UVA CS 6316 – Fall 2015 Graduate: Machine Learning

Lecture 18: Neural Network / Deep Learning

Dr. Yanjun Qi

University of Virginia

Department of Computer Science

Where are we ? ➔ Five major sections of this course

- ❑ Regression (supervised)
- ❑ Classification (supervised)
- ❑ Unsupervised models
- ❑ Learning theory
- ❑ Graphical models

A study comparing Classifiers

An Empirical Comparison of Supervised Learning Algorithms

Rich Caruana CARUANA@CS.CORNELL.EDU
Alexandru Niculescu-Mizil ALEXN@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14853 USA

Abstract

A number of supervised learning methods have been introduced in the last decade. Unfortunately, the last comprehensive empirical evaluation of supervised learning was the Statlog Project in the early 90's. We present a large-scale empirical comparison between ten supervised learning methods: SVMs, neural nets, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps. We also examine the effect that calibrating the models via Platt Scaling and Isotonic Regression has on their performance. An important aspect of our study is

This paper presents results of a large-scale empirical comparison of ten supervised learning algorithms using eight performance criteria. We evaluate the performance of SVMs, neural nets, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps on eleven binary classification problems using a variety of performance metrics: accuracy, F-score, Lift, ROC Area, average precision, precision/recall break-even point, squared error, and cross-entropy. For each algorithm we examine common variations, and thoroughly explore the space of parameters. For example, we compare ten decision tree styles, neural nets of many sizes, SVMs with many kernels, etc.

Because some of the performance metrics we examine

Proceedings of the 23rd International Conference on Machine Learning (ICML `06).

Dr. Yanjun Qi / UVA CS 6316 / f15

A study comparing Classifiers

➔ 11 binary classification problems / 8 metrics

Table 2. Normalized scores for each learning algorithm by metric (average over eleven problems)

MODEL	CAL	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	MEAN	OPT-SEL
BST-DT	PLT	.843*	.779	**.939**	**.963**	**.938**	.929*	**.880**	**.896**	**.896**	**.917**
RF	PLT	.872*	.805	.934*	.957	.931	**.930**	.851	.858	.892	.898
BAG-DT	–	.846	.781	.938*	.962*	.937*	.918	.845	.872	.887*	.899
BST-DT	ISO	.826*	.860*	.929*	.952	.921	.925*	.854	.815	.885	.917*
RF	–	**.872**	.790	.934*	.957	.931	**.930**	.829	.830	.884	.890
BAG-DT	PLT	.841	.774	.938*	.962*	.937*	.918	.836	.852	.882	.895
RF	ISO	.861*	**.861**	.923	.946	.910	.925	.836	.776	.880	.895
BAG-DT	ISO	.826	.843*	.933*	.954	.921	.915	.832	.791	.877	.894
SVM	PLT	.824	.760	.895	.938	.898	.913	.831	.836	.862	.880
ANN	–	.803	.762	.910	.936	.892	.899	.811	.821	.854	.885
SVM	ISO	.813	.836*	.892	.925	.882	.911	.814	.744	.852	.882
ANN	PLT	.815	.748	.910	.936	.892	.899	.783	.785	.846	.875
ANN	ISO	.803	.836	.908	.924	.876	.891	.777	.718	.842	.884
BST-DT	–	.834*	.816	**.939**	**.963**	**.938**	.929*	.598	.605	.828	.851
KNN	PLT	.757	.707	.889	.918	.872	.872	.742	.764	.815	.837
KNN	–	.756	.728	.889	.918	.872	.872	.729	.718	.810	.830
KNN	ISO	.755	.758	.882	.907	.854	.869	.738	.706	.809	.844
BST-STMP	PLT	.724	.651	.876	.908	.853	.845	.716	.754	.791	.808
SVM	–	.817	.804	.895	.938	.899	.913	.514	.467	.781	.810
BST-STMP	ISO	.709	.744	.873	.899	.835	.840	.695	.646	.780	.810
BST-STMP	–	.741	.684	.876	.908	.853	.845	.394	.382	.710	.726
DT	ISO	.648	.654	.818	.838	.756	.778	.590	.589	.709	.774

Proceedings of the 23rd International Conference on Machine Learning (ICML `06).

A study comparing Classifiers
→ 11 binary classification problems

PROBLEM	#ATTR	TRAIN SIZE	TEST SIZE	%POZ
ADULT	14/104	5000	35222	25%
BACT	11/170	5000	34262	69%
COD	15/60	5000	14000	50%
CALHOUS	9	5000	14640	52%
COV_TYPE	54	5000	25000	36%
HS	200	5000	4366	24%
LETTER.P1	16	5000	14000	3%
LETTER.P2	16	5000	14000	53%
MEDIS	63	5000	8199	11%
MG	124	5000	12807	17%
SLAC	59	5000	25000	50%

Proceedings of the 23rd International Conference on Machine Learning (ICML `06).

Today

- Basic Neural Network (NN)
  - single neuron, e.g. logistic regression unit
  - multilayer perceptron (MLP)
  - for multi-class classification, softmax layer
  - More about training NN
- Deep CNN, Deep learning
  - History
  - Why is this breakthrough ?
  - Recent applications

Today

- Basic Neural Network (NN)
  - single neuron, e.g. logistic regression unit
  - multilayer perceptron (MLP)
  - for multi-class classification, softmax layer
  - More about training NN
- Deep CNN, Deep learning
  - History
  - Why is this breakthrough ?
  - Recent applications

Logistic Regression

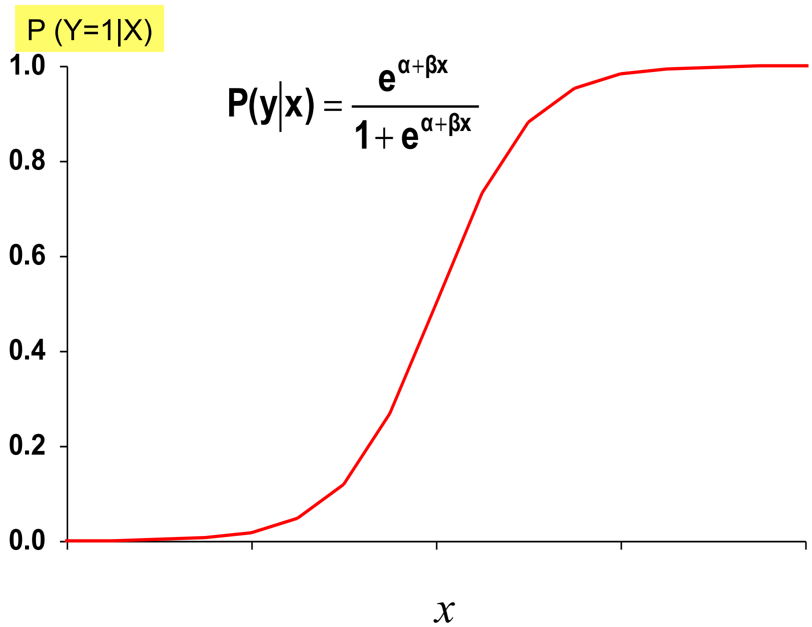
Task → Representation → Score Function → Search/Optimization → Models, Parameters

classification → Log-odds(Y) = linear function of X' s → EPE, with conditional Log-likelihood → Iterative (Newton) method → Logistic weights

