Нейронные сети и их практическое применение.

Лекция 5. Обзор алгоритмов обучения.

Дмитрий Буряк
к.ф.-м.н
dyb04@yandex.ru

Градиентные методы обучения

Минимизация целевой функции :

d - желаемый выход

y - реальный выход

$$E(w) = \frac{1}{2}\sum_j (y_j - d_j)^2$$

Метод обучения - градиентный спуск.

$$E(w+p) = E(w) + [g(w)]^T p + \frac{1}{2} p^T H(w) p + \ldots$$

$$g(w) = \nabla E = [\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_n}]$$

H(w)-матрица вторых производных;

w - минимум *E(w),* если *g(w)=0, H(w)* – положительно определена

Обозначим w_k -решение, полученное на *k*-ом шаге.

Цель: подобрать *p* и η так, чтобы, для очередной точки w_{k+1} выполнялось условие:

$$w_{k+1} = w_k + \eta_k p_k$$

$$E(w_{k+1}) < E(w_k)$$

Схема универсального алгоритма обучения

1. Проверка оптимальности текущего решения w_k .

2. Определение вектора направления оптимизации p_k для точки w_k .

3. Выбор шага η_k в направлении p_k , при котором выполняется условие

$$E(w_{k+1})<E(w_k)$$

4. Определение нового решения $w_{k+1} = w_k + \eta_k p_k$

а также соответствующих ему значений функции ошибки, градиента, матрицы вторых производных.

Переход на 1.

Алгоритм наискорейшего спуска

Линейное приближение функции *E(w)*.
Для выполнения условия $E(w_{k+1})<E(w_k)$, надо $g(w_k)^T p < 0$

$$p_k = -g(w_k)$$

Достоинства:
- небольшая вычислительная сложность;
- невысокие требования к памяти.

Недостатки:
- не учитывает информацию о кривизне функции;
- замедление в случае маленького градиента;
- медленная сходимость.

Наискорейший спуск с моментом

$$\Delta w_k = \eta_k p_k + \alpha(w_k - w_{k-1})$$

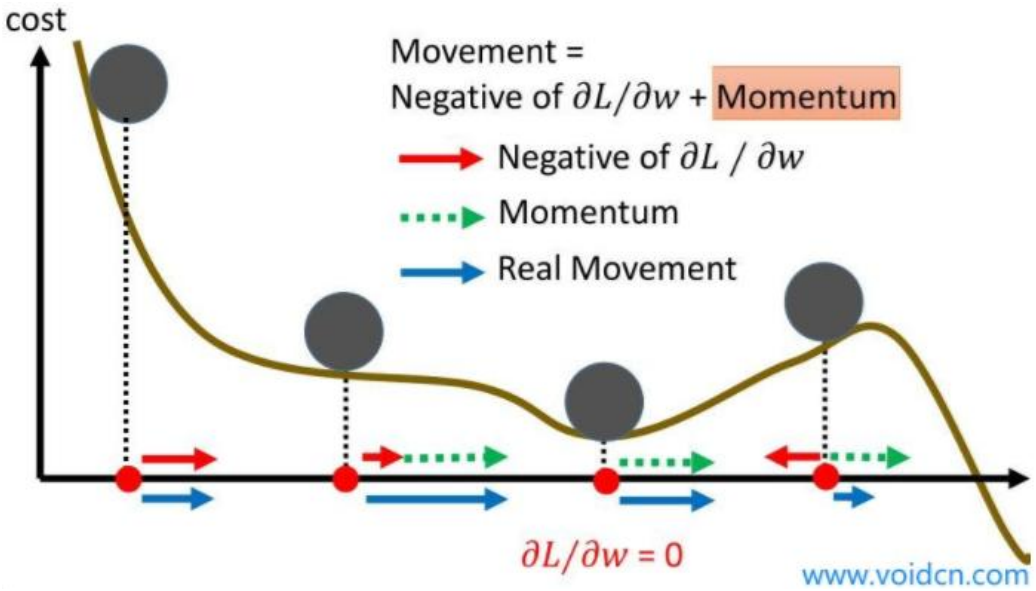
α - коэффициент момента в интервале [0,1]

На плоских участках: $\Delta w_k = \eta p_k + \alpha \Delta w_k$

$$\Delta w_k = \frac{\eta}{1-\alpha} p_k$$

Достоинство: ускорение сходимости на плоских участках, позволяет избежать локальных экстремумов.

Использование момента



Сделаем рекуррентные подстановки.
Введем ряд по t.
Решим разностное уравнение.

$$\Delta w_{ji}(n) = \eta \sum_{t=0}^{n} \alpha^{n-t} \delta_j(t) y_i(t).$$

$$\delta_j(n) y_i(n) = -\partial \mathbf{E}(n)/\partial w_{ji}(n)$$

$$\Delta w_{ji}(n) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial \mathbf{E}(t)}{\partial w_{ji}(t)}$$

1. Для сходимости ряда необходимо $0 \leq |\alpha| < 1$
2. Ускоряет спуск, если знак градиента не меняется
3. Замедляет спуск, если знак градиента изменяется – стабилизирующий эффект.

