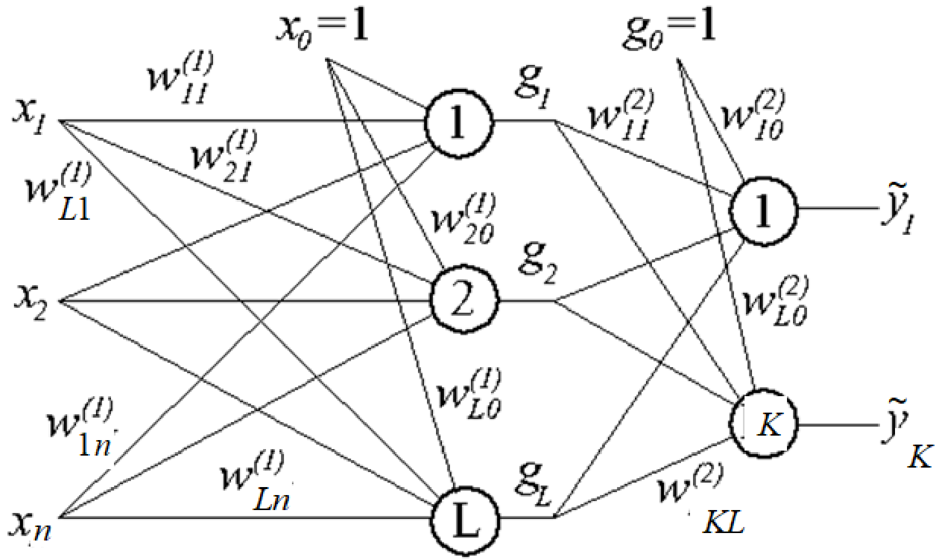
Обучение нейронных сетей

- Обучение одного нейрона
  - алгоритм Розенблатта
- Обучение нейронной сети
  - метод стохастического градиента (SGD)
- Метод обратного распространения ошибок
  - для быстрого вычисления градиента
- Модификации SGD

Алгоритм стохастического градиента:



1.Задать случайные веса $(w_{lj}^{(1)}(0), w_{ql}^{(2)}(0)),\ l=1,\ldots,L,\ q=1,\ldots,K$;

2.Повторять для $t=1,2,\ldots$:

3.Вычислить критерий $Q_\Sigma\left(w(t)\right)$;

4.Для каждого i-го объекта ($i=1,\ldots,N$): положим

$$w_{mk}^{(h)}(t+1) = w_{mk}^{(h)}(t) - \eta \frac{\partial Q(w(t))}{\partial w_{mk}^{(h)}(t)};$$

5.Продолжать 2)-4), пока либо критерий, либо веса не стабилизируются.

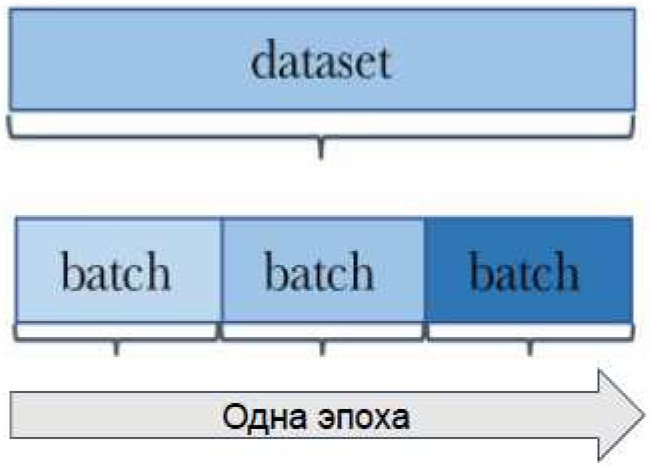
Недостатки алгоритма стохастического градиентного спуска (SGD) при обучении нейронной сети

- возможна медленная сходимость;
- «застревание» в локальном минимуме;
- зависимость от порядка объектов;
- проблема переобучения;

Решения:

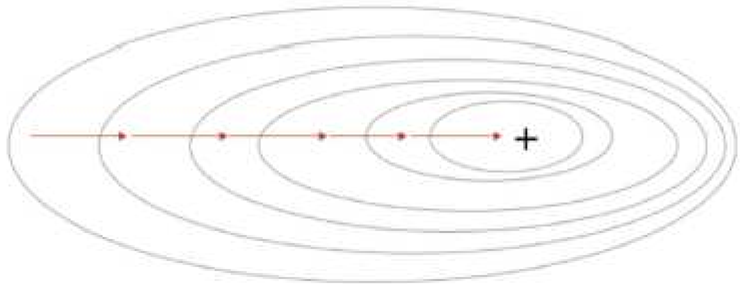
- Подготовка данных (пакеты, перемешивание, нормализация)
- Функции активации без «горизонтальных асимптот»
- Выключение нейронов (dropout)
- Модификации выбора направления спуска;
- Модификации выбора шага (темпа обучения);

Пакеты (batch)



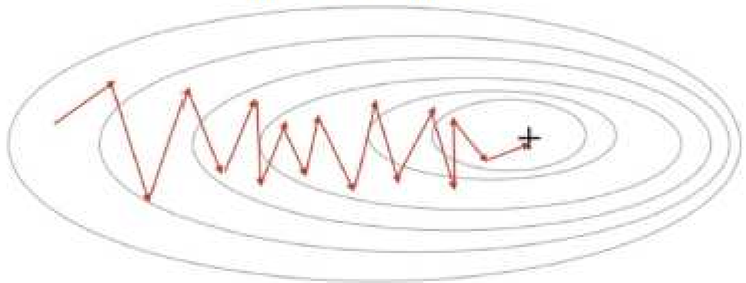
1. Перемешиваем датасет.
2. Разбиваем на батчи размера batch_size
3. Последовательно делаем шаг обучения на каждом.

