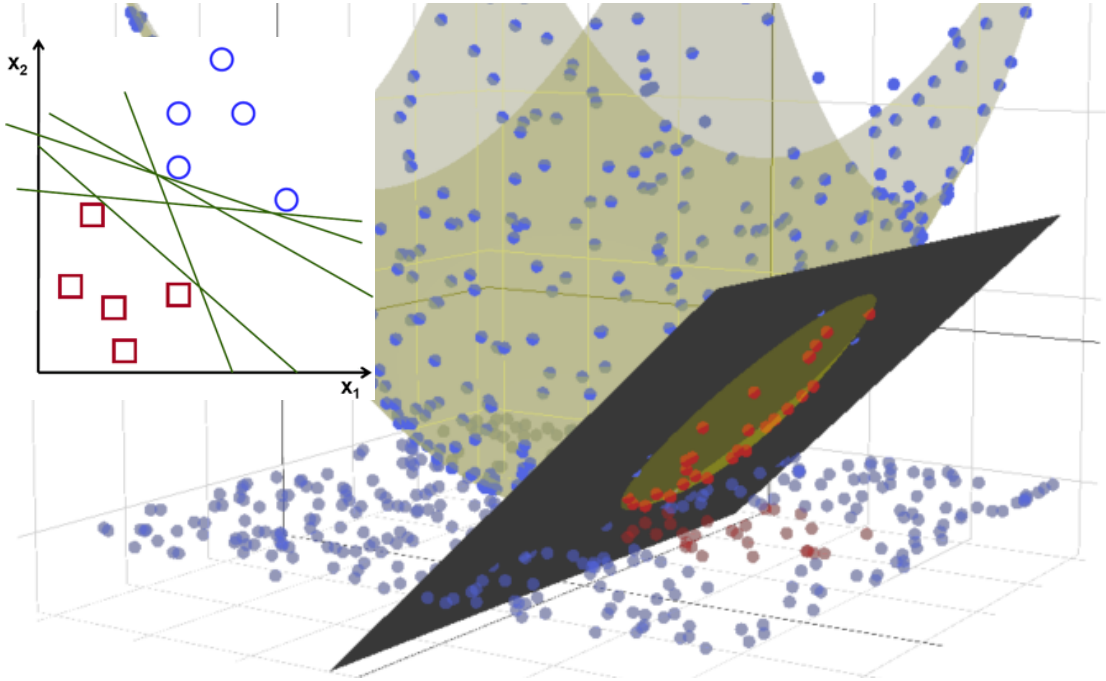
Machine learning

Lecture 5



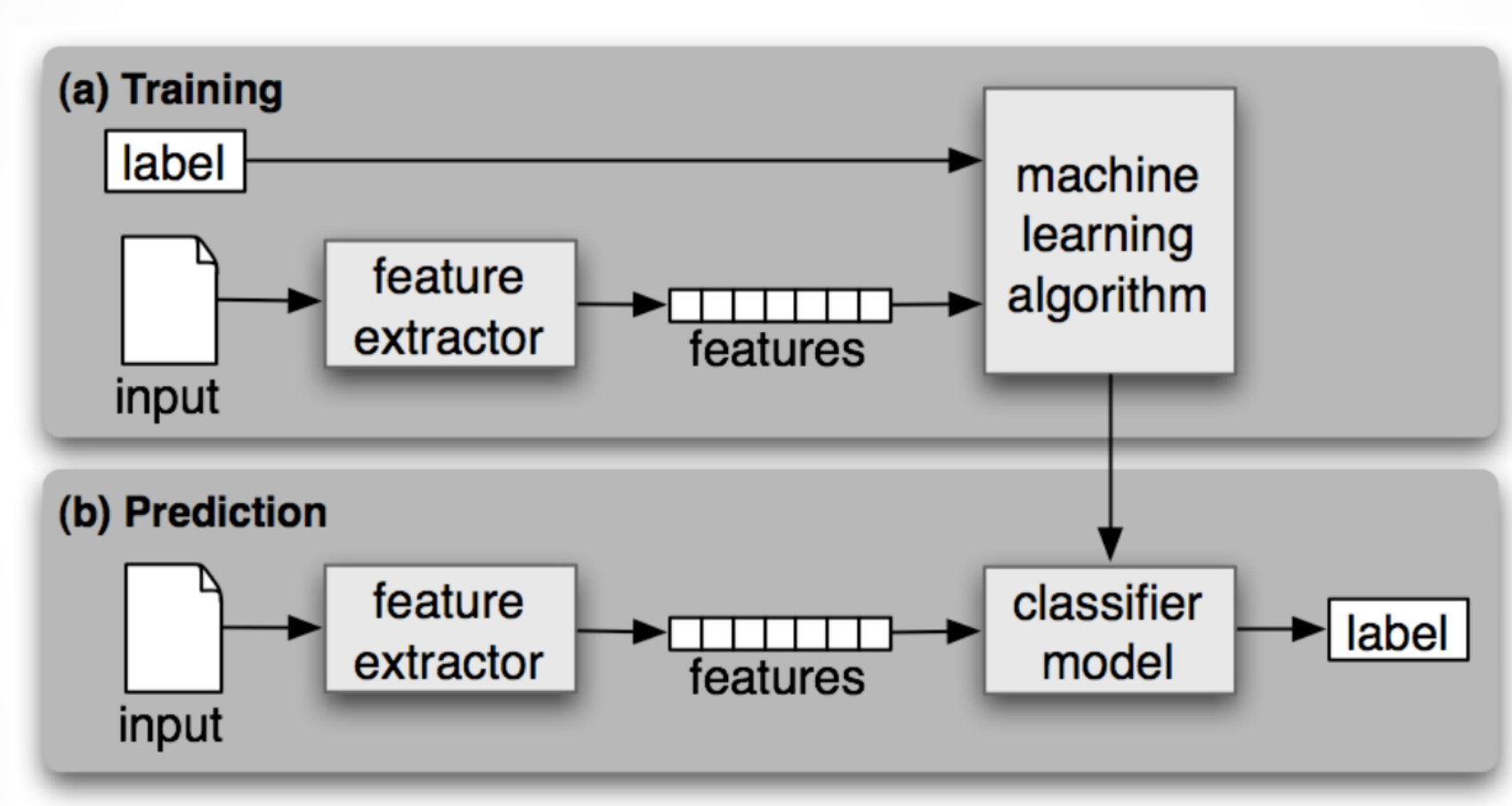
Marcin Wolter

IFJ PAN

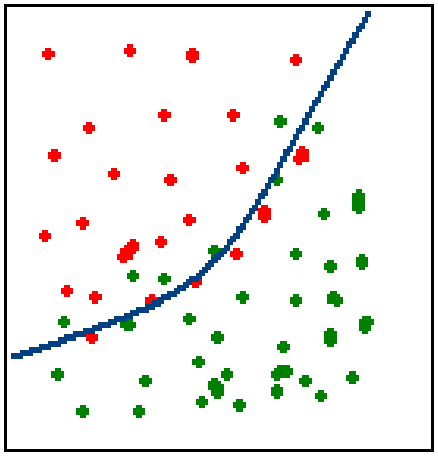
21 kwietnia 2017

- Uczenie -sprawdzanie krzyżowe (cross-validation).
- Optymalizacja parametrów metod uczenia maszynowego.
- Deep learning

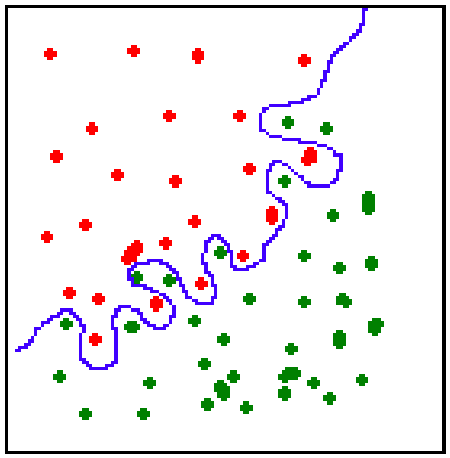
Uczenie z nauczycielem



Przetrenowanie

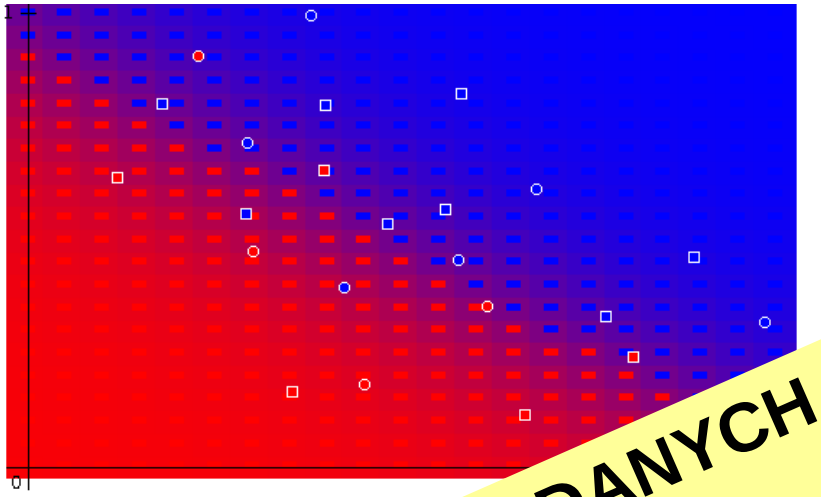


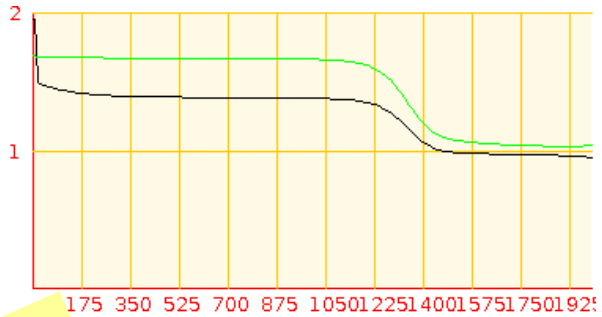
Poprawnie



Przetrenowanie

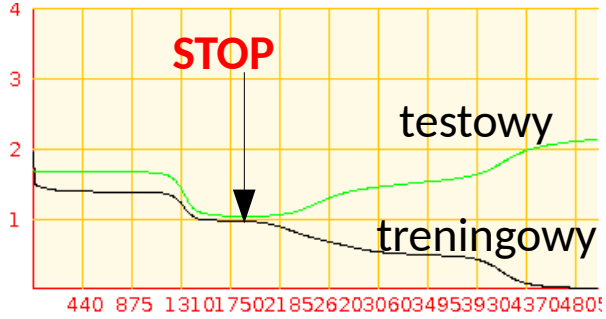
- **Przetrenowanie** – algorytm “uczy się” poszczególnych przypadków, a nie ogólnych zasad.
- Efekt występuje we wszystkich metodach uczących się.
- Remedium – kontrola z użyciem dodatkowego zbioru danych.





zbiór treningowy

zbiór testowy



Przykład z użyciem sieci neuronowej.

ALE UŻYWAMY TYLKO CZĘŚCI DANYCH DO UCZENIA!!!

Jak uczyć z nauczycielem?

- Jak uczyć algorytm aby uniknąć przeuczenia?
- Używać jednego zbioru do uczenia a drugiego do walidacji?
- Prowadzi to do zawyżenia estymatora błędu (bo używamy części danych).

- Drugi problem: w zasadzie aby zminimalizować przeuczenie powinniśmy używać trzeciego zbioru danych do ostatecznego wyznaczenia wydajności algorytmu.

- Jak zoptymalizować parametry algorytmu użytego do uczenia (np. liczba drzew w BDT, liczba ukrytych warstw/nodów w sieci neuronowej)?

Walidacja

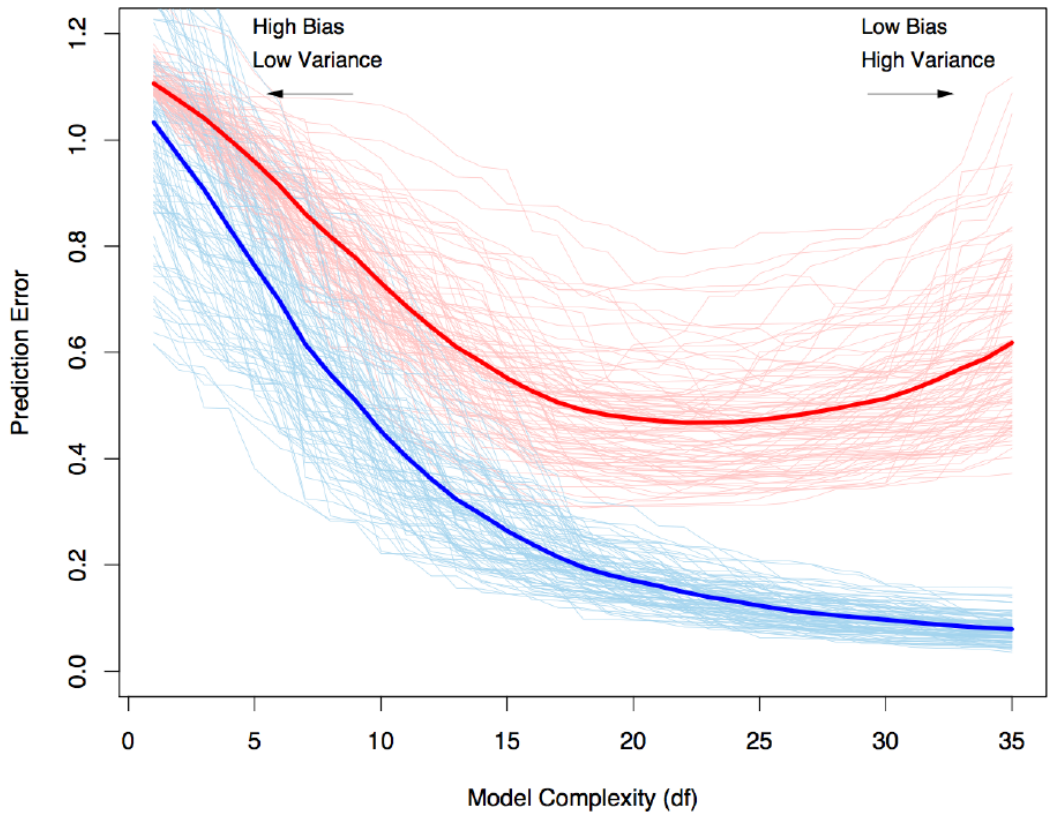


FIGURE 7.1. *Behavior of test sample and training sample error as the model complexity is varied. The light blue curves show the training error* $\overline{\text{err}}$*, while the light red curves show the conditional test error* $\text{Err}_{\mathcal{T}}$ *for* 100 *training sets of size* 50 *each, as the model complexity is increased. The solid curves show the expected test error* Err *and the expected training error* $\text{E}[\overline{\text{err}}]$.

Source: Elements of Statistical Learning

Walidacja krzyżowa – cross-validation

- Mamy niezależne zbiory uczący L_n oraz testowy T_m.
- Poziom błędu klasyfikatora $\hat{d}(\boldsymbol{x}) = \hat{d}(\boldsymbol{x};\ \mathcal{L}_n)$ skonstruowanego za pomocą próby uczącej 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 ponownego podstawienia (te same dane użyte do nauki i do określenia błędu) jest estymatorem obciążonym.
- Zmniejszenie obciążenia: np.. podział danych na dwie części – wykorzystanie tylko części informacji.
- Sprawdzanie krzyżowe – z próby L_n usuwamy przypadek j, trenujemy, sprawdzamy na jednej obserwacji j. Powtarzamy n razy. Otrzymujemy estymator:

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

Estymator granicznie nieobciążony, wymagający obliczeniowo, ma większą wariancję niż e_T.

Walidacja krzyżowa

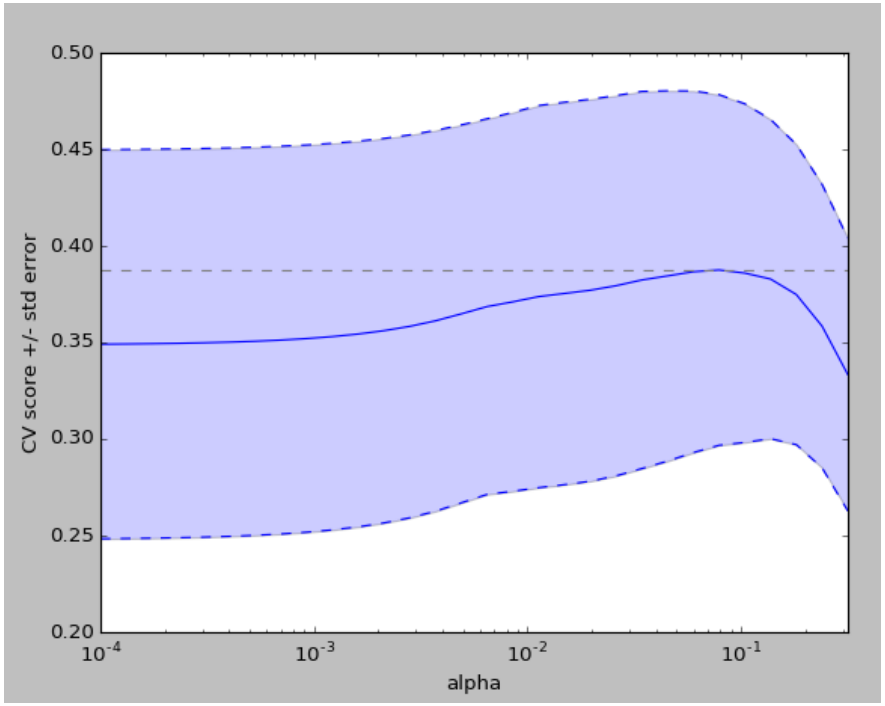
- Rozwiązanie pośrednie – *v-fold cross-validation*
- Podział próby na v podzbiorów, v-1 z nich używamy do uczenia, jeden do sprawdzenia. Powtarzamy v razy.

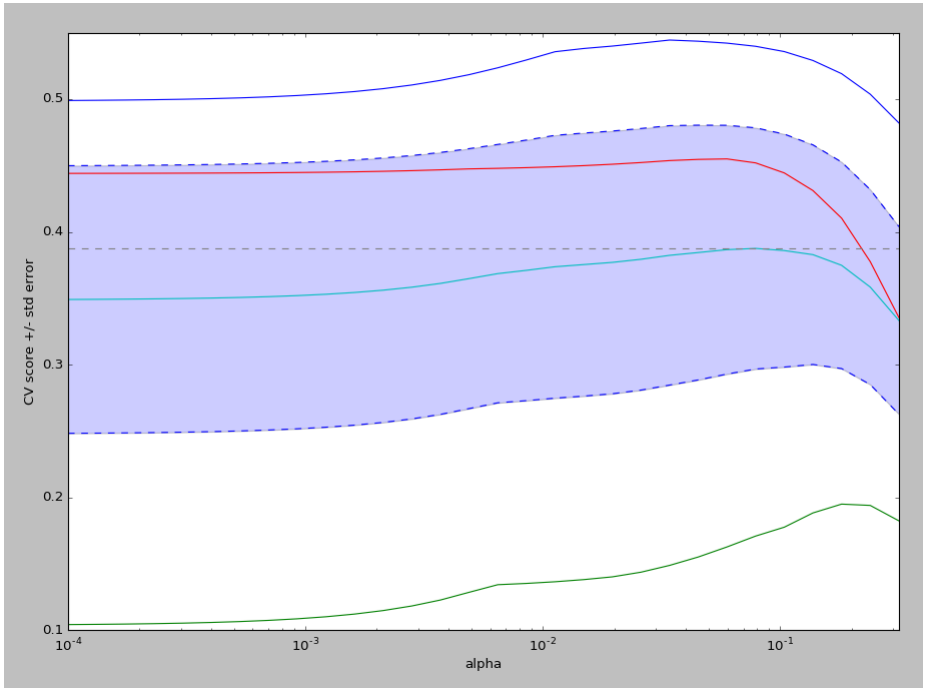
$$\hat{e}_{vCV} = \frac{1}{n}\sum_{i=1}^{v}\sum_{j=1}^{n} I(\boldsymbol{Z}_j \in \tilde{\mathcal{L}}_n^{(i)}) I\left(\hat{d}(\boldsymbol{X}_j; \tilde{\mathcal{L}}_n^{(-i)}) \neq Y_j\right)$$

- Mniejsza ilość obliczeń w porównaniu ze sprawdzaniem krzyżowym.
- Zalecane v ~ 10 w zagadnieniu estymacji poziomu błędu.
- W zagadnieniu wyboru modelu (np. przy dobieraniu parametrów modelu) wybieramy wtedy klasyfikator, dla którego błąd klasyfikacji ma wartość najmniejszą.

Walidacja krzyżowa

- 4-krotny folding
- Znajdowanie zależności CV od alpha (dane medyczne)
- Rysuję średnią i odchylenie standardowe oraz, na drugim rysunku, poszczególne zależności dla każdego v
- Dostaję szacunkowy błąd estymacji CV





Tab. 1.1. Wartości błędów klasyfikacji dla różnych liczebności próby uczącej

	$\hat{d}_{100}$	$\hat{d}_{200}$	$\hat{d}_{500}$	$\hat{d}_{1000}$
$\hat{e}_R$	0.0065	0.0110	0.0148	0.0164
$\hat{e}_{CV}$	0.0115	0.0150	0.0242	0.0200
$\hat{e}_{10CV}$	0.0138	0.0157	0.0239	0.0197
$\hat{e}_{J'}$	0.0143	0.0151	0.0247	0.0198
$\hat{e}_{B_1}$	0.0125	0.0224	0.0362	0.0332
$\hat{e}_{B_2}$	0.0194	0.0325	0.0323	0.0288
$\hat{e}_{.632}$	0.0147	0.0246	0.0280	0.0250
$\hat{e}_{.632+}$	0.0149	0.0248	0.0282	0.0251
$\hat{e}_{\mathcal{T}}$	0.0262	0.0215	0.0197	0.0187

Bootstrap

Ostatni wiersz tab. 1.1 zawiera estymator $\hat{e}_{\mathcal{T}}$ aktualnego poziomu błędu klasyfikatora uzyskany z niezależnie wygenerowanej próby testowej o liczbie elementów $m = 100\,000$. Nie będzie więc dużym nadużyciem (ze względu na liczebność próby testowej), jeżeli ten błąd klasyfikacji przyjmiemy za aktualny poziom błędu klasyfikatorów $\hat{d}_n$.

Wobec powyższej uwagi zauważmy, że dla $n = 100$ wszystkie estymatory zaniżają (niedoestymowują) aktualny poziom błędu. To znaczne obciążenie wszystkich ocen jest efektem zbyt małej próby (pamiętajmy, że obserwacje pochodzą z przestrzeni 8-wymiarowej). Dla $n = 200$ niedoszacowanie aktualnego poziomu błędu obserwujemy dla estymatora ponownego podstawienia, sprawdzania krzyżowego oraz dla estymatora typu jackknife, natomiast w przypadku większych prób – dla wszystkich ocen z wyjątkiem estymatora resubstytucji.

