

- Прикладные модели машинного обучения •

## Нейронные сети с обучением без учителя

Воронцов Константин Вячеславович

`k.v.vorontsov@phystech.edu`

`http://www.MachineLearning.ru/wiki?title=User:Vokov`

Этот курс доступен на странице вики-ресурса

`http://www.MachineLearning.ru/wiki`

«Машинное обучение (курс лекций, К.В.Воронцов)»

- 1 Сети Кохонена для кластеризации и визуализации**
  - Задача кластеризации
  - Обучаемое векторное квантование
  - Карты Кохонена
- 2 Автокодировщики**
  - Задача понижения размерности
  - Методы регуляризации
  - Автокодировщики более сложной структуры
- 3 Развитие идей частичного обучения**
  - Перенос обучения и многозадачное обучение
  - Дистилляция и привилегированное обучение
  - Генеративные состязательные сети (GAN)

## Постановка задачи кластеризации и квантизации

**Дано:**

$X^\ell = \{x_i\}_{i=1}^\ell$  — обучающая выборка объектов,  $x_i \in \mathbb{R}^n$

$\rho^2(x, w) = \|x - w\|^2$  — евклидова метрика в  $\mathbb{R}^n$

**Найти:**

центры кластеров  $w_y \in \mathbb{R}^n$ ,  $y \in Y$ ; модель кластеризации «правило жёсткой конкуренции» (WTA, Winner Takes All):

$$a(x) = \arg \min_{y \in Y} \rho(x, w_y)$$

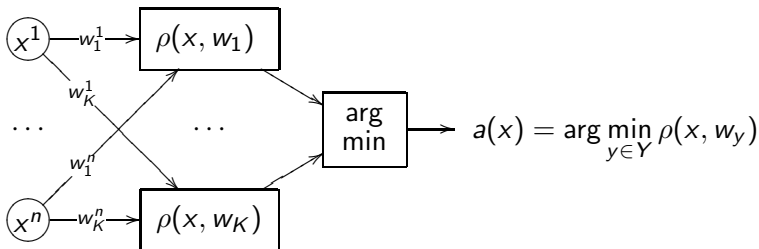
**Критерий:** среднее внутрикластерное расстояние

$$Q(w; X^\ell) = \sum_{i=1}^{\ell} \rho^2(x_i, w_{a(x_i)}) \rightarrow \min_{w_y: y \in Y}$$

*Квантизация данных* — замена  $x_i$  на ближайший центр  $w_{a(x_i)}$

## Сеть Кохонена (сеть с конкурентным обучением)

Структура модели — двухслойная нейронная сеть:



Градиентный шаг в методе SG: для выбранного  $x_i \in X^\ell$

$$w_y := w_y + \eta(x_i - w_y) [a(x_i) = y]$$

Если  $x_i$  относится к кластеру  $y$ , то  $w_y$  сдвигается в сторону  $x_i$

## Алгоритм SG (Stochastic Gradient) для сети Кохонена

**Вход:** выборка  $X^\ell$ ; темп обучения  $\eta$ ; параметр  $\lambda$ ;

**Выход:** центры кластеров  $w_y \in \mathbb{R}^n$ ,  $y \in Y$ ;

инициализировать центры  $w_y$ ,  $y \in Y$ ;

инициализировать текущую оценку функционала:

$$Q := \sum_{i=1}^{\ell} \rho^2(x_i, w_{a(x_i)});$$

**повторять**

выбрать объект  $x_i$  из  $X^\ell$  (например, случайно);

найти ближайший центр:  $y := \arg \min_{y \in Y} \rho(x_i, w_y)$ ;

**градиентный шаг:**  $w_y := w_y + \eta(x_i - w_y)$ ;

оценить значение функционала:

$$Q := (1 - \lambda)Q + \lambda \rho^2(x_i, w_y);$$

**пока** значение  $Q$  и/или веса  $w$  не стабилизируются;

## Жёсткая и мягкая конкуренция

Правило жёсткой конкуренции WTA (winner takes all):

$$w_y := w_y + \eta(x_i - w_y) [a(x_i) = y], \quad y \in Y$$

Недостатки правила WTA:

- медленная скорость сходимости
- некоторые  $w_y$  могут никогда не выбираться

Правило мягкой конкуренции WTM (winner takes most):

$$w_y := w_y + \eta(x_i - w_y) K(\rho(x_i, w_y)), \quad y \in Y$$

где ядро  $K(\rho)$  — неотрицательная невозрастающая функция

Теперь центры всех кластеров смещаются в сторону  $x_i$ ,  
но чем дальше от  $x_i$ , тем меньше величина смещения

