ТЕХНОСФЕРА

Ограниченная машина Больцмана как основа глубоких нейронных сетей

Нестеров Павел

29 января 2015 г.

План

Нервная система до нейробиологии

Рис.: 17 век, Рене Декарт о нервной системе: «Раздражение ступни передаётся по нервам в мозг, взаимодействует там с духом и таким образом порождает ощущение боли».

Нервная система в современном понимании

- В 1906 году врач и гистолог Сантьяго Рамон-и-Кахаль совместно с врачем Камилло Гольджи получают нобелевскую премию за "за работы по структуре нервной системы"; их работы заложили основы нейронной теории нервной системы и современной нейробиологии.

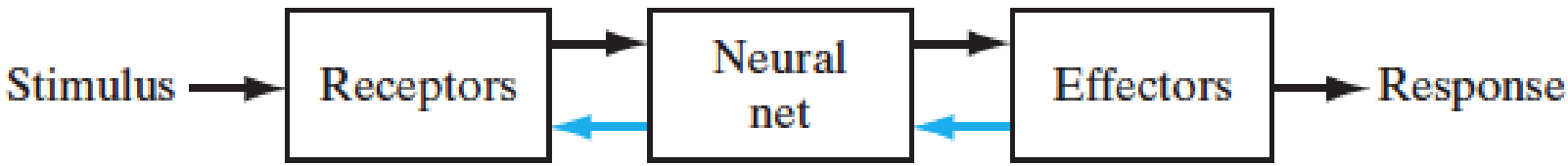
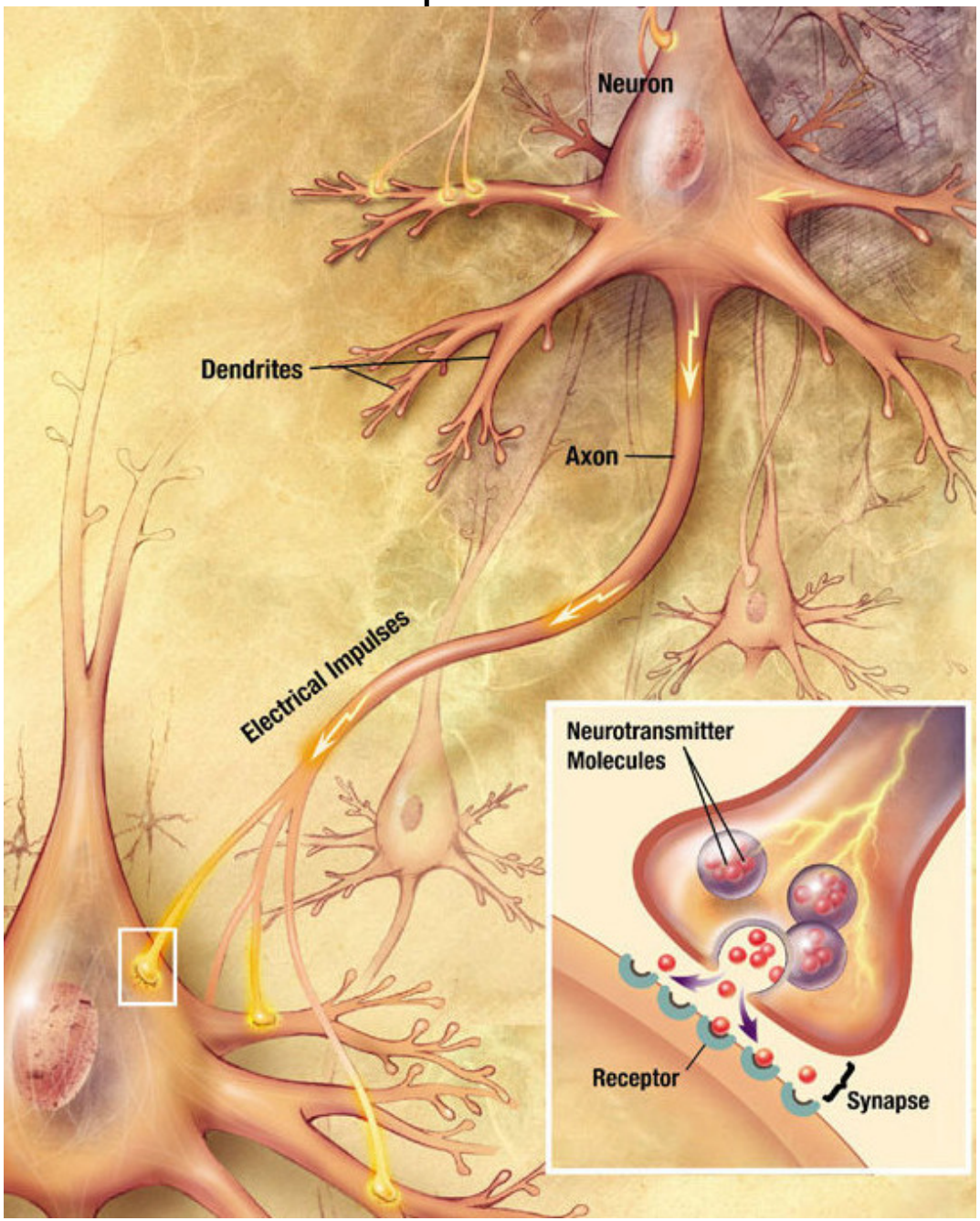


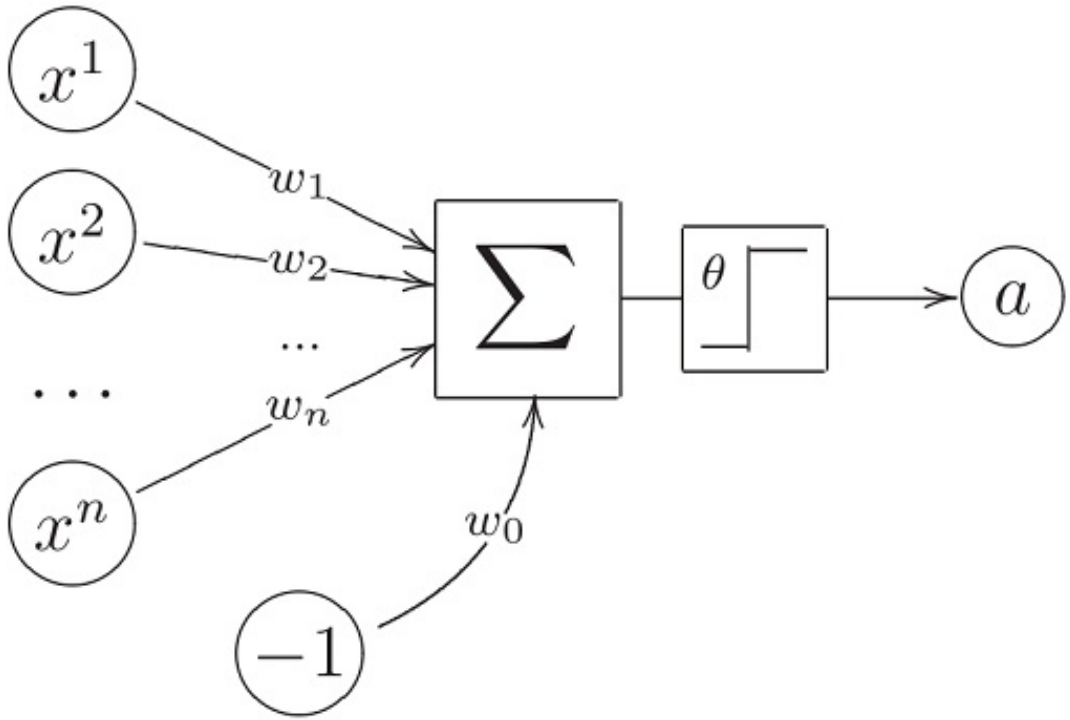
Рис.: Блок-схема нервной системы[1]

[1] Neural Networks and Learning Machines (3rd Edition), Simon O. Haykin

Схема биологического нейрона



Модель МакКаллока-Питтса (1943 год)



- $a(x) = \theta\left(\sum_{j=1}^{n} w_j \cdot x_i - w_0\right)$;
- $\theta(z) = [z \geq 0] = \begin{cases} 0, z<0 \\ 1, z \geq 0 \end{cases}$ - функция Хевисайда;
- эквивалентно линейному классификатору.

Данная модель, с незначительными изменениями, актуальна и по сей день.

Нейронные ансамбли

- Физиолог и нейропсихолог Дональд Хебб разработал теорию взаимосвязи головного мозга и мыслительных процессов в книге "The Organization of Behavior"(1949).
- Нейронный ансамбль - совокупность нейронов, составляющих функциональную группу в высших отделах мозга.
- Нейроансамбль - распределенный способ кодирования информации.
- Нейрон сам по себе генерирует по мимо сигнала еще и шум, но ансамбль в среднем генерирует чистый сигнал (*аналогии с bagging*).

Правила Хебба (1949)

В своей книге Дональд Хебб описал процесс адаптирования нейронов в мозге в процессе обучения, и сформулировал базовые механизмы нейропластичности:

1. если два нейрона по разные стороны от синапсов активируются синхронно, то "вес"синапса слегка возрастает;
2. если два нейрона по разные стороны от синапсов активируются асинхронно, то "вес"синапса слегка ослабевает или синапс удаляется[2].

Эти правила легли в основу зарождающейся теории нейросетей, сегодня в мы можем увидеть этот мета-алгоритм в основных методах обучения нейронных сетей.

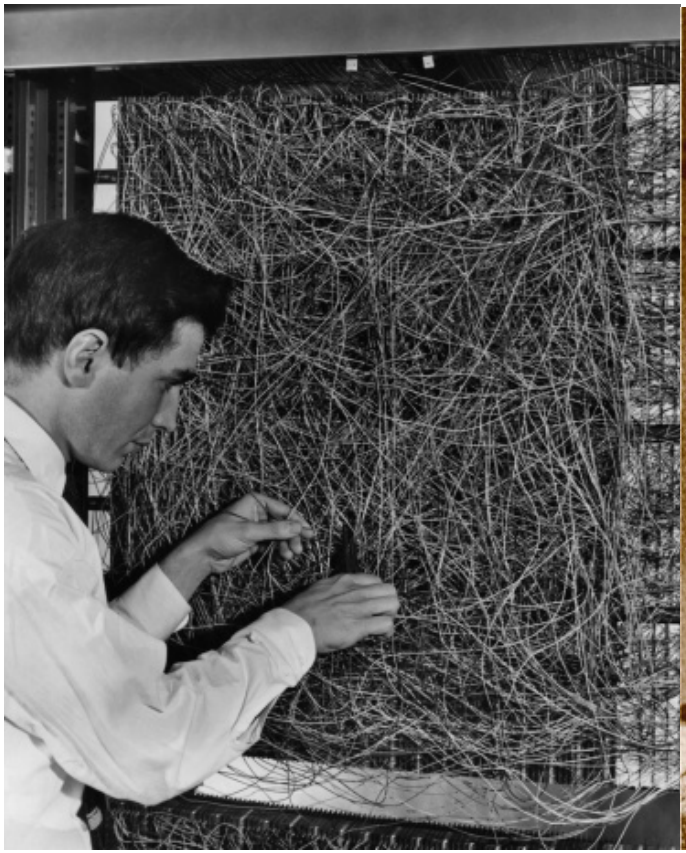
[2] это расширенное правило, в оригинале второй части не было

Однослойный персептрон Розенблатта (1958 год)

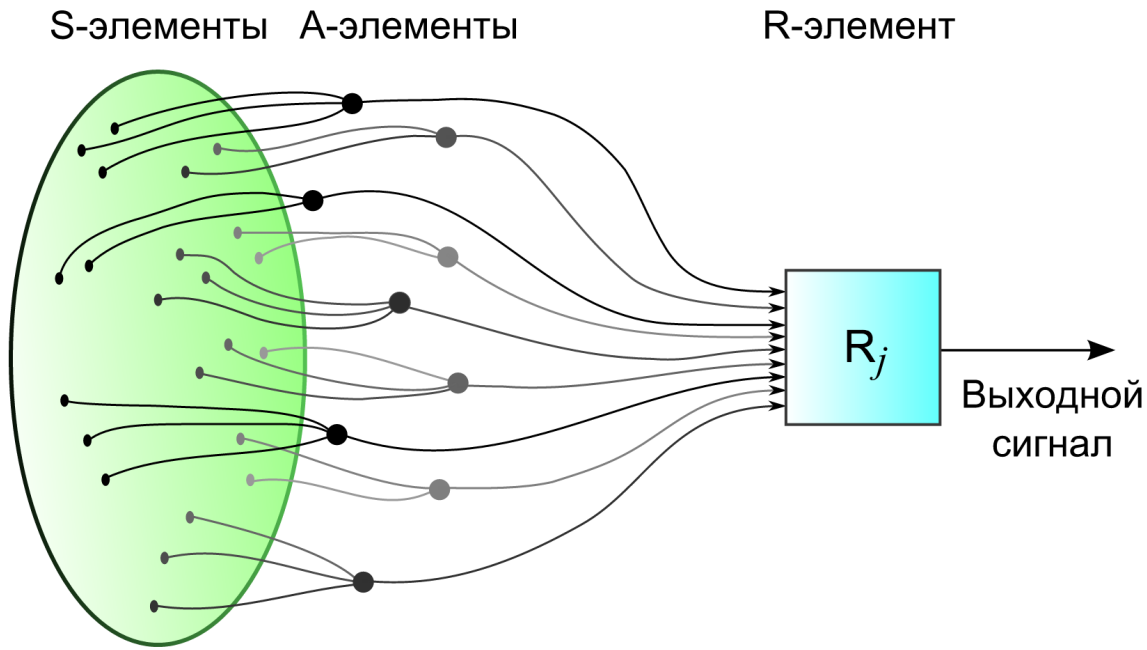
Нейрофизиолог Френк Розенблатт предложил схему устройства, моделирующего процесс человеческого восприятия, и назвал его "персептроном". По мимо этого:

- показал, что персептрон может выполнять базовые логические операции;
- разработал алгоритм обучения такой модели - метод коррекции ошибки;
- доказал сходимость алгоритма (теорема сходимости персептрона), но только для линейно разделимых классов;
- реализовал физический прототип такой модели;
- реализовал первый в мире нейрокомпьютер "MARK-1".

Нейрокомпьютер MARK-1 и Френк Розенблатт



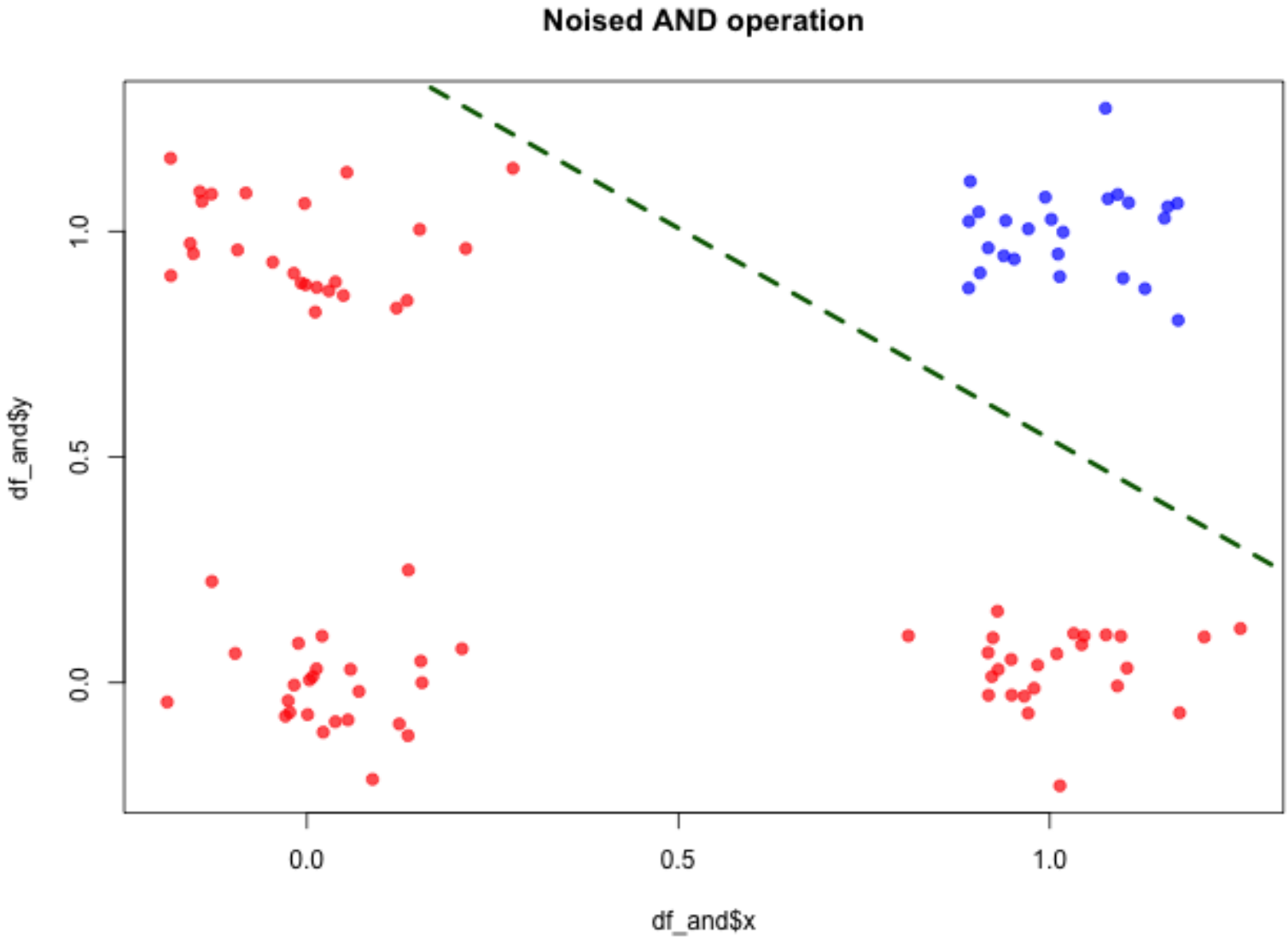
Элементарный персептрон



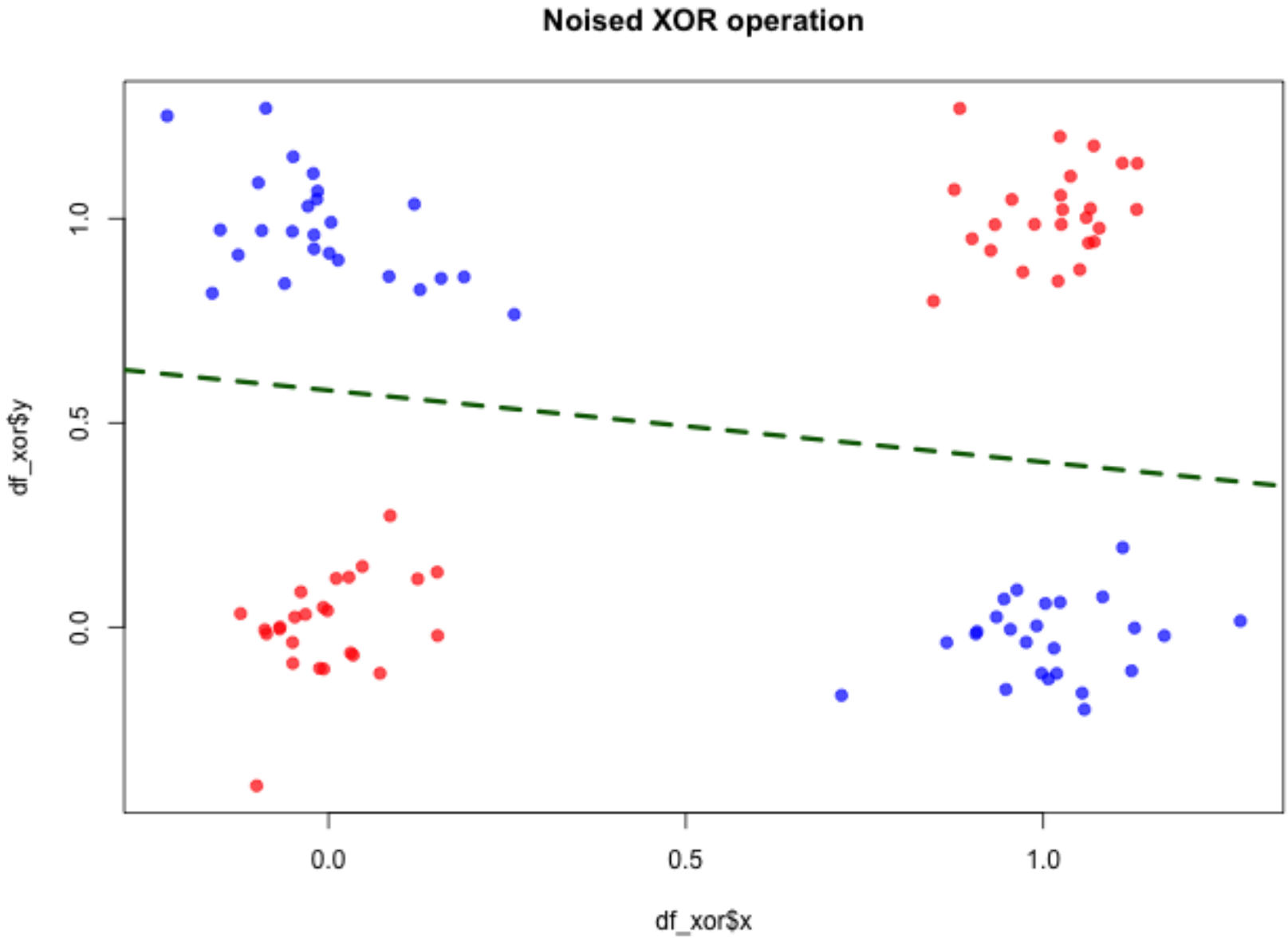
Метод коррекции ошибки

- $\hat{y}_i = 0 \wedge y_i = 1 \Rightarrow \Delta w = \eta(n) \cdot x(n)$
- $\hat{y}_i = 1 \wedge y_i = 0 \Rightarrow \Delta w = -\eta(n) \cdot x(n)$
  - $\eta(n)$ - скорость обучения, зависящая от итерации
  - $x(n)$ - входной образ на итерации n

Недостатки элементарного персептрона, AND



Недостатки элементарного персептрона, XOR



Анимация сходимости для операций AND и XOR

- операция OR - 1layer-net-or.gif
- операция AND - 1layer-net-and1.gif
- операция XOR - 1layer-net-xor.gif [3]

[3] http://theclevermachine.wordpress.com/2014/09/11/a-gentle-introduction-to-artificial-neural-networks/

Доказательства неэффективности нейронных сетей

- В 1969 году математик и исследователь ИИ Марвин Минский провел детальный математический анализ персептрона и опубликовал формальное доказательство ограниченности этой модели.
- *"**There is no reason to suppose** that any of these virtues carry over to the many-layered version. Nevertheless, we consider it to be an important research problem to elucidate (or reject) our intuitive judgement that the extension to multilayer systems is sterile."*[4]
- Отсутствие преимуществ + ограничения модели в итоге поубавили интерес научного сообщества к нейронным сетям, требовалось, что то принципиально новое

[4] Персептроны, Марвин Минский в соавторстве с Сеймуром Папертом, MIT Press, 1969

Период "забвения"

Исследования *искусственных* нейросетей не спеша продолжаются, но в режиме поиска чего-то нового:

- 1972: Т. Кохонен разрабатывает новый тип нейросетей, способных функционировать в качестве памяти;
- 1975-1980: К. Фукусима разрабатывает когнитрон и неокогнитрон, совершенно новый тип многослойной сверточной сети, созданной по аналогии со строением зрительной системы;
- 1982: Дж. Хопфилд разрабатывает новый тип нейросети с обратными связями, выполняющей функции ассоциативной памяти;
- 1986: Дэвид Румельхарт, **Дж. Хинтон** и Рональд Вильямс разрабатывают *вычислительно эффективный* алгоритм обучения многослойных нейросетей - метод обратного распространения ошибки (именно работа этих авторов возрождает интерес к нейронным сетям в мире).

