Московский государственный
технический университет
имени Н.Э. Баумана

Кафедра ИУ5
«Системы обработки информации
и управления»

Адаптированный курс для ГУИМЦ по глубокому обучению

Разные виды оптимизаторов

Канев Антон Игоревич
преподаватель кафедры ИУ5

aikanev@bmstu.ru

Немного теории

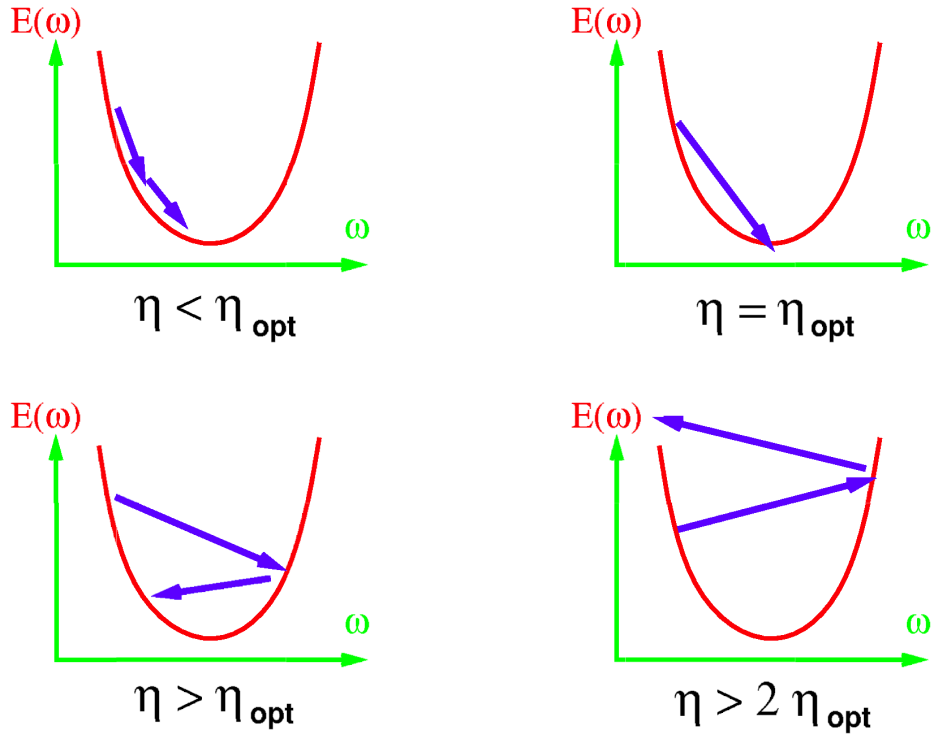
Градиентный спуск в одном измерении

$$\omega \leftarrow \omega - \eta \frac{\partial E}{\partial \omega}$$

gradient of objective function

weight vector

learning rate



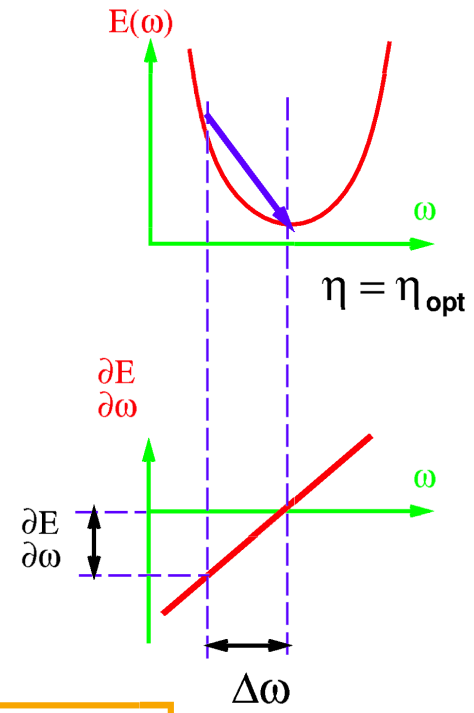
Оптимальный шаг обучения в 1D

Weight change:

$$\Delta\omega = \eta \frac{\partial E}{\partial \omega}$$

Assuming E is quadratic:

$$\frac{\partial^2 E}{\partial \omega^2} \Delta\omega = \frac{\partial E}{\partial \omega}$$



