«Қ.И.СӘТБАЕВ АТЫНДАҒЫ ҚАЗАҚ ҰЛТТЫҚ ТЕХНИКАЛЫҚ ЗЕРТТЕУ УНИВЕРСИТЕТІ»
КОММЕРЦИЯЛЫҚ ЕМЕС АКЦИОНЕРЛІК ҚОҒАМЫ

ҒЫЛЫМИ ЖЕТЕКШІНІҢ

ПІКІРІ

Дипломдық жұмыс
(жұмыс түрінің атауы)
Шералиева Мерей Мұханқызы
(білім алушының Т.А.Ә.)
6В06103 - Математикалық және компьютерлік модельдеу
(мамандық атауы мен шифрі)

Тақырыбы: Кохонен желілерін қолданатын классификациялар. Кохонен қабатының моделін зерттеу

Дипломдық жұмыс авторы өзінің еңбекқорлығымен, тапсырылған жұмысты уақытымен орындауымен ерекшеленеді.
Дипломдық жұмыс 6В06103 «Математикалық және компьютерлік модельдеу» мамандығына қойылатын талапқа сай орындалған, технологиялық аппаратты математикалық моделдеу үшін қолдану теориясы кеңінен қолданылған.

Диплом жұмысы күрделі тақырыптың бір өзекті мәселесін толығымен қамтып, аша алған. Көркемдеу талаптарына сәйкес орындалған.
Жұмыс авторы толығымен 95% жоғары бағалануға лайықты.

Ғылыми жетекші:
«Жоғары математика және моделдеу»
кафедрасының аға оқытушысы
_____ Лукпанова Л.Х.
(қолы)
«17» 05 2024 ж.

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
"Қ. И. СӘТБАЕВ атындағы ҚАЗАҚ ҰЛТТЫҚ ТЕХНИКАЛЫҚ ЗЕРТТЕУ УНИВЕРСИТЕТІ" КЕАҚ

Автоматика және ақпараттық технология институты
Жоғары математика және моделдеу кафедрасы

6B06103 « Математикалық және компьютерлік модельдеу»
мамандығының студенті Шералиева Мерейдін дипломдық жұмысына берілген

СЫН ПІКІР

Мамандығы: 6B06103 « Математикалық және компьютерлік модельдеу»

Дипломдық жұмыс тақырыбы: Кохонен желілерін қолданатын классификациялар. Кохонен қабатының моделін зерттеу.

Орындалды:

а) графикалық бөлімі 9 парақ

ә) түсіндірме жазба - бет

б)математикалық есептерді шешу, модельдерді қорытындылау 9 бет

ЖҰМЫСҚА ҚАТЫСТЫ ЕСКЕРТУЛЕР

Бұл дипломдық жұмыста студент моделдеу әдістерімен және технологиясымен танысып және нақты әдебиеттерді пайдаланған.

Жұмыста әртүрлі оқыту алгоритмдерінің жіктеу есебін шешудегі қателікке әсері зерттелген, сонымен қатар Кохонен нейрондық желісінің топологиясын генерациялау мәселесі қарастырылған.

Әр түрлі математикалық есептерді шешу үшін комьютерде қолданбалы программалар пакетін (Python сияқты т.б.) қолдана білу керек және практикада кеңінен қолданатын есептерді қарастырған.

Дипломдық жұмыс мақсатқа сай және жоғары техникалық дәрежеде орындалған. Жұмысқа қатысты ескертулер ретінде техникалық әдебиеттерді аударуда кеткен кейбір қателіктерді келтіруге болады.

Жұмысқа берілген баға

Дипломдық жұмысқа – 95% «өте жақсы» деген баға қоямын және Шералиева Мерей – 6B06103«Математикалық және компьютерлік модельдеу» мамандығы бойынша бакалавр біліктілігіне лайық деп санаймын.

Сын-пікір беруші
ҚР ҒЖБМ ҒК Ақпараттық және есептеуіш технологиялар институтының аға ғылыми қызметкері, PhD _____ Алғазы К.Т.
«29» қаңтар 2024 ж.

Университеттің жүйе администраторы мен Академиялық мәселелер департаменті директорының ұқсастық есебіне талдау хаттамасы

Жүйе администраторы мен Академиялық мәселелер департаментінің директоры көрсетілген еңбекке қатысты дайындалған Плагиаттың алдын алу және анықтау жүйесінің толық ұқсастық есебімен танысқанын мәлімдейді:

Автор: Шералиева Мерей Мұханқызы

Тақырыбы: Кохонен желілерін қолданатын классификациялар. Кохонен қабатының моделін зерттеу

Жетекшісі: Ляззат Лукпанова

1-ұқсастық коэффициенті (30): 2.1

2-ұқсастық коэффициенті (5): 0

Дәйексөз (35): 0.5

Әріптерді ауыстыру: 2

Аралықтар: 0

Шағын кеңістіктер: 0

Ақ белгілер: 0

Ұқсастық есебін талдай отырып, Жүйе администраторы мен Академиялық мәселелер департаментінің директоры келесі шешімдерді мәлімдейді :

☐ Ғылыми еңбекте табылған ұқсастықтар плагиат болып есептелмейді. Осыған байланысты жұмыс өз бетінше жазылған болып санала отырып, қорғауға жіберіледі.

☐ Осы жұмыстағы ұқсастықтар плагиат болып есептелмейді, бірақ олардың шамадан тыс көптігі еңбектің құндылығына және автордың ғылыми жұмысты өзі жазғанына қатысты күмән тудырады. Осыған байланысты ұқсастықтарды шектеу мақсатында жұмыс қайта өңдеуге жіберілсін.

☐ Еңбекте анықталған ұқсастықтар жосықсыз және плагиаттың белгілері болып саналады немесе мәтіндері қасақана бұрмаланып плагиат белгілері жасырылған. Осыған байланысты жұмыс қорғауға жіберілмейді.

Негіздеме:

Күні *Кафедра меңгерушісі*

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ
БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

«Қ.И.Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті» коммерциялық емес акционерлік қоғамы

Автоматика және ақпараттық технология институты
Жоғары математика және модельдеу кафедрасы

ҚОРҒАУҒА ЖІБЕРІЛДІ
Математика кафедрасының меңгерушісі физика-математика ғылымдарының кандидаты,қауымдастырылған профессор Тулешева Г.А.
"31" 05 2024ж

ДИПЛОМДЫҚ ЖҰМЫС

Тақырыбы: «Кохонен желілерін қолданатын классификациялар. Кохонен қабатының моделін зерттеу»

6B06103 - Математикалық және компьютерлік модельдеу

Орындаған Shey Шералиева М. М.

Рецензент

ҚР ҒҒБМ ҒК Ақпараттық және есептеуіш технологиялар институтының аға ғылыми қызметкері, PhD
К.Т. Алғазы
«29» мамыр 2024 ж.

Ғылыми жетекші

Техника ғылымдарының магистрі, аға оқытушы
Я.Х. Лукпанова
«30» мамыр 2024 ж.

Алматы 2024

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ
БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

"Қ.И.Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті''
коммерциялық емес акционерлік қоғамы

Автоматика және ақпараттық технология институты
Жоғары математика және модельдеу кафедрасы

6В06103 – Математикалық және компьютерлік модельдеу

БЕКІТЕМІН
Математика кафедрасының
меңгерушісі физика- математика
ғылымдарының, қауымдастырылған
профессор Тулешева Г.А.
"31" 05 2024ж

Дипломдық жұмыс орындауға
ТАПСЫРМА

Білім алушы: Шералиева Мерей Мұханқызы
Тақырыбы: Кохонен желілерін қолданатын классификациялар. Кохонен қабатының моделін зерттеу
Университет ректорының 2023 жылғы "4" желтоқсандағы № 548-П/Ө бұйрығымен бекітілген.
Аяқталған жұмысты тапсыру мерзімі: 2024 жылғы" 3-6 " маусым.
Дипломдық жобаның бастапқы деректері:
Дипломдық жұмыста әзірленуге жататын мәселелердің тізбесі немесе дипломдық жұмыстың қысқаша мазмұны:
А Жұмысқа жалпы шолу
Б Жұмыстың құрылым бөлімі
В Алынған мәліметтерге негізделген қорытынды
Ұсынылатын негізгі әдебиеттер 7 кітап.

Дипломдық жұмысты дайындау

КЕСТЕСІ

Бөлімдердің атаулары, әзірленетін мәселелердің тізбесі	Ғылыми жетекшіге ұсыну мерзімдері	Ескерту
Тақырыпқа байланысты арнайы әдибеттерді карастыру	03.02.2024	орындалды
Дипломдық жұмыстың жоспарын құру	06.02.2024	орындалды
Негізгі бөлімді карастыру	01.03.2024	орындалды
Дипломдық жұмысты қорытындылау	16.05.2024	орындалды

Қолтаңбалар

Аяқталған дипломдық жобаға консультанттар мен нормоконтролер оларға катысты жұмыс бөлімдерін көрсете отырып

Бөлімдердің атаулары	Кеңесшілер Т.А.Ә. (мұғ. дәрежесі, атағы)	Қол қойылған Күні	Қолы
Негізгі бөлім	Лукпанова Л.Х. «Жоғары математика және моделдеу» кафедрасының аға оқытушысы	30.05.2024	
Норма бақылаушы	Шатманов Ж.Ж. физ.-мат. ғылымдарының кандидаты,қауымдастырылған профессор	29.05.2024	

Ғылыми жетекшісі _____ Л.Х.Лукпанова

Білім алушы тапсырманы орындауға алды _____ Ш.М.Мұханқызы

Күні «30» наурыз 2024ж

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ
БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

«Қ.И.Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті» коммерциялық емес акционерлік қоғамы

Автоматика және ақпараттық технология институты
Жоғары математика және модельдеу кафедрасы

Шералиева Мерей Мұханқызы

Кохонен желілерін қолданатын классификациялар. Кохонен қабатының моделін зерттеу

ДИПЛОМДЫҚ ЖҰМЫС

6B06103 - Математикалық және компьютерлік модельдеу

Алматы 2024

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ
БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

«Қ.И.Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті» коммерциялық емес акционерлік қоғамы

Автоматика және ақпараттық технология институты
Жоғары математика және модельдеу кафедрасы

ҚОРҒАУҒА ЖІБЕРІЛДІ
Математика кафедрасының
меңгерушісі физика-
математика ғылымдарының
кандидаты,қауымдастырылған
профессор Тулешева Г.А.
_____
«___» _____2024ж

ДИПЛОМДЫҚ ЖҰМЫС

Тақырыбы: «Кохонен желілерін қолданатын классификациялар. Кохонен қабатының моделін зерттеу»

6В06103 - Математикалық және компьютерлік модельдеу

Орындаған
Шералиева М.М.

Рецензент
Аға ғылыми қызметкер,
Философия ғылымдарының докторы (PhD)
_____К.Т. Алгазы
«____» маусым 2024 ж.

Ғылыми жетекші
«Жоғары математика және моделдеу» кафедрасының
аға оқытушысы
_____Л.Х. Лукпанова
«___» маусым 2024 ж.

Алматы 2024

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ
БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

«Қ.И.Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті»
коммерциялық емес акционерлік қоғамы

Автоматика және ақпараттық технология институты
Жоғары математика және модельдеу кафедрасы

6В06103 – Математикалық және компьютерлік модельдеу

БЕКІТЕМІН
Математика кафедрасының
меңгерушісі физика- математика
ғылымдарының, қауымдастырылған
профессор Тулешева Г.А.
_____
«____» _____2024ж

Дипломдық жұмыс орындауға
ТАПСЫРМА

Білім алушы: Шералиева Мерей Мұханқызы
Тақырыбы : Кохонен желілерін қолданатын классификациялар. Кохонен қабатының моделін зерттеу.
Университет ректорының 2023 жылғы «4» желтоқсандағы № 548-П/Ө бұйрығымен бекітілген.
Аяқталған жұмысты тапсыру мерзімі: 2024 жылғы «3-6» маусым.
Дипломдық жобаның бастапқы деректері:
Дипломдық жұмыста әзірленуге жататын мәселелердің тізбесі немесе дипломдық жұмыстың қысқаша мазмұны:
А Жұмысқа жалпы шолу
Б Жұмыстың құрылым бөлімі
В Алынған мәліметтерге негізделген қорытынды
Ұсынылатын негізгі әдебиеттер 7 кітап.

Дипломдық жұмысты дайындау

КЕСТЕСІ

Бөлімдердің атаулары, әзірленетін мәселелердің тізбесі	Ғылыми жетекшіге ұсыну мерзімдері	Ескерту
Тақырыпқа байланысты арнайы әдибеттерді қарастыру	03.02.2024	орындалды
Дипломдық жұмыстың жоспарын құру	06.02.2024	орындалды
Негізгі бөлімді қарастыру	01.03.2024	орындалды
Дипломдық жұмысты қорытындылау	16.05.2024	орындалды

Қолтаңбалар

Аяқталған дипломдық жобаға консультанттар мен нормоконтролер оларға қатысты жұмыс бөлімдерін көрсете отырып

Бөлімдердің атаулары	Кеңесшілер Т.А.Ә. (мұғ. дәрежесі, атағы)	Қол қойылған Күні	Қолы
Негізгі бөлім	Лукпанова Л.Х. «Жоғары математика және моделдеу» кафедрасының аға оқытушысы		
Норма бақылаушы	Шатманов Ж.Ж. физ.-мат. ғылымдарының кандидаты,қауымдастырылған профессор		

Ғылыми жетекшісі _____ Л.Х.Лукпанова

Білім алушы тапсырманы орындауға алды _____ Ш.М.Мұханқызы

Күні «___»_____2024ж

АҢДАТПА

Кохонен желілері-бұл мұғалімсіз оқытуды білетін және деректерді кластерлеу және жіктеу үшін қолданылатын нейрондық желілер. Бұл жұмыс Кохонен желілерін қолдана отырып жіктеу әдісін қарастырады. Негізгі назар жіктеу міндеттері, олардың артықшылықтары мен шектеулері тұрғысынан кохонен желілерінің жұмыс принциптеріне аударылады. Кохонен желілерін әртүрлі деректер түрлері үшін зерттеу және пайдалану ерекшеліктері талданады, сонымен қатар осы жіктеу әдісін қолданудың заманауи тәсілдері мен тенденцияларына талдау жасалады. зерттеу нәтижелері деректерді жіктеуді қажет ететін әртүрлі салаларда Кохонен желілерін практикалық пайдалану үшін пайдалы болуы мүмкін.

