Machine Learning
From Theory to Practice
Feature Design
Text and Image Mining

E. Le Pennec

ÉCOLE POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

Outline

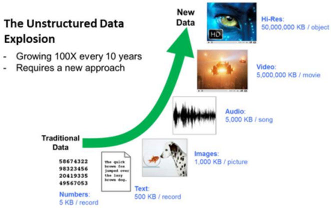
1. Introduction

2. Feature Design
   - Renormalization
   - Basis and Dictionary Learning
   - Categorical Feature Encoding
   - Quantization and Binarization
   - Hashing
   - Pooling

3. Text and Image
   - Text and Bag of Words
   - Image and SIFT
   - Word and Word Vectors
   - Image and Convolutional Networks
   - Text and Recurrent Neural Networks

Outline

Which Description ?



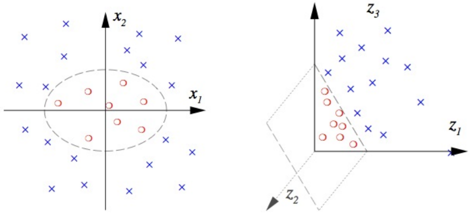
Natural Description

- **Text :** sequence of words.
- **Sound :** waveform.
- **Image :** matrix of pixels.
- **Video :** sequence of images.

- How to construct a *vector* representation that can be fed into a classical learning algorithm ?
- Often dimension reduction issue...
- Focus on Text and Image but...

Feature Design !

$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$$



- Description always matters !

Feature Design

- Art of finding a good description.
- Tremendous impact on performance...

- Example : (SVM) kernel trick !

Outline

Feature

Feature Types

- Quantitative feature :
  - univariate, multivariate, *functional*
  - continuous, discrete
- Categorical feature :
  - Binary, nominal, ordinal
- List, relationship...

Outline

Single Quantitative Feature Transform

- **Idea :** a good prediction algorithm should be invariant to change of scale in the variables.

Renormalization

- centering and standardization of a feature x,
- mapping of x to $[0, 1]$ if max and min are known

- Renormalization is *useless for purely linear methods*... but few methods are purely linear !
- **Idea :** Linear scale may not be the most natural one...

Transformation

- log-scale instead of linear scale,
- Box-Cox transform,
- sigmoid, maxout, rectifier...
- application dependent transformation.

Grouped Quantitative Feature Transform

- **Idea :** a good prediction algorithm should be invariant to the number of variables by group of features.

Grouped Renormalization

- Renormalization of each group of features $x_{i_1}, \ldots, x_{i_K}$ so that each group has the same total energy.

- Renormalization is *useless for purely linear methods*... but few methods are purely linear !

Outline

Quantitative Feature Expansion

- **Idea :** the behavior may not be *linear* in the feature.

Basis decomposition

- Replace a feature x by $\varphi(x) = (\varphi_1(x), \ldots, \varphi_B(x))$ where $(\varphi_b)_{b=1}^{B}$ is a linearly independent family of functions.
- In linear methods, use $\langle \alpha, \varphi(x) \rangle$ instead of αx
- Examples : polynomials, Fourier, wavelets...

- Non parametric estimation technique...

Quantitative Feature Expansion

- **Extension :** the behavior may not be *linear* in some features.

Basis decomposition

- Replace some features $x = (x_1, \dots, x_d)$ by $\varphi(x) = (\varphi_1(x), \dots, \varphi_B(x))$ where $(\varphi_b)_{b=1}^{B}$ is a linearly independent family of functions.
- In linear methods, use $\langle \alpha, \varphi(x) \rangle$ instead of $\langle \alpha, x \rangle$
- Positive definite kernel associated to the scalar product $K(x, y) = \langle \varphi(x), \varphi(y) \rangle$ plays an important role.
- φ is often defined implicitly by a choice of K...

- **Kernel trick :** SVM but also (generalized) linear model as seen for example with *quadratic* logistic modeling.

Quantitative Feature Recombination

- **Idea :** some combinations of the features may be more interesting than any single one.

Decomposition idea

- Obtain $(x_1, \ldots, x_d)$ as a linear combination of some vectors $\varphi_1, \ldots, \varphi_{d'}$ of dimension d'

$$\begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix} \sim \sum_{k'=1}^{d'} x'_{k'} \varphi_{k'}$$

- Use $(x'_1, \ldots, x'_{d'})$ instead of $(x_1, \ldots, x_d)$

- Most important part : choice of φ !

