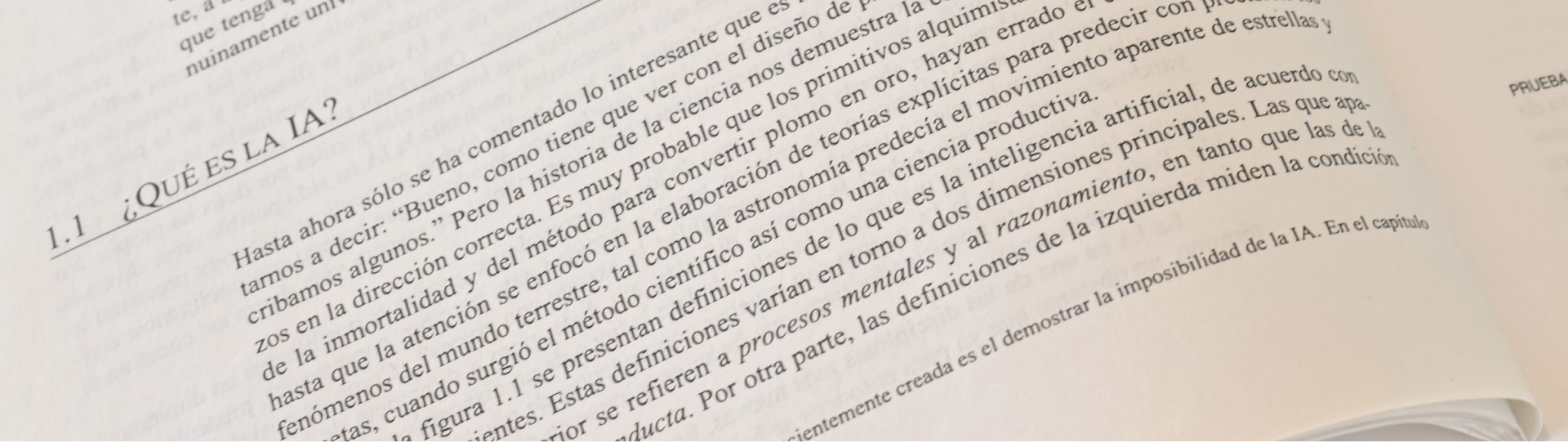
Inteligencia Artificial **con R**

@javierluraschi - RStudio PBC

Hola!

Javier Luraschi

Ingeniero en Software con RStudio y autor de Mastering Spark with R con O'Reilly. Antes trabaje en Microsoft Research y SAP. Estudie Matematicas e Ingenieria en Software.



Inteligencia Artificial


Russell
Norvig
Inteligencia Artificial
Un enfoque moderno
Computer Speech Technology
AYN RAND
DIAMOND
Chocolate
MOMO

"La interesante tarea de lograr que las computadoras piensen... *máquinas con mente*, en su amplio sentido literal." (Haugeland, 1985.)	"El estudio de las facultades mentales mediante el uso de modelos computacionales." (Charniak y McDermott, 1985.)
"[La automatización de] actividades que vinculamos con procesos de pensamiento humano, actividades tales como toma de decisiones, resolución de problemas, aprendizaje..." (Bellman, 1978.)	"El estudio de los cálculos que permiten percibir, razonar y actuar." (Winston, 1992.)
"El arte de crear máquinas con capacidad de realizar funciones que realizadas por personas requieren de inteligencia." (Kurzweil, 1990.)	"Un campo de estudio que se enfoca a la explicación y emulación de la conducta inteligente en función de procesos computacionales." (Schalkoff, 1990.)
"El estudio de cómo lograr que las computadoras realicen tareas que, por el momento, los humanos hacen mejor." (Rich y Knight, 1991.)	"La rama de la ciencia de la computación que se ocupa de la automatización de la conducta inteligente." (Luger y Stubblefield, 1993.)

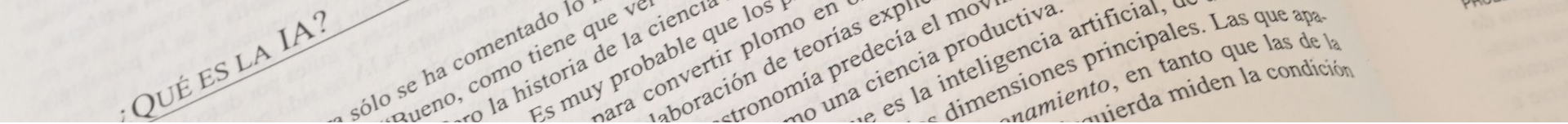
Figura 1.1 Algunas definiciones de IA. Se agrupan en cuatro categorías:

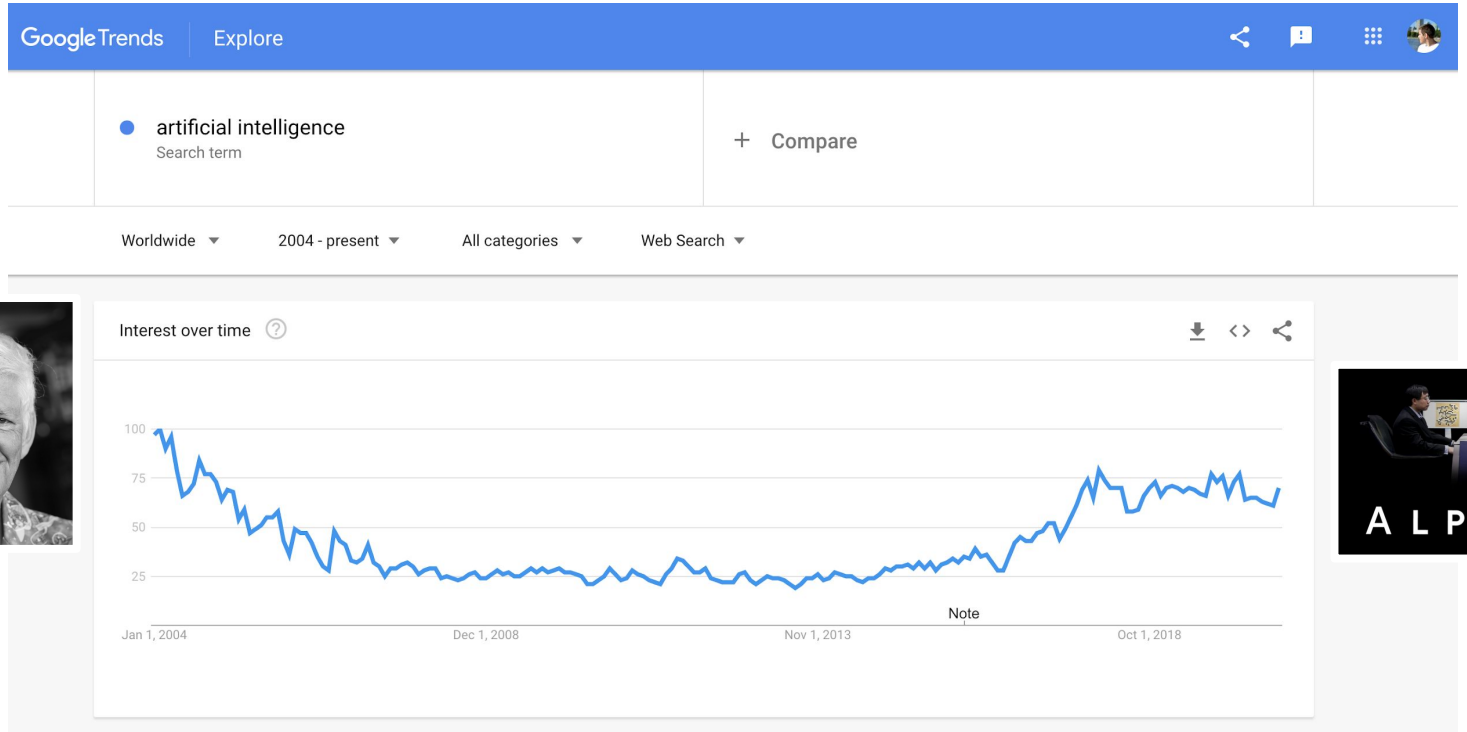
Sistemas que piensan como humanos.	Sistemas que piensan racionalmente.
Sistemas que actúan como humanos.	Sistemas que actúan racionalmente.

RACIONALIDAD

deseable en función de *eficiencia humana*, mientras que las de la derecha lo hacen de conformidad con un concepto de inteligencia *ideal*, denominado **racionalidad**. Se considera que un sistema es racional si hace lo correcto. De acuerdo con lo anterior, en la inteligencia artificial existen cuatro posibles objetivos por alcanzar, como se comenta en el pie de página de la figura 1.1.

A lo largo de la historia se han adoptado los cuatro enfoques anteriormente
luego, existe una tensión entre enfoques centrados en los humanos
El enfoque centrado en el comportamiento humano
empleo de hipótesis y de la
matemáticas e


¿QUÉ ES LA IA?


Google Trends
Explore
artificial intelligence
Search term
+ Compare
Worldwide
2004 - present
All categories
Web Search
Interest over time
100
75
50
25
Note
Jan 1, 2004
Dec 1, 2008
Nov 1, 2013
Oct 1, 2018


ALPHAGO

Mastering the Game of Go with Deep Neural Networks and Tree Search

David Silver[1]*, Aja Huang[1]*, Chris J. Maddison[1], Arthur Guez[1], Laurent Sifre[1], George van den Driessche[1], Julian Schrittwieser[1], Ioannis Antonoglou[1], Veda Panneershelvam[1], Marc Lanctot[1], Sander Dieleman[1], Dominik Grewe[1], John Nham[2], Nal Kalchbrenner[1], Ilya Sutskever[2], Timothy Lillicrap[1], Madeleine Leach[1], Koray Kavukcuoglu[1], Thore Graepel[1], Demis Hassabis[1].

