8. Optimizer

▼ 8.1 Introduction

최적화란, 수학 방정식을 이용하여 최적의 솔루션을 찾는 컴퓨팅 기술을 의미합니다. 우리가 이 방법을 적용해야 하는 머신러닝 분야에서의 핵심 문제는 파라미터를 잘 추정해야하는데요. 아래의 공식을 보시면, 손실함수를 최소화하는 변수 θ의 집합에 대한 값을 찾는 최적화 문제를 푸는 것이 핵심입니다.

최적화 방법에 대해 공부하기 전에 미리 알아야 하는 개념에 대해 introduction에서 소개하고 있습니다.

8.1.1 첫번째로 지역최적화와 전역최적화에 대한 개념입니다.

- 전역 최적화는 식 8.1을 만족하는 θ를 찾는 과정을 말합니다. 8.1을 만족하는 \theta를 전역최적점이라고 합니다.

$$\exists \delta > 0, \ \forall \boldsymbol{\theta} \in \Theta \ \text{s.t.} \ ||\boldsymbol{\theta} - \boldsymbol{\theta}^*|| < \delta, \ \mathcal{L}(\boldsymbol{\theta}^*) \leq \mathcal{L}(\boldsymbol{\theta})$$

일반적으로 전역최적점을 찾는 것은 매우 어려운 일이기 때문에 우리는 지역 최적점을 잘 찾으려는 노력을 주로 합니다. 같은 목적함수 값을 가지는 지역 최적점으로 주변이 둘러 쌓인 경우를 flat local minimum이라고 하며, 만약 주변 점들보다 현저히 낮은 비용을 갖는 경우는 strict local minimum이라고도 불립니다. 이렇게 지역 최적점을 찾는 알고리즘이 시작점에서 같은 상태를 유지하는 점으로 수렴하고 있으면, 전역적으로 수렴한다고 이야기하며, 그 곳을 지역 최적점으로 인식하는데, 이는 전역최적점을 의미하는 것은 아닙니다.

8.1.1.1 지역 혹은 전역 최적점에 대한 최적의 조건들

연속적이고 두 번 미분 가능한 함수의 경우, 지역 최적점과 일치하는 점들을 정확하게 구할 수 있습니다. 이에 대한 필수조건으로, θ^*가 만약 지역최적점이라면 손실함수의 1차 도함수가 0을 만족해야 하며, 헤이시안 행렬 0 이상이거나 0임을 의미합니다.

8.1.2 다음으로 알아야 하는 개념은 제약 최적화와 비제약 최적화입니다.

- 말 그대로 제약 최적화는 제약 조건이 존재하는 경우를 말하며, 비제약 최적화는 제약 조건이 없고 2개 이상의 변수로 구성된 함수의 최소 또는 최댓값을 구하는 최적화 문제를 말합니다. 여기서 제약 조건은 등식 제약과 부등식 제약으로 존재하며, 이러한 제약 조건을 만족하는 집합을 feasible set 실현 가능한 영역이라고 합니다.
- 제약 조건이 많은 문제는 목적함수에 맞는 penalty term을 주어 라그랑지 승수법을 활용하여 문제를 푼다고 하는데, 이에 대한 자세한 내용은 뒤에 나올 라그랑지 승수에 대해 설명하면서 이야기하도록 하겠습니다.

8.1.3 세번째로 알아야 하는 개념은 convex와 nonconvex 최적화에 대한 개념입니다.

- 최적화 문제를 convex 함수로 변환함으로써 gradient descent 등의 방법을 통해 쉽게 해결 가능하다고 합니다.

