O uczeniu maszynowym i sieciach neuronowych

Paweł Rychlikowski

Instytut Informatyki UWr

24 maja 2019

Uczenie maszynowe. Przypomnienie

- Rozważaliśmy uczenie **z nadzorem** (to znaczy sytuację, gdy mamy pewne przykłady i chcemy je uogólnić)
- Rozważamy dwie ważne klasy takich zadań:
  1. Klasyfikacji (wybór klasy dla przykładu, zbiór klas jest skończony i niezbyt duży)
  2. Regresji (wybór wartości liczbowej dla przykładu)

Dane i uczenie

- Dane uczące – zbiór przykładów, często w postaci:

  (wektor-cech1, wynik1)
  (wektor-cech2, wynik2)
  (wektor-cech3, wynik3)
  ...

Podział dostępnych danych

Definicja 1

Zbiór **uczący** jest podstawowym zbiorem, który będziemy wykorzystywać do zdobywania wiedzy o problemie.

Definicja 2

Zbioru **walidacyjnego** używamy do wyboru rodzaju algorytmu lub do wyboru hiperparametrów algorytmu.

Definicja 3

Zbiór **testowy** używany jest **tylko** do ostatecznego testu, który ma nas przekonać, jak dobrze uogólnia rzeczywistość nasz mechanizm.

Do zbioru testowego nawet nie zaglądamy, nie analizujemy błędów, itd!

Klasyczne zadanie klasyfikacji obrazów

MNIST jest zbiorem około 60K czarnobiałych obrazków 28×28 zawierających ręcznie pisane cyfry.

Jest on powszechnie używany do testowania różnych algorytmów uczenia (głównie klasyfikacji, ale nie tylko)

MNIST – przedstawienie danych

MNIST – wyszukiwanie wzorców

- Naturalnym rozwiązaniem jest stworzenie wzorca (lub wzorców) dla każdej cyfry.
- Jak to zrobić?

