Дәріс 6. Градиенттік төмендеу және оңтайландыру

Градиенттік төмендеу

Идея, есептеу және қадамды таңдау

Стохастикалық градиенттік төмендеу және мини-пакеттік градиенттік төмендеу

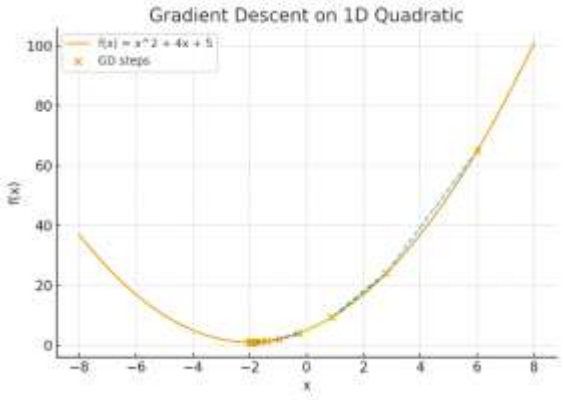
Деректер ғалымының негізгі жұмысы — қол жетімді деректердегі үлгілерді сипаттайтын және жаңа байқаулар үшін ақылға қонымды болжамдар бере алатын идеалды модельді табу.

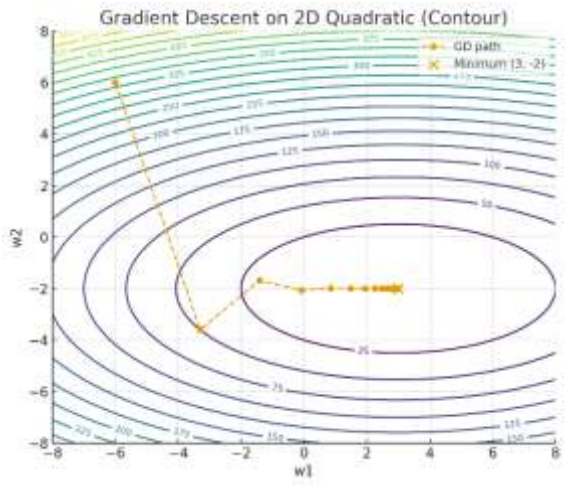
Көптеген машиналық оқыту алгоритмдері оңтайландыруға негізделгендіктен, функцияларды азайту және көбейту жиі кездеседі. Мұны жасау үшін ең негізгі және жалпы құралдардың бірі — градиентті төмендеу. Бұл тәсіл бүгінгі таңда қолданатын көптеген модельдердің негізі болып табылады (сызықтық регрессиядан күрделі терең нейрондық желілерге дейін), және оны түсіну бізге деректер ғылымы әлемінде модельдерді саналы түрде құруға, оқытуға және жетілдіруге мүмкіндік береді.

Градиенттік төмендеу

Градиентті төмендету - қатені(error) немесе шығын(loss) функциясының мәнін азайту үшін қолданылатын итерациялық(қадамдық) оңтайландыру алгоритмі(оптимизация әдісі) болып табылады.

Математикалық деңгейде бұл көп өлшемді функцияның оңтайландыру тапсырмасы болып табылады. Ол машиналық оқыту және терең оқыту модельдерінің өзегін құрайды.





Градиенттердің түсу тұжырымдамасы көп айнымалы функцияны әкелетін бағытта азайту мақсатында оның теріс градиент бағытына қарай жүру болып табылады. Градиент - бұл функцияның максималды өсу бағыты, сол себепті оның қарама-қарсы бағыты максималды азаю бағыты болып табылады.

Жаңартылған параметрлер төмендегідей беріледі:

$$\theta_{\{new\}} = \theta_{\{old\}} - \eta \cdot \nabla L(\theta)$$

$\theta_{\{new\}}$ — модель параметрі (мысалы, салмақтар w және қиян b)

η — оқыту жылдамдығы, ол қадам ұзындығын білдіреді.

$\nabla L(\theta)$ — жоғалту функциясындағы градиент.

Градиенттік түсу есеп мысалы:

Модель теңдеуі:

$$\widehat{\{y\}} = w \cdot x + b$$

Мақсаты: Шынайы мәндер y мен болжаулы мән $\widehat{\{y\}}$-тың арасындағы алшақтықты азайту.

