Vilnius universitetas

Matematikos ir informatikos fakultetas

Duomenų mokslo ir skaitmeninių technologijų institutas

Dirbtinis neuronas - perceptronas

prof. dr. Olga Kurasova

Olga.Kurasova@mif.vu.lt

Dirbtiniai neuroniniai tinklai (DNT)

- Viena iš skaitmeninio 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.

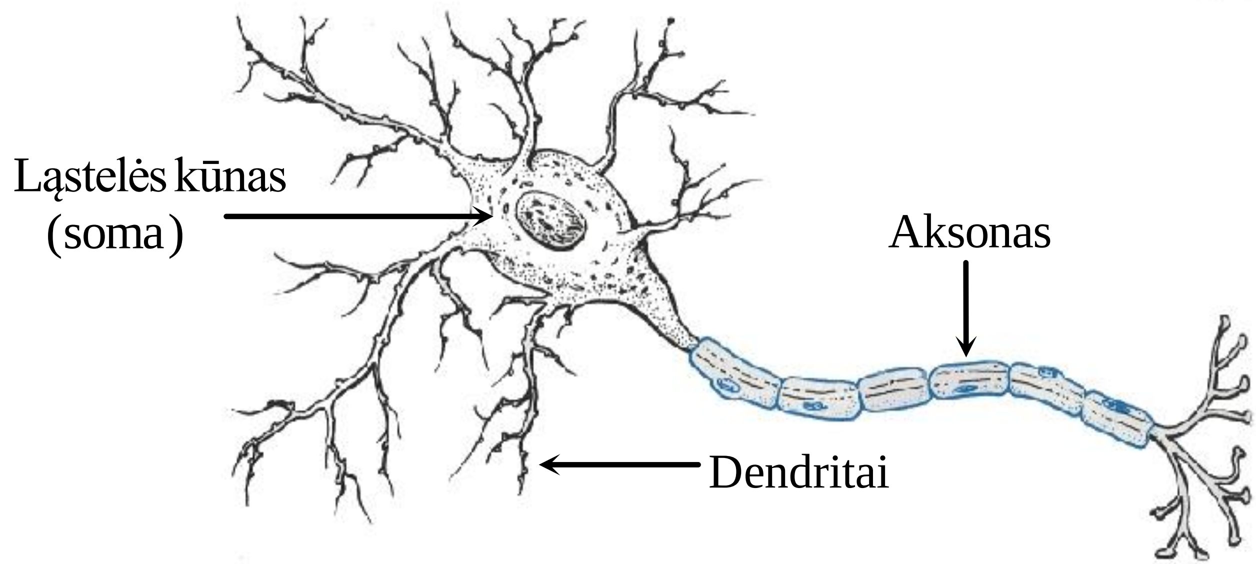
Biologinis neuronas

- Žmogaus **smegenys** susideda iš daugelio (apie 86 milijardus, $\sim 10^{11}$) **neuronų**, sujungtų vienų su kitais.
- Kiekvienas neuronas turi vidutiniškai keletą tūkstančių **jungčių**.
- **Neuronas** – tai ląstelė, galinti generuoti elektrocheminį signalą.

Biologinis neuronas

Neuronas turi

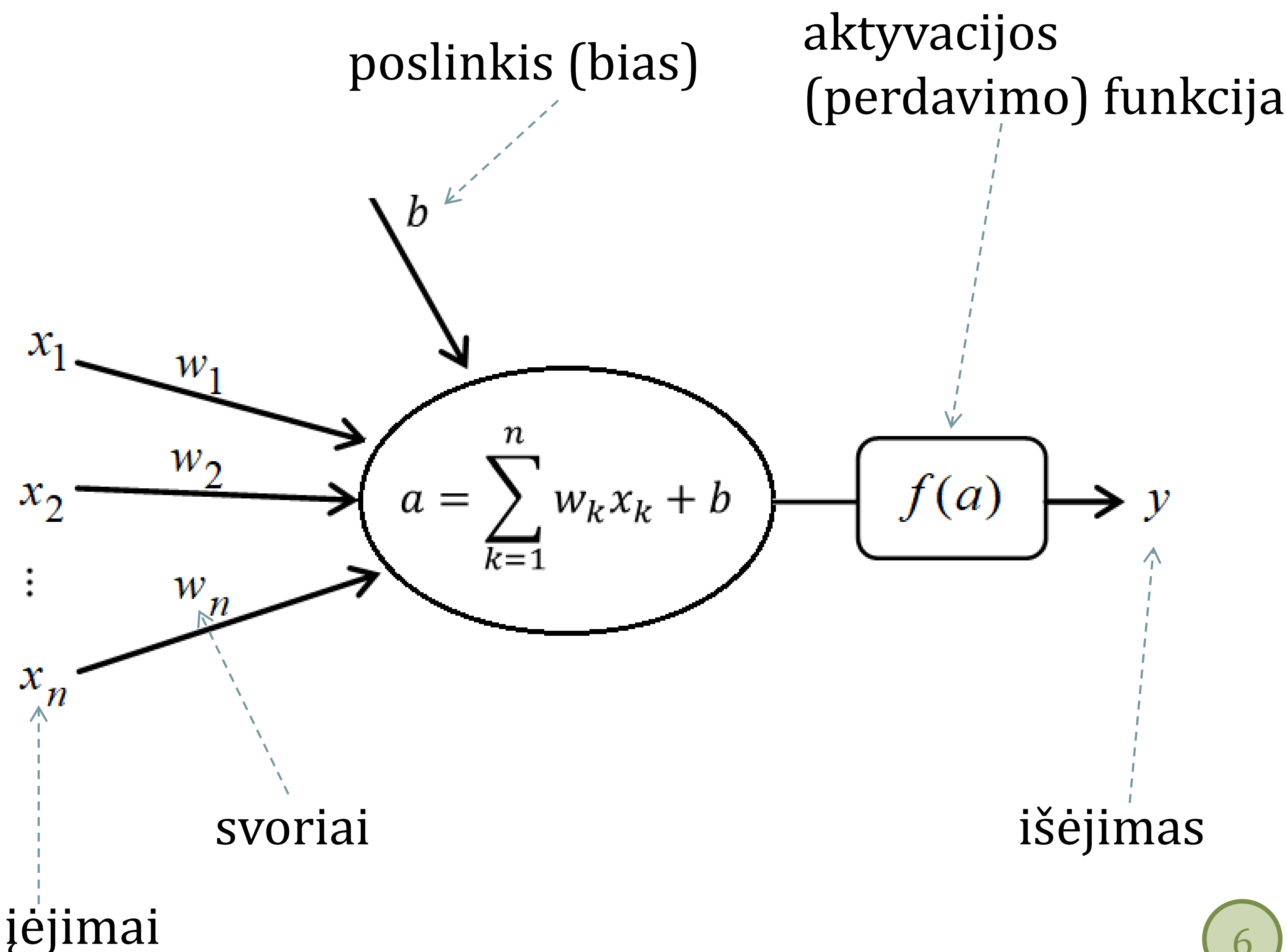
- išsišakojusią įėjimo struktūrą, vadinamuosius **dendritus**,
- ląstelės kūną, vadinamąją **somą**,
- ir besišakojančią išėjimo struktūrą – **aksoną**.



Biologinis neuronas

- Vienos ląstelės aksonas su kitos ląstelės 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, vadinamąjį **sužadinimo slenkstį**.
- Turint **didžiulį skaičių visiškai paprastų elementų**, kurių kiekvienas skaičiuoja svorinę įeinančių signalų sumą ir generuoja binarųjį signalą, jei suminis signalas viršija tam tikrą lygį, **galima atlikti gana sudėtingas užduotis**.

Dirbtinio neurono modelis



Dirbtinio neurono modelis

- Neuronas turi keletą **įėjimų** $x_1, x_2, \ldots, x_n$ (*inputs*).
- Kiekviena įėjimo x_k, $k = 1, \ldots, n$, **jungtis** turi savo perdavimo koeficientą (**svorį**) w_k, $k = 1, \ldots, n$.
- Įprastai įėjimų ir jungčių svorių reikšmės yra **realieji skaičiai**.
- Skaičiuojama įėjimo reikšmių ir svorių sandaugų **suma**, prie kurios dar pridedamas **poslinkis** (*bias*) b.

$$a = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b = \sum_{k=1}^{n} w_kx_k + b$$

Dirbtinio neurono modelis

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

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

kurios reikšmė vadinama **neurono išėjimo reikšme** (*output*).

