Шпаргалка по всем сетям, их классификация и строгое описание

Жариков Илья Николаевич

Московский физико-технический институт
Факультет управления и прикладной математики
Кафедра интеллектуальных систем

28 сентября 2017 г.

Perceptron

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

Описывается следующей моделью:

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

где $\sigma : \mathbb{R} \to \mathbb{R}$ — 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:

$$a(\mathbf{x}) = \sigma_m\left(\langle \mathbf{w}, \mathbf{u} \rangle\right) \qquad \mathbf{u} = \sigma_h\left(\mathbf{W}\mathbf{x}\right),$$

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

Radial Basis Function

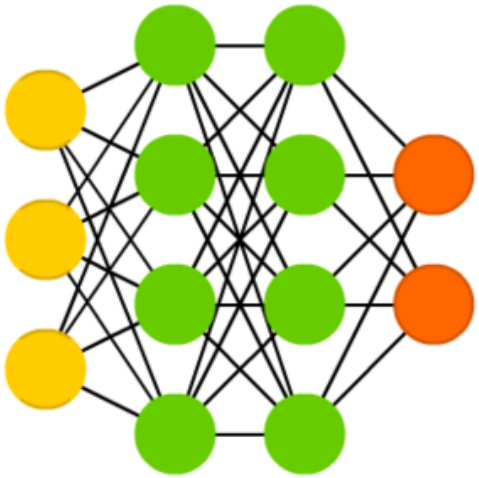
Feed Forward Neural Network, которая использует радиальные базисные функции как функции активации.

Соответственно модель будет такой же:

$$a(\mathbf{x}) = \sigma_m\left(\langle \mathbf{w}, \mathbf{u} \rangle\right) \qquad \mathbf{u} = \sigma_h\left(\mathbf{W}\mathbf{x}\right),$$

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

Deep Feed Forward



Feed Forward Neural Network, которая содержит несколько скрытых слоев.

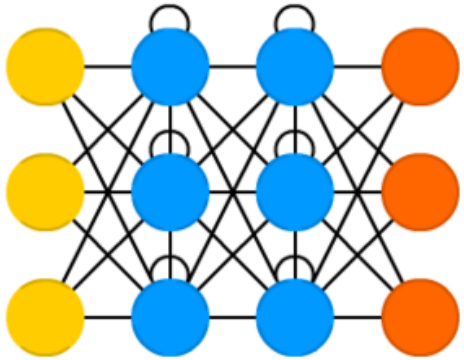
Выходные значения сети $\mathbf{a} \in \mathbb{R}^m$ на объекте $\mathbf{x}$:

$$\mathbf{a}(\mathbf{x}) = \sigma_m\left(\langle \mathbf{w}, \mathbf{u}_2 \rangle\right) \quad \mathbf{u}_2 = \sigma_{h_2}\left(\mathbf{W}_2\mathbf{u}_1\right) \quad \mathbf{u}_1 = \sigma_{h_1}\left(\mathbf{W}_1\mathbf{x}\right),$$

где

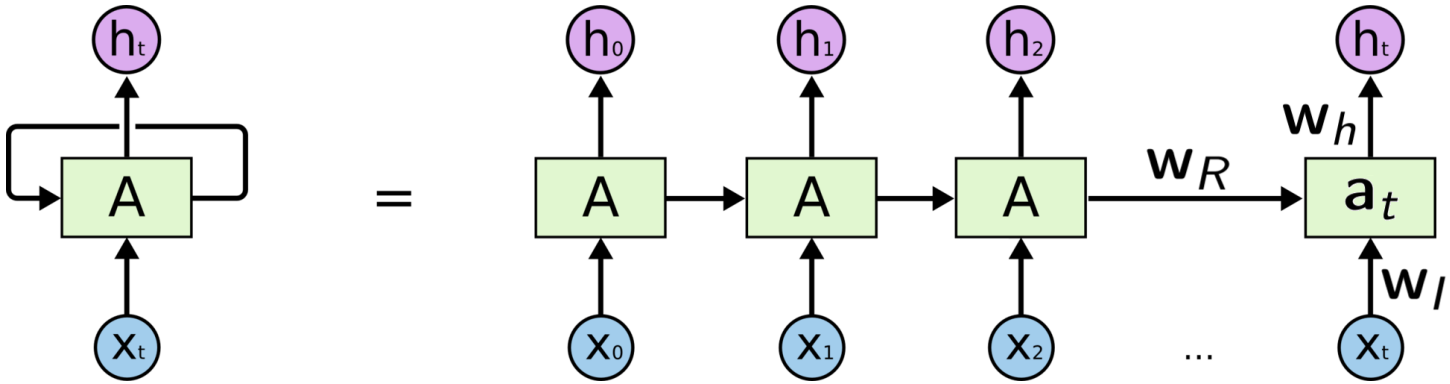
$$\sigma_m : \mathbb{R} \to \mathbb{R}, \qquad \mathbf{w} \in \mathbb{R}^{h_2},$$
$$\sigma_{h_2} : \mathbb{R}^{h_2} \to \mathbb{R}^{h_2}, \qquad \mathbf{W}_2 \in \mathbb{R}^{h_2 \times h_1},$$
$$\sigma_{h_1} : \mathbb{R}^{h_1} \to \mathbb{R}^{h_1}, \qquad \mathbf{W}_1 \in \mathbb{R}^{h_1 \times n}.$$

Recurrent Neural Network



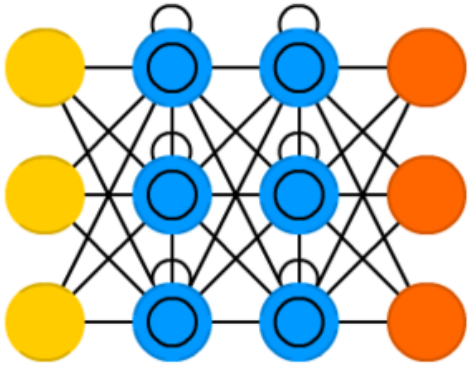
Нейроны получают информацию не только от предыдущего слоя, но и от самих себя в результате предыдущего прохода.