Quantitative Feature Recombination

PCA

- $\varphi_1, \dots, \varphi_d$ are the eigenvectors of the empirical covariance matrix of $(x_1, \dots, x_d)$
- The φ_k are obtained by minimizing recursively :

$$\varphi_k = \operatorname*{argmin}_{\varphi} \frac{1}{n} \sum_{i=1}^{n} \min_{x' \in \mathbf{R}^k} \|(x_{1,i} \dots, x_{d,i})^t - (\sum_{k'=1}^{k-1} x'_{k',i} \varphi_{k'} + x'_{k,i} \varphi)\|^2$$

- The coefficents $x'_{k'}$ are obtained for a new feature by minimizing

$$\|(x_1 \dots, x_d)^t - (\sum_{k=1}^{d'} x'_k \varphi_k)\|^2$$

- Change of basis ! Useless for purely linear method if all the coefficients are kept !

Quantitative Feature Recombination

Dictionary learning (Enforce sparsity)

- Creation of a sparse representation for $x = (x_1, \ldots, x_K)$, i.e. a representation in which most of the weights are 0
- Non trivial minimization problem to find φ_k :

$$(\varphi_1, \ldots, \varphi_{K'}) = \operatorname*{argmin}_{(\varphi_1,\ldots,\varphi_{K'})} \frac{1}{n} \sum_{i=1}^{n} \min_{x' \in \mathbf{R}^{K'}} \|(x_{1,n} \ldots, x_{K,n})^t - (\sum_{k'=1}^{K'} x'_{k'} \varphi_{k'})\|^2 + \lambda \|x'\|_1$$

- The coefficients $x'_{k'}$ are obtained for a new feature by minimizing over $x' \in \mathbf{R}^{K'}$)

$$\|(x_1 \ldots, x_K)^t - (\sum_{k'=1}^{K'} x'_{k'} \varphi_{k'})\|^2 + \lambda \|x'|_1$$

- Non linear transform !
- No simple choice for λ !

Quantitative Feature Recombination

Dimension *Reduction*

- Creation of a vector $x' = (x'_1, \ldots, x'_{d'})$ such that
$$x = (x_1, \ldots, x_d) \leftrightarrow x' = (x'_1, \ldots, x'_{d'})$$
- Criterion based on
  - quality of reconstruction (PCA/SVD extension)
  - distance preservation (MDS extension)

- Non linear transform !
- Freedom on the constrains...
- More general embedding ideas.

Outline

Categorical Feature Encoding

- **Idea :** transform a categorical feature x in *quantitative* ones c (encoding)...

Encodings

- Binary encoding : $x \in \{C_0, C_1\}$
  - binary code : $c(x) = \mathbf{1}_{x=C_1}$
  - symmetrized version : $c(x) = 2\mathbf{1}_{x=C_1} - 1$
- Nominal variable : $x \in \{C_1, \ldots, C_V\}$
  - binary code : $c(x) = (\mathbf{1}_{x=C_2}, \ldots, \mathbf{1}_{x=C_V})$
  - $V - 1$ quantitative variables.
- Ordinal variables : $x \in \{C_1, \ldots, C_V\}$
  - binary code : $c(x) = (\mathbf{1}_{x \geq C_2}, \ldots, \mathbf{1}_{x \geq C_V})$
  - $V - 1$ quantitative variables.
  - Feature selection makes more sense with those feature.

Categorical Feature Encoding

General coding scheme

- Matrix representation :

$$c(x)^t = M \begin{pmatrix} \mathbf{1}_{x=C_0} \\ \vdots \\ \mathbf{1}_{x=C_V} \end{pmatrix}$$

  with $M \in \mathcal{M}_{V-1,V}$ chosen such that its column are linearly independent.

- Examples for $V = 2$:

$$M = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad M = \begin{pmatrix} -1 & 1 & -1 \\ -1 & -1 & 1 \end{pmatrix}$$

$$M = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \qquad M = \begin{pmatrix} -1/3 & 2/3 & -1/3 \\ -1/3 & -1/3 & 2/3 \end{pmatrix}, \ldots$$

- Choice of the coding matters for feature selection

Categorical Feature Interaction

- **Idea :** Capture interaction between features by encoding them jointly.

Full interaction between $x = (x_1, \ldots, x_K)$

- Matrix encoding :

$$c(x)^t = M \begin{pmatrix} \mathbf{1}_{x=(C_{1,1},\ldots,C_{1,K-1},C_{1,K})} \\ \mathbf{1}_{x=(C_{1,1},\ldots,C_{1,K-1},C_{V_K,K})} \\ \mathbf{1}_{x=(C_{1,1},\ldots,C_{2,K-1},C_{1,K})} \\ \vdots \\ \mathbf{1}_{x=(C_{V_1,1},\ldots,C_{V_{K-1},K-1},C_{V_K,K})} \end{pmatrix}$$

with $M \in \mathcal{M}_{\prod_{k=1}^K V_k - 1, \prod_{k=1}^K V_k}$ such that the lines are linearly independent...