Dwa warianty

1. Jeden wzorzec dla wszystkich obrazków danej cyfry.
2. Wiele wzorców (nawet: każda cyfra wzorcem

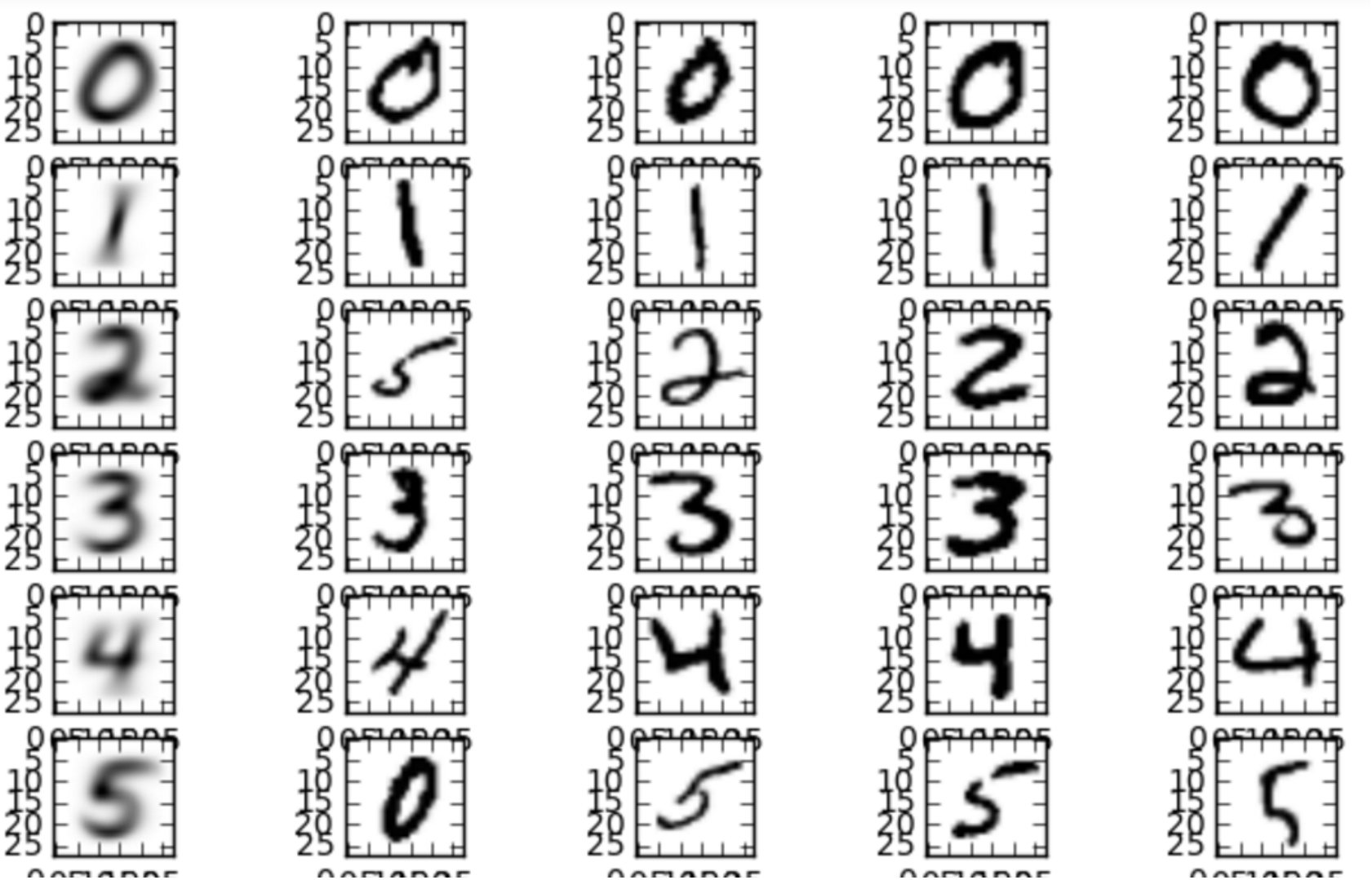
Jeden wzorzec dla cyfry

- Naturalnym wzorcem może być średnia wszystkich egzemplarzy danej cyfry
- Przy klasyfikacji obrazka **o** wybieramy wzorzec **w** najbardziej podobny do **o**
- Co to znaczy podobny?
  - Wysoki iloczyn skalarny?
  - *Wysoki iloczyn skalarny znormalizowanych wektorów?*
  - Niewielka odległość euklidesowa (a może jakaś inna)?

