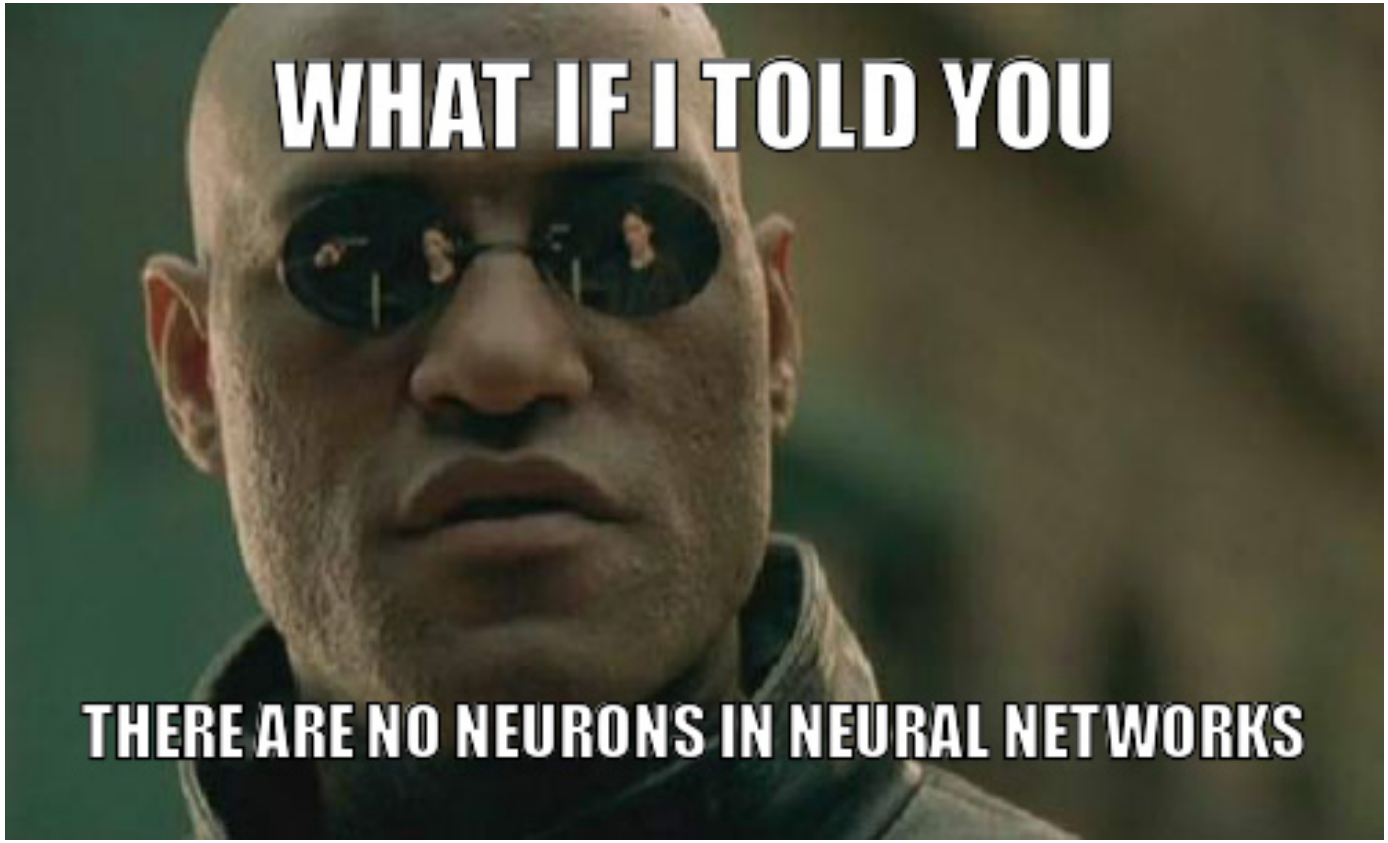
Filip Graliński

Ku sztucznym sieciom neuronowym

Część I

Wstęp

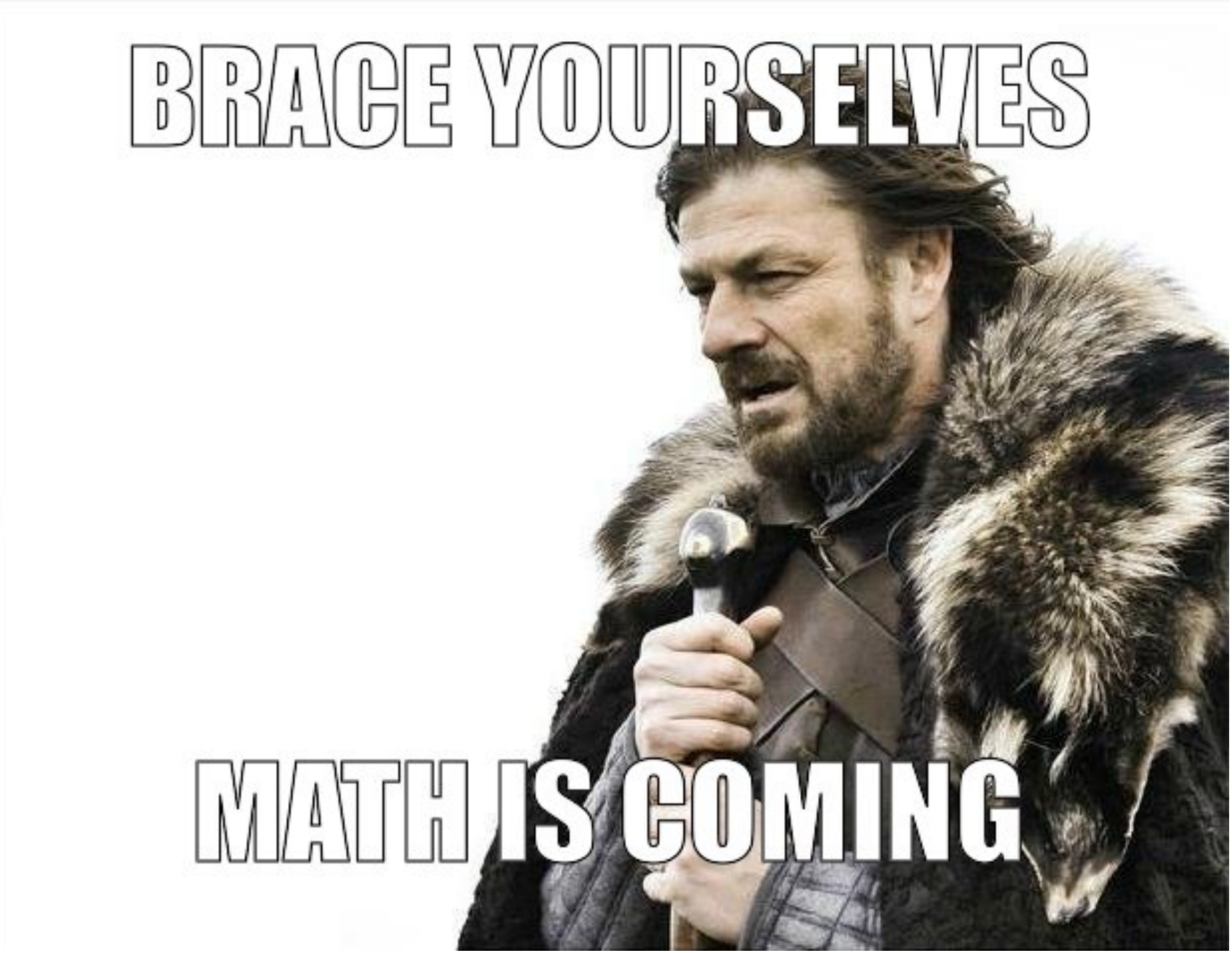

WHAT IF I TOLD YOU
THERE ARE NO NEURONS IN NEURAL NETWORKS

Sztuczna sieć neuronowa to NIE:

- sztuczny mózg
- mózg elektronowy
- symulacja mózgu

To co to jest?
Sztuczna sieć neuronowa to (zazwyczaj) układ wielu jednostek dokonujących regresji logistycznej.

WTF??? – spokojnie, podamy później intuicję bardziej strawne dla programistów

