



ТЕХНОСФЕРА

Лекция n2 Softmax слой Различные топологии

Нестеров Павел

25 марта 2016 г.

План лекции

Вспоминаем прошлую лекцию

Softmax слой

Шаманство

Различные топологии

Радиально базисные сети

Самоорганизующиеся карты Кохонена

Расширяющийся нейронный газ

Автоенкодер

Модифицированная модель нейрона МакКаллока-Питтса

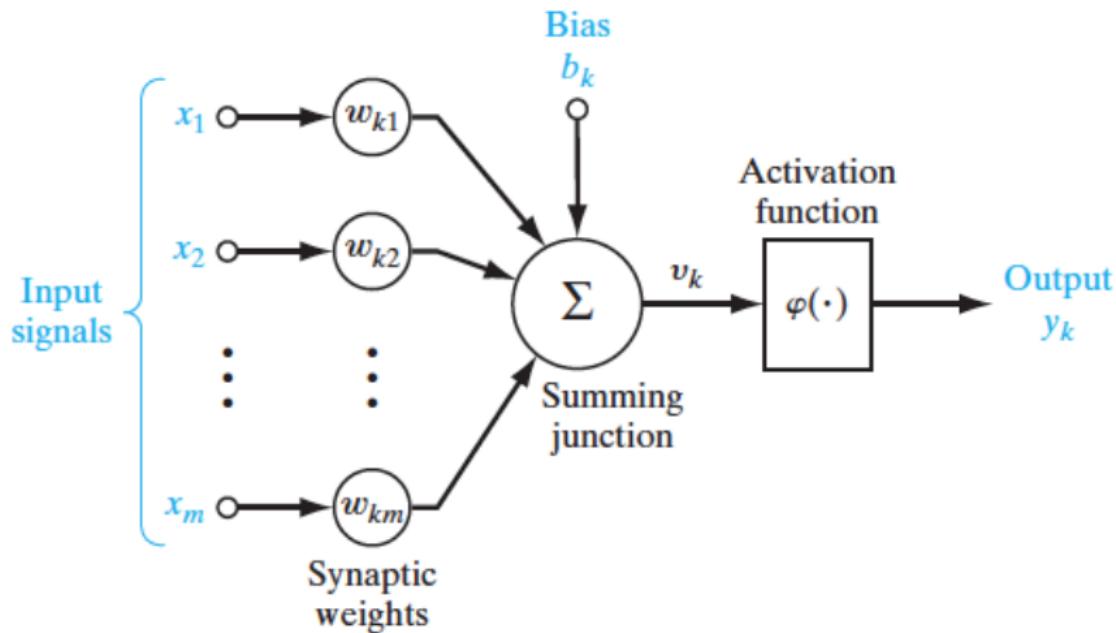


Рис.: Схема искусственного нейрона¹

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

Многослойная нейронная сеть прямого распространения

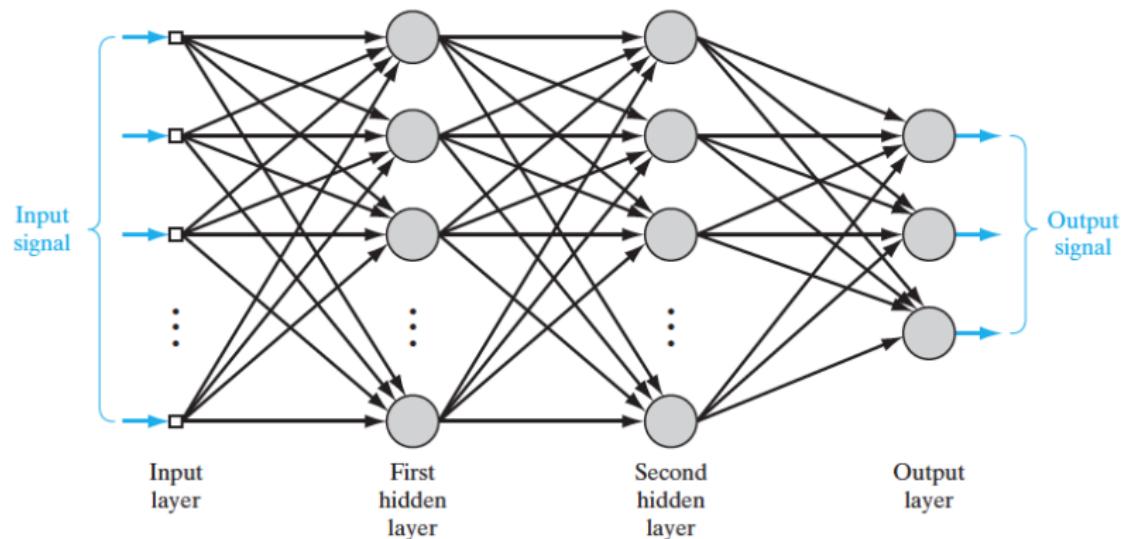


Рис.: Архитектура сети с двумя скрытыми слоями²

²Neural Networks and Learning Machines (3rd Edition), Simon O. Haykin

Алгоритм обратного распространения ошибки

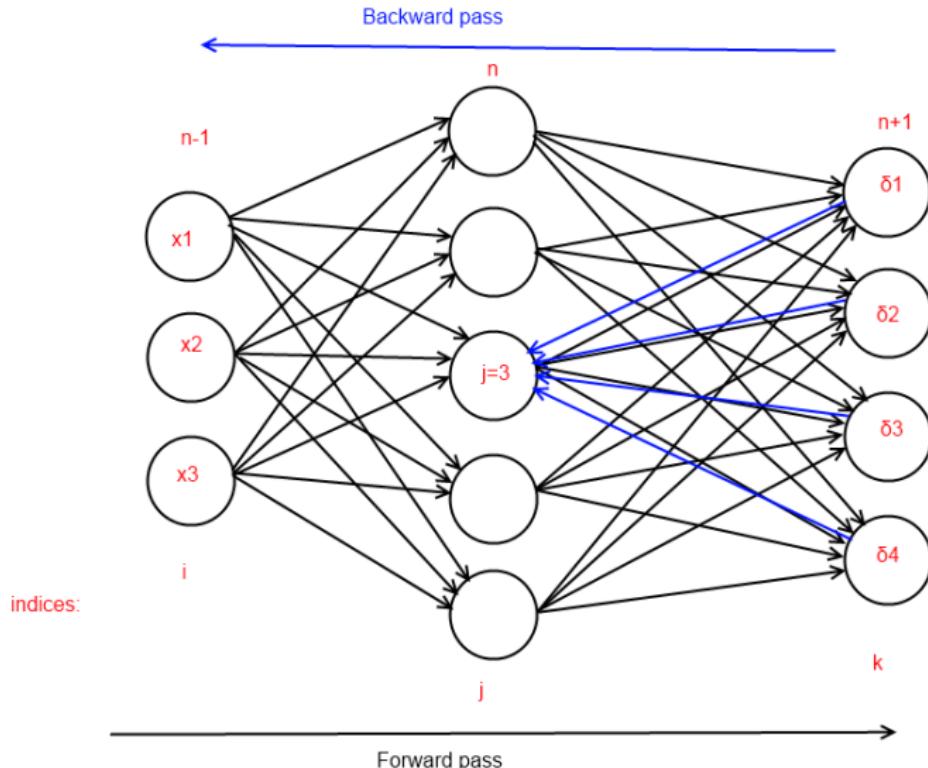


Рис.: Схема прямого (нелинейного) и обратного (линейного) распространения сигнала в сети

Некоторые функции стоимости

Среднеквадратичная ошибка:

- ▶ $E = \frac{1}{2} \sum_{i \in \text{OUTPUT}} (t_i - y_i)^2$
- ▶ $\frac{\partial E}{\partial y_i} = y_i - t_i$

Логарифм правдоподобия Бернулли:

- ▶ $E = - \sum_{i \in \text{OUTPUT}} (t_i \log y_i + (1 - t_i) \log (1 - y_i))$
- ▶ $\frac{\partial E}{\partial y_i} = \frac{t_i}{y_i} - \frac{1 - t_i}{1 - y_i}$

Для каких задач машинного обучения удобны эти функции? А что если необходимо сравнить два вероятностных распределения?

Расстояние Кульбака-Лейблера

Пусть даны две функции распределения, истинная \vec{p} и предполагаемая \vec{q} , одной и той же дискретной случайной величины, принимающей одно из n значений: $p_i = p(x_i)$ и $q_i = q(x_i)$, тогда

$$D_{KL}(\vec{p}, \vec{q}) = \sum_{i=0}^{n-1} p_i \cdot \ln \frac{p_i}{q_i} = \sum_{i=0}^{n-1} p_i \ln p_i - \sum_{i=0}^{n-1} p_i \ln q_i = -H(\vec{p}) + H(\vec{p}, \vec{q}) \quad (1)$$

- ▶ Энтропия: $H(\vec{p}) = -\sum_{i=0}^{n-1} p_i \ln p_i$
- ▶ Перекрестная энтропия: $H(\vec{p}, \vec{q}) = -\sum_{i=0}^{n-1} p_i \ln q_i$

Вспоминаем логистическую регрессию, два класса

- $D = \{(x_i, y_i)\}_{i=1\dots m}, \forall x_i \in \mathbb{R}^n, \forall y_i \in \{0, 1\}$

$$\begin{aligned} P(y = 1|x) &= \frac{P(x|y) \cdot P(y)}{P(x|y) \cdot P(y) + P(x|\bar{y}) \cdot P(\bar{y})} \\ &= \frac{1}{1 + \exp\left(\frac{P(x|y) \cdot P(y)}{P(x|\bar{y}) \cdot P(\bar{y})}\right)} \\ &= \frac{1}{1 + e^{-a}} = \sigma(a) \end{aligned}$$

где \bar{y} это $y = 0$, а так же $P(y = 0) = 1 - P(y = 1)$, $a : h(w) = w^T \cdot x$

- $H(p, q) = -\sum_i p_i \cdot \log q_i = -y \cdot \log \hat{y} - (1 - y) \cdot \log(1 - \hat{y})$
- как увеличить количество классов?

Логистическая регрессия, обобщение на N классов

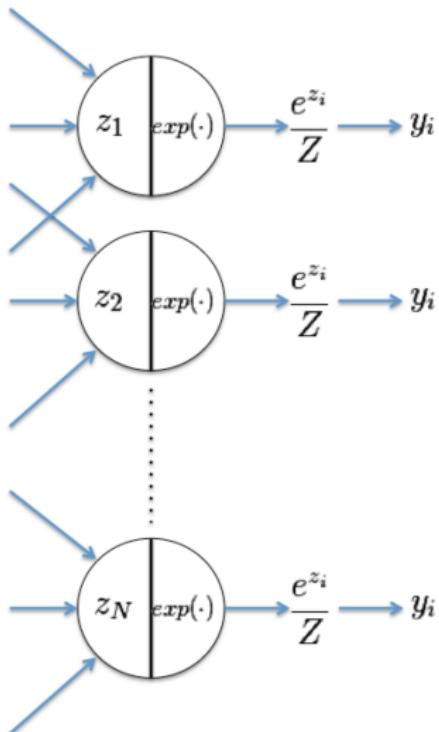
- ▶ $D = \{(x_i, y_i)\}_{i=1\dots m}, \forall x_i \in \mathbb{R}^n, \forall y_i \in \{0, 1, \dots, N\}$
- ▶ $P(y = 1|x) = \frac{1}{1+e^{-a}} = \frac{e^a}{e^a+1} = \frac{1}{Z} \cdot e^a = \frac{1}{Z} \cdot e^{w^T \cdot x}$
- ▶ Z - некоторая нормализующая константа
- ▶ $P(x) = \frac{1}{Z} \cdot e^{-E(x)}$ - распределение Больцмана-Гиббса (почти)

Введем для каждого класса свой вектор весов, получим:

$$P(y = c|x) = \frac{1}{Z} \cdot e^{w_c^T \cdot x} = \frac{e^{w_c^T \cdot x}}{\sum_i e^{w_i^T \cdot x}} \quad (2)$$

- ▶ $\sum_c P(y = c|x) = \sum_c \frac{1}{Z} \cdot e^{w_c^T \cdot x} = \frac{Z}{Z} = 1$
- ▶ как представить в виде нейросети?

Softmax слой



- ▶ $z_j^{(n)} = \sum_{i=0}^{N_{n-1}} w_{ij}^{(n)} x_i^{(n)}$
- ▶ $y_j = \text{SOFTMAX}(z_j) = \frac{e^{z_j}}{Z} = \frac{e^{z_j}}{\sum_k e^{z_k}}$
- ▶ $E(\vec{y}, \vec{t}) = - \sum_{j=1}^{N_n} t_j \cdot \log y_j$
- ▶ $\frac{\partial y_j^{(n)}}{\partial z_j^{(n)}} = ???$

Дифференцирование softmax функции

$$\begin{aligned}\frac{\partial y_j^{(n)}}{\partial z_j^{(n)}} &= \frac{\partial}{\partial z_j} \frac{e^{z_j}}{Z} \\ &= \frac{1}{Z^2} \cdot \left(\frac{\partial e^{z_j}}{\partial z_j} \cdot Z - e^{z_j} \cdot \frac{\partial Z}{\partial z_j} \right) \\ &= \frac{1}{Z^2} \cdot \left(e^{z_j} Z - e^{z_j} \frac{\partial e^{z_j}}{\partial z_j} \right) \\ &= \frac{e^{z_j} Z - (e^{z_j})^2}{Z^2} = \frac{e^{z_j}}{Z} - \left(\frac{e^{z_j}}{Z} \right)^2 \\ &= y_j - y_j^2 \\ &= y_j \cdot (1 - y_j)\end{aligned}$$

Вспомним backprop

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

В итоге получим:

$$\frac{\partial E}{\partial w_{ij}^{(n)}} = x_i^{(n-1)} \frac{\partial E}{\partial z_j^{(n)}} \quad (3)$$

Продолжим с этого момента, с учетом того, что функцией стоимости является перекрестная энтропия (опустим индекс слоя для наглядности):

- ▶ $E(\vec{y}(\vec{z}), \vec{t}) = - \sum_{j=1}^{N_{n-1}} t_j \cdot \log y_j(z_j)$
- ▶ $\frac{\partial E}{\partial z_j} = ???$

Дифференцирование перекрестной энтропии, #1

Раньше было так (для выходного слоя):

$$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial z_j}$$

Теперь стало так:

$$\frac{\partial E}{\partial z_j} = \sum_{i=1}^{N_n} \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial z_j}$$

- ▶ почему так?

