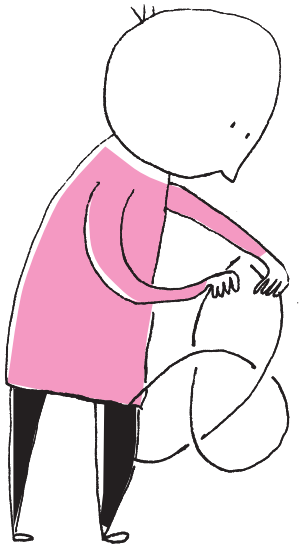
Głębokie uczenie maszyn

*Paweł GORA**

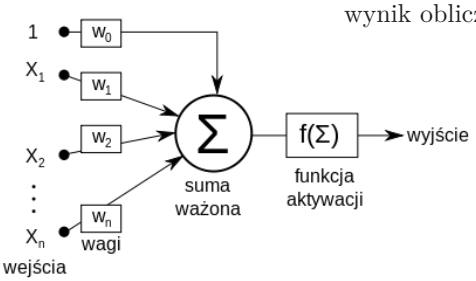
*Instytut Informatyki, Wydział Matematyki, Informatyki i Mechaniki, Uniwersytet Warszawski

Czy zastanawialiście się kiedyś nad tym, czym jest inteligencja? Większość definicji zakłada posiadanie zdolności do rozumowania, uczenia się, adaptacji do zmieniającego się otoczenia. Najczęściej mówi się o inteligencji ludzi i zwierząt – organizmów mających mózg, a więc organ biologiczny. Czy inteligencja musi być jednak domeną jedynie istot, które uważamy obecnie za żywe? Wielu Czytelników słyszało zapewne również o inteligencji „sztucznej", określanej tak dlatego, że nie występuje ona w sposób naturalny w przyrodzie, ale w emergentny sposób pojawia się w maszynach i algorytmach tworzonych dzięki innej inteligencji, np. człowieka.



W ostatnich latach nastąpił rozkwit fascynującej dziedziny nauki, jaką jest sztuczna inteligencja. Coraz częściej słyszymy o tym, że programy komputerowe wygrywają z ludźmi (i to mistrzami świata!) w gry, które wymagają dużego wysiłku intelektualnego (np. szachy, Go). Programy potrafią już rozpoznawać obiekty na zdjęciach i jeździć samochodem lepiej niż większość ludzi, rozpoznają mowę i tłumaczą tekst z jednego języka na drugi, wykazują się też kreatywnością i tworzą nowe dzieła na podstawie widzianych wcześniej obrazów lub tekstów.

Jeden z przełomów w dziedzinie uczenia maszyn i nadawania im (sztucznej) inteligencji związany jest w ostatnich latach z głębokimi sieciami neuronowymi. Słowo „neuronowymi" już powinno dać pewne skojarzenia z umysłem, w końcu nasz mózg składa się właśnie z dużej liczby neuronów, szacowanej na 100 miliardów. Faktycznie, budowa i działanie sztucznych sieci neuronowych zainspirowane zostały badaniami nad ludzkim mózgiem. W pewnym sensie sieci neuronowe zrodziły się właśnie z motywacji, aby maszyny mogły działanie mózgu naśladować. Ich podstawowym budulcem są sztuczne neurony, które otrzymują na wejściu pewne dane i przetwarzają je w określony sposób, przekazując na wyjściu wynik (rys. 1). Neurony mogą być połączone tak, że wynik obliczeń jednego z nich może stanowić sygnał wejściowy do innego.



Rys. 1. Neuron McCullocha-Pittsa – podstawowy budulec sieci neuronowej – źródło: `https://pl.wikipedia.org/wiki/Neuron_McCullocha-Pittsa`

W ten sposób może współdziałać wiele warstw neuronów, przetwarzając dane otrzymane na wejściu aż do obliczenia wyniku zgodnie z określonymi regułami (funkcjami aktywacji) w neuronach oraz wartościami parametrów ich połączeń. Zakładając, że neuron otrzymuje na wejściu dane $x_1, x_2, \ldots, x_n$, które przekazywane są przez połączenia wejściowe z przypisanymi do nich wagami w_1, w_2, …, w_n, wyjście obliczane jest za pomocą reguły:

$$y = f\Big(w_0 + \sum_{i=1}^{n} w_i \cdot x_i\Big),$$

gdzie f jest właśnie funkcją aktywacji, a w_0 jest dodatkowym parametrem neuronu ułatwiającym proces uczenia. Często jako funkcję aktywacji stosuje się funkcję sigmoidalną $\frac{1}{1+e^{-x}}$ oraz funkcję ReLU, czyli $\max(0, x)$.

