Neural Networks 2017

Dr. Wojtek Kowalczyk

wojtek@liacs.nl

Agenda

- Course objectives, organization, grading
- Neural Networks: motivation, history, context
- Deep Learning Revolution
- Statistical Pattern Recognition
- Assignment 0

Key Objectives

1. Learn key concepts behind the “traditional” neural networks (***Perceptron, MLP, Backpropagation***)
2. Learn basics of ***Deep Learning*** *and its* ***applications***
3. Learn key techniques for ***parameter optimization*** (finding optima of functions of many variables)
4. Master modern tools for training deep networks on GPUs (***Python + Theano/Tensorflow/Keras***)
5. Gain ***practical experience*** with applying Neural Networks to various problems

Course Organization

Lectures:

theory + example applications + practical tips

Wednesdays, 11:15-13:00, rooms: 3xB02+10x312

Computer Lab:

Wednesdays, 9:00-10:45, Rooms 302-304, 306-308

Presence during lectures & comp. lab is ***not compulsory***

Grading

- Course value: 6 ects (workload: **2days/week**)

- Final grade is based on two components:
  - written exam (40%):
    various questions/problems related to theory and practice.

  - practical (60%):
    three programming assignments:

    Assignment 0: elementary Python skills (not graded)

    Assignment 1: Linear Models + MLP

    Assignment 2: Deep Neural Networks on GPU’s

    Assignment 3: ***A DL Challenge***

  - **Both parts (exam+prac) must be “passed” (grade > 5.5)!**

Textbook

www.deeplearningbook.org

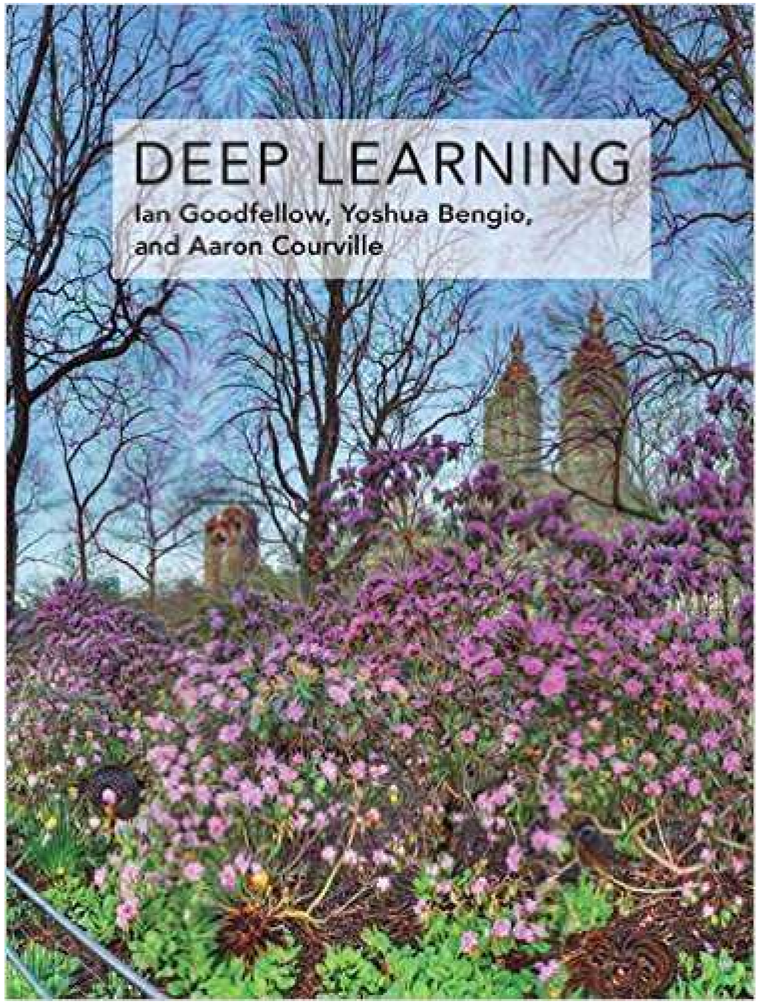
Available on-line; print it to PDF or check an example pdf dump *(a link will follow soon)*

The book is about 800 pages long; don't despair – we will use just 30%!

This week: Chapter 1

Next week: Chapter 5

(Chapters 2-4: "background")



Additional Resources

Deep Learning in Matlab:

- github.com/rasmusbergpalm/DeepLearnToolbox
  (a very concise intro to Deep Learning + Matlab code)

- www.aston.ac.uk/eas/research/groups/ncrg/resources/netlab/

Deep Learning:

- Kevin Duh: Neural Networks and Deep Learning
  cl.naist.jp/~kevinduh/a/deep2014/

- http://deeplearning.net/tutorial/

- Andrew Ng on Deep Learning:
  www.youtube.com/watch?v=n1ViNeWhC24

Neural Networks:

motivation

history

context

How could we program computers to do:

- Optical Character Recognition
- Text2Speech translation
- Speech2Text translation
- Robot arm control
- Face recognition
- Driving a car
- Detecting fraud with credit cards
- Predicting $/€ exchange rate

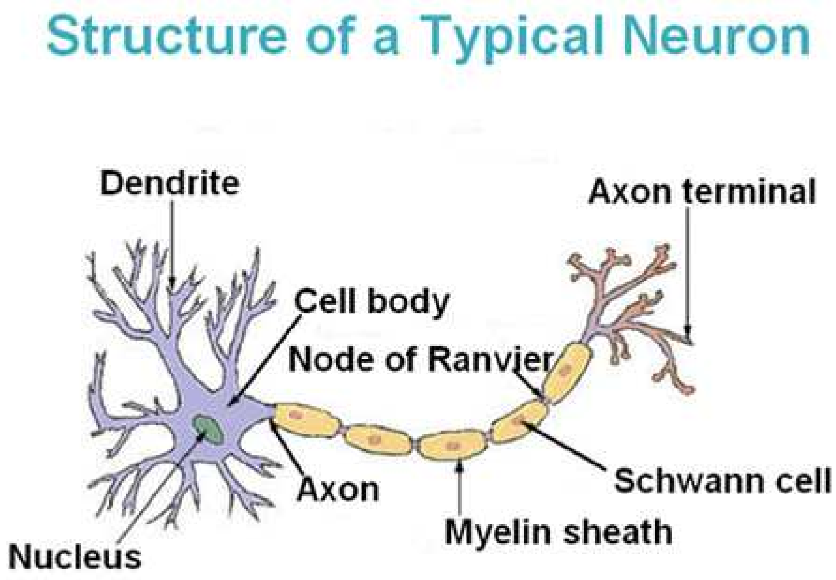
How people do it?

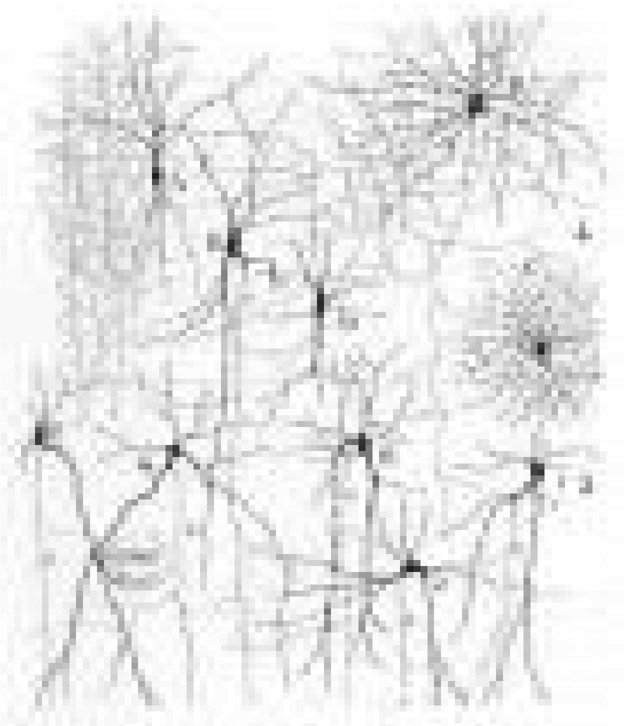
“training data” + “self-learning”

The biological model

Human brain

around 10^{11} neurons, and 10^{15} interconnections (synapses)

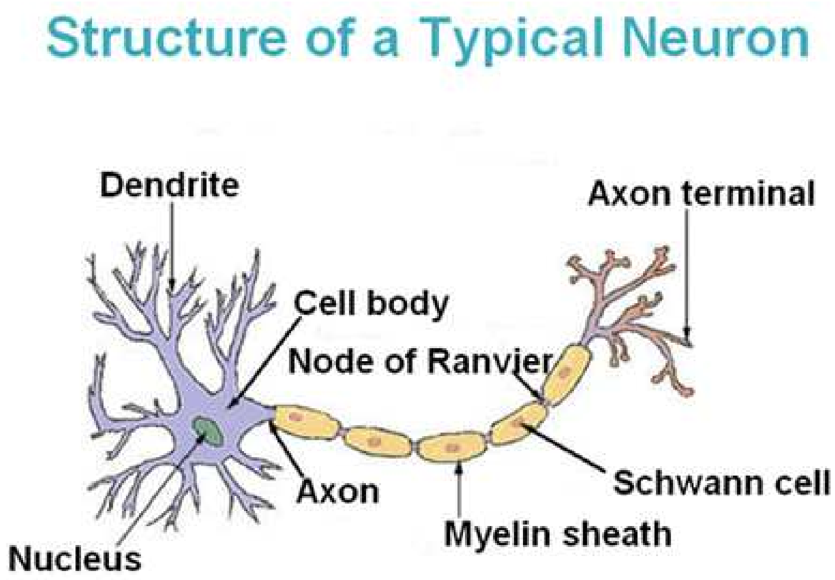




The biological model

How a neuron works ?

