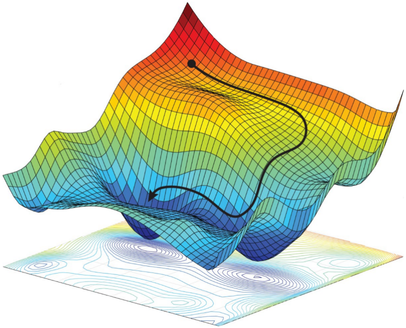
# Особенности настройки глубоких нейросетей

Виктор Китов
victorkitov.github.io



# Содержание

# Метод стохастического градиентного спуска (SGD)

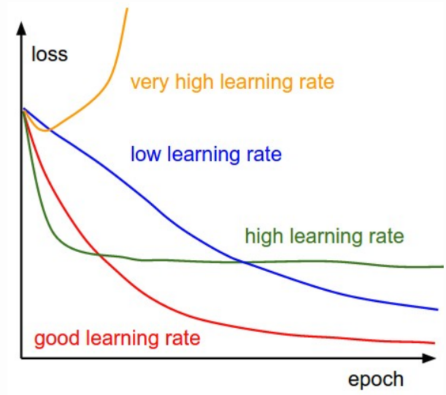
- Обозначим $\mathcal{L}(\widehat{y}, y)$ - функция потерь, $W = \#$[весов], $\varepsilon_t$ - шаг оптимизации на шаге $t$.
- Можем оптимизировать, используя стохастический градиентный спуск:

```
Инициализируем случано w, t = 0

повторять до сходимости:
    сэмплируем случайный объект (x_n, y_n)
    w := w − ε_t ∇_w L(w, x_n, y_n)
   t := t + 1
```

- Эффективнее сэмплировать группу объектов ("минибатч")
- Сходимость: $\|L(w_{t+1}) - L(w_t)\| < threshold$, $t > threshold$, $\|w_{t+1} - w_t\| < threshold$
- Нормализация признаков ускоряет сходимость.

# Выбор шага оптимизации важен для сходимости




- На практике часто берут $\varepsilon = const$ и уменьшают при выходе на стабильные потери.
- Формально: сходимость при достаточно медленном $\varepsilon_t \to 0$.

# SGD с инерцией (momentum)

```
Инициализируем случано w , t = 0, Δw_0 = 0

повторять до сходимости:
    сэмплируем случайный объект (x_n, y_n)
    Δw_{t+1} := αΔw_t − η∇_w L(w, x_n, y_n)
    w := w + Δw_{t+1}
    t := t + 1
```

- SGD с инерцией использует сглаженную по всем наблюдениям более точную оценку градиента
    - можно использовать выше скорость сходимости!

- Инерция Нестерова: "заглядывание вперед" при расчете градиента: $\nabla_w\mathcal{L}(w+\alpha\Delta w_t, x_n, y_n)$

# Оценка градиента

- Вычисление $\nabla\mathcal{L}(w)$ через разностную аппроксимацию ($\varepsilon_i = [0, ...0, \varepsilon, 0, ...0]$)

$$\frac{\partial \mathcal{L}}{\partial w_i} = \frac{\mathcal{L}(w+\varepsilon_i) - \mathcal{L}(w)}{\varepsilon} + O(\varepsilon) \tag{1}$$

или более точная оценка

$$\frac{\partial \mathcal{L}}{\partial w_i} = \frac{\mathcal{L}(w+\varepsilon_i) - \mathcal{L}(w-\varepsilon_i)}{2\varepsilon} + O(\varepsilon^2) \tag{2}$$

имеет вычислительную сложность $O(K^2)$, $K$ - #весов
  - нужно посчитать $K$ производных
  - сложность вычисления каждой: $O(K)$

Алгоритм обратного распространения ошибки (backpropagation) требует только $O(K)$ для расчета всех производных.

