Deep Learning

Alexandre Allauzen, Michèle Sebag
CNRS & Université Paris-Sud

Nov. 14th, 2017

Credit for slides: Yoshua Bengio, Yann Le Cun, Nando de Freitas, Christian Perone, Honglak Lee

Types of Machine Learning problems

WORLD – DATA – USER

Observations	+ **Target**	+ Rewards
Understand **Code**	**Predict** **Classification/Regression**	Decide Action Policy/Strategy
Unsupervised **LEARNING**	**Supervised** **LEARNING**	Reinforcement LEARNING

News

Good News: Neural Nets can be used for all three goals:

- Unsupervised learning — change of representation
- Supervised learning — achieves prediction
- Reinforcement learning — yields the state-action value

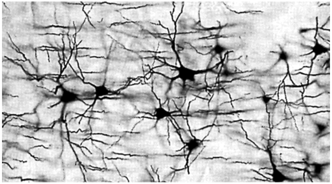
Bad News

- not so easy to learn — optimization
- not so easy to understand — black-box model
- its extensions (to complex/higher order logic domains) require *finesse*

The Deep Learning AIC Master Module

Contents

1. Neural Nets: 1943-1969, 1979-2005
2. Deep Learning: 2005-now
3. Applications



Evaluation

- Project
- Exam
- Voluntary contributions (5mn talks, adding resources)

Pointers

Where

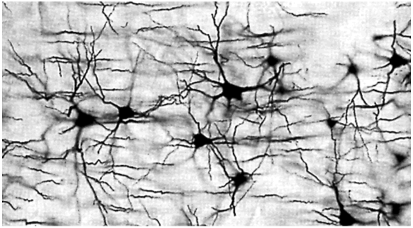
- Hugo La Rochelle
  NNs on YouTube: https://www.youtube.com/playlist?list=PL6Xpj9I5qXYEcOhn7TqghAJ6NAPrNmUBH
- Yann Le Cun
  Collge de France: https://www.college-de-france.fr/site/yann-lecun/
- Nando de Freitas
  Oxford: https://www.youtube.com/watch?v=PlhFWT7vAEw

How: suggested

- See videos *before* the course
- Let's discuss during the course what is unclear, what could be done otherwise, what could be done.

Overview

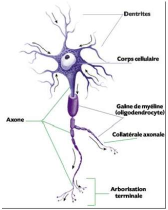
Biological inspiration



Facts

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

10^{-10} computers



Biological inspiration, 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 ?

    https://computervisionblog.wordpress.com/2013/06/01/cats-and-vision-is-vision-acquired-or-innate/

Beware of biological metaphors

- 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

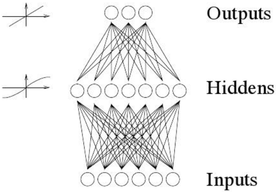
Cells that fire together, wire together

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

Remark: unsupervised learning.

Overview

Neural Nets

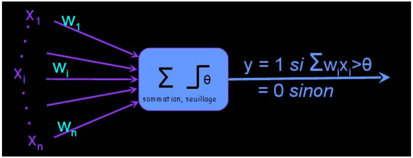


(C) David McKay - Cambridge Univ. Press

History

1943	A neuron as a computable function $y = f(\mathbf{x})$ Intelligence $\rightarrow$ Reasoning $\rightarrow$ Boolean functions	Pitts, McCullough
1960	Connexionism + learning algorithms	Rosenblatt
1969	AI Winter	Minsky-Papert
1989	Back-propagation	Amari, Rumelhart & McClelland, LeCun
1995	Winter again	Vapnik
2005	Deep Learning	Bengio, Hinton

Thresholded neurons Mc Culloch et Pitt 1943



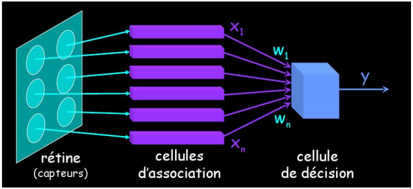
Ingredients

- 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



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

$$\mathbf{x} = (x_1, \ldots, x_d) \mapsto (x_1, \ldots, x_d, 1). \ \ \mathbf{w} = (w_1, \ldots, w_d) \mapsto (w_1, \ldots w_d, -\theta)$$

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

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

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

- If mistake

$$\mathbf{w} \leftarrow \mathbf{w} + y_i . \mathbf{x}_i$$

Enforcing algorithmic stability:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha_t y_\ell . \mathbf{x}_\ell$$

α_t decreases to 0 faster than $1/t$.