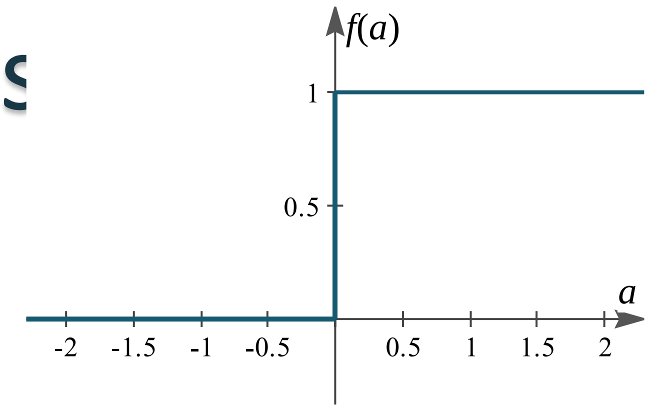
- Paprasčiausia aktyvacijos funkcija yra **slenkstinė**:

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

Aktyvacijos funkcijos

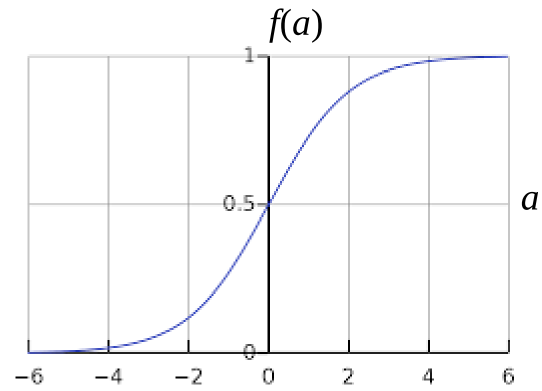
- Slenkstinė

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



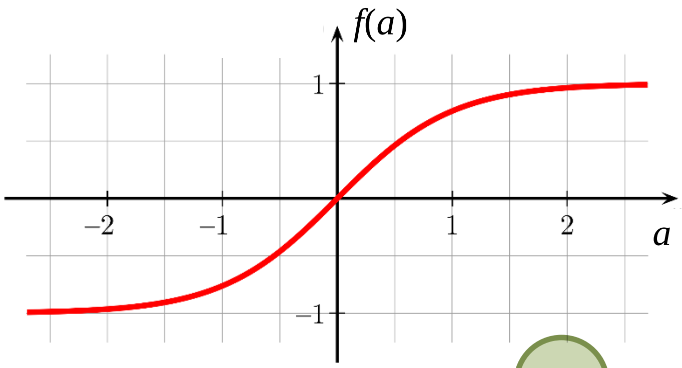
- Sigmoidinė (logistinė)

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



- Hiperbolinis tangentas

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



Aktyvacijos funkcijos

- **Keblumai** su sigmoidinės ir logistinės funkcijų terminais.
- **Sigmoidinė funkcija** – tai matematinė funkcija, kuriai būdinga „S“ formos kreivė.
- Šiai funkcijų grupei priklauso **logistinė funkcija**, **hiperbolinis tangentas** ir kt.
- Tačiau **dirbtinių neuroninių tinklų kontekste** labai dažnai **logistinė** ir **sigmoidinė** vartojami sinonimiškai, o **hiperbolinis tangentas** laikomas atskira funkcija.

https://en.wikipedia.org/wiki/Sigmoid_function

Dirbtinio neurono modelis

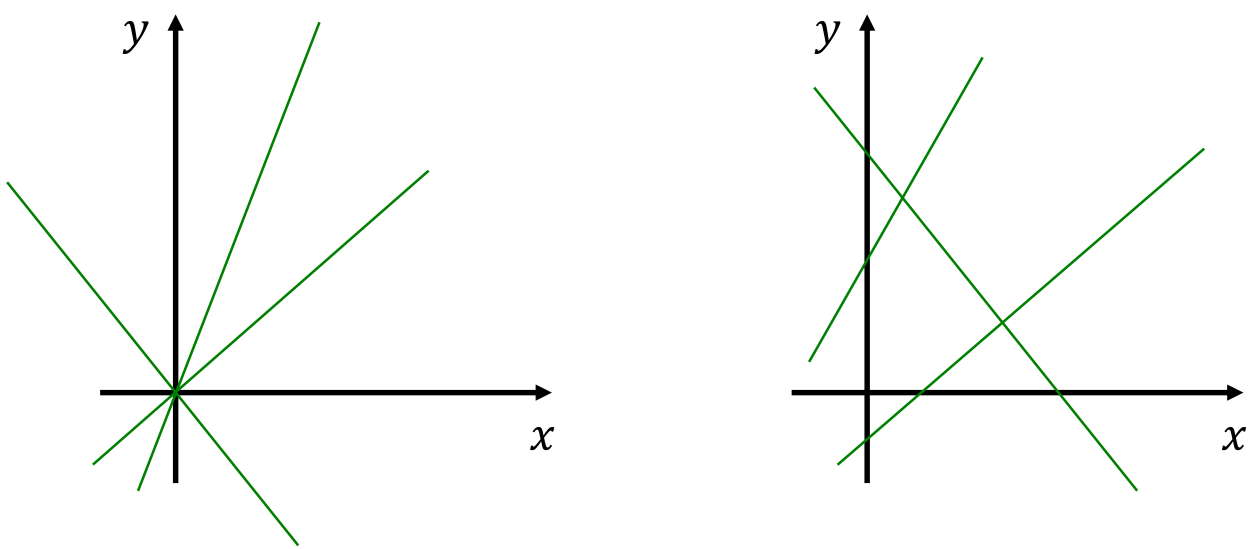
- Pažymėkime **įėjimų vektorių** $X = (x_1, x_2, \ldots, x_n)$, o **svorių vektorių** $W = (w_1, w_2, \ldots, w_n)$.
- Matricine forma tai galima užrašyti:

$$a = W^{\mathrm{T}}X,$$

$$y = f(a) = f\left(W^{\mathrm{T}}X + b\right).$$

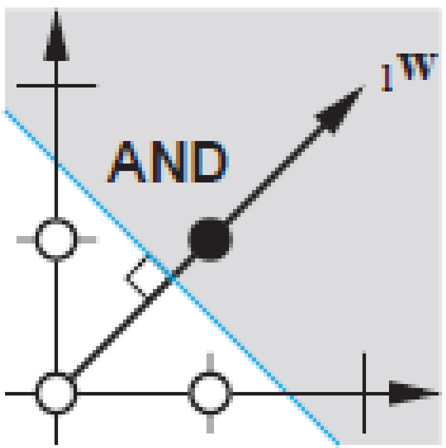
Kas yra poslinkis (bias)?

- Prisiminkime **tiesės lygtį**: $y = ax + b$.
- Jei $b = 0$, tiesė eina per **koordinačių pradžios tašką** (kairėje); jei $b \neq 0$, tiesė neina per koordinačių pradžios tašką (dešinėje).



Kas yra poslinkis (bias)?

- Kaip bus parodyta vėliau, **neurono svoriai** suformuoja klasių skiriamąjį paviršių.
- **Poslinkis** (bias) leidžia lengviau parinkti tinkamas svorių reikšmes.



Kas yra poslinkis (bias)?

- Grįžkime prie dirbtinio neurono **sąsajos su biologiniu neuronu**. Yra nustatyta, kad neuronas sužadinamas tik tuo atveju, jei bendras dendritais gautas signalas viršija tam tikrą lygį, vadinamąjį sužadinimo **slenkstį** (*threshold*).
- **Matematiškai** tai galima užrašyti taip:

$$f(a) = \begin{cases} 1, & \text{jei} \sum_{k=1}^{n} w_k x_k \geq threshold, \\ 0, & \text{jei} \sum_{k=1}^{n} w_k x_k < threshold, \end{cases}$$

- Mokymo metu turėtų būti keičiami ne tik **svorių** w_k, bet ir **slenksčio** (*threshold*) reikšmės.

Kas yra poslinkis (bias)?

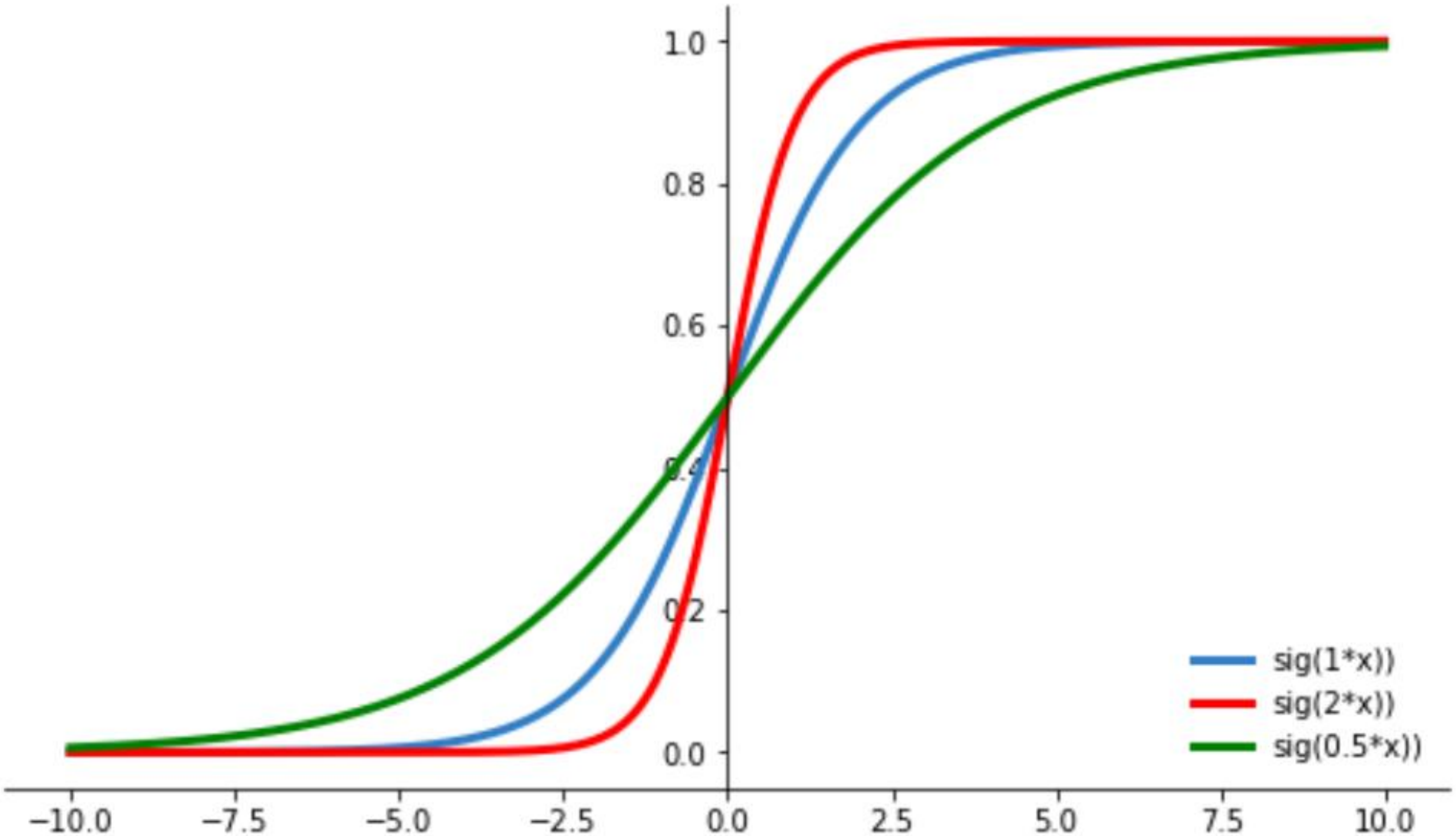
- Perkelkime **slenkstį** (*threshold*) į kitą nelygybės pusę ir pakeiskime jį **poslinkiu** (*bias*) b:

$$f(a) = \begin{cases} 1, & \text{jei} \sum_{k=1}^{n} w_k x_k + b \geq 0, \\ 0, & \text{jei} \sum_{k=1}^{n} w_k x_k + b < 0, \end{cases}$$

- Čia $b = -threshold$.
- Gali būti įvedamas **nulinis įėjimas** x_0, kuris yra pastovus, $x_0 = 1$, tuomet poslinkis (*bias*) tampa **nuliniu svoriu** $b = w_0$ (šie žymėjimai bus naudojami kitose skaidrėse).

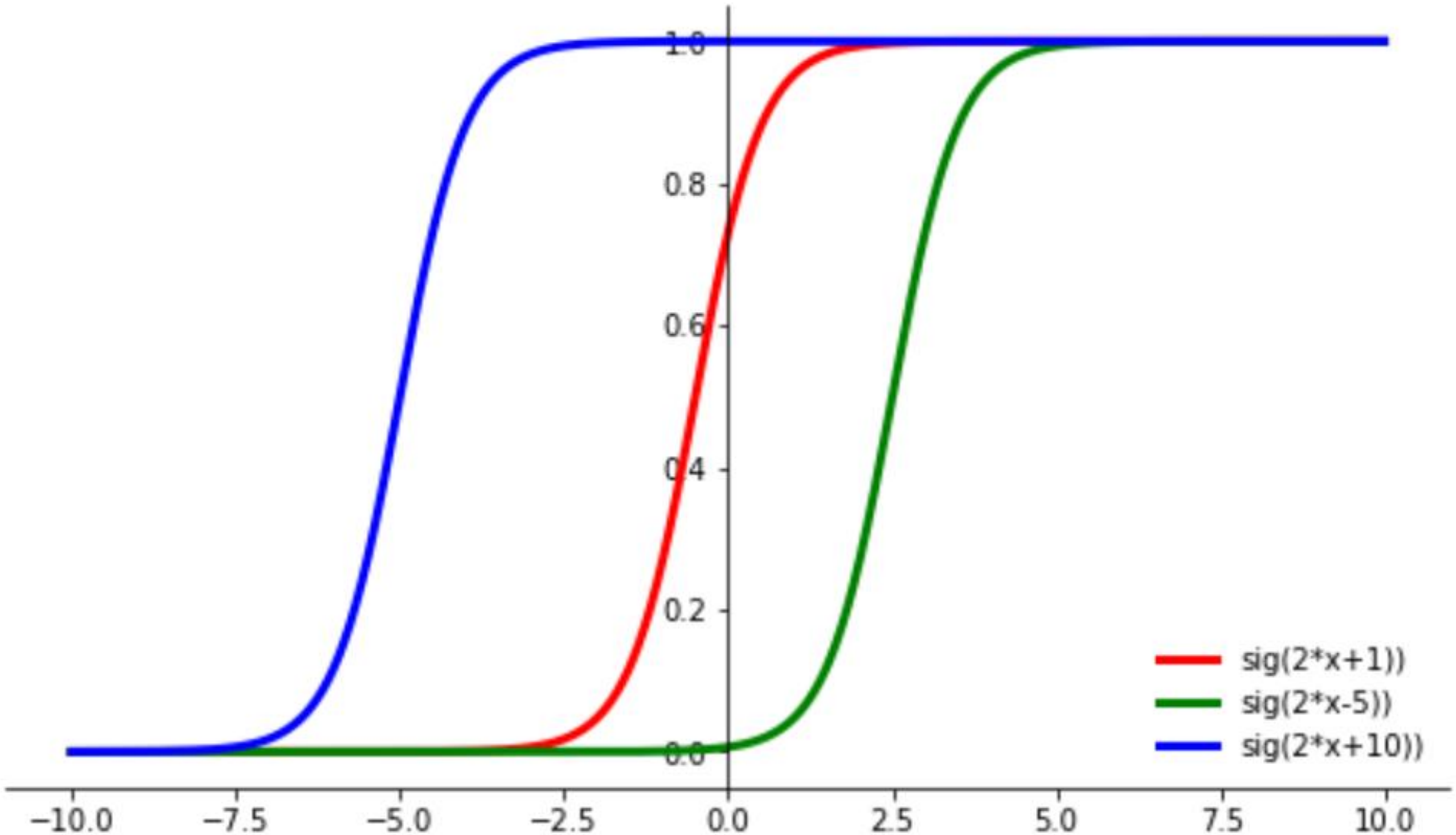
Poslinkio įtaka aktyvacijos funkcijai

Sigmoidinė aktyvacijos funkcija **be poslinkio** (bias)

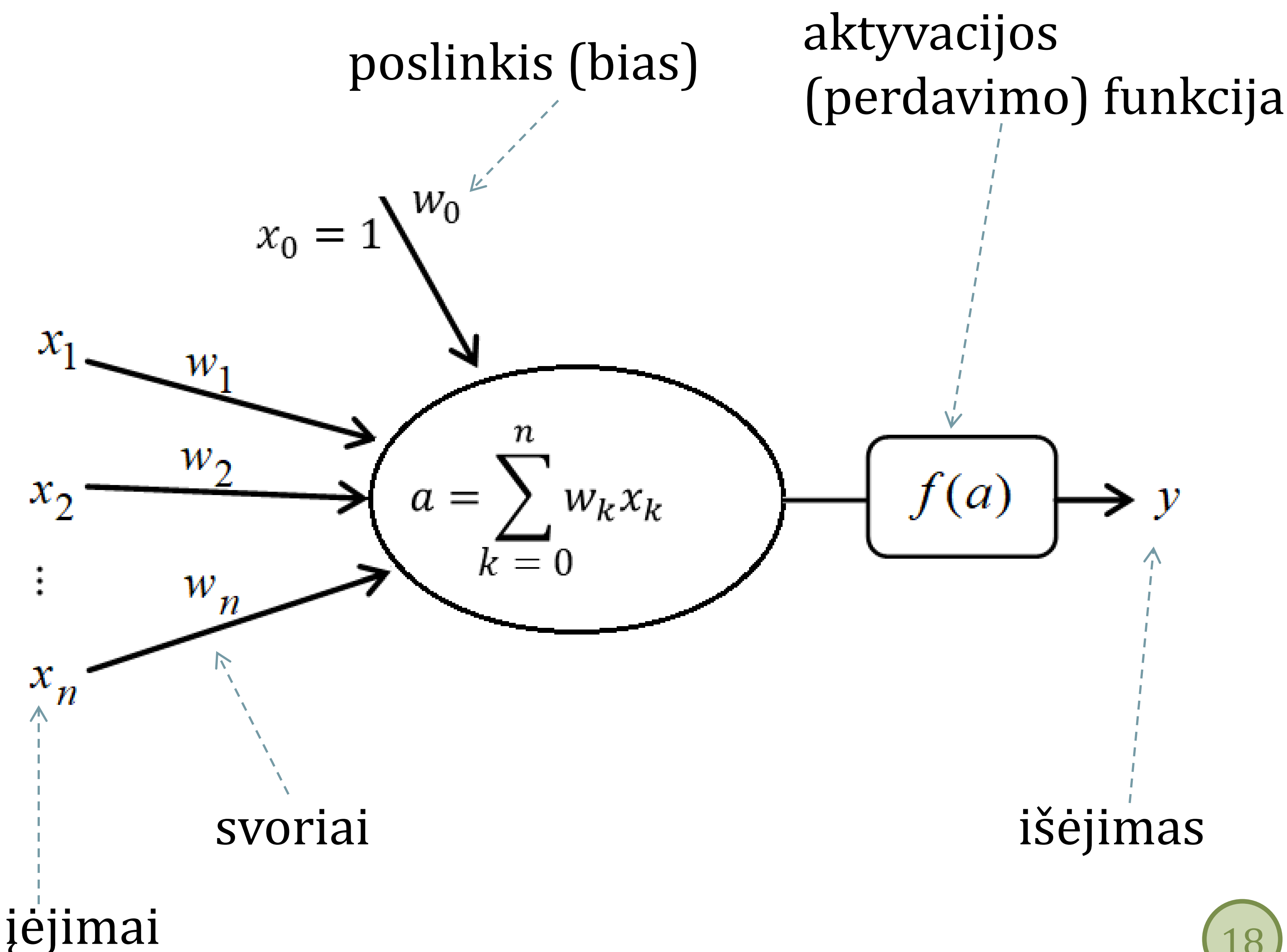


Poslinkio įtaka aktyvacijos funkcijai

Sigmoidinė aktyvacijos funkcija **su poslinkiu** (bias)



Dirbtinio neurono modelis



Neuronas duomenims klasifikuoti

- Dirbtinis **neuronas** dar vadinamas **perceptronu** arba **vienasluoksniu** perceptronu.
- Į **neurono įėjimus** paduodamos požymių x_1, x_2, …, x_n, apibūdinančių duomenis, reikšmės.
- **Neurono išėjime** – duomenų klasių reikšmės.
- Tikslas – **rasti tokias svorių reikšmes** w_0, w_1, w_2, …, w_n, kad apskaičiavus $a = w_0x_0 + w_1x_1 + \ldots + w_nx_n$ ir $f(a)$, **išėjime** y gautos reikšmės **sutaptų su duomenų klasių reikšmėmis**.
- Svorių reikšmės turi būti tokios, kad jos būtų **tinkamos visiems duomenims**.

