Neuronové sítě

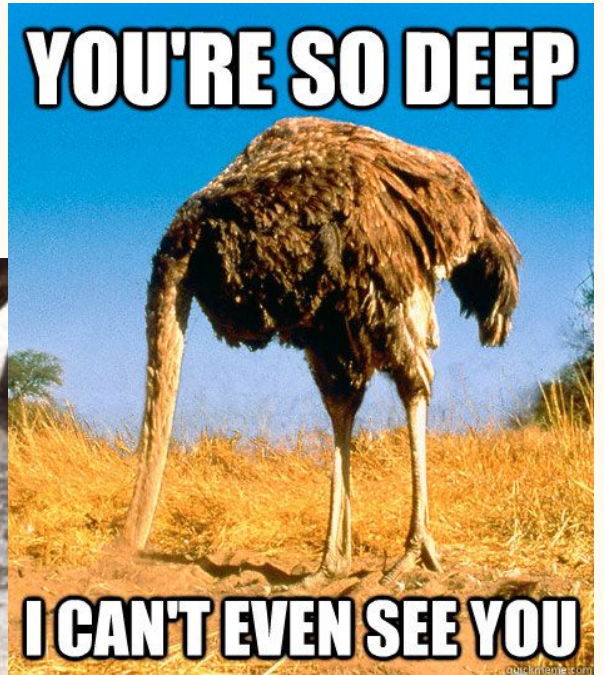
Lukáš Veškrna

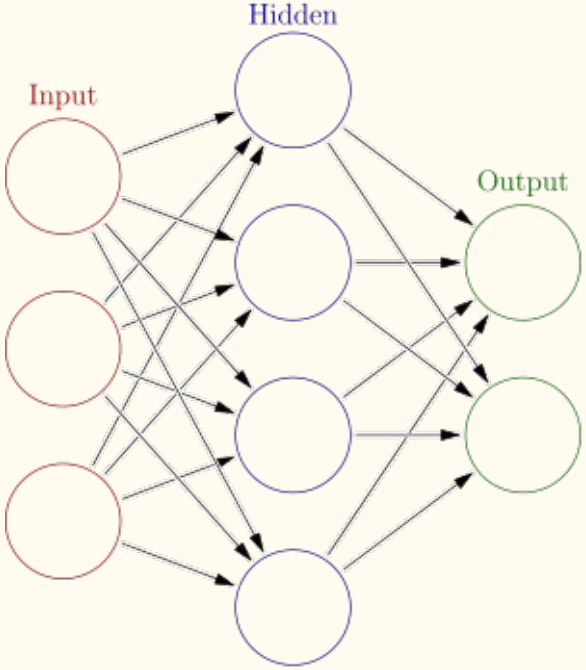
Co jsou neuronové sítě?

Umělá neuronová síť

- Výpočetní model
- Využití v “umělé inteligenci”
- Některé modely jsou velmi jednoduché, jiné složitější
- Deep learning?

Takže co je deep learning?

