

Deep Learning

Sargur Srihari

Topics

- Multilayer Neural Networks
- Deep Convolutional Nets
- Recurrent Nets
- Deep Belief Networks

Deep Learning Summary (LeCun,Bengio,Hinton, Nature 2015)

1. Computational models composed of multiple processing layers
 - To learn representations of data with multiple levels of abstraction
2. Dramatically improved state-of-the-art in:
 - Speech recognition, Visual object recognition, Object detection
 - Other domains: Drug discovery, Genomics
3. Discovers intricate structure in large data sets
 - Using backpropagation to change parameters
 - Compute representation in each layer from previous layer
4. Deep convolutional nets: image proc, video, speech
5. Recurrent nets: sequential data, e,g., text, speech

ML Technology

- ML powers many aspects of modern society
 - Web searches
 - Content filtering on social networks
 - Recommendations on e-commerce websites
 - Consumer products:
 - cameras, smartphones
- ML used to:
 - Identify objects in images
 - Transcribe speech to text
 - Match news items, posts or products with user interests
 - Select relevant results of search
- Increasingly these use deep learning techniques

Limitations of Conventional ML

- Limited in ability to process natural data in raw form
- PR and ML systems require
 - careful engineering and domain expertise to transform raw data, e.g., pixel values, into a feature vector for a classifier

Representation learning

- Methods that allow a machine to be fed with raw data to automatically discover representations needed for detection or classification
- Deep Learning methods are Representation Learning Methods
- Use multiple levels of representation
 - Composing simple but non-linear modules that transform representation at one level (starting with raw input) into a representation at a higher slightly more abstract level
 - Complex functions can be learned
 - Higher layers of representation amplify aspects of input important for discrimination and suppress irrelevant variations

Image Example

- Input is an array of pixel values
 - First stage is presence or absence of edges at particular locations and orientations of image
 - Second layer detects motifs by spotting particular arrangements of edges, regardless of small variations in edge positions
 - Third layer assembles motifs into larger combinations that corresponds to parts of familiar objects
 - Subsequent layers would detect objects as combinations of these parts
- Key aspect of deep learning:
 - These layers of features are not designed by human engineers
 - They are learned from data using a general purpose learning procedure

Applications of Deep Learning

- Major advances in solving problems that have resisted best attempts of AI for many years
- Good at discovering structures in high-dimensional data
 - Many domains of science, government and business
- Beaten other ML techniques at:
 - Predicting activity of potential drug molecules
 - Analysing particle accelerator data
 - Reconstructing brain circuits
 - Predicting effects of mutations in non-coding DNA on gene expression and disease
- Produced promising results for NLP
 - Topic categorization, sentiment analysis, QA, MT

Supervised Learning

- Most common form of ML, deep or not
 - Ex: classify images containing a house, a car, a person or a pet
 - Collect large data set of images of houses, cars, people and pets, each labelled with its category
 - During training, machine is shown image and it produces an output in the form of a vector of scores, one for each category
 - We want desired category to have highest score– which is unlikely before training
- Compute *objective function* that measures error between output scores and desired scores
- Modify internal parameters to reduce error
- In deep learning hundreds of millions of adjustable weights and hundreds of millions of labelled examples

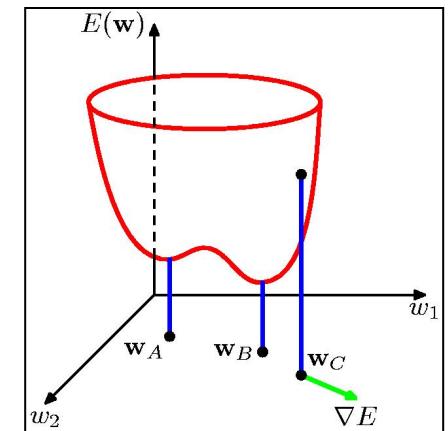
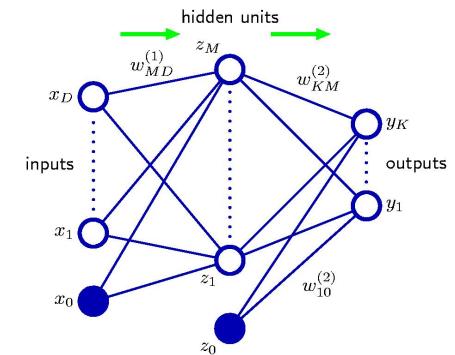
Gradient Descent

- Objective function, averaged over all training examples is a hilly landscape in high-dimensional space of weight values
- Negative gradient vector indicates direction of steepest descent taking it closer to a minimum

Minimizing Error Function

- Goal is to learn the weights w from a labelled set of training samples
- Learning procedure has two stages
 1. Evaluate derivatives of error function $\nabla E(w)$ with respect to weights $w_1,..w_T$
 2. Use derivatives to compute adjustments to weights

$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E(w^{(\tau)})$$



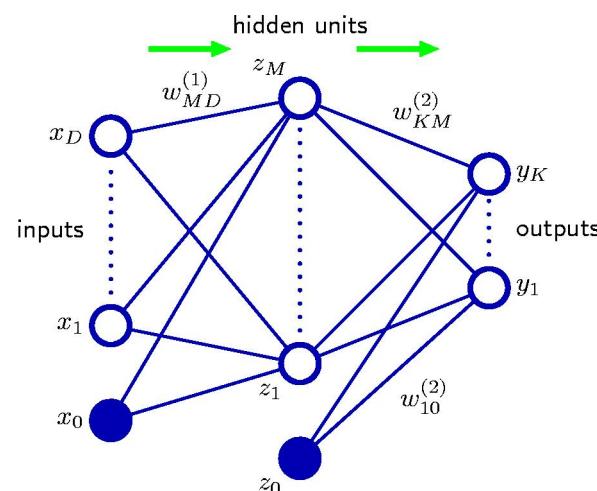
Using derivatives to update weights

- Gradient descent
 - Update the weights using $w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E(w^{(\tau)})$

- Where the gradient vector $\nabla E(w^{(\tau)})$ consists of the vector of derivatives evaluated using back-propagation

$$\nabla E(w) = \frac{d}{dw} E(w) = \begin{bmatrix} \frac{\partial E}{\partial w_{11}^{(1)}} \\ \vdots \\ \frac{\partial E}{\partial w_{MD}^{(1)}} \\ \frac{\partial E}{\partial w_{11}^{(2)}} \\ \vdots \\ \frac{\partial E}{\partial w_{KM}^{(2)}} \end{bmatrix}$$

There are $W = M(D+1) + K(M+1)$ elements in the vector
 Gradient $\nabla E(w^{(\tau)})$ is a $W \times 1$ vector



Stochastic Gradient Descent (SGD)

Most practitioners use SGD for DL

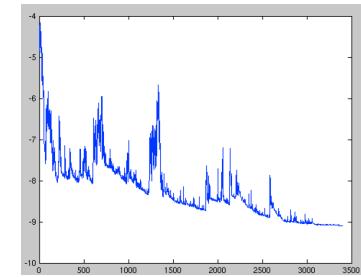
Consists of showing the input vector for a few examples, computing the outputs and the errors, Computing the *average gradient* for those examples, and adjusting the weights accordingly.

