Глубокое обучение

Бекезин Никита

14 июля 2022

Лекция 14: Автоэнкодеры

Agenda

- Autoencoders
- Manifold learning

Автокодировщики

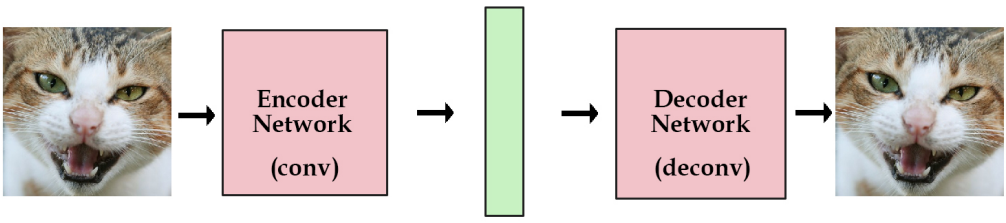
Автокодировщики

- Когда данных много, мы хотим понизить их размерность. Классическое машинное обучение позволяте делать это с помощью метода главных компонент, tsne и других методов. Нейросети также позволяют решать подобную задачу сжатия с минимальными потерями.

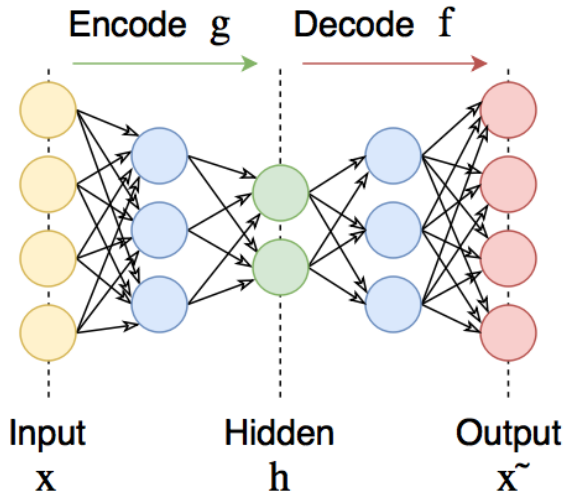
- Понижение размерности — задача обучения без учителя

- Давайте превратим её в обучение с учителем!

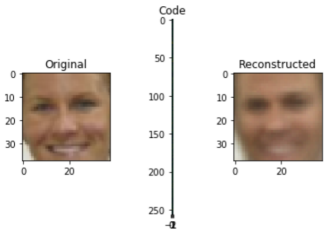
Автокодировщики

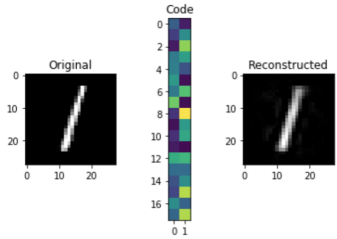


Автокодировщики



Пример сжатия



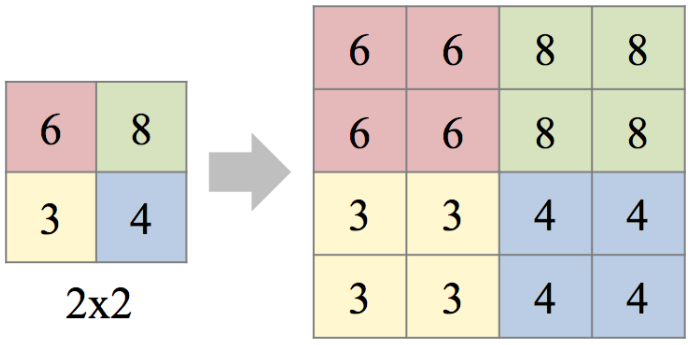


Как используют

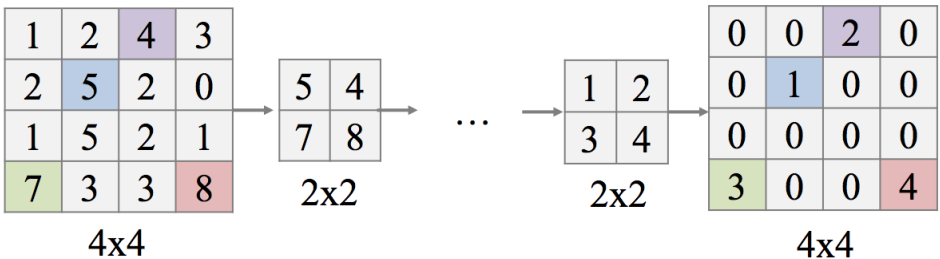
- Для предобучения сетей. Именно так в 2005 началась революция.
- Скрытое представление признаков можно использовать в других моделях в качестве фичей.
- Конструкцию автокодировшика можно немного модернизировать для решения других задач, например для генерирования подписи по картинке (про это позже)