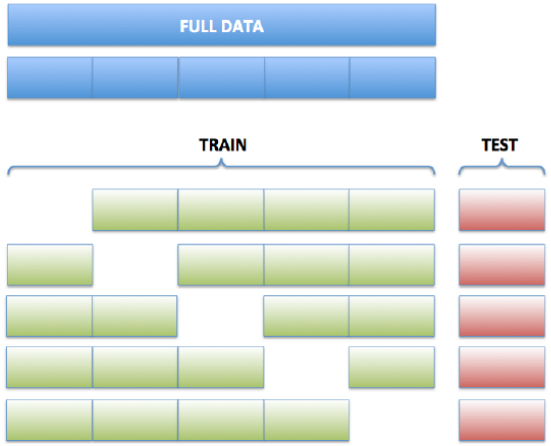
Model performance

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

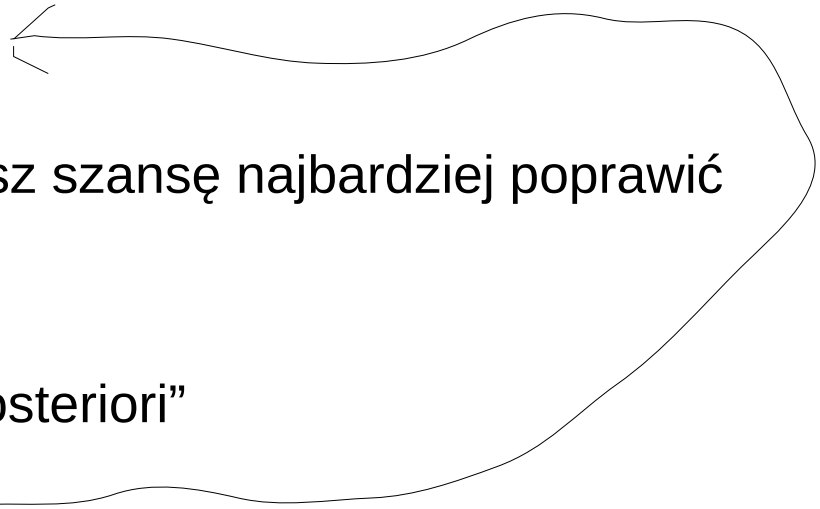
- The validation set is used **strictly for model tuning** (via validation of models with different parameters) and the test set is used to make a final estimate of the generalization error.

Optymalizacja hiperparametrów

- Każda (prawie) metoda ma szereg parametrów (np. struktura sieci neuronowej).
- Powinny one być zoptymalizowane dla danego problemu.
- **Zadanie: dla danego zbioru danych znaleźć taki zestaw parametrów używanej metody uczenia maszynowego, aby estymowany błąd metody była jak najmniejszy.**
- Cechy szczególne naszego problemu:
  - Uzyskanie kolejnego pomiaru jest kosztowne
  - Znaczący poziom szumu
  - Możemy obliczyć wartość funkcji (czyli nasz błąd estymatora) w punkcie **x**, ale nie możemy obliczyć pochodnej.

Optymalizacja hiperparametrów

- Jak optymalizować:
  - „Grid search” - skanowanie możliwych wartości parametrów.
  - „Random search”
  - Jakieś dopasowanie…
- Popularną metodą jest **„optymalizacja bayesowska”**
  - Zbuduj model prawdopodobieństwa
  - Weź rozkład „a priori” parametrów
  - Znajdź dla jakiej wartości parametrów masz szansę najbardziej poprawić model
  - Oblicz wartość błędu
  - Znajdź rozkład prawdopodobieństwa „a posteriori”
  - Powtórz



Jak to działa w praktyce?

- Dopasowanie prostej

  $y(x, w) = w_0 + w_1 x$ do danych.

1) Prior gaussowski, nie ma danych

2) Jeden punkt. Obliczamy likelihood na podstawie tego punktu. Mnożymy rozkład a priori*likelihood, otrzymujemy rozkład a posteriori.

3) Dodajemy drugi punkt i powtarzamy procedurę.

4) Dodajemy następne punkty danych.

Uwaga: dane obarczone szumem.

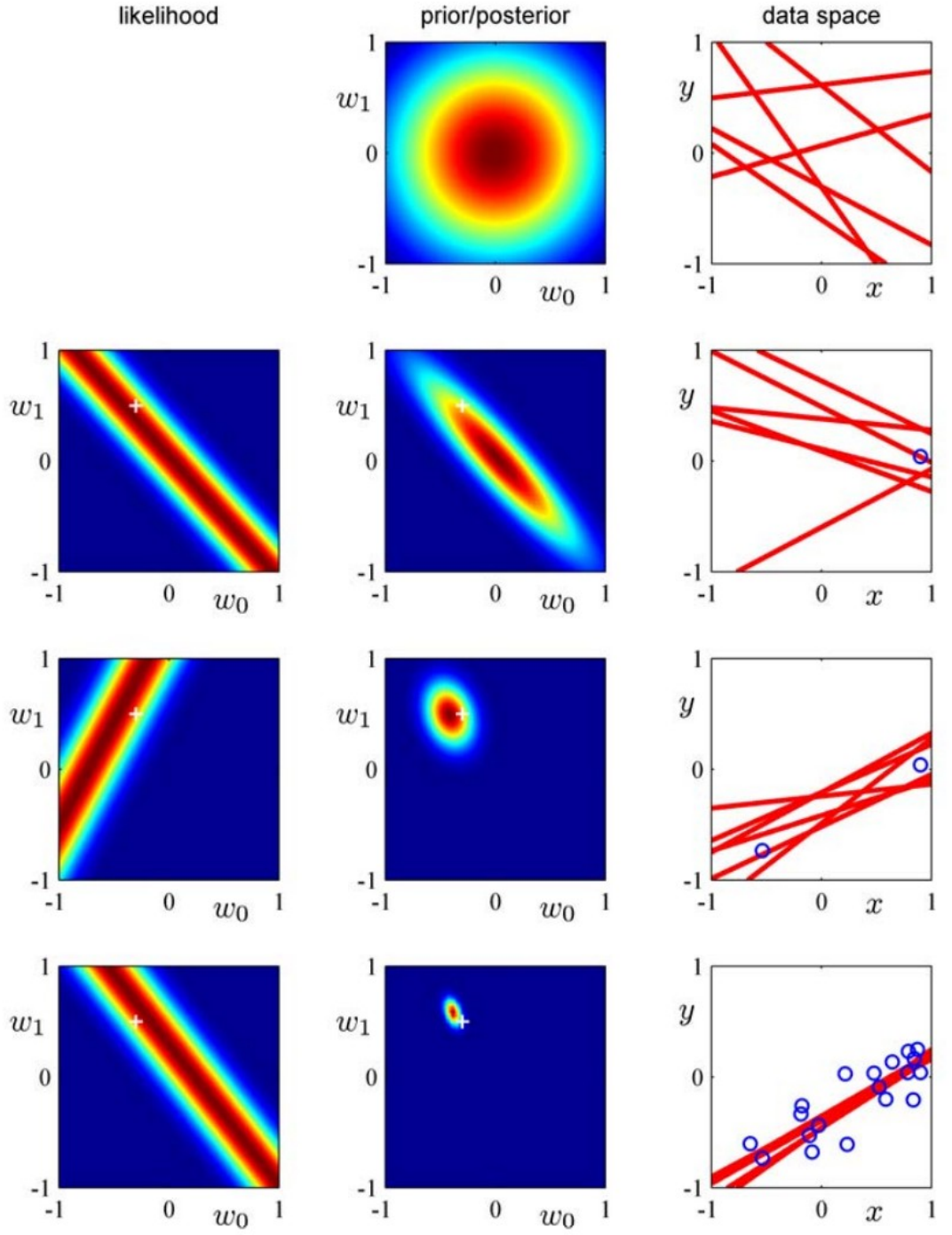
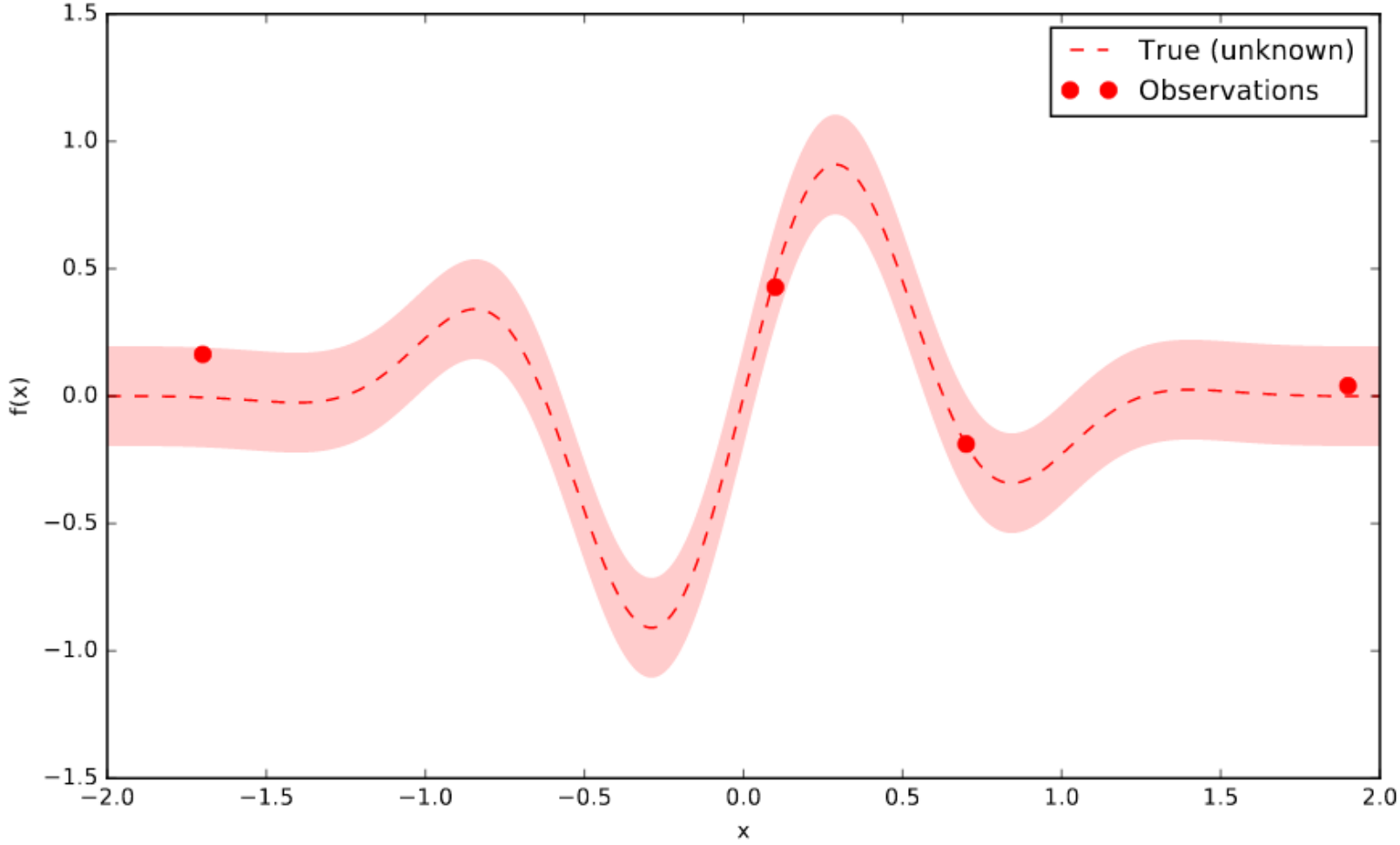


Illustration of sequential Bayesian learning for a simple linear model of the form $y(x, \mathbf{w}) = w_0 + w_1 x$. A detailed description of this figure is given in the text.

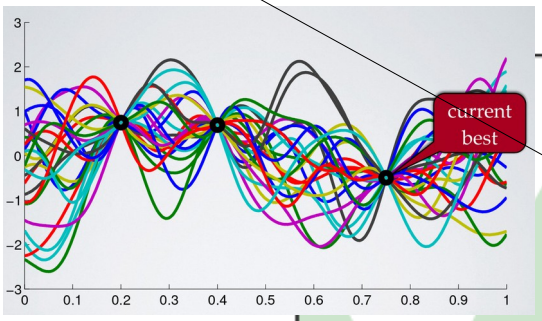
Punkt startowy



Nieznana funkcja (z szumem), cztery obserwacje.
Gdzie przeprowadzić następne (kosztowne) próbkowanie?

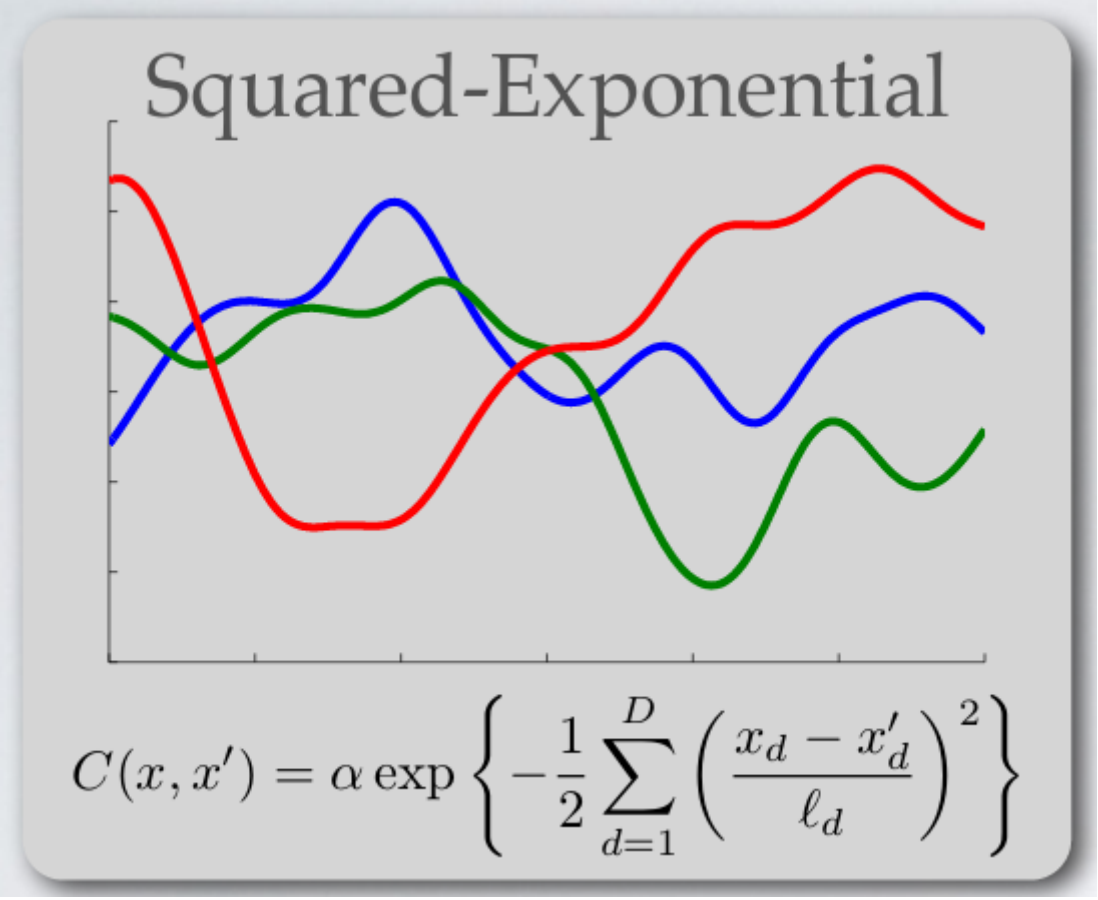
Rozkład funkcji a posteriori

Szereg funkcji



Rozkład a posteriori możliwych funkcji, które mogłyby wygenerować obserwowane dane.

Funkcje a posteriori – Gaussian Processes (GP)



Funkcje muszą być jakoś sparametryzowane. Np. (ale niekoniecznie) mogą to być funkcje Gaussa.

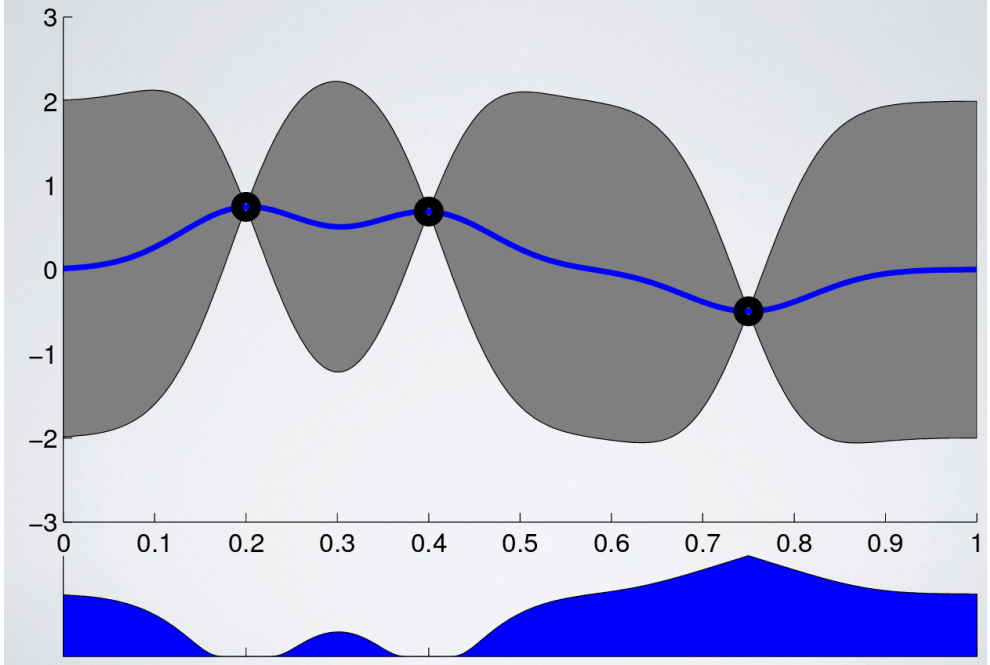
Funkcja akwizycji

- Posterior GP (Gaussian Processes) dają nam średnią z funkcji μ(x) oraz przewidywaną wariancję funkcji $\sigma^2(x)$.
  - Eksploracja (Exploration) – szukamy miejsc z dużą wariancją
  - Eksploatacja (Exploitation) – szukamy miejsc z najwyższą wartością μ(x)
- Funkcja akwizycji musi zbalansować te metody poszukiwania:
  - Probability of Improvement (Kushner 1964):
    - $a_{PI}(x) = \Phi(\gamma(x))$

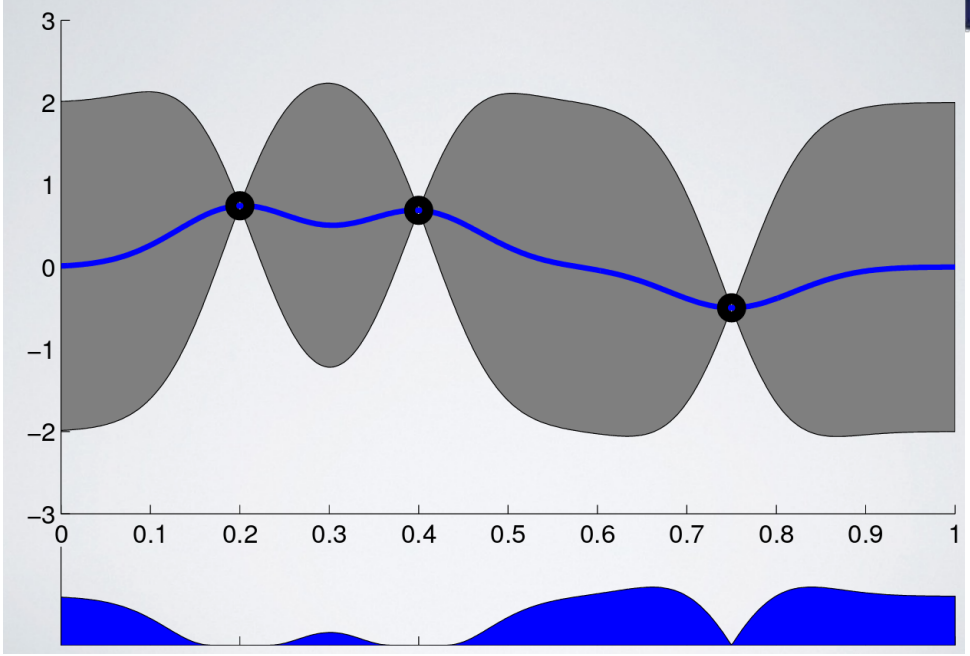
$$\gamma(x) = \frac{f(x_{\text{best}}) - \mu(x)}{\sigma(x)}$$

  - Expected Improvement (Mockus 1978)
  - GP Upper Confidence Bound (Srinivas et al. 2010):
    - $a_{LCB}(x) = \mu(x) - \kappa\sigma(x)$

