Internship Presentation

Daan van Velzen

December 23, 2020

Table of Contents

- Motivation from article Glucocorticoid Receptors[1]
- Single Molecule Microscopy
- Several DNA binding modes with respective diffusion components
- Can Artificial Neural Networks be Used?

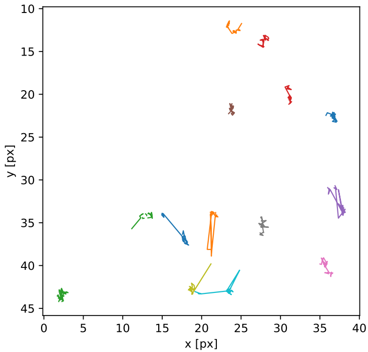


Figure: Single particle tracking with TrackPy[2] of 7326 frames of experimental data returns only 14 trajectories that are longer than 25 frames for 8886 particle localizations.

[1]Keizer et al. (2019)
[2]Allan et al. (2019)

- Classification of diffusion behaviour with ANN's is in use[3]
- Good performance on short trajectories
- Good performance on noisy data
- Assume some trajectories are available with particle tracking

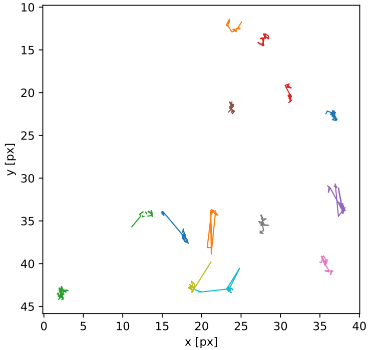


Figure: Single particle tracking with TrackPy[4] of 7326 frames of experimental data returns only 14 trajectories that are longer than 25 frames for 8886 particle localizations.

[3] Granik et al. (2019)

[4] Allan et al. (2019)

Environment

Code for this report is written in Python and makes use of TensorFlow[5] and Keras[6] as machine learning platform.

How to get started

Many great sources available online:
3Blue1Brown: `https://www.3blue1brown.com/`
MIT Deep Learning Course: `http://introtodeeplearning.com/`
Machine Learning Mastery: `https://machinelearningmastery.com/`
Book (Also online):
The Deep Learning Cookbook [7]

[5] Abadi et al. (2016)
[6] Chollet (2015)
[7] Osinga (2018)

Definition (Artificial Neuron)

For some non-linear *activation function* $\phi : \mathbb{R} \to \mathbb{R}$, an *Artificial Neuron* is a function on m inputs x_i of the form

$$a = \phi\left(\sum_{i=1}^{m} w_i x_i + b\right).$$

Here the resulting output a is called the *activation* of the neuron. The w_i are called *weights* and b is called *bias*.

Neural Network

The inputs x_i of a neuron can be either data inputs, or activations from other neurons. By linking neurons together, we make an *Artificial Neural Network*. Neurons are depicted as nodes in network graphs.

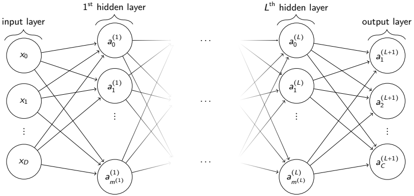


Figure: Network graph of a $(L+1)$-layer dense feed forward neural network with D input units and C output units. The K^{th} hidden layer contains $m^{(K)}$ hidden units.[8]

Example (Feed-forward Neural Network)

Acyclic network. If every input node is the same distance from the output nodes, the architecture of the graph is layered. In fig. 3 the layers are densely connected.

$$a^{(K)} = \phi\left(W^{(k)}a^{(K-1)} + b^{(K)}\right).$$

Trainable parameters

The weights and biases of the individual neurons are the trainable parameters.

[8] Tikz: `https://github.com/davidstutz/latex-resources`

Loss function

In order to learn the model parameters of the ANN, we want to find the parameters that make the model match the output to the labels of a training set as close as possible. The loss function quantifies how well the output of the model approaches the true value.

Example (Loss functions)

Commonly used loss functions are the MSE, MAPE, and cross-entropy.

Overfitting

Training a model on a non-exhaustive sample of a population introduces the risk of overfitting. Overfitting occurs when the performance of a model in-sample continues to improve, but its out of sample performance deteriorates.