Метод переменной метрики

Квадратичное приближение *E(w)*

$$E(w_k + p_k) = E(w_k) + [g(w_k)]^T p_k + \frac{1}{2} p_k^T H(w_k) p_k$$

$$\frac{dE(w_k + p_k)}{dp_k} = 0 \quad g(w_k) + H(w_k) p_k = 0$$

$$p_k = -[H(w_k)]^{-1} g(w_k)$$

Используют приближение *H(w)*: $V_k = V_{k-1} + F(w_k, w_{k-1}, g(w_k), g(w_{k-1}))$

где: V_k - приближение $[H(w_k)]^{-1}$

Достоинство: быстрая сходимость.

Недостатки:

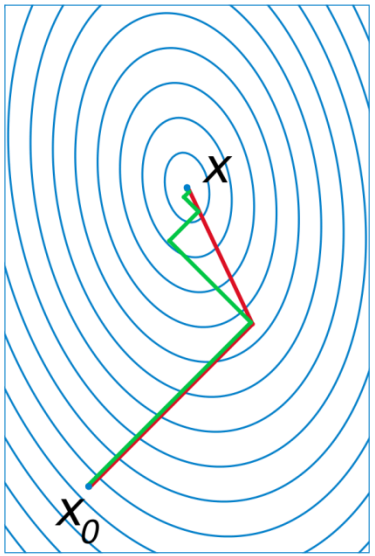
- вычислительная сложность;
- требования к памяти.

Алгоритм сопряженных градиентов

Направление поиска p_k выбирается таким образом, чтобы оно было ортогональным и сопряженным ко всем предыдущим направлениям p_0, p_1,…, p_{k-1}.

$$p_k = -g(w_k) + \beta_{k-1} p_{k-1}$$

$$\beta_{k-1} = \frac{g(w_k)^T (g(w_k) - g(w_{k-1}))}{g(w_{k-1})^T g(w_{k-1})} \quad \text{или} \quad \beta_{k-1} = \frac{g(w_k)^T (g(w_k) - g(w_{k-1}))}{-p_{k-1} g(w_{k-1})}$$

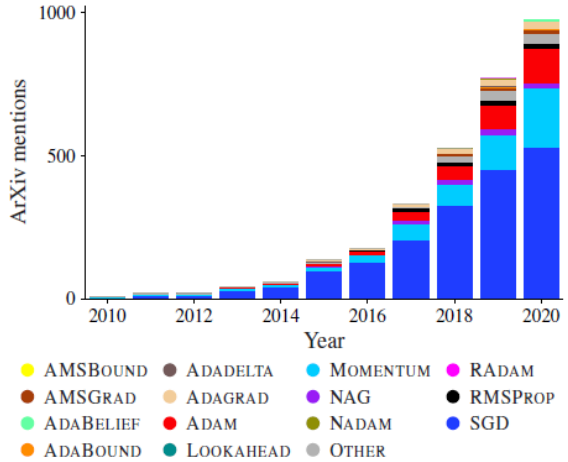


Достоинство:
- быстрее наискорейшего спуска;
- невысокие требования к памяти.

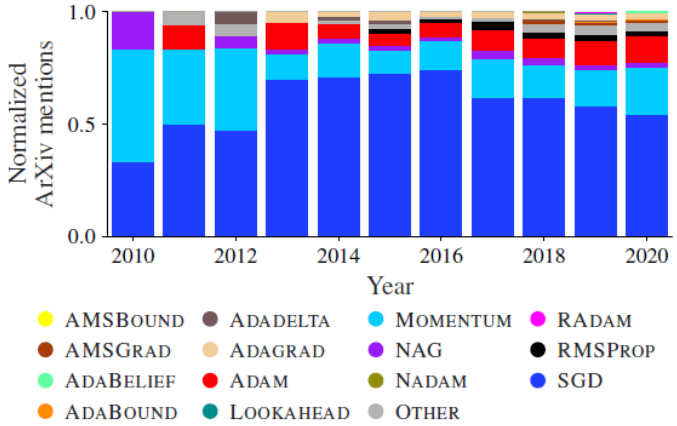
Недостаток: менее эффективен метода переменной метрики.

Выбор алгоритма обучения

- ❑ Существует более 100 алгоритмов оптимизации для НС*.
- ❑ Эмпирическое сравнение эффективности различных алгоритмов.
- ❑ Не существует исследований, которые сравнивают все (почти все) алгоритмы.



