

Master Recherche IAC
Option 2
Apprentissage Statistique & Optimisation
Avancés

Anne Auger — Balazs Kégl — Michèle Sebag
LRI — LAL

Dec. 4th, 2013

Contents

WHO

- | | |
|-----------------------------------|----------|
| ▶ Anne Auger, optimization | TAO, LRI |
| ▶ Balazs Kégl, machine learning | TAO, LAL |
| ▶ Michèle Sebag, machine learning | TAO, LRI |

WHAT

1. Neural Nets
2. Stochastic Optimization
3. Reinforcement Learning
4. Ensemble learning

WHERE: <http://tao.lri.fr/tiki-index.php?page=Courses>

Exam

Final: same as for TC2:

- ▶ Questions
- ▶ Problems

Volunteers

- ▶ Some pointers are in the slides
- ▶ Volunteers: read material, write one page, send it (sebag@lri.fr)

Tutorials/Videlectures

- ▶ <http://www.iro.umontreal.ca/~bengioy/talks/icml2012-YB-tutorial.pdf> (slides 1-56;79-133).

Questionnaire

Admin: Ouassim Ait El Hara

Debriefing

- ▶ What is clear/unclear
- ▶ Pre-requisites
- ▶ Work organization

Hypothesis Space \mathcal{H} / Navigation

	\mathcal{H}	navigation operators
Version Space	Logical	spec / gen
Decision Trees	Logical	specialisation
Neural Networks	Numerical	gradient
Support Vector Machines	Numerical	quadratic opt.
Ensemble Methods	—	adaptation \mathcal{E}

This course

- Neural Nets

Overview

Bio-inspired algorithms

Classical Neural Nets

- History

- Structure

Applications

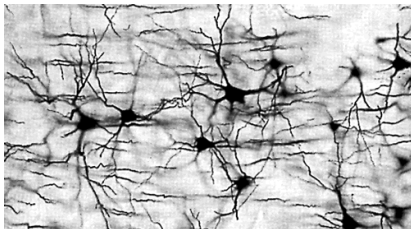
Advances

- Deep NN: Why does it work ?

- Deep NN: Drop-out

- Echo State Networks

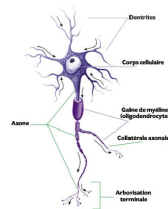
Bio-inspired algorithms



Facts

- ▶ 10^{11} neurons
- ▶ 10^4 connexions per neuron
- ▶ Firing time: $\sim 10^{-3}$ second

10^{-10} computers



Bio-inspired algorithms, 2

Human beings are the best !

- ▶ How do we do ?
 - ▶ What matters is not the number of neurons
as one could think in the 80s, 90s...
 - ▶ Massive parallelism ?
 - ▶ Innate skills ?
= anything we can't yet explain
 - ▶ Is it the training process ?

Beware of bio-inspiration

- ▶ Misleading inspirations (imitate birds to build flying machines)
- ▶ Limitations of the state of the art
- ▶ Difficult for a machine \neq difficult for a human

Synaptic plasticity

Hebb 1949

Conjecture

When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.

Learning rule

Cells that fire together, wire together

If two neurons are simultaneously excited, their connexion weight increases.

Remark: unsupervised learning.

Overview

Bio-inspired algorithms

Classical Neural Nets

History

Structure

Applications

Advances

Deep NN: Why does it work ?

Deep NN: Drop-out

Echo State Networks

History of artificial neural nets (ANN)

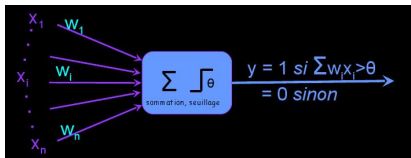
1. Non supervised NNs and logical neurons
2. Supervised NNs: Perceptron and Adaline algorithms
3. The NN winter: theoretical limitations
4. Multi-layer perceptrons.

History

Neural Nets

1940	Formal neurons	Mc Culloch & Pu
	Reinforcement law	Hebb
1950		
1960	Perceptron	Rosenblatt
	Adaline	Widrow
1970		
1980	Networks of	Hopfield
	Multi-layer perceptrons	Rumelhart & Mc Clelland, Le Cun
1990	Radius-based functions	Moody & Darken
	Support Vector Machines	Vapnik
2000		
	Deep Learning	Hinton; Bengio
2010		

Thresholded neurons Mc Culloch et Pitt 1943



Ingredients

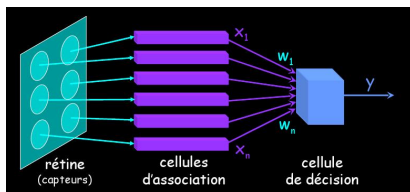
- ▶ Input (dendrites) x_i
- ▶ Weights w_i
- ▶ Threshold θ
- ▶ Output: 1 iff $\sum_i w_i x_i > \theta$

Remarks

- ▶ Neurons \rightarrow Logics \rightarrow Reasoning \rightarrow Intelligence
- ▶ Logical NNs: can represent any boolean function
- ▶ No differentiability.