Gradient Descent

It is impossible and/or computationally infeasible to find the parameters that find the global minimum of the loss function for the entire sample set. Gradient descent is an iterative procedure by which the parameters are updated based on the local gradient of the loss w.r.t. the parameters.

Backpropagation

Backpropagation is a refinement of gradient descent allowing the loss gradient to be calculated from the last layer forward. This saves memory and makes it computationally feasible to use ANN's as a model.

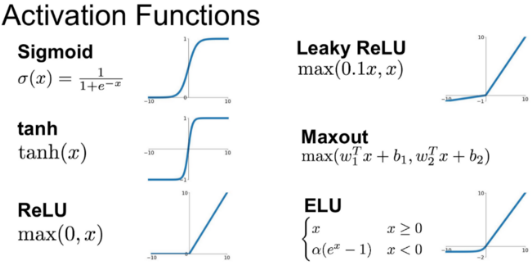


Figure: Overview of commonly used activation functions. From article [9]

Universal Approximation Theorem (refined) [Cybenko, Hornik, Leshno, Pinkus]

A single hidden layer model in which the hidden layer has arbitrary size, can represent any function to arbitrary precision, as long as the activation functions are non-polynomial.

[9] Jadon (2018)

Training

Activation functions must be differentiable a.e. in order for gradient descent to work.

Softmax

The softmax activation function is a differentiable approximation of the arg max function with the sum of the outputs normalized to 1.

$$\mathrm{softmax}_i(\mathrm{x}) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}.$$

Softmax activation is typically used as the last layer of a classification model.

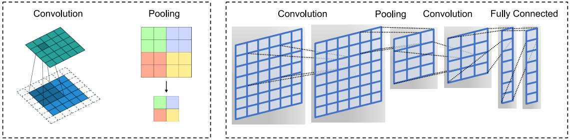


Figure: Graph of convolution as application of a filter to a 2d neighbourhood on the input. Also shown is pooling, or downsampling. From article [10]

Convolutional layers

ANN's do not only consist of densely connected neural layers. Convolutional layers apply a kernel or filter to a neighbourhood of the spatial or temporal data. Subsequent convolutional layers allow for a hierarchy of features to occur, giving rise to "deep learning".

Other layers

Models often include non neuronal layers, such as: Pooling-, Normalizing-, Reshaping- and Regularizing layers.

[10] Maier et al. (2019)

Diffusion Constant

The general expression for the diffusion constant D is

$$\sigma^2 = \langle x^2 \rangle = 2Dt,$$

where x denotes the displacement. We will estimate σ as it is a practical quantity to use in the generation of trajectories.

Data generation

We start out by simulating 20.000 trajectories of 100 steps. The trajectories are generated by generating the the starting point $U \sim \text{Unif}(0, 50)$ and 99 increments $X_i \sim \text{N}(0, 1)$, such that $\sigma X_i \sim \text{N}(0, \sigma^2)$. We then take the cumulative sum along the time dimension.

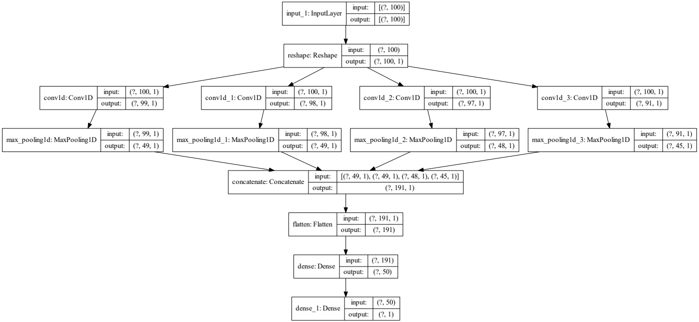
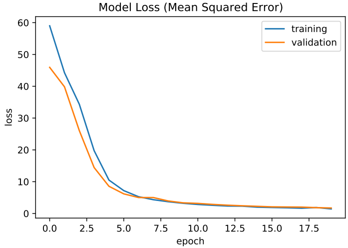


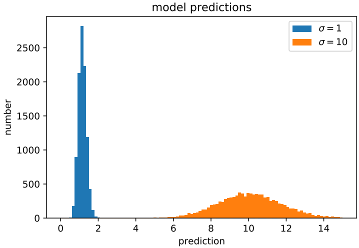
Figure: Architecture of convolutional ANN used to discern between fast and slow diffusion.