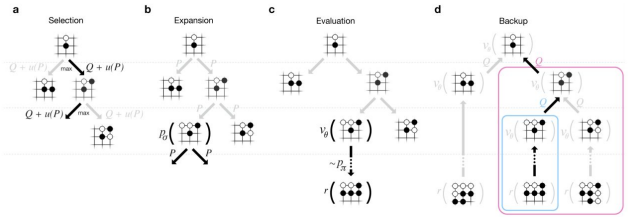


Figure 3: **Monte-Carlo tree search in *AlphaGo*.** **a** Each simulation traverses the tree by selecting the edge with maximum action-value Q, plus a bonus $u(P)$ that depends on a stored prior probability P for that edge. **b** The leaf node may be expanded; the new node is processed once by the policy network p_σ and the output probabilities are stored as prior probabilities P for each action. **c** At the end of a simulation, the leaf node is evaluated in two ways: using the value network v_θ; and by running a rollout to the end of the game with the fast rollout policy p_π, then computing the winner with function r. **d** Action-values Q are updated to track the mean value of all evaluations $r(\cdot)$ and $v_\theta(\cdot)$ in the subtree below that action.

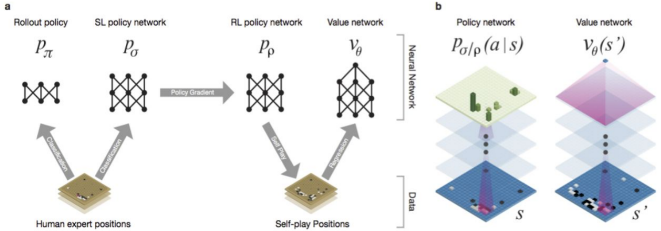


Figure 1: **Neural network training pipeline and architecture. a** A fast rollout policy p_π and supervised learning (SL) policy network p_σ are trained to predict human expert moves in a data-set of positions. A reinforcement learning (RL) policy network p_ρ is initialised to the SL policy network,

AlphaGo	Search threads	CPUs	GPUs	Elo
Asynchronous	1	48	8	2203
Asynchronous	2	48	8	2393
Asynchronous	4	48	8	2564
Asynchronous	8	48	8	2665
Asynchronous	16	48	8	2778
Asynchronous	32	48	8	2867
Asynchronous	40	48	8	2890
Asynchronous	40	48	1	2181
Asynchronous	40	48	2	2738
Asynchronous	40	48	4	2850
Distributed	12	428	64	2937
Distributed	24	764	112	3079
Distributed	40	1202	176	3140
Distributed	64	1920	280	3168

Ciencia de Datos

Monte Carlo: “Computational algorithms that relies on repeated random sampling”

Deep Learning

“Family of machine learning methods based on artificial neural networks, inspired by information processing nodes in biological systems”

Computación Distribuida

“A distributed system is a system whose components are located on different networked computers”

