

Нейронные сети

Искусственная нейронная сеть (ANN) – математическая модель, а также ее программное и аппаратное воплощение, построенная по принципу организации и функционирования биологический нейронных сетей – сетей нервных клеток животного организма.

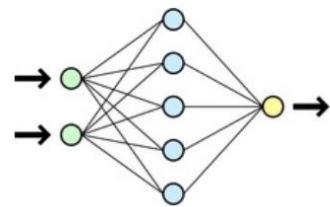
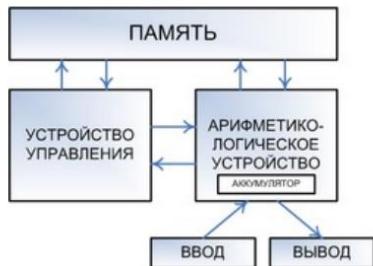
Основным преимуществом нейросетей над обычными алгоритмами вычисления является их возможность обучения. Это обучение заключается в нахождении верных коэффициентов связи между нейронами, а также в обобщении данных и выявлении сложных зависимостей между входными и выходными сигналами.

Нейронные сети находят широкое применение в следующих областях: распознавание, причём это направление в настоящее время самое широкое; предсказание следующего шага, эта особенность применима на торгах и фондовых рынках; классификация входных данных по параметрам, такую функцию выполняют кредитные роботы, которые способны принять решение в одобрении займа человеку, полагаясь на входной набор разных параметров.

Способности нейросетей делают их очень популярными. Их можно научить многому, например, играть в игры, узнавать определённый голос и так далее. Исходя из того, что искусственные сети строятся по принципу биологических сетей, их можно обучить всем процессам, которые человек выполняет неосознанно.

Архитектура машины фон Неймана

Искусственные нейронные сети (ИНС)



Отличия ИНС от архитектуры фон Неймана

	Машина фон Неймана	Биологическая нейронная система
Процессор	Сложный Высокоскоростной Один или несколько	Простой Низкоскоростной Большое количество
Память	Отделена от процессора Локализована Адресация не по содержанию	Интегрирована в процессор Распределенная Адресация по содержанию
Вычисления	Централизованные Последовательные Хранимые программы	Распределенные Параллельные Самообучение
Надежность	Высокая уязвимость	Живучесть
Специализация	Численные и символические операции	Проблемы восприятия
Среда функционирования	Строго определенная Строго ограниченная	Плохо определенная Без ограничений

Биологический нейрон

- Как и любая клетка, нейрон имеет тело (сому), внутри которого располагается ядро. Из сомы нейрона выходят многочисленные отростки, обеспечивающие его взаимосвязь с другими нейронами.
- Выделяется два вида отростков.
 - Первый вид образуют многочисленные тонкие, густо ветвящиеся дендриты, по которым в нейрон поступает информация. Простейший биологический нейрон может иметь несколько дендритов, принимающих информацию от других нейронов.
 - Второй вид представляют более толстые отростки - аксоны, расщепляющиеся на конце на тысячи нервных окончаний – колатералов, передающих информацию на вход другим нейронам. Каждый нейрон имеет только один выходной отросток – аксон.
- Каждый нейрон передаёт возбуждение другим нейронам через особые нервные контакты, называемые синапсами, которые могут располагаться как на соме, так и на дендритах.

Математическая модель нейрона

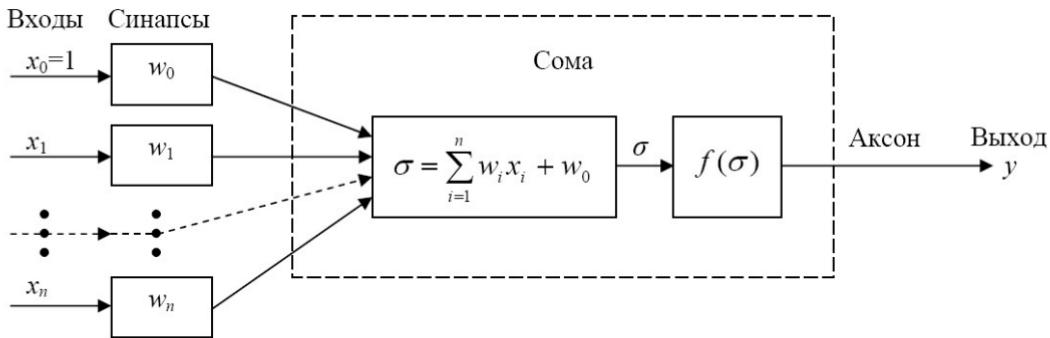
В настоящее время наибольшее распространение получила математическая модель нейрона, представленная следующим уравнением:

$$y = f(\sigma) = f\left(\sum_{i=1}^n w_i x_i + w_0\right),$$

где y – выходной сигнал нейрона (является функцией его состояния);
 $f(\sigma)$ – функция выхода нейрона;
 w_i – вес i -го входа (постоянный коэффициент);
 i – номер входа нейрона;
 n – число входов.

Коэффициенты w_i , представляют веса синаптических связей. Эти коэффициенты моделируют функции синапсов биологических нейронов, по физическому смыслу они эквивалентны электрической проводимости.

Структурная схема нейрона



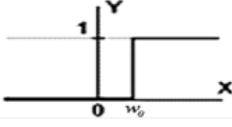
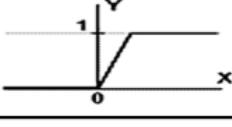
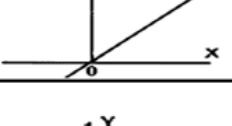
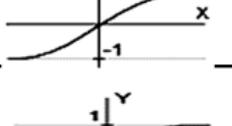
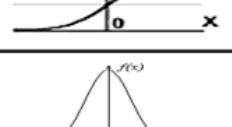
Структурная схема нейрона состоит из $(n+1)$ -го входного блока умножения на коэффициенты w , одного сумматора и выходного блока функционального преобразования. Функция, которую реализует выходной блок, получила название функции активации.

Базовая модель ИНС

Нейронную сеть можно представить себе в виде черного ящика, у которого имеется n входов и m выходов. Кроме этого, имеется набор (матрица) весовых коэффициентов, общее количество которых равно произведению n на m .

Элементы, обозначенные цифрами от 1 до m представляют собой слой нейронов. Первый слой сети, на который подаются входные сигналы $X_1 \dots X_n$, играет роль распределительного слоя нейронной сети имеет связи со всем нейронами следующего слоя, называемого обрабатывающим слоем.