  여기서 convex 함수란, 임의의 두 점을 이은 할선이 두 점을 이은 곡선보다 위에 있거나 같은 위치에 있는 함수를 말합니다. convex하다는 것은 볼록하다로 알고 계시는데, 여기서 말하는 볼록은 아래로 볼록을 의미하며, 정확한 정의는 두 점을 선으로 연결했을때 해당 선 위의 모든 점들이 해당 집합에 포함되는 경우를 말합니다. 책에서는 임의의 두 점을 이은 할선이 두 점을 이은 곡선보다 위에 있는 함수를 strictly convex라고 하며, convex함수에 -값을 붙이면 concave함수가 됩니다.
- 그래서 convex함수로 만들어야 하는 이유는 전역 최솟값을 향해 안정적으로 수렴할 가능성이 높기 때문입니다. 그리고 이전에 제가 얘기했던 지역 최적점을 만족하기 위해서는 헤이시안 행열의 부호가 0보다 커야 한다고 했는데, 이에 대한 정리를 여기서 이야기하고 있습니다. convex하다는 것을 여러 경우에 따라 종류를 나누어 이야기하는데, 우리가 구하고자 하는 목적함수를 두 번 미분한 결과가 0보다 크거나 같으면 convex하다, 0보다 크기만 하면, strictly convex하다, 0보다 크면서 특정 파라미터 m보다 크면 strongly convex하다고 볼 수 있다고 합니다.

8.1.4 마지막으로 알아야하는 개념은 부드러운 최적화와 부드럽지 않은 최적화에 대해 이야기하면서 본격적인 최적화 내용으로 들어갑니다.

- 우선 부드러운 최적화는 목적함수와 제약 조건이 연속적으로 비분 가능한 함수이며, 부드러운 함수에 대한 특성을 립시츠 상수를 사용하여 부드러운 정도를 정량화할 수 있습니다. 립시츠 상수는 함수의 미분이 얼마나 급격하게 변하는지를 나타내는 값으로 다음 식의 변화가 입력 변수간의 거리에 비례하다면 잘 변하지 않는다는 것을 이야기하고 있습니다.
- 반면에 부드럽지 않은 최적화는 목적함수나 제약 조건이 미분 불가능하거나 불연속적인 경우를 다루는 분야로, 해당 문제에서는 연속적인 부분과 그렇지 않은 부분으로 식

을 분할합니다. 그리고 머신러닝 분야에서는 연속적인 부분을 훈련 세트의 손실함수로 정의하고 불연속적인 부분을 전체식의 penalty term처럼 정규화하는 영역으로 정의하여 문제를 해결하고자 합니다.

    - 이렇게 미분이 불가능한 경우에 활용되는 개념으로 subgradient를 소개하고 있습니다. 서브 그래디언트란, 미분이 불가능한 경우에 기울기 대신 사용하는 변수로 기울기의 일반화된 개념이라고 보시면 됩니다.

      절댓값 함수를 예로 들어 설명하고 있는데, 절댓값함수는 0에서 미분이 불가능합니다. 이런 경우, subgradient를 활용하여 다음과 같이 구간을 분할하여 미분값을 적용합니다.

▼ 8.2 First-order methods

1차 미분 방법

이제 본격적으로 최적화에 대한 이야기를 시작하려고 합니다. 이 챕터에서는 목적함수의 1차 미분을 이용하여 반복적으로 최적화 문제에 접근합니다. 이러한 모든 알고리즘은 사용자가 시작점을 지정해야 하며, 학습률에 맞춰 반복횟수 t만큼 업데이트를 수행합니다. 여기서 η_t는 학습률, 스텝 크기를 말하며, d_t는 감소하는 방향이라고 보시면 됩니다. 이 업데이트 과정은 정적인 지점에 도달할때까지 지속됩니다.

이제 책에서는 업데이트를 진행하는 위 식의 변수들에 대해 자세히 설명하고 있습니다.

8.2.1 descent direction

- d가 하강 방향이라고 하면, 그 방향으로 조금만 이동해도 함수값이 감소할 것이라고 보장할 수 있습니다. 수식적으로는 8.16을 만족하는 경우를 말합니다. 여기서 θ는 현재 위치를 말하며, α는 0보다 크면서 충분히 작은 양수를 말합니다. 그리고 현재 위치에서의 그래디언트는 다음과 같이 주어지는데, 이 그래디언트는 현재 위치에서 함수 L이 어떤 방향으로 가장 크게 증가하는지를 가리킵니다. 그래디언트는 함수의 값이 가장 빠르게 증가하는 방향을 나타내므로 이 방향의 반대 방향이 바로 함수를 감소시키는 방향, 즉 하강 방향입니다.

