Głębokie uczenie
Wykład 6

- Bayesowskie sieci neuronowe
- Mixture Density Networks MDN

Marcin Wolter

IFJ PAN

13 stycznia 2021

Uczenie maszynowe vs byesowskie

Uczenie maszynowe

Ucząc funkcji $y = f(x)$ na podstawie **danych treningowych** $\boldsymbol{T} = (\boldsymbol{x}, \boldsymbol{y}) = (x,y)_1, (x,y)_2, \ldots (x,y)_N$ i **więzów** dostajemy jedną estymowaną funkcję $f(x)$

Uczenie bayesowskie

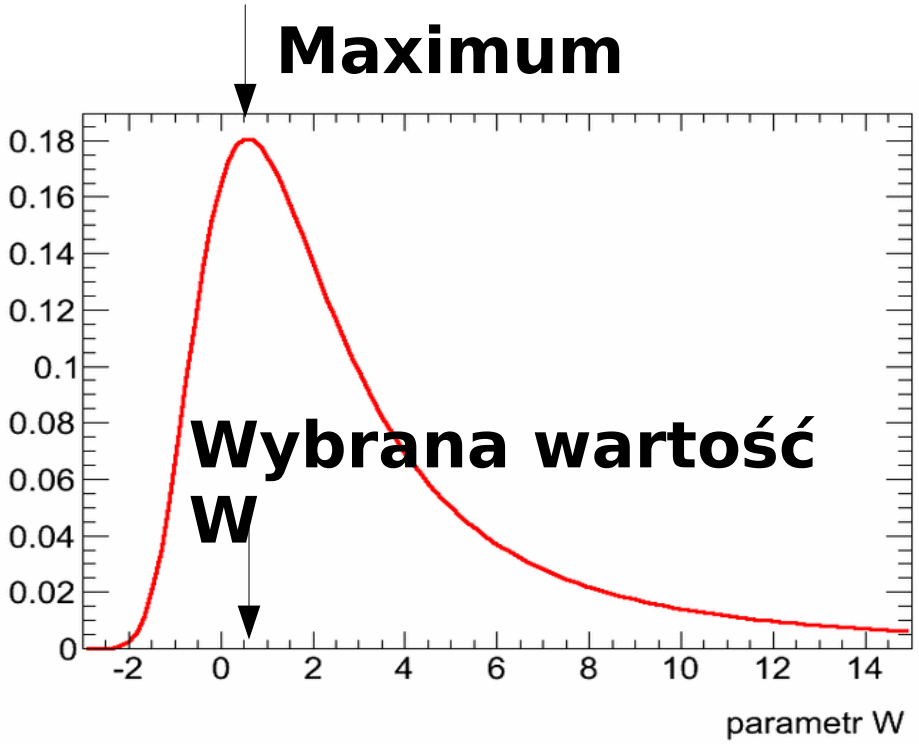
Dla każdej funkcji f(x) z przestrzeni F znajdujemy prawdopodobieństwo *a posteriori* $\boldsymbol{p(f \mid T)}$ na podstawie danych $\mathbf{T= (x, y).}$

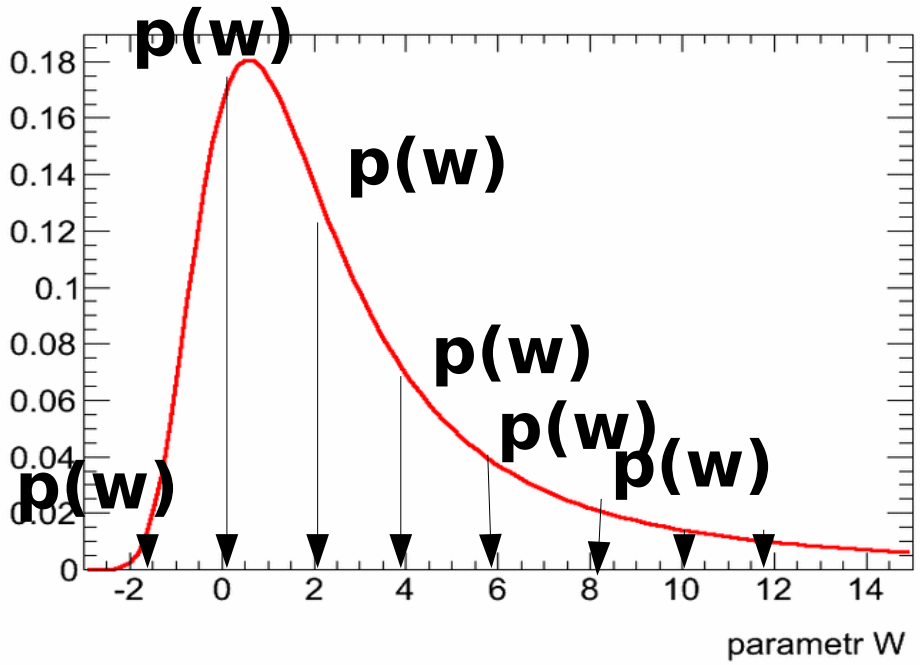
W uczeniu bayesowskim **NIE DOSTAJEMY** jednej, najlepszej funkcji aproksymującej, ale wiele funkcji ważonych ich prawdopodobieństwem.

Prawdopodobieństwo *a posteriori* – prawdopodobieństwo obliczone z użyciem wyników eksperymentu.

Zbiór treningowy $\boldsymbol{T} = (\boldsymbol{x}, \boldsymbol{y})$: zbiór wektorów wejściowych $\boldsymbol{x}$ i wynikowych $\boldsymbol{y}$.

