



Introduction to the Keras API

Mijael Martinez
Tulsa Data Science, March 2018

Part 0: Prerequisites

Before starting to use Keras

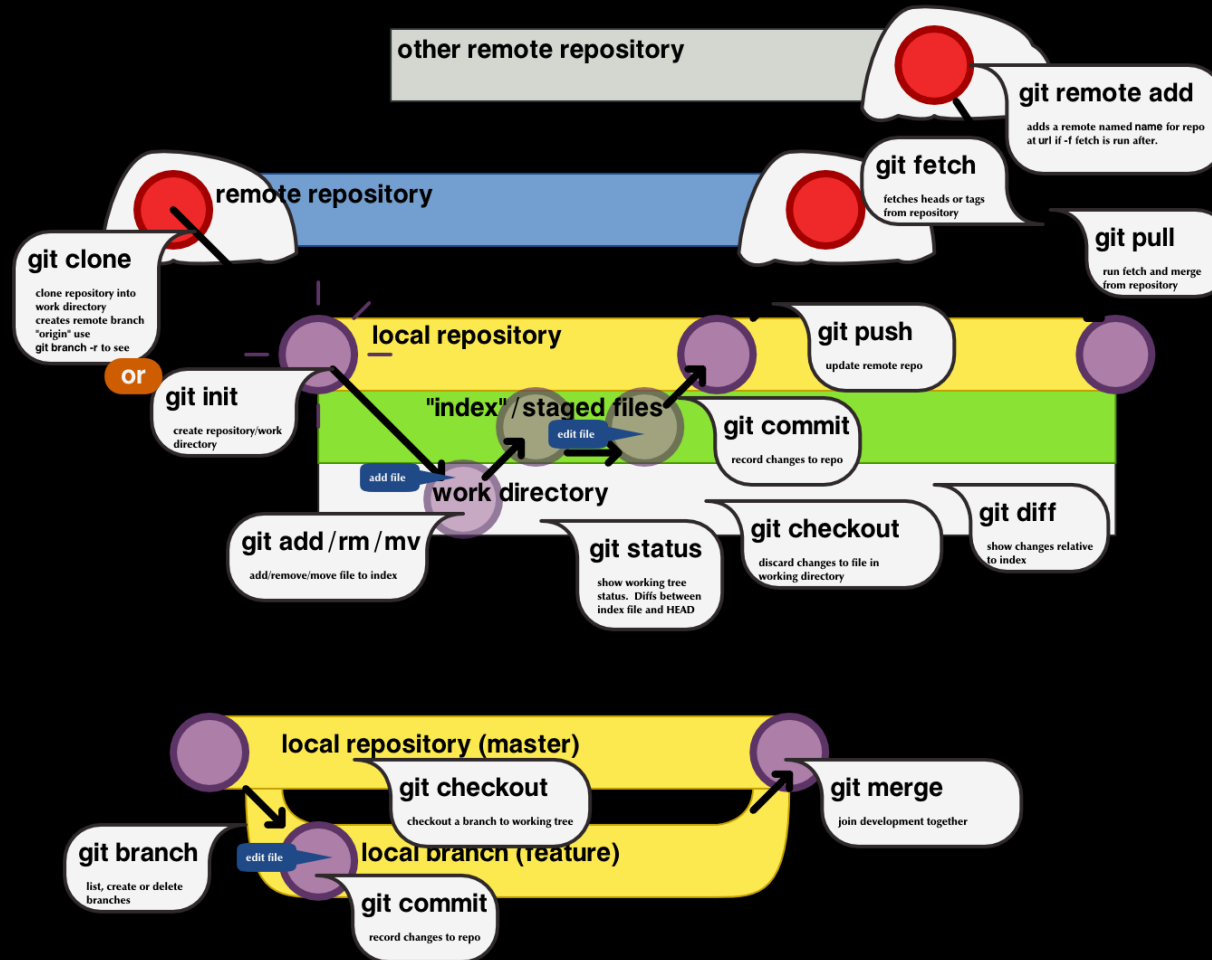
Git

- Version control, and distributed collaboration.
- Mac and Linux should have it pre-installed.
- Windows, download basic tool and client from: <https://git-scm.com/downloads>
 - Alternative (more complete) clients:
 - Sourcetree: <https://www.sourcetreeapp.com/>
 - GitHub Desktop: <https://desktop.github.com/>
- We'll use the command line.