Теорема универсальной аппроксимации[5]

Введем следующие обозначения:

- $\phi(x)$ - не тождественная, ограниченная и монотонно возрастающая функция
- I_n - n-мерный единичный гиперкуб
- $C(I_n)$ - множество непрерывных функций на I_n

$\Rightarrow \forall f \wedge \forall \epsilon > 0 \exists$

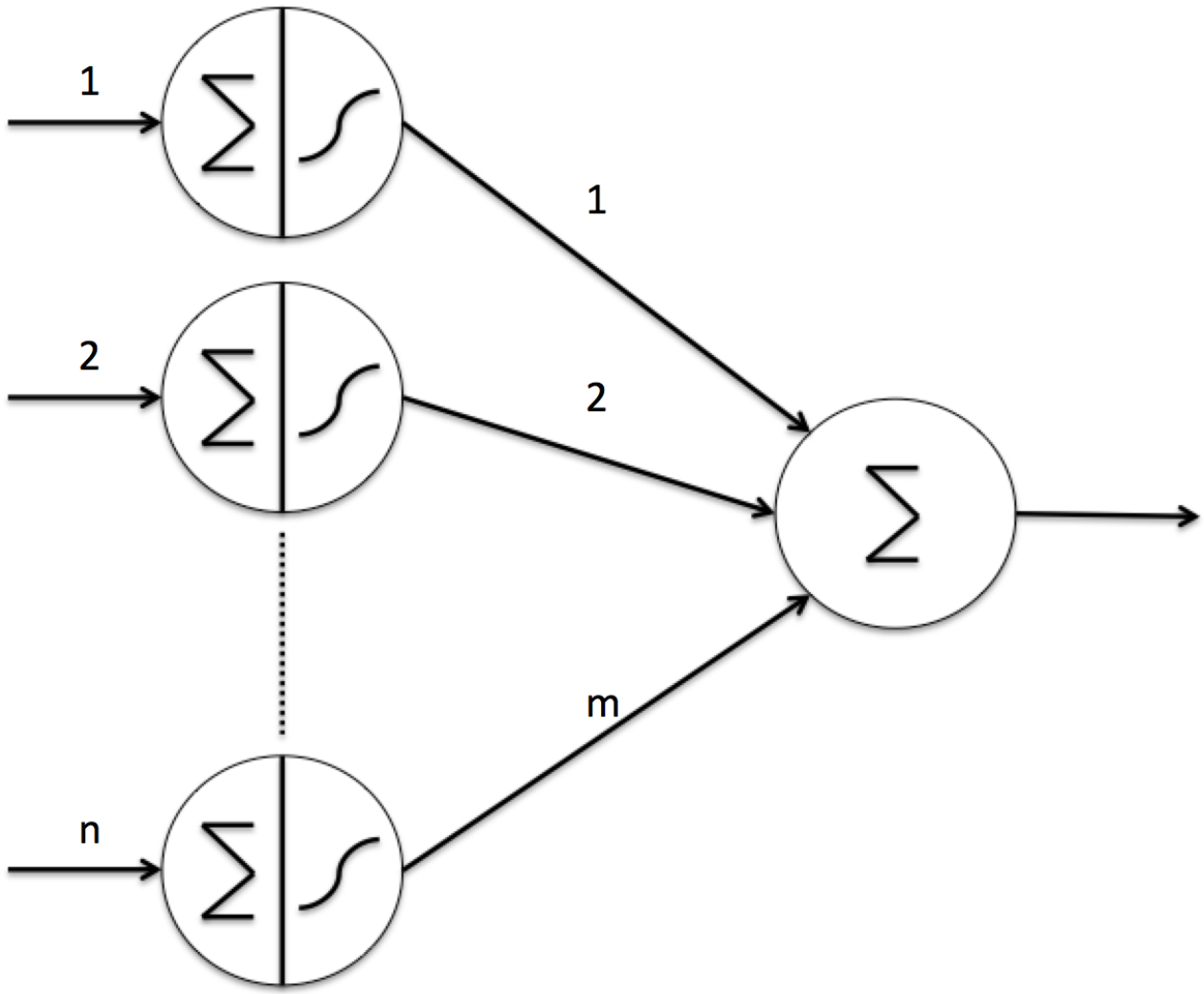
- $m \in \mathbb{N}$
- $\{\beta_i\}_{i=1\ldots m}, \forall \beta_i \in \mathbb{R}$
- $\{\alpha_i\}_{i=1\ldots m}, \forall \alpha_i \in \mathbb{R}$
- $\{w_{ij}\}_{j=1\ldots n, i=1\ldots m}, \forall w_{ij} \in \mathbb{R}$

$\wedge \exists F(x_1, \ldots, x_n) = \sum_{i=1}^{m} \alpha_i \phi\left(\sum_{j=1}^{n} w_{ij} \cdot x_j + \beta_i\right)$:

- $|F(x_1, \ldots, x_n) - f(x_1, \ldots, x_n)| < \epsilon$

[5] Neural Networks and Learning Machines (3rd Edition), Simon O. Haykin

Универсальный аппроксиматор



Демонстрация сходимости нейросети с одним скрытым слоем

- операция XOR - 2layer-net-xor.gif
- бинарная классификация - 2layer-net-ring.gif
- аппроксимация функции sin - 2layer-net-regression-sine.gif
- аппроксимация функции abs - 2layer-net-regression-abs.gif[6]

[6] http://theclevermachine.wordpress.com/2014/09/11/a-gentle-introduction-to-artificial-neural-networks/

Многоснойная нейронная сеть прямого распространения

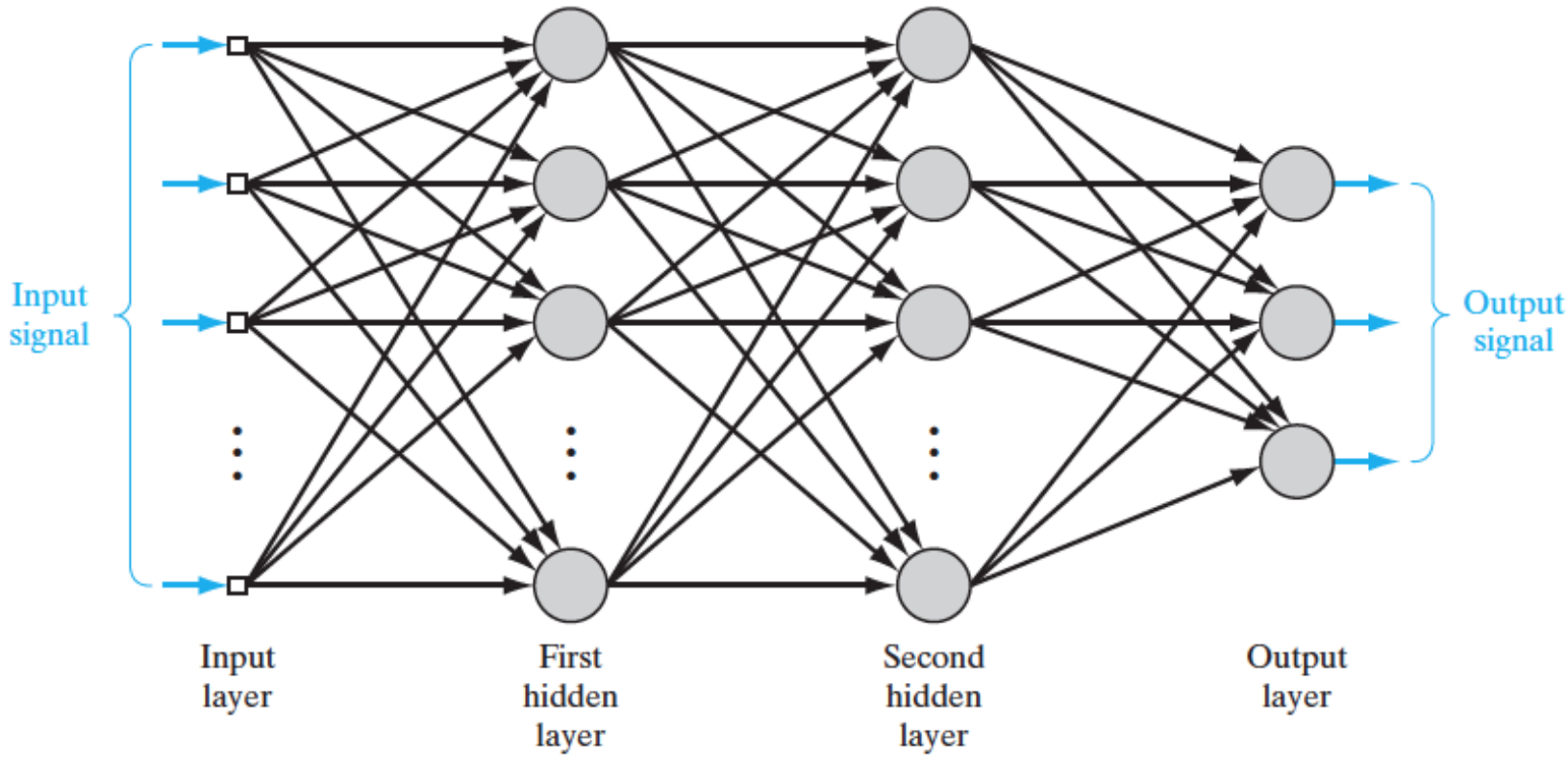


Рис.: Архитектура сети с двумя скрытыми слоями[7]

[7] Neural Networks and Learning Machines (3rd Edition), Simon O. Haykin

Отличие персептрона Румельхарта от персептрона Розенблатта

- Нелинейная функция активации;
- один и более скрытых слоев (до работ Хинтона по ограниченной машине Больцмана, на практике не использовали более двух скрытых слоев, а чаще всего один);
- сигналы на входе и на выходе не обязательно бинарные;
- произвольная архитектура сети;
- ошибка сети интерпретирует в смысле некоторой меры, а не как число неправильных образов в режиме обучения.

Модифицированная модель нейрона МакКаллока-Питтса

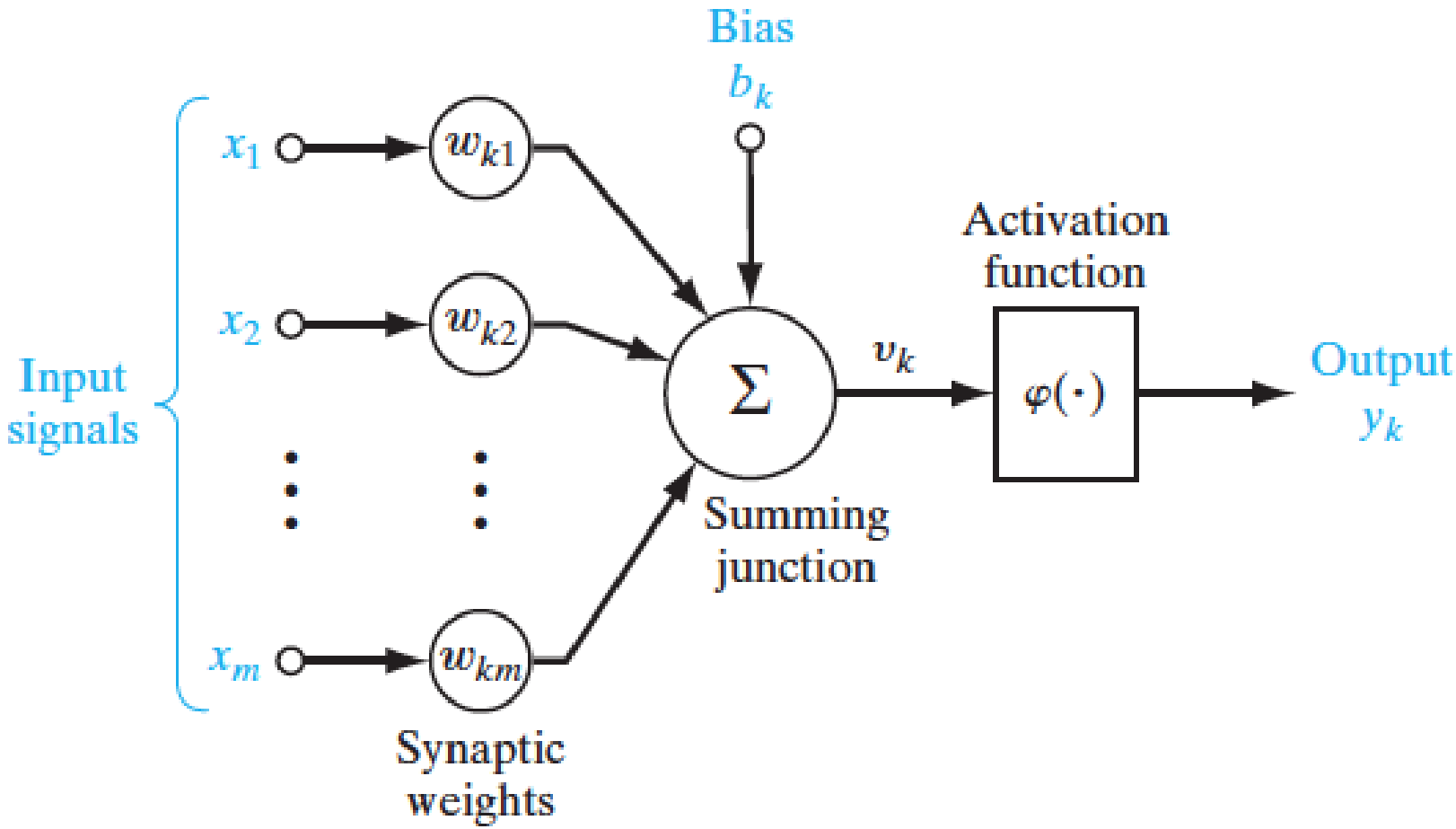


Рис.: Схема искусственного нейрона[8]

[8] Neural Networks and Learning Machines (3rd Edition), Simon O. Haykin

Функция активации

Задача функции активации - ограничить амплитуду выходного значения нейрона; чаще всего для этого используется одна из сигмоидальных (S-образных) функций:

- логистическая функция: $f(z) = \dfrac{1}{1 + e^{-a \cdot z}}, \forall a \in \mathbb{R}$
- гиперболический тангенс: $f(z) = \dfrac{e^{a \cdot x} - e^{-a \cdot x}}{e^{a \cdot x} + e^{-a \cdot x}}, \forall a \in \mathbb{R}$

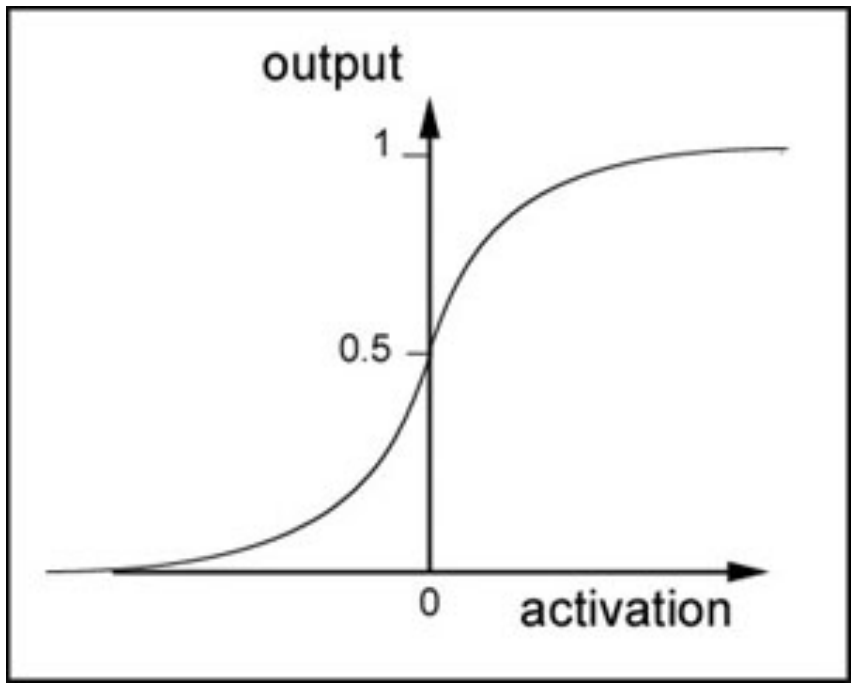
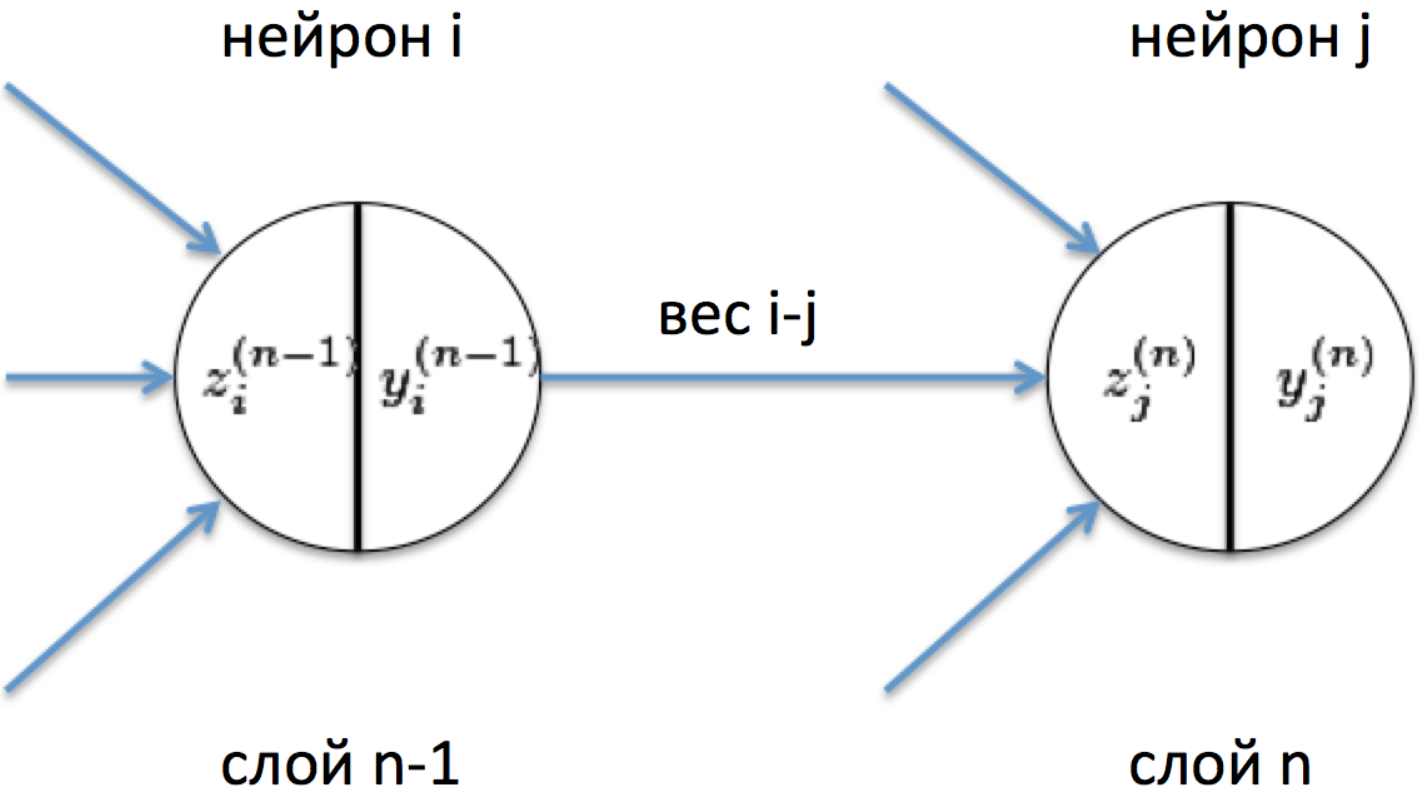


Рис.: Логистический сигмоид

Backprop, обозначения #1

$$z_j^{(n)} = b_j^{(n)} + \sum_{i=1}^{N_{n-1}} w_{ij}^{(n)} x_i^{(n)} = \sum_{i=0}^{N_{n-1}} w_{ij}^{(n)} x_i^{(n)} \quad (1)$$

$$y_k^{(n)} = f_k^{(n)} \left(z_k^{(n)} \right) \quad (2)$$



Backprop, обозначения #2

Обучение с учителем:

- дан набор данных $D = \left\{ \left(x_1, t_1\right), \left(x_2, t_2\right), \ldots, \left(x_{|D|}, t_{|D|}\right) \right\}, x_i \in \mathbb{R}^{N_{\text{INPUT}}}, y_i \in \mathbb{R}^{N_{\text{OUTPUT}}}$
- необходимо построить такое отображение (нейросеть) $f_{\text{NETWORK}} : X \to Y$, которое минимизирует некоторый функционал ошибки $E : \mathbb{R}^{N_{\text{OUTPUT}}} \times \mathbb{R}^{N_{\text{OUTPUT}}} \to \mathbb{R}$, например
  - Евклидово расстояние для задачи регрессии
  - логарифм функции правдоподобия многомерного распределения Бернулли для задачи классификации среди пересекающихся классов
  - кросс-энтропия для задачи классификации среди *не*пересекающихся классов

Градиентный спуск, #1

Алгоритм backprop - это модификация классического градиентного спуска. Параметрами модели являются только веса всех нейронов сети:

$$\delta_{ij}^{(n)} = -\eta \frac{\partial E\left(\vec{y}^{(n)}, \vec{t}\right)}{\partial w_{ij}^{(n)}} \tag{3}$$

- η - скорость обучения (спуска, learning rate)
- $\vec{y}^{(n)}$ - вектор выходов нейросети (выходы последнего слоя)
- $\vec{t}$ - ожидаемые выходы нейросети для текущего примера

Градиентный спуск, #2

- $\dfrac{\partial E}{\partial w_{ij}^{(n)}} = \dfrac{\partial E}{\partial z_j^{(n)}} \dfrac{\partial z_j^{(n)}}{\partial w_{ij}^{(n)}}$
- $\dfrac{\partial z_j^{(n)}}{\partial w_{ij}^{(n)}} = \sum_i \dfrac{\partial w_{ij}^{(n)} x_i^{(n-1)}}{\partial w_{ij}^{(n)}} = x_i^{(n-1)}$

В итоге получим:

$$\frac{\partial E}{\partial w_{ij}^{(n)}} = x_i^{(n-1)} \frac{\partial E}{\partial z_j^{(n)}} \tag{4}$$

Градиентный спуск, выходной слой

$$\frac{\partial E}{\partial w_{ij}^{(n)}} = x_i^{(n-1)} \frac{\partial E}{\partial z_j^{(n)}} = x_i^{(n-1)} \frac{\partial E}{\partial y_j^{(n)}} \frac{\partial y_j^{(n)}}{\partial z_j^{(n)}} \tag{5}$$

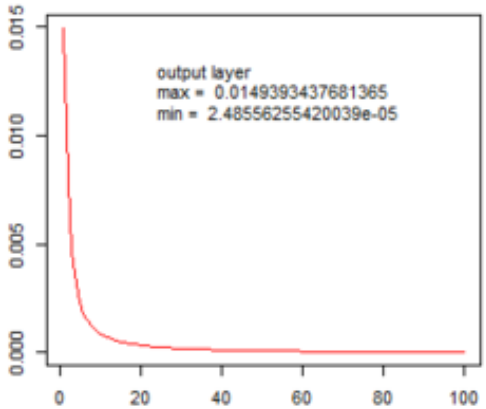
Таким образом при условии дифференцируемости целевой функции и функции активации, вычисление градиента любого из весов выходного слоя становится легко решаемой задачей.

