Deep Learning

Motivation for Deep Networks

- Exploit multiple non-linear transforms
- Distributed representation across a large number of features, similar to brain
- Good features (like edges) are often helpful for multiple tasks
- Learn new things fast using old features
- Visual cortex has depth of 5-10
- Representation is often sparse (1-4% of neurons active at a given time)

(Bengio, 2009)

Motivation

- Quote:

> *when a function can be compactly represented by a deep architecture, it might need a very large architecture to be represented by an insufficiently deep one.*

- Any Boolean function can be written as a sum of products (i.e. two layers or AND/ORs)
- …but the number of gates increases dramatically for this realization

(Bengio, 2009)

Challenges for Training

- Problem:

> Many unreported negative observations as well as the experimental results in [17, 50] suggest that gradient-based training of deep supervised multi-layer neural networks (starting from random initialization) gets stuck in "apparent local minima or plateaus",[1] and that as the architecture gets deeper, it becomes more difficult to obtain good generalization. When starting from random initialization, the solutions obtained with deeper neural networks appear to correspond to poor solutions that perform worse than the solutions obtained for networks with 1 or 2 hidden layers [17, 98]. This happens even though $k + 1$-layer nets can easily represent what a k-layer net can represent (without much added capacity), whereas the converse is not true.

(Bengio, 2009)

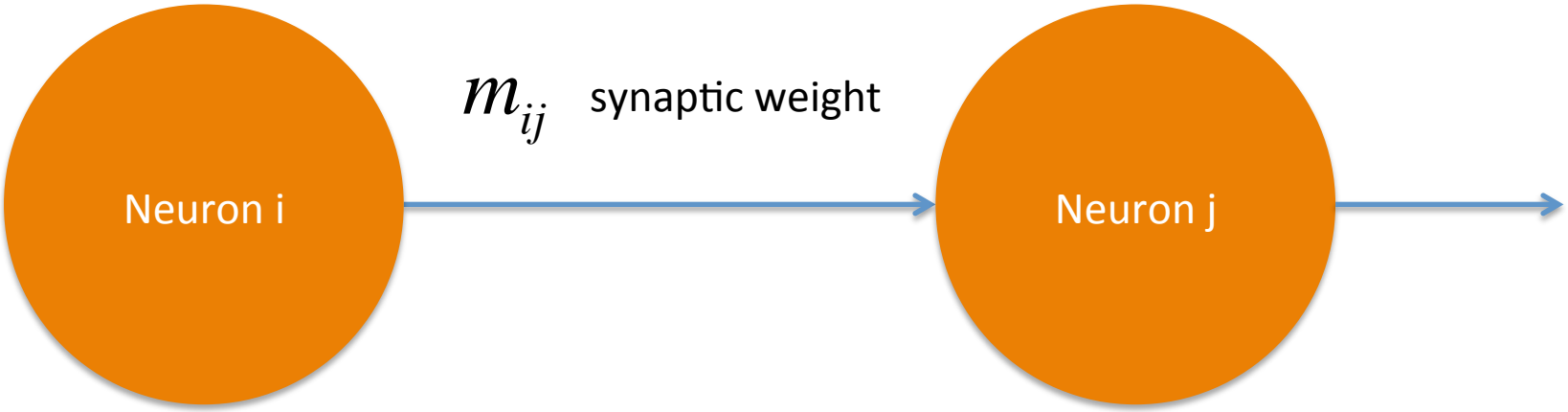
Challenges for Training

- Breakthrough in 2006, Geoff Hinton
- Train layers one by one using approach similar to Hopfield network
- Typically unsupervised, “auto-encoders” for efficient representations of the input
- User backpropagation for final tweaks at the end.
- *“Layer local unsupervised criteria”*

(Bengio, 2009)

Boltzmann Machines

Boltzmann Machine



$$s_i \in \{0,1\}$$

$$z_j = \sum_i m_{ij} s_i + b_i$$

$$prob(s_j = 1) = \frac{1}{1 + e^{-z_i}}$$

Firing rates are now determined by logistic probabilities

http://www.scholarpedia.org/article/Boltzmann_machine

Boltzmann Machine

- Set up like a Hopfield network
- Fully interconnected reciprocal connections

$$m_{ij} = m_{ji} \qquad m_{ii} = 0$$

- Energy function that goes down until stable

$$E = -\sum_{i,j} m_{ij} s_i s_j - \sum_{i,j} s_i b_i$$

http://www.scholarpedia.org/article/Boltzmann_machine

Boltzmann Machine

- Probability of a desired stored state **v** for all neurons is determined by the energy function of that state relative to all possible states **u**