Ciencia de Datos

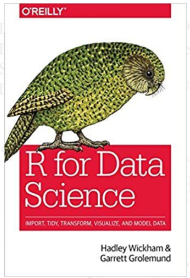
Deep Learning

Datos y Modelos

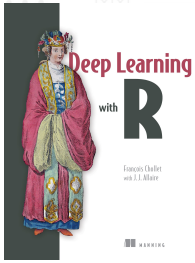
mlflow

Computación Distribuida

Ciencia de Datos



Deep Learning

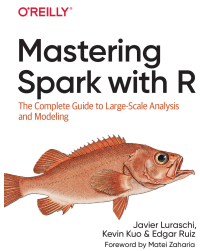


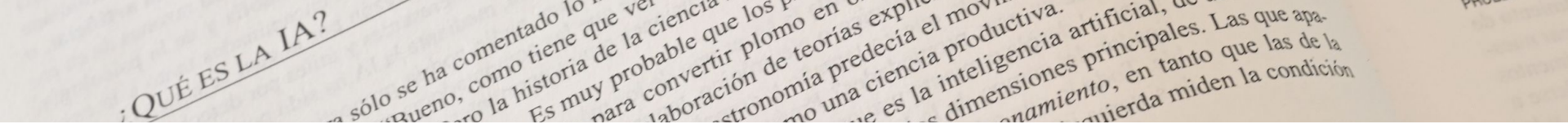
torch.mlverse.org

Datos y Modelos

pins.rstudio.com
mlflow.org

Computación Distribuida





Deep Learning

TensorFlow

Datos

kaggle

Microsoft Azure

GitHub

pins

Google Cloud

RStudio

amazon S3

2016

2017

2018

2019

2020

APACHE Spark™

Computación Distribuida

mlflow

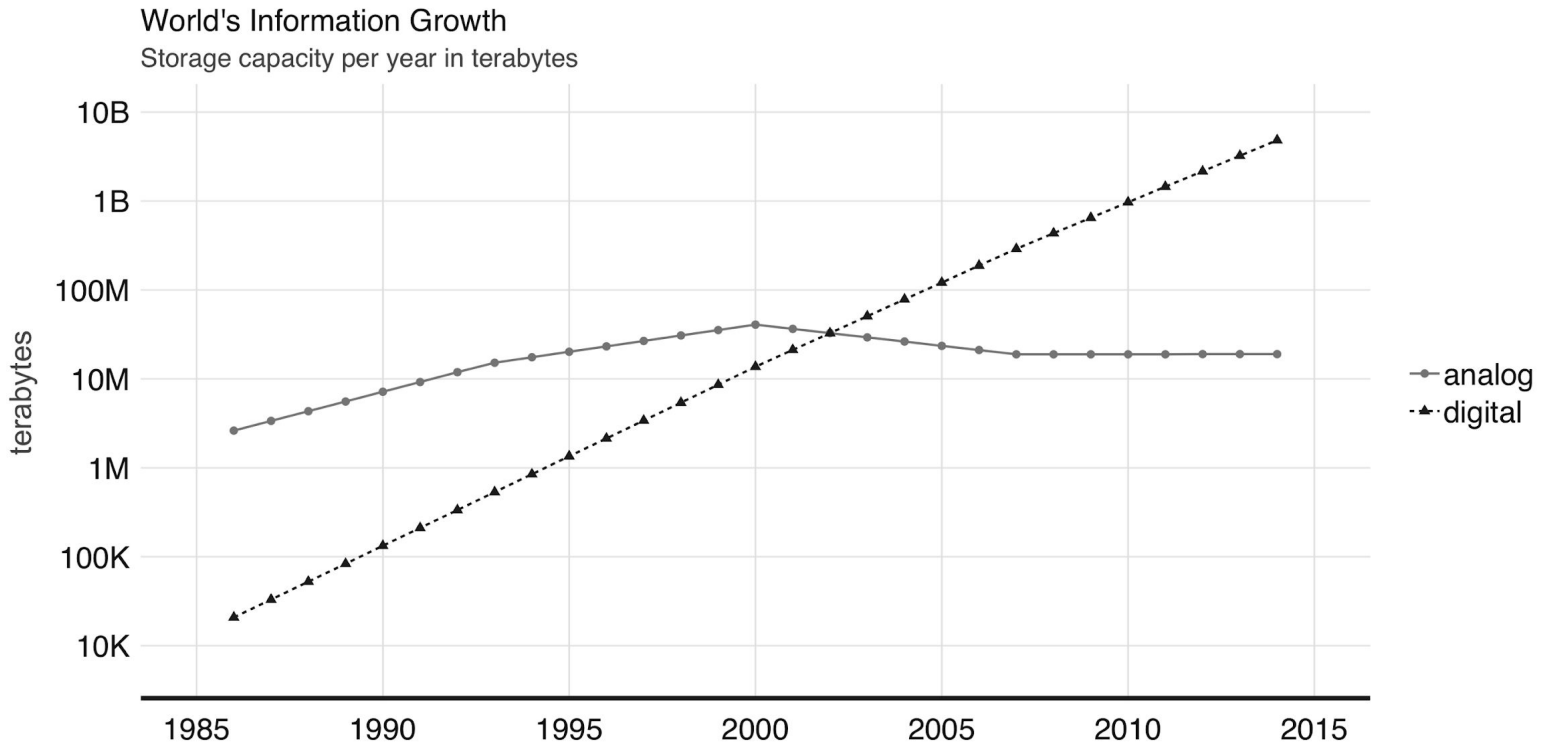
Modelos

torch

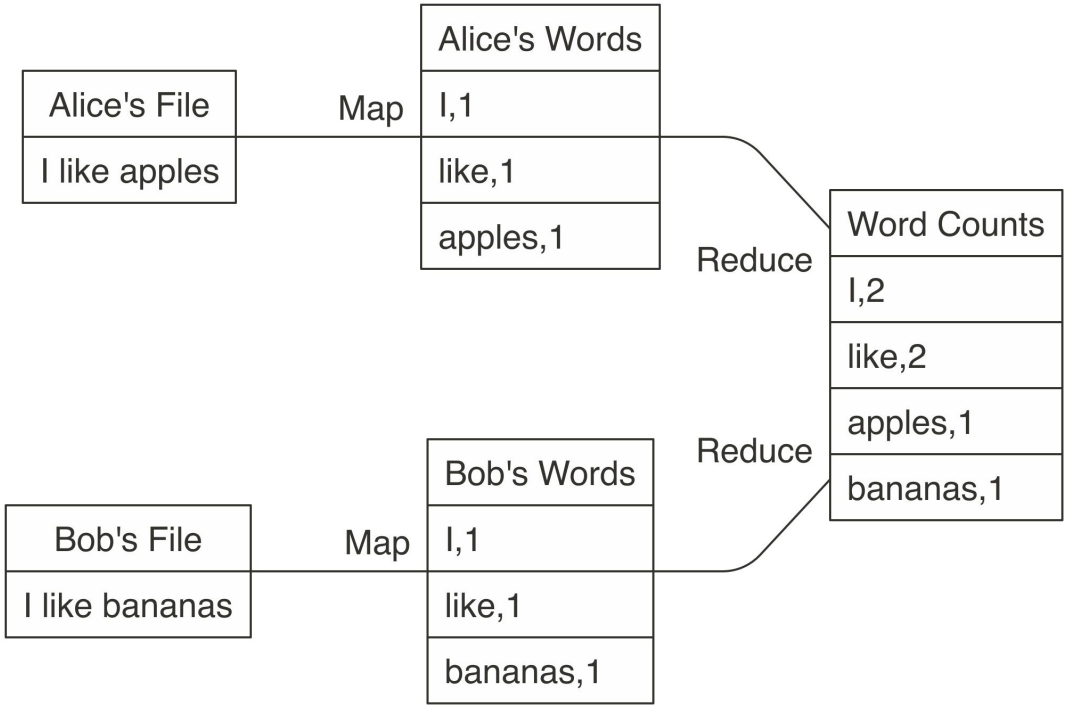
Deep Learning

Computación Distribuida

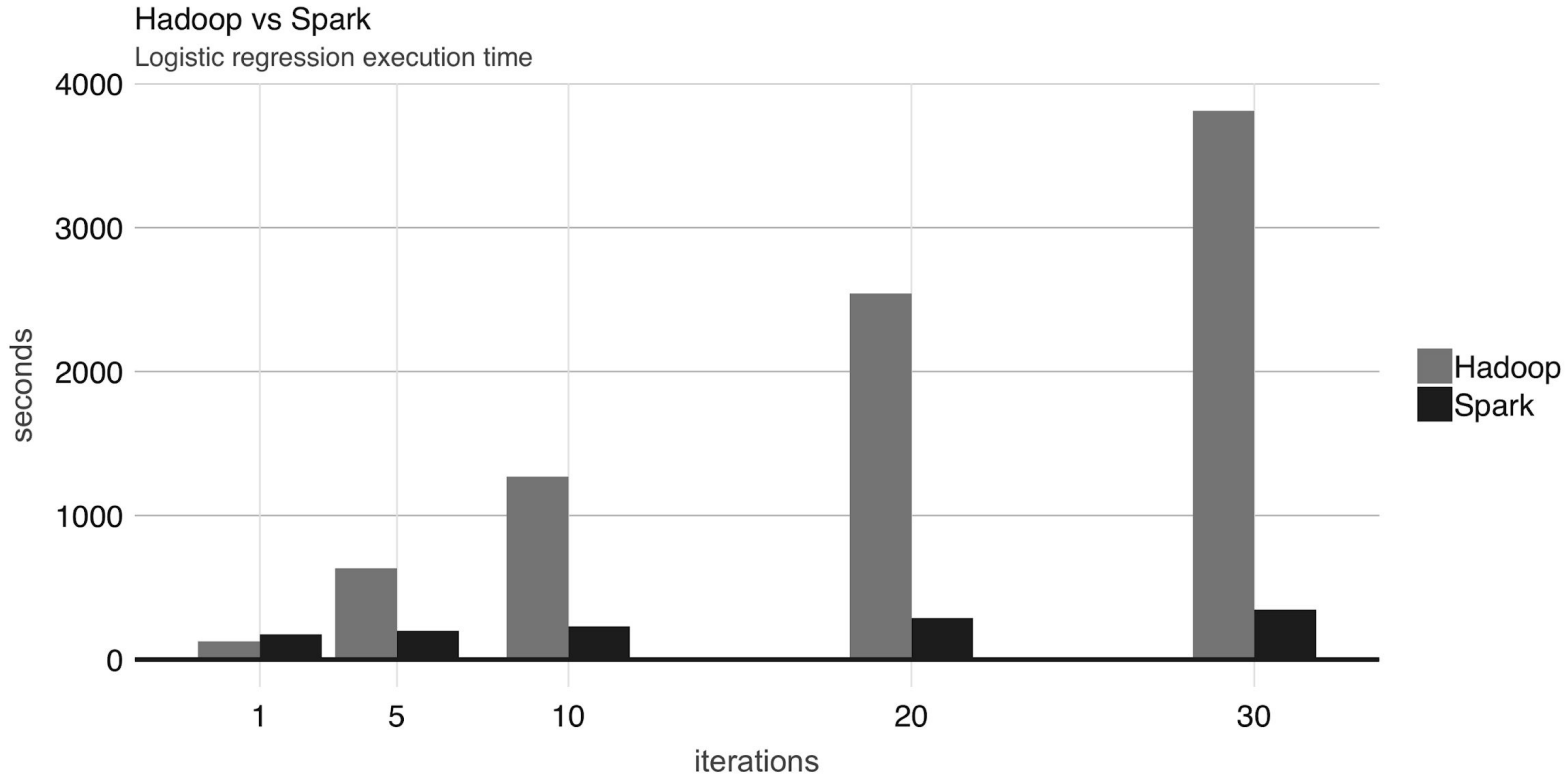
El crecimiento de informacion



Que es el procesamiento distribuido?

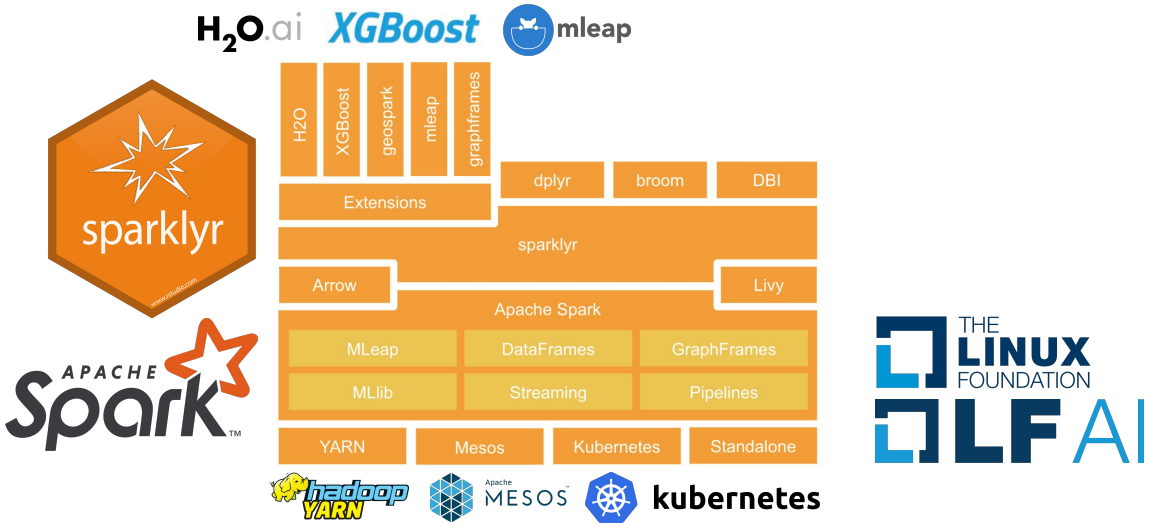


Spark vs Hadoop



sparklyr: R Interface to Spark

Sparklyr is an open-source and modern interface to scale data science and machine learning workflows using Apache Spark™ and R.



Cómo utilizo Spark en R?


```
library(sparklyr)
sc <- spark_connect(master = "local|yarn|mesos|spark|livy")
flights <- copy_to(sc, flights)
```


```
library(dplyr)
flights %>%
  group_by(month, day) %>%
  summarise(count = n(), avg_delay = mean(dep_delay)) %>%
  filter(count > 1000)
```

Qué funcionalidad existe?

```
spark_install()                                  # Install local Spark
sc <- spark_connect(master = "local")            # Connect to Spark cluster

cars <- spark_read_csv(sc, "cars", "input/")     # Read data in Spark

summarize(cars, n = n())                         # Count records with dplyr
dbGetQuery(sc, "SELECT count(*) FROM cars")      # Count records with DBI

ml_linear_regression(cars, mpg ~ wt + cyl)       # Perform linear regression

ml_pipeline(sc) %>%                              # Define Spark pipeline
  ft_r_formula(mpg ~ wt + cyl) %>%               # Add formula transformation
  ml_linear_regression()                         # Add model to pipeline

spark_context(sc) %>% invoke("version")          # Extend sparklyr with Scala
spark_apply(cars, nrow)                          # Extend sparklyr with R

stream_read_csv(sc, "input/") %>%                # Define Spark stream
  filter(mpg > 30) %>%                           # Add dplyr transformation
  stream_write_json("output/")                   # Start processing stream
```

Cual es la historia de sparklyr?

THE LINUX FOUNDATION

Sep 2016 — sparklyr 0.4
R interface for Apache Spark. H2O support.

Jan 2017 — sparklyr 0.5
Livy and dplyr improvements. CDH Certified.

Jul 2017 — sparklyr 0.6
Distributed R and external sources.

Jan 2018 — sparklyr 0.7
Pipelines and Machine Learning.

May 2018 — sparklyr 0.8
Production pipelines and graphs.

Oct 2018 — sparklyr 0.9
Streams and Kubernetes.