Дифференцирование перекрестной энтропии, #2

- ▶ $\frac{\partial E}{\partial z_j} = \sum_{i=1}^{N_n} \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial z_j}$
- ▶ $\frac{\partial E}{\partial y_i}$???

Дифференцирование перекрестной энтропии, #2

► $\frac{\partial E}{\partial z_j} = \sum_{i=1}^{N_n} \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial z_j}$

$$\begin{aligned}\frac{\partial E}{\partial y_i} &= -\frac{\partial}{\partial y_i} \left(\sum_k^{N_n} t_k \cdot \log y_k \right) \\ &= -\frac{\partial}{\partial y_i} (t_i \cdot \log y_i) \\ &= -\frac{t_i}{y_i}\end{aligned}$$

Дифференцирование перекрестной энтропии, #3

- ▶ $\frac{\partial E}{\partial z_j} = \sum_{i=1}^{N_n} \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial z_j}$
- ▶ $\frac{\partial E}{\partial y_i} = -\frac{t_i}{y_i}$
- ▶ $\frac{\partial y_i}{\partial z_j}$???

Дифференцирование перекрестной энтропии, #4

$$\blacktriangleright \frac{\partial E}{\partial z_j} = \sum_{i=1}^{N_n} \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial z_j}$$

$$\blacktriangleright \frac{\partial E}{\partial y_i} = -\frac{t_i}{y_i}$$

$$\frac{\partial y_i}{\partial z_j} = \begin{cases} y_j (1 - y_j), & i = j \\ ???, & i \neq j \end{cases}$$

Дифференцирование перекрестной энтропии, #5

- ▶ $\frac{\partial E}{\partial z_j} = \sum_{i=1}^{N_n} \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial z_j}$
- ▶ $\frac{\partial E}{\partial y_i} = -\frac{t_i}{y_i}$
- ▶ $\frac{\partial y_i}{\partial z_j} = \begin{cases} y_j(1-y_j), & i=j \\ ???, & i \neq j \end{cases}$

$$\begin{aligned}\frac{\partial y_i^{(n)}}{\partial z_j^{(n)}} &= \frac{1}{Z^2} \cdot \left(\frac{\partial e^{z_i}}{\partial z_j} \cdot Z - e^{z_i} \cdot \frac{\partial Z}{\partial z_j} \right) \\ &= \frac{1}{Z^2} (0 - e^{z_i} \cdot e^{z_j}) \\ &= -y_i \cdot y_j\end{aligned}$$

Дифференцирование перекрестной энтропии, #6

- ▶ $\frac{\partial E}{\partial z_j} = \sum_{i=1}^{N_n} \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial z_j}$
- ▶ $\frac{\partial E}{\partial y_i} = -\frac{t_i}{y_i}$
- ▶ $\frac{\partial y_i}{\partial z_j} = \begin{cases} y_j(1-y_j), & i=j \\ -y_i y_j, & i \neq j \end{cases}$
- ▶ $\frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial z_j} = \begin{cases} -t_j(1-y_j), & i=j \\ y_j t_i, & i \neq j \end{cases}$
- ▶ собираем все вместе

Дифференцирование перекрестной энтропии, #7

- ▶ $\frac{\partial E}{\partial z_j} = \sum_{i=1}^{N_n} \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial z_j}$
- ▶ $\frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial z_j} = \begin{cases} -t_j(1-y_j), & i=j \\ y_j t_i, & i \neq j \end{cases}$

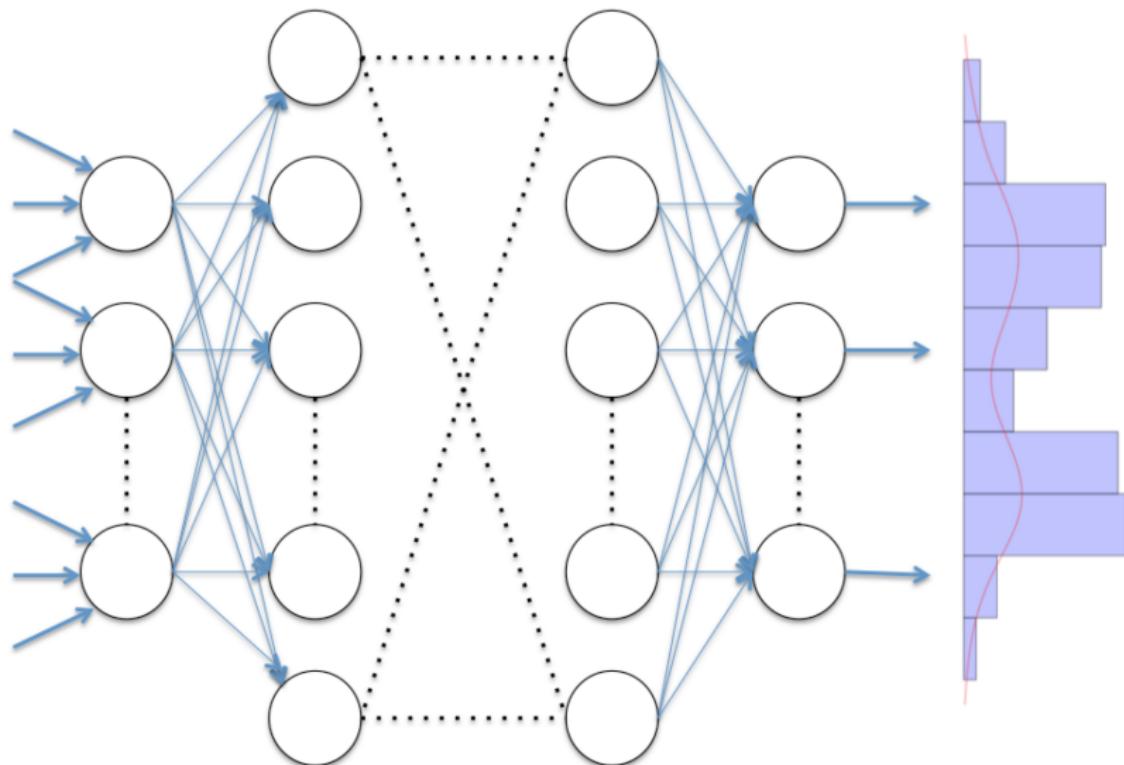
$$\begin{aligned}\frac{\partial E}{\partial z_j} &= -t_j(1-y_j) + \sum_{i=1, i \neq j}^{N_n} y_j t_i \\ &= -t_j + t_j y_j + y_j \cdot \sum_{i=1, i \neq j}^{N_n} t_i \\ &= -t_j + y_j \left(t_j + \cdot \sum_{i=1, i \neq j}^{N_n} t_i \right) \\ &= ???\end{aligned}$$

Дифференцирование перекрестной энтропии, #8

- ▶ $\frac{\partial E}{\partial z_j} = \sum_{i=1}^{N_n} \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial z_j}$
- ▶ $\frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial z_j} = \begin{cases} -t_j(1-y_j), & i=j \\ y_j t_i, & i \neq j \end{cases}$

$$\begin{aligned}\frac{\partial E}{\partial z_j} &= -t_j(1-y_j) + \sum_{i=1, i \neq j}^{N_n} y_j t_i \\ &= -t_j + t_j y_j + y_j \cdot \sum_{i=1, i \neq j}^{N_n} t_i \\ &= -t_j + y_j \left(t_j + \sum_{i=1, i \neq j}^{N_n} t_i \right) \\ &= y_j - t_j\end{aligned}$$

Softmax слой, выводы



Уже рассмотрели

- ▶ регуляризация
- ▶ момент
- ▶ локальная скорость обучения
- ▶ режимы обучения

Перемешивание примеров

- ▶ перемешивать датасет перед новой эпохой³
- ▶ стараться делать такие батчи, которые содержат данные различных классов
- ▶ иногда есть смысл чаще представлять нейросети экземпляры, которые генерируют наибольшее значение ошибки, относительно остальных⁴

³эпоха - полный прогон датасета, хотя возможны и другие варианты семплирования данных для батча (семпла в случае online learning)

⁴следует быть аккуратным, если данные содержат выбросы

Предобработка данных

- ▶ нормализация
- ▶ вычитание среднего
- ▶ масштабирование дисперсии к 1
- ▶ в случае некореллированных признаков можно привести ковариацию к одному значению
- ▶ декорелляция данных

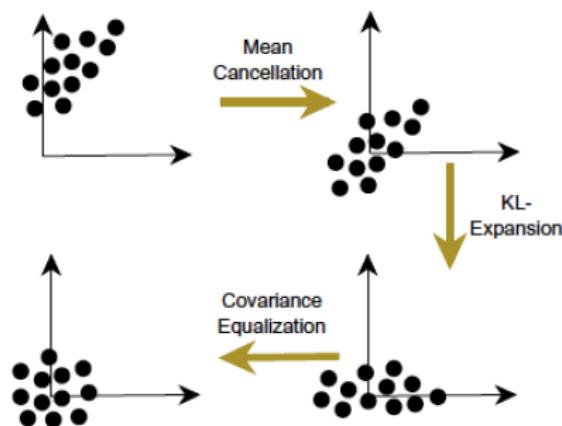


Рис.: Полные процесс предобработки⁵

⁵Efficient BackProp, Yann A. LeCun, Léon Bottou, и другие

Batch Normalization, #1

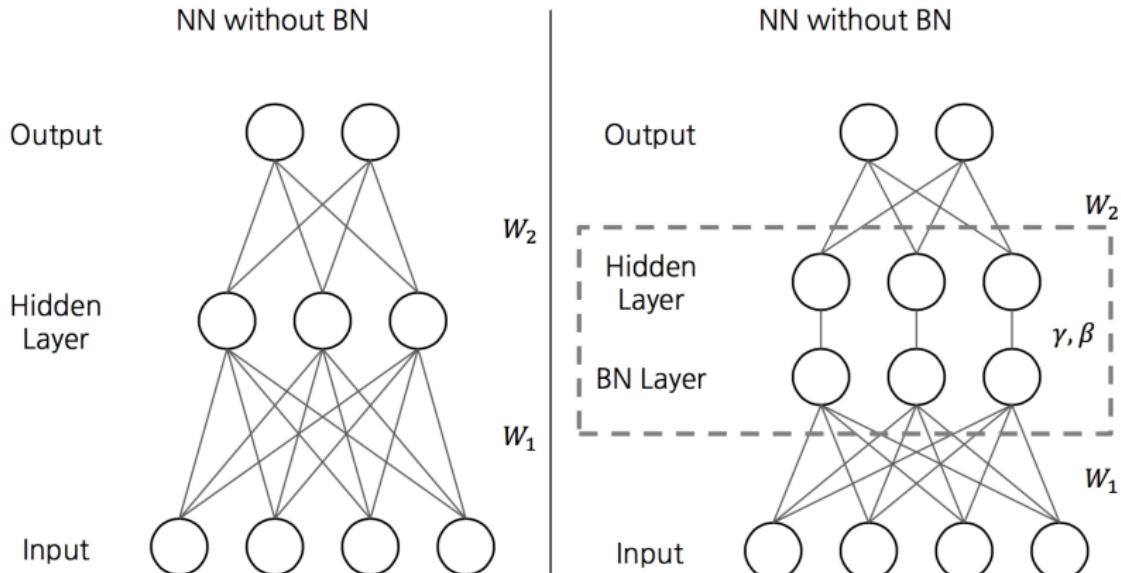


Рис.: Слой BN⁶

⁶<http://arxiv.org/pdf/1502.03167v3.pdf>

Batch Normalization, #2

Input:

- ▶ Input Batch: $\mathcal{B} = \{\vec{x}_1, \dots, \vec{x}_m\}$
- ▶ Parameters to be learned: $\vec{\gamma}, \vec{\beta}$

Output:

- ▶ $\vec{\mu}_{\mathcal{B}} = \frac{1}{m} \sum_{i=1}^m \vec{x}_i$
- ▶ $\vec{\sigma}_{\mathcal{B}}^2 = \frac{1}{m} \sum_{i=1}^m (\vec{x}_i - \vec{\mu}_{\mathcal{B}})^2$
- ▶ $\vec{x}_i^{\text{new}} = \frac{\vec{x}_i - \vec{\mu}_{\mathcal{B}}}{\sqrt{\vec{\sigma}_{\mathcal{B}}^2 + \vec{\epsilon}}}$
- ▶ $\vec{y}_i^{\text{new}} = \vec{\gamma} \cdot \vec{x}_i^{\text{new}} + \vec{\beta} \equiv \mathbf{BN}_{\vec{\gamma}, \vec{\beta}}(\vec{x}_i)$

Скорость обучения, #1

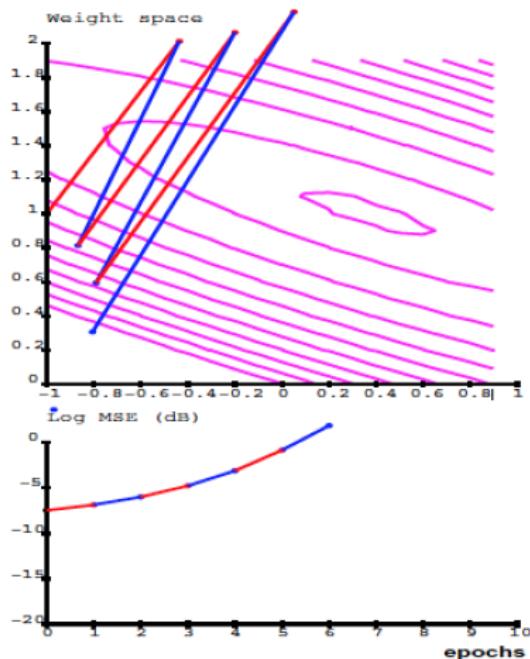
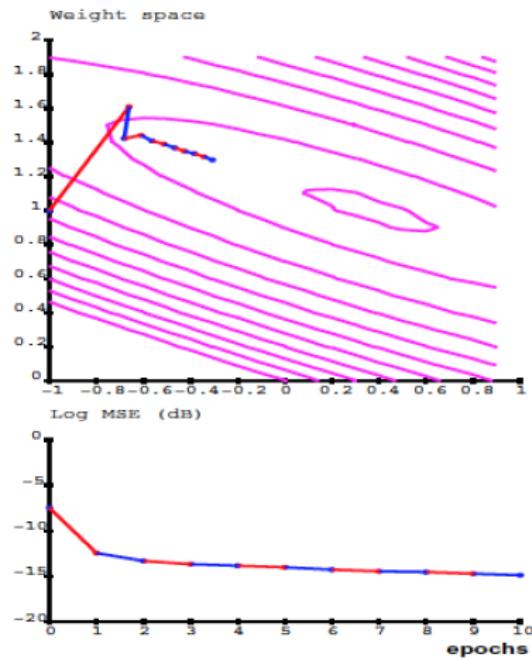