Perceptron

Rosenblatt 1958



$$y = \text{sign}(\sum w_i x_i - \theta)$$

$$\mathbf{x} = (x_1, \dots, x_d) \mapsto (x_1, \dots, x_d, 1).$$

$$\mathbf{w} = (w_1, \dots, w_d) \mapsto (w_1, \dots, w_d, -\theta)$$

$$y = \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle)$$

Learning a Perceptron

Given

- ▶ $\mathcal{E} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathbb{R}^d, y_i \in \{1, -1\}, i = 1 \dots n\}$

For $i = 1 \dots n$, do

- ▶ If no mistake, do nothing

$$\begin{aligned}\text{no mistake} &\Leftrightarrow \langle \mathbf{w}, \mathbf{x} \rangle \text{ same sign as } y \\ &\Leftrightarrow y \langle \mathbf{w}, \mathbf{x} \rangle > 0\end{aligned}$$

- ▶ If mistake

$$\mathbf{w} \leftarrow \mathbf{w} + y \cdot \mathbf{x}_i$$

Enforcing algorithmic stability:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_t y \cdot \mathbf{x}_\ell$$

α_t decreases to 0 faster than $1/t$.

Convergence: upper bounding the number of mistakes

Assumptions:

- ▶ \mathbf{x}_i belongs to $\mathcal{B}(\mathbb{R}^d, C)$ $\|\mathbf{x}_i\| < C$
- ▶ \mathcal{E} is separable, i.e.
exists solution \mathbf{w}^* s.t. $\forall i = 1 \dots n, y_i \langle \mathbf{w}^*, \mathbf{x}_i \rangle > \delta > 0$

Convergence: upper bounding the number of mistakes

Assumptions:

- ▶ \mathbf{x}_i belongs to $\mathcal{B}(\mathbb{R}^d, C)$ $\|\mathbf{x}_i\| < C$
- ▶ \mathcal{E} is separable, i.e.
exists solution \mathbf{w}^* s.t. $\forall i = 1 \dots n, y_i \langle \mathbf{w}^*, \mathbf{x}_i \rangle > \delta > 0$
with $\|\mathbf{w}^*\| = 1$.

Convergence: upper bounding the number of mistakes

Assumptions:

- ▶ \mathbf{x}_i belongs to $\mathcal{B}(\mathbb{R}^d, C)$ $\|\mathbf{x}_i\| < C$
- ▶ \mathcal{E} is separable, i.e.
exists solution \mathbf{w}^* s.t. $\forall i = 1 \dots n, y_i \langle \mathbf{w}^*, \mathbf{x}_i \rangle > \delta > 0$
with $\|\mathbf{w}^*\| = 1$.

Then The perceptron makes at most $(\frac{C}{\delta})^2$ mistakes.

Bounding the number of misclassifications

Proof

Upon the k -th misclassification

for some \mathbf{x}_i

$$\begin{aligned}\mathbf{w}_{k+1} &= \mathbf{w}_k + y_i \mathbf{x}_i \\ \langle \mathbf{w}_{k+1}, \mathbf{w}^* \rangle &= \langle \mathbf{w}_k, \mathbf{w}^* \rangle + y_i \langle \mathbf{x}_i, \mathbf{w}^* \rangle \\ &\geq \langle \mathbf{w}_k, \mathbf{w}^* \rangle + \delta \\ &\geq \langle \mathbf{w}_{k-1}, \mathbf{w}^* \rangle + 2\delta \\ &\geq k\delta\end{aligned}$$

In the meanwhile:

$$\begin{aligned}\|\mathbf{w}_{k+1}\|^2 &= \|\mathbf{w}_k + y_i \mathbf{x}_i\|^2 \leq \|\mathbf{w}_k\|^2 + C^2 \\ &\leq kC^2\end{aligned}$$

Therefore:

$$\sqrt{k}C > k\delta$$

Going farther...

Remark: Linear programming: Find \mathbf{w}, δ such that

$$\begin{aligned} & \text{Max } \delta, \quad \text{subject to} \\ & \quad \forall i = 1 \dots n, \quad y_i \langle \mathbf{w}, \mathbf{x}_i \rangle > \delta \end{aligned}$$

gives the floor to Support Vector Machines...

