Introducción a la librería Pytorch

Diego Andrade Canosa

Índice

- Introducción al curso
- Conceptos básicos de Redes de Neuronas Profundas (repaso)
- Introducción a Pytorch
- Lab1: Entorno del curso
- Referencias y materiales adicionales

- Conceptos clave de Pytorch
  - Tensores
    - Transformaciones
  - Variables y gradientes
  - Diferenciación automática
- Creación y composición de la arquitectura de red
- Los módulos nn.Module y nn.Sequential de Pytorch
  - Tipos de capas en Pytorch
  - Optimizadores y funciones de pérdida
  - Bucles de entrenamiento
    - Pasada Forward
    - Backpropagation

Introducción a la librería Pytorch

Pytorch: Génesis

- Librería creada en el año 2016 por FAIR (Facebook AI Research Lab)
  - Proyecto Adam Paszke (Internship)
  - Sucesor de la librería *torch (*basada en *Lua)*
  - Influenciado por la librería *chainer*
- Cronograma
  - Año 2016: Versión inicial
  - Año 2018: Versión 1.0 (fusión con Caffe)
  - Año 2019: Pytorch Lightning
  - Año 2023 (marzo!): Pytorch 2.0

Fuente

Fortalezas

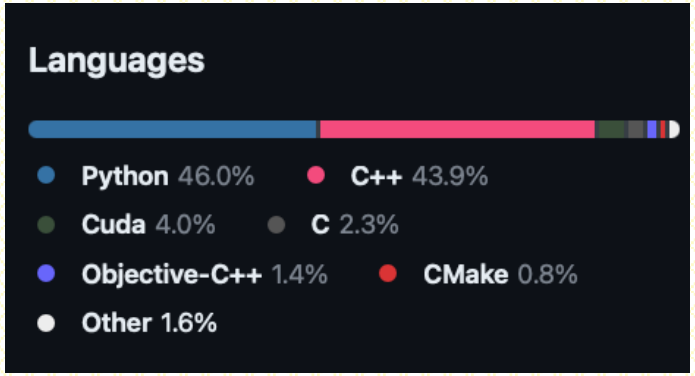
- Gran penetración en la comunidad investigadora
- Ventaja conceptual: Dynamic Computation Graph
- Facilidad de uso
- Flexibilidad
- Mucha base de código abierto en:
  - Visión por Computador
  - Procesado de Lenguaje Natural
  - Sistemas recomendadores
  - Procesado de señal

Ecosistema Pytorch

- Conjunto de herramientas y librerías que enriquecen y complementan al núcleo principal de Pytorch
- https://pytorch.org/ecosystem/
  - Domain-specific
    - NeMO
    - Diffusers
    - PennyLane
    - Transformers
  - HPC
    - Lightning
    - DeepSpeed
    - FairScale https://github.com/facebookresearch/fairscale
    - Accelerate
    - Ray
  - Other
    - TorchMetrics