Uczenie maszynowe i bayesowskie





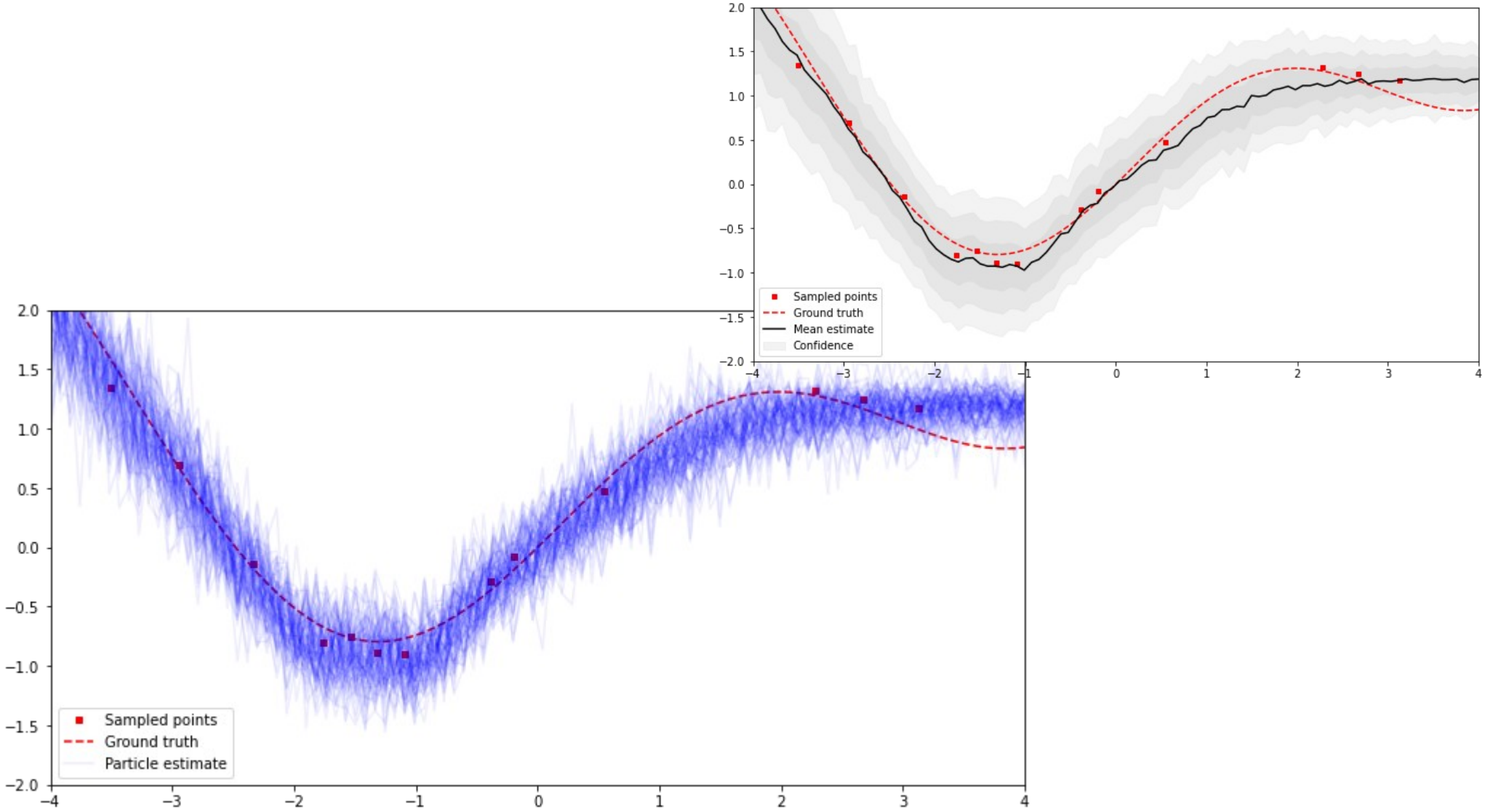
Uczenie maszynowe

Wybieramy jedną funkcję (lub wartość paratru/ów opisującego funkcję).

Uczenie bayesowskie

Każda funkcja (lub wartość parametru) posiada pewne prawdopodobieństwo (wagę).

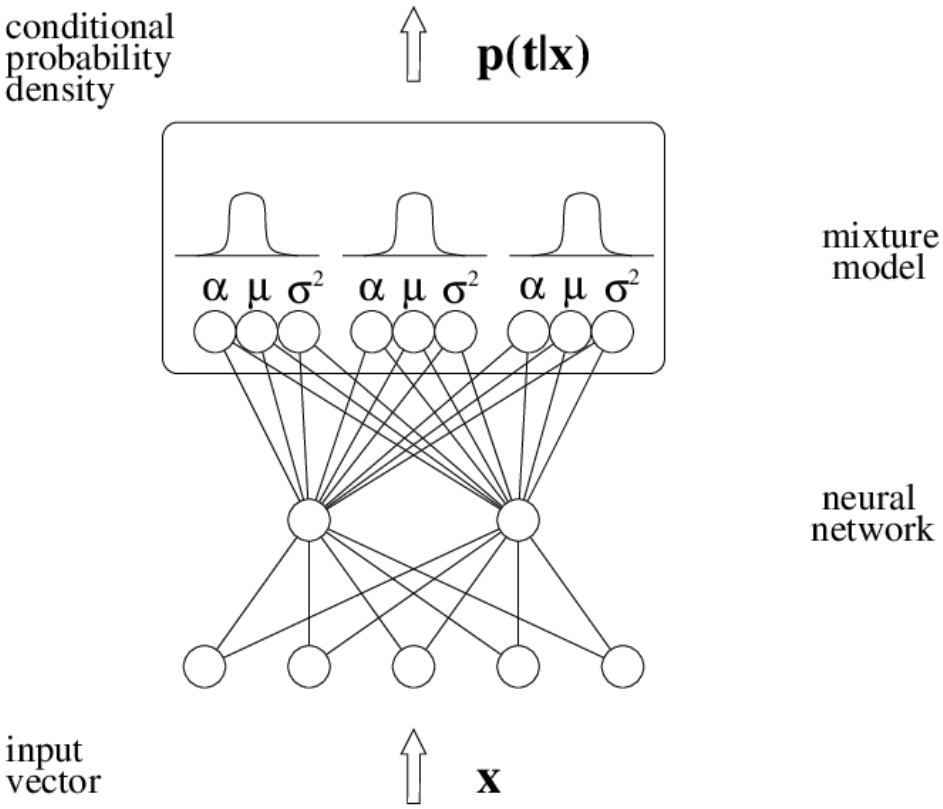
Przykład bayesowskiej sieci neuronowej



- https://github.com/marcinwolter/DeepLearning_2020/blob/main/bnn_regression.ipynb

Mixture Density Network MDN

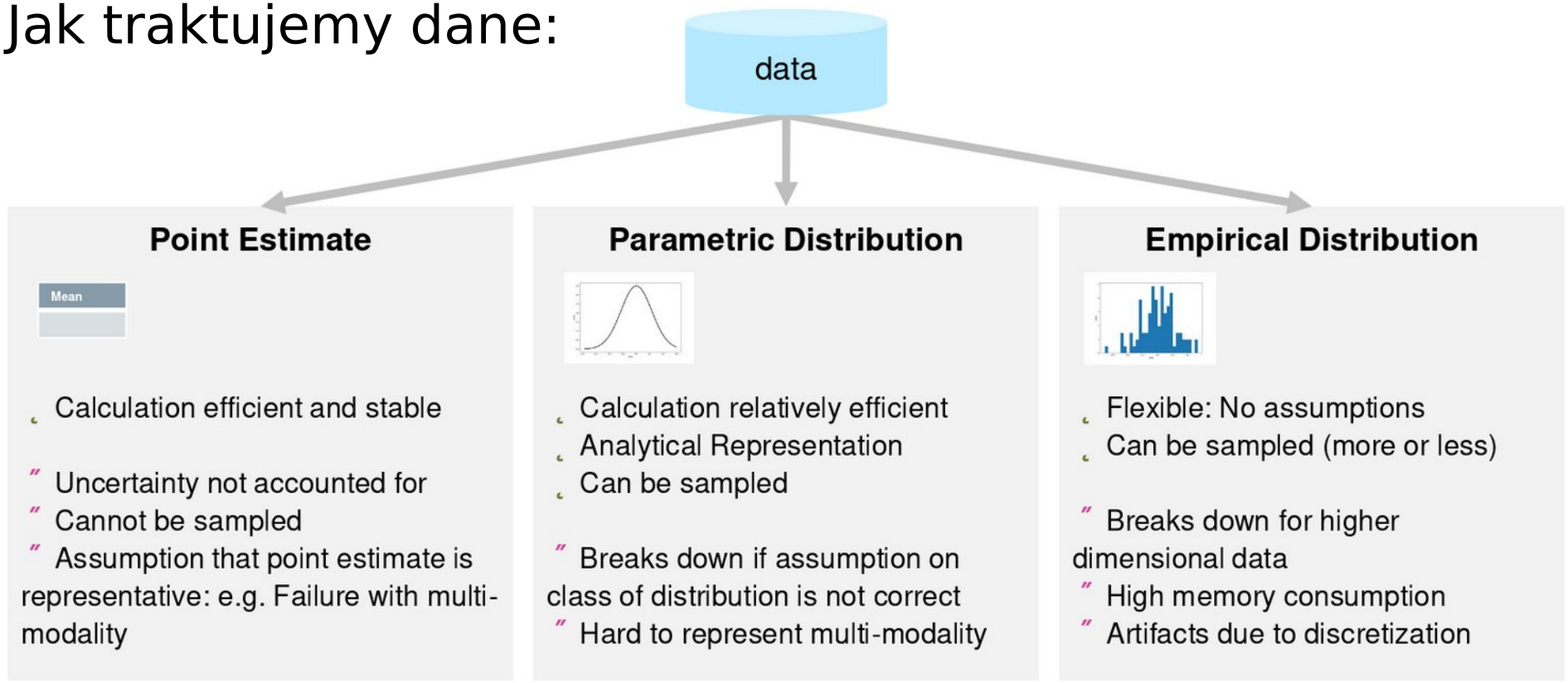
- Musieliśmy trenować bardzo wiele… Zabiera to dużo czasu.
- Może dałoby się to zrobić bardziej wydajnie?
- Idea:
  - Stwórzmy sieć, która zwraca sparametryzowane rozkłady prawdopodobieństwa



Nazywa się Mixture Density Network (MDN).
Rozkład prawdopodobieństwa jest sparametryzowany przez kilka rozkładów Gaussa.