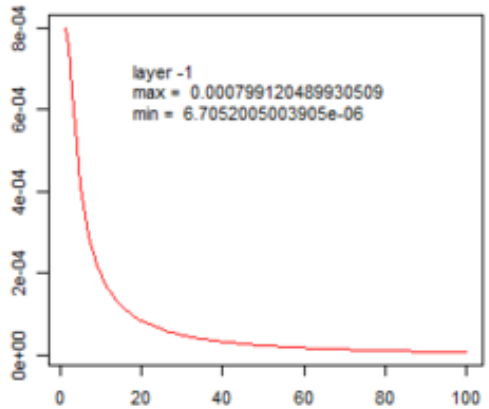
Градиентный спуск, любой скрытый слой

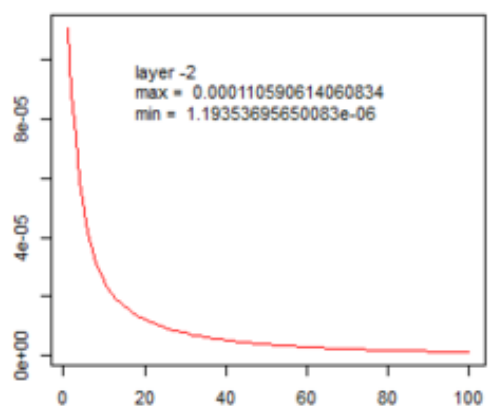
$$
\begin{aligned}
\frac{\partial E}{\partial w_{ij}^{(n)}} &= x_i^{(n-1)} \frac{\partial E}{\partial z_j^{(n)}} \\
&= x_i^{(n-1)} \sum_k \frac{\partial E}{\partial z_k^{(n+1)}} \frac{\partial z_k^{(n+1)}}{\partial z_j^{(n)}} \\
&= x_i^{(n-1)} \sum_k \frac{\partial E}{\partial z_k^{(n+1)}} \frac{\partial z_k^{(n+1)}}{\partial y_j^n} \frac{\partial y_j^n}{z_j^n} \\
&= x_i^{(n-1)} \sum_k w_{ik}^{(n+1)} \frac{\partial E}{\partial z_k^{(n+1)}} \frac{\partial y_j^n}{\partial z_j^n}
\end{aligned}
$$

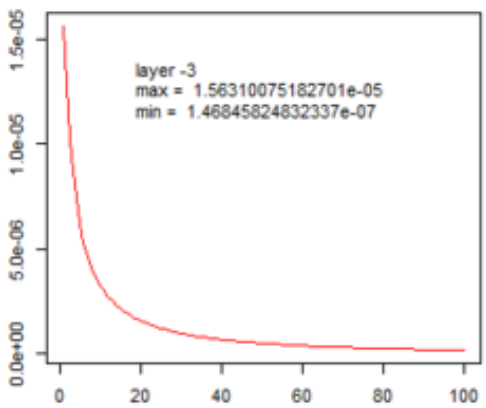
Паралич сети, эксперимент

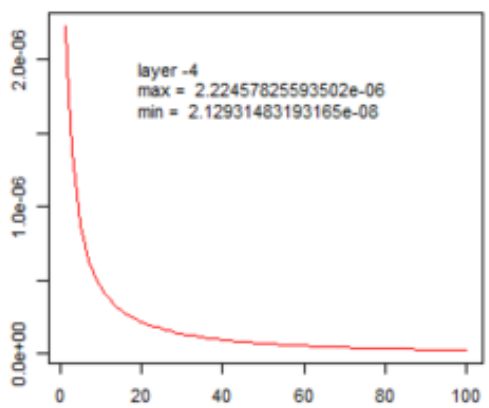
input = 841	layer -5	layer -4	layer -3	layer -2	layer -1	output
neurons	100	100	100	100	100	26

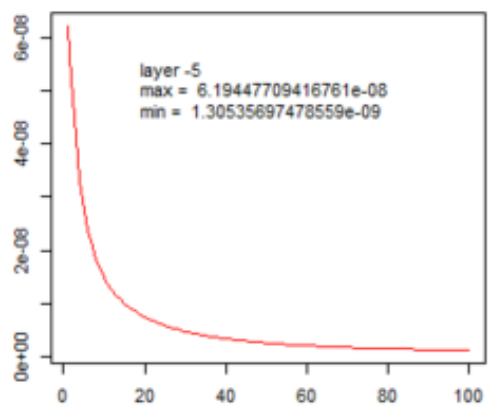




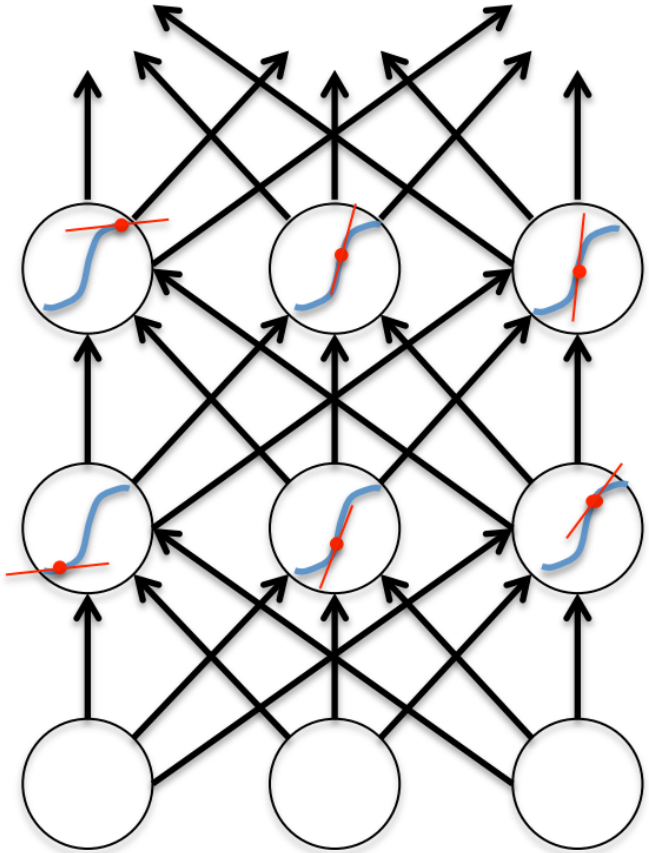








Backprop, прямой проход



Прямой проход в нейросети[9]

- $y_k^{(n)} = \sigma_k^{(n)} \left(\sum_{i=0}^{N_{n-1}} w_{ij}^{(n)} x_i^{(n)} \right)$
- выходные значения каждого нейрона лежат в строго заданных пределах

[9] https://class.coursera.org/neuralnets-2012-001/lecture

Backprop, обратный проход, #1

Вспомним формулу градиента для скрытых слоев:

$$\frac{\partial E}{\partial w_{ij}^{(n)}} = x_i^{(n-1)} \sum_k w_{ik}^{(n+1)} \frac{\partial E}{\partial z_k^{(n+1)}} \frac{\partial y_j^n}{\partial z_j^n}$$

Выполнив замену $\delta_k^{(n)} = \frac{\partial E}{\partial z_k^{(n+1)}}$, получим следующее:

$$\frac{\partial E}{\partial w_{ij}^{(n)}} = x_i^{(n-1)} \frac{\partial y_j^n}{\partial z_j^n} \sum_k w_{ik}^{(n+1)} \delta_k^{(n)} = \sum_k c_k \delta_k^{(n)}$$

- в итоге получилась линейная функция от $\delta_k^{(n)}$.
- $\delta_k^{(n)}$ - локальный градиент/ошибка нейрона *(она как раз и распространяется обратно)*

Backprop, обратный проход, #2

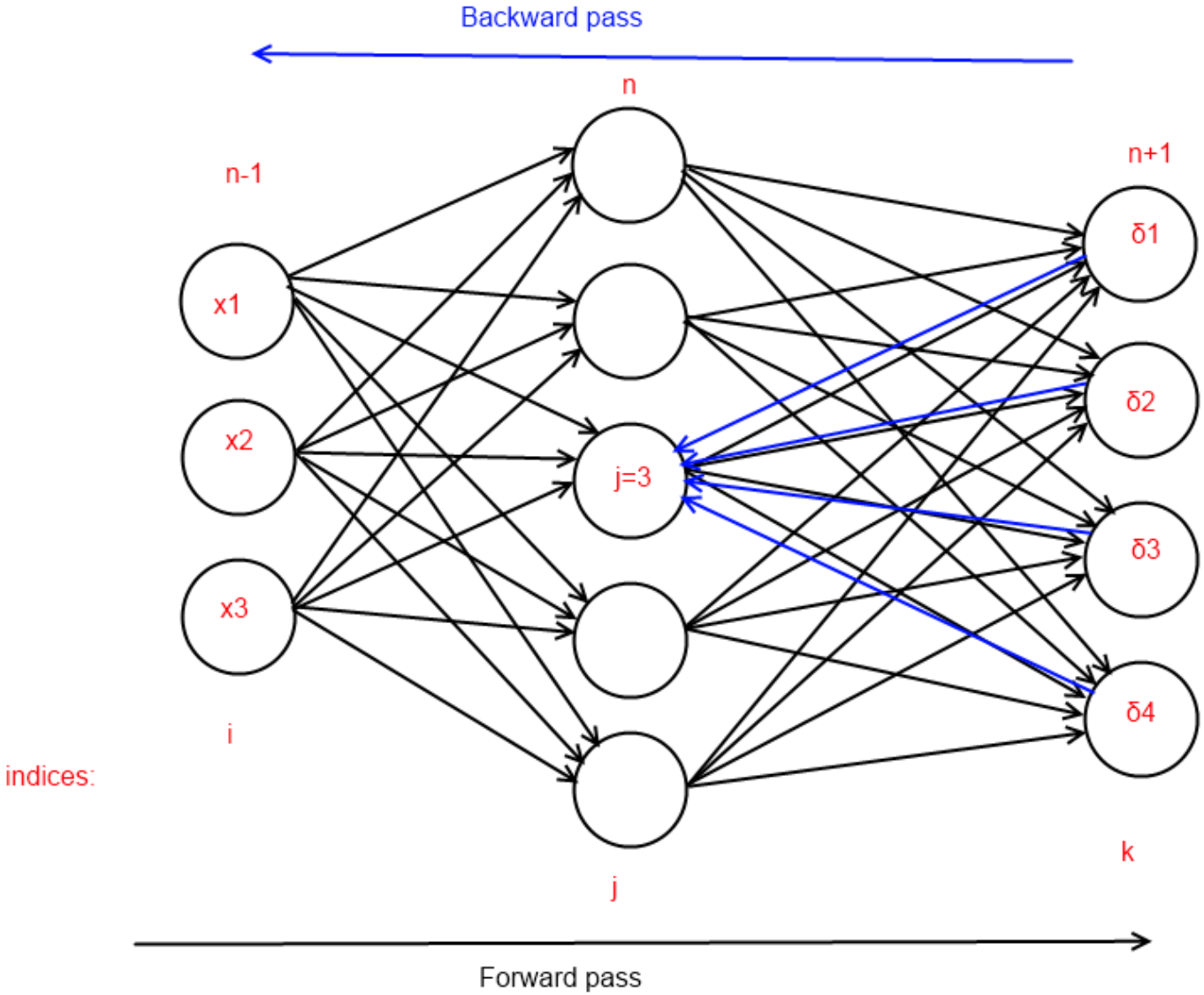


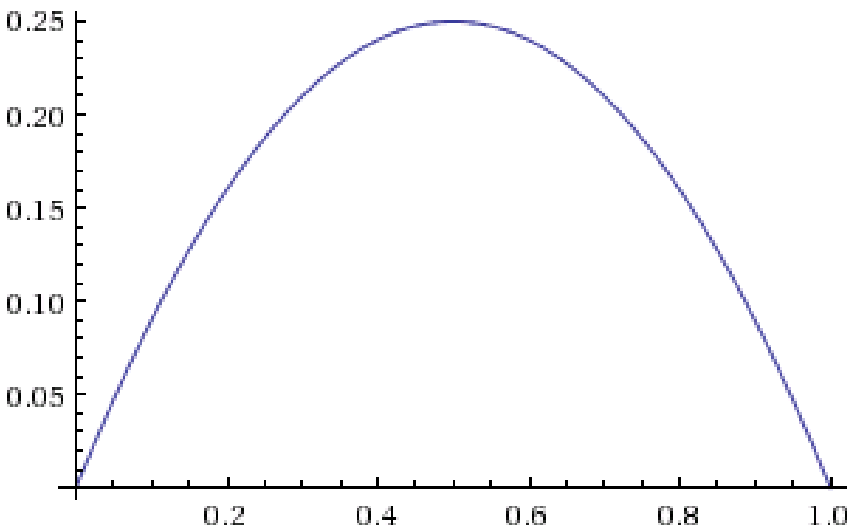
Рис.: Схема прямого (нелинейного) и обратного (линейного) распространения сигнала в сети

Backprop, затухание градиента, #1

Рассмотрим в качестве функции активации функцию логистического сигмоида:

$$
\begin{aligned}
y(z) &= \frac{1}{1+e^{-z}} \\
\frac{\partial y(z)}{\partial z} &= y(z) \cdot (1 - y(z))
\end{aligned}
$$

Построим график значений производной:



- максимум равен $\sigma_{\text{MAX}} = 0.25 = \frac{1}{4}$

Backprop, затухание градиента, #2

Рассмотрим простую сеть:

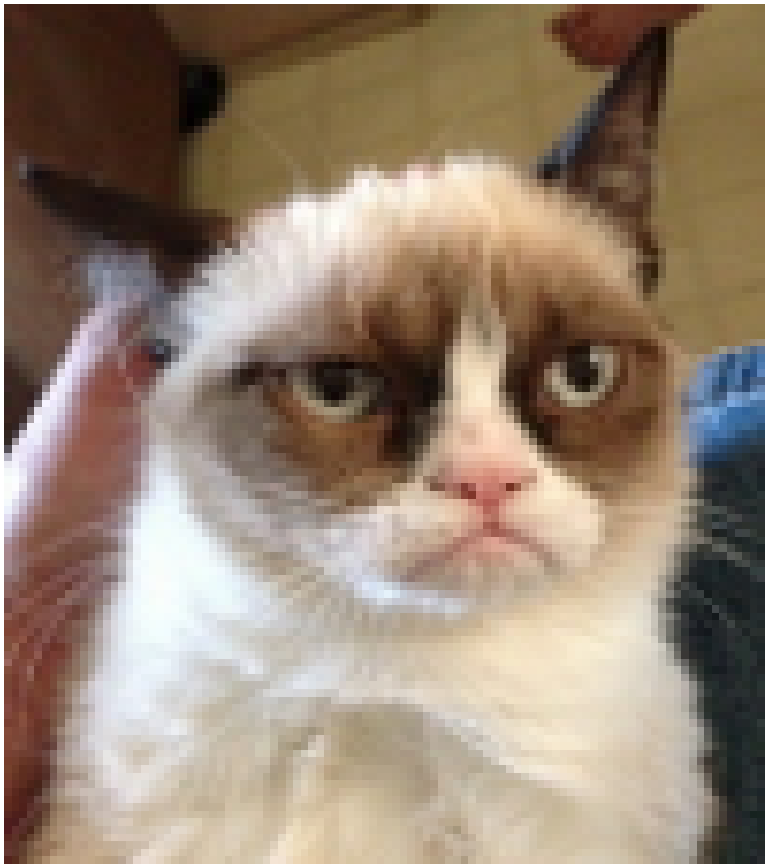
Вычислим градиенты весов для $E(\vec{y}, \vec{t}) = \frac{1}{2}\sum_j (y_j - t_j)^2$:

$$
\begin{aligned}
\frac{\partial E}{\partial w_5} &= y_4(y_5 - t)y_5(1 - y_5) \leq y_4(y_5 - t)\frac{1}{4} \\
\frac{\partial E}{\partial w_4} &= y_3 w_5(y_5 - t)y_4(1 - y_4)y_5(1 - y_5) \leq y_3 w_5(y_5 - t)\frac{1}{4^2} \\
\frac{\partial E}{\partial w_3} &\leq y_2 w_4 w_5(y_5 - t)\frac{1}{4^3} \\
\frac{\partial E}{\partial w_2} &\leq y_1 w_3 w_4 w_5(y_5 - t)\frac{1}{4^4} \\
\frac{\partial E}{\partial w_1} &\leq x w_2 w_3 w_4 w_5(y_5 - t)\frac{1}{4^5}
\end{aligned}
$$

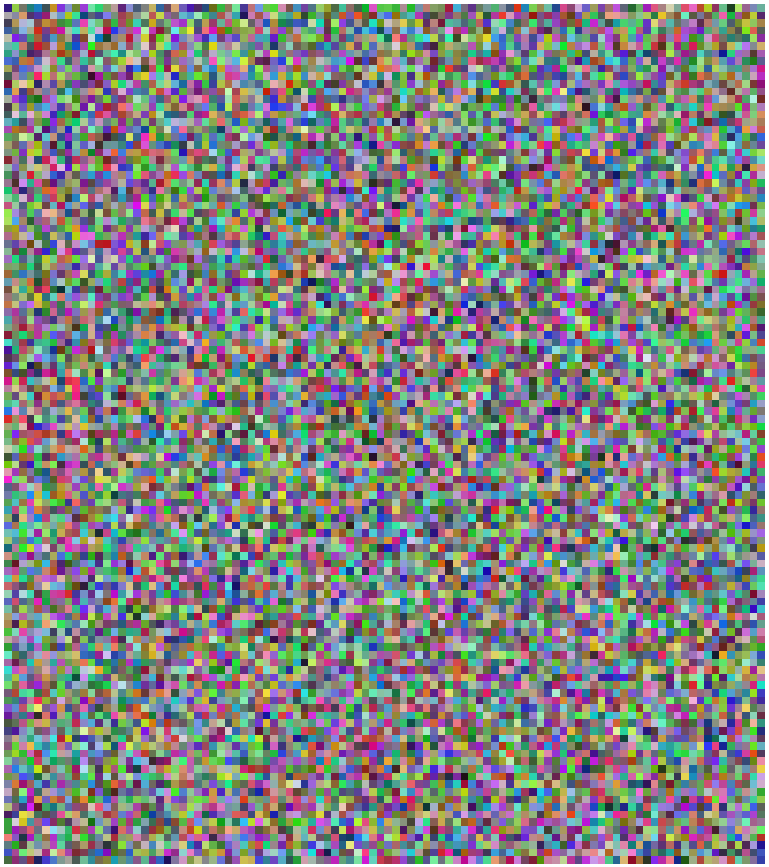
Backprop, паралич сети, выводы

- значение градиента затухает экспоненциально в принципе $\Rightarrow$ сходимость замедляется
- при малых значениях весов этот эффект усиливается
- при больших значения весов значение градиента может экспоненциально возрастать $\Rightarrow$ алгоритм расходится
- не глубокие сети не сильно страдают от этого

Проблема паралича сети, визуализация



(a) Исходное изображение



(b) Образ в первом скрытом слое

Рис.: Оригинал и его образ в первом скрытом слое нейронной сети при количестве нейронов нем равном размерности входного образа при инициализации весов $w_{ij} \sim \mathcal{N}(0, 0.01)$

Обучение без учителя

When we're learning to see, nobody's telling us what the right answers are — we just look. Every so often, your mother says "that's a dog", but that's very little information. You'd be lucky if you got a few bits of information — even one bit per second — that way. The brain's visual system has 10^{14} *neural connections. And you only live for* 10^{9} *seconds. So it's no use learning one bit per second. You need more like* 10^{5} *bits per second. And there's only one place you can get that much information: from the input itself.*[10]

[10] Geoffrey Hinton, 1996 (quoted in (Gorder 2006))

Статистическая механика, #1

Представим некоторую физическую систему с множеством степеней свободы, которая может находится в одном из множества состояний с некоторой вероятностью, а каждому такому состоянию состоянию соответствует некоторая энергия всей системы:

- $p_i \geq 0$ - вероятность состояния i
- $\sum_i p_i = 1$
- E_i - энергия системы в состоянии i

Тогда вероятность состояния i будет описываться распределением Больцмана-Гиббса, при условии термодинамического равновесия между системой и окружающей средой:

$$p_i = \frac{1}{Z} \exp\left(-\frac{E_i}{k_B \cdot T}\right) \tag{6}$$

где

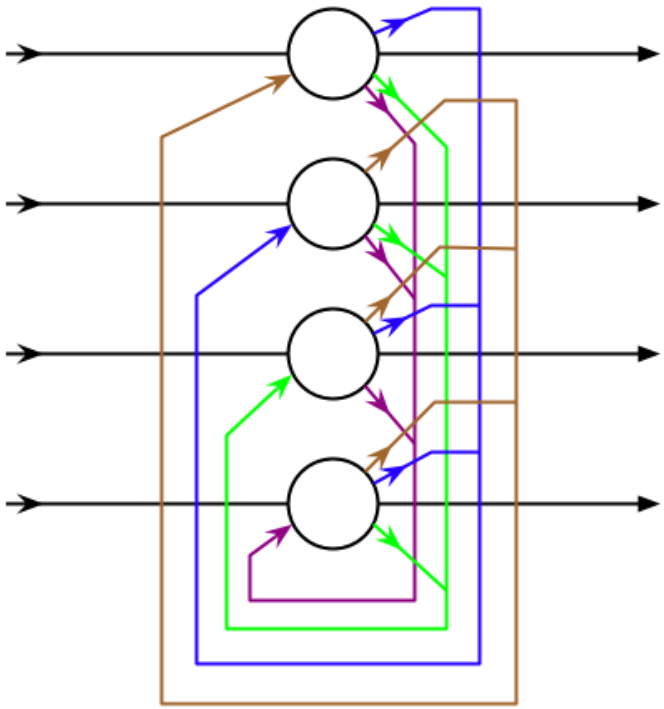
- T - абсолютная температура (K)
- k_B - константа Больцмана (Дж/K)
- $Z = \sum_i \exp\left(-\frac{E_i}{k_B \cdot T}\right)$ - нормализирующая константа (partition function, статсумма)

Статистическая механика, #2

Два важных вывода:

1. состояния с низкой энергией имеют больше шансов возникнуть чем состояния с высокой энергией;
2. при понижении температуры, чаще всего будут возникать состояния из небольшого подмножества состояний с низкой энергией.

Нейросеть Хопфилда



- обратная связь
- пороговая функция активации

Такая сеть (рекуррентная нейронная сеть) может находится как в стабильном состоянии, осциллировать, или даже проявлять признаки детерминированного хаоса.

Хопфилд показал, что при симметричной матрице весов, существует такая функция энергии бинарных состояний системы, что при симуляции система эволюционирует в одно из низко-энергетических состояний.