Рис.: Скорость 1.5 и 2.5 соответственно⁷

⁷Efficient BackProp, Yann A. LeCun, Léon Bottou, и другие

Скорость обучения, #2

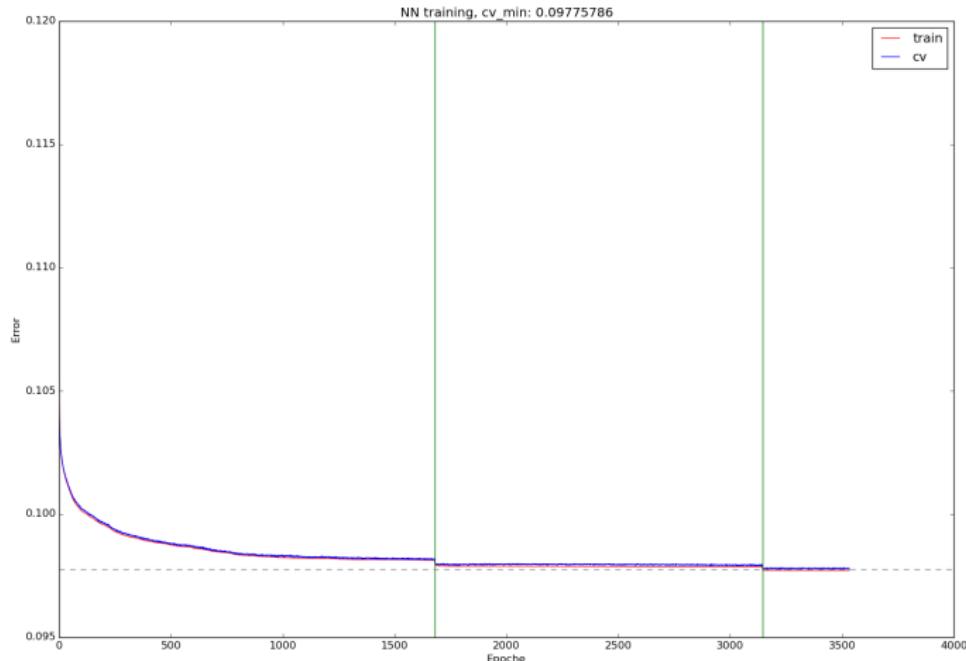


Рис.: Адаптивная скорость обучения и ранняя остановка

Скорость обучения, #3

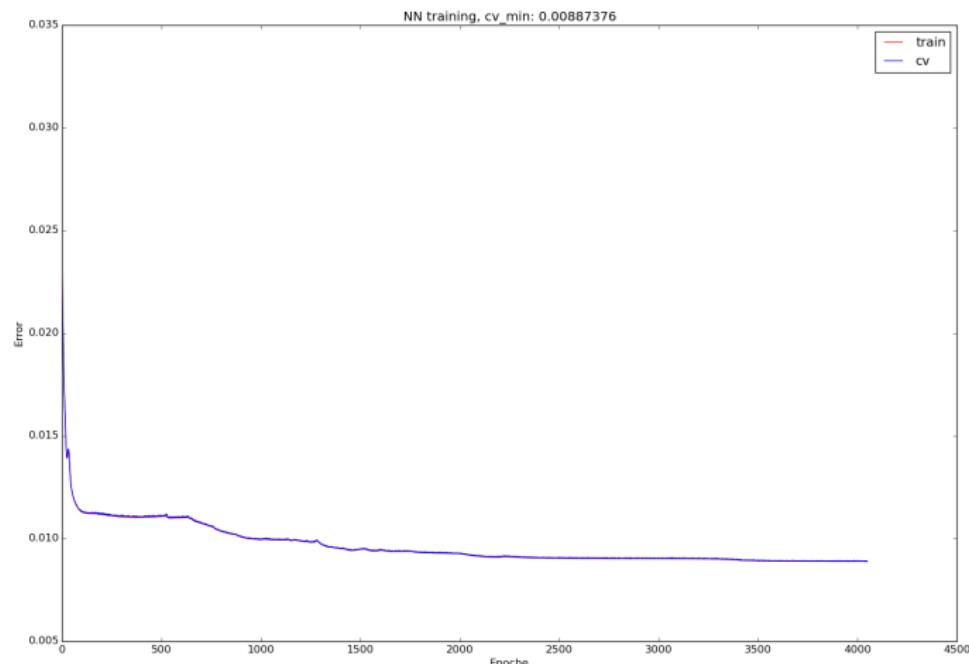


Рис.: Динамика ошибки

Инициализация

Предположим, что:

- ▶ функция активации - логистический сигмоид
- ▶ данные стандартизированы ($\mu = 0, \sigma = 1$)

тогда есть смысл, что бы это распределение сохранялось и внутри сети на скрытых слоях, самый простой способ подойти к этому - инициализировать особым образом веса:

- ▶ $\sigma_{y_j} = (\sum_i w_{ij}^2)^{\frac{1}{2}}$
- ▶ $\sigma_{y_j} \approx 1 \Rightarrow \sigma_w = m^{-\frac{1}{2}}$, где m - количество входов в нейрон⁸

⁸Efficient BackProp, Yann A. LeCun, Léon Bottou, и другие

К вопросу о топологии



Рис.: Inception GoogLeNet⁹

⁹Going Deeper with Convolutions (Christian Szegedy, Wei Liu, ...)

k-means кластеризация, #1

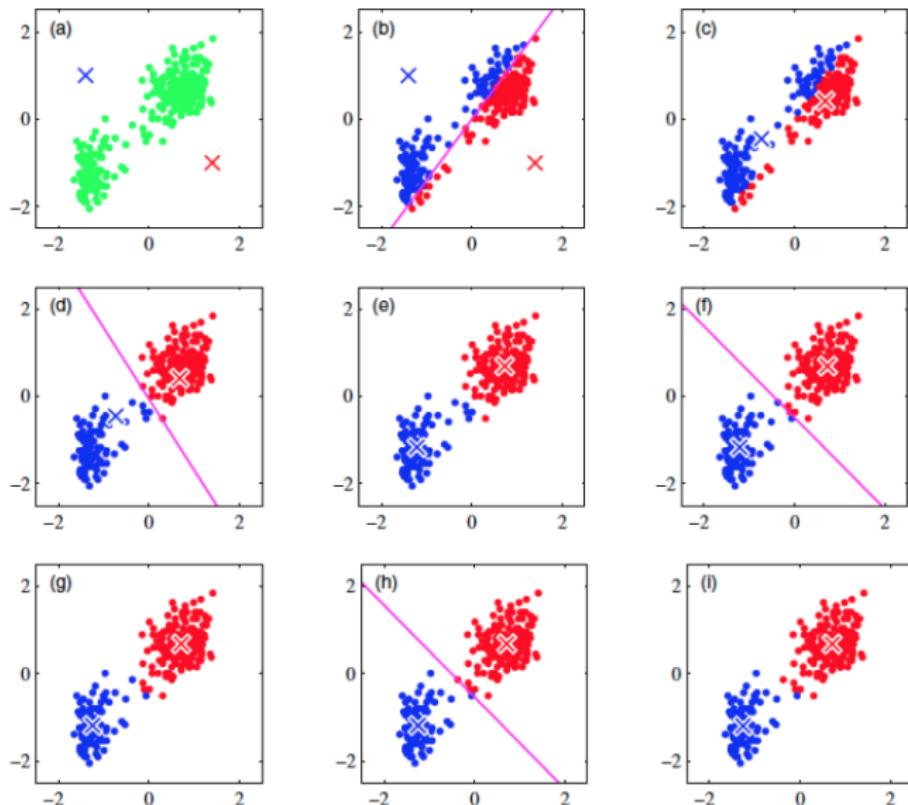


Рис.: Случай $k=2$

k-means кластеризация, #2

Введем обозначения:

- ▶ $C = \{\vec{c}_1, \vec{c}_2, \dots, \vec{c}_k\}$ - множество центроидов,
 $\vec{c}_i = \{c_{i1}, c_{i2}, \dots, c_{in}\}$
- ▶ $X = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_m\}$ - исходное множество данных,
 $\vec{x}_i = \{x_{i1}, x_{i2}, \dots, x_{in}\}$
- ▶ $X_i \in X$ - i -ый кластер, $i = \{1, 2, \dots, k\}$
- ▶ $d : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ - некоторая мера расстояния

Тогда алгоритм k-means минимизирует следующий функционал:

- ▶ *какой?*

k-means кластеризация, #3

Введем обозначения:

- ▶ $C = \{\vec{c}_1, \vec{c}_2, \dots, \vec{c}_k\}$ - множество центроидов,
 $\vec{c}_i = \{c_{i1}, c_{i2}, \dots, c_{in}\}$
- ▶ $X = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_m\}$ - исходное множество данных,
 $\vec{x}_i = \{x_{i1}, x_{i2}, \dots, x_{in}\}$
- ▶ $X_i \subset X$ - i -ый кластер, $X_i \cap X_j = \emptyset$, $i, j = \{1, 2, \dots, k\}$
- ▶ $d : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ - некоторая мера расстояния

Тогда алгоритм k-means минимизирует следующий функционал:

- ▶ $E = \sum_{k=1}^{|C|} \sum_{i=1}^{|X_k|} d^2(\vec{x}_i, \vec{c}_k)$
- ▶ *почему нельзя просто взять и проинтегрировать все это дело?*

k-means кластеризация, #4

Введем обозначения:

- ▶ $C = \{\vec{c}_1, \vec{c}_2, \dots, \vec{c}_k\}$ - множество центроидов,
 $\vec{c}_i = \{c_{i1}, c_{i2}, \dots, c_{in}\}$
- ▶ $X = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_m\}$ - исходное множество данных,
 $\vec{x}_i = \{x_{i1}, x_{i2}, \dots, x_{in}\}$
- ▶ $X_i \in X$ - i -ый кластер, $i = \{1, 2, \dots, k\}$
- ▶ $d : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ - некоторая мера расстояния

Тогда алгоритм k-means минимизирует следующий функционал:

- ▶ $E = \sum_{k=1}^{|C|} \sum_{i=1}^{|X_k|} d^2(\vec{x}_i, \vec{c}_k) = \sum_{i=1}^{|X|} \min_{j=1}^{|C|} d^2(\vec{x}_i, \vec{c}_j)$
- ▶ $\frac{\partial \min_{j=1}^{|C|} d^2(\vec{x}_i, \vec{c}_j)}{\partial c_{ab}} = ???$

k-means кластеризация, #5

Введем обозначения:

- ▶ $C = \{\vec{c}_1, \vec{c}_2, \dots, \vec{c}_k\}$ - множество центроидов,
 $\vec{c}_i = \{c_{i1}, c_{i2}, \dots, c_{in}\}$
- ▶ $X = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_m\}$ - исходное множество данных,
 $\vec{x}_i = \{x_{i1}, x_{i2}, \dots, x_{in}\}$
- ▶ $X_i \in X$ - i -ый кластер, $i = \{1, 2, \dots, k\}$
- ▶ $d : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ - некоторая мера расстояния

Тогда алгоритм k-means минимизирует следующий функционал:

$$\begin{aligned} & \blacktriangleright E = \sum_{k=1}^{|C|} \sum_{i=1}^{|X_k|} d^2(\vec{x}_i, \vec{c}_k) = \sum_{i=1}^{|X|} \min_{j=1}^{|C|} d^2(\vec{x}_i, \vec{c}_j) \\ & \blacktriangleright \frac{\partial \min_{j=1}^{|C|} d^2(\vec{x}_i, \vec{c}_j)}{\partial c_{ab}} = \begin{cases} 0, \min_{j=1}^{|C|} d^2(\vec{x}_i, \vec{c}_j) \neq d^2(\vec{x}_i, \vec{c}_a) \\ \frac{\partial d^2(\vec{x}_i, \vec{c}_a)}{\partial c_{ab}}, \min_{j=1}^{|C|} d^2(\vec{x}_i, \vec{c}_j) = d^2(\vec{x}_i, \vec{c}_a) \end{cases} \end{aligned}$$

