Основы практического использования нейронных сетей.

Лекция 1. Средства работы с НС.

Дмитрий Буряк.
к.ф.-м.н.
dyb04@yandex.ru

Основные темы спецкурса

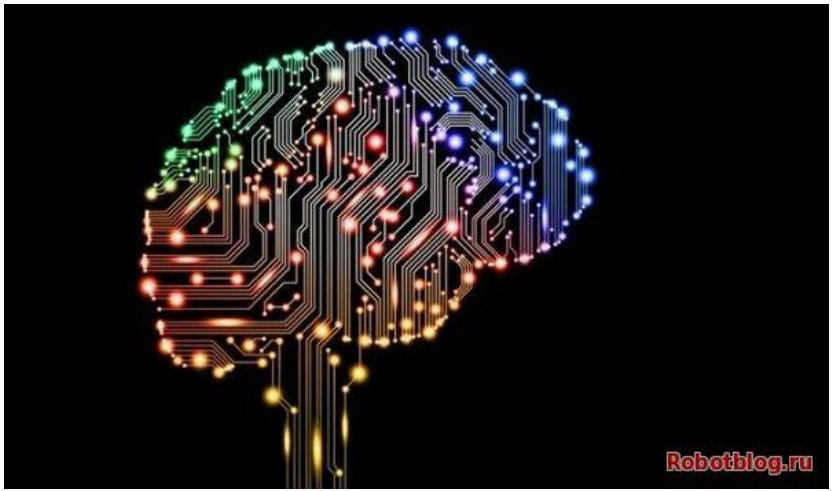
- Программные средства проектирования и реализации НС (Python, Matlab, библиотеки).
- Построение и обучение НС
  - Предобработка данных
  - Оптимизация гиперпараметров НС
  - Инициализация весов
  - Функция ошибки
  - Анализ, оптимизация обученной сети
- Особенности построения глубоких НС
  - Алгоритмы обучения
  - Проблемы при обучении
  - Методы регуляризации
- Обзор современных архитектур глубоких нейронных сетей.
- Практическое задание

Литература

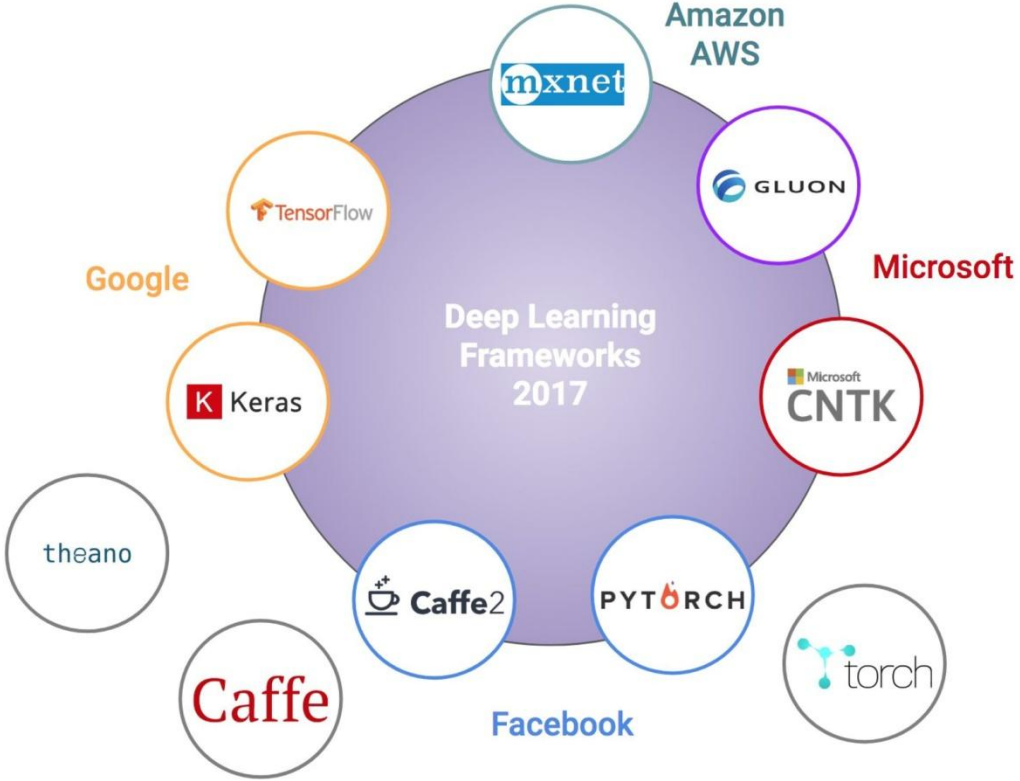
- http://www.deeplearningbook.org/
- http://cs231n.github.io/
- http://neuralnetworksanddeeplearning.com/index.html
- Ш. Франсуа. Глубокое обучение на Python, 2018.

