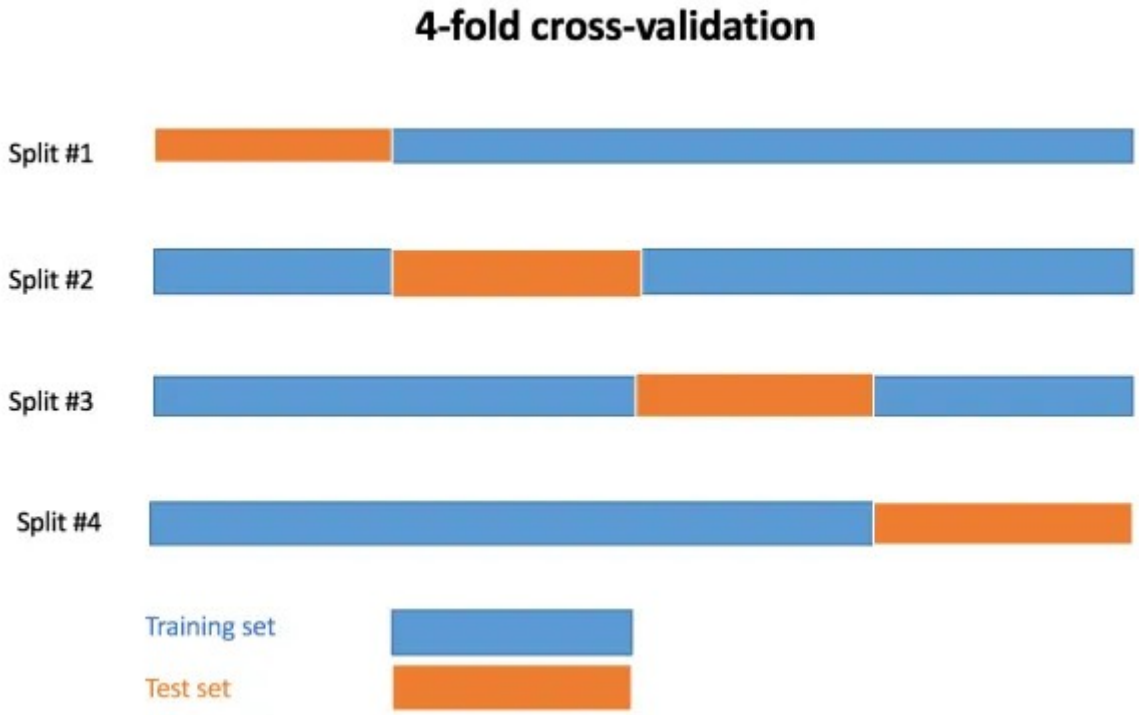
Głębokie uczenie
Wykład 5



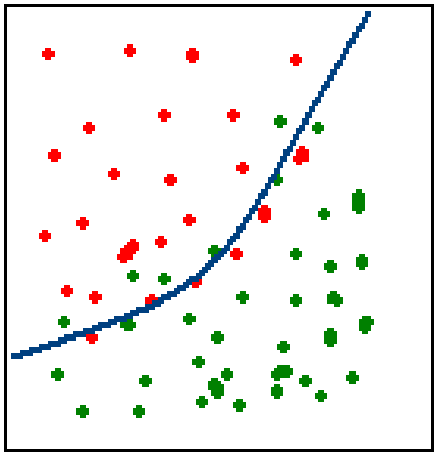
Marcin Wolter

IFJ PAN

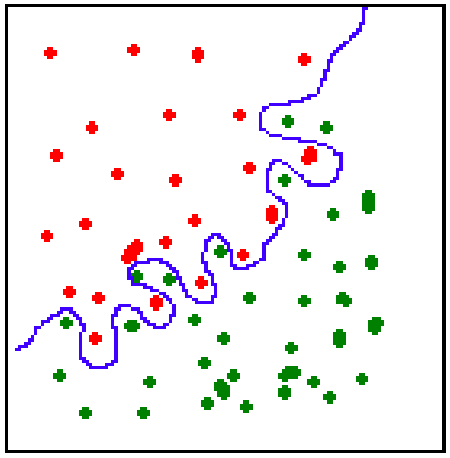
- Trening – walidacja krzyżowa (cross-validation).
- Bayesowskie sieci neuronowe

16 grudnia 2020

Przeuczenie (Overtraining)