k-means кластеризация, #6

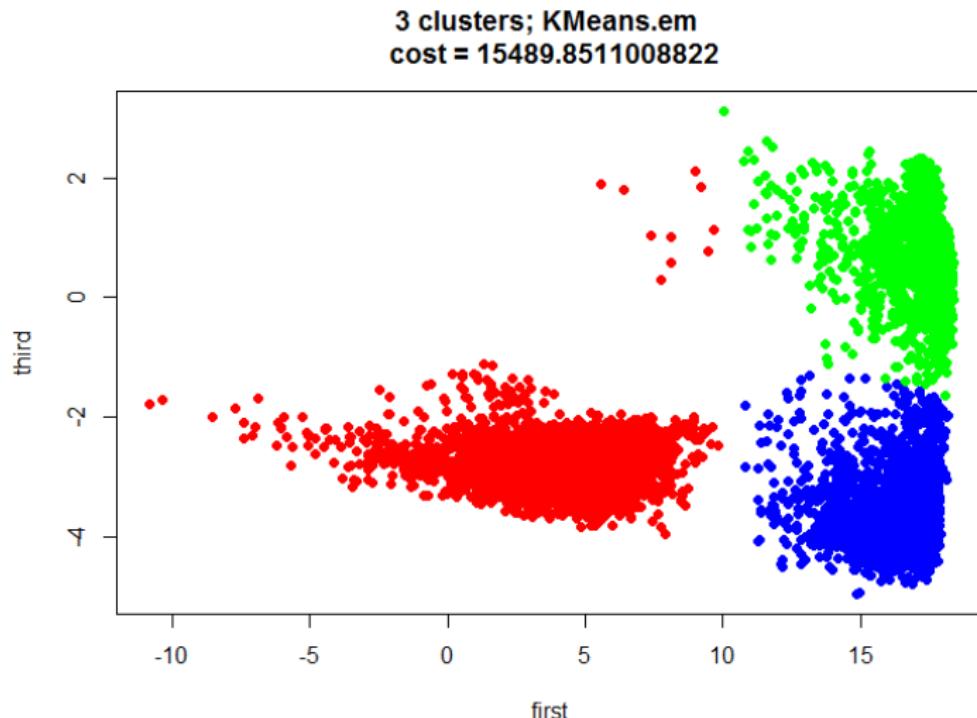


Рис.: Кластеризация EM методом

k-means кластеризация, #7

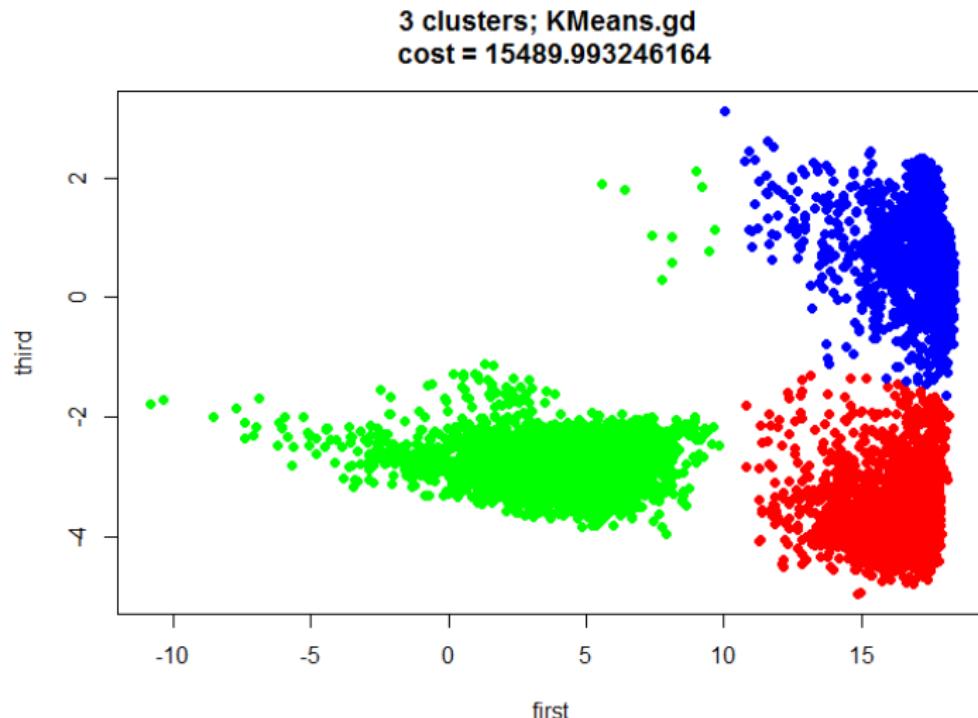


Рис.: Кластеризация методом градиентного спуска

k-means кластеризация, #8

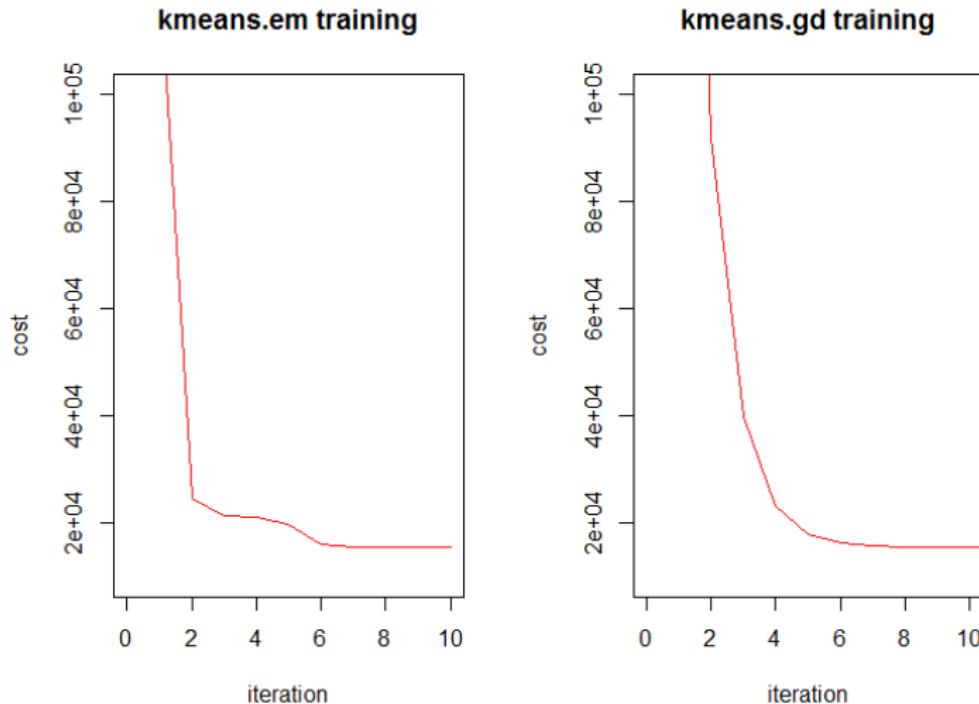


Рис.: Изменение функции стоимости в процессе обучения k-means методом EM и градиентным спуском

Радиально-базисная функция

Радиально-базисная функция (radial-basis function, RBF) - это функция, значение которой зависит только от расстояния до некоторого центра:

- ▶ $\phi(\vec{x}) = \phi(\|\vec{x}\|)$

Примеры ($r = \|\vec{x} - \vec{\mu}\|$):

- ▶ Gaussian: $\phi(\vec{x}) = e^{-(\epsilon \cdot r)^2}$ (часто нормализируют при увеличении размерности входных данных)
- ▶ Multiquadric: $\phi(\vec{x}) = \sqrt{1 + (\epsilon \cdot r)^2}$
- ▶ Polyharmonic spline: $\phi(\vec{x}) = r^k \cdot \ln r$

где $\|\cdot\|$ обычно:

- ▶ Евклидово
- ▶ Махаланобиса

Радиально-базисный нейрон

► $r = \vec{x} - \vec{w}$

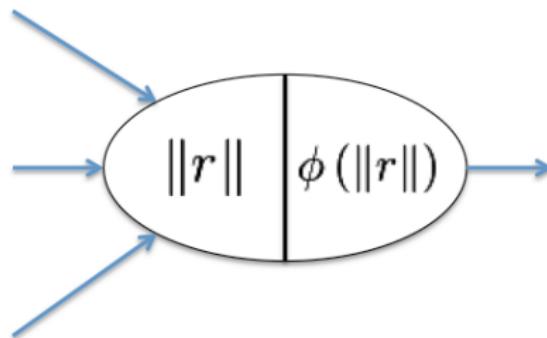


Рис.: RBF-neuron

Радиально-базисная нейросеть, #1

- ▶ кстати а как сделать нейросеть для регрессии? точнее какой нейрон находится на выходе?

Радиально-базисная нейросеть, #2

- ▶ $f(x) = \sum_{i=1}^m w_i \phi(\|\vec{x} - \vec{c}_i\|)$
- ▶ а как можно использовать не размеченные образы?

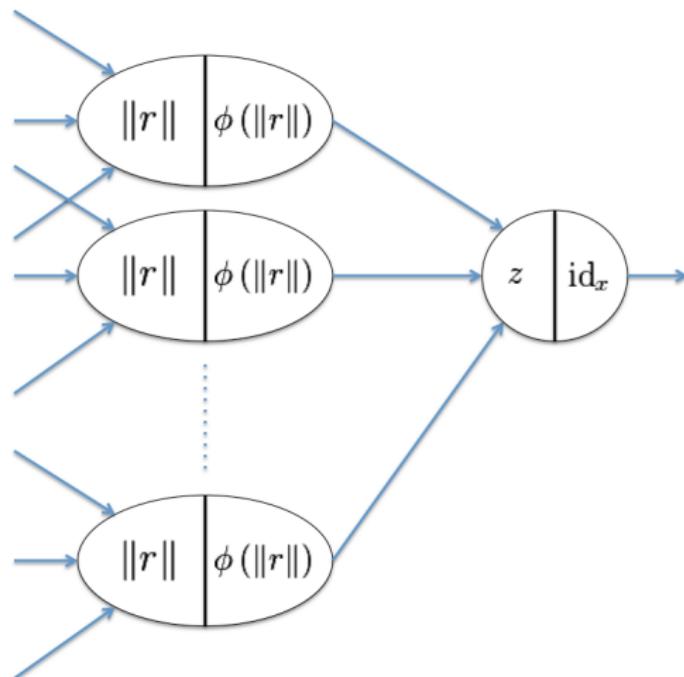


Рис.: RBF-network

Обучение RBFN

Как задумывали авторы¹⁰ в 1988 году:

- ▶ unsupervised pretraining with k-means
- ▶ solve linear regression problem

По современному¹¹:

- ▶ unsupervised pretraining with some clustering algorithm
- ▶ fine-tuning with backprop

¹⁰Broomhead, D. S.; Lowe, David. Radial basis functions, multi-variable functional interpolation and adaptive networks

¹¹Schwenker, Friedhelm; Kestler, Hans A.; Palm, Günther (2001). Three learning phases for radial-basis-function networks

RBF-сеть vs MLP

Оба типа сетей являются универсальными аппроксиматорами!

- ▶ Теорема Ковера (1965): вероятность линейной разделимости возрастает при нелинейном преобразовании оригинальных данных в пространство более высокой размерности
- ▶ Теорема Micchelli (1986): $\{\vec{x}_i\}_{i=1}^N$ - множество различных точек в $\mathbb{R}^n \Rightarrow \Phi = \{\phi_{ij}\}_{i=1..N, j=1..N}$ - не сингулярна, где $\phi_{ij} = \phi(\|\vec{x}_i - \vec{x}_j\|)$
- ▶ не стоит забывать про проклятие размерности: пусть N точек равномерно распределены в единичном шаре размерности n , тогда среднее расстояние от нуля до точек равно

$$d(n, N) = \left(1 - \frac{1}{2}^{\frac{1}{N}}\right)^{\frac{1}{n}}$$

- ▶ $n = 10, N = 500 : d \approx 0.52$
- ▶ $n \rightarrow \infty, N \rightarrow \infty : d \rightarrow 1$

- ▶ есть смысл из пространства малой размерности переходить в большее с использованием RBF

k-means кластеризация, #9

- ▶ как сделать k -means с использованием нейросети?

Победитель получает все

Победитель получает все (или к победителям) - вычислительный шаблон (разновидность competitive learning), мотивированный нейронаукой, используются в вычислительных моделях мозга, особенно для распределённого принятия решения в коре.

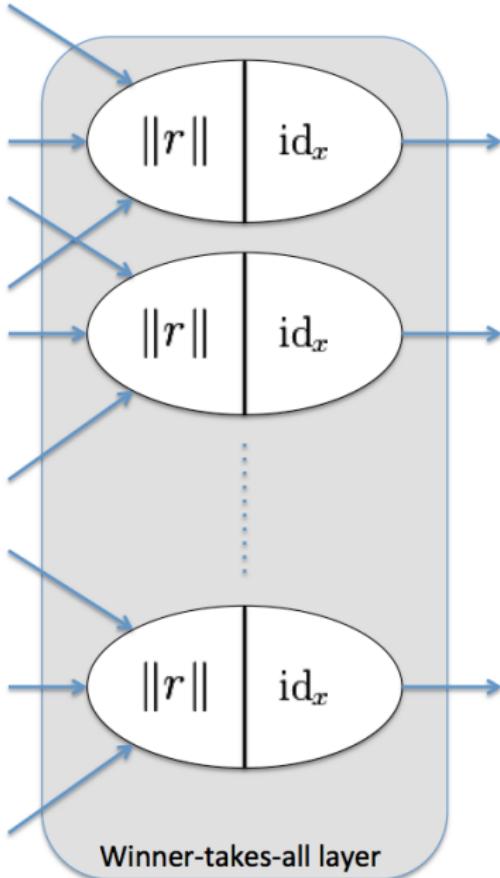
- ▶ иерархические модели зрения (Riesenhuber и др. 1999);
- ▶ модели отборного внимания и признания (Carpenter and Grossberg, 1987; Itti et al. 1998);

В сфере ИНС:

- ▶ формально доказано, что операция "победитель получает всё" в вычислительном отношении сильнее по сравнению с другими нелинейными операциями, такими как пороговая обработка¹²;
"In addition we show that arbitrary continuous functions can be approximated by circuits employing a single soft winner-take-all gate as their only nonlinear operation";
- ▶ нейронная сеть Кохонена (1972);
- ▶ расширяющийся нейронный газ (A Growing Neural Gas Network Learns Topologies, B. Fritzke, 1995).

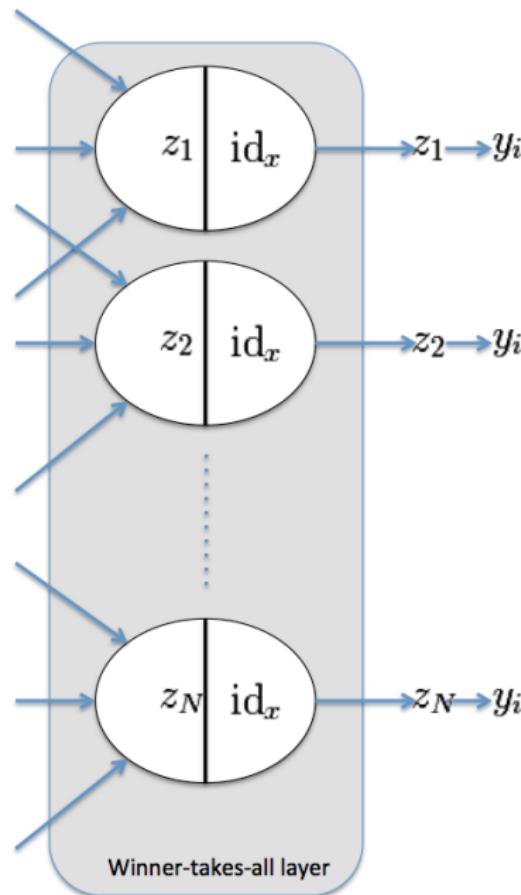
¹²Maass 2000, <http://www.igi.tugraz.at/maass/psfiles/113.pdf>

k-means кластеризация, #11



- ▶ кластеризующий слой
- ▶ $E = \sum_{k=1}^{|C|} \sum_{i=1}^{|X_k|} d^2 (\vec{x}_i, \vec{c}_k) = \sum_{i=1}^{|X|} \min_{j=1}^{|C|} d^2 (\vec{x}_i, \vec{c}_j)$
- ▶ $\frac{\partial \min_{j=1}^{|C|} d^2 (\vec{x}_i, \vec{c}_j)}{\partial c_{ab}} = \begin{cases} 0, & \min_{j=1}^{|C|} d^2 (\vec{x}_i, \vec{c}_j) \neq d^2 (\vec{x}_i, \vec{c}_a) \\ \frac{\partial d^2 (\vec{x}_i, \vec{c}_a)}{\partial c_{ab}}, & \min_{j=1}^{|C|} d^2 (\vec{x}_i, \vec{c}_j) = d^2 (\vec{x}_i, \vec{c}_a) \end{cases}$
- ▶ он может быть составной частью более комплексной модели