Нейросеть Хопфилда, энергия системы, #1

$$E = -\sum_i s_i b_i - \sum_{i<j} s_i s_j w_{ij} \tag{7}$$

- s_i - состояние нейрона i
- b_i - смещение нейрона i
- w_{ij} - вес между нейроном i и j

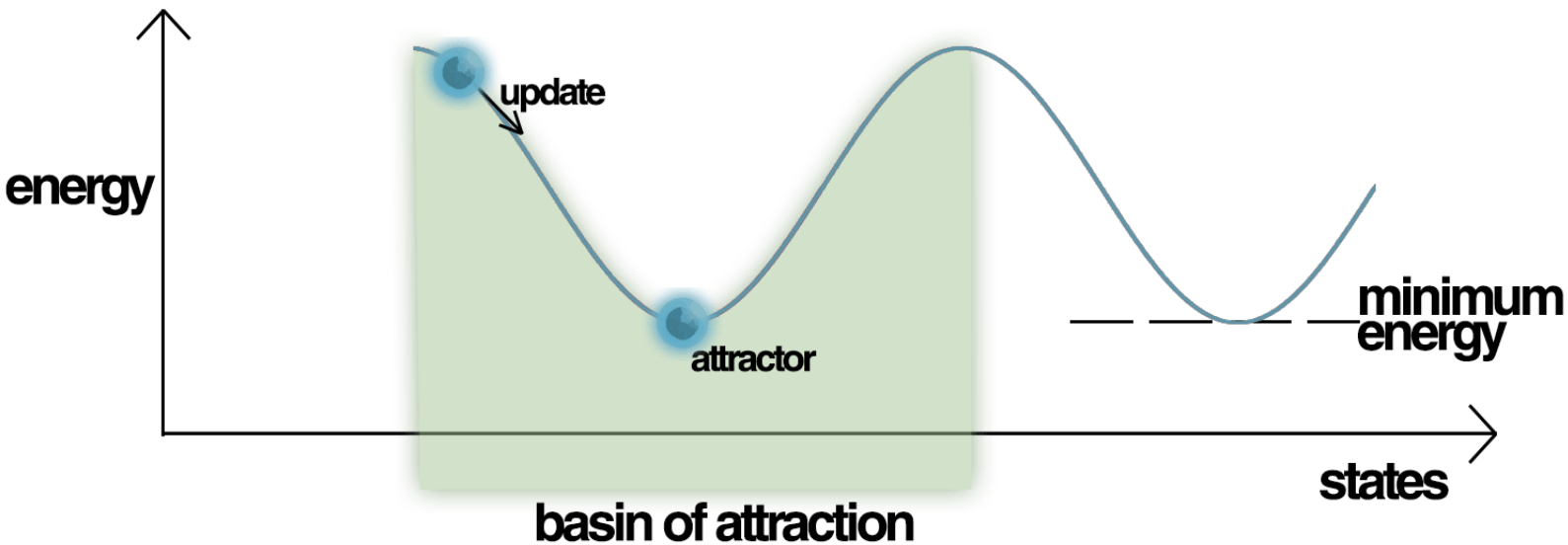


Рис.: Поверхность описываемая энергией сети Хопфилда

Нейросеть Хопфилда, энергия системы, #2

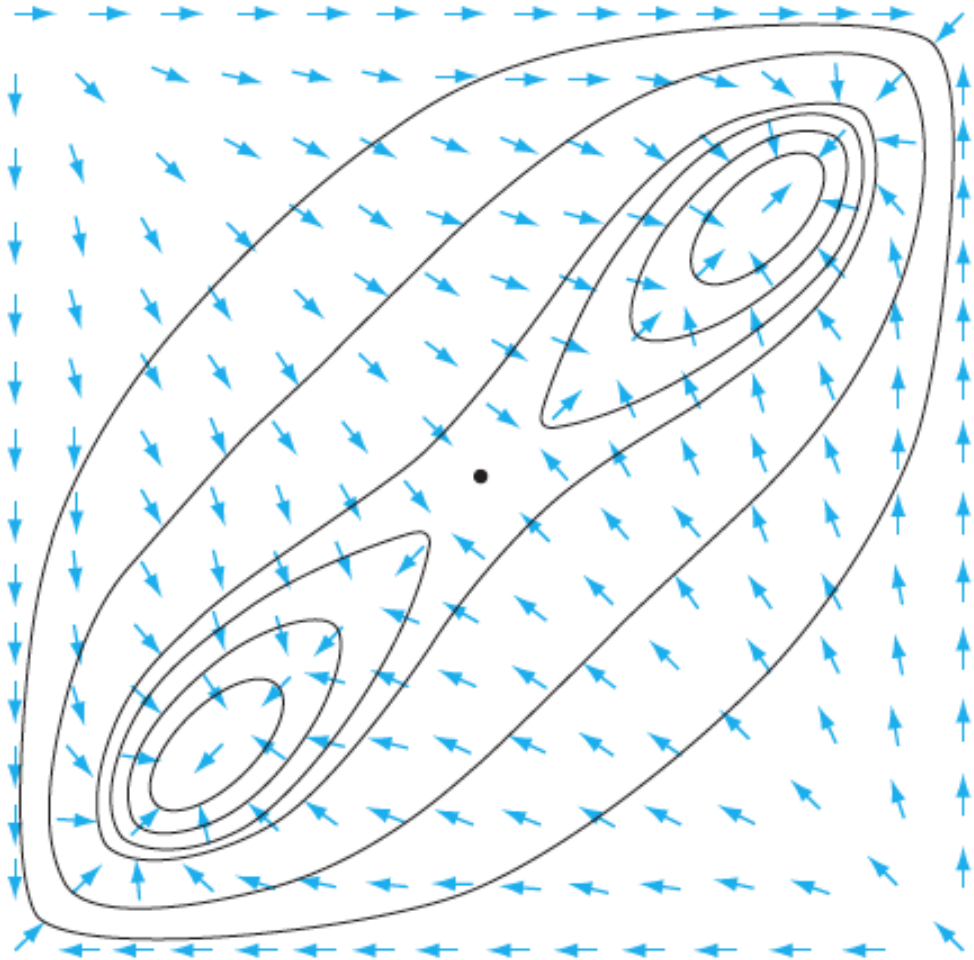
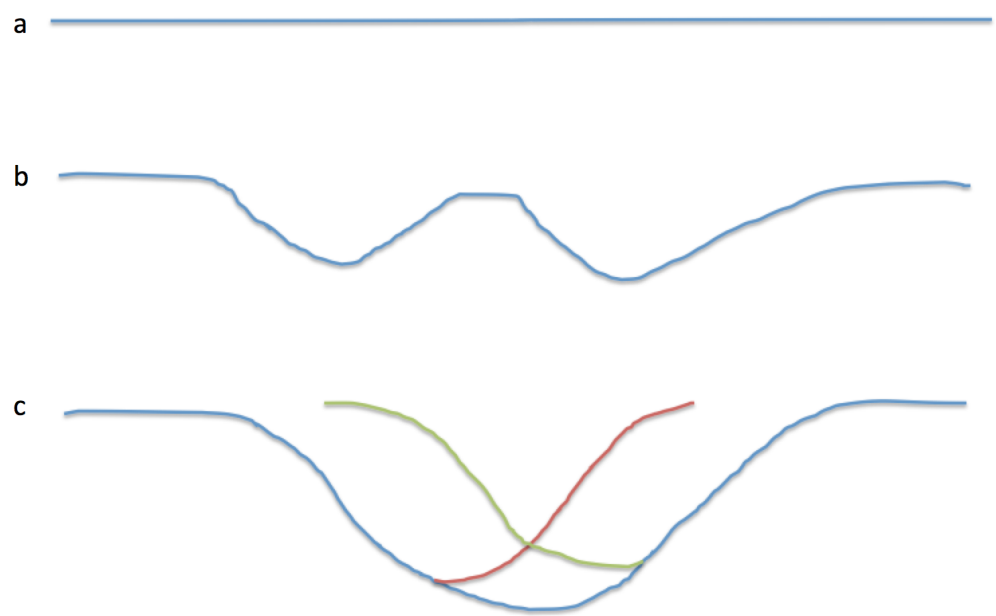


Рис.: Поверхность описываемая энергией сети Хопфилда, два стабильных состояния[11]

[11] Neural Networks and Learning Machines (3rd Edition), Simon O. Haykin

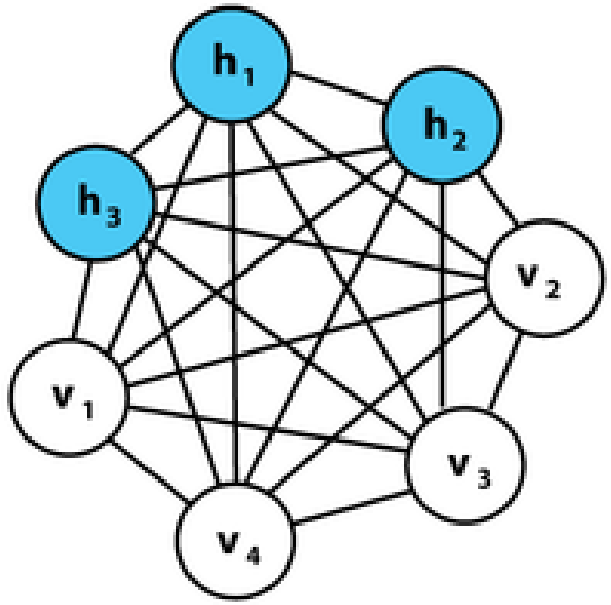
Нейросеть Хопфилда, как ассоциативная память



- a - нет образов в памяти
- b - два образа далеко друг от друга
- c - два образа накладываются друг на друга

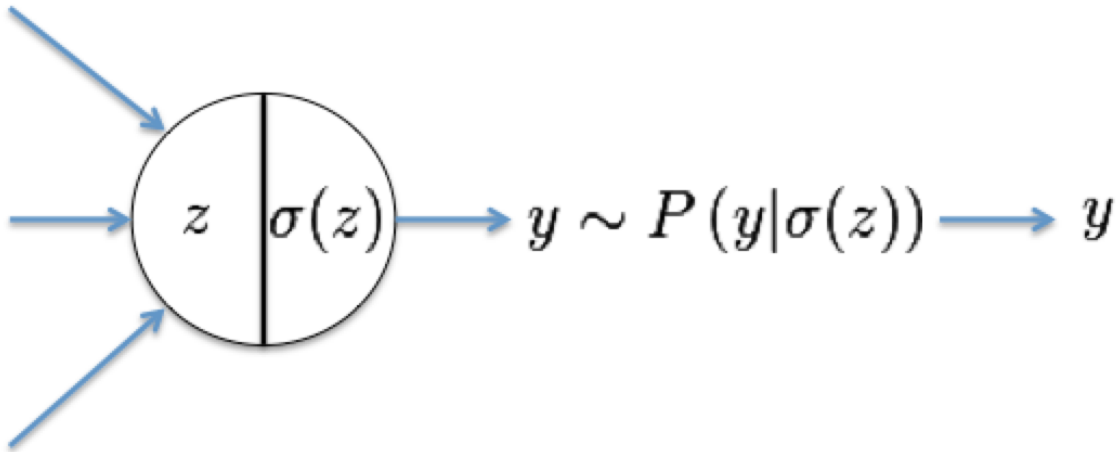
Вместимость $0.15 \cdot N$ на N нейронов.

Машина Больцмана - стохастический генеративный вариант сети Хопфилда



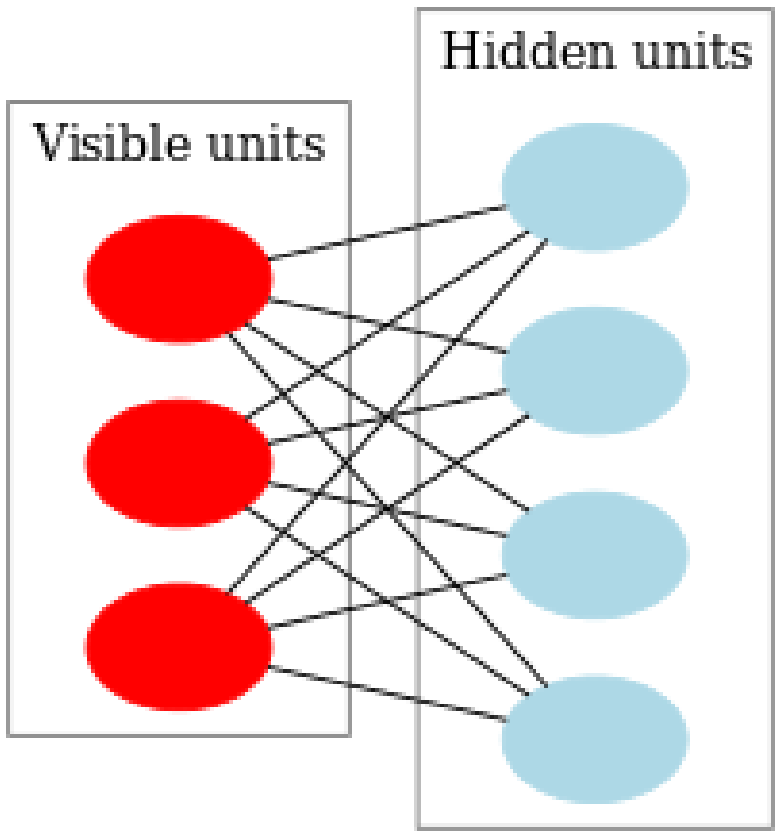
- энергия не изменилась: $E = -\sum_i s_i b_i - \sum_{i<j} s_i s_j w_{ij}$
- симметричная матрица весов $w_{ij} = w_{ji}$, но нет обратных связей: $w_{ii} = 0$
- появились скрытые состояния (система ищет такую конфигурацию скрытых состояний которая лучшим образом описывает видимые состояния)
- $\forall i : s_i \in \{0, 1\}$
- стохастический нейрон

Стохастический нейрон



Ограниченная машина Больцмана

- убираем температуру
- вводим *ограничение* на топологию
- еще аж в 1986 год, гармониум Поля Смоленски



Виды RBM

В зависимости от априрного распределения ассоциированного с видимым и скрытым слоями, различают несколько видов RBM:

- Bernoulli-Bernoulli (binary-binary)
- Gaussian-Bernoulli
- Gaussian-Gaussian
- Poisson-Bernoulli
- и т.д.

Бинарные (Bernoulli-Bernoulli, binary-binary) RBM играют важную роль в глубоком обучении, по аналогии с выводом алгоритма обучения для бинарной ограниченной машины Больцмана, можно вывести аналогичные правила для остальных типов моделей.

RBM, обозначения

- $D = \{\vec{x}_i\}_{i=1\ldots N}$ - множество данных;
- $\vec{v}$, $\vec{h}$ - значения видимых и скрытых нейронов;
- $\vec{a}, \vec{b}, W$ - смещения видимых и скрытых нейронов, и матрица весов;
- n, m - количество видимых и скрытых нейронов;
- $E(\vec{v}, \vec{h}) = -\sum_{i=1}^{n} a_i v_i - \sum_{j=1}^{m} b_j h_j - \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} v_i h_j = -\vec{v}^T \vec{a} - \vec{h}^T \vec{b} - \vec{v}^T W \vec{b}$
- $p(\vec{v}, \vec{h}) = \frac{1}{Z} e^{-E(\vec{v},\vec{h})}$
- $Z = \sum_{r}^{N}\sum_{t}^{M} e^{-E(\vec{v}^{(r)},\vec{h}^{(t)})}$
- $P(\vec{v}) = \sum_{t}^{M} P(\vec{v}, \vec{h}^{(t)}) = \frac{1}{Z}\sum_{t}^{M} e^{-E(\vec{v},\vec{h}^{(t)})}$

Далее значки вектора $\vec{x}$ будут опускаться для простоты.

RBM, функция активации и независимость

Рассмотрим только для скрытого слоя:

$$
\begin{aligned}
P(h_k = 1|v) &= \frac{e^{-E_1}}{e^{-E_1} + e^{-E_0}} \\
&= \frac{1}{1 + e^{E_1 - E_0}} \\
&= \frac{1}{1 + e^{-b_k - \sum_i^n v_i w_{ik}}} \\
&= \sigma\left(b_k + \sum_{i=1}^{n} v_i w_{ik}\right)
\end{aligned}
$$

$$
\begin{aligned}
P(h|v) &= \prod_{j=1}^{m} P(h_j|v) \\
P(v|h) &= \prod_{i=1}^{n} P(v_i|h)
\end{aligned}
$$

RBM, целевая функция

$$E(\vec{v}, \vec{h}) = -\sum_{i=1}^{n} a_i v_i - \sum_{j=1}^{m} b_j h_j - \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} v_i h_j \quad (8)$$

$$P(\vec{v}) = \frac{1}{Z} \sum_{t}^{M} e^{-E(\vec{v}, \vec{h}^{(t)})} \quad (9)$$

- максимизировать вероятность данных при заданной генеративной модели

RBM, дифференцирование P(v), #1

$$E(\vec{v}, \vec{h}) = -\sum_{i=1}^{n} a_i v_i - \sum_{j=1}^{m} b_j h_j - \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} v_i h_j$$

$$\frac{\partial E(v,h)}{\partial w_{ij}} = -v_i h_j \qquad \frac{\partial e^{-E(v,h)}}{\partial w_{ij}} = v_i h_j e^{-E(v,h)}$$

$$\frac{\partial E(v,h)}{\partial a_i} = -v_i \qquad \frac{\partial e^{-E(v,h)}}{\partial a_i} = v_i e^{-E(v,h)}$$

$$\frac{\partial E(v,h)}{\partial b_j} = -h_j \qquad \frac{\partial e^{-E(v,h)}}{\partial b_j} = h_j e^{-E(v,h)}$$

RBM, дифференцирование P(v), #2

$$Z = \sum_r^N \sum_t^M e^{-E(\vec{v}^{(r)}, \vec{h}^{(t)})}$$

$$\begin{aligned}
\frac{\partial Z}{\partial w_{ij}} &= \sum_r^N \sum_t^M v_i^{(r)} h_j^{(t)} e^{-E(v^{(r)}, h^{(t)})} \\
\frac{\partial Z}{\partial a_i} &= \sum_r^N \sum_t^M v_i^{(r)} e^{-E(v^{(r)}, h^{(t)})} \\
\frac{\partial Z}{\partial b_j} &= \sum_r^N \sum_t^M h_j^{(t)} e^{-E(v^{(r)}, h^{(t)})}
\end{aligned}$$

RBM, дифференцирование P(v), #3

$$\begin{aligned}
\frac{\partial P\left(v^{(k)}\right)}{\partial w_{ij}} &= \frac{1}{Z^2}\left(Z\left(\sum_t^M v_i^{(k)} h_j^{(k)} e^{-E\left(v^{(r)},h^{(t)}\right)}\right)\right. \\
&\quad \left. - \left(\sum_t^M e^{-E\left(v^{(r)},h^{(t)}\right)}\right)\left(\sum_r^N \sum_t^M v_i^{(r)} h_j^{(t)} e^{-E\left(v^{(r)},h^{(t)}\right)}\right)\right) \\
\frac{\partial \ln P\left(v^{(k)}\right)}{\partial w_{ij}} &= \frac{1}{P\left(v^{(k)}\right)} \frac{\partial P\left(v^{(k)}\right)}{\partial w_{ij}}
\end{aligned}$$

RBM, дифференцирование P(v), #4

$$\begin{aligned}
\frac{\partial \ln P\left(v^{(k)}\right)}{\partial w_{ij}} &= \sum_{t}^{M} v_i^{(k)} h_j^{(t)} P\left(h^{(t)} | v^{(k)}\right) - \sum_{r}^{N} \sum_{t}^{M} v_i^{(r)} h_j^{(t)} P\left(h^{(t)}, v^{(k)}\right) \\
&= M\left[v_i^{(k)} h_j\right]_{\text{DATA}} - M\left[v_i h_j\right]_{\text{MODEL}} \\
\frac{\partial \ln P\left(v^{(k)}\right)}{\partial a_i} &= v_i^{(k)} - M\left[v_i\right]_{\text{MODEL}} \\
\frac{\partial \ln P\left(v^{(k)}\right)}{\partial b_j} &= M\left[h_j\right]_{\text{DATA}} - M\left[h_j\right]_{\text{MODEL}}
\end{aligned}$$

RBM, правила обновления

$$
\begin{aligned}
\Delta w_{ij} &= \eta \left(M \left[v_i^{(k)} h_j \right]_{\text{DATA}} - M \left[v_i h_j \right]_{\text{MODEL}} \right) \\
\Delta a_i &= \eta \left(v_i^{(k)} - M \left[v_i \right]_{\text{MODEL}} \right) \\
\Delta b_j &= \eta \left(M \left[h_j \right]_{\text{DATA}} - M \left[h_j \right]_{\text{MODEL}} \right)
\end{aligned}
$$

Алгоритм Contrastive Divergence

- Цель: собрать достаточную статистику для оценки $M[\cdots]_{\text{DATA}}$ и $M[\cdots]_{\text{MODEL}}$

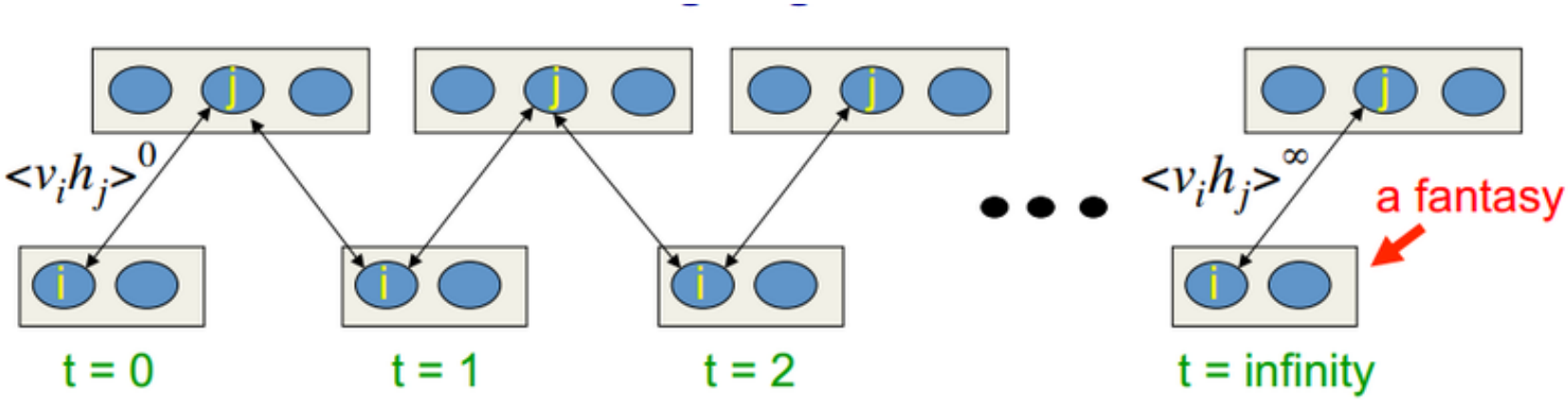


Рис.: Процесс сбора достаточной статистики[12]

- $\Delta w_{ij} = \eta \left(M \left[v_i^{(k)} h_j \right]^{(0)} - M \left[v_i h_j \right]^{(\infty)} \right)$
- $M \left[\cdots \right]^{(0)}$ - позитивная фаза
- $M \left[\cdots \right]^{(\infty)}$ - негативная фаза

[12] https://class.coursera.org/neuralnets-2012-001/lecture

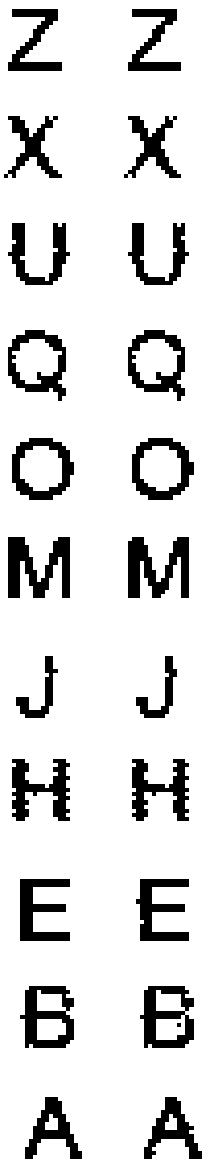
Семплирование по Гиббсу

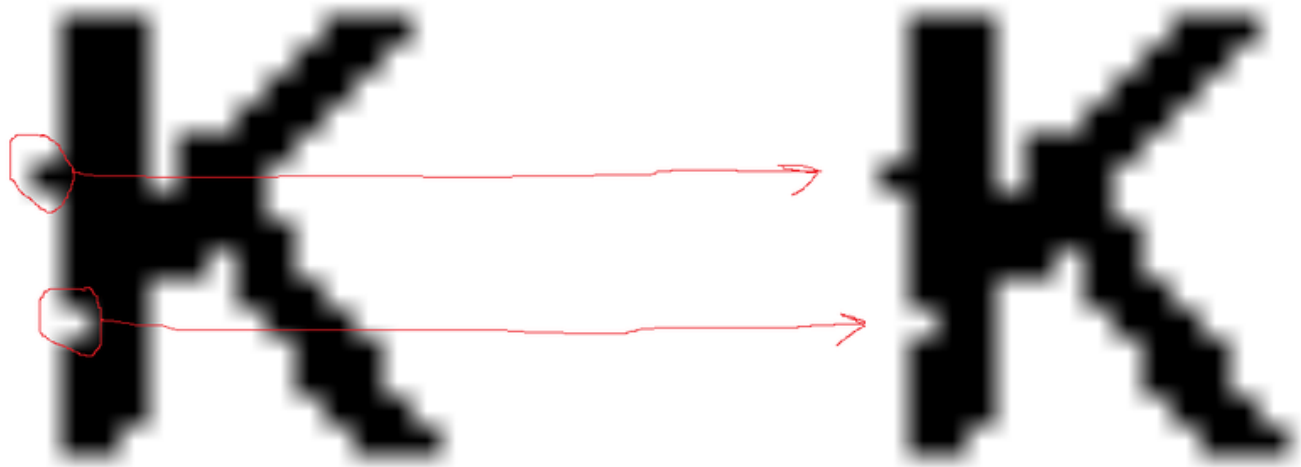
Семплирование по Гиббсу не требуется явно выраженное совместное распределение, а нужны лишь условные вероятности для каждой переменной, входящей в распределение. Алгоритм на каждом шаге берет одну случайную величину и выбирает ее значение при условии фиксированных остальных. Можно показать, что последовательность получаемых значений образуют возвратную цепь Маркова, устойчивое распределение которой является как раз искомым совместным распределением.

