



Vilniaus universitetas
Matematikos ir informatikos fakultetas
Informatikos katedra



Dirbtinis intelektas

prof. dr. Olga Kurasova
Olga.Kurasova@mii.vu.lt

2018

Dirbtinis intelektas. Kas tai?

- Ką žinote ar esate girdėję apie **dirbtinį intelektą**?
- Kur jis **taikomas**? Pateikite pavyzdžių.
- Kaip jis **kuriamas**?
- Gal žinote kokius **dirbtinio intelekto metodus**?
- Ar jau **sukurtas** dirbtinis intelektas?
- Kokios **iškyla pagrindinės problemos** kuriant dirbtinį intelektą?

Dirbtinis intelektas (DI). Kas tai?

- **Intelektas** – tai žmogaus sugebėjimas mąstyti, protas, protingumas.
- **Dirbtinis intelektas** (angl. *artificial intelligence, AI*) – dirbtinai sukurtas intelektas.
- **Dirbtinis intelektas** – tai informatikos mokslo šaka, nagrinėjanti duomenų apdorojimo sistemas, atliekančias paprastai su žmogaus intelektu siejamas funkcijas, tokias kaip samprotavimas, mokymasis ir tobulinimasis.

Dirbtinis intelektas. Kas tai?

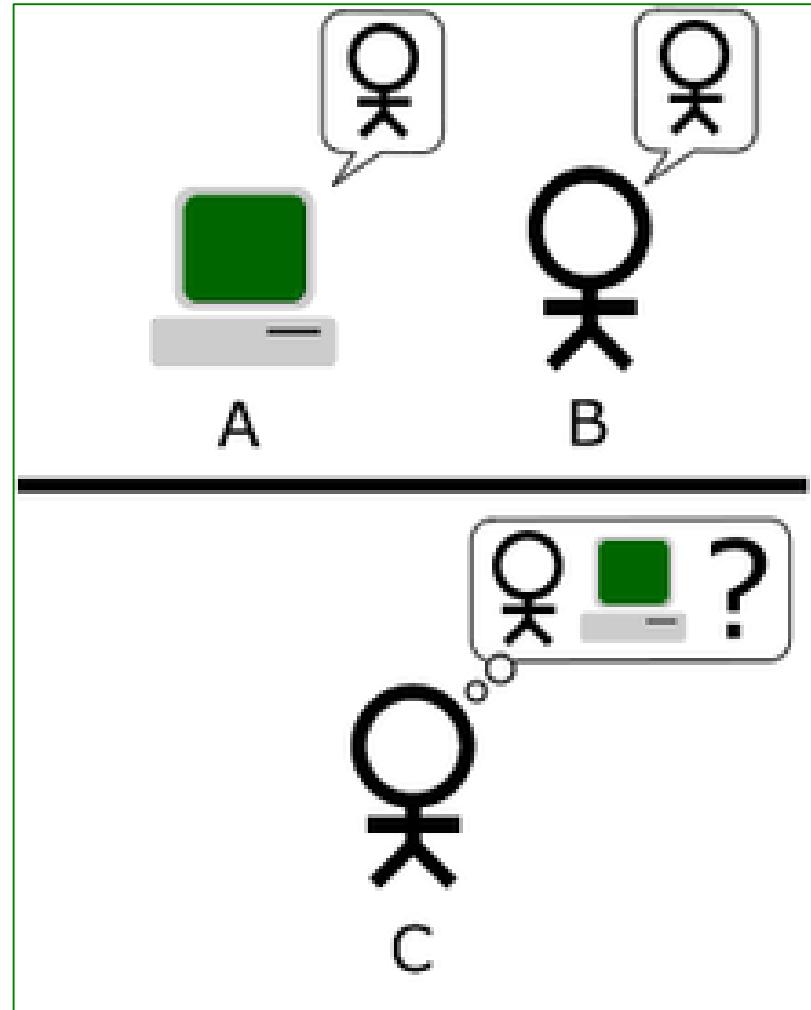
- Dirbtinis intelektas skiriasi nuo įprastų kompiuterinių algoritmų tuo, kad **gali apsimokyti**, ir atlikdamas tą patį veiksma gali elgtis kitaip priklausomai nuo prieš tai atliktu veiksmu.
- Dirbtinio **intelekto tyrimai remiasi** psichologijos ir neurologijos, matematikos ir logikos, komunikacijos teorijos, filosofijos ir lingvistikos mokslų duomenimis.

Tiuringo testas

- **Tiuringo testas** – tai 1950 m. Alano Tiuringo pasiūlytas testas, skirtas išbandyti mašinų sugebėjimą demonstruoti intelektą.
- Testo metu žmogus-teisėjas **natūralia kalba šnekasi** su vienu **žmogumi** ir viena **mašina**. Visi trys eksperimento dalyviai yra izoliuotuose patalpose, kad teisėjas nematytu, kuriuos atsakymus pateikia mašina, o kuriuos žmogus.
- Jei teisėjas pagal atsakymus negali **patikimai atskirti mašinos nuo žmogaus**, sakoma, kad mašina išlaikė testą.
- Kad būtų testuojamas mašinos intelektas, o ne gebėjimas imituoti žmogaus leidžiamus garsus, pokalbis apribojamas **tiktais tekstinėmis žinutėmis**.

Tiuringo testas

A. Turingas teigė, kad kompiuterių galima būtų laikyti mąstančiu, jeigu jis įveiktų testą, per **5 minučių** bendravimą tekstu **įtikindamas 30 %** jį klausinėjančių žmonių, kad jie **bendrauja su gyvu žmogumi**.



Tiuringo testo išlaikymas

- Kompiuterinė programa „**Eugene Gootsman**“, sukurta Rusijoje, apsimetanti 13 metų berniuku iš Ukrainos, **2014-06-07** sėkmingai įtikino 33 % teisėjų, kad jie bendrauja su žmogumi.
- Ši programa tapo **pirmuoju kompiuteriu, įveikusiu** Tiuringo testą.
- Renginys vyko Karališkojoje Mokslų Draugijoje **Londone**.
- Akademiniė bendruomenė įspėja, kad technologija gali būti panaudota kibernetiniuose nusikaltimuose.

Dirbtinio intelekto pirmieji žingsniai

- Pradžia **1943** m. kai W. S. McCulloch ir W. Pitts pasiūlė dirbtinio neurono modelį.
- **1950** m. A. Tiuringas pasiūlė jo vardu pavadintą testą.
- Dirbtinio intelekto termino formaliai pradžia **1956** m. įmonės IBM organizuotoje konferencijoje.
- **1958** m. Rosenblatt sukūrė perceptroną (tiesioginio sklidimo dirbtinių neuroninių tinklą).
- Apie **1965** m. A. L. Samuel sukūrė šachmatų programą.

Skirtumai tarp įprasto programavimo ir dirbtinio intelekto

ĮVESTIS

- **Įprastame:** klaviatūra, pele, iš disko.
- **DI:** vaizdas, garsas, prisilietimas, kvapas, skonis.

VEIKSMAI

- **Įprastame:** manipuliavimas simboliais iš anksto apibrėžtais algoritmais.
- **DI:** apima žinių atvaizdavimą, šablonų sutapimą, paiešką, logiką, problemų sprendimą ir išmokimą.

IŠVESTIS

- **Įprastame:** ekrane, popieriuje, diske.
- **DI:** sintezuota kalba, fizinių objektų manipuliacija, judėjimas erdvėje



Vilniaus universitetas
Matematikos ir informatikos fakultetas
Informatikos katedra



Dirbtinis neuronas – perceptronas

prof. dr. Olga Kurasova
Olga.Kurasova@mii.vu.lt

2018

Dirbtiniai neuroniniai tinklai (DNT)

- Viena iš dirbtinio intelekto sričių yra **dirbtiniai neuroniniai tinklai**, kurie, jei aišku iš konteksto, vadinami tiesiog neuroniniais tinklais.
- Jie pradėti tyrinėti kaip **biologinių neuroninių sistemų modelis**, siekiant išsiaiškinti ir pritaikyti biologinių neuronų sąveikos mechanizmus efektyvesnėms informacijos apdorojimo sistemoms kurti.
- Neuroniniai tinklai **turi galimybę mokytis** iš pavyzdžių.
- Turint duomenų pavyzdžius ir naudojant **mokymo algoritmus**, neuroninis tinklas pritaikomas prie duomenų struktūros ir **išmoksta atpažinti naujus** duomenis, kurie nebuvo naudojami tinklo mokyme.

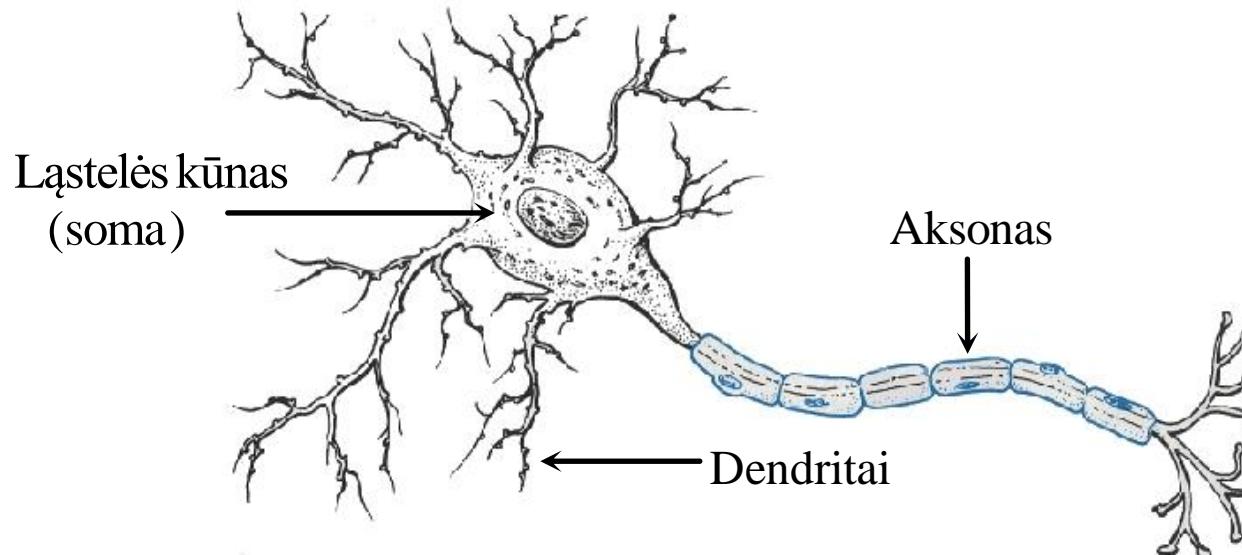
Biologinis neuronas

- Žmogaus **smegenys** susideda iš daugelio (apie 10^{13}) **neuronų**, sujungtų vienų su kitais.
- Kiekvienas neuronas turi vidutiniškai keletą tūkstančių **jungčių**.
- **Neuronas** – tai ląstelė, galinti generuoti elektrocheminį signalą.

Biologinis neuronas

Neuronas turi

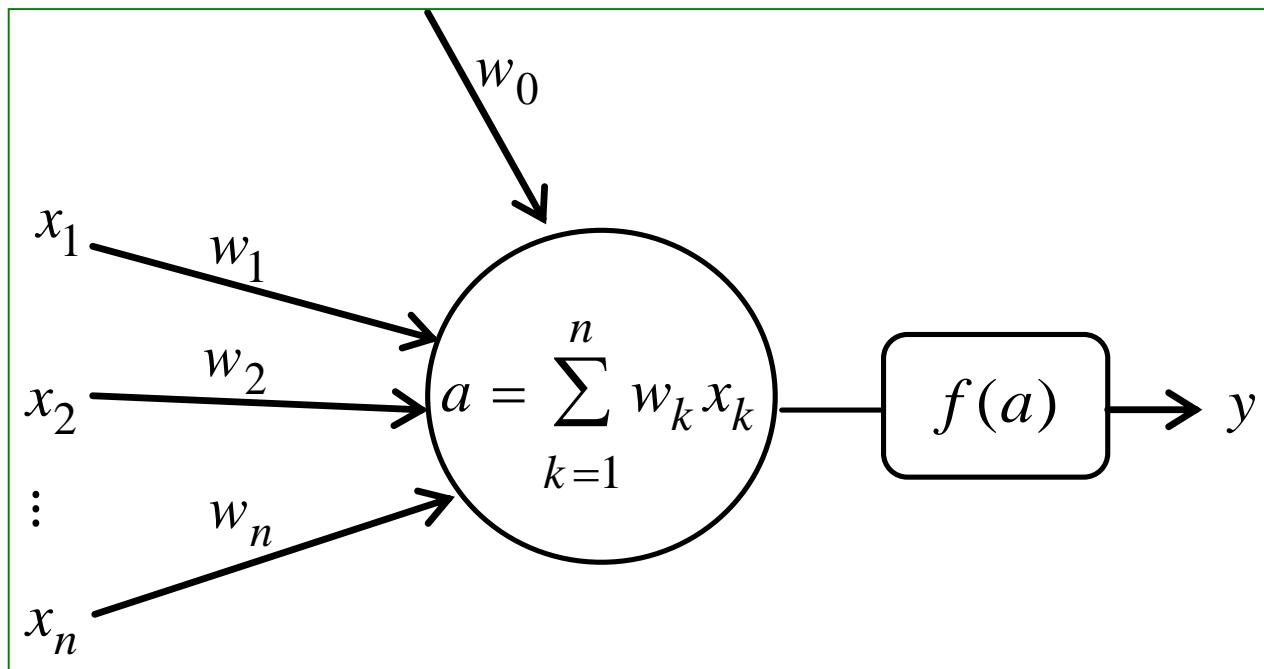
- išsišakojusią įėjimo struktūrą, vadinamusius **dendritus**,
- laštelės kūną, vadinamąją **soma**,
- ir besišakojančią išėjimo struktūrą – **aksoną**.



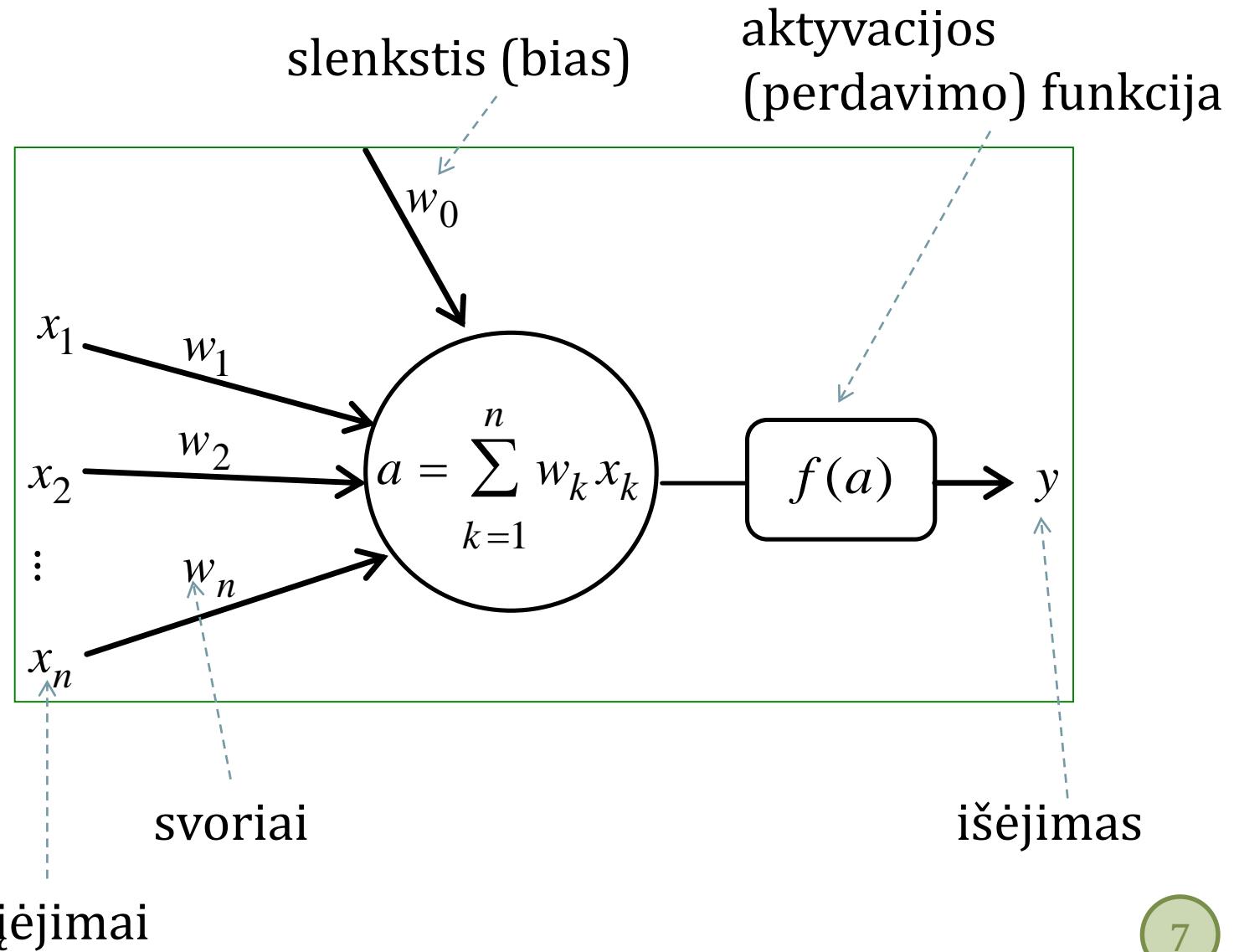
Biologinis neuronas

- Vienos laštelės aksonas su kitos laštelės dendritais jungiasi per **sinapses**.
- Kai sužadinama pakankamai neuronų, prijungtų prie neurono dendritų, tas **neuronas taip pat sužadinamas** ir generuoja elektrocheminį impulsą.
- **Signalas** per sinapses perduodamas kitiems neuronams, kurie vėl gali būti sužadinami.
- Neuronas sužadinamas tik tuo atveju, jei bendras dendritais gautas signalas viršija tam tikrą lygį, vadinančią **sužadinimo slenkstį**.
- Turint **didžiulį skaičių visiškai paprastų elementų**, kurių kiekvienas skaičiuoja svorinę įeinančių signalų sumą ir generuoja binarinį signalą, jei suminis signalas viršija tam tikrą lygį, **galima atlikti gana sudėtingas užduotis**.

Dirbtinio neurono modelis



Dirbtinio neurono modelis



Dirbtinio neurono modelis

- Neuronas turi keletą **įėjimų** x_1, x_2, \dots, x_n .
- Kiekviena įėjimo $x_k, k = 1, \dots, n$, **jungtis** turi savo perdavimo koeficientą (**svorį**) $w_k, k = 1, \dots, n$.
- Iprastai įėjimų ir jungčių svorių reikšmės yra **realieji skaičiai**.
- Skaičiuojama įėjimo reikšmių ir svorių sandaugų **suma**

$$a = w_1 x_1 + w_2 x_2 + \dots + w_n x_n = \sum_{k=1}^n w_k x_k$$

Dirbtinio neurono modelis

- Neuroną apibūdina **aktyvacijos (perdavimo) funkcija**

$$y = f(a) = f\left(\sum_{k=1}^n w_k x_k\right)$$

kurios reikšmė

$$f(a) = \begin{cases} 1, & \text{jei } a \geq w_0, \\ 0, & \text{jei } a < w_0, \end{cases}$$

vadinama **neurono išėjimo reikšme**.

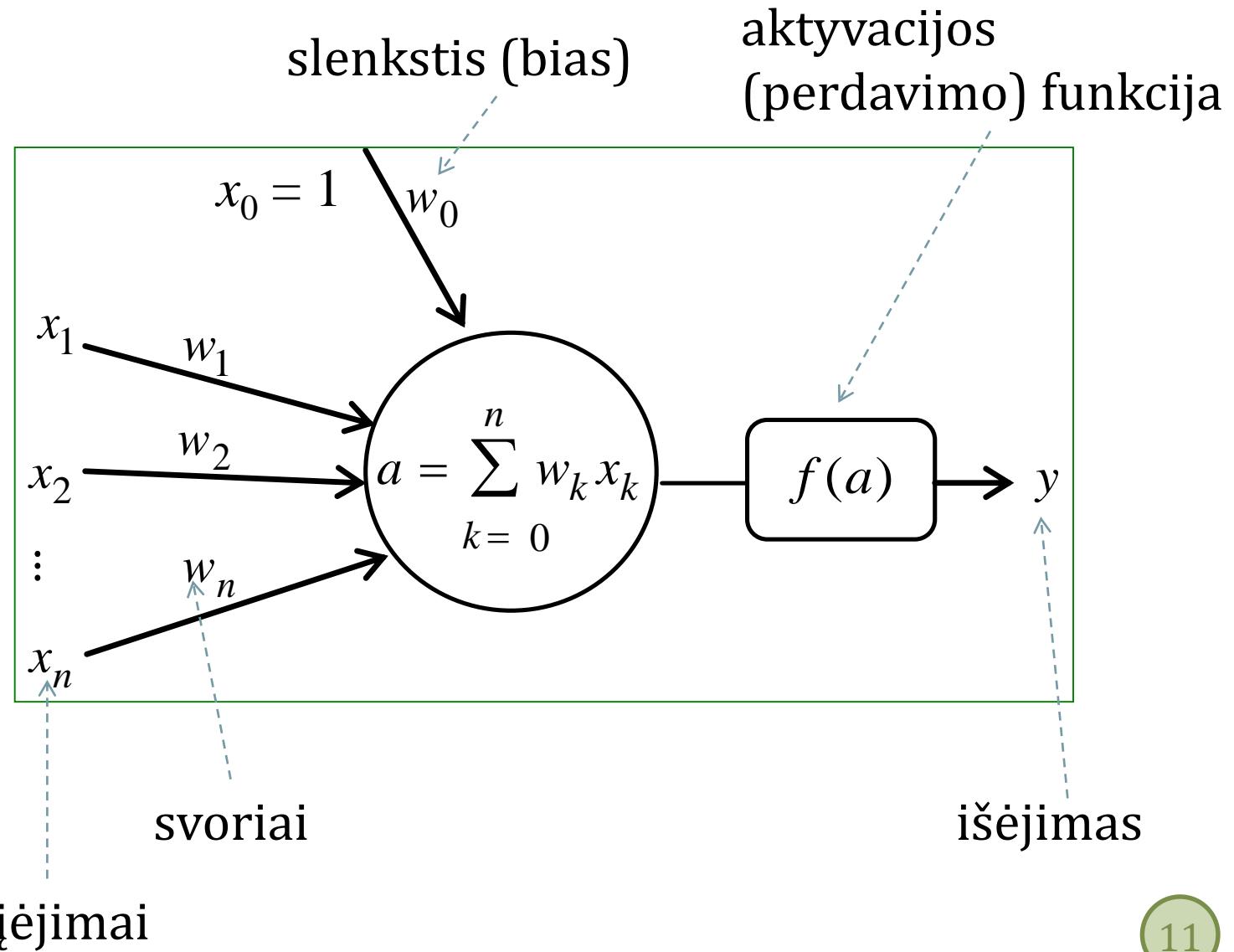
- Čia w_0 yra **slenksčio reikšmė** (bias).

Dirbtinio neurono modelis

- Dažnai yra įvedamas **nulinis iėjimas** x_0 , kuris yra pastovus, $x_0 = 1$, o slenksčio reikšmė tampa **nuliniu svoriu** w_0 .
- Tuomet

$$a = \underbrace{\sum_{k=0}^n w_k x_k}$$

Dirbtinio neurono modelis



Aktyvacijos funkcijos

- Tiesinė

$$f(a) = \begin{cases} 1, & \text{if } a \geq 0 \\ 0, & \text{if } a < 0 \end{cases}$$

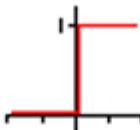
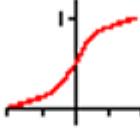
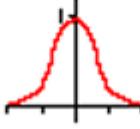
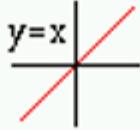
- Sigmoidinė

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

- Hiperbolinis tangentas

$$f(a) = \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$$

Aktyvacijos funkcijos

Slenkstinė (angl. <i>unit step</i>)		$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$
Sigmoidinė (angl. <i>sigmoid</i>)		$f(x) = \frac{1}{1+e^{-\beta x}}$
Gabalais tiesinė (angl. <i>piecewise linear</i>)		$f(x) = \begin{cases} 0 & \text{if } x \leq x_{min} \\ mx+b & \text{if } x_{max} > x > x_{min} \\ 1 & \text{if } x \geq x_{max} \end{cases}$
Gauso (angl. <i>Gaussian</i>)		$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$
Tiesinė (angl. <i>linear</i>)		$f(x) = x$

Neuronas duomenims klasifikuoti

- Dirbtinis **neuronas** dar vadinamas **perceptronu** arba **vienasluoksniu** perceptronu.
- I **neurono įėjimus** paduodami objektus apibūdinančių požymiu x_1, x_2, \dots, x_n reikšmės.
- **Neurono išėjime** – duomenų klasių reikšmės.
- Tikslas – **rasti tokias svorių reikšmes** $w_0, w_1, w_2, \dots, w_n$, kad apskaičiavus $a = w_0x_0 + w_1x_1 + \dots + w_nx_n$ ir $f(a)$, **išėjime** y gautos reikšmės **sutaptu su duomenų klasių reikšmėmis**.
- Svořių reikšmės turi būti tokios, kad jos būtų **tinkamos visiems duomenims**.

Perceptrono mokymas

- Galimybė **mokytis** yra **esminė intelekto savybė**.
- Tinkamų svorių radimas vadinamas neurono (perceptrono) **mokymu**.
- Duomenų aibė, kuri bus naudojama neuronui mokyti, vadinama **mokymo aibe**.
- Tegul turime m mokymo aibės vektorių $X_i = (x_{i1}, x_{i2}, \dots, x_{in})$, $i = 1, \dots, m$, kuriuos vadinsime **įėjimų vektoriais**.
- Šie vektoriai yra susieti su **norima reikšme** t_i (*target*). Tai norima reakcija į vektorių X_i .
- Sprendžiant klasifikavimo uždavinį, **norimos reikšmės yra klasių numeriai**.

Perceptrono mokymas

- **Mokymo procese** svoriai $W = (w_0, w_1, w_2, \dots, w_n)$ keičiami taip, kad tinklo išėjimo reikšmė y_i , gauta iš išjimų pateikus vektorių X_i , būtų kiek galima **artimesnė norimai reikšmei** t_i .
- Tai yra perceptrono veikimo **paklaida** būtų kiek galima **mažesnė**.
- Ši **paklaida** $E(W)$ gali būti apibrėžiama, kaip skirtumų tarp neurono išėjime gautų reikšmių ir norimų reikšmių sumos funkcija

$$E(W) = \sum_{i=1}^m (y_i - t_i)$$

Perceptrono mokymas

- Vadinasi perceptrono mokymo eigoje reikia **minimizuoti paklaidos funkciją $E(W)$.**
- Jeigu ši funkcija yra diferencijuojama pagal svorius, jos minimum galima rasti **gradientiniai optimizavimo metodai**.
- Patogumo dėlei, dažnai minimizuojama **tokios išraiškos funkcija.**

$$E(W) = \frac{1}{2} \sum_{i=1}^m (y_i - t_i)^2$$

Perceptrono mokymas

- Iš pradžių **generuojamos atsitiktinės svorių** w_k reikšmės, įprastai intervale $(0, 1)$.
- Tada gradientinio nusileidimo algoritmu judama antigradiento kryptimi, **svorių reikšmes keičiant** pagal iteracinę formulę

$$w_k(t + 1) = w_k(t) + \Delta w_k(t)$$

- Čia t – iteracijos numeris,

$$\Delta w_k(t) = -\eta \frac{\partial E(W)}{\partial w_k}$$

- η – yra teigiamas daugiklis, kuris vadinamas **mokymo greičiu** (*learning rate*) ir kuriuo reguliuojamas gradientinio optimizavimo žingsnio ilgis.

Perceptrono mokymas

- Iprastai svoriai **pakeičiami pateikus vieną** žėjimo vektorių.
- Mokymo procesas kartojamas **daug kartų** **pateikiant** visus žėjimo vektorius.
- Mokymas **stabdomas** arba atlikus iš anksto nustatyta **iteracijų skaičių**, arba pasiekus norimą **mažą paklaidos reikšmę**.

Perceptrono mokymo pavyzdys

- Tegul mokymo duomenys – loginės funkcijos AND teisingumo lentelė.

	x_1	x_2	t
X_1	1	1	+1
X_2	1	0	-1
X_3	0	1	-1
X_4	0	0	-1

- Dar reikia pridëti vieną stulpelį $x_0=(1, 1, 1, 1)$
- Svorų vektorių sudaro trys komponentės $W(w_0, w_1, w_2)$.

Perceptrono mokymo pavyzdys

- **Tikslas** – rasti tokias svorių reikšmes w_0, w_1, w_2 , kad išėjime y_i , gautos reikšmės sutaptų su norimomis reikšmėmis t_i , t. y. paklaida $E(W) = 0$.
- Naudosime perceptroną, kurio **aktyvacijos funkcija** yra

$$y = f(a) = \begin{cases} +1, & \text{jei } a > 0 \\ -1, & \text{jei } a \leq 0 \end{cases}$$

- Pradinės svorių reikšmės lygios nuliui, $w_k = 0$.
- Paprastumo dėlei, tegul $\eta=1$.

Perceptrono mokymo algoritmas

WHILE (išėjimo reikšmės nelygios trokštamoms
reikšmėms ir iteracijų skaičius neviršija nustatyta)
{ FOR (visiems įėjimo vektoriams X_i)

{

$$a_i = w_0x_0 + w_1x_1 + w_2x_2$$

$$y_i = f(a)$$

IF ($y_i \neq t_i$)

$$w_k(\text{naujas}) = w_k(\text{senas}) + \eta t_i x_{ik}$$

}

}

Paklaidos minimizavimas

- **Minimizuojama** funkcija $E(W) = \frac{1}{2} \sum_{i=1}^m (y_i - t_i)^2$.
- Tai suma paklaidų **kiekvienam** įėjimų vektoriui:
 $E(W) = \sum_{i=1}^m E_i$
- Prisiminkime, kad $y_i = f(a_i) = f(\sum_{k=0}^n w_k x_{ik})$.
- Randama funkcijos **išvestinė** pagal w_k :

$$\frac{\partial E_i(W)}{\partial w_k} = (y_i - t_i) \times \frac{\partial f(a)}{\partial w_k} \times \frac{\partial a}{\partial w_k} = (y_i - t_i)x_{ik}$$

kai $f(\sum_{k=0}^n w_k x_{ik}) = \sum_{k=0}^n w_k x_{ik}$.

- Todėl bendru atveju neurono mokyme galima taikyti **taisykłę**:

$$w_k(t+1) = w_k(t) + \eta(y_i - t_i)x_{ik}$$

Skriamasis paviršius

- Neuronas **padalina** sprendinių aibę į du regionus.
- Nagrinėkime atvejį, kai įėjimų yra tik du x_1, x_2 ir

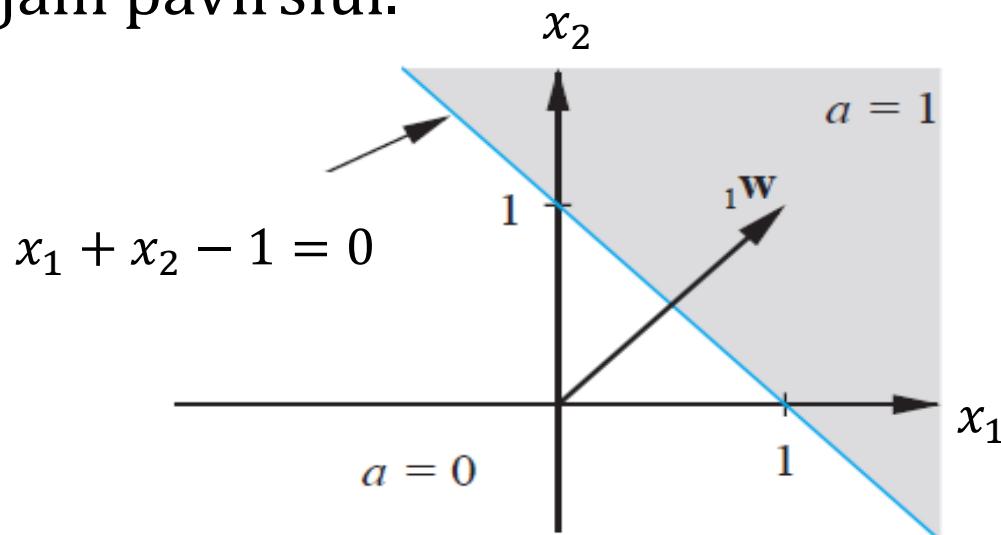
$$y = f(a) = \begin{cases} 1, & \text{jei } a \geq 0 \\ 0, & \text{jei } a < 0 \end{cases}$$

- Tuomet **skriamasis paviršius** (*decision boundary*) (dviejų įėjimų atveju – tai teisė) apibrėžiamas įėjimų vektoriais, kuriems $a = 0$:

$$w_1x_1 + w_2x_2 + w_0 = 0$$

Pavyzdys

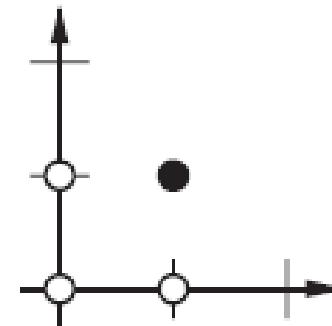
- Tarkime $w_1 = 1$, $w_2 = 1$, $w_0 = -1$. Tuomet **skiriamaasis paviršius** $x_1 + x_2 - 1 = 0$.
- Šios tiesės vienoje puseje, **neurono išėjimas** bus lygus 0, kitoje 1.
- Svorių vektorius turi būti **statmenas (ortogonalus)** skiriamajam paviršiui.



Skriamasis paviršius loginei funkcijai AND (1)

- Nagrinėkime **pavyzdį**, kai funkcija AND apibrėžiama taip:

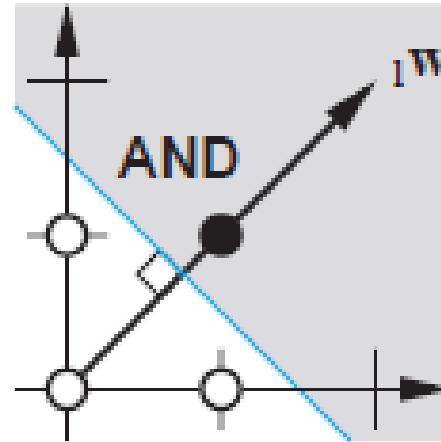
	x_1	x_2	t
X_1	0	0	0
X_2	0	1	0
X_3	1	0	0
X_4	1	1	1



- Tuščias apskritimas atitinka $t = 0$, skrituliukas $t = 1$.

Skriamasis paviršius loginei funkcijai AND (2)

- Reikia nubrėžti skriamąjį paviršių (**melsva tiesė**), kurios vienoje pusėje būtų apskritimai, kitoje – skrituliukas.



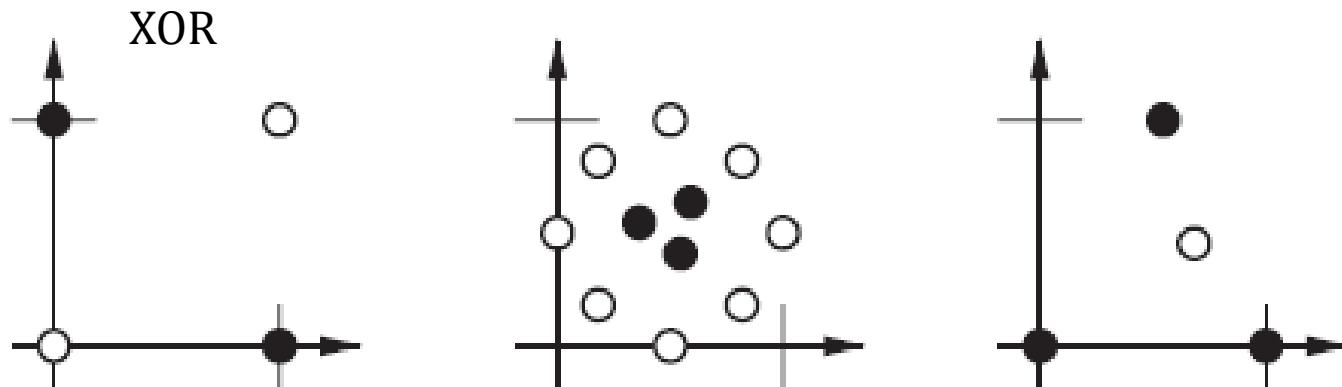
- Dabar reikia pasirinkti svorių vektorių, kuris būtų statmenas skriamajai tiesei. Vienas variantų $W(w_1, w_2) = (2, 2)$.

Skriamasis paviršius loginei funkcijai AND (3)

- Dabar belieka **rasti** w_0 .
- Reikia **parinkti tašką** (x_1, x_2) , esantį ant skriamosios tiesės ir tenkinantį lygybę $w_1x_1 + w_2x_2 + w_0 = 0$.
- Galimas **taškas** $(x_1, x_2) = (1,5, 0)$.
- Tuomet $2 \times 1,5 + 2 \times 0 + w_0 = 0$. Iš čia $w_0 = -3$.

Tiesiškai neatskiriами atvejai

- Deja, daugelyje realių uždavinių **negalima** nubraižyti (suformuoti) **tiesiškai atskiriamo paviršiaus**.
- Tam reikia naudoti **sudėtingesnius neuroninius tinklus**.



„Kišeninis“ algoritmas (1)

- Perceptrono mokymo „**kišeninis**“ algoritmas (*pocket algorithm*) taikomas sprendžiant tiesiškai neatskiriamą uždavinį, ieškant **kiek galima geresnio teisiško atskyrimo**.
- Algoritmo metu „kišenėje“ saugomi gauti svoriai ir **naudojami rasti geriausi**. Svorai keičiami, jei tik randami geresni.

„Kišeninis“ algoritmas (2)

pocket (training_list, max_iteration)

```
w = randomVector()
```

```
best_error = error(w)
```

```
for i in range(0, max_iteration)
```

```
    x=misclassified_sample(w, training_list)
```

```
    w=vector_sum(w, x.y(x))
```

```
    if error(w) < best_error
```

```
        best_w = w
```

```
        best_error = error(w)
```

```
return best_w
```

Mokymo iteracija – mokymo epocha

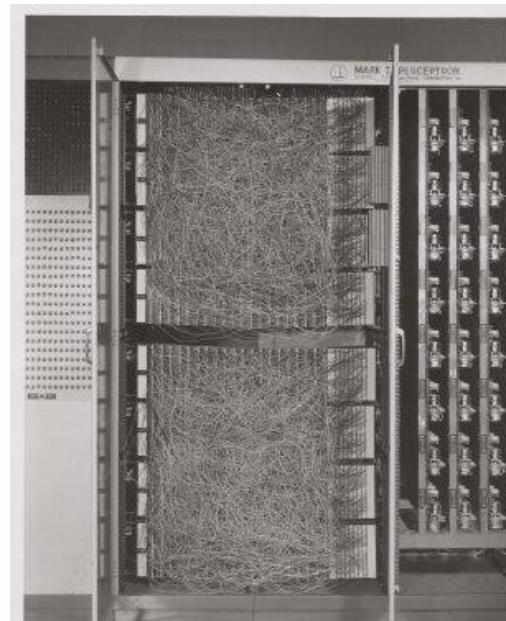
- Neuronų mokyme naudojamos šios sąvokos „**mokymo iteracija**“, „**mokymo epocha**“.
- Koks **skirtumas** tarp jų?
- Mokymo **iteracija** – tai neuronų mokymo proceso dalis, kurios metu apdorojamas vienas įėjimų vektorius.
- Mokymo **epocha** – tai neuronų mokymo proceso dalis, kurios metu apdorojamas visas įėjimų vektorių rinkinys vieną kartą.
- Vienos mokymo epochos metu **ivyksta tiek iteracijų**, kiek yra įėjimo vektorių.

Dirbtinių neuronų ištakos

- **McCulloch-Pitts** (MCP, M-P) neurono modelis (1943)
 - Jėjimai tik (0, 1).
 - Tik slenkstinė aktyvacijos funkcija.
 - Vienintelė w_0 reikšmė visiems jėjimams.
 - Visiems jėjimams vienodi svoriai (teigiami skaičiai).
- **Rosenblatt** perceptronas (1958)
 - Visi svoriai nėra identiški (teigiami ir neigiami skaičiai).
 - Įvairios aktyvacijos funkcijos.
 - Yra mokymo taisyklė.

Mark I Perceptron mašina

- **Pirmoji perceptrono realizacija** buvo sukurta 1957 m. skaičiavimo mašinoje IBM 704 ne kaip programinė, bet **techninė įranga**.
- Ši mašina buvo skirta **vaizdų atpažinimo** (*image recognition*) uždaviniui spręsti.





Vilniaus universitetas
Matematikos ir informatikos fakultetas
Informatikos katedra



Dirbtiniai neuroniniai tinklai

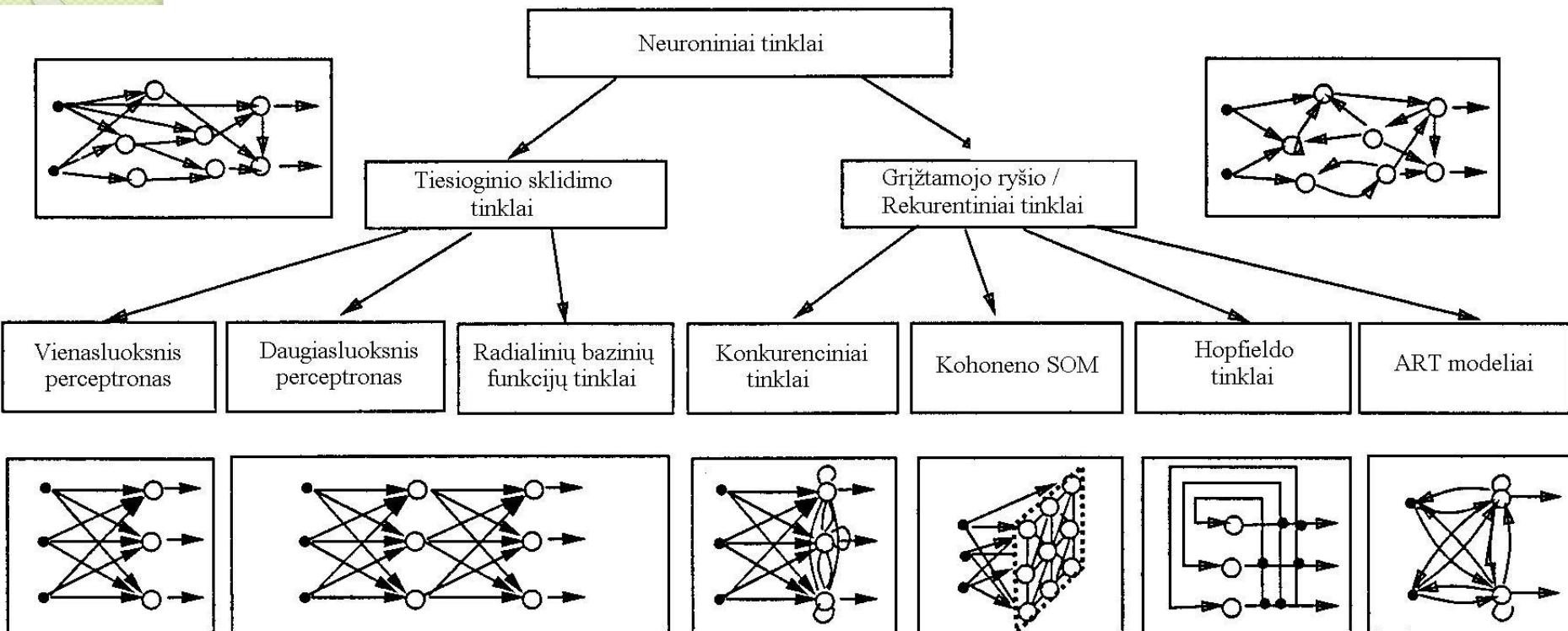
prof. dr. Olga Kurasova
Olga.Kurasova@mii.vu.lt

2018

Neuronų jungimas į tinklus

- Dirbtiniai neurono gali būti jungiami į **dirbtinius neuroninius tinklus (DNT)**.
- Pagal jungimo konstrukciją neuroniniai tinklai **sudaro dvi** pagrindines **grupes**:
 - **tiesioginio sklidimo** (*feedforward*) tinklai, kuriuose nėra grafo ciklų;
 - **grįžtamojo ryšio** (*feedback*) tinklai, kuriuose yra grafo ciklai.

DNT klasifikacija



DNT mokymas

- Nors tikslų mokymo apibrėžimą sunku suformuluoti, dirbtinio **neuroninio tinklo mokymo procesas** apibrėžiamas kaip tinklo struktūros ir jungčių svorių keitimo uždavinys, siekiant, kad tinklas galėtų atlikti jam skirtą užduotį.
- **Skirtingos** tinklų **architektūros** reikalauja skirtingų jų **mokymo algoritmu**.
- Yra trys pagrindinės neuronų **mokymo paradigmos**:
 - mokymo **su mokytoju** algoritmai (*supervised learning*);
 - mokymo **be mokytojo** algoritmai (*unsupervised learning*);
 - **hibridinis** mokymas (*hybrid learning*).

DNT mokymas su mokytoju

- Kalbant apie mokymo su mokytoju algoritmus, yra vartojama sąvoka **norimos išėjimo reikšmės**. Tai iš anksto žinomas reikšmės, pavyzdžiui, klasių numeriai, prognozuojamos reikšmės ir pan.
- **Mokymo su mokytoju atveju** tinklo išėjimų reikšmės, skaičiuojamos kiekvienam įėjimo vektoriui $X_i = (x_{i1}, x_{i2}, \dots, x_{in})$, $i \in \{1, \dots, m\}$, yra tiesiogiai **susijusios su norimomis** tų išėjimų reikšmėmis.
- Mokymo metu **tinklas koreguojamas keičiant svorių vektorių reikšmes** ir siekiant gauti kiek galima mažesnę paklaidą, t. y. ieškoma tokį svorį, kad skirtumas tarp norimų išėjimo reikšmių ir reikšmių, gautų išmokius neuroninį tinklą, būtų kiek galima mažesnis.
- Aptartas **perceptrono mokymas** priskiriamas šio tipo mokymui.

DNT mokymas be mokytojo

- Kartais norimos gauti tinklo išėjimo reikšmės nėra žinomos. Tada naudojami **mokymo be mokytojo algoritmai**.
- Šio tipo metoduose tinklas **mokomas ieškoti koreliacijų ar panašumų** tarp mokymo aibės įėjimų. Cia nėra grįztamojo ryšio, pasakančio, kuris atsakymas bus arba yra teisingas.
- Mokymo be mokytojo algoritmuose **nėra mokytojo signalo**. Turima tik mokymo aibė X , kuri sudaryta iš vektorių X_1, X_2, \dots, X_m .
- Metodų tikslas yra **suskirstyti mokymo duomenis** į tam tikras grupes arba rasti juose kokius nors **reguliarumus** ar **ypatumus**.

DNT hibridinis mokymas

- **Hibridinis mokymas** apima mokymo su mokytoju ir be mokytojo algoritmus:
 - **dalis** tinklo svorių nustatomi pagal mokymą su mokytoju,
 - **kita dalis** gaunama iš mokymo be mokytojo.



Vilniaus universitetas
Matematikos ir informatikos fakultetas
Informatikos katedra



Duomenų klasifikavimas

prof. dr. Olga Kurasova
Olga.Kurasova@mii.vu.lt

2018

Dar apie duomenų klasifikavimą

- Pagal turimus duomenis, kurių klasės yra žinomos, **reikia sukurti mechanizmą** (klasifikatorių), kuris gebėtų priskirti klases duomenims, kuriems jos nėra žinomos.
- Duomenims klasifikuoti taikomi **įvairūs klasifikavimo metodai**: Naive Bayes, k artimiausiu kaimynu, atraminių vektorių, klasifikavimo medžių ir kt.
- Dirbtiniai neuroniniai tinklai taip pat yra plačiai **naudojami duomenims klasifikuoti**.
- Net **vienas neuronas geba** spresti nesudėtingus klasifikavimo uždavinius.

Duomenys klasifikavimui

Sprendžiant **klasifikavimo** uždavinius išskiriami **trijų tipų duomenys**:

- **mokymo duomenys** naudojami klasifikatoriui sukurti,
- **testavimo duomenys** naudojami patikrinti (testuoto) klasifikatoriaus išmokymo klasifikuoti lygi,
- **nauji duomenys**, kurių klasės nėra žinomas, bet taikant sukurta klasifikatorių jos yra nustatomos.

Klasifikavimo tikslumo matai

- Klasifikatorius **turi būti išmokytas** taip, kad gebėtų gerai klasifikuoti duomenys, kurių klasės nėra žinomos.
- Vadinas reikia turėti to **išmokymo įvertinimo matus**.
- Klasifikavimo tikslumui nustatyti dažniausiai vertinami šie matai:
 - **jautumas** (angl. *sensitivity*);
 - **specifiškumas** (angl. *specificity*);
 - **bendras klasifikavimo tikslumas** (angl. *accuracy*).

Klasifikavimo tikslumas

Apibréžkime pagrindines savokas:

- **tikrai teigiamas** (TT) (angl. *true positive*) – objektas X_i priskirtas klasei C_j , ir iš tiesų jis jai priklauso;
- **tikrai neigiamas** (TN) (angl. *true negative*) – objektas X_i nepriskirtas klasei C_j , ir iš tiesų jis jai nepriklauso;
- **klaidingai teigiamas** (KT) (angl. *false positive*) – objektas X_i priskirtas klasei C_j , bet iš tiesų jis jai nepriklauso;
- **klaidingai neigiamas** (KN) (angl. *false negative*) – objektas X_i nepriskirtas klasei C_j , bet iš tiesų jis jai priklauso.

Klasifikavimo matrica

Apskaičiavus šiuos įverčius, sudaroma **klasifikavimo matrica** (angl. *classification* ar *confusion matrix*)

		gauta klasė	
		C_1	C_2
tikroji klasė	C_1	tikrai teigiamas (TT)	klaidingai teigiamas (KT)
	C_2	klaidingai neigiamas (KN)	tikrai neigiamas (TN)

Jautumas ir specifiškumas

Klasifikavimo matų reikšmės yra apskaičiuojamos pagal šias formules:

$$\text{jautumas} = \frac{\text{TT skaičius}}{\text{TT skaičius} + \text{KN skaičius}},$$

$$\text{specifiškumas} = \frac{\text{TN skaičius}}{\text{TN skaičius} + \text{KT skaičius}},$$

$$\text{bendras klasifikavimo tikslumas} = \frac{\text{TT skaičius} + \text{TN skaičius}}{\text{visų objektų skaičius}}.$$

Kryžminė patikra

- Klasifikavimo tikslumas gali priklausyti nuo to, kaip visa **duomenų aibę padalinta į mokymo ir testavimo aibes.**
- Todėl tikslina **klasifikavimą atlikti keliems skirtiniams** tos pačios duomenų aibės mokymo ir testavimo rinkiniams ir **įvertinti vidutinį klasifikavimo tikslumą.**
- Tam tikslui dažnai naudojamas **kryžminės patikros metodas** (angl. *cross validation*).

Kryžminė patikra

- Kryžminės patikros metu duomenų aibė yra **suskaidoma** į q **nesusikertančių blokų** (angl. *folds*).
- Klasifikavimo algoritmas yra **apmokomas** naudojant $q - 1$ bloko duomenis, o likusi duomenų dalis yra panaudojama algoritmui **testuoti**.
- **Fiksuojamos** klasifikavimo matų reikšmės.
- Ši procedūra atliekama q **kartu**, mokymui imant vis kitus $q - 1$ blokus, pabaigoje randamos klasifikavimo matų **vidutinės reikšmės**. Pagal jas vertinamas **sukurto klasifikatoriaus tikslumas**.



Vilnius universitetas
Matematikos ir informatikos fakultetas
Informatikos katedra



DIDIEJI DUOMENYS (BIG DATA). KAS TAI?

prof. dr. Olga Kurasova

Olga.Kurasova@mii.vu.lt

2018



Didieji duomenys. Jų atsiradimo šaltiniai

- Modernios technologijos leidžia generuoti **milžiniškus duomenų kiekius.**
- Pagrindiniai **didžiujų duomenų šaltiniai:**
 - Astronomija,
 - Meteorologija,
 - Genų inžinerija,
 - Medicina,
 - Bankinės ir finansinės sistemos,
 - Socialiniai tinklai,
 - Telekomunikacija,
 - Daiktų internetas (*Internet of Things, IOT*)
 - Saitynas (*web*).
 - Kiti



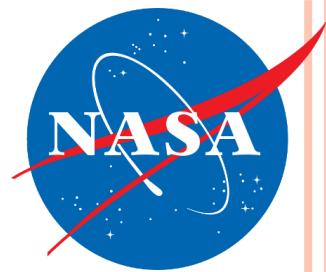
Didieji duomenys



H2020-ICT-2015

- “big data” is when the size of the data itself becomes part of the problem
- “big data” is data that becomes large enough that it cannot be processed using conventional methods

Didžiųjų duomenų ištakos

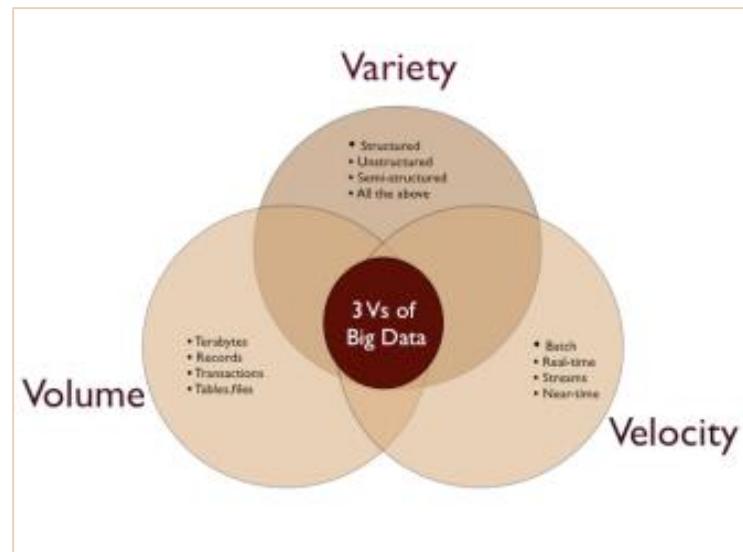


- **Visualization provides** an interesting challenge for computer systems: data sets are generally quite large, taxing the capacities of main memory, local disk, and even remote disk. We call this the problem of **big data**. When data sets do not fit in main memory (in core), or when they do not fit even on local disk, the most common solution is to acquire more resources (1997).
- Big data is high-**volume**, high-**velocity** and/or high-**variety** information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation.

Didieji duomenys (3V)

Big data can be described by the following characteristics:

- **Volume** (didžiulis duomenų kiekis).
- **Variety** (plati duomenų įvairovė).
- **Velocity** (nuolat atsirandantys nauji duomenys).



Didieji duomenys (3V)

Gartner®

Application Delivery Strategies



META Group

Date: 6 February 2001

File: 949

Author: Doug Laney

3D Data Management: Controlling Data Volume, Velocity, and Variety. Current business conditions and mediums are pushing traditional data management principles to their limits, giving rise to novel, more formalized approaches.

Very difficult to define (precisely):

- data is “**big**” if it defies traditional processing & storage paradigms – bigness becomes part of the problem.
- the “**3Vs**”: volume (size), velocity (bytes/s), variety (database, jpeg, video, numbers, text in language X...).
- ...to which we add the 4th V to denote creation of Value (by linking, aggregating, analysing, visualizing...).



Didieji duomenys (4V)



Didieji duomenys (5V)

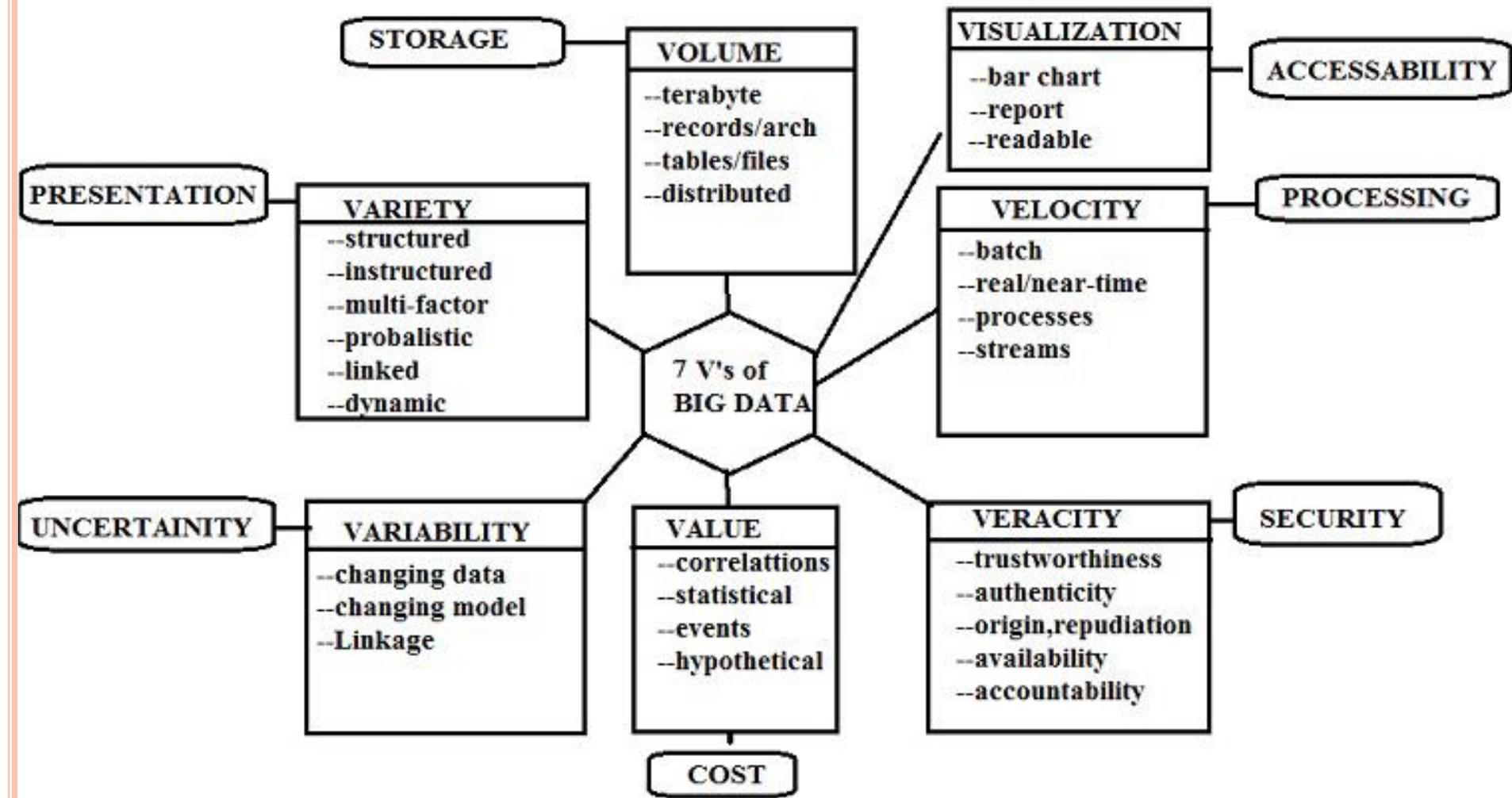
- **Volume** – from terabytes to petabytes and up.
- **Variety** – an expanding universe of data types and sources.
- **Velocity** – accelerated data flow in all directions.
- **Variability** – inconsistent data flows with periodic peaks.
- **Complexity** – the need to correlate and share data across entities.



“**Big data** is not only about analytics, it's about the whole pipeline. So when you think about big data solutions, you have to think about all the different steps: collect, store, organize, analyze, and share,” said Amazon CTO Werner Vogels.



Didieji duomenys (7V)



Didieji duomenys

- **Big data** refers to data being collected in ever-escalating volumes, at increasingly high velocities, and for a widening variety of unstructured formats and variable semantic contexts. Big data can be historical (meaning stored data) or real-time (meaning streamed directly from the source).



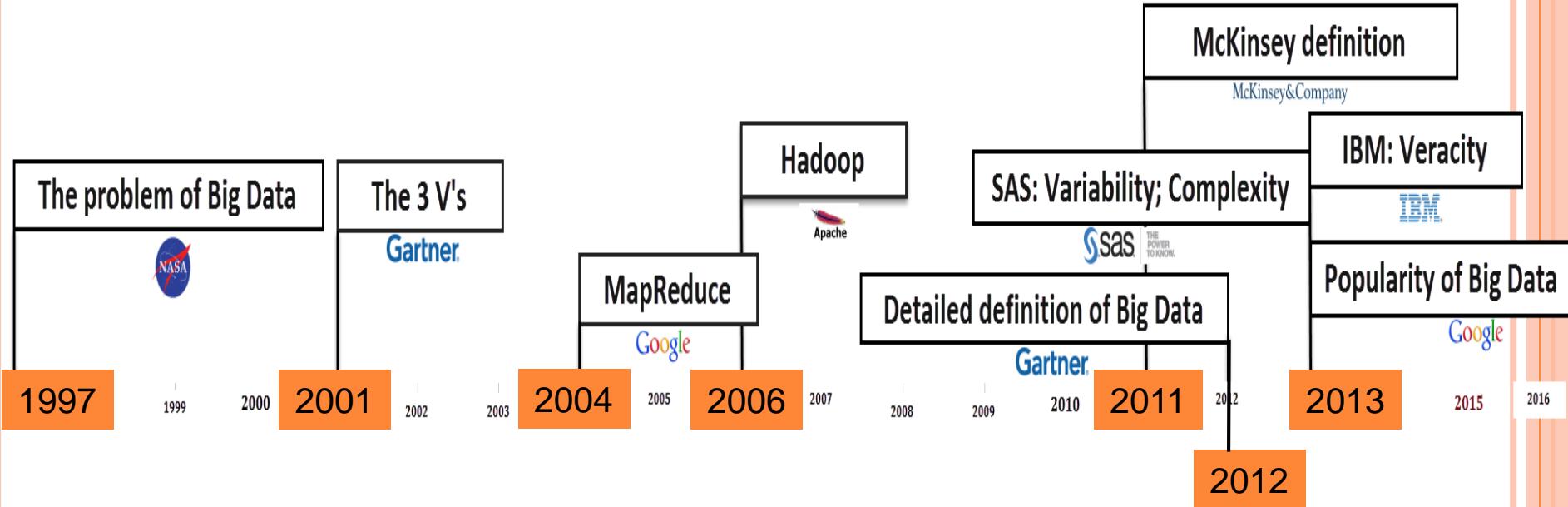
Microsoft

- Every day, we create 2.5 quintillion bytes of data—so much that 90% of the data in the world today has been created in the last two years alone. This data comes from everywhere: sensors used to gather climate information, posts to social media sites, digital pictures and videos, purchase transaction records, and cell phone GPS signals to name a few. This data is **big data**.



10

Didieji duomenys: anksčiau ir dabar



What Types of Data are in Big Data?

- Structured Data
 - Tables, Relational Data ...
- Unstructured Data
 - Raw text, Images, Video, Audio ...
- Semi-structured
 - Hybrid data, such as documents with tables ...
- Metadata
 - Structured data about data
- Streaming Data
 - Data that moves across networks at high speed
- Temporal Data
 - Data including trends/activities in time
- Geospatial Data
 - Data that includes information on positions in space
- Many others ...



Matavimo vienetai

Reikšmė	Žymėjimas	Pavadinimas
1000	kB	kilobyte
1000^2	MB	megabyte
1000^3	GB	gigabyte
1000^4	TB	terabyte
1000^5	PB	petabyte
1000^6	EB	exabyte
1000^7	ZB	zettabyte
1000^8	YB	youttabyte

How Big is Big Data?

- Internet traffic is now ~**5 Zettabyte** per year (IBM)
- **Visa** processes 150 Million transactions per day (VISA)
- **Library of Congress** holds 3.2 Petabytes of data
- 207 Terabytes of video loaded daily on **YouTube** (2012)
- 50 billion **devices connected to the Internet** by 2020 (IDC)
- 50 Billion photos on **Facebook** in 2010
- 400 Million **Tweets** per day (Washington Post)



by the poster created by analysts at Altamira

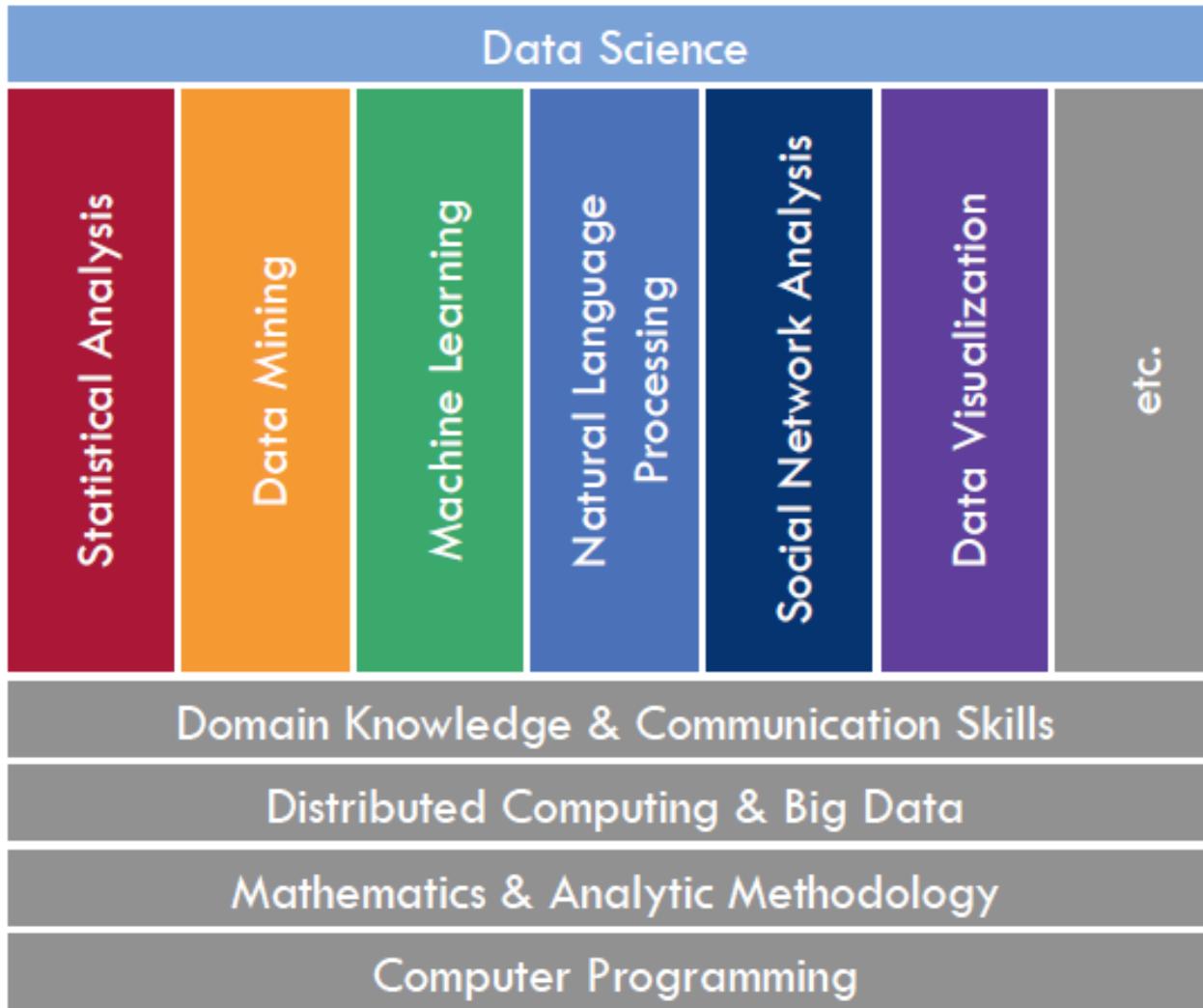
How Big is Big Data?

- Seagate sold 330 Exabytes of **hard drives** in 2011
- LHC produces 500 Exabytes of particle collision data per day **CERN!**
- **iPhone 5s**: 76 Gigaflops
- Fastest **supercomputer**: 50 Petaflops
- Interesting Comparison: **Human Brain** has 100 Billion Neurons (100 Giga-Neurons), 100 Trillion Synapses (100 Tera-Synapses), neurons “fire” 1-1000 times/second (100 Giga-fires to 100 Tera-fires per second)



by the poster created by analysts at Altamira

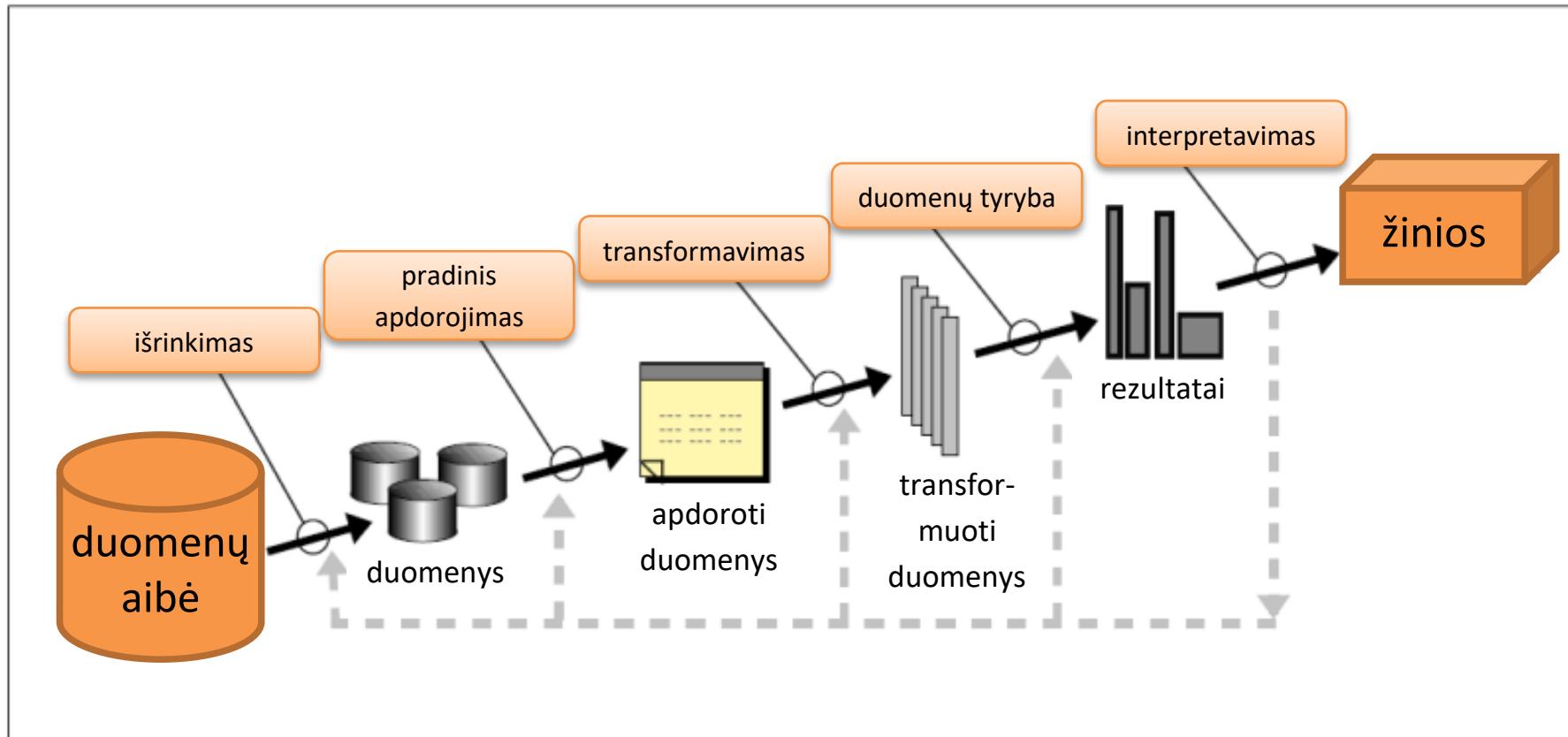
What is a Data Scientist?



