Deep Learning

К. В. Воронцов, А.В. Зухба
vokov@forecsys.ru
a__l@mail.ru

декабрь 2014

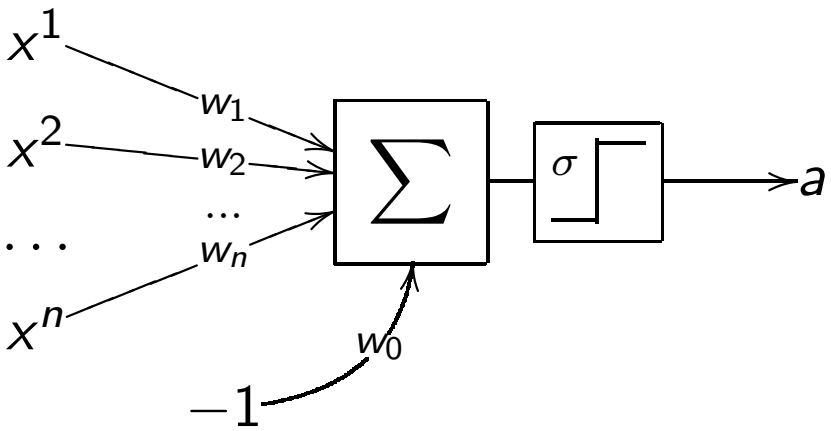
Содержание

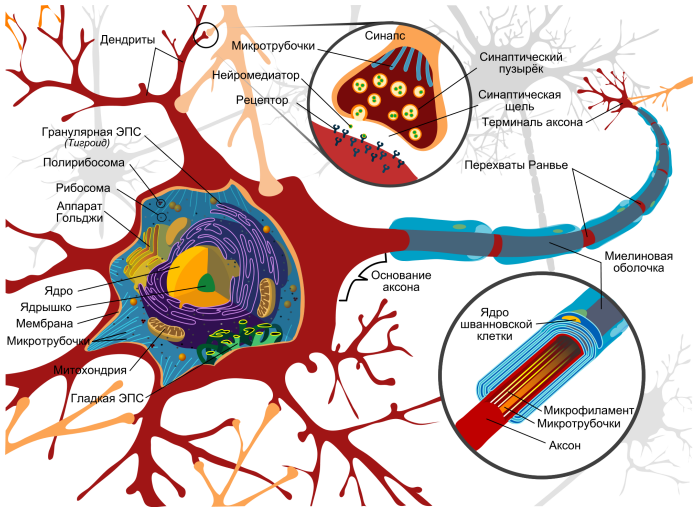
Напоминание: линейная модель нейрона МакКаллока-Питтса

$f_j : X \to \mathbb{R}$, $j = 1, \ldots, n$ — числовые признаки;

$$a(x, w) = \sigma\big(\langle w, x \rangle\big) = \sigma\left(\sum_{j=1}^{n} w_j f_j(x) - w_0 \right),$$

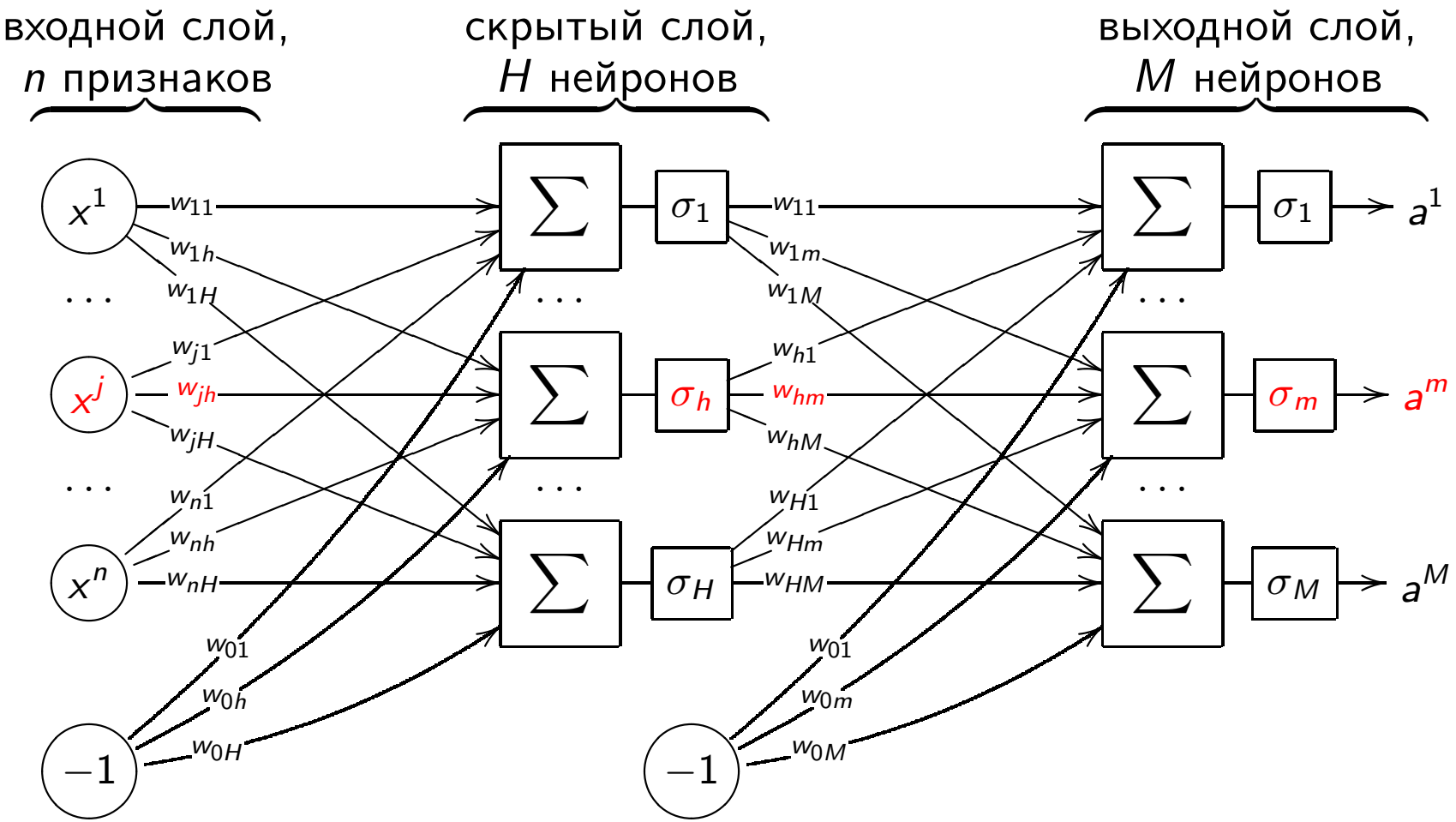
где $w_0, w_1, \ldots, w_n \in \mathbb{R}$ — веса признаков;
$\sigma(s)$ — функция активации (в частности, sign).