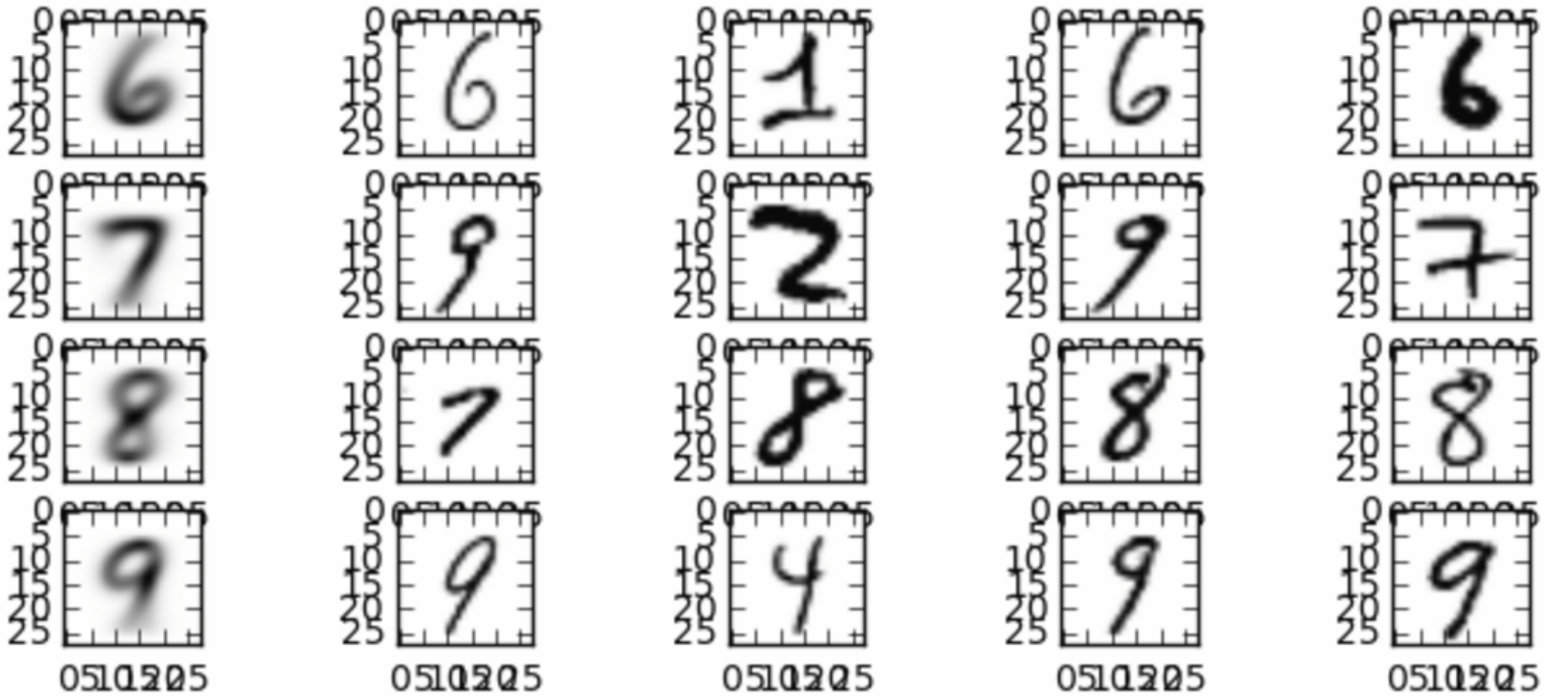
Wyniki eksperymentu

Poprawność klasyfikacji to około **82.1%**

Wyniki klasyfikacji z „wzorcami średnimi”

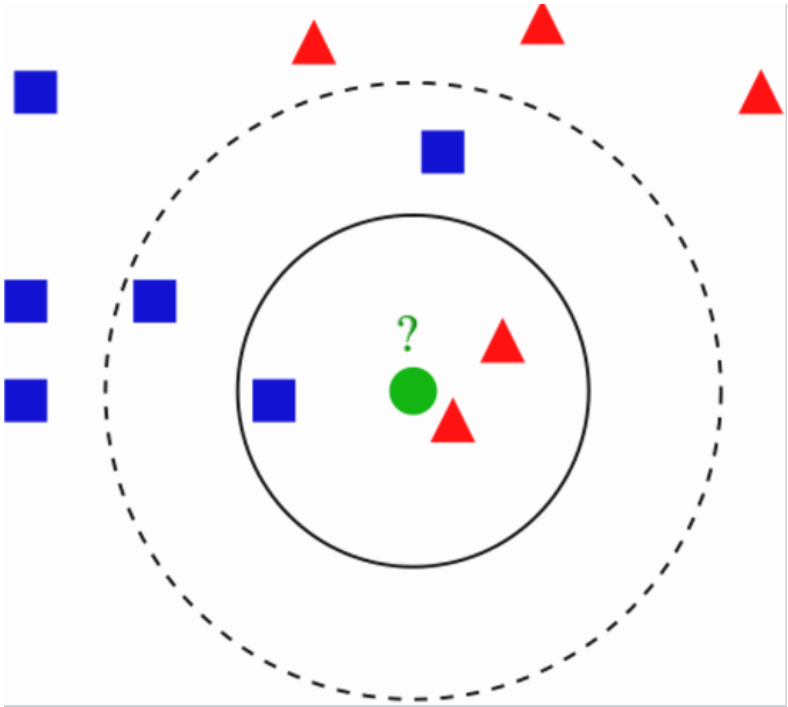


Wyniki klasyfikacji z „wzorcami średnimi”



K najbliższych sąsiadów. KNN

- Pamiętamy wszystkie wektory ze zbioru uczącego.
- Klasyfikując obrazek znajdujemy K najbliższych sąsiadów i pozwalamy im głosować