Process repeated for many small sets of examples from the training set until the average of the objective function stops decreasing.

Called stochastic because each small set of examples gives a noisy estimate of the average gradient over all examples.

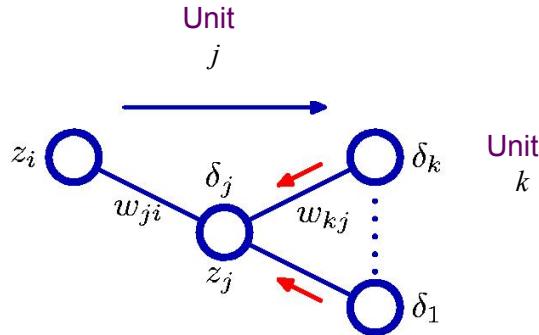
Usually finds a good set of weights quickly compared to elaborate optimization techniques.

After training, performance is measured on a different test set:
tests generalization ability of the machine
— its ability to produce sensible answers on new inputs never seen during training.



Fluctuations in objective as gradient steps are taken in mini batches

Error Backpropagation Algorithm



- Backpropagation Formula

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k$$

- Value of δ for a particular hidden unit can be obtained by propagating the δ 's backward from units higher-up in the network

1. Apply input vector x_n to network and forward propagate through network using

$$a_j = \sum_i w_{ji} z_i \quad \text{and} \quad z_j = h(a_j)$$

2. Evaluate δ_k for all output units using $\delta_k = y_k - t_k$

3. Backpropagate the δ 's using

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k$$

to obtain δ_j for each hidden unit

4. Use $\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i$ to evaluate required derivatives

A Simple Example

- Two-layer network
- Sum-of-squared error
- Output units: *linear activation* functions, i.e., multiple regression

$$y_k = a_k$$

- Hidden units have *logistic sigmoid* activation function

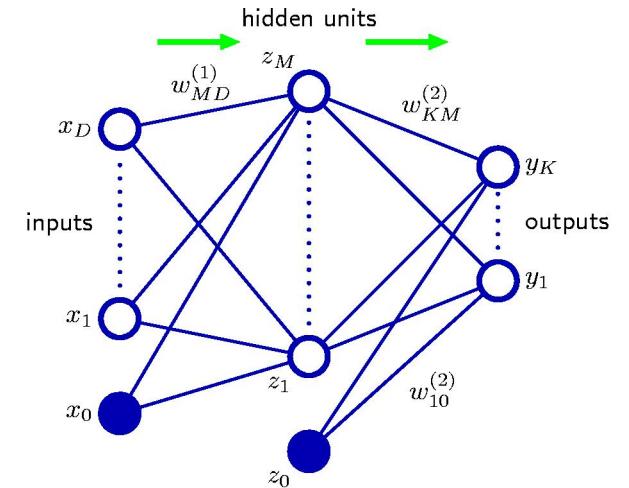
$$h(a) = \tanh(a)$$

where

$$\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$$

simple form for derivative

$$h'(a) = 1 - h(a)^2$$



Standard Sum of Squared Error

$$E_n = \frac{1}{2} \sum_{k=1}^K (y_k - t_k)^2$$

y_k : activation of output unit k
 t_k : corresponding target
 for input x_k

Simple Example: Forward and Backward Prop

For each input in training set:

- Forward Propagation

$$\left\{ \begin{array}{l} a_j = \sum_{i=0}^D w_{ji}^{(1)} x_i \\ z_j = \tanh(a_j) \\ y_k = \sum_{j=0}^M w_{kj}^{(2)} z_j \end{array} \right.$$

- Output differences

$$\delta_k = y_k - t_k$$

- Backward Propagation (δ s for hidden units)

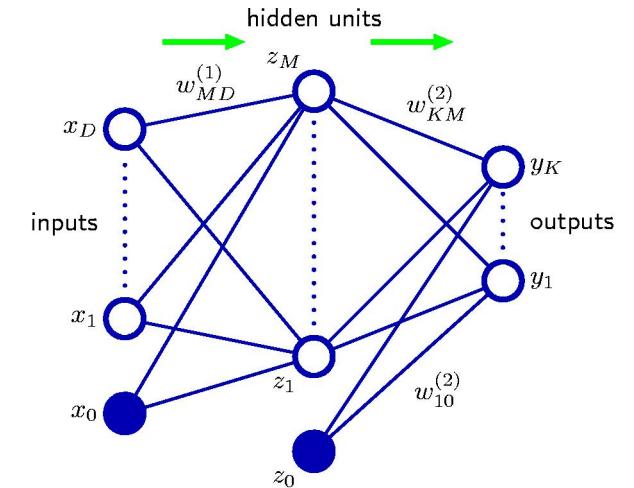
$$\delta_j = (1 - z_j^2) \sum_{k=1}^K w_{kj} \delta_k$$

- Derivatives wrt first layer and second layer weights

$$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \delta_j x_i \quad \frac{\partial E_n}{\partial w_{kj}^{(2)}} = \delta_k z_j$$

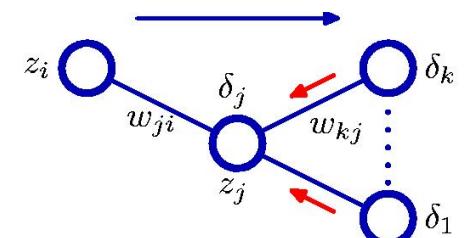
- Batch method

$$\frac{\partial E}{\partial w_{ji}} = \sum_n \frac{\partial E_n}{\partial w_{ji}}$$



