GENERAL ASSEMBLY

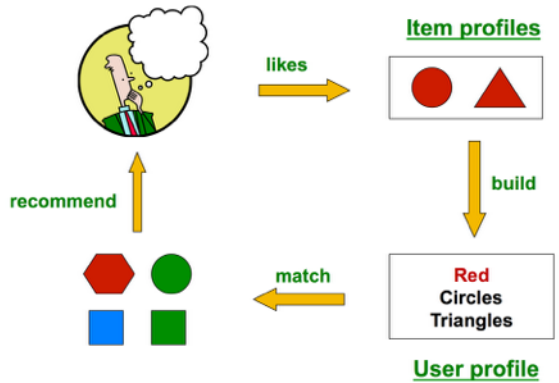
INTRO TO DATA SCIENCE

NEURAL NETWORKS AND DEEP LEARNING

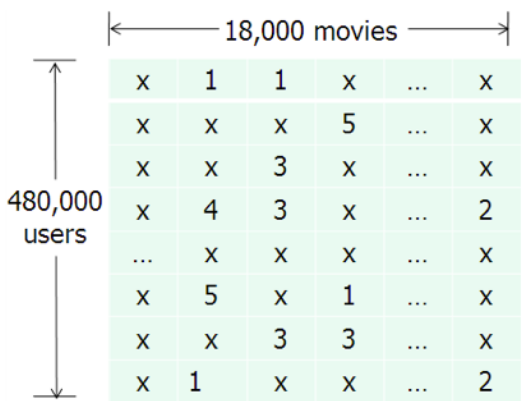
DATA SCIENCE IN THE NEWS

LAST TIME:

I. OVERVIEW
II. CONTENT-BASED FILTERING
III. COLLABORATIVE FILTERING
IV. THE NETFLIX PRIZE



J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org



QUESTIONS?

WHAT WAS THE MOST INTERESTING THING YOU LEARNT?

WHAT WAS THE HARDEST TO GRASP?

I. OVERVIEW
II. ARTIFICIAL NEURAL NETWORKS
III. BACKPROPAGATION
IV. AUTOENCODERS
V. APPLICATIONS OF DEEP LEARNING

VI. PYTHON LAB

- UNDERSTAND WHAT IS A NEURAL NET
- BE ABLE TO EXPLAIN WHY THEY ARE SO POPULAR
- KNOW HOW THE BACK PROPAGATION ALGORITHM WORKS
- KNOW THE ADVANTAGES OF AUTO ENCODERS
- BE ABLE TO APPLY AUTO ENCODERS IN PYTHON
- BE ABLE TO TRAIN AND RUN SIMPLE NN IN SCIKIT-LEARN
- KNOW THE MAIN PACKAGES THAT IMPLEMENT DEEP LEARNING

OVERVIEW

DISCLAIMER:

Deep Learning is a huge field currently in rapid evolution.

Today we scratch the surface …

Q: What is deep learning?

Q: What is deep learning?

A:

- It's (just) a Machine Learning technique
- It's about learning features and representations
- It's based on Artificial Neural Networks

MACHINE LEARNING

so far we've seen examples of **supervised** and **unsupervised** learning:

- Predicting the likelihood of a customer to churn
- Clustering customers by their purchase habits
- …

MACHINE LEARNING

so far we've seen examples of **supervised** and **unsupervised** learning:

- Predicting the likelihood of a customer to churn
- Clustering customers by their purchase habits
- …

There are other tasks that fall into the domain of ML like:

- Object recognition
- Image classification
- Speech to text transcription
- Sentiment analysis
- …

MACHINE LEARNING : TWO STEPS

1) Feature extraction: learning how to represent our data
2) Model Building: learning how to predict new quantities

MACHINE LEARNING : TWO STEPS

1) Feature extraction: learning how to represent our data
2) Model Building: learning how to predict new quantities

Which one is more time consuming?

MACHINE LEARNING : TWO STEPS

1) Feature extraction: learning how to represent our data
2) Model Building: learning how to predict new quantities

Which one is more time consuming? YES, nr. 2!

DEEP LEARNING comes to help on both

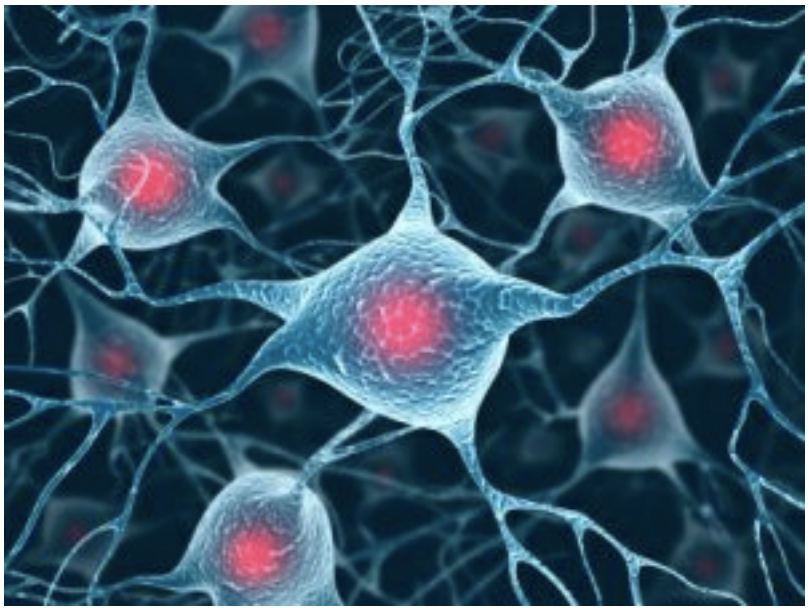
ARTIFICIAL NEURAL NETWORKS

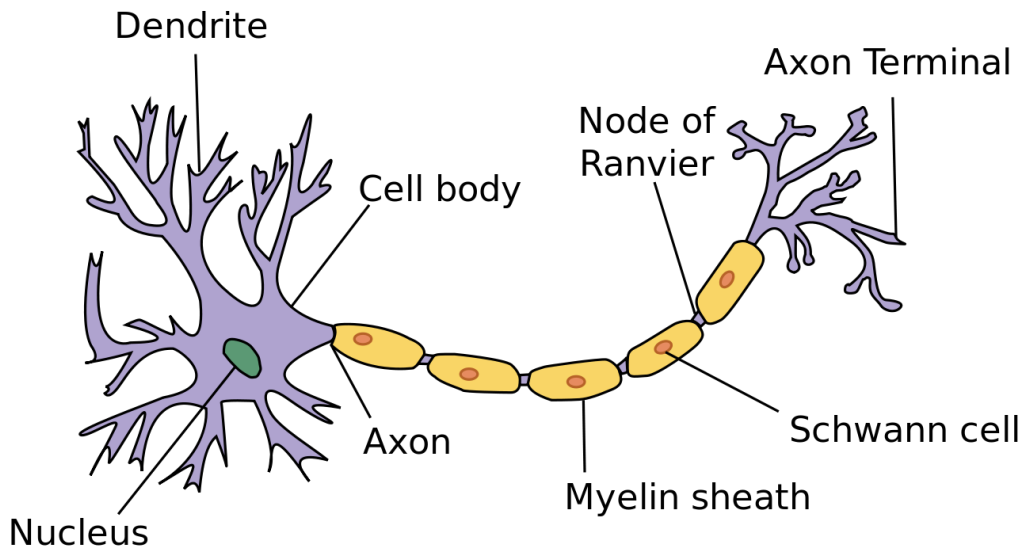
Q: What is an Artificial Neural Network?