Principios de diseño

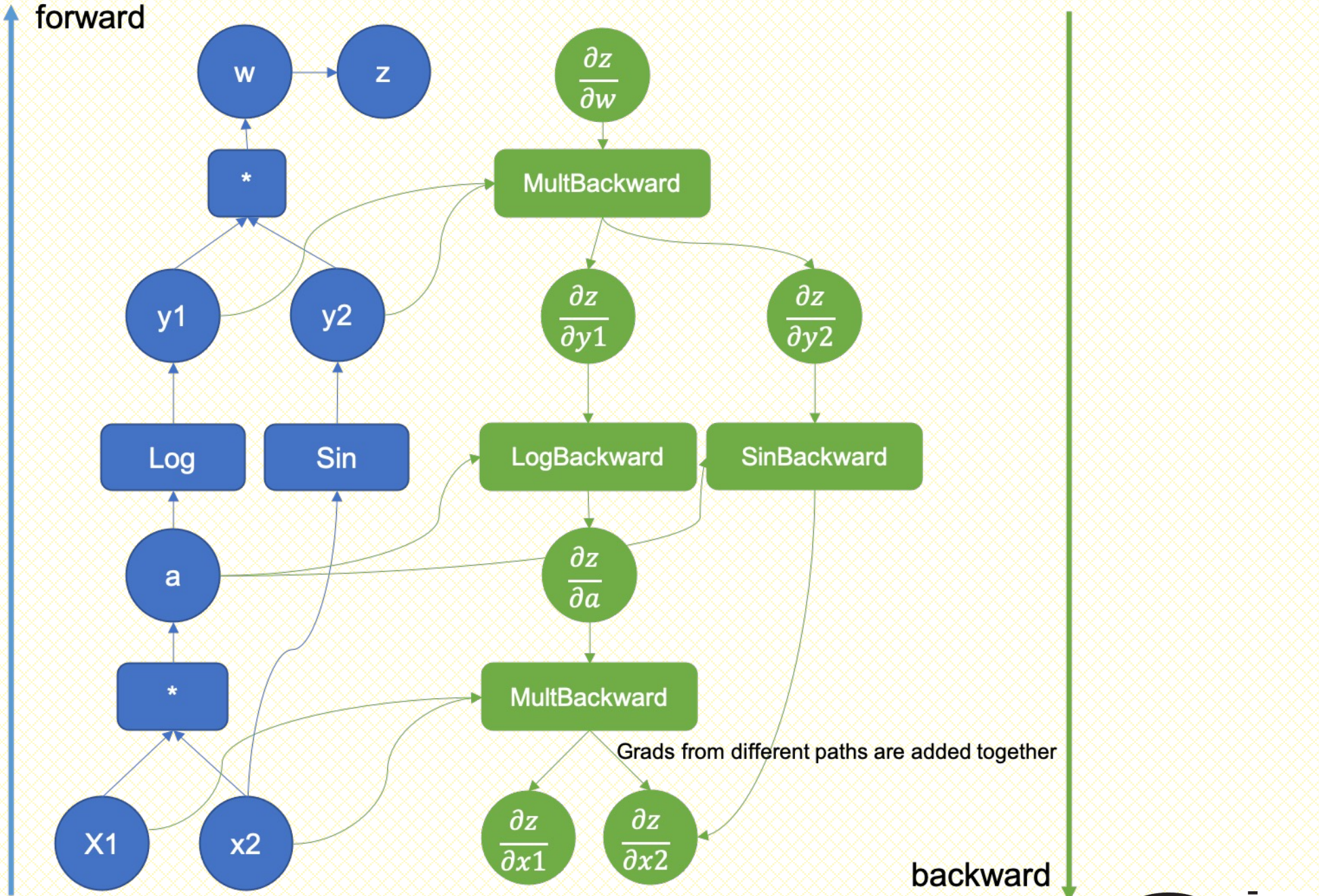
- Principio 1: Usable antes que rápido
- Principio 2: Simple antes que fácil
- Principio 3: Python primero