В том случае, когда нейросеть состоит из большого количества нейронов, вводят термин слоя. Соответственно, есть входной слой, который получает информацию, от скрытых слоев (обычно их не больше 3), которые ее обрабатывают и выходной слой, который выводит результат. У каждого из нейронов есть 2 основных параметра: входные данные (input data) и выходные данные (output data). В случае входного нейрона: $\text{input} = \text{output}$. В остальных, в поле input попадает суммарная информация всех нейронов с предыдущего слоя, после чего, она нормализуется, с помощью функции активации и попадает в поле output

Активационная функция	Формула	Вид
Единичного скачка	$f(x) = \begin{cases} 0, & x < w_0; \\ 1, & x \geq w_0. \end{cases}$	
Линейного порога	$f(x) = \begin{cases} 0, & x < 0; \\ \frac{x}{w_0}, & w_0 > x \geq 0; \\ 1, & x \geq w_0. \end{cases}$	
Линейная	$f(x) = x.$	
Гиперболический тангенс	$f(x) = \frac{e^{ax} - e^{-ax}}{e^{ax} + e^{-ax}}$	
Логистическая Сигмоид	$f(x) = \frac{1}{1 + e^{-ax}}$	
Гаусса	$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{(x-m)^2}{2\sigma^2}}$	

Алгоритм

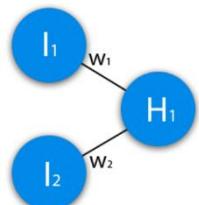
- получить входные значения
- вычислить значение активации (взвешенная сумма входа – пороговое значение)
- преобразовать значение активации с помощью функции активации

Работа ИНС

Представлена часть нейронной сети, где: I – входные нейроны, H – скрытый нейрон, w – веса.

Пусть $w_1=0.4$ и $w_2 = 0.7$

Тогда входные данные нейрона H1 будут следующими: $1*0.4+0*0.7=0.4$



$$1) H_{1\text{input}} = (I_1 * w_1) + (I_2 * w_2)$$

$$2) H_{1\text{output}} = f_{\text{activation}}(H_{1\text{input}})$$

Классификация нейронных сетей



Алгоритмы обучения нейронной сети

Обучение с учителем

- Подготавливается набор обучающих данных. Эти данные представляют собой примеры входных данных и соответствующих им выходов. Сеть учится устанавливать связь между первыми и вторыми
- Нейронная сеть обучается с помощью того или иного алгоритма управляемого обучения, при котором имеющиеся данные используются для корректировки весов и пороговых значений сети таким образом, чтобы минимизировать ошибку прогноза на обучающем множестве.

Обучение без учителя

- Обучающие данные содержат только значения входных переменных.
- Сеть учится распознавать внутреннюю структуру данных.

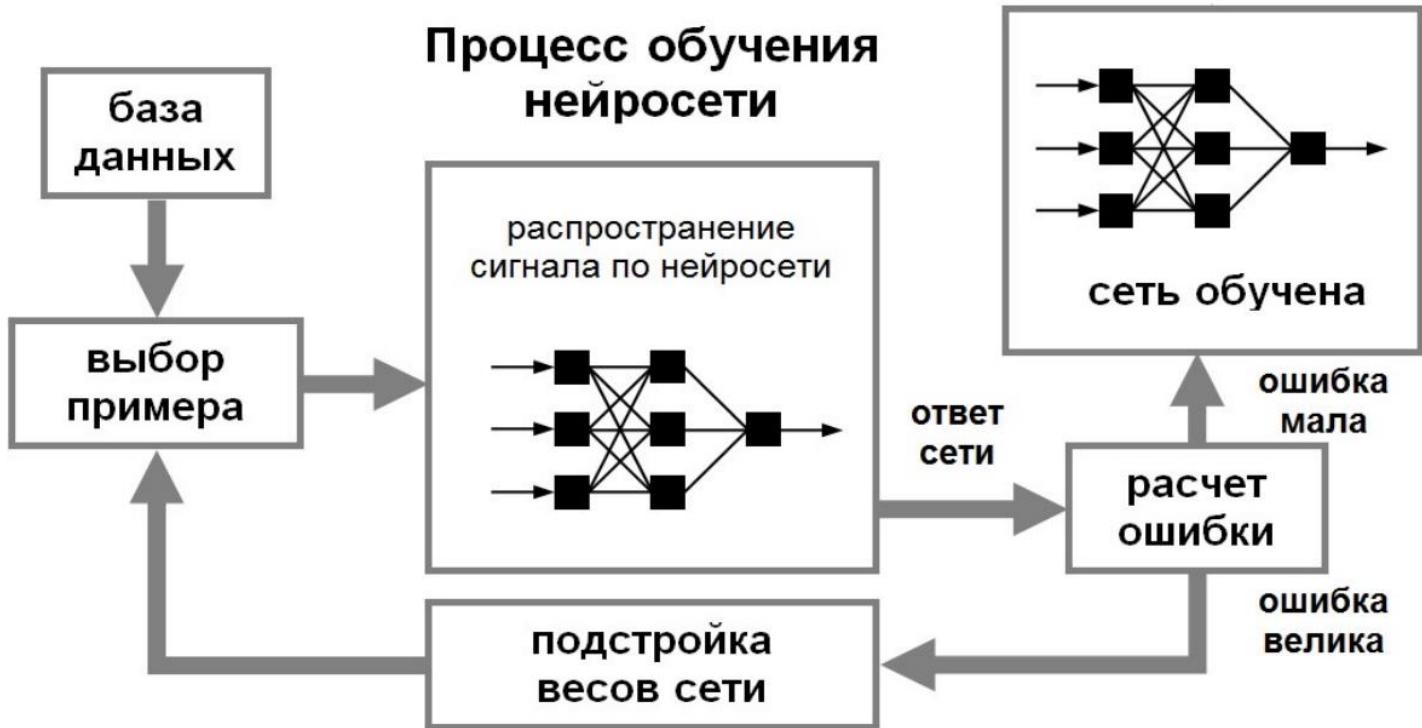
Тренировочный сет — это последовательность данных, которыми оперирует нейронная сеть.

Итерация – своеобразный счетчик, который увеличивается каждый раз, когда нейронная сеть проходит один тренировочный сет. Другими словами, это общее количество тренировочных сетов, пройденных нейронной сетью.

Эпоха – Эпоха увеличивается каждый раз, когда мы проходим весь набор тренировочных сетов. При инициализации нейронной сети эта величина устанавливается в 0 и имеет потолок, задаваемый вручную. Чем больше эпоха, тем лучше натренирована сеть и соответственно, ее результат.

✓ for (int i=0;i<maxEpoch;i++)
for (int j=0;j<trainSet;j++)

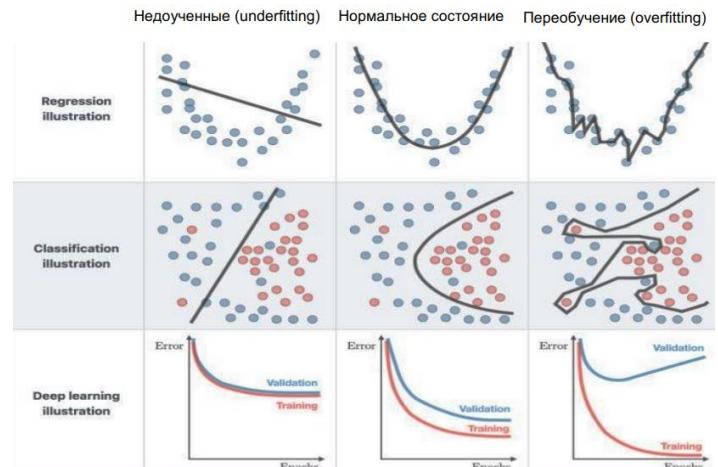
✗ for (int j=0;j<trainSet;j++)
for (int i=0;i<maxEpoch;i++)



Переобучение

Причины возникновения:

- Недоученные (underfitting)
 - Высокая ошибка при обучении
 - Ошибка обучения близка к тестовой
 - Высокое смещение\предвзятости (bias)
- Нормальное состояние
 - Ошибка обучения немного ниже ошибки тестирования
- Переобучение (overfitting)
 - Очень низкая погрешность обучения
 - Ошибка обучения значительно ниже ошибки тестирования
 - Высокая дисперсия



Возможные способы устранения (недоученное):

- Усложнить модель
- Добавить больше функций
- Обучаться дольше

Возможные способы устранения (переобучение):

- Выполнить регуляризацию
- Получить больше данных

Ошибка и функция потерь (Loss Function)

Ошибка — это процентная величина, отражающая расхождение между ожидаемым и полученным ответами. Ошибка формируется каждую эпоху и должна идти на спад.