$$P(y=1|x)=\frac{1}{1+e^{-(\alpha+\beta x)}}$$

Using Logistic Function to Transfer

e.g.
Probability of
disease

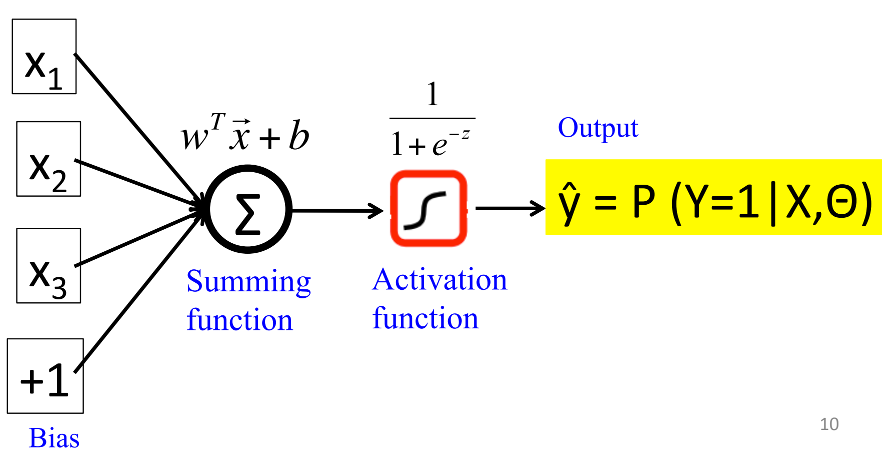


Logistic regression

Logistic regression could be illustrated as a module

On input x, it outputs ŷ:

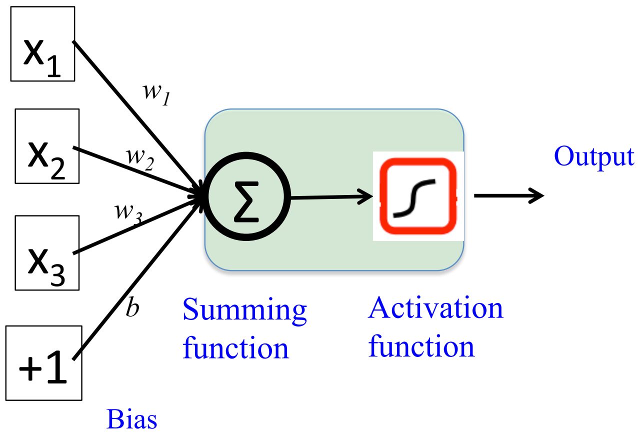
where



Draw a logistic regression unit as:

1 Neron example

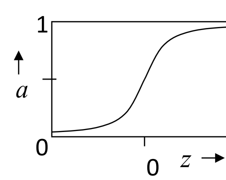
- 1 neuron, e.g.



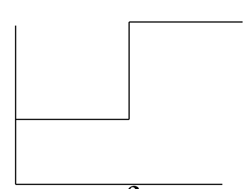
Transfer / Activation functions

- Common ones include:
  - Threshold / Step function: f(v) = 1 if v > c, else -1
  - Sigmoid (s shape func):
    - E.g. logistic func: $f(v) = 1/(1 + e^{-v})$, Range [0, 1]
  - Hyperbolic Tanh : $f(v) = (e^{v} - e^{-v})/(e^{v} + e^{-v})$, Range [-1,1]
- Desirable properties:
  - Monotonic, Nonlinear, Bounded
  - Easily calculated derivative

sigmoid func



$$a = \frac{1}{1+e^{-z}}$$

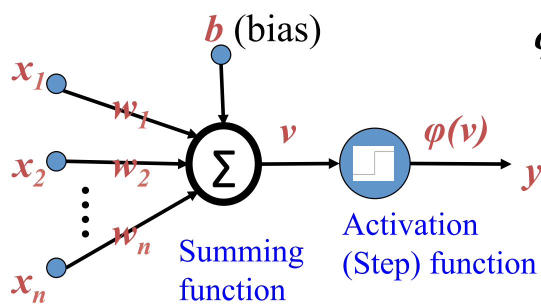


step func

Perceptron: Another 1-Neuron Unit

(Special form of single layer feed forward)

- The perceptron was first proposed by Rosenblatt (1958) is a simple neuron that is used to classify its input into one of two categories.
- A perceptron uses a **step function** that returns +1 if weighted sum of its input large or equal to 0, and -1 otherwise



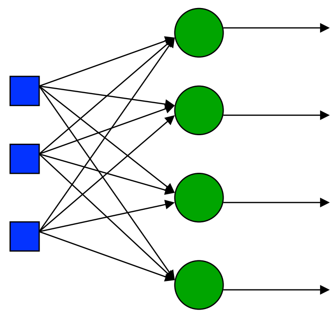
$$\varphi(v) = \begin{cases} +1 \text{ if } v \geq 0 \\ -1 \text{ if } v < 0 \end{cases}$$

Today

- Basic Neural Network (NN)
  - single neuron, e.g. logistic regression unit
  - multilayer perceptron (MLP)
  - for multi-class classification, softmax layer
  - More about training NN
- Deep CNN, Deep learning
  - History
  - Why is this breakthrough ?
  - Recent applications

Single Layer Feed-forward
i.e. (one layer of 4 output neurons)

Input layer of source nodes



Output layer of neurons

$\vec{y}$

input

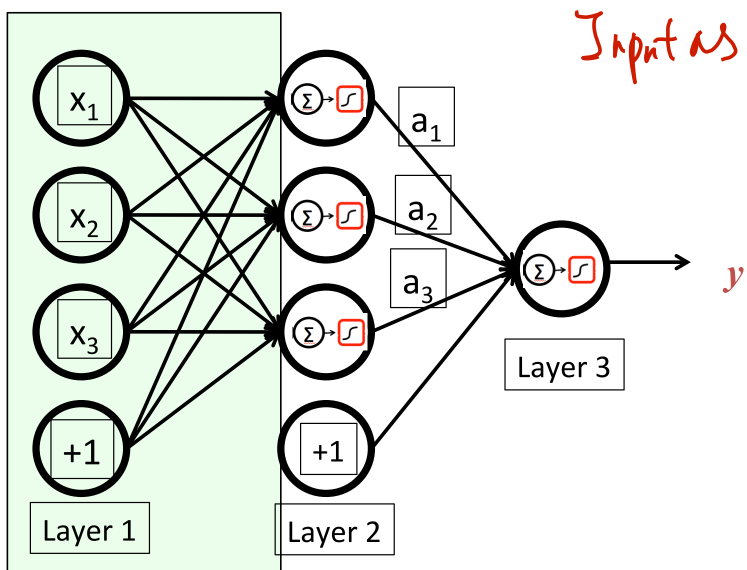
output

Four neurons

Multi-Layer Perceptron (MLP)

String a lot of logistic units together. Example: 3 layer network:

Input as first layer



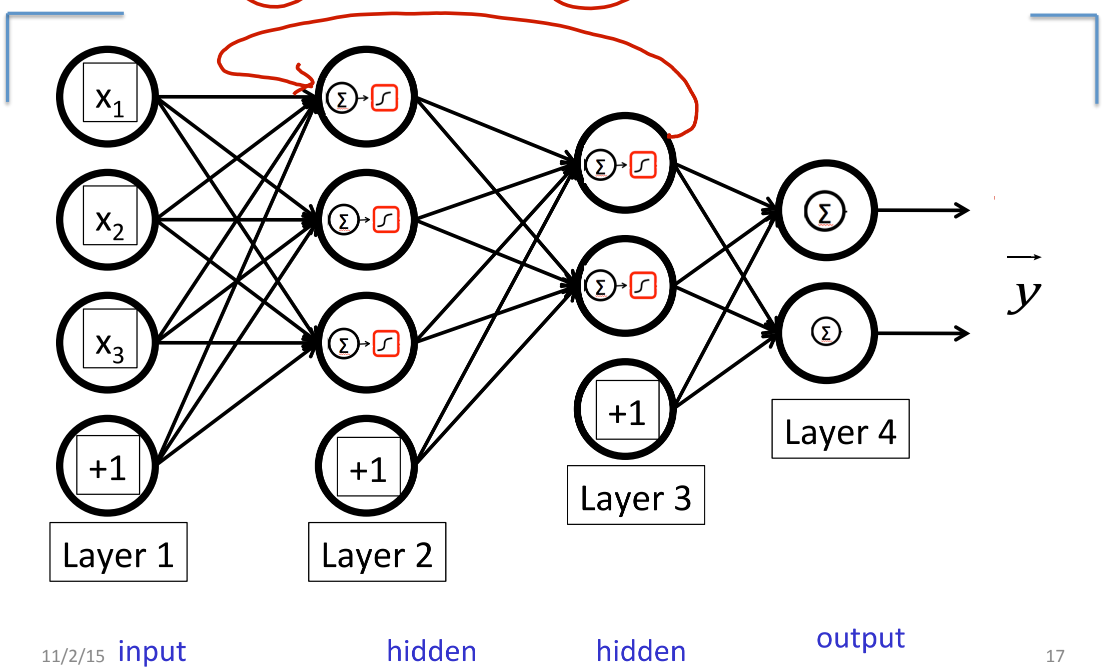
input

hidden

output

Multi-Layer Perceptron (MLP)

Example: 4 layer network with 2 output units:

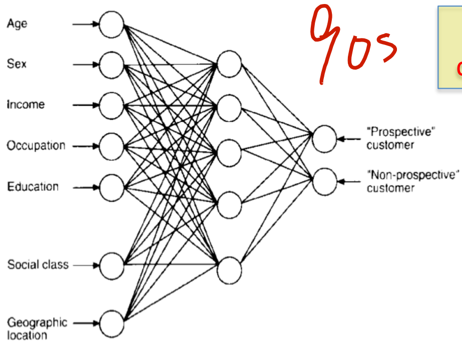


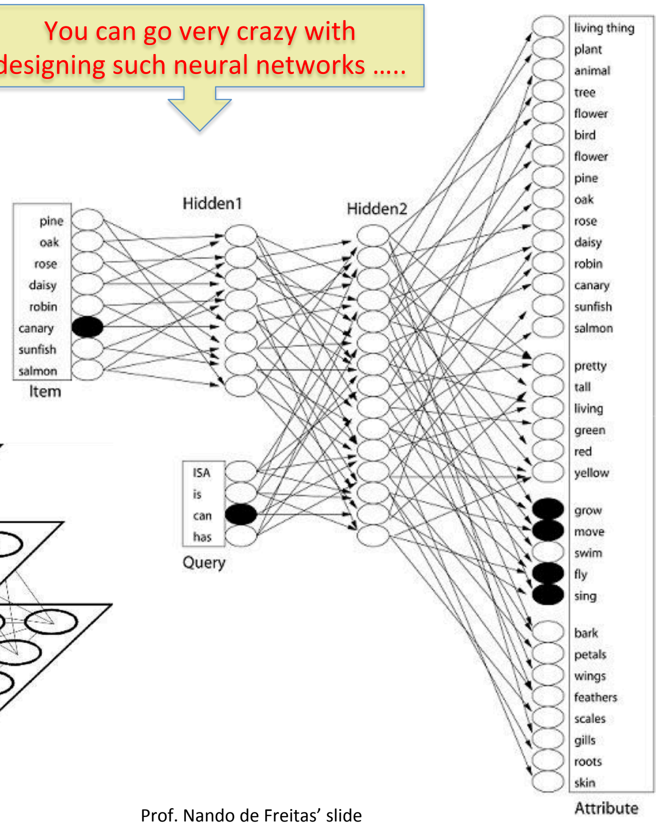
Types of Neural Networks (according to different attributes)

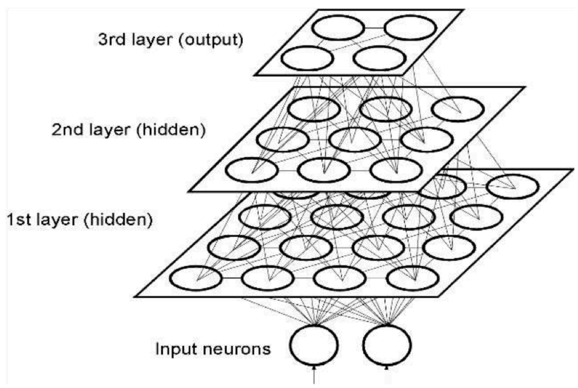
- **Connection Type (e.g.**
  - Static (feed-forward)
  - Dynamic (feedback)
- **Topology (e.g.**
  - Single layer
  - Multilayer
  - Recurrent
  - Recursive
  - Self-organized
- **Learning Methods (e.g.**
  - Supervised
  - Unsupervised

9/05

You can go very crazy with designing such neural networks .....







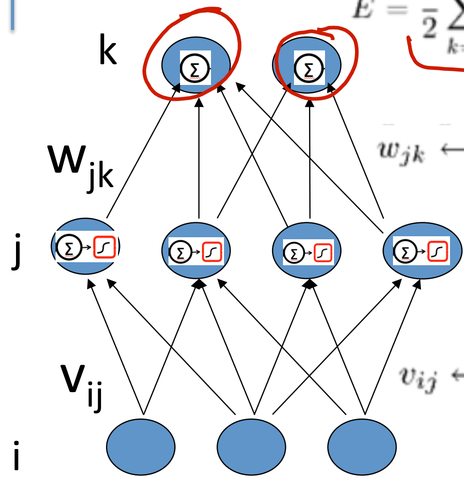
Prof. Nando de Freitas' slide

Today

- Basic Neural Network (NN)
  - single neuron, e.g. logistic regression unit
  - multilayer perceptron (MLP)
  - for multi-class classification, softmax layer
  - More about training NN
- Deep CNN, Deep learning
  - History
  - Why is this breakthrough ?
  - Recent applications

When for Regression

$$E = \frac{1}{2}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2$$

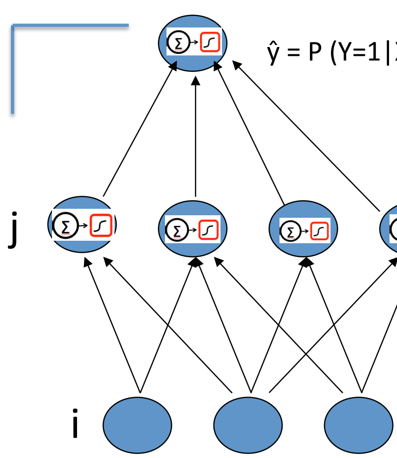


$$w_{jk} \leftarrow w_{jk} - \eta \frac{\partial E}{\partial w_{jk}}$$

$$v_{ij} \leftarrow v_{ij} - \eta \frac{\partial E}{\partial v_{ij}}$$

- Training NN in order to minimize the network total sum squared error (SSE).