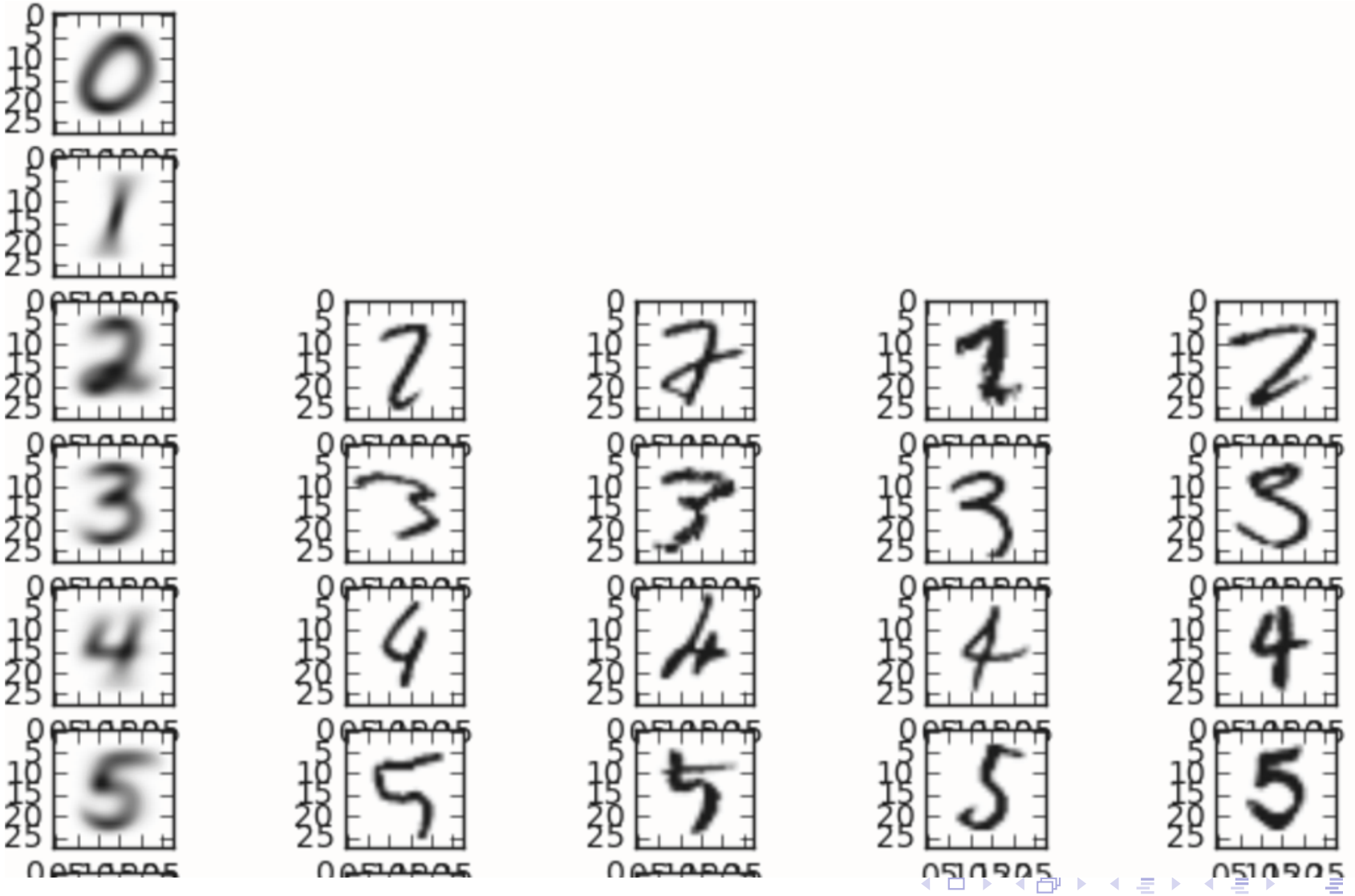


MNIST i KNN

- Podobnieństwo mierzymy iloczynem skalarnym znormalizowanych wektorów (czyli cosinusem)
- Testujemy na próbce (bo inaczej trwa bardzo długo)
- $K = 3$
- Wyniki:
  - Zbiór uczący: **98.55%**
  - Zbiór testowy: około **97%**