Dane zazwyczaj przetwarzane są w jednym kierunku: od danych wejściowych (warstwa wejściowa), przez warstwy ukryte, aż do obliczenia wyniku (warstwa wyjściowa). Możliwe są również inne konfiguracje, jak np. w rekurencyjnych sieciach neuronowych, które znajdują zastosowania np. do analizy szeregów czasowych, rozpoznawania mowy i tłumaczeń tekstów. W takich sieciach połączenia pomiędzy neuronami z różnych warstw mogą tworzyć cykle.

Uczenie to taki proces dostrajania wag połączeń między neuronami, który daje najlepsze wyniki. W jaki sposób uczą się sieci neuronowe? Algorytmy uczenia (nie tylko sieci neuronowych) można podzielić na 3 grupy:

- uczenie z nadzorem (*supervised learning*),
- uczenie bez nadzoru (*unsupervised learning*),
- uczenie ze wzmocnieniem (*reinforcement learning*).

Rozwiązanie zadania M 1552.
Będziemy szukali takich rozwiązań, dla których $4^{a^2} = x^2$ oraz $4^{b^2} = 2x$, gdzie x jest dodatnią liczbą całkowitą; wówczas liczba

$$4^{a^2} + 4^{b^2} + 1 = (x+1)^2$$

będzie kwadratem liczby całkowitej.

Z powyższych równości wynika, że

$$x = 2^{a^2} = 2^{2b^2-1},$$

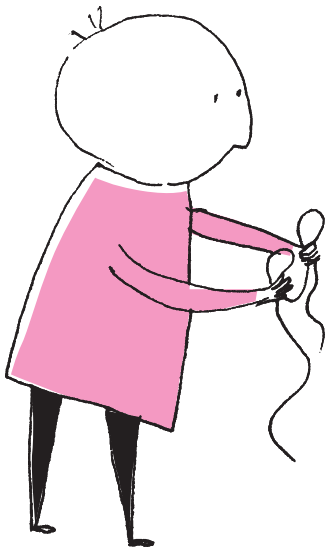
więc rozważane liczby mają szukaną postać, o ile tylko liczby a, b są powiązane zależnością

$$a^2 - 2b^2 = -1.$$

Powyższe znane równanie diofantyczne (tzw. *ujemne równanie Pella*) ma nieskończenie wiele rozwiązań całkowitych (a_n, b_n) danych przez

$$(1+\sqrt{2})^{2n-1} = a_n + b_n\sqrt{2}$$

dla $n \geqslant 1$.



Rozwiązanie zadania M 1554.
Udowodnimy, że $c = 1$.

Zauważmy, że jeżeli $k = n$ i wszystkie połączenia są realizowane między rozłącznymi parami miast, to każda podróż może być złożona z tylko jednego lotu. Stąd $c \leqslant 1$.

Rozważmy następujące doświadczenie trwające k dni. W każdym z miast umieśćmy na początku jednego turystę. Pierwszego dnia zaplanujmy loty pary turystów, którzy znajdują się w miastach, między którymi kursuje najtańszy lot; drugiego dnia – pary turystów, którzy znajdują się w miastach, między którymi jest drugi najtańszy lot itd.

Po k dniach każdy turysta będzie już po podróży mającej tę własność, że każdy kolejny lot był droższy od poprzedniego. Ponadto łącznie wszyscy turyści wykonają w sumie $2k$ lotów, gdyż każdego dnia dokładnie dwóch wsiadło do samolotu. Stąd wniosek, że średnia długość podróży wśród wszystkich turystów jest równa $\frac{2n}{2k} = \frac{n}{k}$ lotów. W związku z tym istnieje turysta, którego wycieczka jest złożona z co najmniej tylu lotów, a zatem $c \geqslant 1$.