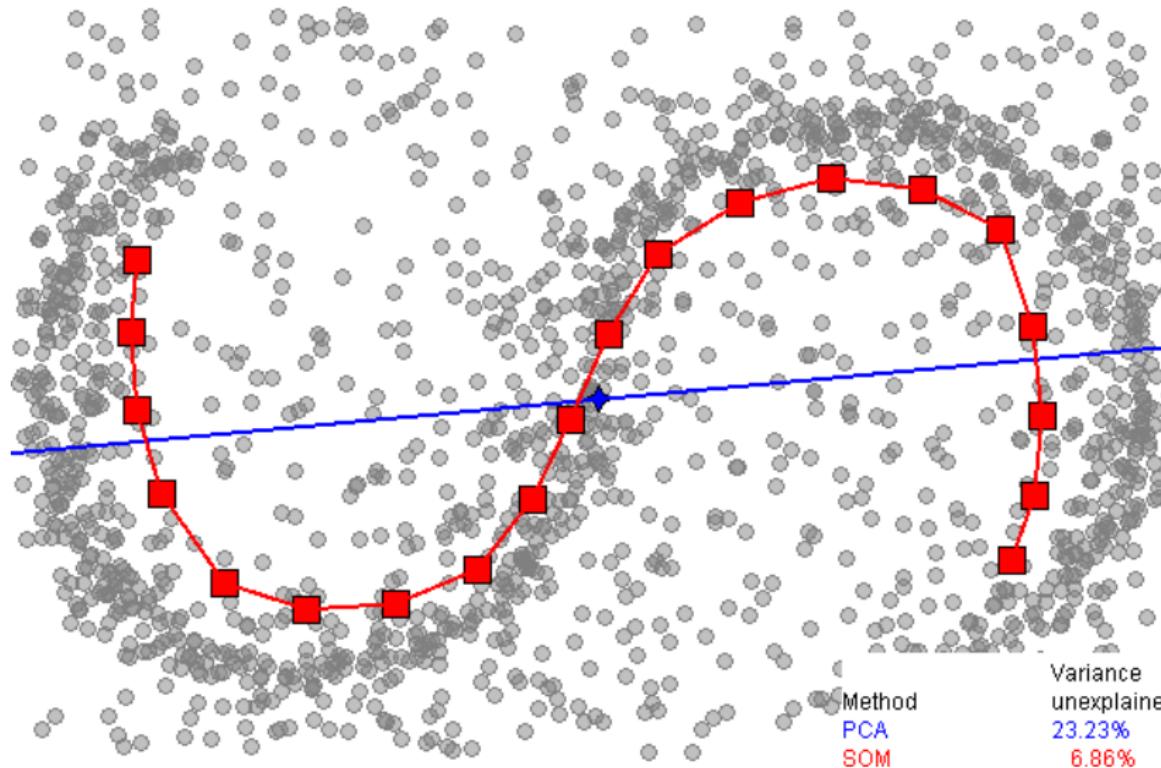
Слой Кохонена



$$\triangleright y_i = \begin{cases} 1, & i = \arg \max \vec{z} \\ 0, & i \neq \arg \max \vec{z} \end{cases}$$

Нелинейное сжатие размерности, #1

Нелинейное сжатие размерности = manifold learning = feature extraction



Нелинейное сжатие размерности, #2

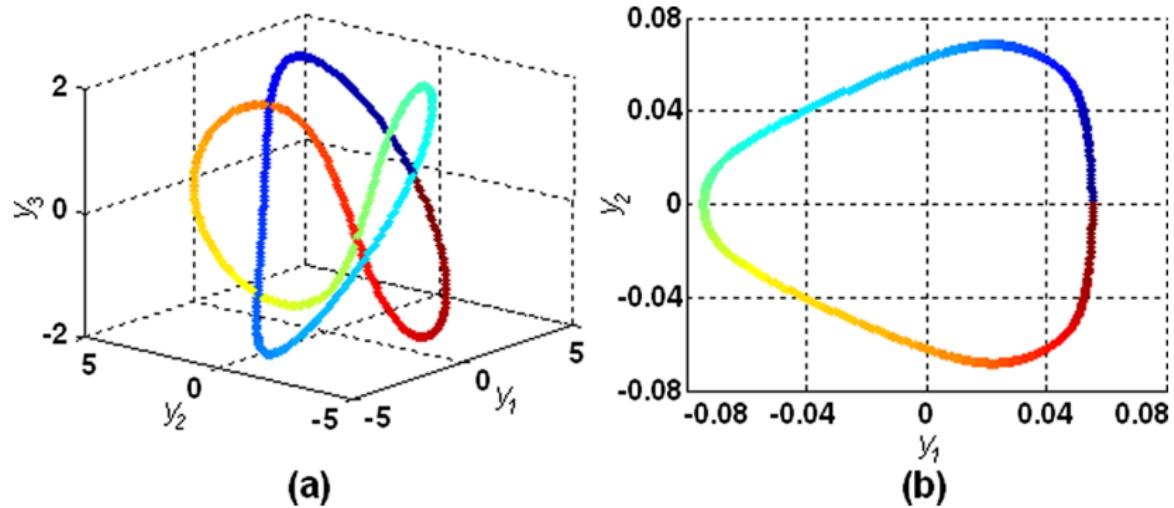


Рис.: Laplacian eigenmap

Самоорганизующиеся карты Кохонена, #1

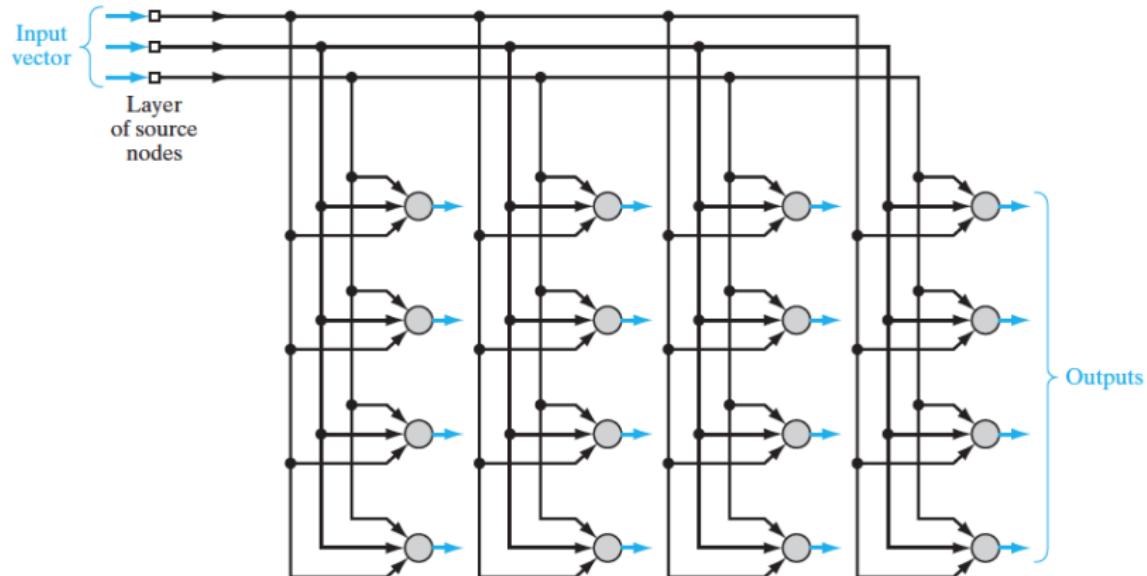
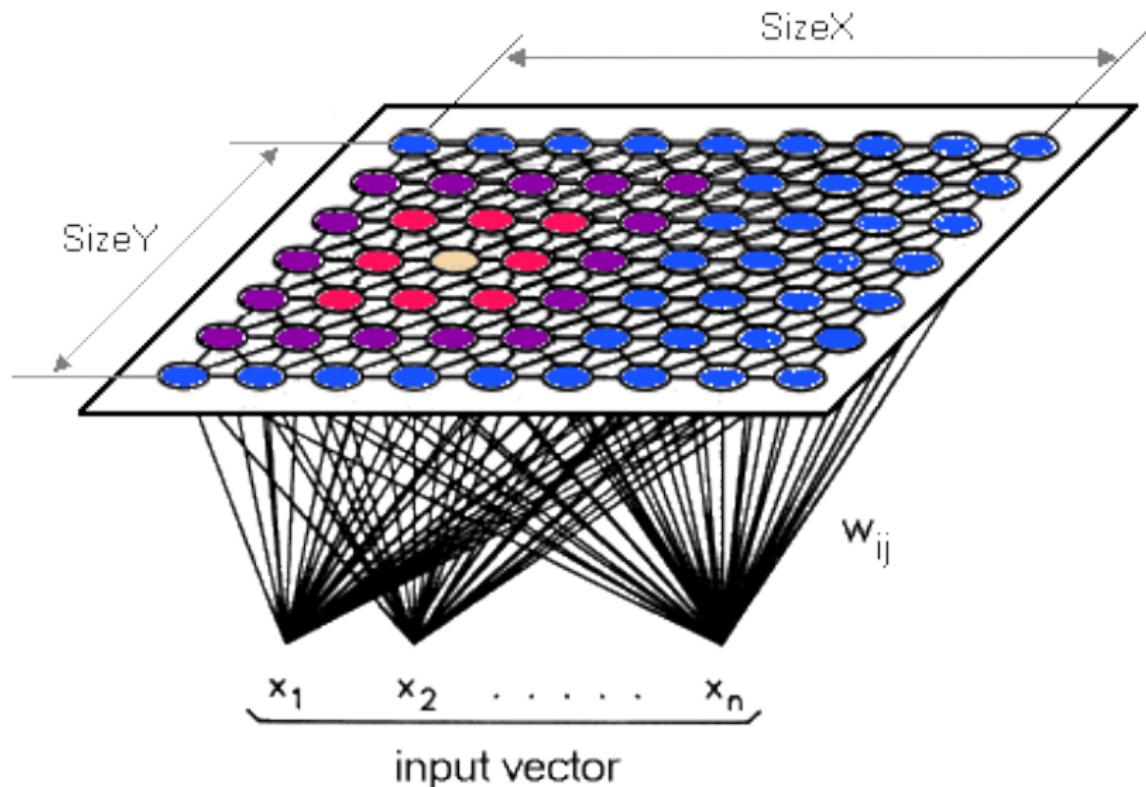


Рис.: Модель SOM сети¹³

¹³Neural Networks and Learning Machines (3rd Edition), Simon O. Haykin

Самоорганизующиеся карты Кохонена, #2



Самоорганизующиеся карты Кохонена, #3

- ▶ априорно задается структура решетки
- ▶ используется метод k-победителей забирают все
- ▶ веса нейронов обновляются последующей формуле
 - ▶ $\Delta \vec{w}_i = \eta(\tau, i_{\text{winner}}, i) \cdot (\vec{x} - \vec{w}_i)$

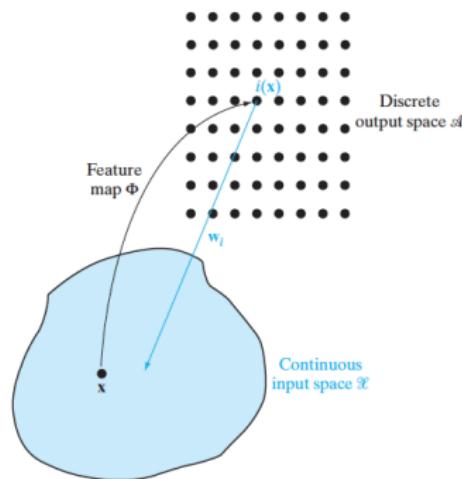


Рис.: SOM как кодировщик¹⁴

¹⁴Neural Networks and Learning Machines (3rd Edition), Simon O. Haykin

Самоорганизующиеся карты Кохонена, #4

- ▶ U-matrix узлов: каждому узлу ставится в соответствие число равное среднему расстоянию до его топологических соседей (заданных решеткой)