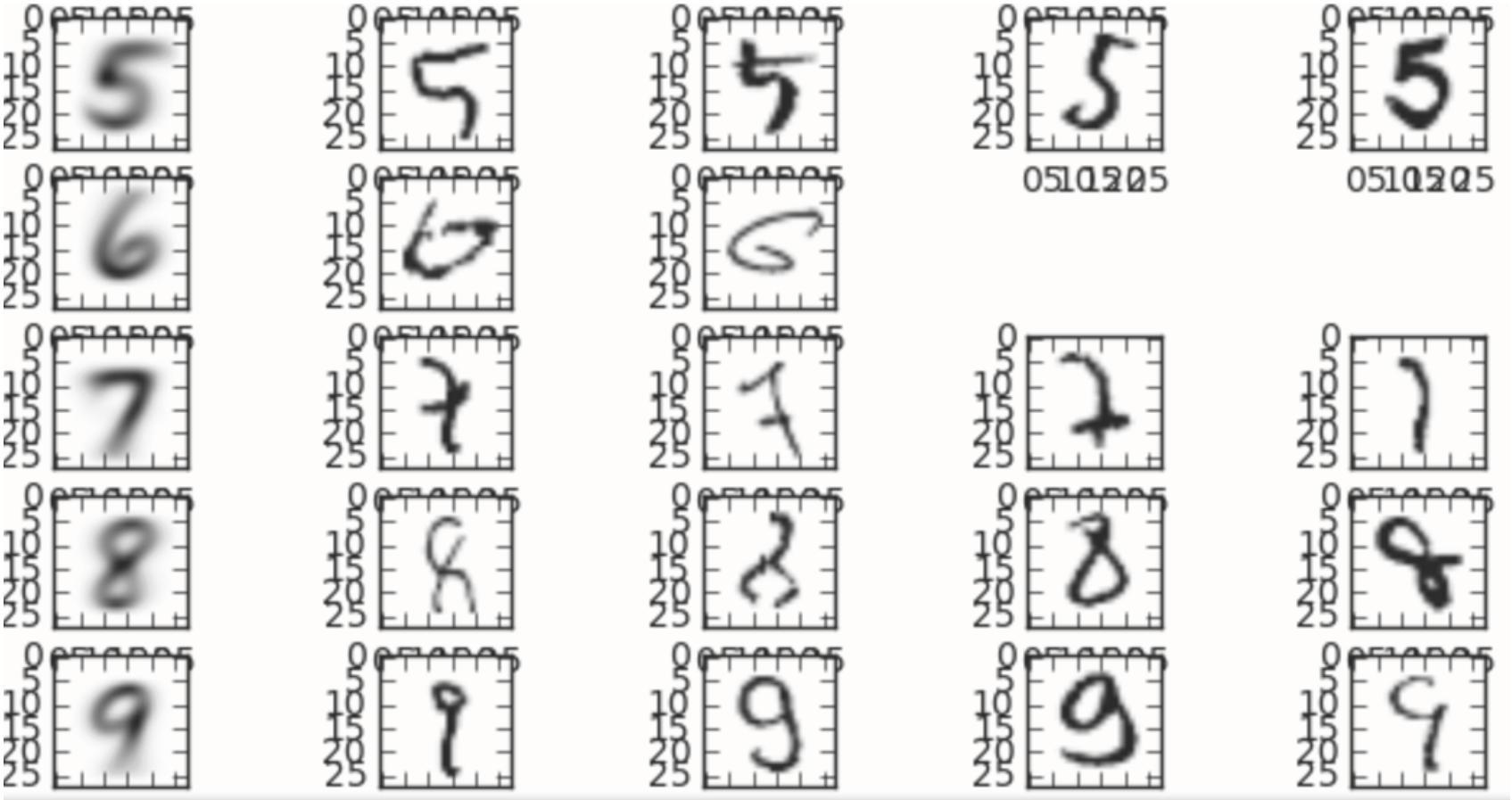
MNIST i KNN. Przykładowe błędy

Z tymi cyframi mieliśmy problemy (zaznaczona prawidłowa klasyfikacja)



MNIST i KNN. Przykładowe błędy

Z tymi cyframi mieliśmy problemy (zaznaczona prawidłowa klasyfikacja)



Uczenie funkcji liniowej

- Klasyfikacji (czy regresji) możemy dokonywać na bazie wartości:

$$\sum_{i=0}^{N} w_i \phi_i(x)$$

- Cechami (dla MNIST-a) są po prostu wartości kolejnych pikseli
- Jedna funkcja „wystarcza” dla binarnego zadania typu: czy cyfra jest piątką? (i jak bardzo)
- Do rozwiązywania MNIST-a potrzebujemy 10 takich funkcji, wybieramy cyfrę dla tej funkcji, która zwraca największą wartość.

Funkcja liniowa i Reversi

- Możemy zdefiniować zadanie uczenia (regresji) dla Reversi:

  Widząc sytuację na planszy w ruchu 20 postaraj się przewidzieć zakończenie gry.

- Cechy: binarne cechy mówiące o zajętości pola.
- Przykładowo:
  Czy pole (4,5) jest czarne?
  Czy pole (1,1) jest białe?

Uwaga

Taki mechanizm byłby użyteczną funkcją heurystyczną, do użycia np. w algorytmie MiniMax.

Funkcja kosztu

Definicja

Funkcja kosztu (loss) opisuje, jak bardzo niezadowoleni jesteśmy z działania naszego mechanizmu (klasyfikatora, przewidywacza wartości).

