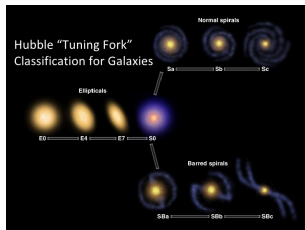


Introduction à l'apprentissage statistique

Stéphane Canu

asi.insa-rouen.fr/enseignants/~scanu

scanu@insa-rouen.fr



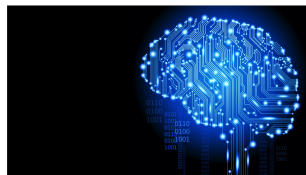
Rencontre GdR ISIS-OG

Questions ouvertes en analyse du signal pour l'astronomie gravitationnelle

8 octobre 2018

Introduction to machine learning

- 1 Introducing example
- 2 Machine learning
 - Definition attempt
 - Hypothesis set and methods
- 3 Deep Learning
- 4 Some practical issues



Machine learning (AI) for SETI



Leonardo Garrido

@LeonardoGarrido

Suivre



Breakthrough Listen is also applying the successful machine learning algorithm to find new kinds of signals that could be coming from extraterrestrial civilizations ...

Traduire le Tweet



AI helps track down mysterious cosmic radio bursts

Using machine learning techniques, Breakthrough Listen researchers have been able to find many more enigmatic fast radio bursts

news.berkeley.edu

17:55 - 14 sept. 2018

Machine learning (AI) for SETI: the paper

arXiv.org > astro-ph > arXiv:1809.03043

Search for ArXiv
Meta | Advanced

Astrophysics > High Energy Astrophysical Phenomena

Fast Radio Burst 121102 Pulse Detection and Periodicity: A Machine Learning Approach

Yunfan Garry Zhang, Vishal Gajjar, Griffin Foster, Andrew Siemion, James Cordes, Casey Law, Yu Wang

(submitted on 9 Sep 2018)

We report the detection of 72 new pulses from the repeating fast radio burst FRB 121102 in Breakthrough Listen C-band (4–8 GHz) observations at the Green Bank Telescope. The new pulses were found with a convolutional neural network in data taken on August 26, 2017, where 21 bursts have been previously detected. Our technique combines neural network detection with dedispersion verification. For the current application we demonstrate its advantage over a traditional brute-force dedispersion algorithm in terms of higher sensitivity, lower false positive rates, and faster computational speed. Together with the 21 previously reported pulses, this observation marks the highest number of FRB 121102 pulses from a single observation, totaling 93 pulses in five hours, including 45 pulses within the first 30 minutes. The number of data points reveal trends in pulse fluence, pulse detection rate, and pulse frequency structure. We introduce a new periodicity search technique, based on the Rayleigh test, to analyze the time of arrivals, with which we exclude with 99% confidence periodicity in time of arrivals with periods larger than 5.1 times the model-dependent time-stamp uncertainty. In particular, we rule out constant periods > 10 ms in the barycentric arrival times, though intrinsic periodicity in the time of emission remains plausible.

Comments: 32 pages, 10 figures

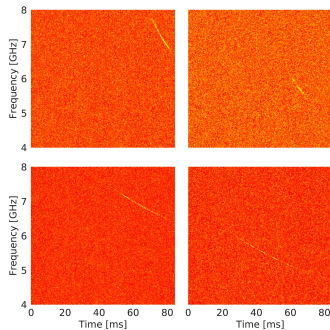
Subjects: High Energy Astrophysical Phenomena (astro-ph.HE), Instrumentation and Methods for Astrophysics (astro-ph.IM)

Cite as: arXiv:1809.03043 [astro-ph.HE]
(or arXiv:1809.03043v1 [astro-ph.HE] for this version)

Submission history

From: Yunfan Zhang G. [view email]

[v1] Sun, 9 Sep 2018 20:50:23 CMT (1371kb)



- August 26, 2017: 21 bursts have been previously detected.
- Machine learning: 72 new detections

Machine learning (AI) for SETI: the machine learning

- Data

- ▶ 5 hours listen observations
- ▶ 400000 images,
- ▶ half of which contain simulated pulses

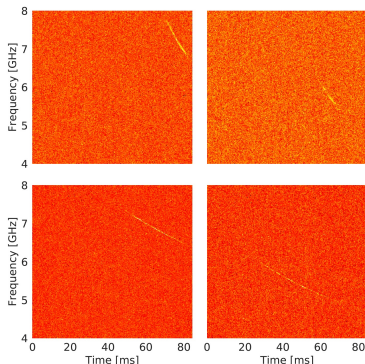
- model

- ▶ inputs: images
- ▶ output: detection (yes/no)
- ▶ convolutional deep network 17 layers
- ▶ 6.2 million trainable parameters

- training

- ▶ training set = 4.5 hours
- ▶ 20 hours on a GPU
- ▶ error converges after 100 epochs to 93%

- combined with dedispersion verification



$$decision = f(image) + noise$$

Plan

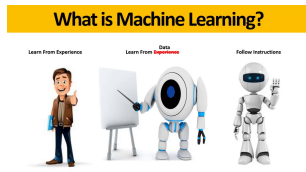
1 Introducing example

2 Machine learning

- Definition attempt
- Hypothesis set and methods

3 Deep Learning

4 Some practical issues



Machine learning

- model

$$decision = f(image) + noise$$

$\mathbb{P}(decision, image)$ unknown .

