Modele generatywne 1: manifold hypothesis

Jacek Tabor

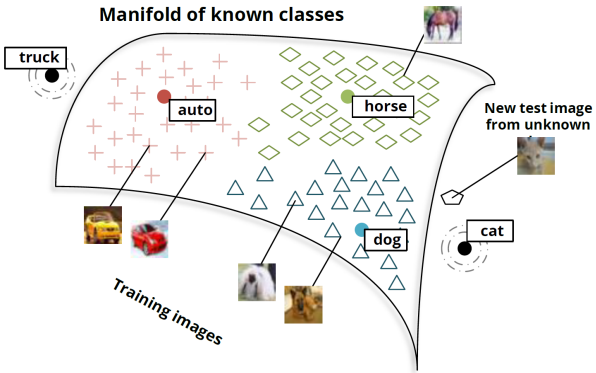
11 października 2023

1 Manifold hypothesis: hipoteza rozmaitości

Twierdzenie 1. *Rzeczywiste dane $X \subset \mathbb{R}^D$ w rzeczywistości leżą na podrozmaitości (podprzestrzeni) $M \subset \mathbb{R}^D$ o dużo mniejszym wymiarze $d << D$.*

Inaczej mówiąc elementy X możemy opisać za pomocą d parametrów, gdzie $d << D$.

Pytanie w jaki sposób można wyznaczyć to d i M.

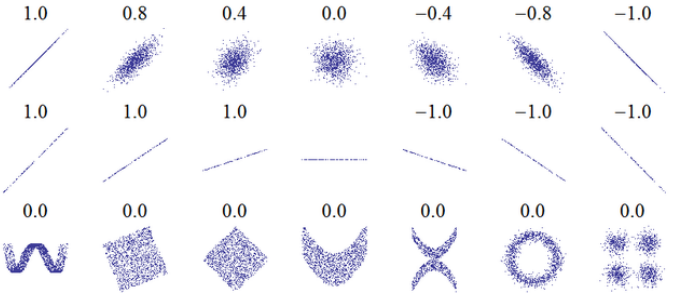


Rysunek 1: Rozmaitość na której leżą dane

2 Dane na płaszczyźnie: intuicja

Dane $(x, y) = (x_i, y_i) \subset \mathbb{R}^2$. Okazuje się, że wtedy możemy łatwo ocenić czy można zredukować wymiar – przydadzą się współczynnik kowariancji i korelacji.

Średnia $\mathbf{mean}x = \bar{x} = \frac{1}{n}\sum_i x_i$ – średni element.



Rysunek 2: Ilustracja współczynnika korelacji

Odchylenie standardowa i wariancja

$$Vx = \frac{1}{n}\sum_i (\bar{x} - x_i)^2,\ \sigma x = \sqrt{Vx}.$$

Współczynnik kowariancji między x i y: mierzymy zależność między x_i i y_i:

$$\mathbf{cov}(x, y) = \frac{1}{n}\sum_i (x_i - \bar{x})(y_i - \bar{y})$$

Po znormalnizowaniu mierzy zależność między zmiennymi:

$$r_{x,y} = \frac{\mathbf{cov}(x,y)}{\sigma x \cdot \sigma y} \in [-1,1].$$

3 Sformułowanie problemu w przypadku liniowym

Mamy zbiór danych $X = (x_i)_{i=1..N} \subset \mathbb{R}^D$ oraz ustalony wymiar $d < D$.

Szukamy podprzestrzeni liniowej wymiaru d, dla której zminimalizowana zostanie funkcja straty:

$$\frac{1}{N}\sum_i d^2(x_i; M),$$

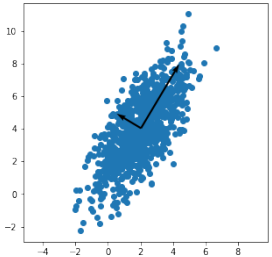
gdzie $d(x; M)$ oznacza odległość x od podprzestrzeni M.

Rzutowanie na podprzestrzeń Zakładamy, że mamy podprzestrzeń liniową M rozpiętą na

$$m + r_1 v_1 + \ldots r_d v_d,$$

gdzie $m, v_i \in \mathbb{R}^D$, $r_i \in \mathbb{R}$ oraz v_i są ortonormalne (czyli prostopadłe do siebie i o długości 1). Wtedy najbliższy punkt do x w M (czyli rzut prostopadły x na M) jest dany wzorem

$$P_M x = m + \sum_{i=1}^{d} \langle x - m, v_i \rangle v_i,$$



Rysunek 3: Algorytm PCA wyznacza kierunki największej wariancji danych (po lewej). Wykorzystując te kierunki dokonujemy obrotu układu współrzędnych w ten sposób, że kolejne współrzędne nowego układu stanowią kierunki wyznaczone przez PCA w kolejności ważności.

gdzie $\langle v, w\rangle = v^T w$ to iloczyn skalarny.