Развернем обратную связь одного нейрона:



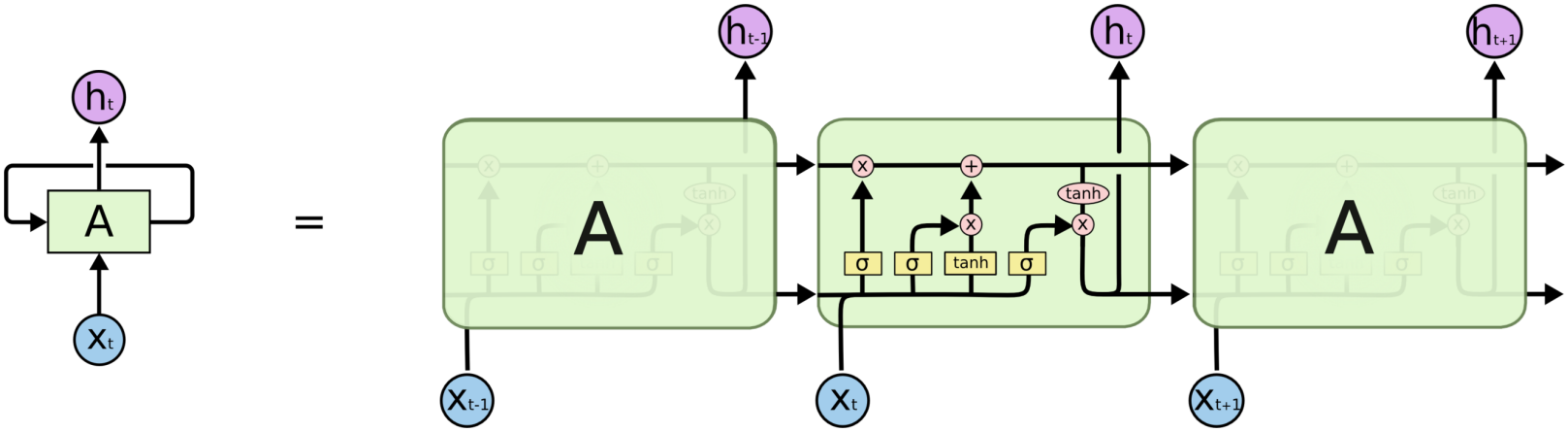
$$\mathbf{h}_t = \sigma_h(\mathbf{W}_h \mathbf{a}_t) \qquad \mathbf{a}_t = \sigma_a(\mathbf{W}_I \mathbf{x}_t + \mathbf{W}_R \mathbf{a}_{t-1})$$

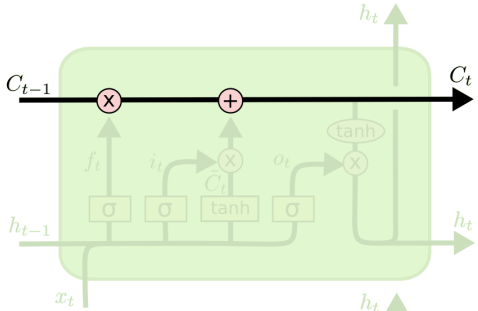
Long / Short Term Memory



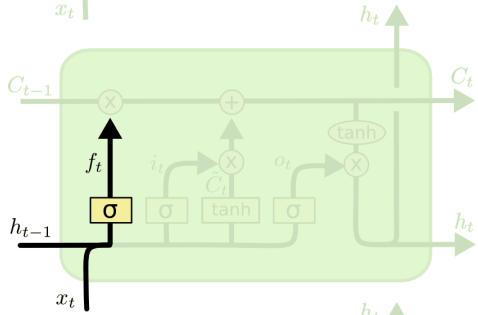
Особая разновидность архитектуры рекуррентных нейронных сетей, способная к обучению долговременным зависимостям.

Повторяющийся модуль в LSTM сети состоит из нескольких взаимодействующих слоев:



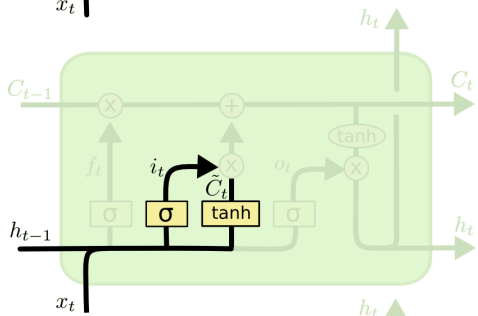


Состояние ячейки (cell state).
Проходит напрямую через всю цепочку.



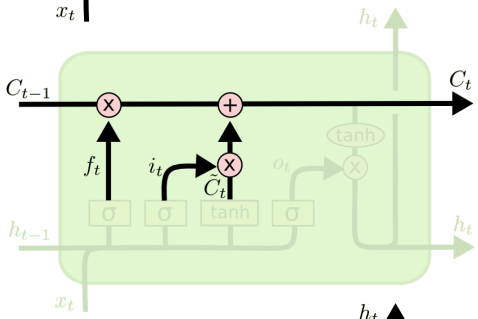
Forget gate layer:

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_f\left[\mathbf{h}_{t-1}, \mathbf{x}_t\right]\right).$$



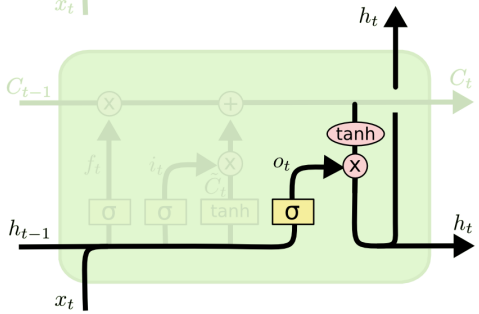
Input gate layer:

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_i\left[\mathbf{h}_{t-1}, \mathbf{x}_t\right]\right),$$
$$\widetilde{\mathbf{C}}_t = \tanh\left(\mathbf{W}_C\left[\mathbf{h}_{t-1}, \mathbf{x}_t\right]\right).$$



Обновление состояния ячейки:

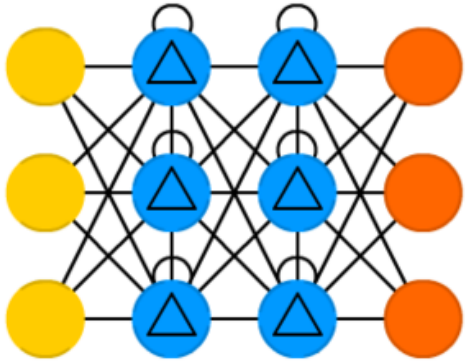
$$\mathbf{C}_t = \mathbf{f}_t * \mathbf{C}_{t-1} + \mathbf{i}_t * \widetilde{\mathbf{C}}_t.$$



Выходные данные:

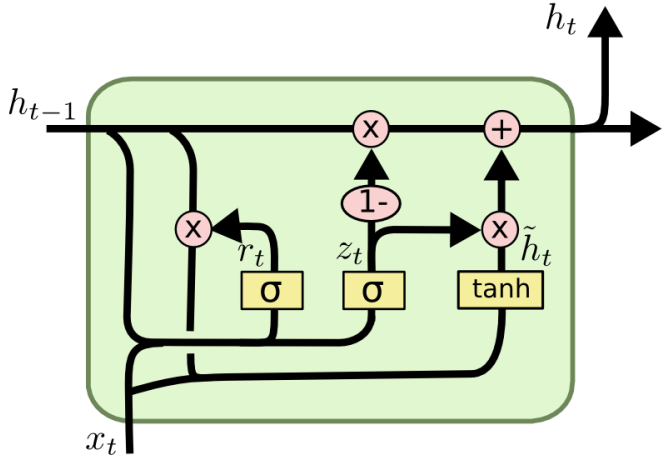
$$\mathbf{o}_t = \sigma\left(\mathbf{W}_o\left[\mathbf{h}_{t-1}, \mathbf{x}_t\right]\right),$$
$$\mathbf{h}_t = \mathbf{o}_t * \tanh\left(\mathbf{C}_t\right).$$

Gated Recurrent Unit



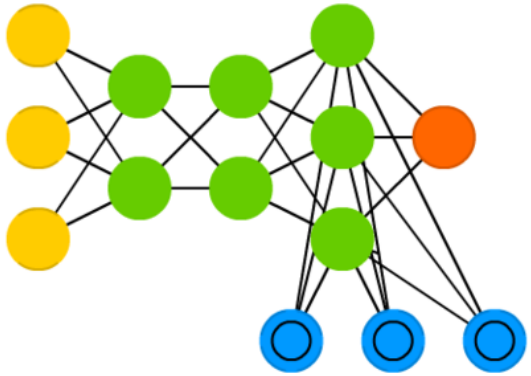
Упрощенная LSTM, в которой фильтры «забывания» и «входа» объединяют в один фильтр «обновления» (update gate).