$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k$$

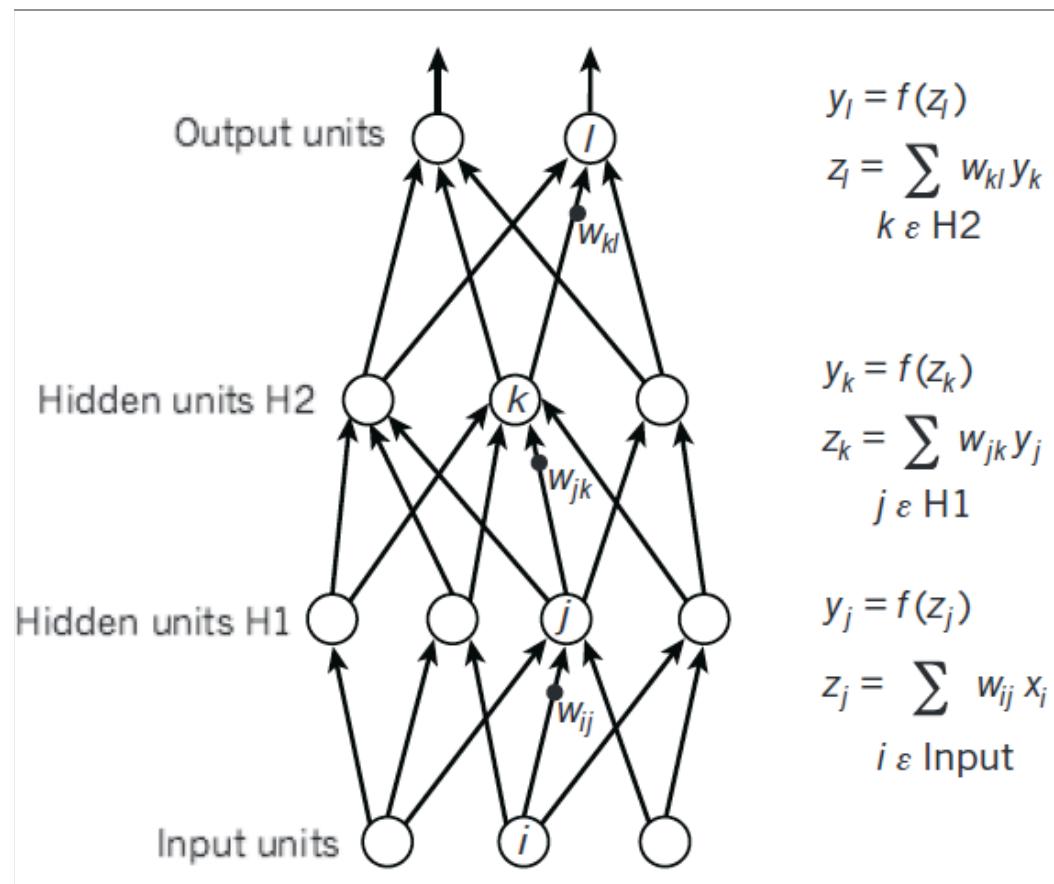
$$h'(a) = 1 - h(a)^2$$



Forward Pass in Neural Net

2 Hidden Layers, 1 Output Layer

Each layer is a module through which one can back-propagate gradients.

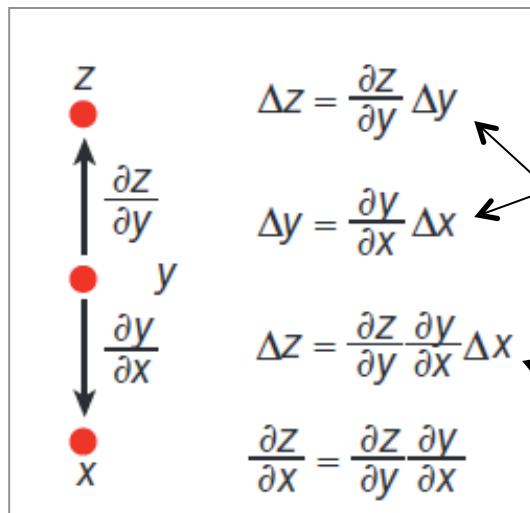


At each layer, we first compute the total input z to each unit, which is a weighted sum of the outputs of the units in the layer below.

Then a non-linear function $f(\cdot)$ is applied to z to get the output of the unit.

Chain Rule of Derivatives

Tells us how two small effects (that of a small change of x on y , and that of y on z) are composed.



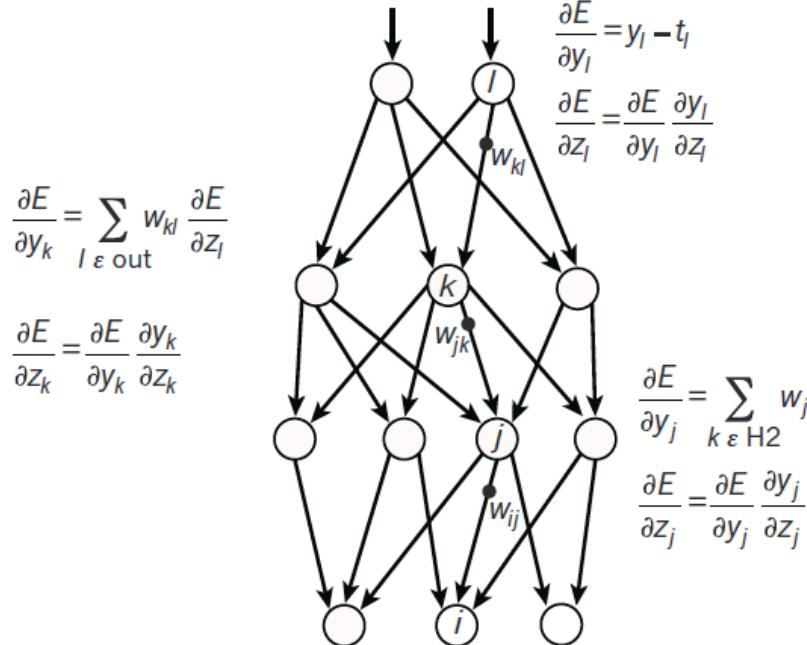
A small change Δx in x gets transformed first into a small change Δy in y by getting multiplied by $\partial y/\partial x$ (that is, the definition of partial derivative). Similarly, the change Δy creates a change Δz in z .

Substituting one equation into the other gives the chain rule of derivatives — how Δx gets turned into Δz through multiplication by the product of $\partial y/\partial x$ and $\partial z/\partial x$. It also works when x , y and z are vectors (and the derivatives are Jacobian matrices).

Equations for Computing the Backward Pass

d

Compare outputs with correct answer to get error derivatives



Two hidden layers

Layers are labelled i, j, k, l

At each hidden layer we compute the error derivative wrt the output of each unit, which is a weighted sum of the error derivatives wrt the total inputs to the units in the layer above.

Convert error derivative wrt the output into the error derivative wrt the input by multiplying it by the gradient of $f(z)$.

At the output layer, the error derivative wrt the output of a unit is computed by differentiating the cost function.

This gives $y_l - t_l$ if the cost function for unit l is $0.5(y_l - t_l)^2$, where t_l is the target value.