Optimal Learning Rate

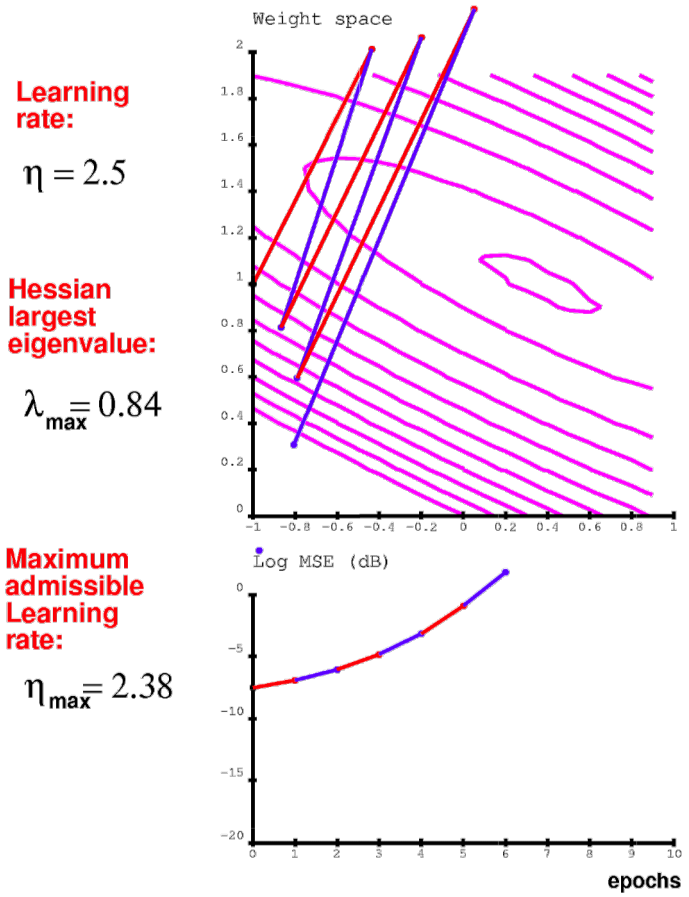
$$\eta_{opt} = \left(\frac{\partial^2 E}{\partial \omega^2} \right)^{-1}$$

Maximum Learning Rate

$$\eta_{max} = 2\,\eta_{opt}$$

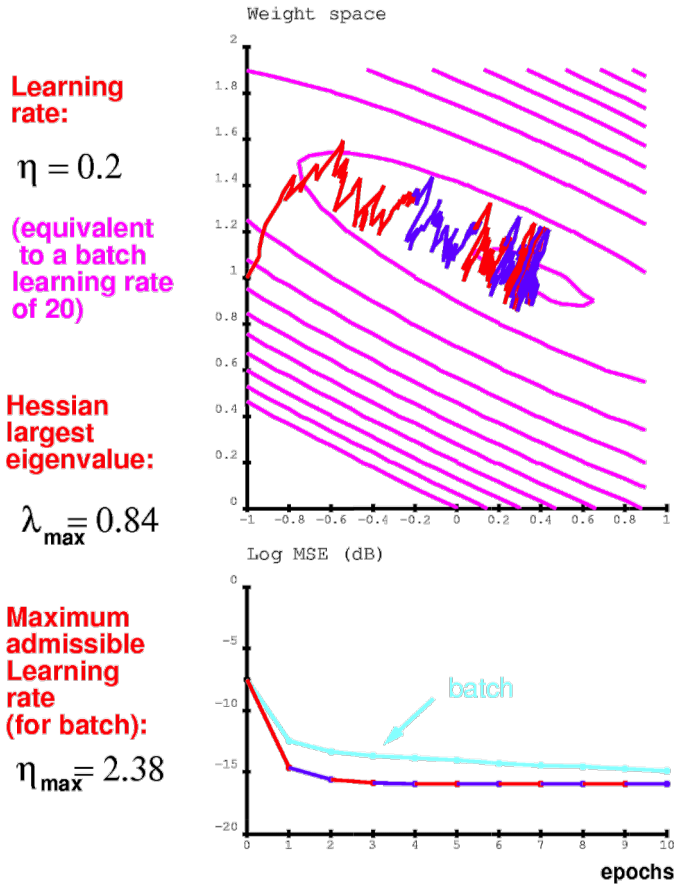
Пакетный градиентный спуск

– набор данных: набор-1 (100 примеров, 2 гауссова распределения) сеть: 1 линейный нейрон, 2 входа, 1 выход. 2 веса, 1 отступ.



Стохастический градиентный спуск

– набор данных: набор-1 (100 примеров, 2 гауссова распределения) сеть: 1 линейный нейрон, 2 входа, 1 выход. 2 веса, 1 отступ.