Su didžiaisiais duomenimis susiję sprendimai ir technologijos

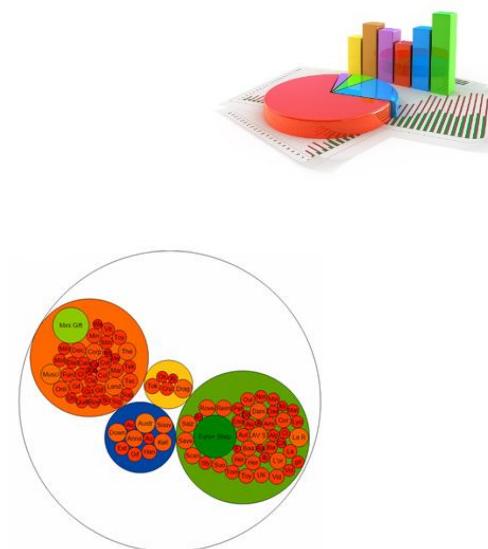
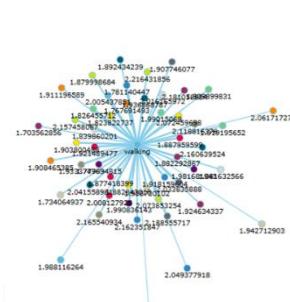
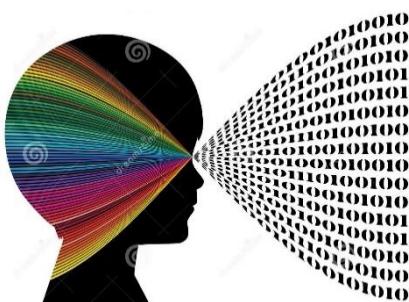


Duomenų tyryba žinių radimo procese



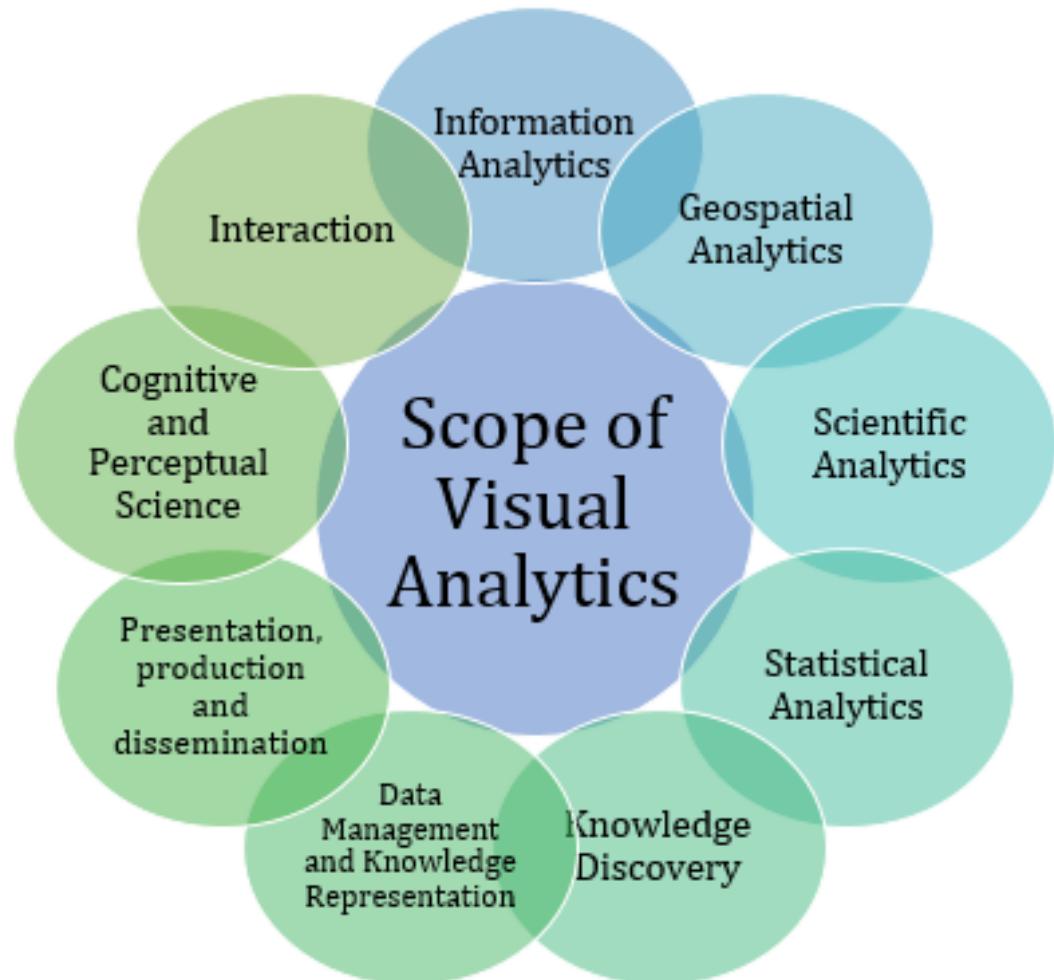
Duomenų vizuali analizė

- **Vizualizuoti** (*lot. visualis – regimas*) – nematomą atvaizdą, daiktą, reiškinį, daryti matomą (Tarptautinių žodžių žodynėlis, 1985).
- **Vizualizavimas** – tai informacijos kodavimas į regimuosius vaizdus.
- **Vizualizavimas** – tai grafinis informacijos pateikimas, palengvinantis informacijos suvokimą.

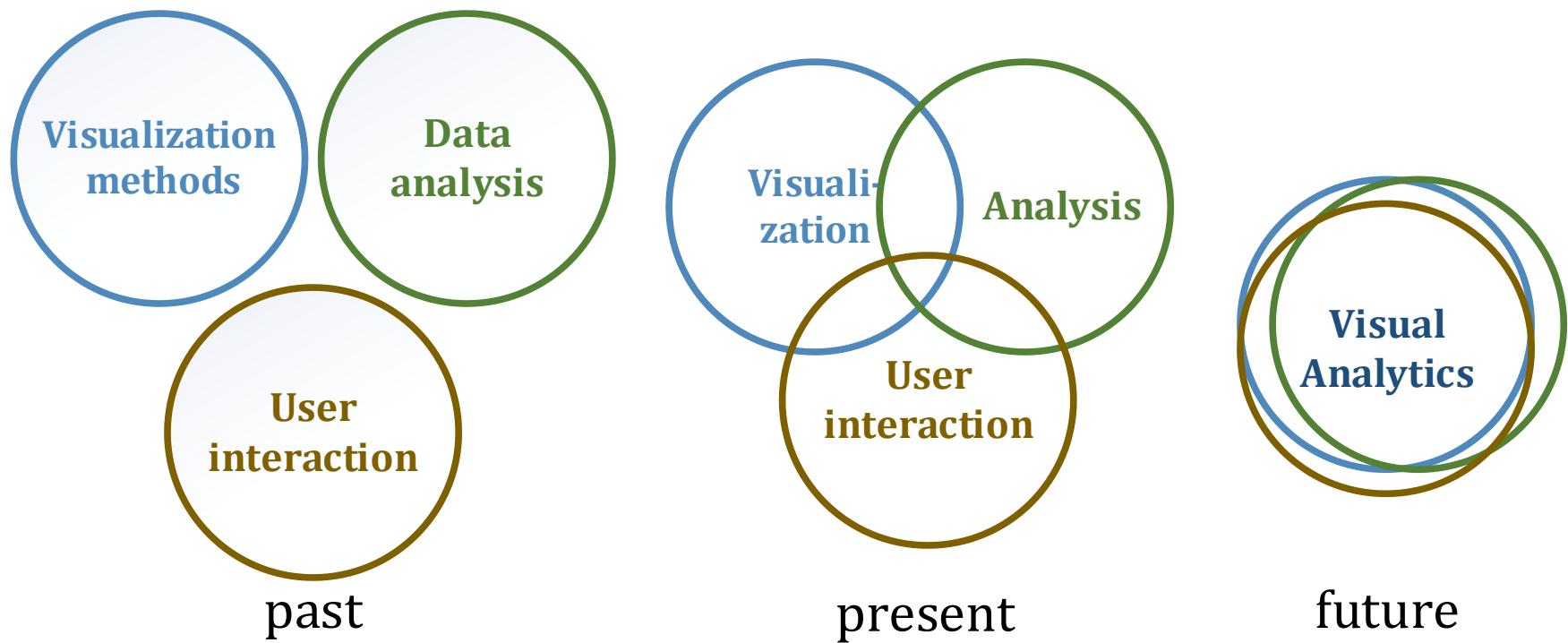


Visual Analytics

Visual Analytics combines automated analysis techniques with **interactive visualizations** for an effective **understanding**, reasoning and **decision making** on the basis of very large and complex datasets.



Vizualizavimas – anksčiau, dabar, ateityje



Visual Analytics as a Web Services

- **SAS** is an integrated system of software solutions for data management, graphics design, statistical and mathematical analysis, business forecasting and decision support.

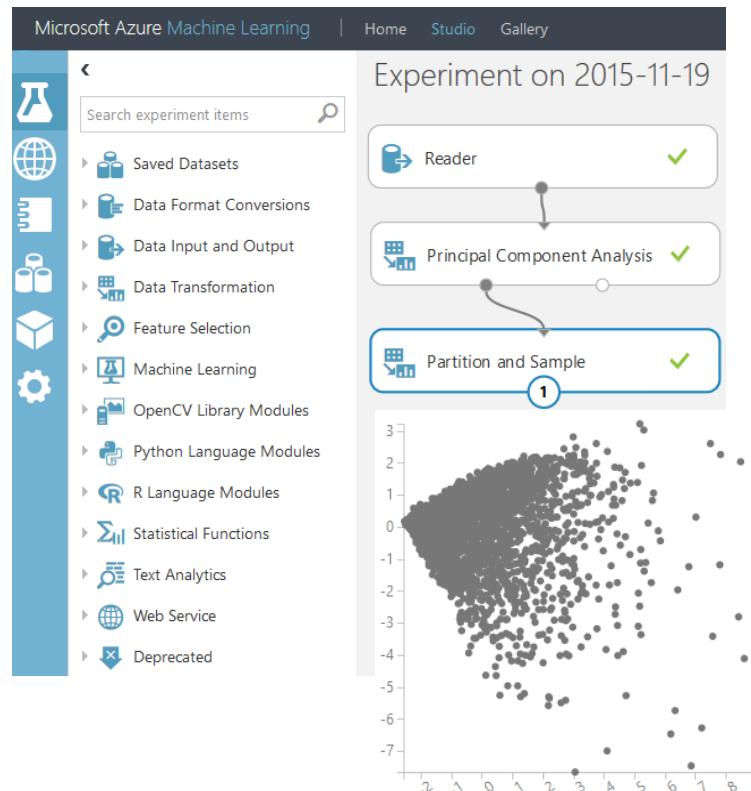


Visual Statistics	Create and modify predictive models faster than the using a visual interface and in-memory processing.
Visual Analytics	Visually explore all data, discover new patterns and publish reports to the web and mobile devices.

- **Oracle** Business Analytics is analytics for insight and innovation. **Oracle Data Visualization Cloud Service** get instant clarity with stunningly visual analysis and self-service discovery.

Visual Analytics as a Web Services

Microsoft Azure is an open, flexible, enterprise-grade cloud computing platform. **Microsoft Azure Machine Learning** is a service to build predictive analytics models and to easily deploy those models for consumption as cloud web services.



Platforms for Visual Analytics

Pentaho, a Hitachi Group Company, is a leading data integration and business analytics company with an enterprise-class, open source-based platform for diverse **big data deployments**.



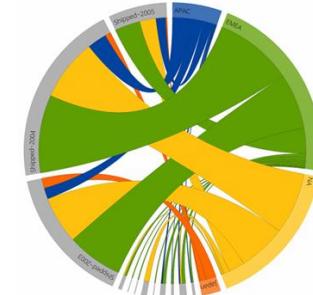
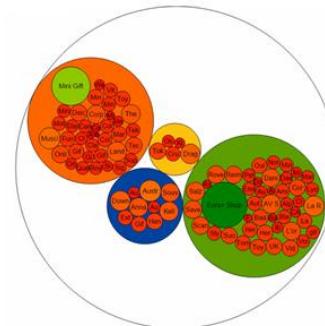
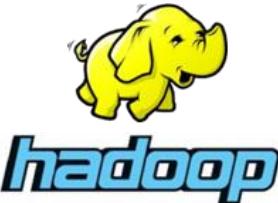
Blended
big data
analytics

Discovery
and
visualiza-
tions

High-
volume
data
processing

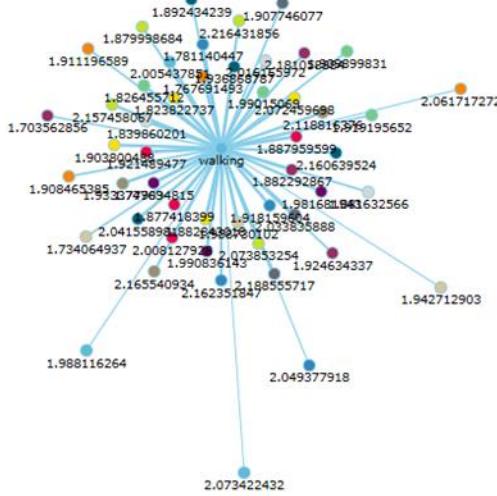
Adaptive
big data
layer

Predictive
analytics &
data
mining



Platforms for Visual Analytics

Amazon Web Services offers a broad set of global compute, storage, database, analytics, application, and deployment services.

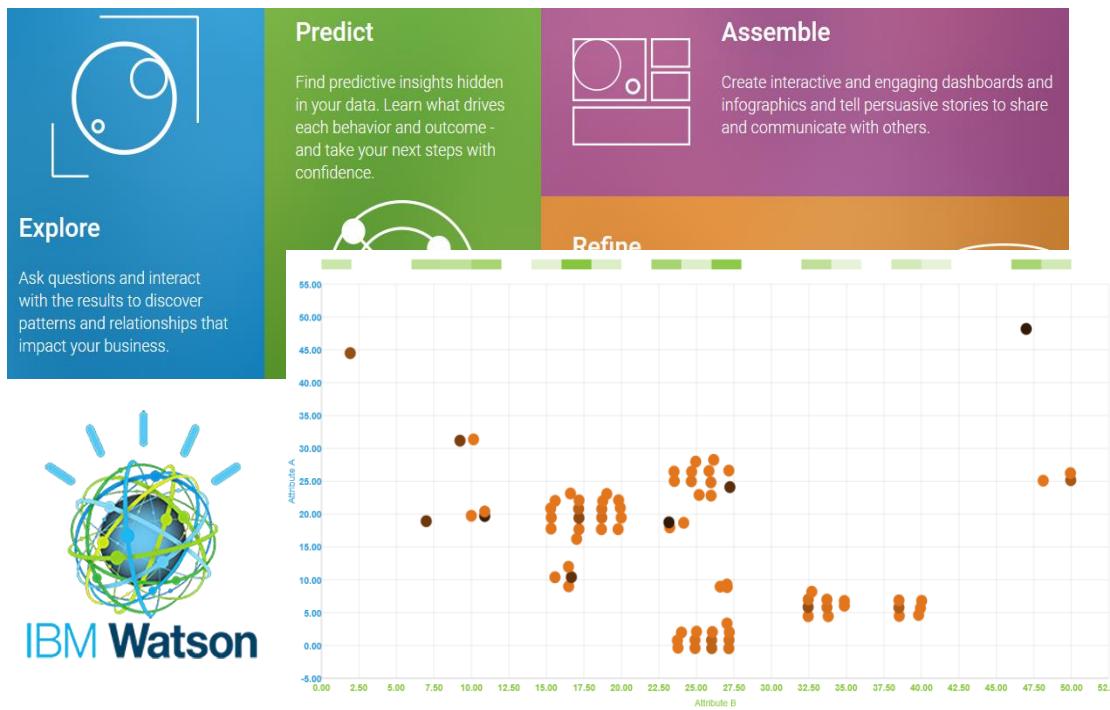


Datameer is big data analytics platform for Hadoop that empowers business users to directly integrate, analyze, and visualize any data.



Cloud based Data Exploration

- **IBM Watson** is a technology platform that uses **natural language processing** and machine learning to reveal insights from large amounts of unstructured data.
- **IBM Watson Analytics** offers the benefits of advanced analytics without the complexity.



Dirbtiniai neuroniniai tinklai ir didieji duomenys

- Dirbtiniai neuroniniai tinklais grįsti metodai **suteikia galingus įrankius** sprendžiant didžiujų duomenų iššūkius.
- Evoliuciniai skaičiavimai, neuroniniai tinklai, neraiškiosios sistemos **geba susidoroti su dideliais kiekiais neapibrėžtumu**, susijusių su didžiujų duomenų įvairumu ir nepastovumu.
- Tačiau didžiujų duomenų kiekiai **kelia didelius iššūkius** ir dirbtiniai neuroniniai tinklais grįstų metodų efektyviam taikymui.





Vilniaus universitetas
Matematikos ir informatikos fakultetas
Informatikos katedra



Duomenų analizė

prof. dr. Olga Kurasova
Olga.Kurasova@mii.vu.lt

2018

Duomenys ir jų analizė

- Tarkime turime **objektus**, kuriuos apibūdina tam tikri **požymiai**.
- **Objektais gali būti** pacientai, įrenginiai, gamybos procesai, gamtos reiškiniai ir kt.
- Objektus žymėkime X_1, X_2, \dots, X_m , o požymius x_1, x_2, \dots, x_n . Čia m – objektų skaičius analizuojamoje aibėje, n – juos apibūdinančių požymių skaičius.
- Tam tikras visų požymių reikšmių rinkinys, nusako vieną konkretų analizuojamos aibės objekta $X_i = (x_{i1}, x_{i2}, \dots, x_{in}), i \in \{1, \dots, m\}$.

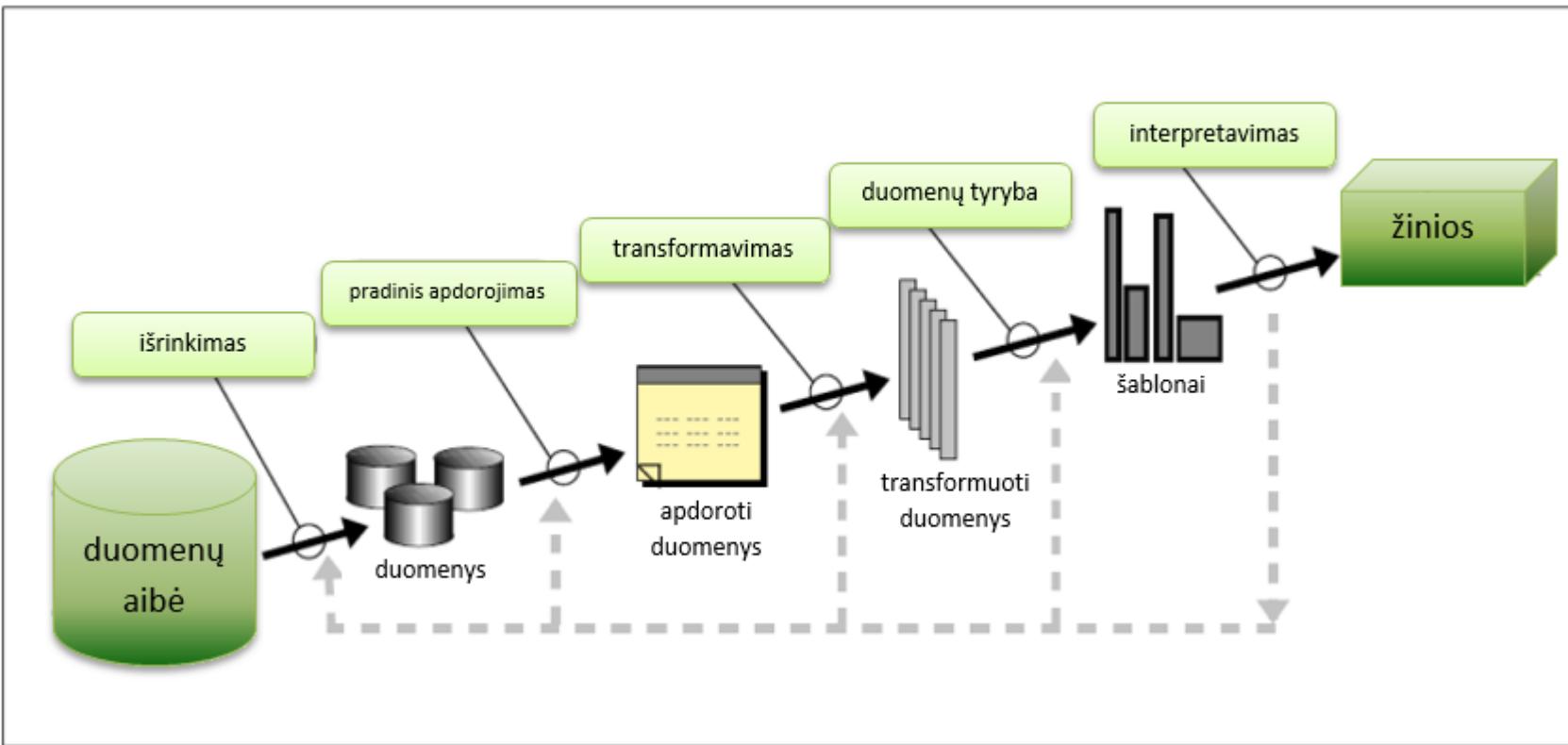
Duomenų analizė

- Vienas iš dažniausiai sprendžiamų uždavinių, pasitelkus skaitmeninio intelekto metodus, yra **duomenų analizė**.
- **Duomenų analizė** – tai procesas, kurio metu iš pradinių duomenų, juos apdorojant įvairiais metodais, gaunama naudinga informacija ir žinios.
- **Duomenų analizė** – tai duomenų nagrinėjimo procesas, kuriuo siekiama išryškinti naudingą informaciją, padaryti išvadas ir padėti sprendimų priėmėjui įgyti naujų žinių.
- Čia svarbios trys sąvokos – **duomenys, informacija, žinios**.

Duomenys, informacija, žinios

- **Duomenys** – tai objektyviai egzistuojantys faktai, vaizdai, garsai, kurie gali būti naudingi tam tikram uždaviniui spręsti.
- **Informacija** – tai duomenys, kurių forma ir turinys yra pateikti būdu, tinkamu naudoti sprendimų priėmimo procese. Duomenys virsta informacija, kai jiems suteikiamas kontekstas ir jie susiejami su tam tikra problema ar sprendimu.
- **Žinios** – tai gebėjimas spręsti problemas, atnaujinti arba sukurti naujas vertes remiantis ankstesne patirtimi, įgūdžiais ar išmokimu. Tai žmogaus proto abstrakcija apie duomenis, jų prasmę, naudą ir sąryšius. Turimos žinios gali virsti informacija, kuri gali būti panaudota naujoms žinioms įgyti.

Duomenų analizė (tyryba) žinių radimo procese



Data Mining in Knowledge Discovery in
Databases

Žinių radimo procesą sudarantys žingsniai:

- Iš visos duomenų aibės **išrenkami** analizuojami duomenys;
- Atliekamas **pradinis duomenų apdorojimas** (valymas, filtravimas, transponavimas, požymių atrinkimas, normavimas);
- Atliekamas **duomenų transformavimas**, kurio metu duomenys paruošiami duomenų analizės metodui ir programinei įrangai tinkama forma;
- **Duomenys analizuojami** įvairias duomenų analizės (tyrybos) metodais;
- Interpretuojami ir vertinami gauti rezultatai, ko pasėkoje išyjamos **naujos žinios**.

Duomenys ir jų analizė

- Apsiribosime duomenų analize, kai požymiai įgyja tam tikras **skaitines reikšmes**. Tuomet duomenų aibė yra matrica (lentelė)

$$X = \{X_1, X_2, \dots, X_m\} = \{x_{ij}, i = 1, \dots, m, j = 1, \dots, n\}.$$

kurios i -oji eilutė yra vektorius (taškas) $X_i \in R^n$,
čia $X_i = (x_{i1}, x_{i2}, \dots, x_{in}), i \in \{1, \dots, m\}$.

		Požymiai			
		x_1	x_2	...	x_n
Objektai	X_1	x_{11}	x_{12}	...	x_{1n}
	X_2	x_{21}	x_{22}	...	x_{2n}

	X_m	x_{m1}	x_{m2}	...	x_{mn}

Duomenys ir jų analizė

Duomenų bazės terminais:

- Duomenų lentelės **eilutės** (objektai) atitinka **įrašus**,
- **Stulpeliai** (požymiai) atitinka **atributus** (laukus).

Duomenų pradinis apdorojimas

Duomenų valymas –

tai veikla, kurios metu patikrinama, ar duomenyse **nėra trūkstamų reikšmių (*missing value*)**,

nustačius, kad tokių reikšmių yra, joms arba **priskiriamos tam tikros reikšmės**, pavyzdžiui, to požymio reikšmių vidurkis,

arba objektai su trūkstamomis reikšmėmis iš analizuojamos duomenų aibės yra **pašalinami**.

Duomenų pradinis apdorojimas

- **Duomenų filtravimas** – tai tam tikromis savybėmis pasižyminčių objektų atmetimas iš nagrinėjamos duomenų aibės.
- Duomenys gali būti filtruojami pagal tam tikras **taisykles**, pavyzdžiui, paliekamos tik tos reikšmės, kurios yra didesnės (ar mažesnės) už nustatyta dydį.
- Tokiu būdu iš duomenų **atmetamos išskirtys** (outliers) – nuo pagrindinės duomenų masės labai nutolusius ir prieštaraujančius jos tendencijoms objektus.
- **Pavyzdžiui**, skaičiuojant gyventojų vidutinę atlyginimą tikslinė atmeti labai didelę atlyginimą gaunančių asmenų įrašus, kadangi tai gali stipriai iškreipti vidurkį.

Duomenų pradinis apdorojimas

- **Duomenų požymių atrinkimas** – tai naujos duomenų aibės, sudarytos tik iš pasirinktų analizuojamų duomenų požymių reikšmių, suformavimas.
- **Duomenų normavimas** – tai procesas, kurio metu suvienodinami duomenų masteliai. Galimi įvairūs normavimo būdai. Dažniausiai:
 1. požymių reikšmės pakeičiamos taip, kad jų vidurkiai būtų lygūs 0, o dispersijos 1;
 2. požymių reikšmės pakeičiamos taip, kad jų minimali reikšmė būtų 0, maksimali 1.

Pagrindiniai duomenų analizės uždaviniai

- **Klasifikavimas** – duomenų priskyrimas klasėms;
- **Atpažinimas** – pagal turimą informaciją ir žinias, atpažįstami žinomi objektai.
- **Prognozavimas** – duomenų reikšmių numatymas;
- **Klasterizavimas** – duomenų suskirstymas į grupes (klasterius) pagal jų panašumą.

Duomenų klasifikavimas (1)

- **Klasifikavimas** – vienas dažniausiu duomenų analizėje sprendžiamų uždavinių.
- Klasifikavimo tikslas – **priskirti** duomenis tam tikrai **klasei**.
- Iprastai daliai duomenų klasės yra žinomos. Pritaikius klasifikavimo metodą, klasės yra **nustatomos duomenims**, kurių klasės nebuvo žinomos.

Duomenų klasifikavimas (2)

- Klasifikavimo uždaviniai dažnai sprendžiami **medicinoje**, siekiant nustatyti **preliminarią diagnozę**.
- Tarkime, turime pacientų kraujo tyrimų duomenis ir žinome, kad dalis pacientų serga tam tikra liga, kiti pacientai yra sveiki. Vadinas, turime dviejų klasių duomenis: **sergantys, sveiki**.
- Taip pat turime pacientus, kurių kraujo tyrimo rezultatai yra žinomi, tačiau **jie nėra priskirti** nei vienai klasei.
- **Klasifikavimo tikslas** – priskirti šiuos pacientus vienai iš dviejų klasių.

Duomenų klasifikavimas (3)

	x_1	x_2	x_3	x_4	Klasė
X_1	85	0,001	24,1	1025	sveikas
X_2	77	0,002	21,3	2036	sveikas
X_3	68	0,015	35,8	1059	sveikas
...
X_{101}	101	0,001	22,4	3011	serga
X_{102}	95	0,001	28,0	2645	serga
...
X_{201}	86	0,002	30,1	2987	???
X_{202}	72	0,010	19,5	1259	???

Atpažinimo uždavinys

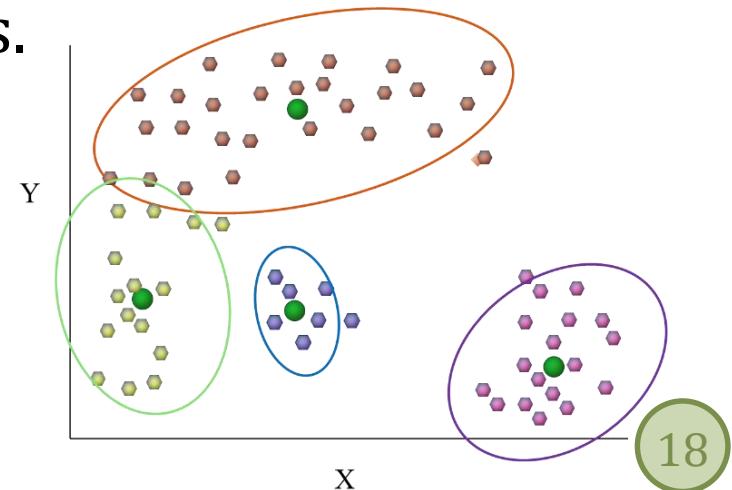
- Vienas iš klasifikavimo uždavinių – yra **atpažinimo uždavinys**.
- Čia duomenys dažniausiai yra **vaizdai** arba **garsai** (signalai).
- Atpažinimo uždavinio tikslas – pagal apmokytus duomenis kitus duomenis priskirti tam tikroms klasėms (**juos atpažinti**).
- **Charakteringi pavyzdžiai** – ranka rašyto teksto atpažinimas, veido atpažinimas, automobilio numerio atpažinimas.

Prognozavimo uždavinys

- Dar vienas populiarus duomenų analizės uždavinys yra **prognozavimas**, kurio metu žinant duomenų dalies požymį reikšmes, nustatomos reikšmės požymiui, kurio reikšmė nežinoma.
- **Prognozavimo tikslas** – iš „istorinių“ duomenų nustatyti reikšmes „ateities“ duomenims.
- Prognozavimo uždaviniai sprendžiami įvairiose srityse. **Pavyzdžiui**, meteorologijoje remiantis ankstesnių metų to paties laikotarpio duomenimis bei pastarojo laikotarpio pokyčių prognozuojama oro temperatūra ateinančiai savaitei.
- Taip pat prognozavimas atliekamas vertybinių popierių biržoje, įvairiose finansinėse rinkose ir kitur.

Duomenų klasterizavimas

- **Klasterizavimo** tikslas – suskirstyti objektus taip, kad panašūs objektai patektų į tą patį klasterį, o skirtini – į skirtingus.
- **Klasteris** – tai panašių objektų grupė.
- Čia svarbu nustatyti objektų **panašumo matą**. Vienas paprasčiausių panašumo matų – gerai žinomas Euklido atstumas.





Vilniaus universitetas
Matematikos ir informatikos fakultetas
Informatikos katedra



Tiesioginio sklidimo dirbtiniai neuroniniai tinklai

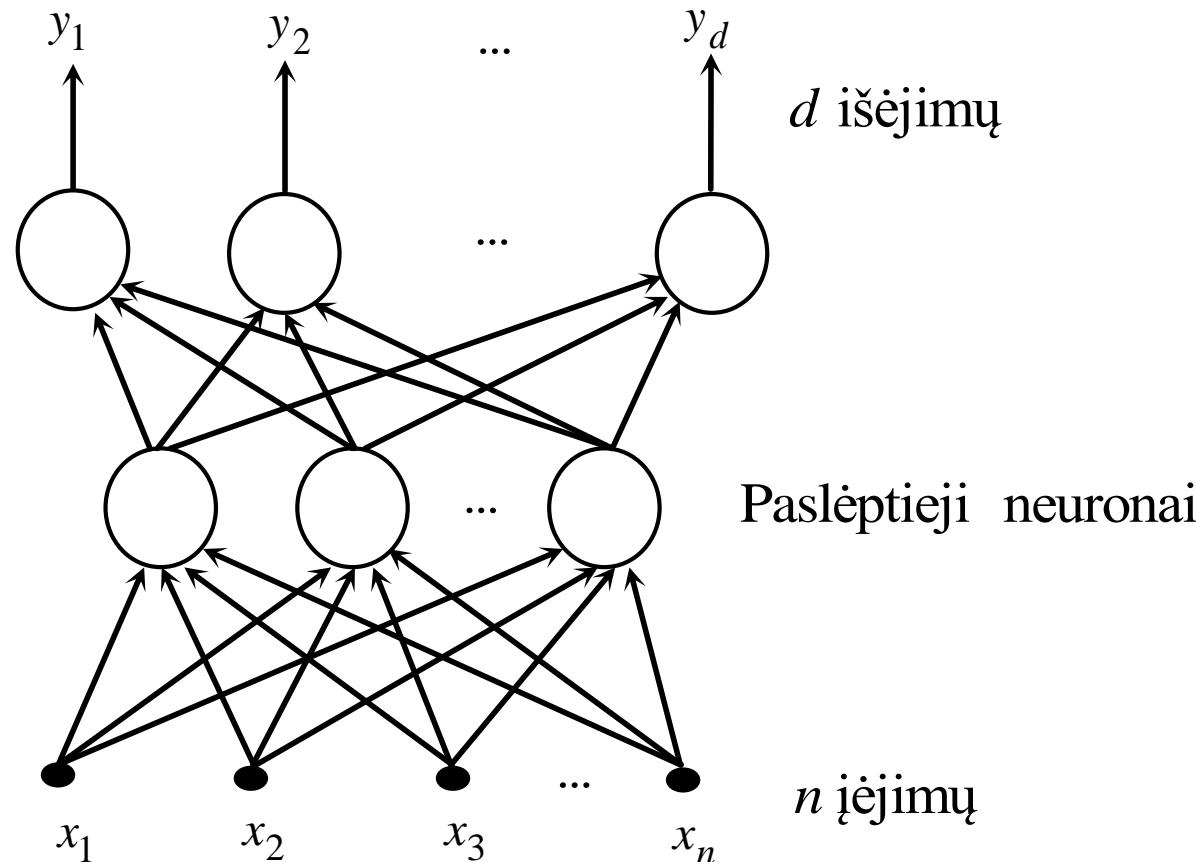
prof. dr. Olga Kurasova
Olga.Kurasova@mii.vu.lt

2018

Daugiasluoksniai tiesioginio sklidimo neuroniniai tinklai

- Turintys daugiau nei vieną neuronų sluoksnį tinklai, kuriuose galimi tik ryšiai iš priekės iš jėjimų išėjimus, yra vadinami
 - **daugiasluoksniais perceptronais** (*multilayer perceptrons*),
 - arba **daugiasluoksniai tiesioginio sklidimo neuroniniai tinklai** (*multilayer feedforward neural networks*).

Tiesioginio sklidimo DNT (1)



Kiekvienas neuronų sluoksnis turi po papildomą išėjimą ir jo jungtis su to sluoksnio neuronais, tačiau paprastumo dėlei jie paveiksle nėra pavaizduoti.

Tiesioginio sklidimo DNT (2)

- Tegul turime daugiasluoksnį neuroninį tinklą, kuriame yra **L sluoksniai**, pažymėtų $l = 0, 1, \dots, L$, čia
 - sluoksnis $l = 0$ žymi **įėjimus**,
 - $l = 1, \dots, L-1$ žymi **paslėptus sluoksnius**,
 - o $l = L$ paskutinį (**išėjimų**) sluoksnį.
- Kiekviename sluoksnje l yra n_l neuronų.

Tiesioginio sklidimo DNT (3)

- Pirmojo sluoksnio j -ojo neurono išėjimo reikšmė y_j yra apskaičiuojama pagal formulę.

$$y_j = f(a_j) = f\left(\sum_{k=0}^n w_{jk} x_k\right), \quad j = 1, \dots, d$$

- čia w_{jk} yra jungties iš k -ojo iėjimo į j -ąjį neuroną svoris.
- Iėjimai į neuronus l -ajame sluoksnje yra neuronų išėjimai $(l - 1)$ -ajame sluoksnje.

Tiesioginio sklidimo DNT (4)

- Todėl kiekvieno j -ojo neurono išėjimo reikšmė l -ajame sluoksnyje yra apskaičiuojama taip:

$$y_j = f(a_j) = f\left(\sum_{k=0}^{n_{l-1}} w_{jk} y_k\right), \quad j = 1, \dots, n_l$$

- čia $f()$ yra neuronų aktyvacijos funkcija,
- w_{jk} – svoriai jungčių, kurios jungia k -ąjį neuroną $(l-1)$ -ajame sluoksnyje su j -uoju neuronu l -ajame sluoksnyje,
- n_{l-1} – neuronų skaičius $(l-1)$ -ajame sluoksnyje, $y_0 = 0$.
- Kairiojoje lygybės pusėje y_j yra l -ojo sluoksnio j -ojo neurono išėjimo reikšmė,
- o dešiniojoje – y_k yra $(l-1)$ -ojo sluoksnio k -ojo neurono išėjimo reikšmė.

„Klaidos skleidimo atgal“ mokymo algoritmas (1)

- Taikant vienasluoksnio perceptrono mokymo idėją daugiasluoksniam neuroniniam tinklui, **būtina žinoti paslėptujų neuronų išėjimų reikšmes.**
- Jei paklaidos ir aktyvacijos funkcijos yra diferencijuojamos, ieškant minimalios paklaidos gali būti **naudojama gradientinio nusileidimo strategija.**
- Algoritmas, kuris realizuoja gradientinio nusileidimo mokymo strategiją daugiasluoksniam tiesioginio sklidimo neuroniniam tinklui, vadinas „**klaidos skleidimo atgal**“ **algoritmu** (*back-propagation learning algorithm*).

„Klaidos skleidimo atgal“ mokymo algoritmas (2)

- Algoritmą sudaro du žingsniai:
 - įėjimų reikšmių „**skleidimas pirmyn**“ iš įėjimų į išėjimų sluoksnį;
 - paklaidos „**skleidimas atgal**“ iš išėjimų į įėjimų sluoksnį.
- Tiek perceptrono mokyme, tiek „klaidos skleidimo atgal“ algoritme naudojama **mokymo su mokytoju strategija**.

„Klaidos skleidimo atgal“ mokymo algoritmas (3)

- Pirmame algoritmo žingsnyje **įėjimų** vektoriui X_i apskaičiuojamas **išėjimų** vektorius $Y_i = (y_{i1}, y_{i2}, \dots, y_{id})$.
- Ivertinama **paklaidos funkcija** $E_i(W)$ išėjimų sluoksnyje L .

$$E_i(W) = \frac{1}{2} \sum_{j=1}^d (y_{ij} - t_{ij})^2$$

- čia t_{ij} – norimo j -ojo išėjimo reakcija į vektorių X_i .
- Tuo baigiamas „**skleidimo pirmyn**“ fazę.

„Klaidos skleidimo atgal“ mokymo algoritmas (4)

- Jei paklaidos funkcija $E_i(W)$ nelygi nuliui, reikia keisti jungčių svorius.
- Panašiai kaip ir vienasluoksniaiame perceptrone, visi svoriai w_{jk} jungčių, kurios jungia k -ajį neuroną ($l - 1$)-ajame sluoksnyje su j -uoju neuronu l -ajame sluoksnyje, keičiami naudojantis formule

$$\Delta w_{jk}^i = -\eta \frac{\partial E_i}{\partial w_{jk}}$$

„Klaidos skleidimo atgal“ mokymo algoritmas (5)

- Dalinės išvestinės išreiškiamos taip:

$$\frac{\partial E_i}{\partial w_{jk}} = \frac{\partial E_i}{\partial a_{ij}} \cdot \frac{\partial a_{ij}}{\partial w_{jk}}$$

- Kadangi $a_{ij} = \sum_{k=0}^{n_{l-1}} w_{jk} y_{ik}$, tai $\frac{\partial a_{ij}}{\partial w_{jk}} = y_{ik}$.
- Tegul $\delta_{ij} = \frac{\partial E_i}{\partial a_{ij}}$, tuomet $\frac{\partial E_i}{\partial w_{jk}} = \delta_{ij} y_{ik}$
- ir $\Delta w_{jk}^i = -\eta \delta_{ij} y_{ik}$,
- čia j -asis neuronas priklauso l -ajam sluoksniui,
 k -asis neuronas priklauso $(l-1)$ -ajam sluoksniui.

„Klaidos skleidimo atgal“ mokymo algoritmas (6)

- Išėjimų sluoksnyje

$$\delta_{ij} = \frac{\partial E_i}{\partial a_{ij}} = f'(a_{ij})(y_{ij} - t_{ij})$$

- čia j -asis neuronas priklauso išėjimų sluoksniniui L .
- Turime rasti $\delta_{ij} = \frac{\partial E_i}{\partial a_{ij}}$ **paslėptiems neuronas**, t. y. kai j -asis neuronas priklauso l -ajam sluoksniniui ir $l < L$.

„Klaidos skleidimo atgal“ mokymo algoritmas (7)

- Naudojantis dalinėmis išvestinėmis, bendruoju atveju galima parašyti

$$\delta_{ij} = \frac{\partial E_i}{\partial a_{ij}} = \sum_{s=1}^{n_{l+1}} \frac{\partial E_i}{\partial a_{is}} \cdot \frac{\partial a_{is}}{\partial a_{ij}}$$

- čia n_{l+1} žymi neuronų $(l + 1)$ -ajame sluoksnyje skaičių.
- Išraiška $\frac{\partial E_i}{\partial a_{is}}$ yra lygi dydžiui δ_{is} , apibrėžtam s -ajam neuronui $(l + 1)$ -ajame sluoksnyje.

„Klaidos skleidimo atgal“ mokymo algoritmas (8)

- Turint
- $$\frac{\partial a_{is}}{\partial a_{ij}} = f'(a_{ij})w_{is}$$

- Paslėptuji j -ujų neuronų

$$\delta_{ij} = f'(a_{ij}) \sum_{s=1}^{n_{l+1}} w_{is} \delta_{is}$$

- čia j -asis neuronas priklauso sluoksniui $l < L$,
 s -asis neuronas priklauso $(l + 1)$ -ajam
sluoksniui.

„Klaidos skleidimo atgal“ mokymo algoritmas (9)

Apibendrinimas:

- Iš pradžių reikia apskaičiuoti δ_{ij} reikšmes **išėjimų sluoksnyje** L .
- Tada palaipsniui skaičiuoti δ_{ij} reikšmes **paslėptiems neuronams** tarpiniuose sluoksniuose $l < L$ naudojantis $(l + 1)$ -ujų sluoksnių δ_{ij} reikšmėmis.
- Kai visi svoriai pakeičiami, **į tinklą pateikiamas sekantis mokymo vektorius** ir procesas kartojamas iš naujo.

Algoritmo **sustojimo kriterijus** yra

- arba iš anksto nustatyta **paklaidos funkcijos slenksčio reikšmė**,
- arba atitinkamas **atliktų iteracijų** (mokymo žingsnių) skaičius.

„Klaidos skleidimo atgal“ mokymo algoritmas (10)

- Norint pagreitinti mokymo procesą, yra **koreguojama mokymo taisyklė** ir gaunama tokia svorių pokyčio formulė:

$$\Delta w_{jk}^i(t) = -\eta \delta_{ij}(t) y_{ik}(t) + \alpha \Delta w_{jk}^i(t-1)$$

- čia t yra iteracijos numeris,
- η – yra teigiamas daugiklis, kuris vadinamas **mokymo greičiu** (*learning rate*).
- α – teigiamai konstanta ($0 < \alpha \leq 1$), vadinamoji **momento konstanta** (*momentum constant*).



Vilniaus universitetas
Matematikos ir informatikos fakultetas
Informatikos katedra



Rašto ženklų atpažinimas taikant dirbtinius neuroninius tinklus

prof. dr. Olga Kurasova
Olga.Kurasova@mii.vu.lt

2018

Rašto ženklų atpažinimas

- **Ranka rašytų ženklų atpažinimas (*Handwriting Recognition*)** – tai kompiuterio gebėjimas gauti ir protingai interpretuoti ranka rašytus ženklus iš įvairių šaltinių, tokiu kaip popieriniai dokumentai, nuotraukos, liečiamieji ekranai ir kt.



Rašto ženklų atpažinimas taikant DNT

- Šiam uždaviniui spręsti taikomi įvairūs metodai.
- **Ranka rašytiems ženklams atpažinti** dažnai taikomi ir tiesioginio sklidimo DNT, mokomi „klaidos skleidimo atgal“ algoritmu.
- Čia **atpažinimo** uždavinys yra **klasifikavimo** į kelias klasses uždavinys.
- Pavyzdžiui, jei norima atpažinti tik **skaitmenis**, tai sprendžiamas **10-ies klasių** klasifikavimo uždavinys.

Duomenys klasifikavimui

Sprendžiant **klasifikavimo** uždavinius išskiriami **trijų tipų duomenys**:

- **mokymo duomenys** naudojami klasifikatoriui sukurti,
- **testavimo duomenys** naudojami patikrinti (testuoto) klasifikatoriaus išmokymo klasifikuoti lygi,
- **nauji duomenys**, kurių klasės nėra žinomas, bet taikant sukurta klasifikatorių jos yra nustatomos.

Skaitmenų atpažinimas taikant DNT (1)

- Sudaromas **neuroninis tinklas**, turintis
 - tiek **įėjimų**, kiek elementų (pixelių) sudaro vienas ranka rašytas ženklas (simbolis),
 - ir tiek **išėjimų**, kiek simbolių norima atpažinti (skaitmenų atveju – 10)
 - **paslėptų neuronų sluoksnių** ir neuronų skaičius juose nustatomas eksperimentiškai

Skaitmenų atpažinimas taikant DNT (2)

Duomenys **įėjimams**:

- Vienas **mokymo duomuo** – tai langelį, kuriame užrašytas ženklas, sudarančiu pikselių spalvų reikšmės, surašytose į vieną vektorių.
- Mokymo aibę turi sudaryti **visų ženklu**, kuriuos norima atpažinti, **egzemploriai**. Be to, turi būti kiekvieno ženklo nors po kelis (ar keliolika) egzempliorių.
- **Pavyzdžiui**, turint ženkla, kuris **juoda** spalva užrašytas **24 x 24 baltame** langelyje, mokymo duomenį, kuris bus pateikiamas į neuroninio tinklo **įėjimus**, sudarys vektorius iš **576** komponenčių.

Skaitmenų atpažinimas taikant DNT (3)

Norimos išėjimų reikšmės:

- Tai vektoriai, **sudaryti iš nulių**, išskyrus vieną poziciją, kuri atitinka norimą atpažinti simbolį. Toje pozicijoje yra išrašytas **vienetas**.
- Pavyzdžiui, jei norima **atpažinti skaitmenis**, tai vektorius (1 0 0 0 0 0 0 0 0) bus norima reikšmė, kai į tinklą bus pateiktas skaitmuo „0“, (0 1 0 0 0 0 0 0 0) – kai „1“ ir t. t.

Skaitmenų atpažinimas taikant DNT (4)

Gaunamos išėjimų reikšmės:

- Keičiant svorių reikšmes, tinklas turi būti išmokomas taip, kad **gautos** išėjimų reikšmės **sutaptų** su **norimomis** reikšmėmis.
- Tačiau 100 procentų sutapimą pasiekti beveik neįmanoma, todėl ženklas priskiriamas tai klasei, kurią atitinka **didžiausia išėjime** gauto vektoriaus komponentės **reikšmė**.
- Pavyzdžiui, jei gaunamas vektorius $(0,05 \ 0 \ 0 \ 0,8 \ 0 \ 0 \ 0 \ 0 \ 0,1 \ 0,05)$, tai skaičiuo priskiriamas **skaitmeniui „3“**.

Skaitmenų atpažinimas taikant DNT (5)

- Tinklas **išmokomas** daug kartų pateikiant mokymo duomenis.
- Viena **iteracija** – tai mokymo proceso dalis, kai į tinklą pateikiamas **vienas** mokymo aibės duomuo.
- Viena **mokymo epocha** – tai mokymo proceso dalis, kai į tinklą pateikiami **visi** mokymo aibės duomenys vieną kartą.
- Skaičiuojama **paklaida** (*sum squared error*)

$$\frac{1}{2} \sum_{i=1}^m \sum_{j=1}^d (y_{ij} - t_{ij})^2$$



Vilniaus universitetas
Matematikos ir informatikos fakultetas
Informatikos katedra



Dirbtiniai neuroniniai tinklai duomenims prognozuoti

prof. dr. Olga Kurasova
Olga.Kurasova@mii.vu.lt

2018

Neuroniniai tinklai prognozavimui

- Dar vienas duomenų analizės uždavinys – **duomenų prognozavimas** (*prediction, forecasting*).
- **Prognozavimo uždaviniai** ypač aktualūs ekonomikoje, finansuose, meteorologijoje ir kt.
- Turint vadinamuosius **istorinius duomenis**, reikia kiek galima tiksliau **numatyti** tam tikro požymio reikšmes ateityje.



Prognozavimo metodai

- Prognozavimo uždaviniai gali būti sprendžiami taikant įvairius **statistinius metodus**, pvz., regresija, slenkančio vidurkio, ARMA, ARIMA, SARIMA ir kt.
- **Tiesioginio sklidimo neuroniniai tinklai** taip pat yra sėkmingai taikomi prognozavimo uždaviniams spręsti.



Regresija – paprasčiausias prognozavimo metodas

- **Regresinės analizės** paskirtis – numatyti priklausomojo kintamojo reikšmę mažiausiai vieno nepriklausomojo kintamojo atžvilgiu ir paaiškinti, kaip nepriklausomo kintamojo pokyčiai veikia priklausomą kintamąjį.
- Turint tokią priklausomybę, galime **prognozuoti** priklausomo kintamojo reikšmes.
- Atliekant regresinę analizę priklausomas tarp dviejų kintamųjų yra išreiškiamas matematine lygtimi, kuri vadina **regresijos lygtimi**.
- Išskiriamos **tiesinė** ir **netiesinė** regresijos. Pirmuoju atveju priklausomas išreiškiamas **tiesės lygtimi**, o antruoju atveju kitokia lygtimi, pvz., polinomu, eksponente ir kt.

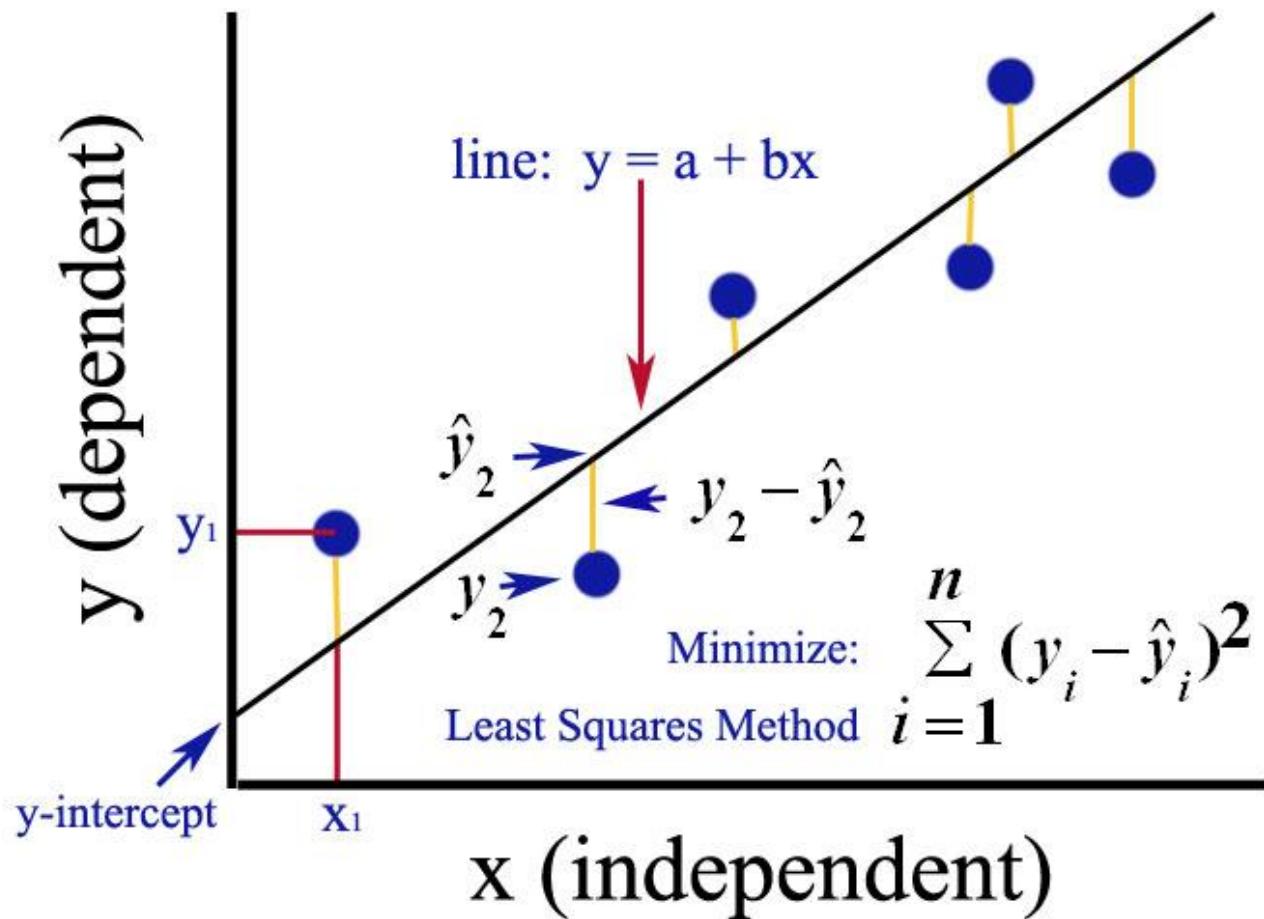
Tiesinė regresijos lygtis

- Jei yra viena priklausomas y ir vienas nepriklausomas kintamasis (požymis) x , tuomet **tiesinės regresijos lygtis** užrašoma taip:

$$y = a + bx + e$$

- Čia e yra atsitiktinė paklaida, atsirandanti dėl matavimo ar kitų duomenų gavimo paklaidų.
- Kai yra žinomi koeficientai a ir b , galima **prognozuoti**, kaip keisis priklausomojo požymio y reikšmės, keičiantis nepriklausomajam požymiui x .
- Naudodamiesi lygtimi galime paskaičiuoti, kaip keisis y reikšmės, esant tokioms x reikšmėms, kurių mes netyrėme, t. y., **galėsime prognozuoti** y reikšmes.

Tiesinė regresija grafiškai



Tiesinės regresijos lygties koeficientų radimas

- Tarkime, turime m stebėjimų metu gautas **duomenų poras** $(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$.
- **Tikslas** – rasti koeficientų a ir b įverčius \hat{a} ir \hat{b} tokius, kad funkcijos $\hat{y}(x) = \hat{a} + \hat{b}x$ reikšmės taškuose x_j kiek galima mažiau skirtūsi nuo y_j reikšmių.
- Gautoji funkcija bus naudojama priklausomojo kintamojo nežinomoms reikšmėms **prognozuoti**.
- Kiekvieną x_j atitinka y_j ir funkcijos $\hat{y}(x)$ reikšmės taškuose x_j .
- Geriausia tinkanti funkcija yra tokia, kurios **skirtumai** $\hat{e} = y_j - \hat{y}(x_j)$ būtų mažiausi, $j = 1, \dots, m$.

Tiesinės regresijos lygties koeficientų radimas

- Įverčiai \hat{a} ir \hat{b} randami vadinauoju **mažiausiu kvadratų metodu**, t. y., minimizuojama kvadratinių sumų paklaidos (KSP) funkcija:

$$\text{KSP} = \sum_{j=1}^m (y_j - \hat{y}(x_j))^2 = \sum_{j=1}^m (y_j - \hat{a} - \hat{b}x_j)^2$$

- Šią funkciją reikia minimizuoti pagal du parametrus \hat{a} ir \hat{b} , t. y., **skaičiuoti dalines išvestines** ir jas prilyginti nuliui.

Tiesinės regresijos lygties koeficientai

- Išsprendus **gautą lygčių sistemą** gaunami tokie sprendiniai:

$$\hat{a} = \frac{1}{m} \sum_{j=1}^m y_j - \hat{b} \frac{1}{m} \sum_{j=1}^m x_j$$

$$\hat{b} = \frac{\sum_{j=1}^m x_j y_j - \frac{1}{m} (\sum_{j=1}^m x_j) \sum_{j=1}^m y_j}{\sum_{j=1}^m x_j^2 - \frac{1}{m} (\sum_{j=1}^m x_j)^2}$$

Tiesinės regresijos įvertinimas

- Reikia nepamiršti, kad parametrai \hat{a} ir \hat{b} yra tik parametrų a ir b įverčiai, kurie bendru atveju gali ir nesutapti, t. y., gautis **liekamoji paklaida**, parodanti, kiek stebėtoji y_j reikšmė skiriasi nuo reikšmės, kurią gautume prognozuodami pagal regresijos tiesę.
- **Liekamųjų paklaidų kvadratų suma** (SSE), skaičiuojama pagal formulę:

$$SSE = \sum_{j=1}^m (y_j - \hat{y}(x_j))^2 = \sum_{j=1}^m (y_j - (\hat{a} + \hat{b}x_j))^2$$

Determinacijos koeficientas

- Dar dažnai vertinamas **determinacijos koeficientas** R^2 , įgyjantis reikšmes nuo 0 iki 1, t. y. $0 < R^2 \leq 1$. Idealiu atveju, $R^2 = 1$.

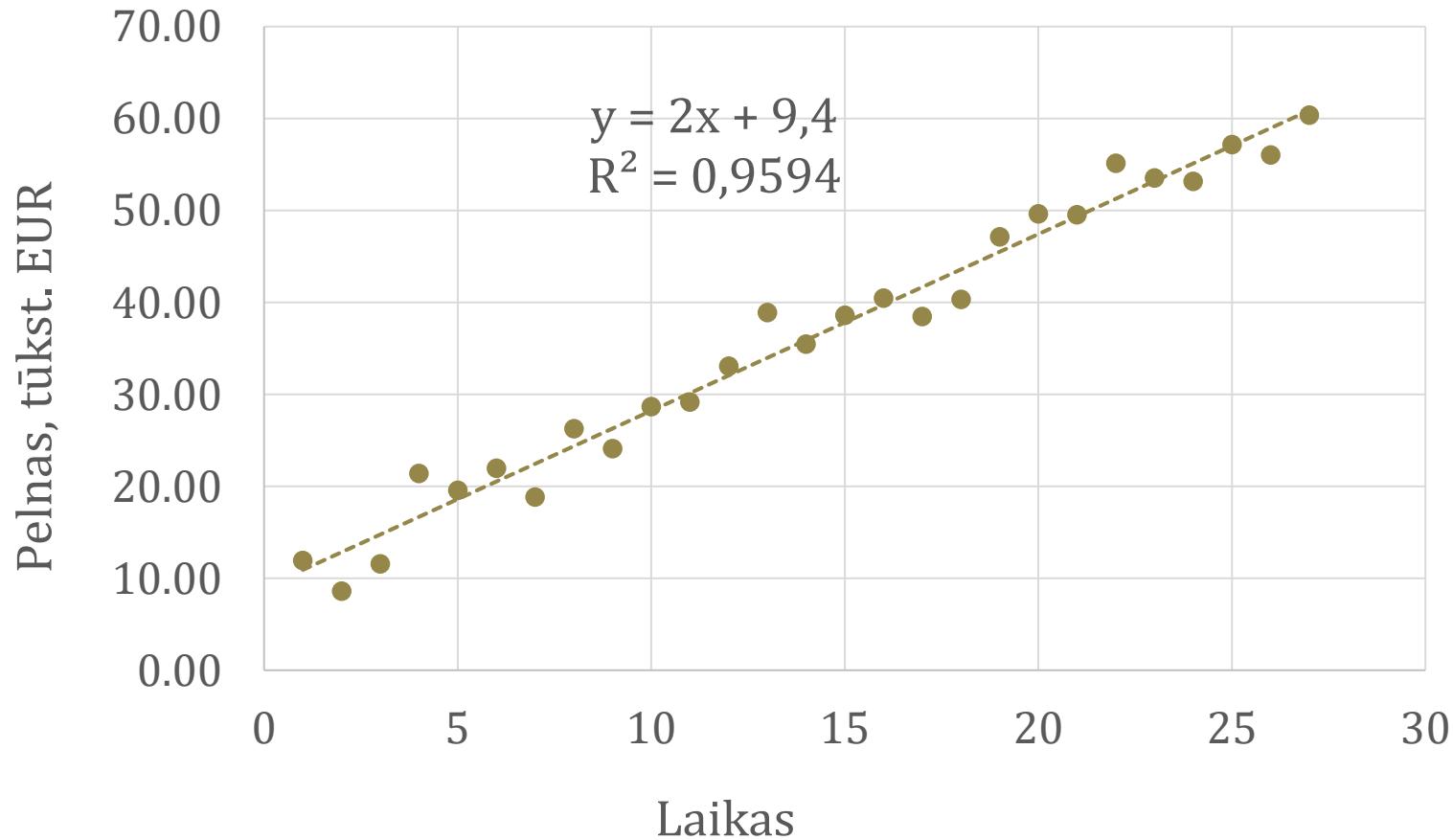
$$R^2 = \frac{\text{SSR}}{\text{SST}}$$

- Čia SSR – regresijos kvadratų suma, SST – visa kvadratų suma

$$\text{SST} = \sum_{j=1}^m \left(y_j - \frac{1}{m} \sum_{k=1}^m y_k \right)^2,$$

$$\text{SSR} = \sum_{j=1}^m \left(\hat{y}(x_j) - \frac{1}{m} \sum_{k=1}^m y_k \right)^2.$$

Tiesinė regresija



Tiesinė regresija kelių kintamujų atveju

- Jei ieškome sąryšio tarp vieno priklausomo kintamojo y ir kelių kitų nepriklausomų x_1, x_2, \dots, x_n , **turime praplėsti** prieš tai aptartą modelį.

- Tuomet **tiesinės regresijos** modelis yra aprašomas tokia lygtimi:

$$y = a + b_1x_1 + b_2x_2 + \cdots + b_nx_n + e,$$

- čia a, b_1, b_2, \dots, b_n yra **regresijos koeficientai**, e – atsitiktinė paklaida.

Kito tipo regresijos

Tiriant vieno nepriklausomo kintamojo x sąryšį nuo priklausomo kintamojo y , galimi šie **regresijos tipai**:

- **eksponentinė:**

$$y = ae^{bx} + e,$$

- **logaritminė:**

$$y = a \ln(x) + b + e,$$

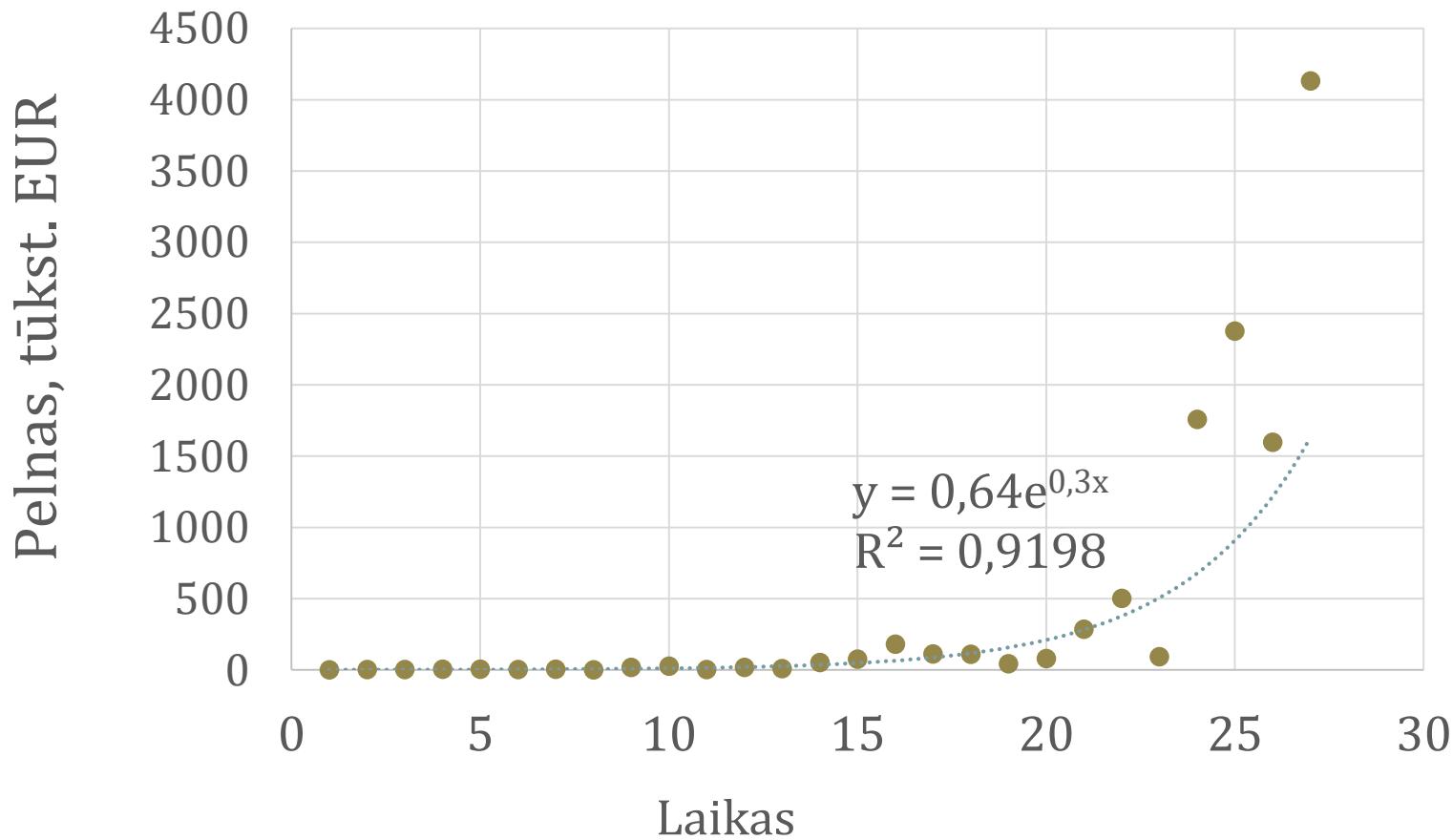
- **laipsninė (rodiklinė):**

$$y = ax^b + e,$$

- **polinominė (n -tojo laipsnio):**

$$y = a + b_1x + b_2x^2 + \cdots + b_nx^n + e.$$

Eksponentinė regresija



Laiko eilutės

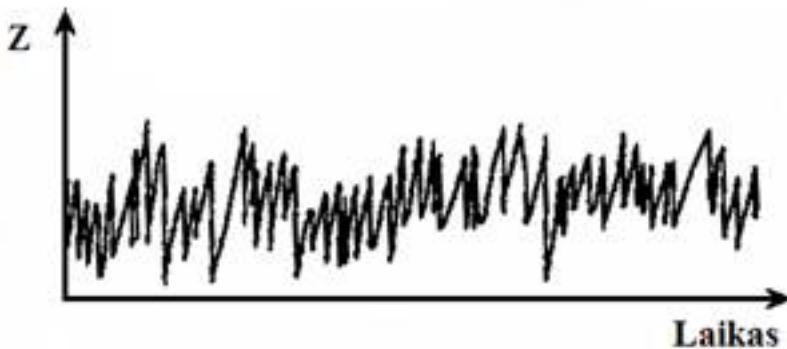
- Iprastai prognozavimo uždaviniai sprendžiami nagrinėjant vadinačias **laiko eilutes**.
- Tarkime tam tikro atsitiktinio dydžio X reikšmės stebimos laikui bėgant. Tokio atsitiktinio dydžio reikšmių seka (X_1, X_2, \dots, X_t) vadinaama **laiko eilute** (*time series*).
- Iprastai laikoma, kad yra žinomas reikšmės $X(t_i)$ laiko momentais $t_1 < t_2 < \dots < t_n$, o visi stebėjimai atliekami **vienodais laiko intervalais**, $t_{i+1} - t_i = \Delta t$.
- **Prognozavimo tikslas** – žinant reikšmes $X(t_1), X(t_2), \dots, X(t_n)$, nustatyti reikšmę $X(t_{n+1})$.

Laiko eilučių dedamosios

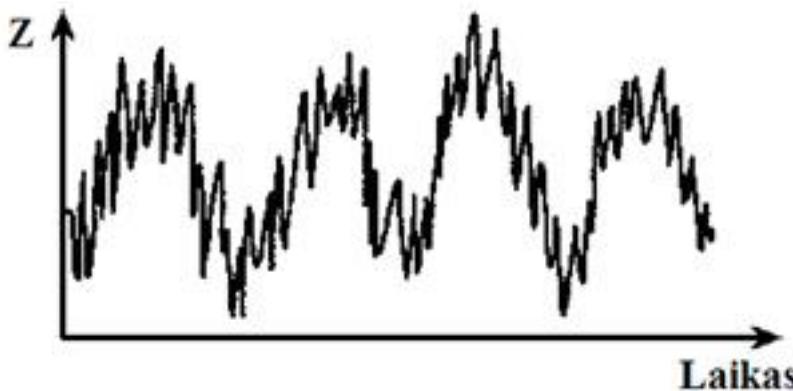
- Dažnai laiko eilutėse stebimos dvi dedamosios: **atsitiktinė** ir **apibrėžtoji**.
- **Apibrėžtosios** dedamosios dalys:
 - **trendas** (atspindi pagrindines bei ilgalaikes laiko eilutės tendencijas, esminius tiriamo proceso bruožus; trendas gali būti tiesinis, eksponentinis ir kt.),
 - **sezoniniai svyravimai** (reguliarus stebimo kintamojo reikšmių didėjimas bei mažėjimas griežtai apibrėžtais laiko periodais),
 - **cikliniai svyravimai** (yra panašūs į sezoninius, tačiau neturi tokio griežto matematinio aprašymo, jų pasikartojimo periodas nėra toks apibrėžtas).

