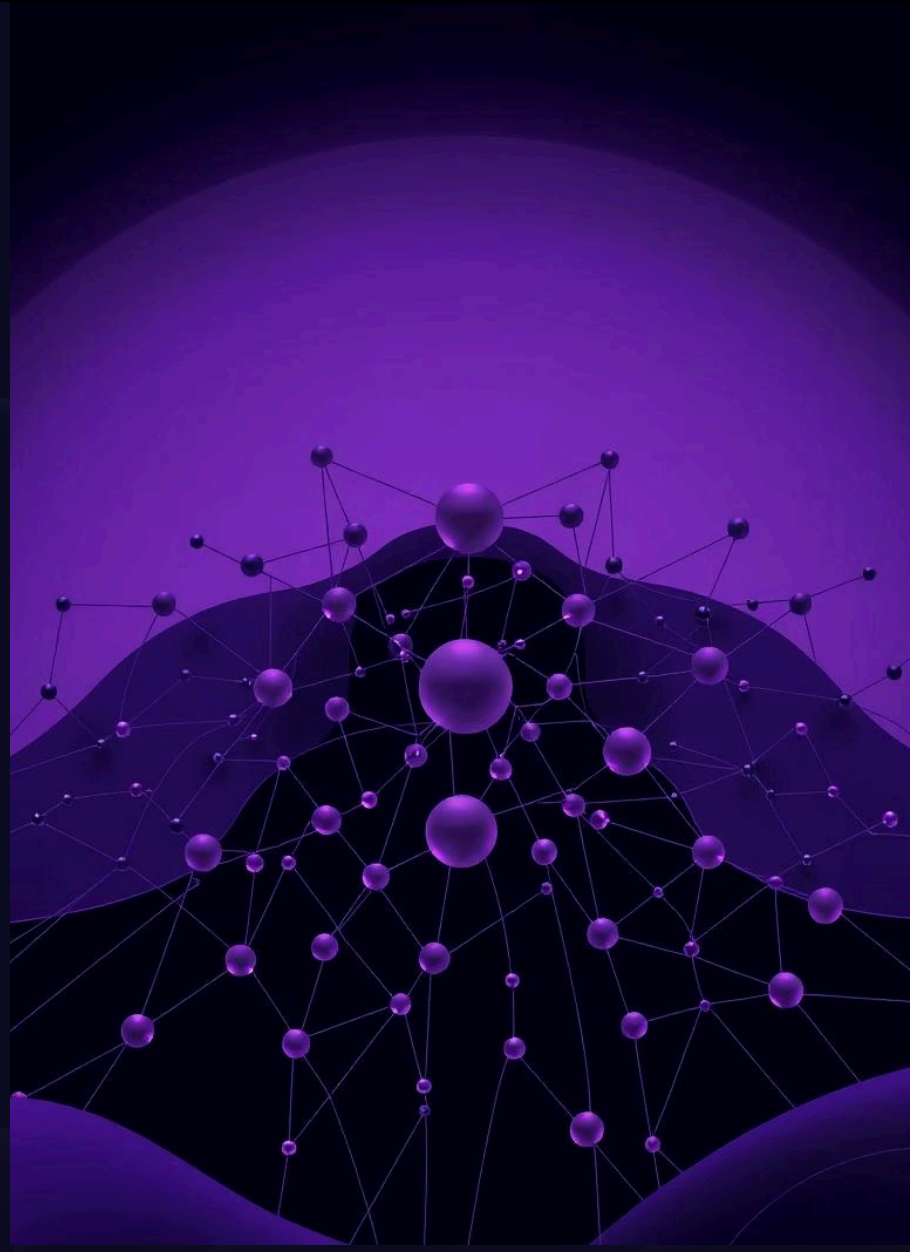
Дәріс 11: Нейрондық желілер



Оқытушы: Сарсембаева Т.С.

Дәрістің мақсаты

Жасанды нейрондық желілердің (ANN) не екенін, олардың адам миының құрылымынан қалай шабыт алғанын және терең оқытудың негізі екенін түсіндіруден басталады. Содан кейін, нейрондық желінің негізгі құрылыс блогы – Фрэнк Розенблаттың бір қабатты перцептроны, оның құрылымы және үйрену ережесі талданады. Дәріс бір қабатты перцептронның шектеулілігін (XOR мәселесі) және оның көп қабатты перцептронның (MLP) пайда болуына қалай әкелгенін көрсетеді.

Оған қоса, көп қабатты перцептронның (MLP) архитектурасы және заманауи активтендіру функциялары зерттеледі. Соңында, MLP-ді үйретудің негізгі механизмі – Қатені кері тарату (Backpropagation) алгоритмі, оның градиенттік түсуге және шығын функциясына қалай сүйенетіні түсіндіріледі.