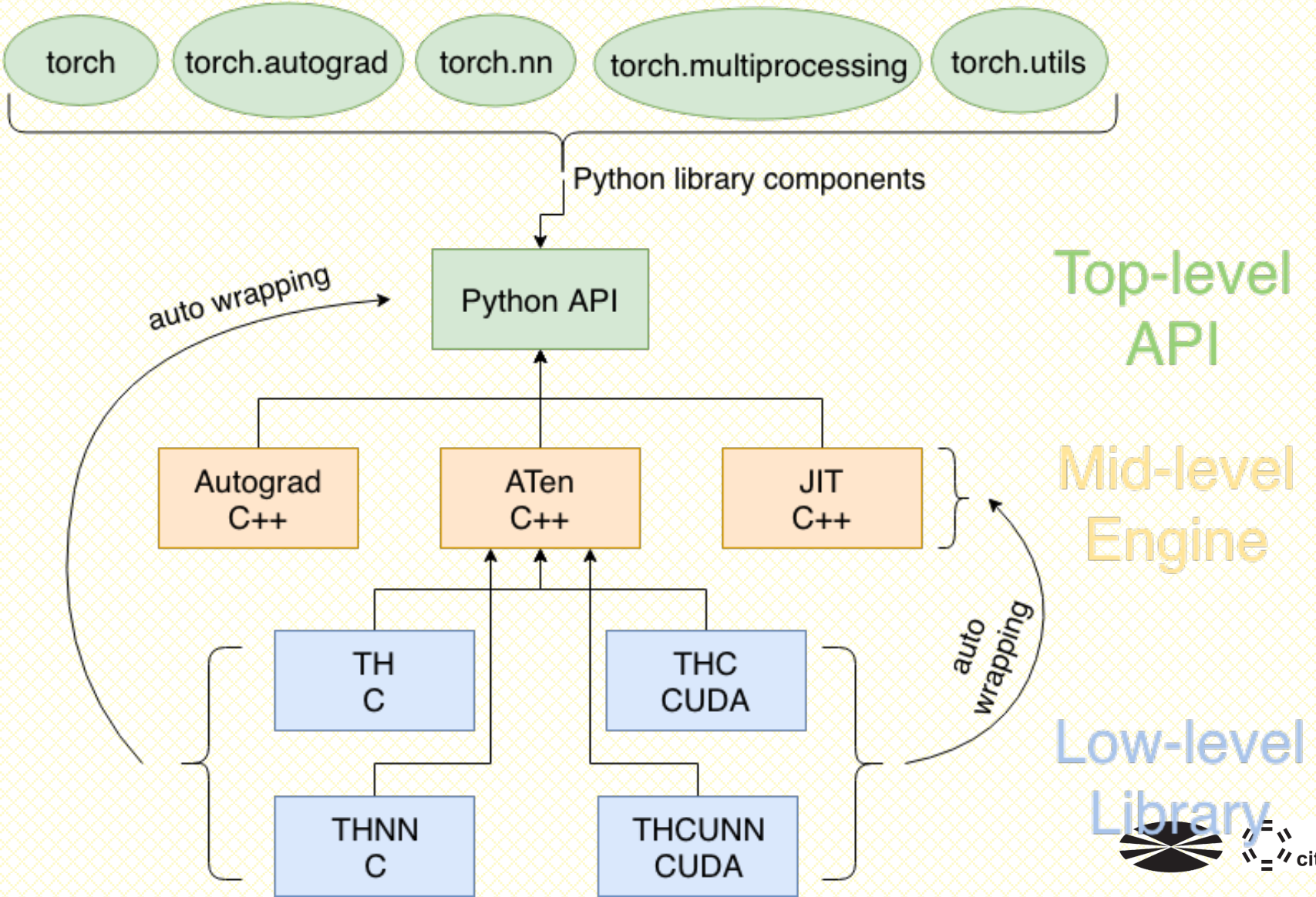
citic

Dynamic Computation Graph (DCG)

- Aproximación dinámica “define by run”, (vs “define and run”, estática)
- Más intuitivo y flexible que la aproximación estática (static computation graph)
- Menos posibilidades de optimización del cómputo
- Es una diferencia básica respecto a TF (<2.0)