Laiko eilučių dedamosios

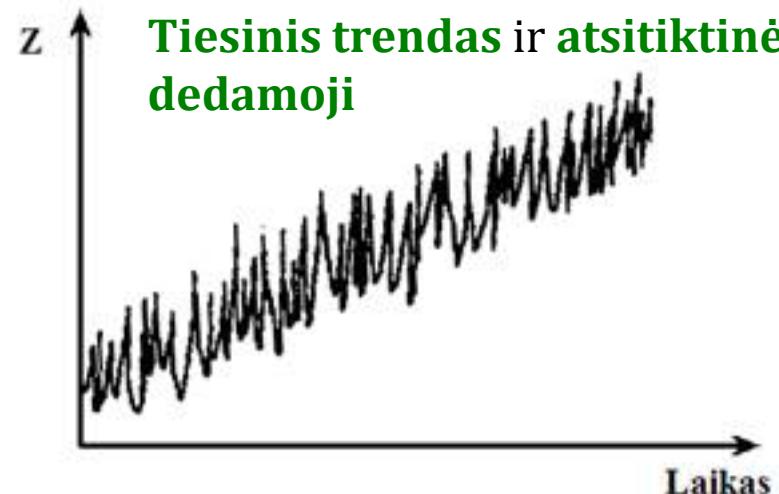
Vien tik **atsitiktinė dedamoji**



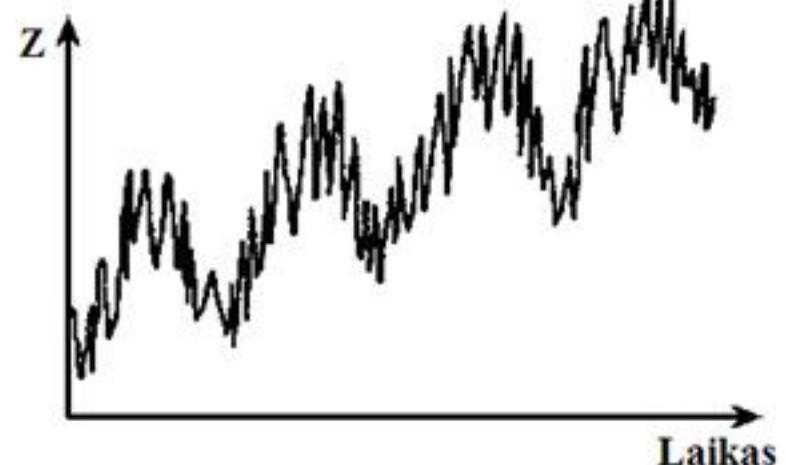
Sezoniniai svyravimai ir atsitiktinė dedamoji



Tiesinis trendas ir atsitiktinė dedamoji

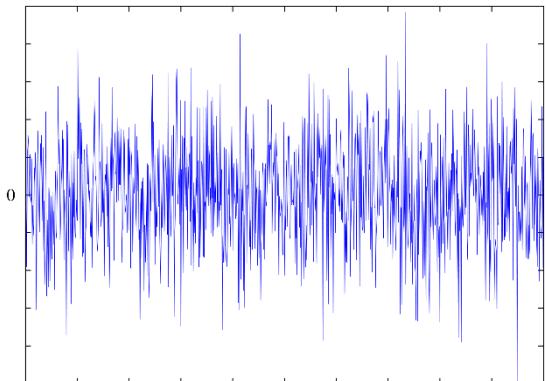


Tiesinis trendas, sezoniniai svyravimai ir atsitiktinė dedamoji

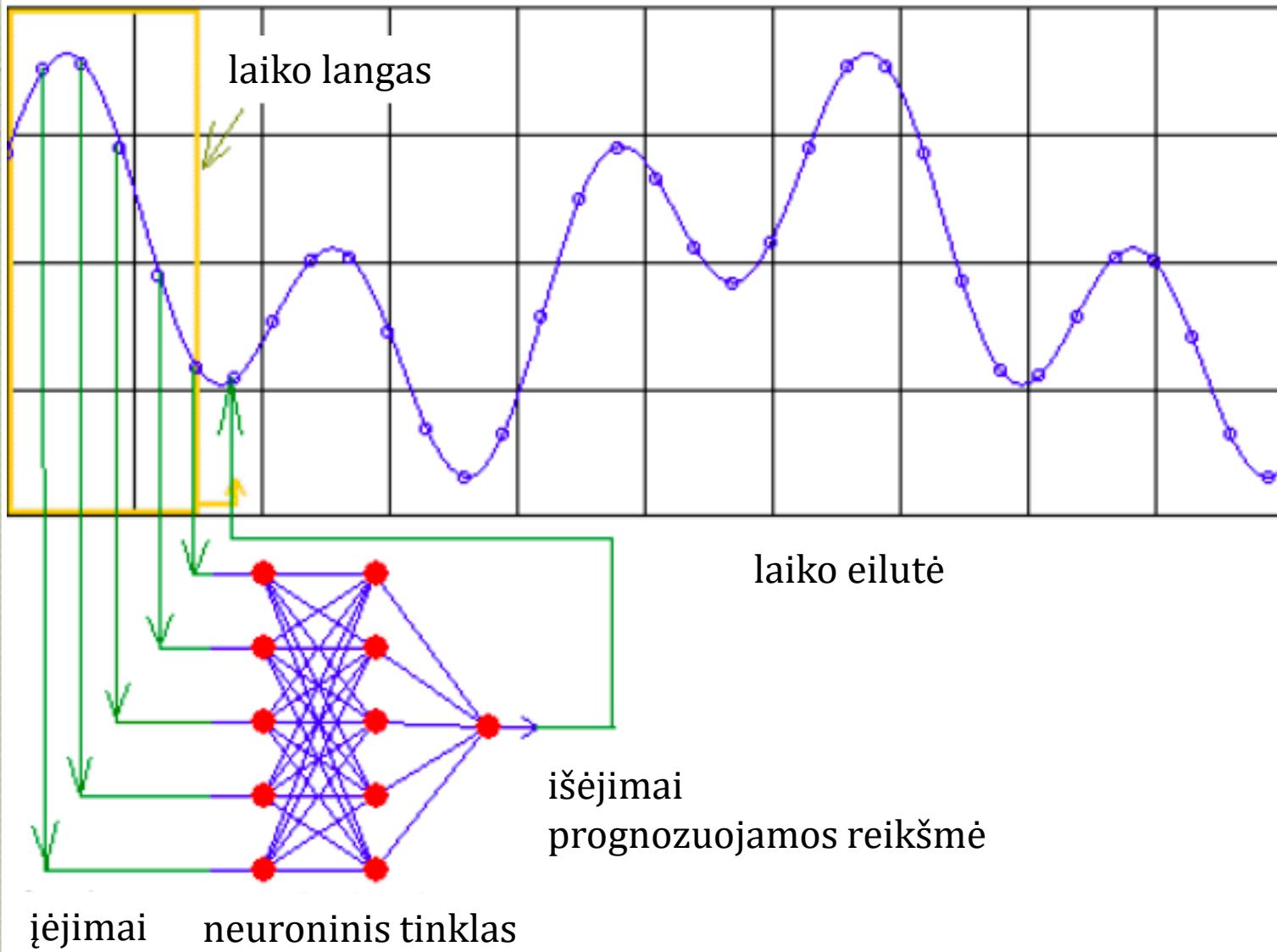


Kodėl DNT?

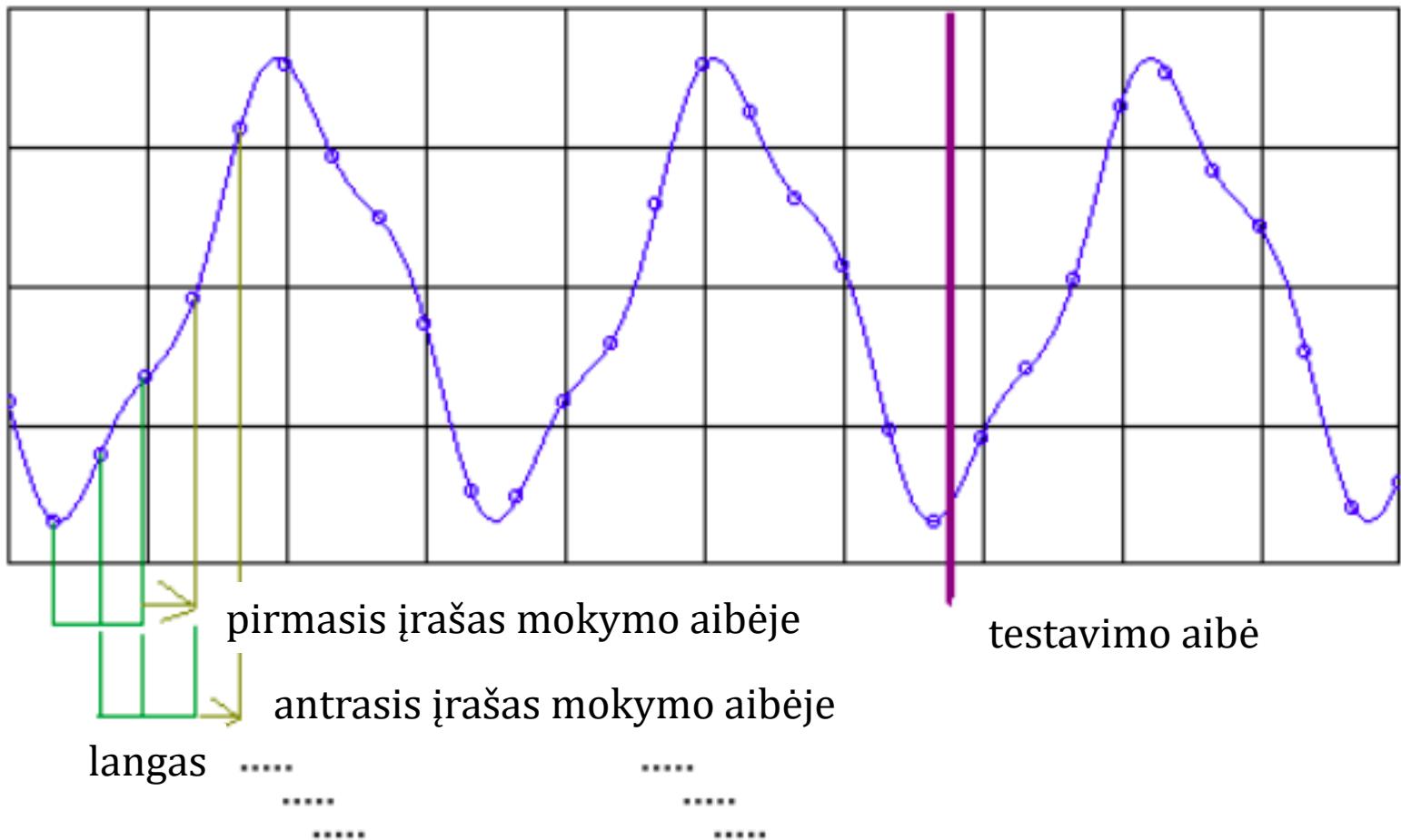
- DNT taikomi duomenis prognozuoti, kai statistiniai metodai **nepajėgūs to padaryti**.
- Sprendžiant realius uždavinius yra sunku nustatyti ar **tai triukšmas**, ar **tikros reikšmės**.
- Taikant **statistinius** prognozavimo metodus, būtina „**atpažinti**“ triukšmo tipą.
- **Baltasis triukšmas** – tai atsitiktinių dydžių seka, kurios vidurkis lygus nuliui, o standartinis nuokrypis lygus vienam.



DNT prognozavimui



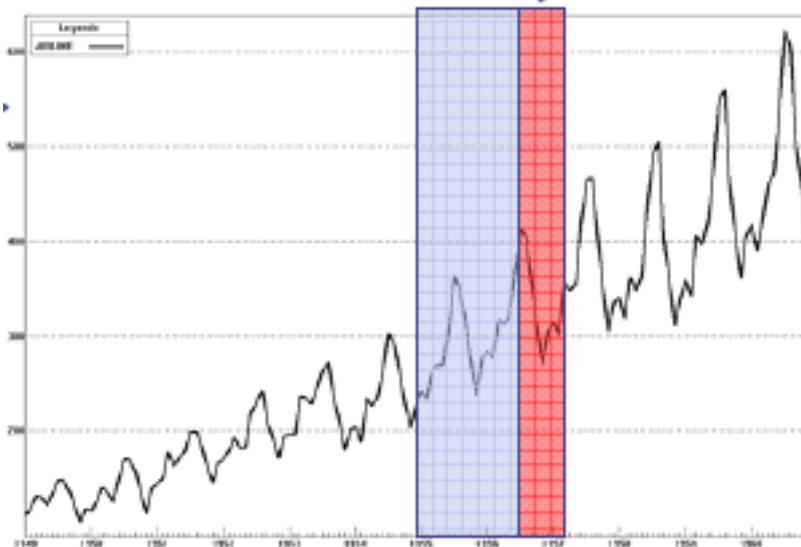
Mokymo ir testavimo duomenys



Prognozavimo pavyzdžiai

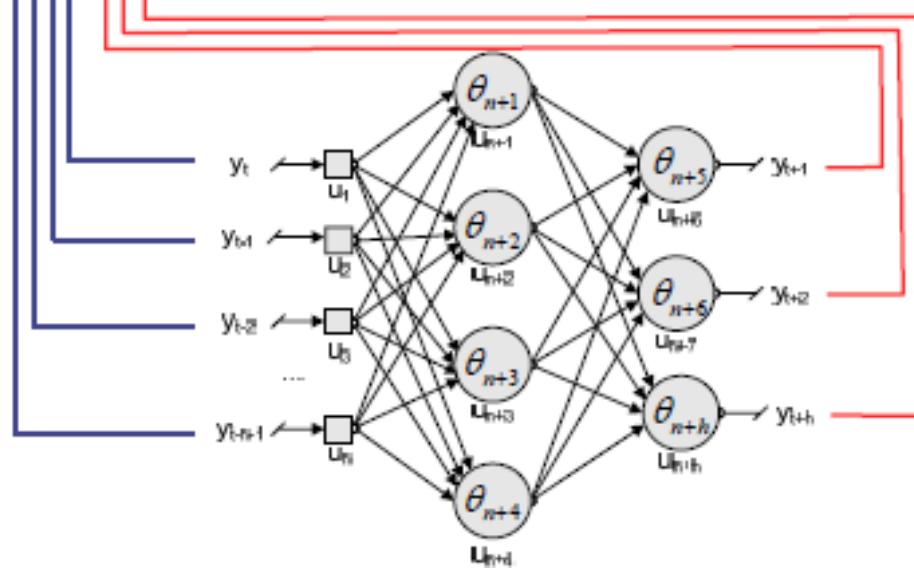
- Matematinės funkcijos atvejis:
 - <http://www.obitko.com/tutorials/neural-network-prediction/function-prediction.html>
- NASDAQ akcijų rinka:
 - <http://www.obitko.com/tutorials/neural-network-prediction/nasdaq-prediction.html>

DNT prognozuojantis kelis išėjimus

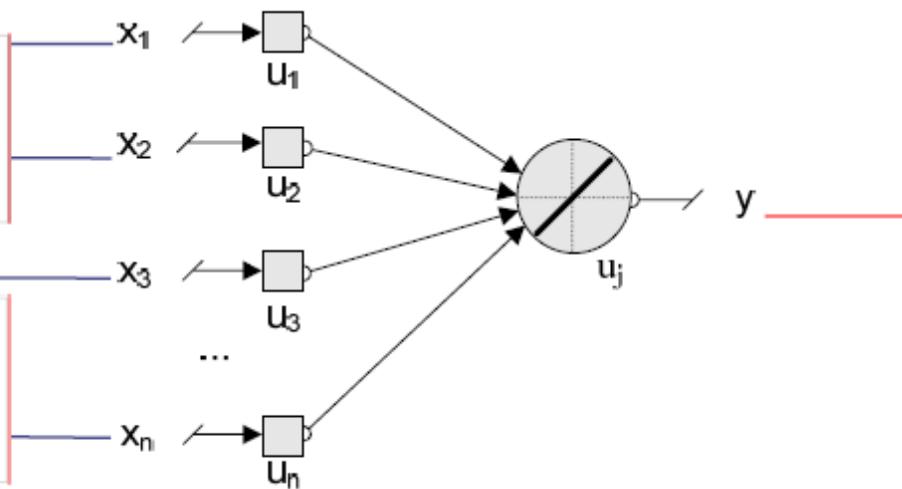
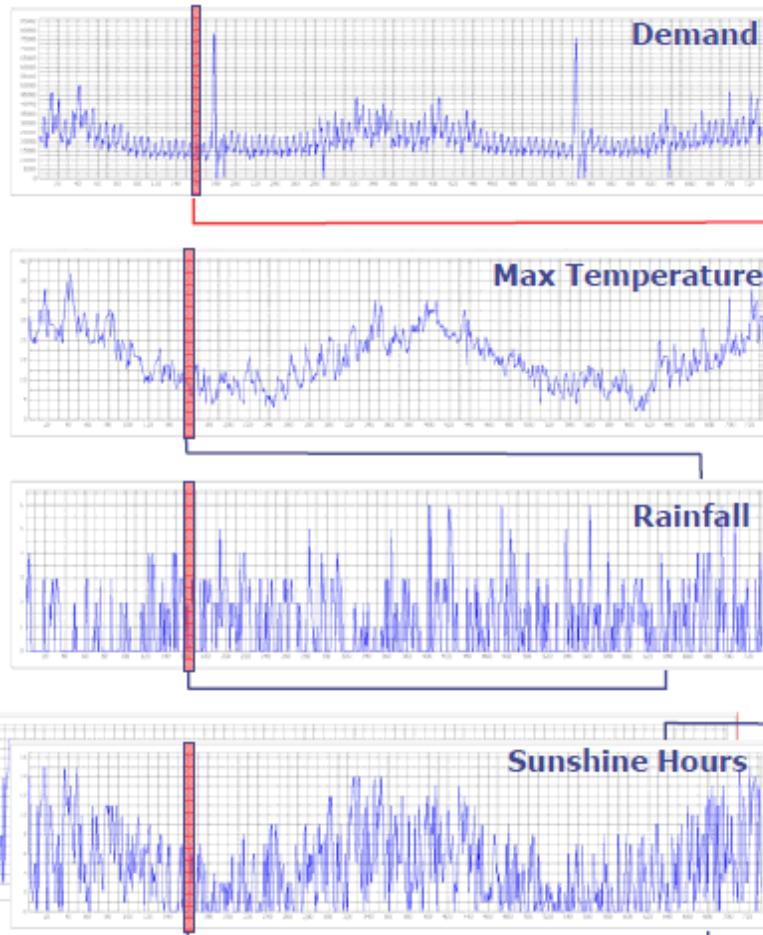


International airline passengers: monthly totals in thousands.

Daug duomenų galima rasti:
<https://datamarket.com/data/list/?q=price:free%20provider:tsdl%20type:dataset>



DNT kelių laiko eilučių atveju



Prognozavimo tikslumo matai

- Tegu y_i yra prognozuojama, o t_i – tikra reikšmė, m – duomenų kiekis.
- Vidutinė absoliuti paklaida (*mean absolute error*)

$$\text{MAE} = \frac{1}{m} \sum_{i=1}^m |t_i - y_i|,$$

- Vidutinė kvadratinė paklaida (*mean squared error*)

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^m (t_i - y_i)^2,$$

- Šaknis iš vidutinės kvadratinės paklaidos (root mean squared error)

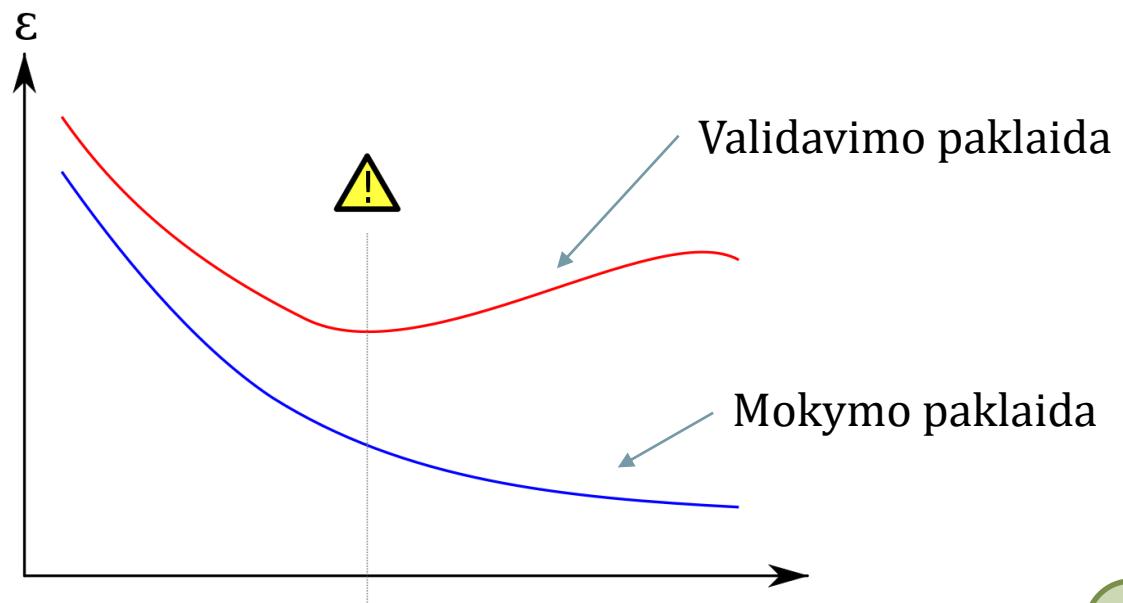
$$\text{RMSE} = \frac{1}{m} \sqrt{\sum_{i=1}^m (t_i - y_i)^2}.$$

Prognozavimo taikant DNT etapai

- **Mokymas** (*training*) – tai procesas, kurio metu į DNT pateikiame mokymo aibės duomenys, siekiant nustatyti tinkamas DNT **svorių reikšmės**.
- **Validavimas** (*validation*) – tai procesas, kurio metu į DNT pateikiame validavimo aibės duomenys siekiant nustatyti tinkamus **kitus** DNT **parametrus** (ne svorius). Šis procesas taip pat reikalingas siekiant „**nepermokinti**“ (*overfitting*) neuroninį tinklą.
- **Testavimas** (*testing*) – tai procesas, kurio metu į DNT pateikiame testavimo aibės duomenys, kurie nebuvome naudojami DNT mokyme, siekiant nustatyti **prognozavimo tikslumą**.

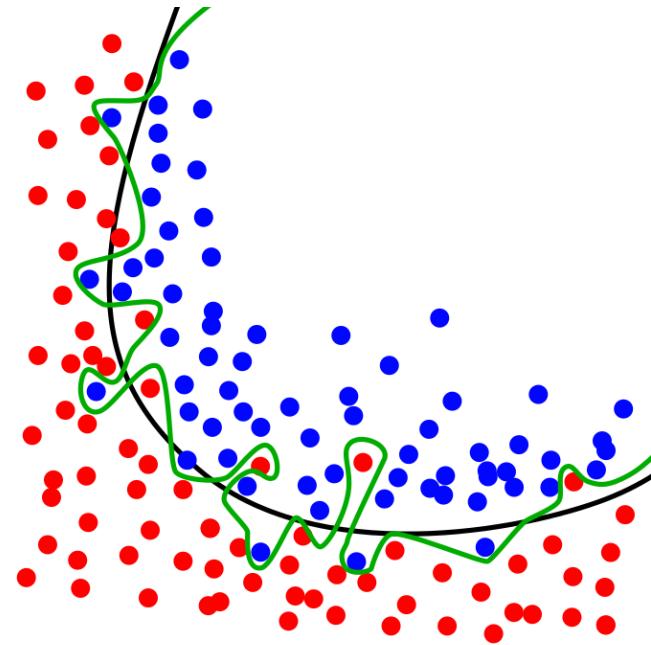
Neuroninio tinklo permokymas

- Mokant neuroninį tinklą, reikia stengtis jo „**nepermokti**“, kuomet tinklas labai prisiderina prie mokymo duomenų, tačiau jis nebus pajègus tiksliai prognozuoti (klasifikuoti) naujus duomenis.



Permokytas modelis

- Nagrinėjamas **dviejų** klasių atvejis.
- **Žalias** skiriama paviršius gautas „permokyto“ modelio, **juodas** – tinkamo modelio.
- Nors žalia kreivė **geriausiai** skiria mokymo aibės klasės, tačiau ji **nepajėgs tiksliai skirti** naujų duomenų klasės.

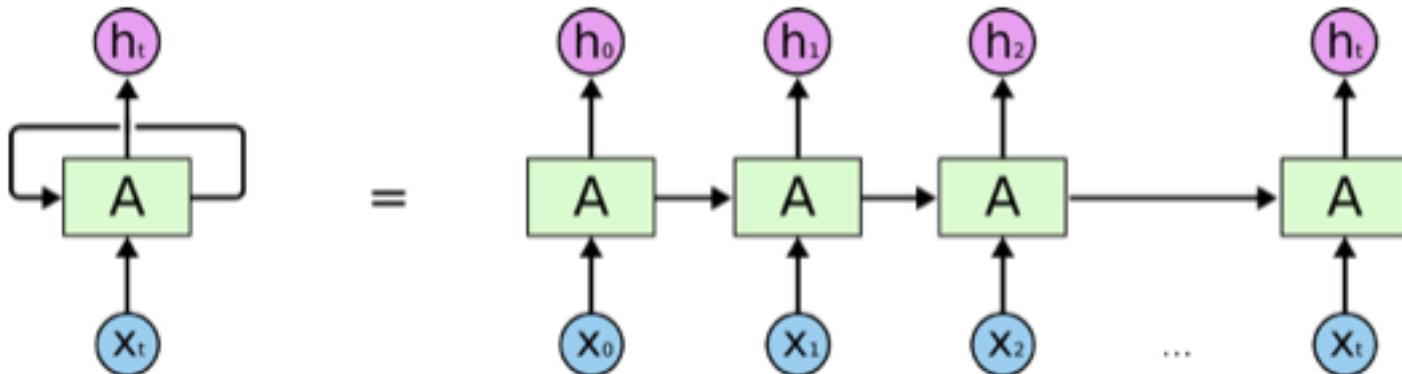
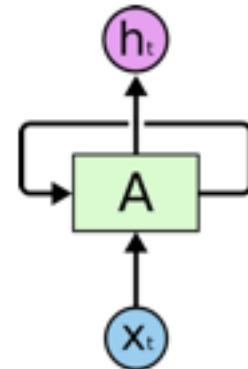


DNT, taikomi duomenims prognozuoti

- Daugiasluoksnis **perceptronas**
- **Rekurentiniai** neuroniniai tinklai:
 - Long short-term memory (LSTM)

Rekurentiniai neuroniniai tinklai

- **Rekurentiniai tinklai** turi ciklus, kurie leidžia informacijai būti perduotai iš vieno tinklo žingsnio į kitą.





Vilniaus universitetas
Matematikos ir informatikos fakultetas
Informatikos katedra



Dirbtiniai neuroniniai tinklai sistemoje WEKA

prof. dr. Olga Kurasova
Olga.Kurasova@mii.vu.lt

2018

Tiesioginio sklidimo DNT sistemoje WEKA

- Šiuo metu yra **sukurta programinių sistemų**, skirtų tik neuroniniams tinklams.
- Yra duomenų tyrybos sistemų, kuriose be kitų metodų yra **įgyvendinti ir neuroniniai tinklai**.
- **WEKA** – tai viena populiariausių duomenų tyrybos sistemų (www.cs.waikato.ac.nz/ml/weka/).
- Tai **atviro kodo nemokama sistema**, turinti plačias duomenų tyrybos funkcijas.

Sistema WEKA (1)

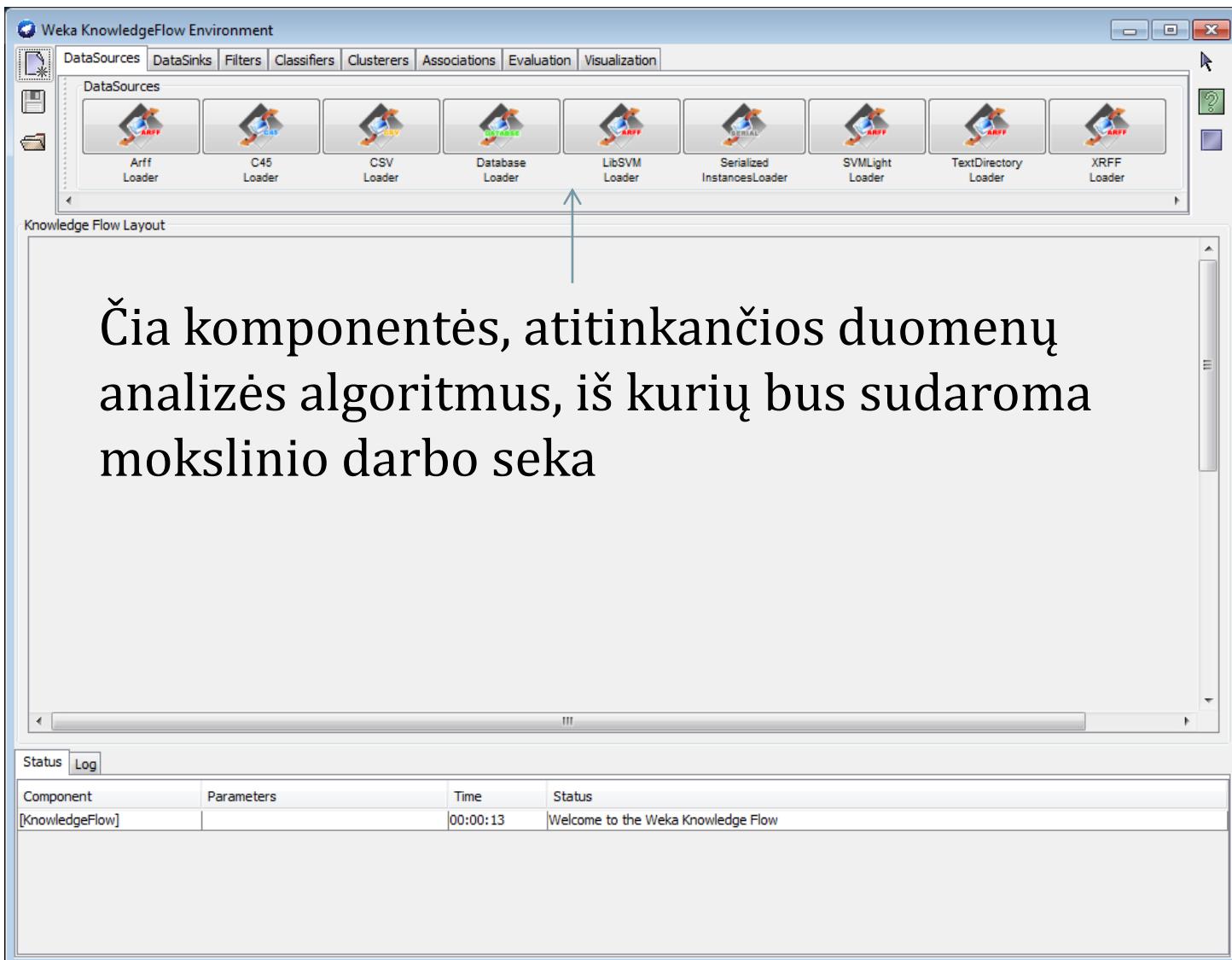
- Sistemoje duomenų tyrybos algoritmus galima taikyti **standartiniu būdu** arba sukūrus vadinamąsias **mokslinio darbo sekas**.
- Mokslinio darbo seka – tai sujungtų **komponenčių junginys**, kur kiekviena komponentė atitinka tam tikrą duomenų įkėlimo, apdorojimo ar tyrybos algoritmą, taip pat rezultatų peržiūrą.
- Tai patogus įrankis, kai norima tokiu pat būdu analizuoti **kitus duomenis**, ar tirti juos, **pakeitus** algoritmu **valdymo parametru reikšmes**.

Sistema WEKA (2)

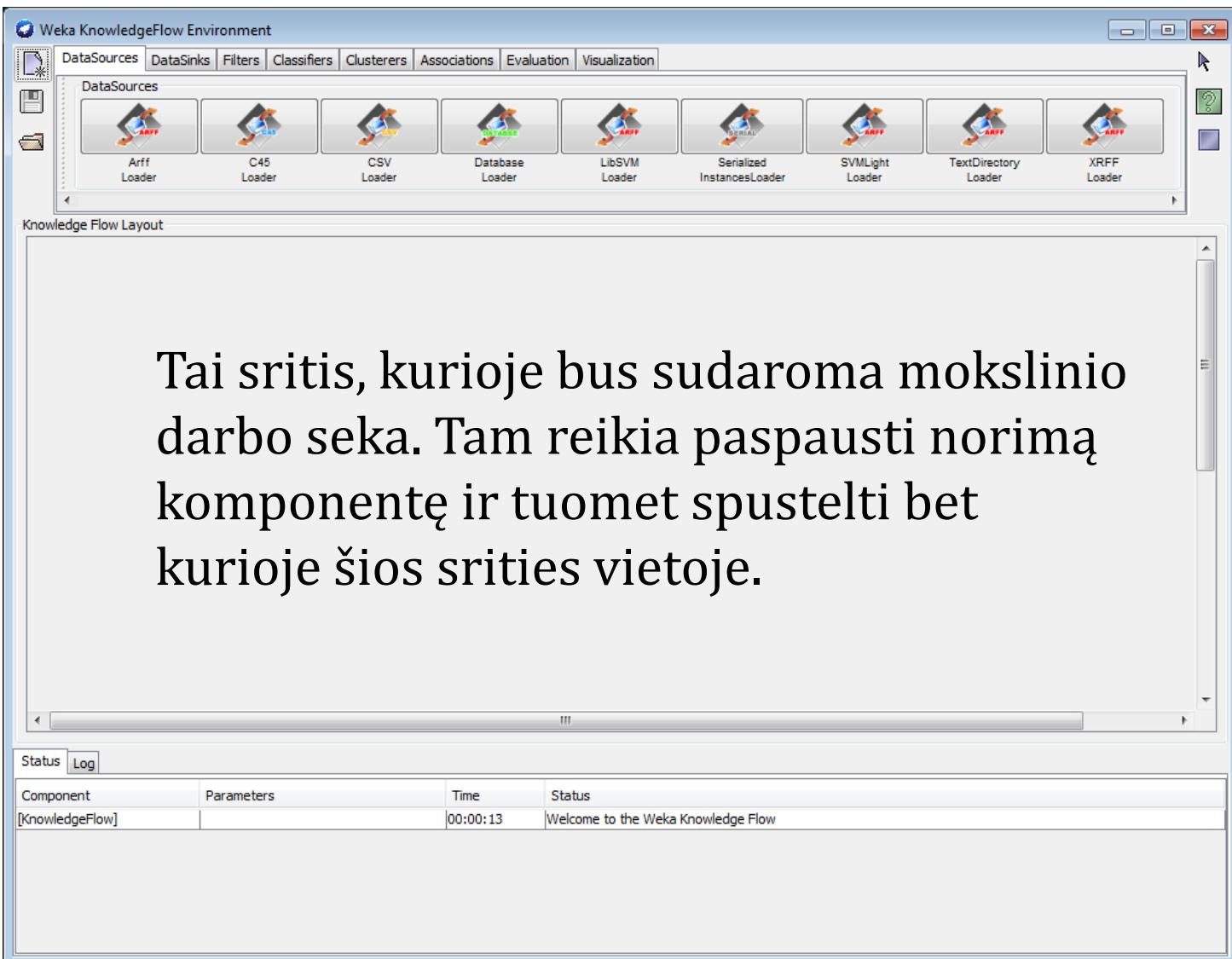
Norint pradėti **kurti mokslinio darbo seką**, reikia paspausti mygtuką ***KnowledgeFlow***.



Sistema WEKA (3)



Sistema WEKA (4)



Duomenų failų formatai sistemoje WEKA

- Sistemoje WEKA galima **įkelti kelių formatų** duomenis, tačiau rekomenduojama naudoti pačios sistemos **arff formatą**.
- Išiegiant sistemą, į kompiuterį **irašomi kelių duomenų failų**, pavyzdžiai. Pvz., C:\Program Files\Weka-3-6\data
- Failus arff formatu galima atverti ir redaguoti **paprastu tekstiniu redaktoriumi**.

Duomenys arff formatu

```
@RELATION iris

@ATTRIBUTE sepallength REAL
@ATTRIBUTE sepalwidth REAL
@ATTRIBUTE petallength REAL
@ATTRIBUTE petalwidth REAL
@ATTRIBUTE class {Iris-setosa,Iris-versicolor,Iris-virginica}

@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
```

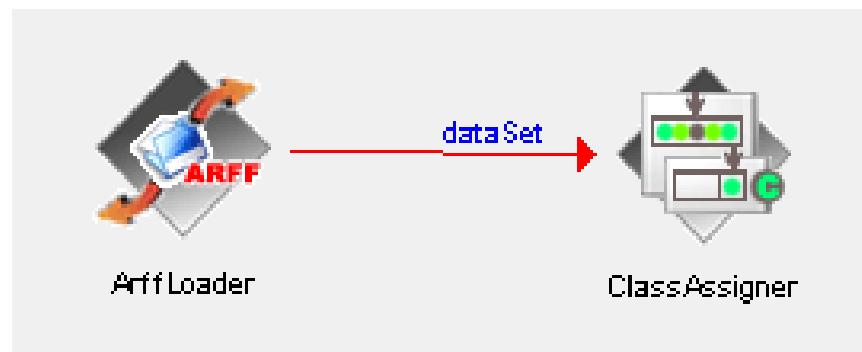
Duomenų įkėlimas sistemoje WEKA

- Norint įkelti duomenis arff formatu, reikia iš kortelės **DataSources** rinktis komponentę **Arff Loader** (nepamiršti spustelti sekos formavimo srityje).
- Du kartus ją spustelėti ir atsiradusiame l **nurodyti norimą failą**.
- Be to, jei bus sprendžiamas klasifikavimo uždavinys, reikia nurodyti, kuris duomenų požymis yra klasė. Tam skirta **Evaluation** kortelėje esanti komponentė **Class Assigner**. Ją du kartus paspaudus, reikia nurodyti tą požymį.



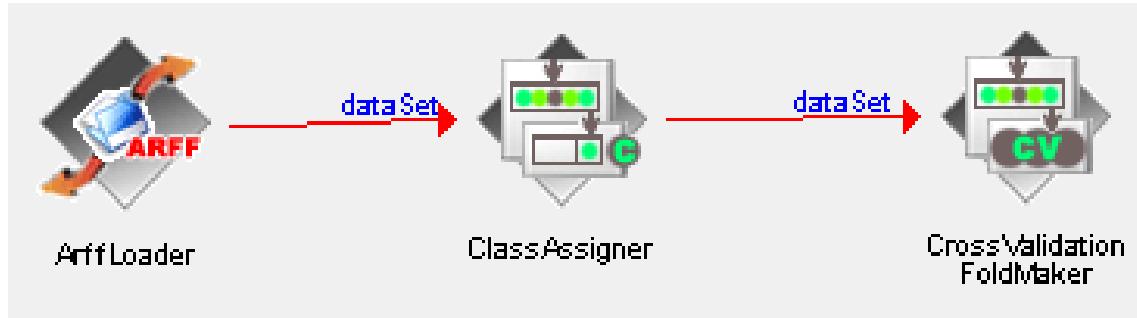
Duomenų įkėlimas sistemoje WEKA

- Dabar belieka **sujungti** šias dvi komponentes.
- Tai atliekama **tokiu būdu:**
 - ant pirmosios komponentės paspaudžiamas **dešinysis pelės mygtukas,**
 - išsirenkama **dataSet**
 - ir **paspaudžiama** ant kitos komponentės.



Kryžminė patikra

- WEKA sistemoje kryžminei patikrai atlikti skirta kortelėje **Evaluation** esanti komponentė **CrossValidation FoldMaker**.
- Du kartus ją spustelėjus, atsiradusiamе lange galima nurodyti norimą **blokų skaičių (folds)**, pagal kurį **bus suskaidoma** duomenų aibė į **mokymo** ir **testavimo** duomenis.



Daugiasluoksnio perceptrono komponentė

- **Daugiasluoksnio perceptrono** komponentės, skirta klasifikavimo uždaviniams spręsti, yra kortelėje **Classifiers** ir vadinasi **Multilayer Perceptron**.
- Ši komponentė turi būti sujungta su komponente **CrossValidation FoldMaker**.
- Iškėlus šią komponentę, ją du kartus spustelėjus pelyte, atsiranda **parametru nustatymo langas**.

MLP pagrindiniai parametrai

- **GUI: True** – bus rodoma tinklo struktūra.
- **hiddenLayers** – kableliais atskirti neuronų skaičiai paslėptuose sluoksniuose.
- **learningRate** – mokymo greičio parametras η .
- **Momentum** – tinklo mokymo taisyklėje naudojama reikšmė.
- **Seed** – atsitiktinių skaičių generavimo „sėkla“.

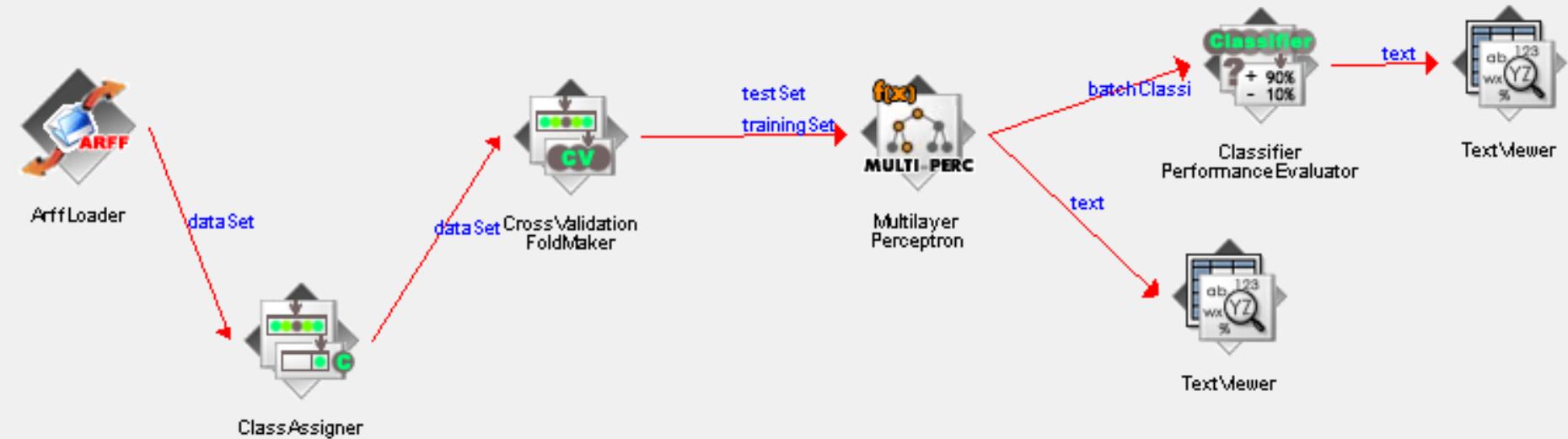
Klasifikavimo tikslumo nustatymas

- Norint pamatyti klasifikavimo rezultatų tikslumą, reikia prie komponentės **Multilayer Perceptron** prijungti kortelėje **Evaluation** esančią komponentę **Classifier PerformanceEvaluator**.
- O prie šios komponentės prijungti kortelėje **Visualization** esančią komponentę **TextViewer**.
- Tuomet reikia ant komponentės paspausti dešinį pelės mygtuką ir pasirinkti **Show Results**.

Perceptrono svoriai

- Norint pamatyti, kokie gauti išmokyto perceptrono svoriai, reikia prie komponentės **Multilayer Perceptron** prijungti komponentę **TextViewer**.
- Tuomet reikia ant komponentės paspausti dešinį pelės mygtuką ir pasirinkti **Show Results**.

Gauta mokslinio darbo seką



Norint **ivykdyti** sukurtą seką reikia ant **ArffLoader** komponentės paspaudus dešinįjį pelės mygtuką, pasirinkti **Start Loading**.

Klasifikavimo rezultatai

==== Evaluation result ===

Scheme: MultilayerPerceptron

Options: -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H "2, 3" -G -R

Relation: iris

Correctly Classified Instances	148	98.6667 %
Incorrectly Classified Instances	2	1.3333 %
Kappa statistic	0.98	
Mean absolute error	0.0273	
Root mean squared error	0.0894	
Relative absolute error	6.1343 %	
Root relative squared error	18.9712 %	
Total Number of Instances	150	

==== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	0	1	1	1	1	Iris-setosa
	0.98	0.01	0.98	0.98	0.98	0.999	Iris-versicolor
	0.98	0.01	0.98	0.98	0.98	0.999	Iris-virginica
Weighted Avg.	0.987	0.007	0.987	0.987	0.987	0.999	

==== Confusion Matrix ===

a	b	c	<- classified as
50	0	0	a = Iris-setosa
0	49	1	b = Iris-versicolor
0	1	49	c = Iris-virginica

Gauti perceptrono svoriai

```
==== Classifier model ====  
  
Scheme: MultilayerPerceptron  
  
Relation: iris  
  
Sigmoid Node 0  
    Inputs      Weights  
    Threshold   4.86297276124753  
    Node 5     -3.8660908414801503  
    Node 6     -4.705134898782587  
    Node 7     -9.11213313626472  
Sigmoid Node 1  
    Inputs      Weights  
    Threshold   -4.667974263875816  
    Node 5     -7.907836952620307  
    Node 6     -4.191198923452797  
    Node 7     9.397064982396241  
Sigmoid Node 2  
    Inputs      Weights  
    Threshold   -7.139047124741141  
    Node 5     6.407314719993383  
    Node 6     4.953708835603616  
    Node 7     2.148788678109934  
Sigmoid Node 3  
    Inputs      Weights  
    Threshold   5.97182308582669  
    Attrib sepallength  1.2928052580000988  
    Attrib sepalwidth   2.6852736928332734  
    Attrib petallength  -7.953115255023941  
    Attrib petalwidth   -9.316709649525436  
Sigmoid Node 4  
    Inputs      Weights  
    Threshold   0.7370366149205487
```

Sukurto klasifikatoriaus išsaugojimas

- Neuroninis tinklas mokomas tam, kad vėliau galėtų **nežinomų klasių duomenis priskirti** vienai iš klasių.
- Tam reikia **išsaugoti** sukurto klasifikatoriaus rezultatus, DNT atveju – **svorių reikšmes**.
- Paspaudus ant **Multilayer Perceptron** komponentės, reikia pasirinkti **Save Model** ir nurodyti saugojimo vietą bei failo vardą.

Duomenų išskaidymas į mokymo ir testavimo

- Mokant neuroninį tinklą, galima naudoti ne tik kryžminės patikros būdą, bet patiemis pradžioje **išskaidyti** į mokymo ir testavimo aibes.
- Tam sistemoje WEKA yra įgyvendinti keli būdai:
 - Naudoti vieną duomenų failą, tačiau komponentės **TrainTest SplitMaker** pagalba duomenis išskaidyti į mokymo ir testavimo.
 - Naudoti du duomenų failus, vieną komponentės **Training SetMaker** pagalba priskirti mokymo duomenis, kitą komponentės **TestSet Maker** pagalba – testavimo.

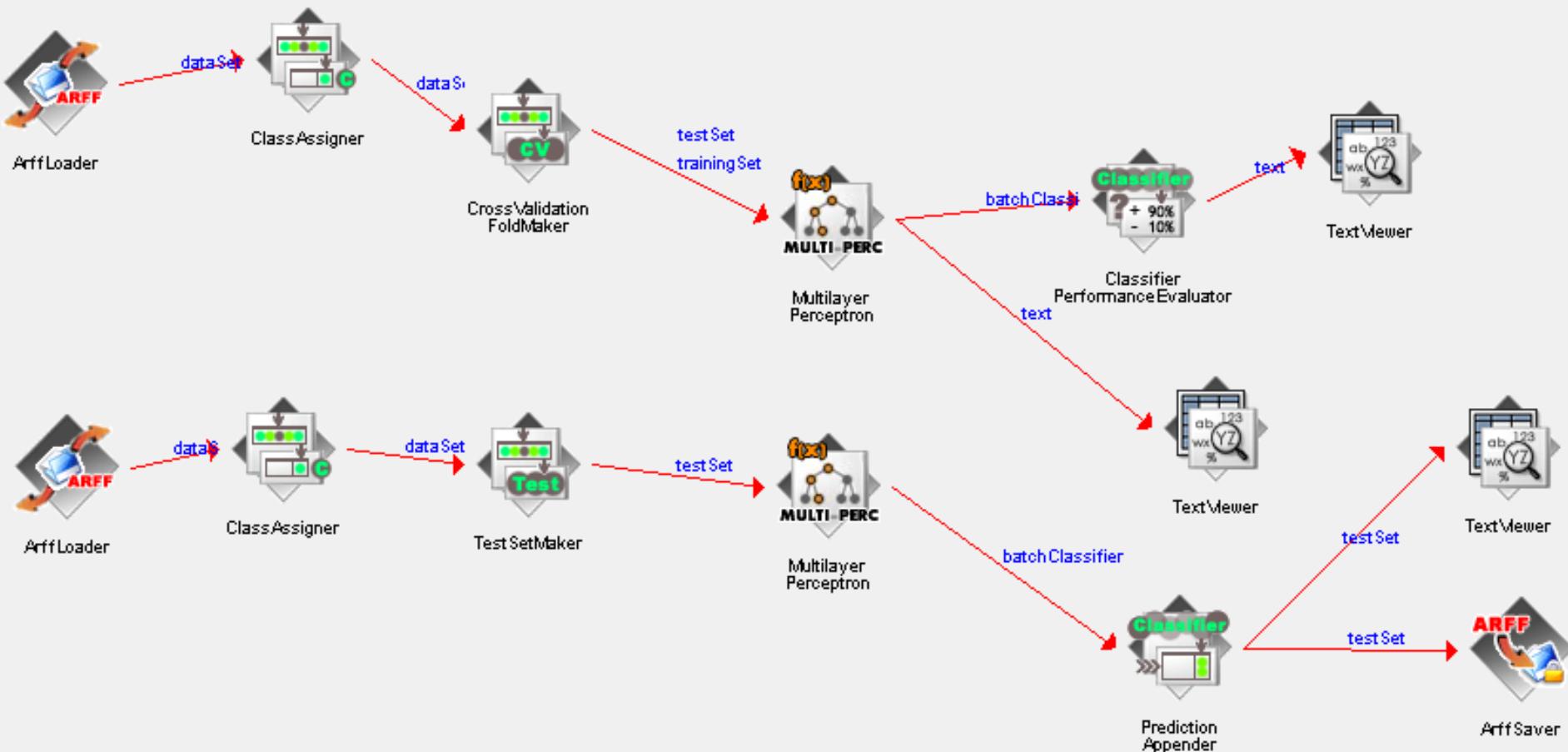
Naujų duomenų priskyrimas klasėms (1)

- Neuroninis tinklas mokomas tam, kad vėliau galėtų **nežinomų klasių duomenis priskirti** vienai iš klasių.
- Jei duomenų **klasė nėra žinoma**, tai arff faile prie vietoj klasių **įrašomas klaustukas**.
- Tuomet reikia įkelti dar vieną **ArffLoader** komponentę, nurodyti tą arff failą, kuriame yra **nauji duomenys** (su nežinomomis klasėmis).
- Taip pat įkelti ir sujungti komponentes **ClassAssigner**, **TestSet Maker**, **Multilayer Perceptron**.

Naujų duomenų priskyrimas klasėms (2)

- Paspaudus ant komponentės **Multilayer Perceptron** dešinį pelės mygtuką, pasirinkti **Load Model** ir nurodyti anksčiau išsaugotą klasifikatoriaus rezultatus (modelį).
- Tuomet prie šios komponentės reikia prijungti **Evaluation** kortelėje esančią komponentę **Prediction Appender**, o prie jos – **Text Viewer**.
- Paspaudus du kartus **Prediction Appender** galima nurodyti, ar bus rodomos klasių priskyrimo tikimybės (TRUE) ar ne (FALSE).
- Paspaudus dešinį palės mygtuką ant komponentės **Text Viewer** ir pasirinkus **Show results**, matomi naujų duomenų klasifikavimo rezultatai.
- Prijungus kortelėje **DataSinks** esančią komponentę **Arff Saver**, klasifikavimo rezultatai bus **išsaugoti** nurodytame faile.

Gautos mokslinio darbo sekos



Duomenų vizualizavimas

- Sistemoje WEKA yra **duomenų vizualizavimo komponentės**, leidžiančios pamatyti analizuojamų duomenų struktūrą.
- Kortelėje **Visualization** pasirinkus komponentę **Scatter PlotMatrix** ir ją prijungus prie komponentės **Class Assigner**, galima pamatyti duomenų išsidėstymą Dekarto koordinačių sistemoje (imant požymių poras).



Vilniaus universitetas
Matematikos ir informatikos fakultetas
Informatikos katedra



Gilusis mokymasis

prof. dr. Olga Kurasova
Olga.Kurasova@mii.vu.lt

2018

Šių dienų raktažodžiai

- **Didieji duomenys** (angl. *big data*)
- **Dirbtinis intelektas** (angl. *artificial intelligence*)
- **Mašininis mokymasis** (angl. *machine learning*)
- **Gilusis mokymasis** (angl. *deep learning*)
- **Gilioji neuroniniai tinklai** (angl. *deep neural networks*)

Mašininis mokymasis

- **Mašininis mokymasis** (angl. *mashine learning*) – tai informatikos sritis, suteikianti kompiuteriams galimybę mokytis be aiškių instrukcijų.
- **Machine learning** is a field of computer science that gives computers the ability to learn without being explicitly programmed (*Wikipedia*).
- **Machine learning** is the use of artificial intelligence to automate algorithms that can learn without explicit instructions (*Intro to Machine Learning Course / Udacity*)

Mašininis mokymasis

Sinonimai:

- Automatinis mokymasis
- Kompiuterių mokymasis
- Sistemų mokymasis

Mašininis mokymasis

Mašininis mokymasis reikalingas sprendžiant **šiuos uždavinius**:

- Klasifikavimas
- Prognozavimas (regresinė analizė)
- Klasterizavimas
- Tankio įvertis (angl. *density estimation*)
- Duomenų dimensijos mažinimas
- Kiti.

Mašininio mokymosi taikymai

- Automated theorem proving^{[39][40]}
- Adaptive websites^[citation needed]
- Affective computing
- Bioinformatics
- Brain–machine interfaces
- Cheminformatics
- Classifying DNA sequences
- Computational anatomy
- Computer vision, including object recognition
- Detecting credit-card fraud
- General game playing^[41]
- Information retrieval
- Internet fraud detection^[28]
- Linguistics
- Marketing
- Machine learning control
- Machine perception
- Medical diagnosis
- Economics
- Insurance
- Natural language processing
- Natural language understanding^[42]
- Optimization and metaheuristic
- Online advertising
- Recommender systems
- Robot locomotion
- Search engines
- Sentiment analysis (or opinion mining)
- Sequence mining
- Software engineering
- Speech and handwriting recognition
- Financial market analysis
- Structural health monitoring
- Syntactic pattern recognition
- Time series forecasting
- User behavior analytics
- Translation^[43]

Brain Differences

Now wait before you start thinking that you can just create a huge neural network and call strong AI, there are some few points to remember:

Just a list:

- The artificial neuron fires totally different than the brain
- A human brain has 100 billion neurons and 100 trillion connections (synapses) and operates on 20 watts(enough to run a dim light bulb) - in comparison the biggest neural network have 10 million neurons and 1 billion connections on 16,000 CPUs (about 3 million watts)
- The brain is limited to 5 types of input data from the 5 senses.
- Children do not learn what a cow is by reviewing 100,000 pictures labelled “cow” and “not cow”, but this is how machine learning works.
- Probably we don't learn by calculating the partial derivative of each neuron related to our initial concept. (By the way we don't know how we learn)

Gilusis mokymasis

- **Gilusis mokymasis** (angl. *deep learning*) – tai plačios mašininio mokymosi metodų šeimos dalis pagrįsta duomenų pateikimo (reprezentatyvumo) mokymusi priešingai nei yra algoritmuose konkretiems uždaviniams spręsti.
- **Deep learning** (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on **learning data representations**, as opposed to task-specific algorithms.

Dimensiškumo prakeiksmas

- Mašininio mokymosi uždaviniuose labai dažna vadinamojo **dimensiškumo prakeiksmo** (angl. *curse of dimensionality*) problema.
- Jei nagrinėjamus duomenis apibūdina daug požymių, tokiu duomenų **dimensija bus didelė**.
- Tuomet objektų (duomenų įrašų) turi **būti labai daug**.

Dimensiškumo prakeiksmas

- Tarkime turime vienmatį atvejį: vienetinio ilgio atkarpoje atidėtus **10 taškus**.
- Norint, kad toks pats tankis išliktų dvimatėje erdvėje, reikia **$10^2=100$ taškų**, trimatėje – **$10^3=1000$ taškų**, n-matėje – **10^n taškų**. Priešingu atveju, duomenis yra **labai reti** (angl. sparse).
- Pavyzdžiui, jei duomenis apibūdina 20 požymių, norint kad algoritmas gerai apsimokyti, reikia jį mokytį **10^{20}** įrašais. Tieki įrašų turėti arba visai **neįmanoma**, arba mokymosi procesas truks **labai ilgą laiką**.
- Tikslinga pasitelkti **dimensijos mažinimo metodus**.

Gilusis mokymasis

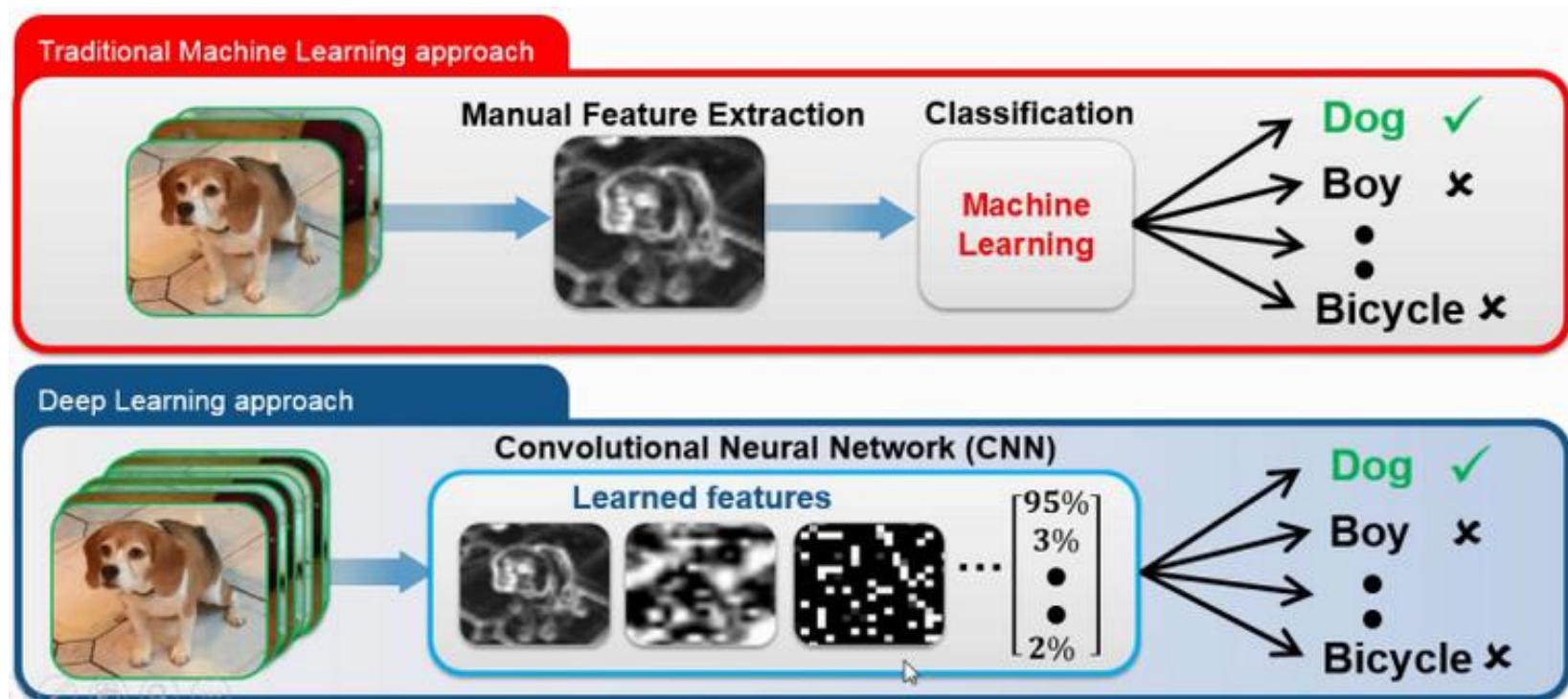
Deep learning is a class of machine learning algorithms that:

- use a **cascade of multiple layers** of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input.
- learn in **supervised** (e.g., classification) and/or **unsupervised** (e.g., pattern analysis) manners.
- learn **multiple levels of representations** that correspond to different levels of abstraction; the levels form a hierarchy of concepts.
- use some form of **gradient descent** for training via backpropagation.

Duomenų reprezentavimas (pateikimas)

- **Duomenų reprezentavimas (pateikimas)** yra labai svarbus aspektas.
- Būtina atsakyti į klausimą, **kokie požymiai geriausiai reprezentuoja** analizuojamus objektus.
- Tai ypač aktualu tokiose srityse, kaip **šnekos, vaizdo** ir **garso** atpažinimas ir pan., kur būtina išgauti tinkamiausius požymius iš turimo signalo ar vaizdo.

Mašininis vs gilusis mokymasis



https://leonardoaraujosantos.gitbooks.io/artificial-intelligence/content/deep_learning.html

Duomenų reprezentavimo (pateikimo) pavyzdys

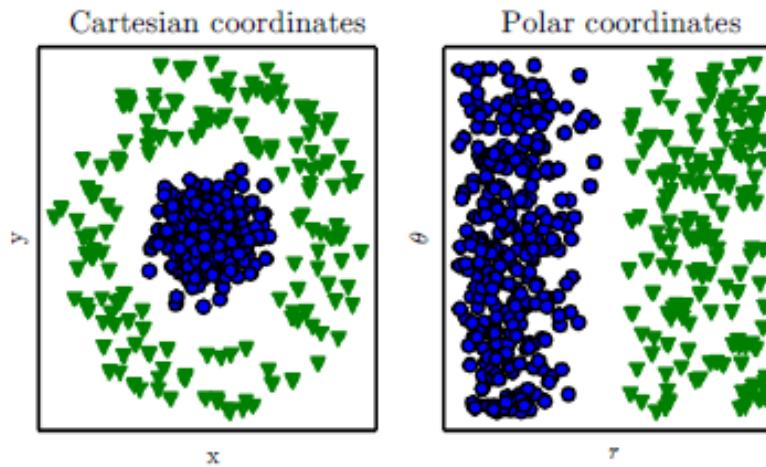
- Pavyzdžiui, įprastai, žmogui atlikti skaičių, pateiktų **arabiškais skaitmenimis**, aritmetinius veiksmus yra nepalyginamai **paprasčiau** nei tą patį atlikti su skaičiais, pateiktais **romėniškais skaitmenimis**.

$$\begin{array}{rcl} \text{III} + \text{VII} & = & \\ \\ \text{II} + \text{V} & = & \end{array}$$

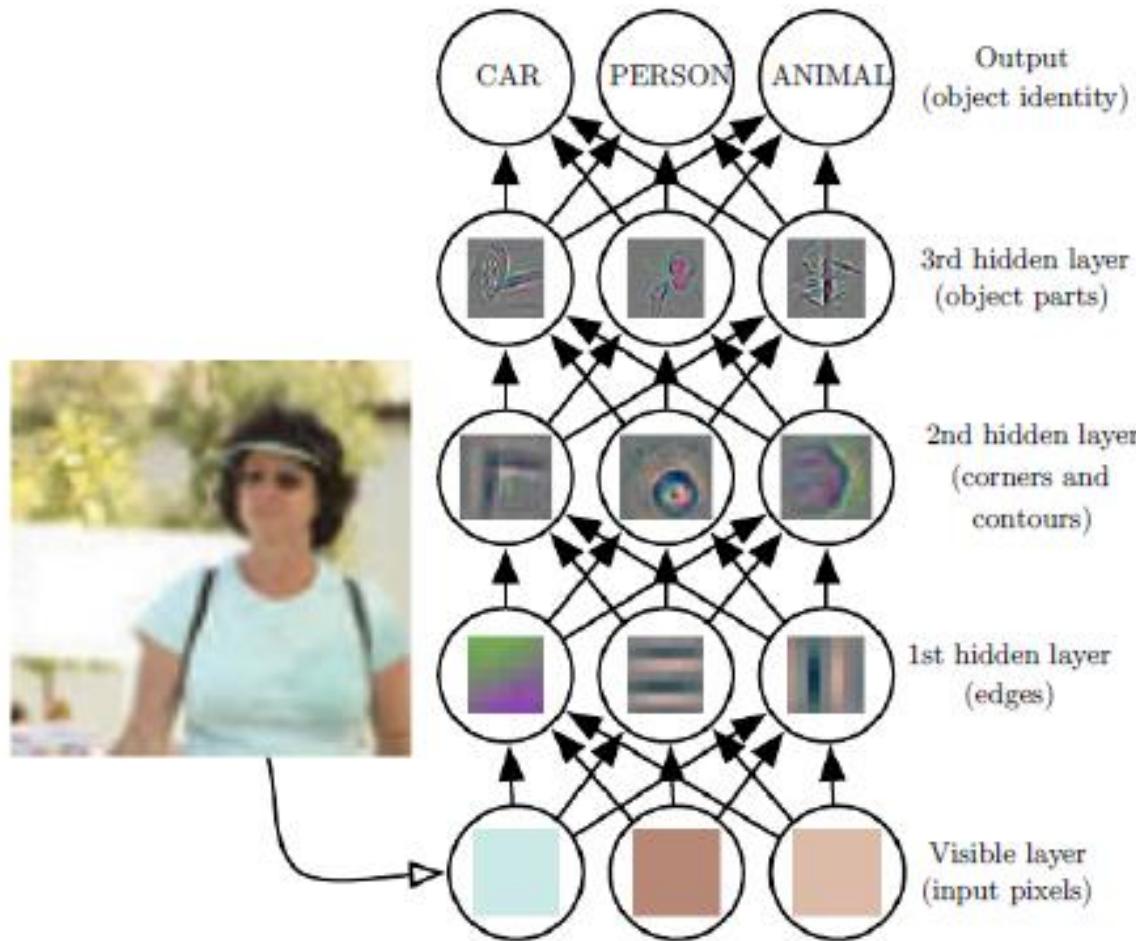
$$\begin{array}{rcl} 23 + 32 & = & \\ 42 + 21 & = & \\ 55 + 12 & = & \end{array}$$

Duomenų reprezentavimo (pateikimo) pavyzdys

- Pavyzdžiui, pateikus tuos pačius duomenis Dekarto ar polinėje koordinačių sistemoje, iš esmės pasikeičia **klasių tiesinio atskyrimo** problema.



Gilusis mokymasis



Duomenų reprezentavimas giliajame mokymesi

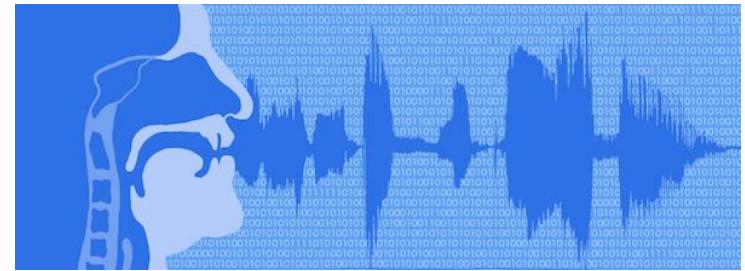
- Duomenų **reprezentavimas** giliajame mokymesi yra ypač svarbus, kadangi čia siekiama, kad **automatiškai** būtų parinkta **geriausia** duomenų reprezentacija.

Gilusis mokymasis

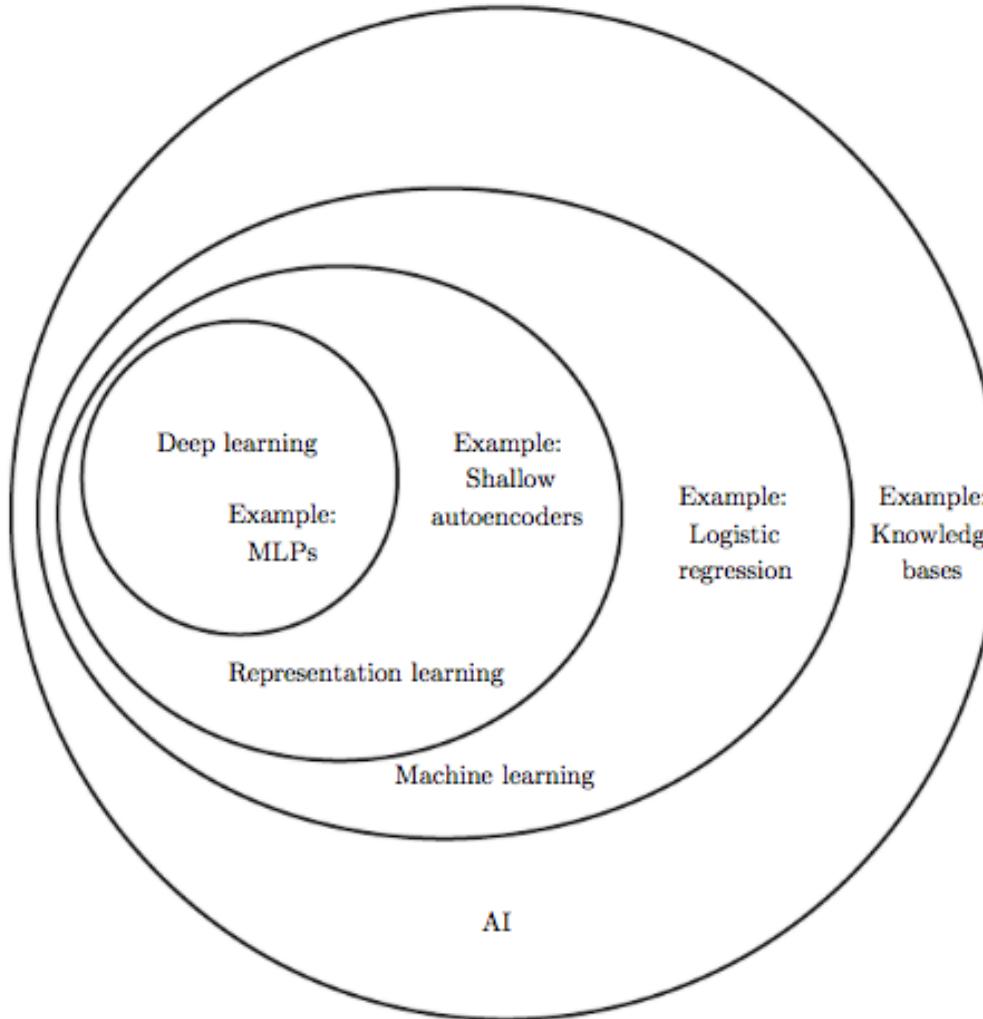
- Deep learning is a particular kind of machine learning that achieves great power and flexibility by representing the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones (Goodfellow et al., 2016)

Gilusis mokymasis

- Dėl savo savybių gilusis mokymasis dažniausiai taikomas
 - **Vaizdų** apdorojime ir analizēje
 - **Garsų** apdorojime ir analizēje (šnekos atpažinimas)
 - **Signalų** apdorojime ir analizēje (kalbos analizē)



Gilusis mokymasis ir dirbtinis intelektas



Gilusis mokymasis: istoriniai aspektai

- Giliojo mokymosi paradigma **nėra naujas dalykas**, bet įvairiai laikotarpiais turėjo įvairius vardus.
- Gilusis mokymasis tapo naudingas kai **išaugo mokymo duomenų kiekis**.
- Giliojo mokymosi modeliai tapo sudėtingesnais, kai **išsiplėtė skaičiavimų infrastruktūra** (tiek programinė, tiek techninė įranga). Tas leidžia spresti sudėtingus taikomuosius uždavinius per sąlyginai trumpą laiką.

Gilusis mokymasis: istoriniai aspektai

ARTIFICIAL INTELLIGENCE

Early artificial intelligence stirs excitement.



MACHINE LEARNING

Machine learning begins to flourish.



DEEP LEARNING

Deep learning breakthroughs drive AI boom.



Gilusis mokymasis: kodėl katinas?