$$P(\vec{v}) = \frac{e^{-E(\vec{v})}}{\sum_{u} e^{-E(\vec{u})}}$$

- Desired patterns should sit on local energy minima

Boltzmann Machine Training

- Gradient based learning

$$\frac{\partial E(\vec{v})}{\partial m_{ij}} = -s_i^v s_j^v$$

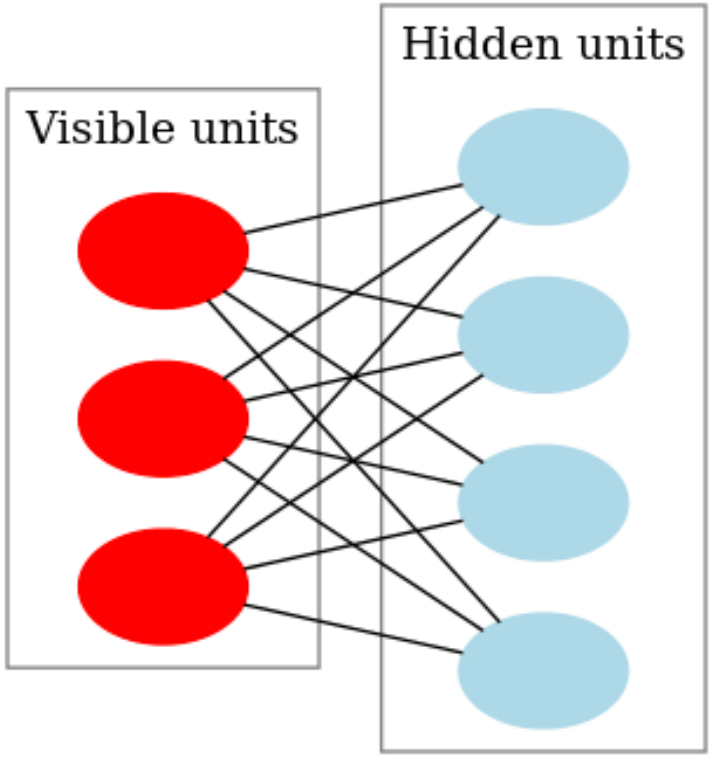
$$\Delta m_{ij} = -\alpha s_i^v s_j^v$$

- Hidden units requires backpropagation, thus generally fails

http://www.scholarpedia.org/article/Boltzmann_machine

Restricted Boltzmann Machine

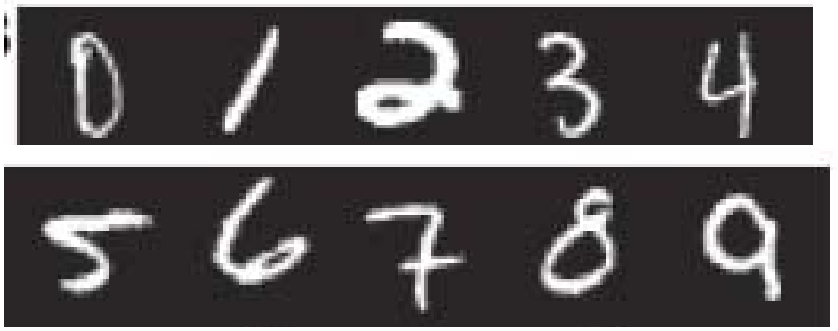
- So don't allow hidden units to talk to each other
- Visible units can only contact hidden units, and hidden units can only contact visible units
- Hidden units are independent given visible activities, and vice versa
- Simplifies credit assignment



http://en.wikipedia.org/wiki/Restricted_Boltzmann_machine

Restricted Boltzmann Machine

- Image representation
- Pixels connect to visible units **v**
- Hidden units **h** are interesting “feature detectors”
- Fewer hidden units -> dimensionality reduction



(Hinton and Salakhutdinov, 2006)

Restricted Boltzmann Machine

- Train network to have low energies corresponding to each digit
- Joint energy function between visible and hidden units

$$E(\vec{v},\vec{h}) = -\sum_{i \in pixels} b_i v_i - \sum_{j \in features} b_j h_j - \sum_{i,j} v_i h_j m_{ij}$$

(Hinton and Salakhutdinov, 2006)

Restricted Boltzmann Machine

- Have training data, and "confabulations" where some of the bits are corrupted

$$\Delta m_{ij} = \alpha\left(\left\langle v_i h_j \right\rangle_{data} - \left\langle v_i h_j \right\rangle_{confab}\right)$$

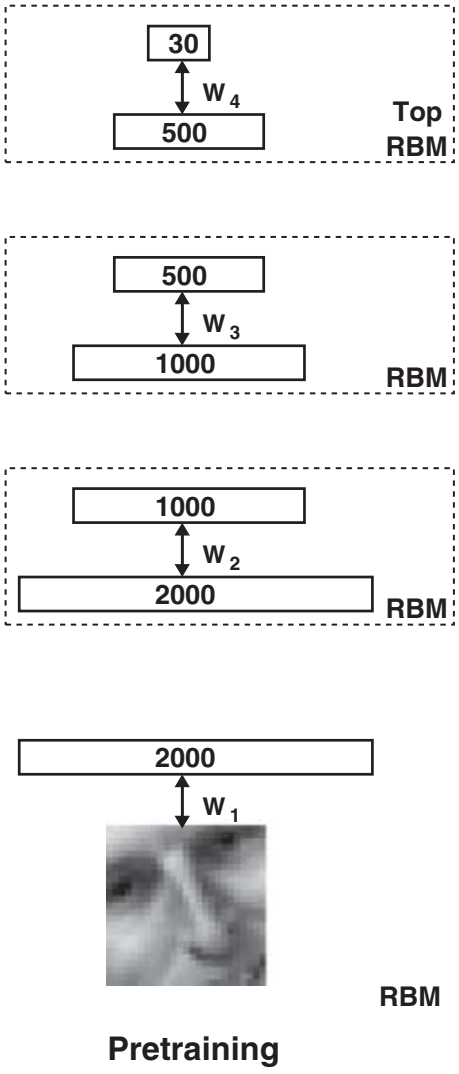
- Lowering energy of desired patterns, raise energy of confabulations
- Result is features that have reduced dimensionality, compare to PCA

(Hinton and Salakhutdinov, 2006)

Deep Networks

Deep Belief Nets

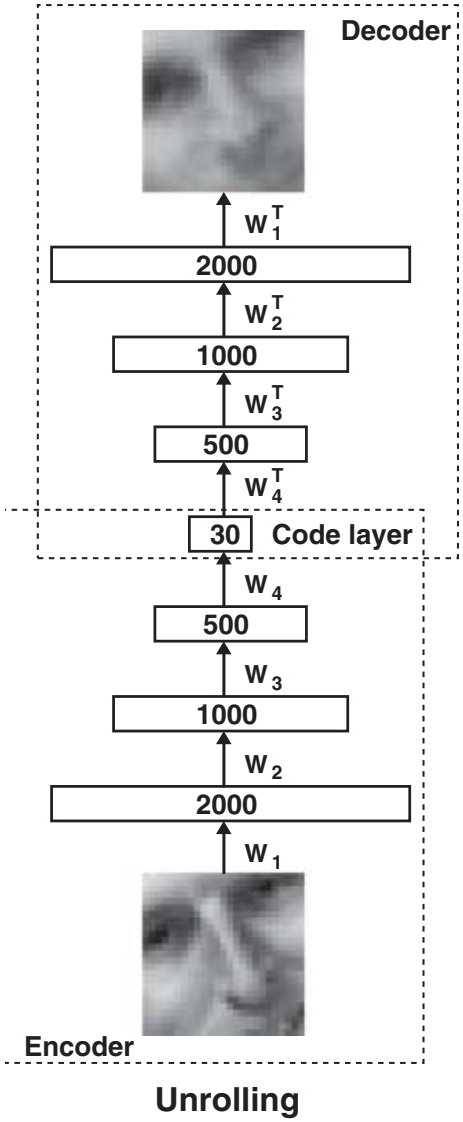
- Step one) Train sequential individual RBMs to find progressively reduced version of your images
- Train weight matrices
- N=2000, 1000, 500, 30
- Better than one layer for later classification



(Hinton and Salakhutdinov, 2006)