Probability of Improvement

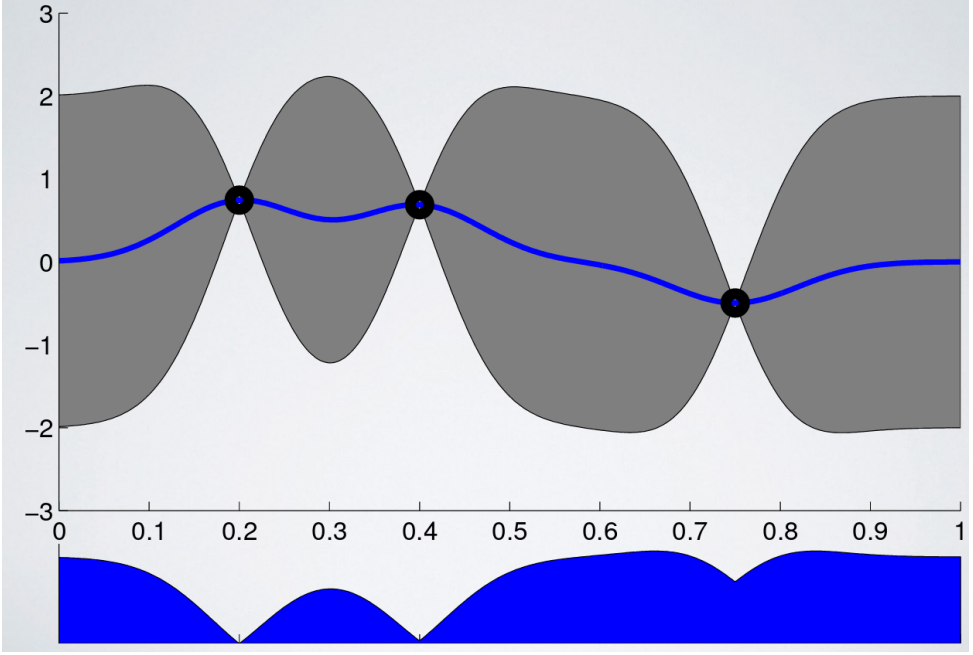


Expected Improvement



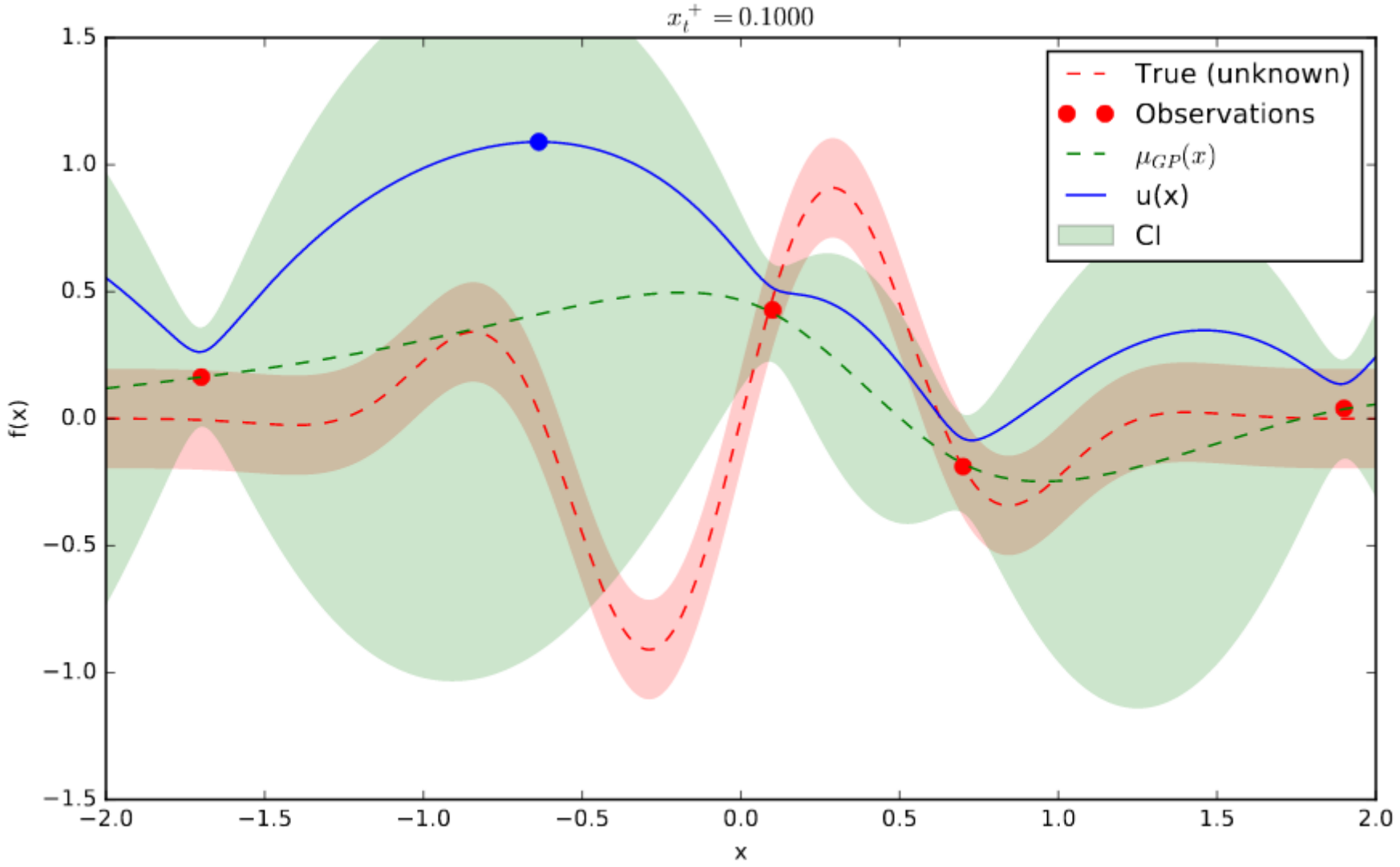
Te funkcje są dość podobne...

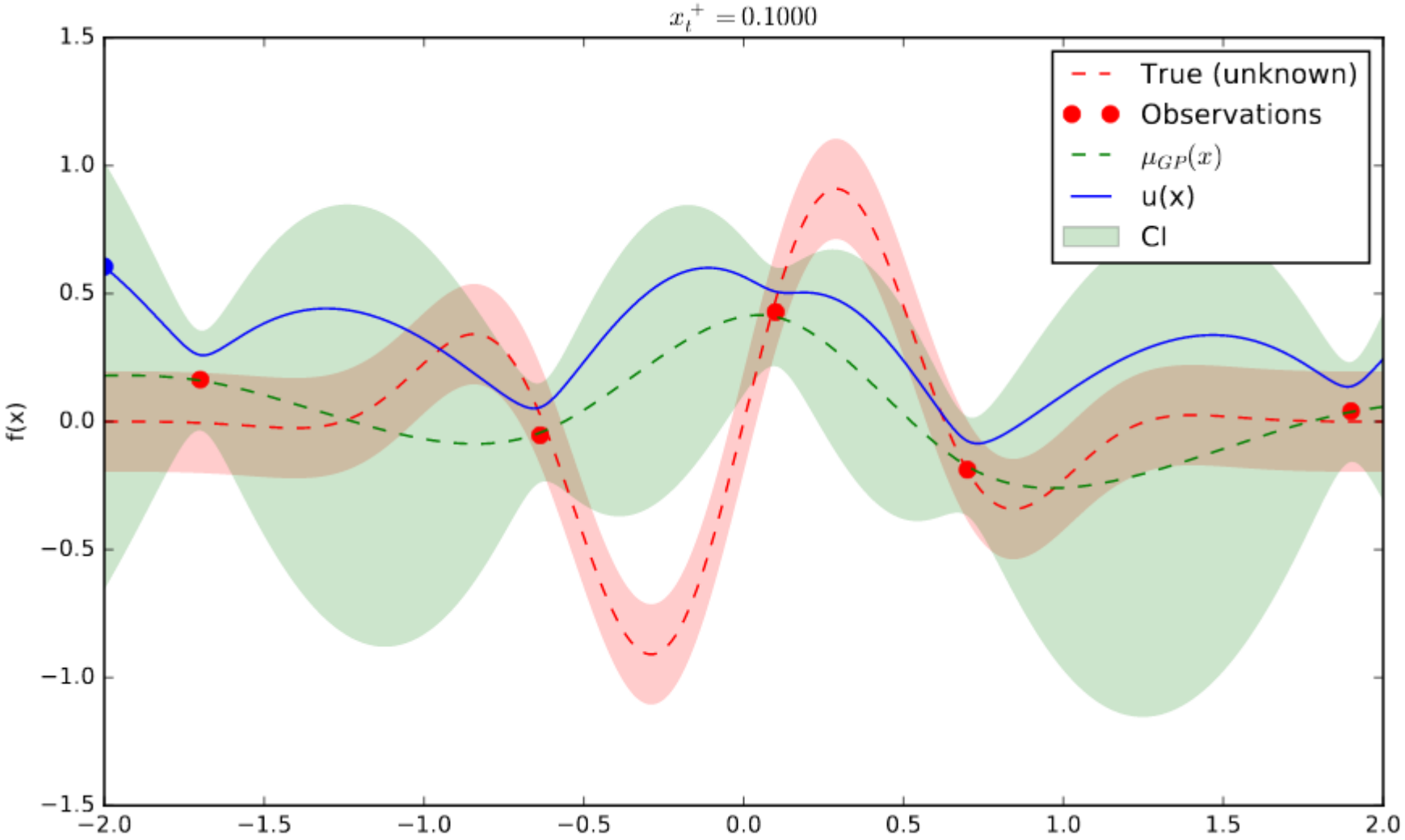
GP Upper (Lower) Confidence Bound



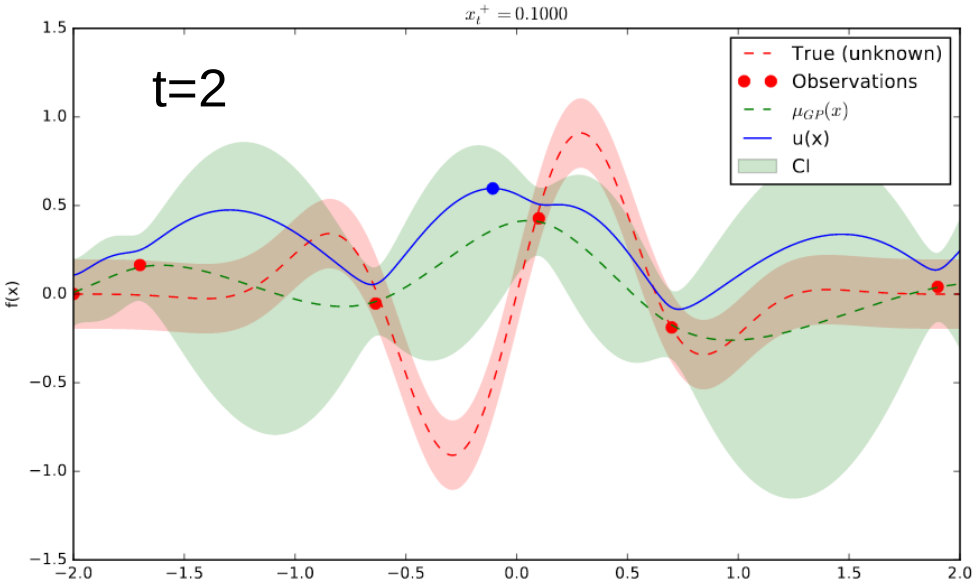
Wybieramy x próbkowania

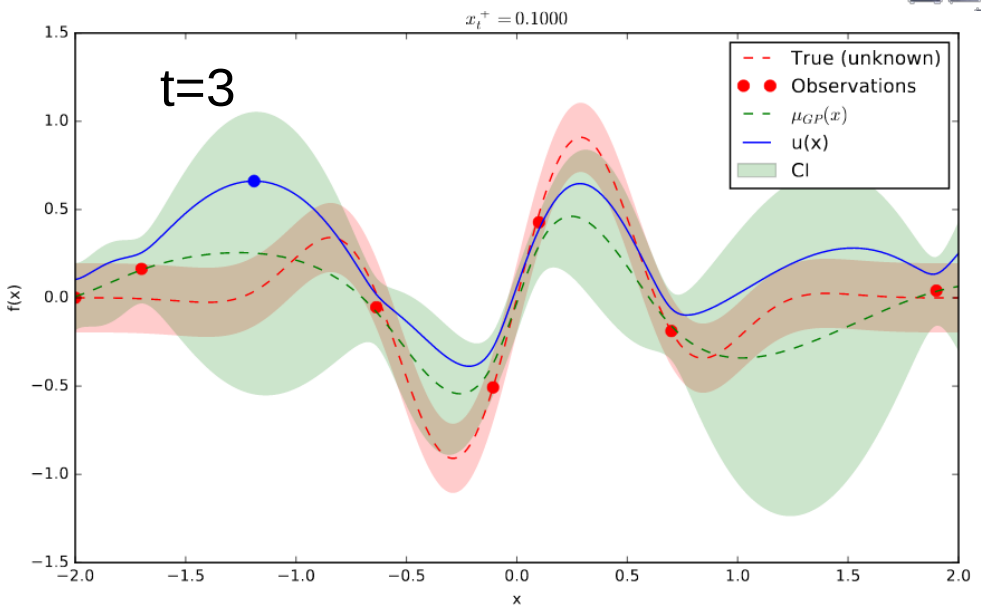
u(x) – funkcja akwizycji

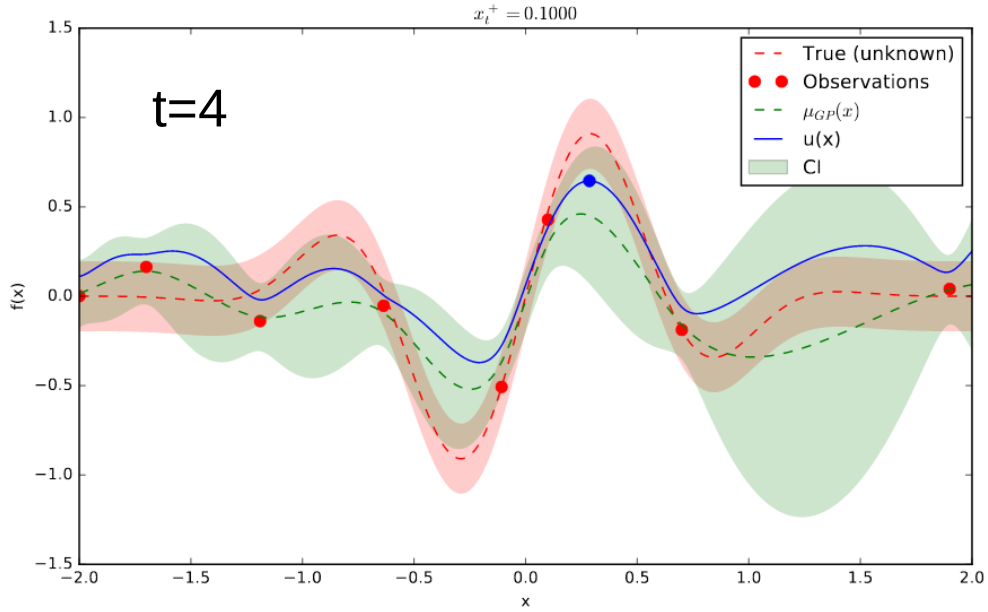


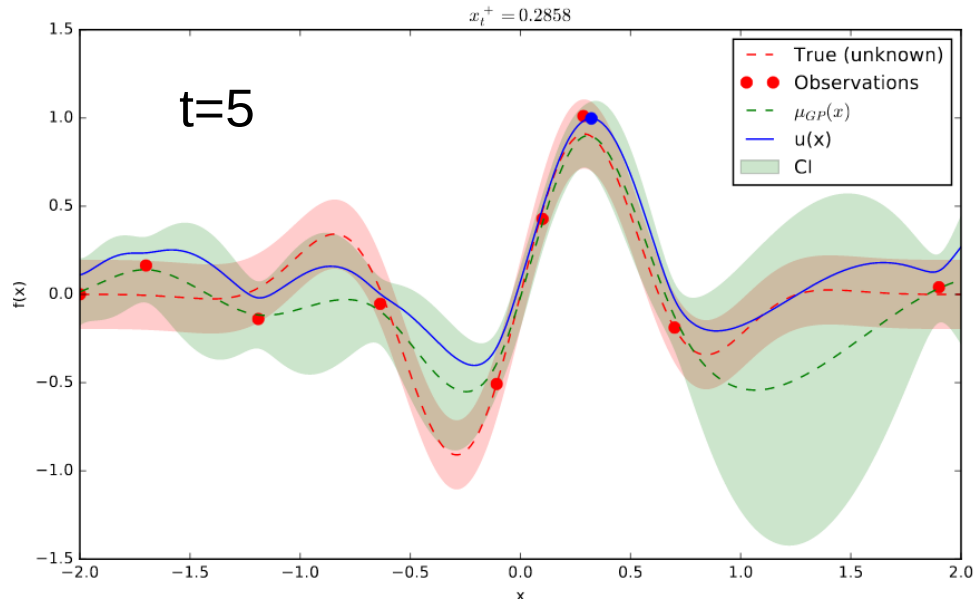


Dokonujemy próbkowania i powtarzamy procedurę...


$x_t^+ = 0.1000$
t=2
True (unknown)
Observations
$\mu_{GP}(x)$
u(x)
CI
f(x)


$x_t^+ = 0.1000$
t=3
True (unknown)
Observations
$\mu_{GP}(x)$
u(x)
CI
f(x)
x


$x_t^+ = 0.1000$
t=4
True (unknown)
Observations
$\mu_{GP}(x)$
u(x)
CI
f(x)
x


$x_t^+ = 0.2858$
t=5
True (unknown)
Observations
$\mu_{GP}(x)$
u(x)
CI
f(x)
x

Ograniczenia

- Optymalizacja Bayesowska zależy od wybranych parametrów
- Oraz wyboru funkcji akwizycji
- I wyboru prioru
- Jest sekwencyjna – kłopoty z paralelizacją.
- Istnieją alternatywne metody, które można zrównoleglić (np. Random Search czy Tree of Parzen Estimators (TPE) używane przez pakiet hyperopt).

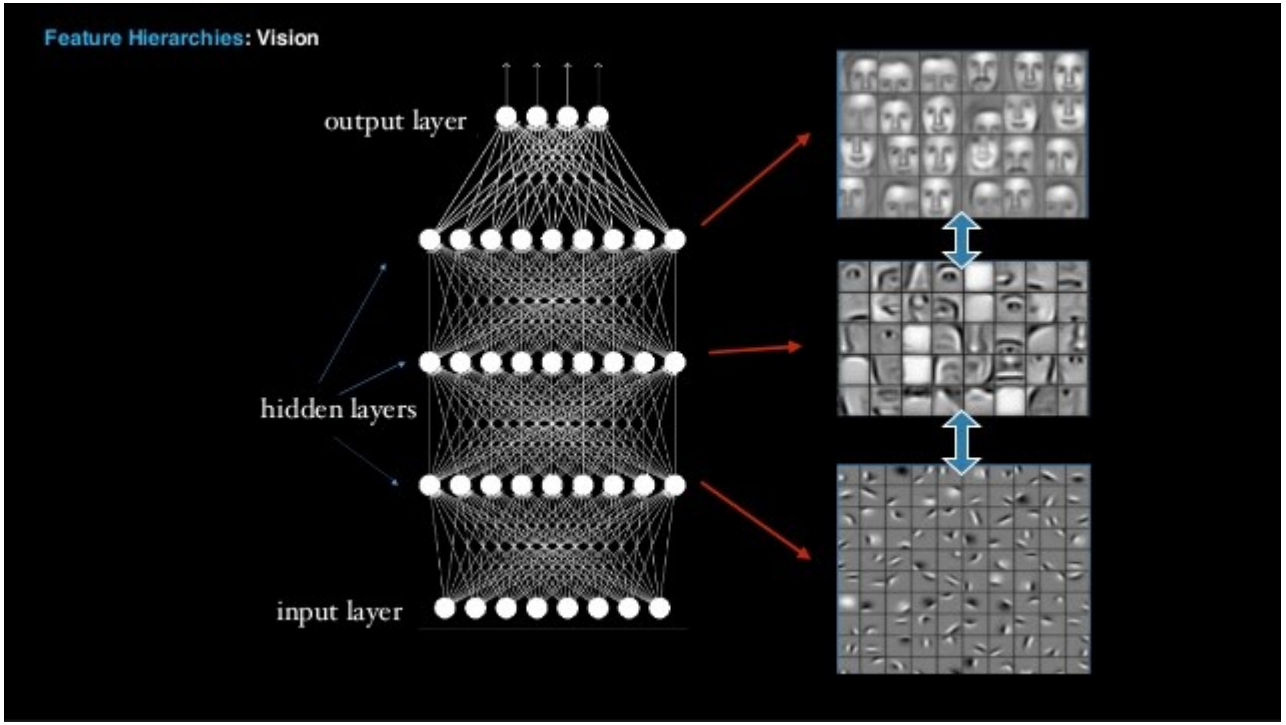
Implementacje

- **Python**
  - Spearmint https://github.com/JasperSnoek/spearmint
  - GPyOpt https://github.com/SheffieldML/GPyOpt
  - RoBO https://github.com/automl/RoBO
  - Scikit-optimize https://github.com/MechCoder/scikit-optimize
- **C++**
  - MOE https://github.com/yelp/MOE

Literatura

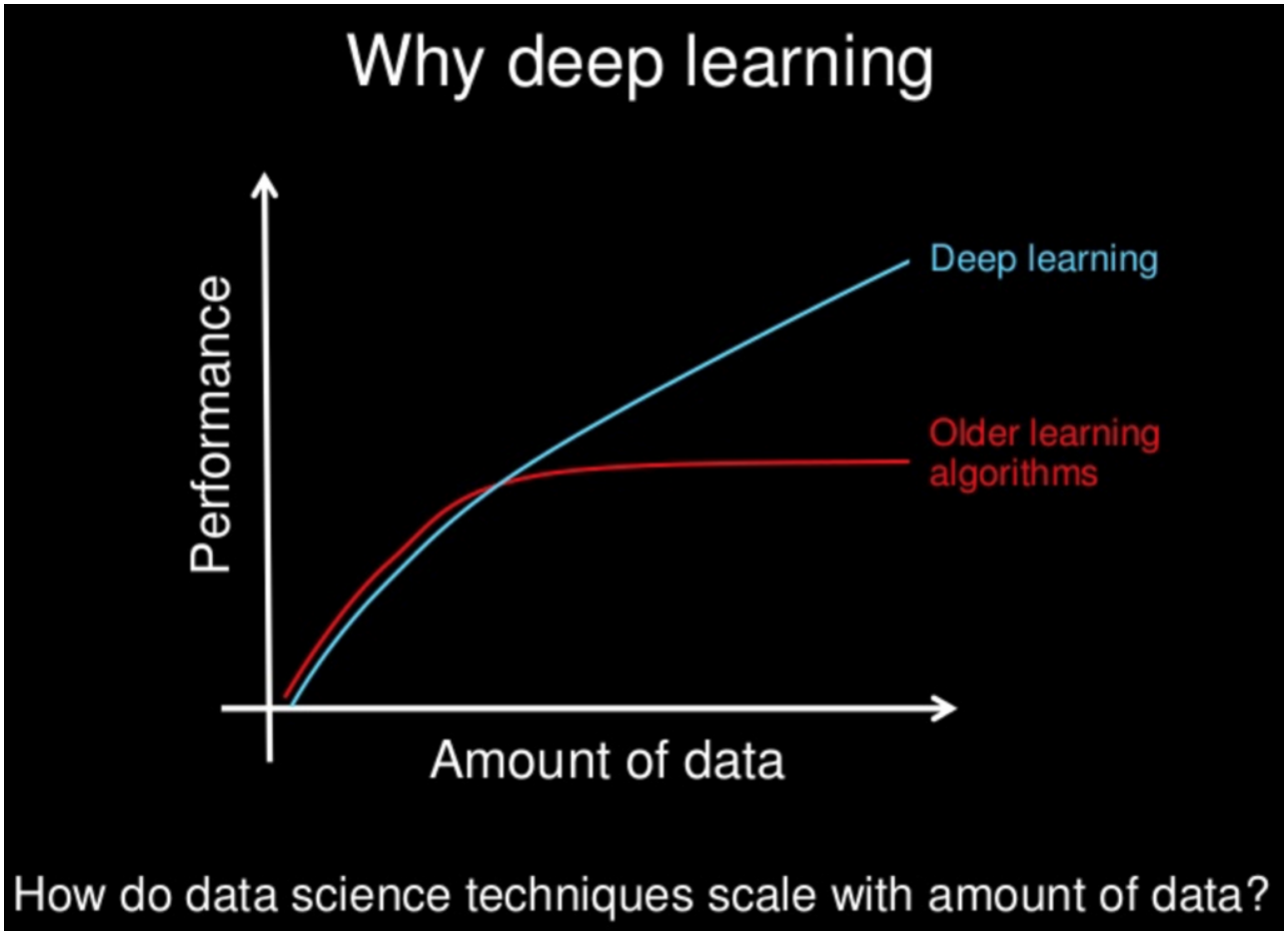
- Brochu, E., Cora, V. M., and De Freitas, N. (2010). A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv preprint arXiv:1012.2599.
- Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and de Freitas, N. (2016). Taking the human out of the loop: A review of bayesian optimization. Proceedings of the IEEE, 104(1):148–175.