- It receives input signals through its dendrites
- The input signals generate difference of electrical potential on the cell membrane
- The difference is propagated to the axon hillock
- A train of electrical impulses is generated along the axon
- The impulses on the axon generate the release in the synaptic space of some neurotransmitters



Learning by changing the topology & thickness of connections

Basic principle

Artificial neural network (ANN)= set of highly interconnected simple processing units (also called neurons)

Processing unit = simplified model of a neuron

Learning = finding the architecture of connections
and their weights

Computer versus Brain

Von Neumann machine

•One or a few high speed (ns) processors with considerable computing power

•One or a few shared high speed buses for communication

•Sequential memory access by address

•Problem-solving knowledge is separated from the computing component
•Hard to be adaptive

Human Brain

- Large number (10^{11}) of low speed processors (ms) with limited computing power
- Large number (10^{15}) of low speed connections
- Content addressable recall (CAM)
- Problem-solving knowledge resides in the connectivity of neurons
- ***Adaptation by changing the connectivity (topology & thickness of connections)***

Human Brain

- **graceful degradation**:
  its performance degrades gracefully under partial damage.
- **self-learning:**
  it can learn (reorganize itself) from experience.
- **self-healing:**
  partial recovery from damage is possible
- **massive parallelism:**
  performs parallel computations extremely efficiently. E.g., complex visual perception takes less than 100 ms, that is, 10 processing steps!
- **intelligence** and **self-awareness** (whatever it means)

ANN History: Roots …

Roots of Neural Networks are in:

Neurobiological studies (more than one century ago):

- How do nerves behave when stimulated by different magnitudes of electric current? Is there a minimal threshold needed for nerves to be activated? Given that no single nerve cell is long enough, how do different nerve cells communicate among each other?

Psychological studies:

- How do animals learn, forget, recognize and perform other types of tasks?

Psycho-physical experiments

- Understand how individual neurons and groups of neurons work.

ANN History: First steps

- **Pitts & McCulloch (1943)**
  - First mathematical model of biological neurons
  - All Boolean operations can be implemented by these neuron-like nodes (with different threshold and excitatory/inhibitory connections).
  - Competitor to Von Neumann model for general purpose computing device
  - Origin of automata theory.
- **Hebb (1949)**
  - Hebbian rule of learning: increase the connection strength between neurons i and j whenever both i and j are activated.
  - Or increase the connection strength between nodes i and j whenever both nodes are simultaneously ON or OFF.

ANN History: Perceptron

- **Early booming (50's – early 60's)**
  - Rosenblatt (1958)
    - Perceptron: network of threshold nodes for pattern classification. Perceptron learning rule – first learning algorithm
    - Perceptron convergence theorem:

      everything that can be represented by a perceptron can be learned
  - Widrow and Hoff (1960, 1962)
    - Learning rule that is based on minimization methods
  - Minsky's attempt to build a general purpose machine with Pitts/McCullock units

ANN History: Setback …

- **The setback (mid 60's – late 70's)**
  - Minsky and Papert publish a book "Perceptrons" (1969):
    - Single layer perceptrons cannot represent (learn) simple functions such as XOR
    - Multi-layer of non-linear units may have greater power but there was no learning algorithm for such nets
    - Scaling problem: connection weights may grow infinitely
  - ***US Defense/Government stop funding research on ANN***

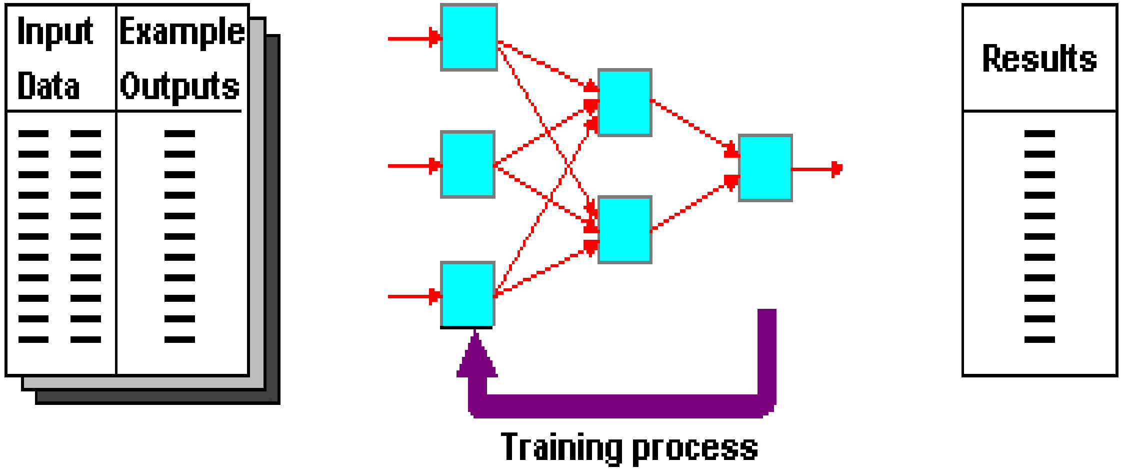
ANN History: Backpropagation

- **Renewed enthusiasm and progress (80’s – 90's)**
  - New techniques
    - Backpropagation learning for multi-layer feed forward nets (with non-linear, differentiable node functions)
    - Physics inspired models (Hopfield net, Boltzmann machine, etc.)
    - Unsupervised learning (LVQ nets, Kohonen nets)
  - Impressive applications (character recognition, speech recognition, text-to-speech transformation, process control, associative memory, etc.)

But:

- Criticism from Statisticians, Neurologists, Biologists, Ordinary Users, ...
- Lots of ad-hoc solutions, "wild creativity"
- A lot of rubbish is produced ...

Supervised Learning



- Training data: inputs & targets (= desired outputs)
- Network: a complicated function with many parameters to tune
- Training: a process of tweaking the parameters to minimize the errors of predictions (outputs should be close to targets)

$$Error = \sum_{Examples} (\text{output}_i - \text{target}_i)^2$$

- Generalization: we want the network to perform well on data that was not used during the training process!

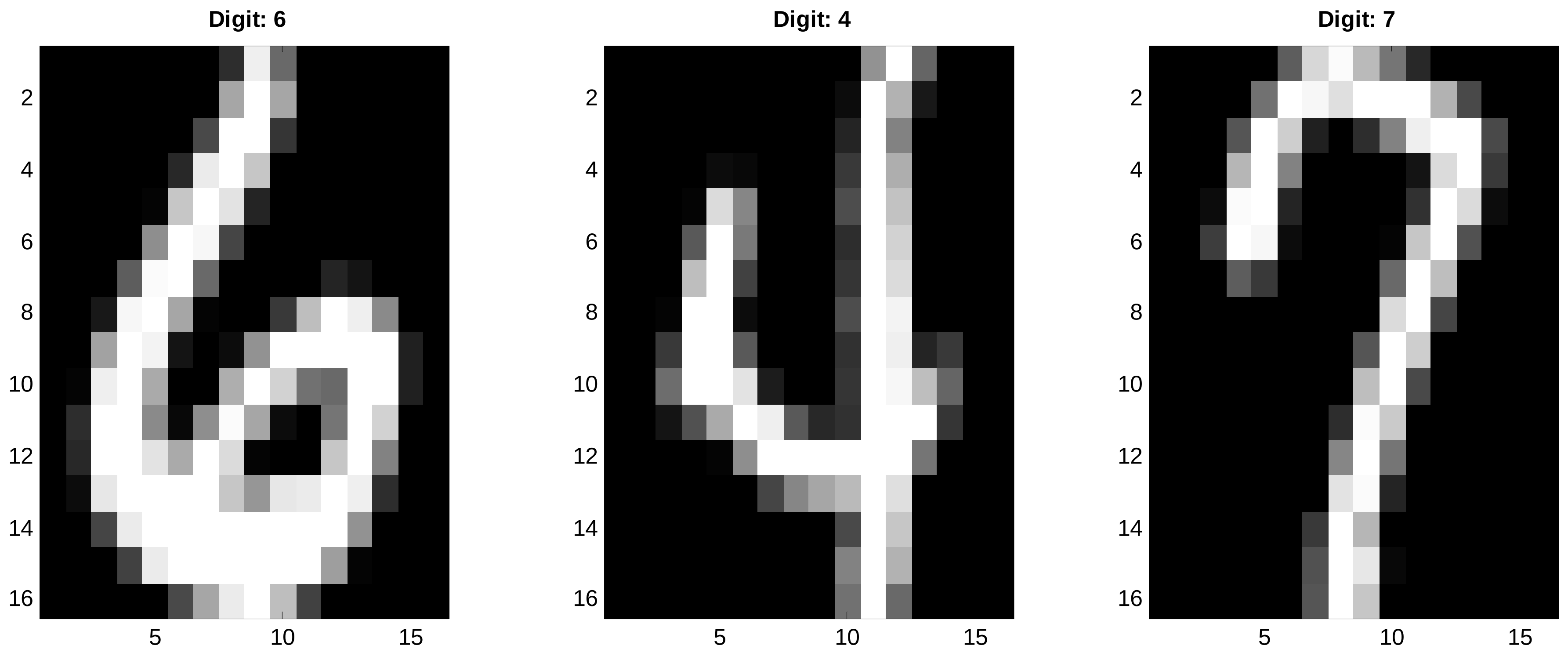
Example: Recognizing Handwritten Digits

Training set: 2000 images of digits

Each image: 16x16 pixels = a vector of 256 components

The network: 256 inputs and 10 outputs

Test set: 1000 images of digits



Backpropagation algorithm

- The backpropagation algorithm searches for weight values that **minimize the total error of the network**
- It consists of the **repeated application of two passes**:
  - **Forward pass**: in this step the network is activated on one example and the **error** of each neuron of the output layer is **computed**.
  - **Backward pass**: in this step the network error is used for updating the weights. Starting at the output layer, **the error is propagated backwards through the network, layer by layer,** with help of the **generalized delta rule**. Finally, all weights are updated.
- No guarantees of convergence
  (when learning rate too big or too small)
- In case of convergence: local (or global) minimum
- In practice: try several starting configurations and learning rates.