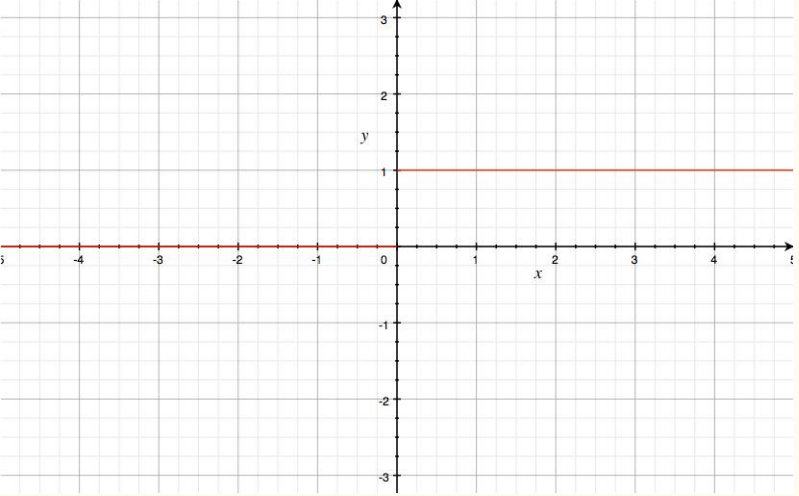




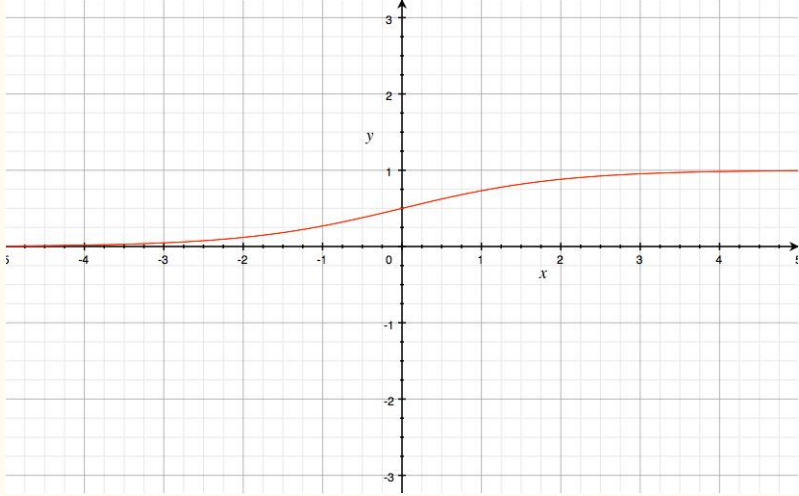
https://upload.wikimedia.org/wikipedia/commons/thumb/4/46/Colored_neural_network.svg/300px-Colored_neural_network.svg.png

Trocha historie

- Není to žádná novinka
- 1943 McCulloch a Pitts
- 1957 Frank Rosenblatt - Perceptron
- Ze začátku se používala skoková přenosová funkce
  - Po nějaké době se ukázalo, že s ní lze řešit pouze lineární problémy