8.2.2 step size

하강 방향에 대해 이해 되었다면, 이제 얼만큼의 간격으로 학습을 진행할 것인지를 다루는 step size 학습률에 대한 개념으로 넘어가겠습니다.

- 고정된 스텝 크기를 사용할 수도 있습니다. 하지만 학습률이 너무 크면 이 방법은 수렴에 실파하며, 너무 작아도 매우 느리게 수렴하다는 것을 그림 8.11에서 보여주고 있습

니다. 일반적으로 립시츠 상수인 L분의 2보다 작은 학습률의 값을 세팅하는것이 수렴을 보장한다고 합니다.

- 고정된 스텝 크기를 사용하는 방법 외에 Line search라는 방법도 소개합니다. 이는 주어진 방향으로 얼마만큼 나아갈지를 결정하기 위해 사용되는 방법으로, 1차 최소화 문제를 풀면서 선택된 방향에 따라 목적함수를 최소화한다. 대표적으로 활용되는 방법 중 하나가 아미조 backtracking method로, 최적의 학습률을 찾는데 너무 많은 시간을 쓰지 않고 목적함수를 효율적으로 줄이는 알고리즘이다. 이 알고리즘은 초기 스텝 크기를 설정하고 아미조 조건을 검사합니다. 아미조 조건은 주로 현재 위치에서의 실제 감소량이 기대한 감소량의 일정 비율이상이어야 한다는 원칙을 기반으로 하며 이 조건을 충족하면 종료, 그렇지 않으면 스텝 크기를 축소하면서 반복합니다.

8.2.3 수렴속도

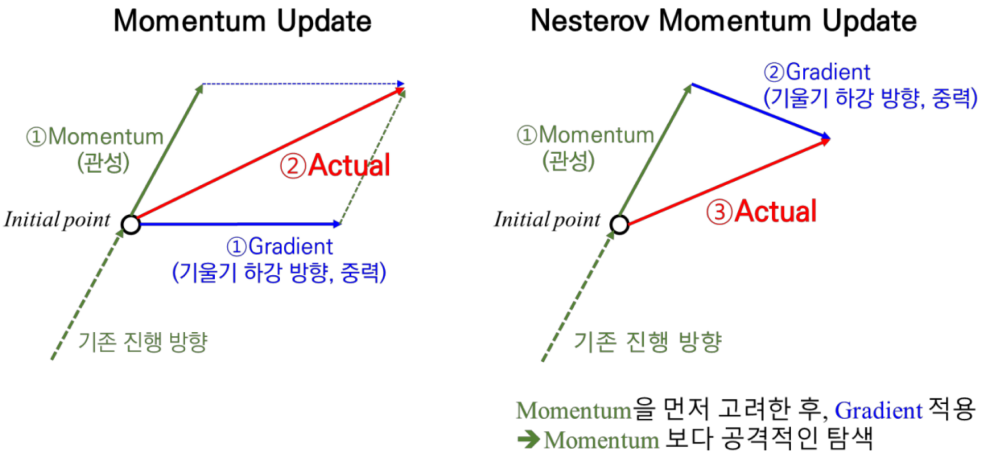
스텝 크기 이외에도 수렴속도가 어떠한지 따져보아야 합니다.

- 이차 목적함수를 예로 들어 설명하고 있는데, 식 8.29에 따르면, 주어진 행렬 A의 고윳값 중 가장 큰 것과 가장 작은 값을 가지고 학습률을 계산할 수 있습니다. 여기서 μ를 κ를 사용하여 다시 정의할 수 있으며, 카파는 행렬 A의 조건수입니다. 조건수란 함수의 입력인 x의 변화율에 대해 y의 변화율이 얼마인지를 나타내는 수로, 높은 조건수를 가지면 작은 변화에 대해 큰 변화가 있다는 의미로 수치적인 안정성이나 수렴 속도에 영향을 어느 정도 주고 있는지 파악할 수 있습니다.