Categorical Feature Interaction

- **Example** of code for $x = (x_1, x_2)$ with respectively 2 and 3 categories :

$$c(x)^t = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{1}_{x=(1,1)} \\ \mathbf{1}_{x=(1,2)} \\ \mathbf{1}_{x=(1,3)} \\ \mathbf{1}_{x=(2,1)} \\ \mathbf{1}_{x=(2,2)} \\ \mathbf{1}_{x=(2,3)} \end{pmatrix}$$

- Interaction of order K leads to a very large number of categories !

Categorical Feature Interaction

- Modified coding scheme to impose a **hierarchical structure** :

$$c(x)^t = M \begin{pmatrix} \mathbf{1}_{x_1 = c_{1,1}} \\ \vdots \\ \mathbf{1}_{x_K = c_{V_K,K}} \\ \mathbf{1}_{(x_1,x_2)=(c_{1,1},c_{1,2})} \\ \vdots \\ \mathbf{1}_{(x_{K-1},x_K)=(c_{V_{K-1},K-1},c_{V_K,K})} \\ \vdots \\ \vdots \\ \mathbf{1}_{x=(c_{1,1},\dots,c_{1,K-1},c_{1,K})} \\ \vdots \\ \mathbf{1}_{x=(c_{V_1,1},\dots,c_{V_{K-1},K-1},c_{V_K,K})} \end{pmatrix}$$

with $M \in \mathcal{M}_{\prod V_k - 1, \sum_{k=1}^K \sum_{i_1 < \dots < i_{k'} < \dots < i_k} \prod V_{i_{k'}}}$ such that :

- its lines are linearly independent
- the first $(\sum_{i_1 < \dots < i_{k'} < \dots < i'_K} \prod V_{i_{k'}}) - 1$ below are equal to zeros after the column $\sum_{k=1}^K \sum_{i_1 < \dots < i_k} \prod V_{i_k}$

Categorical Feature Interaction

- **Example** with $x = (x_1, x_2)$ with respectively 2 and 3 categories

$$c(x)^t = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{1}_{x_1=1} \\ \mathbf{1}_{x_1=2} \\ \mathbf{1}_{x_2=1} \\ \mathbf{1}_{x_2=2} \\ \mathbf{1}_{x_3=3} \\ \mathbf{1}_{x=(1,1)} \\ \mathbf{1}_{x=(1,2)} \\ \mathbf{1}_{x=(1,3)} \\ \mathbf{1}_{x=(2,1)} \\ \mathbf{1}_{x=(2,2)} \\ \mathbf{1}_{x=(2,3)} \end{pmatrix}$$

- Restriction possible to a given order of interaction by using only the first lines !
- Coding variant for ordered categories.
- **Extension** to interaction between a quantitative feature x_1 and a categorical feature x_2 through the mapping

$$(x_1, x_2) \to (x_1, x_1 c(x_2))$$

MFA Renormalization

Categorical value code for a single variable

- Classical redundant dummy coding :

$$\mathbf{X} \in \{1, \ldots, V\} \mapsto P(\mathbf{X}) = (\mathbf{1}_{\mathbf{X}=1}, \ldots, \mathbf{1}_{\mathbf{X}=V})^t$$

- Compute the mean (i.e. the empirical proportion) $\overline{P} = \frac{1}{n} P(\mathbf{X})$
- *Renormalize* $P(\mathbf{X})$ by $1/\sqrt{(V-1)\overline{P}}$:

$$P(\mathbf{X}) \mapsto P^r(\mathbf{X})$$

$$(\mathbf{1}_{\mathbf{X}=1}, \ldots \mathbf{1}_{\mathbf{X}=V}) \mapsto \left(\frac{\mathbf{1}_{\mathbf{X}=1}}{\sqrt{(V-1)\overline{P}_1}}, \ldots, \frac{\mathbf{1}_{\mathbf{X}=V}}{\sqrt{(V-1)\overline{P}_V}} \right)$$

- χ^2 type distance !

Outline

Quantization and Binarization

- **Idea :** Going from quantitative feature to binary features...
- Used in practice mainly for computing reason.

Quantization/Binarization strategy

- Construct a finite size quantifier for the feature.
- Code the feature by a binary code of its quantized version.