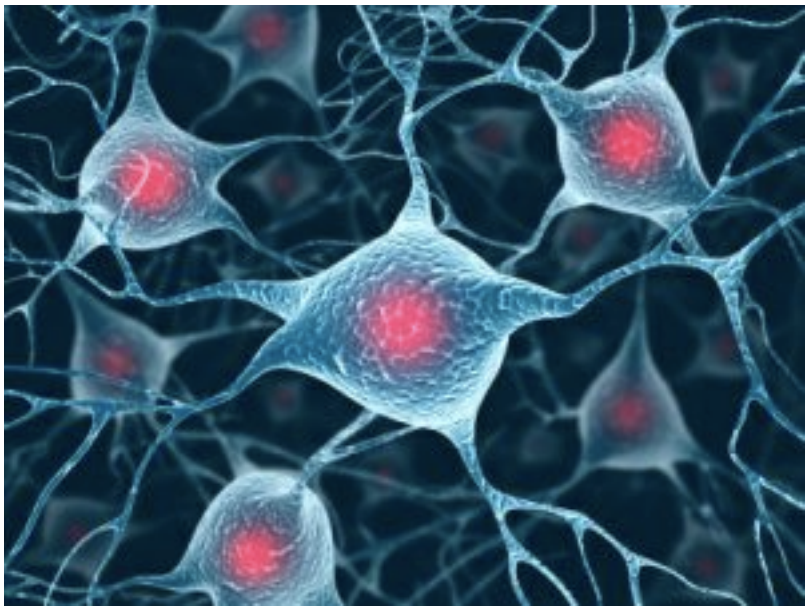
Q: What is an Artificial Neural Network?

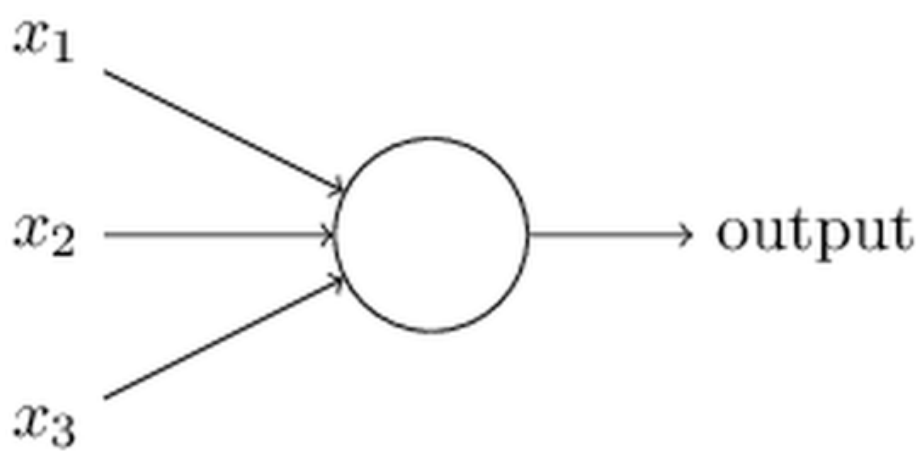
A: A computational system comprised of layers and each layer is built of interconnected perceptrons

=> Built to model the animal nervous system

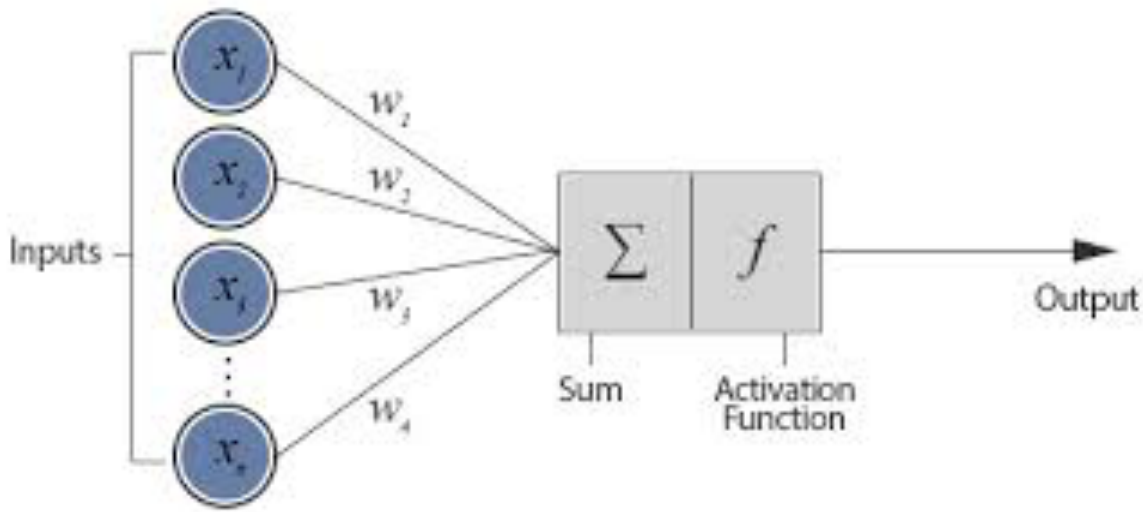



Dendrite
Axon Terminal
Node of
Ranvier
Cell body
Schwann cell
Axon
Myelin sheath
Nucleus




x_1
x_2
x_3
output

Single Perceptron



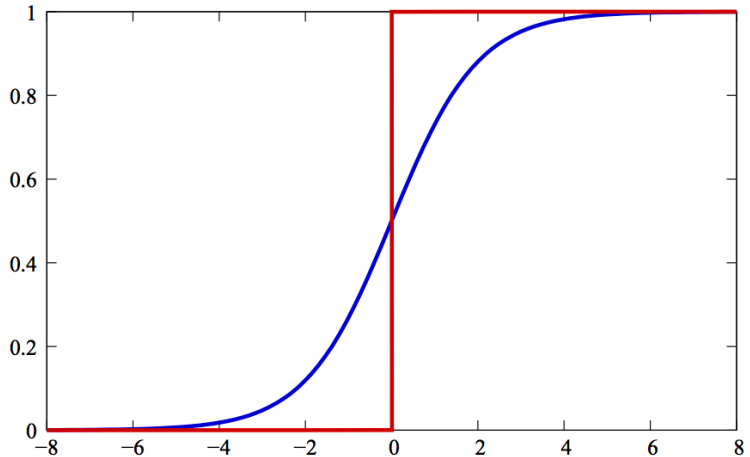
Takes in input and uses an activation function in order to calculate output

Single Perceptron

NOTE:
A single perception can be like a logistic regression in and of itself!

$$f_{log}(z) = \frac{1}{1 + e^{-z}}$$

f_{log} is called logistic function



Takes in input and uses an activation function in order to calculate output

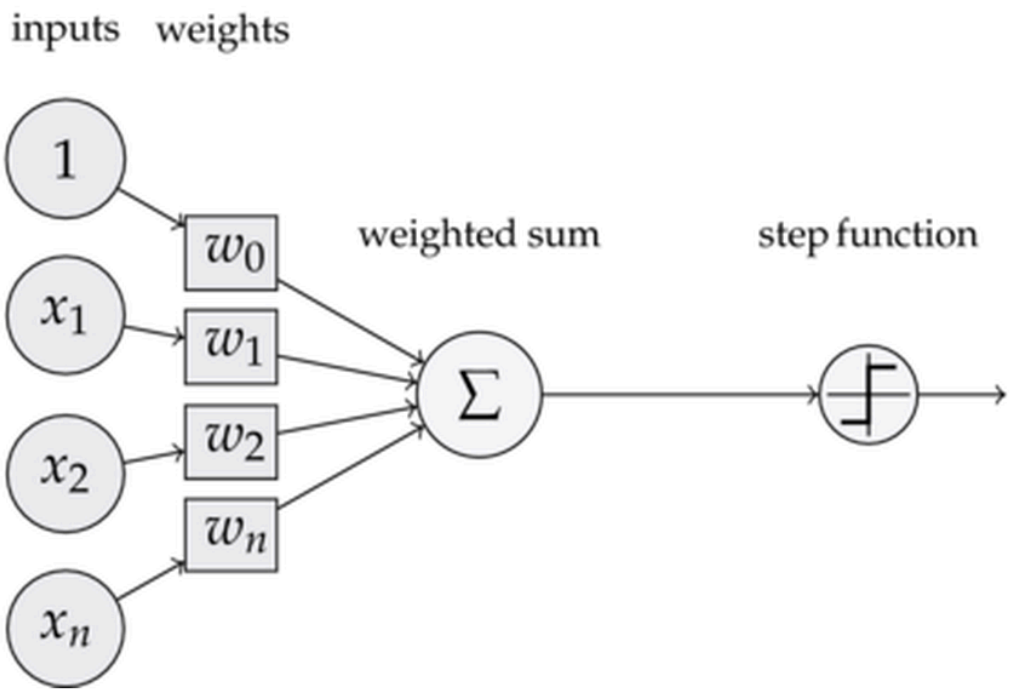
Single Perceptron

But what is z? A weighted sum on the inputs!

