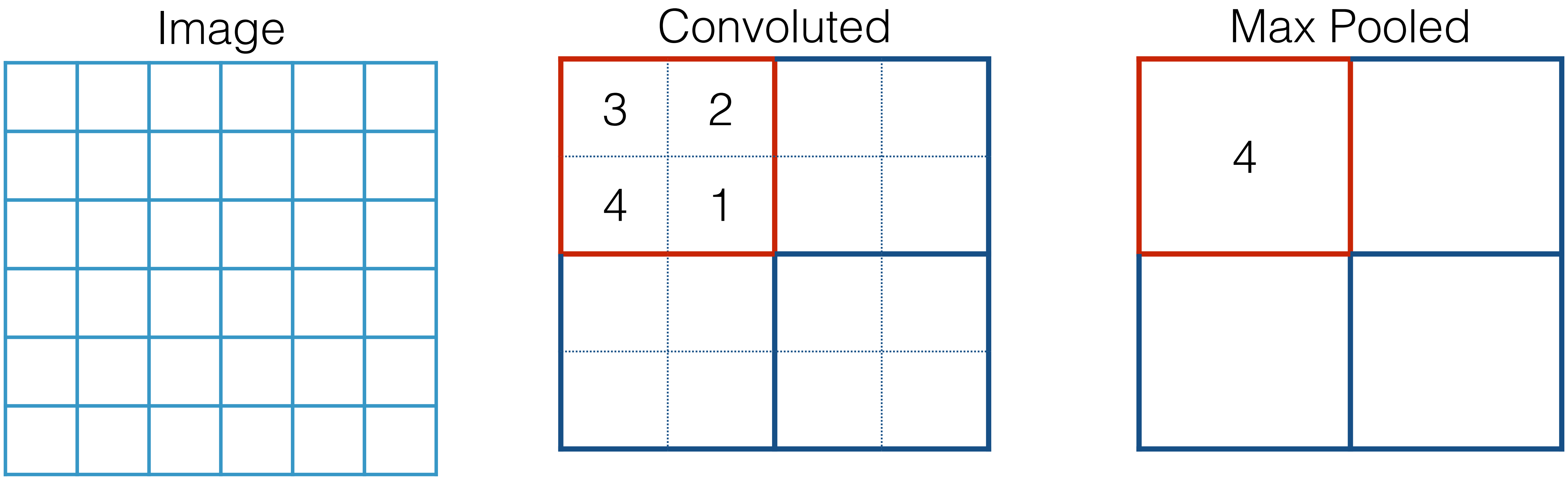
from http://deeplearning.net/tutorial/lenet.html

Convolution



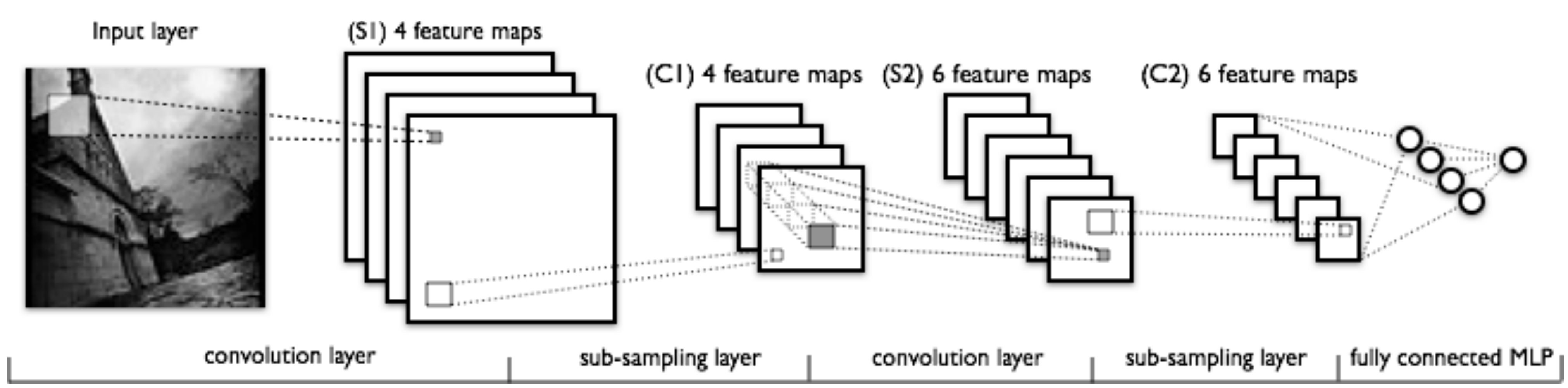
Pooling

- More robust to small changes in features
- Less computations
- Reduce overfitting