Сверточные слои декодера

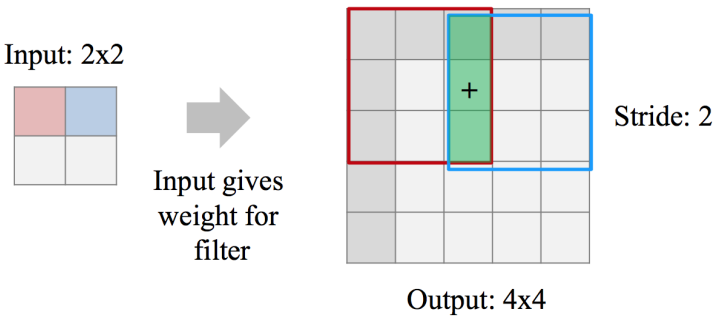
Nearest neighbor unpooling



Max unpooling

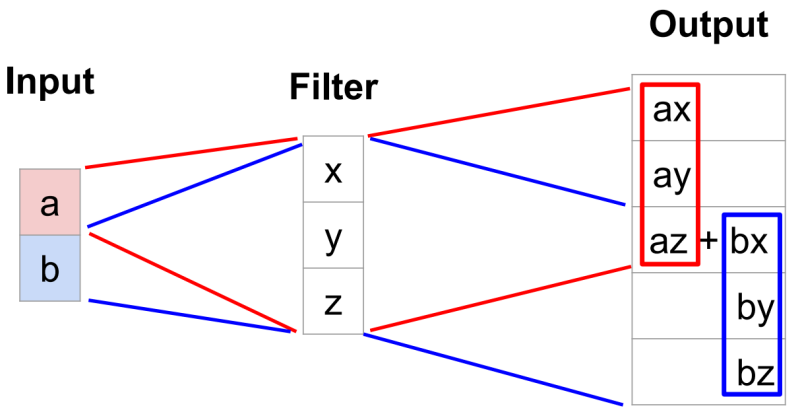


Learnable unpooling: Transpose convolution

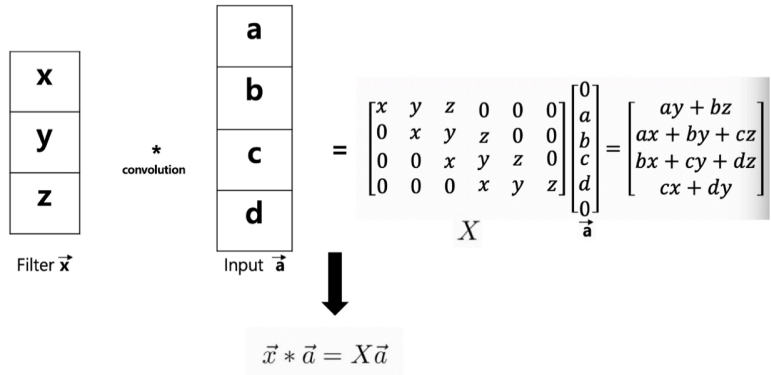


- Каждую клетку надо распаковать в 4 клетки $\Rightarrow$ свёртка 3×3 со сдвигом 2

Пример:



Почему называется Transpose convolution:



Source of a picture

Почему называется Transpose convolution:

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

Source of a picture

Почему называется Transpose convolution:

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

Source of a picture

Типы автокодировщиков

Обычный автокодировщик

В обычном автокодировщике восстанавливаем следующую последовательность: $x = f(g(x))$, изменяя 2 функции $f(x)$ и $g(x)$. Loss в нашем случае - $L(x, f(g(x)))$ Тождественно не можем выучить из-за искусственного ограничения количества нейронов в середине.

- Однослойной автокодировщик по своему действию совпадает с PCA
- Если задаем многослойный автокодировщик может находить достаточно сложные особенности в данных (по сути, правильной архитектуре - любые)
- Можно делать и сверточные, если работаем с изображениями

Denoise autoencoder

Мы пытаемся не просто восстановить выход по входу, но и ещё искусственно добавляем шум к входным данным.
Посутимыпытаемрешитьследующуюзадачу $x = f(g(\hat{x}))$, где $\hat{x}$ зашумленные входные данные. Для изображений шум можно задавать 2-мя способами - затемнять какую-часть изображения или добавлять шум к каждому пикселю. Весь остальной процесс обучения совпадает с обычным автокодировщиком.

Разреженный автокодировщик

Разреженный автокодировщик Теперь мы к нашему лосу добавляем регуляризатор. $L(x, g(f(x))) + \omega(h)$, где $g(h)$ - выход декодера, $h = f(x)$ -выход энкодера, и ограничение накладывается на энкодер. Как ограничения обычно использую L1 или L2 норму. Такой автокодировщик не сможет полностью выучить картинку из-за штрафа при любой архитектуре. Он может расширяться к выходу, пытаясь разложить сигнал на множество статистически независимых сигналов. Используют его также как и обычный автокодировщик, если требуется чтобы получающиеся латентные векторы были более линейно-независимые. Из-за того, что он пытается разложить один сигнал на множество, иногда его для разложения сигнала на составляющие - аналог вайвлет преобразований для аудио.

Собираем свои автокодировщики