Three examples of MLP

NetTalk: a network that reads aloud texts

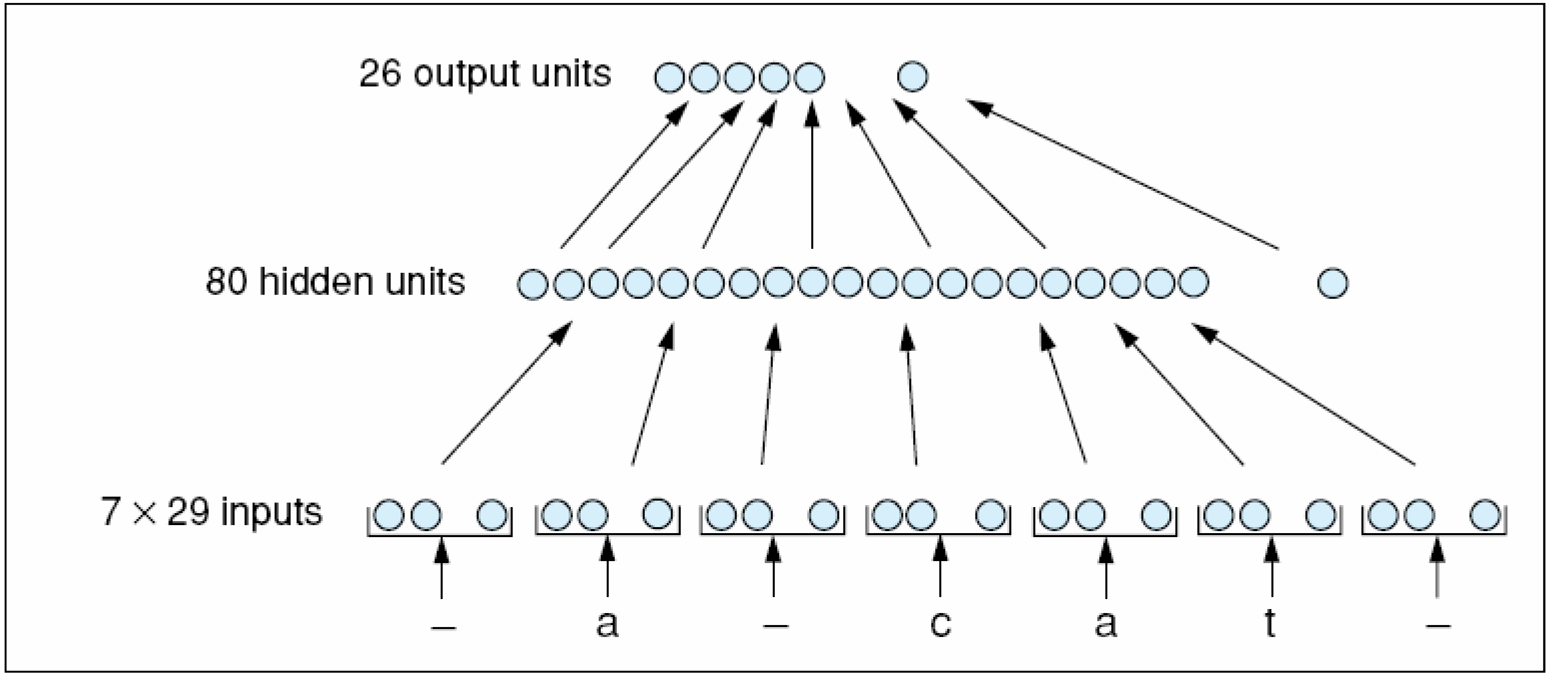
ALVINN: a Neural network that drives a car

Falcon: a real-time system for detecting fraud with credit card transactions

NETtalk (Sejnowski & Rosenberg, 1987)

- The task is to learn to pronounce English text from examples (text-to-speech)
- Training data: a list of <phrase, phonetic representation>
- Input: 7 consecutive characters from written text presented in a moving window that scans text
- Output: phoneme code giving the pronunciation of the letter at the center of the input window
- Network topology: 7x29 binary inputs (26 chars + punctuation marks), 80 hidden units and 26 output units (phoneme code). Sigmoid units in hidden and output layer

NETtalk



NETtalk (contd.)

- Training protocol: 95% accuracy on training set after 50 epochs of training by full gradient descent.
  78% accuracy on a test set

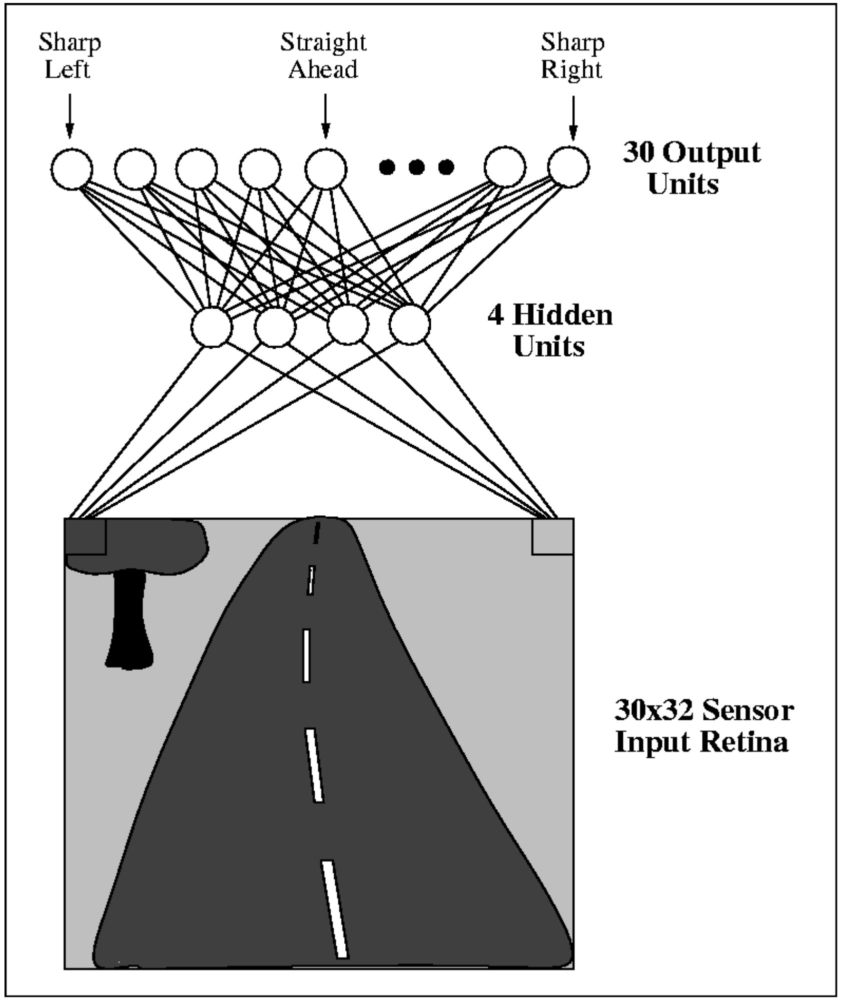
- DEC-talk: a rule-based system crafted by experts (a decade of efforts by many linguists)

- Functionality/Accuracy almost the same

- Try: http://cnl.salk.edu/Media/nettalk.mp3

ALVINN (1989)

D.A. Pomerleau, Autonomous Land Vehicle In a Neural Network (NIPS '89)



FALCON: A Fraud Detection System

http://www.fico.com/en/Products/DMApps/Pages/FICO-Falcon-Fraud-Manager.aspx

Right now, leading institutions around the world trust Fair Isaac's Falcon™ Fraud Manager to protect more than 450 million active credit and debit cards. Why? They know they'll receive regular technological advances which will allow them to stay a step ahead of new and emerging fraud types. Protecting 65% of the world's credit card transactions, Falcon detects fraud with pinpoint accuracy via **proven neural network models** and other proprietary predictive technologies. Debit, credit, oil and retail card issuers in numerous marketplaces rely on Falcon to detect and stop fraudulent transactions - and combat identity theft - in real time.

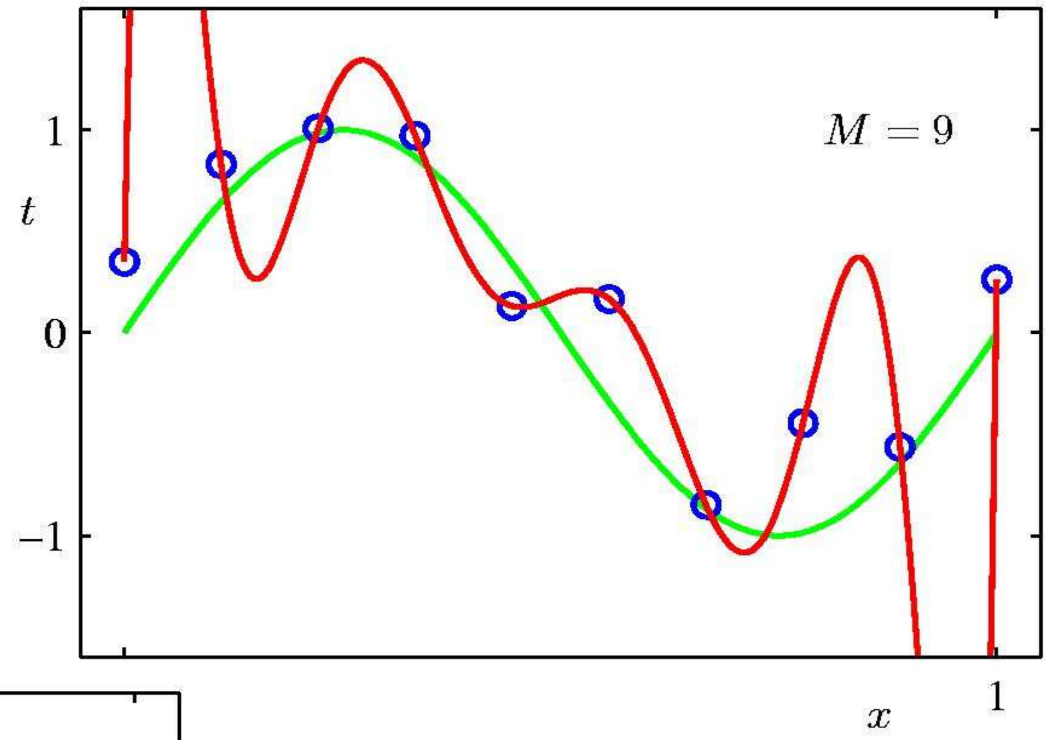
Why only “shallow networks” ?