- $p_0 = \sum_t^M v_i^{(k)} h_j^{(t)} P\left(h^{(t)} | v^{(k)}\right)$ - это оценка реального распределения основываясь на данных
- $p_{k\to\infty} = \sum_r^N \sum_t^M v_i^{(r)} h_j^{(t)} P\left(h^{(t)}, v^{(r)}\right)$ - оценка реального распределения с помощью семплирования по Гиббсу, которое несомненно когда то сойдется в реальному[13]
- $\frac{\partial \ln P\left(v^{(k)}\right)}{\partial w_{ij}} = p_0 - p_{k\to\infty} = 0$

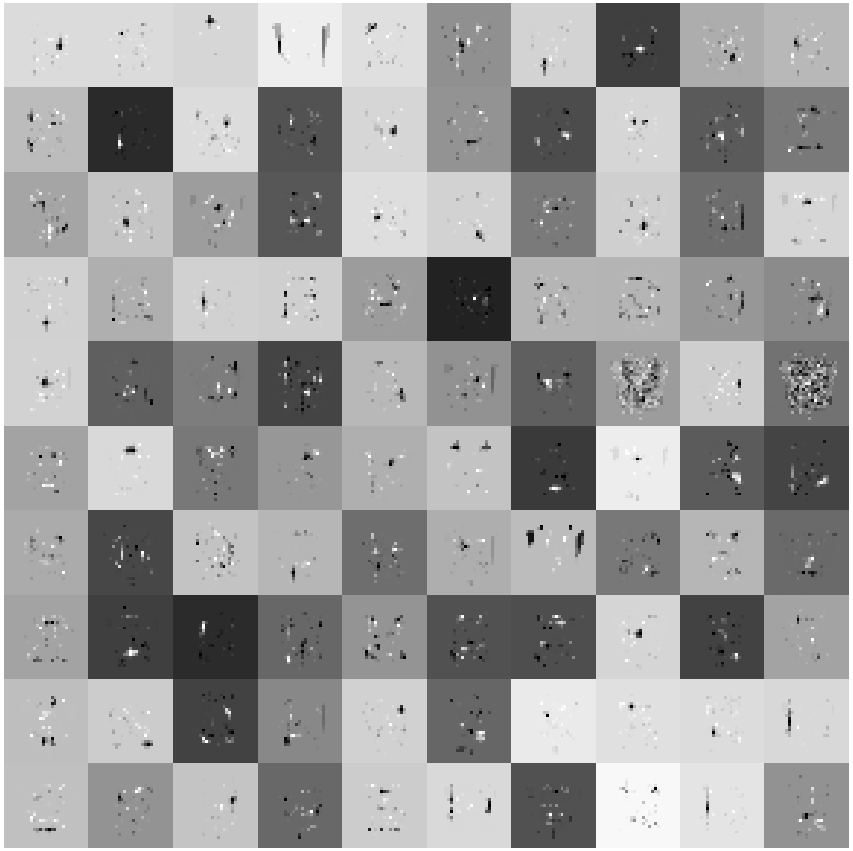
[13] http://www.ini.rub.de/data/documents/tns/masterthesis_janmelchior.pdf

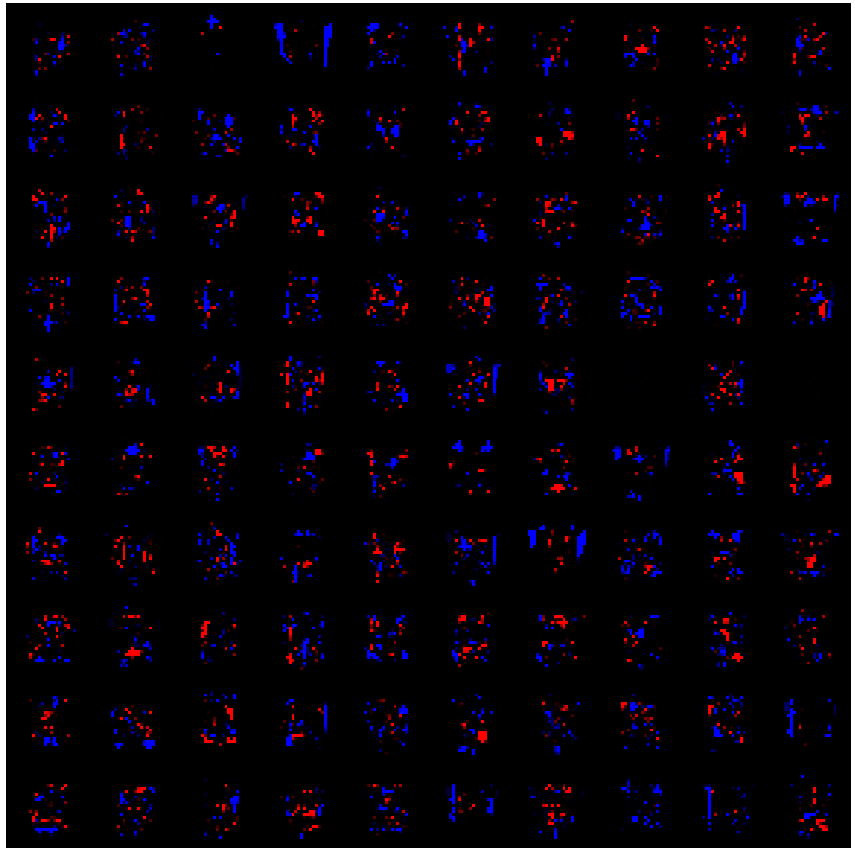
Визуализация восстановленных образов





Визуализация признаков, #1





Визуализация признаков, #2

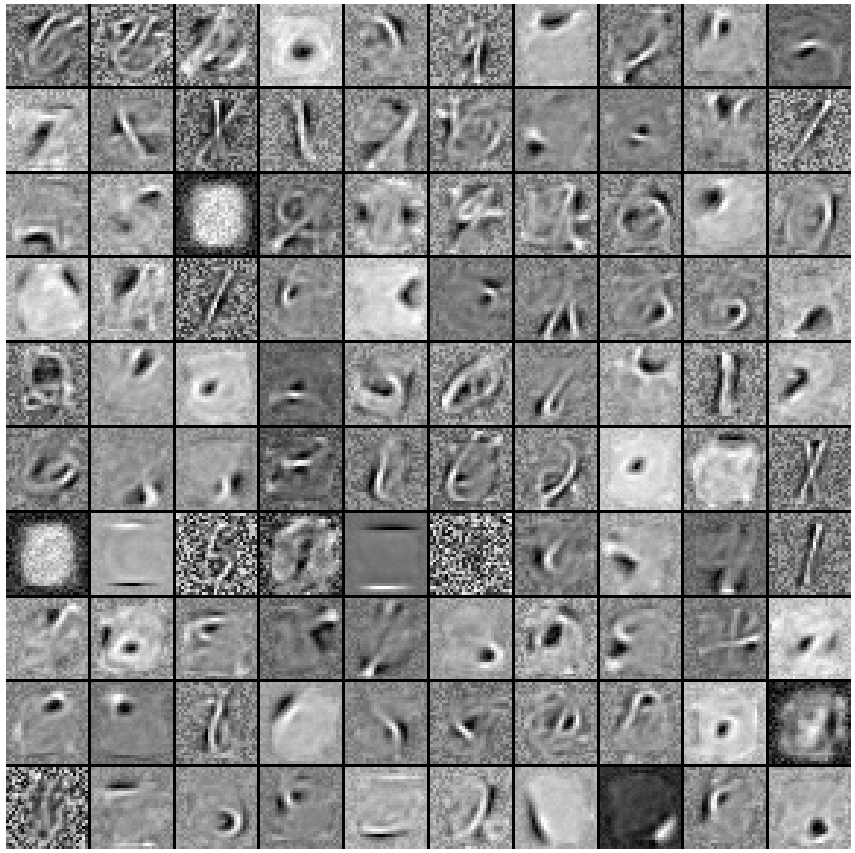
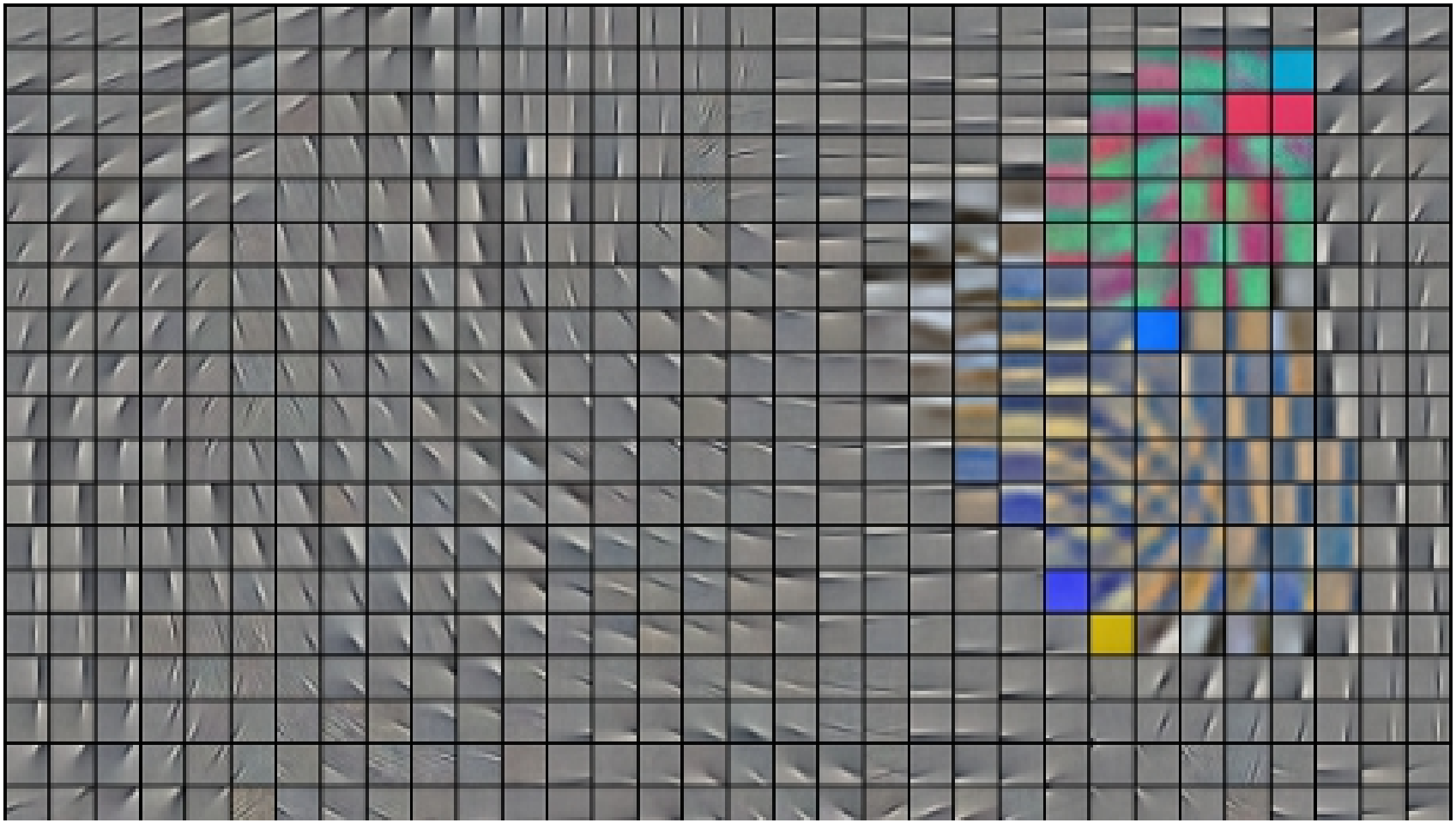


Рис.: Признаки на множестве рукописных цифр MNIST[14]

[14] http://deeplearning.net/

Визуализация признаков, #3



Визуализация признаков, #4

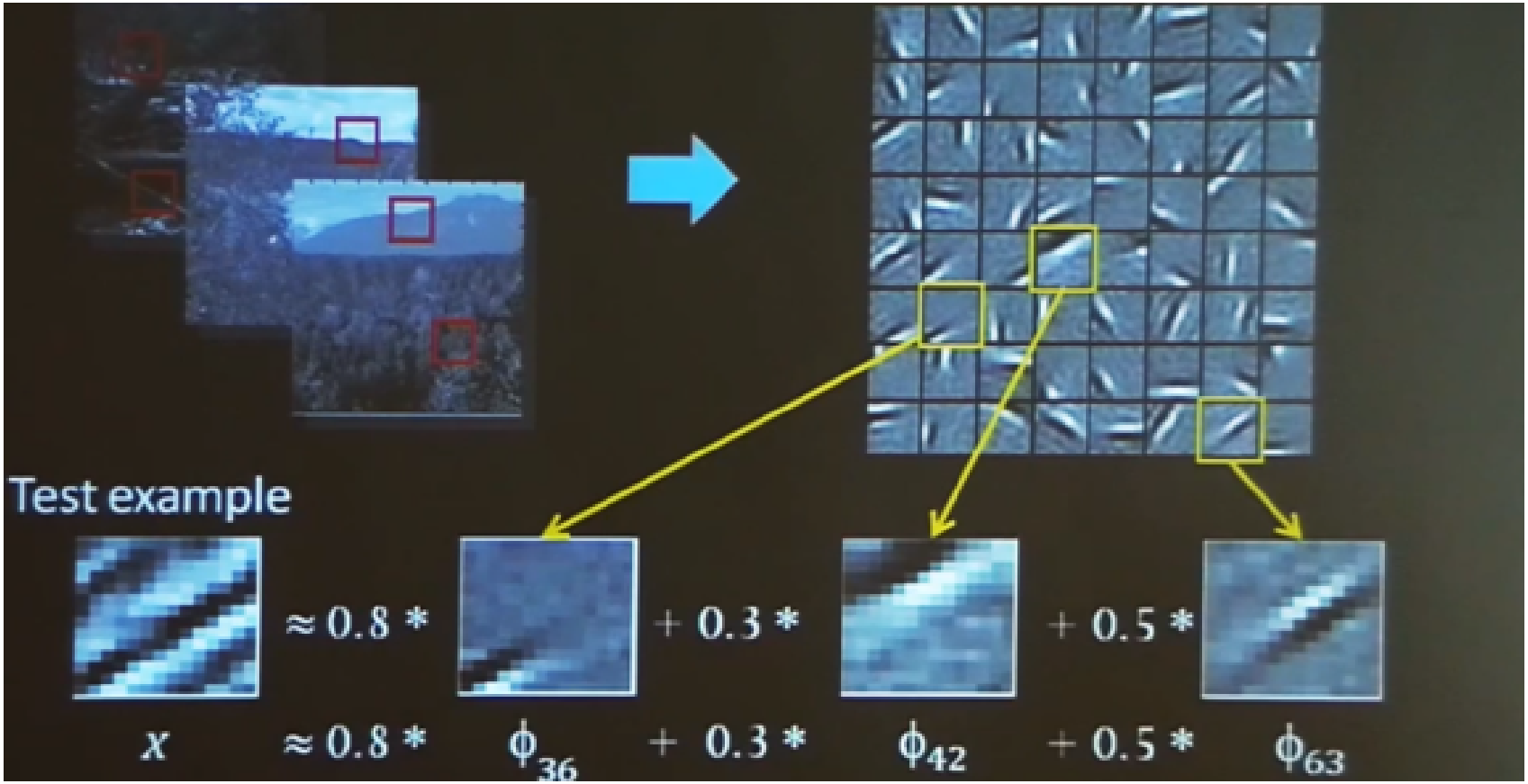
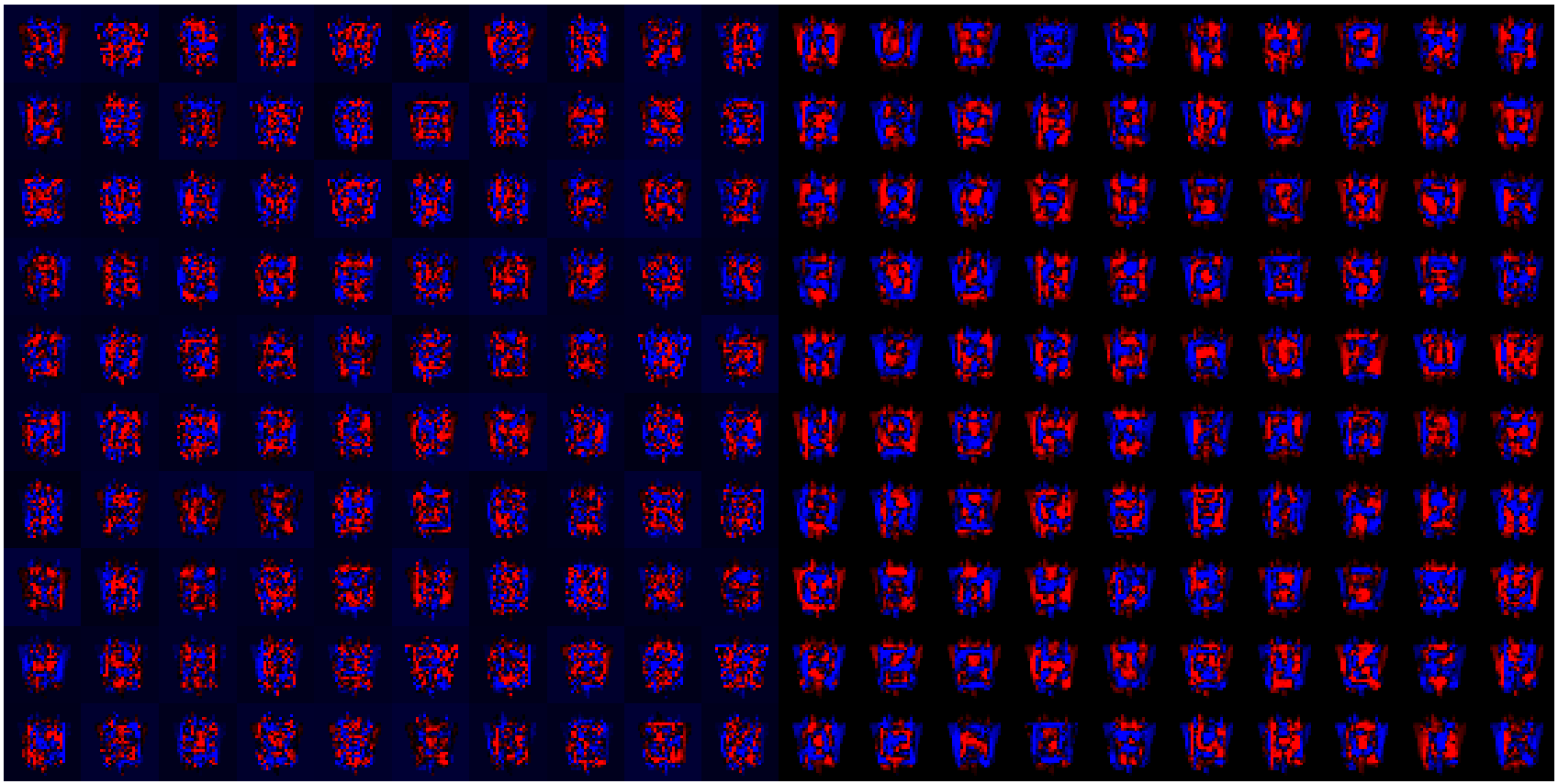


Рис.: RBM как базис[15]