Повторяющийся модуль в архитектуре GRU сети:



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

Neural Turing Machine

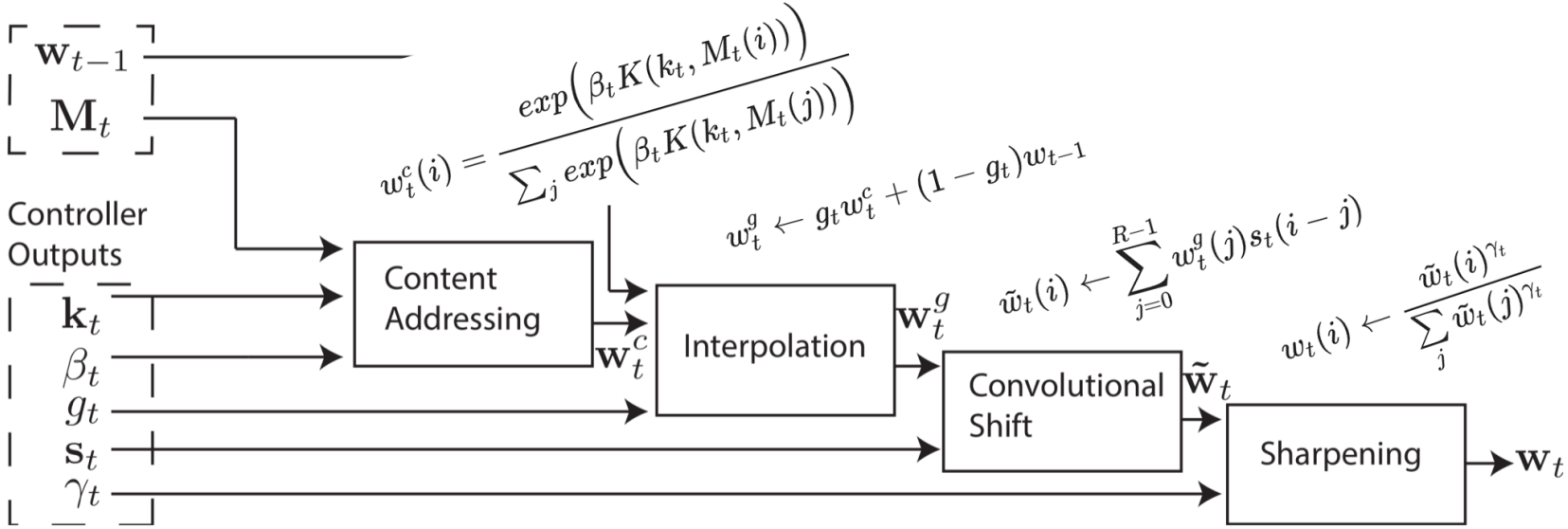


Блок памяти отделен от нейрона.
$\mathbf{M}_t$ — матрица памяти в момент t.
$\mathbf{w}_t$ — весовой вектор (механизм).

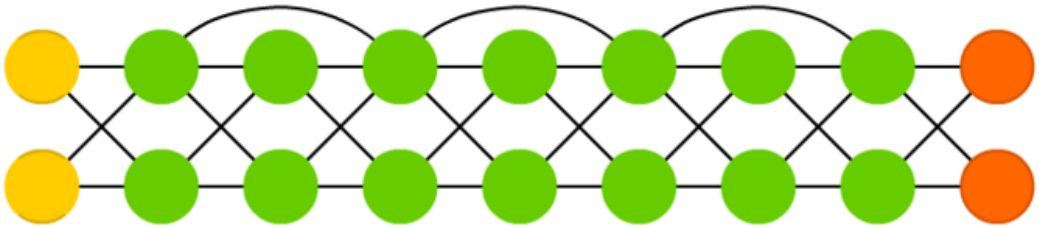
Чтение: $\mathbf{r}_t \leftarrow \mathbf{M}_t^\mathsf{T}\mathbf{w}_t$.

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

Адресация:



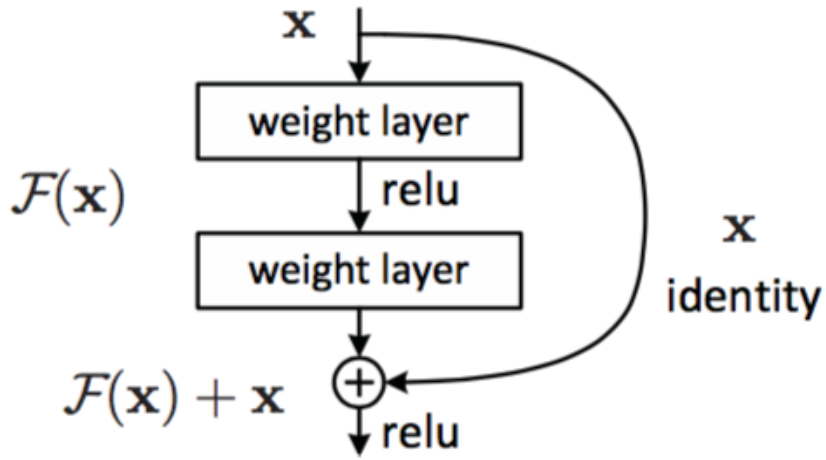
Deep Residual Network



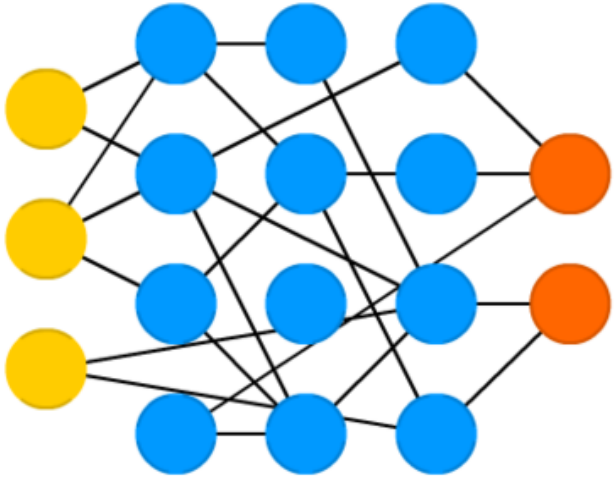
Очень глубокие FFNN с дополнительными связями между слоями.