8.2.4 모멘텀 방법

경사하강법은 평평한 손실함수 영역에서 매우 느리게 움직이기 때문에 saddle point 혹은 local minimum에 빠질 수 있습니다. 따라서 이를 가볍게 툭 치고 넘어갈 관성을 활용하여 보완하고자 하는 방법으로 모멘텀 방식을 제안한 것입니다.

- 모멘텀을 적용하여 θ를 업데이트하는 식은 다음과 같습니다. β가 0이면 경사하강법과 동일한 식이며, 주로 0.9의 값으로 계산합니다.
- 모멘텀 자체만 적용했을 때, 진동현상이 발생할 수 있다. 그래서 Nesterov 모멘텀은 모멘텀을 고려한 후 그래디언트를 적용하는 방식이다.



▼ 8.3 Second-order methods

2차 미분 방법

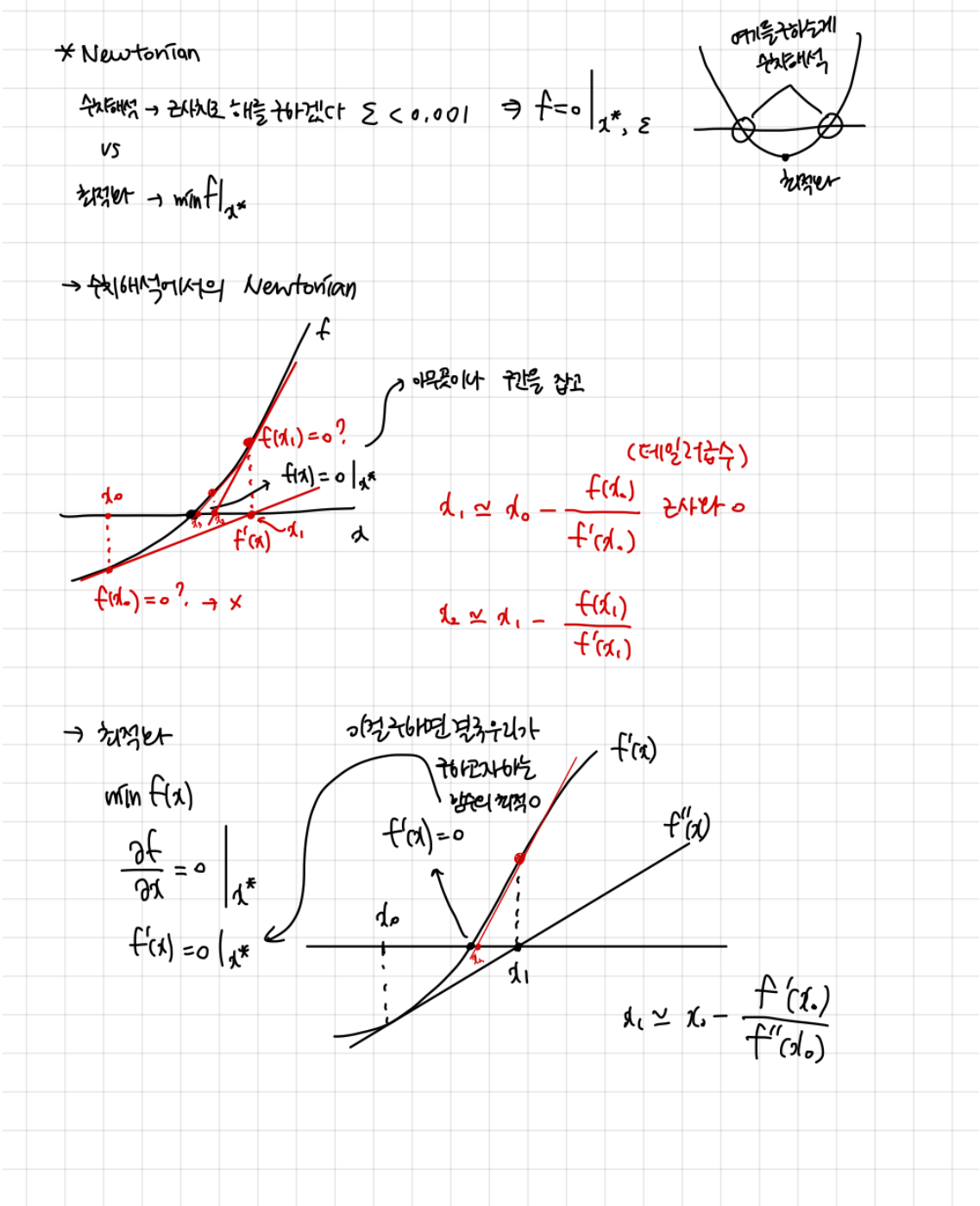
단순히 1차 미분 방법만으로는 함수의 곡률 또는 공간의 형태에서 모델링이 어려워 2차 도함수를 활용하는 방법을 적용하였다.