Основные этапы разработки НС

- Подготовка данных
  - предобработка
  - разделение на выборки
- Проектирование архитектуры.
- Выбор алгоритма обучения.
- Инициализация весов.
- Выбор функции ошибки.
- Проведение обучения
- Тестирование полученной НС



Средства работы с НС



Средства работы с НС

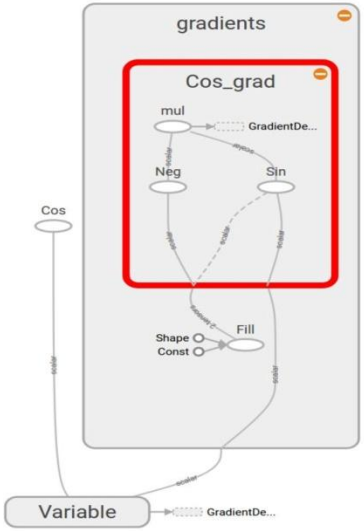
cuDNN

- Необходима низкоуровневая работа с памятью GPU (CUDA - функции)
- Реализованы базовые типы слоёв нейронной сети, градиенты, базовый тип данных - тензор
- Для практической работы с нейронными сетями требуется много времени на реализацию и отладку - практически всегда используются более высокоуровневые средства.
- Максимальная скорость работы по сравнению с более высокоуровневыми средствами

Tensorflow/Theano/Torch

- Автоматическое дифференцирование, вычисления представляются в виде вычислительного графа:

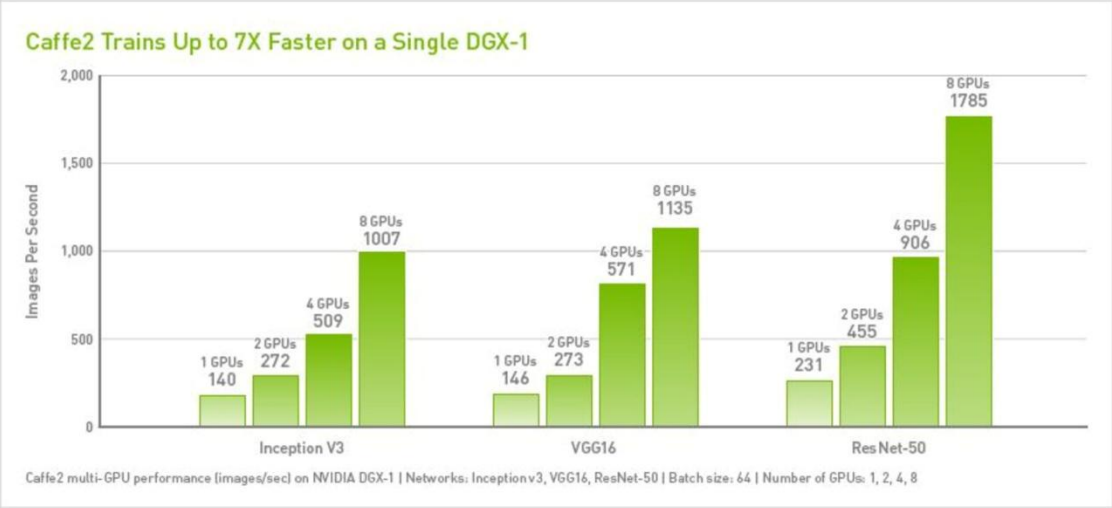
```
x = tf.Variable(initial_value=3.0)

y = tf.cos(x)

train = tf.train.GradientDescentOptimizer().minimize(y) with tf.Session() as sess:

writer = tf.summary.FileWriter('logs', sess.graph) writer.close()
```



- cuDNN используется для вычислений
- Программировать, обучать и тестировать сложные нейронные сети может быть всё еще слишком долго