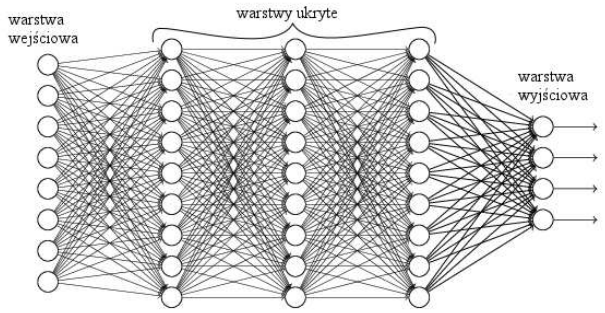
Największe sukcesy przez wiele lat święciło uczenie z nadzorem. W metodzie tej dany jest zbiór par *wejście-wynik* i celem jest znalezienie funkcji obliczanej przez sieć neuronową, która dla danych wejściowych dawałaby na wyjściu wyniki minimalizujące pewną funkcję kosztu (*loss function*). Funkcja kosztu powinna uwzględniać wyniki obliczane przez sieć neuronową oraz prawidłowe wyniki dla danych wejściowych, może być nią np. błąd średniokwadratowy, w praktyce odpowiednie dobranie funkcji kosztu wpływa istotnie na proces uczenia sieci. Sieć trenowana jest na pewnym podzbiorze par wejście-wynik – *zbiorze treningowym*, ale skuteczność sieci i algorytmu uczenia testowane są już na osobnym podzbiorze, *zbiorze testowym*, aby sprawdzić, jak dobrze sieć radzi sobie na nowych danych, na których nie była wcześniej trenowana (od algorytmów określanych jako *inteligentne* oczekiwalibyśmy właśnie, aby potrafiły radzić sobie z nieznanymi wcześniej danymi i sytuacjami!). Często bierze się pod uwagę jeszcze dodatkowy podzbiór, rozłączny z treningowym i testowym – *zbiór walidacyjny*, który ma pomóc w wyborze najlepszej architektury sieci oraz najlepszych algorytmów i wartości parametrów odpowiedzialnych za jej uczenie (tzw. *hiperparametrów*).

Samo trenowanie sieci neuronowej realizuje się najczęściej za pomocą *spadku gradientowego* i algorytmu *propagacji wstecznej* – wagi połączeń pomiędzy neuronami są modyfikowane tak, aby zmniejszać popełniany przez sieć błąd, czyli minimalizować funkcję kosztu, z tego powodu funkcja ta powinna być różniczkowalna względem wag połączeń. Jest to typowy schemat algorytmów uczenia (nie tylko sieci neuronowych), poszczególne algorytmy mogą się różnić pewnymi szczegółami, ale w większości z nich występuje pewien zbiór treningowy, dla którego algorytm dobiera ustawienia minimalizujące pewną funkcję kosztu.

Trochę inaczej sytuacja wygląda w przypadku uczenia bez nadzoru – dla danych wejściowych w zbiorze treningowym może bowiem nie być prawidłowego wyniku. Mimo to okazuje się, że odpowiednio konstruując sieci neuronowe, można nauczyć je wykrywania w danych pewnych wzorców (aby nazwać te wzorce, może być potrzebna ingerencja człowieka lub nieduży zbiór danych z wynikami – tak jak przy uczeniu z nadzorem). Typowym przykładem uczenia bez nadzoru jest analiza skupień – grupowanie danych w rozłączne podzbiory. W przypadku uczenia z nadzorem sieć neuronowa otrzymując na wejściu zbiór, w którym są zdjęcia pieszych i rowerzystów, dla każdego zdjęcia będzie w stanie powiedzieć, czy jest na nim pieszy czy rowerzysta. W przypadku uczenia bez nadzoru sieć neuronowa po przeanalizowaniu całego zbioru danych może powiedzieć: na zdjęciach mamy dwie grupy obiektów, obiekty z danej grupy są do siebie bardzo podobne, ale istotnie różnią się od obiektów z drugiej grupy.