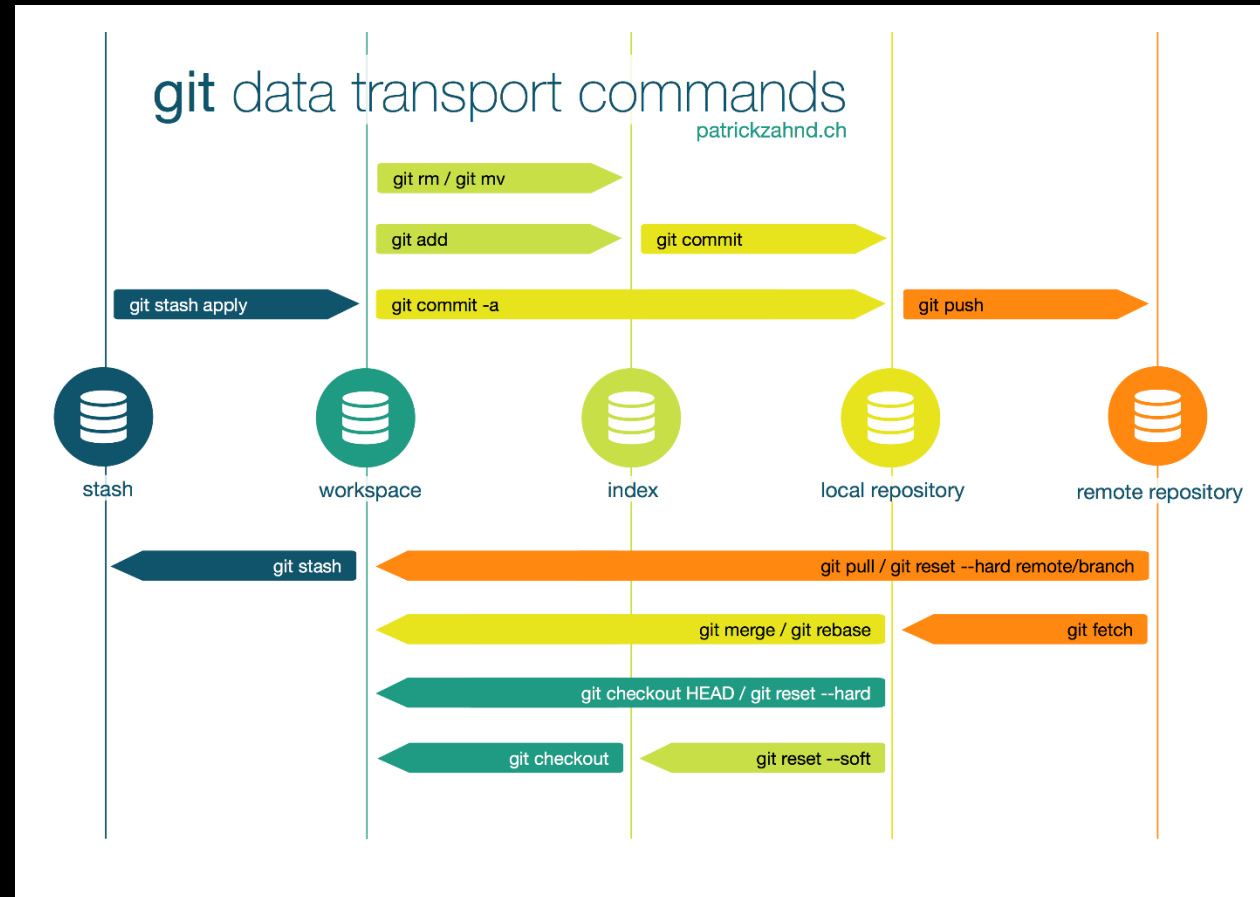
Git

- Open a command prompt window (terminal in Mac/Linux)
- Navigate to a folder of your choosing in your own computer.
 - E.g. “c:\TDS Projects”
- Run command to clone remote repository to your computer.
 - C:\TDS Projects>**git clone** <https://github.com/Tulsa-Data-Science/Basic-tutorials.git>
- Later on:
 - C:\TDS Projects>**git add .**
 - C:\TDS Projects>**git commit -m** “My own commit message.”