- Construction of a quantifier :
  - Binning : use of a (regular) histogram with V
  - V-means : use the centers as *keywords* and assign a point to its nearest keyword.
  - Use $V - 1$ quantiles
  - Vector coding...
- Extreme case : $V = 2$ binarization !

Outline

Hashing

- **Idea :** Reduce the number of values of a nominal feature with categories in a large set $\mathcal{D}$?

Hashing

- Construction of a *hashing* function $H : \mathcal{D} \to \{1, \ldots, V\}$ and use the hashed value instead of the original one.
- The hashing function should be *as injective as possible...* in a probabilistic sense..

- Design of such hashing function is a real art !
- One can sometimes find hashing function such that $d(H(C_k), H(C_{k'})) \sim d'(C_k, C_{k'})$...

Outline

Pooling

- **Idea :** Group features in a non linear way to reduce the number of features.

Pooling

- **Quantitative features :** replace a subset or a list by
  - its min/max,
  - its range,
  - its average (linear...),
  - ...
- **Nominal features :** replace a subset or a list by
  - its histogram
  - its most frequent values
  - ...

- Discretization/Binarization can be use before and after...

Outline

Outline

Text and Bag of Words

- How to transform a **text** into a vector of **categorical features** ?

Bag of Words strategy

- Make a *list* of words,
- Compute a *weight* for each words.

List buiding

- Make a list of all used words with their number of occurrence
- Gather the words in the list having the same stem (stemming)
- Hash the stem using a word specific hashing function (MurmurHash with 32bits for instance)
- Compute the histogram $h_w(d)$

Text and Bag of Words

Weight computation

- Compute the histogram $h_w(d)$
- Apply a renormalization :
  - tf transform (word profile) :
    $$\mathrm{tf}-\mathrm{idf}_w(d) = \mathrm{tf}_w(d)$$
    with $\mathrm{tf}_w(d)$ the frequency within the document d
    $$\mathrm{tf}_w(d) = \frac{h_w(d)}{\sum_w h_w(d)}.$$
  - tf-idf transform (word profile weighted by rarity) :
    $$\mathrm{tf}-\mathrm{idf}_w(d) = \mathrm{idf}_w \times \mathrm{tf}_w(d)$$
    with idf a corpus dependent weight
    $$\mathrm{idf}_w = \log \frac{n}{\sum_{i=1}^n \mathbf{1}_{h_w(d_i)\neq 0}}$$

- Use the vector $\mathrm{tf}(d)$ (or $\mathrm{tf}-\mathrm{idf}(d)$) to describe a document.
- Most classical text preprocessing !

Okapi BM25 for Text Retrieval

Okapi BM25

- Representation (smoothed tf-idf) :
$$\mathrm{bm25}_w(d) = \mathrm{idf}_w \times \frac{(k_1+1)\mathrm{tf}_w(d)}{k_1 + \mathrm{tf}_w(d)}$$
- Match quality for a set of words Q measured by a simple scalar product :
$$\mathrm{BM25}(d, Q) = \sum_{w \in Q} \mathrm{bm25}_w(d)$$

- Extensively used in text retrieval.
- Can be traced back to 1976 !

Unsupervised Text Clustering

Probabilistic latent semantic analysis (PLSA)

- Model :

$$\mathbb{P}\{\mathrm{tf}\} = \sum_{k=1}^{K} \mathbb{P}\{k\}\,\mathbb{P}\{\mathrm{tf}|k\}$$

  with k the (hidden) topic, $\mathbb{P}\{k\}$ a topic probability and $\mathbb{P}\{\mathrm{tf}|k\}$ a multinomial law for a given topic.
- Clustering according to a mixture model

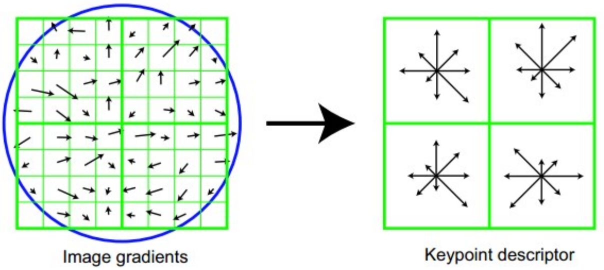
$$\mathbb{P}\{k|\mathrm{tf}\} = \frac{\widehat{\mathbb{P}\{k\}}\widehat{\mathbb{P}\{\mathrm{tf}|k\}}}{\sum_{k'} \widehat{\mathbb{P}\{k'\}}\widehat{\mathbb{P}\{\mathrm{tf}|k'\}}}$$

- Same idea than GMM !
- Bayesian variant called LDA.

