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 Umělé neuronové sítě

1.1.1 Perceptron

1.1.2 Hebbovské účení

1.1.3 Universal approximation theorem

1.1.4 Vícevrstvý Perceptron

1.1.5 Dopředná umělá neuronová síť

1.1.6 Gradient descent

1.1.7 Backpropagation

1.1.8 Strojové učení

1.1.9 Hluboké učení

1.1.10 Konvoluční sítě

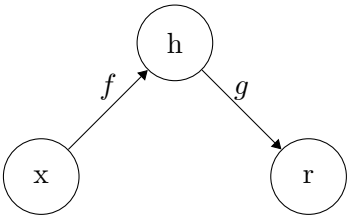
1.2 Redukce dimenzionality

1.2.1 The Curse of Dimensionality

1.2.2 Analýza hlavních komponent

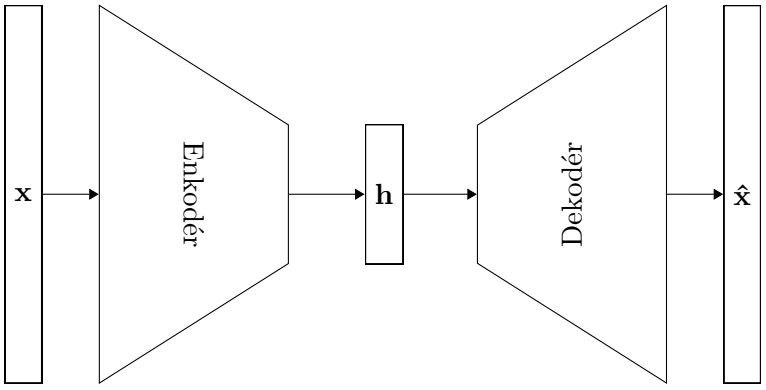
1.3 Autoenkodér

Autoenkodér je typ umělé neuronové sítě se schopností učit se efektivní reprezentace vstupních dat bez učitele. Umělá neuronová síť Autoenkodéru má symetrickou strukturu a skrytou vrstvu h, která popisuje *kód* použitý pro reprezentaci vstupu. Architekturu Autoenkodéru (viz 1.1) 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.



Obrázek 1.1: 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 1.1) lze reprezentovat dopřednou umělou neuronovou sítí. Jejím cílem je **rekonstruovat vstupní data na výstupní vrstvě**. 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. Obecná architektura modulů umělé neuronové sítě Autoenkodéru je zachycena v 1.2.



Obrázek 1.2: Jednotlivé moduly architektury umělé neuronové sítě Autoenkodéru.

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.

1.3.1 Historický pohled

Vícevrstvý Perceptron subsection 1.1.4 je univerzálním aproximátorem subsection 1.1.3 – 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 (*vanishing gradient problem*). Až příchod algoritmu gradientního sestupu subsection 1.1.6, který adresuje problém mizejícího gradientu v aplikacích s použitím konvolučních sítí subsection 1.1.10 a úloh učení se bez učitele, značí počátek moderních metod hlubokého učení. V oblasti hlubokého učení subsection 1.1.9 dochází k emergenci a vývoji řady technik pro řešení úloh učení se 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 *feature extraction*). Později Autoenkodéry nalézají uplatnění zejména v úloháh redukce dimenzionality section 1.2 či fůzi vlastností (*feature fusion*).

Nedávné teoretické propojení Autoenkodéru a Modelů využívajících latentní proměnných section 1.6 však vedlo ke vzniku zcela nové architektury neuronové sítě kombinující charakter redukce dimenzionality Autoenkodéru se statistickými metodami odvozování. To vyneslo Autoenkodéry na popředí v oblasti generativního modelování – této architektuře je věnována kapitola chapter 2.

Byť Autoenkodéry vznikly v kontextu hlubokého učení, není pravidlem že všechny modely Autoenkodéru obsahují vícero skrytých vrstev. Následuje rozdělení Autoenkodérů dle struktury umělé neuronové sítě.