March 2019 — sparklyr 1.0
Arrow, XGBoost, Broom and TFRecords

Jan 2020 — sparklyr 1.1
Delta Lake Barriers

May 2020 — sparklyr 1.2
Foreach, Spark 3.0

Jan 2020 — sparklyr 1.3
HOF, Avro, Serializers

Sep 2020 — Sparklyr 1.4
Tidyr, GPU, weighted sampling

H2O.ai IBM cloudera aws Azure Google Cloud databricks Qubole

sparklyr 1.2: foreach

The foreach package provides the %dopar% operator to iterate over elements in a collection in parallel. Using sparklyr 1.2, you can now register Spark as a backend using registerDoSpark() and then easily iterate over R objects using Spark:

```
library(sparklyr)
library(foreach)
sc <- spark_connect(master = "local", version = "2.4")
registerDoSpark(sc)
foreach(i = 1:3, .combine = 'c') %dopar% {
  sqrt(i)
}
```

```
[1] 1.000000 1.414214 1.732051
```

Since many R packages are based on foreach to perform parallel computation, we can now make use of all those great packages in Spark as well!

sparklyr 1.3: High Level Order functions

```
result_tbl <- coins_tbl %>%
  hof_zip_with(~ .x * .y, dest_col = total_values) %>%
  dplyr::select(total_values)

result_tbl %>% dplyr::pull(total_values)
```

```
[1]  4000 15000 20000 25000
```

```
result_tbl %>%
  dplyr::mutate(zero = dplyr::sql("CAST (0 AS BIGINT)")) %>%
  hof_aggregate(start = zero, ~ .x + .y, expr = total_values, dest_col = total) %>%
  dplyr::select(total) %>%
  dplyr::pull(total)
```

```
[1] 64000
```

sparklyr 1.4: TidyR y GPUs con RAPIDS

```
mtcars_kv_sdf <- mtcars_sdf %>%
  tidyr::pivot_longer(cols = -model, names_to = "key", values_to = "value")
print(mtcars_kv_sdf, n = 5)
```

```
# Source: spark<?> [?? x 3]
  model     key   value
  <chr>     <chr> <dbl>
1 Mazda RX4 am      1
2 Mazda RX4 carb    4
3 Mazda RX4 cyl     6
4 Mazda RX4 disp  160
5 Mazda RX4 drat    3.9
# … with more rows
```

```
library(sparklyr)

sc <- spark_connect(master = "local", version = "3.0.0", packages = "rapids")
dplyr::db_explain(sc, "SELECT 4")
```

```
== Physical Plan ==
*(2) GpuColumnarToRow false
+- GpuProject [4 AS 4#45]
   +- GpuRowToColumnar TargetSize(2147483647)
      +- *(1) Scan OneRowRelation[]
```

sparklyr.ai

THE LINUX FOUNDATION PROJECTS

sparklyr

About Get Started Learning Community

Scale Data Science with Spark and R

About

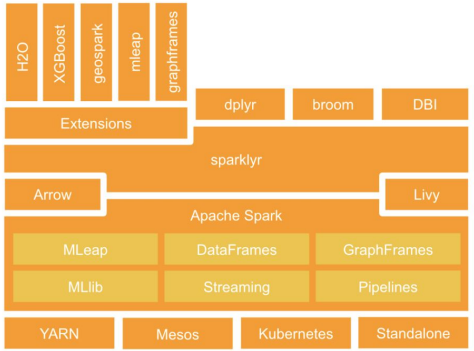
sparklyr is an open-source and modern interface to scale data science and machine learning workflows using Apache Spark™, R, and a rich extension ecosystem.

It enables using Apache Spark with ease using R by providing access to core functionality like installing, connecting and managing Spark and using Spark's MLlib, Spark Structured Streaming and Spark Pipelines from R.

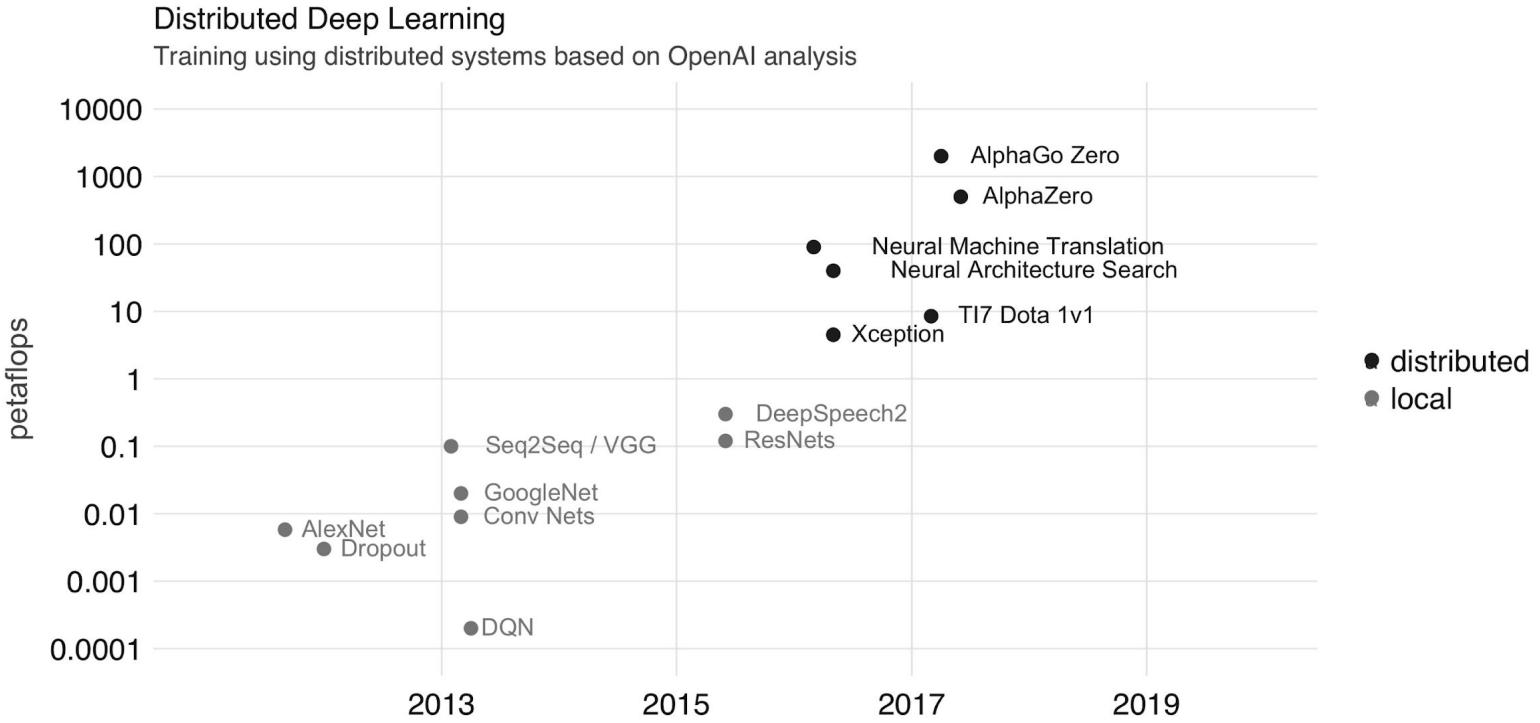
Supports well-known R packages like dplyr, DBI and broom to reduce the cognitive overhead from having to re-learn libraries.

And enables a rich-ecosystem of extensions to use in Spark and R: XGBoost, MLeap, GraphFrames, H2O, and optionally enable Apache Arrow to significantly improve performance.

Through Spark, this allows you to scale your Data Science workflows in Hadoop YARN, Mesos, Kubernetes or Apache Livy.



Cómo está esto relacionado con Deep Learning?



Deep Learning con TensorFlow

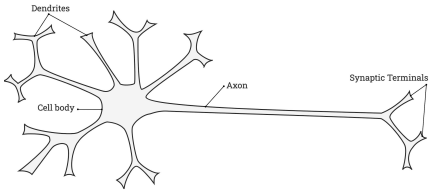
Comprehensive Survey on Deep Learning

II. Deep Neural Network (DNN)

A. The History of DNN

Below is a brief history of neural networks highlighting key events:

- 1943: McCulloch & Pitts show that neurons can be combined to construct a Turing machine (using ANDs, ORs, & NOTs) [44].
- 1958: Rosenblatt shows that perceptron's will converge if what they are trying to learn can be represented [45].
- 1969: Minsky & Papert show the limitations of perceptron's, killing research in neural networks for a decade [46].
- 1985: The backpropagation algorithm by GeoffreyHinton et al [47] revitalizes the field.
- 1988: Neocognitron: a hierarchical neural network capable of visual pattern recognition [48].
- 1998: CNNs with Backpropagation for document analysis by Yan LeCun [49].
- 2006: The Hinton lab solves the training problem for DNNs [50,51].
- 2012 : AlexNet by Alex Krizhevesky in 2012 [7].