Шығын(қателік) функциясы:

$$L = \frac{1}{n} \sum_{\{i=1\}}^{\{n\}} (y\_i - (w\, x\_i + b))\text{^}2$$

Туынды алу және параметрлерді жаңарту ережесі:

$$w = w - \eta \frac{\{\partial L\}}{\{\partial w\}},\ b = b - \eta \frac{\{\partial L\}}{\{\partial b\}}$$

w - салмақ (weight)

b - ығысу (bias)

η - оқу қадамы (learning rate)

L – шығын функциясы (loss function)

$\frac{\{\partial L\}}{\{\partial w\}}, \frac{\{\partial L\}}{\{\partial b\}}$ – градиенттер

Осылайша, әр итерация сайын модель салмақтары аздап өзгереді және қателік азая береді.

Градиенттік түсудегі алгоритмдегі параметрлерді жаңарту үшін бізге оқу жылдамдығы керек. Оқу жылдамдығы (Learning rate) - дегеніміз градиенттік төмендеу алгоритмі арқылы параметрлерді жаңартуда қолданылатын қадам мөлшері.

$$\theta t + 1 = \theta t - \eta \nabla L(\theta t).$$

Яғни, әр қадам сайын параметр мәні кішігірім вектор $\eta \nabla L(\theta t)$ арқылы жаңартылады.

Енді осы оқу жылдамдығының маңызына келетін болсақ:

- Тұрақтылық және жинақылық жылдамдығы тікелей η-ға байланысты.
- Шығыс (loss) беті әдетте үйлесімсіз (poorly conditioned): кейбір жерлерде өте қисық, ал кейбір жерлерде өте тік(өткір) болады.
- Тым үлкен η болса - әр қадам үлкен параметрлік секірістер жасайды; алгоритм минималды «асырып» жібереді; итерация барысында loss өсіп немесе тұрақсыз шуы пайда болады.
- Тым кіші η болса - әр градиенттік төмендеу қадамы өте кішкентай, өте баяу алға жылжиды; практикалық тұрғыдан есептеу ресурстары мен уақыттық мүмкіндігінен іс жүзінде асып кетуі мүмкін.

Идея, есептеу және қадамды таңдау

Идея (Негізгі мағына)

Градиенттік түсу идеясы – функцияның ең кіші мәнін оның "еңкісін" төменге қарай іздеу арқылы табу.

Мысалы, келесі функцияны қарастырайық: $f(v) = v_1^2 + v_2^2 + v_3^2$ яғни, үшеуліктің квадраттарының қосындысы. Біздің мақсатымыз - осы функцияның ең кіші мәнін табу (яғни, (f(v)) функциясының ең төмен мәнін алу).

Ашып қарастыратын болсақ, ((f(v)) ең төмен мәніне ((v = [0, 0, 0])) кезінде жетеді, себебі әрбір санның квадраты теріс болмайды, ал нөл - ең кіші сан.

Бірақ бұл нәтижені аналитикалық түрде есептей алмаймыз. Сондықтан, функцияның градиентін пайдаланып осы минималды сандық түрде табу таңқаларлық емес.

Градиентті есептеу (Есептеу идеясы)

Градиент - функцияның ең үлкен жылдамдық бағытында көрсететін вектор және оның шамасы осы жылдамдықты сипаттайды. Егер біз функцияның мәнін азайтқымыз келсе, онда біз кері бағытта жүруіміз керек.

Біздің функция үшін: $f(v) = v_1^2 + v_2^2 + v_3^2$

Туындылар (градиент) келесі түрде есептеледі: $f(v) = [2v_1, 2v_2, 2v_3]$ Яғни, әрбір координаттың өзгеру жылдамдығы осы айнымалының екі еселік мәніне тең.