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

Конструкция основного блока DRN сети:



Echo State Network



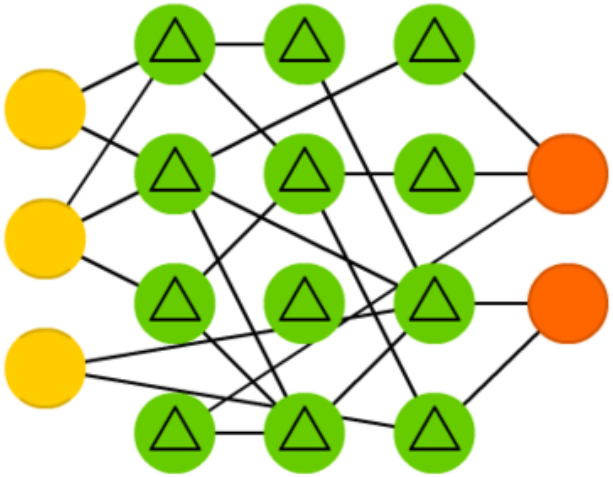
Еще один вид рекуррентных нейросетей.

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



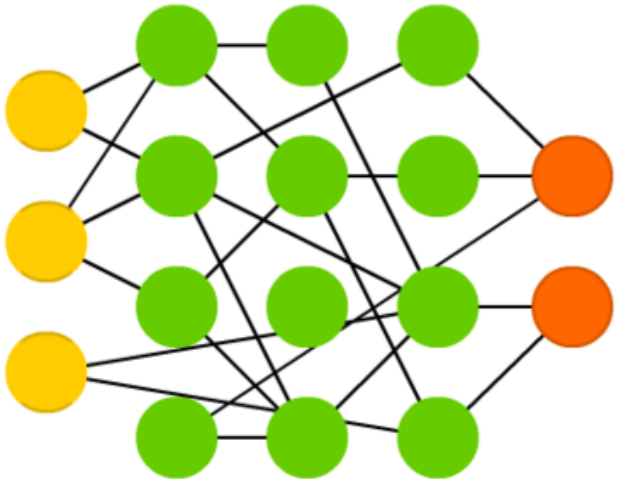
Разновидность импульсных нейронных сетей.

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

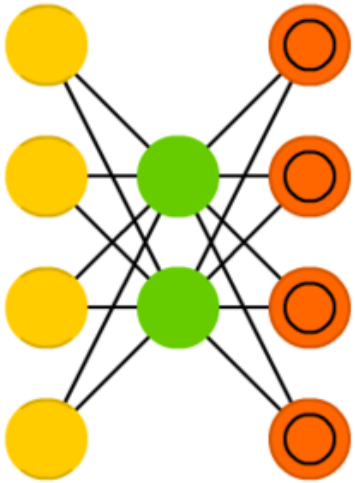


FFNN со случайными связями между нейронами.

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.

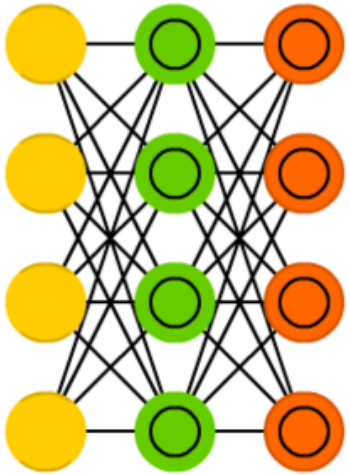
Конструируется таким образом, чтобы не иметь возможность точно скопировать вход на выходе.

Модель имеет следующий вид:

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

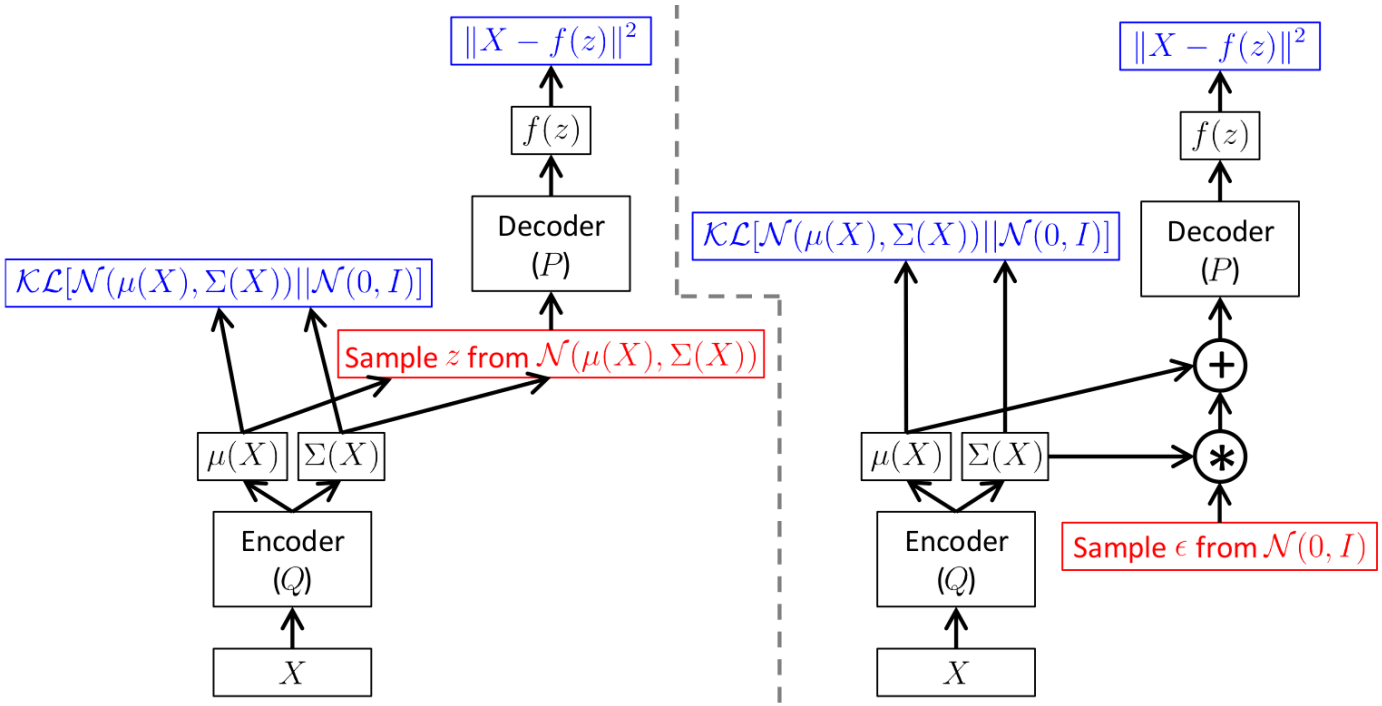
где $\sigma_n : \mathbb{R}^n \to \mathbb{R}^n$, $\sigma_h : \mathbb{R}^h \to \mathbb{R}^h$ — сигмоидные функции активации, веса $\mathbf{W}_m \in \mathbb{R}^{n\times h}$ и $\mathbf{W}_h \in \mathbb{R}^{h\times n}$ $(h < n)$.