$$z = \sum_{i=0}^{n} w_i * x_i$$

Where w is the weight on input x

Single Perceptron

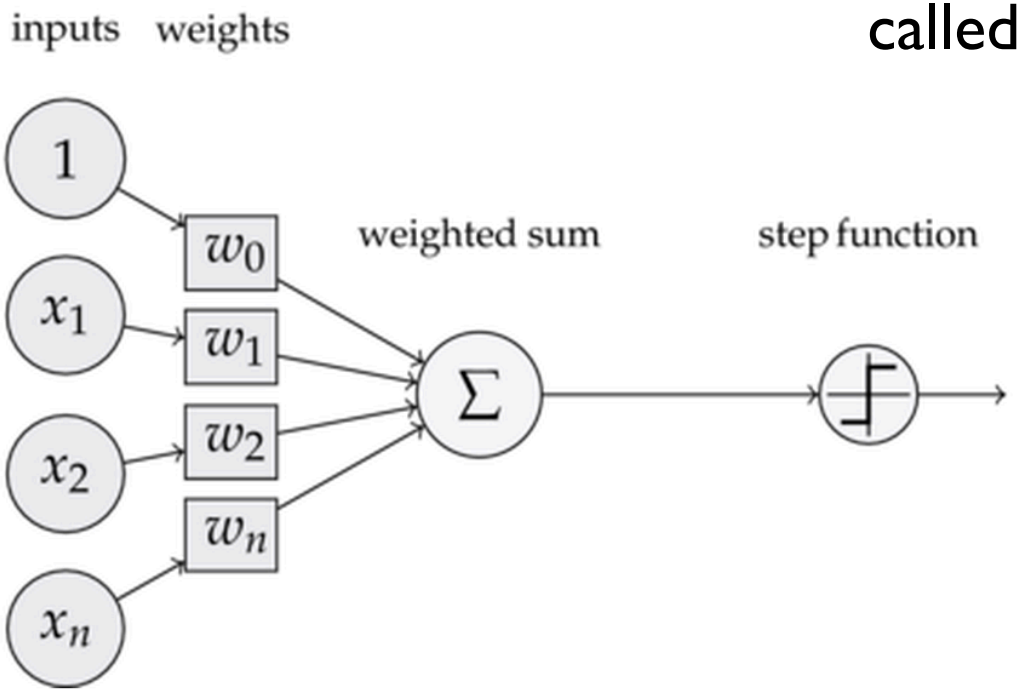


$$f_{log}(z) = \frac{1}{1 + e^{-z}}$$

f_{log} is called logistic function

Single Perceptron

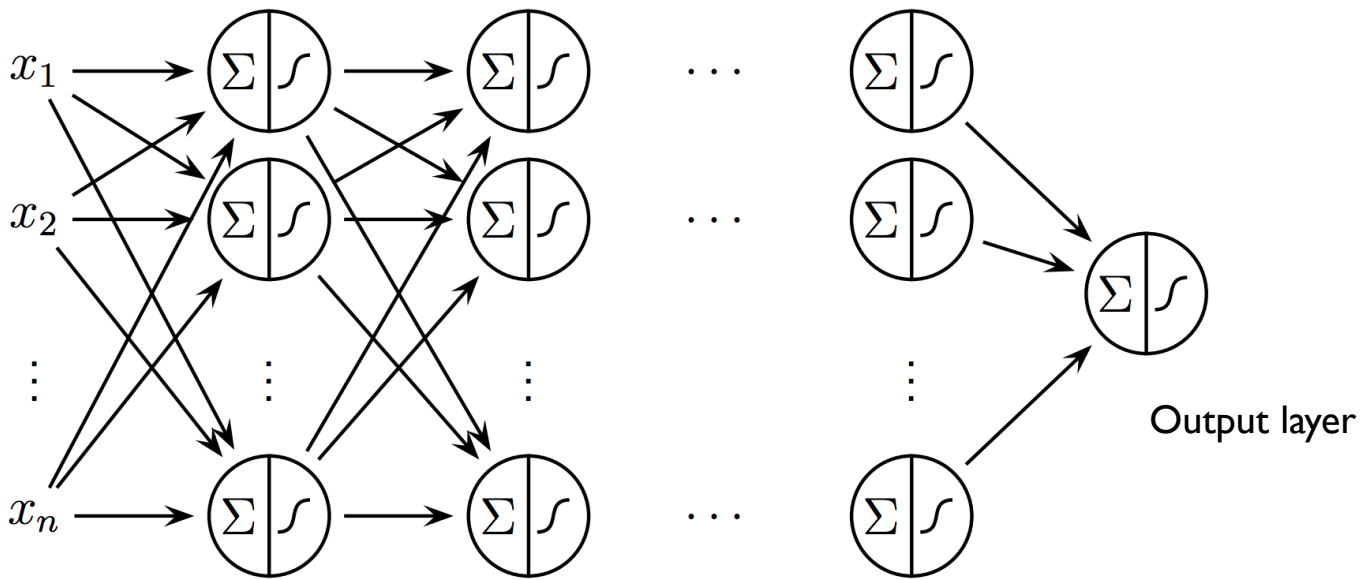
If f(z) if above a threshold, generally called theta, then the neuron "fires"



$$f_{log}(z) = \frac{1}{1 + e^{-z}}$$

f_{log} is called logistic function

A multi layer perceptrons (MLP) is a finite acyclic graph. The nodes are neurons with logistic activation.



Q: But how does it learn?!

Q: But how does it learn?!

A: Backpropagation!!

BACKPROPAGATION

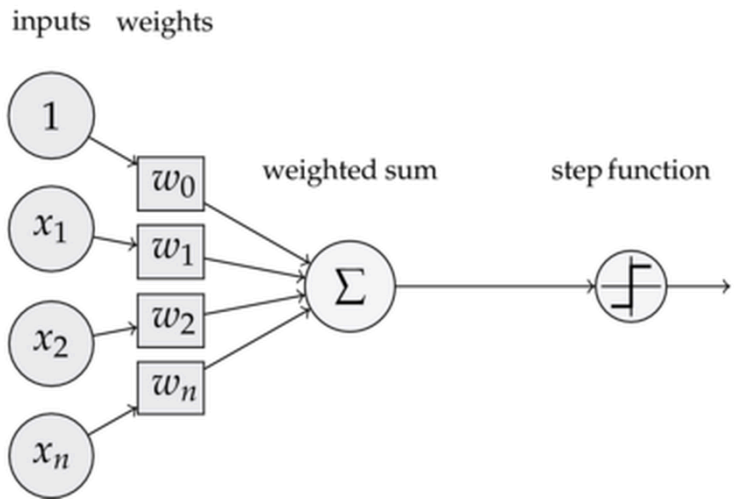
The main problem for the neural network is to learn how to adjust the synaptic weights.

The main problem for the neural network is to learn how to adjust the synaptic weights.