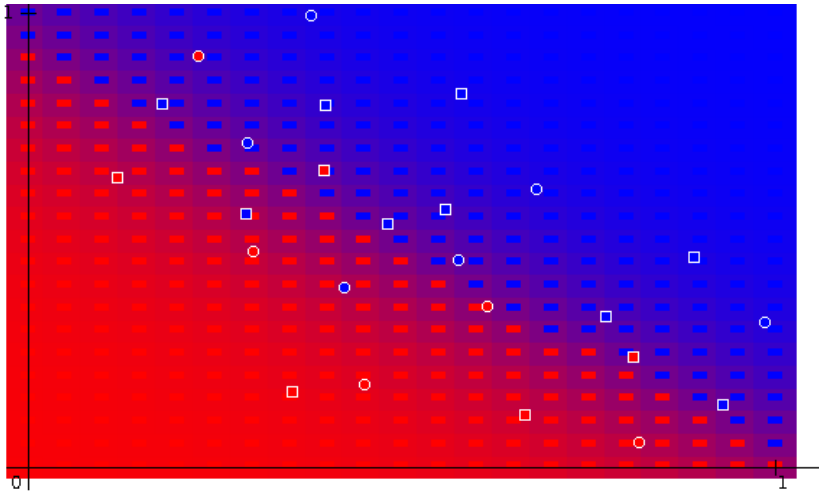


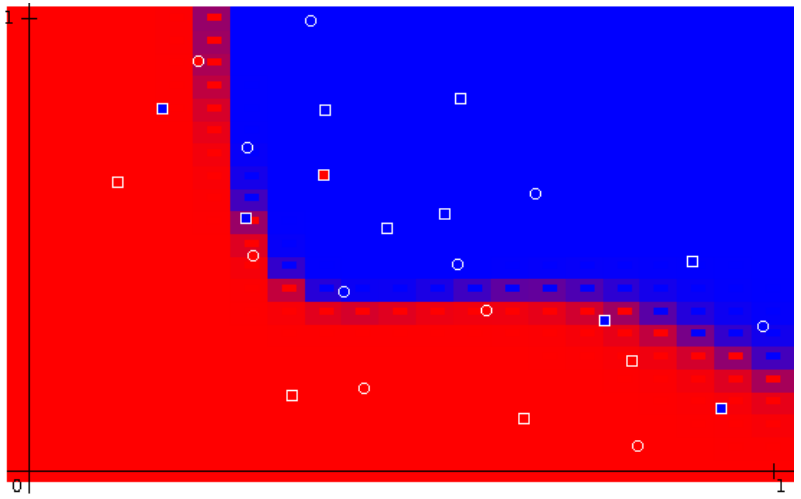
Dobrze

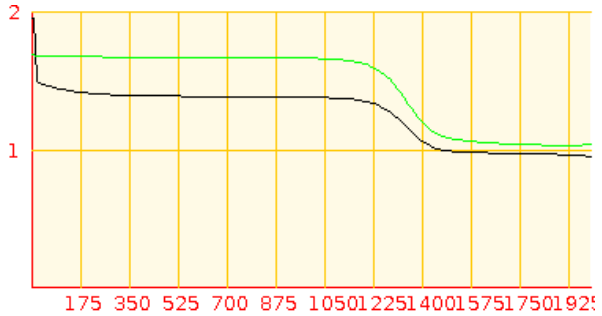


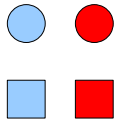
Przetrenowanie

- **Przeuczenie** – Algorytm uczy się poszczególnych przypadków, a nie ogólnych zasad.
- Efekt pojawia się we wszystkich algorytmach uczących
- Remedium – sprawdzenie na oddzielnym zbiorze danych.