MXNet/Caffe2

- MXNet:
  - API: C++, Python, Julia, R и др.
- Caffe2
  - API: Python, Matlab
  - Хорошая поддержка мульти-GPU:



CNTK/Pytorch

- Microsoft CNTK: удобна при работе на Windows
- Pytorch:
    - Набирает популярность в последнее время, быстро развивается
    - Некоторый трейд-офф между возможностями Tensorflow и удобством Keras
    - Уделяется внимание multi-GPU

Keras

- Удобное Python-API для работы с нейронными сетями
- В качестве вычислительного бэкенда используются Theano или Tensorflow
- Предобученные сети
- Большое число реализованных методов обучения, функций ошибки, типов слоёв нейронных сетей
- Есть возможности для создания собственных функций ошибки, слоёв, дополнительных действий во время обучения (механизм callbacks: например, сохранение весов на диск с некоторой периодичностью)

Keras -пример кода [1]

Загрузка данных:

```
batch_size = 128
num_classes = 10
epochs = 12
# input image dimensions
img_rows, img_cols = 28, 28

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

Keras -пример кода [2]

Построение сети:

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
```

Keras -пример кода [3]

Обучение сети:

```
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

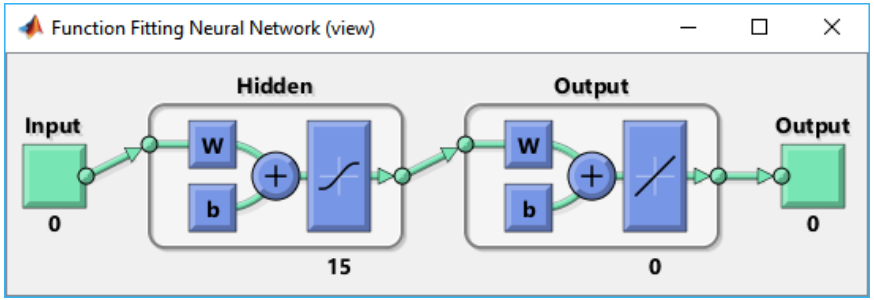
НС в среде Matlab

- ❑ Neural Network toolbox.
- ❑ Поддержка глубоких сетей с 2016г.
- ❑ Встроенные средства распараллеливания (требуется установка Parallel Computing Toolbox).
- ❑ Высокий уровень абстракции при обработке данных.
- ❑ Наличие различных средств обработки данных разной природы в той же среде.
- ❑ Ограниченный набор архитектур и алгоритмов обучения.
- ❑ Сложность внесения изменений в существующие архитектуры и алгоритмы обучения.

Matlab. Создание НС

```
% Загрузка данных
[x,t] = bodyfat_dataset;

% Создание НС
net = fitnet(15);
view(net)

% Разделение выборок
net.divideFcn='dividerand';
net.divideParam
```

t	1x252 double
x	13x252 double




```
Function Parameters for 'dividerand'

Training Ratio trainRatio: 0.7
Validation Ratio valRatio: 0.15
Test Ratio      testRatio: 0.15
```

Matlab. Предобработка данных

%Предобработка данных
net.inputs{1}

```
Neural Network Input

          name: 'Input'
  feedbackOutput: []
     processFcns: {'removeconstantrows', mapminmax}
   processParams: {1x2 cell array of 2 params}
processSettings: {1x2 cell array of 2 settings}
  processedRange: []
   processedSize: 0
           range: []
            size: 0
        userdata: (your custom info)
```

net.inputs{1}.processFcns{2}='mapstd';

net.outputs{2}

```
Neural Network Output

          name: 'Output'
   feedbackInput: []
   feedbackDelay: 0
    feedbackMode: 'none'
     processFcns: {'removeconstantrows', mapminmax}
   processParams: {1x2 cell array of 2 params}
processSettings: {1x2 cell array of 2 settings}
  processedRange: []
   processedSize: 0
           range: []
            size: 0
        userdata: (your custom info)
```

Matlab. Параметры обучения

```
% Выбор алгоритма обучения
net.trainFcn='trainrp';
net.trainParam
net.trainParam.max_fail=30;

net.initFcn
net.performFcn

% Регуляризация
net.performParam
```

```
Function Parameters for 'trainrp'