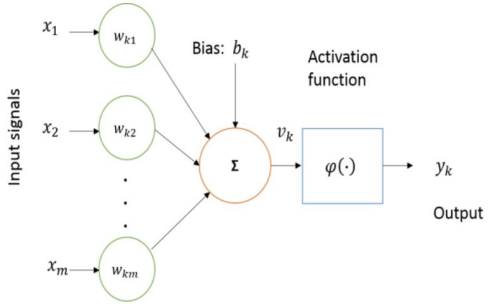


Fig. 9. Basic model of a neuron

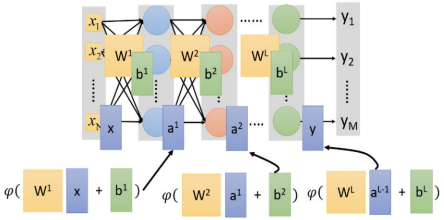


Fig. 10. Neural network model with multiple layers perceptron

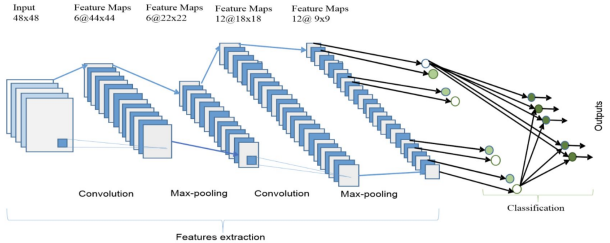
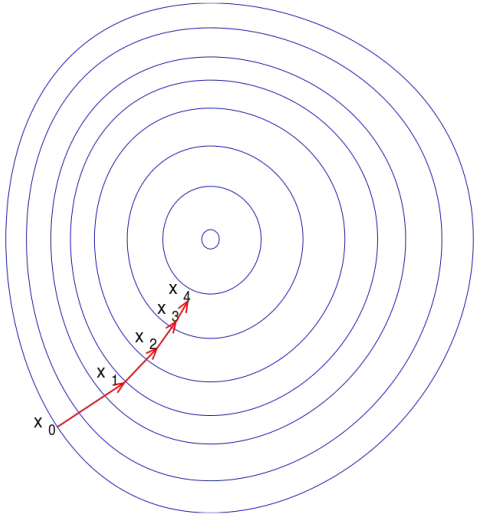


Fig. 11. The overall architecture of the CNN includes an input layer, multiple alternating convolution and max-pooling layers, one fully-connected layer and one classification layer.

Functionality in Deep Learning Frameworks



Automatic Differentiation

Graphic Processing Units (GPUs)

Intro to TensorFlow with Keras

```r
library(keras)

mnist <- dataset_mnist()
mnist$train$x <- mnist$train$x/255
mnist$test$x <- mnist$test$x/255

model <- keras_model_sequential() %>%
  layer_flatten(input_shape = c(28, 28)) %>%
  layer_dense(units = 128, activation = "relu") %>%
  layer_dropout(0.2) %>%
  layer_dense(10, activation = "softmax")

model %>% compile(loss = "sparse_categorical_crossentropy",
                  optimizer = "adam",
                  metrics = "accuracy")

model %>% fit(x = mnist$train$x, y = mnist$train$y,
              epochs = 5, validation_split = 0.3, verbose = 2)

predict(model, mnist$test$x)
```

TensorFlow 2.0: Eager Execution

```
library(tensorflow)
tf$math$cumprod(1:5)
```

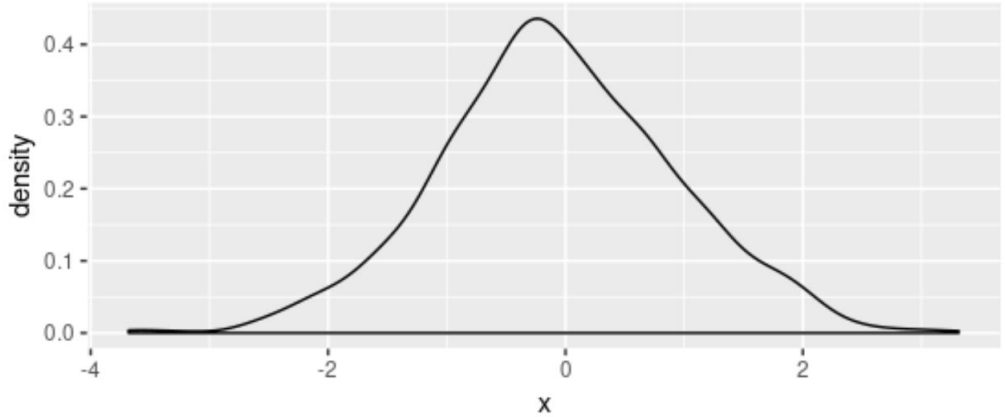
```
Tensor("Cumprod:0", shape=(5,), dtype=int32)
```

```
tf$math$cumprod(1:5)
```

```
tf.Tensor([  1   2   6  24 120], shape=(5,), dtype=int32)
```

tfprobability: R Interface to TensorFlow Probability

```
library(tfprobability)
library(tensorflow)
tfe_enable_eager_execution()

library(ggplot2)

b <- tfb_normal_cdf()
u <- runif(1000)
x <- b %>% tfb_inverse(u) %>% as.numeric()

x %>% data.frame(x = .) %>% ggplot(aes(x = x)) + geom_density()
```



tfdatasets: R Interface to TensorFlow Dataset API

MAPPING

You can map arbitrary transformation functions onto dataset records using the `dataset_map()` function. For example, to transform the "Species" column into a one-hot encoded vector you would do this:

```
dataset <- dataset %>%
  dataset_map(function(record) {
    record$Species <- tf$one_hot(record$Species, 3L)
    record
  })
```

Deep Learning y Computación Distribuida

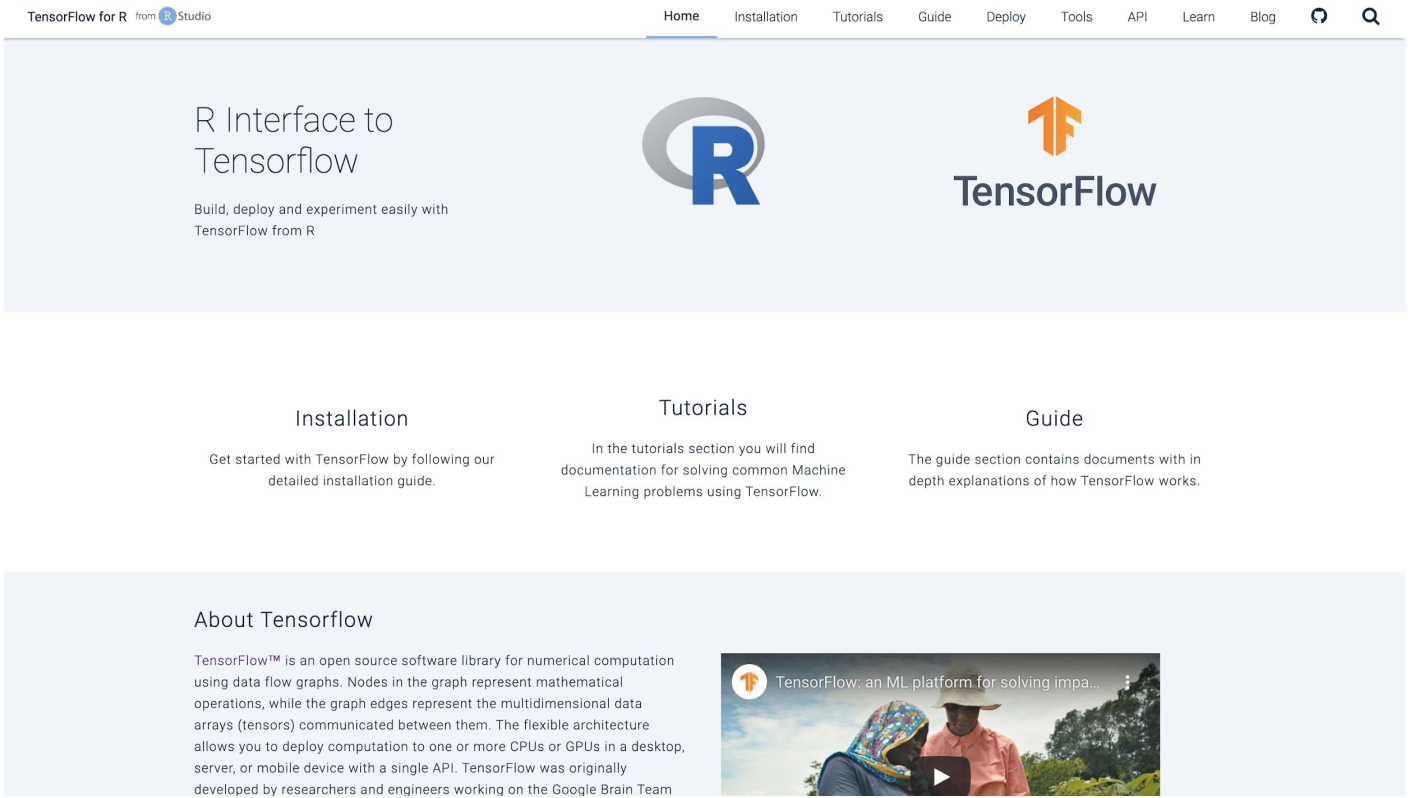
Aug. 24, 2020

Training ImageNet with R

This post explores how to train large datasets with TensorFlow and R. Specifically, we present how to download and repartition ImageNet, followed by training ImageNet across multiple GPUs in distributed environments using TensorFlow and Apache Spark.