## Задача классификации LVQ (Learning Vector Quantization)

**Дано:**

$X^\ell = \{x_i, y_i\}_{i=1}^\ell$  — объекты  $x_i \in \mathbb{R}^n$  с метками классов  $y_i \in Y$   
 $C$  — множество кластеров,  $y(c) \in Y$  — класс кластера  $c \in C$

**Найти:**

центры кластеров  $w_c \in \mathbb{R}^n$ ,  $c \in C$  в модели кластеризации WTA

$$c(x) = \arg \min_{c \in C} \rho(x, w_c)$$

и модель классификации  $a(x) = y(c(x))$

**Критерий:**

min внутрикластерных расстояний в своём классе,

max внутрикластерных расстояний с чужими классами:

$$Q = \sum_{i=1}^{\ell} \rho^2(x_i, w_{c(x_i)}) \left( [y(c(x_i)) = y_i] - [y(c(x_i)) \neq y_i] \right) \rightarrow \min_{w_c: c \in C}$$

## Алгоритм SG (Stochastic Gradient) для LVQ

**Вход:** выборка  $X^\ell$ ; темп обучения  $\eta$ ; параметр  $\lambda$ ;

**Выход:** центры кластеров  $w_c \in \mathbb{R}^n$ ,  $c \in C$ ;

инициализировать центры  $w_c$  и задать  $y(c)$ ,  $c \in C$ ;

инициализировать текущую оценку функционала  $Q$ ;

**повторять**

выбрать объект  $x_i$  из  $X^\ell$  (например, случайно);

вычислить кластеризацию:  $c := \arg \min_{c \in C} \rho(x_i, w_c)$ ;

вычислить классификацию:  $a := y(c)$ ;

$w_c := w_c + \eta(x_i - w_c)$ , если  $a = y_i$ ;

$w_c := w_c - \eta(x_i - w_c)$ , если  $a \neq y_i$ ;

оценить значение функционала:

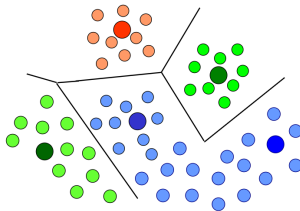
$$Q := (1 - \lambda)Q + \lambda \rho^2(x_i, w_y) \left( [a = y_i] - [a \neq y_i] \right);$$

**пока** значение  $Q$  и/или веса  $w$  не стабилизируются;



## Обучаемое векторное квантование (LVQ)

- кластеризация, реализуемая нейронной сетью
- классификация путём разбиения каждого класса на заданное число кластеров
- позволяет моделировать классы сложной формы
- похоже на отбор эталонов (prototype selection)
- похоже на байесовский классификатор с GMM-классами



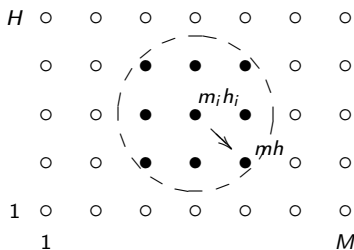
*Teuvo Kohonen. Improved versions of learning vector quantization. 1990*

## Карта Кохонена (Self Organizing Map, SOM)

$Y = \{1, \dots, M\} \times \{1, \dots, H\}$  — прямоугольная сетка кластеров  
 Каждому узлу  $(m, h)$  приписан нейрон Кохонена  $w_{mh} \in \mathbb{R}^n$   
 Наряду с метрикой  $\rho(x_i, x)$  на  $X$  вводится метрика на сетке  $Y$ :

$$r((m_i, h_i), (m, h)) = \sqrt{(m - m_i)^2 + (h - h_i)^2}$$

Окрестность  $(m_i, h_i)$ :



## Обучение карты Кохонена

**Вход:**  $X^\ell$  — обучающая выборка;  $\eta$  — темп обучения;

**Выход:**  $w_{mh} \in \mathbb{R}^n$  — векторы весов,  $m = 1..M$ ,  $h = 1..H$ ;

$w_{mh} := \text{random} \left( -\frac{1}{2MH}, \frac{1}{2MH} \right)$  — инициализация весов;

**повторять**

выбрать объект  $x_i$  из  $X^\ell$  случайным образом;

WTA: вычислить координаты кластера:

$(m_i, h_i) := a(x_i) \equiv \arg \min_{(m,h) \in Y} \rho(x_i, w_{mh});$

**для всех**  $(m, h) \in \text{Окрестность}(m_i, h_i)$

WTM: сделать шаг градиентного спуска:

$w_{mh} := w_{mh} + \eta(x_i - w_{mh}) K(r((m_i, h_i), (m, h)));$

**пока** кластеризация не стабилизируется;

## Интерпретация карт Кохонена

Два типа графиков — цветных карт  $M \times H$ :