Neurono 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 $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$, $i = 1, \ldots, m$, 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**.

Neurono mokymas

- Pradžioje svoriams (ir poslinkiui) $W = (w_0, w_1, w_2, \dots, w_n)$ nustatomos **atsitiktinės reikšmės**, čia $w_0 = b$.
- **Mokymo procese** svoriai $W = (w_0, w_1, w_2, \dots, w_n)$ keičiami taip, kad tinklo išėjimo reikšmė y_i, gauta į įėjimą pateikus vektorių X_i, būtų kiek galima **artimesnė norimai reikšmei** t_i, t. y., yra neurono veikimo **paklaida** būtų kiek galima **mažesnė**.
- Ši **paklaida** $E(W)$ gali būti apibrėžiama, kaip skirtumų tarp neurono išėjime gautų reikšmių ir norimų reikšmių kvadratų sumos funkcija (čia m mokymo duomenų skaičius):

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

- Reikalinga **neurono mokymosi taisyklė** (*learning rule*), t. y. pagal kokią formulę bus keičiami (atnaujinami svoriai ir poslinkis).

Neurono mokymas

- Taigi, neurono mokymo eigoje reikia **minimizuoti paklaidos funkciją** $E(W)$, kuri dar vadinama **nuostolių** funkcija (*loss function, cost function*).
- Jeigu ši funkcija yra diferencijuojama pagal svorius, jos minimumą galima rasti **gradientiniais optimizavimo metodais**.
- Patogumo dėlei, dažnai minimizuojama **tokios išraiškos funkcija**:

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

Neurono mokymas gradientinio nusileidimo būdu

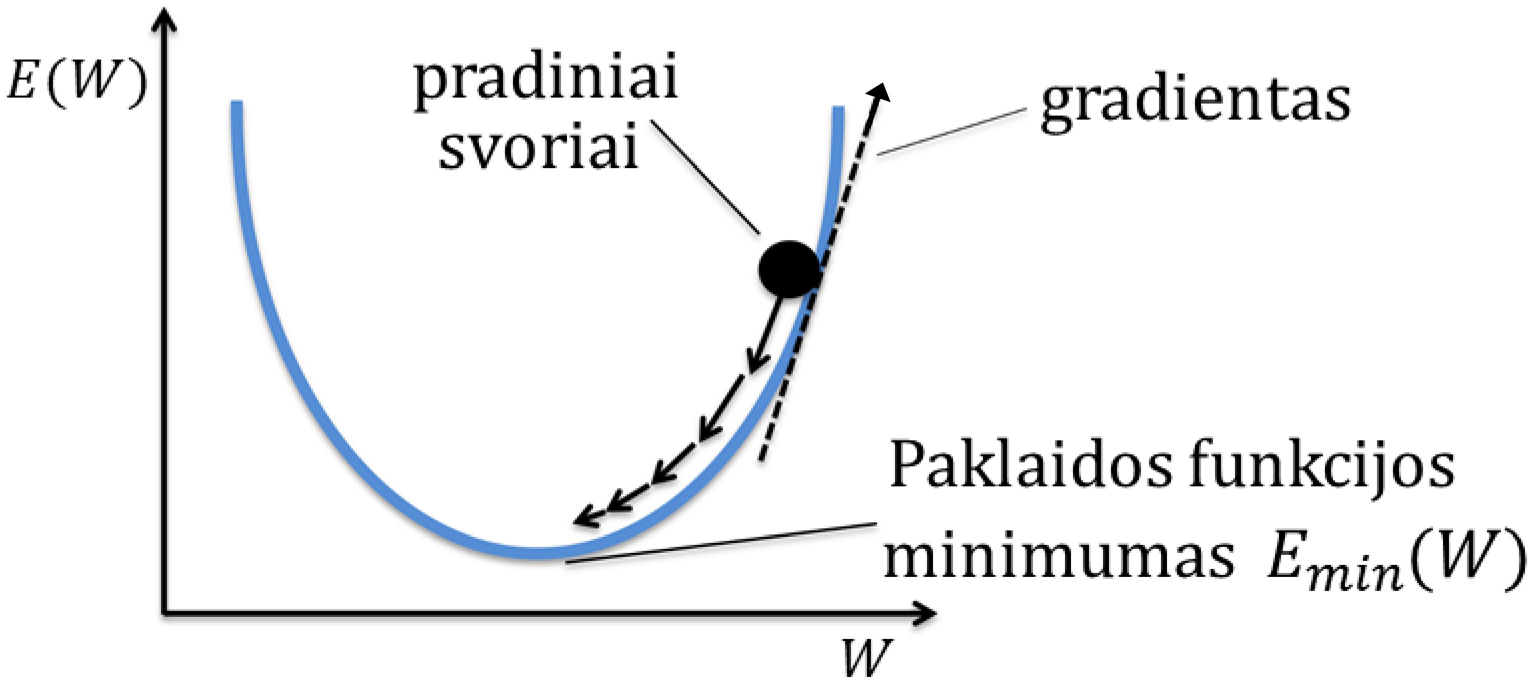
- Iš pradžių **generuojamos atsitiktinės svorių** w_k reikšmės, pvz., intervale (0, 1) arba (–1, 1). Tačiau gali būti taikomos ir kitos pradinių svorių parinkimo strategijos.

- Tada gradientinio nusileidimo algoritmu judama **antigradiento** kryptimi, **svorių reikšmes keičiant** pagal iteracinę formulę

$$w_k \coloneqq w_k - \eta \frac{\partial E(W)}{\partial w_k}, k = 0, \ldots, n$$

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

Gradientinis nusileidimas



Judama antigradiento kryptimi.

Gradientinio nusileidimo tipai

- **Paketinis gradientinis nusileidimas** (*batch gradient descent*),
- **Stochastinis gradientinis nusileidimas** (*stochastic gradient descent*),
- **Mažų paketų gradientinis nusileidimas** (*mini-batch gradient descent*).

Gradientinio nusileidimo **tipas priklauso nuo to**, į **kokį kiekį mokymo duomenų** atsižvelgiama vienai mokymo algoritmo **iteracijai** atlikti.

Paketinis gradientinis nusileidimas

- Tarkime norime minimizuoti paklaidą

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

- Taikant paketinį **gradientinio nusileidimo algoritmą** (*batch gradient descent*) svoriai keičiami pagal formulę:

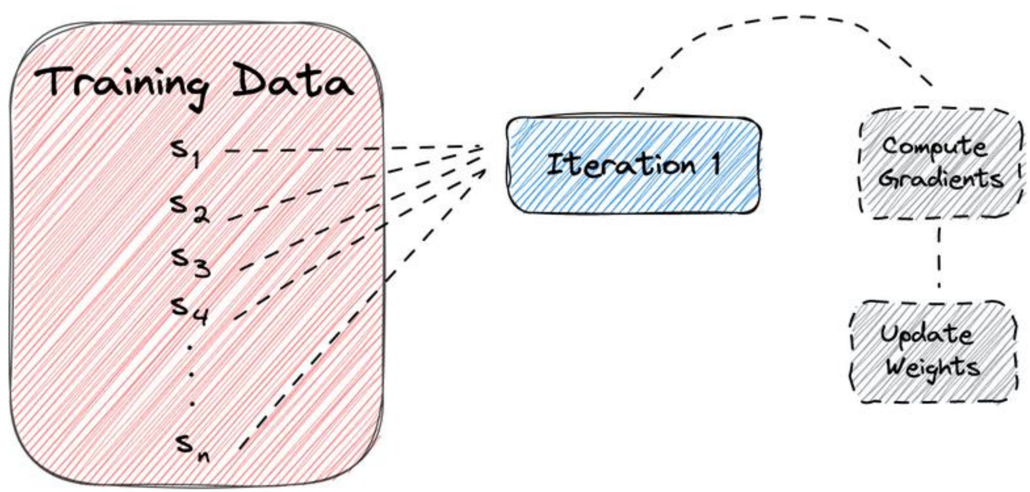
$$w_k := w_k - \eta \nabla E(W) :=$$

$$:= w_k - \eta \frac{1}{m}\sum_{i=1}^{m} \nabla E_i(W).$$

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

Paketinis gradientinis nusileidimas

- Taikant **paketinį gradientinį nusileidimą**, vienos iteracijos metu panaudojami **visi mokymo duomenys** (schemoje $s_i = X_i, n = m$).
- Pirmiausia į neuroną perduodame **visus mokymo duomenis** ir apskaičiuojame **kiekvieno** duomenų įrašo (pavyzdžio) **paklaidos funkcijos gradientą**.
- Tada imame **gradientų vidurkį** ir **atnaujiname svorius** (ir poslinkį) naudodami apskaičiuotą vidurkį.