Autoencoder

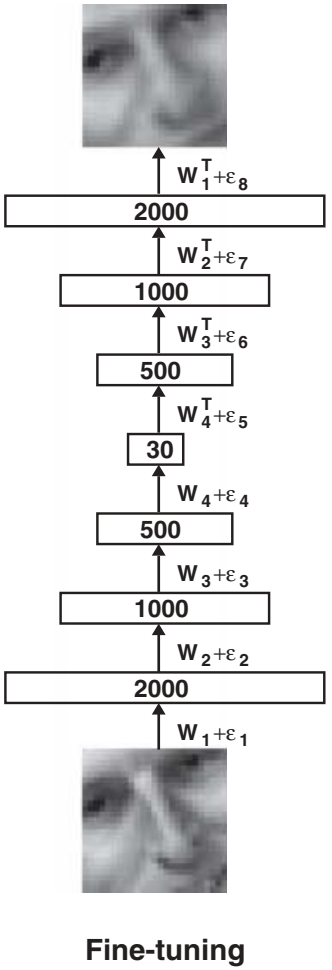
- Step two) Connect them all together
- Step three) Go back out in reverse to reconstruct the image ($w \rightarrow w^T$)
- Makes a network that will generate images from images



(Hinton and Salakhutdinov, 2006)

Autoencoder, optimized

- This network was optimized for representation, not recognition, but the network already does something like the right thing
- Step four) Use back propagation to improve reconstruction from noisy images



Fine-tuning

(Hinton and Salakhutdinov, 2006)

Results

- See how well it can take the data down to 30 numbers, and then reconstruct it

Real Data

Autoencoder

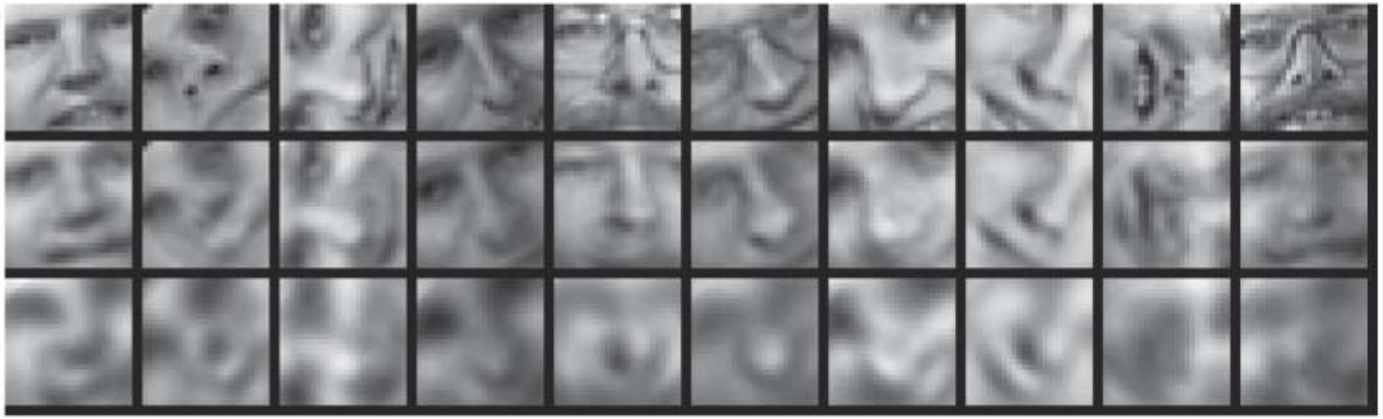
Logistic PCA

Regular PCA

(Hinton and Salakhutdinov, 2006)

Results

- Same thing with faces



Real Data

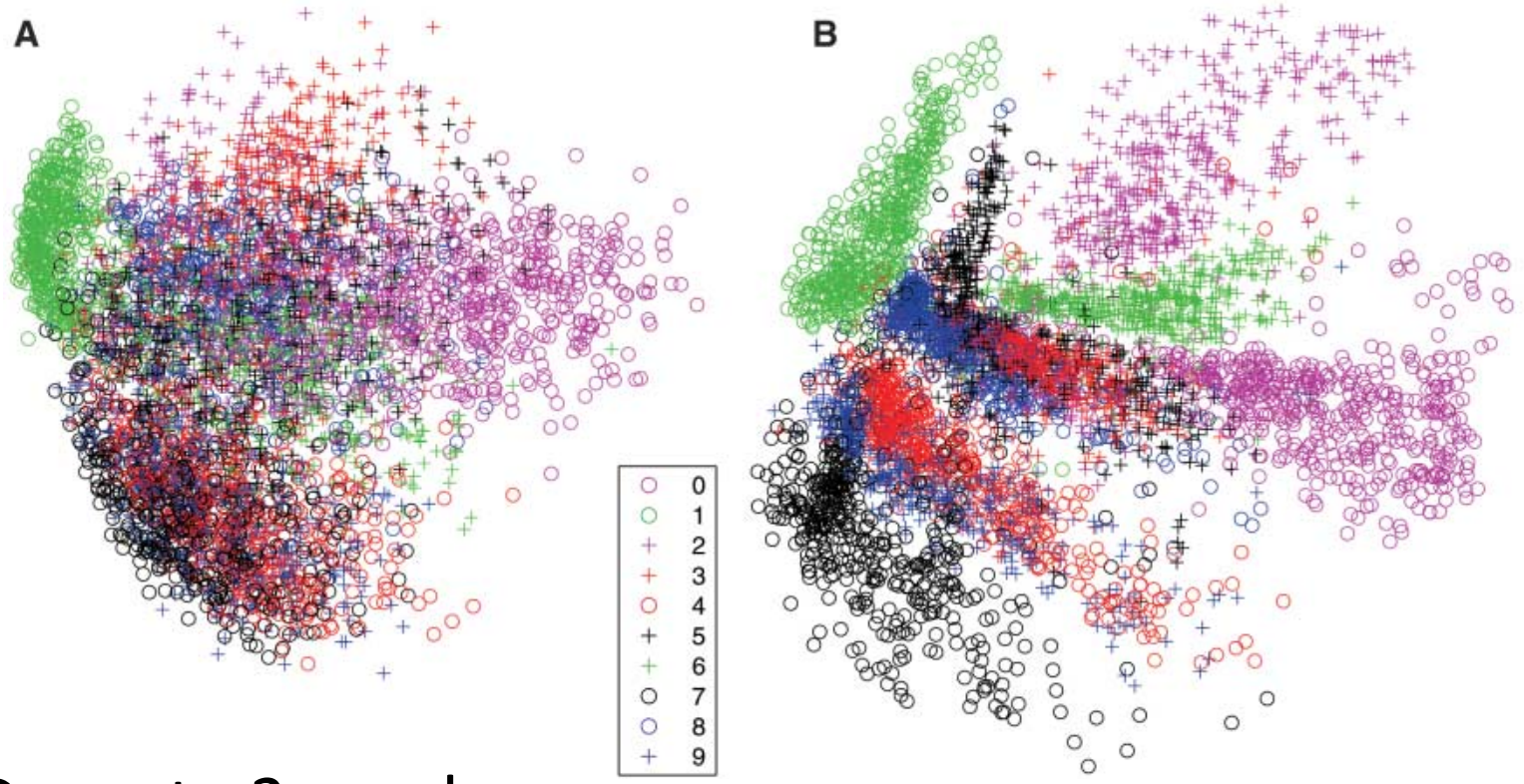
Autoencoder

Regular PCA

(Hinton and Salakhutdinov, 2006)