Once the $\partial E / \partial z_k$ is known, the error-derivative for the weight w_{jk} on the connection from unit j in the layer below is just $y_j \partial E / \partial z_k$.

Choice of non-linear function

- Most popular today is Rectified Linear Unit (ReLU)
 - a half-wave rectifier
$$f(z) = \max(z, 0)$$
- In past decades, neural nets used smoother non-linearities
 - $\tanh(z)$ or $1/(1+\exp(-z))$
- ReLU learns faster in networks with many layers
- Allowing training of deep supervised network without unsupervised pre-training

Nonlinear Functions used in Neural Nets

- Rectified linear unit (ReLU)

$$f(z) = \max(0, z)$$

commonly used in recent years,

- More conventional sigmoids:

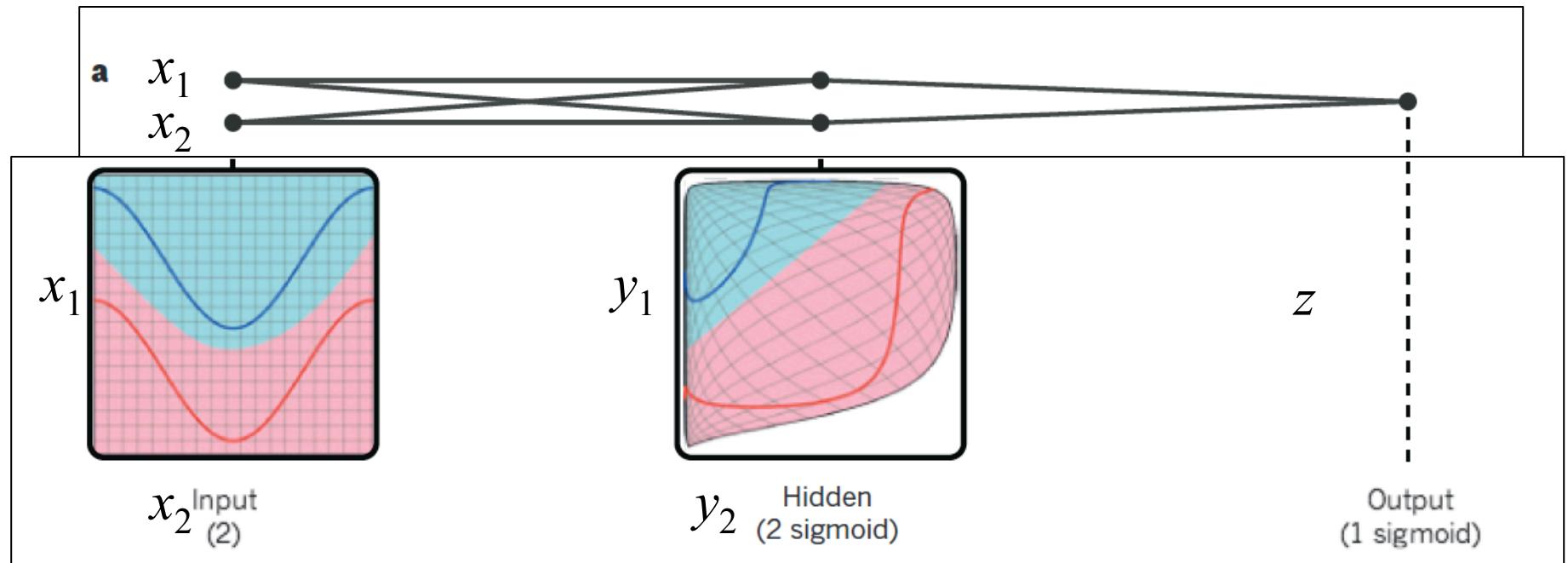
- hyperbolic tangent,

$$f(z) = (\exp(z) - \exp(-z)) / (\exp(z) + \exp(-z)) \text{ and}$$

- logistic function,

$$f(z) = 1 / (1 + \exp(-z))$$

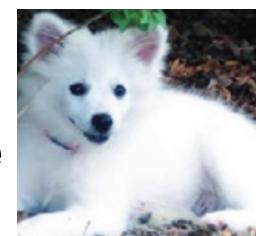
Neural Network with 2 Inputs, 2 HiddenUnits, 1 Output unit



- Input Data (red and blue curves) are not linearly separable
- Network makes them linearly separable
 - By distorting the input space
 - Note that grid is also distorted

Deep versus Shallow Classifiers

- Linear classifiers can only carve the input space into very simple regions
- Image and speech recognition require input-output function to be insensitive to irrelevant variations of the input,
 - e.g., position, orientation and illumination of an object
 - Variations in pitch or accent of speech
 - While being sensitive to minute variations, e.g., white wolf and breed of wolf-like white dog called Samoyed
 - At pixel level two Samoyeds in different positions may be very different, whereas a Samoyed and a wolf in the same position and background may be very similar

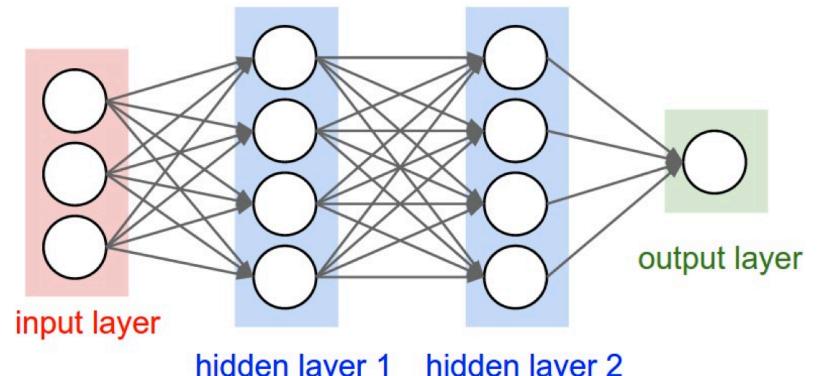


Selectivity-Invariance dilemma

- Shallow classifiers need a good feature extractor
- One that produces representations that are:
 - selective to aspects of image important for discrimination
 - but invariant to irrelevant aspects such as pose of the animal
- Generic features (e.g., Gaussian kernel) do not generalize well far from training examples
- Hand-designing good feature extractors requires engineering skill and domain expertise
- Deep learning learns features automatically