When for classification (e.g. 1 neuron for binary output layer)



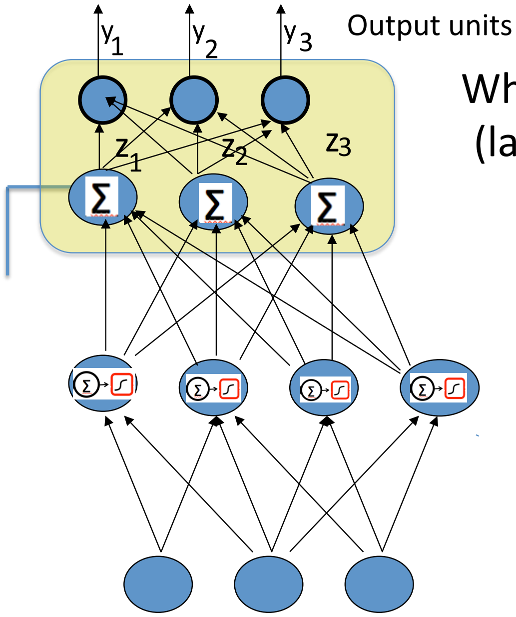
$\hat{y}$ = P (Y=1|X,Θ)

For Bernoulli distribution,
$p(y=1|x)^y(1-p)^{1-y}$

$$E(\theta) = Loss(\theta) = -\sum_{i=1}^{N}\{\log \Pr(Y = y_i | X = x_i)\}$$

$$= -\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)$$

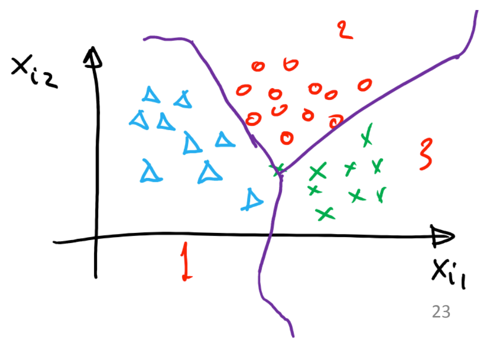
Cross-entropy loss function, OR named as "deviance"



When for multi-class classification (last output layer: softmax layer)

When multi-class output, last layer is softmax output layer ➔ a multinomial logistic regression unit

Y_{i1}	Y_{i2}	Y_{i3}	
0	1	0	class 2
1	0	0	class 1
0	0	1	class 3



Review: Multi-class variable representation

- Multi-class variable ➔

An indicator basis vector representation

– If output variable G has K classes, there will be K indicator variable y_i

Class

g	y_1	y_2	y_3	y_4
3	0	0	1	0
1	1	0	0	0
2	0	1	0	0
4	0	0	0	1
1	1	0	0	0

N

$f_1(x)$, $f_2(x)$, $f_3(x)$, $f_4(x)$

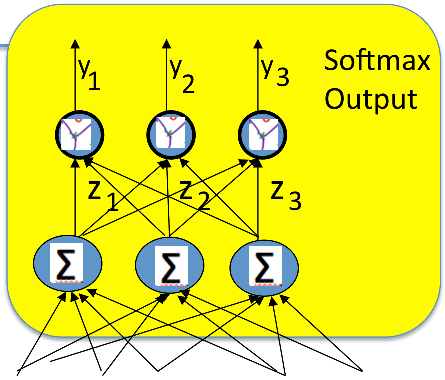
- How to classify to multi-class ?

– Strategy I: learn K different regression functions, then max

$$\hat{G}(x) = \underset{k \in g}{\mathbf{argmax}}\, \hat{f}_k(x)$$

Identify the largest component of $\hat{f}(x)$
And Classify according to Bayes Rule

Strategy II: Use "softmax" layer function for multi-class classification



$$Pr(G = k \mid X = x) = Pr(Y_k = 1 \mid X = x)$$

$$y_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

$$\frac{\partial y_i}{\partial z_i} = y_i\,(1 - y_i)$$

Use "softmax" layer function for multi-class classification

The natural cost function is the negative log prob of the right answer

➔ Cross entropy loss function :

$$E_x() = -\sum_{j=1....K} truey_j \ln y_j = -\sum_j truey_j \ln p(y_j = 1 \mid x)$$

$$\frac{\partial E}{\partial z_i} = \sum_j \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial z_i} = y_i - truey_i$$

Error calculated from Output vs. true

The steepness of function E exactly balances the flatness of the softmax

A special case of softmax for two classes

$$y_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_0}} = \frac{1}{1 + e^{-(z_1 - z_0)}}$$

- So the logistic is just a special case that avoids using redundant parameters:
  - Adding the same constant to both z1 and z0 has no effect.
  - The over-parameterization of the softmax is because the probabilities must add to 1.

Today

- Basic Neural Network (NN)
  - single neuron, e.g. logistic regression unit
  - multilayer perceptron (MLP)
  - for multi-class classification, softmax layer
  - More about training MLP NN
- Deep CNN, Deep learning
  - History
  - Why is this breakthrough ?
  - Recent applications

Backpropagation

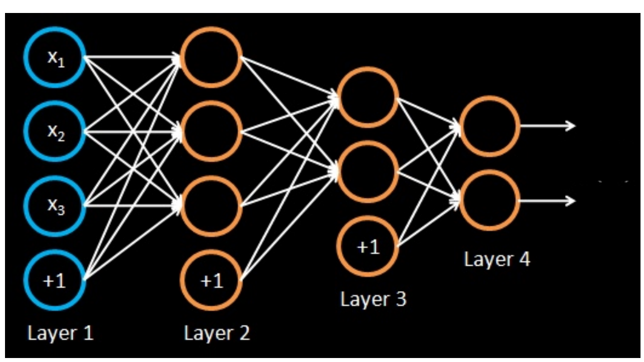
- Using backward recurrence to jointly optimize all parameters
- Requires all activation functions to be differentiable
- Enables flexible design in deep model architecture
- Gradient descent is used to (locally) minimize objective:

$$W^{k+1} = W^{k} - \eta \frac{\partial E}{\partial W^{k}}$$

delta rule

Y. LeCun et al. 1998. Efficient BackProp.
Olivier Bousquet and Ulrike von Luxburg. 2004. Stochastic Learning.

Training a neural network



Given training set (x_1, y_1), (x_2, y_2), (x_3, y_3), ….
Adjust parameters q (for every node) to make: the predicted output close to true label

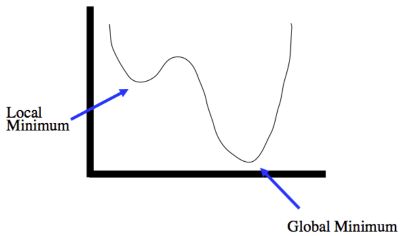
(Use gradient descent. “Backpropagation” algorithm. Susceptible to local optima.)

Review: Linear Regression with Stochastic GD ➔

- We have the following descent rule:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^T\theta - y_i)^2 \qquad \theta_j^{t+1} = \theta_j^t - \alpha \frac{\partial}{\partial \theta_j} J(\theta)\Big|_t$$

SSE



How do we pick α?

1. Tuning set, or
2. Cross validation, or
3. Small for slow, conservative learning

Review: Stochastic GD ➔

- For LR: linear regression, We have the following descent rule:

$$\theta_j^{t+1} = \theta_j^t - \alpha \frac{\partial}{\partial \theta_j} J(\theta)\Big|_t$$

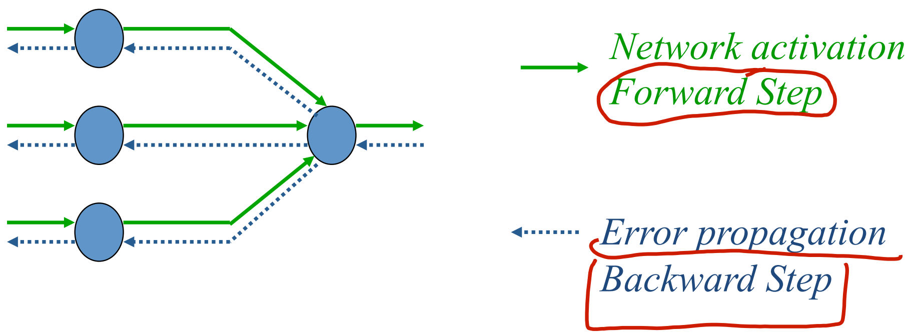
- ➔ For neural network, we have the delta rule