Adaptive Linear Element

Given

$$\mathcal{E} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}, i = 1 \dots n\}$$

Learning

Minimization of a quadratic function

$$\mathbf{w}^* = \operatorname{argmin}\{Err(\mathbf{w}) = \sum (y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle)^2\}$$

Gradient algorithm

$$\mathbf{w}_i = \mathbf{w}_{i-1} + \alpha_i \nabla Err(\mathbf{w}_i)$$

The NN winter

Neural Nets

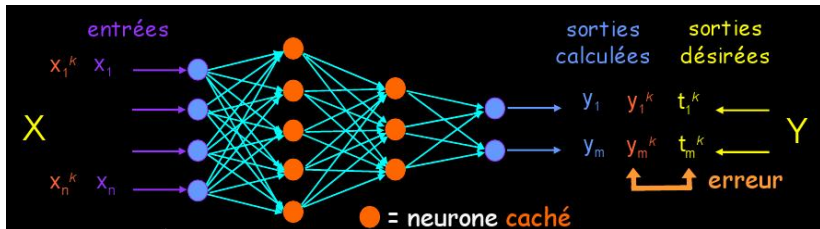
1940	Formal neurons	Mc Culloch & Pu
	Reinforcement law	Hebb
1950		
1960	Perceptron	Rosenblatt
	Adaline	Widrow
1970		
1980	Networks of	Hopfield
	Multi-layer perceptrons	Rumelhart & Mc Clelland, Le Cun
1990	Radius-based functions	Moody & Darken
	Support Vector Machines	Vapnik
2000		
	Deep Learning	Hinton; Bengio
2010		

Limitation of linear hypotheses

The XOR problem.

Minsky Papert 1969

Multi-Layer Perceptrons, Rumelhart McClelland 1986



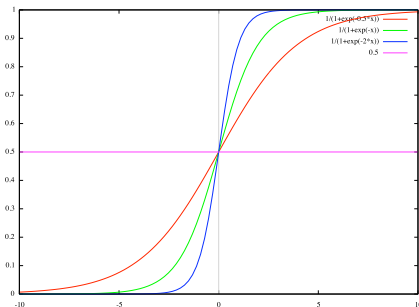
Issues

- ▶ Several layers, non linear separation, addresses the XOR problem
- ▶ A **differentiable** activation function

$$\text{output}(\mathbf{x}) = \frac{1}{1 + \exp\{-\langle \mathbf{w}, \mathbf{x} \rangle\}}$$

The sigmoid function

- ▶ $\sigma(t) = \frac{1}{1+\exp(-a \cdot t)}$, $a > 0$
- ▶ approximates step function (binary decision)
- ▶ linear close to 0
- ▶ Strong increase close to 0
- ▶ $\sigma'(x) = a\sigma(x)(1 - \sigma(x))$



Back-propagation algorithm, Rumelhart McClelland 1986; Le Cun 1986

- ▶ Given (\mathbf{x}, y) a training sample uniformly randomly drawn
- ▶ Set the d entries of the network to $x_1 \dots x_d$
- ▶ Compute iteratively the output of each neuron until final layer: output \hat{y} ;
- ▶ Compare \hat{y} and y $Err(w) = (\hat{y} - y)^2$
- ▶ Modify the NN weights on the last layer based on the gradient value
- ▶ Looking at the previous layer: we know what we would have liked to have as output; infer what we would have liked to have as input, i.e. as output on the previous layer. And back-propagate...
- ▶ Errors on each i -th layer are used to modify the weights used to compute the output of i -th layer from input of i -th layer.

Back-propagation of the gradient

Notations

Input $\mathbf{x} = (x_1, \dots, x_d)$

From input to the first hidden layer

$$z_j^{(1)} = \sum w_{jk} x_k$$

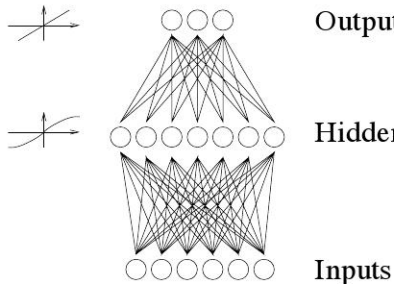
$$x_j^{(1)} = f(z_j^{(1)})$$

From layer i to layer $i + 1$

$$z_j^{(i+1)} = \sum w_{jk}^{(i)} x_k^{(i)}$$

$$x_j^{(i+1)} = f(z_j^{(i+1)})$$

(f : e.g. sigmoid)



Back-propagation of the gradient

Input (\mathbf{x}, y) , $\mathbf{x} \in \mathbb{R}^d$, $y \in \{-1, 1\}$

Phase 1 Propagate information forward

- ▶ For layer $i = 1 \dots \ell$

For every neuron j on layer i

$$z_j^{(i)} = \sum_k w_{j,k}^{(i)} x_k^{(i-1)}$$

$$x_j^{(i)} = f(z_j^{(i)})$$

Phase 2 Compare the target output (y) to what you get ($x_1^{(\ell)}$)

NB: for simplicity one assumes here that there is a single output (the label is a scalar value).

- ▶ Error: difference between $\hat{y} = x_1^{(\ell)}$ and y .

Define

$$e^{sortie} = f'(z_1^{(\ell)})[\hat{y} - y]$$

where $f'(t)$ is the (scalar) derivative of f at point t .

Back-propagation of the gradient

Phase 3 retro-propagate the errors

$$e_j^{(i-1)} = f'(z_j^{(i-1)}) \sum_k w_{kj}^{(i)} e_k^{(i)}$$

Phase 4: Update weights on all layers

$$\Delta w_{ij}^{(k)} = \alpha e_i^{(k)} x_j^{(k-1)}$$

where α is the learning rate (< 1 .)