Show Training Window Feedback   showWindow: true
Show Command Line Feedback showCommandLine: false
Command Line Frequency                show: 25
Maximum Epochs                      epochs: 1000
Maximum Training Time                 time: Inf
Performance Goal                      goal: 0
Minimum Gradient                   min_grad: 1e-05
Maximum Validation Checks          max_fail: 6
Initial Delta                        delta0: 0.07
Delta Increase                     delt_inc: 1.2
Delta Decrease                     delt_dec: 0.5
Maximum Delta                      deltamax: 50
```

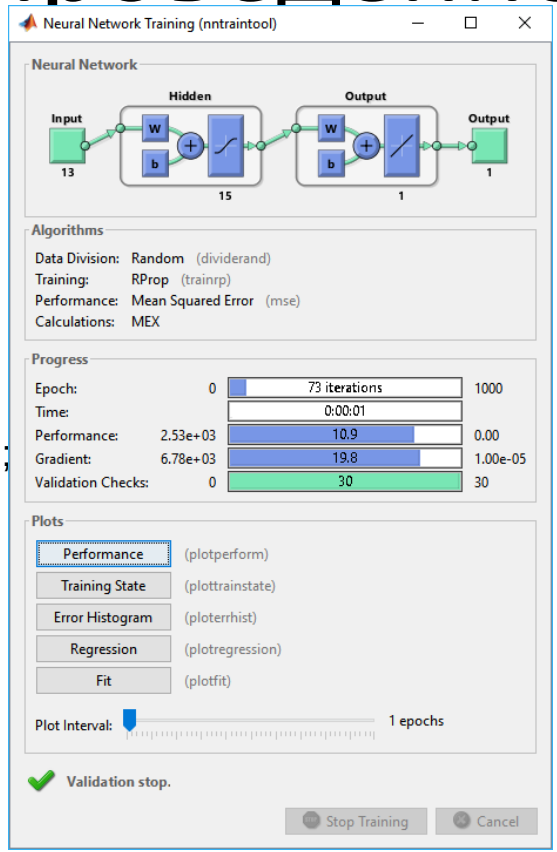
```
ans =

initlay


ans =

mse


ans =


    Function Parameters for 'mse'

    Regularization Ratio regularization: 0
    Normalization         normalization: 'none'
```

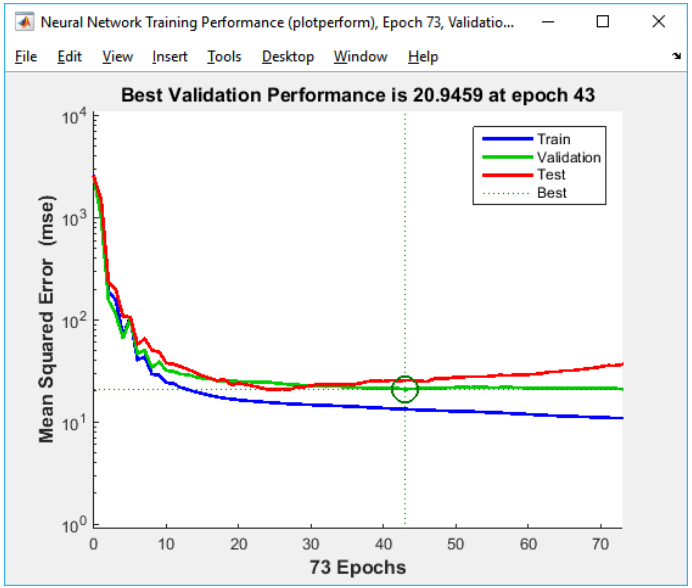
Matlab. Проведение обучения

```
[net,tr] = train(net,x,t);
```

```
% Многоядерная система
pool = gcp;
pool.NumWorkers
net2 =
        train(net1,x,t,'useParallel','yes');
```

```
% GPU
gpuDeviceCount
gpuDevice
gpuDevice(1)
net2 = train(net1,x,t,'useGPU','yes');
```





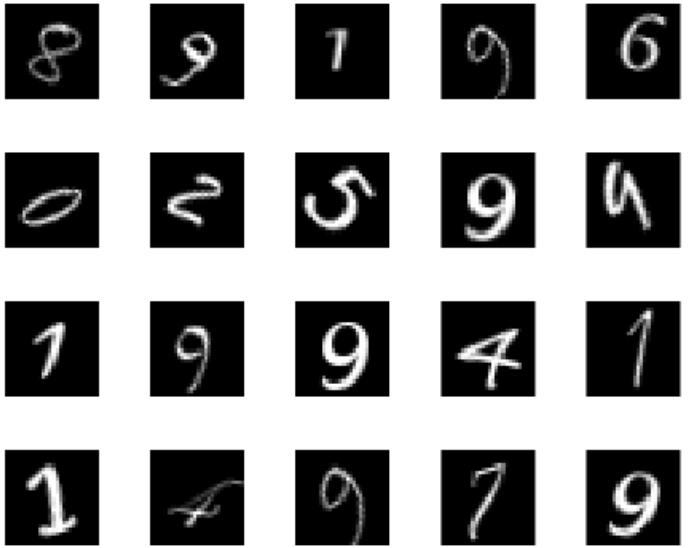
Matlab. Применение НС