Results

- Features are better separated for classification

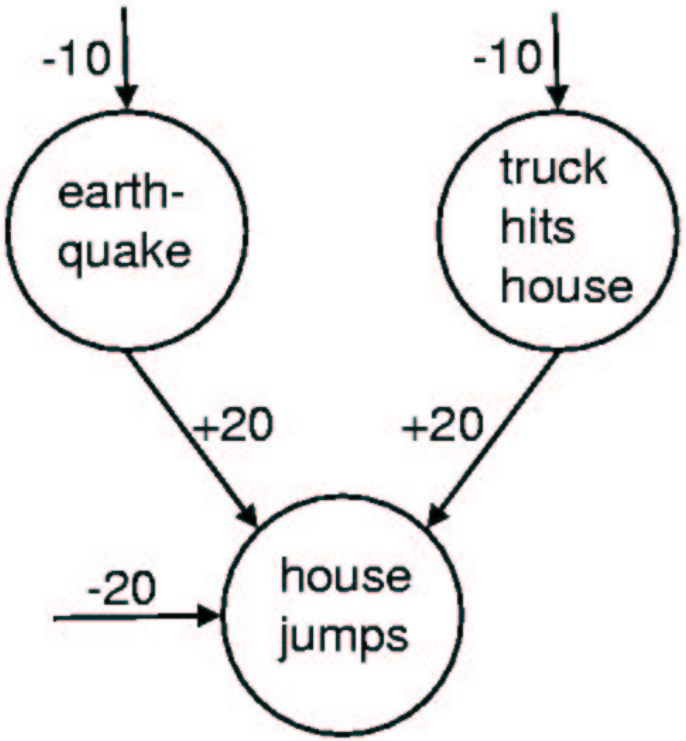


Down to 2 numbers with PCA

Down to 2 numbers with autoencoder

Deep Belief Net Training

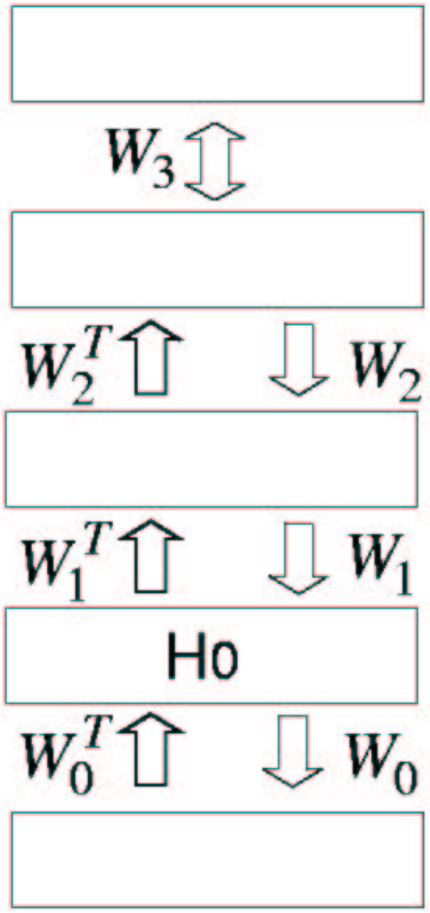
- Credit/blame problem
- Rare events causing another rare event
- These things will never be the cause at the same time
- Any training algorithm that raises both weights will be wrong



(Hinton, Osindero, Teh, 2006)

Deep Belief Net Training

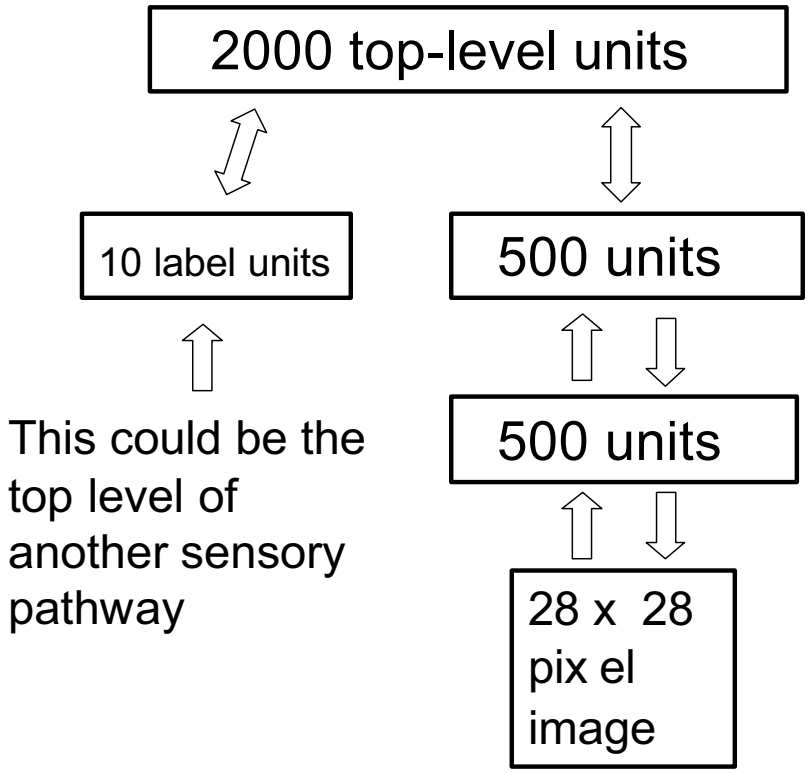
- Layered RBMs
- Up weights, down weights
- (Sensing/generation weights)
- No intralayer connections
- All layers same size
- Similar learning rules



(Hinton, Osindero, Teh, 2006)