Обычный градиентный спуск



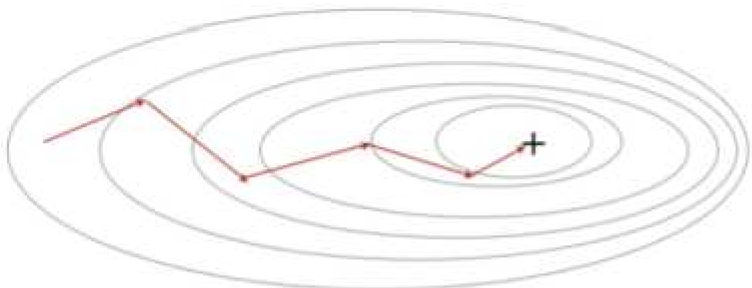
Стохастический градиентный спуск

(маленький батч)

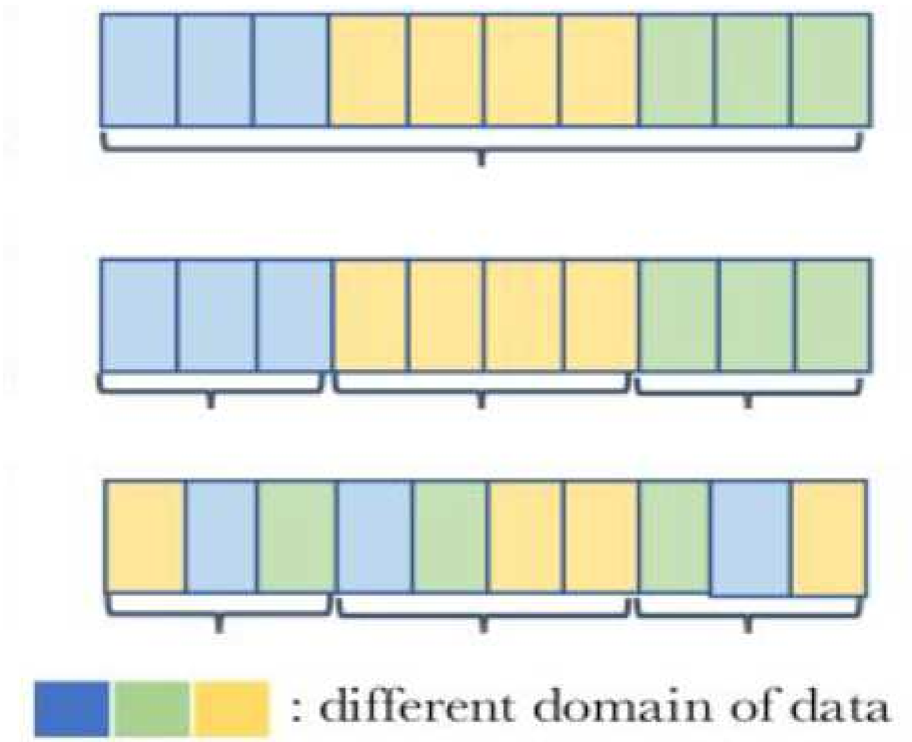


Стохастический градиентный спуск

(большой батч)