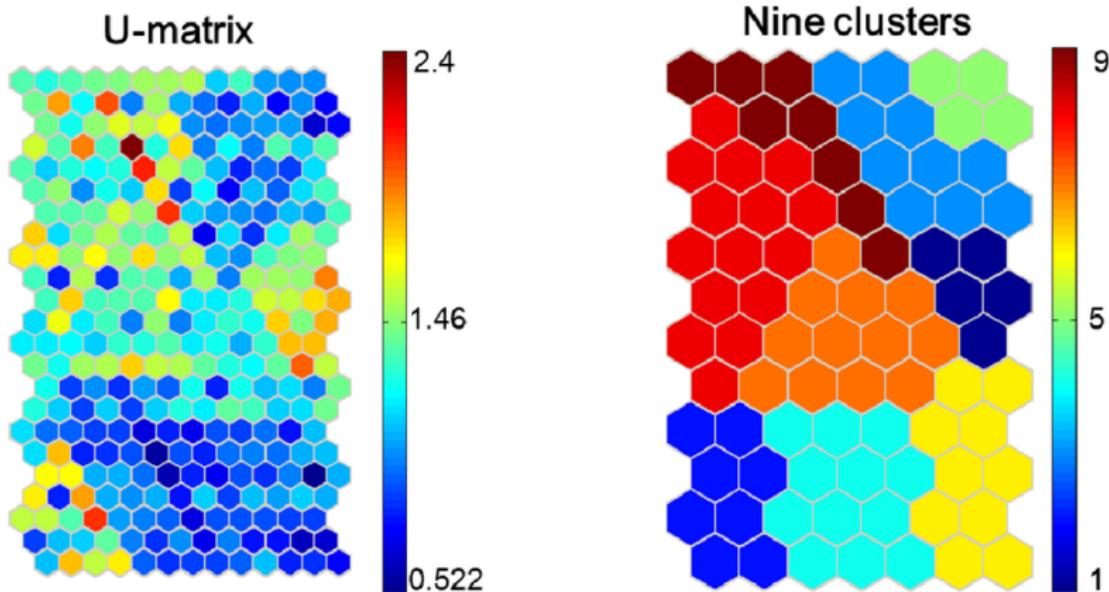


Рис.: U-matrix и k-means кластеризация в новом пространстве¹⁵

¹⁵<http://www.mdpi.com/1660-4601/11/4/3618/htm>

Самоорганизующиеся карты Кохонена, #5

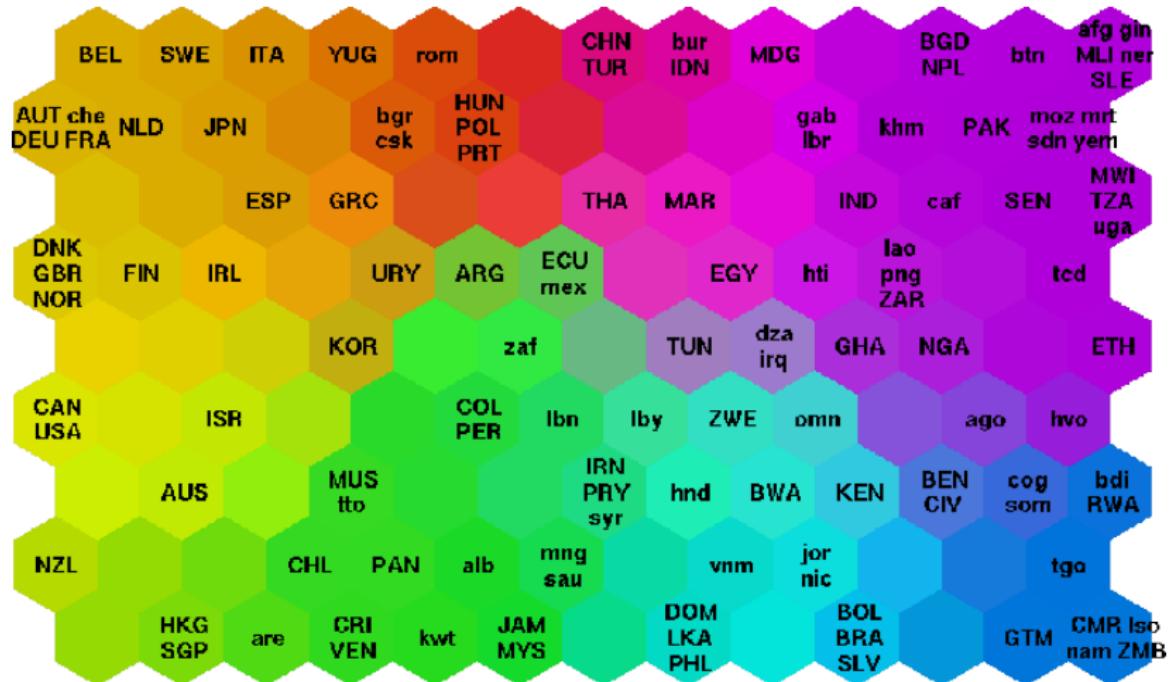


Рис.: Карта построенная на основании 39 показателей стран статистики World Bank¹⁶

¹⁶<http://www.cis.hut.fi/research/som-research/worldmap.html>

Самоорганизующиеся карты Кохонена, #6

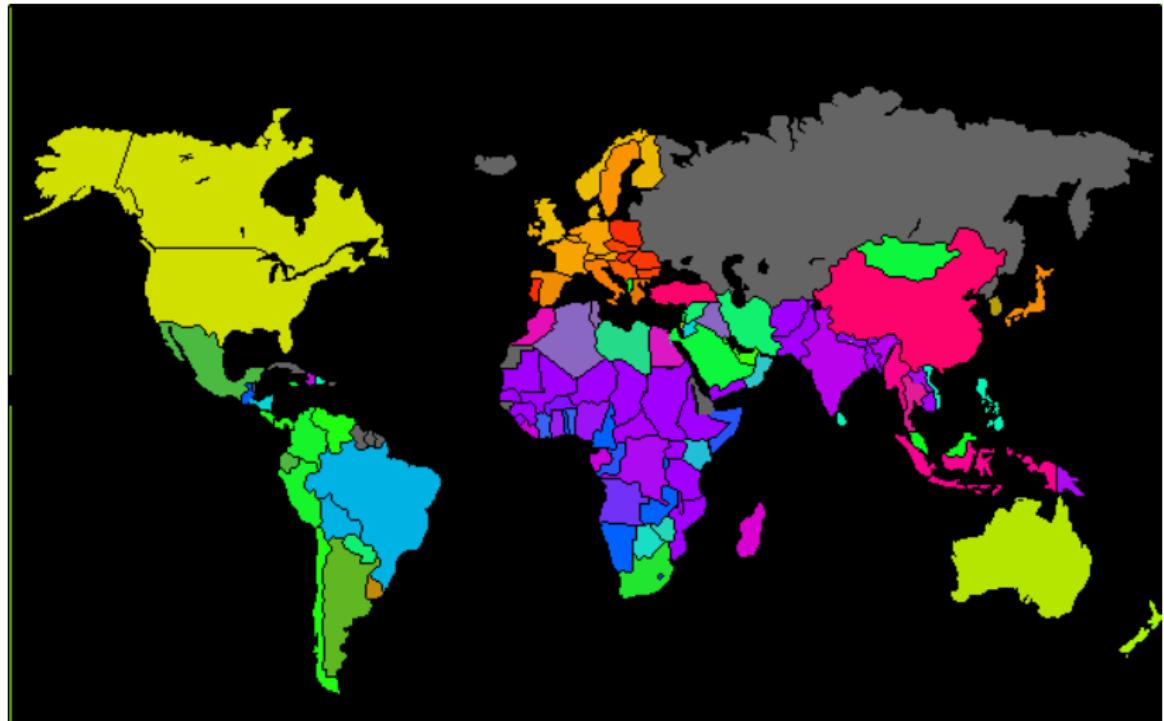


Рис.: Цвета кластеров перенесены на реальную карту¹⁷

¹⁷<http://www.cis.hut.fi/research/som-research/worldmap.html>

Нейропластичность

Нейропластичность - способность изменяться под действием опыта; различают два вида нейропластичности

- ▶ синаптическая ("зрение языком сверточные сети, transfer learning")
- ▶ нейронная/нейрогенез (расширяющийся нейронный газ)

Расширяющийся нейронный газ, #1

- ▶ обладает свойством сохранения топологии, решетка не задается априорно
- ▶ использует Competitive Hebbian Rule (CHR)¹⁸
- ▶ использует принцип нейронного газа, который базируется на сети Кохонена¹⁹

¹⁸ Competitive Hebbian Learning Rule forms perfectly topology preserving maps (T. Martinetz, 1993)

¹⁹ A "neural gas" network learns topologies (T. Martinetz, K. Schulten, 1991)

Competitive Hebbian Rule, #1

- ▶ пусть задан набор центров и набор данных; тогда будем выбирать случайную точку ϵ из данных, и соединять ребром два ближайших к ней центра \Rightarrow получится подмножество триангуляции Делоне в \mathbb{R}^n , расположенное в тех областях \mathbb{R}^n где $P(\epsilon) > 0$

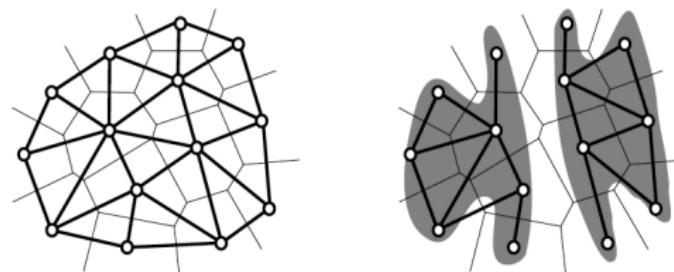


Рис.: Триангуляция Делоне и ее подмножество²⁰

²⁰A Growing Neural Gas Network Learns Topologies (B. Fritzke, 1995)

Competitive Hebbian Rule, #2

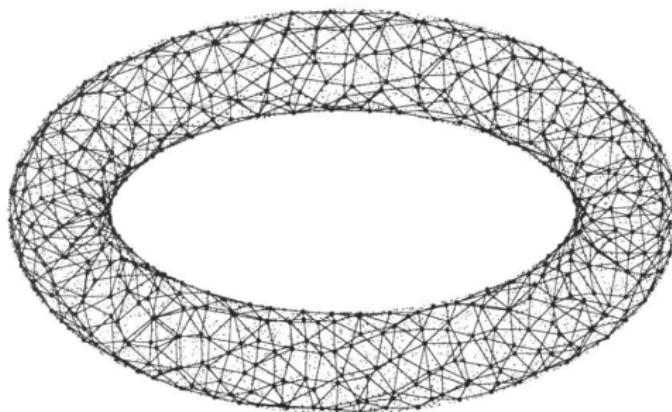


Рис.: Аппроксимация тора²¹

- ▶ а что если изначальное расположение центров выбрано плохо? в тех местах где $P(\epsilon) \approx 0$

²¹Competitive Hebbian Learning Rule forms perfectly topology preserving maps (T. Martinetz, 1993)

Нейронный газ, #1

- ▶ нейронный потому что нейроны в качестве вычислительных юнитов + наследование идеи от сети Кохонена
- ▶ газ - потому что учитываются только парные взаимодействия, а так же сеть заполняет собой подмногообразие в оригинальном многообразии более высокой размерности

Главный принцип похож на сеть Кохонена, но без решетки; или другими словами k-means градиентным спуском, но k-соседей получают все:

- ▶ для каждой точки из набора данных адаптировать k-ближайших центров

Нейронный газ, #2

В итоге алгоритм нейронного газа следующий:

1. сгенерировать случайные центры
2. применить процедуру адаптации центров некоторое количество раз
3. построить топологическую аппроксимацию, используя CHR

А что если в такую модель каким либо способом добавить нейрогенез?

Расширяющийся нейронный газ, #2

0. сгенерировать два случайных центра $\vec{w}_a, \vec{w}_b \in \mathbb{R}^n$
1. семплировать \vec{x} из данных
2. найти два ближайших центра s_1 и s_2
3. увеличить возраст всех ребер исходящих из s_1
4. увеличить локальную ошибку нейрона-победителя:
$$\Delta_{\text{err}}(s_1) = \|\vec{w}_{s_1} - \vec{x}\|^2$$
5. адаптировать нейрон-победитель и всех его топологических соседей:
 - ▶ $w_{s_1} = \eta_b (\vec{x} - \vec{w}_{s_1})$
 - ▶ $w_{s_n} = \eta_n (\vec{x} - \vec{w}_{s_n})$
6. соединить s_1 и s_2 ребром, если его нет; обнулить возраст ребра (CHR)
7. удалить все ребра с возрастом больше a_{\max} ; удалить изолированные узлы
8. нейрогенез если номер итерации $\text{mod } \lambda = 0$
9. уменьшить ошибки всех узлов в d раз
10. остановка или назад к пункту 1

Расширяющийся нейронный газ, #3

Процедура нейрогенеза:

1. найти юнит q с максимальной локальной ошибкой
2. вставить юнит r между q и его топологическим соседом f с максимальной ошибкой: $\vec{w}_r = \frac{1}{2} (\vec{w}_q + \vec{w}_f)$
3. соединить ребрами нейрон r с нейронами q и f ; удалить ребро между q и f
4. уменьшить ошибку q и f в α раз; инициализировать ошибку нейрона r равной ошибке нейрона q

► демонстрация

Развитие идеи нейрорегенеза в ИНС

- ▶ An incremental network for on-line unsupervised classification and topology learning (SOINN)²²
- ▶ An enhanced self-organizing incremental neural network for online unsupervised learning (e-SOINN)²³

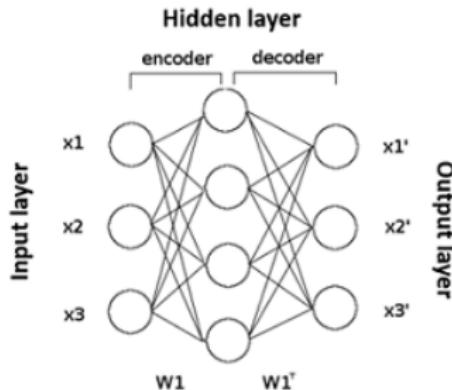
²²<http://www.sciencedirect.com.sci-hub.org/science/article/pii/S0893608005000845>

²³<http://www.sciencedirect.com.sci-hub.org/science/article/pii/S0893608007001190>

Автоенкодер

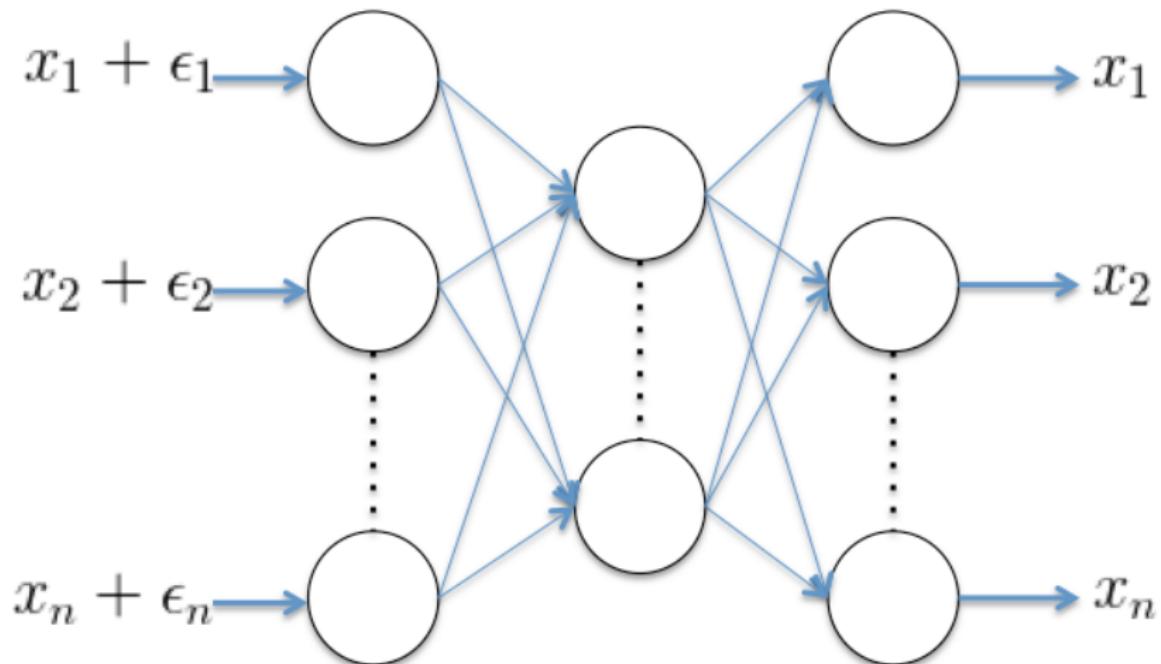
Автоенкодер - многослойный персептрон специальной архитектуры, применяемый для manifold learning (feature extraction)

- ▶ количество входов равно количеству выходов
- ▶ симметрия относительно одного из скрытых слоев (код)