- Šiai laikais **pirmasis pavyzdinis uždavinys** naudojant giliojo mokymo paradigmą buvo:
 - Iš YouTube vaizdų bibliotekos išrinkti **vaizdus su katinais.**



Gilusis mokymasis: šiandien

● deep learning
Paieškos terminas

+ Palyginti

Pasaulyje ▾

Pastarieji 5 metai ▾

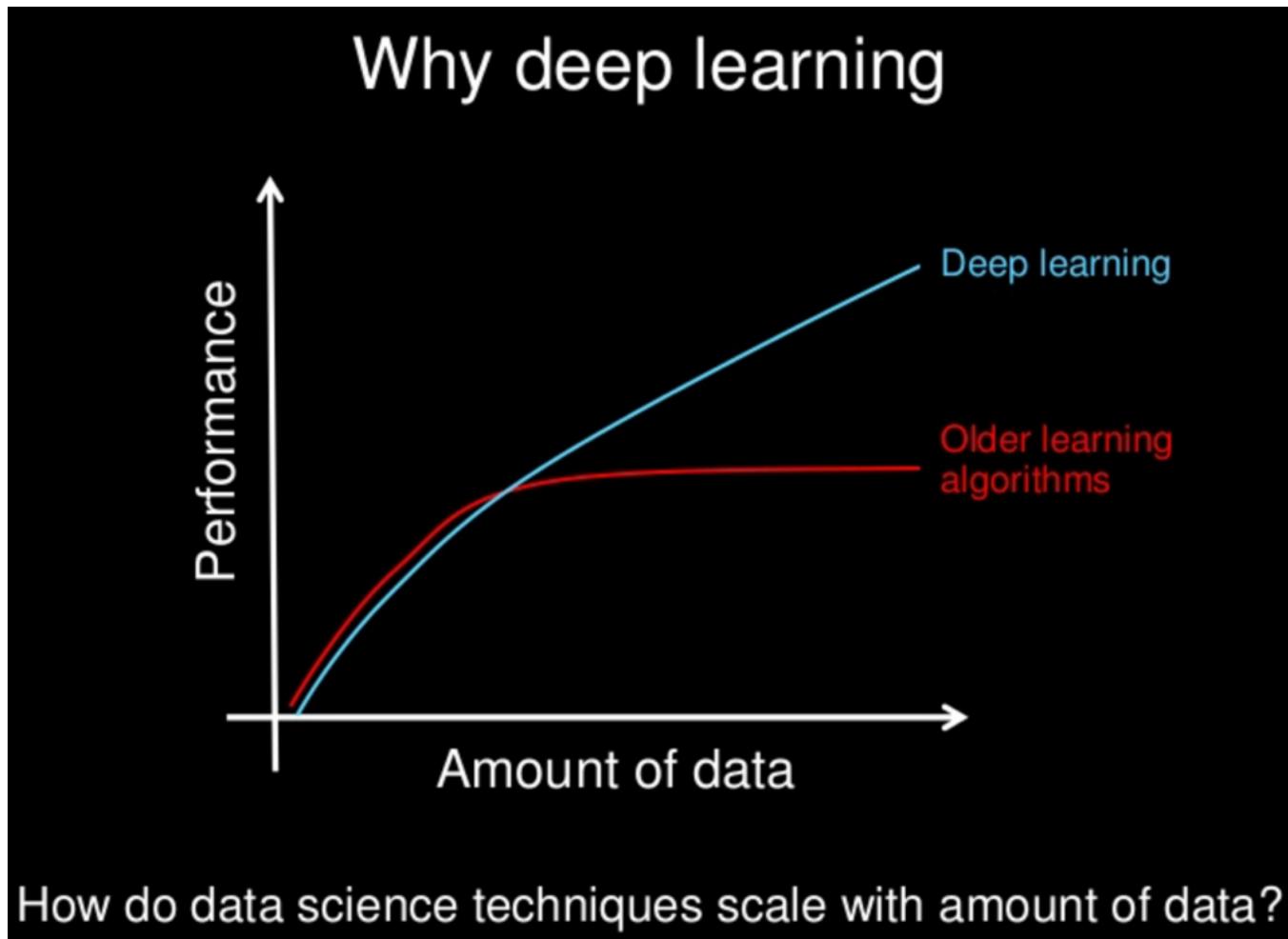
Visos kategorijos ▾

Žiniatinklio paieška ▾

Susidomėjimas per laikotarpį ?



Giliojo mokymosi pajęgumai

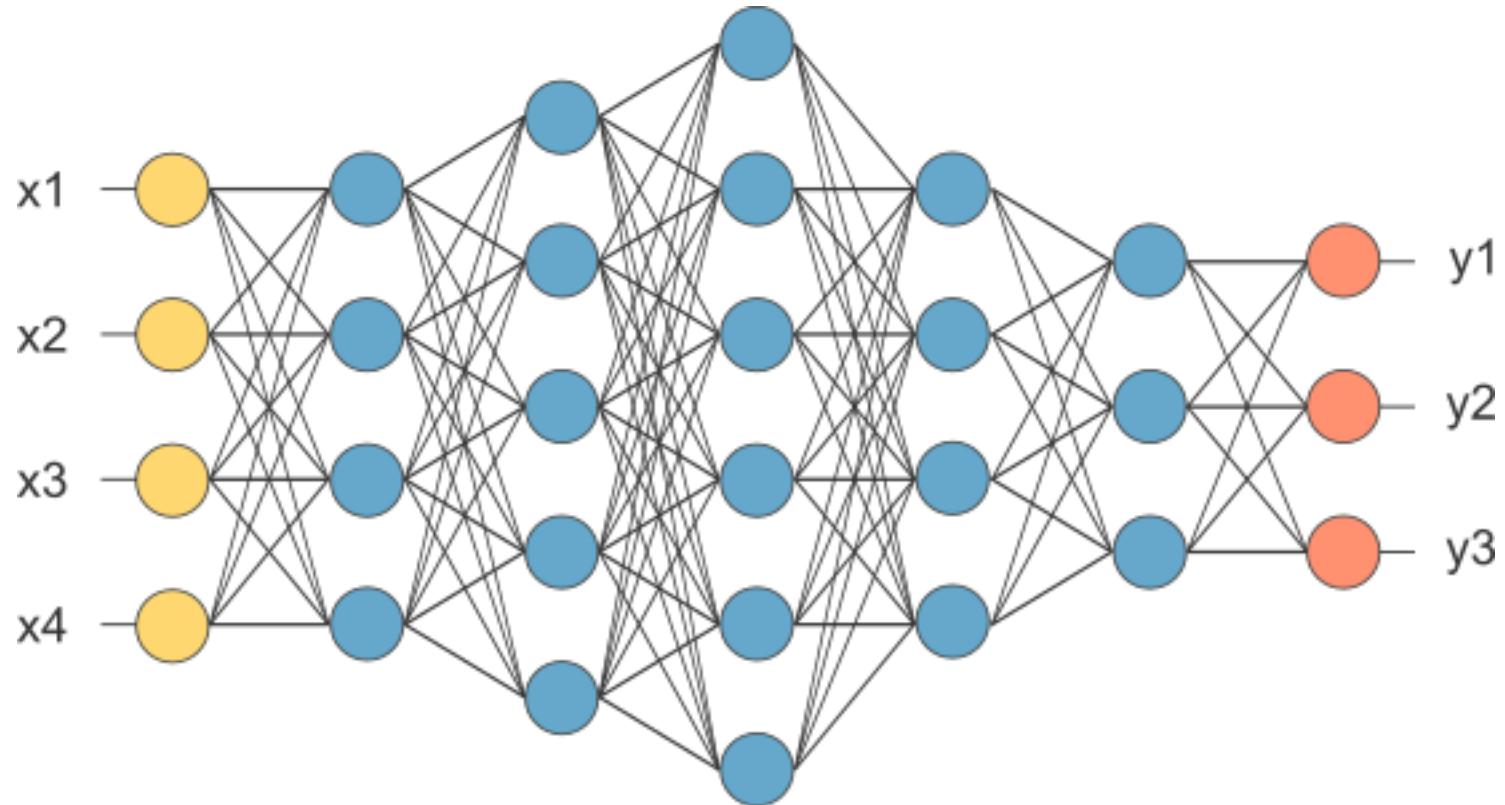


<https://machinelearningmastery.com/what-is-deep-learning/>

Gilioji neuroniniai tinklai

- Nors **tiesioginio sklidimo neuroniniai tinklai** gali turėti norimą kiekį paslėptų sluoksnių ir neuronų skaičių juose, tačiau dėl buvusio skaičiavimų resursų ribotumo, dažnai buvo naudojami **tik vienas ar du paslėpti sluoksniai**, turintis po kelis neuronus.
- Šiuo metu plečiantis skaičiavimo resursų pajėgumams, atsirado galimybė naudoti **daugiau sluoksnių** ir **daugiau neuronų** juose.
- Taip išpopuliarejo **gilioji neuroniniai tinklai**.

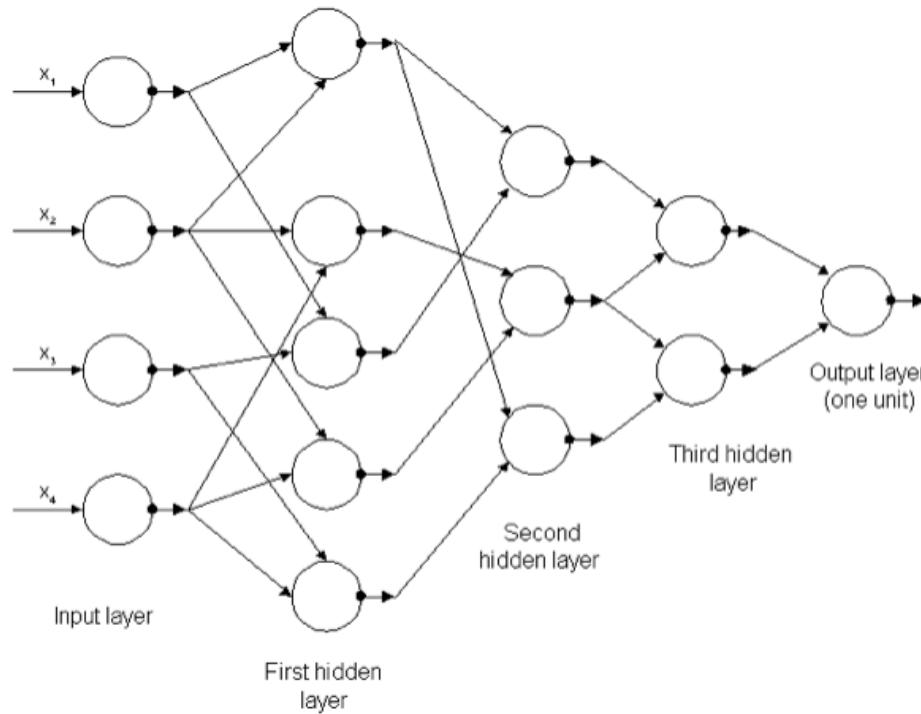
Giliųjų NN struktūra



Paveikslas iš <http://www.opennn.net>

Giliųjų DNT pradžia

- **Giliųjų DNT pradžia** laikoma **1965** m.
- Ivakhnenko, A. G. and Lapa, V. G. (1965). Cybernetic Predicting Devices. CCM Information Corporation.





Vilniaus universitetas
Matematikos ir informatikos fakultetas
Informatikos katedra



Konvoliuciniai neuroniniai tinklai

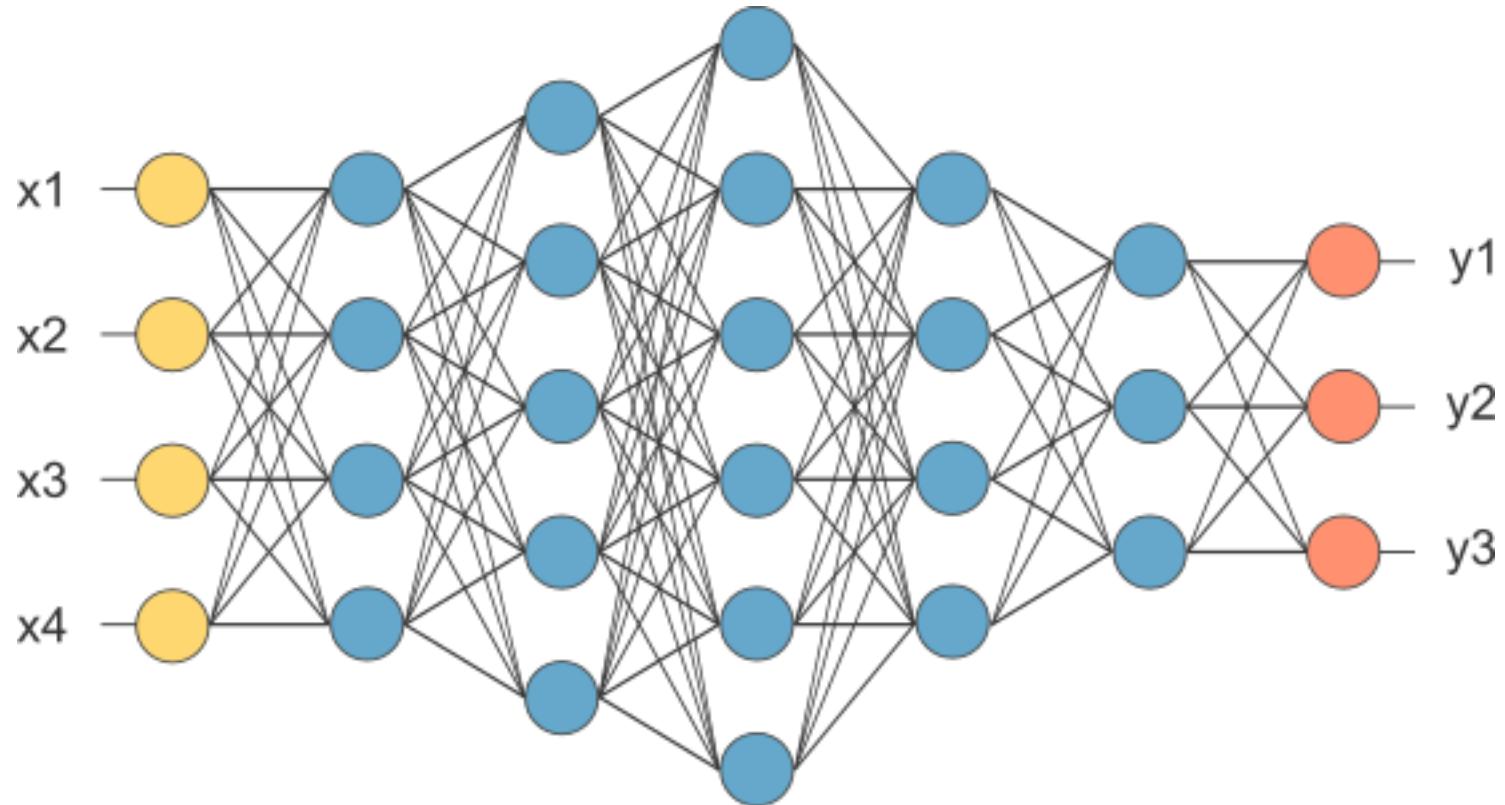
prof. dr. Olga Kurasova
Olga.Kurasova@mii.vu.lt

2018

Gilioji neuroniniai tinklai

- Nors **tiesioginio sklidimo neuroniniai tinklai** gali turėti norimą kiekį paslėptų sluoksnių ir neuronų skaičių juose, tačiau dėl buvusio skaičiavimų resursų ribotumo, dažnai buvo naudojami **tik vienas ar du paslėpti sluoksniai**, turintis po kelis neuronus.
- Šiuo metu plečiantis skaičiavimo resursų pajėgumams, atsirado galimybė naudoti **daugiau sluoksnių** ir **daugiau neuronų** juose.
- Taip išpopuliarejo **gilioji neuroniniai tinklai**.

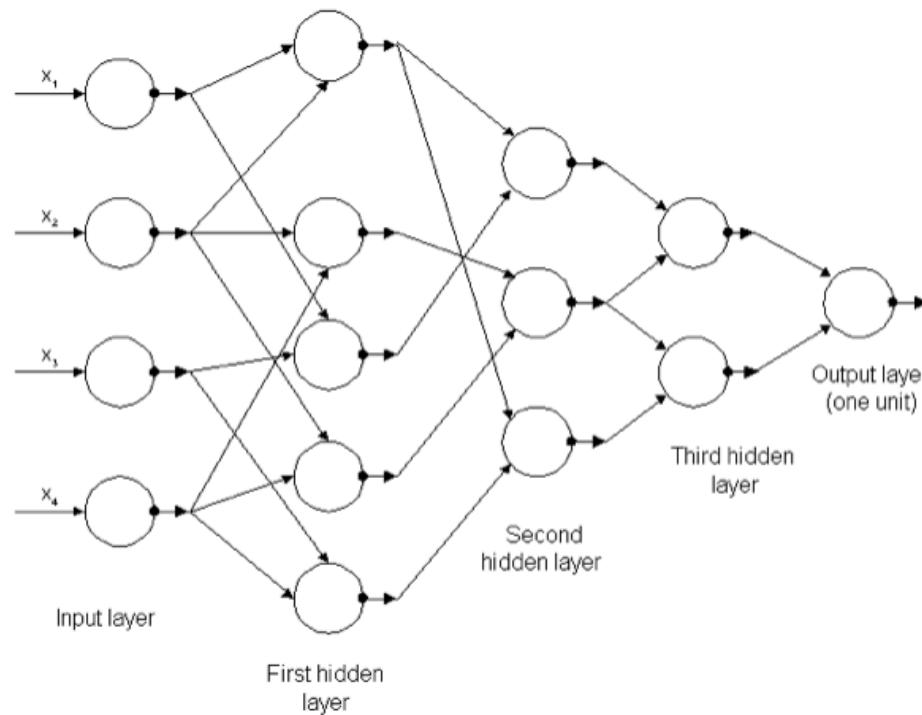
Giliųjų NN struktūra



Paveikslas iš <http://www.opennn.net>

Giliųjų DNT pradžia

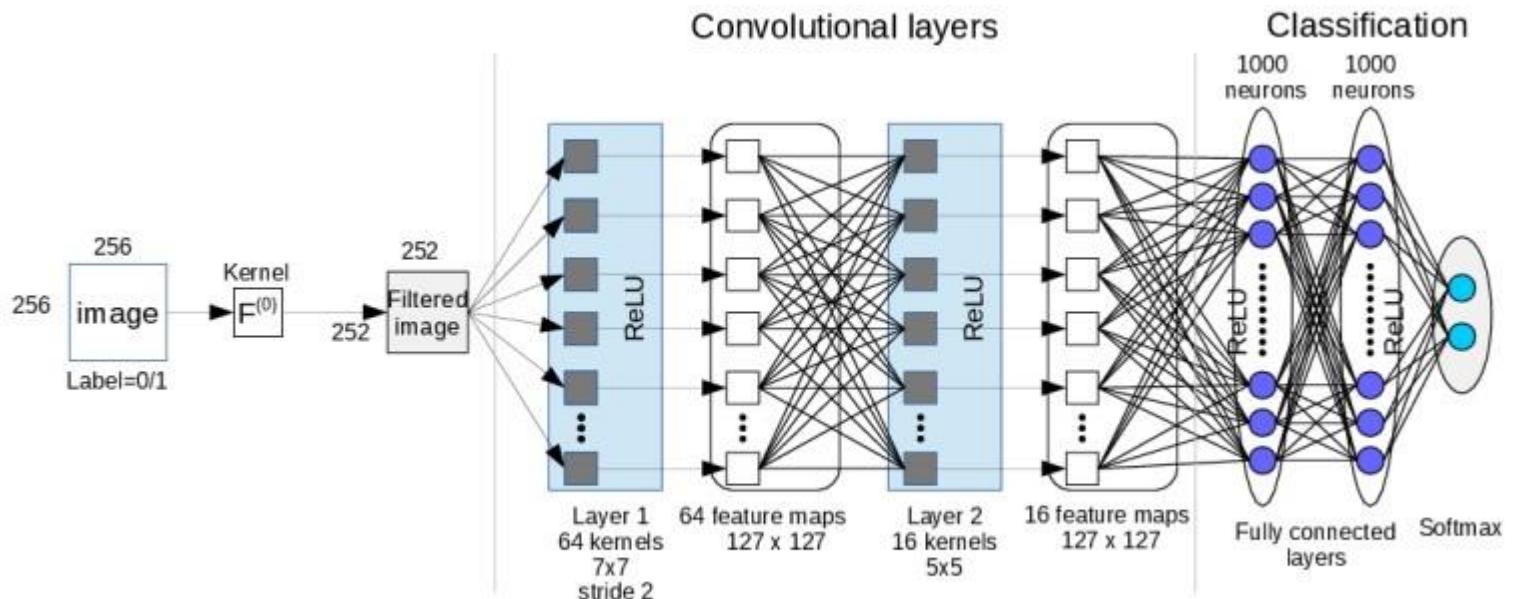
- **Giliųjų DNT pradžia** laikoma **1965** m.
- Ivakhnenko, A. G. and Lapa, V. G. (1965). Cybernetic Predicting Devices. CCM Information Corporation.



Konvoliuciniai neuroniniai tinklai

- **Konvoliuciniai neuroniniai tinklai** (angl. *convolution neural networks, CNN, ConvNet*) – tai vieni populiariausių giliųjų neuroninių tinklų tipų.
- Dar vadinami **sąsukiniais** neuroniniais tinklais.
- Pradininkas **Yann LeCun** (dirbantis Facebook dirbtinio intelekto tyrimų grupėje) (https://en.wikipedia.org/wiki/Yann_LeCun), pirmasis panaudojo CNN **ranka rašytiems skaitmenims atpažinti** (MNIST duomenų aibė).

Konvoliuciniai neuroniniai tinklai



<https://medium.com/@eternalzer0dayx/demystifying-convolutional-neural-networks-ca17bdc75559>

Konvoliucija (sąsuka)

- CNN pavadinimas kilo nuo **matematinės operacijos** – konvoliucija (sąsuka).
- **Konvoliucija** – tai matematinė operacija, kuri paima dvi funkcijas f ir g ir grąžina trečią, kuri, tam tikra prasme, parodo f ir g persidengimo lygi.
- Dažniausiai viena funkcija imama kaip fiksuotas filtras, dar vadinamas **branduoliu** (angl. *kernel*).
- Funkcijų f ir g konvoliucija žymima $f * g$. Ji apibrėžiama kaip **funkcijų sandaugos integralas**, po to, kai viena jų buvo paversta ir padauginta iš -1 .

Konvoliucija (sąsuka)

- Konvoliucija yra **integralinės transformacijos rūšis**:

$$(f * g)(t) = \int_a^b f(\pi)g(t - \pi)d\pi$$

- Integravimo **rėžiai** priklauso nuo funkcijų apibrėžimo srities. Dažniausiai $a = -\infty$ ir $b = +\infty$.

Konvoluciijos savybės

- Komutatyvumas: $f * g = g * f$
- Asociatyvumas: $f * (g * h) = (f * g) * h$
- Distributyvumas: $f * (g + h) = (f * g) + (f * h)$
- Vienetinis elementas: $f * \delta = \delta * f = f$
- Daugybos su skaliaru asociatyvumas:

$$a(f * g) = (af) * g = f * (ag)$$

kiekvienam realiam skaičiui a .

Diskreti konvoliucija

- Iprastai mašininio (giliojo) mokymo atvejais, nagrinėjant duomenis, besikeičiančius laike, susiduriama su **diskrečiais dalykais**.
- Diskrečioms funkcijoms apibrėžiama **diskrečios konvoliucijos** operaciją:

$$s(t) = (f * g)(t) = \sum_{n=-\infty}^{\infty} f(n)g(t - n)$$

- Naudojant šią formulę **konvoliucijos sudėtingumas** yra lygus $O(N^2)$ aritmetinių operacijų N taškams. Tačiau šis dydis gali būti sumažintas iki $O(N \log N)$, panaudojant greitesnius algoritmus.

Konvoliucija vaizdams

- Turint dvimatį vaizdą I kaip įvestį, naudojant **dvimatį branduolių (filtrą) K** :

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n)$$

- **Konvoliucija** yra komutatyvi, tad:

$$S(i, j) = (K * I)(i, j) = \sum_m \sum_n I(i - m, j - n)K(m, n)$$

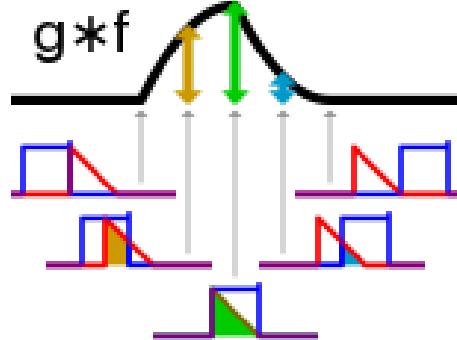
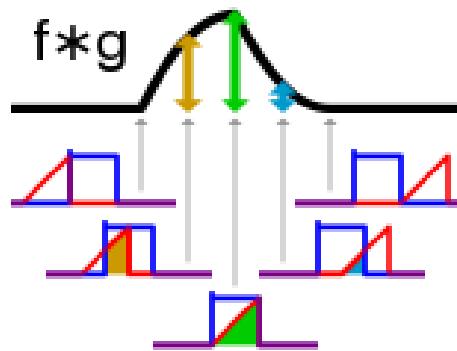
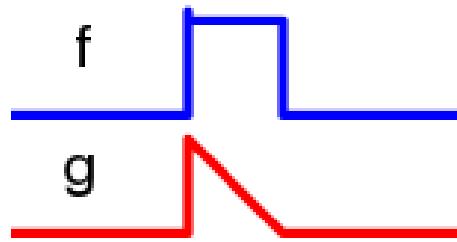
- Įprastai naudojama antroji formulė.

Konvoliucija vs kryžminė koreliacijos

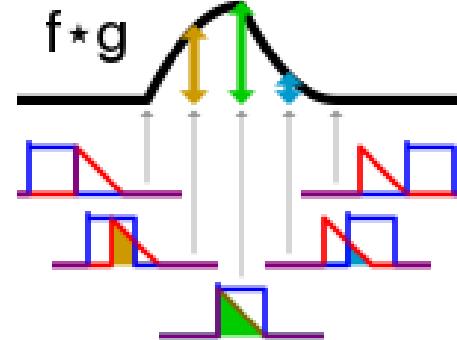
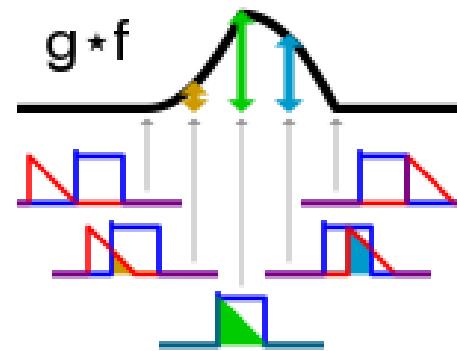
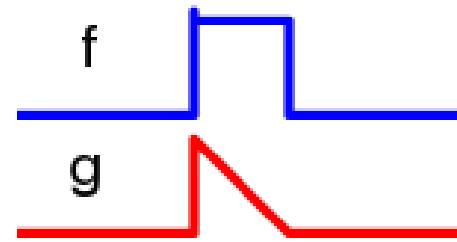
- Kadangi konvoliucija yra labai panaši į **kryžminę koreliaciją** (angl. *cross-correlation*), dažnai šie terminai vartojami kaip sinonimai.
- Kryžminės koreliacijos atveju nėra branduolio funkcijos apvertimo:

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n)$$

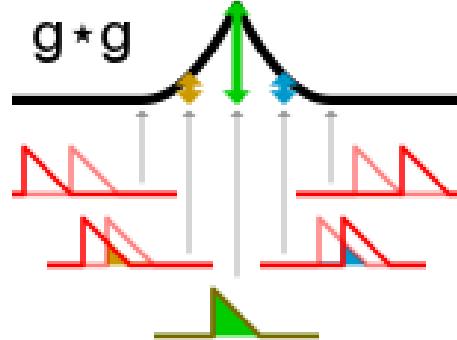
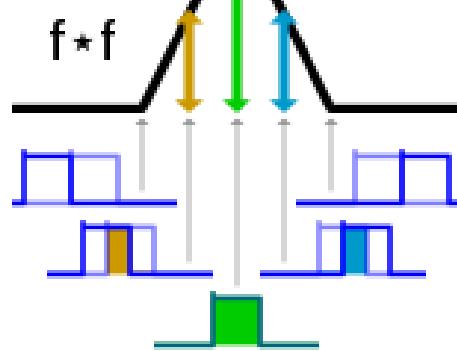
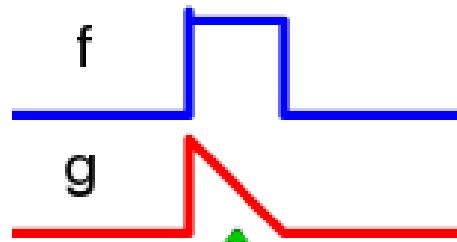
Convolution



Cross-correlation

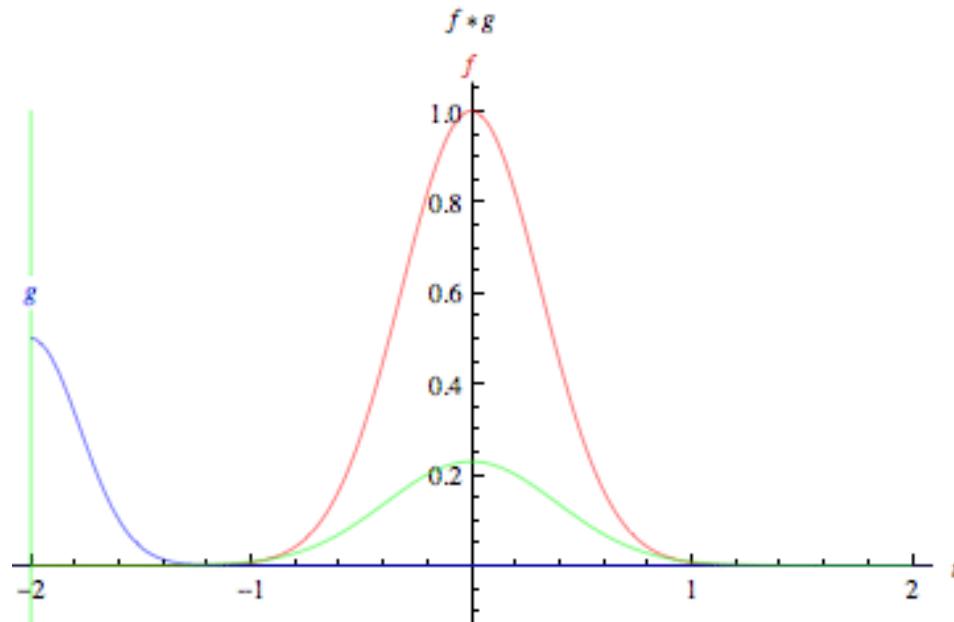


Autocorrelation



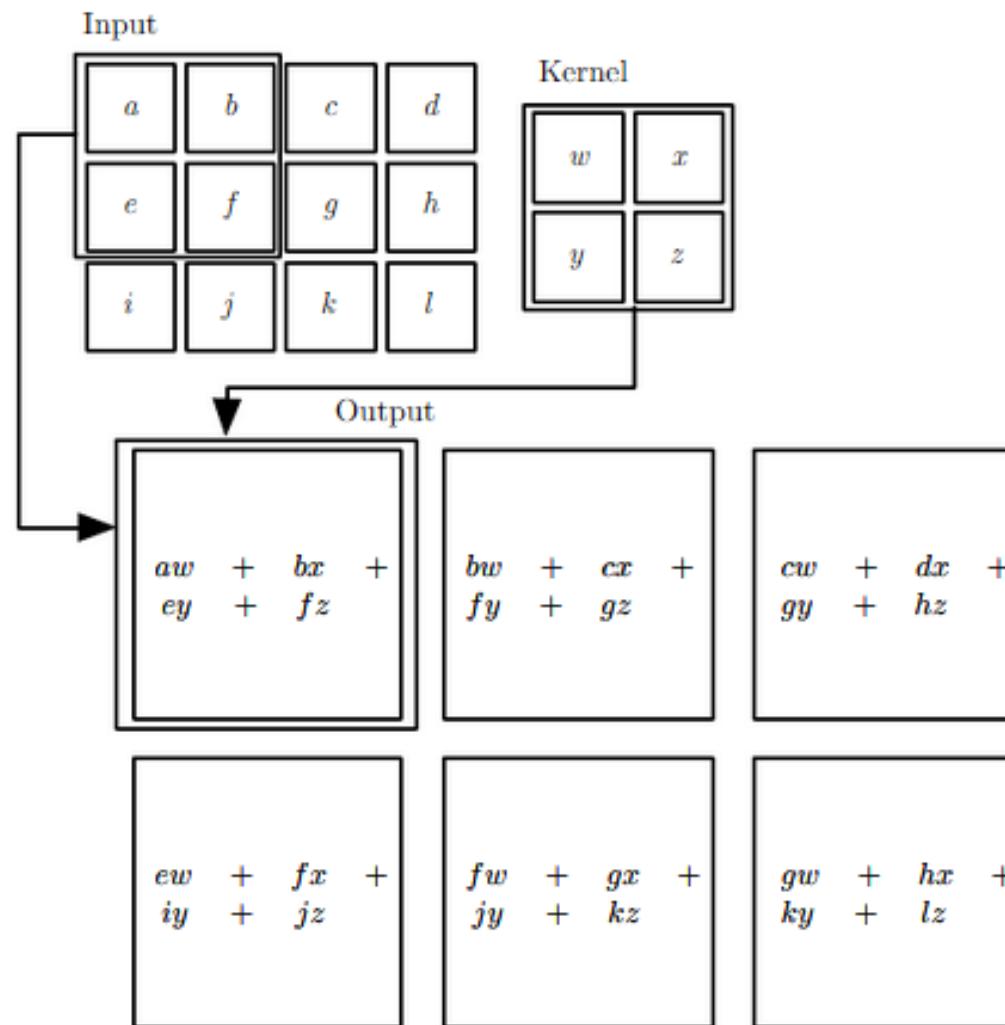
Konvoliucija

- Žalia kreivė parodo **mėlynos** ir **raudonos** kreivių konvoliuciją, t.y. šių funkcijų persidengimą.



Interaktyvų paveiksluką rasite:
<https://skymind.ai/wiki/convolutional-network>

Konvoliucijos pavyzdys (be branduolio apvertimo)



Konvoliucija matricos daugybos prasme

- Diskreti konvoliucija gali būti **išreiškiama daugyba iš matricos**, turinčios kelis elementus, kurie yra lygūs kitiems elementams.
- **Pavyzdžiu**, vienmatei diskrečiai konvoliucijai kiekviena matricos eilutė turi būti lygi ankstesnei eilutei, perustumtai per vieną elementą.

$$\begin{pmatrix} a & b & c & d & e \\ f & a & b & c & d \\ g & f & a & b & c \\ h & g & f & a & b \\ i & h & g & f & a \end{pmatrix}$$

Konvoliucija – išretinta matrica

Atsižvelgiant į tai, kad įprastai branduolio matrica yra daug mažesnė nei vaizdo matrica, be to, keli matricos elementai turi būti lygūs vienas kitam, **konvoliucija atitinka labai išretintą matricą** (angl. *sparse matrix*) – matricą, kurios didžioji dalis elementų lygūs 0.

Konvoliuciniai neuroniniai tinklai

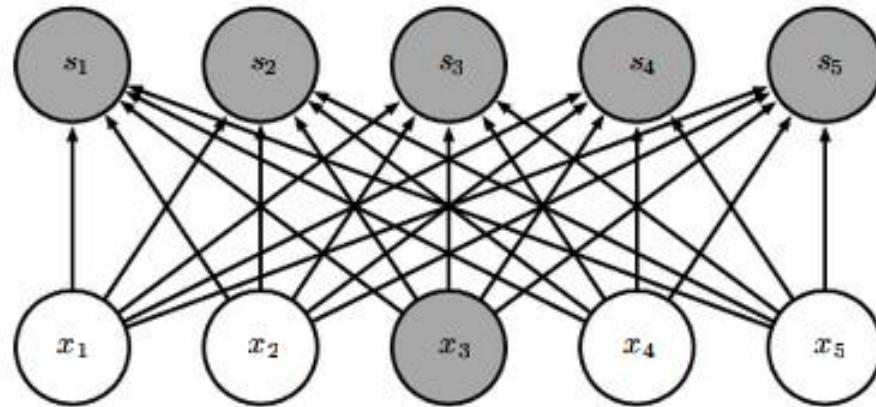
- **Tradiciniuose neuroniniuose tinkluose** naudojama matricos daugyba iš parametru matricos, kur parametrai nurodo sąveiką tarp įvesties ir išvesties. Tai reiškia, kad kiekviena išvestis sąveikauja su kiekvienu įėjimu.
- **Konvoliuciniai tinklai** yra paprasčiausiai neuroniniai tinklai, naudojantys konvoluciujos operacijas vietoj įprastos matricų daugybos mažiausiai viename neuronų sluoksnyje.

Konvoliuciniai neuroniniai tinklai

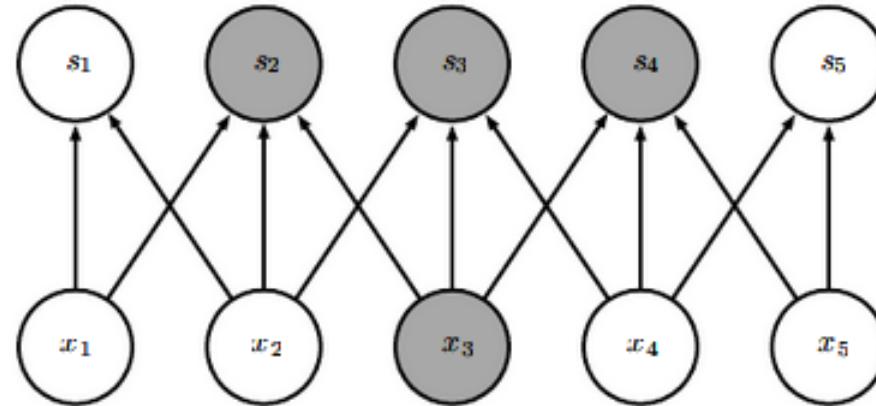
Šie konvoliucijos principai leidžia pagerinti mašininį mokymąsi:

- **išretinta sąveika,**
- **parametru pasidalinimas,**
- **ekvivalentiški atvaizdavimai** (reprezentacijos)

Konvoliuciniai sąryšiai tinkluose: išretinta sąveika



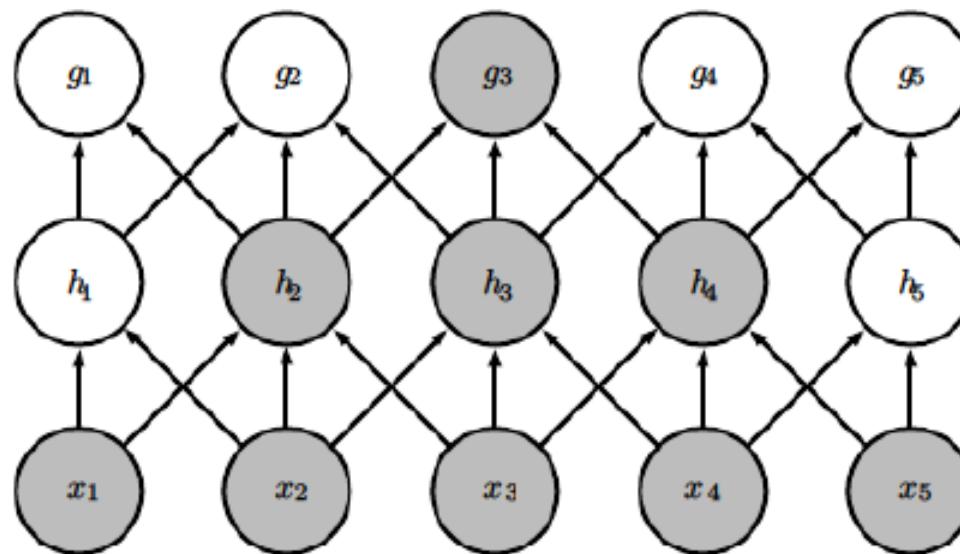
Tradiciniuose
tinkluose (**kiekvienas
su kiekvienu**)



Konvoliuciniuose
tinkluose (**kiekvienas
su tam tikrais**)

Konvoliuciniai sąryšiai tinkluose: išretinta sąveika

- Mazgai **gilesniuose sluoksniuose** gali netiesiogiai sąveikauti su didesne porcija įvesčių.
- Tai leidžia tinklui **efektyviau aprašyti sudėtingus sąryšius** tarp daug kintamųjų, sudarant tokias sąveikas iš paprastų blokų, kur kiekvienas aprašo tik išretintą sąveiką.



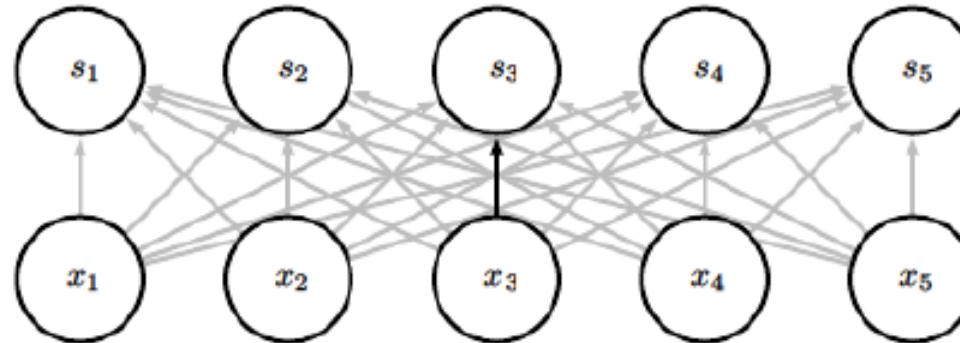
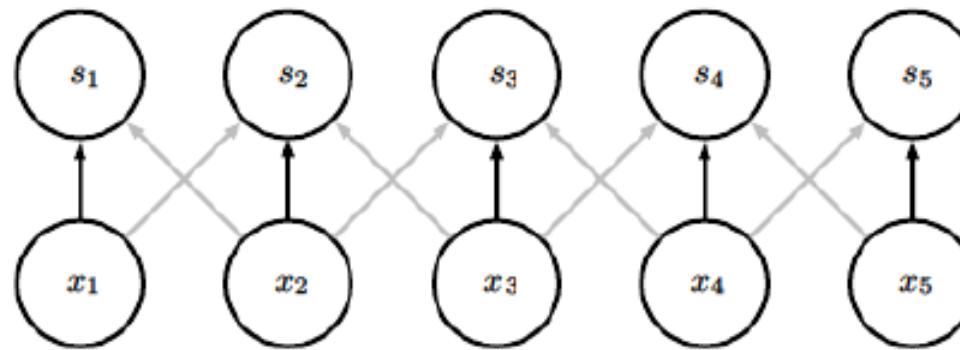
Vaizdų apdorojimas: išretinta sąveika

- Pavyzdžiui, įvesties vaizdas gali būti sudarytas iš tūkstančių ar net milijonų pikselių, tačiau galima nustatyti daug mažesnio skaičiaus, bet **reikšmingus požymius**, tokius kaip kraštai panaudojus branduoli, sudarytą tik iš dešimčių ar šimtų pikselių.
- Tai leidžia saugoti mažesnį kiekį informacijos, taip **sumažinamas būtinės atminties** kiekis, be to, padidinamas modelio statistinis patikimumas.

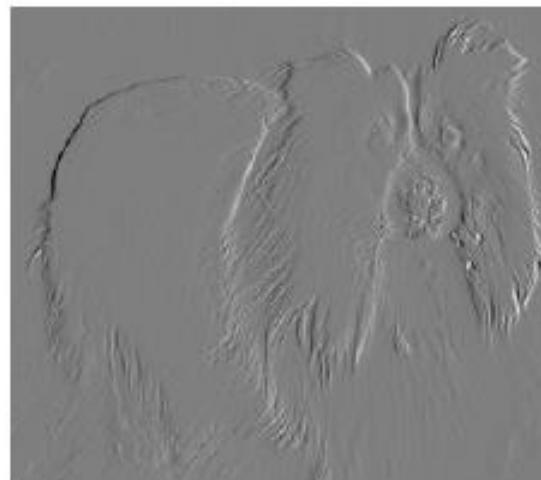
Parametru pasidalinimas

- Kai modelyje **tas pats parametras** naudojamas daugiau nei vienoje funkcijoje.
- **Tradiciniuose tinkluose** perskaičiuojant sluoksnio išėjimus, kiekvienas **svorių matricos** elementas yra **naudojamas tik vieną kartą**.
- **Konvoluciiniuose tinkluose** kiekvienas branduolio narys yra naudojamas kiekvienoje įvesties pozicijoje.
- Konvolucijoje naudojamas **parametru pasidalijimas** reiškia, kad vietoj to, kad kiekvienoje pozicijoje nagrinėti atskirus parametru rinkinius, nagrinėjamas vienas rinkinys.

Parametru pasidalinimas



Išretintos sąveikos ir parametru pasidalijimo įtaka kraštams vaizde nustatyti



Ekvivalentiškumas

- Sakoma, kad **funkcijos yra ekvivalentiškos**, kai pakeitus įvestį, tokiu pat būdu pasikeičia ir išvestis.
- Funkcija $f(x)$ yra **ekvivalenti** funkcija $g(x)$, jei $f(g(x)) = g(f(x))$.
- Konvoliucijos atveju parametru dalijimasis lemia tai, kad sluoksnis turėtų savybę, vadinamą **ekvivalentiškumu** pakeitimo operacijai.
- Tegu g yra bet kokia funkcija, perskaičiuojanti įeitį, pvz., patraukia ją, tuomet **konvoliucijos funkcija yra ekvivalenti** funkcija g .

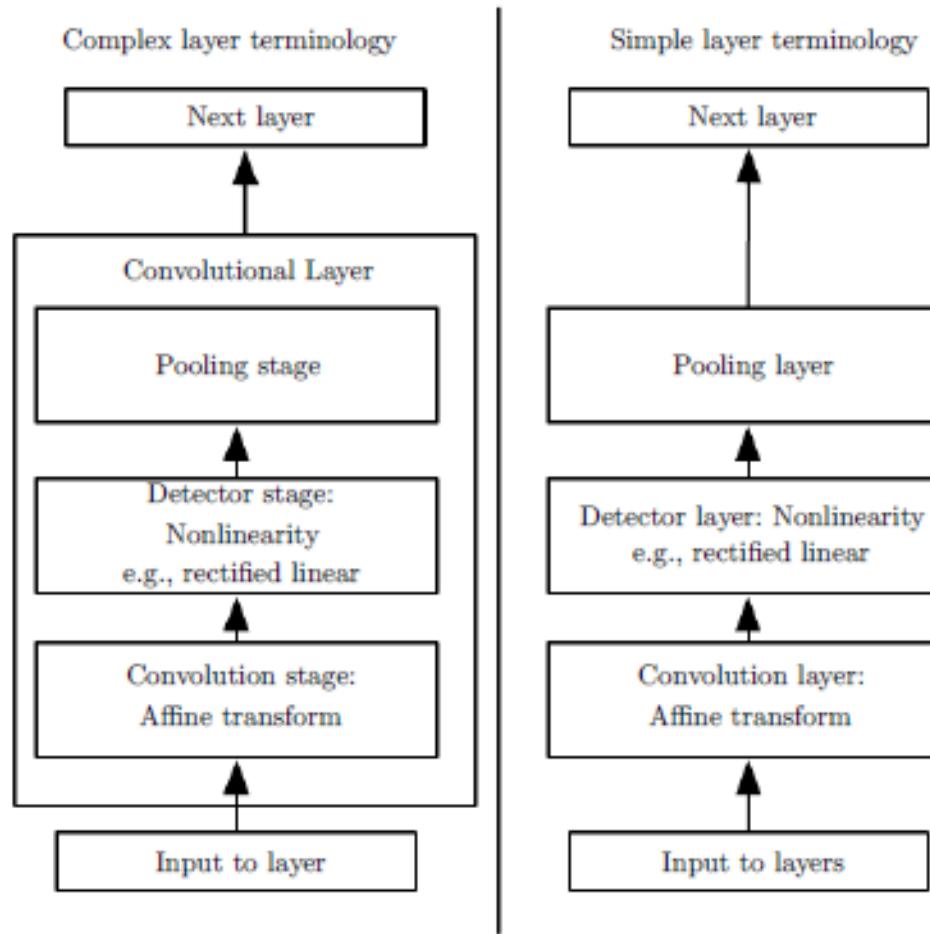
Ekvivalentiškumas

- Tegu I yra funkcija, apskaičiuojanti paveikslo ryškumą (angl. *brightness*), o g – funkcija, **atvaizduojanti vieną funkciją** į kitą taip, kad $I' = g(I)$ būtų funkcija, kur $I'(x, y) = I(x - 1, y)$, t.y. tokia funkcija pastumia kiekvieną pikselį per vieną poziciją į dešinę.
- Jei taikysime šią transformaciją funkcijai I , paskui taikysime konvoliucijos operaciją, **rezultatas bus toks pat**, kurį gautume pradžioje pritaikę konvoliuciją funkcijai I' , paskui transformaciją g .

Ekvivalentiškumas

- Tai naudinga tuomet, kai žinome, kad tam tikra nedidelio skaičiaus kaimyninių pikselių funkcija yra naudinga, kai taikoma **daugelyje įvesties vietų**.
- Pavyzdžiui, apdorojant vaizdus, naudinga **nustatyti briaunas** (kraštus) pirmame konvolucinio tinklo sluoksnuje.
- Briaunos yra beveik visur, taigi, tikslinga pasidalinti šia informacija **per visą paveikslėlį**.
- Tačiau gali būti atveju, kai **nenorima dalintis parametrais** per visą vaizdą. Pavyzdžiui, apdorojant veido vaizdą, kiekviena jo dalis turi skirtingą informaciją.

Konvoliucinių neuroninių tinklų struktūra



Konvoliucinis sluoksnis

- Šio sluoksnio **pagrindinis tikslas** yra nustatyti požymių junginius iš ankstesnių sluoksninių ir atvaizduoti juos į **požymių žemėlapį** (angl. *feature map*).
- Vaizdas yra padalinamas, sukuriami lokalūs fragmentai ir vėliau jie sujungiami į **požymių žemėlapį**, kurio dydis $m_2 \times m_3$.
- Toks žemėlapis saugo informaciją, kaip gerai **požymis atitinka jam taikomą filtrą**.
- Kiekvienas filtras yra mokomas „erdviniu būdu“, atsižvelgiant į **trimatės struktūros vietą**, kuriai jis taikomas.

Konvoliucinis sluoksnis

- Kiekviename sluoksnyje yra m_1 **filtras**.
- Kiek filtrų bus taikoma priklauso nuo požymių žemėlapių **trimatės struktūros gylio**.
- Kiekvienas filtras aptinka **tam tikrą požymį**.
- l -ojo sluoksnio **išvestis** $Y_i^{(l)}$ yra sudaryta iš $m_1^{(l)}$ požymių žemėlapių, kurių dydis $m_2^{(l)} \times m_3^{(l)}$.

Konvoliucinis sluoksnis

- i -tasis požymių žemėlapis, žymimas $Y_i^{(l)}$,
apskaičiuojamas:

$$Y_i^{(l)} = B_i^{(l)} + \sum_{j=1}^{m_1^{(l-1)}} K_{i,j}^{(l)} * Y_j^{(l-1)}$$

- Čia $B_i^{(l)}$ yra *bias* matrica
- $K_{i,j}^{(l)}$ yra **filtras** (branduolis), kurio dydis
 $2h_1^{(l)} + 1 \times 2h_2^{(l)} + 1$.
Jis sujungia j -tąjį požymių žemėlapį $(l - 1)$ -jame
sluoksnje su i -tuoj požymių žemėlapį l -jame
sluoksnje.

Konvoliucinis sluoksnis

- Šių konvoliucinių sluoksninių sujungimas su kitais sluoksniais **leidžia klasifikuoti informaciją** kaip tai atliekama **žmogaus regoje**.
- **Intuityviai suprantama**, kad iš pikselių sudaromi kraštai, iš kraštų formos, iš formų sudėtingesni objektai.

Netiesiškumo sluoksnis

- **Netiesiškumo sluoksnis** (angl. *non-linearity layer*) sudarytas iš aktyvacijos funkcijos, kuri paima požymiu žemėlapį, gautą konvoliucijos sluoksnyje, ir sukuria **aktyvacijos žemėlapį**.
- Kaip ir daugiasluoksniaiame perceptrone, dažniausios **aktyvacijos funkcijos**:
 - sigmoidinė
 - hiperbolinis tangentas
- Pastaruoju metu siūloma naudoti **ištaisymo tiesinę funkciją** (angl. *rectified linear unit* (ReLU)) (žr. tolesnes skaidres).

Netiesiškumo sluoksnis

- Tegu l yra **netiesiškumo sluoksnis**. Jis paima požymiu žemėlapį $Y_i^{(l-1)}$ ir generuoja aktyvacijos žemėlapį $Y_i^{(l)}$:

$$Y_i^{(l)} = f(Y_i^{(l-1)})$$

- $Y_i^{(l)} \in \mathbb{R}^{m_1^{(l)} \times m_2^{(l)} \times m_3^{(l)}}$,
- $Y_i^{(l-1)} \in \mathbb{R}^{m_1^{(l-1)} \times m_2^{(l-1)} \times m_3^{(l-1)}}$,
- $m_1^{(l)} = m_1^{(l-1)} \wedge m_2^{(l)} = m_2^{(l-1)} \wedge m_3^{(l)} = m_3^{(l)}$.
- Tai gali būti interpretuojama kaip $m_1^{(l-1)}$ **dvimačiai požymiu žemėliai**, kurie generuoti $m_1^{(l-1)}$ filtrų ($l - 1$)-ajame konvoliuciniame sluoksnyje. Šių žemėlapiai dydžiai $m_2^{(l-1)} \times m_3^{(l-1)}$.

Ištaisymo sluoksnis

- Ištaisymo sluoksnis (angl. *rectification layer*) reikšmes pakeičia į **absoliučius** jų dydžius.
- Tarkime l yra ištaisymo sluoksnis. Jis paima aktyvacijos žemėlapį $Y_i^{(l-1)}$ iš $(l - 1)$ -ojo netiesiškumo sluoksnio ir sukuria **ištaisytaą aktyvacijos žemėlapį** $Y_i^{(l)}$:

$$Y_i^{(l)} = |Y_i^{(l-1)}|$$

Netiesiškumo ir ištaisymo sluoksnių apjungimas

Kaip ir netiesiškumo sluoksnyje, ištaisymo sluoksnyje požymių žemėlapių dydis nepakinta, todėl dažnai šie **sluoksniai apjungiami į vieną**:

$$Y_i^{(l)} = \left| f\left(Y_i^{(l-1)}\right) \right|$$

Ištaisymo tiesinė funkcija

- **Ištaisymo tiesinė funkcija** (angl. *rectified linear unit*, ReLU) yra specifinė funkcija, apjungianti netiesiškumo ir ištaisymo sluoksnius:

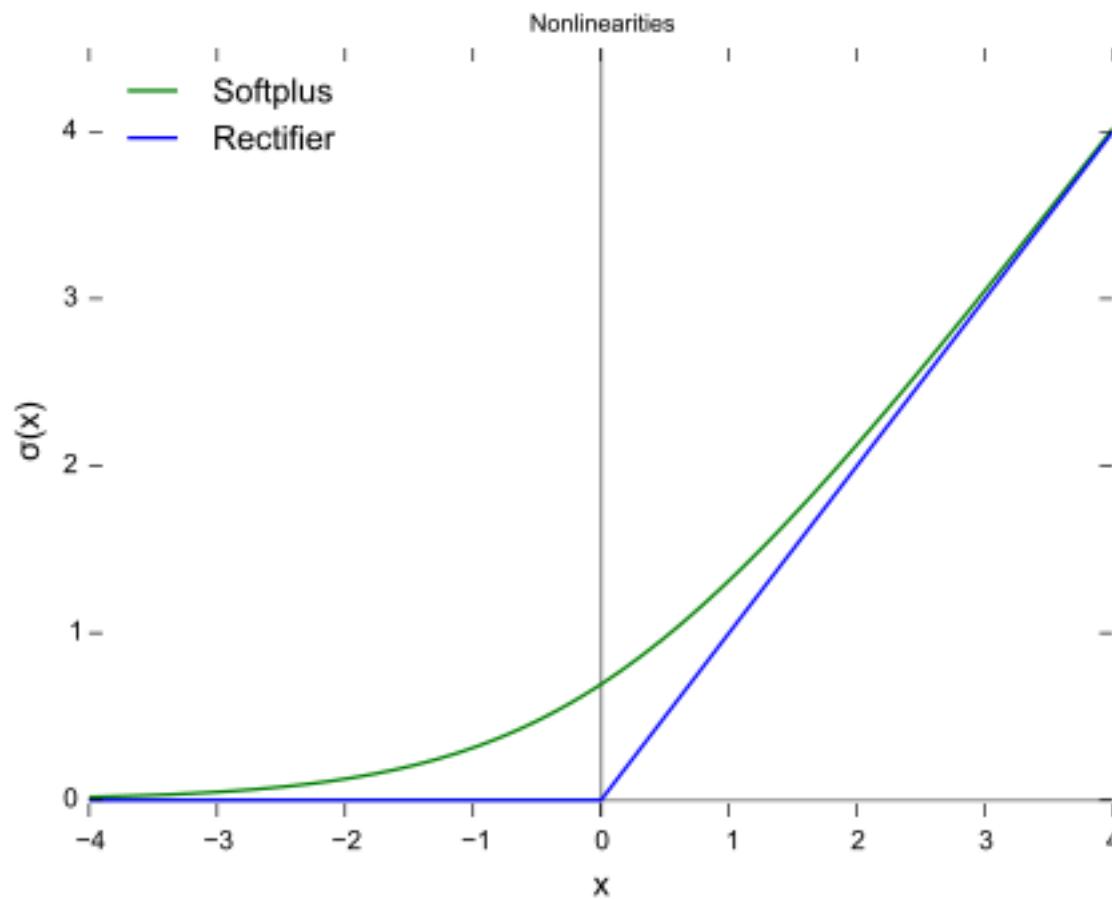
$$f(x) = \max(0, x).$$

- Galimi ir kiti variantai:
- Tolydi aproksimacija yra **softplus** funkcija:

$$f(x) = \ln(1 + e^x)$$

- Jos išvestinė $f'(x) = \frac{1}{1+e^{-x}}$ yra **sigmoidinė** (logistinė funkcija).

Softplus vs Rectifier



Ištaisymo tiesinė funkcija

Ištaisymo tiesinė funkcijos privalumai lyginant su sigmoidine ar hiperboliniu tangentu:

- Efektyviau skleidžiamas **gradientas**. Labai **paprastas gradiento** skaičiavimas (arba 0, arba 1, priklausomai nuo x ženklo).
- Dėl to kad neigiamos reikšmės prilyginamos nuliui, **išretinimas** tampa dar aktualesniu. Toks išretinimas yra naudingas siekiant atsparumo triukšmams.
- Operacijos labai **paprastos**, nėra **atliekama jokių sudėtingų veiksmų**.
- **Pagreitėja** neuroninio tinklo mokymas.

ReLU variantai

Dažniausiai taikomi ReLU variantai:

- **Su triukšmu** (angl. *noisy* ReLU):

$$f(x) = \max(0, x + Y), Y \sim \mathcal{N}(0, \sigma(x))$$

Sėkmingai naudojama apribotose Boltzmeno mašinose (angl. *restricted Boltzmann machines*) **kompiuterinės regos** uždaviniams spręsti

- **Nesandarus** (angl. *leaky* ReLUs):

$$f(x) = \begin{cases} x, & \text{jei } x > 0 \\ 0,01x, & \text{jei } x \leq 0 \end{cases}$$

čia leidžiami maži nenuliniai gradientai.

Sujungimo sluoksnis

- **Sujungimo sluoksnis** (angl. *pooling* arba *subsampling*) yra atsakingas už aktyvacijos žemėlapį dydžio sumažinimą.
- Jis taikomas po kelių konvoliucijos ir netiesiškumo sluoksnį siekiant **sumažinti skaičiavimų apimtis** ir minimizuojant persimokymą.

Sujungimo sluoksnis

- l -tasis **sujungimo sluoksnis** turi du parametrus: **filtras** $F^{(l)}$ ir **žingsnis** $S^{(l)}$.
- Jis paima trimatę įvestį, kurios dydis $m_1^{(l-1)} \times m_2^{(l-1)} \times m_3^{(l-1)}$ ir grąžina dydžio $m_1^{(l)} \times m_2^{(l)} \times m_3^{(l)}$ išvestį, kur:
 - $m_1^{(l)} = m_1^{(l-1)}$
 - $m_2^{(l)} = (m_2^{(l-1)} - F^{(l)})/S^{(l)} + 1$
 - $m_3^{(l)} = (m_3^{(l-1)} - F^{(l)})/S^{(l)} + 1$

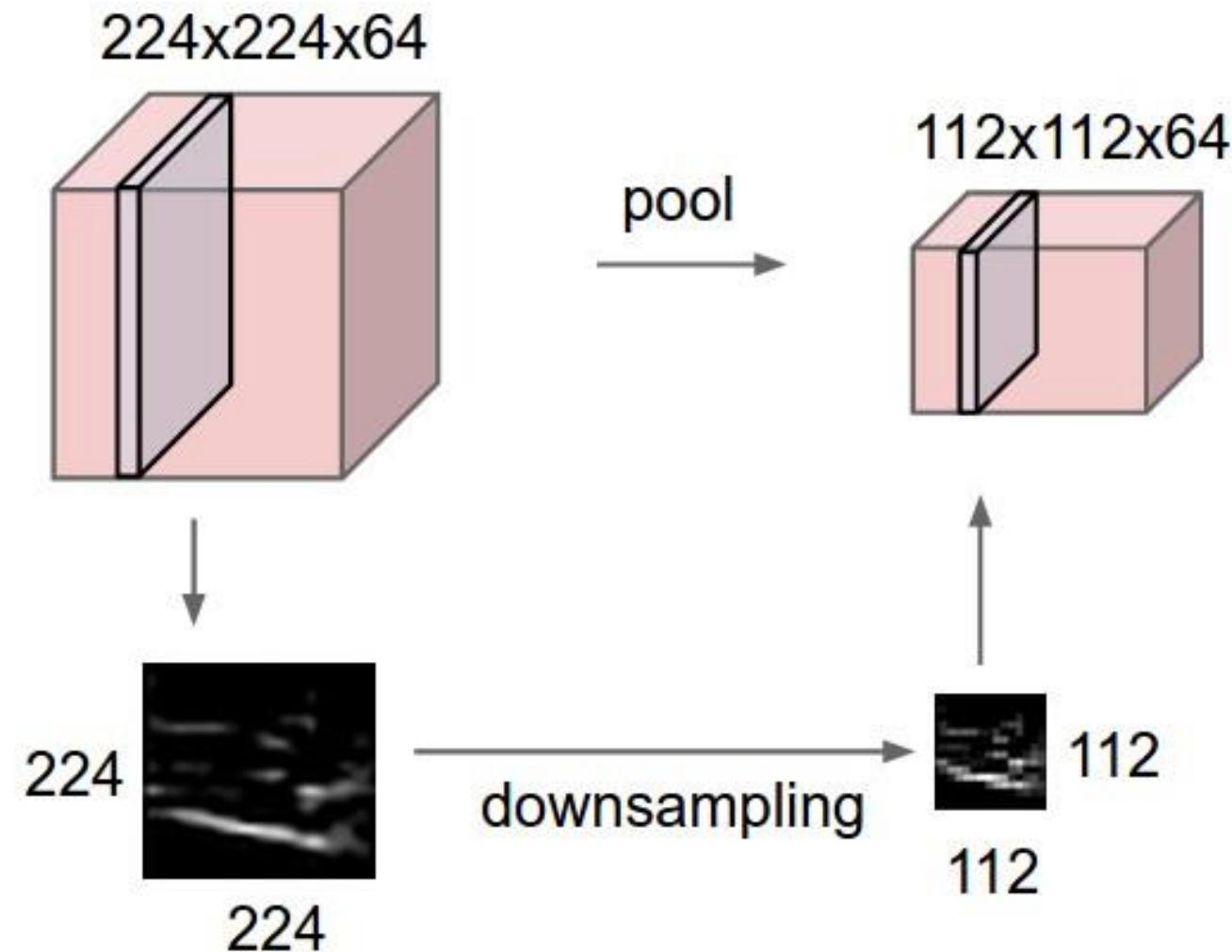
Sujungimo sluoksnis

- Pagrindinė idėja yra pateikti pakeitimo invariacią **atsižvelgiant į atpažinimo uždavinio** specifiką.
- **Požymiu nustatymas** yra daug svarbiau nei tiksliai požymiu vieta.
- Taigi, sujungimo sluoksnyje siekiama išsaugoti nustatytaus požymius, **atmetant mažiau svarbią** informaciją.

Sujungimo sluoksnis

- Apibrėžiamas langas $F^{(l)} \times F^{(l)}$ ir tokiu būdu **mažinimas duomenų kiekis** iki vieneto.
- **Langas yra traukiamas** per $S^{(l)}$ pozicijas. Duomenų mažinimas kartojamas kiekvienoje lango pozicijoje, kol sumažinamas įvesties erdvė.

Sujungimo sluoksnis

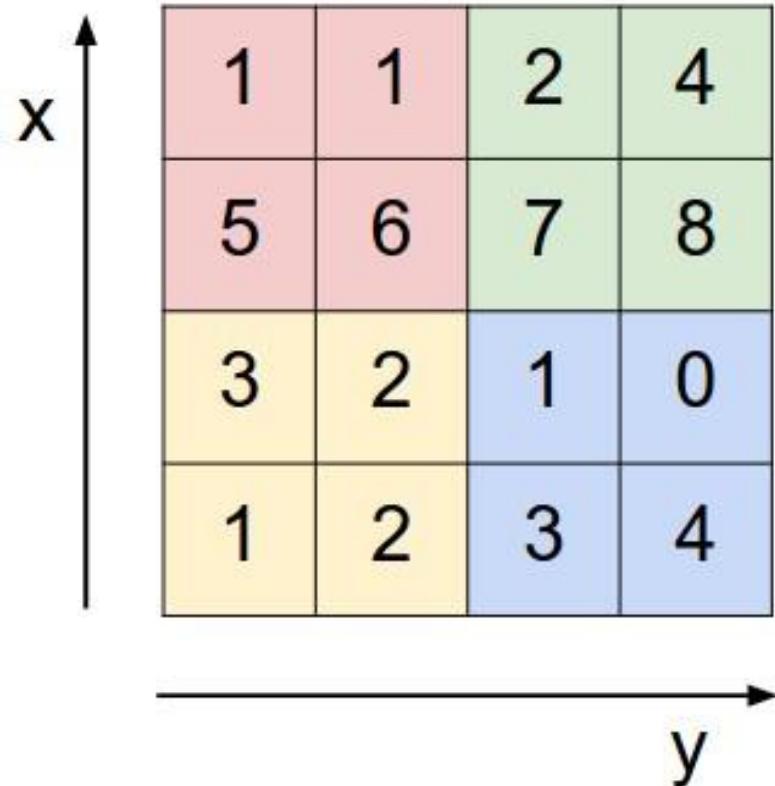


Sujungimo sluoksnis

- Dažniausi metodai: **maksimalus sujungimas** (angl. *max pooling*) ir **vidutinis sujungimas** (angl. *average pooling*).
- Maksimalaus sujungimo atveju ieškoma **didžiausios reikšmės** lange, vidutinio sujungimo – **vidutinės** reikšmės.
- Naudojant **maksimalų sujungimą**, tinklas greičiau apsimoko ir tiksliau sprendžia atpažinimo uždavinius.

Sujungimo sluoksnis

Single depth slice



max pool with 2x2 filters
and stride 2

The resulting 2x2 output tensor from the max pooling operation:

6	8
3	4

Sujungimo sluoksnis

- Sujungimo sluoksnje be mažinimo metodo, svarbus ir **žingsnio** $S^{(l)}$ parinkimas. Jis nurodo, ar langas persidengs, ar ne.
- Jei $F^{(l)} > S^{(l)}$, **langai perdengs** vienas kitą, filtras bus taikomas kelis kartus toje pačioje vietoje.
- Priklausomai nuo šios sąlygos, sujungimo sluoksnis vadinamas **persidengiančiu** ar **nepersidengiančiu** (angl. *overlapping or non-overlapping pooling*)

Pilnai sujungtas sluoksnis

- **Pilnai sujungti sluoksniai** (angl. *fully connected layers*) praktiškai yra daugiasluoksniai perceptronai (dažniausiai dviejų ar trijų paslėptų sluoksnį), kurie atvaizduoja $m_1^{(l-1)} \times m_2^{(l-1)} \times m_3^{(l-1)}$ aktyvacijos trimates įvestis iš ankstesnių sluoksniu junginių **į klasį tikimybių pasiskirstymą**.
- Gaunama $m_1^{(l-i)}$ **išvesčiu**, čia i nurodo perceptrono sluoksnį skaičių.

Pilnai sujungtas sluoksnis

- **Pagrindinis skirtumas** nuo standartinio daugiasluoksnio perceptrono yra tai, kad standartiniu atveju įvestis yra **vektorius**, konvoliucinių tinklų atveju – **trimatė struktūra**.

$$y_i^{(l)} = f(a_i^{(l)})$$

- Jei $(l - 1)$ yra pilnai sujungtas sluoksnis, tai:

$$a_i^{(l)} = \sum_{j=1}^{m_1^{(l-1)}} w_{ij}^{(l)} y_j^{(l-1)}$$

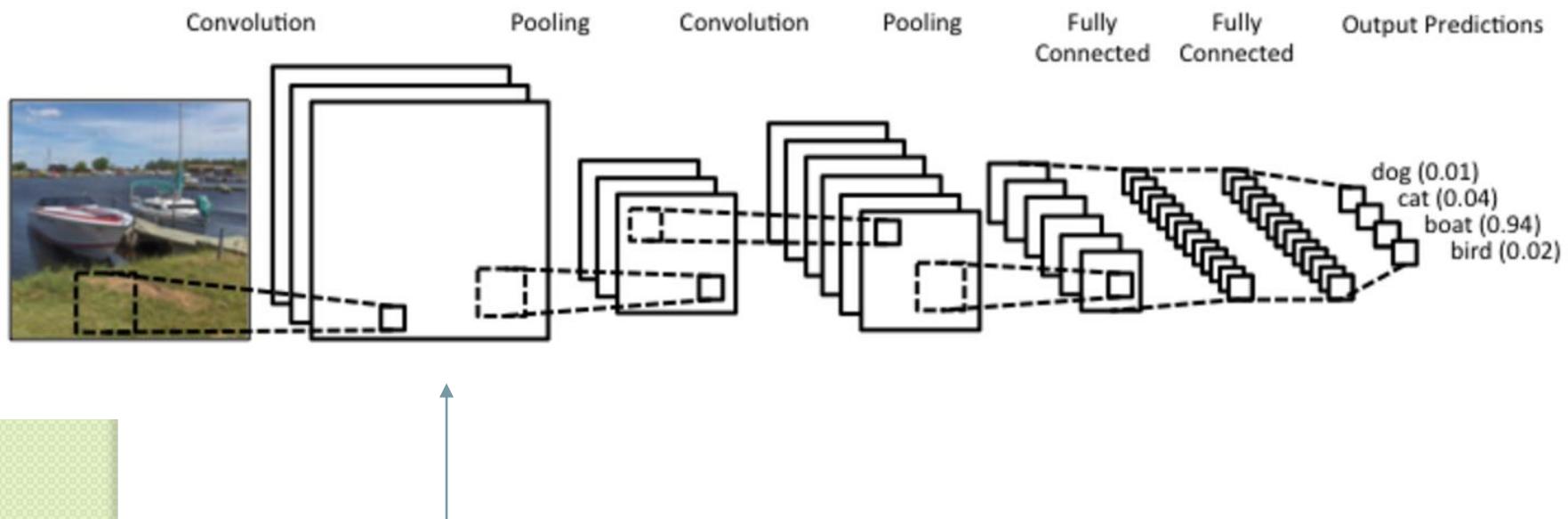
- Priešingu atveju:

$$a_i^{(l)} = \sum_{j=1}^{m_1^{(l-1)}} \sum_{r=1}^{m_2^{(l-1)}} \sum_{s=1}^{m_3^{(l-1)}} w_{ijrs}^{(l)} (Y_j^{(l-1)})_{rs}$$

Tinklo mokymas

- Belieka taip sudarytą tinklą **išmokyti**.
- Tai yra rasti tinkamas **svorių** $w_{ij}^{(l)}$ ir $w_{ijrs}^{(l)}$ reikšmes.
- $w_{ij}^{(l)}$ – svoriai pilnai sujungtame sluoksnyje (analogija su daugiasluoksniu perceptronu).
- $w_{ijrs}^{(l)}$ – filtrų reikšmės (ankstesnėse skaidrėse $K_{i,j}^{(l)}$)
- Siekiama nustatyti **kiekvienos klasės tikimybę**, pagrįstą aktyvavimo žemėlapiais, kurie sudaryti **konvolucijos, netiesiškumo, ištaisymo** ir **pilno sujungimo** sluoksnį.