UWAGA: dla ogólnych rozmaitości (poza przypadkiem podprzestrzeni liniowych) znalezienie najbliższego punktu jest zazwyczaj bardzo trudne albo niemożliwe.

PCA: optymalne rozwiązanie Dane $X = (x_i) \subset \mathbb{R}^D$. Średnia

$$\mathbf{mean} X = \frac{1}{n} \sum_i x_i$$

centralny element.

Macierz kowiarancji to macierz składająca się ze współczynników kowariancji odpowiednich współrzędnych zbioru X

$$\mathbf{cov}X = [\mathbf{cov}(X^l, X^m)]_{l.m=1..D} \in \mathbb{R}^{D\times D},$$

gdzie X^l to ciąg składający się z l-tej współrzędnej zbioru X

Ona zawiera pełne informacje o liniowej zależności danych:

- odpowiada nam dla wszystkie wymiarowości
- wektory własne pozwalają rzutować
- błąd który dokonujemy, to suma wszystkich dalszych wartości własnych

Niech $m = \bar{X}$ oznacza średnią, a $\mathbf{cov}X$ macierz kowariancji zbioru X. Przez λ_i oznaczam kolejne wartości własne tej macierzy (ustawione malejąco), a v_i odpowiadające im wektory własne.

Dla dowolnego wymiaru d:

- optymalna przestrzeń jest rozpięta przez $m + r_1v_1 + \ldots + r_dv_d$, gdzie $r_i \in \mathbb{R}$
- optymalne rzutowanie punkt x (czyli najbliższy)

$$P_M x = m + \sum_{i=1}^{d} \langle x - m, v_i\rangle v_i$$

- sumaryczny uśredniony błąd to $\sum_{j>d} \lambda_j$.

Jeśli przyjrzymy się co tak naprawdę robi PCA, to okaże się, że $v_1, \ldots, v_N$ są wektorami parami prostopadłymi, które opisują kierunki największej wariancji (rozrzutu) danych (patrz rysunek 3). Aby dokonać redukcji zawężamy się do k najbardziej informatywnych kierunków – wartości własne wskazują jaka jest wariancja na poszczególnych kierunkach.

Dodatkowo, ustalenie optymalnego wymiaru d jest trudne. Często zamiast ustalać d, ustala się procent wariancji danych, którą chcemy wyjaśnić (zazwyczaj ustala się 95% lub 99%). Jeśli chcemy wyjaśnić co najmniej $\alpha \in [0, 1]$ wariancji, to bierzemy takie d, by:

$$\frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{D} \lambda_i} \geq \alpha,$$

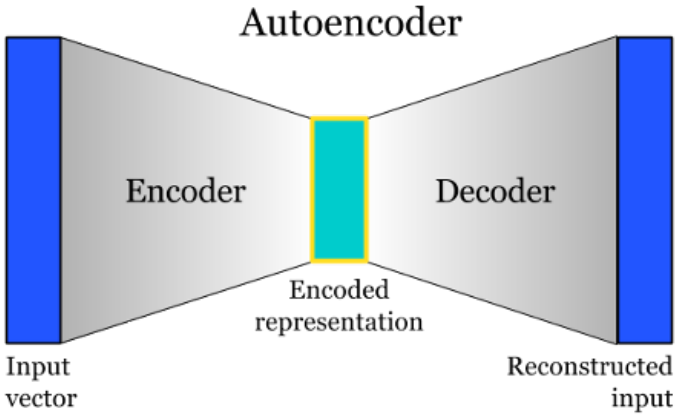
gdzie λ_i to wartości własne posortowane malejąco.

4 Funkcja straty (loss function) i AutoEnkoder

Doprecyzowanie (dla ustalonego d).

Autoenkoder. Mam zbiór danych $X \subset \mathbb{R}^D$ oraz $d \leq D$. Chcemy zakodować elementy X za pomocą d wymiarów. Szukamy $\mathcal{E} : \mathbb{R}^D \to Z, \mathcal{D} : Z \to \mathbb{R}^D$:

$$\mathcal{D}\mathcal{E}x \approx x$$



Rysunek 4: RYSUNEK AUTOENKODERA

Formalizacja:

$$\frac{1}{n}\sum_i \|x_i - \mathcal{D}\mathcal{E}x_i\|^2.$$

Z to tak zwana przestrzeń ukryta (latent space). Tam jest tworzona reprezentcja danych (można dokonywać manipulacji).

5 AutoEnkoder

Koncepcja AutoEnkodera jest nieliniowym uogólnieniem PCA. Chcemy zbudować taką niżej wymiarową reprezentację, która pozwala na rekonstrukcję oryginalnych punktów z tej reprezentacji z możliwie małym błędem.