Git (Optional)



Git (Optional)



Git (Optional)

- More info:
 - Cheat sheet: <https://ndpsoftware.com/git-cheatsheet.html#loc=stash>
 - One-page cheat sheet: <https://jan-krueger.net/wordpress/wp-content/uploads/2007/09/git-cheat-sheet.pdf>
 - Visual Git Reference: <http://marklodato.github.io/visual-git-guide/index-en.html>

TensorFlow Installation (Windows)

- (These steps you already installed the Anaconda package manager.)
- Create a self-contained environment
 - C:\TDS Projects>conda create -n tensorflow pip python=3.5
- Activate your environment
 - C:\TDS Projects>activate tensorflow
- Install TensorFlow
 - C:\TDS Projects>pip install --ignore-installed --upgrade tensorflow

TensorFlow Installation (Mac)

- Install pip and virtualenv
 - ~/TDS Projects>`sudo easy_install pip`
 - ~/TDS Projects>`pip install --upgrade virtualenv`
- Create a self-contained environment
 - ~/TDS Projects>`virtualenv --system-site-packages -p python3 targetDirectory`
- Activate your environment
 - ~/TDS Projects>`cd targetDirectory`
 - ~/TDS Projects>`source ./bin/activate`
- Install TensorFlow
 - ~/TDS Projects> `easy_install -U pip`
 - ~/TDS Projects> `pip3 install --upgrade tensorflow`

TensorFlow Installation (Linux)

- Install pip and virtualenv
 - ~/TDS Projects>`sudo apt-get install python3-pip python3-dev python-virtualenv`
- Create a self-contained environment
 - ~/TDS Projects>`virtualenv --system-site-packages -p python3 targetDirectory`
- Activate your environment
 - ~/TDS Projects>`cd targetDirectory`
 - ~/TDS Projects>`source targetDirectory/bin/activate`
- Install TensorFlow
 - ~/TDS Projects> `easy_install -U pip`
 - ~/TDS Projects> `pip3 install --upgrade tensorflow`

Keras Installation

- (Optional, but recommended. Install later.)
 - HDF5 and h5py, for saving into disk.
 - (tensorflow) C:\TDS Projects>conda install h5py
 - graphviz and pydot, for visualization.
 - (tensorflow) C:\TDS Projects>conda install -c conda-forge graphviz
 - (tensorflow) C:\TDS Projects>conda install -c anaconda pydot
- (tensorflow) C:\TDS Projects>pip install keras

Let's test it!

Quick Example.