Variational Auto Encoder

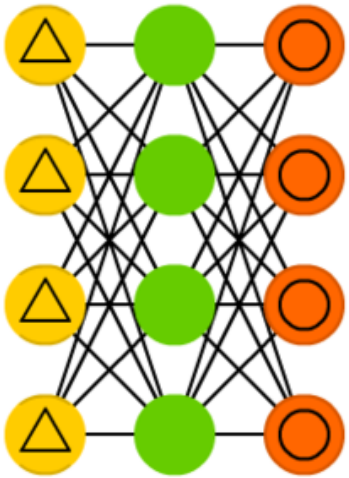


Автоэнкодеры, которые учатся отображать объекты в заданное скрытое пространство и, соответственно, сэмплировать из него.

Более подробная схема:



Denoising Auto Encoder



Автоэнкодер, которому подаем на вход данные с шумом.

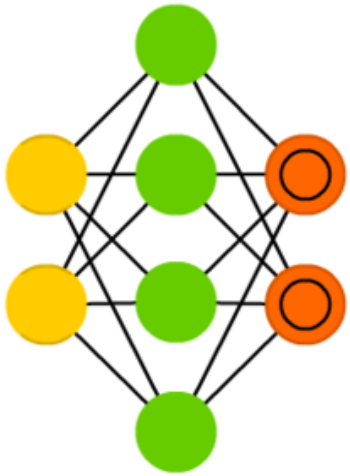
Ошибку вычисляем прежним методом, сравнивая выходной образец с оригиналом без шума.

Модель имеет следующий вид:

$$\mathbf{a}(\mathbf{x}) = \sigma_n\left(\mathbf{W}_n\mathbf{u}\right) \qquad \mathbf{u} = \sigma_h\left(\mathbf{W}_h\widetilde{\mathbf{x}}\right),$$

где $\sigma_n : \mathbb{R}^n \to \mathbb{R}^n$, $\sigma_h : \mathbb{R}^h \to \mathbb{R}^h$ — 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 $\widetilde{\mathbf{x}} \in \mathbb{R}^n$ обозначает зашумленный вектор $\mathbf{x} \in \mathbb{R}^n$.

Sparse Auto Encoder



Антипод автоэнкодера.

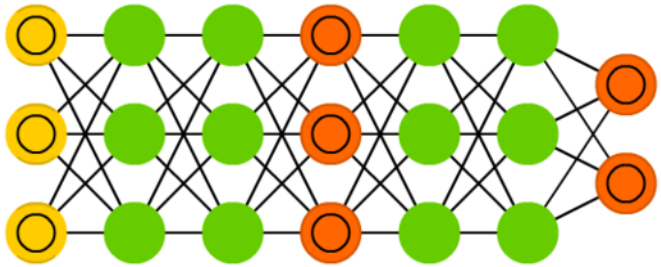
Для обучения требуется ввести штраф за количество активированных нейронов в скрытом слое.

В остальном модель ничем не отличается от модели AE:

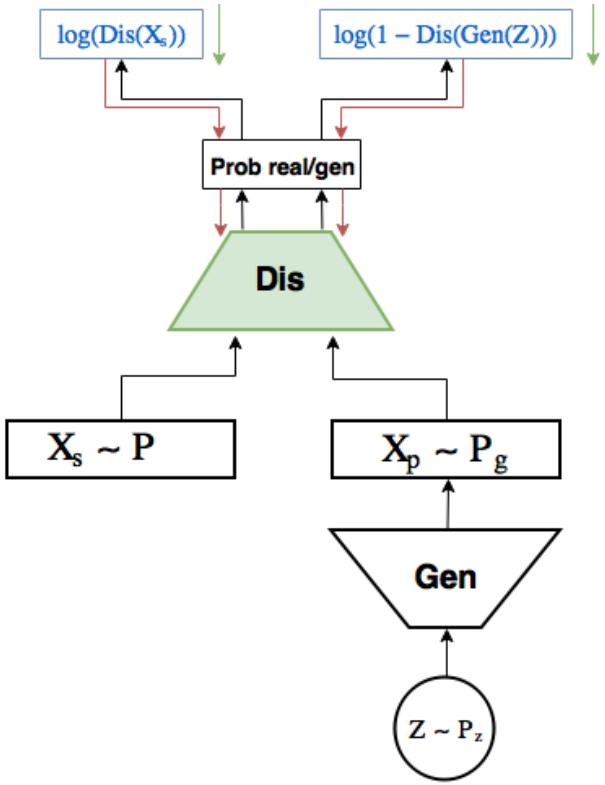
$$\mathbf{a}(\mathbf{x}) = \sigma_n\left(\mathbf{W}_n\mathbf{u}\right) \qquad \mathbf{u} = \sigma_h\left(\mathbf{W}_h\mathbf{x}\right),$$

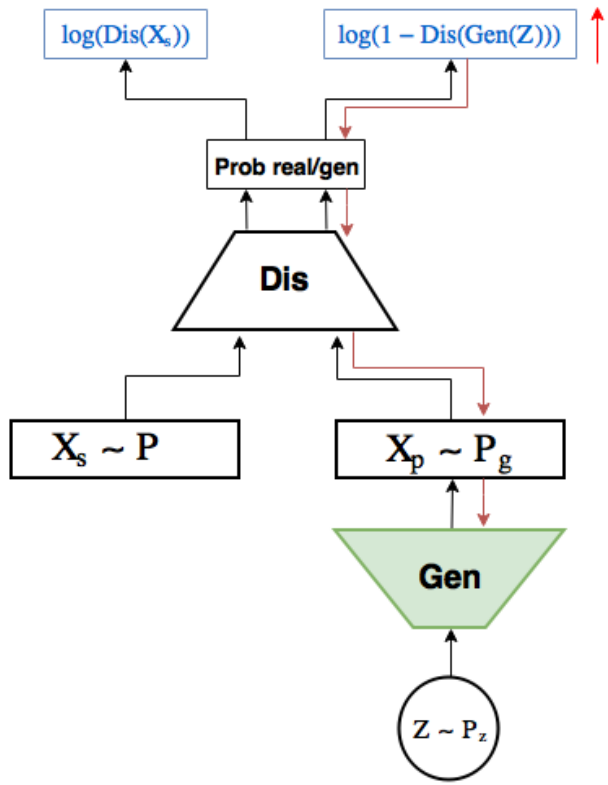
где $\sigma_n : \mathbb{R}^n \to \mathbb{R}^n$, $\sigma_h : \mathbb{R}^h \to \mathbb{R}^h$ — сигмоидные функции активации, веса $\mathbf{W}_m \in \mathbb{R}^{n\times h}$ и $\mathbf{W}_h \in \mathbb{R}^{h\times n}$.

Generative Adversarial Network



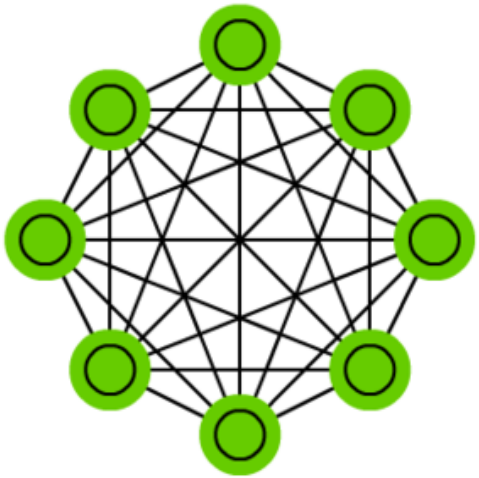
Одна из сетей генерирует данные, а вторая — анализирует.