1.3.2 Mělký autoenkodér

Mělký autoenkodér (*Shallow autoencoder*)

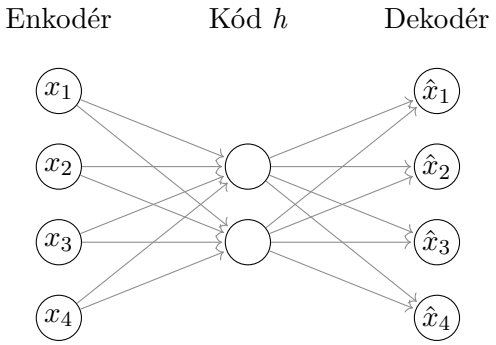
1.3.3 Autoenkodér s neúplnou skrytou vrstvou

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 ty stěžejní vlastnosti trénovacích dat, které následně budou použity pro rekonstrukci.

Trénovací proces Neuplného autoenkodéru je popsán jako minimalizace ztrátové funkce:

$$L(\mathbf{x}, g(f(\mathbf{x}))), \tag{1.1}$$

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



Obrázek 1.3: 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 L je použita *střední kvadratická chyba*, pak se Neuplný autoenkodér naučí stejný *vektorový prostor*, který by byl výsledkem Analýzy hlavních komponentů subsection 1.2.2. V tomto speciálním případě lze ukázat, že Autoenkodér trénovaný na úloze kompresované reprezentace znalostí jako vedlejší efekt provedl Analýzu hlavních komponentů.

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.

Na druhou stranu, má-li Autoenkodér k dispozici příliš mnoho kapacity, může se naučit kopírovat vstupní data na výstupní vrstvu bez extrakce užitečných (charakteristických) vlastností o rozdělení vstupních dat.

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. Proto je třeba Autoenkodéry regularizovat.

Dále tedy budou představeny přístupy k architekturám Autoenkodérů **s využitím regularizace**.

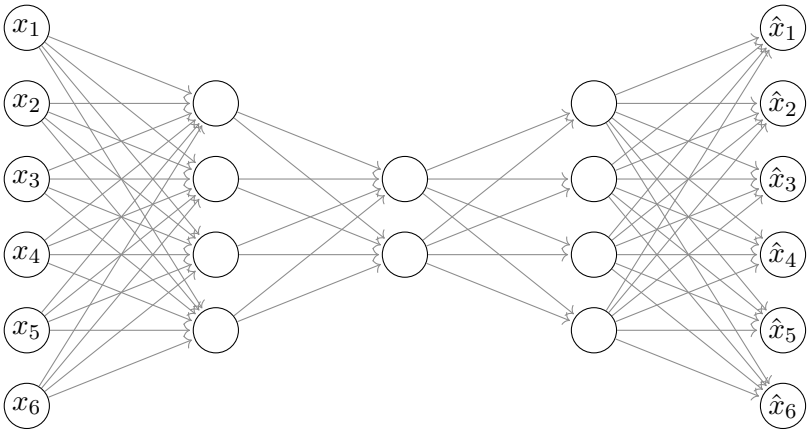
1.3.4 Autoenkodér s rozšířenou skrytou vrstvou

Autoenkodér s rozšířenou skrytou vrstvou (*Overcomplete Autoencoder*) je Autoenkodér, jehož počet neuronů ve skrýté vrstvě je větší než počet neuronů vstupní (a výstupní) vrstvy.

1.3.5 Hlubuký autoenkodér

V sekci subsection 1.3.3 a subsection 1.3.4 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 (*kód*) $\boldsymbol{h}$.



Obrázek 1.4: Deep Autoenkodér.

Z netriviální hloubky dopředné umělé neuronové sítě plyne subsection 1.1.3, 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).