Mixture Density Network (MDN)

Jak traktujemy dane:



Classic Neural Network

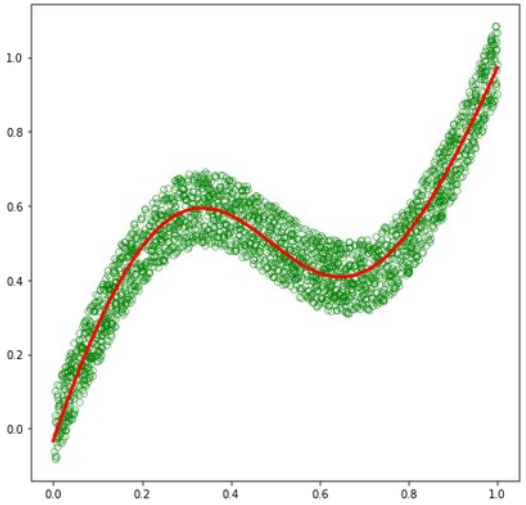
Mixture Density Neural Network

Bayesian Networks

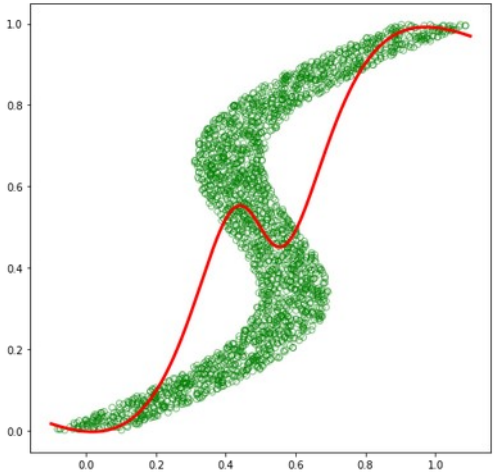
Mixture Density Network (MDN)

- MDN – przydatny formalizm aproksymujący rozkład prawdopodobieństwa.
- Rozkład prawdopodobieństwa warunkowego *a posteriori* $p(y|x)$ jest aproksymowany jako suma kilku rozkładów Gaussa, gdzie rozkłady te są sparametryzowane przez parametry będące funkcją danych wejściowych x.

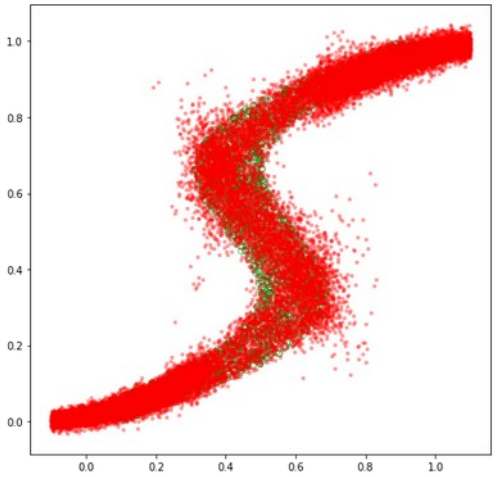
Regression:



Regular network - OK



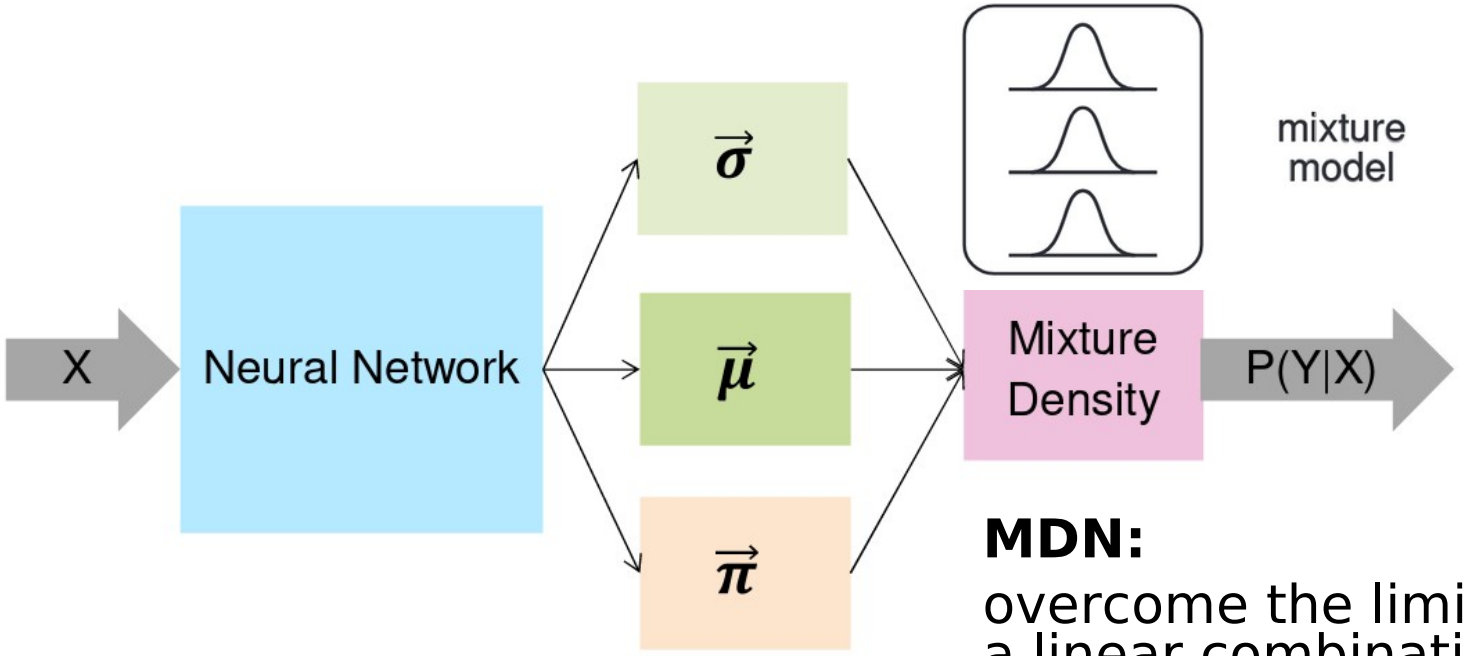
Regular network - problems



MDN sampling - OK

Ciekawa prezentacja:
http://www.dbs.ifi.lmu.de/Lehre/DLAI/WS18-19/script/06_uncertain.pdf

Mixture Density Network (MDN)



$\vec{\sigma}$: standard deviations
$\vec{\mu}$: means
$\vec{\pi}$: cluster weights

Loss: $-\log P(\mathbf{Y}|\mathbf{X})$

MDN:
overcome the limits by using a linear combination of kernel function

MDN zwraca nie tylko maksimum rozkładu prawdopodobieństwa (jak zwyczajna sieć neuronowa), ale całe rozkłady prawdopodobieństwa (czyli też błędy).