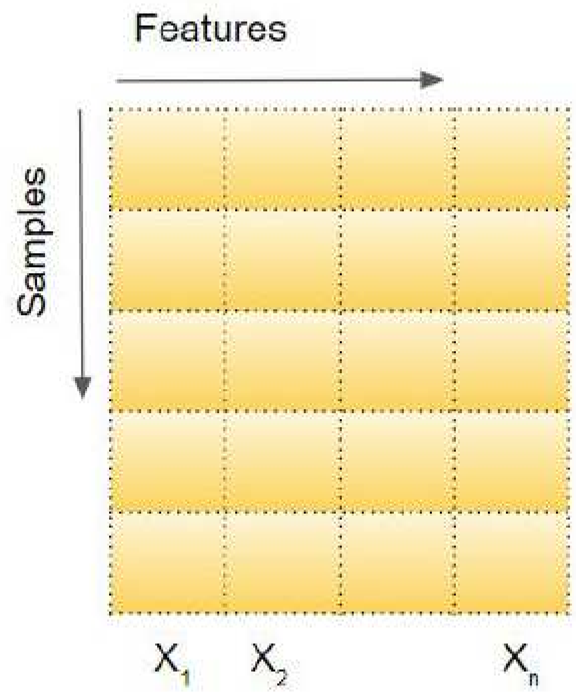


Перемешивание набора данных



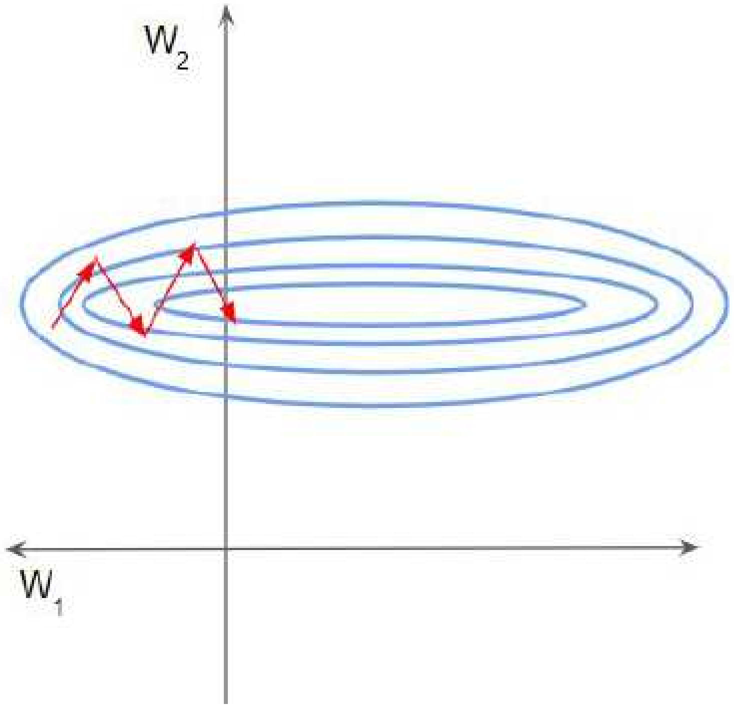
разные классы

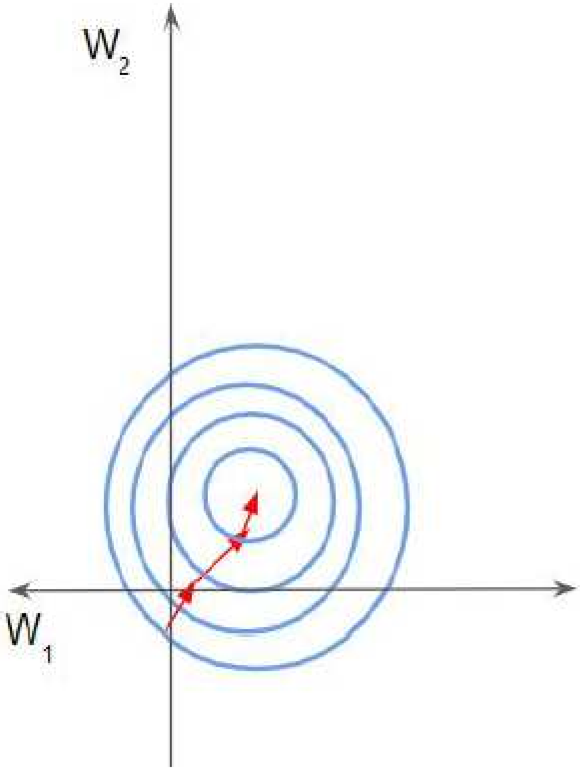
Стандартизация переменных на входе в нейронную сеть



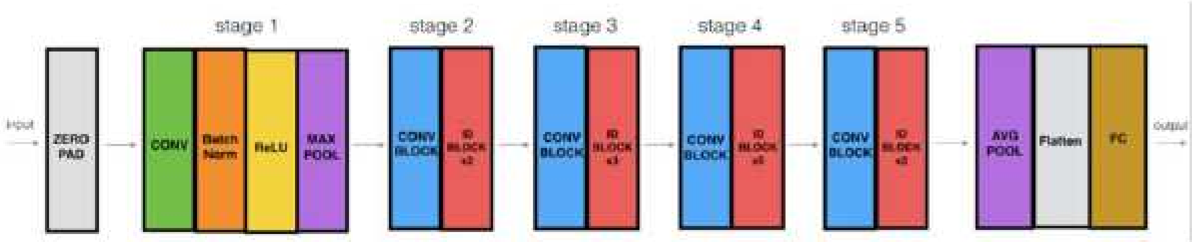
$$X_i = \frac{X_i - Mean_i}{StdDev_i}$$

Стандартизация входных переменных приводит к изменению функции потерь





Весов в нейронных сетях больше, чем примеров в обучающей выборке.



ResNet-50 - популярная ImageNet модель. Более 23 млн весов

ImageNet - около 23 миллионов картинок, 20000 классов.

L2-регуляризация

$$Q(w) = \frac{1}{2}\sum_{i=1}^{N}\sum_{q=1}^{K}\left(\tilde{y}_q(x^{(i)}) - y_q^{(i)}\right)^2 \to \min_w .$$

$$\tilde{Q}(w) = \frac{1}{2}\sum_{i=1}^{N}\sum_{q=1}^{K}\left(\tilde{y}_q(x^{(i)}) - y_q^{(i)}\right)^2 + \lambda\sum_{k,l,h}\left(w_{kl}^{(h)}\right)^2 \to \min_w$$

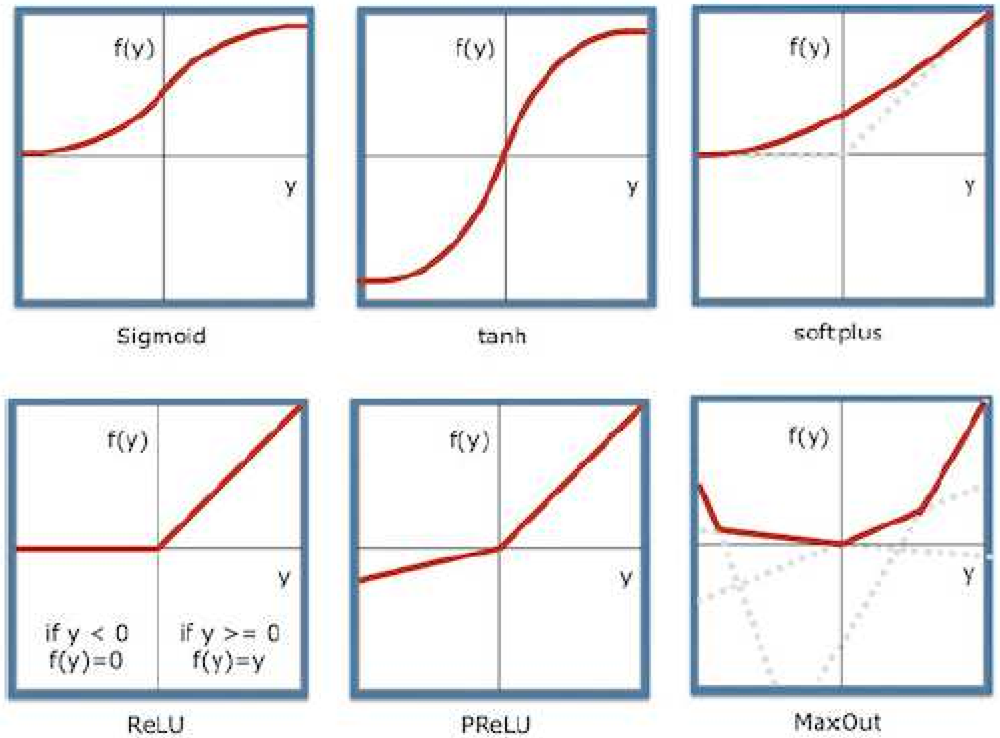
предотвращает увеличение весов до бесконечности

Выбор функции активации

Функции $\sigma(y) = \frac{1}{1+e^{-y}}$ и $\text{th}(y) = \frac{e^y - e^{-y}}{e^y + e^{-y}}$ могут приводить к затуханию градиентов или «параличу сети»

Функция положительной срезки (rectified linear unit)

$\text{ReLU}(y) = \max\{0, y\}$; $\quad \text{PReLU}(y) = \max\{0, y\} + \alpha \min\{0, y\}$



Начальное приближение весов

1. Выравнивание дисперсий выходов в разных слоях:
   $$w_j := \text{uniform}\left(-\frac{1}{\sqrt{h}}, \frac{1}{\sqrt{h}}\right)$$
2. Выравнивание дисперсий градиентов в разных слоях:
   $$w_j := \text{uniform}\left(-\frac{6}{\sqrt{h+m}}, \frac{6}{\sqrt{h+m}}\right),$$