Outline

SIFT

- How to transform an **image** into a vector of **features** ?



SIFT Strategy

- Compute a local descriptor based on local gradient.
- Agregate those measurements by histograms.

SIFT Local Descriptor

- Compute a local scale and a principal orientation
- Compute the gradient at that scale, its norm and its angle with the principal direction
- Quantize this angle with 16 different values (binning)
- On each subwindow of 4×4 subwindow grid oriented with the principal and of size the local scale, compute the sum of the gradient norm for each angle bin renormalized by the number of points in the subwindow.
- Use the $4 \times 4 \times 16$ values as the local descriptor.

SIFT

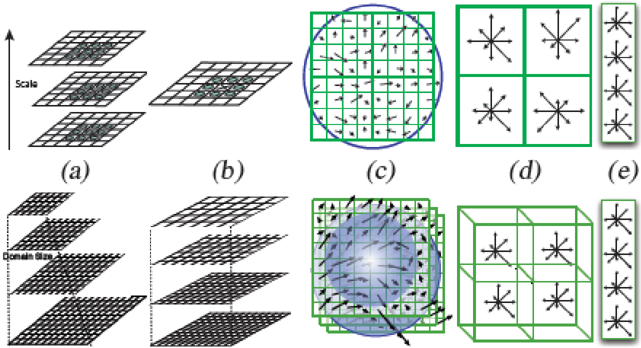
SIFT based representation

- Compute the SIFT descriptor at each point of a regular grid
- Assign each SIFT descriptor to the closest *keyword* obtained by K-means on the whole corpus (typically with $K \sim 2000$)
- Compute a normalized histogram of the counts of those keywords
- Use this 2000 values as the image descriptor

- Classical algorithm with those new features !
- State of the Art in 2005 !
- Better results with CNN.

SIFT

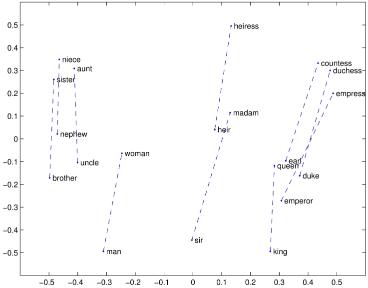
- Enhancement with further scaling/pooling



- Competitive again with respect to CNN !

Outline

Word Vectors



Word Embedding

- Map from the set of words to $\mathbb{R}^d$.
- Each word is associated to a vector.
- Hope that the relationship between two vectors is related to the relationship between the corresponding words !

Word And Context

Look ! $\boxed{\text{A}}$ $\boxed{\text{single}}$ $\boxed{\text{word}}$ $\boxed{\text{and}}$ $\boxed{\text{its}}$ context

Word and Context

- **Idea :** characterize a word w through its relation with its context c...
- **Probabilistic description** :
  - Joint distribution : $f(w, c) = \mathbb{P}\{w, c\}$
  - Conditional distribution(s) : $f(w, c) = \mathbb{P}\{w|c\}$ or $f(w, c) = \mathbb{P}\{c|w\}$.
  - Pointwise mutual information : $f(w, c) = \mathbb{P}\{w, c\} / (\mathbb{P}\{w\} \mathbb{P}\{c\})$
- Word w characterized by the vector $C_w = (f(w, c))_c$ or $C_w = (\log f(w, c))_c$.

- In practice, C is replaced by an estimate on large corpus.
- Very high dimensional model !

A (Naive) SVD Approach

$$\underset{(n_w \times n_c)}{\mathbf{C}} \simeq \underset{(n_w \times r)}{\mathbf{U_r}} \; \underset{(r \times r)}{\Sigma_{r,r}} \; \underset{(r \times n_c)}{\mathbf{V_r^t}}$$

Truncated SVD Approach

- Approximate the code matrix C using the truncated SVD decomposition (best low rank approximation).
- Use as a code

$$C'_w = U_{r,w} \Sigma^{\alpha}_{r,r}$$

with $\alpha \in [0, 1]$.

- Variation possible on C.
- State of the art results but computationally intensive...

A Least Square Approach

- All the previous models corresponds to
$$-log\mathbb{P}\left\{w,c\right\} \sim \langle C'_w, C''_c\rangle + \alpha_w + \beta_c$$

GloVe (Global Vectors)

- Enforce such a fit through a (weighted) least square formulation :
$$\sum_{w,c} h(\mathbb{P}\left\{w,c\right\}) \left\| -log\mathbb{P}\left\{w,c\right\} - \left(\langle C'_w, C''_c\rangle + \alpha_w + \beta_c\right) \right\|^2$$
with h a increasing weight.
- Minimization by alternating least square...