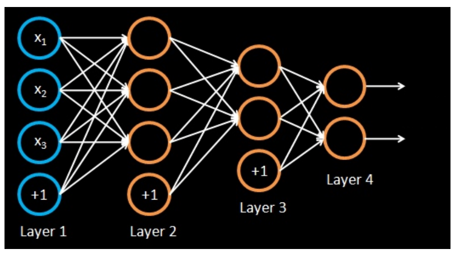
$$\Delta w = -\eta \frac{\partial E}{\partial W^t}$$

$$W^{t+1} = W^t - \eta \frac{\partial E}{\partial W^t} = W^t + \Delta w$$

Backpropagation

- Back-propagation training algorithm





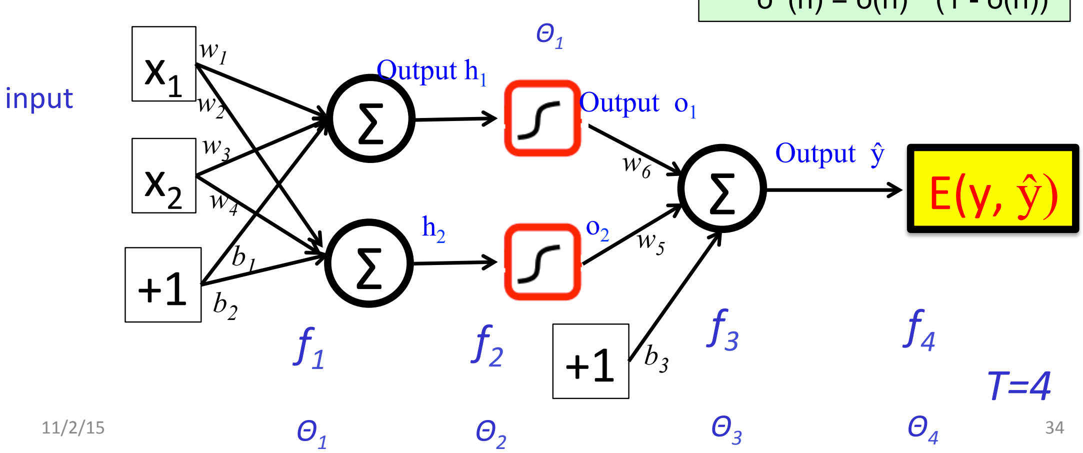
to train this layered network. The stacked layers in our network can be written in a more general form of multi-level functions:

$$l_{\mathbf{x}} = \mathbf{f}_T(\mathbf{f}_{T-1}(...(\mathbf{f}_1(\mathbf{x}))...)),$$

where $l_{\mathbf{x}}$ denotes the loss on a single example $\mathbf{x}$

For instance → for regression

for sigmoid unit o, its derivative is,
o' (h) = o(h) * (1 - o(h))



$\mathbf{f}_i, i \in [1, T]$, the derivative for updating its parameter set $\boldsymbol{\theta}_i$ is using the delta rule:

$$\theta_i^{k+1} = \theta_i^k - \alpha \frac{\partial l}{\partial \theta_i^k}$$

$$\frac{\partial l}{\partial \boldsymbol{\theta}_i} = \boxed{\frac{\partial \mathbf{f}_T}{\partial \mathbf{f}_i}} \times \frac{\partial \mathbf{f}_i}{\partial \boldsymbol{\theta}_i},$$

$\frac{\partial l}{\partial f_i}$

and the first factor on the right can be recursively calculated:

$$\boxed{\frac{\partial \mathbf{f}_T}{\partial \mathbf{f}_i}} = \frac{\partial \mathbf{f}_T}{\partial \mathbf{f}_{i+1}} \times \frac{\partial \mathbf{f}_{i+1}}{\partial \mathbf{f}_i}.$$

Note that $\mathbf{f}$ and θ are usually vectors so $\frac{\partial \mathbf{f}_T}{\partial \mathbf{f}_{i+1}}$ and $\frac{\partial \mathbf{f}_i}{\partial \boldsymbol{\theta}_i}$ are Jacobian matrices, and "$\times$" is matrix multiplication.

$f_4 = E = (y - \hat{y})^2$

$f_3 = \hat{y}$

e.g.

$$\frac{\partial f_4}{\partial f_3} = \frac{\partial (y - \underbrace{\hat{y}}_{f_3(f_2(f_1(x)))})^2}{\partial f_3} = \underbrace{-2(y - \hat{y})}_{\text{output error}}$$

Dr. QI's CIKM 2012 paper/talk

$$\frac{\partial f_T}{\partial f_i} = \frac{\partial f_T}{\partial f_{i+1}} \times \frac{\partial f_{i+1}}{\partial f_i}$$

$$= \frac{\partial f_T}{\partial f_{i+2}} \times \frac{\partial f_{i+2}}{\partial f_{i+1}} \times \frac{\partial f_{i+1}}{\partial f_i}$$

$$\frac{\partial l}{\partial \theta_1} = \frac{\partial f_4}{\partial f_1} \cdot \frac{\partial f_1}{\partial \theta_1}$$

$\Rightarrow$ e.g.

$$\frac{\partial f_4}{\partial f_1} = \frac{\partial f_4}{\partial f_3} \times \frac{\partial f_3}{\partial f_2} \times \frac{\partial f_2}{\partial f_1}$$

$\frac{\partial f_4}{\partial f_3} \Downarrow -2(y - \hat{y})$

output error

i.e. output error propagates backward layer by layer

$$f_3 = \hat{y} = W_6 O_1 + W_5 O_2 + b_3$$

$$f_2 = [O_1, O_2]^T$$

$$O_1 = \frac{1}{1+e^{-h_1}} \Rightarrow \frac{\partial O_1}{\partial h_1} = O_1(1-O_1)$$

$$O_2 = \frac{1}{1+e^{-h_2}}$$

$$f_1 = [h_1, h_2]^T$$

$$h_1 = W_1 X_1 + W_3 X_2 + b_1$$

$$h_2 = W_2 X_1 + W_4 X_2 + b_2$$

$$f_4 = (y-\hat{y})^2$$

$$\frac{\partial E}{\partial W_3} = \frac{\partial f_4}{\partial W_3} = \frac{\partial((y-\hat{y})^2)}{\partial W_3}$$

$$= -2(y-\hat{y})\frac{\partial \hat{y}}{\partial W_3}$$

$$= -2(y-\hat{y})\frac{\partial f_3}{\partial W_3}$$

$$= -2(y-\hat{y})\frac{\partial (W_6 O_1 + W_5 O_2 + b_3)}{\partial W_3}$$

$$= -2(y-\hat{y})\left(W_6 \frac{\partial O_1}{\partial W_3} + W_5 \frac{\partial O_2}{\partial W_3}\right)$$

$$= -2(y-\hat{y}) W_6 \frac{\partial O_1}{\partial W_3}$$

$$= -2(y-\hat{y}) W_6 O_1(1-O_1)\frac{\partial h_1}{\partial W_3}$$

$$= \underbrace{-2(y-\hat{y})}_{\text{output layer}} \underbrace{W_6}_{f_3 \text{ layer}} \underbrace{O_1(1-O_1)}_{f_2 \text{ layer}} \underbrace{X_2}_{f_1 \text{ layer}}$$

for $j = 1$ to MaxIter **do**
  if converge **then**
    break
  end if
  $\mathbf{x}, y \leftarrow$ random sampled data point and label
  calculate loss $l(\mathbf{x}; y)$
  cumulative $\leftarrow 1$
  for $i = \mathsf{T}$ to 1 **do**
    $\frac{\partial l}{\partial \boldsymbol{\theta}_i} \leftarrow \boxed{\text{cumulative}} * \frac{\partial \mathbf{f}_i}{\partial \boldsymbol{\theta}_i}$
    $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i - \lambda \frac{\partial l}{\partial \boldsymbol{\theta}_i}$
    $\boxed{\text{cumulative}} \leftarrow \boxed{\text{cumulative}} * \frac{\partial \mathbf{f}_{i+1}}{\partial \mathbf{f}_i}$
  end for
end for