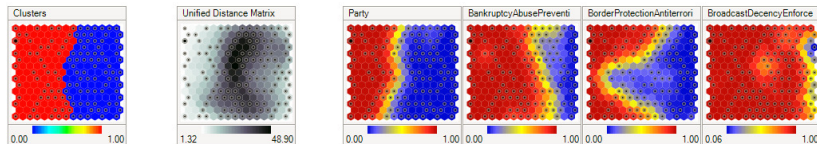
- Цвет узла  $(m, h)$  — локальная плотность в точке  $(m, h)$  — среднее расстояние до  $k$  ближайших точек выборки
- По одной карте на каждый признак:  
цвет узла  $(m, h)$  — значение  $j$ -й компоненты вектора  $w_{m,h}$

**Пример:** задача UCI house-votes (US Congress voting patterns)

Объекты — конгрессмены

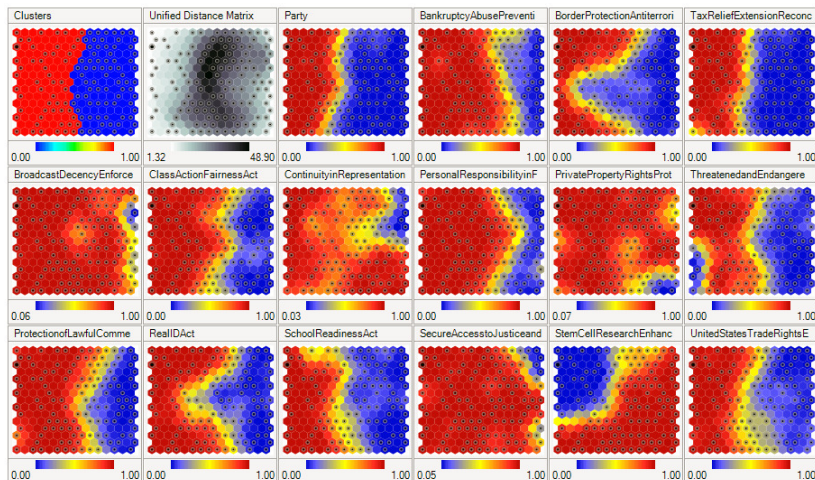
Признаки — результаты голосования по различным вопросам

Есть целевой признак «партия»  $\in \{\text{демократ, республиканец}\}$



## Интерпретация карт Кохонена (продолжение примера)

Пример: задача UCI house-votes (US Congress voting patterns)



## Достоинства и недостатки карт Кохонена

### Достоинства:

- Возможность визуального анализа многомерных данных
- Квантование выборки по кластерам, с автоматическим определением числа непустых кластеров

### Недостатки:

- **Субъективность.** Карта отражает не только кластерную структуру данных, но также зависит от...
  - свойств сглаживающего ядра;
  - (случайной) инициализации;
  - (случайного) выбора  $x_i$  в ходе итераций.
- **Искажения.** Близкие объекты исходного пространства могут переходить в далёкие точки на карте, и наоборот.

Рекомендуется только для разведочного анализа данных.

## Построение автокодировщика — задача обучения без учителя

$X^\ell = \{x_1, \dots, x_\ell\}$  — обучающая выборка

$f: X \rightarrow Z$  — кодировщик (encoder), кодовый вектор  $z = f(x, \alpha)$

$g: Z \rightarrow X$  — декодировщик (decoder), реконструкция  $\hat{x} = g(z, \beta)$

Суперпозиция  $\hat{x} = g(f(x))$  должна восстанавливать исходные  $x_i$ :

$$\mathcal{L}_{AE}(\alpha, \beta) = \sum_{i=1}^{\ell} \mathcal{L}(g(f(x_i, \alpha), \beta), x_i) \rightarrow \min_{\alpha, \beta}$$

Квадратичная функция потерь:  $\mathcal{L}(\hat{x}, x) = \|\hat{x} - x\|^2$

**Пример 1.** Линейный автокодировщик:  $x \in \mathbb{R}^n$ ,  $z \in \mathbb{R}^m$

$$f(x, A) = \underset{m \times n}{A} x, \quad g(z, B) = \underset{n \times m}{B} z$$

**Пример 2.** Двухслойная сеть с функциями активации  $\sigma_f, \sigma_g$ :

$$f(x, A) = \sigma_f(Ax + a), \quad g(z, B) = \sigma_g(Bz + b)$$

## Способы использования автокодировщиков

- Генерация признаков (feature generation)
- Снижение размерности (dimensionality reduction)
- Сжатие данных с минимальными потерями точности
- Более эффективное решение задач обучения с учителем в новом признаковом пространстве
- Обучаемая векторизация объектов, встраиваемая в более глубокие нейросетевые архитектуры
- Послойное предобучение многослойных сетей
- Генерация синтетических объектов, похожих на реальные

---

*Rumelhart, Hinton, Williams.* Learning internal representations by error propagation. 1986.