- Much more efficient than SVD.

A Learning Approach

Supervised Learning Formulation

- Couples (w, c) are positive examples.
- Artificially generate negative examples (w', c') (for instance by copying c and generating w' indepently of c.)
- Model the probability of being positive given (w, c) as a (simple) function of the codes C'_w and C''_c

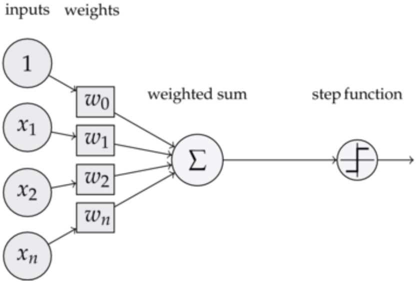
- Word2vec : logistic modeling

$$\mathbb{P}\left\{1|w,c\right\} = \frac{e^{\langle C'_W, C''_c \rangle}}{1 + e^{\langle C'_W, C''_c \rangle}}$$

- State of the art and efficient computation.
- Similar to a factorization of $-\log(\mathbb{P}\left\{w,c\right\}/(\mathbb{P}\left\{w\right\}\mathbb{P}\left\{c\right\}))$!

Outline

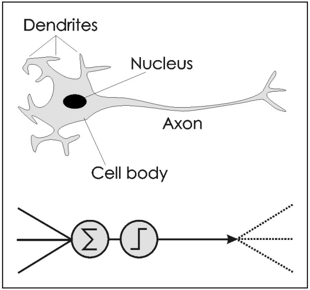
Perceptron



Perceptron (Rosenblatt 1957)

- Inspired from biology.
- Very simple (linear) model !
- Physical implementation and proof of concept.

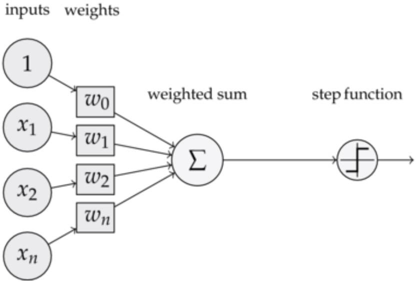
Perceptron



Perceptron (Rosenblatt 1957)

- Inspired from biology.
- Very simple (linear) model !
- Physical implementation and proof of concept.

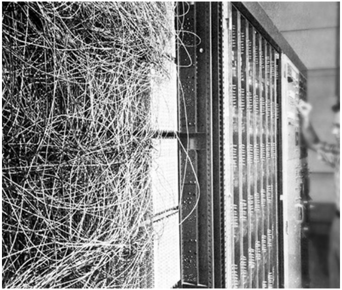
Perceptron



Perceptron (Rosenblatt 1957)

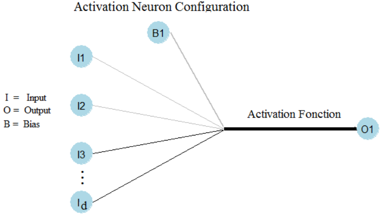
- Inspired from biology.
- Very simple (linear) model !
- Physical implementation and proof of concept.

Perceptron



Perceptron (Rosenblatt 1957)

- Inspired from biology.
- Very simple (linear) model !
- Physical implementation and proof of concept.



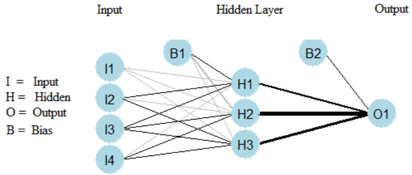
Artificial neuron

- Structure :
  - Mix inputs with a weighted sum,
  - Apply a (non linear) activation function to this sum,
  - Eventually threshold the result to make a decision.
- Weights learned by minimizing a loss function.

Logistic unit

- Structure :
  - Mix inputs with a weighted sum,
  - Apply the logistic function $\sigma(t) = e^t/(1+e^t)$,
  - Threshold at $1/2$ to make a decision !
- Logistic weights learned by minimizing the -log-likelihood.

Multilayer Perceptron



MLP (Rumelhart, McClelland, Hinton - 1986)

- Multilayer Perceptron : cascade of layers of artifical neuron units.
- **Backprop** algorithm to optimize it !

- Construction of a function by composing simple units.
- MLP corresponds to a specific direct acyclic graph structure.
- Non convex optimization problem !