Convergence: upper bounding the number of mistakes

Assumptions:

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

Convergence: upper bounding the number of mistakes

Assumptions:

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

Convergence: upper bounding the number of mistakes

Assumptions:

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

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 $\mathbf{w}, \delta$ such that

$$\begin{aligned} &\mathit{Max}\ \delta, \quad \mathit{subject\ to} \\ &\qquad \forall\, i = 1 \ldots n,\ \ y_i \left\langle \mathbf{w}, \mathbf{x}_i \right\rangle > \delta \end{aligned}$$

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

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

Gradient algorithm

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

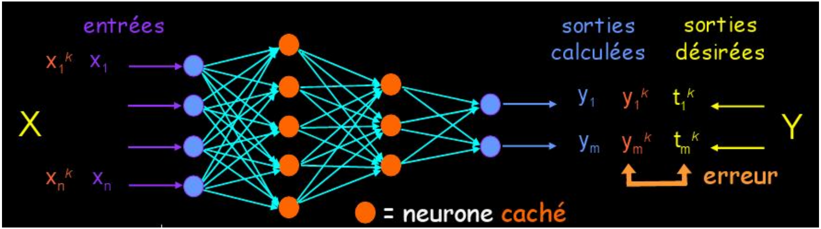
The NN winter

1943	A neuron as a computable function $y = f(\mathbf{x})$	Pitts, McCullough
	Intelligence $\rightarrow$ Reasoning $\rightarrow$ Boolean functions	
1960	Connexionism + learning algorithms	Rosenblatt
1969	**AI Winter**	**Minsky-Papert**
1989	Back-propagation	Amari, Rumelhart & McClelland, LeCun
1995	Winter again	Vapnik
2005	Deep Learning	Bengio, Hinton

Limitation of linear hypotheses: The XOR problem.

Multi-Layer Perceptrons

Rumelhart McClelland 1986



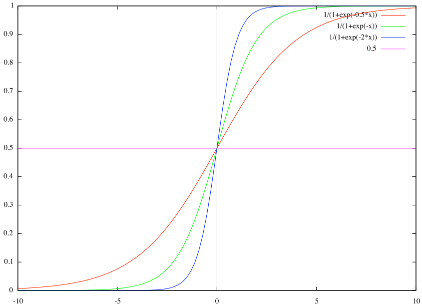
Issues

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

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

The sigmoid function

- $\sigma(t) = \frac{1}{1+exp(-a\,.t)}, a > 0$
- 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

Amari 69, 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, 1

Notations

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

From input to the first hidden layer

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

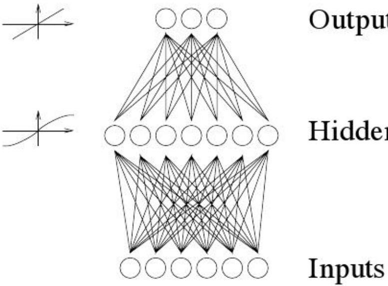
$$x_j^{(1)} = f(z_j^{(1)})$$

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, 2

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

- For layer $i = 1 \ldots \ell$

  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 ($x_1^{(\ell)}$)

assuming scalar output for simplicity

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

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

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

Back-propagation, 3

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

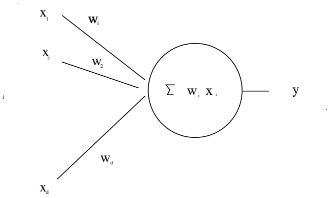
$$\Delta w_{ij}^{(k)} = \alpha e_i^{(k)} x_j^{(k-1)}$$

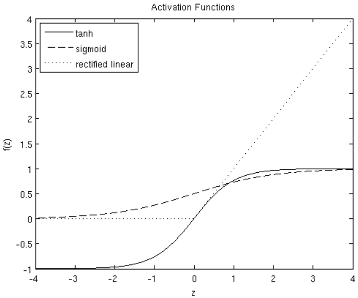
where α is the learning rate < 1.

