Домашнее задание - оптимайзеры

Чэнь Сюаньдун
Группа 519/2
13 Ноября 2022 г.

В чем идея и зачем нужно ускорение (accelerate)

Как найти оптимальное решение в сложной объективной функции породило ряд алгоритма оптимайзеров.

Многие из используемых в настоящее время алгоритма оптимайзеров являются производными и оптимизациями метода градиентного спуска(GD).

Например, BGD, SGD, MBGD, SGD+Momentum, NAG, AdaGrad, AdaDelta, RMSprop, Adam, Nadam и т.д.

SGD используется во многих работах. SGD может достигать экстремальных значений, но это занимает больше времени, чем другие алгоритмы(кроме BGD, MBGD), и может застрять в седловых точках.

Если данные разрежены, использовать адаптивные методы, т.е. Adagrad, Adadelta, RMSprop, Adam. RMSprop, Adadelta, Adam во многих случаях дают схожие результаты.

Adam - это добавление коррекции смещения и импульса к RMSprop. Поскольку градиент становится разреженным, Adam лучше, чем RMSprop. В целом, Adam - лучший выбор.

Если требуется более быстрая сходимость или обучение более сложных нейронных сетей, необходим адаптивный алгоритм.

1 AdaGrad/Adaptive Gradient

Идея:

Алгоритм AdaGrad регулирует скорость обучения в каждом измерении в зависимости от величины градиента независимой переменной в каждом измерении, что позволяет избежать проблемы, связанной с тем, что единую скорость обучения трудно адаптировать ко всем измерениям.

Формула:

$$s_t = s_{t-1} + g_t \odot g_t$$

$$x_t = x_{t-1} - \frac{\eta}{\sqrt{s_t + \epsilon}} \odot g_t$$

W - обновляемый весовой параметр; $g_t = [g_t^1, g_t^2, ..., g_t^n]^T$ - градиент функции потерь относительно x; η - скорость обучения; ϵ - гиперпараметр, используемый для того, чтобы знаменатель не был равен нулю; $g_t \odot g_t = [g_t^1 * g_t^1, g_t^2 * g_t^2, ..., g_t^n * g_t^n]^T$

Преимущества:

Алгоритм Adagrad автоматически регулирует скорость обучения во время обучения, используя большие обновления для менее частых параметров и, наоборот, меньшие обновления для более частых параметров. Поэтому Adagrad хорошо подходит для работы с разреженными данными.

Недостатки:

Поскольку на каждой итерации в знаменатель добавляется положительное число, скорость обучения становится все меньше и меньше или даже близка к нулю. Поэтому скорость обучения на поздних итерациях очень мала, и легко начать неэффективные итерации до нахождения оптимального решения.

2 AdaDelta

Идея:

Вместо того чтобы накапливать все прошлые градиенты, Adadelta ограничивает окно накопленных прошлых градиентов некоторым фиксированным размером. Также использует дополнительную переменную состояния Δx_t, вместо скорости обучения.

Формула:

$$s_t = \rho s_{t-1} + (1 - \rho) g_t \odot g_t$$

$$g_t^{'} = \sqrt{\frac{\Delta x_{t-1} + \epsilon}{s_t + \epsilon}} \odot g_t$$

$$x_t = x_{t-1} - g_t^{'}$$

$$\Delta x_t = \rho \Delta x_{t-1} + (1 - \rho) g_t^{'} \odot g_t^{'}$$

Преимущества:

Нет необходимости вручную задавать скорость обучения.

По сравнению с Adagrad, AdaDelta замедляет затухание скорости обучения.

Недостатки:

На более поздних этапах обучения существует риск повторного возникновения флуктуаций в районе локальных минимумов.

3 Adam/Adaptive Moment Estimation

Идея:

Adam основан на RMSProp с двумя улучшениями: градиентное скользящее среднее и коррекция смещения.

Формула:

$$V_t = \beta_1 * v_{t-1} + (1 - \beta_1) g_t$$

$$s_t = \beta_2 s_{t-1} + (1 - \beta_2) g_t \odot g_t$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_1^t}$$

$$\hat{s}_t = \frac{s_t}{1 - \beta_2^t}$$

$$g_t^{'} = \frac{\eta \hat{v}_t}{\sqrt{\hat{s}_t} + \epsilon}$$

$$x_t = x_{t-1} - g_t^{'}$$

Преимущества:

Алгоритм Adam использует коррекцию смещения. Когда градиент становится разреженным, Adam работает лучше, чем RMSprop.