Deep Learning



1. Co oznacza “deep learning”?

2. Dlaczego daje lepsze rezultaty niż inne metody przy rozpoznawaniu obrazu, mowy i nie tylko?



Odpowiedź:

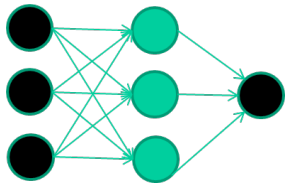
‘Deep Learning’ - użycie sieci neuronowej z wieloma warstwami ukrytymi

Seria warstw ukrytych dokonuje dokonuje identyfikacji cech obiektów (feature identification) I przetwarza je w ciągu operacji: identyfikacja cech→ dalsza identyfikacja → selekcja

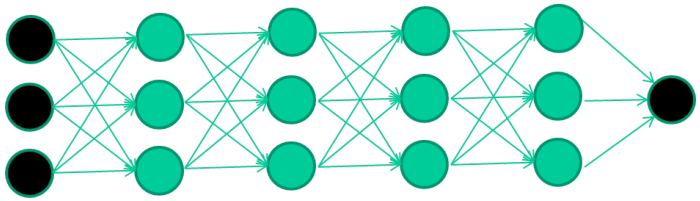
Ale przecież sieci neuronowe znane są od lat 80-tych???

Zawsze mieliśmy dobre algorytmy do uczenia sieci z jedną (ew. dwoma) warstwami ukrytymi.

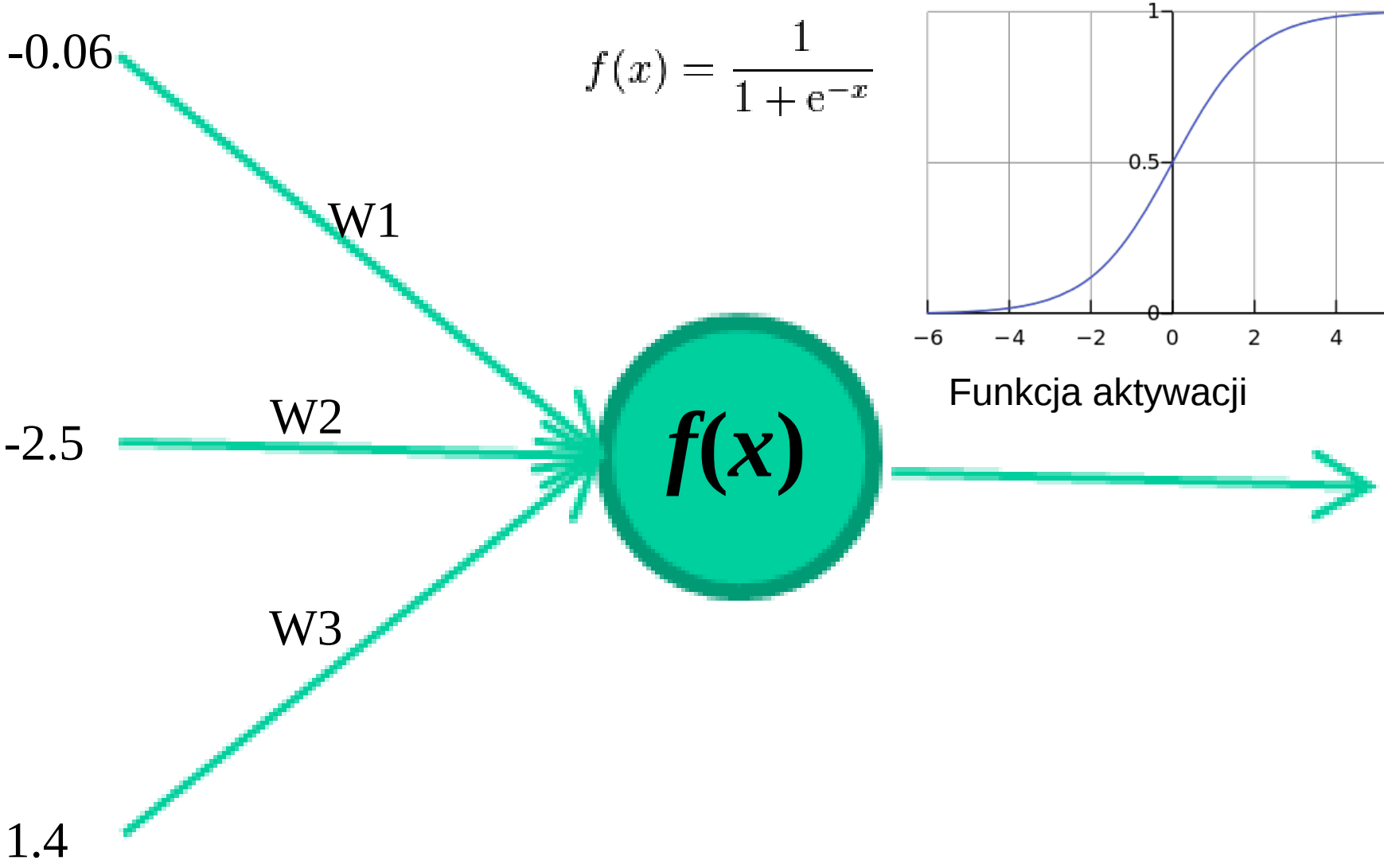
Ale one nie działały przy treningu większej liczby warstw



nowość: algorytmy do treninging sieci z wieloma warstwami!

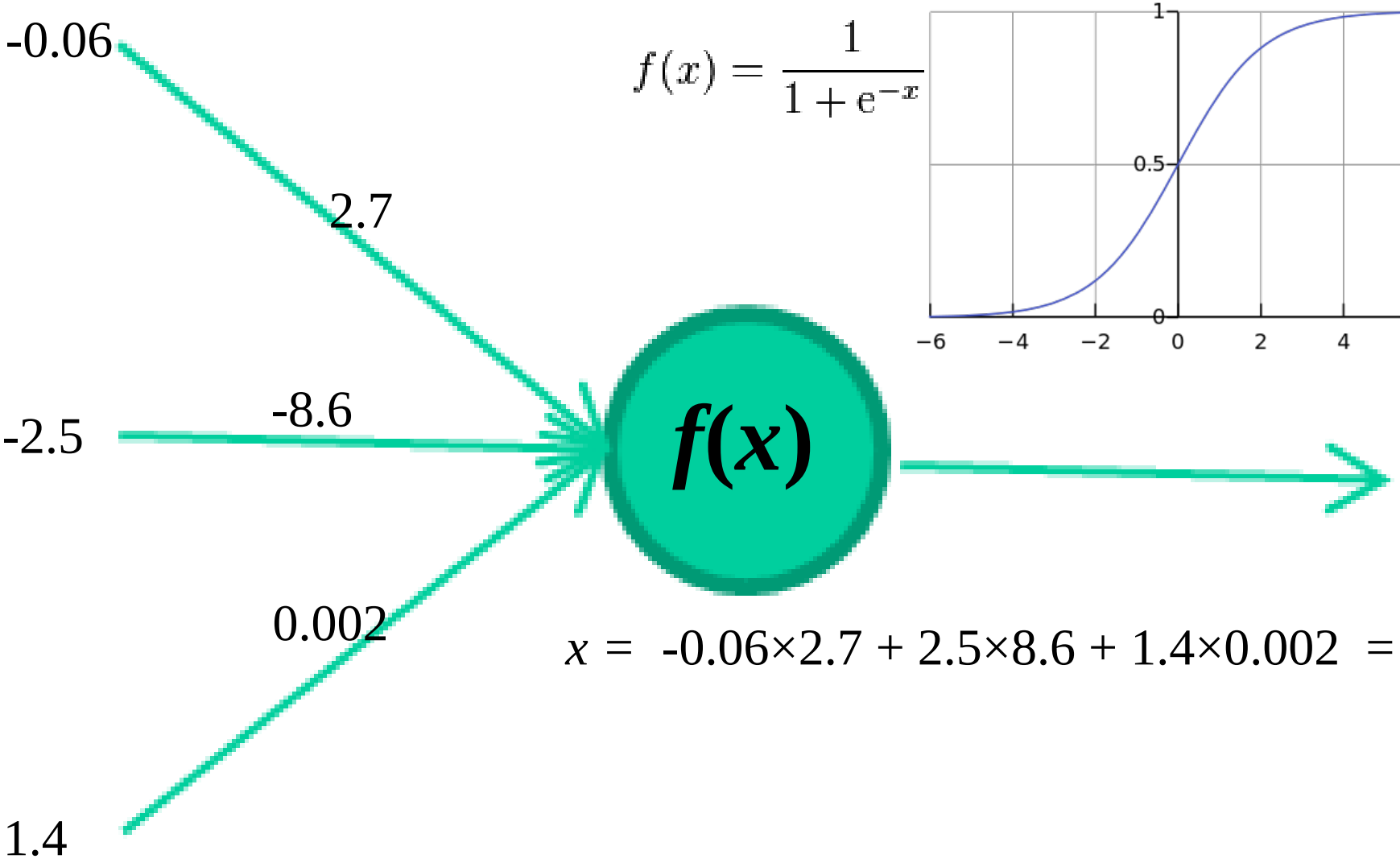


Jak uczymy sieć neuronową?



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

Funkcja aktywacji



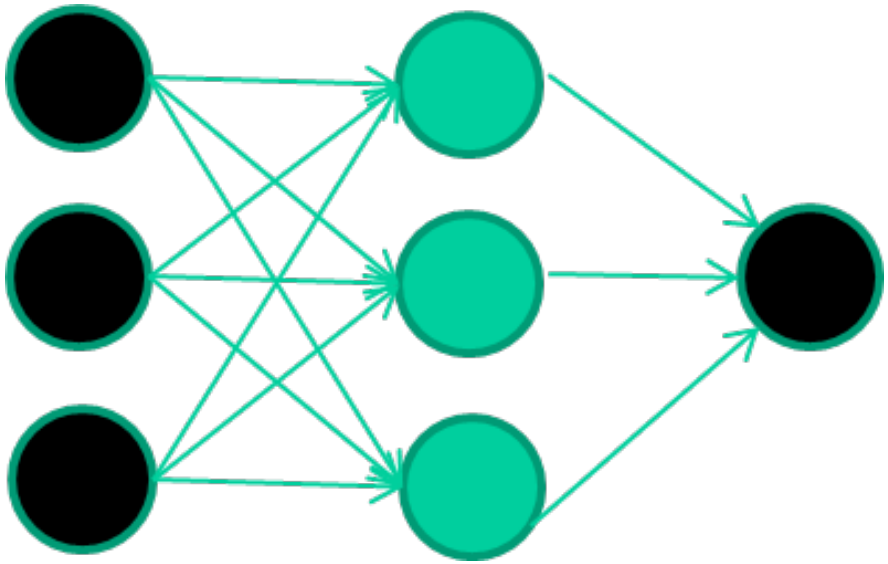
$$f(x) = \frac{1}{1 + e^{-x}}$$

$x = -0.06 \times 2.7 + 2.5 \times 8.6 + 1.4 \times 0.002 = 21.34$

Trening sieci neuronowej

Pola			***Klasa***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc …

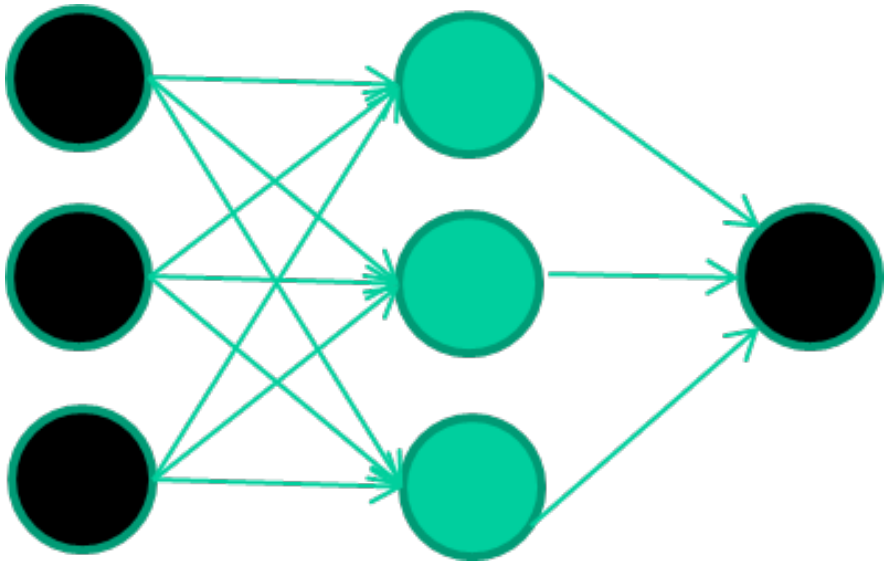


Dane treningowe

Pola			***Klasa***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc …

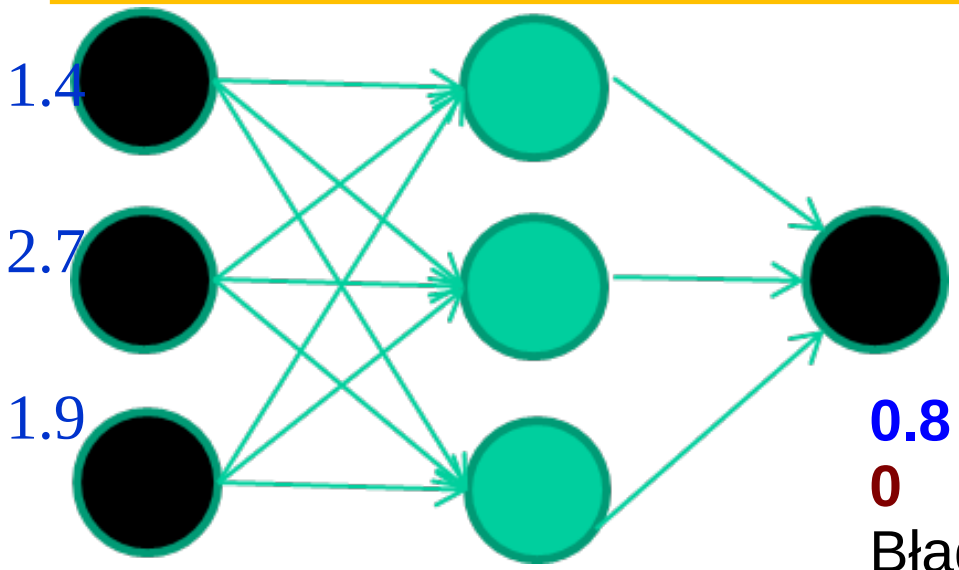
Inicjalizacja przypadkowymi wagami



Dane treningowe

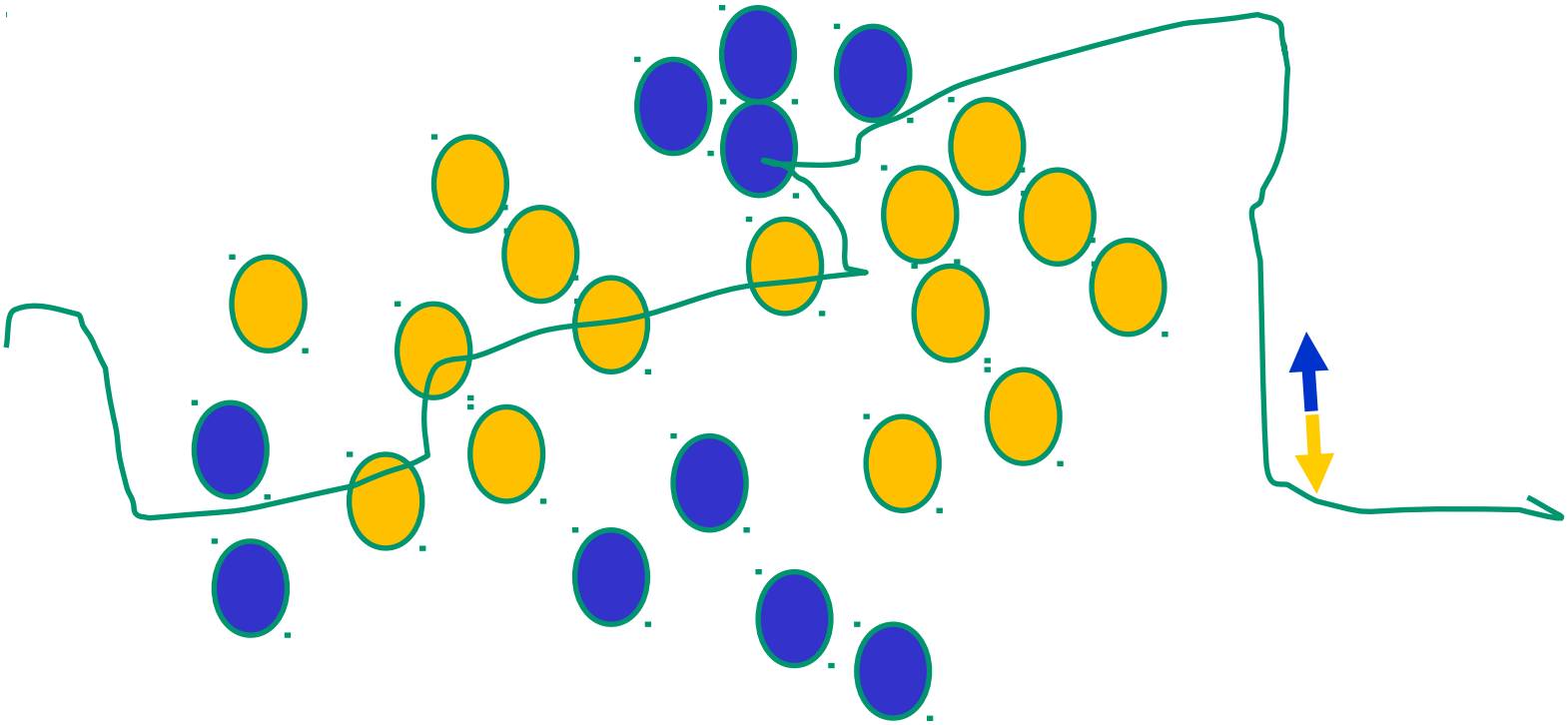
Pola			***Klasa***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

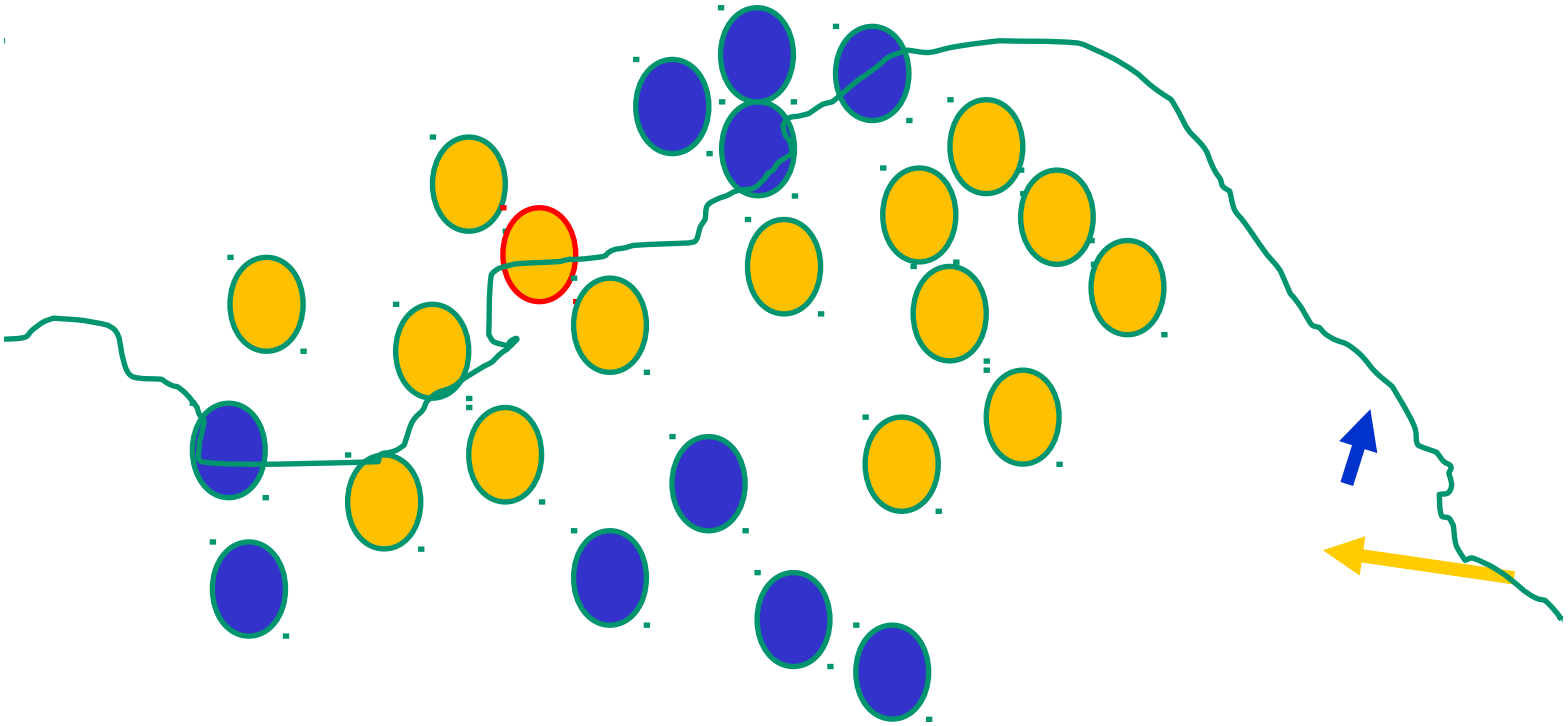
Wczytujemy dane
Przepuszczamy przez sieć
Wynik porównujemy z prawdziwym rezultatem