Convolutional Neural Network (CNN)

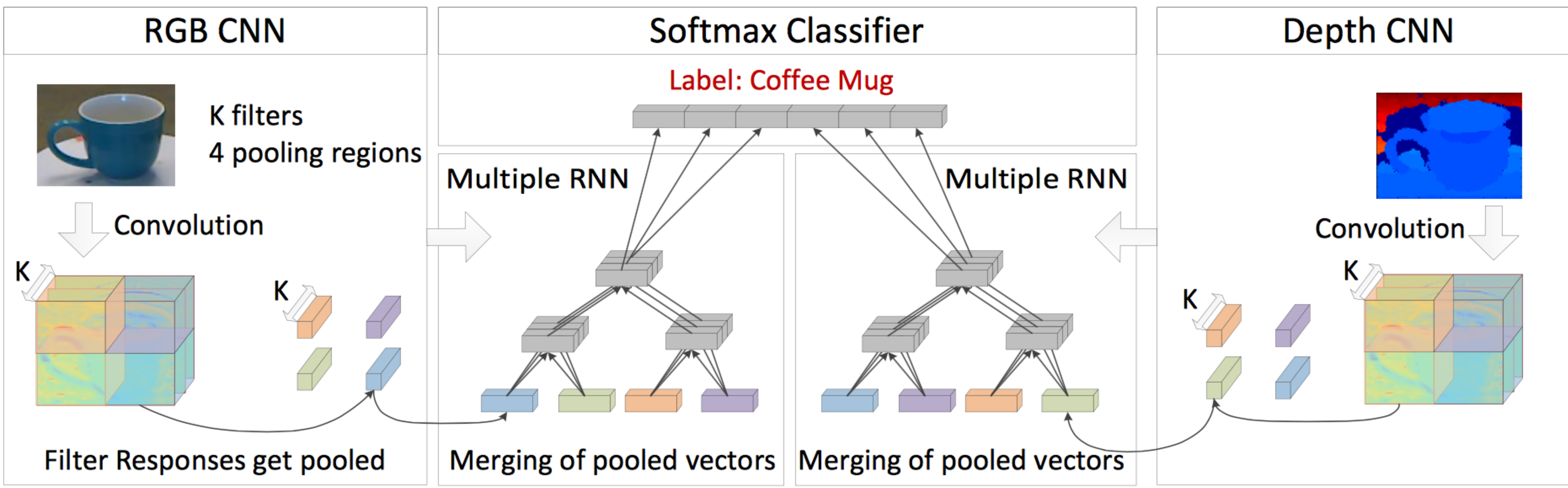
- Structure with different alternating layers
- Convolutional layers
    - topographic structure (fixed 2d position + receptive field)
- Sub-sampling layers
    - Max-Pooling
- traditional MLP at the output



from http://deeplearning.net/tutorial/lenet.html#lenet

Multiple Input Modalities

- Combine different types of input data
- Combine data later in the hierarchy
- Lower levels learn modality specific features



Convolutional-Recursive Deep Learning for 3D Object Classification. Socher et. al (NIPS 2012 665-673)

Overview

- Introduction
- Basic Components
- Topologies
- Recent Ideas
- Applications
- Summary