Konvoliucinis neuroninis tinklas



Sluoksnių tiek, kiek
pritaikysime filtru

Konvoliucinis neuroninis tinklas

Paprastą ir lengvai suprantamą **apibendrinimą** galima rasti [https://github.com/brohrer/public-hosting/raw/master/How CNNs work.pdf](https://github.com/brohrer/public-hosting/raw/master/How%20CNNs%20work.pdf)

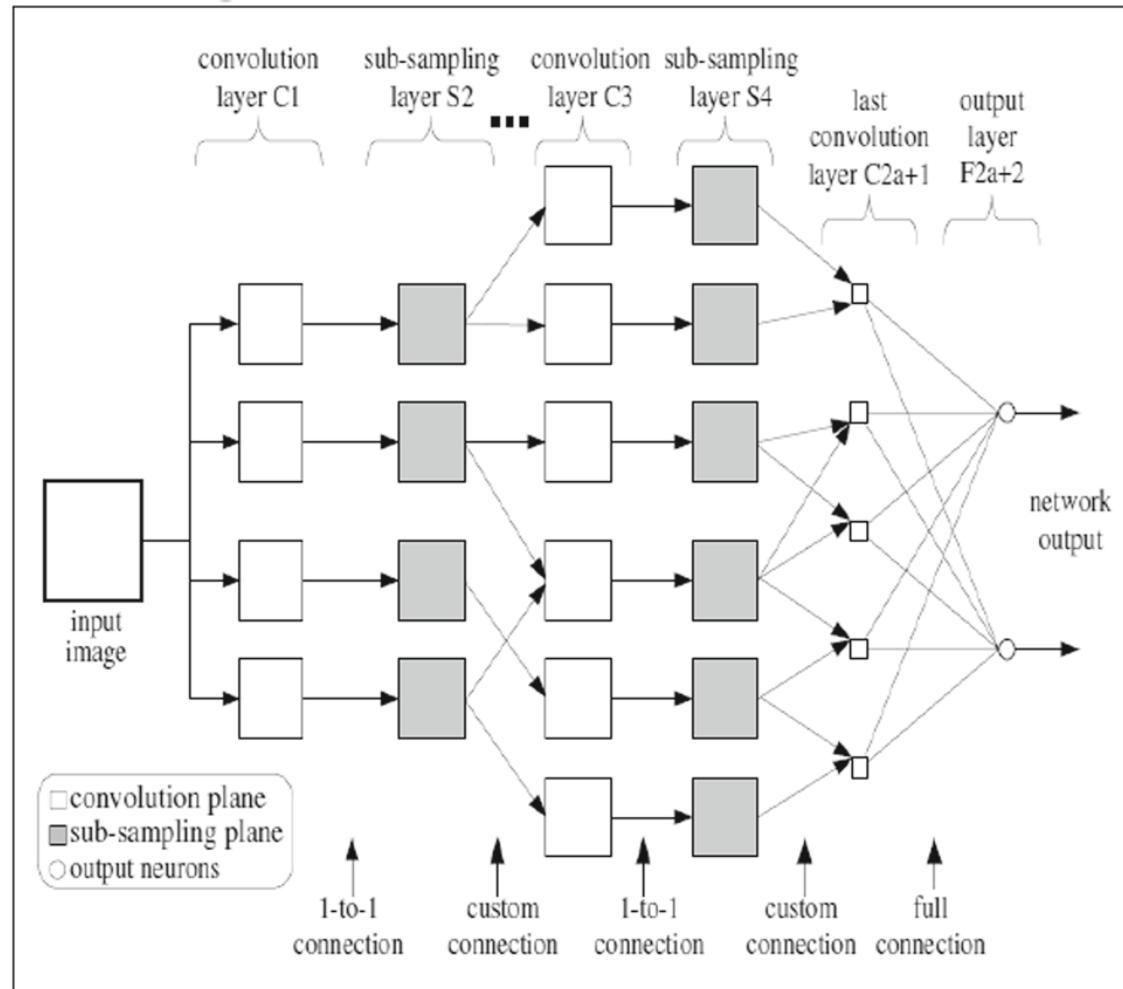
Konvoliucinio neuroninio tinklo sluoksniai

- **Konvolucijos** sluoksnis (**filtrai-svoriai**) (*convolution layer*)
 - **Ištaisymo** sluoksnis (dažnai *ReLU*)
- **Sujungimo** sluoksnis (*pooling layer, downsampling layer, sub-sampling layer*) (dažnai *max-pooling*)
- P. S. Iprastai šie sluoksniai pasikartoja kelis kartus
- **Pilnai sujungtas** sluoksnis (*fully connected layer, dense layer*)

Konvoliucinio sluoksnio parametrai

- **Filtrų skaičius**
- **Filtro** (*branduolio, kernel*) **dydis**
- **Lango žingsnis**: per kiek pikselių langas bus patrauktas (įprastai konvoliuciiniame sluoksnyje lygus 1, sujungimo (*pooling*) lygus 2)

Konvoliucinis neuroninis tinklas veidams atpažinti



Anuse, A., & Vyas, V. (2016). A novel training algorithm for convolutional neural network. *Complex & Intelligent Systems*, 2(3), 221-234

Svorų kiekis

- Tarkime įvestis – 74×74 paveiksliukas.
- C_1 – konvoliucinis sluoksnis su 6 filtrois,
kiekvienas jų yra 11×11 dydžio, dar yra 6 bias.
Svorų skaičius $11 \times 11 \times 6 + 6 = 732$.
- S_2 – sujungimo sluoksnis, neturintis svorų.
- C_3 – konvoliucinis sluoksnis su 16 filtro,
kiekvienas jų yra 11×11 dydžio, dar yra 16 bias.
Svorų skaičius $11 \times 11 \times 16 + 16 = 1952$.
- S_4 – sujungimo sluoksnis, neturintis svorų.

Svorių kiekis

- C_5 – konvoliucinis sluoksnis su 120 filtru, kiekvienas jų yra 11×11 dydžio, dar yra 120 bias. **Svorių skaičius** $11 \times 11 \times 120 + 120 = 14640$.
- F_6 – pilnai sujungtas sluoksnis, turintis 84 neuronus. **Svorių skaičius** $120 \times 84 = 10080$.
- Tinklas skirtas 25 asmenims atpažinti, tad išvesties sluoksnis sudarytas iš 25 neuronų. **Svorių skaičius** $85 \times 25 = 2100$.
- Viso svorių **29504**.

Konvoliucinio neuroninio tinklo mokymas

- Įprastai **tiesioginio sklidimo neuroniniai tinklai** mokomi minimizuojant paklaidos funkciją, taikant **gradientiniu nusileidimu** grįstus algoritmus, pavyzdžiui, klaidos skleidimo atgal.
- **Konvoliuciniai neuroniniai tinklai** mokomi pagal **stochastiniu gradientiniu nusileidimu** grįstus algoritmus. Čia taip pat gali būti taikoma klaidos skleidimo atgal algoritmo strategija.

Gradientinis nusileidimas

- Tarkime norime minimizuoti paklaidą

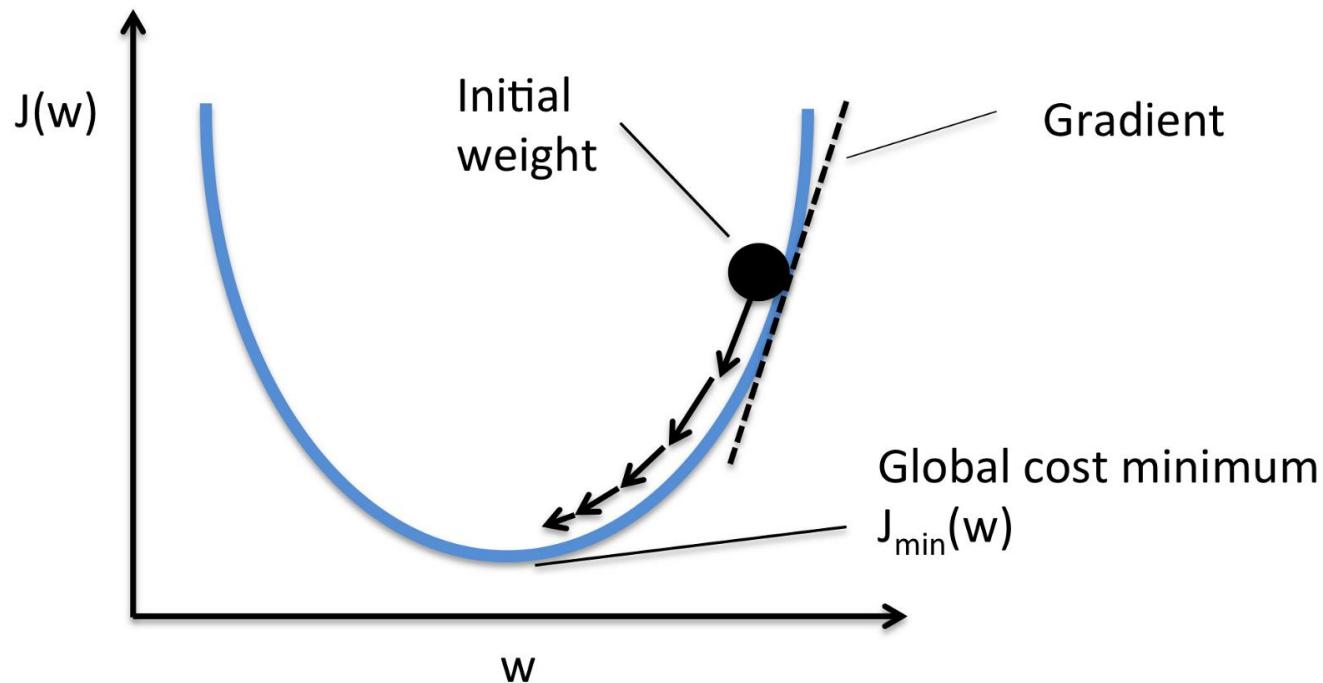
$$E(W) = \frac{1}{m} \sum_{i=1}^m E_i(W)$$

- Taikant **gradientinio nusileidimo algoritma**, svoriai keičiami pagal formulę:

$$\begin{aligned} W(t+1) &= W(t) - \eta \nabla E(W) = \\ &= W(t) - \eta \frac{1}{m} \sum_{i=1}^m \nabla E_i(W). \end{aligned}$$

- T. y. siekiama, kad paklaida būtų **minimali visiems mokymo duomenims**.

Gradientinis nusileidimas



Judama antigradiento kryptimi.

Stochastinis gradientinis nusileidimas

- **Stochastinio gradientinio nusileidimo** (*stochastic gradient descent*, SGD) algoritme tikras funkcijos $E(W)$ gradientas aproksimuojamas gradientu, gautu pagal vieną mokymo duomenų įrašą:

$$W(t + 1) = W(t) - \eta \nabla E_i(W)$$

- T. y. siekiama, kad paklaida būtų **minimali *i*-tajam mokymo duomenų** įrašui.

Stochastinis gradientinis nusileidimas + *momentum*

- Jei **stochastinio gradientinio nusileidimo** algoritme taikomas ***momentum*** metodas, svoriai keičiami pagal formule:

$$W(t + 1) = W(t) - \eta \nabla E_i(W) + \alpha \Delta W$$

- Čia ΔW – svorių pokytis gretimose iteracijose.

Paketo sąvoka neuroniniuose tinkluose

- Neuroniniuose tinkluose dažnai sutinkama **paketo (*batch*)** sąvoka, o tiksliau jos dydis (*batch size*).
- Iprastai apibrėžiama, kad **paketo dydis** – tai objektų, pateikiamų į tinklą, skaičius vienos iteracijos metu. Be to, tai parodo, keliems objektams reikia **verti gradientą** keičiant svorius.
- Taikant tradicinį **SGD** optimizavimo algoritmą, paketo dydis lygus **1**.
- Tačiau gali būti naudojamas ir modifikuotas SGD, kai gradientas vertinamas ne visiems mokymo duomenimis (kaip yra gradientinio nusileidimo algoritme), o tik **esantiems einamajame pakete**.

Adam algoritmas

- Dažnai nuo tinklo mokymo (**optimizavimo**) algoritmo priklauso **mokymo trukmė** (kuri gali svyruoti nuo kelių minučių iki kelių dienų).
- **Adam algoritmas** yra vienas populiariausių algoritmų, naudojamų giliesiems neuroniniams tinklams mokyti.
- Tai **praplėstas stochastinio** gradientinio nusileidimo algoritmas.
- „the name **Adam** is derived from adaptive moment estimation.“

Kingma, D. P., & Ba, J. (2015). Adam: a method for stochastic optimization. In Conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015, <https://arxiv.org/abs/1412.6980>

Adam algoritmas

- Adam algoritmas apjungia dviejų metodų privalumus: AdaGrad (*Adaptive Gradient Algorithm*) ir RMSProp (Root Mean Square Propagation)
- Iprastai stochastinio gradientinio nusileidimo metode naudojama vienintelė mokymo greičio parametro reikšmė (*learning rate*) visų svorių keitime ir ji išlieka pastovi viso mokymo metu.
- Adam algoritme mokymo greičio parametras derinimas mokymo procese.

Algorithm 1: Adam, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t .

Require: α : Stepsize

Require: $\beta_1, \beta_2 \in [0, 1]$: Exponential decay rates for the moment estimates

Require: $f(\theta)$: Stochastic objective function with parameters θ

Require: θ_0 : Initial parameter vector

$m_0 \leftarrow 0$ (Initialize 1st moment vector)

$v_0 \leftarrow 0$ (Initialize 2nd moment vector)

$t \leftarrow 0$ (Initialize timestep)

while θ_t not converged **do**

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)

$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)

$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)

$\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)

$\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)

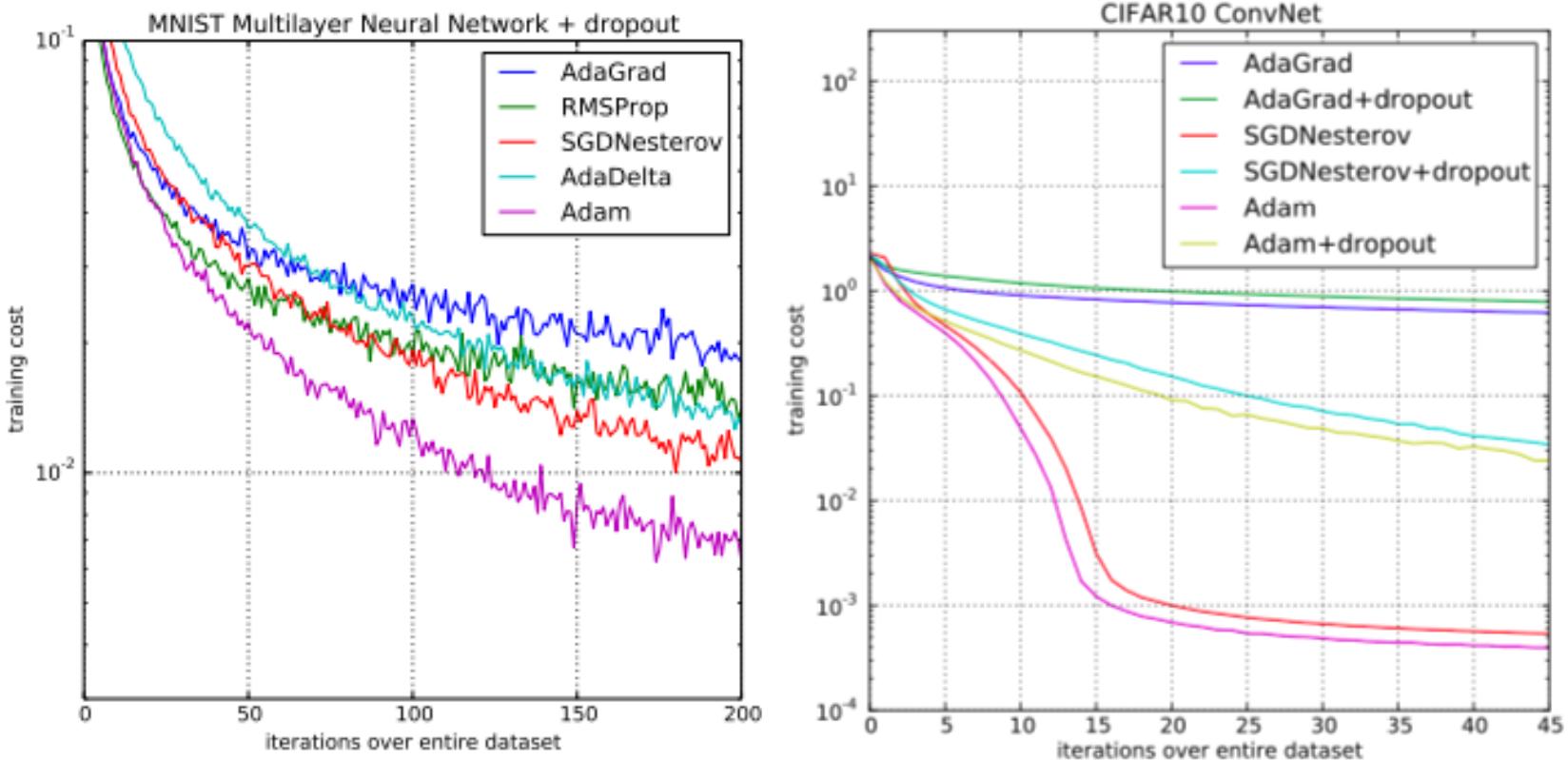
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)

end while

return θ_t (Resulting parameters)

Kingma, D. P., & Ba, J. (2015). Adam: a method for stochastic optimization. In Conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015, <https://arxiv.org/abs/1412.6980>

Adam algoritmas



Kingma, D. P., & Ba, J. (2015). Adam: a method for stochastic optimization. In Conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015, <https://arxiv.org/abs/1412.6980>

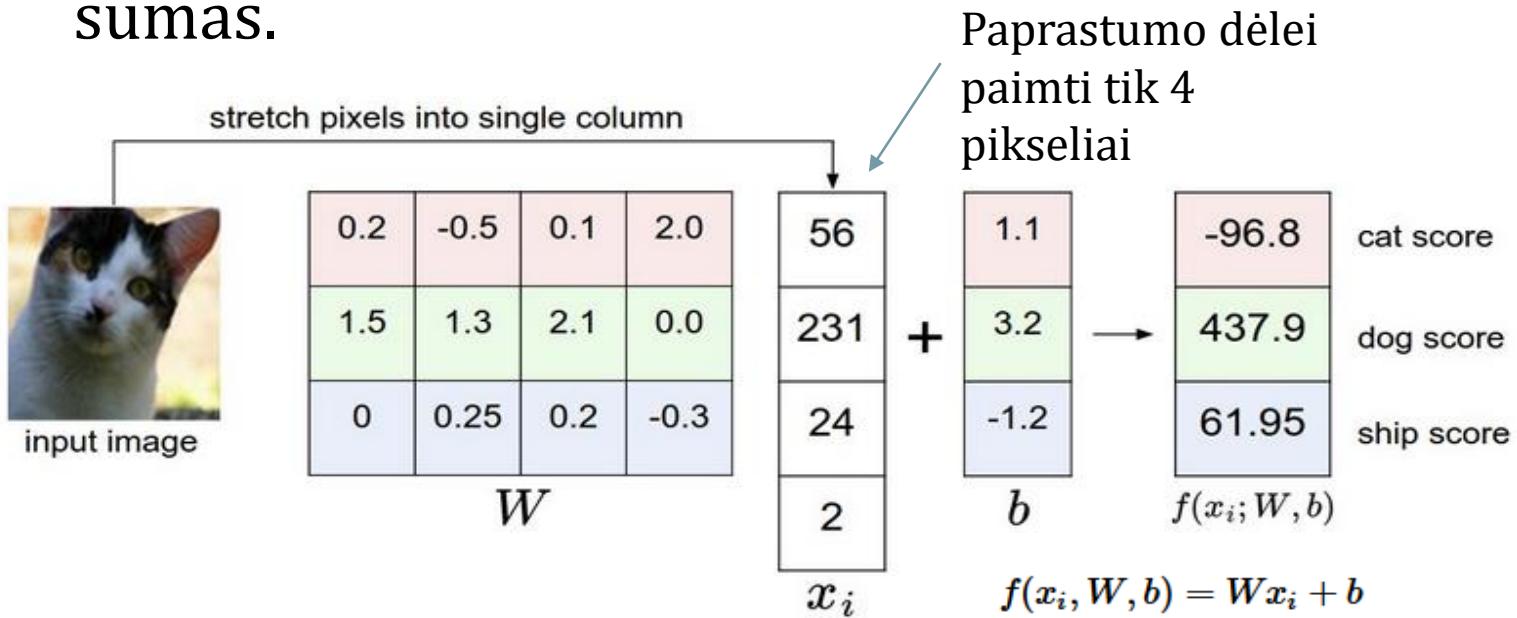
Optimizavimas neuroniniuose tinkluose (apibendrinimas)

- Optimizavimo metu tinklas **mokamas** (parenkami tinkami svoriai)
- **Algoritmai:**
 - Gradientinis nusileidimas (be/su *momentum*)
 - Stochastinis gradientinis nusileidimas (SGD) (be/su *momentum*) (su *Nesterov momentum*)
 - AdaGrad algoritmas
 - RMSProp algoritmas
 - Adam algoritmas
 - kiti

Daugiau info: <http://ruder.io/optimizing-gradient-descent/>

Duomenų priskyrimas klasėms

- **Tiesinis klasifikatorius** skaičiuoja kiekvienos klasės balus – visų pikselių reikšmių svertinės sumas.



Čia svoriai nėra tinkami, kadangi pagal išėjimų reikšmes nustatoma (didžiausia reikšmė 437,9), kad tai šuo (o ne katinas).

Iš <http://cs231n.github.io/linear-classify/>

Vaizdų pirminis apdorojimas

- Naudojant **RGB** spalvų sistemą, kiekvienas vaizdo pikselis gali įgauti reikšmes intervale **[0 ... 255]**.
- Dažnai reikia vaizdus **normuoti**.
- Vienas įprastų būdų – „**centravimas**“, kurio metu iš kiekvienos pikselio reikšmės atimamas vidurkis. Tuomet reikšmės atsiduria apytiksliai intervale **[-127 ... 127]**.
- Tolesnė procedūra pakeičia reikšmes, taip kad būtų intervale **[-1 ... 1]**.

Nuostolių funkcija

- **Nuostolių funkcija** (angl. *loss function, cost function, objective function*) reikalinga siekiant pamatuoti, kaip blogai (gerai) klasifikuojami duomenys.
- Kuo šios funkcijos **reikšmė didesnė**, tuo **klasifikavimas prastesnis**.

Daugiaklasių SVM nuostolių funkcija

- **Daugiaklasių atraminių vektorių klasifikatoriaus nuostolių funkcija** (angl. *Multiclass Support Vector Machine (SVM) loss*) sudaryta taip, kad SVM siekia, kad **teisingos klasės balas būtų aukštesnis** nei neteisingos, taikant tam tikrą fiksotą dydį Δ .
- Tarkime, i -tajam duomenų įrašui turime vaizdą x_i ir klasės žymę y_i .
- Funkcija $f(x_i, W)$ apskaičiuoja klasės balus kiekvienai klasei $s_j = f(x_i, W)_j, j = 1, \dots, K$. Čia K – klasių skaičius.
- Daugiaklasių SVM **nuostolių funkcija (loss)** i -tajam duomenų įrašui (ji turi būti minimizuojama)

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + \Delta)$$

Pavyzdys

- Tarkime turime **tris klasses** ir jų balai $s = [13, -7, 11]$. Teisinga klasė yra **pirmoji**. Tarkime $\Delta = 10$.
- **Gauname** $L_i = \max(0, -7 - 13 + 10) + \max(0, 11 - 13 + 10) = 0 + 8$
- Taigi, siekiama, kad teisingos klasės balas būtų didesnis nei neteisingos **dydžiu** Δ .



Reguliavimas

- Aptarta nuostolių funkcija turi **didelį trūkumą**. Tarkime, kad visiems mokymo duomenų įrašams nuostolių paklaida lygi nuliui. Vadinasi, svorių reikšmių aibė W **nėra vienintelė**.
- Jei esant svoriams W , paklaidos nulinės, tai jos **išliks nulinės ir esant svoriams λW , $\lambda > 1$** . Tokiu būdu svorių reikšmės gali labai išdidėti.
- Tam įvedamas **reguliavimo bauda** (angl. *regularization penalty*).

Reguliavimas

- Dažniausiai naudojama **L1 arba L2 norma:**

$$\text{L1: } R(W) = \sum_k \sum_l |W_{kl}|.$$

$$\text{L2: } R(W) = \sum_k \sum_l W_{kl}^2.$$

- Tuomet **nuostolių funkcija:**

$$L = \frac{1}{m} \sum_i L_i + \lambda R(W).$$

Δ reikšmės nustatymas

- Ši reikšmė gali būti parenkama **kryžminės patikros** metu.
- Dažnai **keičiamos reikšmės** nuo $\Delta = 1$ iki $\Delta = 100$.

Softmax funkcija

- Konvoluciiniuose neuroniniuose tinkluose klasių priskyrimui dažnai naudojama funkcija yra **softmax funkcija**, kuri yra logistinės regresijos apibendrinimas daugeliui klasių:

$$f_i(Z) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}, i = 1, \dots, K$$

- Čia K – klasių skaičius.
- Ši funkcija **mažesnes** reikšmes **sumažina, didesnes padidina**.
- Pvz., $[1 \ 2 \ 3 \ 4 \ 1 \ 2 \ 3] \rightarrow$
 $\rightarrow [0,024 \ 0,064 \ 0,175 \ 0,475 \ 0,024 \ 0,064 \ 0,175]$

Kryžminė entropija

- Taikant *softmax* funkciją, naudojama **kryžminės entropijos** (angl. *cross-entropy*) nuostolių funkcija kiekvienam i -tajam duomenų įrašui (ji turi būti minimizuojama):

$$L_i = -\log \left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \right)$$

- **P.S. Entropija** yra informacijos teorijoje naudojamas dydis, kuris apibūdina vidutinį informacijos kieki, kurį teikia vienas pranešimas (*Wikipedia*).

Kryžminė entropija

- Informacijos teorijos požiūriu, **kryžminė entropija** tarp tikrojo pasiskirstymo p ir i gauto pasiskirstymo q yra apibrėžiama taip:

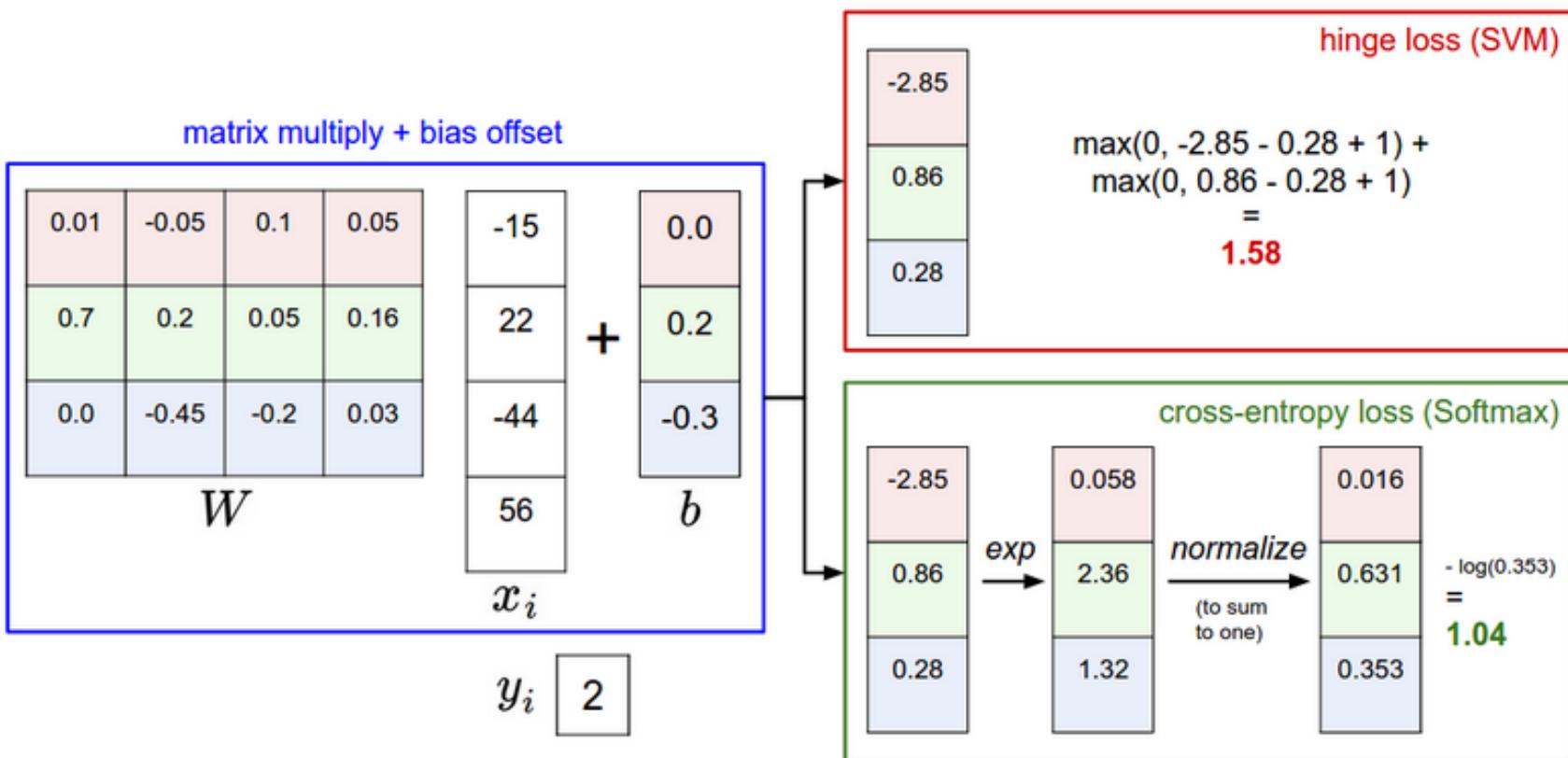
$$H(p, q) = - \sum_x p(x) \log q(x).$$

- Taikant **softmax** klasifikatoriu, minimizuojant **kryžminę entropiją**,
gauta klasių tikimybė $q = \frac{e^{sy_i}}{\sum_j e^{sj}}$,
tikrasis klasių pasiskirstymo $p = [0, \dots 1, \dots 0]$, kurį sudaro nuliukai ir vienas vienetas y_i pozicijoje.

Nesusipainiokime ...

- SVM klasifikatoruje – hinge loss, max-margin loss
- Softmax klasifikatoruje – cross-entropy loss

SVM vs. Softmax



Tinklo persimokymas

- Tinklo **persimokymas** (angl. *overfitting*) dažnai įvyksta, kai duomenų imtis salyginai maža, palyginus su parametru, kuriuos reikia nustatyti mokymo metu (išmokyti), skaičiumi.
- Ši problema ypač aktuali **giliojo mokymosi neuroniniuose tinkluose**.

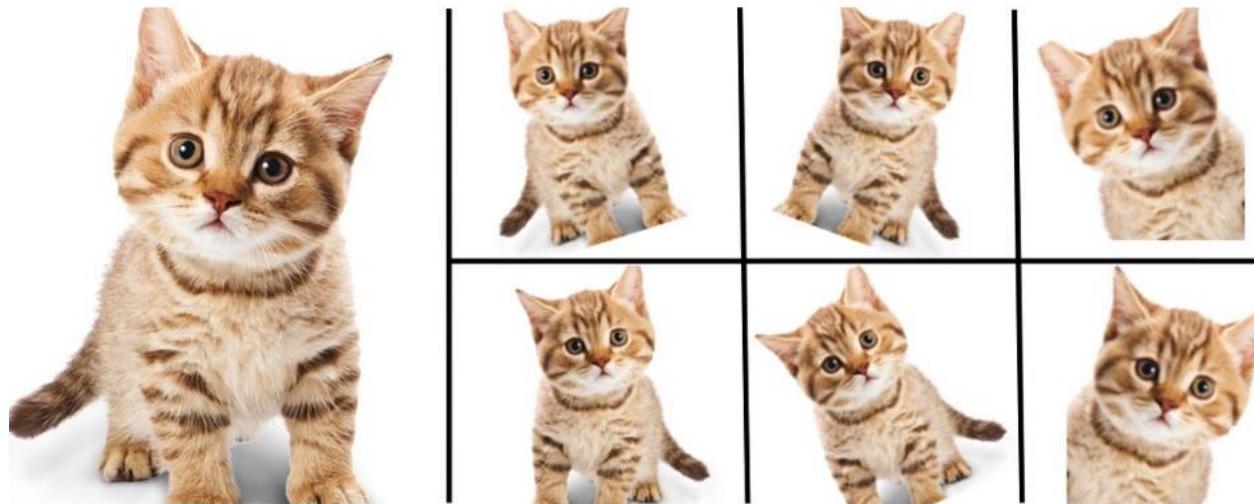
Tinklo persimokymas

Siekiant išvengti tinklo permokymo, taikomos **kelios strategijos**:

- Duomenų padidinimas (*augmentation*),
- Ankstyvas mokymo sustabdymas,
- **Išmetimo sluoksnis** (*dropout*),
- Svorų baudos L1 ir L2.

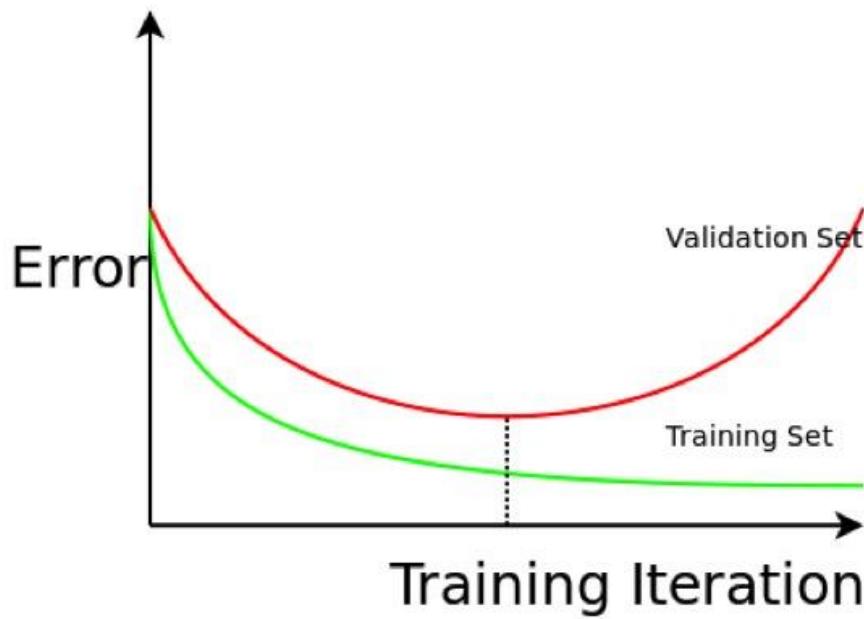
Tinklo persimokymas: duomenų padidinimas (*augmentation*)

- Siekiant gautu daugiau mokymo duomenų, iš turimų yra **sintetinami nauji**.
- Gali būti **taikomi įvairūs būdai**, pavyzdžiui, paveiksliuke atliekamos kelių pikselių transformacijos, tokios kaip pasukimas, mastelio keitimas ir kt.



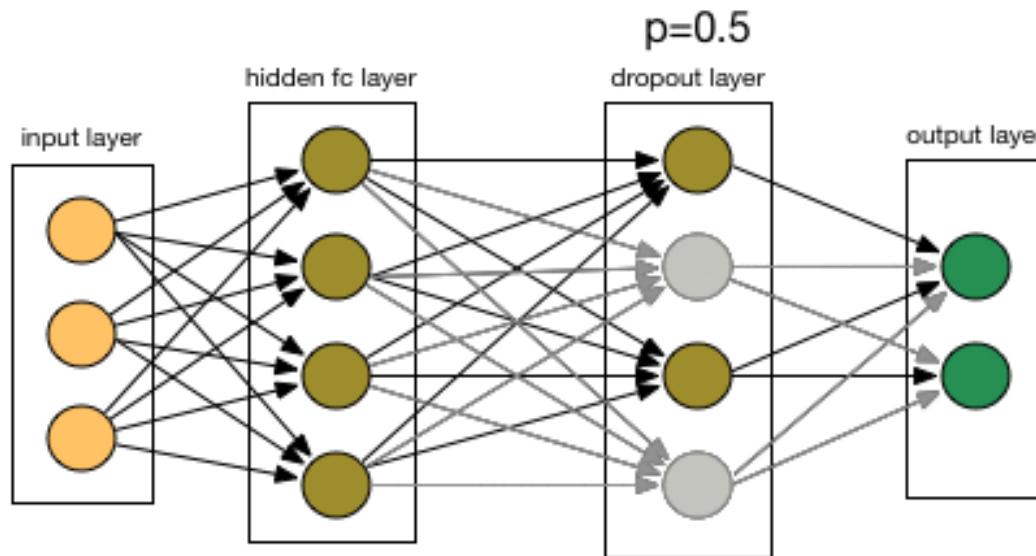
Tinklo persimokymas: duomenų padidinimas, ankstyvas stabdymas

- Tinklo mokymą **reikia stabdyti** tuomet, kai nors paklaida mokymo duomenims mažėja, bet jau pradeda **didėti paklaida validavimo** (testavimo) duomenims.



Tinklo persimokymas: išmetimo sluoksnio naudojimas

- Kiekvienoje mokymo iteracijoje **išmetimo** (angl. *dropout*) sluoksnis atsitiktinai **išmeta** kelis neuronus.



Tinklo persimokymas: išmetimo sluoksnio naudojimas

Kodėl išmetimo **sluoksnis veiksmingas**?

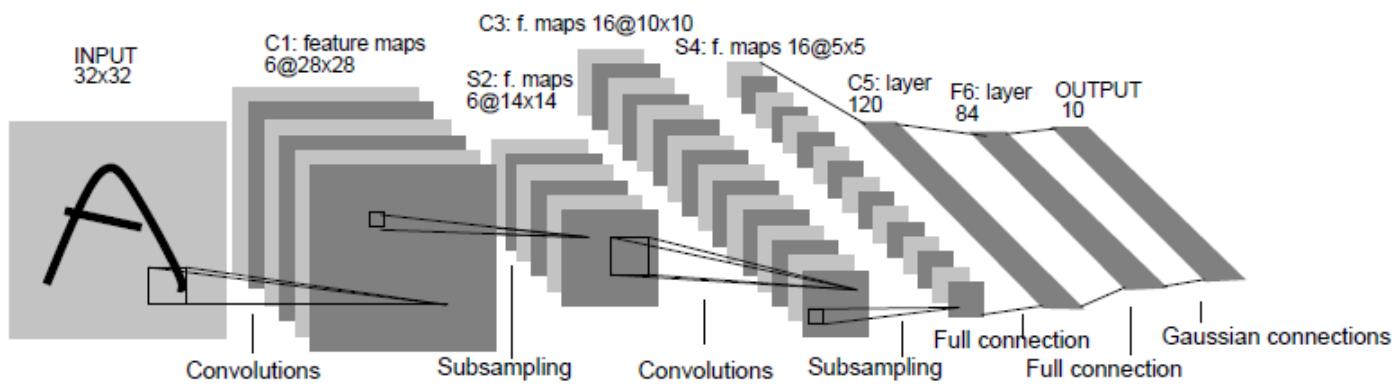
- Neuronai tampa **mažiau jautrūs** kitų neuronų svoriams, taigi, modelis tampa *robastiškas*.
- Išmetimo sluoksnis gali atitikti kelių modelių apibendrinimą, vadinama **ansambliu** (angl. *ensemble*). Pastaruoju metu ansambliai gauna tiksliausius mašininio mokymosi rezultatus.

Apibendrinimas

- Konvoliucijos sluoksniai (**filtrai-svoriai**)
- Ištaisymo sluoksnis (rectified linear units (ReLUs))
- Maksimalaus **sujungimo** sluoksnis (max-pooling)
- **Pilnai sujungtas** sluoksnis (fully connected layer)
- **Soft-max** klasifikavimas

Sėkminges tinklai: LeNet

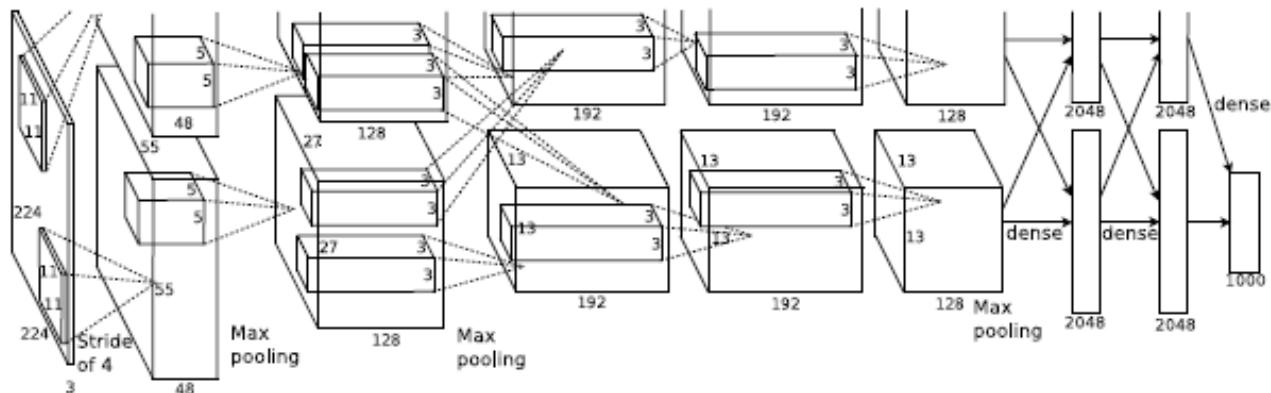
- **LeNet** – autoriai Yann LeCun ir kt., 1998,
rašytiems skaitmenims atpažinti



LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

Sėkminges tinklai: AlexNet

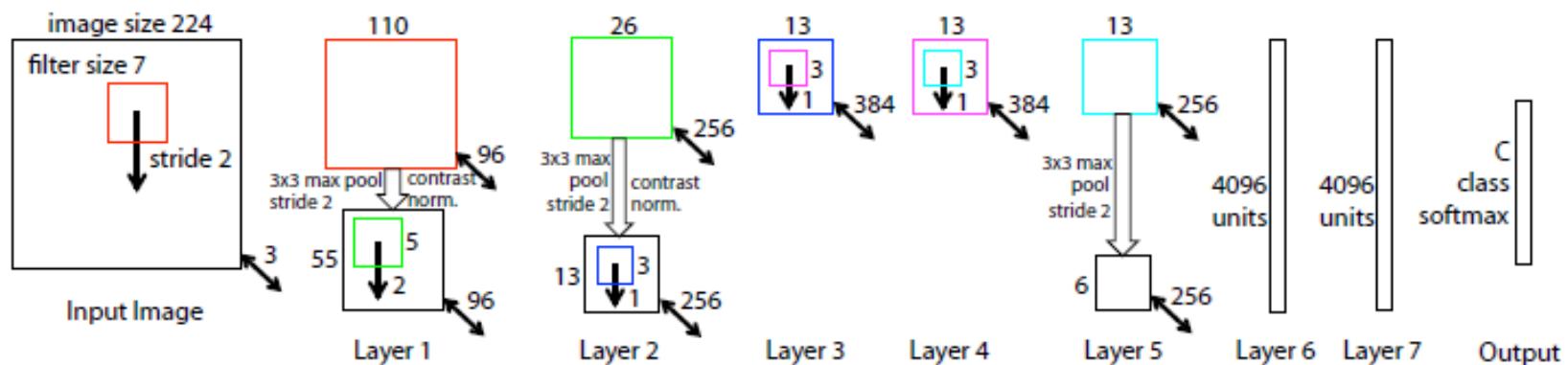
- **AlexNet** – autorai Alex Krizhevsky, Ilya Sutskever ir Geoff Hinton.
- Tai pirmasis darbas, išpopuliarinęs CNN. Dalyvavo konkurse „ImageNet **ILSVRC Challenge** in 2012“



Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Sékmingsi tinklai: ZF Net

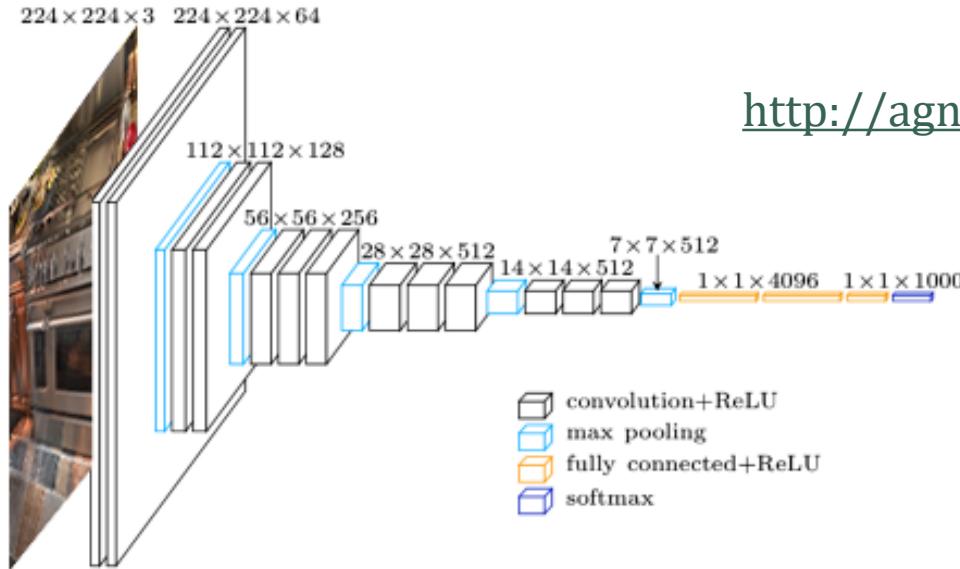
- **ZF Net** – autoriai Matthew Zeiler ir Rob Fergus.
- ILSVRC 2013 (Large Scale Visual Recognition Challenge) nugalėtojas.



Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818-833). Springer, Cham.

Sėkminges tinklai: VGGNet

- **VGGNet** – autoriai Simonyan, K. ir Zisserman ILSVRC 2014 antrosios vienos nugalėtojas.



Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. <https://arxiv.org/abs/1409.1556>

Sėkminges tinklai: ResNet

- **ResNet** (Residual Network) – autorai Kaiming He et al. ILSVRC 2015 nugalėtojas.

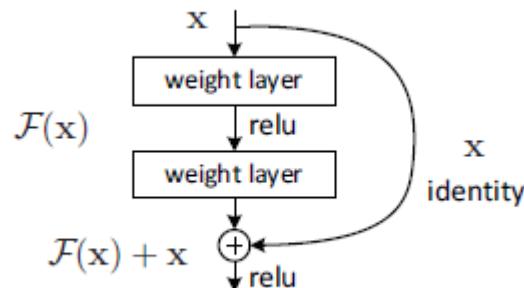
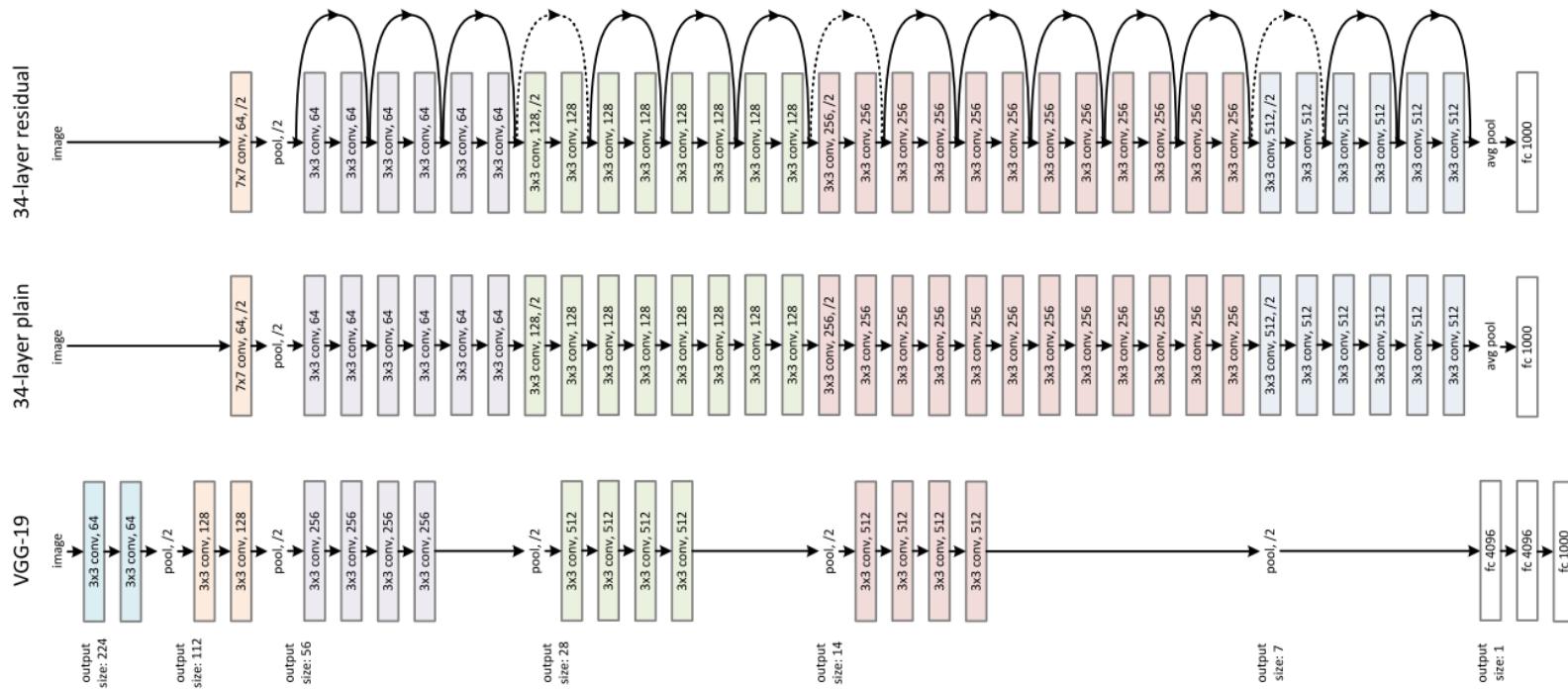


Figure 2. Residual learning: a building block.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
<https://arxiv.org/abs/1512.03385>

Sékmingsi tinklai: ResNet

- **ResNet** (Residual Network) – autoriai Kaiming He et al. ILSVRC 2015 nugalėtojas.



He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
<https://arxiv.org/abs/1512.03385>

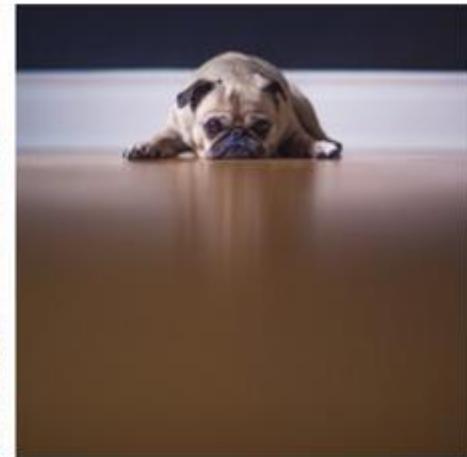
Sėkmingi tinklai: GoogleNet

- **GoogleNet/InceptionV1** – konkurso ILSVRC2014 nugalėtojas. Dabar jau yra **Inception V4** versija.
- Vienas iš būdų **pagerinti** giliųjų neuroninių tinklų veikimą – **padidinti** jų dydį.
- Tačiau tuomet reikia ir **didinti** mokymo duomenų aibę. Be to, stipriai **padidėja** skaičiavimo apimtys.
- **Google** komanda pasiūlė būdą pereiti iš pilnai jungaus į išretinto jungimo architektūrą.

Szegedy C., et al. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

Sėkmingi tinklai: GoogleNet

- **GoogleNet / Inception** sluoksnio idėja – padengti didesnį plotą, tačiau išlaikant vaizduose **gerą rezoliuciją** mažam kiekiui informacijos.

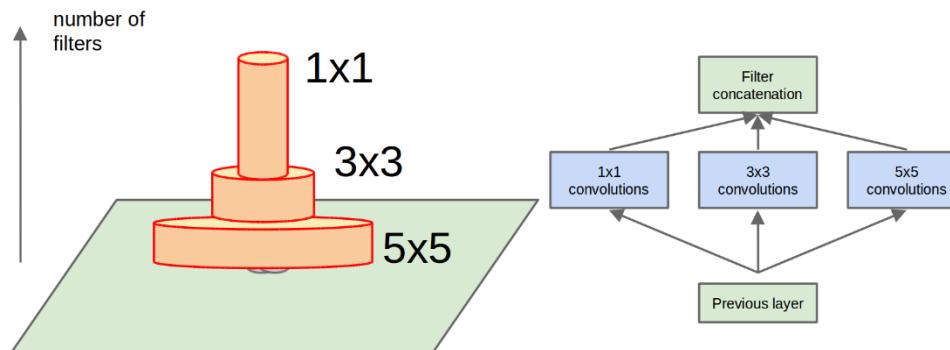


From left: A dog occupying most of the image, a dog occupying a part of it, and a dog occupying very little space (Images obtained from [Unsplash](#)).

<https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202>

Sėkmingi tinklai: GoogleNet

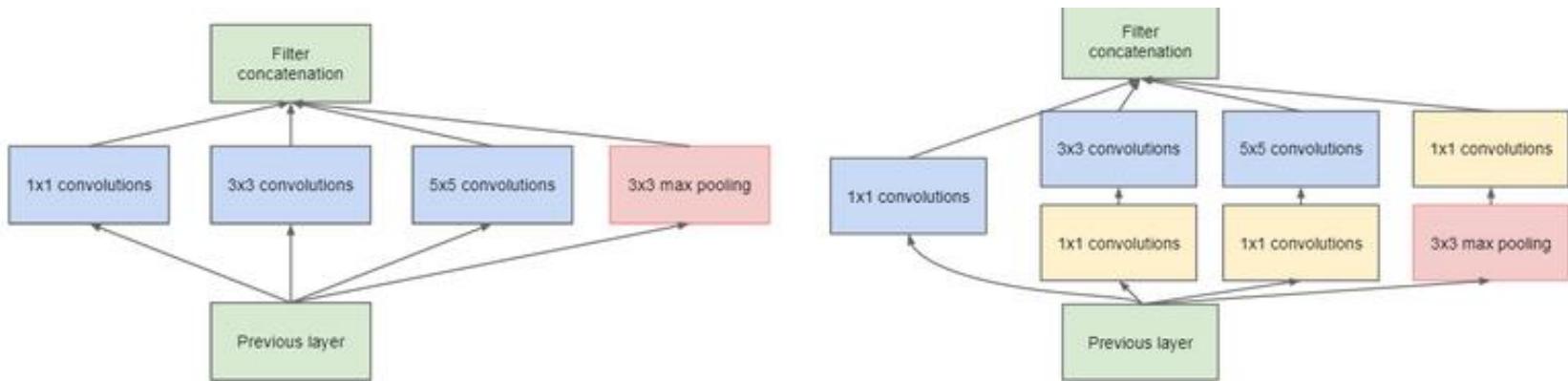
- **GoogleNet / Inception** sluoksnio idėja – padengti didesnį plotą, tačiau išlaikant vaizduose **gerą rezoliuciją** mažam kiekiui informacijos.
- Visų filtrų reikšmės nustatomos **mokymo** metu.



<https://leonardoaraujosantos.gitbooks.io/artificial-intelligence/content/googlenet.html>

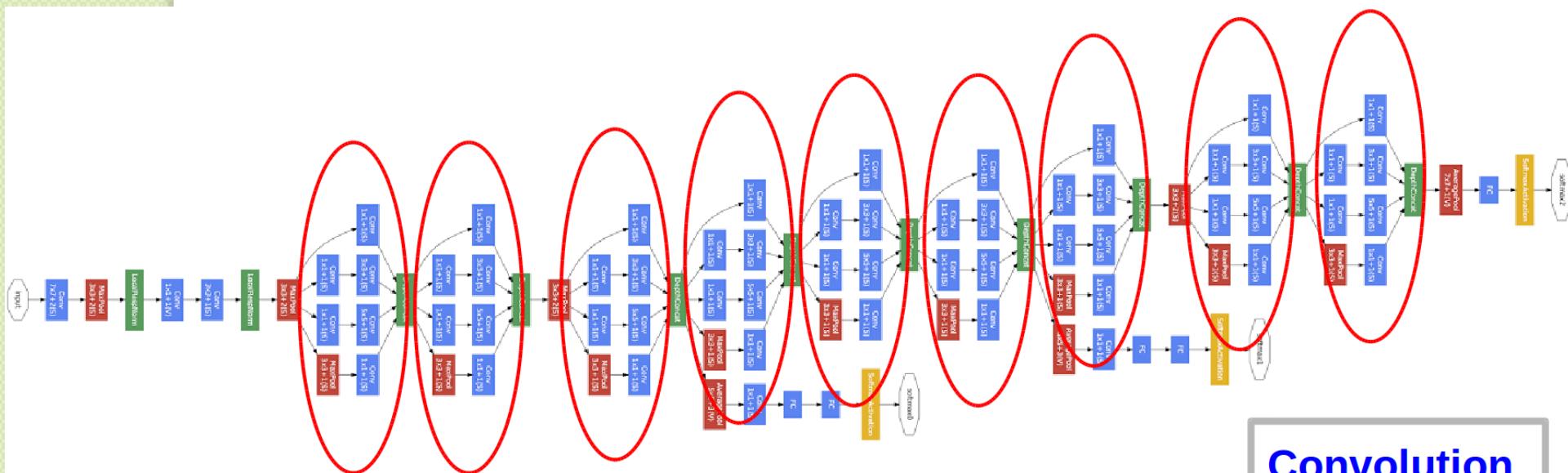
Sėkmingesni tinklai: GoogleNet

- **GoogleNet / Inception** modelyje dar gali būti pridėtas **dimensijos mažinimo** komponentas.



Sėkminges tinklai: GoogleNet

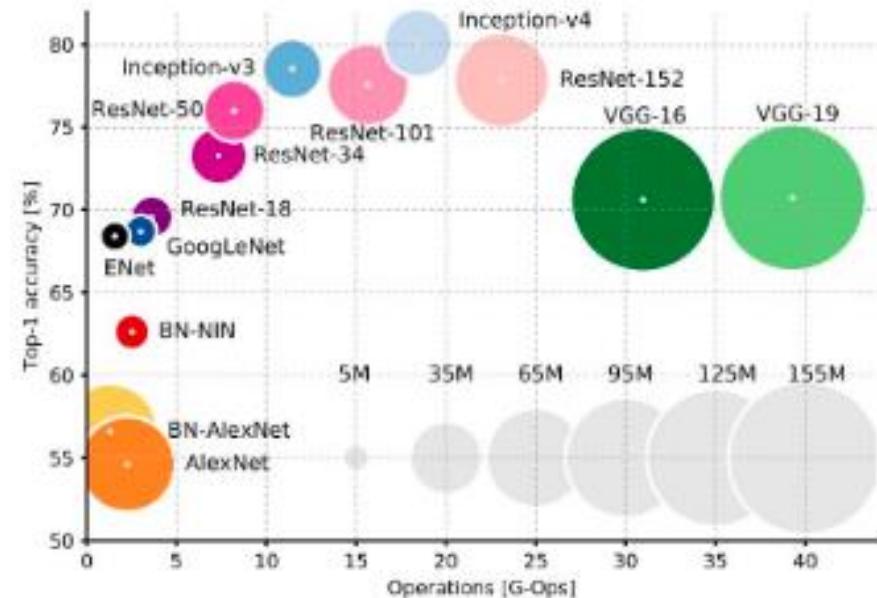
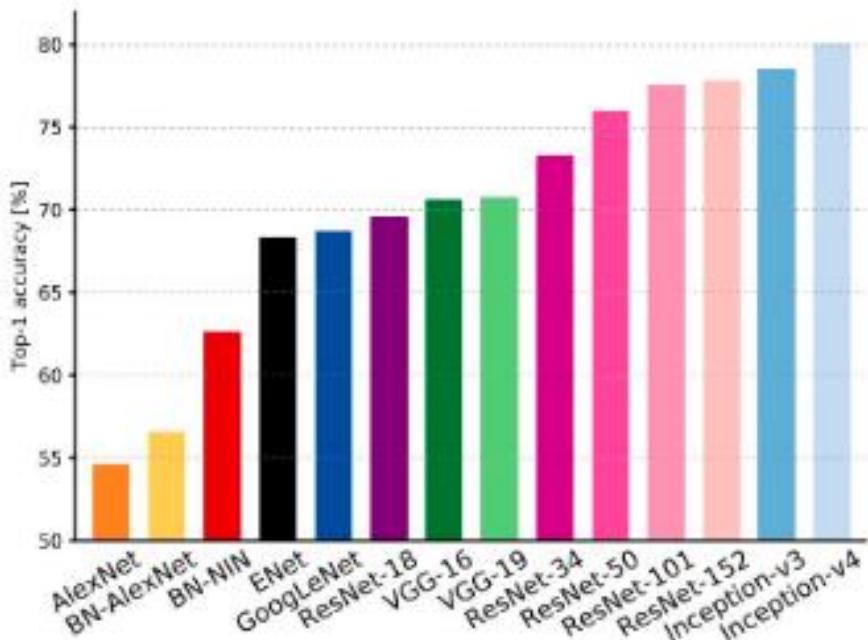
- **GoogleNet/InceptionV1** – konkurso ILSVRC2014 nugalėtojas. Dabar yra **Inception V4** versija.



Convolution
Pooling
Softmax
Concat/Normalize

<https://leonardoaraujosantos.gitbooks.io/artificial-intelligence/content/googlenet.html>

Gilieji neuroniniai tinklai



An Analysis of Deep Neural Network Models for Practical Applications, 2017.

<https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5>

Info

- <https://wiki.tum.de/display/lfdv/Convolutional+Neural+Network+Architectures>
- <http://cs.stanford.edu/people/karpathy/convnets/start.html>
- <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-014-0007-7>
- [https://brohrer.github.io/how convolutional neural networks work.html](https://brohrer.github.io/how_convolutional_neural_networks_work.html)
- <https://cs.stanford.edu/people/karpathy/convnets/demo/mnist.html>



Vilniaus universitetas
Matematikos ir informatikos fakultetas
Informatikos katedra

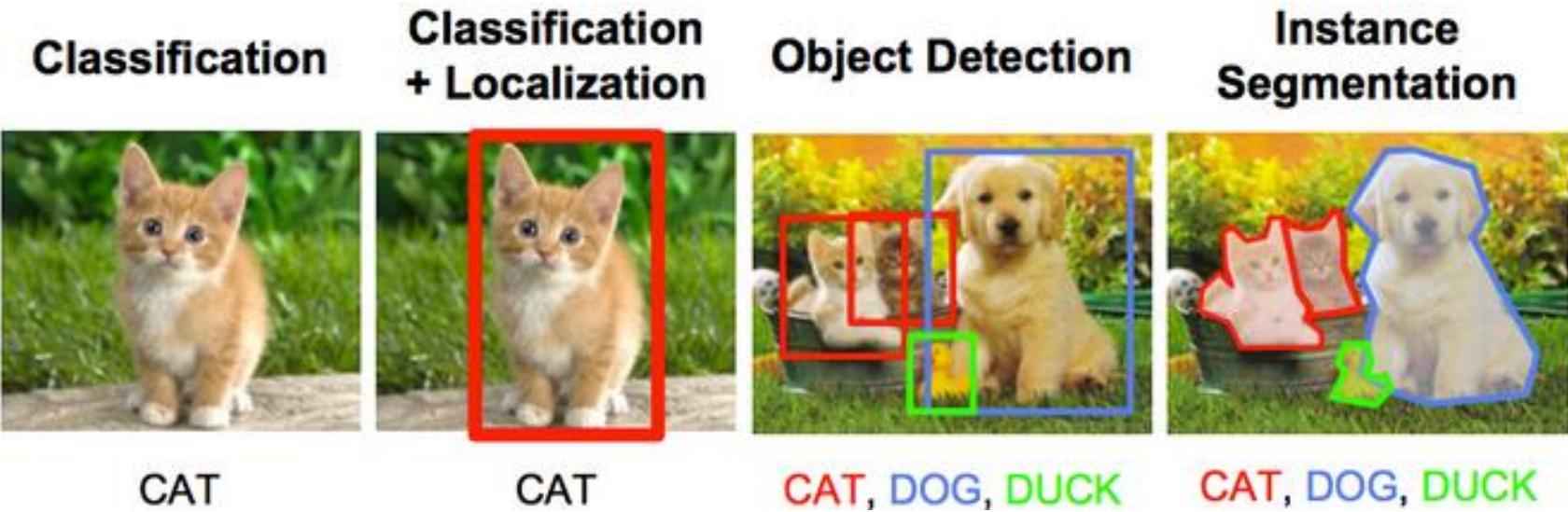


Objektų vaizduose atpažinimas ir vaizdų segmentavimas taikant konvoliucinius neuroninius tinklus

prof. dr. Olga Kurasova
Olga.Kurasova@mii.vu.lt

2018

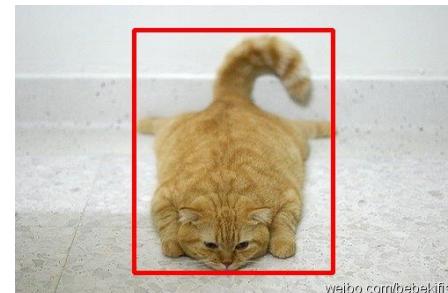
Kompiuterinės regos (*computer vision*) uždaviniai



<https://mathematica.stackexchange.com/questions/141598/object-detection-and-localization-using-neural-network>

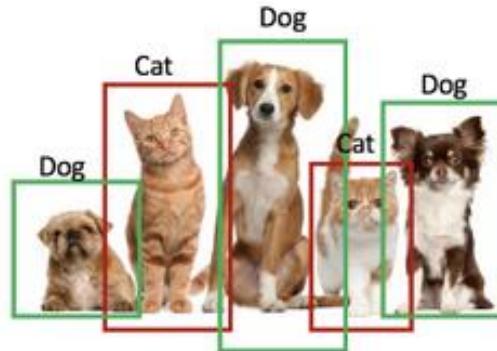
Objektų lokalizavimas

- Dažnai objektams lokalizuoti taikomo **iš anksto apmokyti tinklai (modeliai) (pre-trained)** (YOLO, R-CNN ir kiti).
- Todėl lokalizavimas **greitas** ne tik statiniams vaizdams, bet ir **video**.
- Tiek mokymo duomenyse, tiek gautuose tinklo išvestyse yra **ne klasė** (kaip yra klasifikavimo uždaviniuose), bet objektą apibrėžiantis **stačiakampus (bounding box)**



Objektų lokalizavimas ir klasifikavimas

- Dažnai reikia ne tik nustatyti objekto vietą, bet ir **identifikuoti**, koks tai objektas.
- Vaizde gali būti **keli** to paties tipo objektai.
- Rezultate gaunamas masyvas, kuriame nurodytos **stačiakampio parametrai** (dažnai centro koordinatės, ilgis ir plotis) ir **klasių tikimybės**.



<https://towardsdatascience.com/evolution-of-object-detection-and-localization-algorithms-e241021d8bad>

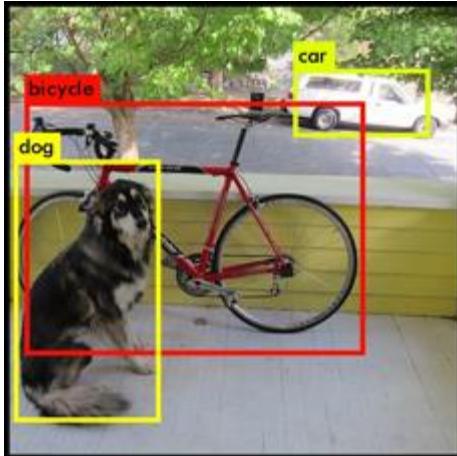
Objektų lokalizavimas ir klasifikavimas

- Naudojant iš **anksto apmokyta** tinklą (jis atpažįsta norimus objektus), per vaizdą **slenkamas langas** ir tikrinama, ar tinklas atpažįsta tame kažkokį objektą – *sliding window detection*
- Greitesnis yra **YOLO** algoritmas (You Only Look Once) – čia visas langas sudalinamas tinkleliu ir ieškomi objektais tinklelio langeliuose.

<https://www.coursera.org/lecture/deep-learning-in-computer-vision/sliding-windows-UlbVI>

<https://tryolabs.com/blog/2017/08/30/object-detection-an-overview-in-the-age-of-deep-learning/>

YOLO (You Only Look Once:)



Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

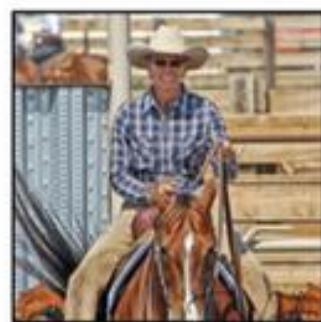
<https://pjreddie.com/darknet/yolo/>

<http://machinethink.net/blog/object-detection-with-yolo/>

<https://hackernoon.com/gentle-guide-on-how-yolo-object-localization-works-with-keras-part-1-aec99277f56f>

R-CNN

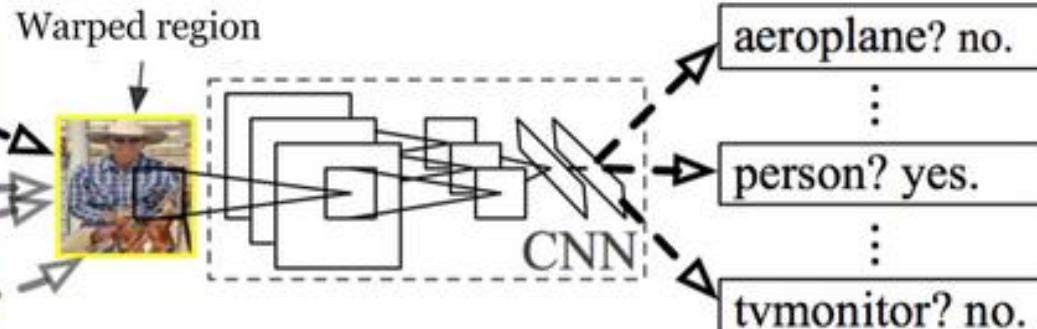
R-CNN ([Girshick et al., 2014](#)) is short for “Region-based Convolutional Neural Networks”. The main idea is composed of two steps. First, using [selective search](#), it identifies a manageable number of bounding-box object region candidates (“region of interest” or “RoI”). And then it extracts CNN features from each region independently for classification.