Funkcja kosztu jest określona na: danych uczących (x,y) oraz wagach (parametrach klasyfikatora). Przy czym dane uczące traktujemy jako parametr, a wagi – jako właściwe argumenty.

Przykładowa funkcja kosztu: błąd średniokwadratowy

Funkcja kosztu (2)

Średniokwadratowa funkcja straty

$$\text{TrainLoss}(\mathbf{w}) = \frac{1}{|D_{\text{train}}|} \sum_{(x,y)\in D_{\text{train}}} (f_{\mathbf{w}}(x) - y)^2$$

Wariant liniowy:

$$f_{\mathbf{w}}(x) = \sum_{i=0}^{N} w_i \phi_i(x)$$

Gradient

- Gradient jest wektorem pochodnych cząstkowych.
- Wskazuje kierunek największego wzrostu funkcji.
- Policzmy gradient na tablicy, dla powyżej zdefiniowanej funkcji kosztu.

Jeszcze o gradiencie

Wzór

Pochodna po w_i

$$\frac{1}{|D_{\text{train}}|} \sum_{(x,y)\in D_{\text{train}}} 2 \cdot (f_{\mathbf{w}}(x) - y) \cdot w_i \phi_i(x)$$

- Obliczenie gradientu wymaga przejścia przez cały zbiór uczący
- Maksymalizacja funkcji: dodawanie gradientu przemnożonego przez małą stałą (minimalizacja: odejmowanie)

Stochastic Gradient Descent

Obliczamy nie cały gradient, tylko jego składnik, związany z jednym egzemplarzem danych uczących i jego dodajemy (przemnożonego przez stałą)

Sieci neuronowe w paru prostych slajdach

- Wybierzemy absolutne minimum tego, co należy wiedzieć o sieciach neuronowych
- Oczywiście nie będzie to w pełni kompletna wiedza.

Slajd 1. Neuron

- Neuron to funkcja $f : \mathcal{R}^n \to \mathcal{R}$

$$f(x_1 \dots x_n) = \sigma(\sum_1^n w_i x_i + b)$$

- σ jest jakąś ustaloną funkcją nieliniową, raczej rosnącą, raczej różniczkowalną, na przykład: $\max(0, v)$, albo $\tanh(v)$
- Wygodna (jak za chwilę zobaczymy) jest notacja wektorowo-macierzowa, w niej mamy:

$$f(\mathbf{x}) = \sigma(\mathbf{w}^T \cdot \mathbf{x} + b)$$

Slajd 2. Prosta sieć neuronowa

- Warstwa to funkcja $\mathcal{R}^n \to \mathcal{R}^m$.
- Najbardziej typowa warstwa wyraża się wzorem:

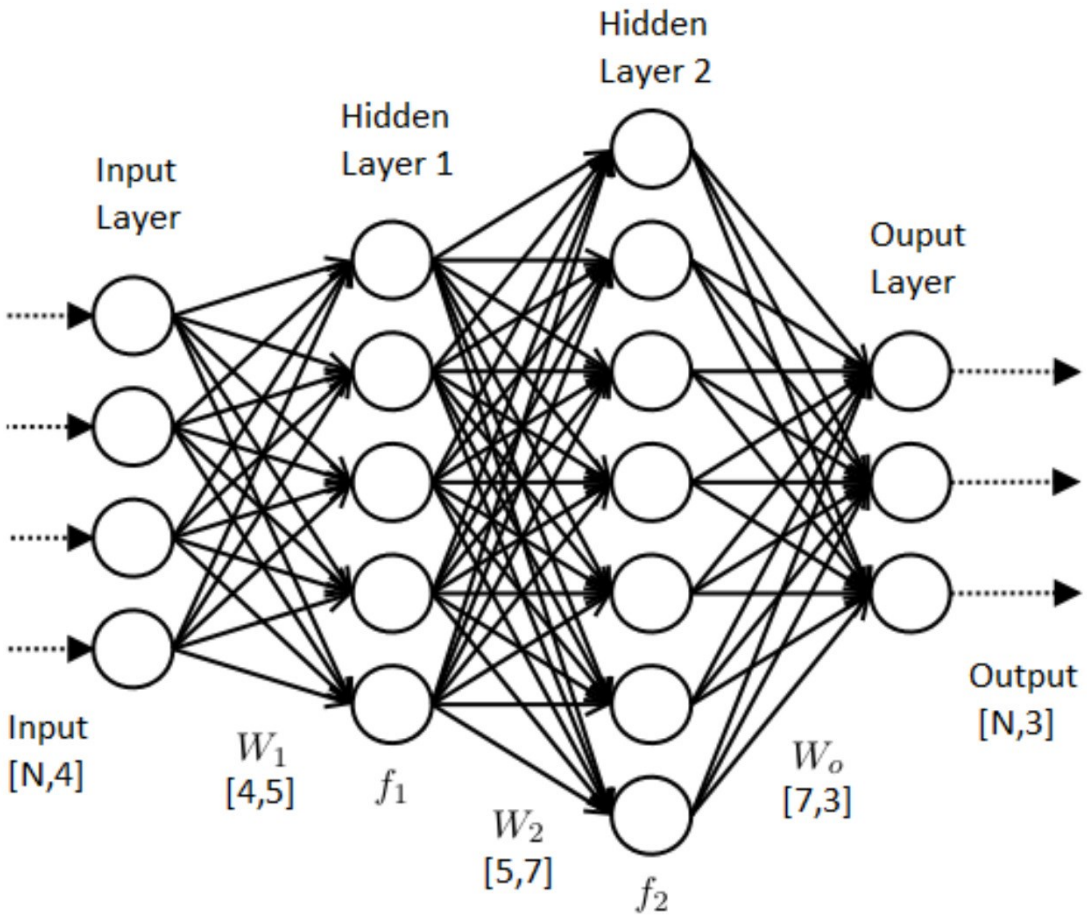
$$L(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