Многослойная нейронная сеть

Пусть для общности $Y = \mathbb{R}^M$, для простоты слоёв только два.



Напоминание: Алгоритм SG (Stochastic Gradient)

Задача минимизации суммарных потерь:

$$Q(w) := \sum_{i=1}^{\ell} \mathscr{L}(w, x_i, y_i) \to \min_w .$$

Вход: выборка X^ℓ; темп обучения η; параметр λ;
Выход: веса $w \equiv \left(w_{jh}, w_{hm}\right) \in \mathbb{R}^{H(n+M+1)+M}$;

1: инициализировать веса w и текущую оценку $Q(w)$;
2: **повторять**
3: выбрать объект x_i из X^ℓ (например, случайно);
4: вычислить потерю $\mathscr{L}_i := \mathscr{L}(w, x_i, y_i)$;
5: градиентный шаг: $w := w - \eta \nabla \mathscr{L}(w, x_i, y_i)$;
6: оценить значение функционала: $Q := (1-\lambda)Q + \lambda \mathscr{L}_i$;
7: **пока** значение Q и/или веса w не стабилизируются;

Задача дифференцирования суперпозиции функций

Выходные значения сети $a^m(x_i)$, $m = 1..M$ на объекте x_i:

$$a^m(x_i) = \sigma_m\left(\sum_{h=0}^{H} w_{hm} u^h(x_i)\right); \qquad u^h(x_i) = \sigma_h\left(\sum_{j=0}^{J} w_{jh} f_j(x_i)\right).$$

Пусть для конкретности $\mathscr{L}_i(w)$ — средний квадрат ошибки:

$$\mathscr{L}_i(w) = \frac{1}{2}\sum_{m=1}^{M}\left(a^m(x_i) - y_i^m\right)^2.$$

Промежуточная задача: найти частные производные

$$\frac{\partial \mathscr{L}_i(w)}{\partial a^m}; \qquad \frac{\partial \mathscr{L}_i(w)}{\partial u^h}.$$

Быстрое вычисление градиента

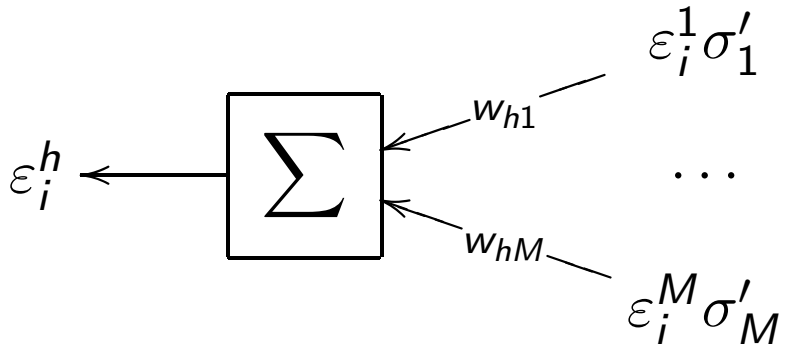
Промежуточная задача: частные производные

$$\frac{\partial \mathscr{L}_i(w)}{\partial a^m} = a^m(x_i) - y_i^m = \varepsilon_i^m$$