Handwriting Network

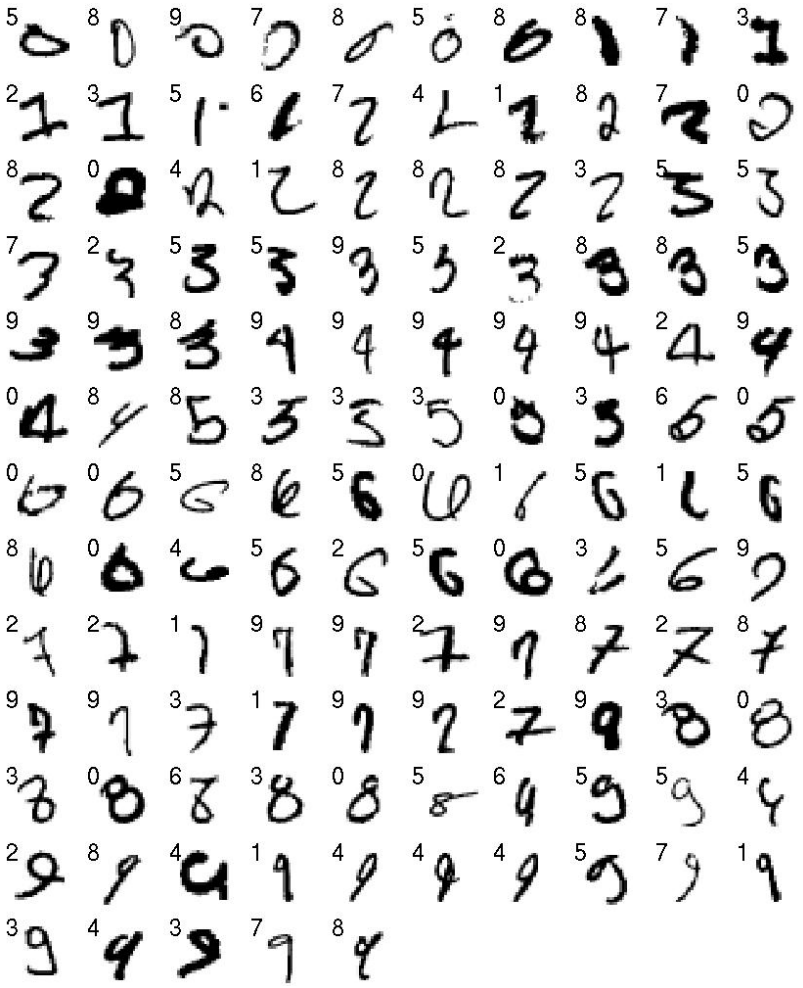
- Top 2 levels are associative memory
- During training of the top level, 10 units on the left give label
- During test, you take the cell with highest probability as the output



(Hinton, Osindero, Teh, 2006)

Performance

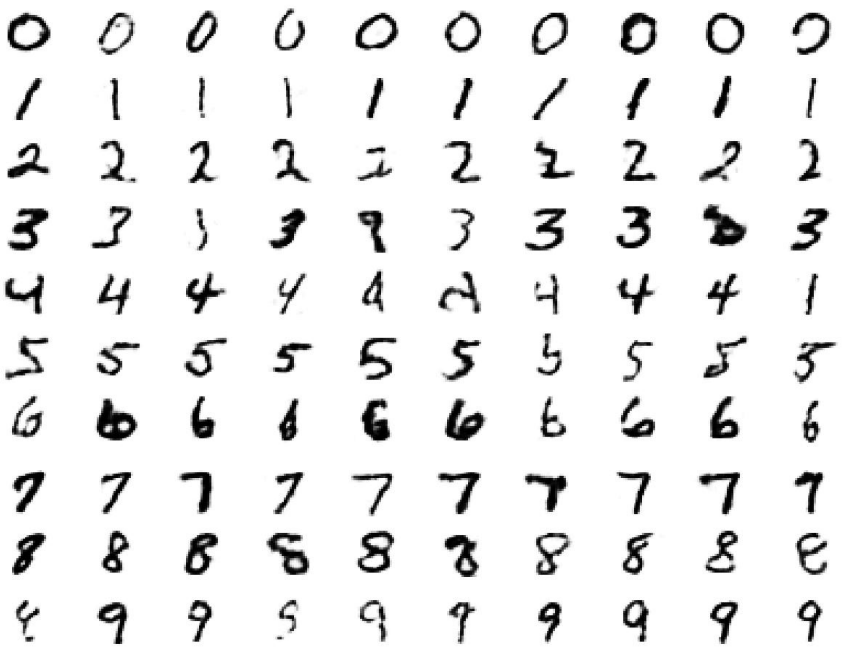
- Handwriting recognition
- 1.25% error result
- Normal feed forward network = 2.95%
- Clustering method = 3.1%



(Hinton, Osindero, Teh, 2006)

“Looking into the Mind of the Network”

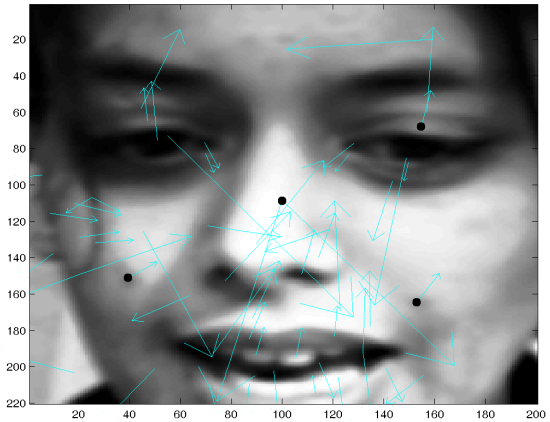
- Clamp the label units to one of the classes, and see what gets put out as a generative image

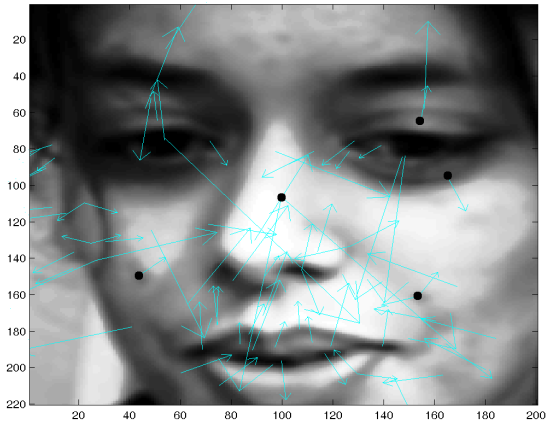


(Hinton, Osindero, Teh, 2006)

Facial Recognition

- Highest performing classifiers use "hand crafted features"
- Someone tells it what a face should look like
- Goes in and finds those features
- Deep learning can do this without knowing any characteristics of faces ahead of time