[15] http://cs.stanford.edu/

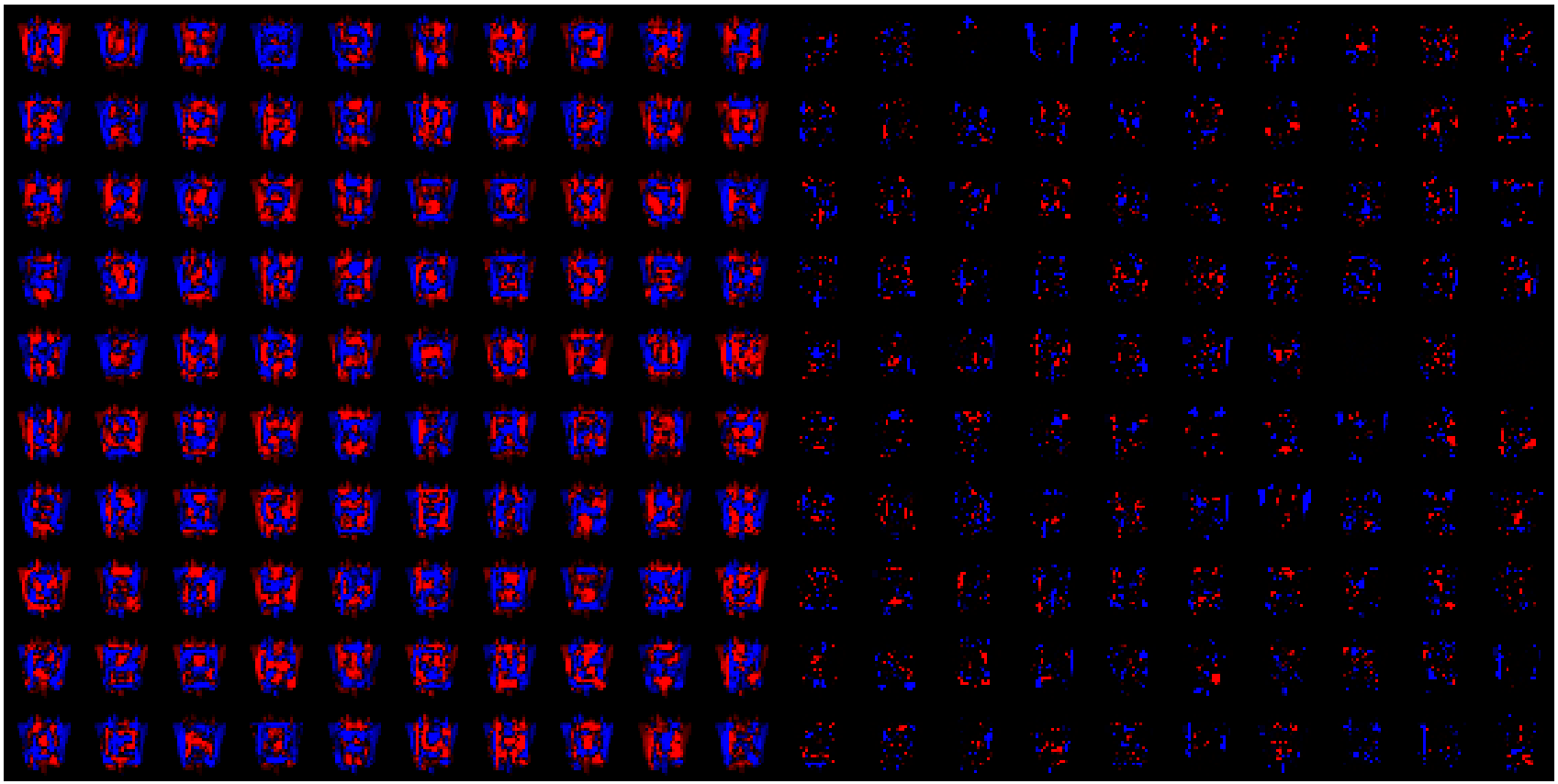
Регуляризация в RBM, #1



(a) RBM, no reg (b) RBM, L2 reg

Рис.: Иллюстрация эффекта регуляризации

Регуляризация в RBM, #2



(a) RBM, L2 reg

(b) RBM, L1 reg

Рис.: Иллюстрация эффекта регуляризации

Критерий остановки

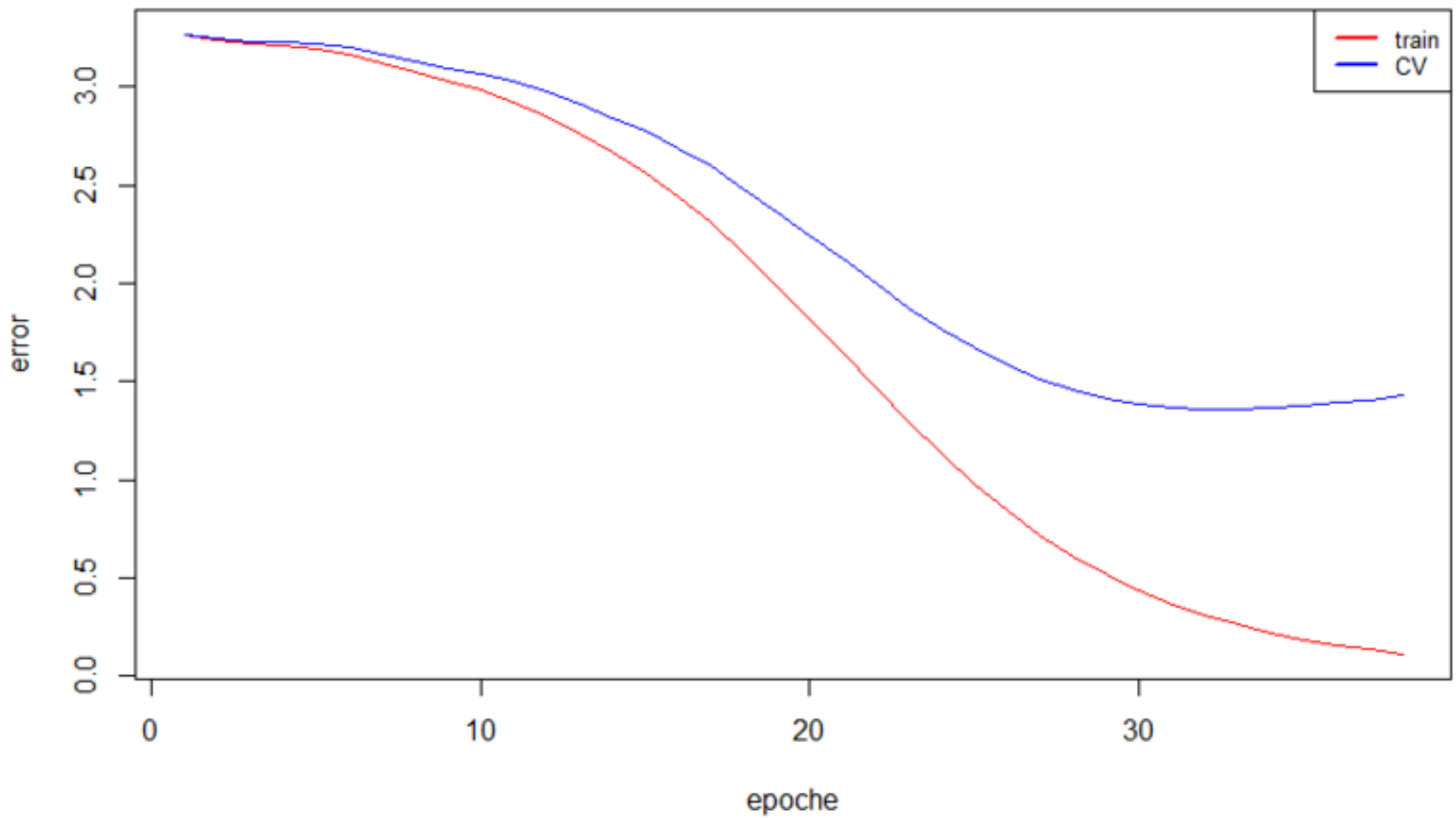


Рис.: Кроссвалидация

Зачем нам глубокие сети?

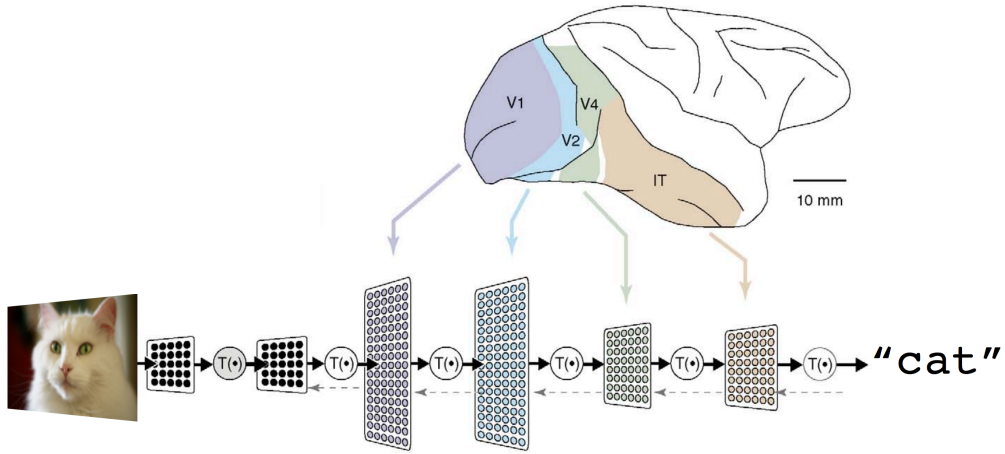


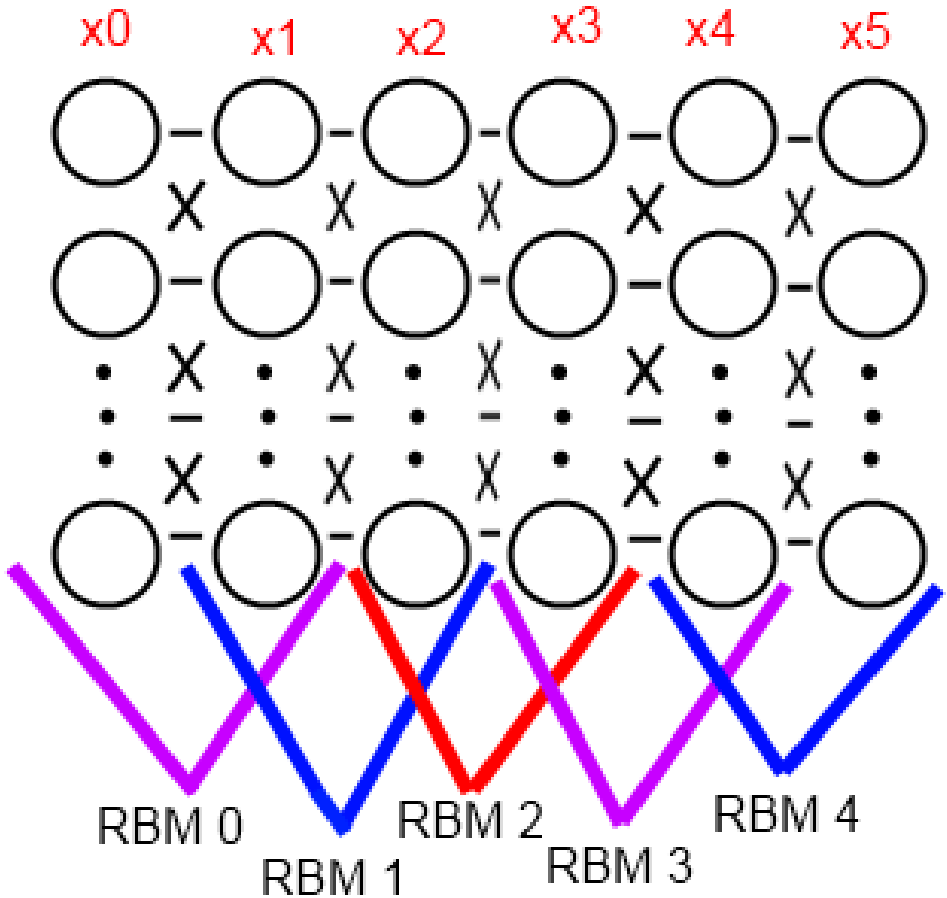
Рис.: Глубокая нейронная сеть[16]

- Согласно стандартной модели зрительной коры головного мозга[17] считается, что каждый следующий нейронный слой выучивает новый уровень абстракции данных [18] (например штрихи, пятна, поверхности, объекты);
- мало того, было показано, что добавление нового слоя в модель улучшает нижнюю вариационную границу логарифма правдоподобия генеративной модели[19].

[16] Из предентации Scale Deep Learning, Jeff Dean
[17] Hubel and Wiesel 1962; Serre et al. 2005; Ranzato et al. 2007
[18] Palmer 1999; Kandel et al. 2000
[19] Learning multiple layers of representation (G. Hinton, 2007)

Решение проблемы паралича сети, идея

- А что если инициализировать веса таким образом, что бы образ оригинального изображения в скрытом пространстве описывал бы прообраз максимально точно?
- Именно это и делает ограниченная машина Больцмана



Жадный алгоритм предобучения

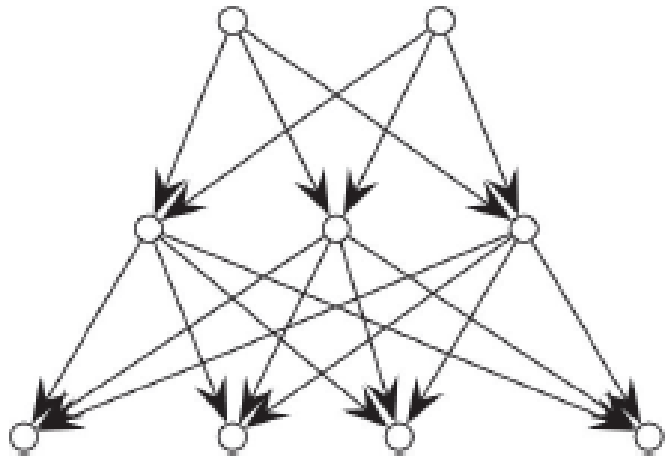
Собственно алгоритм

1. последовательно натренировать каждую пару слоев в глубокой сети (возможно кроме первого и второго скрытого слоя от выходного слоя);
2. осуществить тонкую настройку весов, используя алгоритм обратного распространения ошибки (fine turning).

Некоторые важные преимущества:

- скорость сходимости;
- качество (в смысле выбранной меры);
- возможно использовать не только размеченные образы для обучения с учителем

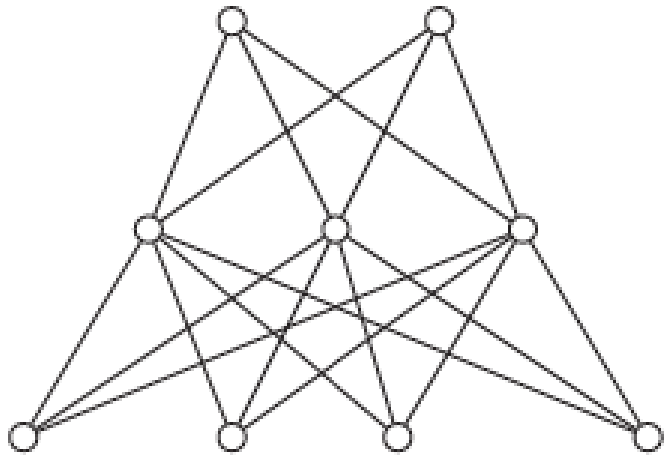
Deep directed network



Для сети с одним видимым слоем и тремя скрытыми функция правдоподобия выглядит следующим образом:

$$p(h_1, h_2, h_3, v|W) = \prod_i \text{Ber}\left(v_i|\sigma\left(h_1^T w_{0i}\right)\right) \cdot \prod_j \text{Ber}\left(h_{1j}|\sigma\left(h_2^T w_{1i}\right)\right) \cdot$$
$$\cdot \prod_k \text{Ber}\left(h_{2k}|\sigma\left(h_3^T w_{2i}\right)\right) \cdot \prod_l \text{Ber}\left(h_{3l}|w_{3l}\right)$$

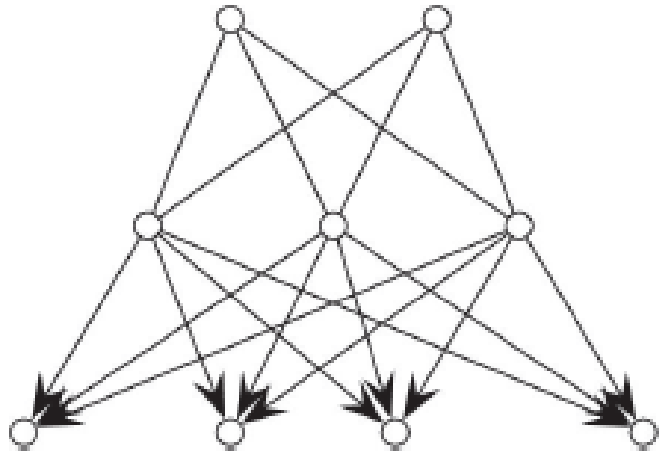
Deep Boltzmann machine



Для сети с одним видимым слоем и тремя скрытыми функция правдоподобия выглядит следующим образом:

$$p(h_1, h_2, h_3, v|W) = \frac{1}{Z} \cdot \exp\left(\sum_{ij} v_i h_{ij} w_{1ij} + \right.$$
$$\left. + \sum_{jk} h_{1j} h_{2k} w_{2jk} + \sum_{kl} h_{2k} h_{3l} w_{3kl}\right)$$

Deep belief network



Для сети с одним видимым слоем и тремя скрытыми функция правдоподобия выглядит следующим образом:

$$p(h_1, h_2, h_3, v|W) = \prod_i \text{Ber}\left(v_i|\sigma\left(h_1^T w_{0i}\right)\right) \cdot \prod_j \text{Ber}\left(h_{1j}|\sigma\left(h_2^T w_{1i}\right)\right) \cdot \cdot \frac{1}{Z} \exp\left(\sum_{kl} h_{2k} h_{3l} w_{3kl}\right)$$

Autoencoder

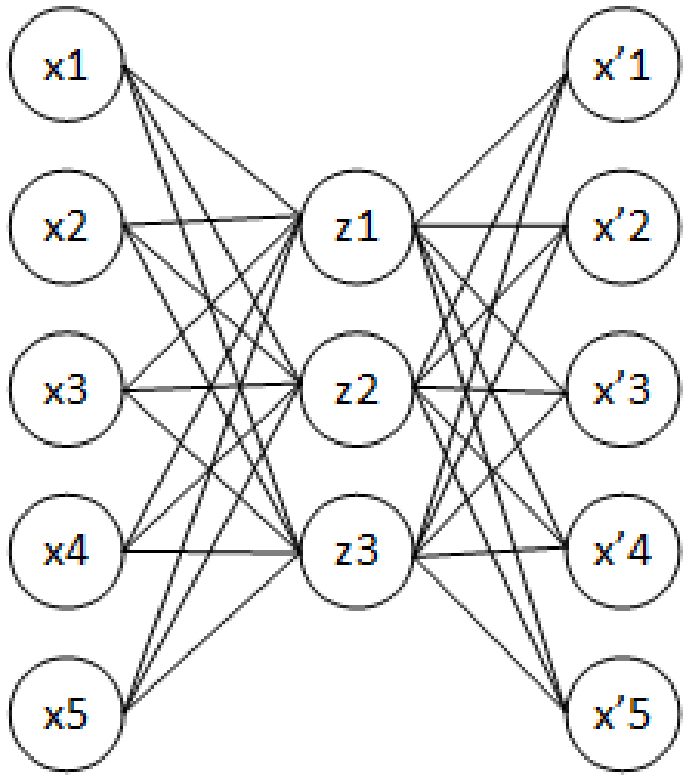


Рис.: Архитектура автоассоциатора: при обучении стремятся получить выходной вектор x' наиболее близким к входному вектору x

Deep autoencoder

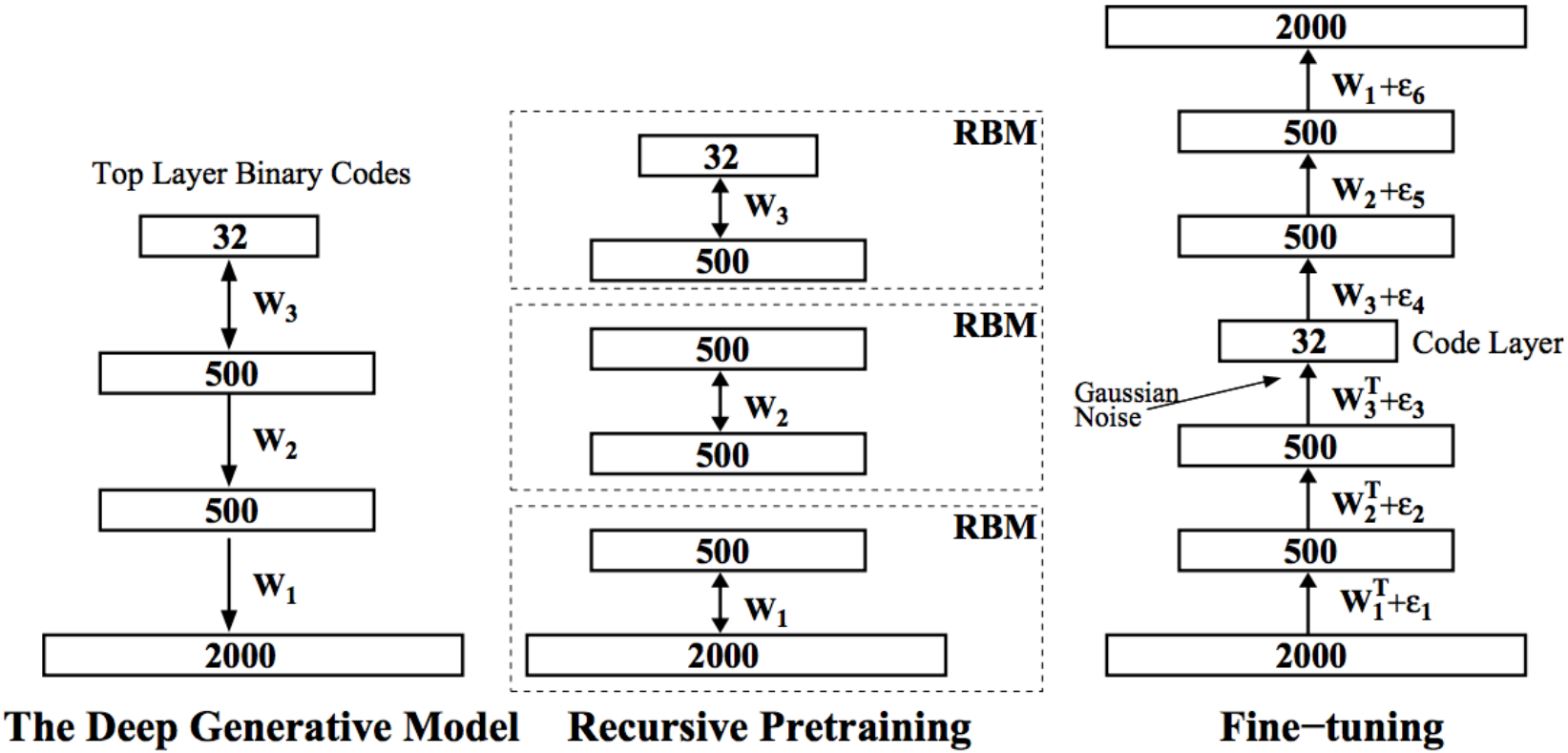


Рис.: Semantic hashing[20]

[20] R. Salakhutdinov, G. Hinton

Discriminative Restricted Boltzmann Machines

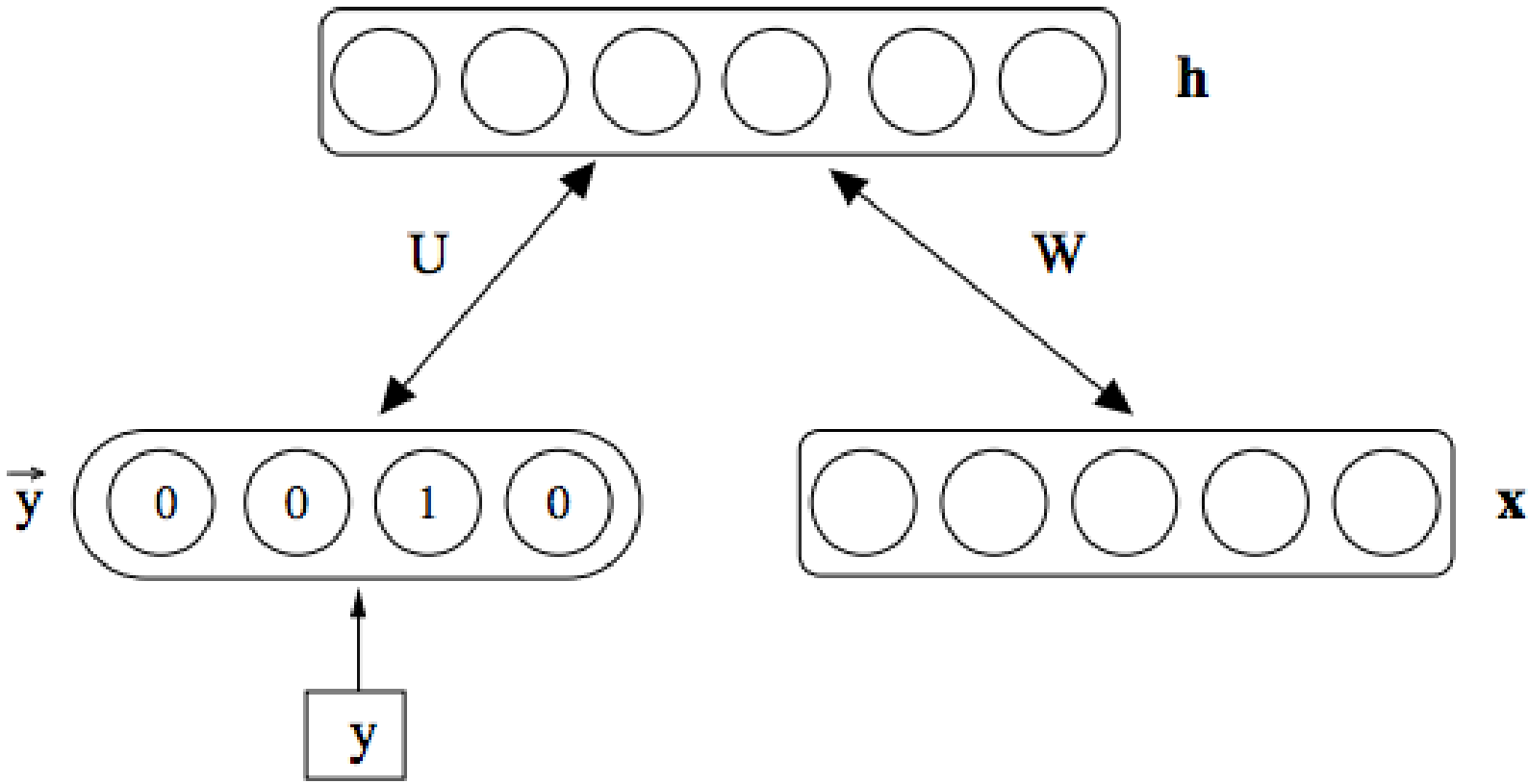


Рис.: Classification using Discriminative Restricted Boltzmann Machines[21]