— это ошибка на выходном слое;

$$\frac{\partial \mathscr{L}_i(w)}{\partial u^h} = \sum_{m=1}^{M} \big(a^m(x_i) - y_i^m\big)\sigma'_m w_{hm} = \sum_{m=1}^{M} \varepsilon_i^m \sigma'_m w_{hm} = \varepsilon_i^h$$

— назовём это *ошибкой на скрытом слое*. Похоже, что ε_i^h вычисляется по ε_i^m, если запустить сеть «задом наперёд»:



Быстрое вычисление градиента

Теперь, имея частные производные $\mathscr{L}_i(w)$ по a^m и u^h, легко выписать градиент $\mathscr{L}_i(w)$ по весам w:

$$\frac{\partial \mathscr{L}_i(w)}{\partial w_{hm}} = \frac{\partial \mathscr{L}_i(w)}{\partial a^m}\frac{\partial a^m}{\partial w_{hm}} = \varepsilon_i^m \sigma'_m u^h(x_i), \quad m = 1..M, \quad h = 0..H;$$

$$\frac{\partial \mathscr{L}_i(w)}{\partial w_{jh}} = \frac{\partial \mathscr{L}_i(w)}{\partial u^h}\frac{\partial u^h}{\partial w_{jh}} = \varepsilon_i^h \sigma'_h f_j(x_i), \quad h = 1..H, \quad j = 0..n;$$

Алгоритм обратного распространения ошибки BackProp:

Вход: $X^\ell = (x_i, y_i)_{i=1}^\ell \subset \mathbb{R}^n \times \mathbb{R}^M$; параметры H, λ, η;
Выход: синаптические веса w_{jh}, w_{hm};

1: ...

Алгоритм BackProp

1: инициализировать веса w_{jh}, w_{hm};
2: **повторять**
3: выбрать объект x_i из X^ℓ (например, случайно);
4: прямой ход:
$u_i^h := \sigma_h\big(\sum_{j=0}^{J} w_{jh} x_i^j\big)$, $h = 1..H$;
$a_i^m := \sigma_m\big(\sum_{h=0}^{H} w_{hm} u_i^h\big)$, $\varepsilon_i^m := a_i^m - y_i^m$, $m = 1..M$;
$\mathscr{L}_i := \sum_{m=1}^{M} (\varepsilon_i^m)^2$;
5: обратный ход:
$\varepsilon_i^h := \sum_{m=1}^{M} \varepsilon_i^m \sigma'_m w_{hm}$, $h = 1..H$;
6: градиентный шаг:
$w_{hm} := w_{hm} - \eta \varepsilon_i^m \sigma'_m u_i^h$, $h = 0..H$, $m = 1..M$;
$w_{jh} := w_{jh} - \eta \varepsilon_i^h \sigma'_h x_i^j$, $j = 0..n$, $h = 1..H$;
7: $Q := (1 - \lambda) Q + \lambda \mathscr{L}_i$;
8: **пока** Q не стабилизируется;

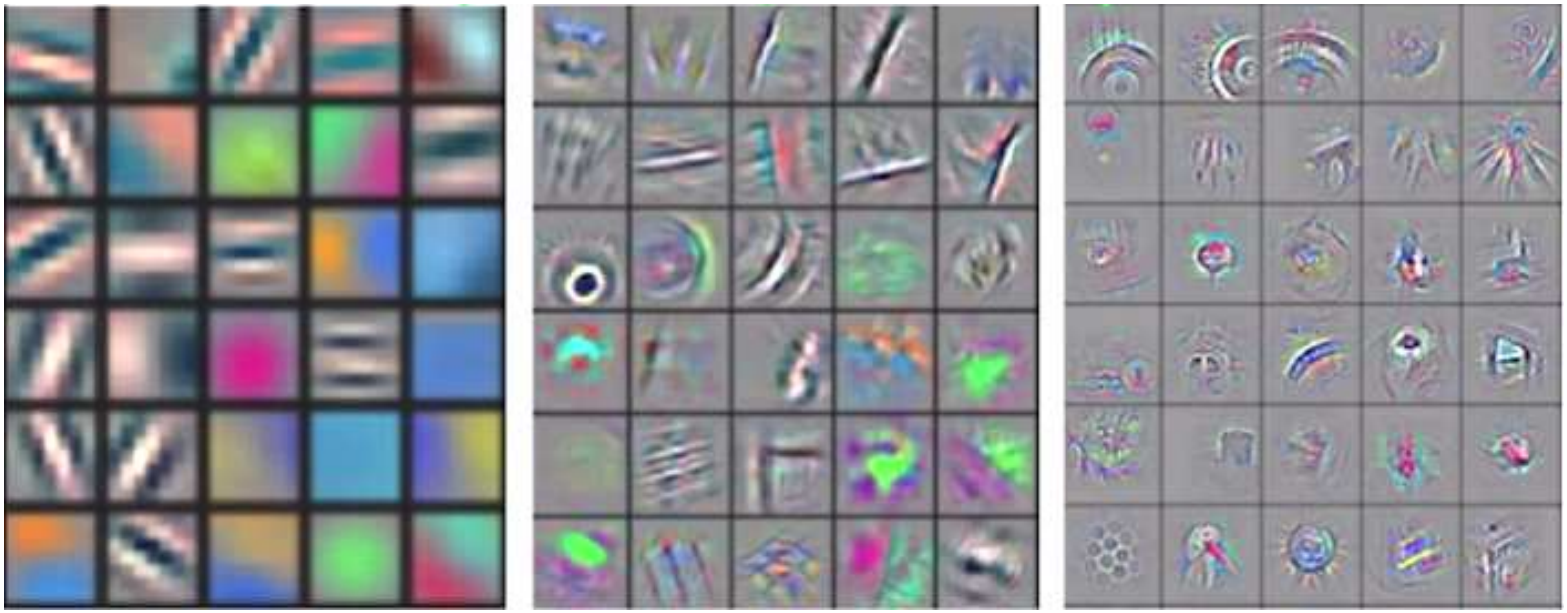
Почему нейросети снова популярны?