- goal = cost

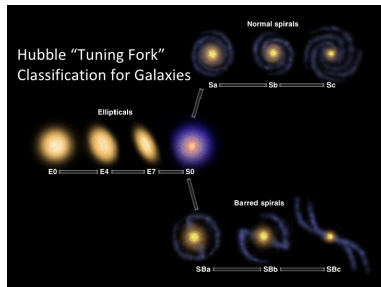
$\hat{f}$ estimate f from data .

minimize the cost of $\hat{f}$.

- issues: induction

statistics + unknown distribution + cost + computational issues .

Exemples de machines capables d'apprendre



tâche	données	performance
prédire le type d'une galaxie	images	taux d'erreur
estimer une fréquence	images	écart quadratique
(découvrir) des types de galaxies	images	entropie de la partition
recommander une ligne de code	programmes	renforcement

d'après <https://www.kaggle.com/c/galaxy-zoo-the-galaxy-challenge>

Les trois (principales) familles d'apprentissage automatique



Apprendre à prédire : apprentissage supervisé

Entrée x sorties y

- $y \in \{0, 1\}$ classification binaire ou multiple

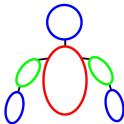


- $y \in \mathbb{R}$ régression



- $y \in \{faible, moyen, fort\}$ ordonnancement

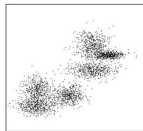
- y structuré



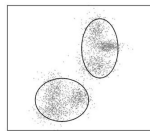
Apprendre à résumer : apprentissage non supervisé

Entrée x seul (pas de y)

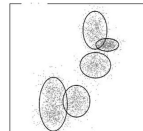
- Catégoriser (clustering) : qui se ressemble s'assemble k-means, CHA



(a) Input data



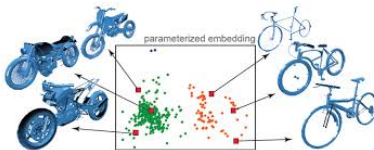
(b) GMM (K=2)



(c) GMM (K=5)

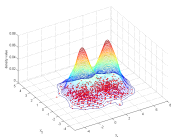
- Représenter: $x \in \mathbb{R}^p \longrightarrow z \in \mathbb{R}^d, d \ll p$

ACP, t-SNE, UMAP

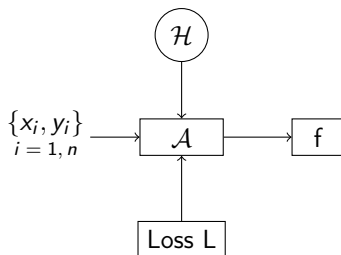


- Estimer une densité $\mathbb{P}(x)$

Parzen,



Apprentissage et ensemble d'hypothèses



apprendre c'est :

sélectionner, parmi des hypothèses, la plus cohérente avec les données

Universalité de l'apprentissage

Résultat d'existence

Théorème d'approximation universel

- soit $\varepsilon > 0$ un réel
- quelle que soit la tâche à apprendre $\forall f^*$
- il existe $\hat{f} \in \mathcal{H}$, telle que

$$\|f^* - \hat{f}\| \leq \varepsilon$$

Choisir un ensemble d'hypothèses $\mathcal{H}$ universel

- plus on a d'exemples, plus le modèle doit pouvoir être complexe
- danger : surapprentissage
- question centrale : adapter la complexité du modèle au problème

Les différents types de modèles universels

- modèles basées sur les variables
 - ▶ la notion de dictionnaire (base, polynômes, ondelettes. . .)

$$\hat{f}(x) = \sum_{k=1}^k \alpha_k \phi_k(x)$$

- modèles basées sur les exemples
 - ▶ plus proches voisins
 - ▶ noyaux (SVM)
- combinaison d'éléments simples (partitions)
 - ▶ les forêts aléatoires
 - ▶ le boosting
- modèles profonds (deep learning)

linéaires vs non linéaires

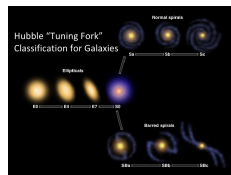
→ C'est un problème de représentation (extraction de caractéristiques)

Machine learning for galaxy morphology prediction

Galaxy Zoo - The Galaxy Challenge at Kaggle

• Data

- ▶ training set with 61,578 images
- ▶ test set of 79,975 images
- ▶ Data augmentation