где h, m — число нейронов в предыдущем и текущем слое

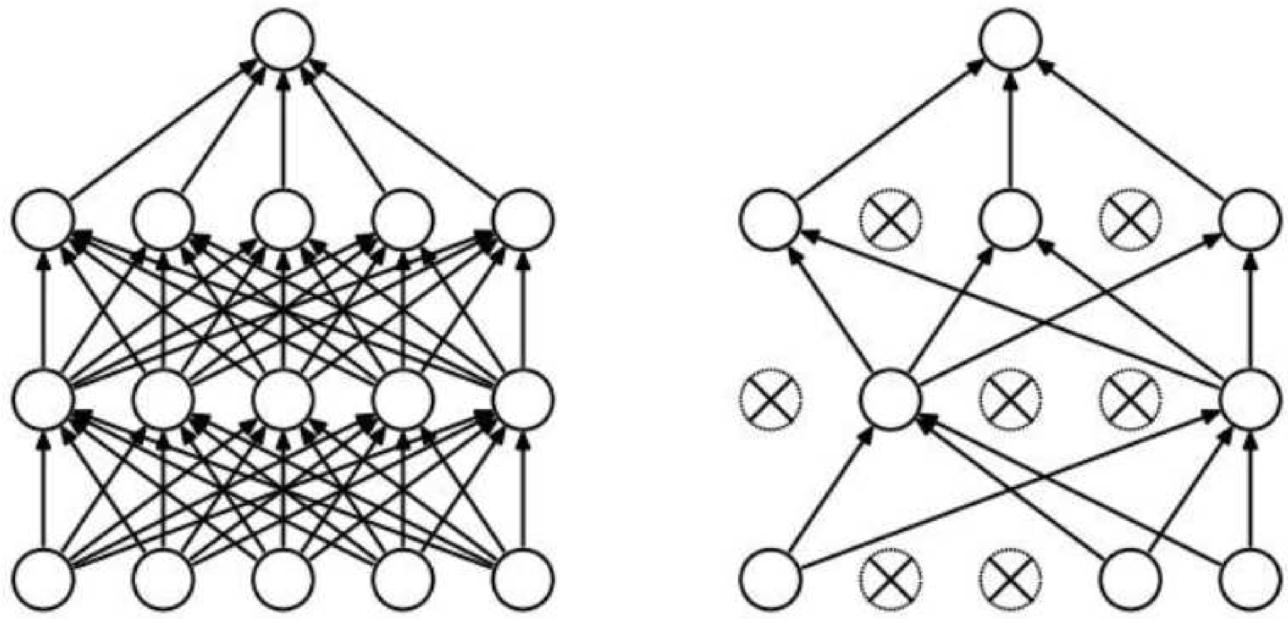
3. Послойное обучение нейронов как линейных моделей:
   - либо по случайной подвыборке $X' \subseteq X^\ell$;
   - либо по случайному подмножеству входов;
   - либо из различных случайных начальных приближений;

тем самым обеспечивается *различность* нейронов.

4. Инициализация весами предобученной модели
5. Инициализация случайным ортогональным базисом

Выключение нейронов

Dropout - слой, который зануляет случайные выходы от предыдущего слоя.



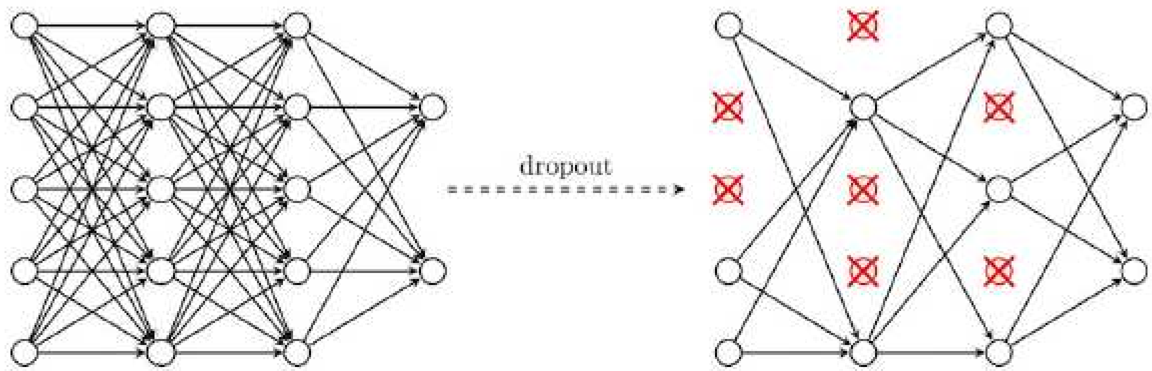
(a) Standard Neural Net (b) After applying dropout.

Этап обучения: делая градиентный шаг $\mathscr{L}_i(w) \to \min\limits_w$, отключаем h-ый нейрон ℓ-го слоя с вероятностью p_ℓ:

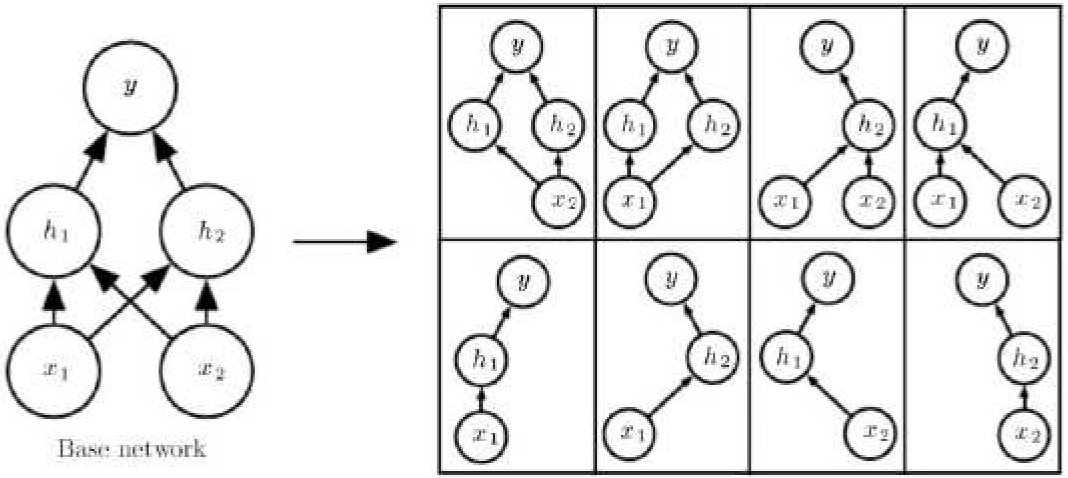
$$x_{ih}^{\ell+1} = \xi_h^\ell \, \sigma_h\Bigl(\sum_j w_{jh} x_{ij}^\ell\Bigr), \qquad \mathsf{P}(\xi_h^\ell = 0) = p_\ell$$