# Пример вычисления градиента[1]

- Рассмотрим бинарную классификацию $y \in \{+1, -1\}$, сеть предсказывает $p(y=+1|x)$:

$$p(y=+1|x) = \frac{1}{1+e^{-w^T x}}$$

- Качество - вероятность истинного класса:

$$S = p(y|x) = \mathbb{I}[y=+1]p(y=+1|x) + \mathbb{I}[y=-1](1-p(y=+1|x)$$

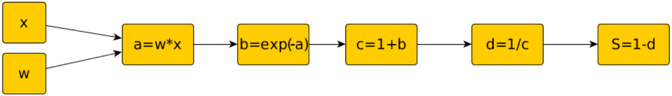
- Предположим $y=-1$, и мы обновляем $w$, чтобы $1-p(y=+1|x)$ была выше:

$$w := w + \varepsilon \frac{\partial S}{\partial w} = w + \varepsilon \frac{\partial [1-p(y=+1|x)]}{\partial w}$$

[1] Пример BackProp для векторного случая.

# Алгоритм обратного распространения ошибки

$$\frac{\partial[1-p(y=+1|x)]}{\partial w}=\frac{\partial}{\partial w}\left[1-\frac{1}{1+e^{-w^{T}x}}\right]-?$$




Рассчитаем все промежуточные активации слева-направо за $O(K)$, а производные - справа-налево за $O(K)$:

$$\frac{\partial S}{\partial d}=-1,\ \frac{\partial S}{\partial c}=\frac{\partial S(d(c))}{\partial c}=\frac{\partial S}{\partial d}\frac{\partial d}{\partial c}=\frac{\partial S}{\partial d}\left(-\frac{1}{c^{2}}\right);\ \frac{\partial S}{\partial b}=$$

$$\frac{\partial S(c(b))}{\partial b}=\frac{\partial S}{\partial c}\frac{\partial c}{\partial b}=\frac{\partial S}{\partial b}\cdot 1;\ \frac{\partial S}{\partial a}=\frac{\partial S(b(a))}{\partial a}=\frac{\partial S}{\partial b}\frac{\partial b}{\partial a}=-\frac{\partial S}{\partial b}e^{-a}$$

$$\frac{\partial S}{\partial w}=\frac{\partial S(a(w))}{\partial w}=\frac{\partial S}{\partial a}\frac{\partial a}{\partial w}=\frac{\partial S}{\partial a}x$$

# Особенности оптимизации нейросетей

- Зависимость $\widehat{y}(x)$ в общем случае невыпукла.
- $\mathcal{L}(\widehat{y}, y)$ - невыпукла => много локальных минимумов.
- На найденный минимум влияют:
    - начальное приближение
    - объекты минибатчей
    - метод обучения и динамика $\varepsilon_t$
- Можно настраивать разными способами, а потом
    - выбрать наилучшее решение по валидации
    - усреднить несколько решений

# Содержание

# Особенности оптимизации нейросетей

- Зависимость $\widehat{y}(x)$ в общем случае невыпукла.
- $\mathcal{L}(\widehat{y}, y)$ - невыпукла => много локальных минимумов.
- На найденный минимум влияют:
    - начальное приближение
    - объекты минибатчей
    - метод обучения и динамика $\varepsilon_t$
- Можно настраивать разными способами, а потом
    - выбрать наилучшее решение по валидации
    - усреднить несколько решений (ансамбль)

# Базовые градиентные методы

- Batch gradient descent: градиентный спуск по всем объектам выборки

$$w_{t+1} := w_t - \eta \nabla_w L(w; X, Y)$$

# Базовые градиентные методы

- Batch gradient descent: градиентный спуск по всем объектам выборки

$$w_{t+1} := w_t - \eta \nabla_w L(w; X, Y)$$

  - медленно, не применим для динамических данных

- Stochastic gradient descent (SGD): стохастический спуск с сэмплированием по 1 объекту

$$w_{t+1} := w_t - \eta \nabla L_w(w; x_i, y_i)$$

  - берём последовательные объекты, но перемешиваем выборку перед каждой эпохой.