Adjusting the learning rate is a main issue

Overview

Neuron as a computational node: input, weights, activation function





$$\mathbf{x} \in \mathbb{R}^d \qquad z = \sum_i w_i x_i \qquad f(z) \in \mathbb{R}$$

Activation functions

- Thresholded — 0 if $z < \mathit{threshold}$, 1 otherwise
- Linear — z
- Sigmoid — $1/(1+e^{-z})$
- Tanh — $\frac{e^z - e^{-z}}{e^z + e^{-z}}$
- Radius-based — e^{-z^2/σ^2}
- Rectified linear (ReLU) — $\max(0, z)$

Learning the weights

An optimization problem: Define a criterion

- Supervised learning classification, regression

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

- Reinforcement learning

$$\pi : \text{ State space } \mathbb{R}^d \mapsto \text{ Action space } \mathbb{R}^{d'}$$

Mnih et al., 2015

Main issues

- Requires a differentiable / continuous activation function
- Non convex optimization problem

Properties of NN

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).

Very bad news

- A non convex (and hard) optimization problem
- Lots of local minima
- Low reproducibility of the results

The curse of NNs

Le Cun 2007

- **The NIPS community has suffered of an acute convexivitis 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/eml07_lecun_wia/

Old Key Issues (many still hold)

Model selection

- Selecting number of neurons, connexion graph — **More $\not\Rightarrow$ Better**
- Which learning criterion — avoid overfitting

Algorithmic choices — a difficult optimization problem

- Enforce stability through relaxation

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

- Decrease the learning rate α with time
- Stopping criterion — early stopping

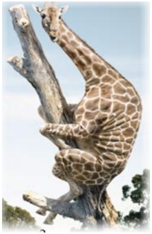
Tricks

- Normalize data
- Initialize **W** small !

Overview

Overview

Toward deeper representations



Invariances matter

- The label of an image is invariant through small translation, homothety, rotation...
- Invariance of labels → Invariance of model

$$y(x) = y(\sigma(x)) \to h(x) = h(\sigma(x))$$

Enforcing invariances

- by augmenting the training set:

$$\mathcal{E} = \{(x_i, y_i)\} \bigcup \{(\sigma(x_i), y_i)\}$$

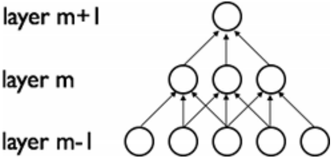
- by structuring the hypothesis space

Convolutional networks

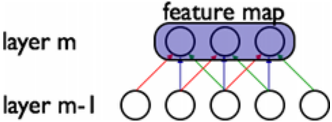
Hubel & Wiesel 1968

Visual cortex of the cat

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



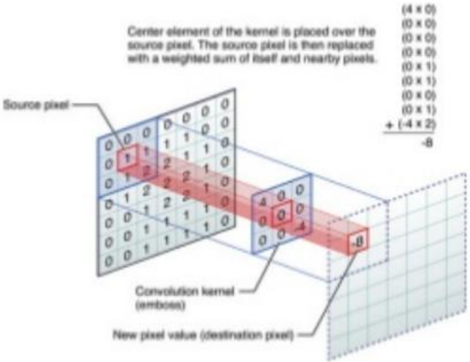
- Layer m: detection of local patterns (same weights)