Error propagation Backward Step

Dr. QI's CIKM 2012 paper/talk

Backpropagation

- 1. Initialize network with random weights
- 2. For all training cases (examples):
  - a. Present training inputs to network and calculate output of each layer, and final layer
  - b. For all layers (starting with the output layer, back to input layer):
    - i. Compare network output with correct output (error function)
    - ii. Adapt weights in current layer

$$W^{t+1} = W^t - \eta \frac{\partial E}{\partial W^t}$$

Backpropagation Algorithm

– Main Idea: error in hidden layers

The ideas of the algorithm can be summarized as follows :

1. Computes the **error term for the output units** using the observed error.

2. From output layer, repeat
   - propagating the error term back to the previous layer and
   - updating the weights between the two layers

   until the earliest hidden layer is reached.

Today

- Basic Neural Network (NN)
  - single neuron, e.g. logistic regression unit
  - multilayer perceptron (MLP)
  - for multi-class classification, softmax layer
  - More about training NN
- Deep CNN, Deep learning
  - History
  - Why is this breakthrough ?
  - Recent applications

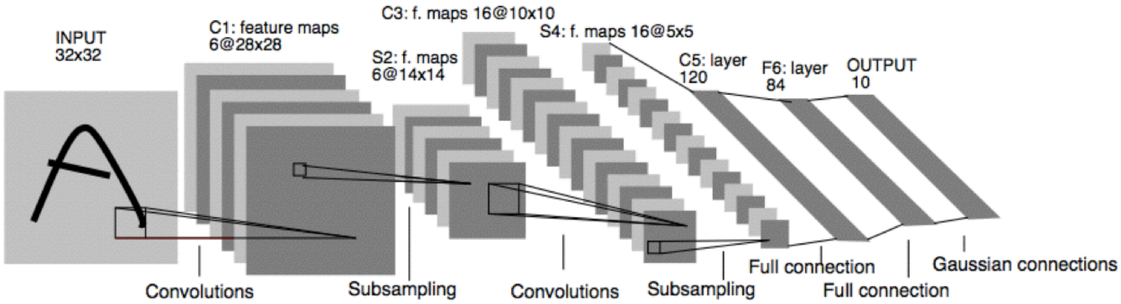
Many classification models invented since late 80's

- Neural networks
- Boosting
- Support Vector Machine
- Maximum Entropy
- Random Forest
- ……

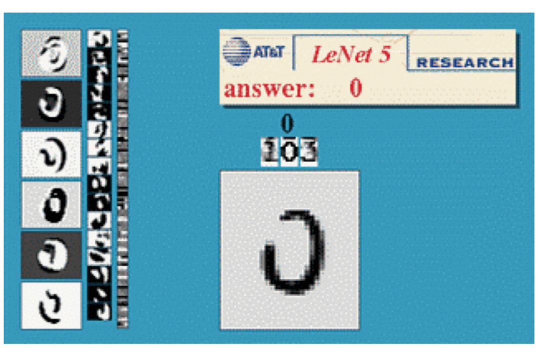
Deep Learning in the 90's

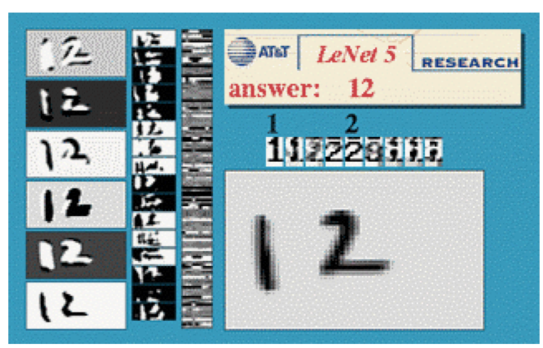
- Prof. Yann LeCun invented Convolutional Neural Networks

- First NN successfully trained with many layers



"LetNet" Early success at OCR





Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11): 2278–2324, 1998.

Between ~2000 to ~2011 Machine Learning Field Interest

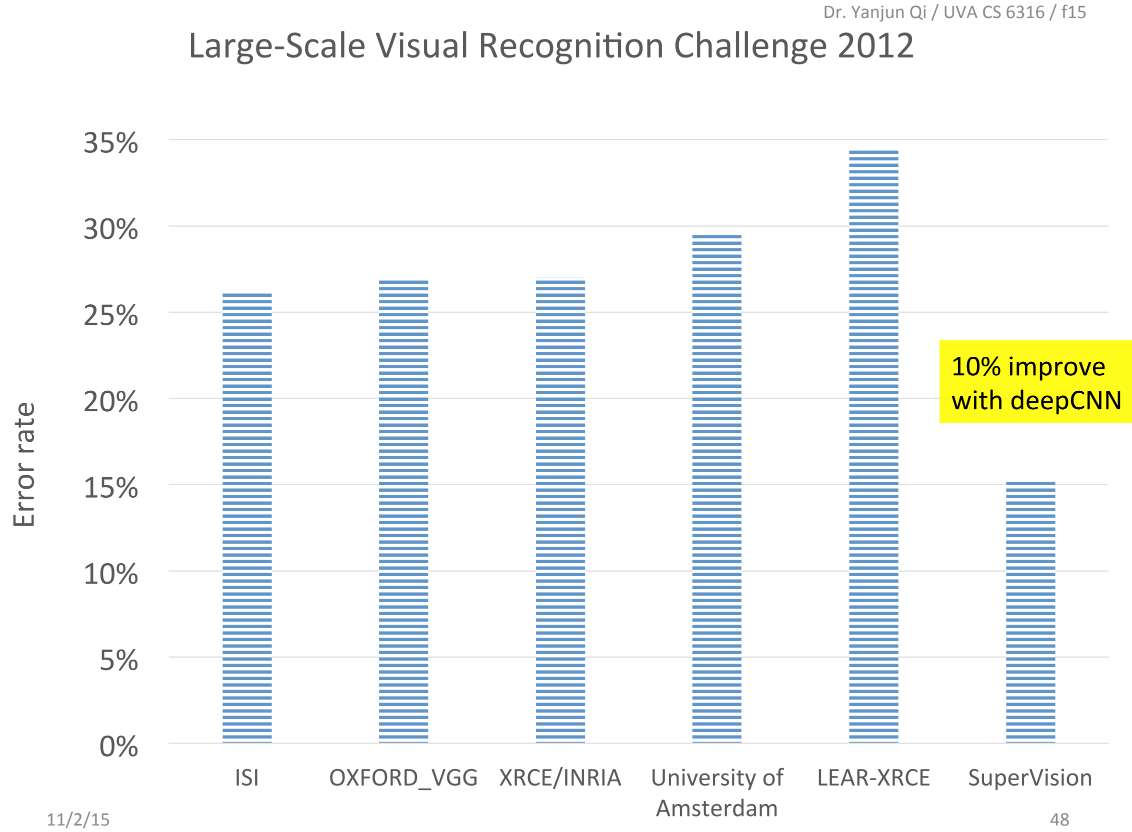
- Learning with Structures !
  - Kernel learning
  - Transfer Learning
  - Semi-supervised
  - Manifold Learning
  - Sparse Learning
  - Structured input-output learning …
  - Graphical model