Deep Learning Architecture

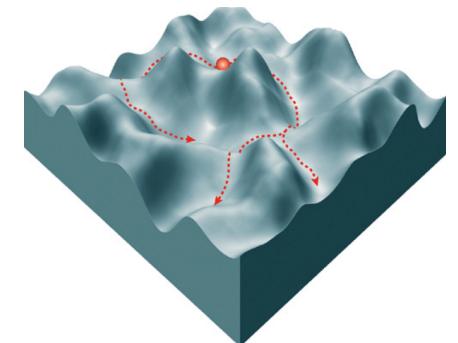
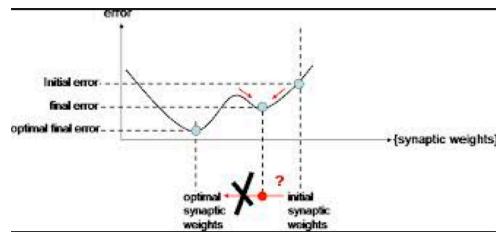
- Multilayer stack of simple modules
- All modules (or most) subject to :
 - Learning
 - Non-linear input-output mappings
- Each module transforms input to improve both selectivity and invariance of the representation
- With depth of 5 to 20 layers can implement extremely intricate functions of input
 - Sensitive to minute details
 - Distinguish Samoyeds from white wolves
 - Insensitive to irrelevant variations
 - Background, pose, lighting, surrounding objects



Local minima is rarely a problem with NNs

- It was thought that simple gradient descent would get trapped in local minima

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E (\mathbf{w}^{(\tau)})$$



- Rarely a problem with large networks
 - Regardless of initial conditions, system always reaches solutions of similar quality
 - Landscape is packed with a combinatorially large number of saddle points where gradient is zero
 - Almost all have similar values of objective function

Pre-training

- Unsupervised learning to create layers of feature detectors
 - No need for labelled data
- Objective of learning in each layer of feature detectors:
 - Ability to reconstruct or model activities of feature detectors (or raw inputs) in layer below
 - By pre-training weights of a deep network could be initialized to sensible values
- Final layer of output units added at top of network and whole deep system fine-tuned using back-propagation
- Worked well in handwritten digit recognition
 - When data was limited

Pre-training in Speech Recognition

- First major application was in speech recognition
- Made possible by advent of fast GPUs
 - Allowed networks to be trained 10 or 20 times faster
- Record-breaking results on speech reco benchmark

From pre-training to ConvNets

- It turned out that pre-training stage was only needed for small data sets
- Convolutional neural networks
 - Type of deep feedforward network
 - Much easier to train and generalized much better than networks with full connectivity between adjacent layers

Convolutional Neural Networks

- Designed to process data that come in the form of multiple arrays
 - E.g., a color image composed of three 2D arrays of pixel intensities in three color channels
- Many data modalities are in the form of multiple arrays:
 - 1D for signals and sequences, including language
 - 2D for images and audio spectrograms
 - 3D for video or volumetric images

Four key ideas behind ConvNets

- Take advantage of properties of natural signals
 1. Local connections
 2. Shared weights
 3. Pooling
 4. Use of many layers

Limitations of Neural Networks

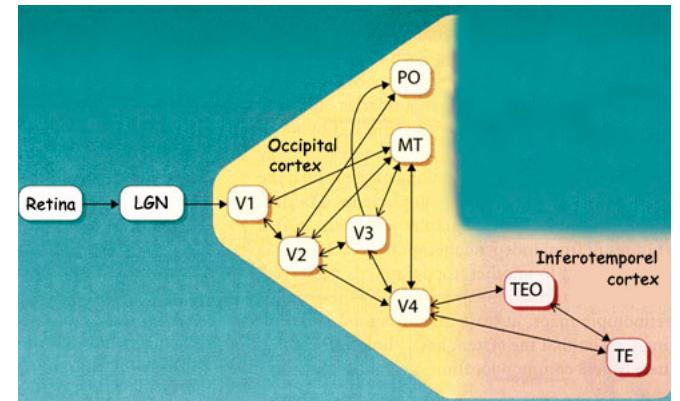
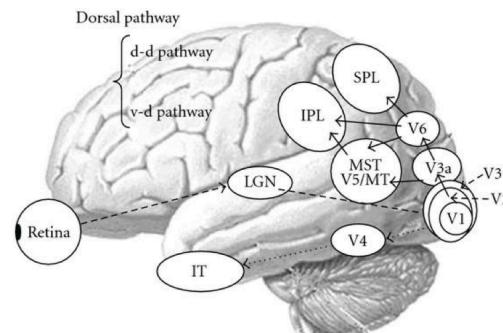
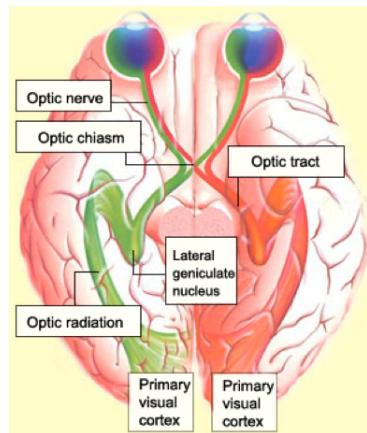
- Need substantial number of training samples
- Slow learning (convergence times)
- Inadequate parameter selection techniques that lead to poor minima

Solution: Exploitation of Local Properties

- Network should exhibit invariance to translation, scaling and elastic deformations
 - A large training set can take care of this
- It ignores a key property of images
 - Nearby pixels are more strongly correlated than distant ones
 - Modern computer vision approaches exploit this property
- Information can be merged at later stages to get higher order features and about whole image

ConvNet Inspired by Visual Neuroscience

- Classic notions of simple cells and complex cells
- Architecture similar to LGN-V1-V2-V4-IT hierarchy in visual cortex ventral pathway
 - LGN: lateral geniculate nucleus receives input from retina
 - 30 different areas of visual cortex: V1 and V2 are principal
 - Infero-Temporal cortex performs object recognition

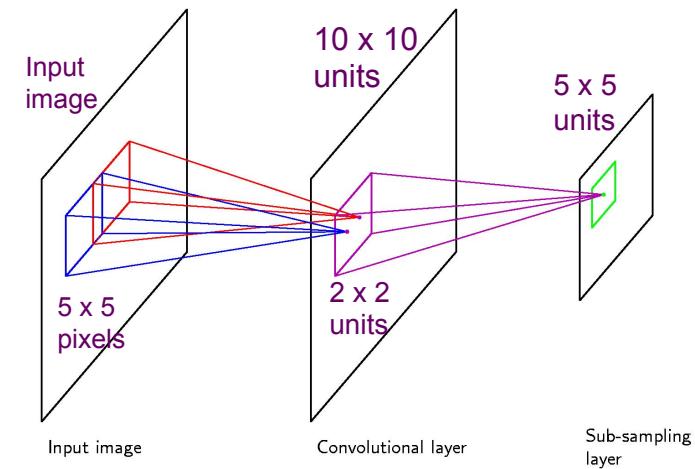


Three Mechanisms of Convolutional Neural Networks

1. Local Receptive Fields
2. Subsampling
3. Weight Sharing

Convolution and Sub-sampling

- Instead of treating input to a fully connected network
- Two layers of Neural networks are used
 1. Layer of convolutional units
 - which consider overlapping regions
 2. Layer of subsampling units
- Several feature maps and sub-sampling
 - Gradual reduction of spatial resolution compensated by increasing no. of features
- Final layer has softmax output
- Whole network trained using backpropagation