• model

- ▶ inputs: images
- ▶ output: proba galaxy morphology
- ▶ SVM, GBM,
- ▶ convolutional deep network

• training accuracy

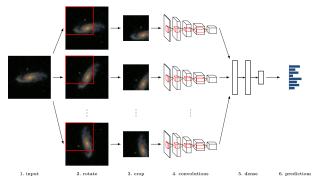
- ▶ SVM 88%
- ▶ Gradient Boosting 90 %
- ▶ deep learning (CNN) 93%

The screenshot shows the Galaxy Zoo - The Galaxy Challenge page on Kaggle. The page title is "Galaxy Zoo - The Galaxy Challenge" and it is sponsored by the Galaxy Zoo project. The challenge is to "Classify the morphologies of distant galaxies in our Universe" and it is worth \$16,000. The page shows the "Public Leaderboard" and "Private Leaderboard" tabs. The "Public Leaderboard" is selected, and it shows a table of the top 8 teams. The table has columns for Rank, Team Name, Kernel, Team Members, Score (0.7491), Entries (43), and Last (5y). The top team is "aedilem" with a score of 0.7491. The second team is "Maxim Miskov" with a score of 0.7752. The third team is "6789" with a score of 0.77889. The fourth team is "simon" with a score of 0.77951. The fifth team is "Julian de Wit" with a score of 0.77952. The sixth team is "Znumbers Zmany" with a score of 0.77963. The seventh team is "Ryan Kaiser" with a score of 0.08072. The eighth team is "Voyager" with a score of 0.08083.

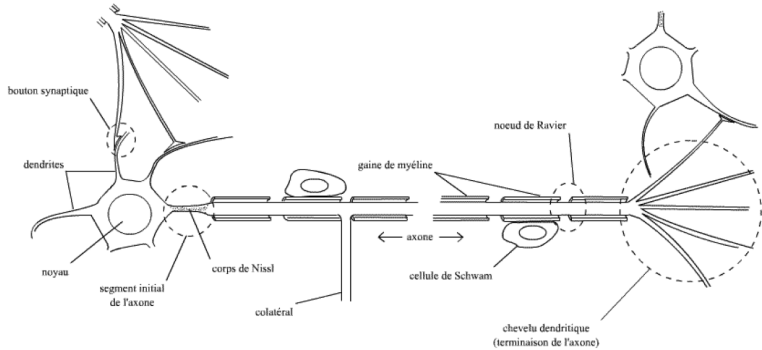
#	Team Name	Kernel	Team Members	Score (0.7491)	Entries (43)	Last (5y)
1	aedilem	Kernel		0.7491	43	5y
2	Maxim Miskov			0.7752	11	5y
3	6789			0.77889	62	5y
4	simon			0.77951	4	5y
5	Julian de Wit			0.77952	19	5y
6	Znumbers Zmany			0.77963	11	5y
7	Ryan Kaiser			0.08072	20	5y
8	Voyager			0.08083	7	5y

Plan

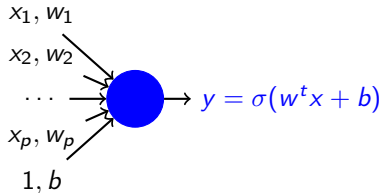
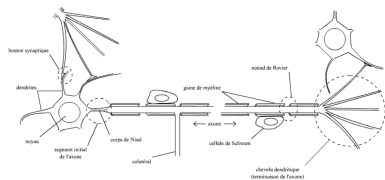
- 1 Introducing example
- 2 Machine learning
 - Definition attempt
 - Hypothesis set and methods
- 3 Deep Learning
- 4 Some practical issues



The biological neuron



McCulloch & Pitts formal neuron 1943



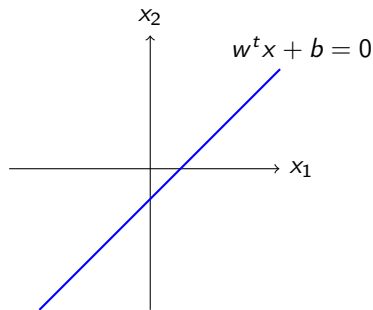
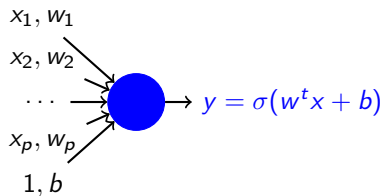
x input $\in \mathbb{R}^p$

