

RĪGAS TEHNISKĀ UNIVERSITĀTE

Datorzinātnes un informācijas tehnoloģijas fakultāte

Mākslīgā intelekta un sistēmu inženierijas katedra

Rinalds Daniels Pikše

bakalaura akadēmisko studiju programmas “Datorsistēmas”

students, stud. apl. nr. 171RDB359

**Salīdzinošā analīze Beiesa neironu tīklu
stabilitātei**

BAKALAURA DARBS: RUDENS SEMESTRA ATSKAITE

Zinātniskais vadītājs Dr.Sc.Ing. Pētnieks

Ēvalds Urtāns

RĪGA 2022

SATURA RĀDĪTĀJS

1. DARBA BŪTĪBA UN AKTUALITĀTE	3
2. INFORMĀCIJAS AVOTU APSTRĀDES REZULTĀTI.....	5
2.1. Praktiskie uzdevumi.....	5
2.1.1. Gravitātes spēle	5
2.2. Analizētie rakstu avoti	9
2.2.1. Monte Carlo Dropout.....	9
2.2.2. Why you should use Bayesian Neural Network	9
2.2.3. Bayesian Inference: The Best 5 Models and 10 Best Practices for Machine Learning	9
IZMANTOTIE INFORMĀCIJAS AVOTI	

1. DARBA BŪTĪBA UN AKTUALITĀTE

Bakalaura darba mērķis: Salīdzināt Beiesa neironu tīklus ar parastajiem neironu tīkliem, izmantojot datu kopu, kura ir ārpus problēmsfēras

Bakalaura darba uzdevumi:

Izprast kā Beiesa neironu tīkli atšķiras no parastiem neironu tīkliem.

Noteikt kā Beiesa neironu tīkli nosaka modeļa nenoteiktību.

Noteikt kā palielināt Beiesa tīklu rezultātu ticamību.

Problēmas nostādne:

Varianti – īsti nezinu kā uzrakstīt

1. Neironu tīkli nespēj pareizi norādīt rezultātus datu kopām, kas atrodas ārpus problēmsfēras, kurai neironu tīkls tika trenēts.

Tēmas aktualitātes pamatojums:

Mašīnāpmācība ir datorzinību lauks, kas mēģina izstrādāt automatizētus modeļus, lai risinātu plaša spektra problēmas. Mašīnāpmācība pēdējos gados ir strauji augusi popularitātē un pielietojumu daudzveidībā dažādās sfērās – tiek aplēsts, ka servisu kopums kas balstīti uz mašīnāpmācību varētu vairāk kā desmitkārsoties līdz 2030. gadam ¹ Pielietojot neironu tīklus svarīgi ir zināt, cik liels ir pārliecības līmenis prognozēm. Šis darbs pievēršas Beiesa neironu tīkliem, kuri ir balstīti uz Beiesa metodēm, kuras sāka izplatīties 1980. gados un ir kļuvuši par plaši pielietotu neironu tīklu grupu. Beiesa neironu tīkli ļauj izprast, kādā mērā uzticēties modeļa gala rezultātam - tie izmanto varbūtību sadalījumus mainīgo vietā nevis nominālu vērtības, tādā veidā ievietojot varbūtības aprēķinu katrā modeļa solī. Izpētot tuvāk Beiesa neironu tīklu īpašības, mašīnāpmācība var kļūt skaidrāka par veikto prognožu atbilstību problēmu risināšanai. Zināt, kādās robežās uzticēties izveidotajiem modeļiem ir būtiski, jo no tiem tiek veikti lēmumi, kas ietekmē aizvien vairāk cilvēku ikdienā.

Tālāk sniedziet izstrādes plānu, kurā ir atspoguļotas bakalaura darba izstrādē veicamas aktivitātes un to izpildes termiņi.

1.1. tabula

¹ Machine Learning as a Service Market Size, Report 2030

Machine Learning as a Service Market Size, Report 2030 (2022). Available at: <https://www.precedenceresearch.com/machine-learning-as-a-service-market> (Accessed: 15 December 2022).

Bakalaura darba izstrādes plāns

Aktivitāte	Terminš
Apgūt parasto neironu tīklu teoriju	01.01.2023
Apgūt Beiesa neironu tīklu teoriju, izmantojot Monte-Carlo Dropout metodi	01.02.2023
Apmācīt abu veidu modeļus un veikt eksperimentus	01.03.2023
Dokumentēt rezultātus, formulēt hipotēzes tālākiem pētījumiem	01.04.2023
Bakalaura darba nodošana	29.05.2023

2. INFORMĀCIJAS AVOTU APSTRĀDES REZULTĀTI

Šajā nodaļā (ja nepieciešams vairākās) ievieto izlasīto informācijas avotu apstrādes rezultātus, kas ir vismaz 7 lappuses teksta.