8.3.1 뉴턴 방법

- 2차 도함수를 적용한 방법 중 하나가 뉴턴 방법이다. 뉴턴 방법은 두 번 미분 가능한 함수에 대하여 2차 미분 테일러 급수로 함수를 근사한 뒤, 근사 함수의 최솟값을 찾으며 해에 접근하는 방법이다.

  다음 식에서 g_t는 1차 도함수이며, H는 2차 도함수 행렬로 양의 정부호여야 합니다.

  알고리즘 8.1에서 제시하는 수도코드를 살펴보면, 손실함수의 1차 미분값을 구하고 2차 미분값을 구한 뒤, 하강방향을 찾아 다음 파라미터를 추정하는 방식이다. 이러한 방식이 적용되는 이유는 수치해석에서의 뉴턴 방정식을 참고하면 더 이해할 수 있습니다.



- 우리가 구해야 하는 Hessian의 연산량이 너무 크거나, singular(특이)한 경우, 쿼사이 뉴턴 메소드를 사용하여 헤이시안을 근사화하여 업데이트를 수행한다. 헤이시안 행렬을 근사화하여 적용하는 것이 BFGS 방법이다. 여기서는 B행렬이 양의 정부호이면서 $s^T y$의 연산결과가 0보다 크면, B행렬도 양의 정부호여야 한다는 positive definiteness를 유지해야 한다. 그리고 헤이시안 행렬을 계산하거나 저장하기 위한 비용이 합리적이지 않을 경우 유용하게 사용하는 방법이 LBFGS이다. 이는 n*n의 헤이

시안 행렬을 저장하는 대신 n차원의 벡터 몇 개만을 유지하여 헤이시안 행렬을 추정하는 방법이다.

- 2차 근사가 유효하지 않을때, 즉 2차 근사로는 함수의 특성을 설정할 수 없는 지점에서는 뉴턴 매소드가 동작하기 어렵다. 따라서 신뢰 영역을 알아내는 방법을 통해 목적함수를 안전하게 2차 함수로 근사화할 수 있는 지역을 찾아 문제를 해결하는 방법도 있다. 여기서 말하는 신뢰 영역을 R_t라고 하고 이를 찾는 모델을 M(δ)라고 한다. 이러한 영역을 잘 찾아 그 안에서 적절한 모델로 근사화하는 신뢰 영역 최적화라고 한다.

  8.50의 식은 현재 위치에서의 목적함수를 그랟디언트와 헤이시안 정보를 활용하여 근사화하고 있다. 제약 조건이 R_t에 있다는 가정하에 제약 조건이 있는 문제도 위 구조를 활용하면 제약이 없는 문제로 변환할 수 있다.

▼ 8.4 SGD

SGD는 목적함수 값의 평균을 최소화하는 것을 목표로 하는 알고리즘을 말한다. 여기서 z는 목적함수에 대한 임의의 입력데이터이며, 8.54에 제시된 식에 대하여 한 번의 파라미터를 업데이트하기 위해 확률적으로 선택된 훈련데이터만 사용하는 방법이다.