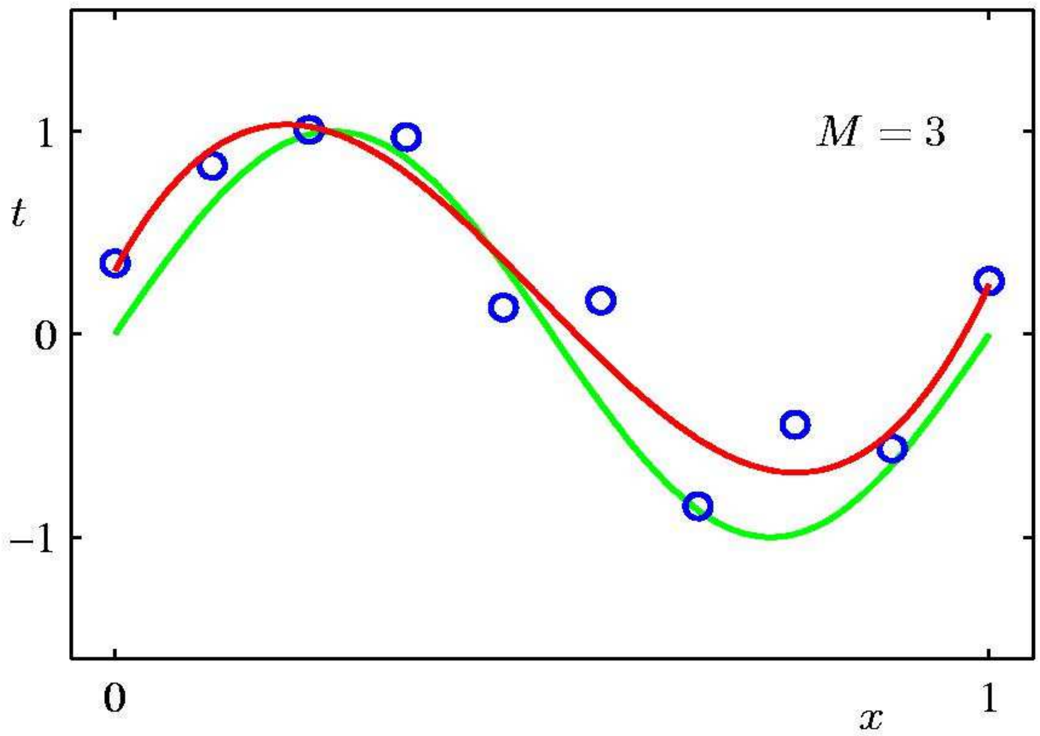
- In practice, "classical" multi-layer networks work fine only for a very small number of hidden layers (typically 1 or 2) - this is an empirical fact ...

- Adding layers is harmful because:
  - the increased number of weights quickly leads to data overfitting (lack of generalization)
  - huge number of (bad) local minima trap the gradient descent algorithm
  - vanishing or exploding gradients (the update rule involves products of many numbers) cause additional problems

Overfitting:

which polynomial better approximates the green line?





The more parameters the higher the risk of overfitting !

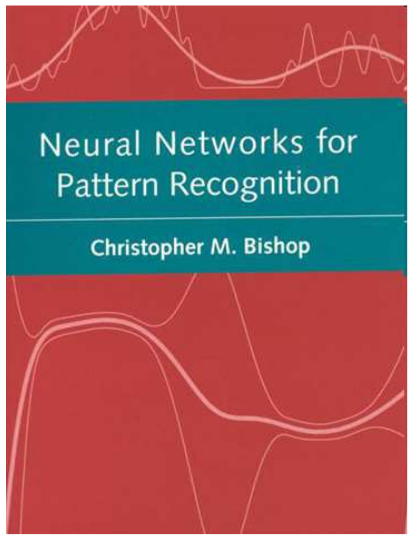
Why do we need many layers?

- In theory, one hidden layer is sufficient to model any function with an arbitrary accuracy; however, the number of required nodes and weights grows exponentially fast
- The deeper the network the less nodes are required to model "complicated" functions
- Consecutive layers "learn" features of the training patterns; from simplest (lower layers) to more complicated (top layers)
- Visual cortex consists of about 10 layers

An end of hype?

Dominance of new techniques (90's - 2005):

- Support Vector Machines (Vapnik)
- Kernel Methods (Vapnik, Scholkopf, ...)
- Ensemble methods (Breiman, Hasti, Tibshirani, Friedman,...)
- Random Forests
- ***Neural Networks lose their prominent role in the fields of Pattern Recognition and Machine Learning***
- ***since 1996 no new textbooks on Neural Networks !***



Deep Learning Revolution

1989 : LeCun et al: A Convolutional Network for OCR (AT&T)

~ 2006 : G. Hinton's group: new ideas, inventions, applications

- Restricted Boltzmann Machine as building block for DNN
- Contrastive Divergence algorithm
- Combine supervised with unsupervised learning
- Deep Belief Networks
- Stacked Auto-Encoders
- Deep Recurrent Networks

Many spectacular applications beating existing approaches

Enabling Factors

- Availability of “Big Data”: the more data we have the better we can train a network

- Powerful Hardware (GPU’s): speedup of the training process by 100-1000 times reduces the training time from years to hours

- New algorithms and architectures: leaving the MLP standard behind …

- ***Many spectacular successes!***

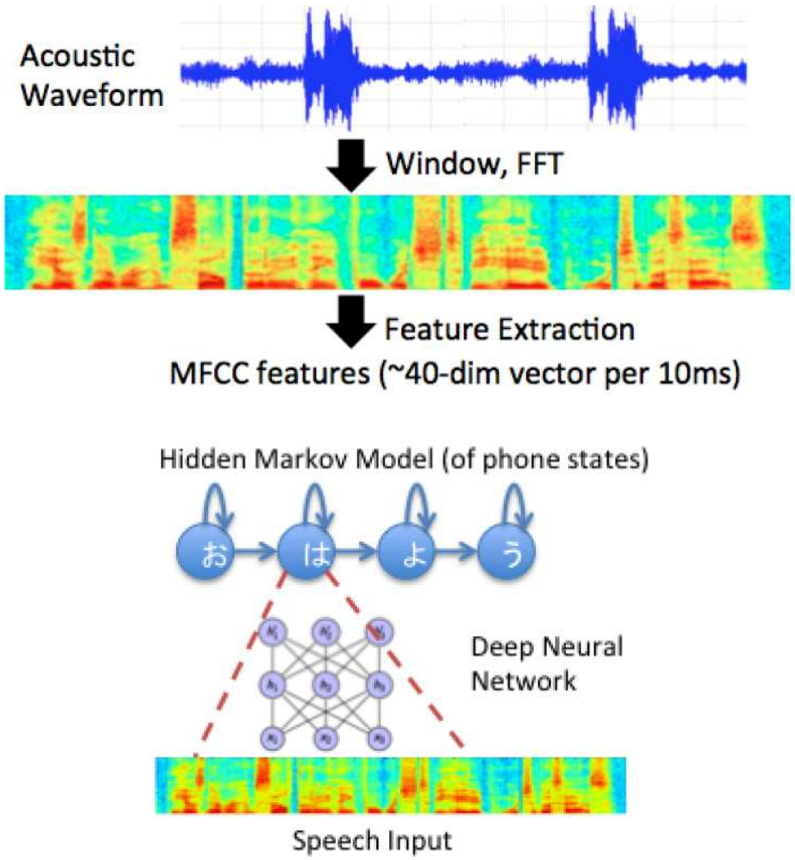
Key Architectures

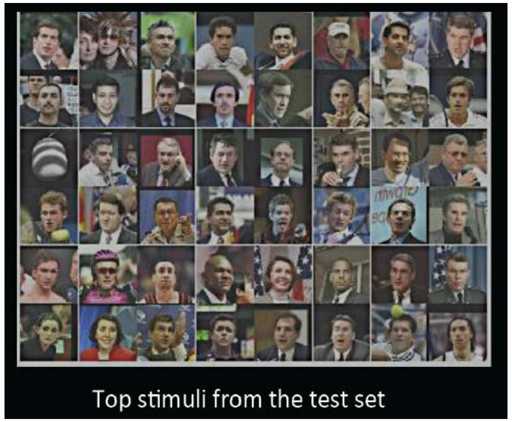
- Convolutional Networks: when adding layers enforce hidden nodes to learn "local features" - that reduces the number of parameters
- Autoencoders: add layers one by one, training them separately to learn a hierarchy of features
- Deep Recurrent Networks: networks for modeling sequential data
- Restricted Boltzmann Machines: a generative model (a pdf) of the training data

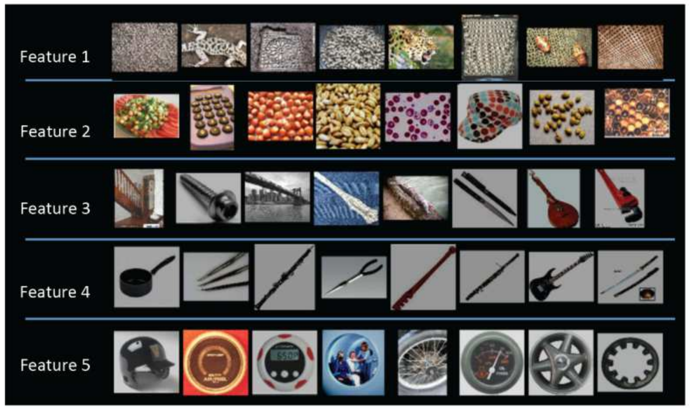
Autonomous Land Vehicle
(DARPA's GrandChallenge 2005 contest)