MSE (Mean Squared Error) – среднеквадратичная ошибка

$$\frac{(i_1 - a_1)^2 + (i_2 - a_2)^2 + \dots + (i_n - a_n)^2}{n}$$

Root MSE – среднеквадратическое отклонение

$$\sqrt{\frac{(i_1 - a_1)^2 + (i_2 - a_2)^2 + \dots + (i_n - a_n)^2}{n}}$$

Кросс-энтропия (или логарифмическая функция потерь – log loss)

$$H(P, Q) = - \sum_x P(x) \log Q(x)$$

Функция потерь (loss function) — отображение результата работы алгоритма, показывающее "стоимость" ошибки.

- 0-1 функция

$$L(a, x) = [a(x) \neq y(x)]$$

- Квадратичная функция

$$L(a, x) = (a(x) - y(x))^2$$

- Hinge loss

$$L(a, x) = \max(0, 1 - a(x) \cdot y(x))$$

- Логистическая

$$L(a, x) = \frac{\ln(1 + e^{-y(x)a(x)})}{\ln 2}$$

- Log loss

$$t(y, x) = \frac{1 + y(x)}{2}, L(a, x) = -t \cdot \ln(a(x)) - (1 - t) \cdot \ln(1 - a(x))$$

Понятие функции потерь тесно связано с эмпирическим риском.

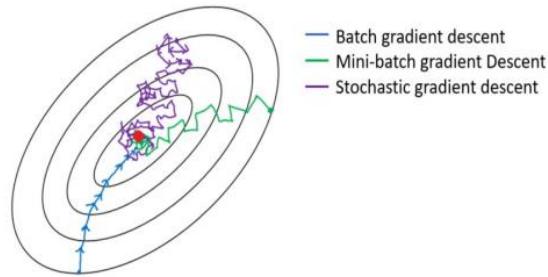
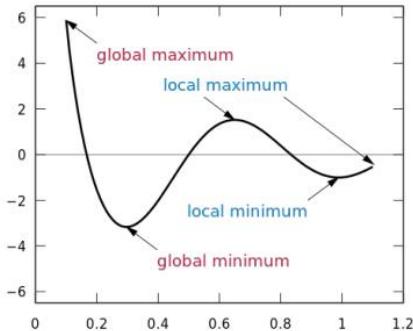
Эмпирический риск — средняя величина ошибки на обучающей выборке:

$$Q(a, X^m) = \frac{1}{m} \sum_{x \in X} L(a, x)$$

Алгоритм стохастического градиентного спуска

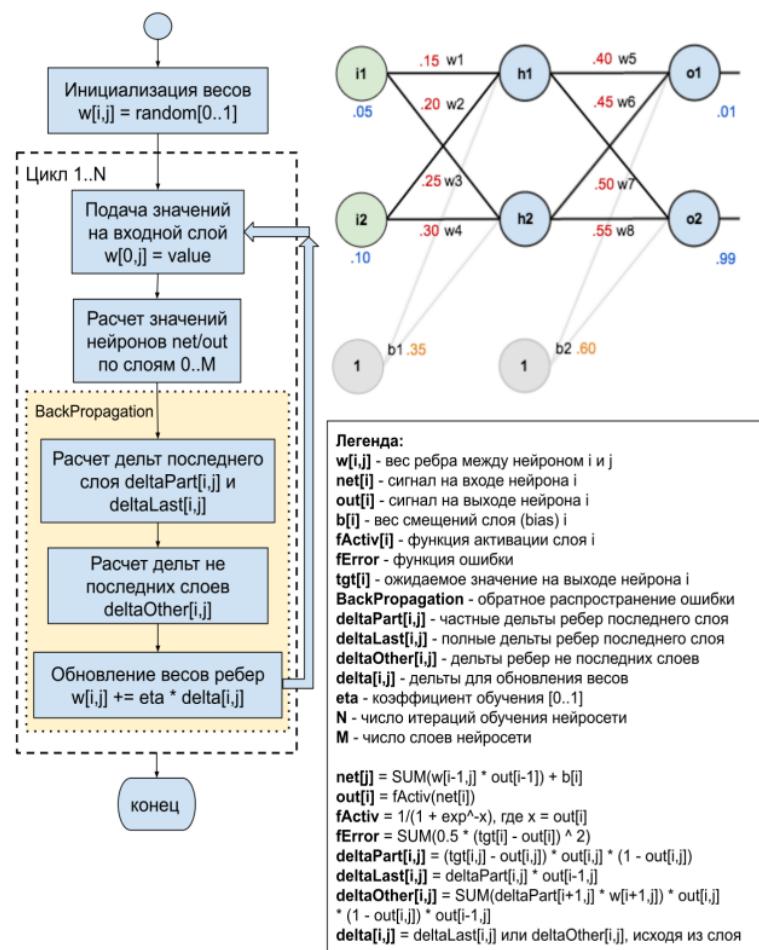
Градиентный спуск — метод нахождения локального минимума или максимума функции с помощью движения вдоль градиента.

Стохастический градиентный спуск (англ. Stochastic gradient descent, SGD) — это итерационный метод для оптимизации целевой функции с подходящими свойствами гладкости (например, дифференцируемость или субдифференцируемость). Его можно рассматривать как стохастическую аппроксимацию оптимизации методом градиентного спуска, поскольку он заменяет реальный градиент, вычисленный из полного набора данных его оценкой, вычисленной из случайно выбранного подмножества данных. Это сокращает задействованные вычислительные ресурсы и помогает достичь более высокой скорости итераций в обмен на более низкую скорость сходимости. Особенно большой эффект достигается в приложениях связанных с обработкой больших данных.

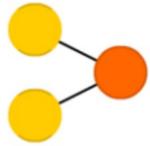


Существует два режима реализации метода обратного распространения ошибки

- Стохастического (stochastic) градиентного спуска
- Пакетного (batch) градиентного спуска



Perceptron



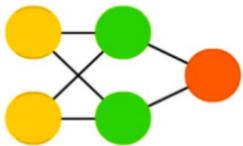
Самая простая нейронная сеть.
Входные элементы напрямую соединены
с выходными с помощью системы весов.