Reference: *Bishop, Christopher M. Mixture density networks. Technical Report NCRG/4288,Aston University, Birmingham, UK, 1994*
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.120.5685&rep=rep1&type=pdf

Działający przykład MDN

Prykład z:
https://github.com/cpmpercussion/keras-mdn-layer
Na dole strony jest instrukcja, jak używać MDN (bardzo prosto).

Przykład dla Google Colaboratory:
https://github.com/marcinwolter/DeepLearning_2020/blob/main/MDN_1D_sine_prediction.ipynb

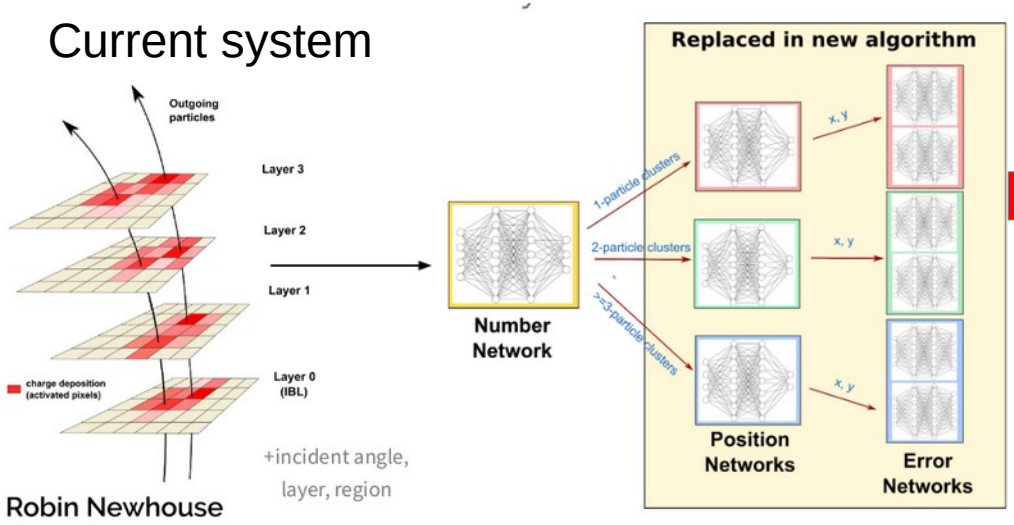
Przykład MDM
- ATLAS cluster position calculation

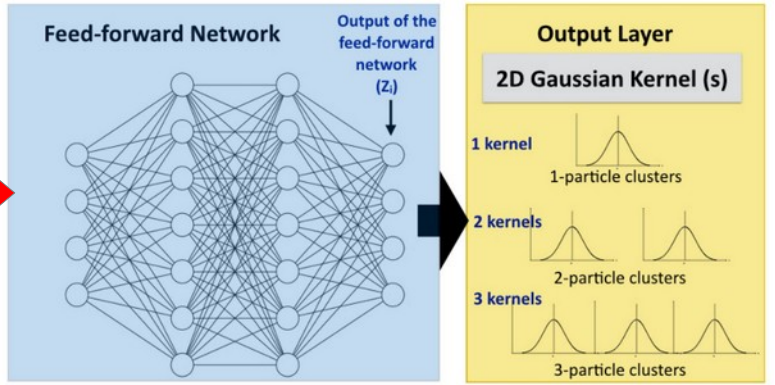
Currently uses a three-step process

1. Number Network: Determine the **number of particles** in this hit
2. Position Network: Determine the **x-y position** of each particle
3. Error Network: Estimate the **error in x and y** for each particle

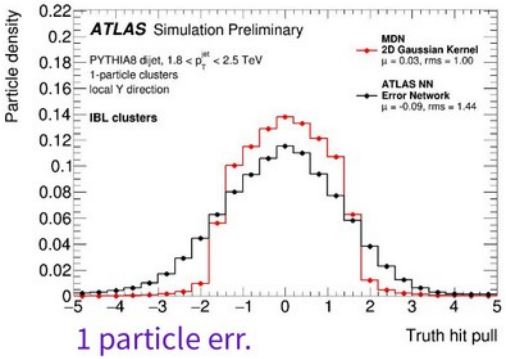
MDN: Feedforward network + Gaussian kernel(s)

Current system

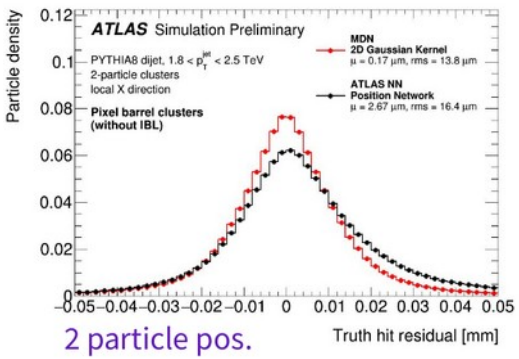




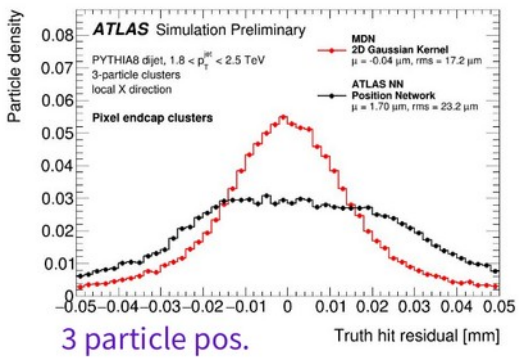
ATLAS studies of Elham E Khoda:
https://indico.cern.ch/event/795039/contributions/3432383/attachments/1856640/3049881/Machine_Learning_in_CTIDE_1.pdf



1 particle err.



2 particle pos.

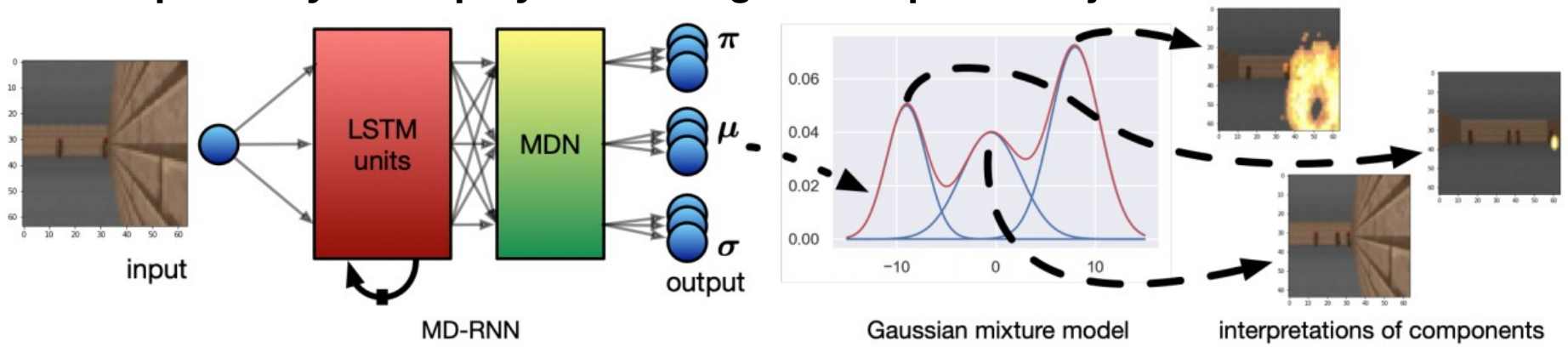


3 particle pos.