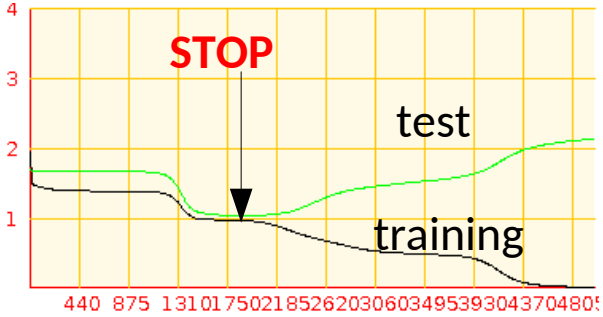






Training sample

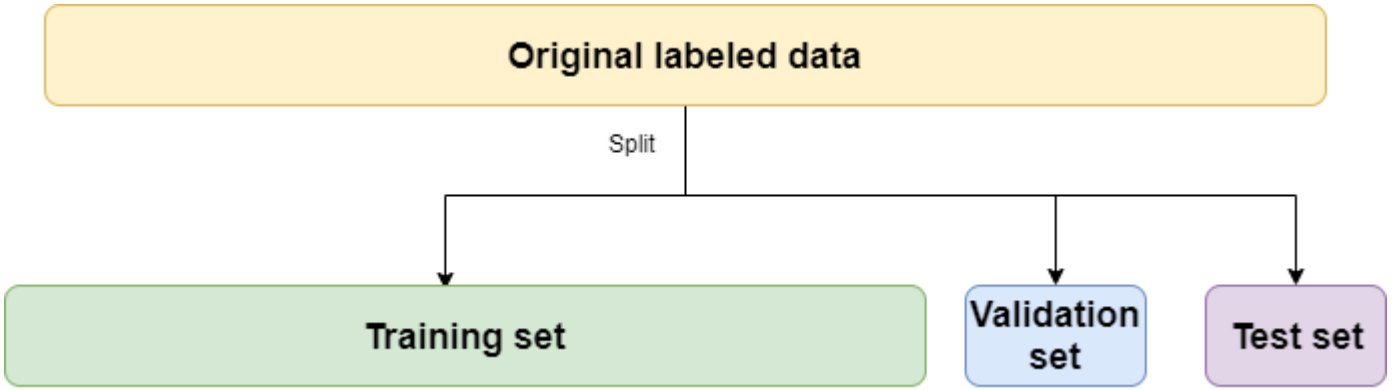
Test sample



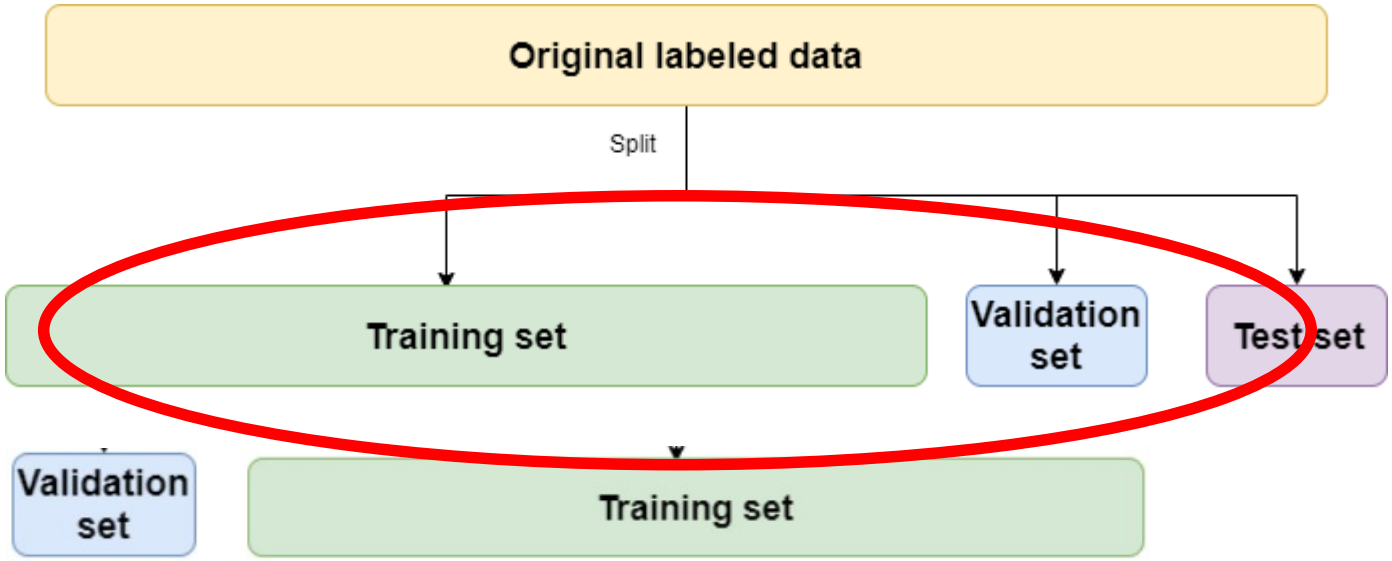
Przykład z siecią neuronową.

Jak ternować sieć neuronową

- Jak zapobiegać **przeuczeniu** (**overtraining)**?
- Zbiory **treningowy** i **walidacyjny**
- **Uwaga:** aby zapobiegać przeuczeniu powinniśmy podzielić dane na trzy podzbiory: treningowy, walidacyjny i testowy:
  - **Treningowy** – do treningu
  - **Walidacyjny** – sprawdzamy w trakcie treningu działanie sieci. Na tej podstawie dobieramy hiperparametry (np. liczba warstw, liczba neuronów, funkcja aktywacji itp.)
  - **Testowy** – sprawdzamy jak działa wytrenowany algorytm



A może inaczej?



- Możemy podzielić dane inaczej pomiędzy zbiór treningowy i testowy
- Możemy stworzyć wiele takich podziałów i wytrenować wiele sieci...
- ... i potem użyć ich razem.

Walidacja krzyżowa (cross-validation)

- Mamy niezależne zbiory treningowy L_n oraz testowy T_m.
- Błąd klasyfikatora $\hat{d}(\boldsymbol{x}) = \hat{d}(\boldsymbol{x};\ \mathcal{L}_n)$ na bazie zbioru terningowego L_n

$$\hat{e}_T = \frac{1}{m}\sum_{j=1}^{m} I\left(\hat{d}(\boldsymbol{X}_j^t; \mathcal{L}_n) \neq Y_j^t\right)$$

- Estymator używający "danych z recyklingu" (te dane do trening i do walidacji) jest obciążony (biased).
- Redukcja obciążenia: podział danych na dwie części (training & validation). Ale wtedy używamy do treningu tylko części danych.
- **Walidacja krzyżowa** – ze zbioru L_n usuwamy tylko jeden przypadek, trenujemy klasyfikator i walidujemy na tym jednym przypadku. Powtarzamy n razy i uzyskujemy estymator będący średnią n estymatorów:

$$\hat{e}_{CV} = \frac{1}{n}\sum_{j=1}^{n} I\left(\hat{d}(\boldsymbol{X}_j; \mathcal{L}_n^{(-j)}) \neq Y_j\right)$$

- Uzyskujemy nieobciążony estymator (w granicy dużych n), niestety kosztem wielkiego zużycia CPU.

Walidacja krzyżowa

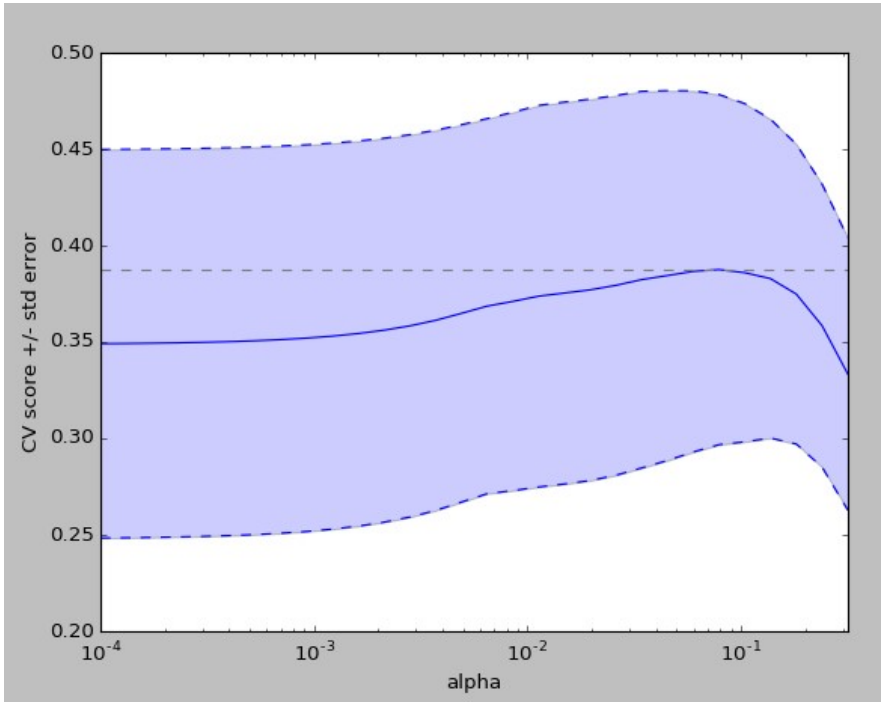
- Rozwiązanie pośrednie – ***k-fold cross-validation***
- Zbióra danych jest podzielony na *k* podzbiorów, k-1 z nich jest używanych do treningu, jeden do walidacji. Procedura jest powtarzana z kolejnymi podzbiorami k razy