Overview

Bio-inspired algorithms

Classical Neural Nets

History

Structure

Applications

Advances

Deep NN: Why does it work ?

Deep NN: Drop-out

Echo State Networks

Neural nets

Ingredients

- ▶ Activation function
- ▶ Connexion topology = directed graph
feedforward (\equiv DAG, directed acyclic graph) or recurrent
- ▶ A (scalar, real-valued) weight on each connexion

Activation(z)

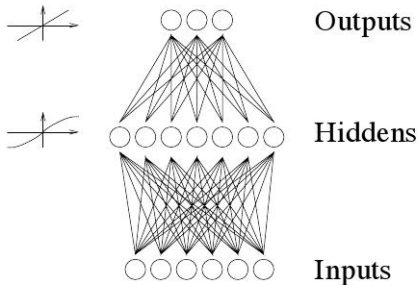
- ▶ thresholded $0 \text{ if } z < \textit{threshold}, 1 \text{ otherwise}$
- ▶ linear z
- ▶ sigmoid $1/(1 + e^{-z})$
- ▶ Radius-based e^{-z^2/σ^2}

Neural nets

Ingredients

- ▶ Activation function
- ▶ Connexion topology = directed graph
feedforward (\equiv DAG, directed acyclic graph) or recurrent
- ▶ A (scalar, real-valued) weight on each connexion

Feedforward NN



Neural nets

Ingredients

- ▶ Activation function
- ▶ Connexion topology = directed graph
feedforward (\equiv DAG, directed acyclic graph) or recurrent
- ▶ A (scalar, real-valued) weight on each connexion

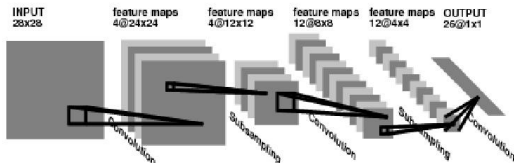
Recurrent NN

- ▶ Propagate until stabilisation
- ▶ Back-propagation does not apply
- ▶ Memory of the recurrent NN: value of hidden neurons
Beware that memory fades exponentially fast
- ▶ Dynamic data (audio, video)

Structure / Connexion graph / Topology

Prior knowledge

- ▶ Invariance under translation, rotation, .. op
- ▶ \rightarrow Complete \mathcal{E} consider $(op(\mathbf{x}_i), y_i)$
- ▶ or use weight sharing: convolutional networks



100,000 weights \rightarrow 2,600 parameters

Details

- ▶ <http://yann.lecun.com/exdb/lenet/>
- ▶ <http://deeplearning.net/tutorial/lenet.html>

Demos

Hubel & Wiesel 1968

Visual cortex of the cat

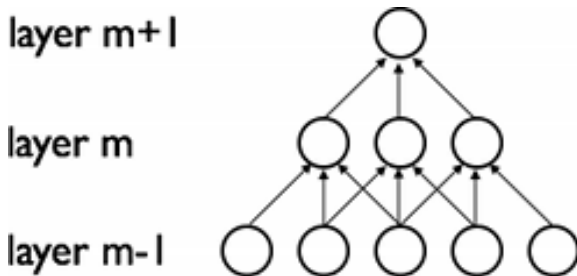
- ▶ cells arranged in such a way that
- ▶ ... each cell observes a fraction of the visual field
- ▶ the union of which covers the whole field

receptive field

Characteristics

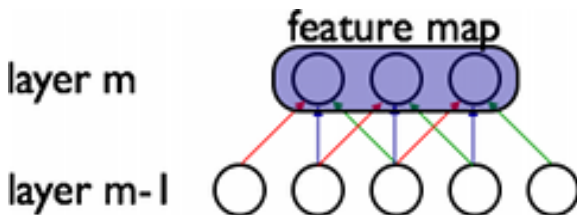
- ▶ Simple cells check the presence of a pattern
- ▶ More complex cells consider a larger receptive field, detect the presence of a pattern up to translation/rotation

Sparse connectivity



- ▶ Reducing the number of weights
- ▶ Layer m : detect local patterns
- ▶ Layer $m + 1$: non linear aggregation, more global field

Convolutional NN: shared weights

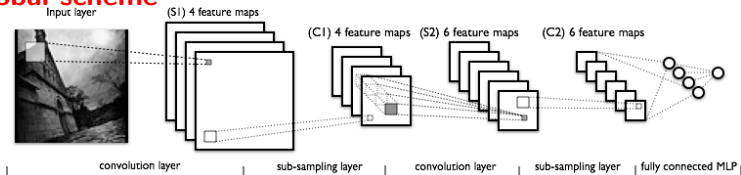


- ▶ Reducing the number of weights
- ▶ through adapting the gradient-based update: the update is averaged over all occurrences of the weight.