[21] H. Larochelle, Y. Bengio

Deep Discriminative Restricted Boltzmann Machines

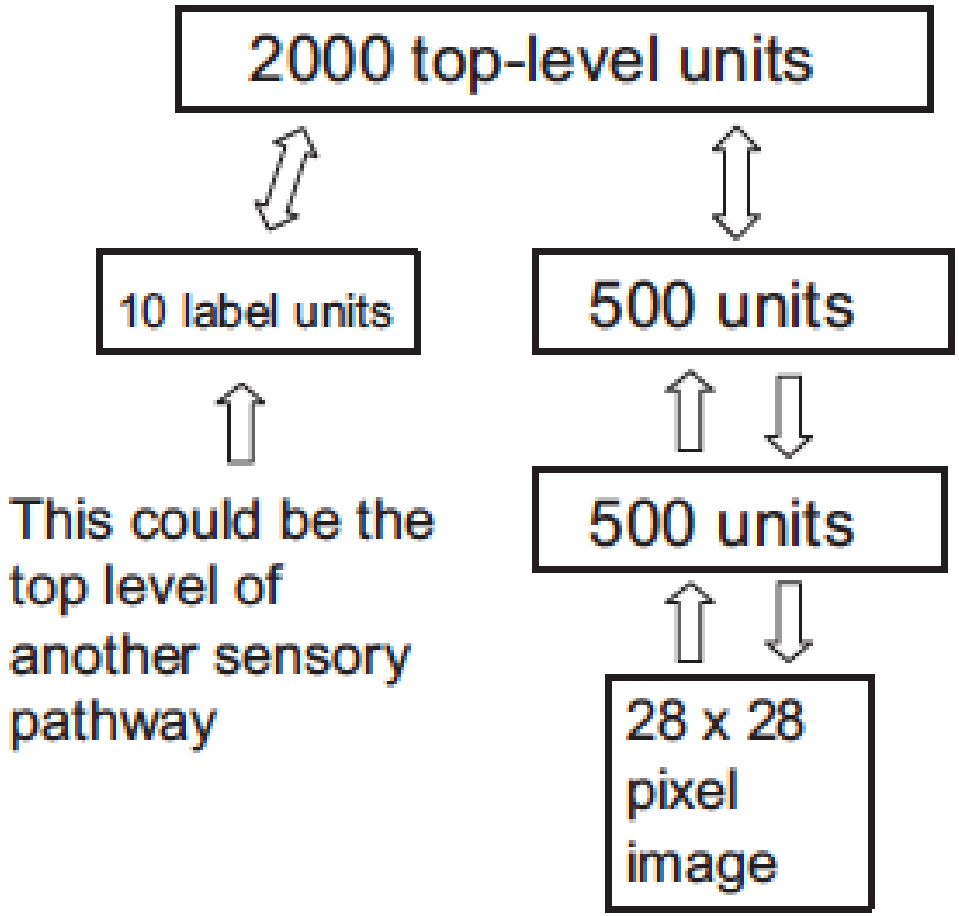


Рис.: RBM for MNIST[22]

[22] G. Hinton

Multimodal Deep Boltzmann Machine

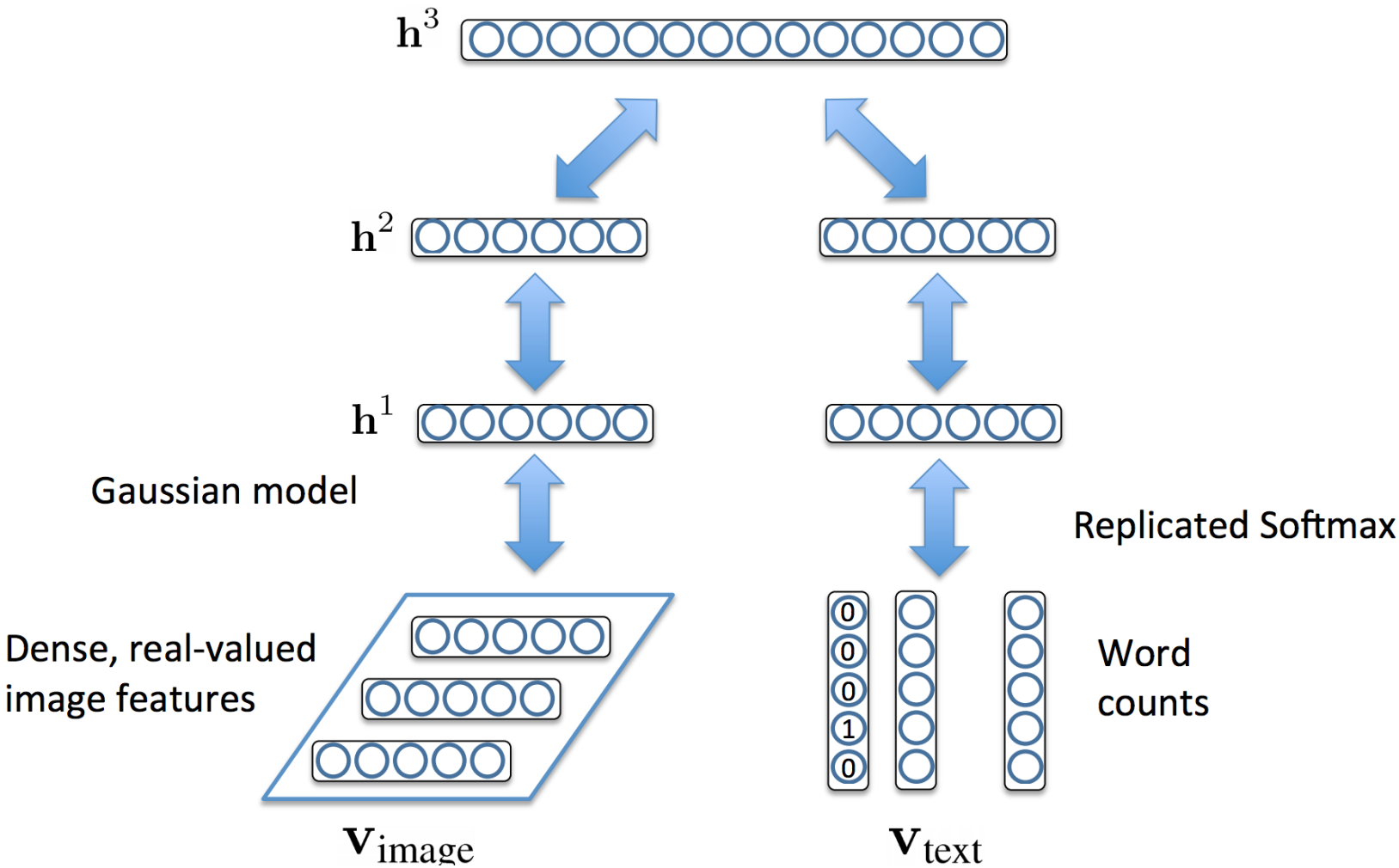


Рис.: Image + Text [23],
`http://videolectures.net/kdd2014_salakhutdinov_deep_learning/`

[23] Srivastava, Salakhutdinov, NIPS 2012

Recurrent Neural Network

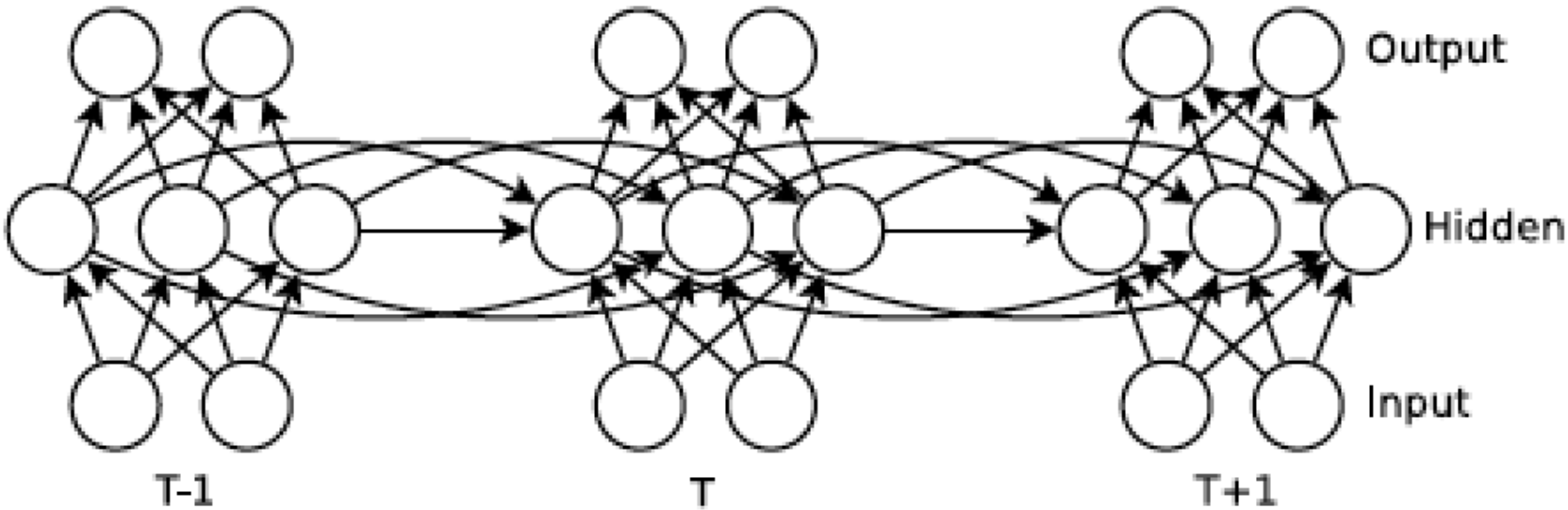


Рис.: RNN - это глубокая нейросеть с общими весами во времени [24]

[24] Generating Text with Recurrent Neural Networks, Sutskever, Martens, Hinton

Convolutional neural network

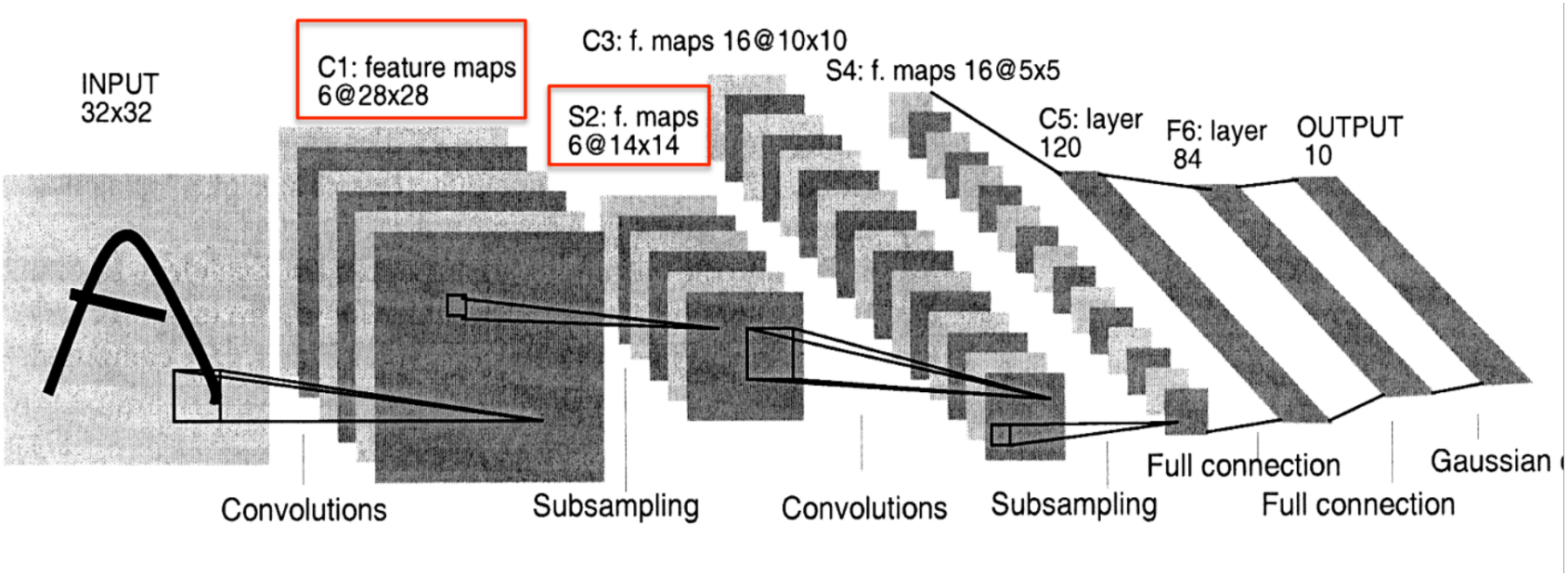


Рис.: LeNet for MNIST[25]

- lenet.gif

[25] Y. LeCunn

Convolution Resticted Boltzmann Machine

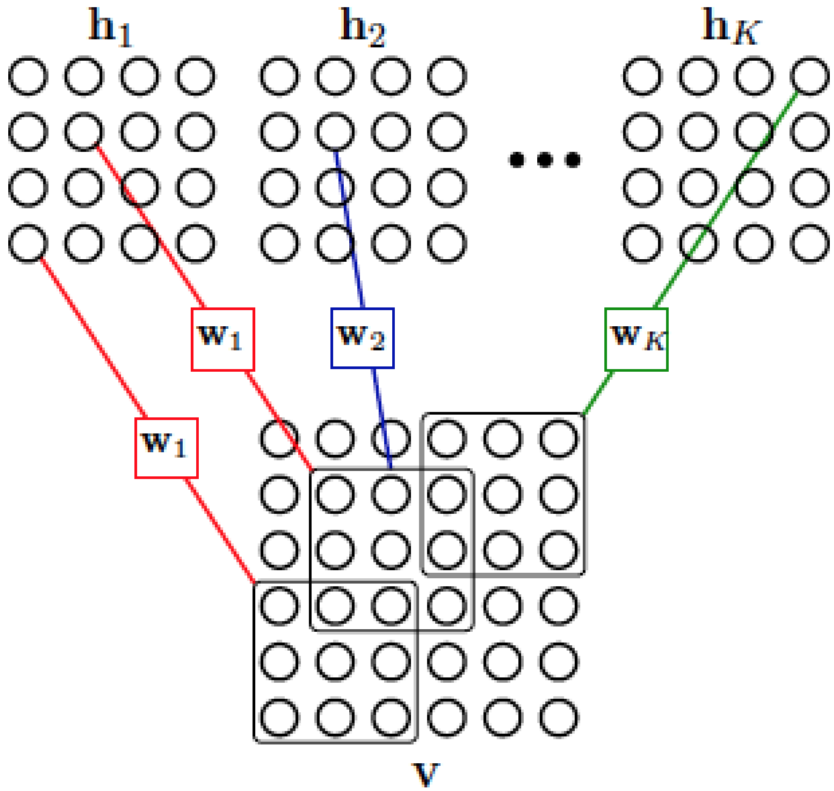
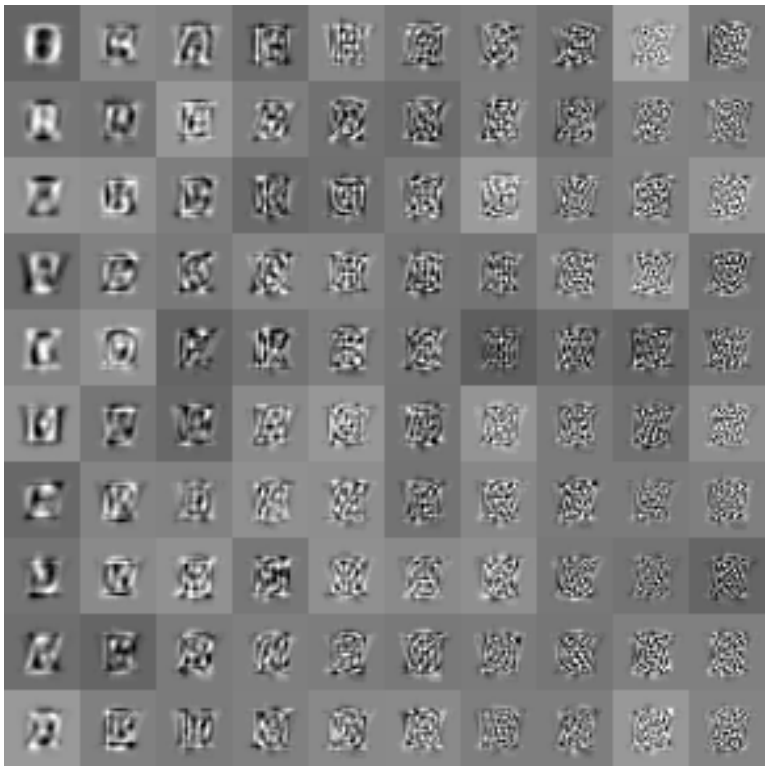
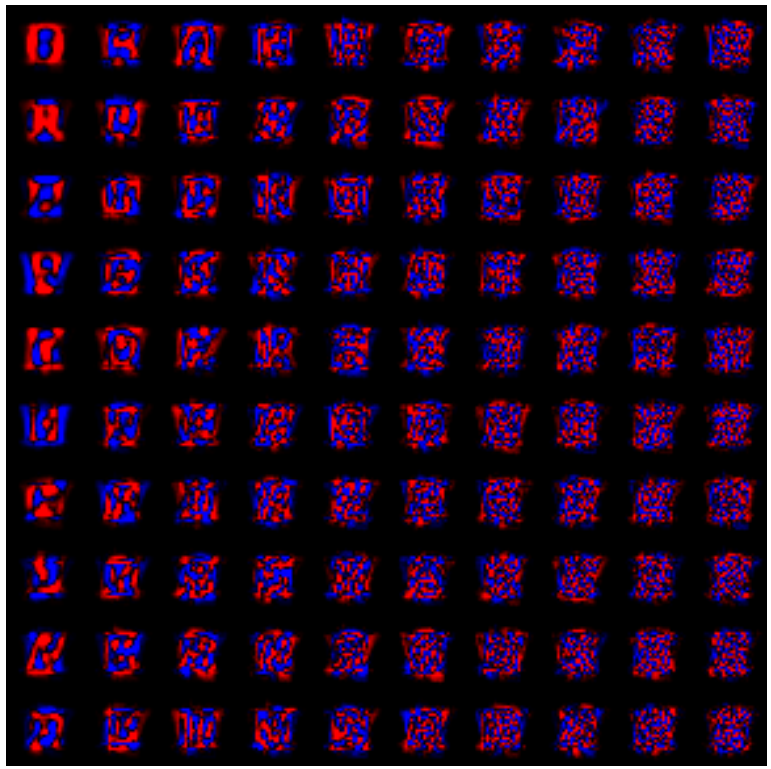


Рис.: Convolution RBM [26]

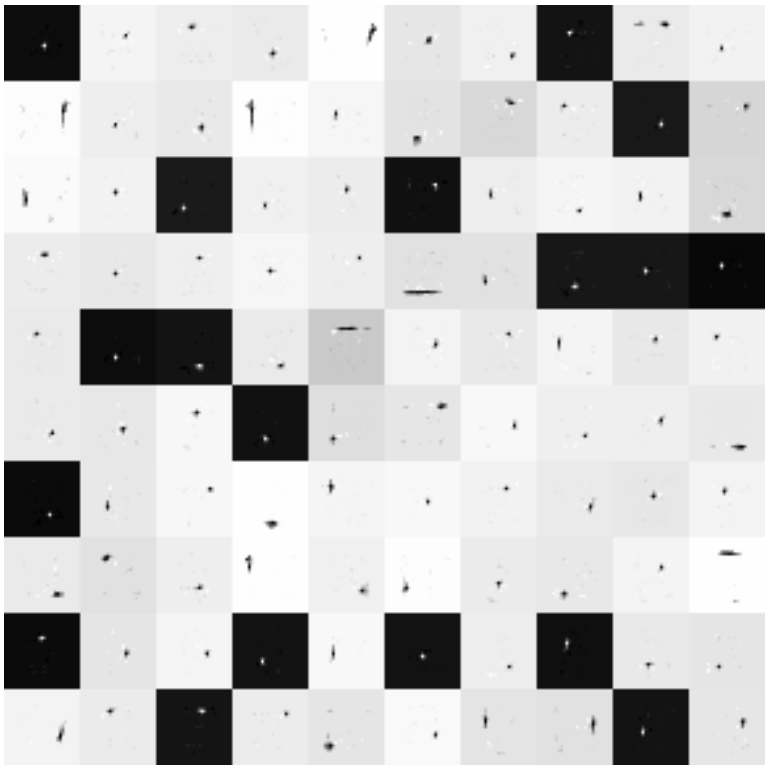
[26] Stacks of Convolutional Restricted Boltzmann Machines for Shift-Invariant Feature Learning, Mohammad Norouzi, Mani Ranjbar, and Greg Mori