- Большинство используемых на сегодняшний день методов придуманы в 80х

Что сделало их актуальными?

- Большие вычислительные возможности
- Большие объемы данных
- Новые методы предобучения, новые архитектуры, новые задачи

Извлечение признаков во время обучения:



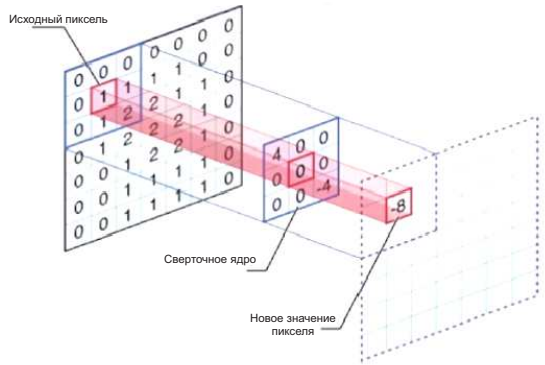
Сверточные нейронные сети

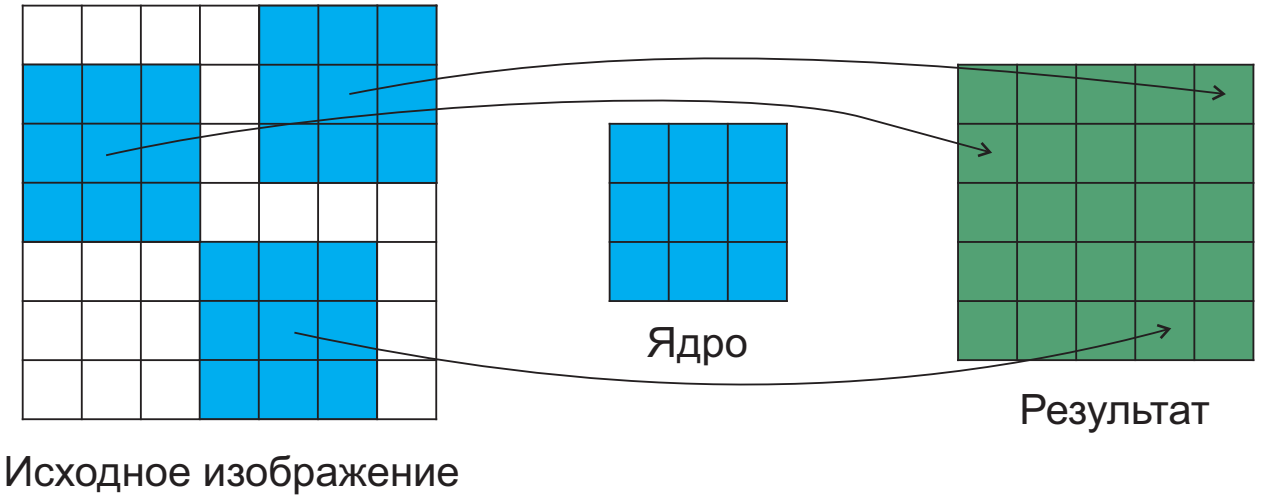
Архитектура выбирается таким образом, чтобы заложить в нее априорные знания из предметной области:

- Пиксель изображения сильнее связан с соседними пикселями (локальная корреляция)
- Объект может встретиться в любой части изображения (инвариантность к перемещению)
- Строим иерархическую структуру и стремимся к выделению паттернов