We can do that by looking at the error of the value predicted by a node

Activation at node j:

$net_j = \sum_i (o_i \times w_{ij})$



Output at node j:

$o_j = f(net_j)$ where $f(net_j) = 1 / (1 + e^{-netj})$

Can calculate the first derivative of the transfer function:

$$f'(net_j) = f(net_j) \times (1.0 - f(net_j))$$

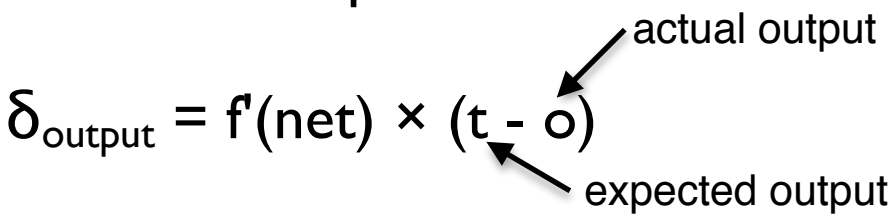
Can calculate the first derivative of the transfer function:

$f'(net_j) = f(net_j) \times (1.0 - f(net_j))$

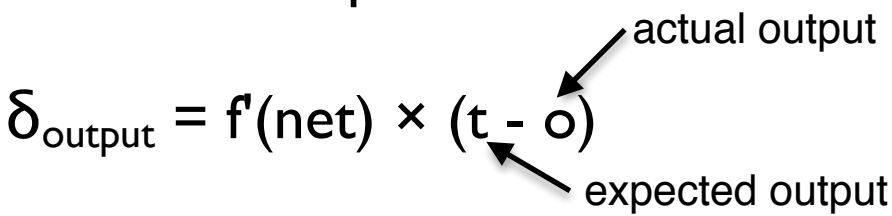
and from that we can calculate the deltas (δ):

$\delta = f'(net) \times (\text{received error})$

deltas for output node is:



deltas for output node is:



delta for hidden node is:

$$\delta_j = f'(net_j) \times \sum_k (\delta_k \times w_{jk})$$

deltas for output node is:

$$\delta_{output} = f'(net) \times (t - o)$$

actual output

expected output

delta for hidden node is:

$$\delta_j = f'(net_j) \times \sum_k (\delta_k \times w_{jk})$$

And the correction to apply to the weight is:

$$dw_{ij} = L \times o_i \times \delta_j$$

learning parameter

BACKPROPAGATION SUMMARY

- **Input:** Set the activation for each node of the the input layer
- **Feedforward:** For each layer, calculate the output
- **Output error δ:** Compute the loss function at the output layer
- **Backpropagate the error:** propagate the output error back using the gradient
- **Adjust weights:** At each stage apply the correction to the weights

General definition of BACKPROPAGATION:

"procedure that is used to compute gradients of a loss function (e.g. a prediction error, but not necessarily, it could be a negative log-likelihood of a probabilistic model) with respect to parameters"

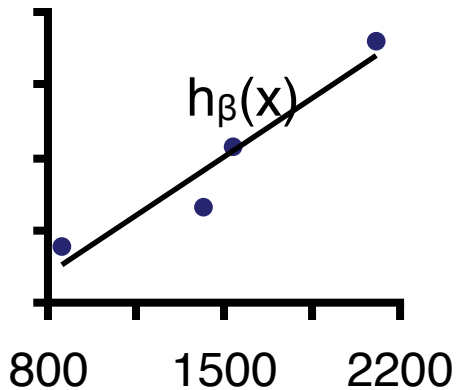
Yoshua Bengio

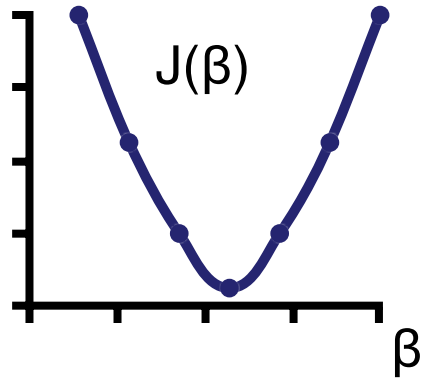
- based on the application of the chain rule
- computationally proceeds 'backwards' with respect to the computations performed to compute the loss itself
- most efficient possible procedure to compute the exact gradient

Remember:

Training set : (x, y)

Hypothesis Function : $h_\beta(x)$





Cost Function: J(β)

GOAL

Find the values for the parameters β that minimize the cost function over the set of training data

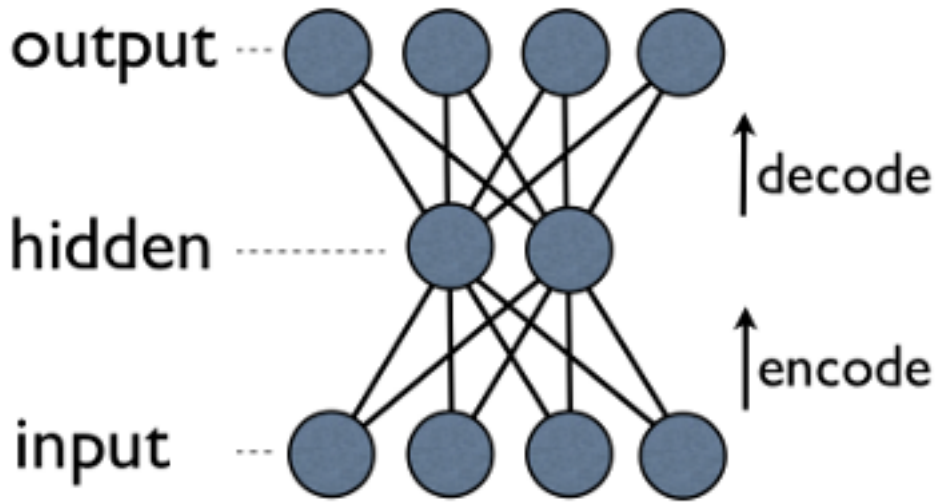
LAB: Neural Network from Scratch

AUTOENCODERS

Autoencoders are a particular class of NN that are very useful for unsupervised feature extraction

Autoencoders are a particular class of NN that are very useful for unsupervised feature extraction.

In Autoencoders the **output** layer has the same size as the **input**



Reconstruction = decoder(encoder(input))

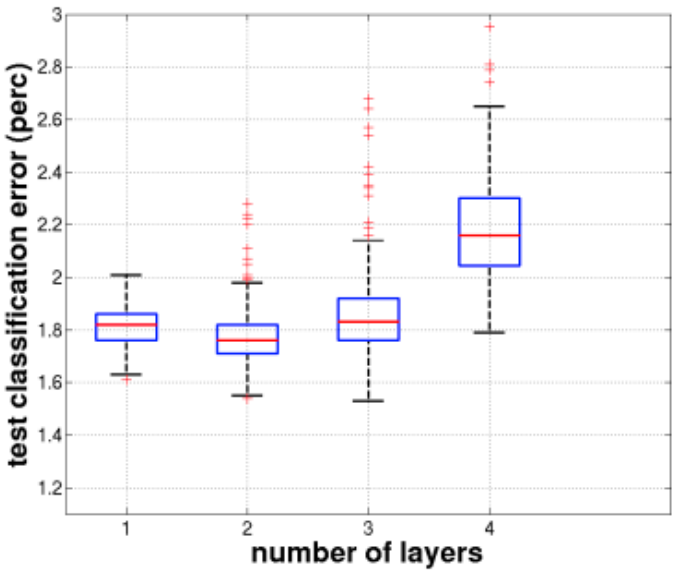
Useful for?