w weight, b bias

σ activation function

y output $\in \mathbb{R}$

The artificial neuron as a linear threshold unit



x input $\in \mathbb{R}^p$

w weight, b bias

a activation, $a = w^t x + b$

σ activation function

ϕ transfer function

y output $\in \mathbb{R}$

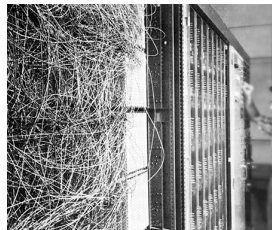
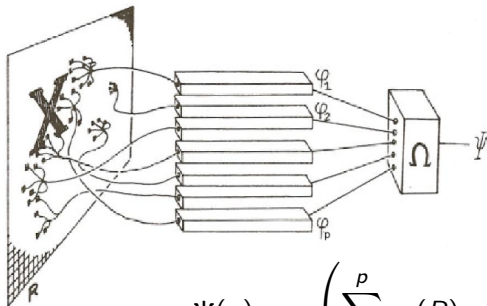
σ activation function (non linear)

$$\begin{array}{ccc} \mathbb{R} & \mapsto & \mathbb{R} \\ a & \rightarrow & y = \sigma(a) \end{array}$$

ϕ transfer function

$$\begin{array}{ccc} \mathbb{R}^p & \mapsto & \mathbb{R} \\ \mathbf{x} & \rightarrow & y = \phi(\mathbf{x}) = \sigma(w^t x + b) \end{array}$$

The formal neuron as a learning machine: fit the w



$$\Psi(x) = \sigma \left(\sum_{j=1}^p \varphi_j(R) w_j + b \right)$$

Rosenblatt's Perceptron, 1958 (Widrow & Hoff's Adaline, 1960)

given n pairs of input-output data $\mathbf{x}_i = \varphi_j(R_i), t_i, i = 1, n$

find w such that

$$\underbrace{\sigma(\mathbf{w}^t \mathbf{x}_i)}_{\text{prediction of the model}} = \underbrace{t_i}_{\text{ground truth}}$$

Cost minimization (energy-based model)

Minimize a loss $\min_{\mathbf{w} \in \mathbb{R}^{p+1}} \sum_{i=1}^n \text{loss}(\mathbf{w}) \quad \text{loss}(\mathbf{w}) = (\sigma(\mathbf{w}^t \mathbf{x}_i) - t_i)^2$

Gradient descent $\mathbf{w} \leftarrow \mathbf{w} - \rho \mathbf{d} \quad \mathbf{d} = \sum_{i=1}^n \nabla_{\mathbf{w}} \text{loss}(\mathbf{w})$

Stochastic gradient $\mathbf{d} = \nabla_{\mathbf{w}} \text{loss}(\mathbf{w})$

Algorithm 1 Gradient epoch

Data: $\mathbf{w}$ initialization, ρ stepsize

Result: $\mathbf{w}$

for $i=1, n$ **do**

$\mathbf{x}_i, t_i \leftarrow$ pick a point i

$\mathbf{d} \leftarrow \mathbf{d} + \nabla_{\mathbf{w}} \text{loss}(\mathbf{w}, \mathbf{x}_i, t_i)$

end

$\mathbf{w} \leftarrow \mathbf{w} - \rho \mathbf{d}$

Algorithm 2 Stochastic gradient

Data: $\mathbf{w}$ initialization, ρ stepsize

Result: $\mathbf{w}$

for $i=1, n$ **do**

$\mathbf{x}_i, t_i \leftarrow$ pick a point i

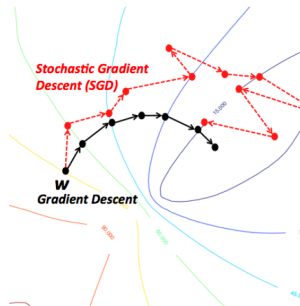
$\mathbf{d} \leftarrow \nabla_{\mathbf{w}} \text{loss}(\mathbf{w}, \mathbf{x}_i, t_i)$

$\mathbf{w} \leftarrow \mathbf{w} - \rho \mathbf{d}$

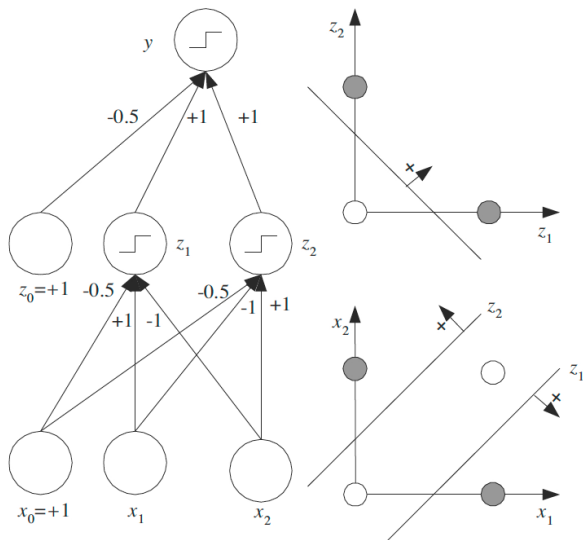
end

Accelerating the stochastic gradient

- stochastic average (mini batch)
 - ▶ parameters (Polyak and Juditsky, 1992)
 - ▶ gradients SAG-A, (Le Roux et al 2012)
 - ▶ variance reduction (Johnson, Zhang, 2013)
- convergence acceleration
 - ▶ Nesterov's method (1983)
 - ▶ momentum (heuristic)
- acceleration and averaging
 - ▶ (Dieuleveut, Flammarion & Bach, 2016)
- stepsize adaptation
 - ▶ RMSprop (Tieleman & Hinton, 2012)
 - ▶ Adaptive Moment Estimation – ADAM (Kingma & Ba, 2015)
 - ▶ AMSGRAD (Reddi et al, BPA ICRL 2018)