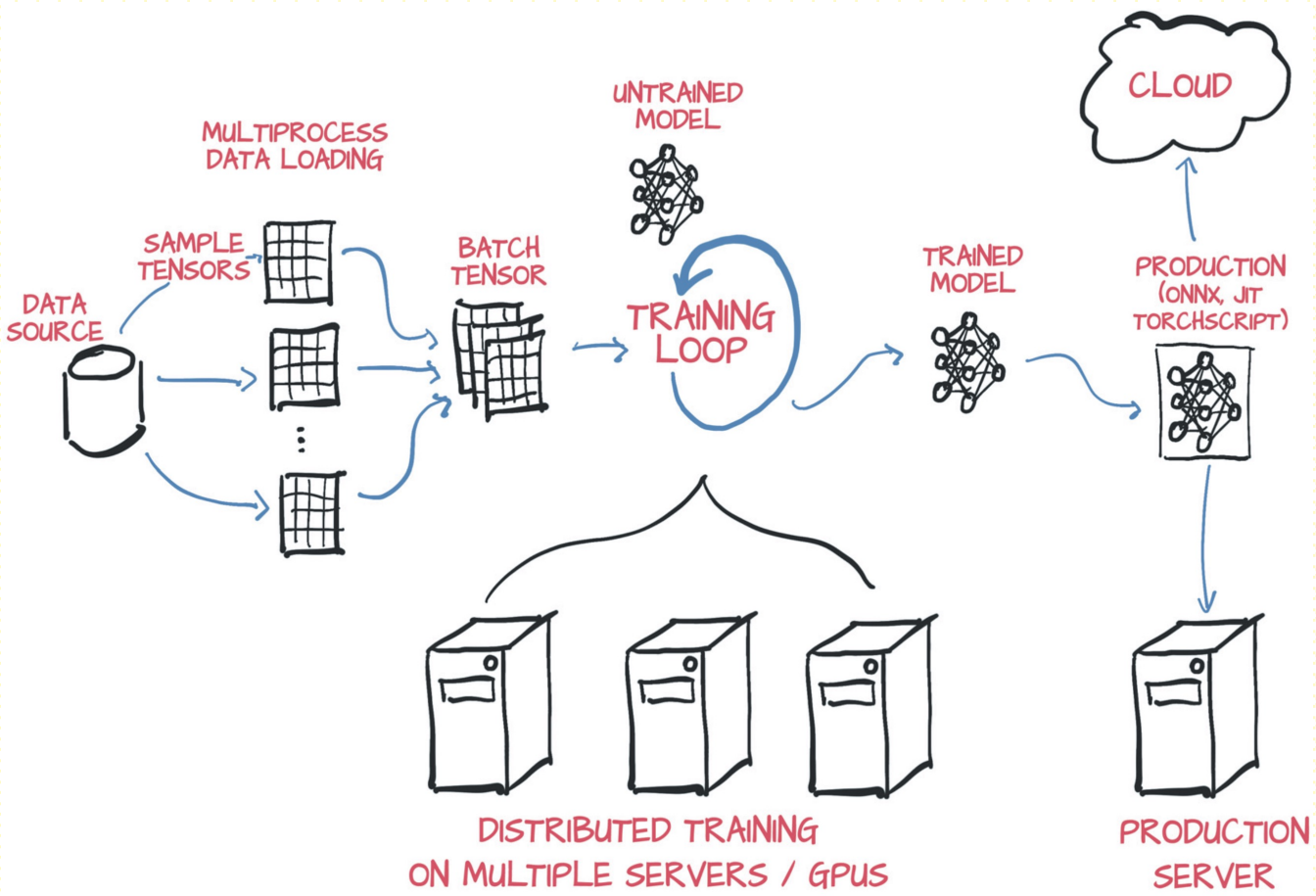

forward
w
z
*
MultBackward
y1
y2
Log
Sin
LogBackward
SinBackward
a
*
MultBackward
X1
x2
Grads from different paths are added together
backward

Estructura de módulos de Pytorch



citic

Pytorch: Uso



citic

Abstracciones

- Tensores: estructura de datos multidimensional donde se almacenan los diversos elementos del modelo: parámetros (pesos), bias, entradas, salidas
- Contenedores: estructuras a las que se asocian los distintos componentes de un modelo, así como su operación
- Optimizadores: artefacto que permite calcular los gradientes de los pesos en función de la salida de la función de pérdida
- Autograd: mecanismo que crea el DCG y permite calcular las derivadas parciales