2.1. Praktiskie uzdevumi

Pirmā darba aktivitāte bakalaura izstrādei, man ir bijusi “Apgūt parasto neironu tīklu teoriju”. Tās ietvaros darba vadītājs, Dr.Sc.Ing. Ēvalds Urtāns, man deva praktiskus uzdevumus, ko pildīt, lai attīstītu savas zināšanas, kas man palīdzēs bakalaura darba izstrādes procesā. Kā pirmos informācijas avotus aprakstīšu atskaites katram no šiem uzdevumiem.

2.1.1. Gravitātes spēle

Šī uzdevuma mērķis bija atkārtot python valodas sintaksi un matricu operācijas. gāja labi, sākumā bija neskaidri kā strukturēt kodu un uz ko fokusēties, taču sanāca atkārtot matricu transformācijas un python sintaksi. Pēc code review sapratu ka svarīgākais bija kombinētās matricu transformācijas un numpy bibliotēkas specifiskās sintakses lietas. Tās pielabojot, sapratu ko mācīties tālāk.

Izveidoto funkciju saraksts:

1. Dot function

```
def dot(X, Y):
    is_transposed = False

    X = np.atleast_2d(X)
    Y = np.atleast_2d(Y)

    if X.shape[1] != Y.shape[0]:
        is_transposed = True
        Y = np.transpose(Y)

    X_rows = X.shape[0]
    Y_columns = Y.shape[1]

    for X_row in range(X_rows):
        for Y_column in range(Y_columns):
            product[X_row, Y_column] = np.sum(X[X_row,:] * Y[:, Y_column])

    if is_transposed:
        product = np.transpose(product)

    if product.shape[0] == 1:
        product = product.flatten()

    return product
```

2. Vector normalization

```
def l2_normalize_vec2d(vec2d):
    length = math.sqrt(vec2d[0]**2 + vec2d[1]**2)
    normalized_vec2d = np.array([vec2d[0]/length, vec2d[1]/length])
```

```
return normalized_vec2d
```

3. Translaiton matrix

```
def translation_mat(dx, dy):
    T = np.array([
        [1.0, 0.0, dx],
        [0.0, 1.0, dy],
        [0.0, 0.0, 1.0]
    ])
    return T
```

4. Scaling matrix

```
def scale_mat(dx, dy):
    T = np.array([
        [dx, 0.0, 0.0],
        [0.0, dy, 0.0],
        [0.0, 0.0, 1.0]
    ])
    return T
```

5. Circle generation

```
def drawCircle(radius):
    detail = 24
    circle = []
    d = 0
    x = 0
    while d < 375:
        circle.append([radius*np.cos(np.radians(d)), radius*np.sin(np.radians(d))])
        d +=375/detail
        x +=1

    return np.array(circle)
```

5. Additions

Pavadot laiku ar spēli mazliet vairāk, pievienoju ekstra elementus spēlei, lai to padarītu jautrāku un vizuāli pievilcīgāku:

5.0 Izveidoju emission particle objektu

```
class EmissionParticle(MovableObject):
    def __init__(self, directionVector, position):
        super().__init__()
        self.speed = .75
        I = np.array([
            [1, 0],
            [0, 1],
        ])

        self.vec_pos = dot(position, I)

        radius = np.random.uniform(0.15, 0.3)

        s = drawCircle(radius)
        self.geometry = s

        directionChangeMatrix = np.array([
            [np.random.uniform(-1.5, -0.5), 0],
            [0, np.random.uniform(-1.5, -0.5)],
        ])
        self.vec_dir = dot(directionVector, directionChangeMatrix)
        self.lifespan = 1
    def update_movement(self, dt):
        self.lifespan -= dt
        super().update_movement(dt)
        self.geometry = self.geometry * .75
        self.speed -= dt * 0.6
        if self.lifespan < 0:
            self.geometry = clearMatrix(self.geometry)
```

```
def createEmissionParticles(player):
    particles = []
    particles.append(EmissionParticle(player.vec_dir, player.vec_pos))
    particles.append(EmissionParticle(player.vec_dir, player.vec_pos))
    particles.append(EmissionParticle(player.vec_dir, player.vec_pos))
    return np.array(particles)
```

5.1 Pievienoju stratēģiju spēles izbeigšanai - ja planēta pietuvojas pārāk tuvu speletajam, spēle beidzas:

5.1.1 noteikt distanci starp diviem objektiem

```
def distanceBetweenTwoObjects(pos1, pos2):
    return np.sum((pos1 - pos2)**2)/2
```