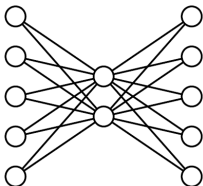
*David Charle et al.* A practical tutorial on autoencoders for nonlinear feature fusion: taxonomy, models, software and guidelines. 2018.



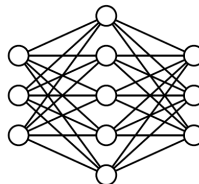
## Архитектуры автокодировщиков

### однослойный кодировщик/декодировщик

снижение размерности

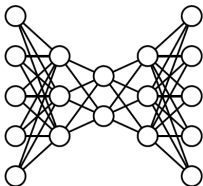


повышение размерности

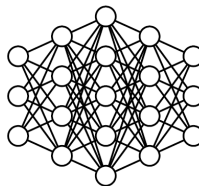


### многослойный кодировщик/декодировщик

снижение размерности



повышение размерности



## Линейный автокодировщик и метод главных компонент

Линейный автокодировщик:  $f(x, A) = Ax$ ,  $g(z, B) = Bz$ ,

$$\mathcal{L}_{AE}(A, B) = \sum_{i=1}^{\ell} \|BAx_i - x_i\|^2 \rightarrow \min_{A, B}$$

Метод главных компонент:  $f(x, U) = U^T x$ ,  $g(z, U) = Uz$ ,  
в матричных обозначениях  $F = (x_1 \dots x_\ell)^T$ ,  $U^T U = I_m$ ,  $G = FU$ ,

$$\|F - GU^T\|^2 = \sum_{i=1}^{\ell} \|UU^T x_i - x_i\|^2 \rightarrow \min_U$$

**Автокодировщик обобщает метод главных компонент:**

- не обязательно  $B = A^T$  (хотя часто именно так и делают)
- произвольные  $A, B$  вместо ортогональных
- нелинейные модели  $f(x, \alpha)$ ,  $g(z, \beta)$  вместо  $Ax, Bz$
- произвольная функция потерь  $\mathcal{L}$  вместо квадратичной
- SG оптимизация вместо сингулярного разложения SVD

## Разреживающие автокодировщики (Sparse AE)

Применение  $L_1$  или  $L_2$ -регуляризации к векторам весов  $\alpha, \beta$ :

$$\mathcal{L}_{\text{AE}}(\alpha, \beta) + \lambda \|\alpha\| + \lambda \|\beta\| \rightarrow \min_{\alpha, \beta}$$

Применение  $L_1$ -регуляризации к кодовым векторам  $z_j$ :

$$\mathcal{L}_{\text{AE}}(\alpha, \beta) + \lambda \sum_{i=1}^{\ell} \sum_{j=1}^m |f_j(x_i, \alpha)| \rightarrow \min_{\alpha, \beta}$$

Энтропийная регуляризация для случая  $f_j \in [0, 1]$ :

$$\mathcal{L}_{\text{AE}}(\alpha, \beta) + \lambda \sum_{j=1}^m \text{KL}(\varepsilon \| \bar{f}_j) \rightarrow \min_{\alpha, \beta},$$

где  $\bar{f}_j = \frac{1}{\ell} \sum_{i=1}^{\ell} f_j(x_i, \alpha)$ ;  $\varepsilon \in (0, 1)$  — близкий к нулю параметр,

$\text{KL}(\varepsilon \| \rho) = \varepsilon \log \frac{\varepsilon}{\rho} + (1 - \varepsilon) \log \frac{1 - \varepsilon}{1 - \rho}$  — KL-дивергенция.

---

*D. Arpit et al. Why regularized auto-encoders learn sparse representation? 2015.*

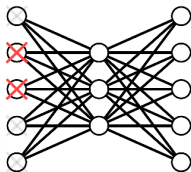
## Шумоподавляющий автокодировщик (Denoising AE)

Устойчивость кодовых векторов  $z_i$  относительно шума в  $x_i$ :

$$\mathcal{L}_{\text{DAE}}(\alpha, \beta) = \sum_{i=1}^{\ell} E_{\tilde{x} \sim q(\tilde{x}|x_i)} \mathcal{L}(g(f(\tilde{x}, \alpha), \beta), x_i) \rightarrow \min_{\alpha, \beta}$$

Вместо вычисления  $E_{\tilde{x}}$  в методе SG объекты  $x_i$  сэмпляются и зашумляются по одному:  $\tilde{x} \sim q(\tilde{x}|x_i)$ . Варианты зашумления:

- $\tilde{x} \sim \mathcal{N}(x_i, \sigma^2 I)$  — гауссовский шум
- обнуление компонент вектора  $x_i$  с вероятностью  $p_0$ :



*P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. ICML-2008.*

## Сжимающий автокодировщик (Contractive AE)

Устойчивость кодовых векторов  $z_i$  относительно шума в  $x_i$ :

$$\mathcal{L}_{AE}(\alpha, \beta) + \lambda \sum_{i=1}^{\ell} \|J_f(x_i)\|^2 \rightarrow \min_{\alpha, \beta}$$

где  $\|J_f(x)\|$  —  $L_2$ -норма матрицы Якоби отображения  $f: X \rightarrow Z$ ,

$$\|J_f(x)\|^2 = \sum_{d=1}^n \sum_{j=1}^m \left( \frac{\partial f_j(x, \alpha)}{\partial x_d} \right)^2$$

В случае  $z = f(x, A) = \sigma(Ax + a)$ , где  $\sigma$  — сигмоида,  $A = (\alpha_{jd})$

$$\|J_f(x)\|^2 = \sum_{d=1}^n \sum_{j=1}^m \left( \alpha_{jd} f_j(x, \alpha) (1 - f_j(x, \alpha)) \right)^2$$

---

*Salah Rifai et al.* Contractive auto-encoders: explicit invariance during feature extraction. ICML-2011.

## Реляционный автокодировщик (Relational AE)

Наряду с потерями реконструкции объектов минимизируем потери реконструкции отношений между объектами:

$$\mathcal{L}_{AE}(\alpha, \beta) + \lambda \sum_{i < j} \mathcal{L}(\sigma(\hat{x}_i^T \hat{x}_j), \sigma(x_i^T x_j)) \rightarrow \min_{\alpha, \beta}$$

где  $\hat{x}_i = g(f(x_i, \alpha), \beta)$  — реконструкция объекта  $x_i$ ,  
 $x_i^T x_j$  — скалярное произведение (близость) пары объектов,  
 $\sigma(s) = (s - s_0)_+$  — пороговая функция с параметром  $s_0$   
(если векторы не близки, то неважно, насколько),  
 $\mathcal{L}(\hat{s}, s)$  — функция потерь, например,  $(\hat{s} - s)^2$ .

Эксперимент: улучшается качество классификации изображений с помощью кодовых векторов на задачах MNIST, CIFAR-10

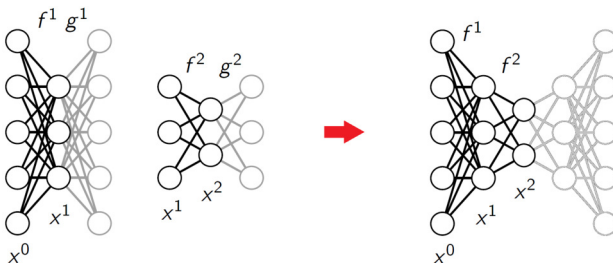
---

Qinxue Meng et al. Relational autoencoder for feature extraction. 2018.

## Многослойный автокодировщик (Stacked AE)

Послойное обучение:  $x^h = f^h(x^{h-1}, \alpha^h)$ ,  $x \equiv x^0$ ,  $z \equiv x^H$

- каждая пара  $f^h, g^h$  обучается по выборке  $\{x_1^{h-1}, \dots, x_\ell^{h-1}\}$
- декодировщик  $g^h$  отбрасывается
- однослойные  $f^1, \dots, f^H$  соединяются в  $H$ -слойный



Тонкая настройка (fine tuning): результат послойного обучения используется как начальное приближение для BackProp

Y. Bengio et al. Greedy layer-wise training of deep networks. NIPS 2007.

## Вариационный автокодировщик (Variational AE)

Строится генеративная модель, способная породить новые объекты  $x$ , похожие на объекты выборки  $X^\ell = \{x_1, \dots, x_\ell\}$

$q_\alpha(z|x)$  — вероятностный кодировщик с параметром  $\alpha$

$p_\beta(\hat{x}|z)$  — вероятностный декодировщик с параметром  $\beta$

Максимизация нижней оценки  $\log$ -правдоподобия:

$$\begin{aligned}\mathcal{L}_{VAE}(\alpha, \beta) &= \sum_{i=1}^{\ell} \log p(x_i) = \sum_{i=1}^{\ell} \log \int q_\alpha(z|x_i) \frac{p_\beta(x_i|z)p(z)}{q_\alpha(z|x_i)} dz \geq \\ &\geq \sum_{i=1}^{\ell} \int q_\alpha(z|x_i) \log \frac{p_\beta(x_i|z)p(z)}{q_\alpha(z|x_i)} dz = \\ &= \sum_{i=1}^{\ell} \int q_\alpha(z|x_i) \log p_\beta(x_i|z) dz - \text{KL}(q_\alpha(z|x_i) \parallel p(z)) \rightarrow \max_{\alpha, \beta}\end{aligned}$$