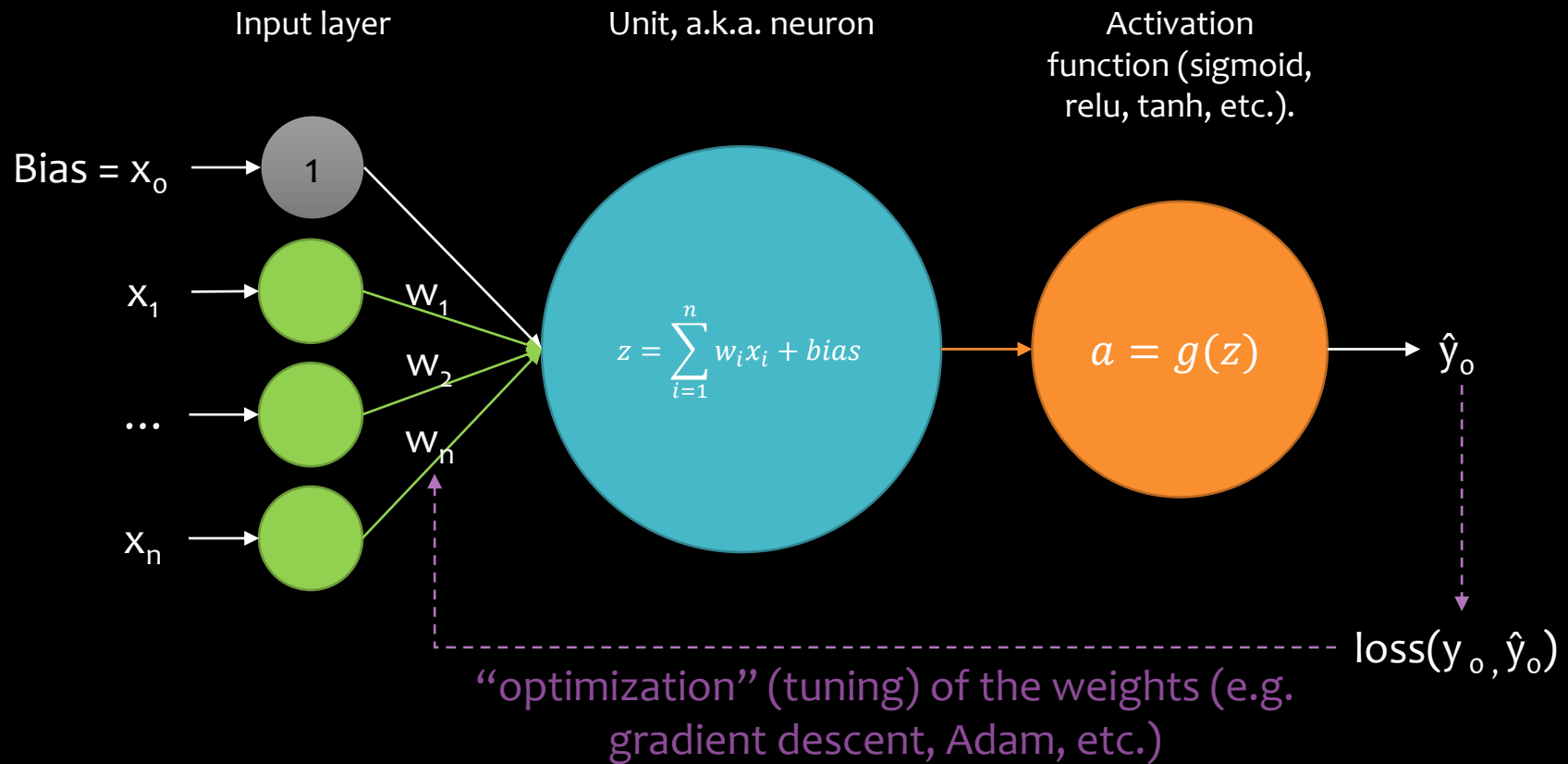
Run example notebook

- Go to the folder created after cloning the repository
- Open the sample notebook
 - (tensorflow) C:\TDS Projects>jupyter notebook "T00 - Keras Jumpstart.ipynb"
- Run each cell by selecting it and pressing "Shift + Enter"
 - You can also "run all cells."
 - "Ctrl + A"
 - "Ctrl + Enter"