1. Input images



2. Extract region
proposals (~2k)



3. Compute CNN features

4. Classify regions

Fig. 1. The architecture of R-CNN. (Image source: [Girshick et al., 2014](#))

<https://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html>

Objektų lokalizavimas ir klasifikavimas: taikymai

- **Veidų** lokalizavimas
- Žmogaus (automobilio ir kt.) **lokalizavimas** ir **sekimas** (pvz., norint suskaičiuoti šiuos objektus)
- Žmogaus atliekamų **veiksmų identifikavimas**
- **Medicininių** vaizdų analizė (nors čia svarbiau segmentavimas, žr. tolesnėse skaidrėse)
- **Satelitinių** (arba **dronų** darytų) vaizdų analizė
- Kiti

Objektų lokalizavimas

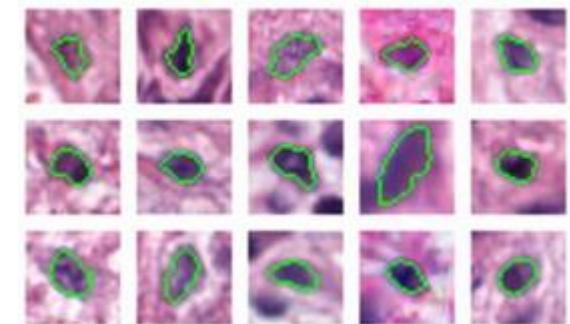
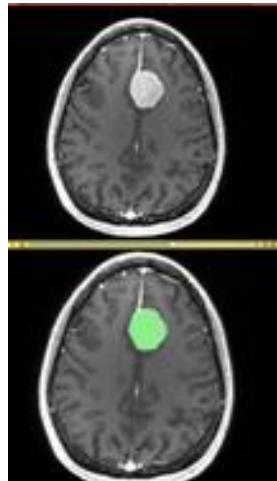
Įdomūs demo:

- <https://jeeliz.com/>
- <https://js.tensorflow.org/>
- https://trackingjs.com/examples/face_camera.html

Vaizdų segmentavimas

- **Vaizdų segmentavimas** (*image segmentation*) – tai procesas, kurio metu skaitmeninis vaizdas padalinamas į tam tikrus segmentus (dalies).
- Dažnai svarbu išskirti objektus ir jų ribas.
- Segmentavimas ypač svarbus **medicininiuose vaizduose**, kai siekiama nustatyti ligos pažeistus regionus.
- Vėliau (arba ir kartu) gali būti sprendžiamas **atpažinimo uždavinys**.

Vaizdų segmentavimo pavyzdžiai



Vaizdų segmentavimo taikymai

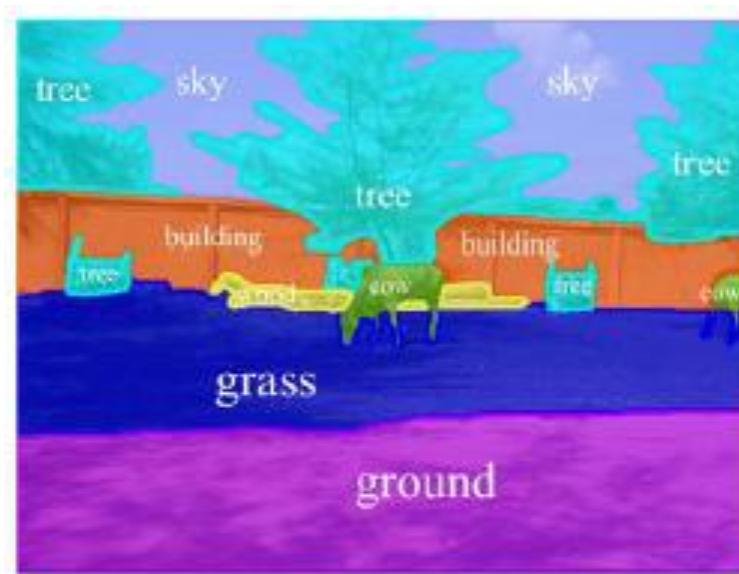
- **Medicininių vaizdų** analizė (kompiuterinė tomografija, magnetinis rezonansas)
- **Objektų** nustatymas
 - Pėsčiųjų nustatymas, veido nustatymas
- **Atpažinimo** uždaviniai
 - Veido atpažinimas, pirštų anspaudų atpažinimas, akies rainelės atpažinimas
- **Eismo kontrolės** sistemos (autonominiai automobiliai)
- **Mikroskopijos** uždaviniai

Vaizdų segmentavimo metodai

- Otsu metodas
- K-vidurkių metodas
- Histogramomis pagrįsti metodai
- Kraštinių (briaunų) nustatymo (*edge detection*) metodai
- SIFT metodas
- Įvairūs kiti vaizdų apdorojimo metodai
- **Dirbtiniai neuroniniai tinklai**

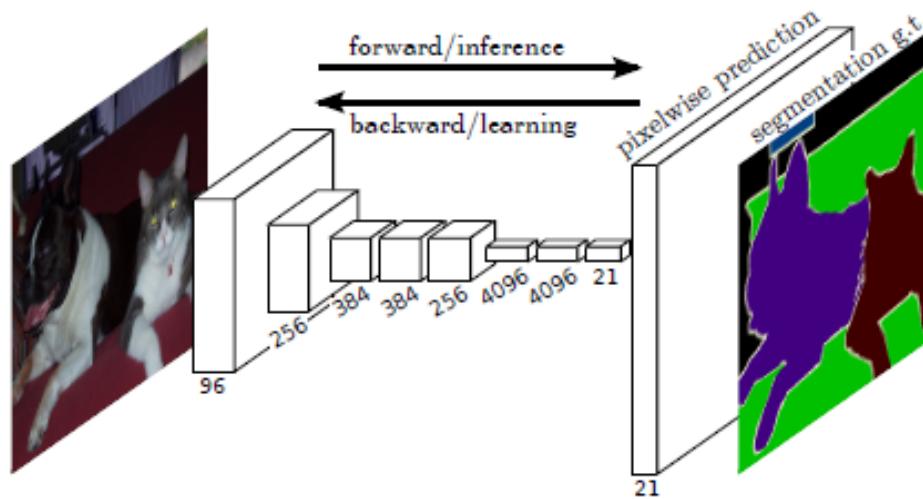
Semantinis segmentavimas

- Segmentavimo uždavinys, kai reikia ne tik nustatyti objektus, bet juos **priskirti ir klasės** (pvz., vanduo, automobilis, dangus, pastatas), vadinamas **semantinio segmentavimo uždaviniu**.



Pilnai konvoliucinis tinklas

- **Pilnai konvoliucinio tinklo** (*Fully Convolutional Networks, FCN*) tikslas – numatyti kiekvieno vaizdo pikselio reikšmę (*pixels-to-pixels*)



Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431-3440).

Pilnai konvoliucinis tinklas

- Pilnai konvoliucinis tinklas sudarytas iš:
 - Konvoliucinių sluoksnių
 - Sujungimo sluoksnių (*pooling*)
 - Atstatymo sluoksnių (*upsampling*)
- Čia nėra pilnai sujungto sluoksnio (*fully connected (dense) layers*).
- Išėjimo sluoksnio dydis sutampa su įvesties dydžiu.

Pilnai konvoliucinis tinklas

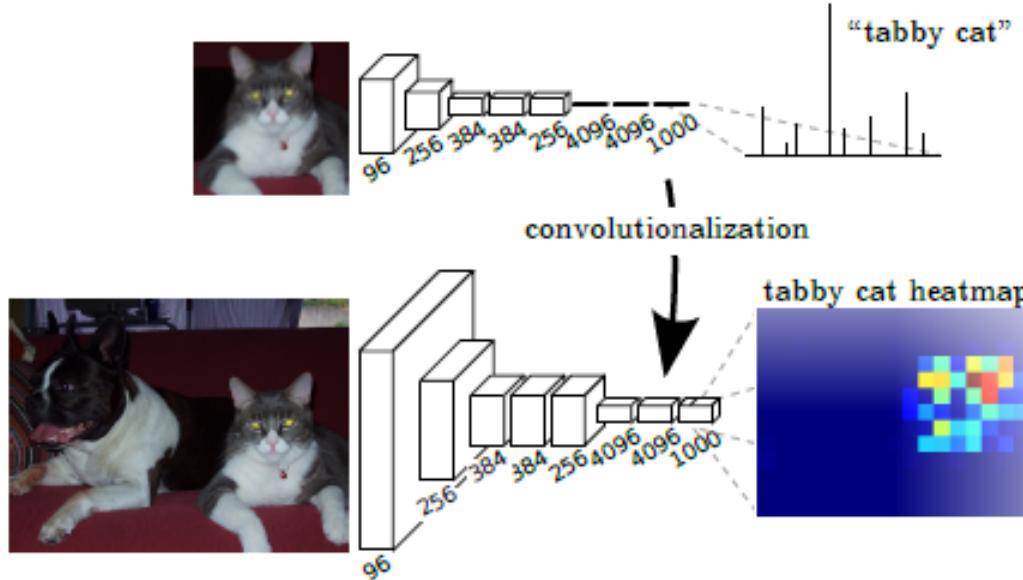
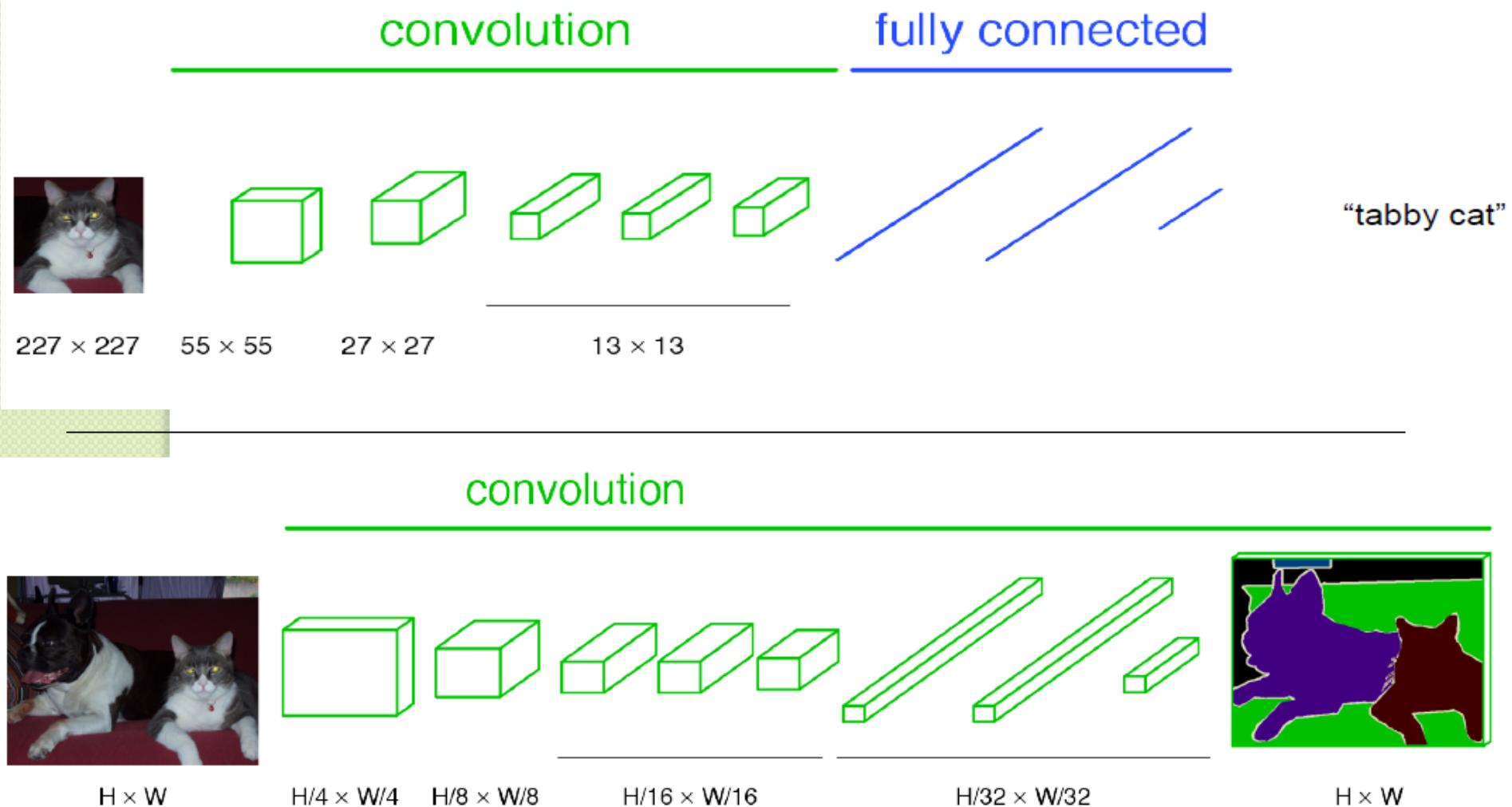


Figure 2. Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. Adding layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end dense learning.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431-3440).

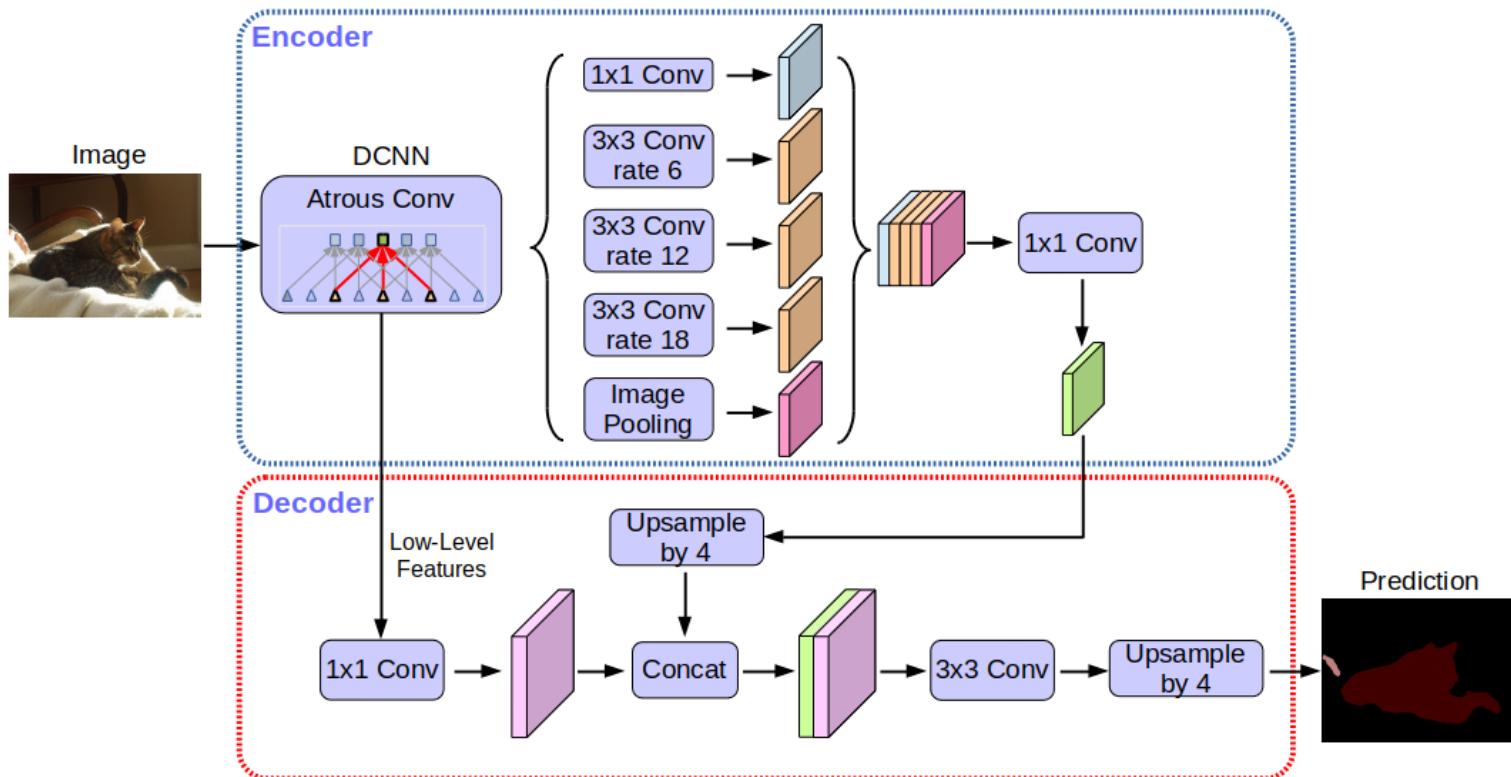
Pilnai konvoliucinis tinklas



https://leonardoaraujosantos.gitbooks.io/artificial-intelligence/content/image_segmentation.html

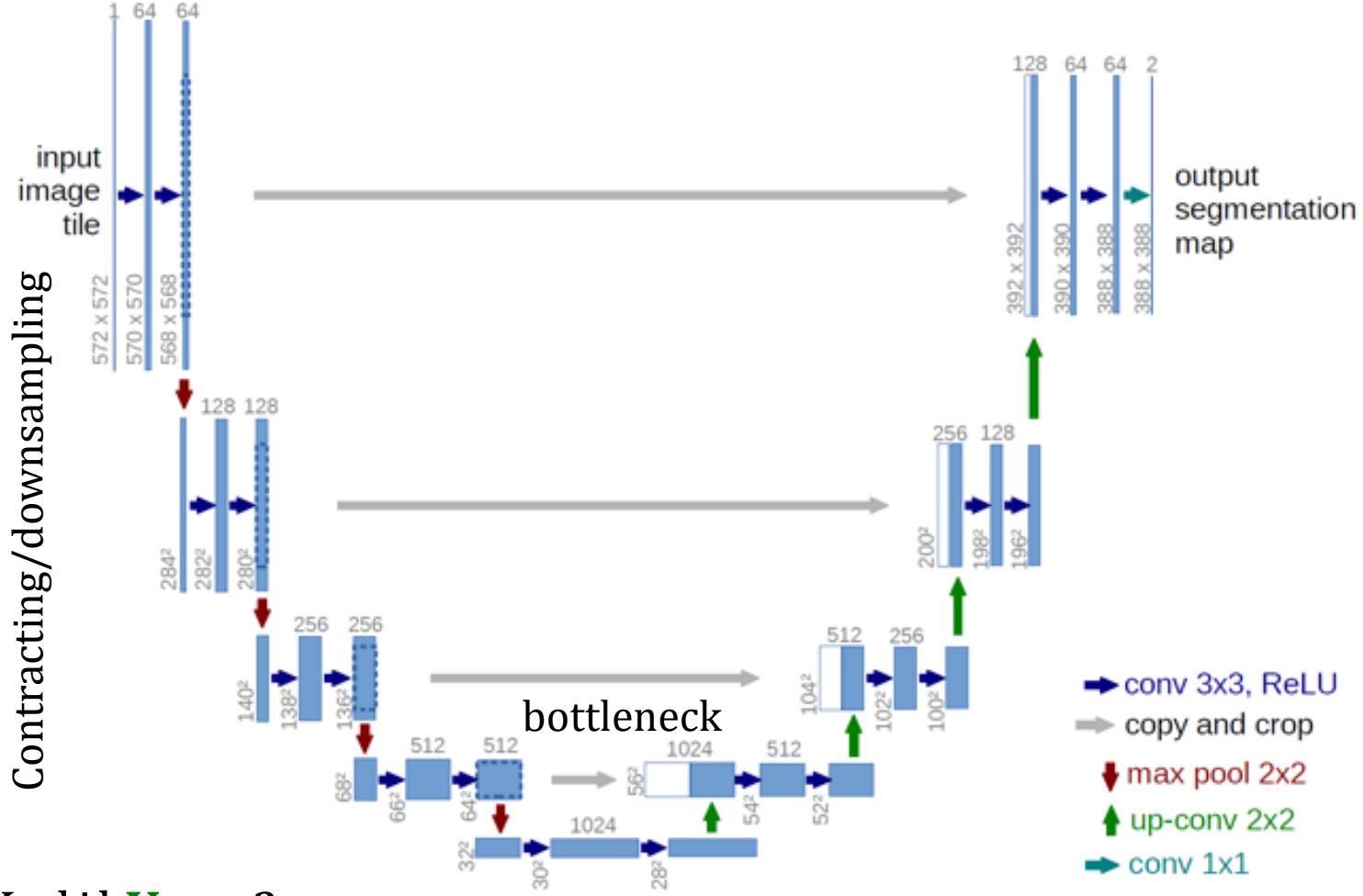
Enkoderis - dekoderis

- Kito tipo tinklai veikia enkoderis – dekoderis principu.



<https://ai.googleblog.com/2018/03/semantic-image-segmentation-with.html>

U-net



Kodėl **U**-net?

Dėl tinklo formos, primenančios **U** raide

U-net

- Tai **patobulintas** pilnai konvoliucinis tinklas.
- U-net – **simetrinis** tinklas.
- **Tikslas** – klasės nustatymas kiekvienam pikseliui.
- Patobulinimas – DeepUNet

Li, R., Liu, W., Yang, L., Sun, S., Hu, W., Zhang, F., & Li, W. (2018). DeepUNet: A deep fully convolutional network for pixel-level sea-land segmentation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*.

U-net

- Ronneberger O., Fischer P., Brox T. (2015) **U-Net: Convolutional Networks for Biomedical Image Segmentation**. In: Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. Lecture Notes in Computer Science, vol 9351. Springer, Cham. DOI:
https://doi.org/10.1007/978-3-319-24574-4_28
- The **full implementation** (based on Caffe) and the trained networks are available at
<http://lmb.informatik.uni-freiburg.de/people/ronneber/u-net>

U-net

Table 2. Segmentation results (IOU) on the ISBI cell tracking challenge 2015.

Name	PhC-U373	DIC-HeLa
IMCB-SG (2014)	0.2669	0.2935
KTH-SE (2014)	0.7953	0.4607
HOUS-US (2014)	0.5323	-
second-best 2015	0.83	0.46
u-net (2015)	0.9203	0.7756

Results from (Ronneberger et all., 2015)

Kokie duomenys reikalingi?

Vaizdai



Tikri poligonai



Nustatyti poligonai



Kokie duomenys reikalingi?

- Taigi, reikia turėti ne tik vaizdus, bet ir juos **anotuoti**, t. y. sudaryti objektų poligonus atitinkančias **kaukes** (*mask*). Tai atliekama naudojantis grafikos redagavimo programomis.
- Šios kaukės bus tinklo **norimos (trokštamos)** reikšmės.
- Jei tai **kelių klasių** uždavinys, dar reikia nurodyti kurią klasę atitinka konkreti kaukė.

Mokymo duomenys

- Jei turime **didelius vaizdus** (pvz., satelitinius, medicininius), kurių kiekviename daug ieškomų objektų, tinklo **mokymo duomenys** paruošiami taip:
 - I tinklo įvesti (input) pateikiamas ne visas vaizdas, tačiau **iškerpami jo fragmentai** (*patch sampling*), tokiu būdu stipriai padidinama mokymo aibė.
 - Iškirptiems objektams dar gali būti panaudojamos įvairios afininės transformacijos (pasukimai ir kt.), tai vadina **duomenų papildymu** (*augmentation*)

Tikslumo matai

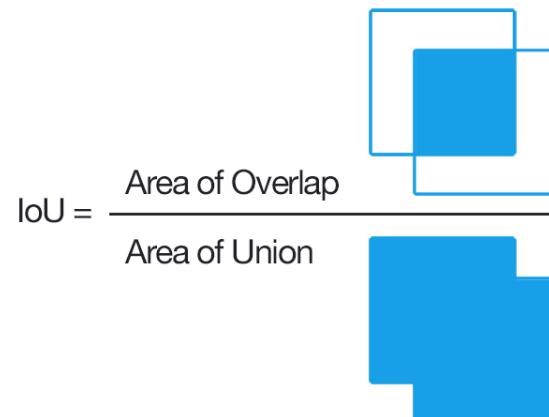
- **Tikslumas** pikselių lygmenyje

$$\text{acc}(P, GT) = \frac{\text{tiksliai nustatyti pikselių skaičius}}{\text{visų pikselių skaičius}}$$

- **Jaccard** metrika (*Intersection over Union*, IoU)

$$\text{jacc}(P(\text{class}), GT(\text{class})) = \frac{|P(\text{class}) \cap GT(\text{class})|}{|P(\text{class}) \cup GT(\text{class})|}$$

- čia P – nustatyta segmentavimo dalis,
GT – norima kaukė.



Dar apie U-net

Nors U-net sukurtas **medicininių** vaizdų segmentavimui, jis sėkmingai taikomas ir kitiems uždaviniams, pvz., **satelitinių** vaizdų segmentavimui



Apibendrinimas

- Liu, X., Deng, Z., & Yang, Y. (2018). Recent progress in semantic image segmentation. *Artificial Intelligence Review*, 1-18.
<https://link.springer.com/article/10.1007/s10462-018-9641-3>
- <https://www.jeremyjordan.me/semantic-segmentation/>



Vilniaus universitetas
Matematikos ir informatikos fakultetas
Informatikos katedra



Įvairūs gilioji neuroniniai tinklai

prof. dr. Olga Kurasova
Olga.Kurasova@mii.vu.lt

2018

Gilieji neuroniniai tinklai

- Nors pastaruoju metu **konvoliuciniai neuroniniai tinklai** yra vienas populiariausiu giliųjų neuroninių tinklų tipas, tačiau **jie nėra vieninteliai** plačiai taikomi įvairiems uždaviniams spręsti.

Gilieji neuroniniai tinklai

Verta nepamiršti **šiuų neuroninių tinklų**:

- Deep Belief Network
- Deep Boltzman Machines
- **Deep Recurrent Neural Network** (Long Short-Term Memory (LSTM) Networks)
- Deep Autoencoders

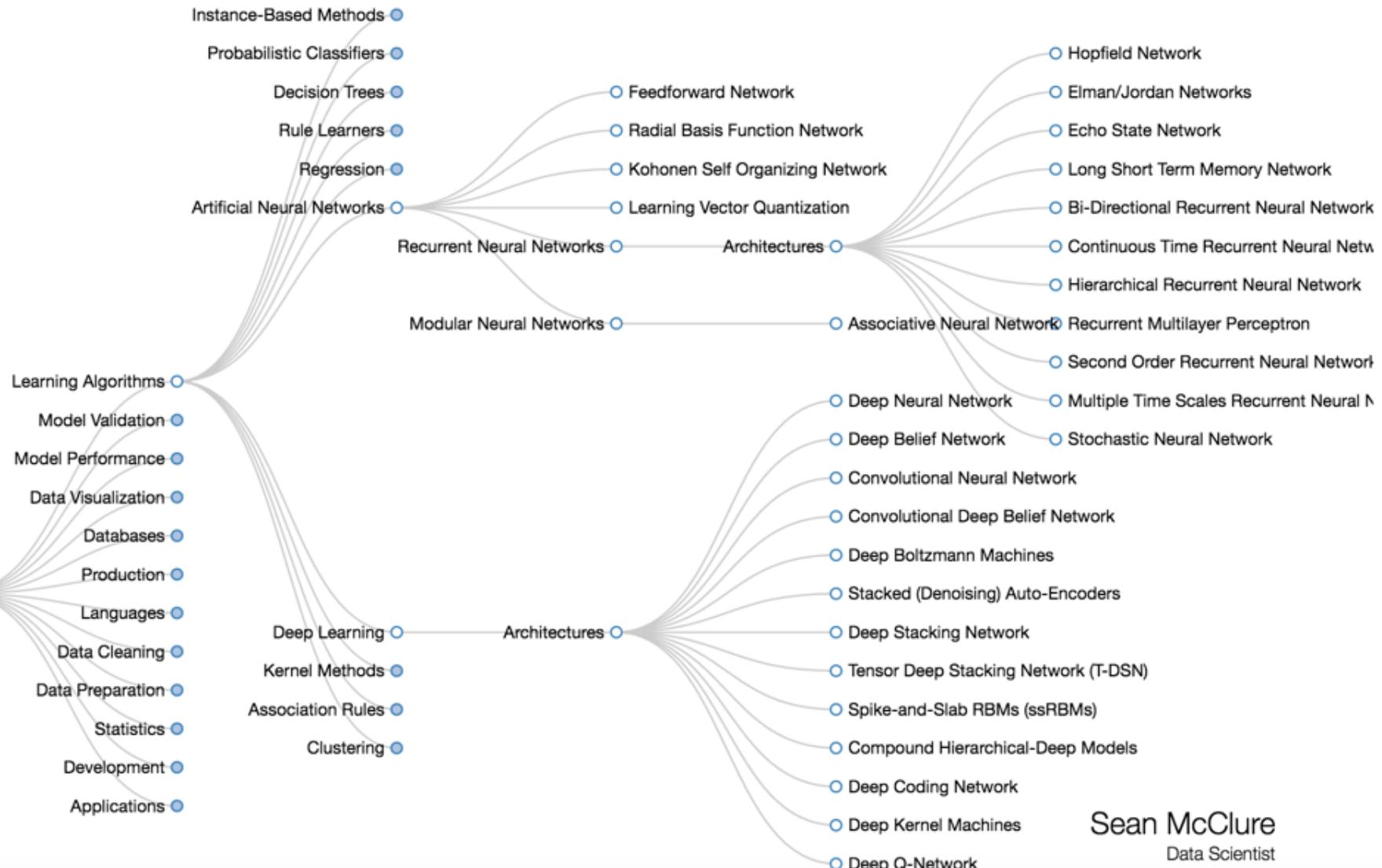
Data Science Ontology

[Tweet](#)[Follow @WorldOfDataSci](#)[Share](#)

Close

click to expand or collapse

click on terminal nodes to see wiki



Sean McClure
Data Scientist

Rekurentiniai neuroniniai tinklai

- **Rekurentiniai neuroniniai tinklai** (angl. *Recurrent Neural Networks, RNN*) – tai neuroniniai tinklai, kurių grafuose yra ciklų.
- Tai leidžia išgyvendinti **laike besikeičiančią veikseną**.
- RNN gali naudoti **jų vidinę atmintį** ir taip apdoroti į įvestis pateikiamas sekas.
- Tai leidžia spręsti tokius uždavinius, kaip **šnekos apdorojimas**.

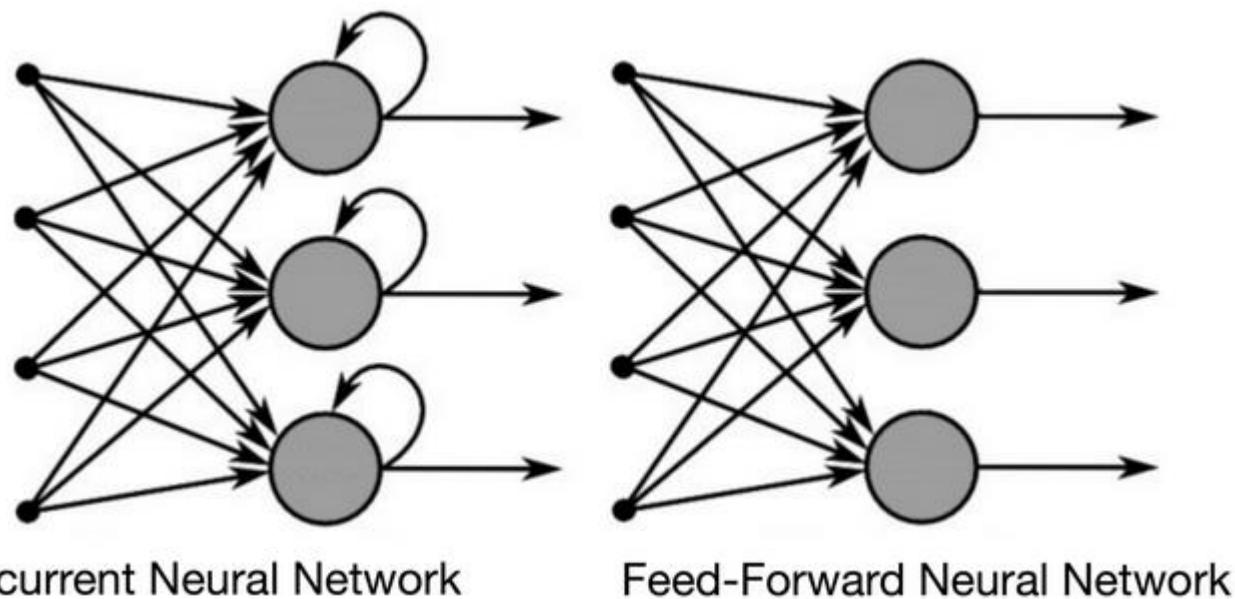
Rekurentiniai neuroniniai tinklai

- **Rekurentiniai neuroniniai tinklai** (*recurrent neural networks, RNN*) kuriame gana seniai (nuo 1980), tačiau pastaruoju metu, kaip ir visi giliojo mokymo neuroniniai tinklai, įgavo naują pagreitį.
- Tai tinklai, **turintys atmintį**, todėl gali puikiai tiki numatyti tai, kas bus ateityje.
- Iprastai jie turi trumpo laikotarpio atmintį (short-term memory), tačiau jie gali būti derinami su LSTM (long short-time memory).
- Jie naudojami **Apple Siri, Google Translate** ir kituose įrankiuose.

Rekurentinių neuroninių tinklų tipai (architektūros)

- **Ilgos trumpalaikės atminties tinklai** (angl. *long short-term memory (LSTM) networks*)
- Hopfieldo tinklai
- Elman/Jordan tinklai
- **Rekursyvūs neuroniniai tinklai** (angl. *recursive neural networks*)
- Rekurentiniai daugiasluoksniai perceptronai (angl. *recurrent multilayer perceptrons*)

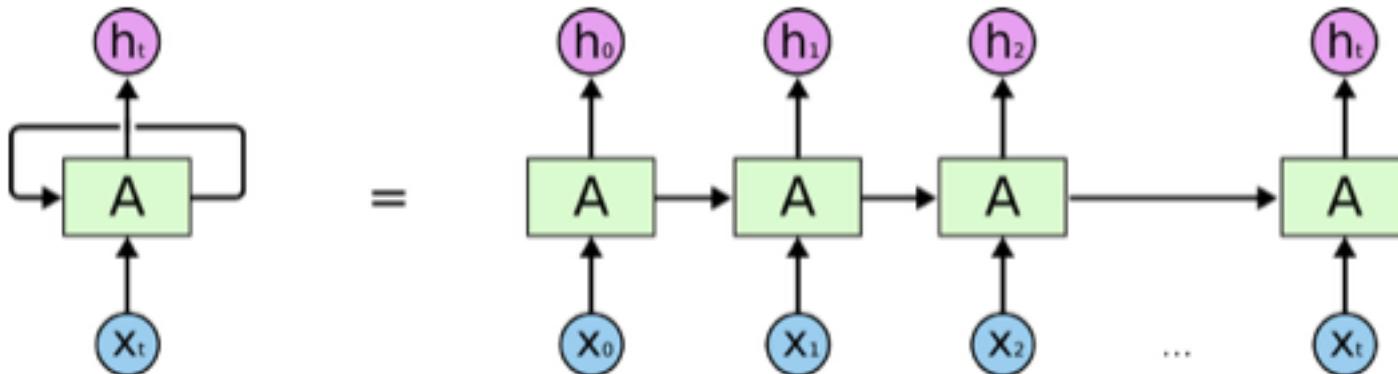
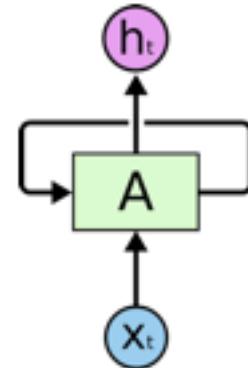
RNN vs FFNN



<https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5>

Rekurentiniai neuroniniai tinklai

- **Rekurentiniai tinklai** turi ciklus, kurie leidžia informacijai būti perduotai iš vieno tinklo žingsnio į kitą.



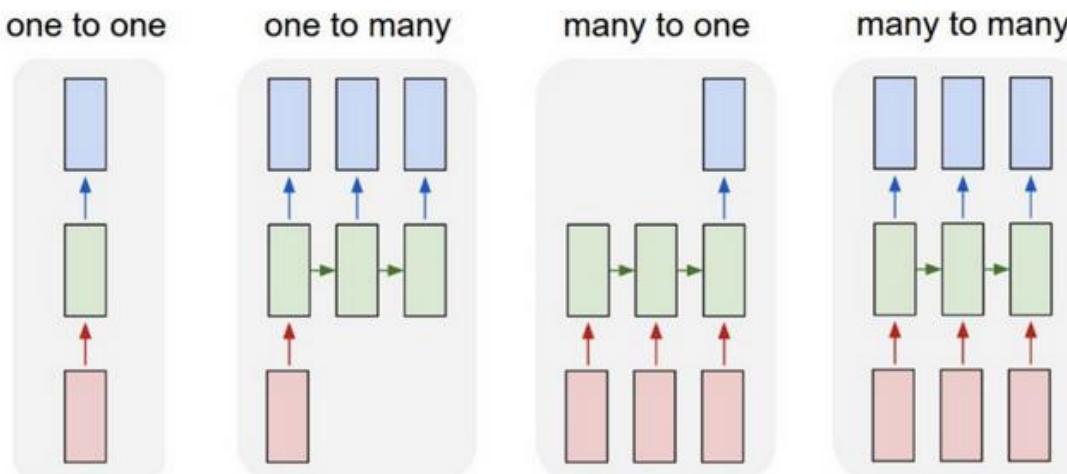
Pavyzdys

- „Imagine you have a normal feed-forward neural network and give it the word „neuron“ as an input and it processes the word **character by character**.“
- „At the time it reaches the character „r“, it has already **forgotten** about „n“, „e“ and „u“, which makes it almost impossible for this type of neural network to predict what character would come next.“

<https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5>

Rekurentiniai neuroniniai tinklai

- RNN mokymo metu pakeičiami svoriai ne tik **dabartinei įvesčiai**, bet ir **prieš tai buvusiai**.
- FFNN atvaizduoja **vieną įvestį i vieną išvestį**.
- RNN gali atvaizduoti vieną įvestį i daug išvesčių, daug i daug (vertimo atveju) arba daug i vieną (pvz., klasifikuojant balsą).



Ilgos trumpalaikės atminties (LSTM) tinklai

- Rekurentinių neuroninių tinklų praplėtimas yra **ilgos trumpalaikės atminties** (Long Short-Term Memory, LSTM) tinklai.
- Pavadinimas kilo nuo to, kad tai yra **trumpalaikės atminties, kuri gali tapti ilga periodą**, modelis.
- Modelis gerai tinka kai norima klasifikuoti, apdoroti ir prognozuoti **laiko eilutes** su nežinomo dydžio **vėlavimais** laike tarp svarbių įvykių.

Ilgos trumpalaikės atminties (LSTM) tinklai

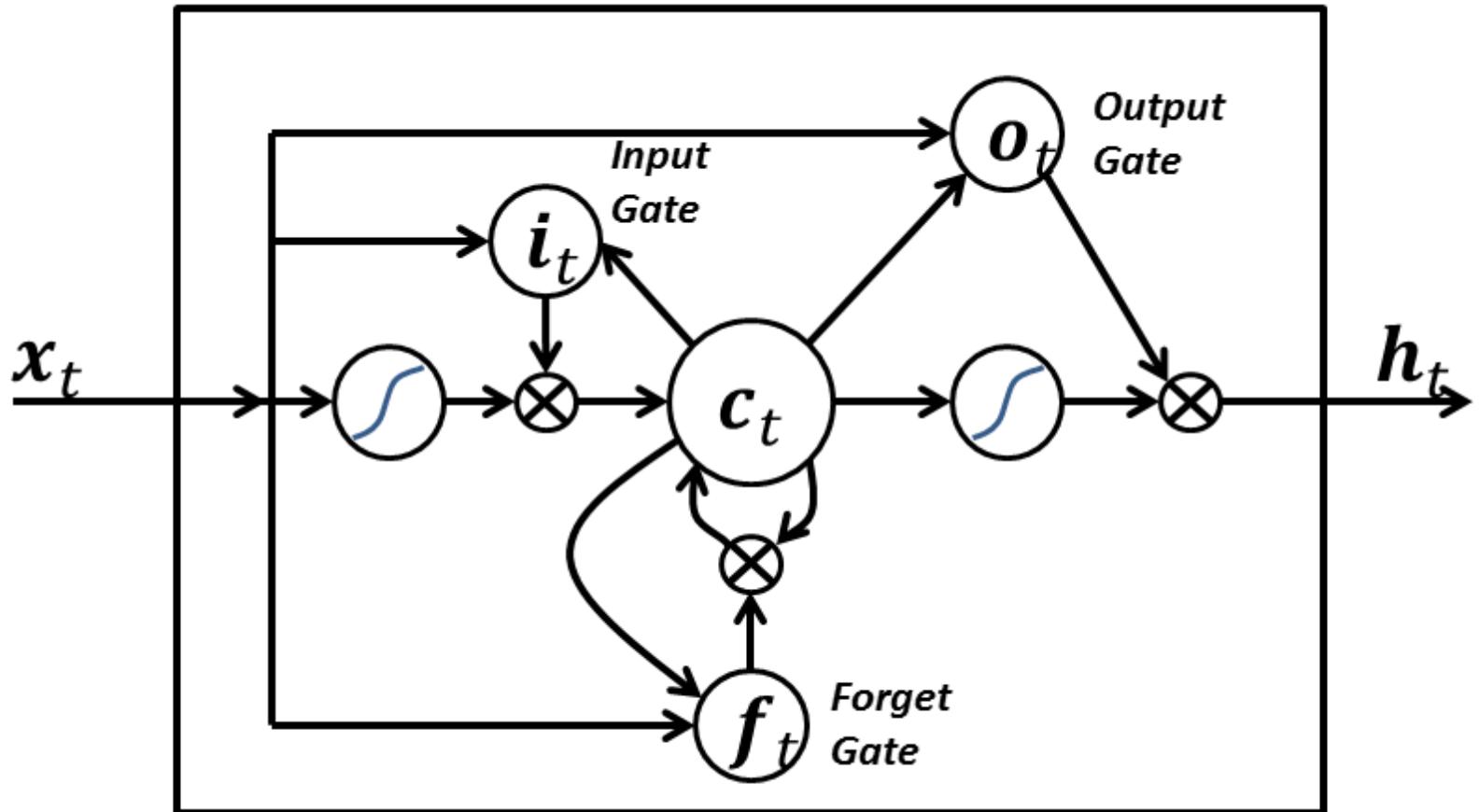
- Šių tinklų **pradžia** siekia **1997** metus.
- **2000** metais jie buvo **patobulinti**.
- Tokios IT srityje lyderiaujančios kompanijos, kaip **Google**, **Apple** ir **Microsoft** naudoja LSTM kaip pagrindinį komponentą daugelyje savo produktų.

Ilgos trumpalaikės atminties (LSTM) tinklai

Įvairūs taikymai:

- Robotų valdymas
- Laiko eilučių prognozavimas
- Šnekos atpažinimas
- Muzikos kūrimas
- Ranka rašomo teksto atpažinimas
- Žmogaus veiksmų atpažinimas
- Baltymų homologijos nustatymas
- Baltymų vietų prognozavimas

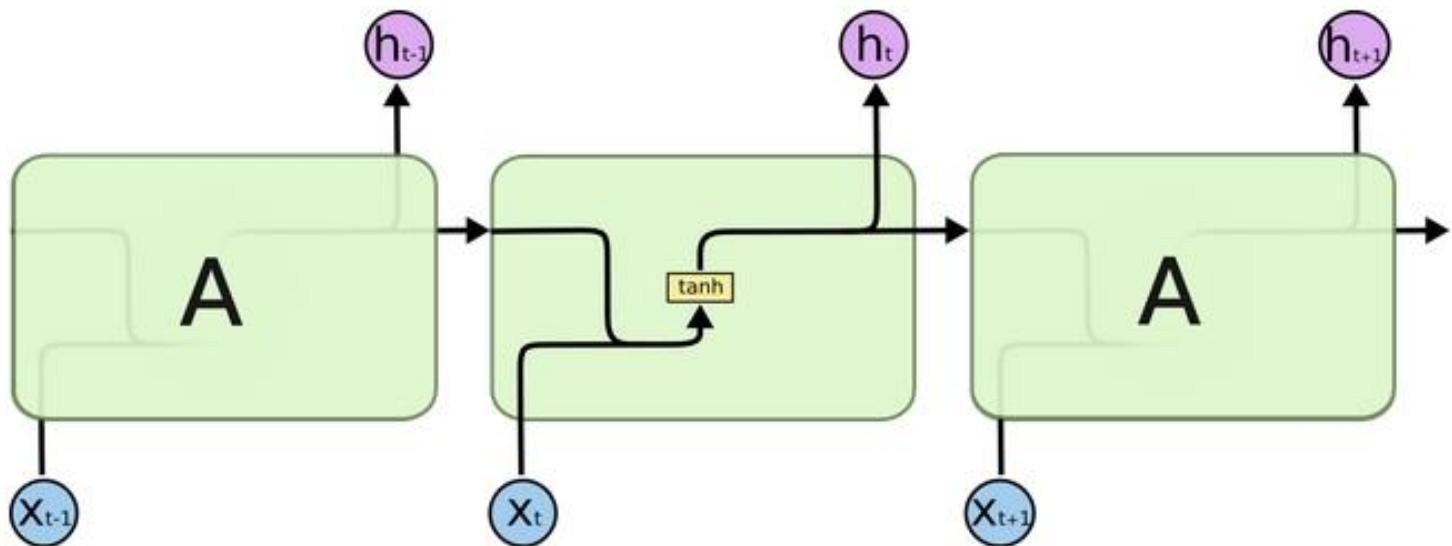
Ilgos trumpalaikės atminties (LSTM) tinklai



Long short-term memory (LSTM) block

Standartinis RNN vs LSTM

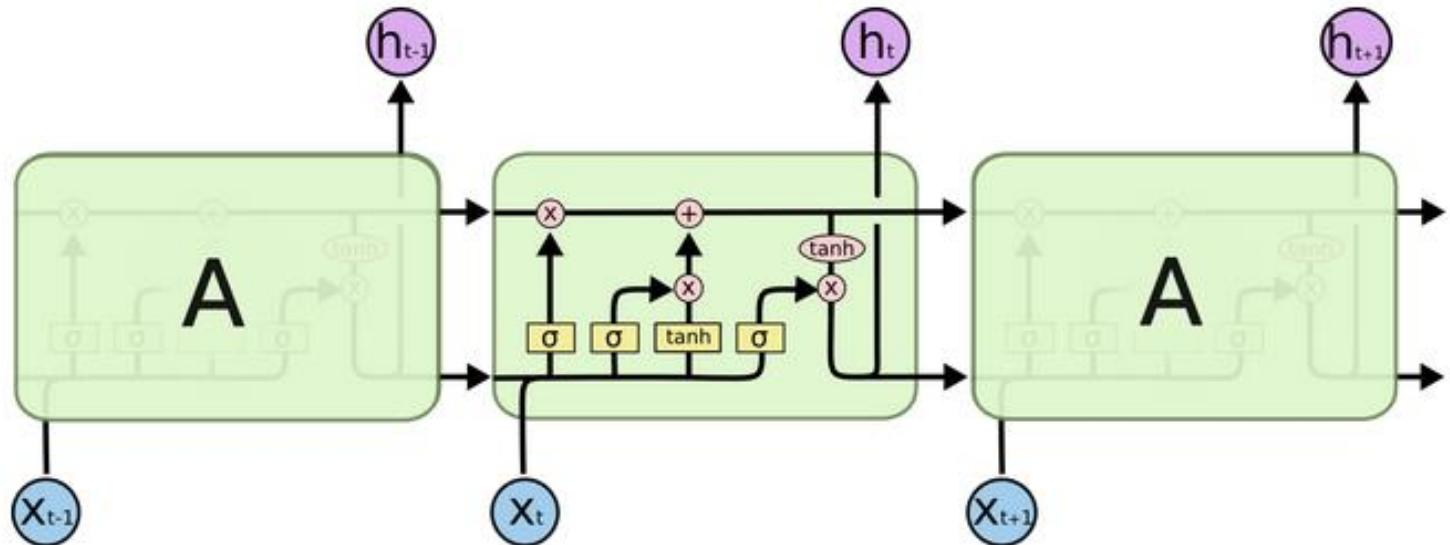
- „All RNNs have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a **very simple structure**, such as a single *tanh* layer.“



The repeating module in a standard RNN contains a single layer.

Standartinis RNN vs LSTM

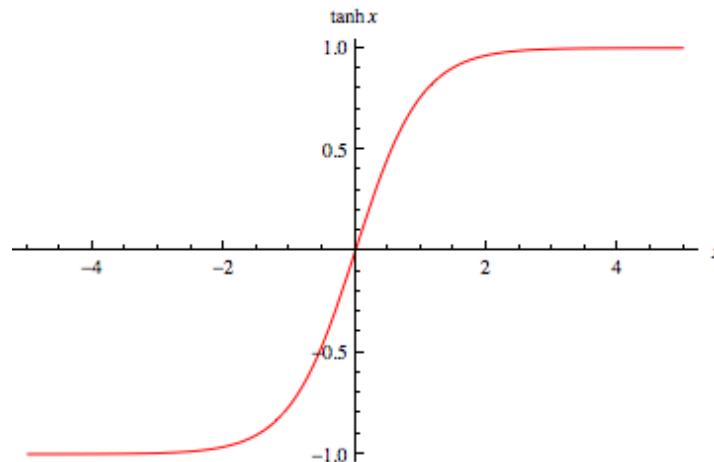
- „LSTMs also have this chain like structure, but the repeating module has a **different structure**. Instead of having a single neural network layer, there are **four**, interacting in a very special way.“



The repeating module in an LSTM contains four interacting layers.

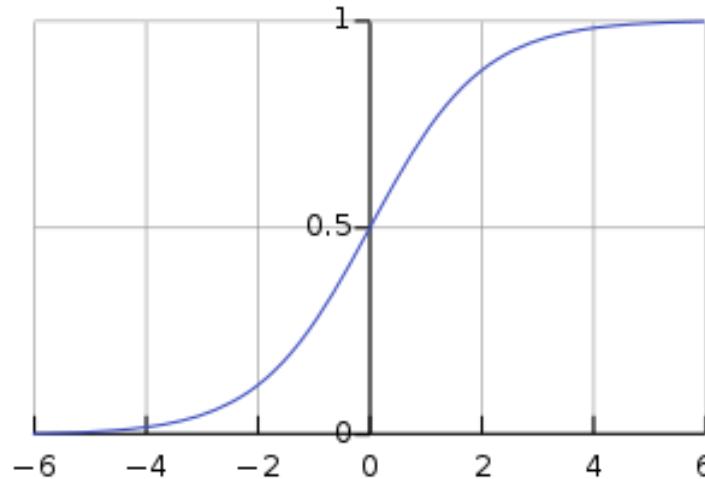
Kodėl \tanh ?

- Siekiant išvengti **gradiento nykimo problemos** (*vanishing gradient*), reikia naudoti funkciją, kurios antroji išvestinė gali išsilaiatyti nelygia nuliui gana ilgą intervalą.
- **Hiperbolinis tangentas** \tanh atitinka šį reikalavimą.



Kodėl sigmoidinė funkcija?

- **Sigmoidinė funkcija** gali išgyti reikšmes 0, 1, kurios gali būti naudojamos „pamiršti“ arba „prisiminti“ informaciją.



Ilgos trumpalaikės atminties (LSTM) tinklai

- **Google** naudoja LSTM šnekai atpažinti telefone, išmaniajame asistente ***Allo*** (<https://allo.google.com>) ir ***Google Translate***.
- **Apple** naudoja LSTM „**Quicktype**“ funkcijai įgyvendinti telefonuose iPhone.
- **Amazon** naudoja LSTM kaip ***Amazon Alexa***.

Amazon Alexa

Alexa is Amazon's cloud-based voice service available on tens of millions of devices from Amazon and third-party device manufacturers. With Alexa, you can build natural voice experiences that offer customers a more intuitive way to interact with the technology they use every day. Our collection of tools, APIs, reference solutions, and documentation make it easy for anyone to build with Alexa.

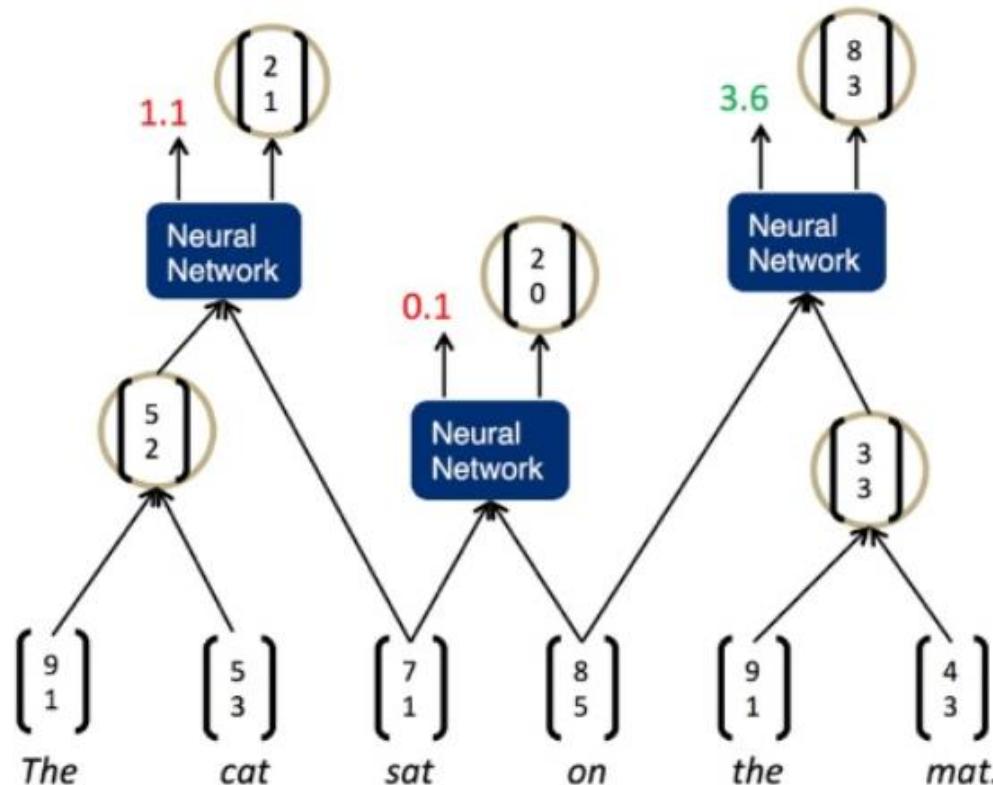
Iš <https://developer.amazon.com/alexa>

Rekursyvūs neuroniniai tinklai

- **Rekursyvūs neuroniniai tinklai** (angl. *recursive neural networks, RvNN*) – tai giliųjų neuroninių tinklų tipas, kuriuose rekursyviai dalijamasi svoriais.
- RvNN sėkmingai taikomas **natūralios šnekos apdorojime**, taip pat uždaviniuose, kuriose būtina apdoroti sekas.

Rekursyvūs neuroniniai tinklai

- Jie turi **medžio struktūrą**, kurioje kiekviename mazge yra neuroninis tinklas.



Iš <https://www.slideshare.net/jiessiecao/parsing-natural-scenes-and-natural-language-with-recursive-neural-networks>



Vilnius universitetas
Matematikos ir informatikos fakultetas
Informatikos katedra



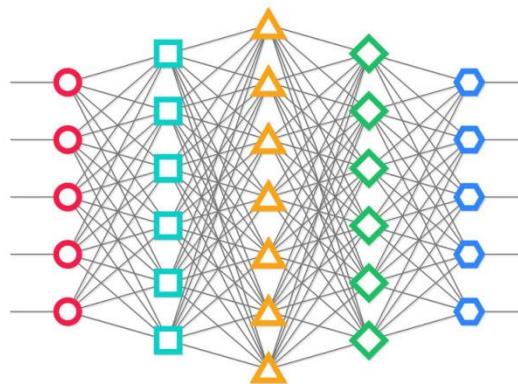
Dirbtiniai neuroniniai tinklai (programinė įranga)

prof. dr. Olga Kurasova
Olga.Kurasova@mii.vu.lt

2018

Neuroninių tinklų programinė įranga

- Pradėjus kurti dirbtinius neuroninius tinklus, atsirado poreikis kurti ir **programinę įranga**, kurioje jei būtu išgyvendinti.
- Atsiradus naujiems neuroninių tinklų tipams ir architektūroms, programinė įranga turi būti nuolat **atnaujinama**.



Neuroniniai tinklai įvairiose sistemose

Neuroniniai tinklai įgyvendinti įvairiose duomenų tyrybos sistemose:

- **WEKA** – daugiasluoksnis perceptronas
- **Orange** – SOM tinklai
- **R** paketas
 - *library(neuralnet)* – daugiasluoksnis perceptronas
 - *library(nnet)* – daugiasluoksnis perceptronas
- **Matlab** – Neural Network Toolbox

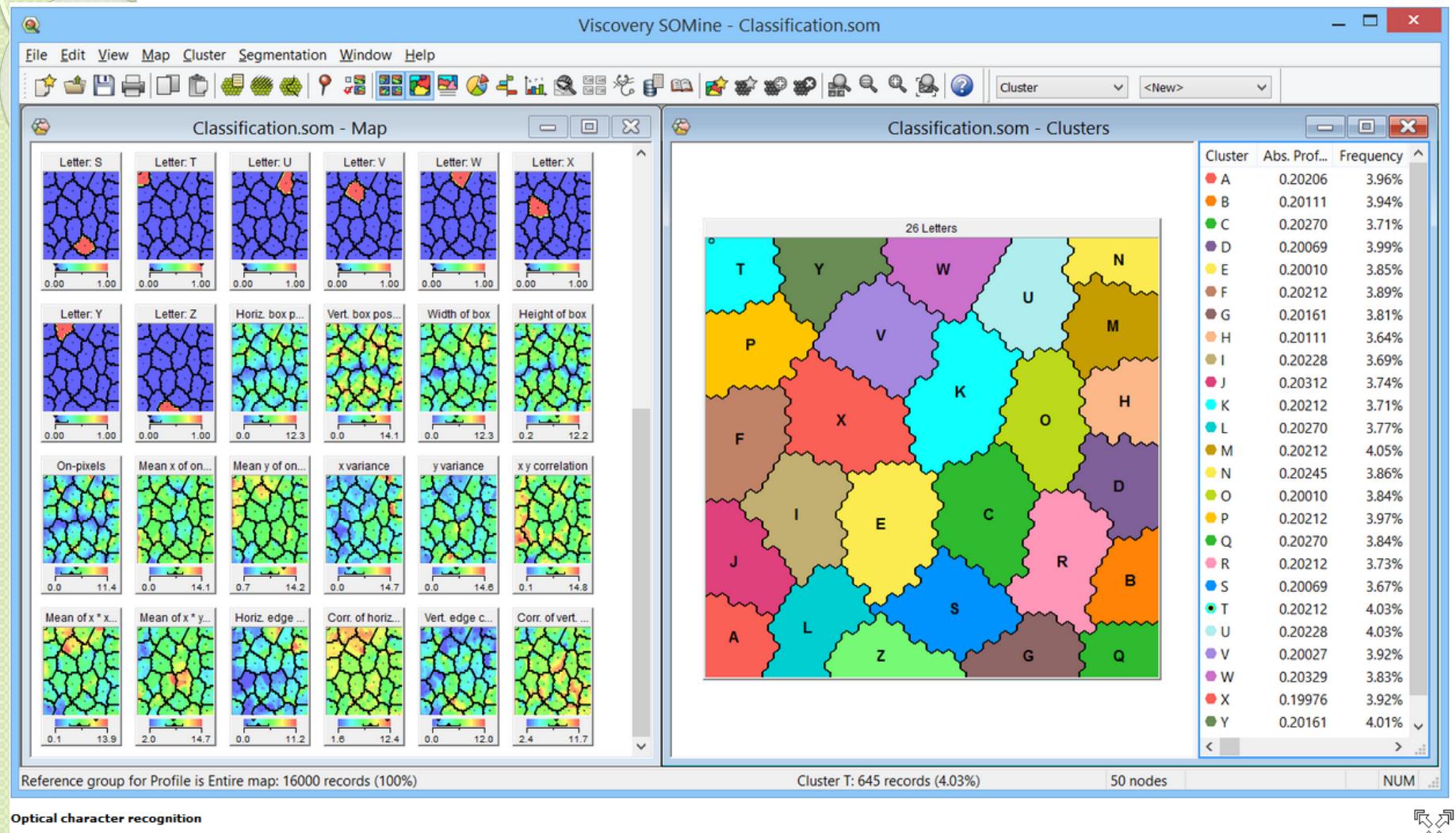


Viscovery SOMine



- **Viscovery SOMine** is a workflow-oriented software suite based on **self-organizing maps** (SOM) and multivariate statistics for explorative data mining and predictive modeling.
- Main functions and features:
 - **Creation** of self-organizing map models using predefined schedules
 - Interactive **SOM visualization** and exploration
 - Visual cluster analysis with integrated visualization of **cluster boundaries** and inner structures
 - **Statistical functions**, such as descriptive statistics, histograms, correlations, PCA, and scatter plots

Viscovery SOMine



From: <https://www.viscovery.net/somine/>

Gilieji neuroniniai tinklai R pakete

Table 1. List of available deep learning methods across the R packages.

PACKAGE	AVAILABLE ARCHITECTURES OF NEURAL NETWORKS
MXNetR	Feed-forward neural network, convolutional neural network (CNN)
darch	Restricted Boltzmann machine, deep belief network
deepnet	Feed-forward neural network, restricted Boltzmann machine, deep belief network, stacked autoencoders
H2O	Feed-forward neural network, deep autoencoders
deepr	Simplify some functions from H2O and deepnet packages



About TensorFlow

TensorFlow™ is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

Iš: <https://www.tensorflow.org/>



TenzorFlow

- Version 1.0.0 was released on **February 11, 2017**.
- **TensorFlow** is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.
- In May 2016, Google announced its **Tensor processing unit** (TPU), an ASIC built specifically for machine learning and tailored for TensorFlow.
- In May 2017, Google announced the **second-generation**, as well as the availability of the TPUs in Google Compute Engine. The second-generation TPUs deliver up to 180 teraflops of performance, and when organized into clusters of 64 TPUs, provide up to 11.5 petaflops.
- In February 2018, Google announced that they were making TPUs available in beta on the **Google Cloud Platform**.



TensorBoard

- **TensorBoard** – tai mokymo proceso vizualizavimo įrankis.
- Mokymo metu programiškai **fiksuojamos tikslumo reikšmės**, kurios atvaizduojamos TensorBoard įrankyje.

The computations you'll use TensorFlow for - like training a massive deep neural network - can be complex and confusing. To make it easier to understand, debug, and optimize TensorFlow programs, we've included a suite of visualization tools called TensorBoard. You can use TensorBoard to visualize your TensorFlow graph, plot quantitative metrics about the execution of your graph, and show additional data like images that pass through it. When TensorBoard is fully configured, it looks like this:



TensorBoard

TensorBoard SCALARS IMAGES GRAPHS > INACTIVE

Show data download links
 Ignore outliers in chart scaling

Tooltip sorting method: default ▾

Smoothing:

Horizontal Axis: STEP RELATIVE WALL

Runs: Write a regex to filter runs

train
 eval

TOGGLE ALL RUNS

/tmp/mnist-logs

accuracy cross entropy

cross entropy

run to download CSV JSON

Name	Smoothed	Value	Step	Time	Relative
eval	0.02591	0.02550	170.0	Mon Sep 12, 15:40:41	8s
train	0.02851	0.03362	166.0	Mon Sep 12, 15:40:40	7s

mean

Keras



- **Keras** is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.
- If you want to quickly build and test a neural network with minimal lines of code, choose **Keras**. With Keras, you can build **simple or very complex neural networks** within a few minutes.

Deep Learning frameworks

Deep Learning frameworks operate at 2 levels of abstraction:

- Lower Level: This is where frameworks like Tensorflow, MXNet, Theano, and PyTorch sit. This is the level where mathematical operations like Generalized Matrix-Matrix multiplication and Neural Network primitives like Convolutional operations are implemented.
- Higher Level: This is where frameworks like Keras sit. At this Level, the lower level primitives are used to implement Neural Network abstraction like Layers and models. Generally, at this level other helpful APIs like model saving and model training are also implemented.

<https://www.quora.com/Is-Keras-better-than-Tensorflow-for-deep-learning>

Neuroniniai tinklai, TensorFlow ir Keras

- **Konvoliuciniai** neuroniniai tinklai
- **Rekurentiniai** neuroniniai tinklai
(LSTM – Long short-term memory)

Theano

- **Theano** is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.

Theano 1.0.0 (15th of November, 2017)

This is a final release of Theano, version [1.0.0](#), with a lot of new features, interface changes, improvements and bug fixes.

Microsoft Cognitive Toolkit (CNTK)

The **Microsoft Cognitive Toolkit** (open source)—previously known as **CNTK**—empowers you to harness the **intelligence** within massive datasets through **deep learning** by providing uncompromised scaling, speed, and accuracy with commercial-grade quality and compatibility with the programming languages and algorithms you already use.



H2O



- **H2O** is an open source, in-memory, distributed, fast, and scalable machine learning and predictive analytics platform that allows you to build **machine learning** models on big data and provides easy productionalization of those models in an enterprise environment.
- **H2O**'s core code is written in **Java**.
- **H2O's REST API** allows access to all the capabilities of H2O from an external program or script via **JSON** over **HTTP**.

From: <http://docs.h2o.ai/h2o/latest-stable/h2o-docs/welcome.html>

H2O and Neural Networks



- H2O's Deep Learning is based on a multi-layer **feedforward artificial neural network** that is trained with stochastic gradient descent using back-propagation.
- The network can contain a **large number of hidden layers** consisting of neurons with tanh, rectifier, and maxout activation functions.
- **Advanced features** such as adaptive learning rate, rate annealing, momentum training, dropout, L1 or L2 regularization, checkpointing, and grid search enable high predictive accuracy.
- Each compute node trains a copy of the global model parameters on **its local data** with multi-threading (asynchronously) and contributes periodically to the global model via model averaging across the network.

H2O and Neural Networks



- The **H2O Deep Water project** supports **CNNs** and **RNNs** through third-party integrations of other deep learning libraries such as TensorFlow, Caffe and MXNet.

Caffe

- **Expression:** models and optimizations are defined as plaintext schemas instead of code.
- **Speed:** for research and industry alike speed is crucial for state-of-the-art models and massive data.
- **Modularity:** new tasks and settings require flexibility and extension.
- **Openness:** scientific and applied progress call for common code, reference models, and reproducibility.
- **Community:** academic research, startup prototypes, and industrial applications all share strength by joint discussion and development in a BSD-2 project.
- Web image classification demo
<http://demo.caffe.berkeleyvision.org/>

Apache MXNet

- **MXNet** – a flexible and efficient library for deep learning.
- Based on the the Gluon API specification, the new **Gluon** library in **Apache MXNet** provides a clear, concise, and simple API for deep learning. It makes it easy to prototype, build, and train deep learning models without sacrificing training speed.





Torch

Torch is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation.

A summary of core features:

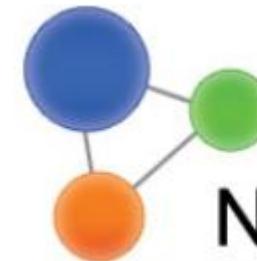
- a powerful N-dimensional array
- lots of routines for indexing, slicing, transposing, ...
- amazing interface to C, via LuaJIT
- linear algebra routines
- neural network, and energy-based models
- numeric optimization routines
- Fast and efficient GPU support
- Embeddable, with ports to iOS and Android backends

From: <http://torch.ch/>

ANN software (more)



neuraldesigner



Neuroph

Java Neural Network Framework



simbrain



Ką rinktis?

- Ar norima **paprastos** ir **vartotojui draugiškos**?
 - *desktop application*
- Ar bus vykdomi **intensyvūs skaičiavimai**?
 - GPU ar debesija pagristi sprendimai
- Ar norima patiems **programuoti**?
 - Pageidautina programavimo kalba
- Ar norima vykti **išsamius tyrimus**?
 - Patogus būdas modifikuoti, automatizuoti.

Jupyter notebook



- The **Jupyter Notebook** is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.
- Uses **include**: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, **machine learning**, and much more.

<http://jupyter.org/index.html>

Jupyter notebook



Galima vykdyti kodą iš karto **web aplinkoje**.

In [2]:

```
# Load the data
train = pd.read_csv("../input/train.csv")
test = pd.read_csv("../input/test.csv")
```

In [3]:

```
Y_train = train["label"]

# Drop 'label' column
X_train = train.drop(labels = ["label"],axis = 1)
```



Vilniaus universitetas
Matematikos ir informatikos fakultetas
Informatikos katedra



Kapsuliniai neuroniniai tinklai

prof. dr. Olga Kurasova
Olga.Kurasova@mii.vu.lt

2018

Kapsuliniai neuroniniai tinklai

- **Kapsuliniai neuroniniai tinklai** (*Capsule Neural Network, CapsNet*) dar vadinami tinklu kapsulėmis.
- Teigama, kad jų veikimas **panašesnis** į biologinių neuronų mechanizmus.
- Pasiūlyti **prieš metus** (2017 m.)

Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in Neural Information Processing Systems* (pp. 3856-3866). <https://arxiv.org/abs/1710.09829>

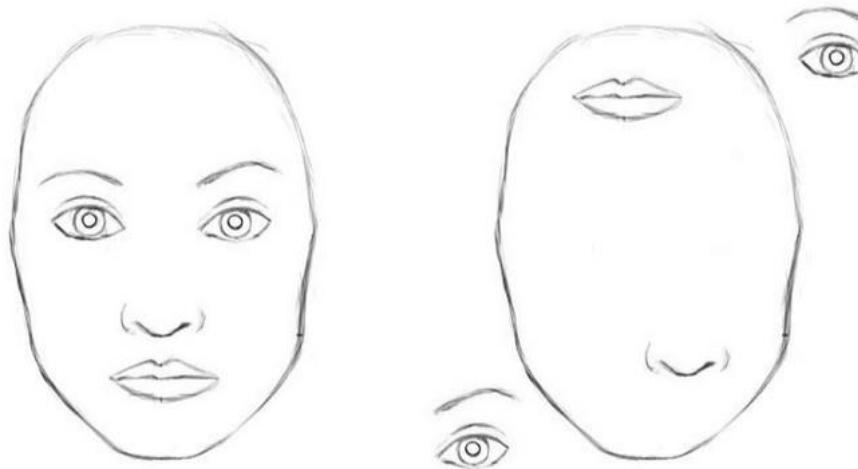
Citatos iš straipsnio:

- Human vision ignores irrelevant details by using a carefully determined sequence of fixation points to ensure that only a tiny fraction of the optic array is ever processed at the highest resolution.
- We show that a discriminatively trained, multi-layer capsule system achieves state-of-the-art performance on MNIST and is considerably better than a convolutional net at recognizing highly overlapping digits.

Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in Neural Information Processing Systems* (pp. 3856-3866). <https://arxiv.org/abs/1710.09829>

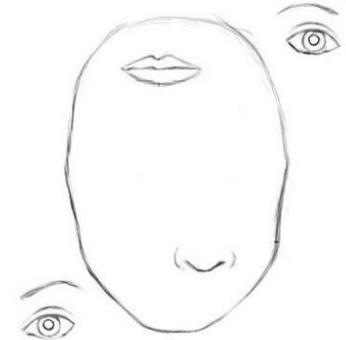
CNN trūkumai

- Nagrinėkime **paprastą pavyzdį**. Iš kokių komponentų sudarytas veidas? Veidas yra ovalios formos, yra dvi akys, nosis ir burna.
- CNN tinklui tai labai svarbios komponentų savybės, tačiau visai **nesvarbi jų padėtis** vieno kito atžvilgiu.