Стохастическое vs пакетное обновление

- Стохастическое обновление обычно НАМНОГО быстрее, чем пакетное обновление. Особенно на больших избыточных наборах данных.
- Вот почему:
  - Представьте что у вас есть обучающий набор из 1000 примеров.
  - Этот обучающий набор состоит из 10 экземпляров по 100 примеров.
- Пакет (батч): вычисление для одного обновления будет в 10 раз больше необходимого
- Стохастическое: будет использоваться избыточность в учебном наборе для получения преимущества обучения. Одна эпоха на большом наборе будет похожа на 10 эпох на меньшем наборе.
- Пакетное обновление будет КАК МИНИМУМ в 10 раз медленнее стохастичного
- В реальной жизни повторения редко происходят, но очень часто обучающие примеры очень избыточны (много примеров похожи друг на друга), что имеет тот же эффект.
- На практике нередки разницы скорости (на порядки) между пакетным и стохастическим обновлением.
- Маленькие пакеты могут использоваться без штрафа, если примеры в минибатче не слишком похожи.

Стохастическое vs пакетное обновление

Стохастическое

- Преимущества:
  - Быстрая сходимость на больших избыточных данных
  - Стохастическая траектория позволяет избежать локальных минимумов
- Недостатки:
  - Продолжает «прыгать», если скорость обучения не уменьшается
  - Теоретические условия сходимости не так понятны, как для пакетного обновления
  - Доказательства сходимости вероятностны
  - Большинство хороших способов ускорения или методов второго порядка не работают со стохастическим градиентом
  - Сложнее распараллелить, чем пакетное обновление

Пакетное

- Преимущества:
  - Гарантированное сходимость к локальному минимуму в простых условиях
  - Много способов и методов второго порядка для ускорения
  - Простые доказательства сходимости
- Недостатки:
  - Болезненно медленный на больших задчах
- Несмотря на длинный список недостатков для стохастичного обновления, это то, что большинство людей используют (что справедливо, по крайней мере, для больших задач).

Стохастический градиентный спуск

$L(f(\boldsymbol{x}(i); \boldsymbol{\theta}), \boldsymbol{y}(i)$ – значение функции потерь
$f(\boldsymbol{x}(i); \boldsymbol{\theta})$ – результат вычисления нейронной сети от входа $\boldsymbol{x}(i)$ и параметров (весов) $\boldsymbol{\theta}$

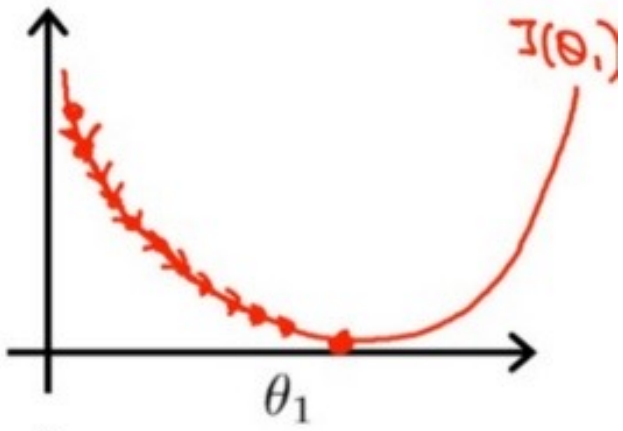
Обновление на k-ой итерации стохастического градиентного спуска (СГС)
Require: скорость обучения ϵ_k
Require: Начальные значения параметров $\boldsymbol{\theta}$
while критерий остановки не выполнен do
Выбрать из обучающего набора мини-батч m примеров $\{\boldsymbol{x}(1), \dots, \boldsymbol{x}(m)\}$ и соответствующие им метки y(i).
Вычислить оценку градиента: $g \leftarrow + (1/\mathrm{m}) \nabla_{\boldsymbol{\theta}} \Sigma_{\mathrm{i}} L(f(\boldsymbol{x}(i); \boldsymbol{\theta}), \boldsymbol{y}(i))$.
Применить обновление:$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon g$.
end while