Количество упоминаний оптимизаторов в статьях на сайте arxiv.org*



Относительное число упоминаний оптимизаторов в статьях на сайте arxiv.org*

** https://arxiv.org/abs/2007.01547*

Стохастический градиентный спуск (SGD)

- Основной алгоритм для глубокого обучения.
- Базируется на принципах градиентного спуска.
- Вычисление градиента по каждому примеру вычислительно затратно → выбор случайных подмножест из обучающей выборки.
- SGD с моментом

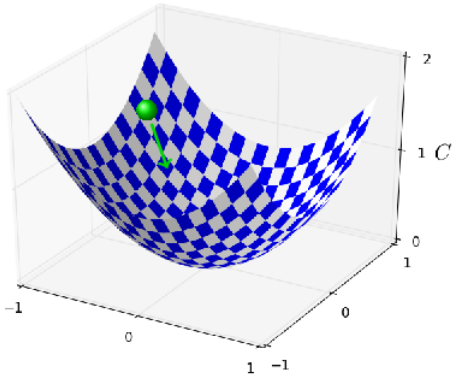
$$w_k \to w'_k = w_k - \eta \frac{\partial C}{\partial w_k}$$

$$b_l \to b'_l = b_l - \eta \frac{\partial C}{\partial b_l}.$$

$$\nabla C \approx \frac{1}{m} \sum_{j=1}^{m} \nabla C_{X_j}$$

$$w_k \to w'_k = w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k}$$

$$b_l \to b'_l = b_l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_l},$$



Алгоритм обучения Adam.

- Момент + учет частоты обновления веса.

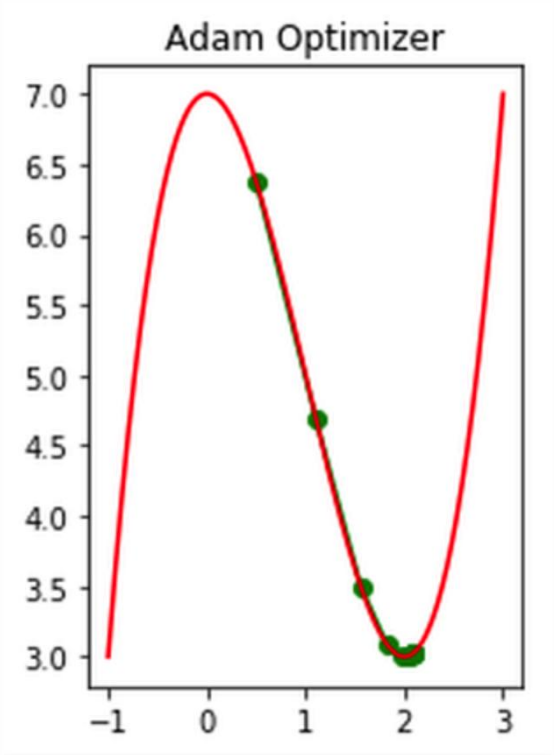
$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \ \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t$$

$$\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$$



Оптимизаторы в Pytorch.

Пакет torch.optim -- коллекция различных оптимизаторов градиентного спуска, от простейшего SGD до современных:

- optim.ASGD,
- optim.Adadelta,
- optim.Adagrad,
- optim.RMSprop,
- optim.Adam

```
torch.manual_seed(42)
a = torch.randn(1, requires_grad=True, dtype=torch.float)
b = torch.randn(1, requires_grad=True, dtype=torch.float)
print("Initial values:",a, b)
lr = 1e-1
n_epochs = 1000
# Defines a SGD optimizer to update the parameters
optimizer = optim.SGD([a, b], lr=lr)

for epoch in range(n_epochs):
    yhat = a + b * x_train
    error = y_train - yhat
    loss = (error ** 2).mean()

    loss.backward()

    #     a -= lr * a.grad
    #     b -= lr * b.grad
   optimizer.step()
   optimizer.zero_grad()
```

Методы оптимизации 2-го порядка

$$C(w+\Delta w) \approx C(w) + \nabla C \cdot \Delta w + \frac{1}{2}\Delta w^T H \Delta w$$

$$\Delta w = -H^{-1}\nabla C$$

- Оценка Гессиана ~ скорость изменения градиента.
- Требует много памяти: 1000000 весов → 10^{12} элементов Гессиана → >3Tb RAM .
- Методы частичной апроксимации Гессиана (L-BFGS).
- Аппроксимация требует использования всех примеров из обучающей выборки.

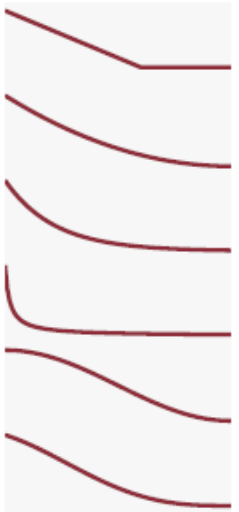
Изменение скорости обучения

- Ступенчатое уменьшение
  - в ***к*** раз каждые ***n*** итераций обучения;
  - уменьшение коэффициента обучения, если ошибка на подтверждающей выборке перестала уменьшаться.