Each pixel patch is 5×5

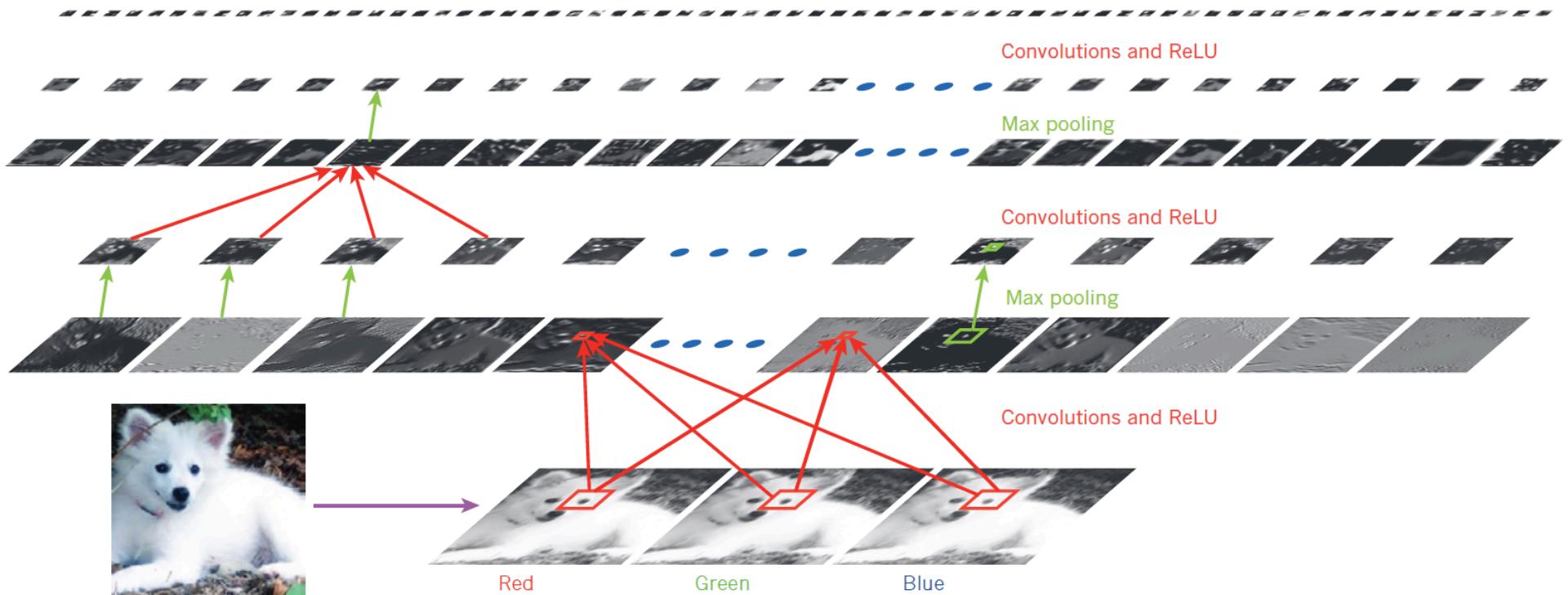
This plane has $10 \times 10 = 100$ neural network units (called a feature map). Weights are same for different planes. So only 25 weights are needed. Due to weight sharing this is equivalent to convolution. Different features have different feature maps

Soft weight sharing

- Reducing the complexity of a network
- Encouraging groups of weights to have similar values
- Only applicable when form of the network can be specified in advance
- Division of weights into groups, mean weight value for each group and spread of values are determined during the learning process

ConvNet for Samoyed...Siberian-Husky

Samoyed (16); Papillon (5.7); Pomeranian (2.7); Arctic fox (1.0); Eskimo dog (0.6); white wolf (0.4); Siberian husky (0.4)



Outputs (not filters) of each layer (horizontally).

Each rectangular image is a feature map corresponding to output for one of the learned features, detected at each of the image positions.

Lower-level features act as oriented edge detectors

Architecture of a typical ConvNet

- Structured as a series of stages
- First few stages are composed of two types of layers and a non-linearity:
 1. Convolutional layer
 - To detect local conjunctions of features from previous layer
 2. Non-linearity
 - ReLU
 3. Pooling layer
 - To merge semantically similar features into one

A convolutional layer unit

- Organized in feature maps
- Each unit connected to local patches in feature maps of previous layer through weights (called a filter bank)
- Result is passed through a ReLU

A pooling layer unit

- Computes maximum of a local patch of units in one feature map
- Neighboring pooling units take input from patches that are shifted by more than one row or column
 - Thereby reducing the dimension of the representation
 - Creating invariance to small shifts and distortions

Backpropagation of Gradients through ConvNet

- As simple as through a regular deep network
- Allow all weights in all filter banks to be trained

ConvNet History

- Neocognitron
 - Similar architecture
 - Did not have end-to-end supervised learning using Backprop
- ConvNet with probabilistic model was used for OCR and handwritten check reading