АННОТАЦИЯ

Сети Кохонена представляют собой нейронные сети, умеющие к обучению без учителя и используемые для кластеризации и классификации данных. В данной работе рассматривается способ классификации с использованием сетей Кохонена. Основной упор делается на принципах работы сетей Кохонена в контексте классификационных задач, их преимуществах и ограничениях. Анализируются особенности изучения и использования сетей Кохонена для разных видов данных, а вдобавок проводится анализ современных подходов и тенденций в применении данного способа классификации. итоги изучения могут быть полезны для практического использования сетей Кохонена в различных областях, требующих классификации данных.

ABSTRACT

Kohonen networks are neural networks capable of learning without a teacher and used for clustering and classifying data. In this paper, a classification method using Kohonen networks is considered. The main focus is on the principles of Kohonen networks in the context of classification tasks, their advantages and limitations. The features of the study and use of Kohonen networks for different types of data are analyzed, and in addition, an analysis of modern approaches and trends in the application of this classification method is carried out. The results of the study can be useful for the practical use of Kohonen networks in various fields requiring data classification.

МАЗМҰНЫ

КІРІСПЕ

Кохонен қабатының моделі деректерді жіктеу мен кластерлеудің қиын мәселелерін шешу үшін Кохонен желілерінің әлеуетін кеңейтудің қызықты тәсілі болып табылады. Соңғы он жылдықтарда кохонен желілері ғалымдардың қызығушылығын тәрбиешісіз оқу қабілетіне және деректер құрылымын көрсету қабілетіне байланысты қызықтырды. Алайда, осы жіктеу есептерін сәтті шешу үшін көбінесе алгоритмнің икемділігі мен қуаты қажет.

Бұл зерттеуде біз Кохонен қабаттарының моделіне назар аударамыз, бұл Кохонен желілерінің классикалық модификациясын қосымша қабаттар мен механизмдерді қосу тәртібімен нақтылауды елестетеді. Кохонен қабаттарының моделі желілерге мәліметтерді тереңірек зерттеуге және күрделі үлгілерді анықтауға мүмкіндік береді, бұл оны әртүрлі жіктеу мәселелерін шешуге арналған күшті құрал етеді.

Бұл кіріспеде біз Кохонен қабаттарының модификациясын зерттеуде не қамтылатынын атап өтеміз. Біз модельдің негізгі принциптерін, оның классикалық Кохонен желілерімен салыстырғанда артықшылықтарын, сондай-ақ әртүрлі салаларда тиімді пайдалану үлгілерін қарастырамыз. Сонымен қатар, біз қазіргі технологиялар мен нарық талаптары контекстінде ұсынылған модификацияның ағымдағы сын-тегеуріндері мен даму бағыттарына назар аударамыз.

1 Кохонен желілері

Кохонен желілерінің ең танымал түрлері архитектурасы мен оқыту алгоритмдері қарастырылады: кіріс векторларын кластерлеуді жүзеге асыратын Кохонен желілері, жазықтықта көпөлшемді деректерді визуализациялауды жүзеге асыратын Кохоненнің өзін — өзі ұйымдастыратын карталары (som-self - gganizing maps), мұғаліммен оқытылатын векторлық кванттау желілері (LVQ — learning vector quantization), қарсы желі тарату

1.1 Кохонен желісін құру принциптері

Кохонен желілері өзін-өзі ұйымдастыратын нейрондық желілерге жатады. Өзін-өзі ұйымдастыратын желі кейбір жалпы қасиеттері бар кіріс векторларының кластерлерін анықтауға мүмкіндік береді.

Кластерлеу-зерттелетін объектілер жиынтығын кластерлер деп аталатын "ұқсас" объектілер топтарына бөлу. "Кластер" терминінің синонимдері (ағылш. Кластер-тромб, байлам, топ) - класс, таксон, қоюлану терминдері. Кластерлеу міндеті жіктеу міндетінен түбегейлі ерекшеленеді. Жіктеу мәселесінің шешімі объектілердің әрқайсысын алдын-ала анықталған сыныптардың біріне жатқызу болып табылады. Кластерлеу тапсырмасында объектіні алдын-ала анықталмаған сыныптардың біріне жатқызу орын алады. Объектілерді кластерлер бойынша бөлу бір мезгілде кластерлерді қалыптастыру кезінде жүзеге асырылады.

Кластерлеу ұқсас деректерді топтастыруға мүмкіндік береді, бұл бірқатар Data Mining мәселелерін шешуді жеңілдетеді:

- Деректерді зерттеу, талдауды жеңілдету. Алынған кластерлердің мазмұнды талдауы үлгілерді анықтауға мүмкіндік береді. Мысалы, ұялы байланыс желісінің клиенттерінің топтарын анықтауға болады, олар үшін жаңа жоспар ұсынуға болады. Басқа мысалдар - сауда желісін сатып алушылар тобын анықтау, нарықты сегментациялау. Кластер мазмұнын талдау әртүрлі кластер объектілеріне әртүрлі талдау әдістерін қолдануға мүмкіндік береді.
- Болжау. Жаңа объектіні кластерлердің біріне жатқызу арқылы объектінің мінез-құлқын болжауға болады, өйткені оның мінез-құлқы кластер объектілеріне ұқсас болады.
- Аномалияларды анықтау. Кластерлерді мазмұнды талдау ауытқуларды анықтауға көмектеседі. Әдетте, бұл бірнеше объектілер кіретін кластерлер.

Әр кластердің мағыналы интерпретациясының рөлін атап өту маңызды. Әр кластерге кластер объектілерінің мәнін көрсететін мағыналы атау берілуі керек. Ол үшін объектілерді кластерге біріктіретін белгілерді анықтау қажет. Бұл кластер объектісінің қасиеттерін статистикалық талдауды қажет етуі мүмкін.

Кохонен желілерінің көмегімен сандық сипаттамалармен сипатталған объектілер кластерленеді.

Ресми түрде кластерлеу мәселесі келесідей сипатталады. $I = \{i_1, i_2, \ldots i_n\}$

$I = \{i_1, i_2, ..., i_n\}$, мәндерінің көптеген объектілері берілген, олардың әрқайсысы x_j вектормен сипатталады, $j = 1, 2, ..., n$ параметрелі. Көптеген С кластерлерін құру қажет және I жиынының F жиынын с жиынына көрсету қажет, яғни $F : I \to C$. Кластерлеудің міндеті көптеген жиынтықты құру

$$C = \{c_1, c_2, ..., c_k, ..., c_g\}$$

мұндағы c_k - I жиынынан "ұқсас" нысандарды қамтитын кластер :

$$c_k = \{i_j, i_p \mid i_j \in I, i_p \in I \; u \; d(i_j, i_p) < \sigma\}, \qquad (1)$$

σ - объектілерді бір кластерге қосу үшін жақындық өлшемін анықтайтын шама, $d(i_j, i_p)$ - нысандар арасындағы қашықтық деп аталатын жақындық өлшемі.

Егер $d(i_j, i_p)$ қашықтығы белгілі бір s мәнінен аз болса, онда объектілер жақын болып саналады және бір кластерге орналастырылады. Әйтпесе, нысандар бір-бірінен ерекшеленеді және әртүрлі кластерлерге орналастырылады деп саналады. Шарт (1) актамдық гипотеза ретінде белгілі.

Кластерлеу векторлар арасындағы қашықтықты пайдалануға негізделген.

Теріс емес $d(x, y)$ саны, егер келесі шарттар орындалса, x және y векторлары арасындағы қашықтық (метрика) деп аталады.

1) $d(x, y) \geq 0$ барлығына х және у
2) $d(x, y) = 0$ сонда ғана $x = y$
3) $d(x, y) = d(y, x)$.
4) $d(x, y) \leq d(x, k) + d(k, y)$ - үшбұрыштың теңсіздігі.

Кохонен желілерінде әдетте евклидтік қашықтық қолданылады:

$$d_E(x, y) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2} = \| x - y \|. \quad (1.1)$$

X және y векторлары арасындағы евклидтік қашықтық-векторлар айырмашылығының евклидтік нормасы немесе x және y нүктелерін байланыстыратын сегменттің ұзындығы.

Евклидтік қашықтық Минковский қашықтығының ерекше жағдайы

$$d_p(x, y) = (\sum_{i=1}^{m} | x_i - y_i |^p)^{1/p} = \| x - y \|_p, \quad (1.2)$$

Мұндағы $\| z \|_p = (\sum_{i=1}^{m} | z_i |^p)^{1/p}$ - $p - z$ векторының нормасы.

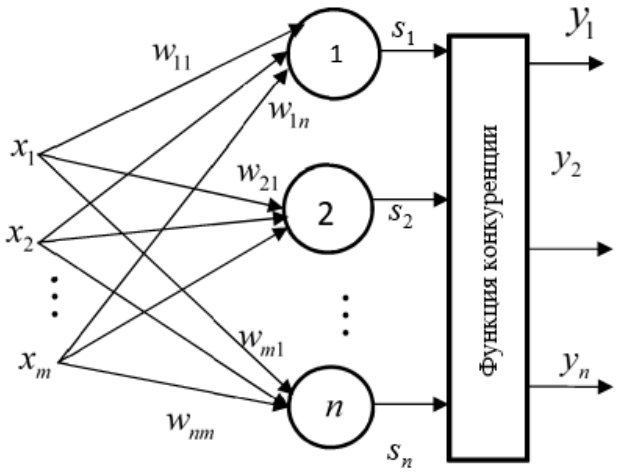
Сонда 2 — норма-Евклид нормасы.

Тағы бір ерекше жағдай-1-норма, ол Манхэттен қашықтығы (қалалық кварталдар қашықтығы)деп аталады

$$d_1(x, y) = \sum_{i=1}^{m} | x_i - y_i |. \qquad (1.3)$$

Манхэттен қашықтығы-бұл манхэттендегідей немесе басқа қалалардағы сияқты координаталық осьтерге параллель жүру арқылы өтетін қашықтық. Қашықтықтың басқа түрлері де белгілі.

Кохонен желісі (сурет. 1) - бір қабатты желі



1- сурет - Кохонен желісінің құрылымы

2-

Желінің әрбір нейроны $x_i = (x_{i1}, x_{i2}, ..., x_{im})$ m-өлшемді кіріс векторының барлық компоненттеріне қосылған. Кіріс векторы-кластерленетін объектілердің бірінің сипаттамасы. Нейрондардың саны желі бөлетін кластерлер санына сәйкес келеді. Кохонен желісінің нейрондары ретінде сызықтық өлшенген қосқыштар қолданылады

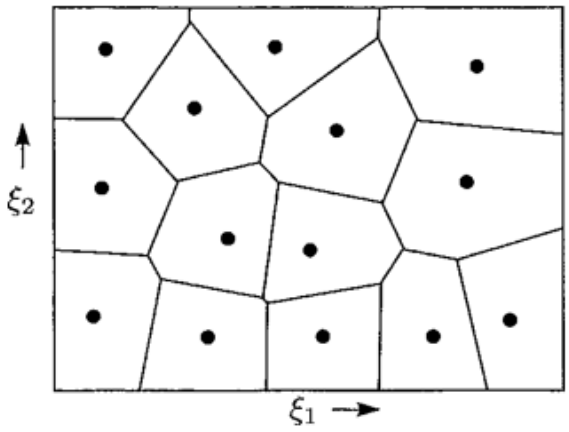
$$s_j = b_j + \sum_{i=1}^{m} w_{ij} x_i \,, \quad (1.4)$$

Мұндағы j - нейрон нөмірі. i - кіріс нөмірі, s_j - адаптивті қосқыштың шығысы, w_{ij} - i кіріс j нейрон салмағы, b_j порог.

Әрбір j нейроны w_j салмақ векторымен сипатталады $w_j = (w_{1j}, w_{2j}, ..., w_{mj})$, мұндағы m – кіріс векторларының компоненттерінің саны. Адаптивті қосқыштардың шығуынан сигнал "жеңімпаз бәрін алады"ережесі бойынша жұмыс істейтін бәсекелестік функциясына түседі. Бәсекелестік функциясы шығудың максималды мәні бар адаптивті қосқышты табады. K осындай қосқыштың нөмірі болсын. Содан кейін желінің шығысында $y_k = 1$ шығыс сигналы пайда болады, қалған шығыс сигналдары нөлге тең. Егер максимум бірнеше қосқыштардың шығуында бір уақытта қол жеткізілсе, онда бірлікке тең шығыс сигналы олардың біріне сәйкес келеді, мысалы, біріншісі.

Кохонен желісін оқыту - қолданылатын кіріс векторларының метрикасы мағынасында жақындарын таразы векторымен алмастырудан болатын қателерді азайтатын салмақ мәндерін таңдау. Бұл тәсіл векторлық кванттау деп аталады және аудио және бейне сигналдарын қысу есептерінде қолданылады. Векторлық

кванттау идеясыномам көп өлшемді кіріс векторларын кодтық кестені құрайтын кішірек өлшемді тірек векторларының шектеулі жиынтығымен ұсыну. Кохонен желісі жағдайында кіріс векторлары жеңімпаз нейрондардың нөмірлерімен (кластер нөмірлері) кодталады. Осылайша, кіріс кеңістігінің кейбір аймағындағы барлық векторлар олардың ең жақын көршісі болып табылатын бірдей тірек векторымен ауыстырылады. Евклидтік қашықтықты пайдалану кезінде кіріс кеңістігі Вороной полиэдріне бөлінеді. Екі өлшемді кеңістікке арналған Вороной политоптарының мысалы 2 суретте келтірілген.