*D.P.Kingma, M.Welling. Auto-encoding Variational Bayes. 2013.*

*C.Doersch. Tutorial on variational autoencoders. 2016.*



## Вариационный автокодировщик (Variational AE)

Оптимизационная задача для вариационного автокодировщика:

$$\sum_{i=1}^{\ell} \underbrace{E_{z \sim q_{\alpha}(z|x_i)} \log p_{\beta}(x_i|z)}_{\substack{\text{качество реконструкции} \\ \approx \log p_{\beta}(x_i|z), z \sim q_{\alpha}(z|x_i)}} - \underbrace{\text{KL}(q_{\alpha}(z|x_i) \parallel p(z))}_{\text{регуляризатор по } \alpha} \rightarrow \max_{\alpha, \beta}$$

где  $p(z)$  — априорное распределение, обычно  $\mathcal{N}(0, \sigma^2 I)$

Репараметризация  $q_{\alpha}(z|x_i)$ :  $z = f(x_i, \alpha, \varepsilon)$ ,  $\varepsilon \sim \mathcal{N}(0, I)$

Метод стохастического градиента:

- сэмплировать  $x_i \sim X^{\ell}$ ,  $\varepsilon \sim \mathcal{N}(0, I)$ ,  $z = f(x_i, \alpha, \varepsilon)$
- градиентный шаг:  
 $\alpha := \alpha + h \nabla_{\alpha} [\log p_{\beta}(x_i|f(x_i, \alpha, \varepsilon)) - \text{KL}(q_{\alpha}(z|x_i) \parallel p(z))];$   
 $\beta := \beta + h \nabla_{\beta} [\log p_{\beta}(x_i|z)];$

Генерация похожих объектов:  $x \sim p_{\beta}(x|f(x_i, \alpha, \varepsilon))$ ,  $\varepsilon \sim \mathcal{N}(0, I)$

## Автокодировщики для векторизации и обучения с учителем

Данные: размеченные  $(x_i, y_i)_{i=1}^k$ , неразмеченные  $(x_i)_{i=k+1}^{\ell}$

Совместное обучение кодировщика, декодировщика и предсказательной модели (классификации, регрессии или др.):

$$\sum_{i=1}^{\ell} \mathcal{L}(g(f(x_i, \alpha), \beta), x_i) + \lambda \sum_{i=\ell+1}^{\ell+k} \tilde{\mathcal{L}}(\hat{y}(f(x_i, \alpha), \gamma), y_i) \rightarrow \min_{\alpha, \beta, \gamma}$$

$z_i = f(x_i, \alpha)$  — кодировщик

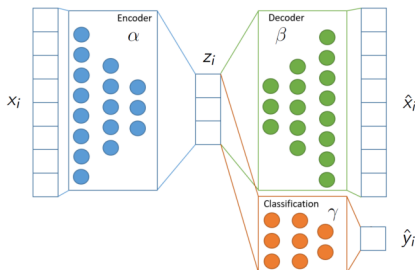
$\hat{x}_i = g(z_i, \beta)$  — декодировщик

$\hat{y}_i = \hat{y}(z_i, \gamma)$  — предиктор

Функции потерь:

$\mathcal{L}(\hat{x}_i, x_i)$  — реконструкция

$\tilde{\mathcal{L}}(\hat{y}_i, y_i)$  — предсказание



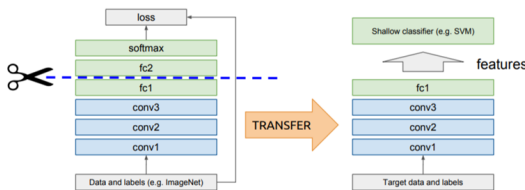
## Предобучение (pre-training), перенос обучения (transfer learning)

Обучение модели векторизации  $z = f(x, \alpha)$  на выборке  $\{x_i\}_{i=1}^{\ell}$ :

$$\sum_{i=1}^{\ell} \mathcal{L}_i(g(f(x_i, \alpha), \beta)) \rightarrow \min_{\alpha, \beta}$$

Обучение целевой модели  $y = g(z, \beta)$  на малых данных:

$$\sum_{i=1}^m \mathcal{L}'_i(g'(f(x'_i, \alpha), \beta')) \rightarrow \min_{\beta'}$$