```
% Вычисление NN
r=net(x(:,tr.testInd));
perf = mse(net,t(tr.testInd),r)
```

```
perf =

    25.6104
```

Matlab. Глубокие НС. Распознавание цифр.

```
% Задать путь к базе изображений
digitDatasetPath =
fullfile(matlabroot,'toolbox','nnet','nndemos', ...
        'nndatasets','DigitDataset');

% Загрузить изображения вместе с метками классов
digitData = imageDatastore(digitDatasetPath, ...
        'IncludeSubfolders',true,'LabelSource','foldernames');


% Сформировать обучающую выборку
trainingNumFiles = 750;
[trainDigitData,testDigitData] = splitEachLabel(digitData, ...
                                    trainingNumFiles,'randomize');
```



Matlab. Глубокие НС. Построние сети.

```
% Формирование параметров слоев
layers = [ imageInputLayer([28 28 1])
           convolution2dLayer(5,20)
           reluLayer
           maxPooling2dLayer(2,'Stride',2)
           fullyConnectedLayer(10)
           softmaxLayer
           classificationLayer() ];
```

- Входной слой 28x28
- Сверточный слой 20 фильтров 5x5
- Слой ReLU
- Слой понижения размерности, 2x2, шаг 2
- Полносвязанный слой, 10 линейных нейронов
- Слой SoftMax
- Выходной слой. Определяет целевую функцию при обучении (cross entropy)

Matlab. Глубокие НС. Обучение.

```
% Задание параметров алгоритма обучения
>> options = trainingOptions('sgdm', ...
                'MaxEpochs',15, ...
                'InitialLearnRate',0.0001, ...
                ‘L2Regularization’, 0.0001, …
                 ‘MiniBatchSize’, 128, …
                 ‘ExecutionEnvironment’, ‘auto’);
>> convnet =
trainNetwork(trainDigitData,layers,options);
```

→ Алгоритм обучения SGD с моментом

→ Количество итераций обучения

→ Начальный коэффициент обучения

→ Коэффициент регуляризации по норме L2

→ Размер пакета

→ Определение типа вычислителя (‘cpu’ , ‘gpu’, ‘multi-gpu’);

Matlab. Глубокие НС. Обучение. Тестирование.

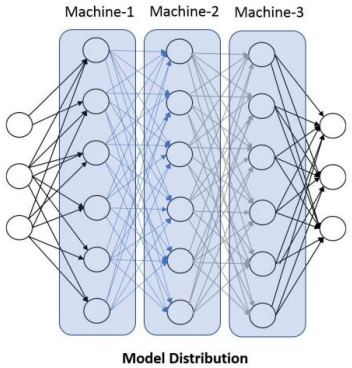
```
% Проведение обучения
convnet =
trainNetwork(trainDigitData, layers, options);
```