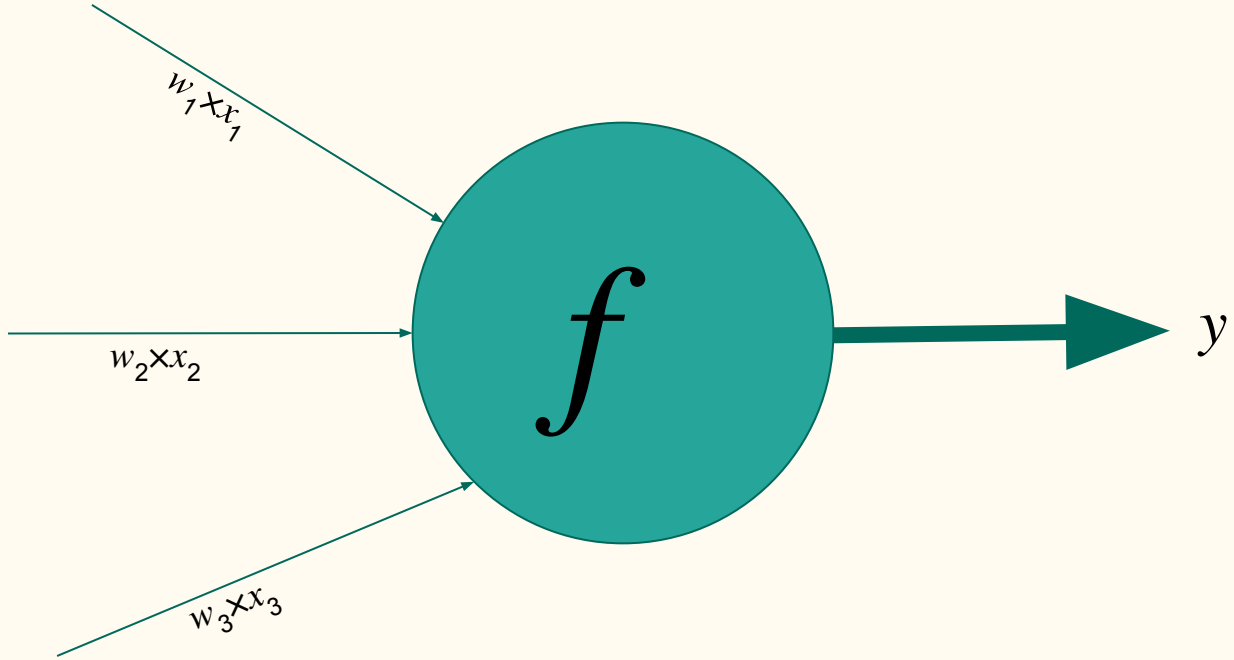


Skoková přenosová funkce



Sigmoida

$$\sigma(x) = \frac{1}{1 + e^{-}x}$$


$w_1 x_1$
$w_2 x_2$
$w_3 x_3$
f
y

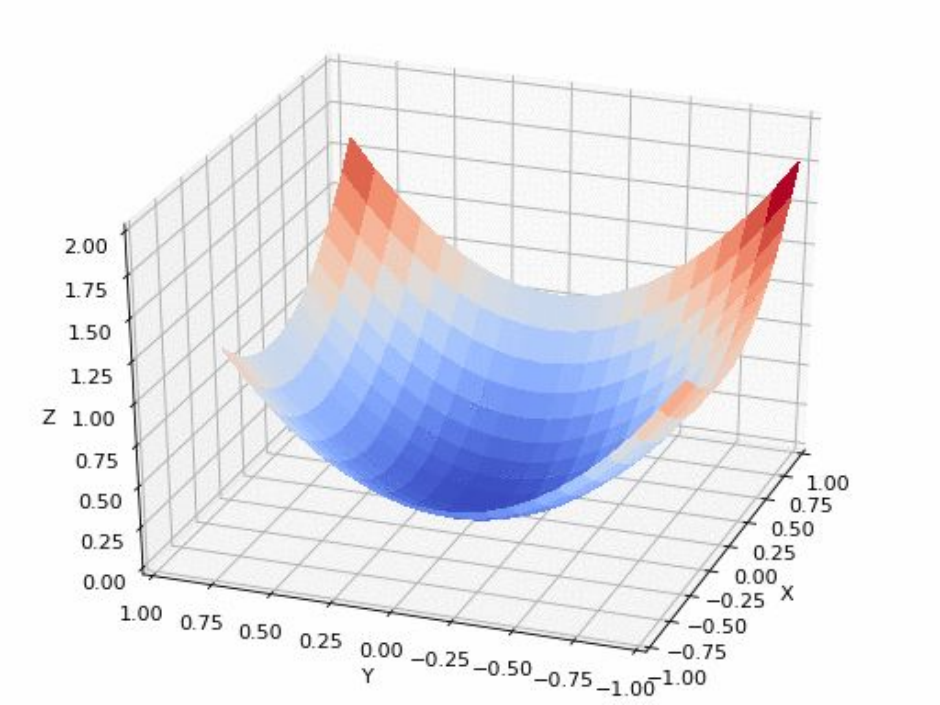
McCulloch-Pitts model

1. x ... vektor vstupů (input vector)
2. w ... vektor vah (weight vector)
3. f ... aktivační funkce
4. Θ ... práh (bias, threshold)
5. y ... výstup neuronu

$$y = f\left(\sum_{i}^{N} w_i x_i + \theta\right)$$

Backpropagation

- Na počátku se neuronová síť sestaví s náhodnými vahami
- Backpropagace (nevím jak je to česky) je algoritmus na úpravu vah v neuronové síťi
- Nejprve vyhodnotíme model na vstupu, ke kterému máme spravný výsledek
- Pak spočítáme odchylku od správného výsledku
- U lineárního modelu:
  - Postupujeme po patrech neuronové sítě od zadu a podle vah spojení rozdělujeme odchylku a upravujeme váhy jednotlivých spojení
- Jenže neuronové sítě nejsou jednoduché a lineární, takže musíme použít složitější algoritmus