Bloques constructores

- API Python
  - torch
  - torch.nn
- Librerías secundarias
  - Torchtext (Texto)
  - Torchaudio (Audio)
  - Torchvision (Visión por computador)
  - Torcharrow (Data frame, tabular data)
  - TorchData (Pipelines de procesado de datos)
  - TorchRec (Sistemas recomendadores)
  - TorchServe (Servidores)
  - Torchx (Lanzador para aplicaciones Pytorch)

torch

- Tensores
  - Definición, Creación, Indexación, Transformación, Random sampling, Serialización, Operadores matemáticos (aritméticos, lógicos, espectrales, Blas/lapack, operaciones foreach, otros)
- Utilidades
- Optimizaciones
- Configuración del engine

https://pytorch.org/docs/stable/torch.html

torch.nn

- Definición de capas
  - Convolucionales, Pooling, Padding, Normalization, Recurrent, Dropout…
- Contenedores (containers)
  - Module, Sequential
- Funciones
  - Distance, Loss, Quantized

https://pytorch.org/docs/stable/nn.html

TorchVision

- Paquete orientado a la resolución de problemas de visión por computador
- Proporciona:
  - Técnicas para transformar y aumentar imágenes (Geometría, color, composición, conversión, auto-augmentation)
  - Datapoints (Imágenes, Vídeo)
  - Modelos (Resnet, Mobilenet, Yolo, Inception, VGG, etc...)
  - Datasets (CIFAR, MNIST, ...)
  - Utilidades
  - Entrada/Salida (para imágenes y vídeos)
  - Extracción de características

TorchAudio

TorchAudio

- Paquete que agrupa varias utilidades relacionadas con el procesamiento de audio y de señal en general
- Proporciona:
  - Modelos (Conformer, HuBERT, DeepSpeech, etc...)
  - Datasets (CMU, GTZAN, LibriSpeech, etc...)
  - Pipelines: permiten usar modelos preentrenados para realizar tareas específicas
  - Componentes para la Entrada/Salida (StreamReader, StreamWriter, play_audio)

TorchText

TorchText

- Paquete que agrupa varias utilidades realizadas con el procesamiento de lenguaje
  - Es un desarrollo incipiente basado en la filosofía de otros paquetes más longevos como torchaudio y torchvision
- Proporciona:
  - Modelos (Roberta)
  - Utilidades (extraer archivo, descargar de url,...)
  - Transformaciones
  - Vocab (mapeado de palabras a índices)
  - Datasets
  - Elementos funcionales
  - Contenedores específicos (MultiheadAttentionContainer)

TorchArrow

- Conjunto de utilidades para el procesado de datos tabulares (Data Frame)
- Proporciona:
  - DataFrame (Creación, Inspección, Transformación, Estadísticas, Artimética)
  - Column: Estructura de datos unidimensional con datos de un único tipo de datos (numérico, string, list)

TorchData

- Paquete con componentes que permiten la construcción de pipelines para la carga de datos
- Proporciona:
  - Pipelines iterables (accesibles elemento a elemento)
  - Pipeline Map-style (accesibles clave-valor)
  - Utilidades
  - DataLoader2: Versión alternativa de torch.utils.data.DataLoader

TorchRec

- Paquete que contiene varias utilidades para construir sistemas recomendadores

TorchServe

- Paquete que contiene utilidades para la creación de servidores de modelos
- Proporciona:
  - API de gestión de modelos
  - API de inferencia
  - Soporte para distintas soluciones tecnológicas (Sagemaker, Mlflow, Kubeflow, etc...)