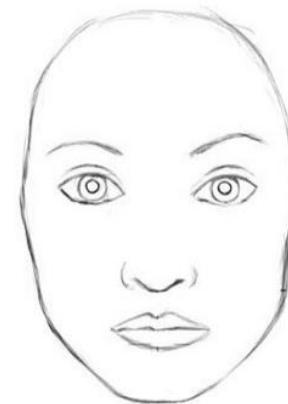


Kas geriau?

```
if (2 eyes && 1 nose && 1 mouth) {  
    It's a face!  
}
```

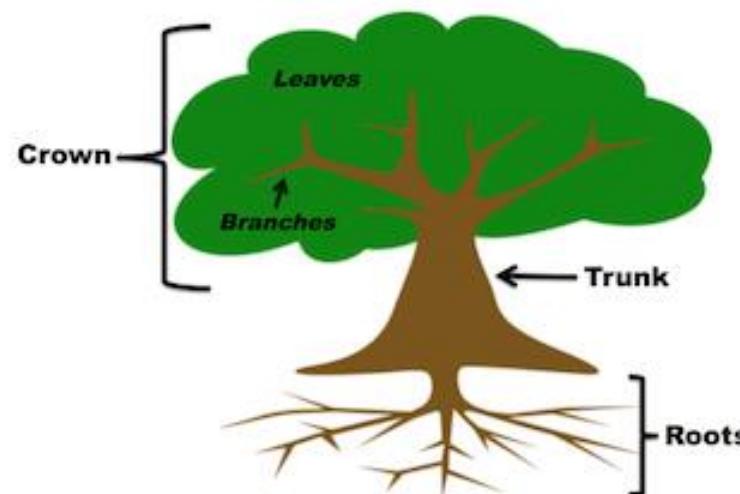


```
if (2 adjacent eyes && nose under eyes && mouth under nose) {  
    It's a face!  
}
```



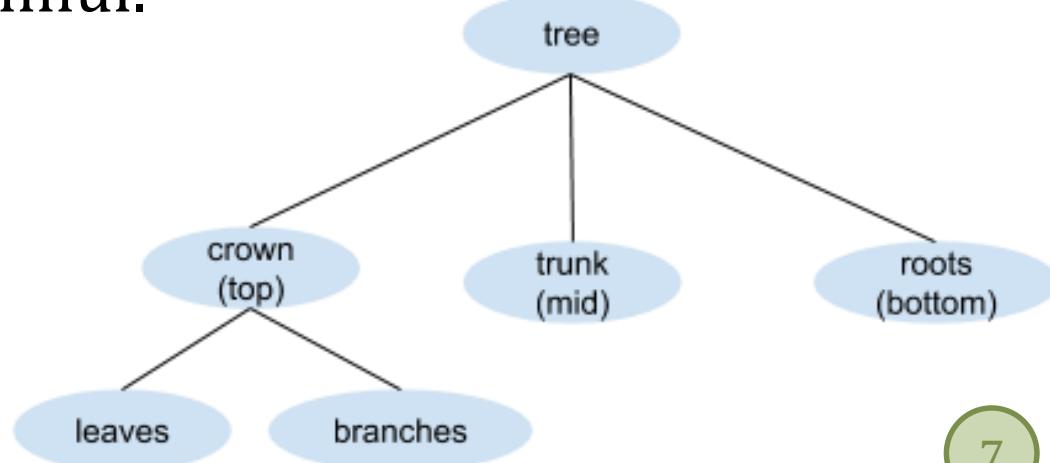
Žmogaus rega

- Objektai **sudaryti iš komponentų** (sudedamųjų dalij). Pvz., medis sudarytas iš kamieno, lapų vainiko ir šaknų.
- Iprastai dalys turi tam tikrą **hierarchiją**. Pvz., lapų vainiką sudaro šakos, šakos turi lapus ir pan.



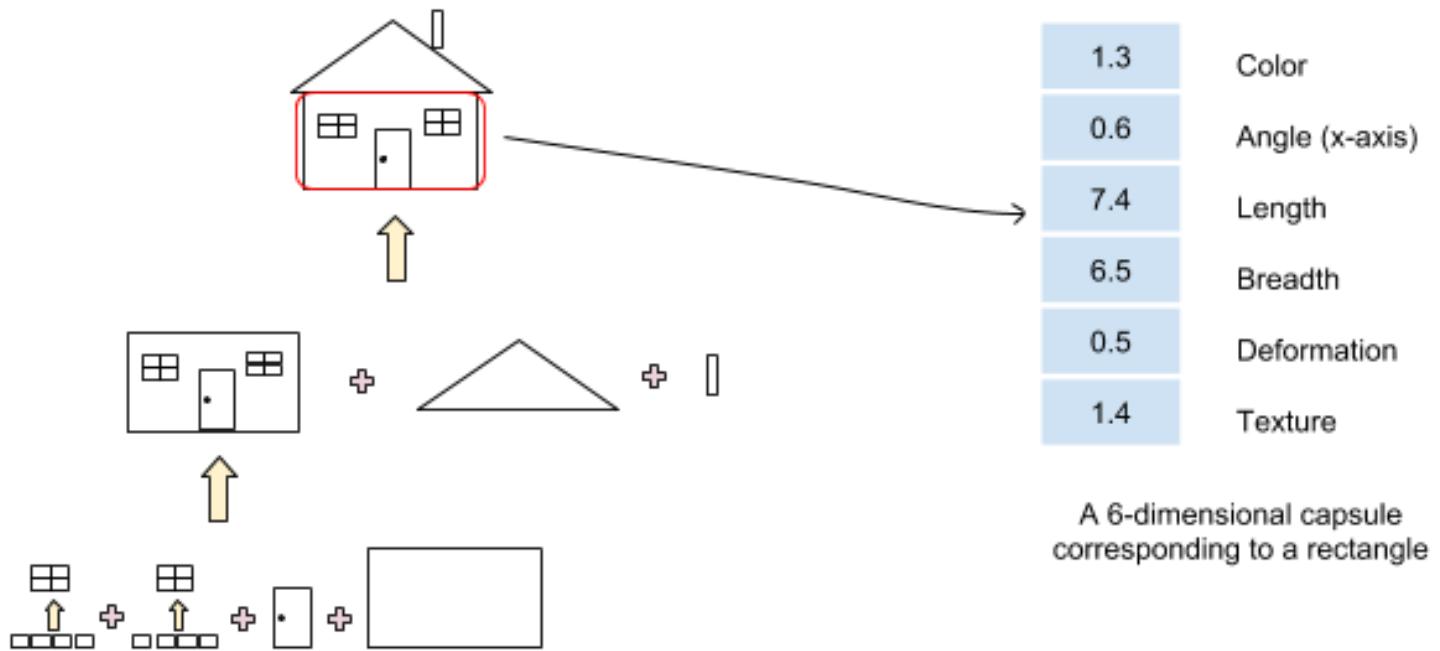
Žmogaus rega

- Kai mes matome tam tikrą objektą, mūsų akys **fiksuoja** tam tikrus taškus, o jų pozicija ir prigimtis **pdeda smegenims atpažinti** šį objektą.
- Smegenys **ne analizuoj**a kiekvieną komponentą tiek detaliai. Sudaroma tam tikra **hierarchija**, kuri padeda atpažinimui.



Pavyzdys

- Nagrinėkime, iš ko sudarytas **namas**. Tegu elementariausias objekto būna **stačiakampis**.
- Sudarykime 6 capsules.



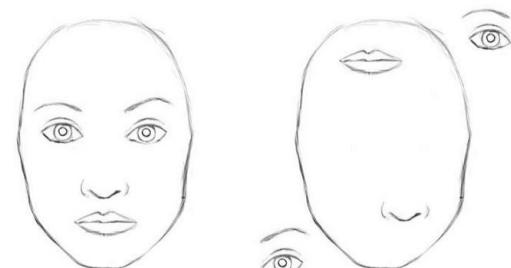
CapsNet vs CNN

- Nustatyta, kad **žmogus**, gavęs vaizdinę informaciją, dekonstruoja hierarchinį aplinkos reprezentavimą ir stengiasi ji sutapatinti su jau **išmoktais šablonais** ir smegenyse saugomus ryšius.
- Objektų reprezentavimas smegenyse nepriklauso nuo juos sudarančių briaunų, kaip yra **CNN paradigmoe**.
- CapsNet tinkle svarbus **objektų ryšys**. Kitais žodžiais – 3D rekonstrukcija.



CNN trūkumai

- „Klaidos skleidimo atgal“ algoritmas reikalauja **didelio kieko duomenų** tinklui išmokyti.
- Pakeitus objekto vietą, neuronai **kitaip aktyvuojami**. Nors CNN tinkluose tai sprendžiama taikant duomenų didinimą (*augmentation*), tačiau tai lieka ne iki galo išspresta problema.
- Naudojant **sujungimo** (*pooling*) sluoksnį prarandamas ryšys objektų.



Kapsulės

- **Kapsulės** – tai neuronų grupės.
- Neuronai sugrupuojami kiekviename sluoksnyje į capsules, siekiant atlikti **vidinį skaičiavimą**, o išėjimuose gaunama „kompaktiška“ informacija.
- Kapsulės išėjimuose gaunami du dalykai:
 - **Tikimybė**, kad objektas priklauso kažkuriui klasei.
 - Apibendrinta **objekto poza – rekonstrukcija** (vieta, orientacija, mastelis, deformacija, spalva) ir kt.

Kapsuliniai neuroniniai tinklai

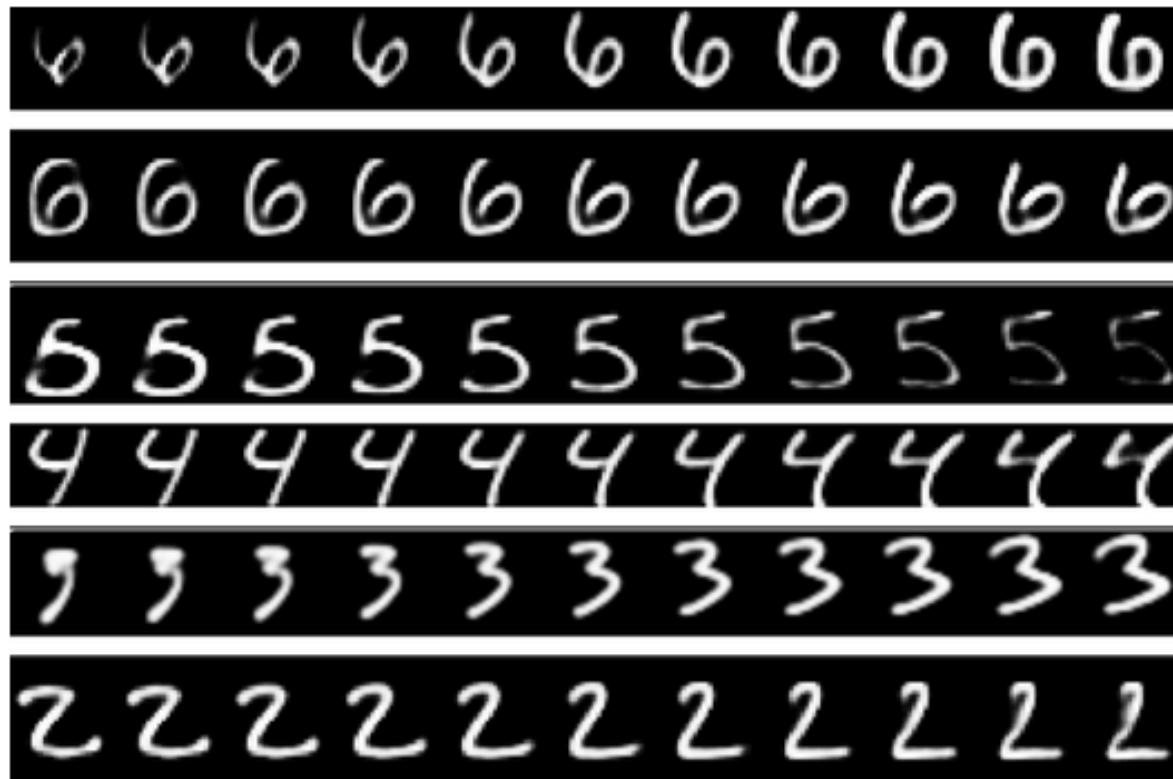
		capsule	VS.	traditional neuron
Input from low-level neurons/capsules		vector(u_i)		scalar(x_i)
Operations	Linear/Affine Transformation	$\hat{u}_{ji} = W_{ij} u_i + B_j$ (Eq. 2)		$a_{ji} = w_{ij} x_i + b_j$
	Weighting	$s_j = \sum_i c_{ij} \hat{u}_{ji}$ (Eq. 2)		$z_j = \sum_{i=1}^3 1 \cdot a_{ji}$
	Summation			
	Non-linearity activation	$v_j = squash(s_j)$ (Eq. 1)		$h_{w,b}(x) = f(z_j)$
output		vector(v_j)		scalar(h)
		$\sum squash(\cdot)$ v_j		
$squash(s) = \frac{\ s\ ^2}{1 + \ s\ ^2} \frac{s}{\ s\ }$				$\sum f(\cdot) h_{w,b}(x)$ <p>$f(\cdot)$: sigmoid, tanh, ReLU, etc.</p>

Capsule = New Version Neuron!
vector in, vector out VS. scalar in, scalar out

<https://github.com/naturomics/CapsNet-Tensorflow>

CapsNet išėjimai

- CapsNet tinklai ne tik **atpažįsta klasę**, bet ir generuoja „vidinius“ **paveikslukus**.



Kapsulių sluoksniai

- **Kapsulės** apjungia savyje neuronų grupes.
- Kapsules sudaro **sluoksnius**.
- **Žemesniame** sluoksnyje kapsulės atitinka paprastus objektus (stačiakampis, trikampis, apskritimas). **Aukštesniame** – sudėtingesnius objektus (durys, langai).
- Būtinas **apjungimo** mechanizmas. Autoriai jį vadina „Dynamic routing between capsules“

Routing algorithm

Procedure 1 Routing algorithm.

```
1: procedure ROUTING( $\hat{u}_{j|i}$ ,  $r$ ,  $l$ )
2:   for all capsule  $i$  in layer  $l$  and capsule  $j$  in layer  $(l + 1)$ :  $b_{ij} \leftarrow 0$ .
3:   for  $r$  iterations do
4:     for all capsule  $i$  in layer  $l$ :  $c_i \leftarrow \text{softmax}(b_i)$             $\triangleright \text{softmax computes Eq. 3}$ 
5:     for all capsule  $j$  in layer  $(l + 1)$ :  $s_j \leftarrow \sum_i c_{ij} \hat{u}_{j|i}$ 
6:     for all capsule  $j$  in layer  $(l + 1)$ :  $v_j \leftarrow \text{squash}(s_j)$             $\triangleright \text{squash computes Eq. 1}$ 
7:     for all capsule  $i$  in layer  $l$  and capsule  $j$  in layer  $(l + 1)$ :  $b_{ij} \leftarrow b_{ij} + \hat{u}_{j|i} \cdot v_j$ 
return  $v_j$ 
```

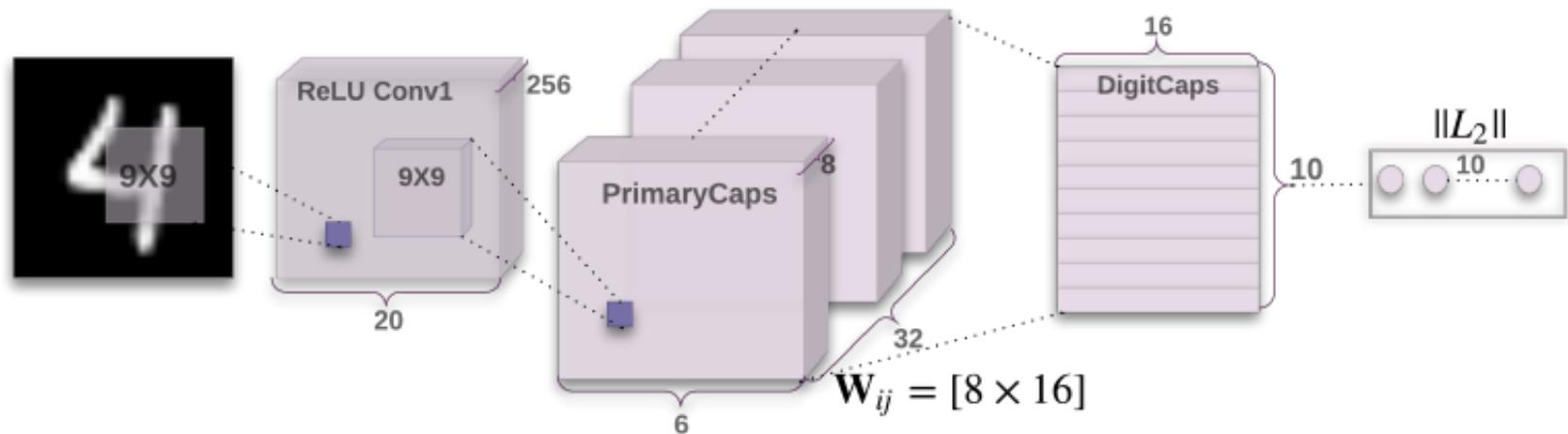
$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|} \quad (1)$$

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \quad (3)$$

Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in Neural Information Processing Systems* (pp. 3856-3866). <https://arxiv.org/abs/1710.09829>

CapsNet architektūra

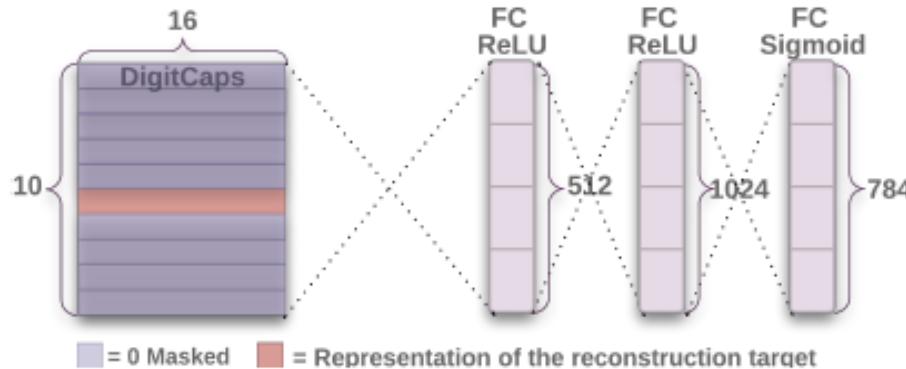
Figure 1: A simple CapsNet with 3 layers. This model gives comparable results to deep convolutional networks (such as Chang and Chen [2015]). The length of the activity vector of each capsule in DigitCaps layer indicates presence of an instance of each class and is used to calculate the classification loss. \mathbf{W}_{ij} is a weight matrix between each $\mathbf{u}_i, i \in (1, 32 \times 6 \times 6)$ in PrimaryCapsules and $\mathbf{v}_j, j \in (1, 10)$.



Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in Neural Information Processing Systems* (pp. 3856-3866). <https://arxiv.org/abs/1710.09829>

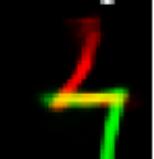
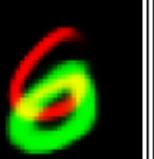
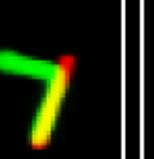
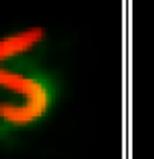
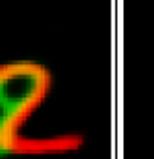
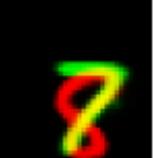
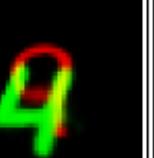
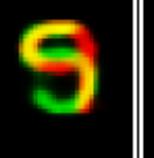
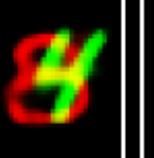
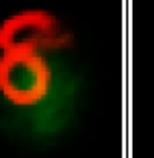
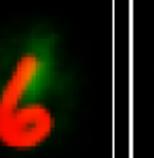
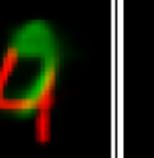
CapsNet architektūra

Figure 2: Decoder structure to reconstruct a digit from the DigitCaps layer representation. The euclidean distance between the image and the output of the Sigmoid layer is minimized during training. We use the true label as reconstruction target during training.



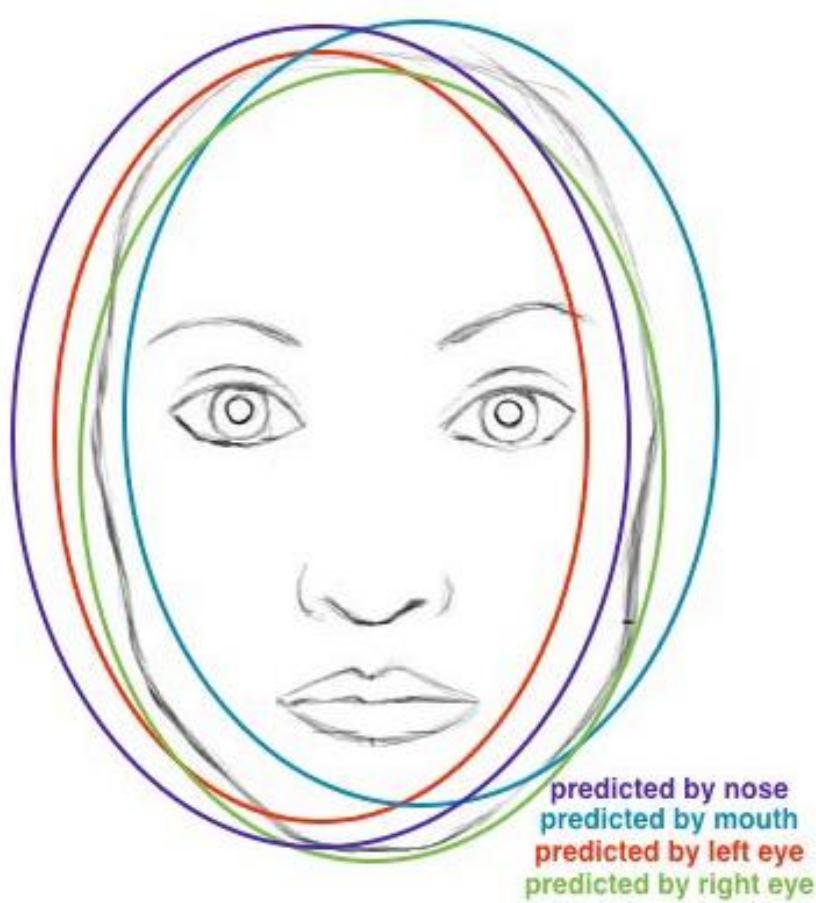
Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in Neural Information Processing Systems* (pp. 3856-3866). <https://arxiv.org/abs/1710.09829>

CapsNet rezultatai

R:(2, 7) L:(2, 7)	R:(6, 0) L:(6, 0)	R:(6, 8) L:(6, 8)	R:(7, 1) L:(7, 1)	*R:(5, 7) L:(5, 0)	*R:(2, 3) L:(4, 3)	R:(2, 8) L:(2, 8)	R:P:(2, 7) L:(2, 8)
							
							
R:(8, 7) L:(8, 7)	R:(9, 4) L:(9, 4)	R:(9, 5) L:(9, 5)	R:(8, 4) L:(8, 4)	*R:(0, 8) L:(1, 8)	*R:(1, 6) L:(7, 6)	R:(4, 9) L:(4, 9)	R:P:(4, 0) L:(4, 9)
							
							

Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in Neural Information Processing Systems* (pp. 3856-3866). <https://arxiv.org/abs/1710.09829>

CapsNet



<https://medium.com/ai%C2%B3-theory-practice-business/understanding-hintons-capsule-networks-part-ii-how-capsules-work-153b6ade9f66>

CapsNet

- Pirmoji CapsNet dalis yra tradicinis **konvoliucinis sluoksnis** ir **ReLU** sluoksnis. Tikslas – išskirti **pagrindinius požymius**, tokius kaip briaunos.
- Antroji dalis – **PrimaryCaps**. Čia pradedama nuo paprastų požymių (*instantiation parameters*), baigiamai – sudėtingais. Naudojama **Squashing** funkcija (nepakeičiant vektoriaus krypties, o tik jo ilgi, rezultatas intervale (0,1)).
- Trečioji dalis – **Rooting** procedūra (vietoj MaxPooling).
- Ketvirtoji dalis – **DigitCaps** (Class Capsules)
- Penktoji dalis – **rekonstrukcija**.

CapsNet privalumai

- Pasiekiamas **aukštesnis** tikslumas (nei CNN).
- Reikalauja **mažiau** mokymo duomenų.
- Tinkamesni vaizdų **segmentavimui** ir objektų **atpažinimui**.
- Atpažįsta **persidengiančius** objektus.

Dar kartą apie CapsNet

- <https://www.analyticsvidhya.com/blog/2018/04/essentials-of-deep-learning-getting-to-know-capsulenets/>
- <https://software.intel.com/en-us/articles/understanding-capsule-network-architecture>
-

CapsNet taikymai

Nors jie sukurti visai neseniai, bet jau pradėti intensyviai **taikyti įvairiems uždaviniams spręsti:**

- Kim, Y., Wang, P., Zhu, Y., & Mihaylova, L. (2018, October). A Capsule Network for Traffic Speed Prediction in Complex Road Networks. In 2018 Sensor Data Fusion: Trends, Solutions, Applications (SDF) (pp. 1-6). IEEE.
- Zhu, Z., Peng, G., Chen, Y., & Gao, H. (2019). A convolutional neural network based on a capsule network with strong generalization for bearing fault diagnosis. Neurocomputing, 323, 62-75.
- Afshar, P., Mohammadi, A., & Plataniotis, K. N. (2018). Brain tumor type classification via capsule networks. arXiv preprint arXiv:1802.10200.
- Daugiau straipsnių:
<http://principlesofdeeplearning.com/index.php/resources/list-of-research-papers-on-capsule-networks/>



Vilniaus universitetas
Matematikos ir informatikos fakultetas
Informatikos katedra



Dirbtiniai neuroniniai tinklai daugiamatičių duomenų dimensija mažinti (vizualizuoti)

prof. dr. Olga Kurasova
Olga.Kurasova@mii.vu.lt

2018

Daugamačiai duomenys. kas tai?

- **Daugamačiai duomenys** – tai objektą charakterizuojančių **savybių** (rodiklių, parametru) reikšmių rinkinys.
- Jei savybių reikšmės x_1, x_2, \dots, x_n yra skaičiai, daugamačius duomenis atitinka **taškai (vektoriai)** $X_i = (x_{i1}, x_{i2}, \dots, x_{in}), i = 1, \dots, m$, čia m – objektų skaičius, n – savybių skaičius.
- Analizuojamų daugamačių duomenų aibė yra **matrica** (lentelė):
$$X = \{X_1, X_2, \dots, X_m\} = \{x_{ij}, i = 1, \dots, m, j = 1, \dots, n\}$$
, kurios i -oji eilutė yra vektorius $X_i \in R^n$, eilutės atitinka analizuojamus objektus, stulpeliai – savybes.

Daugiamatių duomenų pavyzdys (vėžio duomenys)

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	C
X_1	5	1	1	1	2	1	3	1	1	b
X_2	5	4	4	5	7	10	3	2	1	b
X_3	3	1	1	1	2	2	3	1	1	b
X_4	6	8	8	1	3	4	3	7	1	b
X_5	4	1	1	3	2	1	3	1	1	b
X_6	1	1	1	1	2	10	3	1	1	b
X_7	2	1	2	1	2	1	3	1	1	b
X_8	2	1	1	1	2	1	1	1	5	b
X_9	4	2	1	1	2	1	2	1	1	b
...
X_{460}	8	10	10	8	7	10	9	7	1	m
X_{461}	5	3	3	3	2	3	4	4	1	m
X_{462}	8	7	5	10	7	9	5	5	4	m
X_{463}	7	4	6	4	6	1	4	3	1	m
X_{464}	10	7	7	6	4	10	4	1	2	m
X_{465}	7	3	2	10	5	10	5	4	4	m
X_{466}	10	5	5	3	6	7	7	10	1	m
...
X_{699}	4	8	8	5	4	5	10	4	1	m

x_1 – clump thickness,
 x_2 – uniformity of cell size,
 x_3 – uniformity of cell shape,
 x_4 – marginal adhesion,
 x_5 – single epithelial cell size,
 x_6 – bare nuclei,
 x_7 – bland chromatin,
 x_8 – normal nucleoli,
 x_9 – mitoses,
C – class (**benign**, **malignant**)

dimensija (matmenų skaičius)
 $n = 9$
objektų (duomenų elementų)
skaičius $m = 699$

Kokios daugiamatių duomenų ypatybės?

- Turint duomenis, kai $n = 1$, taškus galima atvaizduoti **tiesėje**.
- Turint duomenis, kai $n = 2$ (arba $n = 3$), taškus galima atvaizduoti Dekarto **plokštumoje** (arba erdvėje).
- **Ką daryti**, kai $n > 3$? ☹ ☹ ☹
- Pasitelksime įvairius daugiamatių duomenų **vizualizavimo metodus**.

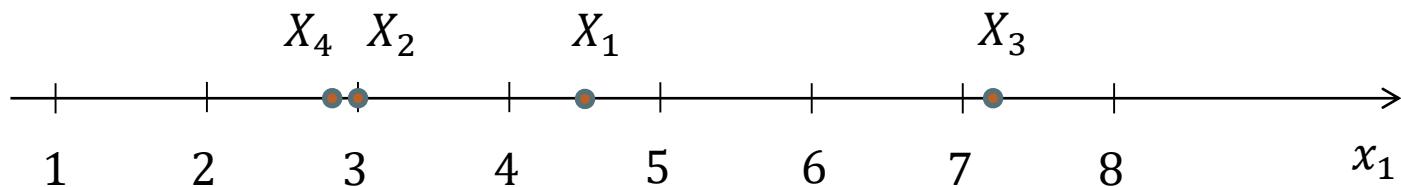
Kai $n = 1$

$$X_1 = (4,5)$$

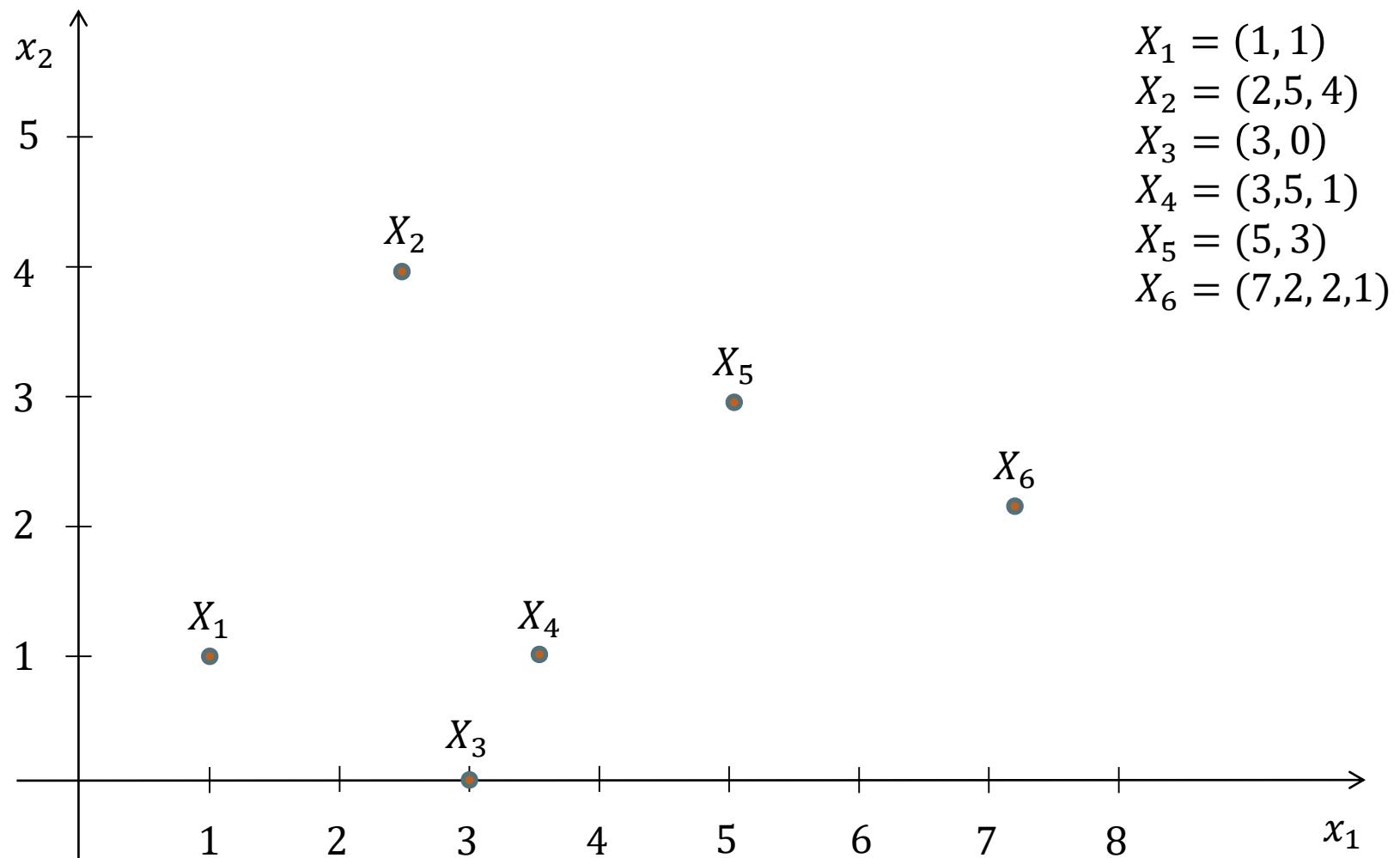
$$X_2 = (3)$$

$$X_3 = (7,2)$$

$$X_4 = (2,8)$$



Kai $n = 2$. taškinis grafikas



Daugiamatių duomenų vizualizavimo metodai

- **Tiesioginio** vizualizavimo metodai (nėra griežto matematinio kriterijaus):
 - Černovo veidai
 - Žvaigždžių metodas
 - Andrews kreivės
 - Lygiagrečiosios koordinatės
 - Kt.
- **Dimensijų mažinimu** grįsti vizualizavimo (projekcijos) metodai:
 - Pagrindinių komponenčių metodas
 - Daugiamatės skalės
 - **Dirbtiniai neuroniniai tinklai** grįsti metodai
 - Kt.



Irisų gėlių duomenys

- Buvo išmatuoti irisų gėlių žiedų:
 - vainiklapių pločiai (x_1)
 - vainiklapių ilgiai (x_2)
 - taurėlapių ilgiai (x_3)
 - taurėlapių pločiai (x_4)
- Matuotos trijų veislių gėlės (po 50 kiekvienos veislės, viso 150)
- $X = \{X_1, X_2, \dots, X_{150}\} = \{x_{ij}, i = 1, \dots, 150, j = 1, \dots, 4\}$

Iris-Setosa



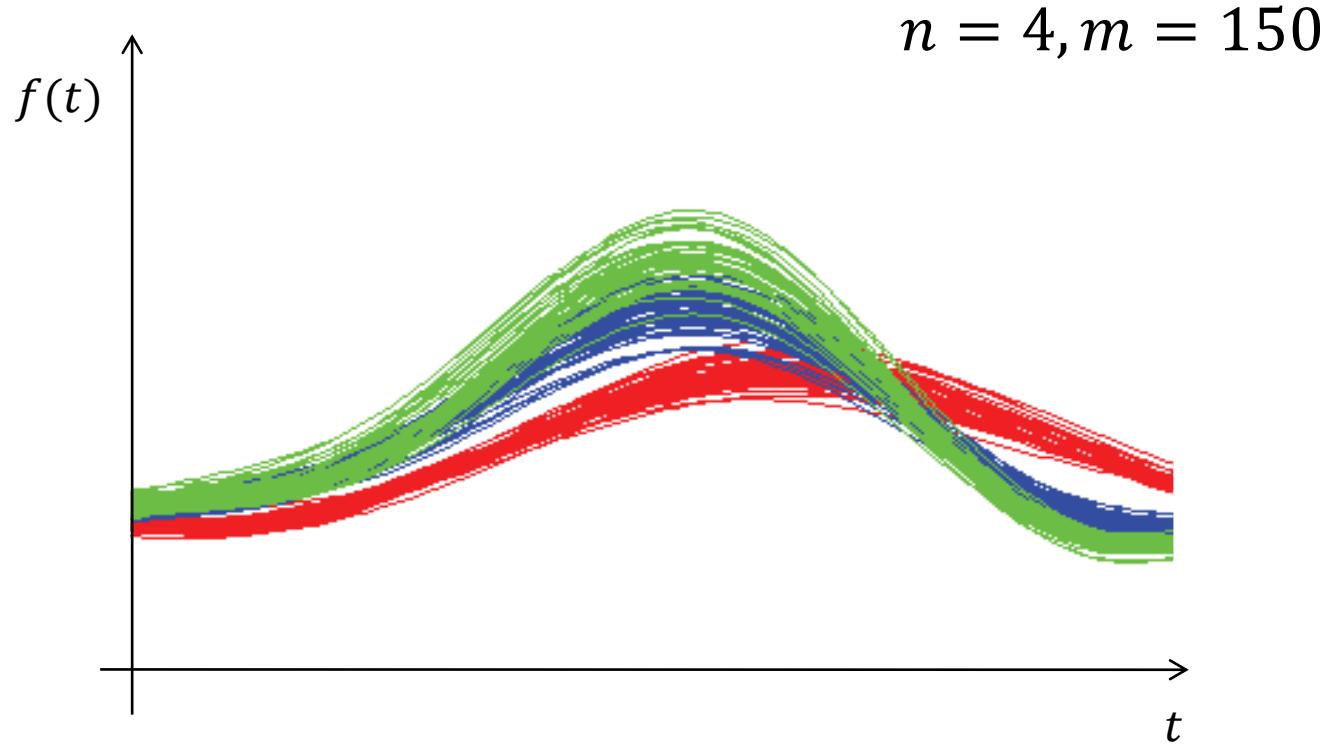
Iris-Versicolor



Iris-Virginica

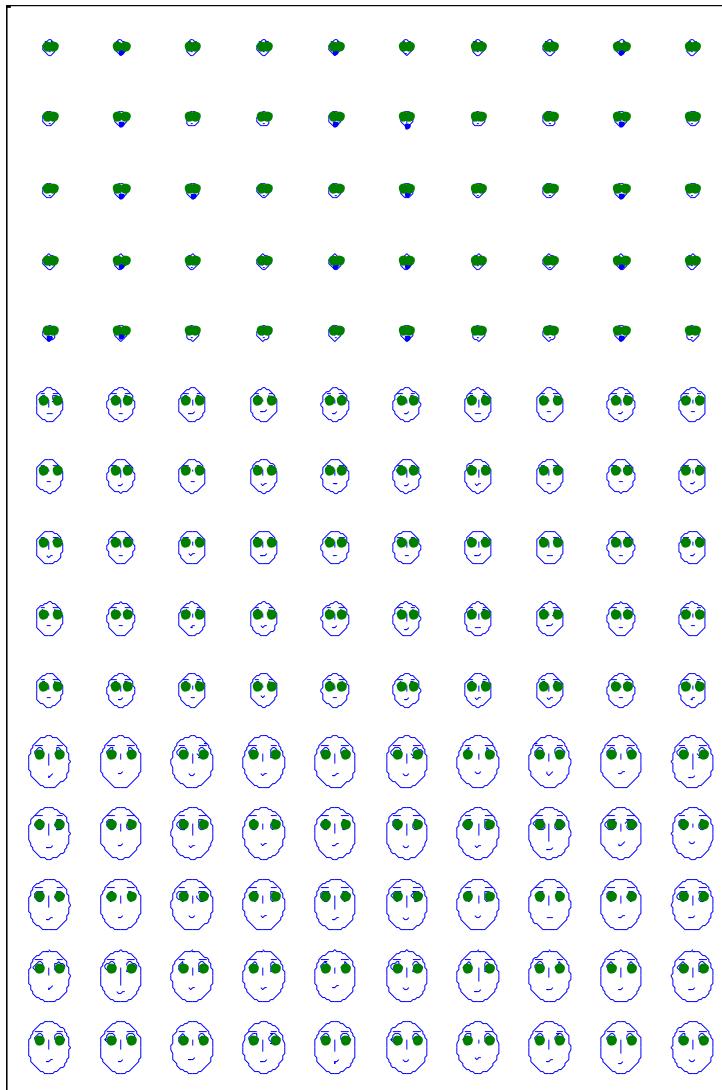


Irisų duomenų vizualizavimas Andrews kreivėmis



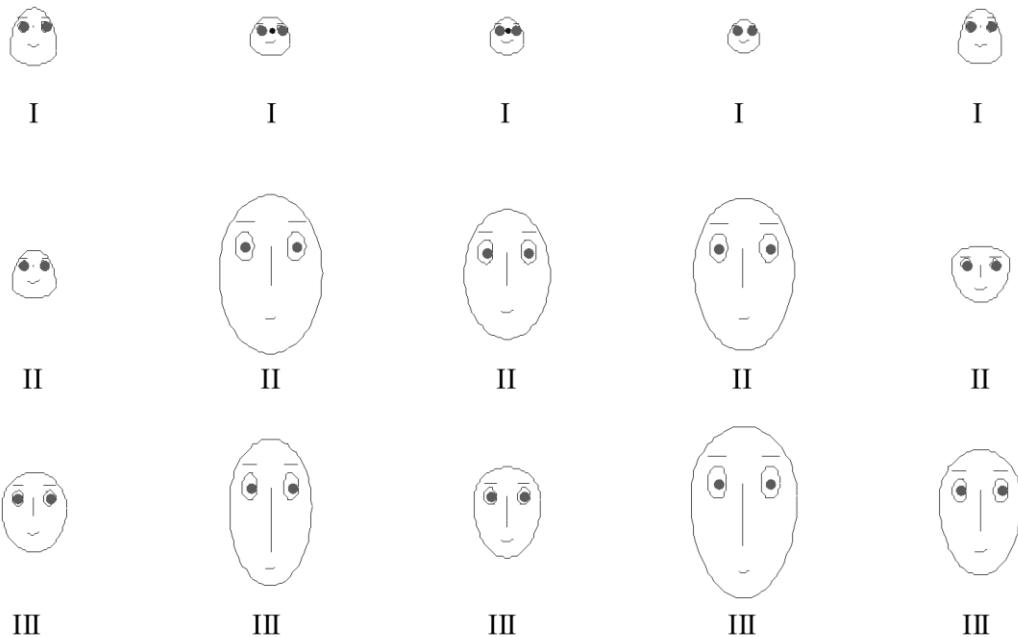
$$f_i(t) = \frac{x_{i1}}{\sqrt{2}} + x_{i2} \sin(t) + x_{i3} \cos(t) + x_{i4} \sin(2t) + x_{i5} \cos(2t) + \dots \quad -\pi < t < \pi$$

Irisų duomenų vizualizavimas Černovo veidais



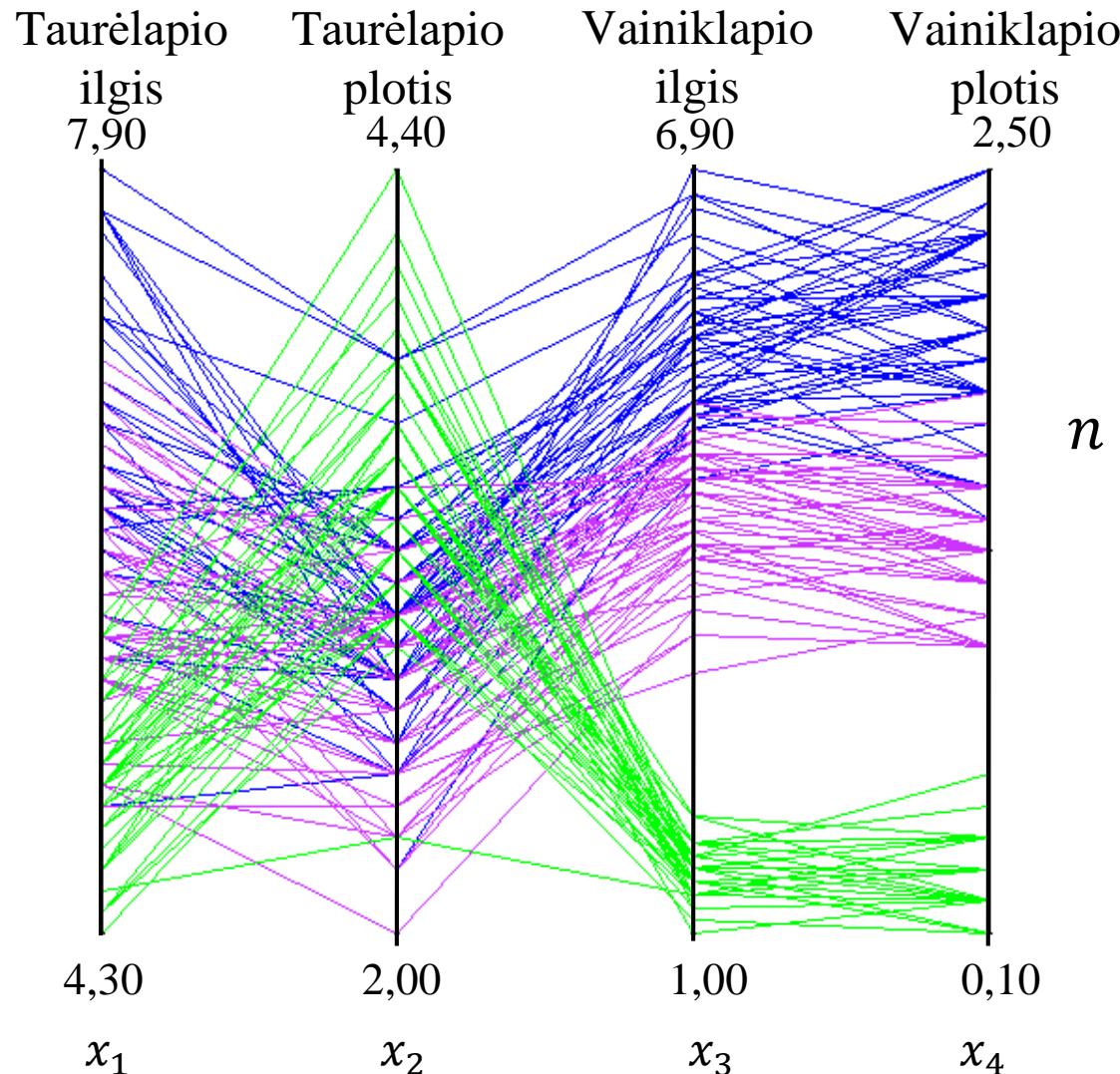
$n = 4, m = 150$

Irisų duomenų vizualizavimas Černovo veidais



x_1 – veido dydis
 x_2 – santykis tarp
kaktos/smakro arkų ilgių
 x_3 – kaktos forma
 x_4 – smakro dydis

Irisų duomenys lygiagrečiose koordinatėse



Dimensijų mažinimu grįstas vizualizavimas

- **Tikslas** – transformuoti daugiamąčius duomenis

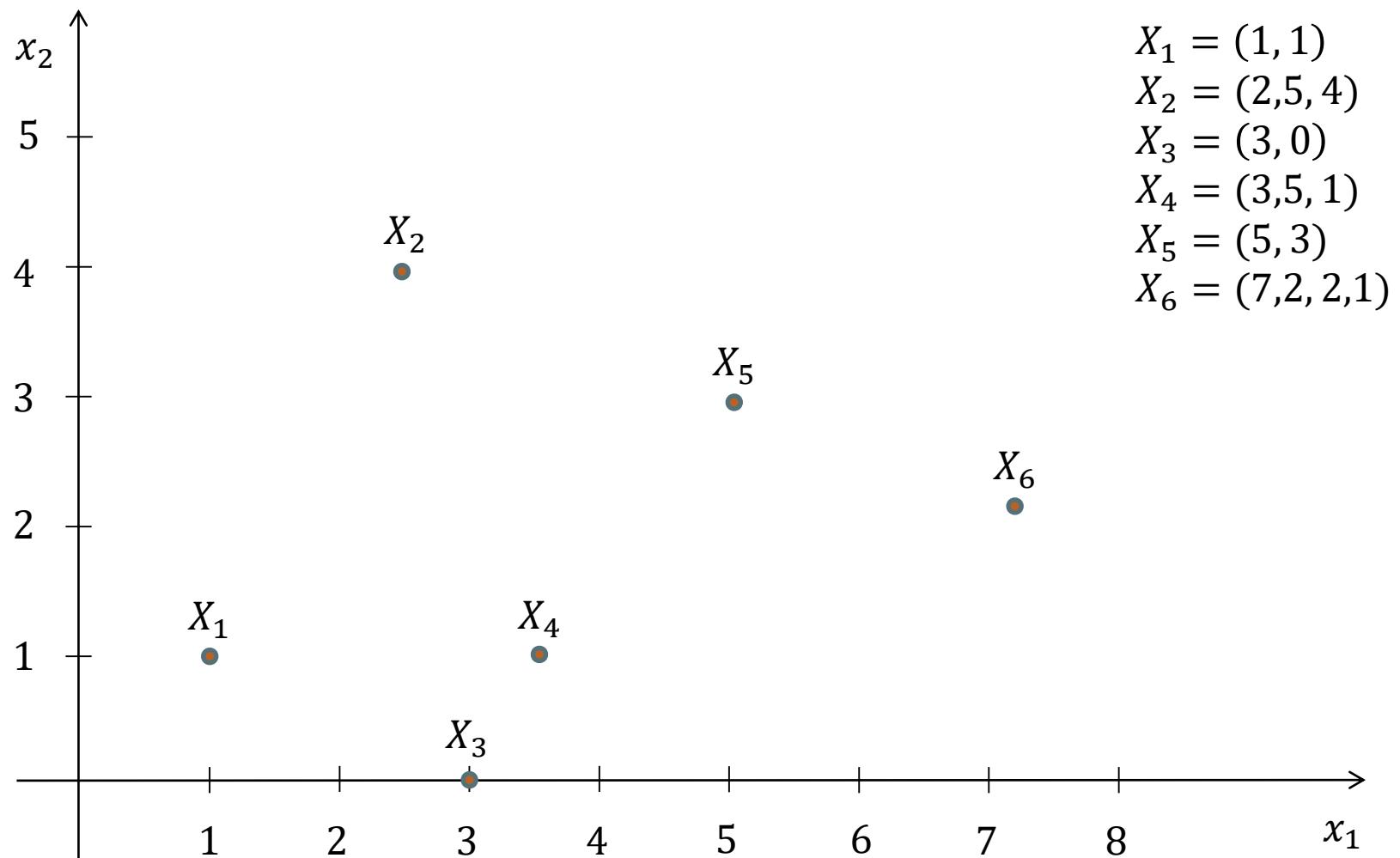
$$X_i = (x_{i1}, x_{i2}, \dots, x_{in}) \in R^n, i = 1, \dots, m,$$

į mažesnės dimensijos erdvės duomenis

$$Y_i = (y_{i1}, y_{i2}, \dots, y_{id}) \in R^d, d < n.$$

- Kai $d = 2$, gautus duomenis (vektorius, taškus) galima **atvaizduoti** įprastoje Dekarto koordinacijų sistemoje.

Kai $n = 2$. taškinis grafikas



Dimensijų mažinimu grįsti vizualizavimo metodai

- Pagrindinių komponenčių analizė (*principal component analysis*)
 - tikslas – **išlaikyti dispersijas**
- Daugiamatičių skalių metodas (*multidimensional scaling*)
 - tikslas – **išlaikyti panašumus** (pvz., atstumus)
- Dirbtiniai neuroniniai tinklais grįsti metodai
 - Iprasti tiesioginio sklidimo neuroniniai tinklai
 - Autoasociatyvieji neuroniniai tinklai
 - Saviorganizuojantys neuroniniai tinklai (*self-organizing maps*),
 - Įvairūs junginiai
- Kiti metodai

Vėžio duomenys

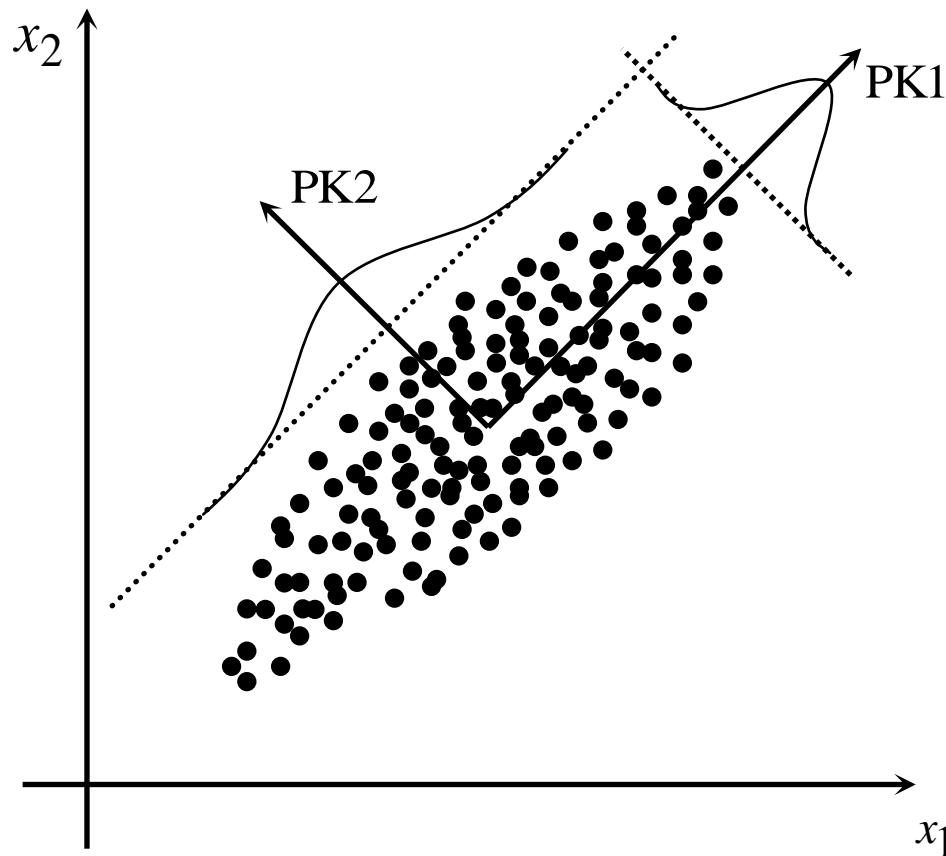
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	C
X_1	5	1	1	1	2	1	3	1	1	b
X_2	5	4	4	5	7	10	3	2	1	b
X_3	3	1	1	1	2	2	3	1	1	b
X_4	6	8	8	1	3	4	3	7	1	b
X_5	4	1	1	3	2	1	3	1	1	b
X_6	1	1	1	1	2	10	3	1	1	b
X_7	2	1	2	1	2	1	3	1	1	b
X_8	2	1	1	1	2	1	1	1	5	b
X_9	4	2	1	1	2	1	2	1	1	b
...
X_{460}	8	10	10	8	7	10	9	7	1	m
X_{461}	5	3	3	3	2	3	4	4	1	m
X_{462}	8	7	5	10	7	9	5	5	4	m
X_{463}	7	4	6	4	6	1	4	3	1	m
X_{464}	10	7	7	6	4	10	4	1	2	m
X_{465}	7	3	2	10	5	10	5	4	4	m
X_{466}	10	5	5	3	6	7	7	10	1	m
...
X_{699}	4	8	8	5	4	5	10	4	1	m

x_1 – clump thickness,
 x_2 – uniformity of cell size,
 x_3 – uniformity of cell shape,
 x_4 – marginal adhesion,
 x_5 – single epithelial cell size,
 x_6 – bare nuclei,
 x_7 – bland chromatin,
 x_8 – normal nucleoli,
 x_9 – mitoses,
C – class (**b**enign, **m**alignant)

Pagrindinių komponenčių analizė (PCA)

- Esminė PCA idėja yra sumažinti duomenų matmenų skaičių atliekant tiesinę transformaciją ir atsisakant dalies po transformacijos gautų naujų komponenčių, kurių **dispersijos yra mažiausios**.
- Didžiausią dispersiją turinti kryptis vadina **pirmaja pagrindine komponente** (PK1). Ji eina per duomenų centrinį tašką. Tai taškas, kurio komponentės yra analizuojamą duomenų aibę sudarančių taškų atskirų komponenčių vidurkiai. Visų taškų vidutinis atstumas iki šios tiesės yra minimalus, t. y. ši tiesė yra kiek galima arčiau visų duomenų taškų.
- **Antrosios pagrindinės komponentės** (PK2) ašis taip pat turi eiti per duomenų centrinį tašką ir ji turi būti statmena pirmosios pagrindinės komponentės ašiai.

Pagrindinių komponenčių analizė (PCA)



Daugiamačių skalių (MDS) metodas (1)

- Ieškoma daugiamačių duomenų projekcijų mažesnės dimensijos erdvėje (dažniausiai R^2 arba R^3), siekiant išlaikyti analizuojamos aibės objektų **artimumus** – panašumus arba skirtingumus.
- Gautuose vaizduose panašūs objektai išdėstomi **arčiau** vieni kitų, o skirtini – **toliau** vieni nuo kitų.
- Dažniausia artumo matas yra **atstumas** (**Euklido**, miesto kvartalų ir kt.).

Daugiamočių skalių (MDS) metodas (2)

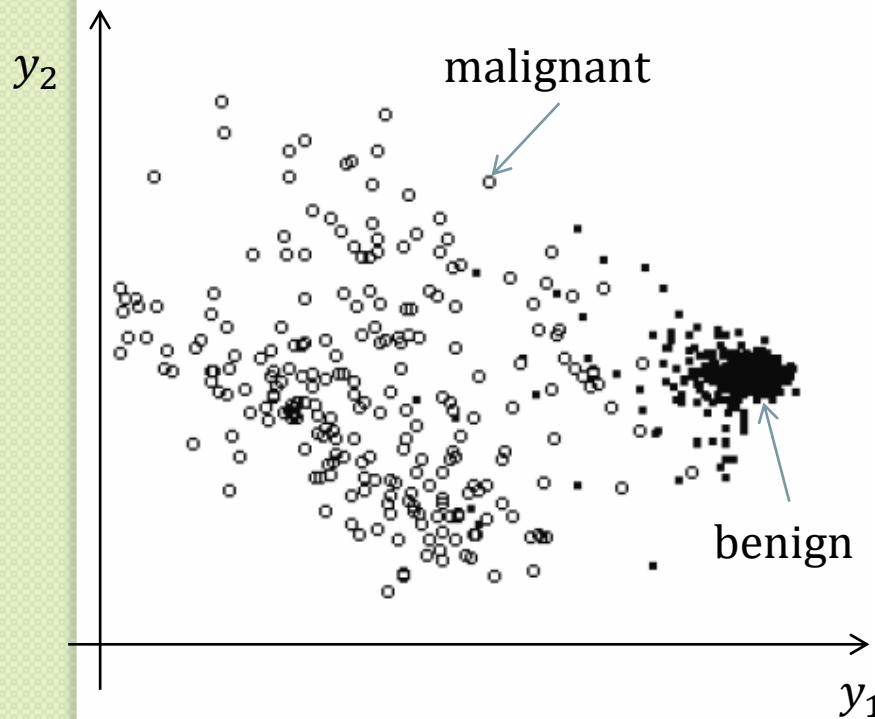
- Tarkime, kiekvieną n -matį vektorių $X_i \in R^n$, $i = 1, \dots, m$, atitinka **mažesnės dimensijos vektorius** $Y_i \in R^d$, $d < n$.
- Atstumą tarp vektorių X_i ir X_j pažymėkime $d(X_i, X_j)$, o atstumą tarp vektorių Y_i ir Y_j – $d(Y_i, Y_j)$.
- Naudojantis MDS metodu, bandoma **atstumus** $d(Y_i, Y_j)$ **priartinti prie atstumu** $d(X_i, X_j)$.
- Jei naudojama kvadratinė paklaidos funkcija, tai **minimizuojama tikslų funkcija** užrašoma:

$$E_{MDS} = \sum_{i < j} w_{ij} (d(X_i, X_j) - d(Y_i, Y_j))^2$$

Vėžio duomenų vizualizavimas MDS metodu

- Daugiamatių duomenų transformavimas į mažesnės dimensijos erdvę

$$(x_{i1}, x_{i2}, \dots, x_{i9}) \rightarrow (y_{i1}, y_{i2})$$

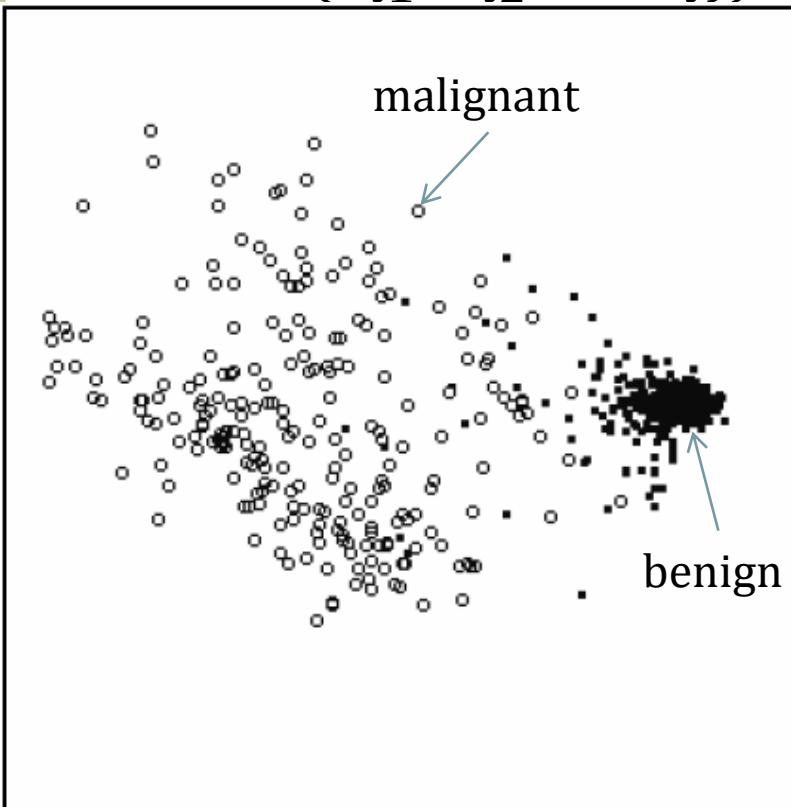


- vienas **taškas** atitinka vieną **paciente**
- vienas **taškas** apjungia visas **devynias savybes**
- vizualus pateikimas padeda lengviau suvokti **informacijos visumą**

Vėžio duomenų vizualizavimas MDS metodu

- Daugiamatių duomenų transformavimas į mažesnės dimensijos erdvę

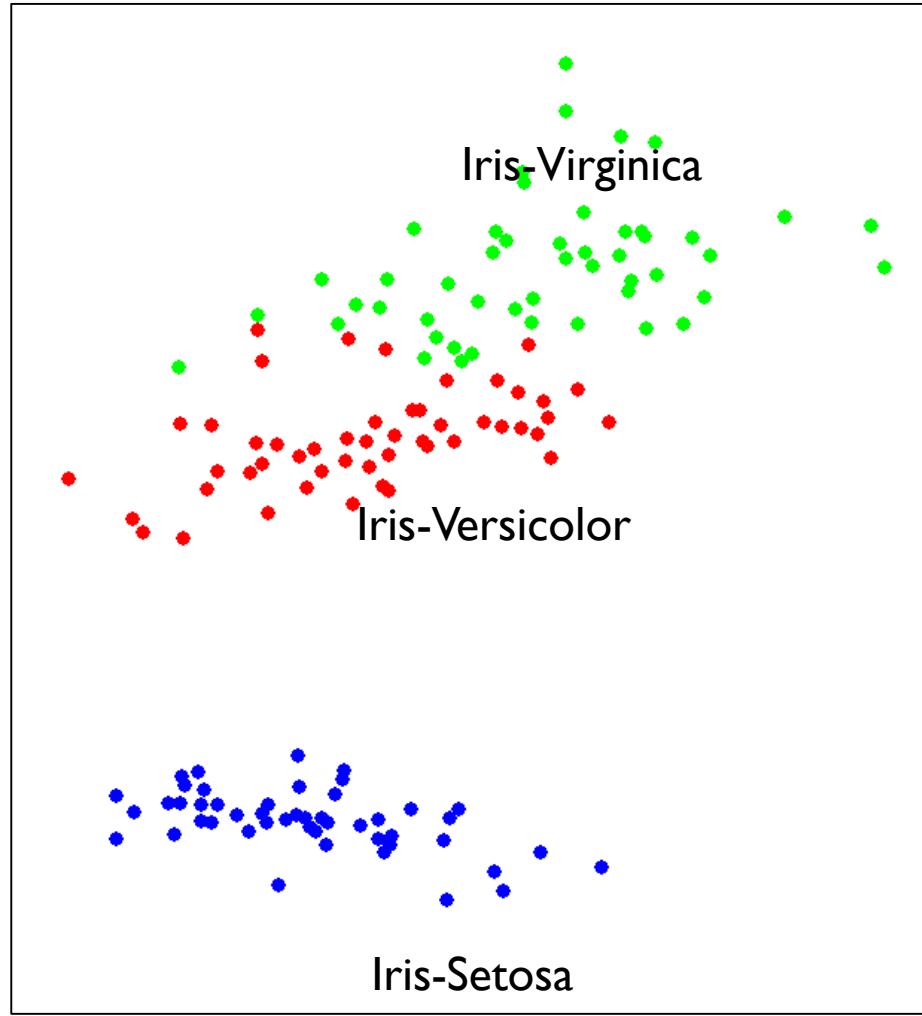
$$(x_{i1}, x_{i2}, \dots, x_{i9}) \rightarrow (y_{i1}, y_{i2})$$



- vienas **taškas** atitinka vieną **paciente**
- vienas **taškas** apjungia visas **devynias savybes**
- vizualus pateikimas padeda lengviau suvokti **informacijos visumą**

Irisų duomenų vizualizavimas MDS metodu

	x_1	x_2	x_3	x_4	C
X_1	5,1	3,5	1,4	0,2	Set.
X_2	4,9	3,0	1,4	0,2	Set.
X_3	4,7	3,2	1,3	0,2	Set.
X_4	4,6	3,1	1,5	0,2	Set.
X_5	5,0	3,6	1,4	0,2	Set.
...
X_{51}	7,0	3,2	4,7	1,4	Ver.
X_{52}	6,4	3,2	4,5	1,5	Ver.
X_{53}	6,9	3,1	4,9	1,5	Ver.
X_{54}	5,5	2,3	4,0	1,3	Ver.
X_{55}	6,5	2,8	4,6	1,5	Ver.
...
X_{101}	5,7	2,8	4,1	1,3	Virg.
X_{102}	6,3	3,3	6,0	2,5	Virg.
X_{103}	5,8	2,7	5,1	1,9	Virg.
X_{104}	7,1	3,0	5,9	2,1	Virg.
X_{105}	6,3	2,9	5,6	1,8	Virg.
...



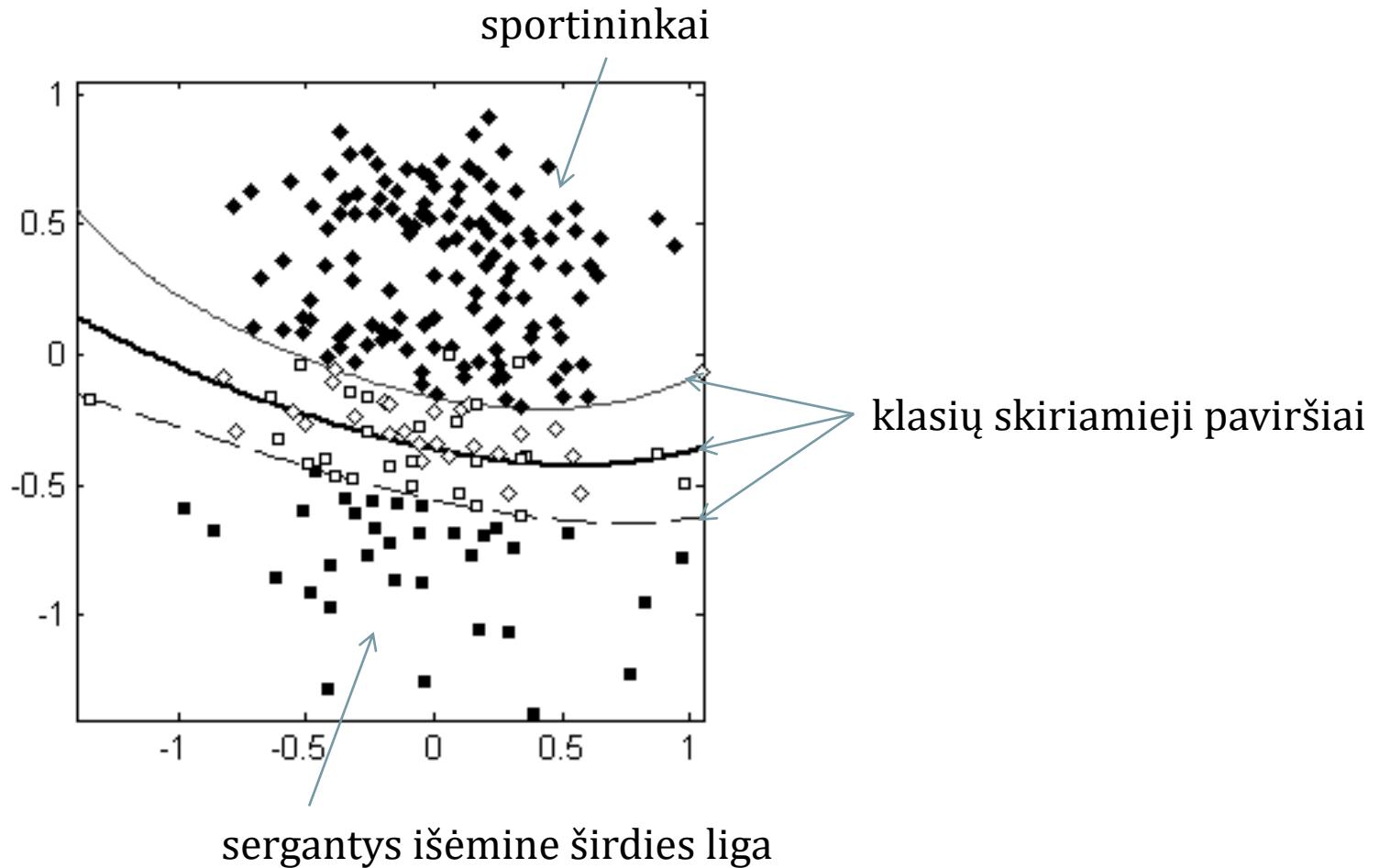
Daugiamatių skalių (MDS) metodas (3)

Įvairūs **optimizavimo** metodai:

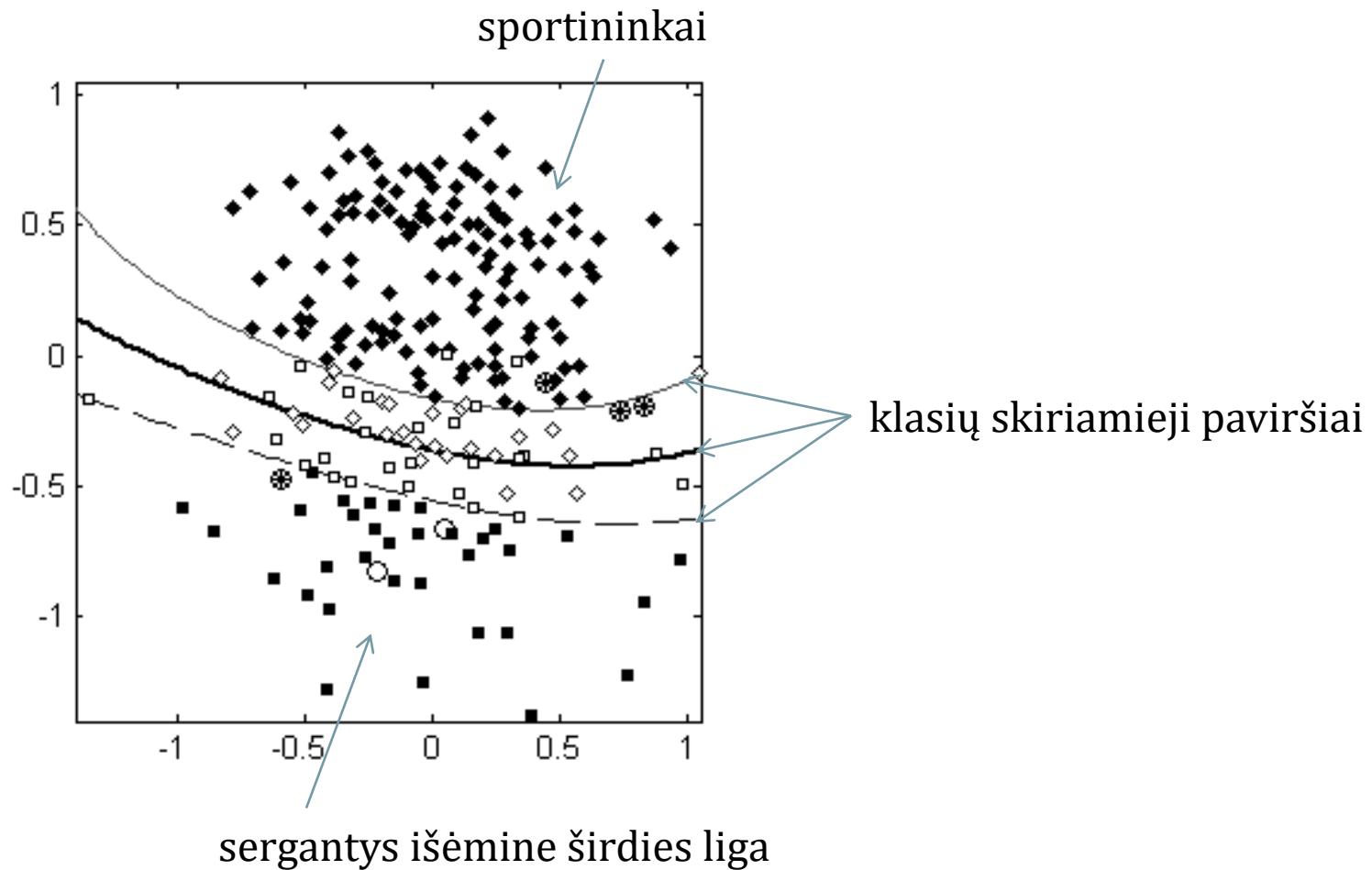
- Gradientinis nusileidimas
- Genetinis algoritmas
- Šakų-rėžių algoritmas
- Funkcijos mažorizavimu grįstas algoritmas
(SMACOF)
- Kt.