First Approach

4 parallell convolutional layers with max pooling and two dense layers.



(a) Learning curve for distinguishing between fast and slow diffusion.



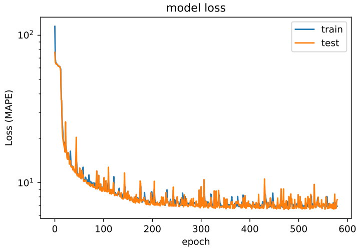
(b) Predicted value of σ for 20.000 randomly generated trajectories with $\sigma = 1$ for 10.000 and $\sigma = 10$ for the other 10.000.

Training

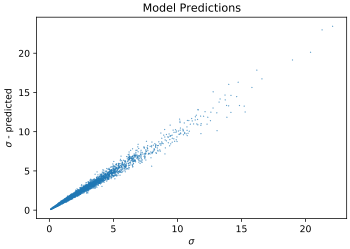
Model with ReLU activations and MSE loss function. 20 epochs on a batch size of 300

Can we estimate σ?

Try generated data instead of fixed dataset. Use the early stopping callback and let training run indefinitely until a plateau is reached. Reuse the model from the previous iteration.



(a) Learning curve for estimation of σ. MAPE loss on log-scale.



(b) Predicted value of σ for 10.000 randomly generated trajectories with $\sigma \sim \text{Weibull}(.7, 1)$.

Performance

The checkpointed most optimal model finally reaches an MAPE of 6,6202 after testing with 10.000 generated trajectories. Meaning that we trained this model to have an expected relative error for σ of around 6,6 percent. We compare this to the performance of $\hat{\sigma} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \langle x \rangle)^2}$, where x denotes displacement. The expected relative error we approximate by bootstrapping. In this manner we find that the MLE estimate for σ slightly outperforms the ANN, as it has a value of $5,8329$ percent after verifying on the same 10.000 generated trajectories.

Alternative problem definition

$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \langle x \rangle)^2 \sim \frac{\sigma^2}{n}\chi^2_{(n-1)}$. This means that $\mathrm{Var}(\hat{\sigma}^2) = \frac{2\sigma^4(n-1)}{n^2}$

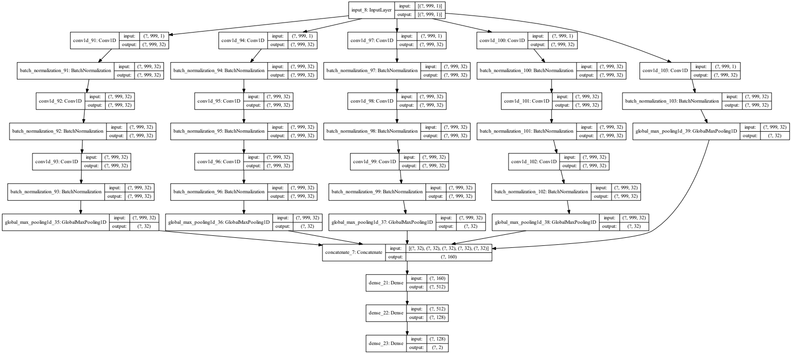
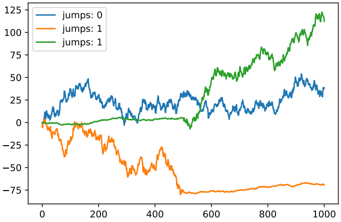
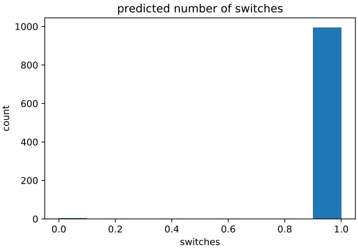


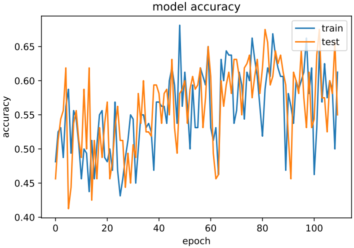
Figure: Classification model as used in the article by Granik et al. (2019). Three layers of convolution and regularization layer and maxpooling, with one shortcut. two hidden dense layers.