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

Markov Chain



Задаются вероятности перехода из текущего состояния в соседние.

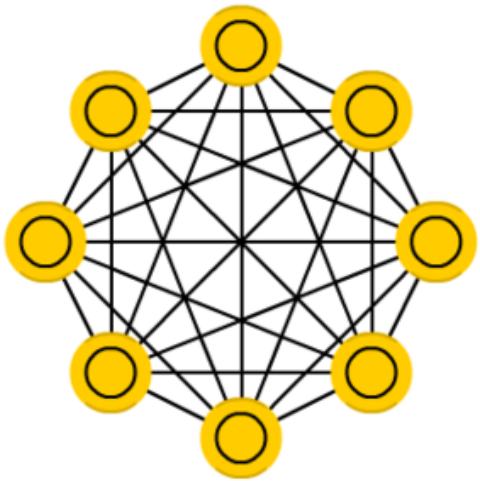
Формирует теоретическую основу для HN и BM.

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

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

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

Hopfield Network



Полносвязная сеть.

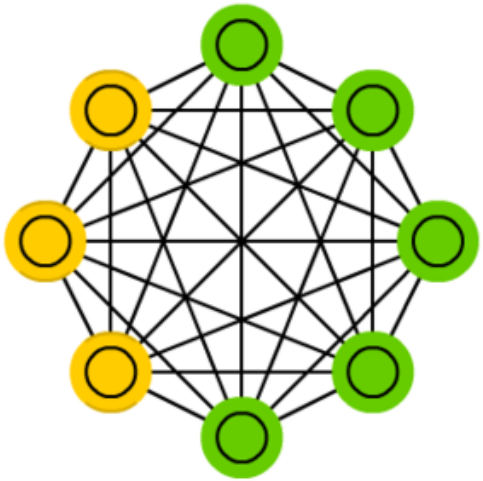
Каждый нейрон служит входным до обучения, скрытым во время него и выходным после.

Пусть $S_i(t) \in \{-1, 1\}$ состояние i-ого нейрона в момент t. Динамика состояния всех нейронов в сети из N нейронов:

$$\mathbf{S}(t+1) = \text{sign}\left(\mathbf{W}\mathbf{S}(t)\right),$$

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.

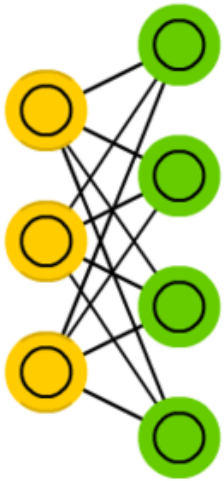
Нет обратных связей.
Состояние каждого нейрона выбирается с определенный вероятностью — стохастический нейрон.

Энергия системы такая же, как и у сети Хопфилда:

$$E = -\mathbf{SWS},$$

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.

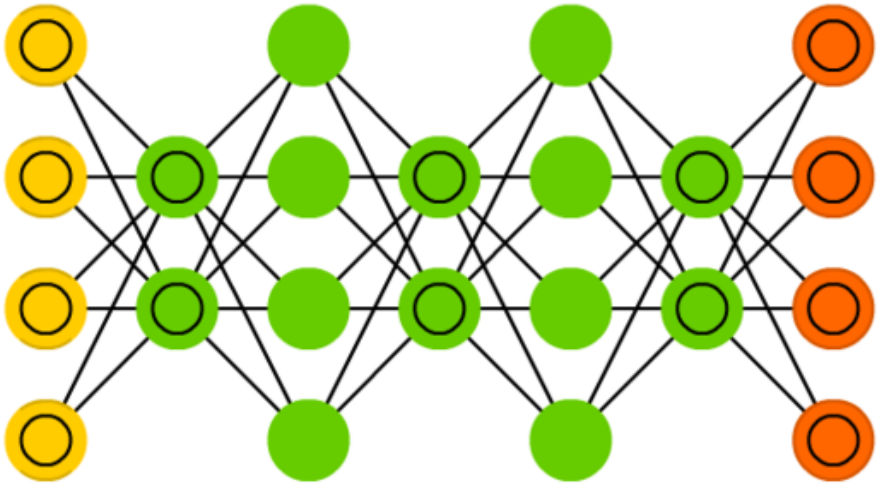
Особенность этой модели в том, что при данном состоянии нейронов одной группы, состояния нейронов другой группы будут независимы друг от друга.

Энергия системы в данном случае:

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

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

Deep Belief Network

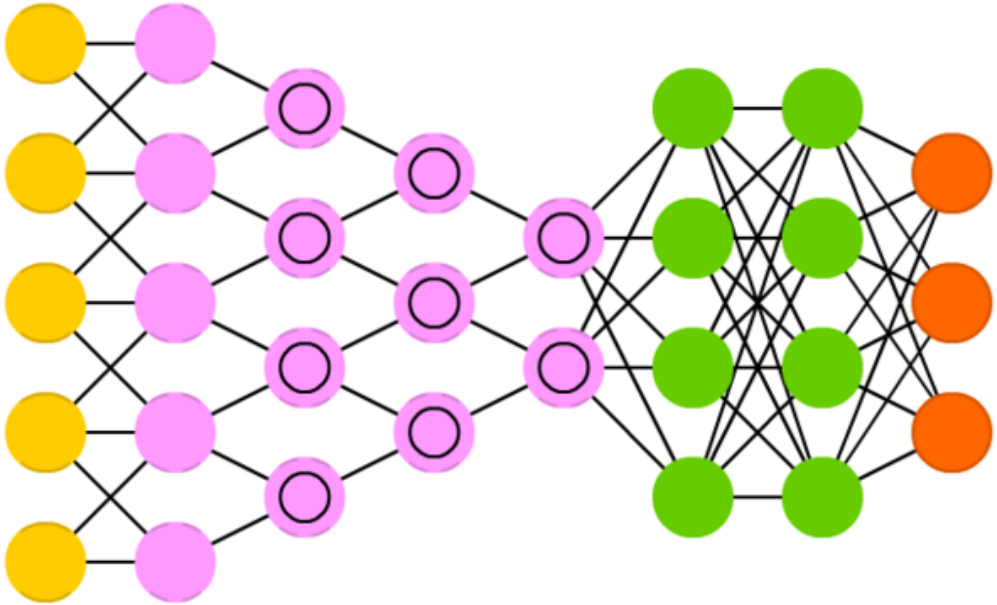


Представляют собой композицию нескольких RBM или VAE.

Скрытый слой каждой подсети служит видимым слоем для следующей.

Обучается методом жадного послойного обучения.