Other success stories

http://cl.naist.jp/~kevinduh/a/deep2014/140121-ResearchSeminar.pdf







LeNet: OCR system for digits

training:
60.000 images

testing:
10.000 images

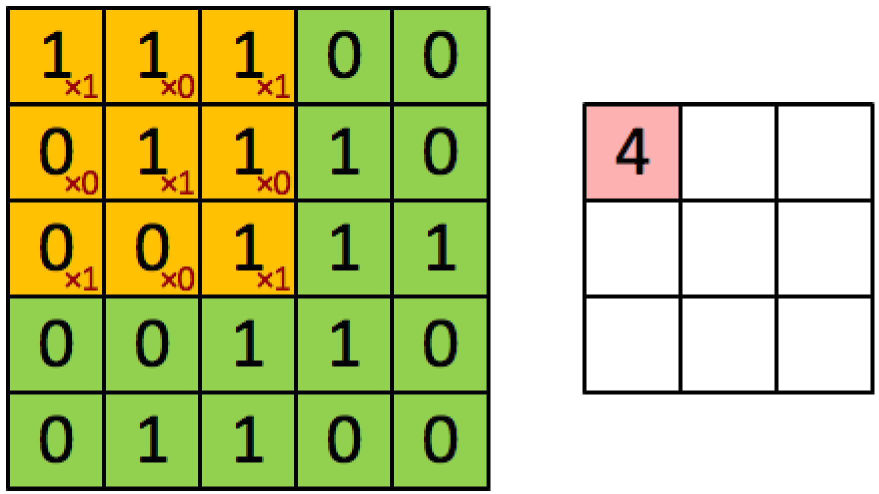
each image:
32x32 pixels

accuracy: 99.7%

A Convolutional Filter

Let us suppose that in an input image we want to find locations that look like a 3x3 cross. Define a matrix F (called a **receptive field** or **convolutional filter**) and "convolve it" with all possible locations in the image. We will get another (smaller) matrix with "degrees of overlap":

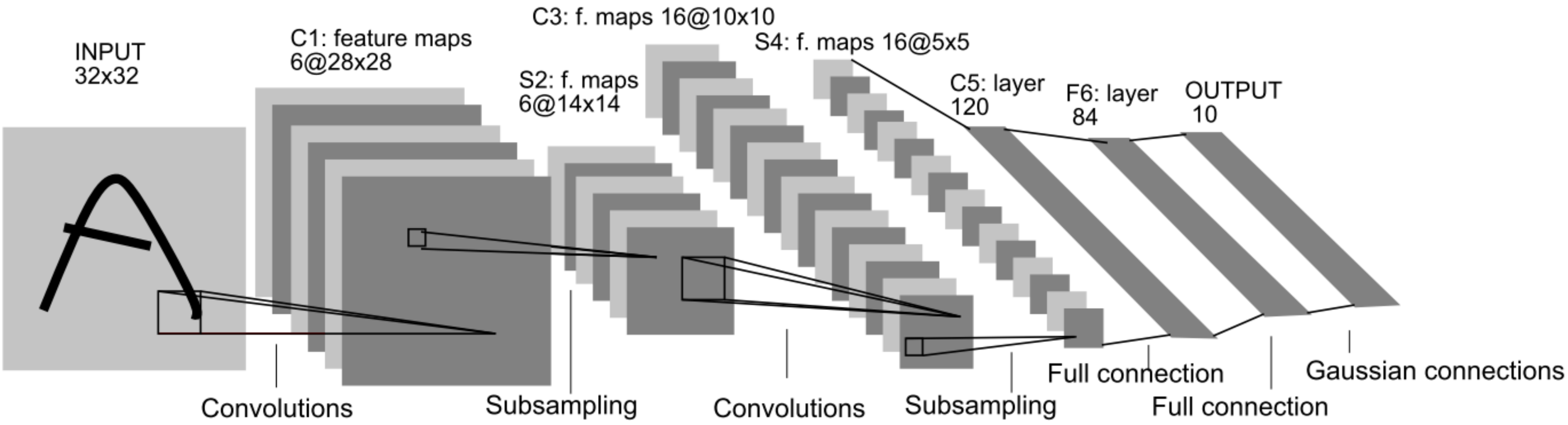
$$F = \begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{matrix}$$



Image

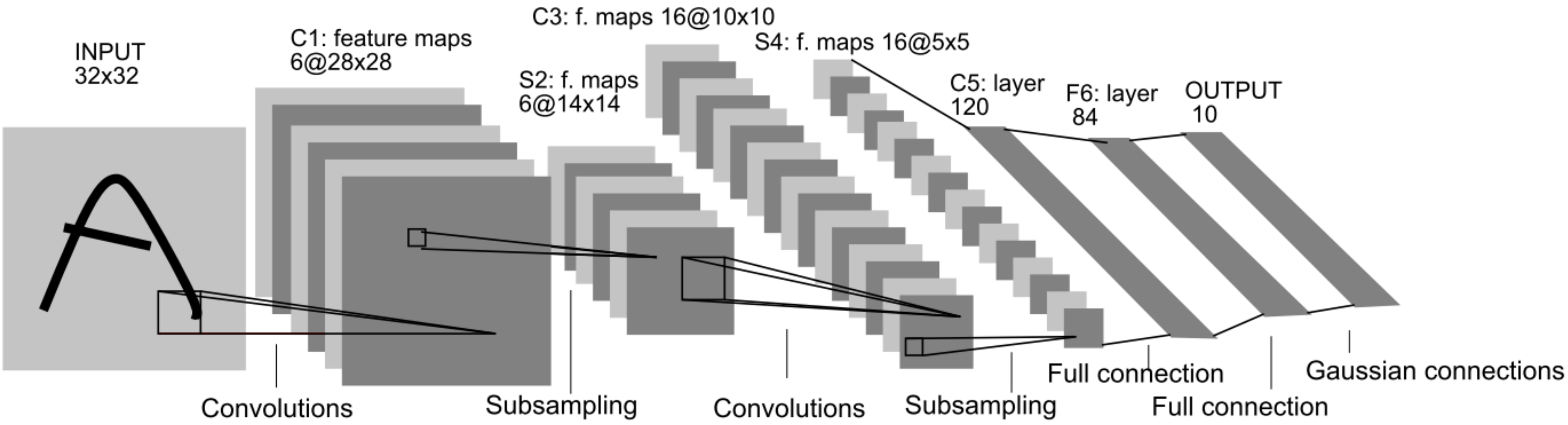
Convolved Feature

LeNet5



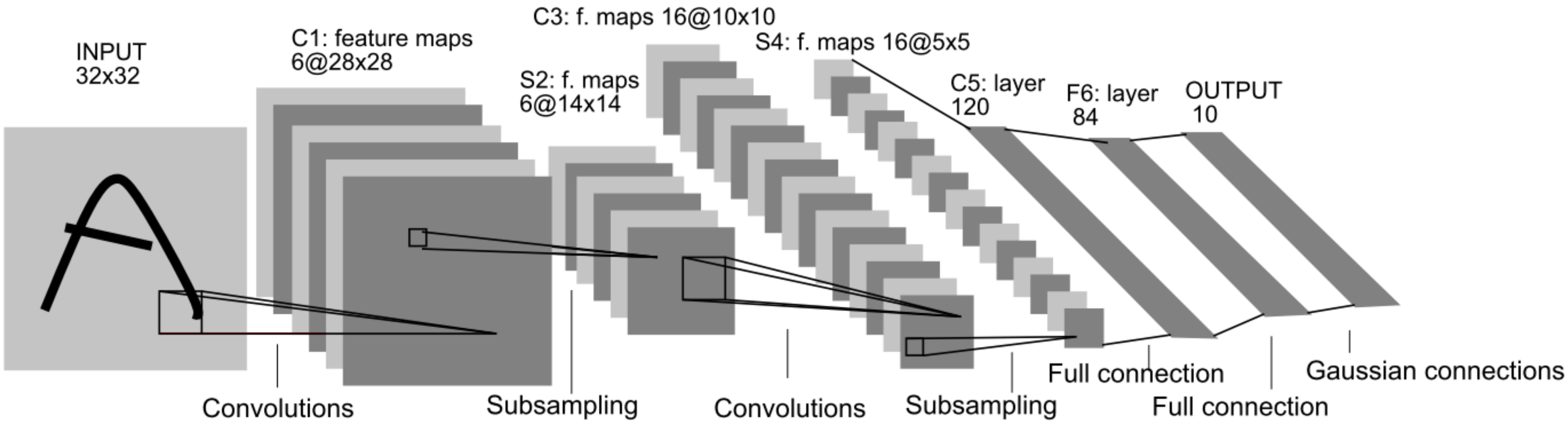
- Input: 32x32 pixel image
- Cx: Convolutional layer
- Sx: Subsample layer
  (reduces image size by averaging 2x2 patches)
- Fx: Fully connected layer