```
% Тестирование НС
YTest = classify(convnet,testDigitData);
TTest = testDigitData.Labels;
>> accuracy =
     sum(YTest == TTest)/numel(TTest);
```

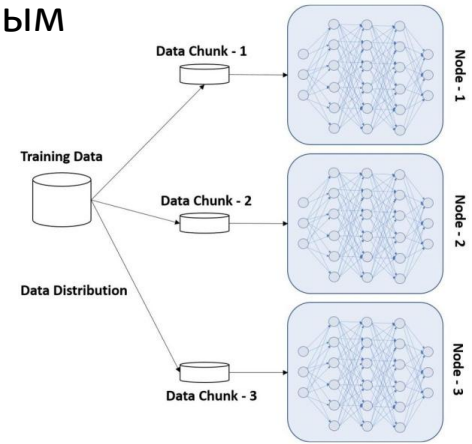
```
Training on single CPU.
Initializing image normalization.
|=========================================================================================|
|     Epoch    |   Iteration  | Time Elapsed |  Mini-batch  |  Mini-batch  | Base Learning|
|              |              |  (seconds)   |     Loss     |   Accuracy   |     Rate     |
|=========================================================================================|
|           1 |           1 |         0.96 |       3.2134 |       11.72% |     1.00e-04 |
|           1 |          50 |        20.37 |       1.1249 |       64.84% |     1.00e-04 |
|           2 |         100 |        39.53 |       0.7620 |       75.00% |     1.00e-04 |
|           3 |         150 |        58.92 |       0.3204 |       92.97% |     1.00e-04 |
|           4 |         200 |        78.05 |       0.4273 |       85.16% |     1.00e-04 |
|           5 |         250 |        96.93 |       0.2355 |       93.75% |     1.00e-04 |
|           6 |         300 |       116.23 |       0.1795 |       95.31% |     1.00e-04 |
|           7 |         350 |       135.32 |       0.1219 |       97.66% |     1.00e-04 |
|           7 |         400 |       154.47 |       0.0803 |       99.22% |     1.00e-04 |
|           8 |         450 |       173.61 |       0.0992 |       96.88% |     1.00e-04 |
|           9 |         500 |       192.70 |       0.0827 |       99.22% |     1.00e-04 |
|          10 |         550 |       210.81 |       0.0486 |      100.00% |     1.00e-04 |
|          11 |         600 |       230.26 |       0.0339 |      100.00% |     1.00e-04 |
|          12 |         650 |       249.73 |       0.0521 |       99.22% |     1.00e-04 |
|          13 |         700 |       268.57 |       0.0430 |       99.22% |     1.00e-04 |
|          13 |         750 |       287.62 |       0.0300 |      100.00% |     1.00e-04 |
|          14 |         800 |       306.71 |       0.0528 |       99.22% |     1.00e-04 |
|          15 |         850 |       326.26 |       0.0228 |      100.00% |     1.00e-04 |
|          15 |         870 |       334.22 |       0.0185 |      100.00% |     1.00e-04 |
|=========================================================================================|
```

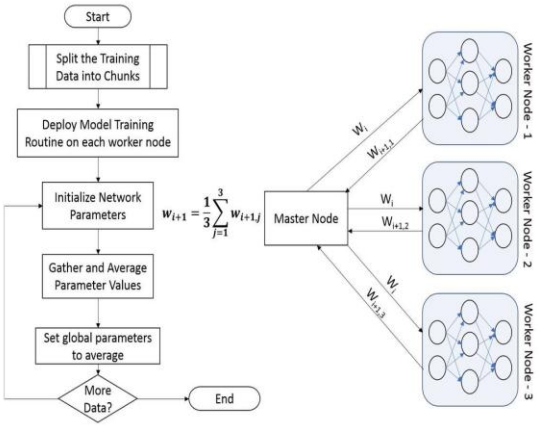
Распределенные вычисления НС

- Распараллеливание модели

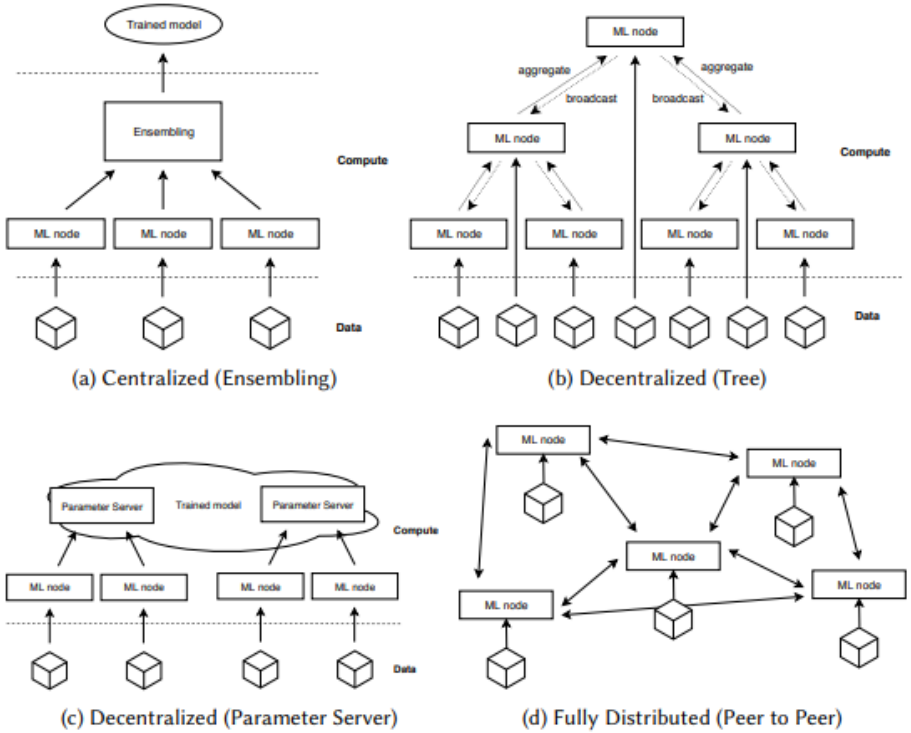


- Распараллеливание по данным





Реализация распределенных вычислений



J.Verbraeken, et al., A Survey on Distributed Machine Learning, 2019

Распределенное обучение

- Ансамбль независимых моделей
  - Все популярные библиотеки DL
- Параллельный синхронный SGD
  - Применение операций AllReduce, модель находится на центральном сервере;