"Winter of Neural Networks" Since 90's ! to ~2010

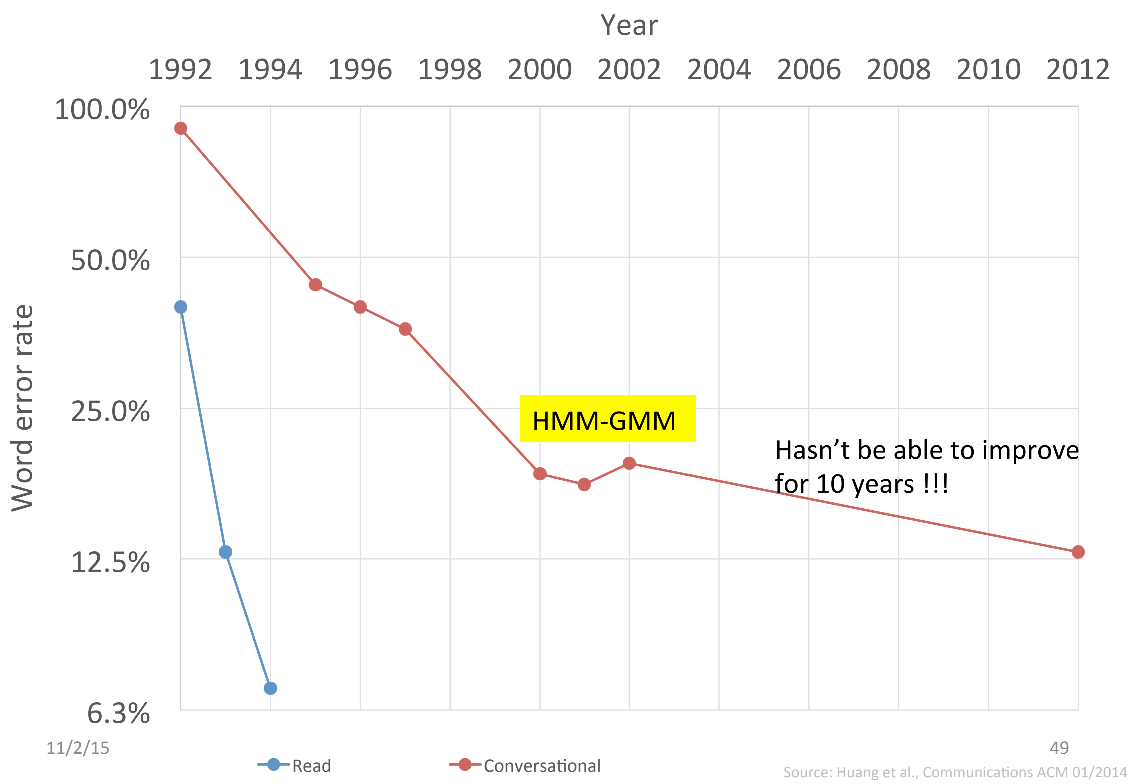
- Non-convex
- Need a lot of tricks to play with
  - How many layers ?
  - How many hidden units per layer ?
  - What topology among layers ? …….
- Hard to perform theoretical analysis

Today

- Basic Neural Network (NN)
  - single neuron, e.g. logistic regression unit
  - multilayer perceptron (MLP)
  - for multi-class classification, softmax layer
  - More about training NN
- Deep CNN, Deep learning
  - History
  - Why is this a breakthrough ?
  - Recent applications



Speech Recognition



Source: Huang et al., Communications ACM 01/2014

MIT Technology Review

10 BREAKTHROUGH TECHNOLOGIES 2013

Introduction | The 10 Technologies | Past Years

Deep Learning
With massive amounts of computational power, machines can now recognize objects and translate speech in real time. Artificial intelligence is finally getting smart.

Temporary Social Media
Messages that quickly self-destruct could enhance the privacy of online communications and make people freer to be spontaneous.

Prenatal DNA Sequencing
Reading the DNA of fetuses will be the next frontier of the genomic revolution. But do you really want to know about the genetic problems or musical aptitude of your unborn child?

Additive Manufacturing
Skeptical about 3-D printing? GE, the world's largest manufacturer, is on the verge of using the technology to make jet parts.

Baxter: The Blue-Collar Robot
Rodney Brooks's newest creation is easy to interact with, but the complex innovations behind the robot show just how hard it is to get along with people.

Memory Implants
A maverick neuroscientist believes he has deciphered the code by which the brain forms long-term memories. Next: testing a prosthetic implant for people suffering from long-term memory loss.

Smart Watches
The designers of the Pebble watch realized that a mobile phone is more useful if you don't have to take it out of your pocket.

Ultra-Efficient Solar Power
Doubling the efficiency of a solar cell would completely change the economics of renewable energy. Nanotechnology just might make it possible.

Big Data from Cheap Phones
Collecting and analyzing information from simple cell phones can provide surprising insights into how people move about and behave – and even help us understand the spread of diseases.

Supergrids
A new high-power circuit breaker could finally make highly efficient DC power grids practical.

WHY BREAKTHROUGH ?

How can we build more intelligent computer / machine ?

- Able to
  - **perceive the world**
  - **understand the world**

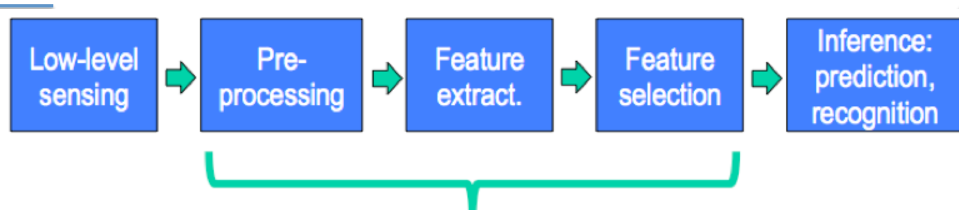
- This needs
  - Basic speech capabilities
  - Basic vision capabilities
  - Language understanding
  - User behavior / emotion understanding
  - Able to think ??

Plenty of Data

- Text: trillions of words of English + other languages
- Visual: billions of images and videos
- Audio: thousands of hours of speech per day
- User activity: queries, user page clicks, map requests, etc,
- Knowledge graph: billions of labeled relational triplets
- ………

Dr. Jeff Dean's talk

Deep Learning Way:
Learning features / Representation from data



Feature Engineering
- ✓ Most critical for accuracy
- ✓ Account for most of the computation for testing
- ✓ Most time-consuming in development cycle
- ✓ Often hand-craft and task dependent in practice

Feature Learning
- ✓ Easily adaptable to new similar tasks
- ✓ Layerwise representation
- ✓ Layer-by-layer unsupervised training
- ✓ Layer-by-layer supervised training

DESIGN ISSUES for **Deep NN**

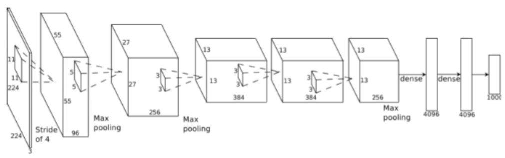
- Data representation
- Network Topology
- Network Parameters
- Training
  - Scaling up with **graphics processors**
  - Scaling up with **Asynchronous SGD**
- Validation

Dr. Yanjun Qi / UVA CS 6316 / f15

Today

- Basic Neural Network (NN)
  - single neuron, e.g. logistic regression unit
  - multilayer perceptron (MLP)
  - for multi-class classification, softmax layer
  - More about training NN
- Deep CNN, Deep learning
  - History
  - Why is this a breakthrough ?
  - Recent applications