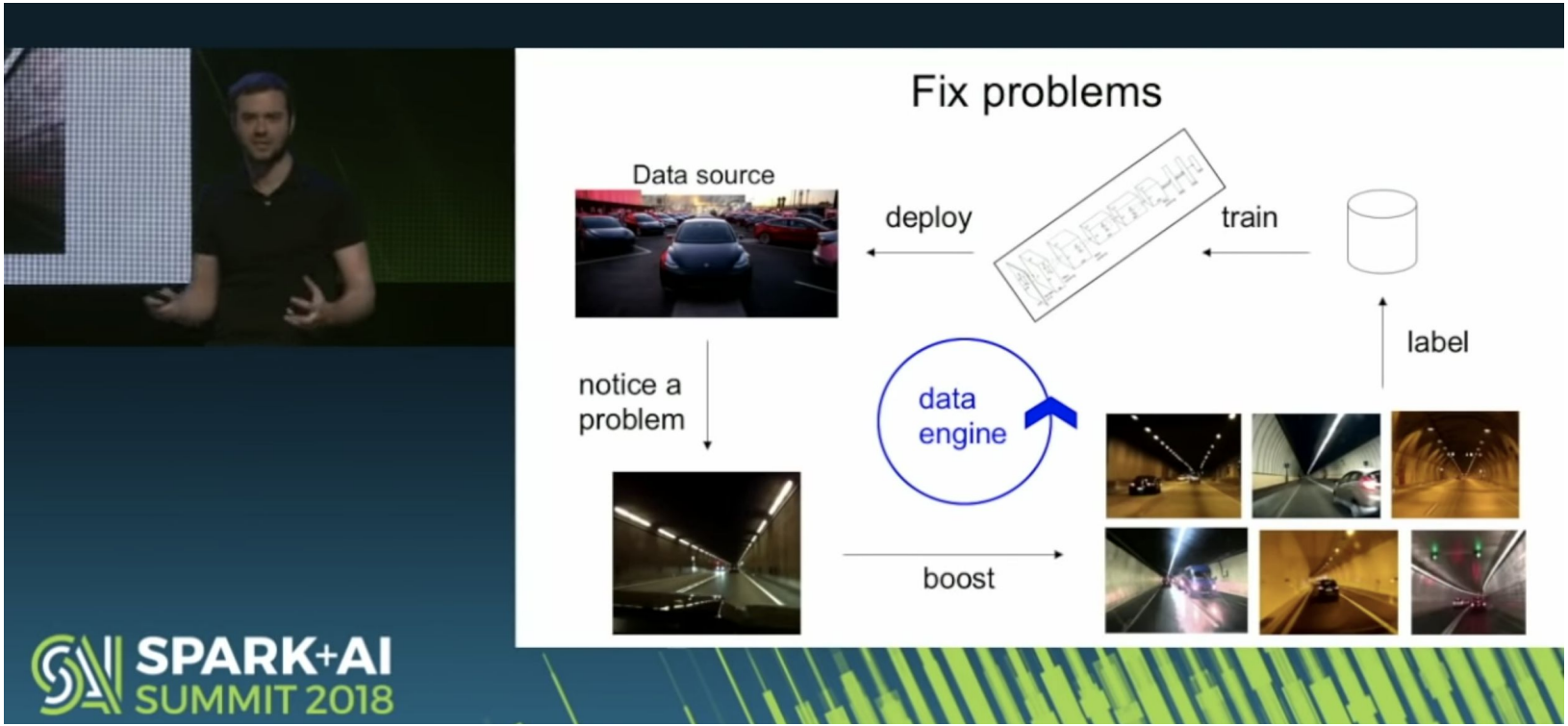
```r
library(sparklyr)
sc <- spark_connect("yarn|mesos|etc", config = list("sparklyr.shell.num-executors" = 16))

sdf_len(sc, 16, repartition = 16) %>%
  spark_apply(function(df, barrier) {
      library(tensorflow)

      Sys.setenv(TF_CONFIG = jsonlite::toJSON(list(
        cluster = list(
          worker = paste(
            gsub(":[0-9]+$", "", barrier$address),
            8000 + seq_along(barrier$address), sep = ":")),
        task = list(type = 'worker', index = barrier$partition)
      ), auto_unbox = TRUE))

      if (is.null(tf_version())) install_tensorflow()

      strategy <- tf$distribute$MultiWorkerMirroredStrategy()

      result <- alexnet::imagenet_partition(partition = barrier$partition) %>%
        alexnet::alexnet_train(strategy = strategy, epochs = 10, parallel = 6)

      result$metrics$accuracy
  }, barrier = TRUE, columns = c(accuracy = "numeric"))
```

tensorflow.rstudio.com



Modelos

Como Tesla administra sus modelos?



Que es MLflow?

MLflow is an open source platform to manage the ML lifecycle, including experimentation, reproducibility, deployment, and a central model registry. MLflow currently offers four components:

MLflow Tracking

Record and query experiments: code, data, config, and results

Read more

MLflow Projects

Package data science code in a format to reproduce runs on any platform

Read more

MLflow Models

Deploy machine learning models in diverse serving environments

Read more

Model Registry

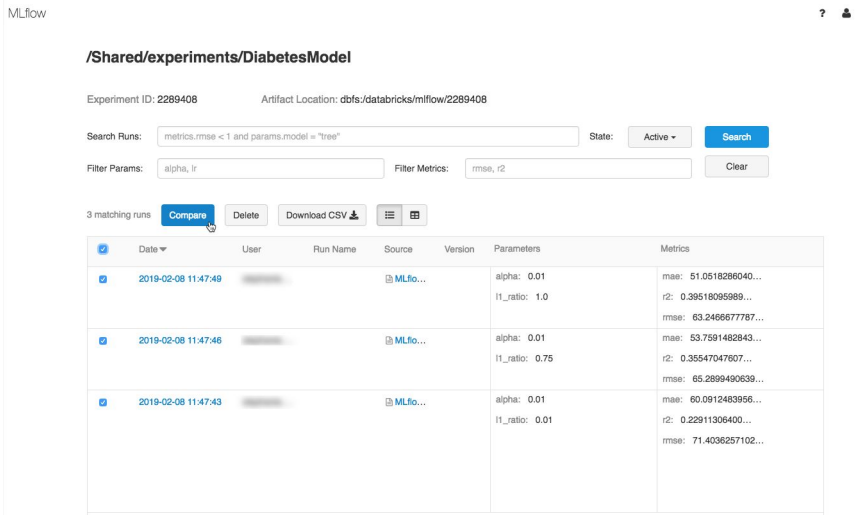
Store, annotate, discover, and manage models in a central repository

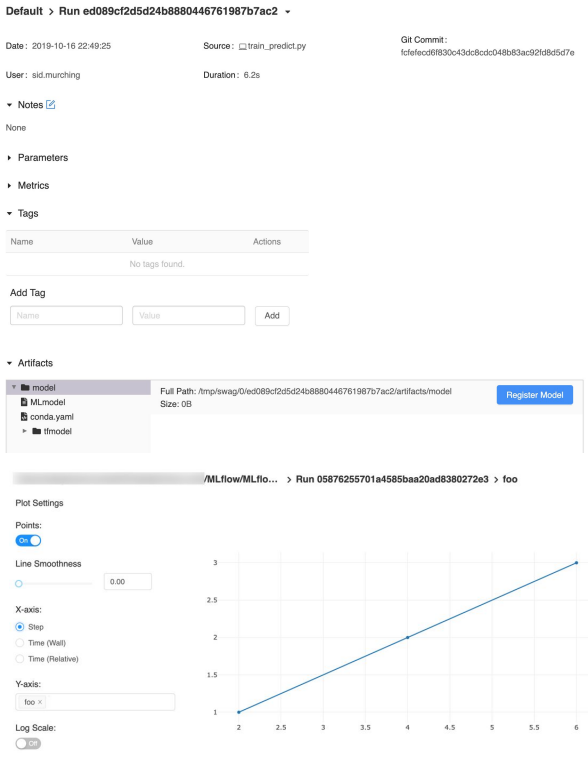
Read more

MLflow con R

```r
library(mlflow)
with(mlflow_start_run(), {
  # set experiments tags
  mlflow_set_tag("project", "wines")

  # log parameters
  mlflow_log_param("lambda", 0.5)

  # compute model
  lm(train, quality ~ .)

  # log metrics
  mlflow_log_metric("aic", metrics$aic())
})
```

Interface Web de MLflow





mlflow.org

mlflow.org/docs/latest/quickstart.html

Search

MLflow

- Quickstart
  - Installing MLflow
  - Downloading the Quickstart
  - Using the Tracking API
  - Viewing the Tracking UI
  - Running MLflow Projects
  - Saving and Serving Models
  - \+ Logging to a Remote Tracking Server
- Tutorials and Examples
- Concepts
- MLflow Tracking
- MLflow Projects
- MLflow Models
- MLflow Model Registry
- MLflow Plugins
- Command-Line Interface
- Search
- Python API

Using the Tracking API

The MLflow Tracking API lets you log metrics and artifacts (files) from your data science code and see a history of your runs. You can try it out by writing a simple Python script as follows (this example is also included in `quickstart/mlflow_tracking.py`):

Python R

```
library(mlflow)

# Log a parameter (key-value pair)
mlflow_log_param("param1", 5)

# Log a metric; metrics can be updated throughout the run
mlflow_log_metric("foo", 1)
mlflow_log_metric("foo", 2)
mlflow_log_metric("foo", 3)

# Log an artifact (output file)
writeLines("Hello world!", "output.txt")
mlflow_log_artifact("output.txt")
```