Max pooling: reduction and invariance

- ▶ Partitioning
 - ▶ Return the max value in the subset
- invariance

Global scheme



Properties

Good news

- ▶ MLP, RBF: universal approximators

For every decent function f ($= f^2$ has a finite integral on every compact of \mathbb{R}^d)

for every $\epsilon > 0$,

there exists some MLP/RBF g such that $\|f - g\| < \epsilon$.

Bad news

- ▶ Not a constructive proof (the solution exists, so what ?)
- ▶ Everything is possible \rightarrow no guarantee (overfitting).

Key issues

Model selection

- ▶ Selecting number of neurons, connexion graph
- ▶ Which learning criterion

overfitting

More \nrightarrow Better

Algorithmic choices

a difficult optimization problem

- ▶ Initialisation **\mathbf{w} small !**
- ▶ Decrease the learning rate with time
- ▶ Enforce stability through relaxation

$$\mathbf{w}_{neo} \leftarrow (1 - \alpha)\mathbf{w}_{old} + \alpha\mathbf{w}_{neo}$$

- ▶ Stopping criterion

Start by normalization of data

$$X \mapsto \frac{x - \text{average}}{\text{variance}}$$

The curse of NNs

● The NIPS community has suffered of an acute convexititis epidemic

- ▶ ML applications seem to have trouble moving beyond logistic regression, SVMs, and exponential-family graphical models.
 - ▶ For a new ML model, convexity is viewed as a virtue
 - ▶ Convexity is sometimes a virtue
 - ▶ But it is often a limitation
-
- ▶ ML theory has essentially never moved beyond convex models
 - the same way control theory has not really moved beyond linear systems
 - ▶ Often, the price we pay for insisting on convexity is an unbearable increase in the size of the model, or the scaling properties of the optimization algorithm [$O(n^2)$, $O(n^3)$...]

http://videolectures.net/em107_lecun_wia/

Pointers

URL

- ▶ course:
`http://neuron.tuke.sk/math.chtf.stuba.sk/pub/vlado/NN_books_texts/Krose_Smagt_neuro-intro.pdf`
- ▶ FAQ: `http://www.faqs.org/faqs/ai-faq/neural-nets/part1/preamble.html`
- ▶ applets
`http://www.lri.fr/~marc/EEAAX/Neurones/tutorial/`
- ▶ codes: PDP++/Emergent (`www.cnbc.cmu.edu/PDP++/`);
SNNS `http://www-ra.informatik.uni-tuebingen.de/SgNNS/...`

Also see

- ▶ NEAT & HyperNEAT Stanley, U. Texas
When no examples available: e.g. robotics.

Overview

Bio-inspired algorithms

Classical Neural Nets

History

Structure

Applications

Advances

Deep NN: Why does it work ?

Deep NN: Drop-out

Echo State Networks

Applications

1. Pattern recognition
 - ▶ Signs (letters, figures)
 - ▶ Faces
 - ▶ Pedestrians
2. Control (navigation)
3. Language

Intuition

Design, the royal road

- ▶ Decompose a system into building blocks
- ▶ which can be specified, implemented and tested independently.

Why looking for another option ?

Intuition

Design, the royal road

- ▶ Decompose a system into building blocks
- ▶ which can be specified, implemented and tested independently.

Why looking for another option ?

- ▶ When the first option does not work or takes too long (face recognition)
- ▶ when dealing with an open world

Proof of concept

- ▶ speech & hand-writing recognition: with enough data, machine learning yields accurate recognition algorithms.
- ▶ hand-crafting → learning

Recognition of letters



Fig. 4. Size-normalized examples from the MNIST database.

Features

- ▶ Input size d : +100
- ▶ \rightarrow large weight vectors :-)
- ▶ Prior knowledge: invariance through (moderate) translation, rotation of pixel data

Convolutional networks

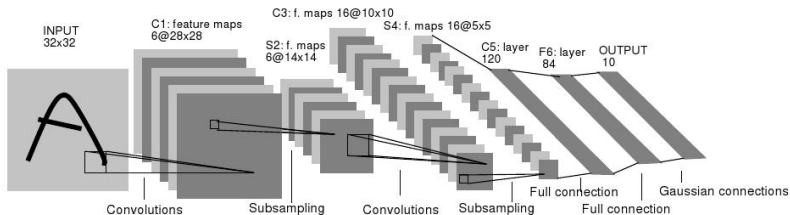


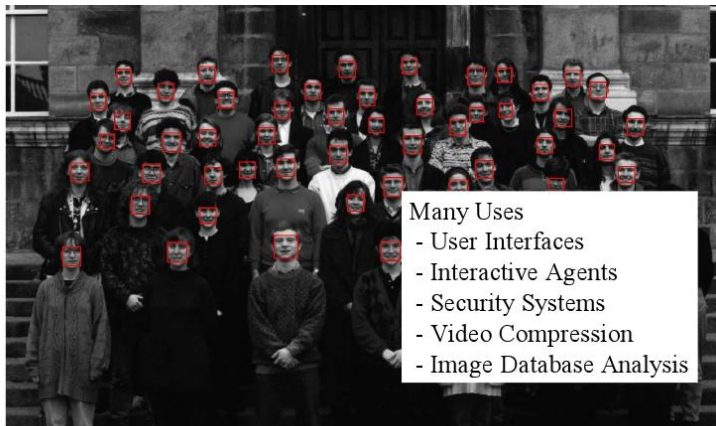
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Lecture

<http://yann.lecun.com/exdb/lenet/>

- Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time-series. In M. A. Arbib, editor, The Handbook of Brain Theory and Neural Networks. MIT Press, 1995.