Python тілінде бұл:

```
def sum_of_squares_gradient(v):
    return [2 * v_i for v_i in v]
```

Қадамды таңдау (Step size немесе Learning Rate)

Әр қадамда параметрлерді(вектордың мәндерін) жаңарту үшін, градиент бағытына қарсы шағын қозғалыс жасаймыз:

$$v_{\{new\}} = v_{\{old\}} - \eta \nabla f(v)$$

- $v_{\{old\}}$ – ағымдағы нүкте,
- $\nabla f(v)$ – осы нүктедегі градиент,
- η – оқу қадамы(learning rate немесе step size)

Python тілінде:

```
def gradient_step(v, gradient, step_size):
    step = [step_size * g for g in gradient]
    return [v_i + step_i for v_i, step_i in zip(v, step)]
```

Градиентті төмендеу тақырыбына келетін болсақ, бұл әдістің бірнеше нұсқалары бар және олар деректердің көлемі, есептеу қуаттылығы және модельдің күрделілігіне байланысты қолданылады.

Градиентті төмендеудің үш жалпы түрі бар: Batch Gradient Descent, Stochastic Gradient Descent және Mini-Batch Gradient Descent.

Batch Gradient Descent

Бұл тәсілде модель салмақтары барлық оқу үлгілерін есепке алғаннан кейін ғана жаңартылады. Басқаша айтқанда, барлық үлгілер бір айналым үшін бір рет ғана қолданылады.

Артықшылықтары:

- Тек тура және тұрақты жаңартулар жасайды.
- Шығын функциясының мәні біртіндеп және ретімен төмендейді.

- Жаһандық минимумға жету ықтималдығы жоғары.

Кемшіліктері:

- Үлкен деректер жиынтықтары үшін өте баяу.
- Жад (RAM) ресурстарын көп қажет етеді.

Қолданылуы:

- Сызықтық регрессияда, логистикалық регрессияда және кіші деректер сценарийлерінде жиі қолданылады.

Stochastic Gradient Descent

SGD әрбір үлгі үшін параметрлерді жаңартады. Ол бір үлгіні өтіп, қатені тексеріп, параметрлерін бірден жаңартады.

Артықшылықтары:

- Жылдам есептеулер.
- Үлкен деректер жиынтықтары үшін сәйкес.
- Шу бойынша жергілікті минимумнан шығуға жақсы мүмкіндік береді.

Кемшіліктері:

- Шығын функциясындағы өзгерістер өте агрессивті (жол бірқалыпты емес).
- Тұрақсыз және кейде баяу біріктіріледі.

Қолданылуы:

- Онлайн оқытуда және терең нейрондық желілерді оқыту кезінде жиі кездеседі.

Mini-Batch Gradient Descent

Бұл техника жоғарыда айтылған екі тәсілдің араласуы болып табылады. Деректер жиынтығы бірнеше мини-пакеттерге бөлініп, жаңартулар солар үшін жүргізіледі.

Артықшылықтары:

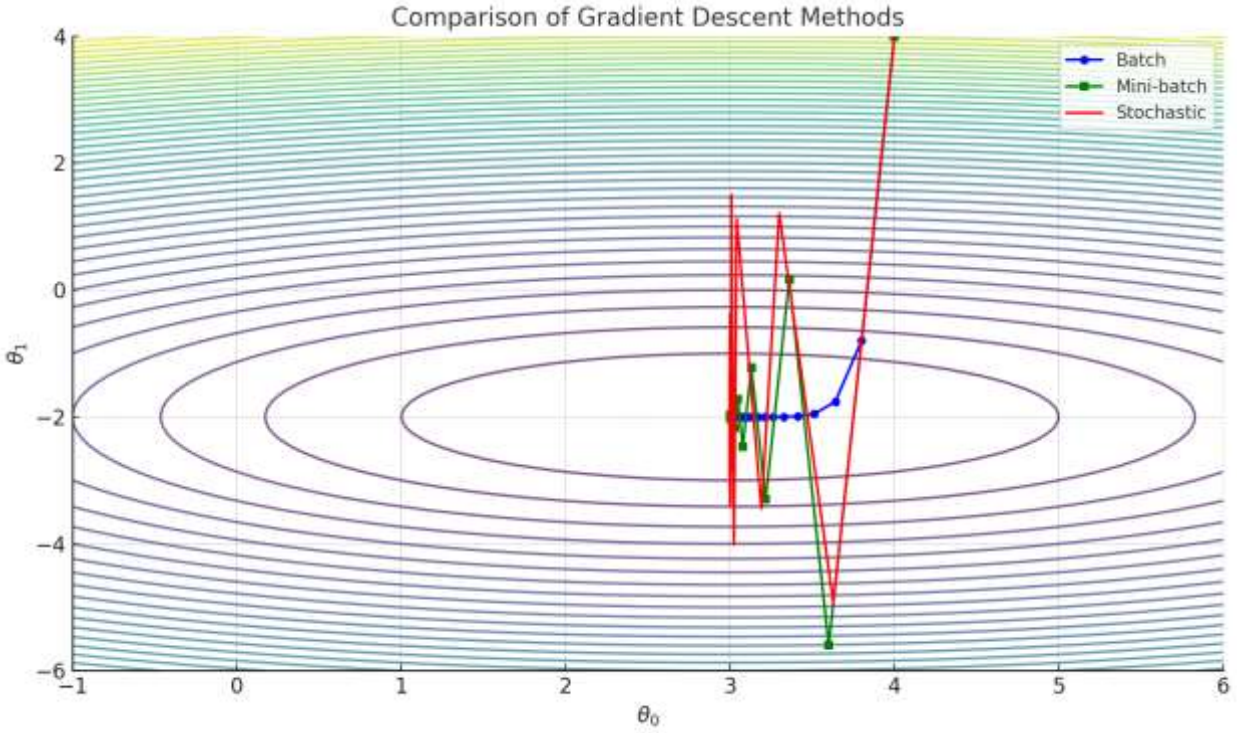
- Есептеу жылдамдығы мен дәлдігі арасындағы тепе-теңдікті қамтамасыз етеді.
- GPU-да параллель есептеулер үшін сәйкес.
- SGD-дегі "шуды" азайтады.
- Үлкен деректер жиынтықтарын өңдеуде тиімді.