Операция свертки

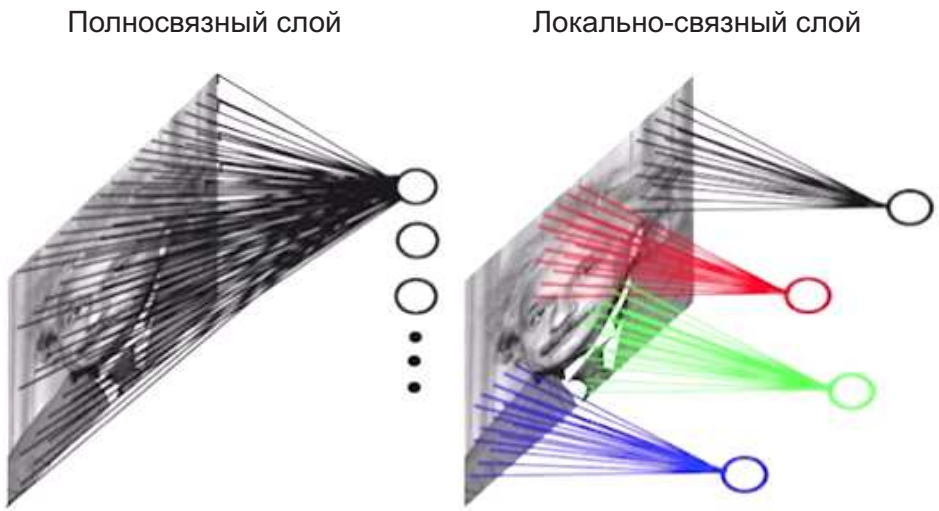
Каждый фрагмент изображения поэлементно умножается на небольшую матрицу весов (ядро), результат суммируется .





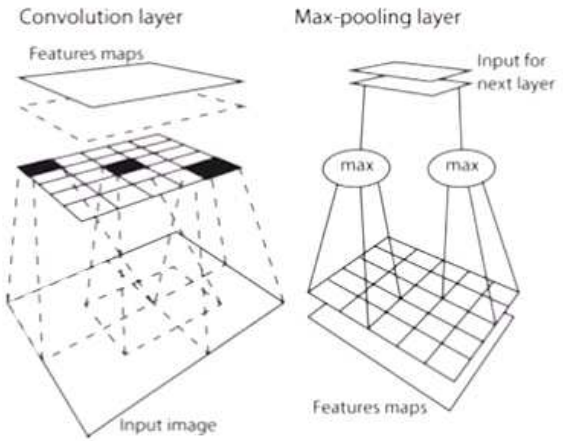
Сверточный слой

- Это локально-связный слой с одинаковыми весами в разных частях изображения
- Закладывает в модель априорное знание о том, что объект может встретиться в любой части изображения

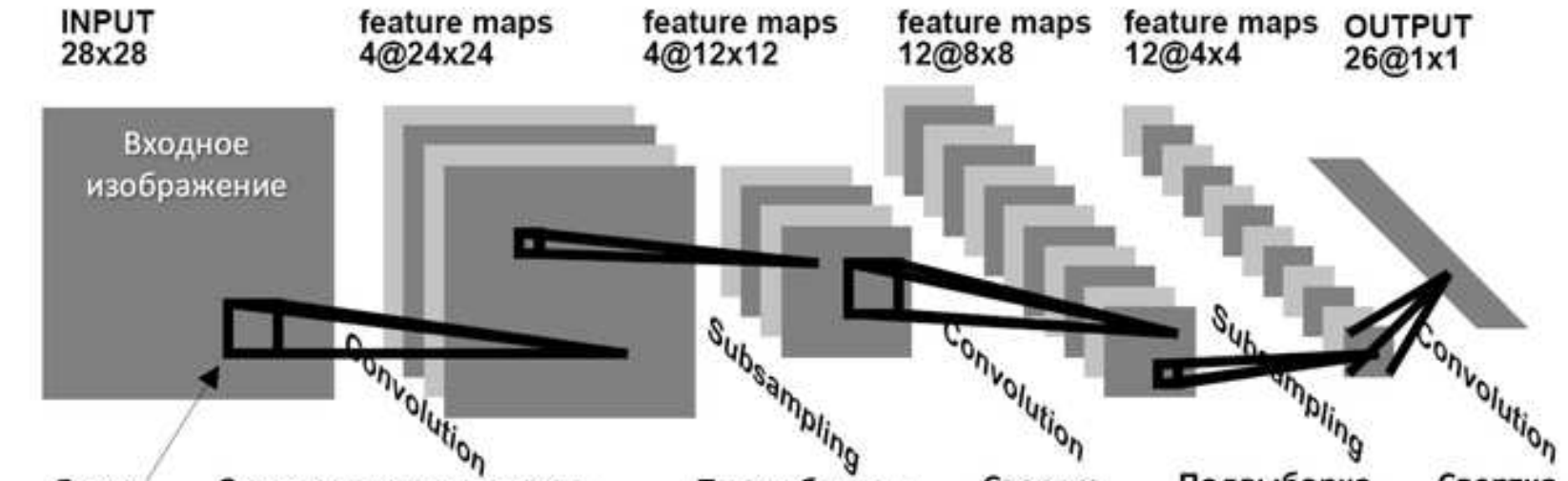


Сверточный сабсемплинга(subsampling)

- Max-сабсемплинг аналогичен сверточному слою, но операция "+"заменена на "max"
- Добавляет устойчивости к небольшим деформациям
- Иногда в русскоязычной литературе называется слой подвыборки



Архитектура сверточной нейронной сети



Ядро свертки – матрица из весов, например, размером 5x5

Свертка – проход ядром свертки по всем нейронам нижнего слоя для активации нейронов верхнего слоя – получение карты признаков (feature maps). Для ядер разного типа получаются разные карты признаков.