Reconstruction = decoder(encoder(input))

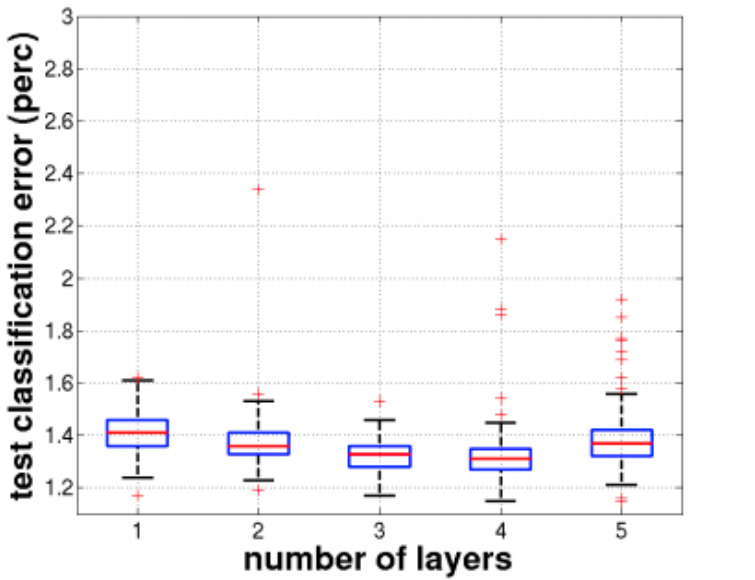
Useful for?