2 - сурет - Воронный политоптарының мысалы.

Көп өлшемді кеңістікте вороной мозаикасы гиперпландар арқылы қалыптасады.

1.2 Кохонен желісін оқыту

Кохонен желілері мұғалімсіз оқытуды пайдаланады. Желіні оқыту үшін бәсекелестік тетіктері қолданылады. Вектор желісінің кірісіне берілген кезде *x* таразы векторы кіріс векторынан ең аз ерекшеленетін Нейронды жеңеді. Жеңімпаз Нейрон үшін арақатынас орындалады

$$d(x, w_j) = \min_{1 \le i \le n} d(x, w_j), \quad (1.5)$$

мұндағы *n*-нейрондардың саны, *j*-жеңімпаз нейронның нөмірі, $d(x, w)$ - *x* және *w* векторлары арасындағы қашықтық. көбінесе қашықтық өлшемі ретінде Евклид өлшемі қолданылады

$$d(x, w_j) = \| x - w_i \| = \sqrt{\sum_{j=1}^{m} (x_j - w_{ji})^2}. \quad (1.6)$$

Басқа қашықтық өлшемдері де қолданылады.

Бәсекелес активтендіру функциясы қосындылардың мәндерін талдайды және шығуда максималды мәні бар бір "жеңімпаз нейроннан" басқа барлық нейрондар үшін 0-ге тең нейрондық шығыстарды құрайды. Осылайша, Шығыс векторында жеңімпаз нейронға сәйкес келетін 1-ге тең жалғыз элемент бар, ал

қалғандары 0-ге тең. Белсенді нейронның нөмірі кіріс векторы ең жақын топты анықтайды.

Кохонен желісінде кіріс мәндерін қалыпқа келтірген жөн. Ол үшін келесі формулалардың бірін қолдану керек:

$$x_{ni} = \frac{x_i}{\sqrt{\sum_{i=1}^{m} x_i^2}},\ x_{ni} = \frac{x_i}{|x_i|}, \qquad (1.7)$$

мұндағы x_{ni} - кіріс векторының нормаланған компоненті.

Кіріс деректерін нормалау желінің оқу жылдамдығына оң әсер етеді.

Оқыту процесінің алдында желіні инициализациялау жүзеге асырылады, яғни таразы векторларының бастапқы жұмысы. Қарапайым жағдайда таразының кездейсоқ мәндері беріледі. Кохонен желісін оқыту процесі бірқатар қадамдарды циклдік қайталаудан тұрады:

1 Кірістерге бастапқы деректерді беру. Бұл әдетте кіріс векторларының бірінің кездейсоқ үлгісі.
2 Әр нейронның шығуын табу.
3 "Жеңімпаз" нейронның анықтамасы (оның салмағы кіріс векторының сәйкес компоненттерінен аз ерекшеленеді) немесе жеңімпаз Нейрон.
4 Кохонен ережесі бойынша" жеңген " нейронның салмағын түзету

5 $$x_i^{(k+1)} = w_i^{(k)} + \eta_i^{(k)}[x - w_i^{(k)}] \qquad (2)$$

6 мұндағы x — кіріс векторы, k-оқу циклінің нөмірі, $\eta_i^{(k)}$ - оқытудың k цикліндегі I-ші нейронның оқу жылдамдығының коэффициенті.
7 Егер оқу аяқталмаса, 1-қадамға өтіңіз. Көбінесе, мысалы, MATLAB-та оқу циклдарының саны беріледі. Қатенің функционалдығының аз мөлшеріне жетуді тексеруге болады

$$E = \frac{1}{Q}\sum_{i=1}^{Q} \| x_i - w_{x_i} \|^2, \qquad (3)$$

мұндағы w_{x_i} - x_i кіріс векторын көрсету кезінде жеңімпаз нейронның салмақ векторы, Q - оқу үлгісінің өлшемі.

Осылайша, таразы векторы кіріс векторына жақын болған нейрон одан да жақын болу үшін жаңартылады. Нәтижесінде, бұл нейрон жақын векторды енгізу кезінде бәсекелестікте жеңіске жетуі мүмкін және айтарлықтай ерекшеленетін векторды беру кезінде ұтылады. Оқу векторларын бірнеше рет бергеннен кейін Вектор кластерге тиесілі болған кезде 1 және вектор кластерге жатпайтын кезде 0 шығаратын нейрон болады. Осылайша, желі кіріс векторларын жіктеуді үйренеді.

Кохонен желісін оқыту кезінде "өлі" нейрондар деп аталатын мәселе туындайды. Кез-келген бәсекелес қабаттың шектеулерінің бірі-кейбір нейрондардың қатыспайтындығы. Бұл кіріс векторларынан айтарлықтай алыс бастапқы салмақ векторлары бар нейрондар оқу қанша уақытқа созылғанына қарамастан ешқашан бәсекелестікке ие болмайтындығымен көрінеді. Нәтижесінде, мұндай векторлар оқытуда қолданылмайды және сәйкес нейрондар

ешқашан жеңімпаз болып табылмайды. Мұндай " сәтсіз нейрондар "" өлі " нейтрондар деп аталады, өйткені олар ешқандай пайдалы қызмет атқармайды. Осылайша, кіріс нейрондардың аз санымен түсіндіріледі. Сондықтан барлық нейрондарды жеңуге мүмкіндік беру керек. Ол үшін оқыту алгоритмі "өлі" нейрондар оқуға қатысатындай етіп өзгертіледі.

Мысалы, оқыту алгоритмі жеңімпаз нейрон белсенділігін жоғалтатындай етіп өзгертіледі. Нейрондық белсенділікті есепке алудың бір әдісі-оқу процесінде әрбір нейронның p_i потенциалын есептеу. Бастапқыда нейрондарға $p_i(0) = \frac{1}{n}$ потенциалы беріледі, мұндағы n - нейрондардың (кластерлердің) саны. Оқытудың k циклінде потенциал ережелер бойынша анықталады:

$$p_i(k) = \begin{cases} p_i(k-1) + \frac{1}{n}, & i \neq j \\ p_i(k-1) - p_{\min}, & i = j \end{cases}, \quad (3.1)$$

мұндағы j - жеңімпаз нейронның нөмірі.

Егер $p_i(k)$ потенциалының мәні болса $p_{\min}$ деңгейінен төмен түседі, содан кейін нейрон қарастырудан шығарылады. $p_{\min} = 0$ кезінде нейрондар күрестен шығарылмайды. $p_{\min} = 1$ кезінде нейрондар кезекпен жеңеді, өйткені әрбір оқу циклінде олардың біреуі ғана күресуге дайын. Іс жүзінде жақсы нәтиже $p_{\min} \approx 0,75$ болғанда.

Matlab «өлі» нейрондармен күресу үшін нейрондардың орын ауыстыруының өзгеруі қолданылады. Өлі нейрондардың сезімталдығын ескеретін сәйкес баптау ережесі Learncon функциясы ретінде жүзеге асырылады және келесідей. Теңшеу процедурасының басында бәсекелес қабаттың барлық нейрондарына бірдей белсенділік параметрі тағайындалады $c_0 = \frac{1}{N}$, мұндағы N бәсекелес қабаттың нейрондарының саны кластерлер санына тең. Орнату процесінде learncon функциясы Бұл параметрді белсенді нейрондар үшін оның мәндері үлкенірек және белсенді емес нейрондар үшін аз болатындай етіп реттейді. Әрекет параметрлері векторының сәйкес формуласы келесідей:

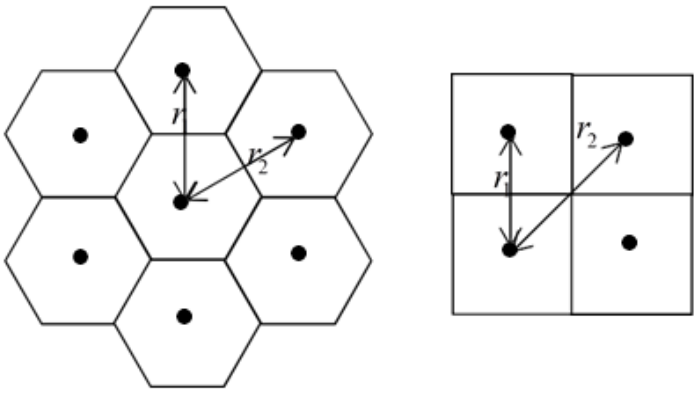
$$c^{(k+1)} = (1 - r_l)c^{(k)} + r_l s^{(k)}, \quad (3.2)$$

мұндағы r - орнату жылдамдығы параметрі; k - оқу циклінің нөмірі; $s^{(k)}$ - k-оқу цикліндегі адаптивті қосқыштардың Шығыс векторы.

2 Кохонен карталары

2.1 Кохонен карталарын құру принциптері

Кохонен карталары (өзін — өзі ұйымдастыратын карталар немесе SOM-self -gganizing map) екі өлшемді картадағы объектілердің көп өлшемді қасиеттерін визуалды түрде көрсетуге арналған. Кохонен карталары жоғары өлшемді кірістерді шағын өлшемді тұрақты массив элементтеріне (әдетте екі өлшемді) көрсетеді. Кохонен карталары кохонен желілеріне ұқсас. Айырмашылық мынада: картада кластерлердің орталықтары болып табылатын нейрондар белгілі бір құрылымға (әдетте екі өлшемді торға) реттелген. Картаны оқыту барысында жеңімпаз нейронның ғана емес, оның көршілерінің де салмағы реттеледі. Нәтижесінде Кохонен желісіндегі белгілі бір метрикаға жақын кіріс векторлары бір нейронға (кластердің ортасына) жатады, ал Кохонен картасында олар торда жақын орналасқан әртүрлі нейрондарға сілтеме жасай алады. Әдетте нейрондар тікбұрышты немесе алтыбұрышты жасушалары бар екі өлшемді тордың түйіндерінде орналасады. Көрші нейрондар картадағы нейрондар арасындағы қашықтықпен анықталады. 3 суретте орталықтарында нейрондар орналасқан алтыбұрышты және тікбұрышты жасушалар көрсетілген. Алтыбұрышты ұяшықтар картадағы объектілер арасындағы декарттық қашықтықты дұрыс көрсетеді, өйткені бұл ұяшықтар үшін іргелес ұяшықтардың центрлері арасындағы қашықтық бірдей.



3 - сурет - Алтыбұрышты және тікбұрышты ұяшықтар

Әрбір ұяшық кохонен желісінің нейронына сәйкес келеді. Яғни, Кохонен картасында нейрондар саны карта ұяшықтарының санына тең және Кохонен желісіндегі нейрондар санынан көп, кластерлер санына тең. Карта ұяшықтарының саны суреттің қажетті бөлшектеріне байланысты және эксперименталды түрде таңдалады.

Әрбір ұяшық үшін ұяшыққа түскен кіріс векторларының таңдалған компонентінің статистикалық сипаттамаларының бірі есептеледі. Осы сипаттаманың шамасына байланысты ұяшық белгілі бір түске боялады.

Кохонен карталары тек карта ұяшықтарын бояу арқылы кластерлік құрылымның болуы және кластерлер саны, жеке айнымалылардың мәндері арасындағы тәуелділіктер туралы гипотеза жасауға мүмкіндік береді. Ұсынылған гипотезалар басқа тәсілдермен тексеріліп, расталуы керек. Карталар деректерді барлау сатысында қолданылады, кез-келген нақты нәтижеге қол жеткізуден гөрі тапсырманы жалпы түсіну үшін.

2.2 Кохонен картасын оқыту

2.2.1 Дәйекті оқыту алгоритмі

Дәйекті (Iterative) оқыту алгоритмінде таразыны жаңарту әрбір оқыту мысалынан кейін жүргізіледі. Нейрондық оқыту Кохоненнің нейрондық желісін оқытуға ұқсас түрде жасалады. Айырмашылық мынада: жеңімпаз нейроннан басқа, жеңімпаз нейронның маңайына (neighborhood) немесе оқу радиусына (radius of learning) кіретін нейрондар оқытылады. Нейрон тиесілі жеңімпаз нейронның айналасы, егер ол мен картадағы жеңімпаз нейронның арасындағы қашықтық белгілі бір мәннен аз болса (оқу процесінде нейрондардың салмағы өзгереді, бірақ олардың картадағы орны өзгермейді). Мұндай алгоритм WTM типті алгоритм деп аталады. Классикалық алгоритмде жеңімпаз нейронның салмағы және оның ішінде жатқан барлық нейрондар Кохоненнің сәл өзгертілген ережесі бойынша оқытылады. Маңайдан тыс орналасқан нейрондардың салмағы өзгермейді. Көршілес өлшем және оқу жылдамдығының коэффициенті оқу циклінің санына қарай мәндері төмендейтін функциялар болып табылады.

Кохонен ережесінің өзгеруі $\eta_i^{(k)}$ оқу жылдамдығының коэффициенті екі бөлікке бөлінеді: көршілік функциясы $\eta_i(d,k)$ және оқу жылдамдығының функциясы $a(k)$

$$\eta_i^{(k)} = \eta_i(d,k) \cdot a(k). \qquad (4)$$

Көршілес функция ретінде немесе тұрақты қолданылады

$$\eta_i(d,k) = \begin{cases} const, & d_i \le \sigma(k), \\ 0, & d_i > \sigma(k) \end{cases} \qquad (5)$$

немесе Гаусс функциясы

$$\eta_i(d,k) = e^{\frac{d_i}{2\sigma^{(k)}}}.$$

Бұл жағдайда ең жақсы нәтиже қашықтықтың Гаусс функциясын қолдану арқылы алынады. (4) және (5) $d_i = \| r_i - r_{cj} \|$ - i-ші Нейрон мен c_j, r_i, r_{cj} жеңімпаз нейроны арасындағы қашықтық - i-ші және жеңімпаз c_j -ші нейрондардың карта торындағы координаттар, $\| r_i - r_{cj} \|$ - карта торындағы i және c_j ұяшықтары арасындағы қашықтық. $\sigma(k)$ функциясы - бұл оқу циклінің санынан азаятын

функция. Ең жиі қолданылатын функция-оқу циклінің нөмірінен сызықтық төмендеу.

