AIOLI - Automatic Music Classification with PyTorch and TensorFlow

Before we start

To participate in the hands-on coding session afterwards, please:

- Make sure you've got Python 2.7 installed
- Download the dataset: `bit.ly/2i0LkQs`
- Download the example code: `github.com/aioli-ffm/music-projects`
- Follow the instructions in: `simple_music_classifier/README.md`

Agenda

Before we start

Agenda

Task and Data

Basics of neural networks

Playing with code

Our task

Our Task: Teaching a computer program to distinguish musical genres

`Audience participation:  Can you guess the genre?`

Our data

What data will we feed into our program?

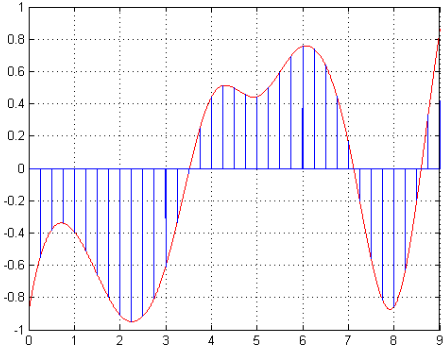
From what we've discovered so far: 2s of audio are sufficient to distinguish the musical genre of most cases.

What data do we expect the program to produce?

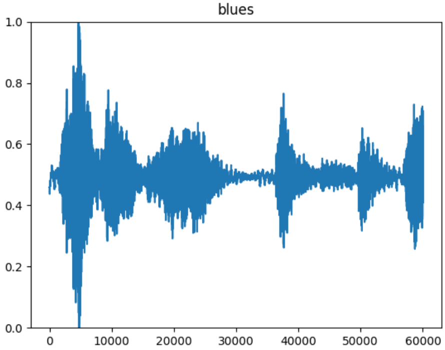
A probability for each genre the program knows; How likely the program thinks it is a given input matches a certain musical genre.

Input representation

Sampling of continous audio

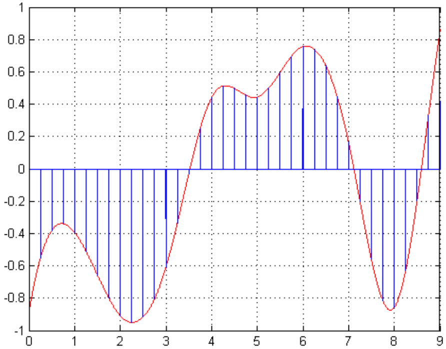


Audio file in time domain



Input representation

Sampling of continous audio



Representation as list of samples

```
[ -18176, -50090, 61573,
  -27710, -55937, 57950,
  -16483, -32160, 49011,
  ...
  -12336, 13795, -59832 ]
```

GTZAN dataset

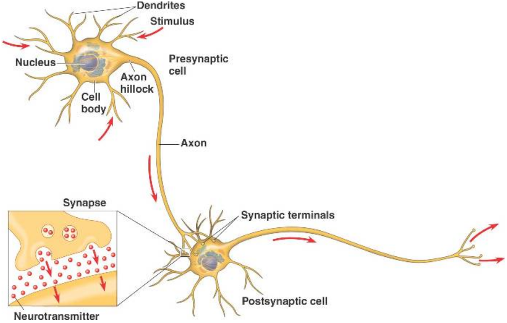
We will be using the GTZAN dataset:

- 10 musical genres
- 100 audio files (mono, samplerate=22050)
- 30s per audio file

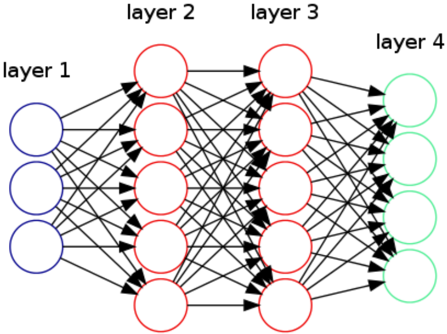
→ 30000s of labeled audio

What is a neural network?

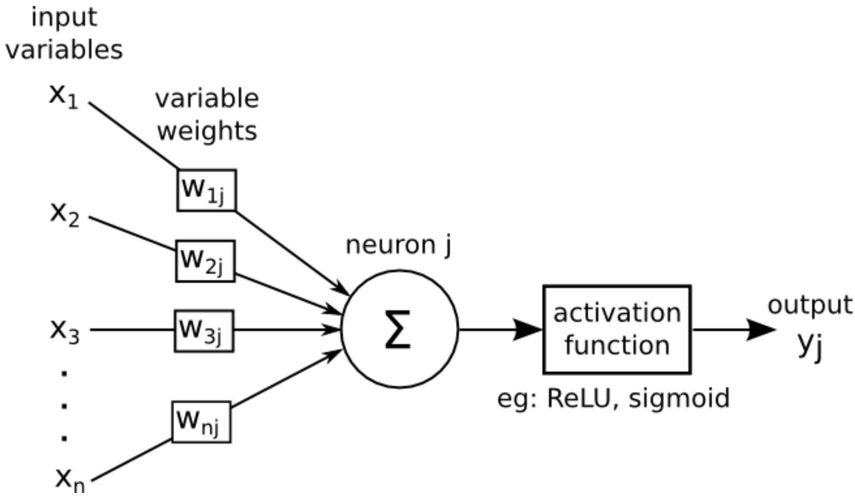
A real neuron



An artificial neural network



Neurons and activation



Training an NN

An output neuron's activation is influenced by:

- The activations of neurons that feed into it
- The weights of incoming connections
- The bias
- The activation function

Given an activation function, how should we change the weights and biases to improve the output activation?
What does it mean to improve?

Backpropagation and Optimization

Optimization and Loss

We...

- will put training data through the neural network
- need to know how bad our output is, therefore:
- define a loss function which judges the output
- will try to minimize this loss function

Backpropagation

- Steps backwards through the network
- If we change a weight or bias by a tiny amount, how will the loss function change?
- Computes in which direction the weights and biases should change to minimize loss

Why we need frameworks

Frameworks are handy for:

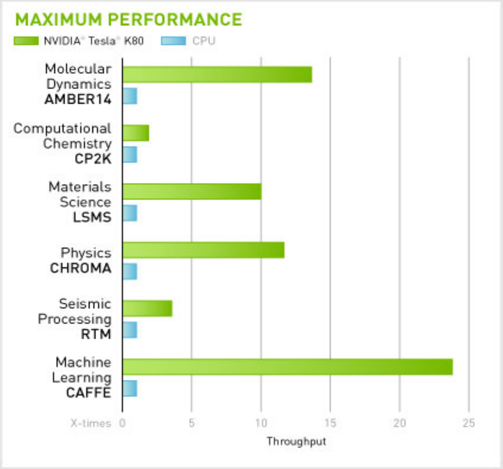
- Abstracting away backpropagation and optimization
- Doing computionally intensive work efficiently
- Parallelization using GPUs
- Minimizing potential error sources

GPU vs CPU for NNs

Speedup through parallelization

- Training NNs can be massively parallelized
- Modern GPUs are basically parallel processing units
- We can achieve huge speedups by using GPUs

Nvidia GPU vs CPU performance



Which frameworks to use?

Tensorflow

- Developed by Google
- Bindings in Python, Java, ...
- General purpose numerical framework
- Powerful, heavily tweakable
- GPU (CUDA) support

PyTorch

- Python first approach
- Easy to get started
- Many examples and tutorials available
- GPU (CUDA) support

Basic structure of an ML script

The basic structure, which we're also using in our example code:

- Data preprocessing and loading
- Model definition
- Running and training
- Cross validation and logging