Негізгі сұрақтар және қысқаша мазмұн

1. Жасанды нейрондық желілер дегеніміз не және олар адам миына қалай еліктейді?
2. Бір қабатты перцептрон дегеніміз не және ол қалай үйренеді?
3. "XOR мәселесі" дегеніміз не және ол нейрондық желілердің дамуына қалай әсер етті?
4. Көп қабатты перцептронның (MLP) архитектурасын сипаттаңыз
5. Қатені кері тарату (Backpropagation) алгоритмі не үшін қажет?
6. Неліктен MLP-де Розенблаттың "баспалдақ функциясы" емес, ReLU немесе Sigmoid сияқты функциялар қолданылады?

Нейрондық желілер (ЖНЖ) - бұл адам миының құрылымына еліктеуге тырысатын есептеу моделі. Ол кіріс деректердегі күрделі заңдылықтарды тануға қабілетті математикалық функциялар жиынтығы. Олардың басты күші – мысалдар арқылы өздігінен үйрену қабілеті.

Нейрондық желілер дегеніміз не?

Соңғы онжылдықта "жасанды интеллект" терминін жиі еститін болдық. Өздігінен жүретін көліктер, дауыстық көмекшілер (Siri, Alexa), медициналық диагноз қою жүйелері және аударма қызметтері – осының барлығының негізінде терең оқыту (Deep Learning) жатыр. Ал терең оқытудың өзегі – бұл жасанды нейрондық желілер (Artificial Neural Networks, ANN).

Жасанды нейрондық желілер – бұл адам миының құрылымына еліктеуге тырысатын есептеу моделі. Адам миы миллиардтаған өзара байланысқан нейрондардан тұрады. Бұл нейрондар электрлік және химиялық сигналдар арқылы ақпаратты қабылдайды, өңдейді және таратады. Дәл осы биологиялық шабыт жасанды нейрондық желілердің негізін қалады.

Бірақ ЖНЖ – бұл мидың толық көшірмесі емес. Бұл – кіріс деректердегі күрделі заңдылықтарды тануға қабілетті, математикалық функциялар мен алгоритмдердің жиынтығы. Олардың басты күші – үйрену қабілеті. Біз оларға "мынау – мысықтың суреті, ал мынау – иттің суреті" деп мыңдаған мысалдарды көрсетеміз, ал желі осы мысалдар арқылы мысық пен итті ажырататын белгілерді өздігінен тауып алуды үйренеді.

Бүгінгі дәрісте біз осы "қара жәшіктің" ішіне үңіліп, оның ең қарапайым құрылыс материалы – перцептроннан бастап, оның қалай үйренетінін – қатені кері тарату әдісіне дейін қарастырамыз.

Перцептрондар: Нейрондық желінің құрылыс блогы

Нейрондық желілерді зерттеуді оның ең қарапайым түрі – перцептроннан бастаған жөн. 1957 жылы Фрэнк Розенблатт ұсынған перцептрон – бұл бір ғана жасанды нейроннан тұратын модель. Ол белгілі бір кірістерді қабылдап, оларды "таразылап", бір ғана шешім (иә/жоқ, 1/0) шығарады.

Бір қабатты перцептрон **(Single-Layer Perceptron)**

Перцептронның құрылымы өте қарапайым:

1. Кірістер (Inputs, x): Бұл біздің деректеріміз (мысалы, суреттің пиксельдері немесе студенттің емтихан бағалары). Бізде n кіріс болуы мүмкін: $x_1, x_2, ..., x_n$
2. Салмақтар (Weights, ω): Әрбір кірістің (x_i) өзінің салмағы (ω_i) болады. Салмақ – бұл сол кірістің соңғы шешімге қаншалықты маңызды екенін көрсететін сан. Егер салмақ жоғары болса, бұл кіріс маңызды; егер төмен болса, маңызды емес. Үйрену процесі – осы салмақтарды дұрыс реттеу.
3. Қосындылау функциясы (Summation Function): Нейрон барлық кірістерді олардың сәйкес салмақтарына көбейтіп, қосындысын табады. Бұған қоса, ығысу (bias, b) деп аталатын қосымша параметр қосылады. Ығысу нейронның "белсендірілуге" қаншалықты бейім екенін реттейді.