Przykład zastosowania

How do Mixture Density RNNs Predict the Future? Kai O. Ellefsen, Charles P. Martin,Jim Torresen
https://arxiv.org/pdf/1901.07859.pdf

Problem: przewidywanie przyszłości w grze komputerowej.



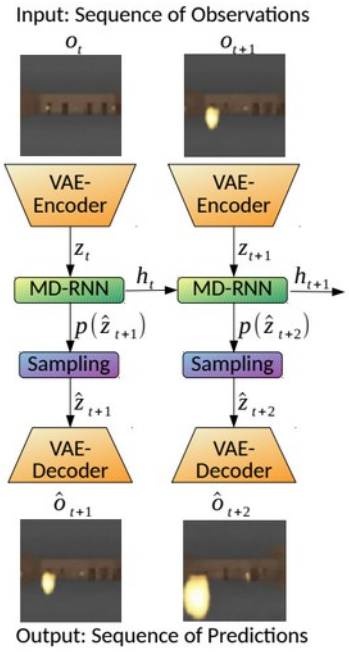


Figure 2. World model predicting future frames by combining a variational autoencoder and an MD-RNN. We follow the architecture suggested in (Ha & Schmidhuber, 2018).

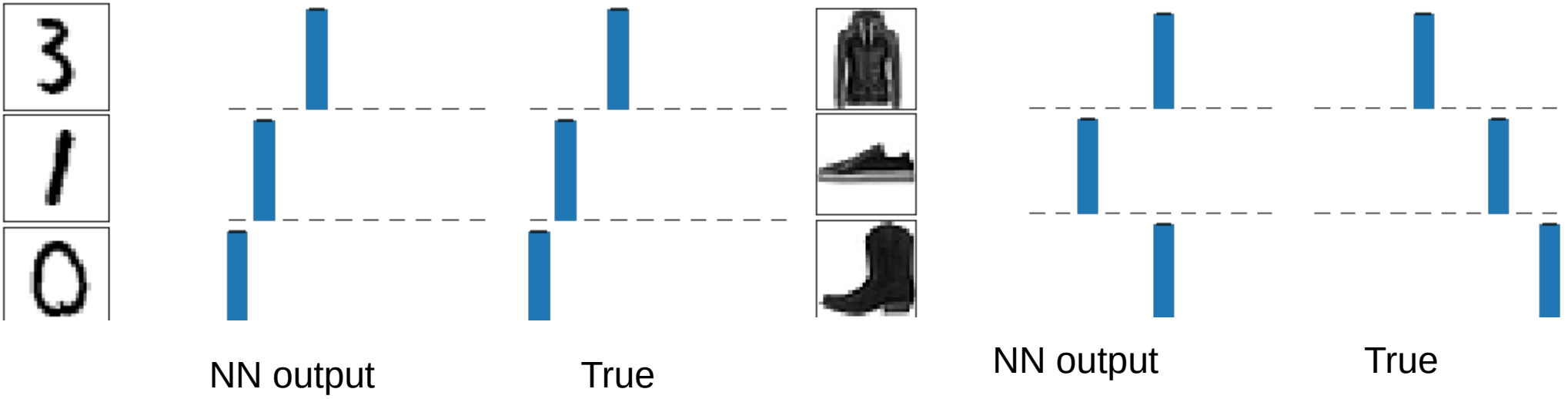
Model zwraca serię przewidywań bazujących na poprzednich klatkach. Do każdego przewidywania podaje też błąd.

Przykład: klasyfikacja ręcznie pisanych cyfr MNIST

- Szkielet programu:

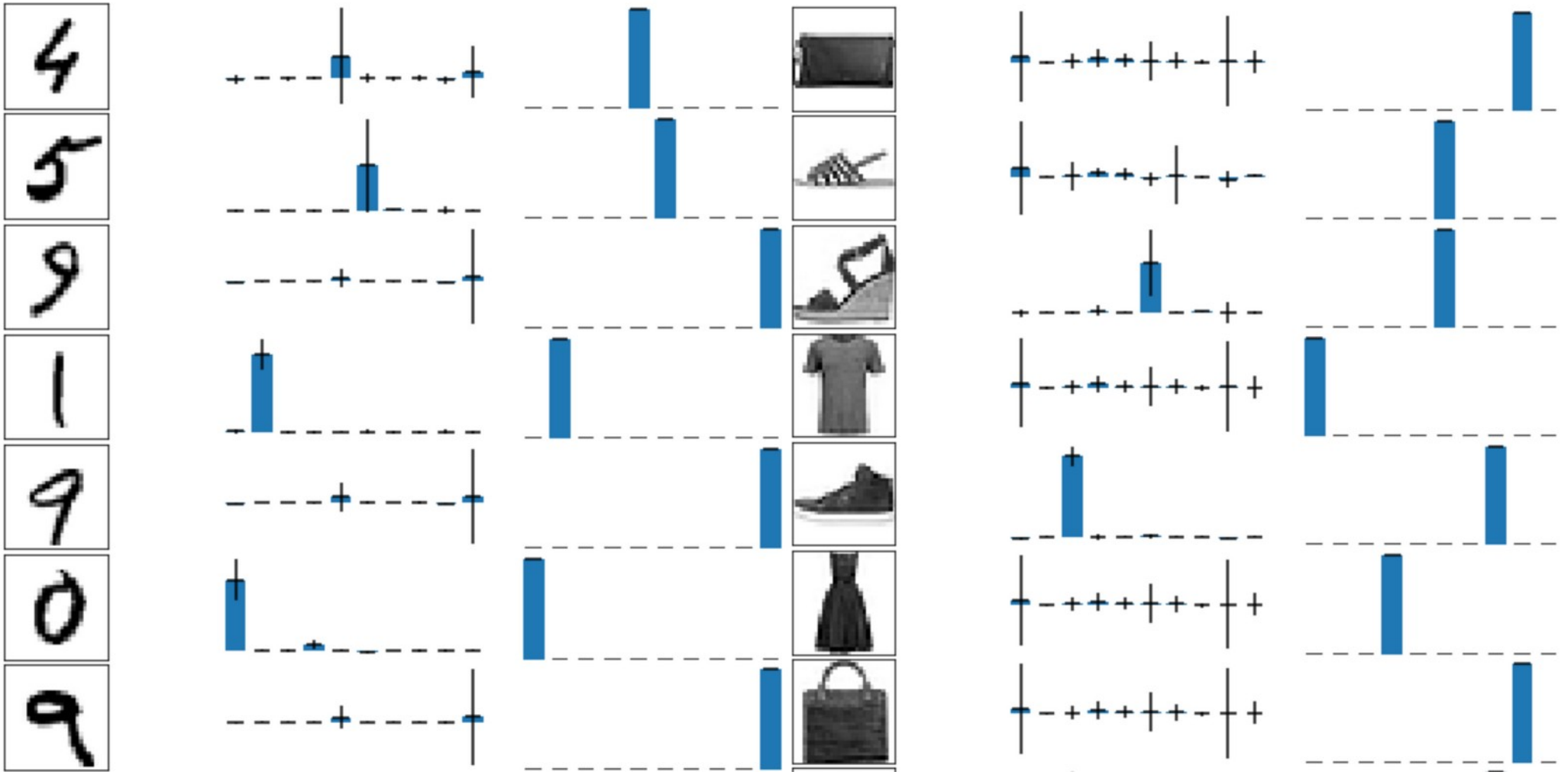
  https://github.com/marcinwolter/DeepLearning_2020/blob/main/simple_mnist_mdn_skeleton.ipynb

- Trenujemy sieć na cyfrach, ale potem testujemy na MNIST fashion:



Nic nie wiemy o błędzie klasyfikatora!