Енді оқу жылдамдығының функциясын қарастырайық $a(k)$. Бұл функция сонымен қатар оқу циклінің санынан азаятын функция болып табылады. Бұл функцияның ең көп қолданылатын екі нұсқасы-сызықтық және түрдің оқу циклінің санына кері пропорционалды

$$a(k) = \frac{A}{k + B}, \quad (5.1)$$

мұндағы А және В-тұрақтылар. Бұл мүмкіндікті қолдану оқу үлгісіндегі барлық векторлардың оқу нәтижесіне шамамен тең үлес қосуына әкеледі.

Оқыту екі негізгі кезеңнен тұрады: бастапқы кезеңде — белгілер кеңістігіндегі салмақ коэффициенттерінің векторларын ретке келтіру кезеңінде оқу жылдамдығы мен радиусының жеткілікті үлкен мәні таңдалады оқыту, бұл нейрондық векторларды үлгідегі мысалдардың таралуына сәйкес орналастыруға мүмкіндік береді (нейрондық векторлардың белгілер кеңістігіндегі орны өзгереді, бірақ картада емес). Соңғы кезеңде — баптау кезеңінде оқу жылдамдығының параметрлерінің мәндері бастапқы деңгейден аз болған кезде таразыларды дәл баптау жүргізіледі. Оқыту желінің қателігі аз болғанша жалғасады.

2.2.2 Пакеттік оқыту алгоритмі

Қазіргі уақытта Кохонен карталарын оқыту үшін кохонен картасын пакеттік оқыту алгоритмі кеңінен қолданылады (Batch-Learning Self -gganizing Map). Бұл алгоритм алдымен барлық мысалдарды ұсынады, содан кейін таразылар жаңартылады. Алгоритм нормаланған кіріс векторларын қолданады. Алгоритм келесі қадамдар орындалатын k-ші өткелдегі бірқатар үзінділерден тұрады.

1. Барлық N кіріс векторлары беріледі және әрбір x_j кіріс векторы мен барлық нейрондардың w_i салмақ векторлары арасындағы евклидтік қашықтық есептеледі. Жеңімпаз нейронның нөмірі анықталады

$$c_j = \arg\min\left\{\| w_i - x_j \|\right\} \quad (5.2)$$

2. Барлық векторлар барлық кіріс векторларының компоненттерінің орташа мәндері ретінде жаңартылады

$$\mathbf{w}_i^{new} = \frac{\sum_{j=1}^{N} h_{c_j,i}\mathbf{x}_j}{\sum_{j=1}^{N} h_{c_j,i}}, \quad (5.3)$$

мұндағы $h_{cj,i}$ - көршілес функция

$$h_{c_j,i} = \exp\left(-\frac{\left\|\mathbf{r}_i - \mathbf{r}_{c_j}\right\|^2}{2\sigma^2(k)}\right), \quad (5.4)$$

r_i и r_{cj} - i-ші және жеңімпаз c_j -ші нейрондардың карта торындағы координаттар,

$\| r_i - r_{cj} \|$ - карта торындағы i және c_j ұяшықтары арасындағы қашықтық, $\sigma(k)$ параметріжаңа өту нөмірінің ұлғаюымен азаяды.

$$\sigma(k) = \sigma_0(1 - k / k_{max}), \quad (5.5)$$

σ_0 - таңдалатын бастапқы мән, k_{max} - өтулердің максималды саны.

Функционалдылық жеткілікті аз болғанша өтулер қайталанады.

2.2.3 Нейрондық газ алгоритмі

Оның динамикасының газ қозғалысына ұқсастығына байланысты аталған нейрондық газ алгоритмі қарастырылған алгоритмдерге қарағанда конвергенция жылдамдығын арттырады. Бұл алгоритмде әрбір k-оқу циклінде барлық нейрондар жеңімпаз нейроннан қашықтығы бойынша сұрыпталады. Әрбір Нейрон үшін көршілік функциясының мәні анықталады.

i-ші Нейрон үшін бұл функция келесідей

$$\eta_i^{(k)} = e^{-\frac{m_i}{\lambda^{(k)}}}, \quad (6)$$

мұндағы m_i - нейрон нөмірі, сұрыптау нәтижесінде алынған (жеңімпаз Нейрон үшін бұл сан нөлге тең); λ - сериялық оқыту алгоритміндегі параметрге ұқсас, Итерация нөмірімен кішірейтілген параметр (ені параметрі).

i-ші нейронның салмақ векторын түзету формула бойынша жүзеге асырылады

$$w_i^{(k+1)} = w_i^{(k)} + \eta_i^{(k)} a_i^{(k)} [x^k - w_i^{(k)}] \quad (6.1)$$

мұндағы $\eta_i^{(k)}$ - көршілік функциясы, (6)формула бойынша анықталады; $a_i^{(k)}$ - оқу жылдамдығының функциясы.

Жеңімпаз нейронның салмақ векторы WTA алгоритмімен нақтыланған. Бірақ WTA алгоритмінен айырмашылығы, барлық нейрондардың салмағы нақтыланады.

Қайталану санының өсуімен $\lambda^{(k)}$ және $a_i^{(k)}$ параметрлері азаюы керек

$$\lambda^{(k)} = \lambda_{\max} \left(\frac{\lambda_{\min}}{\lambda_{\max}} \right)^{k/k_{\max}}, \quad a_i^{(k)} = a_i^{(0)} \left(\frac{a_{\min}}{a_i^{(0)}} \right)^{k/k_{\max}}, \quad (6.2)$$

мұндағы k - ағымдағы Итерация нөмірі, $k_{\max}$ - берілген итерациялардың максималды саны, $\lambda_{\min}$ и $\lambda_{\max}$ - қабылданған минималды және максималды параметр мәндері λ, $a_i^{(0)}$ - оқу жылдамдығының бастапқы мәні, $a_{\min}$ - $k_{\max}$ сәйкес келетін оқу жылдамдығының берілген минималды мәні.

$m_i > k$ кезінде есептеу көлемін азайту үшін, мұндағы k – берілген, болжам бойынша $\eta_i^{(k)} = 0$. Яғни, тек k нейрондары реттеледі. Қарастырылған барлық Алгоритмдер нейрондардың санын анықтауды талап етті. Өсіп келе жатқан

(кеңейетін) нейрондық газдың (Growing Neural Gas) алгоритмі кіріс кластерін жасауға ғана емес, сонымен қатар желіні оқыту процесінде нейрондардың санын есептеуге мүмкіндік береді.

2.2.4 Кохонен картасын инициализациялау

Кохонен картасын жасамас бұрын нейрондар торының конфигурациясын (әдетте алтыбұрышты) және карта нейрондарының санын орнату керек. Нейрондар саны картаның егжей-тегжейлі дәрежесін анықтайды. Айтуынша, нейрондары көп карта көп оқу уақытын қажет етеді.

Әрі қарай инициализация жасалады-нейрондардың салмағына бастапқы мәндерді тағайындау. Қарапайым жағдайда салмақты шағын кездейсоқ сандармен инициализациялауға болады. Желінің оқу уақытын қысқарту тұрғысынан картаның бастапқы күйіне кез келген тапсырыс беру пайдалы екені белгілі. Ең жақсы нәтижелер оқу үлгісінен кездейсоқ таңдалған мысалдардың мәндерінің бастапқы салмақ мәндері ретінде пайдаланылады. Карта салмағының реттелген бастапқы күйін қамтамасыз ететін сызықтық инициализация деп аталатын пайдалы. Ол үшін кіріс векторлары центрленгенге, яғни нөлдік математикалық күтілетін векторларға айналады, орталықтандырылған кіріс векторларынан тұратын $x = [x_1, x_2, ..., x_n]$, матрицасы құрылады және ковариациялық матрица есептеледі

$$K = \frac{1}{n-1} XX^T \quad (6.3)$$

K матрицасының ең үлкен екі меншікті мәні бар (бұл мәндер оң, өйткені матрица оң анықталған). Тікбұрышты немесе алтыбұрышты тұрақты торы бар тіктөртбұрыш салынады, оның өлшемдері k матрицасының ең үлкен екі меншікті мәніне тең. Нейрондық салмақтардың бастапқы мәндері ең үлкен меншікті мәндерге сәйкес келетін *K* матрицасының екі меншікті векторының сызықтық комбинациясы ретінде құрылады. Сызықтық комбинациялардың салмақтары салынған тіктөртбұрыштағы нейронның координаттарына тең алынады. Математикалық тұрғыдан бұл салмақтың бастапқы мәндері *K* матрицасының екі негізгі меншікті векторына созылған ішкі кеңістіктен алынады дегенді білдіреді.

Таразы векторларының құрамдас бөліктерінің бастапқы мәндері реттеледі және карта салмақтарын жақсы жақындатады. Сондықтан оқытуды орнату кезеңінен тікелей бастауға болады. Бірақ сызықтық инициализация салмағы ешқандай кіріс векторларына жақын емес" бос " нейрондарды тудыруы мүмкін. Таразының одан да жақсы бастапқы мәндері негізгі компоненттердің сызықтық емес әдісін қолдану арқылы беріледі.

Кохонен картасын инициализациялау кластерлеу алгоритмдерінің бірін қолдану арқылы жүзеге асырылуы мүмкін, мысалы, *k-means*: кластерлеу алгоритмі арқылы картада қанша Нейрон болса, сонша кластер түзіледі. Әрі қарай, картаны дәл баптау жүргізіледі.