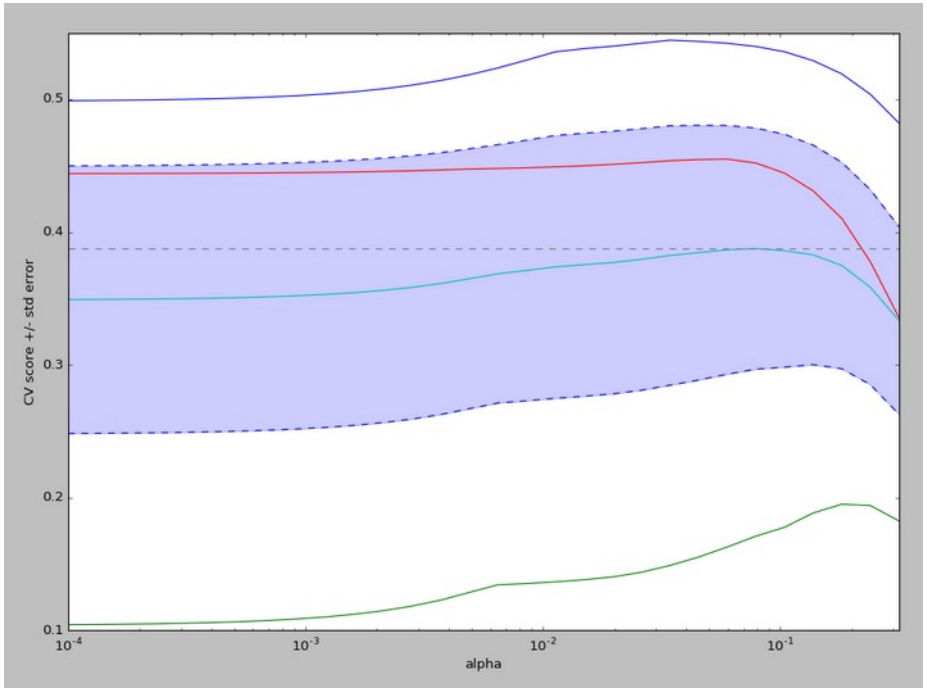
$$\hat{e}_{vCV} = \frac{1}{n} \sum_{i=1}^{v} \sum_{j=1}^{n} I(\boldsymbol{Z}_j \in \tilde{\mathcal{L}}_n^{(i)}) I\left(\hat{d}(\boldsymbol{X}_j; \tilde{\mathcal{L}}_n^{(-i)}) \neq Y_j\right)$$

- Niższe zużycie CPU, tylko *k*<<*n* treningów.
- Typowo używamy k ~ 10 .
- Końcowy klasyfikator w najprostszej wersji jest średnią wszystkich k klasyfikatorów (lub można go zbudować z k klasyfikatorów w inny sposób).

Walidacja krzyżowa

- 4-times folding
- Znajdowanie zależności CV od alpha (dane medyczne)
- Możemy znaleźć średnią i odchylenie standardowe. Na drugim rysunku zależność dla każdego "folding".
- Udało nam się estymować także błąd estymacji funkcji!





Model performance

Test Error

- Partition the original data (randomly) into a training set and a test set. (e.g. 70/30)
- Train a model using the training set and evaluate performance (a single time) on the test set.

K-fold Cross-validation

- Train & test K models as shown.
- Average the model performance over the K test sets.
- Report cross-validated metrics.

Performance Metrics

- Regression: R^2, MSE, RMSE
- Classification: Accuracy, F1, H-measure, Log-loss
- Ranking (Binary Outcome): AUC, Partial AUC

https://www.slideshare.net/0xdata/top-10-data-science-practitioner-pitfalls

Train vs Test vs Valid

Training Set vs. Validation Set vs. Test Set

- If you have “enough” data and plan to do some model tuning, you should really partition your data into three parts — Training, Validation and Test sets.
- There is **no general rule** for how you should partition the data and it will depend on how strong the signal in your data is, but an example could be: 50% Train, 25% Validation and 25% Test

Validation is for Model Tuning

- The validation set is used **strictly for model tuning** (via validation of models with different parameters) and the test set is used to make a final estimate of the generalization error.