Математикалық түрде: $z = (x_1 \cdot \omega_1) + (x_2 \cdot \omega_2) + ... + (x_i \cdot \omega_i) + b$

Немесе векторлық түрде: $z = \omega \cdot x + b$

1. Активтендіру функциясы (Activation Function, f(z)): Қосынды z есептелгеннен кейін, ол активтендіру функциясынан өтеді. Розенблатт перцептронында бұл қарапайым баспалдақ функциясы (step function) болды:

Егер z белгілі бір шек (threshold, θ) мәнінен үлкен болса, нейрон "оянады" (шығыс = 1).

Егер z шектен аз немесе тең болса, нейрон "ұйықтайды" (шығыс = 0).

$y = f(z) = \{ 1 \text{ егер } z > \theta; 0 \text{ егер } z \le \theta \}$

Үйрену ережесі (Perceptron Learning Rule)

Перцептрон қалай үйренеді? Ол қадағаланатын оқыту (supervised learning) арқылы үйренеді. Яғни, біз оған кіріс (x) беріп қана қоймай, "дұрыс жауапты" (t, target) да береміз.

01

Желіге кіріс беріледі

Ол өз болжамын (y) шығарады.

02

Болжам салыстырылады

Болжам (y) дұрыс жауаппен (t) салыстырылады.

03

Қате есептеледі

Қате (error): $e = t - y$

04

Салмақтар жаңартылады

Егер қате болса (1 немесе -1), желі салмақтарды (ω) және ығысуды (b) жаңартады.

Салмақты жаңарту формуласы:

$\omega_i(\text{жаңа}) = \omega_i(\text{ескі}) + \eta \times (t - y) \times x_i$

$b(\text{жаңа}) = b(\text{ескі}) + \eta \times (t - y)$

Мұндағы η – оқыту жылдамдығы (learning rate), бізің "қадамымыздың" өлшемі.

- Егер $t = 1, y = 0$ болса (қате = 1): Салмақтар x_i-ге пропорционалды түрде артады.
- Егер $t = 0, y = 1$ болса (қате = -1): Салмақтар x_i-ге пропорционалды түрде азаяды.

Перцептронның шектеулері: **XOR** мәселесі

Перцептрон AND (ЖӘНЕ), OR (НЕМЕСЕ) сияқты қарапайым логикалық операцияларды оңай үйрене алады. Себебі бұл деректер сызықтық бөлінетін (linearly separable) болып табылады. Яғни, "1" класын "0" класынан түзу сызықпен бөліп алуға болады.

Бірақ перцептрон XOR (Ерекше НЕМЕСЕ) операциясын үйрене алмады.

x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Егер XOR нәтижелерін графикке салсақ, (0,1) және (1,0) нүктелерін (0,0) және (1,1) нүктелерінен бір ғана түзу сызықпен бөлу мүмкін емес екенін көреміз.

Бұл "XOR мәселесі" 1960 жылдардың соңында жасанды интеллект саласында үлкен дағдарысқа ("AI Winter") әкелді. Ғалымдар бір ғана нейронның тым шектеулі екенін түсінді. Шешім? Нейрондарды қабаттарға біріктіру.

Көп қабатты перцептрон (MLP) және тікелей таратылатын желілер

XOR мәселесін шешу үшін бірнеше перцептронды біріктіру керек болды. Бұл көп қабатты перцептрон (Multi-Layer Perceptron, MLP) моделінің пайда болуына әкелді. Бұл – бүгінгі терең оқытудың негізі.

MLP – бұл тікелей таратылатын нейрондық желінің (Feed-forward Network) классикалық мысалы. "Тікелей таратылатын" дегеніміз – ақпарат тек бір бағытта: кірістен шығысқа қарай, ешқандай цикл немесе кері байланыссыз қозғалады.

MLP Архитектурасы

MLP кем дегенде үш қабаттан тұрады:

Кіріс қабаты (Input Layer)

Деректерді қабылдайтын нейрондар. Бұл қабатта есептеу жүрмейді, ол тек деректі келесі қабатқа жібереді. Нейрондар саны кіріс деректердің өлшем (features) санына тең.

Жасырын қабат(тар) (Hidden Layer(s))

Кіріс және шығыс қабаттарының арасында орналасқан. Нағыз "ойлау" процесі осында жүреді. Бұл қабаттар кіріс деректерден күрделірек және абстрактілі белгілерді шығаруға үйренеді. "Терең оқыту" осы жасырын қабаттардың көп болуын білдіреді.

Шығыс қабаты (Output Layer)

Желінің соңғы нәтижесін немесе болжамын шығарады. Нейрондар саны шешілетін мәселеге байланысты.