## Базовые градиентные методы

- Batch gradient descent: градиентный спуск по всем объектам выборки

$$w_{t+1} := w_t - \eta \nabla_w L(w; X, Y)$$

  - медленно, не применим для динамических данных

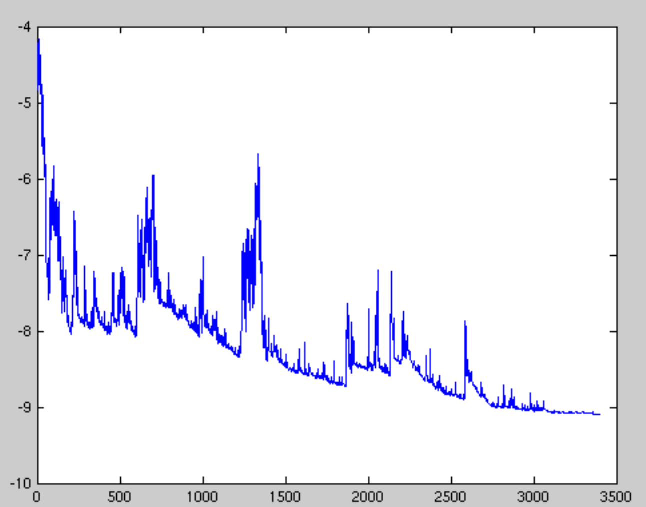
- Stochastic gradient descent (SGD): стохастический спуск с сэмплированием по 1 объекту

$$w_{t+1} := w_t - \eta \nabla L_w(w; x_i, y_i)$$

  - берём последовательные объекты, но перемешиваем выборку перед каждой эпохой.

- SGD быстрее сходится, но
  - неустойчивая оценка градиента
  - слабое использование параллелизации

# Пример сходимости SGD



# Базовые градиентные методы

- Minibatch gradient descent: стохастический спуск с сэмплированием по набору объектов

$$w_{t+1} := w_t - \eta \nabla_w L(w; x_{i+1:i+K}, y_{i+1:i+K})$$

# Базовые градиентные методы

- Minibatch gradient descent: стохастический спуск с сэмплированием по набору объектов

$$w_{t+1} := w_t - \eta \nabla_w L(w; x_{i+1:i+K}, y_{i+1:i+K})$$

  - точнее оценки градиента
  - и быстрее: параллелизация вычислений по минибатчу

# Базовые градиентные методы

- Minibatch gradient descent: стохастический спуск с сэмплированием по набору объектов

$$w_{t+1} := w_t - \eta \nabla_w L(w; x_{i+1:i+K}, y_{i+1:i+K})$$

    - точнее оценки градиента
    - и быстрее: параллелизация вычислений по минибатчу
- Сложности:
    - нужен выбор динамики убывания $\eta$
    - одинаковый шаг для разных весов
        - логичнее веса брать меньше, где ф-ция резко меняется и отвечающие редко встречающимся признакам.
    - застревание в локальных оптимумах и точках перегиба

# Модификации инерции и Нестерова

- SGD с инерцией (momentum), "мяч катится с горы":

$$
\begin{aligned}
v_t &:= \gamma v_{t-1} + \eta \nabla_w L(w_t) \\
w_{t+1} &:= w_t - v_t
\end{aligned}
$$

# Модификации инерции и Нестерова

- SGD с инерцией (momentum), "мяч катится с горы":

$$v_t := \gamma v_{t-1} + \eta \nabla_w L(w_t)$$
$$w_{t+1} := w_t - v_t$$

  - устойчивее градиент (усредняем по градиентам с прошлых итераций)
  - можем брать выше шаг обучения, ускорение

# Модификации инерции и Нестерова

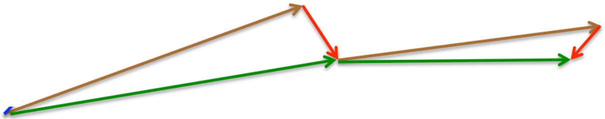
- SGD с инерцией (momentum), "мяч катится с горы":

$$v_t := \gamma v_{t-1} + \eta \nabla_w L(w_t)$$
$$w_{t+1} := w_t - v_t$$

  - устойчивее градиент (усредняем по градиентам с прошлых итераций)
  - можем брать выше шаг обучения, ускорение

- Nesterov Accelerated Gradient (Nesterov Momentum)

$$v_t := \gamma v_{t-1} + \eta \nabla_w L(w_t - \gamma v_{t-1})$$
$$w_{t+1} := w_t - v_t$$