Описывается следующей моделью:

$$a(\mathbf{x}, \mathbf{w}) = \sigma (\langle \mathbf{w}, \mathbf{x} \rangle),$$

где $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ — функция активации (в частности, sign), а
 $\mathbf{w} \in \mathbb{R}^n$ — веса признаков.

Feed Forward



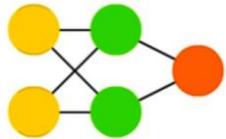
Перцептрон, в котором присутствует
дополнительный скрытый слой.
Нейроны одного слоя между собой
не связаны, при этом каждый нейрон
связан с каждым нейроном соседнего слоя.

Базовая модель описывается следующим способом:

$$a(\mathbf{x}) = \sigma_m (\langle \mathbf{w}, \mathbf{u} \rangle) \quad \mathbf{u} = \sigma_h (\mathbf{Wx}),$$

где $\sigma_m : \mathbb{R} \rightarrow \mathbb{R}$, $\sigma_h : \mathbb{R}^h \rightarrow \mathbb{R}^h$ — функции активации,
а $\mathbf{w} \in \mathbb{R}^h$, $\mathbf{W} \in \mathbb{R}^{h \times n}$ — веса признаков.

Radial Basis Function



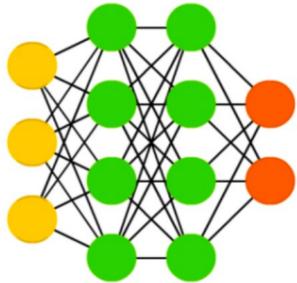
Feed Forward Neural Network, которая использует радиальные базисные функции как функции активации.

Соответственно модель будет такой же:

$$a(x) = \sigma_m (\langle w, u \rangle) \quad u = \sigma_h (Wx),$$

однако $\sigma_h : \mathbb{R}^h \rightarrow \mathbb{R}^h$ — радиально-базисные функции.

Deep Feed Forward



Feed Forward Neural Network, которая содержит несколько скрытых слоев.

Выходные значения сети $a \in \mathbb{R}^m$ на объекте x :

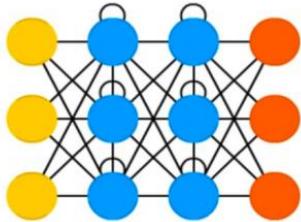
$$a(x) = \sigma_m (\langle w, u_2 \rangle) \quad u_2 = \sigma_{h_2} (W_2 u_1) \quad u_1 = \sigma_{h_1} (W_1 x),$$

где $\sigma_m : \mathbb{R} \rightarrow \mathbb{R}$, $w \in \mathbb{R}^{h_2}$,

$$\sigma_{h_2} : \mathbb{R}^{h_2} \rightarrow \mathbb{R}^{h_2}, \quad W_2 \in \mathbb{R}^{h_2 \times h_1},$$

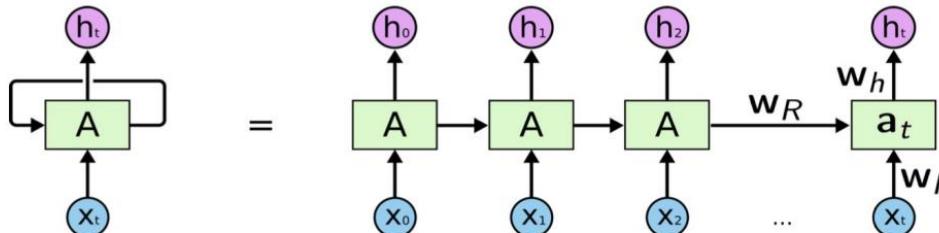
$$\sigma_{h_1} : \mathbb{R}^{h_1} \rightarrow \mathbb{R}^{h_1}, \quad W_1 \in \mathbb{R}^{h_1 \times n}.$$

Recurrent Neural Network



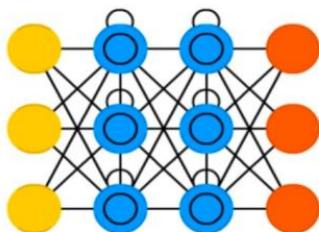
Нейроны получают информацию не только от предыдущего слоя, но и от самих себя в результате предыдущего прохода.

Развернем обратную связь одного нейрона:



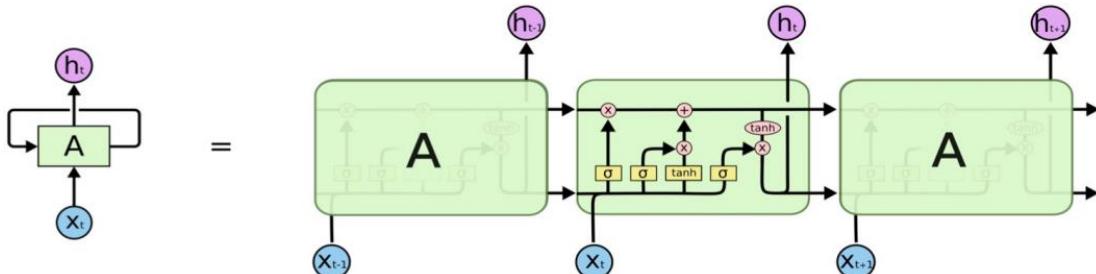
$$h_t = \sigma_h (\mathbf{W}_h a_t) \quad a_t = \sigma_a (\mathbf{W}_I x_t + \mathbf{W}_R a_{t-1})$$

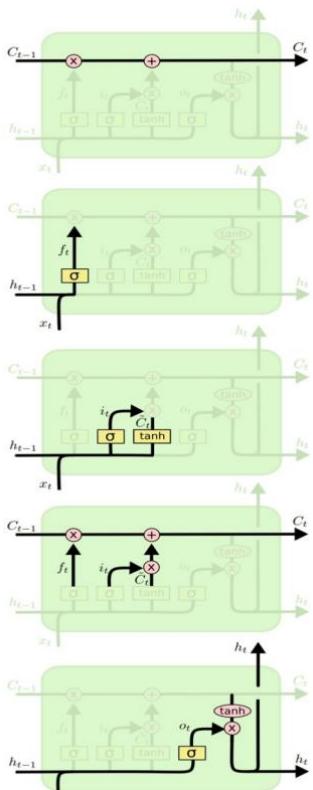
Long / Short Term Memory



Особая разновидность архитектуры рекуррентных нейронных сетей, способная к обучению долговременным зависимостям.

Повторяющийся модуль в LSTM сети состоит из нескольких взаимодействующих слоев:





Состояние ячейки (cell state).
Проходит напрямую через всю цепочку.