Face recognition



Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001

Face recognition

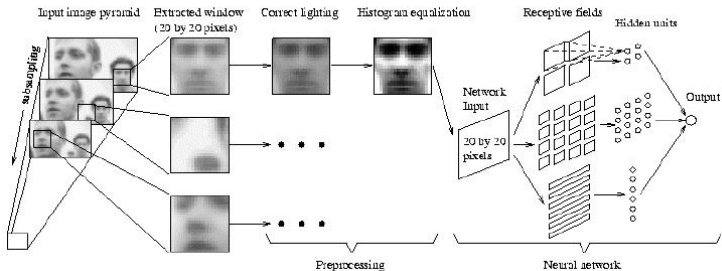
Variability

- ▶ Pose
- ▶ Elements: glasses, beard...
- ▶ Light
- ▶ Expression
- ▶ Orientation

Occlusions

<http://www.ai.mit.edu/courses/6.891/lectnotes/lect12/lect12-slides-6up.pdf>

Face recognition, 2



- ▶ One equation \rightarrow 1 NN
- ▶ NN are **fast**

Face recognition, 3

Oval mask for ignoring
background pixels:



Original window:



Best fit linear function:



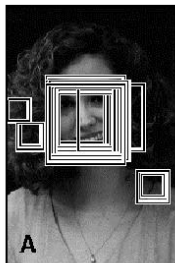
Lighting corrected window:
(linear function subtracted)



Histogram equalized window:



The steps in preprocessing a window. First, a linear function is fit to the intensity values in the window, and then subtracted out, correcting for some extreme lighting conditions. Then, histogram equalization is applied, to correct for different camera gains and to improve contrast. For each of these steps, the mapping is computed based on pixels inside the oval mask, while the mapping is applied to the entire window.



Navigation, control



Lectures, Video

<http://www.cs.nyu.edu/~yann/research/dave/index.html>

Continuous language models

Principle

- ▶ Input: 10,000-dim boolean input (words)
- ▶ Hidden neurons: 500 continuous neurons
- ▶ Goal: from a text window $(w_i \dots w_{i+2k})$, predict
 - ▶ The grammatical tag of the central word w_{i+k}
 - ▶ The next word w_{i+2k+1}
- ▶ Rk: Hidden layer: maps a text window on \mathbb{R}^{500}

Bengio et al. 2001

Continuous language models, Collobert et al. 2008

Improving Word Embedding

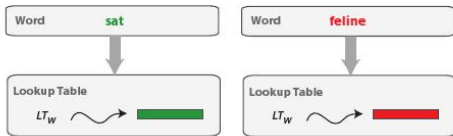


Rare words are not trained properly



Sentences with similar words should be tagged in the same way:

- ★ The cat sat on the mat
- ★ The feline sat on the mat



Wordnet

- ★ pull together linked words
- ★ push apart other pair of words

videlectures

Continuous language models, Collobert et al. 2008

Language Model: Think Massive



Language Model: *"is a sentence actually english or not?"*

Implicitly captures: * syntax * semantics



Bengio & Ducharme (2001) Probability of next word given previous words. Overcomplicated – we do not need probabilities here



English sentence windows: Wikipedia (~ 631M words)

Non-english sentence windows: middle word randomly replaced



Multi-class margin cost:

$$\sum_{s \in \mathcal{S}} \sum_{w \in \mathcal{D}} \max(0, 1 - f(s, w_s^*) + f(s, w))$$

\mathcal{S} : sentence windows \mathcal{D} : dictionary

w_s^* : true middle word in s

$f(s, w)$: network score for sentence s and middle word w

Continuous language models, Collobert et al. 2008

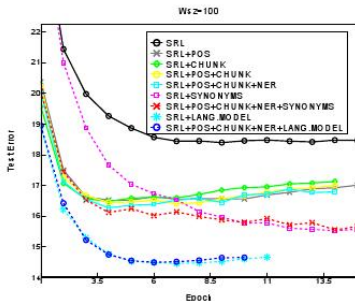
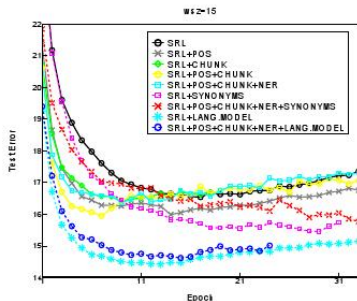
Language Model: Embedding

france	jesus	xbox	reddish	scratched
454	1973	6909	11724	29869
spain	christ	playstation	yellowish	smashed
italy	god	dreamcast	greenish	ripped
russia	resurrection	psNUMBER	brownish	brushed
poland	prayer	snes	bluish	hurled
england	yahweh	wii	creamy	grabbed
denmark	josephus	nes	whitish	tossed
germany	moses	nintendo	blackish	squeezed
portugal	sin	gamecube	silvery	blasted
sweden	heaven	psp	greyish	tangled
austria	salvation	amiga	paler	slashed

Dictionary size: 30,000 words. Even rare words are well embedded.