Applications of ConvNets

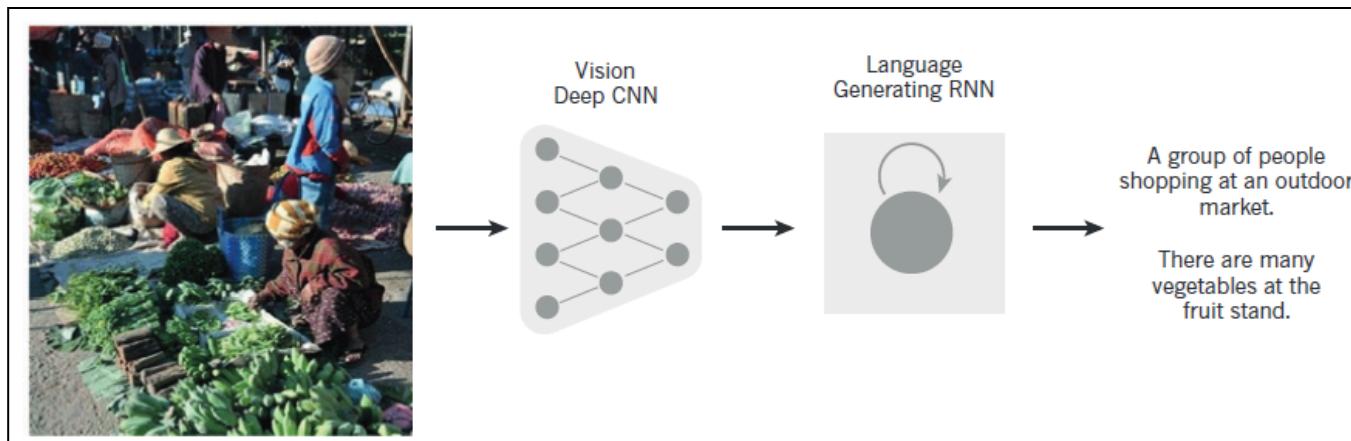
- Applied with great success to images: detection, segmentation and recognition of objects and regions
 - Tasks where abundant data was available
 - Traffic sign recognition
 - Segmentation of biological images
 - Connectomics
 - Detection of faces, text, pedestrians, human bodies in natural images
 - Major recent practical success is face recognition
 - Images can be labeled at pixel level
 - Applications in autonomous mobile robots, and self-driving cars
- Other applications gaining importance
 - Natural language understanding
 - Speech recognition

Success of ConvNet in ImageNet Competition 2012

- Data set of 1 million images from the web
- Contained 1,000 different classes
- Error rate
 - Halved the error rate of best competing computer vision approaches
- Success came from:
 - Efficient use of GPUs
 - ReLUs
 - New regularization technique called *dropout*
 - Techniques to generate new training samples by deforming existing ones

Another stunning success: From images to text

- Combining ConvNets and Recurrent Net Modules
- Caption generated by a recurrent neural network (RNN) taking as input:
 1. Representation generated by a deep CNN
 2. RNN trained to translate high-level representations of images into captions



Better translation of images to captions

- Different focus (lighter patches given more attention)



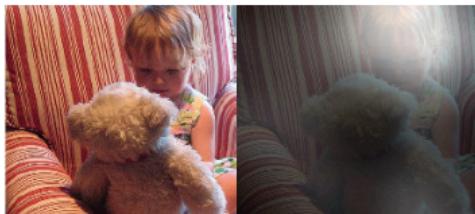
A woman is throwing a **frisbee** in a park.



A **dog** is standing on a hardwood floor.



A **stop** sign is on a road with a mountain in the background



A little girl sitting on a bed with a **teddy bear**.



A group of **people** sitting on a boat in the water.



A **giraffe** standing in a forest with **trees** in the background.

- As it generates each word (**bold**), it exploits it to achieve better translation of images to captions

Recent ConvNet architectures

1. 10 to 20 layers of ReLUs
2. Hundreds of millions of weights
3. Billions of connections between units
4. Training time
 - Would have taken weeks a couple of years ago
 - Advances in hardware, software and parallelization reduces it to a few hours
 - ConvNets are easily amenable to efficient hardware implementations
 - NVIDIA, Mobileye, Intel, Qualcomm and Samsung are developing ConvNet chips for smartphones, cameras, robots and self-driving cars

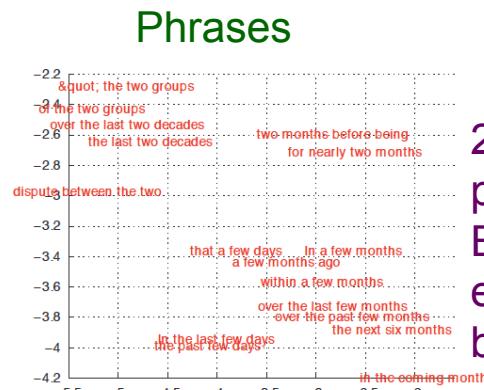
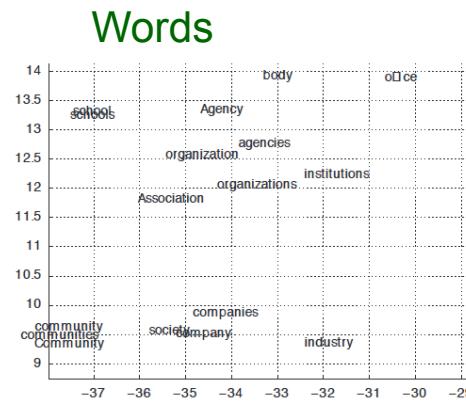
Distributed Representations

- Deep nets have two different exponential advantages over classic learning algorithms. Both advantages arise from
 - power of composition
 - Depend on underlying data-generating distribution having an appropriate compositional structure
1. Learning distributed representations enable generalization to new combinations of the values of learned features beyond those seen during training
 1. E.g., 2^n combinations are possible with n binary features
 2. Composing layers of representations brings another advantage: exponential in depth

Language Processing: Prediction

- Predicting the next word from local context of earlier words
 - Each word presented as a 1-of-N vector
 - In the first layer each word creates a different word vector
 - In the language model, other next layers learn to convert input word vector to output word vector for the predicted word

Word representation for modeling language, non-linearly projected to 2-D using t-SNE algorithm. Semantically similar words are mapped nearby.



2-D representation of phrases learnt by English-French encoder-decoder learnt by RNN

Learnt using backpropagation that jointly learns representation for each word and function that predicts a target quantity (next word or sequence of words for translation)

Logic-inspired vs Neural network-inspired paradigms

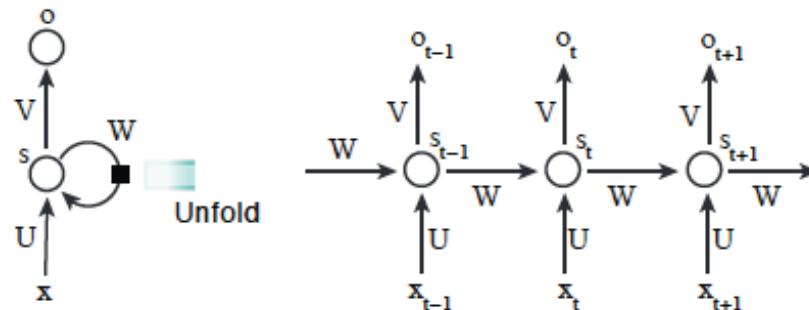
- Logic-inspired paradigm uses
 - Instance of a symbol is something for which the only property is that it is either identical or non-identical to other symbol instances
 - It has no internal structure relevant to its use
 - To reason with symbols they must be bound to variables in judiciously chosen rules of inference
- Neural networks use
 - big activity vectors, big weight matrices and scalar nonlinearities
 - to perform fast intuitive inference that underpins commonsense reasoning