5.1.2 Kā updatot izraisīto spēku spēlētājam, planētai pietuvojoties tuvāk

```
def updateForceOnPlayer(self:MovableObject):
    F = 9.82 * self.radius / distanceBetweenTwoObjects(self.vec_pos, player.vec_pos)*2
    F_vec = l2_normalize_vec2d(self.vec_pos - player.vec_pos)
    player.external_forces[self.planetNumber] = F * F_vec
```

5.1.3 Parbaude vai objekti ir saskrējušies:

```
def isCollided(firstObject:MovableObject, secondObject:MovableObject):
    d_2 = distanceBetweenTwoObjects(firstObject.vec_pos, secondObject.vec_pos)
    if d_2 < 0.2:
        return True
    return False
```

5.1.4 Spēles izbeigšana:

```
def closeWithGameOver():
    plt.text(x=-SPACE_SIZE+9, y=SPACE_SIZE-9, s=f'GAME OVER')
    plt.pause(5)
    global is_running
    is_running = False
```

5.1.5 Planētas update_movement implementācijai pievienoju izveidotās funkcijas:

```
class Planet(MovableObject):
    def __init__(self, name, index, radius):
        super().__init__()
        self.attribute_name = name
        self.speed = 0
        self.planetNumber = index
        print(radius)
        self.radius = radius

        s = drawCircle(self.radius)
        self.geometry = s

        self.vec_pos = np.array([np.random.uniform(-10.0, 10.0), np.random.uniform(-10.0, 10.0)])
        self.speed = 0

    def update_movement(self, dt):
        if isCollided(self, player):
            closeWithGameOver()

        super().update_movement(dt)

        updateForceOnPlayer(self)
```

Gala rezultātā tika izveidota programma, kas ļauj cilvēkam braukt pa vizuālu plakni ar trīsstūra figūru, šai figūrai ir jāizvairās no automātiski uzģenerētam planētām, kas iesūc figūru iekšā sevī, līdzko ta pietuvojās pārāk ātri. Figūrai ir iespējams paātrināt kustību, mainīt virzienu un lidot caur kartes sienām, lai izvairītos no planētām.



Attēlā redzama spēles gaita.

2.1.2. Inversā kinemātika

2.2. Analizētie rakstu avoti

Raksti, kas tika analizēti kopumā bija ļoti vispārīgi, par Beiesa neirālajiem tīkliem un time netika veltīts tik daudz laiks, cik tika veltīts prieks praktiskajiem uzdevumiem.

2.2.1. Monte Carlo Dropout

Michał Oleszak

Monte Carlo Dropout (2021).

Pieejams: <https://towardsdatascience.com/monte-carlo-dropout-7fd52f8b6571>

(Skatīts: 15 December 2022).

TODO: kkads apraksts

2.2.2. Why you should use Bayesian Neural Network

Yeung Wong

Why you should use Bayesian Neural Network? (2021).

Pieejams: <https://towardsdatascience.com/why-you-should-use-bayesian-neural-network-aaf76732c150>

(Accessed: 15 December 2022).

TODO: kkads apraksts

2.2.3. Bayesian Inference: The Best 5 Models and 10 Best Practices for Machine Learning

Anil Tilbe

Bayesian Inference: The Best 5 Models and 10 Best Practices for Machine Learning (2022).

Pieejams: <https://pub.towardsai.net/bayesian-inference-the-best-5-models-and-10-best-practices-for-machine-learning-11238a43929e>

(Skatīts: 15 December 2022).

TODO: kkads apraksts

//kam nebija laiks:

Zoubin Ghahramani University of Cambridge

(2022) Repository.cam.ac.uk. Available at:

<https://www.repository.cam.ac.uk/bitstream/handle/1810/248538/Ghahramani%202015%20Nature.pdf>

(Accessed: 13 December 2022).

Kirill Bykov, Marina M.-C. Hohne, Adelaida Creosteanui , Klaus-Robert Muller, Frederick Klauscheng,

Shinichi Nakajimaa, Marius Kloft

(2022) Arxiv.org. Available at: <https://arxiv.org/pdf/2108.10346.pdf> (Accessed: 13 December 2022).

Monte Carlo Dropout

Monte Carlo Dropout (2021). Available at: [https://towardsdatascience.com/monte-carlo-dropout-](https://towardsdatascience.com/monte-carlo-dropout-7fd52f8b6571)

[7fd52f8b6571](https://towardsdatascience.com/monte-carlo-dropout-7fd52f8b6571) (Accessed: 15 December 2022).

IZMANTOTIE INFORMĀCIJAS AVOTI

Šeit ir jāapraksta visi informācijas avoti, uz kuriem ir atsauces 1. un 2. nodaļā.

Informācijas avotu un atsauču noformēšanā ir jāievēro DITF norādījumi studiju noslēguma darbu noformēšanai, kas ir atrodami DITF mājaslapā "Studijas/Noderīgi dokumenti un saites"