Architektura AutoEnkodera składa się z enkodera $\mathcal{E} : \mathbb{R}^D \to \mathbb{R}^N$, który buduje reprezentację i dekodera $\mathcal{D} : \mathbb{R}^N \to \mathbb{R}^D$, który służy do jej uczenia (rysunek 5). Reprezentacja oznaczana jest jako Z i nazywana przestrzenią ukrytą (ang. latent). Załóżmy zatem, że $z = \mathcal{E}(x)$ jest reprezentacją x. Docelowo chcemy, aby z tej reprezentacji dało się odzyskać x z możliwie małym błędem. Zatem żądamy aby:

$$x \approx \tilde{x} = \mathcal{D}(\mathcal{E}(x)).$$

Do realizacji powyższego celu wykorzystujemy standardowy błąd kwadratowy, zwany w przypadku AE błędem rekonstrukcji:

$$\text{Rec\_Error} = \frac{1}{n} \sum_i \|x_i - \mathcal{D}(\mathcal{E}(x_i))\|^2.$$

Zauważmy, że gdyby udało nam się powyższą wartość zminimalizować do zera, to nasze oryginalne dane leżałyby idealnie na N-wymiarowej (gdzie N to wymiar przestrzeni reprezentacji) rozmaitości zadanej przez $M = \mathcal{D}(Z)$.

W AutoEnkoderze konwolucyjnym część kodująca jest definiowana przez warstwy konwolucyjne ze stridem większym od 1 lub przez kombinację warstw konwolucyjnych ze stridem równym 1 oraz warstw pooling. Dekoder próbuje wykonać „odwrotne” operacje, zamiast zmniejszać rozmiary warstw powiększa je, tak aby wyjście z sieci pasowa-

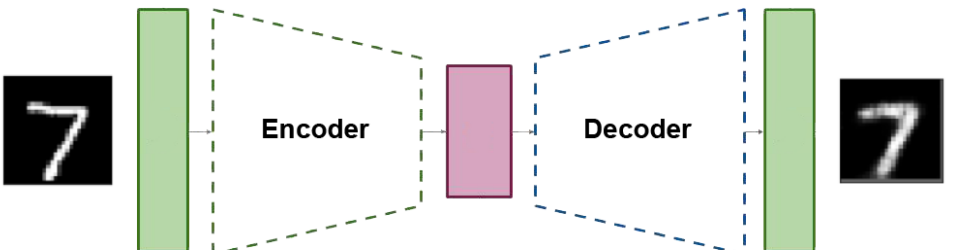
ło do rozmiaru obrazka wejściowego. W tym celu stosuje się zazwyczaj transponowane konwolucje lub interpolację dwuliniową.

MNIST Zaimplementowaliśmy przykładową sieć AE i ją nauczyliśmy na zbiorze MNIST.

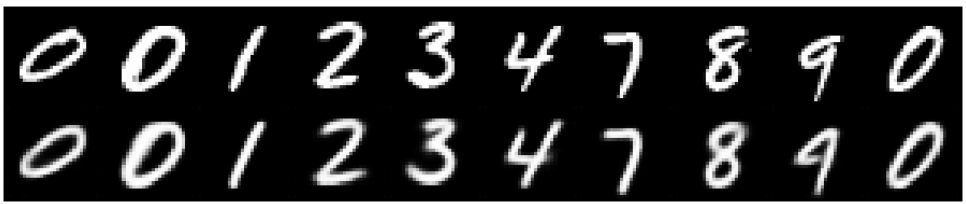
Oryginalny wymiar $D = 28 \cdot 28 = 784$, wymiar przestrzeni ukrytej $d = 32$.

Losowe rekonstrukcje oryginalnych obrazków prezentujemy na rysunku 6. Średni błąd kwadratowy na całym zbiorze testowym dla nauczonej sieci wyniósł 0.01234.

Architekturę AutoEnkodera często stosuje się do usuwania szumu (denoising autonenkoder) czy do uzupełniania brakujących danych (missing imputation). Stosujemy ją także w modelach kompresji, ale wtedy często dodatkowo stosujemy techniki wymuszające dyskretyzację.



Rysunek 5: Schemat architektury sieci AutoEnkoder. *AutoEnkoder* (AE) dla danych graficznych wykorzystuje warstwy konwolucyjne do wyodrębniania cech obrazu wejściowego, reprezentując go w niżej wymiarowej przestrzeni zwanej *przestrzenią ukrytą* (latent space), a następnie próbuje odtworzyć obraz wejściowy z tej reprezentacji tak, aby pasował do wejściowego obrazka. Zatem w architekturze sieci AutoEnkodera możemy wyróżnić część kodującą oraz część dekodującą



Rysunek 6: Górny wiersz przedstawia losowo wybrane oryginalne obrazki ze zbioru testowego MNIST, zaś na dole ich rekonstrukcje uzyskane z sieci AE trenowanej na zbiorze testowym MNIST.