8.4.1

- 이러한 SGD는 주로 유한합 문제에서 활용됩니다. 유한합 문제란 함수가 여러항의 합으로 표현되는 경우를 말하며, 각 단계에서는 하나의 훈련 샘플에 대한 그래디언트를 이용하여 전체 손실 함수의 그래디언트를 대표적으로 근사화하기 때문에 계산 효율성을 얻을 수 있습니다.

8.4.2

- 또한, 선형회귀에 SGD방식을 적용하는 예시를 책에서 제시하고 있습니다. 여기서 손실함수로 MSE를 사용하고, 각 반복에서 훈련 데이터의 무작위 샘플을 선택하고 해당 샘플에 대한 손실 함수의 그래디언트를 계산합니다. 미니배치의 크기를 1로 두고 학습률*에러*현재 입력 데이터로 업데이트 하는 방법을 LMS 알고리즘이라고 합니다.

8.4.3

- 여기에서도 적절한 학습률을 선택하는 것이 중요한데요. 특히, SGD는 학습률에 굉장히 민감합니다. 좋은 학습률을 고르는 heuristic한 방법 중 하나는 작은 학습률에서 시작하여 점진적으로 커지는 방법이 있다. 일정한 학습률을 사용하여 수렴이나 성능향상이 멈출 때까지 일정하게 유지하는 방법보다는, learning rate schedule을 활용하기도 한다. 이론적으로 learning rate schedule이 Robbins-monro조건을 만족하면 SGD는 수렴한다고 합니다. 이 이론의 첫번째 조건은 학습률의 무한 합이 무한대여야하며, 학

습률의 제곱의 무한 합이 유한해야 한다는 것을 나타내고 있습니다. 이 조건이 만족되면, SGD가 안정적으로 수렴할 수 있다는 것을 의미합니다.

이렇게 여러 학습률을 조절하기 위해서 어떤 전략이 필요한지 아래에서 제시하고 있습니다. piecewise constant schedule 방법은 일련의 시간 지점 t에서 학습률을 특정 값으로 조절합니다. 예를 들어, 다음과 같이 설정하여 각 임계점을 지날 때마다 초기 학습률을 \gamma의 배수로 감소시키는데 이를 스텝 감소라고 합니다. 때로는 임계 시간이 훈련 또는 검증 손실이 안정화될 때 추정되어 동적으로 계산될 수 있는데, 이를 reduce-on-platequ라고 합니다.

지수 감소의 경우, 일반적으로 너무 빠르기 때문에 \alpha는 0.5 그리고 \beta는 1로 하는 다항식 감소로 변환할 수 있습니다. 이를 통해 초기에 학습률을 감소시키고, 그 후 다시 서서히 감소시키는데, 이러한 전략을 학습률 웜업이라고 합니다. 또한, 학습률을 주기적으로 여러번 증가하고 감소시키는 순환 학습률을 적용할 수도 있으며, 모델이 유연한 경우, 라인 서치를 통해 그래디언트 노이즈의 분산을 조절하는 전략을 취할 수도 있습니다.

이러한 학습률 조절 전략을 통해 효율적으로 최적화시킬 수 있다는 것을 제시하고 있습니다.

8.4.4

Iterate averaging은 주로 분산이 크거나 노이즈가 있는 경우, 모델의 안정성을 높이기 위해 여러 반복의 결과를 평균하여 최종적인 모델 파라미터를 얻는 방법을 말합니다.

이 과정이 반복된 평균을 취함으로써 통계적 이점을 가질 수 있고, 선형 회귀의 경우, L2 정규화와 동등하다고 합니다. 이 책에서는 다른 참고서를 통해 증명되었다고 이야기하는 것 같아 이해하기 어려워 discussion으로 남겨두었습니다.

8.4.5

그리고 SGD에서 분산을 감소시키는 방법에 대해 이야기합니다. 이 방법들은 파라미터 자체가 아니라 그래디언트의 분산을 감소시키는 방법이다.