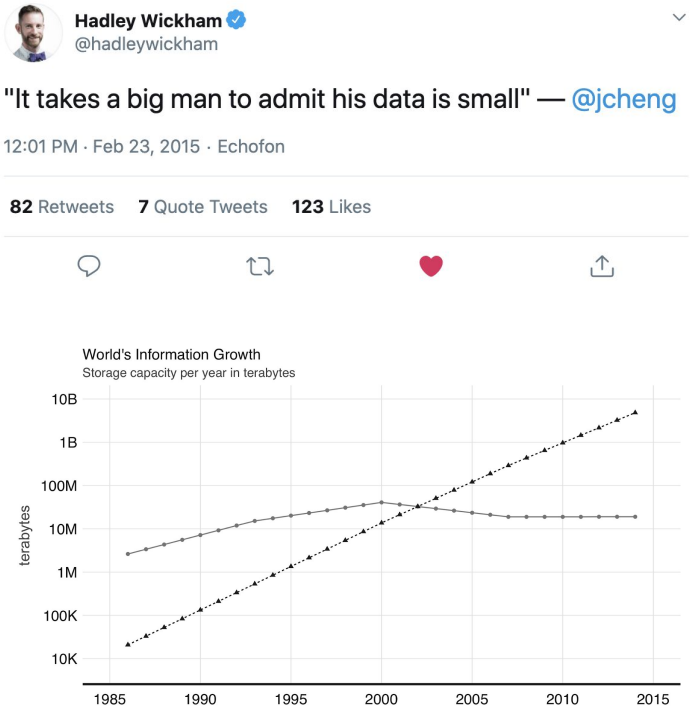
Viewing the Tracking UI

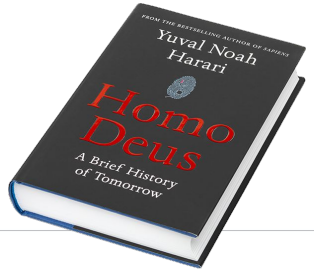
By default, wherever you run your program, the tracking API writes data into files into a local `./mlruns` directory. You can then run MLflow's Tracking UI:

Python R

Datos

“Those who own data own the future” -- Yuval Noah Harari





Dataism

From Wikipedia, the free encyclopedia

Not to be confused with Dadaism.

Dataism is a term that has been used to describe the mindset or philosophy created by the emerging significance of Big data. It was first used by David Brooks in *The New York Times* in 2013.[1] More recently, the term has been expanded to describe what social scientist Yuval Noah Harari has called an emerging ideology or even a new form of religion, in which 'information flow' is the 'supreme value'.[2]

Contents [hide]
1 History
2 Criticism
3 See also
4 References
5 External links

History [edit]

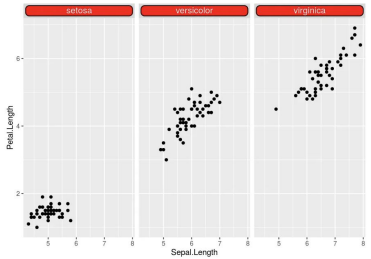
"If you asked me to describe the rising philosophy of the day, I'd say it is Data-ism", wrote David Brooks in *The New York Times* in February 2013.[1] Brooks argued that in a world of increasing complexity, relying on data could reduce cognitive biases and "illuminate patterns of behavior we haven't yet noticed".[1]

In 2015, Steve Lohr's book 'Data-ism' looked at how Big Data is transforming society, using the term to describe the Big Data revolution.[3][4]

In his 2016 book *Homo Deus: A Brief History of Tomorrow*, Yuval Noah Harari argues that all competing political or social structures can be seen as data processing systems: "Dataism declares that the universe consists of data flows, and the value of any phenomenon or entity is determined by its contribution to data processing" and "we may interpret the entire human species as a single data processing system, with individual humans serving as its chips."[2][5] According to Harari, a Dataist should want to "maximise dataflow by connecting to more and more media". Harari predicts that the logical conclusion of this process is that eventually humans will give algorithms the authority to make the most important decisions in their lives, such as whom to marry and which career to pursue.[6][7] Harari argues that Aaron Swartz could be called the "first martyr" of Dataism.[8]

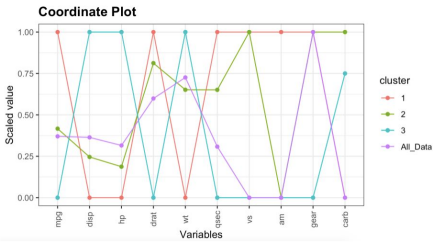
Qué problemas estamos resolviendo?

June 16th posts in R-bloggers use iris, mtcars, have missing data paths and require manually downloading datasets.




```
32.  ### IMPORT RAW DATA
33.
34.  log_info("Loading data")
35.
36.  mydat = fread(data_path)
37.
```

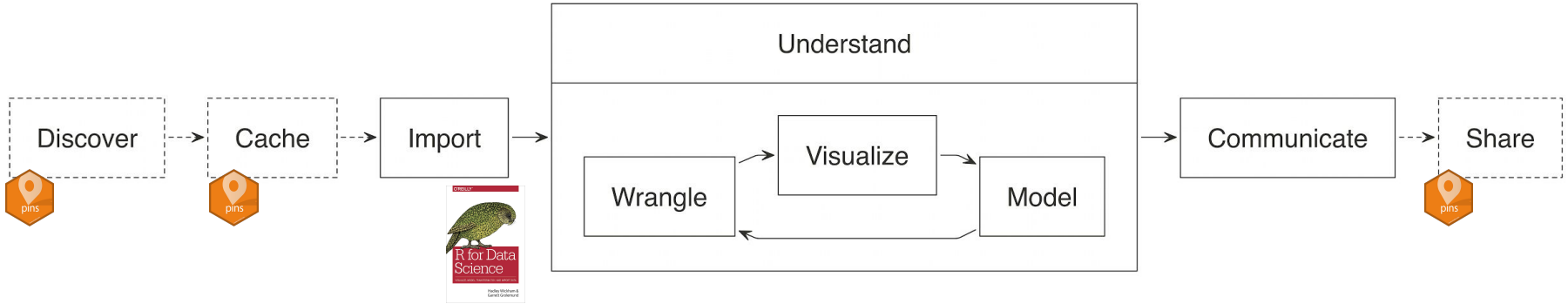
```
coord_plot(data=mtcars2, group_var="cluster",
group_func=median, print_table=TRUE)
```




```
# https://www.data.gouv.fr/fr/datasets/donnees-
hospitalieres-relatives-a-lepidemie-de-covid-19/
fichier_covid <- "donnees/covid.csv"
```

Por qué necesitamos un nuevo paquete?

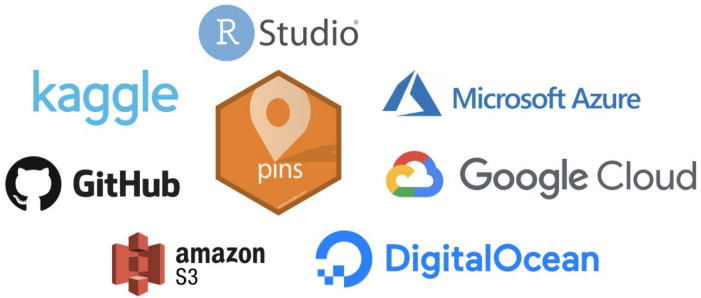
It is often also required for us to discover which dataset to use, cache it locally, and share our datasets with colleagues.



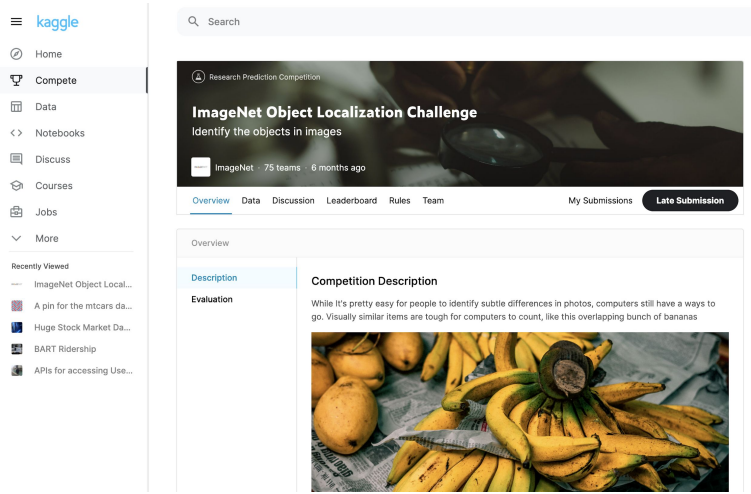
Pin, discover and share

Pin remote resources locally with pin(), work offline and cache results. **Discover** new resources across different boards using pin_find(). **Share** resources in local folders, GitHub, Kaggle, and RStudio Connect by registering new boards with board_register().

```
board_register("______")

pin(1:10, name = "numbers", board = "______")
pin_get("numbers")

pin("http://cs231n.stanford.edu/tiny-imagenet-200.zip",
    name = "tinyimagenet", board = "______")
pin_get("tinyimagenet")

pin_find()
```



Usando pins para descargar ImageNet



El siguiente código descarga **ImageNet**, un dataset de 150GB sobre clasificación de imágenes, de **Kaggle** usando el paquete **pins**.

```
library(pins)
board_register("kaggle", token = "kaggle.json")

pin_get("c/imagenet-object-localization-challenge", board = "kaggle")[1] %>%
  untar(exdir = "/localssd/imagenet/")
```

pins.rstudio.com

pins Home Articles Use Cases Reference News Blog

pins: Pin, Discover and Share Resources

Overview

You can use the `pins` package to:

- **Pin** remote resources locally with `pin()`, work offline and cache results.
- **Discover** new resources across different boards using `pin_find()`.
- **Share** resources in local folders, GitHub, Kaggle, and RStudio Connect by registering new boards with `board_register()`.