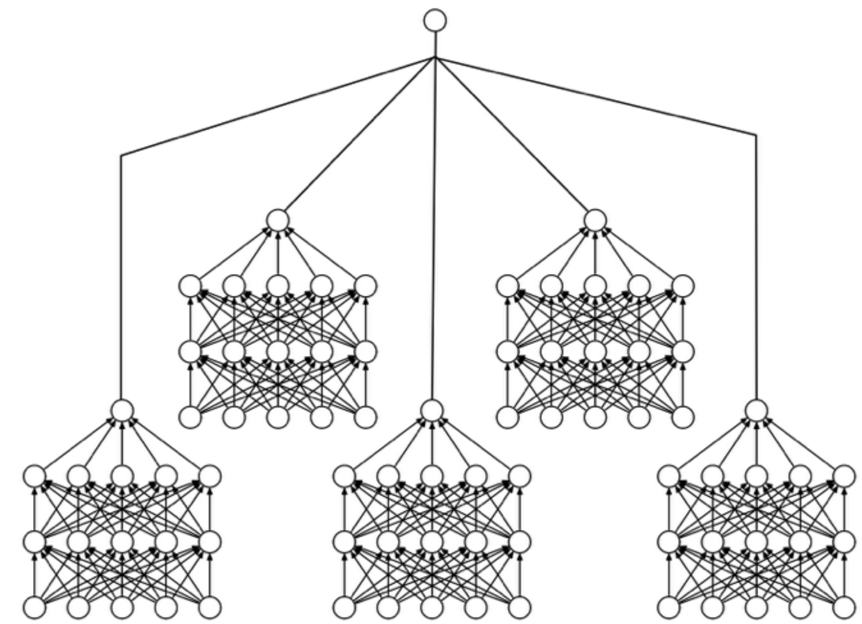
Drop Out

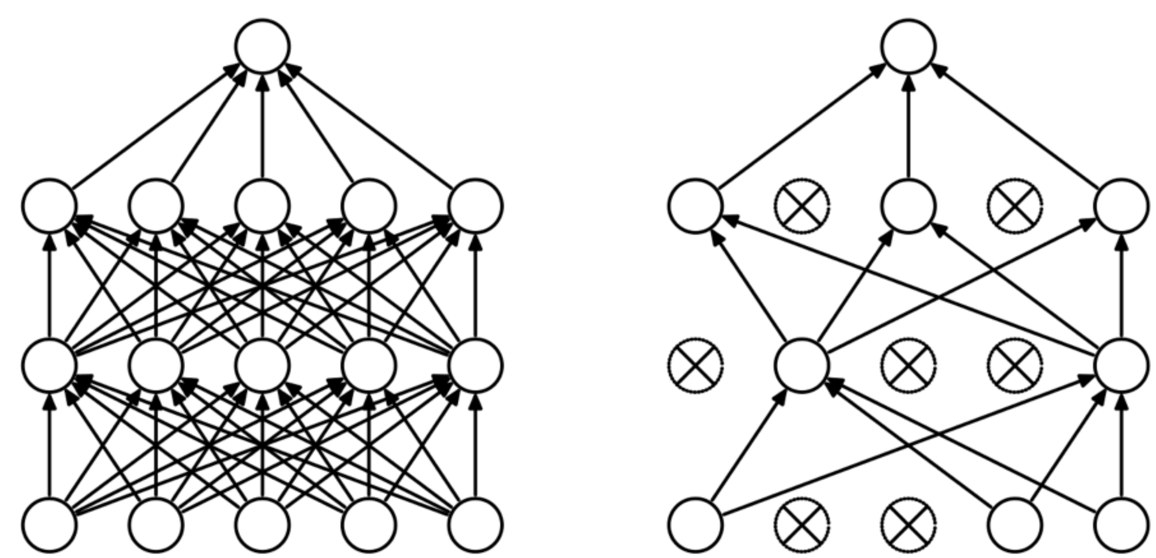
- **Idea:** Reduce overfitting by averaging the outputs of multiple separately trained networks

- **Problem:** Too many parameters to learn, too slow, too much training data required

- **Solution:** Use only one network, but slightly change the structure during training