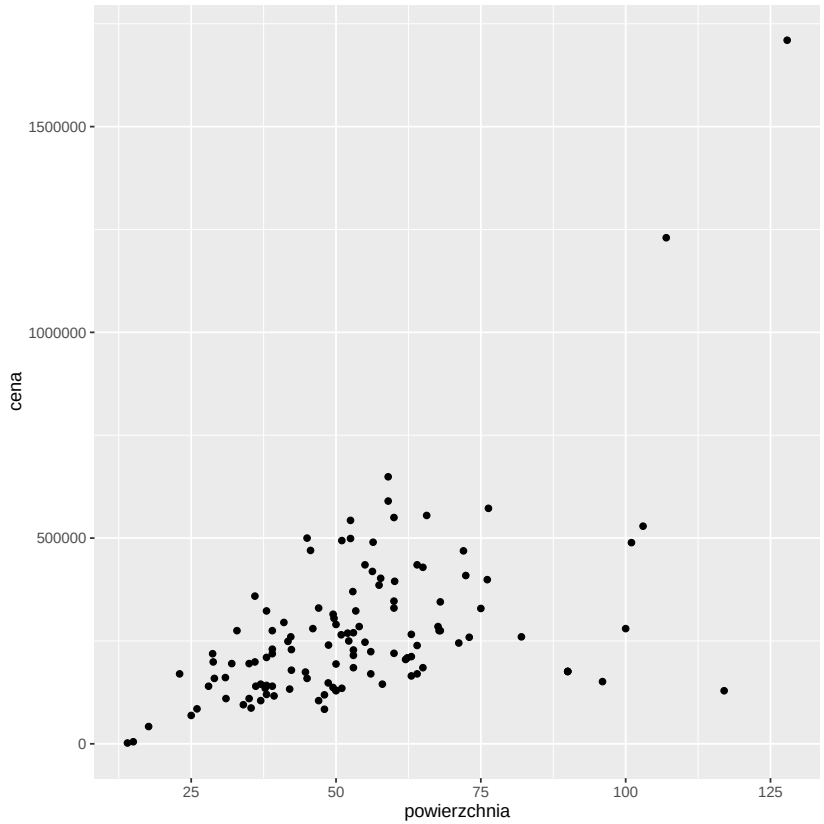

BRACE YOURSELVES
MATH IS COMING

Część II

Regresja logistyczna

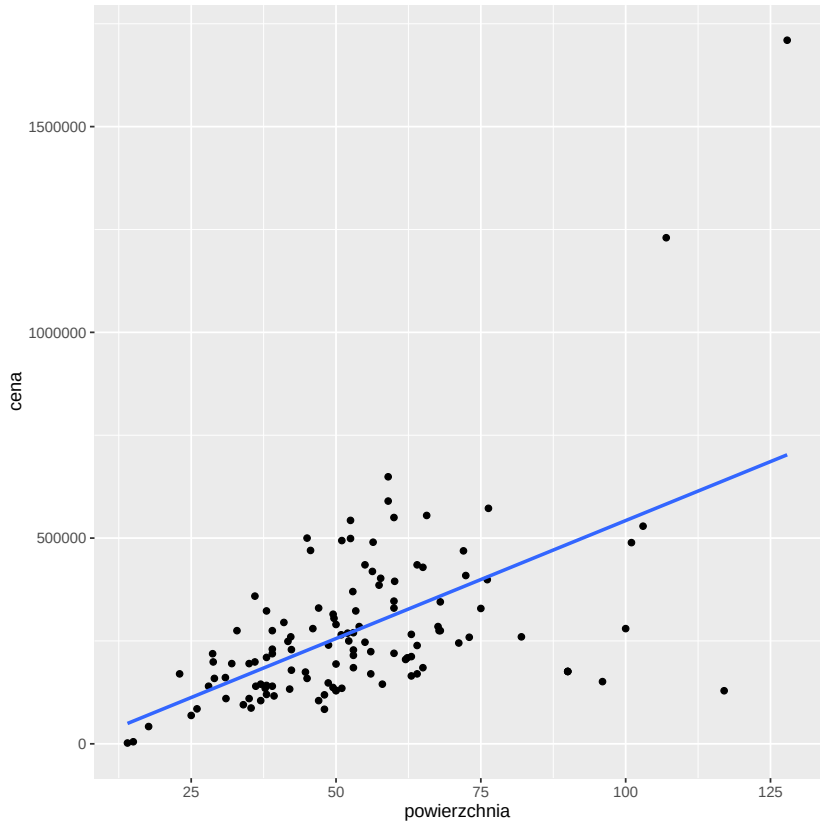
Regresja liniowa

Regresja liniowa jest prosta . . .



Regresja liniowa

Regresja liniowa jest prosta . . .



. . . dosłownie — dopasuj prostą $y = ax + b$ do punktów

Należy odgadnąć a i b tak, aby zminimalizować błąd kwadratowy, tj. wartość:

$$\sum_{i=1}^{n}(y_i - (ax_i + b))^2$$

Regresja liniowa (cd.)