Przykład walidacji krzyżowej

- Bierzemy jako przykład rozpoznawanie ręcznie pisanych cyfr MNIST

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

- Dodajemy walidację krzyżową:

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

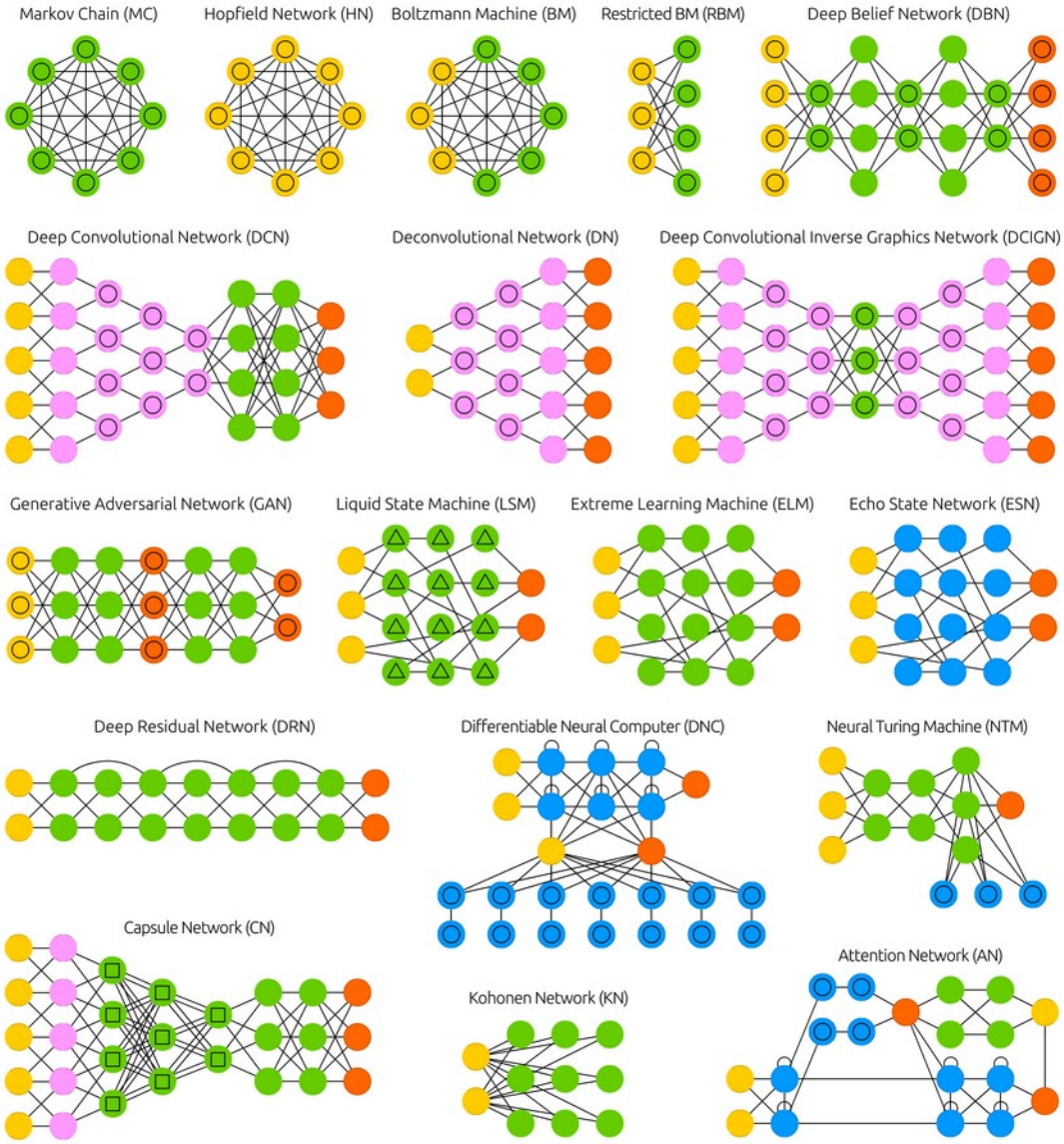
- Używamy

  from sklearn.model_selection import Kfold

  z pakietu sklearn.

- Uwaga: użycie 10-krotnej walidacji krzyżowej daje kombinowany klasyfikator lepszy niż każdy z poszczególnych klasyfikatorów.

Podejście bayesowskie



- Bayesowskie sieci neuronowe
- Mixed Density Network MDN

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

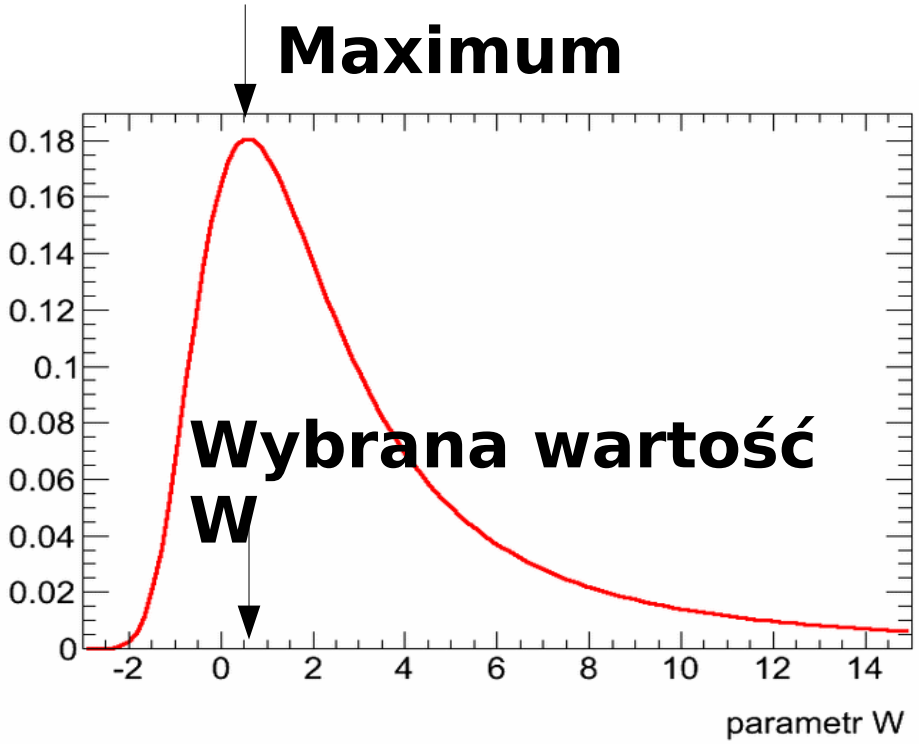
Dla każdej funkcji f(x) z przestrzeni F znajdujemy prawdopodobieństwo *a posteriori* $\boldsymbol{p(f|T)}$ na podstawie danych $\mathbf{T= (x, y).}$