Installation

```
# Install the released version from CRAN:
install.packages("pins")
```

To get a bug fix, or use a feature from the development version, you can install pins from GitHub.

```
# install.packages("remotes")
remotes::install_github("rstudio/pins")
```

Usage

```
library(pins)
```

Links

Download from CRAN at
https://cloud.r-project.org/package=pins

Browse source code at
https://github.com/rstudio/pins

Report a bug at
https://github.com/rstudio/pins/issues

License

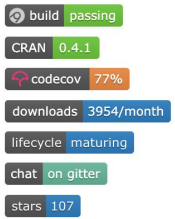
Apache License 2.0

Developers

Javier Luraschi
Author, maintainer

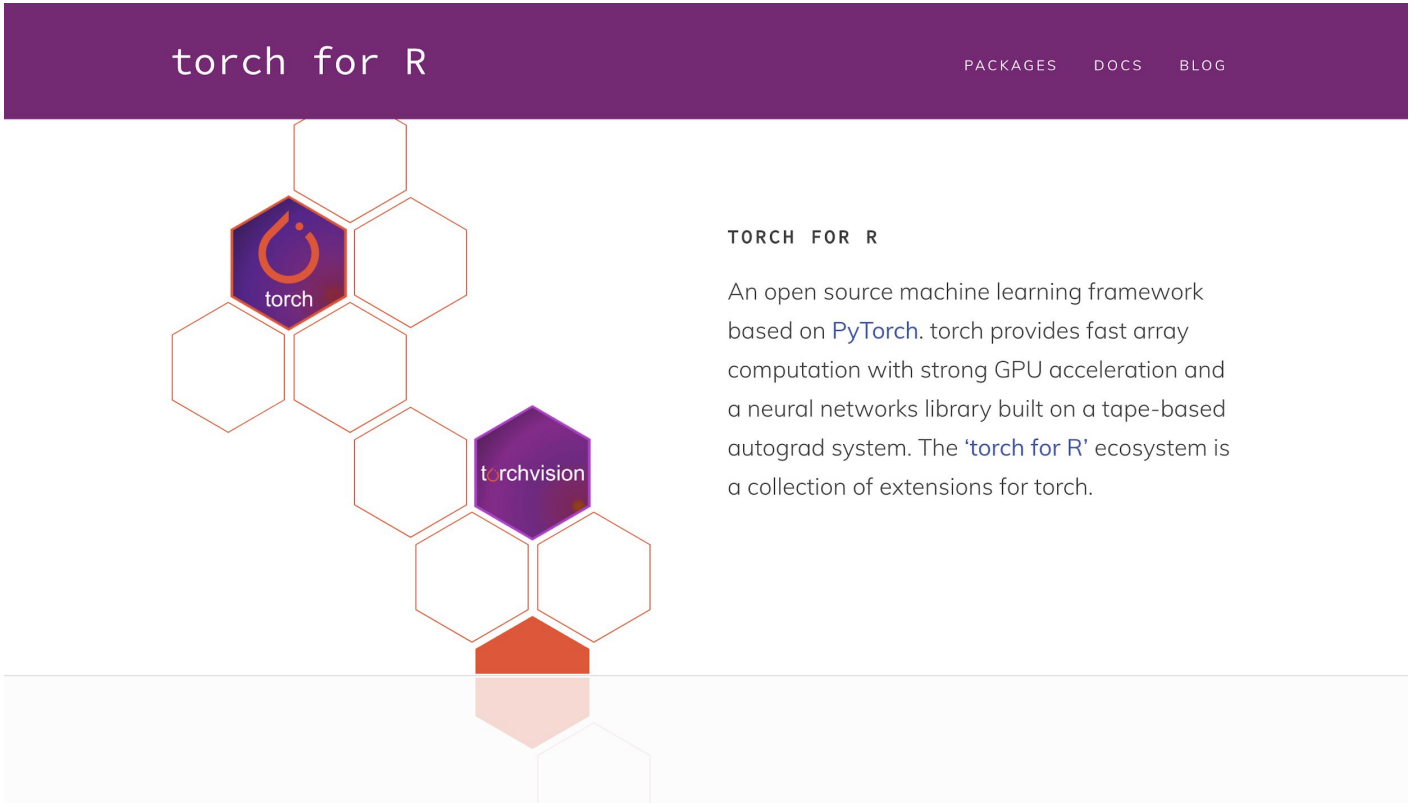
All authors...

Dev status



Deep Learning con PyTorch

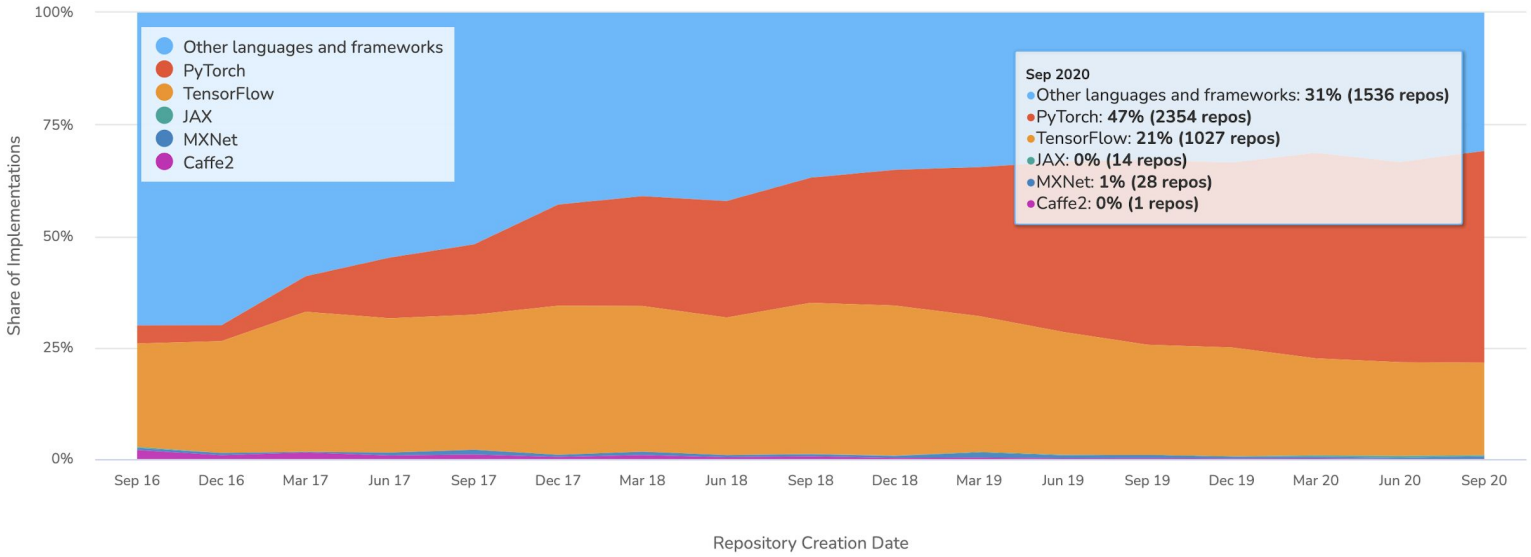
torch.mlverse.org



torch.mlverse.org

Frameworks

Paper Implementations grouped by framework



https://colab.research.google.com/drive/1NdiN9n_a7NEvFpvjPDvxKTshrSWgxZK5?usp=sharing

Gracias!

blogs.rstudio.com/ai


@dfalbel


@yl790


@javierluraschi

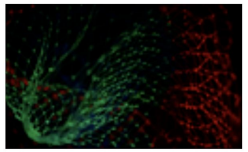

@zkajdan

RStudio AI Blog

Sept. 7, 2020

Introducing sparklyr.flint: A time-series extension for sparklyr

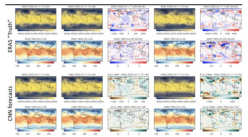
We are pleased to announce that sparklyr.flint, a sparklyr extension for analyzing time series at scale with Flint, is now available on CRAN. Flint is an open-source library for working with time-series in Apache Spark which supports aggregates and joins on time-series datasets.



Sept. 1, 2020

An introduction to weather forecasting with deep learning

A few weeks ago, we showed how to forecast chaotic dynamical systems with deep learning, augmented by a custom constraint derived from domain-specific insight. Global weather is a chaotic system, but of much higher complexity than many tasks commonly addressed with machine and/or deep learning. In this post, we provide a practical introduction featuring a simple deep learning



Aug. 24, 2020

Training ImageNet with R

This post explores how to train large datasets with TensorFlow and R. Specifically, we present how to download and repartition ImageNet, followed by training ImageNet across multiple GPUs in distributed environments using TensorFlow and Apache Spark.

Aug. 18, 2020

Deepfake detection challenge from R

A couple of months ago, Amazon, Facebook, Microsoft, and other contributors initiated a challenge consisting of telling apart real and AI-generated ("fake") videos. We show how to approach this challenge from R.

July 31, 2020

FNN-VAE for noisy time series forecasting

In the last part of this mini-series on forecasting with false nearest neighbors (FNN) loss, we replace the LSTM autoencoder from the previous post by a convolutional VAE, resulting in equivalent prediction performance but significantly lower training time. In addition, we find that FNN regularization is of great help when an underlying deterministic process is obscured by substantial noise.

SUBSCRIBE

Enjoy this blog? Get notified of new posts by email:

Email

Please check this box if you accept the RStudio privacy policy: *

Subscribe

Posts also available at r-bloggers

CATEGORIES

- Audio Processing (2)
- Bayesian Modeling (4)
- Cloud (3)
- Concepts (10)
- Data Management (2)
- Distributed Computing (4)
- Explainability (2)
- Image Recognition & Image Processing (13)
- Meta (3)
- Natural Language Processing (9)
- Packages/Releases (15)
- Privacy & Security (4)
- Probabilistic ML/DL (12)
- Tabular Data (5)
- TensorFlow/Keras (59)