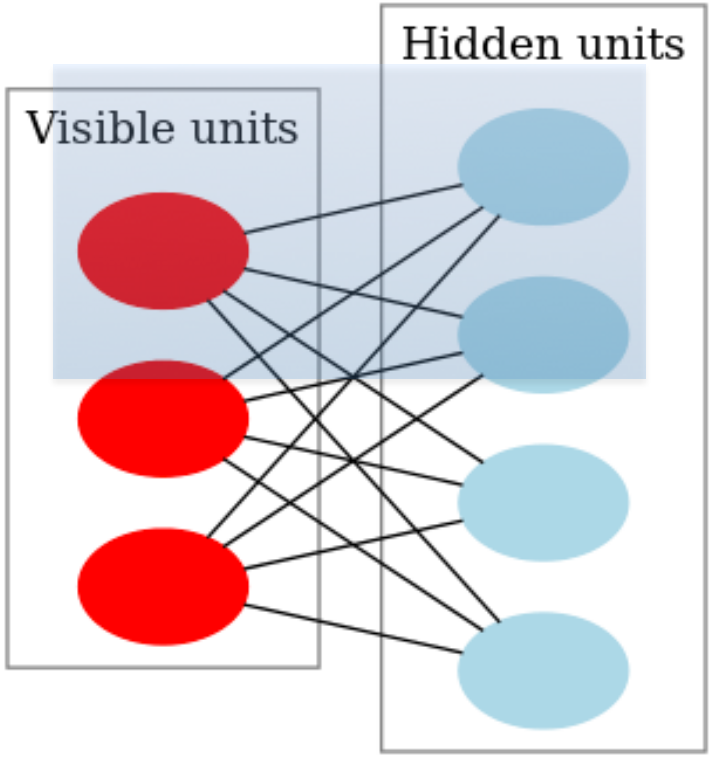




(Bisego et al., 2006)

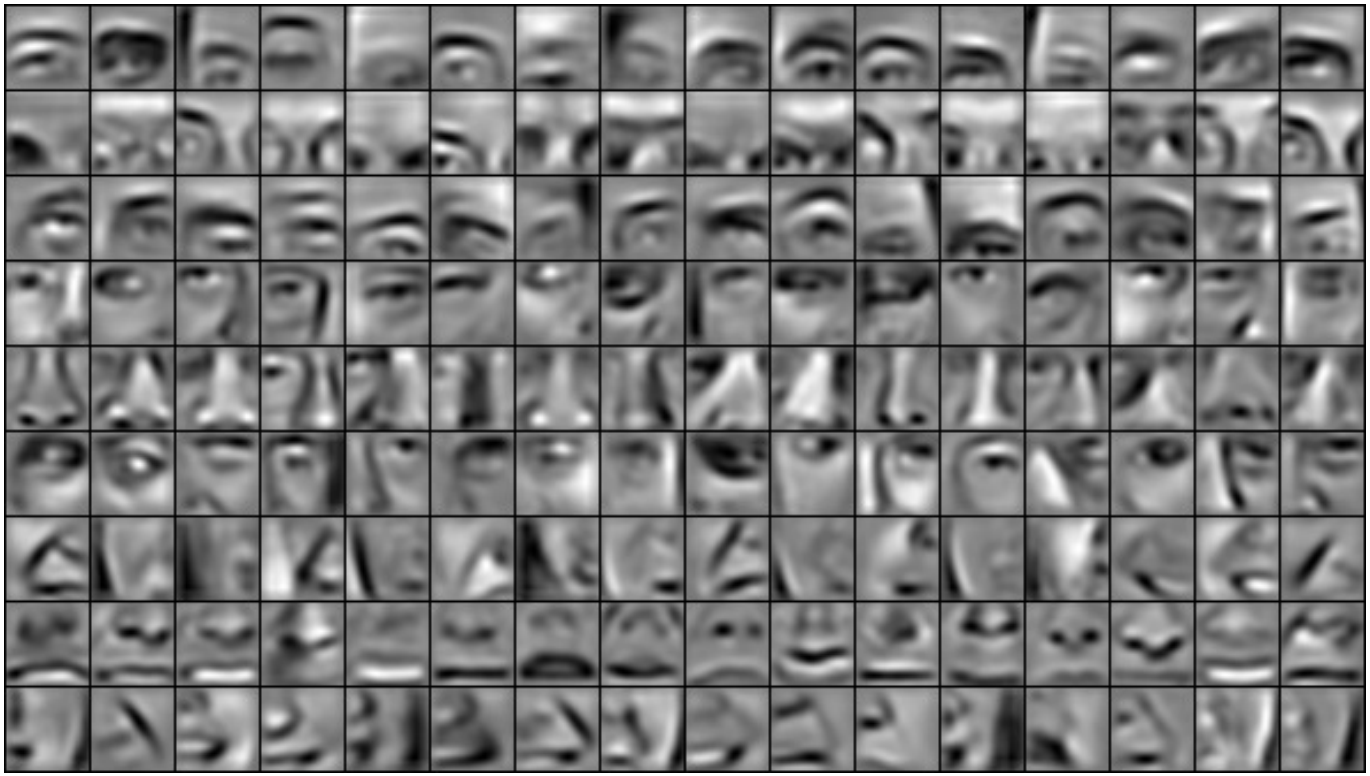
Convolutional RBM

- Similar in that hidden only connects to visible and vice versa
- But now connections are only within a local area
- Features can be important when they're within a particular area



Results

- Sample features from local areas
- 87% correct recognition



(Huang et al., 2012)