Używamy MDN



Mixed Density Networks MDN

- Dość leciwy pomysł (lata 90-te)
- Ale bardzo przydatny: zamiast jednej wartości dostajemy rozkład prawdopodobieństwa!

Choi, Sungjoon, et al. "Uncertainty-aware learning from demonstration using mixture density networks with sampling-free variance modeling." 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018.
https://arxiv.org/pdf/1709.02249.pdf

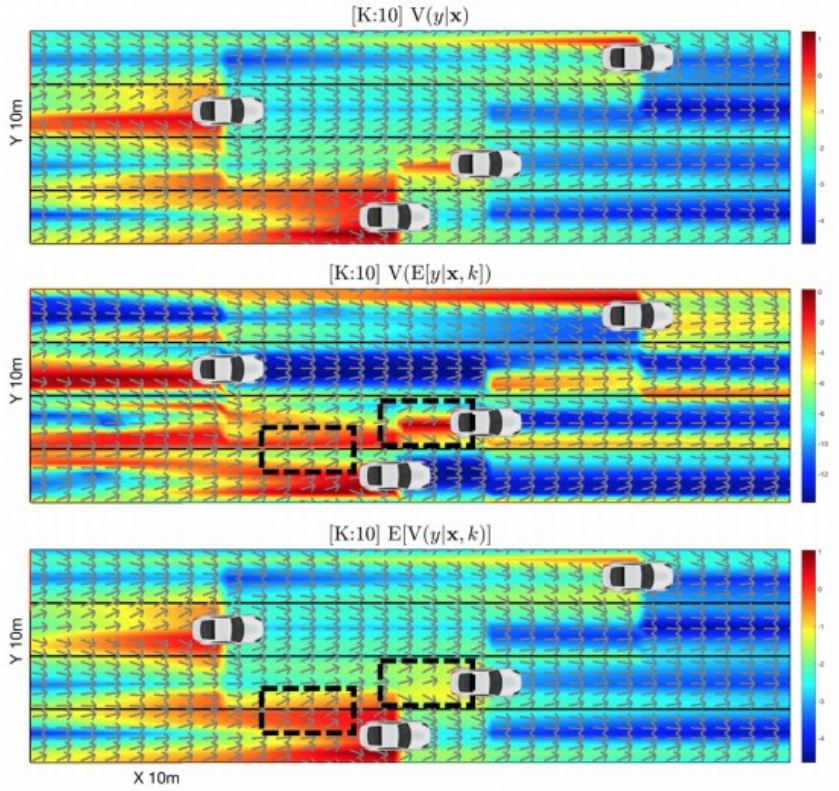
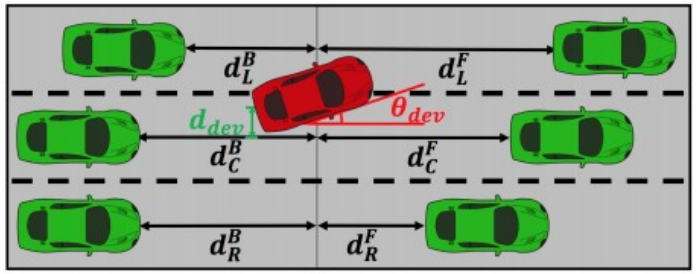
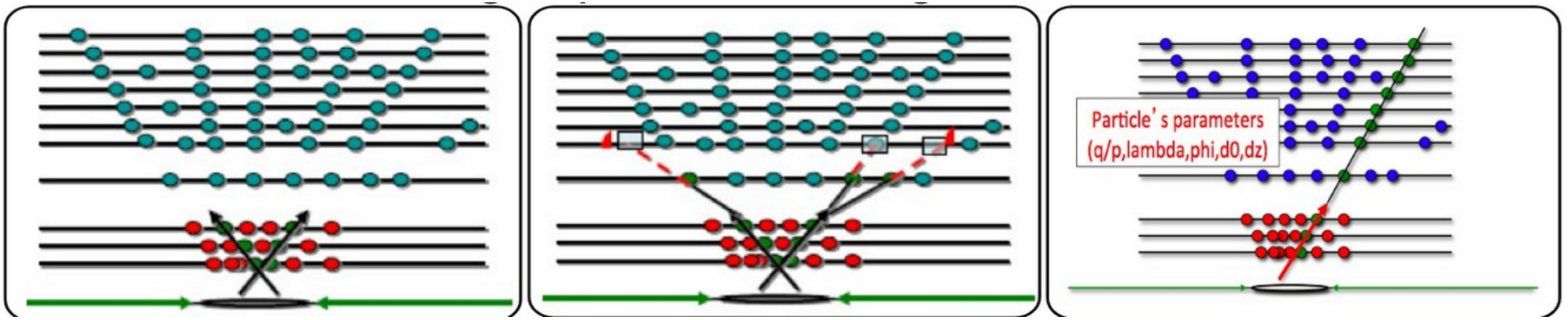


Fig. 5: Different uncertainty measures on tracks.