# Модификации SGD

- Обозначим $g_t = \nabla_w L(w_t)$; $w, g_t \in \mathbb{R}^K$. Операции над векторами поэлементные.
- AdaGrad ($\varepsilon = 10^{-6}$)

$$G_t := G_t + \nabla_w L(w_t)^2$$
$$w_{t+1} := w_t - \frac{\eta}{\sqrt{G_t + \varepsilon}} \cdot \nabla_w L(w_t)$$

- RMSprop

$$G_t := \gamma G_{t-1} + (1-\gamma)\, \nabla_w L(w_t)^2$$
$$w_{t+1} := w_t - \frac{\eta}{\sqrt{G_t + \varepsilon}} \cdot \nabla_w L(w_t)$$

# Модификации SGD

- Adam=RMSprop+инерция
  $(\beta_1 = 0.9,\ \beta_2 = 0.999,\ \varepsilon = 10^{-8})$:

$$m_t := \beta_1 m_{t-1} + (1-\beta_1)\nabla_w L(w_t)$$
$$G_t := \beta_2 G_{t-1} + (1-\beta_2)\nabla_w L(w_t)^2$$
$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\widehat{G}_t = \frac{G_t}{1-\beta_2^t}$$
$$w_{t+1} := w_t - \frac{\eta}{\sqrt{\widehat{G}_t + \varepsilon}} \cdot \widehat{m}_t$$

- Nadam: Adam+Nesterov Accelerated Gradient.

# Модификации SGD

- AMSGrad: позволяет помнить историю без экспоненциального забывания.
    - при этом перенормировка $m_t, v_t$ не используется.

$$
\begin{aligned}
m_t &:= \beta_1 m_{t-1} + (1-\beta_1)\,\nabla_w L(w_t) \\
G_t &:= \beta_2 G_{t-1} + (1-\beta_2)\,\nabla_w L(w_t)^2 \\
\widehat{G}_t &= \max\left(\widehat{G}_{t-1}, G_t\right) \\
w_{t+1} &:= w_t - \frac{\eta}{\sqrt{\widehat{G}_t + \varepsilon}} \odot \widehat{m}_t
\end{aligned}
$$

# Дополнительные улучшения[2]

- Ранняя остановка (early stopping) - борьба с переобучением
- Добавление шума к градиенту - находим оптимум с большей окрестностью:

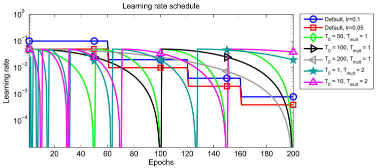
$$g_t := g_t + \mathcal{N}(0, \sigma_t^2)$$
$$\sigma_t^2 = \frac{\eta}{(1+t)^\gamma}$$

- Обучение по расписанию (curriculum learning)
    - сначала обучаемся на простых объектах, потом на сложных
- Параллелизация вычислений SGD.
- Ускорение обучения: batch normalization.

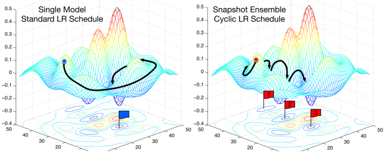
[2] Больше информации: https://ruder.io/deep-learning-optimization-2017/

# Дополнительные улучшения

- LR schedule - закон изменения $\eta$. Подобный закон, приводит к к оптимуму с большей окрестностью:




- Перед каждым увеличением $\eta$ получили некоторую модель=>можем модели комбинировать в композицию:



# Заключение

- Глубокие сети способны сами настраивать сложные признаки.
    - работают лучше, но нужны большие обучающие выборки
- Борьба с переобучением нейросетей:
    - расширение выборки (pretraining, data augmentation)
    - сокращение #нейронов/связей
    - ранняя остановка, $L_1/L_2$ регуляризация, DropOut.
- Функция потерь невыпукла, возможны локальные оптимумы.
- Идеи улучшений SGD: инерция, ускоренный градиент Нестерова, индивидуальные настраиваемые веса для каждого параметра.
- BatchNorm ускоряет и упрощает сходимость.