Non linearity combining linear neurons: the Xor case



Neural networks

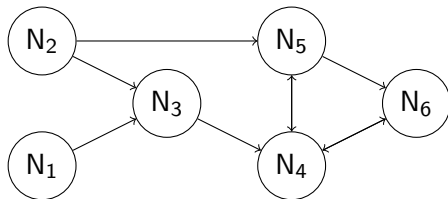
Definition: Neural network

A neural network is an oriented weighted graph of formal neurons

When two neurons are connected (linked by an oriented edge of the graph), the output of the head neuron is used as an input by the tail neuron. It can be seen as a weighted directed graph

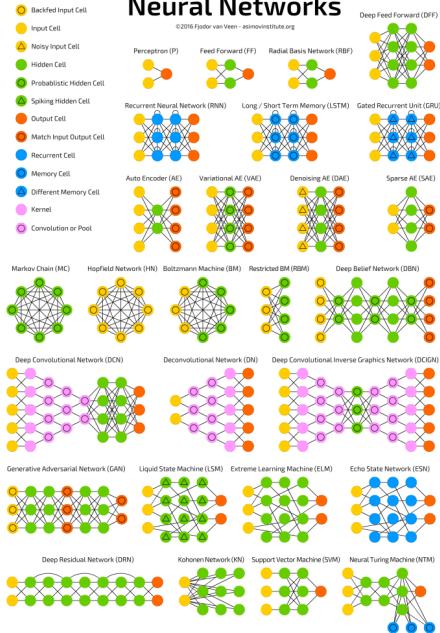
3 different neurons are considered:

- input neurons (connected with the input)
- output neurons
- hidden neurons

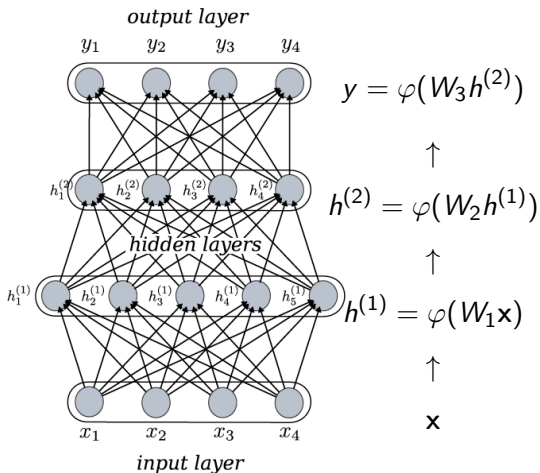


Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org



Neural networks propagation and back propagation



$$\nabla_{W_3} J = (y - y_a) \varphi'(W_3 h^{(2)}) h^{(2)}$$

↓

$$\nabla_{W_2} J = \nabla_{h^{(2)}} J \varphi'(W_2 h^{(1)}) h^{(1)}$$

↓

$$\nabla_{W_1} J =$$

backpropagation = chain rule (autodiff)

Used to learn internal representation W_1, W_2, W_3

Back propagation is differential learning



Yann LeCun

5 janvier · 🌐

OK, Deep Learning has outlived its usefulness as a buzz-phrase.
Deep Learning est mort. Vive Differentiable Programming!

Numpy

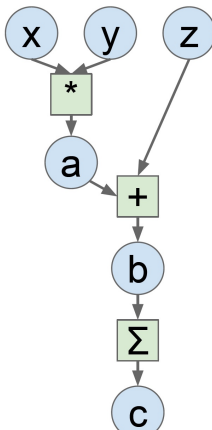
```
import numpy as np
np.random.seed(0)

N, D = 3, 4

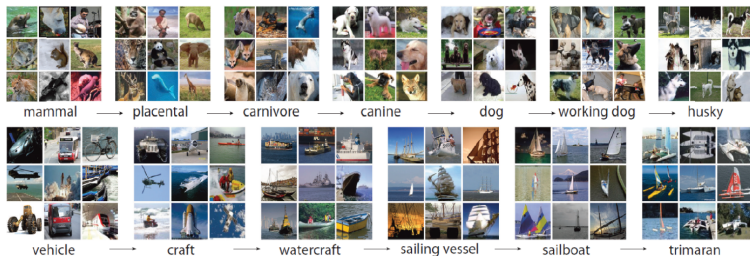
x = np.random.randn(N, D)
y = np.random.randn(N, D)
z = np.random.randn(N, D)

a = x * y
b = a + z
c = np.sum(b)

grad_c = 1.0
grad_b = grad_c * np.ones((N, D))
grad_a = grad_b.copy()
grad_z = grad_b.copy()
grad_x = grad_a * y
```