- Layer $m+1$: non linear aggregation of output of layer m

Ingredients of convolutional networks

1. Local receptive fields (aka kernel or filter)



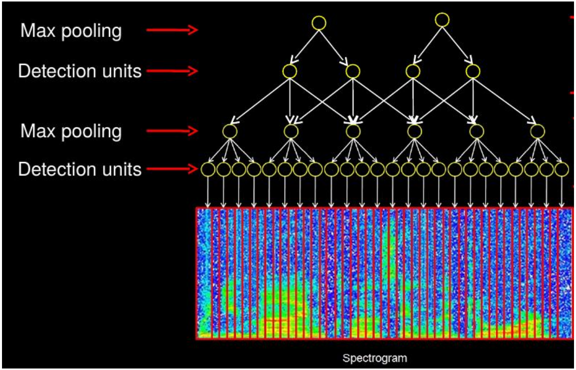
2. Sharing weights

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

Reduces the number of parameters by several orders of magnitude

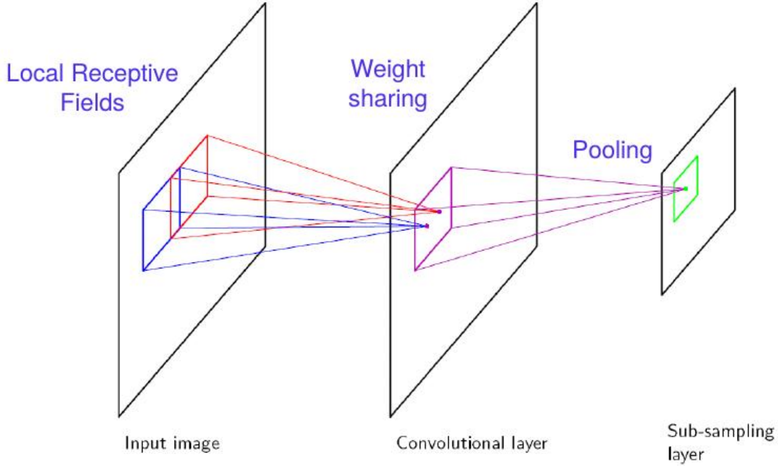
Ingredients of convolutional networks, 2

3. Pooling: reduction and invariance



- Overlapping / non-overlapping regions
- Return the max / the sum of the feature map over the region
- Larger receptive fields (see more of input)

Convolutional networks, summary

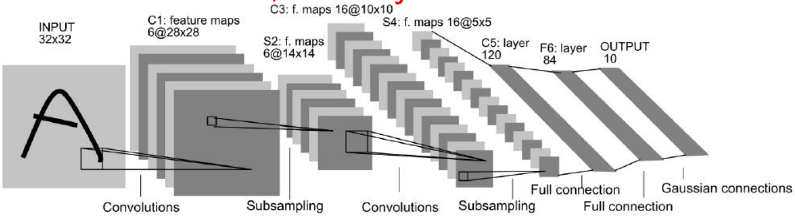


LeCun 1998

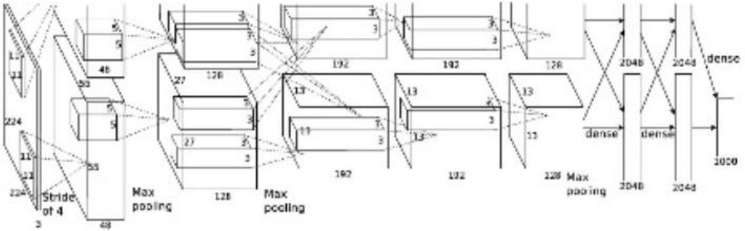
Properties

- Invariance to small transformations (over the region)
- Reducing the number of weights

Convolutional networks, summary



LeCun 1998



Kryzhevsky et al. 2012

Properties

- Invariance to small transformations (over the region)
- Reducing the number of weights
- Usually many convolutional layers

Overview

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton,
Advances in Neural Information Processing Systems 2012

ImageNet

- 15M images
- 22K categories
- Images collected from Web
- Human labelers (Amazons Mechanical Turk crowd-sourcing)
- ImageNet Large Scale Visual Recognition Challenge (ILSVRC-2010)
  - 1K categories
  - 1.2M training images (1000 per category)
  - 50,000 validation images
  - 150,000 testing images
- RGB images with variable-resolution

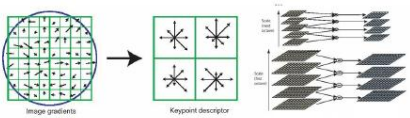
ImageNet

Evaluation

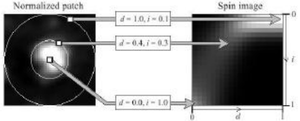
- Guess it right — top-1 error
- Guess the right one among the top 5 — top-5 error

What is new ?

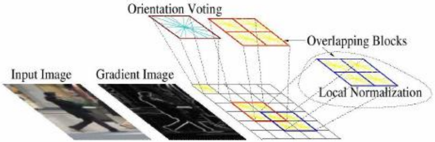
Former state of the art



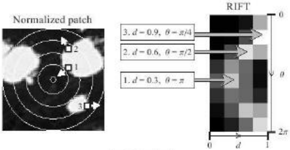
SIFT



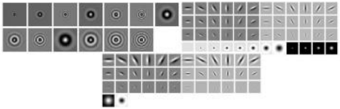
Spin image



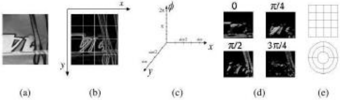
HoG



RIFT



Textons



GLOH

SIFT: scale invariant feature transform
HOG: histogram of oriented gradients
Textons: “vector quantized responses of a linear filter bank”