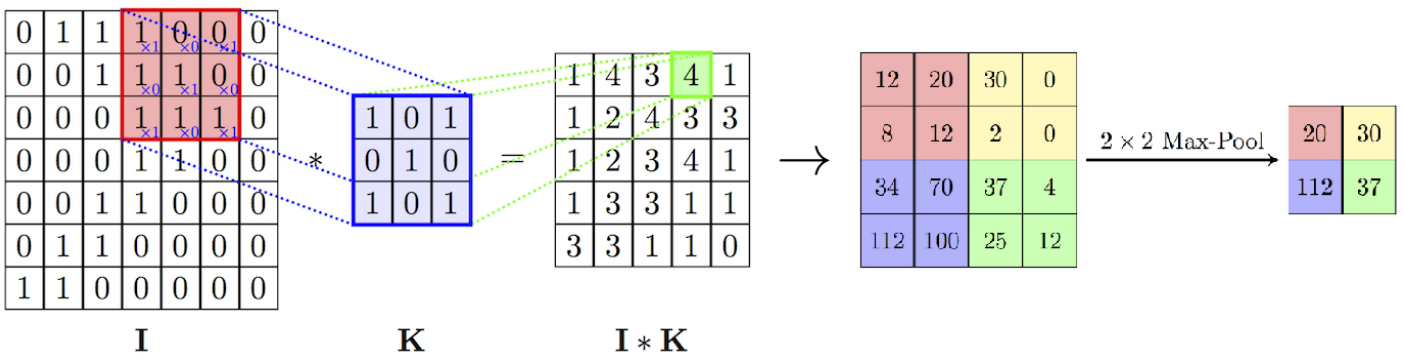
Deep Convolutional Network

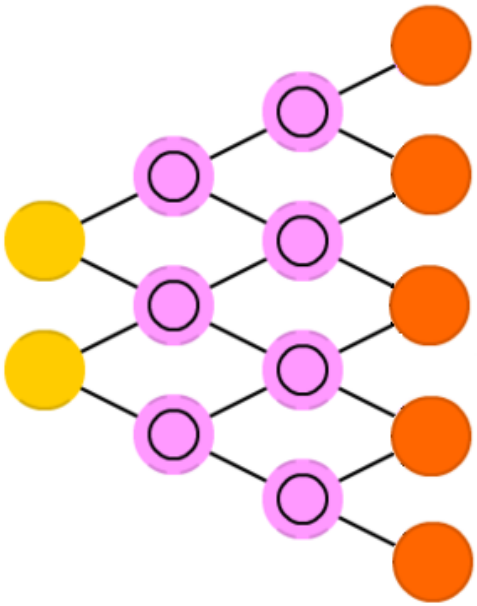


Используются в
основном
для обработки
изображений.

Модель выглядит следующим образом:

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



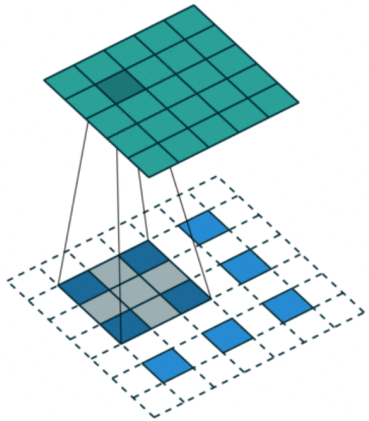


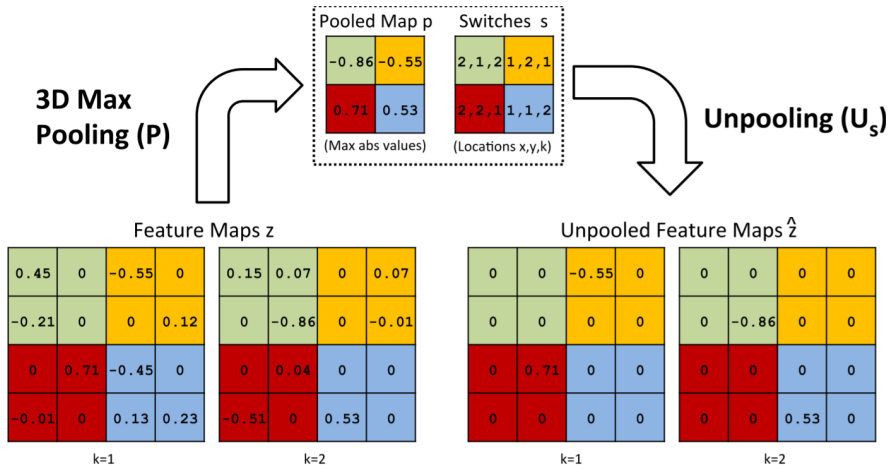
Deconvolutional Network

CNN наоборот.

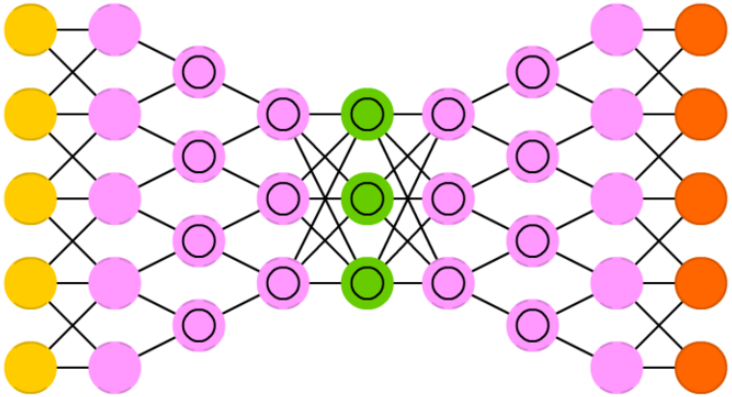
Можно получить большой набор фильтров, которые охватывают всю структуру изображения.

Модель выглядит следующим образом:
FFNN (полносвязные слои) + слои Deconv $\rightarrow$ Pool/Unpool.