Этап применения: включаем все нейроны, но с поправкой:

$$x_{ih}^{\ell+1} = (1 - p_\ell)\sigma_h\Bigl(\sum_j w_{jh} x_{ij}^\ell\Bigr)$$



1. аппроксимируем простое голосование по 2^N сетям с общим набором из N весов, но с различной архитектурой связей
2. регуляризация: из всех сетей выбираем более устойчивую к утрате pN нейронов, моделируя надёжность мозга
3. сокращаем переобучение, заставляя разные части сети решать одну и ту же исходную задачу вместо того, чтобы подстраивать их под компенсацию ошибок друг друга



На практике чаще используют не Dropout, а *Inverted Dropout*.

Этап обучения:

$$x_{ih}^{\ell+1} = \tfrac{1}{1-p_\ell}\xi_h^\ell \sigma_h\Bigl(\sum_j w_{jh}x_{ij}^\ell\Bigr), \qquad \mathsf{P}(\xi_h^\ell = 0) = p_\ell$$

Этап применения не требует ни модификаций, ни знания p_ℓ:

$$x_{ih}^{\ell+1} = \sigma_h\Bigl(\sum_j w_{jh}x_{ij}^\ell\Bigr)$$

L_2-регуляризация предотвращает рост параметров на обучении:

$$\mathscr{L}_i(w) + \tfrac{\lambda}{2}\|w\|^2 \to \min_w$$

Градиентный шаг с Dropout и L_2-регуляризацией:

$$w := w(1-\eta\lambda) - \eta\,\tfrac{1}{1-p_\ell}\xi_h^\ell\,\mathscr{L}_i'(w)$$

Удаление связей
(Optimal brain damage)

Пусть w — локальный минимум $Q(w)$, тогда
$Q(w)$ можно аппроксимировать квадратичной формой:

$$Q(w+\delta) = Q(w) + \tfrac{1}{2}\delta^{\mathsf{T}} Q''(w)\delta + o(\|\delta\|^2),$$

где $Q''(w) = \Bigl(\frac{\partial^2 Q(w)}{\partial w_{jh}\partial w_{j'h'}}\Bigr)$ — гессиан, размера $\bigl(H(n+M+1)+M\bigr)^2$.

Эвристика. Пусть гессиан $Q''(w)$ диагонален, тогда

$$\delta^{\mathsf{T}} Q''(w)\delta = \sum_{j=0}^{n}\sum_{h=1}^{H} \delta_{jh}^2 \frac{\partial^2 Q(w)}{\partial w_{jh}^2} + \sum_{h=0}^{H}\sum_{m=0}^{M} \delta_{hm}^2 \frac{\partial^2 Q(w)}{\partial w_{hm}^2}.$$

Хотим обнулить вес: $w_{jh} + \delta_{jh} = 0$. Как изменится $Q(w)$?

Определение. *Значимость* (salience) веса w_{jh} — это изменение функционала $Q(w)$ при его обнулении: $S_{jh} = w_{jh}^2 \frac{\partial^2 Q(w)}{\partial w_{jh}^2}$.

Удаление связей

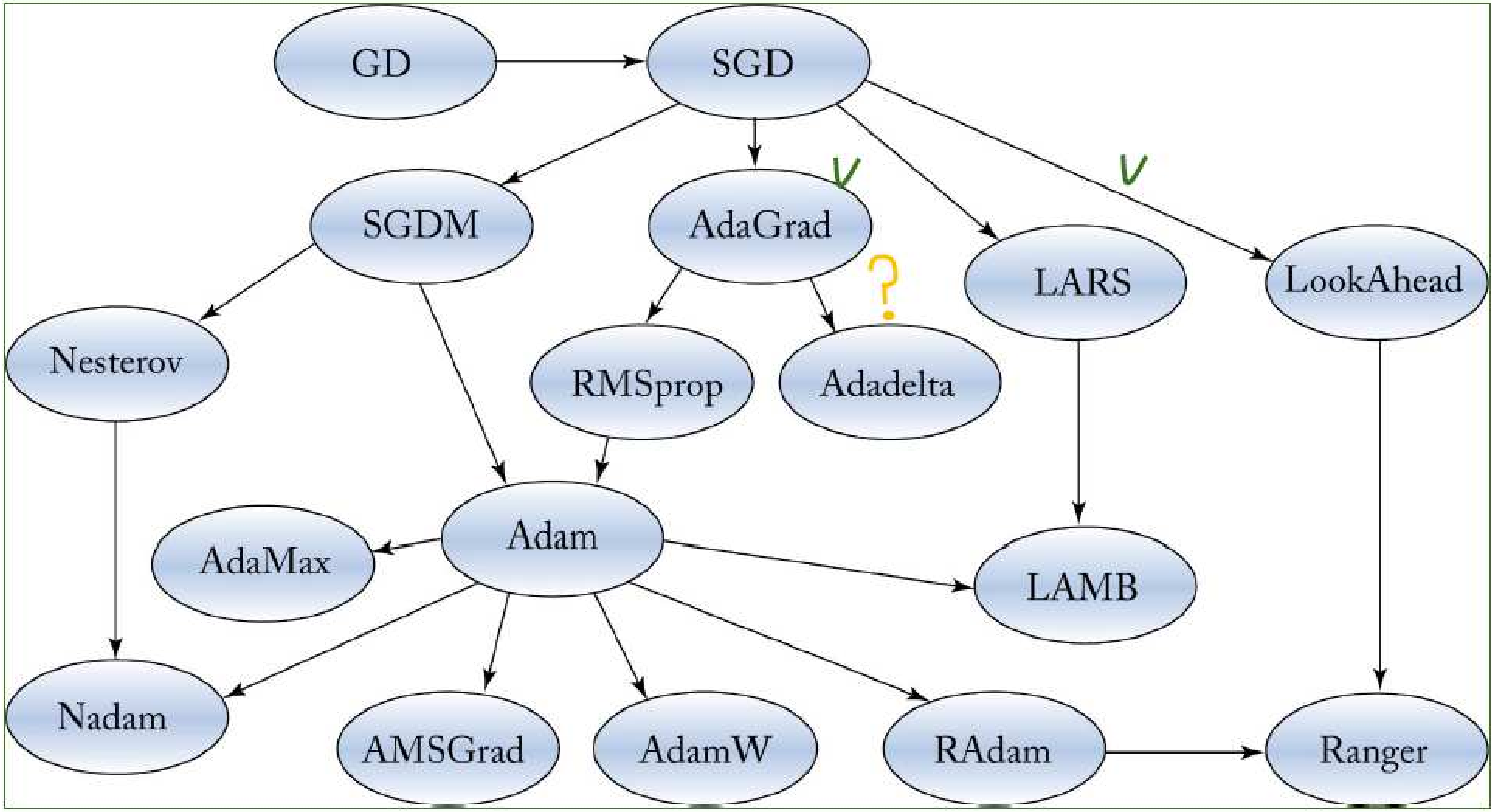
1. В BackProp вычислять вторые производные $\frac{\partial^2 Q}{\partial w_{jh}^2}$, $\frac{\partial^2 Q}{\partial w_{hm}^2}$.
2. Если процесс минимизации $Q(w)$ пришёл в минимум,то
   - упорядочить все веса по убыванию S_{jh};
   - удалить N связей с наименьшей значимостью;
   - снова запустить BackProp.
3. Если $Q(w, X^\ell)$ или $Q(w, X^k)$ существенно ухудшился, то вернуть последние удалённые связи и выйти.