What is new, 2

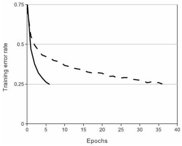
Traditional approach

→ Manually crafted features → Trainable classifier

Deep learning

→ Trainable feature extractor → Trainable classifier

DNN. 1, Tractability



4 layers convolutional

Activation function

- On CIFAR-10: Relu 6 times faster than tanh

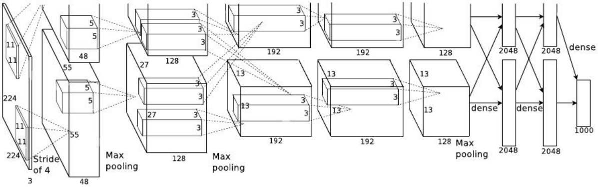
Data augmentation

learn 60 million parameters; 650,000 neurons

- Translation and horizontal symmetries
- Alter RGB intensities
  - PCA, with (p, λ) eigen vector, eigen value
  - Add: $(p_1, p_2, p_3) \times (\alpha\lambda_1, \alpha\lambda_2, \alpha\lambda_3)^t$ to each image, with $\alpha \sim U[0, 1]$

DNN. 2, Architecture

- 1st layer: 96 kernels ($11 \times 11 \times 3$; stride 3)
- Normalized, pooled
- 2nd layer: 256 kernels ($5 \times 5 \times 48$).
- Normalized, pooled
- 3rd layer: 384 kernels ($3 \times 3 \times 256$)
- 4th layer: 384 kernels ($3 \times 3 \times 192$)
- 5th layer: 256 kernels ($3 \times 3 \times 192$)
- followed by 2 fully connexted layers, 4096 neurons each



DNN. 3, Details

Pre-processing

- Variable-resolution images $\rightarrow$ i) down-sampling; ii) rescale
- subtract mean value for each pixel

Results on the test data

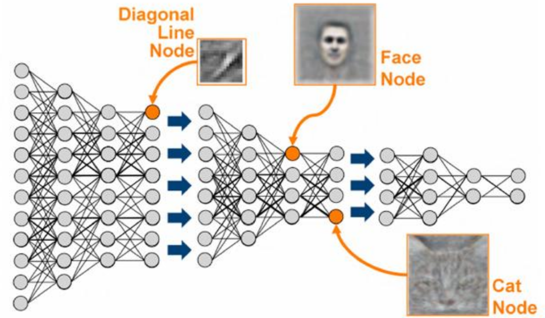
- top-1 error rate: 37.5%
- top-5 error rate: 17.0%

Results on ILSVRC-2012 competition

- 15.3% accuracy
- 2nd best team: **26.2% accuracy**

"Understanding" the result

Interpreting a neuron:
Plotting the input (image) which maximally excites this neuron.



20 millions image from YouTube

"Understanding" the result, 2

Interpreting the representation: Plotting the induced topology

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

Overview

Manifesto for Deep

Bengio, Hinton 2006

1. Grand goal: AI
2. Requisites
   - Computational efficiency
   - Statistical efficiency
   - Prior efficiency: architecture relies on human labor
3. Abstraction is mandatory

Manifesto for Deep

Bengio, Hinton 2006

1. Grand goal: AI

2. Requisites
   - Computational efficiency
   - Statistical efficiency
   - Prior efficiency: architecture relies on **student** labor

3. Abstraction is mandatory

Manifesto for Deep

Bengio, Hinton 2006

1. Grand goal: AI
2. Requisites
   - Computational efficiency
   - Statistical efficiency
   - Prior efficiency: architecture relies on **student** labor
3. Abstraction is mandatory
4. Compositionality principle:

Manifesto for Deep

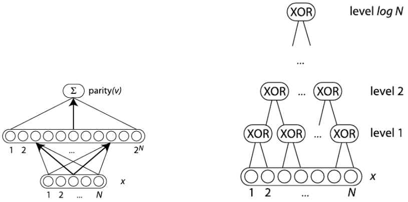
Bengio, Hinton 2006

1. Grand goal: AI

2. Requisites
   - Computational efficiency
   - Statistical efficiency
   - Prior efficiency: architecture relies on **student** labor
3. Abstraction is mandatory
4. Compositionality principle:
   build skills on the top of simpler skills **Piaget**

The importance of being deep

A toy example: *n*-bit parity Hastad 1987



Pros: efficient representation

Deep neural nets are (exponentially) more compact

Cons: poor learning

- More layers $\rightarrow$ more difficult optimization problem
- Getting stuck in poor local optima.