- ▶ накладывая различные новые ограничения на нейросеть можно получить интересные модели, в простейшем случае применяется регуляризация
- ▶ в зависимости от типа данных можно применять различные функции ошибки

Denoising autoencoder, #1



Denoising autoencoder, #2

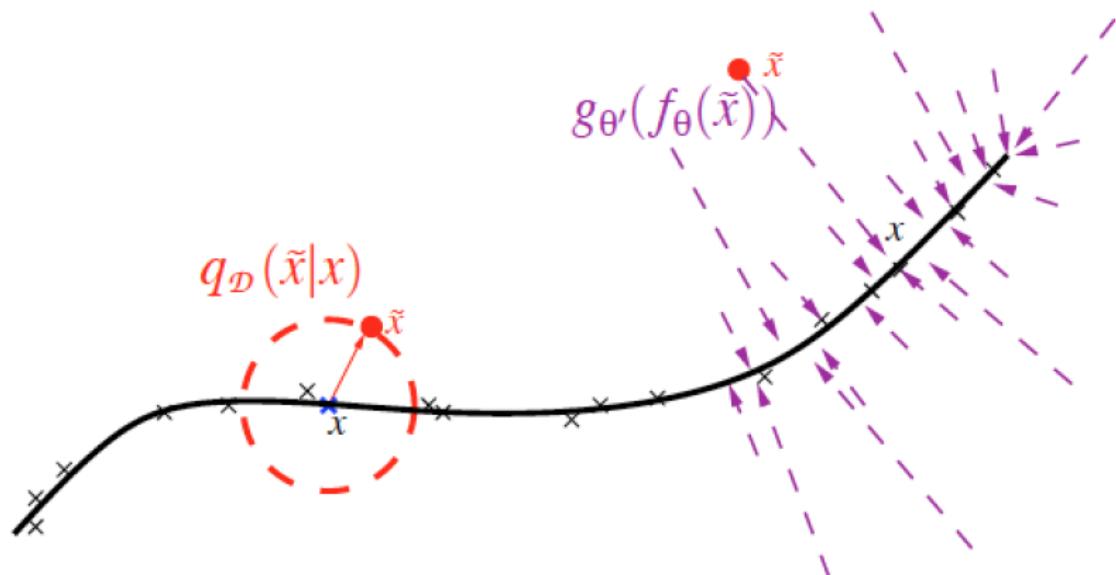


Рис.: Manifold learning²⁴

²⁴Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion, 2010, P. Vincent, Y. Bengio, and others

Denoising autoencoder, #3

Примеры шумов:

- ▶ Гауссовый изотропный шум: $\tilde{x}|x \sim \mathcal{N}(\mu, \sigma^2 \cdot I)$
- ▶ Masked noise: часть элементов обнуляется
- ▶ Salt-and-paper: часть элементов принимают максимальное/минимальное допустимое значение (обычно 0 или 1)

Denoising autoencoder, #4

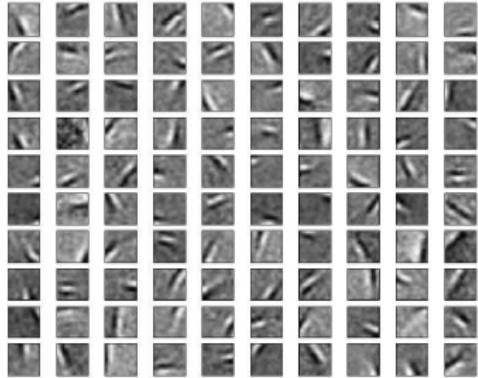
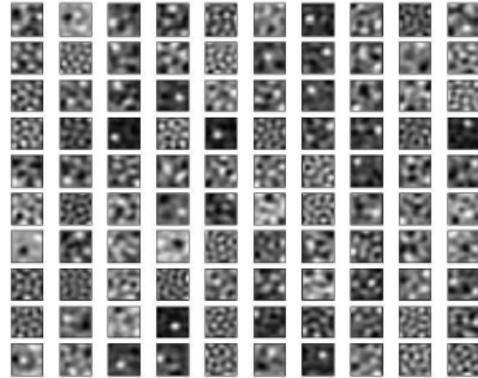


Рис.: Слева автоенкодер с регуляризацией, справа автоенкодер с гауссовым шумом, получились фильтры похожие на фильтры Габора²⁵

²⁵Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion, 2010, P. Vincent, Y. Bengio, and others

Denoising autoencoder, #5

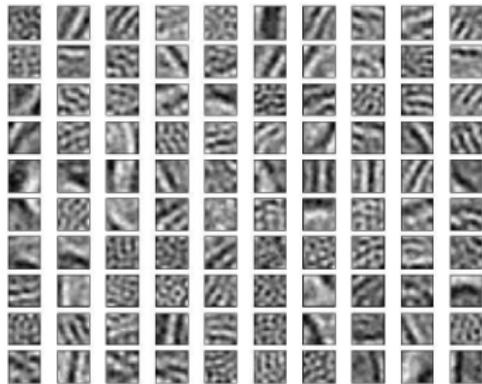
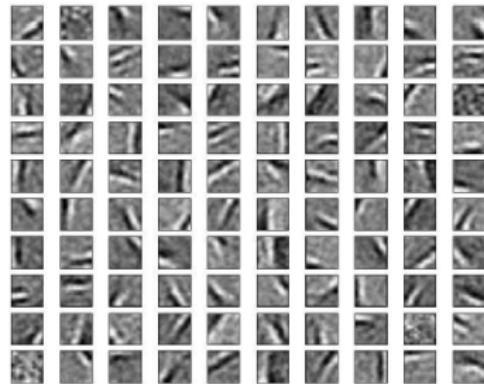
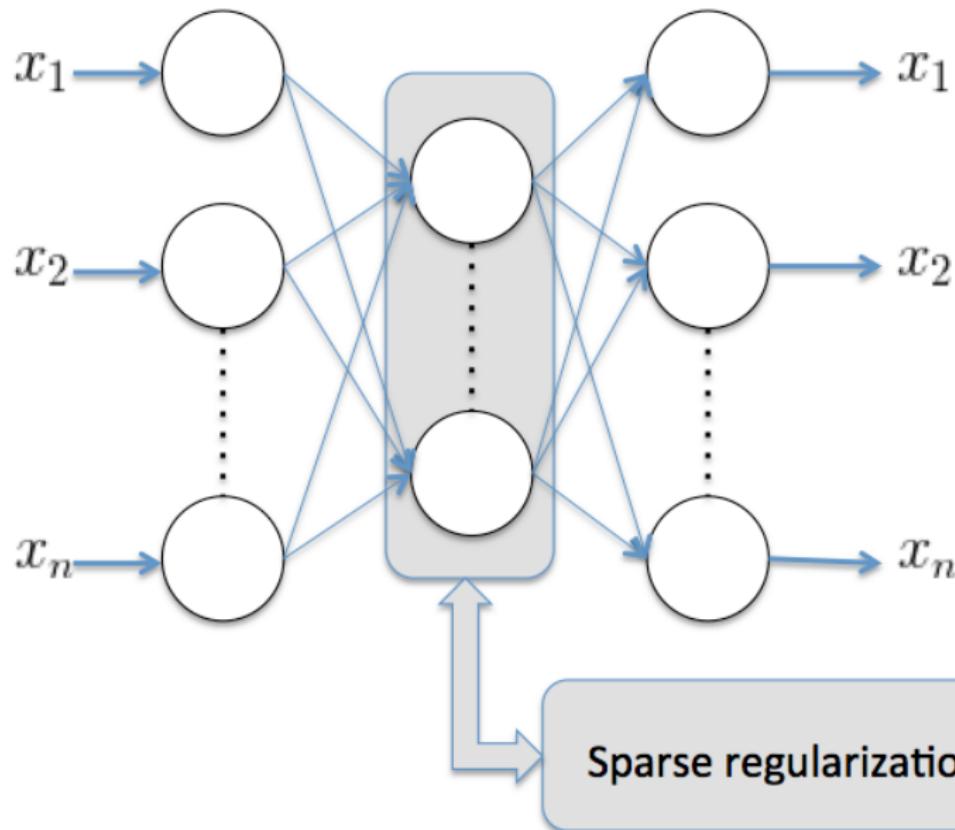


Рис.: Слева автоэнкодер с гауссовым шумом, справа salt-and-paper²⁶

²⁶Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion, 2010, P. Vincent, Y. Bengio, and others

Sparse autoencoder, #1



Sparse autoencoder, #2

- ▶ обозначим значение активации j -ого нейрона скрытого (кодирующего) слоя для элемента данных \vec{x}_i как $a_j(\vec{x}_i)$
- ▶ тогда среднее значение активации j -ого нейрона скрытого слоя:
$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^m a_j(\vec{x}_i)$$
, где m - количество данных в датасете
- ▶ мы хотим уменьшить $\hat{\rho}_j$, сделать его для каждого нейрона примерно равным некоторому априорно заданному числу $\hat{\rho}_j = \rho$ (обычно небольшое число, примерно 0.05)
- ▶ к общей целевой функции минимизации добавим sparsity regularizer:
 - ▶ $R = \sum_{j=1}^h \text{KL}(\rho \parallel \hat{\rho}_j)$, где h - количество нейронов кодирующего слоя
 - ▶ $\text{KL}(\rho \parallel \hat{\rho}) = \rho \log \frac{\rho}{\hat{\rho}} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}}$, это KL расстояние между случайными величинами из распределения Бернулли с математическим ожиданием ρ и $\hat{\rho}_j$

Sparse autoencoder, #3

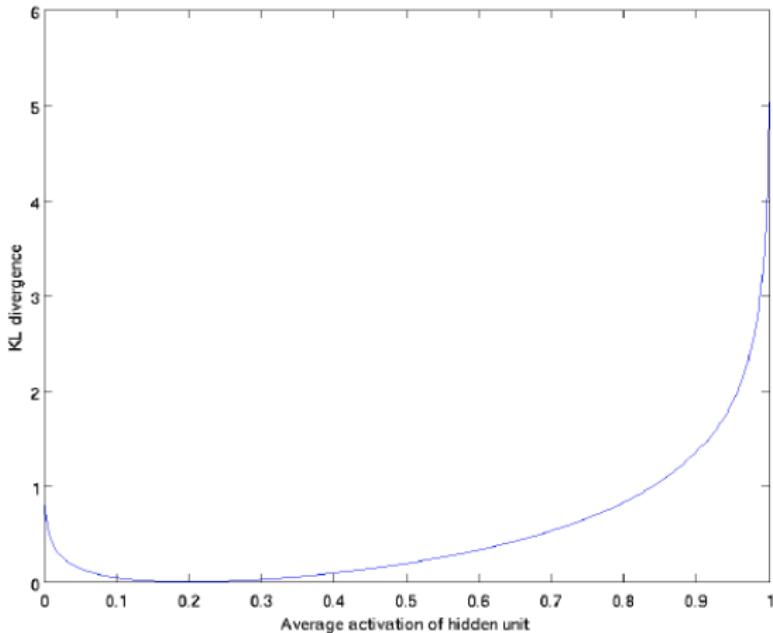
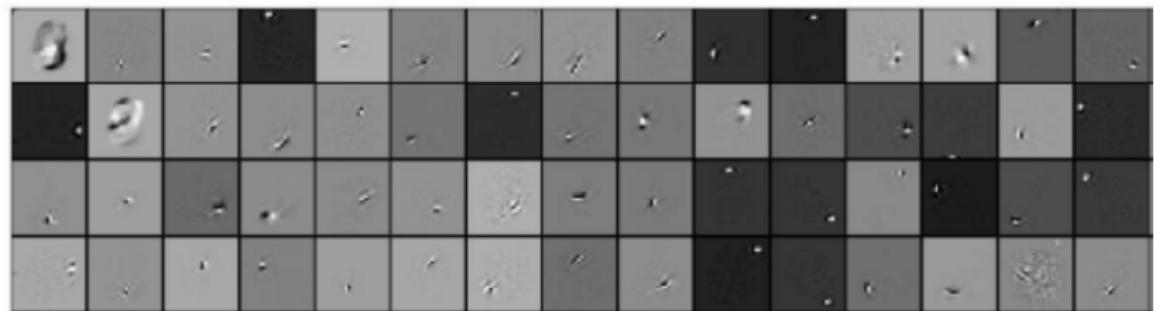
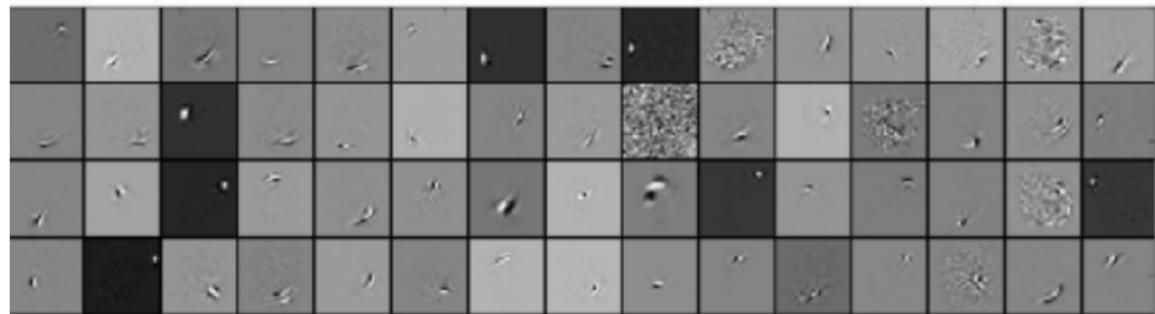