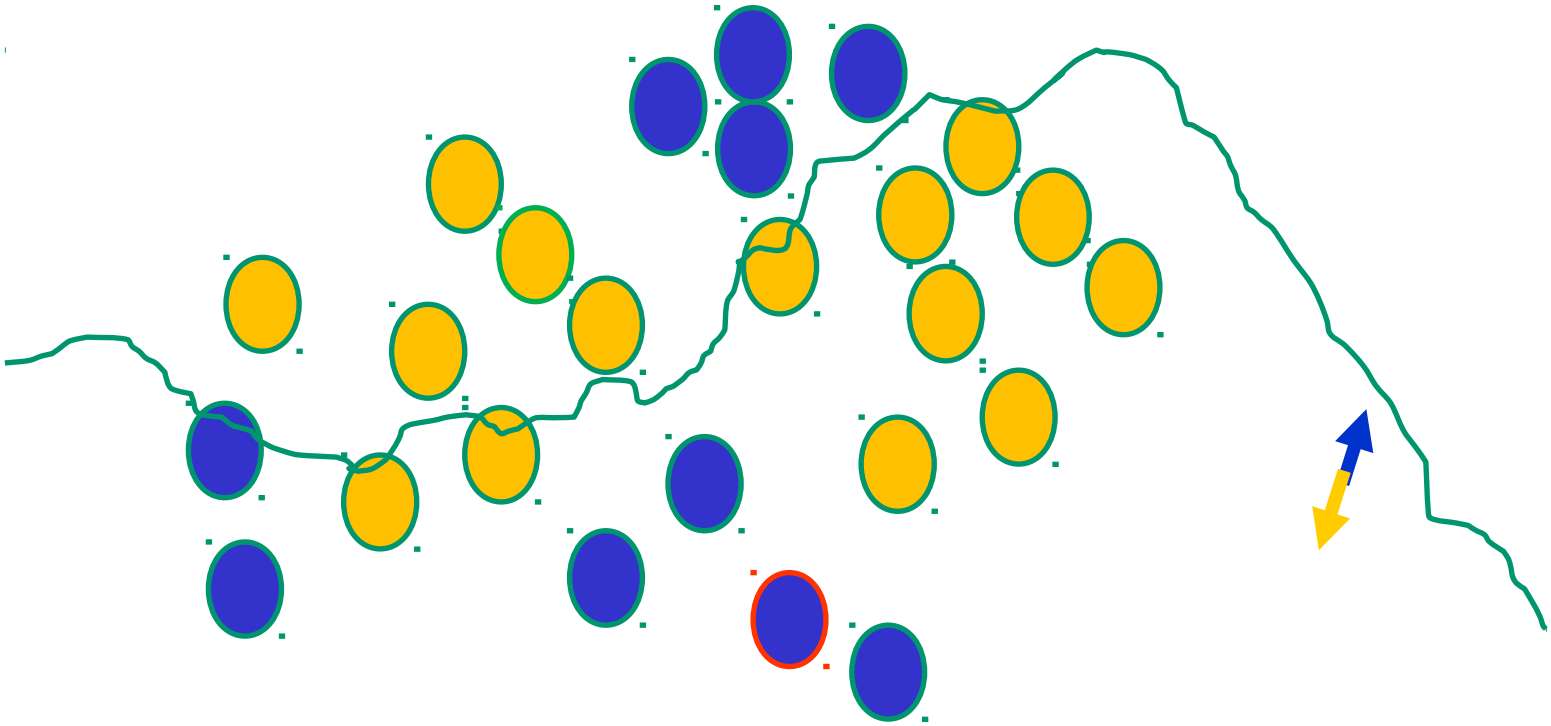


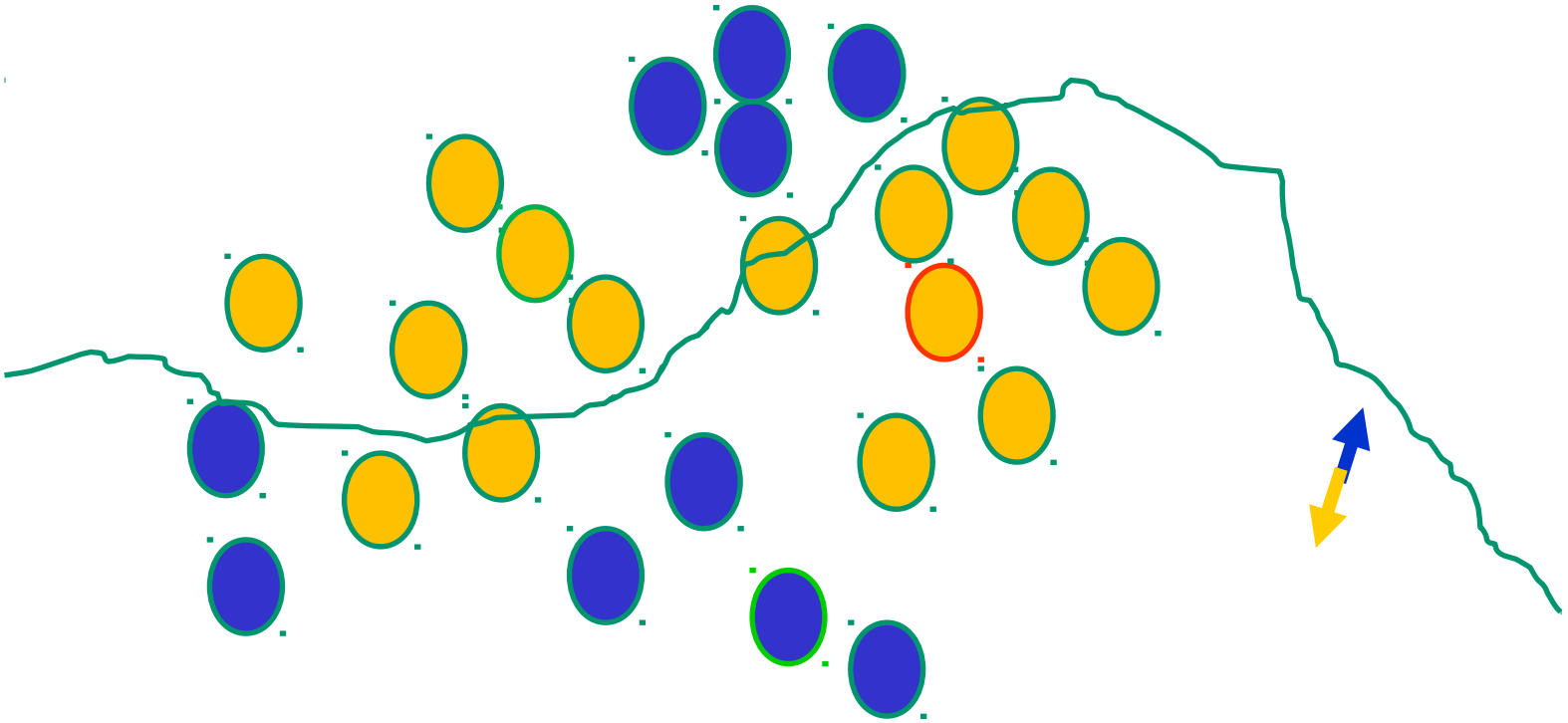
Modyfikujemy wagi na podstawie uzyskanego błędu

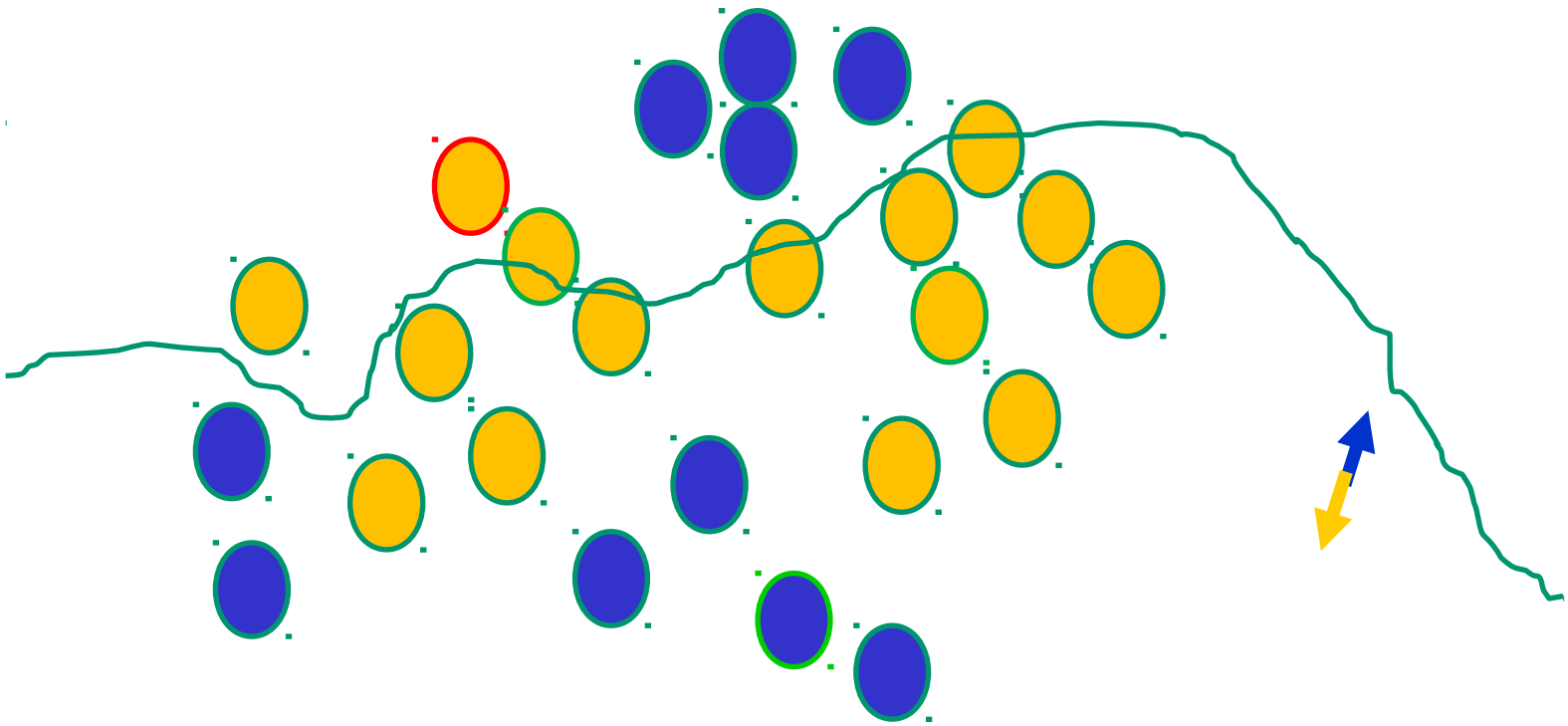
Powtarzamy miliony razy za każdym modyfikując wagi
Algorytmy uczenia zadbają, aby błąd się zmniejszał w kolejnych iteracjach..

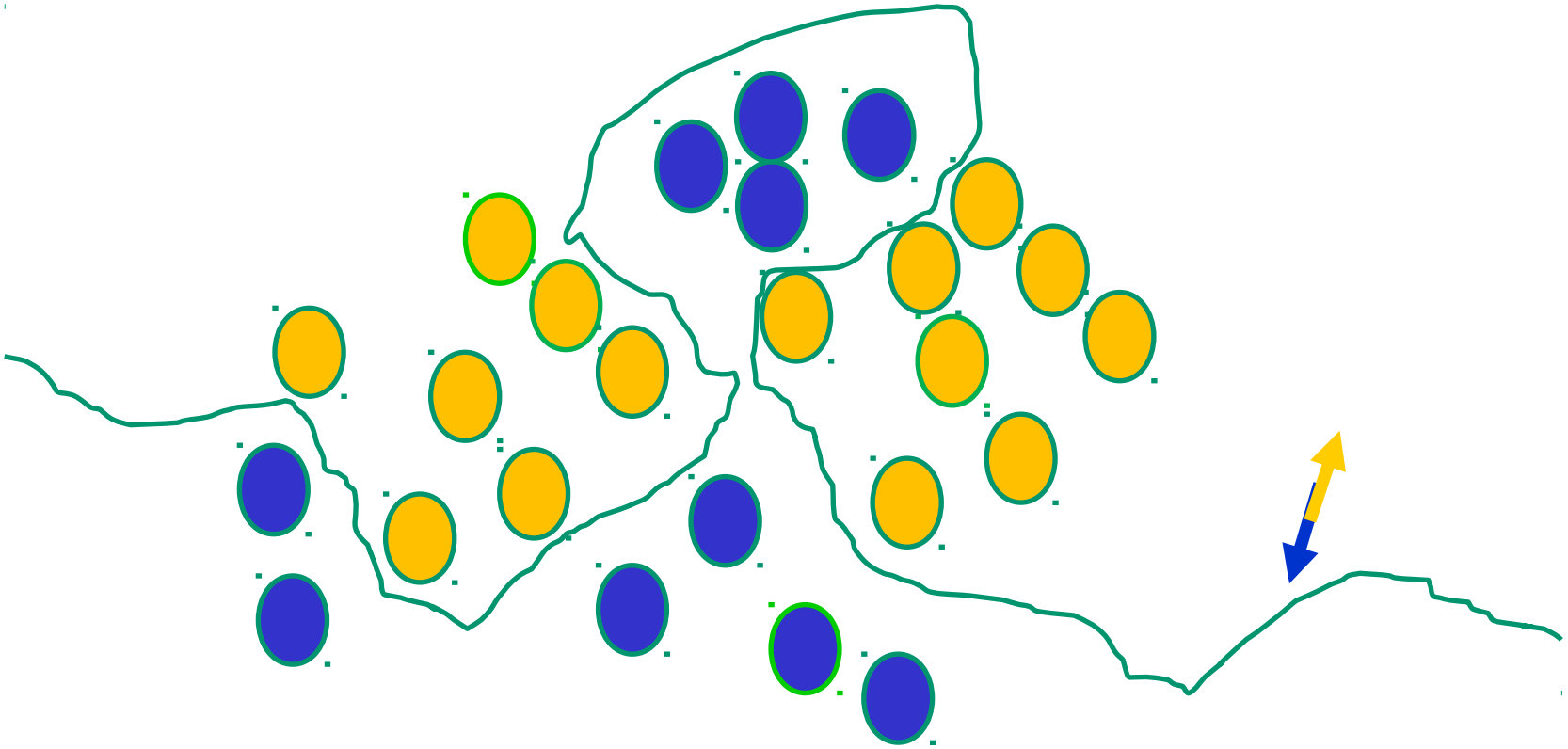












Uwaga

- Jeśli f(x) jest funkcją nieliniową, to sieć z jedną warstwą ukrytą może klasyfikować dowolny problem (odtwarza dowolną funkcję).

- Istnieje zbiór wag, który to zapewnia. Niestety problemem jest znalezienie go.

Jak zidentyfikować cechy (features)?

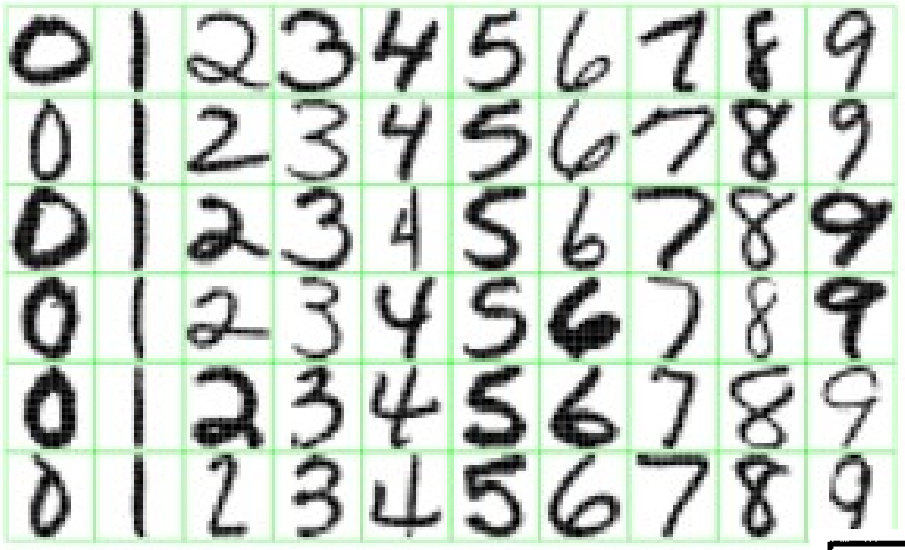
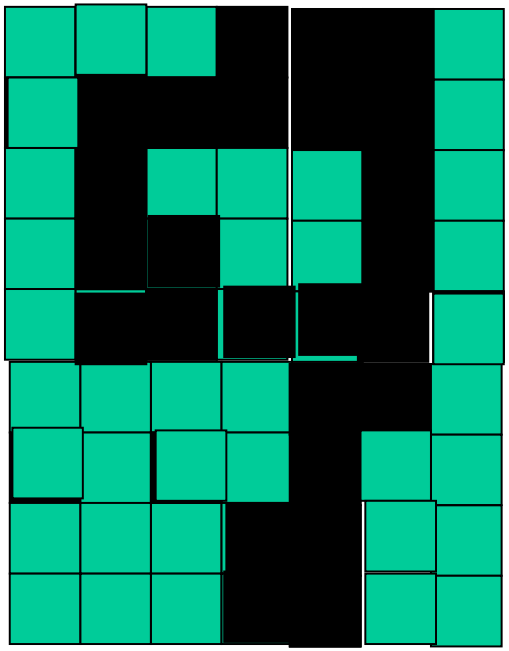
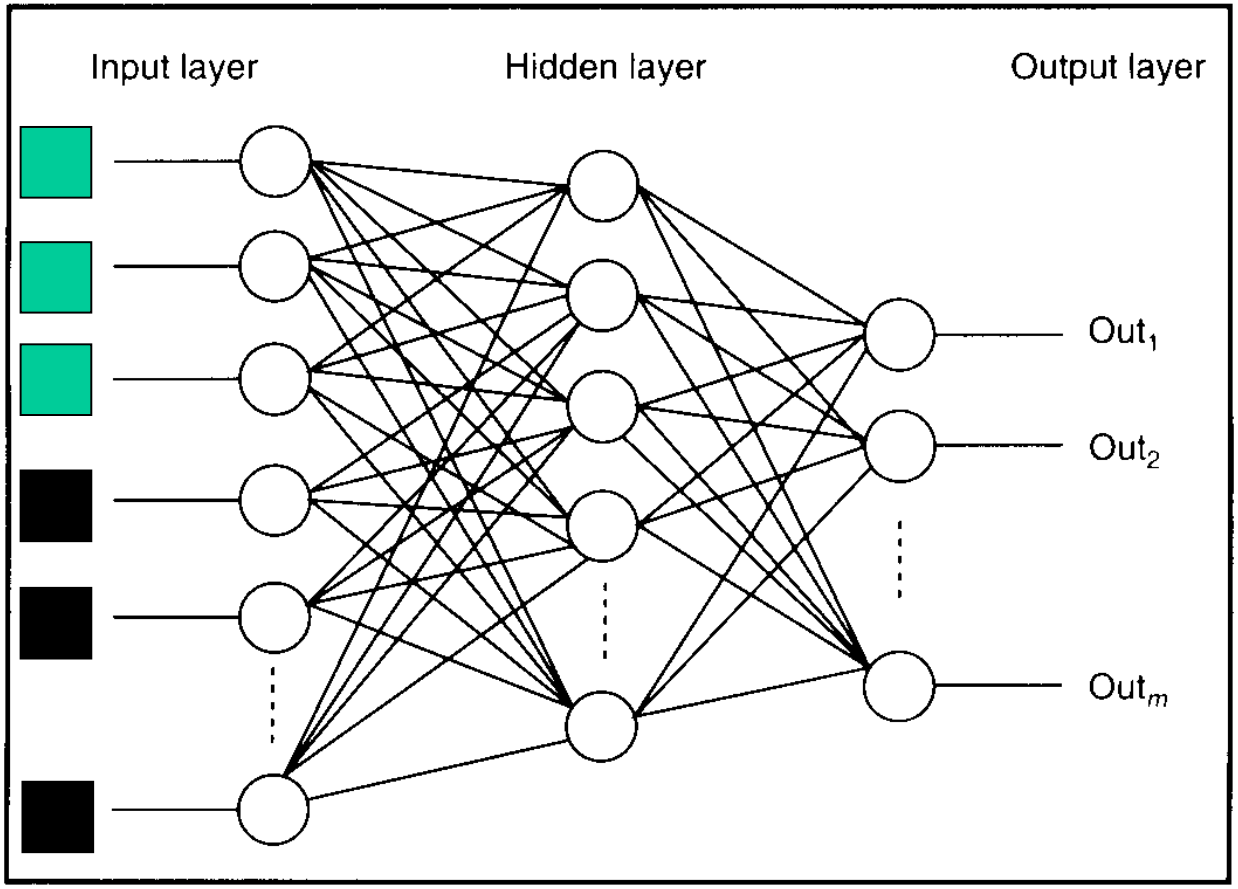