Nicméně u mělkých enkodérů, které jsou rovněž dopřednou sítí, neexistuje možnost představit libovolná omezení a regularizační prvky (například řídkost *kódu* $\boldsymbol{h}$ – viz subsection 1.3.6). A tedy nelze zamezit problému naučiení pouhé identity section 1.3.3. Narozdíl od mělkých autoenkoderů subsection 1.3.2, však mohou Hluboké autoenkodéry (*libovolně přesně*) aproximovat jakékoliv mapování vstupu na kód (*opět s předpokladem dostatečného počtu neuronů skryté vrstvy*).

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

Stacked autoenkodér

Běžnou strategií pro trénování Hlubokého autoenkodéru je hladově předtrénovat (*greedy pre-training*) model individuálním natrénováním většího počtu Mělkých autoenkodérů (představených v sekci subsection 1.3.2), které jsou následně vloženy za sebe. Takto složený Autoenkodér nazýváme Stacked autoenkodér.

1.3.6 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.

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

$$L(\mathbf{x}, g(f(\mathbf{x}))) + \Omega(\boldsymbol{h}), \tag{1.2}$$

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.

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:

$$\hat{p_i} = \frac{1}{|S|} \sum_{x \in S} f_i(x), \tag{1.3}$$

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).

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

$$KL(p \parallel \hat{p_i}) = p \log \frac{p}{\hat{p_i}} + (1-p) \log \frac{1-p}{1-\hat{p_i}}. \tag{1.4}$$

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

$$\Omega_{\check{\mathrm{R}}AE}(W, b; S) = \sum_{i=1}^{c} KL(p \parallel \hat{p}_i), \tag{1.5}$$

kde průměrná hodnota aktivací $\hat{p}_i$ závisí na parametrech enkodéru a množině trénovacích dat S.

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 section 1.3.3).

1.3.7 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:

$$L(\mathbf{x}, g(f(\mathbf{x}))), \tag{1.6}$$

kde L je ztrátová funkce penalizující $g(f(x))$ za odlišnost od $\mathbf{x}$ (např. Euklidovská norma jejich rozdílů). Jak ale bylo ukázáno v section 1.3.3, to umožňuje $f \circ g$ naučit se být pouhou identitu.

Z toho důvodu Denoising autoenkodér (*Denoising Autoencoder*) minimalizuje funkci:

$$L(\boldsymbol{x}, g(f(\tilde{\boldsymbol{x}}))), \tag{1.7}$$

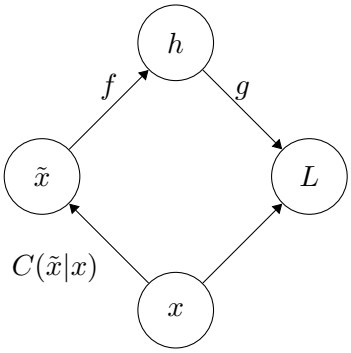
kde $\tilde{\boldsymbol{x}}$ je kopií $\boldsymbol{x}$ která byla úmyslně poškozena procesem $C(\tilde{x}|x)$ (*corruption process*), který reprezentuje podmíněné rozdělení pravděpodobnosti poškozenných vzorků $\tilde{x}$ v závislosti na vzorku vstupních dat x.

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 $p_{data}(\boldsymbol{x})$.

Trénovací procedura Denoising autoenkodéru lze schematicky znázornit následovně (viz 1.5):



Obrázek 1.5: DAE

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:

- 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**

1.3.8 Robustní autoenkodér

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

Robustní autoenkodéry, narozdíl od Denoising autoenkodéru (viz subsection 1.3.7), využivájí alternativně definovanou ztřatovou funkci, která je modifikována pro minimalizaci chyby rekonstrukce. Obecně jsou ne-Gaussovským šumem ovlivněny **méně než triviální mělké autoenkodéry** (viz subsection 1.3.2). Tato alternativní ztrátová funkce je založena na correntropii (*correntropy*, lokalizovaná míra podobnosti), která je v **CITACE** definována následovně:

$$\mathcal{L}_{MCC}(u, v) = -\sum_{k=1}^{d} \mathcal{K}_\sigma(u_k - v_k), \tag{1.8}$$

kde

$$\mathcal{K}_\sigma(\alpha) = \frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{\alpha^2}{2\sigma^2}), \tag{1.9}$$

