Introduction to Deep Learning

Jakob Jelenčič

IJS AILAB E3

17th October 2019

Overview

Deep Learning

- Supervised learning ($Y = f(X)$).
- Unsupervised learning (Autoencoders).
- Deep reinforcement learning (DeepMind's AlphaZero).

Artificial neural network

Neuron's function

- Neurons with state v and output y.
- Synapses with weight w.
- Neuron's function: $v = v_0 + \sum_{k=1}^{m} w_k \cdot y_k$.
- $y = \phi(v)$ (sigmoid, softmax, tanh).

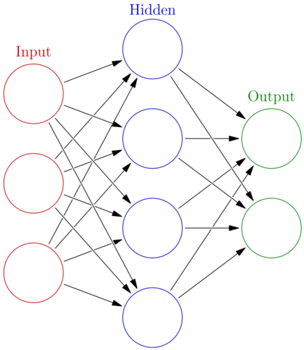


Figure: Number of parameters $= 3 * 4 + 4 + 2 * 4 + 2 = 26$

Loss function

- Arguably the most important choice next to topology.
- Neural nets minimize loss function.
- Usually Mean Squared Error for regression:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

- Usually Cross-entropy for binary classification:

$$H = -(y \cdot log(p) + (1-y) \cdot log(1-p)).$$

Gradient descent

- Let E be MSE loss function: $E = \frac{1}{2}\sum_i (x_i - \hat{x_i})^2$.
- Let y_i^{L+1} be output of a neouron in last layer: $y_i^{L+1} = \phi(v_i^{L+1})$.
- Remember $v_i^{L+1} = \sum_j w_{ji}^{L+1} y_j^L$.

$$\frac{\partial E}{\partial w_{ji}^l} = \frac{\partial E}{\partial y_i^l} \cdot \frac{\partial y_i^l}{\partial v_i^l} \cdot \frac{\partial v_i^l}{\partial w_{ji}^l} = \frac{\partial E}{\partial y_i^l} \cdot \phi'(v_i^l) \cdot y_j^{l-1}.$$

$$\frac{\partial E}{\partial y_i^l} = \sum_k \frac{\partial E}{\partial y_k^{l+1}} \cdot \frac{\partial y_k^{l+1}}{\partial v_k^{l+1}} \cdot \frac{\partial v_k^{l+1}}{\partial y_i^l} = \sum_k \frac{\partial E}{\partial y_k^{l+1}} \cdot \phi'(v_k^{l+1}) \cdot w_{ik}^{l+1}.$$

Optimizer

- Gradient descent algorithms based on chain rule.
- Go to choice: Adam - Stochastic Optimizer based on higher moments.
- Important: Learning rate tuning!

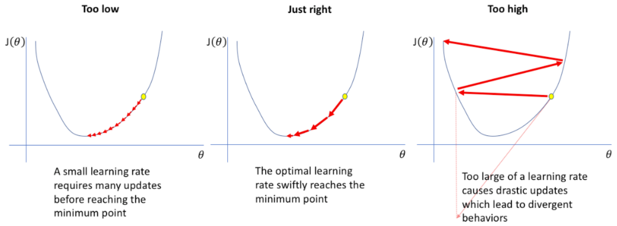

Too low
Just right
Too high
J(θ)
θ
A small learning rate requires many updates before reaching the minimum point
The optimal learning rate swiftly reaches the minimum point
Too large of a learning rate causes drastic updates which lead to divergent behaviors

Figure: https://www.jeremyjordan.me/nn-learning-rate/

- Visual example of 3 parameters space.

Recurrent neural network

- ANN Problem: diminishing gradient when it comes to long sequences.
- Example: $a = 2 + b = 3 \rightarrow c = 5$.
- RNN units has internal memory cells, that allows them to process long sequences.

GRU & LSTM

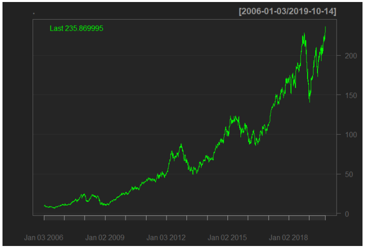
- LSTM & GRU units are special type of cells/neurons designed for capturing long dependencies. Extremely useful when we are working with time series!.
- LSTM has one additional "gates" compared to GRU, which makes it more powerful, yet also more expensive in terms of training.
- GRU has 1 set of parameters, so called hidden state, while LSTM has 2 sets, so called hidden state and cell state. Cell state or memory cell allows LSTM unit to store additional information about input data.
- Naive explanation: GRU "sees" all of the data at once, where LSTM first decide what to show to the second gate. With enough training time LSTM should always outperform GRU, even though GRU tends to converge more quickly.

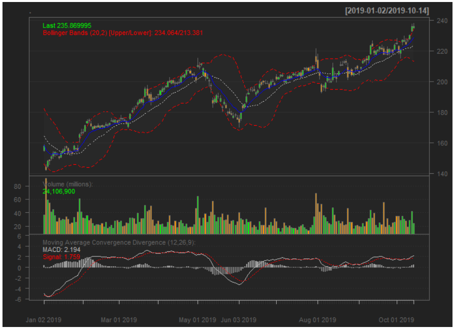
Framework

- Python or R (easiest to learn).
- Keras for high level API (link here).
- TensorFlow for low level backend ops (link here).

Example 1: Keras DNN

- Goal is to predict direction of a trend of Apple stock.
- Simple neural net with 3 dense layers.
- Explaining the "black box" model.
- R script 01 (summary at next slide).

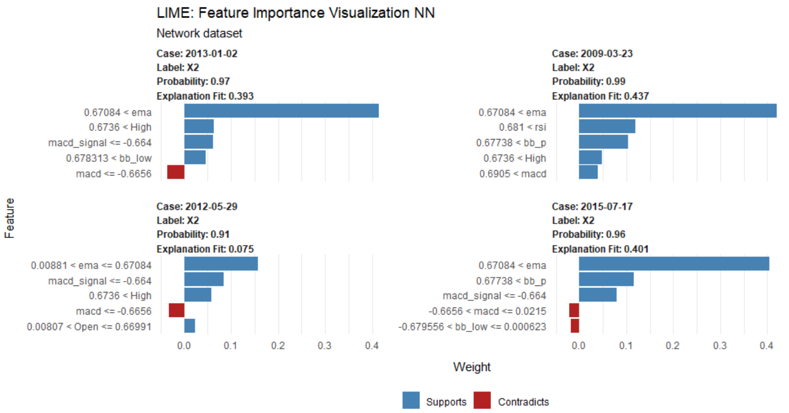




Example 1: Summary

- Normalize data. Otherwise feature with higher scale will probably hold more importance.
- Split data in 3 sets; Train, Test and Validation.
- Stop the training when the model start overfitting.
- Locally explain the model with lime. Keep in mind that here we HAVE to assume local linearity. So two cases that are close to each other should have similar prediction.

Example 1: Summary



Example 2: Keras LSTM

- How to transform data in order to use LSTM.
- Expanding time dimension.
- Harder yet still possible to explain the model (we'll skip it).
- R script 02.

Example 3: Custom TensorFlow learning

- Sometimes Keras API does not allow us enough control and freedom.
- Example: Lets say we work for a broker that values short position α times more than long. Ergo mistakes are now no longer equivalent.
- Solution 1: Optimize Area Under Curve instead of Accuracy and adjust threshold (we'll skip that).
- Solution 2: Optimize Cost function, where False Positive (predicting Increase where it is actually Decrease) is α times more expensive than False Negative.
- R script 03.

Thank you for the attention!

Questions?