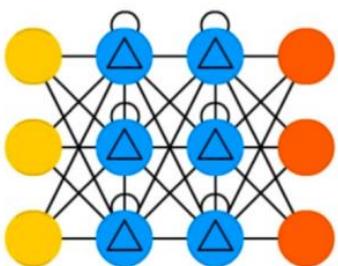
Forget gate layer:
 $f_t = \sigma (\mathbf{W}_f [\mathbf{h}_{t-1}, \mathbf{x}_t]).$

Input gate layer:
 $\mathbf{i}_t = \sigma (\mathbf{W}_i [\mathbf{h}_{t-1}, \mathbf{x}_t]),$
 $\tilde{\mathbf{C}}_t = \tanh (\mathbf{W}_C [\mathbf{h}_{t-1}, \mathbf{x}_t]).$

Обновление состояния ячейки:
 $\mathbf{C}_t = f_t * \mathbf{C}_{t-1} + i_t * \tilde{\mathbf{C}}_t.$

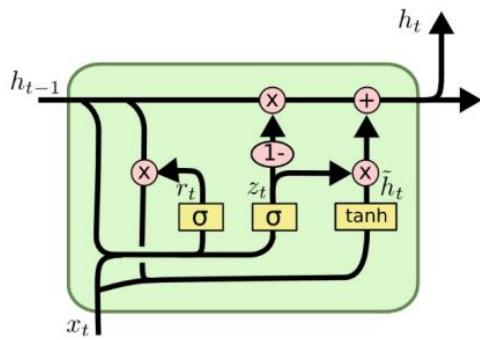
Выходные данные:
 $\mathbf{o}_t = \sigma (\mathbf{W}_o [\mathbf{h}_{t-1}, \mathbf{x}_t]),$
 $\mathbf{h}_t = \mathbf{o}_t * \tanh (\mathbf{C}_t).$

Gated Recurrent Unit



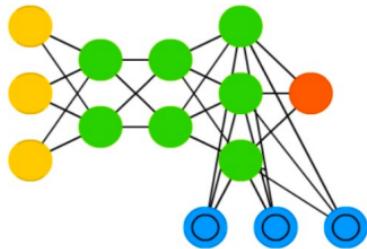
Упрощенная LSTM, в которой фильтры «забывания» и «входа» объединяют в один фильтр «обновления» (update gate).

Повторяющийся модуль в архитектуре GRU сети:



$\mathbf{z}_t = \sigma (\mathbf{W}_z [\mathbf{h}_{t-1}, \mathbf{x}_t]),$
 $\mathbf{r}_t = \sigma (\mathbf{W}_r [\mathbf{h}_{t-1}, \mathbf{x}_t]),$
 $\tilde{\mathbf{h}}_t = \tanh (\mathbf{W} [\mathbf{r}_t * \mathbf{h}_{t-1}, \mathbf{x}_t]),$
 $\mathbf{h}_t = (1 - \mathbf{z}_t) * \mathbf{h}_{t-1} + \mathbf{z}_t * \tilde{\mathbf{h}}_t.$

Neural Turing Machine



Блок памяти отделен от нейрона.

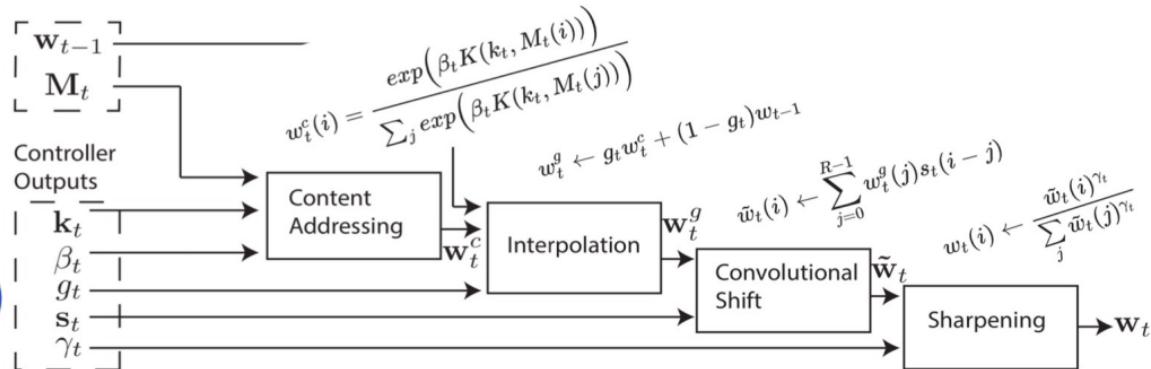
\mathbf{M}_t — матрица памяти в момент t .

\mathbf{w}_t — весовой вектор (механизм).

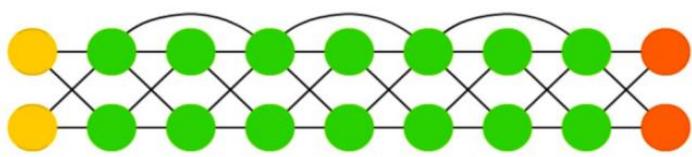
Чтение: $\mathbf{r}_t \leftarrow \mathbf{M}_t^T \mathbf{w}_t$.

Запись: $\mathbf{M}_t \leftarrow \mathbf{M}_{t-1} - \mathbf{M}_{t-1} \circ (\mathbf{w}_t^T \mathbf{e}_t) + \mathbf{w}_t^T \mathbf{a}_t$.

Адресация:



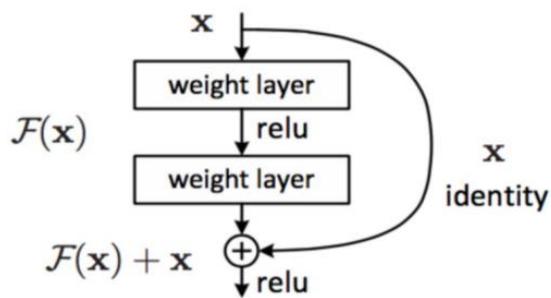
Deep Residual Network



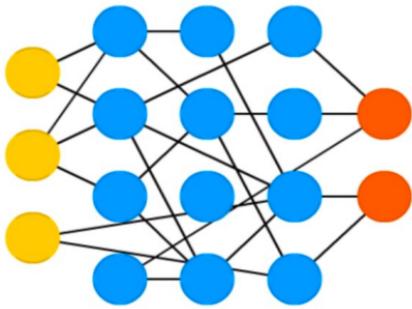
Очень глубокие FFNN с дополнительными связями между слоями.

Было доказано, что сети этого типа на самом деле просто RNN без явного использования времени.

Конструкция основного блока DRN сети:



Echo State Network



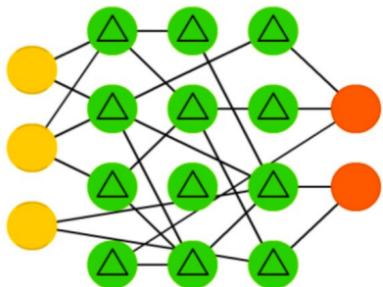
Еще один вид рекуррентных нейросетей.

Быстрее обучаются.
Более устойчивы.

Несколько ключевых отличий:

- Связи между нейронами скрытого слоя случайны;
- Настраиваемыми являются только веса выходного слоя;
- Веса скрытого слоя сети задаются один раз при инициализации сети и не изменяются в процессе ее функционирования.

Liquid State Machine



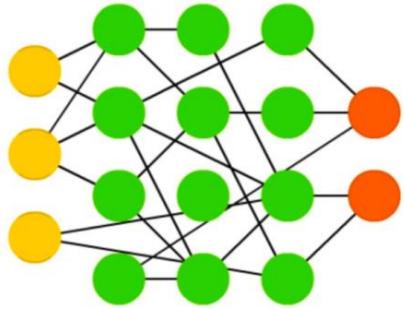
Разновидность импульсных нейронных сетей.