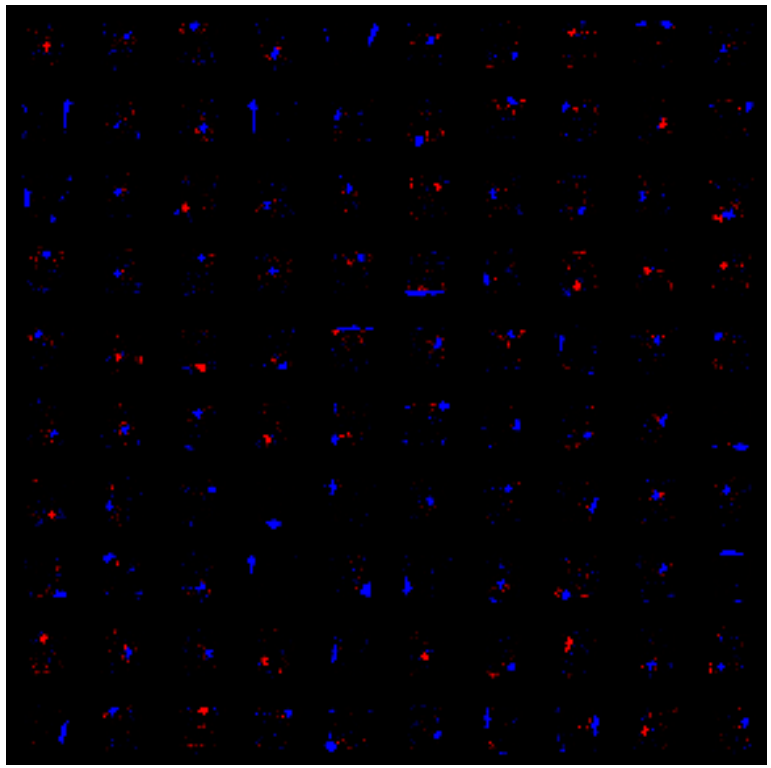
Сжатие размерности пространства, PCA





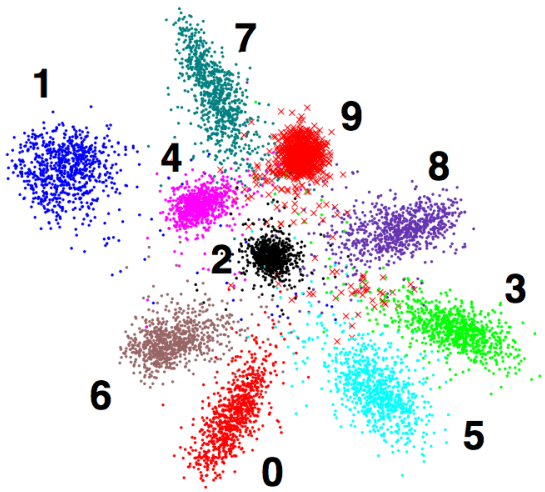
Сжатие размерности пространства, RBM

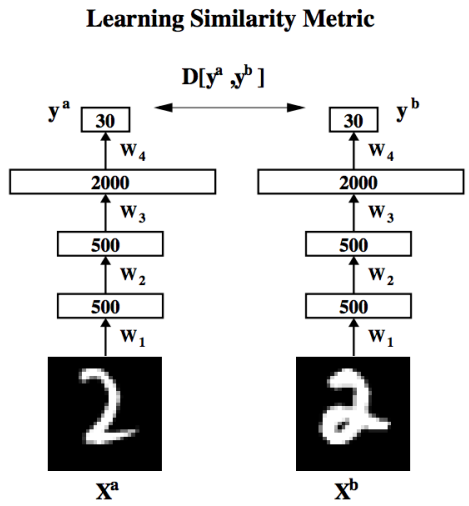




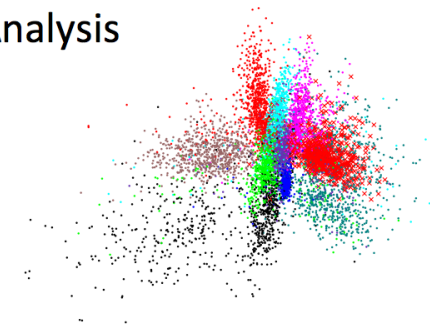
- rbm_features*.png

Learning Similarity Measures

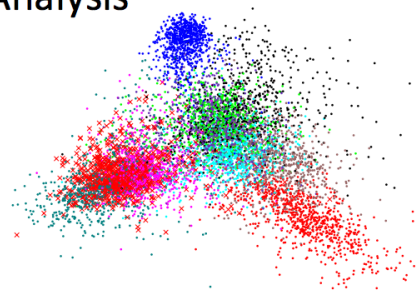




Neighborhood Component Analysis



Linear discriminant Analysis



PCA

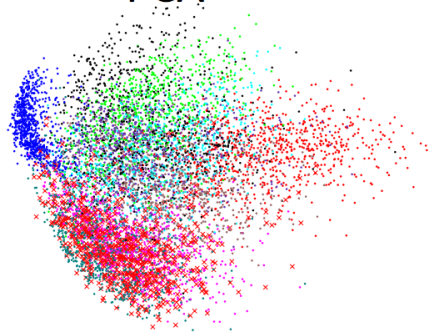


Рис.: New similarities in practice [27],
`http://videolectures.net/kdd2014_salakhutdinov_deep_learning/`

[27] Salakhutdinov and Hinton, AI and Statistics 2007

Neurolinguistic model

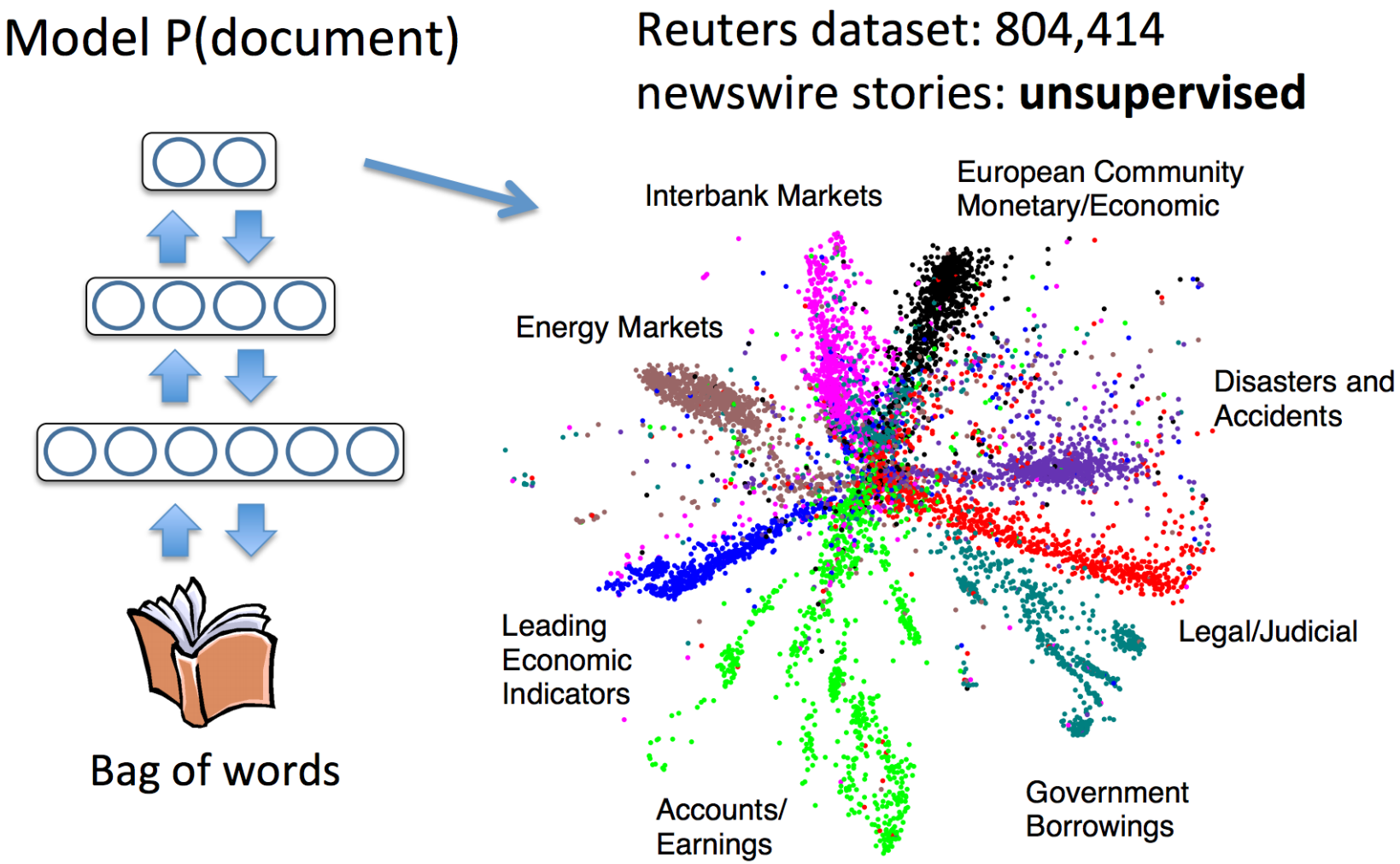


Рис.: Deep generative model [28],
`http://videolectures.net/kdd2014_salakhutdinov_deep_learning/`

[28] Hinton, Salakhutdinov, Science 2006

Multimodal data, #1

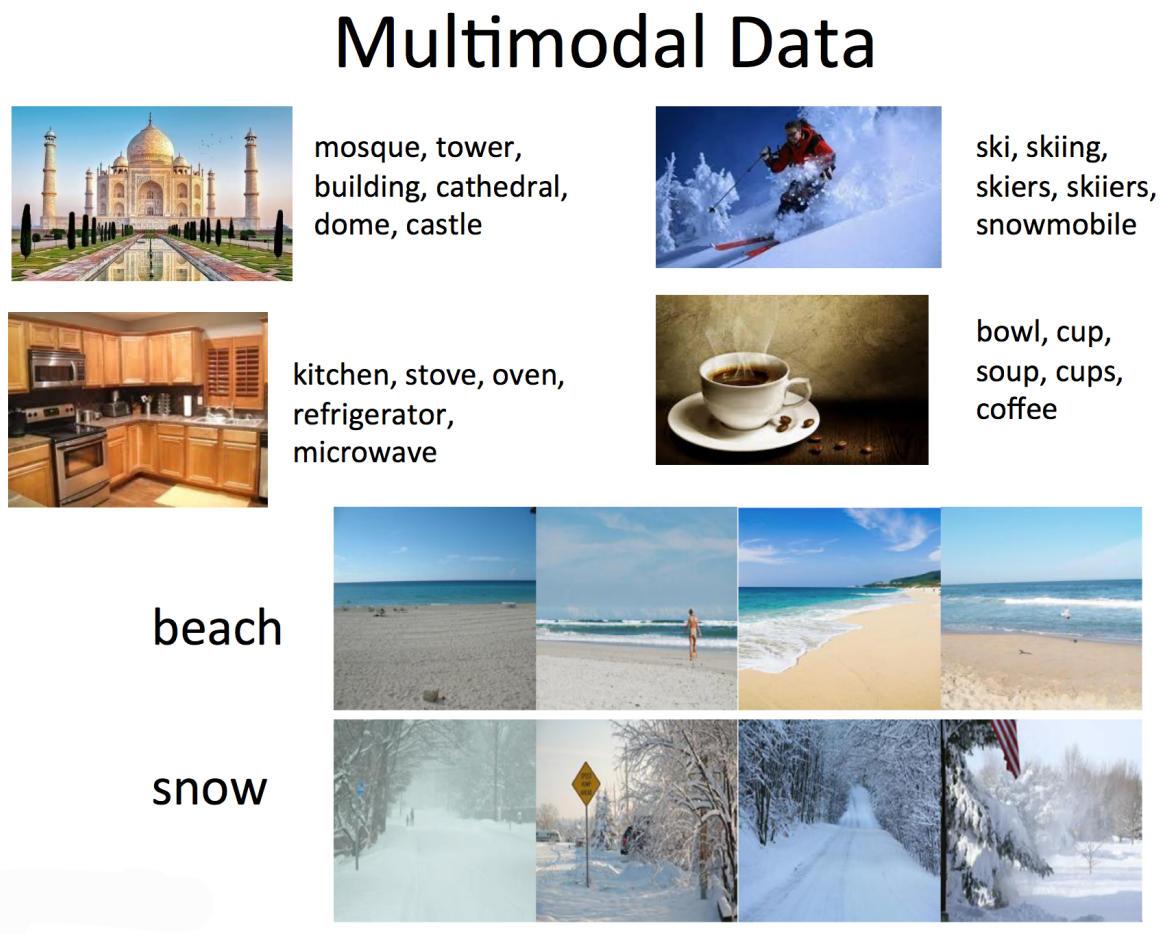


Рис.: Images from text [29],
`http://videolectures.net/kdd2014_salakhutdinov_deep_learning/`

[29] Ryan Kiros, 2014

Multimodal data, #2

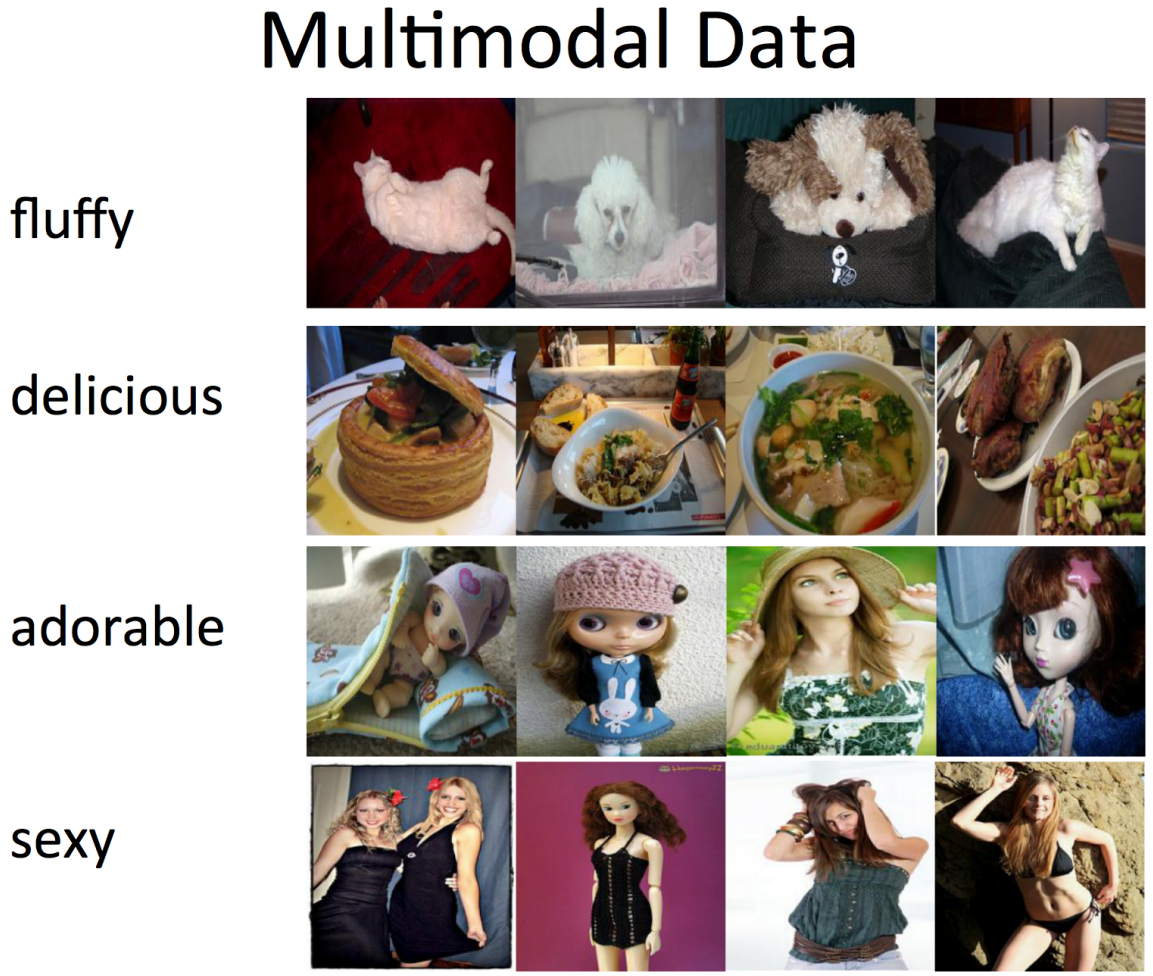


Рис.: Images from text [30],
`http://videolectures.net/kdd2014_salakhutdinov_deep_learning/`

[30] Ryan Kiros, 2014

Text from images

Given	Generated	Given	Generated
	dog, cat, pet, kitten, puppy, ginger, tongue, kitty, dogs, furry		insect, butterfly, insects, bug, butterflies, lepidoptera
	sea, france, boat, mer, beach, river, bretagne, plage, brittany		graffiti, streetart, stencil sticker, urbanart, graff, sanfrancisco
	portrait, child, kid, ritratto, kids, children, boy, cute, boys, italy		canada, nature, sunrise, ontario, fog, mist, bc, morning

Рис.: Text Generated from Images
http://videolectures.net/kdd2014_salakhutdinov_deep_learning/

Natural language from image

Input

Output

A man skiing down the snow covered mountain with a dark sky in the background.

Рис.: Natural Text Generated from Images
`http://videolectures.net/kdd2014_salakhutdinov_deep_learning/`

Multimodal Linguistic Regularities, #1

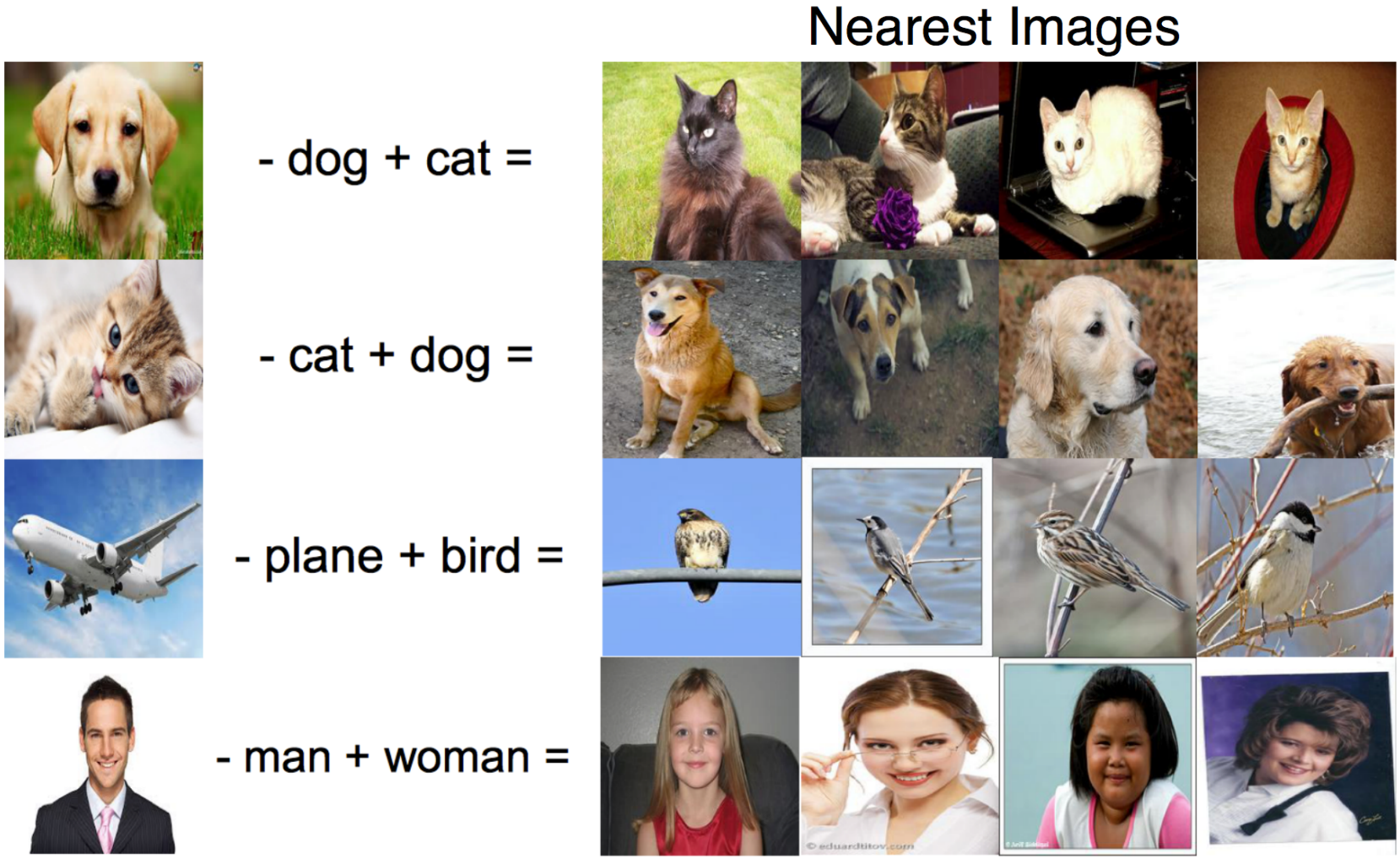


Рис.: Multimodal Linguistic Regularities [31],
`http://videolectures.net/kdd2014_salakhutdinov_deep_learning/`

[31] Ryan Kiros, 2014

Multimodal Linguistic Regularities, #2

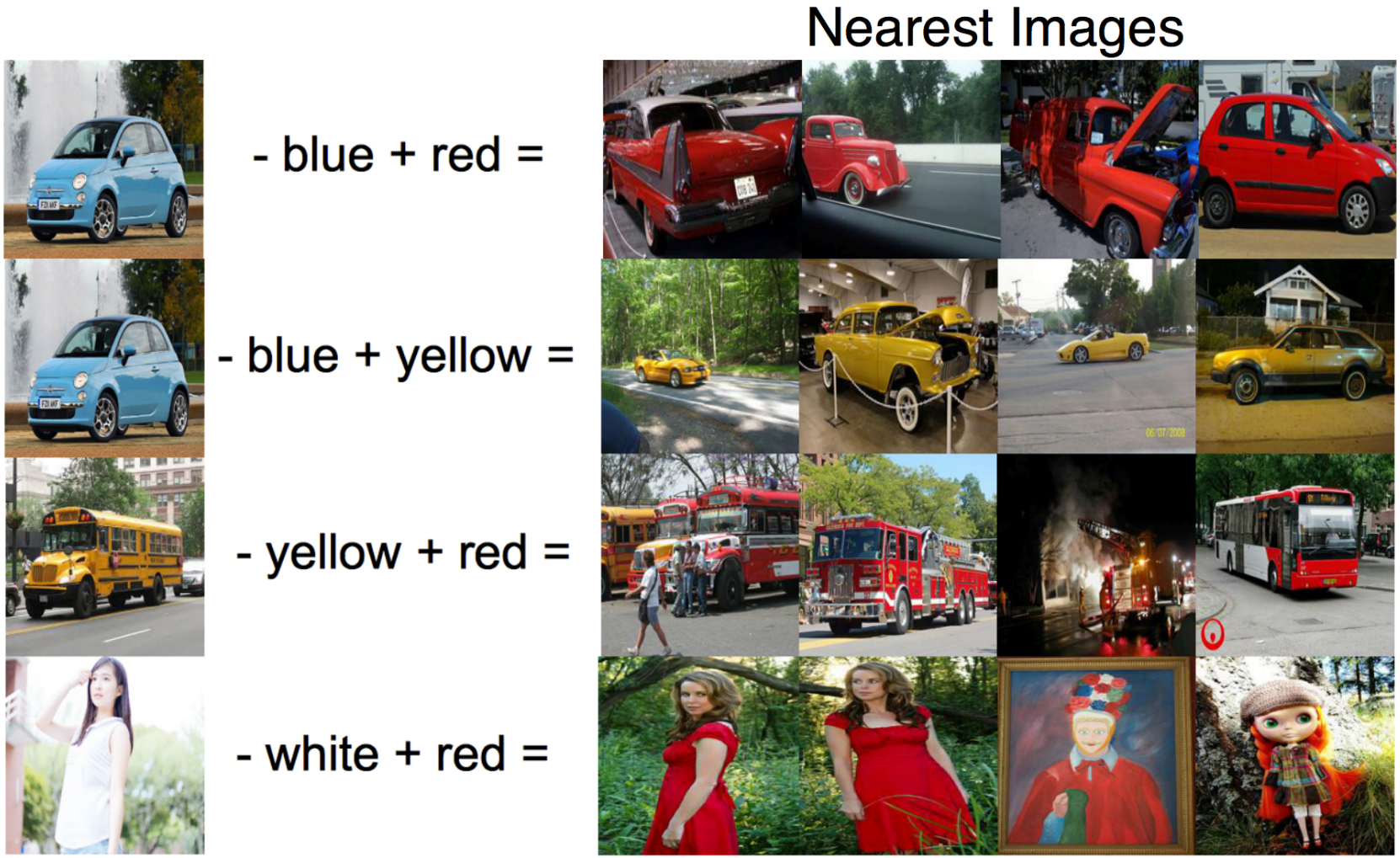


Рис.: Multimodal Linguistic Regularities [32],
`http://videolectures.net/kdd2014_salakhutdinov_deep_learning/`

[32] Ryan Kiros, 2014

Вопросы