The image net database (Deng et al., 2012)



ImageNet = 15 million high-resolution images of 22,000 categories.
Large-Scale Visual Recognition Challenge (a subset of ImageNet)

- 1000 categories.
- 1.2 million training images,
- 50,000 validation images,
- 150,000 testing images.

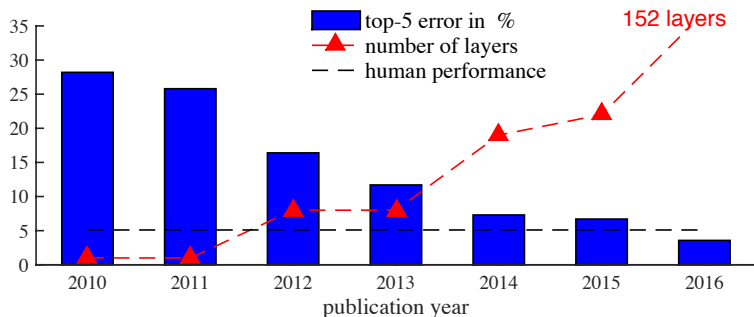
A new fashion in image processing

2012 Teams	%error	2013 Teams	%error	2014 Teams	%error
Supervision (Toronto)	15.3	Clarifai (NYU spinoff)	11.7	GoogLeNet	6.6
ISI (Tokyo)	26.1	NUS (singapore)	12.9	VGG (Oxford)	7.3
VGG (Oxford)	26.9	Zeiler-Fergus (NYU)	13.5	MSRA	8.0
XRCE/INRIA	27.0	A. Howard	13.5	A. Howard	8.1
UvA (Amsterdam)	29.6	OverFeat (NYU)	14.1	DeeperVision	9.5
INRIA/LEAR	33.4	UvA (Amsterdam)	14.2	NUS-BST	9.7
		Adobe	15.2	TTIC-ECP	10.2
		VGG (Oxford)	15.2	XYZ	11.2
		VGG (Oxford)	23.0	UvA	12.1

shallow approaches

deep learning

ImageNet results



2012 Alex Net

2013 ZFNet

2014 VGG

2015 GoogLeNet / Inception

2016 Residual Network

Deep learning for galaxy morphology prediction

- Data

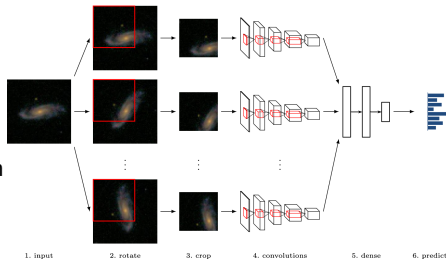
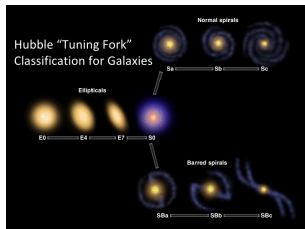
- ▶ training 61,578 / test 79,975 images

- model

- ▶ inputs: 45 by 45 by 3 arrays of RGB
- ▶ 16 different viewpoints
- ▶ outut: 37 answer probabilities
- ▶ four convolutional layers
- ▶ 42 million trainable parameters

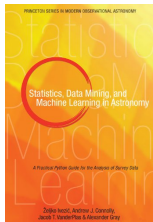
- training

- ▶ Data augmentation, parameter sharing & dropout
- ▶ manual parameter search: more than 100 architectures were evaluated
- ▶ Model averaging
- ▶ 67 hours (accuracy of 98%)

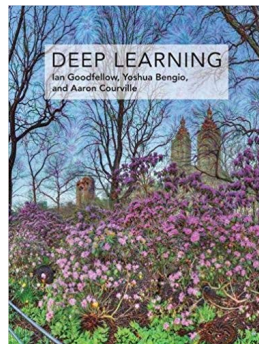
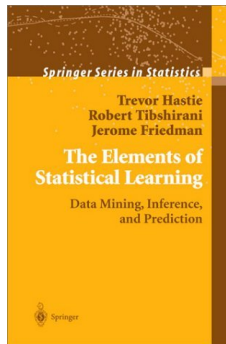
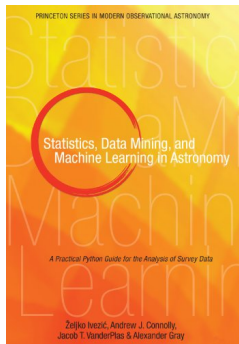