Uczenie bez nadzoru można więc porównać z sytuacją, gdy małe dziecko obserwuje otaczający je świat, nie ma nauczyciela, który by mu ten świat tłumaczył (lub nie jest w stanie tego nauczyciela zrozumieć), a mimo to jest w stanie wykryć pewne wzorce i prawa występujące w świecie, rozpoznać rodziców, jedzenie, zrozumieć grawitację, nauczyć się chodzić. Uczenie z nadzorem można z kolei porównać do sytuacji, w której jest nauczyciel, który pokazuje uczniowi gotowe materiały, pytania i odpowiedzi, zadania i gotowe rozwiązania, a potem oczekuje, że uczeń będzie potrafił rozwiązać zadania trochę inne, do pewnego stopnia podobne do tych przykładowych, ale jednak niespotkane wcześniej. Tak jak w przypadku małych dzieci uczenie bez nadzoru jest typowe dla wieku przedszkolnego, a uczenie z nadzorem jest już typowe dla wieku szkolnego (a więc późniejszego), tak i w przypadku uczenia maszyn czasami stosuje się najpierw uczenie bez nadzoru, aby sieci neuronowe nauczyły się same wykrywać pewne wzorce, których znajomość może potem ułatwić uczenie z nadzorem (jest to przykład *semi-supervised learning* stanowiącego połączenie dwóch wspomnianych podejść).

Trzeci rodzaj uczenia maszyn to tzw. *uczenie ze wzmocnieniem*. W tym przypadku zbiór treningowy nie jest od razu dostępny w całości. Zamiast tego algorytm może być w interakcji ze środowiskiem, w którym się znajduje, wykonując określone akcje na podstawie swoich obserwacji. Każda akcja



Rys. 2. Wielowarstwowa sieć neuronowa – źródło: `http://neuralnetworksanddeeplearning.com/chap5.html`

powoduje przejście do innego stanu i otrzymanie informacji o nagrodzie związanej z wykonaną akcją. Celem algorytmu jest wykonywanie akcji, które spowodują otrzymanie najwyższej nagrody, np. wygranie w grze. Sieci neuronowe mogą być w tym przypadku uczone, jakie akcje wykonywać w danej sytuacji, aby zmaksymalizować swoją nagrodę (takie podejście stosowano m.in. w programie AlphaGo, który w 2016 wygrał w Go z jednym z najlepszych ludzi w tej grze – Lee Sedolem). Tego typu metodę nauczania można z kolei porównać do nauczyciela, który pozwala uczniowi od początku samodzielnie wykonywać działania i podejmować decyzje, dając mu za każdym razem informację zwrotną na temat podejmowanych działań.

Wspomniałem o tym, że sieci neuronowe mogą mieć wiele warstw. Mogą być zorganizowane w sposób hierarchiczny tak, że w kolejnych warstwach następuje przetwarzanie danych z niższego poziomu i wyniki przekazywane są do następnej warstwy (rys. 2). W ten sposób od danych wejściowych, niskopoziomowych, relatywnie prostych, można stopniowo otrzymywać coraz bardziej złożone, wysokopoziomowe informacje. Okazuje się, że w trakcie procesu trenowania sieć neuronowa może nauczyć się pewnych złożonych własności występujących w danych wejściowych, np. pewne neurony aktywują się (tj. funkcja aktywacji przyjmuje wartości dużo większe od zera), gdy w danych występują pewne określone wzorce. Tego typu metody uczenia nazywane są *deep learningiem*, czyli *głębokim uczeniem* – głębokim, bo uczone są sieci mające wiele warstw, a elementy w poszczególnych warstwach uczą się wykrywania i reprezentowania pewnych nieoczywistych (*głębokich*), złożonych własności danych, które zostały użyte w procesie uczenia, w kolejnych warstwach składając w pewnym sensie te złożone własności z elementów prostszych. Jest to jedna z najważniejszych własności decydujących o sukcesie deep learningu – w większości innych metod sztucznej inteligencji potrzebna jest często dość duża wiedza dziedzinowa; to człowiek-naukowiec jest odpowiedzialny za właściwe dobranie atrybutów obiektów występujących w danych, aby metoda dobrze działała. Algorytmy deep learningu mogą same nauczyć się najważniejszych złożonych cech, znacznie przyspieszając pracę naukowców – *nauczycieli maszyn*, oszczędzając ich czas, redukując szanse na popełnienie błędu (poprawiając też w ten sposób skuteczność!), umożliwiając stosowanie tych rozwiązań do wielu rzeczywistych problemów. Dzięki temu algorytmy sztucznej inteligencji do pewnego stopnia zaczynają zastępować człowieka w... projektowaniu algorytmów sztucznej inteligencji! Często potrzeba jednak znacznie więcej danych, aby sieć neuronowa mogła się wystarczająco dobrze potrzebnych cech nauczyć, potrzebna jest również większa moc obliczeniowa, dlatego obliczenia w sieciach neuronowych przyspieszane są obecnie za pomocą kart graficznych lub klastrów wielu maszyn.