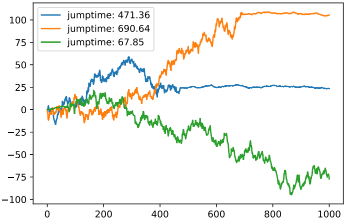
(a) Generated trajectory with switch at fixed position



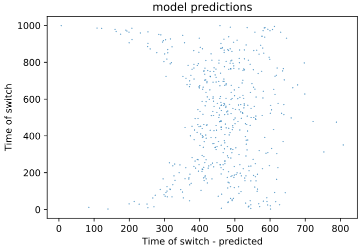
(b) Predicting whether trajectory switched for 1000 trajectories., accuracy = 0.5035



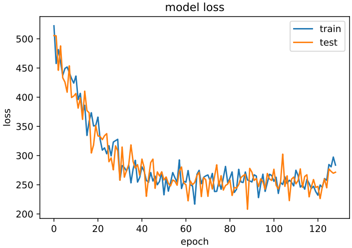
(c) Learning curve for assessing switch or not



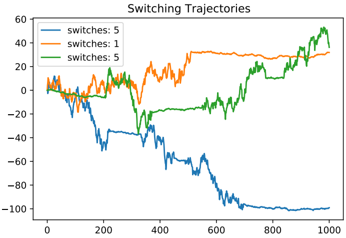
(a) Generated trajectory with one random switch



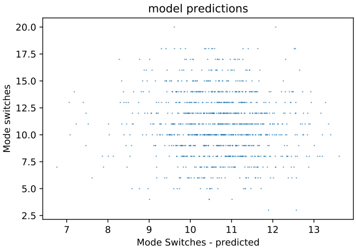
(b) Prediction of time of switch



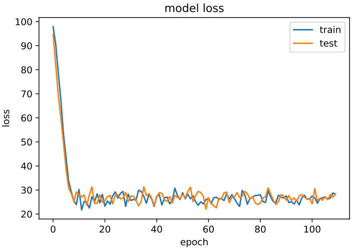
(c) Learning curve for estimation of switch time. Mean average error = 260.2427



(a) Generated trajectory with varying number of switches



(b) Predicted number of switches for 1000 generated trajectories



(c) Learning curve for estimation of number of switches.

- ANN's do not outperform classical methods at this moment.
- Convolutional methods do not seem able to distinguish non-local differences.
- Linear combinations of independent variables with mean 0 have mean 0
- Large number of options makes purposeful improvement difficult
- Methods show great potential, and attaining goal is probably possible.

References I

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*, available at 1603.04467.

Allan, Dan, Casper van der Wel, Nathan Keim, Thomas A Caswell, Devin Wieker, Ruben Verweij, Chaz Reid, Thierry, Lars Grueter, Kieran Ramos, Apiszcz, Zoeith, Rebecca W Perry, François Boulogne, Prashant Sinha, Pfigliozzi, Nicolas Bruot, Leonardo Uieda, Jan Katins, Hadrien Mary, and Aron Ahmadia. 2019. *soft-matter/trackpy: Trackpy v0.4.2*.

Chollet, François. 2015. *Keras*.

Granik, Naor, Lucien E. Weiss, E. Nehme, Maayan Levin, Michael Chein, Eran Perlson, Yael Roichman, and Yoav Shechtman. 2019. *Single-Particle Diffusion Characterization by Deep Learning*, Biophysical Journal **117**, no. 2, 185–192.

Jadon, Shruti. 2018. *Introduction to different activation functions for deep learning*, Medium, Augmenting Humanity **16**.

Keizer, Veer I.P., Stefano Coppola, Adriaan B. Houtsmuller, Bart Geverts, Martin E. Van Royen, Thomas Schmidt, and Marcel J.M. Schaaf. 2019. *Repetitive switching between DNA-binding modes enables target finding by the glucocorticoid receptor*, Journal of Cell Science **132**, no. 5.

References II

Maier, Andreas, Christopher Syben, Tobias Lasser, and Christian Riess. 2019. *A gentle introduction to deep learning in medical image processing*, Zeitschrift für Medizinische Physik **29**, no. 2, 86–101.

Osinga, Douwe. 2018. *Deep Learning Cookbook*, 1st ed. (Rachel Roumeliotis and Jeff Bleiel, eds.), O'Reilly Media, Sebastopol, CA.