- Randomly deactivate nodes during training



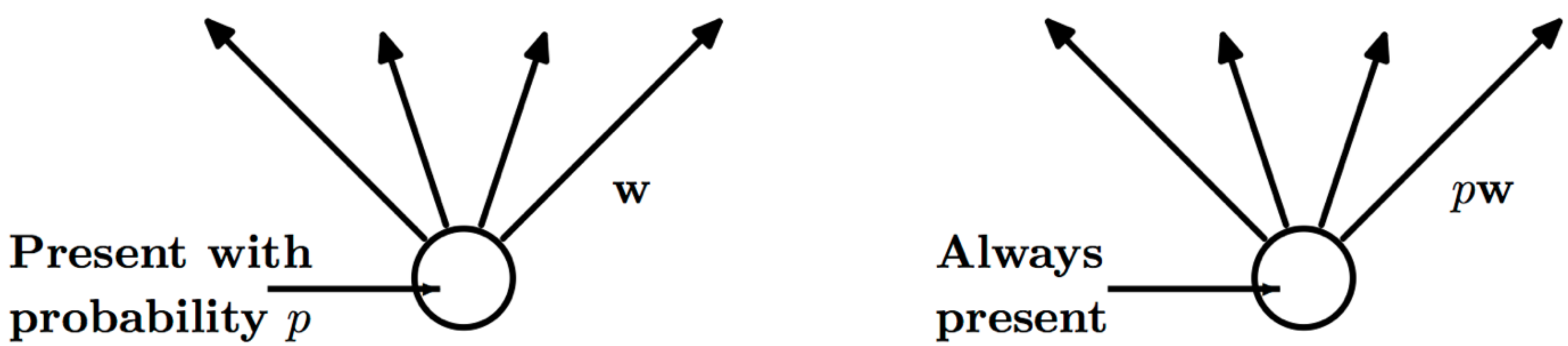


(a) Standard Neural Net (b) After applying dropout.

Dropout: A Simple Way to Prevent Neural Networks from Overfitting; Srivastava et al. 2014
Journal of Machine Learning Research

Drop Out

- Randomly deactivate nodes during training
    - Train "thinned" networks (one randomly thinned network per training case)
    - All thinned networks share all weights
    - Combine all thinned networks at test time



(a) At training time (b) At test time

Dropout: A Simple Way to Prevent Neural Networks from Overfitting; Srivastava et al. 2014 Journal of Machine Learning Research

Drop Out

- Results:
  - Increased robustness, prevents from overfitting, lower generalization error
  - State-of-the-art results on image data sets
  - Improvement on speech data set
  - Text data set: improvement smaller compared to vision and speech data sets

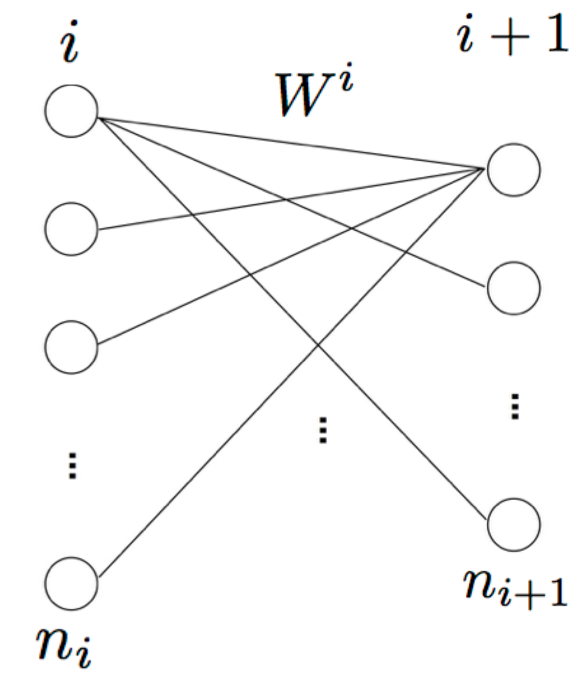
Dropout: A Simple Way to Prevent Neural Networks from Overfitting; Srivastava et al. 2014
Journal of Machine Learning Research

Xavier Initialization

- Avoid amplifying or attenuating the signals in the network
- Normalized initialization of the weights W^i
- Desired properties:

$$\forall(i,i'), Var[z^i] = Var[z^{i'}]$$

$$\forall(i,i'), Var\left[\frac{\partial Cost}{\partial s^i}\right] = Var\left[\frac{\partial Cost}{\partial s^{i'}}\right]$$



$$s^i = z^i W^i + b^i$$

$$z^{i+1} = f(s^i)$$

- Variances of weights:

$$\forall i, \quad n_i\, Var[W^i] = 1 \quad \Rightarrow Var[W^i] = \frac{1}{n_i}$$

$$\forall i, \quad n_{i+1}\, Var[W^i] = 1 \quad \Rightarrow Var[W^i] = \frac{1}{n_{i+1}}$$