Скорость обучения

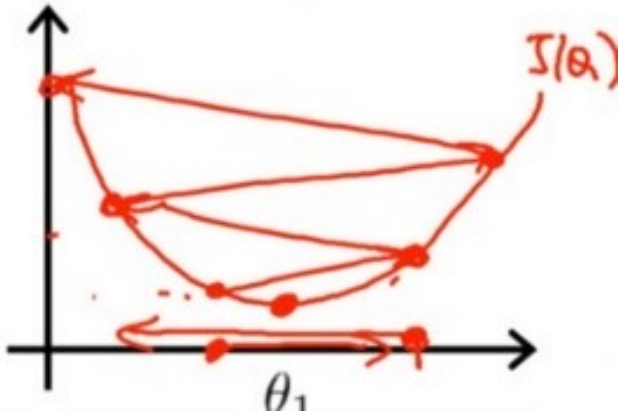
- Параметры (веса) сети меняются при обучении
- Гиперпараметры – нет. Управляем обучением

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

Если α слишком маленькая, то градиентный спуск может быть слишком медленным.



Если α слишком большая, то градиентный спуск может не попасть в точку минимума. Кроме того, сходимость может быть не достигнута.



Стохастический градиентный спуск

- Основной параметр алгоритма СГС – скорость обучения ϵ
- На практике же необходимо постепенно уменьшать скорость обучения со временем
- $\epsilon_k = (1-\alpha)\,\epsilon_k + \alpha\epsilon_k$, где $\alpha = k/\tau$. После τ-й итерации ε остается постоянным.
- Если скорость изменяется линейно, то нужно задать параметры $\varepsilon_0, \varepsilon_\tau$ и τ.

Импульсный метод

Стохастический градиентный спуск (СГС) с учетом импульса

Require: скорость обучения ε, параметр импульса α

Require: начальные значения параметров θ, начальная скорость v

while критерий остановки не выполнен do

Выбрать из обучающего набора мини-пакет m примеров $\{x(1), \ldots, x(m)\}$ и соответствующие им метки $y(i)$.

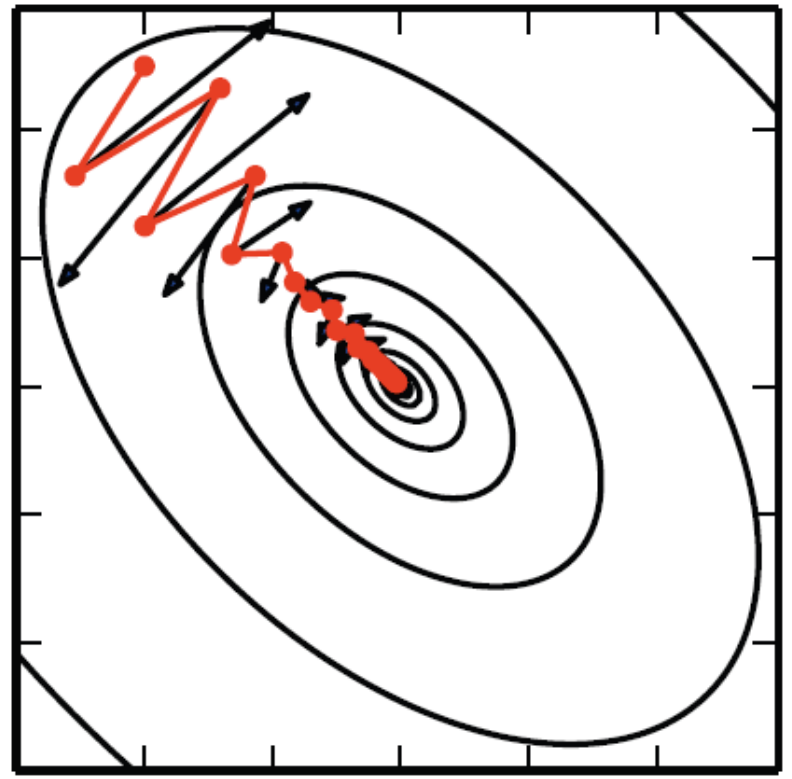
Вычислить оценку градиента:

$g \leftarrow (1/\mathrm{m})\, \nabla_\theta \Sigma_\mathrm{i}\, L(f(x(i); \theta), y(i))$.

Вычислить обновление скорости: $v \leftarrow \alpha v - \varepsilon g$.

Применить обновление: $\theta \leftarrow \theta + v$.

end while



Импульсный алгоритм можно рассматривать как имитацию движения частицы, подчиняющейся динамике Ньютона.