Warto wspomnieć o tym, że w metodach deep learningu elementy w poszczególnych warstwach nie muszą być sztucznymi neuronami, mogą być to również inne transformacje liniowe lub nieliniowe. Głębokie sieci neuronowe nie są więc jedyną realizacją deep learningu i strategii poszukiwania istotnych cech, ale w ostatnich latach okazały się niezwykle skuteczne i znalazły zastosowanie w wielu problemach związanych z rozpoznawaniem złożonych struktur. Na przykład, istnieją już sieci neuronowe potrafiące rozpoznawać obiekty na zdjęciach z większą skutecznością, niż robią to obecnie ludzie (takie metody stosowane są, na przykład, w sensorach pojazdów autonomicznych). Neurony w kolejnych warstwach sieci w procesie trenowania na rzeczywistych obrazach uczą się reprezentowania pewnych cech występujących w tych obrazach. Najniższe warstwy otrzymują na wejściu zwykłe piksele, a kolejne warstwy przetwarzają je do postaci linii, łuków, a potem coraz bardziej skomplikowanych struktur. Z podejściem tym może być jednak pewien problem – w przypadku rozpoznawania złożonych struktur (np. twarzy) na zdjęciach o dużej rozdzielczości potrzebna sieć neuronowa musiałaby mieć wiele warstw i neuronów, a jej trenowanie byłoby bardzo czasochłonnym procesem. Pojawiły się jednak pomysły na specjalne architektury sieci neuronowych, które mogą się sprawdzać bardzo dobrze w takich przypadkach, są to tzw. konwolucyjne (splotowe) sieci neuronowe, w których układ połączeń pomiędzy neuronami inspirowany jest budową narządu wzroku u ludzi i zwierząt. Neurony tworzą wiele warstw, z których każda ma pewne specyficzne zastosowanie. Neurony w warstwie wejściowej grupowane są w obszary o takim samym rozmiarze (np. 5×5 pikseli), neurony z tego samego obszaru połączone są z neuronem z warstwy wyższej, tworząc tzw. filtr, który w trakcie uczenia sieci odpowiada za wykrywanie pewnej cechy obrazu, np. występowanie kreski lub łuku. Ponieważ dana cecha może wystąpić w dowolnym obszarze obrazka i w każdym obszarze chcielibyśmy ją wykrywać w taki sam sposób, więc filtr o takiej samej strukturze połączeń i z takimi samymi wagami powinien występować dla każdego możliwego obszaru. Jeśli mielibyśmy obrazek o wymiarach $m \times n$, a chcielibyśmy skonstruować filtr obejmujący obszar o rozmiarze $k \times l$ (dla $k < m$ i $l < n$), to potrzebowalibyśmy co najmniej $(m - k + 1) \times (n - l + 1)$ neuronów filtrujących daną cechę (czasami rozszerza się obrazek o piksele zerowe na brzegach, np. aby zachować taką samą liczbę neuronów z warstwy wejściowej i neuronów filtrujących). Neurony w danym filtrze współdzielą odpowiadające sobie wagi

(chcemy w końcu, aby każdy neuron z filtra działał tak samo), co ułatwia znacznie proces trenowania sieci. Filtrów wykrywających pewne niskopoziomowe cechy obrazka może być dużo, wszystkie one tworzą tzw. *warstwę konwolucyjną*. Neurony z warstwy konwolucyjnej połączone są następnie z neuronami z funkcją aktywacji ReLU (warstwa ReLU), które ułatwiają trenowanie sieci, a neurony z warstwy ReLU połączone są z neuronami z warstwy *pooling*, które pomagają wydobyć najważniejsze cechy z wcześniejszych warstw (np. poprzez wyliczanie maksymalnej lub średniej wartości z grupy sąsiednich neuronów z wcześniejszej warstwy). Warstwy konwolucyjna, ReLU i pooling mogą służyć do wykrycia złożonych wzorców, a do wsparcia ostatecznej klasyfikacji obiektu stosuje się tzw. *warstwę gęstą* (ang. *dense layer* lub *fully connected layer*), w której wszystkie neurony z warstwy wcześniejszej są połączone ze wszystkimi neuronami z warstwy kolejnej. Warstwy te (oraz ewentualnie inne, przystosowane do konkretnego zagadnienia) mogą występować w sieci wielokrotnie, na różnych poziomach, z różnymi konfiguracjami. Zaprojektowanie dobrej sieci neuronowej jest kluczowym czynnikiem decydującym o jej skuteczności i szybkości trenowania i działania.