2.3 Өзін-өзі ұйымдастыру картасын (SOM) құруға және оқытуға арналған код.

```
import numpy as np
import sys, math, time
import matplotlib.pyplot as plt


class SOM(object):
    def __init__(self):
        pass

    def create(self, width, height, ch):
        self.x = width
        self.y = height
        self.ch = ch
        self.trained = False
def save(self, filename):
    # Save SOM to .npy file.
    if self.trained:
        np.save(filename, self.node_vectors)
        return True
    else:
        return False

def load(self, filename):
    # Load SOM from .npy file.
    self.node_vectors = np.load(filename)
    self.x = self.node_vectors.shape[1]
    self.y = self.node_vectors.shape[0]
    self.ch = self.node_vectors.shape[2]
    self.trained = True
    return True

def get_map_vectors(self):
    # Returns the map vectors.
    if self.trained:
        return self.node_vectors
    else:
        return False

def distance(self, vect_a, vect_b):
    if self.dist_method == 'euclidean':
        dist = np.linalg.norm(vect_a - vect_b)
    elif self.dist_method == 'cosine':
        dist = 1. - np.dot(vect_a, vect_b) / (np.linalg.norm(vect_a) * np.linalg.norm(vect_b))
    return dist

def find_maching_nodes(self, input_arr):
    # This is to be called only when the map is trained.
    if self.trained == False:
        return False
```

```
n_data = input_arr.shape[0]
locations = np.zeros((n_data, 2), dtype=np.int32)
distances = np.zeros((n_data), dtype=np.float32)

print_step = int(n_data / 20)
print_count = 0
for idx in range(n_data):

    if idx % print_step == 0:
        print_count += 1
        sys.stdout.write(f'\rFinding mathing nodes' +
         ' [' + '=' * (print_count) + '>' + '.' * (20 - print_count) + '] ')
data_vect = input_arr[idx]
min_dist = None
x = None
y = None
for y_idx in range(self.y):
    for x_idx in range(self.x):
        node_vect = self.node_vectors[y_idx, x_idx]
        dist = self.distance(data_vect, node_vect)
        if min_dist is None or min_dist > dist:
            min_dist = dist
            x = x_idx
            y = y_idx
        locations[idx, 0] = y
        locations[idx, 1] = x
        distances[idx] = min_dist

    print('Done')
    return locations, distances

def initialize_map(self):
    # Initialize map weight vectors
    ds_mul = np.mean(self.input_arr) / 0.5
    self.node_vectors = np.random.rand(self.y, self.x, self.ch) * ds_mul


def fit(self, input_arr, n_iter, batch_size=32, lr=0.25, random_sampling=1.0,
        neighbor_dist=None, dist_method='euclidean'):
    self.input_arr = input_arr
    self.n_iter = n_iter
    self.batch_size = batch_size
    self.dist_method = dist_method

    start_time = time.time()
    self.initialize_map()

    # Learning rate. This defines how fast the node weights are updated.
    self.lr = lr
    self.lr_decay = 0.8 # lr decay per iteration

    # Neighbor node coverage.
    # This tells that how far from the best matching node the
    # other nodes are updated.
    if neighbor_dist is None:
        neighbor_dist = min(self.x, self.y) / 1.3
    self.nb_dist = int(neighbor_dist)
```

```
        # Rate of neighbor coverage reduction per iteration.
        # Small values = fast decay. Large values = slow decay
        self.nb_decay = 1.5

        # Pad the vector map to allow easy array processing.
        tmp_node_vects = np.zeros((self.y + 2 * self.nb_dist, self.x + 2 * self.nb_dist, self.ch))
        tmp_node_vects[self.nb_dist : self.nb_dist + self.y,
                       self.nb_dist : self.nb_dist + self.x] = self.node_vectors.copy()
        self.node_vectors = tmp_node_vects

        # Calculate number of data points per iteration.
        # random_sampling can vary between 0 and 1. Together with index shuffling
        # it can be used to randomly select fraction of data for each iteration
        # to speed up the training.
        if random_sampling > 1 or random_sampling <= 0:
            random_sampling = 1
        n_data_pts = int(self.input_arr.shape[0] * random_sampling)
    data_idx_arr = np.arange(self.input_arr.shape[0])
    batch_count = math.ceil(n_data_pts / self.batch_size)
    n_per_report_step = int(n_data_pts / 20)

    # Main iteration loop. One iteration means that all data is
    # used once to train the map weights.
    for iteration in range(self.n_iter):

        # Update the neighbor function. Typically the neighbor coverage
        # reduces with increasing iteration number
        self.make_neighbor_function(iteration)

        # Shuffle the data indexes.
        np.random.shuffle(data_idx_arr)

        # Temporary variables
        total_dist = 0
        total_count = 0
        print_count = 0

        # Batch processing loop. The map weight update is done at the end of
        # each batch. Often the batch size is e.g. some tens of data samples.
        # Too large batch size can lead to convergence problems.
        for batch in range(batch_count):

            # Calculate steps (data points) for this batch
            steps_left = n_data_pts - batch * self.batch_size
            if steps_left < self.batch_size:
                steps_in_batch = steps_left
            else:
                steps_in_batch = self.batch_size

            # Create array for storing the best matching node indexes
            bm_node_idx_arr = np.zeros((steps_in_batch, 3), dtype=np.int32)
```

```
# Process each input data point in this batch
for step in range(steps_in_batch):

    # Print progress update on the screen
    if total_count % n_per_report_step == 0:
        print_count += 1
        sys.stdout.write(f'\rProcessing SOM iteration {iteration + 1}/{self.n_iter}' +\
                ' [' + '=' * (print_count) + '>' + '.' * (20 - print_count) + ']')
    total_count += 1

    # Get the input data and calculate distance to the best matching node in the map
    input_idx = data_idx_arr[batch * self.batch_size + step]
    input_vect = self.input_arr[input_idx]
    y, x, dist = self.find_best_matching_node(input_vect)
    bm_node_idx_arr[step, 0] = y
    bm_node_idx_arr[step, 1] = x
    bm_node_idx_arr[step, 2] = input_idx
    total_dist += dist
# Update the map weights at the end of the batch
self.update_node_vectors(bm_node_idx_arr)
     # Print the average input data distance to the best matching node in the map.
     print(f' Average distance = {total_dist / n_data_pts:0.5f}')

     # Update the learnig rate
     self.lr *= self.lr_decay

 # Remove padding from the vector map
 self.node_vectors = self.node_vectors[self.nb_dist : self.nb_dist + self.y,
                                       self.nb_dist : self.nb_dist + self.x]

 # Delete the input data array from the map instance
 del self.input_arr

 end_time = time.time()
 self.trained = True
 print(f'Training done in {end_time - start_time:0.6f} seconds.')
def update_node_vectors(self, bm_node_idx_arr):
    # This method updates the map node weights.
    # Input is one batch of best matching node coordinates and indexes to corresponsing
    # data array locations.

    for idx in range(bm_node_idx_arr.shape[0]):
        node_y = bm_node_idx_arr[idx, 0]
        node_x = bm_node_idx_arr[idx, 1]
        inp_idx = bm_node_idx_arr[idx, 2]
        input_vect = self.input_arr[inp_idx]
old_coeffs = self.node_vectors[node_y + self.y_delta + self.nb_dist, node_x + self.x_delta + self.nb_dist]

update_vect = self.nb_weights * self.lr * (np.expand_dims(input_vect, axis=0) - old_coeffs)

self.node_vectors[node_y + self.y_delta + self.nb_dist,
                  node_x + self.x_delta + self.nb_dist, :] += update_vect
```

```
def find_best_matching_node(self, data_vect):


    min_dist = None
    x = None
    y = None
    for y_idx in range(self.y):
        for x_idx in range(self.x):
            node_vect = self.node_vectors[y_idx + self.nb_dist, x_idx + self.nb_dist]
            dist = self.distance(data_vect, node_vect)
            if min_dist is None or min_dist > dist:
                min_dist = dist
                x = x_idx
                y = y_idx

    return y, x, min_dist
def make_neighbor_function(self, iteration):
    # This method creates Gaussian 'bell' shaped 3D weight array in that is
    # stored in 2D arrays. There is own array for x coordinates, y coordinates and
    # for the weight values. The coordinate zero point is at the center of the Gaussian curve.
    # The bell width reduces when iteration value increases.

    size = self.nb_dist * 2
    sigma = size / (7 + iteration / self.nb_decay)
    self.nb_weights = np.full((size * size, self.ch), 0.0)
    cp = size / 2.0
    p1 = 1.0 / (2 * math.pi * sigma ** 2)
    pdiv = 2.0 * sigma ** 2
    y_delta = []
    x_delta = []
    for y in range(size):
        for x in range(size):
            ep = -1.0 * ((x - cp) ** 2.0 + (y - cp) ** 2.0) / pdiv
            value = p1 * math.e ** ep
            self.nb_weights[y * size + x] = value
            y_delta.append(y - int(cp))
            x_delta.append(x - int(cp))
    self.x_delta = np.array(x_delta, dtype=np.int32)
    self.y_delta = np.array(y_delta, dtype=np.int32)

    self.nb_weights -= self.nb_weights[size // 2]
    self.nb_weights[self.nb_weights < 0] = 0
    self.nb_weights /= np.max(self.nb_weights)
```

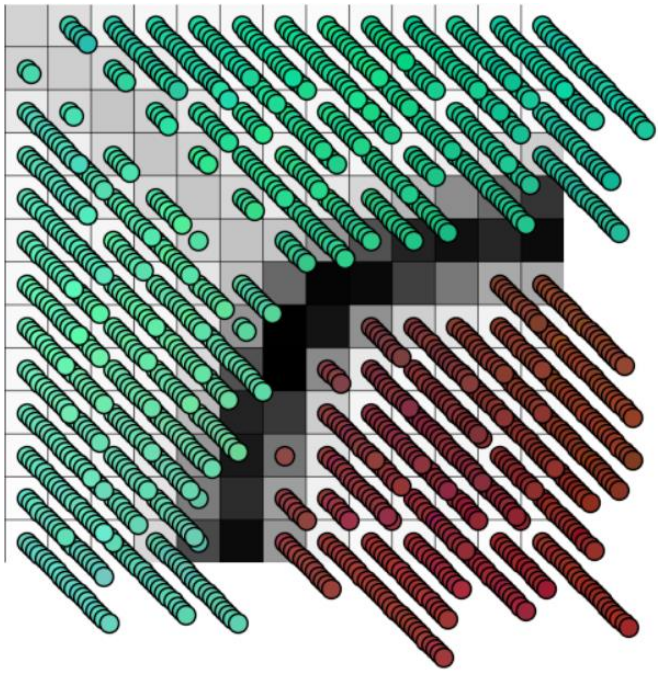
```
def get_umatrix(self):
    # This method creates a map of average vector distances from each node to the nodes
    # above, below, left and right.

    if not self.trained:
        return False

    umatrix = np.zeros((self.y, self.x))

    for map_y in range(self.y):
        for map_x in range(self.x):

            n_dist = 0
            total_dist = 0
if map_y > 0:
    dist_up = self.distance(self.node_vectors[map_y, map_x],
                            self.node_vectors[map_y - 1, map_x])
    total_dist += dist_up
    n_dist += 1

if map_y < self.y - 1:
    dist_down = self.distance(self.node_vectors[map_y, map_x],
                              self.node_vectors[map_y + 1, map_x])
    total_dist += dist_down
    n_dist += 1
        if map_x > 0:
            dist_left = self.distance(self.node_vectors[map_y, map_x],
                                      self.node_vectors[map_y, map_x - 1])
            total_dist += dist_left
            n_dist += 1

        if map_x < self.x - 1:
            dist_right = self.distance(self.node_vectors[map_y, map_x],
                                       self.node_vectors[map_y, map_x + 1])
            total_dist += dist_right
            n_dist += 1

        avg_dist = total_dist / n_dist
        umatrix[map_y, map_x] = avg_dist

 return umatrix
```

```
    def get_component_plane(self, component):
        if not self.trained:
            return False
        cplane = self.node_vectors[:, :, component].copy()
        return cplane


def plot_data_on_map(umatrix, data_locations, data_colors, data_labels=None,
                     node_width=20,
                     node_edge_color=0,
                     data_marker_size=100,
                     invert_umatrix=True,
                     plot_labels=False,
                     dpi=100):

    map_x = umatrix.shape[1]
    map_y = umatrix.shape[0]
    canvas = np.zeros((map_y * node_width, map_x * node_width))

    tmp_umatrix = umatrix.copy()
    tmp_umatrix -= np.min(tmp_umatrix)
    tmp_umatrix /= np.max(tmp_umatrix)

    if invert_umatrix:
        tmp_umatrix = 1 - tmp_umatrix

    for y in range(map_y):
        for x in range(map_x):
            canvas[y * node_width : (y + 1) * node_width,
                   x * node_width : (x + 1) * node_width] = tmp_umatrix[y, x]

if not node_edge_color is None:
    # Draw node borders
    for y in range(map_y):
        canvas[y * node_width, :] = node_edge_color
```

```
    for x in range(map_x):
        canvas[:, x * node_width] = node_edge_color

# Plot the SOM u-matrix as background
plt.figure(figsize=(map_x * node_width / dpi, map_y * node_width / dpi), dpi=dpi)
plt.imshow(canvas, cmap='gray', interpolation='hanning')

# Initialize some temp variables
item_count_map = np.zeros(umatrix.shape)
n_data_pts = data_locations.shape[0]

for i in range(n_data_pts):

    x = data_locations[i, 1]
    y = data_locations[i, 0]
    items_in_cell = item_count_map[y, x]
    item_count_map[y, x] += 1
    x = x * node_width + node_width // 2 + items_in_cell * 5
    y = y * node_width + node_width // 2 + items_in_cell * 5
    plt.scatter(x, y, s=data_marker_size, color=data_colors[i], edgecolors=[0,0,0])

    if plot_labels:
        plt.annotate(str(data_labels[i]), (x + 8, y), size='small')

    plt.axis('off')

filename = 'SOM_mapping_' + str(int(time.time())) + '.png'
plt.savefig(filename)
plt.show()
print(f'Image saved to {filename}')
```

2.3.1 Алгоритм нәтижесі

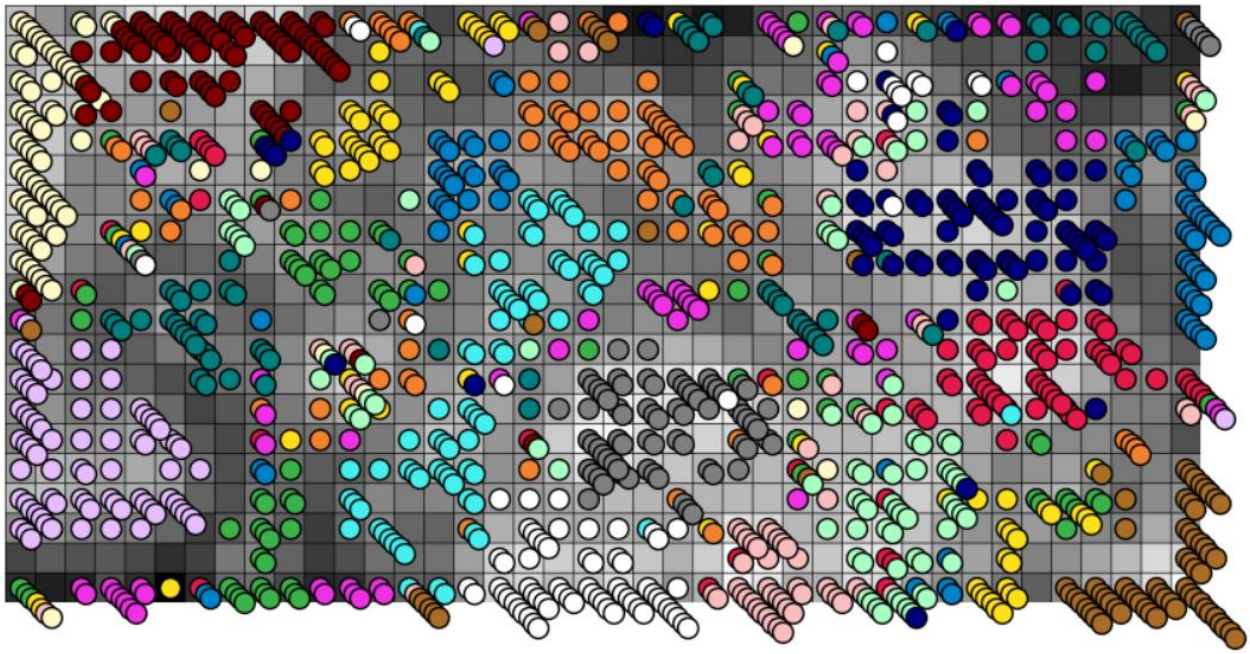
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
import random, time, sys, math
from som import SOM, plot_data_on_map
from mpl_toolkits.mplot3d import Axes3D
from sklearn.datasets import make_blobs
# Define colors for the data points. In this case we just scale the values to [0, 1]
colors = dataset.copy()
colors -= np.min(colors)
colors /= np.max(colors)

plot_data_on_map(umatrix, data_locations, data_colors=colors,
                 node_width=50, data_marker_size=100)
```



4 – сурет

Бұл код SOM-да деректерді визуализациялайды, мұнда әр нүктенің түсі бастапқы деректер жиынтығынан қалыпқа келтірілген мәнге сәйкес келеді және түйіндердің өлшемдері мен ені node_width және data_marker_size параметрлерімен анықталады.