Подвыборка – операция уменьшения размерности. Чаще всего – выбор максимума или поиск среднего по нескольким соседним нейронам.

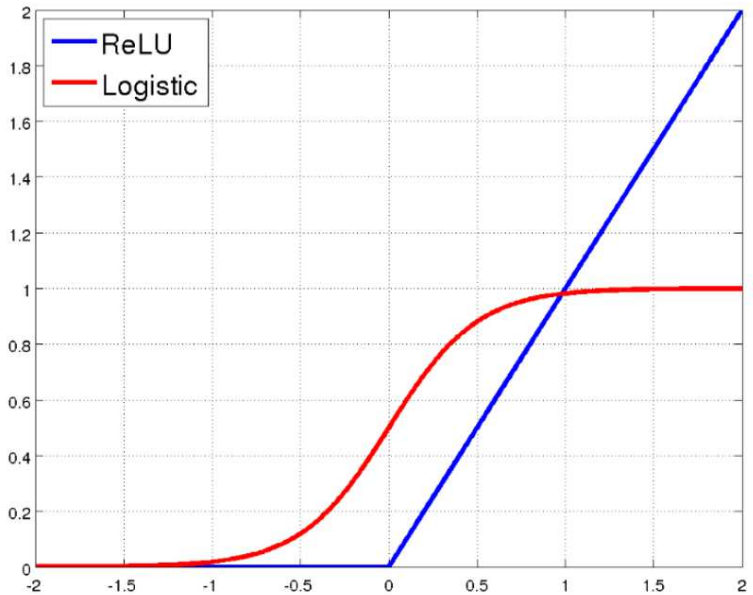
Свертка

Подвыборка

Свертка

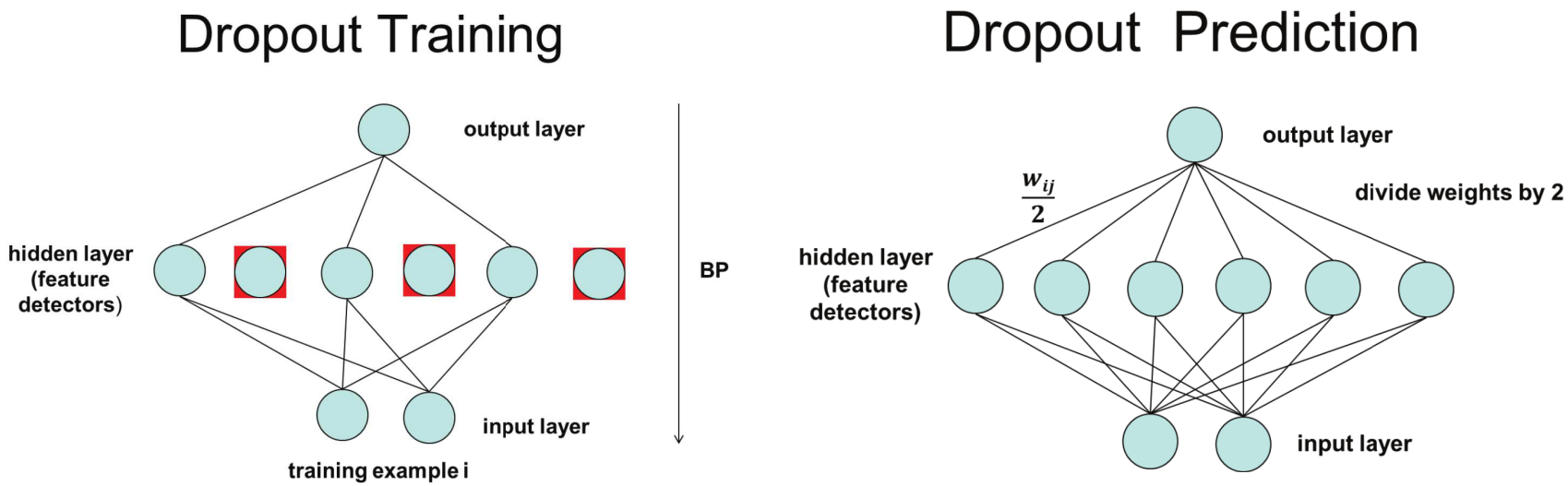
ReLU

- Градиент не затухает
- Выходы уходят в ноль



Dropout

Dropout - способ дешево обучать ансамбли из 2^N моделей, где N - количество нейронов.
Реализует приближение геометрического среднего выходов.