  - Horovod (применяет NVIDIA Collective Communication Library – NCCL); Caffe 2 (NCCL); CNTK (реализована квантизация градиента).
- Параллельный асинхронный SGD
  - Сервер параметров, модель имеет распределенный характер, представление НС в виде графа (TensorFlow, MxNet), возможность реализации распараллеливания модели;
  - TensorFlow, MxNet, CNTK+Distributed Machine Learning Toolkit (DMTK)

Обзор средств разработки(1)

Name	Papers	API	Distribution and Parallelization	Community
Caffe	[80]	CLI, Python, Matlab	No native support for distribution.	Github: 2 StOv: 2,750
Caffe2	n/a	C++, Python	• Decentralized only • Synchronous only • Model quantization supported • Gradient quantization not supported • Communication scheduling not supported	Github: n/a StOv: 116
Chainer	[7, 174]	Python	• Decentralized only • Synchronous only • Model quantization not supported • Gradient quantization not supported • Communication scheduling not supported	Github: 3,939 StOv: 132
CNTK	[158]	C++, C#, Python, Brain-Script	• Centralized and decentralized • Bounded asynchronous training via BMUF [23] • Model quantization not supported • 1-bit gradient quantization [159] supported • Communication scheduling not supported	Github: 138 StOv: 488
DL4j	n/a	Java	• Centralized and decentralized • Synchronous and asynchronous • Model quantization not supported • Modified 1-bit gradient quantization by Strom [43, 167] supported • Communication scheduling not supported	Github: 390 StOv: 243

R. Mayer and H-A Jacobsen, Scalable Deep Learning on Distributed Infrastructures: Challenges, Techniques and Tools, 2019

Обзор средств разработки(2)

Keras	n/a	CNTK, DL4j, TensorFlow, Theano	• Model quantization supported • Higher-level concepts must be implemented in the DL framework that employs Keras	Github: 310 StOv: 14,630
MXNet	[24]	C++, Go, Java-Script, Julia, Matlab, Perl, Python, R, Scala, Wolfram	• Centralized only • Synchronous and asynchronous • Model quantization supported • 2-bit gradient quantization with error-feedback supported [124] • Communication scheduling not supported	Github: 837 StOv: 455
PyTorch	[139]	C++, Python	• Centralized and decentralized • Synchronous and asynchronous • Model quantization not supported • Gradient quantization not supported • Communication scheduling not supported	Github: 3,484 StOv: 2,413
SINGA	[134]	C++, Python	• Centralized and decentralized • Synchronous and asynchronous • Model quantization not supported • Gradient quantization not supported • Communication scheduling not supported	Github: 44 StOv: 0
TensorFlow	[4]	C++, Go, Java, Java-Script, Python, Swift	• Centralized • Synchronous and asynchronous • Model quantization supported • Gradient quantization not supported • Communication scheduling not supported	Github: 10,930 StOv: 39,334
Theano	[15]	Python	No native support for distribution.	Github: 55 StOv: 2,389

R. Mayer and H-A Jacobsen, Scalable Deep Learning on Distributed Infrastructures: Challenges, Techniques and Tools, 2019