Understanding the difficulty of training deep feedforward neural networks; Glorot and Bengio 2010
International conference on artificial intelligence and statistics

Xavier Initialization

- Compromise:

$$\forall i, \quad Var[W^i] = \frac{2}{n_i + n_{i+1}}$$

- Suggestion for implementation (Normalized initialization):

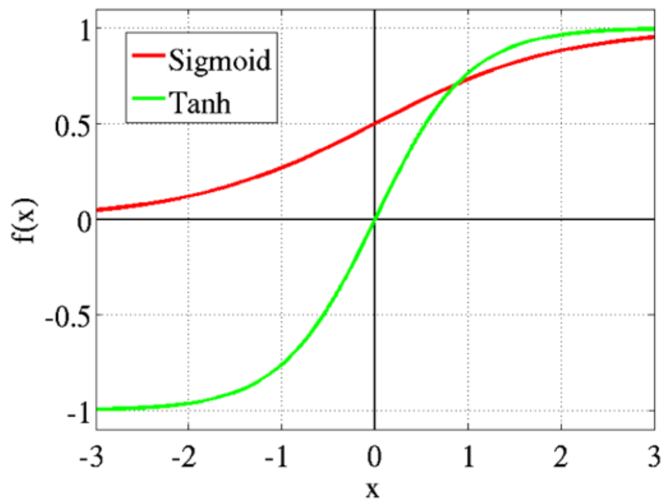
$$W \sim U\Big[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\Big]$$

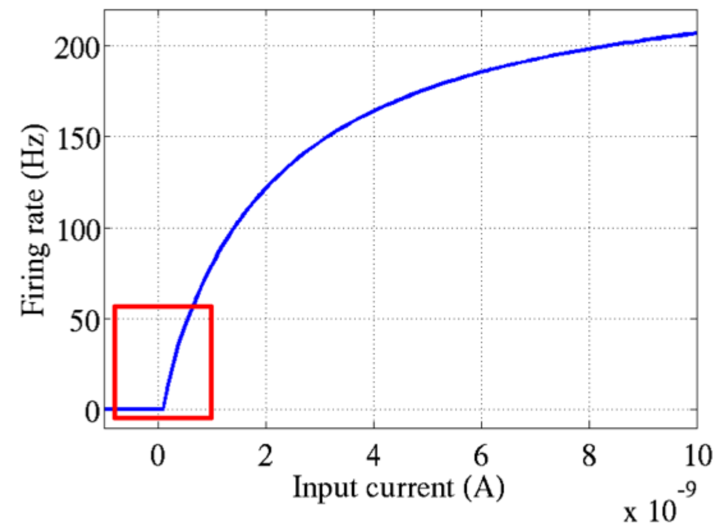
- Result:
  Helps initializing the weights to useful sizes instead of using pre-training on single layers

Understanding the difficulty of training deep feedforward neural networks; Glorot and Bengio 2010
International conference on artificial intelligence and statistics