LeNet 5: Layer C1



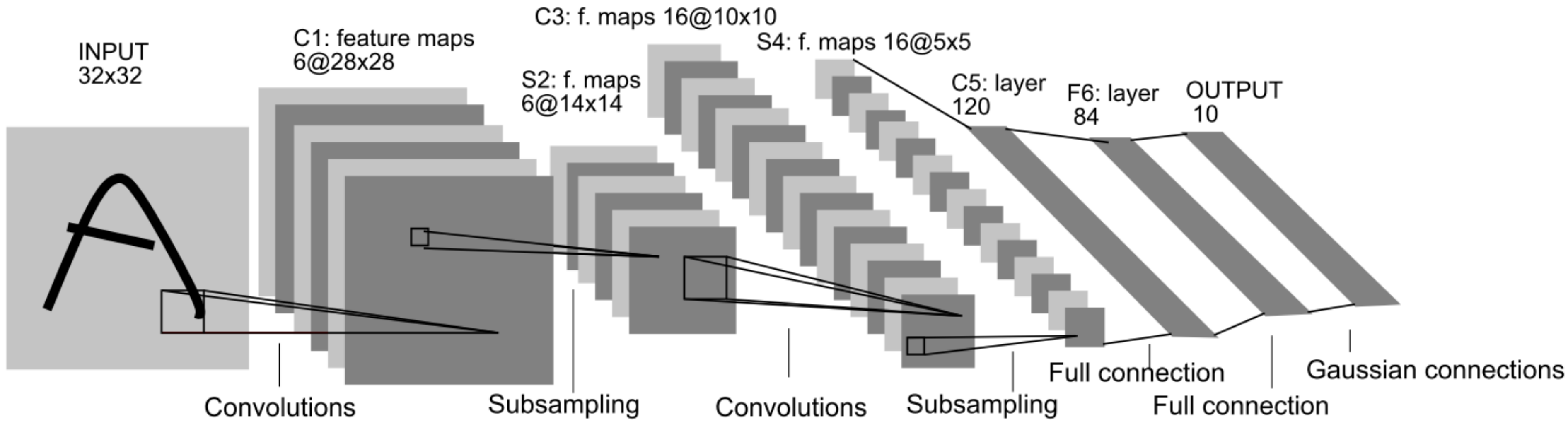
- C1: Convolutional layer with 6 feature maps of size 28x28.
- Each unit of C1 has a 5x5 receptive field in the input layer.
- Shared weights (5*5+1)*6=**156 parameters to learn**
  Connections: 28*28*(5*5+1)*6=122304
- If it was fully connected we had:
  (32*32+1)*(28*28)*6 = **4.821.600 parameters**

LeNet 5: Layer S2



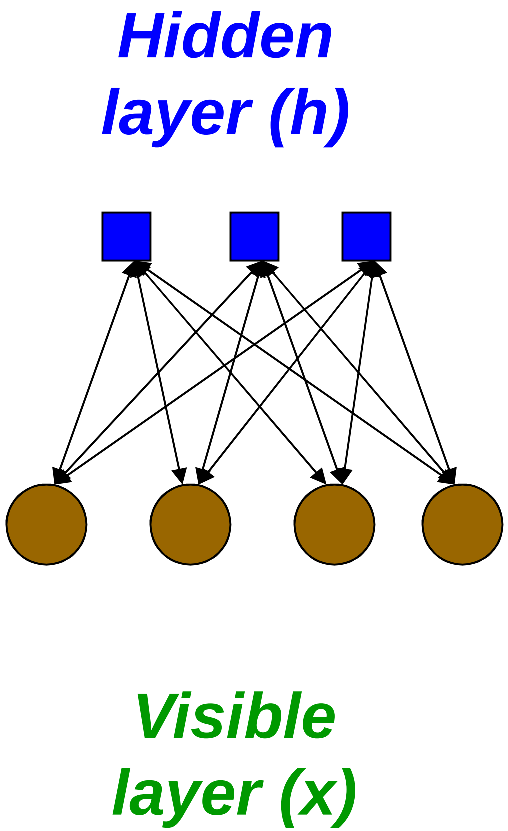
- S2: Subsampling layer with 6 feature maps of size 14x14 2x2 nonoverlapping receptive fields in C1
- Layer S2: 6*2=**12 trainable parameters**.
- Connections: 14*14*(2*2+1)*6=5880

LeNet 5: totals



- The whole network has:
  - 1256 nodes
  - 64.660 connections
  - **9.760 trainable parameters (and not millions!)**

Restricted Boltzmann Machine



- 2-layered network

- $W=(w_{ij})$: matrix of weights
  $b = (b_i)$: biases of visible layer
  $c = (c_j)$: biases of hidden layer

- nodes take values 0 or 1
- **model of probability distr. of $P(x, h)$**
- non-deterministic behavior:

$$P(h_i = 1|x) = \mathrm{sigm}(c_i + W_i x)$$
$$P(x_i = 1|h) = \mathrm{sigm}(b_i + W_i h)$$

RBM as a model of P(x, h)

P(x, y) is defined as:

$$P(x,h) = \frac{e^{-E(x,h)}}{Z}$$

where energy E and normalization Z are given by:

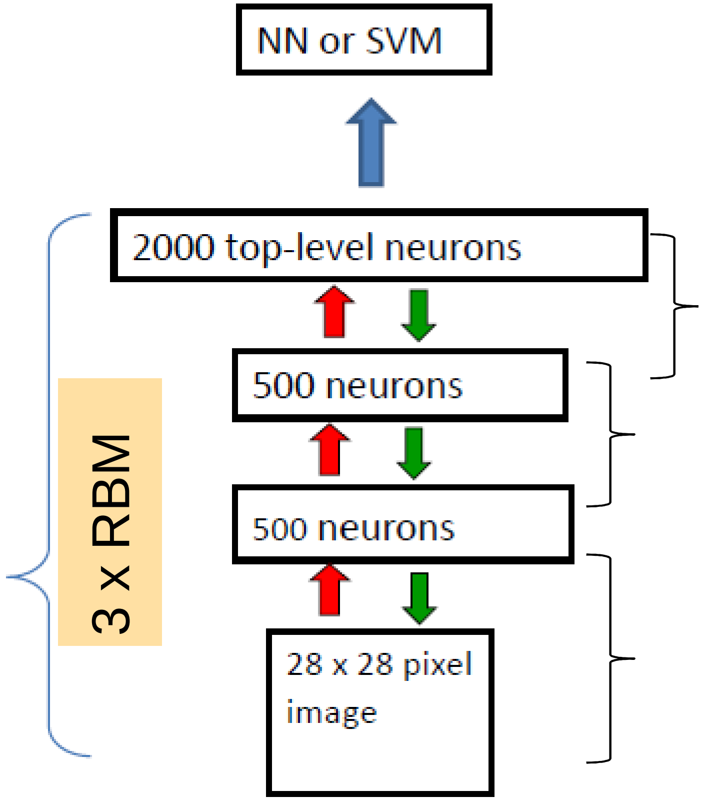
$$E(x,h) = -b'x - c'h - h'Wx$$
$$Z(x,h) = \sum_{x,h} e^{-E(x,h)}$$

Key properties of RBMs

- "unsupervised learning" (no target labels needed)
- Hidden layer learns features that contain information needed to reconstruct the input vectors (feature extractor)
- RBMs can be stacked to form deep networks, with layers trained one after another; they learn more and more complex features (hierarchy of features)
- RBMs are building blocks for deep belief networks, and deep autoencoders

Deep Belief Networks

- the first 3 layers are RBMs
- they are trained one after another with Contrastive Divergence (unsupervised)
- next, an additional output layer is added and the whole network is trained with backpropagation
- Alternatively, instead of adding the final layer one can add a shallow MLP or a conventional classifier (e.g., SVM) and train it with conventional (supervised) methods

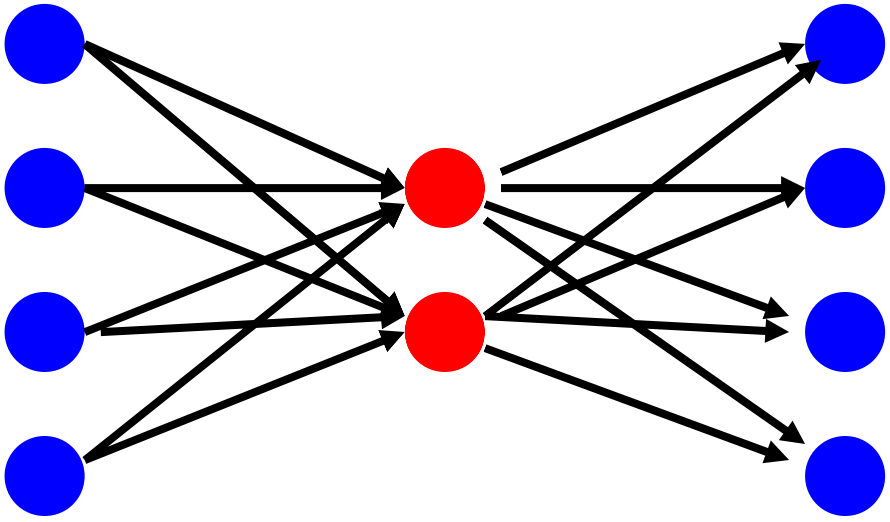


More…

- Why DBN is better than MLP?
  - in MLP “delta’s” vanish (or explode) from layer to layer
  - RBMs “discover” intrinsic features of data
  - RBMs “pre-train” MLP to avoid local minima!
  - RBM trained on unlabelled data!