https://www.baeldung.com/cs/epoch-vs-batch-vs-mini-batch

Stochastinis gradientinis nusileidimas

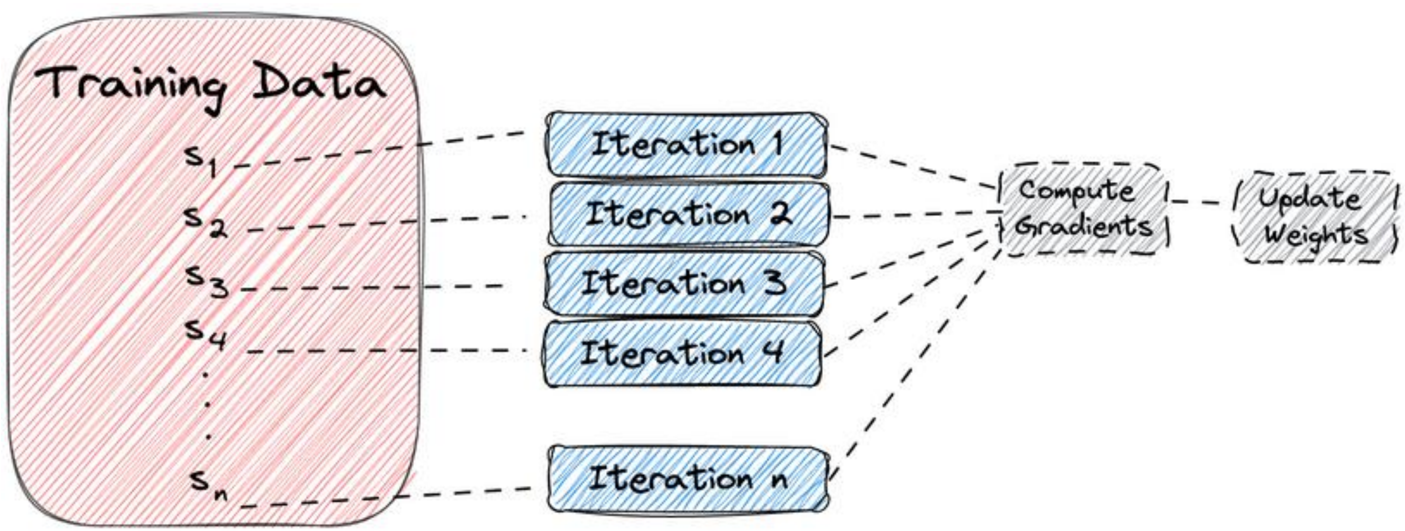
- **Stochastinio gradientinio nusileidimo** (*stochastic gradient descent*, SGD) algoritme tikras funkcijos $E(W)$ gradientas aproksimuojamas gradientu, gautu pagal vieną mokymo duomenų įrašą:

$$w_k := w_k - \eta \nabla E_i(W)$$

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

Stochastinis gradientinis nusileidimas

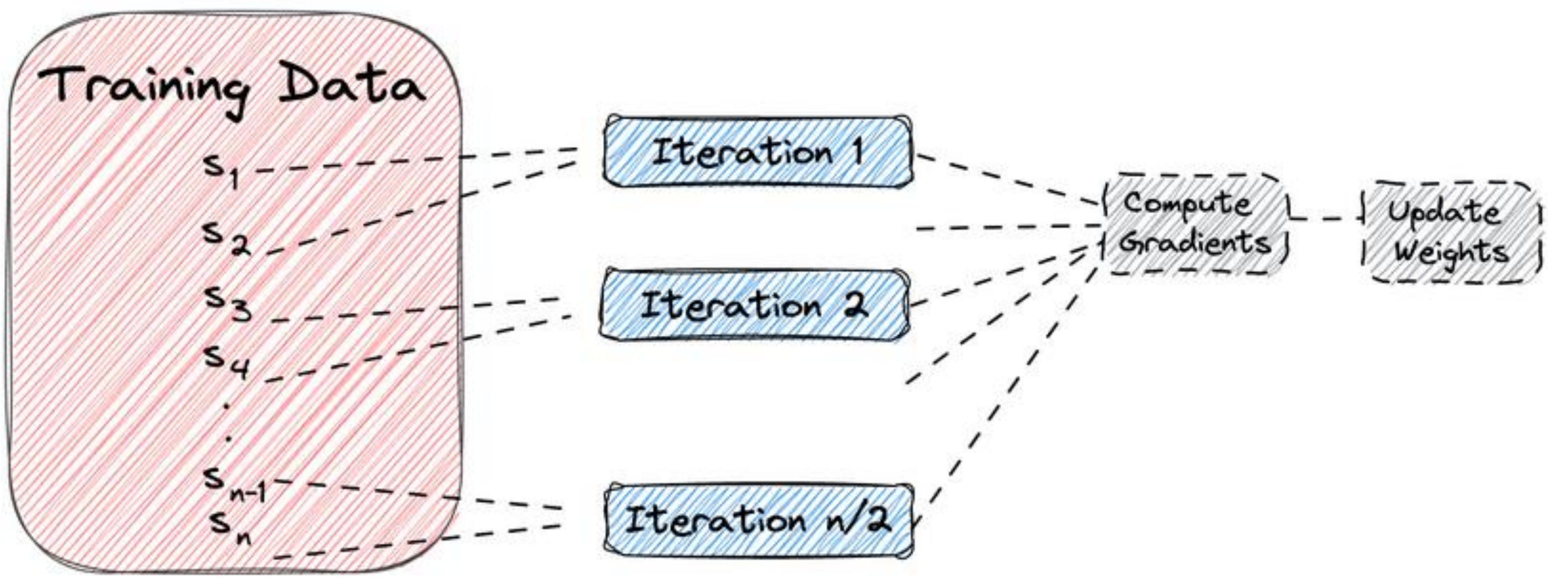
- Taikant **stochastinį gradientinį nusileidimą**, vienos iteracijos metu panaudojamas **tik vienas mokymo duomenų įrašas** (schemoje $s_i = X_i, n = m$).
- T. y. **kiekvienam įrašui** skaičiuojamas gradientas ir atnaujinami svoriai (ir poslinkis).



https://www.baeldung.com/cs/epoch-vs-batch-vs-mini-batch

Mažų paketų gradientinis nusileidimas

- Yra **alternatyva** tarp paketinio gradientinio ir stochastinio gradientinio nusileidimų.
- Čia vienos mokymo iteracijos metu naudojamas duomenų įrašų **paketas** (*batch*), t. y. keli duomenų įrašai.