Достоинства и недостатки

Преимущества:

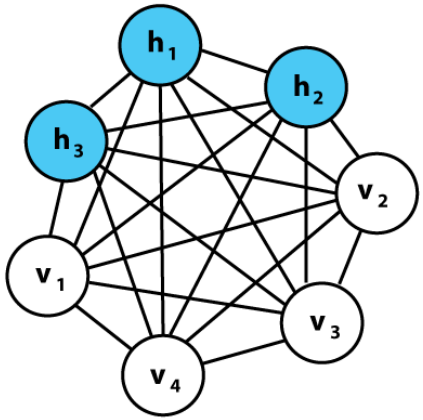
- Один из лучших алгоритмов по распознаванию и классификации изображений.
- По сравнению с полносвязной нейронной сетью (типа перцептрона) — гораздо меньшее количество настраиваемых весов.
- Удобное распараллеливание вычислений, а, следовательно, возможность реализации алгоритмов работы и обучения сети на графических процессорах.
- Относительная устойчивость к повороту и сдвигу распознаваемого изображения.
- Обучение при помощи классического метода обратного распространения ошибки.

Недостатки:

- Требуются априорные знания о локальной корреляции.
- Много варьируемых параметров.

Машина Больцмана

Машина Больцмана - вид стохастической рекуррентной нейронной сети, изобретенной Джеффри Хинтоном и Терри Сейновски в 1985 году. Является случайным марковским полем .

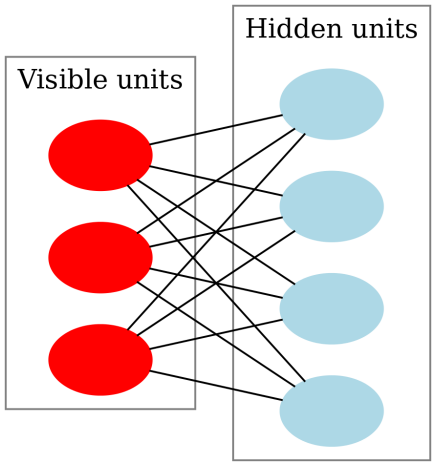


Глобальная энергия: $E = -\sum_{i<j} w_{ij} s_i s_j - \sum_i \theta_i s_i$

- θ_i порог для нейрона i.
- s_i состояние нейрона i. $s_i \in \{0, 1\}$.
- w_{ij} сила связи между нейронами j и i. $w_{ii} = 0$; $w_{ij} = w_{ji}$

Ограниченная машина Больцмана

- Ограниченная машина Больцмана (restricted Boltzmann machine; RBM) является двудольным графом: связи существуют только между скрытыми и видимыми нейронами.
- Из каскада ограниченных машин Больцмана строится так называемая глубокая сеть доверия (deep belief network).



.

При заданном состоянии нейронов одной группы, состояние нейронов другой группы будут независимы друг от друга. (Вспоминаем условную независимость из графических моделей.)

Ограниченная машина Больцмана

- w_{ij} - вес ребра, сила связи между i-м нейроном видимого слоя и j-м нейроном скрытого слоя.
- a_i - смещение видимого нейрона
- b_j - смещение скрытого нейрона
- v_i - состояние видимого нейрона
- h_j - состояние скрытого нейрона

$$E(v,h) = -\sum_{i}^{n} a_i v_i - \sum_{j}^{m} b_j h_j - \sum_{i}^{n}\sum_{j}^{m} w_{ij} v_i h_j$$

Совместная вероятность всевозможных пар v и h:

$$p(v,h) = \frac{1}{Z} exp(-E(v,h))$$

Ограниченная машина Больцмана

Рассмотрим вероятность того, что при заданном v одно из скрытых состояний $h_k = 1$.

$$p(h_k = 1|v) = \sigma(-b - \sum_{i}^{n} v_i w_{ik})$$

При заданном v все h_k Не зависят друг от друга:

$$P(h|v) = \prod_{j}^{m} P(h_j|v)$$

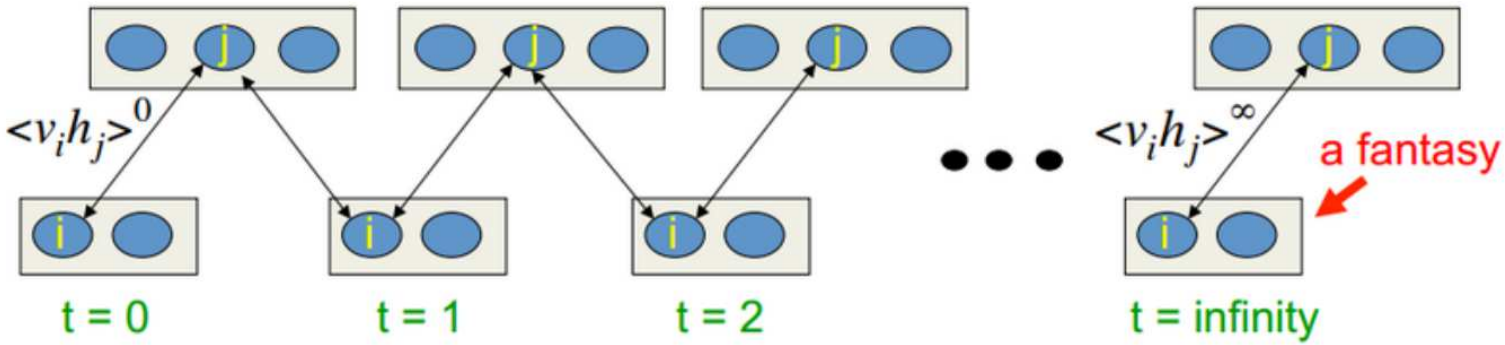
Аналогично для h при заданном v.