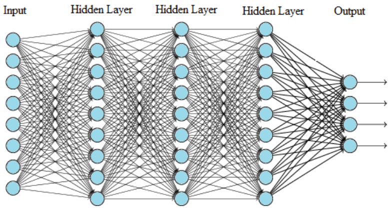
Universal Approximation Theorem

Universal Approximation Theorem (Hornik, 1991)

- A single hidden layer neural network with a linear output unit can approximate any continuous function arbitrarily well, given enough hidden units

- Valid for most activations functions.
- A single hidden layer is sufficient but more may be more efficient.
- No bounds on the number of required units... (Asymptotic flavour)

Deep Neural Network

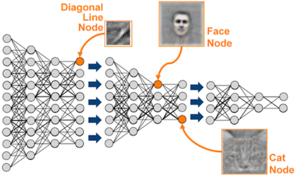


Deep Neural Network structure

- Deep cascade of layers !

- No conceptual novelty optimization becomes a crucial issue.
- Clever initialization, clever stochastic optimization, clever use of GPU...
- Very impressive results !

Deep Learning



Family of Machine Learning algorithm combining :

- a (deep) multilayered structure,
- a clever initalization,
- a clever stochastic fine tuning optimization.

- Examples : Deep Neural Network, Deep (Restricted) Boltzman Machine, Stacked Encoder...
- Appears to be very efficient but lack of theoretical fundation !

Convolutional Network

PROC. OF THE IEEE, NOVEMBER 1998 7

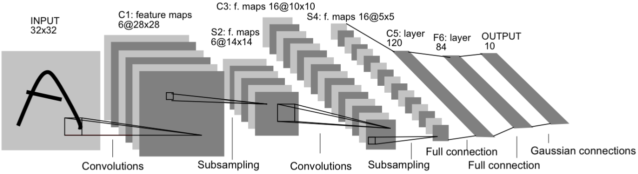
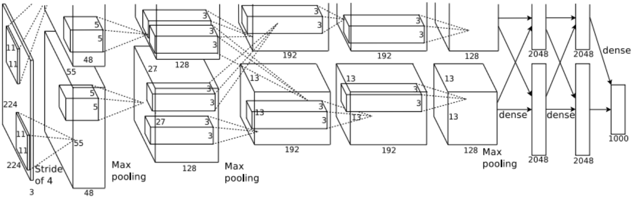


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

- 1989 : 6 Hidden layer architecture (Yann Lecun)
- Drastic reduction of the number of parameters through a translation invariance principle (convolution)
- Requires 3 days of training for 60 000 examples !
- Tremendous improvement.
- Representation learned through the task.

Deep Convolutional Networks



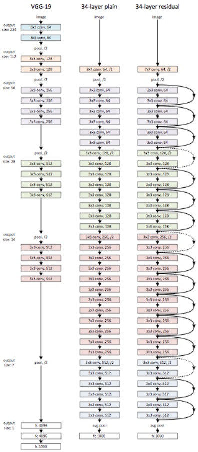
- 2012 : Alexnet (A. Krizhevsky, I. Sutskever, G. Hinton)
- Bigger layers and thus more parameters.
- Clever intialization scheme, RELU, Renormalization and use of GPU.
- 6 days of training for 1.2 millions images.

Deep Convolutional Networks

[1]Inception 5 (GoogLeNet)

Inception 7a

[1]Going Deeper with Convolutions, [C. Szegedy et al, CVPR 2015]



- Deeper and deeper networks ! (GoogLeNet / Residual Neural Network)

Outline

Text as Sequences

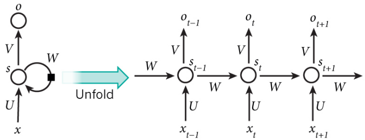
A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed cycle. This creates an internal state of the network which allows it to exhibit dynamic temporal behavior. Unlike feedforward neural networks, RNNs can use their internal memory to process arbitrary sequences of inputs. This makes them applicable to tasks such as unsegmented connected handwriting recognition or speech recognition

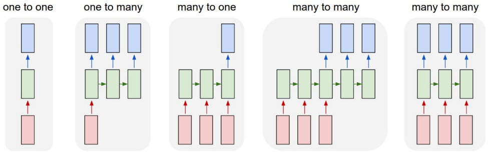
Sequences

- Word = sequence of letters
- Text = sequence of letters/words.

- Capitalize on this structure.

Recurrent Neural Networks





Recurrent Neural Network Unit

- Input seen as a sequence.
- *Simple* computational units with shared weights.
- Information transfer through a context !

- Several architectures !