Regresje liniowa jest łatwa do rozwiązania — wystarczy podstawić do wzoru!

$$\hat{b} = \frac{\sum_{i=1}^{n} x_i y_i - \frac{1}{n} \sum_{i=1}^{n} x_i \sum_{j=1}^{n} y_j}{\sum_{i=1}^{n} x_i^2 - \frac{1}{n} (\sum_{i=1}^{n} x_i)^2}$$

$$\hat{a} = \bar{y} - \hat{b}\,\bar{x}$$

Regresja liniowa (cd.)

Regresje liniowa jest łatwa do rozwiązania — wystarczy podstawić do wzoru!

$$\hat{b} = \frac{\sum_{i=1}^{n} x_i y_i - \frac{1}{n} \sum_{i=1}^{n} x_i \sum_{j=1}^{n} y_j}{\sum_{i=1}^{n} x_i^2 - \frac{1}{n} (\sum_{i=1}^{n} x_i)^2}$$

$$\hat{a} = \bar{y} - \hat{b}\,\bar{x}$$

Na przykład dla mieszkań: b = -30809.203 zł, a = 5733.693 zł/m^2.

Regresja liniowa (cd.)

W praktyce mamy do czynienia z wielowymiarową regresją liniową. Cena mieszkań może być prognozowana na podstawie:

- powierzchni
- liczby pokoi
- piętra
- wieku
- odległości od Dworca Centralnego w Warszawie
- cer zerojedynkowych:
  - czy wielka płyta?
  - czy jest jacuzzi?
  - czy jest grzyb?
  - czy to Kielce?

Regresja liniowa (cd.)

. . . więc uogólniamy na wiele (k) wymiarów:

$$y = w_0 + w_1 x_1 + \ldots + w_k x_k = w_0 + \sum_{j=1}^{k} w_j x_j$$

gdzie:

- $x_1, \ldots, x_k$ – zmienne, na podstawie których zgadujemy
- w_0, $w_1, \ldots, w_k$ – wagi modelu (do wymyślenia na podstawie przykładów)
- y – odgadywana wartość

Też istnieje wzór ładny wzór na wyliczenie wektora wag!

$$\mathbf{w} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

. . . niestety odwracanie macierzy nie jest tanie :(

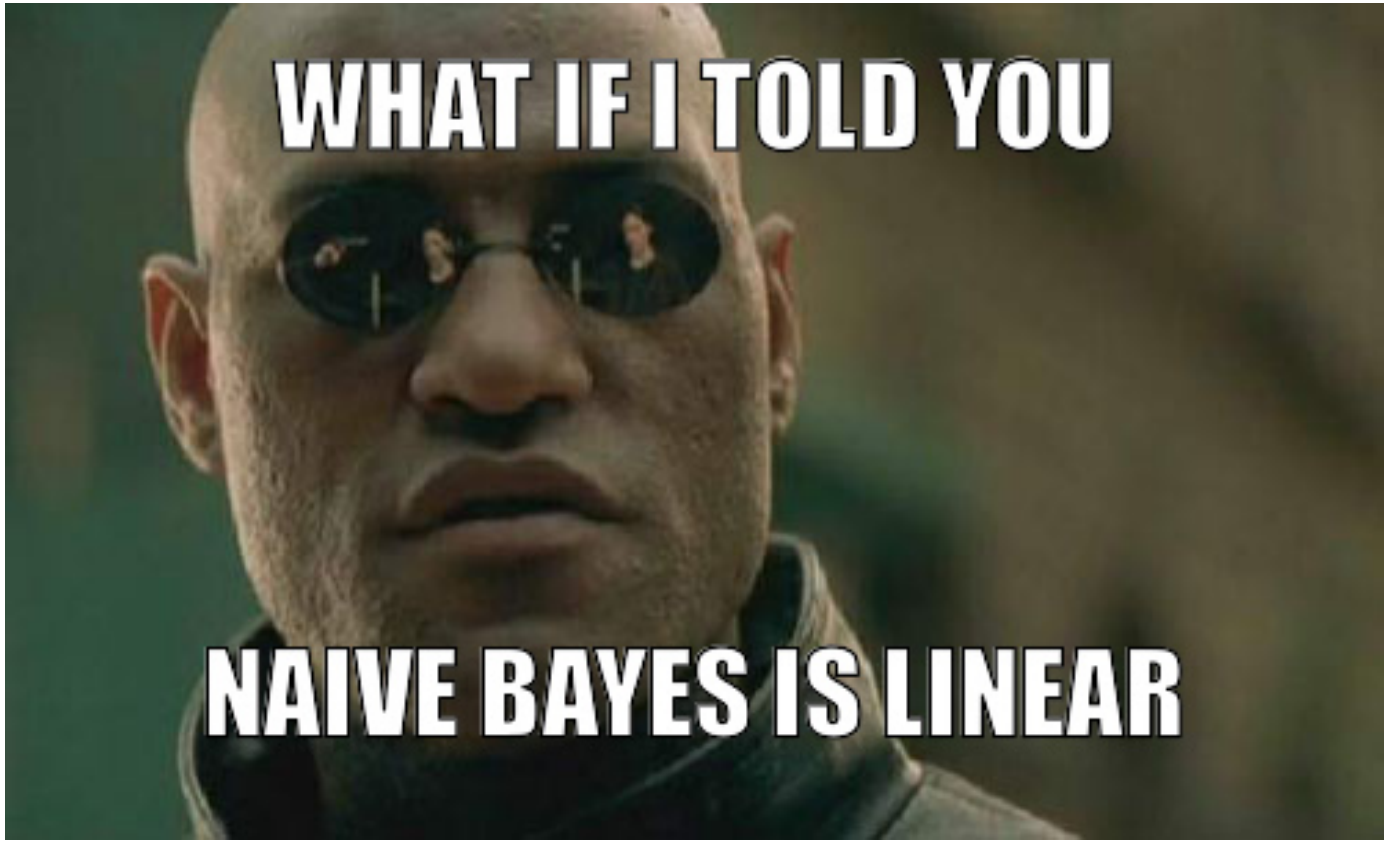
Regresja liniowa – podsumowanie

Regresja liniowa to najprostszy możliwy model:

- im czegoś więcej na wejściu, tym proporcjonalnie (troszkę) więcej/mniej na wyjściu
- nic prostszego nie da się wymyślić (funkcja stała?? LOL)
- niestety model liniowy czasami kompletnie nie ma sensu (np. wzrost człowieka w stosunku do wieku)

To teraz do regresji logistycznej, prowadzą tam dwie drogi:

1. od naiwnego Bayesa
2. od klasyfikacji liniowej


WHAT IF I TOLD YOU
NAIVE BAYES IS LINEAR

Policzmy logarytm szans, że tekst q należy do klasy c (np. że jest spamem). (Szanse to stosunek prawdopodobieństwa zajścia czegoś do prawdopodobieństwa niezajścia). Oznaczmy przez x_i liczbę wystąpień termu t_i w tekście q, teraz:

$$\log \frac{P(c|q)}{P(\bar{c}|q)} = \log \frac{P(c)P(q|c)}{P(\bar{c})P(q|\bar{c})} = \log \frac{P(c)\prod_i P(t_i|c)^{x_i}}{P(\bar{c})\prod_i P(t_i|\bar{c})^{x_i}} =$$

$$= \log(\frac{P(c)}{P(\bar{c})}\prod_i \frac{P(t_i|c)^{x_i}}{P(t_i|\bar{c})^{x_i}}) = \log(\frac{P(c)}{P(\bar{c})}\prod_i(\frac{P(t_i|c)}{P(t_i|\bar{c})})^{x_i}) =$$

$$= \log \frac{P(c)}{P(\bar{c})} + \sum_i x_i \log \frac{P(t_i|c)}{P(t_i|\bar{c})}$$

od naiwnego Bayesa do regresji logistycznej (cd.)

. . . weźmy teraz $w_0 = \log \frac{P(c)}{P(\bar{c})}$, $w_i = \log \frac{P(t_i|c)}{P(t_i|\bar{c})}$ i mamy:

$$\log \frac{P(c|q)}{P(\bar{c}|q)} = w_0 + \sum_i w_i x_i = w_0 + w_1 x_1 + \cdots + w_k x_k$$

A zatem sprowadziliśmy naiwnego Bayesa do liniowej formuły!

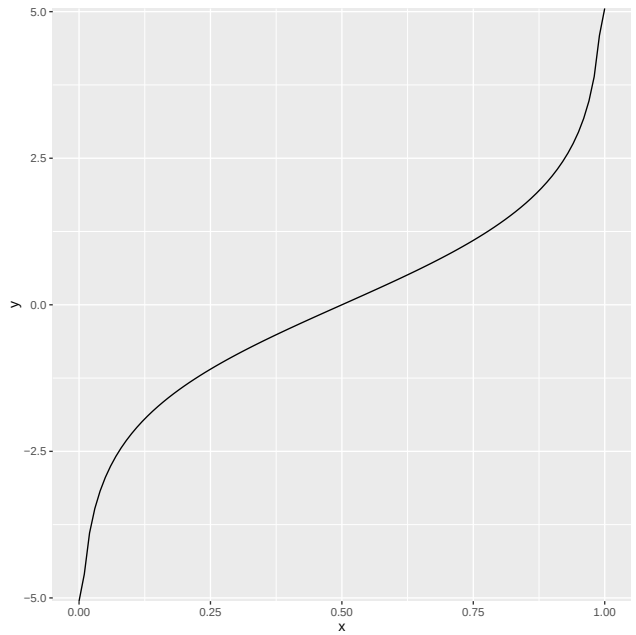
(Wagi nie są odbierane „optymalnie”, lecz na podstawie idiot-modelu, który w praktyce działa całkiem nieźle.)
„Optymalne” dobieranie wag to regresja logistyczna!

Jeszcze popatrzmy na lewą stronę, logarytm szans:

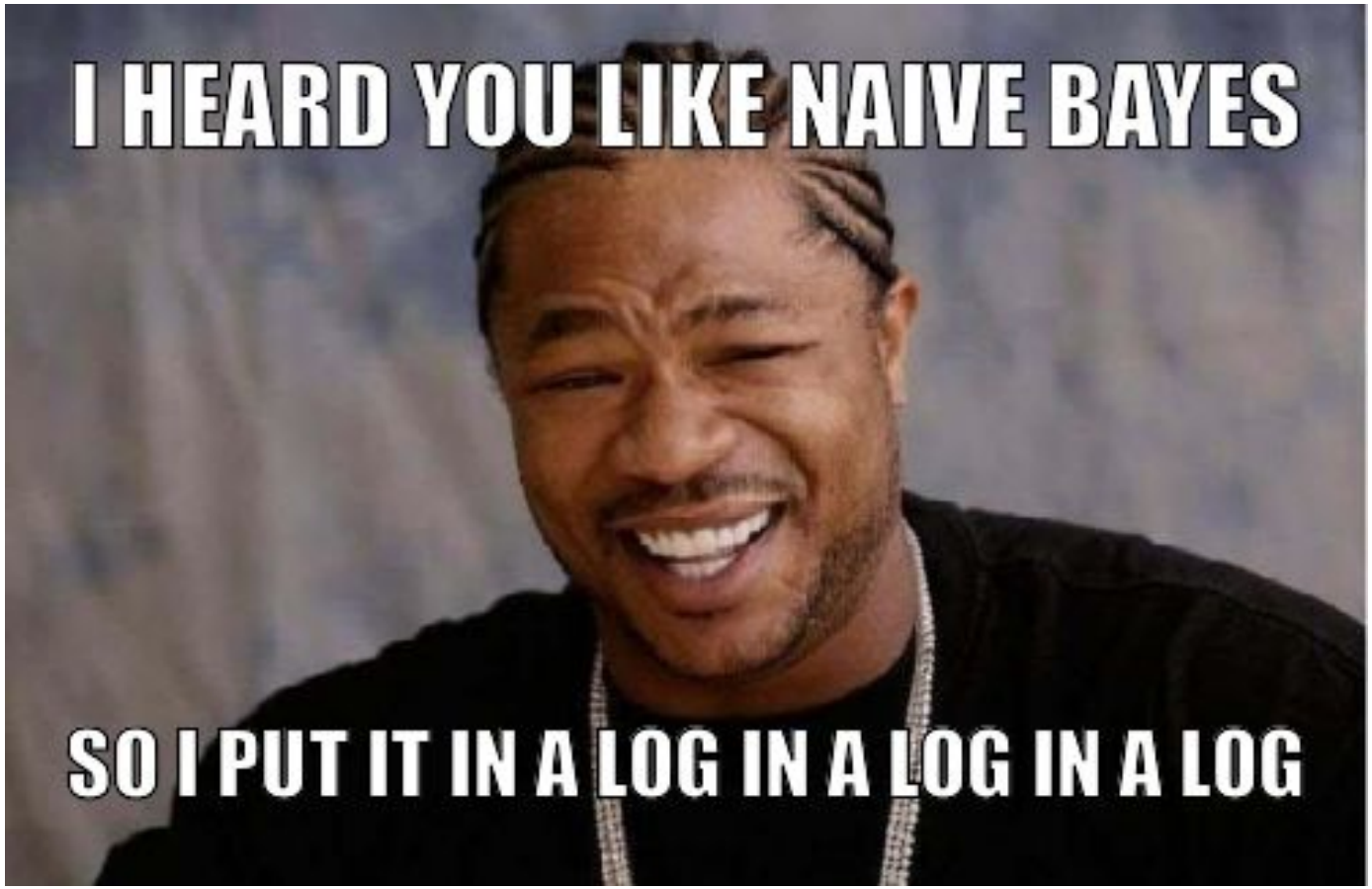
$$\log \frac{P(c|q)}{P(\bar{c}|q)} = \log \frac{P(c|q)}{1 - P(c|q)} = \text{logit}(P(c|q))$$