- **Uwaga:** $\mathbf{W}$ jest macierzą wag (złożoną z wektorów wag), a $\sigma(y_1 \dots y_m) = (\sigma(y_1) \dots \sigma(y_m))$

Definicja

Sieć neuronowa typu MLP jest złożeniem warstw (z różnymi macierzami wag dla każdej warstwy).

Slajd 2b. Prosta sieć neuronowa



Źródło: VIASAT (https://medium.com/coinmonks/the-artificial-neural-networks-handbook-part-1-f9ceb0e376b4)

Slajd 3. Uczenie sieci

Zadanie

Danymi jest ciąg $(\mathbf{x}_i, \mathbf{y}_i)$ opisujący porządane zachowanie sieci S oraz architektura tejże sieci (liczba warstw, ich wymiary, funkcja/funkcje σ).
Chcemy tak dobrać parametry ($\mathbf{W_k}$ oraz $\mathbf{b_k}$) żeby dla każdego i

$$S(\mathbf{x}_i) \approx \mathbf{y}_i$$

Funkcja kosztu

Powyższe zadanie formalizujemy jako zadanie znalezienia takich parametrów, że koszt błędów jest jak najmniejszy. Przykładowo, jeżeli wyjściem jest liczba, to możemy wybrać:

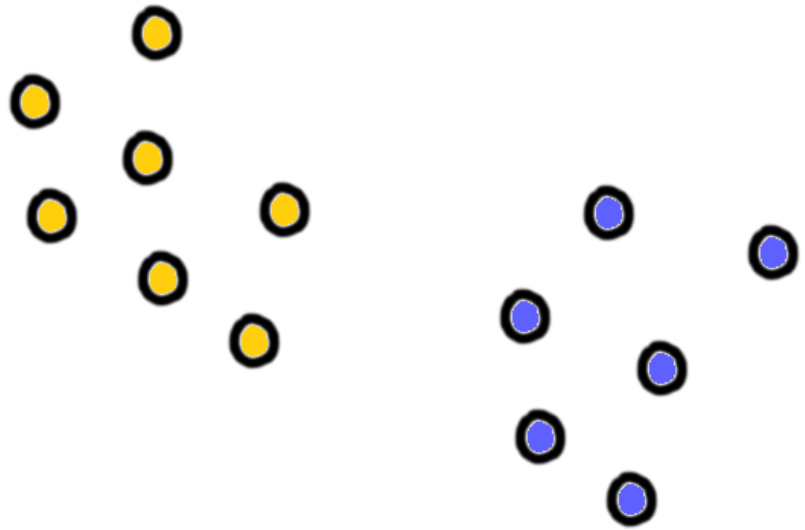
$$\text{Loss}(\theta) = \sum_{i}^{n} (S_\theta(\mathbf{x}_i) - y_i)^2$$

Sieci neuronowe w praktyce. Klasyfikacja

Ogólne założenia (przypadek dwóch klas)