Klasių skiriamieji paviršiai

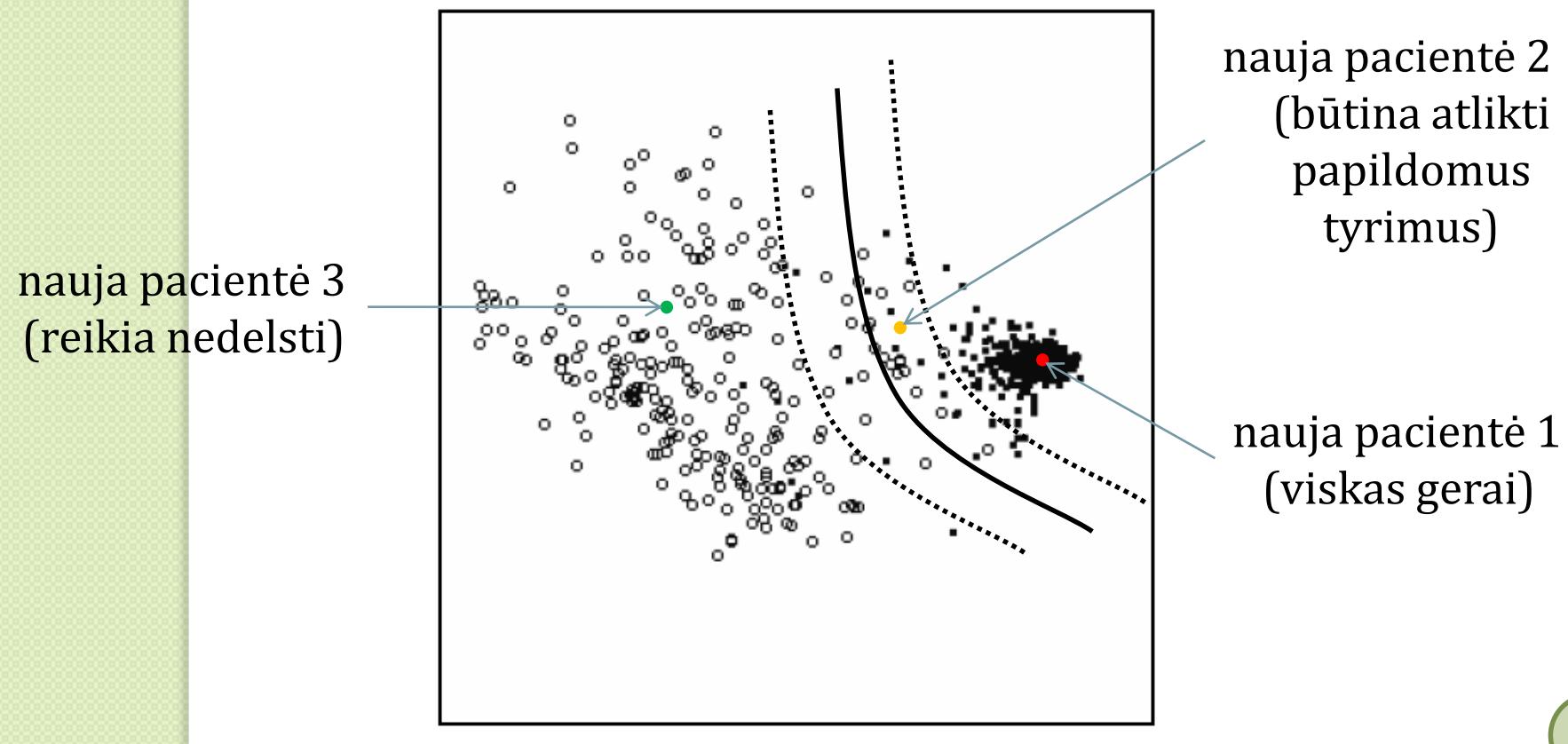


Naujų taškų atidėjimas

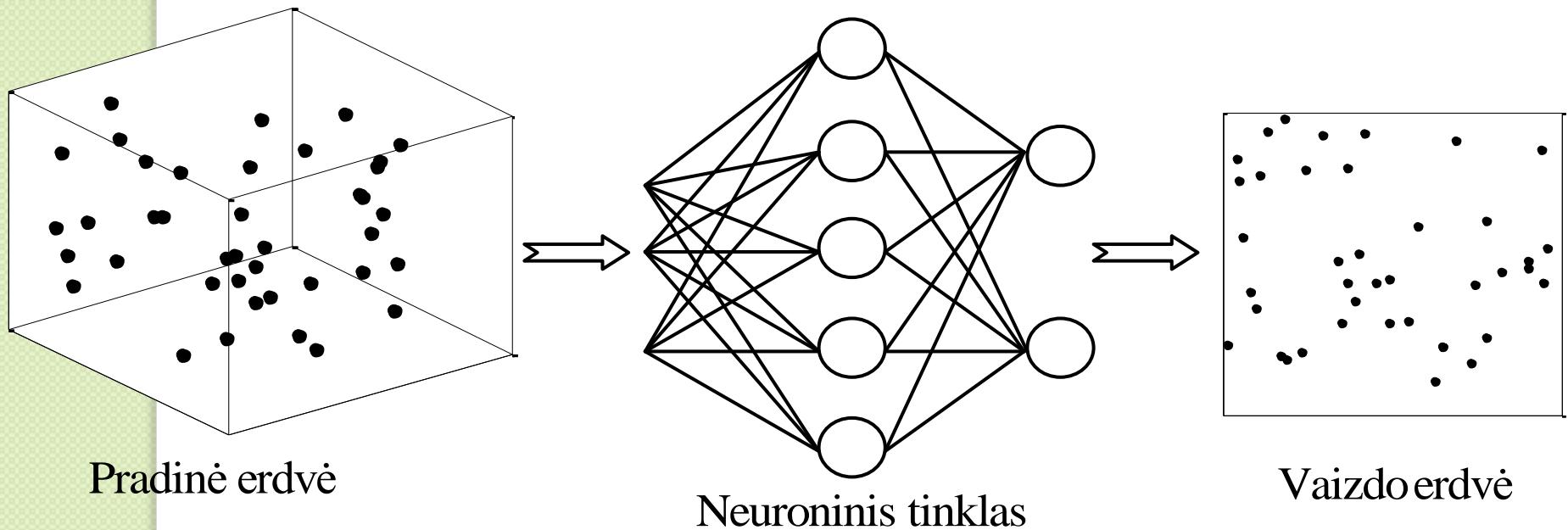


Panaudojimo galimybės

- Pirminės grandies gydytojams **ankstyvai diagnostikai nustatyti**

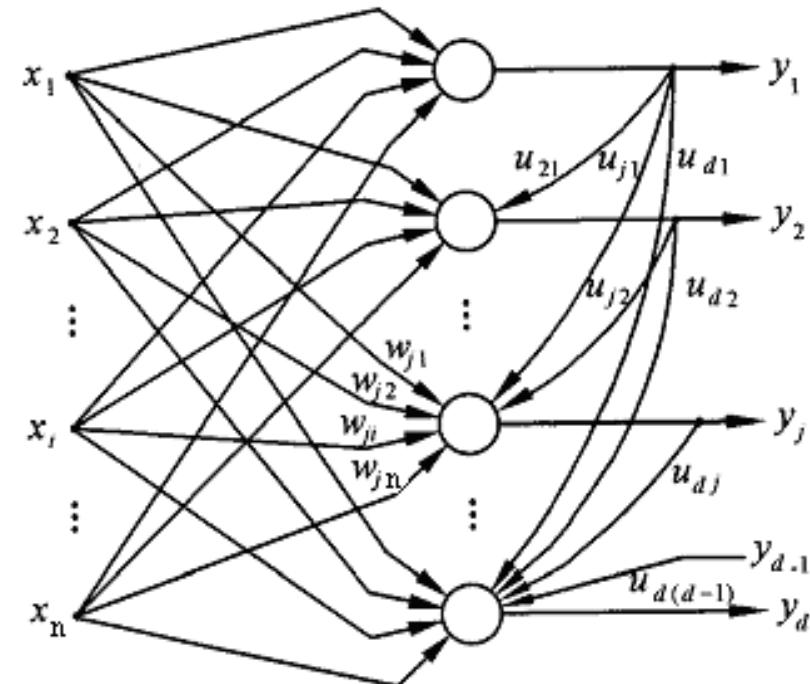
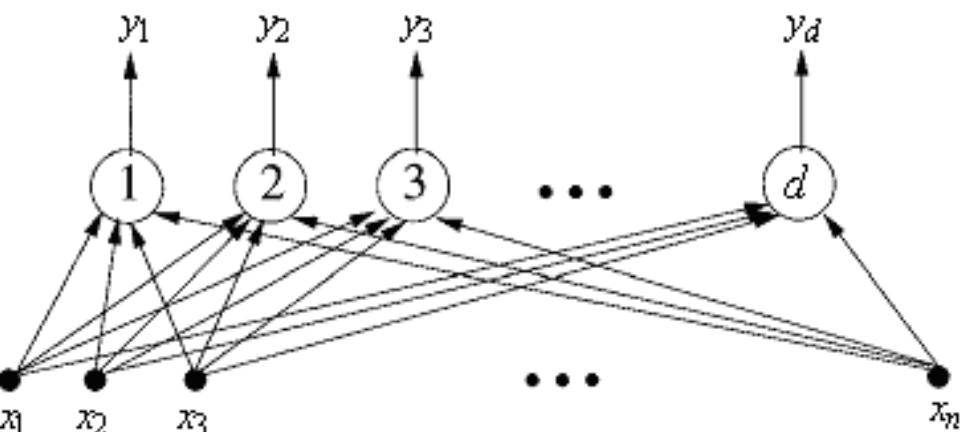


DNT daugiamatiams duomenims vizualizuoti (dimensijai mažinti)



DNT pagrindinėms komponentėms rasti

- **Hebbo** ir **Oja** mokymo taisyklės pagrindinėms komponentėms rasti



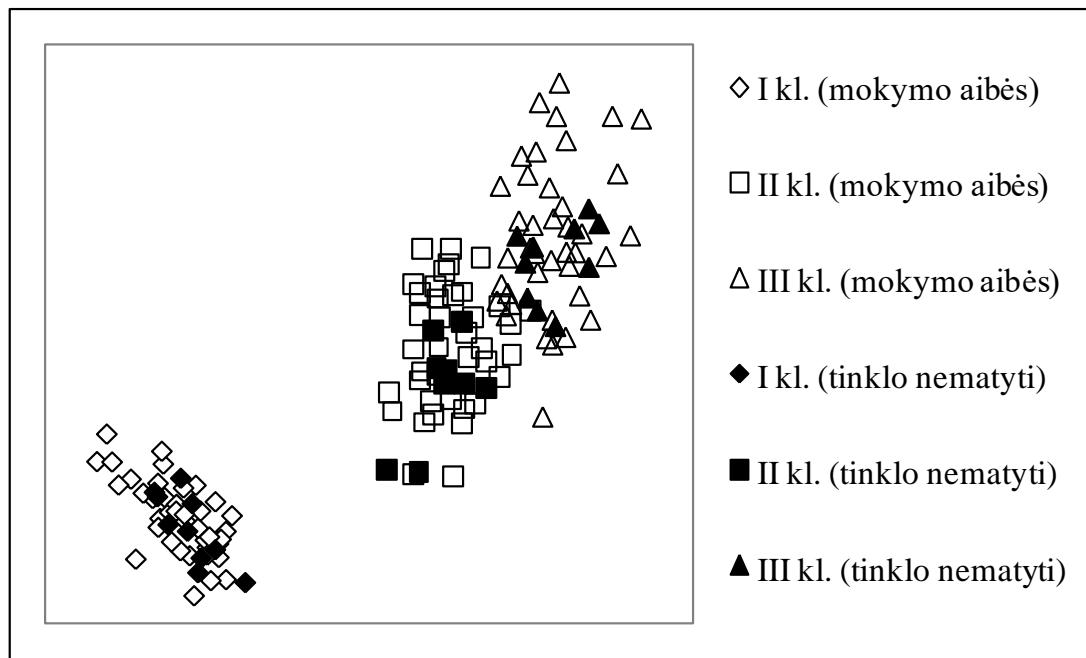
DNT ir daugiamatės skalės (1)

Daugamačių skalių tipo projekcija yra randama taikant **tiesioginio sklidimo neuroninę tinklą**, mokomą įprastiniu „**klaidos skleidimo atgal**“ algoritmu (mokymas su mokytoju).

- Pradžioje gaunamos taškų projekcijos **daugamačių skalių metodu**.
- Tinklas yra **apmokomas** duomenimis, sudarytais iš daugamačių taškų koordinačių, kai norimos išėjimų reikšmės yra **taškų projekcijos**, gautos daugiamatėmis skalėmis.

DNT ir daugiamatės skalės (1)

- Apmokius tinklą, iš jų **pateikiami daugiamatačiai taškai**, kurių projekcijos dar nėra žinomos.
- Tinklo išėjimuose gaunamos **taškų projekcijos** mažesnės dimensijos erdvėje.



Autoasociatyvieji neuroniniai tinklai

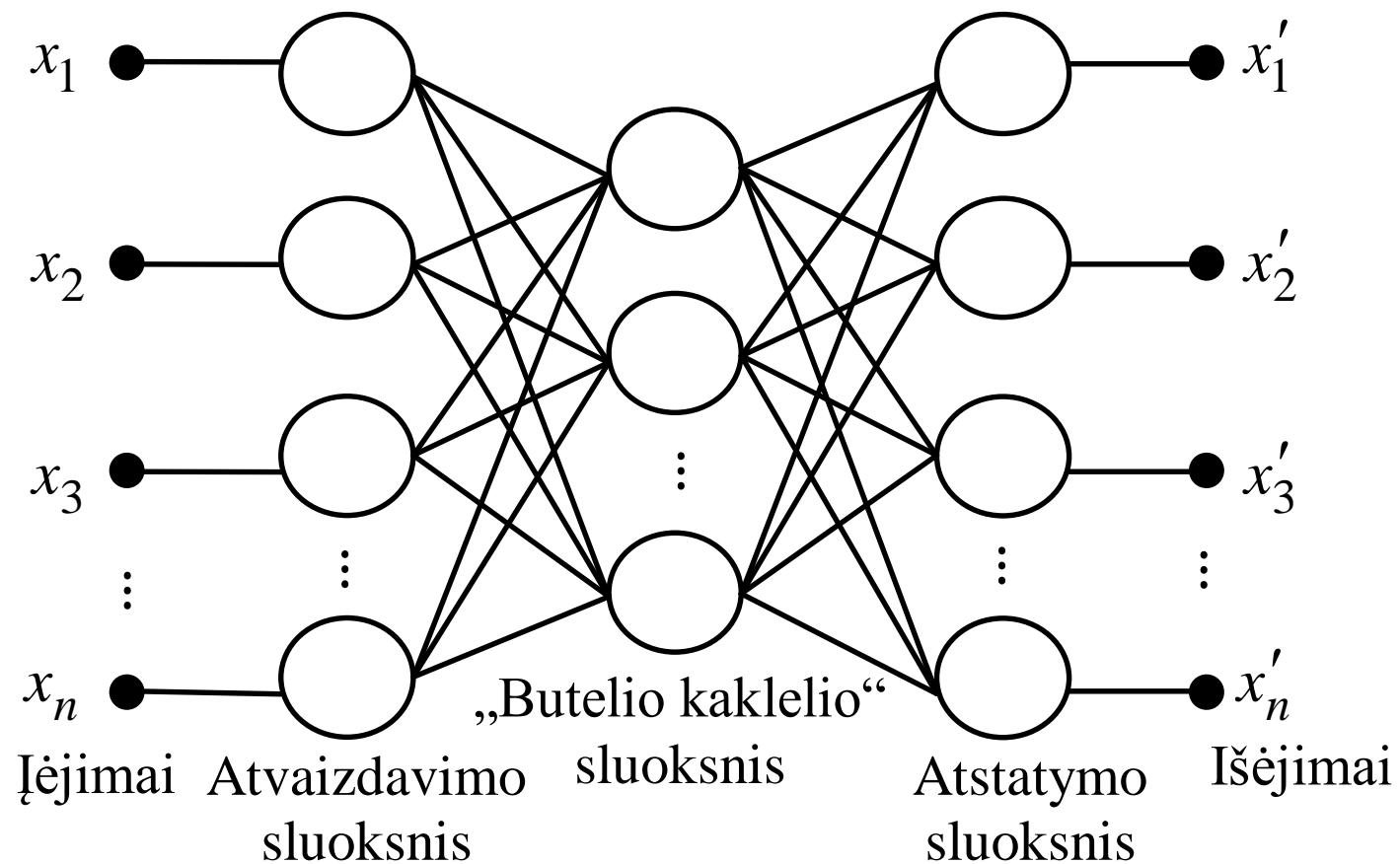
- **Autoasociatyvieji neuroniniai tinklai** (*autoassociative neural networks*) dar dažnai vadinami autokoderių tinklais.
- Jie naudojami matmenų skaičiui mažinti išskiriant d neuronų iš vadinamojo „**butelio kaklelio**“ (*bottleneck*) sluoksnio, sudaryto iš mažiau elementų nei įėjimo ir išėjimo sluoksniai, čia d yra vaizdo erdvės matmenų skaičius.

Autoasociatyvieji neuroniniai tinklai

Autoasociatyvusis neuroninis tinklas sudarytas iš **dviejų dalių**:

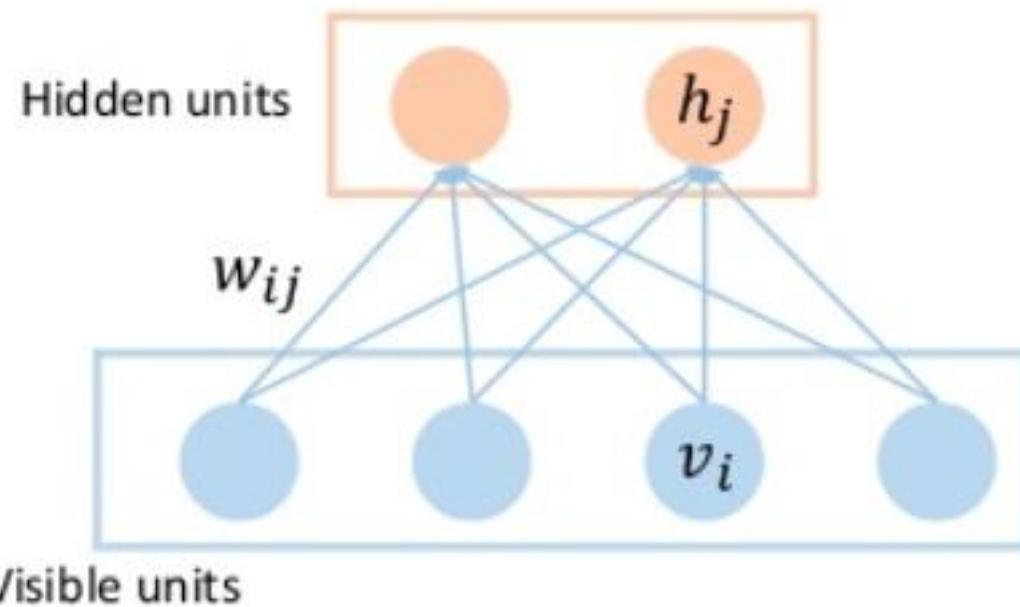
- pirma dalis transformuoja pradinius analizuojamus daugamačius duomenis į mažesnio skaičiaus matmenų erdvę (**atvaizdavimo sluoksnis**),
- o antroji – rekonstruoja (atstato) pradinius duomenis iš gautų projekcijų (**atstatymo sluoksnis**).

Autoasociatyvieji neuroniniai tinklai



Ribota Boltzmann mašina

- **Riboto Boltzmano mašinos** (*Restricted Boltzman Machine*) neuroninis tinklas duomenų dimensijai mažinti veikimas panašus į autoasociatyvius neuroninius tinklus.





Vilniaus universitetas
Matematikos ir informatikos fakultetas
Informatikos katedra



Saviorganizuojantys neuroniniai tinklai (žemėlapiai)

prof. dr. Olga Kurasova
Olga.Kurasova@mii.vu.lt

2018

Saviorganizuojantys neuroniniai tinklai (1)

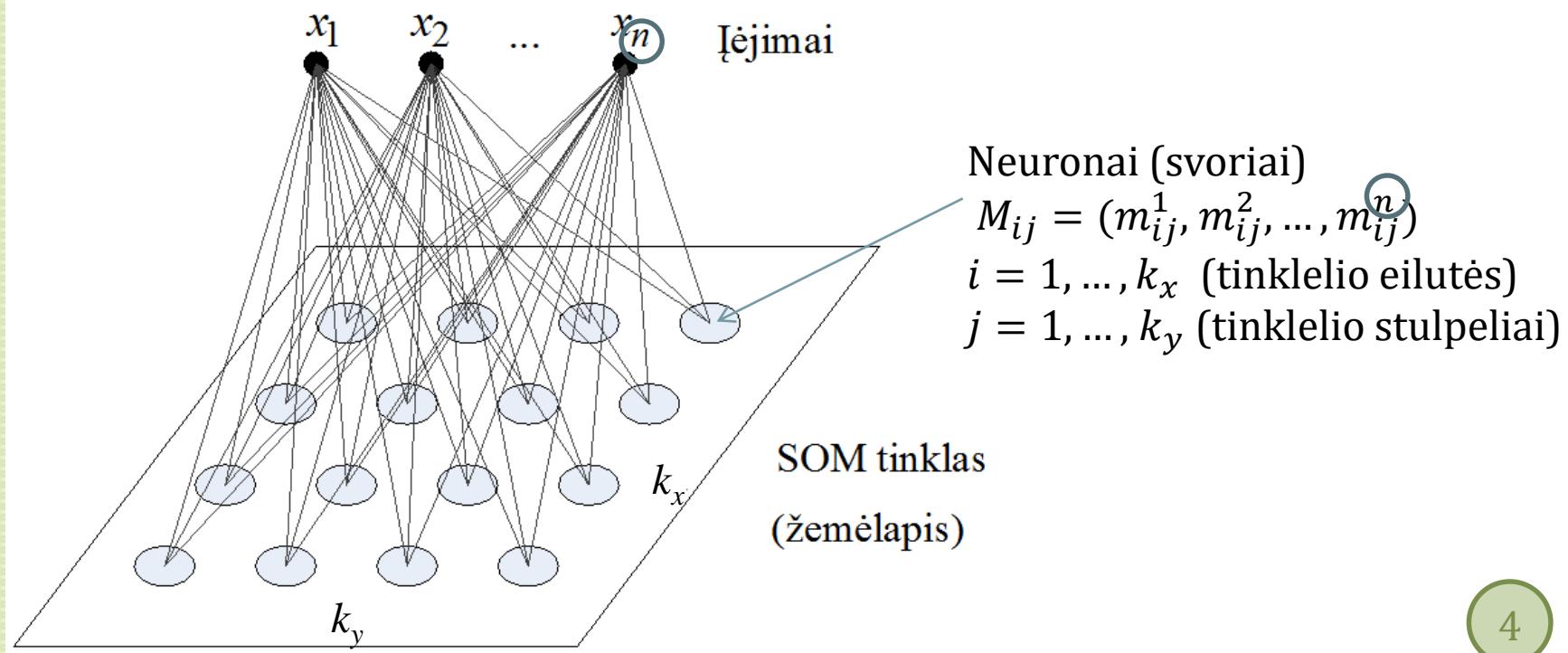
- **Saviorganizuojančius neuroninius tinklus** (žemėlapius, *self-organizing maps*, **SOM**) Suomijos mokslininkas T. Kohonenas pradėjo tyrinėti apie 1982 m., todėl jie dar vadinami Kohoneno neuroniniais tinklais, arba **Kohoneno saviorganizuojančiais žemėlapiais**.
- Iki šiol jie yra **nagrinėjami** daugelio pasaulio mokslininkų ir **plačiai taikomi** įvairiose srityse.
- **Pavadinimas kilo** iš to, kad saviorganizuojantis žemėapis, naudodamas mokymo (įėjimo) aibę, pats save sukuria (**save organizuoja**).

Saviorganizuojantys neuroniniai tinklai (2)

- Pagrindinis **SOM tinklo tikslas** – išlaikyti duomenų topologiją.
- Taškai, **esantys arti** įėjimo vektorių erdvėje, yra atvaizduojami **artis vieni kitų ir SOM** tinkle.
- SOM tinklai gali būti naudojami siekiant **vizualiai pateikti duomenų klasterius** (grupes) ir ieškant daugiamatičių duomenų **projekcijų** į mažesnio skaičiaus matmenų erdvę, paprastai į plokštumą.
- Todėl SOM yra ir **klasterizavimo**, ir **vizualizavimo** metodas.

Saviorganizuojantys neuroniniai tinklai (3)

Saviorganizuojantis neuroninis tinklas yra neuronų, paprastai išdėstyti **dvimačio tinklelio**, dar vadinamo **žemėlapiu** arba **lentele**, mazguose, **masyvas** $M = \{M_{ij}, i = 1, \dots, k_x, j = 1, \dots, k_y\}$.



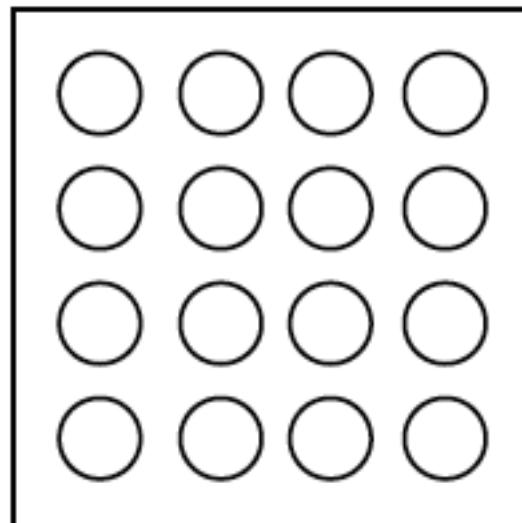
Saviorganizuojantys neuroniniai tinklai (4)

- Būtina suprasti, kad po kiekvienu SOM tinklo neuronu (paveiksle pažymėtu apskritimu) „**slepiasi“ vektorius** (*codebook vector*), kurio komponenčių skaičius sutampa su analizuojamų duomenų požymių skaičiumi.
- Priešingai nei anksčiau nagrinėti tiesioginio sklidimo neuroniniai tinklai, **nulinio įėjimo** SOM tinklas neturi.

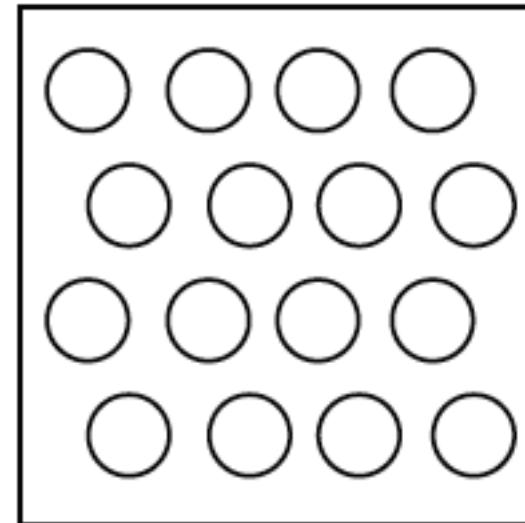
SOM tinklo struktūra

Galimos SOM tinklo struktūros:

- **stačiakampė** (*rectangular*) (a)
- arba **šešiakampė** (*hexagonal*) (b)



a)



b)

SOM tinklo mokymas (1)

- SOM tinklas mokomas **mokymo be mokytojo** būdu.
- Vektorių, apibūdinantį i -osios eilutės j -ajame stulpelyje esantį **neuroną**, pažymėkime $M_{ij} = (m_1^{ij}, m_2^{ij}, \dots, m_n^{ij})$
- Mokymo pradžioje neuronų (vektorius) M_{ij} komponenčių $m_1^{ij}, m_2^{ij}, \dots, m_n^{ij}$ **pradinės** reikšmės dažniausiai **nustatomos atsitiktinai** intervale $(0, 1)$.
- Neuroniniams tinklui **daug kartų pateikiama** skirtingu objektu, nusakomu n -mačiais vektoriais X_1, X_2, \dots, X_m .

SOM tinklo mokymas (2)

- Kiekviename mokymo žingsnyje (iteracijoje) **vienas** mokymo aibės **vektorius** X_k **pateikiamas** į tinklą.
- Vektorius X_k **palyginamas** su visais neuronais M_{ij} : dažniausiai skaičiuojamos **Euklido atstumas** tarp šio vektoriaus ir kiekvieno neurono ($\|X_k - M_{ij}\|$).
- Randama, iki kurio neurono $M_c \in \{M_{ij}\}$ atstumas yra mažiausias; rastas neuronas vadinamas **neuronu (vektoriumi) nugalėtoju** (*neuron (vector) winner*).

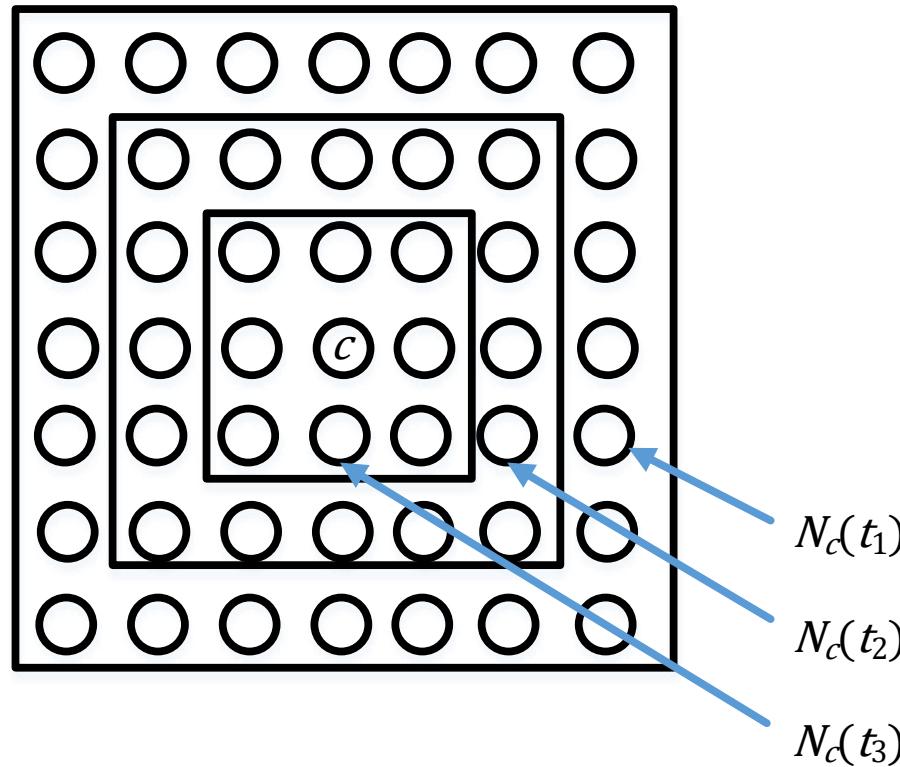
SOM tinklo mokymas (3)

- Visų tinklo neuronų **komponentės** keičiamos naudojantis iteracine formule:
$$M_{ij}(t + 1) = M_{ij}(t) + h_{ij}^c(X_k - M_{ij}(t))$$
- Čia c nurodo **neurono nugalėtojo indeksus** SOM žemėlapyje.
- t – **iteracijos** numeris,
- h_{ij}^c – **kaimynystės funkcija**, kuri yra mažėjanti funkcija ir artėjanti į 0, kai iteracijų skaičius artėja į begalybę; be to jos reikšmės **priklauso ir nuo neurono nugalėtojo vietas** perskaičiuojamo neurono atžvilgiu.

SOM kaimynystės funkcijos

- Galimos įvairios SOM **kaimynystės funkcijos** h_{ij}^c . Populiariosios šios:
 - Burbuliuko: $h_{ij}^c(t) = \begin{cases} \alpha(t), & (i,j) \in N_c \\ 0, & (i,j) \notin N_c \end{cases}$
 - Gauso: $h_{ij}^c(t) = \alpha(t) \cdot \exp\left(\frac{-\|R_c - R_{ij}\|^2}{2(\eta_{ij}^c(t))^2}\right)$
- Čia N_c yra kaimyninių neuronų indeksų aibė aplink neuroną su indeksu c . Dvimačiai vektoriai R_c ir R_{ij} yra neuronų M_c ir M_{ij} indeksai. Indeksai parodo SOM žemėlapyje esančio neurono vietą (eilutės ir stulpelio numerį).
- Parametras η_{ij}^c yra neurono M_{ij} kaimynystės eilės numeris neurono-nugalėtojo M_c atžvilgiu.

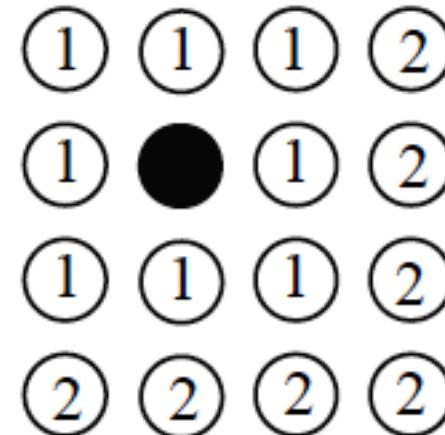
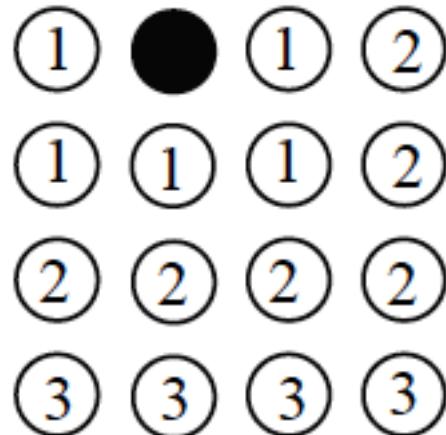
SOM kaimynystės aibės



Kaimynystės aibės $N_c(t_1), N_c(t_2), N_c(t_3)$, čia ($t_1 < t_2 < t_3$)

SOM kaimynystės eilė

- Greta vektoriaus nugalėtojo M_c esantys neuronai vadinami **pirmosios eilės kaimynais** (kaimynystės eilė $\eta_{ij}^c = 1$).
- Greta pirmosios eilės kaimynų esantys neuronai, išskyrus jau paminėtus, vadinami **antrosios eilės kaimynais** (kaimynystės eilė $\eta_{ij}^c = 2$) ir t. t.



SOM mokymo parametras

- Kaimynystės funkcija $\alpha(t)$ yra **mokymo parametras**.
- Dažniausiai:
 - $\alpha(t) = \left(1 - \frac{t}{T}\right)$
 - $\alpha(t) = \frac{1}{t}$
 - $\alpha(t) = (0,005)^{\frac{t}{T}}$
- T yra iteracijų (epochų) skaičius.

```

function SOM_training( $X, M, \hat{e}, k_x, k_y$ )
// įvestis:  $X$  – duomenų aibė,  $M$  – pradiniai neuronai,  $\hat{e}$  – tinklo mokymo epochų skaičius,
//  $k_x, k_y$  – eilučių ir stulpelių skaičius
// išvestis:  $M$  – neuronai
BEGIN
FOR  $t=1$  TO  $\hat{e}$ 
    FOR  $l=1$  TO  $m$  // duomenų aibės vektorius  $X_l$  pateikiamas į neuroninį tinklą
        FOR  $i=1$  TO  $k_x$ 
            FOR  $j=1$  TO  $k_y$ 
                
$$\|M_{ij} - X_l\| := \sqrt{\sum_{p=1}^n (m_p^{ij} - x_{lp})^2}$$
 // skaičiuojamas Euklido atstumas
            END
        END
         $c := \arg \min_{i,j} \|X_l - M_{ij}\|$  //  $\hat{M}_c$  – vektoriaus  $X_l$  neuronas nugalėtojas
        FOR  $i=1$  TO  $k_x$ 
            FOR  $j=1$  TO  $k_y$ 
                
$$M_{ij}(t+1) := M_{ij}(t) + h_{ij}^c(t)(X_l - M_{ij}(t))$$
 // SOM mokymo taisykla
            END
        END
    END // visų vektorių peržiūrėjimo pabaiga
END // mokymo pabaiga
RETURN  $M$ 
END

```

SOM tinklo rezultatas (1)

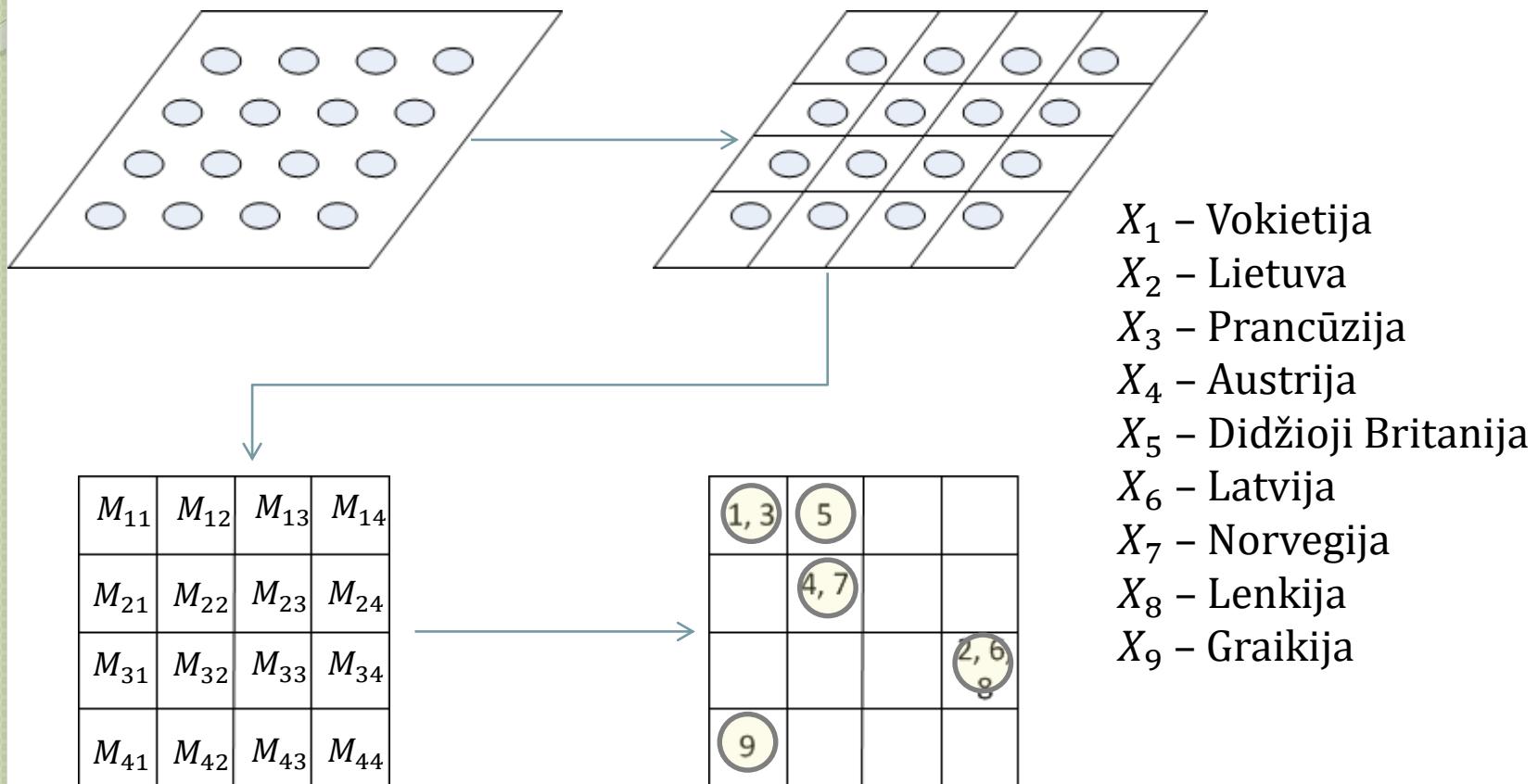
- Po **SOM** **tinklo** mokymo į tinklą pateikiami mokymo aibės arba nauji, dar tinklui „nematyti“, duomenų vektoriai.
- Randamas kiekvieno vektoriaus **neuronas nugalėtojas** ir jis pažymimas SOM žemėlapyje neurono nugalėtojo vietoje.
- Tokiu būdu vektoriai **išsidėsto tarp žemėlapio** (lentelės) elementų.

SOM tinklo rezultatas (2)

Irisų duomenys, pateikti **SOM** tinkle [10 x 10]. Čia pavaizduojami klasių numeriai, bet gali būti pavaizduojami ir duomenų numeriai.

3		3	3	2				1	1
3				2				1	1
3	3		2	2				1	1
3	3	3		2					1
3		3		2					1
		3	2	2	2	2			
3	3	3			2				
2,3	3			2		2			2
2				2		2	2		2
3		2	2	3	2	2	2	2	2

SOM rezultatas



- neuronai-nugalėtojai

SOM tinklo kokybės nustatymas (1)

- Baigus SOM tinklo mokymus, būtina nustatyti jo kokybę.
- Dažniausiai vertinamos dvi paklaidos:
 - **kvantavimo** (*quantization error*)
 - ir **topografinė** (*topographic error*).
- **Kvantavimo paklaida** parodo, kaip tiksliai jau išmokyto tinklo neuronai prisiderina prie mokymo aibės vektorių.
- Tai **vidutinis atstumas** tarp duomenų vektorių ir jų vektorių nugalėtojų:

$$E_{SOM(q)} = \frac{1}{m} \sum_{k=1}^m \| X_k - M_{c(k)} \|$$

SOM tinklo kokybės nustatymas (2)

- **Topografinė paklaida** parodo, kaip gerai SOM tinklas išlaiko analizuojamų duomenų **topografiją**, t. y. tarpusavio išsidėstymą.
- Ji skaičiuojama pagal formulę:

$$E_{SOM(t)} = \frac{1}{m} \sum_{k=1}^m u(X_k)$$

- Jeigu SOM žemėlapyje vektoriaus X_k neuronas nugalėtojas **yra šalia neurono**, iki kurio atstumas nuo X_k yra mažiausias, neskaičiuojant iki neurono nugalėtojo, tai formulėje $u(X_k) = 0$, priešingu atveju $u(X_k) = 1$.

SOM vizualizavimo būdai

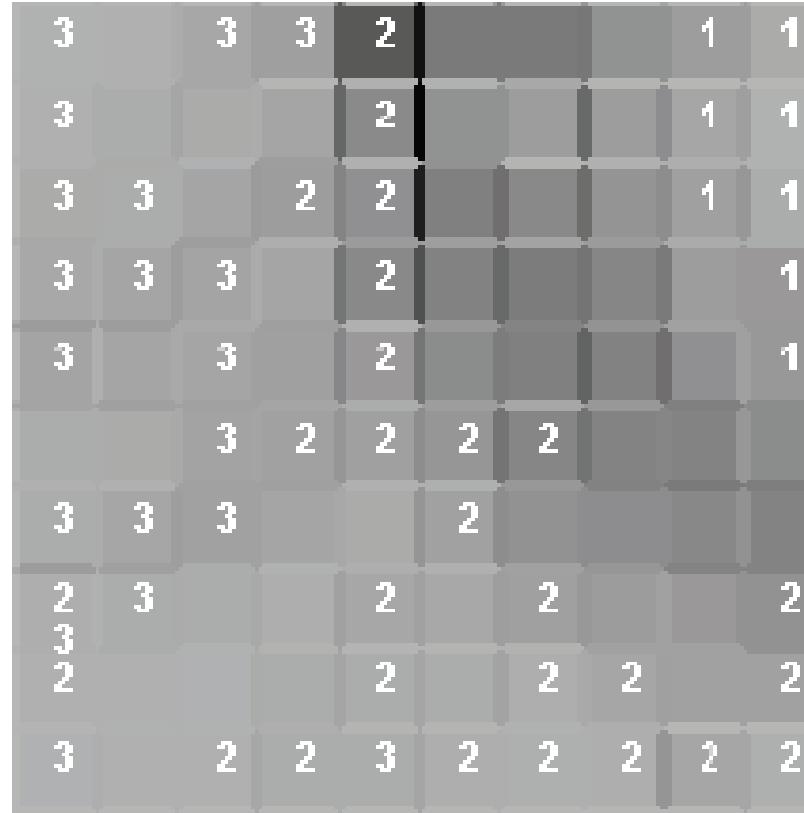
- Paprasčiausia SOM lentelė nėra labai informatyvi, **sunku pasakyti**, kaip toli yra duomenys, esantys gretimuose SOM lentelės langeliuose.
- Todėl **būtina ieškoti** būdu, kaip pagerinti tokio vizualizavimo kokybę.
- **Unifikuota atstumų matrica** (*U-matrica, unified distance matrix*) yra vienas populiariausių SOM vizualizavimo būdu.
- U-matricą sudaro atstumai tarp **kaimynynių** SOM neuronų.

U-matrica

- Paprastumo dėlei nagrinėkime **vienmatį SOM** [1x5] (M_1, M_2, \dots, M_5).
- **U-matrica** bus vienos eilutės ir devynių stulpelių ($u_1, u_{12}, u_2, u_{23}, u_3, u_{34}, u_4, u_{45}, u_5$).
- Čia $u_{ij} = ||M_i - M_j||$ yra **atstumas** tarp kaimyninių neuronų M_i ir M_j , o u_i yra tam tikra reikšmė, pavyzdžiui, vidutinis atstumas tarp kaimyninių reikšmių

$$u_i = \frac{u_{(i-1)i} + u_{i(i+1)}}{2}$$

Irisų duomenys, vizualizuoti u-matrica



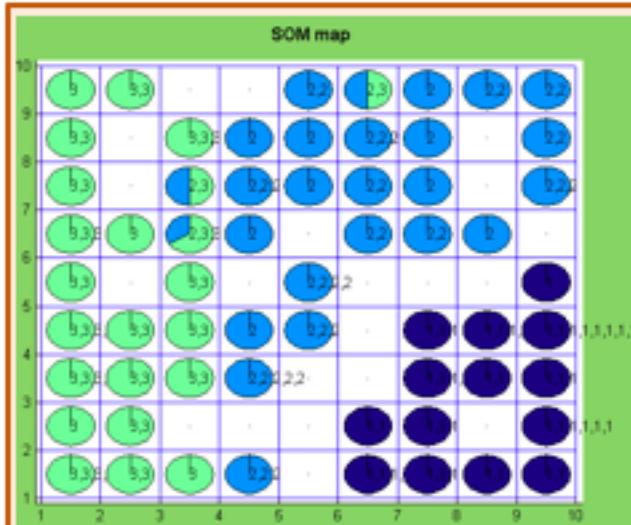
Irisai įvairiai vizualizavimo būdais

3		3	3	2				1	1
3				2				1	1
3	3		2	2				1	1
3	3	3		2				1	
3		3		2				1	
		3	2	2	2	2			
3	3	3			2				
2,3	3			2		2			2
2				2		2	2		2
3		2	2	3	2	2	2	2	2

paprasčiausia SOM lentelė

3	3	3	2				1	1
3			2				1	1
3	3		2	2			1	1
3	3	3		2			1	
3		3		2			1	
		3	2	2	2	2		
3	3	3			2			
2,3	3			2		2		2
2				2		2	2	2
3		2	2	3	2	2	2	2

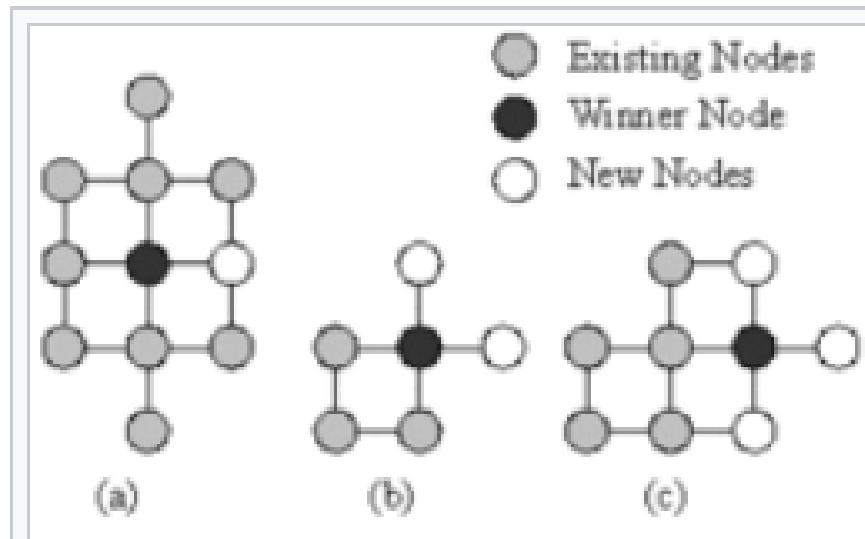
SOM lentelė vizualizuota U-matricos pagalba



SOM lentelė su skritulinėmis diagramomis

SOM praplētimai

Augantis SOM (*growing self-organizing map, GSOM*).



Node growth options in GSOM: (a) one new node, (b) two new nodes and (c) three new nodes.

SOM praplėtimai

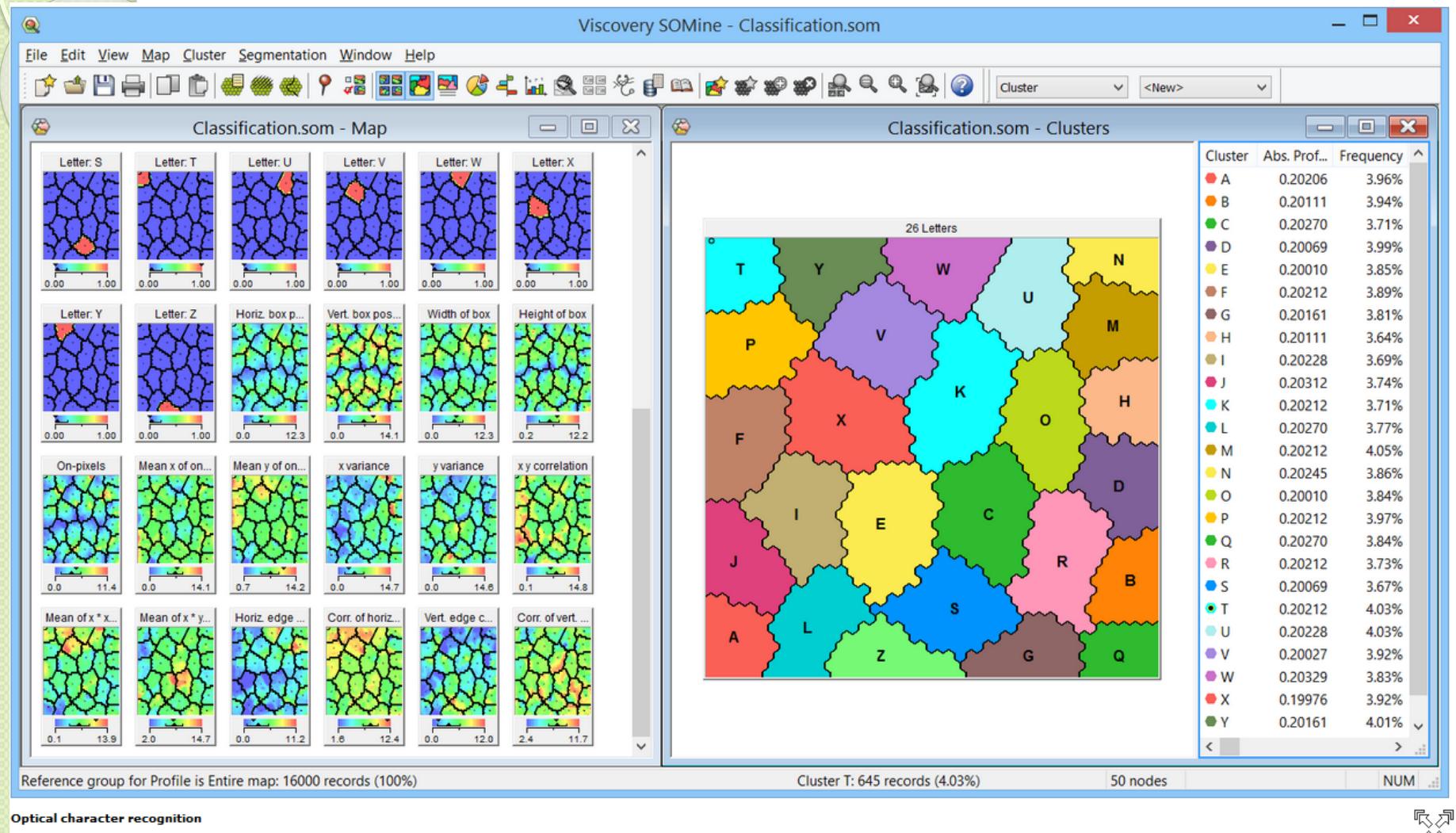
- **Laike prisitaikantis SOM** (*time adaptive self-organizing map, TASOM*), kur kiekvienas neuronas turi prisitaikantį savo mokymo parametrą ir kaimynystės dydį.
- **Generatyvinis topografinis žemėlapis** (*generative topographic map, GTM*) – tai alternatyva SOM'ui.

Viscovery SOMine



- **Viscovery SOMine** is a workflow-oriented software suite based on **self-organizing maps** (SOM) and multivariate statistics for explorative data mining and predictive modeling.
- Main functions and features:
 - **Creation** of self-organizing map models using predefined schedules
 - Interactive **SOM visualization** and exploration
 - Visual cluster analysis with integrated visualization of **cluster boundaries** and inner structures
 - **Statistical functions**, such as descriptive statistics, histograms, correlations, PCA, and scatter plots

Viscovery SOMine



From: <https://www.viscovery.net/somine/>



Vilniaus universitetas
Matematikos ir informatikos fakultetas
Informatikos katedra



Radialinių bazinių funkcijų neuroniniai tinklai

prof. dr. Olga Kurasova
Olga.Kurasova@mii.vu.lt

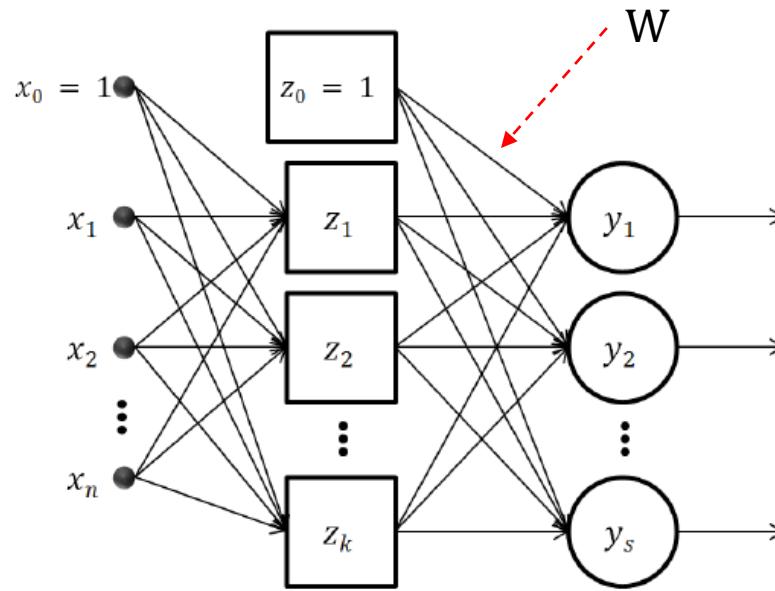
2018

Radialinė bazine funkcija

- **Radialinė bazine funkcija** (RBF) – tai funkcija, kurios reikšmės priklauso tik nuo atstumo nuo pradžios taško, t. y. $\phi(X) = \phi(\|X\|)$ arba iki kito taško μ , vadinamojo centro, t. y. $\phi(X, \mu) = \phi(\|X - \mu\|)$.
- Iprastai norma yra **Euklidinis atstumas**, tačiau gali būti naudojami ir kiti atstumai.

Radialinių bazinių funkcijų (RBF) neuroniniai tinklai

- **Radialinių bazinių funkcijų neuroninis tinklas** (angl. *Radial Basis Function Neural Network, RBF*) padeda išspręsti funkcijų aproksimavimo, laiko eilučių prognozavimo, klasifikavimo ir kitus duomenų analizės uždavinius.



RBF tinklo sandara

- Tinklas susideda iš n **įėjimų**, vieno **paslėpto** neuronų sluoksnio, kuris sudarytas iš k neuronų ir s **išėjimų**.
- Įėjimo duomenų rinkinys $X = (x_0, x_1, x_2, \dots, x_n)$.
- Paslėptas neuronų sluoksnis $Z = (z_0, z_1, z_2, \dots, z_k)$. Šiame sluoksnyje vietoj aktyvavimo funkcijų yra naudojamos **radialinės bazinės funkcijos**, todėl šis sluoksnis dar yra vadintinas radialinių bazinių funkcijų sluoksniu.
- Išėjimų sluoksnis žymimas $Y = (y_1, y_2, \dots, y_s)$.

Radialinės bazine funkcijos

- Radialinių bazinių funkcijų **sužadinimo lygis** priklauso nuo atstumo tarp objekto $X_i = (x_{i1}, x_{i2}, \dots, x_{in})$, $i = 1, \dots, m$, ir **radialinės bazine funkcijos centro** $\mu_j = (\mu_{j1}, \mu_{j2}, \dots, \mu_{jn})$, $j = 1, \dots, k$.
- Bendra radialinių bazinių funkcijų **išraiška** užrašoma taip:

$$z_j = \phi(\|X - \mu_j\|)$$

- Čia $\|X - \mu_j\|$ – atstumas tarp objekto X_i ir centro μ_j , dažniausiai skaičiuojamas **Euklidinis atstumas**, bet gali būti skaičiuojamas ir bet kuris kitas atstumas.

Radialinės bazinės funkcijos

- **Gausinė:**

$$z_j = \exp\left(-\frac{\|X - \mu_j\|^2}{2\sigma^2}\right)$$

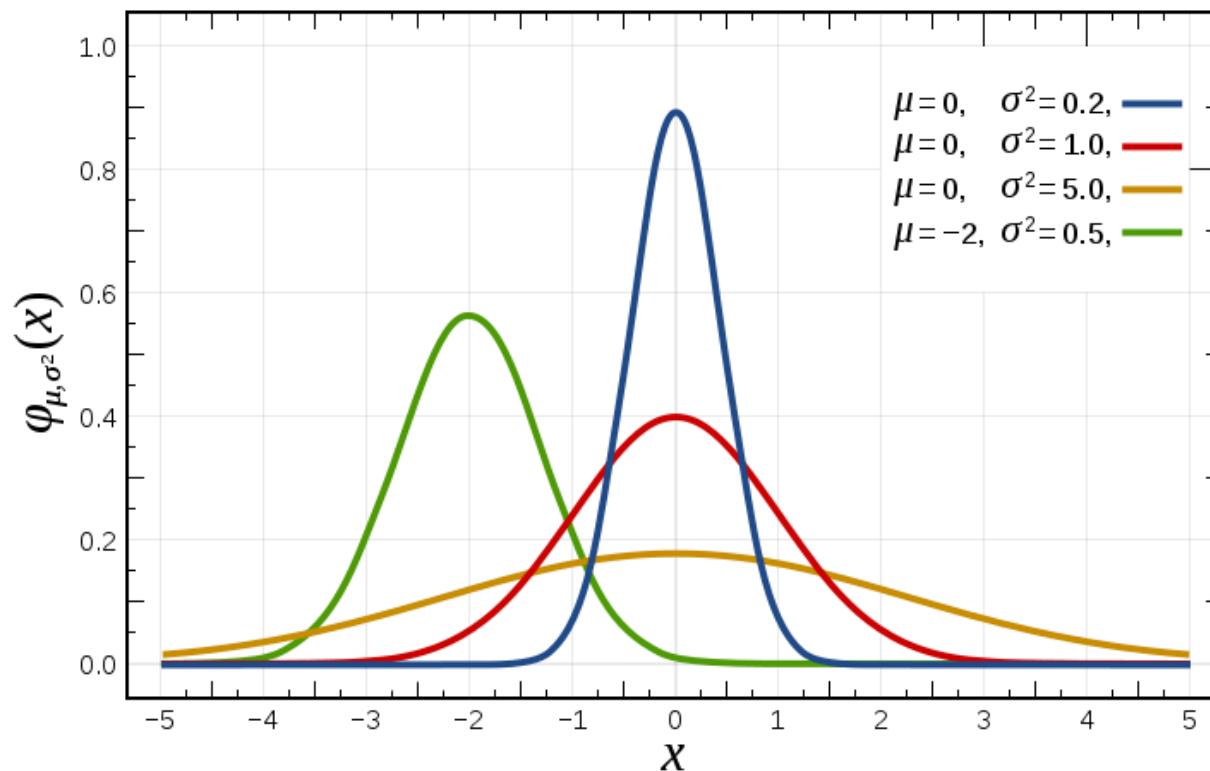
čia σ – pločio parametras.

- **Multikvadratinė:** $z_j = \sqrt{\|X - \mu_j\|^2 + 1}$
- **Multikvadratinė inversija:** $z_j = \frac{1}{\sqrt{\|X - \mu_j\|^2 + 1}}$

Gausinė funkcija

- Dažniausiai RBF neuroniniuose tinkluose yra naudojama **Gausinė** radialinė bazine funkcija.

$$z_j = \exp\left(-\frac{\|X - \mu_j\|^2}{2\sigma^2}\right)$$



RBF tinklo mokymas

Galimi keli RBF tinklo mokymo būdai:

- **Vieno** etapo,
- **Dviejų** etapų,
- **Trijų** etapų.

Vieno etapo RBF tinklo mokymas

- Mokymo metu nustatomi **antrojo sluoksnio svoriai** W mokymo su mokytoju (*supervised*) būdu.
- RBF **centro taškai** μ_j parenkami iš mokymo duomenų aibės,
- **Parametras** σ prilyginamas iš anksto parinktam skaičiui.

Dviejų etapų RBF tinklo mokymas

- Abu RBF tinklo sluoksniai nustatomi atskirai.
- Pradžioje nustatomi RBF **centro taškai** μ_j ir **parametras** σ reikšmė.
- Vėliau randami **antrojo sluoksnio svoriai** W .
- Tai **dažniausiai** naudojamas būdas.

Trijų etapų RBF tinklo mokymas

- Įvykdomas **dviejų etapų** mokymas.
- Vėliau **visa** tinklo struktūra (μ_j, σ, W) suderinama atlikus tam tikrą **optimizavimo** procedūrą.
- Iprastai optimizavimui reikalingi **pradinės kintamujų reikšmės**. Tad čia jo prilyginamos rastoms po dviejų etapų mokymo.

RBF tinklo mokymas

- RBF tinklai įprastai **apmokomi dviem etapais**.
- **Pirmame etape** nustatomi RBF parametrai – centro taškai μ_j ir pločio parametras σ .
- Nustačius šiuos parametrus, RBF reikšmės tampa fiksuotos, todėl likusi tinklo dalis yra ekvivalentiška **vienasluoksniam perceptronui**.
- Todėl **antrame etape** tinklo mokymas vyksta minimizuojant paklaidos funkciją gradientiniu nusileidimo algoritmu (keičiami tinklo svoriai).

RBF tinklo mokymas

- RBF centro taškai μ_j gali būti parenkami dvejopai:
- Paprasčiausiu atveju, **pasirenkami atsitiktinai** nagrinėjamos duomenų aibės taškai ir jie laikomi centro taškais.
- Kitas būdas – juos parinkti taikant **k-vidurkių** klasterizavimo algoritmą.
- Yra ir kitų centru parinkimo būdų.

K-vidurkių algoritmas (1)

- 1) Pasirenkama, **į kiek klasterių** k bus klasterizuojama.
- 2) **Atsitiktinai** parenkami klasterių **centrų taškai**.
- 3) Visi duomenų taškai **priskiriami vienam iš klasterių** pagal jų artumą klasterių centru.
- 4) Perskaičiuojami **klasterių centrali** ir skaičiuojama kvadratinė paklaida (žr. sekančią skaidrę).
- 5) Jei netenkinama **sustojimo salyga**, einama į 3 žingsnių. Priešingu atveju, algoritmas sustabdomas.

K-vidurkių algoritmas (2)

- Vienas iš galimų duomenų panašumo matų yra **kvadratinė paklaida**.
- Tai **Euklido atstumu** tarp kiekvieno klasterio objekto ir klasterio centro kvadratų suma visiems klasteriams:

$$E_{k\text{-means}} = \sum_{j=1}^k \sum_{i=1}^{k_j} \|X_i^j - \mu_j\|^2$$

- Čia k – klasterių skaičius, k_j – j -tajam klasteriui priskirtų duomenų taškų skaičius.
- $\{X_1^j, X_2^j, \dots, X_{k_j}^j\}$ – j -tajam klasteriui priskirtų duomenų taškų aibė.

K-vidurkių algoritmas (3)

K-vidurkių metodo **sustojimo sąlygos**:

- kvadratinės paklaidos reikšmė $E_{k\text{-means}}$ tampa **mažesnė** už pasirinktą slenkstinę reikšmę (norimą mažą skaičių)
- arba duomenų taškai **nebepersiskirsto** kitiems klasteriams.

RBF tinklo mokymas

- Kai centro taškui μ_j ir parametrui σ nustatyti taikomas **k-vidurkių** (k-means) algoritmas, pasirenkama **į kelis klasterius** bus klasterizuojama, priklausomai nuo RBF tinklo struktūros (kiek neuronų pirmame sluoksnyje).
- Visi duomenys k-vidurkių algoritmu **suklasterizuojami** į tiek klasterių. $\{X_1^j, X_2^j, \dots, X_{k_j}^j\}$ – j -tajam klasteriui priskirtų duomenų taškų aibė.
- Gautų **klasterių centrai** tampa centro taškais μ_j .
- Parametras σ gali būti parenkamas kiekvienam klasteriui. Tai gali būti **standartinis nuokrypis**

$$\sigma_j = \frac{1}{k_j} \sum_{i=1}^{k_j} \|X_i^j - \mu_j\|.$$

RBF tinklo mokymas

- Radus centrų taškus ir σ reikšmes, visiems mokymo duomenims **apskaičiuojamos RBF reikšmės**. Gaunama matrica Z .
- Turinti norimą reikšmių vektorių T , **antrojo sluoksnio svoriai W** randami minimizuojant paklaidą:

$$E(W) = \|ZW - T\|^2$$

- Svoriai W gali būti gauti **analitiškai**: $W = Z^+T$, čia Z^+ yra matricos Z pseudo-atvirkštinė matrica, kurios paprastesnė išraiška yra: $Z^+ = (Z^T Z)^{-1} Z^T$.

RBF taikymas duomenims klasifikuoti

- Išėjimų skaičius lygus klasių skaičiui.
- Duomenys suklasterizuojami į pasirinktą k klasterių skaičių.
- Apskaičiuojamos σ reikšmės.
- Visi mokymo duomenys „praleidžiami“ per RBF, pvz., Gausinę funkciją.

RBF taikymas duomenims klasifikuoti

- Tuomet **išėjimų skaičius** lygus klasų skaičiui.
- Duomenys **suklasterizuojami** į pasirinktą k klasterių skaičių, taip randami RBF centro taškai.
- Apskaičiuojamos σ reikšmės (klasterių **standartiniai nuokrypiai**).
- Visi mokymo duomenys „**praleidžiami**“ per RBF, pvz., Gausinę funkciją.
- Randami antrojo sluoksnio **svoriai** W .
- **Nauji duomenys** pateikiami į išmokyta tinklą. Pagal gautas išėjimų reikšmes yra **nustatomos klasės**.
- Patikrinamas **klasifikavimo tikslumas**, t. y., kiek duomenų yra teisingai suklasifikuota.

RBF taikymas funkcijoms aproksimuoti

- I **įėjimus** bus pateikiamos aproksimuojamos funkcijos kintamųjų reikšmės, norimos **išėjimų reikšmės** bus funkcijos reikšmės.
- Pasirenkamas RBF išėjimų skaičius k .
- Pasirenkama paramетro σ reikšmė.
- Taškai, sudaryti iš kintamųjų reikšmių, **suklasterizuojami** į k klasterių.
- Visi taškai „**praleidžiami**“ per RBF, pvz., Gausinę funkciją.
- Randami antrojo sluoksnio **svoriai** W .
- **Taškai** pateikiami į išmokytaą tinklą. Pagal gautas išėjimų reikšmes yra **nustatomos funkcijų reikšmės**.

RBF tinklo taikymo pavyzdžiai

- Paprastus RBF tinklo taikymo pavyzdžius galima rasti adresu:

<http://mccormickml.com/2013/08/15/radial-basis-function-network-rbfn-tutorial/>