gdzie „logit” to funkcja logitowa:

$$\text{logit}(x) = \log \frac{x}{1 - x}$$

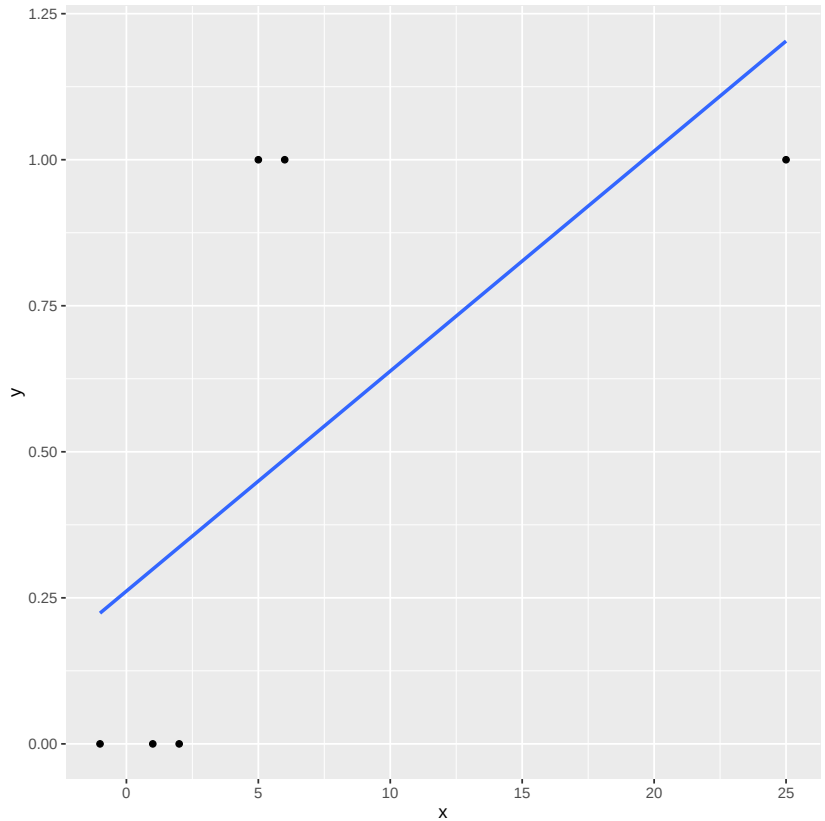


Funkcja logitowa rozciąga wartości $[0, 1]$ (np. prawdopodobieństwa) do $[-\inf, +\inf]$.


I HEARD YOU LIKE NAIVE BAYES
SO I PUT IT IN A LOG IN A LOG IN A LOG

od regresji liniowej do logistycznej

No to spróbujmy inaczej.
Dlaczego nie zastosować regresji liniowej do klasyfikacji?
Powiedzmy, że $y = 1$ to spam, a $y = 0$ — nie-spam...
NIE, to jest głupie! Regresja liniowa nie „lubi" ograniczonych wartości y



od regresji liniowej do logistycznej

1. Przepuścić prawdopodobieństwa przez logit.
2. Zastosować regresję liniową, by ustalić wagi w_i.
3. Odgadywać przez zastosowanie odwróconego logitu.

od regresji liniowej do logistycznej

1. Przepuścić prawdopodobieństwa przez logit.
2. Zastosować regresję liniową, by ustalić wagi w_i.
3. Odgadywać przez zastosowanie odwróconego logitu.
4. PROFIT!

od regresji liniowej do logistycznej

1. Przepuścić prawdopodobieństwa przez logit.
2. Zastosować regresję liniową, by ustalić wagi w_i.
3. Odgadywać przez zastosowanie odwróconego logitu.
4. PROFIT!

Nie tak szybko! Niestety wszystkie przykłady uczące mają albo $y = 0$, albo $y = 1$, a zatem albo $\text{logit}(y) = -\inf$, albo $\text{logit}(y) = +\inf$.
Normalna regresja liniowa uczona na wartościach nieskończonych – niedobrze...

od regresji liniowej do logistycznej

Przedefiniujmy problem

- nasz klasyfikator będzie zwracał prawdopodobieństwo, nie wartości 0/1
  - w sumie słusznie, nie jesteśmy Duchem Św., żeby twierdzić coś na pewno jest albo nie jest
- minimalizujemy log-loss (nie błąd kwadratowy)

$$\text{logloss} = -\frac{1}{n}\sum_{i=1}^{n}(y_i \log(p_i) + (1 - y_i)\log(1 - p_i))$$

  - jeśli chcesz zwracać $p_i = 0$ albo $p_i = 1$, to radzę być pewnym!

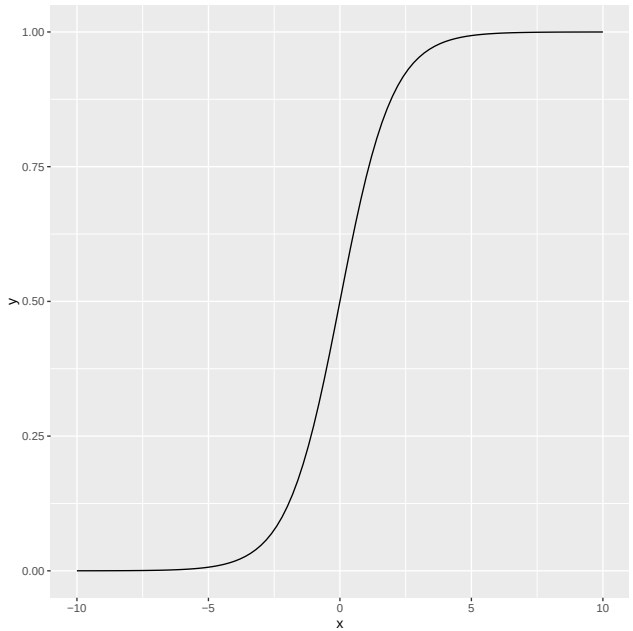
od regresji liniowej do logistycznej

Jednostka dokonująca regresji logistycznej:

$$\hat{p} = \text{logit}^{-1}(w_0 + \sum_{i=1}^{k} w_i x_i)$$

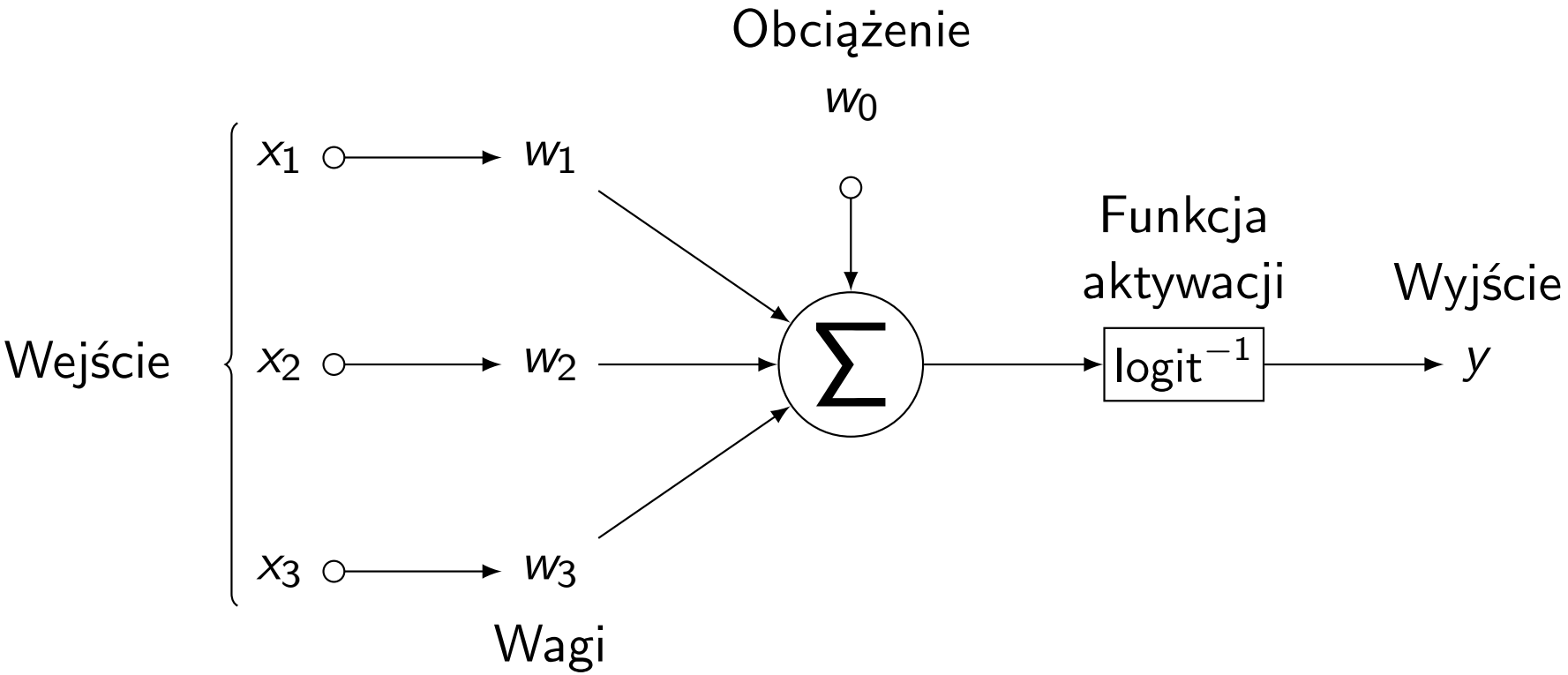
Czyli: najpierw liniowo, a potem nieliniowo.

A tak wygląda logit^{-1}:



(Ściska całą oś w przedział $(0, 1)$.)

od regresji liniowej do logistycznej



od regresji liniowej do logistycznej

A jak nauczyć się wag z przykładów?
Wzoru (z odwracaniem macierzy) podanego wcześniej nie da się zastosować (nieskończoności!)

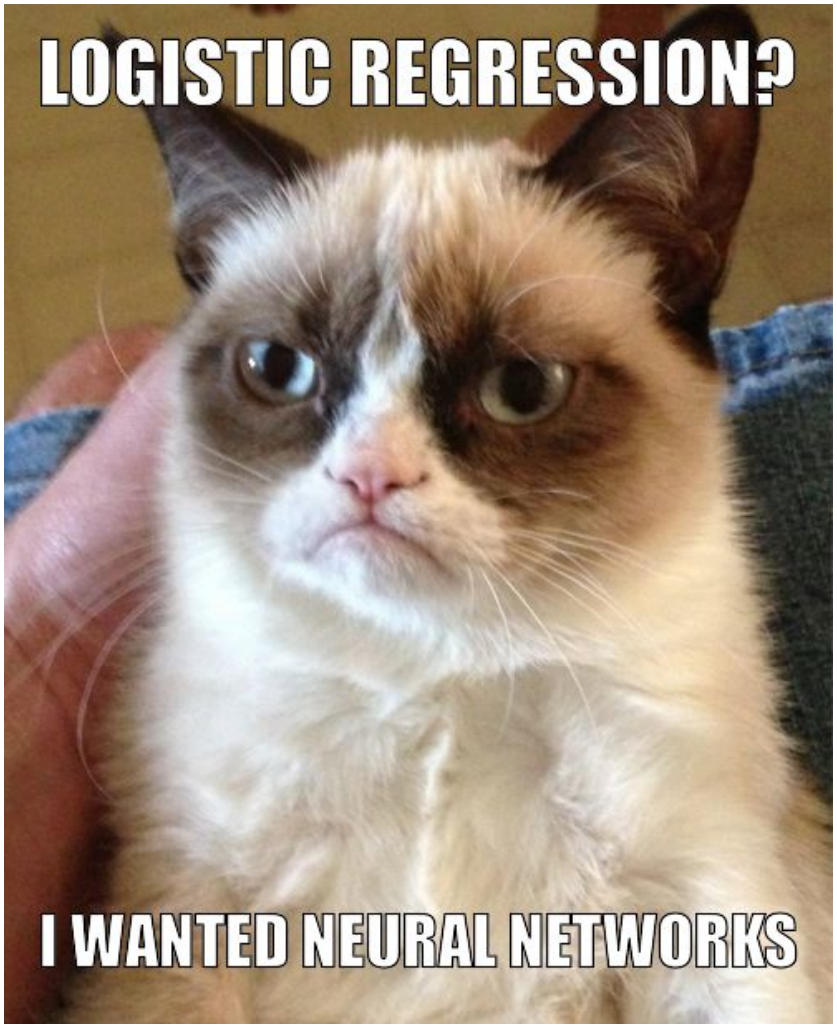
A jak nauczyć się wag z przykładów?
Wzoru (z odwracaniem macierzy) podanego wcześniej nie da się zastosować (nieskończoności!)