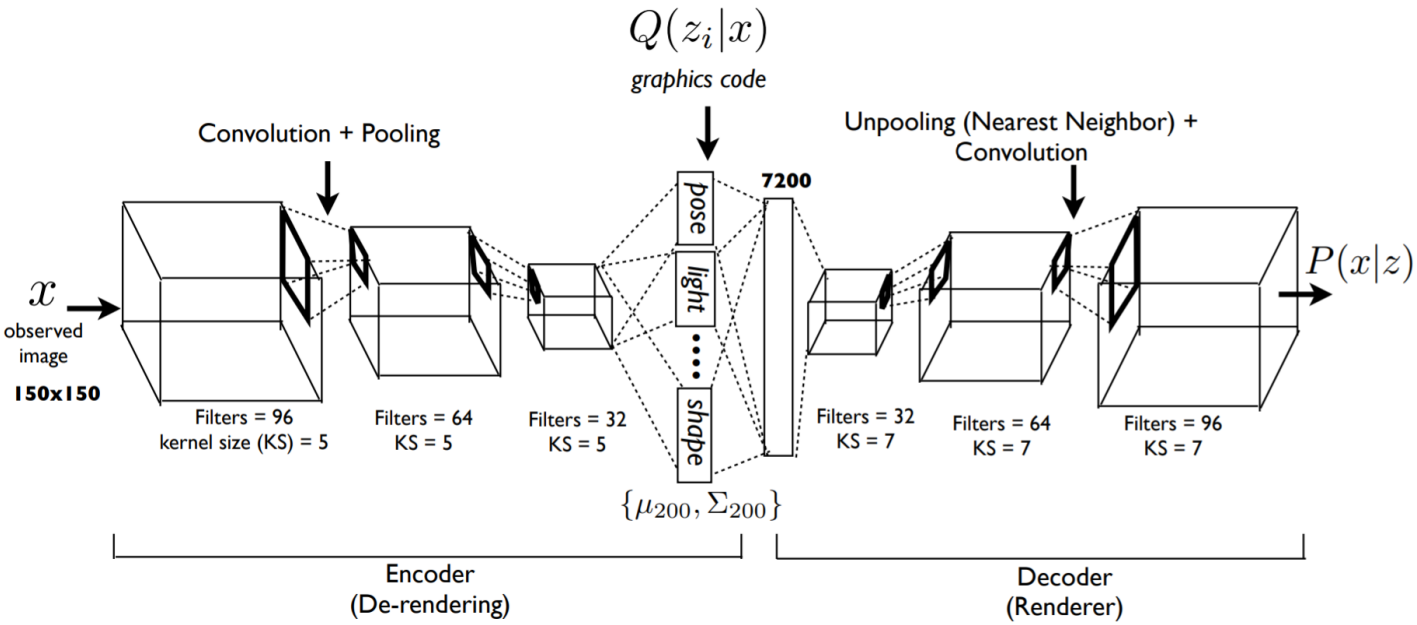


Deep Convolutional Inverse Graphics Network

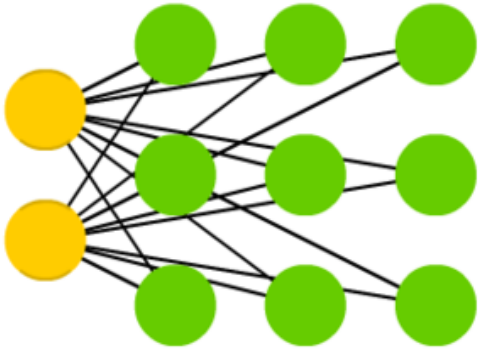


Вариационные автоэнкодеры со сверточными и развертывающими сетями в качестве энкодера и декодера.

Архитектура модели:



Kohonen Network



Обучение без учителя.

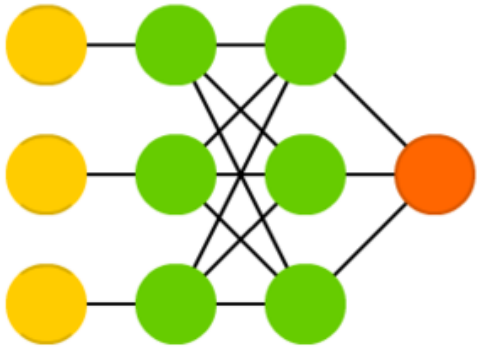
Применяются для визуализации многомерных данных.

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

Support Vector Machine

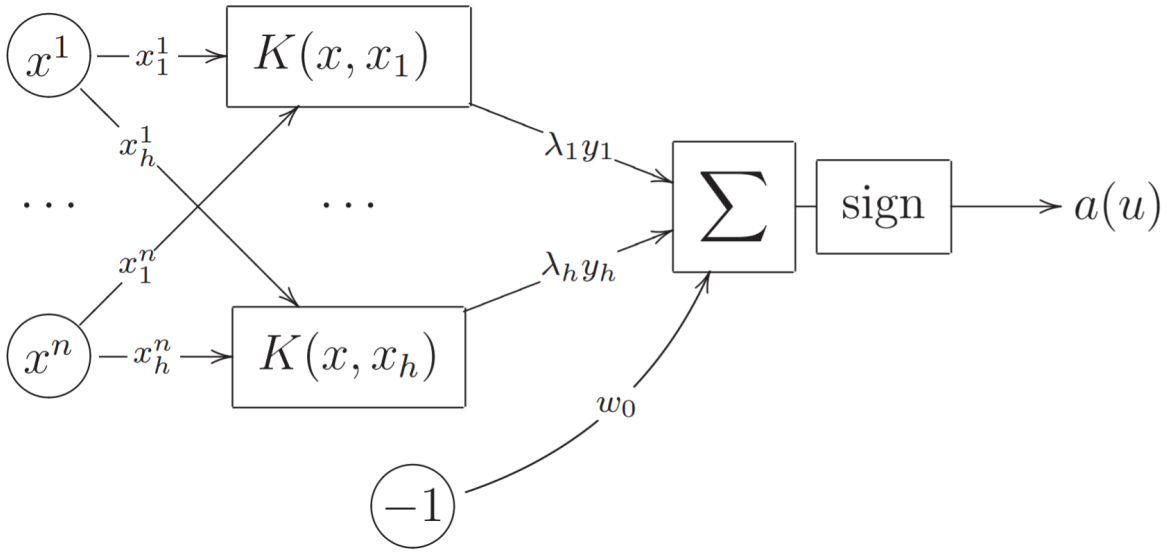


Модель:

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

$$K(u, v) = \tanh\left(k_0 + k_1\langle u, v\rangle\right) \to \sigma.$$
$$K(u, v) = \exp\left(-\beta\|v - u\|^2\right) \to \text{ RBF.}$$

Машина опорных векторов как двухслойная нейросеть:



Полезные ссылки:

- Зоопарк архитектур нейронных сетей.
- Understanding LSTM Networks.
- Explanation of Neural Turing Machines.
- Моделирование нейросети Машина Больцмана.
- Реализация Restricted Boltzmann machine на c#.
- Автоэнкодеры.

Литература:

- Hopfield J. J. Neural networks and physical systems with emergent collective computational abilities //Proceedings of the national academy of sciences. – 1982. – Т. 79. – №. 8. – С. 2554-2558.

- Sejnowski T. Learning and relearning in boltzmann machines //Graphical Models: Foundations of Neural Computation. – 2001. – С. 45.

- Poultney C. et al. Efficient learning of sparse representations with an energy-based model //Advances in neural information processing systems. – 2007. – С. 1137-1144.

- Vincent P. et al. Extracting and composing robust features with denoising autoencoders //Proceedings of the 25th international conference on Machine learning. – ACM, 2008. – С. 1096-1103.

- Bengio Y. et al. Greedy layer-wise training of deep networks //Advances in neural information processing systems. – 2007. – С. 153-160.

- Zeiler M. D. et al. Deconvolutional networks //Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. – IEEE, 2010. – С. 2528-2535.

- Kulkarni T. D. et al. Deep convolutional inverse graphics network //Advances in Neural Information Processing Systems. – 2015. – С. 2539-2547.

- Graves A., Wayne G., Danihelka I. Neural turing machines //arXiv preprint arXiv:1410.5401. – 2014.

- Jaeger H., Haas H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication //science. – 2004. – Т. 304. – №. 5667. – С. 78-80.

- Maass W., Natschläger T., Markram H. Real-time computing without stable states: A new framework for neural computation based on perturbations //Neural computation. – 2002. – Т. 14. – №. 11. – С. 2531-2560.