```
# Load labels for visualization purposes. There have been pre-processed from the flower dataset matlab files.
labels = np.load('flower_labels.npy')

# Define color palette. The values are selected so that those are easily distinguishable from each other.
palette = np.array([[0.90196078, 0.09803922, 0.29411765],
                    [0.23529412, 0.70588235, 0.29411765],
                    [1.        , 0.88235294, 0.09803922],
                    [0.        , 0.50980392, 0.78431373],
                    [0.96078431, 0.50980392, 0.18823529],
                    [0.2745098 , 0.94117647, 0.94117647],
                    [0.94117647, 0.19607843, 0.90196078],
                    [0.98039216, 0.74509804, 0.74509804],
                    [0.        , 0.50196078, 0.50196078],
                    [0.90196078, 0.74509804, 1.        ],
                    [0.66666667, 0.43137255, 0.15686275],
                    [1.        , 0.98039216, 0.78431373],
                    [0.50196078, 0.        , 0.        ],
                    [0.66666667, 1.        , 0.76470588],
                    [0.        , 0.        , 0.50196078],
                    [0.50196078, 0.50196078, 0.50196078],
                    [1.        , 1.        , 1.        ]])


# Define color for each image vector index.
colors = np.zeros((n_images, 3))
for i in range(n_images):
    colors[i] = palette[labels[i] - 1]