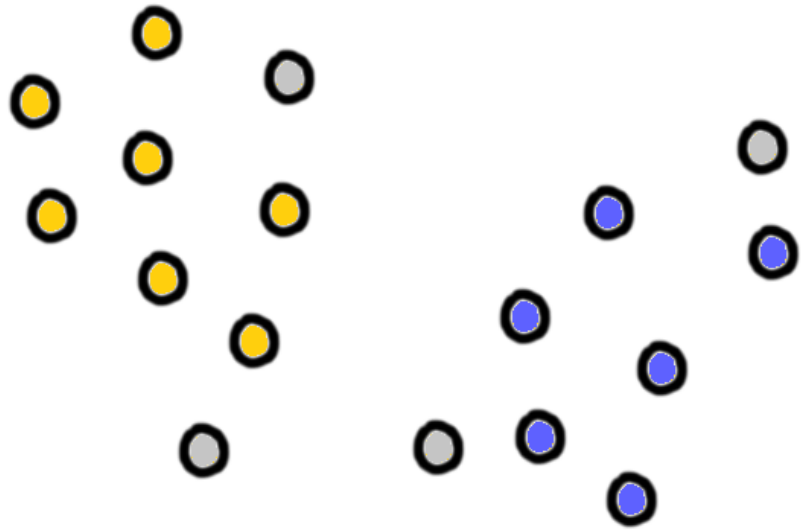
Mamy jakiś zbiór przykładów pozytywnych i negatywnych, interesuje nas mechanizm, który będzie poprawnie klasyfikował nieznane przykłady.

Klasyfikacja w $\mathcal{R}^2$



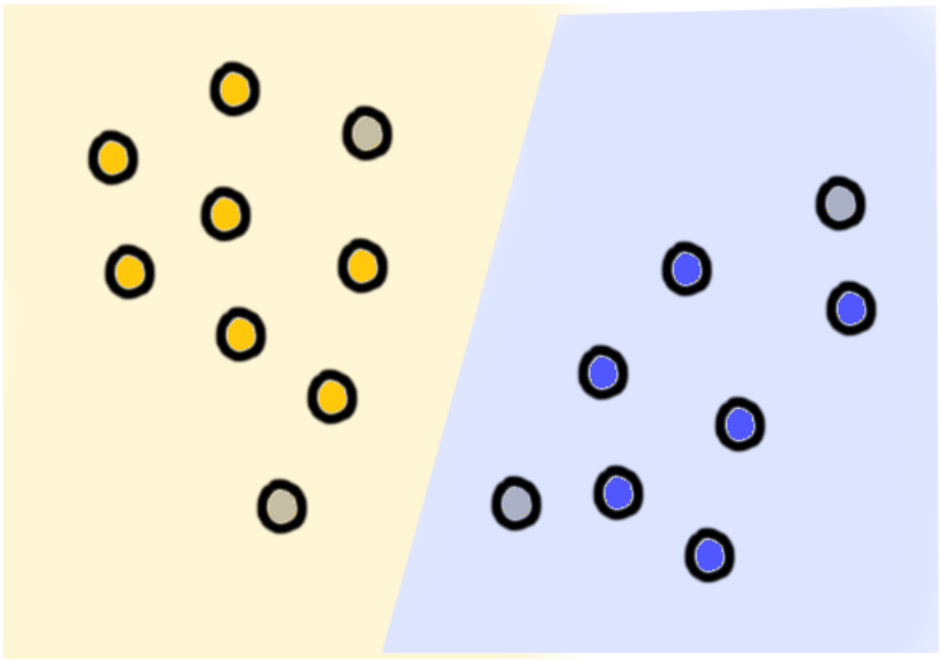
- Dane punkty wraz z informacją o kolorze.
- Mechanizm powinien umieć określać kolor nieznanych punktów.
- Możemy o tym myśleć, jako o „kolorowaniu płaszczyzny”

Klasyfikacja w $\mathcal{R}^2$



- Dane punkty wraz z informacją o kolorze.
- Mechanizm powinien umieć określać kolor nieznanych punktów.
- Możemy o tym myśleć, jako o „kolorowaniu płaszczyzny”

Klasyfikacja w $\mathcal{R}^2$



- Dane punkty wraz z informacją o kolorze.
- Mechanizm powinien umieć określać kolor nieznanych punktów.
- Możemy o tym myśleć, jako o „kolorowaniu płaszczyzny”

Spróbujmy poeksperymentować chwilę z Tensorflow Playground