Активтендіру функциялары (Activation Function)

MLP-ді үйрету үшін біз Розенблаттың "баспалдақ функциясын" қолдана алмаймыз. Неге? Себебі ол дифференциалданбайды. Ал бізге желіні үйрету үшін градиент (туынды) қажет. Бізге тегіс, үздіксіз функциялар керек.

Сигмоид (Sigmoid)

Формуласы: $\sigma(z) = 1/(1 + e^{-z})$. Нәтижені [0, 1] аралығына "қысады". Ықтималдықтарды болжау үшін ыңғайлы.

Гиперболалық тангенс (Tanh)

Формуласы: $\tanh(z) = (e^{z} - e^{-z})/(e^{z} + e^{-z})$. Нәтижені [-1, 1] аралығына "қысады". Көбінесе сигмоидқа қарағанда жақсырақ жұмыс істейді.

ReLU (Rectified Linear Unit)

Формуласы: $R(z) = \max(0, z)$. Егер z оң болса, z-тің өзін қайтарады; егер теріс болса, 0 қайтарады. Бүгінгі күні ең танымал таңдау.

Қатені кері тарату әдісі (Backpropagation)

Бізде енді көп қабатты желі бар. Бірақ оны қалай үйретеміз? Бір қабатты перцептронда біз қатені (t – y) білдік және салмақтарды тікелей жаңарттық.

MLP-де біз тек соңғы шығыс қабатының қатесін білеміз. Ал жасырын қабаттардағы нейрондардың "дұрыс жауабы" қандай болуы керек еді? Біз мұны білмейміз.

Осы мәселені шешу үшін 1980 жылдары Қатені кері тарату (Backpropagation) алгоритмі жасалды. Бұл – бүкіл заманауи терең оқытудың негізгі қозғалтқышы.

Backpropagation екі негізгі тұжырымдамаға сүйенеді:

Шығын функциясы (Training Loss Function)	Градиенттік түсу (Gradient Descent)

1. Оқыту шығыны (Training Loss)

Желінің қаншалықты "жақсы" жұмыс істеп тұрғанын өлшеу үшін бізге бір сан қажет. Бұл сан шығын функциясы деп аталады. Ол желінің болжамы (y_{pre}) мен нақты жауаптың (y_{true}) арасындағы айырмашылықты есептейді.

Мақсатымыз – осы шығынды (Loss, L) барынша азайту.

Жиі қолданылатын шығын функциясы: Орташа квадраттық қате (Mean Squared Error, MSE):

$$L = (1/N) \sum_{i=1}^{N} (y_{true}^{(i)} - y_{pre}^{(i)})^2$$

2. Градиенттік түсу **(Gradient Descent)**

Енді біздің міндетіміз – желідегі барлық салмақтарды (ω) және ығысуларды (b) осы L шығын функциясын азайтатындай етіп өзгерту. Мұны қалай істейміз? Градиенттік түсу әдісі арқылы.

Аналогия: Сіз тұманды таудың басында тұрсыз және ең төменгі нүктеге түскіңіз келеді. Тұманнан жолды көре алмайсыз, бірақ аяғыңыздың астындағы жердің еңісін сезе аласыз. Ең төменге тез түсу үшін сіз ең тік еңіс бағытында жүресіз.

Градиент (∇L) – бұл шығын функциясының барлық салмақтар бойынша алынған жеке туындыларының векторы. Градиент "шығын функциясы ең жылдам өсетін" бағытты көрсетеді. Шығынды азайту үшін біз градиентке қарсы бағытта кішігірім қадам жасауымыз керек.

Салмақты жаңарту ережесі:

$\omega_{жаңа} = \omega_{ескі} - \eta \cdot \partial L/\partial \omega$

Мұндағы η – оқыту жылдамдығы (learning rate), біздің "қадамымыздың" өлшемі.

Backpropagation алгоритмінің қадамдары

Backpropagation – бұл ∂L/∂ω градиентін есептеудің тиімді әдісі. Ол математикадағы тізбектік ережені (chain rule) қолданады.

Тікелей тарату **(Forward Pass)**

Кіріс деректер (x) желіге беріледі. Ақпарат қабаттан қабатқа өтіп, әр нейронның мәні есептеледі. Ең соңында, шығыс қабатында болжам (y_{pre}) алынады. Осы болжам арқылы шығын (L) есептеледі.

Кері тарату **(Backward Pass)**

Міне, ең қызығы осы жерде. Біз қатені кері бағытта таратамыз. Шығыс қабаты: Алдымен, шығынның (L) шығыс қабатының салмақтарына (ω_{out}) қатысты градиентін есептейміз. Жасырын қабат: Енді шығынның жасырын қабаттың салмақтарына (ω_{hi}) қатысты градиентін табуымыз керек. Тізбектік ережені қолдана отырып, біз шығыс қабатынан келген "қате сигналын" аламыз да, оны жасырын қабаттың салмақтарына "таратамыз".