Размер обновления параметров не меняется при масштабировании размера градиента.

Границы размера шага при обновлении параметров ограничены настройкой размера шага суперпараметра.

Он не требует фиксированной объективной функции; поддерживает разреженные градиенты;

Недостатки:

Большие вариации данных могут привести к колебаниям скорости обучения, что приводит к неспособности сходиться на поздних этапах обучения.

Может произойти чрезмерная подгонка признаков, которые появляются раньше, и не найти глобального оптимального решения.

4 Gradient Descent

Формула:

$$\theta = \theta - \alpha \nabla_\theta J(\theta)$$

Преимущества:

Градиент всегда в правильном направлении, т.е. в направлении с наибольшим падением скорости

Недостатки:

Медленное обновление, трудно регулировать скорость обучения

5 Stochastic Gradient Descent

Идея:

при каждом обновлении параметр необходимо просмотреть только одну обучающую выборку для обновления, а затем следующую выборку для следующего обновления.

формула:

$$\frac{\partial J(\theta)}{\partial \theta_j} = (h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

$$\theta = \theta - \alpha \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

Преимущества:

Поскольку вместо функции потерь на всех обучающих данных, функция потерь на конкретных обучающих данных оптимизируется случайным образом на каждой итерации, таким образом, параметры обновляются гораздо быстрее на каждом раунде.

Недостатки:

Необходимо установить хорошую скорость обучения, слишком маленькая приводит к слишком медленной сходимости, слишком большая - к превышению экстремального значения.

Поскольку скорость обучения остается постоянной во время итераций, это иногда приводит к тому, что алгоритм застревает в седловой точке

На более плоских участках алгоритм оптимизации может выполнять итерации раньше и застрять в локальных экстремумах из-за неправильной оценки, прежде чем достигнет экстремумов, когда градиент приблизится к нулю.

6 Mini-Batch Gradient Descent

Идея:

Для каждого обновления параметров использовать небольшую часть выборочной совокупности

Формула:

$$\theta_j = \theta_j - \frac{\alpha}{batch\_size} \sum_{i=k}^{batch\_size} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Преимущества:

Количество итераций, необходимых для сходимости, может быть значительно уменьшено, при этом результаты сходимости близки к градиентному спуску

Параллелизация может быть достигнута.

Недостатки:

Неправильный выбор размера партии (batch size) может вызвать некоторые проблемы.

Размер партии необходимо выбирать вручную

7 SGD+Momentum

Идея:

SGD+Momentum заимствует понятие Momentum из физики, сохраняя направление последнего градиента обновления и комбинируя его с текущим градиентом для определения окончательного направления обновления.

Формула:

$$m_t = \mu m_{t-1} + \alpha \nabla_\theta J(\theta)$$

$$\theta_t = \theta_{t-1} - m_t$$

Преимущества:

Направление обновления параметров будет более благоприятным для сходимости, что поможет уменьшить осцилляции и сходиться быстрее

Недостатки:

Одинаковая скорость обучения используется для всех признаков, но для разреженных признаков предпочтительна большая скорость обучения

8 RMSprop/Root Mean Square Propagation

Идея:

RMSprop использует экспоненциально взвешенное скользящее среднее (exponentially weighted moving average). RMSprop является развитием Adagrad.

Формула:

$$s_t = \gamma s_{t-1} + (1-\gamma) g_t \odot g_t$$
$$x_t = x_{t-1} - \frac{\eta}{\sqrt{s_t + \epsilon}} \odot g_t$$

Преимущества:

RMSprop замедляет затухание скорости обучения.

Недостатки:

Начальная скорость обучения должна быть установлена вручную.

9 NAG/Nesterov Accelerated Gradient

Формула:

$$m_t = \mu m_{t-1} + \alpha \nabla_\theta J(\theta - \mu m_{t-1})$$

$$\theta_t = \theta_{t-1} - m_t$$

Преимущества:

Направление обновления параметров будет более благоприятным для сходимости, что поможет уменьшить осцилляции и сходиться быстрее

Недостатки:

Одинаковая скорость обучения используется для всех признаков, но для разреженных признаков предпочтительна большая скорость обучения