- SVRG는 제어변수를 활용하여 분산을 감소시키는 방법으로 주기적으로 전체 데이터 셋에 대한 그래디언트를 계산하고, 이를 ‘snapshot’이라 부르며, 이를 기준으로 step t에서 확률적 그래디언트와 비교하여 분산을 줄이는 방법을 적용합니다.
- SAGA는 SVRG와 달리, 알고리즘의 시작부분에서 전체 배치의 그래디언트를 한 번만 계산합니다. 특히, N개의 그래디언트 벡터를 저장하여 이전의 지역 그래디언트를 전체 합에서 제거하고 새로운 그래디언트로 더함으로써 근사치를 유지합니다.

- 또한, 딥러닝에서도 이와 같은 방법들을 적용할 수 있지만, SVRG와 같은 분산 감소 방법은 목적 함수의 특정 속성에 의존하는데, 딥러닝에서 일반적으로 사용되는 기술들인 배치 정규화, 데이터 증강, 드롭 아웃과 같은 기법들은 목적함수의 특성을 변경시키거나 무작위성을 추가함으로써 기존의 가정을 깨뜨릴 수 있다고 합니다.

8.4.6

사전 조건을 도입하여 SGD의 문제점을 보완할 수 있습니다. 하지만 그래디언트 추정치의 노이즈로 인해 헤시안을 신뢰성 있게 추정하는 것이 어렵고 데이터 전체에서 필요한 사전 조건 행렬을 사용하여 업데이트 방향을 찾는 것은 비용이 많이 듭니다. 따라서 일반적으로 대각선 사전 조건 행렬을 사용하여 효율적으로 계산하도록 한다고 합니다. 그렇게 변형을 주어 SGD의 문제점을 보완하는 알고리즘을 소개하도록 하겠습니다.

- 지금까지 알아본 SGD, 모멘텀, NAG방법들의 공통점은 모든 파라미터에 동일한 학습률을 적용하여 업데이트하였습니다. 하지만, sparse한 데이터, 즉, 자주 나오지 않은 단어가 포함된 데이터를 가지고 학습이 이루어져야하는 경우, 동일한 가중치가 아닌 각 영역별 가중치를 달리 주어야 하는 것이 어떠할까 하는 아이디어에서 시작된 것이다. 또한, 학습이 깊어질수록 효과적으로 감소할 수 있도록 해야 한다는 아이디어가 반영된 알고리즘이다. 루트 값을 연산하여 각 영역별 가중치를 계산하여 반영하도록 한다.

- 하지만 이러한 Adagrad에서도 학습률이 소실되는 문제가 있었는데, 이를 완화시킨 알고리즘이 RMSProp과 adadelta이다. RMSProp은 이전 누적치와 현재 그래디언트를 제곱하여 가중치평균을 고려하여 파라미터를 업데이트하는 방식을 말하며, adadelta는 몇 개의 그래디언트 정보만 저장하여 파라미터를 업데이트하는 방식을 말한다.

- 여기에 모멘텀과 RMSProp의 개념을 합친 ADAM이라는 알고리즘이 등장하면서 옵티마이저의 절대강자가 등장하였다. 이는 적응형 모멘트 추정으로 불리며, 관성의 개념과 가중치 평균을 함께 고려하도록 설계되었다. 주로 \beta_1은 0.9, \beta_2는 0.999 그리고 입실론은 10^(-6)을 사용한다고 한다. 또한, 초기 설정을 원점으로 두면, 이 점 근처에서만 업데이트 하므로 보정작업을 하여 공정하게 만들어준다.

- 이러한 적응형 학습률 방법도 결국은 초기 설정을 해주어야 한다는 문제가 있다. 이를 보완하기 위해 다음 식에서 Adam을 수정하여 문제를 해결하고자 하였고, 최근의 연구에서 데이터셋 별로 매개변수를 조정하면 일반적으로 adam이 항상 수렴할 수 있다고 보여주었다고 합니다.