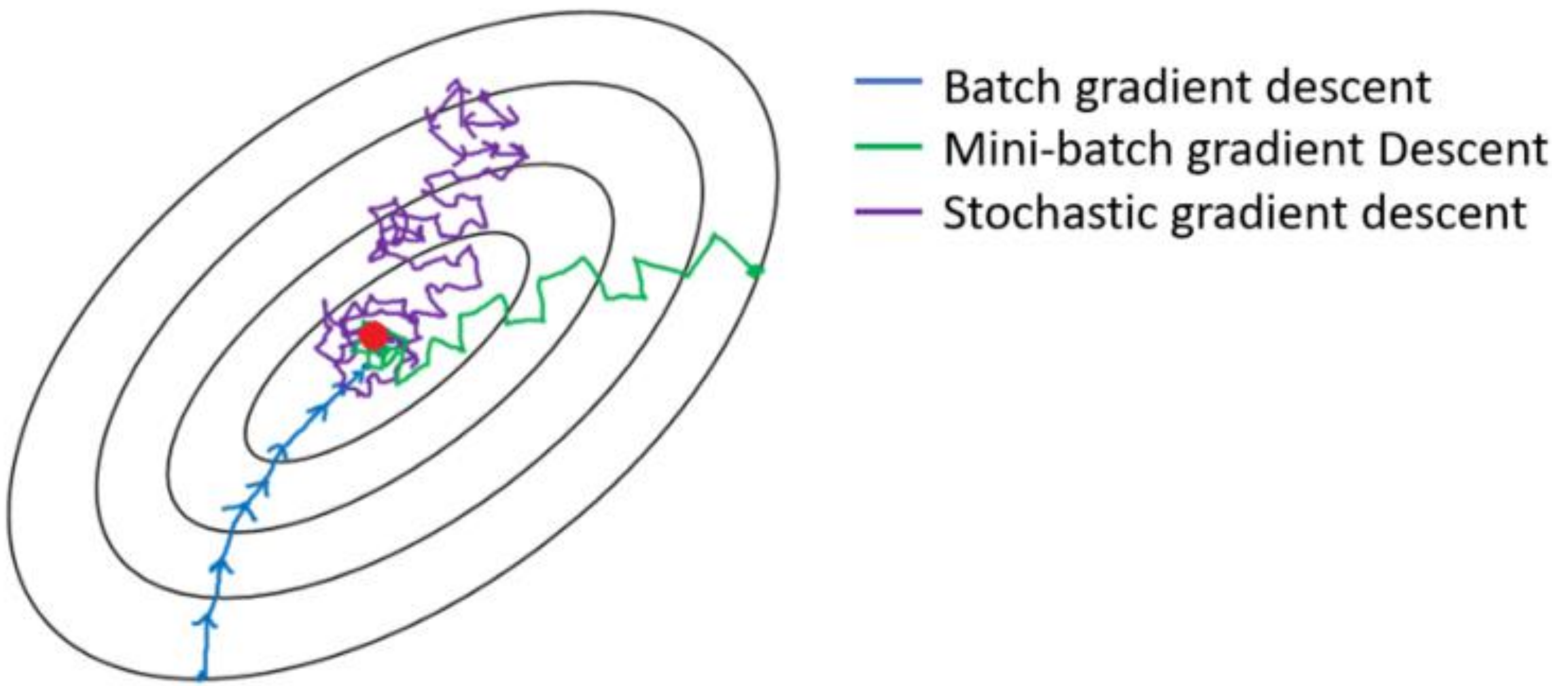


https://www.baeldung.com/cs/epoch-vs-batch-vs-mini-batch

Iteracija ir epocha

- Mokymo **epocha** – tai neuronų mokymo proceso dalis, kurios metu apdorojamas visas mokymo duomenų rinkinys vieną kartą.
- **Paketinio gradientinio nusileidimo** atveju viena epocha atitinka vieną iteraciją.
- **Stochastinio gradientinio nusileidimo** atveju viena epocha atitinka m iteracijų, čia m yra mokymo duomenų kiekis.
- **Mažų paketų gradientinio nusileidimo** atveju viena epocha atitinka $\frac{m}{b}$, čia b – paketo dydis (*batch size*).

Gradientinio nusileidimo tipai



Palyginimas

Paketinis gradientinis nusileidimas	Stochastinis gradientinis nusileidimas
Gradientas apskaičiuojamas naudojant visus mokymo duomenų įrašus.	Apskaičiuojama gradiento aproksimacija naudojant vieną mokymo duomenų įrašą.
Svoriai (ir) poslinkis atnaujinami pateikus visus mokymo duomenų įrašus.	Svoriai (ir) poslinkis atnaujinami pateikus vieną mokymo duomenų įrašą.
Lėtas ir skaičiavimo požiūriu brangus algoritmas.	Greitesnis ir skaičiavimo požiūriu mažiau brangus.
Nerekomenduojamas turint didelę mokymo duomenų aibę.	Gali būti naudojamas turint didelę mokymo duomenų aibę.
Gaunamas optimalus sprendinys, jei pakanka laiko konverguoti.	Gaunamas geras sprendinys, tačiau nebūtinai jis yra optimalus.

P. S. **kompromisas** yra mažų paketų gradientinis nusileidimas.

Neurono mokymo vystymas

- **McCulloch-Pitts** neuronas (1943),
- Klasikinio **Rosenblatt perceptrono** mokymo taisyklė (1958),
- **ADALINE** mokymo taisyklė (1959),
- **Sigmoidinis** neuronas.

Rosenblatt perceptrono mokymo taisyklė

- Tarkime turime **duomenis** $X_i = (x_{i1}, x_{i2}, \dots, x_{in}), i = 1, \dots, m$, kuriuos pateiksime į **neurono įėjimus**. Taip pat turime kiekvienam duomenų įrašui **žymes** (*label*) t_i, kurios sprendžiant klasifikavimo uždavinį atitinka klases.

- Iš pradžių **generuojamos atsitiktinės svorių** w_k, $k = 0, \dots, n$, reikšmės (čia $w_0 = b$).

- Tuomet vykdomas iteracinis procesas, vienos iteracijos metu į neurono įvestį pateikus i-tąjį duomenų įrašą X_i:
  - Suskaičiuojama suma $a_i = \sum_{k=0}^{n} w_k x_{ik}$.
  - Apskaičiuojamos y_i reikšmė pagal **slenkstinę funkciją**. (Rozenblatt naudojo $y_i = f(a_i) = \begin{cases} 1, & \text{jei } a \geq 0, \\ -1, & \text{jei } a < 0, \end{cases}$ tačiau galima naudoti ir standartinę, t. y., vietoj -1 yra 0).
  - Svoriai keičiami (atnaujinami) pagal šią **mokymo taisyklę** (formulę):

$$\text{Jei } t_i \neq y_i \text{, tai } w_k := w_k + \eta(t_i - y_i)(x_{ik})$$

η – yra teigiamas daugiklis, vadinamas **mokymo greičiu** (*learning rate*).

Neurono mokymas

- Svoriai **pakeičiami pateikus vieną** įėjimo vektorių.
- Mokymo procesas kartojamas **daug kartų pateikiant** visus įėjimo vektorius.
- Mokymas **stabdomas** arba atlikus iš anksto nustatytą **iteracijų (ar epochų) skaičių**, arba pasiekus norimą **mažą paklaidos reikšmę**.

ADALINE mokymo taisyklė

- Reikia **minimizuoti** funkciją $E(W) = \frac{1}{2}\sum_{i=1}^{m}(t_i - y_i)^2$.
- Tai suma paklaidų **kiekvienam** įėjimų vektoriui: $E(W) = \sum_{i=1}^{m} E_i$.
- Prisiminkime, kad $y_i = f(a_i) = f(\sum_{k=0}^{n} w_k x_{ik})$.
- **ADALINE** (*Adaptive Linear Neuron*) esminis dalykas – naudojama **tiesinė aktyvacijos funkcija** $f(\sum_{k=0}^{n} w_k x_{ik}) = \sum_{k=0}^{n} w_k x_{ik}$.
- Raskime funkcijos $E(W)$ **išvestinę** pagal w_k:

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

- Svoriai keičiami (atnaujinami) pagal šią **mokymo taisyklę** (formulę):

$$w_k := w_k - \eta(t_i - y_i)(-x_{ik})$$

arba $w_k := w_k + \eta(t_i - y_i)(x_{ik})$.

Rosenblatt perceptronas ir ADALINE

- Tiek mokant **perceptroną**, tiek taikant **ADALINE** mokymosi taisyklę duomenų klasifikavimui atlikti naudojama **slenkstinė funkcija**.
- Abiejų **mokymosi taisyklių formulė yra ta pati**.
- **Perceptronas** atnaujina svorius apskaičiuodamas trokštamos ir prognozuojamos klasės reikšmių skirtumą. Perceptronas visada lygina prognozuojamas vertes +1 arba -1 su trokštamas vertėmis +1 arba -1. Taigi, **perceptronas** mokosi tik tada, kai padaromos klaidos.
- **ADALINE** apskaičiuoja skirtumą tarp tikėtinos klasės vertės t_i (+1 arba -1) ir iš tiesinės funkcijos gautos išėjimo vertės y_i, kuri gali būti bet koks realusis skaičius.
- **ADALINE** gali mokytis net tada, kai nebuvo padarytas klasifikavimo klaidų. Taip yra dėl to, kad prognozuojamos klasės reikšmės y_i neturi įtakos paklaidos skaičiavimui.

Rosenblatt perceptronas ir ADALINE

- Kadangi **ADALINE** mokosi visą laiką, o **perceptronas** – tik po klaidų, **ADALINE** ras sprendimą greičiau nei perceptronas, sprendžiant tą pačią problemą.
- Abiem atvejais taikoma **stochastinio gradientinio nusileidimo** strategija, tik **ADALINE** atveju skaičiuojamas gradientas, o **perceptrono** atveju gradientas nėra skaičiuojamas.

Sigmoidinis neuronas

- **ADALINE** naudojama **tiesinė aktyvacijos funkcija**.
- Tačiau dažniau yra naudojama **sigmoidinė** (logistinė) aktyvacijos funkcija.
- Tuomet neuronas vadinamas **sigmoidiniu**.

Sigmoidinio neurono mokymo taisyklė

- Tarkime turime **duomenis** $X_i = (x_{i1}, x_{i2}, \dots, x_{in})$, $i = 1, \dots, m$, kuriuos pateiksime į **neurono įėjimus**. Taip pat turime kiekvienam duomenų įrašui **žymes** (*label*) t_i, kurios sprendžiant klasifikavimo uždavinį atitinka klases.

- Iš pradžių **generuojamos atsitiktinės svorių** w_k, $k = 0, \dots, n$, reikšmės (čia $w_0 = b$).

- Tuomet vykdomas iteracinis procesas, vienos iteracijos metu į neurono įvestį pateikus i-tąjį duomenų įrašą X_i:
  - Suskaičiuojama suma $a_i = \sum_{k=0}^{n} w_k x_{ik}$.
  - Apskaičiuojamos y_i reikšmė pagal **sigmoidinę funkciją** $y_i = f(a_i) = \frac{1}{1+\mathrm{e}^{-a_i}}$.
  - Svoriai keičiami (atnaujinami) pagal šią **mokymo taisyklę** (formulę):

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

η – yra teigiamas daugiklis, vadinamas **mokymo greičiu** (*learning rate*).

Čia taikomas **stochastinis gradientinis nusileidimas**.