# Plot the vector locations on the map using the colors defined above
plot_data_on_map(umatrix, data_locations, data_colors=colors,
                 node_width=30, data_marker_size=100)
```



5 – сурет

Жоғарыдағы графиктен кейбіреулер бірдей түсті шеңберлер кластері ретінде көрсетілетін сыныптардың жартысына жуығын дұрыс кластерлей алғанын көруге болады. Қалған векторлар Осы сияқты бақыланбайтын кластерлеуге қиын және олар басқа кластардың векторларымен араласады.

3 Бақыланатын векторлық кванттау желілері (LVQ желілері)

LVQ желілері білім алушы векторлық кванттауға (LVQ — Learning Vector Quantization) негізделген және мұғаліммен бірге оқитын Кохонен қабаты болып табылады. LVQ желісін құру үшін N кластерлерінің (нейрондарының) саны , m кластарының саны (n) беріледim) және әр кластердің белгілі бір сыныпқа жатуы. Мысалы, пациенттердің сынақтары 5 кластерге бөлінеді, олардың 2 — сі сау адамдарға, ал 3-і науқастарға сәйкес келеді. Кластерлерді сыныптар бойынша бөлуді оқыту үлгісіндегі сәйкес сыныптардың мысалдарын бөлу сияқты пропорцияларда жасауға болады. Кластер нөмірлері мен сынып нөмірлері арасындағы сәйкестік ерікті болуы мүмкін. Желінің нәтижелерін түсіндірудің қарапайымдылығы үшін кластерлерді дәйекті түрде нөмірлеген жөн. Біздің мысалда алғашқы екі кластер сау адамдар класына, ал келесі 3 кластағы науқастарға сәйкес келеді. Lvq оқыту процесінде нейрондық салмақ желілері оқыту мысалдары мен кластерлердің бір сыныпқа жататындығын ескере отырып реттеледі. Оқытылған LVQ желісі сыныптарды ескере отырып, кіріс векторларын кластерлейді. Мысалы, Біз белгілі бір пациенттің сынақтары науқас адамдарға сәйкес келетін кластерлердің біріне жататынын білеміз. Осы кластерге енген талдаулардың ерекшеліктерін тек осы талдауларды қарау нәтижесінде білуге болады.

LVQ желілерін оқытудың бірнеше алгоритмдері белгілі. Lvq1 деп аталатын қарапайым алгоритм *k*-оқу циклінде келесі түрге ие.

1 Оқу үлгісінің келесі *x* векторы үшін *c* нөмірі бар нейрон бар, ол үшін *x* және оның таразы векторы арасындағы евклидтік қашықтық $w_c^{(k)}$ торы минималды.
2 Жеңімпаз нейронның салмақ векторы түзетіледі:
3 – егер $w_c^{(k)}$ және *x* бір сыныпқа жатса:

4
$$\mathbf{w}_c^{(k+1)} = \mathbf{w}_c^{(k)} + \alpha^{(k)}\left[\mathbf{x} - \mathbf{w}_c^{(k)}\right], \quad (6.4)$$

5 – егер $w_c^{(k)}$ және *x* әр түрлі сыныптарға сыныптарға жатса:

6
$$\mathbf{w}_c^{(k+1)} = \mathbf{w}_c^{(k)} - \alpha^{(k)}\left[\mathbf{x} - \mathbf{w}_c^{(k)}\right], \quad (6.5)$$

7 мұндағы $a^{(k)}$ - оқу жылдамдығының коэффициенті.
8 Басқа нейрондардың салмағы өзгермейді.
9 Келесі оқу векторы таңдалады және 1-қадамға ауысады.
10 1-3 қадамдар осы уақытқа дейін қайталанады, дұрыс жіктелген векторлар саны өсуді тоқтатады.

Егер кіріс векторы желі арқылы дұрыс жіктелсе, онда сәйкес салмақ векторы кіріс векторына қарай жылжиды. Егер кіріс векторы дұрыс жіктелмесе, онда сәйкес салмақ векторы кіріс векторынан қарама-қарсы бағытта жылжиды.

Оқу жылдамдығының коэффициенті $0 < a^{(k)} < 1$ оқу циклі санының өсуімен монотонды түрде азаюы тиіс. Бірақ тіпті бастапқы мәні $a^{(k)}$ дейін аз болуы керек,

мысалы, 0,1-ден аз. LVQ2.1 алгоритмінде lvq1 алгоритмі ережелеріне сәйкес *x* кіріс векторына жақын w_i және w_j таразыларының екі векторы бір уақытта түзетіледі. Сонымен қатар, векторлардың бірі дұрыс сыныпқа жатады, ал екіншісі дұрыс емес. w_i және w_j векторларының *x* кіріс векторына жақындығының критерийі *x*-тің салыстырмалы *s* енінің "терезесіне" енуі болып табылады. Егер d_i және d_j *x*-ден w_i және w_j -ге дейінгі евклидтік қашықтық болса, онда x "терезеге" түседі, егер

$$\min\left(\frac{d_i}{d_j},\frac{d_j}{d_i}\right)>\frac{1-s}{1+s}. \qquad (6.6)$$

"Терезенің" енін 0,2-ден 0,3-ке дейін алу ұсынылады.

LVQ желілерін оқытудың басқа алгоритмдері де белгілі. Таразы векторларының бастапқы мәндері ретінде сәйкес сыныптарға жататын кездейсоқ таңдалған оқу деректер векторларын пайдалануға болады. LVQ желілерін оқыту кезінде қайта оқыту құбылысы болуы мүмкін. Көп қабатты перцептрондар сияқты, қайта оқыту құбылысы әртүрлі мысалдарда ауыспалы оқыту және тестілеу арқылы анықталады.

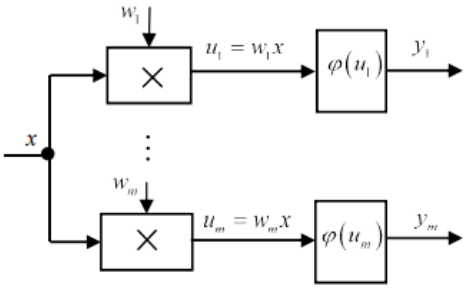
Көптеген кластерлерде LVQ желісінің нәтижелерін түсіндіру біраз күш жұмсауды қажет етеді. Сонымен қатар, көбінесе кіріс векторы кіретін класс маңызды. The екі қабаттан тұратын LVQ желісін іске асыруды қарастырады: бірінші қабат Кохонен қабаты болып табылады және берілген *n* кластер санына кіріс векторларын кластерлейді. Екінші сызықтық қабат осы кластерлерді біріктіреді *m* сыныптар ($n \geq m$). Нәтижесінде желі кіріс векторларын жіктейді. Бұл тәсіл MATLAB жүйелерінде жүзеге асырылады кіріс қабаты lvq желілерін оқыту алгоритмдерінің бірімен оқытылады. Сызықтық қабат оқытылмайды, бірақ кластер мен сынып нөмірлері арасындағы белгілі сәйкестікке байланысты қалыптасады. Бірінші қабат *n* элементтерінің векторын шығарады, олардың біреуі ғана бірлікке тең, қалғандары нөлге тең. Вектордағы бірлік элементінің нөмірі кіріс векторы берілген кластер нөміріне тең. Шығу сызықтық қабатының салмақ матрицасында *m* жолдары бар.

Матрица жолдары сыныптарға сәйкес келеді. Бағандар саны *n*. Бағандар кластерлерге сәйкес келеді. Әр бағанда тек бір элемент бірлікке тең. Бұл элемент кластердің қай сыныпқа жататынын көрсетеді. Бірінші қабаттың Шығыс векторындағы сызықтық қабаттың салмақ матрицасының көбейтіндісі *n* элементтерінің векторын құрайды, оның жалғыз бірлік элементінің нөмірі анықталған сыныптың санына тең.

4 Қарсы таралу желілері

Қарсы тарату желісі (Counterpropagation Network) Р.Хехт-Нильсен (R. Heht-Nielson) ұсынған және екі қабатты желі, оның бірінші қабаты Кохонен қабаты, ал екіншісі С. гроссберг қабаты (S. Grossberg). Қарама-қарсы тарату желілері дәлдігі жағынан көп қабатты перцептроннан төмен, бірақ тез оқытылады және бірқатар пайдалы қасиеттерге ие. Қарсы тарату желілері нормаланған векторларды қолданады.

Гроссберг қабатының нейрондары гроссбергтің шығу жұлдыздары (Outstar) деп аталады. Гроссбергтің шығу жұлдызы (сурет. 8) скалярлық кіріс және векторлық Шығыс бар.



6 - сурет - Гроссбергтің шығу жұлдызы

Гроссбергтің шығу жұлдызы түрлендіруді жүзеге асырады

$$y_i = \varphi(w_i x), \quad i = 1, 2, \ldots, m \qquad (6.7)$$

белсендіру функциясымен (барлық векторлар нормаланған)

$$\varphi(u_i) = \begin{cases} u_i, & \text{если } -1 \le u_i \le 1, \\ 1, & \text{если } 1 < u_i, \\ -1, & \text{если } u_i < -1. \end{cases} \qquad (6.8)$$

Қарсы тарату желісін оқыту екі қадамнан тұрады. Бірінші қадамда бұрын қарастырылған алгоритмдер бойынша кохонен қабаты оқытылады. Оқытылған кохонен қабаты тек бір компоненті нөлден ерекшеленетін векторды шығарады. Екінші қадамда гроссберг қабатының мұғалімімен оқыту жүргізіледі. Гроссберг қабатының салмағын түзету формула бойынша жүзеге асырылады

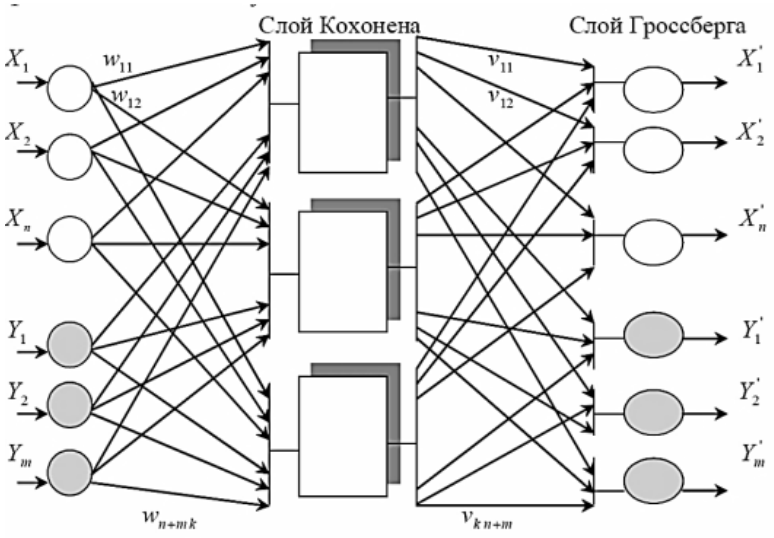
$$w_{ij}^{(k+1)} = w_{ij}^{(k)} + \beta\left(y_j - w_{ij}^{(k)}\right) y_{ki}, \qquad (6.9)$$

мұндағы k-оқу циклінің нөмірі, w_{ij} - j нейронының i-ші салмағы, y_{ki} - Кохонен қабатының i-ші нейронының шығысы (тек бір Кохонен нейроны үшін ол нөлден ерекшеленеді), y_j - қажетті Шығыс векторының j-ші компоненті, β - оқу жылдамдығының коэффициенті.

Кохонен қабатының шығысындағы векторда жеңімпаз нейронға сәйкес келетін бір ғана компонент болғандықтан, іс жүзінде Гроссберг қабатының нейрондарын Кохонен қабатының жеңімпаз нейронымен байланыстыратын Шығыс қабатының таразылары ғана нақтыланады.

Егер оқытылған желіге оқыту үлгісіне жатпайтын вектор берілсе, онда алдымен Кохонен қабатында оның белгілі бір кластерге жататындығы орнатылады, содан кейін Кохонен қабатының жеңімпаз нейронының сигналы Шығыс жұлдызына түседі, оның шығысында элементтері анықталған кластердің центроидының (центрінің) координаттарына тең Шығыс векторы пайда болады.

Қарастырылған желі бір бағытты тікелей тарату желісі болып табылады. Екі бағытты қарсы тарату желілері кіріс векторында Шығыс векторын табуға ғана емес, сонымен қатар белгілі Шығыс векторында сәйкес кіріс векторын табуға мүмкіндік береді. Бұл қасиеттер "қарсы таралу" терминіне байланысты. Екі бағытты қарсы тарату желісінің құрылымы 9 суретте көрсетілген.



7 - сурет - Екі бағытты қарсы тарату желісі

Оқыту процесінде *x* және *y* векторлары желінің кіріс векторлары ретінде де, мақсатты Шығыс векторлары ретінде де қолданылады. *x* векторлары *x`* шығысын үйрету үшін, ал *y* векторлары *y`* шығысын үйрету үшін қолданыладыөнім қабаты. Желіні оқыту бір бағытты қарсы тарату желісін оқыту сияқты жүзеге асырылады. Ресми түрде векторлардың өлшемі ғана өсті. Егер оқытылған желінің кірісіне оқыту кезінде пайдаланылмаған *x* векторы және сәйкес *y* Шығыс векторы берілсе, онда осы векторлардың *x`* жиыны мен *y`* жиынының жуықтамалары шығады. Бірақ мұндай тапсырма практикалық қызығушылық тудырмайды: кейбір картаның кіріс және шығыс векторлары белгілі. Егер кіріске тек x векторы берілсе, онда шығыста *x`* және *y`* жуықтау алынады. Яғни, *x*-тің y-ге картасын табуға болады. Егер біз векторды білсек онда оны тек кіріске беру арқылы біз *x`* және *y ′* аламыз. *x′* алу кері картаны іске асыруды білдіреді *x*.

Кластерлерді көрсету:

Som зерттелгеннен кейін салмақ векторы берілген үлгіге ерекше жақын нейронның әрбір кіріс үлгісін тағайындау реті бойынша кластерлеу процесін орындауға рұқсат етіледі. Әрі қарай, әрбір нейрон кластерлердің бірімен байланысты болады.

Кохонен картасындағы кластерлерді визуализациялау үшін әртүрлі әдістерді қолдануға болады:

1 Түсті картограмма: картадағы әрбір нейрон ол байланыстырылған кластерге сәйкес келетін сәйкес түске боялады. Бұл кластерлердің шекараларын визуалды түрде белгілеуге және олардың картада таралуын бағалауға мүмкіндік береді.
2 Маркерлер немесе белгілер: картадағы әрбір нейрон белгілі бір кластерге жататындығын көрсететін маркермен немесе белгімен белгіленуі мүмкін. Бұл қандай нейрондардың қай кластерге жататынын анық көруге мүмкіндік береді.
3 Кластерлік контурлар: Кохонен картасында кластер шекараларын білдіретін контурларды қайта жасауға болады. Бұл белгілі бір кластерлермен байланысты белгілер кеңістігінің аймақтарын көрсетеді.
4 Қосымша визуалды элементтер: кластерлік визуализацияны жақсарту үшін картаға популяция тығыздығы, градиенттер немесе изоляциялар сияқты қосымша элементтерді қосуға болады.

Визуализация әдісін таңдау ақпараттың сипаттамаларына және талдау мақсаттарына байланысты. Кез-келген әдіс өзінің артықшылықтары мен кемшіліктеріне ие және көбінесе мәліметтер құрылымы туралы толық түсінік алу үшін әртүрлі әдістердің тіркесімі қолданылады.

Кохонен желісі деректердегі кластерлерді тани алады, сонымен қатар сыныптардың жақындығын орнатады. Осылайша, пайдаланушы нейроактивті модельді одан әрі нақтылау үшін деректер құрылымының көрінісін арттыра алады. Егер деректерде сыныптар танылса, оларды белгілеуге болады, содан кейін желі жіктеу мәселелерін шеше алады. Кохонен желілерін сыныптар бұрын берілген жіктеу тапсырмаларында да қолдануға болады - бұдан әрі артықшылық желі әртүрлі сыныптар арасындағы ұқсастықтарды анықтай алады.

Визуализация:

Кохоненнің өзін-өзі ұйымдастыратын карталарын (SOM) қолдана отырып, деректерді визуализациялау көп өлшемді белгілер кеңістігіндегі ақпарат құрылымын зерттеуге және түсінуге арналған қуатты құрал болып табылады. Төменде SOM көмегімен деректерді визуализациялаудың негізгі әдістері келтірілген: Кохоненнің өзін-өзі ұйымдастыратын карталарын (SOM) қолдана отырып, деректерді визуализациялау көп өлшемді белгілер кеңістігіндегі ақпарат құрылымын зерттеуге және түсінуге арналған қуатты құрал болып табылады. Төменде SOM көмегімен деректерді визуализациялаудың негізгі әдістері келтірілген:

1 2D немесе 3D проекциялары: SOM екі өлшемді немесе үш өлшемді жазықтық үшін жоғары өлшемді деректерді көрсетуге мүмкіндік береді, бұл деректерді

визуалды зерттеуді жеңілдетеді. Кохонен картасының нейрондары нүктелер немесе басқа маркерлер түрінде ұсынылуы мүмкін және әрбір нейрон белгілі бір кластерге немесе белгінің мәніне сәйкес боялуы мүмкін.

2 Түсті кодтау: картадағы әрбір Нейронды белгілі бір кластерге жататындығы, оның салмақ мәндері немесе басқа өлшемдер сияқты сипаттамаларына байланысты белгілі бір түске бояуға болады. Бұл деректер топтарын көрнекі түрде бөлуге және олардың құрылымын анықтауға мүмкіндік береді.

3 Градиенттер және жылу карталары: градиенттерді немесе картадағы деректердің тығыздығын анықтау үшін градиенттерді немесе жылу карталарын пайдалануға болады. Бұл әдістер картаның әртүрлі аймақтарындағы деректердің тығыздығын визуализациялауға мүмкіндік береді, бұл тығыз және сирек аймақтарды анықтауға көмектеседі.

4 Анимация: динамикалық процестерді немесе деректердегі өзгерістерді визуализациялау үшін анимацияны пайдалануға болады. Мысалы, анимация деректердің уақыт өте келе қалай өзгеретінін немесе Кохонен картасында кластерлердің қалай қалыптасып, дамитынын көрсете алады.

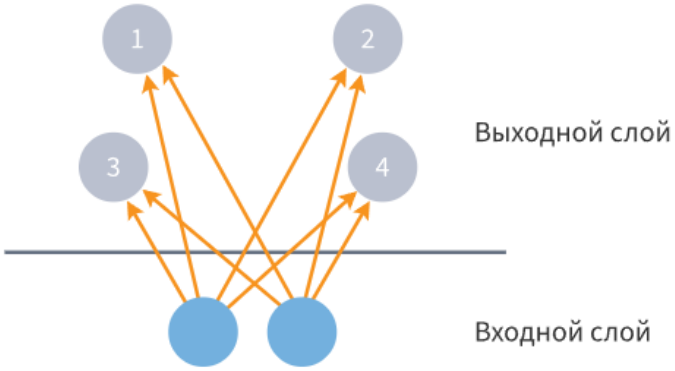
5 Интерактивті визуализация: деректерді тереңірек зерттеу үшін пайдаланушыға Кохонен картасымен өзара әрекеттесуге, оның параметрлерін өзгертуге және деректердің әртүрлі аспектілерін зерттеуге мүмкіндік беретін интерактивті визуализациялар жасауға болады.

Бұл әдістердің әрқайсысының өзіндік артықшылықтары бар және деректердің сипаттамалары мен мақсаттарына байланысты тиімді болуы мүмкін. Тұтастай алғанда, кохоненнің өзін-өзі ұйымдастыратын карталары арқылы деректерді визуализациялау деректер құрылымын көруге, жасырын заңдылықтарды анықтауға және талдау нәтижелерін визуалды түрде түсіндіруге мүмкіндік береді.

Кохонен желісі-кластерлеу мәселесін шешуге арналған нейрондық желінің арнайы түрі. Ол тек екі қабаттан тұрады — кіріс (тарату) және шығыс, оны Кохонен қабаты деп те атайды.

Кохонен желісінде кіріс қабатының әрбір нейроны барлық шығыс нейрондарымен байланысады, ал қабаттардың ішінде байланыс жоқ. Кіріс қабатының нейрондарына кластерленген объектілердің ерекшелік векторлары қолданылады.

Кәдімгі нейрондық желі сияқты, кіріс нейрондары оқу және деректерді өңдеу процесіне қатыспайды, тек кіріс сигналын келесі қабаттың нейрондарына таратады. Кіріс нейрондарының саны белгілер векторының өлшеміне тең (яғни объект белгілерінің саны).



8 - сурет