Figure 1.2: *Examples of handwritten digits from postal envelopes.*





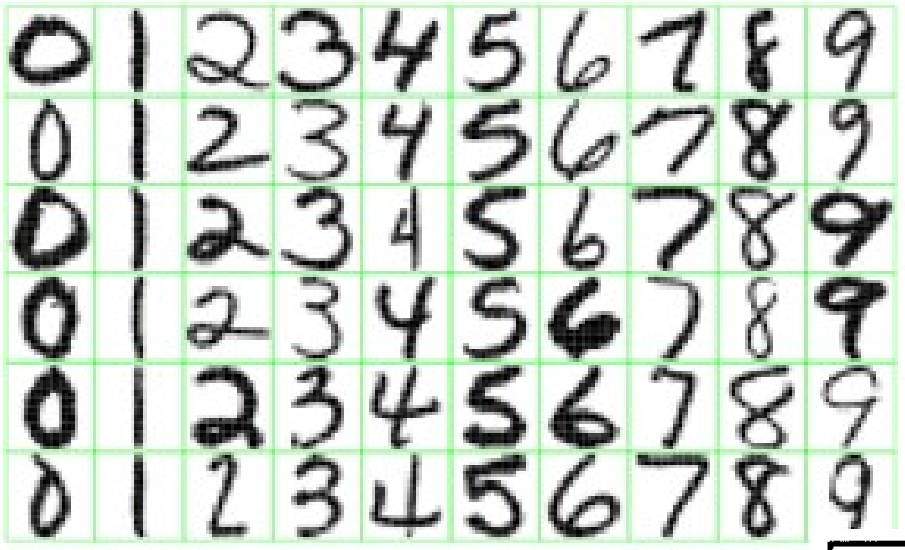
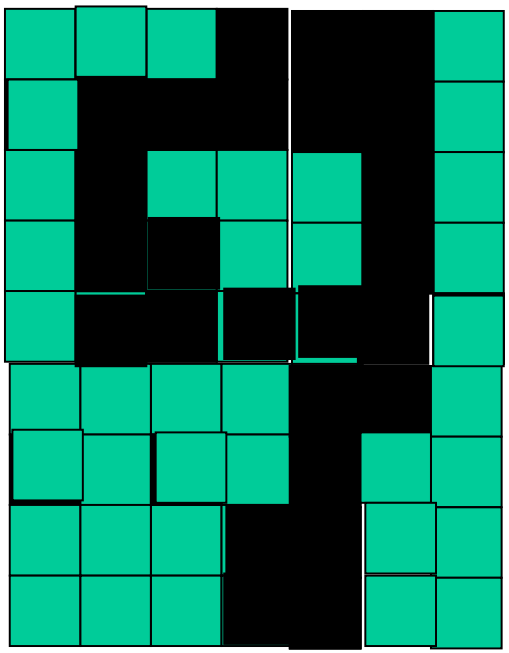
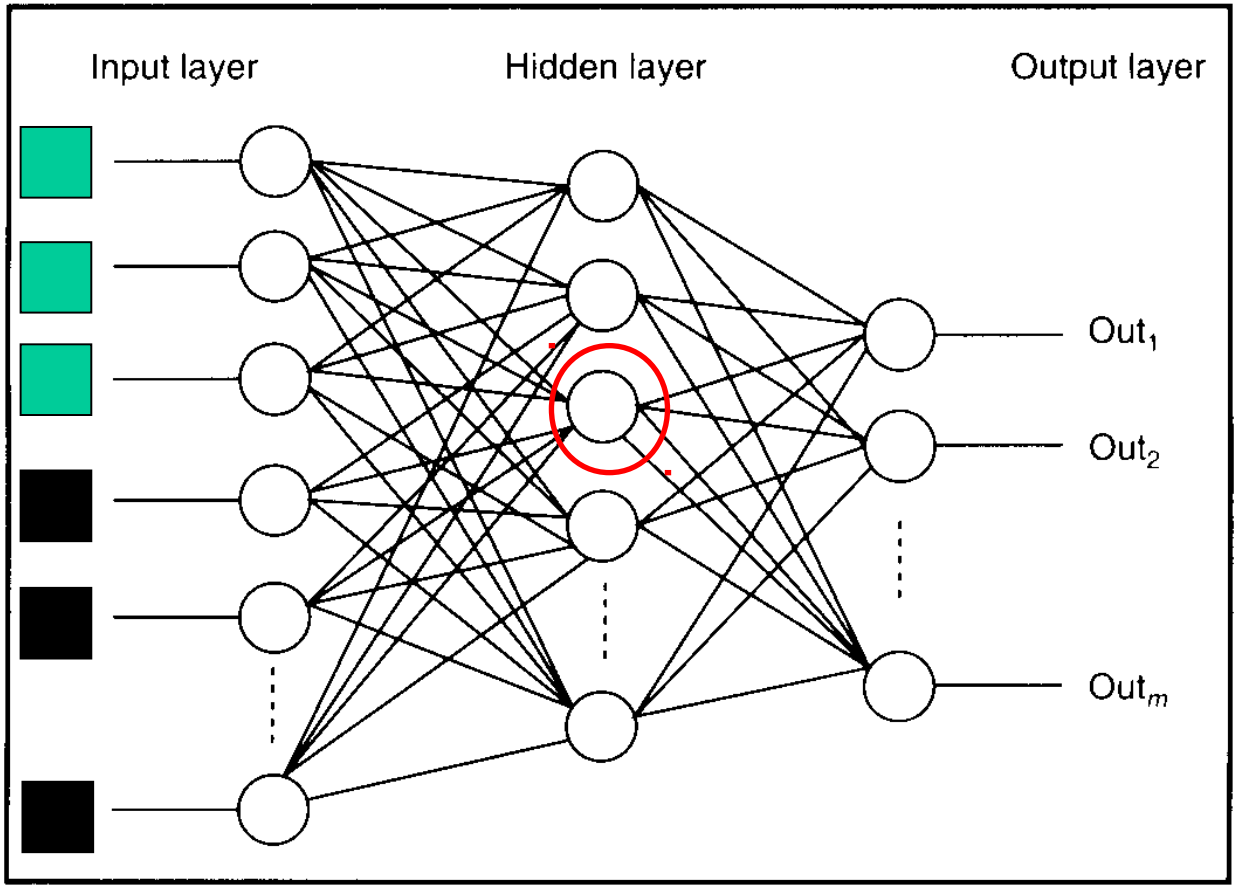


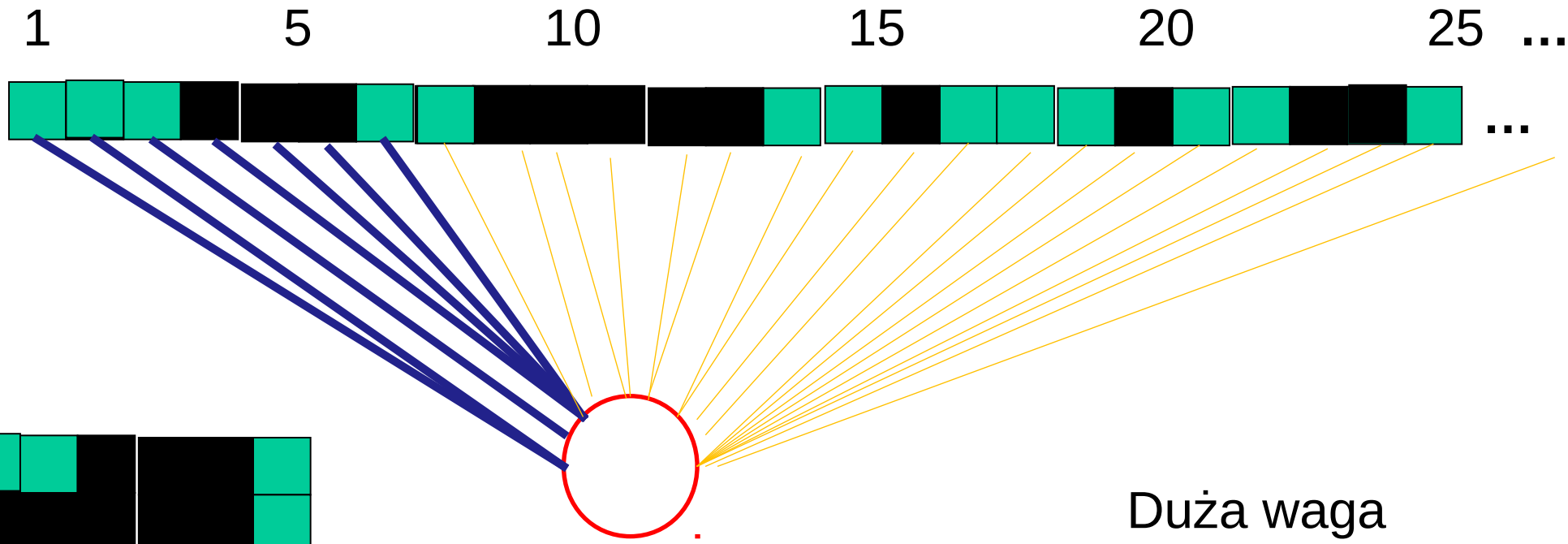
Figure 1.2: *Examples of handwritten digits from postal envelopes.*

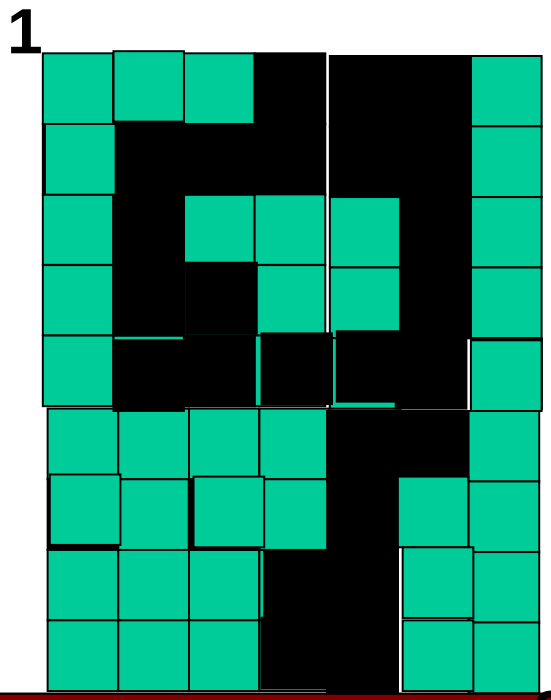


Co robi ten neuron?



Neurony w warstwach ukrytych są samoorganizującymi się detektorami cech

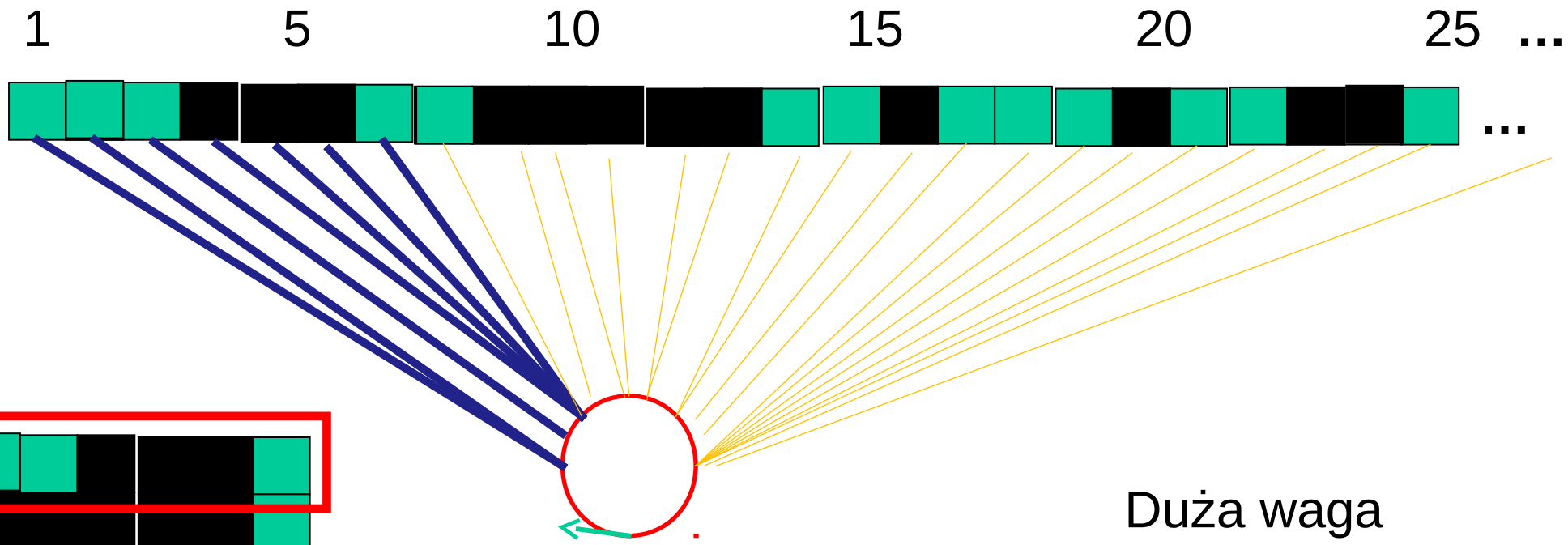


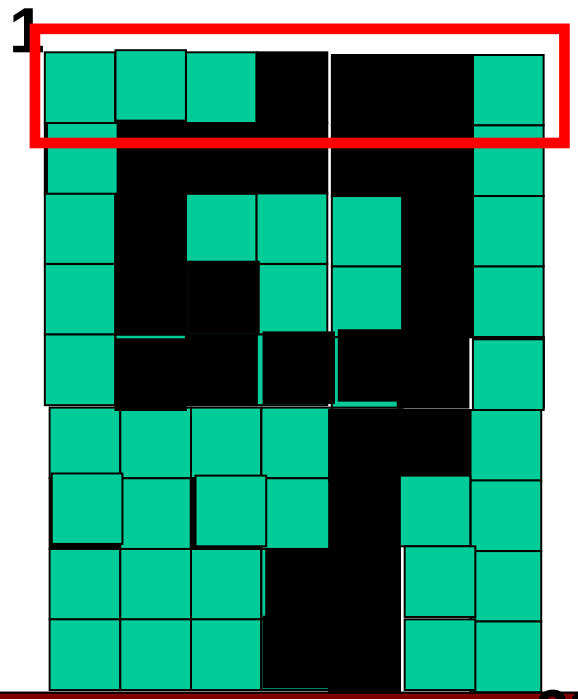


Duża waga

Mała waga

Co on wykrywa?



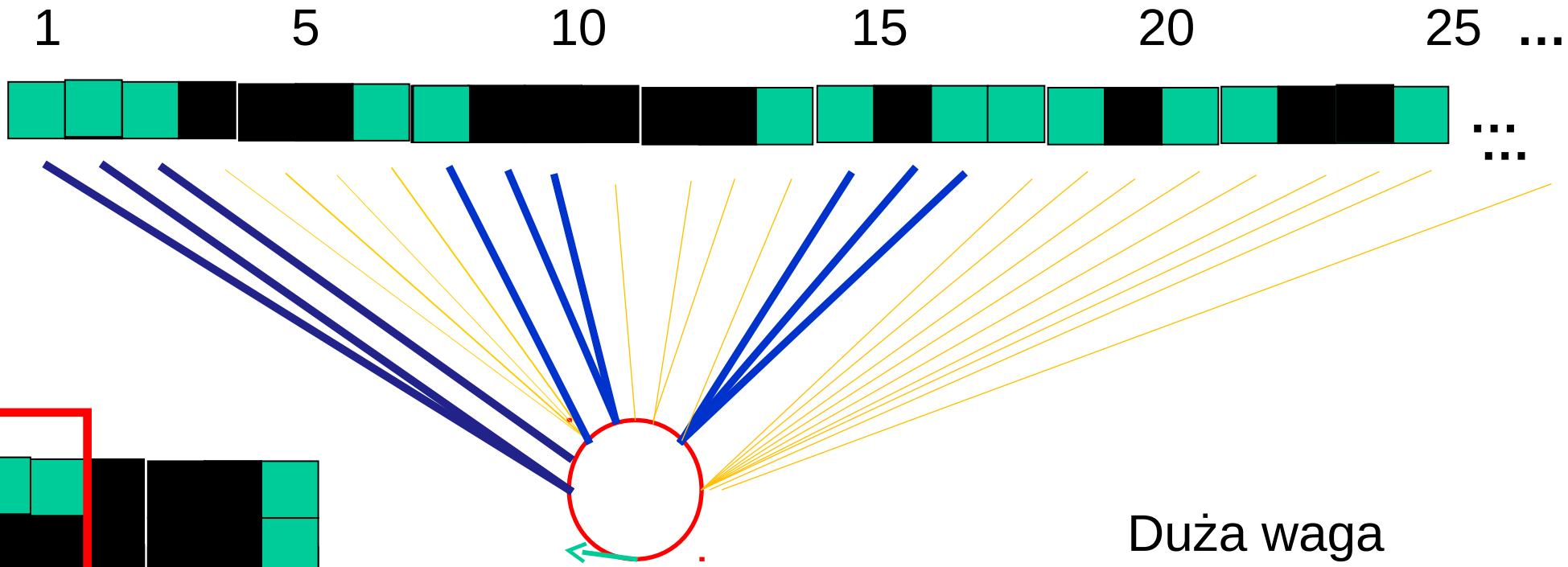


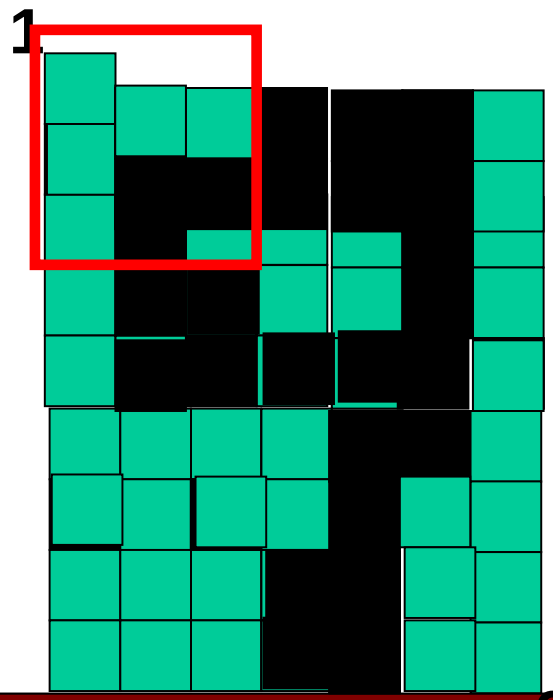
Duża waga

Mała waga

Wysyła silny sygnał, jeśli znajdzie poziomą linię w górnym rzędzie. Ignoruje wszstko inne.

Co on wykrywa?





Duża waga

Mała waga

Wysyła silny sygnał, jeśli znajdzie ciemny obszar w lewym górnym rogu.

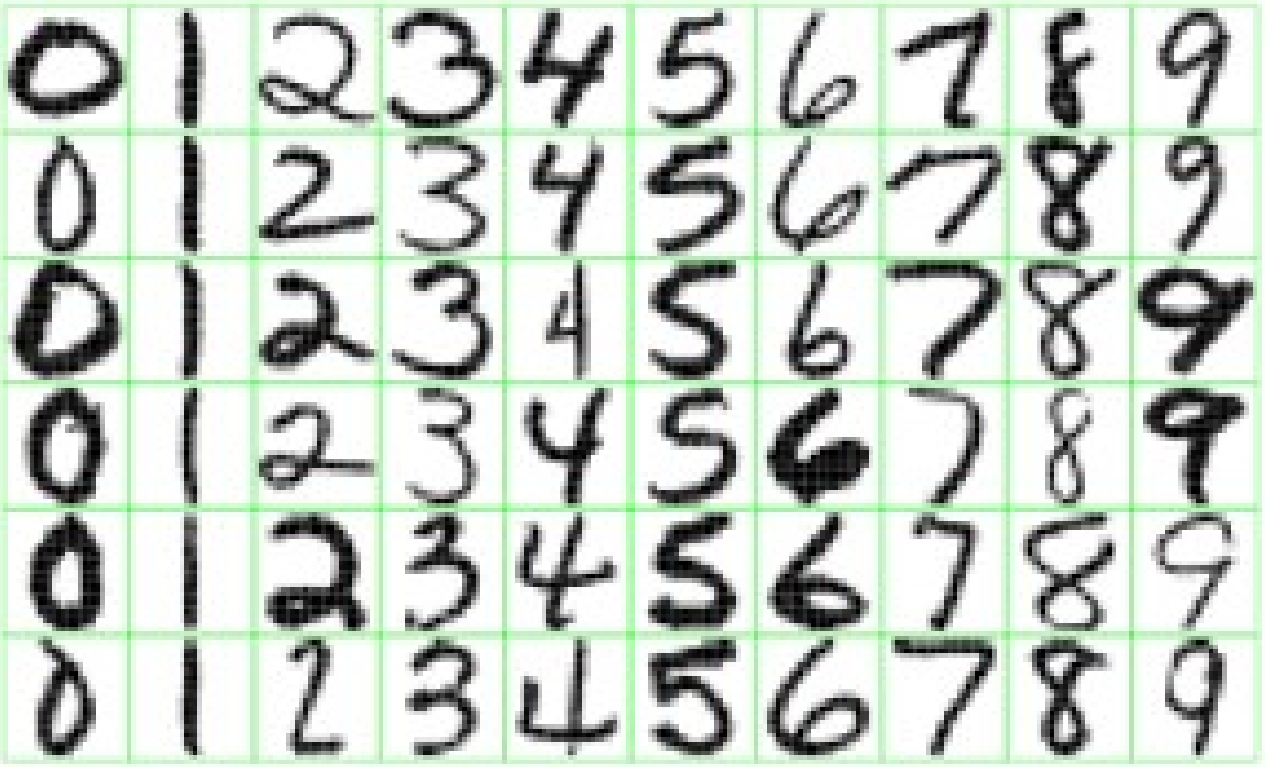


Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*

Jakie cechy powinna wykrywać sieć neuronowa czytająca pismo ręczne?