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

Correntropie měří hustotu pravdepodobnostni, že dvě *události* jsou si rovny (zde událost může být např.:). Correntropie je **výrazně méně ovlivněna odlehlými hodnotami**, než

např. střední kvadratická chyba. Robustní autoenkodér se snaží tuto míru maximalizovat, což intuitivně vede k **vyšší odolnosti** Robustního autoenkodéru **na ne-Gaussovský šum** přítomný ve vstupních datech.

1.3.9 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 autonekodér (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.

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

$$\|J_f(x)\|_F^2 = \sum_{j=1}^{d} \sum_{i=1}^{c} \left(\frac{\partial f_i}{\partial x_j}(x) \right)^2 . \tag{1.10}$$

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:

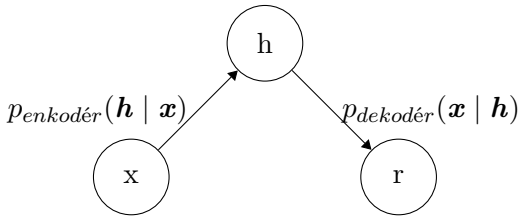
$$\Omega_{CAE}(W, b, S) = \sum_{x \in S} \|J_f(x)\|_F^2. \tag{1.11}$$

Výsledkem je tedy Autoenkodér, jehož dva (lokálně) *podobné* vstupy musejí mít i *podobný* kód. Z Contractive autoenkodéru lze rovněž vzorkovat nové výstupy. Z takto naučeného modelu Autoenkodéru lze generovat nové instance dat: 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.

1.3.10 Stochastický autoenkodér

Struktura stochastického autoenkodéru se vůči struktuře představené v 1.1 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 pravěpodobnosti**.

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 (1.1) tedy dostáváme:



Obrázek 1.6: 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 subsection 1.1.5 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).

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 section 1.6). Model využívající latentní proměnné značíme $p_{model}(\boldsymbol{h}, \boldsymbol{x})$.

Stochastický enkodér je pak definován následovně:

$$p_{enkodér}(\boldsymbol{h} \mid \boldsymbol{x}) = p_{model}(\boldsymbol{h}, \boldsymbol{x}) \tag{1.12}$$

a **stochastický dekodér** následovně:

$$p_{dekodér}(\boldsymbol{x} \mid \boldsymbol{h}) = p_{model}(\boldsymbol{x}, \boldsymbol{h}) \tag{1.13}$$

Výběr nových dat z rozdělení pravděpodobnosti autoenkodéru je detailněji představen v kaptiole chapter 2.

1.3.11 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 chapter 3.