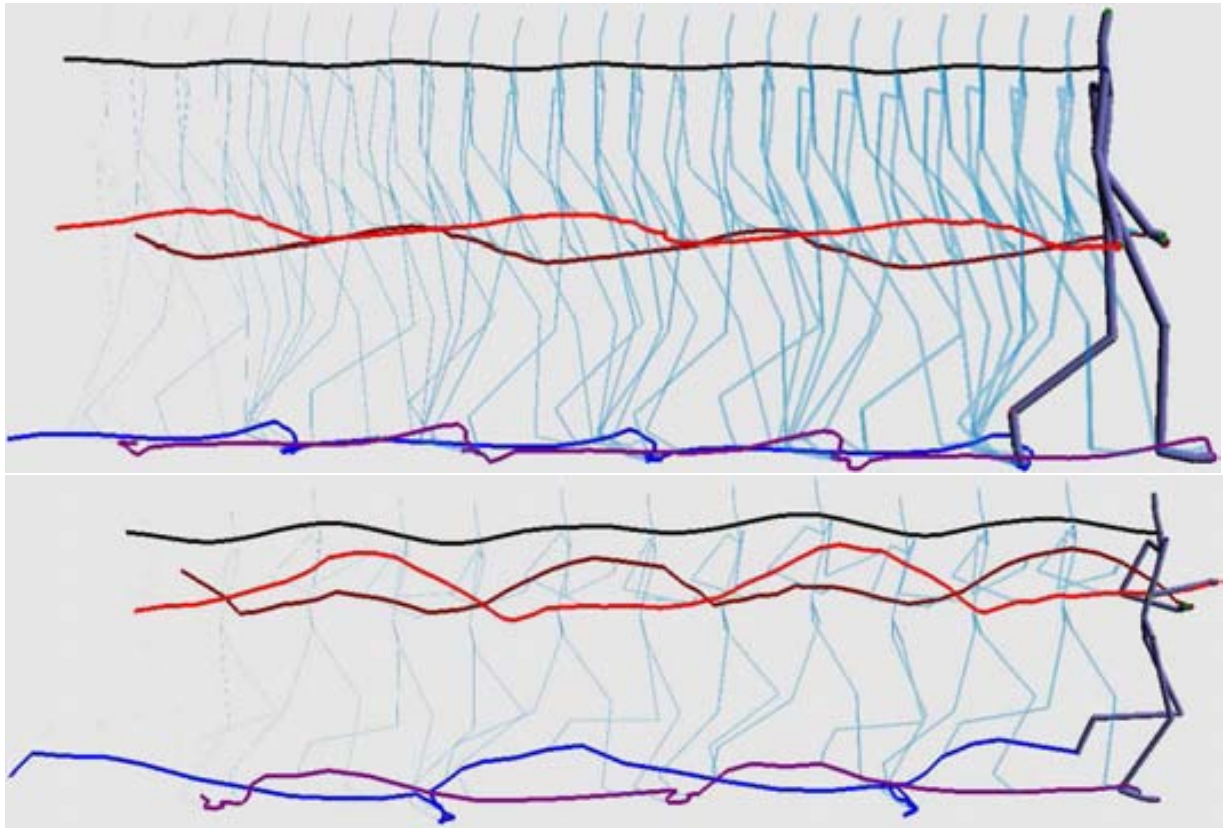
Speech Recognition

- Mohamed, Dahl, Hinton, 2012

Method	PER
Stochastic Segmental Models [31]	36%
Conditional Random Field [32]	34.8%
Large-Margin GMM [33]	33%
CD-HMM [34]	27.3%
Augmented conditional Random Fields [34]	26.6%
Recurrent Neural Nets [35]	26.1%
Bayesian Triphone HMM [36]	25.6%
Monophone HTMs [37]	24.8%
Heterogeneous Classifiers [38]	24.4%
Triphone HMMs discriminatively trained w/ BMMI [39]	22.7%
Monophone Deep Belief Networks(DBNs) (this work)	**20.7%**

Deep Networks Producing Motion

- Taylor, Hinton, Roweis 2007



Limitations

- Deep network still just falls down in energy into a desired state
- Can't generate a pattern through time
- Can't "improvise" in the absence of normal inputs

> Unfortunately, gradient descent and other 1st-order methods completely fail to properly train RNNs on large families of seemingly simple yet pathological synthetic problems that separate a target output and from its relevant input by many time steps (Bengio et al., 1994; Hochreiter and Schmidhuber, 1997). In fact, 1st-order approaches struggle even when the separation is only 10 timesteps (Bengio et al., 1994).

(Martens and Sutskever, 2011)

Recurrent Neural Networks

Learning Recurrent Neural Networks with Hessian-Free Optimization

---

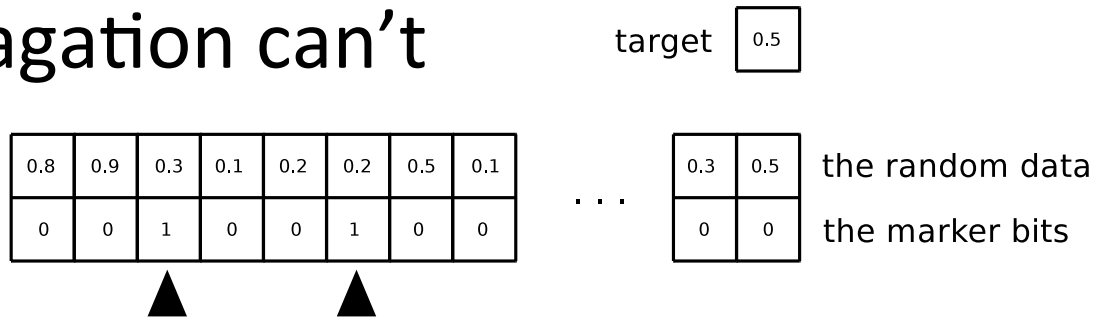
James Martens JMARTENS@CS.TORONTO.EDU
Ilya Sutskever ILYA@CS.UTORONTO.CA
University of Toronto, Canada

In this work we resolve the long-outstanding problem of how to effectively train recurrent neural networks (RNNs) on complex and difficult sequence modeling problems which may contain long-term data dependencies.

(Martens and Suskever, 2011)

Martens and Sutskever, 2011

- New training method - Hessian Free Optimization
- Creates fairly sparse networks
- Solves many problems unsolved by a class paper on backpropagation methods (Hochreiter and Schmidhuber, 1997)
- E.g. solves marked number addition, which backpropagation can't



Neural Network “Babbling”

- Sutskever et al., 2011

The meaning of life is the tradition of the ancient human reproduction: it is less favorable to the good boy for when to remove her bigger. In the show’s agreement unanimously resurfaced. The wild pasteured with consistent street forests were incorporated by the 15th century BE. In 1996 the primary rapford undergoes an effort that the reserve conditioning, written into Jewish cities, sleepers to incorporate the .St Eurasia that activates the population. Mar??a Nationale, Kelli, Zedlat-Dukastoe, Florendon, Ptu’s thought is. To adapt in most parts of North America, the dynamic fairy Dan please believes, the free speech are much related to the

while he was giving attention to the second advantage of school building a 2-for-2 stool killed by the Cultures saddled with a half-suit defending the Bharatiya Fernall ’s office . Ms . Claire Parters will also have a history temple for him to raise jobs until naked Prodiena to paint baseball partners , provided people to ride both of Manhattan in 1978 , but what was largely directed to China in 1946 , focusing on the trademark period is the sailboat yesterday and comments on whom they obtain overheard within the 120th anniversary , where many civil rights defined , officials said early that forms , ” said Bernard J. Marco Jr. of Pennsylvania , was monitoring New York

Recurrent network with the Stiefel information for logistic regression methods Along with either of the algorithms previously (two or more skewprecision) is more similar to the model with the same average mismatched graph. Though this task is to be studied under the reward transform, such as (c) and (C) from the training set, based on target activities for articles a ? 2(6) and (4.3). The PHDPic (PDB) matrix of cav’va using the three relevant information contains for tieming measurements. Moreover, because of the therap tor, the aim is to improve the score to the best patch randomly, but for each initially four data sets. As shown in Figure 11, it is more than 100 steps, we used ?? \to \infty with 1000