- 8.4.6.5에서는 위에 논의된 방법들은 각 매개변수의 학습률을 적응적으로 조절할 수 있지만, 매개변수 간의 상관관계로 인한 잘못된 조건화 문제에 대한 근본적인 문제를 해결하지 못하므로 항상 기대만큼 SGD에 비해 빠른 속도 향상을 제공하지 않는다고 의문을 제기합니다. 보다 빠른 수렴을 얻는 한 가지 방법은 다음의 사전 조건 행렬을 사용하는 것입니다. 하지만 역행렬을 계산하는 과정에서 굉장한 비용이 많이 든다고 합니다. 샴푸 알고리즘은 M에 대한 블록 대각선 근사를 수행하고 모델의 각 레이어에 대해 kronecker 곱 구조를 활용하여 효율적으로 역행렬을 계산합니다. 이 알고리즘이 굉장한 효과를 얻었다고 합니다.

▼ 8.5 Constrained optimization

제약 최적화

introduction에서도 이야기했지만, 제약 최적화는 제약 조건이 있고 2개 이상의 변수로 구성된 함수의 최소 또는 최댓값을 구하는 최적화 문제입니다. 제약 조건은 등식 제약과 부등식 제약이 있습니다. 이러한 제약 조건을 가지고 문제를 푸는 여러가지 방법을 소개하고 있습니다.

8.5.1 라그랑지 승수

- 등식 제약 최적화에서 특정한 등식 제약이 주어진 상황에서 목적함수를 최소화하려고 합니다. 이때 등식 제약 표면 상의 한 점에서는 그래디언트 벡터가 제약 표면에 수직이라는 성질이 있습니다. 더불어 이러한 최적해에서 목적 함수의 그래디언트도 제약 표면에 수직이어야 합니다. 그래서 이 두 벡터는 서로 평행(또는 반대방향)이어야 하며, 이를 나타내기 위해 라그랑지 승수가 등장하게 되었고 다음과 같은 식이 성립하게 된 것입니다. 이러한 라그랑주 승수는 최적 해에서의 제약 조건을 고려하여 목적함수를 최소화하는데 사용됩니다. 이를 만족하는 식을 라그랑지안이라고 하며, 이를 통해 얻은 점을 critical point라고 합니다. 라그랑지 승수를 통해 문제를 해결하는 과정이 담긴 예시는 다음 절에서 소개하고 있고, 이는 미분하는 과정이므로 생략하도록 하겠습니다.

8.5.2 KKT 조건

여기서는 라그랑지 승수에 부등식 조건이 들어가는 경우를 일반화하여 소개하게 있습니다.

주어진 손실함수를 최소화하는데 g(x)가 0보다 작거나 같고 h(x)가 0인 조건을 만족해야 하는 경우, 우선 해당 제약 조건을 람다로 연결하여 하나의 식으로 정리하기 위해 필요한 조건들을 연결된 식과 람다에 대하여 제시하는 과정을 KKT 조건이라고 합니다.

https://www.youtube.com/watch?v=xd641yJvXTk

8.5.3 선형 프로그래밍

선형 프로그래밍은 선형 관계를 갖는 제약 조건 하에서 선형 목적 함수를 최소화하거나 최대화하는 수학적 최적화 기법입니다.

- Simplex 알고리즘은 선형 프로그래밍 문제를 해결하기 위한 반복적인 최적화 알고리즘 중 하나로, 다차원 공간에서 다각형의 꼭짓점을 따라 최적해를 찾는 알고리즘입니다. 이 다각형은 최적해가 존재하는 영역을 표현하며, 알고리즘이 반복되면서 이 다각형의 꼭짓점을 이동하며 최적해를 찾아갑니다.
- 선형 프로그래밍은 비즈니스, 경제학, 과학 등의 응용분야에서 효과적으로 사용되는 문제입니다.

8.5.4 quadtratic programming

쿼드라틱 프로그래밍은 선형 등식 및 부등식 제약 조건이 있는 이차 목적 함수를 최소화하는 문제를 말합니다. 목적함수가 2차함수면서 제약 조건이 일차함수인 경우 2차 계획법에 해당하여 무조건 최적해가 존재하며 이 경우 KKT 조건을 가지고 문제를 풀면 최적해를 구할 수 있습니다.

▼ 8.6 Proximal gradient method

proximal gradient descent는 목적함수를 미분가능한 함수와 그렇지 않은 함수로 분리해서 최적해를 찾는 방법이다.