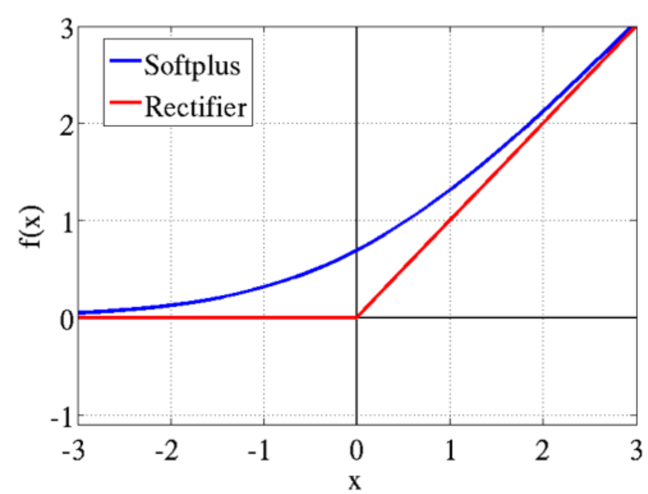
Linear Rectifier Units

- Different (biologically inspired) activation function





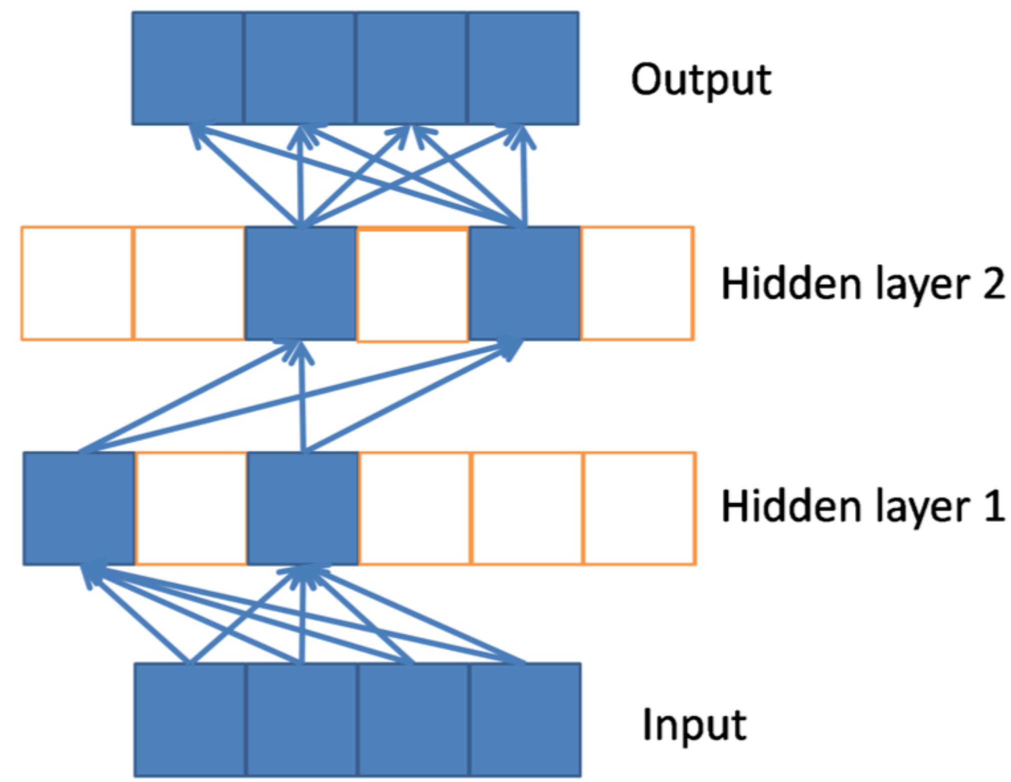
- Use $max(0, x)$ as activation function



Deep Sparse Rectifier Neural Networks; Glorot, Bordes, Bengio 2011
International conference on artificial intelligence and statistics

Linear Rectifier Units

- Only subset of neurons are active
- Computation linear on subset
- Linear within a small input region
- Cheaper computations
- Sparse activation (real zeros in neuron outputs)
- Gradients do not vanish due to activation non-linearities
- Yields similar results with and without pre-training



Deep Sparse Rectifier Neural Networks; Glorot, Bordes, Bengio 2011
International conference on artificial intelligence and statistics

Direct Supervised Training

- When using Xavier initialization, pre-training is replaced

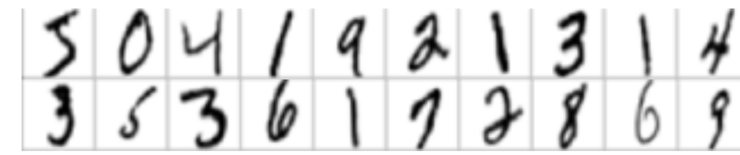
- When using Linear Rectifier Units, pre-training does not give much advantage