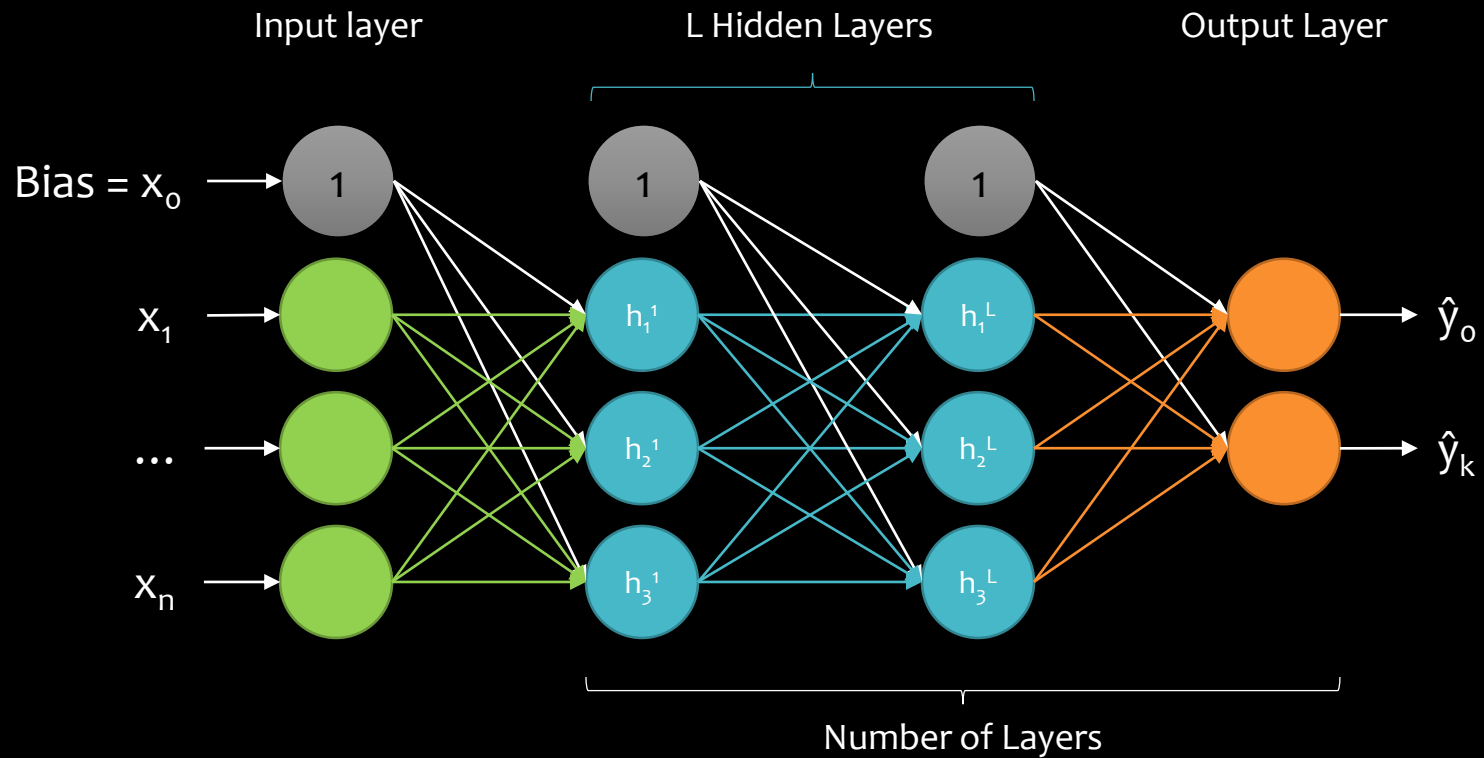
Part 1: Nomenclature

Basic terms in machine learning.

Basic Perceptron



Multiple layers



Vectors, Matrices and Tensors

- Vector:

- Row Vector of n elements: $\mathbf{v} = [v_1 \quad \cdots \quad v_n]$

- Column Vector of m elements: $\mathbf{v} = \begin{bmatrix} v_1 \\ \vdots \\ v_m \end{bmatrix}$

- Matrix:

- Matrix of m x n elements: $M = \begin{bmatrix} m_{11} & \cdots & m_{1n} \\ \vdots & \ddots & \vdots \\ m_{m1} & \cdots & m_{mn} \end{bmatrix}$

- Can be thought of as a vector of vectors:

$$M = [[m_{11} \quad \cdots \quad m_{1n}], \quad \dots, \quad [m_{m1} \quad \cdots \quad m_{mn}]]$$

- Tensor:

- General Arrangement of D dimensions, where each element: $t \in R^D$

- Can be thought of as a vector of vectors of vectors of ...

Other terms

- Hyperparameters:
 - Number of layers, type of layer, number of units per layer
 - Activation function
 - Learning rate, optimization method
- Epochs:
 - Number of times the training goes through the complete training data.
- Cross-validation data:
 - Subset of the data, not used for training, but for testing the network.

Part 2: Why Keras?

Keras and its interaction with a “back-end”.

Machine learning is Math

- The naïve approach performs each calculation one by one, within a multitude of loops (for, while, etc.).
- A first step in optimizing these calculations is “vectorizing” the algorithms
 - It means, performing as many of the operations using linear algebra (vectors, matrices, etc.).
 - This is where hardware such as GPUs and TPUs can excel.
 - $\mathbf{A} = g(\mathbf{Z}) = g(\mathbf{WX} + \mathbf{bias})$
- As the networks grow in complexity, even vectorization becomes complicated and cumbersome.
 - Libraries have been created to abstract these operations, and perform much of the boilerplate automatically.
 - Enter tools such as TensorFlow, Theano, Caffe, Pytorch, etc.

Machine learning is Math

- After a while, even these tools become cumbersome.
- It is necessary to make another abstraction:
 - Enter **Keras** (<https://keras.io/>).
 - An API to specify networks descriptively, and by their *hyperparameters*.
 - Uses TensorFlow, Theano or CNTK as its “back-end” (TensorFlow is the default).
 - You can also train the network, test it, and generate predictions.
 - You can still access the network details, if necessary.

Part 3: MNIST example

(Coming up)