Sigmoidinio neurono mokymo taisyklė

- Reikia **minimizuoti** funkciją $E(W) = \frac{1}{2}\sum_{i=1}^{m}(t_i - y_i)^2$.
- Tai suma paklaidų **kiekvienam** įėjimų vektoriui: $E(W) = \sum_{i=1}^{m} E_i$.
- Prisiminkime, kad $y_i = f(a_i) = f(\sum_{k=0}^{n} w_k x_{ik})$.
- Naudojama **sigmoidinė aktyvacijos funkcija** $f(a_i) = \frac{1}{1+\mathrm{e}^{-a_i}}$. Jos išvestinė $f'(a_i) = f(a_i)\big(1 - f(a_i)\big)$.
- Tuomet funkcijos $E(W)$ **išvestinė** pagal w_k:

$$\frac{\partial E_i(W)}{\partial w_k} = (t_i - y_i) \times \left(-\frac{\partial f(a_i)}{\partial w_k}\right) = (t_i - y_i)(-f(a_i)\big(1 - f(a_i)\big)$$
$$= (y_i - t_i)y_i(1 - y_i).$$

- Svoriai keičiami (atnaujinami) pagal šią **mokymo taisyklę** (formulę):

$$w_k \coloneqq w_k - \eta(y_i - t_i)y_i(1 - y_i)(x_{ik})$$

Neurono mokymas taikant stochastinį gradientinį nusileidimą

- Nustatome **pradinių svorių** (ir poslinkio) $W = (w_0, w_1, w_2, \dots, w_n)$ reikšmes, **mokymo greitį** η, **epochų skaičių** $epochs$, **paklaidos** siekiamo tikslumo reikšmę E_{min}.
- $totalError = \infty, epoch = 0$
- **WHILE** $(totalError > E_{min}$ AND $epoch < epochs)$:
  $//kartojame\ kol\ nėra\ pasiektas\ norimas\ tikslumas\ ir\ nustatytas\ epochų\ skaičius$
  - Sumaišome mokymo duomenų įrašus.
  - $totalError = 0$
  - **FOR** $i = 1,2, \dots, m$:
    - **FOR** $k = 0,1, \dots, n$:
      - $w_k \coloneqq w_k + \eta(t_i - y_i)(x_{ik})$ ($//perceptrono\ ir\ ADALINE\ atveju$)
      - arba $w_k \coloneqq w_k - \eta(y_i - t_i)y_i(1 - y_i)(x_{ik})$ ($//sigmoidinio\ neurono\ atveju$)
    - $error = (t_i - y_i)^2$
    - $totalError = totalError + error$
  - $epoch = epoch + 1$

Neurono mokymas taikant paketinį gradientinį nusileidimą (perceptrono ir ADALINE atvejais)

- Nustatome **pradinių svorių** (ir poslinkio) $W = (w_0, w_1, w_2, \dots, w_n)$ reikšmes, **mokymo greitį** η, **epochų skaičių** *epochs*, **paklaidos** siekiamo tikslumo reikšmę E_{min}.

- $totalError = \infty$, $epoch = 0$

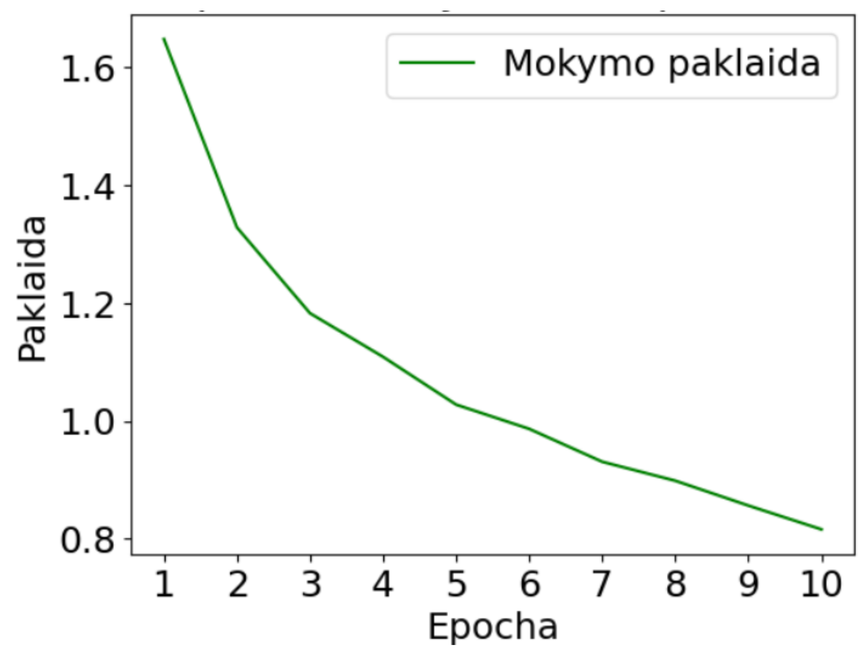
- **WHILE** ($totalError > E_{min}$ AND $epoch < epochs$):
  //kartojame kol nėra pasiektas norimas tikslumas ir nustatytas epochų skaičius
  - Sumaišome mokymo duomenų įrašus.
  - $totalError = 0$
  - $gradientSum = (0, \dots, 0)$
  - **FOR** $i = 1,2, \dots, m$:
    - **FOR** $k = 0,1, \dots, n$:
      - $gradientSum_k = gradientSum_k + (t_i - y_i)x_{ik}$
    - $error = (t_i - y_i)^2$
    - $totalError = totalError + error$
  - **FOR** $k = 0,1, \dots, n$:
    - $w_k := w_k + \eta(gradientSum_k / m)$
  - $epoch = epoch + 1$

Sigmoidinio neurono mokymas taikant paketinį gradientinį nusileidimą

- Nustatome **pradinių svorių** (ir poslinkio) $W = (w_0, w_1, w_2, \dots, w_n)$ reikšmes, **mokymo greitį** η, **epochų skaičių** $epochs$, **paklaidos** siekiamo tikslumo reikšmę E_{min}.
- $totalError = \infty, epoch = 0$
- **WHILE** $(totalError > E_{min}$ AND $epoch < epochs)$:
  //kartojame kol nėra pasiektas norimas tikslumas ir nustatytas epochų skaičius
  - Sumaišome mokymo duomenų įrašus.
  - $totalError = 0$
  - $gradientSum = (0, \dots, 0)$
  - **FOR** $i = 1,2, \dots, m$:
    - **FOR** $k = 0,1, \dots, n$:
      - $gradientSum_k = gradientSum_k + (y_i - t_i) y_i (1 - y_i) x_{ik}$
    - $error = (t_i - y_i)^2$
    - $totalError = totalError + error$
  - **FOR** $k = 0,1, \dots, n$:
    - $w_k := w_k - \eta(gradientSum_k / m)$
  - $epoch = epoch + 1$

Paklaidos priklausomybė nuo epochos numerio

- Mokymo metu verta stebėti **paklaidos priklausomybę nuo epochos** numerio.
- Reikia **kaupti paklaidos reikšmes**, gautas po kiekvienos epochos.
- Ankstesnėse skaidrėse pateiktuose pseudo koduose reiktų $totalError$, gautos **po kiekvienos epochos, kaupti į masyvą**.



Galutinis paklaidos įvertinimas mokymo duomenims

- Kai **neuronas** yra **išmokytas**, t. y. rasti tinkami svoriai (ir poslinkis), teigiama, kad turime **išmokytą modelį**.
- **Modelio kokybei įvertinti** galime apskaičiuoti **bendrą paklaidą mokymo duomenims**.
- Įvedus **validavimo** ir **testavimo** duomenų sąvokas, paklaida bus vertinama ir šiems duomenims.

Galutinis paklaidos įvertinimas mokymo duomenims

- Išmokius neuroną, turime svorius $w_0, w_1, w_2, \dots, w_n$ (čia $w_0 = b$).

- **Paklaida** $E(W) = \frac{1}{m}\sum_{i=1}^{m}(t_i - y_i)^2$, dar vadinama vidutinė kvadratinė paklaida (*mean squared error*, MSE), gali būti skaičiuojama taip:
  - $totalError = 0$
  - **FOR** $i = 1,2, \dots, m$:
    - $a_i = 0$
    - **FOR** $k = 0,1, \dots, n$:
      - $a_i = a_i + w_k x_{ik}$
    - $y_i = f(a_i)$
    - $error_i = (t_i - y_i)^2$
    - $totalError = totalError + error_i$
  - $totalError = totalError/m$

Skiriamasis paviršius

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

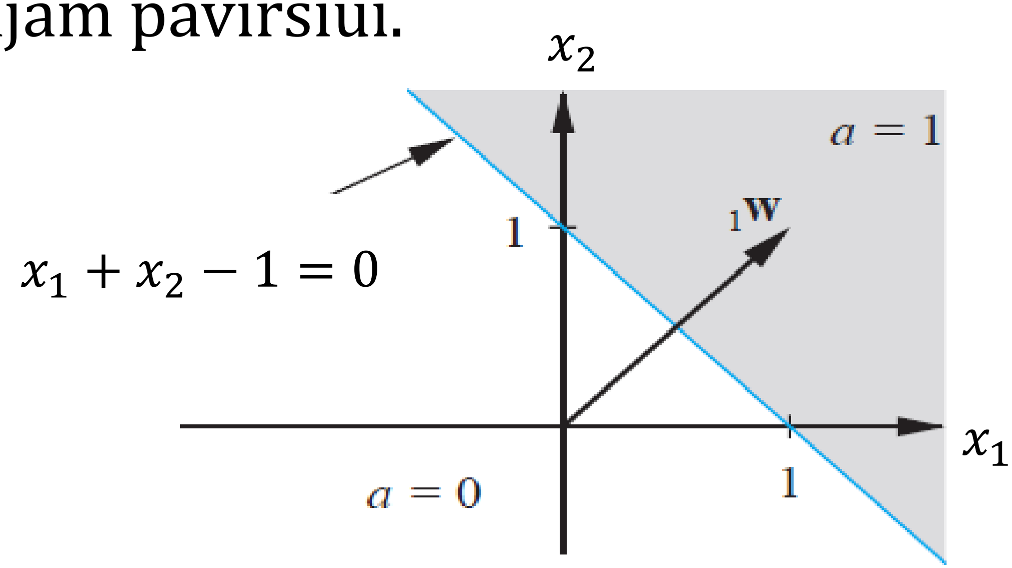
$$y = f(a) = \begin{cases} 1, \text{jei } a \geq 0 \\ 0, \text{jei } a < 0 \end{cases}$$