Metoda gradientu prostego

Schodź wzdłuż lokalnego spadku funkcji błędu.

Regresja liniowa – uczenie

1: Zacznij od byle jakich wag w_i (np. wylosuj)
2: **repeat**
3: Weź losowy przykład uczący $x_1,\ldots,x_n$, y.
4: Oblicz wyjście $\hat{y}$ na podstawie $x_1,\ldots,x_n$.
5: Oblicz funkcję błędu między y a $\hat{y}$.
6: Zmodyfikuj lekko wagi w_i w kierunku spadku funkcji błędu.
7: **until** Błąd stanie się wystarczająco mały


LOGISTIC REGRESSION?
I WANTED NEURAL NETWORKS

Zaraz, zaraz...

Regresja logistyczna sama w sobie też jest potężnym narzędziem. W praktyce polecam Vowpal Wabbit:

- bardzo szybki
- potrafi przemielić miliardy (!) cech
- przykład: `http://gonito.net/challenge/petite-difference-challenge`

Softmax – uogólnienie regresja logistycznej na przypadek wieloklasowy.

- dla każdej klasy c mamy osobny wektor wag $\mathbf{w}^c$
- wynik dla każdej klasy możemy rozpisać jako iloczyn skalarny $\mathbf{w}^c\mathbf{x}$
- wyniki tych iloczynów mogą być dowolnymi wartościami, jak z nich zrobić prawdopodobieństwa klas?
- tak:

$$P(c|\mathbf{x}) = \frac{e^{\mathbf{w}^c\mathbf{x}}}{\sum_{c'} e^{\mathbf{w}^{c'}\mathbf{x}}}$$

Część III

Sztuczne sieci neuronowe

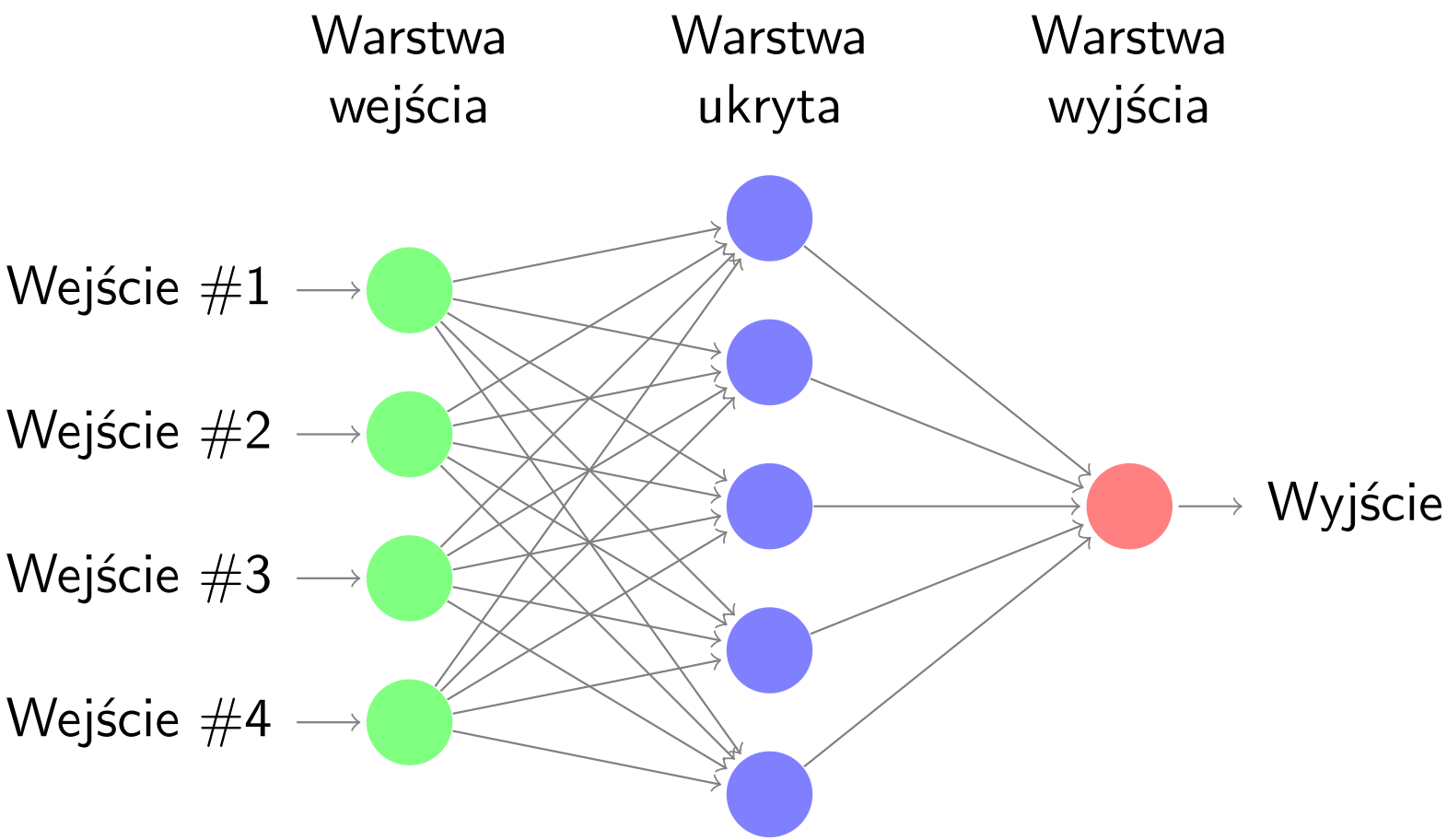
Regresja logistyczna wszystkiego nie załatwi.

Sieci?

A dlaczego nie zrobić sieci jednostek liniowych (zamiast logistycznych)?


YOU SAY A COMBINATION OF LINEAR UNITS
IS ALSO LINEAR

Sztuczna sieć neuronowa



Wyliczanie wyjścia dla zadanego wejścia

- warstwa po warstwie (wprzód) rób regresję logistyczną

Sztuczna sieć neuronowa (cd.)

- jednostki w warstwach ukrytych i wyjściowej realizują regresję logistyczną
- (jednostki w warstwie wejściowej tylko wpuszczają wejście)
- warstw ukrytych może być więcej niż jedna (5, 6, 7, . . .)
- jednostek może być tysiące (albo i więcej. . .)