Отбор признаков с помощью OBD — аналогично.

Суммарная значимость признака: $S_j = \sum_{h=1}^{H} S_{jh}$.

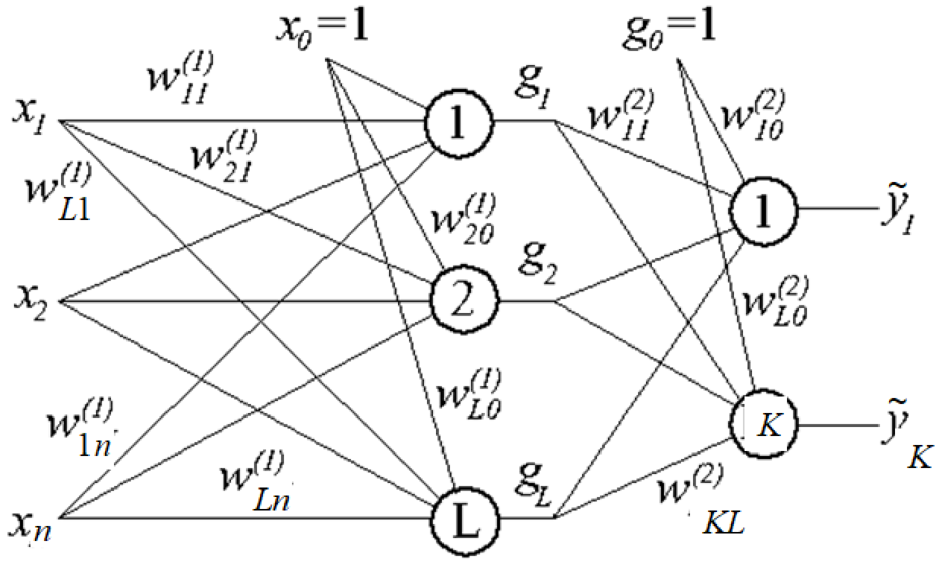
Эмпирический опыт: результат постепенного прореживания обычно лучше, чем BackProp изначально прореженной сети.

Модификации SGD



Source: https://deeplearningsystems.ai/#ch04/#43-optimization-algorithms-minimizing-the-cost

Модификации стохастического градиента: SGD



Пусть $w \equiv w_{mk}^{(h)}$ вес для какого-то нейрона на каком-то слое

$L' = \dfrac{\partial Q(w)}{\partial w}$ - производная функции потерь по весам

Пересчет весов градиентного спуска:

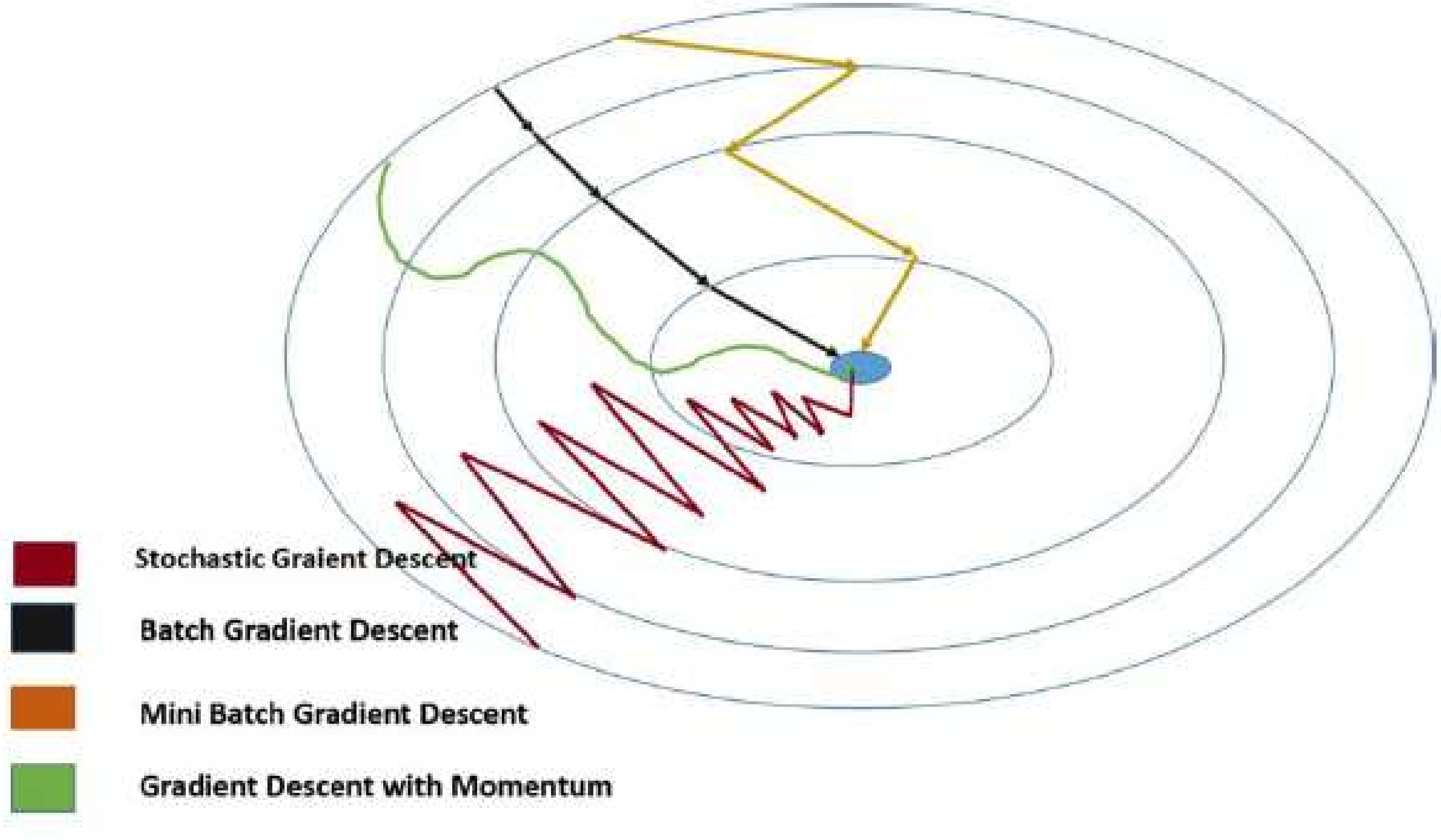
$$w := w - \eta L'$$

Momentum - экспоненциальное скользящее среднее градиента [Б.Т. Поляк, 1964].

$$v := \gamma v + (1-\gamma)L',$$
$$w := w - \eta v$$

где $\eta > 0$ - параметр (темп обучения).

$\gamma \in [0,1]$ - параметр сглаживания



NAG (Nesterov's accelerated gradient)
- стохастический градиент с инерцией [Ю.И. Нестеров, 1983].

$$v := \gamma v + (1-\gamma)\frac{\partial Q(w-\eta\gamma v)}{\partial w},$$
$$w := w - \eta v$$

где $\eta > 0$ - параметр (темп обучения).
$\gamma \in [0,1]$ - параметр сглаживания

RMSProp (running mean square)

выравнивание скоростей изменения весов скользящим средним, ускоряет обучение по весам, которые мало изменялись.

$$v := \gamma v + (1-\gamma) L' \odot L',$$
$$w := w - \eta L' \div \left(\sqrt{v} + \varepsilon\right)$$

где $\eta > 0$ - параметр (темп обучения).
$\gamma \in [0,1]$ - параметр сглаживания
$\odot$ - операция покоординатного умножения векторов
$\div$ - операция покоординатного деления векторов

AdaDelta (Adaptive Learning Rate)

двойная нормировка приращений весов, после которой можно взять темп обучения равным 1.

$$\delta = L' \odot \left(\sqrt{\Delta} + \varepsilon\right) \div \left(\sqrt{v} + \varepsilon\right)$$
$$\Delta := \alpha\Delta + (1-\alpha)\delta \odot \delta$$
$$v := \gamma v + (1-\gamma) L' \odot L',$$
$$w := w - \eta\delta$$

где $\eta > 0$ - параметр (темп обучения).
$\alpha, \gamma \in [0,1]$ - параметры сглаживания
$\odot$ - операция покоординатного умножения векторов
$\div$ - операция покоординатного деления векторов