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

$$w_1 x_1 + w_2 x_2 + w_0 = 0$$

Pavyzdys

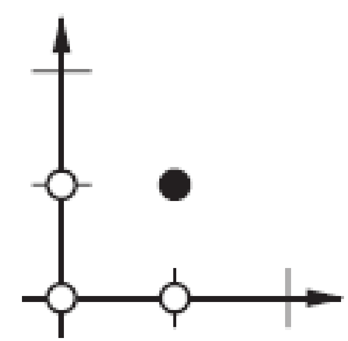
- Tarkime $w_1 = 1, w_2 = 1, w_0 = -1$. Tuomet **skiriamasis paviršius** $x_1 + x_2 - 1 = 0$.
- Šios tiesės vienoje pusėje, **neurono išėjimas** bus lygus 0, kitoje 1.
- Svorių vektorius turi būti **statmenas** (**ortogonalus**) skiriamajam paviršiui.



Skiriamasis paviršius loginei funkcijai AND (1)

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

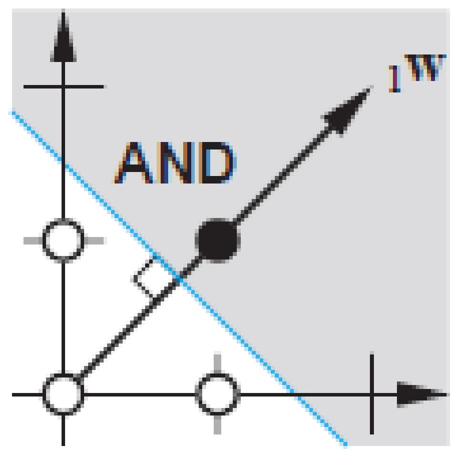
	x_1	x_2	t
X_1	0	0	0
X_2	0	1	0
X_3	1	0	0
X_4	1	1	1



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

Skiriamasis paviršius loginei funkcijai AND (2)

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



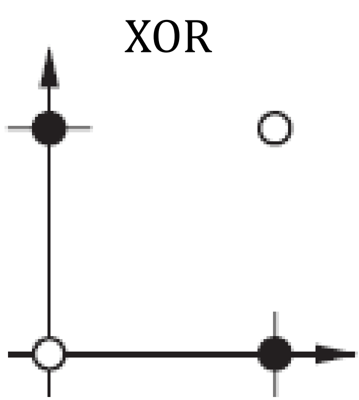
- Dabar reikia pasirinkti svorių vektorių, kuris būtų statmenas skiriamajai tiesei. Vienas variantų $W(w_1, w_2) = (2, 2)$.

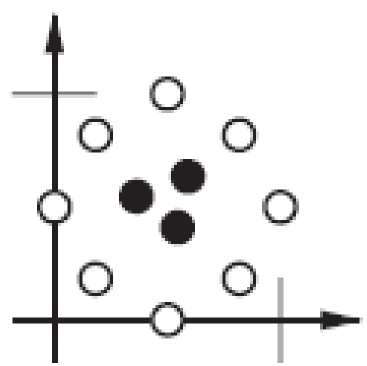
Skiriamasis paviršius loginei funkcijai AND (3)

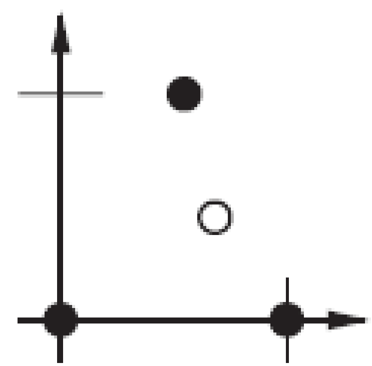
- Dabar belieka **rasti** w_0.
- Reikia **parinkti tašką** (x_1, x_2), esantį ant skiriamosios tiesės ir tenkinantį lygybę $w_1 x_1 + w_2 x_2 + w_0 = 0$.
- Galimas **taškas** $(x_1, x_2) = (1{,}5, 0)$.
- Tuomet $2 \times 1{,}5 + 2 \times 0 + w_0 = 0$. Iš čia $w_0 = -3$.

Tiesiškai neatskiriami 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**. Svoriai keičiami, jei tik randami geresni.

„Kišeninis“ algoritmas (2)

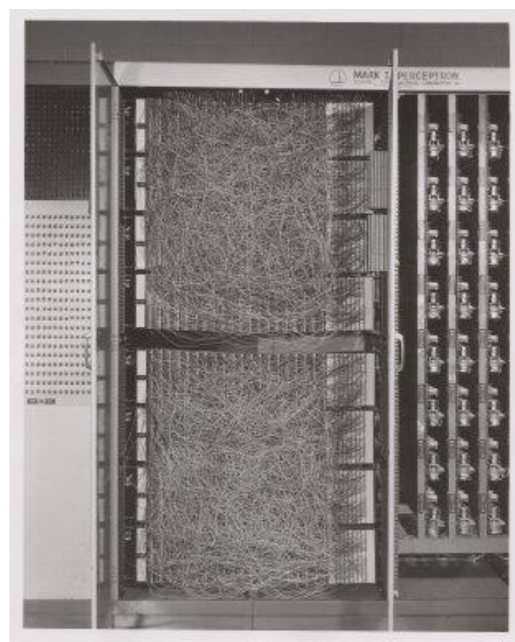
```
pocket (training_list, max_iteration)
        w = randomVector()
        best_error = error(w)
        for i in range(0, max_iteration)
                x=misclassified_sample(w, training_list)
                w=vector_sum(w, x.y(x))
                if error(w) < best_error
                        best_w = w
                        best_error = error(w)
return best_w
```

Dirbtinių neuronų ištakos

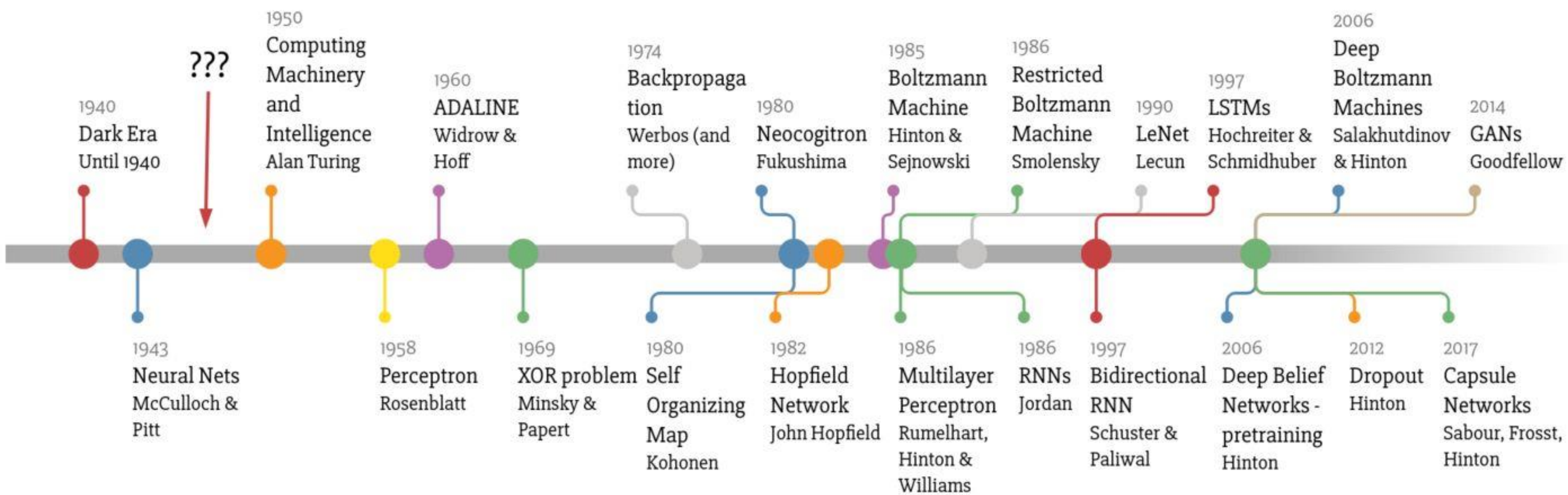
- **McCullogh-Pitts** (MCP, M-P) neurono modelis (1943)
  - Įėjimai tik (0, 1).
  - Tik slenkstinė aktyvacijos funkcija.
  - Vienintelė w_0 reikšmė visiems įėjimams.
  - Visiems įėjimams vienodi svoriai (teigiami skaičiai).
- **Rosenblatt** perceptronas (1958)
  - Visi svoriai nėra identiški (teigiami 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.



Deep Learning Timeline



Made by Favio Vázquez