- Dimensionality Reduction
- De-noising

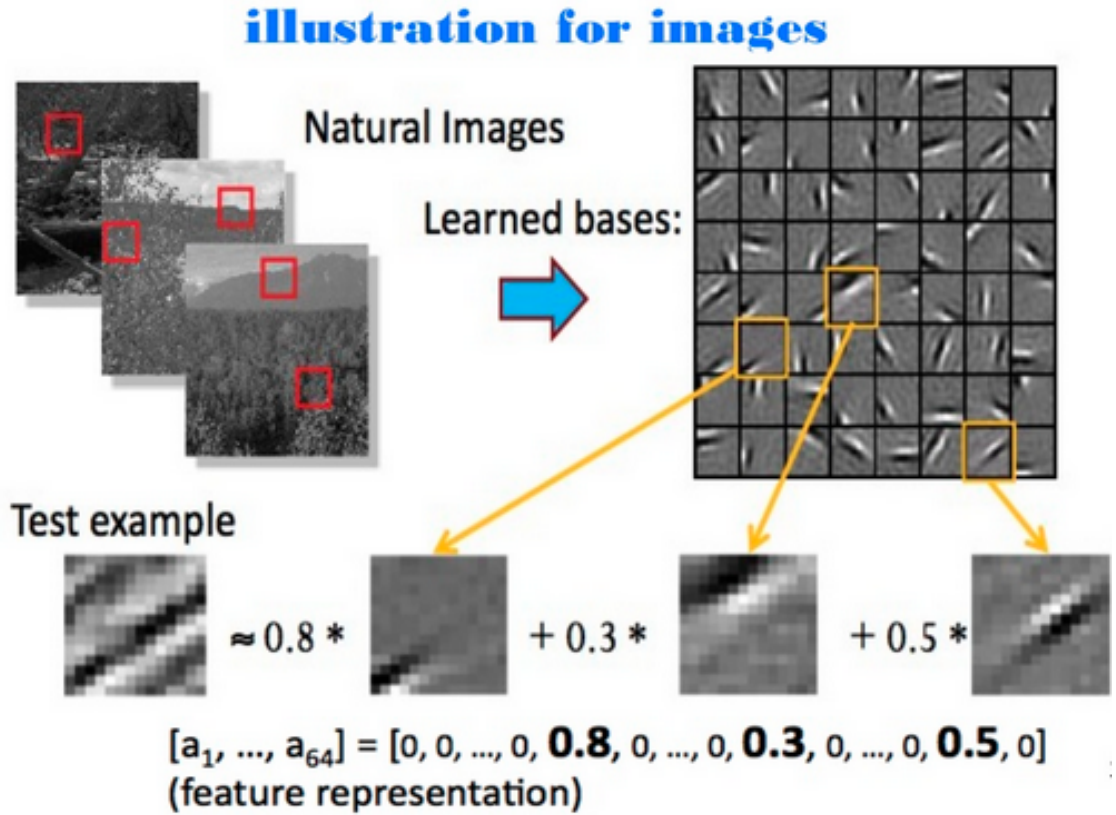
Supervised neural net



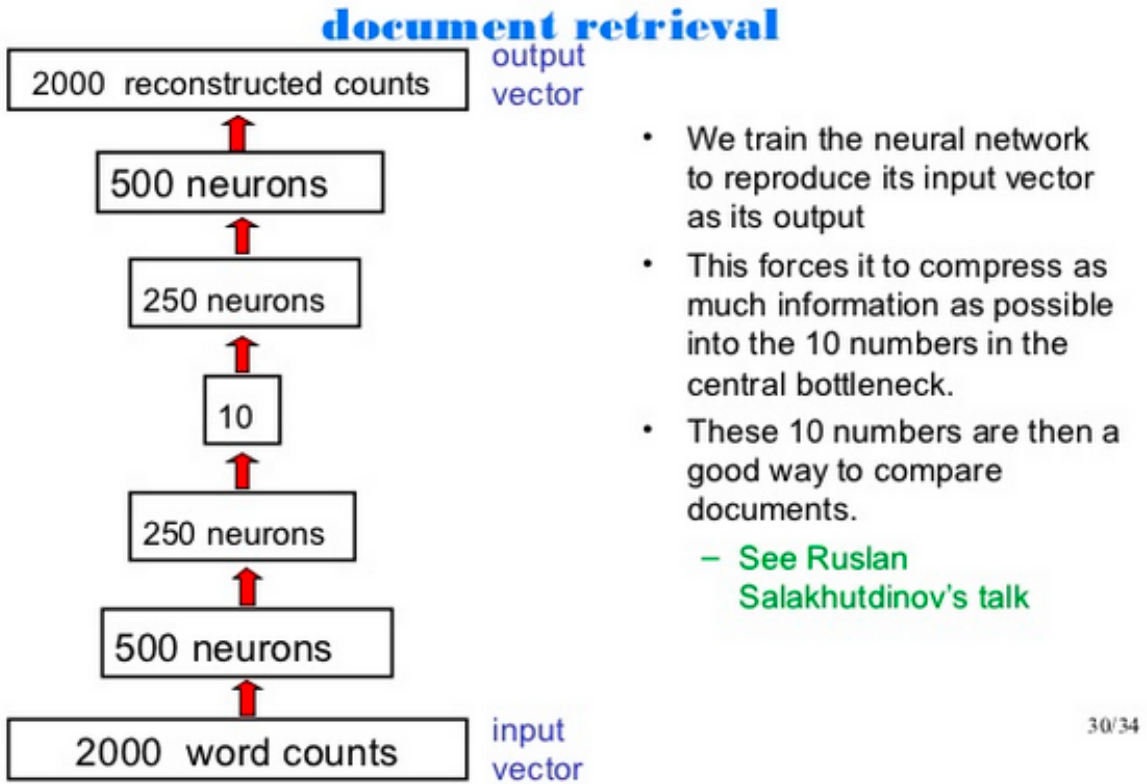
with unsupervised pre-training



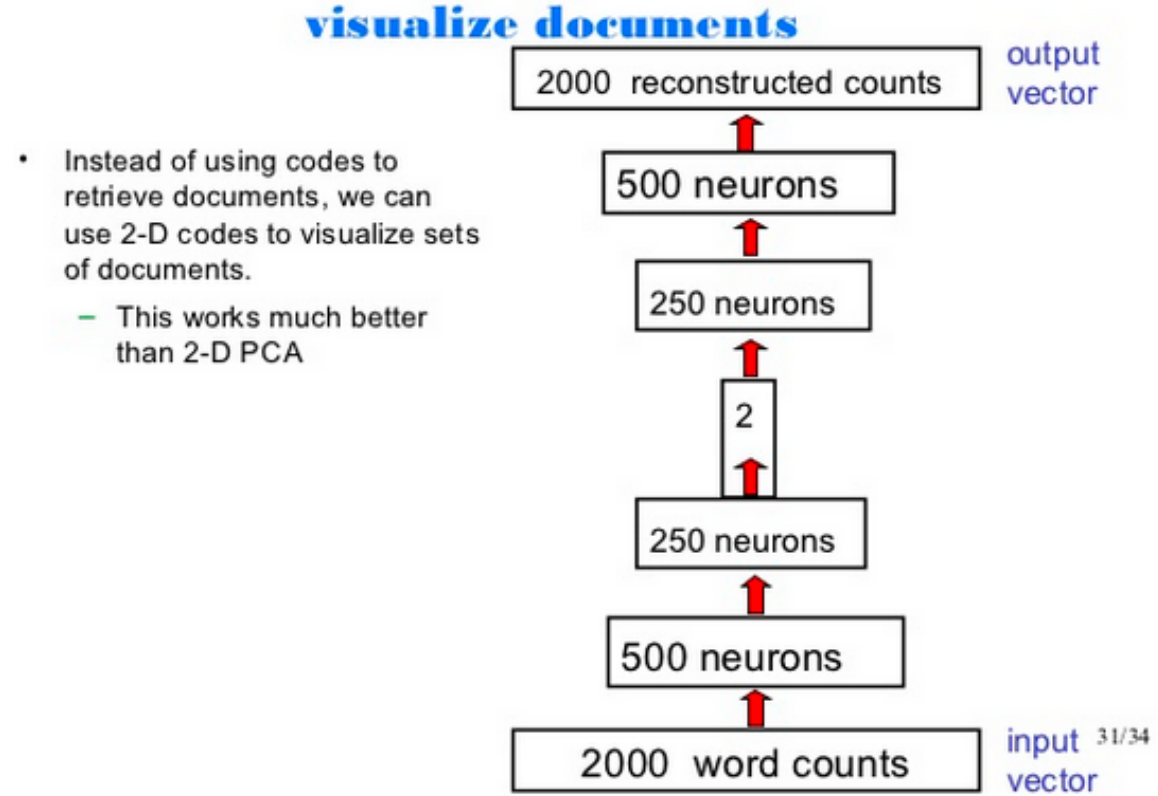
http://jmlr.org/papers/volume11/erhan10a/erhan10a.pdf



http://www.slideshare.net/billlangjun/simple-introduction-to-autoencoder

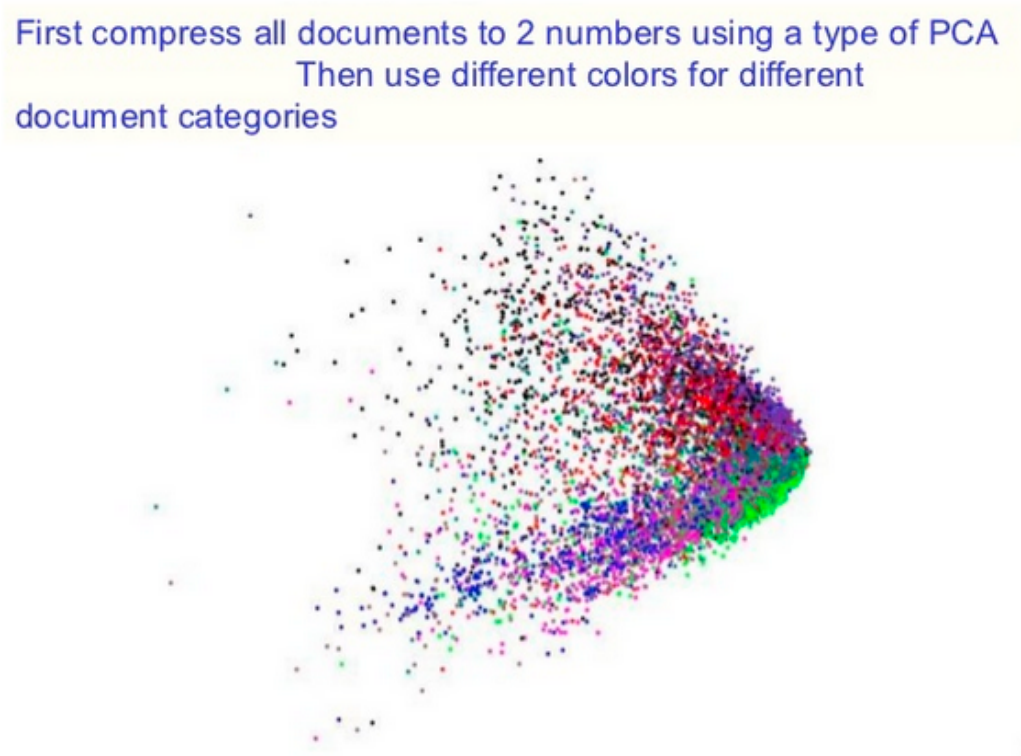


http://www.slideshare.net/billlangjun/simple-introduction-to-autoencoder

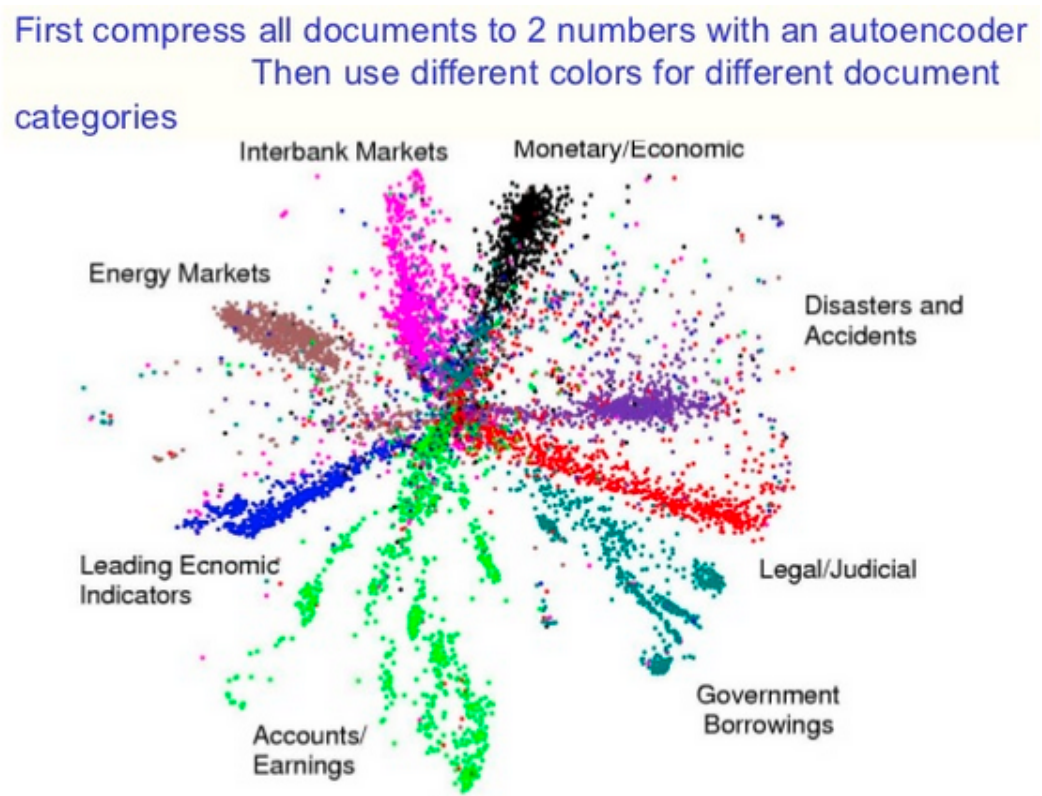


- Instead of using codes to retrieve documents, we can use 2-D codes to visualize sets of documents.
  - This works much better than 2-D PCA