Plan

- 1 Introducing example
- 2 Machine learning
 - Definition attempt
 - Hypothesis set and methods
- 3 Deep Learning
- 4 Some practical issues



A book to begin with machine learning in astronomy



Ivezic, Zeljko, et al. **Statistics, Data Mining, and Machine Learning in Astronomy: A Practical Python Guide for the Analysis of Survey Data**. Princeton University Press, 2014.

astroML: code to begin with machine learning in astronomy

Python module for machine learning and data mining built on numpy, scipy, **scikit-learn**, and matplotlib

The screenshot shows the scikit-learn documentation website. At the top, there's a navigation bar with links for Download, Support, User Guide, Examples, and Reference. A search bar is on the right. Below the navigation bar, there's a sidebar on the left with links for Previous, Next, and Up. The main content area is titled '2.3. Using scikit-learn with Astronomical Data'. It contains a section 'Machine Learning for Astronomy with scikit-learn' which describes scikit-learn as a Python module integrating classic machine learning algorithms. It also mentions that it aims to provide simple and efficient solutions to learning problems. A note at the bottom of this section states that the document is meant to be used with scikit-learn version 0.11+ (i.e. the current state of the master branch at the time of writing: 2012-02-13). Below this, there are three sub-sections: '2.3.1. Astronomy Tutorial setup', '2.3.2. Machine Learning 101: General Concepts', and '2.3.3. Machine Learning 102: Practical Advice'. Each sub-section has a list of bullet points detailing specific topics.

scikit-learn

Download Support User Guide Examples Reference

Google Custom Search Search x

Previous 4. Finding help Next 2.3.1. Astronomy Tutorial Up 2.3.2. Machine Learning 101: General Concepts

This documentation is for scikit-learn version 0.11-git — Other versions

Citing

If you use the software, please consider citing scikit-learn.

This page

2.3. Using scikit-learn with Astronomical Data

Machine Learning for Astronomy with scikit-learn

scikit-learn is a Python module integrating classic machine learning algorithms in the tightly-knit world of scientific Python packages (numpy, scipy, matplotlib).

It aims to provide simple and efficient solutions to learning problems that are accessible to everybody and reusable in various contexts: **machine-learning as a versatile tool for science and engineering**.

Note: This document is meant to be used with **scikit-learn version 0.11+** (i.e. the current state of the master branch at the time of writing: 2012-02-13).

2.3.1. Astronomy Tutorial setup

- 2.3.1.1. Setup

2.3.2. Machine Learning 101: General Concepts

- 2.3.2.1. Features and feature extraction
- 2.3.2.2. Supervised Learning: `model.fit(X, y)`
- 2.3.2.3. Unsupervised Learning: `model.fit(X)`
- 2.3.2.4. Linearly separable data
- 2.3.2.5. Training set, test set and overfitting
- 2.3.2.6. Key takeaway points

2.3.3. Machine Learning 102: Practical Advice

- 2.3.3.1. Bias, Variance, Over-fitting, and Under-fitting
- 2.3.3.2. Cross-Validation and Testing
- 2.3.3.3. Learning Curves
- 2.3.3.4. Summary

astroML: github.com/astroML/astroML
ogrisel.github.io/scikit-learn.org/sklearn-tutorial/tutorial/astronomy

AstroNN: code to begin with deep learning in astronomy

Python module for deep learning built on Keras (and scikit-learn)

Galaxy10 Tutorial

Introduction

This notebook will demonstrate how to train a simple convolutional neural network with astroNN on Galaxy10 dataset to classify galaxy images.

Galaxy10 is a dataset contains 25753 69x69 pixels colored galaxy images (g, r and i band) separated in 10 classes. Galaxy10 images come from Sloan Digital Sky Survey and labels come from Galaxy Zoo.

There is no guarantee on the accuracy of the labels. Moreover, Galaxy10 is not a balanced dataset and it should only be used for educational or experimental purpose. If you use Galaxy10 for research purpose, please cite Galaxy Zoo and Sloan Digital Sky Survey.

For more information on the original classification tree: https://data.galaxyzoo.org/gz_trees/gz_trees.html

Authors and Basic Information

- **Henry Leung** - Astronomy student, University of Toronto - [henrysky](https://github.com/henrysky)
- Project advisor: **Jo Bovy** - Professor, Department of Astronomy and Astrophysics, University of Toronto - [jobovy](https://github.com/jobovy)
- Contact Henry: [henrysky.leung\[at\]mail.utoronto.ca](mailto:henrysky.leung[at]mail.utoronto.ca)
- This tutorial is created on 10/Feb/2018 with Keras 2.1.3, Tensorflow 1.6.0, Nvidia CuDNN 7.0 for CUDA 9.0 (Optional), Python 3.6.3 Win10 x64

This tutorial will be using astroNN

- Galaxy10 description on astroNN: <http://astronn.readthedocs.io/en/latest/galaxy10.html>
- astroNN github: <https://github.com/henrysky/astroNN>
- astroNN documentation: <http://astronn.readthedocs.io/>

Example images of each class from Galaxy10 dataset

Disk, Face-on, No Spiral



Smooth, Completely round



Smooth, in-between round



Smooth, Cigar shaped



Disk, Edge-on, Rounded Bulge



Neural Network for Identifying Exoplanets

THE ASTRONOMICAL JOURNAL, 155:94 [21pp], 2018 February

© 2018, The American Astronomical Society.