Obecnie następuje stopniowe przejście informatyki z etapu, w którym komputery są programowane przez człowieka, do etapu, w którym maszyny są przez człowieka uczone. Wszystko wskazuje na to, że jesteśmy na początku wielkiej rewolucji związanej z pojawieniem się na skalę masową sztucznej inteligencji, rozwiązywaniem przez nią coraz bardziej skomplikowanych problemów lepiej niż ludzie i w związku z tym – ze wspomaganiem lub stopniowym wyręczaniem ludzi w wielu wykonywanych przez nich pracach.

Zadania

Redaguje Łukasz BOŻYK

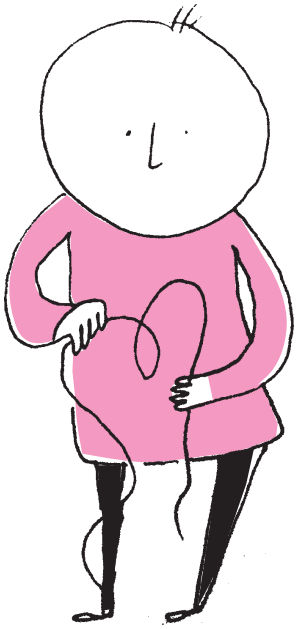
M 1552. Wykazać, że istnieje nieskończenie wiele par liczb całkowitych dodatnich (a, b), dla których liczba

$$4^{a^2} + 4^{b^2} + 1$$

jest kwadratem liczby całkowitej.
Rozwiązanie na str. 2

M 1553. W pewnym kraju jest skończona liczba miast. Między niektórymi parami miast istnieją jednokierunkowe połączenia autobusowe. Sieć komunikacyjna ma tę własność, że nie można zaplanować podróży złożonej z co najmniej jednego kursu, która zaczyna się i kończy w tym samym mieście. Udowodnić, że istnieje miasto, z którego nie można wyjechać autobusem oraz miasto, do którego nie można dojechać.
Rozwiązanie na str. 11

M 1554. W pewnym kraju jest $2n$ miast. Między niektórymi parami miast istnieją dwukierunkowe połączenia lotnicze, których w sumie jest k. Ceny biletów na różnych odcinkach są różne, ale na każdym odcinku cena jest taka sama niezależnie od kierunku lotu. Wyznaczyć najmniejszą stałą c (niezależną od n i k) o tej własności, że zawsze można tak zaplanować podróż złożoną z co najmniej $c \cdot \frac{k}{n}$ przelotów, aby każdy kolejny lot był droższy od poprzedniego.
Rozwiązanie na str. 2



Przygotował Andrzej MAJHOFER

F 943. T.D. Stewart i R.C. Tolman wyznaczyli stosunek ładunku elektronu, e, do jego masy m metodą, w której metaliczne próbki poddawali przyspieszeniom. Wyjaśnij, jak to było możliwe. Przyjmij model swobodnych elektronów w metalu.
Rozwiązanie na str. 10

F 944. Na poziomej, płaskiej elektrodzie metalowej położono jednorodną płaską warstwę dielektryka (izolatora) o grubości d i przenikalności dielektrycznej ε. Na warstwie dielektryka umieszczono kroplę przewodzącej cieczy (elektrolitu) niezwilżającej dielektryka. Jak zmieni się kąt zwilżania θ, gdy do kropli przyłożymy napięcie U względem metalowej elektrody? Napięcia powierzchniowe wynoszą: ciecz-dielektryk γ_{cd}, ciecz-gaz otaczający układ γ_{lg}, a dielektryk-gaz γ_{dg}.
Wskazówka: Napięcie powierzchniowe to energia potrzebna do utworzenia powierzchni rozdziału faz przypadające na jednostkę pola powierzchni. Jest ono również równe sile działającej prostopadle do brzegu takiej powierzchni rozdziału na jednostkę jego długości. Kąt zwilżania to kąt między powierzchniami ciecz-dielektryk i ciecz-gaz.
Rozwiązanie na str. 6