Contrastive Divergence

Алгоритм придуман Хинтоном в 2002 году.

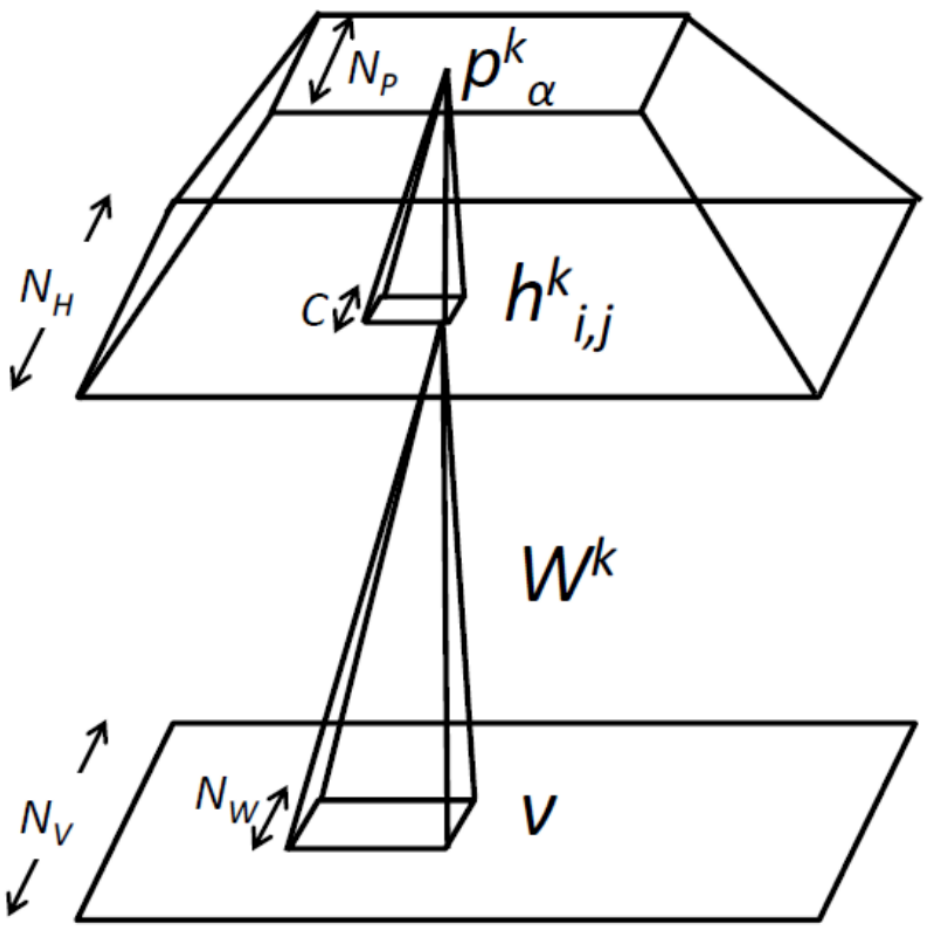
1. состояние входных нейронов приравнивается к входному образу
2. выводятся вероятности скрытого слоя
3. каждому нейрону скрытого слоя ставится в соответствие состояние "1"с вероятностью, равной его текущему состоянию
4. выводятся вероятности видимого слоя на основании скрытого
5. если текущая итерация меньше k, то возврат к шагу 2
6. выводятся вероятности состояний скрытого слоя

Contrastive Divergence



- $\triangle w_{ij} = \eta\left(M[v_i h_j]^{(0)} - M[v_i h_j]^{(\infty)}\right)$
- $\triangle a_i = \eta\left(v_i - M[v_i]^{(\infty)}\right)$
- $\triangle b_j = \eta\left(M[h_j]^{(0)} - M[h_j]^{(\infty)}\right)$

Deep belief network



P^k (pooling layer)

H^k (detection layer)

V (visible layer)

Deep belief network

Сеть глубокого обучения как последовательность ограниченных машин Больцмана

$$P(x, h_1, \ldots, h^l) = \Big(\prod_{k=0}^{l-2} P(h^k | h^{k+1})\Big) P(h^{l-1}, h^l)$$