Application I: Objective Recognition / Image Labeling



72%, 2010

74%, 2011

85%, 2012

89%, 2013

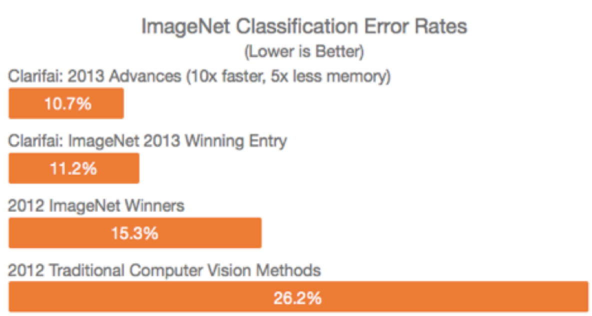
Deep Convolution Neural Network (CNN) won (as Best systems) on "very large-scale" ImageNet competition 2012 / 2013 / 2014

(**training on 1.2 million images [X] vs.1000 different word labels [Y]**)

- 2013, Google Acquired Deep Neural Networks Company headed by Utoronto "Deep Learning" Professor Hinton
- 2013, Facebook Built New Artificial Intelligence Lab headed by NYU "Deep Learning" Professor LeCun

ImageNet Challenge 2013

- Clarifai ConvNet model wins at 11% error rate



- Many other participants used ConvNets
- OverFeat by Pierre Sermanet from NYU: shipped binary program to execute pre-trained models

Olivier Grisel's talk

ImageNet Challenge 2014

- In the mean time Pierre Sermanet had joined other people from Google Brain
- Monster model: GooglеNet now at **6.7% error rate**

Dr. Jeff Dean's talk

Olivier Grisel's talk

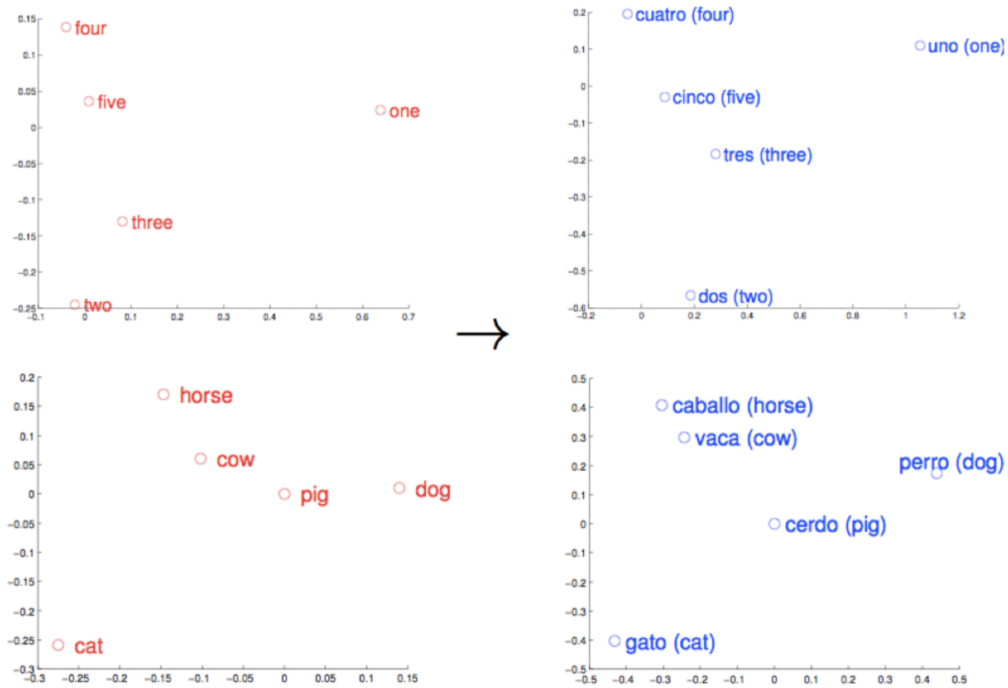
Application II: Semantic Understanding

Word embedding (e.g. google word2vector / skipgram)

- Learn to embed each word into a vector of real values
  - Semantically similar words have closer embedding representations
- Progress in 2013/14
  - Can uncover semantic /syntactic word relationships
    - [king] - [male] + [female] ~= [queen]
    - [Berlin] - [Germany] + [France] ~= [Paris]
    - [eating] - [eat] + [fly] ~= [flying]

Dr. Li Deng's talk

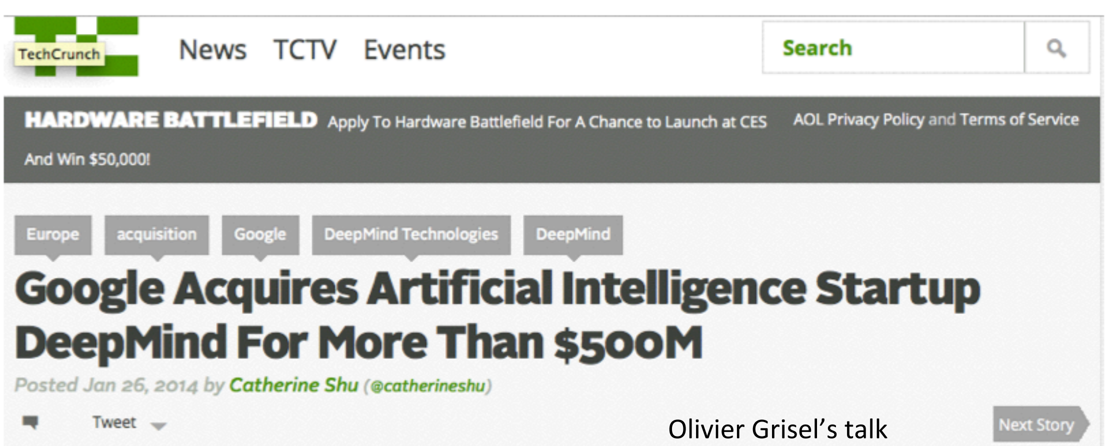
Dr. Jeff Dean's talk



source: Exploiting Similarities among Languages for MT

Application III: Deep Learning to Play Game

- DeepMind: Learning to Play & win dozens of Atari games
  – a new Deep Reinforcement Learning algorithm



Olivier Grisel's talk

Application IV: Deep Learning to Execute and Program

- Google Brain & NYU, October 2014
- RNN trained to map character representations of programs to outputs
- Can learn to emulate a simplistic Python interpreter Limited to one-pass programs with O(n) complexity

Olivier Grisel's talk

Application IV: Deep Learning to Execute and Program

Neural Turing Machines

- Google DeepMind, October 2014 (very new)
- Neural Network coupled to external memory (tape)
- Analogue to a Turing Machine but differentiable
- Can be used to learn to simple programs from example input / output pairs

Olivier Grisel's talk

Summary

- Basic Neural Network (NN)
  - single neuron, e.g. logistic regression unit
  - multilayer perceptron (MLP)
  - for multi-class classification, softmax layer
  - More about training NN
- Deep CNN, Deep learning
  - History
  - Why is this a breakthrough ?
  - Recent applications

References

- Dr. Yann Lecun's deep learning tutorials
- Dr. Li Deng's ICML 2014 Deep Learning Tutorial
- Dr. Kai Yu's deep learning tutorial
- Dr. Rob Fergus' deep learning tutorial
- Prof. Nando de Freitas' slides
- Olivier Grisel's talk at Paris Data Geeks / Open World Forum
- Hastie, Trevor, et al. *The elements of statistical learning*. Vol. 2. No. 1. New York: Springer, 2009.