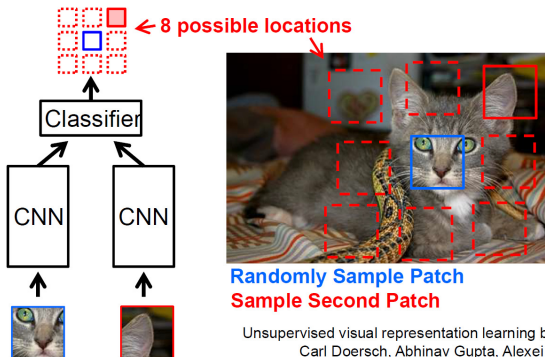


*Sinno Jialin Pan, Qiang Yang. A Survey on Transfer Learning. 2009*

*J. Yosinski et al. How transferable are features in deep neural networks? 2014.*

## Самостоятельное обучение (self-supervised learning)

Модель векторизации  $z = f(x, \alpha)$  обучается предсказывать взаимное расположение пар фрагментов одного изображения



**Преимущество:** сеть выучивает векторные представления объектов без размеченной обучающей выборки.

## Многозадачное обучение (multi-task learning)

$z = f(x, \alpha)$  — векторизация, универсальная для всех моделей  
 $g_t(z, \beta)$  — специфичная часть модели для задачи  $t \in T$

Одновременное обучение модели  $f$  по задачам  $X_t$ ,  $t \in T$ :

$$\sum_{t \in T} \sum_{i \in X_t} \mathcal{L}_{ti}(g_t(f(x_{ti}, \alpha), \beta_t)) \rightarrow \min_{\alpha, \{\beta_t\}}$$

*Обучаемость* (learnability): качество решения отдельной задачи  $\langle X_t, \mathcal{L}_t, g_t \rangle$  улучшается с ростом объёма выборки  $\ell_t = |X_t|$ .

*Learning to learn*: качество решения каждой из задач  $t \in T$  улучшается с ростом как  $\ell_t$ , так и общего числа задач  $|T|$ .

*Few-shot learning*: для решения новой задачи  $t$  достаточно небольшого числа примеров, иногда даже одного.

---

*M. Crawshaw*. Multi-task learning with deep neural networks: a survey. 2020

*Y. Wang et al*. Generalizing from a few examples: a survey on few-shot learning. 2020

## Дистилляция моделей или суррогатное моделирование

Обучение **сложной модели**  $a(x, w)$  «долго, дорого»:

$$\sum_{i=1}^{\ell} \mathcal{L}(a(x_i, w), y_i) \rightarrow \min_w$$

Обучение простой модели  $b(x, w')$ , возможно, на других данных:

$$\sum_{i=1}^k \mathcal{L}(b(x'_i, w'), a(x'_i, w)) \rightarrow \min_{w'}$$

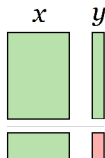
**Примеры задач:**

- замена сложной модели (климат, аэродинамика и др.), которая вычисляется на суперкомпьютере месяцами, «лёгкой» аппроксимирующей суррогатной моделью
- замена сложной нейросети, которая обучается неделями на больших данных, «лёгкой» аппроксимирующей нейросетью с минимизацией числа нейронов и связей

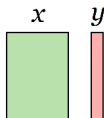
## Обучение с использованием привилегированной информации

LUPI — Learning Using Privileged Information

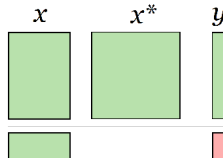
с учителем



без учителя



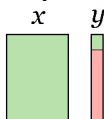
привилегированное (LUPI)



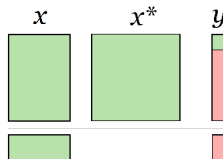
частичное



трансдуктивное



частичное LUPI



V. Vapnik, A. Vashist. A new learning paradigm: Learning Using Privileged Information // Neural Networks. 2009.

## Примеры задач с привилегированной информацией $x^*$

- $x$  — первичная (1D) структура белка  
 $x^*$  — третичная (3D) структура белка  
 $y$  — иерархическая классификация функции белка
- $x$  — предыстория временного ряда  
 $x^*$  — информация о будущем поведении ряда  
 $y$  — прогноз следующей точки ряда
- $x$  — текстовый документ  
 $x^*$  — выделенные ключевые слова или фразы  
 $y$  — категория документа
- $x$  — пара (запрос, документ)  
 $x^*$  — выделенные ассессором ключевые слова или фразы  
 $y$  — оценка релевантности



## Задача обучения с привилегированной информацией

Раздельное обучение модели-ученика и **модели-учителя**:

$$\sum_{i=1}^{\ell} \mathcal{L}(a(x_i, w), y_i) \rightarrow \min_w \quad \sum_{i=1}^{\ell} \mathcal{L}(a(x_i^*, w^*), y_i) \rightarrow \min_{w^*}$$

Модель-ученик обучается повторять ошибки **модели-учителя**:

$$\sum_{i=1}^{\ell} \mathcal{L}(a(x_i, w), y_i) + \mu \mathcal{L}(a(x_i, w), a(x_i^*, w^*)) \rightarrow \min_w$$

Совместное обучение модели-ученика и **модели-учителя**:

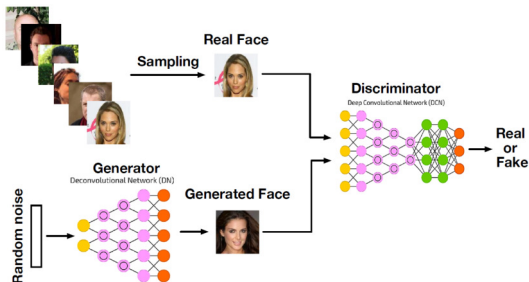
$$\sum_{i=1}^{\ell} \mathcal{L}(a(x_i, w), y_i) + \lambda \mathcal{L}(a(x_i^*, w^*), y_i) + \\ + \mu \mathcal{L}(a(x_i, w), a(x_i^*, w^*)) \rightarrow \min_{w, w^*}$$

---

*D.Lopez-Paz, L.Bottou, B.Scholkopf, V.Vapnik.* Unifying distillation and privileged information. 2016.

## Генеративная состязательная сеть (Generative Adversarial Net)

Генератор  $G(z)$  учится порождать объекты  $x$  из шума  $z$   
Дискриминатор  $D(x)$  учится отличать их от реальных объектов



Antonia Creswell et al. Generative Adversarial Networks: an overview. 2017.

Zhengwei Wang, Qi She, Tomas Ward. Generative Adversarial Networks: a survey and taxonomy. 2019.

Chris Nicholson. A Beginner's Guide to Generative Adversarial Networks.

<https://pathmind.com/wiki/generative-adversarial-network-gan>. 2019.

## Постановка задачи GAN

**Дано:** выборка объектов  $\{x_i\}_{i=1}^m$  из  $X$

**Найти:**

вероятностную генеративную модель  $G(z, \alpha): x \sim p(x|z, \alpha)$

вероятностную дискриминативную модель  $D(x, \beta) = p(1|x, \beta)$

**Критерий:**

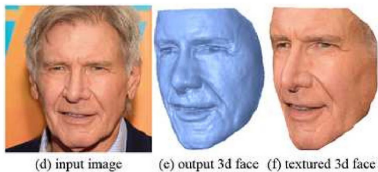
обучение дискриминативной модели  $D$ :

$$\sum_{i=1}^m \ln D(x_i, \beta) + \ln(1 - D(G(z_i, \alpha), \beta)) \rightarrow \max_{\beta}$$

обучение генеративной модели  $G$  по случайному шуму  $\{z_i\}_{i=1}^m$ :

$$\sum_{i=1}^m \ln(1 - D(G(z_i, \alpha), \beta)) \rightarrow \min_{\alpha}$$

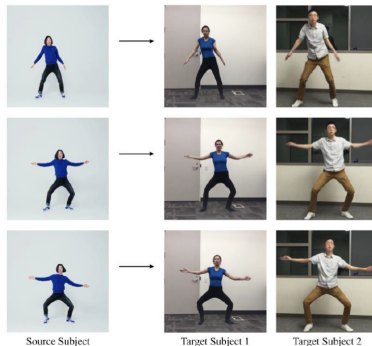
## Примеры GAN для синтеза изображений и видео



(d) input image

(e) output 3d face

(f) textured 3d face



Source Subject

Target Subject 1

Target Subject 2

*Chuan Li, Michael Wand.* Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks. 2016.

*Xiaoxing Zeng, Xiaojiang Peng, Yu Qiao.* DF2Net: A Dense Fine Finer Network for Detailed 3D Face Reconstruction. ICCV-2019.

*Caroline Chan, Shiry Ginosar, Tinghui Zhou, Alexei A. Efros.* Everybody Dance Now. ICCV-2019.

**Когда несколько моделей обучаются одновременно:**

- Автокодировщики: кодер и декодер
- Автокодировщики для классификации или кластеризации
- Многозадачное обучение
- Обучение с привилегированной информацией
- Состязательные сети (GAN) для генерации фейк-объектов

**Когда несколько моделей обучаются последовательно:**

- Перенос обучения (transfer learning)
- Предобучение глубоких сетей для векторизации объектов
- Самостоятельное обучение (self-supervised learning)
- Дистилляция и суррогатное моделирование

**Обучение без учителя всё чаще используется для оптимизации части модели по большим данным**