Modele generatywne na chmurach punktów 3D

Jakub Zadrożny

Maj 2019

1 Wprowadzenie

Reprezentacja punktów 3d jest bardzo ważna bo bla bla bla...

2 Powiązane badania

2.1 Chmury punktów 3D

Silnym narzędziem do pracy z chmurami punktów 3D jest architektura **PointNet** [1], która umożliwa ekstrakcję cech (definiowanych przez wyuczalne parametry) z wejściowej chmury punktów. Istotną zaletą tej atchitektury jest odporność na permutacje punktów chmury wejściowej. Inne rozwiązania osiągnęły ten efekt przetwarzając chmury punktów na inne formaty (np. wykorzystujące voxele), przez co rozmiar przetwarzanych danych wzrasta. Dzięki zastosowaniu serii splotów jednowymiarowych, PointNet osiąga ten sam cel, zachowując względną prostotę modelu.

2.2 Modele generatywne

Klasycznym wynikiem, który zapoczątkował prace nad modelami generatywnymi, jest autoenkoder wariacyjny (**VAE**) [2]. Jego najważniejszą zaletą jest probabilistyczny charakter, dzięki czemu możemy żądać, aby zmienne pośrednie (*latent*) były zradnomizowane i pochodziły z pewnego zadanego rozkładu prawdopodobieństwa (z pewnymi ograniczeniami). Dzięki randomizacji zmiennych pośrednich uzyskujemy zagęszczenie przestrzeni reprezentacji,

przez co niemal każdy punkt może zostać zdekodowany na realistyczne dane. Umożliwia to realizowanie zadań takich jak generowanie syntetycznych, realistycznych danych lub interpolacja znaczeniowa obiektów.

2.3 Półnadzorowana i nienadzorowana klasteryzacja

3 Metody

3.1 Autoenkoder wariacyjny

Autoenkodery wariacyjne są dobrze znane i zbadane pod kątem niektórych zagadnień, np. analizy obrazów. Jednak zastosowanie ich do chmur punktów 3D wymaga opracowania dokładniejszej metody.

Dalej zakładamy, że dysponujemy zbiorem danych

$$\mathcal{X} = \{x_i \in \mathbb{R}^d\}_{i \in I}$$

dla pewnego d – wymiaru. Ponadto zakładamy, że dane te są obserwacjami zmiennej losowej o rozkładzie następującej postaci

$$f(z, x; \theta) = f(z) f(x|z; \theta) \tag{1}$$

gdzie $z \in \mathbb{R}^k$ dla pewnego k (wymiar przestrzeni reprezentacji) – zmienna pośrednia (*latent*), $x \in \mathbb{R}^d$ – zmienna obserwowana, a Θ – parametr rozkładu. Dodatkowo niech

$$\begin{aligned} z &\sim \mathcal{N}(0, I_k) \\ x|z &\sim \mathcal{N}(\mu_x(z; \theta), \mu_\sigma(z; \theta) I_d) \end{aligned} \tag{2}$$

gdzie μ_x, μ_σ są skomplikowanymi obliczeniami wykonywanymi przez sieć neuronową sparametryzowaną przez θ.

3.1.1 ELBO

Naszym celem jest odtworzenie parametrów rozkładu generującego θ oraz rozkładu $f(z|x; \theta)$, który nazywamy *reprezentacją* danych generowanych przez proces opsiany w (1) i (2).

Niestety z powodu zastosowania skomplikowanych, nieliniowych transformacji dokładne odtworzenie rozkładu $f(z|x; \theta)$ jest niemożliwe. W tym celu wprowadzamy pewne przybliżenie tego rozkładu – nazwijmy je $g(z|x; \phi)$.

Niech $g(z|x;\theta)$ będzie gęstością rozkładu normalnego ze średnią $\rho_x(x;\phi)$ i wariancją $\rho_\sigma(x;\phi)$, gdzie ρ_x, ρ_σ są reprezentowane przez sieci neuronowe parametryzowane przez ϕ. Wtedy

$$\begin{aligned} KL(g(z|x;\phi)||f(z|x;\theta)) &= \mathbb{E}_{z\sim g(z|x;\phi)}\left[-\log\frac{f(z|x;\theta)}{g(z|x;\phi)}\right] = \\ &= \mathbb{E}_{z\sim g(z|x;\phi)}\left[-\log\frac{f(z|x;\theta)f(x;\theta)}{g(z|x;\phi)f(x;\theta)}\right] = \\ &= \mathbb{E}_{z\sim g(z|x;\phi)}\left[-\log\frac{f(z|x;\theta)f(x;\theta)}{g(z|x;\phi)} + \log f(x;\theta)\right] \\ &= \mathbb{E}_{z\sim g(z|x;\phi)}\left[-\log\frac{f(z,x;\theta)}{g(z|x;\phi)}\right] + \log f(x;\theta) \end{aligned} \tag{3}$$

gdzie $KL(\cdot\,||\cdot\,)$ jest odległością Kullbacka-Leiblera [3]. Zatem

$$\log f(x;\theta) = KL(g(z|x;\phi)||f(z|x;\theta)) + \mathbb{E}_{z\sim g(z|x;\phi)}\left[\log\frac{f(z,x;\theta)}{g(z|x;\phi)}\right] \tag{4}$$

Ponieważ $KL(\cdot\,||\cdot\,) \geqslant 0$, więc

$$\begin{aligned} \log f(x;\theta) &\geqslant \mathbb{E}_{z\sim g(z|x;\phi)}\left[\log\frac{f(z,x;\theta)}{g(z|x;\phi)}\right] = \\ &= \mathbb{E}_{z\sim g(z|x;\phi)}\left[\log\frac{f(x|z;\theta)f(z)}{g(z|x;\phi)}\right] = \\ &= \mathbb{E}_{z\sim g(z|x;\phi)}\left[\log f(x|z;\theta)\right] - \mathbb{E}_{z\sim g(z|x;\phi)}\left[-\log\frac{f(z)}{g(z|x;\phi)}\right] = \\ &= \mathbb{E}_{z\sim g(z|x;\phi)}\left[\log f(x|z;\theta)\right] - KL(g(z|x;\phi)||f(z)) \end{aligned} \tag{5}$$

Zatem dla dowolnego rozkładu aproksymującego $g(z|x;\phi)$ otrzymujemy dolne ograniczenie na prawdopodobieństwo wygenerowania zaobserwowanych danych. Dlatego część wzoru po prawej stronie od ostatniej równości nazywamy **ELBO** (*evidence lower bound*). Możemy zauważyć, że ostateczna postać wzoru (5) składa się z dwóch części naturalnie odpowiadającymi dwóm celom, które chcemy zoptymalizować: pierwszy składnik odpowiada jakości rekonstrukcji obserwacji ze zmiennej ukrytej z (dlatego nazywany jest kosztem rekonstrukcji), natomiast drugi to odległość KL rozkładu aproksymującego $f(z|x;\theta)$ od naszego założenia na jego temat.

3.1.2 Zadanie optymalizacyjne

Chcemy znaleźć układ parametrów $< \theta, \phi >$, który daje najlepszą gwarancję na prawdopodobieństwo wygenerowania zaobserwowanych danych (ELBO). W tym celu posłużymy się lekko zmodyfikowanym algorytmem SGD. Naszym zadaniem jest znalezienie

$$\begin{aligned} \max_{\theta,\phi} \hat{\mathcal{L}}(\mathcal{X}, \theta, \phi) &= \sum_{i \in I} \mathcal{L}(x_i, \theta, \phi) = \\ &= \sum_{i \in I} \left(\mathbb{E}_{z \sim g(z|x_i;\phi)} \left[\log f(x_i|z;\theta) \right] - KL(g(z|x_i;\phi)||f(z)) \right) \end{aligned} \tag{6}$$

Ponieważ bardziej naturalnym zadaniem jest minimalizowanie funkcji kosztu, to rozwiążemy równoważne zadanie znalezienia

$$\min_{\theta,\phi} -\hat{\mathcal{L}}(\mathcal{X}, \theta, \phi) \tag{7}$$

Żeby posłużyć się algorytmem SGD musimy umieć wyliczać i różniczkować oba składniki funckji $\mathcal{L}$.

3.1.3 Koszt KL

Odległość KL dwóch rozkładów normalnych o następujących parametrach

$$\begin{aligned} \mathcal{N}_0 &\sim \mathcal{N}(\mu_0, \Sigma_0) \\ \mathcal{N}_1 &\sim \mathcal{N}(\mu_1, \Sigma_1) \end{aligned}$$

dla pewncyh $\mu_0,\ \mu_1 \in \mathbb{R}^k,\ \Sigma_0, \Sigma_1 \in \mathbb{R}^{k \times k}$, wynosi

$$KL(\mathcal{N}_0||\mathcal{N}_1) = \frac{1}{2} \left(\mathrm{tr}(\Sigma_1^{-1}\Sigma_0) + (\mu_1 - \mu_0)^T \Sigma_1^{-1} (\mu_1 - \mu_0) - k + \log \frac{\det \Sigma_1}{\det \Sigma_0} \right)$$

Ponieważ zakładamy, że $f(z)$ jest rozkładem $z \sim \mathcal{N}(0, I_k)$, więc

$$KL(g(z|x;\phi)||f(z)) = \frac{1}{2} \sum_{i=1}^{k} \left(\rho_x(x;\phi)_i^2 + \rho_\sigma(x;\phi)_i^2 - \log(\rho_\sigma(x;\phi)_i^2) - 1 \right) \tag{8}$$

Wzór (8) można wyliczać i różniczkować analitycznie.

3.1.4 Koszt rekonstrukcji

Drugiego składnika funckji $\mathcal{L}$, czyli kosztu rekonstrukcji, nie da się wyznaczyć analitycznie. Aby objeść ten problem, możemy metodą Monte Carlo oszacować wartość oczekiwaną przez średnią

$$\mathbb{E}_{z\sim g(z|x;\phi)}\left[\log f(x_i|z;\theta)\right] \sim \frac{1}{m}\sum_{i=1}^{m} -\log f(x|z_i;\theta)$$

gdzie $z_i \sim g(z|x;\phi)$. Taką wartość potrafimy już wyliczyć, ale nie potrafimy propagować gradientu do parametrów ϕ przez zaobserwowane wartości z_i.

Wprowadzimy repearametryzację zmiennych z_i – możemy zauważyć, że zmienna $z_i = \rho_x(x;\phi) + \epsilon_i\rho_\sigma(x;\phi)$ gdzie $\epsilon_i \sim \mathcal{N}(0, I_k)$ ma rozkład $g(z|x;\phi)$ a ponadto możemy propagować gradient do parametrów ϕ. Otrzymaliśmy zatem następujące przybliżenie na $\mathcal{L}$

$$\mathbb{E}_{z\sim g(z|x;\phi)}\left[\log f(x_i|z;\theta)\right] \sim \frac{1}{m}\sum_{i=1}^{m} -\log f(x|z_i = \rho_x(x;\phi) + \epsilon_i\rho_\sigma(x;\phi);\theta)$$

gdzie $\epsilon_i \sim \mathcal{N}(0, I_k)$

Ponieważ $x|z \sim \mathcal{N}(\mu_x(z;\theta), \mu_\sigma(z;\theta)I_d)$, więc

$$-\log f(x|z;\theta) = \sum_{i=1}^{d}\left(\frac{1}{2}\log(2\pi) + \log(\mu_\sigma(z;\theta)_i) + \frac{(x-\mu_x(z;\theta)_i)^2}{2\mu_\sigma(z;\theta)_i^2}\right)$$

jednak metryka ta niezbyt dobrze nadaje się do chmur punktów, ponieważ np. chcielibyśmy uznawać permutację punktów oryginalnej chmury za dobrą rekonstrukcję. Dlatego zamiast wyliczać *stricte* $\log f(x|z;\theta)$ skorzystamy ze zmodyfikowanego *Chamfer distance* danego wzorem

$$CD(\mathcal{X}_1, \mathcal{X}_2) = \sum_{x\in\mathcal{X}_1}\min_{y\in\mathcal{X}_2}||x-y||_2^2 + \sum_{x\in\mathcal{X}_2}\min_{y\in\mathcal{X}_1}||x-y||_2^2 \tag{9}$$

gdzie $\mathcal{X}_1$, $\mathcal{X}_2$ są zbiorami punktów wielowymiarowych. Ściślej mówiąc, możemy potraktować $\mu_x(z;\theta) \in \mathbb{R}^{3\cdot m}$ jako chmurę m punktów trójwymiarowych, oznaczmy ją $\hat{y}$. Ponadto dla $y \in \hat{y}$ niech $\sigma(y)$ oznacza 3-elementowy wektor wariancji $\mu_\sigma(z;\theta)$ utworzony ze składowych odpowiadających y. Za koszt rekonstrukcji przyjmiemy

$$\begin{aligned}\mathcal{L}_{rec}(\hat{x}|z,\theta,\phi) = &\sum_{x\in\hat{x}}\min_{y\in\hat{y}}\left(-\log p_{y,\sigma(y)}(x)\right) + \\ &+\sum_{y\in\hat{y}}\min_{x\in\hat{x}}\left(-\log p_{x,\sigma(y)}(y)\right)\end{aligned} \tag{10}$$

gdzie $p_{v,s}(x)$ jest gęstością rozkładu normalnego o średniej v i macierzy kowariancji sI w punkcie x.

Po usunięcu stałych wyrazów można to zapisać jako

$$\begin{aligned}\mathcal{L}_{rec}(\hat{x}|z,\theta,\phi) = \sum_{x\in\hat{x}} \min_{y\in\hat{y}} \sum_{i=1}^{3} \left(\log(\sigma(y)_i) + \frac{(x_i - y_i)^2}{2\sigma(y)_i^2} \right) + \\ + \sum_{y\in\hat{y}} \min_{x\in\hat{x}} \sum_{i=1}^{3} \left(\log(\sigma(y)_i) + \frac{(x_i - y_i)^2}{2\sigma(y)_i^2} \right)\end{aligned} \tag{11}$$

W obecnej wersji modelu dla uproszczenia przyjęto, że $\mu_\sigma(z;\theta) = \alpha$ dla wszystkich z i niezależnie od parametrów θ (tzn. przyjęto stałą wariancję dla danych wyjściowych). Wtedy wzór (11) upraszcza się do

$$\begin{aligned}\mathcal{L}_{rec}(\hat{x}|z,\theta,\phi) &= \frac{1}{2\alpha^2} \left(\sum_{x\in\hat{x}} \min_{y\in\hat{y}} ||x-y||_2^2 + \sum_{y\in\hat{y}} \min_{x\in\hat{x}} ||x-y||_2^2 \right) = \\ &= \frac{1}{2\alpha^2} CD(\hat{x},\hat{y})\end{aligned} \tag{12}$$

3.2 Model *M2* Kingmy

3.3 GMVAE

4 Implementacja i architektura modeli

4.1 VAE

Metoda opsiana w sekcji 3.1 zawiera abstrakcyjne obliczenia ρ (nazywane dalej **enkoderem**) oraz μ (nazywane dalej **dekoderem**) realizowane przez sieci neuronowe. W tym rozdziale przedstawiono konkretną architekturę nadaną tym obliczeniom.

Enkoder wykorzystuje przytoczoną w sekcji 2.1 i wprowadzoną w [1] architekturę PointNet do ektrakcji 1024 sparametryzowanych (wyuczalnych) cech (*features*) z chmur wejściowych. Te cechy są dalej traktowane jako wejście do niewielkiej sieci MLP (warstwa wejściowa – 1024 neurony, jedna warstwa ukryta – 1024 neurony i warstwa wyjściowa – 256 neuronów). Pomiędzy siecią PointNet i warstwami sieci MLP (z wyjątkiem ostatniej) znajduje się aktywacja ReLU i warstwy *batch normalization* [4]. Ostatnia warstwa reprezentuje parametry rozkładów zmiennych pośrednich (po 128 liczb dla ρ_x, ρ_σ), co oznacza, że wymiar zmiennej pośredniej wynosi 128.

Dekoder posiada znacznie prostszą architekturę – jest to sieć MLP o 4 ukrytych warstwach (kolejno 512, 1024, 1024 i 2048 neuronów) z aktywacjami ReLU (poza warstwą wyjściową). Ostatnia warstwa składa się z $3 * 2048$ neuronów, co odpowiada wymiarowi wejściowych chmur punktów i reprezentuje parametry μ_x.

Pomiędzy enkoderem i dekoderem wykonywany jest trik repearametryzacyjny, tzn. wyliczenie zmiennej pośredniej jako $\rho_x(x_i; \phi) + \epsilon\rho_\sigma(x_i; \phi)$, gdzie $\epsilon \sim \mathcal{N}(0, I)$.

4.2 M2

4.3 GMVAE

4.4 Zbiór danych i trenowanie

Model został wytrenowany na jednej klasie obiektów wybranej z połączonych zbiorów ModelNet40 [5] oraz ShapeNet [6] po uprzednim przekształceniu obu zbiorów do chmur punktów wymiaru 2048 x 3 zgodnie z metodą opisaną w [7]. Do treningu zbiór podzielono na części treningowe i testowe w proporcjach 80%-20%.

Do optymalizacji funckji kosztu danej wzorem (6) użyto metody ADAM [8]. Początkowe *learning rate* wynosiło $2 * 10^{-4}$ i zmniejszało się dwukrotnie, co każde 500 epok. Model trenowany był przez 3000 epok (duża liczba epok wynika z niewielkiej ilości próbek).

Na rys **??** przedstawiona została zmiana całkowitej funkcji kosztu, kosztu rekonstrukcji oraz kosztu KL w czasie. Możemy zaobserwować charakterystyczną dla autoenkoderów wariacyjnych dynamikę kosztu KL – podczas gdy koszt rekonstrukcji stale maleje, koszt KL na początku szybko wzrasta i powoli spada z biegiem kolejnych epok. Początkowy wzrost kosztu KL odpowiada pakowaniu przez model wielu informacji do zmiennej pośredniej, żeby uzyskać drastyczny spadek ogromnego kosztu rekonstrukcji. Jednak gdy koszt rekonstrukcji spada, model zaczyna optymalizować informacje przechowywane w zmiennej pośredniej tak, żeby jej rozład był podobny do $\mathcal{N}(0, I)$, czemu odpowiada powolny spadek kosztu KL.

5 Wyniki eksperymentalne

Na wytrenowanym modelu przeprowadzono kilka eksperymentów mających na celu sprawdzić zarówno zdolności modelu do dokładnej rekonstrukcji, jak i jego możliwości generatywne.

5.1 Rekonstrukcje

Na rys **??** znajdują się oryginalne chmury punktów ze zbioru *Modelnet40* (po lewej) wraz z chmurami zrekonstruowanymi przez dekoder na podstawie zmiennych pośrednich wyliczonych przez enkoder (po prawej).

Jedną z miar jakości rekonstrukcji autoenkoderów jest pokrycie (*coverage*). Pokrycie definiujemy jako procent próbek ze zbioru danych, dla którego najbliższa (w tym przypadku w sensie *Chamfer Distance*) inna próbka spośród całego zbioru dancyh oraz wszystkich rekonstrukcji pochodzi ze zbioru rekonstrukcji. Dla wytrenowanego modelu pokrycie na rozważanym podzbiorze *Modelnetu40* wynosi ...%.

Jedną z największych zalet VAE są jego zdolności generatywne. Klasycznym sposobem demonstracji zdolności generatywnych modelu jest skonstruowanie takiej interpolacji pomiędzy dwoma różnymi obiektami, że każdy z jej kroków pośrednich jest *podobny* (wizualnie lub z użyciem metryki) do próbek z oryginalnego zbioru danych. Rys **??** przedstawiają interpolacje wykonane przez rozważany powyżej model. Można zaobserwować, że kolejne kroki interpolacji coraz bardziej upodobniają obiekt źródłowy do docelowego, jednocześnie zachowując typowe cechy obiektów z oryginalnego zbioru danych.

Własnością wyróżniającą VAE na tle innych enkoderów jest możliwość odgórnego zadania rozkładu zmiennych pośrednich, który model będzie musiał osiągnąć. Dla rozważanego modelu zadano rozkład standardowy wielowymiarowy normalny (o średniej w 0 i identycznościowej macierzy kowariancji). Dzięki temu, możemy tworzyć nowe, nieistniejące w zbiorze danych próbki, przez wylosowanie zmiennej pośredniej z wybranego powyżej rozkładu i przekazanie jej do dekodera. Rys. **??** przedstawia chmury otrzymane w ten właśnie sposób. Możemy zauważyć, że powstałe próbki dobrze pasują do reszty zbioru danych i ponadto prezentują dużą różnorodność (pochodzą z różnych podklas), co świadczy o dużych możliwościach generatywnych modelu.

Rys **??** przedstawiają wykresy median rozkładów zwróconych przez enkoder po zredukowaniu do dwóch wymiarów metodami PCA (górny) i t-SNE

(dolny). Możemy zauważyć, że na obu wykresach mediany rozmieszczone są dość jednostajnie i gęsto na kole o środku w punkcie $(0, 0)$. Oznacza to, że reprezentacja próbek ze zbioru jest *ściśnięta* do zera i gęsta, co umożliwia przeprowadzenie interpolacji oraz tworzenie syntetycznych próbek na podstawie wylosowanych zmiennych pośrednich (jak wyżej).

6 Wnioski

Literatura

[1] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 652–660, 2017.

[2] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[3] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[4] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," pp. 448–456, 2015.

[5] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

[6] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[7] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3d point clouds," *arXiv preprint arXiv:1707.02392*, 2017.

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.