Adam (adaptive momentum) = инерция + RMSProp

$$v := \frac{\alpha v + (1-\alpha) L' \odot L'}{1-\alpha^t},$$

$$\mu := \frac{\gamma v + (1-\gamma) L'}{1-\gamma^t}$$

$$w := w - \eta \mu \div \left(\sqrt{v} + \varepsilon \right)$$

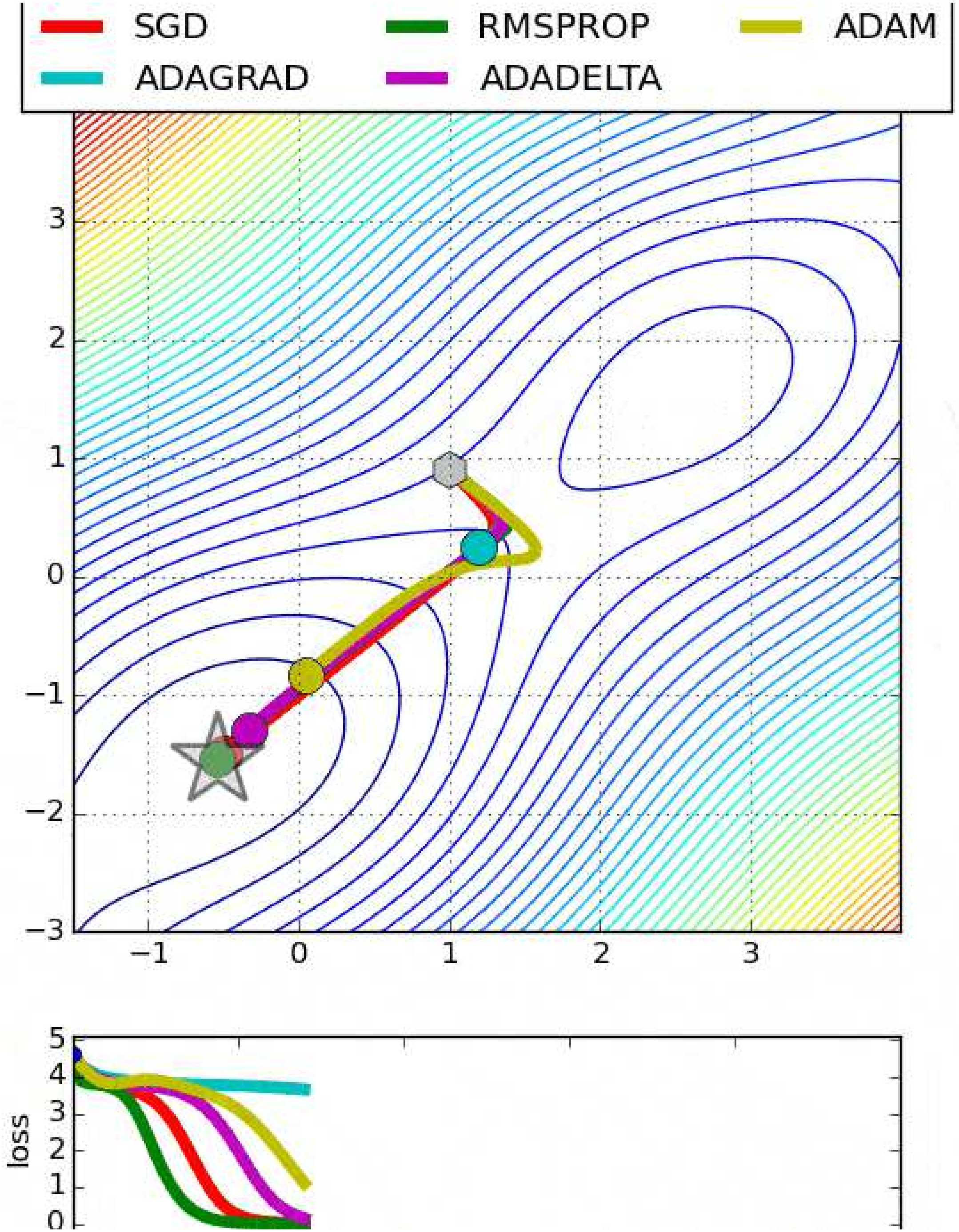
где $\eta > 0$ - параметр (темп обучения).

$\alpha, \gamma \in [0,1]$ - параметры сглаживания

$\odot$ - операция покоординатного умножения векторов

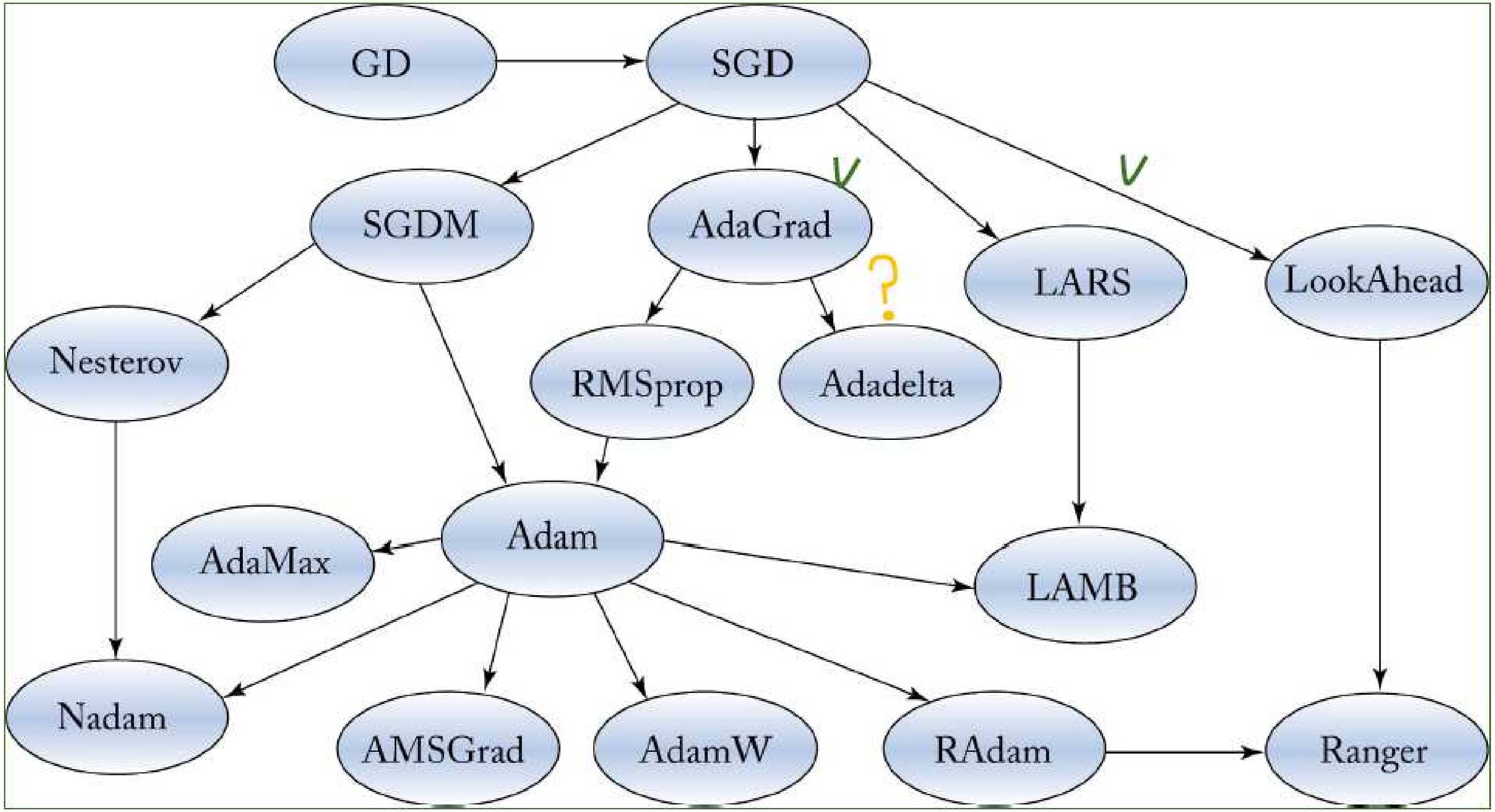
$\div$ - операция покоординатного деления векторов

t - номер итерации



https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

Модификации SGD



Source: https://deeplearningsystems.ai/#ch04/#43-optimization-algorithms-minimizing-the-cost