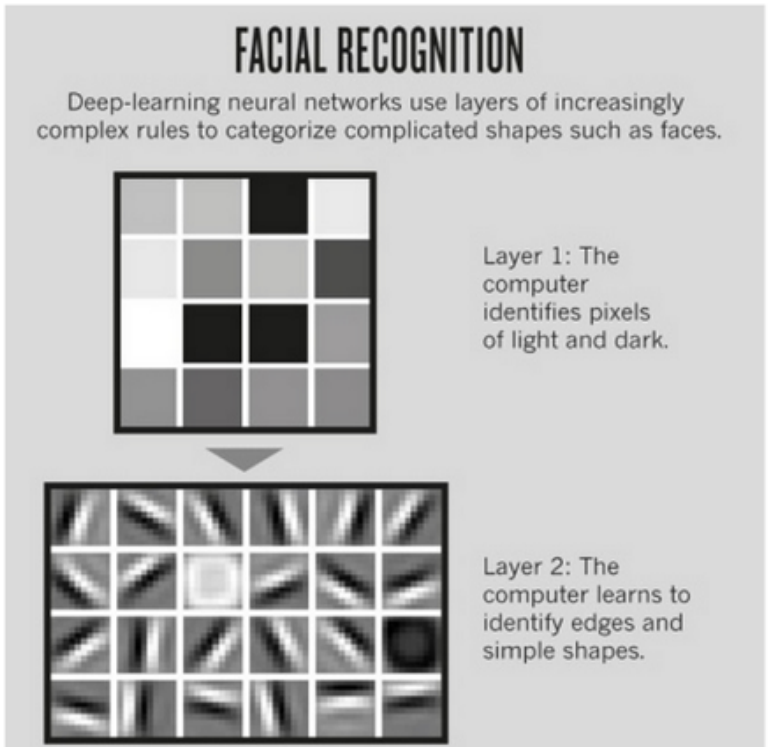
http://www.slideshare.net/billlangjun/simple-introduction-to-autoencoder

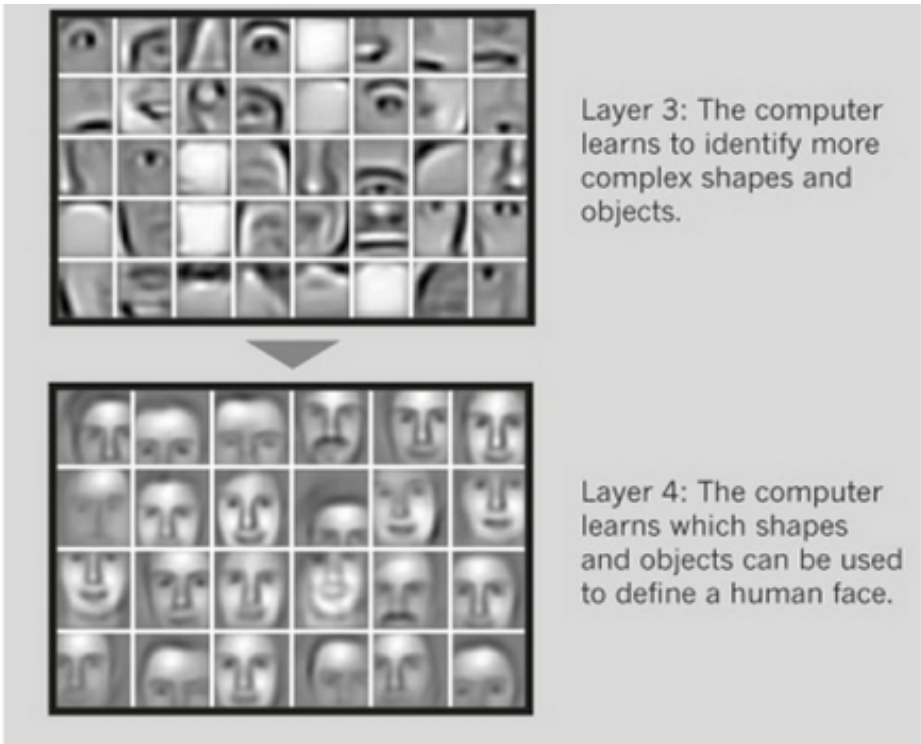


http://www.slideshare.net/billlangjun/simple-introduction-to-autoencoder



http://www.slideshare.net/billlangjun/simple-introduction-to-autoencoder





http://www.nature.com/news/computer-science-the-learning-machines-1.14481

LAB: Autoencoders

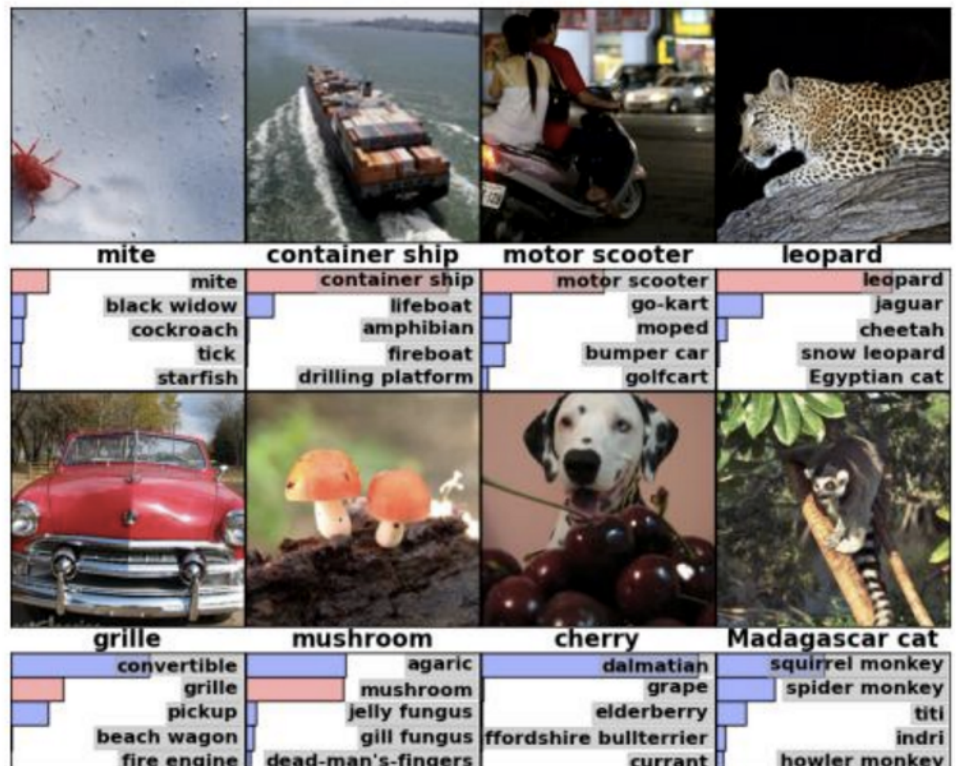
APPLICATIONS OF DEEP LEARNING

Q: why the renaissance of deep learning?