Deep AutoEncoders

- Consider a 4:2:4 MLP (ENCODER):



- Inputs: (0 0 0 1), (0 0 1 0), (0 1 0 0), (1 0 0 0)
- Outputs: (0 0 0 1), (0 0 1 0), (0 1 0 0), (1 0 0 0)

Encoders

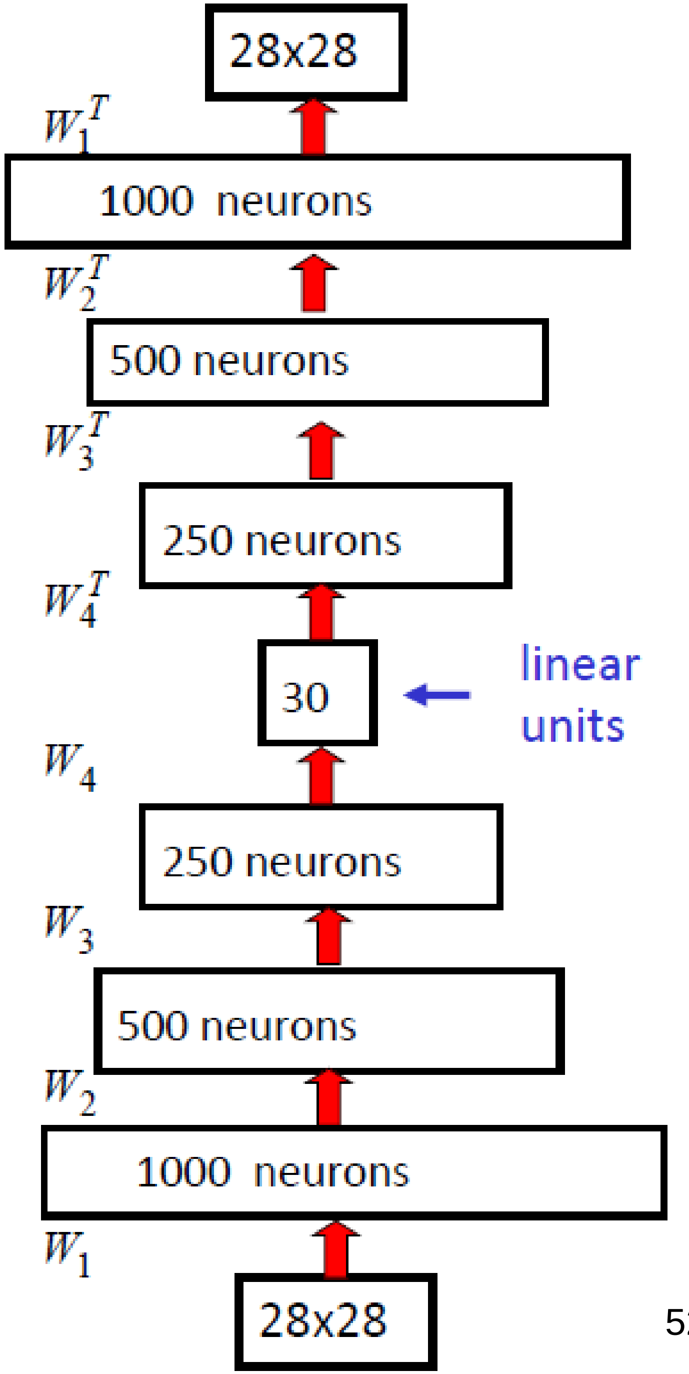
- Encoder network "learns" the concept of binary representation of integers!
- General case: $2^n:n:2^n$ networks
- Somehow the network is able to "extract" the most concise representation of the input!
- The same trick works for "real numbers": PCA networks
- It can be used for dimensionality reduction, data compression, and visualization.

Training Encoders: variants

- Backpropagation (good for shallow architectures)
- Enforced "symmetry of weights"
(upper part = mirror of lower part)
- **Denoising Autoencoders**: the input layers gets a "noisy version of true x"; the target contains the original x ("noise": e.g., 10% of randomly selected pixels set to 0)
- **Deep Autoencoders**: the first "half" is trained unsupervised as a DBN (a stack of RBMs); the upper part is the "mirror" of the lower part; final phase trained by backpropagation

Deep Autoencoders

- They always looked like a really nice way to do non-linear dimensionality reduction:
  - But it is very difficult to optimize deep autoencoders using backpropagation.
- We now have a much better way to optimize them:
  - First train a stack of 4 RBM's
  - Then "unroll" them.
  - Then fine-tune with backprop.

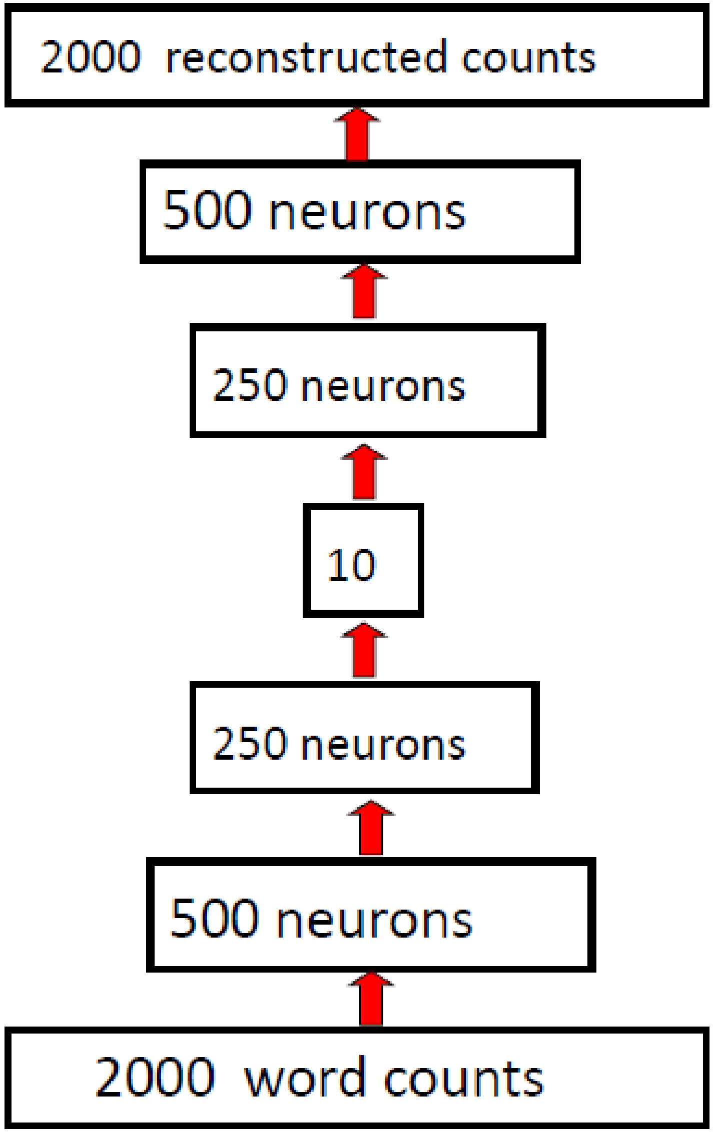


Hinton & Salakhutdinov, 2006; slide form Hinton UCL tutorial

Applications: Classifying text documents

- A document can be characterized by the frequency of words that appear (ie, word counts for some dictionary become feature vector)
- Goals…
  1. Group/cluster similar documents
  2. Find similar documents

How to compress the count vector

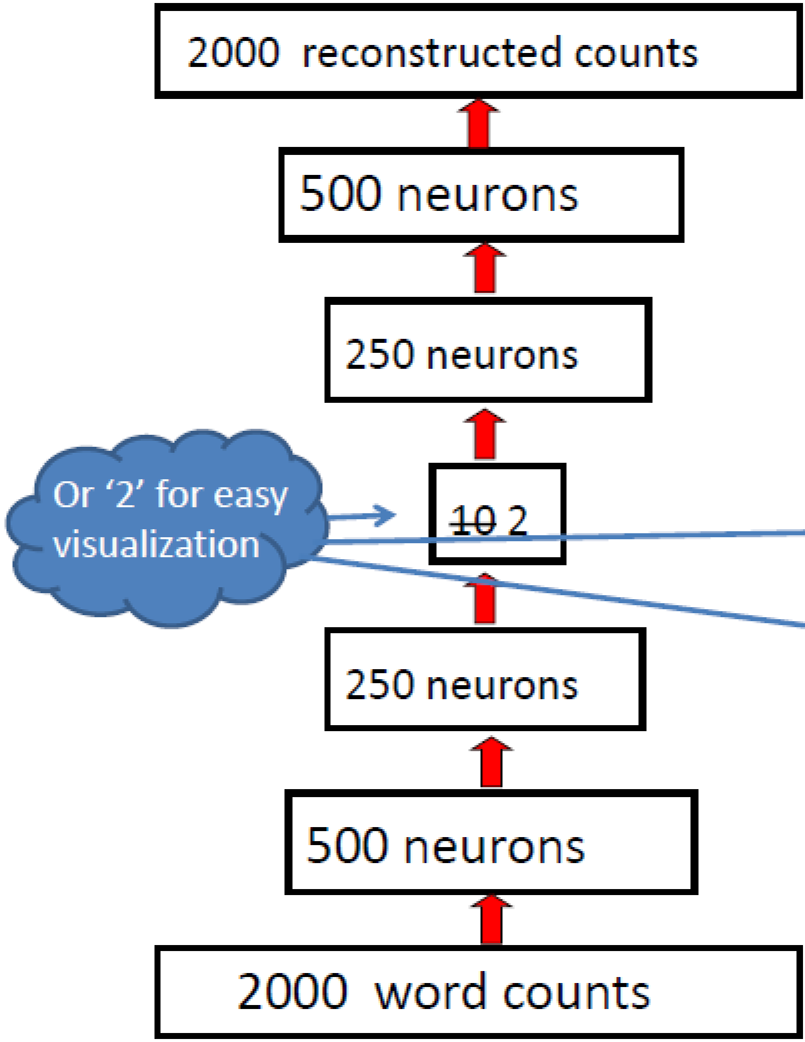


Multi-layer auto-encoder

- Train a model to reproduce its input vector as its output
- This setup forces as much information as possible be compressed and passed thru the 10 numbers in the central bottleneck.
- These 10 numbers are then a good way to compare documents.