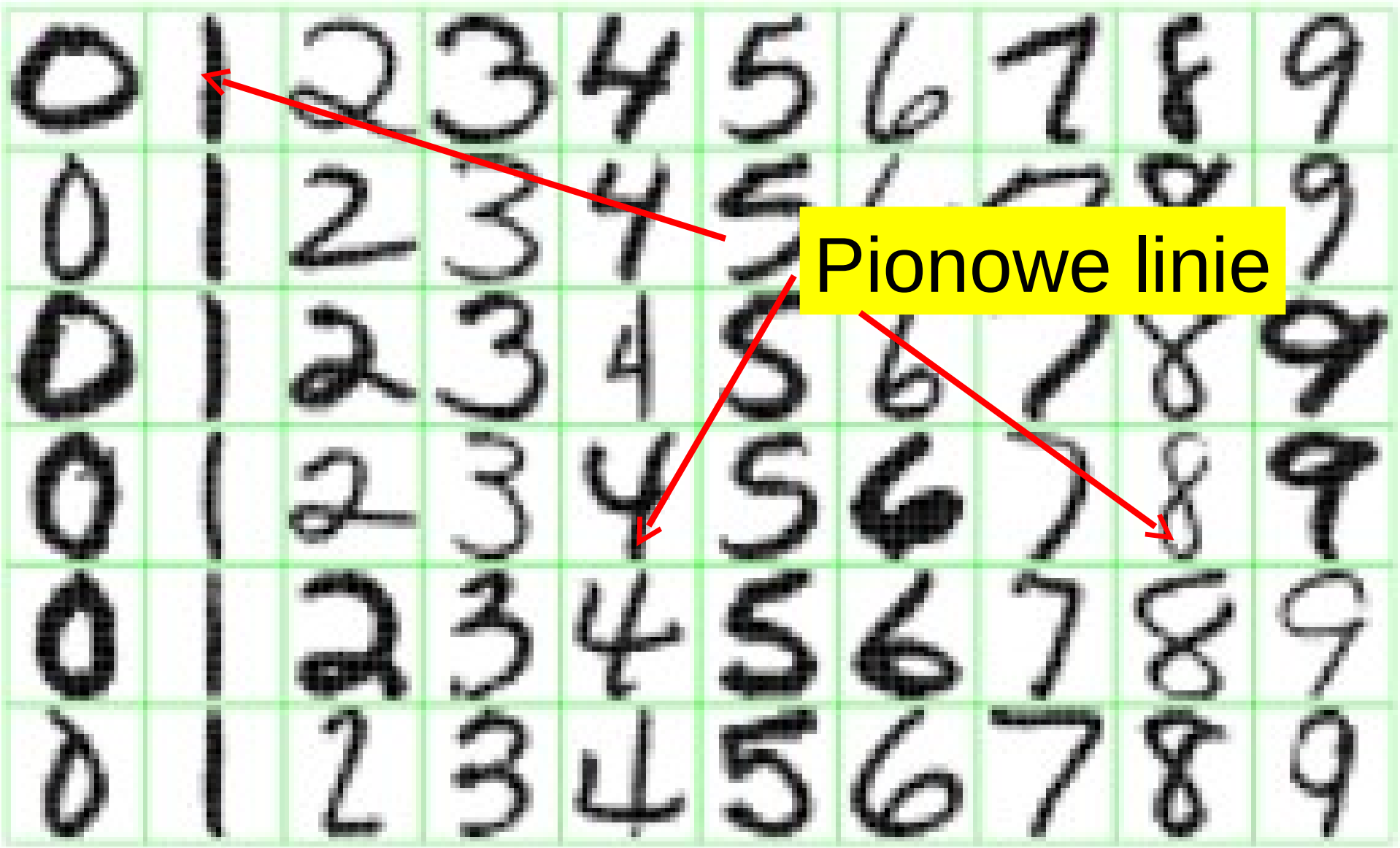


Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*

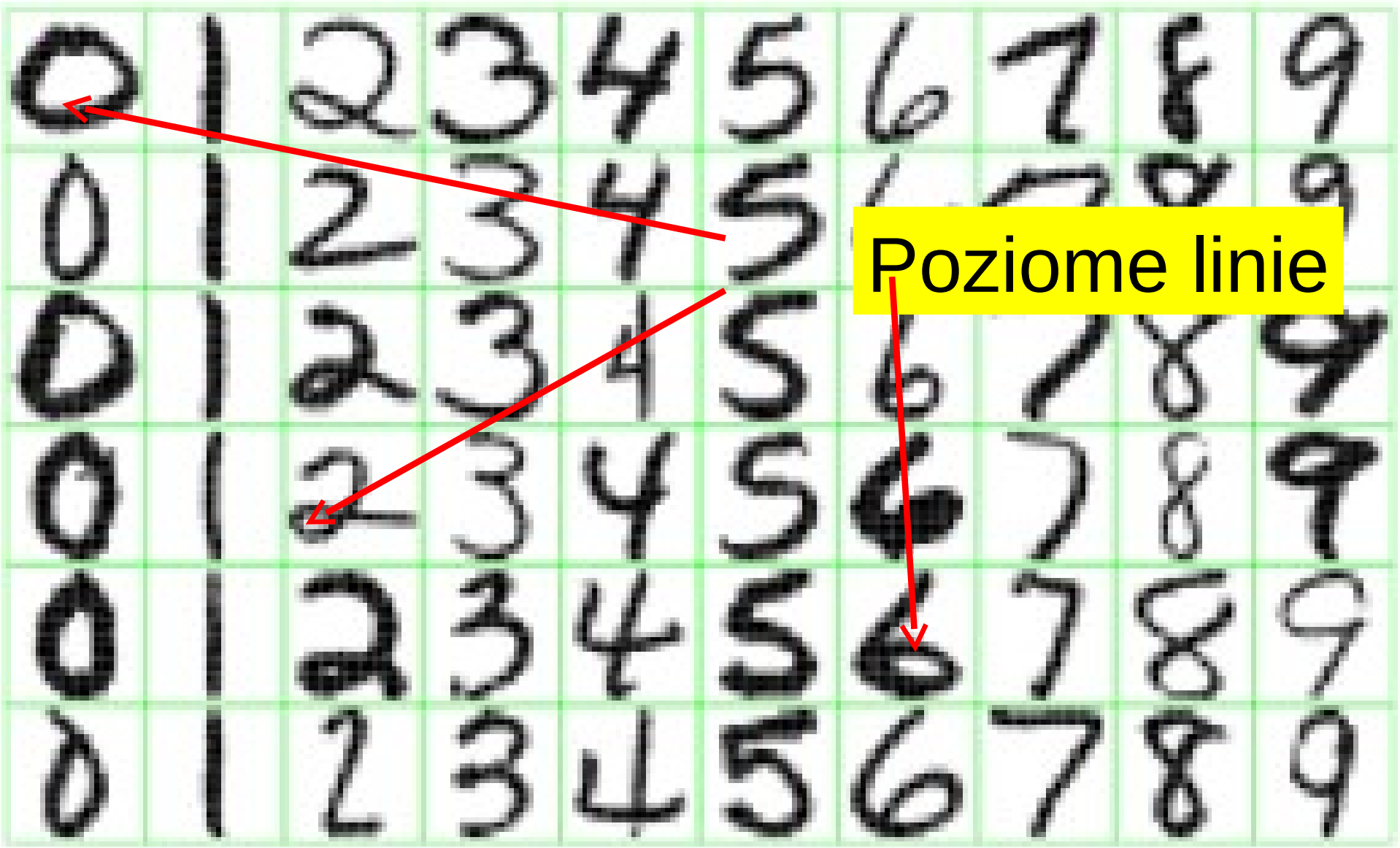
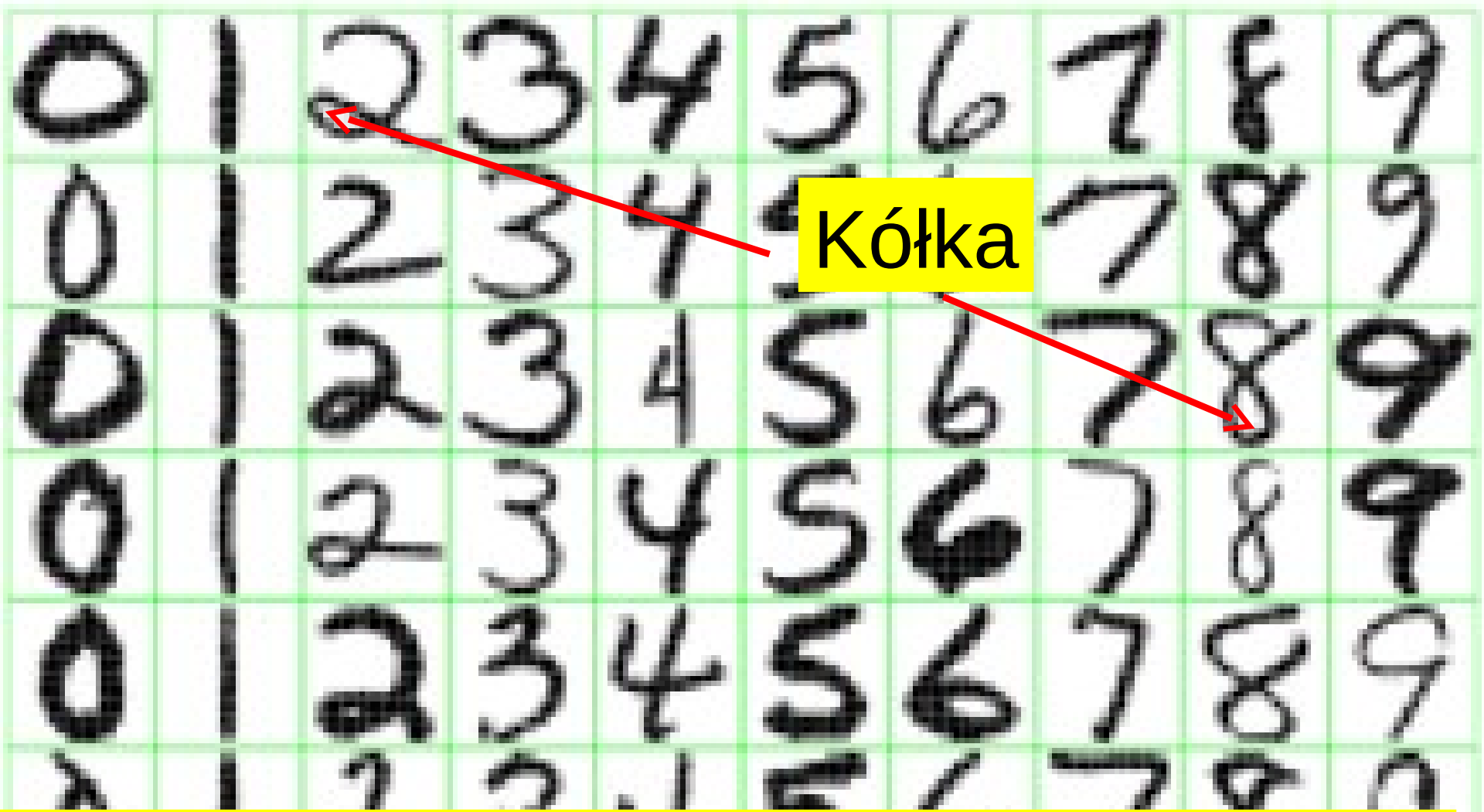


Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*

1

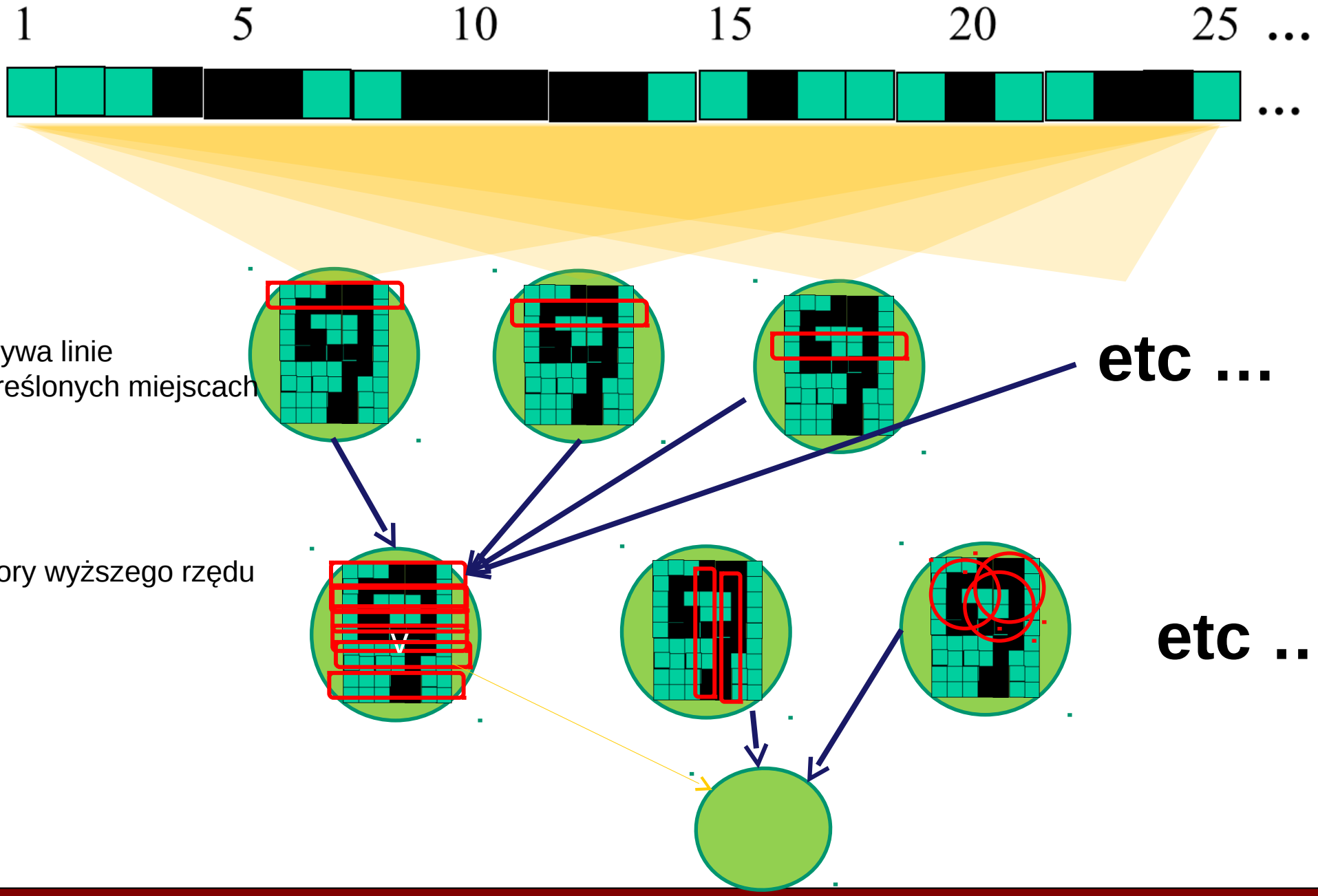


1

A co z niewrażliwością na położenie???

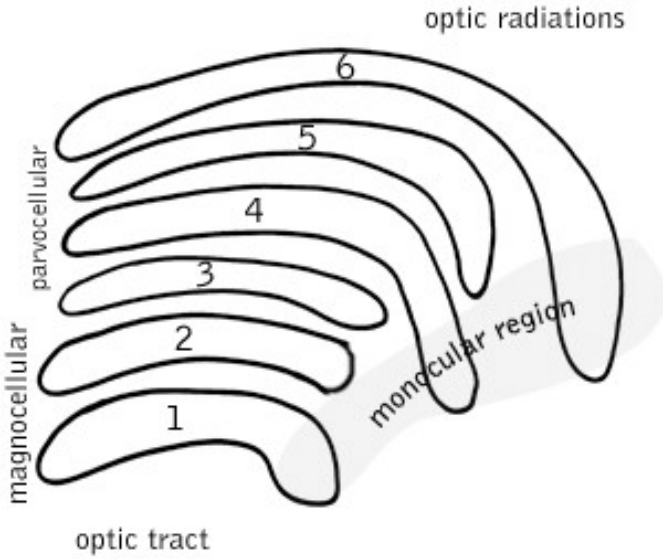
Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*

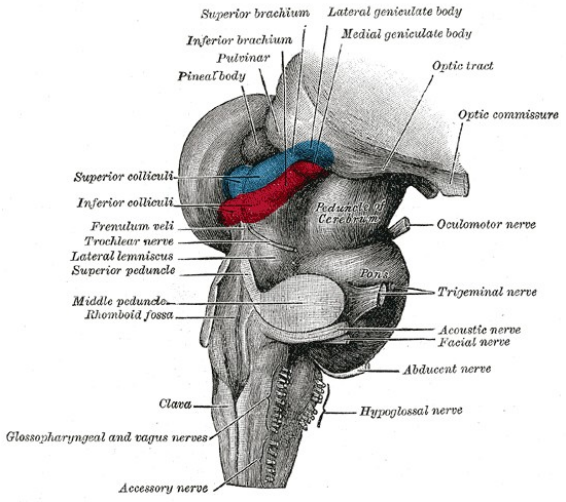
Następne warstwy mogą uczyć się cech wyższego stopnia



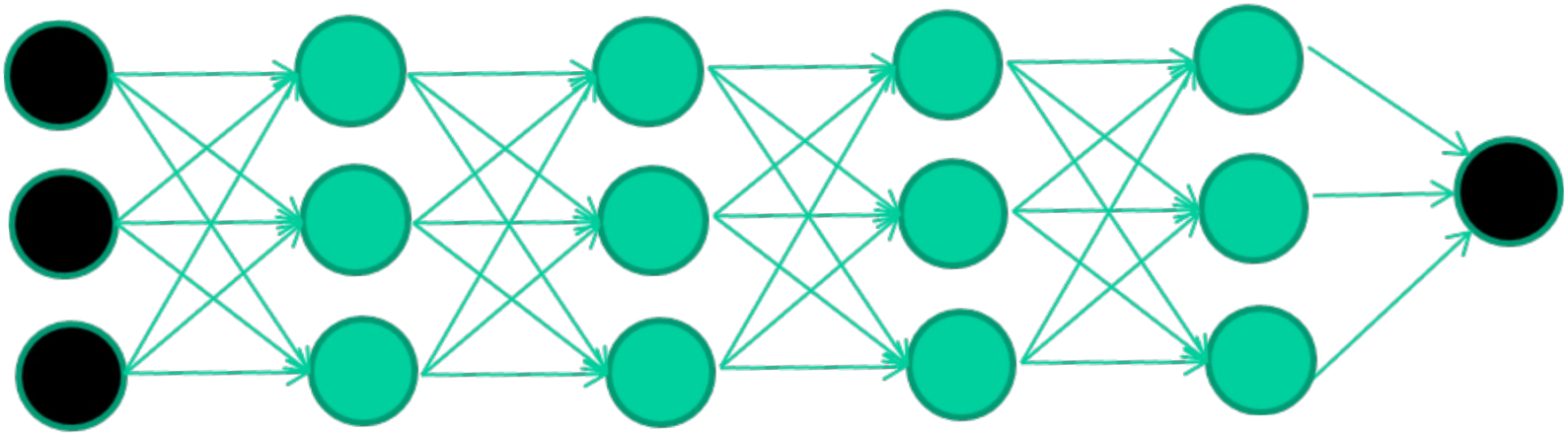
Wielowarstwowa sieć ma sens...

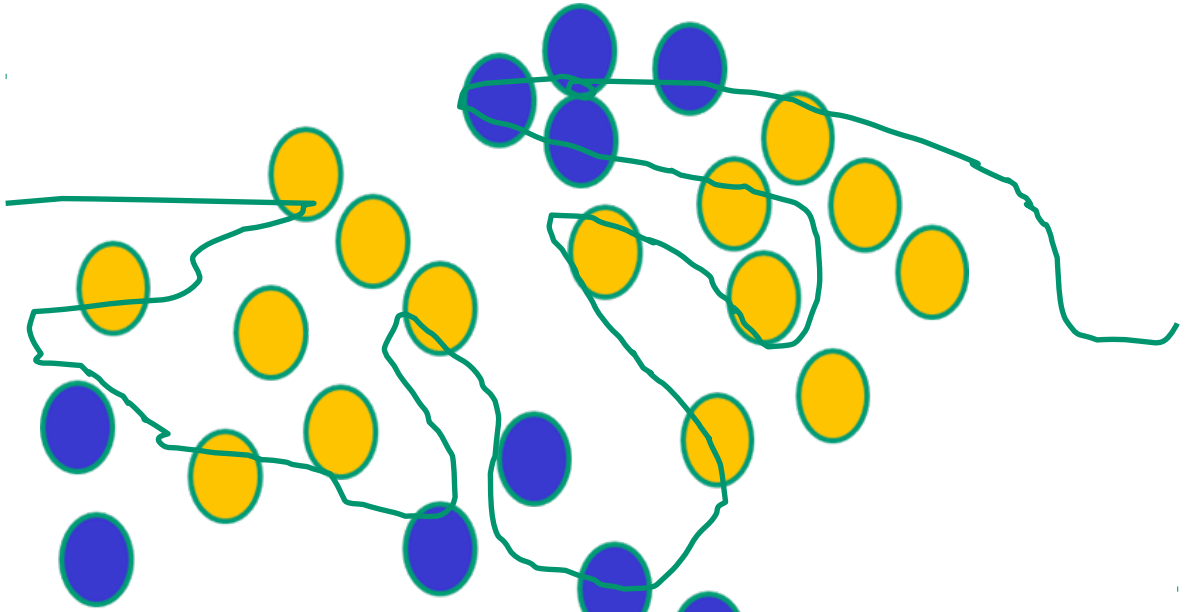
Nasze mózgi prawdopodobnie pracują w ten sposób.



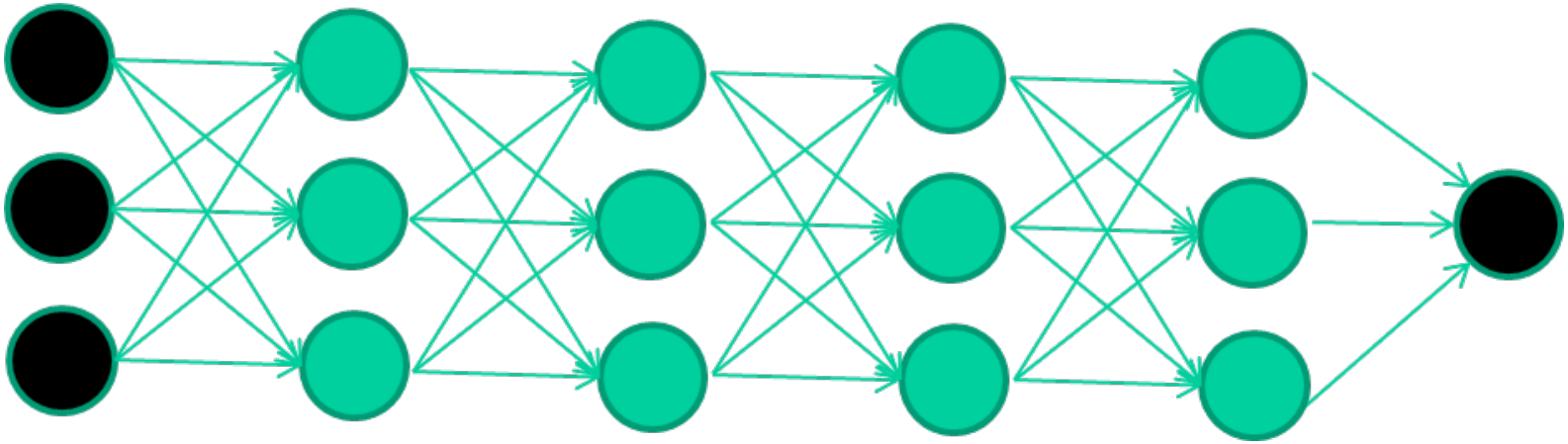


Niestety, do niedawna nie umieliśmy uczyć sieci wielowarstwowych

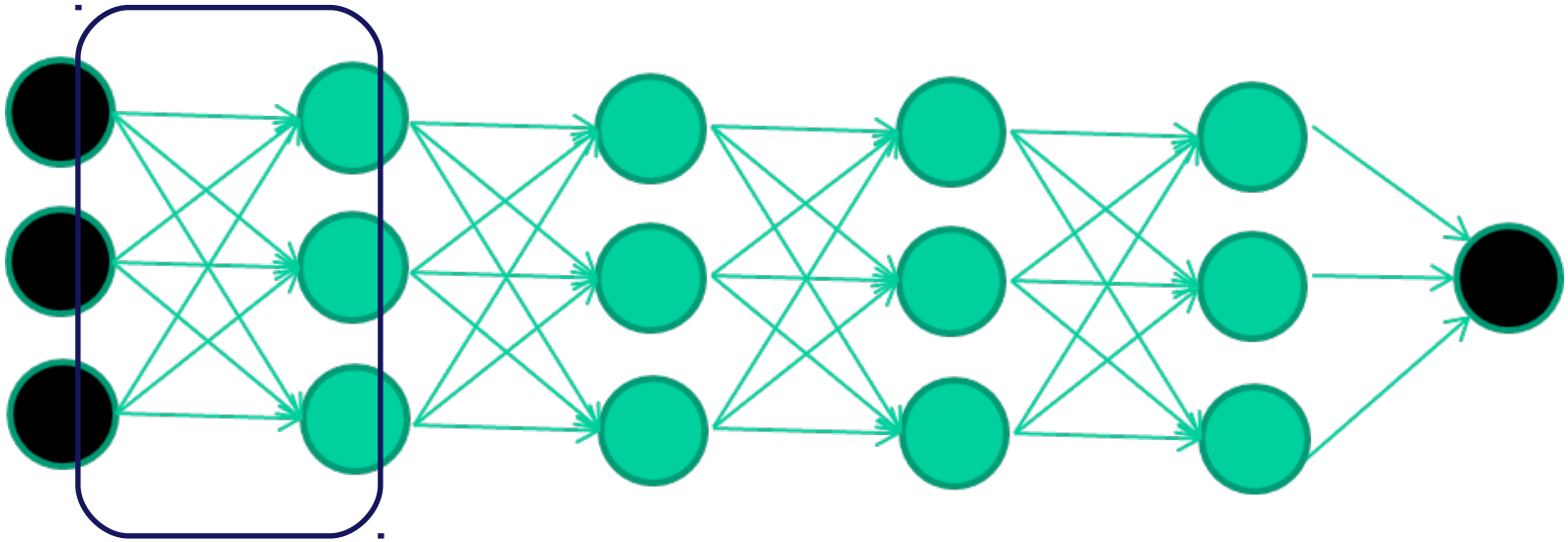




Nowa metoda uczenia (w zarysie)