Подобны ESN.

Несколько ключевых отличий от ESN:

- Вместо сигмоидных функций рассматриваются пороговые функции;
- Как только порог превышен, энергия освобождается и нейрон посылает импульс другим нейронам.

Extreme Learning Machine

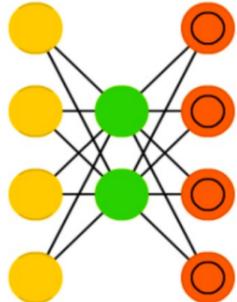


FFNN со случайными связями между нейронами.

Обучаются методом обратного распространения ошибки.

Модель будет такой же, что у FFNN, только с поправкой на случайные связи.

Auto Encoder



Другой способ использования FFNN.
Идея — автоматическое кодирование.

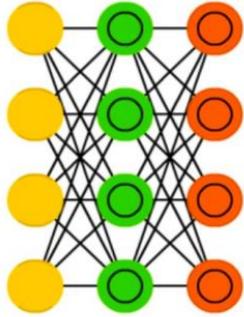
Конструируется таким образом, чтобы не иметь возможность точно скопировать вход на выходе.

Модель имеет следующий вид:

$$\mathbf{a}(\mathbf{x}) = \sigma_n (\mathbf{W}_n \mathbf{u}) \quad \mathbf{u} = \sigma_h (\mathbf{W}_h \mathbf{x}),$$

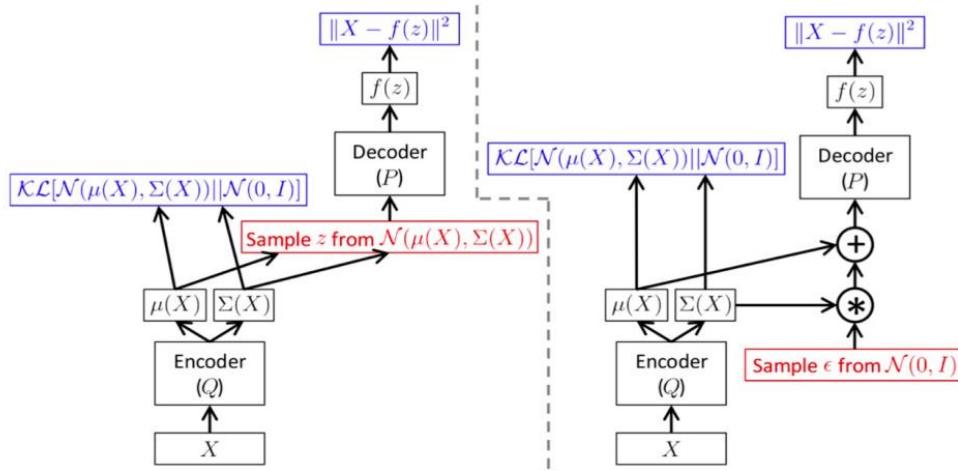
где $\sigma_n : \mathbb{R}^n \rightarrow \mathbb{R}^n$, $\sigma_h : \mathbb{R}^h \rightarrow \mathbb{R}^h$ — сигмоидные функции активации, веса $\mathbf{W}_m \in \mathbb{R}^{n \times h}$ и $\mathbf{W}_h \in \mathbb{R}^{h \times n}$ ($h < n$).

Variational Auto Encoder

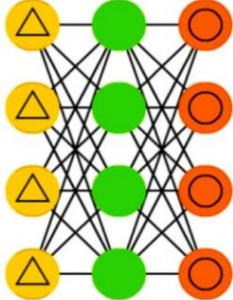


Автоэнкодеры, которые учатся отображать объекты в заданное скрытое пространство и, соответственно, сэмплировать из него.

Более подробная схема:



Denoising Auto Encoder



Автоэнкодер, которому подаем на вход данные с шумом.

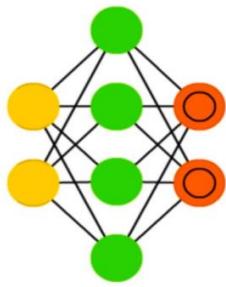
Ошибку вычисляем прежним методом, сравнивая выходной образец с оригиналом без шума.

Модель имеет следующий вид:

$$\mathbf{a}(\mathbf{x}) = \sigma_n (\mathbf{W}_n \mathbf{u}) \quad \mathbf{u} = \sigma_h (\mathbf{W}_h \tilde{\mathbf{x}}),$$

где $\sigma_n : \mathbb{R}^n \rightarrow \mathbb{R}^n$, $\sigma_h : \mathbb{R}^h \rightarrow \mathbb{R}^h$ — сигмоидные функции активации, веса $\mathbf{W}_m \in \mathbb{R}^{n \times h}$ и $\mathbf{W}_h \in \mathbb{R}^{h \times n}$, а вектор $\tilde{\mathbf{x}} \in \mathbb{R}^n$ обозначает зашумленный вектор $\mathbf{x} \in \mathbb{R}^n$.

Sparse Auto Encoder



Антипод автоэнкодера.

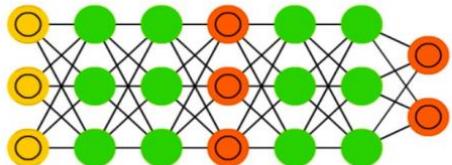
Для обучения требуется ввести штраф за количество активированных нейронов в скрытом слое.

В остальном модель ничем не отличается от модели АЕ:

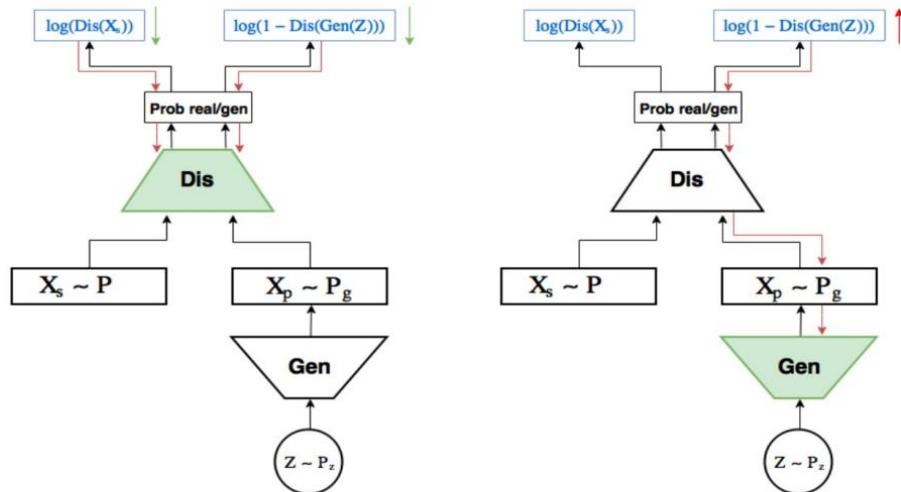
$$a(x) = \sigma_n(W_n u) \quad u = \sigma_h(W_h x),$$

где $\sigma_n : \mathbb{R}^n \rightarrow \mathbb{R}^n$, $\sigma_h : \mathbb{R}^h \rightarrow \mathbb{R}^h$ — сигмоидные функции активации, веса $W_m \in \mathbb{R}^{n \times h}$ и $W_h \in \mathbb{R}^{h \times n}$.