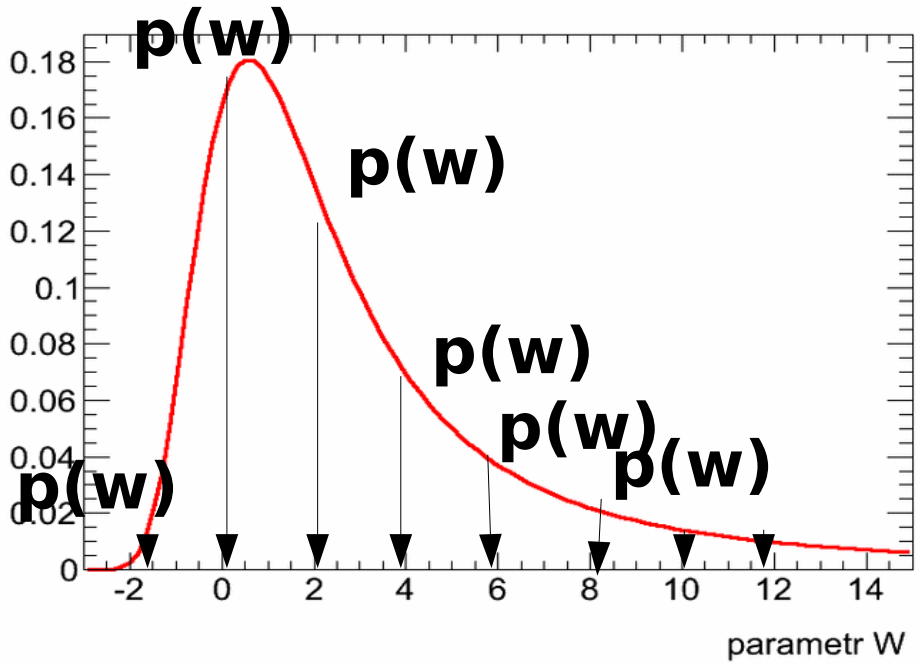
W uczeniu bayesowskim **NIE DOSTAJEMY** jednej, najlepszej funkcji aproksymującej, ale wiele funkcji ważonych ich prawdopodobieństwem.

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

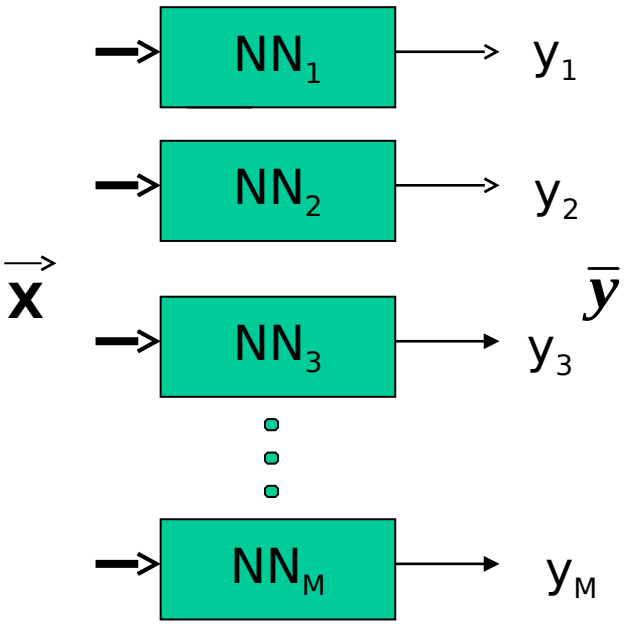
Każda funkcja (lub wartość parametru) posiada pewne prawdopodobieństwo (wagę).

Implementacja sieci bayesowskich

Zamiast wybierać jeden zestaw wag opisujących sieć neuronową znajdujemy rozkład prawdopodobieństwa w całej przestrzeni wag.

Wiele sieci neuronowych

Mając wiele sieci neuronowych możemy znaleźć ważoną średnią I błąd estymacji.



$$\bar{y} = \sum_i a_i \, y_i(x)$$

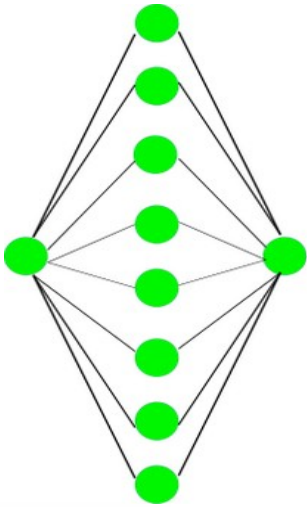
C.M. Bishop
"Neural Networks for Pattern Recognition", Oxford 1995