Adagrad

Алгоритм AdaGrad по отдельности адаптирует скорости обучения всех параметров модели.
Для параметров, по которым частная производная функции потерь наибольшая, скорость обучения уменьшается быстро, а если частная производная мала, то и скорость обучения уменьшается медленнее.
В итоге больший прогресс получается в направлениях пространства параметров со сравнительно пологими склонами

Алгоритм AdaGrad
Require: глобальная скорость обучения ε Require: начальные значения параметров $\boldsymbol{\theta}$
Require: небольшая константа δ, например $10{-}7$, для обеспечения численной устойчивости.
Инициализировать переменную для агрегирования градиента $\boldsymbol{r}{=}\boldsymbol{0}$
while критерий остановки не выполнен do
Выбрать из обучающего набора мини-пакет m примеров {x(1), ..., x(m)} и
соответствующие им метки y(i).
Вычислить градиент: $\boldsymbol{g} \leftarrow (1/\mathrm{m})\, \nabla_{\boldsymbol{\theta}} \Sigma_i L(f(\boldsymbol{x}(i); \boldsymbol{\theta}), \boldsymbol{y}(i))$.

Агрегировать квадраты градиента: $\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{g} \odot \boldsymbol{g}$.
Вычислить обновление: $\Delta\boldsymbol{\theta} \leftarrow -\epsilon \,/\, (\,\delta + \sqrt{r}\,) \odot \boldsymbol{g}$
Применить обновление: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$. end while

RMSProp

AdaGrad уменьшает скорость обучения, принимая во внимание всю историю квадрата градиента, и может случиться так, что скорость станет слишком малой еще до достижения такой выпуклой структуры.
В алгоритме RMSProp используется экспоненциально затухающее среднее, т. е. далекое прошлое отбрасывается

Require: глобальная скорость обучения ε, скорость затухания ρ Require: начальные значения параметров $\boldsymbol{\theta}$
Require: небольшая константа δ, например 10^{-6}, для стабилизации деления на малые числа
Инициализировать переменную для агрегирования градиента $\boldsymbol{r} = 0$
while критерий остановки не выполнен **do**
Выбрать из обучающего набора мини-пакет m примеров $\{\boldsymbol{x}(1), \ldots, \boldsymbol{x}(m)\}$ и соответствующие им метки $\boldsymbol{y}i$.
Вычислить градиент: $\boldsymbol{g} \leftarrow (1/\mathrm{m})\, \nabla_{\boldsymbol{\theta}}\, \Sigma_i\, L(f(\boldsymbol{x}(i); \boldsymbol{\theta}), \boldsymbol{y}(i))$.

Агрегировать квадраты градиента: $\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1 - \rho)\, \boldsymbol{g} \odot \boldsymbol{g}$.
Вычислить обновление параметров: $\Delta\boldsymbol{\theta} \leftarrow - \epsilon / \sqrt{(\delta + r)} \odot \boldsymbol{g}$
Применить обновление: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$.
end while

Adam

«Adam» – сокращение от «adaptive moments» (адаптивные моменты). Его правильнее всего рассматривать как комбинацию RMSProp и импульсного метода

Require: величина шага ε (по умолчанию 0.001).
Require: коэффициенты экспоненциального затухания для оценок моментов ρ и ρ , принадлежащие диапазону [0, 1) (по умолчанию 0.9 и 0.999 соответственно).
Require: небольшая константа δ для обеспечения численной устойчивости (по умолчанию 10^{-8}).
Require: начальные значения параметров $\boldsymbol{\theta}$.
Инициализировать переменные для первого и второго моментов $\boldsymbol{s} = \mathbf{0}, \boldsymbol{r} = \mathbf{0}$
Инициализировать шаг по времени $t = 0$

while критерий остановки не выполнен do
Выбрать из обучающего набора мини-пакет m примеров $\{\boldsymbol{x}(1), \dots, \boldsymbol{x}(m)\}$ и соответствующие им метки y_i.
Вычислить градиент: $\boldsymbol{g} \leftarrow (1/m)\nabla\boldsymbol{\theta}\Sigma i\, L(f(\boldsymbol{x}(i); \boldsymbol{\theta}), \boldsymbol{y}(i))$.
$t \leftarrow t+1$
Обновить смещенную оценку первого момента: $\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1)\boldsymbol{g}$
Обновить смещенную оценку второго момента: $\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2)\boldsymbol{g} \odot \boldsymbol{g}$
Скорректировать смещение первого момента: $s \leftarrow s/(1 - \rho_t)$
Скорректировать смещение второго момента: $r \leftarrow r/(1 - \rho_t)$
Вычислить обновление: $\delta\theta = -\epsilon\, s/\sqrt{(\delta+r)}$
Применить обновление: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$.
end while

Другие оптимизаторы