N-gram vs Neural Language Models

- Standard statistical models count frequencies of short symbol sequences of length upto N
- No of possible sequences is V^N , where V is vocabulary size
- So taking context of more than a handful of words would require very large corpora
- N-grams treat each word as an atomic unit, so they cannot generalize across semantically related sequences
- Neural models can because they associate each word with a vector of real-valued features

Recurrent Neural Networks (RNNs)

- Exciting early use of backpropagation was for training RNNs
 - For tasks involving sequential inputs
 - e.g., speech and language, it is better to use RNNs
- RNNs process input sequence one element at a time
 - maintain in their hidden units a state vector that implicitly contains history of past elements in the sequence
 - Same parameters (matrices U , V , W) are used at each time step

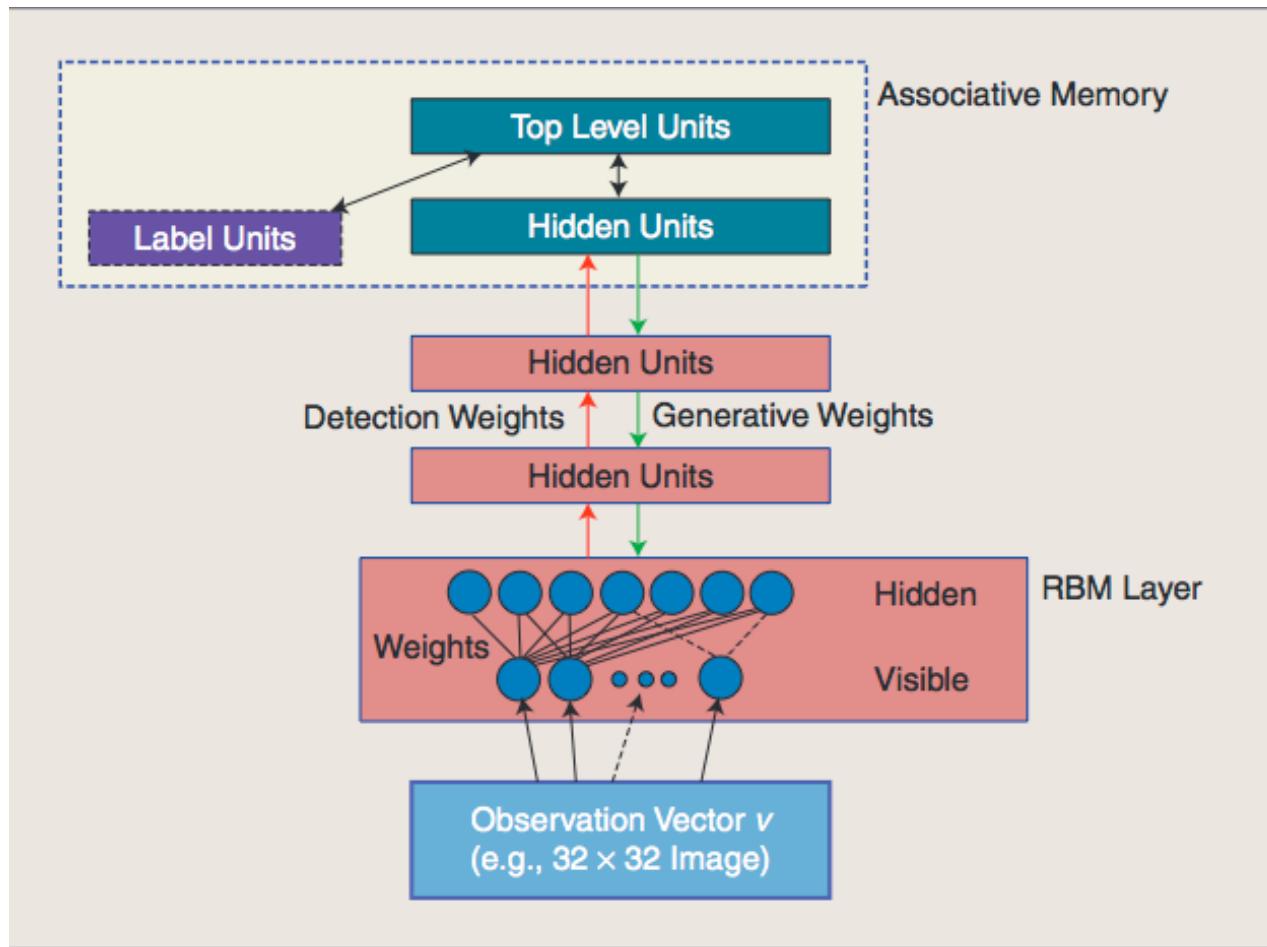


Backpropagation algorithm is applied to the unfolded graph of computational network
 To compute derivative of total error (log-probability of generating right sequence)
 wrt to states s_i and all the parameters

Deep Belief Networks

- DBNs are Generative Models
 - Provide estimates of both $p(x|C_k)$ and $p(C_k|x)$
- Conventional neural networks are discriminative
 - Directly estimate $p(C_k|x)$
- Consist of several layers of Restricted Boltzmann Machines (RBM)
- RBM
 - A form of Markov Random Field

Deep Belief Network Framework



Boltzmann Machine

- Named after Boltzmann Distribution (Or Gibbs Distribution)
 - Gives probability that a system will be in a certain state given its temperature $F(\text{state}) \propto e^{-\frac{E}{kT}}$
 - Where E is energy and k (constant of the distribution) is a product of Boltzmann's constant and thermodynamic temperature
- Energy of Boltzmann network

$$E = - \left(\sum_{i < j} w_{ij} s_i s_j + \sum_i \theta_i s_i \right)$$

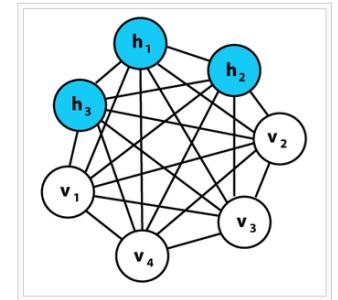
Where:

- w_{ij} is the connection strength between unit j and unit i .
- s_i is the state, $s_i \in \{0, 1\}$, of unit i .
- θ_i is the bias of unit i in the global energy function. ($-\theta_i$ is the activation threshold for the unit.)

The connections in a Boltzmann machine have two restrictions:

- $w_{ii} = 0 \quad \forall i$. (No unit has a connection with itself.)
- $w_{ij} = w_{ji} \quad \forall i, j$. (All connections are symmetric.)

- Training



Restricted Boltzmann Machine

- Generative stochastic ANN