- (dużo, dużo mniej niż w mózgu)

Propagacja wsteczna (*back propagation*)

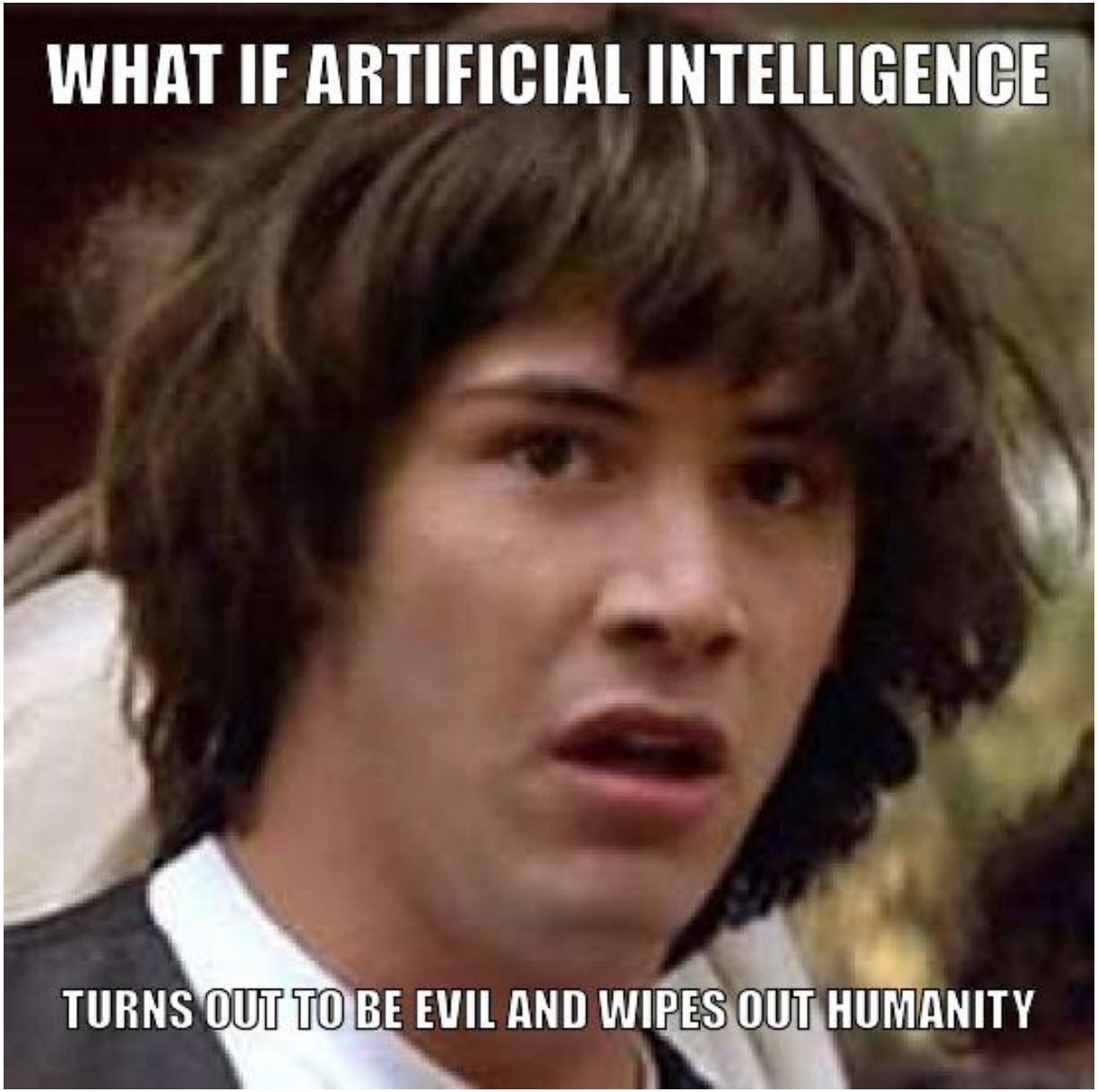
1: Zacznij od byle jakich wag w_i (np. wylosuj)
2: **repeat**
3: Weź losowy przykład uczący $x_1, \ldots, x_n$, y.
4: **for all** Dla każdej warstwy od wejściowej do wyjściowej **do**
5: Oblicz wyjście (wprzód) na podstawie sygnału na wejściu
6: **end for**
7: **for all** Dla każdej warstwy od wyjściowej do wejściowej **do**
8: Zmodyfikuj lekko wagi w kierunku spadku funkcji błędu (różnicy między oczekiwanym a rzeczywistym wyjściem z danej warstwy).
9: **end for**
10: **until** Błąd stanie się wystarczająco mały

Sztuczna sieć neuronowa – intuicje

- uczenie sieci neuronowej to rodzaj *miękkiego* programowania
- regresja logistyczna to rodzaj instrukcji warunkowej
  - **if** $x > 28.11$ **and** $x > -7.02$ **then** ...
  - progi ustalane są automatycznie...
  - ...często takie, których człowiek nigdy by nie wymyślił
- człowiek „programuje”, nadając ogólny strukturę sieci
  - pętle (sieci rekurencyjne)
  - wykonywanie w wielu miejscach tej samej procedury (sieci splotowe)
- wsteczna propagacja wypełnia wartości liczbowe

Część IV

Praktyka


WHAT IF ARTIFICIAL INTELLIGENCE
TURNS OUT TO BE EVIL AND WIPES OUT HUMANITY

Sztuczne sieci neuronowe

- nie przejmą władzy
- nie osiągną samoświadomości
- nie będą knuły
- nie wyślą terminatora

Ale to NIE moda. Sieci neuronowe rozwiązują realne problemy i parę (?) zawodów przez nie zniknie. (Ktoś pamięta zecerów?)

- program AlphaGo (oparty na sieciach neuronowych) pokonuje najlepszego goistę na świecie, Lee Sedola
- sieci neuronowe zmniejszają stopę błędów systemów rozpoznawania mowy o kilka procent
- neuronowe tłumaczenie automatyczne okazuje się lepsze od poprzedniej generacji translatorów (`http://104.131.78.120/`)
- sieci neuronowe podpisują zdjęcia (`http://googleresearch.blogspot.com/2014/11/a-picture-is-worth-thousand-coherent.html`)
- nowy kierunek w sztuce – incepcjonizm

Sieci neuronowe – zrób to sam

F4A88AA
NVIDIA Tesla K40 Workstation Coprocessor

Part Number: F4A88AA
Ean: 0888182122914

Detaliczna **cena** producenta: ~~28 053,20 zł (6 524,00 €)~~
Nasza **cena** (netto): 25 247,88 zł (5 871,60 €)
Nasza cena (brutto): 31 054,89 zł

Dodaj do zapytania ofertowego +

I to nie o przechwałki, kto ma większą. Może chodzić o różnice pomiędzy 1 tygodniem a 2 miesiącami.

Ale do eksperymentów z sieciami wystarczy np. nVidia 960 (ok. 1000 zł).

narzędzia otwartoźródłowe

- numPy
- Theano
- TensorFlow
- Keras