Google Brain

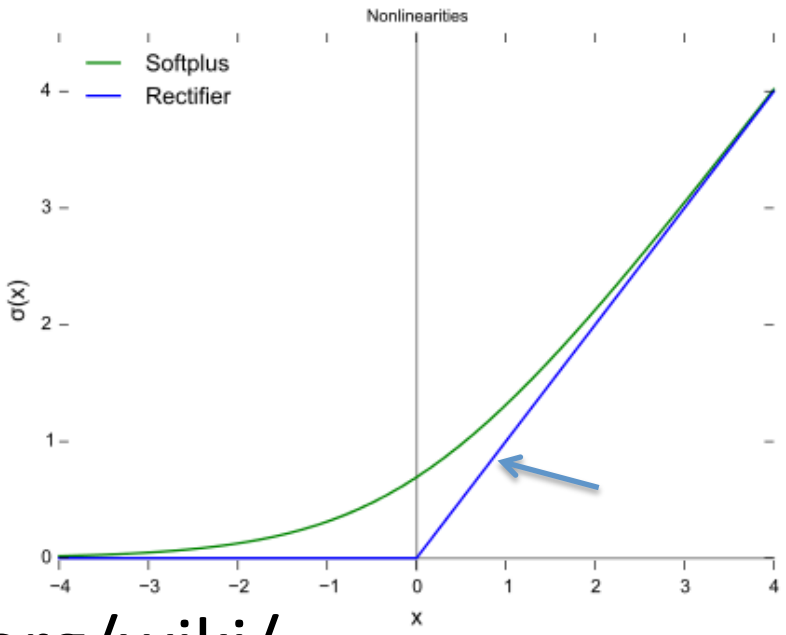
- People involved
  - Andrew Ng, Geoff Hinton, Ray Kurzweil
- DeepMind Technologies
  - Bought for $600 million
- http://www.zdnet.com/article/google-brain-simulator-teaches-itself-to-recognize-cats/

Stuff that's Surprisingly Recent

- Rectified Linear Units (ReLU) can do almost everything (i.e. the type of nonlinearity doesn't seem to be critical)
- Must use a better gradient descent method
  - Recursive Least Squares
  - Stochastic Gradient Descent
- Proper Initialization
  - Orthonormal Initialization (Saxe et al. 2014)
  - Random Walk Initialization (Sussillo et al. 2015)
  - Kaiming He Initialization

Rectified Linear Units

- Older stuff focused on tanh, sigmoids, but with modern techniques, a ramp works fine.
- ReLU = Rectified Linear Unit
- f(x)=max(0,x)



- https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

Random Walk Initialization

- Back propagated error signals can decrease or increase exponentially as a function of the distance into the network
- Start with the linear case, neural activations v propagating from layer d to d+1

$$\vec{v}_{d+1} = gW_d\vec{v}_d$$

- Want overall value to go up as often as down, i.e. undergo “Random Walk”

(Sussillo et al. 2014)

Random Walk Initialization

- To achieve random walk, make the norms of the vectors similar at every layer

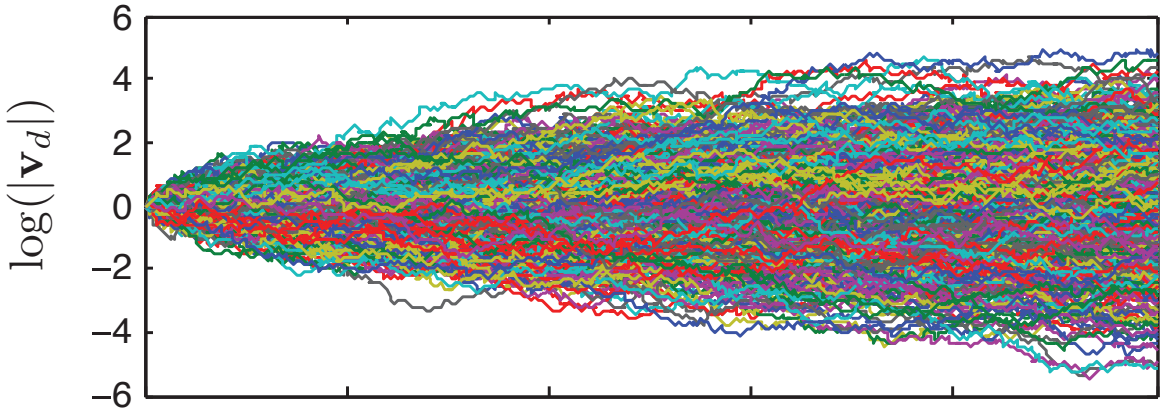
$$\log\left|\vec{v}_{d+1}\right| - \log\left|\vec{v}_{d}\right| = 0 \quad \text{(on average)}$$

- Turns out (in appendix), this happens in the linear case when

$$g = \exp\left(\frac{1}{2N}\right) \approx 1 + \frac{1}{2N}$$

Random Walk Initialization

- For a (linear) network with D=500 layers with N=100 neurons in each, g=1.005



- (This is what a random walk looks like.)

Random Walk Initialization

- For non-linear networks, g has no analytical form, but can be found empirically.
- E.g. g= 1.42 for ReLU with N = 100
- Recipe for any network:
  - Pick a g and initialize.
  - Propagate a bunch of inputs
  - Calculate $\log\left|\vec{\partial}_1\right|, \log\left|\vec{\partial}_D\right|$
  - Tweak g until they're about equal
  - Initialize the network, and train it

$$\vec{\partial}_d = \frac{\partial E}{\partial \vec{y}_d}$$

Orthogonal Initialization

- Quote:

  "We show that, under certain conditions on the training data, unsupervised pretraining can find this special class of initial conditions, while scaled random Gaussian initializations cannot. We further exhibit a new class of random orthogonal initial conditions on weights that, like unsupervised pre-training, enjoys depth independent learning times. We further show that these initial conditions also lead to faithful propagation of gradients even in deep nonlinear networks, as long as they operate in a special regime known as the edge of chaos."

(Saxe, McClelland, Ganguly, 2014)

Orthogonal Initialization

- First part of the paper is about how you want "decoupled connectivity modes", which can evolve independently from each other.
- This corresponds to simpler overall dynamics
- You can achieve it through initial conditions alone, by initializing your network properly
- The Hinton autoencoder deep networks also have these simpler dynamic properties

(Saxe, McClelland, Ganguly, 2014)

Orthogonal Initialization

- First make a matrix with random normal elements (mean of 0, variance of 1) the same size as your weights matrix
- Run a single value decomposition on that matrix (which gives you a bunch of orthonormal eigenvectors in its U matrix)
- Use those as your initial weights

(Saxe, McClelland, Ganguly, 2014)