Continuous language models, Collobert et al. 2008

MTL: Semantic Role Labeling



We get: 14.30%. State-of-the-art: 16.54% – Pradhan et al. (2004)



250× faster than state-of-the-art. ~ 0.01s to label a WSJ sentence.

Continuous language models, Collobert et al. 2008

MTL: Unified Network for NLP

Improved results with Multi-Task Learning (MTL)

Task	Alone	MTL
SRL	18.40%	14.30%
POS	2.95%	2.91%
Chunking – error rate	5.4%	4.9%
Chunking – F1-score	91.5%	93.6%



POS: state-of-the-art $\sim 3\%$



Chunking: Best system had 93.48% F1-score at CoNLL-2000 challenge <http://www.cnts.ua.ac.be/conll2000/chunking>. State-of-the-art is 94.1%. We get 94.9% by using POS features.

Continuous language models, Collobert et al. 2008

Summary



We developed a deep neural network architecture for NLP



Advantages

- * General to any NLP tagging task
- * State-of-the-art performance
- * No hand designed features
- * Joint training
- * Can exploit massive unlabeled data
- * Extremely fast: 0.02s for all tags of a sentence



Inconvenients

- * Neural networks are a powerful tool: hard to handle



Early Impacts

- * Easy to apply to other tasks or languages: extending to Japanese
- * Fast: developed a semantic search system

Overview

Bio-inspired algorithms

Classical Neural Nets

History

Structure

Applications

Advances

Deep NN: Why does it work ?

Deep NN: Drop-out

Echo State Networks

Why does it work ?

Erhan et al, 2010

Possible explanations

- ▶ Reduces overfitting = regularization
- ▶ Unsupervised learning yields a better starting point for gradient \rightarrow better local optimum.

How to test ?

Why does it work ?

Erhan et al, 2010

Possible explanations

- ▶ Reduces overfitting = regularization
- ▶ Unsupervised learning yields a better starting point for gradient → better local optimum.

How to test ?

- ▶ If it's regularization, the effect should decrease as the amount of data increases.

The regularization effect

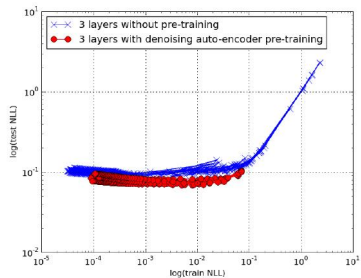
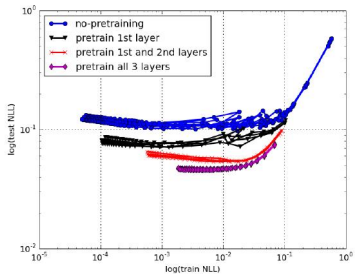
Claim

Representation good for $P(x)$ are also good for $P(y|x)$.

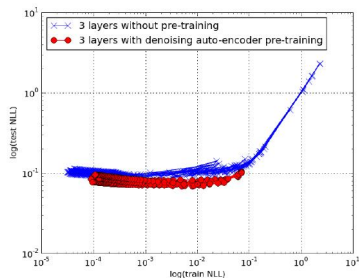
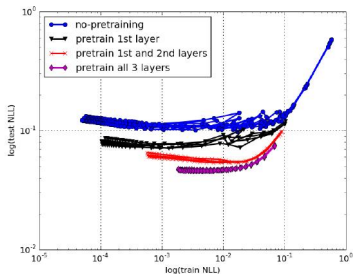
Experimental setting

- ▶ 3 layers, [800,1200] units per layer
- ▶ Unsupervised pre-training (Infinite MNIST) of no layers, 1st layer, 1-2 layers, all layers (other random)
- ▶ Supervised training
- ▶ Test

The regularization effect, 2



The regularization effect, 2



Results

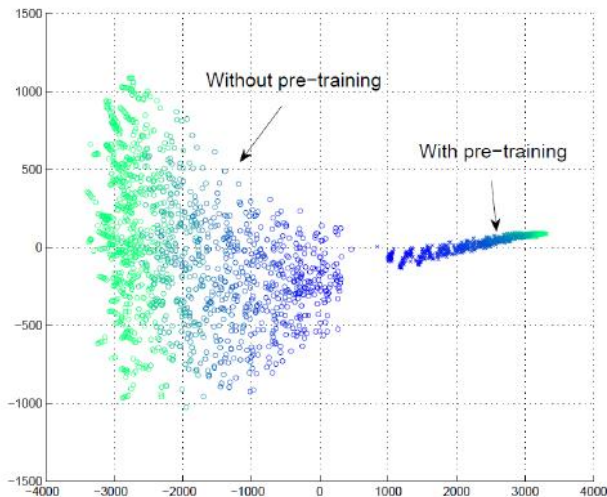
- ▶ Overfitting (left)
- ▶ Limiting top-layer size (20 units) helps.