Auto-encoders

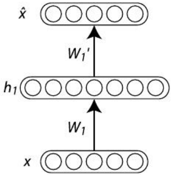
$$\mathcal{E} = \{(\mathbf{x}_i, y_i), x_i \in \mathbb{R}^d, y_i \in \mathbb{R}, i = 1 \ldots n\}$$

First layer

$$\mathbf{x} \longrightarrow h_1 \longrightarrow \hat{\mathbf{x}}$$

- An auto-encoder:

$$\text{Find } W^* = \arg\min_W \left(\sum_i ||W^t o W(\mathbf{x}_i) - x_i||^2 \right)$$



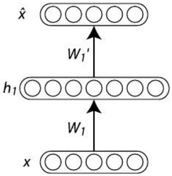
(*) Instead of min squared error, use cross-entropy loss:

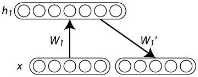
$$\sum_j \mathbf{x}_{i,j} \log \hat{\mathbf{x}}_{i,j} + (1 - \mathbf{x}_{i,j}) \log (1 - \hat{x}_{i,j})$$

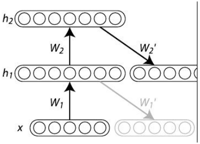
Auto-encoders, 2

First layer $\mathbf{x} \longrightarrow \mathbf{h}_1 \longrightarrow \hat{\mathbf{x}}$

Second layer $\mathbf{h}_1 \longrightarrow \mathbf{h}_2 \longrightarrow \hat{\mathbf{h}_1}$ same, replacing $\mathbf{x}$ with $\mathbf{h}_1$







Discussion

Layerwise training

- Less complex optimization problem (compared to training all layers simultaneously)
- Requires a local criterion: e.g. reconstruction
- Ensures that layer i encodes same information as layer $i + 1$
- But in a more abstract way:
  layer 1 encodes the patterns formed by the (descriptive) features
  layer 2 encodes the patterns formed by the activation of the previous patterns
- When to stop ? trial and error.

Discussion

Layerwise training

- Less complex optimization problem (compared to training all layers simultaneously)
- Requires a local criterion: e.g. reconstruction
- Ensures that layer i encodes same information as layer $i+1$
- But in a more abstract way:
  layer 1 encodes the patterns formed by the (descriptive) features
  layer 2 encodes the patterns formed by the activation of the previous patterns
- When to stop ? trial and error.

Now pre-training is almost obsolete

Gradient problems better understood

- Initialization
- New activation ReLU
- Regularization
- Mooore data
- Better optimization algorithms

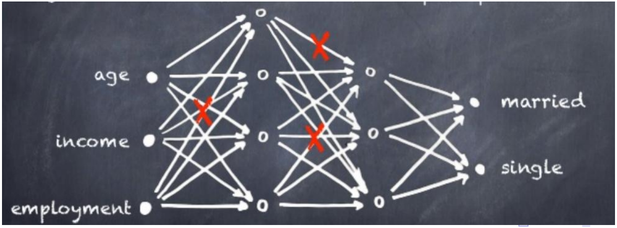
Dropout

Why

- Ensemble learning is effective
- But training several Deep NN is too costly
- The many neurons in a large DNN can "form coalitions".
- Not robust !

How

- During training
  - For each hidden neuron, each sample, each iteration
  - For each input (of this hidden neuron)
  - with probability p (.5), zero the input
  - (double the # iterations needed to converge)
- During validation/test
  - use all input
  - rescale the sum ($\times p$) to preserve average



Recommendations

Ingredients

as of 2015

- ReLU non-linearities
- Cross-entropy loss for classification
- Stochastic Gradient Descent on minibatches
- Shuffle the training samples
- Normalize the input variables (zero mean, unit variance)
- If you cannot overfit, increase the model size; if you can, regularize.

Regularization

- L_2 penalizes large weights
- L_1 penalizes non-zero weights

Adaptive learning rate

- adjusted per neuron to fit the moving average of the last gradients

Hyper-parameters

- Grid search
- Continue training the most promising model

More: Neural Networks, Tricks of the Trade (2012 edition) G. Montavon, G. B. Orr, and K-R Mller eds.

Not covered

- Long Short Term Memory
- Restricted Boltzman Machines
- Natural gradient