More scientific - Pattern Recognition in High Energy Physics

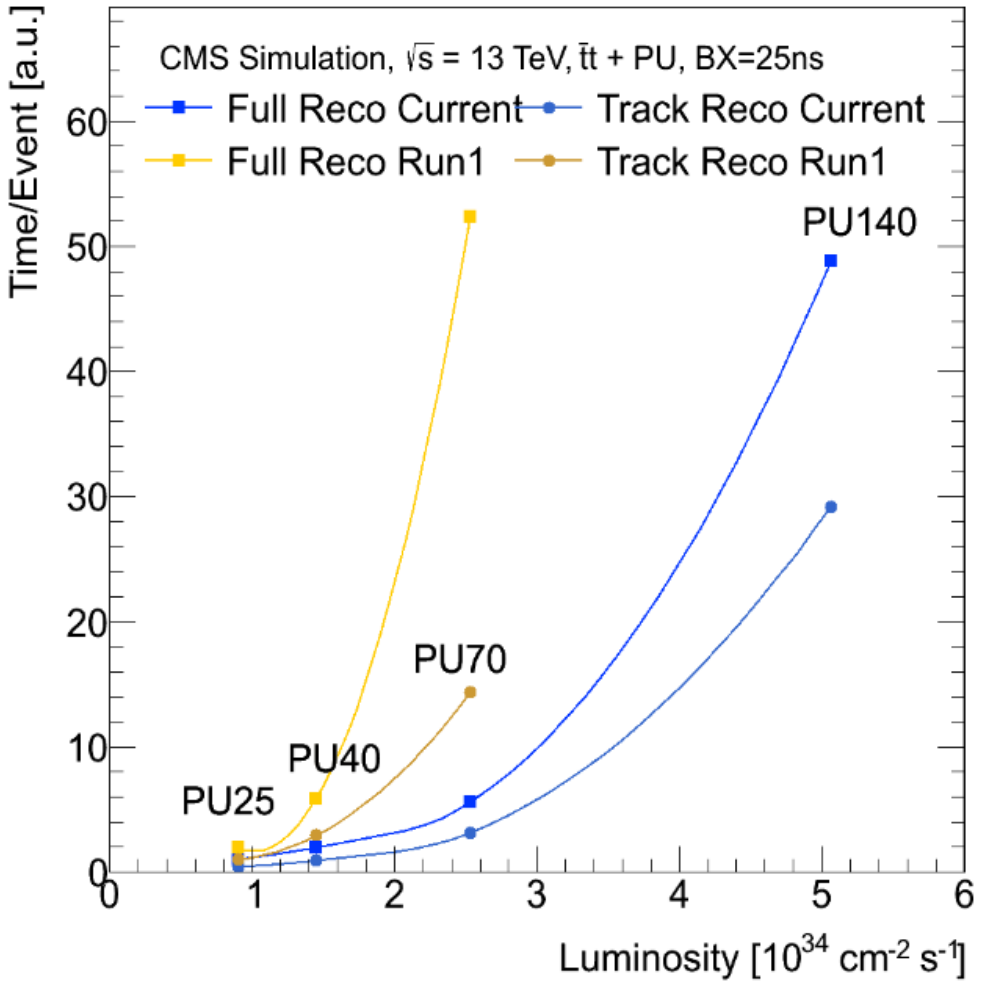


Seeding **Track Building** **Track Fitting**

- **Track seeding** – finding the seeds (initial sets of hits) from which the track starts
- **Track building** ≡ pattern recognition HEP jargon
  - Creating a 2- and 3-dimensional lines and assigning to them all the hits within a certain window
  - Fitted frequently with “robust fit”
- **Track fitting** – final fitting of the track parameters (usually a Kalman filter used for tracking)

Usually this method works fine, is robust and efficient!

So, where is the problem?



CMS experiment simulation

J.-R. Vlimant, Machine Learning for Charged Particle Tracking, MIT, 2018

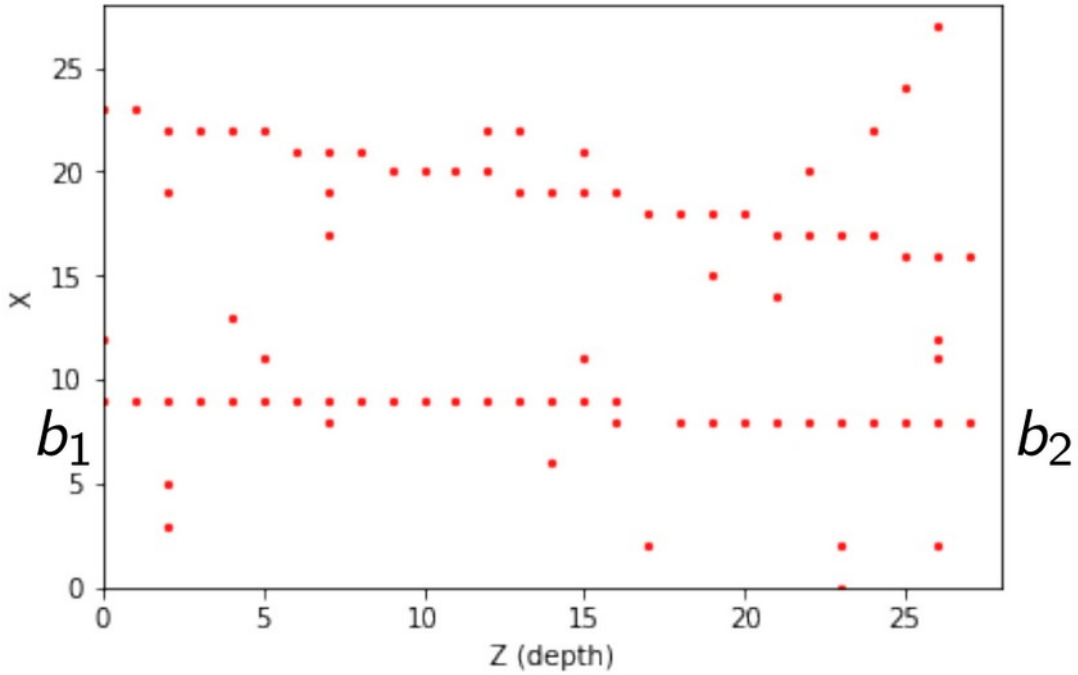
- The time needed to process one event grows quickly (worse than quadratic) with luminosity (number of collisions).
- Huge part of CPU consumption is the track finding.

Deep Neural Network (DNN)?

- Fast, parallel, in principle does pattern recognition “at once”, without looping over hits.
- Also experiments with lower occupancy might profit from DNN’s – higher precision and efficiency.
- There is a HEPTrkx group working on tracking for HEP experiments: *https://heptrkx.github.io/*

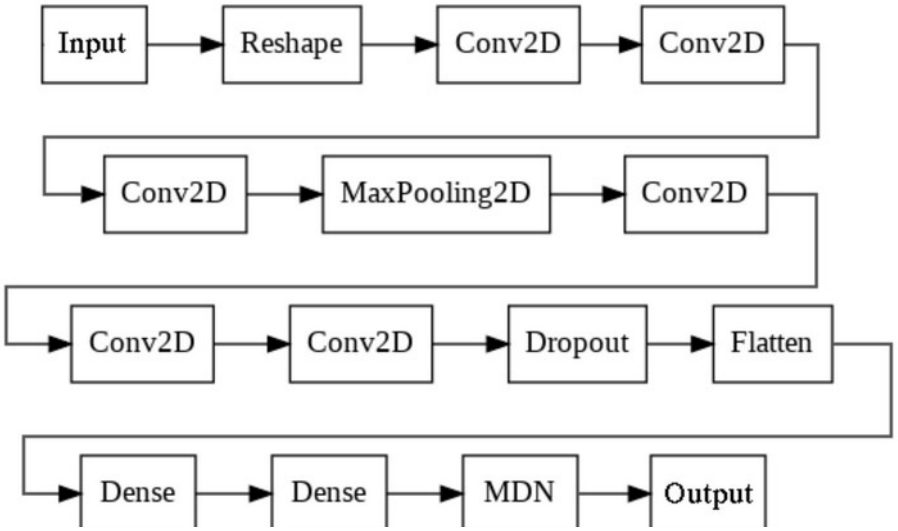
Tracking in 2D toy model

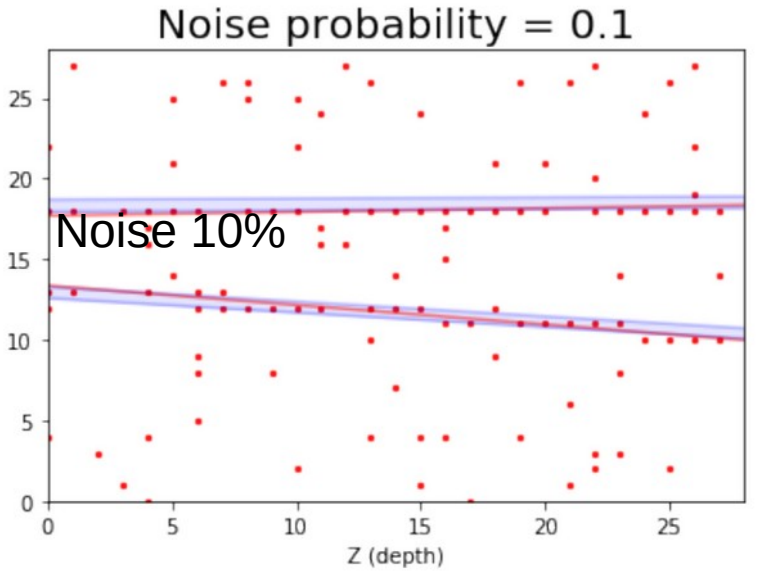
- **CNN returns track parameters (regression)**
- What about many tracks?
  - Solution: add **Mixture Density Network(MDN)** layer to process many tracks.
  - Straight tracks – described by two parameters. Each parameter has associated MDN Gaussian
  - If a number of tracks lower than expected some MDN outputs have very low amplitude.
  - **Important -** we are getting errors of track parameters