https://github.com/pvigier/gradient-descent

Jak závisí chyba na váze synapse

$$\frac{\delta E}{\delta w_{ij}}$$

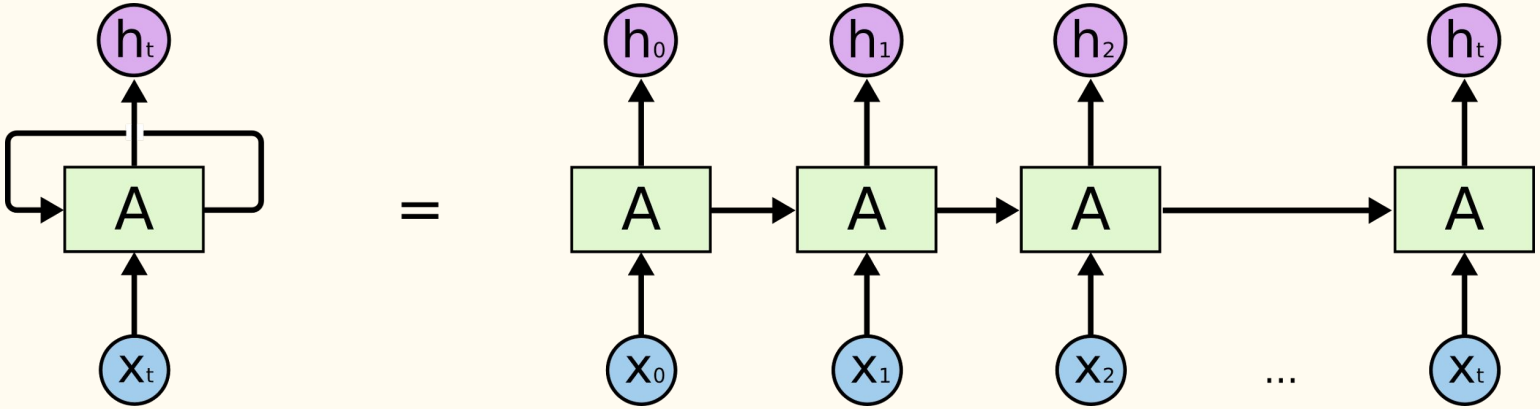
Tohle je ta “lucerna” ke SGD.
Určuje směr svahu.

Další typy neuronových sítí

- Konvoluční neuronové sítě (CNN)
  - Aplikují konvoluční filtry na obrázek a tím dokáží efektivně analyzovat obrázky
- Rekurentní neuronové sítě (RNN)
  - Obsahují cyklické synapse, které jim propůjčují schopnost krátkodobé paměti (LSTM, GRU), dokáží efektivně analyzovat a generovat sekvence
- Autoenkódéry
  - Skládají se z encoderu a decoderu, lze s nimi vyvíjet lepší metody komprese

Můj projekt

Rekurentní neuronové sítě



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

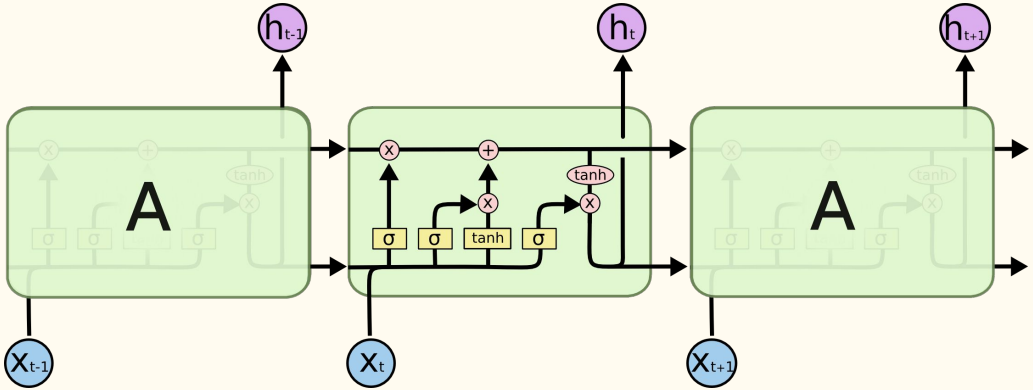
Rekurentní neuronové sítě si pamatují svoje předchozí stavy a ovlivňují podle nich své rozhodnutí.