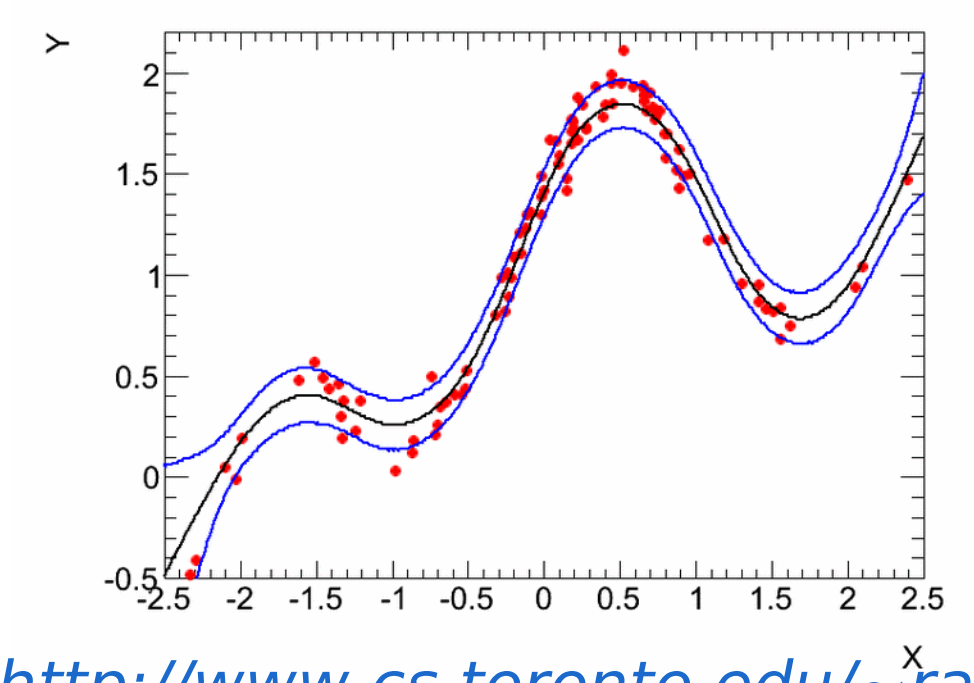
Free software:
Radford Neal, http://www.cs.toronto.edu/~radford/fbm.software.html

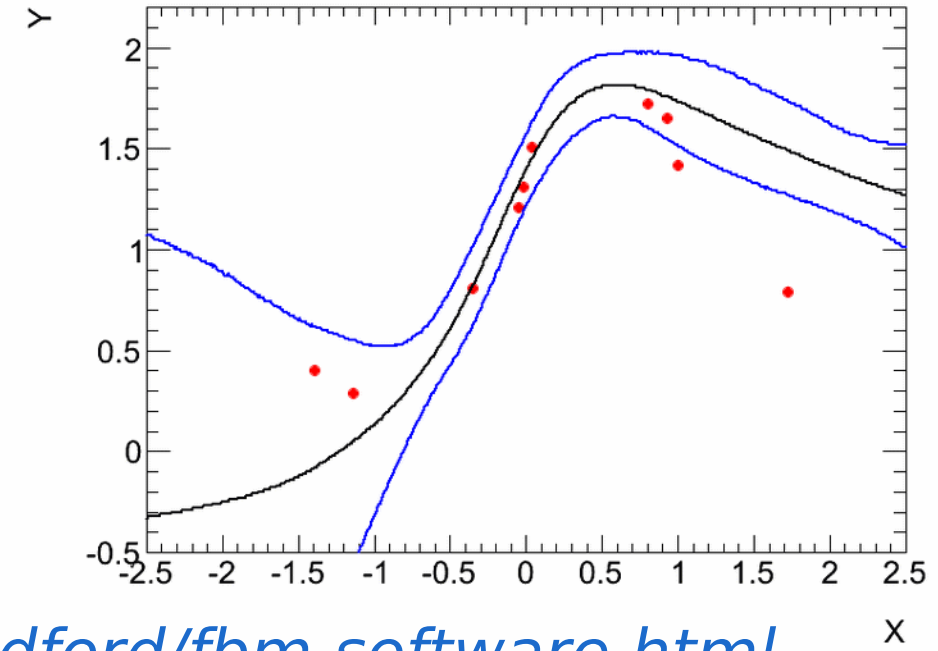
Przykład BNN

Sieć bayesowska z 8 neuronami w warstwie ukrytej

- Dane generowane z funkcji: $y = 0.3 + 0.4x + 0.5\sin(2.7x) + 1.1/(1+x^2)$ z szumem gaussowskim (sigma = 0.1)
- 400 sieci neuronowych, z rozkładu odpowiedzi mamy: medianę i 10% Qnt i 90% Qnt (10% sieci daje przewidywania poniżej dolnej niebieskiej linii i 10% powyżej górnej).
- Przykłady różnej liczby punktów.