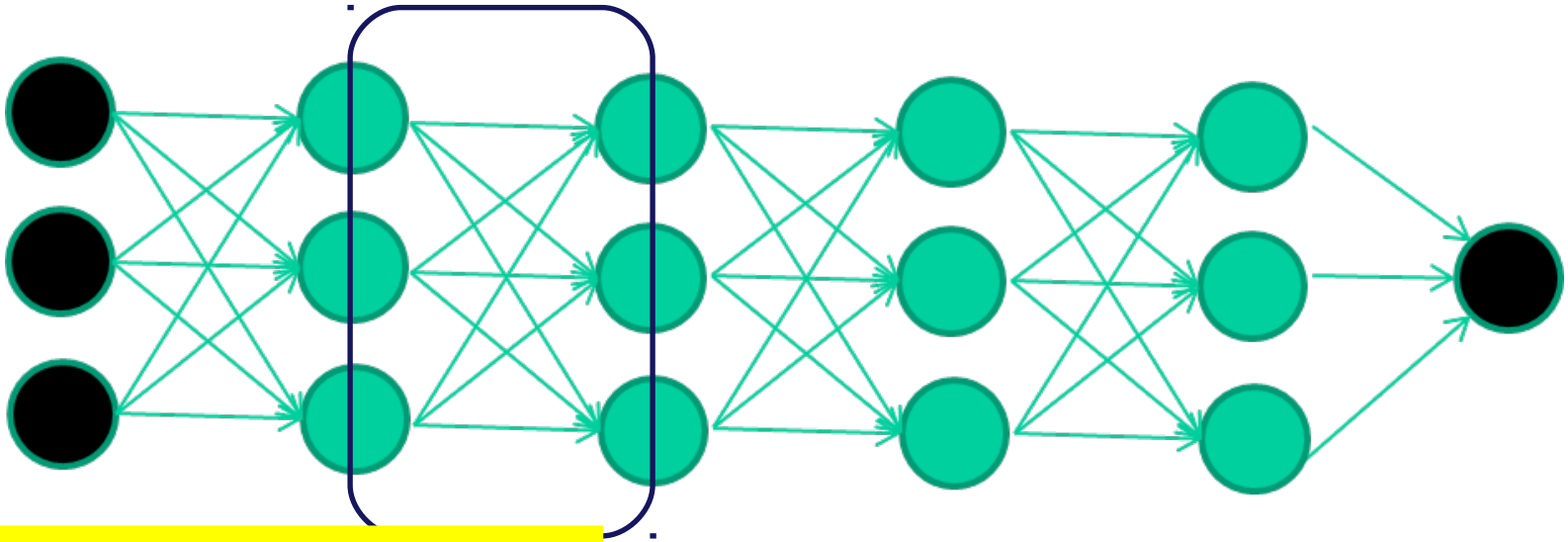


Nowa metoda uczenia (w zarysie)



Trenuj tę warstwę

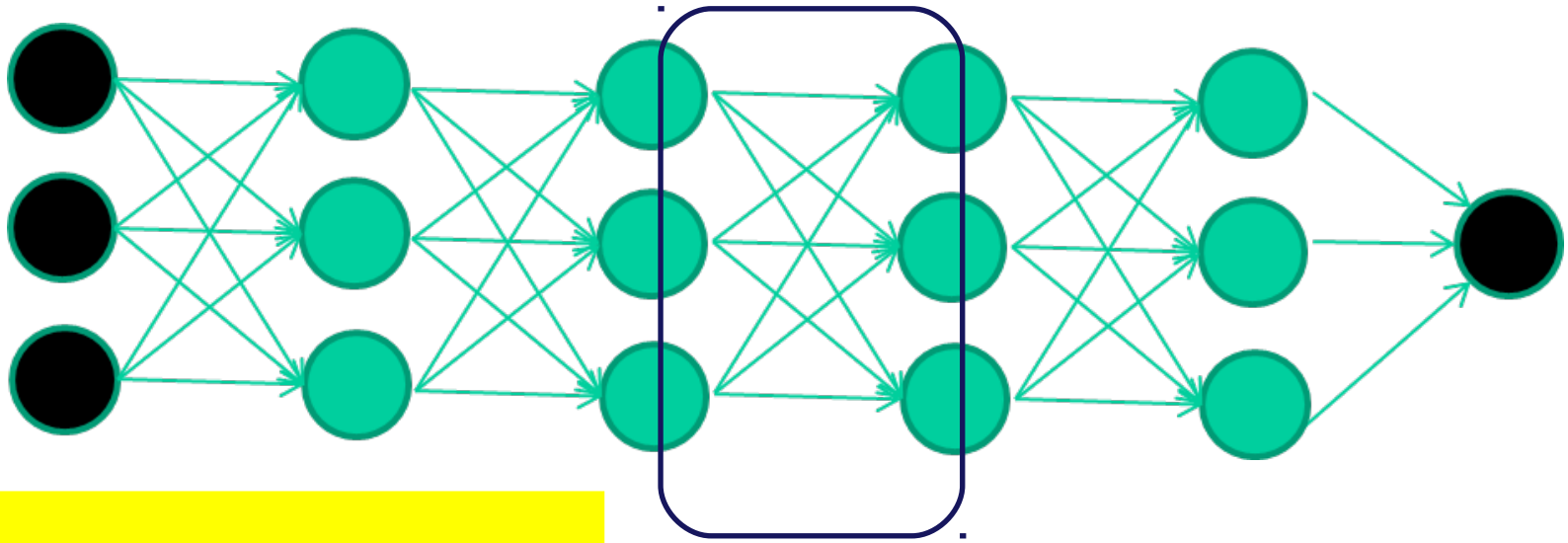
Nowa metoda uczenia (w zarysie)



Trenuj tę warstwę

Potem ta

Nowa metoda uczenia (w zarysie)

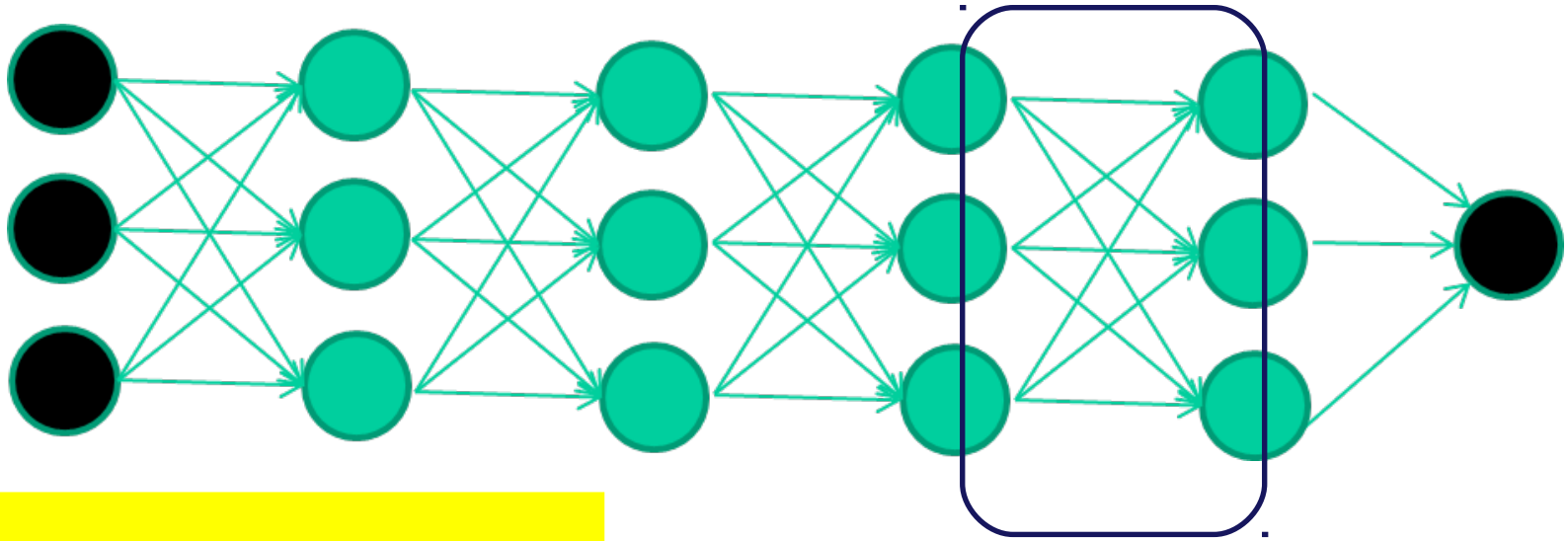


Trenuj tę warstwę

Potem ta

Potem ta

Nowa metoda uczenia (w zarysie)



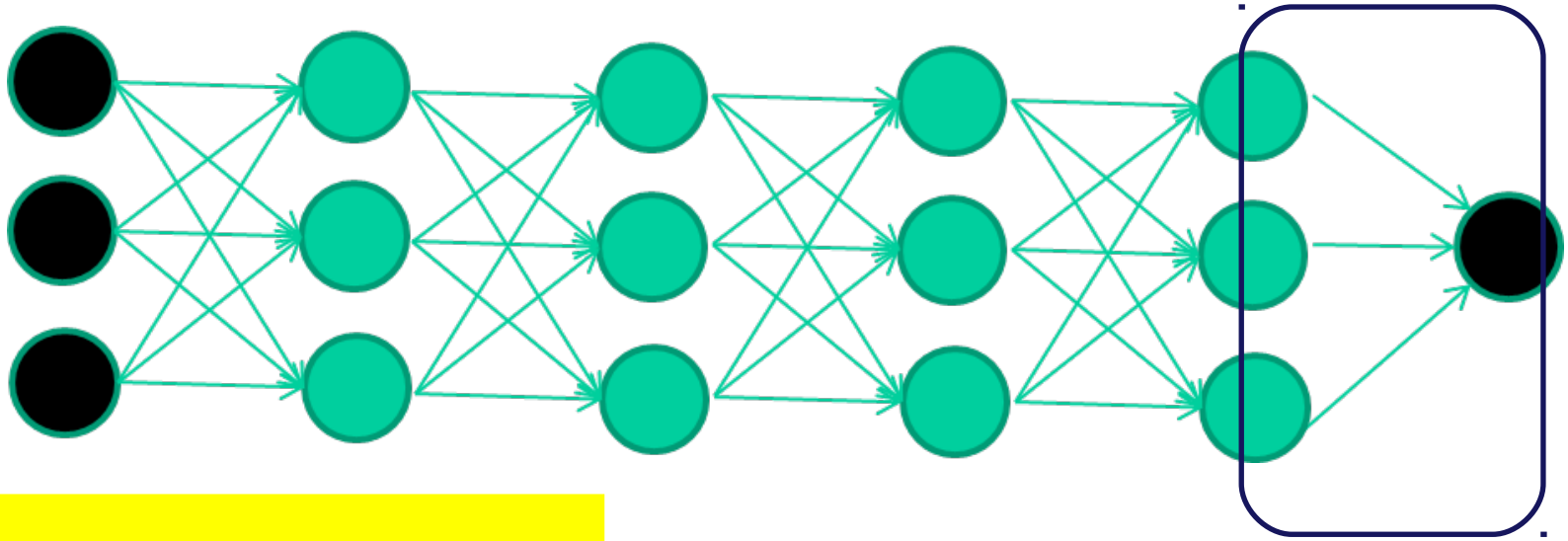
Trenuj tę warstwę

Potem ta

Potem ta

Potem ta

Nowa metoda uczenia (w zarysie)



Trenuj tę warstwę

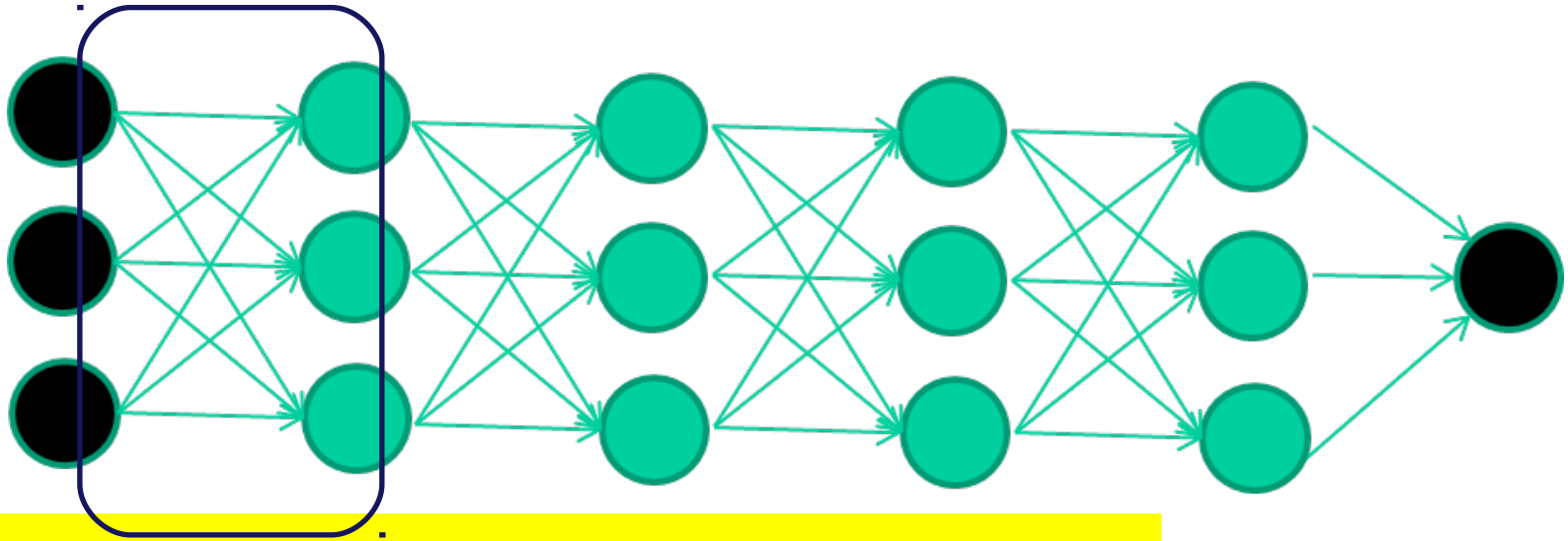
Potem ta

Potem ta

Potem ta

Na koniec ta

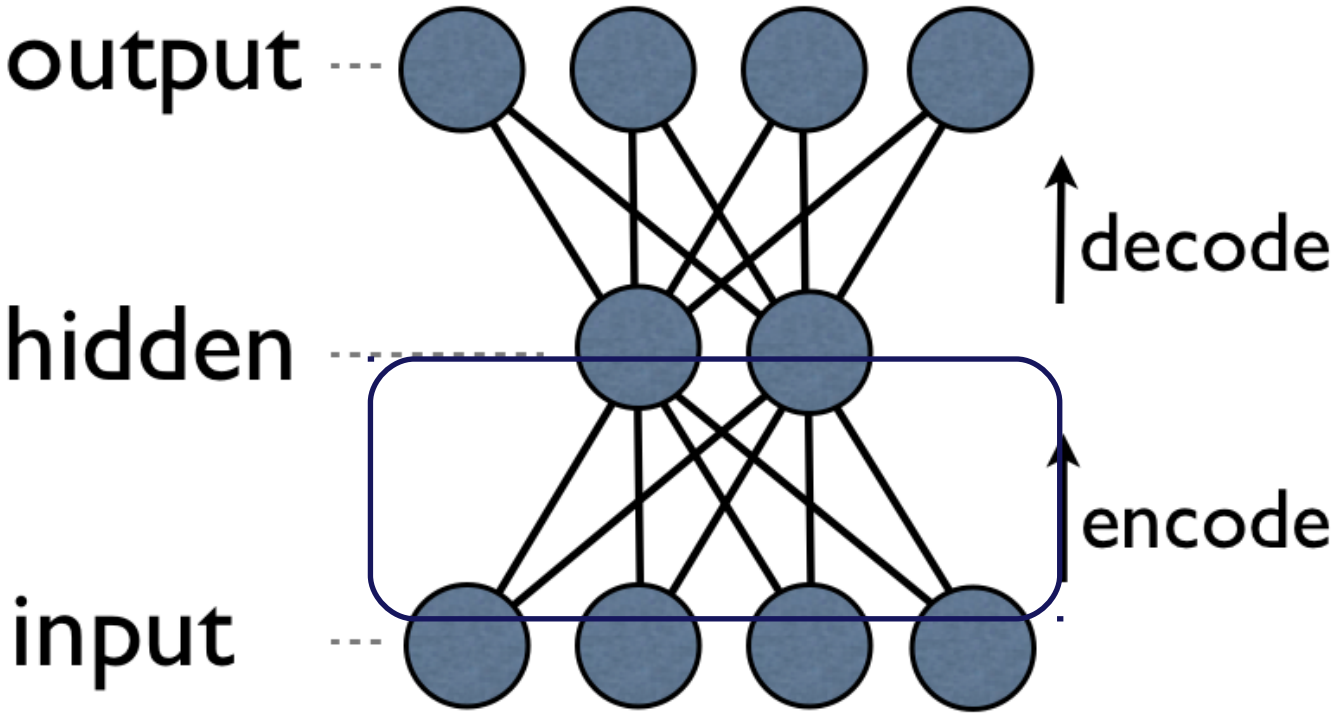
Nowa metoda uczenia (w zarysie)



Każda warstwa ukryta jest automatycznym detektorem cech

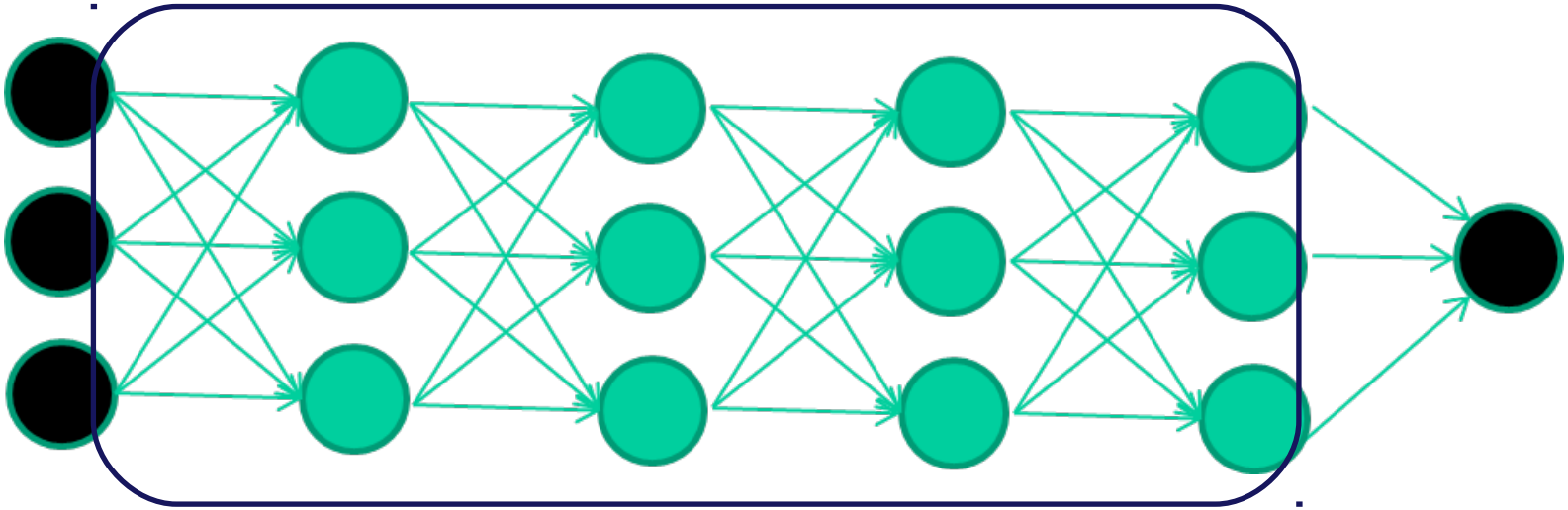
auto-encoder

“auto-encoder” jest wytrenowany tak, aby reprodukował to, co otrzyma na wejściu.

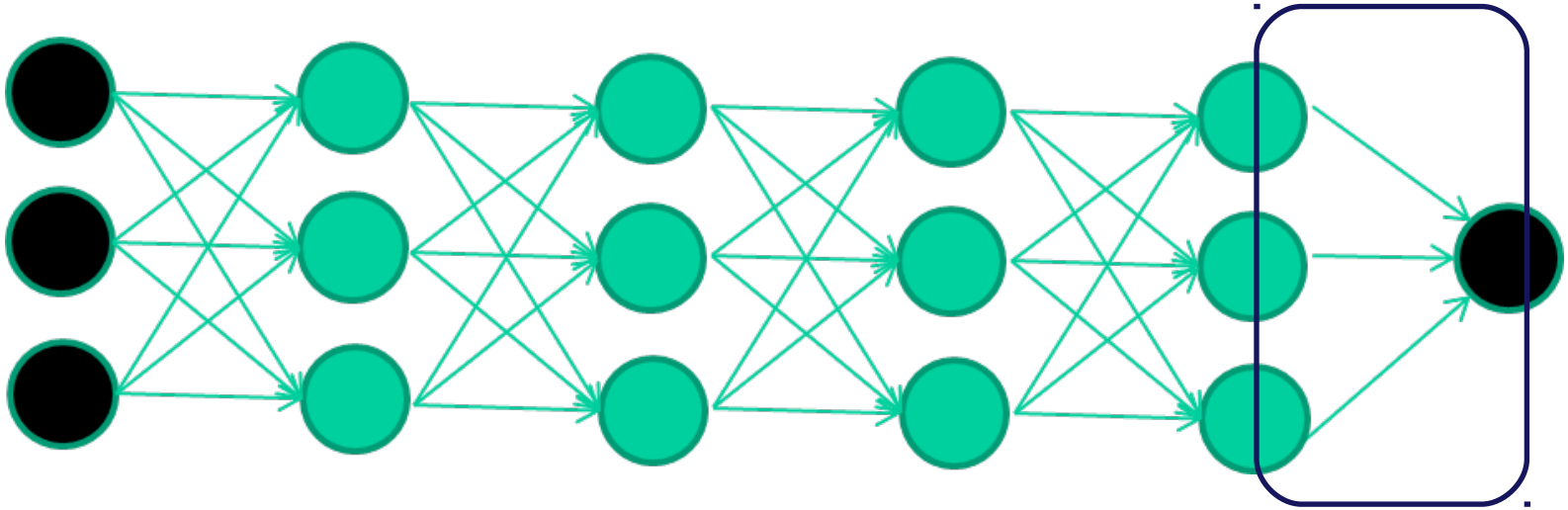


W ukrytej warstwie jest mniej neuronów niż w wejściowej, co zmusza warstwę ukrytą do efektywnego wykrywania ważnych cech

Warstwy ukryte są wytrenowane do identyfikacji cech



Warstwa wyjściowa identyfikuje, do jakiej klasy należy przypadek.



To tyle na dziś...

Applet pokazujący działanie głębokiej sieci neuronowej:

http://cs.stanford.edu/people/karpathy/convnetjs/