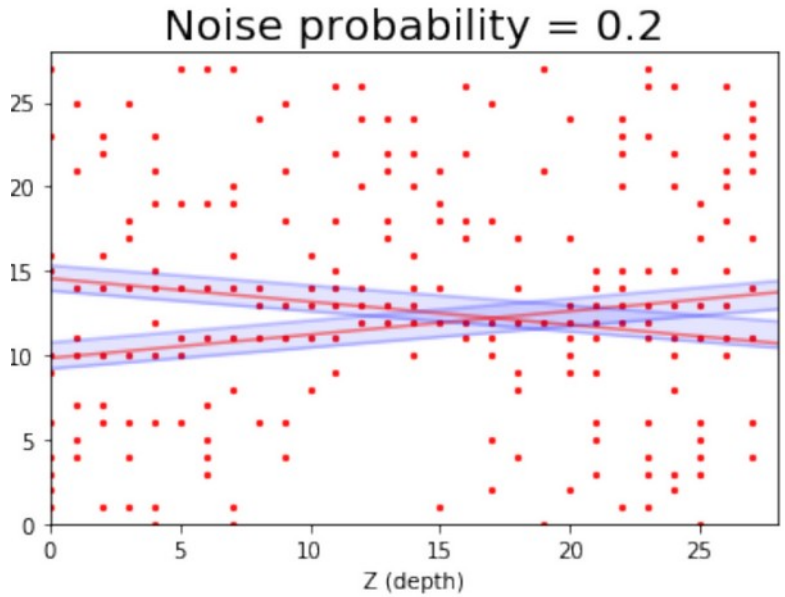


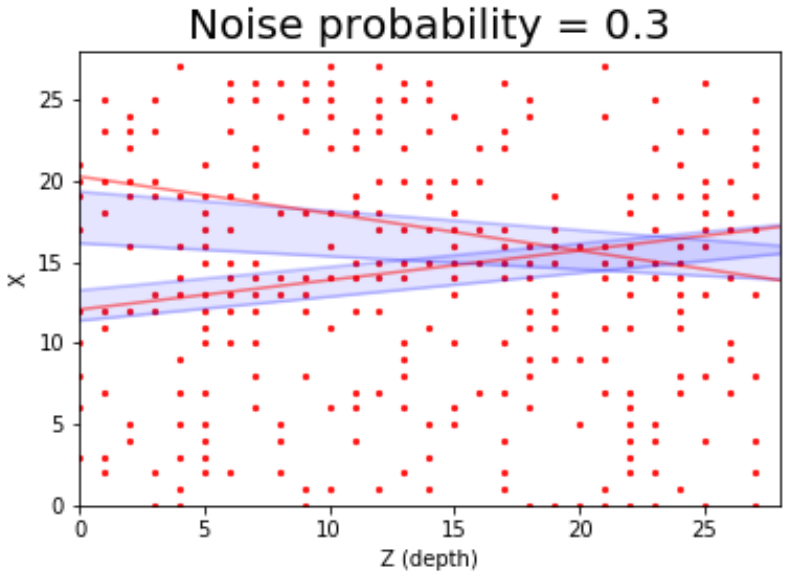
Track described by to interception parameters b_1 and b_2.

Mixed Density Network



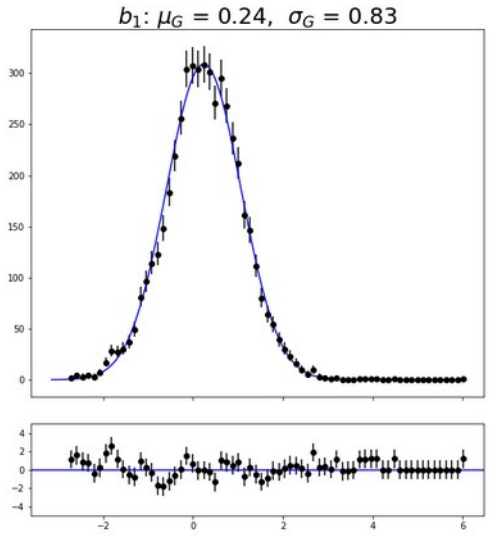


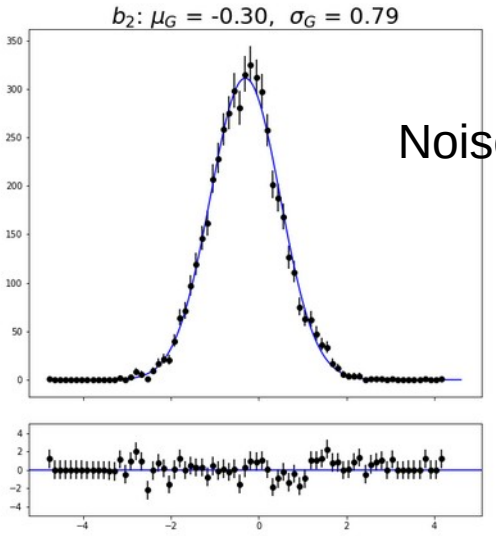




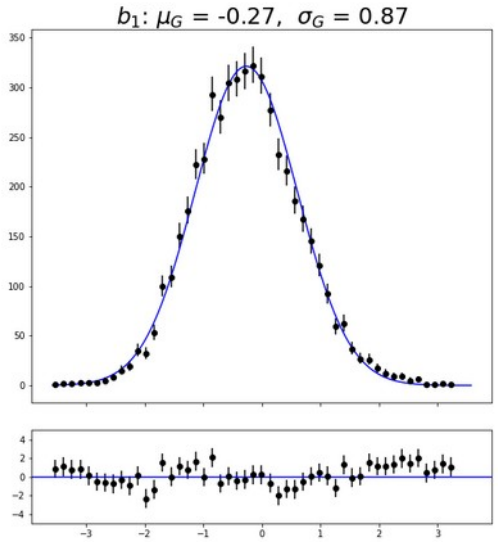
Designed by **Karol Białas** and **Mateusz Słysz** summer students at IFJ:
https://github.com/marcinwolter/MachineLearnin2019/blob/master/dnn_tracking_2D_mdn_multimod.ipynb

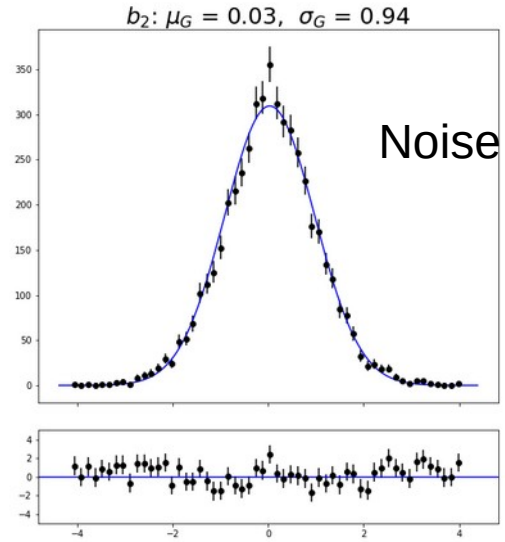
MDN Tracking – error estimation





Noise 10%





Noise 30%

Pull plots have a width not far from 1.

If a random variable x is generated repeatedly with a Gaussian than the pull distribution:

$$g = \frac{x - \mu}{\sigma}$$

will be distributed as a standard Gaussian with mean zero and unit width.