A:

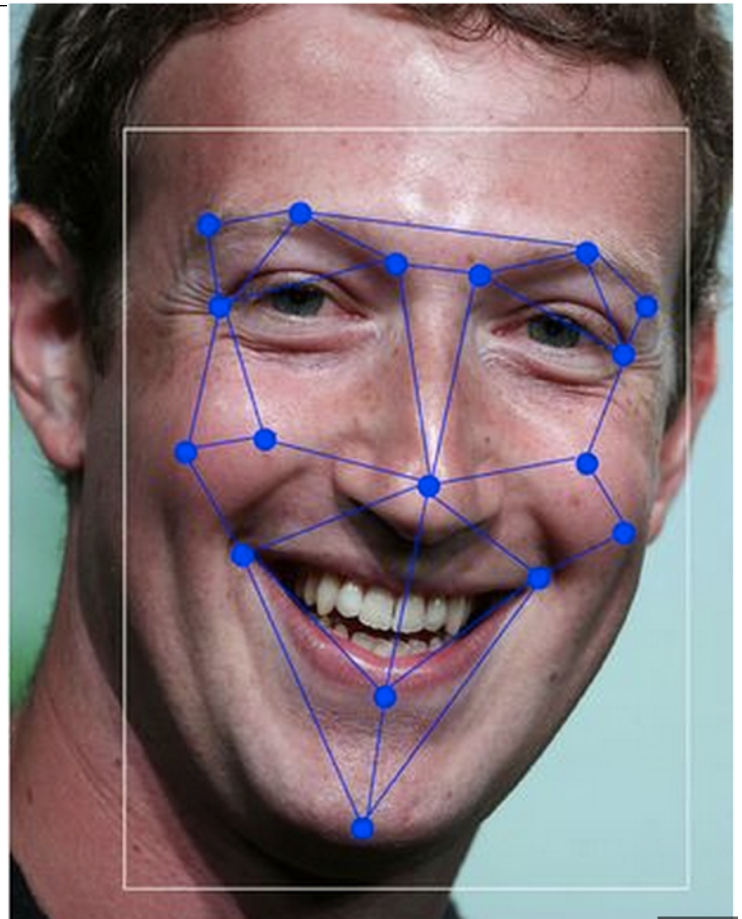
- New techniques for training these complex models (new optimization methods, better initializations …)
- More computational power (GPUs are now used to dramatically speed up the training)
- More data ! (The internet)

Object recognition



http://www.image-net.org/challenges/LSVRC/

Facebook Deep Face



https://research.facebook.com/publications/480567225376225/deepface-closing-the-gap-to-human-level-performance-in-face-verification/

Image captions

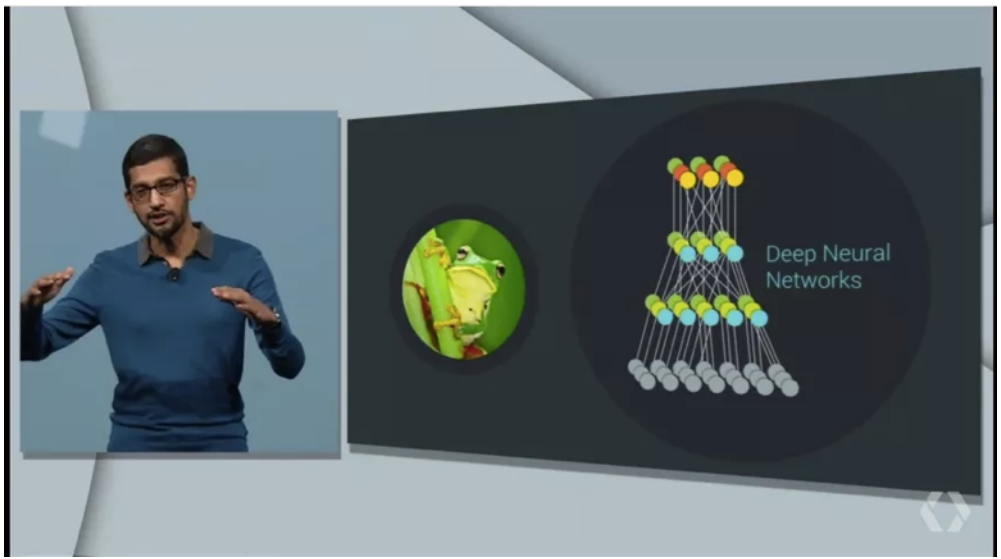
"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

http://cs.stanford.edu/people/karpathy/deepimagesent/

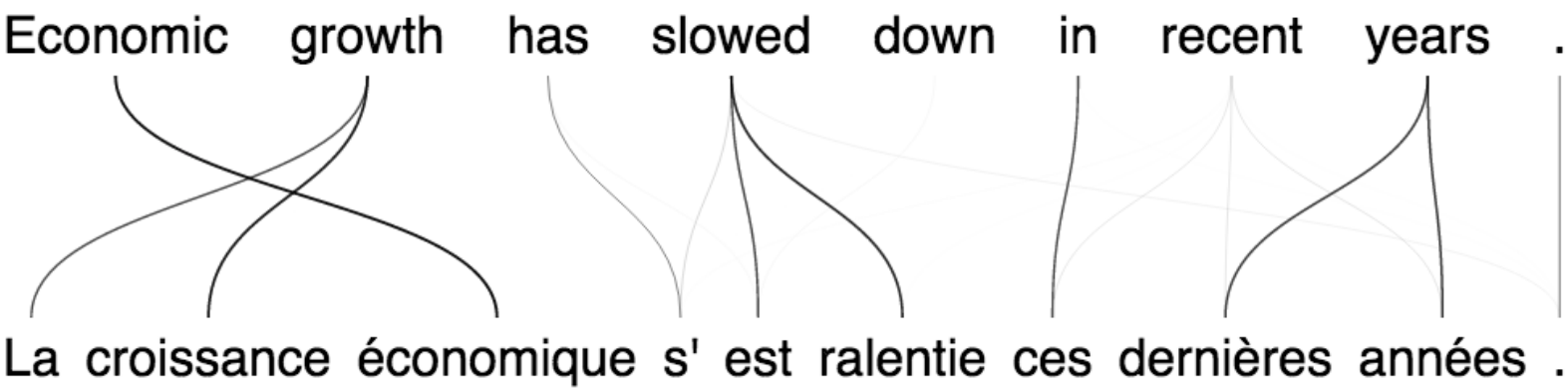
Google Voice

Google says its speech recognition technology now has only an 8% word error rate…



http://venturebeat.com/2015/05/28/google-says-its-speech-recognition-technology-now-has-only-an-8-word-error-rate/

Neural Machine Translation



http://104.131.78.120/#

PROS

- Online model (updates as you go)
  - Doesn't need to be fit all of the time
- Very fast predictions
- Can approximate almost any type of function
- Can be used in a supervised and unsupervised manner
- Super cool

CONS

- Requires many training samples to be considered good
- Hard to describe what is happening
- Requires a lot of hardware / computation power
- Slow to train
- Sklearn only has unsupervised version
- Other versions are difficult to use

The most advanced ANN's use thousand's of neurons which is a lot right?

The most advanced ANN's use thousand's of neurons which is a lot right?

Sure but my dog has billions…….

http://deepdreamgenerator.com/

Google uses a supervised neural network to recognize content in photos.

http://deepdreamgenerator.com/

Google uses a supervised neural network to recognize content in photos.

Turns out if you input an image you can ask the neural network to try and "re-create" the image as well

Other applications

- *Reinforcement Learning or how a machine can learn to play a game* *http://www.wired.co.uk/news/archive/2015-02/25/google-deepmind-atari*
- *Deep Learning of artistic style* *http://www.boredpanda.com/computer-deep-learning-algorithm-painting-masters/*
- *Medicine* *http://blog.kaggle.com/2012/10/31/merck-competition-results-deep-nn-and-gpus-come-out-to-play/*

https://medium.com/the-programmable-chronicles/rolling-in-the-deep-learning-4302bd5c06da

RESOURCES

Python libraries:
Pylearn2: http://deeplearning.net/software/pylearn2/
Lasagne: https://github.com/Lasagne/Lasagne
Keras: https://github.com/fchollet/keras
Deepy: https://github.com/uaca/deepy
Nolearn: https://github.com/dnouri/nolearn
Blocks: https://github.com/mila-udem/blocks
scikit-neuralnetworks: https://github.com/aigamedev/scikit-neuralnetwork

Other very popular:
Torch: http://torch.ch/
Caffe: http://caffe.berkeleyvision.org/
DeepLearning4J: http://deeplearning4j.org/

http://deeplearning.net/software_links/

LAB: Explore Keras and Scikit-Neuralnets