Салмақтарды жаңарту **(Weight Update)**

Барлық салмақтар үшін ∂L/∂ω градиенттері есептелгеннен кейін, біз градиенттік түсу формуласын қолданып, барлық салмақтарды бір уақытта жаңартамыз: $\omega = \omega - \eta \cdot \nabla L$

Осы үш қадам (Forward > Backward > Update) деректер жинағындағы барлық мысалдар үшін мыңдаған рет қайталанады. Осы қайталану процесі эпоха (epoch) деп аталады. Уақыт өте келе, желінің шығыны (L) азайып, оның болжамдары дәлірек бола бастайды.

Қорытынды және әрі қарай зерттеу

Бүгінгі дәрісте біз жасанды нейрондық желілердің негіздерін қарастырдық.

- Біз оның ең қарапайым құрылыс материалы – перцептроннан бастадық және оның сызықтық емес мәселелерді (XOR) шеше алмайтынын көрдік.
- Бұл шектеуді жеңу үшін көп қабатты перцептрондар (MLP) немесе тікелей таратылатын желілер енгізілгенін білдік.
- MLP-ді үйрету үшін бізге баспалдақ функциясының орнына дифференциалданатын активтендіру функциялары (Sigmoid, ReLU) қажет болды.
- Желіні үйретудің негізгі механизмі – Градиенттік түсу арқылы шығын функциясын азайту екенін түсіндік.
- Қатені кері тарату (Backpropagation) – бұл жасырын қабаттардағы миллиардтаған салмақтардың градиентін тиімді есептеуге мүмкіндік беретін алгоритм екенін көрдік.

Бұл тақырып – терең оқыту әлеміне кіріспе ғана. Әрі қарай зерттеу үшін келесі маңызды тақырыптарға назар аударуға болады:

Конволюциялық нейрондық желілер **(CNN)**

Суреттерді, видеоларды және басқа да кеңістіктік деректерді өңдеуге мамандандырылған желілер. Олар CAPTCHA мысалында MLP-ге қарағанда әлдеқайда тиімді.

Рекуррентті нейрондық желілер **(RNN)**

Мәтін, сөйлеу тілі, уақыт қатарлары сияқты тізбекті деректермен жұмыс істеуге арналған желілер. LSTM және GRU архитектуралары осы саласында маңызды.

Оптимизаторлар **(Optimizers)**

Adam, RMSprop сияқты стандартты градиенттік түсуді жақсартатын алгоритмдер. Олар оқыту процесін тез және тұрақты ете түседі.

Регуляризация **(Regularization)**

Желінің оқу деректерін жаттап алуының (Overfitting) алдын алу әдістері. Dropout және L2 регуляризациясы осы саласында маңызды.

Дәрісті меңгеруге арналған бақылау сұрақтары

1. Жасанды нейрондық желі дегеніміз не және оның перцептроннан қандай айырмашылығы бар?
2. Бір қабатты перцептронның салмақтарын жаңарту ережесін (Perceptron Learning Rule) сипаттаңыз.
3. "XOR мәселесі" дегеніміз не және неліктен бұл мәселе бір қабатты перцептрон үшін маңызды шектеу болды?
4. Көп қабатты перцептрон (MLP) архитектурасында "жасырын қабаттың" рөлі қандай?
5. Неліктен нейрондық желілерде ReLU немесе Sigmoid сияқты дифференциалданатын активтендіру функциялары қажет?
6. Қатені кері тарату (Backpropagation) алгоритмінің екі негізгі кезеңін (Forward Pass және Backward Pass) сипаттаңыз.

Әдебиеттер тізімі

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. https://www.deeplearningbook.org
2. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning. Springer. https://hastie.su.domains/ElemStatLearn/
3. Nielsen, M. (2019). Neural Networks and Deep Learning. Determination Press. http://neuralnetworksanddeeplearning.com
4. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer. https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-PRML-book-2006.pdf
5. Géron, A. (2022). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (3rd ed.). O'Reilly Media. https://github.com/ageron/handson-ml3
6. Chollet, F. (2021). Deep Learning with Python (2nd ed.). Manning Publications. https://www.manning.com/books/deep-learning-with-python-second-edition
7. Sarker, I. H. (2021). Data Science and Analytics: An Overview from Data-Driven Smart Computing and Decision Making. SN Computer Science, 2(332). https://doi.org/10.1007/s42979-021-00765-8
8. Pedregosa, F. et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830. https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html