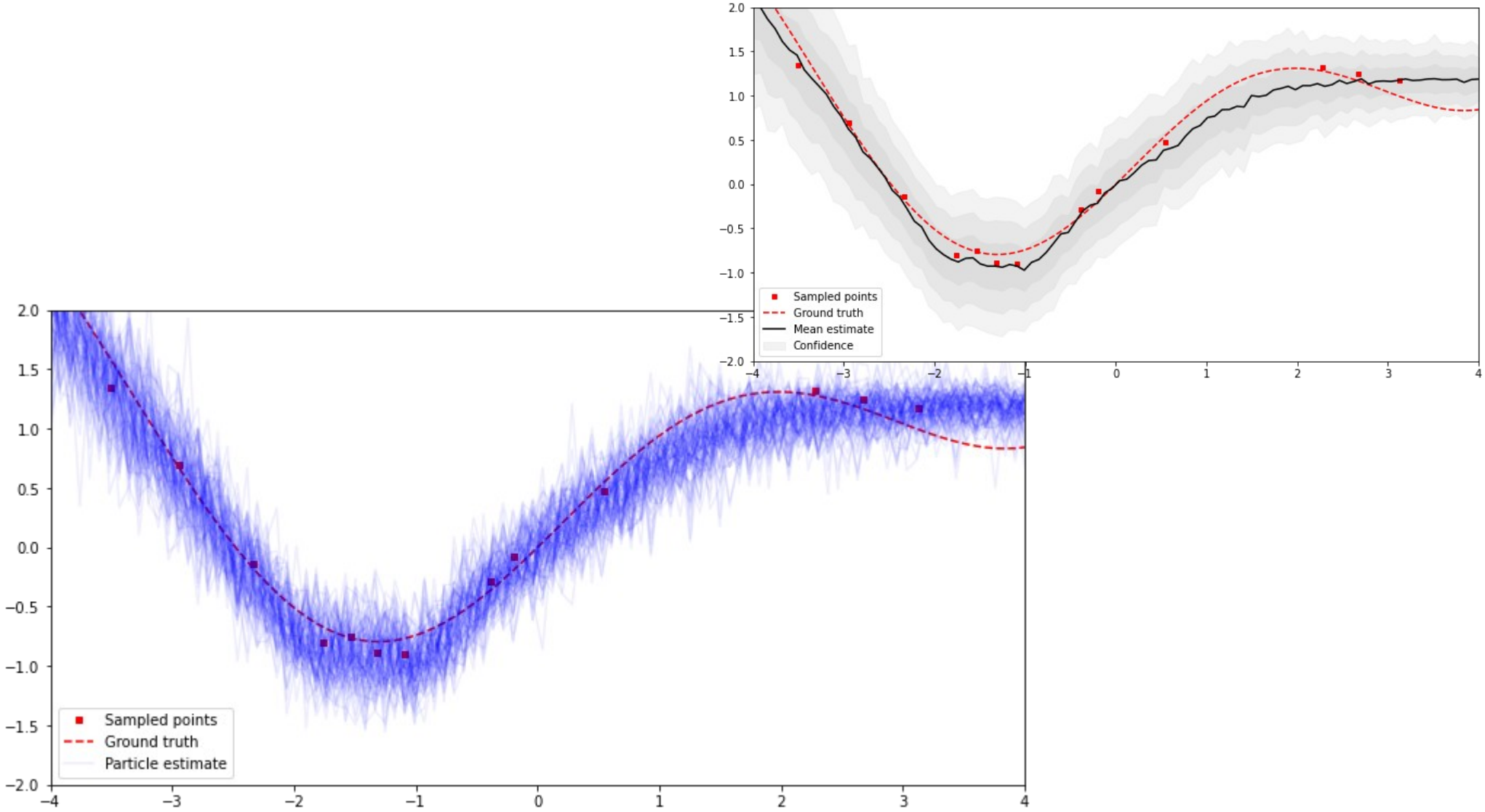






http://www.cs.toronto.edu/~radford/fbm.software.html

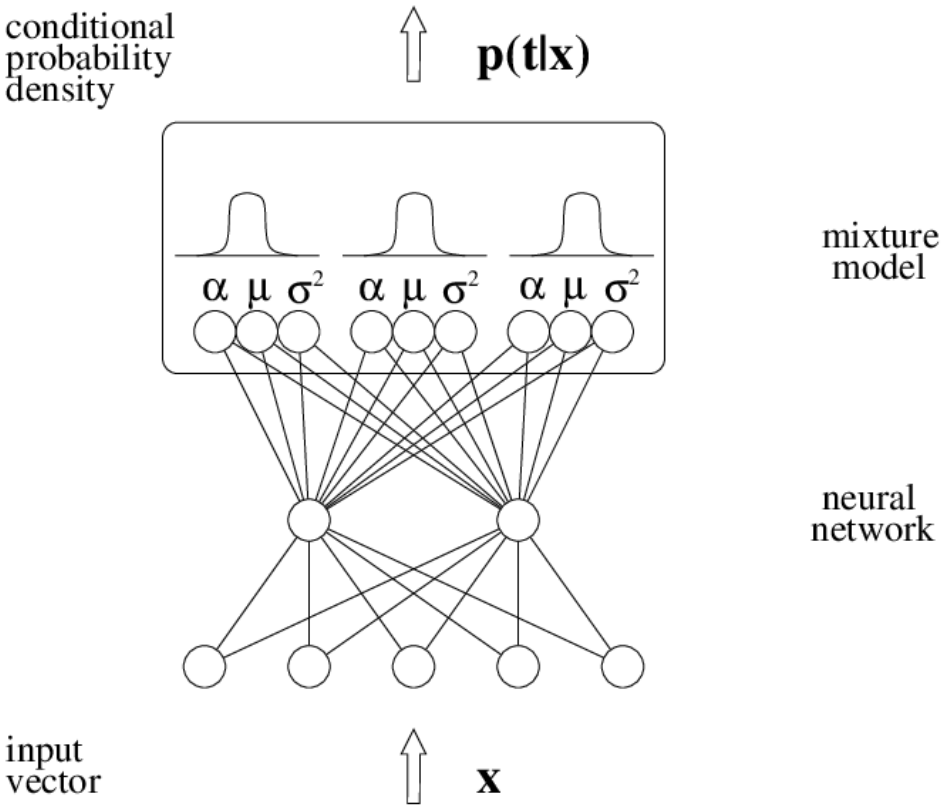
Przykład bayesowskiej sieci neuronowej



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

Następny wykład

- 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 Mixed Density Network (MDN).
Rozkład prawdopodobieństwa jest sparametryzowany przez kilka rozkładów Gaussa.