Generative Adversarial Network

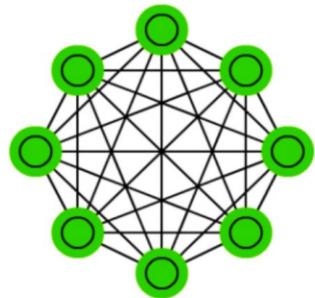


Одна из сетей генерирует данные, а вторая — анализирует.



$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim P} [\log D(x)] + \mathbb{E}_{z \sim P_z} [\log (1 - D(G(z)))]$$

Markov Chain



Задаются вероятности перехода из текущего состояния в соседние.

Формирует теоретическую основу для НН и ВМ.

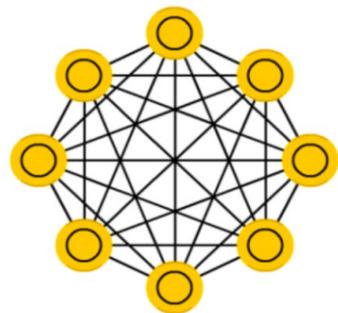
Напоминание:

Последовательность дискретных случайных величин $\{X_n\}_{n \geq 0}$ называется простой цепью Маркова, если:

$$\mathbb{P}(X_{n+1} = i_{n+1} | X_n = i_n, \dots, X_0 = i_0) = \mathbb{P}(X_{n+1} = i_{n+1} | X_n = i_n)$$

Область значений случайных величин $\{X_n\}$ называется пространством состояний цепи.

Hopfield Network



Полносвязная сеть.

Каждый нейрон служит входным до обучения, скрытым во время него и выходным после.

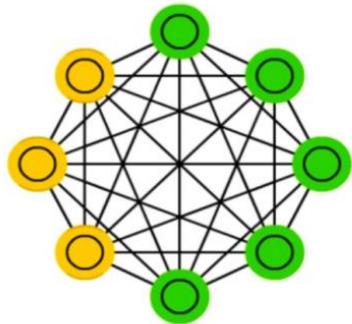
Пусть $S_i(t) \in \{-1, 1\}$ состояние i -ого нейрона в момент t .

Динамика состояния всех нейронов в сети из N нейронов:

$$S(t+1) = \text{sign}(\mathbf{WS}(t)),$$

где матрица $\mathbf{W} \in \mathbb{R}^{N \times N}$ —матрица весовых коэффициентов, описывающая взаимодействия нейронов.

Boltzmann Machine



Аналог скрытых моделей Маркова.

Некоторые нейроны помечены как входные, а некоторые остаются скрытыми.

Нет обратных связей.

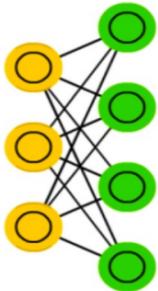
Состояние каждого нейрона выбирается с определенный вероятностью — стохастический нейрон.

Энергия системы такая же, как и у сети Хопфилда:

$$E = -\mathbf{S}^T \mathbf{W} \mathbf{S},$$

где \mathbf{S} — вектор состояний нейронов.

Restricted Boltzmann Machine



Связи существуют только между скрытыми и видимыми нейронами, но при этом отсутствуют между нейронами одного класса.

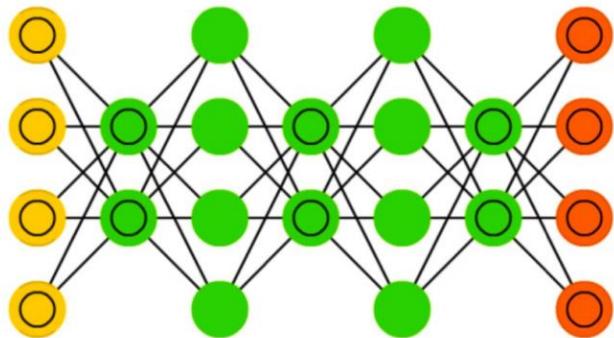
Особенность этой модели в том, что при данном состоянии нейронов одной группы, состояния нейронов другой группы будут независимы друг от друга.

Энергия системы в данном случае:

$$E = -\mathbf{S}_v^T \mathbf{W} \mathbf{S}_h,$$

где $\mathbf{S}_v, \mathbf{S}_h$ — значения видимых и скрытых нейронов.

Deep Belief Network

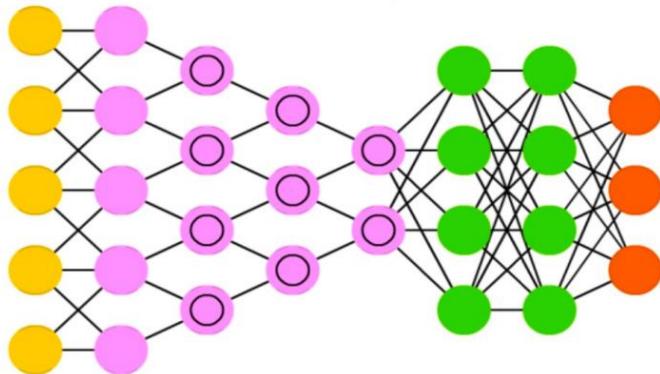


Представляют собой композицию нескольких RBM или VAE.

Скрытый слой каждой подсети служит видимым слоем для следующей.

Обучается методом жадного послойного обучения.

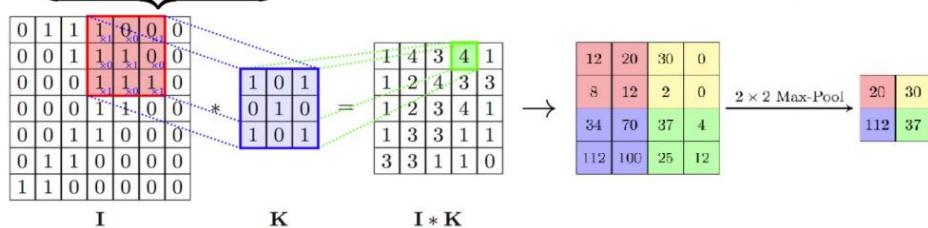
Deep Convolutional Network



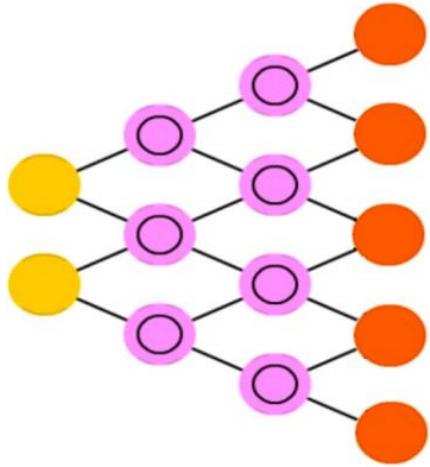
Используются в основном для обработки изображений.