Кемшіліктері:

- Пакет өлшемін дәл реттеу қажет (кіші - тұрақсыз, үлкен - баяу).

Қолданылуы:

- Терең оқытуда, компьютерлік көріністе (CNN), табиғи тіл өңдеуде (NLP) жиі қолданылады.



Түрі	Сипаттамасы	Артықшылығы	Кемшіліктері	Қолдану мысалы
Batch Gradient Descent	Толық деректер жиынтығын пайдаланады. Әрбір итерацияда қате мәні модель бойынша қолданылады.	Дәл нәтиже береді, тегіс түсу траекториясы бар.	Үлкен деректер жиынтығы үшін баяу, көп RAM қажет.	Regression, Neural Networks (шағын dataset)
Stochastic Gradient Descent	Әрбір итерацияда градиент тек бір үлгі үшін есептеледі.	Нақты уақыт (онлайн оқыту) есептеулеріне қатысты өте жылдам.	Траектория тұрақсыз «тербеледі» (бұзылу қате не арттыруды, не азайтуды тудырады).	Online Learning, Deep Learning, Recommendation Systems
Mini-Batch Gradient Descent	Біз деректер жиынтығын көптеген кіші	Тепе-теңдік сақтайды: жылдам және	Топ мөлшерін икемдеу өте	CNN, NLP, Deep Learning

	топтарға бөлеміз және әрбірі үшін градиентті есептейміз.	тұрақты. GPU-мен жұмыс істеуде өте ыңғайлы.	маңызды (әдетте 32-512).	(TensorFlow, PyTorch)

Әдебиеттер тізімі:

1. Kordowich, G., & Jaeger, J. (2023). A Physics Informed Machine Learning Method for Power System Model Parameter Optimization. https://arxiv.org/pdf/2309.16579 arXiv+1

2. «Лекция 9. Градиентный спуск и его модификации для решения задачи оптимизации». (2022, 18 февраля). Курс: «Математика для анализа данных». Лекторы: И. Оселедец, А. Катруца. Teach-In AI. https://teach-in.ru/lecture/2022-02-18-Matematiki-1 teach-in.ru

3. «Gradient Descent: Основы Data Science». NeuroHive. https://neurohive.io/ru/osnovy-data-science/gradient-descent/

4. Ruder, S. (2016). An overview of gradient descent optimization algorithms. https://arxiv.org/abs/1609.04747 arXiv+2arXiv+2

5. Jin, C., Netrapalli, P., Ge, R., Kakade, S. M., & Jordan, M. I. (2019). On Nonconvex Optimization for Machine Learning: Gradients, Stochasticity, and Saddle Points. https://arxiv.org/abs/1902.04811 arXiv