Unsupervised pre-training supports gradient optimization

Experimental setting

- ▶ At each iteration during supervised training, map the $800 \times 800 \times 3$ -dimensional weight vector on \mathbb{R}^2
- ▶ Plot the trajectory of the stochastic gradient (points darker for later iterations)

Unsupervised pre-training and gradient initialization



Results

- ▶ Not the same region
- ▶ Variance much decreased

Overview

Bio-inspired algorithms

Classical Neural Nets

History

Structure

Applications

Advances

Deep NN: Why does it work ?

Deep NN: Drop-out

Echo State Networks

Limitation ?

Literature: Very effective prediction == ensemble learning

- ▶ Learn many hypotheses
- ▶ Average them (we'll see random forests, ensemble of decision trees, next week)

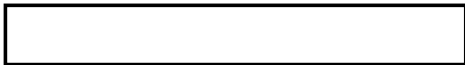
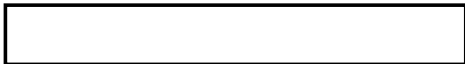
Ensemble of Deep NNs ?

- ▶ Takes for ever to learn...
- ▶ Takes long to test.

An efficient way to average Deep Nets

Hinton et al. 2012

- ▶ Consider a NN with 1 hidden layer
- ▶ When an example is presented, omit each hidden neuron with probability .5
- ▶ == randomly sampling from 2^h NNs
- ▶ Note that all architectures share weights



An efficient way to average Deep Nets, 2

Principle

- ▶ One samples from 2^h NNs
- ▶ Few NNs get trained, and if trained, from very few examples
- ▶ Sharing the weights = regularization (but biasing the weights toward a mean value; not toward 0 as in L_2 regularization)

Good practice

- ▶ If Deep NN overfits, dropout will reduce errors
- ▶ If does not overfit, you should be using a bigger NN (!!!)

Analogy

- ▶ More synapses (10^{14}) than experiences
- ▶ Synapses are cheap, experiences are costly.

Improving neural networks by preventing co-adaptation of feature detectors, 2012

Overview

Bio-inspired algorithms

Classical Neural Nets

History

Structure

Applications

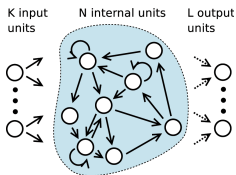
Advances

Deep NN: Why does it work ?

Deep NN: Drop-out

Echo State Networks

Echo State Networks



NIPS 2005 Wshop, Reservoir Computing
= Echo State Network, Jaeger 2001
∪ Liquid State Machine, Maas 2002

Structure

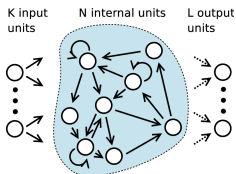
- ▶ N neurones cachés
- ▶ Connexions: aléatoires

matrice \mathcal{G}

$$p(x_i, x_j) \text{ connectés} = r, \quad r \ll 1$$

- ▶ Poids: aléatoires: 1, -1, 0.
- ▶ Stabilité: max. valeur propre de \mathcal{G} (damping factor) < 1
- ▶ Neurones de sortie: combinaison linéaire.

Echo State Networks, 2



A revolution

- ▶ The end of micro-management for NN (only ρ and λ)
- ▶ Training (e.g. for regression) through quadratic optimization

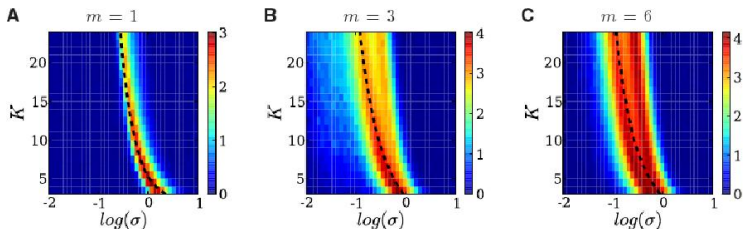
More:

On computational power and the Order Chaos: Phase Transition in Reservoir Computing, Benjamin Schrauder, Lars BÜsing and Robert Legenstein, NIPS 2008.

The competence region of Echo State Networks

Task

- ▶ Input: a random sequence of bits
- ▶ Desired output: same sequence with a delay (the ESN must *memorize* the input).



- ▶ K : number of connexions of a neuron
- ▶ σ : Gaussian weights, drawn from $\mathcal{N}(0, \sigma)$.