torchx

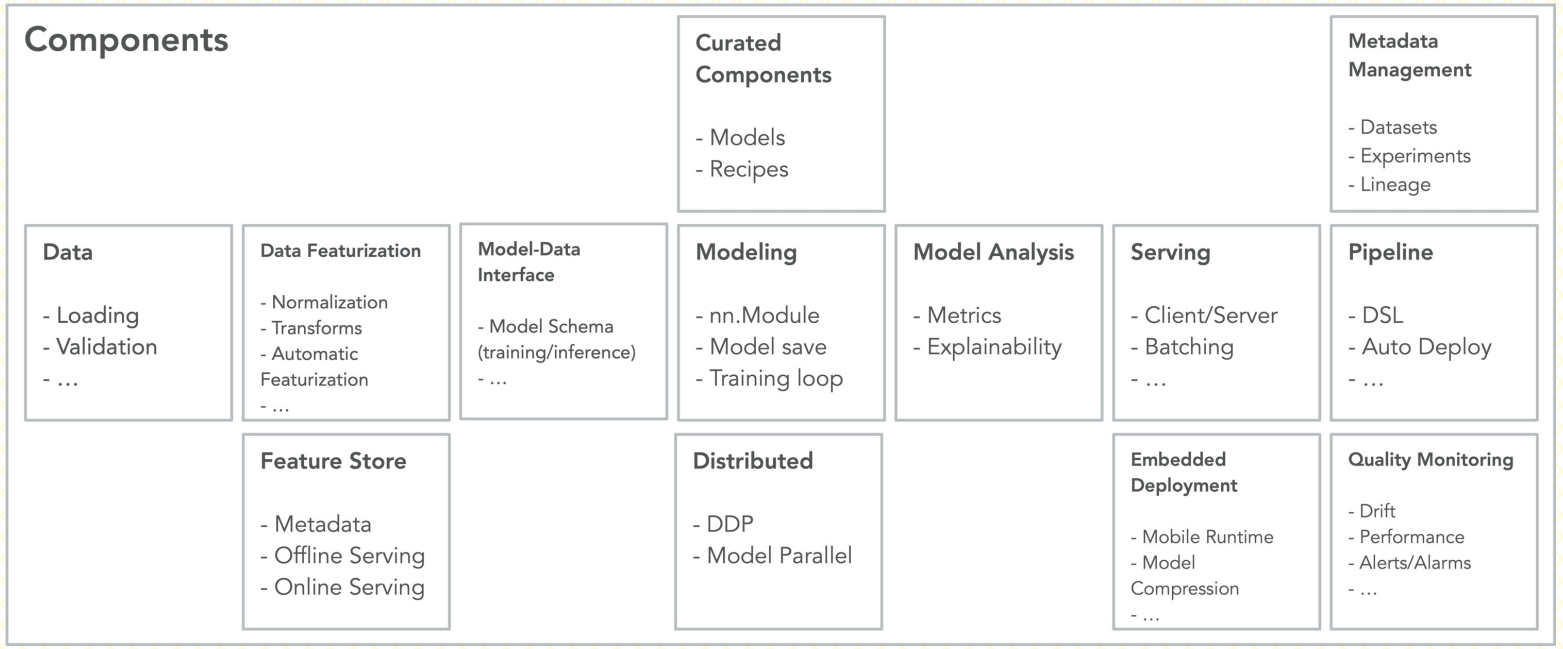
Lanzador para Pytorch

Tipo srun o qsub

Funciona con Slurm, Ray, AWS, Docker, Kubernetes, etc...

Permite ejecución local, remota y distribuida

https://pytorch.org/torchx/latest/quickstart.html



citic

Referencias y materiales adicionales

Introducción al Deep Learning

- https://developer.nvidia.com/blog/deep-learning-nutshell-core-concepts/
- https://mlu-explain.github.io
- https://aws.amazon.com/es/machine-learning/mlu/

Referencias

- Pytorch Internals: http://blog.ezyang.com/2019/05/pytorch-internals/
- Pytorch Design Philosophy: https://pytorch.org/docs/stable/community/design.html
- Pytorch: https://se.ewi.tudelft.nl/desosa2019/chapters/pytorch/
- A Tour of Pytorch Internals (Part I): https://pytorch.org/blog/a-tour-of-pytorch-internals-1/
- Pytorch CheatSheet: https://pytorch.org/tutorials/beginner/ptcheat.html
- Pytorch Deep Learning Framework: Speed+Usability: https://syncedreview.com/2019/12/16/pytorch-deep-learning-framework-speed-usability/