Кохонен желісінің Шығыс нейрондарының саны модель құруы керек кластерлер санына тең және әрбір нейрон белгілі бір кластермен байланысты. Шығулар "жеңімпаз бәрін алады" қағидаты бойынша өңделеді, яғни ең үлкен шығу мәні бар нейрон бірлікті шығарады, ал қалғандарының шығысы нөлге тең болады.

Осылайша, объектінің ұсынылған желісін өңдеу нәтижесінде Шығыс нейрондарының бірі қозу күйіне өтеді және оның шығуында 1, ал қалғандарының шығуында 0 пайда болады. Осыдан кейін объект қозған нейронмен байланысты кластерге жатады.

Кохонен желісін оқыту, әдеттегі нейрондық желі сияқты, нейрондар арасындағы байланыс салмағын реттеуден тұрады, бірақ бәсекеге қабілетті оқыту технологиясын қолдану арқылы жасалады.

Кохонен желілерінің тұжырымдамасын Тево Кохонен ұсынған.

Loginom-да арнайы өңдеуші бар өзін-өзі ұйымдастыратын желілер, бұл кейбір жалпы қасиеттері бар және кохонен желісі болып табылатын кіріс объектілерінің кластерлерін (топтарын) анықтауға мүмкіндік береді.

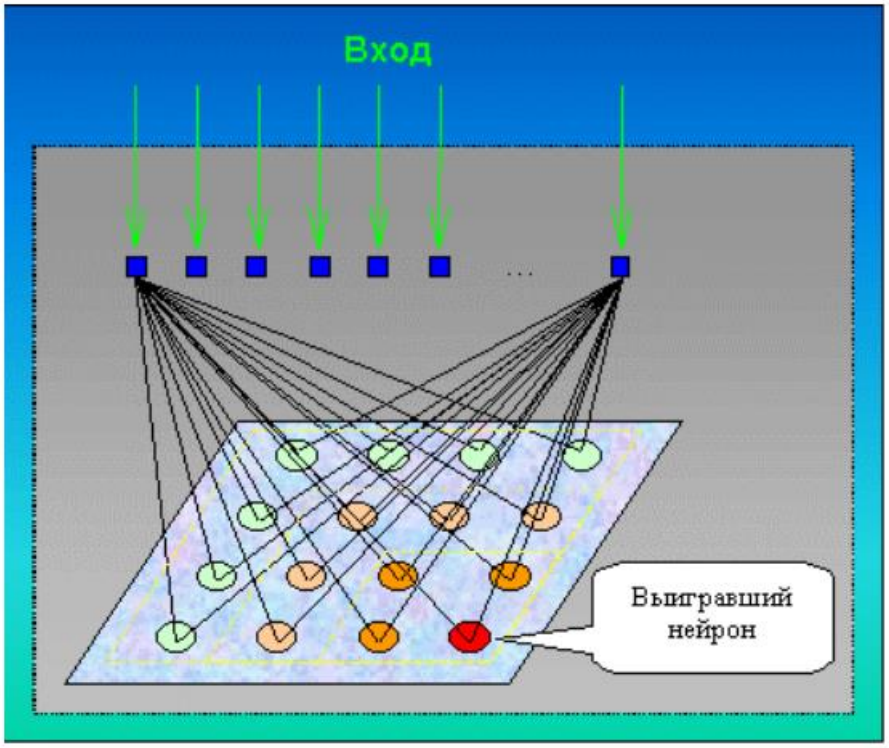
Оқу барысында өзін-өзі ұйымдастыратын карталар m-өлшемді кеңістіктегі кіріс қабатының нүктелерінің орналасу сипатын талдайды және нейрондық желінің шығуында топологиялық тәртіпті және бастапқы деректердің белгілі бір тұрақтылық дәрежесін (яғни векторлардың метрикалық жақындығын) қайталауға тырысады. SOM сәйкестігі wj салмақ коэффициентінің векторын итеративті баптаудан тұрады әрбір Нейрон, j=1,2..., p., жеңімпаз нейронның үлесін ғана емес, сонымен қатар R айналасында орналасқан жақын көршілерін де ескеретін Хэббтің өзгертілген бәсекеге қабілетті оқыту алгоритмі не үшін қолданылады:

1 Инициализация кезеңінде барлық салмақ коэффициенттеріне wij,i=1,2,...,m шамалы кездейсоқ мәндер беріледі;
2 Желі шығыстары кездейсоқ ретпен у кескіндерімен дәйекті түрде беріледі кіріс қабатының объектілері және олардың әрқайсысы үшін ең аз ∑mi=1 (yi-

wtij) қашықтығы бар "жеңімпаз нейрон" (BMU, Best Matching Unit)таңдалады - 1.1 суретті қараңыз;

3 BMU-дің "жақын ортасының" ішкі жиыны анықталады, оның радиусы R әр t итерациясымен азаяды;

4 Жеңімпаз нейронға дейінгі қашықтықты және y векторына жақындығын ескере отырып, таңдалған түйіндердің wij салмақтары қайта есептеледі.



9 - сурет - Кохонен желісіндегі нейрондарды белсендіру схемасы

Алгоритмнің 2-4 қадамдары желінің Шығыс мәндері берілген дәлдікпен тұрақталғанша қайталанады. Бұл ретте көп өлшемді деректерді жазықтыққа проекциялау сапасына som-да бірнеше деңгейде қол жеткізіледі: топологияны сақтау (яғни, бастапқы деректер нүктелері мен оқытылған желі нейрондарының жиындарында көршілік құрылымы бірдей), ретті сақтау (яғни, нүктелердің эквивалентті жұптары арасындағы қашықтық пропорционалды) және кеңістікті қысу кезінде метрикалық қасиеттерді сақтау.

Оқыту нәтижесінде "проекциялық экран" нейрондық синапстардың шамалары екі өлшем бойынша біркелкі өзгеретін реттелген құрылымның қасиеттеріне ие болады. Екі өлшемді тор фрагменттерінің түсі мен орналасуы мәліметтер жиынтығының компоненттерімен байланысты заңдылықтарды талдау үшін қолданылады. Атап айтқанда, әрбір түйінмен (нейронмен) кластерлердің әлеуетті орталықтары ретінде қызмет ете алатын бастапқы объектілердің жергілікті қалыңдауы байланыстырылуы мүмкін.

Желіні оқыту үшін әдетте kohonen пакетіндегі somgrid() және com () функциялары қолданылады. Итерациялық процесс аяқталғаннан кейін plot () функциясы келесі карталар жиынтығын визуализациялауға қол жетімді болады:

- "codes" - тордың таралуы жеке бастапқы айнымалылардың қатысу үлесінің арақатынасы көрсетіледі;
- "counts" - желінің әрбір түйініндегі бастапқы нысандар саны;
- "mapping" - қалыптасқан картадағы бастапқы объектілердің координаттары;
- ""property", "quality", "best.neighbours" - әр түрлі түстермен әр түйіннің қасиеттерінің тұтас жиынтығы бейнеленген: жеке бастапқы айнымалылардың қатысу үлесі, нейрондар арасындағы жұптық немесе орташа қашықтық өлшемдері және т. б.

Жіктеу тапсырмасындағы кохонен гибридті желілерінің тиімділігін зерттеу мұндай шешімнің белгілі бір тапсырма үшін қаншалықты тиімді екенін анықтау үшін бірнеше кезеңдер мен аспектілерді қамтуы мүмкін. Міне, осындай зерттеуді жүргізудің жалпы жоспары:

1 Жіктеу мәселесін анықтау:
   - Шешкіңіз келетін жіктеу мәселесін нақты анықтаңыз. Сіз қолданатын деректер түрін және жіктегіңіз келетін мақсатты сыныптарды көрсетіңіз.
2 Гибридті желіні дамыту:
   - Кохонен гибридті желісіне қандай компоненттер кіретінін шешіңіз. Бұған нейрондық желілердің басқа түрлері, мысалы, конволюциялық нейрондық желілер (CNN) немесе қайталанатын нейрондық желілер (RNN), сондай-ақ деректерді түрлендіруге арналған қосымша қабаттар кіруі мүмкін.
   - Гибридті желі архитектурасын, соның ішінде қабаттар санын, әр қабаттағы нейрондар санын және белсендіру функцияларының сипаттамаларын анықтаңыз.
3 Деректерді дайындау:
   - Деректерді, соның ішінде масштабтауды, қалыпқа келтіруді және оқу және сынақ жиынтықтарына бөлуді дайындаңыз.
4 Гибридті желіні оқыту:
   - Кохоненнің гибридті желісін оқыту деректері бойынша оқытыңыз. Оқу жылдамдығы мен дәуірлердің саны сияқты оқу параметрлерін анықтаңыз.
5 Тиімділікті бағалау:
   - Дәлдік, толықтық, F1 өлшемі және қате матрицасы сияқты жіктеу көрсеткіштерін қолдана отырып, сынақ деректерінде гибридті желінің өнімділігін бағалаңыз.
   - Гибридті желінің өнімділігін басқа жіктеу әдістерімен салыстырыңыз, соның ішінде стандартты нейрондық желілер, тірек векторлық әдіс (SVM), шешуші ағаштар және т. б.
6 Оптимизация:
   - Модельдің оңтайлы параметрлері мен оқу алгоритмдерін анықтау үшін эксперименттер жасаңыз.

- Кросс-тексеру сияқты оңтайландыру әдістерін қолдана отырып, гиперпараметрлерді орнатуды қарастырыңыз.

7 Нәтижелерді түсіндіру:
- Қандай деректер кластары жақсы жіктелгенін және қайсысы қосымша жақсартуларды қажет етуі мүмкін екенін түсіну үшін нәтижелерді талдаңыз.
- Жіктеу үшін қандай белгілер маңызды екенін анықтау үшін кохонен желісінің нейрондарының салмағын зерттеңіз.

8 Нәтижелерді құжаттау және ұсыну:
- Тапсырманың сипаттамасын, модель архитектурасын, нәтижелерін және қорытындыларын қоса, зерттеу есебін жасаңыз.
- Белгілі бір жіктеу тапсырмасы контекстінде гибридті желінің тиімділігін талқылау үшін нәтижелеріңіз бен нәтижелеріңізді көрсетіңіз.

Кохоненнің гибридті желілерінің тиімділігін зерттеу олардың нақты жіктеу мәселелерін шешуге жарамдылығын анықтау үшін мұқият баптауды және талдауды қажет етеді.

Өзін-өзі ұйымдастыратын карта түйіндер немесе нейрондар деп аталатын компоненттерден тұрады. Олардың санын талдаушы белгілейді. Түйіндердің әрқайсысы екі вектормен сипатталады. Біріншісі деп аталады салмақ векторы m, кіріс деректерімен бірдей өлшемге ие. Екіншісі-картадағы түйіннің координаттары болып табылатын R векторы. Кохонен картасы тікбұрышты немесе алтыбұрышты пішінді ұяшықтар арқылы визуалды түрде көрсетіледі; соңғысы жиі қолданылады, өйткені бұл жағдайда іргелес ұяшықтардың орталықтары арасындағы қашықтық бірдей, бұл картаны визуализациялаудың дұрыстығын арттырады.

Бастапқыда кіріс деректерінің өлшемі белгілі, оған сәйкес картаның бастапқы нұсқасы жасалады. Оқу процесінде түйін салмағының векторлары кіріс деректеріне жақындайды. Әрбір бақылау (үлгі) үшін салмақ векторына ең ұқсас түйін таңдалады және оның салмақ векторының мәні бақылауға жақындайды. Сондай-ақ, бақылауға жақын орналасқан бірнеше түйіндердің салмақ векторлары жақындайды, осылайша егер кіріс массивінде екі бақылау ұқсас болса, картада оларға жақын түйіндер сәйкес келеді. Кіріс деректерін сұрыптайтын циклдік оқыту процесі карта рұқсат етілген (талдаушы алдын ала берген) қатеге жеткенде немесе берілген итерациялар саны орындалғанда аяқталады. Осылайша, оқыту нәтижесінде Кохонен картасы кірістерді кластерлерге жіктейді және екі өлшемді жазықтықтағы көпөлшемді кірістерді визуалды түрде көрсетеді, жақын белгілердің векторларын іргелес ұяшықтарға таратады және оларды талданатын нейрондық параметрлерге сәйкес бояйды.

Алгоритм жұмысының нәтижесінде келесі карталар алынады:
- Нейрондық кіріс картасы-карта нейрондарының салмағын реттеу арқылы ішкі кіріс құрылымын визуализациялайды. Әдетте бірнеше кіріс карталары қолданылады, олардың әрқайсысы біреуін көрсетеді және нейронның салмағына байланысты боялады. Карталардың бірінде

талданатын мысалдар үшін шамамен бірдей кірістер қосылатын аймақ белгілі бір түспен белгіленеді.

- Нейрондық Шығыс картасы-кіріс мысалдарының өзара орналасу моделін елестетеді. Картадағы анықталған аймақтар-шығу мәндері ұқсас нейрондардан тұратын кластерлер.
- арнайы карталар-бұл өзін-өзі ұйымдастыратын Кохонен картасының алгоритмін, сондай-ақ оларды сипаттайтын басқа карталарды қолдану нәтижесінде алынған кластерлер картасы.

Желі жұмысы:

- Картаны инициализациялау, яғни түйіндер үшін салмақ векторларының бастапқы жұмысы.

Цикл:

- Келесі бақылауды таңдау (көптеген кірістердің векторы).
- Ол үшін ең жақсы сәйкестік бірлігін табу (best matching unit, BMU, немесе Winner) — картадағы түйін, оның салмақ векторы бақылаудан ең аз ерекшеленеді (аналитик берген метрикада, көбінесе Евклид).
- BMU көршілерінің санын анықтау және оқыту-бақылауға жақындау мақсатында BMU және оның көршілерінің салмақ векторларын өзгерту.
- Карта қатесін анықтау.

Алгоритм:

- Инициализация
- Түйіндердің бастапқы салмағын орнатудың ең көп таралған үш әдісі:
- Барлық координаттарды кездейсоқ сандармен белгілеу.
- Кіріс деректерінен кездейсоқ бақылау мәнінің салмақ векторына тағайындау.
- Кіріс жиынтығының негізгі компоненттеріне созылған сызықтық кеңістіктен салмақ векторларын таңдау.
- Цикл

t-Итерация нөміріи болсын (инициализация 0 санына сәйкес келеді).

- x (t) көптеген кірістерден ерікті бақылау таңдаңыз.
- Одан картаның барлық түйіндерінің салмақ векторларына дейінгі қашықтықты табыңыз және ең жақын M_c(t) түйінін анықтаңыз. Бұл BMU немесе Winner. M_c(t)шарты: $\|x(t) - m_c(t)\| \leq \|x(t) - m_i(t)\|,$ кез келген $m_i(t)$, $m_i(t)$ - түйін салмағының векторы $M_i(t)$. Егер шартты қанағаттандыратын бірнеше түйін болса, BMU олардың арасында кездейсоқ таңдалады.
- Салмақ векторларының өзгеруі

Салмақ векторын келесі формула бойынша өзгереді:

$$m_i(t) = m_i(t-1) + h_{ci}(t) \cdot (x(t) - m_i(t-1)) \quad (7)$$

- BMU көршілері болып табылатын барлық түйіндердің салмақ векторлары қарастырылып отырған бақылауға жақындайды.
- Карта қатесін есептеу

Мысалы, бақылаулар мен оларға сәйкес келетін BMU салмақ векторлары арасындағы арифметикалық қашықтықтың орташа мәні ретінде:

$$\frac{1}{N}\sum_{i=1}^{N}\|x_i - m_c\|, \qquad (7.1)$$

N - кіріс жиынтығы элементтерінің саны.

Модель ерекшеліктері

Шулы деректерге төзімділік, жылдам және басқарылмайтын оқыту, визуализация арқылы көпөлшемді кірістерді жеңілдету мүмкіндігі.

Кохоненнің өзін-өзі ұйымдастыратын карталарын кластерлік талдау үшін кластерлердің саны алдын-ала белгілі болған жағдайда ғана пайдалануға болады.

Маңызды кемшілігі-нейрондық желілердің соңғы нәтижесі желінің бастапқы қондырғыларына байланысты. Екінші жағынан, нейрондық желілер теориялық тұрғыдан кез-келген үздіксіз функцияны жуықтай алады, бұл зерттеушіге модельге қатысты кез-келген гипотезаны алдын-ала қабылдамауға мүмкіндік береді.

Кохонен желілері арқылы жіктеу Кохонен картасында кіріс векторына ең жақын Нейронды табу реті бойынша орын алады. Міне, осы процестің формулалары:

а) Жеңімпазды табу (ең жақын Нейрон):

Әрбір кіріс векторы үшін x және әрбір Нейрон J Кохонен картасында олардың арасындағы қашықтық анықталады. Бұл жағдайда векторлар арасындағы қашықтықты евклидтік қашықтық арқылы есептеуге болады:

Қайда:

- d_j - кіріс векторы мен Нейрон арасындағы қашықтық j;
- x_i - кіріс векторының I компоненті;
- w_{ji} - J Нейрон салмағының і компоненті.

Жеңімпаз (ең жақын Нейрон) ең қысқа қашықтықтағы нейрон d_j болады.

б) Таразыны жаңарту формуласы:

Жеңімпаз анықталғаннан кейін, осы нейронның және оның көршілерінің салмағы кіріс векторы бағытында жаңартылады. Таразыны жаңарту формуласы келесідей:

$$w_{ji}^{новый} = w_{ji}^{старый} + \eta(t)\bullet(x_i - w_{ji}^{старый}) \qquad (7.2)$$

Қайда:

- $w_{ji}^{новый}$ - Нейрон салмағының Жаңа мәні ол і компонент;
- $w_{ji}^{старый}$ - Нейрон салмағының ескі мәні ол і компонент;
- $\eta(t)$ - оқу жылдамдығы (learning rate) уақытқа тәуелді t;

- x_i - кіріс векторының і компоненті.

Әдетте, тұрақты оқуды қамтамасыз ету үшін оқу жылдамдығы уақыт өте келе төмендейді.

ҚОРЫТЫНДЫ

Қорытындылай келе, кохонен желілерінің жіктелуі мұғалімсіз және деректерді талдаусыз оқыту саласындағы қуатты құрал болып табылады. Бұл желілер деректердегі жасырын құрылымдарды анықтауға және оларды жоғары дәлдікпен кластерлеуге қабілетті. Олардың қолданылуы үлгіні тану, деректерді қысу, мәтіндер мен кескіндерді талдау және биология, медицина және қаржы сияқты әртүрлі салаларда кеңінен қолданылады.

Кохонен желілерін жіктеудің негізгі артықшылықтарының бірі-олардың деректердің өзгеруіне бейімделу және мұғалімсіз оқу қабілеті, бұл оларды деректердің үлкен көлемі және/немесе өзгергіштігі жағдайында әсіресе пайдалы етеді. Сонымен қатар, олардың салыстырмалы қарапайымдылығы мен тиімділігі оларды көптеген зерттеушілер мен практиктерге қол жетімді етеді.

Алайда, Кохонен желілерін пайдалану кезінде кейбір шектеулерді ескеру қажет, соның ішінде желі параметрлерін дұрыс конфигурациялау және кластерлердің оңтайлы санын анықтау қажет. Сонымен қатар, Кохонен желілерін жіктеудің тиімділігі деректердің табиғаты мен ерекшелігіне, сондай-ақ таңдалған кластерлеу сапасын бағалау метрикасына байланысты болуы мүмкін.

Тұтастай алғанда, Кохонен желілерінің жіктелуі Деректерді талдаудың маңызды құралы болып табылады, оны деректердегі құрылымдар мен заңдылықтарды анықтау үшін тиімді қолдануға болады, бұл оларды қазіргі деректерді талдау және машиналық оқыту әлемінде таптырмас етеді.

ПАЙДАЛАНЫЛҒАН ӘДЕБИЕТТЕР

1 Осовский С. Нейронные сети для обработки информации. — М.: Финансы и статистика, 2002.

2 Кохонен Т. Самоорганизующиеся карты. — М.: БИНОМ. Лаборатория знаний, 2008.

3 Анализ данных и процессов / А. А. Барсегян, М. С. Куприянов, И. И. Холод, М. Д. Тесс, С. И. Елизаров. — СПб.: БХВ-Петербург, 2009.

4 Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика. — М.: Мир, 1992.

5 Хайкин С. Нейронные сети: полный курс. — М.: Вильямс, 2006.

6 https://jupyter.org/try-jupyter/lab/

7 Ануфриев, И. Е. MatLab 7. Наиболее полное руководство в подлиннике / И. Е. Ануфриев, А. Б. Смирнов, Е. Н. Смирнова. – СПб. : БХВ-Петербург, 2005.