- Экспоненциальное $\eta = \alpha_0 e^{-\beta t}$

- Гиперболическое $\eta = \frac{\alpha_0}{1+\beta t}$

Схемы изменения скорости обучения

Name		Ref.	Illustration
Constant			
Step Decay	constant factor		
	multi-step		
Smooth Decay	linear decay	e.g. (Goodfellow et al., 2016)	
	polynomial decay		
	exponential decay		
	inverse time decay	e.g. (Bottou, 2012)	
	cosine decay	(Loshchilov & Hutter, 2017)	
	linear cosine decay	(Bello et al., 2017)	

** https://arxiv.org/abs/2007.01547*

Подбор на основе направленной минимизации

$$E(w) \to P_2(\eta) = a_2\eta^2 + a_1\eta + a_0$$

$$E(w) \to P_2(\eta) = a_3\eta^3 + a_2\eta^2 + a_1\eta + a_0$$

$$w_1 = w + \eta_1 p_k$$

$$w_2 = w + \eta_2 p_k$$

$$w_3 = w + \eta_3 p_k$$

$$\begin{cases} P_2(\eta_1) = E(w_1) \\ P_2(\eta_2) = E(w_2) \\ P_2(\eta_3) = E(w_3) \end{cases}$$

$$\eta = \frac{-a_1}{2a_2}$$

$$\eta = \frac{-a_2 + \sqrt{a_1{}^2 - 3a_2a_1}}{3a_3}$$

Стохастический метод обучения. Общая схема

1. Выполнить начальную инициализацию весов.

2. Вычислить выход для примера из обучающей выборки.

3. Вычислить ошибку:

4. Выбрать случайным образом вес и изменить его значение на случайную величину.
Если проведенное изменение уменьшает целевую функцию (ошибку), то сохранить его, иначе вернуться к прежнему значению веса.

5. Повторять шаги 2, 3, 4, пока сеть не обучится.

Алгоритм имитации отжига

1. Выполнить начальную инициализацию весов. Задать начальную температуру $T=T_{max}$. Вычислить значение ошибки.

2. Пока $T>0$ повторить L раз действия:
2.1. Выполнить случайную коррекцию весов $w'=w+\Delta w$.
2.2. Вычислить разницу целевых функций $c=E(w')-E(w)$.
2.3. Если $c<0$, то $w=w'$
иначе принять $w=w'$ с вероятностью $P(c) = e^{-\frac{c}{kT}}$

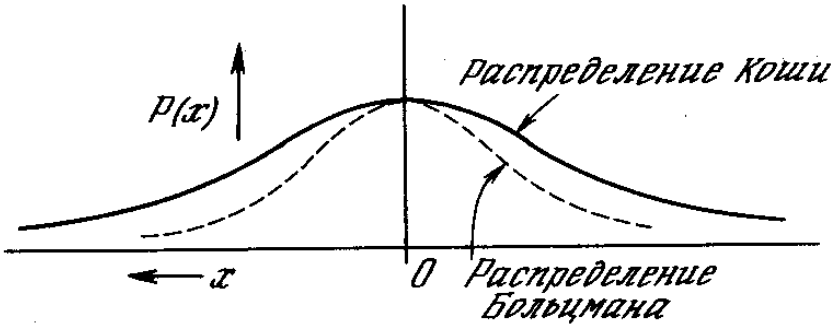
3. Уменьшить температуру T.
4. Повторять 2, 3, 4 пока $T>0$ или значение ошибки больше порога.

Приращение весов Δw: $P(\Delta w) = e^{-\frac{\Delta w^2}{T^2}}$

Алгоритм имитации отжига (Коши)

Скорость сходимости метода Больцмана: $T(t) = \frac{T_{\max}}{\log(1+t)}$

Распределение Коши: $P(x) = \frac{T(t)}{T(t)^2 + x^2}$



Скорость сходимости метода Коши: $T(t) = \frac{T_{\max}}{1+t}$

Комбинированный метод обучения

Объединение традиционного обратного распространения с обучением Коши.

$$\Delta w_k = \beta[\eta_k p_k + \alpha(w_k - w_{k-1})] + (1-\beta)\Delta w^c$$

Вопросы

1. Основные достоинства и недостатки алгоритма наискорейшего спуска?
2. Что препятствует применению алгоритмов, основанных на методах оптимизации второго порядка, для обучения современных нейронных сетей?
3. Какое основное преимущество стохастических алгоритмов обучения перед градиентными методами?