OPEN ACCESS

<https://doi.org/10.3847/1538-3881/aa9e09>



Identifying Exoplanets with Deep Learning: A Five-planet Resonant Chain around Kepler-80 and an Eighth Planet around Kepler-90

Christopher J. Shallue¹ and Andrew Vanderburg^{2,3,4}

¹ Google Brain, 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA; shallue@google.com

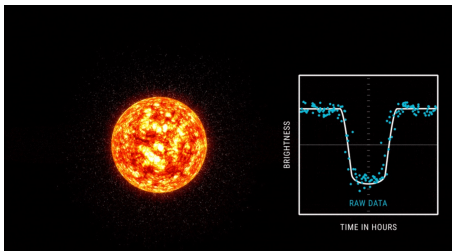
² Department of Astronomy, The University of Texas at Austin, 2515 Speedway, Stop C1400, Austin, TX 78712, USA

³ Harvard-Smithsonian Center for Astrophysics, 60 Garden Street, Cambridge, MA 02138, USA

Received 2017 September 19; revised 2017 November 13; accepted 2017 November 20; published 2018 January 30

Abstract

NASA's *Kepler Space Telescope* was designed to determine the frequency of Earth-sized planets orbiting Sun-like stars, but these planets are on the very edge of the mission's detection sensitivity. Accurately determining the occurrence rate of these planets will require automatically and accurately assessing the likelihood that individual candidates are indeed planets, even at low signal-to-noise ratios. We present a method for classifying potential planet signals using deep learning, a class of machine learning algorithms that have recently become state-of-the-



Brut force training

- Data

- ▶ 3600 planet candidates
- ▶ 9596 astrophysical false positive
- ▶ 2541 nontransiting phenomenon

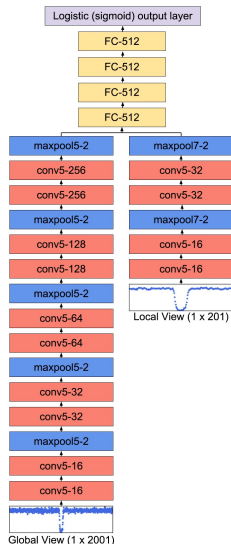
- model

- ▶ inputs: both global and local input
- ▶ convolutional deep network

- training

- ▶ Google vizier for hyper parameters tuning
- ▶ 100 CPUs in parallel

- prediction: 10 copies model averaging



Astronet: Neural Network for Identifying Exoplanets

github.com/tensorflow/models/tree/master/research/astronet

[Code](#) [Issues 1,027](#) [Pull requests 291](#) [Projects 2](#) [Wiki](#) [Insights](#)

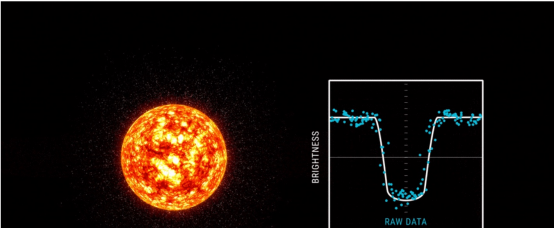
Branch: master [models / research / astronet /](#) [Create new file](#) [Upload files](#) [Find file](#) [History](#)

 **Mark Daoust** Merge pull request #4629 from lamberta/fix-rename-guide-models [View](#) Latest commit fb3fba8 on 27 Jun

..		
 astronet	Update file permission mask in generate_download_script.py.	3 months ago
 docs	Initial AstroNet commit.	7 months ago
 light_curve_util	Add __str__ and __repr__ to periodic_event.Event.	3 months ago
 third_party	Modularize light curve and TCE preprocessing functions for easier re-...	4 months ago
 .gitignore	Internal change.	7 months ago
 README.md	Rename programmers_guide/ to guide/ in tf-models.	3 months ago
 WORKSPACE	Initial AstroNet commit.	7 months ago

[README.md](#)

AstroNet: A Neural Network for Identifying Exoplanets in Light Curves



Conclusion

- Deep learning
 - ▶ pas seulement: XGboost, SVM...
 - ▶ représentation: t-SNE
- Beaucoup de données
- Introduction d'a priori
- Attention à la méthodologie (pour bien généraliser)
- Python