Generování textů

- Používá rekurentní neuronové sítě ke generování textu (což je sekvence)
- Neuronová síť dostává jako vstup deset písmen a předpovídá příští písmeno
- Používá knihovnu keras
- Algoritmus generování:
  - Vezmeme 10 počátečních písmen
  - Opakujeme, dokud nemáme dost textu:
    - Vygenerujeme příští písmenko
    - Poslední písmeno z deseti umístíme na konec textu
    - Nové písmenko vložíme na začátek těch 10 písmen

LSTM

- Gradient Vanishing Problem
- 1997 - Sepp Hochreiter a Jürgen Schmidhuber (2000 vylepšeno týmem Felixe Gerse)
- Tři brány (gates) na kontrolu vstupů ze svých předchozích stavů a z nového vstupu (cell, input gate, output gate, forget gate)



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Ukázka

“whom I gave very listle company of confused, but from the lineert acforrenced me with gorror; and be pleased with the tortures of her protectors to gave most with our imagination was about these were of a prosecting the biter, whilst the hasden. But to me the same candour, page for me. These vision for my brother. Toon after the structure of scene, it was endeavoured to recure to his wille not rank up the idea that his discourse will be ever which he jneeed the motntains, I felt befection. My life had hitherto attended her destruction breati and darkness were the almost frantic hands which beeace the despair in Gondon, wour son, your kins creatures and that nefelches from a visited my hopes, we passed his scarce, if b country,man overhund me; and then with the food of man and was stretched on a wice field for wonder that the procuces ny profitinn of the most having been the andient in his tnrrow a kear which I had been heard continued. Mf, therefore, I subscribed t”

train.py

```python
from keras.models import Sequential
from keras.models import load_model
#from keras.layers.embeddings import Embedding
#from keras.layers.wrappers import TimeDistributed
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Activation
from keras.layers import LSTM
from keras.layers import Lambda
from keras.callbacks import Callback, ModelCheckpoint
from keras.utils import np_utils
import numpy as np
import os
from django.apps import apps
config = apps.get_app_config("text_generate")
print("Successfully loaded modules.")

if config.SEED:
    assert len(config.SEED) == config.SEQUENCE_LEN


with open(config.CORPUS_FILENAME, "rb") as f:
    text = f.read().decode()
vocabulary = [""] + list(sorted(set(text)))
vocab_len = len(vocabulary)
char2int = {c: n for n, c in enumerate(vocabulary)}
int2char = {n: c for n, c in enumerate(vocabulary)}

def preprocess(text):
    data = []
    labels = []

    for cnum in range(len(text) - config.SEQUENCE_LEN):
        inp = text[cnum:cnum+config.SEQUENCE_LEN]
        data.append([char2int[char] for char in inp])
        labels.append(char2int[text[cnum+config.SEQUENCE_LEN]])

    data = np.reshape(data, (len(data), config.SEQUENCE_LEN, 1))
    data = data / float(vocab_len)
    data = data[:(len(data) - len(data)%config.BATCH_SIZE)]
    labels = labels[:(len(data) - len(data)%config.BATCH_SIZE)]
    labels = np_utils.to_categorical(labels, num_classes=vocab_len)
    return data, labels


print("Preprocessing text...")
data, labels = preprocess(text)
print("Successfully preprocessed.")


if not config.SEED:
    SEED = data[np.random.randint(0, len(data))]
    SEED = np.reshape(SEED, (config.SEQUENCE_LEN))
    SEED = SEED * float(vocab_len)
    SEED = SEED.astype(int)
else:
    SEED = np.array([char2int[char] for char in config.SEED])


def generate(model, text_len):
    res = []
    last_chars = list(SEED)
    for i in range(text_len-len(SEED)):
        x = np.reshape(last_chars, (1, config.SEQUENCE_LEN, 1))
        x = x.astype(np.float) / float(vocab_len)
        predicted = model.predict(x)[0]
        c = np.random.choice(len(predicted), p=predicted)
        last_chars.append(c)
        last_chars.pop(0)
        res.append(c)
    return "".join([int2char[i] for i in res])
```

Zdroje

- https://en.wikipedia.org/wiki/Artificial_neural_network
- https://medium.com/@ageitgey/machine-learning-is-fun-80ea3ec3c471
- https://cs.wikipedia.org/wiki/Um%C4%9Bl%C3%A1_neuronov%C3%A1_s%C3%AD%C5%A5
- http://neuralnetworksanddeeplearning.com/
- http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Rashid, T. (2016). Make your own neural network: a gentle journey through the mathematics of neural networks, and making your own using the Python computer language. United States: CreateSpace Independent Publishing.


DRONY!!! VŽUUM!!!