Text Generation

Text Generation Experiment

- Construct a recurrent network to predict the distribution of the next character from the n previous ones.
- Generate a sample for the learned (Markov) models.

iteration 100

tyntd-iafhatawiaoihrdemot lytdws e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e plia tklrgd t o idoe ns,smtt h ne etie h,hregtrs nigtike,aoaenns lng

Text Generation Experiment

- Construct a recurrent network to predict the distribution of the next character from the n previous ones.
- Generate a sample for the learned (Markov) models.

Text Generation

iteration 300

" Tmont thithey" fomesscerliund Keushey. Thom here sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize.

Text Generation Experiment

- Construct a recurrent network to predict the distribution of the next character from the n previous ones.
- Generate a sample for the learned (Markov) models.

Text Generation

iteration 500

we counter. He stutn co des. His stanted out one ofler that concossions and was to gearang reay Jotrets and with fre colt otf paitt thin wall. Which das stimn

Text Generation Experiment

- Construct a recurrent network to predict the distribution of the next character from the n previous ones.
- Generate a sample for the learned (Markov) models.

Text Generation

iteration 700

Aftair fall unsuch that the hall for Prince Velzonski's that me of her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort how, and Gogition is so overelical and ofter.

Text Generation Experiment

- Construct a recurrent network to predict the distribution of the next character from the n previous ones.
- Generate a sample for the learned (Markov) models.

Text Generation

iteration 1200

"Kite vouch !" he repeated by her door. "But I would be done and quarts, feeling, then, son is people...."

Text Generation Experiment

- Construct a recurrent network to predict the distribution of the next character from the n previous ones.
- Generate a sample for the learned (Markov) models.

Text Generation

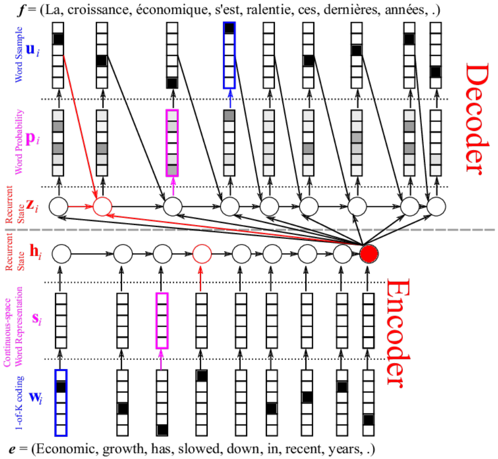
iteration 2000

"Why do what that day," replied Natasha, and wishing to himself the fact the princess, Princess Mary was easier, fed in had oftened him. Pierre aking his soul came to the packs and drove up his father-in-law women.

Text Generation Experiment

- Construct a recurrent network to predict the distribution of the next character from the n previous ones.
- Generate a sample for the learned (Markov) models.

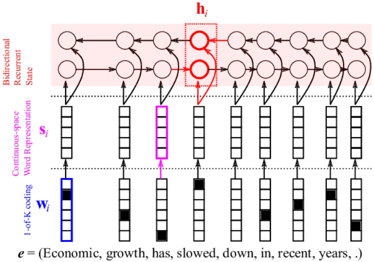
Automatic Translation

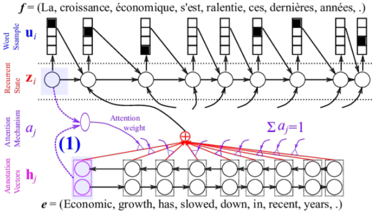


Encoder/Decoder structure

- Word vectors, RNN, stacked structure.

Automatic Translation

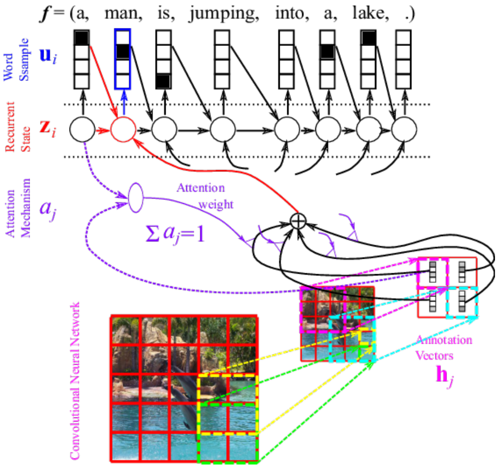




Encoder/Decoder structure

- Much more complex structure : asymmetric, attention order...

Automatic Captioning



Encoder/Decoder structure

- Much more complex structure : asymmetric, attention order...

What's Missing ?

Natural Language Processing

- More (tree) structured parsing.
- Grammar rules / Ontology.
- Stemming...

- A world on his own !