Overview

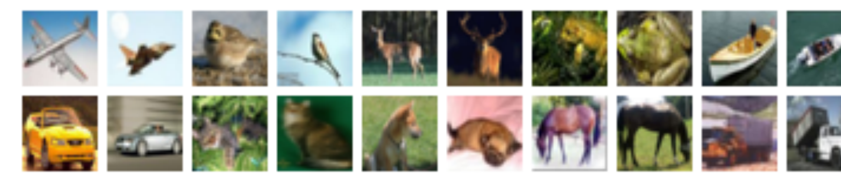
- Introduction
- Basic Components
- Topologies
- Recent Ideas
- Applications
- Summary

Computer Vision

- Handwriting recognition (e.g. MNIST dataset)

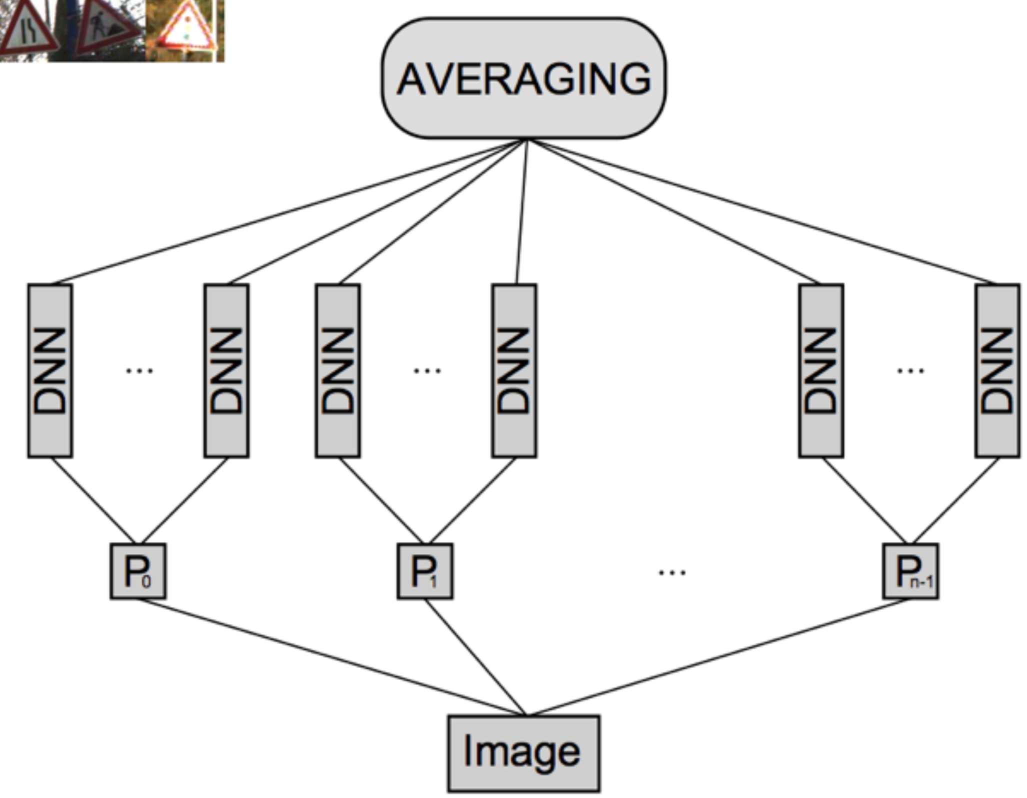


- Image classification (e.g. CIFAR 10 dataset)



- Traffic sign recognition

Dataset	Best result of others [%]	MCDNN [%]	Relative improv. [%]
MNIST	0.39	0.23	41
NIST SD 19	see Table 4	see Table 4	30-80
HWDB1.0 on.	7.61	5.61	26
HWDB1.0 off.	10.01	6.5	35
CIFAR10	18.50	11.21	39
traffic signs	1.69	0.54	72
NORB	5.00	2.70	46



Multi-column Deep Neural Networks for Image Classification; Ciresan, Meier and Schmidhuber 2012
2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Computer Vision

- Pedestrian detection

Pedestrian Detection aided by Deep Learning Semantic Tasks; Tian et al. 2014
arXiv preprint arXiv:1412.0069