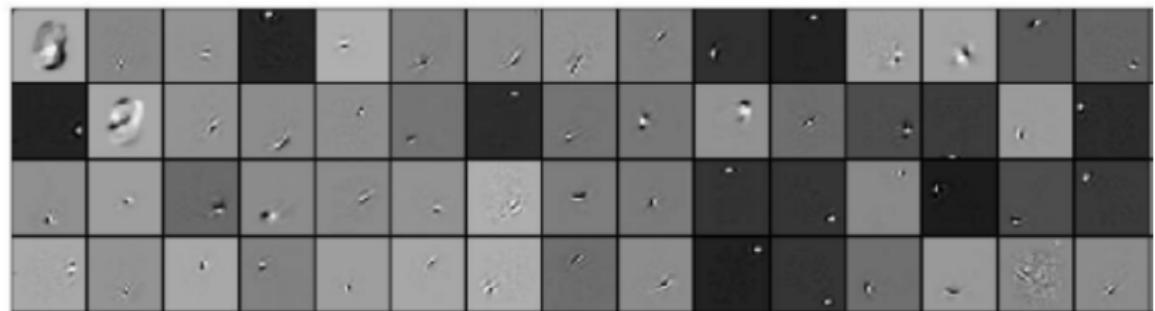
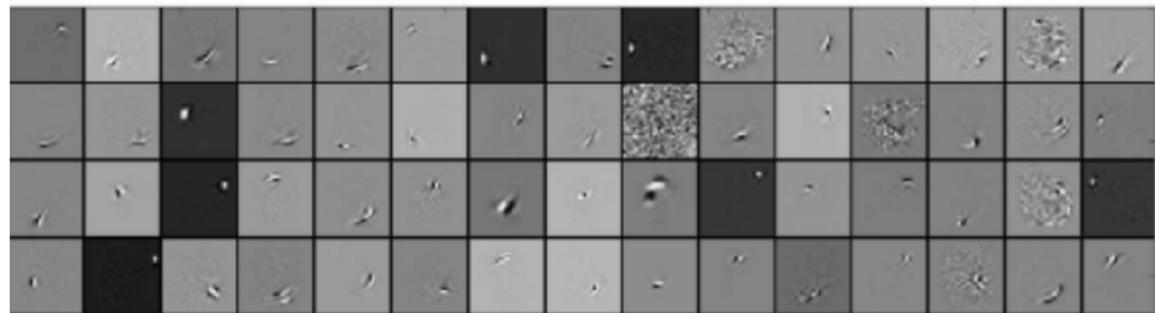
Рис.: KL достигает минимального значение 0 в точке $\hat{\rho}_j = \rho$, а по краям уходит в бесконечность²⁷

²⁷Sparse autoencoder, CS294A Lectur notes, Andrew Ng

Sparse autoencoder, #4



Sparse autoencoder, #4



Auto-Encoding Variational Bayes, #1

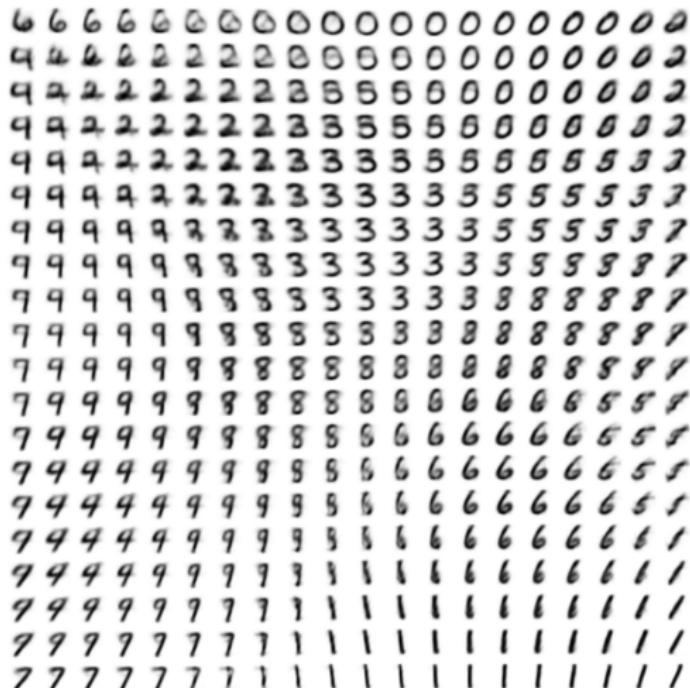


Рис.: MNIST embedding into 2d space²⁸

²⁸<http://arxiv.org/pdf/1312.6114v10.pdf>

Auto-Encoding Variational Bayes, #2



Рис.: FreyFace embedding into 2d space²⁹

²⁹<http://arxiv.org/pdf/1312.6114v10.pdf>

Generative Adversarial Networks

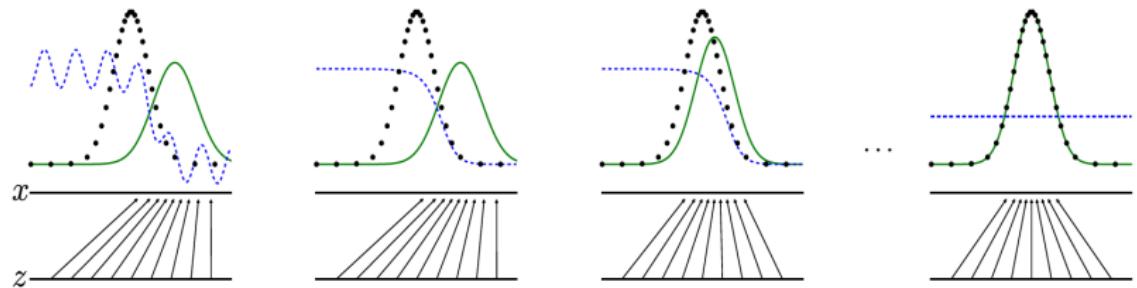


Рис.: Generator vsDiscriminator game³⁰

³⁰<http://arxiv.org/abs/1406.2661>

Deep Convolutional Generative Adversarial Networks, #1

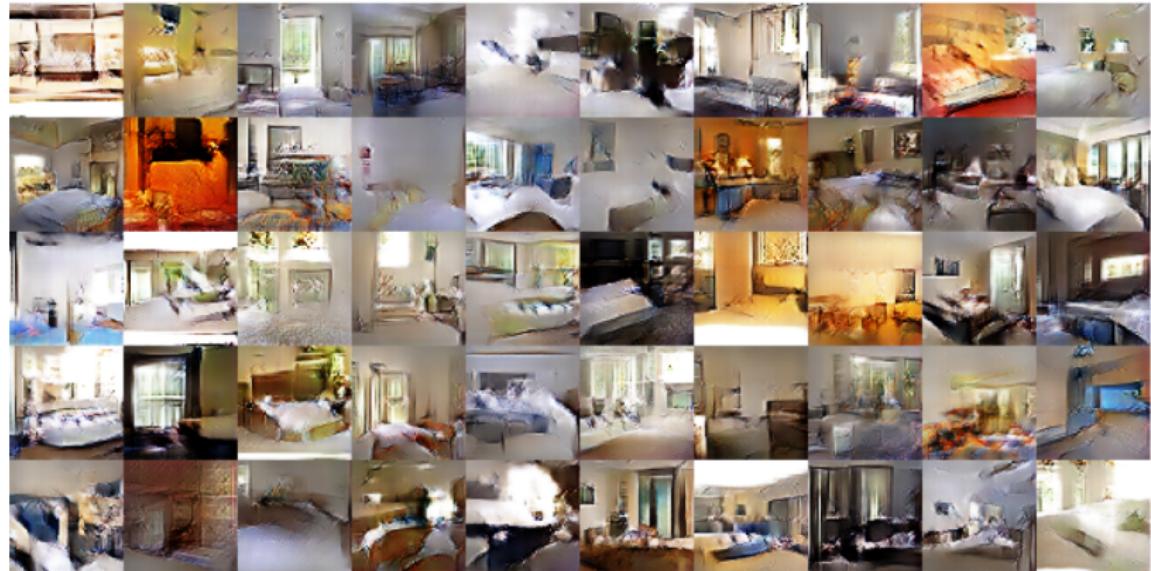


Рис.: Bedrooms after 1 epoch³¹

³¹<http://arxiv.org/pdf/1511.06434v2.pdf>

Deep Convolutional Generative Adversarial Networks, #2



Рис.: Bedrooms after 5 epoch³²

³²<http://arxiv.org/pdf/1511.06434v2.pdf>

Deep Convolutional Generative Adversarial Networks, #3

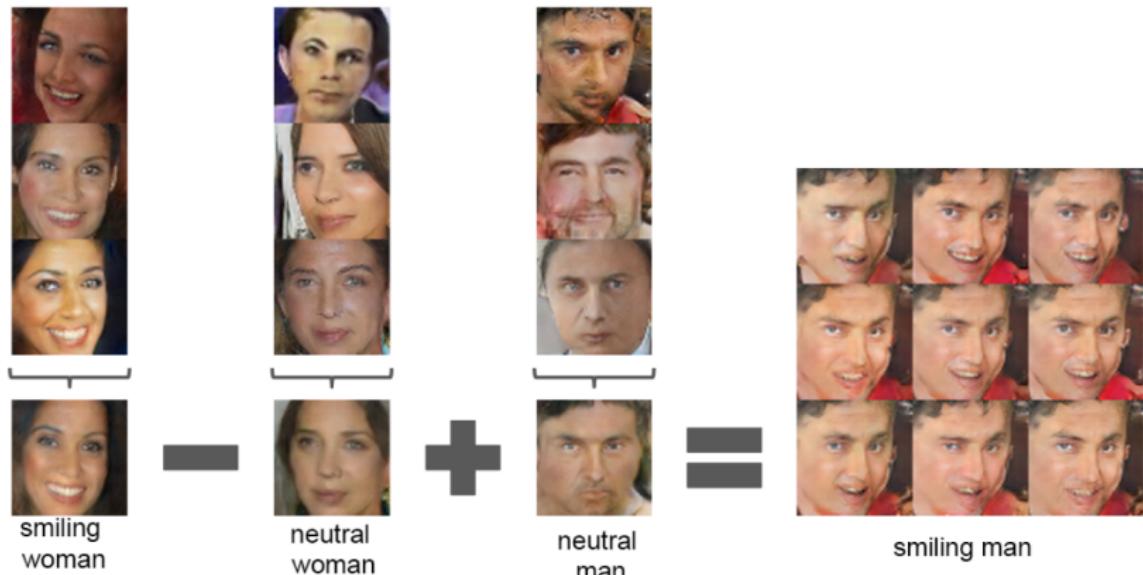


Рис.: Arithmetics in hidden space³³

³³<http://arxiv.org/pdf/1511.06434v2.pdf>

Deep Convolutional Generative Adversarial Networks, #4

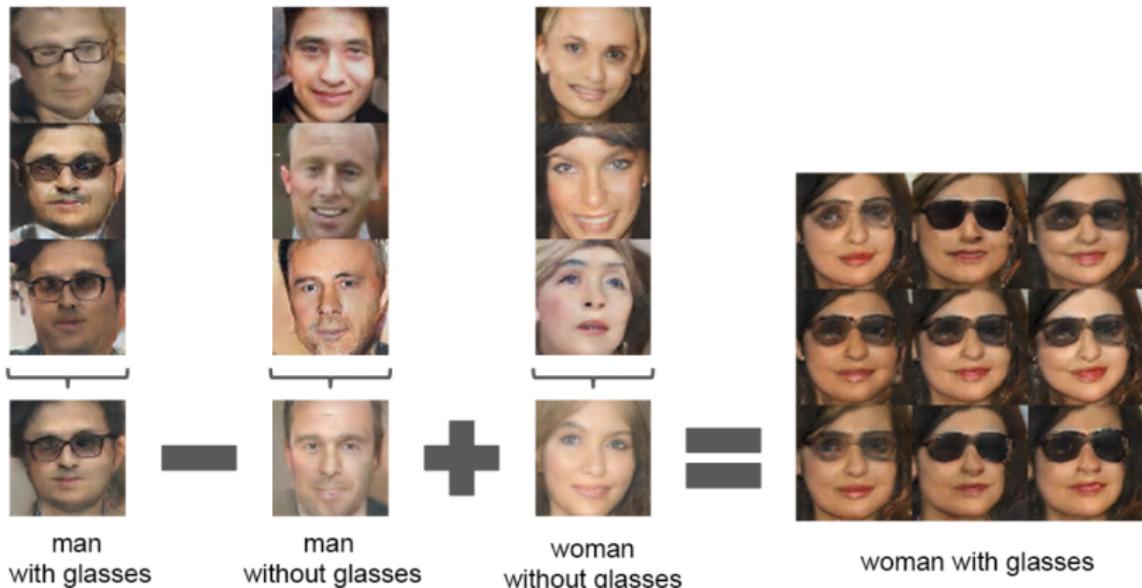


Рис.: Arithmetics in hidden space³⁴

³⁴<http://arxiv.org/pdf/1511.06434v2.pdf>

Deep Convolutional Generative Adversarial Networks, #5

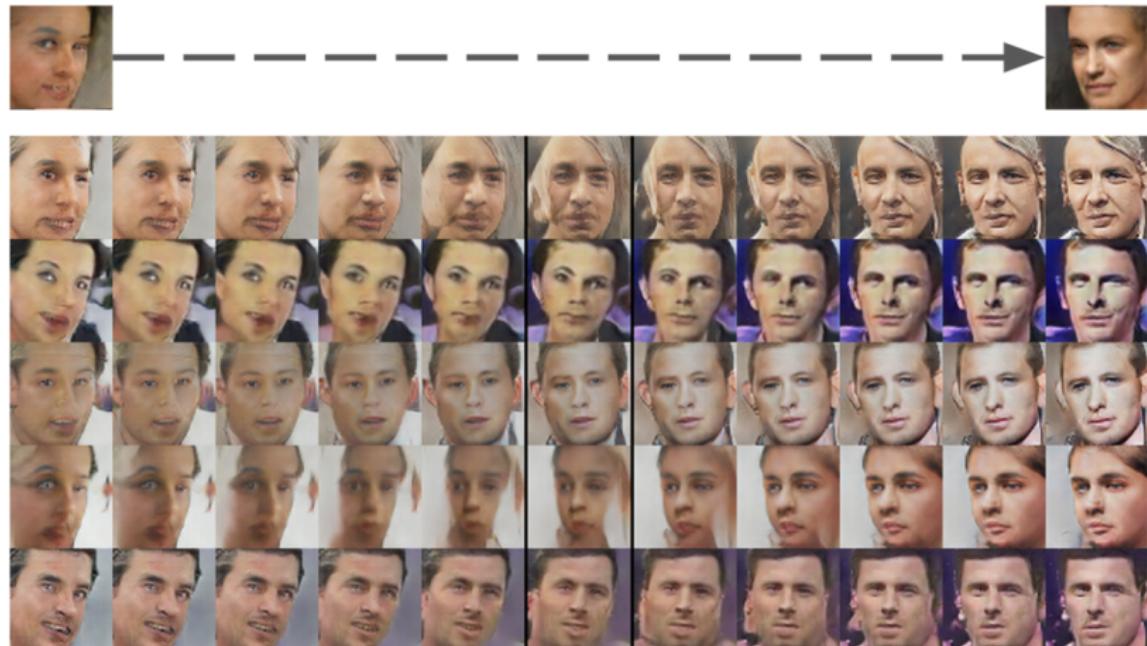


Рис.: Smooth trajectories³⁵

³⁵<http://arxiv.org/pdf/1511.06434v2.pdf>

Вопросы