Модель выглядит следующим образом:

Слои Conv → Pool + FFNN (полносвязные слои) + Softmax



Deconvolutional Network

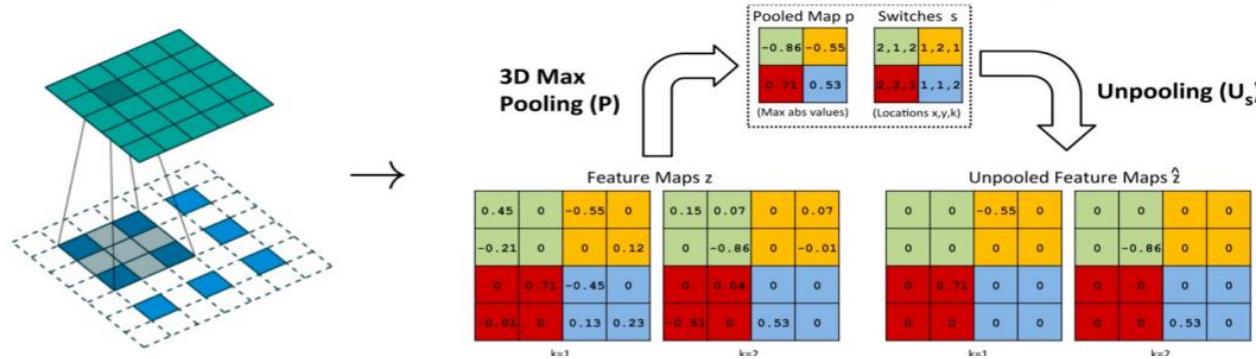


CNN наоборот.

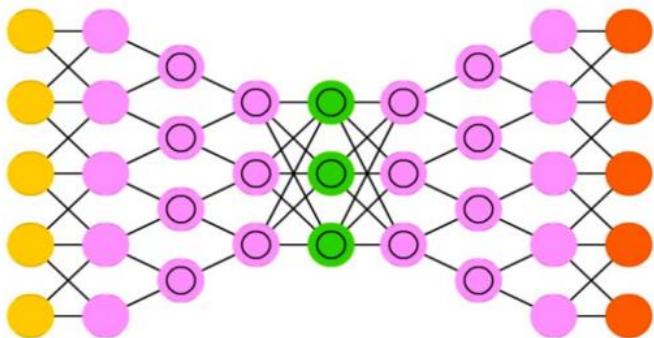
Можно получить большой набор фильтров, которые охватывают всю структуру изображения.

Модель выглядит следующим образом:

FFNN (полносвязные слои) + слои Deconv → Pool/Unpool

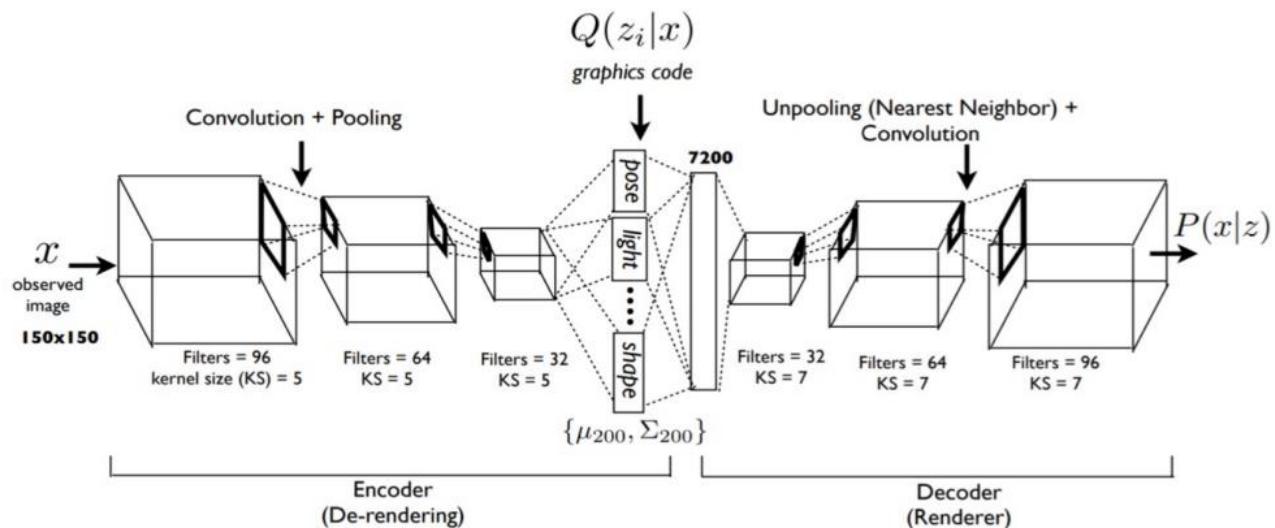


Deep Convolutional Inverse Graphics Network

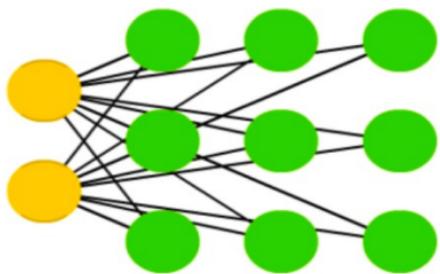


Вариационные автоэнкодеры со сверточными и развертывающими сетями в качестве энкодера и декодера.

Архитектура модели:



Kohonen Network



Обучение без учителя.

Применяются для визуализации многомерных данных.

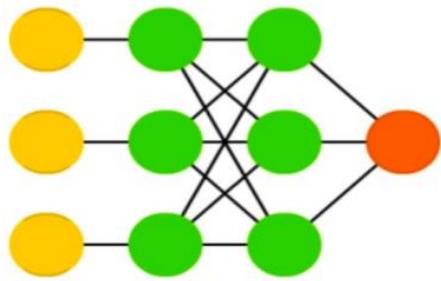
Пусть $\{(m_i, n_j)\}_{i,j=1}^{M,N}$ — узлы сетки.

Каждому узлу сетки приписывается нейрон Кохонена (\mathbf{w}_{mn})

Обучается сеть методом стохастического градиента
(веса инициализируются случайно, η — темп обучения):

- случайно выбирается $\mathbf{x}_i \in X^I$;
- $(m_i, n_i) = \underset{(m,n)}{\operatorname{argmin}} \rho(\mathbf{x}_i, \mathbf{w}_{nm})$;
- $\mathbf{w}_{mn} \rightarrow \mathbf{w}_{mn} = \mathbf{w}_{mn} + \eta (\mathbf{x}_i - \mathbf{w}_{mn}) K(r((m_i, n_i), (m, n)))$

Support Vector Machine



Модель:

$$a(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^h \lambda_i y_i K(\mathbf{x}_i, \mathbf{x}) + w_0 \right)$$

$$K(u, v) = \tanh(k_0 + k_1 \langle u, v \rangle) \rightarrow \sigma.$$

$$K(u, v) = \exp(-\beta \|v - u\|^2) \rightarrow \text{RBF}$$

Машина опорных векторов как двухслойная нейросеть:

