Vysoká škola ekonomická v Praze
Fakulta informatiky a statistiky

Variační autoenkodér a úlohy pozorování v latentním prostoru

BAKALÁŘSKÁ PRÁCE

Studijní program: Aplikovaná informatika

Autor: Tomáš Faltejsek
Vedoucí práce: Ing. Ondřej Vadinský, Ph.D.
Konzultant práce: full consultant's name (incl. degrees)

Praha, květen 2023

Poděkování

Thanks.

Abstract

Jedním z předních rysů lidské inteligence je intuice a schopnost představovat si nové objekty. Variační autoenkodér je inovací na poli pravděpodobnostních modelů, umožňující architekturu modelů schopných syntézy zcela nových dat s využitím pozorování atributů v latentním prostoru. Teoretická charakteristika a možnosti využití variačního autoenkodéru jsou předmětem této bakalářské práce.

Keywords

keyword, important term, another topic, and another one

Obsah

Seznam obrázků

Note: Add a list of figures if the number of figures in the thesis text exceeds 20. A list of diagrams is applicable only if the author distinguishes between a figure and a diagram. The list of diagrams is included if the number of diagrams exceeds 20. This thesis template does not distinguish between a figure and a diagram.

Seznam tabulek

Note: Add a list of tables if the number of tables used in the thesis exceeds 20.

Seznam použitých zkratek

BCC Blind Carbon Copy
CC Carbon Copy
CERT Computer Emergency Response Team
CSS Cascading Styleheets
DOI Digital Object Identifier
HTML Hypertext Markup Language
REST Representational State Transfer
SOAP Simple Object Access Protocol
URI Uniform Resource Identifier
URL Uniform Resource Locator
XML eXtended Markup Language

Note: Add a list of abbreviations if the number of abbreviations used in the thesis exceeds 20 and the abbreviations used are not common.

Úvod

Introduction is a compulsory part of the bachelor's / diploma thesis. The introduction is an introduction to the topic. It elaborates the chosen topic, briefly puts it into context (there may also be a description of the motivation to write the work) and answers the question why the topic was chosen. It puts the topic into context and justifies its necessity and the topicality of the solution. It contains an explicit goal of the work. The text of the thesis goal is identical with the text that is given in the bachelor's thesis assignment, ie with the text that is given in the InSIS system and which is also given in the Abstract section.

Part of the introduction is also a brief introduction to the process of processing the work (a separate part of the actual text of the work is devoted to the method of processing). The introduction may also include a description of the motivation to write the work.

The introduction to the diploma thesis must be more elaborate - this is stated in more detail in the Requirements of the diploma thesis within the Intranet for FIS students.

Here are some sample chapters that recommend how a bachelor's / master's thesis should be set. They primarily describe the use of the LaTeX template, but general advice will also serve users of other systems well.

1. Východiska variačního autoenkodéru

1.1 Strojové učení

Strojové učení je podoblast umělé inteligence zabývající se algoritmy a technikami, které počítačovým systémům umožňují *učit se* z dat.

> „*Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.*“ (Samuel, 1967)[1]

1.1.1 Algoritmus strojového učení

Definici algoritmu strojového učení výstižně shrnuje následující definice (Mitchell, 1997, str. 2):

> „*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*“

Algoritmy strojového učení lze obecně rozdělit na čtyři třídy – učení s učitelem (*supervised learning*), učení bez učitele (*unsupervised learning*), kombinaci učení s učitelem a učení bez učitele (*semi-supervised learning*) a posilovaného učení (*reinforcement learning*).

Toto dělení vychází ze zkušenosti E, resp. míry, do jaké má algoritmus strojového učení **povoleno** interagovat s datovou sadou (resp. jakou možnost taková datová sada nabízí). (I. Goodfellow et al., 2016)

Pro předmět této práce budou blíže představeny pouze oblasti **učení s učitelem** a **učení bez učitele**.

1.1.2 Učení s učitelem

Zkušenost E algoritmů učení bez učitele vychází z datové sady, která může mít celou řadu vlastností a zároveň je **předem známá klasifikace každého datového bodu do definovaných tříd**. Na základě této asociace je natrénovaný algoritmus schopen provést přiřazení dosud neznámého objektu do jedné z tříd definovaných v trénovací sadě dat.

[1] Parafrázováno

Učení s učitelem zahrnuje pozorování několika příkladů náhodného vektoru $\mathbf{x}$ a asociované hodnoty (či vektoru) $\mathbf{y}$. Následuje učení se predikovat $\mathbf{y}$ z $\mathbf{x}$, často na základě odhadu $p(\mathbf{y} \mid \mathbf{x})$.

Samotný termín *učení s učitelem* vychází ze situace, kdy je cílová třída $\mathbf{y}$ poskytnuta jakýmsi instruktorem či učitelem, který systému strojového učení ukazuje očekávané chování. (I. Goodfellow et al., 2016)

1.1.3 Učení bez učitele

Ve vztahu k definici (viz Podsekce 1.1.1) lze říci, že zkušenost E algoritmů učení bez učitele vychází z datové sady, která může mít celou řadu vlastností. Cílem trénování algoritmů učení bez učitele je **naučit se užitečné a charakteristické vlastnosti o struktuře vstupní datové sady**. V kontextu hlubokého učení je pak obvyklým cílem algoritmu naučit se celé rozdělení pravděpodobnosti, které generuje původní datovou sadu (ať už explicitně – např. odhad hustoty, či implicitně – např. úlohy syntézy dat a odstranění šumu). Mezi další techniky strojového učení bez učitele se, mimo jiné, řadí shluková analýza.

Obecně lze říct, že učení bez učitele zahrnuje pozorování několika příkladů náhodného vektoru $\mathbf{x}$ na základě kterého se snaží implicitně či explicitně *naučit* rozdělení pravděpodobnosti $p(\mathbf{x})$, případně *užitečné vlastnosti* tohoto pravděpodobnostního rozdělení.

Na rozdíl od Podsekce 1.1.2, název *učení bez učitele* napovídá, že v procesu učení není zapojen žádný *instruktor* či *učitel*, a tak systém strojového učení sám musí vyvodit smysl a užitečné vlastnosti předložené datové sady. (I. Goodfellow et al., 2016)

1.1.4 „No free lunch theorem“ pro strojové učení

Dle teorie má algoritmus strojového učení schopnost generalizace i z konečné množiny trénovacích dat. Toto tvrzení je ale v rozporu s elementárními principy logiky – indukce obecných pravidel z omezeného vzorku dat je logicky nevalidní. Chceme-li provést indukci obecného pravidla, které popisuje každý prvek množiny, musíme mít k dispozici informaci o každém prvku z množiny. (I. Goodfellow et al., 2016)

Pro logické vyvrácení takto naučené generalizace stačí byť jeden vzorek, který je s tímto pravidlem v nesouladu a nebyl součástí trénovací množiny (tzv. *black swan paradox*).

Oblast strojového učení se tomuto paradoxu z části vyhýbá tím, že pracuje pouze s **pravděpodobnostními pravidly** (oproti zcela určitým pravidlům jako v logické indukci). Algoritmy strojového učení hledají pravidla, která jsou tzv. *probably approximately correct*. (Valiant, 1984)

Ani tento trik však kompletně neřeší představený problém. Zjednodušeně, **"No free lunch theorem"** pro strojové učení tvrdí, že každý klasifikační algoritmus má v průměru, skrze

všechny distribuce generující data, **stejnou chybovost** při klasifikování dosud nepozorovaných datových bodů. (Wolpert, 1996) Jinými slovy, **žádný algoritmus strojového učení není univerzálně lepší než kterýkoliv jiný algoritmus strojového učení**.

Toto tvrzení je pravdivé až při zohlednění *všech možných distribucí generujících data* – tedy v teoretické rovině. Lze pozorovat, že v praktických aplikacích je možné navrhnout algoritmus, který si v určitých distribucích vede v průměru lépe než ostatní algoritmy. (I. Goodfellow et al., 2016)

Cílem této práce je představit Variační autoenkodér – architekturu umělé neuronové sítě, která se těší dobrým výsledkům ve vybraných úlohách učení bez učitele, představených v Kapitola 4, ale v důsledku má i své nedostatky (představené v Kapitola 3) v ostatních třídách úloh.

1.1.5 Regularizace

„No free lunch theorem“ pro strojové učení (představený v Podsekce 1.1.4) implikuje nutnost návrhu algoritmu strojového učení pro konkrétní úlohu, chceme-li aby jeho výkonnost byla vyšší než výkonnost ostatních algoritmů v průměru. Toho lze docílit *zabudováním* určité sady preferencí přímo do algoritmu strojového učení (předpokladem je, že tyto preference jsou v souladu s cílovým problémem, který se algoritmus snaží řešit). (I. Goodfellow et al., 2016)

Chování a výkonnost algoritmu strojového učení lze ovlivnit zvolením velikosti množiny funkcí (a následně konkrétní podoby jejich identity), které jsou v jeho prostoru hypotéz povoleny[2]. V prostoru hypotéz algoritmu lze rovněž vyjádřit preferenci jednoho řešení před druhým. To znamená, že obě takové funkce budou přípustné, ale jedna z nich má preferenci (tedy nepreferovaná funkce bude zvolena právě když je její evaluace vůči trénovací sadě *výrazně* lepší, než preferovaná funkce)[3]. Případně můžeme funkci (nebo množinu funkcí) z prostoru hypotéz vyřadit kompletně (resp. vyjádřit tak nekonečně velkou míru neupřednostnění takové funkce, či množiny funkcí). (I. Goodfellow et al., 2016)

Obecně můžeme regularizovat model přičtením *trestu* k jeho ztrátové funkci. Tento trest nazýváme **regularizační prvek** (*regularizer*) a značíme jej Ω.

Regularizaci pak definujeme jako:

> „*Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error. Regularization is one of the central concerns of the field of machine learning, rivaled in its importance only by optimization.*“ (I. Goodfellow et al., 2016)

[2] Například prostor hypotéz lineární regrese je složen z množiny lineárních funkcí jejího vstupu. Tedy lineární regrese zřejmě bude mít problém věrohodně predikovat hodnotu $\sin(x)$ z x (a stejně tak řešit další nelineární problémy).

[3] Například *weight decay*.

Formu regularizace je tedy nutné pečlivě zvolit s ohledem na typ úlohy, který má algoritmus za cíl řešit. I autoenkodéry (a variační autoenkodéry) mohou být navrhnuty k řešení celé řady problémů. Jednotlivé metody regularizace autoenkodérů jsou představeny v Kapitola 2 a jejich následné aplikace v Kapitola 4.

1.2 Umělá neuronová síť

Umělá neuronová síť je model strojového učení inspirovaný přírodou. Zdá se být intuitivní, že chceme-li napodobit lidskou inteligenci, měli bychom se pro inspiraci podívat na architekturu lidského mozku. V průběhu času se však konstrukce umělých neuronových sítí začala jejich přírodnímu protějšku podstatně vzdalovat. Řada architektur umělých neuronových sítí tvoří biologicky nerealistický model, byť z této původní myšlenky vychází (stejně tak jako letadla vycházejí z přírodního vzoru létajících ptáků, pohybem svými křídly se ve skutečnosti ve vzduchu neodrážejí). (Geron, 2019)

Představení kompletních principů umělých neuronových sítí není východiskem variačního autoenkodéru, nýbrž svým obsahem pokrývají několik monografií – pro úvod například (Chollet, 2017), (Geron, 2019). V této sekci tedy budou představeny pouze stěžejní techniky využívané autoenkodéry.

1.2.1 Architektura umělé neuronové sítě

Pojem **architektura** rozumíme **celkovou strukturu** umělé neuronové sítě – počet neuronů a způsob jakým jsou tyto neurony mezi sebou propojeny, jejich aktivační funkce a podobně.

Většina umělých neuronových sítí je organizována do skupin neuronů zvaných **vrstvy**. Architektury umělých neuronových sítí poté uspořádávají tyto vrstvy do zřetězené struktury, kde každá vrstva je funkcí předcházející vrstvy. (I. Goodfellow et al., 2016)

V takové struktuře je první vrstva dána následovně:

$$\boldsymbol{h}^{(1)} = g^{(1)}\left(\mathbf{W}^{(1)\top}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right), \tag{1.1}$$

a druhá vrstva je dána následovně:

$$\boldsymbol{h}^{(2)} = g^{(2)}\left(\mathbf{W}^{(2)\top}\boldsymbol{h}^{(1)} + \boldsymbol{b}^{(2)}\right). \tag{1.2}$$

Mezi hlavní architektonická rozhodnutí při návrhu umělé neuronové sítě patří volba **hloubky sítě** a **šířky každé z vrstev**.

1.2.2 Anatomie umělé neuronové sítě

Proces trénování umělé neuronové sítě z pravidla zahrnuje následující objekty (Chollet, 2017):

- *Vrstvy* ze kterých je následně složena *síť* (resp. *model*)
- *Vstupní data* (a případně jejich *cílové třídy*)
- *Ztrátová funkce*, která slouží jako signál zpětné vazby použití pro učení modelu
- *Optimizér*, který modifikuje parametry umělé neuronové sítě (např. váhy)

1.2.3 Dopředná umělá neuronová síť

Dopředná umělá neuronová síť je velmi podstatná pro návrh (hlubokých) modelů strojového učení. Cílem dopředné umělé neuronové sítě je aproximovat nějakou funkci $f*$. [4] Dopředná umělá neuronová síť definuje mapovací funkci $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$ a učí se hodnotu parametrů $\boldsymbol{\theta}$, jejichž výsledky je nejlepší aproximace cílové funkce. (I. Goodfellow et al., 2016)

Dopředné umělé neuronové **sítě** nazýváme:

- *Dopředné*, jelikož v této architektuře neexistují žádné zpětnovazební propojení, které by sloužily jako vstup zpět pro sebe sama [5].
- *Sítě*, jelikož jsou typicky reprezentovány kompozicí více různých funkcí. Model takové sítě je asociován s orientovaným acyklickým grafem, který popisuje, jakým způsobem je tato kompozice tvořena. Mějme tři funkce $f^{(1)}$, $f^{(2)}$, $f^{(3)}$, které jsou zřetězeny následovně: $f(\boldsymbol{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\boldsymbol{x})))$. Takto zřetězené struktury nazýváme **vrstvy** umělé neuronové sítě. Délku tohoto zřetězení nazýváme **hloubkou** modelu – odtud *hluboké* neuronové sítě (umělé neuronové sítě s 2 a více skrytými vrstvami). (I. Goodfellow et al., 2016) [6].

1.2.4 Perceptron

Jedná se o jednu z nejjednodušších architektur umělé neuronové sítě (a *model umělého neuronu*) představenou v (Rosenblatt, 1957) inspirovanou principy Hebbovského učení (Hebb, 1949). Perceptron přichází s důležitým principem **numerických hodnot vstupů, výstupů** (oproti pouhým binárním hodnotám) – tzv. *linear treshold unit, LTU* a **vah** propojení mezi jednotlivými neurony. Perceptron byl později kritizován (Minsky a Papert, 1969) za jeho neschopnost řešit triviální problémy (např. *XOR* klasifikace), což eventuálně vedlo ke krátkodobé stagnaci konekcionismu.

Jak se ale ukázalo, některé z těchto limitací lze vyřešit uspořádáním více Perceptronů za sebe do vrstev. (Rumelhart a McClelland, 1987)

Takto uspořádaná umělá neuronová síť se nazývá Vícevrstvý Perceptron.

[4] Například pro klasifikátor, $y = f*(x)$ mapuje vstup x do kategorie y. Převzato z (I. Goodfellow et al., 2016).

[5] Je-li dopředná umělé neuronové sítě obohacena o zpětnovazební propojení se sebou sama, nazýváme ji **Rekurentní neuronová síť**.

[6] V tomto případě je $f^{(1)}$ **první vrstva**, $f^{(2)}$ je **druhá vrstva** a $f^{(3)}$ **třetí vrstva**.

1.2.5 Vícevrstvý Perceptron a Algoritmus zpětné propagace

Vícevrstvý Perceptron je tvořen umělou neuronovou sítí, která se skládá z jedné vstupní vrstvy, **jedné nebo více skrytých vrstev** LTU jednotek, a jedné výstupní vrstvy rovněž složené z LTU jednotek. Součástí každé vrstvy (vyjma výstupní) je tzv. **bias** neuron, který je **plně propojený** s další vrstvou sítě. (Geron, 2019)

Pokud má umělá neuronová síť dvě a více skrytých vrstev, nazýváme ji **hlubokou neuronovou sítí**.

Jak již víme, Vícevrstvý Perceptron adresuje limitace Perceptronu (viz Podsekce 1.2.4). Článek (Rumelhart a McClelland, 1987) ale představil i další revoluční myšlenku – algoritmus **zpětné propagace** [7] (*backpropagation*).

Algoritmus zpětné propagace lze velmi zjednodušeně interpretovat následovně: Pro každou trénovací instanci je vypočten výstup každého jejího neuronu každé jednotlivé vrstvy. Poté je změřena výstupní chyba celé sítě (například střední kvadratická chyba, MSE) a vypočten podíl, kterým každý neuron poslední skryté vrstvy přispěl k hodnotě každého neuronu výstupní vrstvy. Následně je stejným způsobem měřeno, jak moc byla ovlivněna hodnota jednotlivých neuronů poslední skryté vrstvy hodnotami neuronů předchozí skryté vrstvy – a podobný proces se opakuje než algoritmus dosáhne vstupní vrstvy. (Geron, 2019)

Průchod algoritmu sítí lze intuitivně popsat následovně: Pro každou trénovací instanci algoritmus zpětné propagace nejprve učiní predikci cílové hodnoty – tzv. *forward pass*. Poté změří chybu této predikce. Následně zpětně projde každou vrstvu sítě za účelem změření míry přispění k této chybě každým propojením (a jeho vahou) – tzv. *reverse pass*. Závěrem algoritmus *lehce* upraví váhy jednotlivých propojení za účelem snížení chyby – tzv. *Gradient Descent step*. (Geron, 2019)

Pro správný chod algoritmu autoři (Rumelhart a McClelland, 1987) do architektury Vícevrstvého Perceptronu zanesli další zásadní změnu – jako tzv. **aktivační funkci** využili logistickou regresi (zcela běžné je využití i dalších aktivačních funkcí, například ReLU).

1.2.6 Univerzální aproximační teorém

Univerzální aproximační teorém[8] (Hornik, M. B. Stinchcombe et al., 1989), (Cybenko, 1989) tvrdí, že dopředná umělá neuronová síť (s alespoň jednou skrytou vrstvou a *potlačující* aktivační funkcí – např. logistická funkce) je schopna aproximovat libovolnou (borelovsky měřitelnou) funkci s *libovolnou mírou přesnosti*[9].

[7] Algoritmus zpětné propagace byl ve skutečnosti nezávisle objeven více výzkumníky, počínaje (Werbos, 1974). Převzato z (Geron, 2019).

[8] Universal approximation theorem

[9] Derivacemi dopředné umělé neuronové sítě lze stejně tak aproximovat derivace cílové funkce (Hornik, M. Stinchcombe et al., 1990).

Důsledkem Univerzálního aproximačního teorému je, že dostatečně velký **Vícevrstvý perceptron zvládne *reprezentovat* libovolnou funkci**. Ale neexistuje zde jistota, že se algoritmus strojového učení bude schopen takovou funkci *vždy* naučit (I. Goodfellow et al., 2016):

- Zvolený optimizér algoritmu strojového učení použitý pro trénování nemusí být schopen najít parametry umělé neuronové sítě, které korespondují s cílovou funkcí.
- Algoritmus nemusí vybrat správnou funkci v důsledku přeučení.

Dopředné umělé neuronové sítě tak nabízejí univerzální systém pro **reprezentaci** libovolné funkce. **Pro libovolnou funkci existuje dopředná umělá neuronová síť, která ji aproximuje** [10].

1.3 Redukce dimenzionality

Do oblasti redukce dimenzionality patří celá řada technik pro práci s vysokodimenzionálními daty. Cílem této oblasti, na rozdíl od regresních problémů, není predikovat hodnotu cílové proměnné – ale porozumět tvaru dat, se kterými pracuje. Typickou úlohou redukce dimenzionality dat je sestrojit **nízkodimenzionální reprezentaci**, která zachytí *většinu významu* původní, vysokodimenzionální, reprezentace. Tento jev nazýváme hledáním **salientních vlastností** původní sady dat. (Phillips, 2021)

Redukci dimenzionality lze chápat jako úlohu učení bez učitele, ve které se algoritmus strojového učení učí mapování z vysokodimenzionálního prostoru $\boldsymbol{x} \in \mathbb{R}^D$ do nízkodimenzionálního latentního prostoru $\boldsymbol{z} \in \mathbb{R}^L$. (Murphy, 2022)

1.3.1 „The curse of dimensionality“

S rostoucím počtem vstupních proměnných exponenciálně roste počet vzorků dat nutný pro *libovolně přesnou* aproximaci dané funkce. V důsledku je tedy s rostoucí dimenzí vstupních dat značně degradováno chování většiny algoritmů (strojového učení). Tento problém je známý pod termínem **curse of dimensionality**. (Bellman, 1957)

I proto došlo ke vzniku oblasti zvané *feature engineering*. Feature engineering je oblast, která se, mimo jiné, zabývá disciplínou selekce vlastností dat, které budou použity při trénování modelu. Pro automatizovanou selekci vlastností existuje celá řada technik. Výběr podprostoru, který *nejlépe* reprezentuje vlastnosti původních dat je NP-těžký kombinatorický problém (*exhaustive search through all the subsets of features*). Tyto techniky dokonce často vyhodnocují každou vstupní proměnnou nezávisle, což může vést ke zkresleným závěrům o jejich

[10] S ohledem na Podsekce 1.1.4 neexistuje žádný univerzální způsob pro prozkoumání trénovací množiny dat a zvolení funkce, která bude libovolně přesně generalizovat na instance dat, které nejsou součástí této trénovací množiny.

významnosti – naopak je běžné, že proměnné začínají vykazovat určitou míru významnosti **až při vzájemném využití**. (Stańczyk a Jain, 2015)

Výše stanovené důvody vedly k emergenci další disciplíny, a to **extrakce vlastností**, o níž pojednává Podsekce 1.3.2.

1.3.2 Extrakce vlastností

Cílem extrakce vlastností *feature extraction* je najít reprezentaci vstupních dat, která je vhodná pro algoritmus strojového učení, který se chystáme využít (jelikož původní reprezentace může být z mnoha důvodů nevhodná – například vysokodimenzionální). Typicky tak musí dojít k redukci dimenzionality vstupních dat. (H. Liu a Motoda, 1998)

K extrakci nových vlastností lze dojít mnoha způsoby. Existují techniky založené na hledání lineárních kombinací původních vstupních vlastností, například Analýza hlavních komponent (*PCA Analýza*) nebo Lineární diskriminační analýza (*LDA Analýza*).

Pro nelineární redukci dimenzionality lze využít technik tzv. *manifold learningu*, viz (I. Goodfellow et al., 2016, kap. 5.11.3.).

1.3.3 Analýza hlavních komponent (PCA)

Jednou z nejrozšířenějších metod pro redukci dimenzionality je PCA analýza (*principal component analysis*).

V principu se jedná o nalezení lineární a ortogonální projekce vysokodimenzionálních dat $\boldsymbol{x} \in \mathbb{R}^D$ do nízkodimenzionálního podprostoru $\boldsymbol{z} \in \mathbb{R}^L$, tak aby tato nízkodimenzionální reprezentace byla *dobrou aproximací* původních dat. Provedeme-li projekci (resp. **kódování**) $\boldsymbol{x}$, získáme $\mathbf{z} = \mathbf{W}^\top \boldsymbol{x}$ (funkci kódování označme e), a následně **dekódujeme** $\boldsymbol{z}$ abychom získali $\hat{\boldsymbol{x}} = \mathbf{W}\boldsymbol{z}$ (funkci dekódování označme d), pak chceme aby $\hat{\boldsymbol{x}}$ bylo co možná nejblíže $\boldsymbol{x}$ (měřeno pomocí ℓ_2 vzdálenosti). (Murphy, 2022)

Konkrétně můžeme definovat následující **chybu rekonstrukce** (Murphy, 2022):

$$\mathcal{L}(\mathbf{W}) \triangleq \frac{1}{N}\|\boldsymbol{x} - d(e(\boldsymbol{x}_n; \mathbf{W}); \mathbf{W})\|_2^2 \tag{1.3}$$

kde e a d jsou lineární funkce (*mappings*).

Na PCA analýzu tak lze pohlížet jako na algoritmus učení bez učitele, který se učí reprezentaci vstupních dat. Tato reprezentace má dvě kritéria (I. Goodfellow et al., 2016):

- PCA se učí reprezentaci vstupních dat, která má menší dimenzi než původní vstupní reprezentace.
- PCA se učí reprezentaci, jejíž elementy mezi sebou nemají žádnou lineární korelaci.

1.4 Pravděpodobnostní modely

Jak napovídá definice z Podsekce 1.1.1, existuje mnoho druhů úloh T, ke kterým je možné využít algoritmy strojového učení. Tato práce se zabývá především úlohami v pravděpodobnostním kontextu. Oblast strojového učení, která tento typ úloh řeší se nazývá **pravděpodobnostní modelování**.

Pravděpodobnostní modely strojového učení pracují s cílovou proměnnou (třídou) jako s **náhodnou proměnnou, která podléhá nějakému rozdělení pravděpodobnosti**. Toto rozdělení popisuje váženou množinu hodnot, kterých může náhodná proměnná nabýt. (Murphy, 2022)

Pro nutnost adopce pravděpodobnostních modelů hovoří dva hlavní důvody – jedná se o **optimální přístup** pro **rozhodování s faktorem neurčitosti** a poté, pravděpodobnostní modelování je *přirozené* pro řadu inženýrských disciplín (například stochastická optimalizace, operační výzkum, ekonometrie, informační teorie a další). (Murphy, 2022)

1.5 Model využívající latentních proměnných

Model využívající latentních proměnných (*latent variable model*) označíme $p_\theta(\mathbf{x}, \mathbf{z})$.

Jsou-li rozdělení pravděpodobností tohoto modelu parametrizovány neuronovou sítí, nazveme jej **hluboký model využívající latentních proměnných** (*deep latent variable model*, dále jen *DLVM*). DLVM může být podmíněný vůči libovolnému kontextu, tedy $p_\theta(\mathbf{x}, \mathbf{z}|\mathbf{y})$. Důležitou vlastností (a výhodou) DLVM je, že marginální rozdělení p_θ může být velmi komplexní (tedy obsahovat téměř jakékoliv závislosti). Tato expresivní vlastnost činí DLVM atraktivní pro **aproximaci složitých skrytých rozdělení pravděpodobnosti** $p^*(\mathbf{x})$. (Diederik P. Kingma a Max Welling, 2019)

Jedním z nejjednodušších (a nejznámějších) DLVM modelů je faktorizace s následující strukturou (Diederik P. Kingma a Max Welling, 2019):

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{z}) p_\theta(\mathbf{x} \mid \mathbf{z}), \tag{1.4}$$

kde $p_\theta(\mathbf{z})$ a $p_\theta(\mathbf{x} \mid \mathbf{z})$ jsou předem dány.

1.5.1 Latentní proměnné

Latentní proměnné jsou proměnné, které jsou součástí modelu, ale nejsou přímo pozorovatelné (a tím pádem nejsou součástí vstupních dat). Pro označení takových proměnných používáme $\mathbf{z}$. U latentních proměnných se předpokládá jejich vliv na hodnotu cílové proměnné (resp. cílových proměnných). (Diederik P. Kingma a Max Welling, 2019)

1.6 Kullback–Lieblerova divergence

Kullback–Lieblerova divergence (Kullback a Leibler, 1951), dále jen KL divergence, vyjadřuje míru *podobnosti* rozdělení pravděpodobnosti p vůči jinému rozdělení pravděpodobnosti q [11].

Nechť $p(x)$ a $q(x)$ jsou rozdělení pravděpodobnosti diskrétní náhodné veličiny pro x [12]. A dále $p(x) > 0$ a $q(x) > 0$ pro každé $x \in X$. Pak definujeme KL divergenci následovně (Murphy, 2022):

$$D_{KL}(p(x)\|q(x)) = \sum_{x \in X} p(x) \ln \frac{p(x)}{q(x)} \tag{1.5}$$

A spojitou verzi KL divergence následovně (Murphy, 2022):

$$D_{KL}(p(x)\|q(x)) = \int_{-\infty}^{\infty} p(x) \ln \frac{p(x)}{q(x)} dx \tag{1.6}$$

KL divergence je úzce spjata s *relativní entropií* a *teorií informace*. Lze ji tedy interpretovat jako *počet bitů, které jsou* ***dodatečně*** *potřebné* pro kódování vzorků dat nevhodnou distribucí q v porovnání s kódováním stejných vzorků dat jejich původní distribucí p.

Existuje celá řada dalších (a přesnějších) interpretací KL divergence. Pro formální zavedení a definici vlastností KL divergence odkazuji na (Murphy, 2023, kap. 5.1).

Minimalizace KL divergence je využita jako součást trénovacího procesu variačního autoenkodéru (viz Kapitola 3).

1.7 Model využívající latentní proměnné

Čím komplexnější jsou závislosti mezi dimenzemi generativního modelu, tím složitější je jeho trénování.

Mějme například jednoduchou úlohu generování obrázků ručně psaných číslic 0-9. Snažíme-li se generovat obrázek číslice 5, tak víme, že pravá strana obrázku nemůže obsahovat pravou stranu obrázku číslice 0 (v opačném případě se zřejmě nemůže jednat o obrázek číslice 5).

Intuitivně je tak vhodnou strategií, aby model nejprve učinil rozhodnutí o tom, jakou číslici se chystá generovat, než začne konkrétním pixelům přidělovat hodnoty. Takové rozhodnutí nazýváme *latentní proměnnou*. Tedy, před tím, než model začne generovat obrázek, náhodně

[11] Nutno zdůraznit, že ačkoliv lze intuitivně hovořit o *podobnosti* či *vzdálenosti* mezi distribucí p a q, KL divergence **není metrická**. Není symetrická (tedy nesplňuje M3) a neplatí v ní trojúhelníková nerovnost (tedy nesplňuje M4). (Phillips, 2021)

[12] Tedy součet $p(x)$ a $q(x) = 1$

vybere vzorek z z množiny $[0, \ldots, 9]$ a při generovaní obrázku se ujistí, že všechny vygenerované pixely souhlasí s charakteristikami této číslice. V tomto kontextu *latentní* znamená, že pro danou číslici, kterou model vygeneroval, nemusí být nutně známo jaké nastavení této latentní proměnné bylo modelem použito (to bychom museli odvodit například počítačovým viděním). (Doersch, 2021)

Před tím, než lze prohlásit že model je *reprezentací* vstupního datasetu, musíme se ujistit, že pro každý datový bod X v tomto datasetu existuje alespoň jedna konfigurace latentních proměnných modelu, která zajistí že model vygeneruje obrázek *velice podobný* X.

Formálně řekneme, že ve vysokodimenzionálním prostoru $\mathcal{Z}$ existuje vektor latentních proměnných z, ze kterého lze podle *nějaké* hustoty pravděpodobnosti (*probability densiti function, PDF*) $P(z)$, definované skrze $\mathcal{Z}$, jednoduše vzorkovat X. Dále, máme-li množinu deterministických funkcí $f(z; \theta)$, parametrizovaných vektorem θ v nějakém prostoru Θ, kde $f : \mathcal{Z} \times \Theta \to \mathcal{X}$. Byť je f deterministická, je-li z náhodné a θ neměnné, pak $f(z; \theta)$ je náhodná proměnná z prostoru $\mathcal{X}$. Cílem je optimalizovat θ tak, aby bylo možné vzorkovat z z $P(z)$ a aby, s vysokou pravděpodobnostní, každý výstup $f(z; \theta)$ byl podobný příslušným X ze vstupního datasetu. (Doersch, 2021)

Tedy, chceme maximalizovat pravděpodobnost každého X z trénovací množiny skrze celý generativní proces dle:

$$P(X) = \int P(X \mid z; \theta) P(z) dz. \tag{1.7}$$

Kde $f(z; \theta)$ bylo nahrazeno distribucí $P(X \mid z; \theta)$, která umožňuje explicitně vyjádřit závislost X na z (ze zákonu celkové pravděpodobnosti). (Doersch, 2021)

1.7.1 Maximum likelihood

Intuitivní myšlenka za tímto principem, nazývaným *marginal likelihood*, je, že pokud model s vysokou pravděpodobností vyprodukuje množinu vzorků z trénovací množiny, tak je také **pravděpodobné**, že vyprodukuje **podobné vzorky** které nebyly součástí trénovací množiny (a naopak, je **nepravděpodobné** že vyprodukuje **odlišné vzorky vůči vzorkům v trénovací množině**). (Doersch, 2021)

U variačních autoenkodérů je toto výstupní rozdělení pravděpodobnosti často voleno jako Gaussovo, tedy například $P(X \mid z\theta) = \mathcal{N}(X \mid f(z; \theta), \sigma^2 * I)$ Tedy, má střední hodnotu $f(z; \theta)$ a kovarianci rovnu jednotkové matici I krát *nějaká* skalární hodnota σ. (Doersch, 2021)

Obecně (a zejména v rané fázi trénování) model nebude generovat výstupy které jsou *identické* k libovolnému X. V důsledku toho, že u variačního autoenkodéru volíme toto výstupní rozdělení jako Gaussovo, lze provést algoritmus gradientního sestupu (případně jinou optimalizační techniku) za účelem zvýšení $P(x)$ tím, že donutíme $f(z; \theta)$ blížit se k X pro nějaké

z (a tím pádem iterativně zvyšovat pravděpodobnost, že model vygeneruje vzorek podobný vzorku z trénovací množiny). (Doersch, 2021)

Důležitou vlastností pro možnost provést tuto optimalizaci tedy je, aby $P(X \mid z)$ bylo **spočitatelné a spojité v** θ [13]. (Doersch, 2021)

1.8 REWORK: Probabilistic Models and Variational Inference

1.9 (Probabilistic) Graphical models

1.10 Joint optimization

1.11 Latent Variable Models

1.12 Diskriminativní model

1.13 Generativní model

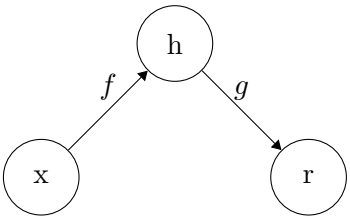
1.14 Advanced inference techniques

1.15 Intractabilities — To je v čem mají VAE výhodu!

[13] Výstupní rozdělení nemusí byt Gaussovo (ale např. lze využít Alternativní rozdělení)

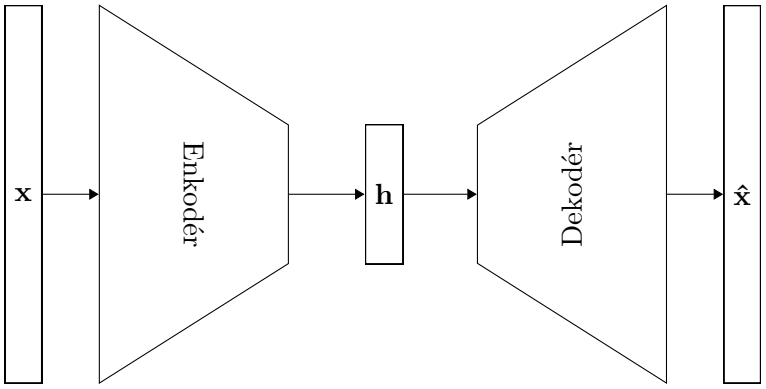
2. Autoenkodér

Autoenkodér je typ umělé neuronové sítě se schopností učit se efektivní reprezentace vstupních dat bez učitele [1]. Umělá neuronová síť autoenkodéru má symetrickou strukturu a skrytou vrstvu $\boldsymbol{h}$, která popisuje *kód* použitý pro reprezentaci vstupu. Architekturu autoenkodéru (viz Obrázek 2.1) lze principiálně rozdělit na dvě části – kódovací funkci $h = f(x)$, resp. **enkodér** a dekódovací funkci $r = g(h)$, resp. **dekodér**. Hovoříme tedy o typu umělé neuronové sítě s *enkodér-dekodér* moduly. Výstupem enkodéru je **kód** vstupu h. Výstupem dekodéru je **rekonstrukce** vstupu r. (I. Goodfellow et al., 2016)



Obrázek 2.1: Obecná struktura autoenkodéru. Ze vstupu x je enkodérem vytvořen kód h (funkce f). Tento kód je následně dekodérem přetaven na rekonstrukci r (funkce g).

Obecnou strukturu (viz Obrázek 2.1) lze reprezentovat dopřednou umělou neuronovou sítí. Jejím cílem je **rekonstruovat vstupní data na výstupní vrstvě** (tzv. *unsupervised learning objective*). Počet vstupů je tak totožný s počtem neuronů ve výstupní vrstvě umělé neuronové sítě (tedy x a r mají stejnou dimenzi). h může mít *menší* či *větší* dimenzi – volba dimenze h se odvíjí od požadovaných vlastností autoenkodéru. Obecnou architekturu modulů umělé neuronové sítě Autoenkodéru zachycuje Obrázek 2.2. (Charte et al., 2018)



Obrázek 2.2: Jednotlivé moduly architektury umělé neuronové sítě autoenkodéru.

[1] V literatuře se můžeme zřídka setkat s zařazením autoenkodérů **obecně** do třídy *semi-supervised* algoritmů strojového učení, například (Chollet, 2017, str. 95). S ohledem na formulaci autoenkodéru představenou v této kapitole je přesnější zařadit **obecnou** architekturu autoenkodérů do třídy algoritmů učení bez učitele. Nutno předeslat, že v architekturách hlubokých autoenkodérů je běžným jevem tzv. *stacked autoenkodér* (viz Podsekce 2.5.1). V instancích tohoto případu pak jednoznačně **lze** hovořit o zařazení autoenkodéru do třídy semi-supervizovaného učení. (Bengio, Lamblin et al., 2006), (Ranzato et al., 2007), (Erhan et al., 2010)

Autoenkodér je trénován k rekonstrukci jeho vstupů. Pokud by se autoenkodér naučil jednoduše určit $x = g(f(x))$ pro každé x, získali bychom *identitu*, která není patřičně užitečná. Proto je při trénování zavedena řada omezení, jejichž účelem je zabránit možnosti naučení autoenkodéru perfektně kopírovat vstupní data. (I. Goodfellow et al., 2016)

2.1 Historický pohled

Vícevrstvý Perceptron (viz Podsekce 1.2.5) je univerzálním aproximátorem – tedy historicky nalézá uplatnění zejména v klasifikačních úlohách učení s učitelem. Sofistikovaný algoritmus se schopností trénování Vícevrstvého Perceptronu s větším počtem skrytých vrstev stále schází, a to zejména v důsledku problému mizejícího gradientu[2] (Hochreiter, 1998). Až příchod algoritmu gradientního sestupu, který adresuje problém mizejícího gradientu v aplikacích konvolučních sítí (Y. LeCun et al., 1989) (a později i úloh učení bez učitele) značí počátek moderních metod hlubokého učení. V oblasti hlubokého učení tak přirozeně dochází k emergenci a vývoji řady technik pro řešení úloh učení bez učitele. V této kapitole je popsána pouze jedna z nich – architektura umělé neuronové sítě založené na *enkodér-dekodér* modulech: **autoenkodér**. Autoenkodéry byly poprvé představeny jako způsob pro předtrénování umělých neuronových sítí formou automatizované extrakce vlastností (viz Podsekce 1.3.2). Později Autoenkodéry nalézají uplatnění zejména v úlohách redukce dimenzionality (viz Sekce 1.3) či fůzi vlastností (*feature fusion*). (Charte et al., 2018)

Nedávné teoretické propojení autoenkodéru a modelů využívajících latentní proměnných (viz Sekce 1.5) však vedlo ke vzniku zcela nové architektury neuronové sítě kombinující charakter redukce dimenzionality autoenkodéru a pravděpodobnostní modely (viz Sekce 1.4), kterou nazýváme **variační autoenkodér**. To vyneslo autoenkodéry na popředí v oblasti generativního modelování. Představení variačního autoenkodéru je věnována Kapitola 3 a možnostem jeho využití celý zbytek práce.

Byť autoenkodéry vznikly v kontextu hlubokého učení, není pravidlem že všechny modely autoenkodéru obsahují vícero skrytých vrstev. Následuje rozdělení autoenkodérů, mimo jiné, dle struktury jejich umělé neuronové sítě.

2.2 Mělký autoenkodér

Mělký autoenkodér (*shallow autoencoder*), resp. jeho umělá neuronová síť, je sestaven pouze ze tří vrstev – vstupní, kód a výstupní. Je to **nejtriviálnější model autoenkodéru**, jelikož jeho skrytá vrstva (kód) je pouze **jednovrstvá**. (Charte et al., 2018)

[2] Vanishing gradient problem

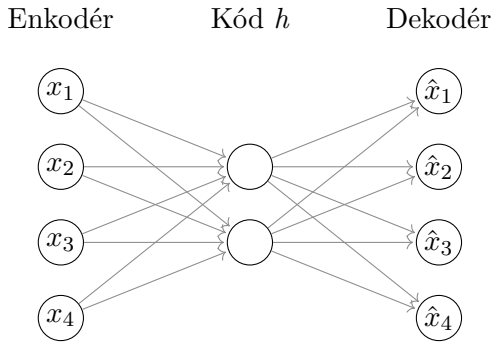
2.3 Autoenkodér s neúplnou skrytou vrstvou

Autoenkodér s neúplnou skrytou vrstvou (*undercomplete autoencoder*) je autoenkodér, jehož dimenze kódu (h) je menší, než dimenze vstupu. Tuto skrytou vrstvu h nazýváme **bottleneck**. Bottleneck je způsob, kterým se autoenkodér s neúplnou skrytou vrstvou učí **kompresované reprezentaci znalostí**. V důsledku bottleneck vrstvy je autoenkodér nucen zachytit pouze salientní vlastnosti trénovacích dat, které následně budou použity pro rekonstrukci. (I. Goodfellow et al., 2016)

Trénovací proces autoenkodéru s neúplnou skrytou vrstvou je popsán jako minimalizace ztrátové funkce:

$$\mathcal{L}(\mathbf{x}, g(f(\mathbf{x}))), \tag{2.1}$$

kde $\mathcal{L}$ je ztrátová funkce, penalizující $g(f(\mathbf{x}))$ za *rozdílnost* vůči $\mathbf{x}$ (např. *střední kvadratická chyba*). (Charte et al., 2018)



Obrázek 2.3: Jednoduchá architektura umělé neuronové sítě Autoenkodéru s neúplnou skrytou vrstvou. Skrytá vrstva představuje *bottleneck*.

Od Analýzy hlavních komponent po Autoenkodér

Máme-li linéární dekodér (Autoenkodér používá pouze linéárni aktivační funkce) a jako ztrátová funkce $\mathcal{L}$ je použita *střední kvadratická chyba*, pak se neuplný autoenkodér naučí stejný *vektorový prostor*, který by byl výsledkem Analýzy hlavních komponent (viz Podsekce 1.3.3). V tomto speciálním případě lze ukázat, že autoenkodér trénovaný na úloze kompresované reprezentace znalostí jako vedlejší efekt provedl Analýzu hlavních komponent. (Baldi a Hornik, 1989), (Kamyshanska a Memisevic, 2013)

Důležitým důsledkem tohoto jevu je, že **Autoenkodéry** s nelineární kódovací funkcí f a nelineární dekódovací funkcní g **jsou schopny učit se obecnější generalizaci** než u Analýzy hlavních komponent. (I. Goodfellow et al., 2016)

Na druhou stranu, má-li autoenkodér k dispozici příliš kapacity, může se naučit pouze kopírovat vstup na výstupní vrstvě bez extrakce salientních vlastností (tedy identické zobrazení).

Problém s naučením pouhého identického zobrazení

Extrémním případem je teoretický scénář, ve kterém je Autoenkodér složen z kódu (h) o jedné vrstvě a velmi výkonného enkodéru. Takový Autoenkodér by se mohl naučit reprezentovat každý vstup x_i kódem i. Dekodér by se pak tyto indexy mohl naučit mapovat zpátky na hodnoty konkrétních trénovacích vzorků dat. Tento příklad se v praxi běžně nenaskytne, nicméne jasně ilustruje, jak může Autoenkodér při úloze kopírování vstupu na výstupní vrstvu selhat naučit se užitečné vlastnosti o vstupních datech, jsou-li restrikce při učení příliš nízké. (I. Goodfellow et al., 2016)

Proto je nutné **autoenkodéry regularizovat** (viz Podsekce 1.1.5).

V dalších sekcích (Sekce 2.6 - Sekce 2.9) jsou tedy představeny přístupy k architekturám Autoenkodérů **s využitím regularizace**.

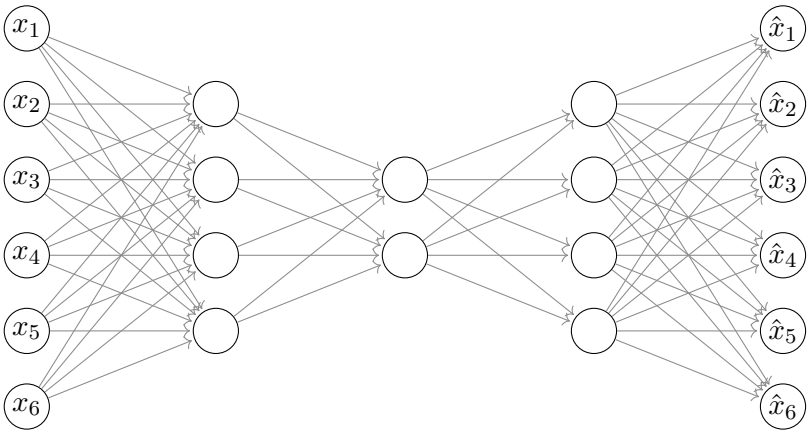
2.4 Autoenkodér s rozšířenou skrytou vrstvou

Autoenkodér s rozšířenou skrytou vrstvou (*overcomplete autoencoder*) je Autoenkodér, jehož počet neuronů ve skrýté vrstvě je větší než počet neuronů vstupní (a výstupní) vrstvy. (Charte et al., 2018)

2.5 Hluboký autoenkodér

V předchozích sekcích (Sekce 2.3, Sekce 2.4) byly představeny autonkodéry s jednovrstvým enkodérem a s jednovrstvým dekodérem. Existují však i autoenkodéry, jejichž enkodér a dekodér moduly jsou vícevrstvé, tedy mají hloubku vyšší než jedna.

Hluboký autoenkodér (*deep* autoenkodér), je autoenkodér s netriviální hlobkou skryté vrstvy (*kódu*) $\boldsymbol{h}$. (Geron, 2019)



Obrázek 2.4: Schéma umělé neuronové sítě hlubokého autoenkodéru s neúplnou skrytou vrstvou.

Z netriviální hloubky dopředné umělé neuronové sítě plyne Podsekce 1.2.6, který garantuje, že dopředná umělá neuronová síť s alespoň jednou skrytou vrstvou dokáže aproximovat (*libovolně přesně*) jakoukoliv funkci (za předpokladu dostatečného počtu neuron skryté vrstvy). (I. Goodfellow et al., 2016)

Nicméně u mělkých enkodérů, které jsou rovněž dopřednou sítí, neexistuje možnost představit libovolná omezení a regularizační prvky (například řídkost *kódu* $\boldsymbol{h}$ – viz Sekce 2.6). A tedy nelze zamezit problému naučení pouhé identity (viz Sekce 2.3). Na rozdíl od mělkých autoenkoderů (viz Sekce 2.2), však mohou hluboké autoenkodéry (*libovolně přesně*) aproximovat jakékoliv mapování vstupu na kód (*opět s předpokladem dostatečného počtu neuronů skryté vrstvy*). (I. Goodfellow et al., 2016), (Charte et al., 2018)

S netriviální hloubkou se rovněž pojí významná redukce výpočetních nákladů spojených s reprezentací některých funkcí. V důsledku možnosti efektivnější reprezentace takových funkcí dochází i k zmenšení nároků na velikost množiny trénovacích dat.

2.5.1 Stacked autoenkodér

Běžnou strategií pro trénování Hlubokého autoenkodéru je hladově předtrénovat (*greedy pretraining*) model individuálním natrénováním většího počtu Mělkých autoenkodérů (předsta-

vených v sekci Sekce 2.2), které jsou následně vloženy za sebe. Takto složený Autoenkodér nazýváme Stacked autoenkodér. (Geron, 2019)

2.6 Řídký autoenkodér

Řídká reprezentace dat (*sparsity*) ve strojovém učení znamená, že většina hodnot daného vzorku je nulová. Motivací pro řídkou reprezentaci dat ve strojovém učení je napodobení chování buněk v primární zrakové oblasti (*V1*) mozku savců. Konkrétně schopnosti odhalit a uložit efektivní kódovací strategie pozorovaných vjemů. (Olshausen a Field, 1997)

Pro sestrojení Řídkého autoenkodérů je tedy nutné představit omezení (regularizační prvek) hodnot aktivací neuronů ve skryté (kódovací) vrstvě $\boldsymbol{h}$ (resp. počtu aktivních neuronů ve skrýte vrstvě).

Řídký autoenkodér (*sparse autoencoder*) je autoenkodér, jehož ztrátová funkce je rozšířena o penalizaci řídkosti kódovací vrstvy $\boldsymbol{h}$ (tzv. *sparsity penalty*) vztahem $\Omega(\boldsymbol{h})$:

$$\mathcal{L}(\mathbf{x}, g(f(\mathbf{x}))) + \Omega(\boldsymbol{h}), \tag{2.2}$$

kde Ω je **regularizační prvek**, jehož cílem je přiblížit hodnoty aktivací neuronů kódovací vrstvy k cílové hodnotě (a zabránit přeučení). Chceme tak penalizovat neurony kódovací vrstvy, které se aktivují příliš často. (I. Goodfellow et al., 2016)

Běžně lze Ω stanovit následovně. Mějme Bernoulliho náhodnou proměnou i modelující aktivace neuronů skrýte (kódovací) vrstvy – můžou tedy nastat dva stavy: neuron skryté vrstvy je buď aktivován, nebo není aktivován. Pro konkrétní vstup x dostaneme:

$$\hat{p_i} = \frac{1}{|S|} \sum_{x \in S} f_i(x), \tag{2.3}$$

kde $f = (f_1, f_2, \ldots, f_c)$, c je počet neuronů skryté (kódovací) vrstvy a $\hat{p_i}$ je průměrná aktivační hodnota neuronu skryté vrstvy (resp. střední hodnota příslušného Bernoulliho schématu). (Charte et al., 2018)

Dále mějme p jako cílové rozdělení aktivací. Kullback-Leibnerova divergence mezi náhodnou proměnnou i a p pak udává rozdíl obou rozdělení:

$$KL(p \parallel \hat{p_i}) = p \log \frac{p}{\hat{p_i}} + (1-p) \log \frac{1-p}{1-\hat{p_i}}. \tag{2.4}$$

Výsledný penalizační prvek Ω pro Řídký autoenkodér má tedy následující podobu:

$$\Omega_{\check{\mathrm{R}}AE}(W, b; S) = \sum_{i=1}^{c} KL(p \parallel \hat{p_i}), \tag{2.5}$$

kde průměrná hodnota aktivací $\hat{p}_i$ závisí na parametrech enkodéru a množině trénovacích dat S. (Charte et al., 2018)

Přičtením tohoto penalizačního prvku ke ztrátové funkci (a následnou minimalizací celkové ztrátové funkce) je autoenkodér nucen **omezit počet aktivních neuronů v skrýté (kódovací) vrstvě**. V důsledku tohoto omezení pak každý neuron skryté vrstvy reprezentuje nějakou **salientní vlastnost** vstupních dat (a rovněž je zamezeno naučení pouhé identity, viz Sekce 2.3). (I. Goodfellow et al., 2016)

2.7 Denoising autoenkodér

Denoising autoenkodér je autoenkodér, který na vstupu obdrží poškozená vstupní data a při trénování je jeho předpovědět originální vstup bez poškození, a ten na výstupní vrstvě vrátit.

Autoenkodéry běžně minimalizují funkci ve tvaru:

$$\mathcal{L}(\mathbf{x}, g(f(\mathbf{x}))), \tag{2.6}$$

kde L je ztrátová funkce penalizující $g(f(x))$ za odlišnost od $\mathbf{x}$ (např. L^2 norma jejich rozdílů). Jak ale bylo ukázáno v Sekce 2.3, to umožňuje $g \circ f$ naučit se být pouhou identitu.

Z toho důvodu denoising autoenkodér minimalizuje funkci:

$$\mathcal{L}(\boldsymbol{x}, g(f(\tilde{\boldsymbol{x}}))), \tag{2.7}$$

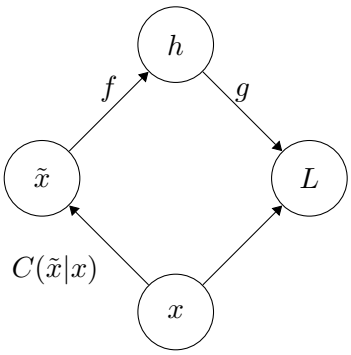
kde $\tilde{\boldsymbol{x}}$ je kopií $\boldsymbol{x}$ která byla úmyslně poškozena procesem $C(\tilde{x}|x)$ (*corruption process*), který reprezentuje podmíněné rozdělení pravděpodobnosti poškozených vzorků $\tilde{x}$ v závislosti na vzorku vstupních dat x. (I. Goodfellow et al., 2016)

Denoising autoenkodér se pak učí **rozdělení rekonstrukce** $p_{reconstruct}(\mathbf{x}|\tilde{\mathbf{x}})$, které je odhadnuto z trénovacích dvojic následovně:

1. Zvolit trénovací vzorek $\boldsymbol{x}$ z množiny trénovacích dat
2. Vygenerovat poškozenou verzi zvoleného vzorku ($\tilde{\boldsymbol{x}}$) procesem C
3. Použít dvojice ($\boldsymbol{x}$, $\tilde{\boldsymbol{x}}$) jako množinu trénovacích dat pro odhadnutí rozdělení rekonstrukce Denoising autoenkodéru $p_{reconstruct}(\mathbf{x}|\tilde{\mathbf{x}}) = p_{decoder}(\boldsymbol{x}|\boldsymbol{h})$, kde $p_{decoder}$ je výstupem funkce dekodéru $g(\boldsymbol{h})$

Při trénování Denoising autoenkodéru jsou funkce f a g nuceny zachytit implicitní strukturu $p_{data}(\boldsymbol{x})$. (I. Goodfellow et al., 2016)

Trénovací proceduru denoising autoenkodéru schématicky zachycuje Obrázek 2.5:



Obrázek 2.5: Procedura trénování denoising autoenkodéru. Převzato z (I. Goodfellow et al., 2016).

Denoising autoenkodér se tedy musí naučit toto poškození odstranit a rekonstruovat tak původní vstup (namísto pouhého naučení se identitě).

Denoising Autoenkodéry jsou příkladem hned dvou jevů (Charte et al., 2018):

- Emergence užitečných vlastností o vstupních datech jako výsledek minimalizace chyby rekonstrukce
- Schopnosti modelů s vysokou kapacitou/rozšířenou skrytou vrstvou fungovat jako Autoenkodér, **za předpokladu že je jim zabráněno naučit se identické zobrazení vstupních dat**

2.8 Robustní autoenkodér

Robustní autoenkodér (*robust autoencoder*) představuje další způsob, jakým se lze vypořádat s *drobným šumem* ve vstupních datech, která má následně rekonstruovat.

Robustní autoenkodéry, narozdíl od denoising autoenkodéru (viz Sekce 2.7), využivájí alternativně definovanou ztřatovou funkci, která je modifikována pro minimalizaci chyby rekonstrukce. Obecně jsou ne-gaussovským šumem ovlivněny **méně než triviální mělké autoenkodéry** (viz Sekce 2.2). Tato alternativní ztrátová funkce je založena na correntropii (*correntropy*, lokalizovaná míra podobnosti), která je v (W. Liu et al., 2006) definována následovně:

$$\mathcal{L}_{MCC}(u, v) = -\sum_{k=1}^{d} \mathcal{K}_\sigma(u_k - v_k), \tag{2.8}$$

kde

$$\mathcal{K}_\sigma(\alpha) = \frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{\alpha^2}{2\sigma^2}), \tag{2.9}$$

a σ je parameter kernelu $\mathcal{K}$.

Correntropie měří hustotu pravdepodobnostni, že dvě *události* jsou si rovny. Correntropie je **výrazně méně ovlivněna odlehlými hodnotami**, než např. střední kvadratická chyba. (W. Liu et al., 2006) Robustní autoenkodér se snaží tuto míru maximalizovat, což intuitivně vede k **vyšší odolnosti** Robustního autoenkodéru **na ne-gaussovský šum** přítomný ve vstupních datech. (Charte et al., 2018)

2.9 Contractive autoenkodér

Přehnaná citlivost na *drobné rozdíly* ve vstupních datech by mohla vést k architektuře Autoenkodéru, která pro velmi podobné vstupy generuje odlišné kódy.

Contractive autoenkodér (*CAE*), je Autoenkodér, který je při trénování omezen regularizačním prvkem, který vynucuje aby derivace kódů ve vztahu k jejich vstupu byly co možná nejmenší. Tedy **dva *podobné* vstupy musí mít vzájemně *podobné* kódy**. Přesněji je dosaženo lokální invariance na přípustně malé změny vstupních dat. (Rifai et al., 2012)

Citlivost na *drobné rozdíly* ve vstupních datech lze měřit pomocí Frobeniovy maticové normy $\|\cdot\|_F$ Jacobiho matice enkodéru (J_f):

$$\|J_f(x)\|_F^2 = \sum_{j=1}^{d} \sum_{i=1}^{c} \left(\frac{\partial f_i}{\partial x_j}(x) \right)^2. \tag{2.10}$$

Čím vyšší je tato hodnota, tím více bude kód nestabilní s ohledem na *drobné rozdíly* ve vstupních datech. Z této metriky je následně sestaven **regularizační prvek** který je připočten k hodnotě ztrátové funkce Contractive Autoenkodéru:

$$\Omega_{CAE}(W, b, S) = \sum_{x \in S} \|J_f(x)\|_F^2. \tag{2.11}$$

Výsledkem je tedy Autoenkodér, jehož dva (lokálně) *podobné* vstupy musejí mít i *podobný* kód. (Charte et al., 2018)

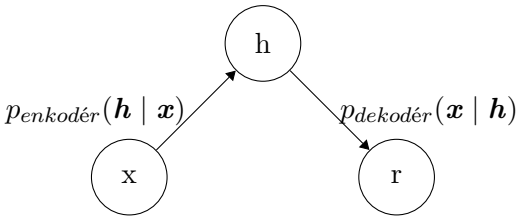
Z contractive autoenkodéru lze rovněž vzorkovat nové výstupy. Jakobián (*Jacobiho determinant*) enkodéru je (jako *drobný šum*) přičten ke kódu vstupu. Takto modifikovaný kód je poté dekodérem přetaven na výstup a dostáváme nový vzorek dat. (I. Goodfellow et al., 2016)

2.10 Stochastický autoenkodér

Struktura stochastického autoenkodéru se vůči obecné struktuře autoenkodéru (Obrázek 2.1) liší reprezentací enkodér a dekodér modulů.

V stochastickém autoenkodéru jsou enkodér a dekodér moduly reprezentovány rozdělením pravděpodobností (nejedná se tedy pouze o funkce, ale moduly zahrnují i určitou míru šumu). Výstup těchto modulů tedy obdržíme **výběrem z příslušného rozdělení pravděpodobnosti**. (I. Goodfellow et al., 2016)

Mějme skrytou vrstvu (*kód*) $\boldsymbol{h}$. Obecně má pro enkodér toto rozdělení podobu $p_{enkodér}(\boldsymbol{h} \mid \boldsymbol{x})$ a pro dekodér $p_{dekodér}(\boldsymbol{x} \mid \boldsymbol{h})$. Modifikací základní struktury autoenkodéru (Obrázek 2.1) tedy dostáváme:



Obrázek 2.6: Obecná struktura Autoenkodéru. Ze vstupu x je enkodérem vytvořen kód h (funkce f). Tento kód je následně dekodérem přetaven na rekonstrukci r (funkce g).

V tradiční dopředné umělé neuronové síti Podsekce 1.2.3 je běžnou strategií pro návrh výstupní vrstvy definování (*výstupního*) rozdělení pravděpodobnosti $p(\boldsymbol{y} \mid \boldsymbol{x})$. Pro návrh ztrátové funkce pak minimalizace záporného logaritmu věrohodnosti $-\log p(\boldsymbol{y} \mid \boldsymbol{x})$. V takové architektuře je $\boldsymbol{x}$ vektor vstupních dat a $\boldsymbol{y}$ vektor cílových proměnných, které se umělá neuronová síť snaží předpovědět (např. štítků jednotlivých tříd). (I. Goodfellow et al., 2016)

S architekturou autoenkodérů se však pojí jeden zásadní rozdíl. V autoenkodéru **je $\boldsymbol{x}$ jak vstupní, tak cílová proměnná**.

Dekodér pak lze interpretovat jako modul poskytující podmíněné rozdělení $p_{dekodér}(\boldsymbol{x} \mid \boldsymbol{h})$. Autoenkodér lze trénovat minimalizací $-\log p_{dekodér}(\boldsymbol{x} \mid \boldsymbol{h})$. Konkrétní podoba ztrátové funkce se odvíjí od požadovaných vlastností autoenkodéru a od přesné podoby modulu dekodéru (pokud hodnoty $\boldsymbol{x} \in \mathbb{R}$, pak jsou pro parametrizaci normálního rozdělení použity linéarní výstupní jednotky, tedy $-\log p_{dekodér}(\boldsymbol{x} \mid \boldsymbol{h})$ vrací *střední kvadtratickou chybu*).

Stochastický autoenkodér tedy **generalizuje kódovací funkci** $f(x)$ na **kódovací rozdělení pravděpodobnosti** $p_{enkodér}(\boldsymbol{h} \mid \boldsymbol{x})$.

Enkodér a dekodér moduly stochastického autoenkodéru tedy lze považovat za **modely využívající latentní proměnné** (*latent variable models*, viz Sekce 1.5). Model využívající latentní proměnné značíme $p_{model}(\boldsymbol{h}, \boldsymbol{x})$. (I. Goodfellow et al., 2016)

Stochastický enkodér je pak definován následovně (I. Goodfellow et al., 2016):

$$p_{enkodér}(\boldsymbol{h} \mid \boldsymbol{x}) = p_{model}(\boldsymbol{h}, \boldsymbol{x}) \tag{2.12}$$

a **stochastický dekodér** následovně (I. Goodfellow et al., 2016):

$$p_{dekodér}(\boldsymbol{x} \mid \boldsymbol{h}) = p_{model}(\boldsymbol{x}, \boldsymbol{h}) \tag{2.13}$$

Výběr nových dat z rozdělení pravděpodobnosti autoenkodéru je detailněji představuje Kapitola 3.

2.11 Ostatní typy autoenkoderů

Na poli výzkumu strojového se autoenkodéry těší velkému úspěchu. Přirozeně tak existuje celá řada architektur a modifikací, které slouží k specifickým účelům. V této kapitole byly představeny pouze ty architektury autoenkodéru, na kterých bude později stavět kapitola Kapitola 4.

V této sekci jsou stručně shrnuty *ostatní* typy autoenkodérů, které nejsou pro předmět nadcházejících kapitol stěžejní, ale obecně se jim dostává vysoké míry využití.

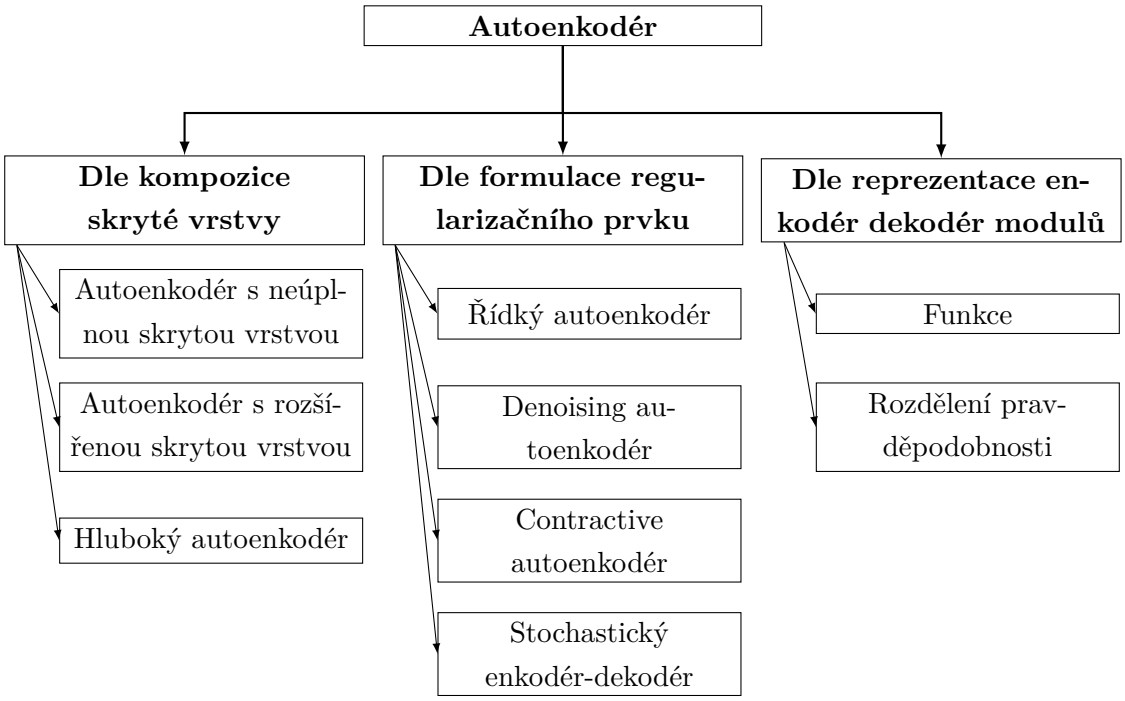
2.11.1 Adversariální autoenkodér

Adversariální autoenkodér (*adversarial autoencoder*) přináší koncept Generativních Adversariálních sítí (I. J. Goodfellow et al., 2014) na pole autoenkodérů. V adversariálním autoenkodéru je kód ($\boldsymbol{h}$) modelován uložením předchozího rozdělení pravděpodobnosti (*prior*). Následně je natrénován *běžný* autoenkodér, současně se *diskriminační* síť snaží odlišit kódy modelu od výběrů dat z předchozího rozdělení pravděpodobnosti. Jelikož generátor (v tomto případě enkodér) je trénován k *přelstění* diskriminátoru, kódy mají tendenci následovat uložené rozdělení pravděpodobnosti. Tím pádem, z adversariálního autoenkodéru lze **provádět výběr zcela nových dat** (*generovat nové vzorky*). (Charte et al., 2018)

2.12 Taxonomie autoenkodérů

Bylo představeno několik tříd autoenkodérů (rozdělení dle navržení umělé neuronové sítě, rozdělení dle regularizačního prvku) a následně popsáno několik dalších druhů autoenkodérů.

Obrázek 2.7 nabízí jejich (nevyčerpávající) taxonomii. Autoenkodéry jsou zde rozděleny podle představených charakteristik jejich reprezentace a zpracování skryté vrstvy.



Obrázek 2.7: Autoenkodéry rozděleny dle charakteristik zpracování kódovací vrstvy

2.13 Využití Autoenkodéru

Představené architektury autoenkodérů nabízejí řadu využití, mezi které se řadí:

- Mapování vysokorozměrných dat do 2D pro vizualizaci
- Učení se abstraktních vlastností o vstupních datech bez učitele pro následné využití v úlohách učení s učitelem
- Komprese dat

nicméně naráží na zásadní problém nespojitosti naučených manifoldů – tedy (s výjimkou stochastického přístupu, viz Sekce 2.10) nemají příliš dobrý výkon v generativních úlohách. Aplikacím úloh pozorování v latentním prostoru je věnována celá Kapitola 4. Pro jejich realizaci je však nutné představit architekturu **variačního autoenkodéru** (viz Kapitola 3), která staví na dosud popsaných konceptech.

3. Variační autoenkodér

Generativní modely (Sekce 1.13), učení se reprezentací (Bengio, Courville et al., 2014) a úlohy učení bez učitele (Podsekce 1.1.3) jsou **klíčové oblastí pro vytvoření inteligentních systémů** (Diederik P. Kingma a Max Welling, 2019), (Yann LeCun, 2022). Variační autoenkodér z těchto principů vychází a propojuje je. Ve snaze o konstrukci takového stroje tak variační autoenkodér hraje důležitou roli.

Variační autoenkodér (D. P. Kingma a M. Welling, 2014), (Rezende et al., 2014) (dále jen *VAE*) je rámec poskytující metodu pro učení víceúčelových hlubokých modelů využívajících latentních proměnných (Sekce 1.5) a příslušných odvozovacích modelů za použití stochastického gradientního sestupu. (Diederik P. Kingma a Max Welling, 2019)

VAE nalézá širokou škálu aplikací v generativním modelování, učení se reprezentací a úlohách učení se bez učitele (resp. semi-supervizovaných úlohách).

Variační autoenkodér jako generativní model a motivace vzniku

Jednou z hlavních divizí v odvětví strojového učení je generativní versus diskriminativní modelování. V diskriminativním modelování je cílem naučit se prediktor na základě pozorování. V generativním modelování je cíl poněkud obecnější – naučit se spojité rozdělení pravděpodobnosti skrze všechny proměnné.

Generativní model simuluje způsob, kterým jsou data generovány v reálném světě. *Modelováním* se ve vědních disciplínách rozumí odhalování generujícího procesu stanovením hypotéz a následném testování těchto hypotéz pozorováním [1]. Generativní modelování má mnoho výhodných vlastností.

První z nich je možnost zabudování předem známých fyzikálních zákonů a omezení do samotného generativního procesu – zatímco neznáme (či nepodstatné) *detaily* můžeme zanedbat formou *šumu* (např. nežádoucí proměnná). Výsledný model je často vysoce intuitivní a dobře interpretovatelný a testováním jej vůči pozorováním lze vyvodit závěry naších teorií o fungování reálného světa.

Dalším důvodem pro snahu o pochopení generativního procesu dat je, že přirozeně zahrnují kauzální vztahy reálného světa. Pozorování a využití těchto kauzálních vztahů má velkou výhodu v schopnosti generalizovat v dosud nepozorovaných situacích, než pouhé korelace [2].

[1] For instance, when meteorologists model the weather they use highly complex partial differential equations to express the underlying physics of the weather. Or when an astronomer models the formation of galaxies s/he encodes in his/her equations of motion the physical laws under which stellar bodies interact. The same is true for biologists, chemists, economists and so on. Modeling in the sciences is in fact almost always generative modeling.

[2] Například, rozumíme-li generativnímu procesu zemětřesení, můžeme tuto znalost využít v Kalifornii i Čile

V diskriminativních metodách se učíme přímé mapování ve stejném směru, v jakém chceme predikovat. Což je oproti generativním modelům opačný směr. Například, lze říci, že obrázek je v reálném světe generován zachycením nějakého objektu, následného vygenerování tohoto objektu ve 3D a poté projekcí na mřížku pixelů. Diskriminativní model s těmito pixelovými hodnotami pracuje přímo jako se vstupem a predikuje jejich cílové štítky.

Zatímco se generativní modely zvládnou učit efektivní reprezentace vstupních dat, mají oproti diskriminativním modelům tendenci činit **silnější předpoklady**, což vede k **vyššímu asymptotickému biasu** pakliže se model **plete**. (Banerjee, 2007) Z tohoto důvodu, plete-li se náš model (a každý model se *témeř vždy* do určité míry plete), a zajimá-li nás čistě naučení se rozlišovat třídy, pak diskriminativní modely v takové úloze (za předpokladu dostatečného množství dat) často vedou k menší chybovosti.

V každém případě se vyplatí studovat proces generování dat jako způsob, kterým lze trénovací proces diskriminátoru (např. klasifikátoru) zušlechťovat. Například, můžeme mít k dispozici množinu vzorků se štítky a řádově větší množinu vzorků bez štítků. V takové úloze semi-supervizovaného učení lze využít generativního modelu dat k zpřesnění klasifikace. (D. P. Kingma a M. Welling, 2014), (Sønderby et al., 2016)

Generativní modelování může být využito i více obecně. Nad generativním modelováním lze uvažovat jako nad doprovodnou činností. Například, predikce bezprostřední budoucnosti nám může pomoct při sestavování užitečných abstrakcí o chování světa, které mohou být následně použity pro řadu dalších úloh predikce. Hledání rozmotaných (*disentangled*), sémanticky významných a statisticky nezávislých kauzálních faktorů variací dat je obecně známo pod pojmem učení se reprezentací bez učitele (*unsupervised representation learning*). A **variační autoenkodéry** jsou pro tento účel hojně uplatňovány.

Na tuto úlohu lze alternativně hledět i jako na implicitní formu regularizace. Nutíme-li vzniklé reprezentace být významnými pro proces generování dat, představujeme bias vůči opačnému procesu (který vstupní data mapuje na neužitečné reprezentace). Doprovodnou činnost predikce bezprostřední budoucnosti tak lze využít k lepšímu porozumění světa v abstraktní úrovni a tím pádem činit přesnější predikce v pozdější fázi.

Variační autoenkodér lze popsat jako dva provázané, byť **nezávisle parametrizované** modely: **enkodér** (resp. rozpoznávací) model a **dekodér** (resp. generativní) model. Tyto dva modely se vzájemně *podporují*. Enkodér model generativnímu modelu doručuje aproximaci jeho posteriorních náhodných latentních proměnných, které jsou potřebné pro úpravu jeho parametrů uvnitř iterace *expectation maximalization* učení. A opačně, generativní model slouží jako *opora* pro naučení významných reprezentací vstupních dat enkodér modelem. Tedy, dle Bayesova pravidla, enkodér je *approximate inverse* generativního modelu.

Jednou z výhod VAE oproti *běžné* variační inferenci (VI) je, že enkodér model (také nazývaný inferenční model) je nyní stochastickou funkcí vstupních proměnných. Narozdíl od VI, kde má každý data-case vlastní variační rozdělení pravděpodobnosti, což je při větším množství dat neefektivní. Enkodér používá jednu množinu parametrů pro model vztahů mezi vstupními

daty a latentními proměnnými. Tento proces nazýváme amortizovanou inferencí. Takový enkodér model může být libovolně komplexní, ale stále zůstává přiměřeně rychlým, jelikož již z principu může být realizován jedním dopředným průchodem z vstupu skrze latentní proměnné. Nevýhodou ale je, že při vzorkování vzniká v gradientech potřebných pro učení tzv. vzorkovací šum. Možná největším přínosem VAE je řešení tohoto šumu použitím tzv. *reparametrizačního triku* – jednoduchou procedurou pro přeorganizování výpočtu gradientů, který snižuje variaci gradientu.

Variační autoenkodér je inspirován Helmholtz Machine (Dayan et al., 1995), což byl první model který využíval enkodér modelu. Nicméně wake-sleep algoritmus, který byl v návrhu využitý, byl silně neefektivní a neoptimalizoval jednoznačné kritérium. U VAE tedy pravidla pro učení následují jednoznačnou aproximaci *maximum likelihood* cíle.

Variační autoenkodéry jsou spojením pravděpodobnostních grafických modelů a hlubokého učení. Generativní model je tvořen bayesovskou sítí ve tvaru $p(\mathbf{x} \mid \mathbf{z})p(\mathbf{z})$. Případně, má-li generativní model víc stochastických latentní vrstev, má síť následující hiearchii: $p(\mathbf{x} \mid \mathbf{z}_L)p(\mathbf{z}_L \mid \mathbf{z_{L-1}}) \dots p(\mathbf{z}_1 \mid \mathbf{z}_0)$. Podobně, enkodér model je také tvořen podmíněnou bayesovskou sítí ve tvaru $q(\mathbf{z} \mid \mathbf{x})$ nebo jako hiearchie, např.: $q(\mathbf{z}_0 \mid \mathbf{z}_1) \dots q(\mathbf{z}_L \mid X)$. Ale každá podmíněná vrstva může být tvořena komplexní (hlubokou) umělou neuronovou sítí, například: $\mathbf{z} \mid \mathbf{x} \sim f(\mathbf{x}, \epsilon)$, kde f je mapování umělé neuronové sítě a ϵ je *šum* (náhodná proměnná). Učící algoritmus VAE je mix klasického (amortizovaného variačního) *expectation maximization*. Ten ale skrze reparametrizační trik provede zpětný průchod skrze všechny vrstvy hluboké umělé neuronové sítě.

Vztah k generativní adversariální síti

3.1 Generování nových vzorků

Pro vygenerování nového vzorku $\boldsymbol{x}$ z modelu VAE je nejprve nutné získat vzorek $\boldsymbol{z}$ z rozdělení pravděpodobnosti kódu $p_{model}(\boldsymbol{z})$. Poté je tento vzorek vstupem pro síť generátoru $g(\boldsymbol{z})$. Na konec je $\boldsymbol{x}$ vzorkováno z rozdělení pravděpodobnosti $p_{model}(\boldsymbol{x}; g(\boldsymbol{z})) = p_{model}(\boldsymbol{x} \mid \boldsymbol{z})$. (D. P. Kingma a M. Welling, 2014)

Při trénování je však pro získání $\boldsymbol{z}$ použit enkodér (*approximate inference network*) $q(\boldsymbol{z} \mid \boldsymbol{x})$ a na $p_{model}(\boldsymbol{x} \mid \boldsymbol{z})$ pak lze pohlížet jako na dekodér. (D. P. Kingma a M. Welling, 2014)

Klíčovým principem VAE je, že může být trénován maximalizací variační dolní meze (*variational lower bound*) $\mathcal{L}(q)$ asociovanou s datovým bodem $\boldsymbol{x}$ následovně (I. Goodfellow et al., 2016):

$$\mathcal{L}(q) = \overbrace{\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})} \log p_{model}(\boldsymbol{z}, \boldsymbol{x})}^{\text{joint log-likelihood}} + \overbrace{\mathcal{H}(q(\mathbf{z} \mid \boldsymbol{x}))}^{\text{entropy}} \tag{3.1}$$

$$= \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})} \log p_{model}(\boldsymbol{x} \mid \boldsymbol{z}) + D_{KL}(q(\mathbf{z} \mid \boldsymbol{x}) \parallel p_{model}(\mathbf{z}))) \tag{3.2}$$

$$\leq \log p_{model}(\boldsymbol{x}) \tag{3.3}$$

První člen Rovnice 3.1 ukazuje *joint log-likelihood* viditelných a skrytých proměnných pod *approximate posterior* skrze latentní proměnné (stejně jako u EM, s rozdílem že používáme *approximate* místo *exact* posterior).

Druhý člen Rovnice 3.1 ukazuje entropii *approximate posterior*. Je-li q Gaussova distribuce, a je-li k predikované střední hodnotě přičten šum, maximalizace této entropie vede k zvyšující se směrodatné odchylce tohoto šumu. Obecně, prvek této entropie zajistí tendenci *variational posterior* přidělit vysokou hodnotu funkce pravdepodobnostni hodnotám $\boldsymbol{z}$ které by mohly generovat $\boldsymbol{x}$ (raději než konvergovat pouze k odhadu jedné hodnoty s *největší pravděpodobností*).

První člen Rovnice 3.2 ukazuje *log-likelihood* rekonstrukce, který lze nalézt i v ostatních typech autoenkodérů (Kapitola 2).

Druhý člen Rovnice 3.2 zajišťuje, aby se *approximate posterior distribution* $q(\mathbf{z} \mid \boldsymbol{x})$ a model prior $p_{model}(\boldsymbol{z})$ k sobě blížily. (I. Goodfellow et al., 2016)

Tradiční přístupy k variačnímu odvozování a učení odvozují q pomocí optimalizačního algoritmu (typicky iterují skrze soustavy *fixed-point* rovnic). Takové přístupy jsou pomalé a výpočetně neefektivní, jelikož často vyžadují schopnost vypočítat $\mathbb{E}_{z\sim q}$ v uzavřeném tvaru.

Hlavní myšlenkou variačních autoenkodérů je natrénování parametrického autoenkodéru (také nazývaného jako *odvozovací síť* či *rozpoznávací model*) **který produkuje parametry q**. Je-li $\boldsymbol{z}$ spojitá proměnná, lze vždy provést algoritmus zpětné propagace skrze vzorky $\boldsymbol{z}$ získané z $q(\boldsymbol{z} \mid \boldsymbol{x}) = q(\boldsymbol{z}; f(\boldsymbol{x}; \boldsymbol{\theta}))$ za účelem spočtení gradientu s respektem k $\boldsymbol{\theta}$. (I. Goodfellow et al., 2016)

Učení VAE pak spočívá čistě v maximalizaci $\mathcal{L}$ s ohledem na parametry enkodéru a dekodéru. (D. P. Kingma a M. Welling, 2014)

3.2 Enkodér modul

V Sekce 1.5 byly představeny hluboké modely využívající latentních proměnných (DLVM). U těchto modelů je problém provést odhad log-likelihood a rozdělení posteriorní pravděpodobnosti.

Rámec VAE poskytuje výpočetně efektivní způsob, kterým lze DLVM společně optimalizovat

s jejich korespondujícími inferenčními modely pomocí stochastického gradientního sestupu [3].

Enkodér modul VAE je pravděpodobnostní enkodér $q_\phi(\mathbf{z} \mid \mathbf{x})$, který na základě datového bodu $\mathbf{x}$ produkuje rozdělení pravděpodobnosti (typický Gaussovo) skrze vŠechny možné hodnoty kódu $\mathbf{z}$ z kterého tento datový bod teoreticky mohl být vygenerován.

Posteriorní inference a úlohy učení DLVM jsou efektivně neřešitelné problémy. VAE **představuje parametrický inferenční model** *parametric inference model* $q_\phi(\mathbf{z} \mid \mathbf{x})$ pro přetavení těchto problémů na efektivně řešitelné. Tento model také nazýváme **enkodér** či **rozpoznávací model**. ϕ označuje parametry tohoto inferenčního modelu, tyto parametry nazýváme **variační parametry**. (Diederik P. Kingma a Max Welling, 2019)

Variační parametry ϕ optimalizujeme tak, aby platila aproximace (D. P. Kingma a M. Welling, 2014):

$$q_\phi(\mathbf{z} \mid \mathbf{x}) \approx p_\theta(\mathbf{z} \mid \mathbf{x}) \tag{3.4}$$

Tato aproximace posteriorního rozdělení pomáhá k optimalizaci *marginal likelihood*.

Stejně jako u DLVM, inferenční model (enkodér) může být (téměř) libovolný orientovaný pravděpodobnostní grafický model (Diederik P. Kingma a Max Welling, 2019):

$$q_\phi(\mathbf{z} \mid \mathbf{x}) = q_\phi(\mathbf{z}_1, \ldots, \mathbf{z}_M \mid \mathbf{x}) = \prod_{j=1}^{M} q_\phi(\mathbf{z}_j \mid Pa(\mathbf{z}_j), \mathbf{x}) \tag{3.5}$$

kde $Pa(\mathbf{z}_j)$ je množina rodičovských proměnných k proměnné $\mathbf{z}_j$ v orientovaném grafu. Podobně jako u DLVM, rozdělení pravděpodobnosti $q_\phi(\mathbf{z} \mid \mathbf{x})$ může být parametrizováno použitím umělé neuronové sítě. V takovém případě variační parametry ϕ zahrnují váhy a biasy této umělé neuronové sítě (D. P. Kingma a M. Welling, 2014):

$$(\boldsymbol{\mu}, \log \boldsymbol{\sigma}) = \text{EnkodérSíť}_\phi(\mathbf{x}) \tag{3.6}$$

$$q_\phi(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma})) \tag{3.7}$$

Pro posteriorní inferenci skrze všechna vstupní data je použita vždy jedna stejná enkodér neuronová síť, tedy **variační parametry jsou sdíleny** [4]. Tato strategie, kterou VAE využívá pro **sdílení variačních parametrů skrze všechny datové vstupy** se nazývá **amortizovaná variační inference** (Gershman a Goodman, 2014). Amortizovanou inferencí se VAE

[3] Společně optimalziovat (*joint optimization*) zde znamená, že generativní model (který mapuje latentní proměnná na data) a inferenční model (který mapuje data na latentní proměnné) jsou optimalizovány zároveň (raději než optimalizovány nezávisle). Ztrátová funkce, která je při trénování optimalizována, zahrnuje oba modely. Touto společnou optimalizací se VAE mohou učit generovat nové vzorky dat, které mají podobné vlastnosti jako data v trénovací množině a odvozovat skryté latentní vztahy proměnných, která generovala pozorovaná data

[4] V kontrastu s tradičními metodami pro variační inferenci, kde jsou parametry iterativně optimalizovány pro každý datový vstup

vyhýbají optimalizační smyčce pro každý datový bod, a mohou tak plně využít možnosti efektivity stochastického gradientního sestupu. (Diederik P. Kingma a Max Welling, 2019)

3.3 Dekodér modul

Dekodér modul VAE je pravděpodobnostní dekodér $p_\theta(\mathbf{x} \mid \mathbf{z})$, který na základě *kódu* $\mathbf{z}$ produkuje rozdělení pravděpodobnosti skrze všechny možné hodnoty, kterých může $\mathbf{x}$ nabývat. (D. P. Kingma a M. Welling, 2014)

3.4 Evidence Lower Bound

Účelovou funkcí VAE je *evidence lower bound*, dále jen ELBO (alternativně se lze setkat s pojmenováním *variační dolní mez*). Typicky je ELBO odvozena pomocí Jensenovi nerovnosti (Wasserman, 2013, Sekce 4.2). Autoři VAE (D. P. Kingma a M. Welling, 2014) však využívají alternativního postup, jenž se chytře vyhýbá použití Jensenovi nerovnosti a nabízí větší míru *tightness* [5]. Pro libovolnou konfiguraci inferenčního modelu $q_\phi(\mathbf{z} \mid \mathbf{x})$, včetně volby variačních parametrů ϕ, dostáváme (úpravy rovnic převzaty z (Diederik P. Kingma a Max Welling, 2019)):

$$\log p_\theta(\mathbf{x}) = \mathbb{E}_{q\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x})] \tag{3.8}$$

$$= \mathbb{E}_{q\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{p_\theta(\mathbf{z} \mid \mathbf{x})}\right]\right] \tag{3.9}$$

$$= \mathbb{E}_{q\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})}\frac{q_\phi(\mathbf{z} \mid \mathbf{x})}{p_\theta(\mathbf{z} \mid \mathbf{x})}\right]\right] \tag{3.10}$$

$$= \underbrace{\mathbb{E}_{q\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})}\right]\right]}_{\iff \mathcal{L}_{\theta,\phi}(\mathbf{x})(ELBO)} + \underbrace{\mathbb{E}_{q\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{q_\phi(\mathbf{z} \mid \mathbf{x})}{p_\theta(\mathbf{z} \mid \mathbf{x})}\right]\right]}_{\iff D_{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p_\theta(\mathbf{z} \mid \mathbf{x}))} \tag{3.11}$$

$$\tag{3.12}$$

Druhý prvek Rovnice 3.12 je Kullback-Lieblerova divergence (Sekce 1.6) mezi $q_\phi(\mathbf{z} \mid \mathbf{x})$ (enkodérem) a $p_\theta(\mathbf{z} \mid \mathbf{x})$ (dekodérem), jejíž hodnota je **nenulová** [6]. Je-li hodnota této KL divergence nulová $\iff$ $q_\phi(\mathbf{z} \mid \mathbf{x})$ je rovno posteriorní distribuci původních dat (*perfektně je rekonstruuje*). (Diederik P. Kingma a Max Welling, 2019)

První prvek Rovnice 3.12 je variační dolní mez (ELBO):

[5] Koncept z matematické teorie míry, který lze intuitivně popsat jako *"sadu měr které příliš rychle do nekonečna"*. (Topsøe, 1974)

[6] $D_{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p_\theta(\mathbf{z} \mid \mathbf{x})) \geq 0$

$$\mathcal{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{q\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x})] \tag{3.13}$$

S ohledem na nenulovost KL divergence je tedy ELBO **dolní mezí**[7] log-likelihood původních dat. (D. P. Kingma a M. Welling, 2014), (I. Goodfellow et al., 2016) Dle Rovnice 3.13 platí:

$$\mathcal{L}_{\theta,\phi}(\mathbf{x}) = \log p_\theta(\mathbf{x}) - D_{KL}(q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p_\theta(\mathbf{z} \mid \mathbf{x})) \tag{3.14}$$

$$\leq \log p_\theta(\mathbf{x}) \tag{3.15}$$

V tomto případě tedy pozoruhodně tato KL divergence určuje hned dvě *vzdálenosti* (Diederik P. Kingma a Max Welling, 2019):

1. KL divergenci aproximace posteriorního rozdělení od původního posteriorního rozdělení (z definice KL divergence)
2. *Mezeru* mezi ELBO $\mathcal{L}_{\theta,\phi}(\mathbf{x})$ a *marginal likelihood* $\log p_\theta(\mathbf{x})$. Tuto *mezeru* také nazýváme mírou *tightness* této meze ($\implies$ vyhnutí se použití Jensenově nerovnosti). Čím lépe $q_\phi(\mathbf{z} \mid \mathbf{x})$ aproximuje $p_\theta(\mathbf{z} \mid \mathbf{x})$, s ohledem na KL divergenci, tím menší tato *mezera* je.

Optimalizační cíle

Při detailnější pohledu z Rovnice 3.15 plynou důsledky maximalizace ELBO $\mathcal{L}_{\theta,\phi}$ s ohledem na parametry $\boldsymbol{\theta}$ a $\boldsymbol{\phi}$. A to sice současná optimalizace dvou kritérií, která jsou pro výkonnost VAE stěžejní (Diederik P. Kingma a Max Welling, 2019):

1. *Přibližná* maximalizace *marginal likelihood* $p_\theta(\mathbf{x})$. Což implikuje *zlepšení* generativního modelu [8].
2. Minimalizace KL divergence aproximace mezi $q_\phi(\mathbf{z} \mid \mathbf{x})$ a původní posteriorní distribucí $p_\theta(\mathbf{z} \mid \mathbf{x})$. Tedy $q_\phi(\mathbf{z} \mid \mathbf{x})$ (enkodér) se *zlepší*.

3.4.1 Metoda stochastická gradientní optimalizace ELBO

Důležitou vlastností ELBO je možnost *joint* optimalizace s ohledem na veškeré její parametry – $\boldsymbol{\phi}$ a $\boldsymbol{\theta}$ – za použití stochastického gradientního sestupu (*stochastic gradient descent, SGD*). Proces lze zahájit náhodnými počátečními hodnotami $\boldsymbol{\phi}$ a $\boldsymbol{\theta}$ a stochasticky optimalizovat jejich hodnoty než dojde ke konvergenci.

Mějme nezávisle a rovnoměrně rozdělená vstupní data, účelovou funkcí ELBO je součet ELBO jednotlivých datových bodů:

$$\mathcal{L}_{\theta,\phi}(\mathcal{D}) = \sum_{x \in \mathcal{D}} \mathcal{L}_{\theta,\phi}(\mathbf{x}) \tag{3.16}$$

[7] Od tud *evidence* ***lower bound***

[8] Tedy *věrohodnosti* rekonstrukce původních dat

ELBO jednotlivých datových bodů a jejich gradienty jsou *obecně* efektivně řešitelné. Dokonce existují estimátory bez biasu, za pomocí kterých lze provést dávkový stochastický gradientní sestup (minibatch).

Gradienty ELBO bez biasu s ohledem na parametry generativního modelu $\boldsymbol{\theta}$ lze jednoduše získat (viz rovnice 2.14 - 2.17 z (Diederik P. Kingma a Max Welling, 2019)).

Spočítat gradienty ELBO bez biasu s ohledem na variační parametry $\boldsymbol{\phi}$ je již složitější, jelikož tato ELBO je uvažována s ohledem na rozdělení pravděpodobnosti $q_\phi(\mathbf{z} \mid \mathbf{x})$, tedy je funkcí ϕ. Nerovnost gradientů ELBO viz rovnice 2.18 - 2.19 z (Diederik P. Kingma a Max Welling, 2019).

V případě **spojitých** latentních proměnných lze využít tzv. **reparametrizačního triku** pro výpočet odhadů bez biasu $\nabla_{\theta,\phi}\mathcal{L}_{\theta,\phi}(\mathbf{x})$. Tento stochastický odhad umožňuje optimalizovat ELBO za použití stochastického gradientního sestupu (viz algorithm 1) [9].

Následuje představení reparametrizačního triku a jeho roli při trénování VAE.

3.5 Reparametrizační trik

Algorithm 1: Stochastic optimization of the ELBO.

Data:

$\mathcal{D}$: Dataset

$q_\phi(\mathbf{z} \mid \mathbf{x})$: Inferenční model (enkodér)

$p_\theta(\mathbf{x}, \mathbf{z})$: Generativní model (dekodér)

Result:

$\boldsymbol{\theta}, \boldsymbol{\phi}$: Naučené parametry

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Inicializace parametrů

while *SGD not converged* **do**

$\mathcal{M} \sim \mathcal{D}$ (Random minibatch of data)

$\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ (Random noise for every datapoint in $\mathcal{M}$)

Compute $\tilde{\mathcal{L}}$ and its gradients $\nabla_{\theta,\phi}(\mathcal{M}, \boldsymbol{\epsilon})$

Update $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ using SGD optimizer

end

3.6 Objective

Ve vztahu k Rovnice 1.7.

[9] Existují i metody pro případ **diskrétních** latentních proměnných, které však pro předmět práce nejsou stěžejní. Pro jejich představení odkazuji na (Diederik P. Kingma a Max Welling, 2019, Sekce 2.9.1.).

V reálném světe, pro většinu z, $P(X \mid z)$ bude téměř nulová (a tedy takové z nepřispěje téměř nic k zpřesnění odhadu $P(X)$). (Doersch, 2021)

Klíčovou myšlenkou VAE je, pokusit se vzorkovat pouze takové hodnoty z, u kterých je velká pravděpodobnost že vyprodukují X, a **vypočítat** $P(X)$ **pouze z nich**. To znamená, že potřebujeme zavést novou funkci $Q(z \mid X)$, která na vstupu bere hodnotu X a vrací rozdělení pravděpodobnosti skrze hodnoty z které mají vysokou pravděpodobnost na vyprodukování X [10]. (Doersch, 2021)

Tedy, relativně jednoduše lze spočítat $E_{z\sim Q}P(X \mid z)$. Ale co když je z vzorkováno z jiného (náhodně zvoleného) rozdělení s hustotou pravděpodobnosti $Q(z)$, které není $\mathcal{N}(\prime, \mathcal{I})$? Jak toto napomáhá k optimalizaci $P(X)$? První je nutné vyjádřit vztah mezi $E_{z\sim Q}P(X \mid z)$ a $P(X)$. (Doersch, 2021)

Vztah mezi $E_{z\sim Q}P(X \mid z)$ a $P(X)$ je jedním ze stavebních kamenů metod variační inference metod obecně. (Doersch, 2021)

Nejprve definujme KL divergenci mezi $P(z \mid X)$ a Q pro libovolné Q (které může, ale nemusí, záviset na X) (Doersch, 2021):

$$\mathcal{D}_{KL}(Q(z) \parallel P(z \mid X)) = E_{z\sim Q}P(X \mid z)(\log Q(z) - \log P(z \mid X)). \tag{3.17}$$

Aplikací Bayesova pravidla na $P(z \mid X)$ do rovnice začleníme $P(X)$ a $P(X \mid z)$ (Doersch, 2021):

$$\mathcal{D}_{KL}(Q(z) \parallel P(z \mid X)) = E_{z\sim Q}(\log Q(z) - \log P(X \mid z) - \log P(z)) + \log P(X). \tag{3.18}$$

Převodem $E_{z\sim Q}$ na prvky KL divergence, negací obou stran a přeskupením prvků získáme (Doersch, 2021):

$$\log P(X) - \mathcal{D}_{KL}(Q(z) \parallel P(z \mid X)) = E_{z\sim Q}(\log P(X \mid z)) - \mathcal{D}_{KL}(Q(z) \parallel P(z)). \tag{3.19}$$

Zde je X neměnné, Q může být *libovolné* rozdělení (**ne** pouze rozdělení, kde jsou X mapována na z která tyto X s vysokou pravděpodobností produkují).

Za účelem odvození $P(X)$ je smysluplné zavést Q které *závisí* na X a minimalizuje $\mathcal{D}_{KL}(Q(z) \parallel P(z))$ (Doersch, 2021):

[10] Zde platí očekávání, že prostor hodnot z, které nastanou při rozdělení $Q(z)$ je výrazně menší, než prostor hodnot apriorního rozdělení $P(z)$

$$\log P(X) - \mathcal{D}_{KL}\left[Q(z \mid X) \parallel P(z \mid X)\right] = E_{z\sim Q}\left[\log P(X \mid z)\right] - \mathcal{D}_{KL}\left[Q(z \mid X) \parallel P(z)\right]. \tag{3.20}$$

Rovnice 3.20 je pro **variační autoenkodéry naprosto stěžejní** a slouží jako jejich jádro [11]. (Doersch, 2021)

Interpretace rovnice

Levá strana rovnice vyjadřuje hodnotu, kterou chceme **maximalizovat**: $\log P(X)$ (plus prvek chyby, která nutí Q produkovat z taková, že jsou schopna rekonstruovat libovolné X). Snažíme se současně maximalizovat $\log P(X)$ a minimalizovat $\mathcal{D}_{KL}\left[Q(z \mid X) \parallel P(z \mid X)\right]$. (Doersch, 2021)

Pravá strana rovnice zaujímá roli jakéhosi **autoenkodéru**. Kde Q kóduje X do z. P jej následně dekóduje aby rekonstruovalo X. Za předpokladu správně zvoleného Q, si přejeme pravou stranu rovnice optimalizovat pomocí stochastického gradientního sestupu (zatím není zřejmé jak). (Doersch, 2021)

Z efektivně neřešitelného $P(z \mid X)$ se nyní stává efektivně řešitelný problém – stačí použít $Q(z \mid X)$ pro jeho výpočet. (Doersch, 2021)

3.7 Optimalizace pomocí stochastického gradientního sestupu

Za účelem optimalizace pravé strany Rovnice 3.20 je nutné specifikovat tvar, kterého bude $Q(z \mid X)$ nabývat.

Zvolíme $Q(z \mid X) = \mathcal{N}(z \mid \mu(X;\theta),\Sigma(X;\theta))$, kde μ a Σ jsou libovolné **deterministické** funkce s parametry θ, které se lze učit z dat. Funkce μ a Σ jsou (typicky) implementovány umělou neuronovou sítí, s tím že Σ musí být diagonální maticí. Takto zvolené $Q(z \mid X)$ má zejména **výpočetní výhodu** – pravá strana rovnice lze nyní jednoznačně vypočítat.

Nyní je tedy poslední prvek pravé strany, $\mathcal{D}_{KL}\left[Q(z \mid X) \parallel P(z \mid X)\right]$, KL divergence mezi dvěma vícerozměrnými Gaussovými rozděleními. Takovou KL divergenci lze spočítat v uzavřeném tvaru jako:

$$\mathcal{D}_{KL}\left[\mathcal{N}(\mu(X), \Sigma(X) \parallel \mathcal{N}(0, I))\right] = \frac{1}{2}\left[\mathrm{tr}(\Sigma_1^{-1}\Sigma_0) + (\mu_1 - \mu_0)^\top \Sigma_1^{-1}(\mu_1 - \mu_0) - k + \log\left(\frac{\det \Sigma_1}{\det \Sigma_0}\right)\right] \tag{3.21}$$

[11] TODO: Historically, this math (particularly Equation 5) was known long before VAEs

kde k je dimenzionlita výsledného rozdělení pravděpodobnosti. Rovnice 3.21 lze zjednodušit následovně:

$$\mathcal{D}_{KL}\left[\mathcal{N}(\mu(X),\Sigma(X)) \parallel \mathcal{N}(0,I)\right] = \frac{1}{2}\left[\mathrm{tr}(\Sigma(X)) + (\mu(X))^{\top}(\mu(X)) - k - \log\det(\Sigma(X))\right]. \tag{3.22}$$

První prvek na pravé straně Rovnice 3.20 je o něco komplikovanější.

Bylo by možné pomocí vzorků odhadnout $E_{Z\sim Q}[\log P(X \mid z)]$, ale pro získání *věrohodného* odhadu je nutné skrze z poslat velké množství vzorků, což je výpočetně náročné. Tedy, jak je u stochastického gradientního sestupu běžné, použijeme pouze jeden vzorek z a použijeme $P(X \mid z)$ pro toto z jako aproximaci $E_{Z\sim Q}[\log P(X \mid z)]$ [12].

Tedy konečná rovnice kterou chceme optimalizovat má následující podobu:

$$E_{X\sim D}\left[\log P(X) - \mathcal{D}_{KL}\left[Q(z \mid X) \parallel P(z \mid X)\right]\right] = E_{X\sim D}\left[E_{z\sim Q}\left[\log P(X \mid z)\right] - \mathcal{D}_{KL}\left[Q(z \mid X) \parallel P(z)\right]\right] \tag{3.23}$$

Vezmeme-li gradient této rovnice, symbol gradientu může být přesunut do očekávané hodnoty. Tedy, můžeme vzorkovat jednu hodnotu z X a jednu hodnotu ze z z rozdělení pravděpodobnosti $Q(z \mid X)$ a následně vypočítat gradient:

$$\log P(X \mid z) - \mathcal{D}_{KL}\left[Q(z \mid X) \parallel P(z)\right]. \tag{3.24}$$

Následně zprůměrujeme gradient této funkce skrze *libovolně velké* množství vzorků z X a z – výsledná hodnota tohoto gradientu konverguje ke gradientu Rovnice 3.23.

Rovnice 3.23 má jeden zásadní problém. $E_{z\sim Q}\left[\log P(X \mid z)\right]$ závisí na parametrech P a na parametrech Q. Tato vazba však v Rovnice 3.24 zmizela.

Aby VAE fungovaly dle očekávání, je nutné nutit Q produkovat takové kódy pro X, které bude P schopno **spolehlivě** dekódovat. Na tento problém lze alternativně pohlížet interpretací Rovnice 3.23 jako sítě vyobrazené TODO FIGURE NO-BACKPROP. Dopředný průchod této sítě funguje bez problémů – a je-li její výstup zprůměrován skrze mnoho vzorků X a z, produkuje správnou a očekávanou hodnotu. Nicméně, je nutné mít schopnost zpětně propagovat chybu skrze vrstvu, která vzorkuje z z $Q(z \mid X)$, což je **nespojitá operace** a tedy nemá žádný gradient. Stochastický gradientní sestup se zpětnou propagací zvládne stochastické vstupy, ale **neumí pracovat se stochastickými jednotkami (neurony) sítě**.

[12] Zde je výhodou, že provést stochastický gradientní sestup skrze všechny různé hodnoty X vzorkované z datové sady D již stochastický gradientní sestup provádíme při trénování.

Řešení tohoto problému nazýváme **reparametrizační trik** [13]. Reparametrizační trik spočívá v přesunutí procesu vzorkování do vstupní vrstvy. Mějme $\mu(X)$ a $\Sigma(X)$ – **střední hodnotu** a **kovarianci** $Q(z \mid X)$. Pak je možné vzorkovat z $\mathcal{N}(\mu(X), \Sigma X)$ – nejprve provedeme vzorkování z $\epsilon \sim \mathcal{N}(0, I)$ a následně spočteme $z = \mu(X) + \Sigma^{\frac{1}{2}}(X) * \epsilon$.

Tedy finální rovnice, jejíž gradient chceme spočítat má následující tvar:

$$E_{X \sim D}\left[E_{\epsilon \sim \mathcal{N}(0,1)}\left[\log P(X \mid z = \mu(X) + \Sigma^{\frac{1}{2}}(X) * \epsilon)\right] - \mathcal{D}_K L\left[Q(z \mid X) \parallel P(z)\right]\right]. \quad (3.25)$$

Kýžená vlastnost Rovnice 3.25 je, že v modelu její **umělé neuronové sítě lze provést zpětnou propagaci**. Toto je znázorněno v umělé neuronové síti TODO FIGURE NO-BACKPROP

[14].

3.8 Formalizace

3.9 Model umělé neuronové sítě

3.10 Nedostatky a omezení

3.11 Rozšíření a aktuální stav poznání

3.12 Pozorování v latentním prostoru

[13] Reparametrizační trik *funguje* pouze pakliže lze vzorkovat z $Q(z \mid X)$ vyhodnocením funkce $h(\eta, X)$, kde η je šum (z distribuce jejíž parametry nejsou učeny z dat). h také musí být spojitá na X, abychom skrze něj mohli provádět zpětnou propagaci. To, mimo jiné, znamená, že $Q(z \mid X)$ a tím padem i $P(z)$ **nemohou být diskrétní rozdělení**. V opačném případě by došlo k nespojitosti prostoru vzorků Q – a tedy neschopnosti generativního modelu generovat vzorky v celém rozsahu, včetně vzorků které nebyl součástí dat (běžný problém autoenkodérů, které uvedla Kapitola 2.)

[14] Žádné očekávané hodnoty (E) Rovnice 3.25 **nezávisí** na parametrech modelu. Tedy do těchto očekávaných hodnot můžeme bezpečně převést symbol gradientu a zároveň dodržet jejich rovnost. Tedy, mějme neměnné X a ϵ, pak je tato funkce **spojitá a deterministická** v parametrech P a Q, což v důsledku znamená možnost zpětné propagace vypočíst gradient algoritmem stochastického gradientního sestupu.

4. Úlohy pozorování v latentním prostoru

4.1 Generativní modelování obrazových dat

4.2 Rekonstrukce obrazových dat

4.3 Interpolace vět

4.4 Detekce anomálií

4.5 Syntéza tabulárních dat

4.6 Komprese

5. Experimenty s modelem variačního autoenkodéru

5.1 Generativní modelování obrazových dat

5.1.1 Vymezení problémové oblasti

5.1.2 Datová sada a předzpracování

5.1.3 Nastavení experimentu

5.1.4 Návrh modelu

5.1.5 Evaluace

5.1.6 Diskuze

5.2 Interpolace vět

Závěr

The conclusion is a mandatory part of the bachelor's / diploma thesis. It contains a summary of the work and comments on the degree of fulfillment of the goal, which was set in the work, or summarizes the answers to the questions that were asked in the introduction.

The conclusion to the diploma thesis must be more elaborate - this is stated in more detail in the Requirements of the diploma thesis within the Intranet for FIS students.

The conclusion is perceived as a chapter, which begins on a separate page and is called the conclusion. The name Conclusion is not numbered. The text of the conclusion itself is divided into paragraphs.

Bibliografie

BALDI, Pierre a HORNIK, Kurt, 1989. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks* [online]. Roč. 2, č. 1, s. 53–58 [cit. 2023-03-09]. ISSN 0893-6080. Dostupné z DOI: 10.1016/0893-6080(89)90014-2.

BANERJEE, Arindam, 2007. An Analysis of Logistic Models: Exponential Family Connections and Online Performance. In: [online]. Society for Industrial a Applied Mathematics, s. 204–215 [cit. 2023-03-25]. Proceedings. ISBN 9780898716306. Dostupné z DOI: 10.1137/1.9781611972771.19.

BELLMAN, Richard, 1957. *Dynamic Programming*. Princeton University Press. ISBN 9780691079516. Google-Books-ID: wdtoPwAACAAJ.

BENGIO, Yoshua, COURVILLE, Aaron a VINCENT, Pascal, 2014. *Representation Learning: A Review and New Perspectives* [online]. 2014-04 [cit. 2023-03-25]. Tech. zpr. arXiv. Dostupné z DOI: 10.48550/arXiv.1206.5538. arXiv:1206.5538 [cs] type: article.

BENGIO, Yoshua, LAMBLIN, Pascal, POPOVICI, Dan a LAROCHELLE, Hugo, 2006. Greedy Layer-Wise Training of Deep Networks. In: *Advances in Neural Information Processing Systems* [online]. MIT Press. Sv. 19 [cit. 2023-03-06]. Dostupné z: https://proceedings.neurips.cc/paper/2006/hash/5da713a690c067105aeb2fae32403405-Abstract.html.

CYBENKO, G., 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems* [online]. Roč. 2, č. 4, s. 303–314 [cit. 2023-03-08]. ISSN 1435-568X. Dostupné z DOI: 10.1007/BF02551274.

DAYAN, P., HINTON, G. E., NEAL, R. M. a ZEMEL, R. S., 1995. The Helmholtz machine. *Neural Computation*. Roč. 7, č. 5, s. 889–904. ISSN 0899-7667. Dostupné z DOI: 10.1162/neco.1995.7.5.889.

DOERSCH, Carl, 2021. *Tutorial on Variational Autoencoders* [online]. 2021-01 [cit. 2023-03-31]. Tech. zpr. arXiv. Dostupné z DOI: 10.48550/arXiv.1606.05908. arXiv:1606.05908 [cs, stat] type: article.

ERHAN, Dumitru, BENGIO, Yoshua, COURVILLE, Aaron, MANZAGOL, Pierre-Antoine, VINCENT, Pascal a BENGIO, Samy, 2010. Why Does Unsupervised Pre-training Help Deep Learning? *Journal of Machine Learning Research* [online]. Roč. 11, č. 19, s. 625–660 [cit. 2023-03-06]. ISSN 1533-7928. Dostupné z: http://jmlr.org/papers/v11/erhan10a.html.

GERON, Aurelien, 2019. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2nd. O'Reilly Media, Inc. ISBN 9781492032649.

GERSHMAN, Samuel J. a GOODMAN, Noah D., 2014.
Amortized Inference in Probabilistic Reasoning. *Cognitive Science*. Roč. 36.

GOODFELLOW, Ian, BENGIO, Yoshua a COURVILLE, Aaron, 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

GOODFELLOW, Ian J., POUGET-ABADIE, Jean, MIRZA, Mehdi, XU, Bing, WARDE-FARLEY, David, OZAIR, Sherjil, COURVILLE, Aaron a BENGIO, Yoshua, 2014. *Generative Adversarial Networks* [online]. 2014-06 [cit. 2023-03-09]. Tech. zpr. arXiv. Dostupné z DOI: 10.48550/arXiv.1406.2661.
arXiv:1406.2661 [cs, stat] type: article.

HEBB, Donald O., 1949.
The organization of behavior: A neuropsychological theory [Hardcover].
New York: Wiley. ISBN 0-8058-4300-0.

HOCHREITER, Sepp, 1998. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions.
International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems [online]. Roč. 06, č. 02, s. 107–116 [cit. 2023-03-09]. ISSN 0218-4885.
Dostupné z DOI: 10.1142/S0218488598000094.

HORNIK, Kurt, STINCHCOMBE, Maxwell a WHITE, Halbert, 1990.
Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks* [online]. Roč. 3, č. 5, s. 551–560 [cit. 2023-03-08]. ISSN 0893-6080. Dostupné z DOI: 10.1016/0893-6080(90)90005-6.

HORNIK, Kurt, STINCHCOMBE, Maxwell B. a WHITE, Halbert L., 1989.
Multilayer feedforward networks are universal approximators. *Neural Networks*. Roč. 2, s. 359–366.

CHARTE, David, CHARTE, Francisco, GARCÍA, Salvador, JESUS, María J. del a HERRERA, Francisco, 2018. A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines. *Information Fusion* [online]. Roč. 44, s. 78–96 [cit. 2023-03-04]. ISSN 15662535.
Dostupné z DOI: 10.1016/j.inffus.2017.12.007. arXiv:1801.01586 [cs].

CHOLLET, François, 2017. *Deep Learning with Python*. Manning. ISBN 9781617294433.

KAMYSHANSKA, Hanna a MEMISEVIC, Roland, 2013. On autoencoder scoring. In.

KINGMA, D. P. a WELLING, M., 2014. Auto-Encoding Variational Bayes [online] [cit. 2023-03-24]. Dostupné z: https://dare.uva.nl/search?identifier=cf65ba0f-d88f-4a49-8ebd-3a7fce86edd7.

KINGMA, Diederik P. a WELLING, Max, 2019.
An Introduction to Variational Autoencoders.
Foundations and Trends® in Machine Learning [online].
Roč. 12, č. 4, s. 307–392 [cit. 2023-03-09]. ISSN 1935-8237, ISSN 1935-8245.
Dostupné z DOI: 10.1561/2200000056. arXiv:1906.02691 [cs, stat].

KULLBACK, S. a LEIBLER, R. A., 1951. On Information and Sufficiency. *The Annals of Mathematical Statistics* [online]. Roč. 22, č. 1, s. 79–86 [cit. 2023-03-09]. ISSN 0003-4851. Dostupné z: https://www.jstor.org/stable/2236703.

LECUN, Y., BOSER, B., DENKER, J. S., HENDERSON, D., HOWARD, R. E., HUBBARD, W. a JACKEL, L. D., 1989. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*. Roč. 1, č. 4, s. 541–551. ISSN 0899-7667. Dostupné z DOI: 10.1162/neco.1989.1.4.541.

LECUN, Yann, 2022. *A Path Towards Autonomous Machine Intelligence* [online] [cit. 2023-03-25]. Dostupné z: https://openreview.net/forum?id=BZ5a1r-kVsf.

LIU, Huan a MOTODA, Hiroshi (ed.), 1998. *Feature Extraction, Construction and Selection* [online]. Boston, MA: Springer US [cit. 2023-03-05]. ISBN 9781461376224 9781461557258. Dostupné z DOI: 10.1007/978-1-4615-5725-8.

LIU, Weifeng, POKHAREL, P.P. a PRINCIPE, J.C., 2006. Correntropy: A Localized Similarity Measure. In: *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, s. 4919–4924. Dostupné z DOI: 10.1109/IJCNN.2006.247192.

MINSKY, M. a PAPERT, S., 1969. *Perceptrons*. Cambridge, MA: MIT Press.

MITCHELL, Tom M., 1997. *Machine Learning*. New York: McGraw-Hill. ISBN 978-0-07-042807-2.

MURPHY, Kevin P., 2022. *Probabilistic Machine Learning: An introduction*. MIT Press. Dostupné také z: probml.ai.

MURPHY, Kevin P., 2023. *Probabilistic Machine Learning: Advanced Topics*. MIT Press. Dostupné také z: http://probml.github.io/book2.

OLSHAUSEN, Bruno A. a FIELD, David J., 1997. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research* [online]. Roč. 37, č. 23, s. 3311–3325 [cit. 2023-03-09]. ISSN 0042-6989. Dostupné z DOI: 10.1016/S0042-6989(97)00169-7.

PHILLIPS, Jeff M., 2021. *Mathematical Foundations for Data Analysis*. Springer International Publishing. ISBN 9783030623401. Google-Books-ID: AUDYzQEACAAJ.

RANZATO, M.A., HUANG, Fu, BOUREAU, Y-Lan a LECUN, Yann, 2007. Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition. In: s. 1–8. Dostupné z DOI: 10.1109/CVPR.2007.383157.

REZENDE, Danilo Jimenez, MOHAMED, Shakir a WIERSTRA, Daan, 2014. *Stochastic Backpropagation and Approximate Inference in Deep Generative Models* [online]. 2014-05 [cit. 2023-03-24]. Tech. zpr. arXiv. Dostupné z DOI: 10.48550/arXiv.1401.4082. arXiv:1401.4082 [cs, stat] type: article.

RIFAI, Salah, BENGIO, Yoshua, DAUPHIN, Yann a VINCENT, Pascal, 2012.
A Generative Process for Sampling Contractive Auto-Encoders [online].
2012-06 [cit. 2023-03-09]. Tech. zpr. arXiv.
Dostupné z DOI: 10.48550/arXiv.1206.6434. arXiv:1206.6434 [cs, stat] type: article.

ROSENBLATT, F., 1957. *The perceptron - A perceiving and recognizing automaton.*
Ithaca, New York, 1957-01. Tech. zpr., 85-460-1. Cornell Aeronautical Laboratory.

RUMELHART, David E. a MCCLELLAND, James L., 1987.
Learning Internal Representations by Error Propagation. In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, s. 318–362.

SAMUEL, Arthur L., 1967.
Some Studies in Machine Learning Using the Game of Checkers. *IBM J. Res. Dev.*
Roč. 44, s. 206–227.

SØNDERBY, Casper, RAIKO, Tapani, MAALØE, Lars, SØNDERBY, Søren a WINTHER, Ole, 2016.
How to Train Deep Variational Autoencoders and Probabilistic Ladder Networks.

STAŃCZYK, Urszula a JAIN, Lakhmi C. (ed.), 2015.
Feature Selection for Data and Pattern Recognition [online].
Berlin, Heidelberg: Springer Berlin Heidelberg [cit. 2023-03-05].
Studies in Computational Intelligence. ISBN 9783662456194 9783662456200.
Dostupné z DOI: 10.1007/978-3-662-45620-0.

TOPSØE, Flemming, 1974. Compactness and Tightness in a Space of Measures with the Topology of Weak Convergence. *Mathematica Scandinavica* [online].
Roč. 34, č. 2, s. 187–210 [cit. 2023-03-26]. ISSN 0025-5521.
Dostupné z: https://www.jstor.org/stable/24490646.

VALIANT, L. G., 1984. A theory of the learnable. *Communications of the ACM* [online].
Roč. 27, č. 11, s. 1134–1142 [cit. 2023-03-08]. ISSN 0001-0782.
Dostupné z DOI: 10.1145/1968.1972.

WASSERMAN, L., 2013. *All of Statistics: A Concise Course in Statistical Inference.*
Springer New York. Springer Texts in Statistics. ISBN 9780387217369.
Dostupné také z: https://books.google.cz/books?id=qrcuBAAAQBAJ.

WERBOS, P. J., 1974.
Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.
Dis. pr. Harvard University.

WOLPERT, David H., 1996.
The Lack of A Priori Distinctions Between Learning Algorithms. *Neural Computation.*
Roč. 8, č. 7, s. 1341–1390. ISSN 0899-7667.
Dostupné z DOI: 10.1162/neco.1996.8.7.1341.

Přílohy

A. Zdrojové kódy modelů