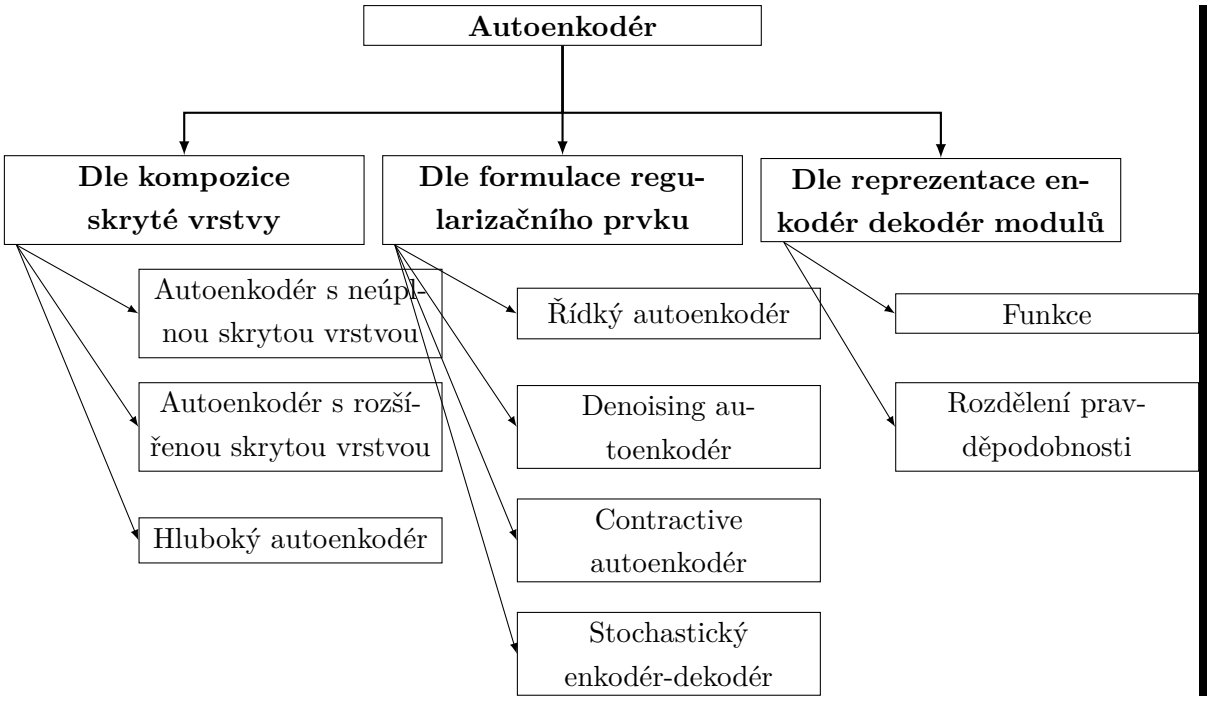
V této sekci budou stučně shrnuty *ostatní* typy autoenkodérů, které nejsou pro předmět nadcházejících kapitol stěžejní, ale obecně se jim dostává vysoké míry využití.

Adversariální autoenkodér

Adversariální autoenkodér (*Adversarial autoencoder*) přináší koncept Generativních Adversariálních sítí **CITACE** na pole autoenkodérů. V adversariálním autoenkodéru je kód ($\boldsymbol{h}$) modelován uložením předchozího statistického rozdělení (*prior*). Následně je natrénován *běžný* autoenkodér, současně se *diskriminativní* síť snaží odlišit kódy modelu od výběrů dat z přechozího statistického rozdělení. 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*).

1.3.12 Taxonomie autoenkodérů

Bylo představeno několik tříd Autoenkoderů (rozdělení dle navržení ANN sítě, rozdělení dle regularizačního prvku) a následně popsáno několik dalších druhú AE. Zde je jejich (nevyčerpávající) taxonomie.



Obrázek 1.7: Autoenkodéry rozděleny dle charakteristik zpracování kódovací vrstvy

1.3.13 Využití Autoenkodéru

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

1.4 Pravděpodobnostní modely a inference pomocí variačního Bayese

1.5 Kullback–Lieblerova divergence

1.6 Modely využívající latentních proměnných // Latent Variable Models

2. Variační autoenkodér

2.1 Evidence Lower Bound

2.2 Reparametrizační trik

2.3 Formalizace

2.4 Model umělé neuronové sítě

2.5 Nedostatky a omezení

2.6 Rozšíření a aktuální stav poznání

2.7 Pozorování v latentním prostoru

3. Úlohy pozorování v latentním prostoru

3.1 Generativní modelování obrazových dat

3.2 Rekonstrukce obrazových dat

3.3 Interpolace vět

3.4 Detekce anomálií

3.5 Syntéza tabulárních dat

3.6 Komprese

4. Experimenty s modelem variačního autoenkodéru

4.1 Generativní modelování obrazových dat

4.1.1 Vymezení problémové oblasti

4.1.2 Datová sada a předzpracování

4.1.3 Nastavení experimentu

4.1.4 Návrh modelu

4.1.5 Evaluace

4.1.6 Diskuze

4.2 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.

Přílohy

A. Zdrojové kódy modelů