Slide modified from Hinton, 2007

How to compress the count vector

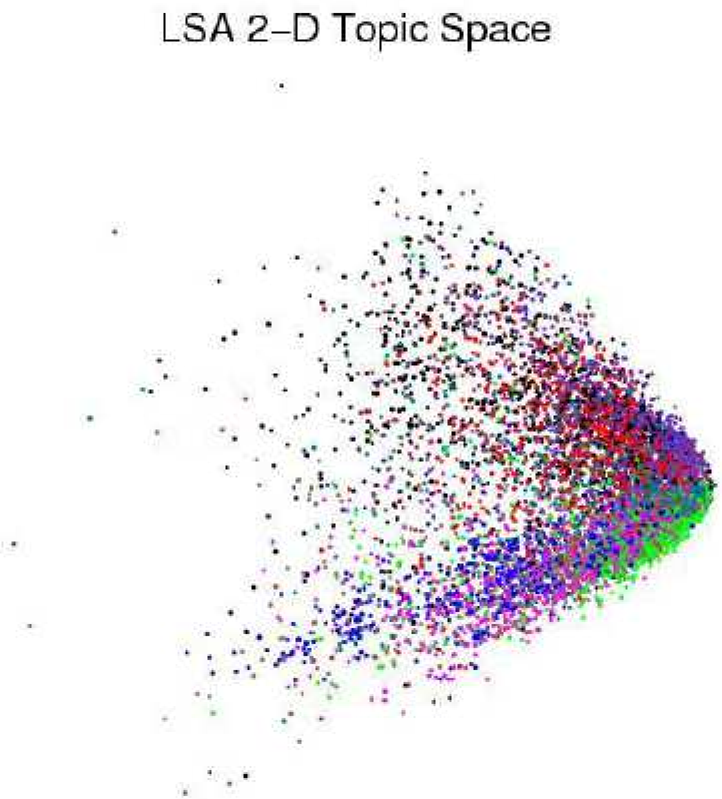


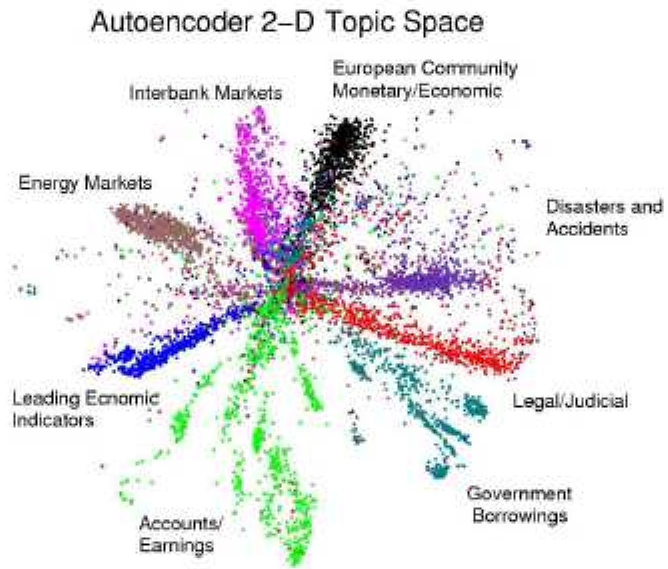
<u>Multi-layer auto-encoder</u>

- Train a model to reproduce its input vector as its output
- This setup forces as much information as possible be compressed and passed thru the ~~10~~ 2 numbers in the central bottleneck.
- These ~~10~~ 2 numbers are then a good way to compare documents.

Slide modified from Hinton, 2007

Clusters

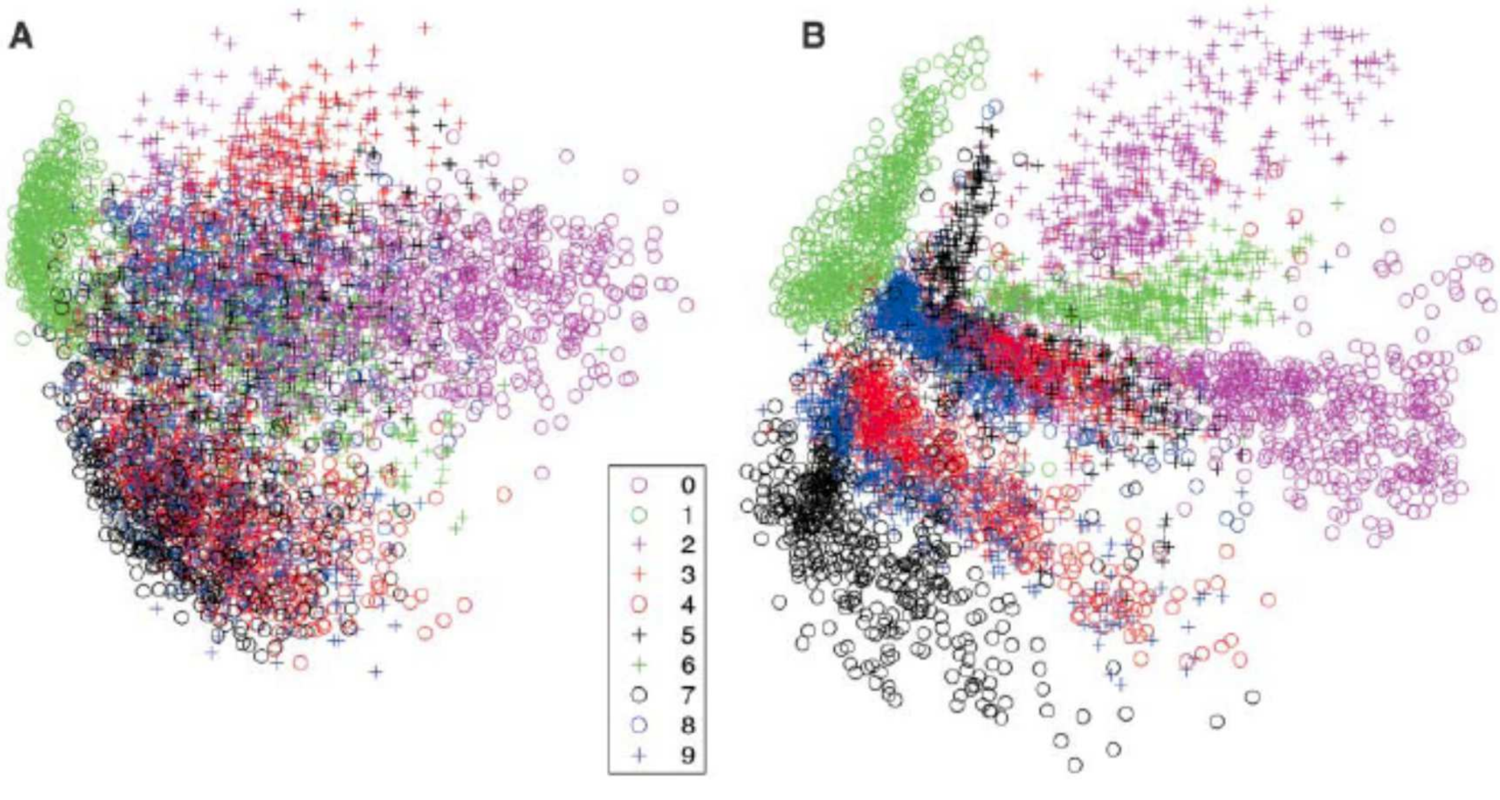




Images from Hinton, 2007

http://www.cs.toronto.edu/~rsalakhu/papers/science.pdf

Fig. 3. (**A**) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (**B**) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (8).



The same trick applied to "digits"

RBM networks and Netflix Challenge

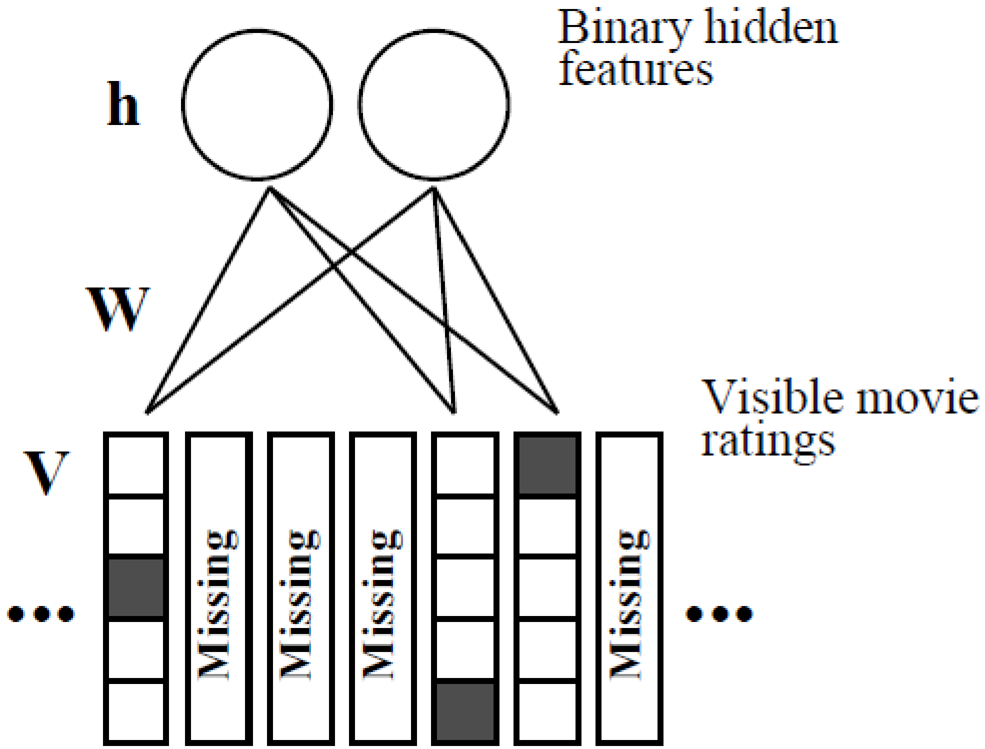
- **100.000.000** rating **records** collected over 1997-2005
- **rating record:**
  <customer_id, movie_id, rating>
- **500.000 customers**
- **18.000 movies**
- **rating** = an integer: **1, 2, 3, 4 or 5**
- Additionally, **3.000.000 test records**:
  <customer_id, movie_id, ? >

**GOAL: fill in "?'s" with numbers,
so the error is minimized!**

$1.000.000 prize for improving Netflix system by 10%

Restricted Boltzmann Machines for Collaborative Filtering

Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
Andriy Mnih AMNIH@CS.TORONTO.EDU
Geoffrey Hinton HINTON@CS.TORONTO.EDU
University of Toronto, 6 King's College Rd., Toronto, Ontario M5S 3G4, Canada



output layer:
e.g., 100 nodes

input layer:
for each movie 5 nodes

Google “zero-shot-translation”

https://research.googleblog.com/2016/11/zero-shot-translation-with-googles.html