- Object recognition, detecting robotic grasps

Deep Learning for Detecting Robotic Grasps; Lenz, Lee and Saxena 2015
The International Journal of Robotics Research

Speech Recognition

- Google
  - Current neural networks have more than 30 layers
  - 8% word error rate compared to 23% in 2013
- Baidu: Deep Speech (Large recurrent neural network)

System	Clean (94)	Noisy (82)	Combined (176)
Apple Dictation	14.24	43.76	26.73
Bing Speech	11.73	36.12	22.05
Google API	6.64	30.47	16.72
wit.ai	7.94	35.06	19.41
Deep Speech	**6.56**	**19.06**	**11.85**

DeepSpeech: Scaling up end-to-end speech recognition; Hannun et al. 2014
arXiv preprint arXiv:1412.5567

Overview

- Introduction
- Basic Components
- Topologies
- Recent Ideas
- Applications
- Summary

Summary

- Basic Components: Restricted Boltzmann Machines
  Auto-Encoders
- Topologies: Stacked RBMs/Auto-Encoders
  Convolutional Neural Networks
- Recent Ideas: Drop Out
  Xavier Initialization
  Linear Rectifier Units
- Applications: Computer Vision
  Speech Recognition

References

- Tutorials:
  http://deeplearning.net/tutorial/index.html (with code samples)
  http://deeplearning.stanford.edu/wiki/index.php/UFLDL_Tutorial

- Fischer, A., & Igel, C. (2012). An introduction to restricted Boltzmann machines. *In Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications* (pp. 14-36). Springer Berlin Heidelberg.

- Hinton, G. E. (2012). A practical guide to training restricted boltzmann machines. *In Neural Networks: Tricks of the Trade* (pp. 599-619). Springer Berlin Heidelberg.

- Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In Neural Networks: Tricks of the Trade (pp. 437-478). Springer Berlin Heidelberg.

- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting*. The Journal of Machine Learning Research, 15(1*), 1929-1958.

- Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics* (pp. 249-256).

- Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics* (pp. 315-323).

- Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 35(8)*, 1798-1828.

- Arel, I., Rose, D. C., & Karnowski, T. P. (2010). Deep machine learning-a new frontier in artificial intelligence research [research frontier]. *Computational Intelligence Magazine, IEEE, 5(4)*, 13-18.

- Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and trends® in Machine Learning, 2(1)*, 1-127.

- Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science, 313(5786)*, 504-507.

References

- Convolutional-Recursive Deep Learning for 3D Object Classi?cation (Socher 2012)
- ImageNet Classification with Deep Convolutional Neural Networks, Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton, NIPS 2012.
- “Improving neural networks by preventing co-adaptation of feature detectors.” Hinton, Geoffrey E., et al. arXiv preprint arXiv:1207.0580 (2012).
- Montúfar, Guido, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. “On the Number of Linear Regions of Deep Neural Networks.” arXiv preprint arXiv:1402.1869 (2014).
- Dauphin, Yann N., Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. “Identifying and attacking the saddle point problem in high-dimensional non-convex optimization.” In Advances in Neural Information Processing Systems, pp. 2933-2941. 2014.
- Min Lin, Qiang Chen, Shuicheng Yan (2013). Network In Network. arXiv:1312.4400
- Bengio, Yoshua, et al. “Greedy layer-wise training of deep networks.” Advances in neural information processing systems 19 (2007): 153.
- Ranzato(2007). Efficient learning of sparse representations with an energy-based model. In NIPSí06.
- Rumelhart(1986). Learning representations by backpropagating errors. Nature, 323, 533ñ536.
- Bengio, Yoshua, and Olivier Delalleau. “On the expressive power of deep architectures.” Algorithmic Learning Theory. Springer Berlin/ Heidelberg, 2011.
- Going Deeper with Convolutions, Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, 19-Sept-2014.

Summary

- Basic Components:
  - Restricted Boltzmann Machines
  - Auto-Encoders
- Topologies:
  - Stacked RBMs/Auto-Encoders
  - Convolutional Neural Networks
- Recent Ideas:
  - Drop Out
  - Xavier Initialization
  - Linear Rectifier Units
- Applications:
  - Computer Vision
  - Speech Recognition